

Computer-Verified Foundations of Metaphysics
and an Ontology of Natural Numbers in Isabelle/HOL

Dissertation
zur Erlangung des Grades eines
Doktors der Naturwissenschaften

am Fachbereich Mathematik und Informatik
der Freien Universität Berlin

vorgelegt von

Daniel Kirchner

Berlin, December 2021

Betreuer: **Prof. Dr. habil Christoph Benz Müller**
Zweitgutachter: **Dr. Edward N. Zalta**
Drittgutachter: **Prof. DDr. Hannes Leitgeb**

Datum der Disputation: **16. Mai 2022**

Selbstständigkeitserklärung

Ich erkläre gegenüber der Freien Universität Berlin, dass ich die vorliegende Dissertation selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe. Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht. Diese Dissertation wurde in gleicher oder ähnlicher Form noch in keinem früheren Promotionsverfahren eingereicht.

Mit einer Prüfung meiner Arbeit durch ein Plagiatsprüfungsprogramm erkläre ich mich einverstanden.

Abstract

We utilize and extend the method of *shallow semantic embeddings* (SSEs) in classical higher-order logic (HOL) to construct a custom theorem proving environment for *abstract objects theory* (AOT) on the basis of Isabelle/HOL.

SSEs are a means for universal logical reasoning by translating a target logic to HOL using a representation of its semantics. AOT is a foundational metaphysical theory, developed by Edward Zalta, that explains the objects presupposed by the sciences as *abstract objects* that reify property patterns. In particular, AOT aspires to provide a philosophically grounded basis for the construction and analysis of the objects of mathematics.

We can support this claim by verifying Uri Nodelman's and Edward Zalta's reconstruction of Frege's theorem: we can confirm that the Dedekind-Peano postulates for natural numbers are consistently derivable in AOT using Frege's method. Furthermore, we can suggest and discuss generalizations and variants of the construction and can thereby provide theoretical insights into, and contribute to the philosophical justification of, the construction.

In the process, we can demonstrate that our method allows for a nearly transparent exchange of results between traditional pen-and-paper-based reasoning and the computerized implementation, which in turn can retain the automation mechanisms available for Isabelle/HOL.

During our work, we could significantly contribute to the evolution of our target theory itself, while simultaneously solving the technical challenge of using an SSE to implement a theory that is based on logical foundations that significantly differ from the meta-logic HOL.

In general, our results demonstrate the fruitfulness of the practice of Computational Metaphysics, i.e. the application of computational methods to metaphysical questions and theories.

Contents

1. Introduction	1
1.1. Motivation	1
1.2. Prior Work	2
1.3. Contributions and Structure of the Thesis	5
1.4. Verified Document Generation and Conventions	6
2. Shallow Semantic Embeddings	7
2.1. Embeddings of Domain-Specific Languages	7
2.2. SSEs as Universal Reasoning Tools	8
2.3. SSE of Quantified Higher-Order Modal Logic	9
2.4. SSEs with Abstraction Layers	11
2.5. Isabelle’s Native Abstraction Mechanisms	13
2.6. Implicit Interpretation and Assignment Functions in SSEs	18
2.7. Reproducing the Syntax of the Target Theory	18
3. Abstract Object Theory	22
3.1. Overview	22
3.2. The Language	24
3.3. The Axiom System	27
3.4. The Deductive System	30
3.5. Interesting Theorems of AOT	36
3.6. Avoiding Known Paradoxes	39
3.7. Extending AOT’s Free Logic to Relations	41
3.8. Further Properties of AOT	43
4. SSE of AOT in Isabelle/HOL	48
4.1. Model Construction	48
4.2. Syntax of the Target Theory	57
4.3. Extending Isabelle’s Outer Syntax	59
4.4. Representation of an Abstract Semantics of AOT	61
4.5. Specifications and the Hilbert-Epsilon-Operator	62
4.6. Axiom System and Deductive System	63
4.7. Meta Theorems	70
4.8. Artifactual Theorems	79
4.9. Discussion	83

5. Natural Numbers in AOT	84
5.1. General Idea of the Construction	85
5.2. Equinumerosity of Relations	85
5.3. The Number of Fs and Hume’s Theorem	90
5.4. The Number Zero	91
5.5. Counting in Possible Worlds	92
5.6. Ancestral Relations and Transitive Closures	93
5.7. Weak Ancestral Relations	95
5.8. Generalized Induction	97
5.9. The Predecessor Relation	98
5.10. Natural Numbers	101
5.11. Zero is a Natural Number	101
5.12. Being a Natural Number is Rigid	102
5.13. Zero Has No Predecessor	102
5.14. No Two Natural Numbers have the Same Successor	102
5.15. Mathematical Induction	103
5.16. Properties of the Predecessor Relation and Natural Numbers	103
5.17. Possible Richness of Objects	104
5.18. Every Number has a Unique Successor	105
5.19. The Predecessor Axiom in Detail	106
5.20. Modelling Possible Richness of Objects	112
5.21. Prospect of an Enhanced Version of the Construction	113
5.22. Summary	115
6. Higher-Order Object Theory	116
6.1. Overview of Higher-Order Object Theory	116
6.2. Applications to Theoretical Mathematics	117
6.3. Bounded Models	118
6.4. Abstract Objects in Unbounded Models	119
7. Conclusion	121
A. Isabelle Theory	123
A.1. Model for the Logic of AOT	123
A.2. Outer Syntax Commands	144
A.3. Approximation of the Syntax of PLM	145
A.4. Semantics	156
A.5. Definitions of AOT	185
A.6. Axioms of AOT	188
A.7. The Deductive System PLM	193
A.8. Basic Logical Objects	338
A.9. Restricted Variables	349
A.10. Extended Relation Comprehension	356
A.11. Possible Worlds	368

A.12.Natural Numbers	419
A.13.Additional Theorems	507
References	521

1. Introduction

1.1. Motivation

The analysis of foundational formal systems using automated theorem provers is as old as automated theorem provers themselves: Already in the middle of the last century, Bertrand Russell was quick to recognize the potential of computational methods, when confronted with the *Logic Theorist*,¹ commonly regarded as the first automated theorem prover, and its ability to prove 38 out of 52 theorems from chapter two of Whitehead and Russell’s *Principia Mathematica*, including a proof more elegant than one of Whitehead and Russell’s own (see [23]):

I am delighted to know that *Principia Mathematica* can now be done by machinery. I wish Whitehead and I had known of this possibility before we both wasted ten years doing it by hand. I am quite willing to believe that everything in deductive logic can be done by a machine.²

However, building up a sound reasoning environment from scratch is a non-trivial task. Consequently, today there is only a limited number of trusted systems that can offer sophisticated interactive and automated reasoning tools like Coq [50], HOL-Light [24] or Isabelle/HOL [39]. Furthermore, most of these systems have at least parts of their logical foundation in common. For example, they are all based on some variation of functional type theory. This may lead to a bias in the computational analysis of foundational theories towards systems that use a similar logical foundation.

The following represents an attempt at overcoming this issue. We utilize the concept of a *shallow semantic embedding* (SSE) with abstraction layers to transfer the merits of the reasoning environment of Isabelle/HOL to a fundamentally different foundational system, namely to Abstract Object Theory (AOT).

While it is not a requirement for our proposed general method, we demonstrate that we can extend Isabelle/HOL by a customized reasoning infrastructure written in Isabelle/ML that allows for an almost entirely transparent transfer of reasoning in our target logic and abstracts the syntactic and inferential differences between Isabelle/HOL and AOT, while still internally using the verified core logic of Isabelle/HOL as semantic backend. This means we effectively construct a dedicated theorem proving environment for AOT that (1) is immediately guaranteed to be sound, (2) can be used to explore the safety of axiomatic extensions of the system and (3) allows for the reuse of the automation infrastructure available for Isabelle/HOL.

¹A system developed by Allen Newell and Herbert Simon at Carnegie Mellon and programmed by J. C. Shaw using the vacuum tubes of the JOHNNIAC computer at the Institute for Advanced Study.

²Letter from Russell to Simon dated 2 November, 1956; preserved in [49], page 208.

While our method can potentially be applied to a multitude of logical systems, AOT is a particularly well-suited target. On the one hand, it aims to be a foundational metaphysical system that can serve as the basis for mathematics and thereby stands in the tradition of Russell and Whitehead’s *Principia Mathematica*, while in fact extending its scope to e.g. linguistics and the sciences in general (see [58]). On the other hand, it is based on logical foundations that significantly differ from classical functional higher-order type-theory and were even argued to be incompatible (see [43]). Initial results of our research (see [29]) demonstrated how our method for formally analyzing models and semantics of such a system can be beneficial and vital for its soundness (see 3.6.2). During our continued work, we could contribute to the evolution of AOT and simultaneously arrived at a model structure and semantics that allows to faithfully reproduce its deductive system in Isabelle/HOL while retaining the existing infrastructure for automated reasoning.³

As a prime result, we can show that the construction of Natural Numbers and the derivation of the Dedekind-Peano postulates, including Mathematical Induction, described in *Principia Logico-Metaphysica* (PLM)⁴ are verifiably sound. Furthermore, we can suggest the generalization of an additional axiom required for this construction, that we believe strengthens the argument that the construction does not require any inherently mathematical axioms.

1.2. Prior Work

Since the time of Russell and the *Logic Theorist*, there has been significant progress both in the development of automated theorem provers in general and in the application of computational methods to metaphysical questions and foundational logical theories in particular. Some of the more recent developments in this area are outlined in the following sections.

1.2.1. Prior Computational Analysis of Abstract Object Theory

The computational analysis of AOT was pioneered by Fitelson and Zalta in [17]. They used the first-order system Prover9 (see [34]) for their work and were able to verify the proofs of the theorems in AOT’s analysis of situations and possible worlds in [65]. Furthermore, they discovered an error in a theorem about Platonic Forms in [46] that had been left as an exercise. Other work with Prover9 that does not target AOT includes the simplification of the reconstruction of Anselm’s ontological argument (in [41], Oppenheimer and Zalta show that only one of the three premises they used in [42] is sufficient) or the reconstruction of theorems in Spinoza’s *Ethics* in [26].

³Note, however, that our embedding currently only extends to the second-order fragment of AOT. We briefly discuss the challenges of representing full higher-order object theory in chapter 6.

⁴PLM is a continuously developed online monograph (see [62]) written by Edward Zalta, that contains the most recent canonical presentation of AOT. This thesis is written relative to the version dated October 13, 2021, archived in [63].

However, there are inherent limitations to the approach of analyzing higher-order theories like AOT with the help of first-order provers. While it is possible to reason about the first-order truth conditions of statements by introducing sort predicates and using a number of special techniques to translate the statements into the less-expressive language of multi-sorted first-order logic (a detailed account of such techniques is given in [1]), the complexity of the resulting representation increases for expressive, higher-order philosophical claims. In general, this approach may be sufficient for analyzing concrete isolated arguments, but it becomes infeasible to construct a natural representation of an entire expressive higher-order theory and its full deductive system (see also [30]).

1.2.2. Prior Work involving Shallow Semantic Embeddings

Independently, the emergence of sophisticated higher-order reasoning environments like Isabelle/HOL allows for a different approach, namely the analysis of arguments and theories directly in higher-order logic by constructing Shallow Semantic Embeddings (SSEs) (see [2]). In contrast to a *deep embedding* which defines the syntax of a target system using an inductive data structure and evaluates statements semantically by recursively traversing this data structure, a *shallow* semantic embedding instead provides a syntactic translation from the target logic to the meta-logic. This is done by reusing as much of the infrastructure of the meta-logic as possible, while *defining* the syntactic elements of the target logic that are not part of the meta-logic by means of a representation of their semantics. Since sets have a natural representation in higher-order logic, this approach works well for any logical system that has a semantics defined in terms of sets. The approach of shallow semantic embeddings is discussed in more detail in chapter 2.

For example, Benzmüller et al. provide an extensive analysis of quantified modal logic using SSEs by means of embedding modal operators based on their Kripke semantics [6, 3, 9]. This allowed for an analysis of Gödel’s ontological argument in second-order S5 modal logic and weaker logics such as KB (see [8, 4]), followed by a range of studies of similar ontological arguments (see e.g. [20]).

Another more recent example of the application of SSEs is the LogiKEy framework for ethical reasoning, normative theories and deontic logics (see [5] and [10]). The goal of LogiKEy is to develop the means for the control and governance of intelligent autonomous systems. The framework is based on a set of SSEs of different deontic logics, combinations thereof, as well as ethico-legal domain theories in higher-order logic with an implementation in Isabelle/HOL.

The advantage of these studies using SSEs compared to the earlier use of first-order systems is that arguments can be represented in their native syntax and are thereby readable and maintainable, while the theorem proving environment is capable of automatically transforming statements into a suitable first-order representation on the fly to allow first-order theorem provers like E (see [48]) or SPASS (see [51]) to perform proof search much like e.g. Prover9 was able to do on a manually constructed first-order representation.

These studies were still mainly concerned with case studies of concrete arguments or with conservative extensions of higher-order logic like quantified higher-order modal logic.

1.2.3. Analysis of AOT with the SSE Approach

Initial results of our own research were reported in [29], in which we applied an extended version of the technique of SSEs to AOT. For AOT no extensive prior analysis of canonical models was available, in contrast to, for example, the extensive analysis of Kripke models for higher-order modal logic that served as theoretical basis for the previous work using SSEs mentioned above. While the so-called Aczel models of object theory (see [60]) provide an important building block for constructing models of AOT in HOL, no full set-theoretic model of object theory had been constructed. In [29] we extended the existing Aczel models to a richer model structure that was capable of approximating the validity of statements of the at the time most recent formulation of the second-order fragment of AOT in *Principia Logico-Metaphysica*.⁵ Furthermore, we introduced the new concept of *abstraction layers*. An abstraction layer consists of a derivation of the axioms and deduction rules of a target system from a given semantics that is then considered as ground truth while "forgetting" the underlying semantic structure, i.e. the reasoning system is prevented from using the semantics for proofs, but is instead configured to solely rely on the derived axioms and deduction rules. Abstraction layers turned out to be a helpful means for reasoning within a target theory without the danger of deriving artifactual theorems (see 4.8), while simultaneously allowing to maintain a flexible semantic backend that can be used to explore axiomatic extensions and variations of the target theory.

A major initial result of this project, reported in [31], was the discovery of an oversight in an early version of PLM that allowed for the reintroduction of a previously known paradox into the system. While multiple quick fixes to restore the consistency of AOT were immediately available, in the aftermath of this result AOT was significantly reworked and improved. The result triggered an extensive debate about the foundations of AOT which culminated in the extension of the free logic of AOT to relations, while previously it was restricted to individual terms only. This evolution of AOT was accompanied by a continuous further development of its embedding in Isabelle/HOL. This mutually beneficial mode of work was described in [30] and resulted in a now stabilized and improved formulation of AOT and a matching embedding of its second-order fragment. The details of this process and its results are the main subject of this thesis.

⁵The respective version of PLM is archived in [64].

1.3. Contributions and Structure of the Thesis

In the following, we first provide a more detailed description of Shallow Semantic Embeddings (chapter 2) and a brief introduction to Abstract Object Theory (chapter 3). Based on that, chapter 4 describes the constructed embedding of the second-order fragment of AOT (as presented in PLM [63]) in Isabelle/HOL.

In the process we highlight the contributions of the embedding to AOT on the one hand and the techniques developed for its implementation on the other hand.

In chapter 5 we present our results on PLM's construction of natural numbers and discuss an extension of AOT with a more general comprehension principle for relations among abstract objects. We also discuss some interesting variations of the construction that may be adopted by PLM in the future.

Finally, in chapter 6 we briefly discuss the issue of applying our method to the full higher-order type-theoretic version of AOT.

Our primary goals are to show that:

- SSEs can not only be used for case studies and the analysis of isolated arguments, but also for implementing the axioms and full deductive system of entire logical theories.
- The above is even feasible for a challenging target like AOT, which itself has the ambition to be a foundational framework and is based on significantly different logical foundations compared to our meta-logic HOL.
- We can reproduce the full deductive system of AOT in readable and usable form while preserving Isabelle's automation mechanisms. Thereby, we can effectively construct a dedicated automated theorem proving environment for AOT.
- Using our method we could significantly contribute to our target theory.
- We can demonstrate the extent of our target theory and the practical feasibility of reproducing complex reasoning in it by reproducing and validating its analysis of natural numbers.
- In the process, we can provide valuable theoretical insights into, and analyze extensions and variations of, the construction of the natural numbers.

1.4. Verified Document Generation and Conventions

This thesis is generated using Isabelle’s document preparation system (see [53]). In particular, all formal statements cited in the thesis are renderings of verified theorems in the embedding, unless specifically stated otherwise and marked with vertical bars at the page margins.⁶

The appendix contains a rendering of the raw theory files of the embedding including all proofs.⁷ The implementation currently consists of around 25,000 lines of Isabelle proof text.⁸ While Isabelle allows producing latex code for raw theories directly,⁹ semantic information (e.g. color-coding of free vs. bound variables) is lost in the process, which reduces the readability. For that reason, we devised a custom theory presentation system similar to Isabelle’s HTML theory presentation that uses PIDE markup information (see [52]) to provide a color-coded rendering of the theory files equipped with hyperlinks for cross-references.¹⁰

Whenever a theorem in the appendix refers to a specific item number in PLM, the corresponding item number can be found in parentheses at the right page margin. While we will sometimes refer to item numbers in PLM directly, we will usually refer to the implementation in the appendix by section and line number and rely on the statement in the appendix being annotated with the item number of the corresponding statement in PLM. In particular, the thesis is written relative to the version of PLM dated October 13, 2021 (see [63]).

While a certain degree of familiarity with the reasoning environment of Isabelle/HOL might be helpful, the fact that reasoning in Isabelle/HOL is designed to be natural and intelligible should allow following the constructions without extensive prior knowledge of Isabelle/HOL. An introduction to reasoning in Isabelle/HOL can be found in [38]. The implementation is written relative to the Isabelle2021-1 (December 2021) release of Isabelle.

⁶With the exception of chapter 6 which is not written relative to an embedding in Isabelle and omits the marking at the page margins.

⁷The corresponding theory files can also be found at [27].

⁸Around 20,000 lines are reasoning in the abstraction layer, i.e. reasoning in the logic of the target theory, while the remainder builds up the required model structure and semantics as well as the syntax representation of AOT.

⁹This mechanism is used for raw theory content that is inlined in the main thesis, but not for the appendix.

¹⁰Therefore, we recommend reading this thesis in digital form.

2. Shallow Semantic Embeddings

2.1. Embeddings of Domain-Specific Languages

In computer science, deep and shallow embeddings have been a traditional means to implement domain-specific languages by embedding them into general-purpose host languages (see for example [22]). A simple example is a language of *expressions* that can be either integer constants, resp. literals, or the addition of two other expressions. If we consider Isabelle/HOL as the host language in this process, the following would constitute a *deep* embedding of this language:

```
datatype expression = Literal int | Addition expression expression
primrec eval :: ⟨expression ⇒ int⟩ where
  ⟨eval (Literal x) = x⟩
  | ⟨eval (Addition x y) = eval x + eval y⟩
```

The deep embedding consists of a (usually recursive) algebraic datatype that captures the syntactic elements of the language to be embedded. This representation of the syntax is then given a semantics by means of an evaluation function that traverses this algebraic datatype.¹ A shallow embedding on the other hand, represents the syntactic elements of a target language directly in their semantic domain. In our example, the semantic domain of expressions is the integers. On this domain, operations are then *defined* directly by means of their semantics:

```
type-synonym expression = int
definition Literal :: ⟨int ⇒ expression⟩ where ⟨Literal x ≡ x⟩
definition Addition :: ⟨expression ⇒ expression ⇒ expression⟩ where ⟨Addition x y ≡ x + y⟩
```

Note that in the shallow embedding, the domain of *expressions* is shared with the meta-language by directly representing expressions in the type to which they evaluate semantically in the deep embedding, namely *int* in the example.

There is a natural correspondence between the deep and shallow representations of this language. In particular it holds that $Deep.eval (Deep.Literal x) = Shallow.Literal x$ and $Deep.eval (Deep.Addition x y) = Shallow.Addition (Deep.eval x) (Deep.eval y)$. So semantic

¹In the setting of logical theories this evaluation function would usually depend on interpretations and assignment functions. However, in our simple example this is not necessary, since the simple language of expressions neither involves constants nor variables (respectively since literals have trivial interpretations).

evaluation is implicit in the shallow embedding. On the other hand there are also differences between the two representations. For example, in the deep embedding adding x to y results in an expression that is different from the expression of adding y to x for distinct x and y , even though they are equivalent under evaluation:

$$\begin{aligned}x \neq y &\implies \text{Deep.Addition } x \ y \neq \text{Deep.Addition } y \ x \\ \text{Deep.eval } (\text{Deep.Addition } x \ y) &= \text{Deep.eval } (\text{Deep.Addition } y \ x)\end{aligned}$$

In contrast, commuted additions are identical in the shallow embedding:

$$\text{Shallow.Addition } x \ y = \text{Shallow.Addition } y \ x$$

In fact, the shallow embedding can be thought of as a *quotient* of the deep embedding under semantic evaluation.

While there are several advantages and disadvantages of using shallow vs. deep embeddings for Domain-Specific languages, we forgo a detailed discussion of them here and focus on shallow embeddings of logical theories in the next sections.

2.2. SSEs as Universal Reasoning Tools

In [2], Benzmüller develops the idea of using *Shallow Semantic Embeddings* (SSEs) in classical higher-order logics (HOL) as a means for universal reasoning.

He notes that while already Leibniz envisioned a *characteristica universalis*, a most universal formal language in which all knowledge (and all arguments) about the world and the sciences can be encoded, in practice, today we rather find a *rich and heterogeneous zoo of different logical systems*.

A solution to this dilemma is the use of a universal *meta*-logic, in which a multitude of logic formalisms can be *embedded*.

While there are multiple such unifying approaches, for example using algebraic logic or category theory as framework, Benzmüller defends the use of SSEs in HOL for pragmatic reasons:

- For HOL there are sophisticated automation tools readily available that have been developed for several decades like e.g. Isabelle/HOL.
- Since HOL itself is very expressive, an embedding into HOL can often be achieved using simple techniques and can result in an elegant and concise representation of the target logic.
- Using a *shallow* embedding approach, the technical overhead of the translation can be kept minimal, which enables the reuse of the automation infrastructure available for the meta-logic.

While we already mentioned a variety of results that were achieved using this general method (see section 1.2.2), in the following we will demonstrate the process of building such an SSE at a simple example.

2.3. SSE of Quantified Higher-Order Modal Logic

An example of a non-classical logic that is used prominently in philosophical arguments is Quantified Higher-Order Modal Logic in various different axiomatizations. While there have been extensive studies of modal logics using SSEs in Isabelle/HOL (see section 1.2.2), we restrict ourselves to the discussion of a simple embedding of S5 modal logic to further illustrate the general concept of SSEs.

A natural semantic basis for SSEs of any modal logic is its Kripke-semantics (see [32]). In general, a Kripke frame consists of a set of possible worlds and a binary relation on these worlds called *accessibility relation*. For S5 there are two versions of semantics, one in which the accessibility relation is an equivalence relation and one in which there is no accessibility relation at all (see [18]). For our purpose the simpler model suffices.²

For possible worlds we can introduce a primitive type w in Isabelle/HOL.³

typedecl w

A Kripke model further involves a relation between possible worlds and modal formulas that is usually read as a formula *being satisfied at* a possible world. So the semantic domain of propositions is the boolean-valued functions acting on (or, equivalently, the sets of) possible worlds. In an SSE we use the semantic domains as types for the object-level terms themselves,⁴ so we can introduce a type o of propositions as synonym of the type of functions mapping possible worlds (of type w) to booleans (type $bool$). This way the proposition can, as a function, be applied to a possible world, yielding *True*, if the proposition is satisfied at that world or *False* otherwise.⁵

type-synonym $o = \langle w \Rightarrow bool \rangle$

A proposition is *valid* in case it is satisfied in all worlds (or, alternatively, in a designated actual world).⁶

definition *valid* :: $\langle o \Rightarrow bool \rangle \langle \models \rightarrow 100 \rangle$ **where**

$\langle \models p \equiv \forall w . p w \rangle$

Now the classical logical operators can be defined as follows (note the bold print for the defined operators versus the non-bold print of the corresponding operators of the meta-logic):

²We will later argue that this is also a natural choice for the particular modal logic of Abstract Object Theory due to its additional actuality operator and rigid definite descriptions, see section 4.7.4.

³A set-theoretic model of HOL would represent this type with a non-empty set of objects that may serve as denotation for objects of type w .

⁴Note that it is also possible to model restrictions on the evaluation domains explicitly, as recently demonstrated in [7].

⁵Note that this choice of a representation of propositions commits us to a modal logic, in which necessary equivalence implies identity. We will later discuss how we can construct a hyperintensional logic instead.

⁶The specification in parentheses after the type of the defined constant, $o \Rightarrow bool$, is *mixfix notation* used to introduce the symbol \models as syntax for the introduced constant *valid* with the specified precedence. The means to introduce custom syntax in Isabelle/HOL are discussed in more detail in section 2.7.

definition *not* :: $\langle \circ \Rightarrow \circ \rangle$ ($\langle \neg \rightarrow \rangle$ [140] 140) **where**

$\langle \neg p \equiv \lambda w . \neg p w \rangle$

definition *imp* :: $\langle \circ \Rightarrow \circ \Rightarrow \circ \rangle$ (**infixl** $\langle \rightarrow \rangle$ 125) **where**

$\langle p \rightarrow q \equiv \lambda w . p w \rightarrow q w \rangle$

definition *conj* :: $\langle \circ \Rightarrow \circ \Rightarrow \circ \rangle$ (**infixl** $\langle \wedge \rangle$ 135) **where**

$\langle p \wedge q \equiv \lambda w . p w \wedge q w \rangle$

definition *disj* :: $\langle \circ \Rightarrow \circ \Rightarrow \circ \rangle$ (**infixl** $\langle \vee \rangle$ 130) **where**

$\langle p \vee q \equiv \lambda w . p w \vee q w \rangle$

The additional modal operators, i.e. the box operator for *necessity* and the diamond operator for *possibility*, can be further defined as:

definition *box* :: $\langle \circ \Rightarrow \circ \rangle$ ($\langle \Box \rightarrow \rangle$ [150] 150) **where**

$\langle \Box p \equiv \lambda w . \forall v . p v \rangle$

definition *dia* :: $\langle \circ \Rightarrow \circ \rangle$ ($\langle \Diamond \rightarrow \rangle$ [150] 150) **where**

$\langle \Diamond p \equiv \lambda w . \exists v . p v \rangle$

Now Isabelle can show automatically that the S5 axioms are valid:

lemma *K*: $\langle \models \Box(p \rightarrow q) \rightarrow (\Box p \rightarrow \Box q) \rangle$

by (*auto simp: box-def imp-def valid-def*)

lemma *T*: $\langle \models \Box p \rightarrow p \rangle$

by (*auto simp: box-def imp-def valid-def*)

lemma *5*: $\langle \models \Diamond p \rightarrow \Box \Diamond p \rangle$

by (*auto simp: box-def dia-def imp-def valid-def*)

The proofs of the axioms are automatically found by **sledgehammer**, Isabelle/HOL's main tool for automation.⁷

So far we have constructed an embedding of propositional S5 modal logic using what is commonly known as *Standard Translation* of modal logic (see [11]). However it is straightforward to enrich this embedding with quantification.⁸

definition *forall* :: $\langle ('a \Rightarrow \circ) \Rightarrow \circ \rangle$ (**binder** $\langle \forall \rangle$ 110) **where**

$\langle \forall x . \varphi x \equiv \lambda w . \forall x . \varphi x w \rangle$

definition *exists* :: $\langle ('a \Rightarrow \circ) \Rightarrow \circ \rangle$ (**binder** $\langle \exists \rangle$ 110) **where**

$\langle \exists x . \varphi x \equiv \lambda w . \exists x . \varphi x w \rangle$

Note that we didn't introduce any particular type for individuals, but stated polymorphic definitions relative to a type variable *'a*. This way the same quantifier can be used for propositions themselves, any desired type for individuals or even properties of any order.⁹

As an example of theorems involving quantifiers and modal logic, we derive the Barcan formulas. **sledgehammer** can again automatically provide proofs.

lemma $\langle \models (\forall x . \Box \varphi x) \rightarrow \Box (\forall x . \varphi x) \rangle$

by (*auto simp: box-def forall-def imp-def valid-def*)

⁷**sledgehammer** is discussed in more detail in the following section.

⁸See also the work by Benz Müller et al. cited in section 1.2.2.

⁹Note that this construction implies a shared domains for objects across possible worlds. An additional meta-logical predicate for *logical existence in a possible world* can be added to model varying domains.

```

lemma <|=  $\Diamond(\exists x . \varphi x) \rightarrow (\exists x . \Diamond\varphi x)$ >
  by (auto simp: dia-def exists-def imp-def valid-def)
lemma <|=  $\Box(\forall x . \varphi x) \rightarrow (\forall x . \Box\varphi x)$ >
  by (auto simp: box-def forall-def imp-def valid-def)
lemma <|=  $(\exists x . \Diamond\varphi x) \rightarrow \Diamond(\exists x . \varphi x)$ >
  by (auto simp: dia-def exists-def imp-def valid-def)

```

However, note that the automatic proofs again unfold the semantic definitions. We have shown that the Barcan formulas are valid in the constructed embedding, but from the proofs we cannot tell which axioms are required for proving them.¹⁰

Depending on the application, it can be enough to be able to tell if a theorem is semantically valid or if a statement semantically follows from a set of assumptions. However, for the purpose of implementing a full logical theory including its own deductive system, semantic validity is not the primary concern, but rather derivability from the formal system.¹¹

Fortunately, it is possible to restrict Isabelle’s automated reasoning tools like **sledgehammer**, s.t. they may not unfold semantic definitions. If this is done at larger scale and in a reliable manner for the purpose of analyzing derivability in a given deductive system, we say that we introduce *abstraction layers* to the SSE.

2.4. SSEs with Abstraction Layers

The concept of enriching traditional SSEs with abstraction layers was first introduced in [29]. The goal is to be able to use the automated reasoning tools provided by a system like Isabelle/HOL not merely to analyze semantic validity of statements in the embedded theory, but to reliably determine the derivability of a statement from the deductive system of the theory itself.

An abstraction layer is simply constructed by proving that the axioms and deduction rules of a target logic are semantically valid in the embedding, after which they are considered as ground truths: all subsequent reasoning in the abstraction layer is restricted to only rely on the derived axioms and rules and may no longer refer to the underlying semantics. Consequently, only theorems derivable in the target logic are derivable in the abstraction layer.¹²

So while abstraction layers are conceptually rather simple, an interesting technical question is how the automation capabilities of the meta-logic can be preserved and reliably restricted to respect the imposed restrictions.

¹⁰As a matter of fact we did not even state any axioms governing implications or quantifiers in the embedded logic.

¹¹Even if the target theory is provably complete with respect to the semantics used for constructing the embedding, i.e. semantic validity implies derivability, we still want to know which axioms and rules can be used to construct a concrete derivation.

¹²Note, however, that this relies on the additional assumption that meta-logical inferences based on the derived axioms and rules correspond to derivations in the target logic, as mentioned in the end of this section.

While Isabelle provides its own mechanisms for abstract reasoning like type **classes**, **locales** and **specifications**, those are not primarily designed for this exact purpose and come with limitations that can make them unsuitable to achieve that purpose on their own, as described in more detail in the following section.

As mentioned in the last section, the main tool for automated reasoning in Isabelle/HOL is **sledgehammer** (see [44]). **sledgehammer** can be invoked during any proof and will try to automatically find a proof for the current proof goal. To that end, simply speaking,¹³ it collects all theorems and definitions derived in the current **theory** context together with all local assumptions (collectively referred to as *facts*) and processes the resulting set of facts heuristically to find a subset of relevant facts. It then encodes the problem of deriving the current goal from the chosen facts in a format that can be consumed by external theorem provers like E [48], SPASS [51], verit [15] or Z3 [37]. This may, for example, involve a translation from higher-order problems to first-order problems. If one of the invoked provers can prove the current goal, **sledgehammer** tries to reconstruct a short proof using Isabelle’s native proving methods (which operate directly on Isabelle’s trusted reasoning core) that can be directly inserted to prove the current goal.¹⁴

The relevant part of the process to consider for the purpose of constructing an abstraction layer is the initial selection of facts from the theory context. We do not want **sledgehammer** to use the equational theorems that unfold our semantic definitions, but instead derive the goals from only the axioms and specific derivational rules that correspond to the rules of the deductive system of the embedded theory. **sledgehammer** allows us to provide some guidance in its choice. It is possible to (1) indicate that a certain set of facts is likely to be helpful in the proof (using *add:*), (2) prevent it from using certain facts (either using *del:* or by marking facts with the special attribute *no-atp*) or (3) to provide it with a specific set of facts to use directly without taking any other facts into account.

Conceptually, option (3) is the best fit for the purpose of abstraction layers and was used in [29]. However, **sledgehammer** will no longer employ its heuristics and machine learning algorithms to filter the provided facts for relevance, but will directly use the provided set. Consequently, the proving power and therefore the usefulness of **sledgehammer** is significantly diminished, especially for larger theories.

In our current implementation, we therefore use option (2) instead. However, this comes with some challenges. While the equational theorems introduced by simple **definitions** can easily be collected and marked, other more advanced constructions in Isabelle like type definitions or **lift-definitions** (see [25]) introduce several theorems implicitly. While it is still possible to collect these theorems manually, the process is cumbersome and error-prone.

¹³For a precise description of the full details of the process refer to [44].

¹⁴Furthermore, for provers like veriT and Z3, *proof reconstruction* using the *smt* tactic is available, i.e. they provide proofs that can (sometimes) be directly replayed relative to Isabelle’s trusted reasoning core. See [19] and [13].

On the other hand, it is not possible to simply exclude *all* theorems that were derived up to a certain point, since this includes the theorems of Isabelle’s *Main* theory, i.e. - among others - the construction of classical higher-order logic on top of Isabelle’s more basic *Pure* logic. This includes theorems **sledgehammer** relies on and disbaring them will leave it non-functional (conceptually, such theorems can be thought of as meta-theorems about derivations in our context).

The solution used in the current embedding of AOT is the use of Isabelle’s internal ML API to automatically collect theorems to be added to an exclusion list. For convenience, a new command **AOT-sledgehammer** is introduced that internally configures **sledgehammer** to use the desired set of theorems and then passes the current proof state to it.¹⁵ With this method we can achieve significantly better proof automation than [29].

It is important to note that abstraction layers still rely on the implicit assumption that meta-logical reasoning about derivations in the target logic is faithfully represented by the meta-logical inferences in Isabelle enabled by the constructed deduction rules.

In particular, the deductive system of our target theory is implemented as meta-rules in Isabelle’s *Pure* logic, while the used automation mechanisms additionally rely on the logic of Isabelle/HOL. Consequently, we need to convince ourselves that resulting inferences are reproducible in the target system and, conversely, that derivations in our target system are exhaustively captured by the rules of our abstraction layer. For our embedding of AOT we sketch such an argument in section 4.7.5.

2.5. Isabelle’s Native Abstraction Mechanisms

While abstraction layers provide a means to insulate reasoning in our embedding from artifactual theorems (i.e. theorems that are merely semantically valid but not derivable in the target theory; see also 4.8), we additionally use Isabelle’s native abstraction mechanisms. This serves to establish an additional intermediate abstraction between the concrete model construction and the derivation of the axioms and deductive system of the target theory, which helps in exploring changes to the model structure without having to adjust the full derivation of the abstraction layer.

2.5.1. Specifications

For example, we extensively use **specifications** (see §11.4 in [55]). A **specification** is used to assert statements about previously uninterpreted constants. The **specification** command opens a proof context that requires the user to show that there exists a concrete instantiation for the given constants, for which the desired statements hold. Internally it then uses Isabelle’s Hilbert-Epsilon-operator *SOME* $x. \varphi x$ to augment the given constants with a concrete definition. We will discuss the technical details of this mechanism in section 4.5. As a consequence, a model of the meta-logic may choose any denotation

¹⁵Alternatively, we allow configuring **sledgehammer** itself to only use the restricted set of theorems.

for the given constants that satisfies the specification, while the existence of such a denotation is guaranteed by the provided witness. However, depending on the use case of this mechanism, care has to be taken to ensure that there actually are non-trivial choices beyond the provided witness.

To illustrate this issue, we showcase the construction of a (hyper-)intensional conjunction in which $p \wedge q$ implies both p and q and vice-versa, but it does not hold that $(p \wedge q) = (q \wedge p)$. We first show a construction that will fail due to the choice of a representation type that implies extensionality:

typedef $o_1 = \langle UNIV::bool\ set \rangle..$ — Introduce an abstract type of propositions o_1 with the universal set of booleans (i.e. $\{True, False\}$) as representation set.¹⁶

definition $valid\text{-}o_1 :: \langle o_1 \Rightarrow bool \rangle$ **where**
 $\langle valid\text{-}o_1\ p \equiv Rep\text{-}o_1\ p \rangle$ — Validity is simply given by the boolean representing the proposition.¹⁷

We introduce an uninterpreted constant for conjunctions with infix syntax.

consts $o_1\text{-}conj :: \langle o_1 \Rightarrow o_1 \Rightarrow o_1 \rangle$ (**infixl** $\langle \wedge \rangle$ 100)

specification ($o_1\text{-}conj$) — We specify our conjunction by introduction and elimination rules.

$o_1\text{-}conjE1: \langle valid\text{-}o_1\ (p \wedge q) \Longrightarrow valid\text{-}o_1\ p \rangle$
 $o_1\text{-}conjE2: \langle valid\text{-}o_1\ (p \wedge q) \Longrightarrow valid\text{-}o_1\ q \rangle$
 $o_1\text{-}conjI: \langle valid\text{-}o_1\ p \Longrightarrow valid\text{-}o_1\ q \Longrightarrow valid\text{-}o_1\ (p \wedge q) \rangle$

We need to prove that there is a term satisfying the above specification. The natural choice is the lifted conjunction on the booleans.

by ($rule\ ext[\mathbf{where}\ x = \langle \lambda\ p\ q.\ Abs\text{-}o_1\ (Rep\text{-}o_1\ p \wedge Rep\text{-}o_1\ q) \rangle]$)
 $(auto\ simp: Abs\text{-}o_1\text{-}inverse\ valid\text{-}o_1\text{-}def)$

However, even though the identity of commuted conjunctions is not part of the **specification**, it is *still* derivable.¹⁸

lemma $\langle p \wedge q = q \wedge p \rangle$

by ($metis\ Rep\text{-}o_1\text{-}inject\ o_1\text{-}conjE1\ o_1\text{-}conjE2\ valid\text{-}o_1\text{-}def$)

The reason is that there is only one choice for a conjunction operator on the booleans that satisfies our specification and this choice is commutative. We can in fact prove that our conjunction has to be identical to the witness we provided:

lemma $\langle (\wedge) = (\lambda\ p\ q.\ Abs\text{-}o_1\ (Rep\text{-}o_1\ p \wedge Rep\text{-}o_1\ q)) \rangle$

by ($metis\ Abs\text{-}o_1\text{-}inject\ Abs\text{-}o_1\text{-}inverse\ UNIV\text{-}I\ o_1\text{-}conjE1\ o_1\text{-}conjE2\ o_1\text{-}conjI$
 $type\text{-}copy\text{-}obj\text{-}one\text{-}point\text{-}absE\ type\text{-}definition\text{-}o_1\ valid\text{-}o_1\text{-}def$)

¹⁶For every type definition using an explicit representation set (**typedef**), we need to prove that the set is non-empty. In the case of the universal set of another type, this proof is trivial, as indicated by the two dots.

¹⁷For any **typedef**, Isabelle introduces constants prefixed with *Abs-* and *Rep-*, mapping the representation type to the defined abstract type and vice-versa.

¹⁸Note that if we constructed abstraction layers as discussed in the last section, **sledgehammer** would be prevented from considering the implicit theorems introduced by the type definition of o_1 (which relate the type to its representation type) and, therefore, would not be able to prove this theorem.

In order to avoid this issue, we cannot simply rely on the **specification** command, but also have to take care that the *types* of the specified constants can actually deliver the desired degree of intensionality. In our example, we can introduce an abstract *intensional type* for propositions that merely has a boolean *extension*. First we introduce an abstract type:

typedecl o_2 — Introduce an abstract type for propositions.

Thus far, a model of HOL satisfying our theory may choose any non-empty set as representation set for objects of type o_2 . To arrive at a meaningful type of propositions, we axiomatically introduce a surjective extension function mapping the abstract propositions to their boolean extension. The surjectivity of the extension function excludes degenerate models in which there is only one proposition.¹⁹

axiomatization $\mathsf{o}_2\text{-ext} :: \langle \mathsf{o}_2 \Rightarrow \mathit{bool} \rangle$ **where**

$\mathsf{o}_2\text{-ext-surj}: \langle \mathit{surj} \ \mathsf{o}_2\text{-ext} \rangle$

definition $\mathit{valid}\text{-}\mathsf{o}_2 :: \langle \mathsf{o}_2 \Rightarrow \mathit{bool} \rangle$ **where**

$\langle \mathit{valid}\text{-}\mathsf{o}_2 \ p \equiv \mathsf{o}_2\text{-ext} \ p \rangle$ — Validity of a proposition is given by its boolean extension.

consts $\mathsf{o}_2\text{-conj} :: \langle \mathsf{o}_2 \Rightarrow \mathsf{o}_2 \Rightarrow \mathsf{o}_2 \rangle$ (**infixl** $\langle \wedge \rangle$ 100)

specification ($\mathsf{o}_2\text{-conj}$) — We again specify our conjunction by introduction and elimination rules.

$\mathsf{o}_2\text{-conjE1}: \langle \mathit{valid}\text{-}\mathsf{o}_2 \ (p \wedge q) \Longrightarrow \mathit{valid}\text{-}\mathsf{o}_2 \ p \rangle$

$\mathsf{o}_2\text{-conjE2}: \langle \mathit{valid}\text{-}\mathsf{o}_2 \ (p \wedge q) \Longrightarrow \mathit{valid}\text{-}\mathsf{o}_2 \ q \rangle$

$\mathsf{o}_2\text{-conjI}: \langle \mathit{valid}\text{-}\mathsf{o}_2 \ p \Longrightarrow \mathit{valid}\text{-}\mathsf{o}_2 \ q \Longrightarrow \mathit{valid}\text{-}\mathsf{o}_2 \ (p \wedge q) \rangle$

We again need to prove the existence of a term satisfying the given specification. Since our extension function is surjective, a natural suitable witness can be constructed using the inverse of the extension function.

by (*rule* $\mathit{exI}[\mathbf{where} \ x = \langle \lambda \ p \ q . (\mathit{inv} \ \mathsf{o}_2\text{-ext}) \ (\mathsf{o}_2\text{-ext} \ p \wedge \mathsf{o}_2\text{-ext} \ q) \rangle]$)
 (*simp add:* $\mathsf{o}_2\text{-ext-surj} \ \mathit{f-inv-into-f} \ \mathit{valid}\text{-}\mathsf{o}_2\text{-def}$)

Now as a consequence of our specification, our conjunction is still commutative *under validity*:

lemma $\langle \mathit{valid}\text{-}\mathsf{o}_2 \ (p \wedge q) = \mathit{valid}\text{-}\mathsf{o}_2 \ (q \wedge p) \rangle$

Note that the proof (found by **sledgehammer**) now solely relies on the properties of (\wedge) given in our specification:

¹⁹Note, that we can also construct an equivalent type without a meta-logical axiom: we can (1) introduce an uninterpreted constant that defines a set of products (or, alternatively, sums) of an additional uninterpreted type of intensions and the type of extensions (*bool* in the example), (2) specify that this set is both non-empty and large enough to allow for a surjective function to the extensions (the universal set of such products will be a witness for this specification) and (3) use this set as representation set for our intensional type. The existence of a surjective extension function will become derivable from the specification. However, we found that the model-finding tool **nitpick** works better with the equivalent axiomatic introduction of an extension function on an abstract type.

using $o_2\text{-conjE1}$ $o_2\text{-conjE2}$ $o_2\text{-conjI}$ **by** *blast*

However, commuted conjunctions are no longer identical. The model-finding tool **nitpick** (see [12]) can provide a counterexample by constructing a model for o_2 that has more than two members.

lemma $\langle (p \wedge q) = (q \wedge p) \rangle$

nitpick[*user-axioms, expect = genuine, show-consts, atoms* $o_2 = p\ q\ r$, *format = 2*]

oops — Abort proof and discard the lemma.

The model given by **nitpick**²⁰ chooses a three-element set for type o_2 . We chose p , q and r as names for these elements. $o_2\text{-ext}$ is modelled as $(p := \text{True}, q := \text{False}, r := \text{False})$ and (\wedge) as $((p, p) := p, (p, q) := q, (p, r) := r, (q, p) := r, (q, q) := q, (q, r) := r, (r, p) := r, (r, q) := r, (r, r) := r)$.

This is indeed one of the minimal (non-degenerate)²¹ models for conjunctions that are classical under validity, but are not identical under commutation. On the other hand, **nitpick** can also *satisfy* the same statement by providing a model with cardinality 2 for type o_2 :

lemma $\langle (p \wedge q) = (q \wedge p) \rangle$

nitpick[*satisfy, user-axioms, expect = genuine, show-consts, atoms* $o_2 = p\ q$, *format = 2*]

oops

Note that for the above it is sufficient to find a concrete choice for p and q , s.t. the identity holds for those two propositions. However, **nitpick** can also produce (in this case the same) model satisfying the identity for all propositions, respectively - equivalently - refute the identity failing to hold:

lemma $\langle \forall p\ q. (p \wedge q) = (q \wedge p) \rangle$ — Satisfy the identity for all p and q .

nitpick[*satisfy, user-axioms, expect = genuine, show-consts, atoms* $o_2 = p\ q$, *format = 2*]

oops

lemma $\langle (p \wedge q) \neq (q \wedge p) \rangle$ — Refute non-identity for arbitrary p and q .

nitpick[*user-axioms, expect = genuine, show-consts, atoms* $o_2 = p\ q$, *format = 2*]

oops

While the above describes a general mechanism that (given a careful choice of types) can be used to force Isabelle to rely on a specific set of specified properties for constants while simultaneously retaining assured consistency,²² the mechanism has limitations.

For instance, **specifications** are limited in their capability to specify polymorphic constants. While it is both possible to provide a shared specification for all types of a

²⁰The precise model may vary for different versions of Isabelle.

²¹The specification for conjunctions alone can also be satisfied in degenerate models, in which either all propositions are true or all propositions are false, i.e. in particular for models with only one proposition. However, we excluded such degenerate models by requiring a surjective extension function, which prevents **nitpick** from considering these degenerate cases.

²²The specification part is guaranteed to be consistent, since we provided an explicit witness in the process; the consistency of the axiom assuring the surjectivity of the extension function is confirmed by **nitpick**.

polymorphic constant, as well as to provide separate specifications for concrete distinct type instantiations of a polymorphic constant, doing both at the same time is in general not possible.

2.5.2. Type Classes and Locales

Isabelle provides further abstraction mechanisms, e.g. type **classes** and **locales**, but each comes with its own limitations. Technically, a **locale** (see §5.7 in [55]) is a functor that maps parameters and a specification to a list of declarations. In practice, this can be used to reason relative to abstract parameters that validate a set of assumptions and then **interpret** the **locale** by proving the assumptions for a concrete instantiation of its parameters. As a result of this interpretation of the locale, all declarations of, and in particular all theorems proven in, the locale will be instantiated to the given parameters and added to the theory context. A limitation of **locales** is that they cannot involve polymorphic assumptions, which prevents us from formulating the full system of AOT abstractly as a single locale.

Type **classes** (see §5.8 in [55]) are technically **locales** that depend on *exactly one* type variable and additionally introduce an axiomatic type class for which, roughly speaking, the parameters of the locale are introduced as global constants that satisfy the locale assumptions. In practice, type classes can be used to define properties on types and reason about any type with those properties. Type classes can then be *instantiated* for a concrete type²³ by proving that the assumptions are satisfied for a concrete definition of the locale parameters at that type.

For example, it is possible to instantiate a type class to products of two generic types (i.e. type variables) of specific sorts. We use this mechanism to inductively define properties of n -ary relations of AOT as relations among arbitrary tuples (see section 4.1).

Ideally, it should be possible to implement the full axiom system and deduction rules of our target system using a system of type classes and locales (which would provide an abstraction layer that is enforced on the logical level) and then merely to validate the consistency of the construction by instantiating, resp. interpreting these type classes and locales using a concrete semantic construction. However, in a complex target system that involves polymorphic axioms and complex interdependencies between its types, this is not always feasible and we have to rely on abstraction layers as described in the last section.

While a full discussion of the subtleties of type **classes** goes beyond the scope of this thesis, the short summary we provided above should be sufficient for understanding our use of type classes in chapter 4. Furthermore, it is important to note that while we use type classes to formulate theorems generically for several types, logically, the type classes can be eliminated for each concrete instantiation of such a theorem with fully specified concrete types.

²³More precisely, a type constructor that may depend on additional types that can be restricted to certain type classes, resp. *sorts*.

2.6. Implicit Interpretation and Assignment Functions in SSEs

Models of logical theories are usually formulated in terms of set-theory. In the following chapters, when we say that we construct *models* of the target logic AOT using our embedding, we do not construct classical set-theoretic models, but our implementation forms a model of AOT in HOL.

While a deep embedding would make interpretation and assignment functions explicit, they remain implicit in shallow embeddings. The meta-logic Isabelle/HOL itself involves constants and variables that are reused to represent the constants and variables of our target system. Consequently, we do not have to construct explicit interpretation and assignment functions, but can rely on HOL’s semantics for constants and variables.

In simple models of HOL,²⁴ every type has a set as its domain and a statement is valid, if it holds for every interpretation of the constants of each type and every assignment of the variables at each type.

A set-theoretic model of the embedded logic can be constructed by lifting a set-theoretic model of HOL through the semantic definitions of the SSE. The model-finding tool **nitpick** [12] can aid in making these lifted models concrete.

Technically, a shallow embedding defines a substructure in the models of HOL, which yields a model of the embedded logic under the defined validity.

2.7. Reproducing the Syntax of the Target Theory

To achieve the goal of constructing a custom theorem proving environment for a new theory by the means of an embedding, the primary concern is achieving a faithful representation of its axioms and deductive system and, thereby, to be able to faithfully reproduce reasoning in the embedded system.

However, for the embedding to be of practical use, it is equally important that the resulting representation is readable and, ideally, that a person that is familiar with the embedded theory, but has limited expertise in the particularities of the meta-logical system in which the theory is embedded, can still use the embedding to reason in the target system without a steep learning curve.

Isabelle’s *Isar* (*Intelligible semi-automated reasoning*) language itself (see [55]) is, as the name suggests, specifically tailored towards being readable. *Isar* makes up the *outer syntax* of an Isabelle theory file and consists of commands that specify theorems and structured proofs acting on Isabelle’s system of terms and types, which are formulated in *inner syntax*. *Inner syntax* is highly customizable. In the examples in the previous sections we already made use of the ability to define new (bold) operators using *mixfix* notation (see §8.2 in [55]). However, we only used the mechanism to provide symbols to be used inside the grammar tree of Isabelle/HOL’s own term structure.²⁵ In general

²⁴Ignoring complications due to e.g. polymorphism.

²⁵Thereby we cannot use symbols that are already used in HOL for our target logic.

Isabelle's inner syntax is described by a context-free priority grammar. It consists of a set of *terminal symbols*, an extensible set of *non-terminal symbols* and a set of *productions* (see §8.4 in [55]). For the purpose of embedding the syntax of a target theory during the construction of SSEs, it stands to reason to use the defined validity as root for the grammar subtree of the embedded language.

Reusing the example of the simple modal logic in section 2.3, this can be done as follows:

type-synonym $\circ_3 = \langle w \Rightarrow \text{bool} \rangle$

This time we do not use mixfix notation to directly introduce syntax for the validity constant.

definition $\text{valid-}\circ_3 :: \langle \circ_3 \Rightarrow \text{bool} \rangle$ **where** $\langle \text{valid-}\circ_3 p \equiv \forall w . p w \rangle$

Instead, we introduce a **nonterminal** as grammar root for the syntax of the embedded language. The nonterminal then serves as purely syntactic type for propositions in the grammar of our sub-language.

nonterminal prop_3

The nonterminal can be used as syntactic type in **syntax** definitions.

syntax $\text{valid-}\circ_3 :: \langle \text{prop}_3 \Rightarrow \text{bool} \rangle$ ($\langle \models \rightarrow 1 \rangle$)

Furthermore, we need to specify how propositions can be produced from terminals in the grammar. We want to use simple identifiers to refer to proposition variables. To that end we introduce a *copy-production* rule (a rule that is not tied to a constant). The terminal *id-position* is used for identifiers with additional markup information (i.e. it contains an encoding of the source position of the identifier to be used in the context of Isabelle/PIDE; see [52]).

syntax $:: \langle \text{id-position} \Rightarrow \text{prop}_3 \rangle$ ($\langle \langle \cdot \rangle \rangle$)

Now we can already construct a simple term in our new syntax:

term $\langle \models p \rangle$

Since we introduce an entirely new grammar subtree that is independent of the inner syntax of HOL, we can now reuse the same symbols for logical connectives as used in HOL (instead of having to use bold versions like in the previous section). We first define the connectives without syntax (here the symbols refer to connectives and operators in the language of HOL):

definition $\text{not-}\circ_3 :: \langle \circ_3 \Rightarrow \circ_3 \rangle$ **where** $\langle \text{not-}\circ_3 p \equiv \lambda w . \neg p w \rangle$

definition $\text{imp-}\circ_3 :: \langle \circ_3 \Rightarrow \circ_3 \Rightarrow \circ_3 \rangle$ **where** $\langle \text{imp-}\circ_3 p q \equiv \lambda w . p w \longrightarrow q w \rangle$

definition $\text{conj-}\circ_3 :: \langle \circ_3 \Rightarrow \circ_3 \Rightarrow \circ_3 \rangle$ **where** $\langle \text{conj-}\circ_3 p q \equiv \lambda w . p w \wedge q w \rangle$

definition $\text{disj-}\circ_3 :: \langle \circ_3 \Rightarrow \circ_3 \Rightarrow \circ_3 \rangle$ **where** $\langle \text{disj-}\circ_3 p q \equiv \lambda w . p w \vee q w \rangle$

definition $\text{box-}\circ_3 :: \langle \circ_3 \Rightarrow \circ_3 \rangle$ **where** $\langle \text{box-}\circ_3 p \equiv \lambda w . \forall v . p v \rangle$

definition $\text{dia-}\circ_3 :: \langle \circ_3 \Rightarrow \circ_3 \rangle$ **where** $\langle \text{dia-}\circ_3 p \equiv \lambda w . \exists v . p v \rangle$

And then define syntax for them in our grammar subtree. The syntax definitions are only used for parsing terms of the syntactic type prop_3 , i.e. terms in the grammar tree spanned by the marker \models introduced above.

2. Shallow Semantic Embeddings

syntax *not-o₃* :: $\langle \text{propo}_3 \Rightarrow \text{propo}_3 \rangle (\langle \neg \rangle [40] 40)$
syntax *imp-o₃* :: $\langle \text{propo}_3 \Rightarrow \text{propo}_3 \Rightarrow \text{propo}_3 \rangle (\text{infixl } \langle \longrightarrow \rangle 25)$
syntax *conj-o₃* :: $\langle \text{propo}_3 \Rightarrow \text{propo}_3 \Rightarrow \text{propo}_3 \rangle (\text{infixl } \langle \wedge \rangle 35)$
syntax *disj-o₃* :: $\langle \text{propo}_3 \Rightarrow \text{propo}_3 \Rightarrow \text{propo}_3 \rangle (\text{infixl } \langle \vee \rangle 30)$
syntax *box-o₃* :: $\langle \text{propo}_3 \Rightarrow \text{propo}_3 \rangle (\langle \Box \rangle [50] 50)$
syntax *dia-o₃* :: $\langle \text{propo}_3 \Rightarrow \text{propo}_3 \rangle (\langle \Diamond \rangle [50] 50)$

Now we can start to produce complex terms in our new syntax:

term $\langle \models \Box p \longrightarrow q \vee \Diamond r \rangle$

However, it is noteworthy that since the introduced grammar subtree is independent of the usual HOL grammar, a lot of details need to be considered. For example, without further work it is not possible to specify the types of terms in our grammar sub-tree. For that to work the :: syntax used in HOL would need to be reintroduced,²⁶ which requires some familiarity with Isabelle’s internals like the purely syntactic constant *-constrain* (see §8.5.4 in [55]).

As a simpler example, we also need to introduce parentheses explicitly in our grammar subtree:

syntax :: $\langle \text{propo}_3 \Rightarrow \text{propo}_3 \rangle (\langle '(-)' \rangle)$
term $\langle \models p \wedge (\Diamond q \longrightarrow p) \rangle$ — Without the above this would not parse.

It is still possible to mix the usual HOL syntax with our newly defined subgrammar to argue about validity:

lemma $\langle (\models \Diamond p \longrightarrow q) \longrightarrow (\neg(\models p) \vee (\models q)) \rangle$
using *dia-o₃-def imp-o₃-def valid-o₃-def* **by** *auto*

In the above the left-most implication and the diamond operator are the implication of the embedded logic and our defined possibility operator. The other logical connectives are the ones of the meta-logic HOL.

While the mechanism described above is sufficient for introducing an accurate representation of the syntax of most target theories that are compatible with the lexical syntax of Isabelle/Pure,²⁷ *reasoning* in the logic of the target theory entails additional challenges. For example, reasoning using Kripke-semantics usually involves proving statements relative to a fixed, but arbitrary possible world - however semantic possible worlds are not part of the syntax of the target theory and managing them can become a distraction. Therefore, we not only define custom inner syntax for the language of AOT, but also extend Isabelle’s outer syntax by custom commands that hide this complexity (see section 4.3).

²⁶Or, alternatively, new syntax could be introduced for the same purpose. In our embedding of AOT we will instead reproduce the fact that PLM implicitly imposes type constraints based on the names of its (meta-)variables.

²⁷Note that AOT does not fall into this category and requires additional and more complex means to arrive at a good approximation of its syntax as described in section 4.2.

In the following chapter we describe our target theory AOT in terms of our defined syntax, while the technical construction of the syntax representation itself is discussed in section 4.2.

3. Abstract Object Theory

The following sections provide a brief introduction to Abstract Object Theory (AOT or *object theory*). While the first section will explain the general idea and motivation of object theory, the subsequent sections reproduce the language and axiom system of AOT as implemented in our embedding. In the process, we hint at the major differences between the formulation of AOT in PLM and its incarnation in our embedding, referencing the discussion of implementational details in the next chapter where applicable. Recall that, as mentioned in section 1.4, all definitions and theorems are cited directly from our embedding and thereby verified by Isabelle. Exceptions to this rule are explicitly stated and marked by vertical bars at the page margins.

We restrict ourselves to the discussion of the second-order fragment of AOT which is the target of our embedding in Isabelle/HOL.¹ The second-order fragment is expressive enough for the analysis of a wide variety of objects occurring in philosophy and mathematics, including Basic Logical Objects like Truth Values and Extensions of Propositions (see A.8, resp. PLM chapter 10); Platonic Forms (see PLM chapter 11); Situations, Worlds, Times, and Stories (see A.11, resp. PLM chapter 12); Concepts (see PLM chapter 13) and Natural Numbers (see A.12, resp. PLM chapter 13).²

The applications of higher-order object theory and the challenges in representing it in Isabelle/HOL are briefly discussed in chapter 6. To get an intuition for the level of expressiveness of full higher-order object theory, note that it can be used to analyze e.g. Zermelo-Fraenkel set-theory itself as one of its abstract objects.

3.1. Overview

AOT is a metaphysical theory inspired by ideas of Ernst Mally and formalized by Edward Zalta. While the theory has been evolving for several decades (see [56, 59]), its most recent canonical presentation is given in *Principia Logico-Metaphysica* (PLM), which is under continuous development and the most recent version of which can be accessed as online monograph (see [62]). This thesis is written relative to the version dated October 13, 2021 (see [63]). A summary similar to the one in this section can also be found in [31].

AOT draws two fundamental distinctions, one between *abstract* and *ordinary* objects, and one between two modes of predication, namely, classical *exemplification* $[F]x$, or more

¹In the following chapters up until chapter 6, we will refer to the second-order fragment of AOT plainly as AOT or *object theory*.

²The chapter numbering of PLM is relative to [63].

generally, $[R]x_1\dots x_n$ and *encoding* $x[F]$, or more generally, $x_1\dots x_n[R]$.³ The variables x_1, x_2, \dots, x_n , resp. x, y, z, \dots , range over both ordinary and abstract objects and we can distinguish claims about these two kinds of objects by using the exemplification predications $O!x$ or $A!x$ to assert, respectively, that x exemplifies *being ordinary* or x exemplifies *being abstract*. Whereas ordinary objects are characterized only by the properties they exemplify, abstract objects may be characterized by both the properties they exemplify and the properties they encode. But only the latter play a role in their identity conditions: $A!x \ \& \ A!y \rightarrow (x = y \equiv \Box \forall F (x[F] \equiv y[F]))$, i.e. abstract objects are identical if and only if they necessarily encode the same properties. The identity for ordinary objects on the other hand is classical: $O!x \ \& \ O!y \rightarrow (x = y \equiv \Box \forall F (([F]x \equiv [F]y))$, i.e., ordinary objects x and y are identical if and only if they necessarily exemplify the same properties. It is axiomatic that ordinary objects fail to encode properties ($O!x \rightarrow \neg \exists F x[F]$), and so only abstract objects can be the subject of true encoding predications. For example, whereas Pinkerton (a real American detective) exemplifies being a detective and all his other properties (and doesn't encode any properties), Sherlock Holmes encodes *being a detective* (and all the other properties attributed to him in the novels), but doesn't exemplify *being a detective*. Holmes, on the other hand, intuitively exemplifies being a fictional character (but doesn't encode this property) and exemplifies any property necessarily implied by *being abstract* (e.g., he exemplifies *not having a mass, not having a shape, etc.*).⁴

The key axiom of AOT is the comprehension principle for abstract objects. It asserts, for every expressible condition on properties (i.e. for every expressible set of properties), that there exists an abstract object that encodes exactly the properties that satisfy the condition; formally: $\exists x (A!x \ \& \ \forall F (x[F] \equiv \varphi\{F\}))$

Here $\varphi\{F\}$ is the notation we use in the embedding to signify that φ may contain a free occurrence of the bound variable F (φ may not contain a free occurrence of x , unless we had explicitly added x in curly braces as well).⁵ Taken together, the comprehension principle and the identity conditions of abstract objects imply that abstract objects can be modelled as elements of the power set of properties: every abstract object uniquely corresponds to a specific set of properties.

Given this basic theory of abstract objects, AOT can elegantly define a wide variety of objects that have been postulated in philosophy or presupposed in the sciences, including

³Note that we use additional square brackets around property terms in exemplification or encoding formulas, except for specific (primitive or defined) constants like $E!$, $O!$ and $A!$. This is a syntactic concession that makes the process of parsing atomic formulas in Isabelle simpler. In AOT's usual notation these square brackets would be omitted, i.e. exemplification would be written as $Fx_1\dots x_n$ and encoding as xF .

⁴He encodes *having a mass, having a shape, etc.*, since these are properties attributed to him, at least implicitly, in the story. As an abstract object, however, he does *not* exemplify these properties, and so exemplifies their negations.

⁵PLM, on the other hand, uses the opposite convention: any *meta-variable* like φ may contain free occurrences of arbitrary variables (even those that are bound at the occurrence of φ) unless explicitly excluded, i.e. instead of $\varphi\{F\}$, PLM simply states φ and uses natural language to add the proviso that x may *not* occur free in φ . See 4.7.2 for a more detailed discussion.

Leibnizian concepts, Platonic forms, possible worlds, natural numbers, logically-defined sets, etc.

Another crucial aspect of the theory is its hyperintensionality: Relation identity is defined in terms of encoding rather than in terms of exemplification. Two properties F and G are stipulated to be identical if they are necessarily *encoded* by the same abstract objects ($F = G \equiv \Box \forall x (x[F] \equiv x[G])$).⁶ The theory does not impose any restrictions on the properties encoded by a particular abstract object. For example, the fact that an abstract object encodes the property $[\lambda x [F]x \ \& \ [G]x]$ does not imply that it also encodes either the property F , or G or even $[\lambda x [G]x \ \& \ [F]x]$ (which, although extensionally equivalent to $[\lambda x [F]x \ \& \ [G]x]$, is a distinct intensional entity).⁷

Therefore, without additional axioms, pairs of materially equivalent properties (in the exemplification sense), and even necessarily equivalent properties, are not forced to be identical. This is a key aspect of the theory that makes it possible to represent the contents of human thought much more accurately than classical exemplification logic would allow. For instance, the properties *being a creature with a heart* and *being a creature with a kidney* may be regarded as distinct properties despite the fact that they are extensionally equivalent. And *being a barber who shaves all and only those persons who don't shave themselves* and *being a set of all those sets that aren't members of themselves* may be regarded as distinct properties, although they are necessarily equivalent (both necessarily fail to be exemplified).

In the following sections, we provide a brief overview of the language, the axiom system and the deductive system of PLM as implemented in our embedding. For the original formulation of the system and a detailed discussion refer to [63], respectively [62].⁸

3.2. The Language

A precise description of AOT's language can be found in PLM chapter 7. In this section we provide a simplified informal description while explaining some of the deviations from AOT's conventions we use in our embedding.

The language distinguishes between constants, variables and terms at each type. The types of the second-order fragment consist of a type of individuals and of a type of n -place relations (for each $n \geq 0$), i.e. relations among n individuals.⁹ Formulas are considered as 0-place relation terms. PLM uses the following naming conventions for referring to the primitive language elements of each type:

⁶Traditionally, one might expect properties to be identical, if they are necessarily *exemplified* by the same objects instead.

⁷Note that this hyperintensionality also extends to propositions. We will see that proposition identity is defined in terms of property identity: $p = q \equiv [\lambda x p] = [\lambda x q]$

⁸At the time of writing both citations refer to the same version of PLM, but in the future [62] will refer to the most recent formulation of PLM, while [63] will contain the archived version of PLM that served as reference for this thesis. Naturally, referenced items and section numbers of PLM will be relative to [63].

⁹We briefly discuss the full higher-order type theory in chapter 6.

- Primitive individual constants: a_1, a_2, \dots
- Individual variables: x_1, x_2, \dots
- Primitive n -place relation constants: P_1^n, P_2^n, \dots
- n -place relation variables: F_1^n, F_2^n, \dots
- A distinguished 1-place relation constant for *being concrete*: $E!$

For increased readability, it allows to use less formal names, e.g. to use x, y, z, \dots in place of x_1, x_2, \dots ; p, q, r, \dots in place of F_1^0, F_2^0, \dots or F, G, H, \dots in place of F_1^1, F_2^1, \dots , etc.¹⁰

Additionally, PLM uses Greek letters for *meta-variables*, i.e. schematic meta-logical variables that may range over all variable names or all terms at a given type. By convention, it associates specific kinds of meta-variables with Greek letters (with additional numbered subscripts as needed) as follows:

- Meta-variables ranging over individual variables: ν, μ
- Meta-variables ranging over individual terms: κ
- Meta-variables ranging over n -place relation terms: Π^n
- Meta-variables ranging over formulas (i.e. zero-place relation terms): φ, ψ, \dots
- Meta-variables ranging over variables of any type: α, β, \dots
- Meta-variables ranging over terms of any type: τ, σ, \dots

PLM's system of constants, variables and meta-variables does not have to be reproduced in all detail in our embedding for the following reasons:

- The embedding collapses all alphabetic variants. This is discussed in more detail in section 4.7.1.
- The embedding implicitly generalizes free variables in theorems to *schematic variables*. This is discussed in more detail in section 4.7.3.
- The distinction between constants and variables is done at the meta-logical level of Isabelle/HOL, i.e. variables and constants of the same type are distinguished by declaring them as constant, resp. using them as variable in the meta-logic.

Furthermore, AOT introduces the following primitive formula- and term-forming operators:

- $\neg\varphi$, the *negation* operator
- $\Box\varphi$, the *necessity* operator
- $\mathcal{A}\varphi$, the *actuality* operator
- $\varphi \rightarrow \psi$, the *conditional* operator
- $\forall\alpha \varphi\{\alpha\}$, the *universal quantifier*¹¹

¹⁰See PLM item (5).

¹¹As mentioned in the previous section, PLM, by default, allows any free variables to occur in instances of a meta-variable, unless specified otherwise. For technical reasons, we choose the opposite convention, i.e. while a meta-variable may still contain any occurrence of variables that would remain *free*, any *bound*

- $\iota x\varphi\{x\}$, the *definite description* operator¹²
- $[\lambda x_1\dots x_n \varphi\{x_1\dots x_n\}]$, the *relation abstraction-* or λ -operator¹³

AOT uses two kinds of atomic formulas, *exemplification* formulas and *encoding* formulas. In PLM exemplification formulas are written as $\Pi\kappa_1\dots\kappa_n$, whereas encoding formulas are written as $\kappa_1\dots\kappa_n\Pi$. In our embedding, we use additional square brackets for easier parsing, i.e. we use:

- $[\Pi]\kappa_1\dots\kappa_n$ for atomic exemplification formulas
- $\kappa_1\dots\kappa_n[\Pi]$ for atomic encoding formulas

Furthermore, PLM allows for extending the above language using two kinds of definitions: definitions by identity and definitions by equivalence. While the inferential role of these definitions will be discussed in more detail in section 3.4.2, for now we rely on an intuitive understanding of their meaning. PLM *defines* multiple concepts that are commonly taken as primitive, such as logical existence and identity. These basic definitions can be found in section 7.2 of PLM and are implemented in our embedding in section A.5. In particular, PLM defines the following:

Derived connectives and quantifiers (see A.5.7):¹⁴

$$\begin{aligned}\varphi \& \psi &\equiv_{df} \neg(\varphi \rightarrow \neg\psi) \\ \varphi \vee \psi &\equiv_{df} \neg\varphi \rightarrow \psi \\ \varphi \equiv \psi &\equiv_{df} (\varphi \rightarrow \psi) \& (\psi \rightarrow \varphi) \\ \exists \alpha \varphi\{\alpha\} &\equiv_{df} \neg\forall \alpha \neg\varphi\{\alpha\} \\ \diamond\varphi &\equiv_{df} \neg\Box\neg\varphi\end{aligned}$$

Logical existence, i.e. the conditions under which a term *denotes* (see A.5.37):

$$\begin{aligned}\kappa\downarrow &\equiv_{df} \exists F [F]\kappa \\ \Pi\downarrow &\equiv_{df} \exists x_1\dots\exists x_n (x_1\dots x_n[\Pi]) \\ \varphi\downarrow &\equiv_{df} [\lambda x \varphi]\downarrow\end{aligned}$$

Being *ordinary* and being *abstract* (see A.5.67):

$$\begin{aligned}O! &=_{df} [\lambda x \diamond E!x] \\ A! &=_{df} [\lambda x \neg\diamond E!x]\end{aligned}$$

Identity of individuals (see A.5.72):

$$\kappa = \kappa' \equiv_{df} O!\kappa \& O!\kappa' \& \Box\forall F (([F]\kappa \equiv [F]\kappa') \vee (A!\kappa \& A!\kappa' \& \Box\forall F (\kappa[F] \equiv \kappa'[F])))$$

Identity of properties, i.e. 1-place relations (see A.5.81):

variables that may occur in an instance of the meta-variable have to be explicitly listed in curly brackets. See 4.7.2 for a more detailed discussion. Also note that while the meta-logical \forall -quantifier in HOL has wide scope, the universal quantifier of AOT has narrow scope and quantifying over complex formulas generally requires parentheses.

¹²Note that similarly to the universal quantifier above, definite descriptions have narrow scope and using complex formulas as matrix requires parentheses.

¹³Note that this includes the zero-place case $[\lambda \varphi]$, read as *that* φ . The scope of the λ -expression is explicitly given with the surrounding square brackets.

¹⁴The diamond operator $\diamond\varphi$ can be read as *possibly* φ . The precedence of the operators is demonstrated in A.5.25.

$$\Pi = \Pi' \equiv_{df} \Pi \downarrow \& \Pi' \downarrow \& \square \forall x (x[\Pi] \equiv x[\Pi'])$$

Identity of propositions, i.e. 0-place relations (see A.5.114):

$$\varphi = \psi \equiv_{df} \varphi \downarrow \& \psi \downarrow \& [\lambda x \varphi] = [\lambda x \psi]$$

Identity of n -place relations ($n \geq 2$):¹⁵

$$\left| \begin{array}{l} \Pi = \Pi' \equiv_{df} \Pi \downarrow \& \Pi' \downarrow \& \forall y_1 \dots \forall y_{n-1} ([\lambda x [\Pi] xy_1 \dots y_{n-1}] = [\lambda x [\Pi'] xy_1 \dots y_{n-1}]) \\ \& [\lambda x [\Pi] y_1 xy_2 \dots y_{n-1}] = [\lambda x [\Pi'] y_1 xy_2 \dots y_{n-1}] \& \dots \& [\lambda x [\Pi] y_1 \dots y_{n-1} x] = \\ [\lambda x [\Pi'] y_1 \dots y_{n-1} x] \end{array} \right|$$

Based on the described language and definitions we can state AOT's axiom system.

3.3. The Axiom System

In the following, we reproduce the full axiom system of the latest formulation of AOT, while commenting on several aspects that are specific to AOT. Unless explicitly noted otherwise, we will directly cite the axioms from our implementation while explaining notational and conceptual differences to the original axiom system. The original axiom system is stated in PLM chapter 8 with detailed explanations. The implementation in our embedding can be found in A.6. Throughout the section we will refer to the statements of the axioms in A.6, which will in turn refer to the item numbers of the respective axioms in PLM.

The first set of axioms build up a Hilbert-style deductive system for negation and implications following Mendelson's [35] system (see A.6.9):

$$\begin{array}{l} \varphi \rightarrow (\psi \rightarrow \varphi) \\ \varphi \rightarrow (\psi \rightarrow \chi) \rightarrow (\varphi \rightarrow \psi \rightarrow (\varphi \rightarrow \chi)) \\ \neg \varphi \rightarrow \neg \psi \rightarrow (\neg \varphi \rightarrow \psi \rightarrow \varphi) \end{array}$$

The next set of axioms constructs a quantifier logic for a free logic with non-denoting terms (see A.6.16, A.6.30). Formulas of the form $\tau \downarrow$ can be read as *the term τ denotes* and refer to the notion of logical existence that was *defined* in the previous section.¹⁶

$$\begin{array}{l} \forall \alpha \varphi\{\alpha\} \rightarrow (\tau \downarrow \rightarrow \varphi\{\tau\}) \\ \forall \alpha (\varphi\{\alpha\} \rightarrow \psi\{\alpha\}) \rightarrow (\forall \alpha \varphi\{\alpha\} \rightarrow \forall \alpha \psi\{\alpha\}) \\ \varphi \rightarrow \forall \alpha \varphi \\ [\Pi] \kappa_1 \dots \kappa_n \rightarrow \Pi \downarrow \& \kappa_1 \dots \kappa_n \downarrow \\ \kappa_1 \dots \kappa_n [\Pi] \rightarrow \Pi \downarrow \& \kappa_1 \dots \kappa_n \downarrow \end{array}$$

The last two axioms in the list above are noteworthy: they establish that if any atomic exemplification or encoding formula is true, then its primary terms are significant.

¹⁵The idea here is that two n -place relations are identical, if they denote and all their projections to $n - 1$ objects are identical. In the embedding it is tricky to reproduce the ellipse notation used for this definition directly, therefore the statement here is *not* cited from the embedding, as indicated by the vertical bars at the margins. The implementation of this definition in the embedding can be found in A.5.107 and is discussed in more detail in section 4.6.4.

¹⁶See section 4.7.2 for a discussion of the free variable notation using curly brackets and slight differences in the formulation compared to PLM. $\kappa_1 \dots \kappa_n \downarrow$ abbreviates $\kappa_1 \downarrow \& \dots \& \kappa_n \downarrow$.

Additionally, PLM establishes a base set of denoting terms with the following axiom:

$\tau \downarrow$, provided τ is a primitive constant, a variable, or a λ -expression in which the initial λ does not bind any variable in any encoding formula subterm.

Reproducing the natural language condition on τ in the embedding is non-trivial (see A.6.19, which uses the auxiliary predicate *INSTANCE-OF-CQT-2* defined in A.4.1283); we discuss the implementation of this axiom in detail in section 4.6.1.

For a simple intuition, note that all λ -expressions, in which every atomic formula in the matrix is an exemplification formula, denote, while special care has to be taken in the presence of encoding formulas.¹⁷ The axiom will be discussed in more detail later in this chapter.

The next axiom states that identical objects or identical relations can be substituted in formulas. Note that the used identity is not primitive, but was *defined* in the last section (see A.6.69).¹⁸

$$\alpha = \beta \rightarrow (\varphi\{\alpha\} \rightarrow \varphi\{\beta\})$$

The following axiom (see A.6.73) is the single *modally fragile axiom* of the system. This is signified by the turnstile operator \vdash . All other axioms are *modally strict* (for simplicity, we assume the corresponding turnstile operator \vdash_{\square} by default and refrain from mentioning it explicitly¹⁹). The distinction is discussed further in section 3.4.7.²⁰

$$\vdash \mathcal{A}\varphi \rightarrow \varphi$$

Intuitively, modally-fragile statements extend to *actual* truths, while modally-strict statements refer to *necessary* truths.

Apart from the above modally-fragile principle, the logic of actuality is governed by the following modally-strict axioms (see A.6.77):

$$\begin{aligned} \mathcal{A}\neg\varphi &\equiv \neg\mathcal{A}\varphi \\ \mathcal{A}(\varphi \rightarrow \psi) &\equiv \mathcal{A}\varphi \rightarrow \mathcal{A}\psi \\ \mathcal{A}\forall\alpha\varphi\{\alpha\} &\equiv \forall\alpha\mathcal{A}\varphi\{\alpha\} \\ \mathcal{A}\varphi &\equiv \mathcal{A}\mathcal{A}\varphi \end{aligned}$$

The logic of necessity and possibility is axiomatized using the classical K, T and 5 axioms of a propositional S5 modal logic (see A.6.91):

$$\square(\varphi \rightarrow \psi) \rightarrow (\square\varphi \rightarrow \square\psi)$$

¹⁷Note that this includes "implicit" encoding formulas that merely occur in the definiens of a defined constant used in the matrix. Those are also counted as encoding formula subterms of the matrix. See PLM items (8) and (17.3).

¹⁸PLM formulates the axiom as: $\alpha = \beta \rightarrow (\varphi \rightarrow \varphi')$, whenever β is *substitutable for* α in φ , and φ' is the result of replacing zero or more free occurrences of α in φ with occurrences of β . This is equivalent to the formulation in the embedding modulo the equivalence of alphabetic variants (see 4.7.1). The precise correspondence is discussed in more detail in section 4.7.2 at the example of the first quantifier axiom above.

¹⁹Respectively, printing of modally-strict statements cited from the embedding is set up in such a way that it does not print the turnstile operator.

²⁰Note that PLM uses $\mathcal{A}\varphi \equiv \varphi$ as axiom instead. However, the right-to-left direction is derivable and future versions of PLM will only use the left-to-right implication instead.

$$\begin{aligned}\Box\varphi &\rightarrow \varphi \\ \Diamond\varphi &\rightarrow \Box\Diamond\varphi\end{aligned}$$

Additionally, PLM states the following axiom (see A.6.101) that requires that there might be a concrete object that is not *actually* concrete, thereby ensuring that the domain of ordinary (i.e. possibly concrete) objects is non-empty²¹ and committing the system against modal collapse.

$$\Diamond\exists x (E!x \ \& \ \neg\mathcal{A}E!x)$$

The classical S5 modal logic is connected to the logic of actuality by the following two axioms (see A.6.108):

$$\begin{aligned}\mathcal{A}\varphi &\rightarrow \Box\mathcal{A}\varphi \\ \Box\varphi &\equiv \mathcal{A}\Box\varphi\end{aligned}$$

Definite descriptions in AOT are governed by the following axiom (see A.6.115), which will allow the derivation of a version of Russell's analysis of descriptions (see section 3.4.6):

$$x = \iota x\varphi\{x\} \equiv \forall z (\mathcal{A}\varphi\{z\} \equiv z = x)$$

A consistent axiomatization of complex relation terms in AOT requires some care. While λ -expressions follow the classical principles of α -, β - and η -conversion, they have to be suitably restricted to denoting terms, since not all λ -expressions are guaranteed to denote. Also note that the embedding generally collapses alphabetic variants (see 4.7.1), so while a version of α -conversion can be stated, it effectively reduces to the statement that denoting λ -expressions are self-identical in our implementation. α -, β - and η -conversion are formulated as follows (see A.6.125):

$$\begin{aligned}[\lambda\nu_1\dots\nu_n \varphi\{\nu_1\dots\nu_n\}]\downarrow &\rightarrow [\lambda\nu_1\dots\nu_n \varphi\{\nu_1\dots\nu_n\}] = [\lambda\mu_1\dots\mu_n \varphi\{\mu_1\dots\mu_n\}] \\ [\lambda x_1\dots x_n \varphi\{x_1\dots x_n\}]\downarrow &\rightarrow ([\lambda x_1\dots x_n \varphi\{x_1\dots x_n\}]x_1\dots x_n \equiv \varphi\{x_1\dots x_n\}) \\ [\lambda x_1\dots x_n [F]x_1\dots x_n] &= F\end{aligned}$$

Note that the last of the above axioms, η -conversion, also has the 0-place case $[\lambda p] = p$.²²

The following axiom of *coexistence* is specific to AOT and, together with generally extending AOT's free logic to relation terms and the refinement of base cases of denoting terms, a main aspect in the evolution of PLM that was originally triggered by its analysis using our embedding. It states that whenever a λ -expression denotes, any λ -expression with a matrix that is necessarily and universally equivalent on all abstracted variables will denote as well (see A.6.143):

$$\begin{aligned}[\lambda\nu_1\dots\nu_n \varphi\{\nu_1\dots\nu_n\}]\downarrow \ \&\ \Box\forall\nu_1\dots\forall\nu_n(\varphi\{\nu_1\dots\nu_n\} \equiv \psi\{\nu_1\dots\nu_n\}) \rightarrow \\ [\lambda\nu_1\dots\nu_n \psi\{\nu_1\dots\nu_n\}]\downarrow\end{aligned}$$

²¹Note that this consequence of the axiom relies, among others, on the fact that AOT allows deriving the Barcan formulas, in particular A.7.3470.

²²While identical by axiom, the syntactically distinct terms p and $[\lambda p]$ in AOT are meant to capture the natural-language distinction between the statement p itself and the statement *that p is true*. Also note that in the embedding the 0-place case is stated separately for η -conversion (see A.6.139) and α -conversion (see A.6.129). β -conversion in PLM is only stated for $n \geq 1$.

This axiom, together with AOT's move to a general free logic for complex terms, is discussed in more detail in section 3.7.

The remaining axioms govern AOT's second mode of predication, *encoding*.

The first of these axioms reduces n -ary encoding to unary encoding of projections as follows:²³

$$x_1 \dots x_n [F] \equiv x_1 [\lambda y [F] y x_2 \dots x_n] \& x_2 [\lambda y [F] x_1 y x_3 \dots x_n] \& \dots \& x_n [\lambda y [F] x_1 \dots x_{n-1} y]$$

The second axiom governing encoding states that encoding is *modally rigid* (see A.6.188):

$$x[F] \rightarrow \Box x[F]$$

Furthermore, as mentioned in the introduction of this chapter, encoding is reserved for *abstract* objects or in other words: ordinary objects do not encode properties (see A.6.192):

$$O!x \rightarrow \neg \exists F x[F]$$

The last axiom is the core axiom of AOT, the *Comprehension Principle for Abstract Objects*. For any expressible condition on properties, there exists an abstract object that encodes exactly those properties that satisfy the condition (see A.6.200):

$$\exists x (A!x \& \forall F (x[F] \equiv \varphi\{F\}))$$

All above axioms are to be understood as axiom *schemata*, i.e. their universal closures are axioms as well. Furthermore, for all axioms except the modally-fragile axiom of actuality, their modal closures (i.e. actualizations and necessitations) are taken as axioms as well.

3.4. The Deductive System

While an implementation of the complete deductive system of PLM chapter 9 can be found in A.7, a full discussion of the entire system would go beyond the scope of this thesis. However, we will discuss some aspects in detail with a particular focus on concepts that are required for the construction of natural numbers in chapter 5.

3.4.1. Primitive and Derived Meta-Rules

Since the axioms of AOT are to be understood as axiom schemata, i.e. their statement includes the statement of adequate closures, a single primitive rule of inference suffices for the deductive system of PLM, i.e. Modus Ponens (see A.7.16):²⁴

If φ and $\varphi \rightarrow \psi$ then ψ .

²³Note that similarly to the definition of n -ary relation identity, the formulation using ellipses is non-trivial to reproduce in the embedding. Therefore we again do *not* cite the axiom directly from the embedding, but state it as given in PLM modulo our notational conventions. The precise implementation in the embedding can be found in A.6.169 and is discussed in more detail in section 4.6.4.

²⁴Note that we are still citing rules directly from the embedding using a special printing mode for meta-rules.

While PLM can refer to structural induction on the complexity of a formula and the length of derivations to derive further meta-rules, by the nature of a Shallow Semantic Embedding, the precise term structure is not preserved, but instead terms are directly represented as objects in their semantic domain, and theorem-hood is not defined by means of derivations but internally constructed in terms of semantic validity. For that reason, in our embedding we derive the rules in question by referring to the semantic properties of the embedding. In particular, we derive the following rules semantically:

The deduction theorem (see A.7.170):

If $\varphi \vdash \psi$ then $\varphi \rightarrow \psi$.

I.e. if assuming φ it can be derived that ψ , then φ implies ψ .

The rule of necessitation RN (see A.7.110 and A.7.106):

If $\vdash_{\Box} \varphi$ then $\Box\varphi$.

The rule RN can only be applied to a formula φ , if φ has a *modally-strict proof*, as signified by \vdash_{\Box} . We discuss this in more detail in section 3.4.7.

The rule of generalization GEN (see A.7.99):

If for arbitrary α : $\varphi\{\alpha\}$ then $\forall\alpha \varphi\{\alpha\}$.

for arbitrary is implemented using a meta-logical all quantifier. This means that φ has to hold for an arbitrary choice of α and therefore independently of any local assumptions about any concrete α . This goes along with PLM's restriction to only allow the application of GEN, if α does not occur free in any assumption.

3.4.2. The Inferential Role of Definitions

PLM uses two kinds of definitions: definitions-by-equivalence $\varphi \equiv_{df} \psi$ and definitions-by-identity $\tau =_{df} \sigma$. While, intuitively, definitions by equivalence imply definiens and definiendum to be equivalent (\equiv) and definitions by identity imply them to be identical ($=$), further care is required when stating their precise inferential roles.

Definitions by Equivalence Since equivalence (\equiv) is itself *defined* using a definition-by-equivalence (as mentioned in section 3.2), equivalence itself cannot be used to specify the inferential role of definitions-by-equivalence. Instead the inferential role has to be formulated in terms of primitives of the language, i.e. in terms of implications.

To that end, PLM formulates a *Rule of Definition by Equivalence* that we reproduce in the embedding as follows (see A.7.118):

If $\varphi \equiv_{df} \psi$ then $\varphi \rightarrow \psi$.

If $\varphi \equiv_{df} \psi$ then $\psi \rightarrow \varphi$.

In other words, a definition-by-equivalence of the form $\varphi \equiv_{df} \psi$ introduces the closures of $\varphi \rightarrow \psi$ and $\psi \rightarrow \varphi$ as necessary axioms.²⁵

²⁵Therefore, the rule has axiomatic character and also has to be derived from the semantics in the appendix. The same is true for the rule of definition by identity below.

The principle that a definition-by-equivalence in fact implies definiens and definiendum to be equivalent becomes a derived rule (see A.7.601):

$$\text{If } \varphi \equiv_{df} \psi \text{ then } \varphi \equiv \psi.$$

However, note that while this also implies *necessary* equivalence of definiens and definiendum (using the rule of necessitation RN mentioned above), in AOT necessary equivalence of propositions does not imply their identity. Another noteworthy subtlety in PLM's use of definitions by equivalence is its convention that such definitions are stated using object-level variables, even though those variables act as meta-variables. For instance, following PLM's conventions, the definition of identity for properties (see 3.2) can be stated as:

$$F = G \equiv_{df} F \downarrow \& G \downarrow \& \Box \forall x (x[F] \equiv x[G])$$

However, replacing F and G by any property term constitutes a valid instance of this definition. In order to avoid confusion for a reader that is not familiar with this convention, we choose to either state the definitions using meta-variables instead,²⁶ or state a derived equivalence as theorem using object-variables in its place (which allows forgoing clauses about the significance of the involved terms in the definiendum). I.e. in the upcoming chapters, instead of stating the full definition-by-equivalence for e.g. property identity, we may illustrate the definition using a simpler theorem using regular object-level variables while dropping significance constraints:

$$F = G \equiv \Box \forall x (x[F] \equiv x[G])$$

This simplification is justified, since most definitions of PLM are formulated in such a way that the definiens implies the significance of all free terms in the definiendum, so unless noted otherwise it can be assumed that the definiendum of a definition-by-equivalence is false for non-denoting terms. A notable example of an exception to this rule is the definition of non-identity: non-identity between two terms holds not only if both terms denote with different denotations, but also if either of them fails to be significant.

Definitions by Identity A subtlety in definitions by identity is the question of when a defined term denotes. This is made explicit in the formulation of the *Rule of Definition by Identity* (see A.7.141):

$$\begin{aligned} &\text{If } \tau\{\alpha_1 \dots \alpha_n\} =_{df} \sigma\{\alpha_1 \dots \alpha_n\} \text{ then} \\ &(\sigma\{\tau_1 \dots \tau_n\} \downarrow \rightarrow \tau\{\tau_1 \dots \tau_n\} = \sigma\{\tau_1 \dots \tau_n\}) \& (\neg \sigma\{\tau_1 \dots \tau_n\} \downarrow \rightarrow \neg \tau\{\tau_1 \dots \tau_n\} \downarrow). \end{aligned}$$

I.e. if the definiens denotes, a definition by identity implies identity, if the definiens fails to denote, a definition by identity implies that the definiendum fails to denote as well. In the simplest case of a definition-by-identity that does not involve any free variables, the definition-by-identity reduces to a plain identity, if the definiens provably denotes.

²⁶For example, property identity may be stated as: $\Pi = \Pi' \equiv_{df} \Pi \downarrow \& \Pi' \downarrow \& \Box \forall x (x[\Pi] \equiv x[\Pi'])$

3.4.3. Reasoning in PLM

Based on the fundamental meta-rules above, PLM derives further theorems and rules governing Negations and Conditionals (see A.7.165); Quantification (see A.7.629); Logical Existence, Identity and Truth (see A.7.913); Actuality and Descriptions (see A.7.1720); Necessity (see A.7.2507); Relations (see A.7.4220); Objects (see A.7.7508) and Propositional Properties (see A.7.8826).

Apart from the specific items discussed in the following sections, it may generally be helpful to be aware of the following derived properties of the deductive system:

- Propositional reasoning in AOT is classical.²⁷
- Modal reasoning can be read semantically as following Kripke-semantics without accessibility relation and with a fixed designated actual world for the actuality operator. In particular, AOT follows an S5 modal logic with actuality operator and Barcan formulas.
- The free logic extends to all types, but all propositions provably denote. Quantifiers range over denoting objects and the defined identity constitutes an *existing identity*, i.e. to be identical two entities need to both denote *and* denote the same thing.²⁸

3.4.4. Restricted Variables

A common theme in abstract object theory is the definition and analysis of certain families of objects. For instance, Possible Worlds, Logical Sets or Natural Numbers are specific families of abstract objects. Furthermore, some constructions involve talking about the Ordinary Objects specifically. To be able to more conveniently state theorems involving such families of objects, PLM introduces a generic mechanism for defining *restricted variables* that range over objects satisfying a certain *restriction condition* (see PLM section 10.5).

A formula ψ that involves a single free, unrestricted variable α , i.e. a formula that can be written as $\psi\{\alpha\}$ in the notational convention of the embedding, is called a *restriction condition*, just in case that it is both *non-empty*, i.e. $\exists\alpha \psi\{\alpha\}$ is a (modally-strict) theorem, and has *strict existential import*, i.e. $\psi\{\tau\} \rightarrow \tau\downarrow$ is a (modally-strict) theorem. PLM distinguishes *restriction conditions*, in which non-emptiness and strict existential import are modally-strict and *weak restriction conditions*, in which neither are required to be modally-strict. Since the parts of PLM implemented in our embedding do not involve weak restriction conditions, the embedding thus far forgoes an implementation of them. However, the current implementation can easily be extended to also cover weak restriction conditions.

²⁷In particular, as stated in PLM item (113), all classical propositional tautologies are theorems of AOT.

²⁸Respectively, denote and satisfy the axiom of the substitution of identicals. Our implementation has the property that PLM's defined identity implies meta-logical identity.

Furthermore, a restriction condition is *rigid*, if additionally $\forall \alpha (\psi\{\alpha\} \rightarrow \Box\psi\{\alpha\})$ is a modally-strict theorem.²⁹

An example of a rigid restriction condition is *being ordinary*, i.e. $O!x$.³⁰

Restricted variables are governed by the following conventions (see PLM item (331)): Let γ be a variable that is restricted by the restriction condition $\psi\{\alpha\}$. Then a quantified formula of the form $\forall \gamma \varphi\{\gamma\}$ is to be understood as an abbreviation of $\forall \alpha (\psi\{\alpha\} \rightarrow \varphi\{\alpha\})$ for an unrestricted variable α . Furthermore, $\exists \gamma \varphi\{\gamma\}$ abbreviates $\exists \alpha (\psi\{\alpha\} \& \varphi\{\alpha\})$ and similar conventions are introduced for definite descriptions, λ -expressions and definitions. For non-rigid restriction conditions, PLM bans the use of free restricted variables in theorem statements and merely allows bound occurrences. However, for rigid restriction conditions PLM allows the use of free restricted variables in theorem statements and extends reasoning in the system by allowing to take $\psi\{\gamma\}$ as modally-strict axiom.

This construction allows natural reasoning with rigidly restricted variables, i.e. the fundamental rules *GEN* and *RN* as well as usual quantifier and modal reasoning (e.g. \forall -elimination, existential introduction, Barcan formulas, etc.) can be extended to restricted variables governed by a rigid restriction condition.

3.4.5. Identity on the Ordinary Objects

While the general definition of identity for individuals was given in section 3.2, PLM also introduces an identity *relation* on the ordinary objects and matching infix notation (see A.7.7582):³¹

$$\begin{aligned} (=_{\mathcal{E}}) &=_{df} [\lambda xy \ O!x \ \& \ O!y \ \& \ \Box \forall F \ ([F]x \equiv [F]y)] \\ x =_{\mathcal{E}} y &\equiv O!x \ \& \ O!y \ \& \ \Box \forall F \ ([F]x \equiv [F]y) \end{aligned}$$

Notably, while the above definition of $=_{\mathcal{E}}$ constitutes a denoting *relation* (the λ -expression does not involve encoding claims and thereby denotes by axiom), general identity of both ordinary *and* abstract objects *does* involve encoding claims and does not constitute a general relation of identity. In particular, the assumption that general identity among individuals is a relation contradicts the existence of indistinguishable abstract objects discussed in section 3.8.1.

Identity on the ordinary objects will play an important role in PLM's analysis of Natural Numbers, discussed in chapter 5.

²⁹While our embedding supports non-rigid restriction condition, the parts of PLM currently implemented involve only rigid restriction conditions.

³⁰It is a theorem that there necessarily exists an ordinary object $\Box \exists x \ O!x$ (see A.7.7510), as a consequence of the modal axiom $\Diamond \exists x \ (E!x \ \& \ \neg \mathcal{A}E!x)$, the definition of *being ordinary* as $O! =_{df} [\lambda x \ \Diamond E!x]$, β -conversion and some modal reasoning. Furthermore, strict existential instance follows from one of the quantifier axioms (see A.6.34). Rigidity is a consequence of the definition of being ordinary (see A.7.4158).

³¹Note that the introduced infix notation $x =_{\mathcal{E}} y$ merely abbreviates $[(=_{\mathcal{E}})]xy$ and the stated equivalence is a theorem derived by β -conversion.

3.4.6. Definite Descriptions

The following axiom, that was already mentioned in section 3.3, governs definite descriptions:

$$x = \iota x\varphi\{x\} \equiv \forall z (\mathcal{A}\varphi\{z\} \equiv z = x)$$

In particular, definite descriptions are *modally rigid* and refer to the object that satisfies the matrix φ in the actual world. While an extensive set of theorems and rules for reasoning with definite descriptions is given in section 9.8 of PLM (see A.7.1720), for an intuitive understanding of descriptions in AOT it suffices to note that they follow a variant of Russell’s analysis of definite descriptions. In particular, atomic formulas involving definite descriptions can be translated to existence claims as follows:³²

$$\begin{aligned} [\text{II}]\iota x\varphi\{x\} &\equiv \exists x (\mathcal{A}\varphi\{x\} \ \& \ \forall z (\mathcal{A}\varphi\{z\} \rightarrow z = x) \ \& \ [\text{II}]x) \\ \iota x\varphi\{x\}[\text{II}] &\equiv \exists x (\mathcal{A}\varphi\{x\} \ \& \ \forall z (\mathcal{A}\varphi\{z\} \rightarrow z = x) \ \& \ x[\text{II}]) \end{aligned}$$

I.e. an atomic formula involving a description is equivalent to there being a unique object that actually satisfies the matrix of the description and this object satisfies the given atomic formula. *The author of PLM is called “Edward Zalta”* is equivalent to *There is a unique object that is actually the author of PLM and this object is called “Edward Zalta”*, respectively to *There is an object that is actually the author of PLM, every other object that is actually the author of PLM is the same object, and this object is called “Edward Zalta”*.

Similarly, a definite description denotes, just in case that there is a unique object that actually satisfies its matrix:

$$\iota x\varphi\{x\}\downarrow \equiv \exists!x(\mathcal{A}\varphi\{x\})$$

E.g. “**the** author of Principia Mathematica” does not denote, since Principia Mathematica was actually coauthored by Russell and Whitehead, i.e. there is no unique object that actually authored Principia Mathematica (even if possibly either either of them might have authored it alone in a non-actual world).

3.4.7. Modally-Strict and Modally-Fragile Theorems

PLM constructs two derivational systems, the first, written as \vdash , is concerned with modally-fragile theorems, while the second, written as \vdash_{\square} , is concerned with modally-strict theorems.³³ The main difference between the two is that the \vdash -system is equipped with the modally-fragile axiom of actuality and its universal (though not its necessary or actual) closures (as mentioned in section 3.3):

³²For simplicity we restrict ourselves to the case of unary exemplification and encoding. Analog principles hold for n -ary exemplification and encoding formulas.

³³To state modally-strict and modally-fragile theorems in our embedding, we also use **AOT-theorem** and **AOT-act-theorem** respectively. Cited statements that are undecorated are modally-strict by default.

$$\vdash \mathcal{A}\varphi \rightarrow \varphi$$

On the other hand, every modally-strict axiom is also part of the \vdash -system. As indicated by this axiom, semantically, the modally-fragile theorems are concerned with all actual truths, whereas modally-strict theorems are concerned with truths that are necessary.³⁴ Consequently, the fundamental meta-rule RN mentioned in section 3.4.1 is restricted to modally-strict derivations: If φ has a modally-strict proof, then its necessitation $\Box\varphi$ is an (again modally-strict) theorem.

PLM's axiom system has the property that the actualization of any modally-fragile axiom (in particular $\mathcal{A}(\mathcal{A}\varphi \rightarrow \varphi)$) is a modally-strict theorem (see A.7.1730).

Consequently, for any formula that is derivable as modally-fragile theorem, i.e. $\vdash \varphi$, it holds that $\vdash_{\Box} \mathcal{A}\varphi$. In particular, it follows from $\vdash \Box\varphi$ that $\vdash_{\Box} \mathcal{A}\Box\varphi$, which implies $\vdash_{\Box} \varphi$. PLM refers to this principle as the *weak converse of RN*.

However, while true in our semantics and derivable in the unextended axiom system of PLM, PLM rejects the weak converse of RN in general. The goal is to explicitly allow augmenting the theory by asserting contingent truths without having to assert their actualization as modally-strict fact. After doing so, the weak converse of RN would indeed fail, so in order to keep the reasoning system robust against additional assertions of this kind, PLM does not allow reasoning using the weak converse of RN. A detailed discussion of this issue can be found in PLM item (71).

While section 4.7.4 hints at a potential way of reproducing this strict distinction using a more complex semantics, for simplicity we refrain from doing so in our embedding and instead rely on our abstraction layer to prevent reasoning using the weak converse of RN, while it remains valid in our semantics.³⁵

3.5. Interesting Theorems of AOT

Before we continue our technical discussion of the specifics of AOT, we give some examples of interesting abstract objects and properties that can be derived about them.

3.5.1. Truth Values as Abstract Objects

An example of AOT's ability to define metaphysical entities as abstract objects and derive interesting properties about them, is truth values.

In AOT, it is possible to provide a *syntactic* definition of what it means to be a truth value of a proposition (see A.8.11):

$$\text{TruthValueOf}(x,p) \equiv_{df} A!x \ \& \ \forall F (x[F] \equiv \exists q ((q \equiv p) \ \& \ F = [\lambda y q]))$$

³⁴Consequently, in our models modally-fragile axioms and theorems are semantically valid in a designated actual world, while modally-strict axioms and theorems are valid in all semantic possible worlds.

³⁵Note that it is still possible to add contingent truths to the modally-fragile system of the embedding and - while it would immediately become derivable semantically - just refrain from adding a modally-strict actualization of the assertion to the abstraction layer.

An abstract object x is the truth value of a proposition p , just in case it encodes a property F , if and only if there is a proposition q that is equivalent to p and F is the propositional property *being a y, such that q*.

We say that an abstract object x encodes a proposition p , if it encodes the property *being a y, s.t. p*, i.e. if $x[\lambda y p]$. Using that notion, it is possible to define two particular truth values, i.e. The True and The False, as *the* abstract object that encodes all propositions that are true, resp. all propositions that are false (see A.8.217):

$$\begin{aligned}\top &=_{df} \iota x(A!x \ \& \ \forall F (x[F] \equiv \exists p (p \ \& \ F = [\lambda y p]))) \\ \perp &=_{df} \iota x(A!x \ \& \ \forall F (x[F] \equiv \exists p (\neg p \ \& \ F = [\lambda y p])))\end{aligned}$$

Now it becomes possible to *derive* natural properties of truth values analytically. For instance:

- There are exactly two truth values (see A.8.312):³⁶
 $\exists x \exists y (TruthValue(x) \ \& \ TruthValue(y) \ \& \ x \neq y \ \& \ \forall z (TruthValue(z) \rightarrow z = x \vee z = y))$
- The True is distinct from The False (see A.8.222): $\top \neq \perp$
- The True and The False are truth values (see A.8.290):³⁷
 $\vdash TruthValue(\top)$
 $\vdash TruthValue(\perp)$
- A proposition is true, iff its truth value is The True and it is false, iff its truth value is The False (see A.8.400):
 $\vdash TruthValueOf(x,p) \rightarrow (p \equiv x = \top)$
 $\vdash TruthValueOf(x,p) \rightarrow (\neg p \equiv x = \perp)$

The analysis of truth values is an example of AOT's ability to define and analyze abstract objects that faithfully represent entities that are usually only considered semantically. AOT's analysis of possible worlds that is discussed in the next section is another example of this feature.

3.5.2. Fundamental Theorems of Possible Worlds

Similarly to truth values, possible worlds are usually solely considered semantically. However, AOT allows to *define* possible worlds as a class of abstract objects and derive fundamental theorems about them.

As a first step, consider the following definition of a *Situation* (see A.11.10):

$$Situation(x) \equiv_{df} A!x \ \& \ \forall F (x[F] \rightarrow Propositional(F))$$

³⁶An abstract object x is a truth value, if it is the truth value of some proposition p :
 $TruthValue(x) \equiv_{df} \exists p TruthValueOf(x,p)$

³⁷Note that since descriptions are modally rigid and refer to objects in the actual world, these theorems are modally-fragile, i.e. *actual*, but not *necessary* truths: While there are necessarily exactly two truth values, different possible worlds have different truth values (exactly two in each world). The defined \top and \perp are the truth values of the actual world.

3. Abstract Object Theory

A situation is an abstract object that only encodes propositional properties. A property F is propositional, just in case that there is a proposition p , s.t. $F = [\lambda y p]$ (see A.7.8829):

$$\text{Propositional}(F) \equiv_{df} \exists p F = [\lambda y p]$$

Being a situation is a *rigid restriction condition*,³⁸ so as explained in section 3.4.4, we can use s as a restricted variable that ranges over situations. A situation makes a proposition true, written $s \models p$, just in case it encodes $[\lambda y p]$:³⁹

$$s \models p \equiv s[\lambda y p]$$

Now a *possible world* can be defined as a situation that possibly makes exactly those propositions true that are true (see A.11.1399):

$$\text{PossibleWorld}(x) \equiv_{df} \text{Situation}(x) \ \& \ \diamond \forall p (x \models p \equiv p)$$

Similarly to situations, it can be shown that being a possible world is a rigid restriction condition, allowing the use of w as a restricted variable ranging over possible worlds.

Now it can be derived that possible worlds are *possible* (i.e. *possibly actual*), *consistent* and *maximal* situations and, furthermore, that any abstract object that is a possible and maximal situation is a possible world:

- $\text{Actual}(x) \equiv_{df} \text{Situation}(x) \ \& \ \forall p (x \models p \rightarrow p)$

A situation is actual, if it only makes true propositions true (see A.11.728).

- $\text{Possible}(x) \equiv_{df} \text{Situation}(x) \ \& \ \diamond \text{Actual}(x)$

A situation is possible, if it is possibly actual (see A.11.1127).

- $\text{Consistent}(x) \equiv_{df} \text{Situation}(x) \ \& \ \neg \exists p (x \models p \ \& \ x \models \neg p)$

A situation is consistent, if it does not make any proposition and its negation true at the same time (see A.11.1039).

- $\text{Maximal}(x) \equiv_{df} \text{Situation}(x) \ \& \ \forall p (x \models p \vee x \models \neg p)$

A situation is maximal, if any proposition is either true or false in it (see A.11.1559).

- $\text{Possible}(w) \ \& \ \text{Consistent}(w) \ \& \ \text{Maximal}(w)$

Possible worlds are possible, consistent and maximal (see A.11.1484, A.11.1508 and A.11.1561).

- $\text{PossibleWorld}(x) \equiv \text{Maximal}(x) \ \& \ \text{Possible}(x)$

An abstract object is a possible world, if and only if it is a maximal and possible situation (see A.11.1610).

The *fundamental theorems of possible worlds* relate the defined possible worlds to possibility and necessity of the modal logic of AOT (see A.11.2236 and A.11.2263):

$$\diamond p \equiv \exists w w \models p$$

³⁸Note that by *being a situation* we refer to the *formula* $\text{Situation}(x)$ in this case. The term $[\lambda x \text{Situation}(x)]$ is not guaranteed to denote. Similarly for *being a possible world* below.

³⁹Note that the double turnstile symbol \models used here is a defined symbol in AOT and not the semantic symbol used in chapter 2 and again starting from section 4.1.3 to symbolize truth conditions in a semantic possible world relative to the meta-logic HOL. Also note that by convention the defined double turnstile symbol \models of AOT is to be understood to have the narrowest possible scope, i.e. $w \models p \ \& \ q$ is to be read as $(w \models p) \ \& \ q$.

$$\Box p \equiv \forall w w \models p$$

A proposition is possible, just in case *some* possible world makes it true, and necessary, just in case *every* possible world makes it true.

Furthermore, it can be shown that the basic connectives and quantifiers are well-behaved with respect to being true in a possible world, i.e. (see A.11.2361 and following):⁴⁰

- $w \models (p \ \& \ q) \equiv w \models p \ \& \ w \models q$
- $w \models (p \rightarrow q) \equiv w \models p \rightarrow w \models q$
- $w \models (p \vee q) \equiv w \models p \vee w \models q$
- $w \models (p \equiv q) \equiv (w \models p \equiv w \models q)$
- $w \models \forall \alpha \varphi\{\alpha\} \equiv \forall \alpha w \models \varphi\{\alpha\}$
- $w \models \exists \alpha \varphi\{\alpha\} \equiv \exists \alpha w \models \varphi\{\alpha\}$

Taken together this reproduces the semantic analysis of AOT with Kripke semantics syntactically within the derivational system of AOT itself. It is a notable feature of AOT that it can, in this sense, accurately reason about its own semantics.

While PLM provides an analysis of a range of further interesting objects, including Logical Sets, Platonic forms, Stories and Fictional Characters and Leibnizian Concepts, a full discussion of them would go beyond the scope of this thesis.

Instead, we discuss some further technical properties of the system of AOT that will be relevant for our discussion of natural numbers in chapter 5.

3.6. Avoiding Known Paradoxes

3.6.1. The Clark-Boolos Paradox

Naive formulations of AOT, in which all λ -expression are assumed to denote relations, are subject to the Clark-Boolos Paradox.⁴¹

In particular consider the λ -expression $[\lambda x \exists F (x[F] \ \& \ \neg[F]x)]$, i.e. *being an object, s.t. there is a property it encodes, but does not exemplify*. The assumption that this property denotes leads to paradox (see A.7.4362): Assuming that the λ -expression denotes, call it K , s.t. $K = [\lambda x \exists F (x[F] \ \& \ \neg[F]x)]$. By the comprehension principle of abstract objects, there is an abstract object a that encodes exactly K and no other properties. Now if a were to exemplify K , it would follow by β -conversion that there is a property that a encodes, but does not exemplify. However, the only property encoded by a is K , which *is* exemplified by a by assumption yielding a contradiction.

⁴⁰Notably, the proofs of the last two theorems were contributed to AOT on the basis of proofs in our embedding.

⁴¹The paradox was discovered by Romane Clark in a formalization of Meinong's theories by William Rapaport who reported it in [47], p. 225. Independently, George Boolos constructs the same paradox in [14], p. 17, under the name *SuperRussell* in an analysis of Frege's foundations of arithmetic.

If, on the other hand, a does not exemplify K , it follows that a encodes K and does not exemplify K , so it serves as witness to the claim $\exists F (a[F] \ \& \ \neg[F]a)$. Thus it follows by β -conversion that a *does* exemplify K yielding a contradiction.

Previous formulations of PLM disbarred λ -expressions like K syntactically: a λ -expression was only considered to be *well-formed*, if its matrix was a so-called *propositional formula*. A formula was defined to be propositional, just in case that it did not contain encoding subformulas. However, an oversight in the precise formulation of these provisos made it possible to reintroduce the paradox as described in the next section.

In the current formulation of PLM, the paradoxical λ -expression is well-formed, but does not fall under the axiom that stipulates base cases of denoting terms (see 3.3): The initial λ binds a variable that occurs in an encoding formula subterm.

Given that the assumption that the λ -expression denotes leads to contradiction, it now provably fails to denote (see A.7.4362):

$$\neg[\lambda x \exists G (x[G] \ \& \ \neg[G]x)]\downarrow$$

3.6.2. Reintroduction of the Clark-Boolos Paradox

When attempting to construct an embedding of a previous formulation of PLM (see [64]) that relied on restricting matrices of λ -expressions to *propositional formulas* as defined in the previous section, we found the following oversight (see [29] and [31]):

Encoding formulas embedded in the matrix of definite descriptions within complex formulas were not considered *encoding subformulas* and thereby such complex formulas were still considered propositional.⁴²

This allowed constructing λ -expressions that were considered well-formed, but (actually) equivalent to the paradoxical Clark-Boolos property K discussed above, namely:

$$K' =_{df} [\lambda x [\lambda y \forall p (p \rightarrow p)]\iota z (z = x \ \& \ \exists F (z[F] \ \& \ \neg[F]z))]$$

Since $[\lambda y \forall p (p \rightarrow p)]$ is (necessarily) universally exemplified by all objects, it being exemplified by a definite description is equivalent to the matrix of the description being *actually* satisfied by a unique object, i.e.:⁴³

AOT-theorem $\langle [\lambda y \forall p (p \rightarrow p)]\iota z (z = x \ \& \ \exists F (z[F] \ \& \ \neg[F]z)) \equiv \exists!z (\mathcal{A}(z = x \ \& \ \exists F (z[F] \ \& \ \neg[F]z))) \rangle$

proof (*safe intro!*: $\equiv I \rightarrow I$)

AOT-assume $\langle [\lambda y \forall p (p \rightarrow p)]\iota z (z = x \ \& \ \exists F (z[F] \ \& \ \neg[F]z)) \rangle$

AOT-hence $\langle \iota z (z = x \ \& \ \exists F (z[F] \ \& \ \neg[F]z))\downarrow \rangle$

using *cqt:5:a[axiom-inst, THEN $\rightarrow E$, THEN $\& E(2)$] by blast*

AOT-thus $\langle \exists!z (\mathcal{A}(z = x \ \& \ \exists F (z[F] \ \& \ \neg[F]z))) \rangle$

using *actual-desc:1[THEN $\equiv E(1)$] by blast*

⁴²While the matrix of a definite description (or a λ -expression) is a *subterm* of any formula containing the description (or λ -expression), it is not a *subformula*. See PLM items (6), (7) and (8).

⁴³We choose this opportunity to demonstrate that reasoning in our embedding is readable and intuitively understandable, by directly proving the equivalence in the syntax of the embedding. The proof was automatically verified during the generation of this document as mentioned in section 1.4.

next

AOT-assume $\langle \exists !z (\mathcal{A}(z = x \ \& \ \exists F (z[F] \ \& \ \neg[F]z))) \rangle$

AOT-hence $\langle \iota z (z = x \ \& \ \exists F (z[F] \ \& \ \neg[F]z)) \downarrow \rangle$

using *actual-desc:1* [*THEN* $\equiv E(\varnothing)$] **by** *simp*

AOT-thus $\langle [\lambda y \forall p (p \rightarrow p)] \iota z (z = x \ \& \ \exists F (z[F] \ \& \ \neg[F]z)) \rangle$

by (*safe intro!*: $\beta \leftarrow C$ *cqt:2* *GEN* $\rightarrow I$)

qed

The left-hand side is equivalent to $[K']x$ by β -conversion (assuming K' is a well-formed, respectively a denoting relation). The right-hand side can be simplified to $\mathcal{A}\exists F (x[F] \ \& \ \neg[F]x)$, so it is equivalent to $\mathcal{A}[K']x$. Thereby, assuming K' denotes yields a modally-fragile proof of a contradiction following the argument given in the previous section.⁴⁴

An obvious solution to this issue would have been to further restrict propositional formulas to not only disbar encoding subformulas, but also encoding formula subterms, i.e. to also disallow encoding formulas embedded in matrices of descriptions and thereby disbaring K' as not well-formed.

However, this had resulted in the loss of the ability to formulate interesting λ -expressions involving descriptions that are safe and were deemed worthwhile to preserve. Therefore, this solution was rejected in favour of extending AOT's free logic to relation terms as described in the next section. In the most recent formulation of AOT, it becomes a theorem that the paradoxical relation K' does not denote on pain of contradiction (see A.7.4560, resp. A.7.4620).

3.7. Extending AOT's Free Logic to Relations

In the aftermath of the discovery of the reintroduction of the Clark-Boolos paradox, AOT's free logic was extended to all its types.⁴⁵

In the process, the definitions for logical existence ($\tau \downarrow$) mentioned in section 3.2 were introduced.⁴⁶ Notably, it is possible to define the conditions under which relation terms denote using *encoding*, i.e. $\Pi \downarrow \equiv_{df} \exists x_1 \dots \exists x_n (x_1 \dots x_n [\Pi])$, while a similar definition using exemplification would fail in the second-order fragment, since there are denoting, but necessarily unexemplified properties, i.e. $\exists x_1 \dots \exists x_n ((\Pi)x_1 \dots x_n)$ may be false for denoting Π .⁴⁷

⁴⁴The proof can also be strengthened to be modally-strict, see A.7.4620.

⁴⁵AOT previously also involved a free logic. However, it was restricted to individual terms to account for non-denoting definite descriptions. While there were λ -expressions that were not considered well-formed syntactically, all λ -expressions that were well-formed were implicitly assumed to denote.

⁴⁶Previously, the free logic for individuals relied on a notion of logical existence that was based on identity, i.e. κ was considered to denote, just in case $\exists x x = \kappa$. While the new definition of logical existence is more primitive, i.e. it is formulated in terms of primitives of the language rather than defined identity, it now becomes a theorem that $\tau \downarrow \equiv \exists \beta \beta = \tau$ (see A.7.1448).

⁴⁷In higher-order object theory, however, it is possible to define existence of relations using higher-order exemplification as $\exists \mathcal{F} [\mathcal{F}]\Pi$.

The switch to a richer free logic also involved multiple changes to the axiom system ultimately resulting in the version given in section 3.3. The quantifier axioms were reformulated using the defined notion of $\tau\downarrow$ for all types. Furthermore, α - and β -conversion were restricted to denoting λ -expressions, the coexistence axiom was added and the base cases for denoting terms were adjusted. The coexistence axiom was based on a similar principle that was discovered as an artifactual theorem of the embedding of AOT at the time.⁴⁸ Recall its statement as:

$$[\lambda\nu_1\dots\nu_n \varphi\{\nu_1\dots\nu_n\}]\downarrow \& \Box\forall\nu_1\dots\forall\nu_n(\varphi\{\nu_1\dots\nu_n\} \equiv \psi\{\nu_1\dots\nu_n\}) \rightarrow [\lambda\nu_1\dots\nu_n \psi\{\nu_1\dots\nu_n\}]\downarrow$$

It is also referred to as *safe extension axiom*, since it merely asserts that a λ -expression with matrix ψ denotes, in case there provably is a denoting λ -expression with a matrix φ , s.t. both matrices are necessarily equivalent on all objects, i.e. in case the extension of the λ -expression is known to be safe. Consequently, the axiom has no impact on the size of models (or on consistency): a model can always choose the same denotation for $[\lambda\nu_1\dots\nu_n \psi\{\nu_1\dots\nu_n\}]$ as it chose for $[\lambda\nu_1\dots\nu_n \varphi\{\nu_1\dots\nu_n\}]$.⁴⁹

Initial versions of PLM that were equipped with a free logic on all types still retained the concept of *propositional formulas* (formulas without encoding subformulas), but dropped the implicit assumption that well-formed λ -expressions (i.e. λ -expressions with propositional matrix) generally denote, but instead excluded λ -expressions with matrices that contain definite descriptions from the base cases of axiomatically denoting terms.

The coexistence axiom allowed to safely derive that certain λ -expressions involving definite descriptions may still denote: Whenever it was possible to eliminate a description from the matrix of a λ -expression using a description-free propositional formula that is necessarily equivalent on all objects, it was safe to derive that the λ -expression denotes. However, due to the fact that no longer all λ -expressions with propositional matrix could be assumed to denote, the distinction between propositional and non-propositional formulas lost most of its relevance. Consequently, the next step was to simplify the system by replacing this syntactic distinction entirely by a restriction of the base cases of denoting terms, i.e. all λ -expressions became well-formed, but only λ -expressions without definite descriptions and without encoding formula subterms were asserted to denote by axiom.

The, at the time of writing, most recent extension of the set of axiomatically denoting λ -expression, which resulted in the formulation given in section 3.3, allowed us to derive necessary and sufficient conditions for λ -expressions to denote, as explained in section 3.8.2 below. A potential further refinement is discussed in section 4.6.1.

⁴⁸In particular $\exists G\Box\forall x_1\dots x_n(Gx_1\dots x_n \equiv \varphi\{x_1\dots x_n\}) \rightarrow \exists F(F = [\lambda x_1\dots x_n \varphi\{x_1\dots x_n\}])$.

⁴⁹However, note that this is not a requirement: While, by construction, both λ -expressions are necessarily equivalent under β -conversion, AOT does not require them to be identical.

3.8. Further Properties of AOT

3.8.1. Indistinguishable Abstract Objects

The issue raised in form of the Clark-Boolos Paradox and its variants in section 3.6 is a general issue of the comprehension principle for abstract object and their identity conditions, which intuitively imply that abstract objects correspond to sets of properties, together with the fact that abstract objects are simultaneously meant to themselves *exemplify* properties:

Properties have exemplification-extensions which are traditionally conceived of as sets of individuals.⁵⁰ However, if abstract objects correspond to sets of properties and exemplification-extensions of properties themselves correspond to sets of objects, one may wonder how this can be achieved consistently: How can abstract objects be sets of properties and simultaneously (in the simplest case of non-modal and extensional properties) elements of properties?

A semantic solution to this dilemma is given by Aczel models which are described in section 4.1.1. But there are also derivable theorems of AOT that serve to clarify how this dangling paradox may be avoided.

In particular, it is derivable that there are distinct, but exemplification-indistinguishable abstract objects (see A.7.8572):

$$\exists x \exists y (A!x \ \& \ A!y \ \& \ x \neq y \ \& \ \forall F ([F]x \equiv [F]y))$$

The comprehension principle for abstract objects requires the existence of sufficient abstract objects, that it has to be the case that some of them collapse under exemplification. In light of avoiding a violation of Cantor's Theorem one may even argue that *most* abstract objects are indistinguishable, since the cardinality of the set of abstract objects is strictly larger than the cardinality of the set of properties.⁵¹

Interestingly, though, it is impossible to independently construct two concrete abstract objects that provably fail to be distinguishable. This is also discussed in section 5.19 in the context of our proposed extended comprehension principle for relations among abstract objects. While PLM's proof of the theorem above (see A.7.8572) uses a slightly different construction, we can provide a proof that makes it explicit that we can construct an abstract object particularly in such a way that there has to be a distinct abstract object that is indistinguishable from it:

AOT-theorem $\langle \exists x \exists y (A!x \ \& \ A!y \ \& \ x \neq y \ \& \ \forall F ([F]x \equiv [F]y)) \rangle$

proof –

— Consider the object a that encodes being indistinguishable from any abstract object that does not encode being indistinguishable from itself.

⁵⁰In a modal setting properties are even associated with multiple sets of objects for different semantic possible worlds or, equivalently, extensions of modal properties are conceived of as mapping objects to sets of possible worlds in which the property is exemplified by the objects.

⁵¹And even the set of properties in turn has a strictly larger cardinality than the set of urelements, which in Aczel models will serve as proxies for abstract objects to define their exemplification behaviour, as described in more detail in section 4.1.1.

AOT-obtain a where a -prop:

$\langle A!a \ \& \ \forall F \ (a[F] \equiv \exists y \ (A!y \ \& \ F = [\lambda z \ \forall G \ ([G]z \equiv [G]y)] \ \& \ \neg y[\lambda z \ \forall G \ ([G]z \equiv [G]y)])) \rangle$
using A -objects[axiom-inst] $\exists E$ [rotated] **by fast**

— We show that a encodes being indistinguishable from itself using a proof by contradiction.

AOT-have 0 : $\langle a[\lambda z \ \forall G \ ([G]z \equiv [G]a)] \rangle$

proof (rule raa -cor:1)

AOT-assume 1 : $\langle \neg a[\lambda z \ \forall G \ ([G]z \equiv [G]a)] \rangle$

AOT-hence $\langle \neg \exists y \ (A!y \ \& \ [\lambda z \ \forall G \ ([G]z \equiv [G]a)] = [\lambda z \ \forall G \ ([G]z \equiv [G]y)] \ \& \ \neg y[\lambda z \ \forall G \ ([G]z \equiv [G]y)]) \rangle$

by (safe intro!: a -prop[THEN &E(2), THEN $\forall E(1)$, THEN $\equiv E(3)$] $cqt:2$)

AOT-hence $\langle \forall y \ \neg (A!y \ \& \ [\lambda z \ \forall G \ ([G]z \equiv [G]a)] = [\lambda z \ \forall G \ ([G]z \equiv [G]y)] \ \& \ \neg y[\lambda z \ \forall G \ ([G]z \equiv [G]y)]) \rangle$

using cqt -further:4[THEN $\rightarrow E$] **by blast**

AOT-hence $\langle \neg (A!a \ \& \ [\lambda z \ \forall G \ ([G]z \equiv [G]a)] = [\lambda z \ \forall G \ ([G]z \equiv [G]a)] \ \& \ \neg a[\lambda z \ \forall G \ ([G]z \equiv [G]a)]) \rangle$

using $\forall E(2)$ **by blast**

moreover AOT-have $\langle A!a \ \& \ [\lambda z \ \forall G \ ([G]z \equiv [G]a)] = [\lambda z \ \forall G \ ([G]z \equiv [G]a)] \ \& \ \neg a[\lambda z \ \forall G \ ([G]z \equiv [G]a)] \rangle$

by (safe intro!: a -prop[THEN &E(1)] &I rule=I:1 $cqt:2$ 1)

ultimately AOT-show $\langle p \ \& \ \neg p \rangle$ **for** p **using** $reductio$ -aa:1 **by blast**

qed

— Hence, by construction, there is an abstract object, s.t. being indistinguishable from it is identical to being indistinguishable from a , but which does not encode being indistinguishable from itself.

AOT-hence $\langle \exists y \ (A!y \ \& \ [\lambda z \ \forall G \ ([G]z \equiv [G]a)] = [\lambda z \ \forall G \ ([G]z \equiv [G]y)] \ \& \ \neg y[\lambda z \ \forall G \ ([G]z \equiv [G]y)]) \rangle$

by (safe intro!: a -prop[THEN &E(2), THEN $\forall E(1)$, THEN $\equiv E(1)$] $cqt:2$)

— Call this object b .

then AOT-obtain b where b -prop:

$\langle A!b \ \& \ [\lambda z \ \forall G \ ([G]z \equiv [G]a)] = [\lambda z \ \forall G \ ([G]z \equiv [G]b)] \ \& \ \neg b[\lambda z \ \forall G \ ([G]z \equiv [G]b)] \rangle$

using $\exists E$ [rotated] **by blast**

— Now a and b are indistinguishable.

AOT-have $\langle \forall G \ ([G]a \equiv [G]b) \rangle$

proof —

AOT-have $\langle [\lambda z \ \forall G \ ([G]z \equiv [G]a)]a \rangle$

by (safe intro!: β -C $cqt:2$ GEN $\equiv I$ $\rightarrow I$)

AOT-hence $\langle [\lambda z \ \forall G \ ([G]z \equiv [G]b)]a \rangle$

using b -prop[THEN &E(1), THEN &E(2)] rule=E **by fast**

thus ?thesis

using $\beta \rightarrow C$ **by blast**

qed

— But while a encodes being indistinguishable from b , b does not encode being indistinguishable from itself and therefore a is not identical to b .

moreover {

AOT-have $\langle a[\lambda z \ \forall G \ ([G]z \equiv [G]b)] \rangle$

using b -prop[THEN &E(1), THEN &E(2)] 0 rule=E **by fast**

AOT-hence $\langle a \neq b \rangle$

by (safe intro!: ab -obey:2[THEN $\rightarrow E$] $\vee I(1)$ $\exists I(1)$ [where $\tau = \langle \langle [\lambda z \ \forall G \ ([G]z \equiv [G]b)] \rangle \rangle$])

$\&I$ b -prop[*THEN* $\&E(2)$] *cqt:2*)

}

— Therefore, a and b are witnesses to the claim of the theorem.

ultimately AOT-have $\langle A!a \ \& \ A!b \ \& \ a \neq b \ \& \ \forall G([G]a \equiv [G]b) \rangle$

using $\&I$ a -prop[*THEN* $\&E(1)$] b -prop[*THEN* $\&E(1)$, *THEN* $\&E(1)$] **by** *blast*

AOT-hence $\langle \exists y(A!a \ \& \ A!y \ \& \ a \neq y \ \& \ \forall G([G]a \equiv [G]y)) \rangle$ **by** (*rule* $\exists I$)

AOT-thus $\langle \exists x \exists y(A!x \ \& \ A!y \ \& \ x \neq y \ \& \ \forall G([G]x \equiv [G]y)) \rangle$ **by** (*rule* $\exists I$)

qed

Notably, the existence of indistinguishable abstract objects can be used to prove that there is no general relation of identity in AOT, i.e. $[\lambda xy \ x = y]$ does not denote:

AOT-theorem $\langle \neg[\lambda xy \ x = y] \downarrow \rangle$

proof (*rule* *raa-cor:2*) — Proof by contradiction.

AOT-assume $0: \langle [\lambda xy \ x = y] \downarrow \rangle$

— Let a and b be witnesses to the theorem discussed above.

AOT-obtain $a \ b$ **where** $1: \langle A!a \ \& \ A!b \ \& \ a \neq b \ \& \ \forall F([F]a \equiv [F]b) \rangle$

using *aclassical2* $\exists E$ [*rotated*] **by** *blast*

— From our assumption and the fact that a is self-identical, it follows that a exemplifies the projection of the identity relation to a .

moreover AOT-have $\langle [\lambda x \ [\lambda xy \ x = y]ax]a \rangle$

by (*safe intro!*: $0 \ \beta \leftarrow C$ *cqt:2* *tuple-denotes*[*THEN* $\equiv_{df} I$] $\&I = I$)

— Since a and b are indistinguishable, b has to exemplify this property as well.

ultimately AOT-have $\langle [\lambda x \ [\lambda xy \ x = y]ax]b \rangle$

by (*safe intro!*: 1 [*THEN* $\&E(2)$, *THEN* $\forall E(1)$, *THEN* $\equiv E(1)$] 0 *cqt:2*)

— Which by beta-conversion yields that a is identical to b .

AOT-hence $\langle a = b \rangle$

by (*safe dest!*: $\beta \rightarrow C$)

— Which contradicts the fact that a and b are distinct by construction.

AOT-thus $\langle p \ \& \ \neg p \rangle$ **for** p

using $1 \ \&E = -infix$ [*THEN* $\equiv_{df} E$] *reductio-aa:1* **by** *blast*

qed

This aspect of AOT will be of notable importance during the construction of natural numbers in chapter 5. In the following section, we will see another prominent example of a theorem of AOT that involves indistinguishable objects and relates to Aczel models.

3.8.2. Necessary and Sufficient Condition for Relations to Denote

The move to a free logic for relation terms and the iterative extension of the base cases of denoting terms mentioned in section 3.7, ultimately allowed us to contribute the following theorem to AOT:

$$[\lambda x \ \varphi\{x\}] \downarrow \equiv \Box \forall x \forall y (\forall F ([F]x \equiv [F]y) \rightarrow (\varphi\{x\} \equiv \varphi\{y\}))$$

A λ -expression denotes, if and only if necessarily its matrix agrees on all indistinguishable objects.

The proof (see A.7.8603) relies on the fact that under the assumption of the right-hand-side, it follows that $\Box \forall y (\exists x (\forall F ([F]x \equiv [F]y) \ \& \ \varphi\{x\} \equiv \varphi\{y\}))$. Now since

$[\lambda y \exists x (\forall F ([F]x \equiv [F]y) \& \varphi\{x\})] \downarrow$ by axiom (by construction the initial λ does not bind a variable that occurs in an encoding formula subterm - in particular it occurs only in the exemplification formula $[F]y$), $[\lambda x \varphi\{x\}] \downarrow$ follows by the coexistence axiom. The left-to-right direction can be shown by instantiating F to $[\lambda x \varphi\{x\}]$ and some modal reasoning.

This theorem has several repercussions. It provides the analytic means to judge whether a λ -expression denotes within the system of AOT itself. Notably, this led to a proof of the existence of world-relative relations and thereby of rigidifying relations, as discussed in more detail in the next section.

Furthermore, it can contribute to a potential reformulation of the construction of natural numbers that does not require a modal axiom that generates ordinary objects. This is mentioned in section 5.21, although at the time of writing, the analysis of this potential change is not yet complete, so the current version of PLM at the time of writing does not yet contain this new enhanced construction.

In general, this theorem is a prime example of the benefits of the semantic analysis of AOT using our embedding that has led to significant theoretical improvements of AOT and may yet allow for further improvements in the future.

Semantically, the theorem is closely related to Aczel models of AOT. The condition of being indistinguishable, $\forall F ([F]x \equiv [F]y)$, semantically corresponds to x and y sharing the same *urelement*. Consequently, the theorem states that λ -expressions denote, if their matrix agrees on objects with the same urelements or, in other words, if they can be represented as functions acting on urelements. A more detailed semantic discussion and a precise construction involving a mapping from individuals to urelements and relations modelled as proposition-valued functions acting on these urelements can be found in chapter 4.

3.8.3. World-Relative Relations and Rigidifying Relations

A notable consequence of the theory of possible worlds outlined in section 3.5.2 and the necessary and sufficient conditions for relations to denote described in the previous section is the fact that world-relative relations denote.

In particular, it can be derived that any denoting λ -expression can be relativized to a possible world, i.e. (see A.11.2920 and A.11.2953):

$$\begin{aligned} [\lambda x \varphi\{x\}] \downarrow &\rightarrow [\lambda x w \models \varphi\{x\}] \downarrow \\ [\lambda x_1 \dots x_n \varphi\{x_1 \dots x_n\}] \downarrow &\rightarrow [\lambda x_1 \dots x_n w \models \varphi\{x_1 \dots x_n\}] \downarrow \end{aligned}$$

This allows for a definition of world-relative relations as follows (see A.11.2992):

$$F_w =_{df} [\lambda x_1 \dots x_n w \models [F]x_1 \dots x_n]$$

Notably, it becomes a theorem that there exist *rigidifying relations*.⁵²

⁵²Zalta refers to [21], in which Daniel Gallin postulates the existence of rigidifying relations as an axiom.

A relation is *rigid*, if exemplifying it is modally collapsed (see A.11.2995):

$$\text{Rigid}(F) \equiv \Box \forall x_1 \dots \forall x_n ([F]x_1 \dots x_n \rightarrow \Box [F]x_1 \dots x_n)$$

And a relation F *rigidifies* a relation G , just in case F is rigid and exemplifying it is equivalent to exemplifying G (see A.11.2999):

$$\text{Rigidifies}(F, G) \equiv_{df} \text{Rigid}(F) \ \& \ \forall x_1 \dots \forall x_n ([F]x_1 \dots x_n \equiv [G]x_1 \dots x_n)$$

World-relative relations can now be used as a witness to show that there exist rigidifying relations (see A.11.3057):

$$\exists F \text{Rigidifies}(F, G)$$

Rigidifying relations will play an important role in the construction of natural numbers described in chapter 5 and their existence previously had to be ensured by stating this last theorem as axiom.

3.8.4. Sixteen Distinct Properties

Another result that can be traced back directly to the construction of the embedding is the derivation of a more refined theorem about minimal models of AOT. While previously PLM merely derived that there are six distinct properties, it is a natural consequence of our constructed models that there are at least sixteen distinct properties. This is due to the fact that there need to be at least two distinguishable individuals (discerned by being ordinary and being abstract) and two possible worlds (required by the axiom asserting the existence of a contingently non-concrete objects object as mentioned in 3.3). Two possible worlds imply that there are at least $2^2 = 4$ distinct propositions. Mapping two discernible objects to 4 propositions can be done in $2^4 = 16$ distinct ways.

And indeed we could construct a proof in the system of AOT itself that verifies that this is not a mere artifact of the model construction, but a proper theorem in AOT. See A.7.6909 for a detailed (though somewhat tedious) proof.

Notably, this result also implies that there is at least $2^{16} = 65536$ distinct abstract objects in minimal models of AOT. On the other hand, models that validate the theory of natural numbers described in chapter 5 involve at least countably infinitely many ordinary objects⁵³ and thereby uncountably many properties and abstract objects.

Before we proceed to discuss AOT's analysis of natural numbers in chapter 5, we describe the technical details of the implementation of AOT in our embedding in the next chapter.

⁵³At least in the current construction. A potential future version of the construction mentioned in section 5.21 may instead require at least countably infinitely many *special urelements*, but not ordinary objects.

4. SSE of AOT in Isabelle/HOL

4.1. Model Construction

While the precise model construction of the embedding can be found in A.1, this section provides a high-level description of this construction. The general idea is based on Aczel models of AOT, which are extended to accommodate for AOT's hyperintensional modal logic on the one hand and its free logic for individual and relation terms on the other hand. Furthermore, we use a system of type classes to construct relations of arbitrary arity as relations among tuples of individuals.¹

Recall that, as mentioned in section 2.6, we do not construct set-theoretic models of AOT, but instead construct models of AOT in HOL, while any set-theoretic model of HOL that validates our construction can be lifted to a set-theoretic model of AOT.

4.1.1. Aczel Models

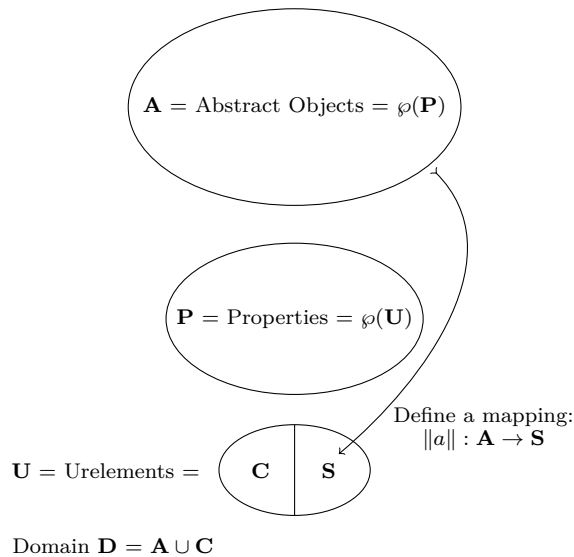


Figure 4.1.: Extensional, non-modal Aczel model of AOT.

¹However, for each fixed arity of relations the type classes can be logically eliminated.

The general structure of our models is based on Aczel models (see [60]). Aczel models are extensional models that validate both the comprehension principle for abstract objects² and classical relation comprehension in the absence of encoding formulas.

Aczel models involve a domain of *urelements* U that is split into *ordinary urelements* C and *special urelements* S . In the extensional, non-modal setting, the power set of the set of urelements suffices for representing properties. Abstract objects in turn are modelled using the power set of properties.

Furthermore, the models involve a (non-injective) mapping from abstract objects to special urelements. The special urelement $||x||$ to which an abstract object x is mapped determines which properties the abstract object x *exemplifies*.

The domain of individuals D is defined as the union of abstract objects and ordinary urelements (resp. ordinary objects).

Any individual $x \in D$ can be associated with an urelement $|x| \in U$:

$$|x| = \begin{cases} x, & \text{if } x \in C \\ ||x||, & \text{if } x \in A \end{cases}$$

Based on this construction the truth conditions for AOT's atomic formulas, i.e. encoding and exemplification, can be defined as follows:

- An object x *exemplifies* a property F , just in case that $|x| \in F$.
- An object x *encodes* a property F , just in case $x \in A$ and $F \in x$.

This construction immediately validates both the identity conditions for abstract objects and the comprehension principle of abstract objects:

- Two abstract objects are identical, if they encode the same properties.
- For every set of properties, there is an abstract object that encodes exactly those properties in the set.

Furthermore, Aczel models validate a restricted version of relation comprehension. Since the truth conditions of any exemplification formula solely depend on the urelement associated with the exemplifying individual, any condition φ on individuals that does not contain encoding claims can equivalently be represented as a condition on urelements. Therefore, for any such condition φ , there exists a relation F that is exemplified by exactly those objects that satisfy φ : $\exists F \forall x (|x| \in F \equiv \varphi\{x\})$, given that φ does not involve encoding claims.

While Aczel models generally demonstrate that abstract objects and encoding can be modelled without being subject to the Clark-Boolos paradox (recall 3.6.1), there are several issues that remain unaddressed, including:

- AOT's relations are not extensional and not even merely intensional, but fully hyperintensional.

²The last axiom in section 3.3, resp. A.6.200 in the embedding: $\exists x (A!x \ \& \ \forall F (x[F] \equiv \varphi\{F\}))$

- Complex individual and relation terms and the free logic of AOT are not modelled explicitly.
- Relation comprehension for formulas in the absence of encoding formulas does not immediately cover all the base cases of axiomatically denoting relation terms as mentioned in section 3.3.
- Aczel models do not cover n -ary relations for $n \geq 2$.³

Therefore, while the models used for our embedding inherit the idea of urelements and a mapping from abstract objects to special urelements, we extend the general model structure for our embedding.

4.1.2. Types of the Embedding

The terms of AOT are represented in our embedding using the following types in the meta-logic:⁴

- Type \circ for formulas, resp. propositions.
- Type κ for individual terms.
- Type $\langle 'a \rangle$ for relation terms. Here $'a$ is a type variable that is restricted to types of class $AOT\text{-}\kappa s$, which is instantiated for κ (yielding unary relations, resp. properties, as $\langle \kappa \rangle$) and arbitrary tuples of type κ (i.e. $\langle \kappa \times \kappa \rangle$ is used to represent two-place-relations, etc.).⁵

In the following, we will briefly explain how each of these types is constructed.⁶ The language elements of AOT (i.e. atomic formulas, logical connectives, quantifiers, complex individual and relation terms) can then be represented by introducing constants that act on objects of these types. We will introduce a custom sub-grammar in the inner syntax of Isabelle/HOL that approximates AOT's syntax and translates to terms involving these constants (as outlined in section 2.7). We will then formulate specifications for the constants that will allow us to derive the axiom system and deduction rules of AOT. The construction of the types will ensure that there are suitable witnesses for these specifications.

The type class $AOT\text{-}Term$ (see A.1.463) is used as a common type class that is instantiated for each of the types above. It involves a single parameter $AOT\text{-}model\text{-}denotes$ of type $'a \Rightarrow bool$, that determines the meta-logical conditions under which a term of type $'a$ denotes. We will explain how $AOT\text{-}model\text{-}denotes$ is instantiated for each type below.

³While in an extensional setting they can be interpreted as sets of tuples of urelements, validating AOT's definition of relation identity in a hyperintensional context requires further care.

⁴Note that types and objects have separate namespaces in Isabelle. Also recall the brief introduction to type classes in section 2.5.2.

⁵Technically, $AOT\text{-}\kappa s$ is instantiated for products of a type of class $AOT\text{-}\kappa$ and a type of $AOT\text{-}\kappa s$, while $AOT\text{-}\kappa$ abstracts the properties of type κ (and is only instantiated for κ).

⁶This will involve introducing additional types, in particular for urelements, that will not be used for representing terms of AOT directly, but merely to construct the types above.

The additional type $'a$ *AOT-var* is defined for each type $'a$ of class *AOT-Term* using the objects of type $'a$, for which *AOT-model-denotes* is *True*, as representation set (see A.1.1240).⁷ This type is used to represent the variables of each of the types above, e.g. κ *AOT-var* will be the type of individual variables. Thereby, variables range exactly over the denoting objects at each type. To be used in place of terms, a variable of type $'a$ *AOT-var* is mapped to its representation type $'a$ using the constant *AOT-term-of-var*.⁸

4.1.3. Hyperintensional Propositions

The hyperintensionality of AOT is modelled at the level of propositions. The construction follows the general method outlined in section 2.5.1.

The type \circ is introduced as a primitive type (see A.1.12). It is used to represent hyperintensional propositions and is associated with modal extensions following Kripke semantics: a primitive type w for semantic possible worlds is introduced (see A.1.20) and it is axiomatized that there be a surjective mapping *AOT-model-do* from propositions of type \circ to Montague intensions, i.e. boolean valued functions on possible worlds (type $w \Rightarrow \text{bool}$; see A.1.21).

We define for a proposition φ of type \circ to be valid in a given semantic possible world v (written $[v \models \varphi]$)⁹, just in case *AOT-model-do* maps p to a Montague intension that evaluates to *True* for v (see A.1.30).

This way, our type of propositions \circ is assured to contain a proposition for each Montague intension, but does not require the collapse of necessarily equivalent propositions:

For any given Montague intension φ , the inverse of *AOT-model-do* yields a proposition of type \circ that is valid in exactly those worlds for which φ evaluates to *True* (see A.1.36).¹⁰

However, the construction allows for the type \circ to contain more propositions than there are Montague intensions. I.e. there may be two distinct objects p and q of type \circ that are necessarily equivalent, i.e. they are valid in the same semantic possible worlds. This can be confirmed by **nitpick**:

lemma $\langle \forall v . [v \models p] \longleftrightarrow [v \models q] \rangle$ **and** $\langle p \neq q \rangle$
nitpick[*satisfy, user-axioms, expect=genuine*]

⁷Since the representation set of a type in Isabelle/HOL cannot be empty, the type class *AOT-Term* involves the assumption that there is an object for which *AOT-model-denotes* is *True*, which has to be proven for each instantiation of the type class and thereby can be assumed for each type of class *AOT-Term*.

⁸Each **typedef** that defines a type using a representation set automatically introduces morphisms, usually prefixed with *Rep-* and *Abs-*, that map objects of the defined type to objects of its representation set and vice-versa. In this case we chose the custom names *AOT-term-of-var* and *AOT-var-of-term* instead of *Rep-AOT-var* and *Abs-AOT-var*.

⁹Note that this use of the double turnstile symbol \models is defined within the meta-logic HOL and distinct from the use in AOT's possible world theory described in section 3.5.2.

¹⁰This fact relies on the surjectivity of *AOT-model-do*. The embedding introduces the notation $\varepsilon_{\circ} w . \varphi w$ for the proposition given by *inv AOT-model-do* φ . We will use such propositions during witness proofs in specifications.

nitpick can find a model in which p and q are represented by two distinct objects, while both of them have the same Montague intension under *AOT-model-do*.

Note, however, that the construction also *allows* for necessary equivalent propositions to be collapsed:

lemma $\langle \forall p q . (\forall v . [v \models p] \leftrightarrow [v \models q]) \longrightarrow p = q \rangle$
nitpick[*satisfy, user-axioms, expect=genuine*]

In this case **nitpick** chooses a model in which the type `o` is isomorphic to the type of Montague intensions $w \Rightarrow \text{bool}$, i.e. there are just as many objects of type `o` as there are Montague intensions.

Just as AOT itself, the model construction does not presuppose the degree of hyperintensionality of propositions.

To instantiate the type class *AOT-Term* for type `o`, we need to define the conditions under which propositions denote. Since in AOT all formulas denote, *AOT-model-denotes* is *True* for all objects of type `o`. Consequently, the type `o` *AOT-var* is isomorphic to the type `o`.

On top of this hyperintensional type of propositions, the logical connectives will later be defined by **specification** as outlined in section 2.5.1.

Notably, previous versions of our embedding (in particular the construction in [29]), modelled hyperintensionality more explicitly by using an additional primitive type s of *intensional states*. Propositions were modelled explicitly as boolean-valued functions acting on states and possible worlds (type $s \Rightarrow w \Rightarrow \text{bool}$). Semantic validity was defined using the evaluation of propositions in a designated *actual* state. The logical connectives were defined to have classical behaviour in the actual state, while their behaviour was left unspecified in non-actual states. While such an explicit construction using intensional states can still serve as a concrete model for our abstract type `o`, the fact that AOT does not presuppose any additional structure on non-actual states allowed us to replace the explicit construction by the more general abstraction described above.

4.1.4. Extended Aczel Model Structure

Our representation is based on Aczel models, so the construction of the types of individuals and relations relies on urelements.

The embedding introduces a type of urelements v (see A.1.57) that is comprised of three separate kinds of urelements:

- Ordinary urelements of type ω (see A.1.45),
- Special urelements of type σ (see A.1.53) and
- Null-urelements of type *null* (see A.1.55).

Following the structure of Aczel models, ordinary urelements are used to model ordinary objects and special urelements determine the exemplification behaviour of abstract objects. The additional null-urelements are introduced to be able to distinguish between non-denoting individual terms (see below).

For simple models, the types of ordinary, special and null urelements can all remain purely abstract types.¹¹

Hyperintensional relations are modelled as proposition-valued functions. In particular, the embedding introduces the type *urrel* (see A.1.63) that is represented by the set of all functions from urelements to propositions (type $v \Rightarrow o$), which map null-urelements to necessarily false propositions.¹² This type of *urrelations* will be in one-to-one correspondence with the denoting property terms, i.e. denoting objects of type $\langle \kappa \rangle$, respectively objects of type $\langle \kappa \rangle$ *AOT-var*.

The additional null-urelements serve to avoid two kinds of artifactual theorems:

- Let p be the proposition denoted by the term $[F] \iota x \varphi \{x\}$ and let q be the proposition denoted by the term $[F] \iota x \psi \{x\}$. Furthermore, assume that provably neither of the descriptions denote, i.e. both $\neg \iota x \varphi \{x\} \downarrow$ and $\neg \iota x \psi \{x\} \downarrow$ are theorems. Now while AOT requires p and q to be necessarily equivalent, in particular they are both necessarily false, it does not (in general) presuppose that p is *identical* to q .¹³ In the embedding this is achieved by allowing descriptions (with distinct matrices) to be mapped to distinct null-urelements to which the urrelation corresponding to F can assign distinct (albeit necessarily false) propositions.¹⁴ While artifactual theorems of this kind could also be avoided by merely allowing exemplification formulas to choose distinct propositions for distinct non-denoting terms, this would not be sufficient to avoid the second kind of artifactual theorems:
- In AOT there may be distinct properties, s.t. for any object exemplifying either of them necessarily results in the same proposition. I.e. $\forall x \Box ([F]x = [G]x)$ does *not* imply $F = G$. The \forall -quantifier ranges over all denoting individuals. If relations were merely modelled as functions from urelements that correspond to denoting individual terms to propositions, the identity would follow, since two functions are identical, if they agree on all arguments. By introducing null-urelements, however, we allow F and G to vary on additional urelements outside of the range of the quantifier.¹⁵

¹¹I.e. a model of HOL may choose (non-empty) domains of any size for each kind of urelements. In chapter 5 we will discuss a more specific construction that is required to validate the additional axioms needed for the construction of natural numbers.

¹²Note that our construction allows for multiple distinct propositions that are necessarily false.

¹³An example of an exception is the case in which the matrices are alphabetic variants of each other or can be transformed into each other by substituting identical subterms, in which case φ and ψ are also meta-logically identical.

¹⁴Note that this is not a mere technicality, but it may be desirable to distinguish e.g. between the proposition *The number smaller than 3 is a natural number* (which fails due to there not being a unique such number) and *The number greater than 3 and smaller than 2 is a natural number* (which fails due to there not being any such number). Furthermore, it might make sense to consider the proposition *The present king of France is a natural number* to be an entirely different proposition than the first two. The embedding allows to assign each of the non-denoting definite descriptions distinct null-urelements and thereby allows the propositions to differ. However, it also allows to choose a model with only a single null-urelement which would collapse these propositions.

¹⁵An alternative approach would be to introduce a primitive type of relations that is merely assigned a proposition-valued function as extension, similarly to how Montague intensions are assigned to the

Note that the additional null-urelements have no impact on minimal models of AOT. In minimal models, propositions are in one-to-one correspondence to Montague intensions: for every boolean valued function on possible worlds there is exactly one proposition. While urrelations have to assign propositions to null-urelements, by construction, urrelations are required to evaluate to necessarily false propositions on null-urelements. Hence, there is only one choice for doing so, namely the single proposition with the constant-false function as Montague intensions. Consequently, the number of relations in minimal models of AOT is unaffected.

As a last ingredient of our Aczel model structure, we require a mapping $\alpha\sigma$ from sets of urrelations (which will be used to represent abstract objects) to special urelements (see A.1.235). As in the basic Aczel model construction, this mapping will determine the exemplification behavior of abstract objects.

For urrelations to become a proper quotient of proposition-valued functions acting on individual *terms*, as described below, we require this mapping to be surjective. However, we can show that any mapping $\alpha\sigma'$ from sets of urrelations to special urelements can be extended to a surjective mapping $\alpha\sigma$ that distinguishes all abstract objects that are distinguished by $\alpha\sigma'$, i.e. if $\alpha\sigma' x \neq \alpha\sigma' y$, then $\alpha\sigma x \neq \alpha\sigma y$. This is possible due to the fact that the set of abstract objects is significantly larger than the set of special urelements. In particular, under any arbitrary mapping from abstract objects to special urelements, there has to be at least one abstract object a that shares the same urelement with an amount of other abstract objects that is larger than the total amount of special urelements (proof by a pigeonhole-style argument, see A.1.73). Therefore, any mapping $\alpha\sigma'$ that is not surjective, can be extended to a surjective mapping by further differentiating the abstract objects that share their urelements with a .

To keep the construction as flexible as possible, we first introduce an uninterpreted constant $\alpha\sigma'$ and then generically extend it to a surjective mapping $\alpha\sigma$ (see A.1.234).

To validate extended relation comprehension we can then augment $\alpha\sigma'$ using a suitable **specification**. The precise construction of $\alpha\sigma'$ needed for extended relation comprehension is discussed in more detail in section 5.19.

Additionally, we introduce the constant *AOT-model-concretew* (see A.1.452) and specify it in such a way, that (1) for every object x (of type ω) there is a possible world w (of type w), s.t. *AOT-model-concretew* $x w$ and (2) there is an object x and a possible world w , s.t. *AOT-model-concretew* $x w \wedge \neg$ *AOT-model-concretew* $x w_0$ (where w_0 is the designated actual world). This constant will be used to construct AOT's relation of being concrete. The specified properties ensure that objects of type ω will be possibly concrete, i.e. ordinary, and that there possibly is an object that is concrete, but not actually concrete, which is asserted by AOT as an axiom. A function that is true for an object x (of type ω) and

primitive type of propositions. However, this would require a polymorphic axiomatization to account for relations of all arities which is incompatible with the model-checking tool **nitpick**. Even only axiomatizing a finite subset of all arities would require **nitpick** to construct significantly larger models and thereby diminish its usefulness. Furthermore, this construction would further complicate validating the definition of n -ary relation identity.

a semantic possible world w (of type w), just in case w is not the actual world w_0 , can serve as witness for the specification.¹⁶

Based on the type of urelements v and the type of urelations $urrel$ we can construct the type κ of individual terms.

4.1.5. Individual Terms and Properties

The type κ (see A.1.430) consists of ordinary objects of type ω (shared with ordinary urelements), abstract objects modelled as sets of urelements (type *urrel set*) and null-objects of type *null* (shared with null-urelements) that will serve to model non-denoting definite descriptions. We can lift the surjective mapping from abstract objects to special urelements $\alpha\sigma$ to a surjective mapping κv from individual terms to urelements (i.e. type $\kappa \Rightarrow v$) (see A.1.434), s.t. for any urelement we can find an object of type κ that is mapped to that urelement (see A.1.439).

To instantiate the type class *AOT-Term* for type κ , we define *AOT-model-denotes* to be *True* for exactly those objects of type κ that are not null-objects.

Relation terms will be defined relative to types of a type class that abstracts individuals and tuples of individuals. We will explain this generic construction below. However, it may be helpful to first consider the case of properties (i.e. type $\langle \kappa \rangle$) specifically, even though in the embedding this case will only occur as a special case of the generic construction.

Property terms (of type $\langle \kappa \rangle$) are represented by proposition-valued functions acting on individuals (type $\kappa \Rightarrow o$). A property term *denotes*, if its representing function φ satisfies the following conditions:

- $\varphi \kappa = \varphi \kappa'$, whenever $\kappa v \kappa = \kappa v \kappa'$, i.e. φ evaluates to the same propositions for objects that have the same urelements.
- φ evaluates to necessarily false propositions for objects of type κ that do not denote.

Consequently, since κv is surjective and urelations have the property to be necessarily unexemplified on null-urelements, denoting property terms are in one-to-one correspondence with urelations (see A.1.691). This is crucial for constructing encoding and validating the comprehension principle of abstract objects, since abstract objects are modelled as sets of urelations.

We can now now construct a function that can later serve as witness for our specification of exemplification. For a property term Π and an individual term κ , we can choose a proposition p , such that:

¹⁶Note that we have to assert the existence of a non-actual world using a meta-logical axiom, see A.1.26. Also note that this construction does not imply that in our embedding no objects will be actually concrete and all ordinary objects will be concrete in all non-actual worlds. While our witness has this additional property, a model of HOL may choose any denotation for *AOT-model-concretew* that merely satisfies the properties of the specification.

- If Π denotes, then $p = \text{Rep-rel } \Pi \ \kappa$, i.e. the proposition resulting from applying the function representing Π to κ . This proposition will, by construction, be necessarily false, if κ does not denote.
- p is a necessarily false proposition otherwise.

Furthermore, the construction allows us to define the meta-logical truth conditions of encoding as follows: κ encodes Π just in case that (1) Π denotes, (2) κ is represented by an abstract object x and (3) the urrelation corresponding to Π is contained in x .

4.1.6. Type Classes for Individual Terms

The type class $AOT\text{-}\kappa s$ is a combination of three more specific type classes:

$AOT\text{-}IndividualTerm$ (see A.1.510), $AOT\text{-}RelationProjection$ (see A.4.407) and $AOT\text{-}Enc$ (see A.4.714). The latter two formulate conditions on relations among objects of their type variable. Therefore, they can only be formulated after a type of relations is introduced. The type of relations itself will be defined relative to the class $AOT\text{-}IndividualTerm$.

The most important parameter of this class is $AOT\text{-}model\text{-}term\text{-}equiv$, an equivalence relation which is satisfied for two objects, if they have common urelements.¹⁷ We furthermore introduce the notion of individual terms to be *regular* and specify a transformation of proposition-valued functions acting on individual terms, s.t. after the transformation the behaviour of the function is solely determined by its values on regular terms. This will be relevant for the definition of n -ary relation identity (see 4.6.4). An unary individual term (i.e. an object of type κ) is always regular, while a tuple will only be regular, if at most one of its elements does not denote.

In the next section, we will introduce relations as proposition-valued functions acting on objects of sort $AOT\text{-}IndividualTerm$. The class $AOT\text{-}RelationProjection$ defines an abstract notion of projections of relations that will be relevant for defining n -ary relation identity. The class $AOT\text{-}Enc$ defines an abstract notion of encoding. Encoding for type κ is specified as explained in the last section, while for tuples it is constructed in such a way that the axiom of n -ary encoding will become derivable. Together, the three type classes form the class $AOT\text{-}\kappa s$.

In the formulation of the axiom system, individuals in ellipses notation will be allowed to have any type of class $AOT\text{-}\kappa s$, and relations will be assumed to act on any type of class $AOT\text{-}\kappa s$.¹⁸ This way axioms about relations can be stated for all arities at the same time (since the concrete type of individuals κ as well as arbitrary iterated products of it, e.g. $\kappa \times \kappa \times \kappa$, are all of class $AOT\text{-}\kappa s$).

¹⁷Note that an *object* of a type of class $AOT\text{-}IndividualTerm$ may itself e.g. be a pair of two objects of type κ , since the product of κ with itself, i.e. type $\kappa \times \kappa$, is also of class $AOT\text{-}IndividualTerm$. $AOT\text{-}model\text{-}term\text{-}equiv$ for pairs is defined as the conjunction of $AOT\text{-}model\text{-}term\text{-}equiv$ on both projections. Consequently, two tuples $(\kappa_1, \dots, \kappa_n)$ and $(\kappa_1', \dots, \kappa_n')$ of objects of type κ are $AOT\text{-}model\text{-}term\text{-}equiv$ -equivalent if for all $1 \leq i \leq n$, κ_i has the same urelement as κ_i' .

¹⁸Unless a statement involves explicit exemplification or encoding formulas that imply restrictions on the type, e.g. a particular arity.

4.1.7. Generic Relation Terms

The generic type of relation terms is defined as the type of proposition-valued functions acting on a type of class *AOT-IndividualTerm* (see A.1.582).

To instantiate the type class *AOT-Term* to our generic type of relation terms, we have to define the conditions under which a relation term denotes.

A relation term denotes, if it is represented by a proposition-valued functions φ on individual terms, such that (see A.1.606):

- φ agrees on *AOT-model-term-equiv*-equivalent terms, i.e. it evaluates to the same proposition for individual terms that share the same urelements.
- For non-denoting individual terms, φ evaluates to necessarily false propositions.
- φ is well-behaved on irregular terms (i.e. on irregular terms it evaluates to the proposition given by *AOT-model-irregular* φ , which solely depends on φ 's behaviour on regular terms). This will be important to validate the definition of n -ary relation identity and is discussed in section 4.6.4. Note that since unary individual terms, i.e. objects of type κ , are always regular, this restriction does not apply to properties of type $\langle \kappa \rangle$.

Consequently, exemplification of denoting relation terms, can (as already indicated for the unary case) simply be modelled by the application of the proposition-valued function representing the relation term to the given individual term (which may be a tuple of terms of type κ), while exemplifying non-denoting relation terms yields a necessarily false proposition.¹⁹

Generic encoding was already described in the last section.

We now have constructed all the required types and prepared the required witnesses for constructing an abstract semantics of AOT using specifications in section 4.4. However, this semantics is formulated using our implementation of AOT's syntax, so in the following two sections we will first briefly discuss how we extend Isabelle's inner syntax by an approximation of the syntax used in PLM and how we extend Isabelle's outer syntax by custom commands used for structured reasoning in the embedding.

4.2. Syntax of the Target Theory

We already discussed the possibility of extending Isabelle's inner syntax in general in section 2.7. Following the method described in that section, we introduce *AOT-prop* as syntactic root type for propositions in AOT (see A.2) and define a custom grammar for AOT on top of it (see A.3). However, Isabelle's high-level mechanisms for defining custom syntax have certain limitations that make an accurate representation of AOT's syntax challenging.

¹⁹By *can be modelled* here we mean that we can construct a witness for the semantic specification of exemplification.

In particular, Isabelle’s lexical analysis is not designed to be configurable. It presupposes that identifiers consist of multiple characters and have to be delimited by whitespace or certain delimiter tokens.

While requiring identifiers to be delimited can be considered as a reasonable syntactic concession, we found that reproducing the compact form of atomic formulas used in PLM results in significantly improved readability.

Therefore we utilize Isabelle’s low-level mechanisms to customize syntax by providing transformations on its abstract syntax tree and its term representation written in Standard ML.

In particular, we use **parse-ast-translations** and **parse-translations** (see §8.4 in [55]) to split what Isabelle would natively regard as a single identifier. That way we are able to e.g. translate the term $[\Pi]_{\kappa\kappa'}$ to $AOT-exe \Pi (\kappa, \kappa')$. The 2-ary exemplification formula is translated to an application of the constant $AOT-exe$ to the relation term and a tuple of individual terms. Similarly, $\kappa\kappa'[\Pi]$ is translated to $AOT-enc (\kappa, \kappa') \Pi$. Involved constants are introduced in A.3 as uninterpreted constants (see A.3.41), which are only later enriched with semantic structure using **specifications** (see A.4 and section 4.4).²⁰

Furthermore, PLM associates the symbols used for its terms with their types, as described in section 3.2. While it is possible to rely on Isabelle’s type inference in most cases, this will not always result in correctly typed terms without additional type annotations which would negatively affect readability.

For that reason, we construct an extensible system for typing terms based on their names. In particular we introduce the command **AOT-register-type-constraints** that can be used to introduce named categories of types and equip them with type constraints both for unary terms and tuples. We then allow registering symbols as variables and meta-variables of a given category with **AOT-register-variable-names** and **AOT-register-metavariablename**. The extensible design allows for reproducing AOT’s concept of *restricted variables* (see 3.4.4) by further associating a term category with a restriction condition (see A.9).²¹

A danger in the extensive use of complex custom syntax is silent errors in the syntactic translations that could result in an expression to be parsed contrary to their intended meaning. To alleviate this danger we define multiple *printing modes*. The embedding can be configured to print terms in an approximation of AOT’s syntax, e.g.:

$$[\Pi]\kappa y \rightarrow p \vee \varphi$$

using *meta-syntax*, an enriched version of HOL’s syntax without complex transformations, e.g.:

$$([\Pi, (\kappa, \langle y \rangle)]) \rightarrow \langle p \rangle \vee \varphi$$

²⁰The type construction discussed in the previous section allows us to construct witnesses for these specifications.

²¹The restriction condition will be added when parsing quantifiers using restricted variables. For rigidly restricted variables a sub-type is introduced that is restricted to all terms that satisfy the restriction condition, allowing to add the restriction condition as axiom for objects of this restricted type.

or as plain HOL terms without any syntactic sugar, e.g.:

$$AOT\text{-imp } (AOT\text{-exe } \Pi (\kappa, AOT\text{-term-of-var } y)) (AOT\text{-disj } (AOT\text{-term-of-var } p) \varphi)$$

Note that while the meta-syntax already involves distracting complexities like the annotation of non-meta-variables using $\langle - \rangle$, additional explicit syntax for exemplification $(\lfloor -, \rfloor)$ and explicit tuples, plain HOL syntax quickly becomes unreadable for complex terms.

For the purpose of implementing a full theory with an extensive body of theorems, we contend that the improved readability outweighs the potential danger of complex syntax transformations, especially given the ability to confirm the accuracy of the translation using less complex printing modes.

4.3. Extending Isabelle’s Outer Syntax

While the syntax transformations described in the last section go a long way in allowing the intuitive statement of terms and formulas of AOT, *reasoning* in the target logic entails additional challenges.

For example, reasoning in the embedding involves keeping track of the semantic possible world in which statements are valid. To avoid this cognitive overhead, we implement a copy of Isabelle’s Isar language in Standard ML that automatically handles semantic possible worlds and allows theorem statements and proofs to be transferred directly from and to PLM without the need of explicitly mentioning semantic possible worlds.

While modally-strict theorems of PLM are valid in all semantic possible worlds, conceptually its proofs work relative to an arbitrary but fixed world. For proving a necessary fact during a proof, e.g. $\Box\varphi$, PLM often reasons by providing a *modally-strict* sub-proof of φ and appealing to the rule RN. In our embedding we reproduce this by introducing an outer syntax command **AOT-modally-strict** $\{$ that opens a block of reasoning relative to a fresh possible world. For example:

```

AOT-theorem  $\langle \Box(\neg\varphi \ \& \ \neg\psi) \rightarrow \Box(\varphi \equiv \psi) \rangle$ 
proof(rule  $\rightarrow I$ )
  AOT-assume  $0: \langle \Box(\neg\varphi \ \& \ \neg\psi) \rangle$ 
  — Start a modally-strict sub-proof.
  AOT-modally-strict  $\{$ 
    AOT-assume  $\langle \neg\varphi \ \& \ \neg\psi \rangle$ 
    AOT-hence  $\langle \varphi \equiv \psi \rangle$ 
    by (metis  $\&E \rightarrow I \equiv I$  reductio-aa:1)
   $\}$ 
  — Conclude the necessitation of the result by RN.
  AOT-hence  $\langle \Box((\neg\varphi \ \& \ \neg\psi) \rightarrow (\varphi \equiv \psi)) \rangle$ 
  by (metis  $\rightarrow I$  RN)
  AOT-thus  $\langle \Box(\varphi \equiv \psi) \rangle$  using  $0$  qml:1[axiom-inst]  $\rightarrow E$  by blast
qed

```

This corresponds to the following proof using Isabelle’s native outer syntax:

```

theorem  $\langle [v \models \Box(\neg\varphi \ \& \ \neg\psi) \rightarrow \Box(\varphi \equiv \psi)] \rangle$ 
proof (rule  $\rightarrow I$ )
  assume  $0: \langle [v \models \Box(\neg\varphi \ \& \ \neg\psi)] \rangle$ 
  {
    fix  $w$  — We choose a fresh possible world for our sub-proof.
    assume  $\langle [w \models \neg\varphi \ \& \ \neg\psi] \rangle$ 
    hence  $\langle [w \models (\varphi \equiv \psi)] \rangle$ 
    by (metis  $\&E \rightarrow I \equiv I$  reductio-aa:1)
  }
  hence  $\langle [v \models \Box((\neg\varphi \ \& \ \neg\psi) \rightarrow (\varphi \equiv \psi))] \rangle$ 
  by (metis  $\rightarrow I$  RN)
thus  $\langle [v \models \Box(\varphi \equiv \psi)] \rangle$  using  $0$  qml:1[axiom-inst]  $\rightarrow E$  by blast
qed

```

Additionally, we introduce the command **AOT-define**, which allows to directly state definitions of PLM (see 3.4.2). Internally, this involves introducing a new constant for the defined entity and setting up the syntax for parsing and printing according to the specified *syntactic* type (while the logical type of the constant is deduced). This new constant is then automatically specified to fulfill the given definition using a mechanism similar to the **specification** command, while the entailed existence proof is constructed automatically.²²

The convenience of this mechanism becomes apparent by inspecting a definition of *exclusive or*:

```

AOT-define xor1 ::  $\langle \varphi \Rightarrow \varphi \Rightarrow \varphi \rangle$  (infixl  $\langle XOR1 \rangle$  10)
  xor1-spec:  $\langle \varphi \ XOR1 \ \psi \equiv_{df} (\varphi \vee \psi) \ \& \ \neg(\varphi \ \& \ \psi) \rangle$ 

```

This is (roughly)²³ the same as:

```

consts xor2 ::  $\langle o \Rightarrow o \Rightarrow o \rangle$ 
syntax xor2 ::  $\langle \varphi \Rightarrow \varphi \Rightarrow \varphi \rangle$  (infixl  $\langle XOR2 \rangle$  10)
specification(xor2)
  xor2-spec:  $\langle AOT\text{-model-equiv-def } \llbracket \varphi \ XOR2 \ \psi \rrbracket \llbracket (\varphi \vee \psi) \ \& \ \neg(\varphi \ \& \ \psi) \rrbracket \rangle$ 
  by (auto intro!: exI[where  $x = \langle \lambda \varphi \ \psi . \varepsilon_o \ w . [w \models (\varphi \vee \psi) \ \& \ \neg(\varphi \ \& \ \psi)] \rangle$ ])
  simp: AOT-model-equiv-def AOT-model-proposition-choice-simp

```

We also introduce auxiliary commands like **AOT-find-theorems** and **AOT-sledgehammer** to aid in constructing proofs. **AOT-find-theorems** works similar to the Isar command **find-theorems**, but automatically parses AOT syntax and generalizes concrete variables to schematic variables for pattern matching. **AOT-sledgehammer** is a wrapper that invokes **sledgehammer** while restricting its search for theorems, s.t. the model-specific theorems are ignored and only the theorems and rules of the abstraction layer are allowed for proofs.

The list of commands can be found in A.2, while the actual ML implementation is available at [27].

²²The existence proofs are generally trivial: the definiens itself can be chosen as witness.

²³**AOT-define** additionally supports our printing modes and performs internal book-keeping needed for example for the substitution methods to recognize the new definition.

4.4. Representation of an Abstract Semantics of AOT

In A.4, we construct an abstract semantics for the primitive (and some of the basic defined) language elements of AOT. The goal of this layer of abstraction is to specify only the properties of the models that are required to derive the axiom system and rules of AOT later.

The defined semantics heavily relies on Isabelle’s **specification** command to abstract specific model choices to more general semantic properties. The model construction merely enables us to construct witnesses for the specifications.

As a simple example, we specify implications by requiring that $\varphi \rightarrow \psi$ is true in a semantic possible world w , just in case φ being true in w implies ψ being true in w (see A.4.21).

More complex examples include the specification of descriptions (see A.4.71) and the joint specification of exemplification and λ -abstraction (see A.4.125).

Notably, we specify necessity (see A.4.32) using validity in all semantic possible worlds and actuality (see A.4.38) using validity in a designated actual world w_0 (see also 4.7.4). Furthermore, we specify AOT’s identity as existing identity of meta-logical terms (see A.4.63), while we derive that this corresponds to AOT’s definition of identity at each type in A.5.72.²⁴

One goal of this intermediate layer of abstraction is to keep the derivation of the abstraction layer that contains the axioms and the deductive system of AOT impervious to minor changes in the model construction.

However, it also eliminates artifactual theorems: instead of simply defining λ -abstraction and exemplification using a concrete model construction, we introduce them using abstracted properties and merely provide a concrete witness that satisfies those properties. This increases the choice of admissible models of HOL validating our construction, since such a model is not restricted to the provided witness, but is merely bound by the abstract properties. This eliminates artifactual theorems that would merely be true for our provided witness, but are not derivable from the required properties.

For example, in the witness proof of the specification of exemplification and λ -abstraction (see A.4.125), we define exemplification, as indicated in the previous sections, as a function exe (type $\langle 'a \rangle \Rightarrow 'a \Rightarrow o$ with $'a$ of sort *AOT-IndividualTerm*) taking a relation term Π and individual terms κs to a proposition p , s.t. if Π denotes, p is given by applying the function representing Π to the individual terms κs , and if Π does not denote, p is a specific, fixed necessarily false proposition. This choice of a witness implies that $[\Pi]\kappa = [\Pi]\kappa'$ for any κ and κ' , whenever Π does not denote. However, since our specification does not imply this fact, the construction still allows for models in which $[\Pi]\kappa$ is a proposition that is distinct from $[\Pi]\kappa'$ for distinct κ and κ' (though both propositions have to be necessarily false).

In this sense, the technical details of the constructed witnesses are not particularly relevant in contrast to that we (1) have chosen representation types and basic definitions

²⁴Logical existence $\tau \downarrow$ is handled similarly.

(e.g. for terms to denote) that *allow* constructing suitable witnesses, (2) our specification is sufficiently strong to validate the axiom system of AOT and (3) our specification is weak enough and our types are general enough to preserve hyperintensionality and avoid most artifactual theorems. The details of our specifications can be found in A.4.

4.5. Specifications and the Hilbert-Epsilon-Operator

As mentioned in section 2.5.1, the **specification** command internally uses Isabelle's native Hilbert-Epsilon-operator $SOME\ x.\ \varphi\ x$. This operator is axiomatized in the meta-logic using the following single principle:

$$\varphi\ x \implies \varphi\ (SOME\ x.\ \varphi\ x)$$

In particular, this implies that the operator behaves like the classical Hilbert-Epsilon-operator, i.e. it holds that $(\exists x.\ \varphi\ x) = \varphi\ (SOME\ x.\ \varphi\ x)$. Consequently, whenever there is a witness for φ , then whatever is true for *everything* that satisfies φ is true for $SOME\ x.\ \varphi\ x$:

$$\llbracket \exists a.\ \varphi\ a; \bigwedge x.\ \varphi\ x \implies \psi\ x \rrbracket \implies \psi\ (SOME\ x.\ \varphi\ x)$$

However, it is noteworthy that this operator obeys the following principle of extensionality:

$$(\forall x.\ \varphi\ x = \psi\ x) \longrightarrow (SOME\ x.\ \varphi\ x) = (SOME\ x.\ \psi\ x)$$

This is due to the fact, that in the meta-logic, extensional equivalence implies identity, i.e. the antecedent implies $\varphi = \psi$ and the consequent follows by substitution of identicals. Therefore, we *cannot* e.g. define an intensional conjunction as follows (we reuse the type o_2 and its defined validity from section 2.5.1):²⁵

definition $o_2\text{-conj}'$ (**infixl** $\langle \wedge' \rangle$ 100) **where**

$$\langle \varphi \wedge' \psi \equiv SOME\ \chi.\ valid\text{-}o_2\ \chi \longleftrightarrow (valid\text{-}o_2\ \varphi \wedge valid\text{-}o_2\ \psi) \rangle$$

Since it holds that $(valid\text{-}o_2\ \chi = (valid\text{-}o_2\ \varphi \wedge valid\text{-}o_2\ \psi)) = (valid\text{-}o_2\ \chi = (valid\text{-}o_2\ \psi \wedge valid\text{-}o_2\ \varphi))$, commutativity of (\wedge') is immediately derivable:

lemma $\langle (p \wedge' q) = (q \wedge' p) \rangle$

unfolding $o_2\text{-conj}'\text{-def}$ **by** *metis*

However, we can avoid this issue, if we do not define the *value* of the conjunction function for specific arguments using the Epsilon-operator, but instead the conjunction function itself, i.e.:

definition $o_2\text{-conj}''$ (**infixl** $\langle \wedge'' \rangle$ 100) **where**

$$\langle (\wedge'') \equiv SOME\ conj.\ \forall\ \varphi\ \psi.\ valid\text{-}o_2\ (conj\ \varphi\ \psi) = (valid\text{-}o_2\ \varphi \wedge valid\text{-}o_2\ \psi) \rangle$$

²⁵Note that in *mixfix* notation a single quote $'$ is used as escape character for distinguishing placeholders - from underscores $'$. A syntactic single quote is therefore given as $''$.

This way, our conjunction has any property that is true for *all possible* functions that behave as conjunction under validity. In other words, any choice for a concrete conjunction is admissible, including intensional ones, as long as it has our required extensional property under validity.²⁶

This is exactly how the **specification** command works: the specification statements are transformed to closed terms by universal generalization and combined via conjunction and the result is used as the matrix of the Hilbert-Epsilon-operator. Given the provided witness, the desired properties of the Hilbert-Epsilon term become derivable.

Note that the extensionality of the Hilbert-Epsilon operator still implies that any other operator defined using a meta-logically equivalent condition is identical, i.e.:

definition $\text{o}_2\text{-conj}'''$ (**infixl** $\langle \wedge'''' \rangle$ 100) **where**

$\langle (\wedge''') \equiv \text{SOME } \text{conj} . \forall \varphi \psi . (\text{valid-o}_2 \psi \wedge \text{valid-o}_2 \varphi) = \text{valid-o}_2 (\text{conj } \varphi \psi) \rangle$

lemma $\langle (\wedge') = (\wedge''') \rangle$

by (*auto intro!*: *Eps-cong simp*: $\text{o}_2\text{-conj}'''$ -def $\text{o}_2\text{-conj}''$ -def)

To avoid this issue completely, we would need to introduce an additional dependency on a meta-logical parameter that is allowed to vary across otherwise meta-logically equivalent definitions.²⁷

4.6. Axiom System and Deductive System

The axiom system as derived in the embedding was already described in section 3.3 and the fundamental meta-rules were mentioned in section 3.4. By construction, most of them can be derived from the abstract semantics using simple, automatically generated proofs.

While the full derivation of the axiom system in the embedding can be found in A.6 and the deductive system of PLM chapter 9 is derived in A.7, in the following, we will focus on some particular axioms, rules and proofs that are challenging to represent in the embedding. This mostly happens due to PLM's statement involving either complex preconditions given in natural language or due to the statement extending over multiple types.

²⁶Note, however, that we still need to make sure that the involved *types* are sufficiently intensional as discussed in section 2.5.1.

²⁷Note that **nitpick** has specific support for the **specification** command: it ignores the underlying definition using the Hilbert-Epsilon operator, and instead solely considers the given specification, see [12]. In that sense, the underlying definition of a **specification** is commonly treated as part of an inaccessible implementational detail of an abstraction layer, even in the meta-logic HOL itself.

4.6.1. Base Cases of Denoting Terms

One of the axioms we mentioned explicitly as difficult to implement in section 3.3 is the second (in PLM's numbering) quantifier axiom which establishes a set of base cases of denoting terms. Recall the formulation of the axiom in PLM (item (39.2)):

$$\tau \downarrow, \text{ provided } \tau \text{ is a primitive constant, a variable, or a } \lambda\text{-expression in which} \\ \text{the initial } \lambda \text{ does not bind any variable in any encoding formula subterm.}$$

We implement this axiom by splitting it up into cases. The first and obvious way to split the axiom is to split it into the separate cases listed in the natural language formulation: constants, variables and λ -expressions.

The embedding does not have to distinguish explicitly between constants and variables: both constants and variables are modelled as entities of the same type (' a AOT-var) and the distinction between constants and variables is done by declaring the entity as a constant or using it as a variable in the meta-logic. So it suffices to state one case for constants *and* variables (see A.6.19):

$$\alpha \downarrow$$

α ranges over all expressions of type ' a AOT-var' (see 4.1.2) and therefore ranges over the denoting objects of type ' a ', which immediately validates $\alpha \downarrow$ semantically. Note that the axiom only extends to *primitive* constants, i.e. it does *not* extend to *defined* constants. In our embedding defined constants are modelled as *terms* of a given type, i.e. directly in the base type ' a ', not the type ' a AOT-var', so the axiom cannot be instantiated to them, as intended.

The remaining case concerns λ -expressions and is more complex to represent. Internally, a λ -expression denotes, just in case that its matrix φ is necessarily equivalent on all denoting objects that share an urelement, or formally (see A.4.269):

$$\begin{aligned} \text{AOT-model-denotes (AOT-lambda } \varphi) = \\ (\forall v \kappa \kappa'. \\ \text{AOT-model-denotes } \kappa \wedge \\ \text{AOT-model-denotes } \kappa' \wedge \text{AOT-model-term-equiv } \kappa \kappa' \longrightarrow \\ [v \models \varphi \kappa] = [v \models \varphi \kappa']) \end{aligned}$$

However, this is a semantic criterion and does not directly correspond to the formulation of above axiom. While, for arbitrary complex terms, we cannot directly capture the syntactic restriction stating that the initial λ does not bind any variable in any encoding formula subterm, we can construct a set of introduction rules for a predicate on matrices that will cover all terms that match the natural language description.

To that end, we define the auxiliary constant *AOT-instance-of-cqt-2* (see A.4.1283). This constant acts on matrices of λ -expressions, i.e. on functions that map entities of a type of class *AOT- κ s* (recall that this may either be an unary individual or a tuple of individuals, see 4.1.6) to propositions.

AOT-instance-of-cqt-2 is true for any such function that agrees on arguments that denote and are *AOT-model-term-equiv*-equivalent, i.e. that has identical values for arguments

that denote and share the same urelements. By construction of λ -expressions the use of any such function as matrix of a λ -expression will result in a denoting relation term.

Now we enrich the abstraction layer with several introduction rules for *AOT-instance-of-cqt-2*:

- Functions that do not depend on their argument correspond to matrices in which the λ -bound variables do not occur. Therefore such functions trivially fall under the formulation of the axiom (see A.4.1304).
- Exemplification formulas of the form $[\Pi]_{\kappa_1 \dots \kappa_n}$ in which the λ -bound variable does not occur in Π fall under the axiom, if all individual terms κ_i do not contain an occurrence of the λ -bound variable in encoding formula subterms. This is captured in another auxiliary constant *AOT-instance-of-cqt-2-exe-arg* (see A.4.1286) described below.
- Let $\nu_1 \dots \nu_n$ be the variables bound by the initial λ . Then an exemplification formula of the form $[\lambda \mu_1 \dots \mu_n \varphi\{\nu_1 \dots \nu_n, \mu_1 \dots \mu_n\}]_{\kappa_1 \dots \kappa_n}$ as matrix falls under the axiom, if (1) all individual terms κ_i fall under the axiom as described below and (2) φ falls under the axiom w.r.t $\nu_1 \dots \nu_n$, i.e. φ does not contain any occurrences of $\nu_1 \dots \nu_n$ in encoding formula subterms, respectively for any $\mu_1 \dots \mu_n$ it holds that $\varphi\{\nu_1 \dots \nu_n, \mu_1 \dots \mu_n\}$ as function on $\nu_1 \dots \nu_n$ satisfies *AOT-instance-of-cqt-2* (see A.4.1431).
- Complex formulas fall under the formulation of the axiom, just in case all its operands fall under the formulation of the axiom. E.g. a negation falls under the axiom, just in case the negated formula falls under the axiom (see A.4.1307).
- Encoding formulas only fall under the axiom, if the λ -bound variables do not occur in them at all. This is already covered in the first case above. However, this may be refined in the future anticipating an upcoming change in PLM as discussed at the end of this section.

The above rules cover all cases except the primary individual terms in exemplification formulas. The additional auxiliary constant *AOT-instance-of-cqt-2-exe-arg* (see A.4.1286) acts on functions taking entities of a type '*a*' of class *AOT- κ s* to entities of a type '*b*' of class *AOT- κ s*. *AOT-instance-of-cqt-2-exe-arg* holds for any such function that sends denoting and *AOT-model-term-equiv*-equivalent arguments to again *AOT-model-term-equiv*-equivalent values. By construction, if the application of any such function to the variables $\nu_1 \dots \nu_n$ occurs as primary individual term in an exemplification formula, then the exemplification formula satisfies the meta-logical definition of *AOT-instance-of-cqt-2* (since the result of the exemplification is known to agree on objects with the same urelements).

Similarly to *AOT-instance-of-cqt-2* we add introduction rules for *AOT-instance-of-cqt-2-exe-arg* to the abstraction layer:

- The identity function falls under *AOT-instance-of-cqt-2-exe-arg* (this is the case in which the λ -bound variables themselves occur as primary individual terms in an exemplification formula; see A.4.1366).
- Constant functions fall under *AOT-instance-of-cqt-2-exe-arg* (this is the case in which the λ -bound variables do *not* occur in a primary individual term of an exemplification formula; see A.4.1371).

- Definite descriptions fall under *AOT-instance-of-cqt-2-exe-arg* just in case their matrix (as function acting on the λ -bound variables) falls under *AOT-instance-of-cqt-2*, i.e. a description may occur in a primary term of an exemplification formula, if its matrix does not contain the λ -bound variables in an encoding formula subterm (see A.4.1392).
- There are further technical introduction rules due to the implementation of n-ary relations as relations acting on tuples (see A.4.1376), e.g. the *fst* and *snd* projections fall under *AOT-instance-of-cqt-2-exe-arg* (i.e. $[\lambda xy [F]x]$ and $[\lambda xy [F]y]$) and the application of the *Pair* function to two terms falls under the axiom, if both terms fall under *AOT-instance-of-cqt-2-exe-arg* (i.e. $[\lambda x [F]\kappa\kappa']$ falls under the axiom, if neither κ nor κ' contain x in an encoding formula subterm).

While the details of this construction are complex, the result is a set of introduction rules that allow proving *AOT-instance-of-cqt-2* exactly for those matrices that fall under the natural language condition of the axiom. The axiom itself is then implemented conditionally: a λ -expression denotes axiomatically, if its matrix satisfies *AOT-instance-of-cqt-2* (see A.6.21). The introduced introduction rules may be used in the abstraction layer, while it is inadmissible to unfold the definition of *AOT-instance-of-cqt-2* itself (i.e. the only matrices for which *AOT-instance-of-cqt-2* is derivable in the abstraction layer are exactly those that satisfy the natural language restriction of PLM's axiom).

Note that at the time of writing, a generalization of the axiom is under discussion that would extend it to the following:²⁸

$\tau\downarrow$, provided τ is a primitive constant, a variable, or a λ -expression in which the initial λ does not bind any variable that is a primary term in an encoding formula subterm.

In an encoding formula $\kappa_1\dots\kappa_n[\Pi]$ only Π as well as κ_1 through κ_n are defined to be primary terms, but no nested term counts as primary term, so this entails strictly more cases than the formulation given above.

In anticipation of this change, this is already validated by the embedding, however, the corresponding introduction rules are not yet added to the abstraction layer to disbar their use for the time being (see A.4.1468).

See 4.8.1 for a discussion of some consequences of this upcoming change.

4.6.2. The Rule of Substitution

Similar to the axiom above, there is also derived rules in PLM that are challenging to reproduce in the embedding. A prominent example is the Rule of Substitution. PLM formulates this rule in item (159) as follows:²⁹

²⁸The precise formulation in the upcoming next version of PLM may vary slightly in its wording, but is likely to extend over the same amount of cases.

²⁹PLM formulates the rule relative to modally-fragile derivations \vdash , but further argues that it is equally valid for modally-strict derivations \vdash_{\square} . Furthermore, it also states a variant in which the precondition is weakened to $\vdash_{\square} \varphi \equiv \chi$, which allows to derive $\vdash_{\square} \square(\varphi \equiv \chi)$ by RN.

If $\vdash \Box(\varphi \equiv \chi)$, then where Γ is any set of formulas and φ' is the result of substituting the formula χ for zero or more occurrences of ψ where the latter is a subformula of φ , $\Gamma \vdash \varphi$ if and only if $\Gamma \vdash \varphi'$.

The notable restriction in this formulation is the proviso that ψ is a *subformula* of φ . Subformulas are defined recursively in PLM item (6) and notably do not entail matrices of descriptions or non-nullary λ -expressions: E.g. the formula φ is *not* a subformula of $[F]\iota x\varphi\{x\}$ or of $[\lambda y \varphi\{y\}]x$.

While the inductive base cases for proving the rule can easily be reproduced in the embedding (see A.7.2702), combining the rule to a single statement in Isabelle is challenging. Therefore we instead provide custom-written proving **methods** that allow applying the rule as intended by PLM. This works by internally analyzing the structure of (the ML representation of)³⁰ the involved formulas in order to choose the appropriate rule that allows to reduce the goal to a substitution in a less complex formulas. In that sense, the proving methods reconstruct the general proof of the rule in PLM by induction on the complexity of the involved formulas at every invocation of the proving method on a concrete formula.

4.6.3. Proofs by Type Distinction

PLM involves proofs that involve a case distinction by type. An example is the theorem that two terms being identical implies that both denote (see A.7.930).

In our embedding, we reproduce this kind of reasoning by introducing a new type class, in this case *AOT-Term-id*, that assumes the statement of the theorem, and then by instantiating this type class to all the types the statement is supposed to apply to. We then augment the type constraints for terms of these types to include the newly defined class.

In a future version of the embedding, we intend to use Standard ML to define a simple outer syntax command (similarly to **AOT-define** discussed in section 4.3) that will hide the complexity of this process and will allow for a more intuitive statement of theorems that are to be proven by type distinction.

4.6.4. Definition of n -ary Relation Identity

Recall the definition of n -ary relation identity of PLM given in section 3.2:

$$\begin{aligned} \Pi = \Pi' \equiv_{df} & \Pi \downarrow \ \& \ \Pi' \downarrow \ \& \ \forall y_1 \dots \forall y_{n-1} \ ([\lambda x \ [\Pi]xy_1 \dots y_{n-1}] = [\lambda x \ [\Pi']xy_1 \dots y_{n-1}] \\ & \& \ [\lambda x \ [\Pi]y_1xy_2 \dots y_{n-1}] = [\lambda x \ [\Pi']y_1xy_2 \dots y_{n-1}] \ \& \ \dots \ \& \ [\lambda x \ [\Pi]y_1 \dots y_{n-1}x] = [\lambda x \\ & \ [\Pi']y_1 \dots y_{n-1}x] \) \end{aligned}$$

³⁰A proving method written in Isabelle/ML can traverse the ML representation of terms and determine structural properties. However, properties determined in this way cannot be used as logical preconditions in inner syntax. They are meta-logical properties that, in general, cannot be represented in the logical layer.

While we can easily represent ellipse notation in terms that are uniform over arities, as e.g. in β -conversion, by choosing a single variable of a type class that can be instantiated to tuples in place of the ellipse list of variables, this definition involves additional conjunctive clauses depending on the arity and is thereby harder to implement.

A solution would be to approximate the statement of the definition by stating it explicitly for finitely many arities.³¹ However, the construction using type class instantiations on product types described in section 4.1 also allows us to state the definition generically, albeit that we have to rely on an auxiliary construction in the meta-logic.

The generic version of the definition in our embedding is the following (see A.5.107):

$$\begin{aligned} \Pi &= \Pi' \equiv_{df} \\ \Pi \downarrow \ \&\ \Pi' \downarrow \ \& \\ \forall x_1 \dots \forall x_n \ (\&AOT\text{-sem-proj-id } \llbracket x_1 \dots x_n \rrbracket \ (\lambda \kappa_1 \kappa_n. \ \llbracket [\Pi] \kappa_1 \dots \kappa_n \rrbracket \ \& \\ & \ (\lambda \kappa_1 \kappa_n. \ \llbracket [\Pi'] \kappa_1 \dots \kappa_n \rrbracket \ \&)) \end{aligned}$$

The quotation marks $\llbracket - \rrbracket$ allow us to inject meta-logical terms into the custom grammar we introduced for AOT syntax and vice-versa. Here ellipses like $x_1 \dots x_n$ are, meta-logically, a single variable $x_1 x_n$ restricted to an arbitrary type of the type class *AOT- κ s*. The auxiliary constant *AOT-sem-proj-id* is defined in the type class *AOT-RelationProjection* (see A.4.407; recall that this is a subclass of *AOT- κ s*). It satisfies an additional restriction on types of the class *AOT-UnaryRelationProjection* (resp. on the concrete type κ ; see A.4.416) and has a concrete definition on products:

$$\begin{aligned} AOT\text{-sem-proj-id } \kappa \ \varphi \ \psi &= \llbracket [\lambda x \ \varphi \{x\}] = [\lambda x \ \psi \{x\}] \rrbracket \\ AOT\text{-sem-proj-id } (\kappa_1, \ \kappa_2 \kappa_n) \ (\lambda(x, \ y_1 y_n). \ \llbracket \varphi \{x, y_1 \dots y_n\} \rrbracket \ \& \\ & \ (\lambda(x, \ y_1 y_n). \ \llbracket \psi \{x, y_1 \dots y_n\} \rrbracket \ \&)) = \\ \llbracket [\lambda x \ \varphi \{x, \kappa_2 \dots \kappa_n\}] = [\lambda x \ \psi \{x, \kappa_2 \dots \kappa_n\}] \rrbracket \ \& \\ & \ \llbracket AOT\text{-sem-proj-id } \kappa_2 \kappa_n \ (\lambda y_1 y_n. \ \llbracket \varphi \{\kappa_1, y_1 \dots y_n\} \rrbracket \ \& \ (\lambda y_1 y_n. \ \llbracket \psi \{\kappa_1, y_1 \dots y_n\} \rrbracket \ \&)) \rrbracket \end{aligned}$$

Note that the outermost identities in these statements are meta-logical identities that thereby allow immediate meta-logical substitution. In the unary case, *AOT-sem-proj-id* reduces to the the identity of the one-place relations given by λ -abstracting the given matrices φ and ψ .

In the product case, it is defined for matrices acting on pairs (of type ' $a \times b$ ') as a conjunction. The first conjunct is the identity of the one-place relations resulting from λ -abstracting x in the applications of the matrices to x and $\kappa_2 \dots \kappa_n$. The second conjunct recursively refers to *AOT-sem-proj-id* on type ' b ' acting on $\kappa_2 \kappa_n$ (corresponding to $\kappa_2 \dots \kappa_n$ in our AOT syntax implementation) and partial applications of the matrices to κ_1 .

Now restricting the generic definition to type κ , yields the following instance:

$$\Pi = \Pi' \equiv_{df} \Pi \downarrow \ \& \ \Pi' \downarrow \ \& \ \forall x \ \llbracket AOT\text{-sem-proj-id } \llbracket x \rrbracket \ (\lambda \kappa. \ \llbracket [\Pi] \kappa \rrbracket \ \& \ (\lambda \kappa. \ \llbracket [\Pi'] \kappa \rrbracket \ \&)) \rrbracket$$

Unfolding the definition of *AOT-sem-proj-id* in the unary case, this yields

$$\Pi = \Pi' \equiv_{df} \Pi \downarrow \ \& \ \Pi' \downarrow \ \& \ \forall x \ (\llbracket [\lambda x \ \Pi] x \rrbracket = \llbracket [\lambda x \ \Pi'] x \rrbracket \rrbracket)$$

While this is technically not a definition of AOT, the implied equivalence is a theorem as a consequence of η -conversion.

³¹In fact, for convenience we do this for arities up to four (see A.5.87).

Restricting the definition to type $\kappa \times \kappa$, yields this instance:

$$\Pi = \Pi' \equiv_{df} \Pi \downarrow \& \Pi' \downarrow \& \forall x \llbracket AOT\text{-sem-proj-id } \llbracket x \rrbracket \rrbracket (\lambda \kappa_1 \kappa_2. \llbracket [\Pi] \kappa_1 \dots \kappa_2 \rrbracket) (\lambda \kappa_1 \kappa_2. \llbracket [\Pi'] \kappa_1 \dots \kappa_2 \rrbracket)$$

Now unfolding the definition of *AOT-sem-proj-id* in the product case (i.e. for type $\kappa \times \kappa$) followed by unfolding it for the recursive unary case, yields the proper definition of 2-ary relation identity:³²

$$\Pi = \Pi' \equiv_{df} \Pi \downarrow \& \Pi' \downarrow \& \forall y ([\lambda z [\Pi] zy] = [\lambda z [\Pi'] zy] \& [\lambda z [\Pi] yz] = [\lambda z [\Pi'] yz])$$

Similarly, instantiating to type $\kappa \times \kappa \times \kappa$ yields ternary relation identity, etc.

While this construction yields the technical means to state the definition of n -ary relation identity as well as the axiom of n -ary encoding generically, properly unfolding the meta-logical definitions can be cumbersome in practice.

For that reason we additionally explicitly derive the definition of identity and the axiom of n -ary encoding for arities up to four, which is more than sufficient for the instances currently used in PLM. For n -ary encoding we currently do not formulate a generic version, even though the same mechanism as above can be applied to this case as well.

In the future, we intend to define a convenient *theorem attribute* (see below) that can be used to immediately instantiate n -ary statements of generic form directly to an arbitrary arity n given as argument to the attribute.

Another subtlety in the definition of n -ary relation identity is the fact that two n -ary relations already have to be identical, if all their projections to unary relations using $n-1$ *denoting* individual terms are identical. However, in order to avoid artifactual theorems, we defined relations as functions that also act on *null-urelements*, resp. on tuples that may involve *null-urelements*. The identity of their projections merely implies that the functions representing the n -ary relations in question evaluate to the same propositions for all tuples of $n-1$ urelements that correspond to denoting individuals (i.e. that are not *null-urelements*) and one urelement that may be a *null-urelement*. This is the reason why in section 4.1.7 we required the behaviour of an n -ary relation on *irregular* individual terms (i.e. tuples that involve more than one *null-urelement*) to be completely determined by the behaviour of the relation on *regular* individual terms (i.e. tuples that involve at most one *null-urelement*). This way the identity of all projections of two n -ary relations to unary relations indeed implies their identity as required for validating the definition, while we still avoid the artifactual theorem that $\forall x_1 \dots \forall x_n ([\Pi] x_1 \dots x_n = [\Pi'] x_1 \dots x_n) \rightarrow \Pi = \Pi'$.

4.6.5. Auxiliary Theorem Attributes

The embedding defines several auxiliary *theorem attributes* that help in reproducing common reasoning patterns of PLM that would otherwise be subject to technical complications.

³²Technically, this additionally involves expanding the n -ary quantifier to two unary quantifiers, one of which can be eliminated.

PLM often prefers stating theorems using free object level variables rather than meta-variables (that would range over potentially non-denoting terms) in order to avoid having to specifically state the precondition that the respective terms denote.

However, whenever a term is trivially known to denote from context, PLM may simply instantiate such theorems directly to terms. This is valid, since it is always possible to apply GEN followed by \forall -elimination for terms to the theorem. To reproduce this transformation within the embedding the theorem attribute *unverify* is introduced (see A.7.705), which takes the variable to be generalized as argument and automatically performs the required transformation on the theorem. Similarly, the attribute *unconstrain* (see A.9.224) can be used to transform a theorem formulated with restricted variables to a theorem involving unconstrained variables with the added precondition that they satisfy the respective restriction conditions.

4.7. Meta Theorems

4.7.1. The Collapse of Alphabetic Variants

We already informally stated that the embedding collapses alphabetic variants. In this section we will define more precisely what this means and justify this collapse.

Isabelle internally represents bound variables using de-Brujin indices (see [16]). We will showcase this mechanism in detail below. As a consequence, terms that are alphabetic variants are meta-logically indistinguishable. To justify representing AOT's bound variables directly using bound variables in Isabelle, we need to show that both (1) AOT's notion of alphabetic variants is equivalent to Isabelle's use of de-Brujin indices and (2) any rule of AOT is still valid if any assumption or the conclusion are replaced by an alphabetic variant (as a generalization of PLM's existing *Rule of Alphabetic Variants*).³³

AOT's Alphabetic Variants align with Isabelle's use of de-Brujin Indices

Internally, Isabelle represents binding notation by function application and abstraction. E.g. if we let Isabelle print the internal ML representation of the term $\forall p (p \rightarrow p)$, we arrive at the following:³⁴

$\forall p (p \rightarrow p)$

Const (AOT-syntax.AOT-forall, (o \Rightarrow o) \Rightarrow o) \$

Abs (p, o,

Const (AOT-syntax.AOT-imp, o \Rightarrow o \Rightarrow o) \$ Bound 0 \$ Bound 0)

³³This includes theorems and axioms by thinking of them as rules with an empty set of assumptions.

³⁴Note that we are not merely talking about a representation in the meta-logic HOL, but about the internal ML representation of HOL terms. Technically, we have setup an *antiquotation* that allows us to print a term together with its internal representation.

While a complete discussion of the ML representation of terms goes beyond the scope of this thesis, it suffices to have a rough understanding of the involved syntax. The atomic terms are typed constants, *Const* (*[identifier]*, *[type]*), bound variables *Bound* [*de-Bruijn index*] and free variables *Free* (*[identifier]*, *[type]*). $\$$ is a binary operator that signifies function application between terms. *Abs* (*[name]*, *[type]*, *[term]*) is the abstraction of *[term]* over a bound variable of type *[type]*. Note that while the internal representation retains the name of the bound variable *p*, it has no logical meaning and is merely used e.g. for term printing, while, logically, occurrences of the bound variables are referred to by *Bound* with a de-Bruijn index. An index of zero refers to the innermost abstraction the bound variable is contained in. An index of one refers to the next outer abstraction, e.g.

$$\forall p (p \rightarrow \forall q (q \rightarrow p))$$

$$\text{Const (AOT-syntax.AOT-forall, } (o \Rightarrow o) \Rightarrow o) \$$$

$$\text{Abs } (p, o,$$

$$\text{Const (AOT-syntax.AOT-imp, } o \Rightarrow o \Rightarrow o) \$ \text{Bound } 0 \$$$

$$(\text{Const (AOT-syntax.AOT-forall, } (o \Rightarrow o) \Rightarrow o) \$$$

$$\text{Abs } (q, o,$$

$$\text{Const (AOT-syntax.AOT-imp, } o \Rightarrow o \Rightarrow o) \$ \text{Bound } 0 \$ \text{Bound } 1)))$$

Note that in the inner abstraction *Bound 0* refers to *q*, while *Bound 1* refers to *p*.

Our claim is that two terms or formulas of AOT are alphabetic variants, if and only if their representation using de-Bruijn indices is the same.

PLM defines alphabetic variants as follows (see PLM item (16)): It refers to two occurrences of a variable as *linked*, if both are free or they are bound by the same occurrence of a variable-binding operator. PLM further introduces *BV-notation* for formulas and terms:³⁵ the BV-notation of a formula φ is $\varphi[\alpha_1, \dots, \alpha_n]$, where $\alpha_1, \dots, \alpha_n$ is the list of all variables that occur bound in φ , including repetitions. Further $\varphi[\beta_1/\alpha_1, \dots, \beta_n/\alpha_n]$ refers to the result of replacing α_i by β_i in $\varphi[\alpha_1, \dots, \alpha_n]$. Now φ' is defined to be an *alphabetic variant* of φ just in case for some *n*:

- $\varphi' = \varphi[\beta_1/\alpha_1, \dots, \beta_n/\alpha_n]$,
- φ' has the same number of bound variable occurrences as φ and so can be written as $\varphi'[\beta_1, \dots, \beta_n]$, and
- for $1 \leq i, j \leq n$, α_i and α_j are linked in $\varphi[\alpha_1, \dots, \alpha_n]$ if and only if β_i and β_j are linked in $\varphi'[\beta_1, \dots, \beta_n]$.

By definition, each group of *linked* variable occurrences in AOT corresponds to exactly one abstraction in Isabelle's internal representation and all de-Bruijn indexed *Bound* terms that refer to this abstraction. Since changing the variable name of a linking group will not affect the de-Bruijn indices, the de-Bruijn representation of two alphabetic variants is therefore the same. Conversely, changing any index in the de-Bruijn representation translates to breaking a linking group as defined in PLM, thereby terms with different de-Bruijn representation are not alphabetic variants.

³⁵In the following we will restrict our discussion to formulas, but the argument applies analogously to terms as well.

Since thereby the formulas and terms that are collapsed in Isabelle's internal representation are exactly the alphabetic variants of AOT, it remains to argue that the collapse is inferentially valid, i.e. AOT allows to freely interchange alphabetic variants in any derivation.

Equivalence of Alphabetic Variants in AOT

Conveniently, PLM itself derives the following *Rule of Alphabetic Variants* (see PLM item (114)):³⁶

| $\Gamma \vdash \varphi$ if and only if $\Gamma \vdash \varphi'$, where φ' is any alphabetic variant of φ . |

It is straightforward to strengthen this further to the following:

| $\Gamma \vdash \varphi$ if and only if $\Gamma' \vdash \varphi'$, where φ' is any alphabetic variant of φ and Γ' is a set of alphabetic variants of Γ , i.e. for every $\psi \in \Gamma$ there is an alphabetic variant ψ' of ψ , s.t. $\psi' \in \Gamma'$, and vice-versa. |

To see that this rule is valid, it suffices to realize that for every $\psi \in \Gamma$ and $\psi' \in \Gamma'$ by the above rule it holds that $\psi \dashv\vdash \psi'$ and hence all premises in Γ are derivable from Γ' and vice-versa. More rigorously, the version with assumptions can be reduced to the version without assumptions by arguing with successive applications of the deduction theorem to eliminate the assumptions, applying the version of the rule without assumptions and then reconstructing the result using modus ponens. This mechanism is shown explicitly in section 4.7.3 for a similar case.

Hence, AOT allows one to freely move from any formula to an alphabetic variant in all theorems and assumptions, justifying the fact that the embedding identifies alphabetic variants.

4.7.2. Free Variable Notation, Substitutability and Bound Variables

As mentioned in chapter 3, PLM allows terms and formulas with arbitrary free variables to be used in place of its meta-variables, except for free variables that are explicitly excluded in natural language. The embedding on the other hand requires one to explicitly mention any variables that are bound at the occurrence of a meta-variable, if they should be allowed to occur in an instance of the meta-variable. This is due to the fact that binders are implemented in the embedding as operators that act on functions. Similarly, the substitution of variables in meta-variables is implemented using function application. For example, PLM formulates the first quantifier axiom as follows (see PLM item (39.1)):

| $\forall \alpha \varphi \rightarrow (\tau \downarrow \rightarrow \varphi_\alpha^\tau)$, provided τ is substitutable for α in φ |

Here φ_α^τ is defined in PLM item (14) as recursively replacing all occurrences of α in φ that are not bound *within φ itself* with τ .

³⁶Note that while PLM states meta-rules using \vdash , unless otherwise noted by convention they apply to both \vdash and $\vdash\Box$. See remark (67) in PLM. We adopt this convention in the following sections.

The precise definition of *being substitutable* can be found in PLM item (15). In particular, it states the following summary:

τ is substitutable at an occurrence of α in φ or σ just in case every occurrence of any variable β free in τ remains an occurrence that is free when τ is substituted for that occurrence of α in φ or σ .

and further:

τ is substitutable for α in φ or σ just in case τ is substitutable at every free occurrence of α in φ or σ .

In the embedding, the same axiom is stated as follows:

$$\forall \alpha \varphi\{\alpha\} \rightarrow (\tau \downarrow \rightarrow \varphi\{\tau\})$$

Internally, φ is a function acting on terms and both $\varphi\{\alpha\}$, resp. $\varphi\{\tau\}$, are the function application of φ to α , resp. τ . The following is the HOL representation of the formula of the axiom:

$$AOT\text{-imp} (AOT\text{-forall} (\lambda \alpha. \varphi \alpha)) (AOT\text{-imp} (AOT\text{-denotes } \tau) (\varphi \tau))$$

The \forall -quantifier is represented as the function application of the constant *AOTforall* to the meta-logical λ -abstraction of φ applied to the bound variable α . The substitution of τ for α in φ is represented as the function application of φ to τ .

As mentioned in section 4.7.1, internally Isabelle represents bound variables using de-Brujn indices that uniquely associate any bound variable with its binder, independently of the name of the variable. β -reduction of the function application of an abstraction to a term merely replaces the bound variables referring to the outermost abstracted variable. Thereby, substitutability is implicit in the construction: applying a meta-variable that is represented as a function to different arguments does not affect variables bound by nested binders.

Therefore, strictly speaking, the implementation of the axiom in the embedding is stronger than the axiom stated in PLM. Consider the following instance of the axiom:

$$\forall \alpha \exists \beta (\beta = \alpha) \rightarrow (\tau \downarrow \rightarrow \exists \beta (\beta = \tau))$$

Here φ is $\exists \beta \beta = \alpha$. Now in PLM's terms, β itself would not be substitutable for α in φ , since substituting β for α directly would result in β being bound by the existence quantifier. However, Isabelle allows this instantiation and resolves this issue by automatically generating an alphabetic variant of the nested binder. The following is the direct result of instantiating φ to $\exists \beta \beta = \alpha$ and τ to β in above axiom:

$$\forall \alpha \exists \beta (\beta = \alpha) \rightarrow (\beta \downarrow \rightarrow \exists \beta' (\beta' = \beta))$$

While this is not a direct instance of the axiom in PLM, we have argued in section 4.7.1 that it is a meta-theorem of AOT that all alphabetic variants are interderivable. Furthermore, for any φ an alphabetic variant can be constructed that makes any τ substitutable for an occurrence of α in φ by replacing all variables bound in φ that occur free in τ by fresh variables.

This signifies one of the main advantages and simultaneously disadvantages of the use of SSEs. While the use of the meta-logical mechanisms to deal with alphabetic variants and binders allows the implementation to forgo a custom implementation of concepts like substitutions and substitutability, this in turn requires a careful meta-theoretical analysis to assure that the resulting implementation remains faithful. However, for practical purposes the advantages outweigh the disadvantages. Not only is a custom implementation of substitutions and alphabetic variants error-prone and cumbersome, since it is at the same time seemingly trivial, but nonetheless implementationally complex, but relying on the meta-logical implementation has also significant advantages for automated reasoning: For example, while by construction Isabelle will see alphabetic variants as identical entities and can freely substitute them, manual substitution, as it would be required for deep embeddings, would require rigorous proofs about recursively defined transformations on the deep syntax representation that can quickly go beyond the limits of the available automation capabilities, even without attempting to prove complex theorems.

4.7.3. Generalizing Free Variables to Schematic Variables

After a theorem is proven in Isabelle, it is implicitly exported to the current theory context in *schematic* form. That means each free variable used in the theorem is implicitly generalized to a *schematic variable* that can be instantiated to any variable or term of the same type. Since the embedding uses distinct types for (denoting) variables and (potentially non-denoting) terms that have the same type in AOT (see 4.6.1), this does *not* mean that any theorem involving AOT variables can be directly instantiated to AOT terms, however, it does mean that all theorems of AOT are implicitly stated using meta-variables ranging over all variable names. As an example the theorem $\forall F ([F]x \rightarrow [F]x)$ not only implicitly asserts its alphabetic variants, e.g. $\forall G ([G]x \rightarrow [G]x)$, but can also be directly instantiated for a different free individual variable, e.g. $\forall G ([G]y \rightarrow [G]y)$. In the notation of AOT this means that we actually state the theorem $\forall G ([G]\nu \rightarrow [G]\nu)$, where ν ranges over all names for individual variables. While PLM does not derive a meta-rule that matches this principle, it is usually a straightforward consequence of a series of applications of the meta-rule of universal generalization GEN followed by applications of the rule of \forall Elimination for variables. However, to formulate this as a general principle, some care has to be taken and we have to additionally rely on the collapse of alphabetic variants.

We start by stating and proving the trivial case as a rule in AOT's system:

| If $\vdash \varphi$, then $\vdash \varphi_\alpha^\beta$ where β is substitutable for α in φ . |

Assume $\vdash \varphi$. Since the derivation of φ does not need any premises, it follows by the rule of universal generalization (GEN) (see section 3.4.1) that $\vdash \forall \alpha \varphi$.³⁷ Since by assumption

³⁷Note that we are using PLM's syntactic convention here, i.e. α may occur free in φ , which using our conventions we would usually signify by writing $\varphi\{\alpha\}$.

β is substitutable for α in φ we can immediately conclude by \forall Elimination (see A.7.643) that $\vdash \varphi_\alpha^\beta$.

However, we want to generalize this rule further to a version that allows for premises and does not require the proviso that β is substitutable for α in φ .

To that end the next step is to generalize above rule to include premises:

If $\Gamma \vdash \varphi$, then $\Gamma_\alpha^\beta \vdash \varphi_\alpha^\beta$ where (1) β is substitutable for α in φ and (2) β is substitutable for α in all $\psi \in \Gamma$ and (3) Γ_α^β is the set of all ψ_α^β for $\psi \in \Gamma$.

One way to show this is by first eliminating all premises in Γ using the deduction theorem (see section 3.4.1) and then referring to the simpler rule above. The resulting theorem will yield φ_α^β from Γ_α^β by successive applications of modus ponens.

In particular, let ψ_1, \dots, ψ_n be the list of premises in Γ , s.t. $\psi_1, \dots, \psi_n \vdash \varphi$.³⁸ By the deduction theorem it follows that $\psi_1, \dots, \psi_{n-1} \vdash \psi_n \rightarrow \varphi$. Continuing to apply the deduction theorem, we end up with $\vdash \psi_1 \rightarrow (\psi_2 \rightarrow (\dots \rightarrow (\psi_n \rightarrow \varphi)\dots))$. By assumption β is substitutable for α in this theorem, hence by the rule above we can conclude that: $\vdash \psi_{1\alpha}^\beta \rightarrow (\psi_{2\alpha}^\beta \rightarrow (\dots \rightarrow (\psi_{n\alpha}^\beta \rightarrow \varphi_\alpha^\beta)\dots))$

Since all $\psi_{i\alpha}^\beta$ are in Γ_α^β , it follows that $\Gamma_\alpha^\beta \vdash \varphi_\alpha^\beta$ by n applications of modus ponens.

What remains is the proviso that β be substitutable for α in φ and in all $\psi \in \Gamma$. However, note that for every φ and Γ we can choose alphabetic variants φ' and Γ' that replace all bound occurrences of β with a fresh variable γ that does not occur in φ or in any $\psi \in \Gamma$.

In the last section we have seen that $\Gamma \vdash \varphi$, if and only if $\Gamma' \vdash \varphi'$. Since β is trivially substitutable for α in φ' and in all $\psi \in \Gamma'$, it follows by the rule above that $\Gamma_\alpha^\beta \vdash \varphi_\alpha^\beta$. Since Isabelle collapses alphabetic variants by eliminating concrete variable names with de-Bruijn indices, this suffices as justification for the schematic generalization of free variables in theorems and rules in the embedding.

To clarify the last argument, consider the following theorem as example:

$$\forall x ([R]xy \rightarrow [R]xy)$$

Isabelle will let us instantiate this theorem using z in place of y , i.e. $\forall x ([R]xz \rightarrow [R]xz)$ is an instance of above theorem.

However, Isabelle will not allow one to *directly* instantiate y to x , since in $\forall x ([R]xx \rightarrow [R]xx)$ (which also happens to be a theorem, but a distinct one) all occurrences of x are *bound* and while Isabelle allows to instantiate *schematic variables* to free variable, it does not allow instantiating them to bound variables.³⁹

But since alphabetic variants are collapsed, the following is *identical* to the original theorem: $\forall z ([R]zy \rightarrow [R]zy)$

In this formulation of the theorem, there is no a naming conflict and we *can* instantiate y to x to get $\forall z ([R]zx \rightarrow [R]zx)$.

³⁸Note the discussion of derivations in PLM item (59).

³⁹To be precise Isabelle *will* in fact allow this kind of instantiation, but only in situations in which it can automatically rename the bound variable on its own, as we do manually in the continuation of the example.

Note that this is still an *instance* of the original theorem, but we just cannot state this instance without simultaneously renaming the bound variable. Even though, internally, the variable names are eliminated, concrete variable names, of course, still make a difference when *parsing* inner syntax.

Given this discussion, the meta-rule derived above together and the justification of the collapse of alphabetic variants, we may conclude that the fact that Isabelle implicitly generalizes free variables to schematic variables remains faithful to the derivational system of AOT.⁴⁰

4.7.4. Trivial Accessibility Relation for the Modal Logic

As already hinted at in section 2.3, choosing a trivial accessibility relation (respectively, equivalently, no accessibility relation at all) is a natural choice for modelling the modal logic of AOT. In fact, it can be shown that if AOT's actuality operator is modelled using a fixed actual world, any accessibility relation used for modelling necessity has to be trivial.

To see this, consider the following simple embedding of a general extensional modal logic with actuality operator, in which the actuality operator is modelled as validity in an arbitrary, but fixed actual world w_0 .

consts $r :: \langle w \Rightarrow w \Rightarrow bool \rangle$

consts $w_0 :: w$

type-synonym $o = \langle w \Rightarrow bool \rangle$

definition $valid :: \langle o \Rightarrow bool \rangle (\langle \models \rightarrow \rangle)$ **where** $\langle valid \equiv \lambda \varphi . \forall w . \varphi w \rangle$

definition $impl :: \langle o \Rightarrow o \Rightarrow o \rangle$ (**infixl** $\langle \rightarrow \rangle$ 8) **where** $\langle impl \equiv \lambda \varphi \psi w . \varphi w \rightarrow \psi w \rangle$

definition $box :: \langle o \Rightarrow o \rangle$ ($\langle \Box \rightarrow \rangle$ [50] 50) **where** $\langle box \equiv \lambda \varphi w . \forall v . r w v \rightarrow \varphi v \rangle$

definition $actual :: \langle o \Rightarrow o \rangle$ ($\langle \mathcal{A} \rightarrow \rangle$ [50] 50) **where** $\langle actual \equiv \lambda \varphi w . \varphi w_0 \rangle$

definition $equiv :: \langle o \Rightarrow o \Rightarrow o \rangle$ (**infixl** $\langle \equiv \rangle$ 10) **where** $\langle equiv \equiv \lambda \varphi \psi w . \varphi w \leftrightarrow \psi w \rangle$

In this simple system, **sledgehammer** can immediately prove that all semantic possible worlds have to be related by the accessibility relation, given the T axiom and one of AOT's axioms of actuality and necessity:

lemma

assumes $\langle \bigwedge \varphi . \models (\Box \varphi \rightarrow \varphi) \rangle$

and $\langle \bigwedge \varphi . \models (\Box \varphi \equiv \mathcal{A} \Box \varphi) \rangle$

shows $\langle \forall x y . r x y \rangle$

by (*metis (mono-tags, opaque-lifting) assms equiv-def actual-def box-def impl-def valid-def*)

However, note that this does not mean that a trivial accessibility relation is in fact the only choice in modelling AOT's modal logic. While the S5 axioms imply that the accessibility relation has to be an equivalence relation, we conjecture that it is possible to model an actuality operator by choosing a different actual world for each equivalence class of worlds induced by the accessibility relation.

⁴⁰Note that for free meta-variables the generalization to schematic form is in fact a requirement for being able to instantiate the meta-variables to arbitrary terms as intended by AOT.

However, independently of potential philosophical issues one may raise against presuming (even if only for the purpose of modelling) that, in different modal contexts, different worlds may be *actual*, an additional practical problem arises: in order to additionally satisfy AOT's axiom for rigid definite descriptions, the description operator would need to become world-relative: instead of choosing the unique object that satisfies the matrix of the description in the globally-fixed actual world, the description operator would have to choose the unique object that satisfies the matrix in the respective actual world of the equivalence class of possible worlds in which the formula containing the description is evaluated.

Allowing the denotation of an individual term to vary depending on modal context constitutes a significant complication for the models. Therefore, our current work forgoes further analysis of this way to extend our representation of AOT. However, such an extension of the models may constitute an interesting topic for future research. We conjecture that it is possible to construct models with a different actual world for each equivalence class of worlds, and that doing so could lead to a means to reproduce the strict distinction between modally-strict and modally-fragile theorems in AOT as follows (recall section 3.4.7): while modally-strict theorems could be modelled as being valid in all possible worlds, i.e. across all equivalence classes in the accessibility relation, modally-fragile axioms could be modelled as being valid in a globally-fixed actual world. This way, adding a contingent truth to the modally-fragile derivation system would merely assert that it holds in the globally-fixed actual world, whereas a modally-strict proof of some statement being *actually true* would require that statement to be true in *all* actual worlds. This would constitute a model in which $\vdash \varphi$ would no longer imply $\vdash_{\Box} \mathcal{A}\varphi$ and, consequently, in which the converse of RN fails (as allowed by PLM), i.e. $\vdash_{\Box} \varphi$ would no longer imply $\vdash \varphi$ (while the former merely requires φ to be valid in all worlds accessible from the globally-fixed actual world, the latter also requires φ to be true even in worlds inaccessible from the global actual world).

4.7.5. Primitive Inferences of Isabelle/Pure and Derivations of AOT

As mentioned in section 2.4, being able to trust the abstraction layer constructed for AOT relies on verifying that inferences in the meta-logic correspond to valid reasoning in the system of PLM, given that the set of available theorems and rules is suitably restricted.

We implement the rules of AOT as rules in Isabelle's Pure logic. The primitive inferences of Pure are described in section 2.3 of [54].⁴¹ In this section we will in particular argue that the rules in figure 4.2, when applied in our abstraction layer, will correspond to valid reasoning in AOT.⁴²

⁴¹In particular figure 4.2 is presented as figure 2.2 in section 2.3.1 of [54].

⁴²As noted below, an exhaustive analysis would also need to consider the richer logic of Isabelle/HOL.

$$\begin{array}{c}
\frac{A \in \Theta}{\vdash A} \text{ (axiom)} \quad \frac{}{A \vdash A} \text{ (assume)} \\
\frac{\Gamma \vdash B[x] \quad x \notin \Gamma}{\Gamma \vdash \bigwedge x. B[x]} (\bigwedge\text{-intro}) \quad \frac{\Gamma \vdash \bigwedge x. B[x]}{\Gamma \vdash B[a]} (\bigwedge\text{-elim}) \\
\frac{\Gamma \vdash B}{\Gamma - A \vdash A \implies B} (\implies\text{-intro}) \quad \frac{\Gamma_1 \vdash A \implies B \quad \Gamma_2 \vdash A}{\Gamma_1 \cup \Gamma_2 \vdash B} (\implies\text{-elim})
\end{array}$$

Figure 4.2.: Primitive inferences of Pure

The meta-logical *axiom* rule corresponds to PLM items (63.1) and (63.3) which state that axioms and theorems of AOT can be used in derivations.

assume corresponds to the special case of PLM item (63.2) given as $\varphi \vdash \varphi$.

\bigwedge -*intro* and \bigwedge -*elim* align with our implementations of PLM's GEN rule and \forall -elimination: Using our notational convention, it is an instance of \bigwedge -*intro* that $\Gamma \vdash \varphi\{\alpha\}$ and $\alpha \notin \Gamma$ implies $\Gamma \vdash$ for arbitrary α : $\varphi\{\alpha\}$. The latter is the precondition of our GEN rule, i.e. we can derive $\forall \alpha \varphi\{\alpha\}$ in AOT. Similarly, the \bigwedge -*elim* rule corresponds to the rule given in A.7.643, which states that we can derive $\varphi\{\beta\}$ from $\forall \alpha \varphi\{\alpha\}$.

Note, however, that the \bigwedge -*intro* and \bigwedge -*elim* rules are not restricted to our defined types of object-level variables of AOT. In particular, they can also be applied to meta-variables ranging over terms of AOT. However, applications of \bigwedge -*intro* and \bigwedge -*elim* to meta-variables exactly corresponds to the fact that PLM allows to instantiate arbitrary terms of a given type in place of its meta-variables.

The \implies -*intro* and \implies -*elim* rules correspond to the deduction theorem (PLM item (75)), which states that if $\Gamma, \varphi \vdash \psi$, then $\Gamma \vdash (\varphi \rightarrow \psi)$ and the meta-rule stated in PLM item (63.5) stating that if $\Gamma_1 \vdash \varphi$ and $\Gamma_2 \vdash (\varphi \rightarrow \psi)$, then $\Gamma_1, \Gamma_2 \vdash \psi$.

Furthermore, Pure is equipped with a primitive equality that allows for substituting terms that are meta-logically equal. In general, PLM's identity corresponds to meta-logical equality on denoting terms (see A.4.63) and non-denoting terms in PLM are *not* meta-logically identical (e.g. recall the fact that non-denoting definite descriptions can be assigned distinct *null*-urelements). While in certain corner cases, the embedding may involve artifactual identities (see 4.8), those cannot be derived without explicit appeals to the semantics. For implicit meta-logical identities that occur in alphabetic variants, we argued in section 4.7.1 that the meta-logical equality is consistent with reasoning in PLM.

While we do not claim that this analysis is exhaustive,⁴³ it nevertheless provides strong evidence for the assumption that reasoning in our abstraction layer is in fact reproducible

⁴³For example, while the rules of our target theory are implemented in the format of rules of Isabelle/Pure, the automated proving methods we use (e.g. *metis*, *meson* and *blast*) work relative to the richer logic of Isabelle/HOL (see chapter 2 of [45]) and for a full account the relevant axioms and inferences of Isabelle/HOL would need to be considered as well.

as derivations in the sense of PLM. For the purpose of a seamless exchange of results between our embedding and PLM, this level of assurance has proven sufficient. In our work we have not encountered a proof in our abstraction layer that could not be reproduced in the system of PLM.

Conversely, the fact that we can derive PLM's axioms and rules in the embedding shows that derivations of PLM can be reproduced in the embedding.

An interesting project for future research may be to implement AOT directly as an object logic of Isabelle/Pure. However, instead of being able to rely on the soundness of Isabelle/HOL as semantic backend, this would require a direct axiomatization of AOT in Pure, which means that we would lose the ability to easily judge the consistency of our representation of AOT and of extensions of its axiom system, which is one of the prime objectives of our current project.

4.8. Artifactual Theorems

In general, artifactual theorems can be defined as follows:

Let T be the target theory and M be the theory in which we are building a model \mathcal{M} of T (so that \mathcal{M} is expressible in M). Then an artifactual theorem ϕ of T relative to M and \mathcal{M} is a formula expressible in the language of T that is derivable in M from \mathcal{M} but which is not derivable in T itself. For example, if T is second-order logic with identity (2OL=) and M is ZF+U (Zermelo-Fraenkel set theory with Urelements) and the model \mathcal{M} in ZF+U represents the predicates of T as sets of Urelements in ZF+U, then the claim:

$$\phi = \forall x(Fx \equiv Gx) \rightarrow F = G$$

becomes derivable in ZF+U from \mathcal{M} , even though it is not a theorem of 2OL=. (In this case, ϕ is interpreted in \mathcal{M} as an instance of the axiom of Extensionality of ZF+U.) This particular ϕ is therefore an artifactual theorem of 2OL= relative to ZF+U and the model \mathcal{M} of predicates as sets.

The abstraction layer we define in our embedding aims to disallow artifactual theorems by limiting theoremhood to what can be derived from the representation of the axioms and rules of T in \mathcal{M} ; thus, appeals to the axioms and rules of M (beyond those that correspond to the axioms and rules of T) are not allowed in the derivations of theorems of T .

We have discussed in section 4.7 that for technical reasons the embedding collapses certain classes of statements (e.g. alphabetic variants), but that this merely extends to statements that are interderivable in AOT itself. As a result we can reasonably assume that well-formed statements of AOT that are provable in the abstraction layer of our embedding also have a derivation in AOT, i.e. only theorems that are derivable from the formal system T itself are derivable from \mathcal{M} using the representation of the axioms and rules of T .

Ideally, the construction of \mathcal{M} is general enough, s.t. even using the full system of axioms and rules of M , no theorem is derivable from \mathcal{M} that does not have a derivation in the formal system of T itself. However, in the case of our embedding, there are still counterexamples.

As a matter of fact, comparing derivability in the abstraction layer of the embedding, respectively in the formal system of PLM itself, with validity in our underlying semantic structure has been the driving force in our collaboration with the authors of AOT.

In particular, whenever a potential artifactual theorem was recognized, this resulted in an analysis of the discrepancy which regularly led to either a further abstraction of the semantics used in the embedding to eliminate the theorem or to an extension of AOT's axiom system itself, in case it turned out that (1) the discrepancy could be resolved by a natural extension of AOT's axiom system, (2) this extension had merit in that it allowed for deriving new interesting theorems in AOT or that it simplified existing derivations and (3) the extension was philosophically justifiable.

An example of a statement that is now a theorem of AOT, but originated as an artifactual theorem of the embedding, is the necessary and sufficient conditions for relations to denote discussed in section 3.8.2. An earlier example is the coexistence axiom discussed in 3.7, the formulation of which was based on a similar principle that was discovered in the analysis of the semantic properties of the embedding at the time.

This process is ongoing and in the remainder of this section we will discuss some examples of remaining artifactual theorems and the current state of their discussion.

4.8.1. Identity of Projections to Indistinguishable Objects

A variant of the fact that there are indistinguishable abstract objects discussed in section 3.8.1 is the fact that for every two-place relation of AOT there are distinct abstract objects, s.t. the projections of the relation to these abstract objects are identical (see A.7.8428 and A.7.8473):

$$\begin{aligned} \exists x \exists y (A!x \ \& \ A!y \ \& \ x \neq y \ \& \ [\lambda z \ [R]zx] = [\lambda z \ [R]zy]) \\ \exists x \exists y (A!x \ \& \ A!y \ \& \ x \neq y \ \& \ [\lambda z \ [R]xz] = [\lambda z \ [R]yz]) \end{aligned}$$

However, the construction used for the embedding makes the following stronger statements true:

$$\begin{aligned} \forall F ([F]a \equiv [F]b) \rightarrow [\lambda x \ [R]xa] = [\lambda x \ [R]xb] \\ \forall F ([F]a \equiv [F]b) \rightarrow [\lambda x \ [R]ax] = [\lambda x \ [R]bx] \end{aligned}$$

This is an artifact of modelling relations as proposition-valued functions acting on urelements. Since being indistinguishable, $\forall F ([F]a \equiv [F]b)$, semantically implies that a and b share the same urelement, the projections are forced to collapse.

However, we already mentioned in section 4.6.1 that it is currently being considered to extend the base cases of denoting λ -expressions. This extension has particular merit in deriving theorems in higher-order object theory. In the second-order fragment it would be a consequence of this change that the following λ -expressions denotes by axiom:

$$[\lambda x y [\lambda z [R]zx]]\downarrow$$

Under this assumption, however, the currently artifactual theorems above become proper theorems of AOT, respectively theorems of the abstraction layer of the embedding (see A.13.807 for a proof). By an analogous proof (see A.13.831), even the following becomes derivable (since the extended axiom will also assert that $[\lambda x y [\lambda z [G]x]]\downarrow$):

$$\forall F ([F]a \equiv [F]b) \rightarrow \forall G ([G]a = [G]b)$$

Semantically, this theorem states that whenever two objects share an urelement, then exemplifying any property results in the same proposition for both of them, which further consolidates the derivational system of AOT with the representation of relations as proposition-valued functions acting on urelements.

So while the theorems above are currently artifactual, they are likely to become proper theorems of the next upcoming version of PLM.

4.8.2. Proposition Identity and Identity of Propositional Relations

AOT's definition of proposition identity reduces proposition identity to the identity of unary propositional relations (see A.5.114):

$$p = q \equiv_{df} p\downarrow \& q\downarrow \& [\lambda x p] = [\lambda x q]$$

However, due to the fact that our semantic specification of exemplification and λ -abstraction (see A.4.125) is polymorphic and simultaneously specifies relations of all arities, it involves the following more general assertion:

$$[\lambda \nu_1 \dots \nu_n \varphi] = [\lambda \nu_1 \dots \nu_n \psi] \implies \varphi = \psi$$

It is a consequence of this more general semantic principle that, for example, the following becomes an artifactual theorem:

$$[\lambda xy p] = [\lambda xy q] \equiv p = q$$

Even though relations are modelled as proposition-valued functions in the embedding, theoretically, it is possible to allow the λ -expressions in question to map to propositions that are merely necessarily equivalent to p , resp. q , but not identical to them. However, since the definition of proposition identity still needs to be validated, this would require splitting the specification of exemplification and λ -expressions into separate cases for relations on unary individual terms and tuples of individual terms (e.g. using an additional system of type classes), which represents a technical challenge. The details of such a modified construction also depend on more general open questions regarding n -ary relation identity and generalized η -conversion, which we will discuss in the next section.

4.8.3. Corner-Cases of Relation Identity

AOT involves two axiomatic, respectively definitional claims about identity between n -ary ($n \geq 2$) relation terms (besides the identity of alphabetic variants), in particular η -conversion:

$$[\lambda x_1 \dots x_n [F]x_1 \dots x_n] = F$$

As well as n -ary relation identity, e.g. for $n = 2$:

$$\Pi = \Pi' \equiv_{df} \Pi \downarrow \ \& \ \Pi' \downarrow \ \& \ \forall y \ ([\lambda z [\Pi]zy] = [\lambda z [\Pi']zy] \ \& \ [\lambda z [\Pi]yz] = [\lambda z [\Pi']yz])$$

However, AOT does not presuppose that nested atomic exemplification formulas in λ -expressions can be arbitrarily contracted to identical relations. For example, none of the following are theorems of AOT:

$$[\lambda xy [\lambda z [F]zy]x] = [\lambda xy [F]xy]$$

$$[\lambda xy [\lambda z [F]xz]y] = [\lambda xy [F]xy]$$

$$[\lambda xy [\lambda z [F]zy]x] = [\lambda xy [\lambda z [F]xz]y]$$

The embedding constructs λ -abstraction and exemplification using the **specification** command by postulating that λ -abstraction and exemplification have to exhibit certain properties (e.g. β - and η -conversion) and by then providing a concrete witness that satisfies these properties.

However, the postulated properties given in the specification go beyond the axioms of AOT they ultimately validate. Validating the axioms of AOT for arbitrary n -ary relations in the embedding while maintaining the definition of n -ary relation identity requires, at least in the current construction of the embedding, additional properties for λ -abstraction and exemplification.⁴⁴

While the artifactual theorems above are validated by the provided witness for our specification, it is currently unknown whether the properties postulated in the specification are sufficient to derive them as artifactual theorem. Neither can **nitpick** provide a counterexample for the theorem, nor can **sledgehammer** construct a proof from the specification, so further manual inspection of the situation is required.

Interestingly, in general, AOT refrains from presuming the identity of its intensional entities, even under conditions that would usually be assumed to imply equality. η -conversion is a notable exception to this principle. However, there are also arguments for *rejecting* η -conversion in an hyperintensional context that is meant to accurately represent human thought and language, see e.g. [36].

So independently of the potential artifactual theorem discussed above, it is an interesting philosophical question whether η -conversion should be presumed by axiom at all. Similarly, there are open questions about the definition of identity of n -place relations in AOT and a potential alternative definition using n -ary encoding as discussed in PLM [63] item (37). Curiously, while the current definition of n -ary relation identity reduces the identity of ternary relations to the identity of all their projections to unary relations, the identity of all their projections to two-place relations does not imply the identity of direct projections to unary relations (without a more general contraction principle) and therefore does not imply the identity of the ternary relations.

⁴⁴For example, the property named *AOT-sem-exe-denoting* in A.4.125 is solely used for validating n -ary relation identity.

We expect that a future more extensive analysis of this issue will, similarly to previous artifactual theorems, result in further theoretical insights, ultimately followed by either an enhancement of the formulation of AOT or a refined embedding, in which e.g. the above might provably not be theorems, even outside the abstraction layer.

4.9. Discussion

We have described an implementation of the second-order fragment of AOT in classical higher-order logic by means of an SSE that can accurately reproduce AOT's reasoning in an abstraction layer. While our semantic backend is not provably free of artifactual theorems, this can be explained due to the fact that AOT does not itself presuppose a strong and exhaustive *intended semantics*, relative to which a completeness result is intended and could be achieved. On the contrary, the authors of AOT explicitly try to avoid letting the axioms and deductive system of AOT be *driven by semantics*, but rather aspire to devise a philosophically justifiable formal system that stands independently of a set-theoretic semantics and in which notions like truth values and possible worlds can instead be analysed as objects of the system itself:

It is important to remember that the formal semantics simply provides a set-theoretic framework in which models of the metaphysical theory may be constructed. The models serve the heuristic purpose of helping us to visualize or picture the theory in a rigorous way. It is extremely important not to confuse the models of the theory with the world itself. [...] So the goal of our enterprise is not to build a model, but rather to construct a formal theory that correctly mirrors the structure the world may have and, as a result, correctly reflects the entailments among the data.⁴⁵

Nevertheless, our semantic analysis could significantly contribute especially to the theoretical understanding of the conditions, in AOT, under which relations exist. These existence questions require rigorous philosophical consideration and can have a profound impact on the axiom system (recall e.g. 3.7).

While there are open questions e.g. concerning the identity of n -ary relation terms, we anticipate these questions to be the subject of future debate that will, similar to past examples of similar discussions, result in both theoretical insights and an improved implementation.

Given our discussion of the general system of AOT in the previous chapter and its implementation in our embedding in this chapter, we are now suitably equipped to discuss our implementation of PLM's construction of natural numbers, including the extended model construction required for validating its additional axioms.

⁴⁵Edward Zalta in [59] section 2.4.

5. Natural Numbers in AOT

While AOT can represent mathematical theories themselves as *abstract objects* (see chapter 6), it distinguishes this analysis of *Theoretical Mathematics* from the notion of *Natural Mathematics*. *Natural Mathematics* consists of ordinary, pretheoretic claims about mathematical objects and arises directly as abstraction of exemplification patterns rather than being based on the axioms of some mathematical theory (see item (304) in PLM¹).

Following this idea, Uri Nodelman's and Edward Zalta's claim in PLM chapter 14 is that natural numbers can be naturally defined within object theory and the laws they abide by up to and including Second-Order Peano Arithmetic can be derived without having to appeal to any intrinsically mathematical axioms or notions.

We have reproduced parts of this construction in our implementation² and arrived at the following results:

- The construction of natural numbers is sound and the Dedekind-Peano postulates, including mathematical induction, are consistently derivable.
- We could model the additional axioms required for the construction in our framework.
- We could generalize one of the aforementioned axioms, strengthening the theoretical basis and justification of the construction.
- We can analyze variations of the construction that may be adopted in the future.

In particular, we can derive the Dedekind-Peano postulates about Natural Numbers as follows:

1. Zero is a natural number.
2. No natural number has Zero as its successor.
3. If a natural number k succeeds the numbers n and m , then $n = m$.
4. Every natural number has a Successor.
5. Mathematical Induction: If (1) Zero exemplifies a property F and (2) for any number n , it follows from the fact that n exemplifies F that the successor of n exemplifies F , then F is exemplified by all natural numbers.

¹As in the previous chapters, we refer to at the time of writing most recent version of PLM, dated October 13 2021, which will continue to be available at [63].

²At the time of writing the implementation encompasses the construction of natural numbers and the Dedekind-Peano postulates. We expect a full implementation of the derivation of Second-Order Peano Arithmetic in the foreseeable future.

Furthermore, the contributions to the general evolution of AOT we described in the previous chapters have had repercussions on the details of the construction. We will describe this interaction in more detail in the following sections, while reproducing the construction Nodelman and Zalta present in PLM chapter 14.

5.1. General Idea of the Construction

The strategy for constructing natural numbers in AOT basically follows the idea of Frege's Theorem (see [57]). Frege showed that the Peano axioms can be derived from *Hume's Principle* using Second-Order Logic. Hume's Principle states that the number of F s is equal to the number of G s if and only if F and G are *equinumerous*. Two relations are *equinumerous*, if and only if there is a one-to-one correspondence between them or, in other words, if and only if there is a bijection between the objects exemplifying F and the objects exemplifying G .

Frege himself derived Hume's Principle from *Basic Law V*, which together with second-order logic leads to Russel's Paradox. However, deriving Peano arithmetic *from* Hume's Principle itself does not require *Basic Law V*. In PLM's chapter 14, Nodelman and Zalta propose a definition of *equinumerosity* and *the number of F s* in object theory and are able to derive Hume's Principle. Based on that, they present a definition of Natural Numbers and a consistent derivation of the Dedekind-Peano postulates.

5.2. Equinumerosity of Relations

On the basis of traditional mathematical training based on set theory and functional logic, the seemingly most natural conception of *equinumerosity* involves the notion of a bijection. Two properties are equinumerous (i.e., intuitively, they are exemplified by "the same number" of objects), if and only if there is a bijection between the sets of objects exemplifying them.

However, this conception of equinumerosity relies on objects of theoretical mathematics and their axiomatization (sets, functions, bijections). While object theory can in fact define those notions as well, it takes relations to be the more primitive, fundamental concept and thereby prefers a definition in terms of relations alone.

The concept of there being a bijection between the sets of objects exemplifying two properties can equivalently be captured by the notion of there being a *one-to-one correspondence* between the properties.

5.2.1. One-to-One Correspondences

A *one-to-one correspondence* between the properties F and G is a relation R , s.t. (1) for every object x that exemplifies F , there is a unique object y exemplifying G , s.t. x bears

R to y and conversely (2) for every object y that exemplifies G , there is a unique object x exemplifying F , s.t. x bears R to y . Formally (see A.12.12):³

$$R \mid: F \text{ }_{1-1} \longleftrightarrow G \equiv \\ \forall x ([F]x \rightarrow \exists !y (([G]y \ \& \ [R]xy))) \ \& \ \forall y ([G]y \rightarrow \exists !x (([F]x \ \& \ [R]xy)))$$

The relation to a bijection is readily apparent: for any object exemplified by F , the relation R identifies a unique object exemplified by G and vice-versa.

However, this unrestricted notion of a one-to-one correspondence is not well-suited for a definition of equinumerosity that validates Hume's principle in AOT. The intuitive reason for this is that abstract objects cannot be counted. In particular, recall that there are distinct, but exemplification-indistinguishable abstract objects (see section 3.8.1 and A.7.8572):

$$\exists x \exists y (A!x \ \& \ A!y \ \& \ x \neq y \ \& \ \forall F ([F]x \equiv [F]y))$$

Based on this fact, we can prove that there is no one-to-one correspondence between $A!$ and itself:

AOT-theorem $\langle \neg \exists R \ R \mid: A! \text{ }_{1-1} \longleftrightarrow A! \rangle$

proof (*rule* *raa-cor:2*) — Proof by contradiction.

AOT-assume $\langle \exists R \ R \mid: A! \text{ }_{1-1} \longleftrightarrow A! \rangle$ — Assume the contrary.

then AOT-obtain R **where** 0 : $\langle R \mid: A! \text{ }_{1-1} \longleftrightarrow A! \rangle$ — Let R be a witness.

using $\exists E$ **by** *metis*

— By definition of a one-to-one correspondence it follows that:

AOT-hence $\langle \forall x ([A!]x \rightarrow \exists !y ([A!]y \ \& \ [R]xy)) \rangle$

using *1-1-cor*[*THEN* $\equiv_{df} E$] **&E** **by** *blast*

— Now let a and b be witnesses to the theorem cited above.

moreover AOT-obtain $a \ b$ **where** 1 : $\langle A!a \ \& \ A!b \ \& \ a \neq b \ \& \ \forall F ([F]a \equiv [F]b) \rangle$

using *aclassical2* $\exists E$ **by** *blast*

— Taken together, this means there has to be a unique abstract object to which a bears R .

ultimately AOT-have $\langle \exists !y ([A!]y \ \& \ [R]ay) \rangle$

using $\forall E(2)$ **&E** $\rightarrow E$ **by** *blast*

— Now let c be a witness, s.t. c is abstract and a bears R to c .

then AOT-obtain c **where** 2 : $\langle A!c \ \& \ [R]ac \rangle$

using $\&E(1)$ $\exists E$ *uniqueness:1*[*THEN* $\equiv_{df} E$] **by** *blast*

— By beta-conversion it follows that a exemplifies *being an x that bears R to c* .

AOT-hence $\langle [\lambda x [R]xc]a \rangle$

by (*auto intro!*: $\beta \leftarrow C$ *cqt:2* *dest:* $\&E$)

— Since by construction a and b exemplify the same properties, the same holds true for b .

AOT-hence $\langle [\lambda x [R]xc]b \rangle$

by (*safe intro!*: 1 [*THEN* $\&E(2)$, *THEN* $\forall E(1)$, *THEN* $\equiv E(1)$]) *cqt:2*[*lambda*]

— Again by beta conversion it follows that b bears R to c .

AOT-hence 5 : $\langle [R]bc \rangle$

using $\beta \rightarrow C$ **by** *blast*

³Note that as mentioned in section 3.4.2, instead of stating the original definitions-by-equivalence of AOT that involve additional significance clauses, we may instead illustrate the definitions in simpler form using derived equivalences formulated using object-level variables throughout this chapter. In each case the full definition in the appendix is referenced.

— Now the following is a consequence of the assumption that $A!$ is in one-to-one correspondence to itself:

AOT-have $\langle \forall x \forall y \forall z ([A!]x \ \& \ [A!]y \ \& \ [A!]z \rightarrow ([R]xz \ \& \ [R]yz \rightarrow x = y)) \rangle$
using $eq-1-1$ [*unverify* $F \ G$, $OF \ oa-exist:2$, $OF \ oa-exist:2$, $THEN \ \equiv E(1)$,
 $THEN \ fFG:4$ [$THEN \ \equiv_{df} E$], $THEN \ \&E(1)$,
 $THEN \ fFG:2$ [$THEN \ \equiv_{df} E$], $THEN \ \&E(2)$, $OF \ 0$].

— In particular this is true for a , b and c .

AOT-hence $\langle [A!]a \ \& \ [A!]b \ \& \ [A!]c \rightarrow ([R]ac \ \& \ [R]bc \rightarrow a = b) \rangle$
using $\forall E(2)$ **by** *blast*

— But we already established that a , b and c are abstract, as well as that a bears R to c and b bears R to c , so a and b have to be identical.

AOT-hence $\langle a = b \rangle$
using 1 [$THEN \ \&E(1)$] $2 \ 5 \ \&E \rightarrow E \ \&I$ **by** *metis*

— Which contradicts a and b being distinct by construction.

AOT-thus $\langle p \ \& \ \neg p \rangle$ **for** p
using $1 \ =-infix \ \equiv_{df} E \ \&I \ \&E \ raa-cor:1$ **by** *fast*

qed

So if *equinumerosity* was defined in terms of the existence of a full one-to-one correspondence, $A!$ would not be equinumerous to itself and consequently equinumerosity would not be an equivalence relation. However, Frege’s construction assumes that equinumerosity partitions all properties into equivalence classes, i.e. that equinumerosity *is* an equivalence relation. While it is an interesting question for future research, whether a variant of the construction was possible, in which equinumerosity merely was a partial equivalence relation (and consequently not all properties could be counted, resp. *the number of Fs* would not denote for every F), the construction in the current version of PLM at the time of writing chooses to stay closer to Frege’s original method. In particular, Nodelman and Zalta restrict their analysis to *ordinary objects* and we will therefore choose this option for the main discussion in this chapter.

In section 5.21 we will discuss alternative options to address the issue that may lead to an enhanced version of the construction in the future.

5.2.2. One-to-One Correspondences on the Ordinary Objects

As mentioned in the introduction of this chapter, natural mathematics arises from abstracting exemplification patterns. In case of natural numbers, those patterns in particular need to be among objects that can be counted. While abstract objects in general cannot, ordinary objects can always naturally be counted. Hence Nodelman and Zalta, following [60], introduce the notion of one-to-one correspondences *among the ordinary objects*. To that end, they introduce the restricted variables u , v , r , s that range over only the ordinary objects.⁴ Using these restricted variables, a one-to-one correspondence _{E} among the ordinary objects can be defined in the same way as a full one-to-one correspondence (see A.12.182):

⁴Recall the discussion of restricted variables in section 3.4.4.

$$R \mid: F \text{ }_{1-1} \longleftrightarrow_E G \equiv \\ \forall u ([F]u \rightarrow \exists !v (([G]v \ \& \ [R]uv))) \ \& \ \forall v ([G]v \rightarrow \exists !u (([F]u \ \& \ [R]uv)))$$

5.2.3. Definition of Equinumerosity

Based on one-to-one correspondences_E on the ordinary objects, *equinumerosity* on the ordinary objects can be defined as suggested above: two relations are equinumerous_E, if and only if there is a one-to-one correspondence_E on the ordinary objects between them (see A.12.187):⁵

$$F \approx_E G \equiv_{df} \exists R \ R \mid: F \text{ }_{1-1} \longleftrightarrow_E G$$

Equinumerosity on the ordinary objects is indeed an equivalence relation (see A.12.211):

$$\begin{aligned} F \approx_E F \\ F \approx_E G \rightarrow G \approx_E F \\ F \approx_E G \ \& \ G \approx_E H \rightarrow F \approx_E H \end{aligned}$$

Reflexivity can be shown by using the identity on the ordinary objects ($=_E$) (see 3.4.5) as witness for the existence of a one-to-one-correspondence_E between any property and itself. Note that this is only possible, since, in contrast to general identity, identity on the ordinary objects constitutes a (denoting) relation.

Symmetry is a simple consequence of the symmetry of the definition of one-to-one correspondences_E.

Transitivity requires a slightly more verbose proof (see A.12.276), that hinges on the fact that $[\lambda xy \ O!x \ \& \ O!y \ \& \ \exists v ([G]v \ \& \ [R_1]xv \ \& \ [R_2]vy)]$ can be chosen as a witness for the existence of a one-to-one-correspondence_E between F and H , if R_1 is a one-to-one-correspondence_E between F and G and R_2 is a one-to-one-correspondence_E between G and H .

5.2.4. Properties of Equinumerosity

Nodelman and Zalta continue to derive a variety of properties of equinumerosity that are helpful for the remainder of the construction. While a full account of the progression of theorems can be found in PLM, respectively in our implementation in A.12, the following is a selection of noteworthy auxiliary theorems:

Properties that are unexemplified on the ordinary objects are equinumerous (any relation may serve as one-to-one-correspondence_E between them; see A.12.712):

$$\neg \exists u [F]u \ \& \ \neg \exists v [H]v \rightarrow F \approx_E H$$

A property F that is exemplified by some ordinary object is *not* equinumerous to a property H that is not exemplified by any ordinary object (proof by contradiction, since

⁵In the following sections we will drop the explicit mention of the restriction to the ordinary objects and simply talk about *equinumerosity* and being *equinumerous* instead of *equinumerosity on the ordinary objects*, resp. *equinumerous_E*.

the existence of a one-to-one correspondence_E would imply that H is exemplified by an ordinary object; see A.12.737):

$$\exists u [F]u \ \& \ \neg \exists v [H]v \rightarrow \neg F \approx_E H$$

Respectively, removing (adding) ordinary objects from (to) a pair of equinumerous properties yields equinumerous properties (see A.12.772):⁶

$$\begin{aligned} F \approx_E G \ \& \ [F]u \ \& \ [G]v \rightarrow F^{-u} \approx_E G^{-v} \\ F^{-u} \approx_E G^{-v} \ \& \ [F]u \ \& \ [G]v \rightarrow F \approx_E G \end{aligned}$$

Properties that are equivalent on the ordinary objects (written as \equiv_E) are equinumerous (see A.10.74, A.12.1755):⁷

$$\begin{aligned} F \equiv_E G \equiv \forall u ([F]u \equiv [G]u) \\ F \equiv_E G \rightarrow F \approx_E G \end{aligned}$$

5.2.5. Modal Properties of Equinumerosity

It is noteworthy that, in general, equinumerosity is *not* modally rigid. In particular, it is provable that there are relations that are possibly equinumerous and possibly not equinumerous (see A.12.1514):

$$\exists F \exists G \ (\diamond(F \approx_E G \ \& \ \diamond \neg F \approx_E G))$$

As a simple example consider a property that is necessarily unexemplified and another property that is *actually* unexemplified, but *possibly* exemplified by some ordinary object.⁸ While such properties are equinumerous in the actual world, there is no one-to-one-correspondence_E between them in the possible world, in which the second property is exemplified by an object.

We will see in the next section that for this reason it makes sense to use the actual world as a reference for the definition of *numbering properties*.

In any modal context, it is possible to express equinumerosity relative to the behaviour of properties in the actual world. In particular the following is a (modally-strict) theorem (see A.12.1896):

$$F \approx_E G \equiv \forall H ([\lambda z \mathcal{A}[H]z] \approx_E F \equiv [\lambda z \mathcal{A}[H]z] \approx_E G)$$

I.e. two properties F and G are equinumerous, if and only if for all properties H , F is equinumerous to *actually exemplifying* H just in case that G is. Furthermore, for *rigid* properties,⁹ equinumerosity is modally collapsed (see A.12.1951):

$$Rigid(F) \ \& \ Rigid(G) \rightarrow \square(F \approx_E G \rightarrow \square F \approx_E G)$$

⁶The statements rely on the following definition of F^{-x} , i.e. *being an F that is not x* : $F^{-x} =_{df} [\lambda z [F]z \ \& \ z \neq_E x]$. The proofs rely on constructing suitable one-to-one correspondences_E by case analysis.

⁷The identity on the ordinary objects ($=_E$) can be chosen as witness for the existence of a one-to-one-correspondence_E.

⁸The existence of such properties is guaranteed by the fact that by axiom there is an object that is not actually, but possibly concrete as mentioned in section 3.3.

⁹I.e. properties that are modally collapsed: $Rigid(F) \equiv \square \forall x ([F]x \rightarrow \square[F]x)$, see also 3.8.3.

The proofs of the last two theorems hinge on the existence of *rigidifying* relations. Recall the earlier discussion of this topic in section 3.8.3 - notably, in earlier versions of PLM, the existence of rigidifying relations had to be ensured by axiom. In the current formulation of AOT, the necessary and sufficient conditions for relations to denote that we contributed to the theory (see section 3.8.2), can be used to prove the existence of *world-indexed properties* that can serve as witnesses for the existence of rigidifying relations, thereby eliminating the need for the additional axiom.

5.3. The Number of Fs and Hume's Theorem

To state Hume's Theorem, additionally to the definition of *equinumerosity* above, a definition of *The Number of Fs* (written as $\#F$) is required. To that end Nodelman and Zalta first define what it means for an object to number a property as follows (see A.12.2177):

$$\text{Numbers}(x, G) \equiv A!x \ \& \ \forall F \ (x[F] \equiv [\lambda z \ \mathcal{A}[F]z] \approx_E G)$$

An abstract object x numbers a property G , if it encodes exactly those properties, such that *actually exemplifying* them is equinumerous to G . An alternative choice would be to forgo the actuality operator and merely require that x encodes exactly those properties that are themselves equinumerous to G .¹⁰ However, we noted in the last section that equinumerosity is (in general) not modally rigid, so such a definition would have the undesirable consequence that numbering properties would depend on modal context and consequently that every possible world would need its own sets of numbers (see 5.5). To avoid this issue the actual world is used as a reference. Later in this chapter, we will show that this does *not* mean that it is impossible to count in non-actual worlds and that this definition is consistent with the pretheoretic intuition of one group of natural numbers that can count objects at any possible world (see 5.5).

Now *The Number of Gs* can simply be defined as *the* object that numbers G (see A.12.2503):

$$\#G =_{df} \iota x \text{Numbers}(x, G)$$

Using these definitions Hume's principle becomes derivable as a theorem (see A.12.2589):

$$\vdash \#F = \#G \equiv F \approx_E G$$

Note that, due to the fact that AOT's definite descriptions are modally rigid and refer to objects in the actual world, this theorem is not modally strict.¹¹ However, the following variants are necessary facts with modally-strict proofs (see A.12.2602):

$$\begin{aligned} & \exists x (\text{Numbers}(x, F) \ \& \ \text{Numbers}(x, G)) \equiv F \approx_E G \\ & \exists x \exists y (\text{Numbers}(x, F) \ \& \ \forall z (\text{Numbers}(z, F) \rightarrow z = x) \ \& \ \text{Numbers}(y, G) \ \& \\ & \quad \forall z (\text{Numbers}(z, G) \rightarrow z = y) \ \& \\ & \quad x = y) \equiv \\ & F \approx_E G \end{aligned}$$

¹⁰In fact, earlier versions of the construction used this definition (see e.g. [60]).

¹¹Recall that this is signified by the turnstile symbol \vdash and recall the discussion in section 3.4.7.

Note that the last theorem corresponds to a translation of the descriptions in Hume's theorem according to Russell's analysis of definite descriptions.

5.4. The Number Zero

Given the fact that we defined numbers by means of the properties they number, which in turn is - informally speaking - based on how many objects those properties exemplify, a natural definition of the number Zero arises. The number Zero is the object that numbers the empty property, to be more precise the number of *being a non-self-identical ordinary object* (see A.12.2683).¹²

$$0 =_{df} \#[\lambda x O!x \ \& \ x \neq_E x]$$

Note that while the above definition introduces the number Zero as (abstract) object, we have not defined the notion of a *Natural Number* yet, nor shown that the number Zero indeed *is* a natural number. The definition of *Natural Number* will rely on introducing a *predecessor* relation and, intuitively speaking, defining that an abstract object is a natural number if there is a series of objects starting at Zero, ending at the given abstract object, s.t. two consecutive objects in that series bear the predecessor relation to each other. While we will describe this construction in detail in the following sections, we can already define the strictly more general¹³ notion of a *Natural Cardinal* and it will immediately follow that Zero is a natural cardinal. An object x is a natural cardinal, just in case that there is a property G , s.t. x is the number of G s (see A.12.2570):

$$NaturalCardinal(x) \equiv_{df} \exists G \ x = \#G$$

By the definition of the number Zero, it becomes immediately apparent that Zero is a natural cardinal (see A.12.2688):¹⁴

$$NaturalCardinal(0)$$

¹²To be precise being a non-self-identical_E object (see section 3.4.5). This distinction is non-trivial: While $O!x \ \& \ x \neq_E x \equiv O!x \ \& \ x \neq x$ is a theorem, due to the hyperintensionality of object theory, it does not have to be the case that $[\lambda x O!x \ \& \ x \neq_E x]$ and $[\lambda x O!x \ \& \ x \neq x]$ are the same property (as a matter of fact it is not even asserted *a priori* that the latter even denotes a property at all). So $\#[\lambda x O!x \ \& \ x \neq_E x]$ and $\#[\lambda x O!x \ \& \ x \neq x]$ are not the same object *a priori*, even though it is a theorem that they are identical. But this theorem has to appeal to the fact that both properties are equinumerous and to Hume's Theorem. Further examples of terms denoting the number Zero are $\#[\lambda x x \neq x]$ and $\#[\lambda x \exists p (p \ \& \ \neg p)]$.

¹³It is a theorem that $\#O!$ is a natural cardinal that is infinite and not a natural number (see A.12.5456).

¹⁴However, note that the proof has to appeal to the fact that $\#G\downarrow$ (see A.12.2505) as well as the fact that $[\lambda x O!x \ \& \ x \neq_E x]\downarrow$ by axiom.

5.5. Counting in Possible Worlds

In section 5.3, we mentioned the use of the actual world as reference for defining numbering properties and hinted at the fact that this is justified and consistent with pretheoretic intuition. We can now discuss this in more detail at the example of the number Zero.

The number of a property is defined as rigid definite description and thereby uses the actual world as frame of reference. In particular, using the number Zero as example, this means the following (see A.12.2918):

$$\neg\exists u \mathcal{A}[F]u \equiv \#F = 0$$

If and only if a property F is not *actually* exemplified by any ordinary object, the number of that property is Zero. This may seem counter-intuitive at first: the stated theorem is modally-strict and thereby a *necessary* fact. So in any possible world, even in worlds in which F could be exemplified, the number of F is still Zero, if F is not *actually* exemplified. However, this is merely due to the fact that definite descriptions are rigid and themselves refer to objects in the actual world.

Moving away from the rigidly defined *number of Fs*, it is a modally-strict theorem (and thereby a *necessary* fact), that Zero numbers any property that is not exemplified by any ordinary object (see A.12.2844):

$$\begin{aligned} \neg\exists u [F]u &\equiv \text{Numbers}(0, F) \\ \Box(\neg\exists u [F]u &\equiv \text{Numbers}(0, F)) \end{aligned}$$

I.e. Zero numbers empty properties in all possible worlds. A different take on this is the fact that any object that *possibly* numbers a necessarily empty property is the number Zero (see A.13.5):

$$\Diamond \text{Numbers}(x, [\lambda z O!z \ \& \ z \neq_E z]) \rightarrow x = 0$$

By contrast, if numbering a property had been defined without using the actual world as reference, then "the" number Zero would be a different abstract object in different possible worlds:

If we define *Numbers'* without the use of the actuality operator, s.t.:

$$\text{Numbers}'(x, G) \equiv A!x \ \& \ \forall F (x[F] \equiv F \approx_E G)$$

Then it is a theorem (see A.13.46) that:

$$\begin{aligned} \exists x \exists y (\Diamond \text{Numbers}'(x, [\lambda z O!z \ \& \ z \neq_E z]) \ \& \ \Diamond \text{Numbers}'(y, [\lambda z O!z \ \& \ z \neq_E z]) \ \& \\ x \neq y) \end{aligned}$$

I.e. there would be distinct abstract objects that might count necessarily empty properties. This is clearly contrary to the pretheoretic intuition that numbers are universal, i.e. that counting empty properties in any world will yield one and the same number Zero.

On the other hand, we can further consolidate the use of the actual world as reference frame, by realizing that we *can* talk about the numbers of properties in different worlds, despite the rigidity of definite descriptions and the use of the actual world as reference for defining numbers. We again use the number Zero as example and can show that if

and only if a property F is not exemplified in a given possible world w , then *the number of exemplifying F at w* is Zero (see A.12.2987):¹⁵

$$w \models \neg \exists u [F]u \equiv \#F_w = 0$$

5.6. Ancestral Relations and Transitive Closures

As mentioned above, natural numbers will, informally speaking, be defined by the means of series of objects that bear a (yet to be introduced) predecessor relation to each other. However, traditionally, a series of objects relies on it being possible to index its objects using a continuous sequence of natural numbers. Since our goal is to *define* natural numbers, using this traditional notion of a series is not an option. Instead we construct *ancestral relations*. In particular the *strong ancestral* of a relation will match the concept of the transitive closure of the relation. Natural numbers will be defined as the objects to which the number Zero bears the *weak ancestral* (i.e. the transitive and reflexive¹⁶ closure) of the predecessor relation, i.e. the number Zero itself or any object that is transitively preceded by Zero.

The first step in this process is to define being a *hereditary* property with respect to a relation, which will lead to a definition of the *strong ancestral* of a relation.

5.6.1. Properties that are Hereditary with respect to a Relation

A property F is *hereditary* w.r.t. a relation R , if and only if for every pair of objects x and y , s.t. x bears R to y , if x exemplifies F , then y exemplifies F (see A.12.3117):

$$\text{Hereditary}(F, R) \equiv \forall x \forall y ([R]xy \rightarrow ([F]x \rightarrow [F]y))$$

Intuitively, a relation R defines sequences of objects as follows: we call a list of objects x_1, \dots, x_n an R -induced sequence, if for every $0 < i < n$, x_i bears R to x_{i+1} . Then F being hereditary w.r.t. R means that any R -induced sequence starting in F (i.e. starting with an object exemplified by F), is completely contained in F (i.e. every object in the sequence exemplifies F as well).

Or in other words, a property F is hereditary w.r.t a relation R , if " F -ness" is inherited from all objects that exemplify F to the objects that are R -related to them.

5.6.2. Strong Ancestral of a Relation and Transitive Closures

Using the above definition, we can introduce the *Strong Ancestral* of a relation R , which is written as R^* (see A.12.3125):¹⁷

¹⁵Recall the discussion of AOT's theory of Possible Worlds and world-indexed properties in section 3.8.3.

¹⁶We will see that reflexivity will have to be restricted to a specific domain.

¹⁷Note that while PLM uses R^* for the strong ancestral, i.e. the transitive closure, of R and later R^+ for the weak ancestral, i.e. the transitive and reflexive closure, of R , Isabelle's HOL library uses the opposite convention, i.e. it uses r^+ as transitive and r^* as reflexive-transitive closure.

$$R^* =_{df} [\lambda xy \forall F (\forall z ([R]xz \rightarrow [F]z) \& Hereditary(F,R) \rightarrow [F]y)]$$

An object x bears R^* to y , just in case that y exemplifies every property F that is hereditary w.r.t R and that is exemplified by all objects to which x bears R . To convince ourselves that this definition captures the transitive closure of R , we may fix x and consider a property F , s.t. $\forall z ([R]xz \rightarrow [F]z) \& Hereditary(F,R)$. Any such property F is exemplified by all objects immediately following x in an R -induced sequence (first conjunct) as well as all objects in any longer R -induced sequence starting at x (second conjunct). Hence (informally thinking of properties as sets) any such F contains all objects that are transitively related to x . Note, however, that F may exemplify additional objects that are not part of any R -induced sequence. However, the intersection of all such properties F yields the smallest set of objects that are transitively related to x , which is *exactly* those objects that *are* transitively related to x .

It is interesting to note that there is a different way to define the transitive closure of a relation R , namely:

The transitive closure of a relation R is the intersection of all transitive relations R' that are contained in R . As a matter of fact, we can state this definition in AOT as well, and prove it to be equivalent to the strong ancestral of R .

First we define for a relation to be transitive as follows:

AOT-define *Transitive* :: $\langle \tau \Rightarrow \varphi \rangle (\langle Transitive'(-) \rangle)$
 $\langle Transitive(R) \equiv_{df} \forall x \forall y \forall z ([R]xy \& [R]yz \rightarrow [R]xz) \rangle$

Next we can define for a relation to be contained in another relation, or alternatively, moving away from set-theoretic concepts, for a relation to entail another relation:

AOT-define *Entails* :: $\langle \tau \Rightarrow \tau \Rightarrow \varphi \rangle (\langle Entails'(-,-) \rangle)$
 $\langle Entails(R,R') \equiv_{df} \forall x \forall y ([R]xy \rightarrow [R']xy) \rangle$

Being in the intersection of all transitive relations entailed by R means exemplifying all of them. Hence we can define the transitive closure of R as follows:

AOT-define *TransitiveClosure* :: $\langle \tau \Rightarrow \Pi \rangle (\langle TransitiveClosure'(-) \rangle)$
 $\langle TransitiveClosure(R) =_{df} [\lambda xy \forall R' (Transitive(R') \& Entails(R,R') \rightarrow [R']xy) \rangle$

Now we can prove that this definition of a transitive closure is equivalent to the definition of a strong ancestral above:

AOT-theorem $\langle [TransitiveClosure(R)]xy \equiv [R^*]xy \rangle$

proof (*safe intro!*: $\equiv I \rightarrow I$)

AOT-assume $\langle [TransitiveClosure(R)]xy \rangle$

AOT-hence $\langle [\lambda xy \forall R' (Transitive(R') \& Entails(R,R') \rightarrow [R']xy)]xy \rangle$

by (*auto intro: rule-id-df:2:a[OF TransitiveClosure] intro!: cqt:2*)

AOT-hence $\langle \forall R' (Transitive(R') \& Entails(R,R') \rightarrow [R']xy) \rangle$

using $\beta \rightarrow C$ **by fast**

AOT-hence $\langle Transitive(R^*) \& Entails(R,R^*) \rightarrow [R^*]xy \rangle$

using $\forall E(1)$ *rule-id-df:2:b[OF ances-df] hered:2* **by blast**

— The following is a consequence of PLM's theorems about strong ancestral relations (see A.12.3136 and A.12.3222).

```

moreover AOT-have  $\langle \text{Transitive}(R^*) \ \& \ \text{Entails}(R, R^*) \rangle$ 
  by (auto intro!:  $\&I \ \text{Entails}[\text{THEN} \equiv_{df} I] \ \text{Transitive}[\text{THEN} \equiv_{df} I] \ \text{GEN} \rightarrow I$ 
    simp: anc-her:1[ $\text{THEN} \rightarrow E$ ] anc-her:6[ $\text{THEN} \rightarrow E$ ])
ultimately AOT-show  $\langle [R^*]xy \rangle$ 
  using  $\rightarrow E \ \&I$  by blast
next
AOT-assume  $0$ :  $\langle [R^*]xy \rangle$ 
AOT-have  $\langle \forall R' (\text{Transitive}(R') \ \& \ \text{Entails}(R, R') \rightarrow [R']xy) \rangle$ 
proof (safe intro!:  $\text{GEN} \rightarrow I$ ; frule  $\&E(1)$ ; drule  $\&E(2)$ )
  fix  $R'$ 
  AOT-assume transitive:  $\langle \text{Transitive}(R') \rangle$  and entails:  $\langle \text{Entails}(R, R') \rangle$ 
  — The following is an instance of another theorem about strong ancestral relations (see A.12.3148).
  AOT-have  $\langle [R^*]xy \ \& \ \forall z ([R]xz \rightarrow [\lambda z [R']xz]z) \ \& \ \text{Hereditary}([\lambda z [R']xz], R) \rightarrow [\lambda z [R']xz]y \rangle$ 
  by (rule anc-her:2[unvarify  $F$ ] cqt:2[lambda])
  moreover AOT-have  $\langle \text{Hereditary}([\lambda z [R']xz], R) \rangle$ 
  proof (safe intro!: hered:1[ $\text{THEN} \equiv_{df} I$ ]  $\&I$  cqt:2  $\text{GEN} \rightarrow I$ )
  fix  $z \ y$ 
  AOT-assume  $\langle [R]zy \rangle$  and  $\langle [\lambda z [R']xz]z \rangle$ 
  AOT-hence  $\langle [R']zy \rangle$  and  $\langle [R']xz \rangle$ 
  using entails by (auto dest:  $\text{Entails}[\text{THEN} \equiv_{df} E] \ \forall E(2) \rightarrow E \ \beta \rightarrow C$ )
  AOT-hence  $\langle [R']xy \rangle$ 
  using transitive by (auto dest!:  $\text{Transitive}[\text{THEN} \equiv_{df} E] \ \text{dest}: \forall E(2) \rightarrow E \ \text{intro!}: \&I$ )
  AOT-thus  $\langle [\lambda z [R']xz]y \rangle$ 
  by (auto intro!:  $\beta \leftarrow C$  cqt:2)
  qed
moreover AOT-have  $\langle \forall z ([R]xz \rightarrow [\lambda z [R']xz]z) \rangle$ 
  using entails[ $\text{THEN} \ \text{Entails}[\text{THEN} \equiv_{df} E]$ ]
  by (auto intro!:  $\text{GEN} \rightarrow I \ \beta \leftarrow C$  cqt:2 dest:  $\forall E(2) \rightarrow E$ )
  ultimately AOT-have  $\langle [\lambda z [R']xz]y \rangle$ 
  using  $0 \ \&I \rightarrow E$  by auto
  AOT-thus  $\langle [R']xy \rangle$ 
  by (rule  $\beta \rightarrow C$ )
  qed
AOT-thus  $\langle [\text{TransitiveClosure}(R)]xy \rangle$ 
  by (auto intro: rule-id-df:2:b[OF  $\text{TransitiveClosure}$ ]
    intro!:  $\beta \leftarrow C$  cqt:2 tuple-denotes[ $\text{THEN} \equiv_{df} I, \ \text{OF} \ \&I$ ])
  qed

```

5.7. Weak Ancestral Relations

As mentioned above, our goal is to define being a natural number as either being Zero or being an object, s.t. Zero bears the strong ancestral of the to-be-defined predecessor relation to it. This matches the notion of the *weak ancestral* of the predecessor relation. Traditionally, the weak ancestral of a relation R^+ is defined, s.t. an object x bears R^+ to an object y , if and only if either x bears the strong ancestral R^* to y or $x = y$.

However, recall that in AOT there is no general relation of identity, i.e. $[\lambda xy x = y]$ does not denote (see 3.8.1). Consequently, the immediate candidate for defining the weak ancestral of a relation $[\lambda xy [R^*]xy \vee x = y]$ does not denote for arbitrary choices of R .¹⁸ For this reason Nodelman and Zalta proceed by introducing *rigid one-to-one relations*. Rigid one-to-one relations induce a notion of identity on their *domain* that is consistent with general identity (on this domain), but constitutes a denoting relation.

5.7.1. Rigid One-to-One Relations

For a relation to be *one-to-one* is related to the notion of a function being injective. A relation R is *one-to-one*, if whenever two objects x and y bear R to the same object z , then x and y are identical (see A.12.3256):

$$1-1(R) \equiv \forall x \forall y \forall z ([R]xz \ \& \ [R]yz \rightarrow x = y)$$

Note, however, that one-to-one relations are more general than injective functions, since the criterion to be one-to-one does not imply that the relation is *functional*, i.e. that each object in its domain is related to exactly one object.

An object x is in the domain of a relation R , just in case that there is an object y , s.t. x bears R to y (see A.12.3322):

$$InDomainOf(x,R) \equiv \exists y [R]xy$$

While the predecessor relation will in fact be a functional relation, a relation that relates a single object to all other objects, but no other object to any object, is an example of a one-to-one relation that's succinctly non-functional. However, in order to introduce a restricted notion of identity, functionality is not a requirement.

On the other hand, in order to simplify modal reasoning and to be able to introduce well-behaved restricted variables, it is helpful to only consider *rigid* one-to-one relations. A rigid one-to-one relation is a relation that is one-to-one and rigid (see A.11.2995, A.12.3259):¹⁹

$$Rigid_{1-1}(R) \equiv 1-1(R) \ \& \ Rigid(R)$$

Since being a rigid one-to-one relation is a rigid restriction condition, we can introduce well-behaved restricted variables that range over them.²⁰

In the following we will use \mathcal{R} as a restricted variable ranging over rigid one-to-one relations.²¹

5.7.2. Identity Restricted to the Domain of Rigid One-to-one Relations

For a variable \mathcal{R} that is restricted to rigid one-to-one relations, a restricted notion of identity can now be defined as follows (see A.12.3372):

¹⁸For example, if R is a necessarily empty relation, the matrix of $[\lambda xy [R^*]xy \vee x = y]$ is necessarily equivalent to $[\lambda xy x = y]$ for all x and y and $[\lambda xy [R^*]xy \vee x = y]$ fails to denote by co-existence.

¹⁹Recall the discussion about rigid relations in section 3.8.3.

²⁰Recall the discussion of restricted variables in section 3.4.4.

²¹Note that PLM uses \underline{R} . However, in our framework choosing \mathcal{R} is simpler for technical reasons.

$$(=_{\mathcal{R}}) =_{df} [\lambda xy \exists z ([\mathcal{R}]xz \ \& \ [\mathcal{R}]yz)]$$

Note that in contrast to general identity, the definiens of \mathcal{R} -identity (trivially) denotes a proper relation.

By β -conversion and using infix notation, two objects x and y are \mathcal{R} -identical, just in case that there is an object to which they are both \mathcal{R} -related (see A.12.3379):

$$x =_{\mathcal{R}} y \equiv \exists z ([\mathcal{R}]xz \ \& \ [\mathcal{R}]yz)$$

Since \mathcal{R} is restricted to rigid one-to-one relations, the resulting identity relation is exactly the restriction of general identity to the domain of \mathcal{R} (see A.13.266):

$$x =_{\mathcal{R}} y \equiv InDomainOf(x, \mathcal{R}) \ \& \ InDomainOf(y, \mathcal{R}) \ \& \ x = y$$

Consequently, the defined identity is a partial equivalence relation that is reflexive on the domain of \mathcal{R} (see A.12.3470):

$$InDomainOf(x, \mathcal{R}) \rightarrow x =_{\mathcal{R}} x$$

$$x =_{\mathcal{R}} y \rightarrow y =_{\mathcal{R}} x$$

$$x =_{\mathcal{R}} y \ \& \ y =_{\mathcal{R}} z \rightarrow x =_{\mathcal{R}} z$$

A simple example of a rigid one-to-one-relation is the identity on the ordinary objects ($=_E$), the domain of which is the ordinary objects (see A.13.768 and A.13.788).

5.7.3. The Weak Ancestral of a Relation

Based on the concept of \mathcal{R} -identity, the *weak ancestral* \mathcal{R}^+ of a rigid one-to-one relation \mathcal{R} can be defined as follows (see A.12.3529):

$$\mathcal{R}^+ =_{df} [\lambda xy [\mathcal{R}^*]xy \vee x =_{\mathcal{R}} y]$$

Restricting to the domain of \mathcal{R} , two objects are now exactly in the weak ancestral relation \mathcal{R}^+ if they are either transitively \mathcal{R} -related (i.e. in the strong ancestral relation \mathcal{R}^*) or identical:

$$InDomainOf(x, \mathcal{R}) \rightarrow ([\mathcal{R}^+]xy \equiv [\mathcal{R}^*]xy \vee x = y)$$

In other words, the weak ancestral of a relation is its transitive and reflexive closure, with reflexivity being restricted to the domain of the relation.

5.8. Generalized Induction

In order to understand the formulation of generalized induction, first consider the following theorem that Nodelman and Zalta prove before even introducing weak ancestral relations, but which already has "inductive character" (see A.12.3160):

$$[F]x \ \& \ [R^*]xy \ \& \ Hereditary(F, R) \rightarrow [F]y$$

While this may not look like an inductive principle as stated, unfolding the definition of *Hereditary*, this is equivalent (under some trivial transformations) to the following:

AOT-theorem *pre-ind'*: $\langle [F]z \ \& \ \forall x \forall y ([R]xy \rightarrow ([F]x \rightarrow [F]y)) \rightarrow \forall y ([R]^*zy \rightarrow [F]y) \rangle$
proof (*safe intro!*: $\rightarrow I$ *GEN*)
 fix y
 AOT-assume $\langle [F]z \ \& \ \forall x \forall y ([R]xy \rightarrow ([F]x \rightarrow [F]y)) \rangle$
 AOT-hence $\langle [F]z \ \& \ Hereditary(F,R) \rangle$
 by (*AOT-subst-def hered:1*) (*auto intro!*: $\&I$ *cqt:2 elim:* $\&E$)
 moreover AOT-assume $\langle [R]^*zy \rangle$
 moreover AOT-have $\langle [F]z \ \& \ [R]^*zy \ \& \ Hereditary(F,R) \rightarrow [F]y \rangle$
 using *anc-her:3*. — This is an instance of the theorem cited above.
 ultimately AOT-show $\langle [F]y \rangle$
 using $\&I$ $\&E$ $\rightarrow E$ **by** *metis*
qed

I.e. if an object z exemplifies F and F is inherited via R , then any object that is transitively R -related to z exemplifies F .

Pretend for a moment that R was a well-defined predecessor relation and z the number Zero. Then this theorem would imply that if (1) F holds for Zero and (2) for any x and y , s.t. x precedes y , if x exemplifies F , then y exemplifies F , then F holds for all numbers transitively preceded by Zero (and since F also holds for Zero by assumption this would trivially imply that F holds for any natural number).

In principle, this is how mathematical induction will be derived. However, it is inconvenient that the induction step in this formulation ranges over the full domain of all objects. Instead, it should be sufficient to consider all natural numbers.

By instantiating F to $[\lambda x [F]x \ \& \ [\mathcal{R}^+]zx]$, we arrive at the following theorem, which PLM refers to as *Generalized Induction* (see A.12.3851):²²

$$[F]z \ \& \ \forall x \ \forall y ([\mathcal{R}^+]zx \ \& \ [\mathcal{R}^+]zy \rightarrow ([\mathcal{R}]xy \rightarrow ([F]x \rightarrow [F]y))) \rightarrow \forall x ([\mathcal{R}^+]zx \rightarrow [F]x)$$

In this formulation, the induction step merely ranges over objects to which z bears the weak ancestral relation of \mathcal{R} . Thinking of \mathcal{R} as the predecessor relation and z as the number Zero, this will be exactly the natural numbers. I.e. instantiating this generalized principle of induction to the predecessor relation and the number Zero, yields classical mathematical induction (relative to the upcoming definition of natural numbers).

5.9. The Predecessor Relation

5.9.1. Definition

While the definition of the predecessor relation is rather straightforward, the interesting question will be whether it actually denotes a relation, which we will discuss in detail

²²Note that (1) $[\mathcal{R}^+]zy$ for any y implies $[\mathcal{R}^+]zz$, yielding $[\lambda x [F]x \ \& \ [\mathcal{R}^+]zx]z$ in all cases in which the consequent of the strengthened theorem does not trivially hold (i.e. if $\neg \exists y [\mathcal{R}^+]zy$) and (2) that the consequent of the weaker theorem can be strengthened since $[\mathcal{R}^+]zy$ either implies (a) $z = y$, in which case $[F]y$ follows from the assumption $[F]z$, or it implies (b) $[\mathcal{R}^*]zy$, in which case the consequent of the weaker principle yields $[F]y$. The additional fact that $[\mathcal{R}]xy$ and $[\mathcal{R}^+]zx$ imply $[\mathcal{R}^+]zy$ is sufficient to arrive at the strengthened theorem.

in section 5.9.2. For the moment assume that the λ -expression in the definiens of the following definition denotes (see A.12.4288):

$$\mathbb{P} =_{df} [\lambda xy \exists F \exists u ([F]u \ \& \ Numbers(y,F) \ \& \ Numbers(x,F^{-u})]$$

Given the assumption that this relation denotes, it follows by β -conversion that (see A.12.4294):

$$\mathbb{P}xy \equiv \exists F \exists u ([F]u \ \& \ Numbers(y,F) \ \& \ Numbers(x,F^{-u}))$$

So an object x precedes an object y just in case there is a property F and an ordinary object u , s.t. u exemplifies F , y numbers F and x numbers being an F other than u (via the definition $F^{-u} =_{df} [\lambda z [F]z \ \& \ z \neq_E u]$).

This is a variant of Frege's definition of the successor relation.²³ The idea can be clarified by considering how the first natural numbers are related w.r.t. this relation:

- The number Zero numbers properties that are not exemplified by any ordinary object. Hence there cannot be a property F that is exemplified by an object u , s.t. Zero numbers F , which means that Zero is not preceded by any object.
- The number One numbers properties that are exemplified by a single ordinary object.²⁴ Hence any property F numbered by One is exemplified by some ordinary object u . Furthermore, F^{-u} , i.e. being an object exemplifying F other than u , is not exemplified by any ordinary object. Hence Zero is the unique predecessor of One.
- The number Two numbers properties that are exemplified by two distinct ordinary objects. Being an object that exemplifies any of these properties other than any particular object the given property exemplifies, is a property exemplified by only one ordinary object, hence numbered by One, i.e. One precedes Two, etc.

5.9.2. Assuring that the Predecessor Relation Denotes

It is important to realize that the λ -expression used in the definition above does not trivially denote a relation in AOT. *Numbering a property* is an encoding claim: an object numbers a property G , just in case it encodes all properties, s.t. actually exemplifying it is equinumerous to G . In general, encoding claims cannot be abstracted to denoting properties and relations.²⁵

In fact it is easy to see that the minimal model of AOT does not validate this axiom: the minimal model contains one ordinary urelement (resp. one ordinary object) and one special urelement. Since special urelements determine the exemplification extensions of abstract objects, there being only one special urelement implies that all abstract objects

²³Nodelman and Zalta argue in favour of a predecessor relation due to the fact that in contrast to a successor relation, the argument order of the predecessor relation matches the numerical order of objects in the relation. Apart from that, the notions are interchangeable, i.e. *Succeeds*(y,x) is exactly $\mathbb{P}xy$.

²⁴While we have not formally introduced any number other than Zero, we consider this intuitive understanding helpful in clarifying the idea of the predecessor relation. The number One will formally be introduced later in this chapter.

²⁵Recall the discussion in section 3.6.

exemplify the same properties and relations. This implies in particular that either all objects are preceded by Zero (including Zero itself) or no object is, i.e. $\mathbb{P}00$ or $\neg\exists x \mathbb{P}0x$. However, we have already (informally) argued above that Zero is not preceded by any object.²⁶ Hence in this model it would have to hold that $\neg\exists x \mathbb{P}0x$. However, since the minimal model still contains one ordinary object, the number One can be constructed and (again as argued above) is preceded by Zero, i.e. $\mathbb{P}01$, which yields a contradiction. Nodelman and Zalta assert that the predecessor relation denotes by axiom and emphasize that the relation is not inherently mathematical and no mathematical primitives are needed to assert, as an axiom, that it denotes (see PLM item (782)). In particular, they argue that expressions of the form $Numbers(y,F)$, while seemingly mathematical in nature, can be eliminated, since they are *defined* in terms of primitives of AOT. Furthermore, they argue that the relation merely asserts the existence of an ordering relation on abstract objects and ordering relations can, in general, be expressed in entirely logical terms.

However, even if one concedes that the axiom is not inherently mathematical, it can be objected that it is rather *ad-hoc*: rather than asserting a general principle according to which encoding claims can be abstracted to relations, it singles out a specific relation and this relation is, after all, used to *define* a concept that is very much mathematical in nature. Furthermore, the axiom is not trivially consistent: as we have seen, minimal models of the base system of AOT do not validate it.

Using our embedding we can, however, contribute to this situation in two ways:

- We can show that the axiom is consistent by constructing models that validate it.
- We can generalize the axiom to an independently justifiable comprehension principle for relations among abstract objects, s.t. it becomes a theorem that the predecessor relation in particular denotes.

We defer a more detailed discussion to section 5.19 and in the following continue to reproduce the construction of natural numbers and the derivation of the Dedekind-Peano postulates as given by Nodelman and Zalta in PLM.

5.9.3. The Predecessor Relation as Rigid One-to-One Relation.

It can be derived that the predecessor relation is modally rigid: $Rigid(\mathbb{P})$, respectively $\mathbb{P}xy \rightarrow \Box\mathbb{P}xy$. While the full proofs can be found in A.12.4301, it is noteworthy that it again requires that one argue by appealing to *rigidifying* relations: by the theorem governing the predecessor relation given above, $\mathbb{P}xy$ implies that there exists a property F and an ordinary object u , s.t. $[F]u \ \& \ Numbers(y,F) \ \& \ Numbers(x,F^{-u})$. However, none of the conjuncts are guaranteed to be necessary. But we may refer to the fact that for any property F there exists a property G that *rigidifies* F and this property G can serve as witness for the claim that $\Box\mathbb{P}xy$.

²⁶Both $\neg\exists x \mathbb{P}x0$ and $\mathbb{P}01$ are formally derived in A.12.4498, resp. A.12.5437.

Furthermore, it is a consequence of a modally-strict variant of Hume's principle that the predecessor relation is one-to-one (see A.12.4426): $1-1(\mathbf{P})$.

Consequently, the Predecessor Relation is a rigid one-to-one relation and we can instantiate the definition of the *strong* ancestral to \mathbf{P} (see A.12.4468):

$$\mathbf{P}^* = [\lambda xy \forall F (\forall z (\mathbf{P}xz \rightarrow [F]z) \ \& \ Hereditary(F, \mathbf{P}) \rightarrow [F]y)]$$

Furthermore, being \mathbf{P} -identical as well as the *weak* ancestral of \mathbf{P} are also well-defined (see A.12.4540):

$$\begin{aligned} x =_{\mathbf{P}} y &\equiv \exists z (\mathbf{P}xz \ \& \ \mathbf{P}yz) \\ \mathbf{P}^+ &= [\lambda xy [\mathbf{P}^*]xy \vee x =_{\mathbf{P}} y] \end{aligned}$$

Before we continue to define natural numbers, note that it is already derivable that the number Zero neither has a direct nor a transitive predecessor (see A.12.4498): $\neg \exists x \mathbf{P}x0$ respectively $\neg \exists x [\mathbf{P}^*]x0$

5.10. Natural Numbers

Using the infrastructure introduced in the past sections, we can now follow through with the strategy described in the beginning of the chapter and define *being a natural number* as being an object, s.t. Zero bears the weak ancestral of the predecessor relation to it (see A.12.4577):

$$\mathbf{N} =_{df} [\lambda x [\mathbf{P}^+]0x]$$

Since being a natural number trivially denotes, it follows by β -conversion that (see A.12.4582):

$$\mathbf{N}x \equiv [\mathbf{P}^+]0x$$

5.11. Zero is a Natural Number

The first Dedekind-Peano postulate can now be derived (see A.12.4588):

$$\mathbf{N}0$$

Interestingly, both in Frege's original work and in Zalta's initial reconstruction (see [60]) the weak ancestral was defined using general identity and consequently $[\mathbf{P}^+]00$ is a simple consequence of the fact that Zero is self-identical. However, due to the construction via rigid one-to-one relations this theorem requires a non-trivial proof: $[\mathbf{P}^+]00$ by definition is just the case if either $[\mathbf{P}^*]00$ (which was already refuted above) or $0 =_{\mathbf{P}} 0$.

However, $0 =_{\mathbf{P}} 0$ is not a simple consequence of the fact that $0 = 0$, but additionally requires that $InDomainOf(0, \mathbf{P})$, respectively that $\exists y \mathbf{P}0y$, i.e. the proof effectively requires to construct the number One as witness.²⁷

²⁷The number One can for example be introduced as the number of any relation exemplified by exactly one ordinary object. Since it is a theorem (see A.7.7510) that there is an ordinary object $\exists x O!x$, we can choose a to be a witness to this existential claim and choose $\#[\lambda x O!x \ \& \ x =_E a]$ as a witness to $\exists y \mathbf{P}0y$.

Preliminary working versions of the chapter of PLM left this non-trivial proof as an exercise referring to it being a trivial consequence of the self-identity of the number Zero. Trying to prove the statement in the embedding showed that additional work is required due to the changes in the construction compared to previous versions and we were able to notify Nodelman and Zalta both that the proof is non-trivial and suggest to them the proof given in A.12.4588 and outlined in the footnote of the last paragraph.²⁸

5.12. Being a Natural Number is Rigid

From the generalized principle of induction when instantiating F to $[\lambda x \Box \mathbb{N}x]$ and \mathcal{R} to \mathbb{P} , it follows that $\mathbb{N}x \rightarrow \Box \mathbb{N}x$ and consequently that *Rigid*(\mathbb{N}) (see A.12.4631, A.12.4689). Since furthermore Zero is a witness to the existence of natural numbers and it is easy to prove that $\mathbb{N}\kappa \rightarrow \kappa \downarrow$,²⁹ *being a natural number is a rigid restriction condition* and it is possible to introduce well-behaved restricted variables ranging over the natural numbers (recall section 3.4.4).

In the following the variable names m , n , k , i and j are used to range over natural numbers.

5.13. Zero Has No Predecessor

We have already mentioned the fact that $\neg \exists x \mathbb{P}x0$ above, but we can now restate this theorem *a fortiori* for variables restricted to natural numbers, which constitutes the second Dedekind-Peano postulate (as mentioned earlier this formulation is equivalent to the assertion that Zero is not the successor of any natural number; see A.12.4714):

$$\neg \exists n \mathbb{P}n0$$

5.14. No Two Natural Numbers have the Same Successor

The third Dedekind-Peano postulate is a general property of any one-to-one relation, but can be stated explicitly using restricted variables for natural numbers (on which \mathbb{P} -identity matches general identity) as follows (see A.12.4725):

$$\forall n \forall m \forall k (\mathbb{P}nk \ \& \ \mathbb{P}mk \rightarrow n = m)$$

Whenever two natural numbers n and m precede the same natural number k (or, equivalently, if n and m have the same successor), they have to be identical.

²⁸Note that the chapter was under heavy revision at the time and this omission would likely have been independently discovered eventually. However, it is one of the merits of working in a computer-verified setting that such omissions become immediately apparent.

²⁹It is a simple consequence of one of the quantifier axioms mentioned in section 3.3.

5.15. Mathematical Induction

Furthermore, we can now derive Mathematical Induction (see A.12.4738):

$$\forall F ([F]0 \ \& \ \forall n \ \forall m (\mathbf{P}nm \rightarrow ([F]n \rightarrow [F]m)) \rightarrow \forall n [F]n)$$

If a property (1) is exemplified by the number Zero and (2) it being exemplified by a natural number implies it being exemplified by its successor, then all natural numbers exemplify that property.³⁰ This is a simple consequence of instantiating generalized induction (recall section 5.8) to the predecessor relation.

Thereby the fifth Dedekind-Peano postulate is derivable. Note, however, that we haven't yet derived the fourth postulate, i.e. that every natural number has a unique successor. The construction so far is validated by the minimal models of AOT that are extended to validate the predecessor axiom (i.e. in which the predecessor relation denotes). Validating the predecessor axiom involves increasing the number of special urelements in the model (see 5.19), but it does not require to increase the number of ordinary urelements/objects, so there are still models with only a single ordinary urelement/object, in which the predecessor relation denotes. However, in such models the only natural numbers are Zero and One and the number One does not have a successor. For that reason, Nodelman and Zalta extend the system by another axiom, which we will discuss below after stating a few more derived properties of the predecessor relation and natural numbers.

5.16. Properties of the Predecessor Relation and Natural Numbers

Successors of natural numbers are (transitively) natural numbers (see A.12.4766):

$$\begin{aligned} \mathbf{P}nx &\rightarrow \mathbf{N}x \\ [\mathbf{P}^*]nx &\rightarrow \mathbf{N}x \\ [\mathbf{P}^+]nx &\rightarrow \mathbf{N}x \end{aligned}$$

Predecessors of natural numbers are (transitively) natural numbers (see A.12.4824):

$$\begin{aligned} \mathbf{P}xn &\rightarrow \mathbf{N}x \\ [\mathbf{P}^*]xn &\rightarrow \mathbf{N}x \\ [\mathbf{P}^+]xn &\rightarrow \mathbf{N}x \end{aligned}$$

Natural numbers are natural cardinals (see A.12.4865):

$$\mathbf{N}x \rightarrow \text{NaturalCardinal}(x)$$

The predecessor relation is functional (see A.12.4899):

³⁰Note that, strictly speaking, our natural language formulation rather corresponds to the derived theorem $\forall F ([F]0 \ \& \ \forall n ([F]n \rightarrow [F]n') \rightarrow \forall n [F]n)$ (see A.13.273), where n' is defined as *the* successor of n , resp. *the* natural number that is preceded by n (see A.12.5336). However, this formulation can only be derived after proving that every number *has* a (unique) successor.

$$\begin{aligned} & \mathbb{P}xy \ \& \ \mathbb{P}xz \rightarrow y = z \\ & \mathbb{P}nm \ \& \ \mathbb{P}nk \rightarrow m = k \end{aligned}$$

5.17. Possible Richness of Objects

As mentioned above, the construction so far is valid in models with only a single ordinary object and consequently with only two natural numbers, which is not sufficient to derive that every natural number has a successor.

The following modal axiom, by which Nodelman and Zalta proceed to extend the system, changes this (see A.12.4956):

$$\exists x (\mathbb{N}x \ \& \ x = \#G) \rightarrow \diamond \exists y (E!y \ \& \ \forall u (\mathcal{A}[G]u \rightarrow u \neq_E y))$$

If there is a natural number which numbers G , then there might have been a concrete object y which is distinct from every ordinary object that *actually* exemplifies G . We will explain in detail how we extend our models to be able to validate this axiom in section 5.20. In summary, the axiom requires extending the domain of ordinary urelements/objects to an at least countably infinite set.

This axiom requires some justification, especially given the claim that the construction is *purely logical* and does not require to presuppose any intrinsically mathematical claims. Traditionally, a system is no longer considered to be *purely logical*, if it asserts the existence of more than one object.³¹ While Nodelman and Zalta agree with this principle, they argue (see PLM item (799)) that it only extends to *concrete* objects. While above axiom does imply that the domain of *ordinary* objects (recall that *being ordinary* is defined as *being possibly concrete*) is at least countably infinite, it does not imply that there is even a single object that is *actually concrete*. Nodelman and Zalta further argue that on the one hand it is in fact common for logical systems to assert the existence of more than one *abstract* object, for example that there are two distinct truth values, the True and the False,³² and that on the other hand logicians traditionally work under the assumption that *the domain of objects might be of any size*, which they take as a modal claim: while logic may not presuppose that the domain of concrete objects has any particular size, it allows for the *possibility* of the domain being of any size, i.e. it is valid for a logic to presuppose that there *may possibly* be more than one object, as long as that does not imply that there is *actually* more than one (concrete) object.

Independently of the question whether this axiom may or may not be considered as *purely logical*, towards which we refrain from presuming to pass judgement either way, we certainly agree that it captures a pretheoretic intuition: it can be considered as a prerequisite of talking about natural numbers to be able to imagine that no matter how many objects we currently consider that there *possibly might have been* yet another

³¹E.g. PLM cites Boolos [14]: “In logic, we ban the empty domain as a concession to technical convenience but draw the line there: We firmly believe that the existence of even two objects, let alone infinitely many, cannot be guaranteed by logic alone.”

³²In particular, they refer to Frege’s logic.

object, even though for doing so we do not need to be able to point to this object in the actual world (i.e. it may not be concrete, but merely *possibly concrete*).

While this may serve as justification for the axiom, Frege's original construction does not rely on a similar assumption, but can use the number of the property *being less than or equal to n*, $\#[\lambda x x \leq n]$, as witness for a successor of any natural number n . In the presented construction that relies on equinumerosity among the ordinary objects, this is not an option: since natural numbers are abstract, being a natural number smaller or equal to n is only exemplified by abstract objects and therefore unexemplified by ordinary objects. Thus $\#[\lambda x x \leq n]$ is Zero and, in particular, cannot serve as the successor of any number.

However, we will discuss two variants of the construction in section 5.21 in which *discernible* abstract objects *can* be counted (and in which natural numbers, in particular, will be discernible). This allows for the construction of a successor of n as $\#[\lambda x x \leq n]$, thereby eliminating the need for this axiom.

5.18. Every Number has a Unique Successor

The axiom above is sufficient to derive the last Dedekind-Peano postulate, i.e. that every natural number has a unique successor (see A.12.5249):

$$\forall n \exists ! m (\mathbb{P}nm)$$

Every natural number n is a natural cardinal and, by definition (see A.12.2570), natural cardinals are the number of some property and thus $NaturalCardinal(n) \rightarrow \exists G n = \#G$.

Let G be a property such that $n = \#G$.

Now the axiom implies that there is an ordinary object v , s.t. G does not actually exemplify v . This requires an appeal to the Barcan formulas (in particular A.7.3470) and relies on the additional fact (see A.12.5215) that:

$$\diamond \forall u (\mathcal{A}[G]u \rightarrow u \neq_E v) \rightarrow \forall u (\mathcal{A}[G]u \rightarrow u \neq_E v)$$

Hence, since $n = \#G$ implies that n numbers $[\lambda x \mathcal{A}[G]x]$ (see A.12.2742), the object that numbers $[\lambda x \mathcal{A}[G]x \vee x =_E v]$ can be used as witness for a successor of n .

Uniqueness follows from the fact that the predecessor relation is functional.

Hence, it is possible to define *the* successor n' of a natural number n as *the* natural number that is preceded by n :

$$n' =_{df} \iota x (\mathbb{N}x \ \& \ \mathbb{P}nx)$$

Numerals can be defined as iterated successors, e.g. $1 =_{df} 0'$.

While PLM continues to derive further theorems of Number Theory, defines mathematical functions and operations, including recursively defined functions such as addition, and proceeds to derive Second-Order Dedekind-Peano arithmetic, we will conclude our discussion of the topic here and instead discuss in more detail how we modelled the two required additional axioms.

5.19. The Predecessor Axiom in Detail

Recall that the predecessor axiom of PLM is stated as follows (see A.12.4284):

$$[\lambda xy \exists F \exists u ([F]u \ \& \ Numbers(y,F) \ \& \ Numbers(x,F^{-u}))]\downarrow$$

In section 5.9.2 we have already established that the relation in question distinguishes certain abstract objects that number properties and that this relation does *not* denote in the minimal models of the base system of AOT. We also have already discussed that there cannot be a relation in AOT that generally distinguishes between arbitrary abstract objects (in particular $[\lambda xy \ x = y]$ does not denote; see 3.8.1). So we need to determine what is special about the abstract objects that are distinguished by the predecessor relation and allows us to construct models for it.

To that end, we first show that the predecessor relation coexists with *numbering a property*. In particular we can prove the following (see A.12.3905):

$$[\lambda xy \exists F \exists u ([F]u \ \& \ Numbers(y,F) \ \& \ Numbers(x,F^{-u}))]\downarrow \equiv \forall F [\lambda x \ Numbers(x,F)]\downarrow$$

So to validate the predecessor axiom, we can equivalently construct models in which $[\lambda x \ Numbers(x,F)]\downarrow$ is a theorem. Recall that *numbering a property* is equivalent to the following (see A.12.2180):

$$Numbers(x,G) \equiv A!x \ \& \ \forall F (x[F] \equiv [\lambda z \ \mathcal{A}[F]z] \approx_E G)$$

So while *numbering a property* is a condition on the properties an abstract object encodes, it requires the abstract object to encode an entire class of properties, namely all properties, s.t. *actually* exemplifying them is equinumerous_E to the numbered property. Further recall that being *equinumerous_E*, informally speaking, means to be exemplified by the same amount of *ordinary* objects.

This is the crucial fact that allows us to construct suitable models: while we need to distinguish between abstract objects based on the properties they *encode*, the condition under which these abstract objects encode or do not encode properties solely depends on the exemplification patterns of those properties on the *ordinary* objects.

In our models, two abstract objects are exemplification-distinguishable, if they are mapped to distinct *special urelements*. If we wanted to be able to distinguish between abstract objects in general based on the exemplification patterns of the properties they encode, this would mean that there had to be a distinct *special urelements* for any such pattern. Exemplification of a property is a functions from *urelements* (including special urelements) to modal truth conditions (i.e. functions from semantic possible worlds to booleans).

Therefore, if we wanted to assign distinct special urelements based on *general* exemplification patterns, we would need an injective function from exemplification patterns (i.e. sets of functions acting on urelements) to special urelements, which would be in violation of Cantor's theorem.

However, fortunately, we only need to distinguish between exemplification patterns on *ordinary* objects. Since the domains of special urelements and ordinary urelements are

independent, it is consistently possible to construct special urelements in such a way that there can be an injective function mapping distinct sets of functions *acting on ordinary urelements alone* to distinct special urelements.

In our general models we choose an *abstract* type σ as type of special urelements.³³ In our extended models that validate the predecessor axiom, we instead *define* the type σ using the set of objects of type $(\omega \Rightarrow w \Rightarrow \text{bool}) \text{ set} \times (\omega \Rightarrow w \Rightarrow \text{bool}) \text{ set} \times \sigma'$ as representation set.³⁴

Recall that the type ω is the type of ordinary urelements and w is the type of semantic possible worlds. σ' is an additional abstract type of *very special urelements* that will retain the model's ability to distinguish between abstract objects beyond those that differ in exemplification patterns on the ordinary objects. So in these models, special urelements are tuples of two sets of property extensions on ordinary objects and a very special urelement. We refer to the first set of extensions as the *intersection set of ordinary property extensions* and to the second copy as the *union set of ordinary property extensions*.

When we map an abstract object a to this new type of special urelements, we insert a property extension on the ordinary objects into the intersection set, just in case a encodes *all* properties with this extension on the ordinary objects. And we insert an extension into the union set, just in case that there *exists* a property with that extension (on the ordinary objects) that is encoded by a .

We use this construction as witness for a specification of the mapping $\alpha\sigma'$, which will then be extended to a surjective mapping $\alpha\sigma$ as explained in section 4.1.4.

This construction *forces* two abstract objects to be assigned different special urelements, in case either (1) one of them encodes a property with a given exemplification extension on the ordinary object, while the other doesn't encode any such property, or (2) one of them encodes all properties with a given extension on the ordinary object, while the other fails to encode at least one such property.

Furthermore, the construction still *allows* two abstract objects to be assigned different special urelements, in case they differ only in encoding properties with the same extension on the ordinary objects (by assigning them distinct *very special* urelements).

This extended model validates the following two axioms (see A.6.245, A.6.252):

- $\Pi \downarrow \& A!x \& A!y \& \forall F \Box([F]x \equiv [F]y) \rightarrow (\forall G (\forall u \Box([G]u \equiv [\Pi]u) \rightarrow x[G]) \equiv \forall G (\forall u \Box([G]u \equiv [\Pi]u) \rightarrow y[G]))$
- $\Pi \downarrow \& A!x \& A!y \& \forall F \Box([F]x \equiv [F]y) \rightarrow (\exists G (\forall u \Box([G]u \equiv [\Pi]u) \& x[G]) \equiv \exists G (\forall u \Box([G]u \equiv [\Pi]u) \& y[G]))$

I.e. if two abstract objects are (exemplification-)indistinguishable, then (1) if either one encodes all properties that are necessarily equivalent on the ordinary objects to any given denoting property term Π , then the other also encodes all these properties, and

³³I.e. we allow any non-empty domain for σ in models of the meta-logic without restriction.

³⁴A smaller subset of the set of such triples (a, b, s) , e.g. for which it always holds that $a \subseteq b$ and for which $a = b$ implies $s = s_0$ for some fixed s_0 , would suffice.

(2) if either one encodes any property that is necessarily equivalent to Π on the ordinary objects, there is also such a property that is encoded by the other.

While this formulation of the axioms is rather complex and not particularly intuitive, we can equivalently (given the necessary and sufficient conditions for relation terms to denote described in section 3.8.2) state them as follows (see A.10.445, A.10.458):

$$\begin{aligned} & [\lambda x \exists G (\Box G \equiv_E F \ \& \ x[G])] \downarrow \\ & [\lambda x \exists G (\Box G \equiv_E F \ \& \ \neg x[G])] \downarrow \end{aligned}$$

I.e. (1) *encoding a property that is necessarily equivalent on the ordinary objects to a given property F* denotes a property and (2) *not encoding a property that is necessarily equivalent on the ordinary objects to a given property F* denotes a property.³⁵

The following comprehension principles are derivable from the fact that above properties denote (see A.10.473, A.10.531):

$$\begin{aligned} & \Box \forall F \forall G (\Box G \equiv_E F \rightarrow (\varphi\{F\} \equiv \varphi\{G\})) \rightarrow [\lambda x \exists F (\varphi\{F\} \ \& \ x[F])] \downarrow \\ & \Box \forall F \forall G (\Box G \equiv_E F \rightarrow (\varphi\{F\} \equiv \varphi\{G\})) \rightarrow [\lambda x \exists F (\varphi\{F\} \ \& \ \neg x[F])] \downarrow \end{aligned}$$

We call φ a *condition on extensions on ordinary objects*, just in case it satisfies the antecedent, i.e. just in case that $\Box \forall F \forall G (\Box G \equiv_E F \rightarrow (\varphi\{F\} \equiv \varphi\{G\}))$. Then the comprehension principles state that for any condition φ on extensions on ordinary objects, both *encoding a property that satisfies φ* and *not encoding a property that satisfies φ* denote properties.³⁶

In combination these two principles yield the following (see A.10.633):³⁷

$$\Box \forall F \forall G (\Box G \equiv_E F \rightarrow (\varphi\{F\} \equiv \varphi\{G\})) \rightarrow [\lambda x \forall F (x[F] \equiv \varphi\{F\})] \downarrow$$

I.e. for every condition φ on extensions on ordinary objects, *encoding exactly those properties that satisfy φ* denotes a property.

It is easy to show that *being an F , s.t. actually exemplifying F is equinumerous to G* , is a condition on extensions on ordinary objects. Hence it is a consequence of this last comprehension principle that $[\lambda x \forall F (x[F] \equiv [\lambda z \mathcal{A}[F]z] \approx_E G)] \downarrow$ and thereby *numbering a property* denotes by coexistence (see A.12.4236).

Justification of the Comprehension Principles

While the predecessor axiom singles out a particular relation among abstract objects for the sole purpose of defining a mathematical relation, the comprehension principles we suggest provide a general means to construct relations among abstract objects based on specific encoding patterns in a manner that is provably consistent, but also independently justifiable.

³⁵Note that these properties coexist with their negations, i.e. $[\lambda x \forall G (\Box G \equiv_E F \rightarrow \neg x[G])] \downarrow$ is equivalent to the first, $[\lambda x \forall G (\Box G \equiv_E F \rightarrow x[G])] \downarrow$ is equivalent to the second.

³⁶See also A.10.591 and A.10.613 for derived variants of these principles.

³⁷However, note that above principles are stronger, i.e. they are *not* derivable from the combined principle.

In general, the burden of justification rather lies in the fact that some abstract objects *are* exemplification-indistinguishable: let R_t be the relation *thinking about*, s.t. $[R_t]xy$ can be read as x is thinking about y . Then for two distinct abstract objects a and b to be exemplification-indistinguishable implies that it is impossible for anyone to think about one without thinking about the other: $\forall x \square([R_t]xa \equiv [R_t]xb)$, resp. $\neg\Diamond\exists x ([R_t]xa \ \& \ \neg[R_t]xb \vee [R_t]xb \ \& \ \neg[R_t]xa)$.

While the existence of such objects is justifiable, it is not necessarily a pretheoretic intuition. Interestingly, it is not possible to *independently* construct two abstract objects that are in fact exemplification-indistinguishable: while it is provable that there *exist* such pairs of objects, the construction always has to rely on constructing one of the objects particularly in such a way that it cannot be distinguished from the other.³⁸ Whenever two abstract objects are constructed independently, a model can generally choose two distinct special urelements for them, thereby making them distinguishable. Only if the construction of the second abstract object *depends* on the choice of a special urelement for the first and forces both objects to be collapsed under the mapping from abstract objects to special urelements, this becomes infeasible.

This helps in reconciling the fact that there are indistinguishable abstract objects with the following pretheoretic intuition: given two independent abstract objects, we can always find ourselves thinking about one, but not the other. However, we can conceive of concepts that e.g. themselves involve *being indistinguishable from other abstract objects*, for which a clever construction in fact yields distinct concepts that are indistinguishable.³⁹

So while we can always consistently distinguish between *particular, independent* abstract objects, given that there still *are* indistinguishable abstract objects, we cannot formulate a completely general principle that allows for distinguishing *arbitrary* abstract objects.

However, our suggested comprehension principles are restricted to abstract objects that have encoding conditions that differ in exemplification patterns on the *ordinary* objects. If for two abstract objects we can point to a pattern among the ordinary objects, s.t. one of the object involves this pattern (i.e. it encodes a property that satisfies this pattern), but the other one doesn't involve this pattern at all (i.e. it encodes no property that satisfies this pattern), we have a concrete criterion for telling the objects apart.⁴⁰ The same can be said, if one of the object fails to fully encode such a pattern (i.e. there is a property with this pattern on the ordinary objects that it doesn't encode), while the other encodes all properties with this pattern.

The third, combined principle (which is weaker than the first two principles, but strong enough for *numbering a property* to denote) is seemingly even easier to justify: if an abstract object encodes *exactly* those properties that satisfy a given pattern on the

³⁸And even this is only possible for specific choices of a first abstract object: For example, we cannot construct an abstract object that is indistinguishable from the null-object (that encodes no properties) since we can always conceive of a model that maps the null-object to a designated special urelement that no other abstract object maps to.

³⁹Recall the discussion in section 3.8.1.

⁴⁰Respectively, equivalently by 3.8.2, for allowing a property that tells them apart to denote.

ordinary objects, then it is fully determined by this pattern, so in this sense we can *identify* such abstract objects with the respective patterns on the ordinary objects they encode.

Assuming that there are distinct patterns among the ordinary objects that are indistinguishable seems hardly justifiable. However, this relies on a particular understanding of what it means to encode patterns among the ordinary objects that may not be completely intuitive, as conceded in the next section.

However, our construction already shows that it is not necessary to justify the predecessor relation directly as a denoting relation: We can generalize the issue to the question of when abstract objects can be assured to be exemplification-distinguishable. In this more general question we no longer see any ties to Mathematics whatsoever, but rather a metaphysical discussion of the nature of abstract objects and relations among them.

Caveats of the Comprehension Principles

While the comprehension principles suggested above have some justification and allow for deriving that useful encoding conditions such as *numbering a property* can be abstracted to properties, they are not the only conceivable way of generically extending AOT with relations among abstract objects.

In particular, it does *not* follow from the suggested principles that any of the following properties denote:

- $[\lambda x \forall F (x[F] \rightarrow \Box \forall z ([F]z \rightarrow O!z))]$, i.e. *encoding only properties that are necessarily restricted to ordinary objects*.
- $[\lambda x x[\lambda z O!z \ \& \ \varphi\{z\}]]$, i.e. *encoding a particular pattern among the ordinary objects*.
- $[\lambda x \text{ExtensionOf}(x, [\lambda z O!z \ \& \ [G]z])]$ where *ExtensionOf*(x, G) is defined by PLM as $\text{ExtensionOf}(x, G) \equiv_{df} A!x \ \& \ G\downarrow \ \& \ \forall F (x[F] \equiv \forall z ([F]z \equiv [G]z))$ (see A.13.298).

The notion of an *extension on the ordinary objects* we used above would have to be defined as (see A.13.300):

$$\text{OrdinaryExtensionOf}(x, G) \equiv_{df} A!x \ \& \ G\downarrow \ \& \ \forall F (x[F] \equiv \forall z (O!z \rightarrow ([F]z \equiv [G]z)))$$

With this definition, $[\lambda x \text{OrdinaryExtensionOf}(x, G)]\downarrow$ is derivable from the suggested principles (see A.13.303). However, using this conception of extensions on ordinary objects as the basis for our comprehension principles, has some potentially counter-intuitive implications:

If one abstract objects encodes exactly the property *being an ordinary table*, and another abstract object encodes exactly *being an ordinary table or being abstract*, our comprehension principles are not sufficient for telling them apart. Both objects involve the same pattern among the ordinary objects, but neither encodes it fully, since, for instance, neither encodes *being an ordinary table or being a natural number*, which also has the same exemplification pattern among the ordinary objects.

The third, combined principle alone cannot even distinguish between an object that encodes exactly *being an ordinary table* and an object that encodes exactly *being a mathematician* - neither of these objects are *fully determined by a pattern on the ordinary objects* in the sense of our principles, since neither encodes *all* properties with this pattern.⁴¹

Relation to Leibnizian Concepts and Platonic Forms

Despite the concessions above, our comprehension principles align well with the analysis of other philosophical objects in AOT. PLM defines for an abstract object to be the Leibnizian Concept of a property as follows (see A.13.412):⁴²

$$\text{ConceptOf}(x, G) \equiv_{df} C!x \ \& \ (G\downarrow \ \& \ \forall F \ (x[F] \equiv G \Rightarrow F))$$

An object x is a concept of G , just in case it encodes exactly those properties that are necessarily implied by G , using the following definition of necessary implications between properties (see A.13.353):

$$F \Rightarrow G \equiv_{df} G\downarrow \ \& \ F\downarrow \ \& \ \Box\forall x \ ([F]x \rightarrow [G]x)$$

Now our comprehension principles make it derivable that *being a concept of H* is a property, if H necessarily implies *being ordinary* (see A.13.414):

$$H \Rightarrow O! \rightarrow [\lambda x \ \text{ConceptOf}(x, H)]\downarrow$$

So concepts of properties that do not involve abstract objects can always be distinguished from other abstract objects.

Reusing the example above, the concept of *being an ordinary table* *does* encode *being an ordinary table or being abstract*, since the former necessarily implies the latter. In fact it encodes all properties that are necessarily equivalent on the ordinary objects to *being an ordinary table*, since all those properties are necessarily implied by *being an ordinary table*.

Consequently, concepts of properties that necessarily imply being ordinary and possibly differ on some ordinary object become provably distinguishable. In particular, it becomes a *theorem* that the concept of *being an ordinary table* is discernible from the concept of *being a mathematician* (assuming that these properties are not necessarily exemplified by the same objects).

Further examples of theorems that can be derived from our comprehension principles are (see A.13.703 and A.13.720):

$$H \Rightarrow O! \rightarrow [\lambda x \ \mathbf{c}_H \preceq x]\downarrow$$

$$H \Rightarrow O! \rightarrow [\lambda x \ x \preceq \mathbf{c}_H]\downarrow$$

⁴¹However, they become distinguishable on the bases of the first principle above, since we can find a pattern among ordinary objects one of the abstract objects encodes, while the other one doesn't (assuming mathematicians aren't tables).

⁴²*Being a concept $C!$* is defined as $C! =_{df} A!$.

I.e. both *including* and *being included by* the concept of a property H denote, given that H necessarily implies being ordinary.⁴³

Thick platonic forms are defined similarly to Leibnizian concepts of properties (see A.13.738):

$$\text{FormOf}(x, G) \equiv_{df} A!x \ \& \ G\downarrow \ \& \ \forall F (x[F] \equiv G \Rightarrow F)$$

So we can also derive that *being the (thick) platonic form of H* denotes a property, if H necessarily implies *being ordinary* (see A.13.740):

$$H \Rightarrow O! \rightarrow [\lambda x \text{FormOf}(x, H)]\downarrow$$

This shows that our comprehension principles are by no means *ad hoc* and have relevant implications for philosophical objects beyond the natural numbers. A detailed study of the implications of these principles will be an interesting topic for future research.

However, given the prospect of a move from abstracting patterns among *ordinary* objects to abstracting patterns among *discernible* objects instead, an even more interesting question may be whether similar general comprehension principles can be formulated for distinguishing objects that encode different patterns among *discernible* objects. We will discuss this further in section 5.21.

5.20. Modelling Possible Richness of Objects

Recall that the axiom of possible richness of objects was stated as follows (see A.12.4956):

$$\exists x (\mathbb{N}x \ \& \ x = \#G) \rightarrow \diamond \exists y (E!y \ \& \ \forall u (\mathcal{A}[G]u \rightarrow u \neq_E y))$$

Compared with the predecessor axiom, modelling possible richness of objects is straightforward. The axiom implies that there are countably infinitely many ordinary (even though potentially not *actually*, but merely *possibly* concrete) objects, so in our models we simply require there being a surjection from our type ω of ordinary urelements to Isabelle's type of natural numbers *nat*. While deriving the axiom from this change in the model is still non-trivial, we can prove (notably, our proof relies on the extended relation comprehension principles we introduced for modelling the predecessor relation as well as AOT's defined mathematical induction), that *being a natural number* in the models implies encoding only properties that are actually exemplified by only finitely many ordinary objects. Thereby, whenever a natural number numbers a property, it is only actually exemplified by a finite number of ordinary objects and since we have required infinitely many ordinary objects in our model, we can produce a witness to the claim of the axiom (modulo some further modal reasoning).

Furthermore, there is no way to model the axiom *without* extending the domain of ordinary objects in the model to infinitely many objects.

So for this axiom, the more interesting issue compared to modelling it is whether it can be philosophically justified as a purely logical axiom or not (see 5.17). While we

⁴³The definitions of \mathbf{c}_G (see A.13.551) and \preceq (see A.13.408) can be found in appendix A.13, which implements fragments of the theory of concepts given in PLM chapter 13.

do not presume to judge whether the justification provided by Nodelman and Zalta in PLM item (799), resp. in [60], is sufficient to establish this axiom as purely logical, we certainly agree that it captures a natural and intuitive conception of *counting*.

Interestingly, however, it may be possible to eliminate the axiom altogether and more closely reproduce Frege's proof that every natural number has a successor as discussed in the next section.

5.21. Prospect of an Enhanced Version of the Construction

At the time of writing, there is an ongoing debate concerning variations of the analysis of natural numbers. In particular, instead of restricting the analysis to ordinary objects, identity on the ordinary objects and equinumerosity on the ordinary object, Nodelman and Zalta brought up the idea to instead follow the same basic construction relative to *discernible objects*.⁴⁴

Being discernible, $D!$, can be defined as the following relation:

$$D! =_{df} [\lambda x \square \forall y (y \neq x \rightarrow \exists F \neg([F]y \equiv [F]x))]$$

Using the necessary and sufficient conditions for relations to denote discussed in section 3.8.2, it can be shown that $D!$ denotes.⁴⁵ Furthermore, just as *being ordinary*, *being discernible* is a rigid restriction condition. Similar to $=_E$ on the ordinary objects, a relation of identity on the discernible objects $=_D$ can be defined as $[\lambda xy \square \forall F ([F]x \equiv [F]y)]$, i.e. for discernible objects *being indistinguishable* implies identity. The construction up until the modal axiom of section 5.17 can be preserved without any major changes. Being equinumerous $_D$ can be defined just as equinumerous $_E$ (see section 5.2.3), but relative to a one-to-one correspondence $_D$ on discernible objects, which in turn can be defined just as a one-to-one correspondence $_E$ (see section 5.2.2), but using restricted variables ranging over discernible instead of ordinary objects.

The fact that *numbering a property* coexists with the predecessor relation described in section 5.19 is invariant under this change. Moreover, natural numbers will themselves become discernible (since by Hume's theorem for two objects numbering the same properties implies their identity). This allows for abandoning the modal axiom for possible richness of ordinary objects and instead to more closely follow Frege's construction, in which the successor of a number n is defined as the number of the property *being smaller-or-equal to n* , i.e. $n' = \#[\lambda m m \leq n]$, yielding $\mathbb{P}nn'$.

At the time of writing, we have prototypes for models of this new derivation available. In these models we restrict the domain of ordinary urelements to be at most countably infinite (i.e. either finite or in bijection to the natural numbers), and require the domain

⁴⁴Personal correspondence of 14 October 2021 and 2 November 201. They are now revising the chapter on number theory on the basis of this idea.

⁴⁵Note that due to the matrix involving a non-identity claim and identity on individuals being defined in terms of encoding, the λ -expression does not denote axiomatically.

of special urelements to be countably infinite.⁴⁶ From this restriction it can be derived that the class of cardinal numbers that measure the size of sets of discernible objects is itself a countable set.⁴⁷ Since abstract objects that number properties will be in one-to-one correspondence with the cardinals of sets of discernible urelements,⁴⁸ they can thus injectively be mapped into the special urelements, making them discernible. Hence this validates the theorem that *numbering a property* denotes and consequently yields models for the predecessor axiom.

As mentioned in section 5.19, it is an interesting question whether similar general comprehension principles can be formulated for distinguishing objects that encode different patterns among *discernible* objects, as we could suggest for patterns among *ordinary* objects. However, since discerning abstract object based on patterns among discernible objects yields new discernible objects, there is an increased danger of general comprehension principles for encoding patterns among discernible objects to become self-referential and thereby inconsistent. So while we expect to be able to formulate meta-theorems about the conditions under which it will be safe to assert the existence of relations among abstract objects that encode patterns among discernible objects,⁴⁹ it is unclear if we will be able to arrive at general comprehension principles that can be formulated in the theory itself.

In general, the price of being able to eliminate the modal axiom described in section 5.17 using the new construction will be that the predecessor axiom will become stronger and may have to rely on independent means of justification.

Another similar variant of the construction, for which we have already constructed full models (see [28]), does not restrict the domain of objects that can be counted at all, but instead of counting distinct objects rather counts equivalence classes of objects that are indistinguishable.⁵⁰ This involves weakening the unique existence used in one-to-one correspondences to uniqueness up to distinguishability, i.e. we define unique existence_D as follows:

$$\exists!_D \alpha \varphi\{\alpha\} \equiv_{df} \exists \alpha (\varphi\{\alpha\} \ \& \ \forall \beta (\varphi\{\beta\} \rightarrow \beta =_D \alpha))$$

One-to-one correspondences and equinumerosity are then constructed relative to this restricted notion of unique existence:

$$\begin{aligned} R \mid: F \text{ }_{1-1} \longleftrightarrow_D G &\equiv \\ \forall x ([F]x \rightarrow \exists!_D y ([G]y \ \& \ [R]xy)) \ \& \ \forall y ([G]y \rightarrow \exists!_D x ([F]x \ \& \ [R]xy)) & \\ F \approx_D G &\equiv_{df} \exists R \ R \mid: F \text{ }_{1-1} \longleftrightarrow_D G \end{aligned}$$

⁴⁶In a more general construction, it would be sufficient to require there being countably infinitely many special urelements that serve as proxies for discernible objects, while allowing an arbitrary number of special urelements for indiscernible objects.

⁴⁷For every n , there is one cardinal number for finite sets of n discernibles, and additionally there is one cardinal for countably infinite sets of discernibles.

⁴⁸In another variant mentioned below they will be in one-to-one correspondence with the cardinals of sets of arbitrary urelements.

⁴⁹For example, any axiom that implies that certain abstract objects become discernible can be consistently modelled, as long as it discerns at most countably many abstract objects.

⁵⁰I.e. indistinguishable objects belong to the same equivalence class and objects belonging to different equivalence classes are distinguishable.

While a construction based on discernible objects ignores objects that are indiscernible for the purpose of counting, i.e. a property that is exemplified by two indistinguishable abstract objects and no other objects is counted by Zero, this construction would count such objects in bulk, i.e. the same property would be counted by One. For properties that are only exemplified by discernible objects, both constructions are equivalent (i.e. such properties are equinumerous in the first variant if and only if they are equinumerous in the second).

5.22. Summary

In summary, we can conclude that the construction of natural numbers and the derivation of the Dedekind-Peano postulates given in PLM is provably sound. While the construction relies on additional axioms, we can say that:

- PLM can present reasonable justifications for both axioms.
- The predecessor axiom in the current construction can be generalized to comprehension principles that are independently justifiable, which strengthens the argument that the axiom is not intrinsically mathematical.
- In a future construction, the modal axiom of possible richness of objects may no longer be required, eliminating the need for its justification.
- It will be an interesting question for future research to determine whether the predecessor axiom can be similarly generalized in this future construction.

Methodologically, we can conclude that:

- Our embedding can successfully reproduce even complex constructions and reasoning in our target system AOT.
- We can achieve our goal to provide a natural and readable implementation that accurately reproduces syntax and reasoning in AOT without the need of keeping complex translations in mind.
- The automation infrastructure of Isabelle can be preserved even for complex constructions in the target system.
- Using our method we could provide insights into the construction and efficiently analyze potential extensions.

6. Higher-Order Object Theory

While the second-order fragment of AOT is expressive enough for a variety of applications, including applications in *natural mathematics*, as demonstrated in the last chapter at the example of the analysis of natural numbers, the theory can be generalized to a full type-theoretic higher-order version. A notable application of this generalized version of AOT is the analysis of *theoretical mathematics*.

While natural mathematics involves the construction of mathematical objects directly by abstracting exemplification patterns, and their properties are derived from the principles of AOT itself, theoretical mathematics involves analyzing mathematical theories themselves (as well as their objects, axioms and relations) as abstract objects.

While a full discussion of the type-theoretic version of AOT is beyond the scope of this thesis, this chapter will provide a short, informal overview of its construction and the challenges in constructing an embedding of it in Isabelle/HOL.

Note that while we reuse the notational conventions of our embedding for consistency with the last chapters (e.g. we use square brackets in exemplification and encoding formulas and the free-variable notation discussed in section 4.7.2), this chapter is *not* written relative to an Isabelle representation, so in contrast to the last chapters, none of the statements and terms are cited from an embedding. We forgo marking the statements in this chapter using vertical bars at the page margin.

6.1. Overview of Higher-Order Object Theory

Our description is based on an at the time of writing unpublished draft of a chapter of PLM. However, the full type-theoretic version of AOT is also already discussed in [61] and a simplified version serves as the basis of the upcoming paper *A Defense of Logicism* jointly authored by Hannes Leitgeb, Uri Nodelman and Edward Zalta (see [33]).

We already hinted at AOT's system of types in section 3.2. Formally, it involves the following types:

- i is a type.
- Whenever t_1, \dots, t_n are types ($n \geq 0$), $\langle t_1, \dots, t_n \rangle$ is a type.

i is the primitive type of individuals, $\langle t_1, \dots, t_n \rangle$ is the type of relations among n objects of the respective types t_1, \dots, t_n . Zero-place relations, i.e. relations of type $\langle \rangle$, form the type of propositions. $\langle i \rangle$ is the type of properties among individuals. $\langle \langle i \rangle \rangle$ is the type of properties of properties of individuals. $\langle \langle i \rangle, \langle \rangle \rangle$ is the type of binary relations between properties and propositions, etc.

The distinction between exemplification and encoding is reproduced for higher-order types, i.e. the language involves exemplification formulas of the form $[\tau^{(t_1, \dots, t_n)}] \tau^{\tau_1} \dots \tau^{\tau_n}$ and encoding formulas of the form $\tau^{\tau_1} \dots \tau^{\tau_n} [\tau^{(t_1, \dots, t_n)}]$.

Furthermore, the distinction between ordinary and abstract objects is generalized to all types. I.e. for every type t there is a distinguished constant $E^{(t)}$ exemplified by all concrete objects of type t , which yields definitions of *being ordinary* and *being abstract* at every type.

While the definitions and axiom system are similar to the second-order version described in sections 3.2 and 3.3, there are some notable differences. The following is a non-exhaustive list:

- Relation identity for relations of type $\langle t \rangle$ is defined as:¹

$$F = G \equiv_{df} ([O!]F \ \& \ [O!]G \ \& \ \Box \forall x(x[F] \equiv x[G])) \vee ([A!]F \ \& \ [A!]G \ \& \ \Box \forall \mathcal{H}(F[\mathcal{H}] \equiv G[\mathcal{H}]))$$
- It is axiomatic that significant λ -expressions denote ordinary relations.
- η -conversion is restricted to ordinary relations.

Notably, the comprehension principle for abstract objects is retained at all types t . I.e. let α be of type t and F be of type $\langle t \rangle$, then the following is an axiom:

$$\exists \alpha([A!]\alpha \ \& \ \forall F(\alpha[F] \equiv \varphi\{F\}))$$

6.2. Applications to Theoretical Mathematics

The analysis of Theoretical Mathematics in higher-order object theory was described in [61] and a variant is discussed in [33].

While a full-discussion of the subtleties involved again goes beyond the scope of this thesis, we illustrate the general idea at the example of the representation of Zermelo-Fraenkel set-theory as an abstract object ZF in higher-order AOT.

Technically, a mathematical theory in AOT is a *situation*, i.e. an abstract object that encodes only propositional properties.² So we can reuse the notation $T \models p$ as the proposition p is true in theory T .

One of the cornerstones of the analysis is the *Importation Principle*, stated in [33] as follows:

When φ is a closed theorem of T , then $T \models \varphi^*$ shall be an axiom, where φ^* is the result of indexing every occurrence of a term or predicate of T to T .

So taking S as ZF 's property of *being a set*, it is a theorem of ZF that:

$$\vdash_{ZF} \neg \exists y([S]y \ \& \ y \in \emptyset)$$

This theorem can be imported to AOT using the following instance of the Importation Principle:

¹ n -ary relation identity for $n \geq 2$ and proposition identity are extended in a similar manner to account for abstract n -place relations, resp. propositions.

²Recall the discussion in section 3.5.2.

$$ZF \models \neg \exists y ([S_{ZF}]y \ \& \ y \in_{ZF} \emptyset_{ZF})$$

Furthermore, the involved indexed terms of ZF are in turn abstract objects in AOT, e.g.

$$\emptyset_{ZF} = \iota x ([A!]x \ \& \ \forall F (x[F] \equiv ZF \models [F]\emptyset_{ZF}))$$

$$S_{ZF} = \iota F ([A!]F \ \& \ \forall \mathcal{F} (F[\mathcal{F}] \equiv ZF \models [\mathcal{F}]S_{ZF}))$$

$$\in_{ZF} = \iota R ([A!]R \ \& \ \forall \mathcal{R} (R[\mathcal{R}] \equiv ZF \models [\mathcal{R}]\in_{ZF}))$$

Exemplifying properties in ZF can be translated to encoding claims in AOT. E.g. in ZF, \emptyset exemplifies the property $[\lambda x \neg \exists y ([S]y \ \& \ y \in x)]$. This property can be captured as an *abstract property* in AOT that is *encoded* by \emptyset_{ZF} :³

$$\emptyset_{ZF} [[\lambda x \neg \exists y ([S_{SF}]y \ \& \ y \in_{ZF} x)]_{ZF}]$$

While a detailed account of the construction and its implications is the topic of the upcoming paper [33], we will discuss the general issue of embedding higher-order AOT in Isabelle/HOL in the next sections.

6.3. Bounded Models

[33] constructs minimal extensional models for the simplified version of higher-order AOT it uses for its argumentation. This construction defines the *height* of a type t , written $h(t)$, and the *width* of a type t , written $w(t)$ as:

- $h(i) = 0$
- $h(\langle \rangle) = 1$
- $h(\langle t_1, \dots, t_n \rangle) = 1 + \max\{h(t_1), \dots, h(t_n)\}$
- $w(i) = 1$
- $w(\langle \rangle) = 1$
- $w(\langle t_1, \dots, t_n \rangle) = \sum_1^k w(t_k)$

[33] then presents a concrete model construction for bounded languages $\mathcal{L}_{n,m}$ that are *cut off* at width n and height m , i.e. the well-formed expressions of the language $\mathcal{L}_{n,m}$ are the expressions of the unbounded language \mathcal{L} in which only terms of type t are well-formed, if $w(t) \leq n$ and $h(t) \leq m$. In particular, types of height m only involve ordinary objects, not abstract objects. For example, the second-order fragment described in the last chapters, is cut off at height 1 : while it involves abstract individuals, all relations and propositions are ordinary. Furthermore, while the second-order fragment considers properties of objects (height 1), it does not consider higher-order relations like properties of properties or properties of propositions.⁴

While we expect it to be feasible to construct a representation in Isabelle/HOL that allows for an arbitrary parameter as cut-off in height (and potentially width, though it may be possible to keep width unbounded), we expect the details of such a construction

³While λ -expressions in higher-order AOT are ordinary, theory-indexed λ -expressions are abstract.

⁴Note that the cut-off involves subtle changes in the precise formulation of the definitions and the axiom system.

to be non-trivial due to the non-uniform nature of the representation sets of types. We leave the construction of such an embedding to future research.

6.4. Abstract Objects in Unbounded Models

While, arguably, a construction of models for higher-order object theory with a fixed, but arbitrary cutoff may be sufficient for all intents and purposes, the issue of constructing unbounded models (resp. an unrestricted embedding of higher-order AOT in HOL) is nevertheless interesting: theoretically, it may provide insights into the relative strength of higher-order AOT compared to HOL. Technically, unbounded models have the advantage of being uniform in all types, which is beneficial for a generic implementation.

However, if we consider the extent of the generalized comprehension principle of abstract objects and the identity conditions of abstract objects, it becomes clear that the construction of such models is not trivial.

In particular, note that the comprehension principle for abstract individuals has the following instance:

$$\exists x ([A!]x \ \& \ \forall F (x[F] \equiv ([O!]F \ \& \ \varphi\{F\} \vee [A!]F \ \& \ \forall \mathcal{F} (F[\mathcal{F}] \equiv \psi\{\mathcal{F}\}))))$$

Such an abstract object x (at type i) encodes all ordinary properties F (at type $\langle i \rangle$) that satisfy an arbitrary condition φ and all abstract properties F that encode exactly those properties of properties \mathcal{F} (at type $\langle\langle i \rangle\rangle$) that satisfy an arbitrary condition ψ on \mathcal{F} .

Now for two such abstract objects (at type i) to be identical, they not only have to encode the same ordinary properties (at type $\langle i \rangle$), but also the same abstract properties (at type $\langle i \rangle$). Those abstract properties in turn are identical, if they encode the same properties of properties (at type $\langle\langle i \rangle\rangle$).

This can be iterated further, since there are also abstract properties of properties among individuals that may encode properties of properties of properties among individuals, etc. pp.

While we leave a more detailed and rigorous analysis to future research, we try to informally illustrate the expected size of the set of abstract objects in unbounded models.

Thinking in terms of Aczel models, let O_t be the set of ordinary objects at type t and S_t the set of special urelements of type t . Now the set of relations among objects of type t , i.e. $O_{\langle t \rangle}$ will be at least as large as the power set $\mathcal{P}(O_t \cup S_t)$. For simplicity, we consider minimal, extensional Aczel models, in which we have $O_{\langle t \rangle} = \mathcal{P}(O_t \cup S_t)$.

If we restrict ourselves to unary relations and write 0 for the type of ordinary individuals i , 1 for the type of relations among individuals $\langle i \rangle$ and so on, i.e. in general we choose $n + 1$ for unary relations among the type we identified with n , we get the following:

$$\begin{aligned} O_1 &= \mathcal{P}(O_0 \cup S_0) \\ O_2 &= \mathcal{P}(O_1 \cup S_1) \\ O_3 &= \mathcal{P}(O_2 \cup S_2) \\ &\dots \end{aligned}$$

Now if we, solely for the purpose of arriving at a crude size estimate, further assume O_0 is empty and $S_i = S_0 = S$, we get:

$$\begin{aligned} O_0 &= \emptyset \\ O_1 &= \mathcal{P}(O_0 \cup S) = \mathcal{P}(S) \\ O_2 &= \mathcal{P}(O_1 \cup S) = \mathcal{P}(\mathcal{P}(S) \cup S) \supseteq \mathcal{P}(\mathcal{P}(S)) \cup \mathcal{P}(S) \\ O_3 &= \mathcal{P}(O_2 \cup S) = \mathcal{P}(\mathcal{P}(\mathcal{P}(S) \cup S) \cup S) \supseteq \mathcal{P}(\mathcal{P}(\mathcal{P}(S))) \cup \mathcal{P}(\mathcal{P}(S)) \cup \mathcal{P}(S) \\ &\dots \end{aligned}$$

Now if we assume that S has only one element and identify it with $\mathcal{P}(\emptyset)$, and (informally for the purpose of illustrating) consider the limit O_ω of relations at countably infinite height, we arrive at a model of the natural numbers, i.e. $|O_\omega| \geq |\mathbb{N}|$.

The set of abstract objects at type $m - 1$ is the power set of ordinary and abstract objects of type m , i.e. $A_{m-1} = \mathcal{P}(O_m \cup A_m)$. So we get:

$$\begin{aligned} A_{m-1} &= \mathcal{P}(O_m \cup A_m) \\ A_{m-2} &= \mathcal{P}(O_{m-1} \cup A_{m-1}) = \mathcal{P}(O_{m-1} \cup \mathcal{P}(O_m \cup A_m)) \\ A_{m-3} &= \mathcal{P}(O_{m-2} \cup A_{m-2}) = \mathcal{P}(O_{m-2} \cup \mathcal{P}(O_{m-1} \cup \mathcal{P}(O_m \cup A_m))) \\ &\dots \\ A_0 &= \mathcal{P}(O_1 \cup \mathcal{P}(O_2 \cup \mathcal{P}(O_3 \cup \mathcal{P}(\dots \cup A_m)\dots))) \end{aligned}$$

In particular, no finite application of power set operations is enough to construct A_0 from the (illustrative) limit set A_ω , which in turn would be the power set of O_ω , i.e. of a set at least as large as the natural numbers.

While this informal argument may not hold up to scrutiny, it is safe to say that the set of abstract objects in an unbounded model of higher-order object theory will be huge. We wouldn't be surprised if a future more rigorous analysis were to conclude that the set of abstract individuals in non-trivial models of higher-order AOT had to be sufficiently large to form a model of ZF itself (resp. that the cardinality of A_0 is strongly inaccessible).

Consequently, a verifiably sound implementation relative to the unextended background theory of Isabelle/HOL may be challenging, since the expressive power of higher-order AOT may be on par with or even exceed the expressive power of this choice of a meta-logic. However, even if this turns out to be the case, it may be possible to construct a representation based on a stronger extension of Isabelle/HOL, for example HOLZF [40], which axiomatizes the ZF universe itself as a type in HOL. The feasibility of such an embedding as well as the question of the relative strength of higher-order object theory compared to HOL, are interesting questions for future research.

7. Conclusion

We have presented an implementation of a foundational metaphysical theory in an automated reasoning environment by leveraging and extending the concept of *shallow semantic embeddings* (SSEs) in classical higher-order logic.

Methodologically, we could demonstrate that:

- The SSE approach is scalable and can not only be used to analyze isolated arguments, but can also be applied to full metaphysical theories.
- We can construct an accurate implementation of the axioms and deductive system of the target theory using abstraction layers.
- The automation infrastructure of Isabelle/HOL can be preserved and applied to construct proofs that accurately correspond to derivations in the target system.

While some constructions and modes of reasoning in a target system may be challenging to reproduce in an embedding, we developed several techniques to address such cases, including the definition of custom theorem attributes and proving methods and the extension of Isabelle’s Isar language by specialized outer syntax commands. Furthermore, we devised a system of syntax translations on a custom sub-grammar of Isabelle’s inner syntax to construct an accurate representation of the syntax of our target theory.

Using these techniques, it is not only possible to technically reproduce the logic of a target theory, but also to construct a nearly transparent representation of its syntax and reasoning flow. This allows for an efficient and effortless exchange of results between traditional pen-and-paper based reasoning and the computerized implementation.

This way, we can effectively arrive at a dedicated automated theorem proving environment for our target system, while retaining a verifiably consistent meta-logical backend.

The construction of such a framework is not merely a technical exercise, but can trigger a fruitful exchange that, in our case, led to significant improvements of the analyzed theory itself.

In particular, in the application of our method to the second-order fragment of *Abstract Object Theory* (AOT), we could demonstrate that:

- A semantic implementation can serve as a flexible backend that can be used to explore variations and axiomatic extensions of the target system.
- Our semantic analysis could significantly contribute especially to the theoretical understanding of the conditions, in AOT, under which relations exist. This has led to considerable improvements in the formulation of AOT.

- We can verify complex constructions and reasoning within a given axiomatization of the target system and efficiently analyze the effects of variations and extensions of such constructions.

Concretely, we can confirm that AOT can serve as a sound basis for a variant of Frege's construction of natural numbers. We can verify that the Dedekind-Peano postulates thus become consistently derivable in AOT.

We could contribute to the evolution of this construction and provide insights into the nature of its required additional axioms, and into variants of the construction. This includes a generalization of one of the axioms that may serve to strengthen the philosophical justification of the construction.

Interestingly, our results simultaneously support the use of HOL as universal meta-logic in that we can demonstrate that the SSE approach can be used to accurately represent even challenging logical theories, while our results also strengthen the position of our target theory AOT as foundational system in confirming its ability to provide a philosophically grounded construction of mathematical objects.

In this context, an attempt of an implementation of the full type-theoretic higher-order version of AOT using the SSE approach, as well as the formal analysis of its relative strength compared to HOL and ZF are fascinating opportunities for future research.

A. Isabelle Theory

A.1. Model for the Logic of AOT

```
1  (*<*)
2  theory AOT_model
3    imports Main "HOL-Cardinals.Cardinals"
4  begin
5
6  declare[[typedef_overloaded]]
7  (*>*)
8
9  section<Model for the Logic of AOT>
10
11 text<We introduce a primitive type for hyperintensional propositions.>
12 typedef o
13
14 text<To be able to model modal operators following Kripke semantics,
15 we introduce a primitive type for possible worlds and assert, by axiom,
16 that there is a surjective function mapping propositions to the
17 boolean-valued functions acting on possible worlds. We call the result
18 of applying this function to a proposition the Montague intension
19 of the proposition.>
20 typedef w -<The primitive type of possible worlds.>
21 axiomatization AOT_model_do :: <o $\Rightarrow$ (w $\Rightarrow$ bool)> where
22   do_surj: <surj AOT_model_do>
23
24 text<The axioms of PLM require the existence of a non-actual world.>
25 consts w0 :: w -<The designated actual world.>
26 axiomatization where AOT_model_nonactual_world: < $\exists w . w \neq w_0$ >
27
28 text<Validity of a proposition in a given world can now be modelled as the result
29 of applying that world to the Montague intension of the proposition.>
30 definition AOT_model_valid_in :: <w $\Rightarrow$ o $\Rightarrow$ bool> where
31   <AOT_model_valid_in w  $\varphi \equiv$  AOT_model_do  $\varphi$  w>
32
33 text<By construction, we can choose a proposition for any given Montague intension,
34 s.t. the proposition is valid in a possible world iff the Montague intension
35 evaluates to true at that world.>
36 definition AOT_model_proposition_choice :: <(w $\Rightarrow$ bool)  $\Rightarrow$  o> (binder < $\varepsilon_o$  > 8)
37   where < $\varepsilon_o$  w.  $\varphi$  w  $\equiv$  (inv AOT_model_do)  $\varphi$ >
38 lemma AOT_model_proposition_choice_simp: <AOT_model_valid_in w ( $\varepsilon_o$  w.  $\varphi$  w) =  $\varphi$  w>
39   by (simp add: surj_f_inv_f[OF do_surj] AOT_model_valid_in_def
40       AOT_model_proposition_choice_def)
41
42 text<Nitpick can trivially show that there are models for the axioms above.>
43 lemma <True> nitpick[satisfy, user_axioms, expect = genuine] ..
44
45 typedef  $\omega$  -<The primitive type of ordinary objects/urelements.>
46
47 text<Validating extended relation comprehension requires a large set of
48 special urelements. For simple models that do not validate extended
49 relation comprehension (and consequently the predecessor axiom in the
50 theory of natural numbers), it suffices to use a primitive type as @<text  $\sigma$ >,
51 i.e. @<theory_text <typedef  $\sigma$ >>.>
52 typedef  $\sigma'$ 
53 typedef  $\sigma =$  <UNIV::((w  $\Rightarrow$  w  $\Rightarrow$  bool) set  $\times$  (w  $\Rightarrow$  w  $\Rightarrow$  bool) set  $\times$   $\sigma'$ ) set> ..
```

```

54
55 typedef1 null - <Null-urelements representing non-denoting terms.>
56
57 datatype v = ωv ω | σv σ | is_nullv: nullv null - <Type of urelements>
58
59 text<Urrelations are proposition-valued functions on urelements.
60   Urrelations are required to evaluate to necessarily false propositions for
61   null-urelements (note that there may be several distinct necessarily false
62   propositions).>
63 typedef urrel = <{ φ . ∀ x w . ¬AOT_model_valid_in w (φ (nullv x)) }>
64   by (rule exI[where x=<λ x . (εo w . ¬is_nullv x)>])
65   (auto simp: AOT_model_proposition_choice_simp)
66
67 text<Abstract objects will be modelled as sets of urrelations and will
68   have to be mapped surjectively into the set of special urelements.
69   We show that any mapping from abstract objects to special urelements
70   has to involve at least one large set of collapsed abstract objects.
71   We will use this fact to extend arbitrary mappings from abstract objects
72   to special urelements to surjective mappings.>
73 lemma ασ_pigeonhole:
74   - <For any arbitrary mapping @term ασ from sets of urrelations to special
75     urelements, there exists an abstract object x, s.t. the cardinal of the set
76     of special urelements is strictly smaller than the cardinal of the set of
77     abstract objects that are mapped to the same urelement as x under @term ασ.>
78   <∃x . |UNIV::σ set| <o |{y . ασ x = ασ y}|>
79   for ασ :: <urrel set ⇒ σ>
80 proof(rule ccontr)
81   have card_σ_set_set_bound: <|UNIV::σ set set| ≤o |UNIV::urrel set|>
82   proof -
83     let ?pick = <λu s . εo w . case u of (σv s') ⇒ s' ∈ s | _ ⇒ False>
84     have <∃f :: σ set ⇒ urrel . inj f>
85     proof
86       show <inj (λs . Abs_urrel (λu . ?pick u s))>
87       proof(rule injI)
88         fix x y
89         assume <Abs_urrel (λu . ?pick u x) = Abs_urrel (λu . ?pick u y)>
90         hence <(λu . ?pick u x) = (λu . ?pick u y)>
91           by (auto intro!: Abs_urrel_inject[THEN iffD1]
92             simp: AOT_model_proposition_choice_simp)
93         hence <AOT_model_valid_in wo (?pick (σv s) x) =
94           AOT_model_valid_in wo (?pick (σv s) y)>
95           for s by metis
96         hence <(s ∈ x) = (s ∈ y)> for s
97           by (auto simp: AOT_model_proposition_choice_simp)
98         thus <x = y>
99           by blast
100       qed
101     qed
102     thus ?thesis
103     by (metis card_of_image inj_imp_surj_inv)
104   qed
105
106 text<Assume, for a proof by contradiction, that there is no large collapsed set.>
107 assume <∄x . |UNIV::σ set| <o |{y . ασ x = ασ y}|>
108 hence A: <∀x . |{y . ασ x = ασ y}| ≤o |UNIV::σ set|>
109   by auto
110 have union_univ: <(⋃x ∈ range(inv ασ) . {y . ασ x = ασ y}) = UNIV>
111   by auto (meson f_inv_into_f range_eqI)
112
113 text<We refute by case distinction: there is either finitely many or
114   infinitely many special urelements and in both cases we can derive
115   a contradiction from the assumption above.>
116 {

```

```

117 text<Finite case.>
118 assume finite_σ_set: <finite (UNIV::σ set)>
119 hence finite_collapsed: <finite {y . ασ x = ασ y}> for x
120   using A card_of_ordLeq_infinite by blast
121 hence 0: <∀x . card {y . ασ x = ασ y} ≤ card (UNIV::σ set)>
122   by (metis A finite_σ_set card_of_ordLeq inj_on_iff_card_le)
123 have 1: <card (range (inv ασ)) ≤ card (UNIV::σ set)>
124   using finite_σ_set card_image_le by blast
125 hence 2: <finite (range (inv ασ))>
126   using finite_σ_set by blast
127
128 define n where <n = card (UNIV::urrel set set)>
129 define m where <m = card (UNIV::σ set)>
130
131 have <n = card (⋃x ∈ range(inv ασ) . {y . ασ x = ασ y})>
132   unfolding n_def using union_univ by argo
133 also have <... ≤ (∑i∈range (inv ασ). card {y . ασ i = ασ y})>
134   using card_UN_le 2 by blast
135 also have <... ≤ (∑i∈range (inv ασ). card (UNIV::σ set))>
136   by (metis (no_types, lifting) 0 sum_mono)
137 also have <... ≤ card (range (inv ασ)) * card (UNIV::σ set)>
138   using sum_bounded_above by auto
139 also have <... ≤ card (UNIV::σ set) * card (UNIV::σ set)>
140   using 1 by force
141 also have <... = m*m>
142   unfolding m_def by blast
143 finally have n_upper: <n ≤ m*m>.
144
145 have <finite (⋃x ∈ range(inv ασ) . {y . ασ x = ασ y})>
146   using 2 finite_collapsed by blast
147 hence finite_αset: <finite (UNIV::urrel set set)>
148   using union_univ by argo
149
150 have <2^2^m = (2::nat)^(card (UNIV::σ set set))>
151   by (metis Pow_UNIV card_Pow finite_σ_set m_def)
152 moreover have <card (UNIV::σ set set) ≤ (card (UNIV::urrel set))>
153   using card_σ_set_set_bound
154   by (meson Finite_Set.finite_set card_of_ordLeq finite_αset
155       finite_σ_set inj_on_iff_card_le)
156 ultimately have <2^2^m ≤ (2::nat)^(card (UNIV::urrel set))>
157   by simp
158 also have <... = n>
159   unfolding n_def
160   by (metis Finite_Set.finite_set Pow_UNIV card_Pow finite_αset)
161 finally have <2^2^m ≤ n> by blast
162 hence <2^2^m ≤ m*m> using n_upper by linarith
163 moreover {
164   have <(2::nat)^(2^m) ≥ (2^(m + 1))>
165     by (metis Suc_eq_plus1 Suc_leI less_exp one_le_numeral power_increasing)
166   also have <(2^(m + 1)) = (2::nat) * 2^m>
167     by auto
168   have <m < 2^m>
169     by (simp add: less_exp)
170   hence <m*m < (2^m)*(2^m)>
171     by (simp add: mult_strict_mono)
172   moreover have <... = 2^(m+m)>
173     by (simp add: power_add)
174   ultimately have <m*m < 2^(m + m)> by presburger
175   moreover have <m+m ≤ 2^m>
176     proof (induct m)
177       case 0
178       thus ?case by auto
179     next

```

```

180     case (Suc m)
181     thus ?case
182     by (metis Suc_leI less_exp mult_2 mult_le_mono2 power_Suc)
183   qed
184   ultimately have <math>m * m < 2^{2^m}</math>
185     by (meson less_le_trans one_le_numeral power_increasing)
186 }
187 ultimately have False by auto
188 }
189 moreover {
190   text<Infinite case.>
191   assume <infinite (UNIV:: $\sigma$  set)>
192   hence Cinf $\sigma$ : <Cinfinite |UNIV:: $\sigma$  set|>
193     by (simp add: cinfinite_def)
194   have 1: <|range (inv  $\alpha\sigma$ )|  $\leq$  |UNIV:: $\sigma$  set|>
195     by auto
196   have 2: < $\forall i \in \text{range (inv } \alpha\sigma). |\{y . \alpha\sigma i = \alpha\sigma y\}| \leq$  |UNIV:: $\sigma$  set|>
197   proof
198     fix i :: <urrel set>
199     assume <math>i \in \text{range (inv } \alpha\sigma)</math>
200     show <math>|\{y . \alpha\sigma i = \alpha\sigma y\}| \leq |UNIV:: $\sigma$  set|>
201       using A by blast
202   qed
203   have <math>|\bigcup ((\lambda i. \{y. \alpha\sigma i = \alpha\sigma y\}) ` (\text{range (inv } \alpha\sigma)))| \leq
204     |Sigma (range (inv  $\alpha\sigma$ )) ( $\lambda i. \{y. \alpha\sigma i = \alpha\sigma y\})|$ >
205     using card_of_UNION_Sigma by blast
206   hence <math>|\text{UNIV::urrel set set}| \leq
207     |Sigma (range (inv  $\alpha\sigma$ )) ( $\lambda i. \{y. \alpha\sigma i = \alpha\sigma y\})|$ >
208     using union_univ by argo
209   moreover have <math>|\text{Sigma (range (inv } \alpha\sigma)) (\lambda i. \{y. \alpha\sigma i = \alpha\sigma y\})| \leq |UNIV:: $\sigma$  set|>
210     using card_of_Sigma_ordLeq_Cinfinite[OF Cinf $\sigma$ , OF 1, OF 2] by blast
211   ultimately have <math>|\text{UNIV::urrel set set}| \leq |UNIV:: $\sigma$  set|>
212     using ordLeq_transitive by blast
213   moreover {
214     have <math>|\text{UNIV::}\sigma \text{ set}| < |UNIV:: $\sigma$  set set|>
215       by auto
216     moreover have <math>|\text{UNIV::}\sigma \text{ set set}| \leq |UNIV::urrel set|>
217       using card_ $\sigma$ _set_set_bound by blast
218     moreover have <math>|\text{UNIV::urrel set}| < |UNIV::urrel set set|>
219       by auto
220     ultimately have <math>|\text{UNIV::}\sigma \text{ set}| < |UNIV::urrel set set|>
221       by (metis ordLess_imp_ordLeq ordLess_ordLeq_trans)
222   }
223   ultimately have False
224     using not_ordLeq_ordLess by blast
225 }
226 ultimately show False by blast
227 qed
228
229 text<We introduce a mapping from abstract objects (i.e. sets of urrelations) to
230 special urelements @<math>\langle \alpha\sigma \rangle</math> that is surjective and distinguishes all
231 abstract objects that are distinguished by a (not necessarily surjective)
232 mapping @<math>\langle \alpha\sigma' \rangle</math>. @<math>\langle \alpha\sigma' \rangle</math> will be used to model extended relation
233 comprehension.>
234 consts  $\alpha\sigma'$  :: <urrel set  $\Rightarrow$   $\sigma$ >
235 consts  $\alpha\sigma$  :: <urrel set  $\Rightarrow$   $\sigma$ >
236
237 specification( $\alpha\sigma$ )
238    $\alpha\sigma$ _surj: <surj  $\alpha\sigma$ >
239    $\alpha\sigma$ _ $\alpha\sigma'$ : < $\alpha\sigma x = \alpha\sigma y \implies \alpha\sigma' x = \alpha\sigma' y$ >
240 proof -
241   obtain x where x_prop: <math>|\text{UNIV::}\sigma \text{ set}| < | $\{y. \alpha\sigma' x = \alpha\sigma' y\}|$ >
242     using  $\alpha\sigma$ _pigeonhole by blast

```



```

243 have <∃f :: urrel set ⇒ σ . f ' {y. ασ' x = ασ' y} = UNIV ∧ f x = ασ' x>
244 proof -
245   have <∃f :: urrel set ⇒ σ . f ' {y. ασ' x = ασ' y} = UNIV>
246     by (simp add: x_prop card_of_ordLeq2 ordLess_imp_ordLeq)
247   then obtain f :: <urrel set ⇒ σ> where <f ' {y. ασ' x = ασ' y} = UNIV>
248     by presburger
249   moreover obtain a where <f a = ασ' x> and <ασ' a = ασ' x>
250     by (smt (verit, best) calculation UNIV_I image_iff mem_Collect_eq)
251   ultimately have <(f (a := f x, x := f a)) ' {y. ασ' x = ασ' y} = UNIV ∧
252     (f (a := f x, x := f a)) x = ασ' x>
253     by (auto simp: image_def)
254   thus ?thesis by blast
255 qed
256 then obtain f where fimage: <f ' {y. ασ' x = ασ' y} = UNIV>
257   and fx: <f x = ασ' x>
258   by blast
259
260 define ασ :: <urrel set ⇒ σ> where
261   <ασ ≡ λ urrels . if ασ' urrels = ασ' x ∧ f urrels ∉ range ασ'
262     then f urrels
263     else ασ' urrels>
264
265 have <surj ασ>
266 proof -
267   {
268     fix s :: σ
269     {
270       assume <s ∈ range ασ'>
271       hence 0: <ασ' (inv ασ' s) = s>
272         by (meson f_inv_into_f)
273       {
274         assume <s = ασ' x>
275         hence <ασ x = s>
276           using ασ_def fx by presburger
277         hence <∃f . ασ (f s) = s>
278           by auto
279       }
280       moreover {
281         assume <s ≠ ασ' x>
282         hence <ασ (inv ασ' s) = s>
283           unfolding ασ_def 0 by presburger
284         hence <∃f . ασ (f s) = s>
285           by blast
286       }
287       ultimately have <∃f . ασ (f s) = s>
288         by blast
289     }
290     moreover {
291       assume <s ∉ range ασ'>
292       moreover obtain urrels where <f urrels = s> and <ασ' x = ασ' urrels>
293         by (smt (verit, best) UNIV_I fimage image_iff mem_Collect_eq)
294       ultimately have <ασ urrels = s>
295         using ασ_def by presburger
296       hence <∃f . ασ (f s) = s>
297         by (meson f_inv_into_f range_eqI)
298     }
299     ultimately have <∃f . ασ (f s) = s>
300       by blast
301   }
302   thus ?thesis
303     by (metis surj_def)
304 qed
305 moreover have <∀x y. ασ x = ασ y ⟶ ασ' x = ασ' y>
306   by (metis ασ_def rangeI)

```

```

306   ultimately show ?thesis
307   by blast
308 qed
309
310 text<For extended models that validate extended relation comprehension
311   (and consequently the predecessor axiom), we specify which
312   abstract objects are distinguished by @{const  $\alpha\sigma'$ }.>
313
314 definition urrel_to_wrel :: <urrel  $\Rightarrow$  ( $\omega \Rightarrow w \Rightarrow$  bool)> where
315   <urrel_to_wrel  $\equiv$   $\lambda$  r u w . AOT_model_valid_in w (Rep_urrel r ( $\omega v$  u))>
316 definition wrel_to_urrel :: <( $\omega \Rightarrow w \Rightarrow$  bool)  $\Rightarrow$  urrel> where
317   <wrel_to_urrel  $\equiv$   $\lambda$   $\varphi$  . Abs_urrel
318     ( $\lambda$  u .  $\varepsilon_o$  w . case u of  $\omega v$  x  $\Rightarrow$   $\varphi$  x w | _  $\Rightarrow$  False)>
319
320 definition AOT_urrel_wequiv :: <urrel  $\Rightarrow$  urrel  $\Rightarrow$  bool> where
321   <AOT_urrel_wequiv  $\equiv$   $\lambda$  r s .  $\forall$  u v . AOT_model_valid_in v (Rep_urrel r ( $\omega v$  u)) =
322     AOT_model_valid_in v (Rep_urrel s ( $\omega v$  u))>
323
324 lemma urrel_wrel_quot: <Quotient3 AOT_urrel_wequiv urrel_to_wrel wrel_to_urrel>
325 proof(rule Quotient3I)
326   show <urrel_to_wrel (wrel_to_urrel a) = a> for a
327     unfolding wrel_to_urrel_def urrel_to_wrel_def
328     apply (rule ext)
329     apply (subst Abs_urrel_inverse)
330     by (auto simp: AOT_model_proposition_choice_simp)
331 next
332   show <AOT_urrel_wequiv (wrel_to_urrel a) (wrel_to_urrel a)> for a
333     unfolding wrel_to_urrel_def AOT_urrel_wequiv_def
334     apply (subst (1 2) Abs_urrel_inverse)
335     by (auto simp: AOT_model_proposition_choice_simp)
336 next
337   show <AOT_urrel_wequiv r s = (AOT_urrel_wequiv r r  $\wedge$  AOT_urrel_wequiv s s  $\wedge$ 
338     urrel_to_wrel r = urrel_to_wrel s)> for r s
339   proof
340     assume <AOT_urrel_wequiv r s>
341     hence <AOT_model_valid_in v (Rep_urrel r ( $\omega v$  u)) =
342       AOT_model_valid_in v (Rep_urrel s ( $\omega v$  u))> for u v
343       using AOT_urrel_wequiv_def by metis
344     hence <urrel_to_wrel r = urrel_to_wrel s>
345       unfolding urrel_to_wrel_def
346       by simp
347     thus <AOT_urrel_wequiv r r  $\wedge$  AOT_urrel_wequiv s s  $\wedge$ 
348       urrel_to_wrel r = urrel_to_wrel s>
349       unfolding AOT_urrel_wequiv_def
350       by auto
351   next
352     assume <AOT_urrel_wequiv r r  $\wedge$  AOT_urrel_wequiv s s  $\wedge$ 
353       urrel_to_wrel r = urrel_to_wrel s>
354     hence <AOT_model_valid_in v (Rep_urrel r ( $\omega v$  u)) =
355       AOT_model_valid_in v (Rep_urrel s ( $\omega v$  u))> for u v
356       by (metis urrel_to_wrel_def)
357     thus <AOT_urrel_wequiv r s>
358       using AOT_urrel_wequiv_def by presburger
359   qed
360 qed
361
362 specification ( $\alpha\sigma'$ )
363    $\alpha\sigma\_eq\_ord\_exts\_all$ :
364     < $\alpha\sigma'$  a =  $\alpha\sigma'$  b  $\implies$  ( $\bigwedge$  s . urrel_to_wrel s = urrel_to_wrel r  $\implies$  s  $\in$  a)
365      $\implies$  ( $\bigwedge$  s . urrel_to_wrel s = urrel_to_wrel r  $\implies$  s  $\in$  b)>
366    $\alpha\sigma\_eq\_ord\_exts\_ex$ :
367     < $\alpha\sigma'$  a =  $\alpha\sigma'$  b  $\implies$  ( $\exists$  s . s  $\in$  a  $\wedge$  urrel_to_wrel s = urrel_to_wrel r)
368      $\implies$  ( $\exists$  s . s  $\in$  b  $\wedge$  urrel_to_wrel s = urrel_to_wrel r)>

```

```

369 proof -
370   define  $\alpha\sigma_{\text{wit\_intersection}}$  where
371     < $\alpha\sigma_{\text{wit\_intersection}} \equiv \lambda \text{urrels} .$ 
372       {ordext .  $\forall \text{urrel} . \text{urrel\_to\_}\omega\text{rel } \text{urrel} = \text{ordext} \longrightarrow \text{urrel} \in \text{urrels}$ }>
373   define  $\alpha\sigma_{\text{wit\_union}}$  where
374     < $\alpha\sigma_{\text{wit\_union}} \equiv \lambda \text{urrels} .$ 
375       {ordext .  $\exists \text{urrel} \in \text{urrels} . \text{urrel\_to\_}\omega\text{rel } \text{urrel} = \text{ordext}$ }>
376
377   let ? $\alpha\sigma_{\text{wit}}$  = < $\lambda \text{urrels} .$ 
378     let ordexts =  $\alpha\sigma_{\text{wit\_intersection}}$  urrels in
379     let ordexts' =  $\alpha\sigma_{\text{wit\_union}}$  urrels in
380     (ordexts, ordexts', undefined)>
381   define  $\alpha\sigma_{\text{wit}}$  :: <urrel set  $\Rightarrow \sigma$ > where
382     < $\alpha\sigma_{\text{wit}} \equiv \lambda \text{urrels} . \text{Abs}_{\sigma} (? \alpha\sigma_{\text{wit}} \text{urrels})$ >
383   {
384     fix a b :: <urrel set> and r s
385     assume < $\alpha\sigma_{\text{wit}} a = \alpha\sigma_{\text{wit}} b$ >
386     hence 0: <{ordext.  $\forall \text{urrel} . \text{urrel\_to\_}\omega\text{rel } \text{urrel} = \text{ordext} \longrightarrow \text{urrel} \in a$ } =
387       {ordext.  $\forall \text{urrel} . \text{urrel\_to\_}\omega\text{rel } \text{urrel} = \text{ordext} \longrightarrow \text{urrel} \in b$ }>
388     unfolding  $\alpha\sigma_{\text{wit\_def}}$  Let_def
389     apply (subst (asm) Abs $_{\sigma}$ _inject)
390     by (auto simp:  $\alpha\sigma_{\text{wit\_intersection\_def}}$   $\alpha\sigma_{\text{wit\_union\_def}}$ )
391     assume <urrel_to_ $\omega$ rel s = urrel_to_ $\omega$ rel r  $\Longrightarrow$  s  $\in$  a> for s
392     hence <urrel_to_ $\omega$ rel r  $\in$ 
393       {ordext.  $\forall \text{urrel} . \text{urrel\_to\_}\omega\text{rel } \text{urrel} = \text{ordext} \longrightarrow \text{urrel} \in a$ }>
394     by auto
395     hence <urrel_to_ $\omega$ rel r  $\in$ 
396       {ordext.  $\forall \text{urrel} . \text{urrel\_to\_}\omega\text{rel } \text{urrel} = \text{ordext} \longrightarrow \text{urrel} \in b$ }>
397     using 0 by blast
398     moreover assume <urrel_to_ $\omega$ rel s = urrel_to_ $\omega$ rel r>
399     ultimately have <s  $\in$  b>
400     by blast
401   }
402   moreover {
403     fix a b :: <urrel set> and s r
404     assume < $\alpha\sigma_{\text{wit}} a = \alpha\sigma_{\text{wit}} b$ >
405     hence 0: <{ordext.  $\exists \text{urrel} \in a . \text{urrel\_to\_}\omega\text{rel } \text{urrel} = \text{ordext}$ } =
406       {ordext.  $\exists \text{urrel} \in b . \text{urrel\_to\_}\omega\text{rel } \text{urrel} = \text{ordext}$ }>
407     unfolding  $\alpha\sigma_{\text{wit\_def}}$ 
408     apply (subst (asm) Abs $_{\sigma}$ _inject)
409     by (auto simp: Let_def  $\alpha\sigma_{\text{wit\_intersection\_def}}$   $\alpha\sigma_{\text{wit\_union\_def}}$ )
410     assume <s  $\in$  a>
411     hence <urrel_to_ $\omega$ rel s  $\in$  {ordext.  $\exists \text{urrel} \in a . \text{urrel\_to\_}\omega\text{rel } \text{urrel} = \text{ordext}$ }>
412     by blast
413     moreover assume <urrel_to_ $\omega$ rel s = urrel_to_ $\omega$ rel r>
414     ultimately have <urrel_to_ $\omega$ rel r  $\in$ 
415       {ordext.  $\exists \text{urrel} \in b . \text{urrel\_to\_}\omega\text{rel } \text{urrel} = \text{ordext}$ }>
416     using "0" by argo
417     hence < $\exists s . s \in b \wedge \text{urrel\_to\_}\omega\text{rel } s = \text{urrel\_to\_}\omega\text{rel } r$ >
418     by blast
419   }
420   ultimately show ?thesis
421   by (safe intro!: exI[where x= $\alpha\sigma_{\text{wit}}$ ]; metis)
422 qed
423
424 text<We enable the extended model version.>
425 abbreviation (input) AOT_ExtendedModel where <AOT_ExtendedModel  $\equiv$  True>
426
427 text<Individual terms are either ordinary objects, represented by ordinary urelements,
428   abstract objects, modelled as sets of urelations, or null objects, used to
429   represent non-denoting definite descriptions.>
430 datatype  $\kappa$  =  $\omega\kappa$   $\omega$  |  $\alpha\kappa$  <urrel set> | is_null $\kappa$ : null $\kappa$  null
431

```

```

432 text<The mapping from abstract objects to urelements can be naturally
433   lifted to a surjective mapping from individual terms to urelements.>
434 primrec  $\kappa\nu$  ::  $\langle \kappa \Rightarrow \nu \rangle$  where
435    $\langle \kappa\nu (\omega\kappa\ x) = \omega\nu\ x \rangle$ 
436 |  $\langle \kappa\nu (\alpha\kappa\ x) = \sigma\nu (\alpha\sigma\ x) \rangle$ 
437 |  $\langle \kappa\nu (\text{null}\kappa\ x) = \text{null}\nu\ x \rangle$ 
438
439 lemma  $\kappa\nu\_surj$ :  $\langle surj\ \kappa\nu \rangle$ 
440   using  $\alpha\sigma\_surj$  by (metis  $\kappa\nu.simps(1)$   $\kappa\nu.simps(2)$   $\kappa\nu.simps(3)$   $\nu.exhaust$   $surj\_def$ )
441
442 text<By construction if the urelement of an individual term is exemplified by
443   an urelation, it cannot be a null-object.>
444 lemma  $urrel\_null\_false$ :
445   assumes  $\langle AOT\_model\_valid\_in\ w\ (Rep\_urrel\ f\ (\kappa\nu\ x)) \rangle$ 
446   shows  $\langle \neg is\_null\kappa\ x \rangle$ 
447   by (metis (mono_tags, lifting)  $assms$   $Rep\_urrel$   $\kappa.collapse(3)$   $\kappa\nu.simps(3)$ 
448      $mem\_Collect\_eq$ )
449
450 text<AOT requires any ordinary object to be @emph <possibly concrete> and that
451   there is an object that is not actually, but possibly concrete.>
452 consts  $AOT\_model\_concretew$  ::  $\langle \omega \Rightarrow w \Rightarrow bool \rangle$ 
453 specification ( $AOT\_model\_concretew$ )
454    $AOT\_model\_w\_concrete\_in\_some\_world$ :
455    $\langle \exists w . AOT\_model\_concretew\ x\ w \rangle$ 
456    $AOT\_model\_contingent\_object$ :
457    $\langle \exists x\ w . AOT\_model\_concretew\ x\ w \wedge \neg AOT\_model\_concretew\ x\ w_0 \rangle$ 
458   by (rule  $exI[where\ x = \lambda\_ w . w \neq w_0]$ ) (auto  $simp: AOT\_model\_nonactual\_world$ )
459
460 text<We define a type class for AOT's terms specifying the conditions under which
461   objects of that type denote and require the set of denoting terms to be
462   non-empty.>
463 class  $AOT\_Term$  =
464   fixes  $AOT\_model\_denotes$  ::  $\langle 'a \Rightarrow bool \rangle$ 
465   assumes  $AOT\_model\_denoting\_ex$ :  $\langle \exists x . AOT\_model\_denotes\ x \rangle$ 
466
467 text<All types except the type of propositions involve non-denoting terms. We
468   define a refined type class for those.>
469 class  $AOT\_IncompleteTerm$  =  $AOT\_Term$  +
470   assumes  $AOT\_model\_nondenoting\_ex$ :  $\langle \exists x . \neg AOT\_model\_denotes\ x \rangle$ 
471
472 text<Generic non-denoting term.>
473 definition  $AOT\_model\_nondenoting$  ::  $\langle 'a :: AOT\_IncompleteTerm \rangle$  where
474    $\langle AOT\_model\_nondenoting \equiv SOME\ \tau . \neg AOT\_model\_denotes\ \tau \rangle$ 
475 lemma  $AOT\_model\_nondenoting$ :  $\langle \neg AOT\_model\_denotes\ (AOT\_model\_nondenoting) \rangle$ 
476   using  $someI\_ex[OF\ AOT\_model\_nondenoting\_ex]$ 
477   unfolding  $AOT\_model\_nondenoting\_def$  by blast
478
479 text<@const  $AOT\_model\_denotes$  can trivially be extended to products of types.>
480 instantiation  $prod$  ::  $(AOT\_Term, AOT\_Term) AOT\_Term$ 
481 begin
482 definition  $AOT\_model\_denotes\_prod$  ::  $\langle 'a \times 'b \Rightarrow bool \rangle$  where
483    $\langle AOT\_model\_denotes\_prod \equiv \lambda(x,y) . AOT\_model\_denotes\ x \wedge AOT\_model\_denotes\ y \rangle$ 
484 instance proof
485   show  $\langle \exists x :: 'a \times 'b . AOT\_model\_denotes\ x \rangle$ 
486     by (simp  $add: AOT\_model\_denotes\_prod\_def\ AOT\_model\_denoting\_ex$ )
487 qed
488 end
489
490 text<We specify a transformation of proposition-valued functions on terms, s.t.
491   the result is fully determined by @emph <regular> terms. This will be required
492   for modelling n-ary relations as functions on tuples while preserving AOT's
493   definition of n-ary relation identity.>
494 locale  $AOT\_model\_irregular\_spec$  =

```

```

495 fixes AOT_model_irregular :: <'a ⇒ o> ⇒ 'a ⇒ o>
496 and AOT_model_regular :: <'a ⇒ bool>
497 and AOT_model_term_equiv :: <'a ⇒ 'a ⇒ bool>
498 assumes AOT_model_irregular_false:
499   <¬AOT_model_valid_in w (AOT_model_irregular φ x)>
500 assumes AOT_model_irregular_equiv:
501   <AOT_model_term_equiv x y ⇒
502     AOT_model_irregular φ x = AOT_model_irregular φ y>
503 assumes AOT_model_irregular_eqI:
504   <(∧ x . AOT_model_regular x ⇒ φ x = ψ x) ⇒
505     AOT_model_irregular φ x = AOT_model_irregular ψ x>
506
507 text<We introduce a type class for individual terms that specifies being regular,
508   being equivalent (i.e. conceptually @{emph <sharing urelements>}) and the
509   transformation on proposition-valued functions as specified above.>
510 class AOT_IndividualTerm = AOT_IncompleteTerm +
511   fixes AOT_model_regular :: <'a ⇒ bool>
512   fixes AOT_model_term_equiv :: <'a ⇒ 'a ⇒ bool>
513   fixes AOT_model_irregular :: <'a ⇒ o> ⇒ 'a ⇒ o>
514   assumes AOT_model_irregular_nondenoting:
515     <¬AOT_model_regular x ⇒ ¬AOT_model_denotes x>
516   assumes AOT_model_term_equiv_part_equivp:
517     <equivp AOT_model_term_equiv>
518   assumes AOT_model_term_equiv_denotes:
519     <AOT_model_term_equiv x y ⇒ (AOT_model_denotes x = AOT_model_denotes y)>
520   assumes AOT_model_term_equiv_regular:
521     <AOT_model_term_equiv x y ⇒ (AOT_model_regular x = AOT_model_regular y)>
522   assumes AOT_model_irregular:
523     <AOT_model_irregular_spec AOT_model_irregular AOT_model_regular
524       AOT_model_term_equiv>
525
526 interpretation AOT_model_irregular_spec AOT_model_irregular AOT_model_regular
527   AOT_model_term_equiv
528   using AOT_model_irregular .
529
530 text<Our concrete type for individual terms satisfies the type class of
531   individual terms.
532   Note that all unary individuals are regular. In general, an individual term
533   may be a tuple and is regular, if at most one tuple element does not denote.>
534 instantiation κ :: AOT_IndividualTerm
535 begin
536 definition AOT_model_term_equiv_κ :: <κ ⇒ κ ⇒ bool> where
537   <AOT_model_term_equiv_κ ≡ λ x y . κV x = κV y>
538 definition AOT_model_denotes_κ :: <κ ⇒ bool> where
539   <AOT_model_denotes_κ ≡ λ x . ¬is_nullκ x>
540 definition AOT_model_regular_κ :: <κ ⇒ bool> where
541   <AOT_model_regular_κ ≡ λ x . True>
542 definition AOT_model_irregular_κ :: <(κ ⇒ o) ⇒ κ ⇒ o> where
543   <AOT_model_irregular_κ ≡ SOME φ . AOT_model_irregular_spec φ
544     AOT_model_regular AOT_model_term_equiv>
545 instance proof
546   show <∃x :: κ. AOT_model_denotes x>
547     by (rule exI[where x=<ωκ undefined>])
548     (simp add: AOT_model_denotes_κ_def)
549 next
550   show <∃x :: κ. ¬AOT_model_denotes x>
551     by (rule exI[where x=<>nullκ undefined>])
552     (simp add: AOT_model_denotes_κ_def AOT_model_regular_κ_def)
553 next
554   show "¬AOT_model_regular x ⇒ ¬ AOT_model_denotes x" for x :: κ
555     by (simp add: AOT_model_regular_κ_def)
556 next
557   show <equivp (AOT_model_term_equiv :: κ ⇒ κ ⇒ bool)>

```

```

558     by (rule equivpI; rule reflpI exI sympI transpI)
559     (simp_all add: AOT_model_term_equiv_κ_def)
560 next
561   fix x y :: κ
562   show <AOT_model_term_equiv x y  $\implies$  AOT_model_denotes x = AOT_model_denotes y>
563     by (metis AOT_model_denotes_κ_def AOT_model_term_equiv_κ_def κ.exhaust_disc
564           κV.simps v.disc(1,3,5,6) is_ακ_def is_ωκ_def is_nullκ_def)
565 next
566   fix x y :: κ
567   show <AOT_model_term_equiv x y  $\implies$  AOT_model_regular x = AOT_model_regular y>
568     by (simp add: AOT_model_regular_κ_def)
569 next
570   have "AOT_model_irregular_spec (λ φ (x::κ) . εo w . False)
571         AOT_model_regular AOT_model_term_equiv"
572     by standard (auto simp: AOT_model_proposition_choice_simp)
573   thus <AOT_model_irregular_spec (AOT_model_irregular:: (κ $\implies$ o)  $\implies$  κ  $\implies$  o)
574         AOT_model_regular AOT_model_term_equiv>
575     unfolding AOT_model_irregular_κ_def by (metis (no_types, lifting) someI_ex)
576 qed
577 end
578
579 text<We define relations among individuals as proposition valued functions.
580   @{emph <Denoting>} unary relations (among @{typ κ}) will match the
581   urrelations introduced above.>
582 typedef 'a rel (<<_>>) = <UNIV::('a::AOT_IndividualTerm  $\implies$  o) set> ..
583 setup_lifting type_definition_rel
584
585 text<We will use the transformation specified above to "fix" the behaviour of
586   functions on irregular terms when defining @{text <λ>}-expressions.>
587 definition fix_irregular :: <('a::AOT_IndividualTerm  $\implies$  o)  $\implies$  ('a  $\implies$  o)> where
588   <fix_irregular  $\equiv$  λ φ x . if AOT_model_regular x
589     then φ x else AOT_model_irregular φ x>
590 lemma fix_irregular_denoting:
591   <AOT_model_denotes x  $\implies$  fix_irregular φ x = φ x>
592   by (meson AOT_model_irregular_nondenoting fix_irregular_def)
593 lemma fix_irregular_regular:
594   <AOT_model_regular x  $\implies$  fix_irregular φ x = φ x>
595   by (meson AOT_model_irregular_nondenoting fix_irregular_def)
596 lemma fix_irregular_irregular:
597   <¬AOT_model_regular x  $\implies$  fix_irregular φ x = AOT_model_irregular φ x>
598   by (simp add: fix_irregular_def)
599
600 text<Relations among individual terms are (potentially non-denoting) terms.
601   A relation denotes, if it agrees on all equivalent terms (i.e. terms sharing
602   urelements), is necessarily false on all non-denoting terms and is
603   well-behaved on irregular terms.>
604 instantiation rel :: (AOT_IndividualTerm) AOT_IncompleteTerm
605 begin
606 text<\linelabel{AOT_model_denotes_rel}>
607 lift_definition AOT_model_denotes_rel :: <<'a>  $\implies$  bool> is
608   <λ φ . (∀ x y . AOT_model_term_equiv x y  $\longrightarrow$  φ x = φ y)  $\wedge$ 
609     (∀ w x . AOT_model_valid_in w (φ x)  $\longrightarrow$  AOT_model_denotes x)  $\wedge$ 
610     (∀ x . ¬AOT_model_regular x  $\longrightarrow$  φ x = AOT_model_irregular φ x)> .
611 instance proof
612   have <AOT_model_irregular (fix_irregular φ) x = AOT_model_irregular φ x>
613     for φ and x :: 'a
614     by (rule AOT_model_irregular_eqI) (simp add: fix_irregular_def)
615   thus <∃ x :: <'a> . AOT_model_denotes x>
616     by (safe intro!: exI[where x=<Abs_rel (fix_irregular (λx. εo w . False))>])
617     (transfer; auto simp: AOT_model_proposition_choice_simp fix_irregular_def
618       AOT_model_irregular_equiv AOT_model_term_equiv_regular
619       AOT_model_irregular_false)
619 next

```

```

621 show <∃f :: <'a> . ¬AOT_model_denotes f>
622   by (rule exI[where x=<Abs_rel (λx. εo w . True)>]);
623     auto simp: AOT_model_denotes_rel.abs_eq AOT_model_nondenoting_ex
624           AOT_model_proposition_choice_simp)
625 qed
626 end
627
628 text<Auxiliary lemmata.>
629
630 lemma AOT_model_term_equiv_eps:
631   shows <AOT_model_term_equiv (Eps (AOT_model_term_equiv κ)) κ>
632     and <AOT_model_term_equiv κ (Eps (AOT_model_term_equiv κ))>
633     and <AOT_model_term_equiv κ κ' ⇒
634           (Eps (AOT_model_term_equiv κ)) = (Eps (AOT_model_term_equiv κ'))>
635   apply (metis AOT_model_term_equiv_part_equivp equivp_def someI_ex)
636   apply (metis AOT_model_term_equiv_part_equivp equivp_def someI_ex)
637   by (metis AOT_model_term_equiv_part_equivp equivp_def)
638
639 lemma AOT_model_denotes_Abs_rel_fix_irregularI:
640   assumes <∧ x y . AOT_model_term_equiv x y ⇒ φ x = φ y>
641     and <∧ w x . AOT_model_valid_in w (φ x) ⇒ AOT_model_denotes x>
642     shows <AOT_model_denotes (Abs_rel (fix_irregular φ))>
643 proof -
644   have <AOT_model_irregular φ x = AOT_model_irregular
645         (λx. if AOT_model_regular x then φ x else AOT_model_irregular φ x) x>
646     if <¬ AOT_model_regular x>
647     for x
648     by (rule AOT_model_irregular_eqI) auto
649   thus ?thesis
650   unfolding AOT_model_denotes_rel.rep_eq
651   using assms by (auto simp: AOT_model_irregular_false Abs_rel_inverse
652                             AOT_model_irregular_equiv fix_irregular_def
653                             AOT_model_term_equiv_regular)
654 qed
655
656 lemma AOT_model_term_equiv_rel_equiv:
657   assumes <AOT_model_denotes x>
658     and <AOT_model_denotes y>
659     shows <AOT_model_term_equiv x y = (∀ Π w . AOT_model_denotes Π ⇒
660           AOT_model_valid_in w (Rep_rel Π x) = AOT_model_valid_in w (Rep_rel Π y))>
661 proof
662   assume <AOT_model_term_equiv x y>
663   thus <∀ Π w . AOT_model_denotes Π ⇒ AOT_model_valid_in w (Rep_rel Π x) =
664         AOT_model_valid_in w (Rep_rel Π y)>
665     by (simp add: AOT_model_denotes_rel.rep_eq)
666 next
667   have 0: <(AOT_model_denotes x' ∧ AOT_model_term_equiv x' y) =
668           (AOT_model_denotes y' ∧ AOT_model_term_equiv y' y)>
669     if <AOT_model_term_equiv x' y'> for x' y'
670     by (metis that AOT_model_term_equiv_denotes AOT_model_term_equiv_part_equivp
671           equivp_def)
672   assume <∀ Π w . AOT_model_denotes Π ⇒ AOT_model_valid_in w (Rep_rel Π x) =
673           AOT_model_valid_in w (Rep_rel Π y)>
674   moreover have <AOT_model_denotes (Abs_rel (fix_irregular
675         (λ x . εo w . AOT_model_denotes x ∧ AOT_model_term_equiv x y)))>
676     (is "AOT_model_denotes ?r")
677     by (rule AOT_model_denotes_Abs_rel_fix_irregularI)
678     (auto simp: 0 AOT_model_denotes_rel.rep_eq Abs_rel_inverse fix_irregular_def
679           AOT_model_proposition_choice_simp AOT_model_irregular_false)
680   ultimately have <AOT_model_valid_in w (Rep_rel ?r x) =
681           AOT_model_valid_in w (Rep_rel ?r y)> for w
682     by blast
683   thus <AOT_model_term_equiv x y>

```



```

684     by (simp add: Abs_rel_inverse AOT_model_proposition_choice_simp
685           fix_irregular_denoting[OF assms(1)] AOT_model_term_equiv_part_equivp
686           fix_irregular_denoting[OF assms(2)] assms equivp_refl)
687 qed
688
689 text<Denoting relations among terms of type @{typ  $\kappa$ } correspond to urelations.>
690
691 definition rel_to_urrel :: << $\kappa$ >  $\Rightarrow$  urrel> where
692   <rel_to_urrel  $\equiv$   $\lambda$   $\Pi$  . Abs_urrel ( $\lambda$   $u$  . Rep_rel  $\Pi$  (SOME  $x$  .  $\kappa V$   $x = u$ )>
693 definition urrel_to_rel :: <urrel  $\Rightarrow$  < $\kappa$ >> where
694   <urrel_to_rel  $\equiv$   $\lambda$   $\varphi$  . Abs_rel ( $\lambda$   $x$  . Rep_urrel  $\varphi$  ( $\kappa V$   $x$ )>
695 definition AOT_rel_equiv :: <<'a::AOT_IndividualTerm>  $\Rightarrow$  <'a>  $\Rightarrow$  bool> where
696   <AOT_rel_equiv  $\equiv$   $\lambda$   $f$   $g$  . AOT_model_denotes  $f \wedge$  AOT_model_denotes  $g \wedge f = g$ >
697
698 lemma urrel_quotient3: <Quotient3 AOT_rel_equiv rel_to_urrel urrel_to_rel>
699 proof (rule Quotient3I)
700   have <( $\lambda$ u. Rep_urrel  $a$  ( $\kappa V$  (SOME  $x$  .  $\kappa V$   $x = u$ ))) = ( $\lambda$ u. Rep_urrel  $a$   $u$ )> for  $a$ 
701     by (rule ext) (metis (mono_tags, lifting)  $\kappa V$ _surj surj_f_inv_f verit_sko_ex')
702   thus <rel_to_urrel (urrel_to_rel  $a$ ) =  $a$ > for  $a$ 
703     by (simp add: Abs_rel_inverse rel_to_urrel_def urrel_to_rel_def
704               Rep_urrel_inverse)
705 next
706   show <AOT_rel_equiv (urrel_to_rel  $a$ ) (urrel_to_rel  $a$ )> for  $a$ 
707     unfolding AOT_rel_equiv_def urrel_to_rel_def
708     by transfer (simp add: AOT_model_regular_ $\kappa$ _def AOT_model_denotes_ $\kappa$ _def
709                       AOT_model_term_equiv_ $\kappa$ _def urrel_null_false)
710 next
711   {
712     fix  $a$ 
713     assume < $\forall w$   $x$  . AOT_model_valid_in  $w$  ( $a$   $x$ )  $\longrightarrow$   $\neg$  is_null $\kappa$   $x$ >
714     hence <( $\lambda$ u.  $a$  (SOME  $x$  .  $\kappa V$   $x = u$ ))  $\in$ 
715           { $\varphi$  .  $\forall x$   $w$  .  $\neg$  AOT_model_valid_in  $w$  ( $\varphi$  (null $v$   $x$ ))}>
716       by (simp; metis (mono_tags, lifting)  $\kappa$ .exhaust_disc  $\kappa V$ .sims  $v$ .disc(1,3,5)
717            $v$ .disc(6) is_ $\alpha\kappa$ _def is_ $\omega\kappa$ _def someI_ex)
718   } note 1 = this
719   {
720     fix  $r$   $s$  :: < $\kappa \Rightarrow o$ >
721     assume A: < $\forall x$   $y$  . AOT_model_term_equiv  $x$   $y$   $\longrightarrow$   $r$   $x = r$   $y$ >
722     assume < $\forall w$   $x$  . AOT_model_valid_in  $w$  ( $r$   $x$ )  $\longrightarrow$  AOT_model_denotes  $x$ >
723     hence 2: <( $\lambda$ u.  $r$  (SOME  $x$  .  $\kappa V$   $x = u$ ))  $\in$ 
724           { $\varphi$  .  $\forall x$   $w$  .  $\neg$  AOT_model_valid_in  $w$  ( $\varphi$  (null $v$   $x$ ))}>
725       using 1 AOT_model_denotes_ $\kappa$ _def by meson
726     assume B: < $\forall x$   $y$  . AOT_model_term_equiv  $x$   $y$   $\longrightarrow$   $s$   $x = s$   $y$ >
727     assume < $\forall w$   $x$  . AOT_model_valid_in  $w$  ( $s$   $x$ )  $\longrightarrow$  AOT_model_denotes  $x$ >
728     hence 3: <( $\lambda$ u.  $s$  (SOME  $x$  .  $\kappa V$   $x = u$ ))  $\in$ 
729           { $\varphi$  .  $\forall x$   $w$  .  $\neg$  AOT_model_valid_in  $w$  ( $\varphi$  (null $v$   $x$ ))}>
730       using 1 AOT_model_denotes_ $\kappa$ _def by meson
731     assume <Abs_urrel ( $\lambda$ u.  $r$  (SOME  $x$  .  $\kappa V$   $x = u$ )) =
732           Abs_urrel ( $\lambda$ u.  $s$  (SOME  $x$  .  $\kappa V$   $x = u$ )>
733     hence 4: < $r$  (SOME  $x$  .  $\kappa V$   $x = u$ ) =  $s$  (SOME  $x$  :: $\kappa$  .  $\kappa V$   $x = u$ )> for  $u$ 
734       unfolding Abs_urrel_inject[OF 2 3] by metis
735     have < $r$   $x = s$   $x$ > for  $x$ 
736       using 4[of < $\kappa V$   $x$ >]
737       by (metis (mono_tags, lifting) A B AOT_model_term_equiv_ $\kappa$ _def someI_ex)
738     hence < $r = s$ > by auto
739   }
740   thus <AOT_rel_equiv  $r$   $s =$  (AOT_rel_equiv  $r$   $r \wedge$  AOT_rel_equiv  $s$   $s \wedge$ 
741     rel_to_urrel  $r =$  rel_to_urrel  $s$ )> for  $r$   $s$ 
742     unfolding AOT_rel_equiv_def rel_to_urrel_def
743     by transfer auto
744 qed
745
746 lemma urrel_quotient:

```



```

747 <Quotient AOT_rel_equiv rel_to_urrel urrel_to_rel
748   (<\x y. AOT_rel_equiv x x ^ rel_to_urrel x = y>)
749 using Quotient3_to_Quotient[OF urrel_quotient3] by auto
750
751 text<Unary individual terms are always regular and equipped with encoding and
752   concreteness. The specification of the type class anticipates the required
753   properties for deriving the axiom system.>
754 class AOT_UnaryIndividualTerm =
755   fixes AOT_model_enc :: '<a => <'a::AOT_IndividualTerm> => bool>
756   and AOT_model_concrete :: <w => 'a => bool>
757   assumes AOT_model_unary_regular:
758     <AOT_model_regular x> - <All unary individual terms are regular.>
759   and AOT_model_enc_relid:
760     <AOT_model_denotes F ==>
761     AOT_model_denotes G ==>
762     (<\ x . AOT_model_enc x F <-> AOT_model_enc x G>
763      ==> F = G>
764   and AOT_model_A_objects:
765     <\x . AOT_model_denotes x ^
766     (<\w. ~ AOT_model_concrete w x> ^
767     (<\F. AOT_model_denotes F ==> AOT_model_enc x F = \phi F>)>
768   and AOT_model_contingent:
769     <\x w. AOT_model_concrete w x ^ ~ AOT_model_concrete w_0 x>
770   and AOT_model_nocoder:
771     <AOT_model_concrete w x ==> ~AOT_model_enc x F>
772   and AOT_model_concrete_equiv:
773     <AOT_model_term_equiv x y ==>
774     AOT_model_concrete w x = AOT_model_concrete w y>
775   and AOT_model_concrete_denotes:
776     <AOT_model_concrete w x ==> AOT_model_denotes x>
777   - <The following are properties that will only hold in the extended models.>
778   and AOT_model_enc_indistinguishable_all:
779     <AOT_ExtendedModel ==>
780     AOT_model_denotes a ==> ~(<\ w . AOT_model_concrete w a> ==>
781     AOT_model_denotes b ==> ~(<\ w . AOT_model_concrete w b> ==>
782     AOT_model_denotes II ==>
783     (<\ II' . AOT_model_denotes II' ==>
784     (<\ v . AOT_model_valid_in v (Rep_rel II' a) =
785     AOT_model_valid_in v (Rep_rel II' b)>)) ==>
786     (<\ II' . AOT_model_denotes II' ==>
787     (<\ v x . \ w . AOT_model_concrete w x ==>
788     AOT_model_valid_in v (Rep_rel II' x) =
789     AOT_model_valid_in v (Rep_rel II x)>)) ==>
790     AOT_model_enc a II') ==>
791     (<\ II' . AOT_model_denotes II' ==>
792     (<\ v x . \ w . AOT_model_concrete w x ==>
793     AOT_model_valid_in v (Rep_rel II' x) =
794     AOT_model_valid_in v (Rep_rel II x)>)) ==>
795     AOT_model_enc b II')>
796   and AOT_model_enc_indistinguishable_ex:
797     <AOT_ExtendedModel ==>
798     AOT_model_denotes a ==> ~(<\ w . AOT_model_concrete w a> ==>
799     AOT_model_denotes b ==> ~(<\ w . AOT_model_concrete w b> ==>
800     AOT_model_denotes II ==>
801     (<\ II' . AOT_model_denotes II' ==>
802     (<\ v . AOT_model_valid_in v (Rep_rel II' a) =
803     AOT_model_valid_in v (Rep_rel II' b)>)) ==>
804     (<\ II' . AOT_model_denotes II' ^ AOT_model_enc a II' ^
805     (<\ v x . (<\ w . AOT_model_concrete w x> ==>
806     AOT_model_valid_in v (Rep_rel II' x) =
807     AOT_model_valid_in v (Rep_rel II x)>)) ==>
808     (<\ II' . AOT_model_denotes II' ^ AOT_model_enc b II' ^
809     (<\ v x . (<\ w . AOT_model_concrete w x> ==>

```

```

810         AOT_model_valid_in v (Rep_rel II' x) =
811         AOT_model_valid_in v (Rep_rel II x)))>
812
813 text<Instantiate the class of unary individual terms for our concrete type of
814 individual terms @{\typ \kappa}.>
815 instantiation \kappa :: AOT_UnaryIndividualTerm
816 begin
817
818 definition AOT_model_enc_\kappa :: <\kappa \Rightarrow <\kappa> \Rightarrow bool> where
819   <AOT_model_enc_\kappa \equiv \lambda x F .
820     case x of \alpha\kappa a \Rightarrow AOT_model_denotes F \wedge rel_to_urrel F \in a
821       | _ \Rightarrow False>
822 primrec AOT_model_concrete_\kappa :: <w \Rightarrow \kappa \Rightarrow bool> where
823   <AOT_model_concrete_\kappa w (\omega\kappa x) = AOT_model_concrete_\omega w x>
824 | <AOT_model_concrete_\kappa w (\alpha\kappa x) = False>
825 | <AOT_model_concrete_\kappa w (null\kappa x) = False>
826
827 lemma AOT_meta_A_objects_\kappa:
828   <\exists x :: \kappa. AOT_model_denotes x \wedge
829     (\forall w. \neg AOT_model_concrete w x) \wedge
830     (\forall F. AOT_model_denotes F \longrightarrow AOT_model_enc x F = \varphi F)> for \varphi
831 apply (rule exI[where x=<\alpha\kappa {f . \varphi (urrel_to_rel f)}>])
832 apply (simp add: AOT_model_enc_\kappa_def AOT_model_denotes_\kappa_def)
833 by (metis (no_types, lifting) AOT_rel_equiv_def urrel_quotient
834     Quotient_rep_abs_fold_unmap)
835
836 instance proof
837   show <AOT_model_regular x> for x :: \kappa
838     by (simp add: AOT_model_regular_\kappa_def)
839 next
840   fix F G :: <<\kappa>>
841   assume <AOT_model_denotes F>
842   moreover assume <AOT_model_denotes G>
843   moreover assume <\bigwedge x. AOT_model_enc x F = AOT_model_enc x G>
844   moreover obtain x where <\forall G. AOT_model_denotes G \longrightarrow AOT_model_enc x G = (F = G)>
845     using AOT_meta_A_objects_\kappa by blast
846   ultimately show <F = G> by blast
847 next
848   show <\exists x :: \kappa. AOT_model_denotes x \wedge
849     (\forall w. \neg AOT_model_concrete w x) \wedge
850     (\forall F. AOT_model_denotes F \longrightarrow AOT_model_enc x F = \varphi F)> for \varphi
851     using AOT_meta_A_objects_\kappa .
852 next
853   show <\exists (x :: \kappa) w. AOT_model_concrete w x \wedge \neg AOT_model_concrete w_0 x>
854     using AOT_model_concrete_\kappa.simps(1) AOT_model_contingent_object by blast
855 next
856   show <AOT_model_concrete w x \Longrightarrow \neg AOT_model_enc x F> for w and x :: \kappa and F
857     by (metis AOT_model_concrete_\kappa.simps(2) AOT_model_enc_\kappa_def \kappa.case_eq_if
858         \kappa.collapse(2))
859 next
860   show <AOT_model_concrete w x = AOT_model_concrete w y>
861     if <AOT_model_term_equiv x y>
862     for x y :: \kappa and w
863     using that by (induct x; induct y; auto simp: AOT_model_term_equiv_\kappa_def)
864 next
865   show <AOT_model_concrete w x \Longrightarrow AOT_model_denotes x> for w and x :: \kappa
866     by (metis AOT_model_concrete_\kappa.simps(3) AOT_model_denotes_\kappa_def \kappa.collapse(3))
867 (* Extended models only *)
868 next
869   fix \kappa \kappa' :: \kappa and II II' :: <<\kappa>> and w :: w
870   assume ext: <AOT_ExtendedModel>
871   assume <AOT_model_denotes \kappa>
872   moreover assume <\nexists w. AOT_model_concrete w \kappa>

```

```

873 ultimately obtain a where a_def: < $\alpha\kappa$  a =  $\kappa$ >
874   by (metis AOT_model_ω_concrete_in_some_world AOT_model_concrete_κ.simps(1)
875         AOT_model_denotes_κ_def κ.discI(3) κ.exhaust_sel)
876 assume <AOT_model_denotes  $\kappa'$ >
877 moreover assume < $\nexists w$ . AOT_model_concrete w  $\kappa'$ >
878 ultimately obtain b where b_def: < $\alpha\kappa$  b =  $\kappa'$ >
879   by (metis AOT_model_ω_concrete_in_some_world AOT_model_concrete_κ.simps(1)
880         AOT_model_denotes_κ_def κ.discI(3) κ.exhaust_sel)
881 assume <AOT_model_denotes  $\Pi'$   $\implies$  AOT_model_valid_in w (Rep_rel  $\Pi'$   $\kappa$ ) =
882         AOT_model_valid_in w (Rep_rel  $\Pi'$   $\kappa'$ )> for  $\Pi'$  w
883 hence <AOT_model_valid_in w (Rep_urrel r ( $\kappa$  v  $\kappa$ )) =
884         AOT_model_valid_in w (Rep_urrel r ( $\kappa$  v  $\kappa'$ ))> for r
885   by (metis AOT_rel_equiv_def Abs_rel_inverse Quotient3_rel_rep
886         iso_tuple_UNIV_I urrel_quotient3 urrel_to_rel_def)
887 hence <let r = (Abs_urrel ( $\lambda$  u .  $\varepsilon_o$  w . u =  $\kappa$  v  $\kappa$ )) in
888         AOT_model_valid_in w (Rep_urrel r ( $\kappa$  v  $\kappa$ )) =
889         AOT_model_valid_in w (Rep_urrel r ( $\kappa$  v  $\kappa'$ ))>
890   by presburger
891 hence  $\alpha\sigma$ _eq: < $\alpha\sigma$  a =  $\alpha\sigma$  b>
892   unfolding Let_def
893   apply (subst (asm) (1 2) Abs_urrel_inverse)
894   using AOT_model_proposition_choice_simp a_def b_def by force+
895 assume  $\Pi$ _den: <AOT_model_denotes  $\Pi$ >
896 have < $\exists r$  .  $\forall x$  . Rep_rel  $\Pi$  ( $\omega\kappa$  x) = Rep_urrel r ( $\omega v$  x)>
897   apply (rule exI[where x=<rel_to_urrel  $\Pi$ >])
898   apply auto
899   unfolding rel_to_urrel_def
900   apply (subst Abs_urrel_inverse)
901   apply auto
902   apply (metis (mono_tags, lifting) AOT_model_denotes_κ_def
903         AOT_model_denotes_rel.rep_eq κ.exhaust_disc  $\kappa v$ .simps(1,2,3)
904         <AOT_model_denotes  $\Pi$ > v.disc(8,9) v.distinct(3)
905         is_ακ_def is_ωκ_def verit_sko_ex')
906   by (metis (mono_tags, lifting) AOT_model_denotes_rel.rep_eq
907         AOT_model_term_equiv_κ_def  $\kappa v$ .simps(1)  $\Pi$ _den verit_sko_ex')
908 then obtain r where r_prop: <Rep_rel  $\Pi$  ( $\omega\kappa$  x) = Rep_urrel r ( $\omega v$  x)> for x
909   by blast
910 assume <AOT_model_denotes  $\Pi'$   $\implies$ 
911   ( $\bigwedge v$  x.  $\exists w$ . AOT_model_concrete w x  $\implies$ 
912     AOT_model_valid_in v (Rep_rel  $\Pi'$  x) =
913     AOT_model_valid_in v (Rep_rel  $\Pi$  x))  $\implies$  AOT_model_enc  $\kappa$   $\Pi'$ > for  $\Pi'$ 
914 hence <AOT_model_denotes  $\Pi'$   $\implies$ 
915   ( $\bigwedge v$  x. AOT_model_valid_in v (Rep_rel  $\Pi'$  ( $\omega\kappa$  x)) =
916     AOT_model_valid_in v (Rep_rel  $\Pi$  ( $\omega\kappa$  x)))  $\implies$  AOT_model_enc  $\kappa$   $\Pi'$ > for  $\Pi'$ 
917   by (metis AOT_model_concrete_κ.simps(2) AOT_model_concrete_κ.simps(3)
918         κ.exhaust_disc is_ακ_def is_ωκ_def is_nullκ_def)
919 hence <( $\bigwedge v$  x. AOT_model_valid_in v (Rep_urrel r ( $\omega v$  x)) =
920         AOT_model_valid_in v (Rep_rel  $\Pi$  ( $\omega\kappa$  x)))  $\implies$  r  $\in$  a> for r
921   unfolding a_def[symmetric] AOT_model_enc_κ_def apply simp
922   by (smt (verit, best) AOT_rel_equiv_def Abs_rel_inverse Quotient3_def
923          $\kappa v$ .simps(1) iso_tuple_UNIV_I urrel_quotient3 urrel_to_rel_def)
924 hence <( $\bigwedge v$  x. AOT_model_valid_in v (Rep_urrel r' ( $\omega v$  x)) =
925         AOT_model_valid_in v (Rep_urrel r ( $\omega v$  x)))  $\implies$  r'  $\in$  a> for r'
926   unfolding r_prop.
927 hence < $\bigwedge s$ . urrel_to_ωrel s = urrel_to_ωrel r  $\implies$  s  $\in$  a>
928   by (metis urrel_to_ωrel_def)
929 hence 0: < $\bigwedge s$ . urrel_to_ωrel s = urrel_to_ωrel r  $\implies$  s  $\in$  b>
930   using  $\alpha\sigma$ _eq_ord_exts_all  $\alpha\sigma$ _eq ext  $\alpha\sigma$ _α $\sigma'$  by blast
931
932 assume  $\Pi'$ _den: <AOT_model_denotes  $\Pi'$ >
933 assume < $\exists w$ . AOT_model_concrete w x  $\implies$  AOT_model_valid_in v (Rep_rel  $\Pi'$  x) =
934         AOT_model_valid_in v (Rep_rel  $\Pi$  x)> for v x
935 hence <AOT_model_valid_in v (Rep_rel  $\Pi'$  ( $\omega\kappa$  x)) =

```

```

936     AOT_model_valid_in v (Rep_rel Π (ωκ x))> for v x
937     using AOT_model_ω_concrete_in_some_world AOT_model_concrete_κ.simps(1)
938     by presburger
939 hence <AOT_model_valid_in v (Rep_urrel (rel_to_urrel Π') (ωv x)) =
940     AOT_model_valid_in v (Rep_urrel r (ωv x))> for v x
941     by (smt (verit, best) AOT_rel_equiv_def Abs_rel_inverse Quotient3_def
942     κv.simps(1) iso_tuple_UNIV_I r_prop urrel_quotient3 urrel_to_rel_def Π'_den)
943 hence <urrel_to_ωrel (rel_to_urrel Π') = urrel_to_ωrel r>
944     by (metis (full_types) AOT_urrel_ωequiv_def Quotient3_def urrel_ωrel_quot)
945 hence <rel_to_urrel Π' ∈ b> using 0 by blast
946 thus <AOT_model_enc κ' Π'>
947     unfolding b_def[symmetric] AOT_model_enc_κ_def by (auto simp: Π'_den)
948 next
949 fix κ κ' :: κ and Π Π' :: <<κ>> and w :: w
950 assume ext: <AOT_ExtendedModel>
951 assume <AOT_model_denotes κ>
952 moreover assume <∃w. AOT_model_concrete w κ>
953 ultimately obtain a where a_def: <ακ a = κ>
954     by (metis AOT_model_ω_concrete_in_some_world AOT_model_concrete_κ.simps(1)
955     AOT_model_denotes_κ_def κ.discI(3) κ.exhaust_sel)
956 assume <AOT_model_denotes κ'>
957 moreover assume <∃w. AOT_model_concrete w κ'>
958 ultimately obtain b where b_def: <ακ b = κ'>
959     by (metis AOT_model_ω_concrete_in_some_world AOT_model_concrete_κ.simps(1)
960     AOT_model_denotes_κ_def κ.discI(3) κ.exhaust_sel)
961 assume <AOT_model_denotes Π' ⇒ AOT_model_valid_in w (Rep_rel Π' κ) =
962     AOT_model_valid_in w (Rep_rel Π' κ')> for Π' w
963 hence <AOT_model_valid_in w (Rep_urrel r (κv κ)) =
964     AOT_model_valid_in w (Rep_urrel r (κv κ'))> for r
965     by (metis AOT_rel_equiv_def Abs_rel_inverse Quotient3_rel_rep
966     iso_tuple_UNIV_I urrel_quotient3 urrel_to_rel_def)
967 hence <let r = (Abs_urrel (λ u . εo w . u = κv κ)) in
968     AOT_model_valid_in w (Rep_urrel r (κv κ)) =
969     AOT_model_valid_in w (Rep_urrel r (κv κ'))>
970     by presburger
971 hence ασ_eq: <ασ a = ασ b>
972     unfolding Let_def
973     apply (subst (asm) (1 2) Abs_urrel_inverse)
974     using AOT_model_proposition_choice_simp a_def b_def by force+
975 assume Π_den: <AOT_model_denotes Π>
976 have <∃r . ∀ x . Rep_rel Π (ωκ x) = Rep_urrel r (ωv x)>
977     apply (rule exI[where x=<rel_to_urrel Π>])
978     apply auto
979     unfolding rel_to_urrel_def
980     apply (subst Abs_urrel_inverse)
981     apply auto
982     apply (metis (mono_tags, lifting) AOT_model_denotes_κ_def
983     AOT_model_denotes_rel.rep_eq κ.exhaust_disc κv.simps(1,2,3)
984     <AOT_model_denotes Π> v.disc(8) v.disc(9) v.distinct(3)
985     is_ακ_def is_ωκ_def verit_sko_ex')
986     by (metis (mono_tags, lifting) AOT_model_denotes_rel.rep_eq
987     AOT_model_term_equiv_κ_def κv.simps(1) Π_den verit_sko_ex')
988 then obtain r where r_prop: <Rep_rel Π (ωκ x) = Rep_urrel r (ωv x)> for x
989     by blast
990
991 assume <∃Π'. AOT_model_denotes Π' ∧
992     AOT_model_enc κ Π' ∧
993     (∀v x. (∃w. AOT_model_concrete w x) → AOT_model_valid_in v (Rep_rel Π' x) =
994     AOT_model_valid_in v (Rep_rel Π x))>
995 then obtain Π' where
996     Π'_den: <AOT_model_denotes Π'> and
997     κ_enc_Π': <AOT_model_enc κ Π'> and
998     Π'_prop: <∃w. AOT_model_concrete w x ⇒

```

```

999             AOT_model_valid_in v (Rep_rel Π' x) =
1000             AOT_model_valid_in v (Rep_rel Π x)> for v x
1001   by blast
1002   have <AOT_model_valid_in v (Rep_rel Π' (ωκ x)) =
1003         AOT_model_valid_in v (Rep_rel Π (ωκ x))> for x v
1004   by (simp add: AOT_model_ω_concrete_in_some_world Π'_prop)
1005   hence 0: <AOT_urrel_ωequiv (rel_to_urrel Π') (rel_to_urrel Π)>
1006     unfolding AOT_urrel_ωequiv_def
1007   by (smt (verit) AOT_rel_equiv_def Abs_rel_inverse Quotient3_def
1008         κv.simps(1) iso_tuple_UNIV_I urrel_quotient3 urrel_to_rel_def
1009         Π_den Π'_den)
1010   have <rel_to_urrel Π' ∈ a>
1011   and <urrel_to_ωrel (rel_to_urrel Π') = urrel_to_ωrel (rel_to_urrel Π)>
1012   apply (metis AOT_model_enc_κ_def κ.simps(11) κ_enc_Π' a_def)
1013   by (metis Quotient3_rel 0 urrel_ωrel_quot)
1014   hence <∃s. s ∈ b ∧ urrel_to_ωrel s = urrel_to_ωrel (rel_to_urrel Π)>
1015     using ασ_eq_ord_exts_ex ασ_eq_ext ασ_ασ' by blast
1016   then obtain s where
1017     s_prop: <s ∈ b ∧ urrel_to_ωrel s = urrel_to_ωrel (rel_to_urrel Π)>
1018   by blast
1019   then obtain Π" where
1020     Π"_prop: <rel_to_urrel Π" = s> and Π"_den: <AOT_model_denotes Π">
1021   by (metis AOT_rel_equiv_def Quotient3_def urrel_quotient3)
1022   moreover have <AOT_model_enc κ' Π">
1023   by (metis AOT_model_enc_κ_def Π"_den Π"_prop κ.simps(11) b_def s_prop)
1024   moreover have <AOT_model_valid_in v (Rep_rel Π" x) =
1025         AOT_model_valid_in v (Rep_rel Π x)>
1026     if <∃w. AOT_model_concrete w x> for v x
1027   proof(insert that)
1028     assume <∃w. AOT_model_concrete w x>
1029     then obtain u where x_def: <x = ωκ u>
1030     by (metis AOT_model_concrete_κ.simps(2,3) κ.exhaust)
1031     show <AOT_model_valid_in v (Rep_rel Π" x) =
1032           AOT_model_valid_in v (Rep_rel Π x)>
1033       unfolding x_def
1034     by (smt (verit, best) AOT_rel_equiv_def Abs_rel_inverse Quotient3_def
1035           Π"_den Π"_prop Π_den κv.simps(1) iso_tuple_UNIV_I s_prop
1036           urrel_quotient3 urrel_to_ωrel_def urrel_to_rel_def)
1037   qed
1038   ultimately show <∃Π'. AOT_model_denotes Π' ∧ AOT_model_enc κ' Π' ∧
1039         (∀v x. (∃w. AOT_model_concrete w x) → AOT_model_valid_in v (Rep_rel Π' x) =
1040               AOT_model_valid_in v (Rep_rel Π x))>
1041     apply (safe intro!: exI[where x=Π'])
1042   by auto
1043   qed
1044   end
1045
1046   text<Products of unary individual terms and individual terms are individual terms.
1047         A tuple is regular, if at most one element does not denote. I.e. a pair is
1048         regular, if the first (unary) element denotes and the second is regular (i.e.
1049         at most one of its recursive tuple elements does not denote), or the first does
1050         not denote, but the second denotes (i.e. all its recursive tuple elements
1051         denote).>
1052   instantiation prod :: (AOT_UnaryIndividualTerm, AOT_IndividualTerm) AOT_IndividualTerm
1053   begin
1054   definition AOT_model_regular_prod :: <'a×'b ⇒ bool> where
1055     <AOT_model_regular_prod ≡ λ (x,y) . AOT_model_denotes x ∧ AOT_model_regular y ∨
1056           ¬AOT_model_denotes x ∧ AOT_model_denotes y>
1057   definition AOT_model_term_equiv_prod :: <'a×'b ⇒ 'a×'b ⇒ bool> where
1058     <AOT_model_term_equiv_prod ≡ λ (x1,y1) (x2,y2) .
1059           AOT_model_term_equiv x1 x2 ∧ AOT_model_term_equiv y1 y2>
1060   function AOT_model_irregular_prod :: <('a×'b ⇒ o) ⇒ 'a×'b ⇒ o> where
1061     AOT_model_irregular_prod2: <AOT_model_denotes x ⇒

```

```

1062   AOT_model_irregular  $\varphi$  (x,y) =
1063   AOT_model_irregular ( $\lambda y. \varphi$  (SOME x' . AOT_model_term_equiv x x', y)) y>
1064 | AOT_model_irregular_proj1:  $\langle \neg$ AOT_model_denotes x  $\wedge$  AOT_model_denotes y  $\implies$ 
1065   AOT_model_irregular  $\varphi$  (x,y) =
1066   AOT_model_irregular ( $\lambda x. \varphi$  (x, SOME y' . AOT_model_term_equiv y y')) x>
1067 | AOT_model_irregular_prod_generic:  $\langle \neg$ AOT_model_denotes x  $\wedge$   $\neg$ AOT_model_denotes y  $\implies$ 
1068   AOT_model_irregular  $\varphi$  (x,y) =
1069   (SOME  $\Phi$  . AOT_model_irregular_spec  $\Phi$  AOT_model_regular AOT_model_term_equiv)
1070    $\varphi$  (x,y)>
1071   by auto blast
1072 termination using "termination" by blast
1073
1074 instance proof
1075   obtain x :: 'a and y :: 'b where
1076      $\langle \neg$ AOT_model_denotes x  $\rangle$  and  $\langle \neg$ AOT_model_denotes y  $\rangle$ 
1077     by (meson AOT_model_nondenoting_ex AOT_model_denoting_ex)
1078   thus  $\langle \exists x::'a \times 'b. \neg$ AOT_model_denotes x  $\rangle$ 
1079     by (auto simp: AOT_model_denotes_prod_def AOT_model_regular_prod_def)
1080 next
1081   show  $\langle$ equivp (AOT_model_term_equiv :: 'a  $\times$  'b  $\implies$  'a  $\times$  'b  $\implies$  bool) $\rangle$ 
1082     by (rule equivpI; rule reflpI sympI transpI;
1083         simp add: AOT_model_term_equiv_prod_def AOT_model_term_equiv_part_equivp
1084             equivp_reflprod_case_eq_if case_prod_unfold equivp_symp)
1085     (metis equivp_transp[OF AOT_model_term_equiv_part_equivp])
1086 next
1087   show  $\langle \neg$ AOT_model_regular x  $\implies$   $\neg$  AOT_model_denotes x  $\rangle$  for x :: 'a  $\times$  'b
1088     by (metis (mono_tags, lifting) AOT_model_denotes_prod_def case_prod_unfold
1089         AOT_model_irregular_nondenoting AOT_model_regular_prod_def)
1090 next
1091   fix x y :: 'a  $\times$  'b
1092   show  $\langle$ AOT_model_term_equiv x y  $\implies$  AOT_model_denotes x = AOT_model_denotes y  $\rangle$ 
1093     by (metis (mono_tags, lifting) AOT_model_denotes_prod_def case_prod_beta
1094         AOT_model_term_equiv_denotes AOT_model_term_equiv_prod_def )
1095 next
1096   fix x y :: 'a  $\times$  'b
1097   show  $\langle$ AOT_model_term_equiv x y  $\implies$  AOT_model_regular x = AOT_model_regular y  $\rangle$ 
1098     by (induct x; induct y;
1099         simp add: AOT_model_term_equiv_prod_def AOT_model_regular_prod_def)
1100     (meson AOT_model_term_equiv_denotes AOT_model_term_equiv_regular)
1101 next
1102   interpret sp: AOT_model_irregular_spec  $\langle \lambda \varphi$  (x::'a  $\times$  'b) .  $\varepsilon_0$  w . False  $\rangle$ 
1103     AOT_model_regular AOT_model_term_equiv
1104   by (simp add: AOT_model_irregular_spec_def AOT_model_proposition_choice_simp)
1105 have ex_spec:  $\langle \exists \varphi :: ('a \times 'b \implies o) \implies 'a \times 'b \implies o .$ 
1106   AOT_model_irregular_spec  $\varphi$  AOT_model_regular AOT_model_term_equiv  $\rangle$ 
1107   using sp.AOT_model_irregular_spec_axioms by blast
1108 have some_spec:  $\langle$ AOT_model_irregular_spec
1109   (SOME  $\varphi :: ('a \times 'b \implies o) \implies 'a \times 'b \implies o .$ 
1110   AOT_model_irregular_spec  $\varphi$  AOT_model_regular AOT_model_term_equiv)
1111   AOT_model_regular AOT_model_term_equiv  $\rangle$ 
1112   using someI_ex[OF ex_spec] by argo
1113 interpret sp_some: AOT_model_irregular_spec
1114    $\langle$ SOME  $\varphi :: ('a \times 'b \implies o) \implies 'a \times 'b \implies o .$ 
1115   AOT_model_irregular_spec  $\varphi$  AOT_model_regular AOT_model_term_equiv  $\rangle$ 
1116   AOT_model_regular AOT_model_term_equiv
1117   using some_spec by blast
1118 show  $\langle$ AOT_model_irregular_spec (AOT_model_irregular :: ('a  $\times$  'b  $\implies$  o)  $\implies$  'a  $\times$  'b  $\implies$  o)
1119   AOT_model_regular AOT_model_term_equiv  $\rangle$ 
1120 proof
1121   have  $\langle \neg$ AOT_model_valid_in w (AOT_model_irregular  $\varphi$  (a, b))  $\rangle$ 
1122     for w  $\varphi$  and a :: 'a and b :: 'b
1123     by (induct arbitrary:  $\varphi$  rule: AOT_model_irregular_prod.induct)
1124     (auto simp: AOT_model_irregular_false sp_some.AOT_model_irregular_false)

```



```

1125   thus "¬AOT_model_valid_in w (AOT_model_irregular  $\varphi$  x)" for w  $\varphi$  and x :: <'a×'b>
1126   by (induct x)
1127 next
1128   {
1129     fix x1 y1 :: 'a and x2 y2 :: 'b and  $\varphi$  :: <'a×'b⇒o>
1130     assume x1y1_equiv: <AOT_model_term_equiv x1 y1>
1131     moreover assume x2y2_equiv: <AOT_model_term_equiv x2 y2>
1132     ultimately have xy_equiv: <AOT_model_term_equiv (x1,x2) (y1,y2)>
1133     by (simp add: AOT_model_term_equiv_prod_def)
1134     {
1135       assume <AOT_model_denotes x1>
1136       moreover hence <AOT_model_denotes y1>
1137         using AOT_model_term_equiv_denotes AOT_model_term_equiv_regular
1138         x1y1_equiv x2y2_equiv by blast
1139       ultimately have <AOT_model_irregular  $\varphi$  (x1,x2) =
1140         AOT_model_irregular  $\varphi$  (y1,y2)>
1141       using AOT_model_irregular_equiv AOT_model_term_equiv_eps(3)
1142       x1y1_equiv x2y2_equiv by fastforce
1143     }
1144     moreover {
1145       assume < AOT_model_denotes x1 ∧ AOT_model_denotes x2>
1146       moreover hence < AOT_model_denotes y1 ∧ AOT_model_denotes y2>
1147         by (meson AOT_model_term_equiv_denotes x1y1_equiv x2y2_equiv)
1148       ultimately have <AOT_model_irregular  $\varphi$  (x1,x2) =
1149         AOT_model_irregular  $\varphi$  (y1,y2)>
1150       using AOT_model_irregular_equiv AOT_model_term_equiv_eps(3)
1151       x1y1_equiv x2y2_equiv by fastforce
1152     }
1153     moreover {
1154       assume denotes_x: <(¬AOT_model_denotes x1 ∧ ¬AOT_model_denotes x2)>
1155       hence denotes_y: <(¬AOT_model_denotes y1 ∧ ¬AOT_model_denotes y2)>
1156       by (meson AOT_model_term_equiv_denotes AOT_model_term_equiv_regular
1157         x1y1_equiv x2y2_equiv)
1158       have eps_eq: <Eps (AOT_model_term_equiv x1) = Eps (AOT_model_term_equiv y1)>
1159       by (simp add: AOT_model_term_equiv_eps(3) x1y1_equiv)
1160       have <AOT_model_irregular  $\varphi$  (x1,x2) = AOT_model_irregular  $\varphi$  (y1,y2)>
1161       using denotes_x denotes_y
1162       using sp_some.AOT_model_irregular_equiv xy_equiv by auto
1163     }
1164     moreover {
1165       assume denotes_x: <¬AOT_model_denotes x1 ∧ AOT_model_denotes x2>
1166       hence denotes_y: <¬AOT_model_denotes y1 ∧ AOT_model_denotes y2>
1167       by (meson AOT_model_term_equiv_denotes x1y1_equiv x2y2_equiv)
1168       have eps_eq: <Eps (AOT_model_term_equiv x2) = Eps (AOT_model_term_equiv y2)>
1169       by (simp add: AOT_model_term_equiv_eps(3) x2y2_equiv)
1170       have <AOT_model_irregular  $\varphi$  (x1,x2) = AOT_model_irregular  $\varphi$  (y1,y2)>
1171       using denotes_x denotes_y
1172       using AOT_model_irregular_nondenoting calculation(2) by blast
1173     }
1174     ultimately have <AOT_model_irregular  $\varphi$  (x1,x2) = AOT_model_irregular  $\varphi$  (y1,y2)>
1175     using AOT_model_term_equiv_denotes AOT_model_term_equiv_regular
1176     sp_some.AOT_model_irregular_equiv x1y1_equiv x2y2_equiv xy_equiv
1177     by blast
1178   } note 0 = this
1179   show <AOT_model_term_equiv x y ⇒
1180     AOT_model_irregular  $\varphi$  x = AOT_model_irregular  $\varphi$  y>
1181   for x y :: <'a×'b> and  $\varphi$ 
1182   by (induct x; induct y; simp add: AOT_model_term_equiv_prod_def 0)
1183 next
1184 fix  $\varphi$   $\psi$  :: <'a×'b ⇒ o>
1185 assume <AOT_model_regular x ⇒  $\varphi$  x =  $\psi$  x> for x
1186 hence < $\varphi$  (x, y) =  $\psi$  (x, y)>
1187 if <AOT_model_denotes x ∧ AOT_model_regular y ∨

```

```

1188     ¬AOT_model_denotes x ∧ AOT_model_denotes y > for x y
1189     using that unfolding AOT_model_regular_prod_def by simp
1190 hence <AOT_model_irregular φ (x,y) = AOT_model_irregular ψ (x,y)>
1191     for x :: 'a and y :: 'b
1192 proof (induct arbitrary: ψ φ rule: AOT_model_irregular_prod.induct)
1193   case (1 x y φ)
1194   thus ?case
1195     apply simp
1196     by (meson AOT_model_irregular_eqI AOT_model_irregular_nondenoting
1197         AOT_model_term_equiv_denotes AOT_model_term_equiv_eps(1))
1198 next
1199   case (2 x y φ)
1200   thus ?case
1201     apply simp
1202     by (meson AOT_model_irregular_nondenoting AOT_model_term_equiv_denotes
1203         AOT_model_term_equiv_eps(1))
1204 next
1205   case (3 x y φ)
1206   thus ?case
1207     apply simp
1208     by (metis (mono_tags, lifting) AOT_model_regular_prod_def case_prod_conv
1209         sp_some.AOT_model_irregular_eqI surj_pair)
1210 qed
1211 thus <AOT_model_irregular φ x = AOT_model_irregular ψ x> for x :: <'a×'b>
1212   by (metis surjective_pairing)
1213 qed
1214 qed
1215 end
1216
1217 text<Introduction rules for term equivalence on tuple terms.>
1218 lemma AOT_meta_prod_equivI:
1219   shows "∧ (a::'a::AOT_UnaryIndividualTerm) x (y :: 'b::AOT_IndividualTerm) .
1220         AOT_model_term_equiv x y ⇒ AOT_model_term_equiv (a,x) (a,y)"
1221   and "∧ (x::'a::AOT_UnaryIndividualTerm) y (b :: 'b::AOT_IndividualTerm) .
1222         AOT_model_term_equiv x y ⇒ AOT_model_term_equiv (x,b) (y,b)"
1223   unfolding AOT_model_term_equiv_prod_def
1224   by (simp add: AOT_model_term_equiv_part_equivp equivp_refl)+
1225
1226 text<The type of propositions are trivial instances of terms.>
1227
1228 instantiation o :: AOT_Term
1229 begin
1230 definition AOT_model_denotes_o :: <o ⇒ bool> where
1231   <AOT_model_denotes_o ≡ λ_. True>
1232 instance proof
1233   show <∃x::o. AOT_model_denotes x>
1234     by (simp add: AOT_model_denotes_o_def)
1235 qed
1236 end
1237
1238 text<AOT's variables are modelled by restricting the type of terms to those terms
1239   that denote.>
1240 typedef 'a AOT_var = <{ x :: 'a::AOT_Term . AOT_model_denotes x }>
1241 morphisms AOT_term_of_var AOT_var_of_term
1242 by (simp add: AOT_model_denoting_ex)
1243
1244 text<Simplify automatically generated theorems and rules.>
1245 declare AOT_var_of_term_induct[induct del]
1246         AOT_var_of_term_cases[cases del]
1247         AOT_term_of_var_induct[induct del]
1248         AOT_term_of_var_cases[cases del]
1249 lemmas AOT_var_of_term_inverse = AOT_var_of_term_inverse[simplified]
1250 and AOT_var_of_term_inject = AOT_var_of_term_inject[simplified]

```



```

1251 and AOT_var_of_term_induct =
1252   AOT_var_of_term_induct[simplified, induct type: AOT_var]
1253 and AOT_var_of_term_cases =
1254   AOT_var_of_term_cases[simplified, cases type: AOT_var]
1255 and AOT_term_of_var = AOT_term_of_var[simplified]
1256 and AOT_term_of_var_cases =
1257   AOT_term_of_var_cases[simplified, induct pred: AOT_term_of_var]
1258 and AOT_term_of_var_induct =
1259   AOT_term_of_var_induct[simplified, induct pred: AOT_term_of_var]
1260 and AOT_term_of_var_inverse = AOT_term_of_var_inverse[simplified]
1261 and AOT_term_of_var_inject = AOT_term_of_var_inject[simplified]
1262
1263 text<Equivalence by definition is modelled as necessary equivalence.>
1264 consts AOT_model_equiv_def :: <o ⇒ o ⇒ bool>
1265 specification(AOT_model_equiv_def)
1266   AOT_model_equiv_def: <AOT_model_equiv_def  $\varphi$   $\psi$  = ( $\forall v$  . AOT_model_valid_in v  $\varphi$  =
1267     AOT_model_valid_in v  $\psi$ )>
1268   by (rule exI[where x=< $\lambda \varphi \psi$  .  $\forall v$  . AOT_model_valid_in v  $\varphi$  =
1269     AOT_model_valid_in v  $\psi$ >]) simp
1270
1271 text<Identity by definition is modelled as identity for denoting terms plus
1272   co-denoting.>
1273 consts AOT_model_id_def :: <('b ⇒ 'a::AOT_Term) ⇒ ('b ⇒ 'a) ⇒ bool>
1274 specification(AOT_model_id_def)
1275   AOT_model_id_def: <(AOT_model_id_def  $\tau$   $\sigma$ ) = ( $\forall \alpha$  . if AOT_model_denotes ( $\sigma$   $\alpha$ )
1276     then  $\tau$   $\alpha$  =  $\sigma$   $\alpha$ 
1277     else  $\neg$ AOT_model_denotes ( $\tau$   $\alpha$ )>>
1278   by (rule exI[where x=" $\lambda \tau \sigma$  .  $\forall \alpha$  . if AOT_model_denotes ( $\sigma$   $\alpha$ )
1279     then  $\tau$   $\alpha$  =  $\sigma$   $\alpha$ 
1280     else  $\neg$ AOT_model_denotes ( $\tau$   $\alpha$ )"]])
1281
1282 blast
1283 text<To reduce definitions by identity without free variables to definitions
1284   by identity with free variables acting on the unit type, we give the unit type
1285   a trivial instantiation to @{class AOT_Term}.>
1286 instantiation unit :: AOT_Term
1287 begin
1288 definition AOT_model_denotes_unit :: <unit ⇒ bool> where
1289   <AOT_model_denotes_unit  $\equiv \lambda$ _. True>
1290 instance proof qed(simp add: AOT_model_denotes_unit_def)
1291 end
1292
1293 text<Modally-strict and modally-fragile axioms are as necessary,
1294   resp. actually valid propositions.>
1295 definition AOT_model_axiom where
1296   <AOT_model_axiom  $\equiv \lambda \varphi$  .  $\forall v$  . AOT_model_valid_in v  $\varphi$ >
1297 definition AOT_model_act_axiom where
1298   <AOT_model_act_axiom  $\equiv \lambda \varphi$  . AOT_model_valid_in  $w_0$   $\varphi$ >
1299
1300 lemma AOT_model_axiomI:
1301   assumes < $\bigwedge v$  . AOT_model_valid_in v  $\varphi$ >
1302   shows <AOT_model_axiom  $\varphi$ >
1303   unfolding AOT_model_axiom_def using assms ..
1304
1305 lemma AOT_model_act_axiomI:
1306   assumes <AOT_model_valid_in  $w_0$   $\varphi$ >
1307   shows <AOT_model_act_axiom  $\varphi$ >
1308   unfolding AOT_model_act_axiom_def using assms .
1309
1310 (*<*)
1311 end
1312 (*>*)

```

A.2. Outer Syntax Commands

```
1  (*<*)
2  theory AOT_commands
3    imports AOT_model "HOL-Eisbach.Eisbach_Tools"
4    keywords "AOT_define" :: thy_decl
5             and "AOT_theorem" :: thy_goal
6             and "AOT_lemma" :: thy_goal
7             and "AOT_act_theorem" :: thy_goal
8             and "AOT_act_lemma" :: thy_goal
9
10           and "AOT_axiom" :: thy_goal
11           and "AOT_act_axiom" :: thy_goal
12
13           and "AOT_assume" :: prf_asm % "proof"
14           and "AOT_have" :: prf_goal % "proof"
15           and "AOT_hence" :: prf_goal % "proof"
16           and "AOT_modally_strict {" :: prf_open % "proof"
17           and "AOT_actually {" :: prf_open % "proof"
18           and "AOT_obtain" :: prf_asm_goal % "proof"
19           and "AOT_show" :: prf_asm_goal % "proof"
20           and "AOT_thus" :: prf_asm_goal % "proof"
21
22           and "AOT_find_theorems" :: diag
23           and "AOT_sledgehammer" :: diag
24           and "AOT_sledgehammer_only" :: diag
25           and "AOT_syntax_print_translations" :: thy_decl
26           and "AOT_no_syntax_print_translations" :: thy_decl
27  begin
28  (*>*)
29
30  section<Outer Syntax Commands>
31
32  nonterminal AOT_prop
33  nonterminal  $\varphi$ 
34  nonterminal  $\varphi'$ 
35  nonterminal  $\tau$ 
36  nonterminal  $\tau'$ 
37  nonterminal "AOT_axiom"
38  nonterminal "AOT_act_axiom"
39  ML_file AOT_keys.ML
40  ML_file AOT_commands.ML
41  setup<AOT_Theorems.setup>
42  setup<AOT_Definitions.setup>
43  setup<AOT_no_atp.setup>
44
45  (*<*)
46  end
47  (*>*)
```

A.3. Approximation of the Syntax of PLM

```

1  (*<*)
2  theory AOT_syntax
3    imports AOT_commands
4    keywords "AOT_register_variable_names" :: thy_decl
5      and "AOT_register_metavariable_names" :: thy_decl
6      and "AOT_register_premise_set_names" :: thy_decl
7      and "AOT_register_type_constraints" :: thy_decl
8    abbrevs "actually" = " $\mathcal{A}$ "
9      and "necessarily" = " $\square$ "
10     and "possibly" = " $\diamond$ "
11     and "the" = " $\iota$ "
12     and "lambda" = " $[\lambda \bullet]$ "
13     and "being such that" = " $[\lambda \bullet]$ "
14     and "forall" = " $\forall$ "
15     and "exists" = " $\exists$ "
16     and "equivalent" = " $\equiv$ "
17     and "not" = " $\neg$ "
18     and "implies" = " $\rightarrow$ "
19     and "equal" = " $=$ "
20     and "by definition" = " $_{df}$ "
21     and "df" = " $_{df}$ "
22     and "denotes" = " $\downarrow$ "
23  begin
24  (*>*)
25
26  section<Approximation of the Syntax of PLM>
27
28  locale AOT_meta_syntax
29  begin
30  notation AOT_model_valid_in (" $[_ \models _]$ ")
31  notation AOT_model_axiom (" $\square[_]$ ")
32  notation AOT_model_act_axiom (" $\mathcal{A}[_]$ ")
33  end
34  locale AOT_no_meta_syntax
35  begin
36  no_notation AOT_model_valid_in (" $[_ \models _]$ ")
37  no_notation AOT_model_axiom (" $\square[_]$ ")
38  no_notation AOT_model_act_axiom (" $\mathcal{A}[_]$ ")
39  end
40
41  consts AOT_denotes :: <'a::AOT_Term  $\Rightarrow$  o>
42    AOT_imp :: <[o, o]  $\Rightarrow$  o>
43    AOT_not :: <o  $\Rightarrow$  o>
44    AOT_box :: <o  $\Rightarrow$  o>
45    AOT_act :: <o  $\Rightarrow$  o>
46    AOT_forall :: <('a::AOT_Term  $\Rightarrow$  o)  $\Rightarrow$  o>
47    AOT_eq :: <'a::AOT_Term  $\Rightarrow$  'a::AOT_Term  $\Rightarrow$  o>
48    AOT_desc :: <('a::AOT_UnaryIndividualTerm  $\Rightarrow$  o)  $\Rightarrow$  'a>
49    AOT_exe :: <<'a::AOT_IndividualTerm>  $\Rightarrow$  'a  $\Rightarrow$  o>
50    AOT_lambda :: <('a::AOT_IndividualTerm  $\Rightarrow$  o)  $\Rightarrow$  <'a>>
51    AOT_lambda0 :: <o  $\Rightarrow$  o>
52    AOT_concrete :: <<'a::AOT_UnaryIndividualTerm> AOT_var>
53
54  nonterminal  $\kappa_s$  and  $\Pi$  and  $\Pi_0$  and  $\alpha$  and exe_arg and exe_args
55    and lambda_args and desc and free_var and free_vars
56    and AOT_props and AOT_premises and AOT_world_relative_prop
57
58  syntax "_AOT_process_frees" :: < $\varphi \Rightarrow \varphi'$ > ("_")
59  "_AOT_verbatim" :: <any  $\Rightarrow \varphi$ > (<<_>>)
60  "_AOT_verbatim" :: <any  $\Rightarrow \tau$ > (<<_>>)
61  "_AOT_quoted" :: < $\varphi' \Rightarrow$  any> (<<_>>)

```

```

62     "_AOT_quoted" :: <τ' ⇒ any> (<<_>>)
63     "" :: <φ ⇒ φ> (<'(_')>)
64     "_AOT_process_frees" :: <τ ⇒ τ'> ("_")
65     "" :: <κs ⇒ τ> ("_")
66     "" :: <Π ⇒ τ> ("_")
67     "" :: <φ ⇒ τ> (<'(_')>)
68     "_AOT_term_var" :: <id_position ⇒ τ> ("_")
69     "_AOT_term_var" :: <id_position ⇒ φ> ("_")
70     "_AOT_exe_vars" :: <id_position ⇒ exe_arg> ("_")
71     "_AOT_lambda_vars" :: <id_position ⇒ lambda_args> ("_")
72     "_AOT_var" :: <id_position ⇒ α> ("_")
73     "_AOT_vars" :: <id_position ⇒ any>
74     "_AOT_verbatim" :: <any ⇒ α> (<<_>>)
75     "_AOT_valid" :: <w ⇒ φ' ⇒ bool> (<[_ ⊨ _]>)
76     "_AOT_denotes" :: <τ ⇒ φ> (<_↓>)
77     "_AOT_imp" :: <[φ, φ] ⇒ φ> (infixl <-> 25)
78     "_AOT_not" :: <φ ⇒ φ> (<_> [50] 50)
79     "_AOT_not" :: <φ ⇒ φ> (<¬_> [50] 50)
80     "_AOT_box" :: <φ ⇒ φ> (<□_> [49] 54)
81     "_AOT_act" :: <φ ⇒ φ> (<A_> [49] 54)
82     "_AOT_all" :: <α ⇒ φ ⇒ φ> (<∀_ _> [1,40])
83 syntax (input)
84     "_AOT_all_ellipse"
85         :: <id_position ⇒ id_position ⇒ φ ⇒ φ> (<∀...∀_ _> [1,40])
86 syntax (output)
87     "_AOT_all_ellipse"
88         :: <id_position ⇒ id_position ⇒ φ ⇒ φ> (<∀...∀_'(_')> [1,40])
89 syntax
90     "_AOT_eq" :: <[τ, τ] ⇒ φ> (infixl <=> 50)
91     "_AOT_desc" :: <α ⇒ φ ⇒ desc> ("ℓ_" [1,1000])
92     "" :: <desc ⇒ κs> ("_")
93     "_AOT_lambda" :: <lambda_args ⇒ φ ⇒ Π> (<[λ_ _]>)
94     "_explicitRelation" :: <τ ⇒ Π> ("[]")
95     "" :: <κs ⇒ exe_arg> ("_")
96     "" :: <exe_arg ⇒ exe_args> ("_")
97     "_AOT_exe_args" :: <exe_arg ⇒ exe_args ⇒ exe_args> ("__")
98     "_AOT_exe_arg_ellipse" :: <id_position ⇒ id_position ⇒ exe_arg> ("..._")
99     "_AOT_lambda_arg_ellipse"
100         :: <id_position ⇒ id_position ⇒ lambda_args> ("..._")
101     "_AOT_term_ellipse" :: <id_position ⇒ id_position ⇒ τ> ("..._")
102     "_AOT_exe" :: <Π ⇒ exe_args ⇒ φ> (<_>)
103     "_AOT_enc" :: <exe_args ⇒ Π ⇒ φ> (<_>)
104     "_AOT_lambda0" :: <φ ⇒ Π0> (<[λ _]>)
105     "" :: <Π0 ⇒ φ> ("_")
106     "" :: <Π0 ⇒ τ> ("_")
107     "_AOT_concrete" :: <Π> (<E!>)
108     "" :: <any ⇒ exe_arg> (<<_>>)
109     "" :: <desc ⇒ free_var> ("_")
110     "" :: <Π ⇒ free_var> ("_")
111     "_AOT_appl" :: <id_position ⇒ free_vars ⇒ φ> ("'[_']")
112     "_AOT_appl" :: <id_position ⇒ free_vars ⇒ τ> ("'[_']")
113     "_AOT_appl" :: <id_position ⇒ free_vars ⇒ free_vars> ("'[_']")
114     "_AOT_appl" :: <id_position ⇒ free_vars ⇒ free_vars> ("'[_']")
115     "_AOT_term_var" :: <id_position ⇒ free_var> ("_")
116     "" :: <any ⇒ free_var> (<<_>>)
117     "" :: <free_var ⇒ free_vars> ("_")
118     "_AOT_args" :: <free_var ⇒ free_vars ⇒ free_vars> ("_,_")
119     "_AOT_free_var_ellipse" :: <id_position ⇒ id_position ⇒ free_var> ("..._")
120 syntax "_AOT_premises"
121         :: <AOT_world_relative_prop ⇒ AOT_premises ⇒ AOT_premises> (infixr <,> 3)
122     "_AOT_world_relative_prop" :: "φ ⇒ AOT_world_relative_prop" ("_")
123     "" :: "AOT_world_relative_prop ⇒ AOT_premises" ("_")
124     "_AOT_prop" :: <AOT_world_relative_prop ⇒ AOT_prop> (<_>)

```

```

125     "" :: <AOT_prop ⇒ AOT_props> (<_>)
126     "_AOT_derivable" :: "AOT_premises ⇒  $\varphi$ ' ⇒ AOT_prop" (infixl <|→> 2)
127     "_AOT_nec_derivable" :: "AOT_premises ⇒  $\varphi$ ' ⇒ AOT_prop" (infixl <|→□> 2)
128     "_AOT_theorem" :: " $\varphi$ ' ⇒ AOT_prop" (<|→ _>)
129     "_AOT_nec_theorem" :: " $\varphi$ ' ⇒ AOT_prop" (<|→□ _>)
130     "_AOT_equiv_def" :: < $\varphi$  ⇒  $\varphi$  ⇒ AOT_prop> (infixl <≡df> 3)
131     "_AOT_axiom" :: " $\varphi$ ' ⇒ AOT_axiom" (<_>)
132     "_AOT_act_axiom" :: " $\varphi$ ' ⇒ AOT_act_axiom" (<_>)
133     "_AOT_axiom" :: " $\varphi$ ' ⇒ AOT_prop" (<_ ∈  $\Lambda$ □>)
134     "_AOT_act_axiom" :: " $\varphi$ ' ⇒ AOT_prop" (<_ ∈  $\Lambda$ >)
135     "_AOT_id_def" :: < $\tau$  ⇒  $\tau$  ⇒ AOT_prop> (infixl <=df> 3)
136     "_AOT_for_arbitrary"
137     :: <id_position ⇒ AOT_prop ⇒ AOT_prop> (<for arbitrary _ : _> [1000,1] 1)
138 syntax (output) "_lambda_args" :: <any ⇒ patterns ⇒ patterns> ("_")
139
140 translations
141   "[w |=  $\varphi$ ]" => "CONST AOT_model_valid_in w  $\varphi$ "
142
143 AOT_syntax_print_translations
144   "[w |=  $\varphi$ ]" <= "CONST AOT_model_valid_in w  $\varphi$ "
145
146 ML_file AOT_syntax.ML
147
148 AOT_register_type_constraints
149   Individual: <_::AOT_UnaryIndividualTerm> <_::AOT_IndividualTerm> and
150   Proposition: o and
151   Relation: <<_::AOT_IndividualTerm>> and
152   Term: <_::AOT_Term>
153
154 AOT_register_variable_names
155   Individual: x y z  $\nu$   $\mu$  a b c d and
156   Proposition: p q r s and
157   Relation: F G H P Q R S and
158   Term:  $\alpha$   $\beta$   $\gamma$   $\delta$ 
159
160 AOT_register_metavariable_names
161   Individual:  $\kappa$  and
162   Proposition:  $\varphi$   $\psi$   $\chi$   $\vartheta$   $\zeta$   $\xi$   $\Theta$  and
163   Relation:  $\Pi$  and
164   Term:  $\tau$   $\sigma$ 
165
166 AOT_register_premise_set_names  $\Gamma$   $\Delta$   $\Lambda$ 
167
168 parse_ast_translation<[
169   (syntax_const<_AOT_var>, K AOT_check_var),
170   (syntax_const<_AOT_exe_vars>, K AOT_split_exe_vars),
171   (syntax_const<_AOT_lambda_vars>, K AOT_split_lambda_args)
172 ]>
173
174 translations
175   "_AOT_denotes  $\tau$ " => "CONST AOT_denotes  $\tau$ "
176   "_AOT_imp  $\varphi$   $\psi$ " => "CONST AOT_imp  $\varphi$   $\psi$ "
177   "_AOT_not  $\varphi$ " => "CONST AOT_not  $\varphi$ "
178   "_AOT_box  $\varphi$ " => "CONST AOT_box  $\varphi$ "
179   "_AOT_act  $\varphi$ " => "CONST AOT_act  $\varphi$ "
180   "_AOT_eq  $\tau$   $\tau$ '" => "CONST AOT_eq  $\tau$   $\tau$ '"
181   "_AOT_lambda0  $\varphi$ " => "CONST AOT_lambda0  $\varphi$ "
182   "_AOT_concrete" => "CONST AOT_term_of_var (CONST AOT_concrete)"
183   "_AOT_lambda  $\alpha$   $\varphi$ " => "CONST AOT_lambda (_abs  $\alpha$   $\varphi$ )"
184   "_explicitRelation  $\Pi$ " => " $\Pi$ "
185
186 AOT_syntax_print_translations
187   "_AOT_lambda (_lambda_args x y)  $\varphi$ " <= "CONST AOT_lambda (_abs (_pattern x y)  $\varphi$ )"

```

```

188   "_AOT_lambda (_lambda_args x y)  $\varphi$ " <= "CONST AOT_lambda (_abs (_patterns x y)  $\varphi$ )"
189   "_AOT_lambda x  $\varphi$ " <= "CONST AOT_lambda (_abs x  $\varphi$ )"
190   "_lambda_args x (_lambda_args y z)" <= "_lambda_args x (_patterns y z)"
191   "_lambda_args (x y z)" <= "_lambda_args (_tuple x (_tuple_arg (_tuple y z)))"
192
193
194 AOT_syntax_print_translations
195   "_AOT_imp  $\varphi \psi$ " <= "CONST AOT_imp  $\varphi \psi$ "
196   "_AOT_not  $\varphi$ " <= "CONST AOT_not  $\varphi$ "
197   "_AOT_box  $\varphi$ " <= "CONST AOT_box  $\varphi$ "
198   "_AOT_act  $\varphi$ " <= "CONST AOT_act  $\varphi$ "
199   "_AOT_all  $\alpha \varphi$ " <= "CONST AOT_forall (_abs  $\alpha \varphi$ )"
200   "_AOT_all  $\alpha \varphi$ " <= "CONST AOT_forall ( $\lambda\alpha. \varphi$ )"
201   "_AOT_eq  $\tau \tau'$ " <= "CONST AOT_eq  $\tau \tau'$ "
202   "_AOT_desc x  $\varphi$ " <= "CONST AOT_desc (_abs x  $\varphi$ )"
203   "_AOT_desc x  $\varphi$ " <= "CONST AOT_desc ( $\lambda x. \varphi$ )"
204   "_AOT_lambda0  $\varphi$ " <= "CONST AOT_lambda0  $\varphi$ "
205   "_AOT_concrete" <= "CONST AOT_term_of_var (CONST AOT_concrete)"
206
207 translations
208   "_AOT_appl  $\varphi$  (_AOT_args a b)" => "_AOT_appl ( $\varphi$  a) b"
209   "_AOT_appl  $\varphi$  a" => " $\varphi$  a"
210
211
212 parse_translation<
213 [
214   (syntax_const<_AOT_var>, parseVar true),
215   (syntax_const<_AOT_vars>, parseVar false),
216   (syntax_const<_AOT_valid>, fn ctxt => fn [w,x] =>
217     const<AOT_model_valid_in> $ w $ x),
218   (syntax_const<_AOT_quoted>, fn ctxt => fn [x] => x),
219   (syntax_const<_AOT_process_frees>, fn ctxt => fn [x] => processFrees ctxt x),
220   (syntax_const<_AOT_world_relative_prop>, fn ctxt => fn [x] => let
221     val (x, premises) = processFreesAndPremises ctxt x
222     val (world::formulas) = Variable.variant_frees ctxt [x]
223     (( $\forall$ ", dummyT)::(map (fn _ => (" $\varphi$ ", dummyT)) premises))
224     val term = HLogic.mk_Trueprop
225       (@{const AOT_model_valid_in} $ Free world $ processFrees ctxt x)
226     val term = fold (fn (premise,form) => fn trm =>
227       @{const "Pure.imp"} $
228       HLogic.mk_Trueprop
229         (Const (const_name<Set.member>, dummyT) $ Free form $ premise) $
230         (Term.absfree (Term.dest_Free (dropConstraints premise)) trm $ Free form)
231     ) (ListPair.zipEq (premises,formulas)) term
232     val term = fold (fn (form) => fn trm =>
233       Const (const_name<Pure.all>, dummyT) $
234       (Term.absfree form trm)
235     ) formulas term
236     val term = Term.absfree world term
237     in term end),
238   (syntax_const<_AOT_prop>, fn ctxt => fn [x] => let
239     val world = case (AOT_ProofData.get ctxt) of SOME w => w
240     | _ => raise Fail "Expected world to be stored in the proof state."
241     in x $ world end),
242   (syntax_const<_AOT_theorem>, fn ctxt => fn [x] =>
243     HLogic.mk_Trueprop (@{const AOT_model_valid_in} $ @{const w0} $ x)),
244   (syntax_const<_AOT_axiom>, fn ctxt => fn [x] =>
245     HLogic.mk_Trueprop (@{const AOT_model_axiom} $ x)),
246   (syntax_const<_AOT_act_axiom>, fn ctxt => fn [x] =>
247     HLogic.mk_Trueprop (@{const AOT_model_act_axiom} $ x)),
248   (syntax_const<_AOT_nec_theorem>, fn ctxt => fn [trm] => let
249     val world = singleton (Variable.variant_frees ctxt [trm]) (" $\forall$ ", @{typ w})
250     val trm = HLogic.mk_Trueprop (@{const AOT_model_valid_in} $ Free world $ trm)

```

```

251   val trm = Term.absfree world trm
252   val trm = Const (const_name<Pure.all>, dummyT) $ trm
253   in trm end),
254 (syntax_const<_AOT_derivable>, fn ctxt => fn [x,y] => let
255   val world = case (AOT_ProofData.get ctxt) of SOME w => w
256   | _ => raise Fail "Expected world to be stored in the proof state."
257   in foldPremises world x y end),
258 (syntax_const<_AOT_nec_derivable>, fn ctxt => fn [x,y] => let
259   in Const (const_name<Pure.all>, dummyT) $
260     Abs ("v", dummyT, foldPremises (Bound 0) x y) end),
261 (syntax_const<_AOT_for_arbitrary>, fn ctxt => fn [_ $ var $ pos, trm] => let
262   val trm = Const (const_name<Pure.all>, dummyT) $
263     (Const ("_constrainAbs", dummyT) $ Term.absfree (Term.dest_Free var) trm $ pos)
264   in trm end),
265 (syntax_const<_AOT_equiv_def>, parseEquivDef),
266 (syntax_const<_AOT_exe>, parseExe),
267 (syntax_const<_AOT_enc>, parseEnc)
268 ]
269 >
270
271 parse_ast_translation<
272 [
273 (syntax_const<_AOT_exe_arg_ellipse>, parseEllipseList "_AOT_term_vars"),
274 (syntax_const<_AOT_lambda_arg_ellipse>, parseEllipseList "_AOT_vars"),
275 (syntax_const<_AOT_free_var_ellipse>, parseEllipseList "_AOT_term_vars"),
276 (syntax_const<_AOT_term_ellipse>, parseEllipseList "_AOT_term_vars"),
277 (syntax_const<_AOT_all_ellipse>, fn ctx => fn [a,b,c] =>
278   Ast.mk_appl (Ast.Constant const_name<AOT_forall>) [
279     Ast.mk_appl (Ast.Constant "_abs") [parseEllipseList "_AOT_vars" ctx [a,b],c]
280   ])
281 ]
282 >
283
284 syntax (output)
285   "_AOT_individual_term" :: <'a => tuple_args> ("_")
286   "_AOT_individual_terms" :: <tuple_args => tuple_args => tuple_args> ("__")
287   "_AOT_relation_term" :: <'a => Π>
288   "_AOT_any_term" :: <'a => τ>
289
290
291 print_ast_translation<AOT_syntax_print_ast_translations[
292 (syntax_const<_AOT_individual_term>, AOT_print_individual_term),
293 (syntax_const<_AOT_relation_term>, AOT_print_relation_term),
294 (syntax_const<_AOT_any_term>, AOT_print_generic_term)
295 ]>
296
297 AOT_syntax_print_translations
298   "_AOT_individual_terms (_AOT_individual_term x) (_AOT_individual_terms (_tuple y z))"
299   <= "_AOT_individual_terms (_tuple x (_tuple_args y z))"
300   "_AOT_individual_terms (_AOT_individual_term x) (_AOT_individual_term y)"
301   <= "_AOT_individual_terms (_tuple x (_tuple_arg y))"
302   "_AOT_individual_terms (_tuple x y)" <= "_AOT_individual_term (_tuple x y)"
303   "_AOT_exe (_AOT_relation_term Π) (_AOT_individual_term κ)" <= "CONST AOT_exe Π κ"
304   "_AOT_denotes (_AOT_any_term κ)" <= "CONST AOT_denotes κ"
305
306 AOT_define AOT_conj :: <[φ, φ] => φ> (infixl <&> 35) <φ & ψ ≡df ¬(φ → ¬ψ)>
307 declare "AOT_conj"[AOT del, AOT_defs del]
308 AOT_define AOT_disj :: <[φ, φ] => φ> (infixl <V> 35) <φ ∨ ψ ≡df ¬φ → ψ>
309 declare "AOT_disj"[AOT del, AOT_defs del]
310 AOT_define AOT_equiv :: <[φ, φ] => φ> (infixl <≡> 20) <φ ≡ ψ ≡df (φ → ψ) & (ψ → φ)>
311 declare "AOT_equiv"[AOT del, AOT_defs del]
312 AOT_define AOT_dia :: <φ => φ> (<◇> [49] 54) <◇φ ≡df ¬□¬φ>
313 declare "AOT_dia"[AOT del, AOT_defs del]

```

```

314
315 context AOT_meta_syntax
316 begin
317 notation AOT_dia ("◇_" [49] 54)
318 notation AOT_conj (infixl <&> 35)
319 notation AOT_disj (infixl <∨> 35)
320 notation AOT_equiv (infixl <≡> 20)
321 end
322 context AOT_no_meta_syntax
323 begin
324 no_notation AOT_dia ("◇_" [49] 54)
325 no_notation AOT_conj (infixl <&> 35)
326 no_notation AOT_disj (infixl <∨> 35)
327 no_notation AOT_equiv (infixl <≡> 20)
328 end
329
330
331 print_translation <
332 AOT_syntax_print_translations
333 [
334   AOT_preserve_binder_abs_tr'
335     const_syntax<AOT_forall>
336     syntax_const<_AOT_all>
337     (syntax_const<_AOT_all_ellipse>, true)
338     const_name<AOT_imp>,
339   AOT_binder_trans @{theory} @{binding "AOT_forall_binder"} syntax_const<_AOT_all>,
340   Syntax_Trans.preserve_binder_abs_tr'
341     const_syntax<AOT_desc>
342     syntax_const<_AOT_desc>,
343   AOT_binder_trans @{theory} @{binding "AOT_desc_binder"} syntax_const<_AOT_desc>,
344   AOT_preserve_binder_abs_tr'
345     const_syntax<AOT_lambda>
346     syntax_const<_AOT_lambda>
347     (syntax_const<_AOT_lambda_arg_ellipse>, false)
348     const_name<undefined>,
349   AOT_binder_trans
350     @{theory}
351     @{binding "AOT_lambda_binder"}
352     syntax_const<_AOT_lambda>
353 ]
354 >
355
356 parse_translation<
357 [(syntax_const<_AOT_id_def>, parseIdDef)]
358 >
359
360 parse_ast_translation<[
361   (syntax_const<_AOT_all>,
362     AOT_restricted_binder const_name<AOT_forall> const_name<AOT_imp>),
363   (syntax_const<_AOT_desc>,
364     AOT_restricted_binder const_name<AOT_desc> const_name<AOT_conj>)
365 ]>
366
367 AOT_define AOT_exists :: <α ⇒ φ ⇒ φ> <<"AOT_exists φ" ≡df ¬∀α ¬φ{α}>>
368 declare AOT_exists[AOT_del, AOT_defs del]
369 syntax "_AOT_exists" :: <α ⇒ φ ⇒ φ> ("∃_ _" [1,40])
370
371 AOT_syntax_print_translations
372   "_AOT_exists α φ" <= "CONST AOT_exists (_abs α φ)"
373   "_AOT_exists α φ" <= "CONST AOT_exists (λα. φ)"
374
375 parse_ast_translation<
376 [(syntax_const<_AOT_exists>,

```



```

377   AOT_restricted_binder const_name<AOT_exists> const_name<AOT_conj>)]
378 >
379
380 context AOT_meta_syntax
381 begin
382 notation AOT_exists (binder "∃" 8)
383 end
384 context AOT_no_meta_syntax
385 begin
386 no_notation AOT_exists (binder "∃" 8)
387 end
388
389
390 syntax (input)
391   "_AOT_exists_ellipse" :: <id_position ⇒ id_position ⇒ φ ⇒ φ> (<∃...∃_ _> [1,40])
392 syntax (output)
393   "_AOT_exists_ellipse" :: <id_position ⇒ id_position ⇒ φ ⇒ φ> (<∃...∃_ '(<_>)> [1,40])
394 parse_ast_translation<[(syntax_const<_AOT_exists_ellipse>, fn ctx => fn [a,b,c] =>
395   Ast.mk_appl (Ast.Constant "AOT_exists")
396     [Ast.mk_appl (Ast.Constant "_abs") [parseEllipseList "_AOT_vars" ctx [a,b],c]]]>
397 print_translation<AOT_syntax_print_translations [
398   AOT_preserve_binder_abs_tr'
399   const_syntax<AOT_exists>
400   syntax_const<_AOT_exists>
401   (syntax_const<_AOT_exists_ellipse>,true) const_name<AOT_conj>,
402   AOT_binder_trans
403   @{theory}
404   @{binding "AOT_exists_binder"}
405   syntax_const<_AOT_exists>
406 ]>
407
408
409
410 syntax "_AOT_DDDOT" :: "φ" ("...")
411 syntax "_AOT_DDDOT" :: "φ" ("...")
412 parse_translation<[(syntax_const<_AOT_DDDOT>, parseDDOT)]>
413
414 print_translation<AOT_syntax_print_translations
415 [(const_syntax<Pure.all>, fn ctx => fn [Abs (_, _),
416   Const (const_syntax<HOL.Trueprop>, _) $
417   (Const (const_syntax<AOT_model_valid_in>, _) $ Bound 0 $ y))] => let
418   val y = (Const (syntax_const<_AOT_process_frees>, dummyT) $ y)
419   in (Const (syntax_const<_AOT_nec_theorem>, dummyT) $ y) end
420 | [p as Abs (name, _),
421   Const (const_syntax<HOL.Trueprop>, _) $
422   (Const (const_syntax<AOT_model_valid_in>, _) $ w $ y)]]
423 => (Const (syntax_const<_AOT_for_arbitrary>, dummyT) $
424   (Const ("_bound", dummyT) $ Free (name, dummyT)) $
425   (Term.betapply (p, (Const ("_bound", dummyT) $ Free (name, dummyT))))))
426 ),
427
428 (const_syntax<AOT_model_valid_in>, fn ctx =>
429   fn [w as (Const ("_free", _) $ Free (v, _)), y] => let
430     val is_world = (case (AOT_ProofData.get ctx)
431       of SOME (Free (w, _)) => Name.clean w = Name.clean v | _ => false)
432     val y = (Const (syntax_const<_AOT_process_frees>, dummyT) $ y)
433     in if is_world then y else Const (syntax_const<_AOT_valid>, dummyT) $ w $ y end
434 | [Const (const_syntax<w₀>, _) , y] => let
435     val y = (Const (syntax_const<_AOT_process_frees>, dummyT) $ y)
436     in case (AOT_ProofData.get ctx) of SOME (Const (const_name<w₀>, _)) => y |
437       _ => Const (syntax_const<_AOT_theorem>, dummyT) $ y end
438 | [Const ("_var", _) $ _, y] => let
439     val y = (Const (syntax_const<_AOT_process_frees>, dummyT) $ y)

```

```

440   in Const (syntax_const<_AOT_nec_theorem>, dummyT) $ y end
441   ),
442   (const_syntax<AOT_model_axiom>, fn ctxt => fn [trm] =>
443     Const (syntax_const<_AOT_axiom>, dummyT) $
444     (Const (syntax_const<_AOT_process_frees>, dummyT) $ trm)),
445   (const_syntax<AOT_model_act_axiom>, fn ctxt => fn [trm] =>
446     Const (syntax_const<_AOT_axiom>, dummyT) $
447     (Const (syntax_const<_AOT_process_frees>, dummyT) $ trm)),
448   (syntax_const<_AOT_process_frees>, fn _ => fn [t] => let
449     fun mapAppls (x as Const ("_free", _) $
450       Free (_, Type ("_ignore_type", [Type ("fun", _)])))
451       = (Const ("_AOT_raw_appl", dummyT) $ x)
452     | mapAppls (x as Const ("_free", _) $ Free (_, Type ("fun", _)))
453       = (Const ("_AOT_raw_appl", dummyT) $ x)
454     | mapAppls (x as Const ("_var", _) $
455       Var (_, Type ("_ignore_type", [Type ("fun", _)])))
456       = (Const ("_AOT_raw_appl", dummyT) $ x)
457     | mapAppls (x as Const ("_var", _) $ Var (_, Type ("fun", _)))
458       = (Const ("_AOT_raw_appl", dummyT) $ x)
459     | mapAppls (x $ y) = mapAppls x $ mapAppls y
460     | mapAppls (Abs (x,y,z)) = Abs (x,y, mapAppls z)
461     | mapAppls x = x
462   in mapAppls t end
463   )
464 ]
465 >
466
467 print_ast_translation<AOT_syntax_print_ast_translations
468 let
469 fun handleTermOfVar x kind name = (
470 let
471 val _ = case kind of "_free" => () | "_var" => () | "_bound" => () | _ => raise Match
472 in
473 case printVarKind name
474   of (SingleVariable name) => Ast.Appl [Ast.Constant kind, Ast.Variable name]
475     | (Ellipses (s, e)) => Ast.Appl [Ast.Constant "_AOT_free_var_ellipse",
476     Ast.Appl [Ast.Constant kind, Ast.Variable s],
477     Ast.Appl [Ast.Constant kind, Ast.Variable e]
478     ]
479     | Verbatim name => Ast.mk_appl (Ast.Constant "_AOT_quoted")
480     [Ast.mk_appl (Ast.Constant "_AOT_term_of_var") [x]]
481 end
482 )
483 fun termOfVar ctxt (Ast.Appl [Ast.Constant "_constrain",
484   x as Ast.Appl [Ast.Constant kind, Ast.Variable name], _]) = termOfVar ctxt x
485   | termOfVar ctxt (x as Ast.Appl [Ast.Constant kind, Ast.Variable name])
486     = handleTermOfVar x kind name
487   | termOfVar ctxt (x as Ast.Appl [Ast.Constant rep, y]) = (
488 let
489 val (restr,_) = Local_Theory.raw_theory_result (fn thy => (
490 let
491 val restrs = Syntab.dest (AOT_Restriction.get thy)
492 val restr = List.find (fn (n,(_,Const (c,t))) => (
493   c = rep orelse c = Lexicon.unmark_const rep) | _ => false) restrs
494 in
495 (restr,thy)
496 end
497 )) ctxt
498 in
499 case restr of SOME r => Ast.Appl [Ast.Constant (const_syntax<AOT_term_of_var>), y]
500   | _ => raise Match
501 end)
502

```

```

503 in
504 [(const_syntax<AOT_term_of_var>, fn ctxt => fn [x] => termOfVar ctxt x),
505 ("_AOT_raw_appl", fn ctxt => fn t::a::args => let
506 fun applyTermOfVar (t as Ast.Appl (Ast.Constant const_syntax<AOT_term_of_var>::[x]))
507   = (case try (termOfVar ctxt) x of SOME y => y | _ => t)
508   | applyTermOfVar y = (case try (termOfVar ctxt) y of SOME x => x | _ => y)
509 val ts = fold (fn a => fn b => Ast.mk_appl (Ast.Constant syntax_const<_AOT_args>)
510   [b,applyTermOfVar a]) args (applyTermOfVar a)
511 in Ast.mk_appl (Ast.Constant syntax_const<_AOT_appl>) [t,ts] end)]
512 end
513 >
514
515 context AOT_meta_syntax
516 begin
517 notation AOT_denotes ("_↓")
518 notation AOT_imp (infixl "→" 25)
519 notation AOT_not ("¬" [50] 50)
520 notation AOT_box ("□" [49] 54)
521 notation AOT_act ("ℳ" [49] 54)
522 notation AOT_forall (binder "∀" 8)
523 notation AOT_eq (infixl "=" 50)
524 notation AOT_desc (binder "ℓ" 100)
525 notation AOT_lambda (binder "λ" 100)
526 notation AOT_lambda0 ("[λ _]")
527 notation AOT_exe ("(|_,_)")
528 notation AOT_model_equiv_def (infixl "≡df" 10)
529 notation AOT_model_id_def (infixl "=df" 10)
530 notation AOT_term_of_var ("⟨_⟩")
531 notation AOT_concrete ("E!")
532 end
533 context AOT_no_meta_syntax
534 begin
535 no_notation AOT_denotes ("_↓")
536 no_notation AOT_imp (infixl "→" 25)
537 no_notation AOT_not ("¬" [50] 50)
538 no_notation AOT_box ("□" [49] 54)
539 no_notation AOT_act ("ℳ" [49] 54)
540 no_notation AOT_forall (binder "∀" 8)
541 no_notation AOT_eq (infixl "=" 50)
542 no_notation AOT_desc (binder "ℓ" 100)
543 no_notation AOT_lambda (binder "λ" 100)
544 no_notation AOT_lambda0 ("[λ _]")
545 no_notation AOT_exe ("(|_,_)")
546 no_notation AOT_model_equiv_def (infixl "≡df" 10)
547 no_notation AOT_model_id_def (infixl "=df" 10)
548 no_notation AOT_term_of_var ("⟨_⟩")
549 no_notation AOT_concrete ("E!")
550 end
551
552 bundle AOT_syntax
553 begin
554 declare[[show_AOT_syntax=true, show_question_marks=false, eta_contract=false]]
555 end
556
557 bundle AOT_no_syntax
558 begin
559 declare[[show_AOT_syntax=false, show_question_marks=true]]
560 end
561
562 parse_translation<
563 [("_AOT_restriction", fn ctxt => fn [Const (name,_)] =>
564 let
565 val (restr, ctxt) = ctxt |> Local_Theory.raw_theory_result

```

```

566   (fn thy => (Option.map fst (Syntab.lookup (AOT_Restriction.get thy) name), thy))
567 val restr = case restr of SOME x => x
568   | _ => raise Fail ("Unknown restricted type: " ^ name)
569 in restr end
570 )]
571 >
572
573 print_translation<
574 AOT_syntax_print_translations
575 [
576   (const_syntax<AOT_model_equiv_def>, fn ctxt => fn [x,y] =>
577     Const (syntax_const<_AOT_equiv_def>, dummyT) $
578     (Const (syntax_const<_AOT_process_frees>, dummyT) $ x) $
579     (Const (syntax_const<_AOT_process_frees>, dummyT) $ y))
580 ]
581 >
582
583 print_translation<
584 AOT_syntax_print_translations [
585   (const_syntax<AOT_model_id_def>, fn ctxt =>
586     fn [lhs as Abs (lhsName, lhsTy, lhsTrm), rhs as Abs (rhsName, rhsTy, rhsTrm)] =>
587       let
588         val (name, _) = Name.variant lhsName
589           (Term.declare_term_names rhsTrm (Term.declare_term_names lhsTrm Name.context));
590         val lhs = Term.betapply (lhs, Const ("_bound", dummyT) $ Free (name, lhsTy))
591         val rhs = Term.betapply (rhs, Const ("_bound", dummyT) $ Free (name, rhsTy))
592       in
593         Const (const_syntax<AOT_model_id_def>, dummyT) $ lhs $ rhs
594       end
595     | [Const (const_syntax<case_prod>, _) $ lhs,
596       Const (const_syntax<case_prod>, _) $ rhs] =>
597       Const (const_syntax<AOT_model_id_def>, dummyT) $ lhs $ rhs
598     | [Const (const_syntax<case_unit>, _) $ lhs,
599       Const (const_syntax<case_unit>, _) $ rhs] =>
600       Const (const_syntax<AOT_model_id_def>, dummyT) $ lhs $ rhs
601     | [x, y] =>
602       Const (syntax_const<_AOT_id_def>, dummyT) $
603         (Const (syntax_const<_AOT_process_frees>, dummyT) $ x) $
604         (Const (syntax_const<_AOT_process_frees>, dummyT) $ y)
605   )]>
606
607 text<Special marker for printing propositions as theorems
608   and for pretty-printing AOT terms.>
609 definition print_as_theorem :: <o ⇒ bool> where
610   <print_as_theorem ≡ λ φ . ∀v . [v ⊨ φ]>
611 lemma print_as_theoremI:
612   assumes <∧ v . [v ⊨ φ]>
613   shows <print_as_theorem φ>
614   using assms by (simp add: print_as_theorem_def)
615 attribute_setup print_as_theorem =
616   <Scan.succeed (Thm.rule_attribute []
617     (K (fn thm => thm RS @{{thm print_as_theoremI}}))>
618   "Print as theorem."
619 print_translation<AOT_syntax_print_translations [
620   (const_syntax<print_as_theorem>, fn ctxt => fn [x] =>
621     (Const (syntax_const<_AOT_process_frees>, dummyT) $ x))
622 ]>
623
624 definition print_term :: <'a ⇒ 'a> where <print_term ≡ λ x . x>
625 syntax "_AOT_print_term" :: <τ ⇒ 'a> (<AOT'_TERM[_]>)
626 translations
627   "_AOT_print_term φ" => "CONST print_term (_AOT_process_frees φ)"
628 print_translation<AOT_syntax_print_translations [

```

```
629   (const_syntax<print_term>, fn ctxt => fn [x] =>
630     (Const (syntax_const<_AOT_process_frees>, dummyT) $ x))
631 ]>
632
633
634 (* To enable meta syntax: *)
635 (* interpretation AOT_meta_syntax. *)
636 (* To disable meta syntax: *)
637 interpretation AOT_no_meta_syntax.
638
639 (* To enable AOT syntax (takes precedence over meta syntax;
640    can be done locally using "including" or "include"): *)
641 unbundle AOT_syntax
642 (* To disable AOT syntax (restoring meta syntax or no syntax;
643    can be done locally using "including" or "include"): *)
644 (* unbundle AOT_no_syntax *)
645
646 (*<*)
647 end
648 (*>*)
649
```

A.4. Semantics

```

1  (*<*)
2  theory AOT_semantics
3    imports AOT_syntax
4  begin
5  (*>*)
6
7  section<Abstract Semantics for AOT>
8
9  specification(AOT_denotes)
10   - <Relate object level denoting to meta-denoting. AOT's definitions of
11     denoting will become derivable at each type.>
12   AOT_sem_denotes: <[w ⊨ τ↓] = AOT_model_denotes τ>
13   by (rule exI[where x=<λ τ . εo w . AOT_model_denotes τ>])
14     (simp add: AOT_model_proposition_choice_simp)
15
16 lemma AOT_sem_var_induct[induct type: AOT_var]:
17   assumes AOT_denoting_term_case: <∧ τ . [v ⊨ τ↓] ⇒ [v ⊨ φ{τ}]>
18   shows <[v ⊨ φ{α}]>
19   by (simp add: AOT_denoting_term_case AOT_sem_denotes AOT_term_of_var)
20
21 text<\linelabel{AOT_imp_spec}>
22 specification(AOT_imp)
23   AOT_sem_imp: <[w ⊨ φ → ψ] = ([w ⊨ φ] → [w ⊨ ψ])>
24   by (rule exI[where x=<λ φ ψ . εo w . ([w ⊨ φ] → [w ⊨ ψ])>])
25     (simp add: AOT_model_proposition_choice_simp)
26
27 specification(AOT_not)
28   AOT_sem_not: <[w ⊨ ¬φ] = (¬[w ⊨ φ])>
29   by (rule exI[where x=<λ φ . εo w . ¬[w ⊨ φ]>])
30     (simp add: AOT_model_proposition_choice_simp)
31
32 text<\linelabel{AOT_box_spec}>
33 specification(AOT_box)
34   AOT_sem_box: <[w ⊨ □φ] = (∀ w . [w ⊨ φ])>
35   by (rule exI[where x=<λ φ . εo w . ∀ w . [w ⊨ φ]>])
36     (simp add: AOT_model_proposition_choice_simp)
37
38 text<\linelabel{AOT_act_spec}>
39 specification(AOT_act)
40   AOT_sem_act: <[w ⊨  $\mathcal{A}\varphi$ ] = [w0 ⊨ φ]>
41   by (rule exI[where x=<λ φ . εo w . [w0 ⊨ φ]>])
42     (simp add: AOT_model_proposition_choice_simp)
43
44 text<Derived semantics for basic defined connectives.>
45 lemma AOT_sem_conj: <[w ⊨ φ & ψ] = ([w ⊨ φ] ∧ [w ⊨ ψ])>
46   using AOT_conj AOT_model_equiv_def AOT_sem_imp AOT_sem_not by auto
47 lemma AOT_sem_equiv: <[w ⊨ φ ≡ ψ] = ([w ⊨ φ] = [w ⊨ ψ])>
48   using AOT_equiv AOT_sem_conj AOT_model_equiv_def AOT_sem_imp by auto
49 lemma AOT_sem_disj: <[w ⊨ φ ∨ ψ] = ([w ⊨ φ] ∨ [w ⊨ ψ])>
50   using AOT_disj AOT_model_equiv_def AOT_sem_imp AOT_sem_not by auto
51 lemma AOT_sem_dia: <[w ⊨ ◇φ] = (∃ w . [w ⊨ φ])>
52   using AOT_dia AOT_sem_box AOT_model_equiv_def AOT_sem_not by auto
53
54 specification(AOT_forall)
55   AOT_sem_forall: <[w ⊨ ∀α φ{α}] = (∀ τ . [w ⊨ τ↓] → [w ⊨ φ{τ}])>
56   by (rule exI[where x=<λ op . εo w . ∀ τ . [w ⊨ τ↓] → [w ⊨ «op τ»>])
57     (simp add: AOT_model_proposition_choice_simp)
58
59 lemma AOT_sem_exists: <[w ⊨ ∃α φ{α}] = (∃ τ . [w ⊨ τ↓] ∧ [w ⊨ φ{τ}])>
60   unfolding AOT_exists[unfolded AOT_model_equiv_def, THEN spec]
61   by (simp add: AOT_sem_forall AOT_sem_not)

```

```

62
63 text<\linelabel{AOT_eq_spec}>
64 specification(AOT_eq)
65   - <Relate identity to denoting identity in the meta-logic. AOT's definitions
66     of identity will become derivable at each type.>
67   AOT_sem_eq: <[w ⊨ τ = τ'] = ([w ⊨ τ↓] ∧ [w ⊨ τ'↓] ∧ τ = τ')>
68   by (rule exI[where x=<λ τ τ' . εo w . [w ⊨ τ↓] ∧ [w ⊨ τ'↓] ∧ τ = τ']])
69     (simp add: AOT_model_proposition_choice_simp)
70
71 text<\linelabel{AOT_desc_spec}>
72 specification(AOT_desc)
73   - <Descriptions denote, if there is a unique denoting object satisfying the
74     matrix in the actual world.>
75   AOT_sem_desc_denotes: <[w ⊨ ιx(φ{x})↓] = (∃! κ . [w0 ⊨ κ↓] ∧ [w0 ⊨ φ{κ}])>
76   - <Denoting descriptions satisfy their matrix in the actual world.>
77   AOT_sem_desc_prop: <[w ⊨ ιx(φ{x})↓] ⇒ [w0 ⊨ φ{ιx(φ{x})}]>
78   - <Uniqueness of denoting descriptions.>
79   AOT_sem_desc_unique: <[w ⊨ ιx(φ{x})↓] ⇒ [w ⊨ κ↓] ⇒ [w0 ⊨ φ{κ}] ⇒
80     [w ⊨ ιx(φ{x}) = κ]>
81 proof -
82   have <∃x::'a . ¬AOT_model_denotes x>
83     using AOT_model_nondenoting_ex
84     by blast
85   text<Note that we may choose a distinct non-denoting object for each matrix.
86     We do this explicitly merely to convince ourselves that our specification
87     can still be satisfied.>
88   then obtain nondenoting :: <'a ⇒ o> ⇒ 'a where
89     nondenoting: <∀ φ . ¬AOT_model_denotes (nondenoting φ)>
90     by fast
91   define desc where
92     <desc = (λ φ . if (∃! κ . [w0 ⊨ κ↓] ∧ [w0 ⊨ φ{κ}])
93       then (THE κ . [w0 ⊨ κ↓] ∧ [w0 ⊨ φ{κ}])
94       else nondenoting φ)>
95   {
96     fix φ :: <'a ⇒ o>
97     assume ex1: <∃! κ . [w0 ⊨ κ↓] ∧ [w0 ⊨ φ{κ}]>
98     then obtain κ where x_prop: "[w0 ⊨ κ↓] ∧ [w0 ⊨ φ{κ}]"
99       unfolding AOT_sem_denotes by blast
100    moreover have "(desc φ) = κ"
101      unfolding desc_def using x_prop ex1 by fastforce
102    ultimately have "[w0 ⊨ «desc φ»↓] ∧ [w0 ⊨ «φ (desc φ)»]"
103      by blast
104  } note 1 = this
105  moreover {
106    fix φ :: <'a ⇒ o>
107    assume nex1: <∄! κ . [w0 ⊨ κ↓] ∧ [w0 ⊨ φ{κ}]>
108    hence "(desc φ) = nondenoting φ" by (simp add: desc_def AOT_sem_denotes)
109    hence "[w ⊨ ¬«desc φ»↓]" for w
110      by (simp add: AOT_sem_denotes nondenoting AOT_sem_not)
111  }
112  ultimately have desc_denotes_simp:
113    <[w ⊨ «desc φ»↓] = (∃! κ . [w0 ⊨ κ↓] ∧ [w0 ⊨ φ{κ}])> for φ w
114  by (simp add: AOT_sem_denotes desc_def nondenoting)
115  have <(∀ φ w. [w ⊨ «desc φ»↓] = (∃! κ. [w0 ⊨ κ↓] ∧ [w0 ⊨ φ{κ}])) ∧
116    (∀ φ w. [w ⊨ «desc φ»↓] → [w0 ⊨ «φ (desc φ)»]) ∧
117    (∀ φ w κ. [w ⊨ «desc φ»↓] → [w ⊨ κ↓] → [w0 ⊨ φ{κ}] →
118      [w ⊨ «desc φ» = κ])>
119  by (insert 1; auto simp: desc_denotes_simp AOT_sem_eq AOT_sem_denotes
120    desc_def nondenoting)
121  thus ?thesis
122  by (safe intro!: exI[where x=desc]; presburger)
123 qed
124

```

```

125 text<\linelabel{AOT_exe_lambda_spec}>
126 specification(AOT_exe AOT_lambda)
127   - <Truth conditions of exemplification formulas.>
128   AOT_sem_exe: <[w ⊨ [Π] κ1...κn] = ([w ⊨ Π↓] ∧ [w ⊨ κ1...κn↓] ∧
129     [w ⊨ «Rep_rel Π κ1κn»])>
130   - <η-conversion for denoting terms; equivalent to AOT's axiom>
131   AOT_sem_lambda_eta: <[w ⊨ Π↓] ⇒ [w ⊨ [λν1...νn [Π] ν1...νn] = Π]>
132   - <β-conversion; equivalent to AOT's axiom>
133   AOT_sem_lambda_beta: <[w ⊨ [λν1...νn φ{ν1...νn}]↓] ⇒ [w ⊨ κ1...κn↓] ⇒
134     [w ⊨ [λν1...νn φ{ν1...νn}]κ1...κn] = [w ⊨ φ{κ1...κn}]>
135   - <Necessary and sufficient conditions for relations to denote. Equivalent
136     to a theorem of AOT and used to derive the base cases of denoting relations
137     (cqt.2).>
138   AOT_sem_lambda_denotes: <[w ⊨ [λν1...νn φ{ν1...νn}]↓] =
139     (∀ v κ1κn κ1'κn' . [v ⊨ κ1...κn↓] ∧ [v ⊨ κ1'...κn'↓] ∧
140     (∀ Π v . [v ⊨ Π↓] → [v ⊨ [Π] κ1...κn] = [v ⊨ [Π] κ1'...κn'])) →
141     [v ⊨ φ{κ1...κn}] = [v ⊨ φ{κ1'...κn'}]>
142   - <Equivalent to AOT's coexistence axiom.>
143   AOT_sem_lambda_coex: <[w ⊨ [λν1...νn φ{ν1...νn}]↓] ⇒
144     (∀ w κ1κn . [w ⊨ κ1...κn↓] → [w ⊨ φ{κ1...κn}] = [w ⊨ ψ{κ1...κn}] ⇒
145     [w ⊨ [λν1...νn ψ{ν1...νn}]↓]>
146   - <Only the unary case of the following should hold, but our specification
147     has to range over all types. We might move @{const AOT_exe} and
148     @{const AOT_lambda} to type classes in the future to solve this.>
149   AOT_sem_lambda_eq_prop_eq: <«[λν1...νn φ]» = «[λν1...νn ψ]» ⇒ φ = ψ>
150   - <The following is solely required for validating n-ary relation identity
151     and has the danger of implying artifactual theorems. Possibly avoidable
152     by moving @{const AOT_exe} and @{const AOT_lambda} to type classes.>
153   AOT_sem_exe_denoting: <[w ⊨ Π↓] ⇒ AOT_exe Π κs = Rep_rel Π κs>
154   - <The following is required for validating the base cases of denoting
155     relations (cqt.2). A version of this meta-logical identity will
156     become derivable in future versions of AOT, so this will ultimately not
157     result in artifactual theorems.>
158   AOT_sem_exe_equiv: <AOT_model_term_equiv x y ⇒ AOT_exe Π x = AOT_exe Π y>
159 proof -
160   have <∃ x :: <'a> . ¬AOT_model_denotes x>
161     by (rule exI[where x=<Abs_rel (λ x . εo w . True)>])
162     (meson AOT_model_denotes_rel.abs_eq AOT_model_nondenoting_ex
163       AOT_model_proposition_choice_simp)
164   define exe :: <<'a> ⇒ 'a ⇒ o> where
165     <exe ≡ λ Π κs . if AOT_model_denotes Π
166       then Rep_rel Π κs
167       else (εo w . False)>
168   define lambda :: <<'a⇒o> ⇒ <'a>> where
169     <lambda ≡ λ φ . if AOT_model_denotes (Abs_rel φ)
170       then (Abs_rel φ)
171       else
172         if (∀ κ κ' w . (AOT_model_denotes κ ∧ AOT_model_term_equiv κ κ') →
173           [w ⊨ «φ κ»] = [w ⊨ «φ κ'»])
174         then
175           Abs_rel (fix_irregular (λ x . if AOT_model_denotes x
176             then φ (SOME y . AOT_model_term_equiv x y)
177             else (εo w . False)))
178         else
179           Abs_rel φ
180   have fix_irregular_denoting_simp[simp]:
181     <fix_irregular (λx. if AOT_model_denotes x then φ x else ψ x) κ = φ κ>
182     if <AOT_model_denotes κ>
183     for κ :: 'a and φ ψ
184     by (simp add: that fix_irregular_denoting)
185   have denoting_eps_cong[cong]:
186     <[w ⊨ «φ (Eps (AOT_model_term_equiv κ))»] = [w ⊨ «φ κ»]>
187     if <AOT_model_denotes κ>

```



```

188   and <∀ κ κ'. AOT_model_denotes κ ∧ AOT_model_term_equiv κ κ' →
189         (∀ w. [w ⊨ «φ κ»] = [w ⊨ «φ κ'»])>
190   for w :: w and κ :: 'a and φ :: '<a⇒o>
191   using that AOT_model_term_equiv_eps(2) by blast
192   have exe_rep_rel: <[w ⊨ «exe Π κ₁κₙ»] = ([w ⊨ Π↓] ∧ [w ⊨ κ₁...κₙ↓] ∧
193         [w ⊨ «Rep_rel Π κ₁κₙ»])> for w Π κ₁κₙ
194   by (metis AOT_model_denotes_rel.rep_eq exe_def AOT_sem_denotes
195         AOT_model_proposition_choice_simp)
196   moreover have <[w ⊨ «Π»↓] ⇒ [w ⊨ «lambda (exe Π)»] = «Π»> for Π w
197   by (auto simp: Rep_rel_inverse lambda_def AOT_sem_denotes AOT_sem_eq
198         AOT_model_denotes_rel_def Abs_rel_inverse exe_def)
199   moreover have lambda_denotes_beta:
200     <[w ⊨ «exe (lambda φ) κ »] = [w ⊨ «φ κ»]>
201     if <[w ⊨ «lambda φ»↓]> and <[w ⊨ «κ»↓]>
202     for φ κ w
203   using that unfolding exe_def AOT_sem_denotes
204   by (auto simp: lambda_def Abs_rel_inverse split: if_split_asm)
205   moreover have lambda_denotes_simp:
206     <[w ⊨ «lambda φ»↓] = (∀ v κ₁κₙ κ₁'κₙ' . [v ⊨ κ₁...κₙ↓] ∧ [v ⊨ κ₁'...κₙ'↓] ∧
207     (∀ Π v . [v ⊨ Π↓] → [v ⊨ «exe Π κ₁κₙ»] = [v ⊨ «exe Π κ₁'κₙ'»]) →
208     [v ⊨ φ{κ₁...κₙ}] = [v ⊨ φ{κ₁'...κₙ'}])> for φ w
209   proof
210     assume <[w ⊨ «lambda φ»↓]>
211     hence <AOT_model_denotes (lambda φ)>
212       unfolding AOT_sem_denotes by simp
213     moreover have <[w ⊨ «φ κ»] ⇒ [w ⊨ «φ κ'»]>
214       and <[w ⊨ «φ κ'»] ⇒ [w ⊨ «φ κ»]>
215       if <AOT_model_denotes κ> and <AOT_model_term_equiv κ κ'>
216       for w κ κ'
217     by (metis (no_types, lifting) AOT_model_denotes_rel.abs_eq lambda_def
218         that calculation)+
219     ultimately show <∀ v κ₁κₙ κ₁'κₙ' . [v ⊨ κ₁...κₙ↓] ∧ [v ⊨ κ₁'...κₙ'↓] ∧
220     (∀ Π v . [v ⊨ Π↓] → [v ⊨ «exe Π κ₁κₙ»] = [v ⊨ «exe Π κ₁'κₙ'»]) →
221     [v ⊨ φ{κ₁...κₙ}] = [v ⊨ φ{κ₁'...κₙ'}]>
222     unfolding AOT_sem_denotes
223     by (metis (no_types) AOT_sem_denotes lambda_denotes_beta)
224   next
225     assume <∀ v κ₁κₙ κ₁'κₙ' . [v ⊨ κ₁...κₙ↓] ∧ [v ⊨ κ₁'...κₙ'↓] ∧
226     (∀ Π v . [v ⊨ Π↓] → [v ⊨ «exe Π κ₁κₙ»] = [v ⊨ «exe Π κ₁'κₙ'»]) →
227     [v ⊨ φ{κ₁...κₙ}] = [v ⊨ φ{κ₁'...κₙ'}]>
228     hence <[w ⊨ «φ κ»] = [w ⊨ «φ κ'»]>
229     if <AOT_model_denotes κ ∧ AOT_model_denotes κ' ∧ AOT_model_term_equiv κ κ'>
230     for w κ κ'
231     using that
232     by (auto simp: AOT_sem_denotes)
233     (meson AOT_model_term_equiv_rel_equiv AOT_sem_denotes exe_rep_rel)+
234     hence <[w ⊨ «φ κ»] = [w ⊨ «φ κ'»]>
235     if <AOT_model_denotes κ ∧ AOT_model_term_equiv κ κ'>
236     for w κ κ'
237     using that AOT_model_term_equiv_denotes by blast
238     hence <AOT_model_denotes (lambda φ)>
239     by (auto simp: lambda_def Abs_rel_inverse AOT_model_denotes_rel.abs_eq
240         AOT_model_irregular_equiv AOT_model_term_equiv_eps(3)
241         AOT_model_term_equiv_regular fix_irregular_def AOT_sem_denotes
242         AOT_model_term_equiv_denotes AOT_model_proposition_choice_simp
243         AOT_model_irregular_false
244         split: if_split_asm
245         intro: AOT_model_irregular_eqI)
246     thus <[w ⊨ «lambda φ»↓]>
247     by (simp add: AOT_sem_denotes)
248   qed
249   moreover have <[w ⊨ «lambda ψ»↓]>
250     if <[w ⊨ «lambda φ»↓]>

```

```

251   and <∀ w κ1κn . [w ⊨ κ1...κn] → [w ⊨ φ{κ1...κn}] = [w ⊨ ψ{κ1...κn}]>
252   for φ ψ w using that unfolding lambda_denotes_simp by auto
253   moreover have <[w ⊨ Π] ⇒ exe Π κs = Rep_rel Π κs> for Π κs w
254   by (simp add: exe_def AOT_sem_denotes)
255   moreover have <lambda (λx. p) = lambda (λx. q) ⇒ p = q> for p q
256   unfolding lambda_def
257   by (auto split: if_split_asm simp: Abs_rel_inject fix_irregular_def)
258   (meson AOT_model_irregular_nondenoting AOT_model_denoting_ex)+
259   moreover have <AOT_model_term_equiv x y ⇒ exe Π x = exe Π y> for x y Π
260   unfolding exe_def
261   by (meson AOT_model_denotes_rel.rep_eq)
262   note calculation = calculation this
263   show ?thesis
264     apply (safe intro!: exI[where x=exe] exI[where x=lambda])
265     using calculation apply simp_all
266     using lambda_denotes_simp by blast+
267 qed
268
269 lemma AOT_model_lambda_denotes:
270   <AOT_model_denotes (AOT_lambda φ) = (∀ v κ κ' .
271     AOT_model_denotes κ ∧ AOT_model_denotes κ' ∧ AOT_model_term_equiv κ κ' →
272     [v ⊨ «φ κ»] = [v ⊨ «φ κ'»])>
273 proof(safe)
274   fix v and κ κ' :: 'a
275   assume <AOT_model_denotes (AOT_lambda φ)>
276   hence 1: <AOT_model_denotes κ1κn ∧
277     AOT_model_denotes κ1'κn' ∧
278     (∀Π v . AOT_model_denotes Π → [v ⊨ [Π]κ1...κn] = [v ⊨ [Π]κ1'...κn']) →
279     [v ⊨ φ{κ1...κn}] = [v ⊨ φ{κ1'...κn'}]> for κ1κn κ1'κn' v
280   using AOT_sem_lambda_denotes[simplified AOT_sem_denotes] by blast
281 {
282   fix v and κ1κn κ1'κn' :: 'a
283   assume d: <AOT_model_denotes κ1κn ∧ AOT_model_denotes κ1'κn' ∧
284     AOT_model_term_equiv κ1κn κ1'κn'>
285   hence <∀Π w . AOT_model_denotes Π → [w ⊨ [Π]κ1...κn] = [w ⊨ [Π]κ1'...κn']>
286     by (metis AOT_sem_exe_equiv)
287   hence <[v ⊨ φ{κ1...κn}] = [v ⊨ φ{κ1'...κn'}]> using d 1 by auto
288 }
289 moreover assume <AOT_model_denotes κ>
290 moreover assume <AOT_model_denotes κ'>
291 moreover assume <AOT_model_term_equiv κ κ'>
292 ultimately show <[v ⊨ «φ κ»] ⇒ [v ⊨ «φ κ'»]>
293   and <[v ⊨ «φ κ'»] ⇒ [v ⊨ «φ κ»>
294   by auto
295 next
296   assume 0: <∀ v κ κ' . AOT_model_denotes κ ∧ AOT_model_denotes κ' ∧
297     AOT_model_term_equiv κ κ' → [v ⊨ «φ κ»] = [v ⊨ «φ κ'»]>
298   {
299     fix κ1κn κ1'κn' :: 'a
300     assume den: <AOT_model_denotes κ1κn>
301     moreover assume den': <AOT_model_denotes κ1'κn'>
302     assume <∀Π v . AOT_model_denotes Π →
303       [v ⊨ [Π]κ1...κn] = [v ⊨ [Π]κ1'...κn']>
304     hence <∀Π v . AOT_model_denotes Π →
305       [v ⊨ «Rep_rel Π κ1κn»] = [v ⊨ «Rep_rel Π κ1'κn'»]>
306       by (simp add: AOT_sem_denotes AOT_sem_exe den den')
307     hence "AOT_model_term_equiv κ1κn κ1'κn'"
308       unfolding AOT_model_term_equiv_rel_equiv[OF den, OF den']
309       by argo
310     hence <[v ⊨ φ{κ1...κn}] = [v ⊨ φ{κ1'...κn'}]> for v
311     using den den' 0 by blast
312   }
313   thus <AOT_model_denotes (AOT_lambda φ)>

```

```

314     unfolding AOT_sem_lambda_denotes[simplified AOT_sem_denotes]
315     by blast
316 qed
317
318 specification (AOT_lambda0)
319   AOT_sem_lambda0: "AOT_lambda0  $\varphi = \varphi$ "
320   by (rule exI[where x=< $\lambda x. x$ >]) simp
321
322 specification(AOT_concrete)
323   AOT_sem_concrete: <[ $w \models [E!] \kappa$ ] =
324     AOT_model_concrete  $w \kappa$ >
325   by (rule exI[where x=<AOT_var_of_term (Abs_rel
326     ( $\lambda x . \varepsilon_o w . AOT_model_concrete w x$ ))>];
327     subst AOT_var_of_term_inverse)
328   (auto simp: AOT_model_unary_regular AOT_model_concrete_denotes
329     AOT_model_concrete_equiv AOT_model_regular_ $\kappa$ _def
330     AOT_model_proposition_choice_simp AOT_sem_exe Abs_rel_inverse
331     AOT_model_denotes_rel_def AOT_sem_denotes)
332
333 lemma AOT_sem_equiv_defI:
334   assumes < $\bigwedge v . [v \models \varphi] \implies [v \models \psi]$ >
335     and < $\bigwedge v . [v \models \psi] \implies [v \models \varphi]$ >
336     shows <AOT_model_equiv_def  $\varphi \psi$ >
337     using AOT_model_equiv_def assms by blast
338
339 lemma AOT_sem_id_defI:
340   assumes < $\bigwedge \alpha v . [v \models \langle\sigma \alpha\rangle\downarrow] \implies [v \models \langle\tau \alpha\rangle = \langle\sigma \alpha\rangle]$ >
341   assumes < $\bigwedge \alpha v . \neg[v \models \langle\sigma \alpha\rangle\downarrow] \implies [v \models \neg\langle\tau \alpha\rangle\downarrow]$ >
342   shows <AOT_model_id_def  $\tau \sigma$ >
343   using assms
344   unfolding AOT_sem_denotes AOT_sem_eq AOT_sem_not
345   using AOT_model_id_def[THEN iffD2]
346   by metis
347
348 lemma AOT_sem_id_def2I:
349   assumes < $\bigwedge \alpha \beta v . [v \models \langle\sigma \alpha \beta\rangle\downarrow] \implies [v \models \langle\tau \alpha \beta\rangle = \langle\sigma \alpha \beta\rangle]$ >
350   assumes < $\bigwedge \alpha \beta v . \neg[v \models \langle\sigma \alpha \beta\rangle\downarrow] \implies [v \models \neg\langle\tau \alpha \beta\rangle\downarrow]$ >
351   shows <AOT_model_id_def (case_prod  $\tau$ ) (case_prod  $\sigma$ )>
352   apply (rule AOT_sem_id_defI)
353   using assms by (auto simp: AOT_sem_conj AOT_sem_not AOT_sem_eq AOT_sem_denotes
354     AOT_model_denotes_prod_def)
355
356 lemma AOT_sem_id_defE1:
357   assumes <AOT_model_id_def  $\tau \sigma$ >
358     and <[ $v \models \langle\sigma \alpha\rangle\downarrow]$ >
359     shows <[ $v \models \langle\tau \alpha\rangle = \langle\sigma \alpha\rangle]$ >
360   using assms by (simp add: AOT_model_id_def AOT_sem_denotes AOT_sem_eq)
361
362 lemma AOT_sem_id_defE2:
363   assumes <AOT_model_id_def  $\tau \sigma$ >
364     and < $\neg[v \models \langle\sigma \alpha\rangle\downarrow]$ >
365     shows < $\neg[v \models \langle\tau \alpha\rangle\downarrow]$ >
366   using assms by (simp add: AOT_model_id_def AOT_sem_denotes AOT_sem_eq)
367
368 lemma AOT_sem_id_defOI:
369   assumes < $\bigwedge v . [v \models \sigma\downarrow] \implies [v \models \tau = \sigma]$ >
370     and < $\bigwedge v . \neg[v \models \sigma\downarrow] \implies [v \models \neg\tau\downarrow]$ >
371   shows <AOT_model_id_def (case_unit  $\tau$ ) (case_unit  $\sigma$ )>
372   apply (rule AOT_sem_id_defI)
373   using assms
374   by (simp_all add: AOT_sem_conj AOT_sem_eq AOT_sem_not AOT_sem_denotes
375     AOT_model_denotes_unit_def case_unit_Unity)
376

```

```

377 lemma AOT_sem_id_def0E1:
378   assumes <AOT_model_id_def (case_unit  $\tau$ ) (case_unit  $\sigma$ )>
379     and <[v  $\models$   $\sigma$ ] $\downarrow$ >
380     shows <[v  $\models$   $\tau = \sigma$ >
381   by (metis (full_types) AOT_sem_id_defE1 assms(1) assms(2) case_unit_Unity)
382
383 lemma AOT_sem_id_def0E2:
384   assumes <AOT_model_id_def (case_unit  $\tau$ ) (case_unit  $\sigma$ )>
385     and < $\neg$ [v  $\models$   $\sigma$ ] $\downarrow$ >
386     shows < $\neg$ [v  $\models$   $\tau$ ] $\downarrow$ >
387   by (metis AOT_sem_id_defE2 assms(1) assms(2) case_unit_Unity)
388
389 lemma AOT_sem_id_def0E3:
390   assumes <AOT_model_id_def (case_unit  $\tau$ ) (case_unit  $\sigma$ )>
391     and <[v  $\models$   $\sigma$ ] $\downarrow$ >
392     shows <[v  $\models$   $\tau$ ] $\downarrow$ >
393   using AOT_sem_id_def0E1[OF assms]
394   by (simp add: AOT_sem_eq AOT_sem_denotes)
395
396 lemma AOT_sem_ordinary_def_denotes: <[w  $\models$  [ $\lambda x \diamond [E!]x$ ] $\downarrow$ >
397   unfolding AOT_sem_denotes AOT_model_lambda_denotes
398   by (auto simp: AOT_sem_dia AOT_model_concrete_equiv
399     AOT_sem_concrete AOT_sem_denotes)
400 lemma AOT_sem_abstract_def_denotes: <[w  $\models$  [ $\lambda x \neg \diamond [E!]x$ ] $\downarrow$ >
401   unfolding AOT_sem_denotes AOT_model_lambda_denotes
402   by (auto simp: AOT_sem_dia AOT_model_concrete_equiv
403     AOT_sem_concrete AOT_sem_denotes AOT_sem_not)
404
405 text<Relation identity is constructed using an auxiliary abstract projection
406   mechanism with suitable instantiations for @ $\{\text{typ } \kappa\}$  and products.>
407 class AOT_RelationProjection =
408   fixes AOT_sem_proj_id :: <'a::AOT_IndividualTerm  $\Rightarrow$  ('a  $\Rightarrow$  o)  $\Rightarrow$  ('a  $\Rightarrow$  o)  $\Rightarrow$  o>
409   assumes AOT_sem_proj_id_prop:
410     <[v  $\models$   $\Pi = \Pi'$ ] =
411     [v  $\models$   $\Pi$ ] $\downarrow$  &  $\Pi'$  $\downarrow$  &  $\forall \alpha$  (<AOT_sem_proj_id  $\alpha$  ( $\lambda \tau . \ll[\Pi]\tau\gg$ ) ( $\lambda \tau . \ll[\Pi']\tau\gg$ )>>)]>
412     and AOT_sem_proj_id_refl:
413     <[v  $\models$   $\tau$ ] $\downarrow$   $\implies$  [v  $\models$  [ $\lambda \nu_1 \dots \nu_n \varphi\{\nu_1 \dots \nu_n\}$ ] = [ $\lambda \nu_1 \dots \nu_n \varphi\{\nu_1 \dots \nu_n\}$ ]]  $\implies$ 
414     [v  $\models$  <AOT_sem_proj_id  $\tau \varphi \varphi$ >>
415
416 class AOT_UnaryRelationProjection = AOT_RelationProjection +
417   assumes AOT_sem_unary_proj_id:
418     <AOT_sem_proj_id  $\kappa \varphi \psi = \ll[\lambda \nu_1 \dots \nu_n \varphi\{\nu_1 \dots \nu_n\}] = [\lambda \nu_1 \dots \nu_n \psi\{\nu_1 \dots \nu_n\}]\gg$ >
419
420 instantiation  $\kappa$  :: AOT_UnaryRelationProjection
421 begin
422 definition AOT_sem_proj_id_ $\kappa$  :: < $\kappa \Rightarrow (\kappa \Rightarrow o) \Rightarrow (\kappa \Rightarrow o) \Rightarrow o$ > where
423   <AOT_sem_proj_id_ $\kappa$   $\kappa \varphi \psi = \ll[\lambda z \varphi\{z\}] = [\lambda z \psi\{z\}]\gg$ >
424 instance proof
425   show <[v  $\models$   $\Pi = \Pi'$ ] =
426     [v  $\models$   $\Pi$ ] $\downarrow$  &  $\Pi'$  $\downarrow$  &  $\forall \alpha$  (<AOT_sem_proj_id  $\alpha$  ( $\lambda \tau . \ll[\Pi]\tau\gg$ ) ( $\lambda \tau . \ll[\Pi']\tau\gg$ )>>)]>
427   for v and  $\Pi \Pi'$  :: << $\kappa$ >>
428   unfolding AOT_sem_proj_id_ $\kappa$ _def
429   by (simp add: AOT_sem_eq AOT_sem_conj AOT_sem_denotes AOT_sem_forall)
430   (metis AOT_sem_denotes AOT_model_denoting_ex AOT_sem_eq AOT_sem_lambda_eta)
431 next
432   show <AOT_sem_proj_id  $\kappa \varphi \psi = \ll[\lambda \nu_1 \dots \nu_n \varphi\{\nu_1 \dots \nu_n\}] = [\lambda \nu_1 \dots \nu_n \psi\{\nu_1 \dots \nu_n\}]\gg$ >
433   for  $\kappa$  ::  $\kappa$  and  $\varphi \psi$ 
434   unfolding AOT_sem_proj_id_ $\kappa$ _def ..
435 next
436   show <[v  $\models$  <AOT_sem_proj_id  $\tau \varphi \varphi$ >>
437     if <[v  $\models$   $\tau$ ] $\downarrow$ > and <[v  $\models$  [ $\lambda \nu_1 \dots \nu_n \varphi\{\nu_1 \dots \nu_n\}] = [\lambda \nu_1 \dots \nu_n \varphi\{\nu_1 \dots \nu_n\}]]$ >
438     for  $\tau$  ::  $\kappa$  and v  $\varphi$ 
439   using that by (simp add: AOT_sem_proj_id_ $\kappa$ _def AOT_sem_eq)

```

```

440 qed
441 end
442
443 instantiation prod ::
444   ("{AOT_UnaryRelationProjection, AOT_UnaryIndividualTerm}", AOT_RelationProjection)
445   AOT_RelationProjection
446 begin
447 definition AOT_sem_proj_id_prod :: <'a×'b ⇒ ('a×'b ⇒ o) ⇒ ('a×'b ⇒ o) ⇒ o> where
448   <AOT_sem_proj_id_prod ≡ λ (x,y) φ ψ . «[λz «φ (z,y)»] = [λz «ψ (z,y)»] &
449     «AOT_sem_proj_id y (λ a . φ (x,a)) (λ a . ψ (x,a))»»>
450 instance proof
451   text<This is the main proof that allows to derive the definition of n-ary
452     relation identity. We need to show that our defined projection identity
453     implies relation identity for relations on pairs of individual terms.>
454   fix v and Π Π' :: <<'a×'b>>
455   have AOT_meta_proj_denotes1: <AOT_model_denotes (Abs_rel (λz. AOT_exe Π (z, β)))>
456     if <AOT_model_denotes Π> for Π :: <<'a×'b>> and β
457     using that unfolding AOT_model_denotes_rel.rep_eq
458     apply (auto simp: Abs_rel_inverse AOT_meta_prod_equivI(2) AOT_sem_denotes
459           that intro!: AOT_sem_exe_equiv)
460     apply (metis AOT_model_denotes_prod_def AOT_sem_exe case_proxD)
461     using AOT_model_unary_regular by blast
462 {
463   fix κ :: 'a and Π :: <<'a×'b>>
464   assume Π_denotes: <AOT_model_denotes Π>
465   assume α_denotes: <AOT_model_denotes κ>
466   hence <AOT_exe Π (κ, x) = AOT_exe Π (κ, y)>
467     if <AOT_model_term_equiv x y> for x y :: 'b
468     by (simp add: AOT_meta_prod_equivI(1) AOT_sem_exe_equiv that)
469   moreover have <AOT_model_denotes κ1'κn'>
470     if <[w ⊨ [Π] κ κ1'...κn']> for w κ1'κn'
471     by (metis that AOT_model_denotes_prod_def AOT_sem_exe
472         AOT_sem_denotes case_proxD)
473   moreover {
474     fix x :: 'b
475     assume x_irregular: <¬AOT_model_regular x>
476     hence prod_irregular: <¬AOT_model_regular (κ, x)>
477       by (metis (no_types, lifting) AOT_model_irregular_nondenoting
478         AOT_model_regular_prod_def case_proxD)
479     hence <(¬AOT_model_denotes κ ∨ ¬AOT_model_regular x) ∧
480       (AOT_model_denotes κ ∨ ¬AOT_model_denotes x)>
481       unfolding AOT_model_regular_prod_def by blast
482     hence x_nonden: <¬AOT_model_regular x>
483       by (simp add: α_denotes)
484     have <Rep_rel Π (κ, x) = AOT_model_irregular (Rep_rel Π) (κ, x)>
485       using AOT_model_denotes_rel.rep_eq Π_denotes prod_irregular by blast
486     moreover have <AOT_model_irregular (Rep_rel Π) (κ, x) =
487       AOT_model_irregular (λz. Rep_rel Π (κ, z)) x>
488       using Π_denotes x_irregular prod_irregular x_nonden
489       using AOT_model_irregular_prod_generic
490       apply (induct arbitrary: Π x rule: AOT_model_irregular_prod.induct)
491       by (auto simp: α_denotes AOT_model_irregular_nondenoting
492           AOT_model_regular_prod_def AOT_meta_prod_equivI(2)
493           AOT_model_denotes_rel.rep_eq AOT_model_term_equiv_eps(1)
494           intro!: AOT_model_irregular_eqI)
495     ultimately have
496       <AOT_exe Π (κ, x) = AOT_model_irregular (λz. AOT_exe Π (κ, z)) x>
497       unfolding AOT_sem_exe_denoting[simplified AOT_sem_denotes, OF Π_denotes]
498       by auto
499   }
500   ultimately have <AOT_model_denotes (Abs_rel (λz. AOT_exe Π (κ, z)))>
501     by (simp add: Abs_rel_inverse AOT_model_denotes_rel.rep_eq)
502 } note AOT_meta_proj_denotes2 = this

```

```

503 {
504   fix  $\kappa_1' \kappa_n'$  :: 'b and  $\Pi$  :: <<'a × 'b>>
505   assume  $\Pi$ _denotes: <AOT_model_denotes  $\Pi$ >
506   assume  $\beta$ _denotes: <AOT_model_denotes  $\kappa_1' \kappa_n'$ >
507   hence <AOT_exe  $\Pi$  (x,  $\kappa_1' \kappa_n'$ ) = AOT_exe  $\Pi$  (y,  $\kappa_1' \kappa_n'$ )>
508     if <AOT_model_term_equiv x y> for x y :: 'a
509     by (simp add: AOT_meta_prod_equivI(2) AOT_sem_exe_equiv that)
510   moreover have <AOT_model_denotes  $\kappa$ >
511     if <[w ⊨ [[ $\Pi$ ] $\kappa$   $\kappa_1'$  ...  $\kappa_n'$ ]]> for w  $\kappa$ 
512     by (metis that AOT_model_denotes_prod_def AOT_sem_exe
513         AOT_sem_denotes case_prodD)
514   moreover {
515     fix x :: 'a
516     assume <¬AOT_model_regular x>
517     hence <False>
518       using AOT_model_unary_regular by blast
519     hence
520       <AOT_exe  $\Pi$  (x,  $\kappa_1' \kappa_n'$ ) = AOT_model_irregular ( $\lambda z$ . AOT_exe  $\Pi$  (z,  $\kappa_1' \kappa_n'$ )) x>
521       unfolding AOT_sem_exe_denoting[simplified AOT_sem_denotes, OF  $\Pi$ _denotes]
522       by auto
523   }
524   ultimately have <AOT_model_denotes (Abs_rel ( $\lambda z$ . AOT_exe  $\Pi$  (z,  $\kappa_1' \kappa_n'$ )))>
525     by (simp add: Abs_rel_inverse AOT_model_denotes_rel.rep_eq)
526 } note AOT_meta_proj_denotes1 = this
527 {
528   assume  $\Pi$ _denotes: <AOT_model_denotes  $\Pi$ >
529   assume  $\Pi'$ _denotes: <AOT_model_denotes  $\Pi'$ >
530   have  $\Pi$ _proj2_den: <AOT_model_denotes (Abs_rel ( $\lambda z$ . Rep_rel  $\Pi$  ( $\alpha$ , z)))>
531     if <AOT_model_denotes  $\alpha$ > for  $\alpha$ 
532     using that AOT_meta_proj_denotes2[OF  $\Pi$ _denotes]
533     AOT_sem_exe_denoting[simplified AOT_sem_denotes, OF  $\Pi$ _denotes] by simp
534   have  $\Pi'$ _proj2_den: <AOT_model_denotes (Abs_rel ( $\lambda z$ . Rep_rel  $\Pi'$  ( $\alpha$ , z)))>
535     if <AOT_model_denotes  $\alpha$ > for  $\alpha$ 
536     using that AOT_meta_proj_denotes2[OF  $\Pi'$ _denotes]
537     AOT_sem_exe_denoting[simplified AOT_sem_denotes, OF  $\Pi'$ _denotes] by simp
538   have  $\Pi$ _proj1_den: <AOT_model_denotes (Abs_rel ( $\lambda z$ . Rep_rel  $\Pi$  (z,  $\alpha$ )))>
539     if <AOT_model_denotes  $\alpha$ > for  $\alpha$ 
540     using that AOT_meta_proj_denotes1[OF  $\Pi$ _denotes]
541     AOT_sem_exe_denoting[simplified AOT_sem_denotes, OF  $\Pi$ _denotes] by simp
542   have  $\Pi'$ _proj1_den: <AOT_model_denotes (Abs_rel ( $\lambda z$ . Rep_rel  $\Pi'$  (z,  $\alpha$ )))>
543     if <AOT_model_denotes  $\alpha$ > for  $\alpha$ 
544     using that AOT_meta_proj_denotes1[OF  $\Pi'$ _denotes]
545     AOT_sem_exe_denoting[simplified AOT_sem_denotes, OF  $\Pi'$ _denotes] by simp
546   {
547     fix  $\kappa$  :: 'a and  $\kappa_1' \kappa_n'$  :: 'b
548     assume < $\Pi = \Pi'$ >
549     assume <AOT_model_denotes ( $\kappa, \kappa_1' \kappa_n'$ )>
550     hence <AOT_model_denotes  $\kappa$ > and beta_denotes: <AOT_model_denotes  $\kappa_1' \kappa_n'$ >
551       by (auto simp: AOT_model_denotes_prod_def)
552     have <AOT_model_denotes  $\ll[\lambda z$  [[ $\Pi$ ] $z$   $\kappa_1'$  ...  $\kappa_n'$ ]]>>
553       by (rule AOT_model_lambda_denotes[THEN iffD2])
554       (metis AOT_sem_exe_denoting AOT_meta_prod_equivI(2)
555           AOT_model_denotes_rel.rep_eq AOT_sem_denotes
556           AOT_sem_exe_denoting  $\Pi$ _denotes)
557     moreover have  $\ll[\lambda z$  [[ $\Pi$ ] $z$   $\kappa_1'$  ...  $\kappa_n'$ ]] =  $\ll[\lambda z$  [[ $\Pi'$ ] $z$   $\kappa_1'$  ...  $\kappa_n'$ ]]>>
558       by (simp add: < $\Pi = \Pi'$ >)
559     moreover have <[v ⊨  $\ll$ AOT_sem_proj_id  $\kappa_1' \kappa_n'$  ( $\lambda \kappa_1' \kappa_n'$ .  $\ll$ [[ $\Pi$ ] $\kappa$   $\kappa_1'$  ...  $\kappa_n'$ ]]
560         ( $\lambda \kappa_1' \kappa_n'$ .  $\ll$ [[ $\Pi'$ ] $\kappa$   $\kappa_1'$  ...  $\kappa_n'$ ]]>>)>
561       unfolding < $\Pi = \Pi'$ > using beta_denotes
562       by (rule AOT_sem_proj_id_refl[unfolded AOT_sem_denotes];
563           simp add: AOT_sem_denotes AOT_sem_eq AOT_model_lambda_denotes)
564     (metis AOT_meta_prod_equivI(1) AOT_model_denotes_rel.rep_eq
565         AOT_sem_exe AOT_sem_exe_denoting  $\Pi'$ _denotes)

```

```

566 ultimately have <[v | = «AOT_sem_proj_id (κ, κ1'κn') (λ κ1κn . «[Π] κ1...κn»)
567                                     (λ κ1κn . «[Π'] κ1...κn»)»>
568   unfolding AOT_sem_proj_id_prod_def
569   by (simp add: AOT_sem_denotes AOT_sem_conj AOT_sem_eq)
570 }
571 moreover {
572   assume <∧ α . AOT_model_denotes α ⇒
573     [v | = «AOT_sem_proj_id α (λ κ1κn . «[Π] κ1...κn») (λ κ1κn . «[Π'] κ1...κn»)»>
574   hence 0: <AOT_model_denotes κ ⇒ AOT_model_denotes κ1'κn' ⇒
575     AOT_model_denotes «[λz [Π]z κ1'...κn']» ∧
576     AOT_model_denotes «[λz [Π']z κ1'...κn']» ∧
577     «[λz [Π]z κ1'...κn']» = «[λz [Π']z κ1'...κn']» ∧
578     [v | = «AOT_sem_proj_id κ1'κn' (λκ1κn. «[Π]κ κ1...κn»)
579                                     (λκ1κn. «[Π']κ κ1...κn»)»> for κ κ1'κn'
580   unfolding AOT_sem_proj_id_prod_def
581   by (auto simp: AOT_sem_denotes AOT_sem_conj AOT_sem_eq
582       AOT_model_denotes_prod_def)
583   obtain αden :: 'a and βden :: 'b where
584     αden: <AOT_model_denotes αden> and βden: <AOT_model_denotes βden>
585     using AOT_model_denoting_ex by metis
586   {
587     fix κ :: 'a
588     assume αdenotes: <AOT_model_denotes κ>
589     have 1: <[v | = «AOT_sem_proj_id κ1'κn' (λκ1'κn'. «[Π]κ κ1'...κn')
590                                     (λκ1'κn'. «[Π']κ κ1'...κn')»>
591       if <AOT_model_denotes κ1'κn'> for κ1'κn'
592       using that 0 using αdenotes by blast
593     hence <[v | = «AOT_sem_proj_id β (λz. Rep_rel Π (κ, z))
594                                     (λz. Rep_rel Π' (κ, z))»>
595       if <AOT_model_denotes β> for β
596       using that
597       unfolding AOT_sem_exe_denoting[simplified AOT_sem_denotes, OF Π_denotes]
598       AOT_sem_exe_denoting[simplified AOT_sem_denotes, OF Π'_denotes]
599       by blast
600     hence <Abs_rel (λz. Rep_rel Π (κ, z)) = Abs_rel (λz. Rep_rel Π' (κ, z))>
601       using AOT_sem_proj_id_prop[of v <Abs_rel (λz. Rep_rel Π (κ, z))>
602                                     <Abs_rel (λz. Rep_rel Π' (κ, z))>,
603       simplified AOT_sem_eq AOT_sem_conj AOT_sem_forall
604       AOT_sem_denotes, THEN iffD2]
605       Π_proj2_den[OF αdenotes] Π'_proj2_den[OF αdenotes]
606     unfolding AOT_sem_exe_denoting[simplified AOT_sem_denotes, OF Π_denotes]
607       AOT_sem_exe_denoting[simplified AOT_sem_denotes,
608       OF Π_proj2_den[OF αdenotes]]
609       AOT_sem_exe_denoting[simplified AOT_sem_denotes,
610       OF Π'_proj2_den[OF αdenotes]]
611     by (metis Abs_rel_inverse UNIV_I)
612     hence "Rep_rel Π (κ, β) = Rep_rel Π' (κ, β)" for β
613     by (simp add: Abs_rel_inject[simplified]) meson
614   } note αdenotes = this
615   {
616     fix κ1'κn' :: 'b
617     assume βden: <AOT_model_denotes κ1'κn'>
618     have 1: «[λz [Π]z κ1'...κn']» = «[λz [Π']z κ1'...κn']»
619     using 0 βden AOT_model_denoting_ex by blast
620     hence <Abs_rel (λz. Rep_rel Π (z, κ1'κn') =
621       Abs_rel (λz. Rep_rel Π' (z, κ1'κn')> (is <?a = ?b>)
622     apply (safe intro!: AOT_sem_proj_id_prop[of v <?a> <?b>,
623       simplified AOT_sem_eq AOT_sem_conj AOT_sem_forall
624       AOT_sem_denotes, THEN iffD2, THEN conjunct2, THEN conjunct2]
625       Π_proj1_den[OF βden] Π'_proj1_den[OF βden])
626     unfolding AOT_sem_exe_denoting[simplified AOT_sem_denotes, OF Π_denotes]
627       AOT_sem_exe_denoting[simplified AOT_sem_denotes, OF Π'_denotes]
628       AOT_sem_exe_denoting[simplified AOT_sem_denotes,

```



```

629                                     OF  $\Pi\_proj1\_den$ [OF  $\beta den$ ]]
630       AOT_sem_exe_denoting[simplified AOT_sem_denotes,
631                             OF  $\Pi'\_proj1\_den$ [OF  $\beta den$ ]]
632     by (subst (0 1) Abs_rel_inverse; simp?)
633       (metis (no_types, lifting) AOT_model_denotes_rel.abs_eq
634         AOT_model_lambda_denotes AOT_sem_denotes AOT_sem_eq
635         AOT_sem_unary_proj_id  $\Pi\_proj1\_den$ [OF  $\beta den$ ])
636   hence <Rep_rel  $\Pi$  ( $x, \kappa_1' \kappa_n'$ ) = Rep_rel  $\Pi'$  ( $x, \kappa_1' \kappa_n'$ )> for x
637     by (simp add: Abs_rel_inject)
638     metis
639 } note  $\beta denotes = this$ 
640 {
641   fix  $\alpha :: 'a$  and  $\beta :: 'b$ 
642   assume <AOT_model_regular ( $\alpha, \beta$ )>
643   moreover {
644     assume <AOT_model_denotes  $\alpha \wedge$  AOT_model_regular  $\beta$ >
645     hence <Rep_rel  $\Pi$  ( $\alpha, \beta$ ) = Rep_rel  $\Pi'$  ( $\alpha, \beta$ )>
646       using  $\alpha denotes$  by presburger
647   }
648   moreover {
649     assume < $\neg$ AOT_model_denotes  $\alpha \wedge$  AOT_model_denotes  $\beta$ >
650     hence <Rep_rel  $\Pi$  ( $\alpha, \beta$ ) = Rep_rel  $\Pi'$  ( $\alpha, \beta$ )>
651       by (simp add:  $\beta denotes$ )
652   }
653   ultimately have <Rep_rel  $\Pi$  ( $\alpha, \beta$ ) = Rep_rel  $\Pi'$  ( $\alpha, \beta$ )>
654     by (metis (no_types, lifting) AOT_model_regular_prod_def case_prodD)
655 }
656 hence <Rep_rel  $\Pi =$  Rep_rel  $\Pi'$ >
657   using  $\Pi denotes$ [unfolded AOT_model_denotes_rel.rep_eq,
658     THEN conjunct2, THEN conjunct2, THEN spec, THEN mp]
659   using  $\Pi' denotes$ [unfolded AOT_model_denotes_rel.rep_eq,
660     THEN conjunct2, THEN conjunct2, THEN spec, THEN mp]
661   using AOT_model_irregular_eqI[of <Rep_rel  $\Pi$ > <Rep_rel  $\Pi'$ > <(_,_)>]
662   using AOT_model_irregular_nondenoting by fastforce
663   hence < $\Pi = \Pi'$ >
664     by (rule Rep_rel_inject[THEN iffD1])
665 }
666 ultimately have < $\Pi = \Pi' = (\forall \kappa . AOT\_model\_denotes \kappa \longrightarrow$ 
667   [ $v \models \ll AOT\_sem\_proj\_id \kappa (\lambda \kappa_1 \kappa_n . \ll [\Pi] \kappa_1 \dots \kappa_n \gg)$ 
668     ( $\lambda \kappa_1 \kappa_n . \ll [\Pi'] \kappa_1 \dots \kappa_n \gg)$ )>>
669   by auto
670 }
671 thus <[ $v \models \Pi = \Pi'$ ] = [ $v \models \Pi \downarrow \& \Pi' \downarrow \&$ 
672    $\forall \alpha (\ll AOT\_sem\_proj\_id \alpha (\lambda \kappa_1 \kappa_n . \ll [\Pi] \kappa_1 \dots \kappa_n \gg) (\lambda \kappa_1 \kappa_n . \ll [\Pi'] \kappa_1 \dots \kappa_n \gg)$ )>>]
673   by (auto simp: AOT_sem_eq AOT_sem_denotes AOT_sem_forall AOT_sem_conj)
674 next
675   fix v and  $\varphi :: \langle 'a \times 'b \Rightarrow o \rangle$  and  $\tau :: \langle 'a \times 'b \rangle$ 
676   assume <[ $v \models \tau \downarrow$ ]>
677   moreover assume <[ $v \models [\lambda z_1 \dots z_n \ll \varphi z_1 z_n \gg] = [\lambda z_1 \dots z_n \ll \varphi z_1 z_n \gg]$ ]>
678   ultimately show <[ $v \models \ll AOT\_sem\_proj\_id \tau \varphi \varphi \gg$ ]>
679     unfolding AOT_sem_proj_id_prod_def
680     using AOT_sem_proj_id_refl[of v "snd  $\tau$ " " $\lambda b. \varphi$  (fst  $\tau, b)$ "]
681     by (auto simp: AOT_sem_eq AOT_sem_conj AOT_sem_denotes
682       AOT_model_denotes_prod_def AOT_model_lambda_denotes
683       AOT_meta_prod_equivI)
684 qed
685 end
686
687 text<Sanity-check to verify that n-ary relation identity follows.>
688 lemma <[ $v \models \Pi = \Pi'$ ] = [ $v \models \Pi \downarrow \& \Pi' \downarrow \& \forall x \forall y ([\lambda z [\Pi] z y] = [\lambda z [\Pi'] z y] \&$ 
689    $[\lambda z [\Pi] x z] = [\lambda z [\Pi'] x z])$ ]>
690   for  $\Pi :: \langle \langle \kappa \times \kappa \rangle \rangle$ 
691   by (auto simp: AOT_sem_proj_id_prop[of v  $\Pi \Pi'$ ] AOT_sem_proj_id_prod_def

```



```

692         AOT_sem_conj AOT_sem_denotes AOT_sem_forall AOT_sem_unary_proj_id
693         AOT_model_denotes_prod_def)
694 lemma <[v ⊨ Π = Π'] = [v ⊨ Π↓ & Π'↓ & ∀x1∀x2∀x3 (
695   [λz [Π]z x2 x3] = [λz [Π']z x2 x3] &
696   [λz [Π]x1 z x3] = [λz [Π']x1 z x3] &
697   [λz [Π]x1 x2 z] = [λz [Π']x1 x2 z])]>
698   for Π :: <<κ×κ×κ>>
699   by (auto simp: AOT_sem_proj_id_prop[of v Π Π'] AOT_sem_proj_id_prod_def
700       AOT_sem_conj AOT_sem_denotes AOT_sem_forall AOT_sem_unary_proj_id
701       AOT_model_denotes_prod_def)
702 lemma <[v ⊨ Π = Π'] = [v ⊨ Π↓ & Π'↓ & ∀x1∀x2∀x3∀x4 (
703   [λz [Π]z x2 x3 x4] = [λz [Π']z x2 x3 x4] &
704   [λz [Π]x1 z x3 x4] = [λz [Π']x1 z x3 x4] &
705   [λz [Π]x1 x2 z x4] = [λz [Π']x1 x2 z x4] &
706   [λz [Π]x1 x2 x3 z] = [λz [Π']x1 x2 x3 z])]>
707   for Π :: <<κ×κ×κ×κ>>
708   by (auto simp: AOT_sem_proj_id_prop[of v Π Π'] AOT_sem_proj_id_prod_def
709       AOT_sem_conj AOT_sem_denotes AOT_sem_forall AOT_sem_unary_proj_id
710       AOT_model_denotes_prod_def)
711
712 text<n-ary Encoding is constructed using a similar mechanism as n-ary relation
713   identity using an auxiliary notion of projection-encoding.>
714 class AOT_Enc =
715   fixes AOT_enc :: <'a ⇒ <'a::AOT_IndividualTerm> ⇒ o>
716   and AOT_proj_enc :: <'a ⇒ ('a ⇒ o) ⇒ o>
717   assumes AOT_sem_enc_denotes:
718     <[v ⊨ «AOT_enc κ1κn Π»] ⇒ [v ⊨ κ1...κn↓] ∧ [v ⊨ Π↓]>
719   assumes AOT_sem_enc_proj_enc:
720     <[v ⊨ «AOT_enc κ1κn Π»] =
721     [v ⊨ Π↓ & «AOT_proj_enc κ1κn (λ κ1κn. «[Π]κ1...κn»)]>
722   assumes AOT_sem_proj_enc_denotes:
723     <[v ⊨ «AOT_proj_enc κ1κn φ»] ⇒ [v ⊨ κ1...κn↓]>
724   assumes AOT_sem_enc_nec:
725     <[v ⊨ «AOT_enc κ1κn Π»] ⇒ [w ⊨ «AOT_enc κ1κn Π»]>
726   assumes AOT_sem_proj_enc_nec:
727     <[v ⊨ «AOT_proj_enc κ1κn φ»] ⇒ [w ⊨ «AOT_proj_enc κ1κn φ»]>
728   assumes AOT_sem_universal_encoder:
729     <∃ κ1κn. [v ⊨ κ1...κn↓] ∧ (∀ Π . [v ⊨ Π↓] → [v ⊨ «AOT_enc κ1κn Π»]) ∧
730     (∀ φ . [v ⊨ [λz1...zn φ{z1...zn}]↓] → [v ⊨ «AOT_proj_enc κ1κn φ»])>
731
732 AOT_syntax_print_translations
733   "_AOT_enc (_AOT_individual_term κ) (_AOT_relation_term Π)" <= "CONST AOT_enc κ Π"
734
735 context AOT_meta_syntax
736 begin
737 notation AOT_enc ("_{_,_}")
738 end
739 context AOT_no_meta_syntax
740 begin
741 no_notation AOT_enc ("_{_,_}")
742 end
743
744 text<Unary encoding additionally has to satisfy the axioms of unary encoding and
745   the definition of property identity.>
746 class AOT_UnaryEnc = AOT_UnaryIndividualTerm +
747   assumes AOT_sem_enc_eq: <[v ⊨ Π↓ & Π'↓ & □∀ν (ν[Π] ≡ ν[Π']) → Π = Π']>
748   and AOT_sem_A_objects: <[v ⊨ ∃x (¬◇[E!]x & ∀F (x[F] ≡ φ{F}))]>
749   and AOT_sem_unary_proj_enc: <AOT_proj_enc x ψ = AOT_enc x «[λz ψ{z}]»>
750   and AOT_sem_nocoder: <[v ⊨ [E!]κ] ⇒ ¬[w ⊨ «AOT_enc κ Π»]>
751   and AOT_sem_ind_eq: <([v ⊨ κ↓] ∧ [v ⊨ κ'↓] ∧ κ = (κ')) =
752     ([v ⊨ [λx ◇[E!]x]κ] ∧
753     [v ⊨ [λx ◇[E!]x]κ'] ∧
754     (∀ v Π . [v ⊨ Π↓] → [v ⊨ [Π]κ] = [v ⊨ [Π]κ']))>

```

```

755     ∨ ([v ⊨ [λx ¬◇[E!]x]κ] ∧
756       [v ⊨ [λx ¬◇[E!]x]κ'] ∧
757       (∀ v Π . [v ⊨ Π↓] → [v ⊨ κ[Π]] = [v ⊨ κ'[Π]]))>
758
759     (* only extended models *)
760     and AOT_sem_enc_indistinguishable_all:
761       <AOT_ExtendedModel ==>
762         [v ⊨ [λx ¬◇[E!]x]κ] ==>
763         [v ⊨ [λx ¬◇[E!]x]κ'] ==>
764         (∧ Π' . [v ⊨ Π'↓] ==> (∧ w . [w ⊨ [Π']κ] = [w ⊨ [Π']κ'])) ==>
765         [v ⊨ Π↓] ==>
766         (∧ Π' . [v ⊨ Π'↓] ==> (∧ κ₀ . [v ⊨ [λx ◇[E!]x]κ₀] ==>
767           (∧ w . [w ⊨ [Π']κ₀] = [w ⊨ [Π]κ₀])) ==> [v ⊨ κ[Π']]) ==>
768         (∧ Π' . [v ⊨ Π'↓] ==> (∧ κ₀ . [v ⊨ [λx ◇[E!]x]κ₀] ==>
769           (∧ w . [w ⊨ [Π']κ₀] = [w ⊨ [Π]κ₀])) ==> [v ⊨ κ'[Π']])>
770     and AOT_sem_enc_indistinguishable_ex:
771       <AOT_ExtendedModel ==>
772         [v ⊨ [λx ¬◇[E!]x]κ] ==>
773         [v ⊨ [λx ¬◇[E!]x]κ'] ==>
774         (∧ Π' . [v ⊨ Π'↓] ==> (∧ w . [w ⊨ [Π']κ] = [w ⊨ [Π']κ'])) ==>
775         [v ⊨ Π↓] ==>
776         ∃ Π' . [v ⊨ Π'↓] ∧ [v ⊨ κ[Π']] ∧
777           (∀ κ₀ . [v ⊨ [λx ◇[E!]x]κ₀] →
778             (∀ w . [w ⊨ [Π']κ₀] = [w ⊨ [Π]κ₀])) ==>
779         ∃ Π' . [v ⊨ Π'↓] ∧ [v ⊨ κ'[Π']] ∧
780           (∀ κ₀ . [v ⊨ [λx ◇[E!]x]κ₀] →
781             (∀ w . [w ⊨ [Π']κ₀] = [w ⊨ [Π]κ₀]))>
782
783     text<We specify encoding to align with the model-construction of encoding.>
784     consts AOT_sem_enc_κ :: <κ ⇒ <κ> ⇒ o>
785     specification(AOT_sem_enc_κ)
786       AOT_sem_enc_κ:
787       <[v ⊨ «AOT_sem_enc_κ κ Π»] =
788         (AOT_model_denotes κ ∧ AOT_model_denotes Π ∧ AOT_model_enc κ Π)>
789       by (rule exI[where x=<λ κ Π . ε₀ w . AOT_model_denotes κ ∧ AOT_model_denotes Π ∧
790         AOT_model_enc κ Π>])
791       (simp add: AOT_model_proposition_choice_simp AOT_model_enc_κ_def κ.case_eq_if)
792
793     text<We show that @{typ κ} satisfies the generic properties of n-ary encoding.>
794     instantiation κ :: AOT_Enc
795     begin
796     definition AOT_enc_κ :: <κ ⇒ <κ> ⇒ o> where
797       <AOT_enc_κ ≡ AOT_sem_enc_κ>
798     definition AOT_proj_enc_κ :: <κ ⇒ (κ ⇒ o) ⇒ o> where
799       <AOT_proj_enc_κ ≡ λ κ φ . AOT_enc κ «[λz «φ z»»>
800     lemma AOT_enc_κ_meta:
801       <[v ⊨ κ[Π]] = (AOT_model_denotes κ ∧ AOT_model_denotes Π ∧ AOT_model_enc κ Π)>
802       for κ::κ
803       using AOT_sem_enc_κ unfolding AOT_enc_κ_def by auto
804     instance proof
805       fix v and κ :: κ and Π
806       show <[v ⊨ «AOT_enc κ Π»] ==> [v ⊨ κ↓] ∧ [v ⊨ Π↓]>
807         unfolding AOT_sem_denotes
808         using AOT_enc_κ_meta by blast
809     next
810       fix v and κ :: κ and Π
811       show <[v ⊨ κ[Π]] = [v ⊨ Π↓ & «AOT_proj_enc κ (λ κ' . «[Π]κ'»»)>
812       proof
813         assume enc: <[v ⊨ κ[Π]]>
814         hence Π_denotes: <AOT_model_denotes Π>
815           by (simp add: AOT_enc_κ_meta)
816         hence Π_eta_denotes: <AOT_model_denotes «[λz [Π]z»>
817           using AOT_sem_denotes AOT_sem_eq AOT_sem_lambda_eta by metis

```

```

818   show <[v ⊨ Π↓ & «AOT_proj_enc κ (λ κ. «[Π]κ»)»>
819     using AOT_sem_lambda_eta[simplified AOT_sem_denotes AOT_sem_eq, OF Π_denotes]
820     using Π_eta_denotes Π_denotes
821     by (simp add: AOT_sem_conj AOT_sem_denotes AOT_proj_enc_κ_def enc)
822   next
823     assume <[v ⊨ Π↓ & «AOT_proj_enc κ (λ κ. «[Π]κ»)»>
824     hence Π_denotes: "AOT_model_denotes Π" and eta_enc: "[v ⊨ κ[λz [Π]z]]"
825       by (auto simp: AOT_sem_conj AOT_sem_denotes AOT_proj_enc_κ_def)
826     thus <[v ⊨ κ[Π]]>
827       using AOT_sem_lambda_eta[simplified AOT_sem_denotes AOT_sem_eq, OF Π_denotes]
828       by auto
829   qed
830 next
831   show <[v ⊨ «AOT_proj_enc κ φ»] ⇒ [v ⊨ κ↓] > for v and κ :: κ and φ
832     by (simp add: AOT_enc_κ_meta AOT_sem_denotes AOT_proj_enc_κ_def)
833 next
834   fix v w and κ :: κ and Π
835   assume <[v ⊨ κ[Π]]>
836   thus <[w ⊨ κ[Π]]>
837     by (simp add: AOT_enc_κ_meta)
838 next
839   fix v w and κ :: κ and φ
840   assume <[v ⊨ «AOT_proj_enc κ φ»]>
841   thus <[w ⊨ «AOT_proj_enc κ φ»]>
842     by (simp add: AOT_enc_κ_meta AOT_proj_enc_κ_def)
843 next
844   show <∃κ::κ. [v ⊨ κ↓] ∧ (∀ Π . [v ⊨ Π↓] → [v ⊨ κ[Π]]) ∧
845     (∀ φ . [v ⊨ [λz φ{z}]↓] → [v ⊨ «AOT_proj_enc κ φ»])> for v
846     by (rule exI[where x=<ακ UNIV>])
847     (simp add: AOT_sem_denotes AOT_enc_κ_meta AOT_model_enc_κ_def
848       AOT_model_denotes_κ_def AOT_proj_enc_κ_def)
849   qed
850 end
851
852 text<We show that @{typ κ} satisfies the properties of unary encoding.>
853 instantiation κ :: AOT_UnaryEnc
854 begin
855 instance proof
856   fix v and Π Π' :: <<κ>>
857   show <[v ⊨ Π↓ & Π'↓ & □∀ν (ν[Π] ≡ ν[Π']) → Π = Π']>
858     apply (simp add: AOT_sem_forall AOT_sem_eq AOT_sem_imp AOT_sem_equiv
859       AOT_enc_κ_meta AOT_sem_conj AOT_sem_denotes AOT_sem_box)
860     using AOT_meta_A_objects_κ by fastforce
861 next
862   fix v and φ :: <<κ> ⇒ o>
863   show <[v ⊨ ∃x (¬◊[E!]x & ∀F (x[F] ≡ φ{F}))]>
864     using AOT_model_A_objects[of "λ Π . [v ⊨ φ{Π}]]"
865     by (auto simp: AOT_sem_denotes AOT_sem_exists AOT_sem_conj AOT_sem_not
866       AOT_sem_dia AOT_sem_concrete AOT_enc_κ_meta AOT_sem_equiv
867       AOT_sem_forall)
868 next
869   show <AOT_proj_enc x ψ = AOT_enc x (AOT_lambda ψ)> for x :: κ and ψ
870     by (simp add: AOT_proj_enc_κ_def)
871 next
872   show <[v ⊨ [E!]κ] ⇒ ¬ [w ⊨ κ[Π]]> for v w and κ :: κ and Π
873     by (simp add: AOT_enc_κ_meta AOT_sem_concrete AOT_model_nocoder)
874 next
875   fix v and κ κ' :: κ
876   show <([v ⊨ κ↓] ∧ [v ⊨ κ'↓] ∧ κ = κ') =
877     ([v ⊨ [λx ◊[E!]x]κ] ∧
878     [v ⊨ [λx ◊[E!]x]κ'] ∧
879     (∀ v Π . [v ⊨ Π↓] → [v ⊨ [Π]κ] = [v ⊨ [Π]κ']))
880     ∨ ([v ⊨ [λx ¬◊[E!]x]κ] ∧

```

```

881         [v ⊨ [λx ¬◇[E!]x]κ'] ∧
882         (∀ v II . [v ⊨ II] → [v ⊨ κ[II]] = [v ⊨ κ'[II]]))>
883     (is <?lhs = (?ordeq ∨ ?abseq)>)
884 proof -
885 {
886     assume 0: <[v ⊨ κ] ∧ [v ⊨ κ'] ∧ κ = κ'>
887     {
888         assume <is_ωκ κ'>
889         hence <[v ⊨ [λx ◇[E!]x]κ']>
890             apply (subst AOT_sem_lambda_beta[OF AOT_sem_ordinary_def_denotes, of v κ'])
891             apply (simp add: "0")
892             apply (simp add: AOT_sem_dia)
893             using AOT_sem_concrete AOT_model_ω_concrete_in_some_world is_ωκ_def by force
894         hence <?ordeq> unfolding 0[THEN conjunct2, THEN conjunct2] by auto
895     }
896     moreover {
897         assume <is_ακ κ'>
898         hence <[v ⊨ [λx ¬◇[E!]x]κ']>
899             apply (subst AOT_sem_lambda_beta[OF AOT_sem_abstract_def_denotes, of v κ'])
900             apply (simp add: "0")
901             apply (simp add: AOT_sem_not AOT_sem_dia)
902             using AOT_sem_concrete is_ακ_def by force
903         hence <?abseq> unfolding 0[THEN conjunct2, THEN conjunct2] by auto
904     }
905     ultimately have <?ordeq ∨ ?abseq>
906         by (meson "0" AOT_sem_denotes AOT_model_denotes_κ_def κ.exhaust_disc)
907     }
908     moreover {
909         assume ordeq: <?ordeq>
910         hence κ_denotes: <[v ⊨ κ] and κ'_denotes: <[v ⊨ κ']>
911             by (simp add: AOT_sem_denotes AOT_sem_exe)+
912         hence <is_ωκ κ> and <is_ωκ κ'>
913             by (metis AOT_model_concrete_κ.simps(2) AOT_model_denotes_κ_def
914                 AOT_sem_concrete AOT_sem_denotes AOT_sem_dia AOT_sem_lambda_beta
915                 AOT_sem_ordinary_def_denotes κ.collapse(2) κ.exhaust_disc ordeq)+
916         have denotes: <[v ⊨ [λz «εo w . κv z = κv κ»]↓]>
917             unfolding AOT_sem_denotes AOT_model_lambda_denotes
918             by (simp add: AOT_model_term_equiv_κ_def)
919         hence "[v ⊨ [λz «εo w . κv z = κv κ»]κ] = [v ⊨ [λz «εo w . κv z = κv κ»]κ']"
920             using ordeq by (simp add: AOT_sem_denotes)
921         hence <[v ⊨ «κ»] ∧ [v ⊨ «κ'»] ∧ κ = κ'>
922             unfolding AOT_sem_lambda_beta[OF denotes, OF κ_denotes]
923                 AOT_sem_lambda_beta[OF denotes, OF κ'_denotes]
924             using κ'_denotes <is_ωκ κ'> <is_ωκ κ> is_ωκ_def
925                 AOT_model_proposition_choice_simp by force
926     }
927     moreover {
928         assume 0: <?abseq>
929         hence κ_denotes: <[v ⊨ κ] and κ'_denotes: <[v ⊨ κ']>
930             by (simp add: AOT_sem_denotes AOT_sem_exe)+
931         hence <¬is_ωκ κ> and <¬is_ωκ κ'>
932             by (metis AOT_model_ω_concrete_in_some_world AOT_model_concrete_κ.simps(1)
933                 AOT_sem_concrete AOT_sem_dia AOT_sem_exe AOT_sem_lambda_beta
934                 AOT_sem_not κ.collapse(1) 0)+
935         hence <is_ακ κ> and <is_ακ κ'>
936             by (meson AOT_sem_denotes AOT_model_denotes_κ_def κ.exhaust_disc
937                 κ_denotes κ'_denotes)+
938         then obtain x y where κ_def: <κ = ακ x> and κ'_def: <κ' = ακ y>
939             using is_ακ_def by auto
940         {
941             fix r
942             assume <r ∈ x>
943             hence <[v ⊨ κ[«urrel_to_rel r»]]>

```

```

944   unfolding  $\kappa\_def$ 
945   unfolding AOT_enc_ $\kappa\_meta$ 
946   unfolding AOT_model_enc_ $\kappa\_def$ 
947   apply (simp add: AOT_model_denotes_ $\kappa\_def$ )
948   by (metis (mono_tags) AOT_rel_equiv_def Quotient_def urel_quotient)
949   hence  $\langle [v \models \kappa' [\llbracket \text{urrel\_to\_rel } r \rrbracket]] \rangle$ 
950   using AOT_enc_ $\kappa\_meta$  0 by (metis AOT_sem_enc_denotes)
951   hence  $\langle r \in y \rangle$ 
952   unfolding  $\kappa'\_def$ 
953   unfolding AOT_enc_ $\kappa\_meta$ 
954   unfolding AOT_model_enc_ $\kappa\_def$ 
955   apply (simp add: AOT_model_denotes_ $\kappa\_def$ )
956   using Quotient_abs_rep urel_quotient by fastforce
957 }
958 moreover {
959   fix r
960   assume  $\langle r \in y \rangle$ 
961   hence  $\langle [v \models \kappa' [\llbracket \text{urrel\_to\_rel } r \rrbracket]] \rangle$ 
962   unfolding  $\kappa'\_def$ 
963   unfolding AOT_enc_ $\kappa\_meta$ 
964   unfolding AOT_model_enc_ $\kappa\_def$ 
965   apply (simp add: AOT_model_denotes_ $\kappa\_def$ )
966   by (metis (mono_tags) AOT_rel_equiv_def Quotient_def urel_quotient)
967   hence  $\langle [v \models \kappa [\llbracket \text{urrel\_to\_rel } r \rrbracket]] \rangle$ 
968   using AOT_enc_ $\kappa\_meta$  0 by (metis AOT_sem_enc_denotes)
969   hence  $\langle r \in x \rangle$ 
970   unfolding  $\kappa\_def$ 
971   unfolding AOT_enc_ $\kappa\_meta$ 
972   unfolding AOT_model_enc_ $\kappa\_def$ 
973   apply (simp add: AOT_model_denotes_ $\kappa\_def$ )
974   using Quotient_abs_rep urel_quotient by fastforce
975 }
976 ultimately have "x = y" by blast
977 hence  $\langle [v \models \kappa \downarrow] \wedge [v \models \kappa' \downarrow] \wedge \kappa = \kappa' \rangle$ 
978   using  $\kappa'\_def$   $\kappa'\_denotes$   $\kappa\_def$  by blast
979 }
980 ultimately show ?thesis
981   unfolding AOT_sem_denotes
982   by auto
983 qed
984 (* Only extended model *)
985 next
986   fix v and  $\kappa \ \kappa' :: \kappa$  and  $\Pi \ \Pi' :: \langle \langle \kappa \rangle \rangle$ 
987   assume ext:  $\langle \text{AOT\_ExtendedModel} \rangle$ 
988   assume  $\langle [v \models [\lambda x \rightarrow \diamond [E!] x] \kappa] \rangle$ 
989   hence  $\langle \text{is\_}\alpha\kappa \ \kappa \rangle$ 
990     by (metis AOT_model_ $\omega$ _concrete_in_some_world AOT_model_concrete_ $\kappa$ .simps(1)
991           AOT_model_denotes_ $\kappa\_def$  AOT_sem_concrete AOT_sem_denotes AOT_sem_dia
992           AOT_sem_exe AOT_sem_lambda_beta AOT_sem_not  $\kappa$ .collapse(1)  $\kappa$ .exhaust_disc)
993   hence  $\kappa\_abs$ :  $\langle \neg (\exists w . \text{AOT\_model\_concrete } w \ \kappa) \rangle$ 
994     using is_ $\alpha\kappa\_def$  by fastforce
995   have  $\kappa\_den$ :  $\langle \text{AOT\_model\_denotes } \kappa \rangle$ 
996     by (simp add: AOT_model_denotes_ $\kappa\_def$   $\kappa$ .distinct_disc(5)  $\langle \text{is\_}\alpha\kappa \ \kappa \rangle$ )
997   assume  $\langle [v \models [\lambda x \rightarrow \diamond [E!] x] \kappa' \rangle$ 
998   hence  $\langle \text{is\_}\alpha\kappa \ \kappa' \rangle$ 
999     by (metis AOT_model_ $\omega$ _concrete_in_some_world AOT_model_concrete_ $\kappa$ .simps(1)
1000           AOT_model_denotes_ $\kappa\_def$  AOT_sem_concrete AOT_sem_denotes AOT_sem_dia
1001           AOT_sem_exe AOT_sem_lambda_beta AOT_sem_not  $\kappa$ .collapse(1)
1002            $\kappa$ .exhaust_disc)
1003   hence  $\kappa'\_abs$ :  $\langle \neg (\exists w . \text{AOT\_model\_concrete } w \ \kappa') \rangle$ 
1004     using is_ $\alpha\kappa\_def$  by fastforce
1005   have  $\kappa'\_den$ :  $\langle \text{AOT\_model\_denotes } \kappa' \rangle$ 
1006     by (meson AOT_model_denotes_ $\kappa\_def$   $\kappa$ .distinct_disc(6)  $\langle \text{is\_}\alpha\kappa \ \kappa' \rangle$ )

```

```

1007   assume <[v ⊨ Π'↓] ⇒ [w ⊨ [Π']κ] = [w ⊨ [Π']κ']> for Π' w
1008   hence indist: <[v ⊨ «Rep_rel Π' κ»] = [v ⊨ «Rep_rel Π' κ'»]>
1009     if <AOT_model_denotes Π'> for Π' v
1010     by (metis AOT_sem_denotes AOT_sem_exe κ'_den κ_den that)
1011   assume κ_enc_cond: <[v ⊨ Π'↓] ⇒
1012     (∧ κ₀ w. [v ⊨ [λx ◇ [E!]x]κ₀] ⇒
1013       [w ⊨ [Π']κ₀] = [w ⊨ [Π]κ₀]) ⇒
1014     [v ⊨ κ[Π']]> for Π'
1015   assume Π'_den': <[v ⊨ Π'↓]>
1016   hence Π'_den: <AOT_model_denotes Π'>
1017     using AOT_sem_denotes by blast
1018 {
1019   fix Π' :: <<κ>>
1020   assume Π'_den: <AOT_model_denotes Π'>
1021   hence Π'_den': <[v ⊨ Π'↓]>
1022     by (simp add: AOT_sem_denotes)
1023   assume 1: <∃w. AOT_model_concrete w x ⇒
1024     [v ⊨ «Rep_rel Π' x»] = [v ⊨ «Rep_rel Π x»]> for v x
1025   {
1026     fix κ₀ :: κ and w
1027     assume <[v ⊨ [λx ◇ [E!]x]κ₀]>
1028     hence <is_ωκ κ₀>
1029       by (smt (z3) AOT_model_concrete_κ.simps(2) AOT_model_denotes_κ_def
1030           AOT_sem_concrete AOT_sem_denotes AOT_sem_dia AOT_sem_exe
1031           AOT_sem_lambda_beta κ.exhaust_disc is_ακ_def)
1032     then obtain x where x_prop: <κ₀ = ωκ x>
1033       using is_ωκ_def by blast
1034     have <∃w . AOT_model_concrete w (ωκ x)>
1035       by (simp add: AOT_model_ω_concrete_in_some_world)
1036     hence <[v ⊨ «Rep_rel Π' (ωκ x)»] = [v ⊨ «Rep_rel Π (ωκ x)»]> for v
1037       using 1 by blast
1038     hence <[w ⊨ [Π']κ₀] = [w ⊨ [Π]κ₀]> unfolding x_prop
1039       by (simp add: AOT_sem_exe AOT_sem_denotes AOT_model_denotes_κ_def
1040           Π'_den Π'_den)
1041   } note 2 = this
1042   have <[v ⊨ κ[Π']]>
1043     using κ_enc_cond[OF Π'_den', OF 2]
1044     by metis
1045   hence <AOT_model_enc κ Π'>
1046     using AOT_enc_κ_meta by blast
1047 } note κ_enc_cond = this
1048   hence <AOT_model_denotes Π' ⇒
1049     (∧ v x. ∃w. AOT_model_concrete w x ⇒
1050       [v ⊨ «Rep_rel Π' x»] = [v ⊨ «Rep_rel Π x»]) ⇒
1051     AOT_model_enc κ Π'> for Π'
1052   by blast
1053   assume Π'_den': <[v ⊨ Π'↓]>
1054   hence Π'_den: <AOT_model_denotes Π'>
1055     using AOT_sem_denotes by blast
1056   assume ord_indist: <[v ⊨ [λx ◇ [E!]x]κ₀] ⇒
1057     [w ⊨ [Π']κ₀] = [w ⊨ [Π]κ₀]> for κ₀ w
1058 {
1059   fix w and κ₀ :: κ
1060   assume 0: <∃w. AOT_model_concrete w κ₀>
1061   hence <[v ⊨ [λx ◇ [E!]x]κ₀]>
1062     using AOT_model_concrete_denotes AOT_sem_concrete AOT_sem_denotes AOT_sem_dia
1063     AOT_sem_lambda_beta AOT_sem_ordinary_def_denotes by blast
1064   hence <[w ⊨ [Π']κ₀] = [w ⊨ [Π]κ₀]>
1065     using ord_indist by metis
1066   hence <[w ⊨ «Rep_rel Π' κ₀»] = [w ⊨ «Rep_rel Π κ₀»]>
1067     by (metis AOT_model_concrete_denotes AOT_sem_denotes AOT_sem_exe Π'_den Π'_den 0)
1068 } note ord_indist = this
1069   have <AOT_model_enc κ' Π'>

```

```

1070     using AOT_model_enc_indistinguishable_all
1071     [OF ext, OF  $\kappa$ _den, OF  $\kappa$ _abs, OF  $\kappa'$ _den, OF  $\kappa'$ _abs, OF  $\Pi$ _den]
1072     indist  $\kappa$ _enc_cond  $\Pi'$ _den ord_indist by blast
1073 thus <[v  $\models$   $\kappa'$ [ $\Pi'$ ]]>
1074     using AOT_enc_ $\kappa$ _meta  $\Pi'$ _den  $\kappa'$ _den by blast
1075 next
1076 fix v and  $\kappa$   $\kappa'$  ::  $\kappa$  and  $\Pi$   $\Pi'$  :: << $\kappa$ >>
1077 assume ext: <AOT_ExtendedModel>
1078 assume <[v  $\models$  [ $\lambda$ x  $\rightarrow$   $\diamond$ [E!]x] $\kappa$ ]>
1079 hence <is_ $\alpha$  $\kappa$   $\kappa$ >
1080     by (metis AOT_model_ $\omega$ _concrete_in_some_world AOT_model_concrete_ $\kappa$ .simps(1)
1081         AOT_model_denotes_ $\kappa$ _def AOT_sem_concrete AOT_sem_denotes AOT_sem_dia
1082         AOT_sem_exe AOT_sem_lambda_beta AOT_sem_not  $\kappa$ .collapse(1)
1083          $\kappa$ .exhaust_disc)
1084 hence  $\kappa$ _abs: < $\neg$ ( $\exists$  w . AOT_model_concrete w  $\kappa$ )>
1085     using is_ $\alpha$  $\kappa$ _def by fastforce
1086 have  $\kappa$ _den: <AOT_model_denotes  $\kappa$ >
1087     by (simp add: AOT_model_denotes_ $\kappa$ _def  $\kappa$ .distinct_disc(5) <is_ $\alpha$  $\kappa$   $\kappa$ >)
1088 assume <[v  $\models$  [ $\lambda$ x  $\rightarrow$   $\diamond$ [E!]x] $\kappa'$ ]>
1089 hence <is_ $\alpha$  $\kappa$   $\kappa'$ >
1090     by (metis AOT_model_ $\omega$ _concrete_in_some_world AOT_model_concrete_ $\kappa$ .simps(1)
1091         AOT_model_denotes_ $\kappa$ _def AOT_sem_concrete AOT_sem_denotes AOT_sem_dia
1092         AOT_sem_exe AOT_sem_lambda_beta AOT_sem_not  $\kappa$ .collapse(1)
1093          $\kappa$ .exhaust_disc)
1094 hence  $\kappa'$ _abs: < $\neg$ ( $\exists$  w . AOT_model_concrete w  $\kappa'$ )>
1095     using is_ $\alpha$  $\kappa$ _def by fastforce
1096 have  $\kappa'$ _den: <AOT_model_denotes  $\kappa'$ >
1097     by (meson AOT_model_denotes_ $\kappa$ _def  $\kappa$ .distinct_disc(6) <is_ $\alpha$  $\kappa$   $\kappa'$ >)
1098 assume <[v  $\models$   $\Pi'$ ] $\downarrow$   $\implies$  [w  $\models$  [ $\Pi'$ ] $\kappa$ ] = [w  $\models$  [ $\Pi'$ ] $\kappa'$ ]> for  $\Pi'$  w
1099 hence indist: <[v  $\models$  «Rep_rel  $\Pi'$   $\kappa$ »] = [v  $\models$  «Rep_rel  $\Pi'$   $\kappa'$ »]>
1100     if <AOT_model_denotes  $\Pi'$ > for  $\Pi'$  v
1101     by (metis AOT_sem_denotes AOT_sem_exe  $\kappa'$ _den  $\kappa$ _den that)
1102 assume  $\Pi$ _den': <[v  $\models$   $\Pi$ ] $\downarrow$ >
1103 hence  $\Pi$ _den: <AOT_model_denotes  $\Pi$ >
1104     using AOT_sem_denotes by blast
1105 assume < $\exists$  $\Pi'$ . [v  $\models$   $\Pi'$ ] $\downarrow$   $\wedge$  [v  $\models$   $\kappa$ [ $\Pi'$ ]]  $\wedge$ 
1106     ( $\forall$  $\kappa_0$ . [v  $\models$  [ $\lambda$ x  $\diamond$ [E!]x] $\kappa_0$ ]  $\rightarrow$ 
1107     ( $\forall$ w. [w  $\models$  [ $\Pi'$ ] $\kappa_0$ ] = [w  $\models$  [ $\Pi$ ] $\kappa_0$ ]))>
1108 then obtain  $\Pi'$  where
1109      $\Pi'$ _den: <[v  $\models$   $\Pi'$ ] $\downarrow$ > and
1110      $\Pi'$ _enc: <[v  $\models$   $\kappa$ [ $\Pi'$ ]]> and
1111      $\Pi'$ _prop: < $\forall$  $\kappa_0$ . [v  $\models$  [ $\lambda$ x  $\diamond$ [E!]x] $\kappa_0$ ]  $\rightarrow$ 
1112     ( $\forall$ w. [w  $\models$  [ $\Pi'$ ] $\kappa_0$ ] = [w  $\models$  [ $\Pi$ ] $\kappa_0$ ])>
1113     by blast
1114 have <AOT_model_denotes  $\Pi'$ >
1115     using AOT_enc_ $\kappa$ _meta  $\Pi'$ _enc by force
1116 moreover have <AOT_model_enc  $\kappa$   $\Pi'$ >
1117     using AOT_enc_ $\kappa$ _meta  $\Pi'$ _enc by blast
1118 moreover have <( $\exists$ w. AOT_model_concrete w  $\kappa_0$ )  $\implies$ 
1119     [v  $\models$  «Rep_rel  $\Pi'$   $\kappa_0$ »] = [v  $\models$  «Rep_rel  $\Pi$   $\kappa_0$ »]> for  $\kappa_0$  v
1120 proof -
1121     assume 0: < $\exists$ w. AOT_model_concrete w  $\kappa_0$ >
1122     hence <[v  $\models$  [ $\lambda$ x  $\diamond$ [E!]x] $\kappa_0$ ]> for v
1123         using AOT_model_concrete_denotes AOT_sem_concrete AOT_sem_denotes AOT_sem_dia
1124         AOT_sem_lambda_beta AOT_sem_ordinary_def_denotes by blast
1125     hence < $\forall$ w. [w  $\models$  [ $\Pi'$ ] $\kappa_0$ ] = [w  $\models$  [ $\Pi$ ] $\kappa_0$ ]> using  $\Pi'$ _prop by blast
1126     thus <[v  $\models$  «Rep_rel  $\Pi'$   $\kappa_0$ »] = [v  $\models$  «Rep_rel  $\Pi$   $\kappa_0$ »]>
1127         by (meson "0" AOT_model_concrete_denotes AOT_sem_denotes AOT_sem_exe  $\Pi$ _den
1128             calculation(1))
1129 qed
1130 ultimately have < $\exists$  $\Pi'$ . AOT_model_denotes  $\Pi'$   $\wedge$  AOT_model_enc  $\kappa$   $\Pi'$   $\wedge$ 
1131     ( $\forall$ v x. ( $\exists$ w. AOT_model_concrete w x)  $\rightarrow$ 
1132     [v  $\models$  «Rep_rel  $\Pi'$  x»] = [v  $\models$  «Rep_rel  $\Pi$  x»])>

```

```

1133   by blast
1134 hence <∃Π'. AOT_model_denotes Π' ∧ AOT_model_enc κ' Π' ∧
1135         (∀v x. (∃w. AOT_model_concrete w x) →
1136              [v ⊨ «Rep_rel Π' x»] = [v ⊨ «Rep_rel Π x»])>
1137   using AOT_model_enc_indistinguishable_ex
1138         [OF ext, OF κ_den, OF κ_abs, OF κ'_den, OF κ'_abs, OF Π_den]
1139   indist by blast
1140 then obtain Π" where
1141   Π"_den: <AOT_model_denotes Π">
1142   and Π"_enc: <AOT_model_enc κ' Π">
1143   and Π"_prop: <(∃w. AOT_model_concrete w x) ⇒
1144                [v ⊨ «Rep_rel Π" x»] = [v ⊨ «Rep_rel Π x»]> for v x
1145   by blast
1146 have <[v ⊨ Π"↓]>
1147   by (simp add: AOT_sem_denotes Π"_den)
1148 moreover have <[v ⊨ κ' [Π"]>
1149   by (simp add: AOT_enc_κ_meta Π"_den Π"_enc κ'_den)
1150 moreover have <[v ⊨ [λx ◇ [E!]x]κ₀] ⇒
1151                (∀w. [w ⊨ [Π"]κ₀] = [w ⊨ [Π]κ₀])> for κ₀
1152 proof -
1153   assume <[v ⊨ [λx ◇ [E!]x]κ₀]>
1154   hence <∃w. AOT_model_concrete w κ₀>
1155     by (metis AOT_sem_concrete AOT_sem_dia AOT_sem_exe AOT_sem_lambda_beta)
1156   thus <∀w. [w ⊨ [Π"]κ₀] = [w ⊨ [Π]κ₀]>
1157     using Π"_prop
1158     by (metis AOT_sem_denotes AOT_sem_exe Π"_den Π"_den)
1159 qed
1160 ultimately show <∃Π'. [v ⊨ Π'↓] ∧ [v ⊨ κ' [Π']] ∧
1161                   (∀κ₀. [v ⊨ [λx ◇ [E!]x]κ₀] →
1162                    (∀w. [w ⊨ [Π']κ₀] = [w ⊨ [Π]κ₀]))>
1163   by (safe intro!: exI[where x=Π"]) blast+
1164 qed
1165 end
1166
1167 text<Define encoding for products using projection-encoding.>
1168 instantiation prod :: (AOT_UnaryEnc, AOT_Enc) AOT_Enc
1169 begin
1170 definition AOT_proj_enc_prod :: <'a × 'b ⇒ ('a × 'b ⇒ o) ⇒ o> where
1171   <AOT_proj_enc_prod ≡ λ (κ, κ') φ . «κ [λν «φ (ν, κ')»] &
1172                                     «AOT_proj_enc κ' (λν. φ (κ, ν))»>>>
1173 definition AOT_enc_prod :: <'a × 'b ⇒ <'a × 'b> ⇒ o> where
1174   <AOT_enc_prod ≡ λ κ Π . «Π↓ & «AOT_proj_enc κ (λ κ₁ κ₂ . «[Π]κ₁ ... κ₂»»)>>>
1175 instance proof
1176   show <[v ⊨ κ₁ ... κₙ [Π]] ⇒ [v ⊨ κ₁ ... κₙ ↓] ∧ [v ⊨ Π↓]>
1177     for v and κ₁ κₙ :: <'a × 'b> and Π
1178     unfolding AOT_enc_prod_def
1179     apply (induct κ₁ κₙ; simp add: AOT_sem_conj AOT_sem_denotes AOT_proj_enc_prod_def)
1180     by (metis AOT_sem_denotes AOT_model_denotes_prod_def AOT_sem_enc_denotes
1181          AOT_sem_proj_enc_denotes case_prodI)
1182 next
1183   show <[v ⊨ κ₁ ... κₙ [Π]] =
1184         [v ⊨ «Π»↓ & «AOT_proj_enc κ₁ κₙ (λ κ₁ κ₂ . «[Π]κ₁ ... κ₂»»)>
1185     for v and κ₁ κₙ :: <'a × 'b> and Π
1186     unfolding AOT_enc_prod_def ..
1187 next
1188   show <[v ⊨ «AOT_proj_enc κs φ»] ⇒ [v ⊨ «κs»↓]>
1189     for v and κs :: <'a × 'b> and φ
1190     by (metis (mono_tags, lifting)
1191         AOT_sem_conj AOT_sem_denotes AOT_model_denotes_prod_def
1192         AOT_sem_enc_denotes AOT_sem_proj_enc_denotes
1193         AOT_proj_enc_prod_def case_prod_unfold)
1194 next
1195   fix v w Π and κ₁ κₙ :: <'a × 'b>

```



```

1196 show <[w ⊨ κ1...κn[Π]]> if <[v ⊨ κ1...κn[Π]]> for v w Π and κ1κn :: <'a×'b>
1197   by (metis (mono_tags, lifting)
1198       AOT_enc_prod_def AOT_sem_enc_proj_enc AOT_sem_conj AOT_sem_denotes
1199       AOT_sem_proj_enc_nec AOT_proj_enc_prod_def case_prod_unfold that)
1200 next
1201 show <[w ⊨ «AOT_proj_enc κ1κn φ»]> if <[v ⊨ «AOT_proj_enc κ1κn φ»]>
1202   for v w φ and κ1κn :: <'a×'b>
1203   by (metis (mono_tags, lifting)
1204       that AOT_sem_enc_proj_enc AOT_sem_conj AOT_sem_denotes
1205       AOT_sem_proj_enc_nec AOT_proj_enc_prod_def case_prod_unfold)
1206 next
1207 fix v
1208 obtain κ :: 'a where a_prop: <[v ⊨ κ↓] ∧ (∀ Π . [v ⊨ Π↓] → [v ⊨ κ[Π]])>
1209   using AOT_sem_universal_encoder by blast
1210 obtain κ1'κn' :: 'b where b_prop:
1211   <[v ⊨ κ1'...κn'↓] ∧ (∀ φ . [v ⊨ [λν1...νn «φ ν1νn»]↓] →
1212     [v ⊨ «AOT_proj_enc κ1'κn' φ»])>
1213   using AOT_sem_universal_encoder by blast
1214 have <AOT_model_denotes «[λν1...νn [«Π»]ν1...νn κ1'...κn' ]»>
1215   if <AOT_model_denotes Π> for Π :: <<'a×'b>>
1216   unfolding AOT_model_lambda_denotes
1217   by (metis AOT_meta_prod_equivI(2) AOT_sem_exe_equiv)
1218 moreover have <AOT_model_denotes «[λν1...νn [«Π»]κ ν1...νn]»>
1219   if <AOT_model_denotes Π> for Π :: <<'a×'b>>
1220   unfolding AOT_model_lambda_denotes
1221   by (metis AOT_meta_prod_equivI(1) AOT_sem_exe_equiv)
1222 ultimately have 1: <[v ⊨ «(κ, κ1'κn')»↓]>
1223   and 2: <(∀ Π . [v ⊨ Π↓] → [v ⊨ κ κ1'...κn'[Π]])>
1224   using a_prop b_prop
1225   by (auto simp: AOT_sem_denotes AOT_enc_κ_meta AOT_model_enc_κ_def
1226       AOT_model_denotes_κ_def AOT_model_denotes_prod_def
1227       AOT_enc_prod_def AOT_proj_enc_prod_def AOT_sem_conj)
1228 have <AOT_model_denotes «[λz1...zn «φ (z1zn, κ1'κn')»]»>
1229   if <AOT_model_denotes «[λz1...zn φ{z1...zn}]»> for φ :: <'a×'b ⇒ o>
1230   using that
1231   unfolding AOT_model_lambda_denotes
1232   by (metis (no_types, lifting) AOT_sem_denotes AOT_model_denotes_prod_def
1233       AOT_meta_prod_equivI(2) b_prop case_prodI)
1234 moreover have <AOT_model_denotes «[λz1...zn «φ (κ, z1zn)»]»>
1235   if <AOT_model_denotes «[λz1...zn φ{z1...zn}]»> for φ :: <'a×'b ⇒ o>
1236   using that
1237   unfolding AOT_model_lambda_denotes
1238   by (metis (no_types, lifting) AOT_sem_denotes AOT_model_denotes_prod_def
1239       AOT_meta_prod_equivI(1) a_prop case_prodI)
1240 ultimately have 3:
1241   <[v ⊨ «(κ, κ1'κn')»↓] ∧ (∀ φ . [v ⊨ [λz1...zn φ{z1...zn}]↓] →
1242     [v ⊨ «AOT_proj_enc (κ, κ1'κn') φ»])>
1243   using a_prop b_prop
1244   by (auto simp: AOT_sem_denotes AOT_enc_κ_meta AOT_model_enc_κ_def
1245       AOT_model_denotes_κ_def AOT_enc_prod_def AOT_proj_enc_prod_def
1246       AOT_sem_conj AOT_model_denotes_prod_def)
1247 show <∃κ1κn::'a×'b. [v ⊨ κ1...κn↓] ∧ (∀ Π . [v ⊨ Π↓] → [v ⊨ κ1...κn[Π]]) ∧
1248   (∀ φ . [v ⊨ [λz1...zn «φ z1zn»]↓] →
1249     [v ⊨ «AOT_proj_enc κ1κn φ»])>
1250   apply (rule exI[where x=<(κ, κ1'κn')>]) using 1 2 3 by blast
1251 qed
1252 end
1253
1254 text<Sanity-check to verify that n-ary encoding follows.>
1255 lemma <[v ⊨ κ1κ2[Π]] = [v ⊨ Π↓ & κ1[λν [Π]νκ2] & κ2[λν [Π]κ1ν]]>
1256   for κ1 :: "'a::AOT_UnaryEnc" and κ2 :: "'b::AOT_UnaryEnc"
1257   by (simp add: AOT_sem_conj AOT_enc_prod_def AOT_proj_enc_prod_def
1258       AOT_sem_unary_proj_enc)

```

```

1259 lemma <[v | = κ1κ2κ3[III]] =
1260     [v | = II↓ & κ1[λν [II]νκ2κ3] & κ2[λν [II]κ1νκ3] & κ3[λν [II]κ1κ2ν]]>
1261   for κ1 κ2 κ3 :: "'a::AOT_UnaryEnc"
1262   by (simp add: AOT_sem_conj AOT_enc_prod_def AOT_proj_enc_prod_def
1263         AOT_sem_unary_proj_enc)
1264
1265 lemma AOT_sem_vars_denote: <[v | = α1...αn↓]>
1266   by induct simp
1267
1268 text<Combine the introduced type classes and register them as
1269     type constraints for individual terms.>
1270 class AOT_κs = AOT_IndividualTerm + AOT_RelationProjection + AOT_Enc
1271 class AOT_κ = AOT_κs + AOT_UnaryIndividualTerm +
1272     AOT_UnaryRelationProjection + AOT_UnaryEnc
1273
1274 instance κ :: AOT_κ by standard
1275 instance prod :: (AOT_κ, AOT_κs) AOT_κs by standard
1276
1277 AOT_register_type_constraints
1278   Individual: <_::AOT_κ> <_::AOT_κs> and
1279   Relation: <<_::AOT_κs>>
1280
1281 text<We define semantic predicates to capture the conditions of cqt.2 (i.e.
1282     the base cases of denoting terms) on matrices of @{text λ}-expressions.>
1283 definition AOT_instance_of_cqt_2 :: <('a::AOT_κs ⇒ o) ⇒ bool> where
1284   <AOT_instance_of_cqt_2 ≡ λ φ . ∀ x y . AOT_model_denotes x ∧ AOT_model_denotes y ∧
1285     AOT_model_term_equiv x y ⟶ φ x = φ y>
1286 definition AOT_instance_of_cqt_2_exe_arg :: <('a::AOT_κs ⇒ 'b::AOT_κs) ⇒ bool> where
1287   <AOT_instance_of_cqt_2_exe_arg ≡ λ φ . ∀ x y .
1288     AOT_model_denotes x ∧ AOT_model_denotes y ∧ AOT_model_term_equiv x y ⟶
1289     AOT_model_term_equiv (φ x) (φ y)>
1290
1291 text<@{text λ}-expressions with a matrix that satisfies our predicate denote.>
1292 lemma AOT_sem_cqt_2:
1293   assumes <AOT_instance_of_cqt_2 φ>
1294   shows <[v | = [λν1...νn φ{ν1...νn}]↓]>
1295   using assms
1296   by (metis AOT_instance_of_cqt_2_def AOT_model_lambda_denotes AOT_sem_denotes)
1297
1298 syntax AOT_instance_of_cqt_2 :: <id_position ⇒ AOT_prop>
1299   ("INSTANCE'_OF'_CQT'_2'(_)'")
1300
1301 text<Prove introduction rules for the predicates that match the natural language
1302     restrictions of the axiom.>
1303 named_theorems AOT_instance_of_cqt_2_intro
1304 lemma AOT_instance_of_cqt_2_intros_const[AOT_instance_of_cqt_2_intro]:
1305   <AOT_instance_of_cqt_2 (λα. φ)>
1306   by (simp add: AOT_instance_of_cqt_2_def AOT_sem_denotes AOT_model_lambda_denotes)
1307 lemma AOT_instance_of_cqt_2_intros_not[AOT_instance_of_cqt_2_intro]:
1308   assumes <AOT_instance_of_cqt_2 φ>
1309   shows <AOT_instance_of_cqt_2 (λτ. «¬φ{τ}»)>
1310   using assms
1311   by (metis (no_types, lifting) AOT_instance_of_cqt_2_def)
1312 lemma AOT_instance_of_cqt_2_intros_imp[AOT_instance_of_cqt_2_intro]:
1313   assumes <AOT_instance_of_cqt_2 φ> and <AOT_instance_of_cqt_2 ψ>
1314   shows <AOT_instance_of_cqt_2 (λτ. «φ{τ} → ψ{τ}»)>
1315   using assms
1316   by (auto simp: AOT_instance_of_cqt_2_def AOT_sem_denotes
1317         AOT_model_lambda_denotes AOT_sem_imp)
1318 lemma AOT_instance_of_cqt_2_intros_box[AOT_instance_of_cqt_2_intro]:
1319   assumes <AOT_instance_of_cqt_2 φ>
1320   shows <AOT_instance_of_cqt_2 (λτ. «□φ{τ}»)>
1321   using assms

```

```

1322   by (auto simp: AOT_instance_of_cqt_2_def AOT_sem_denotes
1323           AOT_model_lambda_denotes AOT_sem_box)
1324 lemma AOT_instance_of_cqt_2_intros_act[AOT_instance_of_cqt_2_intro]:
1325   assumes <AOT_instance_of_cqt_2  $\varphi$ >
1326   shows <AOT_instance_of_cqt_2 ( $\lambda\tau. \ll \mathcal{A}\varphi\{\tau\} \gg$ )>
1327   using assms
1328   by (auto simp: AOT_instance_of_cqt_2_def AOT_sem_denotes
1329           AOT_model_lambda_denotes AOT_sem_act)
1330 lemma AOT_instance_of_cqt_2_intros_diamond[AOT_instance_of_cqt_2_intro]:
1331   assumes <AOT_instance_of_cqt_2  $\varphi$ >
1332   shows <AOT_instance_of_cqt_2 ( $\lambda\tau. \ll \diamond\varphi\{\tau\} \gg$ )>
1333   using assms
1334   by (auto simp: AOT_instance_of_cqt_2_def AOT_sem_denotes
1335           AOT_model_lambda_denotes AOT_sem_dia)
1336 lemma AOT_instance_of_cqt_2_intros_conj[AOT_instance_of_cqt_2_intro]:
1337   assumes <AOT_instance_of_cqt_2  $\varphi$ > and <AOT_instance_of_cqt_2  $\psi$ >
1338   shows <AOT_instance_of_cqt_2 ( $\lambda\tau. \ll \varphi\{\tau\} \ \& \ \psi\{\tau\} \gg$ )>
1339   using assms
1340   by (auto simp: AOT_instance_of_cqt_2_def AOT_sem_denotes
1341           AOT_model_lambda_denotes AOT_sem_conj)
1342 lemma AOT_instance_of_cqt_2_intros_disj[AOT_instance_of_cqt_2_intro]:
1343   assumes <AOT_instance_of_cqt_2  $\varphi$ > and <AOT_instance_of_cqt_2  $\psi$ >
1344   shows <AOT_instance_of_cqt_2 ( $\lambda\tau. \ll \varphi\{\tau\} \ \vee \ \psi\{\tau\} \gg$ )>
1345   using assms
1346   by (auto simp: AOT_instance_of_cqt_2_def AOT_sem_denotes
1347           AOT_model_lambda_denotes AOT_sem_disj)
1348 lemma AOT_instance_of_cqt_2_intros_equip[AOT_instance_of_cqt_2_intro]:
1349   assumes <AOT_instance_of_cqt_2  $\varphi$ > and <AOT_instance_of_cqt_2  $\psi$ >
1350   shows <AOT_instance_of_cqt_2 ( $\lambda\tau. \ll \varphi\{\tau\} \ \equiv \ \psi\{\tau\} \gg$ )>
1351   using assms
1352   by (auto simp: AOT_instance_of_cqt_2_def AOT_sem_denotes
1353           AOT_model_lambda_denotes AOT_sem_equiv)
1354 lemma AOT_instance_of_cqt_2_intros_forall[AOT_instance_of_cqt_2_intro]:
1355   assumes < $\bigwedge \alpha. \text{AOT\_instance\_of\_cqt\_2 } (\Phi \ \alpha)$ >
1356   shows <AOT_instance_of_cqt_2 ( $\lambda\tau. \ll \forall \alpha \ \Phi\{\alpha, \tau\} \gg$ )>
1357   using assms
1358   by (auto simp: AOT_instance_of_cqt_2_def AOT_sem_denotes
1359           AOT_model_lambda_denotes AOT_sem_forall)
1360 lemma AOT_instance_of_cqt_2_intros_exists[AOT_instance_of_cqt_2_intro]:
1361   assumes < $\bigwedge \alpha. \text{AOT\_instance\_of\_cqt\_2 } (\Phi \ \alpha)$ >
1362   shows <AOT_instance_of_cqt_2 ( $\lambda\tau. \ll \exists \alpha \ \Phi\{\alpha, \tau\} \gg$ )>
1363   using assms
1364   by (auto simp: AOT_instance_of_cqt_2_def AOT_sem_denotes
1365           AOT_model_lambda_denotes AOT_sem_exists)
1366 lemma AOT_instance_of_cqt_2_intros_exe_arg_self[AOT_instance_of_cqt_2_intro]:
1367   <AOT_instance_of_cqt_2_exe_arg ( $\lambda x. \ x$ )>
1368   unfolding AOT_instance_of_cqt_2_exe_arg_def AOT_instance_of_cqt_2_def
1369   AOT_sem_lambda_denotes
1370   by (auto simp: AOT_model_term_equiv_part_equivp equivp_refl AOT_sem_denotes)
1371 lemma AOT_instance_of_cqt_2_intros_exe_arg_const[AOT_instance_of_cqt_2_intro]:
1372   <AOT_instance_of_cqt_2_exe_arg ( $\lambda x. \ \kappa$ )>
1373   unfolding AOT_instance_of_cqt_2_exe_arg_def AOT_instance_of_cqt_2_def
1374   by (auto simp: AOT_model_term_equiv_part_equivp equivp_refl
1375           AOT_sem_denotes AOT_sem_lambda_denotes)
1376 lemma AOT_instance_of_cqt_2_intros_exe_arg_fst[AOT_instance_of_cqt_2_intro]:
1377   <AOT_instance_of_cqt_2_exe_arg fst>
1378   unfolding AOT_instance_of_cqt_2_exe_arg_def AOT_instance_of_cqt_2_def
1379   by (simp add: AOT_model_term_equiv_prod_def case_prod_beta)
1380 lemma AOT_instance_of_cqt_2_intros_exe_arg_snd[AOT_instance_of_cqt_2_intro]:
1381   <AOT_instance_of_cqt_2_exe_arg snd>
1382   unfolding AOT_instance_of_cqt_2_exe_arg_def AOT_instance_of_cqt_2_def
1383   by (simp add: AOT_model_term_equiv_prod_def AOT_sem_denotes AOT_sem_lambda_denotes)
1384 lemma AOT_instance_of_cqt_2_intros_exe_arg_Pair[AOT_instance_of_cqt_2_intro]:

```

```

1385   assumes <AOT_instance_of_cqt_2_exe_arg  $\varphi$ > and <AOT_instance_of_cqt_2_exe_arg  $\psi$ >
1386   shows <AOT_instance_of_cqt_2_exe_arg ( $\lambda\tau. \text{Pair } (\varphi \ \tau) \ (\psi \ \tau)$ )>
1387   using assms
1388   unfolding AOT_instance_of_cqt_2_exe_arg_def AOT_instance_of_cqt_2_def
1389           AOT_sem_denotes AOT_sem_lambda_denotes AOT_model_term_equiv_prod_def
1390           AOT_model_denotes_prod_def
1391   by auto
1392 lemma AOT_instance_of_cqt_2_intros_desc[AOT_instance_of_cqt_2_intro]:
1393   assumes < $\bigwedge z :: 'a :: \text{AOT\_}\kappa. \text{AOT\_instance\_of\_cqt\_2 } (\Phi \ z)$ >
1394   shows <AOT_instance_of_cqt_2_exe_arg ( $\lambda \kappa :: 'b :: \text{AOT\_}\kappa. \ll \text{LZ}(\Phi\{z, \kappa\}) \gg$ )>
1395 proof -
1396   have 0: < $\bigwedge \kappa \ \kappa'. \text{AOT\_model\_denotes } \kappa \wedge \text{AOT\_model\_denotes } \kappa' \wedge$   

1397           AOT_model_term_equiv  $\kappa \ \kappa' \implies$   

1398            $\Phi \ z \ \kappa = \Phi \ z \ \kappa'$ > for z
1399   using assms
1400   unfolding AOT_instance_of_cqt_2_def
1401           AOT_sem_denotes AOT_model_lambda_denotes by force
1402 {
1403   fix  $\kappa \ \kappa'$ 
1404   have < $\ll \text{LZ}(\Phi\{z, \kappa\}) \gg = \ll \text{LZ}(\Phi\{z, \kappa'\}) \gg$ >
1405     if <AOT_model_denotes  $\kappa \wedge \text{AOT\_model\_denotes } \kappa' \wedge \text{AOT\_model\_term\_equiv } \kappa \ \kappa'$ >
1406     using 0[OF that]
1407     by auto
1408   moreover have <AOT_model_term_equiv x x> for x :: <'a::AOT_κ>
1409     by (metis AOT_instance_of_cqt_2_exe_arg_def
1410             AOT_instance_of_cqt_2_intros_exe_arg_const
1411             AOT_model_A_objects AOT_model_term_equiv_denotes
1412             AOT_model_term_equiv_eps(1))
1413   ultimately have <AOT_model_term_equiv  $\ll \text{LZ}(\Phi\{z, \kappa\}) \gg \ll \text{LZ}(\Phi\{z, \kappa'\}) \gg$ >
1414     if <AOT_model_denotes  $\kappa \wedge \text{AOT\_model\_denotes } \kappa' \wedge \text{AOT\_model\_term\_equiv } \kappa \ \kappa'$ >
1415     using that by simp
1416 }
1417 thus ?thesis using 0
1418   unfolding AOT_instance_of_cqt_2_exe_arg_def
1419   by simp
1420 qed
1421
1422 lemma AOT_instance_of_cqt_2_intros_exe_const[AOT_instance_of_cqt_2_intro]:
1423   assumes <AOT_instance_of_cqt_2_exe_arg  $\kappa s$ >
1424   shows <AOT_instance_of_cqt_2 ( $\lambda x :: 'b :: \text{AOT\_}\kappa s. \text{AOT\_exe } \Pi \ (\kappa s \ x)$ )>
1425   using assms
1426   unfolding AOT_instance_of_cqt_2_def AOT_sem_denotes AOT_model_lambda_denotes
1427           AOT_sem_disj AOT_sem_conj
1428           AOT_sem_not AOT_sem_box AOT_sem_act AOT_instance_of_cqt_2_exe_arg_def
1429           AOT_sem_equiv AOT_sem_imp AOT_sem_forall AOT_sem_exists AOT_sem_dia
1430   by (auto intro!: AOT_sem_exe_equiv)
1431 lemma AOT_instance_of_cqt_2_intros_exe_lam[AOT_instance_of_cqt_2_intro]:
1432   assumes < $\bigwedge y. \text{AOT\_instance\_of\_cqt\_2 } (\lambda x. \varphi \ x \ y)$ >
1433   and <AOT_instance_of_cqt_2_exe_arg  $\kappa s$ >
1434   shows <AOT_instance_of_cqt_2 ( $\lambda \kappa_1 \kappa_n :: 'b :: \text{AOT\_}\kappa s. \ll [\lambda \nu_1 \dots \nu_n. \varphi\{\kappa_1 \dots \kappa_n, \nu_1 \dots \nu_n\}] \ll \kappa s \ \kappa_1 \kappa_n \gg \gg$ )>
1435 proof -
1436 {
1437   fix x y :: 'b
1438   assume <AOT_model_denotes x>
1439   moreover assume <AOT_model_denotes y>
1440   moreover assume <AOT_model_term_equiv x y>
1441   moreover have 1: < $\varphi \ x = \varphi \ y$ >
1442     using assms calculation unfolding AOT_instance_of_cqt_2_def by blast
1443   ultimately have <AOT_exe (AOT_lambda ( $\varphi \ x$ )) ( $\kappa s \ x$ ) =  

1444           AOT_exe (AOT_lambda ( $\varphi \ y$ )) ( $\kappa s \ y$ )>
1445   unfolding 1
1446   apply (safe intro!: AOT_sem_exe_equiv)

```

```

1448     by (metis AOT_instance_of_cqt_2_exe_arg_def assms(2))
1449 }
1450 thus ?thesis
1451 unfolding AOT_instance_of_cqt_2_def
1452         AOT_instance_of_cqt_2_exe_arg_def
1453 by blast
1454 qed
1455 lemma AOT_instance_of_cqt_2_intro_prod[AOT_instance_of_cqt_2_intro]:
1456   assumes <math>\langle \bigwedge x . \text{AOT\_instance\_of\_cqt\_2 } (\varphi x) \rangle</math>
1457     and <math>\langle \bigwedge x . \text{AOT\_instance\_of\_cqt\_2 } (\lambda z . \varphi z x) \rangle</math>
1458   shows <math>\langle \text{AOT\_instance\_of\_cqt\_2 } (\lambda(x,y) . \varphi x y) \rangle</math>
1459   using assms unfolding AOT_instance_of_cqt_2_def
1460   by (auto simp add: AOT_model_lambda_denotes AOT_sem_denotes
1461       AOT_model_denotes_prod_def
1462       AOT_model_term_equiv_prod_def)
1463
1464 text<The following are already derivable semantically, but not yet added
1465 to @{attribute AOT_instance_of_cqt_2_intro}. They will be added with the
1466 next planned extension of axiom cqt:2.>
1467 named_theorems AOT_instance_of_cqt_2_intro_next
1468 definition AOT_instance_of_cqt_2_enc_arg :: <math>\langle 'a::\text{AOT\_}\kappa\text{s} \Rightarrow 'b::\text{AOT\_}\kappa\text{s} \rangle \Rightarrow \text{bool}></math> where
1469   <math>\langle \text{AOT\_instance\_of\_cqt\_2\_enc\_arg} \equiv \lambda \varphi . \forall x y z .</math>
1470     AOT_model_denotes x  $\wedge$  AOT_model_denotes y  $\wedge$  AOT_model_term_equiv x y  $\longrightarrow$ 
1471     AOT_enc ( $\varphi$  x) z = AOT_enc ( $\varphi$  y) z >
1472 definition AOT_instance_of_cqt_2_enc_rel :: <math>\langle 'a::\text{AOT\_}\kappa\text{s} \Rightarrow \langle 'b::\text{AOT\_}\kappa\text{s} \rangle \rangle \Rightarrow \text{bool}></math> where
1473   <math>\langle \text{AOT\_instance\_of\_cqt\_2\_enc\_rel} \equiv \lambda \varphi . \forall x y z .</math>
1474     AOT_model_denotes x  $\wedge$  AOT_model_denotes y  $\wedge$  AOT_model_term_equiv x y  $\longrightarrow$ 
1475     AOT_enc z ( $\varphi$  x) = AOT_enc z ( $\varphi$  y) >
1476 lemma AOT_instance_of_cqt_2_intros_enc[AOT_instance_of_cqt_2_intro_next]:
1477   assumes <math>\langle \text{AOT\_instance\_of\_cqt\_2\_enc\_rel } \Pi \rangle</math> and <math>\langle \text{AOT\_instance\_of\_cqt\_2\_enc\_arg } \kappa\text{s} \rangle</math>
1478   shows <math>\langle \text{AOT\_instance\_of\_cqt\_2 } (\lambda x . \text{AOT\_enc } (\kappa\text{s } x) \ll \ll \Pi x \gg \gg) \rangle</math>
1479   using assms
1480   unfolding AOT_instance_of_cqt_2_def AOT_sem_denotes AOT_model_lambda_denotes
1481     AOT_instance_of_cqt_2_enc_rel_def AOT_sem_not AOT_sem_box AOT_sem_act
1482     AOT_sem_dia AOT_sem_conj AOT_sem_disj AOT_sem_equiv AOT_sem_imp
1483     AOT_sem_forall AOT_sem_exists AOT_instance_of_cqt_2_enc_arg_def
1484   by fastforce+
1485 lemma AOT_instance_of_cqt_2_enc_arg_intro_const[AOT_instance_of_cqt_2_intro_next]:
1486   <math>\langle \text{AOT\_instance\_of\_cqt\_2\_enc\_arg } (\lambda x . c) \rangle</math>
1487   unfolding AOT_instance_of_cqt_2_enc_arg_def by simp
1488 lemma AOT_instance_of_cqt_2_enc_arg_intro_desc[AOT_instance_of_cqt_2_intro_next]:
1489   assumes <math>\langle \bigwedge z :: 'a::\text{AOT\_}\kappa . \text{AOT\_instance\_of\_cqt\_2 } (\Phi z) \rangle</math>
1490   shows <math>\langle \text{AOT\_instance\_of\_cqt\_2\_enc\_arg } (\lambda \kappa :: 'b::\text{AOT\_}\kappa . \ll \ll z(\Phi\{z,\kappa\}) \gg \gg) \rangle</math>
1491 proof -
1492   have 0: <math>\langle \bigwedge \kappa \kappa' . \text{AOT\_model\_denotes } \kappa \wedge \text{AOT\_model\_denotes } \kappa' \wedge</math>
1493     AOT_model_term_equiv  $\kappa \kappa' \implies$ 
1494      $\Phi z \kappa = \Phi z \kappa' \rangle$  for z
1495   using assms
1496   unfolding AOT_instance_of_cqt_2_def
1497     AOT_sem_denotes AOT_model_lambda_denotes by force
1498   {
1499     fix  $\kappa \kappa'$ 
1500     have <math>\ll \ll z(\Phi\{z,\kappa\}) \gg = \ll \ll z(\Phi\{z,\kappa'\}) \gg \gg</math>
1501       if <math>\langle \text{AOT\_model\_denotes } \kappa \wedge \text{AOT\_model\_denotes } \kappa' \wedge \text{AOT\_model\_term\_equiv } \kappa \kappa' \rangle</math>
1502       using 0[OF that]
1503       by auto
1504   }
1505   thus ?thesis using 0
1506   unfolding AOT_instance_of_cqt_2_enc_arg_def by meson
1507 qed
1508 lemma AOT_instance_of_cqt_2_enc_rel_intro[AOT_instance_of_cqt_2_intro_next]:
1509   assumes <math>\langle \bigwedge \kappa' . \text{AOT\_instance\_of\_cqt\_2 } (\lambda \kappa :: 'a::\text{AOT\_}\kappa\text{s} . \varphi \kappa \kappa') \rangle</math>
1510   shows <math>\langle \text{AOT\_instance\_of\_cqt\_2\_enc\_rel } (\lambda \kappa :: 'a::\text{AOT\_}\kappa\text{s} . \text{AOT\_lambda } (\lambda \kappa' . \varphi \kappa \kappa')) \rangle</math>

```

```

1511 proof -
1512   {
1513     fix x y :: 'a and z :: 'b
1514     assume <AOT_model_term_equiv x y>
1515     moreover assume <AOT_model_denotes x>
1516     moreover assume <AOT_model_denotes y>
1517     ultimately have < $\varphi x = \varphi y$ >
1518       using assms unfolding AOT_instance_of_cqt_2_def by blast
1519     hence <AOT_enc z (AOT_lambda ( $\varphi x$ )) = AOT_enc z (AOT_lambda ( $\varphi y$ ))>
1520       by simp
1521   }
1522   thus ?thesis
1523     unfolding AOT_instance_of_cqt_2_enc_rel_def by auto
1524 qed
1525
1526 text<Further restrict unary individual variables to type @{typ  $\kappa$ } (rather
1527   than class @{class AOT_ $\kappa$ } only) and define being ordinary and being abstract.>
1528 AOT_register_type_constraints
1529   Individual: < $\kappa$ > <_::AOT_ $\kappa$ S>
1530
1531 AOT_define AOT_ordinary :: <II> (<O!>) <O! =df [ $\lambda x \langle E!x \rangle$ >
1532 declare AOT_ordinary[AOT del, AOT_defs del]
1533 AOT_define AOT_abstract :: <II> (<A!>) <A! =df [ $\lambda x \neg \langle E!x \rangle$ >
1534 declare AOT_abstract[AOT del, AOT_defs del]
1535
1536 context AOT_meta_syntax
1537 begin
1538 notation AOT_ordinary ("O!")
1539 notation AOT_abstract ("A!")
1540 end
1541 context AOT_no_meta_syntax
1542 begin
1543 no_notation AOT_ordinary ("O!")
1544 no_notation AOT_abstract ("A!")
1545 end
1546
1547 no_translations
1548 "_AOT_concrete" => "CONST AOT_term_of_var (CONST AOT_concrete)"
1549 parse_translation<
1550 [(syntax_const<_AOT_concrete>, fn _ => fn [] =>
1551   Const (const_name<AOT_term_of_var>, dummyT)
1552   $ Const (const_name<AOT_concrete>, typ<< $\kappa$ > AOT_var)))]
1553 >
1554
1555 text<Auxiliary lemmata.>
1556 lemma AOT_sem_ordinary: "<O!> = << $\lambda x \langle E!x \rangle$ >>"
1557   using AOT_ordinary[THEN AOT_sem_id_defOE1] AOT_sem_ordinary_def_denotes
1558   by (auto simp: AOT_sem_eq)
1559 lemma AOT_sem_abstract: "<A!> = << $\lambda x \neg \langle E!x \rangle$ >>"
1560   using AOT_abstract[THEN AOT_sem_id_defOE1] AOT_sem_abstract_def_denotes
1561   by (auto simp: AOT_sem_eq)
1562 lemma AOT_sem_ordinary_denotes: <[w  $\models$  O! $\downarrow$ ]>
1563   by (simp add: AOT_sem_ordinary AOT_sem_ordinary_def_denotes)
1564 lemma AOT_meta_abstract_denotes: <[w  $\models$  A! $\downarrow$ ]>
1565   by (simp add: AOT_sem_abstract AOT_sem_abstract_def_denotes)
1566 lemma AOT_model_abstract_ $\alpha\kappa$ : < $\exists a . \kappa = \alpha\kappa a$ > if <[v  $\models$  A! $\kappa$ ]>
1567   using that[unfolded AOT_sem_abstract, simplified
1568     AOT_meta_abstract_denotes[unfolded AOT_sem_abstract, THEN AOT_sem_lambda_beta,
1569     OF that[simplified AOT_sem_exe, THEN conjunct2, THEN conjunct1]]]
1570   apply (simp add: AOT_sem_not AOT_sem_dia AOT_sem_concrete)
1571   by (metis AOT_model_ $\omega$ _concrete_in_some_world AOT_model_concrete_ $\kappa$ .simps(1)
1572     AOT_model_denotes_ $\kappa$ _def AOT_sem_denotes AOT_sem_exe  $\kappa$ .exhaust_disc
1573     is_ $\alpha\kappa$ _def is_ $\omega\kappa$ _def that)

```



```

1574 lemma AOT_model_ordinary_ωκ: <∃ a . κ = ωκ a > if <[v ⊨ 0!κ]>
1575   using that[unfolded AOT_sem_ordinary, simplified
1576     AOT_sem_ordinary_denotes[unfolded AOT_sem_ordinary, THEN AOT_sem_lambda_beta,
1577       OF that[simplified AOT_sem_exe, THEN conjunct2, THEN conjunct1]]]
1578   apply (simp add: AOT_sem_dia AOT_sem_concrete)
1579   by (metis AOT_model_concrete_κ.simps(2) AOT_model_concrete_κ.simps(3)
1580     κ.exhaust_disc is_ακ_def is_ωκ_def is_nullκ_def)
1581 lemma AOT_model_ωκ_ordinary: <[v ⊨ 0!«ωκ x»]>
1582   by (metis AOT_model_abstract_ακ AOT_model_denotes_κ_def AOT_sem_abstract
1583     AOT_sem_denotes AOT_sem_ind_eq AOT_sem_ordinary κ.disc(7) κ.distinct(1))
1584 lemma AOT_model_ακ_ordinary: <[v ⊨ A!«ακ x»]>
1585   by (metis AOT_model_denotes_κ_def AOT_model_ordinary_ωκ AOT_sem_abstract
1586     AOT_sem_denotes AOT_sem_ind_eq AOT_sem_ordinary κ.disc(8) κ.distinct(1))
1587 AOT_theorem prod_denotesE: assumes <<(κ1, κ2)»↓> shows <κ1↓ & κ2↓>
1588   using assms by (simp add: AOT_sem_denotes AOT_sem_conj AOT_model_denotes_prod_def)
1589 declare prod_denotesE[AOT del]
1590 AOT_theorem prod_denotesI: assumes <κ1↓ & κ2↓> shows <<(κ1, κ2)»↓>
1591   using assms by (simp add: AOT_sem_denotes AOT_sem_conj AOT_model_denotes_prod_def)
1592 declare prod_denotesI[AOT del]
1593
1594
1595 text<Prepare the derivation of the additional axioms that are validated by
1596   our extended models.>
1597 locale AOT_ExtendedModel =
1598   assumes AOT_ExtendedModel: <AOT_ExtendedModel>
1599 begin
1600 lemma AOT_sem_indistinguishable_ord_enc_all:
1601   assumes Πden: <[v ⊨ Π↓]>
1602   assumes Ax: <[v ⊨ A!x]>
1603   assumes Ay: <[v ⊨ A!y]>
1604   assumes indist: <[v ⊨ ∀F □([F]x ≡ [F]y)]>
1605   shows
1606     <[v ⊨ ∀G(∀z(0!z → □([G]z ≡ [Π]z)) → x[G])] =
1607     [v ⊨ ∀G(∀z(0!z → □([G]z ≡ [Π]z)) → y[G])>
1608 proof -
1609   have 0: <[v ⊨ [λx →◇[E!]x]x]>
1610     using Ax by (simp add: AOT_sem_abstract)
1611   have 1: <[v ⊨ [λx →◇[E!]x]y]>
1612     using Ay by (simp add: AOT_sem_abstract)
1613   {
1614     assume <[v ⊨ ∀G(∀z (0!z → □([G]z ≡ [Π]z)) → x[G])>
1615     hence 3: <[v ⊨ ∀G(∀z([λx ◇[E!]x]z → □([G]z ≡ [Π]z)) → x[G])>
1616       by (simp add: AOT_sem_ordinary)
1617     {
1618       fix Π' :: <<κ>>
1619       assume 1: <[v ⊨ Π'↓]>
1620       assume 2: <[v ⊨ [λx ◇[E!]x]z → □([Π']z ≡ [Π]z)]> for z
1621       have <[v ⊨ x[Π']]>
1622         using 3
1623         by (auto simp: AOT_sem_forall AOT_sem_imp AOT_sem_box AOT_sem_denotes)
1624         (metis (no_types, lifting) 1 2 AOT_term_of_var_cases AOT_sem_box
1625           AOT_sem_denotes AOT_sem_imp)
1626     } note 3 = this
1627     fix Π' :: <<κ>>
1628     assume Πden: <[v ⊨ Π'↓]>
1629     assume 4: <[v ⊨ ∀z (0!z → □([Π']z ≡ [Π]z))>
1630     {
1631       fix κ0
1632       assume <[v ⊨ [λx ◇[E!]x]κ0>
1633       hence <[v ⊨ 0!κ0>
1634         using AOT_sem_ordinary by metis
1635       moreover have <[v ⊨ κ0↓]>
1636       using calculation by (simp add: AOT_sem_exe)

```

```

1637     ultimately have <[v ⊨ □([Π']κ₀ ≡ [Π]κ₀)]>
1638     using 4 by (auto simp: AOT_sem_forall AOT_sem_imp)
1639 } note 4 = this
1640 have <[v ⊨ y[Π']]>
1641   apply (rule AOT_sem_enc_indistinguishable_all[OF AOT_ExtendedModel])
1642   apply (fact 0)
1643   apply (auto simp: 0 1 Π_den indist[simplified AOT_sem_forall
1644     AOT_sem_box AOT_sem_equiv])
1645
1646   apply (rule 3)
1647   apply auto[1]
1648   using 4
1649   by (auto simp: AOT_sem_imp AOT_sem_equiv AOT_sem_box)
1650 }
1651 moreover {
1652 {
1653   assume <[v ⊨ ∀G(∀z (0!z → □([G]z ≡ [Π]z)) → y[G])]>
1654   hence 3: <[v ⊨ ∀G(∀z ([λx ◇[E!]x]z → □([G]z ≡ [Π]z)) → y[G])]>
1655     by (simp add: AOT_sem_ordinary)
1656   {
1657     fix Π' :: <<κ>>
1658     assume 1: <[v ⊨ Π'↓]>
1659     assume 2: <[v ⊨ [λx ◇[E!]x]z → □([Π']z ≡ [Π]z)]> for z
1660     have <[v ⊨ y[Π']]>
1661       using 3
1662       apply (auto simp: AOT_sem_forall AOT_sem_imp AOT_sem_box AOT_sem_denotes)
1663       by (metis (no_types, lifting) 1 2 AOT_model.AOT_term_of_var_cases
1664         AOT_sem_box AOT_sem_denotes AOT_sem_imp)
1665   } note 3 = this
1666   fix Π' :: <<κ>>
1667   assume Π_den: <[v ⊨ Π'↓]>
1668   assume 4: <[v ⊨ ∀z (0!z → □([Π']z ≡ [Π]z))]>
1669   {
1670     fix κ₀
1671     assume <[v ⊨ [λx ◇[E!]x]κ₀]>
1672     hence <[v ⊨ 0!κ₀]>
1673       using AOT_sem_ordinary by metis
1674     moreover have <[v ⊨ κ₀↓]>
1675       using calculation by (simp add: AOT_sem_exe)
1676     ultimately have <[v ⊨ □([Π']κ₀ ≡ [Π]κ₀)]>
1677       using 4 by (auto simp: AOT_sem_forall AOT_sem_imp)
1678   } note 4 = this
1679   have <[v ⊨ x[Π']]>
1680     apply (rule AOT_sem_enc_indistinguishable_all[OF AOT_ExtendedModel])
1681     apply (fact 1)
1682     apply (auto simp: 0 1 Π_den indist[simplified AOT_sem_forall
1683       AOT_sem_box AOT_sem_equiv])
1684
1685     apply (rule 3)
1686     apply auto[1]
1687     using 4
1688     by (auto simp: AOT_sem_imp AOT_sem_equiv AOT_sem_box)
1689   }
1690 }
1691 ultimately show <[v ⊨ ∀G (∀z (0!z → □([G]z ≡ [Π]z)) → x[G])] =
1692   [v ⊨ ∀G (∀z (0!z → □([G]z ≡ [Π]z)) → y[G])>
1693   by (auto simp: AOT_sem_forall AOT_sem_imp)
1694 qed
1695
1696 lemma AOT_sem_indistinguishable_ord_enc_ex:
1697   assumes Π_den: <[v ⊨ Π↓]>
1698   assumes Ax: <[v ⊨ A!x]>
1699   assumes Ay: <[v ⊨ A!y]>
1700   assumes indist: <[v ⊨ ∀F □([F]x ≡ [F]y)]>
1701   shows <[v ⊨ ∃G(∀z (0!z → □([G]z ≡ [Π]z)) & x[G])] =

```



```

1700     [v ⊨ ∃G(∀z(O!z → □([G]z ≡ [II]z)) & y[G])] >
1701 proof -
1702   have Aux: <[v ⊨ [λx ◇[E!]x]κ] = ([v ⊨ [λx ◇[E!]x]κ] ∧ [v ⊨ κ↓])> for v κ
1703   using AOT_sem_exe by blast
1704   AOT_modally_strict {
1705     fix x y
1706     AOT_assume Π_den: <[II]↓>
1707     AOT_assume 2: <∀F □([F]x ≡ [F]y)>
1708     AOT_assume <A!x>
1709     AOT_hence 0: <[λx ¬◇[E!]x]x>
1710       by (simp add: AOT_sem_abstract)
1711     AOT_assume <A!y>
1712     AOT_hence 1: <[λx ¬◇[E!]x]y>
1713       by (simp add: AOT_sem_abstract)
1714     {
1715       AOT_assume <∃G(∀z (O!z → □([G]z ≡ [II]z)) & x[G])>
1716       then AOT_obtain Π'
1717         where Π'_den: <Π'↓>
1718           and Π'_indist: <∀z (O!z → □([Π']z ≡ [II]z))>
1719           and x_enc_Π': <x[Π']>
1720       by (meson AOT_sem_conj AOT_sem_exists)
1721       {
1722         fix κ₀
1723         AOT_assume <[λx ◇[E!]x]κ₀>
1724         AOT_hence <□([Π']κ₀ ≡ [II]κ₀)>
1725           using Π'_indist
1726           by (auto simp: AOT_sem_exe AOT_sem_imp AOT_sem_exists AOT_sem_conj
1727               AOT_sem_ordinary AOT_sem_forall)
1728       } note 3 = this
1729       AOT_have <∀z ([λx ◇[E!]x]z → □([Π']z ≡ [II]z))>
1730         using Π'_indist by (simp add: AOT_sem_ordinary)
1731       AOT_obtain Π'' where
1732         Π''_den: <Π''↓> and
1733         Π''_indist: <[λx ◇[E!]x]κ₀ → □([Π'']κ₀ ≡ [II]κ₀)> and
1734         y_enc_Π'': <y[Π'']> for κ₀
1735       using AOT_sem_enc_indistinguishable_ex[OF AOT_ExtendedModel,
1736         OF 0, OF 1, rotated, OF Π'_den,
1737         OF exI[where x=Π'], OF conjI, OF Π'_den, OF conjI,
1738         OF x_enc_Π', OF allI, OF impI,
1739         OF 3[simplified AOT_sem_box AOT_sem_equiv], simplified, OF
1740         2[simplified AOT_sem_forall AOT_sem_equiv AOT_sem_box,
1741           THEN spec, THEN mp, THEN spec], simplified]
1742       unfolding AOT_sem_imp AOT_sem_box AOT_sem_equiv by blast
1743       {
1744         AOT_have <Π''↓>
1745         and <∀x ([λx ◇[E!]x]x → □([Π'']x ≡ [II]x))>
1746         and <y[Π'']>
1747         apply (simp add: Π''_den)
1748         apply (simp add: AOT_sem_forall Π''_indist)
1749         by (simp add: y_enc_Π'')
1750       } note 2 = this
1751       AOT_have <∃G(∀z (O!z → □([G]z ≡ [II]z)) & y[G])>
1752         apply (auto simp: AOT_sem_exists AOT_sem_ordinary
1753             AOT_sem_imp AOT_sem_box AOT_sem_forall AOT_sem_equiv AOT_sem_conj)
1754         using 2[simplified AOT_sem_box AOT_sem_equiv AOT_sem_imp AOT_sem_forall]
1755         by blast
1756     }
1757   } note 0 = this
1758   AOT_modally_strict {
1759     {
1760       fix x y
1761       AOT_assume Π_den: <[II]↓>
1762       moreover AOT_assume <∀F □([F]x ≡ [F]y)>

```

```

1763     moreover AOT_have < $\forall F \square([F]y \equiv [F]x)$ >
1764     using calculation(2)
1765     by (auto simp: AOT_sem_forall AOT_sem_box AOT_sem_equiv)
1766     moreover AOT_assume <A!x>
1767     moreover AOT_assume <A!y>
1768     ultimately AOT_have < $\exists G (\forall z (O!z \rightarrow \square([G]z \equiv [II]z)) \ \& \ x[G]) \equiv$ 
1769      $\exists G (\forall z (O!z \rightarrow \square([G]z \equiv [II]z)) \ \& \ y[G])$ >
1770     using 0 by (auto simp: AOT_sem_equiv)
1771   }
1772   have 1: <[v]  $\models \forall F \square([F]y \equiv [F]x)$ >
1773     using indist
1774     by (auto simp: AOT_sem_forall AOT_sem_box AOT_sem_equiv)
1775   thus <[v]  $\models \exists G (\forall z (O!z \rightarrow \square([G]z \equiv [II]z)) \ \& \ x[G])$ > =
1776     <[v]  $\models \exists G (\forall z (O!z \rightarrow \square([G]z \equiv [II]z)) \ \& \ y[G])$ >
1777     using assms
1778     by (auto simp: AOT_sem_imp AOT_sem_conj AOT_sem_equiv 0)
1779   }
1780 qed
1781 end
1782
1783
1784 (* Collect all theorems that are not in Main and not declared [AOT]
1785    and store them in a blacklist. *)
1786 setup<setup_AOT_no_atp>
1787 bundle AOT_no_atp begin declare AOT_no_atp[no_atp] end
1788 (* Can be used as: "including AOT_no_atp sledgehammer" or
1789    "sledgehammer(del: AOT_no_atp) *)
1790
1791 (*<*)
1792 end
1793 (*>*)

```

A.5. Definitions of AOT

```

1  theory AOT_Definitions
2    imports AOT_semantics
3  begin
4
5  section<Definitions of AOT>
6
7  AOT_theorem "conventions:1": < $\varphi \ \& \ \psi \equiv_{df} \neg(\varphi \rightarrow \neg\psi)$ > (18.1)
8    using AOT_conj.
9  AOT_theorem "conventions:2": < $\varphi \ \vee \ \psi \equiv_{df} \neg\varphi \rightarrow \psi$ > (18.2)
10   using AOT_disj.
11  AOT_theorem "conventions:3": < $\varphi \equiv \psi \equiv_{df} (\varphi \rightarrow \psi) \ \& \ (\psi \rightarrow \varphi)$ > (18.3)
12   using AOT_equiv.
13  AOT_theorem "conventions:4": < $\exists\alpha \ \varphi\{\alpha\} \equiv_{df} \neg\forall\alpha \ \neg\varphi\{\alpha\}$ > (18.4)
14   using AOT_exists.
15  AOT_theorem "conventions:5": < $\diamond\varphi \equiv_{df} \neg\Box\neg\varphi$ > (18.5)
16   using AOT_dia.
17
18  declare "conventions:1"[AOT_defs] "conventions:2"[AOT_defs]
19         "conventions:3"[AOT_defs] "conventions:4"[AOT_defs]
20         "conventions:5"[AOT_defs]
21
22  notepad
23  begin
24    fix  $\varphi \ \psi \ \chi$ 
25    text<\linealabel{precedence}>
26    have "conventions3[1]": << $\varphi \rightarrow \psi \equiv \neg\psi \rightarrow \neg\varphi$ > = << $(\varphi \rightarrow \psi) \equiv (\neg\psi \rightarrow \neg\varphi)$ >> (19)
27      by blast
28    have "conventions3[2]": << $\varphi \ \& \ \psi \rightarrow \chi$ > = << $(\varphi \ \& \ \psi) \rightarrow \chi$ >> (19)
29      and << $\varphi \ \vee \ \psi \rightarrow \chi$ > = << $(\varphi \ \vee \ \psi) \rightarrow \chi$ >>
30      by blast+
31    have "conventions3[3]": << $\varphi \ \vee \ \psi \ \& \ \chi$ > = << $(\varphi \ \vee \ \psi) \ \& \ \chi$ >> (19)
32      and << $\varphi \ \& \ \psi \ \vee \ \chi$ > = << $(\varphi \ \& \ \psi) \ \vee \ \chi$ >>
33      by blast+ - <Note that PLM instead generally uses parenthesis in these cases.>
34  end
35
36
37  AOT_theorem "existence:1": < $\kappa\downarrow \equiv_{df} \exists F \ [F] \ \kappa$ > (20.1)
38    by (simp add: AOT_sem_denotes AOT_sem_exists AOT_model_equiv_def)
39      (metis AOT_sem_denotes AOT_sem_exe AOT_sem_lambda_beta AOT_sem_lambda_denotes)
40  AOT_theorem "existence:2": < $\Pi\downarrow \equiv_{df} \exists x_1 \dots \exists x_n \ x_1 \dots x_n \ [\Pi]$ > (20.2)
41    using AOT_sem_denotes AOT_sem_enc_denotes AOT_sem_universal_encoder
42    by (simp add: AOT_sem_denotes AOT_sem_exists AOT_model_equiv_def) blast
43  AOT_theorem "existence:2[1]": < $\Pi\downarrow \equiv_{df} \exists x \ x \ [\Pi]$ > (20.2)
44    using "existence:2"[of  $\Pi$ ] by simp
45  AOT_theorem "existence:2[2]": < $\Pi\downarrow \equiv_{df} \exists x \exists y \ xy \ [\Pi]$ > (20.2)
46    using "existence:2"[of  $\Pi$ ]
47    by (simp add: AOT_sem_denotes AOT_sem_exists AOT_model_equiv_def
48              AOT_model_denotes_prod_def)
49  AOT_theorem "existence:2[3]": < $\Pi\downarrow \equiv_{df} \exists x \exists y \exists z \ xyz \ [\Pi]$ > (20.2)
50    using "existence:2"[of  $\Pi$ ]
51    by (simp add: AOT_sem_denotes AOT_sem_exists AOT_model_equiv_def
52              AOT_model_denotes_prod_def)
53  AOT_theorem "existence:2[4]": < $\Pi\downarrow \equiv_{df} \exists x_1 \exists x_2 \exists x_3 \exists x_4 \ x_1 x_2 x_3 x_4 \ [\Pi]$ > (20.2)
54    using "existence:2"[of  $\Pi$ ]
55    by (simp add: AOT_sem_denotes AOT_sem_exists AOT_model_equiv_def
56              AOT_model_denotes_prod_def)
57
58  AOT_theorem "existence:3": < $\varphi\downarrow \equiv_{df} [\lambda x \ \varphi]\downarrow$ > (20.3)
59    by (simp add: AOT_sem_denotes AOT_model_denotes_o_def AOT_model_equiv_def
60              AOT_model_lambda_denotes)
61

```

```

62 declare "existence:1"[AOT_defs] "existence:2"[AOT_defs] "existence:2[1]"[AOT_defs]
63       "existence:2[2]"[AOT_defs] "existence:2[3]"[AOT_defs]
64       "existence:2[4]"[AOT_defs] "existence:3"[AOT_defs]
65
66
67 AOT_theorem "oa:1": <O! =df [λx ◊E!x]> using AOT_ordinary . (22.1)
68 AOT_theorem "oa:2": <A! =df [λx ¬◊E!x]> using AOT_abstract . (22.2)
69
70 declare "oa:1"[AOT_defs] "oa:2"[AOT_defs]
71
72 AOT_theorem "identity:1": (23.1)
73   <x = y ≡df ([O!]x & [O!]y & □∀F ([F]x ≡ [F]y)) ∨
74     ([A!]x & [A!]y & □∀F (x[F] ≡ y[F]))>
75   unfolding AOT_model_equiv_def
76   using AOT_sem_ind_eq[of _ x y]
77   by (simp add: AOT_sem_ordinary AOT_sem_abstract AOT_sem_conj
78         AOT_sem_box AOT_sem_equiv AOT_sem_forall AOT_sem_disj AOT_sem_eq
79         AOT_sem_denotes)
80
81 AOT_theorem "identity:2": (23.2)
82   <F = G ≡df F↓ & G↓ & □∀x(x[F] ≡ x[G])>
83   using AOT_sem_enc_eq[of _ F G]
84   by (auto simp: AOT_model_equiv_def AOT_sem_imp AOT_sem_denotes AOT_sem_eq
85         AOT_sem_conj AOT_sem_forall AOT_sem_box AOT_sem_equiv)
86
87 AOT_theorem "identity:3[2]": (23.3)
88   <F = G ≡df F↓ & G↓ & ∀y([λz [F]zy] = [λz [G]zy] & [λz [F]yz] = [λz [G]yz])>
89   by (auto simp: AOT_model_equiv_def AOT_sem_proj_id_prop[of _ F G]
90         AOT_sem_proj_id_prod_def AOT_sem_conj AOT_sem_denotes
91         AOT_sem_forall AOT_sem_unary_proj_id AOT_model_denotes_prod_def)
92 AOT_theorem "identity:3[3]": (23.3)
93   <F = G ≡df F↓ & G↓ & ∀y1∀y2([λz [F]zy1y2] = [λz [G]zy1y2] &
94     [λz [F]y1zy2] = [λz [G]y1zy2] &
95     [λz [F]y1y2z] = [λz [G]y1y2z])>
96   by (auto simp: AOT_model_equiv_def AOT_sem_proj_id_prop[of _ F G]
97         AOT_sem_proj_id_prod_def AOT_sem_conj AOT_sem_denotes
98         AOT_sem_forall AOT_sem_unary_proj_id AOT_model_denotes_prod_def)
99 AOT_theorem "identity:3[4]": (23.3)
100   <F = G ≡df F↓ & G↓ & ∀y1∀y2∀y3([λz [F]zy1y2y3] = [λz [G]zy1y2y3] &
101     [λz [F]y1zy2y3] = [λz [G]y1zy2y3] &
102     [λz [F]y1y2zy3] = [λz [G]y1y2zy3] &
103     [λz [F]y1y2y3z] = [λz [G]y1y2y3z])>
104   by (auto simp: AOT_model_equiv_def AOT_sem_proj_id_prop[of _ F G]
105         AOT_sem_proj_id_prod_def AOT_sem_conj AOT_sem_denotes
106         AOT_sem_forall AOT_sem_unary_proj_id AOT_model_denotes_prod_def)
107 AOT_theorem "identity:3": (23.3)
108   <F = G ≡df F↓ & G↓ & ∀x1...∀xn «AOT_sem_proj_id x1xn (λ τ . AOT_exe F τ)
109     (λ τ . AOT_exe G τ)»>
110   by (auto simp: AOT_model_equiv_def AOT_sem_proj_id_prop[of _ F G]
111         AOT_sem_proj_id_prod_def AOT_sem_conj AOT_sem_denotes
112         AOT_sem_forall AOT_sem_unary_proj_id AOT_model_denotes_prod_def)
113
114 AOT_theorem "identity:4": (23.4)
115   <p = q ≡df p↓ & q↓ & [λx p] = [λx q]>
116   by (auto simp: AOT_model_equiv_def AOT_sem_eq AOT_sem_denotes AOT_sem_conj
117         AOT_model_lambda_denotes AOT_sem_lambda_eq_prop_eq)
118
119 declare "identity:1"[AOT_defs] "identity:2"[AOT_defs] "identity:3[2]"[AOT_defs]
120       "identity:3[3]"[AOT_defs] "identity:3[4]"[AOT_defs] "identity:3"[AOT_defs]
121       "identity:4"[AOT_defs]
122
123 AOT_define AOT_nonidentical :: <τ ⇒ τ ⇒ φ> (infixl "≠" 50)
124   "=-infix": <τ ≠ σ ≡df ¬(τ = σ)> (24)

```

```

125
126 context AOT_meta_syntax
127 begin
128 notation AOT_nonidentical (infixl "≠" 50)
129 end
130 context AOT_no_meta_syntax
131 begin
132 no_notation AOT_nonidentical (infixl "≠" 50)
133 end
134
135
136 text<The following are purely technical pseudo-definitions required due to
137     our internal implementation of n-ary relations and ellipses using tuples.>
138 AOT_theorem tuple_denotes: <<<( $\tau, \tau'$ )>>\downarrow \equiv_{df} \tau\downarrow \& \tau'\downarrow>
139   by (simp add: AOT_model_denotes_prod_def AOT_model_equiv_def
140         AOT_sem_conj AOT_sem_denotes)
141 AOT_theorem tuple_identity_1: <<<( $\tau, \tau'$ )>> = <<( $\sigma, \sigma'$ )>> \equiv_{df} (\tau = \sigma) \& (\tau' = \sigma')>
142   by (auto simp: AOT_model_equiv_def AOT_sem_conj AOT_sem_eq
143         AOT_model_denotes_prod_def AOT_sem_denotes)
144 AOT_theorem tuple_forall: <<\forall\alpha_1\dots\forall\alpha_n \varphi\{\alpha_1\dots\alpha_n\} \equiv_{df} \forall\alpha_1(\forall\alpha_2\dots\forall\alpha_n \varphi\{\langle\langle\alpha_1, \alpha_2\alpha_n\rangle\rangle\})>>
145   by (auto simp: AOT_model_equiv_def AOT_sem_forall AOT_sem_denotes
146         AOT_model_denotes_prod_def)
147 AOT_theorem tuple_exists: <<\exists\alpha_1\dots\exists\alpha_n \varphi\{\alpha_1\dots\alpha_n\} \equiv_{df} \exists\alpha_1(\exists\alpha_2\dots\exists\alpha_n \varphi\{\langle\langle\alpha_1, \alpha_2\alpha_n\rangle\rangle\})>>
148   by (auto simp: AOT_model_equiv_def AOT_sem_exists AOT_sem_denotes
149         AOT_model_denotes_prod_def)
150 declare tuple_denotes[AOT_defs] tuple_identity_1[AOT_defs] tuple_forall[AOT_defs]
151         tuple_exists[AOT_defs]
152
153 end
154

```

A.6. Axioms of AOT

```

1  (*<*)
2  theory AOT_Axioms
3    imports AOT_Definitions
4  begin
5  (*>*)
6
7  section<Axioms of PLM>
8
9  AOT_axiom "pl:1": < $\varphi \rightarrow (\psi \rightarrow \varphi)$ > (38.1)
10 by (auto simp: AOT_sem_imp AOT_model_axiomI)
11 AOT_axiom "pl:2": < $(\varphi \rightarrow (\psi \rightarrow \chi)) \rightarrow ((\varphi \rightarrow \psi) \rightarrow (\varphi \rightarrow \chi))$ > (38.2)
12 by (auto simp: AOT_sem_imp AOT_model_axiomI)
13 AOT_axiom "pl:3": < $(\neg\varphi \rightarrow \neg\psi) \rightarrow ((\neg\varphi \rightarrow \psi) \rightarrow \varphi)$ > (38.3)
14 by (auto simp: AOT_sem_imp AOT_sem_not AOT_model_axiomI)
15
16 AOT_axiom "cqt:1": < $\forall\alpha \varphi\{\alpha\} \rightarrow (\tau\downarrow \rightarrow \varphi\{\tau\})$ > (39.1)
17 by (auto simp: AOT_sem_denotes AOT_sem_forall AOT_sem_imp AOT_model_axiomI)
18
19 AOT_axiom "cqt:2[const_var]": < $\alpha\downarrow$ > (39.2)
20 using AOT_sem_vars_denote by (rule AOT_model_axiomI)
21 AOT_axiom "cqt:2[lambda]": (39.2)
22   assumes <INSTANCE_OF_CQT_2( $\varphi$ )>
23   shows < $[\lambda\nu_1 \dots \nu_n \varphi\{\nu_1 \dots \nu_n\}]\downarrow$ >
24   by (auto intro!: AOT_model_axiomI AOT_sem_cqt_2[OF assms])
25 AOT_axiom "cqt:2[lambda0]": (39.2)
26   shows < $[\lambda \varphi]\downarrow$ >
27   by (auto intro!: AOT_model_axiomI
28       simp: AOT_sem_lambda_denotes "existence:3"[unfolded AOT_model_equiv_def])
29
30 AOT_axiom "cqt:3": < $\forall\alpha (\varphi\{\alpha\} \rightarrow \psi\{\alpha\}) \rightarrow (\forall\alpha \varphi\{\alpha\} \rightarrow \forall\alpha \psi\{\alpha\})$ > (39.3)
31 by (simp add: AOT_sem_forall AOT_sem_imp AOT_model_axiomI)
32 AOT_axiom "cqt:4": < $\varphi \rightarrow \forall\alpha \varphi$ > (39.4)
33 by (simp add: AOT_sem_forall AOT_sem_imp AOT_model_axiomI)
34 AOT_axiom "cqt:5:a": < $[\Pi] \kappa_1 \dots \kappa_n \rightarrow (\Pi\downarrow \ \& \ \kappa_1 \dots \kappa_n\downarrow)$ > (39.5.a)
35 by (simp add: AOT_sem_conj AOT_sem_denotes AOT_sem_exe
36     AOT_sem_imp AOT_model_axiomI)
37 AOT_axiom "cqt:5:a[1]": < $[\Pi] \kappa \rightarrow (\Pi\downarrow \ \& \ \kappa\downarrow)$ > (39.5.a)
38 using "cqt:5:a" AOT_model_axiomI by blast
39 AOT_axiom "cqt:5:a[2]": < $[\Pi] \kappa_1 \kappa_2 \rightarrow (\Pi\downarrow \ \& \ \kappa_1\downarrow \ \& \ \kappa_2\downarrow)$ > (39.5.a)
40 by (rule AOT_model_axiomI)
41   (metis AOT_model_denotes_prod_def AOT_sem_conj AOT_sem_denotes AOT_sem_exe
42     AOT_sem_imp case_prodd)
43 AOT_axiom "cqt:5:a[3]": < $[\Pi] \kappa_1 \kappa_2 \kappa_3 \rightarrow (\Pi\downarrow \ \& \ \kappa_1\downarrow \ \& \ \kappa_2\downarrow \ \& \ \kappa_3\downarrow)$ > (39.5.a)
44 by (rule AOT_model_axiomI)
45   (metis AOT_model_denotes_prod_def AOT_sem_conj AOT_sem_denotes AOT_sem_exe
46     AOT_sem_imp case_prodd)
47 AOT_axiom "cqt:5:a[4]": < $[\Pi] \kappa_1 \kappa_2 \kappa_3 \kappa_4 \rightarrow (\Pi\downarrow \ \& \ \kappa_1\downarrow \ \& \ \kappa_2\downarrow \ \& \ \kappa_3\downarrow \ \& \ \kappa_4\downarrow)$ > (39.5.a)
48 by (rule AOT_model_axiomI)
49   (metis AOT_model_denotes_prod_def AOT_sem_conj AOT_sem_denotes AOT_sem_exe
50     AOT_sem_imp case_prodd)
51 AOT_axiom "cqt:5:b": < $\kappa_1 \dots \kappa_n [\Pi] \rightarrow (\Pi\downarrow \ \& \ \kappa_1 \dots \kappa_n\downarrow)$ > (39.5.b)
52 using AOT_sem_enc_denotes
53 by (auto intro!: AOT_model_axiomI simp: AOT_sem_conj AOT_sem_denotes AOT_sem_imp)+
54 AOT_axiom "cqt:5:b[1]": < $\kappa [\Pi] \rightarrow (\Pi\downarrow \ \& \ \kappa\downarrow)$ > (39.5.b)
55 using "cqt:5:b" AOT_model_axiomI by blast
56 AOT_axiom "cqt:5:b[2]": < $\kappa_1 \kappa_2 [\Pi] \rightarrow (\Pi\downarrow \ \& \ \kappa_1\downarrow \ \& \ \kappa_2\downarrow)$ > (39.5.b)
57 by (rule AOT_model_axiomI)
58   (metis AOT_model_denotes_prod_def AOT_sem_conj AOT_sem_denotes
59     AOT_sem_enc_denotes AOT_sem_imp case_prodd)
60 AOT_axiom "cqt:5:b[3]": < $\kappa_1 \kappa_2 \kappa_3 [\Pi] \rightarrow (\Pi\downarrow \ \& \ \kappa_1\downarrow \ \& \ \kappa_2\downarrow \ \& \ \kappa_3\downarrow)$ > (39.5.b)
61 by (rule AOT_model_axiomI)

```

```

62     (metis AOT_model_denotes_prod_def AOT_sem_conj AOT_sem_denotes
63           AOT_sem_enc_denotes AOT_sem_imp case_prodd)
64 AOT_axiom "cqt:5:b[4]": < $\kappa_1\kappa_2\kappa_3\kappa_4$  [II]  $\rightarrow$  (II $\downarrow$  &  $\kappa_1\downarrow$  &  $\kappa_2\downarrow$  &  $\kappa_3\downarrow$  &  $\kappa_4\downarrow$ )> (39.5.b)
65   by (rule AOT_model_axiomI)
66     (metis AOT_model_denotes_prod_def AOT_sem_conj AOT_sem_denotes
67           AOT_sem_enc_denotes AOT_sem_imp case_prodd)
68
69 AOT_axiom "l-identity": < $\alpha = \beta \rightarrow (\varphi\{\alpha\} \rightarrow \varphi\{\beta\})$ > (41)
70   by (rule AOT_model_axiomI)
71     (simp add: AOT_sem_eq AOT_sem_imp)
72
73 AOT_act_axiom "logic-actual": < $\mathcal{A}\varphi \rightarrow \varphi$ > (43)
74   by (rule AOT_model_act_axiomI)
75     (simp add: AOT_sem_act AOT_sem_imp)
76
77 AOT_axiom "logic-actual-nec:1": < $\mathcal{A}\neg\varphi \equiv \neg\mathcal{A}\varphi$ > (44.1)
78   by (rule AOT_model_axiomI)
79     (simp add: AOT_sem_act AOT_sem_equiv AOT_sem_not)
80 AOT_axiom "logic-actual-nec:2": < $\mathcal{A}(\varphi \rightarrow \psi) \equiv (\mathcal{A}\varphi \rightarrow \mathcal{A}\psi)$ > (44.2)
81   by (rule AOT_model_axiomI)
82     (simp add: AOT_sem_act AOT_sem_equiv AOT_sem_imp)
83
84 AOT_axiom "logic-actual-nec:3": < $\mathcal{A}(\forall\alpha \varphi\{\alpha\}) \equiv \forall\alpha \mathcal{A}\varphi\{\alpha\}$ > (44.3)
85   by (rule AOT_model_axiomI)
86     (simp add: AOT_sem_act AOT_sem_equiv AOT_sem_forall AOT_sem_denotes)
87 AOT_axiom "logic-actual-nec:4": < $\mathcal{A}\varphi \equiv \mathcal{A}\mathcal{A}\varphi$ > (44.4)
88   by (rule AOT_model_axiomI)
89     (simp add: AOT_sem_act AOT_sem_equiv)
90
91 AOT_axiom "qml:1": < $\Box(\varphi \rightarrow \psi) \rightarrow (\Box\varphi \rightarrow \Box\psi)$ > (45.1)
92   by (rule AOT_model_axiomI)
93     (simp add: AOT_sem_box AOT_sem_imp)
94 AOT_axiom "qml:2": < $\Box\varphi \rightarrow \varphi$ > (45.2)
95   by (rule AOT_model_axiomI)
96     (simp add: AOT_sem_box AOT_sem_imp)
97 AOT_axiom "qml:3": < $\Diamond\varphi \rightarrow \Box\Diamond\varphi$ > (45.3)
98   by (rule AOT_model_axiomI)
99     (simp add: AOT_sem_box AOT_sem_dia AOT_sem_imp)
100
101 AOT_axiom "qml:4": < $\Diamond\exists x (E!x \ \& \ \neg\mathcal{A}E!x)$ > (45.4)
102   using AOT_sem_concrete AOT_model_contingent
103   by (auto intro!: AOT_model_axiomI
104       simp: AOT_sem_box AOT_sem_dia AOT_sem_imp AOT_sem_exists
105           AOT_sem_denotes AOT_sem_conj AOT_sem_not AOT_sem_act
106           AOT_sem_exe)+
107
108 AOT_axiom "qml-act:1": < $\mathcal{A}\varphi \rightarrow \Box\mathcal{A}\varphi$ > (46.1)
109   by (rule AOT_model_axiomI)
110     (simp add: AOT_sem_act AOT_sem_box AOT_sem_imp)
111 AOT_axiom "qml-act:2": < $\Box\varphi \equiv \mathcal{A}\Box\varphi$ > (46.2)
112   by (rule AOT_model_axiomI)
113     (simp add: AOT_sem_act AOT_sem_box AOT_sem_equiv)
114
115 AOT_axiom descriptions: < $x = \iota x(\varphi\{x\}) \equiv \forall z(\mathcal{A}\varphi\{z\} \equiv z = x)$ > (47)
116 proof (rule AOT_model_axiomI)
117   AOT_modally_strict {
118     AOT_show < $x = \iota x(\varphi\{x\}) \equiv \forall z(\mathcal{A}\varphi\{z\} \equiv z = x)$ >
119     by (induct; simp add: AOT_sem_equiv AOT_sem_forall AOT_sem_act AOT_sem_eq
120         (metis (no_types, opaque_lifting) AOT_sem_desc_denotes AOT_sem_desc_prop
121             AOT_sem_denotes))
122   }
123 qed
124

```

```

125 AOT_axiom "lambda-predicates:1": (48.1)
126   <[ $\lambda\nu_1 \dots \nu_n \varphi\{\nu_1 \dots \nu_n\}$ ] $\downarrow$   $\rightarrow$  [ $\lambda\nu_1 \dots \nu_n \varphi\{\nu_1 \dots \nu_n\}$ ] = [ $\lambda\mu_1 \dots \mu_n \varphi\{\mu_1 \dots \mu_n\}$ ] $\rangle$ 
127   by (rule AOT_model_axiomI)
128     (simp add: AOT_sem_denotes AOT_sem_eq AOT_sem_imp)
129 AOT_axiom "lambda-predicates:1[zero]": <[ $\lambda p$ ] $\downarrow$   $\rightarrow$  [ $\lambda p$ ] = [ $\lambda p$ ] $\rangle$  (48.1)
130   by (rule AOT_model_axiomI)
131     (simp add: AOT_sem_denotes AOT_sem_eq AOT_sem_imp)
132 AOT_axiom "lambda-predicates:2": (48.2)
133   <[ $\lambda x_1 \dots x_n \varphi\{x_1 \dots x_n\}$ ] $\downarrow$   $\rightarrow$  ([ $\lambda x_1 \dots x_n \varphi\{x_1 \dots x_n\}$ ] $x_1 \dots x_n$   $\equiv$   $\varphi\{x_1 \dots x_n\}$ ) $\rangle$ 
134   by (rule AOT_model_axiomI)
135     (simp add: AOT_sem_equiv AOT_sem_imp AOT_sem_lambda_beta AOT_sem_vars_denote)
136 AOT_axiom "lambda-predicates:3": <[ $\lambda x_1 \dots x_n [F]x_1 \dots x_n$ ] =  $F$  $\rangle$  (48.3)
137   by (rule AOT_model_axiomI)
138     (simp add: AOT_sem_lambda_eta AOT_sem_vars_denote)
139 AOT_axiom "lambda-predicates:3[zero]": <[ $\lambda p$ ] =  $p$  $\rangle$  (48.3)
140   by (rule AOT_model_axiomI)
141     (simp add: AOT_sem_eq AOT_sem_lambda0 AOT_sem_vars_denote)
142
143 AOT_axiom "safe-ext": (49)
144   <([ $\lambda\nu_1 \dots \nu_n \varphi\{\nu_1 \dots \nu_n\}$ ] $\downarrow$  &  $\Box \forall \nu_1 \dots \forall \nu_n (\varphi\{\nu_1 \dots \nu_n\} \equiv \psi\{\nu_1 \dots \nu_n\})$ )  $\rightarrow$ 
145   [ $\lambda\nu_1 \dots \nu_n \psi\{\nu_1 \dots \nu_n\}$ ] $\downarrow$  $\rangle$ 
146   using AOT_sem_lambda_coex
147   by (auto intro!: AOT_model_axiomI simp: AOT_sem_imp AOT_sem_denotes AOT_sem_conj
148       AOT_sem_equiv AOT_sem_box AOT_sem_forall)
149 AOT_axiom "safe-ext[2]": (49)
150   <([ $\lambda\nu_1 \nu_2 \varphi\{\nu_1, \nu_2\}$ ] $\downarrow$  &  $\Box \forall \nu_1 \forall \nu_2 (\varphi\{\nu_1, \nu_2\} \equiv \psi\{\nu_1, \nu_2\})$ )  $\rightarrow$ 
151   [ $\lambda\nu_1 \nu_2 \psi\{\nu_1, \nu_2\}$ ] $\downarrow$  $\rangle$ 
152   using "safe-ext"[where  $\varphi = \lambda(x,y). \varphi x y$ ]
153   by (simp add: AOT_model_axiom_def AOT_sem_denotes AOT_model_denotes_prod_def
154       AOT_sem_forall AOT_sem_imp AOT_sem_conj AOT_sem_equiv AOT_sem_box)
155 AOT_axiom "safe-ext[3]": (49)
156   <([ $\lambda\nu_1 \nu_2 \nu_3 \varphi\{\nu_1, \nu_2, \nu_3\}$ ] $\downarrow$  &  $\Box \forall \nu_1 \forall \nu_2 \forall \nu_3 (\varphi\{\nu_1, \nu_2, \nu_3\} \equiv \psi\{\nu_1, \nu_2, \nu_3\})$ )  $\rightarrow$ 
157   [ $\lambda\nu_1 \nu_2 \nu_3 \psi\{\nu_1, \nu_2, \nu_3\}$ ] $\downarrow$  $\rangle$ 
158   using "safe-ext"[where  $\varphi = \lambda(x,y,z). \varphi x y z$ ]
159   by (simp add: AOT_model_axiom_def AOT_model_denotes_prod_def AOT_sem_forall
160       AOT_sem_denotes AOT_sem_imp AOT_sem_conj AOT_sem_equiv AOT_sem_box)
161 AOT_axiom "safe-ext[4]": (49)
162   <([ $\lambda\nu_1 \nu_2 \nu_3 \nu_4 \varphi\{\nu_1, \nu_2, \nu_3, \nu_4\}$ ] $\downarrow$  &
163    $\Box \forall \nu_1 \forall \nu_2 \forall \nu_3 \forall \nu_4 (\varphi\{\nu_1, \nu_2, \nu_3, \nu_4\} \equiv \psi\{\nu_1, \nu_2, \nu_3, \nu_4\})$ )  $\rightarrow$ 
164   [ $\lambda\nu_1 \nu_2 \nu_3 \nu_4 \psi\{\nu_1, \nu_2, \nu_3, \nu_4\}$ ] $\downarrow$  $\rangle$ 
165   using "safe-ext"[where  $\varphi = \lambda(x,y,z,w). \varphi x y z w$ ]
166   by (simp add: AOT_model_axiom_def AOT_model_denotes_prod_def AOT_sem_forall
167       AOT_sem_denotes AOT_sem_imp AOT_sem_conj AOT_sem_equiv AOT_sem_box)
168
169 AOT_axiom "nary-encoding[2]": (50)
170   < $x_1 x_2 [F] \equiv x_1 [\lambda y [F]y x_2]$  &  $x_2 [\lambda y [F]x_1 y]$  $\rangle$ 
171   by (rule AOT_model_axiomI)
172     (simp add: AOT_sem_conj AOT_sem_equiv AOT_enc_prod_def AOT_proj_enc_prod_def
173       AOT_sem_unary_proj_enc AOT_sem_vars_denote)
174 AOT_axiom "nary-encoding[3]": (50)
175   < $x_1 x_2 x_3 [F] \equiv x_1 [\lambda y [F]y x_2 x_3]$  &  $x_2 [\lambda y [F]x_1 y x_3]$  &  $x_3 [\lambda y [F]x_1 x_2 y]$  $\rangle$ 
176   by (rule AOT_model_axiomI)
177     (simp add: AOT_sem_conj AOT_sem_equiv AOT_enc_prod_def AOT_proj_enc_prod_def
178       AOT_sem_unary_proj_enc AOT_sem_vars_denote)
179 AOT_axiom "nary-encoding[4]": (50)
180   < $x_1 x_2 x_3 x_4 [F] \equiv x_1 [\lambda y [F]y x_2 x_3 x_4]$  &
181      $x_2 [\lambda y [F]x_1 y x_3 x_4]$  &
182      $x_3 [\lambda y [F]x_1 x_2 y x_4]$  &
183      $x_4 [\lambda y [F]x_1 x_2 x_3 y]$  $\rangle$ 
184   by (rule AOT_model_axiomI)
185     (simp add: AOT_sem_conj AOT_sem_equiv AOT_enc_prod_def AOT_proj_enc_prod_def
186       AOT_sem_unary_proj_enc AOT_sem_vars_denote)
187

```



```

188 AOT_axiom encoding: <x[F] → □x[F]> (51)
189   using AOT_sem_enc_nec
190   by (auto intro!: AOT_model_axiomI simp: AOT_sem_imp AOT_sem_box)
191
192 AOT_axiom nocoder: <O!x → ¬∃F x[F]> (52)
193   by (auto intro!: AOT_model_axiomI
194       simp: AOT_sem_imp AOT_sem_not AOT_sem_exists AOT_sem_ordinary
195            AOT_sem_dia
196            AOT_sem_lambda_beta[OF AOT_sem_ordinary_def_denotes,
197                                   OF AOT_sem_vars_denote])
198   (metis AOT_sem_nocoder)
199
200 AOT_axiom "A-objects": <∃x (A!x & ∀F(x[F] ≡ φ{F}))> (53)
201 proof(rule AOT_model_axiomI)
202   AOT_modally_strict {
203     AOT_obtain κ where <κ↓ & □¬E!κ & ∀F (κ[F] ≡ φ{F})>
204     using AOT_sem_A_objects[of _ φ]
205     by (auto simp: AOT_sem_imp AOT_sem_box AOT_sem_forall AOT_sem_exists
206              AOT_sem_conj AOT_sem_not AOT_sem_dia AOT_sem_denotes
207              AOT_sem_equiv) blast
208     AOT_thus <∃x (A!x & ∀F(x[F] ≡ φ{F}))>
209     unfolding AOT_sem_exists
210     by (auto intro!: exI[where x=κ]
211         simp: AOT_sem_lambda_beta[OF AOT_sem_abstract_def_denotes]
212              AOT_sem_box AOT_sem_dia AOT_sem_not AOT_sem_denotes
213              AOT_var_of_term_inverse AOT_sem_conj
214              AOT_sem_equiv AOT_sem_forall AOT_sem_abstract)
215   }
216 qed
217
218 AOT_theorem universal_closure:
219   assumes <for arbitrary α: φ{α} ∈ Λ□>
220   shows <∀α φ{α} ∈ Λ□>
221   using assms
222   by (metis AOT_term_of_var_cases AOT_model_axiom_def AOT_sem_denotes AOT_sem_forall)
223
224 AOT_theorem act_closure:
225   assumes <φ ∈ Λ□>
226   shows <Aφ ∈ Λ□>
227   using assms by (simp add: AOT_model_axiom_def AOT_sem_act)
228
229 AOT_theorem nec_closure:
230   assumes <φ ∈ Λ□>
231   shows <□φ ∈ Λ□>
232   using assms by (simp add: AOT_model_axiom_def AOT_sem_box)
233
234 AOT_theorem universal_closure_act:
235   assumes <for arbitrary α: φ{α} ∈ Λ>
236   shows <∀α φ{α} ∈ Λ>
237   using assms
238   by (metis AOT_term_of_var_cases AOT_model_act_axiom_def AOT_sem_denotes
239           AOT_sem_forall)
240
241 text<The following are not part of PLM and only hold in the extended models.
242       They are a generalization of the predecessor axiom.>
243 context AOT_ExtendedModel
244 begin
245 AOT_axiom indistinguishable_ord_enc_all:
246   <II↓ & A!x & A!y & ∀F □([F]x ≡ [F]y) →
247   ((∀G(∀z(O!z → □([G]z ≡ [II]z)) → x[G])) ≡
248    ∀G(∀z(O!z → □([G]z ≡ [II]z)) → y[G]))>
249   by (rule AOT_model_axiomI)
250   (auto simp: AOT_sem_equiv AOT_sem_imp AOT_sem_conj

```

```
251         AOT_sem_indistinguishable_ord_enc_all)
252 AOT_axiom indistinguishable_ord_enc_ex:
253   <II↓ & A!x & A!y & ∀F □([F]x ≡ [F]y) →
254   ((∃G(∀z(O!z → □([G]z ≡ [II]z)) & x[G])) ≡
255   ∃G(∀z(O!z → □([G]z ≡ [II]z)) & y[G]))>
256   by (rule AOT_model_axiomI)
257     (auto simp: AOT_sem_equiv AOT_sem_imp AOT_sem_conj
258       AOT_sem_indistinguishable_ord_enc_ex)
259 end
260
261 (*<*)
262 end
263 (*>*)
```

A.7. The Deductive System PLM

```

1  (*<*)
2  theory AOT_PLM
3    imports AOT_Axioms
4  begin
5  (*>*)
6
7  section<The Deductive System PLM>
8  text<\label{PLM: 9}>
9
10 (* constrain sledgehammer to the abstraction layer *)
11 unbundle AOT_no_atp
12
13 subsection<Primitive Rule of PLM: Modus Ponens>
14 text<\label{PLM: 9.1}>
15
16 AOT_theorem "modus-ponens":
17   assumes < $\varphi$ > and < $\varphi \rightarrow \psi$ >
18   shows < $\psi$ >
19   (* NOTE: semantics needed *)
20   using assms by (simp add: AOT_sem_imp)
21 lemmas MP = "modus-ponens"
22
23 subsection<(Modally Strict) Proofs and Derivations>
24 text<\label{PLM: 9.2}>
25
26 AOT_theorem "non-con-thm-thm":
27   assumes < $\vdash_{\square} \varphi$ >
28   shows < $\vdash \varphi$ >
29   using assms by simp
30
31 AOT_theorem "vdash-properties:1[1]":
32   assumes < $\varphi \in \Lambda$ >
33   shows < $\vdash \varphi$ >
34   (* NOTE: semantics needed *)
35   using assms unfolding AOT_model_act_axiom_def by blast
36
37 text<Convenience attribute for instantiating modally-fragile axioms.>
38 attribute_setup act_axiom_inst =
39   <Scan.succeed (Thm.rule_attribute []
40     (K (fn thm => thm RS @{thm "vdash-properties:1[1]"}))>>
41   "Instantiate modally fragile axiom as modally fragile theorem."
42
43 AOT_theorem "vdash-properties:1[2]":
44   assumes < $\varphi \in \Lambda_{\square}$ >
45   shows < $\vdash_{\square} \varphi$ >
46   (* NOTE: semantics needed *)
47   using assms unfolding AOT_model_axiom_def by blast
48
49 text<Convenience attribute for instantiating modally-strict axioms.>
50 attribute_setup axiom_inst =
51   <Scan.succeed (Thm.rule_attribute []
52     (K (fn thm => thm RS @{thm "vdash-properties:1[2]"}))>>
53   "Instantiate axiom as theorem."
54
55 text<Convenience methods and theorem sets for applying "cqt:2".>
56 method cqt_2_lambda_inst_prover =
57   (fast intro: AOT_instance_of_cqt_2_intro)
58 method "cqt:2[lambda]" =
59   (rule "cqt:2[lambda]"[axiom_inst]; cqt_2_lambda_inst_prover)
60 lemmas "cqt:2" =
61   "cqt:2[const_var]"[axiom_inst] "cqt:2[lambda]"[axiom_inst]

```

```

62   AOT_instance_of_cqt_2_intro
63 method "cqt:2" = (safe intro!: "cqt:2")
64
65 AOT_theorem "vdash-properties:3":                                     (63.3)
66   assumes <math>\vdash_{\Box} \varphi</math>
67   shows <math>\Gamma \vdash \varphi</math>
68   using assms by blast
69
70 AOT_theorem "vdash-properties:5":                                     (63.5)
71   assumes <math>\langle \Gamma_1 \vdash \varphi \rangle</math> and <math>\langle \Gamma_2 \vdash \varphi \rightarrow \psi \rangle</math>
72   shows <math>\langle \Gamma_1, \Gamma_2 \vdash \psi \rangle</math>
73   using MP assms by blast
74
75 AOT_theorem "vdash-properties:6":                                     (63.6)
76   assumes <math>\langle \varphi \rangle</math> and <math>\langle \varphi \rightarrow \psi \rangle</math>
77   shows <math>\langle \psi \rangle</math>
78   using MP assms by blast
79
80 AOT_theorem "vdash-properties:8":                                     (63.8)
81   assumes <math>\langle \Gamma \vdash \varphi \rangle</math> and <math>\langle \varphi \vdash \psi \rangle</math>
82   shows <math>\langle \Gamma \vdash \psi \rangle</math>
83   using assms by argo
84
85 AOT_theorem "vdash-properties:9":                                     (63.9)
86   assumes <math>\langle \varphi \rangle</math>
87   shows <math>\langle \psi \rightarrow \varphi \rangle</math>
88   using MP "pl:1"[axiom_inst] assms by blast
89
90 AOT_theorem "vdash-properties:10":                                   (63.10)
91   assumes <math>\langle \varphi \rightarrow \psi \rangle</math> and <math>\langle \varphi \rangle</math>
92   shows <math>\langle \psi \rangle</math>
93   using MP assms by blast
94 lemmas "→E" = "vdash-properties:10"
95
96 subsection<Two Fundamental Metarules: GEN and RN>
97 text<\label{PLM: 9.3}>
98
99 AOT_theorem "rule-gen":                                             (66)
100  assumes <math>\langle \text{for arbitrary } \alpha: \varphi\{\alpha\} \rangle</math>
101  shows <math>\langle \forall \alpha \varphi\{\alpha\} \rangle</math>
102  (* NOTE: semantics needed *)
103  using assms by (metis AOT_var_of_term_inverse AOT_sem_denotes AOT_sem_forall)
104  lemmas GEN = "rule-gen"
105
106 AOT_theorem "RN[prem]":                                             (68)
107  assumes <math>\langle \Gamma \vdash_{\Box} \varphi \rangle</math>
108  shows <math>\langle \Box \Gamma \vdash_{\Box} \Box \varphi \rangle</math>
109  by (meson AOT_sem_box assms image_iff) (* NOTE: semantics needed *)
110 AOT_theorem RN:                                                     (68)
111  assumes <math>\langle \vdash_{\Box} \varphi \rangle</math>
112  shows <math>\langle \Box \varphi \rangle</math>
113  using "RN[prem]" assms by blast
114
115 subsection<The Inferential Role of Definitions>
116 text<\label{PLM: 9.4}>
117
118 AOT_axiom "df-rules-formulas[1]":                                    (72)
119  assumes <math>\langle \varphi \equiv_{df} \psi \rangle</math>
120  shows <math>\langle \varphi \rightarrow \psi \rangle</math>
121  (* NOTE: semantics needed *)
122  using assms
123  by (auto simp: assms AOT_model_axiomI AOT_model_equiv_def AOT_sem_imp)
124 AOT_axiom "df-rules-formulas[2]":                                    (72)

```

```

125   assumes < $\varphi \equiv_{df} \psi$ >
126   shows < $\psi \rightarrow \varphi$ >
127   (* NOTE: semantics needed *)
128   using assms
129   by (auto simp: AOT_model_axiomI AOT_model_equiv_def AOT_sem_imp)
130   (* NOTE: for convenience also state the above as regular theorems *)
131   AOT_theorem "df-rules-formulas[3]": (72)
132     assumes < $\varphi \equiv_{df} \psi$ >
133     shows < $\varphi \rightarrow \psi$ >
134     using "df-rules-formulas[1]"[axiom_inst, OF assms].
135   AOT_theorem "df-rules-formulas[4]": (72)
136     assumes < $\varphi \equiv_{df} \psi$ >
137     shows < $\psi \rightarrow \varphi$ >
138     using "df-rules-formulas[2]"[axiom_inst, OF assms].
139
140
141   AOT_axiom "df-rules-terms[1]": (73)
142     assumes < $\tau\{\alpha_1 \dots \alpha_n\} =_{df} \sigma\{\alpha_1 \dots \alpha_n\}$ >
143     shows <( $\sigma\{\tau_1 \dots \tau_n\} \downarrow \rightarrow \tau\{\tau_1 \dots \tau_n\} = \sigma\{\tau_1 \dots \tau_n\}$ ) &
144           ( $\neg\sigma\{\tau_1 \dots \tau_n\} \downarrow \rightarrow \neg\tau\{\tau_1 \dots \tau_n\} \downarrow$ )>
145     (* NOTE: semantics needed *)
146     using assms
147     by (simp add: AOT_model_axiomI AOT_sem_conj AOT_sem_imp AOT_sem_eq
148             AOT_sem_not AOT_sem_denotes AOT_model_id_def)
149   AOT_axiom "df-rules-terms[2]": (73)
150     assumes < $\tau =_{df} \sigma$ >
151     shows <( $\sigma \downarrow \rightarrow \tau = \sigma$ ) & ( $\neg\sigma \downarrow \rightarrow \neg\tau \downarrow$ )>
152     by (metis "df-rules-terms[1]" case_unit_Unity assms)
153   (* NOTE: for convenience also state the above as regular theorems *)
154   AOT_theorem "df-rules-terms[3]": (73)
155     assumes < $\tau\{\alpha_1 \dots \alpha_n\} =_{df} \sigma\{\alpha_1 \dots \alpha_n\}$ >
156     shows <( $\sigma\{\tau_1 \dots \tau_n\} \downarrow \rightarrow \tau\{\tau_1 \dots \tau_n\} = \sigma\{\tau_1 \dots \tau_n\}$ ) &
157           ( $\neg\sigma\{\tau_1 \dots \tau_n\} \downarrow \rightarrow \neg\tau\{\tau_1 \dots \tau_n\} \downarrow$ )>
158     using "df-rules-terms[1]"[axiom_inst, OF assms].
159   AOT_theorem "df-rules-terms[4]": (73)
160     assumes < $\tau =_{df} \sigma$ >
161     shows <( $\sigma \downarrow \rightarrow \tau = \sigma$ ) & ( $\neg\sigma \downarrow \rightarrow \neg\tau \downarrow$ )>
162     using "df-rules-terms[2]"[axiom_inst, OF assms].
163
164   subsection<The Theory of Negations and Conditionals>
165   text<\label{PLM: 9.5}>
166
167   AOT_theorem "if-p-then-p": < $\varphi \rightarrow \varphi$ > (74)
168     by (meson "pl:1"[axiom_inst] "pl:2"[axiom_inst] MP)
169
170   AOT_theorem "deduction-theorem": (75)
171     assumes < $\varphi \vdash \psi$ >
172     shows < $\varphi \rightarrow \psi$ >
173     (* NOTE: semantics needed *)
174     using assms by (simp add: AOT_sem_imp)
175   lemmas CP = "deduction-theorem"
176   lemmas " $\rightarrow$ I" = "deduction-theorem"
177
178   AOT_theorem "ded-thm-cor:1": (76.1)
179     assumes < $\Gamma_1 \vdash \varphi \rightarrow \psi$ > and < $\Gamma_2 \vdash \psi \rightarrow \chi$ >
180     shows < $\Gamma_1, \Gamma_2 \vdash \varphi \rightarrow \chi$ >
181     using " $\rightarrow$ E" " $\rightarrow$ I" assms by blast
182   AOT_theorem "ded-thm-cor:2": (76.2)
183     assumes < $\Gamma_1 \vdash \varphi \rightarrow (\psi \rightarrow \chi)$ > and < $\Gamma_2 \vdash \psi$ >
184     shows < $\Gamma_1, \Gamma_2 \vdash \varphi \rightarrow \chi$ >
185     using " $\rightarrow$ E" " $\rightarrow$ I" assms by blast
186
187   AOT_theorem "ded-thm-cor:3": (76.3)

```

```

188   assumes <math>\varphi \rightarrow \psi</math> and <math>\psi \rightarrow \chi</math>
189   shows <math>\varphi \rightarrow \chi</math>
190   using "→E" "→I" assms by blast
191 declare "ded-thm-cor:3"[trans]
192 AOT_theorem "ded-thm-cor:4":
193   assumes <math>\varphi \rightarrow (\psi \rightarrow \chi)</math> and <math>\psi</math>
194   shows <math>\varphi \rightarrow \chi</math>
195   using "→E" "→I" assms by blast
196
197 lemmas "Hypothetical Syllogism" = "ded-thm-cor:3"
198
199 AOT_theorem "useful-tautologies:1": <math>\neg\neg\varphi \rightarrow \varphi</math>
200   by (metis "pl:3"[axiom_inst] "→I" "Hypothetical Syllogism")
201 AOT_theorem "useful-tautologies:2": <math>\varphi \rightarrow \neg\neg\varphi</math>
202   by (metis "pl:3"[axiom_inst] "→I" "ded-thm-cor:4")
203 AOT_theorem "useful-tautologies:3": <math>\neg\varphi \rightarrow (\varphi \rightarrow \psi)</math>
204   by (meson "ded-thm-cor:4" "pl:3"[axiom_inst] "→I")
205 AOT_theorem "useful-tautologies:4": <math>(\neg\psi \rightarrow \neg\varphi) \rightarrow (\varphi \rightarrow \psi)</math>
206   by (meson "pl:3"[axiom_inst] "Hypothetical Syllogism" "→I")
207 AOT_theorem "useful-tautologies:5": <math>(\varphi \rightarrow \psi) \rightarrow (\neg\psi \rightarrow \neg\varphi)</math>
208   by (metis "useful-tautologies:4" "Hypothetical Syllogism" "→I")
209
210 AOT_theorem "useful-tautologies:6": <math>(\varphi \rightarrow \neg\psi) \rightarrow (\psi \rightarrow \neg\varphi)</math>
211   by (metis "→I" MP "useful-tautologies:4")
212
213 AOT_theorem "useful-tautologies:7": <math>(\neg\varphi \rightarrow \psi) \rightarrow (\neg\psi \rightarrow \varphi)</math>
214   by (metis "→I" MP "useful-tautologies:3" "useful-tautologies:5")
215
216 AOT_theorem "useful-tautologies:8": <math>\varphi \rightarrow (\neg\psi \rightarrow \neg(\varphi \rightarrow \psi))</math>
217   by (metis "→I" MP "useful-tautologies:5")
218
219 AOT_theorem "useful-tautologies:9": <math>(\varphi \rightarrow \psi) \rightarrow ((\neg\varphi \rightarrow \psi) \rightarrow \psi)</math>
220   by (metis "→I" MP "useful-tautologies:6")
221
222 AOT_theorem "useful-tautologies:10": <math>(\varphi \rightarrow \neg\psi) \rightarrow ((\varphi \rightarrow \psi) \rightarrow \neg\varphi)</math>
223   by (metis "→I" MP "pl:3"[axiom_inst])
224
225 AOT_theorem "dn-i-e:1":
226   assumes <math>\varphi</math>
227   shows <math>\neg\neg\varphi</math>
228   using MP "useful-tautologies:2" assms by blast
229 lemmas "¬¬I" = "dn-i-e:1"
230 AOT_theorem "dn-i-e:2":
231   assumes <math>\neg\neg\varphi</math>
232   shows <math>\varphi</math>
233   using MP "useful-tautologies:1" assms by blast
234 lemmas "¬¬E" = "dn-i-e:2"
235
236 AOT_theorem "modus-tollens:1":
237   assumes <math>\varphi \rightarrow \psi</math> and <math>\neg\psi</math>
238   shows <math>\neg\varphi</math>
239   using MP "useful-tautologies:5" assms by blast
240 AOT_theorem "modus-tollens:2":
241   assumes <math>\varphi \rightarrow \neg\psi</math> and <math>\psi</math>
242   shows <math>\neg\varphi</math>
243   using "¬¬I" "modus-tollens:1" assms by blast
244 lemmas MT = "modus-tollens:1" "modus-tollens:2"
245
246 AOT_theorem "contraposition:1[1]":
247   assumes <math>\varphi \rightarrow \psi</math>
248   shows <math>\neg\psi \rightarrow \neg\varphi</math>
249   using "→I" MT(1) assms by blast
250 AOT_theorem "contraposition:1[2]":

```

```

251   assumes <¬ψ → ¬φ>
252   shows <φ → ψ>
253   using "→I" "¬¬E" MT(2) assms by blast
254
255 AOT_theorem "contraposition:2":
256   assumes <φ → ¬ψ>
257   shows <ψ → ¬φ>
258   using "→I" MT(2) assms by blast
259
260 AOT_theorem "reductio-aa:1":
261   assumes <¬φ ⊢ ¬ψ> and <¬φ ⊢ ψ>
262   shows <φ>
263   using "→I" "¬¬E" MT(2) assms by blast
264 AOT_theorem "reductio-aa:2":
265   assumes <φ ⊢ ¬ψ> and <φ ⊢ ψ>
266   shows <¬φ>
267   using "reductio-aa:1" assms by blast
268 lemmas "RAA" = "reductio-aa:1" "reductio-aa:2"
269
270 AOT_theorem "exc-mid": <φ ∨ ¬φ>
271   using "df-rules-formulas[4]" "if-p-then-p" MP
272   "conventions:2" by blast
273
274 AOT_theorem "non-contradiction": <¬(φ & ¬φ)>
275   using "df-rules-formulas[3]" MT(2) "useful-tautologies:2"
276   "conventions:1" by blast
277
278 AOT_theorem "con-dis-taut:1": <(φ & ψ) → φ>
279   by (meson "→I" "df-rules-formulas[3]" MP RAA(1) "conventions:1")
280 AOT_theorem "con-dis-taut:2": <(φ & ψ) → ψ>
281   by (metis "→I" "df-rules-formulas[3]" MT(2) RAA(2)
282   "¬¬E" "conventions:1")
283 lemmas "Conjunction Simplification" = "con-dis-taut:1" "con-dis-taut:2"
284
285 AOT_theorem "con-dis-taut:3": <φ → (φ ∨ ψ)>
286   by (meson "contraposition:1[2]" "df-rules-formulas[4]"
287   MP "→I" "conventions:2")
288 AOT_theorem "con-dis-taut:4": <ψ → (φ ∨ ψ)>
289   using "Hypothetical Syllogism" "df-rules-formulas[4]"
290   "pl:1"[axiom_inst] "conventions:2" by blast
291 lemmas "Disjunction Addition" = "con-dis-taut:3" "con-dis-taut:4"
292
293 AOT_theorem "con-dis-taut:5": <φ → (ψ → (φ & ψ))>
294   by (metis "contraposition:2" "Hypothetical Syllogism" "→I"
295   "df-rules-formulas[4]" "conventions:1")
296 lemmas Adjunction = "con-dis-taut:5"
297
298 AOT_theorem "con-dis-taut:6": <(φ & φ) ≡ φ>
299   by (metis Adjunction "→I" "df-rules-formulas[4]" MP
300   "Conjunction Simplification"(1) "conventions:3")
301 lemmas "Idempotence of &" = "con-dis-taut:6"
302
303 AOT_theorem "con-dis-taut:7": <(φ ∨ φ) ≡ φ>
304 proof -
305   {
306     AOT_assume <φ ∨ φ>
307     AOT_hence <¬φ → φ>
308     using "conventions:2"[THEN "df-rules-formulas[3]"] MP by blast
309     AOT_hence <φ> using "if-p-then-p" RAA(1) MP by blast
310   }
311 moreover {
312     AOT_assume <φ>
313     AOT_hence <φ ∨ φ> using "Disjunction Addition"(1) MP by blast

```

```

314 }
315 ultimately AOT_show <( $\varphi \vee \varphi \equiv \varphi$ )>
316   using "conventions:3"[THEN "df-rules-formulas[4]"] MP
317   by (metis Adjunction " $\rightarrow$ I")
318 qed
319 lemmas "Idempotence of  $\vee$ " = "con-dis-taut:7"
320
321
322 AOT_theorem "con-dis-i-e:1": (86.1)
323   assumes < $\varphi$ > and < $\psi$ >
324   shows < $\varphi \ \& \ \psi$ >
325   using Adjunction MP assms by blast
326 lemmas "&I" = "con-dis-i-e:1"
327
328 AOT_theorem "con-dis-i-e:2:a": (86.2.a)
329   assumes < $\varphi \ \& \ \psi$ >
330   shows < $\varphi$ >
331   using "Conjunction Simplification"(1) MP assms by blast
332 AOT_theorem "con-dis-i-e:2:b": (86.2.b)
333   assumes < $\varphi \ \& \ \psi$ >
334   shows < $\psi$ >
335   using "Conjunction Simplification"(2) MP assms by blast
336 lemmas "&E" = "con-dis-i-e:2:a" "con-dis-i-e:2:b"
337
338 AOT_theorem "con-dis-i-e:3:a": (86.3.a)
339   assumes < $\varphi$ >
340   shows < $\varphi \vee \psi$ >
341   using "Disjunction Addition"(1) MP assms by blast
342 AOT_theorem "con-dis-i-e:3:b": (86.3.b)
343   assumes < $\psi$ >
344   shows < $\varphi \vee \psi$ >
345   using "Disjunction Addition"(2) MP assms by blast
346 AOT_theorem "con-dis-i-e:3:c": (86.3.c)
347   assumes < $\varphi \vee \psi$ > and < $\varphi \rightarrow \chi$ > and < $\psi \rightarrow \Theta$ >
348   shows < $\chi \vee \Theta$ >
349   by (metis "con-dis-i-e:3:a" "Disjunction Addition"(2)
350           "df-rules-formulas[3]" MT(1) RAA(1)
351           "conventions:2" assms)
352 lemmas " $\vee$ I" = "con-dis-i-e:3:a" "con-dis-i-e:3:b" "con-dis-i-e:3:c"
353
354 AOT_theorem "con-dis-i-e:4:a": (86.4.a)
355   assumes < $\varphi \vee \psi$ > and < $\varphi \rightarrow \chi$ > and < $\psi \rightarrow \chi$ >
356   shows < $\chi$ >
357   by (metis MP RAA(2) "df-rules-formulas[3]" "conventions:2" assms)
358 AOT_theorem "con-dis-i-e:4:b": (86.4.b)
359   assumes < $\varphi \vee \psi$ > and < $\neg\varphi$ >
360   shows < $\psi$ >
361   using "con-dis-i-e:4:a" RAA(1) " $\rightarrow$ I" assms by blast
362 AOT_theorem "con-dis-i-e:4:c": (86.4.c)
363   assumes < $\varphi \vee \psi$ > and < $\neg\psi$ >
364   shows < $\varphi$ >
365   using "con-dis-i-e:4:a" RAA(1) " $\rightarrow$ I" assms by blast
366 lemmas " $\vee$ E" = "con-dis-i-e:4:a" "con-dis-i-e:4:b" "con-dis-i-e:4:c"
367
368 AOT_theorem "raa-cor:1": (87.1)
369   assumes < $\neg\varphi \vdash \psi \ \& \ \neg\psi$ >
370   shows < $\varphi$ >
371   using "&E" " $\vee$ E"(3) " $\vee$ I"(2) RAA(2) assms by blast
372 AOT_theorem "raa-cor:2": (87.2)
373   assumes < $\varphi \vdash \psi \ \& \ \neg\psi$ >
374   shows < $\neg\varphi$ >
375   using "raa-cor:1" assms by blast
376 AOT_theorem "raa-cor:3": (87.3)

```



```

377   assumes <φ> and <¬ψ ⊢ ¬φ>
378   shows <ψ>
379   using RAA assms by blast
380 AOT_theorem "raa-cor:4": (87.4)
381   assumes <¬φ> and <¬ψ ⊢ φ>
382   shows <ψ>
383   using RAA assms by blast
384 AOT_theorem "raa-cor:5": (87.5)
385   assumes <φ> and <ψ ⊢ ¬φ>
386   shows <¬ψ>
387   using RAA assms by blast
388 AOT_theorem "raa-cor:6": (87.6)
389   assumes <¬φ> and <ψ ⊢ φ>
390   shows <¬ψ>
391   using RAA assms by blast
392
393 AOT_theorem "oth-class-taut:1:a": <(φ → ψ) ≡ ¬(φ & ¬ψ)> (88.1.a)
394   by (rule "conventions:3"[THEN "df-rules-formulas[4]", THEN "→E"])
395     (metis "&E" "&I" "raa-cor:3" "→I" MP)
396 AOT_theorem "oth-class-taut:1:b": <¬(φ → ψ) ≡ (φ & ¬ψ)> (88.1.b)
397   by (rule "conventions:3"[THEN "df-rules-formulas[4]", THEN "→E"])
398     (metis "&E" "&I" "raa-cor:3" "→I" MP)
399 AOT_theorem "oth-class-taut:1:c": <(φ → ψ) ≡ (¬φ ∨ ψ)> (88.1.c)
400   by (rule "conventions:3"[THEN "df-rules-formulas[4]", THEN "→E"])
401     (metis "&I" "∨I"(1, 2) "∨E"(3) "→I" MP "raa-cor:1")
402
403 AOT_theorem "oth-class-taut:2:a": <(φ & ψ) ≡ (ψ & φ)> (88.2.a)
404   by (rule "conventions:3"[THEN "df-rules-formulas[4]", THEN "→E"])
405     (meson "&I" "&E" "→I")
406 lemmas "Commutativity of &" = "oth-class-taut:2:a"
407 AOT_theorem "oth-class-taut:2:b": <(φ & (ψ & χ)) ≡ ((φ & ψ) & χ)> (88.2.b)
408   by (rule "conventions:3"[THEN "df-rules-formulas[4]", THEN "→E"])
409     (metis "&I" "&E" "→I")
410 lemmas "Associativity of &" = "oth-class-taut:2:b"
411 AOT_theorem "oth-class-taut:2:c": <(φ ∨ ψ) ≡ (ψ ∨ φ)> (88.2.c)
412   by (rule "conventions:3"[THEN "df-rules-formulas[4]", THEN "→E"])
413     (metis "&I" "∨I"(1, 2) "∨E"(1) "→I")
414 lemmas "Commutativity of ∨" = "oth-class-taut:2:c"
415 AOT_theorem "oth-class-taut:2:d": <(φ ∨ (ψ ∨ χ)) ≡ ((φ ∨ ψ) ∨ χ)> (88.2.d)
416   by (rule "conventions:3"[THEN "df-rules-formulas[4]", THEN "→E"])
417     (metis "&I" "∨I"(1, 2) "∨E"(1) "→I")
418 lemmas "Associativity of ∨" = "oth-class-taut:2:d"
419 AOT_theorem "oth-class-taut:2:e": <(φ ≡ ψ) ≡ (ψ ≡ φ)> (88.2.e)
420   by (rule "conventions:3"[THEN "df-rules-formulas[4]", THEN "→E"]; rule "&I";
421     metis "&I" "df-rules-formulas[4]" "conventions:3" "&E"
422     "Hypothetical Syllogism" "→I" "df-rules-formulas[3]")
423 lemmas "Commutativity of ≡" = "oth-class-taut:2:e"
424 AOT_theorem "oth-class-taut:2:f": <(φ ≡ (ψ ≡ χ)) ≡ ((φ ≡ ψ) ≡ χ)> (88.2.f)
425   using "conventions:3"[THEN "df-rules-formulas[4]"]
426     "conventions:3"[THEN "df-rules-formulas[3]"]
427     "→I" "→E" "&E" "&I"
428   by metis
429 lemmas "Associativity of ≡" = "oth-class-taut:2:f"
430
431 AOT_theorem "oth-class-taut:3:a": <φ ≡ φ> (88.3.a)
432   using "&I" "vdash-properties:6" "if-p-then-p"
433     "df-rules-formulas[4]" "conventions:3" by blast
434 AOT_theorem "oth-class-taut:3:b": <φ ≡ ¬¬φ> (88.3.b)
435   using "&I" "useful-tautologies:1" "useful-tautologies:2" "→E"
436     "df-rules-formulas[4]" "conventions:3" by blast
437 AOT_theorem "oth-class-taut:3:c": <¬(φ ≡ ¬φ)> (88.3.c)
438   by (metis "&E" "→E" RAA "df-rules-formulas[3]" "conventions:3")
439

```

```

440 AOT_theorem "oth-class-taut:4:a": <( $\varphi \rightarrow \psi$ )  $\rightarrow$  (( $\psi \rightarrow \chi$ )  $\rightarrow$  ( $\varphi \rightarrow \chi$ ))> (88.4.a)
441   by (metis " $\rightarrow$ E" " $\rightarrow$ I")
442 AOT_theorem "oth-class-taut:4:b": <( $\varphi \equiv \psi$ )  $\equiv$  ( $\neg\varphi \equiv \neg\psi$ )> (88.4.b)
443   using "conventions:3"[THEN "df-rules-formulas[4]"]
444     "conventions:3"[THEN "df-rules-formulas[3]"]
445     " $\rightarrow$ I" " $\rightarrow$ E" "&E" "&I" RAA by metis
446 AOT_theorem "oth-class-taut:4:c": <( $\varphi \equiv \psi$ )  $\rightarrow$  (( $\varphi \rightarrow \chi$ )  $\equiv$  ( $\psi \rightarrow \chi$ ))> (88.4.c)
447   using "conventions:3"[THEN "df-rules-formulas[4]"]
448     "conventions:3"[THEN "df-rules-formulas[3]"]
449     " $\rightarrow$ I" " $\rightarrow$ E" "&E" "&I" by metis
450 AOT_theorem "oth-class-taut:4:d": <( $\varphi \equiv \psi$ )  $\rightarrow$  (( $\chi \rightarrow \varphi$ )  $\equiv$  ( $\chi \rightarrow \psi$ ))> (88.4.d)
451   using "conventions:3"[THEN "df-rules-formulas[4]"]
452     "conventions:3"[THEN "df-rules-formulas[3]"]
453     " $\rightarrow$ I" " $\rightarrow$ E" "&E" "&I" by metis
454 AOT_theorem "oth-class-taut:4:e": <( $\varphi \equiv \psi$ )  $\rightarrow$  (( $\varphi \& \chi$ )  $\equiv$  ( $\psi \& \chi$ ))> (88.4.e)
455   using "conventions:3"[THEN "df-rules-formulas[4]"]
456     "conventions:3"[THEN "df-rules-formulas[3]"]
457     " $\rightarrow$ I" " $\rightarrow$ E" "&E" "&I" by metis
458 AOT_theorem "oth-class-taut:4:f": <( $\varphi \equiv \psi$ )  $\rightarrow$  (( $\chi \& \varphi$ )  $\equiv$  ( $\chi \& \psi$ ))> (88.4.f)
459   using "conventions:3"[THEN "df-rules-formulas[4]"]
460     "conventions:3"[THEN "df-rules-formulas[3]"]
461     " $\rightarrow$ I" " $\rightarrow$ E" "&E" "&I" by metis
462 AOT_theorem "oth-class-taut:4:g": <( $\varphi \equiv \psi$ )  $\equiv$  (( $\varphi \& \psi$ )  $\vee$  ( $\neg\varphi \& \neg\psi$ ))> (88.4.g)
463 proof(safe intro!: "conventions:3"[THEN "df-rules-formulas[4]"], THEN " $\rightarrow$ E")
464   "&I" " $\rightarrow$ I"
465   dest!: "conventions:3"[THEN "df-rules-formulas[3]"], THEN " $\rightarrow$ E")
466   AOT_show < $\varphi \& \psi \vee (\neg\varphi \& \neg\psi)$ > if < $\varphi \rightarrow \psi$ > & < $\psi \rightarrow \varphi$ >
467     using "&E" "\I" " $\rightarrow$ E" "&I" "raa-cor:1" " $\rightarrow$ I" "\VE" that by metis
468 next
469   AOT_show < $\psi$ > if < $\varphi \& \psi \vee (\neg\varphi \& \neg\psi)$ > and < $\varphi$ >
470     using that "\VE" "&E" "raa-cor:3" by blast
471 next
472   AOT_show < $\varphi$ > if < $\varphi \& \psi \vee (\neg\varphi \& \neg\psi)$ > and < $\psi$ >
473     using that "\VE" "&E" "raa-cor:3" by blast
474 qed
475 AOT_theorem "oth-class-taut:4:h": < $\neg(\varphi \equiv \psi) \equiv ((\varphi \& \neg\psi) \vee (\neg\varphi \& \psi))$ > (88.4.h)
476 proof (safe intro!: "conventions:3"[THEN "df-rules-formulas[4]"], THEN " $\rightarrow$ E")
477   "&I" " $\rightarrow$ I")
478   AOT_show < $\varphi \& \neg\psi \vee (\neg\varphi \& \psi)$ > if < $\neg(\varphi \equiv \psi)$ >
479     by (metis that "&I" "\I"(1, 2) " $\rightarrow$ I" MT(1) "df-rules-formulas[4]"
480         "raa-cor:3" "conventions:3")
481 next
482   AOT_show < $\neg(\varphi \equiv \psi)$ > if < $\varphi \& \neg\psi \vee (\neg\varphi \& \psi)$ >
483     by (metis that "&E" "\VE"(2) " $\rightarrow$ E" "df-rules-formulas[3]"
484         "raa-cor:3" "conventions:3")
485 qed
486 AOT_theorem "oth-class-taut:5:a": <( $\varphi \& \psi$ )  $\equiv$   $\neg(\neg\varphi \vee \neg\psi)$ > (88.5.a)
487   using "conventions:3"[THEN "df-rules-formulas[4]"]
488     " $\rightarrow$ I" " $\rightarrow$ E" "&E" "&I" "\I" "\VE" RAA by metis
489 AOT_theorem "oth-class-taut:5:b": <( $\varphi \vee \psi$ )  $\equiv$   $\neg(\neg\varphi \& \neg\psi)$ > (88.5.b)
490   using "conventions:3"[THEN "df-rules-formulas[4]"]
491     " $\rightarrow$ I" " $\rightarrow$ E" "&E" "&I" "\I" "\VE" RAA by metis
492 AOT_theorem "oth-class-taut:5:c": < $\neg(\varphi \& \psi) \equiv (\neg\varphi \vee \neg\psi)$ > (88.5.c)
493   using "conventions:3"[THEN "df-rules-formulas[4]"]
494     " $\rightarrow$ I" " $\rightarrow$ E" "&E" "&I" "\I" "\VE" RAA by metis
495 AOT_theorem "oth-class-taut:5:d": < $\neg(\varphi \vee \psi) \equiv (\neg\varphi \& \neg\psi)$ > (88.5.d)
496   using "conventions:3"[THEN "df-rules-formulas[4]"]
497     " $\rightarrow$ I" " $\rightarrow$ E" "&E" "&I" "\I" "\VE" RAA by metis
498
499 lemmas DeMorgan = "oth-class-taut:5:c" "oth-class-taut:5:d"
500
501 AOT_theorem "oth-class-taut:6:a": (88.6.a)
502   <( $\varphi \& (\psi \vee \chi)$ )  $\equiv$  (( $\varphi \& \psi$ )  $\vee$  ( $\varphi \& \chi$ ))>

```

```

503   using "conventions:3"[THEN "df-rules-formulas[4]"
504     "→I" "→E" "&E" "&I" "∨I" "∨E" RAA by metis
505 AOT_theorem "oth-class-taut:6:b": (88.6.b)
506   <( $\varphi \vee (\psi \ \& \ \chi)$ )  $\equiv$  ( $(\varphi \vee \psi) \ \& \ (\varphi \vee \chi)$ )>
507   using "conventions:3"[THEN "df-rules-formulas[4]"
508     "→I" "→E" "&E" "&I" "∨I" "∨E" RAA by metis
509
510 AOT_theorem "oth-class-taut:7:a": <( $(\varphi \ \& \ \psi) \rightarrow \chi$ )  $\rightarrow$  ( $\varphi \rightarrow (\psi \rightarrow \chi)$ )> (88.7.a)
511   by (metis "&I" "→E" "→I")
512 lemmas Exportation = "oth-class-taut:7:a"
513 AOT_theorem "oth-class-taut:7:b": <( $\varphi \rightarrow (\psi \rightarrow \chi)$ )  $\rightarrow$  ( $(\varphi \ \& \ \psi) \rightarrow \chi$ )> (88.7.b)
514   by (metis "&E" "→E" "→I")
515 lemmas Importation = "oth-class-taut:7:b"
516
517 AOT_theorem "oth-class-taut:8:a": (88.8.a)
518   <( $\varphi \rightarrow (\psi \rightarrow \chi)$ )  $\equiv$  ( $\psi \rightarrow (\varphi \rightarrow \chi)$ )>
519   using "conventions:3"[THEN "df-rules-formulas[4]" "→I" "→E" "&E" "&I"
520     by metis
521 lemmas Permutation = "oth-class-taut:8:a"
522 AOT_theorem "oth-class-taut:8:b": (88.8.b)
523   <( $\varphi \rightarrow \psi$ )  $\rightarrow$  ( $(\varphi \rightarrow \chi) \rightarrow (\psi \ \& \ \chi)$ )>
524   by (metis "&I" "→E" "→I")
525 lemmas Composition = "oth-class-taut:8:b"
526 AOT_theorem "oth-class-taut:8:c": (88.8.c)
527   <( $\varphi \rightarrow \chi$ )  $\rightarrow$  ( $(\psi \rightarrow \chi) \rightarrow ((\varphi \vee \psi) \rightarrow \chi)$ )>
528   by (metis "∨E"(2) "→E" "→I" RAA(1))
529 AOT_theorem "oth-class-taut:8:d": (88.8.d)
530   <( $(\varphi \rightarrow \psi) \ \& \ (\chi \rightarrow \Theta)$ )  $\rightarrow$  ( $(\varphi \ \& \ \chi) \rightarrow (\psi \ \& \ \Theta)$ )>
531   by (metis "&E" "&I" "→E" "→I")
532 lemmas "Double Composition" = "oth-class-taut:8:d"
533 AOT_theorem "oth-class-taut:8:e": (88.8.e)
534   <( $(\varphi \ \& \ \psi) \equiv (\varphi \ \& \ \chi) \equiv (\varphi \rightarrow (\psi \equiv \chi))$ )>
535   by (metis "conventions:3"[THEN "df-rules-formulas[4]"
536     "conventions:3"[THEN "df-rules-formulas[3]"
537     "→I" "→E" "&E" "&I")
538 AOT_theorem "oth-class-taut:8:f": (88.8.f)
539   <( $(\varphi \ \& \ \psi) \equiv (\chi \ \& \ \psi) \equiv (\psi \rightarrow (\varphi \equiv \chi))$ )>
540   by (metis "conventions:3"[THEN "df-rules-formulas[4]"
541     "conventions:3"[THEN "df-rules-formulas[3]"
542     "→I" "→E" "&E" "&I")
543 AOT_theorem "oth-class-taut:8:g": (88.8.g)
544   <( $\psi \equiv \chi$ )  $\rightarrow$  ( $(\varphi \vee \psi) \equiv (\varphi \vee \chi)$ )>
545   by (metis "conventions:3"[THEN "df-rules-formulas[4]"
546     "conventions:3"[THEN "df-rules-formulas[3]"
547     "→I" "→E" "&E" "&I" "∨I" "∨E"(1))
548 AOT_theorem "oth-class-taut:8:h": (88.8.h)
549   <( $\psi \equiv \chi$ )  $\rightarrow$  ( $(\psi \vee \varphi) \equiv (\chi \vee \varphi)$ )>
550   by (metis "conventions:3"[THEN "df-rules-formulas[4]"
551     "conventions:3"[THEN "df-rules-formulas[3]"
552     "→I" "→E" "&E" "&I" "∨I" "∨E"(1))
553 AOT_theorem "oth-class-taut:8:i": (88.8.i)
554   <( $\varphi \equiv (\psi \ \& \ \chi)$ )  $\rightarrow$  ( $\psi \rightarrow (\varphi \equiv \chi)$ )>
555   by (metis "conventions:3"[THEN "df-rules-formulas[4]"
556     "conventions:3"[THEN "df-rules-formulas[3]"
557     "→I" "→E" "&E" "&I")
558
559 AOT_theorem "intro-elim:1": (89.1)
560   assumes < $\varphi \vee \psi$ > and < $\varphi \equiv \chi$ > and < $\psi \equiv \Theta$ >
561   shows < $\chi \vee \Theta$ >
562   by (metis assms "∨I"(1, 2) "∨E"(1) "→I" "→E" "&E"(1)
563     "conventions:3"[THEN "df-rules-formulas[3]"
564
565 AOT_theorem "intro-elim:2": (89.2)

```

```

566   assumes <math>\varphi \rightarrow \psi</math> and <math>\psi \rightarrow \varphi</math>
567   shows <math>\varphi \equiv \psi</math>
568   by (meson "&I" "conventions:3" "df-rules-formulas[4]" MP assms)
569 lemmas "≡I" = "intro-elim:2"
570
571 AOT_theorem "intro-elim:3:a": (89.3.a)
572   assumes <math>\varphi \equiv \psi</math> and <math>\varphi</math>
573   shows <math>\psi</math>
574   by (metis "&V I"(1) "→I" "&V E"(1) "intro-elim:1" assms)
575 AOT_theorem "intro-elim:3:b": (89.3.b)
576   assumes <math>\varphi \equiv \psi</math> and <math>\psi</math>
577   shows <math>\varphi</math>
578   using "intro-elim:3:a" "Commutativity of ≡" assms by blast
579 AOT_theorem "intro-elim:3:c": (89.3.c)
580   assumes <math>\varphi \equiv \psi</math> and <math>\neg\varphi</math>
581   shows <math>\neg\psi</math>
582   using "intro-elim:3:b" "raa-cor:3" assms by blast
583 AOT_theorem "intro-elim:3:d": (89.3.d)
584   assumes <math>\varphi \equiv \psi</math> and <math>\neg\psi</math>
585   shows <math>\neg\varphi</math>
586   using "intro-elim:3:a" "raa-cor:3" assms by blast
587 AOT_theorem "intro-elim:3:e": (89.3.e)
588   assumes <math>\varphi \equiv \psi</math> and <math>\psi \equiv \chi</math>
589   shows <math>\varphi \equiv \chi</math>
590   by (metis "≡I" "→I" "intro-elim:3:a" "intro-elim:3:b" assms)
591 declare "intro-elim:3:e"[trans]
592 AOT_theorem "intro-elim:3:f": (89.3.f)
593   assumes <math>\varphi \equiv \psi</math> and <math>\varphi \equiv \chi</math>
594   shows <math>\chi \equiv \psi</math>
595   by (metis "≡I" "→I" "intro-elim:3:a" "intro-elim:3:b" assms)
596 lemmas "≡E" = "intro-elim:3:a" "intro-elim:3:b" "intro-elim:3:c"
597           "intro-elim:3:d" "intro-elim:3:e" "intro-elim:3:f"
598
599 declare "Commutativity of ≡"[THEN "≡E"(1), sym]
600
601 AOT_theorem "rule-eq-df:1": (90.1)
602   assumes <math>\varphi \equiv_{df} \psi</math>
603   shows <math>\varphi \equiv \psi</math>
604   by (simp add: "≡I" "df-rules-formulas[3]" "df-rules-formulas[4]" assms)
605 lemmas "≡Df" = "rule-eq-df:1"
606 AOT_theorem "rule-eq-df:2": (90.2)
607   assumes <math>\varphi \equiv_{df} \psi</math> and <math>\varphi</math>
608   shows <math>\psi</math>
609   using "≡Df" "≡E"(1) assms by blast
610 lemmas "≡_{df}E" = "rule-eq-df:2"
611 AOT_theorem "rule-eq-df:3": (90.3)
612   assumes <math>\varphi \equiv_{df} \psi</math> and <math>\psi</math>
613   shows <math>\varphi</math>
614   using "≡Df" "≡E"(2) assms by blast
615 lemmas "≡_{df}I" = "rule-eq-df:3"
616
617 AOT_theorem "df-simplify:1": (91.1)
618   assumes <math>\varphi \equiv (\psi \ \& \ \chi)</math> and <math>\psi</math>
619   shows <math>\varphi \equiv \chi</math>
620   by (metis "&E"(2) "&I" "≡E"(1, 2) "≡I" "→I" assms)
621 (* Note: this is a slight variation from PLM *)
622 AOT_theorem "df-simplify:2": (91.2)
623   assumes <math>\varphi \equiv (\psi \ \& \ \chi)</math> and <math>\chi</math>
624   shows <math>\varphi \equiv \psi</math>
625   by (metis "&E"(1) "&I" "≡E"(1, 2) "≡I" "→I" assms)
626 lemmas "≡S" = "df-simplify:1" "df-simplify:2"
627
628 subsection<The Theory of Quantification>

```

```

629 text<\label{PLM: 9.6}>
630
631 AOT_theorem "rule-ui:1": (93.1)
632   assumes < $\forall\alpha \varphi\{\alpha\}$ > and < $\tau\downarrow$ >
633   shows < $\varphi\{\tau\}$ >
634   using " $\rightarrow E$ " "cqt:1"[axiom_inst] assms by blast
635 AOT_theorem "rule-ui:2[const_var]": (93.2)
636   assumes < $\forall\alpha \varphi\{\alpha\}$ >
637   shows < $\varphi\{\beta\}$ >
638   by (simp add: "rule-ui:1" "cqt:2[const_var]"[axiom_inst] assms)
639 AOT_theorem "rule-ui:2[lambda]": (93.2)
640   assumes < $\forall F \varphi\{F\}$ > and <INSTANCE_OF_CQT_2( $\psi$ )>
641   shows < $\varphi\{[\lambda\nu_1\dots\nu_n \psi\{\nu_1\dots\nu_n\}]\}$ >
642   by (simp add: "rule-ui:1" "cqt:2[lambda]"[axiom_inst] assms)
643 AOT_theorem "rule-ui:3": (93.3)
644   assumes < $\forall\alpha \varphi\{\alpha\}$ >
645   shows < $\varphi\{\alpha\}$ >
646   by (simp add: "rule-ui:2[const_var]" assms)
647 lemmas "VE" = "rule-ui:1" "rule-ui:2[const_var]"
648           "rule-ui:2[lambda]" "rule-ui:3"
649
650 AOT_theorem "cqt-orig:1[const_var]": < $\forall\alpha \varphi\{\alpha\} \rightarrow \varphi\{\beta\}$ > (95.1)
651   by (simp add: "VE"(2) " $\rightarrow I$ ")
652 AOT_theorem "cqt-orig:1[lambda]": (95.1)
653   assumes <INSTANCE_OF_CQT_2( $\psi$ )>
654   shows < $\forall F \varphi\{F\} \rightarrow \varphi\{[\lambda\nu_1\dots\nu_n \psi\{\nu_1\dots\nu_n\}]\}$ >
655   by (simp add: "VE"(3) " $\rightarrow I$ " assms)
656 AOT_theorem "cqt-orig:2": < $\forall\alpha (\varphi \rightarrow \psi\{\alpha\}) \rightarrow (\varphi \rightarrow \forall\alpha \psi\{\alpha\})$ > (95.2)
657   by (metis " $\rightarrow I$ " GEN "vdash-properties:6" "VE"(4))
658 AOT_theorem "cqt-orig:3": < $\forall\alpha \varphi\{\alpha\} \rightarrow \varphi\{\alpha\}$ > (95.3)
659   using "cqt-orig:1[const_var]".
660
661 AOT_theorem universal: (96)
662   assumes <for arbitrary  $\beta$ :  $\varphi\{\beta\}$ >
663   shows < $\forall\alpha \varphi\{\alpha\}$ >
664   using GEN assms .
665 lemmas "VI" = universal
666
667 (* Generalized mechanism for  $\forall I$  followed by  $\forall E$  *)
668 ML<
669 fun get_instantiated_allI ctxt varname thm = let
670   val trm = Thm.concl_of thm
671   val trm =
672     case trm of (@{const Trueprop} $ (@{const AOT_model_valid_in} $ _ $ x)) => x
673     | _ => raise Term.TERM ("Expected simple theorem.", [trm])
674   fun extractVars (Const (const_name<AOT_term_of_var>, _) $ Var v) =
675     (if fst (fst v) = fst varname then [Var v] else [])
676     | extractVars (t1 $ t2) = extractVars t1 @ extractVars t2
677     | extractVars (Abs (_, _, t)) = extractVars t
678     | extractVars _ = []
679   val vars = extractVars trm
680   val vars = fold Term.add_vars vars []
681   val var = hd vars
682   val trmty =
683     case (snd var) of (Type (type_name<AOT_var>, [t])) => (t)
684     | _ => raise Term.TYPER ("Expected variable type.", [snd var], [Var var])
685   val trm = Abs (Term.string_of_vname (fst var), trmty, Term.abstract_over (
686     Const (const_name<AOT_term_of_var>, Type ("fun", [snd var, trmty]))
687     $ Var var, trm))
688   val trm = Thm.cterm_of (Context.proof_of ctxt) trm
689   val ty = hd (Term.add_tvars (Thm.prop_of @{thm "VI"}) [])
690   val typ = Thm.ctyp_of (Context.proof_of ctxt) trmty
691   val allthm = Drule.instantiate_normalize (TVars.make [(ty, typ)], Vars.empty) @{thm "VI"}

```

```

692 val phi = hd (Term.add_vars (Thm.prop_of allthm) [])
693 val allthm = Drule.instantiate_normalize (TVars.empty, Vars.make [(phi, trm)]) allthm
694 in
695 allthm
696 end
697 >
698
699 attribute_setup "∀I" =
700   <Scan.lift (Scan.repeat1 Args.var) >> (fn args => Thm.rule_attribute []
701     (fn ctxt => fn thm => fold (fn arg => fn thm =>
702       thm RS get_instantiated_allI ctxt arg thm) args thm))>
703   "Quantify over a variable in a theorem using GEN."
704
705 attribute_setup "unvarify" =
706   <Scan.lift (Scan.repeat1 Args.var) >> (fn args => Thm.rule_attribute []
707     (fn ctxt => fn thm =>
708       let
709         fun get_inst_allI arg thm = thm RS get_instantiated_allI ctxt arg thm
710         val thm = fold get_inst_allI args thm
711         val thm = fold (K (fn thm => thm RS @{thm "VE"(1)})) args thm
712       in
713         thm
714       end))>
715   "Generalize a statement about variables to a statement about denoting terms."
716
717 (* Note: rereplace-lem does not apply to the embedding *)
718
719 AOT_theorem "cqt-basic:1": <∀α∀β φ{α,β} ≡ ∀β∀α φ{α,β}> (99.1)
720   by (metis "≡I" "VE"(2) "VI" "→I")
721
722 AOT_theorem "cqt-basic:2": (99.2)
723   <∀α(φ{α} ≡ ψ{α}) ≡ (∀α(φ{α} → ψ{α}) & ∀α(ψ{α} → φ{α}))>
724   proof (rule "≡I"; rule "→I")
725     AOT_assume <∀α(φ{α} ≡ ψ{α})>
726     AOT_hence <φ{α} ≡ ψ{α}> for α using "VE"(2) by blast
727     AOT_hence <φ{α} → ψ{α}> and <ψ{α} → φ{α}> for α
728       using "≡E"(1,2) "→I" by blast+
729     AOT_thus <∀α(φ{α} → ψ{α}) & ∀α(ψ{α} → φ{α})>
730       by (auto intro: "&I" "VI")
731   next
732     AOT_assume <∀α(φ{α} → ψ{α}) & ∀α(ψ{α} → φ{α})>
733     AOT_hence <φ{α} → ψ{α}> and <ψ{α} → φ{α}> for α
734       using "VE"(2) "&E" by blast+
735     AOT_hence <φ{α} ≡ ψ{α}> for α
736       using "≡I" by blast
737     AOT_thus <∀α(φ{α} ≡ ψ{α})> by (auto intro: "VI")
738   qed
739
740 AOT_theorem "cqt-basic:3": <∀α(φ{α} ≡ ψ{α}) → (∀α φ{α} ≡ ∀α ψ{α})> (99.3)
741   proof(rule "→I")
742     AOT_assume <∀α(φ{α} ≡ ψ{α})>
743     AOT_hence 1: <φ{α} ≡ ψ{α}> for α using "VE"(2) by blast
744     {
745       AOT_assume <∀α φ{α}>
746       AOT_hence <∀α ψ{α}> using 1 "VI" "VE"(4) "≡E" by metis
747     }
748     moreover {
749       AOT_assume <∀α ψ{α}>
750       AOT_hence <∀α φ{α}> using 1 "VI" "VE"(4) "≡E" by metis
751     }
752     ultimately AOT_show <∀α φ{α} ≡ ∀α ψ{α}>
753     using "≡I" "→I" by auto
754   qed

```

```

755
756 AOT_theorem "cqt-basic:4": < $\forall\alpha(\varphi\{\alpha\} \ \& \ \psi\{\alpha\}) \rightarrow (\forall\alpha \ \varphi\{\alpha\} \ \& \ \forall\alpha \ \psi\{\alpha\})$ > (99.4)
757 proof(rule "→I")
758   AOT_assume 0: < $\forall\alpha(\varphi\{\alpha\} \ \& \ \psi\{\alpha\})$ >
759   AOT_have < $\varphi\{\alpha\}$ > and < $\psi\{\alpha\}$ > for  $\alpha$  using "VE"(2) 0 "&E" by blast+
760   AOT_thus < $\forall\alpha \ \varphi\{\alpha\} \ \& \ \forall\alpha \ \psi\{\alpha\}$ >
761   by (auto intro: "VI" "&I")
762 qed
763
764 AOT_theorem "cqt-basic:5": < $(\forall\alpha_1\dots\forall\alpha_n(\varphi\{\alpha_1\dots\alpha_n\})) \rightarrow \varphi\{\alpha_1\dots\alpha_n\}$ > (99.5)
765 using "cqt-orig:3" by blast
766
767 AOT_theorem "cqt-basic:6": < $\forall\alpha\forall\alpha \ \varphi\{\alpha\} \equiv \forall\alpha \ \varphi\{\alpha\}$ > (99.6)
768 by (meson "≡I" "→I" GEN "cqt-orig:1[const_var]")
769
770 AOT_theorem "cqt-basic:7": < $(\varphi \rightarrow \forall\alpha \ \psi\{\alpha\}) \equiv \forall\alpha(\varphi \rightarrow \psi\{\alpha\})$ > (99.7)
771 by (metis "→I" "vdash-properties:6" "rule-ui:3" "≡I" GEN)
772
773 AOT_theorem "cqt-basic:8": < $(\forall\alpha \ \varphi\{\alpha\} \ \vee \ \forall\alpha \ \psi\{\alpha\}) \rightarrow \forall\alpha (\varphi\{\alpha\} \ \vee \ \psi\{\alpha\})$ > (99.8)
774 by (simp add: "VI"(3) "→I" GEN "cqt-orig:1[const_var]")
775
776 AOT_theorem "cqt-basic:9": (99.9)
777 < $(\forall\alpha (\varphi\{\alpha\} \rightarrow \psi\{\alpha\}) \ \& \ \forall\alpha (\psi\{\alpha\} \rightarrow \chi\{\alpha\})) \rightarrow \forall\alpha(\varphi\{\alpha\} \rightarrow \chi\{\alpha\})$ >
778 proof -
779   {
780     AOT_assume < $\forall\alpha (\varphi\{\alpha\} \rightarrow \psi\{\alpha\})$ >
781     moreover AOT_assume < $\forall\alpha (\psi\{\alpha\} \rightarrow \chi\{\alpha\})$ >
782     ultimately AOT_have < $\varphi\{\alpha\} \rightarrow \psi\{\alpha\}$ > and < $\psi\{\alpha\} \rightarrow \chi\{\alpha\}$ > for  $\alpha$ 
783     using "VE" by blast+
784     AOT_hence < $\varphi\{\alpha\} \rightarrow \chi\{\alpha\}$ > for  $\alpha$  by (metis "→E" "→I")
785     AOT_hence < $\forall\alpha(\varphi\{\alpha\} \rightarrow \chi\{\alpha\})$ > using "VI" by fast
786   }
787   thus ?thesis using "&I" "→I" "&E" by meson
788 qed
789
790 AOT_theorem "cqt-basic:10": (99.10)
791 < $(\forall\alpha(\varphi\{\alpha\} \equiv \psi\{\alpha\}) \ \& \ \forall\alpha(\psi\{\alpha\} \equiv \chi\{\alpha\})) \rightarrow \forall\alpha (\varphi\{\alpha\} \equiv \chi\{\alpha\})$ >
792 proof(rule "→I"; rule "VI")
793   fix  $\beta$ 
794   AOT_assume < $\forall\alpha(\varphi\{\alpha\} \equiv \psi\{\alpha\}) \ \& \ \forall\alpha(\psi\{\alpha\} \equiv \chi\{\alpha\})$ >
795   AOT_hence < $\varphi\{\beta\} \equiv \psi\{\beta\}$ > and < $\psi\{\beta\} \equiv \chi\{\beta\}$ > using "&E" "VE" by blast+
796   AOT_thus < $\varphi\{\beta\} \equiv \chi\{\beta\}$ > using "≡I" "≡E" by blast
797 qed
798
799 AOT_theorem "cqt-basic:11": < $\forall\alpha(\varphi\{\alpha\} \equiv \psi\{\alpha\}) \equiv \forall\alpha (\psi\{\alpha\} \equiv \varphi\{\alpha\})$ > (99.11)
800 proof (rule "≡I"; rule "→I")
801   AOT_assume 0: < $\forall\alpha(\varphi\{\alpha\} \equiv \psi\{\alpha\})$ >
802   {
803     fix  $\alpha$ 
804     AOT_have < $\varphi\{\alpha\} \equiv \psi\{\alpha\}$ > using 0 "VE" by blast
805     AOT_hence < $\psi\{\alpha\} \equiv \varphi\{\alpha\}$ > using "≡I" "≡E" "→I" "→E" by metis
806   }
807   AOT_thus < $\forall\alpha(\psi\{\alpha\} \equiv \varphi\{\alpha\})$ > using "VI" by fast
808 next
809   AOT_assume 0: < $\forall\alpha(\psi\{\alpha\} \equiv \varphi\{\alpha\})$ >
810   {
811     fix  $\alpha$ 
812     AOT_have < $\psi\{\alpha\} \equiv \varphi\{\alpha\}$ > using 0 "VE" by blast
813     AOT_hence < $\varphi\{\alpha\} \equiv \psi\{\alpha\}$ > using "≡I" "≡E" "→I" "→E" by metis
814   }
815   AOT_thus < $\forall\alpha(\varphi\{\alpha\} \equiv \psi\{\alpha\})$ > using "VI" by fast
816 qed
817

```



```

818 AOT_theorem "cqt-basic:12": <∀α φ{α} → ∀α (ψ{α} → φ{α})> (99.12)
819   by (simp add: "∀E"(2) "→I" GEN)
820
821 AOT_theorem "cqt-basic:13": <∀α φ{α} ≡ ∀β φ{β}> (99.13)
822   using "≡I" "→I" by blast
823
824 AOT_theorem "cqt-basic:14": (99.14)
825   <(∀α1...∀αn (φ{α1...αn} → ψ{α1...αn})) →
826   ((∀α1...∀αn φ{α1...αn}) → (∀α1...∀αn ψ{α1...αn}))>
827   using "cqt:3"[axiom_inst] by auto
828
829 AOT_theorem "cqt-basic:15": (99.15)
830   <(∀α1...∀αn (φ → ψ{α1...αn})) → (φ → (∀α1...∀αn ψ{α1...αn}))>
831   using "cqt-orig:2" by auto
832
833 AOT_theorem "universal-cor": (100)
834   assumes <for arbitrary β: φ{β}>
835   shows <∀α φ{α}>
836   using GEN assms .
837
838 AOT_theorem "existential:1": (101.1)
839   assumes <φ{τ}> and <τ↓>
840   shows <∃α φ{α}>
841 proof(rule "raa-cor:1")
842   AOT_assume <¬∃α φ{α}>
843   AOT_hence <∀α ¬φ{α}>
844     using "≡defI" "conventions:4" RAA "&I" by blast
845   AOT_hence <¬φ{τ}> using assms(2) "∀E"(1) "→E" by blast
846   AOT_thus <φ{τ} & ¬φ{τ}> using assms(1) "&I" by blast
847 qed
848
849 AOT_theorem "existential:2[const_var]": (101.2)
850   assumes <φ{β}>
851   shows <∃α φ{α}>
852   using "existential:1" "cqt:2[const_var]"[axiom_inst] assms by blast
853
854 AOT_theorem "existential:2[lambda]": (101.2)
855   assumes <φ{[λν1...νn ψ{ν1...νn}]> and <INSTANCE_OF_CQT_2(ψ)>
856   shows <∃α φ{α}>
857   using "existential:1" "cqt:2[lambda]"[axiom_inst] assms by blast
858 lemmas "∃I" = "existential:1" "existential:2[const_var]"
859           "existential:2[lambda]"
860
861 AOT_theorem "instantiation": (102)
862   assumes <for arbitrary β: φ{β} ⊢ ψ> and <∃α φ{α}>
863   shows <ψ>
864   by (metis (no_types, lifting) "≡defE" GEN "raa-cor:3" "conventions:4" assms)
865 lemmas "∃E" = "instantiation"
866
867 AOT_theorem "cqt-further:1": <∀α φ{α} → ∃α φ{α}> (103.1)
868   using "∀E"(4) "∃I"(2) "→I" by metis
869
870 AOT_theorem "cqt-further:2": <¬∀α φ{α} → ∃α ¬φ{α}> (103.2)
871   using "∀I" "∃I"(2) "→I" RAA by metis
872
873 AOT_theorem "cqt-further:3": <∀α φ{α} ≡ ¬∃α ¬φ{α}> (103.3)
874   using "∀E"(4) "∃E" "→I" RAA
875   by (metis "cqt-further:2" "≡I" "modus-tollens:1")
876
877 AOT_theorem "cqt-further:4": <¬∃α φ{α} → ∀α ¬φ{α}> (103.4)
878   using "∀I" "∃I"(2)"→I" RAA by metis
879
880 AOT_theorem "cqt-further:5": <∃α (φ{α} & ψ{α}) → (∃α φ{α} & ∃α ψ{α})> (103.5)

```



```

881   by (metis (no_types, lifting) "&E" "&I" "∃E" "∃I"(2) "→I")
882
883 AOT_theorem "cqt-further:6": <∃α (φ{α} ∨ ψ{α}) → (∃α φ{α} ∨ ∃α ψ{α})>      (103.6)
884   by (metis (mono_tags, lifting) "∃E" "∃I"(2) "∨E"(3) "∨I"(1, 2) "→I" RAA(2))
885
886 (* NOTE: vacuous in the embedding *)
887 AOT_theorem "cqt-further:7": <∃α φ{α} ≡ ∃β φ{β}>      (103.7)
888   by (simp add: "oth-class-taut:3:a")
889
890 AOT_theorem "cqt-further:8":      (103.8)
891   <(∀α φ{α} & ∀α ψ{α}) → ∀α (φ{α} ≡ ψ{α})>
892   by (metis (mono_tags, lifting) "&E" "≡I" "∨E"(2) "→I" GEN)
893
894 AOT_theorem "cqt-further:9":      (103.9)
895   <(¬∃α φ{α} & ¬∃α ψ{α}) → ∀α (φ{α} ≡ ψ{α})>
896   by (metis (mono_tags, lifting) "&E" "≡I" "∃I"(2) "→I" GEN "raa-cor:4")
897
898 AOT_theorem "cqt-further:10":      (103.10)
899   <(∃α φ{α} & ¬∃α ψ{α}) → ¬∀α (φ{α} ≡ ψ{α})>
900 proof(rule "→I"; rule "raa-cor:2")
901   AOT_assume 0: <∃α φ{α} & ¬∃α ψ{α}>
902   then AOT_obtain α where <φ{α}> using "∃E" "&E"(1) by metis
903   moreover AOT_assume <∀α (φ{α} ≡ ψ{α})>
904   ultimately AOT_have <ψ{α}> using "∨E"(4) "≡E"(1) by blast
905   AOT_hence <∃α ψ{α}> using "∃I" by blast
906   AOT_thus <∃α ψ{α} & ¬∃α ψ{α}> using 0 "&E"(2) "&I" by blast
907 qed
908
909 AOT_theorem "cqt-further:11": <∃α∃β φ{α,β} ≡ ∃β∃α φ{α,β}>      (103.11)
910   using "≡I" "→I" "∃I"(2) "∃E" by metis
911
912 subsection<Logical Existence, Identity, and Truth>
913 text<\label{PLM: 9.7}>
914
915 AOT_theorem "log-prop-prop:1": <[λ φ]↓>      (104.1)
916   using "cqt:2[lambda0]"[axiom_inst] by auto
917
918 AOT_theorem "log-prop-prop:2": <φ↓>      (104.2)
919   by (rule "≡dfI"[OF "existence:3"]) "cqt:2[lambda]"
920
921 AOT_theorem "exist-nec": <τ↓ → □τ↓>      (106)
922 proof -
923   AOT_have <∀β □β↓>
924   by (simp add: GEN RN "cqt:2[const_var]"[axiom_inst])
925   AOT_thus <τ↓ → □τ↓>
926   using "cqt:1"[axiom_inst] "→E" by blast
927 qed
928
929 (* TODO: replace this mechanism by a "proof by types" command *)
930 class AOT_Term_id = AOT_Term +
931   assumes "t=t-proper:1"[AOT]: <[v | = τ = τ' → τ↓]>      (107.1)
932   and "t=t-proper:2"[AOT]: <[v | = τ = τ' → τ'↓]>      (107.2)
933
934 instance κ :: AOT_Term_id
935 proof
936   AOT_modally_strict {
937     AOT_show <κ = κ' → κ↓> for κ κ'
938     proof(rule "→I")
939       AOT_assume <κ = κ'>
940       AOT_hence <O!κ ∨ A!κ>
941       by (rule "∨I"(3)[OF "≡dfE"[OF "identity:1"]])
942       (meson "→I" "∨I"(1) "&E"(1))+
943       AOT_thus <κ↓>

```

```

944     by (rule "VE"(1))
945     (metis "cqt:5:a"[axiom_inst] "→I" "→E" "&E"(2))+
946   qed
947 }
948 next
949   AOT_modally_strict {
950     AOT_show < $\kappa = \kappa' \rightarrow \kappa' \downarrow$ > for  $\kappa \kappa'$ 
951     proof(rule "→I")
952       AOT_assume < $\kappa = \kappa'$ >
953       AOT_hence < $\text{O!}\kappa' \vee \text{A!}\kappa'$ >
954       by (rule "VI"(3)[OF "≡dfE"[OF "identity:1"]])
955       (meson "→I" "VI" "&E")+
956       AOT_thus < $\kappa' \downarrow$ >
957       by (rule "VE"(1))
958       (metis "cqt:5:a"[axiom_inst] "→I" "→E" "&E"(2))+
959     qed
960   }
961 qed
962
963 instance rel :: (AOT_κs) AOT_Term_id
964 proof
965   AOT_modally_strict {
966     AOT_show < $\Pi = \Pi' \rightarrow \Pi \downarrow$ > for  $\Pi \Pi' :: \langle\langle 'a \rangle\rangle$ 
967     proof(rule "→I")
968       AOT_assume < $\Pi = \Pi'$ >
969       AOT_thus < $\Pi \downarrow$ > using "≡dfE"[OF "identity:3"[of  $\Pi \Pi'$ ]] "&E" by blast
970     qed
971   }
972 next
973   AOT_modally_strict {
974     AOT_show < $\Pi = \Pi' \rightarrow \Pi' \downarrow$ > for  $\Pi \Pi' :: \langle\langle 'a \rangle\rangle$ 
975     proof(rule "→I")
976       AOT_assume < $\Pi = \Pi'$ >
977       AOT_thus < $\Pi' \downarrow$ > using "≡dfE"[OF "identity:3"[of  $\Pi \Pi'$ ]] "&E" by blast
978     qed
979   }
980 qed
981
982 instance o :: AOT_Term_id
983 proof
984   AOT_modally_strict {
985     fix  $\varphi \psi$ 
986     AOT_show < $\varphi = \psi \rightarrow \varphi \downarrow$ >
987     proof(rule "→I")
988       AOT_assume < $\varphi = \psi$ >
989       AOT_thus < $\varphi \downarrow$ > using "≡dfE"[OF "identity:4"[of  $\varphi \psi$ ]] "&E" by blast
990     qed
991   }
992 next
993   AOT_modally_strict {
994     fix  $\varphi \psi$ 
995     AOT_show < $\varphi = \psi \rightarrow \psi \downarrow$ >
996     proof(rule "→I")
997       AOT_assume < $\varphi = \psi$ >
998       AOT_thus < $\psi \downarrow$ > using "≡dfE"[OF "identity:4"[of  $\varphi \psi$ ]] "&E" by blast
999     qed
1000   }
1001 qed
1002
1003 instance prod :: (AOT_Term_id, AOT_Term_id) AOT_Term_id
1004 proof
1005   AOT_modally_strict {
1006     fix  $\tau \tau' :: \langle 'a \times 'b \rangle$ 

```

```

1007   AOT_show < $\tau = \tau' \rightarrow \tau \downarrow$ >
1008   proof (induct  $\tau$ ; induct  $\tau'$ ; rule " $\rightarrow$ I")
1009     fix  $\tau_1 \tau_1' :: 'a$  and  $\tau_2 \tau_2' :: 'b$ 
1010     AOT_assume <<( $\tau_1, \tau_2$ )>> = <<( $\tau_1', \tau_2'$ )>>
1011     AOT_hence <( $\tau_1 = \tau_1'$ ) & ( $\tau_2 = \tau_2'$ )> by (metis " $\equiv_{df}E$ " tuple_identity_1)
1012     AOT_hence < $\tau_1 \downarrow$ > and < $\tau_2 \downarrow$ >
1013     using "t=t-proper:1" "&E" "vdash-properties:10" by blast+
1014     AOT_thus <<( $\tau_1, \tau_2$ )>>  $\downarrow$  by (metis " $\equiv_{df}I$ " "&I" tuple_denotes)
1015   qed
1016 }
1017 next
1018   AOT_modally_strict {
1019     fix  $\tau \tau' :: <'a \times 'b>$ 
1020     AOT_show < $\tau = \tau' \rightarrow \tau \downarrow$ >
1021     proof (induct  $\tau$ ; induct  $\tau'$ ; rule " $\rightarrow$ I")
1022       fix  $\tau_1 \tau_1' :: 'a$  and  $\tau_2 \tau_2' :: 'b$ 
1023       AOT_assume <<( $\tau_1, \tau_2$ )>> = <<( $\tau_1', \tau_2'$ )>>
1024       AOT_hence <( $\tau_1 = \tau_1'$ ) & ( $\tau_2 = \tau_2'$ )> by (metis " $\equiv_{df}E$ " tuple_identity_1)
1025       AOT_hence < $\tau_1 \downarrow$ > and < $\tau_2 \downarrow$ >
1026       using "t=t-proper:2" "&E" "vdash-properties:10" by blast+
1027       AOT_thus <<( $\tau_1', \tau_2'$ )>>  $\downarrow$  by (metis " $\equiv_{df}I$ " "&I" tuple_denotes)
1028     qed
1029   }
1030 qed
1031
1032 (* This is the end of the "proof by types" and
1033    makes the results available on new theorems *)
1034 AOT_register_type_constraints
1035   Term: <_::AOT_Term_id> <_::AOT_Term_id>
1036 AOT_register_type_constraints
1037   Individual: < $\kappa$ > <_::{AOT_ks, AOT_Term_id}>
1038 AOT_register_type_constraints
1039   Relation: <<_::{AOT_ks, AOT_Term_id}>>
1040
1041 AOT_theorem "id-rel-nec-equiv:1":
1042   < $\Pi = \Pi' \rightarrow \Box \forall x_1 \dots \forall x_n ((\Box x_1 \dots x_n \equiv [\Pi'] x_1 \dots x_n))$ >
1043 proof(rule " $\rightarrow$ I")
1044   AOT_assume assumption: < $\Pi = \Pi'$ >
1045   AOT_hence < $\Pi \downarrow$ > and < $\Pi' \downarrow$ >
1046   using "t=t-proper:1" "t=t-proper:2" MP by blast+
1047   moreover AOT_have < $\forall F \forall G (F = G \rightarrow ((\Box \forall x_1 \dots \forall x_n (([F] x_1 \dots x_n \equiv [F] x_1 \dots x_n)) \rightarrow$ 
1048      $\Box \forall x_1 \dots \forall x_n ([F] x_1 \dots x_n \equiv [G] x_1 \dots x_n)))$ >
1049   apply (rule GEN)+ using "1-identity"[axiom_inst] by force
1050   ultimately AOT_have < $\Pi = \Pi' \rightarrow ((\Box \forall x_1 \dots \forall x_n ((\Box x_1 \dots x_n \equiv [\Pi] x_1 \dots x_n)) \rightarrow$ 
1051      $\Box \forall x_1 \dots \forall x_n ((\Box x_1 \dots x_n \equiv [\Pi'] x_1 \dots x_n)))$ >
1052   using " $\forall E$ "(1) by blast
1053   AOT_hence <(( $\Box \forall x_1 \dots \forall x_n ((\Box x_1 \dots x_n \equiv [\Pi] x_1 \dots x_n)) \rightarrow$ 
1054      $\Box \forall x_1 \dots \forall x_n ((\Box x_1 \dots x_n \equiv [\Pi'] x_1 \dots x_n))$ )>
1055   using assumption " $\rightarrow E$ " by blast
1056   moreover AOT_have < $\Box \forall x_1 \dots \forall x_n ((\Box x_1 \dots x_n \equiv [\Pi] x_1 \dots x_n))$ >
1057   by (simp add: RN "oth-class-taut:3:a" "universal-cor")
1058   ultimately AOT_show < $\Box \forall x_1 \dots \forall x_n ((\Box x_1 \dots x_n \equiv [\Pi'] x_1 \dots x_n))$ >
1059   using " $\rightarrow E$ " by blast
1060 qed
1061
1062 AOT_theorem "id-rel-nec-equiv:2": < $\varphi = \psi \rightarrow \Box(\varphi \equiv \psi)$ >
1063 proof(rule " $\rightarrow$ I")
1064   AOT_assume assumption: < $\varphi = \psi$ >
1065   AOT_hence < $\varphi \downarrow$ > and < $\psi \downarrow$ >
1066   using "t=t-proper:1" "t=t-proper:2" MP by blast+
1067   moreover AOT_have < $\forall p \forall q (p = q \rightarrow ((\Box(p \equiv p) \rightarrow \Box(p \equiv q))))$ >
1068   apply (rule GEN)+ using "1-identity"[axiom_inst] by force
1069   ultimately AOT_have < $\varphi = \psi \rightarrow (\Box(\varphi \equiv \varphi) \rightarrow \Box(\varphi \equiv \psi))$ >

```

```

1070     using "∀E"(1) by blast
1071 AOT_hence <□(φ ≡ φ) → □(φ ≡ ψ)>
1072     using assumption "→E" by blast
1073 moreover AOT_have <□(φ ≡ φ)>
1074     by (simp add: RN "oth-class-taut:3:a" "universal-cor")
1075 ultimately AOT_show <□(φ ≡ ψ)>
1076     using "→E" by blast
1077 qed
1078
1079 AOT_theorem "rule=E": (110)
1080     assumes <φ{τ}> and <τ = σ>
1081     shows <φ{σ}>
1082 proof -
1083     AOT_have <τ↓> and <σ↓>
1084     using assms(2) "t=t-proper:1" "t=t-proper:2" "→E" by blast+
1085 moreover AOT_have <∀α∀β(α = β → (φ{α} → φ{β}))>
1086     apply (rule GEN)+ using "l-identity"[axiom_inst] by blast
1087 ultimately AOT_have <τ = σ → (φ{τ} → φ{σ})>
1088     using "∀E"(1) by blast
1089 AOT_thus <φ{σ}> using assms "→E" by blast
1090 qed
1091
1092 AOT_theorem "propositions-lemma:1": <[λ φ] = φ> (111.1)
1093 proof -
1094     AOT_have <φ↓> by (simp add: "log-prop-prop:2")
1095 moreover AOT_have <∀p [λ p] = p>
1096     using "lambda-predicates:3[zero]"[axiom_inst] "∀I" by fast
1097 ultimately AOT_show <[λ φ] = φ>
1098     using "∀E" by blast
1099 qed
1100
1101 AOT_theorem "propositions-lemma:2": <[λ φ] ≡ φ> (111.2)
1102 proof -
1103     AOT_have <[λ φ] ≡ [λ φ]> by (simp add: "oth-class-taut:3:a")
1104     AOT_thus <[λ φ] ≡ φ> using "propositions-lemma:1" "rule=E" by blast
1105 qed
1106
1107 text<propositions-lemma:3 through propositions-lemma:5 hold implicitly>
1108
1109 AOT_theorem "propositions-lemma:6": <(φ ≡ ψ) ≡ ([λ φ] ≡ [λ ψ])> (111.6)
1110     by (metis "≡E"(1) "≡E"(5) "Associativity of ≡" "propositions-lemma:2")
1111
1112 text<dr-alphabetic-rules holds implicitly>
1113
1114 AOT_theorem "oa-exist:1": <0!↓> (115.1)
1115 proof -
1116     AOT_have <[λx ◇[E!]x]↓> by "cqt:2[lambda]"
1117     AOT_hence 1: <0! = [λx ◇[E!]x]>
1118     using "df-rules-terms[4]"[OF "oa:1", THEN "&E"(1)] "→E" by blast
1119     AOT_show <0!↓> using "t=t-proper:1"[THEN "→E", OF 1] by simp
1120 qed
1121
1122 AOT_theorem "oa-exist:2": <A!↓> (115.2)
1123 proof -
1124     AOT_have <[λx ¬◇[E!]x]↓> by "cqt:2[lambda]"
1125     AOT_hence 1: <A! = [λx ¬◇[E!]x]>
1126     using "df-rules-terms[4]"[OF "oa:2", THEN "&E"(1)] "→E" by blast
1127     AOT_show <A!↓> using "t=t-proper:1"[THEN "→E", OF 1] by simp
1128 qed
1129
1130 AOT_theorem "oa-exist:3": <0!x ∨ A!x> (115.3)
1131 proof(rule "raa-cor:1")
1132     AOT_assume <¬(0!x ∨ A!x)>

```

```

1133 AOT_hence A: <¬0!x> and B: <¬A!x>
1134   using "Disjunction Addition"(1) "modus-tollens:1"
1135     "∨I"(2) "raa-cor:5" by blast+
1136 AOT_have C: <0! = [λx ◇[E!]x]>
1137   by (rule "df-rules-terms[4]"[OF "oa:1", THEN "&E"(1), THEN "→E"]) "cqt:2"
1138 AOT_have D: <A! = [λx ¬◇[E!]x]>
1139   by (rule "df-rules-terms[4]"[OF "oa:2", THEN "&E"(1), THEN "→E"]) "cqt:2"
1140 AOT_have E: <¬[λx ◇[E!]x]>
1141   using A C "rule=E" by fast
1142 AOT_have F: <¬[λx ¬◇[E!]x]>
1143   using B D "rule=E" by fast
1144 AOT_have G: <[λx ◇[E!]x]x ≡ ◇[E!]x>
1145   by (rule "lambda-predicates:2"[axiom_inst, THEN "→E"]) "cqt:2"
1146 AOT_have H: <[λx ¬◇[E!]x]x ≡ ¬◇[E!]x>
1147   by (rule "lambda-predicates:2"[axiom_inst, THEN "→E"]) "cqt:2"
1148 AOT_show <¬◇[E!]x & ¬¬◇[E!]x> using G E "≡E" H F "≡E" "&I" by metis
1149 qed
1150
1151 AOT_theorem "p-identity-thm2:1": <F = G ≡ □∀x(x[F] ≡ x[G])> (116.1)
1152 proof -
1153   AOT_have <F = G ≡ F↓ & G↓ & □∀x(x[F] ≡ x[G])>
1154     using "identity:2" "df-rules-formulas[3]" "df-rules-formulas[4]"
1155       "→E" "&E" "≡I" "→I" by blast
1156   moreover AOT_have <F↓> and <G↓>
1157     by (auto simp: "cqt:2[const_var]"[axiom_inst])
1158   ultimately AOT_show <F = G ≡ □∀x(x[F] ≡ x[G])>
1159     using "≡S"(1) "&I" by blast
1160 qed
1161
1162 AOT_theorem "p-identity-thm2:2[2]": (116.2)
1163   <F = G ≡ ∀y₁([λx [F]xy₁] = [λx [G]xy₁] & [λx [F]y₁x] = [λx [G]y₁x])>
1164 proof -
1165   AOT_have <F = G ≡ F↓ & G↓ &
1166     ∀y₁([λx [F]xy₁] = [λx [G]xy₁] & [λx [F]y₁x] = [λx [G]y₁x])>
1167     using "identity:3[2]" "df-rules-formulas[3]" "df-rules-formulas[4]"
1168       "→E" "&E" "≡I" "→I" by blast
1169   moreover AOT_have <F↓> and <G↓>
1170     by (auto simp: "cqt:2[const_var]"[axiom_inst])
1171   ultimately show ?thesis
1172     using "≡S"(1) "&I" by blast
1173 qed
1174
1175 AOT_theorem "p-identity-thm2:2[3]": (116.2)
1176   <F = G ≡ ∀y₁∀y₂([λx [F]xy₁y₂] = [λx [G]xy₁y₂] &
1177     [λx [F]y₁xy₂] = [λx [G]y₁xy₂] &
1178     [λx [F]y₁y₂x] = [λx [G]y₁y₂x])>
1179 proof -
1180   AOT_have <F = G ≡ F↓ & G↓ & ∀y₁∀y₂([λx [F]xy₁y₂] = [λx [G]xy₁y₂] &
1181     [λx [F]y₁xy₂] = [λx [G]y₁xy₂] &
1182     [λx [F]y₁y₂x] = [λx [G]y₁y₂x])>
1183     using "identity:3[3]" "df-rules-formulas[3]" "df-rules-formulas[4]"
1184       "→E" "&E" "≡I" "→I" by blast
1185   moreover AOT_have <F↓> and <G↓>
1186     by (auto simp: "cqt:2[const_var]"[axiom_inst])
1187   ultimately show ?thesis
1188     using "≡S"(1) "&I" by blast
1189 qed
1190
1191 AOT_theorem "p-identity-thm2:2[4]": (116.2)
1192   <F = G ≡ ∀y₁∀y₂∀y₃([λx [F]xy₁y₂y₃] = [λx [G]xy₁y₂y₃] &
1193     [λx [F]y₁xy₂y₃] = [λx [G]y₁xy₂y₃] &
1194     [λx [F]y₁y₂xy₃] = [λx [G]y₁y₂xy₃] &
1195     [λx [F]y₁y₂y₃x] = [λx [G]y₁y₂y₃x])>

```

```

1196 proof -
1197   AOT_have <F = G ≡ F↓ & G↓ & ∀y1∀y2∀y3([λx [F]xy1y2y3] = [λx [G]xy1y2y3] &
1198                                     [λx [F]y1xy2y3] = [λx [G]y1xy2y3] &
1199                                     [λx [F]y1y2xy3] = [λx [G]y1y2xy3] &
1200                                     [λx [F]y1y2y3x] = [λx [G]y1y2y3x])>
1201   using "identity:3[4]" "df-rules-formulas[3]" "df-rules-formulas[4]"
1202   "→E" "&E" "≡I" "→I" by blast
1203   moreover AOT_have <F↓> and <G↓>
1204   by (auto simp: "cqt:2[const_var]"[axiom_inst])
1205   ultimately show ?thesis
1206   using "≡S"(1) "&I" by blast
1207 qed
1208
1209 AOT_theorem "p-identity-thm2:2": (116.2)
1210 <F = G ≡ ∀x1...∀xn «AOT_sem_proj_id x1xn (λ τ . «[F]τ») (λ τ . «[G]τ»)»>
1211 proof -
1212   AOT_have <F = G ≡ F↓ & G↓ &
1213           ∀x1...∀xn «AOT_sem_proj_id x1xn (λ τ . «[F]τ») (λ τ . «[G]τ»)»>
1214   using "identity:3" "df-rules-formulas[3]" "df-rules-formulas[4]"
1215   "→E" "&E" "≡I" "→I" by blast
1216   moreover AOT_have <F↓> and <G↓>
1217   by (auto simp: "cqt:2[const_var]"[axiom_inst])
1218   ultimately show ?thesis
1219   using "≡S"(1) "&I" by blast
1220 qed
1221
1222 AOT_theorem "p-identity-thm2:3": (116.3)
1223 <p = q ≡ [λx p] = [λx q]>
1224 proof -
1225   AOT_have <p = q ≡ p↓ & q↓ & [λx p] = [λx q]>
1226   using "identity:4" "df-rules-formulas[3]" "df-rules-formulas[4]"
1227   "→E" "&E" "≡I" "→I" by blast
1228   moreover AOT_have <p↓> and <q↓>
1229   by (auto simp: "cqt:2[const_var]"[axiom_inst])
1230   ultimately show ?thesis
1231   using "≡S"(1) "&I" by blast
1232 qed
1233
1234 class AOT_Term_id_2 = AOT_Term_id + assumes "id-eq:1": <[v |= α = α]> (117.1)
1235
1236 instance κ :: AOT_Term_id_2
1237 proof
1238   AOT_modally_strict {
1239     fix x
1240     {
1241       AOT_assume <O!x>
1242       moreover AOT_have <□∀F([F]x ≡ [F]x)>
1243       using RN GEN "oth-class-taut:3:a" by fast
1244       ultimately AOT_have <O!x & O!x & □∀F([F]x ≡ [F]x)> using "&I" by simp
1245     }
1246     moreover {
1247       AOT_assume <A!x>
1248       moreover AOT_have <□∀F(x[F] ≡ x[F])>
1249       using RN GEN "oth-class-taut:3:a" by fast
1250       ultimately AOT_have <A!x & A!x & □∀F(x[F] ≡ x[F])> using "&I" by simp
1251     }
1252     ultimately AOT_have <(O!x & O!x & □∀F([F]x ≡ [F]x)) ∨
1253                       (A!x & A!x & □∀F(x[F] ≡ x[F]))>
1254     using "oa-exist:3" "∨I"(1) "∨I"(2) "∨E"(3) "raa-cor:1" by blast
1255     AOT_thus <x = x>
1256     using "identity:1"[THEN "df-rules-formulas[4]"] "→E" by blast
1257   }
1258 qed

```

```

1259
1260 instance rel :: ("{AOT_κs,AOT_Term_id_2}") AOT_Term_id_2
1261 proof
1262   AOT_modally_strict {
1263     fix F :: "<'a> AOT_var"
1264     AOT_have 0: <[λx1...xn [F]x1...xn] = F>
1265       by (simp add: "lambda-predicates:3"[axiom_inst])
1266     AOT_have <[λx1...xn [F]x1...xn]↓>
1267       by "cqt:2[lambda]"
1268     AOT_hence <[λx1...xn [F]x1...xn] = [λx1...xn [F]x1...xn]>
1269       using "lambda-predicates:1"[axiom_inst] "→E" by blast
1270     AOT_show <F = F> using "rule=E" 0 by force
1271   }
1272 qed
1273
1274 instance o :: AOT_Term_id_2
1275 proof
1276   AOT_modally_strict {
1277     fix p
1278     AOT_have 0: <[λ p] = p>
1279       by (simp add: "lambda-predicates:3[zero]"[axiom_inst])
1280     AOT_have <[λ p]↓>
1281       by (rule "cqt:2[lambda0]"[axiom_inst])
1282     AOT_hence <[λ p] = [λ p]>
1283       using "lambda-predicates:1[zero]"[axiom_inst] "→E" by blast
1284     AOT_show <p = p> using "rule=E" 0 by force
1285   }
1286 qed
1287
1288 instance prod :: (AOT_Term_id_2, AOT_Term_id_2) AOT_Term_id_2
1289 proof
1290   AOT_modally_strict {
1291     fix α :: "<'a×'b> AOT_var"
1292     AOT_show <α = α>
1293     proof (induct)
1294       AOT_show <τ = τ> if <τ↓> for τ :: "<'a×'b>"
1295       using that
1296       proof (induct τ)
1297         fix τ1 :: 'a and τ2 :: 'b
1298         AOT_assume <<(τ1,τ2)>↓>
1299         AOT_hence <τ1↓> and <τ2↓>
1300         using "≡dfE" "&E" tuple_denotes by blast+
1301         AOT_hence <τ1 = τ1> and <τ2 = τ2>
1302         using "id-eq:1"[unvarify α] by blast+
1303         AOT_thus <<(τ1, τ2)> = <<(τ1, τ2)>>
1304         by (metis "≡dfI" "&I" tuple_identity_1)
1305       qed
1306     qed
1307   }
1308 qed
1309
1310 AOT_register_type_constraints
1311   Term: <_::AOT_Term_id_2> <_::AOT_Term_id_2>
1312 AOT_register_type_constraints
1313   Individual: <κ> <_::AOT_κs, AOT_Term_id_2>
1314 AOT_register_type_constraints
1315   Relation: <<_::AOT_κs, AOT_Term_id_2>>
1316
1317 AOT_theorem "id-eq:2": <α = β → β = α> (117.2)
1318   by (meson "rule=E" "deduction-theorem")
1319
1320 AOT_theorem "id-eq:3": <α = β & β = γ → α = γ> (117.3)
1321   using "rule=E" "→I" "&E" by blast

```

```

1322
1323 AOT_theorem "id-eq:4": < $\alpha = \beta \equiv \forall \gamma (\alpha = \gamma \equiv \beta = \gamma)$ > (117.4)
1324 proof (rule "≡I"; rule "→I")
1325   AOT_assume 0: < $\alpha = \beta$ >
1326   AOT_hence 1: < $\beta = \alpha$ > using "id-eq:2" "→E" by blast
1327   AOT_show < $\forall \gamma (\alpha = \gamma \equiv \beta = \gamma)$ >
1328     by (rule GEN) (metis "≡I" "→I" 0 "1" "rule=E")
1329 next
1330   AOT_assume < $\forall \gamma (\alpha = \gamma \equiv \beta = \gamma)$ >
1331   AOT_hence < $\alpha = \alpha \equiv \beta = \alpha$ > using "∀E"(2) by blast
1332   AOT_hence < $\alpha = \alpha \rightarrow \beta = \alpha$ > using "≡E"(1) "→I" by blast
1333   AOT_hence < $\beta = \alpha$ > using "id-eq:1" "→E" by blast
1334   AOT_thus < $\alpha = \beta$ > using "id-eq:2" "→E" by blast
1335 qed
1336
1337 AOT_theorem "rule=I:1": (118.1)
1338   assumes < $\tau \downarrow$ >
1339   shows < $\tau = \tau$ >
1340 proof -
1341   AOT_have < $\forall \alpha (\alpha = \alpha)$ >
1342     by (rule GEN) (metis "id-eq:1")
1343   AOT_thus < $\tau = \tau$ > using assms "∀E" by blast
1344 qed
1345
1346 AOT_theorem "rule=I:2[const_var]": " $\alpha = \alpha$ " (118.2)
1347   using "id-eq:1".
1348
1349 AOT_theorem "rule=I:2[lambda]": (118.2)
1350   assumes <INSTANCE_OF_CQT_2( $\varphi$ )>
1351   shows "[ $\lambda \nu_1 \dots \nu_n \varphi\{\nu_1 \dots \nu_n\}$ ] = [ $\lambda \nu_1 \dots \nu_n \varphi\{\nu_1 \dots \nu_n\}$ ]"
1352 proof -
1353   AOT_have < $\forall \alpha (\alpha = \alpha)$ >
1354     by (rule GEN) (metis "id-eq:1")
1355   moreover AOT_have < $[\lambda \nu_1 \dots \nu_n \varphi\{\nu_1 \dots \nu_n\}] \downarrow$ >
1356     using assms by (rule "cqt:2[lambda]"[axiom_inst])
1357   ultimately AOT_show < $[\lambda \nu_1 \dots \nu_n \varphi\{\nu_1 \dots \nu_n\}] = [\lambda \nu_1 \dots \nu_n \varphi\{\nu_1 \dots \nu_n\}]$ >
1358     using assms "∀E" by blast
1359 qed
1360
1361 lemmas "=I" = "rule=I:1" "rule=I:2[const_var]" "rule=I:2[lambda]"
1362
1363 AOT_theorem "rule-id-df:1": (120.1)
1364   assumes < $\tau\{\alpha_1 \dots \alpha_n\} =_{df} \sigma\{\alpha_1 \dots \alpha_n\}$ > and < $\sigma\{\tau_1 \dots \tau_n\} \downarrow$ >
1365   shows < $\tau\{\tau_1 \dots \tau_n\} = \sigma\{\tau_1 \dots \tau_n\}$ >
1366 proof -
1367   AOT_have < $\sigma\{\tau_1 \dots \tau_n\} \downarrow \rightarrow \tau\{\tau_1 \dots \tau_n\} = \sigma\{\tau_1 \dots \tau_n\}$ >
1368     using "df-rules-terms[3]" assms(1) "&E" by blast
1369   AOT_thus < $\tau\{\tau_1 \dots \tau_n\} = \sigma\{\tau_1 \dots \tau_n\}$ >
1370     using assms(2) "→E" by blast
1371 qed
1372
1373 AOT_theorem "rule-id-df:1[zero]": (120.1)
1374   assumes < $\tau =_{df} \sigma$ > and < $\sigma \downarrow$ >
1375   shows < $\tau = \sigma$ >
1376 proof -
1377   AOT_have < $\sigma \downarrow \rightarrow \tau = \sigma$ >
1378     using "df-rules-terms[4]" assms(1) "&E" by blast
1379   AOT_thus < $\tau = \sigma$ >
1380     using assms(2) "→E" by blast
1381 qed
1382
1383 AOT_theorem "rule-id-df:2:a": (120.2.a)
1384   assumes < $\tau\{\alpha_1 \dots \alpha_n\} =_{df} \sigma\{\alpha_1 \dots \alpha_n\}$ > and < $\sigma\{\tau_1 \dots \tau_n\} \downarrow$ > and < $\varphi\{\tau\{\tau_1 \dots \tau_n\}\}$ >

```



```

1385   shows <φ{σ{τ1...τn}}>
1386 proof -
1387   AOT_have <τ{τ1...τn} = σ{τ1...τn}> using "rule-id-df:1" assms(1,2) by blast
1388   AOT_thus <φ{σ{τ1...τn}}> using assms(3) "rule=E" by blast
1389 qed
1390
1391 AOT_theorem "rule-id-df:2:a[2]": (120.2.a)
1392   assumes <τ{«(α1,α2)»} =df σ{«(α1,α2)»}>
1393     and <σ{«(τ1,τ2)»}↓>
1394     and <φ{τ{«(τ1,τ2)»}}>
1395   shows <φ{σ{«(τ1::'a::AOT_Term_id_2,τ2::'b::AOT_Term_id_2)»}}>
1396 proof -
1397   AOT_have <τ{«(τ1,τ2)»} = σ{«(τ1,τ2)»}>
1398     using "rule-id-df:1" assms(1,2) by auto
1399   AOT_thus <φ{σ{«(τ1,τ2)»}}> using assms(3) "rule=E" by blast
1400 qed
1401
1402 AOT_theorem "rule-id-df:2:a[zero]": (120.2.a)
1403   assumes <τ =df σ> and <σ↓> and <φ{τ}>
1404   shows <φ{σ}>
1405 proof -
1406   AOT_have <τ = σ> using "rule-id-df:1[zero]" assms(1,2) by blast
1407   AOT_thus <φ{σ}> using assms(3) "rule=E" by blast
1408 qed
1409
1410 lemmas "=dfE" = "rule-id-df:2:a" "rule-id-df:2:a[zero]"
1411
1412 AOT_theorem "rule-id-df:2:b": (120.2.b)
1413   assumes <τ{α1...αn} =df σ{α1...αn}> and <σ{τ1...τn}↓> and <φ{σ{τ1...τn}}>
1414   shows <φ{τ{τ1...τn}}>
1415 proof -
1416   AOT_have <τ{τ1...τn} = σ{τ1...τn}>
1417     using "rule-id-df:1" assms(1,2) by blast
1418   AOT_hence <σ{τ1...τn} = τ{τ1...τn}>
1419     using "rule=E" "=I"(1) "t=t-proper:1" "→E" by fast
1420   AOT_thus <φ{τ{τ1...τn}}> using assms(3) "rule=E" by blast
1421 qed
1422
1423 AOT_theorem "rule-id-df:2:b[2]": (120.2.b)
1424   assumes <τ{«(α1,α2)»} =df σ{«(α1,α2)»}>
1425     and <σ{«(τ1,τ2)»}↓>
1426     and <φ{σ{«(τ1,τ2)»}}>
1427   shows <φ{τ{«(τ1::'a::AOT_Term_id_2,τ2::'b::AOT_Term_id_2)»}}>
1428 proof -
1429   AOT_have <τ{«(τ1,τ2)»} = σ{«(τ1,τ2)»}>
1430     using "=I"(1) "rule-id-df:2:a[2]" RAA(1) assms(1,2) "→I" by metis
1431   AOT_hence <σ{«(τ1,τ2)»} = τ{«(τ1,τ2)»}>
1432     using "rule=E" "=I"(1) "t=t-proper:1" "→E" by fast
1433   AOT_thus <φ{τ{«(τ1,τ2)»}}> using assms(3) "rule=E" by blast
1434 qed
1435
1436 AOT_theorem "rule-id-df:2:b[zero]": (120.2.b)
1437   assumes <τ =df σ> and <σ↓> and <φ{σ}>
1438   shows <φ{τ}>
1439 proof -
1440   AOT_have <τ = σ> using "rule-id-df:1[zero]" assms(1,2) by blast
1441   AOT_hence <σ = τ>
1442     using "rule=E" "=I"(1) "t=t-proper:1" "→E" by fast
1443   AOT_thus <φ{τ}> using assms(3) "rule=E" by blast
1444 qed
1445
1446 lemmas "=dfI" = "rule-id-df:2:b" "rule-id-df:2:b[zero]"
1447

```

1448 AOT_theorem "free-thms:1": $\langle \tau \downarrow \equiv \exists \beta (\beta = \tau) \rangle$ (121.1)
1449 by (metis "E" "rule=I:1" "t=t-proper:2" " \rightarrow I" " \exists I"(1) " \equiv I" " \rightarrow E")
1450
1451 AOT_theorem "free-thms:2": $\langle \forall \alpha \varphi\{\alpha\} \rightarrow (\exists \beta (\beta = \tau) \rightarrow \varphi\{\tau\}) \rangle$ (121.2)
1452 by (metis "E" "rule=E" "cqt:2[const_var]"[axiom_inst] " \rightarrow I" " \forall E"(1))
1453
1454 AOT_theorem "free-thms:3[const_var]": $\langle \exists \beta (\beta = \alpha) \rangle$ (121.3)
1455 by (meson " \exists I"(2) "id-eq:1")
1456
1457 AOT_theorem "free-thms:3[lambda]": (121.3)
1458 assumes $\langle \text{INSTANCE_OF_CQT_2}(\varphi) \rangle$
1459 shows $\langle \exists \beta (\beta = [\lambda \nu_1 \dots \nu_n \varphi\{\nu_1 \dots \nu_n\}]) \rangle$
1460 by (meson "=I"(3) assms "cqt:2[lambda]"[axiom_inst] "existential:1")
1461
1462 AOT_theorem "free-thms:4[rel]": (121.4)
1463 $\langle ([\text{II}] \kappa_1 \dots \kappa_n \vee \kappa_1 \dots \kappa_n [\text{II}]) \rightarrow \exists \beta (\beta = \text{II}) \rangle$
1464 by (metis "rule=I:1" "&E"(1) " \forall E"(1) "cqt:5:a"[axiom_inst]
1465 "cqt:5:b"[axiom_inst] " \rightarrow I" " \exists I"(1))
1466
1467 AOT_theorem "free-thms:4[vars]": (121.4)
1468 $\langle ([\text{II}] \kappa_1 \dots \kappa_n \vee \kappa_1 \dots \kappa_n [\text{II}]) \rightarrow \exists \beta_1 \dots \exists \beta_n (\beta_1 \dots \beta_n = \kappa_1 \dots \kappa_n) \rangle$
1469 by (metis "rule=I:1" "&E"(2) " \forall E"(1) "cqt:5:a"[axiom_inst]
1470 "cqt:5:b"[axiom_inst] " \rightarrow I" " \exists I"(1))
1471
1472 AOT_theorem "free-thms:4[1,rel]": (121.4)
1473 $\langle ([\text{II}] \kappa \vee \kappa [\text{II}]) \rightarrow \exists \beta (\beta = \text{II}) \rangle$
1474 by (metis "rule=I:1" "&E"(1) " \forall E"(1) "cqt:5:a"[axiom_inst]
1475 "cqt:5:b"[axiom_inst] " \rightarrow I" " \exists I"(1))
1476 AOT_theorem "free-thms:4[1,1]": (121.4)
1477 $\langle ([\text{II}] \kappa \vee \kappa [\text{II}]) \rightarrow \exists \beta (\beta = \kappa) \rangle$
1478 by (metis "rule=I:1" "&E"(2) " \forall E"(1) "cqt:5:a"[axiom_inst]
1479 "cqt:5:b"[axiom_inst] " \rightarrow I" " \exists I"(1))
1480
1481 AOT_theorem "free-thms:4[2,rel]": (121.4)
1482 $\langle ([\text{II}] \kappa_1 \kappa_2 \vee \kappa_1 \kappa_2 [\text{II}]) \rightarrow \exists \beta (\beta = \text{II}) \rangle$
1483 by (metis "rule=I:1" "&E"(1) " \forall E"(1) "cqt:5:a[2]"[axiom_inst]
1484 "cqt:5:b[2]"[axiom_inst] " \rightarrow I" " \exists I"(1))
1485 AOT_theorem "free-thms:4[2,1]": (121.4)
1486 $\langle ([\text{II}] \kappa_1 \kappa_2 \vee \kappa_1 \kappa_2 [\text{II}]) \rightarrow \exists \beta (\beta = \kappa_1) \rangle$
1487 by (metis "rule=I:1" "&E" " \forall E"(1) "cqt:5:a[2]"[axiom_inst]
1488 "cqt:5:b[2]"[axiom_inst] " \rightarrow I" " \exists I"(1))
1489 AOT_theorem "free-thms:4[2,2]": (121.4)
1490 $\langle ([\text{II}] \kappa_1 \kappa_2 \vee \kappa_1 \kappa_2 [\text{II}]) \rightarrow \exists \beta (\beta = \kappa_2) \rangle$
1491 by (metis "rule=I:1" "&E"(2) " \forall E"(1) "cqt:5:a[2]"[axiom_inst]
1492 "cqt:5:b[2]"[axiom_inst] " \rightarrow I" " \exists I"(1))
1493 AOT_theorem "free-thms:4[3,rel]": (121.4)
1494 $\langle ([\text{II}] \kappa_1 \kappa_2 \kappa_3 \vee \kappa_1 \kappa_2 \kappa_3 [\text{II}]) \rightarrow \exists \beta (\beta = \text{II}) \rangle$
1495 by (metis "rule=I:1" "&E"(1) " \forall E"(1) "cqt:5:a[3]"[axiom_inst]
1496 "cqt:5:b[3]"[axiom_inst] " \rightarrow I" " \exists I"(1))
1497 AOT_theorem "free-thms:4[3,1]": (121.4)
1498 $\langle ([\text{II}] \kappa_1 \kappa_2 \kappa_3 \vee \kappa_1 \kappa_2 \kappa_3 [\text{II}]) \rightarrow \exists \beta (\beta = \kappa_1) \rangle$
1499 by (metis "rule=I:1" "&E" " \forall E"(1) "cqt:5:a[3]"[axiom_inst]
1500 "cqt:5:b[3]"[axiom_inst] " \rightarrow I" " \exists I"(1))
1501 AOT_theorem "free-thms:4[3,2]": (121.4)
1502 $\langle ([\text{II}] \kappa_1 \kappa_2 \kappa_3 \vee \kappa_1 \kappa_2 \kappa_3 [\text{II}]) \rightarrow \exists \beta (\beta = \kappa_2) \rangle$
1503 by (metis "rule=I:1" "&E" " \forall E"(1) "cqt:5:a[3]"[axiom_inst]
1504 "cqt:5:b[3]"[axiom_inst] " \rightarrow I" " \exists I"(1))
1505 AOT_theorem "free-thms:4[3,3]": (121.4)
1506 $\langle ([\text{II}] \kappa_1 \kappa_2 \kappa_3 \vee \kappa_1 \kappa_2 \kappa_3 [\text{II}]) \rightarrow \exists \beta (\beta = \kappa_3) \rangle$
1507 by (metis "rule=I:1" "&E"(2) " \forall E"(1) "cqt:5:a[3]"[axiom_inst]
1508 "cqt:5:b[3]"[axiom_inst] " \rightarrow I" " \exists I"(1))
1509 AOT_theorem "free-thms:4[4,rel]": (121.4)
1510 $\langle ([\text{II}] \kappa_1 \kappa_2 \kappa_3 \kappa_4 \vee \kappa_1 \kappa_2 \kappa_3 \kappa_4 [\text{II}]) \rightarrow \exists \beta (\beta = \text{II}) \rangle$

```

1511   by (metis "rule=I:1" "&E"(1) "∨E"(1) "cqt:5:a[4]"[axiom_inst]
1512         "cqt:5:b[4]"[axiom_inst] "→I" "∃I"(1))
1513 AOT_theorem "free-thms:4[4,1]": (121.4)
1514   <([II] κ1κ2κ3κ4 ∨ κ1κ2κ3κ4 [II]) → ∃β (β = κ1)>
1515   by (metis "rule=I:1" "&E" "∨E"(1) "cqt:5:a[4]"[axiom_inst]
1516         "cqt:5:b[4]"[axiom_inst] "→I" "∃I"(1))
1517 AOT_theorem "free-thms:4[4,2]": (121.4)
1518   <([II] κ1κ2κ3κ4 ∨ κ1κ2κ3κ4 [II]) → ∃β (β = κ2)>
1519   by (metis "rule=I:1" "&E" "∨E"(1) "cqt:5:a[4]"[axiom_inst]
1520         "cqt:5:b[4]"[axiom_inst] "→I" "∃I"(1))
1521 AOT_theorem "free-thms:4[4,3]": (121.4)
1522   <([II] κ1κ2κ3κ4 ∨ κ1κ2κ3κ4 [II]) → ∃β (β = κ3)>
1523   by (metis "rule=I:1" "&E" "∨E"(1) "cqt:5:a[4]"[axiom_inst]
1524         "cqt:5:b[4]"[axiom_inst] "→I" "∃I"(1))
1525 AOT_theorem "free-thms:4[4,4]": (121.4)
1526   <([II] κ1κ2κ3κ4 ∨ κ1κ2κ3κ4 [II]) → ∃β (β = κ4)>
1527   by (metis "rule=I:1" "&E"(2) "∨E"(1) "cqt:5:a[4]"[axiom_inst]
1528         "cqt:5:b[4]"[axiom_inst] "→I" "∃I"(1))
1529
1530 AOT_theorem "ex:1:a": <∀α α↓> (123.1.a)
1531   by (rule GEN) (fact "cqt:2[const_var]"[axiom_inst])
1532 AOT_theorem "ex:1:b": <∀α∃β(β = α)> (123.1.b)
1533   by (rule GEN) (fact "free-thms:3[const_var]")
1534
1535 AOT_theorem "ex:2:a": <□α↓> (123.2.a)
1536   by (rule RN) (fact "cqt:2[const_var]"[axiom_inst])
1537 AOT_theorem "ex:2:b": <□∃β(β = α)> (123.2.b)
1538   by (rule RN) (fact "free-thms:3[const_var]")
1539
1540 AOT_theorem "ex:3:a": <□∀α α↓> (123.3.a)
1541   by (rule RN) (fact "ex:1:a")
1542 AOT_theorem "ex:3:b": <□∀α∃β(β = α)> (123.3.b)
1543   by (rule RN) (fact "ex:1:b")
1544
1545 AOT_theorem "ex:4:a": <∀α □α↓> (123.4.a)
1546   by (rule GEN; rule RN) (fact "cqt:2[const_var]"[axiom_inst])
1547 AOT_theorem "ex:4:b": <∀α□∃β(β = α)> (123.4.b)
1548   by (rule GEN; rule RN) (fact "free-thms:3[const_var]")
1549
1550 AOT_theorem "ex:5:a": <□∀α □α↓> (123.5.a)
1551   by (rule RN) (simp add: "ex:4:a")
1552 AOT_theorem "ex:5:b": <□∀α□∃β(β = α)> (123.5.b)
1553   by (rule RN) (simp add: "ex:4:b")
1554
1555 AOT_theorem "all-self=:1": <□∀α(α = α)> (124.1)
1556   by (rule RN; rule GEN) (fact "id-eq:1")
1557 AOT_theorem "all-self=:2": <∀α□(α = α)> (124.2)
1558   by (rule GEN; rule RN) (fact "id-eq:1")
1559
1560 AOT_theorem "id-nec:1": <α = β → □(α = β)> (125.1)
1561 proof(rule "→I")
1562   AOT_assume <α = β>
1563   moreover AOT_have <□(α = α)>
1564     by (rule RN) (fact "id-eq:1")
1565   ultimately AOT_show <□(α = β)> using "rule=E" by fast
1566 qed
1567
1568 AOT_theorem "id-nec:2": <τ = σ → □(τ = σ)> (125.2)
1569 proof(rule "→I")
1570   AOT_assume asm: <τ = σ>
1571   moreover AOT_have <τ↓>
1572     using calculation "t=t-proper:1" "→E" by blast
1573   moreover AOT_have <□(τ = τ)>

```

```

1574     using calculation "all-self=:2" "\V"(1) by blast
1575     ultimately AOT_show <□(τ = σ)> using "rule=E" by fast
1576 qed
1577
1578 AOT_theorem "term-out:1": <φ{α} ≡ ∃β (β = α & φ{β})> (126.1)
1579 proof (rule "≡I"; rule "→I")
1580   AOT_assume asm: <φ{α}>
1581   AOT_show <∃β (β = α & φ{β})>
1582     by (rule "∃I"(2)[where β=α]; rule "&I")
1583     (auto simp: "id-eq:1" asm)
1584 next
1585   AOT_assume 0: <∃β (β = α & φ{β})>
1586   AOT_obtain β where <β = α & φ{β}>
1587     using "∃E"[rotated, OF 0] by blast
1588   AOT_thus <φ{α}> using "&E" "rule=E" by blast
1589 qed
1590
1591 AOT_theorem "term-out:2": <τ↓ → (φ{τ} ≡ ∃α(α = τ & φ{α}))> (126.2)
1592 proof(rule "→I")
1593   AOT_assume <τ↓>
1594   moreover AOT_have <∀α (φ{α} ≡ ∃β (β = α & φ{β}))>
1595     by (rule GEN) (fact "term-out:1")
1596   ultimately AOT_show <φ{τ} ≡ ∃α(α = τ & φ{α})>
1597     using "\VE" by blast
1598 qed
1599
1600 AOT_theorem "term-out:3": (126.3)
1601 <(φ{α} & ∀β(φ{β} → β = α)) ≡ ∀β(φ{β} ≡ β = α)>
1602 apply (rule "≡I"; rule "→I")
1603   apply (frule "&E"(1))
1604   apply (drule "&E"(2))
1605   apply (rule GEN; rule "≡I"; rule "→I")
1606 using "rule-ui:2[const_var]" "\vdash-properties:5"
1607   apply blast
1608   apply (meson "rule=E" "id-eq:1")
1609   apply (rule "&I")
1610 using "id-eq:1" "≡E"(2) "rule-ui:3"
1611   apply blast
1612   apply (rule GEN; rule "→I")
1613 using "≡E"(1) "rule-ui:2[const_var]"
1614   by blast
1615
1616 (* Note: generalized alphabetic variant of the last theorem. *)
1617 AOT_theorem "term-out:4": (126.4)
1618 <(φ{β} & ∀α(φ{α} → α = β)) ≡ ∀α(φ{α} ≡ α = β)>
1619   using "term-out:3" .
1620
1621 (* TODO: Provide a nicer mechanism for introducing custom binders. *)
1622 AOT_define AOT_exists_unique :: <α ⇒ φ ⇒ φ> "uniqueness:1": (127.1)
1623 <<AOT_exists_unique φ> ≡df ∃α (φ{α} & ∀β (φ{β} → β = α))>
1624 syntax (input) "_AOT_exists_unique" :: <α ⇒ φ ⇒ φ> ("∃!_ _" [1,40])
1625 syntax (output) "_AOT_exists_unique" :: <α ⇒ φ ⇒ φ> ("∃!_'(_)'") [1,40])
1626 AOT_syntax_print_translations
1627   "_AOT_exists_unique τ φ" <= "CONST AOT_exists_unique (_abs τ φ)"
1628 syntax
1629   "_AOT_exists_unique_ellipse" :: <id_position ⇒ id_position ⇒ φ ⇒ φ>
1630   (<∃!_...∃!_ _> [1,40])
1631 parse_ast_translation<
1632 [(syntax_const<_AOT_exists_unique_ellipse>,
1633   fn ctx => fn [a,b,c] => Ast.mk_appl (Ast.Constant "AOT_exists_unique")
1634     [parseEllipseList "_AOT_vars" ctx [a,b],c]),
1635 (syntax_const<_AOT_exists_unique>,
1636   AOT_restricted_binder

```

```

1637     const_name<AOT_exists_unique>
1638     const_syntax<AOT_conj>]]>
1639 print_translation<AOT_syntax_print_translations [
1640   AOT_preserve_binder_abs_tr'
1641   const_syntax<AOT_exists_unique>
1642   syntax_const<_AOT_exists_unique>
1643   (syntax_const<_AOT_exists_unique_ellipse>, true)
1644   const_name<AOT_conj>,
1645   AOT_binder_trans
1646   @{theory}
1647   @{binding "AOT_exists_unique_binder"}
1648   syntax_const<_AOT_exists_unique>
1649 ]>
1650
1651
1652 context AOT_meta_syntax
1653 begin
1654 notation AOT_exists_unique (binder "∃!" 20)
1655 end
1656 context AOT_no_meta_syntax
1657 begin
1658 no_notation AOT_exists_unique (binder "∃!" 20)
1659 end
1660
1661 AOT_theorem "uniqueness:2": <∃!α φ{α} ≡ ∃α∀β(φ{β} ≡ β = α)> (127.2)
1662 proof(rule "≡I"; rule "→I")
1663   AOT_assume <∃!α φ{α}>
1664   AOT_hence <∃α (φ{α} & ∀β (φ{β} → β = α))>
1665     using "uniqueness:1" "≡dfE" by blast
1666   then AOT_obtain α where <φ{α} & ∀β (φ{β} → β = α)>
1667     using "instantiate"[rotated] by blast
1668   AOT_hence <∀β(φ{β} ≡ β = α)>
1669     using "term-out:3" "≡E" by blast
1670   AOT_thus <∃α∀β(φ{β} ≡ β = α)>
1671     using "∃I" by fast
1672 next
1673   AOT_assume <∃α∀β(φ{β} ≡ β = α)>
1674   then AOT_obtain α where <∀β (φ{β} ≡ β = α)>
1675     using "instantiate"[rotated] by blast
1676   AOT_hence <φ{α} & ∀β (φ{β} → β = α)>
1677     using "term-out:3" "≡E" by blast
1678   AOT_hence <∃α (φ{α} & ∀β (φ{β} → β = α))>
1679     using "∃I" by fast
1680   AOT_thus <∃!α φ{α}>
1681     using "uniqueness:1" "≡dfI" by blast
1682 qed
1683
1684 AOT_theorem "uni-most": <∃!α φ{α} → ∀β∀γ((φ{β} & φ{γ}) → β = γ)> (128)
1685 proof(rule "→I"; rule GEN; rule GEN; rule "→I")
1686   fix β γ
1687   AOT_assume <∃!α φ{α}>
1688   AOT_hence <∃α∀β(φ{β} ≡ β = α)>
1689     using "uniqueness:2" "≡E" by blast
1690   then AOT_obtain α where <∀β(φ{β} ≡ β = α)>
1691     using "instantiate"[rotated] by blast
1692   moreover AOT_assume <φ{β} & φ{γ}>
1693   ultimately AOT_have <β = α> and <γ = α>
1694     using "VE"(2) "&E" "≡E"(1,2) by blast+
1695   AOT_thus <β = γ>
1696     by (metis "rule=E" "id-eq:2" "→E")
1697 qed
1698
1699 AOT_theorem "nec-exist-!": <∀α(φ{α} → □φ{α}) → (∃!α φ{α} → ∃!α □φ{α})> (129)

```

```

1700 proof (rule "→I"; rule "→I")
1701   AOT_assume a: <∀α(φ{α} → □φ{α})>
1702   AOT_assume <∃!α φ{α}>
1703   AOT_hence <∃α (φ{α} & ∀β (φ{β} → β = α))>
1704     using "uniqueness:1" "≡dfE" by blast
1705   then AOT_obtain α where ξ: <φ{α} & ∀β (φ{β} → β = α)>
1706     using "instantiation"[rotated] by blast
1707   AOT_have <□φ{α}>
1708     using ξ a "&E" "∀E" "→E" by fast
1709   moreover AOT_have <∀β (□φ{β} → β = α)>
1710     apply (rule GEN; rule "→I")
1711     using ξ [THEN "&E"(2), THEN "∀E"(2), THEN "→E"]
1712     "qml:2"[axiom_inst, THEN "→E"] by blast
1713   ultimately AOT_have <(□φ{α} & ∀β (□φ{β} → β = α))>
1714     using "&I" by blast
1715   AOT_thus <∃!α □φ{α}>
1716     using "uniqueness:1" "≡dfI" "∃I" by fast
1717 qed
1718
1719 subsection<The Theory of Actuality and Descriptions>
1720 text<\label{PLM: 9.8}>
1721
1722 AOT_theorem "act-cond": <A(φ → ψ) → (Aφ → Aψ)> (130)
1723   using "→I" "≡E"(1) "logic-actual-nec:2"[axiom_inst] by blast
1724
1725 AOT_theorem "nec-imp-act": <□φ → Aφ> (131)
1726   by (metis "act-cond" "contraposition:1[2]" "≡E"(4)
1727     "qml:2"[THEN act_closure, axiom_inst]
1728     "qml-act:2"[axiom_inst] RAA(1) "→E" "→I")
1729
1730 AOT_theorem "act-conj-act:1": <A(Aφ → φ)> (132.1)
1731   using "→I" "≡E"(2) "logic-actual-nec:2"[axiom_inst]
1732     "logic-actual-nec:4"[axiom_inst] by blast
1733
1734 AOT_theorem "act-conj-act:2": <A(φ → Aφ)> (132.2)
1735   by (metis "→I" "≡E"(2, 4) "logic-actual-nec:2"[axiom_inst]
1736     "logic-actual-nec:4"[axiom_inst] RAA(1))
1737
1738 AOT_theorem "act-conj-act:3": <(Aφ & Aψ) → A(φ & ψ)> (132.3)
1739 proof -
1740   AOT_have <□(φ → (ψ → (φ & ψ)))>
1741     by (rule RN) (fact Adjunction)
1742   AOT_hence <A(φ → (ψ → (φ & ψ)))>
1743     using "nec-imp-act" "→E" by blast
1744   AOT_hence <Aφ → A(ψ → (φ & ψ))>
1745     using "act-cond" "→E" by blast
1746   moreover AOT_have <A(ψ → (φ & ψ)) → (Aψ → A(φ & ψ))>
1747     by (fact "act-cond")
1748   ultimately AOT_have <Aφ → (Aψ → A(φ & ψ))>
1749     using "→I" "→E" by metis
1750   AOT_thus <(Aφ & Aψ) → A(φ & ψ)>
1751     by (metis Importation "→E")
1752 qed
1753
1754 AOT_theorem "act-conj-act:4": <A(Aφ ≡ φ)> (132.4)
1755 proof -
1756   AOT_have <(A(Aφ → φ) & A(φ → Aφ)) → A((Aφ → φ) & (φ → Aφ))>
1757     by (fact "act-conj-act:3")
1758   moreover AOT_have <A(Aφ → φ) & A(φ → Aφ)>
1759     using "&I" "act-conj-act:1" "act-conj-act:2" by simp
1760   ultimately AOT_have ζ: <A((Aφ → φ) & (φ → Aφ))>
1761     using "→E" by blast
1762   AOT_have <A(((Aφ → φ) & (φ → Aφ)) → (Aφ ≡ φ))>

```

```

1763     using "conventions:3"[THEN "df-rules-formulas[2]",
1764           THEN act_closure, axiom_inst] by blast
1765 AOT_hence <math>\mathcal{A}((\mathcal{A}\varphi \rightarrow \varphi) \ \& \ (\varphi \rightarrow \mathcal{A}\varphi)) \rightarrow \mathcal{A}(\mathcal{A}\varphi \equiv \varphi)>
1766     using "act-cond" "\rightarrow E" by blast
1767 AOT_thus <math>\mathcal{A}(\mathcal{A}\varphi \equiv \varphi)> \text{ using } \zeta \text{ "\rightarrow E" by blast}
1768 qed
1769
1770 (* TODO: Consider introducing AOT_inductive. *)
1771 inductive arbitrary_actualization for  $\varphi$  where
1772   <math>\langle \text{arbitrary\_actualization } \varphi \ll \mathcal{A}\varphi \gg\rangle
1773 | <math>\langle \text{arbitrary\_actualization } \varphi \ll \mathcal{A}\psi \gg\rangle \text{ if } \langle \text{arbitrary\_actualization } \varphi \ \psi \rangle
1774 declare arbitrary_actualization.cases[AOT]
1775       arbitrary_actualization.induct[AOT]
1776       arbitrary_actualization.simps[AOT]
1777       arbitrary_actualization.intros[AOT]
1778 syntax arbitrary_actualization :: <math>\langle \varphi' \Rightarrow \varphi' \Rightarrow \text{AOT\_prop} \rangle
1779   ("ARBITRARY\_ACTUALIZATION'(_,_)")
1780
1781 notepad
1782 begin
1783   AOT_modally_strict {
1784     fix  $\varphi$ 
1785     AOT_have <math>\langle \text{ARBITRARY\_ACTUALIZATION}(\mathcal{A}\varphi \equiv \varphi, \mathcal{A}(\mathcal{A}\varphi \equiv \varphi)) \rangle
1786       using AOT_PLM.arbitrary_actualization.intros by metis
1787     AOT_have <math>\langle \text{ARBITRARY\_ACTUALIZATION}(\mathcal{A}\varphi \equiv \varphi, \mathcal{A}\mathcal{A}(\mathcal{A}\varphi \equiv \varphi)) \rangle
1788       using AOT_PLM.arbitrary_actualization.intros by metis
1789     AOT_have <math>\langle \text{ARBITRARY\_ACTUALIZATION}(\mathcal{A}\varphi \equiv \varphi, \mathcal{A}\mathcal{A}\mathcal{A}(\mathcal{A}\varphi \equiv \varphi)) \rangle
1790       using AOT_PLM.arbitrary_actualization.intros by metis
1791   }
1792 end
1793
1794
1795 AOT_theorem "closure-act:1": (133.1)
1796   assumes <math>\langle \text{ARBITRARY\_ACTUALIZATION}(\mathcal{A}\varphi \equiv \varphi, \psi) \rangle
1797   shows <math>\langle \psi \rangle
1798 using assms proof(induct)
1799   case 1
1800   AOT_show <math>\langle \mathcal{A}(\mathcal{A}\varphi \equiv \varphi) \rangle
1801     by (simp add: "act-conj-act:4")
1802 next
1803   case (2  $\psi$ )
1804   AOT_thus <math>\langle \mathcal{A}\psi \rangle
1805     by (metis arbitrary_actualization.simps "\equiv E"(1)
1806         "logic-actual-nec:4"[axiom_inst])
1807 qed
1808
1809 AOT_theorem "closure-act:2": <math>\langle \forall \alpha \ \mathcal{A}(\mathcal{A}\varphi\{\alpha\} \equiv \varphi\{\alpha\}) \rangle (133.2)
1810   by (simp add: "act-conj-act:4" "\forall I")
1811
1812 AOT_theorem "closure-act:3": <math>\langle \mathcal{A}\forall \alpha \ \mathcal{A}(\mathcal{A}\varphi\{\alpha\} \equiv \varphi\{\alpha\}) \rangle (133.3)
1813   by (metis (no_types, lifting) "act-conj-act:4" "\equiv E"(1,2) "\forall I"
1814       "logic-actual-nec:3"[axiom_inst]
1815       "logic-actual-nec:4"[axiom_inst])
1816
1817 AOT_theorem "closure-act:4": <math>\langle \mathcal{A}\forall \alpha_1 \dots \forall \alpha_n \ \mathcal{A}(\mathcal{A}\varphi\{\alpha_1 \dots \alpha_n\} \equiv \varphi\{\alpha_1 \dots \alpha_n\}) \rangle (133.4)
1818   using "closure-act:3" .
1819
1820 AOT_act_theorem "RA[1]": (134)
1821   assumes <math>\langle \vdash \varphi \rangle
1822   shows <math>\langle \vdash \mathcal{A}\varphi \rangle
1823   - <math>\langle \text{While this proof is rejected in PLM,}
1824     \text{ we merely state it as modally-fragile rule,}
1825     \text{ which addresses the concern in PLM.} \rangle

```



```

1826   using "¬¬E" assms "≡E"(3) "logic-actual"[act_axiom_inst]
1827       "logic-actual-nec:1"[axiom_inst] "modus-tollens:2" by blast
1828 AOT_theorem "RA[2]": (134)
1829   assumes <math>\vdash_{\square} \varphi>
1830   shows <math>\vdash_{\square} \mathcal{A}\varphi>
1831   - <This rule is in fact a consequence of RN and
1832     does not require an appeal to the semantics itself.>
1833   using RN assms "nec-imp-act" "vdash-properties:5" by blast
1834 AOT_theorem "RA[3]":
1835   assumes <math>\Gamma \vdash_{\square} \varphi>
1836   shows <math>\mathcal{A}\Gamma \vdash_{\square} \mathcal{A}\varphi>
1837   text<This rule is only derivable from the semantics,
1838     but apparently no proof actually relies on it.
1839     If this turns out to be required, it is valid to derive it from the
1840     semantics just like RN, but we refrain from doing so, unless necessary.>
1841   (* using assms by (meson AOT_sem_act imageI) *)
1842   oops - <discard the rule>
1843
1844 AOT_act_theorem "ANeg:1": <math>\neg \mathcal{A}\varphi \equiv \neg \varphi> (137.1)
1845   by (simp add: "RA[1]" "contraposition:1[1]" "deduction-theorem"
1846       "≡I" "logic-actual"[act_axiom_inst])
1847
1848 AOT_act_theorem "ANeg:2": <math>\neg \mathcal{A}\neg \varphi \equiv \varphi> (137.2)
1849   using "ANeg:1" "≡I" "≡E"(5) "useful-tautologies:1"
1850       "useful-tautologies:2" by blast
1851
1852 AOT_theorem "Act-Basic:1": <math>\mathcal{A}\varphi \vee \mathcal{A}\neg \varphi> (138.1)
1853   by (meson "∀I"(1,2) "≡E"(2) "logic-actual-nec:1"[axiom_inst] "raa-cor:1")
1854
1855 AOT_theorem "Act-Basic:2": <math>\mathcal{A}(\varphi \ \& \ \psi) \equiv (\mathcal{A}\varphi \ \& \ \mathcal{A}\psi)> (138.2)
1856 proof (rule "≡I"; rule "→I")
1857   AOT_assume <math>\mathcal{A}(\varphi \ \& \ \psi)>
1858   moreover AOT_have <math>\mathcal{A}((\varphi \ \& \ \psi) \rightarrow \varphi)>
1859     by (simp add: "RA[2]" "Conjunction Simplification"(1))
1860   moreover AOT_have <math>\mathcal{A}((\varphi \ \& \ \psi) \rightarrow \psi)>
1861     by (simp add: "RA[2]" "Conjunction Simplification"(2))
1862   ultimately AOT_show <math>\mathcal{A}\varphi \ \& \ \mathcal{A}\psi>
1863     using "act-cond"[THEN "→E", THEN "→E"] "&I" by metis
1864 next
1865   AOT_assume <math>\mathcal{A}\varphi \ \& \ \mathcal{A}\psi>
1866   AOT_thus <math>\mathcal{A}(\varphi \ \& \ \psi)>
1867     using "act-conj-act:3" "vdash-properties:6" by blast
1868 qed
1869
1870 AOT_theorem "Act-Basic:3": <math>\mathcal{A}(\varphi \equiv \psi) \equiv (\mathcal{A}(\varphi \rightarrow \psi) \ \& \ \mathcal{A}(\psi \rightarrow \varphi))> (138.3)
1871 proof (rule "≡I"; rule "→I")
1872   AOT_assume <math>\mathcal{A}(\varphi \equiv \psi)>
1873   moreover AOT_have <math>\mathcal{A}((\varphi \equiv \psi) \rightarrow (\varphi \rightarrow \psi))>
1874     by (simp add: "RA[2]" "deduction-theorem" "≡E"(1))
1875   moreover AOT_have <math>\mathcal{A}((\varphi \equiv \psi) \rightarrow (\psi \rightarrow \varphi))>
1876     by (simp add: "RA[2]" "deduction-theorem" "≡E"(2))
1877   ultimately AOT_show <math>\mathcal{A}(\varphi \rightarrow \psi) \ \& \ \mathcal{A}(\psi \rightarrow \varphi)>
1878     using "act-cond"[THEN "→E", THEN "→E"] "&I" by metis
1879 next
1880   AOT_assume <math>\mathcal{A}(\varphi \rightarrow \psi) \ \& \ \mathcal{A}(\psi \rightarrow \varphi)>
1881   AOT_hence <math>\mathcal{A}((\varphi \rightarrow \psi) \ \& \ (\psi \rightarrow \varphi))>
1882     by (metis "act-conj-act:3" "vdash-properties:10")
1883   moreover AOT_have <math>\mathcal{A}(((\varphi \rightarrow \psi) \ \& \ (\psi \rightarrow \varphi)) \rightarrow (\varphi \equiv \psi))>
1884     by (simp add: "conventions:3" "RA[2]" "df-rules-formulas[2]"
1885         "vdash-properties:1[2]")
1886   ultimately AOT_show <math>\mathcal{A}(\varphi \equiv \psi)>
1887     using "act-cond"[THEN "→E", THEN "→E"] by metis
1888 qed

```



```

1889
1890 AOT_theorem "Act-Basic:4": <( $\mathcal{A}(\varphi \rightarrow \psi) \ \& \ \mathcal{A}(\psi \rightarrow \varphi)$ )  $\equiv$  ( $\mathcal{A}\varphi \equiv \mathcal{A}\psi$ )> (138.4)
1891 proof (rule "≡I"; rule "→I")
1892   AOT_assume 0: < $\mathcal{A}(\varphi \rightarrow \psi) \ \& \ \mathcal{A}(\psi \rightarrow \varphi)$ >
1893   AOT_show < $\mathcal{A}\varphi \equiv \mathcal{A}\psi$ >
1894     using 0 "&E" "act-cond"[THEN "→E", THEN "→E"] "≡I" "→I" by metis
1895 next
1896   AOT_assume < $\mathcal{A}\varphi \equiv \mathcal{A}\psi$ >
1897   AOT_thus < $\mathcal{A}(\varphi \rightarrow \psi) \ \& \ \mathcal{A}(\psi \rightarrow \varphi)$ >
1898     by (metis "→I" "logic-actual-nec:2"[axiom_inst] "≡E"(1,2) "&I")
1899 qed
1900
1901 AOT_theorem "Act-Basic:5": < $\mathcal{A}(\varphi \equiv \psi) \equiv (\mathcal{A}\varphi \equiv \mathcal{A}\psi)$ > (138.5)
1902   using "Act-Basic:3" "Act-Basic:4" "≡E"(5) by blast
1903
1904 AOT_theorem "Act-Basic:6": < $\mathcal{A}\varphi \equiv \Box\mathcal{A}\varphi$ > (138.6)
1905   by (simp add: "≡I" "qml:2"[axiom_inst] "qml-act:1"[axiom_inst])
1906
1907 AOT_theorem "Act-Basic:7": < $\mathcal{A}\Box\varphi \rightarrow \Box\mathcal{A}\varphi$ > (138.7)
1908   by (metis "Act-Basic:6" "→I" "→E" "≡E"(1,2) "nec-imp-act"
1909     "qml-act:2"[axiom_inst])
1910
1911 AOT_theorem "Act-Basic:8": < $\Box\varphi \rightarrow \Box\mathcal{A}\varphi$ > (138.8)
1912   using "Hypothetical Syllogism" "nec-imp-act" "qml-act:1"[axiom_inst] by blast
1913
1914 AOT_theorem "Act-Basic:9": < $\mathcal{A}(\varphi \vee \psi) \equiv (\mathcal{A}\varphi \vee \mathcal{A}\psi)$ > (138.9)
1915 proof (rule "≡I"; rule "→I")
1916   AOT_assume < $\mathcal{A}(\varphi \vee \psi)$ >
1917   AOT_thus < $\mathcal{A}\varphi \vee \mathcal{A}\psi$ >
1918   proof (rule "raa-cor:3")
1919     AOT_assume < $\neg(\mathcal{A}\varphi \vee \mathcal{A}\psi)$ >
1920     AOT_hence < $\neg\mathcal{A}\varphi \ \& \ \neg\mathcal{A}\psi$ >
1921       by (metis "≡E"(1) "oth-class-taut:5:d")
1922     AOT_hence < $\mathcal{A}\neg\varphi \ \& \ \mathcal{A}\neg\psi$ >
1923       using "logic-actual-nec:1"[axiom_inst, THEN "≡E"(2)] "&E" "&I" by metis
1924     AOT_hence < $\mathcal{A}(\neg\varphi \ \& \ \neg\psi)$ >
1925       using "≡E" "Act-Basic:2" by metis
1926     moreover AOT_have < $\mathcal{A}(\neg\varphi \ \& \ \neg\psi) \equiv \neg(\varphi \vee \psi)$ >
1927       using "RA[2]" "≡E"(6) "oth-class-taut:3:a" "oth-class-taut:5:d" by blast
1928     moreover AOT_have < $\mathcal{A}(\neg\varphi \ \& \ \neg\psi) \equiv \mathcal{A}(\neg(\varphi \vee \psi))$ >
1929       using calculation(2) by (metis "Act-Basic:5" "≡E"(1))
1930     ultimately AOT_have < $\mathcal{A}(\neg(\varphi \vee \psi))$ > using "≡E" by blast
1931     AOT_thus < $\neg\mathcal{A}(\varphi \vee \psi)$ >
1932     using "logic-actual-nec:1"[axiom_inst, THEN "≡E"(1)] by auto
1933   qed
1934 next
1935   AOT_assume < $\mathcal{A}\varphi \vee \mathcal{A}\psi$ >
1936   AOT_thus < $\mathcal{A}(\varphi \vee \psi)$ >
1937     by (meson "RA[2]" "act-cond" "\veeI"(1) "\veeE"(1) "Disjunction Addition"(1,2))
1938   qed
1939
1940 AOT_theorem "Act-Basic:10": < $\mathcal{A}\exists\alpha \varphi\{\alpha\} \equiv \exists\alpha \mathcal{A}\varphi\{\alpha\}$ > (138.10)
1941 proof -
1942   AOT_have  $\vartheta$ : < $\neg\mathcal{A}\forall\alpha \neg\varphi\{\alpha\} \equiv \neg\forall\alpha \mathcal{A}\neg\varphi\{\alpha\}$ >
1943     by (rule "oth-class-taut:4:b"[THEN "≡E"(1)])
1944     (metis "logic-actual-nec:3"[axiom_inst])
1945   AOT_have  $\xi$ : < $\neg\forall\alpha \mathcal{A}\neg\varphi\{\alpha\} \equiv \neg\forall\alpha \neg\mathcal{A}\varphi\{\alpha\}$ >
1946     by (rule "oth-class-taut:4:b"[THEN "≡E"(1)])
1947     (rule "logic-actual-nec:1"[THEN universal_closure,
1948       axiom_inst, THEN "cqt-basic:3"[THEN "→E"]])
1949   AOT_have < $\mathcal{A}(\exists\alpha \varphi\{\alpha\}) \equiv \mathcal{A}(\neg\forall\alpha \neg\varphi\{\alpha\})$ >
1950     using "conventions:4"[THEN "df-rules-formulas[1]",
1951       THEN act_closure, axiom_inst]

```

```

1952     "conventions:4"[THEN "df-rules-formulas[2]",
1953         THEN act_closure, axiom_inst]
1954     "Act-Basic:4"[THEN "≡E"(1)] "&I" "Act-Basic:5"[THEN "≡E"(2)] by metis
1955     also AOT_have <... ≡ ¬∀α ¬φ{α}>
1956     by (simp add: "logic-actual-nec:1" "vdash-properties:1[2]")
1957     also AOT_have <... ≡ ¬∀α A ¬φ{α}> using ∅ by blast
1958     also AOT_have <... ≡ ¬∀α ¬A φ{α}> using ξ by blast
1959     also AOT_have <... ≡ ∃α A φ{α}>
1960     using "conventions:4"[THEN "≡Df"] by (metis "≡E"(6) "oth-class-taut:3:a")
1961     finally AOT_show <A∃α φ{α} ≡ ∃α Aφ{α}> .
1962 qed
1963
1964
1965 AOT_theorem "Act-Basic:11": (138.11)
1966   <A∀α(φ{α} ≡ ψ{α}) ≡ ∀α(Aφ{α} ≡ Aψ{α})>
1967 proof(rule "≡I"; rule "→I")
1968   AOT_assume <A∀α(φ{α} ≡ ψ{α})>
1969   AOT_hence <∀αA(φ{α} ≡ ψ{α})>
1970     using "logic-actual-nec:3"[axiom_inst, THEN "≡E"(1)] by blast
1971   AOT_hence <A(φ{α} ≡ ψ{α})> for α using "∀E" by blast
1972   AOT_hence <Aφ{α} ≡ Aψ{α}> for α by (metis "Act-Basic:5" "≡E"(1))
1973   AOT_thus <∀α(Aφ{α} ≡ Aψ{α})> by (rule "∀I")
1974 next
1975   AOT_assume <∀α(Aφ{α} ≡ Aψ{α})>
1976   AOT_hence <Aφ{α} ≡ Aψ{α}> for α using "∀E" by blast
1977   AOT_hence <A(φ{α} ≡ ψ{α})> for α by (metis "Act-Basic:5" "≡E"(2))
1978   AOT_hence <∀α A(φ{α} ≡ ψ{α})> by (rule "∀I")
1979   AOT_thus <A∀α(φ{α} ≡ ψ{α})>
1980     using "logic-actual-nec:3"[axiom_inst, THEN "≡E"(2)] by fast
1981 qed
1982
1983 AOT_act_theorem "act-quant-uniq": (139)
1984   <∀β(Aφ{β} ≡ β = α) ≡ ∀β(φ{β} ≡ β = α)>
1985 proof(rule "≡I"; rule "→I")
1986   AOT_assume <∀β(Aφ{β} ≡ β = α)>
1987   AOT_hence <Aφ{β} ≡ β = α> for β using "∀E" by blast
1988   AOT_hence <φ{β} ≡ β = α> for β
1989     using "≡I" "→I" "RA[1]" "≡E"(1,2) "logic-actual"[act_axiom_inst] "→E"
1990     by metis
1991   AOT_thus <∀β(φ{β} ≡ β = α)> by (rule "∀I")
1992 next
1993   AOT_assume <∀β(φ{β} ≡ β = α)>
1994   AOT_hence <φ{β} ≡ β = α> for β using "∀E" by blast
1995   AOT_hence <Aφ{β} ≡ β = α> for β
1996     using "≡I" "→I" "RA[1]" "≡E"(1,2) "logic-actual"[act_axiom_inst] "→E"
1997     by metis
1998   AOT_thus <∀β(Aφ{β} ≡ β = α)> by (rule "∀I")
1999 qed
2000
2001 AOT_act_theorem "fund-cont-desc": <x = ιx(φ{x}) ≡ ∀z(φ{z} ≡ z = x)> (140)
2002   using descriptions[axiom_inst] "act-quant-uniq" "≡E"(5) by fast
2003
2004 AOT_act_theorem hintikka: <x = ιx(φ{x}) ≡ (φ{x} & ∀z (φ{z} → z = x))> (141)
2005   using "Commutativity of ≡"[THEN "≡E"(1)] "term-out:3"
2006     "fund-cont-desc" "≡E"(5) by blast
2007
2008
2009 locale russell_axiom =
2010   fixes ψ
2011   assumes ψ_denotes_asm: "[v ⊨ ψ{κ}] ⇒ [v ⊨ κ↓]"
2012 begin
2013 AOT_act_theorem "russell-axiom": (142)
2014   <ψ{ιx φ{x}} ≡ ∃x(φ{x} & ∀z(φ{z} → z = x) & ψ{x})>

```

```

2015 proof -
2016   AOT_have b: <∀x (x = ιx φ{x} ≡ (φ{x} & ∀z(φ{z} → z = x)))>
2017   using hintikka "∀I" by fast
2018   show ?thesis
2019   proof(rule "≡I"; rule "→I")
2020     AOT_assume c: <ψ{ιx φ{x}}>
2021     AOT_hence d: <ιx φ{x}↓>
2022     using ψ_denotes_asm by blast
2023     AOT_hence <∃y (y = ιx φ{x})>
2024     by (metis "rule=I:1" "existential:1")
2025     then AOT_obtain a where a_def: <a = ιx φ{x}>
2026     using "instantiation"[rotated] by blast
2027     moreover AOT_have <a = ιx φ{x} ≡ (φ{a} & ∀z(φ{z} → z = a))>
2028     using b "∀E" by blast
2029     ultimately AOT_have <φ{a} & ∀z(φ{z} → z = a)>
2030     using "≡E" by blast
2031     moreover AOT_have <ψ{a}>
2032     proof -
2033       AOT_have 1: <∀x∀y(x = y → y = x)>
2034       by (simp add: "id-eq:2" "universal-cor")
2035       AOT_have <a = ιx φ{x} → ιx φ{x} = a>
2036       by (rule "∀E"(1)[where τ="ιx φ{x}"]; rule "∀E"(2)[where β=a])
2037       (auto simp: 1 d "universal-cor")
2038       AOT_thus <ψ{a}>
2039       using a_def c "rule=E" "→E" by blast
2040     qed
2041     ultimately AOT_have <φ{a} & ∀z(φ{z} → z = a) & ψ{a}> by (rule "&I")
2042     AOT_thus <∃x(φ{x} & ∀z(φ{z} → z = x) & ψ{x})> by (rule "∃I")
2043   next
2044     AOT_assume <∃x(φ{x} & ∀z(φ{z} → z = x) & ψ{x})>
2045     then AOT_obtain b where g: <φ{b} & ∀z(φ{z} → z = b) & ψ{b}>
2046     using "instantiation"[rotated] by blast
2047     AOT_hence h: <b = ιx φ{x} ≡ (φ{b} & ∀z(φ{z} → z = b))>
2048     using b "∀E" by blast
2049     AOT_have <φ{b} & ∀z(φ{z} → z = b)> and j: <ψ{b}>
2050     using g "&E" by blast+
2051     AOT_hence <b = ιx φ{x}> using h "≡E" by blast
2052     AOT_thus <ψ{ιx φ{x}}> using j "rule=E" by blast
2053   qed
2054 qed
2055 end
2056
2057 interpretation "russell-axiom[exe,1]": russell_axiom <λ κ . «[II]κ»>
2058   by standard (metis "cqt:5:a[1]" [axiom_inst, THEN "→E"] "&E"(2))
2059 interpretation "russell-axiom[exe,2,1,1]": russell_axiom <λ κ . «[III]κκ'»>
2060   by standard (metis "cqt:5:a[2]" [axiom_inst, THEN "→E"] "&E")
2061 interpretation "russell-axiom[exe,2,1,2]": russell_axiom <λ κ . «[III]κ'κ»>
2062   by standard (metis "cqt:5:a[2]" [axiom_inst, THEN "→E"] "&E"(2))
2063 interpretation "russell-axiom[exe,2,2]": russell_axiom <λ κ . «[III]κκ»>
2064   by standard (metis "cqt:5:a[2]" [axiom_inst, THEN "→E"] "&E"(2))
2065 interpretation "russell-axiom[exe,3,1,1]": russell_axiom <λ κ . «[III]κκ'κ»>
2066   by standard (metis "cqt:5:a[3]" [axiom_inst, THEN "→E"] "&E")
2067 interpretation "russell-axiom[exe,3,1,2]": russell_axiom <λ κ . «[III]κ'κκ»>
2068   by standard (metis "cqt:5:a[3]" [axiom_inst, THEN "→E"] "&E")
2069 interpretation "russell-axiom[exe,3,1,3]": russell_axiom <λ κ . «[III]κ'κ'κ»>
2070   by standard (metis "cqt:5:a[3]" [axiom_inst, THEN "→E"] "&E"(2))
2071 interpretation "russell-axiom[exe,3,2,1]": russell_axiom <λ κ . «[III]κκκ'»>
2072   by standard (metis "cqt:5:a[3]" [axiom_inst, THEN "→E"] "&E")
2073 interpretation "russell-axiom[exe,3,2,2]": russell_axiom <λ κ . «[III]κκ'κ»>
2074   by standard (metis "cqt:5:a[3]" [axiom_inst, THEN "→E"] "&E"(2))
2075 interpretation "russell-axiom[exe,3,2,3]": russell_axiom <λ κ . «[III]κ'κκ»>
2076   by standard (metis "cqt:5:a[3]" [axiom_inst, THEN "→E"] "&E"(2))
2077 interpretation "russell-axiom[exe,3,3]": russell_axiom <λ κ . «[III]κκκ»>

```

```

2078   by standard (metis "cqt:5:a[3]"[axiom_inst, THEN "→E"] "&E"(2))
2079
2080 interpretation "russell-axiom[enc,1]": russell_axiom <λ κ . «κ [II]»>
2081   by standard (metis "cqt:5:b[1]"[axiom_inst, THEN "→E"] "&E"(2))
2082 interpretation "russell-axiom[enc,2,1]": russell_axiom <λ κ . «κκ' [II]»>
2083   by standard (metis "cqt:5:b[2]"[axiom_inst, THEN "→E"] "&E")
2084 interpretation "russell-axiom[enc,2,2]": russell_axiom <λ κ . «κ'κ [II]»>
2085   by standard (metis "cqt:5:b[2]"[axiom_inst, THEN "→E"] "&E"(2))
2086 interpretation "russell-axiom[enc,2,3]": russell_axiom <λ κ . «κκκ [II]»>
2087   by standard (metis "cqt:5:b[2]"[axiom_inst, THEN "→E"] "&E"(2))
2088 interpretation "russell-axiom[enc,3,1,1]": russell_axiom <λ κ . «κκ'κ" [II]»>
2089   by standard (metis "cqt:5:b[3]"[axiom_inst, THEN "→E"] "&E")
2090 interpretation "russell-axiom[enc,3,1,2]": russell_axiom <λ κ . «κ'κκ" [II]»>
2091   by standard (metis "cqt:5:b[3]"[axiom_inst, THEN "→E"] "&E")
2092 interpretation "russell-axiom[enc,3,1,3]": russell_axiom <λ κ . «κ'κ"κ [II]»>
2093   by standard (metis "cqt:5:b[3]"[axiom_inst, THEN "→E"] "&E"(2))
2094 interpretation "russell-axiom[enc,3,2,1]": russell_axiom <λ κ . «κκκ' [II]»>
2095   by standard (metis "cqt:5:b[3]"[axiom_inst, THEN "→E"] "&E")
2096 interpretation "russell-axiom[enc,3,2,2]": russell_axiom <λ κ . «κκ'κ [II]»>
2097   by standard (metis "cqt:5:b[3]"[axiom_inst, THEN "→E"] "&E"(2))
2098 interpretation "russell-axiom[enc,3,2,3]": russell_axiom <λ κ . «κ'κκ [II]»>
2099   by standard (metis "cqt:5:b[3]"[axiom_inst, THEN "→E"] "&E"(2))
2100 interpretation "russell-axiom[enc,3,3]": russell_axiom <λ κ . «κκκ [II]»>
2101   by standard (metis "cqt:5:b[3]"[axiom_inst, THEN "→E"] "&E"(2))
2102
2103 AOT_act_theorem "!-exists:1": <ιx φ{x}↓ ≡ ∃!x φ{x}> (143.1)
2104 proof(rule "≡I"; rule "→I")
2105   AOT_assume <ιx φ{x}↓>
2106   AOT_hence <∃y (y = ιx φ{x})> by (metis "rule=I:1" "existential:1")
2107   then AOT_obtain a where <a = ιx φ{x}>
2108     using "instantiate1"[rotated] by blast
2109   AOT_hence <φ{a} & ∀z (φ{z} → z = a)>
2110     using hintikka "≡E" by blast
2111   AOT_hence <∃x (φ{x} & ∀z (φ{z} → z = x))>
2112     by (rule "∃I")
2113   AOT_thus <∃!x φ{x}>
2114     using "uniqueness:1"[THEN "≡dfI"] by blast
2115 next
2116   AOT_assume <∃!x φ{x}>
2117   AOT_hence <∃x (φ{x} & ∀z (φ{z} → z = x))>
2118     using "uniqueness:1"[THEN "≡dfE"] by blast
2119   then AOT_obtain b where <φ{b} & ∀z (φ{z} → z = b)>
2120     using "instantiate1"[rotated] by blast
2121   AOT_hence <b = ιx φ{x}>
2122     using hintikka "≡E" by blast
2123   AOT_thus <ιx φ{x}↓>
2124     by (metis "t=t-proper:2" "vdash-properties:6")
2125 qed
2126
2127 AOT_act_theorem "!-exists:2": <∃y(y=ιx φ{x}) ≡ ∃!x φ{x}> (143.2)
2128   using "!-exists:1" "free-thms:1" "≡E"(6) by blast
2129
2130 AOT_act_theorem "y-in:1": <x = ιx φ{x} → φ{x}> (144.1)
2131   using "&E"(1) "→I" hintikka "≡E"(1) by blast
2132
2133 (* Note: generalized alphabetic variant of the last theorem *)
2134 AOT_act_theorem "y-in:2": <z = ιx φ{x} → φ{z}> using "y-in:1". (144.2)
2135
2136 AOT_act_theorem "y-in:3": <ιx φ{x}↓ → φ{ιx φ{x}}> (144.3)
2137 proof(rule "→I")
2138   AOT_assume <ιx φ{x}↓>
2139   AOT_hence <∃y (y = ιx φ{x})>
2140     by (metis "rule=I:1" "existential:1")

```

```

2141   then AOT_obtain a where <a =  $\iota x \varphi\{x\}$ >
2142     using "instantiation"[rotated] by blast
2143   moreover AOT_have < $\varphi\{a\}$ >
2144     using calculation hintikka " $\equiv$ E"(1) "&E" by blast
2145   ultimately AOT_show < $\varphi\{\iota x \varphi\{x\}\}$ > using "rule=E" by blast
2146 qed
2147
2148 AOT_act_theorem "y-in:4": < $\exists y (y = \iota x \varphi\{x\}) \rightarrow \varphi\{\iota x \varphi\{x\}\}$ > (144.4)
2149   using "y-in:3"[THEN " $\rightarrow$ E"] "free-thms:1"[THEN " $\equiv$ E"(2)] " $\rightarrow$ I" by blast
2150
2151
2152 AOT_theorem "act-quant-nec": (145)
2153   < $\forall \beta (\mathcal{A}\varphi\{\beta\} \equiv \beta = \alpha) \equiv \forall \beta (\mathcal{A}\mathcal{A}\varphi\{\beta\} \equiv \beta = \alpha)$ >
2154 proof(rule " $\equiv$ I"; rule " $\rightarrow$ I")
2155   AOT_assume < $\forall \beta (\mathcal{A}\varphi\{\beta\} \equiv \beta = \alpha)$ >
2156   AOT_hence < $\mathcal{A}\varphi\{\beta\} \equiv \beta = \alpha$ > for  $\beta$  using " $\forall$ E" by blast
2157   AOT_hence < $\mathcal{A}\mathcal{A}\varphi\{\beta\} \equiv \beta = \alpha$ > for  $\beta$ 
2158     by (metis "Act-Basic:5" "act-conj-act:4" " $\equiv$ E"(1) " $\equiv$ E"(5))
2159   AOT_thus < $\forall \beta (\mathcal{A}\mathcal{A}\varphi\{\beta\} \equiv \beta = \alpha)$ >
2160     by (rule " $\forall$ I")
2161 next
2162   AOT_assume < $\forall \beta (\mathcal{A}\mathcal{A}\varphi\{\beta\} \equiv \beta = \alpha)$ >
2163   AOT_hence < $\mathcal{A}\mathcal{A}\varphi\{\beta\} \equiv \beta = \alpha$ > for  $\beta$  using " $\forall$ E" by blast
2164   AOT_hence < $\mathcal{A}\varphi\{\beta\} \equiv \beta = \alpha$ > for  $\beta$ 
2165     by (metis "Act-Basic:5" "act-conj-act:4" " $\equiv$ E"(1) " $\equiv$ E"(6))
2166   AOT_thus < $\forall \beta (\mathcal{A}\varphi\{\beta\} \equiv \beta = \alpha)$ >
2167     by (rule " $\forall$ I")
2168 qed
2169
2170 AOT_theorem "equi-desc-descA:1": < $x = \iota x \varphi\{x\} \equiv x = \iota x (\mathcal{A}\varphi\{x\})$ > (146.1)
2171 proof -
2172   AOT_have < $x = \iota x \varphi\{x\} \equiv \forall z (\mathcal{A}\varphi\{z\} \equiv z = x)$ >
2173     using descriptions[axiom_inst] by blast
2174   also AOT_have < $\dots \equiv \forall z (\mathcal{A}\mathcal{A}\varphi\{z\} \equiv z = x)$ >
2175   proof(rule " $\equiv$ I"; rule " $\rightarrow$ I"; rule " $\forall$ I")
2176     AOT_assume < $\forall z (\mathcal{A}\varphi\{z\} \equiv z = x)$ >
2177     AOT_hence < $\mathcal{A}\varphi\{a\} \equiv a = x$ > for a
2178       using " $\forall$ E" by blast
2179     AOT_thus < $\mathcal{A}\mathcal{A}\varphi\{a\} \equiv a = x$ > for a
2180       by (metis "Act-Basic:5" "act-conj-act:4" " $\equiv$ E"(1) " $\equiv$ E"(5))
2181   next
2182     AOT_assume < $\forall z (\mathcal{A}\mathcal{A}\varphi\{z\} \equiv z = x)$ >
2183     AOT_hence < $\mathcal{A}\mathcal{A}\varphi\{a\} \equiv a = x$ > for a
2184       using " $\forall$ E" by blast
2185     AOT_thus < $\mathcal{A}\varphi\{a\} \equiv a = x$ > for a
2186       by (metis "Act-Basic:5" "act-conj-act:4" " $\equiv$ E"(1) " $\equiv$ E"(6))
2187   qed
2188   also AOT_have < $\dots \equiv x = \iota x (\mathcal{A}\varphi\{x\})$ >
2189     using "Commutativity of  $\equiv$ "[THEN " $\equiv$ E"(1)] descriptions[axiom_inst] by fast
2190   finally show ?thesis .
2191 qed
2192
2193 AOT_theorem "equi-desc-descA:2": < $\iota x \varphi\{x\} \downarrow \rightarrow \iota x \varphi\{x\} = \iota x (\mathcal{A}\varphi\{x\})$ > (146.2)
2194 proof(rule " $\rightarrow$ I")
2195   AOT_assume < $\iota x \varphi\{x\} \downarrow$ >
2196   AOT_hence < $\exists y (y = \iota x \varphi\{x\})$ >
2197     by (metis "rule=I:1" "existential:1")
2198   then AOT_obtain a where <a =  $\iota x \varphi\{x\}$ >
2199     using "instantiation"[rotated] by blast
2200   moreover AOT_have <a =  $\iota x (\mathcal{A}\varphi\{x\})$ >
2201     using calculation "equi-desc-descA:1"[THEN " $\equiv$ E"(1)] by blast
2202   ultimately AOT_show < $\iota x \varphi\{x\} = \iota x (\mathcal{A}\varphi\{x\})$ >
2203     using "rule=E" by fast

```

```

2204 qed
2205
2206 AOT_theorem "nec-hintikka-scheme": (147)
2207   <x =  $\iota x \varphi\{x} \equiv \mathcal{A}\varphi\{x}$  &  $\forall z(\mathcal{A}\varphi\{z} \rightarrow z = x)$ >
2208 proof -
2209   AOT_have <x =  $\iota x \varphi\{x} \equiv \forall z(\mathcal{A}\varphi\{z} \equiv z = x)$ >
2210     using descriptions[axiom_inst] by blast
2211   also AOT_have <...  $\equiv (\mathcal{A}\varphi\{x} \text{ & } \forall z(\mathcal{A}\varphi\{z} \rightarrow z = x))$ >
2212     using "Commutativity of  $\equiv$ " [THEN " $\equiv$ "(1)] "term-out:3" by fast
2213   finally show ?thesis.
2214 qed
2215
2216 AOT_theorem "equiv-desc-eq:1": (148.1)
2217   < $\mathcal{A}\forall x(\varphi\{x} \equiv \psi\{x}) \rightarrow \forall x (x = \iota x \varphi\{x} \equiv x = \iota x \psi\{x})$ >
2218 proof(rule " $\rightarrow$ I"; rule " $\forall$ I")
2219   fix  $\beta$ 
2220   AOT_assume < $\mathcal{A}\forall x(\varphi\{x} \equiv \psi\{x})$ >
2221   AOT_hence < $\mathcal{A}(\varphi\{x} \equiv \psi\{x})$ > for x
2222     using "logic-actual-nec:3"[axiom_inst, THEN " $\equiv$ "(1)] " $\forall$ E"(2) by blast
2223   AOT_hence 0: < $\mathcal{A}\varphi\{x} \equiv \mathcal{A}\psi\{x}$ > for x
2224     by (metis "Act-Basic:5" " $\equiv$ "(1))
2225   AOT_have < $\beta = \iota x \varphi\{x} \equiv \mathcal{A}\varphi\{\beta} \text{ & } \forall z(\mathcal{A}\varphi\{z} \rightarrow z = \beta)$ >
2226     using "nec-hintikka-scheme" by blast
2227   also AOT_have <...  $\equiv \mathcal{A}\psi\{\beta} \text{ & } \forall z(\mathcal{A}\psi\{z} \rightarrow z = \beta)$ >
2228 proof (rule " $\equiv$ I"; rule " $\rightarrow$ I")
2229   AOT_assume 1: < $\mathcal{A}\varphi\{\beta} \text{ & } \forall z(\mathcal{A}\varphi\{z} \rightarrow z = \beta)$ >
2230   AOT_hence < $\mathcal{A}\varphi\{z} \rightarrow z = \beta$ > for z
2231     using "&E" " $\forall$ E" by blast
2232   AOT_hence < $\mathcal{A}\psi\{z} \rightarrow z = \beta$ > for z
2233     using 0 " $\equiv$ E" " $\rightarrow$ I" " $\rightarrow$ E" by metis
2234   AOT_hence < $\forall z(\mathcal{A}\psi\{z} \rightarrow z = \beta)$ >
2235     using " $\forall$ I" by fast
2236   moreover AOT_have < $\mathcal{A}\psi\{\beta}$ >
2237     using "&E" 0 [THEN " $\equiv$ "(1)] 1 by blast
2238   ultimately AOT_show < $\mathcal{A}\psi\{\beta} \text{ & } \forall z(\mathcal{A}\psi\{z} \rightarrow z = \beta)$ >
2239     using "&I" by blast
2240 next
2241   AOT_assume 1: < $\mathcal{A}\psi\{\beta} \text{ & } \forall z(\mathcal{A}\psi\{z} \rightarrow z = \beta)$ >
2242   AOT_hence < $\mathcal{A}\psi\{z} \rightarrow z = \beta$ > for z
2243     using "&E" " $\forall$ E" by blast
2244   AOT_hence < $\mathcal{A}\varphi\{z} \rightarrow z = \beta$ > for z
2245     using 0 " $\equiv$ E" " $\rightarrow$ I" " $\rightarrow$ E" by metis
2246   AOT_hence < $\forall z(\mathcal{A}\varphi\{z} \rightarrow z = \beta)$ >
2247     using " $\forall$ I" by fast
2248   moreover AOT_have < $\mathcal{A}\varphi\{\beta}$ >
2249     using "&E" 0 [THEN " $\equiv$ "(2)] 1 by blast
2250   ultimately AOT_show < $\mathcal{A}\varphi\{\beta} \text{ & } \forall z(\mathcal{A}\varphi\{z} \rightarrow z = \beta)$ >
2251     using "&I" by blast
2252   qed
2253   also AOT_have <...  $\equiv \beta = \iota x \psi\{x}$ >
2254     using "Commutativity of  $\equiv$ " [THEN " $\equiv$ "(1)] "nec-hintikka-scheme" by blast
2255   finally AOT_show < $\beta = \iota x \varphi\{x} \equiv \beta = \iota x \psi\{x}$ > .
2256 qed
2257
2258 AOT_theorem "equiv-desc-eq:2": (148.2)
2259   < $\iota x \varphi\{x} \downarrow \text{ & } \mathcal{A}\forall x(\varphi\{x} \equiv \psi\{x}) \rightarrow \iota x \varphi\{x} = \iota x \psi\{x}$ >
2260 proof (rule " $\rightarrow$ I")
2261   AOT_assume < $\iota x \varphi\{x} \downarrow \text{ & } \mathcal{A}\forall x(\varphi\{x} \equiv \psi\{x})$ >
2262   AOT_hence 0: < $\exists y (y = \iota x \varphi\{x})$ > and
2263     1: < $\forall x (x = \iota x \varphi\{x} \equiv x = \iota x \psi\{x})$ >
2264     using "&E" "free-thms:1" [THEN " $\equiv$ "(1)] "equiv-desc-eq:1" " $\rightarrow$ E" by blast+
2265   then AOT_obtain a where <a =  $\iota x \varphi\{x}$ >
2266     using "instantiation"[rotated] by blast

```

```

2267   moreover AOT_have <a =  $\iota x \psi\{x\}$ >
2268     using calculation 1 " $\forall E$ " " $\equiv E$ "(1) by fast
2269   ultimately AOT_show < $\iota x \varphi\{x\} = \iota x \psi\{x\}$ >
2270     using "rule=E" by fast
2271 qed
2272
2273 AOT_theorem "equiv-desc-eq:3": (148.3)
2274   < $\iota x \varphi\{x\} \downarrow$  &  $\Box \forall x (\varphi\{x\} \equiv \psi\{x\}) \rightarrow \iota x \varphi\{x\} = \iota x \psi\{x\}$ >
2275   using " $\rightarrow I$ " "equiv-desc-eq:2"[THEN " $\rightarrow E$ ", OF "&I"] "&E"
2276     "nec-imp-act"[THEN " $\rightarrow E$ "] by metis
2277
2278 (* Note: this is a special case of "exist-nec" *)
2279 AOT_theorem "equiv-desc-eq:4": < $\iota x \varphi\{x\} \downarrow \rightarrow \Box \iota x \varphi\{x\} \downarrow$ > (148.4)
2280 proof(rule " $\rightarrow I$ ")
2281   AOT_assume < $\iota x \varphi\{x\} \downarrow$ >
2282   AOT_hence < $\exists y (y = \iota x \varphi\{x\})$ >
2283     by (metis "rule=I:1" "existential:1")
2284   then AOT_obtain a where <a =  $\iota x \varphi\{x\}$ >
2285     using "instantiation"[rotated] by blast
2286   AOT_thus < $\Box \iota x \varphi\{x\} \downarrow$ >
2287     using "ex:2:a" "rule=E" by fast
2288 qed
2289
2290 AOT_theorem "equiv-desc-eq:5": < $\iota x \varphi\{x\} \downarrow \rightarrow \exists y \Box (y = \iota x \varphi\{x\})$ > (148.5)
2291 proof(rule " $\rightarrow I$ ")
2292   AOT_assume < $\iota x \varphi\{x\} \downarrow$ >
2293   AOT_hence < $\exists y (y = \iota x \varphi\{x\})$ >
2294     by (metis "rule=I:1" "existential:1")
2295   then AOT_obtain a where <a =  $\iota x \varphi\{x\}$ >
2296     using "instantiation"[rotated] by blast
2297   AOT_hence < $\Box (a = \iota x \varphi\{x\})$ >
2298     by (metis "id-nec:2" "vdash-properties:10")
2299   AOT_thus < $\exists y \Box (y = \iota x \varphi\{x\})$ >
2300     by (rule " $\exists I$ ")
2301 qed
2302
2303 AOT_act_theorem "equiv-desc-eq2:1": (149.1)
2304   < $\forall x (\varphi\{x\} \equiv \psi\{x\}) \rightarrow \forall x (x = \iota x \varphi\{x\} \equiv x = \iota x \psi\{x\})$ >
2305   using " $\rightarrow I$ " "logic-actual"[act_axiom_inst, THEN " $\rightarrow E$ "]
2306     "equiv-desc-eq:1"[THEN " $\rightarrow E$ "]
2307     "RA[1]" "deduction-theorem" by blast
2308
2309 AOT_act_theorem "equiv-desc-eq2:2": (149.2)
2310   < $\iota x \varphi\{x\} \downarrow$  &  $\forall x (\varphi\{x\} \equiv \psi\{x\}) \rightarrow \iota x \varphi\{x\} = \iota x \psi\{x\}$ >
2311   using " $\rightarrow I$ " "logic-actual"[act_axiom_inst, THEN " $\rightarrow E$ "]
2312     "equiv-desc-eq:2"[THEN " $\rightarrow E$ ", OF "&I"]
2313     "RA[1]" "deduction-theorem" "&E" by metis
2314
2315 context russell_axiom
2316 begin
2317 AOT_theorem "nec-russell-axiom": (150)
2318   < $\psi\{\iota x \varphi\{x\}\} \equiv \exists x (\mathcal{A}\varphi\{x\} \ \& \ \forall z (\mathcal{A}\varphi\{z\} \rightarrow z = x) \ \& \ \psi\{x\})$ >
2319 proof -
2320   AOT_have b: < $\forall x (x = \iota x \varphi\{x\} \equiv (\mathcal{A}\varphi\{x\} \ \& \ \forall z (\mathcal{A}\varphi\{z\} \rightarrow z = x)))$ >
2321     using "nec-hintikka-scheme" " $\forall I$ " by fast
2322   show ?thesis
2323 proof(rule " $\equiv I$ "; rule " $\rightarrow I$ ")
2324   AOT_assume c: < $\psi\{\iota x \varphi\{x\}\}$ >
2325   AOT_hence d: < $\iota x \varphi\{x\} \downarrow$ >
2326     using  $\psi$ _denotes_asm by blast
2327   AOT_hence < $\exists y (y = \iota x \varphi\{x\})$ >
2328     by (metis "rule=I:1" "existential:1")
2329   then AOT_obtain a where a_def: <a =  $\iota x \varphi\{x\}$ >

```



```

2330     using "instantiation"[rotated] by blast
2331 moreover AOT_have <a =  $\iota x \varphi\{x\} \equiv (\mathcal{A}\varphi\{a\} \ \& \ \forall z(\mathcal{A}\varphi\{z\} \rightarrow z = a))$ >
2332     using b "VE" by blast
2333 ultimately AOT_have < $\mathcal{A}\varphi\{a\} \ \& \ \forall z(\mathcal{A}\varphi\{z\} \rightarrow z = a)$ >
2334     using "E" by blast
2335 moreover AOT_have < $\psi\{a\}$ >
2336 proof -
2337   AOT_have 1: < $\forall x \forall y(x = y \rightarrow y = x)$ >
2338     by (simp add: "id-eq:2" "universal-cor")
2339   AOT_have <a =  $\iota x \varphi\{x\} \rightarrow \iota x \varphi\{x\} = a$ >
2340     by (rule "VE"(1)[where  $\tau = \langle \iota x \varphi\{x\} \rangle$ "]; rule "VE"(2)[where  $\beta = a$ ])
2341     (auto simp: d "universal-cor" 1)
2342   AOT_thus < $\psi\{a\}$ >
2343     using a_def c "rule=E" " $\rightarrow$ E" by metis
2344 qed
2345 ultimately AOT_have < $\mathcal{A}\varphi\{a\} \ \& \ \forall z(\mathcal{A}\varphi\{z\} \rightarrow z = a) \ \& \ \psi\{a\}$ >
2346     by (rule "I")
2347 AOT_thus < $\exists x(\mathcal{A}\varphi\{x\} \ \& \ \forall z(\mathcal{A}\varphi\{z\} \rightarrow z = x) \ \& \ \psi\{x\})$ >
2348     by (rule "EI")
2349 next
2350 AOT_assume < $\exists x(\mathcal{A}\varphi\{x\} \ \& \ \forall z(\mathcal{A}\varphi\{z\} \rightarrow z = x) \ \& \ \psi\{x\})$ >
2351 then AOT_obtain b where g: < $\mathcal{A}\varphi\{b\} \ \& \ \forall z(\mathcal{A}\varphi\{z\} \rightarrow z = b) \ \& \ \psi\{b\}$ >
2352     using "instantiation"[rotated] by blast
2353 AOT_hence h: <b =  $\iota x \varphi\{x\} \equiv (\mathcal{A}\varphi\{b\} \ \& \ \forall z(\mathcal{A}\varphi\{z\} \rightarrow z = b))$ >
2354     using b "VE" by blast
2355 AOT_have < $\mathcal{A}\varphi\{b\} \ \& \ \forall z(\mathcal{A}\varphi\{z\} \rightarrow z = b)$ > and j: < $\psi\{b\}$ >
2356     using g "&E" by blast+
2357 AOT_hence <b =  $\iota x \varphi\{x\}$ >
2358     using h "E" by blast
2359 AOT_thus < $\psi\{\iota x \varphi\{x\}\}$ >
2360     using j "rule=E" by blast
2361 qed
2362 qed
2363 end
2364
2365 AOT_theorem "actual-desc:1": < $\iota x \varphi\{x\} \downarrow \equiv \exists! x \mathcal{A}\varphi\{x\}$ > (151.1)
2366 proof (rule "EI"; rule " $\rightarrow$ I")
2367   AOT_assume < $\iota x \varphi\{x\} \downarrow$ >
2368   AOT_hence < $\exists y (y = \iota x \varphi\{x\})$ >
2369     by (metis "rule=I:1" "existential:1")
2370   then AOT_obtain a where <a =  $\iota x \varphi\{x\}$ >
2371     using "instantiation"[rotated] by blast
2372   moreover AOT_have <a =  $\iota x \varphi\{x\} \equiv \forall z(\mathcal{A}\varphi\{z\} \equiv z = a)$ >
2373     using descriptions[axiom_inst] by blast
2374   ultimately AOT_have < $\forall z(\mathcal{A}\varphi\{z\} \equiv z = a)$ >
2375     using "E" by blast
2376   AOT_hence < $\exists x \forall z(\mathcal{A}\varphi\{z\} \equiv z = x)$ > by (rule "EI")
2377   AOT_thus < $\exists! x \mathcal{A}\varphi\{x\}$ >
2378     using "uniqueness:2"[THEN "E"(2)] by fast
2379 next
2380 AOT_assume < $\exists! x \mathcal{A}\varphi\{x\}$ >
2381 AOT_hence < $\exists x \forall z(\mathcal{A}\varphi\{z\} \equiv z = x)$ >
2382     using "uniqueness:2"[THEN "E"(1)] by fast
2383 then AOT_obtain a where < $\forall z(\mathcal{A}\varphi\{z\} \equiv z = a)$ >
2384     using "instantiation"[rotated] by blast
2385 moreover AOT_have <a =  $\iota x \varphi\{x\} \equiv \forall z(\mathcal{A}\varphi\{z\} \equiv z = a)$ >
2386     using descriptions[axiom_inst] by blast
2387 ultimately AOT_have <a =  $\iota x \varphi\{x\}$ >
2388     using "E" by blast
2389 AOT_thus < $\iota x \varphi\{x\} \downarrow$ >
2390     by (metis "t=t-proper:2" "vdash-properties:6")
2391 qed
2392

```



```

2393 AOT_theorem "actual-desc:2": <x =  $\iota x \varphi\{x\} \rightarrow \mathcal{A}\varphi\{x\}$ > (151.2)
2394   using "&E"(1) "contraposition:1[2]" "≡E"(1) "nec-hintikka-scheme"
2395     "reductio-aa:2" "vdash-properties:9" by blast
2396
2397 (* Note: generalized alphabetic variant of the last theorem *)
2398 AOT_theorem "actual-desc:3": <z =  $\iota x \varphi\{x\} \rightarrow \mathcal{A}\varphi\{z\}$ > (151.3)
2399   using "actual-desc:2".
2400
2401 AOT_theorem "actual-desc:4": < $\iota x \varphi\{x\} \downarrow \rightarrow \mathcal{A}\varphi\{\iota x \varphi\{x\}\}$ > (151.4)
2402 proof(rule "→I")
2403   AOT_assume < $\iota x \varphi\{x\} \downarrow$ >
2404   AOT_hence < $\exists y (y = \iota x \varphi\{x\})$ > by (metis "rule=I:1" "existential:1")
2405   then AOT_obtain a where <a =  $\iota x \varphi\{x\}$ > using "instantiation"[rotated] by blast
2406   AOT_thus < $\mathcal{A}\varphi\{\iota x \varphi\{x\}\}$ >
2407     using "actual-desc:2" "rule=E" "→E" by fast
2408 qed
2409
2410 AOT_theorem "actual-desc:5": < $\iota x \varphi\{x\} = \iota x \psi\{x\} \rightarrow \mathcal{A}\forall x(\varphi\{x\} \equiv \psi\{x\})$ > (151.5)
2411 proof(rule "→I")
2412   AOT_assume 0: < $\iota x \varphi\{x\} = \iota x \psi\{x\}$ >
2413   AOT_hence  $\varphi\_down$ : < $\iota x \varphi\{x\} \downarrow$ > and  $\psi\_down$ : < $\iota x \psi\{x\} \downarrow$ >
2414     using "t=t-proper:1" "t=t-proper:2" "vdash-properties:6" by blast+
2415   AOT_hence < $\exists y (y = \iota x \varphi\{x\})$ > and < $\exists y (y = \iota x \psi\{x\})$ >
2416     by (metis "rule=I:1" "existential:1")+
2417   then AOT_obtain a and b where a_eq: <a =  $\iota x \varphi\{x\}$ > and b_eq: <b =  $\iota x \psi\{x\}$ >
2418     using "instantiation"[rotated] by metis
2419
2420   AOT_have < $\forall \alpha \forall \beta (\alpha = \beta \rightarrow \beta = \alpha)$ >
2421     by (rule "∀I"; rule "∀I"; rule "id-eq:2")
2422   AOT_hence < $\forall \beta (\iota x \varphi\{x\} = \beta \rightarrow \beta = \iota x \varphi\{x\})$ >
2423     using "∀E"  $\varphi\_down$  by blast
2424   AOT_hence < $\iota x \varphi\{x\} = \iota x \psi\{x\} \rightarrow \iota x \psi\{x\} = \iota x \varphi\{x\}$ >
2425     using "∀E"  $\psi\_down$  by blast
2426   AOT_hence 1: < $\iota x \psi\{x\} = \iota x \varphi\{x\}$ > using 0
2427     "→E" by blast
2428
2429   AOT_have < $\mathcal{A}\varphi\{x\} \equiv \mathcal{A}\psi\{x\}$ > for x
2430   proof(rule "≡I"; rule "→I")
2431     AOT_assume < $\mathcal{A}\varphi\{x\}$ >
2432     moreover AOT_have < $\mathcal{A}\varphi\{x\} \rightarrow x = a$ > for x
2433       using "nec-hintikka-scheme"[THEN "≡E"(1), OF a_eq, THEN "&E"(2)]
2434       "∀E" by blast
2435     ultimately AOT_have <x = a>
2436       using "→E" by blast
2437     AOT_hence <x =  $\iota x \varphi\{x\}$ >
2438       using a_eq "rule=E" by blast
2439     AOT_hence <x =  $\iota x \psi\{x\}$ >
2440       using 0 "rule=E" by blast
2441     AOT_thus < $\mathcal{A}\psi\{x\}$ >
2442       by (metis "actual-desc:3" "vdash-properties:6")
2443   next
2444     AOT_assume < $\mathcal{A}\psi\{x\}$ >
2445     moreover AOT_have < $\mathcal{A}\psi\{x\} \rightarrow x = b$ > for x
2446       using "nec-hintikka-scheme"[THEN "≡E"(1), OF b_eq, THEN "&E"(2)]
2447       "∀E" by blast
2448     ultimately AOT_have <x = b>
2449       using "→E" by blast
2450     AOT_hence <x =  $\iota x \psi\{x\}$ >
2451       using b_eq "rule=E" by blast
2452     AOT_hence <x =  $\iota x \varphi\{x\}$ >
2453       using 1 "rule=E" by blast
2454     AOT_thus < $\mathcal{A}\varphi\{x\}$ >
2455       by (metis "actual-desc:3" "vdash-properties:6")

```

```

2456 qed
2457 AOT_hence <mathcal{A}(\varphi\{x\} \equiv \psi\{x\})> for x
2458   by (metis "Act-Basic:5" "≡E"(2))
2459 AOT_hence <mathcal{A}(\forall x \mathcal{A}(\varphi\{x\} \equiv \psi\{x\}))>
2460   by (rule "∀I")
2461 AOT_thus <mathcal{A}\forall x (\varphi\{x\} \equiv \psi\{x\})>
2462   using "logic-actual-nec:3"[axiom_inst, THEN "≡E"(2)] by fast
2463 qed
2464
2465 AOT_theorem "!box-desc:1": <math>\langle \exists!x \Box\varphi\{x\} \rightarrow \forall y (y = \iota x \varphi\{x\} \rightarrow \varphi\{y\}) \rangle</math> (152.1)
2466 proof(rule "→I")
2467   AOT_assume <math>\langle \exists!x \Box\varphi\{x\} \rangle</math>
2468   AOT_hence ζ: <math>\langle \exists x (\Box\varphi\{x\} \ \& \ \forall z (\Box\varphi\{z\} \rightarrow z = x)) \rangle</math>
2469     using "uniqueness:1"[THEN "≡dfE"] by blast
2470   then AOT_obtain b where ϑ: <math>\langle \Box\varphi\{b\} \ \& \ \forall z (\Box\varphi\{z\} \rightarrow z = b) \rangle</math>
2471     using "instantiation"[rotated] by blast
2472   AOT_show <math>\langle \forall y (y = \iota x \varphi\{x\} \rightarrow \varphi\{y\}) \rangle</math>
2473   proof(rule GEN; rule "→I")
2474     fix y
2475     AOT_assume <math>\langle y = \iota x \varphi\{x\} \rangle</math>
2476     AOT_hence <math>\langle \mathcal{A}\varphi\{y\} \ \& \ \forall z (\mathcal{A}\varphi\{z\} \rightarrow z = y) \rangle</math>
2477       using "nec-hintikka-scheme"[THEN "≡E"(1)] by blast
2478     AOT_hence <math>\langle \mathcal{A}\varphi\{b\} \rightarrow b = y \rangle</math>
2479       using "&E" "∀E" by blast
2480     moreover AOT_have <math>\langle \mathcal{A}\varphi\{b\} \rangle</math>
2481       using ϑ[THEN "&E"(1)] by (metis "nec-imp-act" "→E")
2482     ultimately AOT_have <math>\langle b = y \rangle</math>
2483       using "→E" by blast
2484     moreover AOT_have <math>\langle \varphi\{b\} \rangle</math>
2485       using ϑ[THEN "&E"(1)] by (metis "qml:2"[axiom_inst] "→E")
2486     ultimately AOT_show <math>\langle \varphi\{y\} \rangle</math>
2487       using "rule=E" by blast
2488   qed
2489 qed
2490
2491 AOT_theorem "!box-desc:2": (152.2)
2492   <math>\langle \forall x (\varphi\{x\} \rightarrow \Box\varphi\{x\}) \rightarrow (\exists!x \varphi\{x\} \rightarrow \forall y (y = \iota x \varphi\{x\} \rightarrow \varphi\{y\})) \rangle</math>
2493 proof(rule "→I"; rule "→I")
2494   AOT_assume <math>\langle \forall x (\varphi\{x\} \rightarrow \Box\varphi\{x\}) \rangle</math>
2495   moreover AOT_assume <math>\langle \exists!x \varphi\{x\} \rangle</math>
2496   ultimately AOT_have <math>\langle \exists!x \Box\varphi\{x\} \rangle</math>
2497     using "nec-exist-!"[THEN "→E", THEN "→E"] by blast
2498   AOT_thus <math>\langle \forall y (y = \iota x \varphi\{x\} \rightarrow \varphi\{y\}) \rangle</math>
2499     using "!box-desc:1" "→E" by blast
2500 qed
2501
2502 (* Note: vacuous in the embedding. *)
2503 AOT_theorem "dr-alphabetic-thm": <math>\langle \iota\nu \varphi\{\nu\} \downarrow \rightarrow \iota\nu \varphi\{\nu\} = \iota\mu \varphi\{\mu\} \rangle</math> (153)
2504   by (simp add: "rule=I:1" "→I")
2505
2506 subsection<The Theory of Necessity>
2507 text<\label{PLM: 9.9}>
2508
2509 AOT_theorem "RM:1[prem]": (156.1)
2510   assumes <math>\langle \Gamma \vdash_{\Box} \varphi \rightarrow \psi \rangle</math>
2511   shows <math>\langle \Box\Gamma \vdash_{\Box} \Box\varphi \rightarrow \Box\psi \rangle</math>
2512 proof -
2513   AOT_have <math>\langle \Box\Gamma \vdash_{\Box} \Box(\varphi \rightarrow \psi) \rangle</math>
2514     using "RN[prem]" assms by blast
2515   AOT_thus <math>\langle \Box\Gamma \vdash_{\Box} \Box\varphi \rightarrow \Box\psi \rangle</math>
2516     by (metis "qml:1"[axiom_inst] "→E")
2517 qed
2518

```

```

2519 AOT_theorem "RM:1": (156.1)
2520   assumes <math>\vdash_{\Box} \varphi \rightarrow \psi</math>
2521   shows <math>\vdash_{\Box} \Box\varphi \rightarrow \Box\psi</math>
2522   using "RM:1[pre]" assms by blast
2523
2524 lemmas RM = "RM:1" (156)
2525
2526 AOT_theorem "RM:2[pre]": (156.2)
2527   assumes <math>\Gamma \vdash_{\Box} \varphi \rightarrow \psi</math>
2528   shows <math>\langle \Box\Gamma \vdash_{\Box} \Diamond\varphi \rightarrow \Diamond\psi \rangle</math>
2529 proof -
2530   AOT_have <math>\langle \Gamma \vdash_{\Box} \neg\psi \rightarrow \neg\varphi \rangle</math>
2531     using assms
2532     by (simp add: "contraposition:1[1]")
2533   AOT_hence <math>\langle \Box\Gamma \vdash_{\Box} \Box\neg\psi \rightarrow \Box\neg\varphi \rangle</math>
2534     using "RM:1[pre]" by blast
2535   AOT_thus <math>\langle \Box\Gamma \vdash_{\Box} \Diamond\varphi \rightarrow \Diamond\psi \rangle</math>
2536     by (meson "≡dfE" "≡dfI" "conventions:5" "→I" "modus-tollens:1")
2537 qed
2538
2539 AOT_theorem "RM:2": (156.2)
2540   assumes <math>\vdash_{\Box} \varphi \rightarrow \psi</math>
2541   shows <math>\vdash_{\Box} \Diamond\varphi \rightarrow \Diamond\psi</math>
2542   using "RM:2[pre]" assms by blast
2543
2544 lemmas "RM $\Diamond$ " = "RM:2"
2545
2546 AOT_theorem "RM:3[pre]": (156.3)
2547   assumes <math>\Gamma \vdash_{\Box} \varphi \equiv \psi</math>
2548   shows <math>\langle \Box\Gamma \vdash_{\Box} \Box\varphi \equiv \Box\psi \rangle</math>
2549 proof -
2550   AOT_have <math>\langle \Gamma \vdash_{\Box} \varphi \rightarrow \psi \rangle</math> and <math>\langle \Gamma \vdash_{\Box} \psi \rightarrow \varphi \rangle</math>
2551     using assms "≡E" "→I" by metis+
2552   AOT_hence <math>\langle \Box\Gamma \vdash_{\Box} \Box\varphi \rightarrow \Box\psi \rangle</math> and <math>\langle \Box\Gamma \vdash_{\Box} \Box\psi \rightarrow \Box\varphi \rangle</math>
2553     using "RM:1[pre]" by metis+
2554   AOT_thus <math>\langle \Box\Gamma \vdash_{\Box} \Box\varphi \equiv \Box\psi \rangle</math>
2555     by (simp add: "≡I")
2556 qed
2557
2558 AOT_theorem "RM:3": (156.3)
2559   assumes <math>\vdash_{\Box} \varphi \equiv \psi</math>
2560   shows <math>\vdash_{\Box} \Box\varphi \equiv \Box\psi</math>
2561   using "RM:3[pre]" assms by blast
2562
2563 lemmas RE = "RM:3"
2564
2565 AOT_theorem "RM:4[pre]": (156.4)
2566   assumes <math>\Gamma \vdash_{\Box} \varphi \equiv \psi</math>
2567   shows <math>\langle \Box\Gamma \vdash_{\Box} \Diamond\varphi \equiv \Diamond\psi \rangle</math>
2568 proof -
2569   AOT_have <math>\langle \Gamma \vdash_{\Box} \varphi \rightarrow \psi \rangle</math> and <math>\langle \Gamma \vdash_{\Box} \psi \rightarrow \varphi \rangle</math>
2570     using assms "≡E" "→I" by metis+
2571   AOT_hence <math>\langle \Box\Gamma \vdash_{\Box} \Diamond\varphi \rightarrow \Diamond\psi \rangle</math> and <math>\langle \Box\Gamma \vdash_{\Box} \Diamond\psi \rightarrow \Diamond\varphi \rangle</math>
2572     using "RM:2[pre]" by metis+
2573   AOT_thus <math>\langle \Box\Gamma \vdash_{\Box} \Diamond\varphi \equiv \Diamond\psi \rangle</math>
2574     by (simp add: "≡I")
2575 qed
2576
2577 AOT_theorem "RM:4": (156.4)
2578   assumes <math>\vdash_{\Box} \varphi \equiv \psi</math>
2579   shows <math>\vdash_{\Box} \Diamond\varphi \equiv \Diamond\psi</math>
2580   using "RM:4[pre]" assms by blast
2581

```

```

2582 lemmas "RE◇" = "RM:4"
2583
2584 AOT_theorem "KBasic:1": <□φ → □(ψ → φ)> (157.1)
2585   by (simp add: RM "pl:1"[axiom_inst])
2586
2587 AOT_theorem "KBasic:2": <□¬φ → □(φ → ψ)> (157.2)
2588   by (simp add: RM "useful-tautologies:3")
2589
2590 AOT_theorem "KBasic:3": <□(φ & ψ) ≡ (□φ & □ψ)> (157.3)
2591 proof (rule "≡I"; rule "→I")
2592   AOT_assume <□(φ & ψ)>
2593   AOT_thus <□φ & □ψ>
2594     by (meson RM "&I" "Conjunction Simplification"(1, 2) "→E")
2595 next
2596   AOT_have <□φ → □(ψ → (φ & ψ))>
2597     by (simp add: "RM:1" Adjunction)
2598   AOT_hence <□φ → (□ψ → □(φ & ψ))>
2599     by (metis "Hypothetical Syllogism" "qml:1"[axiom_inst])
2600   moreover AOT_assume <□φ & □ψ>
2601   ultimately AOT_show <□(φ & ψ)>
2602     using "→E" "&E" by blast
2603 qed
2604
2605 AOT_theorem "KBasic:4": <□(φ ≡ ψ) ≡ (□(φ → ψ) & □(ψ → φ))> (157.4)
2606 proof -
2607   AOT_have ϑ: <□((φ → ψ) & (ψ → φ)) ≡ (□(φ → ψ) & □(ψ → φ))>
2608     by (fact "KBasic:3")
2609   AOT_modally_strict {
2610     AOT_have <(φ ≡ ψ) ≡ ((φ → ψ) & (ψ → φ))>
2611       by (fact "conventions:3"[THEN "≡Df"])
2612   }
2613   AOT_hence ξ: <□(φ ≡ ψ) ≡ □((φ → ψ) & (ψ → φ))>
2614     by (rule RE)
2615   with ξ and ϑ AOT_show <□(φ ≡ ψ) ≡ (□(φ → ψ) & □(ψ → φ))>
2616     using "≡E"(5) by blast
2617 qed
2618
2619 AOT_theorem "KBasic:5": <(□(φ → ψ) & □(ψ → φ)) → (□φ ≡ □ψ)> (157.5)
2620 proof -
2621   AOT_have <□(φ → ψ) → (□φ → □ψ)>
2622     by (fact "qml:1"[axiom_inst])
2623   moreover AOT_have <□(ψ → φ) → (□ψ → □φ)>
2624     by (fact "qml:1"[axiom_inst])
2625   ultimately AOT_have <(□(φ → ψ) & □(ψ → φ)) → ((□φ → □ψ) & (□ψ → □φ))>
2626     by (metis "&I" MP "Double Composition")
2627   moreover AOT_have <((□φ → □ψ) & (□ψ → □φ)) → (□φ ≡ □ψ)>
2628     using "conventions:3"[THEN "≡dfI"] "→I" by blast
2629   ultimately AOT_show <(□(φ → ψ) & □(ψ → φ)) → (□φ ≡ □ψ)>
2630     by (metis "Hypothetical Syllogism")
2631 qed
2632
2633 AOT_theorem "KBasic:6": <□(φ ≡ ψ) → (□φ ≡ □ψ)> (157.6)
2634   using "KBasic:4" "KBasic:5" "deduction-theorem" "≡E"(1) "→E" by blast
2635 AOT_theorem "KBasic:7": <((□φ & □ψ) ∨ (□¬φ & □¬ψ)) → □(φ ≡ ψ)> (157.7)
2636 proof (rule "→I"; drule "√E"(1); (rule "→I")?)
2637   AOT_assume <□φ & □ψ>
2638   AOT_hence <□φ> and <□ψ> using "&E" by blast+
2639   AOT_hence <□(φ → ψ)> and <□(ψ → φ)> using "KBasic:1" "→E" by blast+
2640   AOT_hence <□(φ → ψ) & □(ψ → φ)> using "&I" by blast
2641   AOT_thus <□(φ ≡ ψ)> by (metis "KBasic:4" "≡E"(2))
2642 next
2643   AOT_assume <□¬φ & □¬ψ>
2644   AOT_hence 0: <□(¬φ & ¬ψ)> using "KBasic:3"[THEN "≡E"(2)] by blast

```

```

2645   AOT_modally_strict {
2646     AOT_have <( $\neg\varphi \ \& \ \neg\psi$ )  $\rightarrow$  ( $\varphi \equiv \psi$ )>
2647     by (metis "&E"(1) "&E"(2) "deduction-theorem" " $\equiv$ I" "reductio-aa:1")
2648   }
2649   AOT_hence < $\Box(\neg\varphi \ \& \ \neg\psi) \rightarrow \Box(\varphi \equiv \psi)$ >
2650   by (rule RM)
2651   AOT_thus < $\Box(\varphi \equiv \psi)$ > using 0 " $\rightarrow$ E" by blast
2652 qed(auto)
2653
2654 AOT_theorem "KBasic:8": < $\Box(\varphi \ \& \ \psi) \rightarrow \Box(\varphi \equiv \psi)$ > (157.8)
2655 by (meson "RM:1" "&E"(1) "&E"(2) "deduction-theorem" " $\equiv$ I")
2656 AOT_theorem "KBasic:9": < $\Box(\neg\varphi \ \& \ \neg\psi) \rightarrow \Box(\varphi \equiv \psi)$ > (157.9)
2657 by (metis "RM:1" "&E"(1) "&E"(2) "deduction-theorem" " $\equiv$ I" "raa-cor:4")
2658 AOT_theorem "KBasic:10": < $\Box\varphi \equiv \Box\neg\neg\varphi$ > (157.10)
2659 by (simp add: "RM:3" "oth-class-taut:3:b")
2660 AOT_theorem "KBasic:11": < $\neg\Box\varphi \equiv \Diamond\neg\varphi$ > (157.11)
2661 proof (rule " $\equiv$ I"; rule " $\rightarrow$ I")
2662   AOT_show < $\Diamond\neg\varphi$ > if < $\neg\Box\varphi$ >
2663   using that " $\equiv$ dfI" "conventions:5" "KBasic:10" " $\equiv$ E"(3) by blast
2664 next
2665   AOT_show < $\neg\Box\varphi$ > if < $\Diamond\neg\varphi$ >
2666   using " $\equiv$ dfE" "conventions:5" "KBasic:10" " $\equiv$ E"(4) that by blast
2667 qed
2668 AOT_theorem "KBasic:12": < $\Box\varphi \equiv \neg\Diamond\neg\varphi$ > (157.12)
2669 proof (rule " $\equiv$ I"; rule " $\rightarrow$ I")
2670   AOT_show < $\neg\Diamond\neg\varphi$ > if < $\Box\varphi$ >
2671   using " $\neg$ I" "KBasic:11" " $\equiv$ E"(3) that by blast
2672 next
2673   AOT_show < $\Box\varphi$ > if < $\neg\Diamond\neg\varphi$ >
2674   using "KBasic:11" " $\equiv$ E"(1) "reductio-aa:1" that by blast
2675 qed
2676 AOT_theorem "KBasic:13": < $\Box(\varphi \rightarrow \psi) \rightarrow (\Diamond\varphi \rightarrow \Diamond\psi)$ > (157.13)
2677 proof -
2678   AOT_have < $\varphi \rightarrow \psi \vdash_{\Box} \varphi \rightarrow \psi$ > by blast
2679   AOT_hence < $\Box(\varphi \rightarrow \psi) \vdash_{\Box} \Diamond\varphi \rightarrow \Diamond\psi$ >
2680   using "RM:2[prem]" by blast
2681   AOT_thus < $\Box(\varphi \rightarrow \psi) \rightarrow (\Diamond\varphi \rightarrow \Diamond\psi)$ > using " $\rightarrow$ I" by blast
2682 qed
2683 lemmas "K $\Diamond$ " = "KBasic:13"
2684 AOT_theorem "KBasic:14": < $\Diamond\Box\varphi \equiv \neg\Box\Diamond\neg\varphi$ > (157.14)
2685 by (meson "RE $\Diamond$ " "KBasic:11" "KBasic:12" " $\equiv$ E"(6) "oth-class-taut:3:a")
2686 AOT_theorem "KBasic:15": < $(\Box\varphi \vee \Box\psi) \rightarrow \Box(\varphi \vee \psi)$ > (157.15)
2687 proof -
2688   AOT_modally_strict {
2689     AOT_have < $\varphi \rightarrow (\varphi \vee \psi)$ > and < $\psi \rightarrow (\varphi \vee \psi)$ >
2690     by (auto simp: "Disjunction Addition"(1) "Disjunction Addition"(2))
2691   }
2692   AOT_hence < $\Box\varphi \rightarrow \Box(\varphi \vee \psi)$ > and < $\Box\psi \rightarrow \Box(\varphi \vee \psi)$ >
2693   using RM by blast+
2694   AOT_thus < $(\Box\varphi \vee \Box\psi) \rightarrow \Box(\varphi \vee \psi)$ >
2695   by (metis " $\vee$ E"(1) "deduction-theorem")
2696 qed
2697
2698 AOT_theorem "KBasic:16": < $(\Box\varphi \ \& \ \Diamond\psi) \rightarrow \Diamond(\varphi \ \& \ \psi)$ > (157.16)
2699 by (meson "KBasic:13" "RM:1" Adjunction "Hypothetical Syllogism"
2700      Importation " $\rightarrow$ E")
2701
2702 AOT_theorem "rule-sub-lem:1:a": (158.1.a)
2703 assumes < $\vdash_{\Box} \Box(\psi \equiv \chi)$ >
2704 shows < $\vdash_{\Box} \neg\psi \equiv \neg\chi$ >
2705 using "qml:2"[axiom_inst, THEN " $\rightarrow$ E", OF assms]
2706      " $\equiv$ E"(1) "oth-class-taut:4:b" by blast
2707

```

```

2708 AOT_theorem "rule-sub-lem:1:b": (158.1.b)
2709   assumes <math>\vdash_{\Box} \Box(\psi \equiv \chi)>
2710   shows <math>\vdash_{\Box} (\psi \rightarrow \Theta) \equiv (\chi \rightarrow \Theta)>
2711   using "qml:2"[axiom_inst, THEN "\rightarrowE", OF assms]
2712   using "oth-class-taut:4:c" "vdash-properties:6" by blast
2713
2714 AOT_theorem "rule-sub-lem:1:c": (158.1.c)
2715   assumes <math>\vdash_{\Box} \Box(\psi \equiv \chi)>
2716   shows <math>\vdash_{\Box} (\Theta \rightarrow \psi) \equiv (\Theta \rightarrow \chi)>
2717   using "qml:2"[axiom_inst, THEN "\rightarrowE", OF assms]
2718   using "oth-class-taut:4:d" "vdash-properties:6" by blast
2719
2720 AOT_theorem "rule-sub-lem:1:d": (158.1.d)
2721   assumes <math>\text{for arbitrary } \alpha: \vdash_{\Box} \Box(\psi\{\alpha\} \equiv \chi\{\alpha\})>
2722   shows <math>\vdash_{\Box} \forall \alpha \psi\{\alpha\} \equiv \forall \alpha \chi\{\alpha\}>
2723   proof -
2724     AOT_modally_strict {
2725       AOT_have <math>\langle \forall \alpha (\psi\{\alpha\} \equiv \chi\{\alpha\}) \rangle
2726       using "qml:2"[axiom_inst, THEN "\rightarrowE", OF assms] "\forallI" by fast
2727       AOT_hence 0: <math>\langle \psi\{\alpha\} \equiv \chi\{\alpha\} \rangle \text{ for } \alpha \text{ using "\forallE" by blast}
2728       AOT_show <math>\langle \forall \alpha \psi\{\alpha\} \equiv \forall \alpha \chi\{\alpha\} \rangle
2729       proof (rule "\equivI"; rule "\rightarrowI")
2730         AOT_assume <math>\langle \forall \alpha \psi\{\alpha\} \rangle
2731         AOT_hence <math>\langle \psi\{\alpha\} \rangle \text{ for } \alpha \text{ using "\forallE" by blast}
2732         AOT_hence <math>\langle \chi\{\alpha\} \rangle \text{ for } \alpha \text{ using 0 "\equivE" by blast}
2733         AOT_thus <math>\langle \forall \alpha \chi\{\alpha\} \rangle \text{ by (rule "\forallI")}
2734       next
2735         AOT_assume <math>\langle \forall \alpha \chi\{\alpha\} \rangle
2736         AOT_hence <math>\langle \chi\{\alpha\} \rangle \text{ for } \alpha \text{ using "\forallE" by blast}
2737         AOT_hence <math>\langle \psi\{\alpha\} \rangle \text{ for } \alpha \text{ using 0 "\equivE" by blast}
2738         AOT_thus <math>\langle \forall \alpha \psi\{\alpha\} \rangle \text{ by (rule "\forallI")}
2739       qed
2740     }
2741   qed
2742
2743 AOT_theorem "rule-sub-lem:1:e": (158.1.e)
2744   assumes <math>\vdash_{\Box} \Box(\psi \equiv \chi)>
2745   shows <math>\vdash_{\Box} [\lambda \psi] \equiv [\lambda \chi]>
2746   using "qml:2"[axiom_inst, THEN "\rightarrowE", OF assms]
2747   using "\equivE"(1) "propositions-lemma:6" by blast
2748
2749 AOT_theorem "rule-sub-lem:1:f": (158.1.f)
2750   assumes <math>\vdash_{\Box} \Box(\psi \equiv \chi)>
2751   shows <math>\vdash_{\Box} \mathcal{A}\psi \equiv \mathcal{A}\chi>
2752   using "qml:2"[axiom_inst, THEN "\rightarrowE", OF assms, THEN "RA[2]"]
2753   by (metis "Act-Basic:5" "\equivE"(1))
2754
2755 AOT_theorem "rule-sub-lem:1:g": (158.1.g)
2756   assumes <math>\vdash_{\Box} \Box(\psi \equiv \chi)>
2757   shows <math>\vdash_{\Box} \Box\psi \equiv \Box\chi>
2758   using "KBasic:6" assms "vdash-properties:6" by blast
2759
2760 text<Note that instead of deriving @<math>\text{rule-sub-lem:2}</math>,
2761   @<math>\text{rule-sub-lem:3}</math>, @<math>\text{rule-sub-lem:4}</math>,
2762   and @<math>\text{rule-sub-nec}</math>, we construct substitution methods instead.>
2763
2764 class AOT_subst =
2765   fixes AOT_subst :: "( $\alpha \Rightarrow \text{bool}$ )  $\Rightarrow$  bool"
2766   and AOT_subst_cond :: "( $\alpha \Rightarrow \alpha \Rightarrow \text{bool}$ )"
2767   assumes AOT_subst:
2768     "AOT_subst  $\varphi \implies$  AOT_subst_cond  $\psi \chi \implies$  [ $v \models \langle \varphi \psi \rangle \equiv \langle \varphi \chi \rangle$ ]"
2769
2770 named_theorems AOT_substI

```

```

2771
2772 instantiation o :: AOT_subst
2773 begin
2774
2775 inductive AOT_subst_o where
2776   AOT_subst_o_id[AOT_substI]:
2777     <AOT_subst_o (λφ. φ)>
2778   | AOT_subst_o_const[AOT_substI]:
2779     <AOT_subst_o (λφ. ψ)>
2780   | AOT_subst_o_not[AOT_substI]:
2781     <AOT_subst_o Θ ⇒ AOT_subst_o (λ φ. «¬Θ{φ}»)>
2782   | AOT_subst_o_imp[AOT_substI]:
2783     <AOT_subst_o Θ ⇒ AOT_subst_o Ξ ⇒ AOT_subst_o (λ φ. «Θ{φ} → Ξ{φ}»)>
2784   | AOT_subst_o_lambda0[AOT_substI]:
2785     <AOT_subst_o Θ ⇒ AOT_subst_o (λ φ. (AOT_lambda0 (Θ φ)))>
2786   | AOT_subst_o_act[AOT_substI]:
2787     <AOT_subst_o Θ ⇒ AOT_subst_o (λ φ. «AΘ{φ}»)>
2788   | AOT_subst_o_box[AOT_substI]:
2789     <AOT_subst_o Θ ⇒ AOT_subst_o (λ φ. «□Θ{φ}»)>
2790   | AOT_subst_o_by_def[AOT_substI]:
2791     <(λ ψ . AOT_model_equiv_def (Θ ψ) (Ξ ψ)) ⇒
2792       AOT_subst_o Ξ ⇒ AOT_subst_o Θ>
2793
2794
2795 definition AOT_subst_cond_o where
2796   <AOT_subst_cond_o ≡ λ ψ χ . ∀ v . [v ⊨ ψ ≡ χ]>
2797
2798 instance
2799 proof
2800   fix ψ χ :: o and φ :: <o ⇒ o>
2801   assume cond: <AOT_subst_cond ψ χ>
2802   assume <AOT_subst φ>
2803   moreover AOT_have <⊢□ ψ ≡ χ>
2804     using cond unfolding AOT_subst_cond_o_def by blast
2805   ultimately AOT_show <⊢□ φ{ψ} ≡ φ{χ}>
2806   proof (induct arbitrary: ψ χ)
2807     case AOT_subst_o_id
2808     thus ?case
2809       using "≡E"(2) "oth-class-taut:4:b" "rule-sub-lem:1:a" by blast
2810   next
2811     case (AOT_subst_o_const ψ)
2812     thus ?case
2813       by (simp add: "oth-class-taut:3:a")
2814   next
2815     case (AOT_subst_o_not Θ)
2816     thus ?case
2817       by (simp add: RN "rule-sub-lem:1:a")
2818   next
2819     case (AOT_subst_o_imp Θ Ξ)
2820     thus ?case
2821       by (meson RN "≡E"(5) "rule-sub-lem:1:b" "rule-sub-lem:1:c")
2822   next
2823     case (AOT_subst_o_lambda0 Θ)
2824     thus ?case
2825       by (simp add: RN "rule-sub-lem:1:e")
2826   next
2827     case (AOT_subst_o_act Θ)
2828     thus ?case
2829       by (simp add: RN "rule-sub-lem:1:f")
2830   next
2831     case (AOT_subst_o_box Θ)
2832     thus ?case
2833       by (simp add: RN "rule-sub-lem:1:g")

```

```

2834 next
2835   case (AOT_subst_o_by_def  $\Theta \exists$ )
2836   AOT_modally_strict {
2837     AOT_have  $\langle \exists\{\psi\} \equiv \exists\{\chi\} \rangle$ 
2838     using AOT_subst_o_by_def by simp
2839     AOT_thus  $\langle \Theta\{\psi\} \equiv \Theta\{\chi\} \rangle$ 
2840     using "≡Df"[OF AOT_subst_o_by_def(1), of _  $\psi$ ]
2841           "≡Df"[OF AOT_subst_o_by_def(1), of _  $\chi$ ]
2842     by (metis "≡E"(6) "oth-class-taut:3:a")
2843   }
2844 qed
2845 qed
2846 end
2847
2848 instantiation "fun" :: (AOT_Term_id_2, AOT_subst) AOT_subst
2849 begin
2850
2851 definition AOT_subst_cond_fun ::  $\langle ('a \Rightarrow 'b) \Rightarrow ('a \Rightarrow 'b) \Rightarrow \text{bool} \rangle$  where
2852    $\langle \text{AOT\_subst\_cond\_fun} \equiv \lambda \varphi \psi . \forall \alpha . \text{AOT\_subst\_cond} (\varphi (\text{AOT\_term\_of\_var } \alpha))$ 
2853      $(\psi (\text{AOT\_term\_of\_var } \alpha)) \rangle$ 
2854
2855 inductive AOT_subst_fun ::  $\langle (( 'a \Rightarrow 'b ) \Rightarrow o) \Rightarrow \text{bool} \rangle$  where
2856   AOT_subst_fun_const[AOT_substI]:
2857      $\langle \text{AOT\_subst\_fun} (\lambda \varphi . \psi) \rangle$ 
2858   | AOT_subst_fun_id[AOT_substI]:
2859      $\langle \text{AOT\_subst } \Pi \Longrightarrow \text{AOT\_subst\_fun} (\lambda \varphi . \Pi (\varphi (\text{AOT\_term\_of\_var } \alpha))) \rangle$ 
2860   | AOT_subst_fun_all[AOT_substI]:
2861      $\langle \text{AOT\_subst } \Pi \Longrightarrow (\bigwedge \alpha . \text{AOT\_subst\_fun} (\Theta (\text{AOT\_term\_of\_var } \alpha))) \Longrightarrow$ 
2862        $\text{AOT\_subst\_fun} (\lambda \varphi :: 'a \Rightarrow 'b . \Pi \langle \forall \alpha \langle \Theta (\alpha :: 'a) \varphi \rangle \rangle) \rangle$ 
2863   | AOT_subst_fun_not[AOT_substI]:
2864      $\langle \text{AOT\_subst } \Pi \Longrightarrow \text{AOT\_subst\_fun} (\lambda \varphi . \langle \neg \langle \Pi \varphi \rangle \rangle) \rangle$ 
2865   | AOT_subst_fun_imp[AOT_substI]:
2866      $\langle \text{AOT\_subst } \Pi \Longrightarrow \text{AOT\_subst } \Theta \Longrightarrow \text{AOT\_subst\_fun} (\lambda \varphi . \langle \langle \Pi \varphi \rangle \rightarrow \langle \Theta \varphi \rangle \rangle) \rangle$ 
2867   | AOT_subst_fun_lambda0[AOT_substI]:
2868      $\langle \text{AOT\_subst } \Theta \Longrightarrow \text{AOT\_subst\_fun} (\lambda \varphi . (\text{AOT\_lambda0 } (\Theta \varphi))) \rangle$ 
2869   | AOT_subst_fun_act[AOT_substI]:
2870      $\langle \text{AOT\_subst } \Theta \Longrightarrow \text{AOT\_subst\_fun} (\lambda \varphi . \langle \mathcal{A} \langle \Theta \varphi \rangle \rangle) \rangle$ 
2871   | AOT_subst_fun_box[AOT_substI]:
2872      $\langle \text{AOT\_subst } \Theta \Longrightarrow \text{AOT\_subst\_fun} (\lambda \varphi . \langle \Box \langle \Theta \varphi \rangle \rangle) \rangle$ 
2873   | AOT_subst_fun_def[AOT_substI]:
2874      $\langle (\bigwedge \varphi . \text{AOT\_model\_equiv\_def} (\Theta \varphi) (\Pi \varphi)) \Longrightarrow$ 
2875        $\text{AOT\_subst\_fun } \Pi \Longrightarrow \text{AOT\_subst\_fun } \Theta \rangle$ 
2876
2877 instance proof
2878   fix  $\psi \chi :: \langle 'a \Rightarrow 'b \rangle$  and  $\varphi :: \langle ('a \Rightarrow 'b) \Rightarrow o \rangle$ 
2879   assume  $\langle \text{AOT\_subst } \varphi \rangle$ 
2880   moreover assume cond:  $\langle \text{AOT\_subst\_cond } \psi \chi \rangle$ 
2881   ultimately AOT_show  $\langle \vdash_{\Box} \langle \varphi \psi \rangle \equiv \langle \varphi \chi \rangle \rangle$ 
2882   proof(induct)
2883     case (AOT_subst_fun_const  $\psi$ )
2884     then show ?case by (simp add: "oth-class-taut:3:a")
2885   next
2886   case (AOT_subst_fun_id  $\Pi x$ )
2887   then show ?case by (simp add: AOT_subst AOT_subst_cond_fun_def)
2888   next
2889   next
2890   case (AOT_subst_fun_all  $\Pi \Theta$ )
2891   AOT_have  $\langle \vdash_{\Box} \Box (\Theta\{\alpha, \langle \psi \rangle\} \equiv \Theta\{\alpha, \langle \chi \rangle\}) \rangle$  for  $\alpha$ 
2892   using AOT_subst_fun_all.hyps(3) AOT_subst_fun_all.prem1 RN by presburger
2893   thus ?case using AOT_subst[OF AOT_subst_fun_all(1)]
2894   by (simp add: RN "rule-sub-lem:1:d"
2895       AOT_subst_cond_fun_def AOT_subst_cond_o_def)
2896   next

```



```

2897 case (AOT_subst_fun_not II)
2898 then show ?case by (simp add: RN "rule-sub-lem:1:a")
2899 next
2900 case (AOT_subst_fun_imp II  $\Theta$ )
2901 then show ?case
2902   unfolding AOT_subst_cond_fun_def AOT_subst_cond_o_def
2903   by (meson " $\equiv$ E"(5) "oth-class-taut:4:c" "oth-class-taut:4:d" " $\rightarrow$ E")
2904 next
2905 case (AOT_subst_fun_lambda0  $\Theta$ )
2906 then show ?case by (simp add: RN "rule-sub-lem:1:e")
2907 next
2908 case (AOT_subst_fun_act  $\Theta$ )
2909 then show ?case by (simp add: RN "rule-sub-lem:1:f")
2910 next
2911 case (AOT_subst_fun_box  $\Theta$ )
2912 then show ?case by (simp add: RN "rule-sub-lem:1:g")
2913 next
2914 case (AOT_subst_fun_def  $\Theta$  II)
2915 then show ?case
2916   by (meson "df-rules-formulas[3]" "df-rules-formulas[4]" " $\equiv$ I" " $\equiv$ E"(5))
2917 qed
2918 qed
2919 end
2920
2921 ML<
2922 fun prove_AOT_subst_tac ctxt = REPEAT (SUBGOAL (fn (trm,_) => let
2923   fun findHeadConst (Const x) = SOME x
2924     | findHeadConst (A $ _) = findHeadConst A
2925     | findHeadConst _ = NONE
2926   fun findDef (Const (const_name<AOT_model_equiv_def>, _) $ lhs $ _)
2927     = findHeadConst lhs
2928     | findDef (A $ B) = (case findDef A of SOME x => SOME x | _ => findDef B)
2929     | findDef (Abs (_,_,c)) = findDef c
2930     | findDef _ = NONE
2931   val const_opt = (findDef trm)
2932   val defs = case const_opt of SOME const => List.filter (fn thm => let
2933     val concl = Thm.concl_of thm
2934     val thmconst = (findDef concl)
2935     in case thmconst of SOME (c,_) => fst const = c | _ => false end)
2936     (AOT_Definitions.get ctxt)
2937     | _ => []
2938   val tac = case defs of
2939     [] => safe_step_tac (ctxt addSIs @ {thms AOT_substI}) 1
2940     | _ => resolve_tac ctxt defs 1
2941   in tac end) 1)
2942 fun getSubstThm ctxt reversed phi p q = let
2943   val p_ty = Term.type_of p
2944   val abs = HOLogic.mk_Trueprop (@{const AOT_subst(_)} $ phi)
2945   val abs = Syntax.check_term ctxt abs
2946   val substThm = Goal.prove ctxt [] [] abs
2947     (fn {context=ctxt, prems=_} => prove_AOT_subst_tac ctxt)
2948   val substThm = substThm RS @ {thm AOT_subst}
2949   in if reversed then let
2950     val substThm = Drule.instantiate_normalize
2951       (TVars.empty, Vars.make [((("χ", 0), p_ty), Thm.cterm_of ctxt p),
2952         ((("ψ", 0), p_ty), Thm.cterm_of ctxt q))] substThm
2953     val substThm = substThm RS @ {thm " $\equiv$ E"(1)}
2954     in substThm end
2955   else
2956     let
2957     val substThm = Drule.instantiate_normalize
2958       (TVars.empty, Vars.make [((("ψ", 0), p_ty), Thm.cterm_of ctxt p),
2959         ((("χ", 0), p_ty), Thm.cterm_of ctxt q))] substThm

```

```

2960   val substThm = substThm RS @{thm "≡E"(2)}
2961   in substThm end end
2962 >
2963
2964 method_setup AOT_subst = <
2965 Scan.option (Scan.lift (Args.parens (Args.$$$ "reverse")) -
2966 Scan.lift (Parse.embedded_inner_syntax - Parse.embedded_inner_syntax) -
2967 Scan.option (Scan.lift (Args.$$$ "for" - Args.colon) |-
2968 Scan.repeat1 (Scan.lift (Parse.embedded_inner_syntax) -
2969 Scan.option (Scan.lift (Args.$$$ ":@" |- Parse.embedded_inner_syntax))))
2970 » (fn ((reversed,(raw_p,raw_q)),raw_bounds) => (fn ctxt =>
2971 (Method.SIMPLE_METHOD (Subgoal.FOCUS (fn {context = ctxt, params = _,
2972   prems = prems, asms = asms, concl = concl, schematics = _} =>
2973 let
2974 val thms = prems
2975 val ctxt' = ctxt
2976 val ctxt = Context_Position.set_visible false ctxt
2977 val raw_bounds = case raw_bounds of SOME bounds => bounds | _ => []
2978
2979 val ctxt = (fold (fn (bound, ty) => fn ctxt =>
2980   let
2981     val bound = AOT_read_term @{nonterminal τ'} ctxt bound
2982     val ty = Option.map (Syntax.read_typ ctxt) ty
2983     val ctxt = case ty of SOME ty => let
2984       val bound = Const ("_type_constraint_", Type ("fun", [ty,ty])) $ bound
2985       val bound = Syntax.check_term ctxt bound
2986     in Variable.declare_term bound ctxt end | _ => ctxt
2987     in ctxt end)) raw_bounds ctxt
2988
2989 val p = AOT_read_term @{nonterminal φ'} ctxt raw_p
2990 val p = Syntax.check_term ctxt p
2991 val ctxt = Variable.declare_term p ctxt
2992 val q = AOT_read_term @{nonterminal φ'} ctxt raw_q
2993 val q = Syntax.check_term ctxt q
2994 val ctxt = Variable.declare_term q ctxt
2995
2996 val bounds = (map (fn (bound, _) =>
2997   Syntax.check_term ctxt (AOT_read_term @{nonterminal τ'} ctxt bound)
2998 )) raw_bounds
2999 val p = fold (fn bound => fn p =>
3000   Term.abs ("α", Term.type_of bound) (Term.abstract_over (bound,p)))
3001   bounds p
3002 val p = Syntax.check_term ctxt p
3003 val p_ty = Term.type_of p
3004
3005 val pat = @{const Trueprop} $
3006   (@{const AOT_model_valid_in} $ Var (("w",0), @{typ w}) $
3007     (Var (("φ",0), Type (type_name<fun>, [p_ty, @{typ o}]))) $ p))
3008 val univ = Unify.matchers (Context.Proof ctxt) [(pat, Thm.term_of concl)]
3009 val univ = hd (Seq.list_of univ) (* TODO: consider all matches *)
3010 val phi = the (Envir.lookup univ
3011   (("φ",0), Type (type_name<fun>, [p_ty, @{typ o}])))
3012
3013 val q = fold (fn bound => fn q =>
3014   Term.abs ("α", Term.type_of bound) (Term.abstract_over (bound,q))) bounds q
3015 val q = Syntax.check_term ctxt q
3016
3017 (* Reparse to report bounds as fixes. *)
3018 val ctxt = Context_Position.restore_visible ctxt' ctxt
3019 val ctxt' = ctxt
3020 fun unsource str = fst (Input.source_content (Syntax.read_input str))
3021 val (_,ctxt') = Proof_Context.add_fixes (map (fn (str,_) =>
3022   (Binding.make (unsource str, Position.none), NONE, Mixfix.NoSyn)) raw_bounds)

```

```

3023   ctxt'
3024   val _ = (map (fn (x,_) =>
3025     Syntax.check_term ctxt (AOT_read_term @{nonterminal  $\tau'$ } ctxt' x)))
3026     raw_bounds
3027   val _ = AOT_read_term @{nonterminal  $\varphi'$ } ctxt' raw_p
3028   val _ = AOT_read_term @{nonterminal  $\varphi'$ } ctxt' raw_q
3029   val reversed = case reversed of SOME _ => true | _ => false
3030   val simpThms = [@{thm AOT_subst_cond_o_def}, @{thm AOT_subst_cond_fun_def}]
3031   in
3032     resolve_tac ctxt [getSubstThm ctxt reversed phi p q] 1
3033   THEN simp_tac (ctxt addsimps simpThms) 1
3034   THEN (REPEAT (resolve_tac ctxt [@{thm allI}] 1))
3035   THEN (TRY (resolve_tac ctxt thms 1))
3036   end
3037 ) ctxt 1))))
3038 >
3039
3040 method_setup AOT_subst_def = <
3041 Scan.option (Scan.lift (Args.parens (Args.$$$ "reverse")))) -
3042   Attrib.thm
3043   » (fn (reversed,fact) => (fn ctxt =>
3044     (Method.SIMPLE_METHOD (Subgoal.FOCUS (fn {context = ctxt, params = _,
3045       prems = prems, asms = asms, concl = concl, schematics = _} =>
3046       let
3047         val c = Thm.concl_of fact
3048         val (lhs, rhs) = case c of (const<Trueprop> $
3049           (const<AOT_model_equiv_def> $ lhs $ rhs)) => (lhs, rhs)
3050         | _ => raise Fail "Definition expected."
3051         val substCond = HLogic.mk_Trueprop
3052           (Const (const_name<AOT_subst_cond>, dummyT) $ lhs $ rhs)
3053         val substCond = Syntax.check_term
3054           (Proof_Context.set_mode Proof_Context.mode_schematic ctxt)
3055           substCond
3056         val simpThms = [@{thm AOT_subst_cond_o_def},
3057           @{thm AOT_subst_cond_fun_def},
3058           fact RS @{thm "≡Df"}]
3059         val substCondThm = Goal.prove ctxt [] [] substCond
3060           (fn {context=ctxt, prems=prems} =>
3061             (SUBGOAL (fn (trm,int) =>
3062               auto_tac (ctxt addsimps simpThms)) 1))
3063         val substThm = substCondThm RSN (2,@{thm AOT_subst})
3064         in
3065           resolve_tac ctxt [substThm RS
3066             (case reversed of NONE => @{thm "≡E"(2)} | _ => @{thm "≡E"(1)})] 1
3067         THEN prove_AOT_subst_tac ctxt
3068         THEN (TRY (resolve_tac ctxt prems 1))
3069         end
3070       ) ctxt 1))))
3071 >
3072
3073 method_setup AOT_subst_thm = <
3074 Scan.option (Scan.lift (Args.parens (Args.$$$ "reverse")))) -
3075   Attrib.thm
3076   » (fn (reversed,fact) => (fn ctxt =>
3077     (Method.SIMPLE_METHOD (Subgoal.FOCUS (fn {context = ctxt, params = _,
3078       prems = prems, asms = asms, concl = concl, schematics = _} =>
3079       let
3080         val c = Thm.concl_of fact
3081         val (lhs, rhs) = case c of
3082           (const<Trueprop> $
3083             (const<AOT_model_valid_in> $ _ $
3084               (const<AOT_equiv> $ lhs $ rhs))) => (lhs, rhs)
3085         | _ => raise Fail "Equivalence expected."

```

```

3086
3087 val substCond = HLogic.mk_Trueprop
3088   (Const (const_name<AOT_subst_cond>, dummyT) $ lhs $ rhs)
3089 val substCond = Syntax.check_term
3090   (Proof_Context.set_mode Proof_Context.mode_schematic ctxt)
3091   substCond
3092 val simpThms = [@{thm AOT_subst_cond_o_def},
3093   @{thm AOT_subst_cond_fun_def},
3094   fact]
3095 val substCondThm = Goal.prove ctxt [] [] substCond
3096   (fn {context=ctxt, prems=prems} =>
3097     (SUBGOAL (fn (trm,int) => auto_tac (ctxt addsimps simpThms)) 1))
3098 val substThm = substCondThm RSN (2,@{thm AOT_subst})
3099 in
3100 resolve_tac ctxt [substThm RS
3101   (case reversed of NONE => @{thm "≡E"(2)} | _ => @{thm "≡E"(1)})] 1
3102 THEN prove_AOT_subst_tac ctxt
3103 THEN (TRY (resolve_tac ctxt prems 1))
3104 end
3105 ) ctxt 1)))
3106 >
3107
3108 AOT_theorem "rule-sub-remark:1[1]": (160.1)
3109   assumes <⊢ A!x ≡ ¬◇E!x> and <¬A!x>
3110   shows <¬¬◇E!x>
3111   by (AOT_subst (reverse) <¬◇E!x> <A!x>)
3112     (auto simp: assms)
3113
3114 AOT_theorem "rule-sub-remark:1[2]": (160.1)
3115   assumes <⊢ A!x ≡ ¬◇E!x> and <¬¬◇E!x>
3116   shows <¬A!x>
3117   by (AOT_subst <A!x> <¬◇E!x>)
3118     (auto simp: assms)
3119
3120 AOT_theorem "rule-sub-remark:2[1]": (160.2)
3121   assumes <⊢ [R]xy ≡ ([R]xy & ([Q]a ∨ ¬[Q]a))>
3122     and <p → [R]xy>
3123   shows <p → [R]xy & ([Q]a ∨ ¬[Q]a)>
3124   by (AOT_subst_thm (reverse) assms(1)) (simp add: assms(2))
3125
3126 AOT_theorem "rule-sub-remark:2[2]": (160.2)
3127   assumes <⊢ [R]xy ≡ ([R]xy & ([Q]a ∨ ¬[Q]a))>
3128     and <p → [R]xy & ([Q]a ∨ ¬[Q]a)>
3129   shows <p → [R]xy>
3130   by (AOT_subst_thm assms(1)) (simp add: assms(2))
3131
3132 AOT_theorem "rule-sub-remark:3[1]": (160.3)
3133   assumes <for arbitrary x: ⊢ A!x ≡ ¬◇E!x>
3134     and <∃x A!x>
3135   shows <∃x ¬◇E!x>
3136   by (AOT_subst (reverse) <¬◇E!x> <A!x> for: x)
3137     (auto simp: assms)
3138
3139 AOT_theorem "rule-sub-remark:3[2]": (160.3)
3140   assumes <for arbitrary x: ⊢ A!x ≡ ¬◇E!x>
3141     and <∃x ¬◇E!x>
3142   shows <∃x A!x>
3143   by (AOT_subst <A!x> <¬◇E!x> for: x)
3144     (auto simp: assms)
3145
3146 AOT_theorem "rule-sub-remark:4[1]": (160.4)
3147   assumes <⊢ ¬¬[P]x ≡ [P]x> and <⊢ ¬¬[P]x>
3148   shows <⊢ [P]x>

```

```

3149   by (AOT_subst_thm (reverse) assms(1)) (simp add: assms(2))
3150
3151 AOT_theorem "rule-sub-remark:4[2]": (160.4)
3152   assumes <math>\vdash_{\square} \neg\neg[P]x \equiv [P]x</math> and <math>\langle \mathcal{A}[P]x \rangle</math>
3153   shows <math>\langle \mathcal{A}\neg\neg[P]x \rangle</math>
3154   by (AOT_subst_thm assms(1)) (simp add: assms(2))
3155
3156 AOT_theorem "rule-sub-remark:5[1]": (160.5)
3157   assumes <math>\vdash_{\square} (\varphi \rightarrow \psi) \equiv (\neg\psi \rightarrow \neg\varphi)</math> and <math>\langle \square(\varphi \rightarrow \psi) \rangle</math>
3158   shows <math>\langle \square(\neg\psi \rightarrow \neg\varphi) \rangle</math>
3159   by (AOT_subst_thm (reverse) assms(1)) (simp add: assms(2))
3160
3161 AOT_theorem "rule-sub-remark:5[2]": (160.5)
3162   assumes <math>\vdash_{\square} (\varphi \rightarrow \psi) \equiv (\neg\psi \rightarrow \neg\varphi)</math> and <math>\langle \square(\neg\psi \rightarrow \neg\varphi) \rangle</math>
3163   shows <math>\langle \square(\varphi \rightarrow \psi) \rangle</math>
3164   by (AOT_subst_thm assms(1)) (simp add: assms(2))
3165
3166 AOT_theorem "rule-sub-remark:6[1]": (160.6)
3167   assumes <math>\vdash_{\square} \psi \equiv \chi</math> and <math>\langle \square(\varphi \rightarrow \psi) \rangle</math>
3168   shows <math>\langle \square(\varphi \rightarrow \chi) \rangle</math>
3169   by (AOT_subst_thm (reverse) assms(1)) (simp add: assms(2))
3170
3171 AOT_theorem "rule-sub-remark:6[2]": (160.6)
3172   assumes <math>\vdash_{\square} \psi \equiv \chi</math> and <math>\langle \square(\varphi \rightarrow \chi) \rangle</math>
3173   shows <math>\langle \square(\varphi \rightarrow \psi) \rangle</math>
3174   by (AOT_subst_thm assms(1)) (simp add: assms(2))
3175
3176 AOT_theorem "rule-sub-remark:7[1]": (160.7)
3177   assumes <math>\vdash_{\square} \varphi \equiv \neg\neg\varphi</math> and <math>\langle \square(\varphi \rightarrow \varphi) \rangle</math>
3178   shows <math>\langle \square(\neg\neg\varphi \rightarrow \varphi) \rangle</math>
3179   by (AOT_subst_thm (reverse) assms(1)) (simp add: assms(2))
3180
3181 AOT_theorem "rule-sub-remark:7[2]": (160.7)
3182   assumes <math>\vdash_{\square} \varphi \equiv \neg\neg\varphi</math> and <math>\langle \square(\neg\neg\varphi \rightarrow \varphi) \rangle</math>
3183   shows <math>\langle \square(\varphi \rightarrow \varphi) \rangle</math>
3184   by (AOT_subst_thm assms(1)) (simp add: assms(2))
3185
3186 AOT_theorem "KBasic2:1": <math>\langle \square\neg\varphi \equiv \neg\Diamond\varphi \rangle</math> (161.1)
3187   by (meson "conventions:5" "contraposition:2"
3188         "Hypothetical Syllogism" "df-rules-formulas[3]"
3189         "df-rules-formulas[4]" "\equiv I" "useful-tautologies:1")
3190
3191 AOT_theorem "KBasic2:2": <math>\langle \Diamond(\varphi \vee \psi) \equiv (\Diamond\varphi \vee \Diamond\psi) \rangle</math> (161.2)
3192 proof -
3193   AOT_have <math>\langle \Diamond(\varphi \vee \psi) \equiv \Diamond(\neg\neg\varphi \ \& \ \neg\neg\psi) \rangle</math>
3194     by (simp add: "RE\Diamond" "oth-class-taut:5:b")
3195   also AOT_have <math>\langle \dots \equiv \neg\square(\neg\neg\varphi \ \& \ \neg\neg\psi) \rangle</math>
3196     using "KBasic:11" "\equiv E"(6) "oth-class-taut:3:a" by blast
3197   also AOT_have <math>\langle \dots \equiv \neg(\square\neg\varphi \ \& \ \square\neg\psi) \rangle</math>
3198     using "KBasic:3" "\equiv E"(1) "oth-class-taut:4:b" by blast
3199   also AOT_have <math>\langle \dots \equiv \neg(\neg\Diamond\varphi \ \& \ \neg\Diamond\psi) \rangle</math>
3200     using "KBasic2:1"
3201     by (AOT_subst <math>\langle \square\neg\varphi \rangle \langle \neg\Diamond\varphi \rangle</math>; AOT_subst <math>\langle \square\neg\psi \rangle \langle \neg\Diamond\psi \rangle</math>;
3202         auto simp: "oth-class-taut:3:a")
3203   also AOT_have <math>\langle \dots \equiv \neg\neg(\Diamond\varphi \ \vee \ \Diamond\psi) \rangle</math>
3204     using "\equiv E"(6) "oth-class-taut:3:b" "oth-class-taut:5:b" by blast
3205   also AOT_have <math>\langle \dots \equiv \Diamond\varphi \ \vee \ \Diamond\psi \rangle</math>
3206     by (simp add: "\equiv I" "useful-tautologies:1" "useful-tautologies:2")
3207   finally show ?thesis .
3208 qed
3209
3210 AOT_theorem "KBasic2:3": <math>\langle \Diamond(\varphi \ \& \ \psi) \rightarrow (\Diamond\varphi \ \& \ \Diamond\psi) \rangle</math> (161.3)
3211   by (metis "RM\Diamond" "\& I" "Conjunction Simplification"(1,2))

```

```

3212         "→I" "modus-tollens:1" "reductio-aa:1")
3213
3214 AOT_theorem "KBasic2:4": <◇(φ → ψ) ≡ (□φ → ◇ψ)> (161.4)
3215 proof -
3216   AOT_have <◇(φ → ψ) ≡ ◇(¬φ ∨ ψ)>
3217     by (AOT_subst <φ → ψ> <¬φ ∨ ψ>)
3218       (auto simp: "oth-class-taut:1:c" "oth-class-taut:3:a")
3219   also AOT_have <... ≡ ◇¬φ ∨ ◇ψ>
3220     by (simp add: "KBasic2:2")
3221   also AOT_have <... ≡ ¬□φ ∨ ◇ψ>
3222     by (AOT_subst <¬□φ> <◇¬φ>)
3223       (auto simp: "KBasic:11" "oth-class-taut:3:a")
3224   also AOT_have <... ≡ □φ → ◇ψ>
3225     using "≡E"(6) "oth-class-taut:1:c" "oth-class-taut:3:a" by blast
3226   finally show ?thesis .
3227 qed
3228
3229 AOT_theorem "KBasic2:5": <◇◇φ ≡ ¬□□¬φ> (161.5)
3230 using "conventions:5"[THEN "≡Df"]
3231 by (AOT_subst <◇φ> <¬□¬φ>;
    AOT_subst <◇¬□¬φ> <¬□¬¬□¬φ>;
    AOT_subst (reverse) <¬¬□¬φ> <□¬φ>)
3232   (auto simp: "oth-class-taut:3:b" "oth-class-taut:3:a")
3233
3234
3235
3236
3237 AOT_theorem "KBasic2:6": <□(φ ∨ ψ) → (□φ ∨ ◇ψ)> (161.6)
3238 proof(rule "→I"; rule "raa-cor:1")
3239   AOT_assume <□(φ ∨ ψ)>
3240   AOT_hence <□(¬φ → ψ)>
3241     using "conventions:2"[THEN "≡Df"]
3242     by (AOT_subst (reverse) <¬φ → ψ> <φ ∨ ψ>) simp
3243   AOT_hence 1: <◇¬φ → ◇ψ>
3244     using "KBasic:13" "vdash-properties:10" by blast
3245   AOT_assume <¬(□φ ∨ ◇ψ)>
3246   AOT_hence <¬□φ> and <¬◇ψ>
3247     using "&E" "≡E"(1) "oth-class-taut:5:d" by blast+
3248   AOT_thus <◇ψ & ¬◇ψ>
3249     using "&I"(1) 1[THEN "→E"] "KBasic:11" "≡E"(4) "raa-cor:3" by blast
3250 qed
3251
3252 AOT_theorem "KBasic2:7": <(□(φ ∨ ψ) & ◇¬φ) → ◇ψ> (161.7)
3253 proof(rule "→I"; frule "&E"(1); drule "&E"(2))
3254   AOT_assume <□(φ ∨ ψ)>
3255   AOT_hence 1: <□φ ∨ ◇ψ>
3256     using "KBasic2:6" "∀I"(2) "∀E"(1) by blast
3257   AOT_assume <◇¬φ>
3258   AOT_hence <¬□φ> using "KBasic:11" "≡E"(2) by blast
3259   AOT_thus <◇ψ> using 1 "∀E"(2) by blast
3260 qed
3261
3262 AOT_theorem "T-S5-fund:1": <φ → ◇φ> (162.1)
3263 by (meson "≡df I" "conventions:5" "contraposition:2"
    "Hypothetical Syllogism" "→I" "qml:2"[axiom_inst])
3264 lemmas "T◇" = "T-S5-fund:1"
3265
3266
3267 AOT_theorem "T-S5-fund:2": <◇□φ → □φ> (162.2)
3268 proof(rule "→I")
3269   AOT_assume <◇□φ>
3270   AOT_hence <¬□◇¬φ>
3271     using "KBasic:14" "≡E"(4) "raa-cor:3" by blast
3272   moreover AOT_have <◇¬φ → □◇¬φ>
3273     by (fact "qml:3"[axiom_inst])
3274   ultimately AOT_have <¬◇¬φ>

```

```

3275     using "modus-tollens:1" by blast
3276     AOT_thus <□φ> using "KBasic:12" "≡E"(2) by blast
3277 qed
3278 lemmas "5◇" = "T-S5-fund:2"
3279
3280 AOT_theorem "Act-Sub:1": <Aφ ≡ ¬A¬φ> (163.1)
3281   by (AOT_subst <A¬φ> <¬Aφ>)
3282     (auto simp: "logic-actual-nec:1"[axiom_inst] "oth-class-taut:3:b")
3283
3284 AOT_theorem "Act-Sub:2": <◇φ ≡ A◇φ> (163.2)
3285   using "conventions:5"[THEN "≡Df"]
3286   by (AOT_subst <◇φ> <¬□¬φ>)
3287     (metis "deduction-theorem" "≡I" "≡E"(1) "≡E"(2) "≡E"(3)
3288       "logic-actual-nec:1"[axiom_inst] "qml-act:2"[axiom_inst])
3289
3290 AOT_theorem "Act-Sub:3": <Aφ → ◇φ> (163.3)
3291   using "conventions:5"[THEN "≡Df"]
3292   by (AOT_subst <◇φ> <¬□¬φ>)
3293     (metis "Act-Sub:1" "→I" "≡E"(4) "nec-imp-act" "reductio-aa:2" "→E")
3294
3295 AOT_theorem "Act-Sub:4": <Aφ ≡ ◇Aφ> (163.4)
3296 proof (rule "≡I"; rule "→I")
3297   AOT_assume <Aφ>
3298   AOT_thus <◇Aφ> using "T◇" "vdash-properties:10" by blast
3299 next
3300   AOT_assume <◇Aφ>
3301   AOT_hence <¬□¬Aφ>
3302     using "≡dfE" "conventions:5" by blast
3303   AOT_hence <¬□A¬φ>
3304     by (AOT_subst <A¬φ> <¬Aφ>)
3305     (simp add: "logic-actual-nec:1"[axiom_inst])
3306   AOT_thus <Aφ>
3307     using "Act-Basic:1" "Act-Basic:6" "∨E"(3) "≡E"(4)
3308       "reductio-aa:1" by blast
3309 qed
3310
3311 AOT_theorem "Act-Sub:5": <◇Aφ → A◇φ> (163.5)
3312   by (metis "Act-Sub:2" "Act-Sub:3" "Act-Sub:4" "→I" "≡E"(1) "≡E"(2) "→E")
3313
3314 AOT_theorem "S5Basic:1": <◇φ ≡ □◇φ> (164.1)
3315   by (simp add: "≡I" "qml:2"[axiom_inst] "qml:3"[axiom_inst])
3316
3317 AOT_theorem "S5Basic:2": <□φ ≡ ◇□φ> (164.2)
3318   by (simp add: "T◇" "5◇" "≡I")
3319
3320 AOT_theorem "S5Basic:3": <φ → □◇φ> (164.3)
3321   using "T◇" "Hypothetical Syllogism" "qml:3"[axiom_inst] by blast
3322 lemmas "B" = "S5Basic:3"
3323
3324 AOT_theorem "S5Basic:4": <◇□φ → φ> (164.4)
3325   using "5◇" "Hypothetical Syllogism" "qml:2"[axiom_inst] by blast
3326 lemmas "B◇" = "S5Basic:4"
3327
3328 AOT_theorem "S5Basic:5": <□φ → □□φ> (164.5)
3329   using "RM:1" "B" "5◇" "Hypothetical Syllogism" by blast
3330 lemmas "4" = "S5Basic:5"
3331
3332 AOT_theorem "S5Basic:6": <□φ ≡ □□φ> (164.6)
3333   by (simp add: "4" "≡I" "qml:2"[axiom_inst])
3334
3335 AOT_theorem "S5Basic:7": <◇◇φ → ◇φ> (164.7)
3336   using "conventions:5"[THEN "≡Df"] "oth-class-taut:3:b"
3337   by (AOT_subst <◇◇φ> <¬□¬◇φ>);

```

```

3338     AOT_subst <◇φ> <¬□¬φ>;
3339     AOT_subst (reverse) <¬¬□¬φ> <□¬φ>;
3340     AOT_subst (reverse) <□□¬φ> <□¬φ>
3341     (auto simp: "S5Basic:6" "if-p-then-p")
3342
3343 lemmas "4◇" = "S5Basic:7"
3344
3345 AOT_theorem "S5Basic:8": <◇◇φ ≡ ◇φ> (164.8)
3346   by (simp add: "4◇" "T◇" "≡I")
3347
3348 AOT_theorem "S5Basic:9": <□(φ ∨ □ψ) ≡ (□φ ∨ □ψ)> (164.9)
3349   apply (rule "≡I"; rule "→I")
3350   using "KBasic2:6" "5◇" "∨I"(3) "if-p-then-p" "vdash-properties:10"
3351   apply blast
3352   by (meson "KBasic:15" "4" "∨I"(3) "∨E"(1) "Disjunction Addition"(1)
3353       "con-dis-taut:7" "intro-elim:1" "Commutativity of ∨")
3354
3355 AOT_theorem "S5Basic:10": <□(φ ∨ ◇ψ) ≡ (□φ ∨ ◇ψ)> (164.10)
3356 proof(rule "≡I"; rule "→I")
3357   AOT_assume <□(φ ∨ ◇ψ)>
3358   AOT_hence <□φ ∨ ◇◇ψ>
3359     by (meson "KBasic2:6" "∨I"(2) "∨E"(1))
3360   AOT_thus <□φ ∨ ◇ψ>
3361     by (meson "B◇" "4" "4◇" "T◇" "∨I"(3))
3362 next
3363   AOT_assume <□φ ∨ ◇ψ>
3364   AOT_hence <□φ ∨ □◇ψ>
3365     by (meson "S5Basic:1" "B◇" "S5Basic:6" "T◇" "5◇" "∨I"(3) "intro-elim:1")
3366   AOT_thus <□(φ ∨ ◇ψ)>
3367     by (meson "KBasic:15" "∨I"(3) "∨E"(1) "Disjunction Addition"(1,2))
3368 qed
3369
3370 AOT_theorem "S5Basic:11": <◇(φ & ◇ψ) ≡ (◇φ & ◇ψ)> (164.11)
3371 proof -
3372   AOT_have <◇(φ & ◇ψ) ≡ ◇(¬(¬φ ∨ ¬◇ψ))>
3373     by (AOT_subst <φ & ◇ψ> <¬(¬φ ∨ ¬◇ψ)>)
3374     (auto simp: "oth-class-taut:5:a" "oth-class-taut:3:a")
3375   also AOT_have <... ≡ ◇(¬(¬φ ∨ □¬ψ))>
3376     by (AOT_subst <□¬ψ> <¬◇ψ>)
3377     (auto simp: "KBasic2:1" "oth-class-taut:3:a")
3378   also AOT_have <... ≡ ¬□(¬φ ∨ □¬ψ)>
3379     using "KBasic:11" "≡E"(6) "oth-class-taut:3:a" by blast
3380   also AOT_have <... ≡ ¬(□¬φ ∨ □¬ψ)>
3381     using "S5Basic:9" "≡E"(1) "oth-class-taut:4:b" by blast
3382   also AOT_have <... ≡ ¬(¬◇φ ∨ ¬◇ψ)>
3383     using "KBasic2:1"
3384     by (AOT_subst <□¬φ> <¬◇φ>; AOT_subst <□¬ψ> <¬◇ψ>)
3385     (auto simp: "oth-class-taut:3:a")
3386   also AOT_have <... ≡ ◇φ & ◇ψ>
3387     using "≡E"(6) "oth-class-taut:3:a" "oth-class-taut:5:a" by blast
3388   finally show ?thesis .
3389 qed
3390
3391 AOT_theorem "S5Basic:12": <◇(φ & □ψ) ≡ (◇φ & □ψ)> (164.12)
3392 proof (rule "≡I"; rule "→I")
3393   AOT_assume <◇(φ & □ψ)>
3394   AOT_hence <◇φ & ◇□ψ>
3395     using "KBasic2:3" "vdash-properties:6" by blast
3396   AOT_thus <◇φ & □ψ>
3397     using "5◇" "&I" "&E"(1) "&E"(2) "vdash-properties:6" by blast
3398 next
3399   AOT_assume <◇φ & □ψ>
3400   moreover AOT_have <(□□ψ & ◇φ) → ◇(φ & □ψ)>

```



```

3401   by (AOT_subst < $\varphi \ \& \ \Box\psi \ \> \ \Box\psi \ \& \ \varphi \ >)$ 
3402     (auto simp: "Commutativity of &" "KBasic:16")
3403   ultimately AOT_show < $\Diamond(\varphi \ \& \ \Box\psi) \ >$ 
3404     by (metis "4" "&I" "Conjunction Simplification"(1,2) " $\rightarrow E$ ")
3405 qed
3406
3407 AOT_theorem "S5Basic:13": < $\Box(\varphi \ \rightarrow \ \Box\psi) \ \equiv \ \Box(\Diamond\varphi \ \rightarrow \ \psi) \ >$  (164.13)
3408 proof (rule " $\equiv I$ ")
3409   AOT_modally_strict {
3410     AOT_have < $\Box(\varphi \ \rightarrow \ \Box\psi) \ \rightarrow \ (\Diamond\varphi \ \rightarrow \ \psi) \ >$ 
3411       by (meson "KBasic:13" "B $\Diamond$ " "Hypothetical Syllogism" " $\rightarrow I$ ")
3412   }
3413   AOT_hence < $\Box\Box(\varphi \ \rightarrow \ \Box\psi) \ \rightarrow \ \Box(\Diamond\varphi \ \rightarrow \ \psi) \ >$ 
3414     by (rule RM)
3415   AOT_thus < $\Box(\varphi \ \rightarrow \ \Box\psi) \ \rightarrow \ \Box(\Diamond\varphi \ \rightarrow \ \psi) \ >$ 
3416     using "4" "Hypothetical Syllogism" by blast
3417 next
3418   AOT_modally_strict {
3419     AOT_have < $\Box(\Diamond\varphi \ \rightarrow \ \psi) \ \rightarrow \ (\varphi \ \rightarrow \ \Box\psi) \ >$ 
3420       by (meson "B" "Hypothetical Syllogism" " $\rightarrow I$ " "qml:1"[axiom_inst])
3421   }
3422   AOT_hence < $\Box\Box(\Diamond\varphi \ \rightarrow \ \psi) \ \rightarrow \ \Box(\varphi \ \rightarrow \ \Box\psi) \ >$ 
3423     by (rule RM)
3424   AOT_thus < $\Box(\Diamond\varphi \ \rightarrow \ \psi) \ \rightarrow \ \Box(\varphi \ \rightarrow \ \Box\psi) \ >$ 
3425     using "4" "Hypothetical Syllogism" by blast
3426 qed
3427
3428 AOT_theorem "derived-S5-rules:1": (165.1)
3429   assumes < $\Gamma \ \vdash_{\Box} \ \Diamond\varphi \ \rightarrow \ \psi \ >$ 
3430   shows < $\Box\Gamma \ \vdash_{\Box} \ \varphi \ \rightarrow \ \Box\psi \ >$ 
3431 proof -
3432   AOT_have < $\Box\Gamma \ \vdash_{\Box} \ \Box\Diamond\varphi \ \rightarrow \ \Box\psi \ >$ 
3433     using assms by (rule "RM:1[pre]m")
3434   AOT_thus < $\Box\Gamma \ \vdash_{\Box} \ \varphi \ \rightarrow \ \Box\psi \ >$ 
3435     using "B" "Hypothetical Syllogism" by blast
3436 qed
3437
3438 AOT_theorem "derived-S5-rules:2": (165.2)
3439   assumes < $\Gamma \ \vdash_{\Box} \ \varphi \ \rightarrow \ \Box\psi \ >$ 
3440   shows < $\Box\Gamma \ \vdash_{\Box} \ \Diamond\varphi \ \rightarrow \ \psi \ >$ 
3441 proof -
3442   AOT_have < $\Box\Gamma \ \vdash_{\Box} \ \Diamond\varphi \ \rightarrow \ \Diamond\Box\psi \ >$ 
3443     using assms by (rule "RM:2[pre]m")
3444   AOT_thus < $\Box\Gamma \ \vdash_{\Box} \ \Diamond\varphi \ \rightarrow \ \psi \ >$ 
3445     using "B $\Diamond$ " "Hypothetical Syllogism" by blast
3446 qed
3447
3448 AOT_theorem "BFs:1": < $\forall\alpha \ \Box\varphi\{\alpha\} \ \rightarrow \ \Box\forall\alpha \ \varphi\{\alpha\} \ >$  (166.1)
3449 proof -
3450   AOT_modally_strict {
3451     AOT_have < $\Diamond\forall\alpha \ \Box\varphi\{\alpha\} \ \rightarrow \ \Diamond\Box\varphi\{\alpha\} \ >$  for  $\alpha$ 
3452       using "cqt-orig:3" by (rule "RM $\Diamond$ ")
3453     AOT_hence < $\Diamond\forall\alpha \ \Box\varphi\{\alpha\} \ \rightarrow \ \forall\alpha \ \varphi\{\alpha\} \ >$ 
3454       using "B $\Diamond$ " " $\forall I$ " " $\rightarrow E$ " " $\rightarrow I$ " by metis
3455   }
3456   thus ?thesis
3457     using "derived-S5-rules:1" by blast
3458 qed
3459 lemmas "BF" = "BFs:1"
3460
3461 AOT_theorem "BFs:2": < $\Box\forall\alpha \ \varphi\{\alpha\} \ \rightarrow \ \forall\alpha \ \Box\varphi\{\alpha\} \ >$  (166.2)
3462 proof -
3463   AOT_have < $\Box\forall\alpha \ \varphi\{\alpha\} \ \rightarrow \ \Box\varphi\{\alpha\} \ >$  for  $\alpha$ 

```

```

3464     using RM "cqt-orig:3" by metis
3465     thus ?thesis
3466     using "cqt-orig:2"[THEN "→E"] "∀I" by metis
3467 qed
3468 lemmas "CBF" = "BFs:2"
3469
3470 AOT_theorem "BFs:3": <◇∃α φ{α} → ∃α ◇φ{α}> (166.3)
3471 proof(rule "→I")
3472   AOT_modally_strict {
3473     AOT_have <□∀α ¬φ{α} ≡ ∀α □¬φ{α}>
3474     using BF CBF "≡I" by blast
3475   } note ∅ = this
3476
3477   AOT_assume <◇∃α φ{α}>
3478   AOT_hence <¬□¬(∃α φ{α})>
3479   using "≡dfE" "conventions:5" by blast
3480   AOT_hence <¬□∀α ¬φ{α}>
3481   apply (AOT_subst <∀α ¬φ{α}> <¬(∃α φ{α})>)
3482   using "≡dfI" "conventions:3" "conventions:4" "&I"
3483   "contraposition:2" "cqt-further:4"
3484   "df-rules-formulas[3]" by blast
3485   AOT_hence <¬∀α □¬φ{α}>
3486   apply (AOT_subst (reverse) <∀α □¬φ{α}> <□∀α ¬φ{α}>)
3487   using ∅ by blast
3488   AOT_hence <¬∀α ¬¬□¬φ{α}>
3489   by (AOT_subst (reverse) <¬¬□¬φ{α}> <□¬φ{α}> for: α)
3490   (simp add: "oth-class-taut:3:b")
3491   AOT_hence <∃α ¬□¬φ{α}>
3492   by (rule "conventions:4"[THEN "≡dfI"])
3493   AOT_thus <∃α ◇φ{α}>
3494   using "conventions:5"[THEN "≡dfI"]
3495   by (AOT_subst <◇φ{α}> <¬□¬φ{α}> for: α)
3496 qed
3497 lemmas "BF◇" = "BFs:3"
3498
3499 AOT_theorem "BFs:4": <∃α ◇φ{α} → ◇∃α φ{α}> (166.4)
3500 proof(rule "→I")
3501   AOT_assume <∃α ◇φ{α}>
3502   AOT_hence <¬∀α ¬◇φ{α}>
3503   using "conventions:4"[THEN "≡dfE"] by blast
3504   AOT_hence <¬∀α □¬φ{α}>
3505   using "KBasic2:1"
3506   by (AOT_subst <□¬φ{α}> <¬◇φ{α}> for: α)
3507   moreover AOT_have <∀α □¬φ{α} ≡ □∀α ¬φ{α}>
3508   using "≡I" "BF" "CBF" by metis
3509   ultimately AOT_have 1: <¬□∀α ¬φ{α}>
3510   using "≡E"(3) by blast
3511   AOT_show <◇∃α φ{α}>
3512   apply (rule "conventions:5"[THEN "≡dfI"])
3513   apply (AOT_subst <∃α φ{α}> <¬∀α ¬φ{α}>)
3514   apply (simp add: "conventions:4" "≡dfI")
3515   apply (AOT_subst <¬¬∀α ¬φ{α}> <∀α ¬φ{α}>)
3516   by (auto simp: 1 "≡I" "useful-tautologies:1" "useful-tautologies:2")
3517 qed
3518 lemmas "CBF◇" = "BFs:4"
3519
3520 AOT_theorem "sign-S5-thm:1": <∃α □φ{α} → □∃α φ{α}> (167.1)
3521 proof(rule "→I")
3522   AOT_assume <∃α □φ{α}>
3523   then AOT_obtain α where <□φ{α}> using "∃E" by metis
3524   moreover AOT_have <□α↓>
3525   by (simp add: "ex:1:a" "rule-ui:2[const_var]" RN)
3526   moreover AOT_have <□φ{τ}, □τ↓ ⊢ □∃α φ{α}> for τ

```

```

3527   proof -
3528     AOT_have <math>\varphi\{\tau\}, \tau\downarrow \vdash_{\square} \exists\alpha \varphi\{\alpha}> using "existential:1" by blast
3529     AOT_thus <math>\square\varphi\{\tau\}, \square\tau\downarrow \vdash_{\square} \square\exists\alpha \varphi\{\alpha}>
3530     using "RN[prem]"["where  $\Gamma = \{\varphi \tau, \langle\tau\downarrow\rangle\}$ ", simplified] by blast
3531   qed
3532   ultimately AOT_show <math>\square\exists\alpha \varphi\{\alpha}> by blast
3533 qed
3534 lemmas Buridan = "sign-S5-thm:1"
3535
3536 AOT_theorem "sign-S5-thm:2": <math>\langle\Diamond\forall\alpha \varphi\{\alpha\} \rightarrow \forall\alpha \Diamond\varphi\{\alpha\}> (167.2)
3537 proof -
3538   AOT_have <math>\langle\forall\alpha (\Diamond\forall\alpha \varphi\{\alpha\} \rightarrow \Diamond\varphi\{\alpha\})>
3539   by (simp add: "RM $\Diamond$ " "cqt-orig:3" " $\forall I$ ")
3540   AOT_thus <math>\langle\Diamond\forall\alpha \varphi\{\alpha\} \rightarrow \forall\alpha \Diamond\varphi\{\alpha\}>
3541   using " $\forall E$ "(4) " $\forall I$ " " $\rightarrow E$ " " $\rightarrow I$ " by metis
3542 qed
3543 lemmas "Buridan $\Diamond$ " = "sign-S5-thm:2"
3544
3545 AOT_theorem "sign-S5-thm:3": (167.3)
3546 <math>\langle\Diamond\exists\alpha (\varphi\{\alpha\} \& \psi\{\alpha\}) \rightarrow \Diamond(\exists\alpha \varphi\{\alpha\} \& \exists\alpha \psi\{\alpha\})>
3547 apply (rule "RM:2")
3548 by (metis (no_types, lifting) " $\exists E$ " " $\& I$ " " $\& E$ "(1) " $\& E$ "(2) " $\rightarrow I$ " " $\exists I$ "(2))
3549
3550 AOT_theorem "sign-S5-thm:4": <math>\langle\Diamond\exists\alpha (\varphi\{\alpha\} \& \psi\{\alpha\}) \rightarrow \Diamond\exists\alpha \varphi\{\alpha\}> (167.4)
3551 apply (rule "RM:2")
3552 by (meson "instantiation" " $\& E$ "(1) " $\rightarrow I$ " " $\exists I$ "(2))
3553
3554 AOT_theorem "sign-S5-thm:5": (167.5)
3555 <math>\langle(\square\forall\alpha (\varphi\{\alpha\} \rightarrow \psi\{\alpha\}) \& \square\forall\alpha (\psi\{\alpha\} \rightarrow \chi\{\alpha\})) \rightarrow \square\forall\alpha (\varphi\{\alpha\} \rightarrow \chi\{\alpha\})>
3556 proof -
3557   {
3558     fix  $\varphi'$   $\psi'$   $\chi'$ 
3559     AOT_assume <math>\langle\vdash_{\square} \varphi' \& \psi' \rightarrow \chi'>
3560     AOT_hence <math>\langle\square\varphi' \& \square\psi' \rightarrow \square\chi'>
3561     using "RN[prem]"["where  $\Gamma = \{\varphi', \psi'\}$ "] apply simp
3562     using " $\& E$ " " $\& I$ " " $\rightarrow E$ " " $\rightarrow I$ " by metis
3563   } note R = this
3564   show ?thesis by (rule R; fact AOT)
3565 qed
3566
3567 AOT_theorem "sign-S5-thm:6": (167.6)
3568 <math>\langle(\square\forall\alpha (\varphi\{\alpha\} \equiv \psi\{\alpha\}) \& \square\forall\alpha (\psi\{\alpha\} \equiv \chi\{\alpha\})) \rightarrow \square\forall\alpha (\varphi\{\alpha\} \equiv \chi\{\alpha\})>
3569 proof -
3570   {
3571     fix  $\varphi'$   $\psi'$   $\chi'$ 
3572     AOT_assume <math>\langle\vdash_{\square} \varphi' \& \psi' \rightarrow \chi'>
3573     AOT_hence <math>\langle\square\varphi' \& \square\psi' \rightarrow \square\chi'>
3574     using "RN[prem]"["where  $\Gamma = \{\varphi', \psi'\}$ "] apply simp
3575     using " $\& E$ " " $\& I$ " " $\rightarrow E$ " " $\rightarrow I$ " by metis
3576   } note R = this
3577   show ?thesis by (rule R; fact AOT)
3578 qed
3579
3580 AOT_theorem "exist-nec2:1": <math>\langle\Diamond\tau\downarrow \rightarrow \tau\downarrow> (168.1)
3581 using "B $\Diamond$ " "RM $\Diamond$ " "Hypothetical Syllogism" "exist-nec" by blast
3582
3583 AOT_theorem "exists-nec2:2": <math>\langle\Diamond\tau\downarrow \equiv \square\tau\downarrow>
3584 by (meson "Act-Sub:3" "Hypothetical Syllogism" "exist-nec"
3585 "exist-nec2:1" " $\equiv I$ " "nec-imp-act")
3586
3587 AOT_theorem "exists-nec2:3": <math>\langle\neg\tau\downarrow \rightarrow \square\neg\tau\downarrow>
3588 using "KBasic2:1" " $\rightarrow I$ " "exist-nec2:1" " $\equiv E$ "(2) "modus-tollens:1" by blast
3589

```

```

3590 AOT_theorem "exists-nec2:4": <math>\langle \Diamond \neg \tau \downarrow \equiv \Box \neg \tau \downarrow \rangle
3591   by (metis "Act-Sub:3" "KBasic:12" "\toI" "exist-nec" "exists-nec2:3"
3592       "\equivI" "\equivE"(4) "nec-imp-act" "reductio-aa:1")
3593
3594 AOT_theorem "id-nec2:1": <math>\langle \Diamond \alpha = \beta \rightarrow \alpha = \beta \rangle \tag{169.1}
3595   using "B\Diamond" "RM\Diamond" "Hypothetical Syllogism" "id-nec:1" by blast
3596
3597 AOT_theorem "id-nec2:2": <math>\langle \alpha \neq \beta \rightarrow \Box \alpha \neq \beta \rangle \tag{169.2}
3598   apply (AOT_subst <math>\langle \alpha \neq \beta \rangle \langle \neg(\alpha = \beta) \rangle)
3599   using "--infix"[THEN "\equivDf"] apply blast
3600   using "KBasic2:1" "\toI" "id-nec2:1" "\equivE"(2) "modus-tollens:1" by blast
3601
3602 AOT_theorem "id-nec2:3": <math>\langle \Diamond \alpha \neq \beta \rightarrow \alpha \neq \beta \rangle \tag{169.3}
3603   apply (AOT_subst <math>\langle \alpha \neq \beta \rangle \langle \neg(\alpha = \beta) \rangle)
3604   using "--infix"[THEN "\equivDf"] apply blast
3605   by (metis "KBasic:11" "\toI" "id-nec:2" "\equivE"(3) "reductio-aa:2" "\toE")
3606
3607 AOT_theorem "id-nec2:4": <math>\langle \Diamond \alpha = \beta \rightarrow \Box \alpha = \beta \rangle \tag{169.4}
3608   using "Hypothetical Syllogism" "id-nec2:1" "id-nec:1" by blast
3609
3610 AOT_theorem "id-nec2:5": <math>\langle \Diamond \alpha \neq \beta \rightarrow \Box \alpha \neq \beta \rangle \tag{169.5}
3611   using "id-nec2:3" "id-nec2:2" "\toI" "\toE" by metis
3612
3613 AOT_theorem "sc-eq-box-box:1": <math>\langle \Box(\varphi \rightarrow \Box \varphi) \equiv (\Diamond \varphi \rightarrow \Box \varphi) \rangle \tag{170.1}
3614   apply (rule "\equivI"; rule "\toI")
3615   using "KBasic:13" "5\Diamond" "Hypothetical Syllogism" "\toE" apply blast
3616   by (metis "KBasic2:1" "KBasic:1" "KBasic:2" "S5Basic:13" "\equivE"(2)
3617       "raa-cor:5" "\toE")
3618
3619 AOT_theorem "sc-eq-box-box:2": <math>\langle (\Box(\varphi \rightarrow \Box \varphi) \vee (\Diamond \varphi \rightarrow \Box \varphi)) \rightarrow (\Diamond \varphi \equiv \Box \varphi) \rangle \tag{170.2}
3620   by (metis "Act-Sub:3" "KBasic:13" "5\Diamond" "\veeE"(2) "\toI" "\equivI"
3621       "nec-imp-act" "raa-cor:2" "\toE")
3622
3623 AOT_theorem "sc-eq-box-box:3": <math>\langle \Box(\varphi \rightarrow \Box \varphi) \rightarrow (\neg \Box \varphi \equiv \Box \neg \varphi) \rangle \tag{170.3}
3624 proof (rule "\toI"; rule "\equivI"; rule "\toI")
3625   AOT_assume <math>\langle \Box(\varphi \rightarrow \Box \varphi) \rangle
3626   AOT_hence <math>\langle \Diamond \varphi \rightarrow \Box \varphi \rangle using "sc-eq-box-box:1" "\equivE" by blast
3627   moreover AOT_assume <math>\langle \neg \Box \varphi \rangle
3628   ultimately AOT_have <math>\langle \neg \Diamond \varphi \rangle
3629     using "modus-tollens:1" by blast
3630   AOT_thus <math>\langle \Box \neg \varphi \rangle
3631     using "KBasic2:1" "\equivE"(2) by blast
3632 next
3633   AOT_assume <math>\langle \Box(\varphi \rightarrow \Box \varphi) \rangle
3634   moreover AOT_assume <math>\langle \Box \neg \varphi \rangle
3635   ultimately AOT_show <math>\langle \neg \Box \varphi \rangle
3636     using "modus-tollens:1" "qml:2"[axiom_inst] "\toE" by blast
3637 qed
3638
3639 AOT_theorem "sc-eq-box-box:4": \tag{170.4}
3640   <math>\langle (\Box(\varphi \rightarrow \Box \varphi) \ \& \ \Box(\psi \rightarrow \Box \psi)) \rightarrow ((\Box \varphi \equiv \Box \psi) \rightarrow \Box(\varphi \equiv \psi)) \rangle
3641 proof (rule "\toI"; rule "\toI")
3642   AOT_assume  $\vartheta$ : <math>\langle \Box(\varphi \rightarrow \Box \varphi) \ \& \ \Box(\psi \rightarrow \Box \psi) \rangle
3643   AOT_assume  $\xi$ : <math>\langle \Box \varphi \equiv \Box \psi \rangle
3644   AOT_hence <math>\langle (\Box \varphi \ \& \ \Box \psi) \vee (\neg \Box \varphi \ \& \ \neg \Box \psi) \rangle
3645     using "\equivE"(4) "oth-class-taut:4:g" "raa-cor:3" by blast
3646   moreover {
3647     AOT_assume <math>\langle \Box \varphi \ \& \ \Box \psi \rangle
3648     AOT_hence <math>\langle \Box(\varphi \equiv \psi) \rangle
3649     using "KBasic:3" "KBasic:8" "\equivE"(2) "vdash-properties:10" by blast
3650   }
3651   moreover {
3652     AOT_assume <math>\langle \neg \Box \varphi \ \& \ \neg \Box \psi \rangle

```

```

3653   moreover AOT_have <¬□φ ≡ □¬φ> and <¬□ψ ≡ □¬ψ>
3654     using φ "Conjunction Simplification"(1,2)
3655     "sc-eq-box-box:3" "→E" by metis+
3656   ultimately AOT_have <□¬φ & □¬ψ>
3657     by (metis "&I" "Conjunction Simplification"(1,2)
3658         "≡E"(4) "modus-tollens:1" "raa-cor:3")
3659   AOT_hence <□(φ ≡ ψ)>
3660     using "KBasic:3" "KBasic:9" "≡E"(2) "→E" by blast
3661 }
3662 ultimately AOT_show <□(φ ≡ ψ)>
3663   using "∀E"(2) "reductio-aa:1" by blast
3664 qed
3665
3666 AOT_theorem "sc-eq-box-box:5":
3667   <(□(φ → □φ) & □(ψ → □ψ)) → □((φ ≡ ψ) → □(φ ≡ ψ))>
3668 proof (rule "→I")
3669   AOT_assume <(□(φ → □φ) & □(ψ → □ψ))>
3670   AOT_hence <□(□(φ → □φ) & □(ψ → □ψ))>
3671     using 4[THEN "→E"] "&E" "&I" "KBasic:3" "≡E"(2) by metis
3672   moreover AOT_have <□(□(φ → □φ) & □(ψ → □ψ)) → □((φ ≡ ψ) → □(φ ≡ ψ))>
3673 proof (rule RM; rule "→I"; rule "→I")
3674   AOT_modally_strict {
3675     AOT_assume A: <(□(φ → □φ) & □(ψ → □ψ))>
3676     AOT_hence <φ → □φ> and <ψ → □ψ>
3677       using "&E" "qml:2"[axiom_inst] "→E" by blast+
3678     moreover AOT_assume <φ ≡ ψ>
3679     ultimately AOT_have <□φ ≡ □ψ>
3680       using "→E" "qml:2"[axiom_inst] "≡E" "≡I" by meson
3681     moreover AOT_have <(□φ ≡ □ψ) → □(φ ≡ ψ)>
3682       using A "sc-eq-box-box:4" "→E" by blast
3683     ultimately AOT_show <□(φ ≡ ψ)> using "→E" by blast
3684   }
3685   qed
3686   ultimately AOT_show <□((φ ≡ ψ) → □(φ ≡ ψ))> using "→E" by blast
3687 qed
3688
3689 AOT_theorem "sc-eq-box-box:6": <□(φ → □φ) → ((φ → □ψ) → □(φ → ψ))>
3690 proof (rule "→I"; rule "→I"; rule "raa-cor:1")
3691   AOT_assume <¬□(φ → ψ)>
3692   AOT_hence <◇¬(φ → ψ)>
3693     by (metis "KBasic:11" "≡E"(1))
3694   AOT_hence <◇(φ & ¬ψ)>
3695     by (AOT_subst <φ & ¬ψ> <¬(φ → ψ)>)
3696     (meson "Commutativity of ≡" "≡E"(1) "oth-class-taut:1:b")
3697   AOT_hence <◇φ> and 2: <◇¬ψ>
3698     using "KBasic2:3"[THEN "→E"] "&E" by blast+
3699   moreover AOT_assume <□(φ → □φ)>
3700   ultimately AOT_have <□φ>
3701     by (metis "≡E"(1) "sc-eq-box-box:1" "→E")
3702   AOT_hence φ
3703     using "qml:2"[axiom_inst, THEN "→E"] by blast
3704   moreover AOT_assume <φ → □ψ>
3705   ultimately AOT_have <□ψ>
3706     using "→E" by blast
3707   moreover AOT_have <¬□ψ>
3708     using 2 "KBasic:12" "¬¬I" "intro-elim:3:d" by blast
3709   ultimately AOT_show <□ψ & ¬□ψ>
3710     using "&I" by blast
3711 qed
3712
3713 AOT_theorem "sc-eq-box-box:7": <□(φ → □φ) → ((φ →  $\mathcal{A}\psi$ ) →  $\mathcal{A}(\varphi \rightarrow \psi)$ )>
3714 proof (rule "→I"; rule "→I"; rule "raa-cor:1")
3715   AOT_assume <¬ $\mathcal{A}(\varphi \rightarrow \psi)$ >

```

```

3716 AOT_hence <math>\mathcal{A} \neg(\varphi \rightarrow \psi)>
3717   by (metis "Act-Basic:1" "\VE"(2))
3718 AOT_hence <math>\mathcal{A}(\varphi \ \& \ \neg\psi)>
3719   by (AOT_subst <math>\langle \varphi \ \& \ \neg\psi \rangle \langle \neg(\varphi \rightarrow \psi) \rangle)
3720     (meson "Commutativity of  $\equiv$ " "\equiv"(1) "oth-class-taut:1:b")
3721 AOT_hence <math>\mathcal{A}\varphi> \text{ and } 2: \langle \mathcal{A}\neg\psi \rangle
3722   using "Act-Basic:2"[THEN "\equiv"(1)] "\&E" by blast+
3723 AOT_hence <math>\langle \diamond\varphi \rangle
3724   by (metis "Act-Sub:3" "\rightarrowE")
3725 moreover AOT_assume <math>\langle \Box(\varphi \rightarrow \Box\varphi) \rangle
3726 ultimately AOT_have <math>\langle \Box\varphi \rangle
3727   by (metis "\equiv"(1) "sc-eq-box-box:1" "\rightarrowE")
3728 AOT_hence  $\varphi$ 
3729   using "qml:2"[axiom_inst, THEN "\rightarrowE"] by blast
3730 moreover AOT_assume <math>\langle \varphi \rightarrow \mathcal{A}\psi \rangle
3731 ultimately AOT_have <math>\langle \mathcal{A}\psi \rangle
3732   using "\rightarrowE" by blast
3733 moreover AOT_have <math>\langle \neg\mathcal{A}\psi \rangle
3734   using 2 by (meson "Act-Sub:1" "\equiv"(4) "raa-cor:3")
3735 ultimately AOT_show <math>\langle \mathcal{A}\psi \ \& \ \neg\mathcal{A}\psi \rangle
3736   using "\&I" by blast
3737 qed
3738
3739 AOT_theorem "sc-eq-fur:1": <math>\langle \diamond\mathcal{A}\varphi \equiv \Box\mathcal{A}\varphi \rangle \tag{172.1}
3740   using "Act-Basic:6" "Act-Sub:4" "\equiv"(6) by blast
3741
3742 AOT_theorem "sc-eq-fur:2": <math>\langle \Box(\varphi \rightarrow \Box\varphi) \rightarrow (\mathcal{A}\varphi \equiv \varphi) \rangle \tag{172.2}
3743   by (metis "B\diamond" "Act-Sub:3" "KBasic:13" "T\diamond" "Hypothetical Syllogism"
3744     "\rightarrowI" "\equivI" "nec-imp-act")
3745
3746 AOT_theorem "sc-eq-fur:3": \tag{172.3}
3747   <math>\langle \Box\forall x (\varphi\{x\} \rightarrow \Box\varphi\{x\}) \rightarrow (\exists!x \varphi\{x\} \rightarrow \ulcorner \varphi\{x\} \urcorner) \rangle
3748 proof (rule "\rightarrowI"; rule "\rightarrowI")
3749   AOT_assume <math>\langle \Box\forall x (\varphi\{x\} \rightarrow \Box\varphi\{x\}) \rangle
3750   AOT_hence A: <math>\langle \forall x \Box(\varphi\{x\} \rightarrow \Box\varphi\{x\}) \rangle
3751     using CBF "\rightarrowE" by blast
3752   AOT_assume <math>\langle \exists!x \varphi\{x\} \rangle
3753   then AOT_obtain a where a_def: <math>\langle \varphi\{a\} \ \& \ \forall y (\varphi\{y\} \rightarrow y = a) \rangle
3754     using "\existsE"[rotated 1, OF "uniqueness:1"[THEN "\equiv_{df}E"]] by blast
3755   moreover AOT_have <math>\langle \Box\varphi\{a\} \rangle
3756     using calculation A "\VE"(2) "qml:2"[axiom_inst] "\rightarrowE" "\&E"(1) by blast
3757   AOT_hence <math>\langle \mathcal{A}\varphi\{a\} \rangle
3758     using "nec-imp-act" "\rightarrowE" by blast
3759   moreover AOT_have <math>\langle \forall y (\mathcal{A}\varphi\{y\} \rightarrow y = a) \rangle
3760 proof (rule "\forallI"; rule "\rightarrowI")
3761   fix b
3762   AOT_assume <math>\langle \mathcal{A}\varphi\{b\} \rangle
3763   AOT_hence <math>\langle \diamond\varphi\{b\} \rangle
3764     using "Act-Sub:3" "\rightarrowE" by blast
3765   moreover {
3766     AOT_have <math>\langle \Box(\varphi\{b\} \rightarrow \Box\varphi\{b\}) \rangle
3767       using A "\VE"(2) by blast
3768     AOT_hence <math>\langle \diamond\varphi\{b\} \rightarrow \Box\varphi\{b\} \rangle
3769       using "KBasic:13" "5\diamond" "Hypothetical Syllogism" "\rightarrowE" by blast
3770   }
3771   ultimately AOT_have <math>\langle \Box\varphi\{b\} \rangle
3772     using "\rightarrowE" by blast
3773   AOT_hence <math>\langle \varphi\{b\} \rangle
3774     using "qml:2"[axiom_inst] "\rightarrowE" by blast
3775   AOT_thus <math>\langle b = a \rangle
3776     using a_def[THEN "\&E"(2)] "\VE"(2) "\rightarrowE" by blast
3777 qed
3778 ultimately AOT_have <math>\langle \mathcal{A}\varphi\{a\} \ \& \ \forall y (\mathcal{A}\varphi\{y\} \rightarrow y = a) \rangle

```

```

3779   using "&I" by blast
3780   AOT_hence <∃x (Aφ{x} & ∀y (Aφ{y} → y = x))>
3781   using "∃I" by fast
3782   AOT_hence <∃!x Aφ{x}>
3783   using "uniqueness:1"[THEN "≡dfI"] by fast
3784   AOT_thus <∃x φ{x}↓>
3785   using "actual-desc:1"[THEN "≡E"(2)] by blast
3786 qed
3787
3788 AOT_theorem "sc-eq-fur:4":
3789 <□∀x (φ{x} → □φ{x}) → (x = ∃x φ{x} ≡ (φ{x} & ∀z (φ{z} → z = x)))>
3790 proof (rule "→I")
3791   AOT_assume <□∀x (φ{x} → □φ{x})>
3792   AOT_hence <∀x □(φ{x} → □φ{x})>
3793   using CBF "→E" by blast
3794   AOT_hence A: <Aφ{α} ≡ φ{α}> for α
3795   using "sc-eq-fur:2" "∀E" "→E" by fast
3796   AOT_show <x = ∃x φ{x} ≡ (φ{x} & ∀z (φ{z} → z = x))>
3797   proof (rule "≡I"; rule "→I")
3798     AOT_assume <x = ∃x φ{x}>
3799     AOT_hence B: <Aφ{x} & ∀z (Aφ{z} → z = x)>
3800     using "nec-hintikka-scheme"[THEN "≡E"(1)] by blast
3801     AOT_show <φ{x} & ∀z (φ{z} → z = x)>
3802     proof (rule "&I"; (rule "∀I"; rule "→I")?)
3803       AOT_show <φ{x}>
3804       using A B[THEN "&E"(1)] "≡E"(1) by blast
3805     next
3806       AOT_show <z = x> if <φ{z}> for z
3807       using that B[THEN "&E"(2)] "∀E"(2) "→E" A[THEN "≡E"(2)] by blast
3808     qed
3809   next
3810     AOT_assume B: <φ{x} & ∀z (φ{z} → z = x)>
3811     AOT_have <Aφ{x} & ∀z (Aφ{z} → z = x)>
3812     proof (rule "&I"; (rule "∀I"; rule "→I")?)
3813       AOT_show <Aφ{x}>
3814       using B[THEN "&E"(1)] A[THEN "≡E"(2)] by blast
3815     next
3816       AOT_show <b = x> if <Aφ{b}> for b
3817       using A[THEN "≡E"(1)] that
3818       B[THEN "&E"(2), THEN "∀E"(2), THEN "→E"] by blast
3819     qed
3820     AOT_thus <x = ∃x φ{x}>
3821     using "nec-hintikka-scheme"[THEN "≡E"(2)] by blast
3822   qed
3823 qed
3824
3825 AOT_theorem "id-act:1": <α = β ≡ Aα = β>
3826 by (meson "Act-Sub:3" "Hypothetical Syllogism"
3827 "id-nec2:1" "id-nec:2" "≡I" "nec-imp-act")
3828
3829 AOT_theorem "id-act:2": <α ≠ β ≡ Aα ≠ β>
3830 proof (AOT_subst <α ≠ β> <¬(α = β)>)
3831   AOT_modally_strict {
3832     AOT_show <α ≠ β ≡ ¬(α = β)>
3833     by (simp add: "=-infix" "≡Df")
3834   }
3835 next
3836   AOT_show <¬(α = β) ≡ A¬(α = β)>
3837   proof (safe intro!: "≡I" "→I")
3838     AOT_assume <¬α = β>
3839     AOT_hence <¬Aα = β> using "id-act:1" "≡E"(3) by blast
3840     AOT_thus <A¬α = β>
3841     using "¬¬E" "Act-Sub:1" "≡E"(3) by blast

```

```

3842 next
3843   AOT_assume <A¬α = β>
3844   AOT_hence <¬Aα = β>
3845     using "¬¬I" "Act-Sub:1" "≡E"(4) by blast
3846   AOT_thus <¬α = β>
3847     using "id-act:1" "≡E"(4) by blast
3848 qed
3849 qed
3850
3851 AOT_theorem "A-Exists:1": <A∃!α φ{α} ≡ ∃!α Aφ{α}> (174.1)
3852 proof -
3853   AOT_have <A∃!α φ{α} ≡ A∃α∀β (φ{β} ≡ β = α)>
3854     by (AOT_subst <∃!α φ{α}> <∃α∀β (φ{β} ≡ β = α)>)
3855       (auto simp add: "oth-class-taut:3:a" "uniqueness:2")
3856   also AOT_have <... ≡ ∃α A∀β (φ{β} ≡ β = α)>
3857     by (simp add: "Act-Basic:10")
3858   also AOT_have <... ≡ ∃α∀β A(φ{β} ≡ β = α)>
3859     by (AOT_subst <A∀β (φ{β} ≡ β = α)> <∀β A(φ{β} ≡ β = α)> for: α)
3860       (auto simp: "logic-actual-nec:3"[axiom_inst] "oth-class-taut:3:a")
3861   also AOT_have <... ≡ ∃α∀β (Aφ{β} ≡ Aβ = α)>
3862     by (AOT_subst (reverse) <Aφ{β} ≡ Aβ = α>
3863         <A(φ{β} ≡ β = α)> for: α β :: 'a)
3864       (auto simp: "Act-Basic:5" "cqt-further:7")
3865   also AOT_have <... ≡ ∃α∀β (Aφ{β} ≡ β = α)>
3866     by (AOT_subst (reverse) <Aβ = α> <β = α> for: α β :: 'a)
3867       (auto simp: "id-act:1" "cqt-further:7")
3868   also AOT_have <... ≡ ∃!α Aφ{α}>
3869     using "uniqueness:2" "Commutativity of ≡"[THEN "≡E"(1)] by fast
3870   finally show ?thesis.
3871 qed
3872
3873 AOT_theorem "A-Exists:2": <ιx φ{x}↓ ≡ A∃!x φ{x}> (174.2)
3874 by (AOT_subst <A∃!x φ{x}> <∃!x Aφ{x}>)
3875   (auto simp: "actual-desc:1" "A-Exists:1")
3876
3877 AOT_theorem "id-act-desc:1": <ιx (x = y)↓> (175.1)
3878 proof(rule "existence:1"[THEN "≡defI"]; rule "∃I")
3879   AOT_show <[λx E!x → E!x]ιx (x = y)>
3880   proof (rule "russell-axiom[exe,1].nec-russell-axiom"[THEN "≡E"(2)];
3881     rule "∃I"; (rule "&I")+
3882     AOT_show <Ay = y> by (simp add: "RA[2]" "id-eq:1")
3883   next
3884     AOT_show <∀z (Az = y → z = y)>
3885     apply (rule "∀I")
3886     using "id-act:1"[THEN "≡E"(2)] "→I" by blast
3887   next
3888     AOT_show <[λx E!x → E!x]y>
3889     proof (rule "lambda-predicates:2"[axiom_inst, THEN "→E", THEN "≡E"(2)])
3890       AOT_show <[λx E!x → E!x]↓>
3891       by "cqt:2[lambda]"
3892     next
3893       AOT_show <E!y → E!y>
3894       by (simp add: "if-p-then-p")
3895     qed
3896   qed
3897 next
3898   AOT_show <[λx E!x → E!x]↓>
3899   by "cqt:2[lambda]"
3900 qed
3901
3902 AOT_theorem "id-act-desc:2": <y = ιx (x = y)> (175.2)
3903 by (rule descriptions[axiom_inst, THEN "≡E"(2)];
3904   rule "∀I"; rule "id-act:1"[symmetric])

```



```

3905
3906 AOT_theorem "pre-en-eq:1[1]": <x1[F] → □x1[F]> (176.1)
3907   by (simp add: encoding "vdash-properties:1[2]")
3908
3909 AOT_theorem "pre-en-eq:1[2]": <x1x2[F] → □x1x2[F]> (176.1)
3910 proof (rule "→I")
3911   AOT_assume <x1x2[F]>
3912   AOT_hence <x1[λy [F]yx2] and <x2[λy [F]x1y]>
3913     using "nary-encoding[2]"[axiom_inst, THEN "≡E"(1)] "&E" by blast+
3914   moreover AOT_have <[λy [F]yx2]↓> by "cqt:2"
3915   moreover AOT_have <[λy [F]x1y]↓> by "cqt:2"
3916   ultimately AOT_have <□x1[λy [F]yx2] and <□x2[λy [F]x1y]>
3917     using encoding[axiom_inst, unvarify F] "→E" "&I" by blast+
3918   note A = this
3919   AOT_hence <□(x1[λy [F]yx2] & x2[λy [F]x1y])>
3920     using "KBasic:3"[THEN "≡E"(2)] "&I" by blast
3921   AOT_thus <□x1x2[F]>
3922     by (rule "nary-encoding[2]"[axiom_inst, THEN RN,
3923       THEN "KBasic:6"[THEN "→E"],
3924       THEN "≡E"(2)])
3925 qed
3926
3927 AOT_theorem "pre-en-eq:1[3]": <x1x2x3[F] → □x1x2x3[F]> (176.1)
3928 proof (rule "→I")
3929   AOT_assume <x1x2x3[F]>
3930   AOT_hence <x1[λy [F]yx2x3}] and <x2[λy [F]x1yx3}] and <x3[λy [F]x1x2y]>
3931     and <x2[λy [F]x1yx3}] and <x3[λy [F]x1x2y]>
3932     using "nary-encoding[3]"[axiom_inst, THEN "≡E"(1)] "&E" by blast+
3933   moreover AOT_have <[λy [F]yx2x3}]↓> by "cqt:2"
3934   moreover AOT_have <[λy [F]x1yx3}]↓> by "cqt:2"
3935   moreover AOT_have <[λy [F]x1x2y]↓> by "cqt:2"
3936   ultimately AOT_have <□x1[λy [F]yx2x3}] and <□x2[λy [F]x1yx3}] and <□x3[λy [F]x1x2y]>
3937     and <□x2[λy [F]x1yx3}] and <□x3[λy [F]x1x2y]>
3938     using encoding[axiom_inst, unvarify F] "→E" by blast+
3939   note A = this
3940   AOT_have B: <□(x1[λy [F]yx2x3}] & x2[λy [F]x1yx3}] & x3[λy [F]x1x2y])>
3941     by (rule "KBasic:3"[THEN "≡E"(2)] "&I" A)+
3942   AOT_thus <□x1x2x3[F]>
3943     by (rule "nary-encoding[3]"[axiom_inst, THEN RN,
3944       THEN "KBasic:6"[THEN "→E"], THEN "≡E"(2)])
3945 qed
3946
3947 AOT_theorem "pre-en-eq:1[4]": <x1x2x3x4[F] → □x1x2x3x4[F]> (176.1)
3948 proof (rule "→I")
3949   AOT_assume <x1x2x3x4[F]>
3950   AOT_hence <x1[λy [F]yx2x3x4}] and <x2[λy [F]x1yx3x4}] and <x3[λy [F]x1x2yx4}] and <x4[λy [F]x1x2x3y]>
3951     and <x2[λy [F]x1yx3x4}] and <x3[λy [F]x1x2yx4}] and <x4[λy [F]x1x2x3y]>
3952     using "nary-encoding[4]"[axiom_inst, THEN "≡E"(1)] "&E" by metis+
3953   moreover AOT_have <[λy [F]yx2x3x4}]↓> by "cqt:2"
3954   moreover AOT_have <[λy [F]x1yx3x4}]↓> by "cqt:2"
3955   moreover AOT_have <[λy [F]x1x2yx4}]↓> by "cqt:2"
3956   moreover AOT_have <[λy [F]x1x2x3y]↓> by "cqt:2"
3957   ultimately AOT_have <□x1[λy [F]yx2x3x4}] and <□x2[λy [F]x1yx3x4}] and <□x3[λy [F]x1x2yx4}] and <□x4[λy [F]x1x2x3y]>
3958     and <□x2[λy [F]x1yx3x4}] and <□x3[λy [F]x1x2yx4}] and <□x4[λy [F]x1x2x3y]>
3959     using "→E" encoding[axiom_inst, unvarify F] by blast+
3960   note A = this
3961   AOT_have B: <□(x1[λy [F]yx2x3x4}] &

```

```

3978           x2 [λy [F] x1 y x3 x4] &
3979           x3 [λy [F] x1 x2 y x4] &
3980           x4 [λy [F] x1 x2 x3 y]) >
3971   by (rule "KBasic:3"[THEN "≡E"(2)] "&I" A)+
3972   AOT_thus <□x1 x2 x3 x4 [F]>
3973   by (rule "nary-encoding[4]" [axiom_inst, THEN RN,
3974         THEN "KBasic:6"[THEN "→E"], THEN "≡E"(2)])
3975 qed
3976
3977 AOT_theorem "pre-en-eq:2[1]": <¬x1 [F] → □¬x1 [F]> (176.2)
3978 proof (rule "→I"; rule "raa-cor:1")
3979   AOT_assume <¬□¬x1 [F]>
3980   AOT_hence <◇x1 [F]>
3981   by (rule "conventions:5"[THEN "≡dfI"])
3982   AOT_hence <x1 [F]>
3983   by (rule "S5Basic:13"[THEN "≡E"(1), OF "pre-en-eq:1[1]"[THEN RN],
3984       THEN "qml:2"[axiom_inst, THEN "→E"], THEN "→E"])
3985   moreover AOT_assume <¬x1 [F]>
3986   ultimately AOT_show <x1 [F] & ¬x1 [F]> by (rule "&I")
3987 qed
3988 AOT_theorem "pre-en-eq:2[2]": <¬x1 x2 [F] → □¬x1 x2 [F]> (176.2)
3989 proof (rule "→I"; rule "raa-cor:1")
3990   AOT_assume <¬□¬x1 x2 [F]>
3991   AOT_hence <◇x1 x2 [F]>
3992   by (rule "conventions:5"[THEN "≡dfI"])
3993   AOT_hence <x1 x2 [F]>
3994   by (rule "S5Basic:13"[THEN "≡E"(1), OF "pre-en-eq:1[2]"[THEN RN],
3995       THEN "qml:2"[axiom_inst, THEN "→E"], THEN "→E"])
3996   moreover AOT_assume <¬x1 x2 [F]>
3997   ultimately AOT_show <x1 x2 [F] & ¬x1 x2 [F]> by (rule "&I")
3998 qed
3999
4000 AOT_theorem "pre-en-eq:2[3]": <¬x1 x2 x3 [F] → □¬x1 x2 x3 [F]> (176.2)
4001 proof (rule "→I"; rule "raa-cor:1")
4002   AOT_assume <¬□¬x1 x2 x3 [F]>
4003   AOT_hence <◇x1 x2 x3 [F]>
4004   by (rule "conventions:5"[THEN "≡dfI"])
4005   AOT_hence <x1 x2 x3 [F]>
4006   by (rule "S5Basic:13"[THEN "≡E"(1), OF "pre-en-eq:1[3]"[THEN RN],
4007       THEN "qml:2"[axiom_inst, THEN "→E"], THEN "→E"])
4008   moreover AOT_assume <¬x1 x2 x3 [F]>
4009   ultimately AOT_show <x1 x2 x3 [F] & ¬x1 x2 x3 [F]> by (rule "&I")
4010 qed
4011
4012 AOT_theorem "pre-en-eq:2[4]": <¬x1 x2 x3 x4 [F] → □¬x1 x2 x3 x4 [F]> (176.2)
4013 proof (rule "→I"; rule "raa-cor:1")
4014   AOT_assume <¬□¬x1 x2 x3 x4 [F]>
4015   AOT_hence <◇x1 x2 x3 x4 [F]>
4016   by (rule "conventions:5"[THEN "≡dfI"])
4017   AOT_hence <x1 x2 x3 x4 [F]>
4018   by (rule "S5Basic:13"[THEN "≡E"(1), OF "pre-en-eq:1[4]"[THEN RN],
4019       THEN "qml:2"[axiom_inst, THEN "→E"], THEN "→E"])
4020   moreover AOT_assume <¬x1 x2 x3 x4 [F]>
4021   ultimately AOT_show <x1 x2 x3 x4 [F] & ¬x1 x2 x3 x4 [F]> by (rule "&I")
4022 qed
4023
4024 AOT_theorem "en-eq:1[1]": <◇x1 [F] ≡ □x1 [F]> (177.1)
4025   using "pre-en-eq:1[1]"[THEN RN] "sc-eq-box-box:2" "∀I" "→E" by metis
4026 AOT_theorem "en-eq:1[2]": <◇x1 x2 [F] ≡ □x1 x2 [F]> (177.1)
4027   using "pre-en-eq:1[2]"[THEN RN] "sc-eq-box-box:2" "∀I" "→E" by metis
4028 AOT_theorem "en-eq:1[3]": <◇x1 x2 x3 [F] ≡ □x1 x2 x3 [F]> (177.1)
4029   using "pre-en-eq:1[3]"[THEN RN] "sc-eq-box-box:2" "∀I" "→E" by fast
4030 AOT_theorem "en-eq:1[4]": <◇x1 x2 x3 x4 [F] ≡ □x1 x2 x3 x4 [F]> (177.1)

```

```

4031   using "pre-en-eq:1[4]"[THEN RN] "sc-eq-box-box:2" "∀I" "→E" by fast
4032
4033 AOT_theorem "en-eq:2[1]": <x1[F] ≡ □x1[F]> (177.2)
4034   by (simp add: "≡I" "pre-en-eq:1[1]" "qml:2"[axiom_inst])
4035 AOT_theorem "en-eq:2[2]": <x1x2[F] ≡ □x1x2[F]> (177.2)
4036   by (simp add: "≡I" "pre-en-eq:1[2]" "qml:2"[axiom_inst])
4037 AOT_theorem "en-eq:2[3]": <x1x2x3[F] ≡ □x1x2x3[F]> (177.2)
4038   by (simp add: "≡I" "pre-en-eq:1[3]" "qml:2"[axiom_inst])
4039 AOT_theorem "en-eq:2[4]": <x1x2x3x4[F] ≡ □x1x2x3x4[F]> (177.2)
4040   by (simp add: "≡I" "pre-en-eq:1[4]" "qml:2"[axiom_inst])
4041
4042 AOT_theorem "en-eq:3[1]": <◇x1[F] ≡ x1[F]> (177.3)
4043   using "T◇" "derived-S5-rules:2"[OF "pre-en-eq:1[1]" "≡I" by blast
4044 AOT_theorem "en-eq:3[2]": <◇x1x2[F] ≡ x1x2[F]> (177.3)
4045   using "T◇" "derived-S5-rules:2"[OF "pre-en-eq:1[2]" "≡I" by blast
4046 AOT_theorem "en-eq:3[3]": <◇x1x2x3[F] ≡ x1x2x3[F]> (177.3)
4047   using "T◇" "derived-S5-rules:2"[OF "pre-en-eq:1[3]" "≡I" by blast
4048 AOT_theorem "en-eq:3[4]": <◇x1x2x3x4[F] ≡ x1x2x3x4[F]> (177.3)
4049   using "T◇" "derived-S5-rules:2"[OF "pre-en-eq:1[4]" "≡I" by blast
4050
4051 AOT_theorem "en-eq:4[1]": (177.4)
4052   <(x1[F] ≡ y1[G]) ≡ (□x1[F] ≡ □y1[G])>
4053   apply (rule "≡I"; rule "→I"; rule "≡I"; rule "→I")
4054   using "qml:2"[axiom_inst, THEN "→E"] "≡E"(1,2) "en-eq:2[1]" by blast+
4055 AOT_theorem "en-eq:4[2]": (177.4)
4056   <(x1x2[F] ≡ y1y2[G]) ≡ (□x1x2[F] ≡ □y1y2[G])>
4057   apply (rule "≡I"; rule "→I"; rule "≡I"; rule "→I")
4058   using "qml:2"[axiom_inst, THEN "→E"] "≡E"(1,2) "en-eq:2[2]" by blast+
4059 AOT_theorem "en-eq:4[3]": (177.4)
4060   <(x1x2x3[F] ≡ y1y2y3[G]) ≡ (□x1x2x3[F] ≡ □y1y2y3[G])>
4061   apply (rule "≡I"; rule "→I"; rule "≡I"; rule "→I")
4062   using "qml:2"[axiom_inst, THEN "→E"] "≡E"(1,2) "en-eq:2[3]" by blast+
4063 AOT_theorem "en-eq:4[4]": (177.4)
4064   <(x1x2x3x4[F] ≡ y1y2y3y4[G]) ≡ (□x1x2x3x4[F] ≡ □y1y2y3y4[G])>
4065   apply (rule "≡I"; rule "→I"; rule "≡I"; rule "→I")
4066   using "qml:2"[axiom_inst, THEN "→E"] "≡E"(1,2) "en-eq:2[4]" by blast+
4067
4068 AOT_theorem "en-eq:5[1]": (177.5)
4069   <□(x1[F] ≡ y1[G]) ≡ (□x1[F] ≡ □y1[G])>
4070   apply (rule "≡I"; rule "→I")
4071   using "en-eq:4[1]"[THEN "≡E"(1)] "qml:2"[axiom_inst, THEN "→E"]
4072   apply blast
4073   using "sc-eq-box-box:4"[THEN "→E", THEN "→E"]
4074   "&I"[OF "pre-en-eq:1[1]"[THEN RN], OF "pre-en-eq:1[1]"[THEN RN]]
4075   by blast
4076 AOT_theorem "en-eq:5[2]": (177.5)
4077   <□(x1x2[F] ≡ y1y2[G]) ≡ (□x1x2[F] ≡ □y1y2[G])>
4078   apply (rule "≡I"; rule "→I")
4079   using "en-eq:4[2]"[THEN "≡E"(1)] "qml:2"[axiom_inst, THEN "→E"]
4080   apply blast
4081   using "sc-eq-box-box:4"[THEN "→E", THEN "→E"]
4082   "&I"[OF "pre-en-eq:1[2]"[THEN RN], OF "pre-en-eq:1[2]"[THEN RN]]
4083   by blast
4084 AOT_theorem "en-eq:5[3]": (177.5)
4085   <□(x1x2x3[F] ≡ y1y2y3[G]) ≡ (□x1x2x3[F] ≡ □y1y2y3[G])>
4086   apply (rule "≡I"; rule "→I")
4087   using "en-eq:4[3]"[THEN "≡E"(1)] "qml:2"[axiom_inst, THEN "→E"]
4088   apply blast
4089   using "sc-eq-box-box:4"[THEN "→E", THEN "→E"]
4090   "&I"[OF "pre-en-eq:1[3]"[THEN RN], OF "pre-en-eq:1[3]"[THEN RN]]
4091   by blast
4092 AOT_theorem "en-eq:5[4]": (177.5)
4093   <□(x1x2x3x4[F] ≡ y1y2y3y4[G]) ≡ (□x1x2x3x4[F] ≡ □y1y2y3y4[G])>

```

```

4094 apply (rule "≡I"; rule "→I")
4095 using "en-eq:4[4]" [THEN "≡E"(1)] "qml:2"[axiom_inst, THEN "→E"]
4096 apply blast
4097 using "sc-eq-box-box:4"[THEN "→E", THEN "→E"]
4098 "&I"[OF "pre-en-eq:1[4]" [THEN RN], OF "pre-en-eq:1[4]" [THEN RN]]
4099 by blast
4100
4101 AOT_theorem "en-eq:6[1]":
4102   <(x1[F] ≡ y1[G]) ≡ □(x1[F] ≡ y1[G])> (177.6)
4103   using "en-eq:5[1]" [symmetric] "en-eq:4[1]" "≡E"(5) by fast
4104 AOT_theorem "en-eq:6[2]":
4105   <(x1x2[F] ≡ y1y2[G]) ≡ □(x1x2[F] ≡ y1y2[G])> (177.6)
4106   using "en-eq:5[2]" [symmetric] "en-eq:4[2]" "≡E"(5) by fast
4107 AOT_theorem "en-eq:6[3]":
4108   <(x1x2x3[F] ≡ y1y2y3[G]) ≡ □(x1x2x3[F] ≡ y1y2y3[G])> (177.6)
4109   using "en-eq:5[3]" [symmetric] "en-eq:4[3]" "≡E"(5) by fast
4110 AOT_theorem "en-eq:6[4]":
4111   <(x1x2x3x4[F] ≡ y1y2y3y4[G]) ≡ □(x1x2x3x4[F] ≡ y1y2y3y4[G])> (177.6)
4112   using "en-eq:5[4]" [symmetric] "en-eq:4[4]" "≡E"(5) by fast
4113
4114 AOT_theorem "en-eq:7[1]": <¬x1[F] ≡ □¬x1[F]> (177.7)
4115   using "pre-en-eq:2[1]" "qml:2"[axiom_inst] "≡I" by blast
4116 AOT_theorem "en-eq:7[2]": <¬x1x2[F] ≡ □¬x1x2[F]> (177.7)
4117   using "pre-en-eq:2[2]" "qml:2"[axiom_inst] "≡I" by blast
4118 AOT_theorem "en-eq:7[3]": <¬x1x2x3[F] ≡ □¬x1x2x3[F]> (177.7)
4119   using "pre-en-eq:2[3]" "qml:2"[axiom_inst] "≡I" by blast
4120 AOT_theorem "en-eq:7[4]": <¬x1x2x3x4[F] ≡ □¬x1x2x3x4[F]> (177.7)
4121   using "pre-en-eq:2[4]" "qml:2"[axiom_inst] "≡I" by blast
4122
4123 AOT_theorem "en-eq:8[1]": <◇¬x1[F] ≡ ¬x1[F]> (177.8)
4124   using "en-eq:2[1]" [THEN "oth-class-taut:4:b"[THEN "≡E"(1)]]
4125   "KBasic:11" "≡E"(5) [symmetric] by blast
4126 AOT_theorem "en-eq:8[2]": <◇¬x1x2[F] ≡ ¬x1x2[F]> (177.8)
4127   using "en-eq:2[2]" [THEN "oth-class-taut:4:b"[THEN "≡E"(1)]]
4128   "KBasic:11" "≡E"(5) [symmetric] by blast
4129 AOT_theorem "en-eq:8[3]": <◇¬x1x2x3[F] ≡ ¬x1x2x3[F]> (177.8)
4130   using "en-eq:2[3]" [THEN "oth-class-taut:4:b"[THEN "≡E"(1)]]
4131   "KBasic:11" "≡E"(5) [symmetric] by blast
4132 AOT_theorem "en-eq:8[4]": <◇¬x1x2x3x4[F] ≡ ¬x1x2x3x4[F]> (177.8)
4133   using "en-eq:2[4]" [THEN "oth-class-taut:4:b"[THEN "≡E"(1)]]
4134   "KBasic:11" "≡E"(5) [symmetric] by blast
4135
4136 AOT_theorem "en-eq:9[1]": <◇¬x1[F] ≡ □¬x1[F]> (177.9)
4137   using "en-eq:7[1]" "en-eq:8[1]" "≡E"(5) by blast
4138 AOT_theorem "en-eq:9[2]": <◇¬x1x2[F] ≡ □¬x1x2[F]> (177.9)
4139   using "en-eq:7[2]" "en-eq:8[2]" "≡E"(5) by blast
4140 AOT_theorem "en-eq:9[3]": <◇¬x1x2x3[F] ≡ □¬x1x2x3[F]> (177.9)
4141   using "en-eq:7[3]" "en-eq:8[3]" "≡E"(5) by blast
4142 AOT_theorem "en-eq:9[4]": <◇¬x1x2x3x4[F] ≡ □¬x1x2x3x4[F]> (177.9)
4143   using "en-eq:7[4]" "en-eq:8[4]" "≡E"(5) by blast
4144
4145 AOT_theorem "en-eq:10[1]": <Ax1[F] ≡ x1[F]> (177.10)
4146   by (metis "Act-Sub:3" "deduction-theorem" "≡I" "≡E"(1)
4147   "nec-imp-act" "en-eq:3[1]" "pre-en-eq:1[1]" )
4148 AOT_theorem "en-eq:10[2]": <Ax1x2[F] ≡ x1x2[F]> (177.10)
4149   by (metis "Act-Sub:3" "deduction-theorem" "≡I" "≡E"(1)
4150   "nec-imp-act" "en-eq:3[2]" "pre-en-eq:1[2]" )
4151 AOT_theorem "en-eq:10[3]": <Ax1x2x3[F] ≡ x1x2x3[F]> (177.10)
4152   by (metis "Act-Sub:3" "deduction-theorem" "≡I" "≡E"(1)
4153   "nec-imp-act" "en-eq:3[3]" "pre-en-eq:1[3]" )
4154 AOT_theorem "en-eq:10[4]": <Ax1x2x3x4[F] ≡ x1x2x3x4[F]> (177.10)
4155   by (metis "Act-Sub:3" "deduction-theorem" "≡I" "≡E"(1)
4156   "nec-imp-act" "en-eq:3[4]" "pre-en-eq:1[4]" )

```

```

4157
4158 AOT_theorem "oa-facts:1": <O!x → □O!x> (178.1)
4159 proof(rule "→I")
4160   AOT_modally_strict {
4161     AOT_have <[λx ◇E!x]x ≡ ◇E!x>
4162     by (rule "lambda-predicates:2"[axiom_inst, THEN "→E"]) "cqt:2"
4163   } note ϑ = this
4164   AOT_assume <O!x>
4165   AOT_hence <[λx ◇E!x]x>
4166   by (rule "=dfE"(2)[OF AOT_ordinary, rotated 1]) "cqt:2"
4167   AOT_hence <◇E!x> using ϑ[THEN "≡E"(1)] by blast
4168   AOT_hence <□◇E!x> using "qml:3"[axiom_inst, THEN "→E"] by blast
4169   AOT_hence <□[λx ◇E!x]x>
4170   by (AOT_subst <[λx ◇E!x]x> <◇E!x>)
4171   (auto simp: ϑ)
4172   AOT_thus <□O!x>
4173   by (rule "=dfI"(2)[OF AOT_ordinary, rotated 1]) "cqt:2"
4174 qed
4175
4176 AOT_theorem "oa-facts:2": <A!x → □A!x> (178.2)
4177 proof(rule "→I")
4178   AOT_modally_strict {
4179     AOT_have <[λx ¬◇E!x]x ≡ ¬◇E!x>
4180     by (rule "lambda-predicates:2"[axiom_inst, THEN "→E"]) "cqt:2"
4181   } note ϑ = this
4182   AOT_assume <A!x>
4183   AOT_hence <[λx ¬◇E!x]x>
4184   by (rule "=dfE"(2)[OF AOT_abstract, rotated 1]) "cqt:2"
4185   AOT_hence <¬◇E!x> using ϑ[THEN "≡E"(1)] by blast
4186   AOT_hence <□¬E!x> using "KBasic2:1"[THEN "≡E"(2)] by blast
4187   AOT_hence <□□¬E!x> using "4"[THEN "→E"] by blast
4188   AOT_hence <□¬◇E!x>
4189   using "KBasic2:1"
4190   by (AOT_subst (reverse) <¬◇E!x> <□¬E!x>) blast
4191   AOT_hence <□[λx ¬◇E!x]x>
4192   by (AOT_subst <[λx ¬◇E!x]x> <¬◇E!x>)
4193   (auto simp: ϑ)
4194   AOT_thus <□A!x>
4195   by (rule "=dfI"(2)[OF AOT_abstract, rotated 1]) "cqt:2[lambda]"
4196 qed
4197
4198 AOT_theorem "oa-facts:3": <◇O!x → O!x> (178.3)
4199 using "oa-facts:1" "B◇" "RM◇" "Hypothetical Syllogism" by blast
4200 AOT_theorem "oa-facts:4": <◇A!x → A!x> (178.4)
4201 using "oa-facts:2" "B◇" "RM◇" "Hypothetical Syllogism" by blast
4202
4203 AOT_theorem "oa-facts:5": <◇O!x ≡ □O!x> (178.5)
4204 by (meson "Act-Sub:3" "Hypothetical Syllogism" "≡I" "nec-imp-act"
4205 "oa-facts:1" "oa-facts:3")
4206
4207 AOT_theorem "oa-facts:6": <◇A!x ≡ □A!x> (178.6)
4208 by (meson "Act-Sub:3" "Hypothetical Syllogism" "≡I" "nec-imp-act"
4209 "oa-facts:2" "oa-facts:4")
4210
4211 AOT_theorem "oa-facts:7": <O!x ≡  $\mathcal{A}$ O!x> (178.7)
4212 by (meson "Act-Sub:3" "Hypothetical Syllogism" "≡I" "nec-imp-act"
4213 "oa-facts:1" "oa-facts:3")
4214
4215 AOT_theorem "oa-facts:8": <A!x ≡  $\mathcal{A}$ A!x> (178.8)
4216 by (meson "Act-Sub:3" "Hypothetical Syllogism" "≡I" "nec-imp-act"
4217 "oa-facts:2" "oa-facts:4")
4218
4219 subsection<The Theory of Relations>

```

```

4220 text<\label{PLM: 9.10}>
4221
4222 AOT_theorem "beta-C-meta": (179)
4223   <[ $\lambda\mu_1 \dots \mu_n \varphi\{\mu_1 \dots \mu_n, \nu_1 \dots \nu_n\}] \downarrow \rightarrow$ 
4224   ( $[\lambda\mu_1 \dots \mu_n \varphi\{\mu_1 \dots \mu_n, \nu_1 \dots \nu_n\}] \nu_1 \dots \nu_n \equiv \varphi\{\nu_1 \dots \nu_n, \nu_1 \dots \nu_n\}$ )>
4225   using "lambda-predicates:2"[axiom_inst] by blast
4226
4227 AOT_theorem "beta-C-cor:1": (181.1)
4228   <( $\forall \nu_1 \dots \forall \nu_n ([\lambda\mu_1 \dots \mu_n \varphi\{\mu_1 \dots \mu_n, \nu_1 \dots \nu_n\}] \downarrow)$ )  $\rightarrow$ 
4229    $\forall \nu_1 \dots \forall \nu_n ([\lambda\mu_1 \dots \mu_n \varphi\{\mu_1 \dots \mu_n, \nu_1 \dots \nu_n\}] \nu_1 \dots \nu_n \equiv \varphi\{\nu_1 \dots \nu_n, \nu_1 \dots \nu_n\})$ >
4230   apply (rule "cqt-basic:14"[where 'a='a, THEN " $\rightarrow$ E"])
4231   using "beta-C-meta" " $\forall$ I" by fast
4232
4233 AOT_theorem "beta-C-cor:2": (181.2)
4234   <[ $\lambda\mu_1 \dots \mu_n \varphi\{\mu_1 \dots \mu_n\}] \downarrow \rightarrow$ 
4235    $\forall \nu_1 \dots \forall \nu_n ([\lambda\mu_1 \dots \mu_n \varphi\{\mu_1 \dots \mu_n\}] \nu_1 \dots \nu_n \equiv \varphi\{\nu_1 \dots \nu_n\})$ >
4236   apply (rule " $\rightarrow$ I"; rule " $\forall$ I")
4237   using "beta-C-meta"[THEN " $\rightarrow$ E"] by fast
4238
4239 (* TODO: add better syntax parsing for INSTANCE_OF_CQT_2 *)
4240 theorem "beta-C-cor:3": (181.3)
4241   assumes  $\langle \lambda \nu_1 \nu_n. \text{AOT\_instance\_of\_cqt\_2 } (\varphi \text{ (AOT\_term\_of\_var } \nu_1 \nu_n)) \rangle$ 
4242   shows  $\langle [v \models \forall \nu_1 \dots \forall \nu_n ([\lambda\mu_1 \dots \mu_n \varphi\{\nu_1 \dots \nu_n, \mu_1 \dots \mu_n\}] \nu_1 \dots \nu_n \equiv$ 
4243    $\varphi\{\nu_1 \dots \nu_n, \nu_1 \dots \nu_n\})] \rangle$ 
4244   using "cqt:2[lambda]"[axiom_inst, OF assms]
4245   "beta-C-cor:1"[THEN " $\rightarrow$ E"] " $\forall$ I" by fast
4246
4247 AOT_theorem "betaC:1:a":  $\langle [\lambda\mu_1 \dots \mu_n \varphi\{\mu_1 \dots \mu_n\}] \kappa_1 \dots \kappa_n \vdash_{\square} \varphi\{\kappa_1 \dots \kappa_n\} \rangle$  (182.1.a)
4248 proof -
4249   AOT_modally_strict {
4250     AOT_assume  $\langle [\lambda\mu_1 \dots \mu_n \varphi\{\mu_1 \dots \mu_n\}] \kappa_1 \dots \kappa_n \rangle$ 
4251     moreover AOT_have  $\langle [\lambda\mu_1 \dots \mu_n \varphi\{\mu_1 \dots \mu_n\}] \downarrow \rangle$  and  $\langle \kappa_1 \dots \kappa_n \downarrow \rangle$ 
4252     using calculation "cqt:5:a"[axiom_inst, THEN " $\rightarrow$ E"] "&E" by blast+
4253     ultimately AOT_show  $\langle \varphi\{\kappa_1 \dots \kappa_n\} \rangle$ 
4254     using "beta-C-cor:2"[THEN " $\rightarrow$ E", THEN " $\forall$ E"(1), THEN " $\equiv$ E"(1)] by blast
4255   }
4256 qed
4257
4258 AOT_theorem "betaC:1:b":  $\langle \neg \varphi\{\kappa_1 \dots \kappa_n\} \vdash_{\square} \neg [\lambda\mu_1 \dots \mu_n \varphi\{\mu_1 \dots \mu_n\}] \kappa_1 \dots \kappa_n \rangle$  (182.1.b)
4259   using "betaC:1:a" "raa-cor:3" by blast
4260
4261 lemmas " $\beta \rightarrow C$ " = "betaC:1:a" "betaC:1:b"
4262
4263 AOT_theorem "betaC:2:a": (182.2.a)
4264    $\langle [\lambda\mu_1 \dots \mu_n \varphi\{\mu_1 \dots \mu_n\}] \downarrow, \kappa_1 \dots \kappa_n \downarrow, \varphi\{\kappa_1 \dots \kappa_n\} \vdash_{\square}$ 
4265    $[\lambda\mu_1 \dots \mu_n \varphi\{\mu_1 \dots \mu_n\}] \kappa_1 \dots \kappa_n \rangle$ 
4266 proof -
4267   AOT_modally_strict {
4268     AOT_assume 1:  $\langle [\lambda\mu_1 \dots \mu_n \varphi\{\mu_1 \dots \mu_n\}] \downarrow \rangle$ 
4269     and 2:  $\langle \kappa_1 \dots \kappa_n \downarrow \rangle$ 
4270     and 3:  $\langle \varphi\{\kappa_1 \dots \kappa_n\} \rangle$ 
4271     AOT_hence  $\langle [\lambda\mu_1 \dots \mu_n \varphi\{\mu_1 \dots \mu_n\}] \kappa_1 \dots \kappa_n \rangle$ 
4272     using "beta-C-cor:2"[THEN " $\rightarrow$ E", OF 1, THEN " $\forall$ E"(1), THEN " $\equiv$ E"(2)]
4273     by blast
4274   }
4275   AOT_thus  $\langle [\lambda\mu_1 \dots \mu_n \varphi\{\mu_1 \dots \mu_n\}] \downarrow, \kappa_1 \dots \kappa_n \downarrow, \varphi\{\kappa_1 \dots \kappa_n\} \vdash_{\square}$ 
4276    $[\lambda\mu_1 \dots \mu_n \varphi\{\mu_1 \dots \mu_n\}] \kappa_1 \dots \kappa_n \rangle$ 
4277   by blast
4278 qed
4279
4280 AOT_theorem "betaC:2:b": (182.2.b)
4281    $\langle [\lambda\mu_1 \dots \mu_n \varphi\{\mu_1 \dots \mu_n\}] \downarrow, \kappa_1 \dots \kappa_n \downarrow, \neg [\lambda\mu_1 \dots \mu_n \varphi\{\mu_1 \dots \mu_n\}] \kappa_1 \dots \kappa_n \vdash_{\square}$ 
4282    $\neg \varphi\{\kappa_1 \dots \kappa_n\} \rangle$ 

```

```

4283   using "betaC:2:a" "raa-cor:3" by blast
4284
4285 lemmas " $\beta \leftarrow C$ " = "betaC:2:a" "betaC:2:b"
4286
4287 AOT_theorem "eta-conversion-lemma1:1": < $\Pi \downarrow \rightarrow [\lambda x_1 \dots x_n [\Pi] x_1 \dots x_n] = \Pi$ > (184.1)
4288   using "lambda-predicates:3"[axiom_inst] " $\forall I$ " " $\forall E(1)$ " " $\rightarrow I$ " by fast
4289
4290 (* Note: generalized alphabetic variant of the last theorem *)
4291 AOT_theorem "eta-conversion-lemma1:2": < $\Pi \downarrow \rightarrow [\lambda \nu_1 \dots \nu_n [\Pi] \nu_1 \dots \nu_n] = \Pi$ > (184.2)
4292   using "eta-conversion-lemma1:1".
4293
4294 text<Note: not explicitly part of PLM.>
4295 AOT_theorem id_sym:
4296   assumes < $\tau = \tau'$ >
4297   shows < $\tau' = \tau$ >
4298   using "rule=E"[where  $\varphi = \lambda \tau' . \langle \tau' = \tau \rangle$ ", rotated 1, OF assms]
4299     "I"(1)[OF "t=t-proper:1"[THEN " $\rightarrow E$ ", OF assms]] by auto
4300 declare id_sym[sym]
4301
4302 text<Note: not explicitly part of PLM.>
4303 AOT_theorem id_trans:
4304   assumes < $\tau = \tau'$ > and < $\tau' = \tau''$ >
4305   shows < $\tau = \tau''$ >
4306   using "rule=E" assms by blast
4307 declare id_trans[trans]
4308
4309 method " $\eta C$ " for  $\Pi :: \langle \langle 'a :: \{AOT\_Term\_id\_2, AOT\_\kappa s\} \rangle \rangle =$ 
4310   (match conclusion in "[ $v \models \tau[\Pi] = \tau'\{\Pi\}$ ]" for  $v \tau \tau' \Rightarrow$  <
4311     rule "rule=E"[rotated 1, OF "eta-conversion-lemma1:2"
4312       [THEN " $\rightarrow E$ ", of  $v$  " $\langle [\Pi] \rangle$ ", symmetric]]>)
4313
4314 AOT_theorem "sub-des-lam:1": (186.1)
4315   < $[\lambda z_1 \dots z_n \chi\{z_1 \dots z_n, \iota x \varphi\{x\}\}] \downarrow \& \iota x \varphi\{x\} = \iota x \psi\{x\} \rightarrow$ 
4316      $[\lambda z_1 \dots z_n \chi\{z_1 \dots z_n, \iota x \varphi\{x\}\}] = [\lambda z_1 \dots z_n \chi\{z_1 \dots z_n, \iota x \psi\{x\}\}]$ >
4317 proof(rule " $\rightarrow I$ ")
4318   AOT_assume A: < $[\lambda z_1 \dots z_n \chi\{z_1 \dots z_n, \iota x \varphi\{x\}\}] \downarrow \& \iota x \varphi\{x\} = \iota x \psi\{x\}$ >
4319   AOT_show < $[\lambda z_1 \dots z_n \chi\{z_1 \dots z_n, \iota x \varphi\{x\}\}] = [\lambda z_1 \dots z_n \chi\{z_1 \dots z_n, \iota x \psi\{x\}\}]$ >
4320   using "rule=E"[where  $\varphi = \lambda \tau . \langle [\lambda z_1 \dots z_n \chi\{z_1 \dots z_n, \iota x \varphi\{x\}\}] =$ 
4321      $[\lambda z_1 \dots z_n \chi\{z_1 \dots z_n, \tau\}] \rangle$ ",
4322     OF "I"(1)[OF A[THEN "&E"(1)], OF A[THEN "&E"(2)]]
4323   by blast
4324 qed
4325
4326 AOT_theorem "sub-des-lam:2": (186.2)
4327   < $\iota x \varphi\{x\} = \iota x \psi\{x\} \rightarrow \chi\{\iota x \varphi\{x\}\} = \chi\{\iota x \psi\{x\}\}$ > for  $\chi :: \langle \kappa \Rightarrow o \rangle$ 
4328   using "rule=E"[where  $\varphi = \lambda \tau . \langle \chi\{\iota x \varphi\{x\}\} = \chi\{\tau\} \rangle$ ",
4329     OF "I"(1)[OF "log-prop-prop:2"]] " $\rightarrow I$ " by blast
4330
4331 AOT_theorem "prop-equiv": < $F = G \equiv \forall x (x[F] \equiv x[G])$ > (187)
4332 proof(rule " $\equiv I$ "; rule " $\rightarrow I$ ")
4333   AOT_assume < $F = G$ >
4334   AOT_thus < $\forall x (x[F] \equiv x[G])$ >
4335   by (rule "rule=E"[rotated]) (fact "oth-class-taut:3:a"[THEN GEN])
4336 next
4337   AOT_assume < $\forall x (x[F] \equiv x[G])$ >
4338   AOT_hence < $x[F] \equiv x[G]$ > for  $x$ 
4339   using " $\forall E$ " by blast
4340   AOT_hence < $\Box(x[F] \equiv x[G])$ > for  $x$ 
4341   using "en-eq:6[1]"[THEN " $\equiv E(1)$ "] by blast
4342   AOT_hence < $\forall x \Box(x[F] \equiv x[G])$ >
4343   by (rule GEN)
4344   AOT_hence < $\Box \forall x (x[F] \equiv x[G])$ >
4345   using BF[THEN " $\rightarrow E$ "] by fast

```



```

4346   AOT_thus "F = G"
4347     using "p-identity-thm2:1"[THEN "≡E"(2)] by blast
4348 qed
4349
4350 AOT_theorem "relations:1": (189.1)
4351   assumes <INSTANCE_OF_CQT_2(φ)>
4352   shows <∃F □∀x1...∀xn ([F]x1...xn ≡ φ{x1...xn})>
4353   apply (rule "∃I"(1)[where τ="«[λx1...xn φ{x1...xn}]»"]
4354   using "cqt:2[lambda]"[OF assms, axiom_inst]
4355     "beta-C-cor:2"[THEN "→E", THEN RN] by blast+
4356
4357 AOT_theorem "relations:2": (189.2)
4358   assumes <INSTANCE_OF_CQT_2(φ)>
4359   shows <∃F □∀x ([F]x ≡ φ{x})>
4360   using "relations:1" assms by blast
4361
4362 AOT_theorem "block-paradox:1": <¬[λx ∃G (x[G] & ¬[G]x)]↓> (190.1)
4363 proof(rule "raa-cor:2")
4364   let ?K="«[λx ∃G (x[G] & ¬[G]x)]»"
4365   AOT_assume A: <<?K>>↓>
4366   AOT_have <∃x (A!x & ∀F (x[F] ≡ F = «?K»))>
4367     using "A-objects"[axiom_inst] by fast
4368   then AOT_obtain a where ξ: <A!a & ∀F (a[F] ≡ F = «?K»)>
4369     using "∃E"[rotated] by blast
4370   AOT_show <p & ¬p> for p
4371   proof (rule "∨E"(1)[OF "exc-mid"]; rule "→I")
4372     AOT_assume B: <[«?K>]a>
4373     AOT_hence <∃G (a[G] & ¬[G]a)>
4374       using "β→C" A by blast
4375     then AOT_obtain P where <a[P] & ¬[P]a>
4376       using "∃E"[rotated] by blast
4377     moreover AOT_have <P = [«?K>]>
4378       using ξ[THEN "&E"(2), THEN "∨E"(2), THEN "≡E"(1)]
4379       calculation[THEN "&E"(1)] by blast
4380     ultimately AOT_have <¬[«?K>]a>
4381       using "rule=E" "&E"(2) by fast
4382     AOT_thus <p & ¬p>
4383     using B RAA by blast
4384   next
4385     AOT_assume B: <¬[«?K>]a>
4386     AOT_hence <¬∃G (a[G] & ¬[G]a)>
4387       using "β←C" "cqt:2[const_var]"[of a, axiom_inst] A by blast
4388     AOT_hence C: <∀G ¬(a[G] & ¬[G]a)>
4389       using "cqt-further:4"[THEN "→E"] by blast
4390     AOT_have <∀G (a[G] → [G]a)>
4391       by (AOT_subst <a[G] → [G]a> <¬(a[G] & ¬[G]a)> for: G)
4392       (auto simp: "oth-class-taut:1:a" C)
4393     AOT_hence <a[«?K>] → [«?K>]a>
4394       using "∨E" A by blast
4395     moreover AOT_have <a[«?K>]>
4396       using ξ[THEN "&E"(2), THEN "∨E"(1), OF A, THEN "≡E"(2)]
4397       using "=I"(1)[OF A] by blast
4398     ultimately AOT_show <p & ¬p>
4399     using B "→E" RAA by blast
4400   qed
4401 qed
4402
4403 AOT_theorem "block-paradox:2": <¬∃F ∀x ([F]x ≡ ∃G(x[G] & ¬[G]x))> (190.2)
4404 proof(rule RAA(2))
4405   AOT_assume <∃F ∀x ([F]x ≡ ∃G (x[G] & ¬[G]x))>
4406   then AOT_obtain F where F_prop: <∀x ([F]x ≡ ∃G (x[G] & ¬[G]x))>
4407     using "∃E"[rotated] by blast
4408   AOT_have <∃x (A!x & ∀G (x[G] ≡ G = F))>

```



```

4409   using "A-objects"[axiom_inst] by fast
4410 then AOT_obtain a where  $\xi$ :  $\langle A!a \ \& \ \forall G \ (a[G] \equiv G = F) \rangle$ 
4411   using "E"[rotated] by blast
4412 AOT_show  $\langle \neg \exists F \ \forall x \ ([F]x \equiv \exists G \ (x[G] \ \& \ \neg [G]x)) \rangle$ 
4413 proof (rule "VE"(1)[OF "exc-mid"]; rule " $\rightarrow$ I")
4414   AOT_assume B:  $\langle [F]a \rangle$ 
4415   AOT_hence  $\langle \exists G \ (a[G] \ \& \ \neg [G]a) \rangle$ 
4416     using F_prop[THEN "VE"(2), THEN "E"(1)] by blast
4417   then AOT_obtain P where  $\langle a[P] \ \& \ \neg [P]a \rangle$ 
4418     using "E"[rotated] by blast
4419   moreover AOT_have  $\langle P = F \rangle$ 
4420     using  $\xi$ [THEN "&E"(2), THEN "VE"(2), THEN "E"(1)]
4421     calculation[THEN "&E"(1)] by blast
4422   ultimately AOT_have  $\langle \neg [F]a \rangle$ 
4423     using "rule=E" "&E"(2) by fast
4424   AOT_thus  $\langle \neg \exists F \ \forall x \ ([F]x \equiv \exists G \ (x[G] \ \& \ \neg [G]x)) \rangle$ 
4425     using B RAA by blast
4426 next
4427   AOT_assume B:  $\langle \neg [F]a \rangle$ 
4428   AOT_hence  $\langle \neg \exists G \ (a[G] \ \& \ \neg [G]a) \rangle$ 
4429     using "oth-class-taut:4:b"[THEN "E"(1),
4430     OF F_prop[THEN "VE"(2)[of _ _ a]], THEN "E"(1)]
4431     by simp
4432   AOT_hence C:  $\langle \forall G \ \neg (a[G] \ \& \ \neg [G]a) \rangle$ 
4433     using "cqt-further:4"[THEN " $\rightarrow$ E"] by blast
4434   AOT_have  $\langle \forall G \ (a[G] \ \rightarrow \ [G]a) \rangle$ 
4435     by (AOT_subst  $\langle a[G] \ \rightarrow \ [G]a \ \rightarrow \ \neg (a[G] \ \& \ \neg [G]a) \rangle$  for: G)
4436     (auto simp: "oth-class-taut:1:a" C)
4437   AOT_hence  $\langle a[F] \ \rightarrow \ [F]a \rangle$ 
4438     using "VE" by blast
4439   moreover AOT_have  $\langle a[F] \rangle$ 
4440     using  $\xi$ [THEN "&E"(2), THEN "VE"(2), of F, THEN "E"(2)]
4441     using "I"(2) by blast
4442   ultimately AOT_show  $\langle \neg \exists F \ \forall x \ ([F]x \equiv \exists G \ (x[G] \ \& \ \neg [G]x)) \rangle$ 
4443     using B " $\rightarrow$ E" RAA by blast
4444 qed
4445 qed(simp)
4446
4447 AOT_theorem "block-paradox:3":  $\langle \neg \forall y \ [\lambda z \ z = y] \downarrow \rangle$  (190.3)
4448 proof(rule RAA(2))
4449   AOT_assume  $\vartheta$ :  $\langle \forall y \ [\lambda z \ z = y] \downarrow \rangle$ 
4450   AOT_have  $\langle \exists x \ (A!x \ \& \ \forall F \ (x[F] \equiv \exists y \ (F = [\lambda z \ z = y] \ \& \ \neg y[F]))) \rangle$ 
4451     using "A-objects"[axiom_inst] by force
4452   then AOT_obtain a where
4453     a_prop:  $\langle A!a \ \& \ \forall F \ (a[F] \equiv \exists y \ (F = [\lambda z \ z = y] \ \& \ \neg y[F])) \rangle$ 
4454     using "E"[rotated] by blast
4455   AOT_have  $\zeta$ :  $\langle a[\lambda z \ z = a] \equiv \exists y \ ([\lambda z \ z = a] = [\lambda z \ z = y] \ \& \ \neg y[\lambda z \ z = a]) \rangle$ 
4456     using  $\vartheta$ [THEN "VE"(2)] a_prop[THEN "&E"(2), THEN "VE"(1)] by blast
4457   AOT_show  $\langle \neg \forall y \ [\lambda z \ z = y] \downarrow \rangle$ 
4458   proof (rule "VE"(1)[OF "exc-mid"]; rule " $\rightarrow$ I")
4459     AOT_assume A:  $\langle a[\lambda z \ z = a] \rangle$ 
4460     AOT_hence  $\langle \exists y \ ([\lambda z \ z = a] = [\lambda z \ z = y] \ \& \ \neg y[\lambda z \ z = a]) \rangle$ 
4461       using  $\zeta$ [THEN "E"(1)] by blast
4462     then AOT_obtain b where b_prop:  $\langle [\lambda z \ z = a] = [\lambda z \ z = b] \ \& \ \neg b[\lambda z \ z = a] \rangle$ 
4463       using "E"[rotated] by blast
4464     moreover AOT_have  $\langle a = a \rangle$  by (rule "I")
4465     moreover AOT_have  $\langle [\lambda z \ z = a] \downarrow \rangle$  using  $\vartheta$  "VE" by blast
4466     moreover AOT_have  $\langle a \downarrow \rangle$  using "cqt:2[const_var]"[axiom_inst] .
4467     ultimately AOT_have  $\langle [\lambda z \ z = a]a \rangle$  using " $\beta \leftarrow C$ " by blast
4468     AOT_hence  $\langle [\lambda z \ z = b]a \rangle$  using "rule=E" b_prop[THEN "&E"(1)] by fast
4469     AOT_hence  $\langle a = b \rangle$  using " $\beta \rightarrow C$ " by blast
4470     AOT_hence  $\langle b[\lambda z \ z = a] \rangle$  using A "rule=E" by fast
4471     AOT_thus  $\langle \neg \forall y \ [\lambda z \ z = y] \downarrow \rangle$  using b_prop[THEN "&E"(2)] RAA by blast

```

```

4472 next
4473   AOT_assume A: <¬a[λz z = a]>
4474   AOT_hence <¬∃y ([λz z = a] = [λz z = y] & ¬y[λz z = a])>
4475     using ζ "oth-class-taut:4:b"[THEN "≡E"(1), THEN "≡E"(1)] by blast
4476   AOT_hence <∀y ¬([λz z = a] = [λz z = y] & ¬y[λz z = a])>
4477     using "cqt-further:4"[THEN "→E"] by blast
4478   AOT_hence <¬([λz z = a] = [λz z = a] & ¬a[λz z = a])>
4479     using "∀E" by blast
4480   AOT_hence <[λz z = a] = [λz z = a] → a[λz z = a]>
4481     by (metis "&I" "deduction-theorem" "raa-cor:4")
4482   AOT_hence <a[λz z = a]> using "=I"(1) ∅[THEN "∀E"(2)] "→E" by blast
4483   AOT_thus <¬∀y [λz z = y]↓> using A RAA by blast
4484 qed
4485 qed(simp)
4486
4487 AOT_theorem "block-paradox:4": <¬∀y ∃F ∀x([F]x ≡ x = y)> (190.4)
4488 proof(rule RAA(2))
4489   AOT_assume ∅: <∀y ∃F ∀x([F]x ≡ x = y)>
4490   AOT_have <∃x (A!x & ∀F (x[F] ≡ ∃z (∀y([F]y ≡ y = z) & ¬z[F])))>
4491     using "A-objects"[axiom_inst] by force
4492   then AOT_obtain a where
4493     a_prop: <A!a & ∀F (a[F] ≡ ∃z (∀y([F]y ≡ y = z) & ¬z[F]))>
4494     using "∃E"[rotated] by blast
4495   AOT_obtain F where F_prop: <∀x ([F]x ≡ x = a)>
4496     using ∅[THEN "∀E"(2)] "∃E"[rotated] by blast
4497   AOT_have ζ: <a[F] ≡ ∃z (∀y ([F]y ≡ y = z) & ¬z[F])>
4498     using a_prop[THEN "&E"(2), THEN "∀E"(2)] by blast
4499   AOT_show <¬∀y ∃F ∀x([F]x ≡ x = y)>
4500   proof (rule "∀E"(1)[OF "exc-mid"]; rule "→I")
4501     AOT_assume A: <a[F]>
4502     AOT_hence <∃z (∀y ([F]y ≡ y = z) & ¬z[F])>
4503       using ζ[THEN "≡E"(1)] by blast
4504     then AOT_obtain b where b_prop: <∀y ([F]y ≡ y = b) & ¬b[F]>
4505       using "∃E"[rotated] by blast
4506     moreover AOT_have <[F]a>
4507       using F_prop[THEN "∀E"(2), THEN "≡E"(2)] "=I"(2) by blast
4508     ultimately AOT_have <a = b>
4509       using "∀E"(2) "≡E"(1) "&E" by fast
4510     AOT_hence <a = b>
4511       using "β→C" by blast
4512     AOT_hence <b[F]>
4513       using A "rule=E" by fast
4514     AOT_thus <¬∀y ∃F ∀x([F]x ≡ x = y)>
4515       using b_prop[THEN "&E"(2)] RAA by blast
4516   next
4517     AOT_assume A: <¬a[F]>
4518     AOT_hence <¬∃z (∀y ([F]y ≡ y = z) & ¬z[F])>
4519       using ζ "oth-class-taut:4:b"[THEN "≡E"(1), THEN "≡E"(1)] by blast
4520     AOT_hence <∀z ¬(∀y ([F]y ≡ y = z) & ¬z[F])>
4521       using "cqt-further:4"[THEN "→E"] by blast
4522     AOT_hence <¬(∀y ([F]y ≡ y = a) & ¬a[F])>
4523       using "∀E" by blast
4524     AOT_hence <∀y ([F]y ≡ y = a) → a[F]>
4525       by (metis "&I" "deduction-theorem" "raa-cor:4")
4526     AOT_hence <a[F]> using F_prop "→E" by blast
4527     AOT_thus <¬∀y ∃F ∀x([F]x ≡ x = y)>
4528       using A RAA by blast
4529   qed
4530 qed(simp)
4531
4532 AOT_theorem "block-paradox:5": <¬∃F∀x∀y([F]xy ≡ y = x)> (190.5)
4533 proof(rule "raa-cor:2")
4534   AOT_assume <∃F∀x∀y([F]xy ≡ y = x)>

```

```

4535 then AOT_obtain F where F_prop: <∀x∀y([F]xy ≡ y = x)>
4536   using "∃E"[rotated] by blast
4537 {
4538   fix x
4539   AOT_have 1: <∀y([F]xy ≡ y = x)>
4540     using F_prop "∀E" by blast
4541   AOT_have 2: <[λz [F]xz]↓> by "cqt:2"
4542   moreover AOT_have <∀y([λz [F]xz]y ≡ y = x)>
4543     proof(rule "∀I")
4544       fix y
4545       AOT_have <[λz [F]xz]y ≡ [F]xy>
4546         using "beta-C-meta"[THEN "→E"] 2 by fast
4547       also AOT_have <... ≡ y = x>
4548         using 1 "∀E" by fast
4549       finally AOT_show <[λz [F]xz]y ≡ y = x>.
4550     qed
4551   ultimately AOT_have <∃F∀y([F]y ≡ y = x)>
4552     using "∃I" by fast
4553 }
4554 AOT_hence <∀x∃F∀y([F]y ≡ y = x)>
4555   by (rule GEN)
4556 AOT_thus <∀x∃F∀y([F]y ≡ y = x) & ¬∀x∃F∀y([F]y ≡ y = x)>
4557   using "&I" "block-paradox:4" by blast
4558 qed
4559
4560 AOT_act_theorem "block-paradox2:1": (191.1)
4561   <∀x [G]x → ¬[λx [G]ℓy (y = x & ∃H (x[H] & ¬[H]x))]>
4562 proof(rule "→I"; rule "raa-cor:2")
4563   AOT_assume antecedant: <∀x [G]x>
4564   AOT_have Lemma: <∀x ([G]ℓy(y = x & ∃H (x[H] & ¬[H]x)) ≡ ∃H (x[H] & ¬[H]x))>
4565   proof(rule GEN)
4566     fix x
4567     AOT_have A: <[G]ℓy (y = x & ∃H (x[H] & ¬[H]x)) ≡
4568       ∃!y (y = x & ∃H (x[H] & ¬[H]x))>
4569     proof(rule "≡I"; rule "→I")
4570       AOT_assume <[G]ℓy (y = x & ∃H (x[H] & ¬[H]x))>
4571       AOT_hence <ℓy (y = x & ∃H (x[H] & ¬[H]x))↓>
4572         using "cqt:5:a"[axiom_inst, THEN "→E", THEN "&E"(2)] by blast
4573       AOT_thus <∃!y (y = x & ∃H (x[H] & ¬[H]x))>
4574         using "!-exists:1"[THEN "≡E"(1)] by blast
4575     next
4576     AOT_assume A: <∃!y (y = x & ∃H (x[H] & ¬[H]x))>
4577     AOT_obtain a where a_1: <a = x & ∃H (x[H] & ¬[H]x)>
4578       and a_2: <∀z (z = x & ∃H (x[H] & ¬[H]x) → z = a)>
4579     using "uniqueness:1"[THEN "≡dfE", OF A] "&E" "∃E"[rotated] by blast
4580     AOT_have a_3: <[G]a>
4581     using antecedant "∀E" by blast
4582     AOT_show <[G]ℓy (y = x & ∃H (x[H] & ¬[H]x))>
4583       apply (rule "russell-axiom[exe,1].russell-axiom"[THEN "≡E"(2)])
4584       apply (rule "∃I"(2))
4585     using a_1 a_2 a_3 "&I" by blast
4586   qed
4587   also AOT_have B: <... ≡ ∃H (x[H] & ¬[H]x)>
4588   proof (rule "≡I"; rule "→I")
4589     AOT_assume A: <∃!y (y = x & ∃H (x[H] & ¬[H]x))>
4590     AOT_obtain a where <a = x & ∃H (x[H] & ¬[H]x)>
4591     using "uniqueness:1"[THEN "≡dfE", OF A] "&E" "∃E"[rotated] by blast
4592     AOT_thus <∃H (x[H] & ¬[H]x)> using "&E" by blast
4593   next
4594     AOT_assume <∃H (x[H] & ¬[H]x)>
4595     AOT_hence <x = x & ∃H (x[H] & ¬[H]x)>
4596     using "id-eq:1" "&I" by blast
4597     moreover AOT_have <∀z (z = x & ∃H (x[H] & ¬[H]x) → z = x)>

```

```

4598     by (simp add: "Conjunction Simplification"(1) "universal-cor")
4599     ultimately AOT_show <∃!y (y = x & ∃H (x[H] & ¬[H]x))>
4600     using "uniqueness:1"[THEN "≐dfI"] "&I" "∃I"(2) by fast
4601   qed
4602   finally AOT_show <([G]ly(y = x & ∃H (x[H] & ¬[H]x)) ≐ ∃H (x[H] & ¬[H]x))> .
4603   qed
4604
4605   AOT_assume A: <[λx [G]ly (y = x & ∃H (x[H] & ¬[H]x))>↓>
4606   AOT_have ∅: <∀x ([λx [G]ly (y = x & ∃H (x[H] & ¬[H]x))]x ≐
4607     [G]ly(y = x & ∃H (x[H] & ¬[H]x)))>
4608     using "beta-C-meta"[THEN "→E", OF A] "∀I" by fast
4609   AOT_have <∀x ([λx [G]ly (y = x & ∃H (x[H] & ¬[H]x))]x ≐ ∃H (x[H] & ¬[H]x))>
4610     using ∅ Lemma "cqt-basic:10"[THEN "→E"] "&I" by fast
4611   AOT_hence <∃F ∀x ([F]x ≐ ∃H (x[H] & ¬[H]x))>
4612     using "∃I"(1) A by fast
4613   AOT_thus <(∃F ∀x ([F]x ≐ ∃H (x[H] & ¬[H]x))) &
4614     (¬∃F ∀x ([F]x ≐ ∃H (x[H] & ¬[H]x)))>
4615     using "block-paradox:2" "&I" by blast
4616   qed
4617
4618   text<Note: Strengthens the above to a modally-strict theorem.
4619     Not explicitly part of PLM.>
4620   AOT_theorem "block-paradox2:1[strict]":
4621     <∀x A[G]x → ¬[λx [G]ly (y = x & ∃H (x[H] & ¬[H]x))>↓>
4622   proof(rule "→I"; rule "raa-cor:2")
4623     AOT_assume antecedant: <∀x A[G]x>
4624     AOT_have Lemma: <A∀x ([G]ly(y = x & ∃H (x[H] & ¬[H]x)) ≐ ∃H (x[H] & ¬[H]x))>
4625     proof(safe intro!: GEN "Act-Basic:5"[THEN "≐E"(2)]
4626       "logic-actual-nec:3"[axiom_inst, THEN "≐E"(2)])
4627       fix x
4628       AOT_have A: <A[G]ly (y = x & ∃H (x[H] & ¬[H]x)) ≐
4629         ∃!y A(y = x & ∃H (x[H] & ¬[H]x))>
4630       proof(rule "≐I"; rule "→I")
4631         AOT_assume <A[G]ly (y = x & ∃H (x[H] & ¬[H]x))>
4632         moreover AOT_have <□([G]ly (y = x & ∃H (x[H] & ¬[H]x)) →
4633           □ly (y = x & ∃H (x[H] & ¬[H]x))>↓>
4634       proof(rule RN; rule "→I")
4635         AOT_modally_strict {
4636           AOT_assume <[G]ly (y = x & ∃H (x[H] & ¬[H]x))>
4637           AOT_hence <ly (y = x & ∃H (x[H] & ¬[H]x))>↓>
4638             using "cqt:5:a"[axiom_inst, THEN "→E", THEN "&E"(2)] by blast
4639           AOT_thus <□ly (y = x & ∃H (x[H] & ¬[H]x))>↓>
4640             using "exist-nec"[THEN "→E"] by blast
4641         }
4642       qed
4643       ultimately AOT_have <A□ly (y = x & ∃H (x[H] & ¬[H]x))>↓>
4644         using "act-cond"[THEN "→E", THEN "→E"] "nec-imp-act"[THEN "→E"] by blast
4645       AOT_hence <ly (y = x & ∃H (x[H] & ¬[H]x))>↓>
4646         using "Act-Sub:3" "B◇" "vdash-properties:10" by blast
4647       AOT_thus <∃!y A(y = x & ∃H (x[H] & ¬[H]x))>
4648         using "actual-desc:1"[THEN "≐E"(1)] by blast
4649     next
4650     AOT_assume A: <∃!y A(y = x & ∃H (x[H] & ¬[H]x))>
4651     AOT_obtain a where a_1: <A(a = x & ∃H (x[H] & ¬[H]x))>
4652       and a_2: <∀z (A(z = x & ∃H (x[H] & ¬[H]x)) → z = a)>
4653       using "uniqueness:1"[THEN "≐dfE", OF A] "&E" "∃E"[rotated] by blast
4654     AOT_have a_3: <A[G]a>
4655       using antecedant "∀E" by blast
4656     moreover AOT_have <a = ly(y = x & ∃H (x[H] & ¬[H]x))>
4657       using "nec-hintikka-scheme"[THEN "≐E"(2), OF "&I"] a_1 a_2 by auto
4658     ultimately AOT_show <A[G]ly (y = x & ∃H (x[H] & ¬[H]x))>
4659     using "rule=E" by fast
4660   qed

```

(191.1)

```

4661 also AOT_have B: <...  $\equiv \mathcal{A}\exists H (x[H] \ \& \ \neg[H]x)$ >
4662 proof (rule "≡I"; rule "→I")
4663   AOT_assume A: < $\exists!y \ \mathcal{A}(y = x \ \& \ \exists H (x[H] \ \& \ \neg[H]x))$ >
4664   AOT_obtain a where < $\mathcal{A}(a = x \ \& \ \exists H (x[H] \ \& \ \neg[H]x))$ >
4665     using "uniqueness:1"[THEN "≡dfE", OF A] "&E" "∃E"[rotated] by blast
4666   AOT_thus < $\mathcal{A}\exists H (x[H] \ \& \ \neg[H]x)$ >
4667     using "Act-Basic:2"[THEN "≡E"(1), THEN "&E"(2)] by blast
4668 next
4669 AOT_assume < $\mathcal{A}\exists H (x[H] \ \& \ \neg[H]x)$ >
4670 AOT_hence < $\mathcal{A}x = x \ \& \ \mathcal{A}\exists H (x[H] \ \& \ \neg[H]x)$ >
4671   using "id-eq:1" "&I" "RA[2]" by blast
4672 AOT_hence < $\mathcal{A}(x = x \ \& \ \exists H (x[H] \ \& \ \neg[H]x))$ >
4673   using "act-conj-act:3" "Act-Basic:2" "≡E" by blast
4674 moreover AOT_have < $\forall z (\mathcal{A}(z = x \ \& \ \exists H (x[H] \ \& \ \neg[H]x)) \ \rightarrow \ z = x)$ >
4675 proof(safe intro!: GEN "→I")
4676   fix z
4677   AOT_assume < $\mathcal{A}(z = x \ \& \ \exists H (x[H] \ \& \ \neg[H]x))$ >
4678   AOT_hence < $\mathcal{A}(z = x)$ >
4679     using "Act-Basic:2"[THEN "≡E"(1), THEN "&E"(1)] by blast
4680   AOT_thus < $z = x$ >
4681     by (metis "id-act:1" "intro-elim:3:b")
4682 qed
4683 ultimately AOT_show < $\exists!y \ \mathcal{A}(y = x \ \& \ \exists H (x[H] \ \& \ \neg[H]x))$ >
4684   using "uniqueness:1"[THEN "≡dfI"] "&I" "∃I"(2) by fast
4685 qed
4686 finally AOT_show < $(\mathcal{A}[G]\iota y (y = x \ \& \ \exists H (x[H] \ \& \ \neg[H]x))) \equiv \mathcal{A}\exists H (x[H] \ \& \ \neg[H]x)$ >.
4687 qed
4688
4689 AOT_assume A: < $[\lambda x \ [G]\iota y (y = x \ \& \ \exists H (x[H] \ \& \ \neg[H]x))]\downarrow$ >
4690 AOT_hence < $\mathcal{A}[\lambda x \ [G]\iota y (y = x \ \& \ \exists H (x[H] \ \& \ \neg[H]x))]\downarrow$ >
4691   using "exist-nec" "→E" "nec-imp-act"[THEN "→E"] by blast
4692 AOT_hence < $\mathcal{A}([\lambda x \ [G]\iota y (y = x \ \& \ \exists H (x[H] \ \& \ \neg[H]x))]\downarrow \ \& \$ 
4693    $\forall x ([G]\iota y (y = x \ \& \ \exists H (x[H] \ \& \ \neg[H]x))) \equiv \exists H (x[H] \ \& \ \neg[H]x))$ >
4694   using Lemma "Act-Basic:2"[THEN "≡E"(2)] "&I" by blast
4695 moreover AOT_have < $\mathcal{A}([\lambda x \ [G]\iota y (y = x \ \& \ \exists H (x[H] \ \& \ \neg[H]x))]\downarrow \ \& \$ 
4696    $\forall x ([G]\iota y (y = x \ \& \ \exists H (x[H] \ \& \ \neg[H]x))) \equiv \exists H (x[H] \ \& \ \neg[H]x))$ 
4697    $\rightarrow \mathcal{A}\exists p (p \ \& \ \neg p)$ >
4698 proof (rule "logic-actual-nec:2"[axiom_inst, THEN "≡E"(1)];
4699   rule "RA[2]"; rule "→I")
4700 AOT_modally_strict {
4701   AOT_assume 0: < $[\lambda x \ [G]\iota y (y = x \ \& \ \exists H (x[H] \ \& \ \neg[H]x))]\downarrow \ \& \$ 
4702      $\forall x ([G]\iota y (y = x \ \& \ \exists H (x[H] \ \& \ \neg[H]x))) \equiv \exists H (x[H] \ \& \ \neg[H]x)$ >
4703   AOT_have < $\exists F \ \forall x ([F]x \equiv \exists G (x[G] \ \& \ \neg[G]x))$ >
4704   proof(rule "∃I"(1))
4705     AOT_show < $\forall x ([\lambda x \ [G]\iota y (y = x \ \& \ \exists H (x[H] \ \& \ \neg[H]x))]\downarrow \ \& \$ 
4706        $\forall x ([G]\iota y (y = x \ \& \ \exists H (x[H] \ \& \ \neg[H]x))) \equiv \exists H (x[H] \ \& \ \neg[H]x))$ >
4707     proof(safe intro!: GEN "≡I" "→I" "β←C" dest!: "β→C")
4708       fix x
4709       AOT_assume < $[G]\iota y (y = x \ \& \ \exists H (x[H] \ \& \ \neg[H]x))$ >
4710       AOT_thus < $\exists H (x[H] \ \& \ \neg[H]x)$ >
4711         using 0 "&E" "∀E"(2) "≡E"(1) by blast
4712     next
4713     fix x
4714     AOT_assume < $\exists H (x[H] \ \& \ \neg[H]x)$ >
4715     AOT_thus < $[G]\iota y (y = x \ \& \ \exists H (x[H] \ \& \ \neg[H]x))$ >
4716       using 0 "&E" "∀E"(2) "≡E"(2) by blast
4717   qed(auto intro!: 0[THEN "&E"(1)] "cqt:2")
4718 next
4719 AOT_show < $[\lambda x \ [G]\iota y (y = x \ \& \ \exists H (x[H] \ \& \ \neg[H]x))]\downarrow$ >
4720   using 0 "&E"(1) by blast
4721 qed
4722 AOT_thus < $\exists p (p \ \& \ \neg p)$ >
4723   using "block-paradox:2" "reductio-aa:1" by blast
4724 }

```

```

4724 qed
4725 ultimately AOT_have <math>\mathcal{A}\exists p (p \ \& \ \neg p)</math>
4726   using "→E" by blast
4727 AOT_hence <math>\exists p \ \mathcal{A}(p \ \& \ \neg p)</math>
4728   by (metis "Act-Basic:10" "intro-elim:3:a")
4729 then AOT_obtain p where <math>\mathcal{A}(p \ \& \ \neg p)</math>
4730   using "∃E"[rotated] by blast
4731 moreover AOT_have <math>\neg \mathcal{A}(p \ \& \ \neg p)</math>
4732   using "non-contradiction"[THEN "RA[2]"]
4733   by (meson "Act-Sub:1" "¬¬I" "intro-elim:3:d")
4734 ultimately AOT_show <math>p \ \& \ \neg p</math> for p
4735   by (metis "raa-cor:3")
4736 qed
4737
4738 AOT_act_theorem "block-paradox2:2": (191.2)
4739   <math>\langle \exists G \ \neg[\lambda x \ [G] \ \ell y \ (y = x \ \& \ \exists H \ (x[H] \ \& \ \neg[H]x))] \downarrow \rangle
4740 proof(rule "∃I"(1))
4741   AOT_have 0: <math>\langle [\lambda x \ \forall p \ (p \ \rightarrow p)] \downarrow \rangle
4742     by "cqt:2[lambda]"
4743 moreover AOT_have <math>\langle \forall x \ [\lambda x \ \forall p \ (p \ \rightarrow p)]x \rangle
4744   apply (rule GEN)
4745   apply (rule "beta-C-cor:2"[THEN "→E", OF 0, THEN "∃E"(2), THEN "≡E"(2)])
4746   using "if-p-then-p" GEN by fast
4747 moreover AOT_have <math>\langle \forall G \ (\forall x \ [G]x \ \rightarrow \ \neg[\lambda x \ [G] \ \ell y \ (y = x \ \& \ \exists H \ (x[H] \ \& \ \neg[H]x))] \downarrow) \rangle
4748   using "block-paradox2:1" "∀I" by fast
4749 ultimately AOT_show <math>\langle \neg[\lambda x \ [\lambda x \ \forall p \ (p \ \rightarrow p)] \ \ell y \ (y = x \ \& \ \exists H \ (x[H] \ \& \ \neg[H]x))] \downarrow \rangle
4750   using "∃E"(1) "→E" by blast
4751 qed("cqt:2[lambda]")
4752
4753 AOT_theorem propositions: <math>\langle \exists p \ \Box(p \ \equiv \ \varphi) \rangle (192)
4754 proof(rule "∃I"(1))
4755   AOT_show <math>\langle \Box(\varphi \ \equiv \ \varphi) \rangle
4756   by (simp add: RN "oth-class-taut:3:a")
4757 next
4758   AOT_show <math>\langle \varphi \downarrow \rangle
4759   by (simp add: "log-prop-prop:2")
4760 qed
4761
4762 AOT_theorem "pos-not-equiv-ne:1": (193.1)
4763   <math>\langle (\diamond \neg \forall x_1 \dots \forall x_n \ ([F]x_1 \dots x_n \ \equiv \ [G]x_1 \dots x_n)) \ \rightarrow \ F \neq G \rangle
4764 proof (rule "→I")
4765   AOT_assume <math>\langle \diamond \neg \forall x_1 \dots \forall x_n \ ([F]x_1 \dots x_n \ \equiv \ [G]x_1 \dots x_n) \rangle
4766   AOT_hence <math>\langle \neg \Box \forall x_1 \dots \forall x_n \ ([F]x_1 \dots x_n \ \equiv \ [G]x_1 \dots x_n) \rangle
4767   using "KBasic:11"[THEN "≡E"(2)] by blast
4768   AOT_hence <math>\langle \neg(F = G) \rangle
4769   using "id-rel-nec-equiv:1" "modus-tollens:1" by blast
4770   AOT_thus <math>\langle F \neq G \rangle
4771   using "=-infix"[THEN "≡dfI"] by blast
4772 qed
4773
4774 AOT_theorem "pos-not-equiv-ne:2": <math>\langle (\diamond \neg(\varphi\{F\} \ \equiv \ \varphi\{G\})) \ \rightarrow \ F \neq G \rangle (193.2)
4775 proof (rule "→I")
4776   AOT_modally_strict {
4777     AOT_have <math>\langle \neg(\varphi\{F\} \ \equiv \ \varphi\{G\}) \ \rightarrow \ \neg(F = G) \rangle
4778     proof (rule "→I"; rule "raa-cor:2")
4779       AOT_assume 1: <math>\langle F = G \rangle
4780       AOT_hence <math>\langle \varphi\{F\} \ \rightarrow \ \varphi\{G\} \rangle
4781       using "l-identity"[axiom_inst, THEN "→E"] by blast
4782       moreover {
4783         AOT_have <math>\langle G = F \rangle
4784         using 1 id_sym by blast
4785         AOT_hence <math>\langle \varphi\{G\} \ \rightarrow \ \varphi\{F\} \rangle
4786         using "l-identity"[axiom_inst, THEN "→E"] by blast

```

```

4787   }
4788   ultimately AOT_have <φ{F} ≡ φ{G}>
4789     using "≡I" by blast
4790   moreover AOT_assume <¬(φ{F} ≡ φ{G})>
4791   ultimately AOT_show <(φ{F} ≡ φ{G}) & ¬(φ{F} ≡ φ{G})>
4792     using "&I" by blast
4793   qed
4794 }
4795 AOT_hence <◇¬(φ{F} ≡ φ{G}) → ◇¬(F = G)>
4796   using "RM:2[prem]" by blast
4797 moreover AOT_assume <◇¬(φ{F} ≡ φ{G})>
4798 ultimately AOT_have 0: <◇¬(F = G)> using "→E" by blast
4799 AOT_have <◇(F ≠ G)>
4800   by (AOT_subst <F ≠ G <¬(F = G)>>
4801     (auto simp: "--infix" "≡Df" 0))
4802 AOT_thus <F ≠ G>
4803   using "id-nec2:3"[THEN "→E"] by blast
4804 qed
4805
4806 AOT_theorem "pos-not-equiv-ne:2[zero]": <(◇¬(φ{p} ≡ φ{q})) → p ≠ q> (193.2)
4807 proof (rule "→I")
4808   AOT_modally_strict {
4809     AOT_have <¬(φ{p} ≡ φ{q}) → ¬(p = q)>
4810     proof (rule "→I"; rule "raa-cor:2")
4811       AOT_assume 1: <p = q>
4812       AOT_hence <φ{p} → φ{q}>
4813         using "l-identity"[axiom_inst, THEN "→E"] by blast
4814       moreover {
4815         AOT_have <q = p>
4816           using 1 id_sym by blast
4817         AOT_hence <φ{q} → φ{p}>
4818           using "l-identity"[axiom_inst, THEN "→E"] by blast
4819       }
4820       ultimately AOT_have <φ{p} ≡ φ{q}>
4821         using "≡I" by blast
4822       moreover AOT_assume <¬(φ{p} ≡ φ{q})>
4823       ultimately AOT_show <(φ{p} ≡ φ{q}) & ¬(φ{p} ≡ φ{q})>
4824         using "&I" by blast
4825     qed
4826   }
4827 AOT_hence <◇¬(φ{p} ≡ φ{q}) → ◇¬(p = q)>
4828   using "RM:2[prem]" by blast
4829 moreover AOT_assume <◇¬(φ{p} ≡ φ{q})>
4830 ultimately AOT_have 0: <◇¬(p = q)> using "→E" by blast
4831 AOT_have <◇(p ≠ q)>
4832   by (AOT_subst <p ≠ q <¬(p = q)>>
4833     (auto simp: 0 "--infix" "≡Df"))
4834 AOT_thus <p ≠ q>
4835   using "id-nec2:3"[THEN "→E"] by blast
4836 qed
4837
4838 AOT_theorem "pos-not-equiv-ne:3": (193.3)
4839 <(¬∀x1...∀xn ([F]x1...xn ≡ [G]x1...xn)) → F ≠ G>
4840   using "→I" "pos-not-equiv-ne:1"[THEN "→E"] "T◇"[THEN "→E"] by blast
4841
4842 AOT_theorem "pos-not-equiv-ne:4": <(¬(φ{F} ≡ φ{G})) → F ≠ G> (193.4)
4843   using "→I" "pos-not-equiv-ne:2"[THEN "→E"] "T◇"[THEN "→E"] by blast
4844
4845 AOT_theorem "pos-not-equiv-ne:4[zero]": <(¬(φ{p} ≡ φ{q})) → p ≠ q> (193.4)
4846   using "→I" "pos-not-equiv-ne:2[zero]"[THEN "→E"]
4847     "T◇"[THEN "→E"] by blast
4848
4849 AOT_define relation_negation :: "II ⇒ II" ("¬")

```



```

4850   "df-relation-negation": "[F]- =df [λx1...xn ¬[F]x1...xn}]" (194)
4851
4852 nonterminal φneg
4853 syntax "" :: "φneg ⇒ τ" ("_")
4854 syntax "" :: "φneg ⇒ φ" ("'(_)'")
4855
4856 AOT_define relation_negation_0 :: <φ ⇒ φneg> ("'(_)'-")
4857   "df-relation-negation[zero]": "(p)- =df [λ ¬p]" (194)
4858
4859 AOT_theorem "rel-neg-T:1": <[λx1...xn ¬[[II]x1...xn}]↓> (195.1)
4860   by "cqt:2[lambda]"
4861
4862 AOT_theorem "rel-neg-T:1[zero]": <[λ ¬φ]↓> (195.1)
4863   using "cqt:2[lambda0]"[axiom_inst] by blast
4864
4865 AOT_theorem "rel-neg-T:2": <[[II]- = [λx1...xn ¬[[II]x1...xn}]> (195.2)
4866   using "=I"(1)[OF "rel-neg-T:1"]
4867   by (rule "=dfI"(1)[OF "df-relation-negation", OF "rel-neg-T:1"])
4868
4869 AOT_theorem "rel-neg-T:2[zero]": <(φ)- = [λ ¬φ]> (195.2)
4870   using "=I"(1)[OF "rel-neg-T:1[zero]"]
4871   by (rule "=dfI"(1)[OF "df-relation-negation[zero]", OF "rel-neg-T:1[zero]"])
4872
4873 AOT_theorem "rel-neg-T:3": <[[II]-↓> (195.3)
4874   using "=dfI"(1)[OF "df-relation-negation", OF "rel-neg-T:1"]
4875   "rel-neg-T:1" by blast
4876
4877 AOT_theorem "rel-neg-T:3[zero]": <(φ)-↓> (195.3)
4878   using "log-prop-prop:2" by blast
4879
4880 AOT_theorem "thm-relation-negation:1": <[F]-x1...xn ≡ ¬[F]x1...xn}> (197.1)
4881 proof -
4882   AOT_have <[F]-x1...xn ≡ [λx1...xn ¬[F]x1...xn}]x1...xn}>
4883     using "rule=E"[rotated, OF "rel-neg-T:2"]
4884     "rule=E"[rotated, OF "rel-neg-T:2"[THEN id_sym]]
4885     "→I" "≡I" by fast
4886   also AOT_have <... ≡ ¬[F]x1...xn}>
4887     using "beta-C-meta"[THEN "→E", OF "rel-neg-T:1"] by fast
4888   finally show ?thesis.
4889 qed
4890
4891 AOT_theorem "thm-relation-negation:2": <¬[F]-x1...xn ≡ [F]x1...xn}> (197.2)
4892   apply (AOT_subst <[F]x1...xn <¬¬[F]x1...xn}>)
4893   apply (simp add: "oth-class-taut:3:b")
4894   apply (rule "oth-class-taut:4:b"[THEN "≡E"(1)])
4895   using "thm-relation-negation:1".
4896
4897 AOT_theorem "thm-relation-negation:3": <((p)-) ≡ ¬p> (197.3)
4898 proof -
4899   AOT_have <(p)- = [λ ¬p]> using "rel-neg-T:2[zero]" by blast
4900   AOT_hence <((p)-) ≡ [λ ¬p]>
4901     using "df-relation-negation[zero]" "log-prop-prop:2"
4902     "oth-class-taut:3:a" "rule-id-df:2:a" by blast
4903   also AOT_have <[λ ¬p] ≡ ¬p>
4904     by (simp add: "propositions-lemma:2")
4905   finally show ?thesis.
4906 qed
4907
4908 AOT_theorem "thm-relation-negation:4": <(¬((p)-) ≡ p> (197.4)
4909   using "thm-relation-negation:3"[THEN "≡E"(1)]
4910   "thm-relation-negation:3"[THEN "≡E"(2)]
4911   "≡I" "→I" RAA by metis
4912

```



```

4913 AOT_theorem "thm-relation-negation:5": <[F] ≠ [F]-> (197.5)
4914 proof -
4915   AOT_have <¬([F] = [F]-)>
4916   proof (rule RAA(2))
4917     AOT_show <[F]x1...xn → [F]x1...xn> for x1xn
4918     using "if-p-then-p".
4919   next
4920     AOT_assume <[F] = [F]->
4921     AOT_hence <[F]- = [F]> using id_sym by blast
4922     AOT_hence <[F]x1...xn ≡ ¬[F]x1...xn> for x1xn
4923     using "rule=E" "thm-relation-negation:1" by fast
4924     AOT_thus <¬([F]x1...xn → [F]x1...xn)> for x1xn
4925     using "≡E" RAA by metis
4926   qed
4927   thus ?thesis
4928     using "≡dfI" "=-infix" by blast
4929 qed
4930
4931 AOT_theorem "thm-relation-negation:6": <p ≠ (p)-> (197.6)
4932 proof -
4933   AOT_have <¬(p = (p)-)>
4934   proof (rule RAA(2))
4935     AOT_show <p → p>
4936     using "if-p-then-p".
4937   next
4938     AOT_assume <p = (p)->
4939     AOT_hence <(p)- = p> using id_sym by blast
4940     AOT_hence <p ≡ ¬p>
4941     using "rule=E" "thm-relation-negation:3" by fast
4942     AOT_thus <¬(p → p)>
4943     using "≡E" RAA by metis
4944   qed
4945   thus ?thesis
4946     using "≡dfI" "=-infix" by blast
4947 qed
4948
4949 AOT_theorem "thm-relation-negation:7": <(p)- = (¬p)> (197.7)
4950 apply (rule "df-relation-negation[zero]"[THEN "≡dfE"(1)])
4951 using "cqt:2[lambda0]"[axiom_inst] "rel-neg-T:2[zero]"
4952 "propositions-lemma:1" id_trans by blast+
4953
4954 AOT_theorem "thm-relation-negation:8": <p = q → (¬p) = (¬q)> (197.8)
4955 proof(rule "→I")
4956   AOT_assume <p = q>
4957   moreover AOT_have <(¬p)↓> using "log-prop-prop:2".
4958   moreover AOT_have <(¬p) = (¬p)> using calculation(2) "=I" by blast
4959   ultimately AOT_show <(¬p) = (¬q)>
4960     using "rule=E" by fast
4961 qed
4962
4963 AOT_theorem "thm-relation-negation:9": <p = q → (p)- = (q)-> (197.9)
4964 proof(rule "→I")
4965   AOT_assume <p = q>
4966   AOT_hence <(¬p) = (¬q)> using "thm-relation-negation:8" "→E" by blast
4967   AOT_thus <(p)- = (q)->
4968     using "thm-relation-negation:7" id_sym id_trans by metis
4969 qed
4970
4971 AOT_define Necessary :: <Π ⇒ φ> ("Necessary'(_)'")
4972 "contingent-properties:1": (198.1)
4973 <Necessary([F]) ≡df □∀x1...∀xn [F]x1...xn>
4974
4975 AOT_define Necessary0 :: <φ ⇒ φ> ("Necessary0'(_)'")

```

```

4976 "contingent-properties:1[zero]": (198.1)
4977 <Necessary0(p) ≡df □p>
4978
4979 AOT_define Impossible :: <Π ⇒ φ> ("Impossible'(_)")
4980 "contingent-properties:2": (198.2)
4981 <Impossible([F]) ≡df F↓ & □∀x1...∀xn ¬[F]x1...xn>
4982
4983 AOT_define Impossible0 :: <φ ⇒ φ> ("Impossible0'(_)")
4984 "contingent-properties:2[zero]": (198.2)
4985 <Impossible0(p) ≡df □¬p>
4986
4987 AOT_define NonContingent :: <Π ⇒ φ> ("NonContingent'(_)")
4988 "contingent-properties:3": (198.3)
4989 <NonContingent([F]) ≡df Necessary([F]) ∨ Impossible([F])>
4990
4991 AOT_define NonContingent0 :: <φ ⇒ φ> ("NonContingent0'(_)")
4992 "contingent-properties:3[zero]": (198.3)
4993 <NonContingent0(p) ≡df Necessary0(p) ∨ Impossible0(p)>
4994
4995 AOT_define Contingent :: <Π ⇒ φ> ("Contingent'(_)")
4996 "contingent-properties:4": (198.4)
4997 <Contingent([F]) ≡df F↓ & ¬(Necessary([F]) ∨ Impossible([F]))>
4998
4999 AOT_define Contingent0 :: <φ ⇒ φ> ("Contingent0'(_)")
5000 "contingent-properties:4[zero]": (198.4)
5001 <Contingent0(p) ≡df ¬(Necessary0(p) ∨ Impossible0(p))>
5002
5003
5004 AOT_theorem "thm-cont-prop:1": <NonContingent([F]) ≡ NonContingent([F]~)> (200.1)
5005 proof (rule "≡I"; rule "→I")
5006   AOT_assume <NonContingent([F])>
5007   AOT_hence <Necessary([F]) ∨ Impossible([F])>
5008     using "≡dfE"[OF "contingent-properties:3"] by blast
5009   moreover {
5010     AOT_assume <Necessary([F])>
5011     AOT_hence <□(∀x1...∀xn [F]x1...xn)>
5012       using "≡dfE"[OF "contingent-properties:1"] by blast
5013     moreover AOT_modally_strict {
5014       AOT_assume <∀x1...∀xn [F]x1...xn>
5015       AOT_hence <[F]x1...xn> for x1xn using "∀E" by blast
5016       AOT_hence <¬[F]~x1...xn> for x1xn
5017         by (meson "≡E"(6) "oth-class-taut:3:a"
5018             "thm-relation-negation:2" "≡E"(1))
5019       AOT_hence <∀x1...∀xn ¬[F]~x1...xn> using "∀I" by fast
5020     }
5021     ultimately AOT_have <□(∀x1...∀xn ¬[F]~x1...xn)>
5022       using "RN[prem]"[where Γ="{«∀x1...∀xn [F]x1...xn»}]", simplified] by blast
5023     AOT_hence <Impossible([F]~)>
5024       using "≡Df"[OF "contingent-properties:2", THEN "≡S"(1),
5025             OF "rel-neg-T:3", THEN "≡E"(2)]
5026     by blast
5027   }
5028   moreover {
5029     AOT_assume <Impossible([F])>
5030     AOT_hence <□(∀x1...∀xn ¬[F]x1...xn)>
5031       using "≡Df"[OF "contingent-properties:2", THEN "≡S"(1),
5032             OF "cqt:2[const_var]"[axiom_inst], THEN "≡E"(1)]
5033     by blast
5034     moreover AOT_modally_strict {
5035       AOT_assume <∀x1...∀xn ¬[F]x1...xn>
5036       AOT_hence <¬[F]x1...xn> for x1xn using "∀E" by blast
5037       AOT_hence <[F]~x1...xn> for x1xn
5038       by (meson "≡E"(6) "oth-class-taut:3:a"

```

```

5039         "thm-relation-negation:1" "≡E"(1)
5040     AOT_hence <∀x1...∀xn [F]-x1...xn> using "∀I" by fast
5041 }
5042 ultimately AOT_have <□(∀x1...∀xn [F]-x1...xn)>
5043     using "RN[prem]" [where Γ="{«∀x1...∀xn ¬[F]-x1...xn»}"] by blast
5044 AOT_hence <Necessary([F]-)>
5045     using "≡dfI"[OF "contingent-properties:1"] by blast
5046 }
5047 ultimately AOT_have <Necessary([F]-) ∨ Impossible([F]-)>
5048     using "∨E"(1) "∀I" "→I" by metis
5049 AOT_thus <NonContingent([F]-)>
5050     using "≡dfI"[OF "contingent-properties:3"] by blast
5051 next
5052 AOT_assume <NonContingent([F]-)>
5053 AOT_hence <Necessary([F]-) ∨ Impossible([F]-)>
5054     using "≡dfE"[OF "contingent-properties:3"] by blast
5055 moreover {
5056     AOT_assume <Necessary([F]-)>
5057     AOT_hence <□(∀x1...∀xn [F]-x1...xn)>
5058         using "≡dfE"[OF "contingent-properties:1"] by blast
5059     moreover AOT_modally_strict {
5060         AOT_assume <∀x1...∀xn [F]-x1...xn>
5061         AOT_hence <[F]-x1...xn> for x1xn using "∨E" by blast
5062         AOT_hence <¬[F]-x1...xn> for x1xn
5063             by (meson "≡E"(6) "oth-class-taut:3:a"
5064                 "thm-relation-negation:1" "≡E"(2))
5065         AOT_hence <∀x1...∀xn ¬[F]-x1...xn> using "∀I" by fast
5066     }
5067     ultimately AOT_have <□∀x1...∀xn ¬[F]-x1...xn>
5068         using "RN[prem]" [where Γ="{«∀x1...∀xn [F]-x1...xn»}"] by blast
5069     AOT_hence <Impossible([F])>
5070         using "≡Df"[OF "contingent-properties:2", THEN "≡S"(1),
5071             OF "cqt:2[const_var]"[axiom_inst], THEN "≡E"(2)]
5072     by blast
5073 }
5074 moreover {
5075     AOT_assume <Impossible([F]-)>
5076     AOT_hence <□(∀x1...∀xn ¬[F]-x1...xn)>
5077         using "≡Df"[OF "contingent-properties:2", THEN "≡S"(1),
5078             OF "rel-neg-T:3", THEN "≡E"(1)]
5079     by blast
5080     moreover AOT_modally_strict {
5081         AOT_assume <∀x1...∀xn ¬[F]-x1...xn>
5082         AOT_hence <¬[F]-x1...xn> for x1xn using "∨E" by blast
5083         AOT_hence <[F]-x1...xn> for x1xn
5084             using "thm-relation-negation:1"[THEN
5085                 "oth-class-taut:4:b"[THEN "≡E"(1)], THEN "≡E"(1)]
5086                 "useful-tautologies:1"[THEN "→E"] by blast
5087         AOT_hence <∀x1...∀xn [F]-x1...xn> using "∀I" by fast
5088     }
5089     ultimately AOT_have <□(∀x1...∀xn [F]-x1...xn)>
5090         using "RN[prem]" [where Γ="{«∀x1...∀xn ¬[F]-x1...xn»}"] by blast
5091     AOT_hence <Necessary([F])>
5092         using "≡dfI"[OF "contingent-properties:1"] by blast
5093 }
5094 ultimately AOT_have <Necessary([F]) ∨ Impossible([F])>
5095     using "∨E"(1) "∀I" "→I" by metis
5096 AOT_thus <NonContingent([F])>
5097     using "≡dfI"[OF "contingent-properties:3"] by blast
5098 qed
5099
5100 AOT_theorem "thm-cont-prop:2": <Contingent([F]) ≡ ◇∃x [F]-x & ◇∃x ¬[F]-x> (200.2)
5101 proof -

```

```

5102 AOT_have <Contingent([F]) ≡ ¬(Necessary([F]) ∨ Impossible([F]))>
5103   using "contingent-properties:4"[THEN "≡Df", THEN "≡S"(1),
5104         OF "cqt:2[const_var]"[axiom_inst]]
5105   by blast
5106 also AOT_have <... ≡ ¬Necessary([F]) & ¬Impossible([F])>
5107   using "oth-class-taut:5:d" by fastforce
5108 also AOT_have <... ≡ ¬Impossible([F]) & ¬Necessary([F])>
5109   by (simp add: "Commutativity of &")
5110 also AOT_have <... ≡ ◇∃x [F]x & ¬Necessary([F])>
5111 proof (rule "oth-class-taut:4:e"[THEN "→E"])
5112   AOT_have <¬Impossible([F]) ≡ ¬□¬ ∃x [F]x>
5113     apply (rule "oth-class-taut:4:b"[THEN "≡E"(1)])
5114     apply (AOT_subst <∃x [F]x> <¬ ∀x ¬[F]x>)
5115     apply (simp add: "conventions:4" "≡Df")
5116     apply (AOT_subst (reverse) <¬¬∀x ¬[F]x> <∀x ¬[F]x>)
5117     apply (simp add: "oth-class-taut:3:b")
5118     using "contingent-properties:2"[THEN "≡Df", THEN "≡S"(1),
5119           OF "cqt:2[const_var]"[axiom_inst]]
5120   by blast
5121 also AOT_have <... ≡ ◇∃x [F]x>
5122   using "conventions:5"[THEN "≡Df", symmetric] by blast
5123 finally AOT_show <¬Impossible([F]) ≡ ◇∃x [F]x> .
5124 qed
5125 also AOT_have <... ≡ ◇∃x [F]x & ◇∃x ¬[F]x>
5126 proof (rule "oth-class-taut:4:f"[THEN "→E"])
5127   AOT_have <¬Necessary([F]) ≡ ¬□¬∃x ¬[F]x>
5128     apply (rule "oth-class-taut:4:b"[THEN "≡E"(1)])
5129     apply (AOT_subst <∃x ¬[F]x> <¬ ∀x ¬¬[F]x>)
5130     apply (simp add: "conventions:4" "≡Df")
5131     apply (AOT_subst (reverse) <¬¬[F]x> <[F]x> for: x)
5132     apply (simp add: "oth-class-taut:3:b")
5133     apply (AOT_subst (reverse) <¬¬∀x [F]x> <∀x [F]x>)
5134     by (auto simp: "oth-class-taut:3:b" "contingent-properties:1" "≡Df")
5135 also AOT_have <... ≡ ◇∃x ¬[F]x>
5136   using "conventions:5"[THEN "≡Df", symmetric] by blast
5137 finally AOT_show <¬Necessary([F]) ≡ ◇∃x ¬[F]x>.
5138 qed
5139 finally show ?thesis.
5140 qed
5141
5142 AOT_theorem "thm-cont-prop:3":
5143   <Contingent([F]) ≡ Contingent([F]⁻)> for F::<<κ> AOT_var
5144 proof -
5145   {
5146     fix II :: <<κ>>
5147     AOT_assume <II↓>
5148     moreover AOT_have <∀F (Contingent([F]) ≡ ◇∃x [F]x & ◇∃x ¬[F]x)>
5149       using "thm-cont-prop:2" GEN by fast
5150     ultimately AOT_have <Contingent([II]) ≡ ◇∃x [II]x & ◇∃x ¬[II]x>
5151       using "thm-cont-prop:2" "∀E" by fast
5152   } note 1 = this
5153 AOT_have <Contingent([F]) ≡ ◇∃x [F]x & ◇∃x ¬[F]x>
5154   using "thm-cont-prop:2" by blast
5155 also AOT_have <... ≡ ◇∃x ¬[F]x & ◇∃x [F]x>
5156   by (simp add: "Commutativity of &")
5157 also AOT_have <... ≡ ◇∃x [F]⁻x & ◇∃x [F]x>
5158   by (AOT_subst <[F]⁻x> <¬[F]x> for: x)
5159     (auto simp: "thm-relation-negation:1" "oth-class-taut:3:a")
5160 also AOT_have <... ≡ ◇∃x [F]⁻x & ◇∃x ¬[F]⁻x>
5161   by (AOT_subst (reverse) <[F]x> <¬[F]⁻x> for: x)
5162     (auto simp: "thm-relation-negation:2" "oth-class-taut:3:a")
5163 also AOT_have <... ≡ Contingent([F]⁻)>
5164   using 1[OF "rel-neg-T:3", symmetric] by blast

```

(200.3)

```

5165   finally show ?thesis.
5166 qed
5167
5168 AOT_define concrete_if_concrete :: <Π> ("L")
5169   L_def: <L =df [λx E!x → E!x]>
5170
5171 AOT_theorem "thm-noncont-e-e:1": <Necessary(L)> (201.1)
5172 proof -
5173   AOT_modally_strict {
5174     fix x
5175     AOT_have <[λx E!x → E!x]↓> by "cqt:2[lambda]"
5176     moreover AOT_have <x↓> using "cqt:2[const_var]"[axiom_inst] by blast
5177     moreover AOT_have <E!x → E!x> using "if-p-then-p" by blast
5178     ultimately AOT_have <[λx E!x → E!x]x>
5179       using "β←C" by blast
5180   }
5181   AOT_hence 0: <□∀x [λx E!x → E!x]x>
5182     using RN GEN by blast
5183   show ?thesis
5184     apply (rule "=dfI"(2)[OF L_def])
5185     apply "cqt:2[lambda]"
5186     by (rule "contingent-properties:1"[THEN "≡dfI", OF 0])
5187 qed
5188
5189 AOT_theorem "thm-noncont-e-e:2": <Impossible([L]~)> (201.2)
5190 proof -
5191   AOT_modally_strict {
5192     fix x
5193
5194     AOT_have 0: <∀F (¬[F]~x ≡ [F]x)>
5195       using "thm-relation-negation:2" GEN by fast
5196     AOT_have <¬[λx E!x → E!x]~x ≡ [λx E!x → E!x]x>
5197       by (rule 0[THEN "∀E"(1)]) "cqt:2[lambda]"
5198     moreover {
5199       AOT_have <[λx E!x → E!x]↓> by "cqt:2[lambda]"
5200       moreover AOT_have <x↓> using "cqt:2[const_var]"[axiom_inst] by blast
5201       moreover AOT_have <E!x → E!x> using "if-p-then-p" by blast
5202       ultimately AOT_have <[λx E!x → E!x]x>
5203         using "β←C" by blast
5204     }
5205     ultimately AOT_have <¬[λx E!x → E!x]~x>
5206       using "≡E" by blast
5207   }
5208   AOT_hence 0: <□∀x ¬[λx E!x → E!x]~x>
5209     using RN GEN by fast
5210   show ?thesis
5211     apply (rule "=dfI"(2)[OF L_def])
5212     apply "cqt:2[lambda]"
5213     apply (rule "contingent-properties:2"[THEN "≡dfI"]; rule "&I")
5214     using "rel-neg-T:3"
5215     apply blast
5216     using 0
5217     by blast
5218 qed
5219
5220 AOT_theorem "thm-noncont-e-e:3": <NonContingent(L)> (201.3)
5221   using "thm-noncont-e-e:1"
5222   by (rule "contingent-properties:3"[THEN "≡dfI", OF "∀I"(1)])
5223
5224 AOT_theorem "thm-noncont-e-e:4": <NonContingent([L]~)> (201.4)
5225 proof -
5226   AOT_have 0: <∀F (NonContingent([F]) ≡ NonContingent([F]~))>
5227     using "thm-cont-prop:1" "∀I" by fast

```

```

5228   moreover AOT_have 1: <L↓>
5229     by (rule "=dfI"(2)[OF L_def]) "cqt:2[lambda]" +
5230   AOT_show <NonContingent([L]⁻)>
5231     using "∀E"(1)[OF 0, OF 1, THEN "≡E"(1), OF "thm-noncont-e-e:3"] by blast
5232 qed
5233
5234 AOT_theorem "thm-noncont-e-e:5":
5235   <∃F ∃G (F ≠ «G::<κ>» & NonContingent([F]) & NonContingent([G]))>
5236 proof (rule "∃I")+
5237   {
5238     AOT_have <∀F [F] ≠ [F]⁻>
5239       using "thm-relation-negation:5" GEN by fast
5240     moreover AOT_have <L↓>
5241       by (rule "=dfI"(2)[OF L_def]) "cqt:2[lambda]" +
5242     ultimately AOT_have <L ≠ [L]⁻>
5243       using "∀E" by blast
5244   }
5245   AOT_thus <L ≠ [L]⁻ & NonContingent(L) & NonContingent([L]⁻)>
5246     using "thm-noncont-e-e:3" "thm-noncont-e-e:4" "&I" by metis
5247 next
5248   AOT_show <[L]⁻↓>
5249     using "rel-neg-T:3" by blast
5250 next
5251   AOT_show <L↓>
5252     by (rule "=dfI"(2)[OF L_def]) "cqt:2[lambda]" +
5253 qed
5254
5255 AOT_theorem "lem-cont-e:1": <◇∃x ([F]x & ◇¬[F]x) ≡ ◇∃x (¬[F]x & ◇[F]x)>
5256 proof -
5257   AOT_have <◇∃x ([F]x & ◇¬[F]x) ≡ ∃x ◇([F]x & ◇¬[F]x)>
5258     using "BF◇" "CBF◇" "≡I" by blast
5259   also AOT_have <... ≡ ∃x (◇[F]x & ◇¬[F]x)>
5260     by (AOT_subst <◇([F]x & ◇¬[F]x)> <◇[F]x & ◇¬[F]x> for: x)
5261       (auto simp: "S5Basic:11" "cqt-further:7")
5262   also AOT_have <... ≡ ∃x (◇¬[F]x & ◇[F]x)>
5263     by (AOT_subst <◇¬[F]x & ◇[F]x> <◇[F]x & ◇¬[F]x> for: x)
5264       (auto simp: "Commutativity of &" "cqt-further:7")
5265   also AOT_have <... ≡ ∃x ◇(¬[F]x & ◇[F]x)>
5266     by (AOT_subst <◇(¬[F]x & ◇[F]x)> <◇¬[F]x & ◇[F]x> for: x)
5267       (auto simp: "S5Basic:11" "oth-class-taut:3:a")
5268   also AOT_have <... ≡ ◇∃x (¬[F]x & ◇[F]x)>
5269     using "BF◇" "CBF◇" "≡I" by fast
5270   finally show ?thesis.
5271 qed
5272
5273 AOT_theorem "lem-cont-e:2":
5274   <◇∃x ([F]x & ◇¬[F]x) ≡ ◇∃x ([F]⁻x & ◇¬[F]⁻x)>
5275 proof -
5276   AOT_have <◇∃x ([F]x & ◇¬[F]x) ≡ ◇∃x (¬[F]x & ◇[F]x)>
5277     using "lem-cont-e:1".
5278   also AOT_have <... ≡ ◇∃x ([F]⁻x & ◇¬[F]⁻x)>
5279     apply (AOT_subst <¬[F]⁻x> <[F]x> for: x)
5280     apply (simp add: "thm-relation-negation:2")
5281     apply (AOT_subst <[F]⁻x> <¬[F]x> for: x)
5282     apply (simp add: "thm-relation-negation:1")
5283     by (simp add: "oth-class-taut:3:a")
5284   finally show ?thesis.
5285 qed
5286
5287 AOT_theorem "thm-cont-e:1": <◇∃x (E!x & ◇¬E!x)>
5288 proof (rule "CBF◇"[THEN "→E"])
5289   AOT_have <∃x ◇(E!x & ¬AE!x)>
5290     using "qml:4"[axiom_inst] "BF◇"[THEN "→E"] by blast

```

```

5291 then AOT_obtain a where <◇(E!a & ¬A!a)>
5292   using "∃E"[rotated] by blast
5293 AOT_hence ϑ: <◇E!a & ◇¬A!a>
5294   using "KBasic2:3"[THEN "→E"] by blast
5295 AOT_have ξ: <◇E!a & ◇A¬E!a>
5296   by (AOT_subst <A¬E!a> <¬A!a>)
5297     (auto simp: "logic-actual-nec:1"[axiom_inst] ϑ)
5298 AOT_have ζ: <◇E!a & A¬E!a>
5299   by (AOT_subst <A¬E!a> <◇A¬E!a>)
5300     (auto simp add: "Act-Sub:4" ξ)
5301 AOT_hence <◇E!a & ◇¬E!a>
5302   using "&E" "&I" "Act-Sub:3"[THEN "→E"] by blast
5303 AOT_hence <◇(E!a & ◇¬E!a)>
5304   using "S5Basic:11"[THEN "≡E"(2)] by simp
5305 AOT_thus <∃x ◇(E!x & ◇¬E!x)>
5306   using "∃I"(2) by fast
5307 qed
5308
5309 AOT_theorem "thm-cont-e:2": <◇∃x (¬E!x & ◇E!x)> (203.2)
5310 proof -
5311   AOT_have <∀F (◇∃x ([F]x & ◇¬[F]x) ≡ ◇∃x (¬[F]x & ◇[F]x))>
5312     using "lem-cont-e:1" GEN by fast
5313   AOT_hence <(◇∃x (E!x & ◇¬E!x) ≡ ◇∃x (¬E!x & ◇E!x))>
5314     using "∀E"(2) by blast
5315   thus ?thesis using "thm-cont-e:1" "≡E" by blast
5316 qed
5317
5318 AOT_theorem "thm-cont-e:3": <◇∃x E!x> (203.3)
5319 proof (rule "CBF◇"[THEN "→E"])
5320   AOT_obtain a where <◇(E!a & ◇¬E!a)>
5321     using "∃E"[rotated, OF "thm-cont-e:1"[THEN "BF◇"[THEN "→E"]]] by blast
5322   AOT_hence <◇E!a>
5323     using "KBasic2:3"[THEN "→E", THEN "&E"(1)] by blast
5324   AOT_thus <∃x ◇E!x> using "∃I" by fast
5325 qed
5326
5327 AOT_theorem "thm-cont-e:4": <◇∃x ¬E!x> (203.4)
5328 proof (rule "CBF◇"[THEN "→E"])
5329   AOT_obtain a where <◇(E!a & ◇¬E!a)>
5330     using "∃E"[rotated, OF "thm-cont-e:1"[THEN "BF◇"[THEN "→E"]]] by blast
5331   AOT_hence <◇◇¬E!a>
5332     using "KBasic2:3"[THEN "→E", THEN "&E"(2)] by blast
5333   AOT_hence <◇¬E!a>
5334     using "4◇"[THEN "→E"] by blast
5335   AOT_thus <∃x ◇¬E!x> using "∃I" by fast
5336 qed
5337
5338 AOT_theorem "thm-cont-e:5": <Contingent([E!])> (203.5)
5339 proof -
5340   AOT_have <∀F (Contingent([F]) ≡ ◇∃x [F]x & ◇∃x ¬[F]x)>
5341     using "thm-cont-prop:2" GEN by fast
5342   AOT_hence <Contingent([E!]) ≡ ◇∃x E!x & ◇∃x ¬E!x>
5343     using "∀E"(2) by blast
5344   thus ?thesis
5345     using "thm-cont-e:3" "thm-cont-e:4" "≡E"(2) "&I" by blast
5346 qed
5347
5348 AOT_theorem "thm-cont-e:6": <Contingent([E!]~)> (203.6)
5349 proof -
5350   AOT_have <∀F (Contingent([«F::<κ>»]) ≡ Contingent([F]~))>
5351     using "thm-cont-prop:3" GEN by fast
5352   AOT_hence <Contingent([E!]) ≡ Contingent([E!]~)>
5353     using "∀E"(2) by fast

```



```

5354   thus ?thesis using "thm-cont-e:5" "≡E" by blast
5355 qed
5356
5357 AOT_theorem "thm-cont-e:7": (203.7)
5358   <∃F∃G (Contingent(⟦F::<κ>⟧) & Contingent(⟦G⟧) & F ≠ G)>
5359 proof (rule "∃I")+
5360   AOT_have <∀F [⟦F::<κ>⟧] ≠ [F]⁻>
5361     using "thm-relation-negation:5" GEN by fast
5362   AOT_hence <[E!] ≠ [E!]⁻>
5363     using "∀E" by fast
5364   AOT_thus <Contingent([E!]) & Contingent([E!]⁻) & [E!] ≠ [E!]⁻>
5365     using "thm-cont-e:5" "thm-cont-e:6" "&I" by metis
5366 next
5367   AOT_show <E!⁻↓>
5368     by (fact AOT)
5369 qed("cqt:2")
5370
5371 AOT_theorem "property-facts:1": (204.1)
5372   <NonContingent(⟦F⟧) → ¬∃G (Contingent(⟦G⟧) & G = F)>
5373 proof (rule "→I"; rule "raa-cor:2")
5374   AOT_assume <NonContingent(⟦F⟧)>
5375   AOT_hence 1: <Necessary(⟦F⟧) ∨ Impossible(⟦F⟧)>
5376     using "contingent-properties:3"[THEN "≡dfE"] by blast
5377   AOT_assume <∃G (Contingent(⟦G⟧) & G = F)>
5378   then AOT_obtain G where <Contingent(⟦G⟧) & G = F>
5379     using "∃E"[rotated] by blast
5380   AOT_hence <Contingent(⟦F⟧)> using "rule=E" "&E" by blast
5381   AOT_hence <¬(Necessary(⟦F⟧) ∨ Impossible(⟦F⟧))>
5382     using "contingent-properties:4"[THEN "≡Df", THEN "≡S"(1),
5383       OF "cqt:2[const_var]"[axiom_inst], THEN "≡E"(1)] by blast
5384   AOT_thus <(Necessary(⟦F⟧) ∨ Impossible(⟦F⟧)) &
5385     ¬(Necessary(⟦F⟧) ∨ Impossible(⟦F⟧))>
5386     using 1 "&I" by blast
5387 qed
5388
5389 AOT_theorem "property-facts:2": (204.2)
5390   <Contingent(⟦F⟧) → ¬∃G (NonContingent(⟦G⟧) & G = F)>
5391 proof (rule "→I"; rule "raa-cor:2")
5392   AOT_assume <Contingent(⟦F⟧)>
5393   AOT_hence 1: <¬(Necessary(⟦F⟧) ∨ Impossible(⟦F⟧))>
5394     using "contingent-properties:4"[THEN "≡Df", THEN "≡S"(1),
5395       OF "cqt:2[const_var]"[axiom_inst], THEN "≡E"(1)] by blast
5396   AOT_assume <∃G (NonContingent(⟦G⟧) & G = F)>
5397   then AOT_obtain G where <NonContingent(⟦G⟧) & G = F>
5398     using "∃E"[rotated] by blast
5399   AOT_hence <NonContingent(⟦F⟧)>
5400     using "rule=E" "&E" by blast
5401   AOT_hence <Necessary(⟦F⟧) ∨ Impossible(⟦F⟧)>
5402     using "contingent-properties:3"[THEN "≡dfE"] by blast
5403   AOT_thus <(Necessary(⟦F⟧) ∨ Impossible(⟦F⟧)) &
5404     ¬(Necessary(⟦F⟧) ∨ Impossible(⟦F⟧))>
5405     using 1 "&I" by blast
5406 qed
5407
5408 AOT_theorem "property-facts:3": (204.3)
5409   <L ≠ [L]⁻ & L ≠ E! & L ≠ E!⁻ & [L]⁻ ≠ [E!]⁻ & E! ≠ [E!]⁻>
5410 proof -
5411   AOT_have noneqI: <Π ≠ Π'> if <φ{Π}> and <¬φ{Π'}> for φ and Π Π' :: <<κ>>
5412     apply (rule "-infix"[THEN "≡dfI"]; rule "raa-cor:2")
5413     using "rule=E"[where φ=φ and τ=Π and σ = Π'] that "&I" by blast
5414   AOT_have contingent_denotes: <Π↓> if <Contingent(⟦Π⟧)> for Π :: <<κ>>
5415     using that "contingent-properties:4"[THEN "≡dfE", THEN "&E"(1)] by blast
5416   AOT_have not_noncontingent_if_contingent:

```



```

5417   <¬NonContingent([II])> if <Contingent([II])> for II :: <<κ>>
5418 proof(rule RAA(2))
5419   AOT_show <¬(Necessary([II]) ∨ Impossible([II]))>
5420     using that "contingent-properties:4"[THEN "≡Df", THEN "≡S"(1),
5421               OF contingent_denotes[OF that], THEN "≡E"(1)]
5422     by blast
5423 next
5424   AOT_assume <NonContingent([II])>
5425   AOT_thus <Necessary([II]) ∨ Impossible([II])>
5426     using "contingent-properties:3"[THEN "≡dfE"] by blast
5427 qed
5428
5429 show ?thesis
5430 proof (safe intro!: "&I")
5431   AOT_show <L ≠ [L]->
5432     apply (rule "=dfI"(2)[OF L_def])
5433     apply "cqt:2[lambda]"
5434     apply (rule "∀E"(1)[where φ="λ II . «II ≠ [II]-»"])
5435     apply (rule GEN) apply (fact AOT)
5436     by "cqt:2[lambda]"
5437 next
5438   AOT_show <L ≠ E!->
5439     apply (rule noneqI)
5440     using "thm-noncont-e-e:3"
5441       not_noncontingent_if_contingent[OF "thm-cont-e:5"]
5442     by auto
5443 next
5444   AOT_show <L ≠ E!->
5445     apply (rule noneqI)
5446     using "thm-noncont-e-e:3" apply fast
5447     apply (rule not_noncontingent_if_contingent)
5448     apply (rule "∀E"(1)[
5449       where φ="λ II . «Contingent([II]) ≡ Contingent([II]-)»,
5450       rotated, OF contingent_denotes, THEN "≡E"(1), rotated])
5451     using "thm-cont-prop:3" GEN apply fast
5452     using "thm-cont-e:5" by fast+
5453 next
5454   AOT_show <[L]- ≠ E!->
5455     apply (rule noneqI)
5456     using "thm-noncont-e-e:4" apply fast
5457     apply (rule not_noncontingent_if_contingent)
5458     apply (rule "∀E"(1)[
5459       where φ="λ II . «Contingent([II]) ≡ Contingent([II]-)»,
5460       rotated, OF contingent_denotes, THEN "≡E"(1), rotated])
5461     using "thm-cont-prop:3" GEN apply fast
5462     using "thm-cont-e:5" by fast+
5463 next
5464   AOT_show <E! ≠ E!->
5465     apply (rule "=dfI"(2)[OF L_def])
5466     apply "cqt:2[lambda]"
5467     apply (rule "∀E"(1)[where φ="λ II . «II ≠ [II]-»"])
5468     apply (rule GEN) apply (fact AOT)
5469     by "cqt:2"
5470 qed
5471 qed
5472
5473 AOT_theorem "thm-cont-propos:1":
5474   <NonContingent0(p) ≡ NonContingent0((p)-)>
5475 proof(rule "≡I"; rule "→I")
5476   AOT_assume <NonContingent0(p)>
5477   AOT_hence <Necessary0(p) ∨ Impossible0(p)>
5478     using "contingent-properties:3[zero]"[THEN "≡dfE"] by blast
5479   moreover {

```

(205.1)

```

5480   AOT_assume <Necessary0(p)>
5481   AOT_hence 1: <□p>
5482     using "contingent-properties:1[zero]" [THEN "≡dfE"] by blast
5483   AOT_have <□¬((p)⁻)>
5484     by (AOT_subst <¬((p)⁻)> <p>)
5485     (auto simp add: 1 "thm-relation-negation:4")
5486   AOT_hence <Impossible0((p)⁻)>
5487     by (rule "contingent-properties:2[zero]" [THEN "≡dfI"])
5488 }
5489 moreover {
5490   AOT_assume <Impossible0(p)>
5491   AOT_hence 1: <□¬p>
5492     by (rule "contingent-properties:2[zero]" [THEN "≡dfE"])
5493   AOT_have <□((p)⁻)>
5494     by (AOT_subst <((p)⁻)> <¬p>)
5495     (auto simp: 1 "thm-relation-negation:3")
5496   AOT_hence <Necessary0((p)⁻)>
5497     by (rule "contingent-properties:1[zero]" [THEN "≡dfI"])
5498 }
5499 ultimately AOT_have <Necessary0((p)⁻) ∨ Impossible0((p)⁻)>
5500   using "∨E"(1) "∨I" "→I" by metis
5501 AOT_thus <NonContingent0((p)⁻)>
5502   using "contingent-properties:3[zero]" [THEN "≡dfI"] by blast
5503 next
5504 AOT_assume <NonContingent0((p)⁻)>
5505 AOT_hence <Necessary0((p)⁻) ∨ Impossible0((p)⁻)>
5506   using "contingent-properties:3[zero]" [THEN "≡dfE"] by blast
5507 moreover {
5508   AOT_assume <Impossible0((p)⁻)>
5509   AOT_hence 1: <□¬((p)⁻)>
5510     by (rule "contingent-properties:2[zero]" [THEN "≡dfE"])
5511   AOT_have <□p>
5512     by (AOT_subst (reverse) <p> <¬((p)⁻)>)
5513     (auto simp: 1 "thm-relation-negation:4")
5514   AOT_hence <Necessary0(p)>
5515     using "contingent-properties:1[zero]" [THEN "≡dfI"] by blast
5516 }
5517 moreover {
5518   AOT_assume <Necessary0((p)⁻)>
5519   AOT_hence 1: <□((p)⁻)>
5520     by (rule "contingent-properties:1[zero]" [THEN "≡dfE"])
5521   AOT_have <□¬p>
5522     by (AOT_subst (reverse) <¬p> <((p)⁻)>)
5523     (auto simp: 1 "thm-relation-negation:3")
5524   AOT_hence <Impossible0(p)>
5525     by (rule "contingent-properties:2[zero]" [THEN "≡dfI"])
5526 }
5527 ultimately AOT_have <Necessary0(p) ∨ Impossible0(p)>
5528   using "∨E"(1) "∨I" "→I" by metis
5529 AOT_thus <NonContingent0(p)>
5530   using "contingent-properties:3[zero]" [THEN "≡dfI"] by blast
5531 qed
5532
5533 AOT_theorem "thm-cont-propos:2": <Contingent0(φ) ≡ ◇φ & ◇¬φ> (205.2)
5534 proof -
5535   AOT_have <Contingent0(φ) ≡ ¬(Necessary0(φ) ∨ Impossible0(φ))>
5536     using "contingent-properties:4[zero]" [THEN "≡Df"] by simp
5537   also AOT_have <... ≡ ¬Necessary0(φ) & ¬Impossible0(φ)>
5538     by (fact AOT)
5539   also AOT_have <... ≡ ¬Impossible0(φ) & ¬Necessary0(φ)>
5540     by (fact AOT)
5541   also AOT_have <... ≡ ◇φ & ◇¬φ>
5542     apply (AOT_subst <◇φ> <¬□¬φ>)

```

```

5543     apply (simp add: "conventions:5" "≡Df")
5544   apply (AOT_subst <Impossible0( $\varphi$ )> < $\Box\neg\varphi$ >)
5545     apply (simp add: "contingent-properties:2[zero]" "≡Df")
5546   apply (AOT_subst (reverse) < $\Diamond\neg\varphi$ > < $\neg\Box\varphi$ >)
5547     apply (simp add: "KBasic:11")
5548   apply (AOT_subst <Necessary0( $\varphi$ )> < $\Box\varphi$ >)
5549     apply (simp add: "contingent-properties:1[zero]" "≡Df")
5550   by (simp add: "oth-class-taut:3:a")
5551   finally show ?thesis.
5552 qed
5553
5554 AOT_theorem "thm-cont-propos:3": <Contingent0( $p$ )  $\equiv$  Contingent0( $((p)^{\sim})$ )> (205.3)
5555 proof -
5556   AOT_have <Contingent0( $p$ )  $\equiv$   $\Diamond p$  &  $\Diamond\neg p$ > using "thm-cont-propos:2".
5557   also AOT_have < $\dots \equiv \Diamond\neg p$  &  $\Diamond p$ > by (fact AOT)
5558   also AOT_have < $\dots \equiv \Diamond((p)^{\sim})$  &  $\Diamond p$ >
5559     by (AOT_subst < $((p)^{\sim})$ > < $\neg p$ >)
5560       (auto simp: "thm-relation-negation:3" "oth-class-taut:3:a")
5561   also AOT_have < $\dots \equiv \Diamond((p)^{\sim})$  &  $\Diamond\neg((p)^{\sim})$ >
5562     by (AOT_subst < $\neg((p)^{\sim})$ > < $p$ >)
5563       (auto simp: "thm-relation-negation:4" "oth-class-taut:3:a")
5564   also AOT_have < $\dots \equiv$  Contingent0( $((p)^{\sim})$ )>
5565     using "thm-cont-propos:2"[symmetric] by blast
5566   finally show ?thesis.
5567 qed
5568
5569 AOT_define noncontingent_prop :: < $\varphi$ > ("p0")
5570   p0_def: "(p0) =df ( $\forall x (E!x \rightarrow E!x)$ )"
5571
5572 AOT_theorem "thm-noncont-propos:1": <Necessary0( $(p_0)$ )> (206.1)
5573 proof(rule "contingent-properties:1[zero]" [THEN "≡dfI"])
5574   AOT_show < $\Box(p_0)$ >
5575     apply (rule "=dfI"(2)[OF p0_def])
5576     using "log-prop-prop:2" apply simp
5577     using "if-p-then-p" RN GEN by fast
5578 qed
5579
5580 AOT_theorem "thm-noncont-propos:2": <Impossible0( $((p_0)^{\sim})$ )> (206.2)
5581 proof(rule "contingent-properties:2[zero]" [THEN "≡dfI"])
5582   AOT_show < $\Box\neg((p_0)^{\sim})$ >
5583     apply (AOT_subst < $((p_0)^{\sim})$ > < $\neg p_0$ >)
5584     using "thm-relation-negation:3" GEN "\ $\forall E$ "(1)[rotated, OF "log-prop-prop:2"]
5585     apply fast
5586     apply (AOT_subst (reverse) < $\neg\neg p_0$ > < $p_0$ >)
5587     apply (simp add: "oth-class-taut:3:b")
5588     apply (rule "=dfI"(2)[OF p0_def])
5589     using "log-prop-prop:2" apply simp
5590     using "if-p-then-p" RN GEN by fast
5591 qed
5592
5593 AOT_theorem "thm-noncont-propos:3": <NonContingent0( $(p_0)$ )> (206.3)
5594   apply(rule "contingent-properties:3[zero]" [THEN "≡dfI"])
5595   using "thm-noncont-propos:1" "\ $\forall I$ " by blast
5596
5597 AOT_theorem "thm-noncont-propos:4": <NonContingent0( $((p_0)^{\sim})$ )> (206.4)
5598   apply(rule "contingent-properties:3[zero]" [THEN "≡dfI"])
5599   using "thm-noncont-propos:2" "\ $\forall I$ " by blast
5600
5601 AOT_theorem "thm-noncont-propos:5": (206.5)
5602   < $\exists p\exists q (NonContingent0((p)) \& NonContingent0((q)) \& p \neq q)$ >
5603 proof(rule "\ $\exists I$ ")+
5604   AOT_have 0: < $\varphi \neq (\varphi)^{\sim}$ > for  $\varphi$ 
5605     using "thm-relation-negation:6" "\ $\forall I$ "

```

```

5606     "∀E"(1)[rotated, OF "log-prop-prop:2"] by fast
5607     AOT_thus <NonContingent0((p0)) & NonContingent0((p0)-) & (p0) ≠ (p0)->
5608     using "thm-noncont-propos:3" "thm-noncont-propos:4" "&I" by auto
5609 qed(auto simp: "log-prop-prop:2")
5610
5611 AOT_act_theorem "no-cnac": <¬∃x(E!x & ¬AE!x)> (207)
5612 proof(rule "raa-cor:2")
5613   AOT_assume <∃x(E!x & ¬AE!x)>
5614   then AOT_obtain a where a: <E!a & ¬AE!a>
5615   using "∃E"[rotated] by blast
5616   AOT_hence <A¬E!a>
5617   using "&E" "logic-actual-nec:1"[axiom_inst, THEN "≡E"(2)] by blast
5618   AOT_hence <¬E!a>
5619   using "logic-actual"[act_axiom_inst, THEN "→E"] by blast
5620   AOT_hence <E!a & ¬E!a>
5621   using a "&E" "&I" by blast
5622   AOT_thus <p & ¬p> for p using "raa-cor:1" by blast
5623 qed
5624
5625 AOT_theorem "pos-not-pna:1": <¬A∃x (E!x & ¬AE!x)> (208.1)
5626 proof(rule "raa-cor:2")
5627   AOT_assume <A∃x (E!x & ¬AE!x)>
5628   AOT_hence <∃x A(E!x & ¬AE!x)>
5629   using "Act-Basic:10"[THEN "≡E"(1)] by blast
5630   then AOT_obtain a where <A(E!a & ¬AE!a)>
5631   using "∃E"[rotated] by blast
5632   AOT_hence 1: <AE!a & A¬AE!a>
5633   using "Act-Basic:2"[THEN "≡E"(1)] by blast
5634   AOT_hence <¬AAE!a>
5635   using "&E"(2) "logic-actual-nec:1"[axiom_inst, THEN "≡E"(1)] by blast
5636   AOT_hence <¬AE!a>
5637   using "logic-actual-nec:4"[axiom_inst, THEN "≡E"(1)] RAA by blast
5638   AOT_thus <p & ¬p> for p using 1[THEN "&E"(1)] "&I" "raa-cor:1" by blast
5639 qed
5640
5641 AOT_theorem "pos-not-pna:2": <◇¬∃x(E!x & ¬AE!x)> (208.2)
5642 proof (rule RAA(1))
5643   AOT_show <¬A∃x (E!x & ¬AE!x)>
5644   using "pos-not-pna:1" by blast
5645 next
5646   AOT_assume <¬◇¬∃x (E!x & ¬AE!x)>
5647   AOT_hence <□∃x (E!x & ¬AE!x)>
5648   using "KBasic:12"[THEN "≡E"(2)] by blast
5649   AOT_thus <A∃x (E!x & ¬AE!x)>
5650   using "nec-imp-act"[THEN "→E"] by blast
5651 qed
5652
5653 AOT_theorem "pos-not-pna:3": <∃x (◇E!x & ¬AE!x)> (208.3)
5654 proof -
5655   AOT_obtain a where <◇(E!a & ¬AE!a)>
5656   using "qml:4"[axiom_inst] "BF◇"[THEN "→E"] "∃E"[rotated] by blast
5657   AOT_hence ϑ: <◇E!a> and ξ: <◇¬AE!a>
5658   using "KBasic2:3"[THEN "→E"] "&E" by blast+
5659   AOT_have <¬□AE!a>
5660   using ξ "KBasic:11"[THEN "≡E"(2)] by blast
5661   AOT_hence <¬AE!a>
5662   using "Act-Basic:6"[THEN "oth-class-taut:4:b"[THEN "≡E"(1)],
5663     THEN "≡E"(2)] by blast
5664   AOT_hence <◇E!a & ¬AE!a> using ϑ "&I" by blast
5665   thus ?thesis using "∃I" by fast
5666 qed
5667
5668 AOT_define contingent_prop :: φ ("q0")

```

```

5669   q0_def: <(q0) =df (∃x (E!x & ¬AE!x))>
5670
5671 AOT_theorem q0_prop: <◇q0 & ◇¬q0>
5672   apply (rule "=dfI"(2)[OF q0_def])
5673   apply (fact "log-prop-prop:2")
5674   apply (rule "&I")
5675   apply (fact "qml:4"[axiom_inst])
5676   by (fact "pos-not-pna:2")
5677
5678 AOT_theorem "basic-prop:1": <Contingent0((q0))> (209.1)
5679 proof(rule "contingent-properties:4[zero]"[THEN "≡dfI"])
5680   AOT_have <¬Necessary0((q0)) & ¬Impossible0((q0))>
5681   proof (rule "&I";
5682         rule "=dfI"(2)[OF q0_def];
5683         (rule "log-prop-prop:2" | rule "raa-cor:2"))
5684     AOT_assume <Necessary0(∃x (E!x & ¬AE!x))>
5685     AOT_hence <□∃x (E!x & ¬AE!x)>
5686       using "contingent-properties:1[zero]"[THEN "≡dfE"] by blast
5687     AOT_hence <A∃x (E!x & ¬AE!x)>
5688       using "Act-Basic:8"[THEN "→E"] "qml:2"[axiom_inst, THEN "→E"] by blast
5689     AOT_thus <A∃x (E!x & ¬AE!x) & ¬A∃x (E!x & ¬AE!x)>
5690       using "pos-not-pna:1" "&I" by blast
5691   next
5692     AOT_assume <Impossible0(∃x (E!x & ¬AE!x))>
5693     AOT_hence <□¬(∃x (E!x & ¬AE!x))>
5694       using "contingent-properties:2[zero]"[THEN "≡dfE"] by blast
5695     AOT_hence <¬◇(∃x (E!x & ¬AE!x))>
5696       using "KBasic2:1"[THEN "≡E"(1)] by blast
5697     AOT_thus <◇(∃x (E!x & ¬AE!x)) & ¬◇(∃x (E!x & ¬AE!x))>
5698       using "qml:4"[axiom_inst] "&I" by blast
5699   qed
5700   AOT_thus <¬(Necessary0((q0)) ∨ Impossible0((q0)))>
5701     using "oth-class-taut:5:d" "≡E"(2) by blast
5702 qed
5703
5704 AOT_theorem "basic-prop:2": <∃p Contingent0((p))> (209.2)
5705   using "∃I"(1)[rotated, OF "log-prop-prop:2"] "basic-prop:1" by blast
5706
5707 AOT_theorem "basic-prop:3": <Contingent0((q0)~)> (209.3)
5708   apply (AOT_subst <((q0)~)> <¬q0>)
5709   apply (insert "thm-relation-negation:3" "∀I"
5710         "∀E"(1)[rotated, OF "log-prop-prop:2"]; fast)
5711   apply (rule "contingent-properties:4[zero]"[THEN "≡dfI"])
5712   apply (rule "oth-class-taut:5:d"[THEN "≡E"(2)])
5713   apply (rule "&I")
5714   apply (rule "contingent-properties:1[zero]"[THEN "df-rules-formulas[3]",
5715         THEN "useful-tautologies:5"[THEN "→E"], THEN "→E"])
5716   apply (rule "conventions:5"[THEN "≡dfE"])
5717   apply (rule "=dfE"(2)[OF q0_def])
5718   apply (rule "log-prop-prop:2")
5719   apply (rule q0_prop[THEN "&E"(1)])
5720   apply (rule "contingent-properties:2[zero]"[THEN "df-rules-formulas[3]",
5721         THEN "useful-tautologies:5"[THEN "→E"], THEN "→E"])
5722   apply (rule "conventions:5"[THEN "≡dfE"])
5723   by (rule q0_prop[THEN "&E"(2)])
5724
5725 AOT_theorem "basic-prop:4": (209.4)
5726   <∃p∃q (p ≠ q & Contingent0(p) & Contingent0(q))>
5727 proof(rule "∃I")+
5728   AOT_have 0: <φ ≠ (φ)~> for φ
5729     using "thm-relation-negation:6" "∀I"
5730     "∀E"(1)[rotated, OF "log-prop-prop:2"] by fast
5731   AOT_show <(q0)~ ≠ (q0)~ & Contingent0(q0) & Contingent0((q0)~)>

```

```

5732     using "basic-prop:1" "basic-prop:3" "&I" 0 by presburger
5733 qed(auto simp: "log-prop-prop:2")
5734
5735 AOT_theorem "proposition-facts:1": (210.1)
5736   <NonContingent0(p) → ¬∃q (Contingent0(q) & q = p)>
5737 proof(rule "→I"; rule "raa-cor:2")
5738   AOT_assume <NonContingent0(p)>
5739   AOT_hence 1: <Necessary0(p) ∨ Impossible0(p)>
5740     using "contingent-properties:3[zero]"[THEN "≡dfE"] by blast
5741   AOT_assume <∃q (Contingent0(q) & q = p)>
5742   then AOT_obtain q where <Contingent0(q) & q = p>
5743     using "∃E"[rotated] by blast
5744   AOT_hence <Contingent0(p)>
5745     using "rule=E" "&E" by fast
5746   AOT_thus <(Necessary0(p) ∨ Impossible0(p)) &
5747     ¬(Necessary0(p) ∨ Impossible0(p))>
5748     using "contingent-properties:4[zero]"[THEN "≡dfE"] 1 "&I" by blast
5749 qed
5750
5751 AOT_theorem "proposition-facts:2": (210.2)
5752   <Contingent0(p) → ¬∃q (NonContingent0(q) & q = p)>
5753 proof(rule "→I"; rule "raa-cor:2")
5754   AOT_assume <Contingent0(p)>
5755   AOT_hence 1: <¬(Necessary0(p) ∨ Impossible0(p))>
5756     using "contingent-properties:4[zero]"[THEN "≡dfE"] by blast
5757   AOT_assume <∃q (NonContingent0(q) & q = p)>
5758   then AOT_obtain q where <NonContingent0(q) & q = p>
5759     using "∃E"[rotated] by blast
5760   AOT_hence <NonContingent0(p)>
5761     using "rule=E" "&E" by fast
5762   AOT_thus <(Necessary0(p) ∨ Impossible0(p)) &
5763     ¬(Necessary0(p) ∨ Impossible0(p))>
5764     using "contingent-properties:3[zero]"[THEN "≡dfE"] 1 "&I" by blast
5765 qed
5766
5767 AOT_theorem "proposition-facts:3": (210.3)
5768   <(p0) ≠ (p0)- & (p0) ≠ (q0) & (p0) ≠ (q0)- & (p0)- ≠ (q0)- & (q0) ≠ (q0)->
5769 proof -
5770   {
5771     fix χ φ ψ
5772     AOT_assume <χ{φ}>
5773     moreover AOT_assume <¬χ{ψ}>
5774     ultimately AOT_have <¬(χ{φ} ≡ χ{ψ})>
5775       using RAA "≡E" by metis
5776     moreover {
5777       AOT_have <∀p∀q ((¬(χ{p} ≡ χ{q})) → p ≠ q)>
5778         by (rule "∀I"; rule "∀I"; rule "pos-not-equiv-ne:4[zero]")
5779       AOT_hence <((¬(χ{φ} ≡ χ{ψ})) → φ ≠ ψ)>
5780         using "∀E" "log-prop-prop:2" by blast
5781     }
5782     ultimately AOT_have <φ ≠ ψ>
5783     using "→E" by blast
5784   } note 0 = this
5785   AOT_have contingent_neg: <Contingent0(φ) ≡ Contingent0(((φ)-)> for φ
5786     using "thm-cont-propos:3" "∀I"
5787     "∀E"(1)[rotated, OF "log-prop-prop:2"] by fast
5788   AOT_have not_noncontingent_if_contingent:
5789     <¬NonContingent0(φ)> if <Contingent0(φ)> for φ
5790     apply (rule "contingent-properties:3[zero]"[THEN "≡Df",
5791       THEN "oth-class-taut:4:b"[THEN "≡E"(1)], THEN "≡E"(2)])
5792     using that "contingent-properties:4[zero]"[THEN "≡dfE"] by blast
5793   show ?thesis
5794     apply (rule "&I")+

```

```

5795     using "thm-relation-negation:6" "∀I"
5796         "∀E"(1)[rotated, OF "log-prop-prop:2"]
5797         apply fast
5798         apply (rule 0)
5799     using "thm-noncont-propos:3" apply fast
5800         apply (rule not_noncontingent_if_contingent)
5801         apply (fact AOT)
5802         apply (rule 0)
5803     apply (rule "thm-noncont-propos:3")
5804         apply (rule not_noncontingent_if_contingent)
5805         apply (rule contingent_neg[THEN "≡E"(1)])
5806         apply (fact AOT)
5807         apply (rule 0)
5808     apply (rule "thm-noncont-propos:4")
5809         apply (rule not_noncontingent_if_contingent)
5810         apply (rule contingent_neg[THEN "≡E"(1)])
5811         apply (fact AOT)
5812     using "thm-relation-negation:6" "∀I"
5813         "∀E"(1)[rotated, OF "log-prop-prop:2"] by fast
5814 qed
5815
5816 AOT_define ContingentlyTrue :: <φ ⇒ φ> ("ContingentlyTrue'(_)'")
5817     "cont-tf:1": <ContingentlyTrue(p) ≡df p & ◇¬p> (211.1)
5818
5819 AOT_define ContingentlyFalse :: <φ ⇒ φ> ("ContingentlyFalse'(_)'")
5820     "cont-tf:2": <ContingentlyFalse(p) ≡df ¬p & ◇p> (211.2)
5821
5822 AOT_theorem "cont-true-cont:1": (212.1)
5823     <ContingentlyTrue((p)) → Contingent0((p))>
5824 proof(rule "→I")
5825     AOT_assume <ContingentlyTrue((p))>
5826     AOT_hence 1: <p> and 2: <◇¬p> using "cont-tf:1"[THEN "≡dfE"] "&E" by blast+
5827     AOT_have <¬Necessary0((p))>
5828         apply (rule "contingent-properties:1[zero]"[THEN "≡df",
5829             THEN "oth-class-taut:4:b"[THEN "≡E"(1)], THEN "≡E"(2)])
5830         using 2 "KBasic:11"[THEN "≡E"(2)] by blast
5831     moreover AOT_have <¬Impossible0((p))>
5832         apply (rule "contingent-properties:2[zero]"[THEN "≡df",
5833             THEN "oth-class-taut:4:b"[THEN "≡E"(1)], THEN "≡E"(2)])
5834         apply (rule "conventions:5"[THEN "≡dfE"])
5835         using "T◇"[THEN "→E", OF 1].
5836     ultimately AOT_have <¬(Necessary0((p)) ∨ Impossible0((p)))>
5837         using DeMorgan(2)[THEN "≡E"(2)] "&I" by blast
5838     AOT_thus <Contingent0((p))>
5839         using "contingent-properties:4[zero]"[THEN "≡dfI"] by blast
5840 qed
5841
5842 AOT_theorem "cont-true-cont:2": (212.2)
5843     <ContingentlyFalse((p)) → Contingent0((p))>
5844 proof(rule "→I")
5845     AOT_assume <ContingentlyFalse((p))>
5846     AOT_hence 1: <¬p> and 2: <◇p> using "cont-tf:2"[THEN "≡dfE"] "&E" by blast+
5847     AOT_have <¬Necessary0((p))>
5848         apply (rule "contingent-properties:1[zero]"[THEN "≡df",
5849             THEN "oth-class-taut:4:b"[THEN "≡E"(1)], THEN "≡E"(2)])
5850         using "KBasic:11"[THEN "≡E"(2)] "T◇"[THEN "→E", OF 1] by blast
5851     moreover AOT_have <¬Impossible0((p))>
5852         apply (rule "contingent-properties:2[zero]"[THEN "≡df",
5853             THEN "oth-class-taut:4:b"[THEN "≡E"(1)], THEN "≡E"(2)])
5854         apply (rule "conventions:5"[THEN "≡dfE"])
5855         using 2.
5856     ultimately AOT_have <¬(Necessary0((p)) ∨ Impossible0((p)))>
5857         using DeMorgan(2)[THEN "≡E"(2)] "&I" by blast

```



```

5858   AOT_thus <Contingent0((p))>
5859     using "contingent-properties:4[zero]"[THEN "≡dfI"] by blast
5860 qed
5861
5862 AOT_theorem "cont-true-cont:3":
5863   <ContingentlyTrue((p)) ≡ ContingentlyFalse(((p)~)>
5864 proof(rule "≡I"; rule "→I")
5865   AOT_assume <ContingentlyTrue((p))>
5866   AOT_hence 0: <p & ◇¬p> using "cont-tf:1"[THEN "≡dfE"] by blast
5867   AOT_have 1: <ContingentlyFalse(¬p)>
5868     apply (rule "cont-tf:2"[THEN "≡dfI"])
5869     apply (AOT_subst (reverse) <¬¬p> p)
5870     by (auto simp: "oth-class-taut:3:b" 0)
5871   AOT_show <ContingentlyFalse(((p)~)>
5872     apply (AOT_subst <((p)~)> <¬p>)
5873     by (auto simp: "thm-relation-negation:3" 1)
5874 next
5875   AOT_assume 1: <ContingentlyFalse(((p)~)>
5876   AOT_have <ContingentlyFalse(¬p)>
5877     by (AOT_subst (reverse) <¬p> <((p)~)>)
5878     (auto simp: "thm-relation-negation:3" 1)
5879   AOT_hence <¬¬p & ◇¬p> using "cont-tf:2"[THEN "≡dfE"] by blast
5880   AOT_hence <p & ◇¬p>
5881     using "&I" "&E" "useful-tautologies:1"[THEN "→E"] by metis
5882   AOT_thus <ContingentlyTrue((p))>
5883     using "cont-tf:1"[THEN "≡dfI"] by blast
5884 qed
5885
5886 AOT_theorem "cont-true-cont:4":
5887   <ContingentlyFalse((p)) ≡ ContingentlyTrue(((p)~)>
5888 proof(rule "≡I"; rule "→I")
5889   AOT_assume <ContingentlyFalse(p)>
5890   AOT_hence 0: <¬p & ◇p>
5891     using "cont-tf:2"[THEN "≡dfE"] by blast
5892   AOT_have <¬p & ◇¬¬p>
5893     by (AOT_subst (reverse) <¬¬p> p)
5894     (auto simp: "oth-class-taut:3:b" 0)
5895   AOT_hence 1: <ContingentlyTrue(¬p)>
5896     by (rule "cont-tf:1"[THEN "≡dfI"])
5897   AOT_show <ContingentlyTrue(((p)~)>
5898     by (AOT_subst <((p)~)> <¬p>)
5899     (auto simp: "thm-relation-negation:3" 1)
5900 next
5901   AOT_assume 1: <ContingentlyTrue(((p)~)>
5902   AOT_have <ContingentlyTrue(¬p)>
5903     by (AOT_subst (reverse) <¬p> <((p)~)>)
5904     (auto simp add: "thm-relation-negation:3" 1)
5905   AOT_hence 2: <¬p & ◇¬¬p> using "cont-tf:1"[THEN "≡dfE"] by blast
5906   AOT_have <◇p>
5907     by (AOT_subst p <¬¬p>)
5908     (auto simp add: "oth-class-taut:3:b" 2[THEN "&E"(2)])
5909   AOT_hence <¬p & ◇p> using 2[THEN "&E"(1)] "&I" by blast
5910   AOT_thus <ContingentlyFalse(p)>
5911     by (rule "cont-tf:2"[THEN "≡dfI"])
5912 qed
5913
5914 AOT_theorem "cont-true-cont:5":
5915   <(ContingentlyTrue((p)) & Necessary0((q))) → p ≠ q>
5916 proof (rule "→I"; frule "&E"(1); drule "&E"(2); rule "raa-cor:1")
5917   AOT_assume <ContingentlyTrue((p))>
5918   AOT_hence <◇¬p>
5919     using "cont-tf:1"[THEN "≡dfE"] "&E" by blast
5920   AOT_hence 0: <¬□p> using "KBasic:11"[THEN "≡E"(2)] by blast

```



```

5921 AOT_assume <Necessary0((q))>
5922 moreover AOT_assume <¬(p ≠ q)>
5923 AOT_hence <p = q>
5924   using "--infix"[THEN "≡Df",
5925           THEN "oth-class-taut:4:b"[THEN "≡E"(1)],
5926           THEN "≡E"(1)]
5927   "useful-tautologies:1"[THEN "→E"] by blast
5928 ultimately AOT_have <Necessary0((p))> using "rule=E" id_sym by blast
5929 AOT_hence <□p>
5930   using "contingent-properties:1[zero]"[THEN "≡dfE"] by blast
5931 AOT_thus <□p & ¬□p> using 0 "&I" by blast
5932 qed
5933
5934 AOT_theorem "cont-true-cont:6": (212.6)
5935   <(ContingentlyFalse((p)) & Impossible0((q))) → p ≠ q>
5936 proof (rule "→I"; frule "&E"(1); drule "&E"(2); rule "raa-cor:1")
5937   AOT_assume <ContingentlyFalse((p))>
5938   AOT_hence <◇p>
5939     using "cont-tf:2"[THEN "≡dfE"] "&E" by blast
5940   AOT_hence 1: <¬□¬p>
5941     using "conventions:5"[THEN "≡dfE"] by blast
5942   AOT_assume <Impossible0((q))>
5943   moreover AOT_assume <¬(p ≠ q)>
5944   AOT_hence <p = q>
5945     using "--infix"[THEN "≡Df",
5946             THEN "oth-class-taut:4:b"[THEN "≡E"(1)],
5947             THEN "≡E"(1)]
5948     "useful-tautologies:1"[THEN "→E"] by blast
5949   ultimately AOT_have <Impossible0((p))> using "rule=E" id_sym by blast
5950   AOT_hence <□¬p>
5951     using "contingent-properties:2[zero]"[THEN "≡dfE"] by blast
5952   AOT_thus <□¬p & ¬□¬p> using 1 "&I" by blast
5953 qed
5954
5955 AOT_act_theorem "q0cf:1": <ContingentlyFalse(q0)> (213.1)
5956   apply (rule "cont-tf:2"[THEN "≡dfI"])
5957   apply (rule "=dfI"(2)[OF q0_def])
5958   apply (fact "log-prop-prop:2")
5959   apply (rule "&I")
5960   apply (fact "no-cnac")
5961   by (fact "qml:4"[axiom_inst])
5962
5963 AOT_act_theorem "q0cf:2": <ContingentlyTrue(((q0)¬)> (213.2)
5964   apply (rule "cont-tf:1"[THEN "≡dfI"])
5965   apply (rule "=dfI"(2)[OF q0_def])
5966   apply (fact "log-prop-prop:2")
5967   apply (rule "&I")
5968   apply (rule "thm-relation-negation:3"
5969           [unvary p, OF "log-prop-prop:2", THEN "≡E"(2)])
5970   apply (fact "no-cnac")
5971   apply (rule "rule=E"[rotated,
5972           OF "thm-relation-negation:7"
5973           [unvary p, OF "log-prop-prop:2", THEN id_sym]])
5974   apply (AOT_subst (reverse) <¬¬(∃x (E!x & ¬AE!x))> <∃x (E!x & ¬AE!x)>)
5975   by (auto simp: "oth-class-taut:3:b" "qml:4"[axiom_inst])
5976
5977 AOT_theorem "cont-tf-thm:1": <∃p ContingentlyTrue((p))> (215.1)
5978 proof (rule "∀E"(1)[OF "exc-mid"]; rule "→I"; rule "∃I")
5979   AOT_assume <q0>
5980   AOT_hence <q0 & ◇¬q0> using q0_prop[THEN "&E"(2)] "&I" by blast
5981   AOT_thus <ContingentlyTrue(q0)>
5982     by (rule "cont-tf:1"[THEN "≡dfI"])
5983 next

```

```

5984   AOT_assume <¬q₀>
5985   AOT_hence <¬q₀ & ◇q₀> using q₀_prop[THEN "&E"(1)] "&I" by blast
5986   AOT_hence <ContingentlyFalse(q₀)>
5987     by (rule "cont-tf:2"[THEN "≡dfI"])
5988   AOT_thus <ContingentlyTrue((q₀)⁻)>
5989     by (rule "cont-true-cont:4"[unvarify p,
5990       OF "log-prop-prop:2", THEN "≡E"(1)])
5991 qed(auto simp: "log-prop-prop:2")
5992
5993
5994 AOT_theorem "cont-tf-thm:2": <∃p ContingentlyFalse((p))> (215.2)
5995 proof(rule "VE"(1)[OF "exc-mid"]; rule "→I"; rule "∃I")
5996   AOT_assume <q₀>
5997   AOT_hence <q₀ & ◇¬q₀> using q₀_prop[THEN "&E"(2)] "&I" by blast
5998   AOT_hence <ContingentlyTrue(q₀)>
5999     by (rule "cont-tf:1"[THEN "≡dfI"])
6000   AOT_thus <ContingentlyFalse((q₀)⁻)>
6001     by (rule "cont-true-cont:3"[unvarify p,
6002       OF "log-prop-prop:2", THEN "≡E"(1)])
6003 next
6004   AOT_assume <¬q₀>
6005   AOT_hence <¬q₀ & ◇q₀> using q₀_prop[THEN "&E"(1)] "&I" by blast
6006   AOT_thus <ContingentlyFalse(q₀)>
6007     by (rule "cont-tf:2"[THEN "≡dfI"])
6008 qed(auto simp: "log-prop-prop:2")
6009
6010 AOT_theorem "property-facts1:1": <∃F∃x ([F]x & ◇¬[F]x)> (217.1)
6011 proof -
6012   fix x
6013   AOT_obtain p₁ where <ContingentlyTrue((p₁))>
6014     using "cont-tf-thm:1" "∃E"[rotated] by blast
6015   AOT_hence 1: <p₁ & ◇¬p₁> using "cont-tf:1"[THEN "≡dfE"] by blast
6016   AOT_modally_strict {
6017     AOT_have <for arbitrary p: ⊢□ ([λz p]x ≡ p)>
6018       by (rule "beta-C-cor:3"[THEN "VE"(2)]) cqt_2_lambda_inst_prover
6019     AOT_hence <for arbitrary p: ⊢□ □ ([λz p]x ≡ p)>
6020       by (rule RN)
6021     AOT_hence <∀p □([λz p]x ≡ p)> using GEN by fast
6022     AOT_hence <□([λz p₁]x ≡ p₁)> using "VE" by fast
6023   } note 2 = this
6024   AOT_hence <□([λz p₁]x ≡ p₁)> using "VE" by blast
6025   AOT_hence <[λz p₁]x>
6026     using 1[THEN "&E"(1)] "qml:2"[axiom_inst, THEN "→E"] "≡E"(2) by blast
6027   moreover AOT_have <◇¬[λz p₁]x>
6028     using 2[THEN "qml:2"[axiom_inst, THEN "→E"]]
6029     apply (AOT_subst <[λz p₁]x> <p₁>)
6030     using 1[THEN "&E"(2)] by blast
6031   ultimately AOT_have <[λz p₁]x & ◇¬[λz p₁]x> using "&I" by blast
6032   AOT_hence <∃x ([λz p₁]x & ◇¬[λz p₁]x)> using "∃I"(2) by fast
6033   moreover AOT_have <[λz p₁]↓> by "cqt:2[lambda]"
6034   ultimately AOT_show <∃F∃x ([F]x & ◇¬[F]x)> by (rule "∃I"(1))
6035 qed
6036
6037 AOT_theorem "property-facts1:2": <∃F∃x (¬[F]x & ◇[F]x)> (217.2)
6038 proof -
6039   fix x
6040   AOT_obtain p₁ where <ContingentlyFalse((p₁))>
6041     using "cont-tf-thm:2" "∃E"[rotated] by blast
6042   AOT_hence 1: <¬p₁ & ◇p₁> using "cont-tf:2"[THEN "≡dfE"] by blast
6043   AOT_modally_strict {
6044     AOT_have <for arbitrary p: ⊢□ ([λz p]x ≡ p)>
6045       by (rule "beta-C-cor:3"[THEN "VE"(2)]) cqt_2_lambda_inst_prover
6046     AOT_hence <for arbitrary p: ⊢□ (¬[λz p]x ≡ ¬p)>

```

```

6047     using "oth-class-taut:4:b" "≡E" by blast
6048     AOT_hence <for arbitrary p:  $\vdash_{\square} \square(\neg[\lambda z p]x \equiv \neg p)$ >
6049     by (rule RN)
6050     AOT_hence < $\forall p \square(\neg[\lambda z p]x \equiv \neg p)$ > using GEN by fast
6051     AOT_hence < $\square(\neg[\lambda z p_1]x \equiv \neg p_1)$ > using "VE" by fast
6052 } note 2 = this
6053 AOT_hence < $\square(\neg[\lambda z p_1]x \equiv \neg p_1)$ > using "VE" by blast
6054 AOT_hence 3: < $\neg[\lambda z p_1]x$ >
6055     using 1[THEN "&E"(1)] "qml:2"[axiom_inst, THEN " $\rightarrow$ E"] "≡E"(2) by blast
6056 AOT_modally_strict {
6057     AOT_have <for arbitrary p:  $\vdash_{\square} ([\lambda z p]x \equiv p)$ >
6058     by (rule "beta-C-cor:3"[THEN "VE"(2)]) cqt_2_lambda_inst_prover
6059     AOT_hence <for arbitrary p:  $\vdash_{\square} \square([\lambda z p]x \equiv p)$ >
6060     by (rule RN)
6061     AOT_hence < $\forall p \square([\lambda z p]x \equiv p)$ > using GEN by fast
6062     AOT_hence < $\square([\lambda z p_1]x \equiv p_1)$ > using "VE" by fast
6063 } note 4 = this
6064 AOT_have < $\diamond[\lambda z p_1]x$ >
6065     using 4[THEN "qml:2"[axiom_inst, THEN " $\rightarrow$ E"]]
6066     apply (AOT_subst < $[\lambda z p_1]x$ > < $p_1$ >)
6067     using 1[THEN "&E"(2)] by blast
6068 AOT_hence < $\neg[\lambda z p_1]x$  &  $\diamond[\lambda z p_1]x$ > using 3 "&I" by blast
6069 AOT_hence < $\exists x (\neg[\lambda z p_1]x$  &  $\diamond[\lambda z p_1]x)$ > using "EI"(2) by fast
6070 moreover AOT_have < $[\lambda z p_1]x \downarrow$ > by "cqt:2[lambda]"
6071 ultimately AOT_show < $\exists F \exists x (\neg[F]x$  &  $\diamond[F]x)$ > by (rule "EI"(1))
6072 qed
6073
6074 context
6075 begin
6076
6077 private AOT_lemma eqnotnec_123_Aux_ζ: < $[L]x \equiv (E!x \rightarrow E!x)$ >
6078     apply (rule "=afI"(2)[OF L_def])
6079     apply "cqt:2[lambda]"
6080     apply (rule "beta-C-meta"[THEN " $\rightarrow$ E"])
6081     by "cqt:2[lambda]"
6082
6083 private AOT_lemma eqnotnec_123_Aux_ω: < $[\lambda z \varphi]x \equiv \varphi$ >
6084     by (rule "beta-C-meta"[THEN " $\rightarrow$ E"]) "cqt:2[lambda]"
6085
6086 private AOT_lemma eqnotnec_123_Aux_θ: < $\varphi \equiv \forall x([L]x \equiv [\lambda z \varphi]x)$ >
6087 proof(rule "≡I"; rule " $\rightarrow$ I"; (rule "VI"?)
6088     fix x
6089     AOT_assume 1: < $\varphi$ >
6090     AOT_have < $[L]x \equiv (E!x \rightarrow E!x)$ > using eqnotnec_123_Aux_ζ.
6091     also AOT_have < $\dots \equiv \varphi$ >
6092     using "if-p-then-p" 1 "≡I" " $\rightarrow$ I" by simp
6093     also AOT_have < $\dots \equiv [\lambda z \varphi]x$ >
6094     using "Commutativity of ≡"[THEN "≡E"(1)] eqnotnec_123_Aux_ω by blast
6095     finally AOT_show < $[L]x \equiv [\lambda z \varphi]x$ >.
6096 next
6097     fix x
6098     AOT_assume < $\forall x([L]x \equiv [\lambda z \varphi]x)$ >
6099     AOT_hence < $[L]x \equiv [\lambda z \varphi]x$ > using "VE" by blast
6100     also AOT_have < $\dots \equiv \varphi$ > using eqnotnec_123_Aux_ω.
6101     finally AOT_have < $\varphi \equiv [L]x$ >
6102     using "Commutativity of ≡"[THEN "≡E"(1)] by blast
6103     also AOT_have < $\dots \equiv E!x \rightarrow E!x$ > using eqnotnec_123_Aux_ζ.
6104     finally AOT_show < $\varphi$ > using "≡E" "if-p-then-p" by fast
6105 qed
6106 private lemmas eqnotnec_123_Aux_ξ =
6107     eqnotnec_123_Aux_θ[THEN "oth-class-taut:4:b"[THEN "≡E"(1)],
6108     THEN "conventions:3"[THEN "≡Df", THEN "≡E"(1), THEN "&E"(1)],
6109     THEN "RM◇"]

```

```

6110 private lemmas eqnotnec_123_Aux_ξ' =
6111   eqnotnec_123_Aux_ϑ[
6112     THEN "conventions:3"[THEN "≡Df", THEN "≡E"(1), THEN "&E"(1)],
6113     THEN "RM◇"]
6114
6115 AOT_theorem "eqnotnec:1": <∃F∃G(∀x([F]x ≡ [G]x) & ◇¬∀x([F]x ≡ [G]x))> (219.1)
6116 proof-
6117   AOT_obtain p1 where <ContingentlyTrue(p1)>
6118     using "cont-tf-thm:1" "∃E"[rotated] by blast
6119   AOT_hence <p1 & ◇¬p1> using "cont-tf:1"[THEN "≡dfE"] by blast
6120   AOT_hence <∀x ([L]x ≡ [λz p1]x) & ◇¬∀x([L]x ≡ [λz p1]x)>
6121     apply - apply (rule "&I")
6122     using "&E" eqnotnec_123_Aux_ϑ[THEN "≡E"(1)]
6123     eqnotnec_123_Aux_ξ "→E" by fast+
6124   AOT_hence <∃G (∀x([L]x ≡ [G]x) & ◇¬∀x([L]x ≡ [G]x))>
6125     by (rule "∃I") "cqt:2[lambda]"
6126   AOT_thus <∃F∃G (∀x([F]x ≡ [G]x) & ◇¬∀x([F]x ≡ [G]x))>
6127     apply (rule "∃I")
6128     by (rule "=dfI"(2)[OF L_def]) "cqt:2[lambda]" +
6129 qed
6130
6131 AOT_theorem "eqnotnec:2": <∃F∃G(¬∀x([F]x ≡ [G]x) & ◇∀x([F]x ≡ [G]x))> (219.2)
6132 proof-
6133   AOT_obtain p1 where <ContingentlyFalse(p1)>
6134     using "cont-tf-thm:2" "∃E"[rotated] by blast
6135   AOT_hence <¬p1 & ◇p1> using "cont-tf:2"[THEN "≡dfE"] by blast
6136   AOT_hence <¬∀x ([L]x ≡ [λz p1]x) & ◇∀x([L]x ≡ [λz p1]x)>
6137     apply - apply (rule "&I")
6138     using eqnotnec_123_Aux_ϑ[THEN "oth-class-taut:4:b"[THEN "≡E"(1)],
6139       THEN "≡E"(1)]
6140     "&E" eqnotnec_123_Aux_ξ' "→E" by fast+
6141   AOT_hence <∃G (¬∀x([L]x ≡ [G]x) & ◇∀x([L]x ≡ [G]x))>
6142     by (rule "∃I") "cqt:2[lambda]"
6143   AOT_thus <∃F∃G (¬∀x([F]x ≡ [G]x) & ◇∀x([F]x ≡ [G]x))>
6144     apply (rule "∃I")
6145     by (rule "=dfI"(2)[OF L_def]) "cqt:2[lambda]" +
6146 qed
6147
6148 AOT_theorem "eqnotnec:3": <∃F∃G(⊆¬∀x([F]x ≡ [G]x) & ◇∀x([F]x ≡ [G]x))> (219.3)
6149 proof-
6150   AOT_have <¬⊆q0>
6151     apply (rule "=dfI"(2)[OF q0_def])
6152     apply (fact "log-prop-prop:2")
6153     by (fact AOT)
6154   AOT_hence <⊆¬q0>
6155     using "logic-actual-nec:1"[axiom_inst, THEN "≡E"(2)] by blast
6156   AOT_hence <⊆¬∀x ([L]x ≡ [λz q0]x)>
6157     using eqnotnec_123_Aux_ϑ[THEN "oth-class-taut:4:b"[THEN "≡E"(1)],
6158       THEN "conventions:3"[THEN "≡Df", THEN "≡E"(1), THEN "&E"(1)],
6159       THEN "RA[2]", THEN "act-cond"[THEN "→E"], THEN "→E"] by blast
6160   moreover AOT_have <◇∀x ([L]x ≡ [λz q0]x)>
6161     using eqnotnec_123_Aux_ξ'[THEN "→E"] q0_prop[THEN "&E"(1)] by blast
6162   ultimately AOT_have <⊆¬∀x ([L]x ≡ [λz q0]x) & ◇∀x ([L]x ≡ [λz q0]x)>
6163     using "&I" by blast
6164   AOT_hence <∃G (⊆¬∀x([L]x ≡ [G]x) & ◇∀x([L]x ≡ [G]x))>
6165     by (rule "∃I") "cqt:2[lambda]"
6166   AOT_thus <∃F∃G (⊆¬∀x([F]x ≡ [G]x) & ◇∀x([F]x ≡ [G]x))>
6167     apply (rule "∃I")
6168     by (rule "=dfI"(2)[OF L_def]) "cqt:2[lambda]" +
6169 qed
6170
6171 end
6172

```

```

6173 AOT_theorem "eqnotnec:4": < $\forall F \exists G (\forall x ([F]x \equiv [G]x) \ \& \ \Diamond \neg \forall x ([F]x \equiv [G]x))$ > (219.4)
6174 proof(rule GEN)
6175   fix F
6176   AOT_have Aux_A: < $\vdash_{\Box} \psi \rightarrow \forall x ([F]x \equiv [\lambda z [F]z \ \& \ \psi]x)$ > for  $\psi$ 
6177   proof(rule "→I"; rule GEN)
6178     AOT_modally_strict {
6179       fix x
6180       AOT_assume 0: < $\psi$ >
6181       AOT_have < $[\lambda z [F]z \ \& \ \psi]x \equiv [F]x \ \& \ \psi$ >
6182         by (rule "beta-C-meta"[THEN "→E"]) "cqt:2[lambda]"
6183       also AOT_have < $\dots \equiv [F]x$ >
6184         apply (rule "≡I"; rule "→I")
6185         using "∨E"(3)[rotated, OF "useful-tautologies:2"[THEN "→E"], OF 0] "&E"
6186         apply blast
6187         using 0 "&I" by blast
6188       finally AOT_show < $[F]x \equiv [\lambda z [F]z \ \& \ \psi]x$ >
6189         using "Commutativity of ≡"[THEN "≡E"(1)] by blast
6190     }
6191   qed
6192
6193   AOT_have Aux_B: < $\vdash_{\Box} \psi \rightarrow \forall x ([F]x \equiv [\lambda z [F]z \ \& \ \psi \ \vee \ \neg \psi]x)$ > for  $\psi$ 
6194   proof (rule "→I"; rule GEN)
6195     AOT_modally_strict {
6196       fix x
6197       AOT_assume 0: < $\psi$ >
6198       AOT_have < $[\lambda z ([F]z \ \& \ \psi) \ \vee \ \neg \psi]x \equiv (([F]x \ \& \ \psi) \ \vee \ \neg \psi)$ >
6199         by (rule "beta-C-meta"[THEN "→E"]) "cqt:2[lambda]"
6200       also AOT_have < $\dots \equiv [F]x$ >
6201         apply (rule "≡I"; rule "→I")
6202         using "∨E"(3)[rotated, OF "useful-tautologies:2"[THEN "→E"], OF 0]
6203           "&E"
6204         apply blast
6205         apply (rule "∨I"(1)) using 0 "&I" by blast
6206       finally AOT_show < $[F]x \equiv [\lambda z ([F]z \ \& \ \psi) \ \vee \ \neg \psi]x$ >
6207         using "Commutativity of ≡"[THEN "≡E"(1)] by blast
6208     }
6209   qed
6210
6211   AOT_have Aux_C:
6212     < $\vdash_{\Box} \Diamond \neg \psi \rightarrow \Diamond \neg \forall z ([\lambda z [F]z \ \& \ \psi]z \equiv [\lambda z [F]z \ \& \ \psi \ \vee \ \neg \psi]z)$ > for  $\psi$ 
6213   proof(rule "RM◇"; rule "→I"; rule "raa-cor:2")
6214     AOT_modally_strict {
6215       AOT_assume 0: < $\neg \psi$ >
6216       AOT_assume < $\forall z ([\lambda z [F]z \ \& \ \psi]z \equiv [\lambda z [F]z \ \& \ \psi \ \vee \ \neg \psi]z)$ >
6217       AOT_hence < $[\lambda z [F]z \ \& \ \psi]z \equiv [\lambda z [F]z \ \& \ \psi \ \vee \ \neg \psi]z$ > for z
6218         using "∨E" by blast
6219       moreover AOT_have < $[\lambda z [F]z \ \& \ \psi]z \equiv [F]z \ \& \ \psi$ > for z
6220         by (rule "beta-C-meta"[THEN "→E"]) "cqt:2[lambda]"
6221       moreover AOT_have < $[\lambda z ([F]z \ \& \ \psi) \ \vee \ \neg \psi]z \equiv (([F]z \ \& \ \psi) \ \vee \ \neg \psi)$ > for z
6222         by (rule "beta-C-meta"[THEN "→E"]) "cqt:2[lambda]"
6223       ultimately AOT_have < $[F]z \ \& \ \psi \equiv (([F]z \ \& \ \psi) \ \vee \ \neg \psi)$ > for z
6224         using "Commutativity of ≡"[THEN "≡E"(1)] "≡E"(5) by meson
6225       moreover AOT_have < $(([F]z \ \& \ \psi) \ \vee \ \neg \psi)$ > for z using 0 "∨I" by blast
6226       ultimately AOT_have < $\psi$ > using "≡E" "&E" by metis
6227       AOT_thus < $\psi \ \& \ \neg \psi$ > using 0 "&I" by blast
6228     }
6229   qed
6230
6231   AOT_have Aux_D: < $\Box \forall z ([F]z \equiv [\lambda z [F]z \ \& \ \psi]z) \rightarrow$ 
6232     < $(\Diamond \neg \forall x ([\lambda z [F]z \ \& \ \psi]x \equiv [\lambda z [F]z \ \& \ \psi \ \vee \ \neg \psi]x) \equiv$ 
6233     < $\Diamond \neg \forall x ([F]x \equiv [\lambda z [F]z \ \& \ \psi \ \vee \ \neg \psi]x))$ > for  $\psi$ 
6234   proof (rule "→I")
6235     AOT_assume A: < $\Box \forall z ([F]z \equiv [\lambda z [F]z \ \& \ \psi]z)$ >

```

```

6236 AOT_show <math>\Diamond \neg \forall x ([\lambda z [F]z \ \& \ \psi]x \equiv [\lambda z [F]z \ \& \ \psi \ \vee \ \neg \psi]x) \equiv</math>
6237 <math>\Diamond \neg \forall x ([F]x \equiv [\lambda z [F]z \ \& \ \psi \ \vee \ \neg \psi]x)>
6238 proof(rule "≡I"; rule "KBasic:13"[THEN "→E"];
6239 rule "RN[prem]"[where Γ="{« $\forall z ([F]z \equiv [\lambda z [F]z \ \& \ \psi]z)$ »}", simplified];
6240 (rule "useful-tautologies:5"[THEN "→E"]; rule "→I")?)
6241 AOT_modally_strict {
6242 AOT_assume <math>\forall z ([F]z \equiv [\lambda z [F]z \ \& \ \psi]z)>
6243 AOT_hence 1: <math>\langle [F]z \equiv [\lambda z [F]z \ \& \ \psi]z \rangle</math> for z
6244 using "∀E" by blast
6245 AOT_assume <math>\forall x ([F]x \equiv [\lambda z [F]z \ \& \ \psi \ \vee \ \neg \psi]x)>
6246 AOT_hence 2: <math>\langle [F]z \equiv [\lambda z [F]z \ \& \ \psi \ \vee \ \neg \psi]z \rangle</math> for z
6247 using "∀E" by blast
6248 AOT_have <math>\langle [\lambda z [F]z \ \& \ \psi]z \equiv [\lambda z [F]z \ \& \ \psi \ \vee \ \neg \psi]z \rangle</math> for z
6249 using "≡E" 1 2 by meson
6250 AOT_thus <math>\langle \forall x ([\lambda z [F]z \ \& \ \psi]x \equiv [\lambda z [F]z \ \& \ \psi \ \vee \ \neg \psi]x) \rangle</math>
6251 by (rule GEN)
6252 }
6253 next
6254 AOT_modally_strict {
6255 AOT_assume <math>\forall z ([F]z \equiv [\lambda z [F]z \ \& \ \psi]z)>
6256 AOT_hence 1: <math>\langle [F]z \equiv [\lambda z [F]z \ \& \ \psi]z \rangle</math> for z
6257 using "∀E" by blast
6258 AOT_assume <math>\langle \forall x ([\lambda z [F]z \ \& \ \psi]x \equiv [\lambda z [F]z \ \& \ \psi \ \vee \ \neg \psi]x) \rangle</math>
6259 AOT_hence 2: <math>\langle [\lambda z [F]z \ \& \ \psi]z \equiv [\lambda z [F]z \ \& \ \psi \ \vee \ \neg \psi]z \rangle</math> for z
6260 using "∀E" by blast
6261 AOT_have <math>\langle [F]z \equiv [\lambda z [F]z \ \& \ \psi \ \vee \ \neg \psi]z \rangle</math> for z
6262 using 1 2 "≡E" by meson
6263 AOT_thus <math>\langle \forall x ([F]x \equiv [\lambda z [F]z \ \& \ \psi \ \vee \ \neg \psi]x) \rangle</math>
6264 by (rule GEN)
6265 }
6266 qed(auto simp: A)
6267 qed
6268
6269 AOT_obtain p1 where p1_prop: <math>\langle p_1 \ \& \ \Diamond \neg p_1 \rangle</math>
6270 using "cont-tf-thm:1" "∃E"[rotated]
6271 "cont-tf:1"[THEN "≡dfE"] by blast
6272 {
6273 AOT_assume 1: <math>\langle \Box \forall x ([F]x \equiv [\lambda z [F]z \ \& \ p_1]x) \rangle</math>
6274 AOT_have 2: <math>\langle \forall x ([F]x \equiv [\lambda z [F]z \ \& \ p_1 \ \vee \ \neg p_1]x) \rangle</math>
6275 using Aux_B[THEN "→E", OF p1_prop[THEN "&E"(1)]] .
6276 AOT_have <math>\langle \Diamond \neg \forall x ([\lambda z [F]z \ \& \ p_1]x \equiv [\lambda z [F]z \ \& \ p_1 \ \vee \ \neg p_1]x) \rangle</math>
6277 using Aux_C[THEN "→E", OF p1_prop[THEN "&E"(2)]] .
6278 AOT_hence 3: <math>\langle \Diamond \neg \forall x ([F]x \equiv [\lambda z [F]z \ \& \ p_1 \ \vee \ \neg p_1]x) \rangle</math>
6279 using Aux_D[THEN "→E", OF 1, THEN "≡E"(1)] by blast
6280 AOT_hence <math>\langle \forall x ([F]x \equiv [\lambda z [F]z \ \& \ p_1 \ \vee \ \neg p_1]x) \ \& \ \Diamond \neg \forall x ([F]x \equiv [\lambda z [F]z \ \& \ p_1 \ \vee \ \neg p_1]x) \rangle</math>
6281 using 2 "&I" by blast
6282 AOT_hence <math>\langle \exists G (\forall x ([F]x \equiv [G]x) \ \& \ \Diamond \neg \forall x ([F]x \equiv [G]x)) \rangle</math>
6283 by (rule "∃I"(1)) "cqt:2[lambda]"
6284 }
6285 }
6286 moreover {
6287 AOT_assume 2: <math>\langle \neg \Box \forall x ([F]x \equiv [\lambda z [F]z \ \& \ p_1]x) \rangle</math>
6288 AOT_hence <math>\langle \Diamond \neg \forall x ([F]x \equiv [\lambda z [F]z \ \& \ p_1]x) \rangle</math>
6289 using "KBasic:11"[THEN "≡E"(1)] by blast
6290 AOT_hence <math>\langle \forall x ([F]x \equiv [\lambda z [F]z \ \& \ p_1]x) \ \& \ \Diamond \neg \forall x ([F]x \equiv [\lambda z [F]z \ \& \ p_1]x) \rangle</math>
6291 using Aux_A[THEN "→E", OF p1_prop[THEN "&E"(1)]] "&I" by blast
6292 AOT_hence <math>\langle \exists G (\forall x ([F]x \equiv [G]x) \ \& \ \Diamond \neg \forall x ([F]x \equiv [G]x)) \rangle</math>
6293 by (rule "∃I"(1)) "cqt:2[lambda]"
6294 }
6295 ultimately AOT_show <math>\langle \exists G (\forall x ([F]x \equiv [G]x) \ \& \ \Diamond \neg \forall x ([F]x \equiv [G]x)) \rangle</math>
6296 using "∀E"(1)[OF "exc-mid"] "→I" by blast
6297 qed
6298

```

```

6299 AOT_theorem "eqnotnec:5": < $\forall F \exists G (\neg \forall x ([F]x \equiv [G]x) \ \& \ \Diamond \forall x ([F]x \equiv [G]x))$ > (219.5)
6300 proof(rule GEN)
6301   fix F
6302   AOT_have Aux_A: < $\vdash_{\Box} \Diamond \psi \rightarrow \Diamond \forall x ([F]x \equiv [\lambda z [F]z \ \& \ \psi]x)$ > for  $\psi$ 
6303   proof(rule "RM $\Diamond$ "; rule " $\rightarrow$ I"; rule GEN)
6304     AOT_modally_strict {
6305       fix x
6306       AOT_assume 0: < $\psi$ >
6307       AOT_have < $[\lambda z [F]z \ \& \ \psi]x \equiv [F]x \ \& \ \psi$ >
6308         by (rule "beta-C-meta"[THEN " $\rightarrow$ E"]) "cqt:2[lambda]"
6309       also AOT_have < $\dots \equiv [F]x$ >
6310         apply (rule " $\equiv$ I"; rule " $\rightarrow$ I")
6311         using " $\forall$ E"(3)[rotated, OF "useful-tautologies:2"[THEN " $\rightarrow$ E"], OF 0] "&E"
6312         apply blast
6313         using 0 "&I" by blast
6314       finally AOT_show < $[F]x \equiv [\lambda z [F]z \ \& \ \psi]x$ >
6315         using "Commutativity of  $\equiv$ "[THEN " $\equiv$ E"(1)] by blast
6316     }
6317   qed
6318
6319   AOT_have Aux_B: < $\vdash_{\Box} \Diamond \psi \rightarrow \Diamond \forall x ([F]x \equiv [\lambda z [F]z \ \& \ \psi \vee \neg \psi]x)$ > for  $\psi$ 
6320   proof (rule "RM $\Diamond$ "; rule " $\rightarrow$ I"; rule GEN)
6321     AOT_modally_strict {
6322       fix x
6323       AOT_assume 0: < $\psi$ >
6324       AOT_have < $[\lambda z ([F]z \ \& \ \psi) \vee \neg \psi]x \equiv (([F]x \ \& \ \psi) \vee \neg \psi)$ >
6325         by (rule "beta-C-meta"[THEN " $\rightarrow$ E"]) "cqt:2[lambda]"
6326       also AOT_have < $\dots \equiv [F]x$ >
6327         apply (rule " $\equiv$ I"; rule " $\rightarrow$ I")
6328         using " $\forall$ E"(3)[rotated, OF "useful-tautologies:2"[THEN " $\rightarrow$ E"], OF 0] "&E"
6329         apply blast
6330         apply (rule " $\vee$ I"(1)) using 0 "&I" by blast
6331       finally AOT_show < $[F]x \equiv [\lambda z ([F]z \ \& \ \psi) \vee \neg \psi]x$ >
6332         using "Commutativity of  $\equiv$ "[THEN " $\equiv$ E"(1)] by blast
6333     }
6334   qed
6335
6336   AOT_have Aux_C: < $\vdash_{\Box} \neg \psi \rightarrow \neg \forall z ([\lambda z [F]z \ \& \ \psi]z \equiv [\lambda z [F]z \ \& \ \psi \vee \neg \psi]z)$ > for  $\psi$ 
6337   proof(rule " $\rightarrow$ I"; rule "raa-cor:2")
6338     AOT_modally_strict {
6339       AOT_assume 0: < $\neg \psi$ >
6340       AOT_assume < $\forall z ([\lambda z [F]z \ \& \ \psi]z \equiv [\lambda z [F]z \ \& \ \psi \vee \neg \psi]z)$ >
6341       AOT_hence < $[\lambda z [F]z \ \& \ \psi]z \equiv [\lambda z [F]z \ \& \ \psi \vee \neg \psi]z$ > for z
6342         using " $\forall$ E" by blast
6343       moreover AOT_have < $[\lambda z [F]z \ \& \ \psi]z \equiv [F]z \ \& \ \psi$ > for z
6344         by (rule "beta-C-meta"[THEN " $\rightarrow$ E"]) "cqt:2[lambda]"
6345       moreover AOT_have < $[\lambda z ([F]z \ \& \ \psi) \vee \neg \psi]z \equiv (([F]z \ \& \ \psi) \vee \neg \psi)$ > for z
6346         by (rule "beta-C-meta"[THEN " $\rightarrow$ E"]) "cqt:2[lambda]"
6347       ultimately AOT_have < $[F]z \ \& \ \psi \equiv (([F]z \ \& \ \psi) \vee \neg \psi)$ > for z
6348         using "Commutativity of  $\equiv$ "[THEN " $\equiv$ E"(1)] " $\equiv$ E"(5) by meson
6349       moreover AOT_have < $(([F]z \ \& \ \psi) \vee \neg \psi)$ > for z
6350         using 0 " $\vee$ I" by blast
6351       ultimately AOT_have < $\psi$ > using " $\equiv$ E" "&E" by metis
6352       AOT_thus < $\psi \ \& \ \neg \psi$ > using 0 "&I" by blast
6353     }
6354   qed
6355
6356   AOT_have Aux_D: < $\forall z ([F]z \equiv [\lambda z [F]z \ \& \ \psi]z) \rightarrow$ 
6357     < $(\neg \forall x ([\lambda z [F]z \ \& \ \psi]x \equiv [\lambda z [F]z \ \& \ \psi \vee \neg \psi]x) \equiv$ 
6358       < $\neg \forall x ([F]x \equiv [\lambda z [F]z \ \& \ \psi \vee \neg \psi]x))$ > for  $\psi$ 
6359   proof (rule " $\rightarrow$ I"; rule " $\equiv$ I";
6360     (rule "useful-tautologies:5"[THEN " $\rightarrow$ E"]; rule " $\rightarrow$ I"?)
6361     AOT_modally_strict {

```



```

6362   AOT_assume <∀z ([F]z ≡ [λz [F]z & ψ]z)>
6363   AOT_hence 1: <[F]z ≡ [λz [F]z & ψ]z> for z
6364     using "∀E" by blast
6365   AOT_assume <∀x ([F]x ≡ [λz [F]z & ψ ∨ ¬ψ]x)>
6366   AOT_hence 2: <[F]z ≡ [λz [F]z & ψ ∨ ¬ψ]z> for z
6367     using "∀E" by blast
6368   AOT_have <[λz [F]z & ψ]z ≡ [λz [F]z & ψ ∨ ¬ψ]z> for z
6369     using "≡E" 1 2 by meson
6370   AOT_thus <∀x ([λz [F]z & ψ]x ≡ [λz [F]z & ψ ∨ ¬ψ]x)>
6371     by (rule GEN)
6372 }
6373 next
6374   AOT_modally_strict {
6375     AOT_assume <∀z ([F]z ≡ [λz [F]z & ψ]z)>
6376     AOT_hence 1: <[F]z ≡ [λz [F]z & ψ]z> for z
6377       using "∀E" by blast
6378     AOT_assume <∀x ([λz [F]z & ψ]x ≡ [λz [F]z & ψ ∨ ¬ψ]x)>
6379     AOT_hence 2: <[λz [F]z & ψ]z ≡ [λz [F]z & ψ ∨ ¬ψ]z> for z
6380       using "∀E" by blast
6381     AOT_have <[F]z ≡ [λz [F]z & ψ ∨ ¬ψ]z> for z
6382       using 1 2 "≡E" by meson
6383     AOT_thus <∀x ([F]x ≡ [λz [F]z & ψ ∨ ¬ψ]x)>
6384       by (rule GEN)
6385   }
6386 qed
6387
6388 AOT_obtain p1 where p1_prop: <¬p1 & ◇p1>
6389   using "cont-tf-thm:2" "∃E"[rotated] "cont-tf:2"[THEN "≡defE"] by blast
6390 {
6391   AOT_assume 1: <∀x([F]x ≡ [λz [F]z & p1]x)>
6392   AOT_have 2: <◇∀x([F]x ≡ [λz [F]z & p1 ∨ ¬p1]x)>
6393     using Aux_B[THEN "→E", OF p1_prop[THEN "&E"(2)]] .
6394   AOT_have <¬∀x([λz [F]z & p1]x ≡ [λz [F]z & p1 ∨ ¬p1]x)>
6395     using Aux_C[THEN "→E", OF p1_prop[THEN "&E"(1)]] .
6396   AOT_hence 3: <¬∀x([F]x ≡ [λz [F]z & p1 ∨ ¬p1]x)>
6397     using Aux_D[THEN "→E", OF 1, THEN "≡E"(1)] by blast
6398   AOT_hence <¬∀x([F]x ≡ [λz [F]z & p1 ∨ ¬p1]x) &
6399     ◇∀x([F]x ≡ [λz [F]z & p1 ∨ ¬p1]x)>
6400     using 2 "&I" by blast
6401   AOT_hence <∃G (¬∀x ([F]x ≡ [G]x) & ◇∀x([F]x ≡ [G]x))>
6402     by (rule "∃I"(1)) "cqt:2[lambda]"
6403 }
6404 moreover {
6405   AOT_assume 2: <¬∀x([F]x ≡ [λz [F]z & p1]x)>
6406   AOT_hence <¬∀x([F]x ≡ [λz [F]z & p1]x)>
6407     using "KBasic:11"[THEN "≡E"(1)] by blast
6408   AOT_hence <¬∀x ([F]x ≡ [λz [F]z & p1]x) &
6409     ◇∀x([F]x ≡ [λz [F]z & p1]x)>
6410     using Aux_A[THEN "→E", OF p1_prop[THEN "&E"(2)]] "&I" by blast
6411   AOT_hence <∃G (¬∀x ([F]x ≡ [G]x) & ◇∀x([F]x ≡ [G]x))>
6412     by (rule "∃I"(1)) "cqt:2[lambda]"
6413 }
6414 ultimately AOT_show <∃G (¬∀x ([F]x ≡ [G]x) & ◇∀x([F]x ≡ [G]x))>
6415   using "∀E"(1)[OF "exc-mid"] "→I" by blast
6416 qed
6417
6418 AOT_theorem "eqnotnec:6": <∀F∃G(⊧¬∀x([F]x ≡ [G]x) & ◇∀x([F]x ≡ [G]x))>
6419 proof(rule GEN)
6420   fix F
6421   AOT_have Aux_A: <⊧◇ψ → ◇∀x([F]x ≡ [λz [F]z & ψ]x)> for ψ
6422   proof(rule "RM◇"; rule "→I"; rule GEN)
6423     AOT_modally_strict {
6424       fix x

```



```

6425 AOT_assume 0: < $\psi$ >
6426 AOT_have < $[\lambda z [F]z \ \& \ \psi]x \equiv [F]x \ \& \ \psi$ >
6427   by (rule "beta-C-meta"[THEN " $\rightarrow$ E"]) "cqt:2[lambda]"
6428 also AOT_have < $\dots \equiv [F]x$ >
6429   apply (rule " $\equiv$ I"; rule " $\rightarrow$ I")
6430   using " $\vee$ E"(3)[rotated, OF "useful-tautologies:2"[THEN " $\rightarrow$ E"], OF 0]
6431   "&E"
6432   apply blast
6433   using 0 "&I" by blast
6434 finally AOT_show < $[F]x \equiv [\lambda z [F]z \ \& \ \psi]x$ >
6435   using "Commutativity of  $\equiv$ "[THEN " $\equiv$ E"(1)] by blast
6436 }
6437 qed
6438
6439 AOT_have Aux_B: < $\vdash_{\square} \Diamond \psi \rightarrow \Diamond \forall x ([F]x \equiv [\lambda z [F]z \ \& \ \psi \vee \neg \psi]x)$ > for  $\psi$ 
6440 proof (rule "RM $\Diamond$ "; rule " $\rightarrow$ I"; rule GEN)
6441   AOT_modally_strict {
6442     fix x
6443     AOT_assume 0: < $\psi$ >
6444     AOT_have < $[\lambda z ([F]z \ \& \ \psi) \vee \neg \psi]x \equiv (([F]x \ \& \ \psi) \vee \neg \psi)$ >
6445       by (rule "beta-C-meta"[THEN " $\rightarrow$ E"]) "cqt:2[lambda]"
6446     also AOT_have < $\dots \equiv [F]x$ >
6447       apply (rule " $\equiv$ I"; rule " $\rightarrow$ I")
6448       using " $\vee$ E"(3)[rotated, OF "useful-tautologies:2"[THEN " $\rightarrow$ E"], OF 0] "&E"
6449       apply blast
6450       apply (rule " $\vee$ I"(1)) using 0 "&I" by blast
6451     finally AOT_show < $[F]x \equiv [\lambda z ([F]z \ \& \ \psi) \vee \neg \psi]x$ >
6452       using "Commutativity of  $\equiv$ "[THEN " $\equiv$ E"(1)] by blast
6453   }
6454 qed
6455
6456 AOT_have Aux_C:
6457   < $\vdash_{\square} \mathcal{A} \neg \psi \rightarrow \mathcal{A} \neg \forall z ([\lambda z [F]z \ \& \ \psi]z \equiv [\lambda z [F]z \ \& \ \psi \vee \neg \psi]z)$ > for  $\psi$ 
6458 proof (rule "act-cond"[THEN " $\rightarrow$ E"]; rule "RA[2]"; rule " $\rightarrow$ I"; rule "raa-cor:2")
6459   AOT_modally_strict {
6460     AOT_assume 0: < $\neg \psi$ >
6461     AOT_assume < $\forall z ([\lambda z [F]z \ \& \ \psi]z \equiv [\lambda z [F]z \ \& \ \psi \vee \neg \psi]z)$ >
6462     AOT_hence < $[\lambda z [F]z \ \& \ \psi]z \equiv [\lambda z [F]z \ \& \ \psi \vee \neg \psi]z$ > for z
6463       using " $\forall$ E" by blast
6464     moreover AOT_have < $[\lambda z [F]z \ \& \ \psi]z \equiv [F]z \ \& \ \psi$ > for z
6465       by (rule "beta-C-meta"[THEN " $\rightarrow$ E"]) "cqt:2[lambda]"
6466     moreover AOT_have < $[\lambda z ([F]z \ \& \ \psi) \vee \neg \psi]z \equiv (([F]z \ \& \ \psi) \vee \neg \psi)$ > for z
6467       by (rule "beta-C-meta"[THEN " $\rightarrow$ E"]) "cqt:2[lambda]"
6468     ultimately AOT_have < $[F]z \ \& \ \psi \equiv (([F]z \ \& \ \psi) \vee \neg \psi)$ > for z
6469       using "Commutativity of  $\equiv$ "[THEN " $\equiv$ E"(1)] " $\equiv$ E"(5) by meson
6470     moreover AOT_have < $(([F]z \ \& \ \psi) \vee \neg \psi)$ > for z
6471       using 0 " $\vee$ I" by blast
6472     ultimately AOT_have < $\psi$ > using " $\equiv$ E" "&E" by metis
6473     AOT_thus < $\psi \ \& \ \neg \psi$ > using 0 "&I" by blast
6474   }
6475 qed
6476
6477 AOT_have < $\square (\forall z ([F]z \equiv [\lambda z [F]z \ \& \ \psi]z) \rightarrow$ 
6478   < $\neg \forall x ([\lambda z [F]z \ \& \ \psi]x \equiv [\lambda z [F]z \ \& \ \psi \vee \neg \psi]x) \equiv$ 
6479   < $\neg \forall x ([F]x \equiv [\lambda z [F]z \ \& \ \psi \vee \neg \psi]x))$ > for  $\psi$ 
6480 proof (rule RN; rule " $\rightarrow$ I")
6481   AOT_modally_strict {
6482     AOT_assume < $\forall z ([F]z \equiv [\lambda z [F]z \ \& \ \psi]z)$ >
6483     AOT_thus < $\neg \forall x ([\lambda z [F]z \ \& \ \psi]x \equiv [\lambda z [F]z \ \& \ \psi \vee \neg \psi]x) \equiv$ 
6484       < $\neg \forall x ([F]x \equiv [\lambda z [F]z \ \& \ \psi \vee \neg \psi]x)$ >
6485     apply -
6486     proof (rule " $\equiv$ I"; (rule "useful-tautologies:5"[THEN " $\rightarrow$ E"]; rule " $\rightarrow$ I")?)
6487       AOT_assume < $\forall z ([F]z \equiv [\lambda z [F]z \ \& \ \psi]z)$ >

```

```

6488     AOT_hence 1: <[F]z ≡ [λz [F]z & ψ]z> for z
6489         using "∀E" by blast
6490     AOT_assume <∀x ([F]x ≡ [λz [F]z & ψ ∨ ¬ψ]x)>
6491     AOT_hence 2: <[F]z ≡ [λz [F]z & ψ ∨ ¬ψ]z> for z
6492         using "∀E" by blast
6493     AOT_have <[λz [F]z & ψ]z ≡ [λz [F]z & ψ ∨ ¬ψ]z> for z
6494         using "≡E" 1 2 by meson
6495     AOT_thus <∀x ([λz [F]z & ψ]x ≡ [λz [F]z & ψ ∨ ¬ψ]x)>
6496         by (rule GEN)
6497     next
6498     AOT_assume <∀z ([F]z ≡ [λz [F]z & ψ]z)>
6499     AOT_hence 1: <[F]z ≡ [λz [F]z & ψ]z> for z
6500         using "∀E" by blast
6501     AOT_assume <∀x ([λz [F]z & ψ]x ≡ [λz [F]z & ψ ∨ ¬ψ]x)>
6502     AOT_hence 2: <[λz [F]z & ψ]z ≡ [λz [F]z & ψ ∨ ¬ψ]z> for z
6503         using "∀E" by blast
6504     AOT_have <[F]z ≡ [λz [F]z & ψ ∨ ¬ψ]z> for z
6505         using 1 2 "≡E" by meson
6506     AOT_thus <∀x ([F]x ≡ [λz [F]z & ψ ∨ ¬ψ]x)>
6507         by (rule GEN)
6508     qed
6509 }
6510 qed
6511 AOT_hence <ℳ(∀z ([F]z ≡ [λz [F]z & ψ]z) →
6512   (¬∀x ([λz [F]z & ψ]x ≡ [λz [F]z & ψ ∨ ¬ψ]x) ≡
6513     ¬∀x ([F]x ≡ [λz [F]z & ψ ∨ ¬ψ]x)))> for ψ
6514     using "nec-imp-act"[THEN "→E"] by blast
6515 AOT_hence <ℳ∀z ([F]z ≡ [λz [F]z & ψ]z) →
6516   ℳ(¬∀x ([λz [F]z & ψ]x ≡ [λz [F]z & ψ ∨ ¬ψ]x) ≡
6517     ¬∀x ([F]x ≡ [λz [F]z & ψ ∨ ¬ψ]x))> for ψ
6518     using "act-cond"[THEN "→E"] by blast
6519 AOT_hence Aux_D: <ℳ∀z ([F]z ≡ [λz [F]z & ψ]z) →
6520   (ℳ¬∀x ([λz [F]z & ψ]x ≡ [λz [F]z & ψ ∨ ¬ψ]x) ≡
6521     ℳ¬∀x ([F]x ≡ [λz [F]z & ψ ∨ ¬ψ]x))> for ψ
6522     by (auto intro!: "→I" "Act-Basic:5"[THEN "≡E"(1)] dest!: "→E")
6523
6524 AOT_have <¬ℳq₀>
6525     apply (rule "=defI"(2)[OF q₀_def])
6526     apply (fact "log-prop-prop:2")
6527     by (fact AOT)
6528 AOT_hence q₀_prop_1: <ℳ¬q₀>
6529     using "logic-actual-nec:1"[axiom_inst, THEN "≡E"(2)] by blast
6530 {
6531     AOT_assume 1: <ℳ∀x([F]x ≡ [λz [F]z & q₀]x)>
6532     AOT_have 2: <◇∀x([F]x ≡ [λz [F]z & q₀ ∨ ¬q₀]x)>
6533         using Aux_B[THEN "→E", OF q₀_prop[THEN "&E"(1)]] .
6534     AOT_have <ℳ¬∀x([λz [F]z & q₀]x ≡ [λz [F]z & q₀ ∨ ¬q₀]x)>
6535         using Aux_C[THEN "→E", OF q₀_prop_1] .
6536     AOT_hence 3: <ℳ¬∀x([F]x ≡ [λz [F]z & q₀ ∨ ¬q₀]x)>
6537         using Aux_D[THEN "→E", OF 1, THEN "≡E"(1)] by blast
6538     AOT_hence <ℳ¬∀x([F]x ≡ [λz [F]z & q₀ ∨ ¬q₀]x) &
6539       ◇∀x([F]x ≡ [λz [F]z & q₀ ∨ ¬q₀]x)>
6540         using 2 "&I" by blast
6541     AOT_hence <∃G (ℳ¬∀x ([F]x ≡ [G]x) & ◇∀x([F]x ≡ [G]x))>
6542         by (rule "∃I"(1)) "cqt:2[lambda]"
6543 }
6544 moreover {
6545     AOT_assume 2: <¬ℳ∀x([F]x ≡ [λz [F]z & q₀]x)>
6546     AOT_hence <ℳ¬∀x([F]x ≡ [λz [F]z & q₀]x)>
6547         using "logic-actual-nec:1"[axiom_inst, THEN "≡E"(2)] by blast
6548     AOT_hence <ℳ¬∀x ([F]x ≡ [λz [F]z & q₀]x) & ◇∀x([F]x ≡ [λz [F]z & q₀]x)>
6549         using Aux_A[THEN "→E", OF q₀_prop[THEN "&E"(1)]] "&I" by blast
6550     AOT_hence <∃G (ℳ¬∀x ([F]x ≡ [G]x) & ◇∀x([F]x ≡ [G]x))>

```

```

6551     by (rule "∃I"(1)) "cqt:2[lambda]"
6552   }
6553   ultimately AOT_show <∃G (A¬∀x ([F]x ≡ [G]x) & ◇∀x([F]x ≡ [G]x))>
6554   using "∨E"(1)[OF "exc-mid"] "→I" by blast
6555 qed
6556
6557 AOT_theorem "oa-contingent:1": <O! ≠ A!> (220.1)
6558 proof(rule "≡dfI"[OF "-infix"]; rule "raa-cor:2")
6559   fix x
6560   AOT_assume 1: <O! = A!>
6561   AOT_hence <[λx ◇E!x] = A!>
6562     by (rule "=dfE"(2)[OF AOT_ordinary, rotated]) "cqt:2[lambda]"
6563   AOT_hence <[λx ◇E!x] = [λx ¬◇E!x]>
6564     by (rule "=dfE"(2)[OF AOT_abstract, rotated]) "cqt:2[lambda]"
6565   moreover AOT_have <[λx ◇E!x]x ≡ ◇E!x>
6566     by (rule "beta-C-meta"[THEN "→E"]) "cqt:2[lambda]"
6567   ultimately AOT_have <[λx ¬◇E!x]x ≡ ◇E!x>
6568     using "rule=E" by fast
6569   moreover AOT_have <[λx ¬◇E!x]x ≡ ¬◇E!x>
6570     by (rule "beta-C-meta"[THEN "→E"]) "cqt:2[lambda]"
6571   ultimately AOT_have <◇E!x ≡ ¬◇E!x>
6572     using "≡E"(6) "Commutativity of ≡"[THEN "≡E"(1)] by blast
6573   AOT_thus "(◇E!x ≡ ¬◇E!x) & ¬(◇E!x ≡ ¬◇E!x)"
6574     using "oth-class-taut:3:c" "&I" by blast
6575 qed
6576
6577 AOT_theorem "oa-contingent:2": <O!x ≡ ¬A!x> (220.2)
6578 proof -
6579   AOT_have <O!x ≡ [λx ◇E!x]x>
6580     apply (rule "≡I"; rule "→I")
6581     apply (rule "=dfE"(2)[OF AOT_ordinary])
6582     apply "cqt:2[lambda]"
6583     apply argo
6584     apply (rule "=dfI"(2)[OF AOT_ordinary])
6585     apply "cqt:2[lambda]"
6586     by argo
6587   also AOT_have <... ≡ ◇E!x>
6588     by (rule "beta-C-meta"[THEN "→E"]) "cqt:2[lambda]"
6589   also AOT_have <... ≡ ¬¬◇E!x>
6590     using "oth-class-taut:3:b".
6591   also AOT_have <... ≡ ¬[λx ¬◇E!x]x>
6592     by (rule "beta-C-meta"[THEN "→E"],
6593         THEN "oth-class-taut:4:b"[THEN "≡E"(1)], symmetric])
6594     "cqt:2"
6595   also AOT_have <... ≡ ¬A!x>
6596     apply (rule "≡I"; rule "→I")
6597     apply (rule "=dfI"(2)[OF AOT_abstract])
6598     apply "cqt:2[lambda]"
6599     apply argo
6600     apply (rule "=dfE"(2)[OF AOT_abstract])
6601     apply "cqt:2[lambda]"
6602     by argo
6603   finally show ?thesis.
6604 qed
6605
6606 AOT_theorem "oa-contingent:3": <A!x ≡ ¬O!x> (220.3)
6607   by (AOT_subst <A!x> <¬¬A!x>)
6608     (auto simp add: "oth-class-taut:3:b" "oa-contingent:2"[THEN
6609         "oth-class-taut:4:b"[THEN "≡E"(1)], symmetric])
6610
6611 AOT_theorem "oa-contingent:4": <Contingent(O!)> (220.4)
6612 proof (rule "thm-cont-prop:2"[unvarify F, OF "oa-exist:1", THEN "≡E"(2)];
6613     rule "&I")

```

```

6614 AOT_have <◇∃x E!x> using "thm-cont-e:3" .
6615 AOT_hence <∃x ◇E!x> using "BF◇"[THEN "→E"] by blast
6616 then AOT_obtain a where <◇E!a> using "∃E"[rotated] by blast
6617 AOT_hence <[λx ◇E!x]a>
6618   by (rule "beta-C-meta"[THEN "→E", THEN "≡E"(2), rotated]) "cqt:2"
6619 AOT_hence <O!a>
6620   by (rule "=dfI"(2)[OF AOT_ordinary, rotated]) "cqt:2"
6621 AOT_hence <∃x O!x> using "∃I" by blast
6622 AOT_thus <◇∃x O!x> using "T◇"[THEN "→E"] by blast
6623 next
6624 AOT_obtain a where <A!a>
6625   using "A-objects"[axiom_inst] "∃E"[rotated] "&E" by blast
6626 AOT_hence <¬O!a> using "oa-contingent:3"[THEN "≡E"(1)] by blast
6627 AOT_hence <∃x ¬O!x> using "∃I" by fast
6628 AOT_thus <◇∃x ¬O!x> using "T◇"[THEN "→E"] by blast
6629 qed
6630
6631 AOT_theorem "oa-contingent:5": <Contingent(A!)> (220.5)
6632 proof (rule "thm-cont-prop:2"[unvarify F, OF "oa-exist:2", THEN "≡E"(2)];
6633   rule "&I")
6634   AOT_obtain a where <A!a>
6635     using "A-objects"[axiom_inst] "∃E"[rotated] "&E" by blast
6636 AOT_hence <∃x A!x> using "∃I" by fast
6637 AOT_thus <◇∃x A!x> using "T◇"[THEN "→E"] by blast
6638 next
6639 AOT_have <◇∃x E!x> using "thm-cont-e:3" .
6640 AOT_hence <∃x ◇E!x> using "BF◇"[THEN "→E"] by blast
6641 then AOT_obtain a where <◇E!a> using "∃E"[rotated] by blast
6642 AOT_hence <[λx ◇E!x]a>
6643   by (rule "beta-C-meta"[THEN "→E", THEN "≡E"(2), rotated]) "cqt:2[lambda]"
6644 AOT_hence <O!a>
6645   by (rule "=dfI"(2)[OF AOT_ordinary, rotated]) "cqt:2[lambda]"
6646 AOT_hence <¬A!a> using "oa-contingent:2"[THEN "≡E"(1)] by blast
6647 AOT_hence <∃x ¬A!x> using "∃I" by fast
6648 AOT_thus <◇∃x ¬A!x> using "T◇"[THEN "→E"] by blast
6649 qed
6650
6651 AOT_theorem "oa-contingent:7": <O!¬x ≡ ¬A!¬x> (220.7)
6652 proof -
6653   AOT_have <O!x ≡ ¬A!x>
6654     using "oa-contingent:2" by blast
6655   also AOT_have <... ≡ A!¬x>
6656     using "thm-relation-negation:1"[symmetric, unvarify F, OF "oa-exist:2"].
6657   finally AOT_have 1: <O!x ≡ A!¬x>.
6658
6659   AOT_have <A!x ≡ ¬O!x>
6660     using "oa-contingent:3" by blast
6661   also AOT_have <... ≡ O!¬x>
6662     using "thm-relation-negation:1"[symmetric, unvarify F, OF "oa-exist:1"].
6663   finally AOT_have 2: <A!x ≡ O!¬x>.
6664
6665   AOT_show <O!¬x ≡ ¬A!¬x>
6666     using 1[THEN "oth-class-taut:4:b"[THEN "≡E"(1)]]
6667     "oa-contingent:3"[of _ x] 2[symmetric]
6668     "≡E"(5) by blast
6669 qed
6670
6671 AOT_theorem "oa-contingent:6": <O!¬ ≠ A!¬> (220.6)
6672 proof (rule "=-infix"[THEN "≡dfI"]; rule "raa-cor:2")
6673   AOT_assume 1: <O!¬ = A!¬>
6674   fix x
6675   AOT_have <A!¬x ≡ O!¬x>
6676   apply (rule "rule=E"[rotated, OF 1])

```

```

6677   by (fact "oth-class-taut:3:a")
6678   AOT_hence <A!~x ≡ ¬A!~x>
6679   using "oa-contingent:7" "≡E" by fast
6680   AOT_thus <(A!~x ≡ ¬A!~x) & ¬(A!~x ≡ ¬A!~x)>
6681   using "oth-class-taut:3:c" "&I" by blast
6682 qed
6683
6684 AOT_theorem "oa-contingent:8": <Contingent(O!~)> (220.8)
6685   using "thm-cont-prop:3"[unvarify F, OF "oa-exist:1", THEN "≡E"(1),
6686   OF "oa-contingent:4"].
6687
6688 AOT_theorem "oa-contingent:9": <Contingent(A!~)> (220.9)
6689   using "thm-cont-prop:3"[unvarify F, OF "oa-exist:2", THEN "≡E"(1),
6690   OF "oa-contingent:5"].
6691
6692 AOT_define WeaklyContingent :: <Π ⇒ φ> (<WeaklyContingent'('_)>)
6693   "df-cont-nec": (221)
6694   <WeaklyContingent([F]) ≡df Contingent([F]) & ∀x (◇[F]x → □[F]x)>
6695
6696 AOT_theorem "cont-nec-fact1:1": (222.1)
6697   <WeaklyContingent([F]) ≡ WeaklyContingent([F]~)>
6698 proof -
6699   AOT_have <WeaklyContingent([F]) ≡ Contingent([F]) & ∀x (◇[F]x → □[F]x)>
6700   using "df-cont-nec"[THEN "≡df"] by blast
6701   also AOT_have <... ≡ Contingent([F]~) & ∀x (◇[F]x → □[F]x)>
6702   apply (rule "oth-class-taut:8:f"[THEN "≡E"(2)]; rule "→I")
6703   using "thm-cont-prop:3".
6704   also AOT_have <... ≡ Contingent([F]~) & ∀x (◇[F]~x → □[F]~x)>
6705   proof (rule "oth-class-taut:8:e"[THEN "≡E"(2)];
6706     rule "→I"; rule "≡I"; rule "→I"; rule GEN; rule "→I")
6707     fix x
6708     AOT_assume 0: <∀x (◇[F]x → □[F]x)>
6709     AOT_assume 1: <◇[F]~x>
6710     AOT_have <◇¬[F]x>
6711     by (AOT_subst (reverse) <¬[F]x> <[F]~x>)
6712     (auto simp add: "thm-relation-negation:1" 1)
6713     AOT_hence 2: <¬□[F]x>
6714     using "KBasic:11"[THEN "≡E"(2)] by blast
6715     AOT_show <□[F]~x>
6716     proof (rule "raa-cor:1")
6717       AOT_assume 3: <¬□[F]~x>
6718       AOT_have <¬□¬[F]x>
6719       by (AOT_subst (reverse) <¬[F]x> <[F]~x>)
6720       (auto simp add: "thm-relation-negation:1" 3)
6721       AOT_hence <◇[F]x>
6722       using "conventions:5"[THEN "≡dfI"] by simp
6723       AOT_hence <□[F]x> using 0 "∀E" "→E" by fast
6724       AOT_thus <□[F]x & ¬□[F]x> using "&I" 2 by blast
6725     qed
6726   next
6727   fix x
6728   AOT_assume 0: <∀x (◇[F]~x → □[F]~x)>
6729   AOT_assume 1: <◇[F]x>
6730   AOT_have <◇¬[F]~x>
6731   by (AOT_subst <¬[F]~x> <[F]x>)
6732   (auto simp: "thm-relation-negation:2" 1)
6733   AOT_hence 2: <¬□[F]~x>
6734   using "KBasic:11"[THEN "≡E"(2)] by blast
6735   AOT_show <□[F]x>
6736   proof (rule "raa-cor:1")
6737     AOT_assume 3: <¬□[F]~x>
6738     AOT_have <¬□¬[F]~x>
6739     by (AOT_subst <¬[F]~x> <[F]x>)

```

```

6740     (auto simp add: "thm-relation-negation:2" 3)
6741     AOT_hence <◇[F]-x>
6742     using "conventions:5"[THEN "≡dfI"] by simp
6743     AOT_hence <□[F]-x> using 0 "∀E" "→E" by fast
6744     AOT_thus <□[F]-x & ¬□[F]-x> using "&I" 2 by blast
6745   qed
6746   qed
6747   also AOT_have <... ≡ WeaklyContingent([F]-)>
6748   using "df-cont-nec"[THEN "≡Df", symmetric] by blast
6749   finally show ?thesis.
6750 qed
6751
6752 AOT_theorem "cont-nec-fact1:2": (222.2)
6753   <(WeaklyContingent([F]) & ¬WeaklyContingent([G])) → F ≠ G>
6754 proof (rule "→I"; rule "--infix"[THEN "≡dfI"]; rule "raa-cor:2")
6755   AOT_assume 1: <WeaklyContingent([F]) & ¬WeaklyContingent([G])>
6756   AOT_hence <WeaklyContingent([F])> using "&E" by blast
6757   moreover AOT_assume <F = G>
6758   ultimately AOT_have <WeaklyContingent([G])>
6759   using "rule=E" by blast
6760   AOT_thus <WeaklyContingent([G]) & ¬WeaklyContingent([G])>
6761   using 1 "&I" "&E" by blast
6762 qed
6763
6764 AOT_theorem "cont-nec-fact2:1": <WeaklyContingent(O!)> (223.1)
6765 proof (rule "df-cont-nec"[THEN "≡dfI"]; rule "&I")
6766   AOT_show <Contingent(O!)>
6767   using "oa-contingent:4".
6768 next
6769   AOT_show <∀x (◇[O!]x → □[O!]x)>
6770   apply (rule GEN; rule "→I")
6771   using "oa-facts:5"[THEN "≡E"(1)] by blast
6772 qed
6773
6774
6775 AOT_theorem "cont-nec-fact2:2": <WeaklyContingent(A!)> (223.2)
6776 proof (rule "df-cont-nec"[THEN "≡dfI"]; rule "&I")
6777   AOT_show <Contingent(A!)>
6778   using "oa-contingent:5".
6779 next
6780   AOT_show <∀x (◇[A!]x → □[A!]x)>
6781   apply (rule GEN; rule "→I")
6782   using "oa-facts:6"[THEN "≡E"(1)] by blast
6783 qed
6784
6785 AOT_theorem "cont-nec-fact2:3": <¬WeaklyContingent(E!)> (223.3)
6786 proof (rule "df-cont-nec"[THEN "≡Df",
6787   THEN "oth-class-taut:4:b"[THEN "≡E"(1)],
6788   THEN "≡E"(2)];
6789   rule DeMorgan(1)[THEN "≡E"(2)]; rule "∀I"(2); rule "raa-cor:2")
6790   AOT_have <◇∃x (E!x & ¬ $\mathcal{A}E!x$ )> using "qml:4"[axiom_inst].
6791   AOT_hence <∃x ◇(E!x & ¬ $\mathcal{A}E!x$ )> using "BF◇"[THEN "→E"] by blast
6792   then AOT_obtain a where <◇(E!a & ¬ $\mathcal{A}E!a$ )> using "∃E"[rotated] by blast
6793   AOT_hence 1: <◇E!a & ◇¬ $\mathcal{A}E!a$ > using "KBasic2:3"[THEN "→E"] by simp
6794   moreover AOT_assume <∀x (◇[E!]x → □[E!]x)>
6795   ultimately AOT_have <□E!a> using "&E" "∀E" "→E" by fast
6796   AOT_hence < $\mathcal{A}E!a$ > using "nec-imp-act"[THEN "→E"] by blast
6797   AOT_hence <□ $\mathcal{A}E!a$ > using "qml-act:1"[axiom_inst, THEN "→E"] by blast
6798   moreover AOT_have <¬□ $\mathcal{A}E!a$ >
6799   using "KBasic:11"[THEN "≡E"(2)] 1[THEN "&E"(2)] by meson
6800   ultimately AOT_have <□ $\mathcal{A}E!a$  & ¬□ $\mathcal{A}E!a$ > using "&I" by blast
6801   AOT_thus <p & ¬p> for p using "raa-cor:1" by blast
6802 qed

```

```

6803
6804 AOT_theorem "cont-nec-fact2:4": <¬WeaklyContingent(L)> (223.4)
6805   apply (rule "df-cont-nec"[THEN "≡Df",
6806           THEN "oth-class-taut:4:b"[THEN "≡E"(1)],
6807           THEN "≡E"(2)]);
6808   rule DeMorgan(1)[THEN "≡E"(2)]; rule "∀I"(1)
6809   apply (rule "contingent-properties:4"
6810           [THEN "≡Df",
6811           THEN "oth-class-taut:4:b"[THEN "≡E"(1)],
6812           THEN "≡E"(2)]);
6813   apply (rule DeMorgan(1)[THEN "≡E"(2)];
6814   rule "∀I"(2);
6815   rule "useful-tautologies:2"[THEN "→E"]
6816   using "thm-noncont-e-e:3"[THEN "contingent-properties:3"[THEN "≡dfE"]].
6817
6818 AOT_theorem "cont-nec-fact2:5": <O! ≠ E! & O! ≠ E!⁻ & O! ≠ L & O! ≠ L⁻> (223.5)
6819 proof -
6820   AOT_have 1: <L↓>
6821   by (rule "=dfI"(2)[OF L_def]) "cqt:2[lambda]" +
6822   {
6823     fix φ and Π Π' :: <<κ>>
6824     AOT_have A: <¬(φ{Π'} ≡ φ{Π})> if <φ{Π}> and <¬φ{Π'}>
6825     proof (rule "raa-cor:2")
6826       AOT_assume <φ{Π'} ≡ φ{Π}>
6827       AOT_hence <φ{Π'}> using that(1) "≡E" by blast
6828       AOT_thus <φ{Π'} & ¬φ{Π'}> using that(2) "&I" by blast
6829     qed
6830     AOT_have <Π' ≠ Π> if <Π↓> and <Π'↓> and <φ{Π}> and <¬φ{Π'}>
6831     using "pos-not-equiv-ne:4"[unvarify F G, THEN "→E",
6832           OF that(1,2), OF A[OF that(3, 4)]] .
6833   } note 0 = this
6834   show ?thesis
6835   apply (safe intro!: "&I"; rule 0)
6836   apply "cqt:2"
6837   using "oa-exist:1" apply blast
6838   using "cont-nec-fact2:3" apply fast
6839   apply (rule "useful-tautologies:2"[THEN "→E"])
6840   using "cont-nec-fact2:1" apply fast
6841   using "rel-neg-T:3" apply fast
6842   using "oa-exist:1" apply blast
6843   using "cont-nec-fact1:1"[THEN "oth-class-taut:4:b"[THEN "≡E"(1)],
6844           THEN "≡E"(1), rotated, OF "cont-nec-fact2:3"] apply fast
6845   apply (rule "useful-tautologies:2"[THEN "→E"])
6846   using "cont-nec-fact2:1" apply blast
6847   apply (rule "=dfI"(2)[OF L_def]; "cqt:2[lambda]")
6848   using "oa-exist:1" apply fast
6849   using "cont-nec-fact2:4" apply fast
6850   apply (rule "useful-tautologies:2"[THEN "→E"])
6851   using "cont-nec-fact2:1" apply fast
6852   using "rel-neg-T:3" apply fast
6853   using "oa-exist:1" apply fast
6854   apply (rule "cont-nec-fact1:1"[unvarify F,
6855           THEN "oth-class-taut:4:b"[THEN "≡E"(1)],
6856           THEN "≡E"(1), rotated, OF "cont-nec-fact2:4"])
6857   apply (rule "=dfI"(2)[OF L_def]; "cqt:2[lambda]")
6858   apply (rule "useful-tautologies:2"[THEN "→E"])
6859   using "cont-nec-fact2:1" by blast
6860   qed
6861
6862 AOT_theorem "cont-nec-fact2:6": <A! ≠ E! & A! ≠ E!⁻ & A! ≠ L & A! ≠ L⁻> (223.6)
6863 proof -
6864   AOT_have 1: <L↓>
6865   by (rule "=dfI"(2)[OF L_def]) "cqt:2[lambda]" +

```



```

6866 {
6867   fix  $\varphi$  and  $\Pi \Pi'$  :: << $\kappa$ >>
6868   AOT_have A: < $\neg(\varphi\{\Pi'\} \equiv \varphi\{\Pi\})$ > if < $\varphi\{\Pi\}$ > and < $\neg\varphi\{\Pi'\}$ >
6869   proof (rule "raa-cor:2")
6870     AOT_assume < $\varphi\{\Pi'\} \equiv \varphi\{\Pi\}$ >
6871     AOT_hence < $\varphi\{\Pi'\}$ > using that(1) "≡E" by blast
6872     AOT_thus < $\varphi\{\Pi'\} \& \neg\varphi\{\Pi'\}$ > using that(2) "&I" by blast
6873   qed
6874   AOT_have < $\Pi' \neq \Pi$ > if < $\Pi\downarrow$ > and < $\Pi'\downarrow$ > and < $\varphi\{\Pi\}$ > and < $\neg\varphi\{\Pi'\}$ >
6875     using "pos-not-equiv-ne:4"[unvarify F G, THEN "→E",
6876       OF that(1,2), OF A[OF that(3, 4)]]].
6877 } note 0 = this
6878 show ?thesis
6879   apply(safe intro!: "&I"; rule 0)
6880   apply "cqt:2"
6881   using "oa-exist:2" apply blast
6882   using "cont-nec-fact2:3" apply fast
6883   apply (rule "useful-tautologies:2"[THEN "→E"])
6884   using "cont-nec-fact2:2" apply fast
6885   using "rel-neg-T:3" apply fast
6886   using "oa-exist:2" apply blast
6887   using "cont-nec-fact1:1"[THEN "oth-class-taut:4:b"[THEN "≡E"(1)],
6888     THEN "≡E"(1), rotated, OF "cont-nec-fact2:3"] apply fast
6889   apply (rule "useful-tautologies:2"[THEN "→E"])
6890   using "cont-nec-fact2:2" apply blast
6891   apply (rule "=_dfI"(2)[OF L_def]; "cqt:2[lambda]")
6892   using "oa-exist:2" apply fast
6893   using "cont-nec-fact2:4" apply fast
6894   apply (rule "useful-tautologies:2"[THEN "→E"])
6895   using "cont-nec-fact2:2" apply fast
6896   using "rel-neg-T:3" apply fast
6897   using "oa-exist:2" apply fast
6898   apply (rule "cont-nec-fact1:1"[unvarify F,
6899     THEN "oth-class-taut:4:b"[THEN "≡E"(1)],
6900     THEN "≡E"(1), rotated, OF "cont-nec-fact2:4"])
6901   apply (rule "=_dfI"(2)[OF L_def]; "cqt:2[lambda]")
6902   apply (rule "useful-tautologies:2"[THEN "→E"])
6903   using "cont-nec-fact2:2" by blast
6904 qed
6905
6906 AOT_define necessary_or_contingently_false :: < $\varphi \Rightarrow \varphi$ > (" $\Delta$ _" [49] 54)
6907   < $\Delta p \equiv_{df} \Box p \vee (\neg \mathcal{A}p \& \diamond p)$ >
6908
6909 AOT_theorem sixteen:
6910 shows < $\exists F_1 \exists F_2 \exists F_3 \exists F_4 \exists F_5 \exists F_6 \exists F_7 \exists F_8 \exists F_9 \exists F_{10} \exists F_{11} \exists F_{12} \exists F_{13} \exists F_{14} \exists F_{15} \exists F_{16}$  (
6911   < $F_1 :: \langle \kappa \rangle \neq F_2 \& F_1 \neq F_3 \& F_1 \neq F_4 \& F_1 \neq F_5 \& F_1 \neq F_6 \& F_1 \neq F_7 \&$ 
6912   < $F_1 \neq F_8 \& F_1 \neq F_9 \& F_1 \neq F_{10} \& F_1 \neq F_{11} \& F_1 \neq F_{12} \& F_1 \neq F_{13} \&$ 
6913   < $F_1 \neq F_{14} \& F_1 \neq F_{15} \& F_1 \neq F_{16} \&$ 
6914   < $F_2 \neq F_3 \& F_2 \neq F_4 \& F_2 \neq F_5 \& F_2 \neq F_6 \& F_2 \neq F_7 \& F_2 \neq F_8 \&$ 
6915   < $F_2 \neq F_9 \& F_2 \neq F_{10} \& F_2 \neq F_{11} \& F_2 \neq F_{12} \& F_2 \neq F_{13} \& F_2 \neq F_{14} \&$ 
6916   < $F_2 \neq F_{15} \& F_2 \neq F_{16} \&$ 
6917   < $F_3 \neq F_4 \& F_3 \neq F_5 \& F_3 \neq F_6 \& F_3 \neq F_7 \& F_3 \neq F_8 \& F_3 \neq F_9 \& F_3 \neq F_{10} \&$ 
6918   < $F_3 \neq F_{11} \& F_3 \neq F_{12} \& F_3 \neq F_{13} \& F_3 \neq F_{14} \& F_3 \neq F_{15} \& F_3 \neq F_{16} \&$ 
6919   < $F_4 \neq F_5 \& F_4 \neq F_6 \& F_4 \neq F_7 \& F_4 \neq F_8 \& F_4 \neq F_9 \& F_4 \neq F_{10} \& F_4 \neq F_{11} \&$ 
6920   < $F_4 \neq F_{12} \& F_4 \neq F_{13} \& F_4 \neq F_{14} \& F_4 \neq F_{15} \& F_4 \neq F_{16} \&$ 
6921   < $F_5 \neq F_6 \& F_5 \neq F_7 \& F_5 \neq F_8 \& F_5 \neq F_9 \& F_5 \neq F_{10} \& F_5 \neq F_{11} \& F_5 \neq F_{12} \&$ 
6922   < $F_5 \neq F_{13} \& F_5 \neq F_{14} \& F_5 \neq F_{15} \& F_5 \neq F_{16} \&$ 
6923   < $F_6 \neq F_7 \& F_6 \neq F_8 \& F_6 \neq F_9 \& F_6 \neq F_{10} \& F_6 \neq F_{11} \& F_6 \neq F_{12} \& F_6 \neq F_{13} \&$ 
6924   < $F_6 \neq F_{14} \& F_6 \neq F_{15} \& F_6 \neq F_{16} \&$ 
6925   < $F_7 \neq F_8 \& F_7 \neq F_9 \& F_7 \neq F_{10} \& F_7 \neq F_{11} \& F_7 \neq F_{12} \& F_7 \neq F_{13} \& F_7 \neq F_{14} \&$ 
6926   < $F_7 \neq F_{15} \& F_7 \neq F_{16} \&$ 
6927   < $F_8 \neq F_9 \& F_8 \neq F_{10} \& F_8 \neq F_{11} \& F_8 \neq F_{12} \& F_8 \neq F_{13} \& F_8 \neq F_{14} \& F_8 \neq F_{15} \&$ 
6928   < $F_8 \neq F_{16} \&$ 

```

(224)


```

6929  F9 ≠ F10 & F9 ≠ F11 & F9 ≠ F12 & F9 ≠ F13 & F9 ≠ F14 & F9 ≠ F15 & F9 ≠ F16 &
6930  F10 ≠ F11 & F10 ≠ F12 & F10 ≠ F13 & F10 ≠ F14 & F10 ≠ F15 & F10 ≠ F16 &
6931  F11 ≠ F12 & F11 ≠ F13 & F11 ≠ F14 & F11 ≠ F15 & F11 ≠ F16 &
6932  F12 ≠ F13 & F12 ≠ F14 & F12 ≠ F15 & F12 ≠ F16 &
6933  F13 ≠ F14 & F13 ≠ F15 & F13 ≠ F16 &
6934  F14 ≠ F15 & F14 ≠ F16 &
6935  F15 ≠ F16)>
6936  proof -
6937    AOT_have Delta_pos: <Δφ → ◇φ> for φ
6938    proof(rule "→I")
6939      AOT_assume <Δφ>
6940      AOT_hence <□φ ∨ (¬Aφ & ◇φ)>
6941        using "≡dfE"[OF necessary_or_contingently_false] by blast
6942      moreover {
6943        AOT_assume <□φ>
6944        AOT_hence <◇φ>
6945        by (metis "B◇" "T◇" "vdash-properties:10")
6946      }
6947      moreover {
6948        AOT_assume <¬Aφ & ◇φ>
6949        AOT_hence <◇φ>
6950        using "&E" by blast
6951      }
6952      ultimately AOT_show <◇φ>
6953      by (metis "∨E"(2) "raa-cor:1")
6954    qed
6955
6956    AOT_have act_and_not_nec_not_delta: <¬Δφ> if <Aφ> and <¬□φ> for φ
6957      using "≡dfE" "&E"(1) "∨E"(2) necessary_or_contingently_false
6958        "raa-cor:3" that(1,2) by blast
6959    AOT_have act_and_pos_not_not_delta: <¬Δφ> if <Aφ> and <◇¬φ> for φ
6960      using "KBasic:11" act_and_not_nec_not_delta "≡E"(2) that(1,2) by blast
6961    AOT_have impossible_delta: <¬Δφ> if <¬◇φ> for φ
6962      using Delta_pos "modus-tollens:1" that by blast
6963    AOT_have not_act_and_pos_delta: <Δφ> if <¬Aφ> and <◇φ> for φ
6964      by (meson "≡dfI" "&I" "∨I"(2) necessary_or_contingently_false that(1,2))
6965    AOT_have nec_delta: <Δφ> if <□φ> for φ
6966      using "≡dfI" "∨I"(1) necessary_or_contingently_false that by blast
6967
6968    AOT_obtain a where a_prop: <A!a>
6969      using "A-objects"[axiom_inst] "∃E"[rotated] "&E" by blast
6970    AOT_obtain b where b_prop: <◇[E!]b & ¬A[E!]b>
6971      using "pos-not-pna:3" using "∃E"[rotated] by blast
6972
6973    AOT_have b_ord: <[0!]b>
6974    proof(rule "=dfI"(2)[OF AOT_ordinary])
6975      AOT_show <[λx ◇[E!]x]↓> by "cqt:2[lambda]"
6976    next
6977      AOT_show <[λx ◇[E!]x]b>
6978      proof (rule "β←C"(1); ("cqt:2[lambda]")?)
6979        AOT_show <b↓> by (rule "cqt:2[const_var]"[axiom_inst])
6980        AOT_show <◇[E!]b> by (fact b_prop[THEN "&E"(1)])
6981      qed
6982    qed
6983
6984    AOT_have nec_not_L_neg: <□¬[L]x> for x
6985      using "thm-noncont-e-e:2" "contingent-properties:2"[THEN "≡dfE"] "&E"
6986        CBF[THEN "→E"] "∨E" by blast
6987    AOT_have nec_L: <□[L]x> for x
6988      using "thm-noncont-e-e:1" "contingent-properties:1"[THEN "≡dfE"]
6989        CBF[THEN "→E"] "∨E" by blast
6990
6991    AOT_have act_ord_b: <A[0!]b>

```

```

6992   using b_ord "≡E"(1) "oa-facts:7" by blast
6993 AOT_have delta_ord_b: <Δ[0!]b>
6994   by (meson "≡dfI" b_ord "∀I"(1) necessary_or_contingently_false
6995       "oa-facts:1" "→E")
6996 AOT_have not_act_ord_a: <¬A[0!]a>
6997   by (meson a_prop "≡E"(1) "≡E"(3) "oa-contingent:3" "oa-facts:7")
6998 AOT_have not_delta_ord_a: <¬Δ[0!]a>
6999   by (metis Delta_pos "≡E"(4) not_act_ord_a "oa-facts:3" "oa-facts:7"
7000       "reductio-aa:1" "→E")
7001
7002 AOT_have not_act_abs_b: <¬A[A!]b>
7003   by (meson b_ord "≡E"(1) "≡E"(3) "oa-contingent:2" "oa-facts:8")
7004 AOT_have not_delta_abs_b: <¬Δ[A!]b>
7005 proof(rule "raa-cor:2")
7006   AOT_assume <Δ[A!]b>
7007   AOT_hence <◇[A!]b>
7008     by (metis Delta_pos "vdash-properties:10")
7009   AOT_thus <[A!]b & ¬[A!]b>
7010     by (metis b_ord "&I" "≡E"(1) "oa-contingent:2"
7011         "oa-facts:4" "→E")
7012 qed
7013 AOT_have act_abs_a: <A[A!]a>
7014   using a_prop "≡E"(1) "oa-facts:8" by blast
7015 AOT_have delta_abs_a: <Δ[A!]a>
7016   by (metis "≡dfI" a_prop "oa-facts:2" "→E" "∀I"(1)
7017       necessary_or_contingently_false)
7018
7019 AOT_have not_act_concrete_b: <¬A[E!]b>
7020   using b_prop "&E"(2) by blast
7021 AOT_have delta_concrete_b: <Δ[E!]b>
7022 proof (rule "≡dfI"[OF necessary_or_contingently_false];
7023        rule "∀I"(2); rule "&I")
7024   AOT_show <¬A[E!]b> using b_prop "&E"(2) by blast
7025 next
7026   AOT_show <◇[E!]b> using b_prop "&E"(1) by blast
7027 qed
7028 AOT_have not_act_concrete_a: <¬A[E!]a>
7029 proof (rule "raa-cor:2")
7030   AOT_assume <A[E!]a>
7031   AOT_hence 1: <◇[E!]a> by (metis "Act-Sub:3" "→E")
7032   AOT_have <[A!]a> by (simp add: a_prop)
7033   AOT_hence <[λx ¬◇[E!]x]a>
7034     by (rule "=dfE"(2)[OF AOT_abstract, rotated]) "cqt:2"
7035   AOT_hence <¬◇[E!]a> using "β→C"(1) by blast
7036   AOT_thus <◇[E!]a & ¬◇[E!]a> using 1 "&I" by blast
7037 qed
7038 AOT_have not_delta_concrete_a: <¬Δ[E!]a>
7039 proof (rule "raa-cor:2")
7040   AOT_assume <Δ[E!]a>
7041   AOT_hence 1: <◇[E!]a> by (metis Delta_pos "vdash-properties:10")
7042   AOT_have <[A!]a> by (simp add: a_prop)
7043   AOT_hence <[λx ¬◇[E!]x]a>
7044     by (rule "=dfE"(2)[OF AOT_abstract, rotated]) "cqt:2[lambda]"
7045   AOT_hence <¬◇[E!]a> using "β→C"(1) by blast
7046   AOT_thus <◇[E!]a & ¬◇[E!]a> using 1 "&I" by blast
7047 qed
7048
7049 AOT_have not_act_q_zero: <¬Aq0>
7050   by (meson "log-prop-prop:2" "pos-not-pna:1"
7051       q0_def "reductio-aa:1" "rule-id-df:2:a[zero]")
7052 AOT_have delta_q_zero: <Δq0>
7053 proof(rule "≡dfI"[OF necessary_or_contingently_false];
7054        rule "∀I"(2); rule "&I")

```

```

7055   AOT_show < $\neg \mathcal{A}q_0$ > using not_act_q_zero.
7056   AOT_show < $\diamond q_0$ > by (meson "&E"(1) q0_prop)
7057 qed
7058 AOT_have act_not_q_zero: < $\mathcal{A}\neg q_0$ >
7059   using "Act-Basic:1" "\VE"(2) not_act_q_zero by blast
7060 AOT_have not_delta_not_q_zero: < $\neg \Delta \neg q_0$ >
7061   using "\equivE" "conventions:5" "Act-Basic:1" act_and_not_nec_not_delta
7062     "&E"(1) "\VE"(2) not_act_q_zero q0_prop by blast
7063
7064 AOT_have < $[L^-] \downarrow$ > by (simp add: "rel-neg-T:3")
7065 moreover AOT_have < $\neg \mathcal{A}[L^-]b \ \& \ \neg \Delta[L^-]b \ \& \ \neg \mathcal{A}[L^-]a \ \& \ \neg \Delta[L^-]a$ >
7066 proof (safe intro!: "&I")
7067   AOT_show < $\neg \mathcal{A}[L^-]b$ >
7068     by (meson "\equivE"(1) "logic-actual-nec:1"[axiom_inst] "nec-imp-act"
7069         nec_not_L_neg "\rightarrowE")
7070   AOT_show < $\neg \Delta[L^-]b$ >
7071     by (meson Delta_pos "KBasic2:1" "\equivE"(1)
7072         "modus-tollens:1" nec_not_L_neg)
7073   AOT_show < $\neg \mathcal{A}[L^-]a$ >
7074     by (meson "\equivE"(1) "logic-actual-nec:1"[axiom_inst]
7075         "nec-imp-act" nec_not_L_neg "\rightarrowE")
7076   AOT_show < $\neg \Delta[L^-]a$ >
7077     using Delta_pos "KBasic2:1" "\equivE"(1) "modus-tollens:1"
7078       nec_not_L_neg by blast
7079 qed
7080 ultimately AOT_obtain F0 where < $\neg \mathcal{A}[F_0]b \ \& \ \neg \Delta[F_0]b \ \& \ \neg \mathcal{A}[F_0]a \ \& \ \neg \Delta[F_0]a$ >
7081   using "\existsI"(1)[rotated, THEN "\existsE"[rotated]] by fastforce
7082 AOT_hence < $\neg \mathcal{A}[F_0]b$ > and < $\neg \Delta[F_0]b$ > and < $\neg \mathcal{A}[F_0]a$ > and < $\neg \Delta[F_0]a$ >
7083   using "&E" by blast+
7084 note props = this
7085
7086 let ?II = "\ll[\lambda y [A!]y \ \& \ q_0]\gg"
7087 AOT_modally_strict {
7088   AOT_have < $\ll[\lambda y [A!]y \ \& \ q_0]\gg \downarrow$ > by "cqt:2[lambda]"
7089 } note 1 = this
7090 moreover AOT_have < $\neg \mathcal{A}[\ll[\lambda y [A!]y \ \& \ q_0]\gg]b \ \& \ \neg \Delta[\ll[\lambda y [A!]y \ \& \ q_0]\gg]b \ \& \ \neg \mathcal{A}[\ll[\lambda y [A!]y \ \& \ q_0]\gg]a \ \& \ \Delta[\ll[\lambda y [A!]y \ \& \ q_0]\gg]a$ >
7091 proof (safe intro!: "&I"; AOT_subst < $[\lambda y [A!]y \ \& \ q_0]x$ > < $A!x \ \& \ q_0$ > for: x)
7092   AOT_show < $\neg \mathcal{A}([A!]b \ \& \ q_0)$ >
7093     using "Act-Basic:2" "&E"(1) "\equivE"(1) not_act_abs_b "raa-cor:3" by blast
7094 next AOT_show < $\neg \Delta([A!]b \ \& \ q_0)$ >
7095     by (metis Delta_pos "KBasic2:3" "&E"(1) "\equivE"(4) not_act_abs_b
7096         "oa-facts:4" "oa-facts:8" "raa-cor:3" "\rightarrowE")
7097 next AOT_show < $\neg \mathcal{A}([A!]a \ \& \ q_0)$ >
7098     using "Act-Basic:2" "&E"(2) "\equivE"(1) not_act_q_zero
7099       "raa-cor:3" by blast
7100 next AOT_show < $\Delta([A!]a \ \& \ q_0)$ >
7101   proof (rule not_act_and_pos_delta)
7102     AOT_show < $\neg \mathcal{A}([A!]a \ \& \ q_0)$ >
7103       using "Act-Basic:2" "&E"(2) "\equivE"(4) not_act_q_zero
7104         "raa-cor:3" by blast
7105   next AOT_show < $\diamond([A!]a \ \& \ q_0)$ >
7106     by (metis "&I" "\rightarrowE" Delta_pos "KBasic:16" "&E"(1) delta_abs_a
7107         "\equivE"(1) "oa-facts:6" q0_prop)
7108   qed
7109 qed(auto simp: "beta-C-meta"[THEN "\rightarrowE", OF 1])
7110 ultimately AOT_obtain F1 where < $\neg \mathcal{A}[F_1]b \ \& \ \neg \Delta[F_1]b \ \& \ \neg \mathcal{A}[F_1]a \ \& \ \Delta[F_1]a$ >
7111   using "\existsI"(1)[rotated, THEN "\existsE"[rotated]] by fastforce
7112 AOT_hence < $\neg \mathcal{A}[F_1]b$ > and < $\neg \Delta[F_1]b$ > and < $\neg \mathcal{A}[F_1]a$ > and < $\Delta[F_1]a$ >
7113   using "&E" by blast+
7114 note props = props this
7115
7116 let ?II = "\ll[\lambda y [A!]y \ \& \ \neg q_0]\gg"
7117 AOT_modally_strict {

```

```

7118   AOT_have <[<<?II>>]↓> by "cqt:2[lambda]"
7119 } note 1 = this
7120 moreover AOT_have <¬ $\mathcal{A}$ [<<?II>>]b & ¬ $\Delta$ [<<?II>>]b &  $\mathcal{A}$ [<<?II>>]a & ¬ $\Delta$ [<<?II>>]a>
7121 proof (safe intro!: "&I"; AOT_subst <[ $\lambda$ y A!y & ¬q0]x> <A!x & ¬q0> for: x)
7122   AOT_show <¬ $\mathcal{A}$ [A!]b & ¬q0>
7123     using "Act-Basic:2" "&E"(1) "≡E"(1) not_act_abs_b "raa-cor:3" by blast
7124 next AOT_show <¬ $\Delta$ [A!]b & ¬q0>
7125   by (meson "RM◇" Delta_pos "Conjunction Simplification"(1) "≡E"(4)
7126     "modus-tollens:1" not_act_abs_b "oa-facts:4" "oa-facts:8")
7127 next AOT_show < $\mathcal{A}$ [A!]a & ¬q0>
7128   by (metis "Act-Basic:1" "Act-Basic:2" act_abs_a "&I" "∨E"(2)
7129     "≡E"(3) not_act_q_zero "raa-cor:3")
7130 next AOT_show <¬ $\Delta$ [A!]a & ¬q0>
7131 proof (rule act_and_not_nec_not_delta)
7132   AOT_show < $\mathcal{A}$ [A!]a & ¬q0>
7133     by (metis "Act-Basic:1" "Act-Basic:2" act_abs_a "&I" "∨E"(2)
7134       "≡E"(3) not_act_q_zero "raa-cor:3")
7135 next
7136   AOT_show <¬□([A!]a & ¬q0)>
7137     by (metis "KBasic2:1" "KBasic:3" "&E"(1) "&E"(2) "≡E"(4)
7138       q0_prop "raa-cor:3")
7139 qed
7140 qed(auto simp: "beta-C-meta"[THEN "→E", OF 1])
7141 ultimately AOT_obtain F2 where <¬ $\mathcal{A}$ [F2]b & ¬ $\Delta$ [F2]b &  $\mathcal{A}$ [F2]a & ¬ $\Delta$ [F2]a>
7142   using "∃I"(1)[rotated, THEN "∃E"[rotated]] by fastforce
7143 AOT_hence <¬ $\mathcal{A}$ [F2]b> and <¬ $\Delta$ [F2]b> and < $\mathcal{A}$ [F2]a> and <¬ $\Delta$ [F2]a>
7144   using "&E" by blast+
7145 note props = props this
7146
7147 AOT_have abstract_prop: <¬ $\mathcal{A}$ [A!]b & ¬ $\Delta$ [A!]b &  $\mathcal{A}$ [A!]a &  $\Delta$ [A!]a>
7148   using act_abs_a "&I" delta_abs_a not_act_abs_b not_delta_abs_b
7149   by presburger
7150 then AOT_obtain F3 where <¬ $\mathcal{A}$ [F3]b & ¬ $\Delta$ [F3]b &  $\mathcal{A}$ [F3]a &  $\Delta$ [F3]a>
7151   using "∃I"(1)[rotated, THEN "∃E"[rotated]] "oa-exist:2" by fastforce
7152 AOT_hence <¬ $\mathcal{A}$ [F3]b> and <¬ $\Delta$ [F3]b> and < $\mathcal{A}$ [F3]a> and < $\Delta$ [F3]a>
7153   using "&E" by blast+
7154 note props = props this
7155
7156 AOT_have <¬ $\mathcal{A}$ [E!]b &  $\Delta$ [E!]b & ¬ $\mathcal{A}$ [E!]a & ¬ $\Delta$ [E!]a>
7157   by (meson "&I" delta_concrete_b not_act_concrete_a
7158     not_act_concrete_b not_delta_concrete_a)
7159 then AOT_obtain F4 where <¬ $\mathcal{A}$ [F4]b &  $\Delta$ [F4]b & ¬ $\mathcal{A}$ [F4]a & ¬ $\Delta$ [F4]a>
7160   using "∃I"(1)[rotated, THEN "∃E"[rotated]]
7161   by fastforce
7162 AOT_hence <¬ $\mathcal{A}$ [F4]b> and < $\Delta$ [F4]b> and <¬ $\mathcal{A}$ [F4]a> and <¬ $\Delta$ [F4]a>
7163   using "&E" by blast+
7164 note props = props this
7165
7166 AOT_modally_strict {
7167   AOT_have <[ $\lambda$ y q0]↓> by "cqt:2[lambda]"
7168 } note 1 = this
7169 moreover AOT_have <¬ $\mathcal{A}$ [ $\lambda$ y q0]b &  $\Delta$ [ $\lambda$ y q0]b & ¬ $\mathcal{A}$ [ $\lambda$ y q0]a &  $\Delta$ [ $\lambda$ y q0]a>
7170   by (safe intro!: "&I"; AOT_subst <[ $\lambda$ y q0]b> <q0> for: b)
7171   (auto simp: not_act_q_zero delta_q_zero "beta-C-meta"[THEN "→E", OF 1])
7172 ultimately AOT_obtain F5 where <¬ $\mathcal{A}$ [F5]b &  $\Delta$ [F5]b & ¬ $\mathcal{A}$ [F5]a &  $\Delta$ [F5]a>
7173   using "∃I"(1)[rotated, THEN "∃E"[rotated]]
7174   by fastforce
7175 AOT_hence <¬ $\mathcal{A}$ [F5]b> and < $\Delta$ [F5]b> and <¬ $\mathcal{A}$ [F5]a> and < $\Delta$ [F5]a>
7176   using "&E" by blast+
7177 note props = props this
7178
7179 let ?II = "<<[ $\lambda$ y [E!]y ∨ ([A!]y & ¬q0)>>"
7180 AOT_modally_strict {

```

```

7181   AOT_have <[<?II>]↓> by "cqt:2[lambda]"
7182 } note 1 = this
7183 moreover AOT_have <¬A[<?II>]b & Δ[<?II>]b & A[<?II>]a & ¬Δ[<?II>]a>
7184 proof(safe intro!: "&I";
7185   AOT_subst <[λy E!y ∨ (A!y & ¬q₀)]x> <E!x ∨ (A!x & ¬q₀)> for: x)
7186   AOT_have <A¬([A!]b & ¬q₀)>
7187   by (metis "Act-Basic:1" "Act-Basic:2" abstract_prop "&E"(1) "VE"(2)
7188     "≡E"(1) "raa-cor:3")
7189 moreover AOT_have <¬A[E!]b>
7190   using b_prop "&E"(2) by blast
7191 ultimately AOT_have 2: <A¬([E!]b & ¬([A!]b & ¬q₀))>
7192   by (metis "Act-Basic:2" "Act-Sub:1" "&I" "≡E"(3) "raa-cor:1")
7193 AOT_have <A¬([E!]b ∨ ([A!]b & ¬q₀))>
7194   by (AOT_subst <¬([E!]b ∨ ([A!]b & ¬q₀))> <¬[E!]b & ¬([A!]b & ¬q₀)>)
7195   (auto simp: "oth-class-taut:5:d" 2)
7196 AOT_thus <¬A([E!]b ∨ ([A!]b & ¬q₀))>
7197   by (metis "¬I" "Act-Sub:1" "≡E"(4))
7198 next
7199 AOT_show <Δ([E!]b ∨ ([A!]b & ¬q₀))>
7200 proof (rule not_act_and_pos_delta)
7201   AOT_show <¬A([E!]b ∨ ([A!]b & ¬q₀))>
7202   by (metis "Act-Basic:2" "Act-Basic:9" "VE"(2) "raa-cor:3"
7203     "Conjunction Simplification"(1) "≡E"(4)
7204     "modus-tollens:1" not_act_abs_b not_act_concrete_b)
7205 next
7206 AOT_show <◇([E!]b ∨ ([A!]b & ¬q₀))>
7207   using "KBasic2:2" b_prop "&E"(1) "VI"(1) "≡E"(3) "raa-cor:3" by blast
7208 qed
7209 next AOT_show <A([E!]a ∨ ([A!]a & ¬q₀))>
7210   by (metis "Act-Basic:1" "Act-Basic:2" "Act-Basic:9" act_abs_a "&I"
7211     "VI"(2) "VE"(2) "≡E"(3) not_act_q_zero "raa-cor:1")
7212 next AOT_show <¬Δ([E!]a ∨ ([A!]a & ¬q₀))>
7213 proof (rule act_and_not_nec_not_delta)
7214   AOT_show <A([E!]a ∨ ([A!]a & ¬q₀))>
7215   by (metis "Act-Basic:1" "Act-Basic:2" "Act-Basic:9" act_abs_a "&I"
7216     "VI"(2) "VE"(2) "≡E"(3) not_act_q_zero "raa-cor:1")
7217 next
7218 AOT_have <□¬[E!]a>
7219   by (metis "≡afI" "conventions:5" "&I" "VI"(2)
7220     necessary_or_contingently_false
7221     not_act_concrete_a not_delta_concrete_a "raa-cor:3")
7222 moreover AOT_have <◇¬([A!]a & ¬q₀)>
7223   by (metis "KBasic2:1" "KBasic:11" "KBasic:3"
7224     "&E"(1,2) "≡E"(1) q₀_prop "raa-cor:3")
7225 ultimately AOT_have <◇(¬[E!]a & ¬([A!]a & ¬q₀))>
7226   by (metis "KBasic:16" "&I" "vdash-properties:10")
7227 AOT_hence <◇¬([E!]a ∨ ([A!]a & ¬q₀))>
7228   by (metis "RE◇" "≡E"(2) "oth-class-taut:5:d")
7229 AOT_thus <¬□([E!]a ∨ ([A!]a & ¬q₀))>
7230   by (metis "KBasic:12" "≡E"(1) "raa-cor:3")
7231 qed
7232 qed(auto simp: "beta-C-meta"[THEN "→E", OF 1])
7233 ultimately AOT_obtain F₆ where <¬A[F₆]b & Δ[F₆]b & A[F₆]a & ¬Δ[F₆]a>
7234   using "∃I"(1)[rotated, THEN "∃E"[rotated]] by fastforce
7235 AOT_hence <¬A[F₆]b> and <Δ[F₆]b> and <A[F₆]a> and <¬Δ[F₆]a>
7236   using "&E" by blast+
7237 note props = props this
7238
7239 let ?II = "«[λy [A!]y ∨ [E!]y»"
7240 AOT_modally_strict {
7241   AOT_have <[<?II>]↓> by "cqt:2[lambda]"
7242 } note 1 = this
7243 moreover AOT_have <¬A[<?II>]b & Δ[<?II>]b & A[<?II>]a & Δ[<?II>]a>

```

```

7244 proof(safe intro!: "&I"; AOT_subst <[ $\lambda y$  A!y  $\vee$  E!y]x> <A!x  $\vee$  E!x> for: x)
7245   AOT_show < $\neg \mathcal{A}([A!]b \vee [E!]b)$ >
7246     using "Act-Basic:9" " $\vee E$ "(2) " $\equiv E$ "(4) not_act_abs_b
7247     not_act_concrete_b "raa-cor:3" by blast
7248 next AOT_show < $\Delta([A!]b \vee [E!]b)$ >
7249   proof (rule not_act_and_pos_delta)
7250     AOT_show < $\neg \mathcal{A}([A!]b \vee [E!]b)$ >
7251       using "Act-Basic:9" " $\vee E$ "(2) " $\equiv E$ "(4) not_act_abs_b
7252       not_act_concrete_b "raa-cor:3" by blast
7253   next AOT_show < $\Diamond([A!]b \vee [E!]b)$ >
7254     using "KBasic2:2" b_prop "&E"(1) " $\vee I$ "(2) " $\equiv E$ "(2) by blast
7255   qed
7256 next AOT_show < $\mathcal{A}([A!]a \vee [E!]a)$ >
7257   by (meson "Act-Basic:9" act_abs_a " $\vee I$ "(1) " $\equiv E$ "(2))
7258 next AOT_show < $\Delta([A!]a \vee [E!]a)$ >
7259   proof (rule nec_delta)
7260     AOT_show < $\Box([A!]a \vee [E!]a)$ >
7261     by (metis "KBasic:15" act_abs_a act_and_not_nec_not_delta
7262         "Disjunction Addition"(1) delta_abs_a "raa-cor:3" " $\rightarrow E$ ")
7263   qed
7264 qed(auto simp: "beta-C-meta"[THEN " $\rightarrow E$ ", OF 1])
7265 ultimately AOT_obtain F7 where < $\neg \mathcal{A}[F_7]b$  &  $\Delta[F_7]b$  &  $\mathcal{A}[F_7]a$  &  $\Delta[F_7]a$ >
7266   using " $\exists I$ "(1)[rotated, THEN " $\exists E$ "[rotated]] by fastforce
7267 AOT_hence < $\neg \mathcal{A}[F_7]b$ > and < $\Delta[F_7]b$ > and < $\mathcal{A}[F_7]a$ > and < $\Delta[F_7]a$ >
7268   using "&E" by blast+
7269 note props = props this
7270
7271 let ?II = " $\llbracket \lambda y [O!]y \ \& \ \neg[E!]y \rrbracket$ "
7272 AOT_modally_strict {
7273   AOT_have < $\llbracket ?II \rrbracket \downarrow$ > by "cqt:2[lambda]"
7274 } note 1 = this
7275 moreover AOT_have < $\mathcal{A}[\llbracket ?II \rrbracket]b$  &  $\neg \Delta[\llbracket ?II \rrbracket]b$  &  $\neg \mathcal{A}[\llbracket ?II \rrbracket]a$  &  $\neg \Delta[\llbracket ?II \rrbracket]a$ >
7276 proof(safe intro!: "&I"; AOT_subst <[ $\lambda y$  O!y &  $\neg E!y$ ]x> <O!x &  $\neg E!x$ > for: x)
7277   AOT_show < $\mathcal{A}([O!]b \ \& \ \neg[E!]b)$ >
7278     by (metis "Act-Basic:1" "Act-Basic:2" act_ord_b "&I" " $\vee E$ "(2)
7279         " $\equiv E$ "(3) not_act_concrete_b "raa-cor:3")
7280 next AOT_show < $\neg \Delta([O!]b \ \& \ \neg[E!]b)$ >
7281   by (metis (no_types, opaque_lifting) "conventions:5" "Act-Sub:1" "RM:1"
7282       act_and_not_nec_not_delta "act-conj-act:3"
7283       act_ord_b b_prop "&I" "&E"(1) "Conjunction Simplification"(2)
7284       "df-rules-formulas[3]"
7285       " $\equiv E$ "(3) "raa-cor:1" " $\rightarrow E$ ")
7286 next AOT_show < $\neg \mathcal{A}([O!]a \ \& \ \neg[E!]a)$ >
7287   using "Act-Basic:2" "&E"(1) " $\equiv E$ "(1) not_act_ord_a "raa-cor:3" by blast
7288 next AOT_have < $\neg \Diamond([O!]a \ \& \ \neg[E!]a)$ >
7289   by (metis "KBasic2:3" "&E"(1) " $\equiv E$ "(4) not_act_ord_a "oa-facts:3"
7290       "oa-facts:7" "raa-cor:3" "vdash-properties:10")
7291   AOT_thus < $\neg \Delta([O!]a \ \& \ \neg[E!]a)$ >
7292     by (rule impossible_delta)
7293 qed(auto simp: "beta-C-meta"[THEN " $\rightarrow E$ ", OF 1])
7294 ultimately AOT_obtain F8 where < $\mathcal{A}[F_8]b$  &  $\neg \Delta[F_8]b$  &  $\neg \mathcal{A}[F_8]a$  &  $\neg \Delta[F_8]a$ >
7295   using " $\exists I$ "(1)[rotated, THEN " $\exists E$ "[rotated]] by fastforce
7296 AOT_hence < $\mathcal{A}[F_8]b$ > and < $\neg \Delta[F_8]b$ > and < $\neg \mathcal{A}[F_8]a$ > and < $\neg \Delta[F_8]a$ >
7297   using "&E" by blast+
7298 note props = props this
7299
7300 let ?II = " $\llbracket \lambda y \neg[E!]y \ \& \ ([O!]y \vee q_0) \rrbracket$ "
7301 AOT_modally_strict {
7302   AOT_have < $\llbracket ?II \rrbracket \downarrow$ > by "cqt:2[lambda]"
7303 } note 1 = this
7304 moreover AOT_have < $\mathcal{A}[\llbracket ?II \rrbracket]b$  &  $\neg \Delta[\llbracket ?II \rrbracket]b$  &  $\neg \mathcal{A}[\llbracket ?II \rrbracket]a$  &  $\Delta[\llbracket ?II \rrbracket]a$ >
7305 proof(safe intro!: "&I";
7306   AOT_subst <[ $\lambda y \neg E!y \ \& \ (O!y \vee q_0)$ ]x> < $\neg E!x \ \& \ (O!x \vee q_0)$ > for: x)

```

```

7307   AOT_show <mathcal{A}(\neg[E!]b \ \& \ ([O!]b \ \vee \ q_0))>
7308     by (metis "Act-Basic:1" "Act-Basic:2" "Act-Basic:9" act_ord_b "&I"
7309         "\veeI"(1) "\veeE"(2) "\equivE"(3) not_act_concrete_b "raa-cor:1")
7310 next AOT_show <mathcal{A}(\neg[E!]b \ \& \ ([O!]b \ \vee \ q_0))>
7311   proof (rule act_and_pos_not_not_delta)
7312     AOT_show <mathcal{A}(\neg[E!]b \ \& \ ([O!]b \ \vee \ q_0))>
7313       by (metis "Act-Basic:1" "Act-Basic:2" "Act-Basic:9" act_ord_b "&I"
7314           "\veeI"(1) "\veeE"(2) "\equivE"(3) not_act_concrete_b "raa-cor:1")
7315   next
7316     AOT_show <mathcal{A}(\neg[E!]b \ \& \ ([O!]b \ \vee \ q_0))>
7317     proof (AOT_subst <mathcal{A}(\neg[E!]b \ \& \ ([O!]b \ \vee \ q_0))> <mathcal{A}([E!]b \ \vee \ \neg([O!]b \ \vee \ q_0))>)
7318       AOT_modally_strict {
7319         AOT_show <mathcal{A}(\neg[E!]b \ \& \ ([O!]b \ \vee \ q_0)) \equiv \mathcal{A}([E!]b \ \vee \ \neg([O!]b \ \vee \ q_0))>
7320         by (metis "&I" "&E"(1,2) "\veeI"(1,2) "\veeE"(2)
7321             "\rightarrowI" "\equivI" "reductio-aa:1")
7322       }
7323     next
7324       AOT_show <mathcal{A}([E!]b \ \vee \ \neg([O!]b \ \vee \ q_0))>
7325       using "KBasic2:2" b_prop "&E"(1) "\veeI"(1) "\equivE"(3)
7326           "raa-cor:3" by blast
7327     qed
7328   qed
7329 next
7330 AOT_show <mathcal{A}(\neg[E!]a \ \& \ ([O!]a \ \vee \ q_0))>
7331   using "Act-Basic:2" "Act-Basic:9" "&E"(2) "\veeE"(3) "\equivE"(1)
7332       not_act_ord_a not_act_q_zero "reductio-aa:2" by blast
7333 next
7334 AOT_show <mathcal{A}(\neg[E!]a \ \& \ ([O!]a \ \vee \ q_0))>
7335   proof (rule not_act_and_pos_delta)
7336     AOT_show <mathcal{A}(\neg[E!]a \ \& \ ([O!]a \ \vee \ q_0))>
7337       by (metis "Act-Basic:2" "Act-Basic:9" "&E"(2) "\veeE"(3) "\equivE"(1)
7338           not_act_ord_a not_act_q_zero "reductio-aa:2")
7339   next
7340     AOT_have <mathcal{A}(\neg[E!]a)>
7341       using "KBasic2:1" "\equivE"(2) not_act_and_pos_delta not_act_concrete_a
7342           not_delta_concrete_a "raa-cor:5" by blast
7343     moreover AOT_have <mathcal{A}([O!]a \ \vee \ q_0)>
7344       by (metis "KBasic2:2" "&E"(1) "\veeI"(2) "\equivE"(3) q_0_prop "raa-cor:3")
7345     ultimately AOT_show <mathcal{A}(\neg[E!]a \ \& \ ([O!]a \ \vee \ q_0))>
7346       by (metis "KBasic:16" "&I" "vdash-properties:10")
7347     qed
7348     qed(auto simp: "beta-C-meta"[THEN "\rightarrowE", OF 1])
7349     ultimately AOT_obtain F9 where <mathcal{A}[F9]b \ \& \ \neg\mathcal{A}[F9]b \ \& \ \neg\mathcal{A}[F9]a \ \& \ \mathcal{A}[F9]a>
7350     using "\existsI"(1)[rotated, THEN "\existsE"[rotated]] by fastforce
7351     AOT_hence <mathcal{A}[F9]b> and <\neg\mathcal{A}[F9]b> and <\neg\mathcal{A}[F9]a> and <\mathcal{A}[F9]a>
7352     using "&E" by blast+
7353     note props = props this
7354
7355     AOT_modally_strict {
7356       AOT_have <mathcal{A}(\lambda y. \neg q_0)> by "cqt:2[lambda]"
7357     } note 1 = this
7358     moreover AOT_have <mathcal{A}(\lambda y. \neg q_0)b \ \& \ \neg\mathcal{A}(\lambda y. \neg q_0)b \ \& \ \mathcal{A}(\lambda y. \neg q_0)a \ \& \ \neg\mathcal{A}(\lambda y. \neg q_0)a>
7359     by (safe intro!: "&I"; AOT_subst <mathcal{A}(\lambda y. \neg q_0)x> <\neg q_0> for: x)
7360     (auto simp: act_not_q_zero not_delta_not_q_zero
7361         "beta-C-meta"[THEN "\rightarrowE", OF 1])
7362     ultimately AOT_obtain F10 where <mathcal{A}[F10]b \ \& \ \neg\mathcal{A}[F10]b \ \& \ \mathcal{A}[F10]a \ \& \ \neg\mathcal{A}[F10]a>
7363     using "\existsI"(1)[rotated, THEN "\existsE"[rotated]] by fastforce
7364     AOT_hence <mathcal{A}[F10]b> and <\neg\mathcal{A}[F10]b> and <\mathcal{A}[F10]a> and <\neg\mathcal{A}[F10]a>
7365     using "&E" by blast+
7366     note props = props this
7367
7368     AOT_modally_strict {
7369       AOT_have <mathcal{A}(\lambda y. \neg[E!]y)> by "cqt:2[lambda]"

```



```

7370 } note 1 = this
7371 moreover AOT_have <mathcal{A}[\lambda y \neg[E!]y]b & \neg\Delta[\lambda y \neg[E!]y]b &
7372           mathcal{A}[\lambda y \neg[E!]y]a & \Delta[\lambda y \neg[E!]y]a>
7373 proof (safe intro!: "&I"; AOT_subst <mathcal{A}[\lambda y \neg[E!]y]x> <\neg[E!]x> for: x)
7374   AOT_show <mathcal{A}\neg[E!]b>
7375     using "Act-Basic:1" "\VE"(2) not_act_concrete_b by blast
7376 next AOT_show <\neg\Delta\neg[E!]b>
7377   using "\equiv_{df}E" "conventions:5" "Act-Basic:1" act_and_not_nec_not_delta
7378     b_prop "&E"(1) "\VE"(2) not_act_concrete_b by blast
7379 next AOT_show <mathcal{A}\neg[E!]a>
7380   using "Act-Basic:1" "\VE"(2) not_act_concrete_a by blast
7381 next AOT_show <\Delta\neg[E!]a>
7382   using "KBasic2:1" "\equiv"(2) nec_delta not_act_and_pos_delta
7383     not_act_concrete_a not_delta_concrete_a "reductio-aa:1"
7384   by blast
7385 qed(auto simp: "beta-C-meta"[THEN "\toE", OF 1])
7386 ultimately AOT_obtain F11 where <mathcal{A}[F11]b & \neg\Delta[F11]b & mathcal{A}[F11]a & \Delta[F11]a>
7387   using "\existsI"(1)[rotated, THEN "\existsE"[rotated]] by fastforce
7388 AOT_hence <mathcal{A}[F11]b> and <\neg\Delta[F11]b> and <mathcal{A}[F11]a> and <\Delta[F11]a>
7389   using "&E" by blast+
7390 note props = props this
7391
7392 AOT_have <mathcal{A}[0!]b & \Delta[0!]b & \neg\mathcal{A}[0!]a & \neg\Delta[0!]a>
7393   by (simp add: act_ord_b "&I" delta_ord_b not_act_ord_a not_delta_ord_a)
7394 then AOT_obtain F12 where <mathcal{A}[F12]b & \Delta[F12]b & \neg\mathcal{A}[F12]a & \neg\Delta[F12]a>
7395   using "oa-exist:1" "\existsI"(1)[rotated, THEN "\existsE"[rotated]] by fastforce
7396 AOT_hence <mathcal{A}[F12]b> and <\Delta[F12]b> and <\neg\mathcal{A}[F12]a> and <\neg\Delta[F12]a>
7397   using "&E" by blast+
7398 note props = props this
7399
7400 let ?II = "<<mathcal{A}[\lambda y [0!]y \vee q_0]>>"
7401 AOT_modally_strict {
7402   AOT_have <mathcal{A}[\llbracket?II\rrbracket]b & \Delta[\llbracket?II\rrbracket]b & \neg\mathcal{A}[\llbracket?II\rrbracket]a & \Delta[\llbracket?II\rrbracket]a>
7403 } note 1 = this
7404 moreover AOT_have <mathcal{A}[\llbracket?II\rrbracket]b & \Delta[\llbracket?II\rrbracket]b & \neg\mathcal{A}[\llbracket?II\rrbracket]a & \Delta[\llbracket?II\rrbracket]a>
7405 proof (safe intro!: "&I"; AOT_subst <mathcal{A}[\llbracket?II\rrbracket]x > <[0!]x \vee q_0> for: x)
7406   AOT_show <mathcal{A}[\llbracket?II\rrbracket]b >
7407     by (meson "Act-Basic:9" act_ord_b "\VI"(1) "\equiv"(2))
7408 next AOT_show <\Delta[\llbracket?II\rrbracket]b >
7409   by (meson "KBasic:15" b_ord "\VI"(1) nec_delta "oa-facts:1" "\toE")
7410 next AOT_show <\neg\mathcal{A}[\llbracket?II\rrbracket]a >
7411   using "Act-Basic:9" "\VE"(2) "\equiv"(4) not_act_ord_a
7412     not_act_q_zero "raa-cor:3" by blast
7413 next AOT_show <\Delta[\llbracket?II\rrbracket]a >
7414   proof (rule not_act_and_pos_delta)
7415     AOT_show <\neg\mathcal{A}[\llbracket?II\rrbracket]a >
7416       using "Act-Basic:9" "\VE"(2) "\equiv"(4) not_act_ord_a
7417         not_act_q_zero "raa-cor:3" by blast
7418     next AOT_show <\Diamond[\llbracket?II\rrbracket]a >
7419       using "KBasic2:2" "&E"(1) "\VI"(2) "\equiv"(2) q_0_prop by blast
7420   qed
7421 qed(auto simp: "beta-C-meta"[THEN "\toE", OF 1])
7422 ultimately AOT_obtain F13 where <mathcal{A}[F13]b & \Delta[F13]b & \neg\mathcal{A}[F13]a & \Delta[F13]a>
7423   using "\existsI"(1)[rotated, THEN "\existsE"[rotated]] by fastforce
7424 AOT_hence <mathcal{A}[F13]b> and <\Delta[F13]b> and <\neg\mathcal{A}[F13]a> and <\Delta[F13]a>
7425   using "&E" by blast+
7426 note props = props this
7427
7428 let ?II = "<<mathcal{A}[\lambda y [0!]y \vee \neg q_0]>>"
7429 AOT_modally_strict {
7430   AOT_have <mathcal{A}[\llbracket?II\rrbracket]b & \Delta[\llbracket?II\rrbracket]b & \mathcal{A}[\llbracket?II\rrbracket]a & \neg\Delta[\llbracket?II\rrbracket]a>
7431 } note 1 = this
7432 moreover AOT_have <mathcal{A}[\llbracket?II\rrbracket]b & \Delta[\llbracket?II\rrbracket]b & \mathcal{A}[\llbracket?II\rrbracket]a & \neg\Delta[\llbracket?II\rrbracket]a>

```



```

7433 proof (safe intro!: "&I"; AOT_subst <[ $\lambda y$  0!y  $\vee \neg q_0$ ]x> <0!x  $\vee \neg q_0$ > for: x)
7434   AOT_show < $\mathcal{A}([0!]b \vee \neg q_0)$ >
7435   by (meson "Act-Basic:9" act_not_q_zero " $\vee I$ "(2) " $\equiv E$ "(2))
7436 next AOT_show < $\Delta([0!]b \vee \neg q_0)$ >
7437   by (meson "KBasic:15" b_ord " $\vee I$ "(1) nec_delta "oa-facts:1" " $\rightarrow E$ ")
7438 next AOT_show < $\mathcal{A}([0!]a \vee \neg q_0)$ >
7439   by (meson "Act-Basic:9" act_not_q_zero " $\vee I$ "(2) " $\equiv E$ "(2))
7440 next AOT_show < $\neg \Delta([0!]a \vee \neg q_0)$ >
7441   proof (rule act_and_pos_not_not_delta)
7442     AOT_show < $\mathcal{A}([0!]a \vee \neg q_0)$ >
7443     by (meson "Act-Basic:9" act_not_q_zero " $\vee I$ "(2) " $\equiv E$ "(2))
7444   next
7445     AOT_have < $\Box \neg [0!]a$ >
7446     using "KBasic2:1" " $\equiv E$ "(2) not_act_and_pos_delta
7447     not_act_ord_a not_delta_ord_a "raa-cor:6" by blast
7448   moreover AOT_have < $\Diamond q_0$ >
7449     by (meson "&E"(1) q_0_prop)
7450   ultimately AOT_have 2: < $\Diamond(\neg [0!]a \ \& \ q_0)$ >
7451     by (metis "KBasic:16" "&I" "vdash-properties:10")
7452   AOT_show < $\Diamond \neg([0!]a \vee \neg q_0)$ >
7453   proof (AOT_subst (reverse) < $\neg([0!]a \vee \neg q_0)$ > < $\neg [0!]a \ \& \ q_0$ >)
7454     AOT_modally_strict {
7455       AOT_show < $\neg [0!]a \ \& \ q_0 \equiv \neg([0!]a \vee \neg q_0)$ >
7456       by (metis "&I" "&E"(1) "&E"(2) " $\vee I$ "(1) " $\vee I$ "(2)
7457           " $\vee E$ "(3) "deduction-theorem" " $\equiv I$ " "raa-cor:3")
7458     }
7459   next
7460     AOT_show < $\Diamond(\neg [0!]a \ \& \ q_0)$ >
7461     using "2" by blast
7462   qed
7463   qed
7464   qed(auto simp: "beta-C-meta"[THEN " $\rightarrow E$ ", OF 1])
7465   ultimately AOT_obtain F14 where < $\mathcal{A}[F_{14}]b \ \& \ \Delta[F_{14}]b \ \& \ \mathcal{A}[F_{14}]a \ \& \ \neg \Delta[F_{14}]a$ >
7466   using " $\exists I$ "(1)[rotated, THEN " $\exists E$ "[rotated]] by fastforce
7467   AOT_hence < $\mathcal{A}[F_{14}]b$ > and < $\Delta[F_{14}]b$ > and < $\mathcal{A}[F_{14}]a$ > and < $\neg \Delta[F_{14}]a$ >
7468   using "&E" by blast+
7469   note props = props this
7470
7471   AOT_have <[L] $\downarrow$ >
7472   by (rule "=def I"(2)[OF L_def]) "cqt:2[lambda]" +
7473   moreover AOT_have < $\mathcal{A}[L]b \ \& \ \Delta[L]b \ \& \ \mathcal{A}[L]a \ \& \ \Delta[L]a$ >
7474   proof (safe intro!: "&I")
7475     AOT_show < $\mathcal{A}[L]b$ >
7476     by (meson nec_L "nec-imp-act" "vdash-properties:10")
7477   next AOT_show < $\Delta[L]b$ > using nec_L nec_delta by blast
7478   next AOT_show < $\mathcal{A}[L]a$ > by (meson nec_L "nec-imp-act" " $\rightarrow E$ ")
7479   next AOT_show < $\Delta[L]a$ > using nec_L nec_delta by blast
7480   qed
7481   ultimately AOT_obtain F15 where < $\mathcal{A}[F_{15}]b \ \& \ \Delta[F_{15}]b \ \& \ \mathcal{A}[F_{15}]a \ \& \ \Delta[F_{15}]a$ >
7482   using " $\exists I$ "(1)[rotated, THEN " $\exists E$ "[rotated]] by fastforce
7483   AOT_hence < $\mathcal{A}[F_{15}]b$ > and < $\Delta[F_{15}]b$ > and < $\mathcal{A}[F_{15}]a$ > and < $\Delta[F_{15}]a$ >
7484   using "&E" by blast+
7485   note props = props this
7486
7487   show ?thesis
7488   by (rule " $\exists I$ "(2)[where  $\beta=F_0$ ]; rule " $\exists I$ "(2)[where  $\beta=F_1$ ];
7489       rule " $\exists I$ "(2)[where  $\beta=F_2$ ]; rule " $\exists I$ "(2)[where  $\beta=F_3$ ];
7490       rule " $\exists I$ "(2)[where  $\beta=F_4$ ]; rule " $\exists I$ "(2)[where  $\beta=F_5$ ];
7491       rule " $\exists I$ "(2)[where  $\beta=F_6$ ]; rule " $\exists I$ "(2)[where  $\beta=F_7$ ];
7492       rule " $\exists I$ "(2)[where  $\beta=F_8$ ]; rule " $\exists I$ "(2)[where  $\beta=F_9$ ];
7493       rule " $\exists I$ "(2)[where  $\beta=F_{10}$ ]; rule " $\exists I$ "(2)[where  $\beta=F_{11}$ ];
7494       rule " $\exists I$ "(2)[where  $\beta=F_{12}$ ]; rule " $\exists I$ "(2)[where  $\beta=F_{13}$ ];
7495       rule " $\exists I$ "(2)[where  $\beta=F_{14}$ ]; rule " $\exists I$ "(2)[where  $\beta=F_{15}$ ];

```

```

7496     safe intro!: "&I")
7497 (match conclusion in "[?v | = [F] ≠ [G]]" for F G ⇒ <
7498   match props in A: "[?v | = ¬φ{F}]" for φ ⇒ <
7499     match (φ) in "λa . ?p" ⇒ <fail> | "λa . a" ⇒ <fail> | _ ⇒ <
7500     match props in B: "[?v | = φ{G}]" ⇒ <
7501     fact "pos-not-equiv-ne:4"[where F=F and G=G and φ=φ, THEN "→E",
7502           OF "oth-class-taut:4:h"[THEN "≡E"(2)],
7503           OF "Disjunction Addition"(2)[THEN "→E"],
7504           OF "&I", OF A, OF B]>>>>)+
7505 qed
7506
7507 subsection<The Theory of Objects>
7508 text<\label{PLM: 9.11}>
7509
7510 AOT_theorem "o-objects-exist:1": <□∃x O!x> (225.1)
7511 proof(rule RN)
7512   AOT_modally_strict {
7513     AOT_obtain a where <◇(E!a & ¬A[E!]a)>
7514     using "∃E"[rotated, OF "qml:4"[axiom_inst, THEN "BF◇"[THEN "→E"]]]
7515     by blast
7516     AOT_hence 1: <◇E!a> by (metis "KBasic2:3" "&E"(1) "→E")
7517     AOT_have <[λx ◇[E!]x]a>
7518     proof (rule "β←C"(1); "cqt:2[lambda]"?)
7519       AOT_show <a↓> using "cqt:2[const_var]"[axiom_inst] by blast
7520     next
7521       AOT_show <◇E!a> by (fact 1)
7522     qed
7523     AOT_hence <O!a> by (rule "=dfI"(2)[OF AOT_ordinary, rotated]) "cqt:2"
7524     AOT_thus <∃x [O!]x> by (rule "∃I")
7525   }
7526 qed
7527
7528 AOT_theorem "o-objects-exist:2": <□∃x A!x> (225.2)
7529 proof (rule RN)
7530   AOT_modally_strict {
7531     AOT_obtain a where <[A!]a>
7532     using "A-objects"[axiom_inst] "∃E"[rotated] "&E" by blast
7533     AOT_thus <∃x A!x> using "∃I" by blast
7534   }
7535 qed
7536
7537 AOT_theorem "o-objects-exist:3": <□¬∀x O!x> (225.3)
7538   by (rule RN)
7539     (metis (no_types, opaque_lifting) "∃E" "cqt-orig:1[const_var]"
7540       "≡E"(4) "modus-tollens:1" "o-objects-exist:2" "oa-contingent:2"
7541       "qml:2"[axiom_inst] "reductio-aa:2")
7542
7543 AOT_theorem "o-objects-exist:4": <□¬∀x A!x> (225.4)
7544   by (rule RN)
7545     (metis (mono_tags, opaque_lifting) "∃E" "cqt-orig:1[const_var]"
7546       "≡E"(1) "modus-tollens:1" "o-objects-exist:1" "oa-contingent:2"
7547       "qml:2"[axiom_inst] "→E")
7548
7549 AOT_theorem "o-objects-exist:5": <□¬∀x E!x> (225.5)
7550 proof (rule RN; rule "raa-cor:2")
7551   AOT_modally_strict {
7552     AOT_assume <∀x E!x>
7553     moreover AOT_obtain a where abs: <A!a>
7554     using "o-objects-exist:2"[THEN "qml:2"[axiom_inst, THEN "→E"]]]
7555     "∃E"[rotated] by blast
7556     ultimately AOT_have <E!a> using "∀E" by blast
7557     AOT_hence 1: <◇E!a> by (metis "T◇" "→E")
7558     AOT_have <[λy ◇E!y]a>

```

```

7559   proof (rule " $\beta \leftarrow C$ "(1); "cqt:2[lambda]"?)
7560     AOT_show <a> using "cqt:2[const_var]"[axiom_inst].
7561   next
7562     AOT_show < $\langle \Diamond E!a \rangle$ > by (fact 1)
7563   qed
7564   AOT_hence <O!a>
7565     by (rule " $=_{df} I$ "(2)[OF AOT_ordinary, rotated]) "cqt:2[lambda]"
7566   AOT_hence < $\neg A!a$ > by (metis " $\equiv E$ "(1) "oa-contingent:2")
7567   AOT_thus <p &  $\neg p$ > for p using abs by (metis "raa-cor:3")
7568 }
7569 qed
7570
7571 AOT_theorem partition: < $\neg \exists x (O!x \ \& \ A!x)$ > (226)
7572 proof(rule "raa-cor:2")
7573   AOT_assume < $\exists x (O!x \ \& \ A!x)$ >
7574   then AOT_obtain a where <O!a & A!a>
7575     using " $\exists E$ "[rotated] by blast
7576   AOT_thus <p &  $\neg p$ > for p
7577     by (metis "&E"(1) "Conjunction Simplification"(2) " $\equiv E$ "(1)
7578         "modus-tollens:1" "oa-contingent:2" "raa-cor:3")
7579   qed
7580
7581 AOT_define eq_E :: < $\Pi$ > ("'( $=_E$ )")
7582   " $=_E$ ": <( $=_E$ )  $=_{df} [\lambda xy \ O!x \ \& \ O!y \ \& \ \Box \forall F ([F]x \ \equiv \ [F]y)]$ > (227)
7583
7584 syntax "_AOT_eq_E_infix" :: < $\tau \Rightarrow \tau \Rightarrow \varphi$ > (infixl " $=_E$ " 50)
7585 translations
7586   "_AOT_eq_E_infix  $\kappa \ \kappa'$ " == "CONST AOT_exe (CONST eq_E) (CONST Pair  $\kappa \ \kappa'$ )"
7587 print_translation<
7588 AOT_syntax_print_translations
7589 [(const_syntax<AOT_exe>, fn ctxt => fn [
7590   Const (const_name<eq_E>, _),
7591   Const (const_syntax<Pair>, _) $ lhs $ rhs
7592 ] => Const (syntax_const<_AOT_eq_E_infix>, dummyT) $ lhs $ rhs)]>
7593
7594 text<Note: Not explicitly mentioned as theorem in PLM.>
7595 AOT_theorem " $=_E$ [denotes]": <[[ $=_E$ ]] $\downarrow$ > (227)
7596   by (rule " $=_{df} I$ "(2)[OF " $=_E$ "]) "cqt:2[lambda]"
7597
7598 AOT_theorem " $=_E$ -simple:1": < $x \ =_E \ y \ \equiv \ (O!x \ \& \ O!y \ \& \ \Box \forall F ([F]x \ \equiv \ [F]y))$ > (230.1)
7599 proof -
7600   AOT_have 1: <[[ $\lambda xy \ [O!]x \ \& \ [O!]y \ \& \ \Box \forall F ([F]x \ \equiv \ [F]y)]$ ] $\downarrow$ > by "cqt:2"
7601   show ?thesis
7602     apply (rule " $=_{df} I$ "(2)[OF " $=_E$ "]; "cqt:2[lambda]"?)
7603     using "beta-C-meta"[THEN " $\rightarrow E$ ", OF 1, unvarify  $\nu_1 \nu_n$ , of "(_,_)",
7604           OF tuple_denotes[THEN " $\equiv_{df} I$ ", OF "&I",
7605           OF "cqt:2[const_var]"[axiom_inst],
7606           OF "cqt:2[const_var]"[axiom_inst]]
7607     by fast
7608   qed
7609
7610 AOT_theorem " $=_E$ -simple:2": < $x \ =_E \ y \ \rightarrow \ x \ = \ y$ > (230.2)
7611 proof (rule " $\rightarrow I$ ")
7612   AOT_assume < $x \ =_E \ y$ >
7613   AOT_hence <O!x & O!y &  $\Box \forall F ([F]x \ \equiv \ [F]y)$ >
7614     using " $=_E$ -simple:1"[THEN " $\equiv E$ "(1)] by blast
7615   AOT_thus < $x \ = \ y$ >
7616     using " $\equiv_{df} I$ "[OF "identity:1"] " $\forall I$ " by blast
7617   qed
7618
7619 AOT_theorem "id-nec3:1": < $x \ =_E \ y \ \equiv \ \Box(x \ =_E \ y)$ > (231.1)
7620 proof (rule " $\equiv I$ "; rule " $\rightarrow I$ ")
7621   AOT_assume < $x \ =_E \ y$ >

```

```

7622 AOT_hence <O!x & O!y &  $\square\forall F$  ([F]x  $\equiv$  [F]y)>
7623   using "E-simple:1" "E" by blast
7624 AOT_hence < $\square$ O!x &  $\square$ O!y &  $\square\square\forall F$  ([F]x  $\equiv$  [F]y)>
7625   by (metis "S5Basic:6" "&I" "&E"(1) "&E"(2) "E"(4)
7626       "oa-facts:1" "raa-cor:3" "vdash-properties:10")
7627 AOT_hence < $\square$ (O!x & O!y &  $\square\forall F$  ([F]x  $\equiv$  [F]y))>
7628   by (metis "&E"(1) "&E"(2) "E"(2) "KBasic:3" "&I")
7629 AOT_thus < $\square$ (x =E y)>
7630   using "E-simple:1"
7631   by (AOT_subst <x =E y> <O!x & O!y &  $\square\forall F$  ([F]x  $\equiv$  [F]y)>) auto
7632 next
7633   AOT_assume < $\square$ (x =E y)>
7634   AOT_thus <x =E y> using "qml:2"[axiom_inst, THEN " $\rightarrow$ E"] by blast
7635 qed
7636
7637 AOT_theorem "id-nec3:2": < $\diamond$ (x =E y)  $\equiv$  x =E y> (231.2)
7638   by (meson "RE $\diamond$ " "S5Basic:2" "id-nec3:1" "E"(1,5) "Commutativity of  $\equiv$ ")
7639
7640 AOT_theorem "id-nec3:3": < $\diamond$ (x =E y)  $\equiv$   $\square$ (x =E y)> (231.3)
7641   by (meson "id-nec3:1" "id-nec3:2" "E"(5))
7642
7643 syntax "_AOT_non_eq_E" :: <II> ("'( $\neq_E$ )'")
7644 translations
7645   (II) "(<math>\neq_E</math>)" == (II) "(=E)~"
7646 syntax "_AOT_non_eq_E_infix" :: < $\tau \Rightarrow \tau \Rightarrow \varphi$ > (infix1 "<math>\neq_E</math>" 50)
7647 translations
7648   "_AOT_non_eq_E_infix  $\kappa \kappa$ " ==
7649   "CONST AOT_exe (CONST relation_negation (CONST eq_E)) (CONST Pair  $\kappa \kappa$ )"
7650 print_translation<
7651 AOT_syntax_print_translations
7652 [(const_syntax<AOT_exe>, fn ctxt => fn [
7653   Const (const_syntax<relation_negation>, _) $ Const (const_name<eq_E>, _),
7654   Const (const_syntax<Pair>, _) $ lhs $ rhs
7655 ] => Const (syntax_const<_AOT_non_eq_E_infix>, dummyT) $ lhs $ rhs)]>
7656 AOT_theorem "thm-neg=E": <x  $\neq_E$  y  $\equiv$   $\neg$ (x =E y)> (233)
7657 proof -
7658   AOT_have  $\vartheta$ : <[ $\lambda x_1 \dots x_2 \neg (=E) x_1 \dots x_2$ ]  $\downarrow$ > by "cqt:2"
7659   AOT_have <x  $\neq_E$  y  $\equiv$  [ $\lambda x_1 \dots x_2 \neg (=E) x_1 \dots x_2$ ]xy>
7660     by (rule "=dfI"(1)[OF "df-relation-negation", OF  $\vartheta$ ])
7661     (meson "oth-class-taut:3:a")
7662   also AOT_have <...  $\equiv$   $\neg$ (=E)xy>
7663     by (safe intro!: "beta-C-meta"[THEN " $\rightarrow$ E", unvarify  $\nu_1 \nu_n$ ] "cqt:2"
7664         tuple_denotes[THEN "=dfI"] "&I")
7665   finally show ?thesis.
7666 qed
7667
7668 AOT_theorem "id-nec4:1": <x  $\neq_E$  y  $\equiv$   $\square$ (x  $\neq_E$  y)> (234.1)
7669 proof -
7670   AOT_have <x  $\neq_E$  y  $\equiv$   $\neg$ (x =E y)> using "thm-neg=E".
7671   also AOT_have <...  $\equiv$   $\neg\diamond$ (x =E y)>
7672     by (meson "id-nec3:2" "E"(1) "Commutativity of  $\equiv$ " "oth-class-taut:4:b")
7673   also AOT_have <...  $\equiv$   $\square\neg$ (x =E y)>
7674     by (meson "KBasic:2:1" "E"(2) "Commutativity of  $\equiv$ ")
7675   also AOT_have <...  $\equiv$   $\square$ (x  $\neq_E$  y)>
7676     by (AOT_subst (reverse) < $\neg$ (x =E y)> <x  $\neq_E$  y>)
7677     (auto simp: "thm-neg=E" "oth-class-taut:3:a")
7678   finally show ?thesis.
7679 qed
7680
7681 AOT_theorem "id-nec4:2": < $\diamond$ (x  $\neq_E$  y)  $\equiv$  (x  $\neq_E$  y)> (234.2)
7682   by (meson "RE $\diamond$ " "S5Basic:2" "id-nec4:1" "E"(2,5) "Commutativity of  $\equiv$ ")
7683
7684 AOT_theorem "id-nec4:3": < $\diamond$ (x  $\neq_E$  y)  $\equiv$   $\square$ (x  $\neq_E$  y)> (234.3)

```

```

7685   by (meson "id-nec4:1" "id-nec4:2" "≡E"(5))
7686
7687 AOT_theorem "id-act2:1": <x =E y ≡  $\mathcal{A}x$  =E y> (235.1)
7688   by (meson "Act-Basic:5" "Act-Sub:2" "RA[2]" "id-nec3:2" "≡E"(1,6))
7689 AOT_theorem "id-act2:2": <x ≠E y ≡  $\mathcal{A}x$  ≠E y> (235.2)
7690   by (meson "Act-Basic:5" "Act-Sub:2" "RA[2]" "id-nec4:2" "≡E"(1,6))
7691
7692 AOT_theorem "ord=Eequiv:1": <0!x → x =E x> (236.1)
7693 proof (rule "→I")
7694   AOT_assume 1: <0!x>
7695   AOT_show <x =E x>
7696     apply (rule "=defI"(2) [OF "=E"]) apply "cqt:2[lambda]"
7697     apply (rule "β←C"(1))
7698     apply "cqt:2[lambda]"
7699     apply (simp add: "&I" "cqt:2[const_var]" [axiom_inst] prod_denotesI)
7700   by (simp add: "1" RN "&I" "oth-class-taut:3:a" "universal-cor")
7701 qed
7702
7703 AOT_theorem "ord=Eequiv:2": <x =E y → y =E x> (236.2)
7704 proof (rule CP)
7705   AOT_assume 1: <x =E y>
7706   AOT_hence 2: <x = y> by (metis "=E-simple:2" "vdash-properties:10")
7707   AOT_have <0!x> using 1 by (meson "&E"(1) "=E-simple:1" "≡E"(1))
7708   AOT_hence <x =E x> using "ord=Eequiv:1" "→E" by blast
7709   AOT_thus <y =E x> using "rule=E"[rotated, OF 2] by fast
7710 qed
7711
7712 AOT_theorem "ord=Eequiv:3": <(x =E y & y =E z) → x =E z> (236.3)
7713 proof (rule CP)
7714   AOT_assume 1: <x =E y & y =E z>
7715   AOT_hence <x = y & y = z>
7716     by (metis "&I" "&E"(1) "&E"(2) "=E-simple:2" "vdash-properties:6")
7717   AOT_hence <x = z> by (metis "id-eq:3" "vdash-properties:6")
7718   moreover AOT_have <x =E x>
7719     using 1 [THEN "&E"(1)] "&E"(1) "=E-simple:1" "≡E"(1)
7720     "ord=Eequiv:1" "→E" by blast
7721   ultimately AOT_show <x =E z>
7722     using "rule=E" by fast
7723 qed
7724
7725 AOT_theorem "ord=≡E:=1": <(0!x ∨ 0!y) → □(x = y ≡ x =E y)> (237.1)
7726 proof (rule CP)
7727   AOT_assume <0!x ∨ 0!y>
7728   moreover {
7729     AOT_assume <0!x>
7730     AOT_hence <□0!x> by (metis "oa-facts:1" "vdash-properties:10")
7731     moreover {
7732       AOT_modally_strict {
7733         AOT_have <0!x → (x = y ≡ x =E y)>
7734         proof (rule "→I"; rule "≡I"; rule "→I")
7735           AOT_assume <0!x>
7736           AOT_hence <x =E x> by (metis "ord=Eequiv:1" "→E")
7737           moreover AOT_assume <x = y>
7738           ultimately AOT_show <x =E y> using "rule=E" by fast
7739         next
7740           AOT_assume <x =E y>
7741           AOT_thus <x = y> by (metis "=E-simple:2" "→E")
7742         qed
7743       }
7744       AOT_hence <□0!x → □(x = y ≡ x =E y)> by (metis "RM:1")
7745     }
7746     ultimately AOT_have <□(x = y ≡ x =E y)> using "→E" by blast
7747   }

```

```

7748 moreover {
7749   AOT_assume <O!y>
7750   AOT_hence <□O!y> by (metis "oa-facts:1" "vdash-properties:10")
7751   moreover {
7752     AOT_modally_strict {
7753       AOT_have <O!y → (x = y ≡ x =E y)>
7754       proof (rule "→I"; rule "≡I"; rule "→I")
7755         AOT_assume <O!y>
7756         AOT_hence <y =E y> by (metis "ord=Eequiv:1" "→E")
7757         moreover AOT_assume <x = y>
7758         ultimately AOT_show <x =E y> using "rule=E" id_sym by fast
7759       next
7760         AOT_assume <x =E y>
7761         AOT_thus <x = y> by (metis "=E-simple:2" "→E")
7762       qed
7763     }
7764     AOT_hence <□O!y → □(x = y ≡ x =E y)> by (metis "RM:1")
7765   }
7766   ultimately AOT_have <□(x = y ≡ x =E y)> using "→E" by blast
7767 }
7768 ultimately AOT_show <□(x = y ≡ x =E y)> by (metis "√E"(3) "raa-cor:1")
7769 qed
7770
7771 AOT_theorem "ord-=E=:2": <O!y → [λx x = y]↓> (237.2)
7772 proof (rule "→I"; rule "safe-ext"[axiom_inst, THEN "→E"]; rule "&I")
7773   AOT_show <[λx x =E y]↓> by "cqt:2[lambda]"
7774 next
7775   AOT_assume <O!y>
7776   AOT_hence 1: <□(x = y ≡ x =E y)> for x
7777     using "ord-=E=:1" "→E" "√I" by blast
7778   AOT_have <□(x =E y ≡ x = y)> for x
7779     by (AOT_subst <x =E y ≡ x = y> <x = y ≡ x =E y>)
7780     (auto simp add: "Commutativity of ≡" 1)
7781   AOT_hence <∀x □(x =E y ≡ x = y)> by (rule GEN)
7782   AOT_thus <□∀x (x =E y ≡ x = y)> by (rule BF[THEN "→E"])
7783 qed
7784
7785
7786 AOT_theorem "ord-=E=:3": <[λxy O!x & O!y & x = y]↓> (237.3)
7787 proof (rule "safe-ext[2]"[axiom_inst, THEN "→E"]; rule "&I")
7788   AOT_show <[λxy O!x & O!y & x =E y]↓> by "cqt:2[lambda]"
7789 next
7790   AOT_show <□∀x∀y ([O!]x & [O!]y & x =E y ≡ [O!]x & [O!]y & x = y)>
7791   proof (rule RN; rule GEN; rule GEN; rule "≡I"; rule "→I")
7792     AOT_modally_strict {
7793       AOT_show <[O!]x & [O!]y & x = y> if <[O!]x & [O!]y & x =E y> for x y
7794       by (metis "&I" "&E"(1) "Conjunction Simplification"(2) "=E-simple:2"
7795           "modus-tollens:1" "raa-cor:1" that)
7796     }
7797   next
7798     AOT_modally_strict {
7799       AOT_show <[O!]x & [O!]y & x =E y> if <[O!]x & [O!]y & x = y> for x y
7800       apply (safe intro!: "&I")
7801       apply (metis that[THEN "&E"(1), THEN "&E"(1)])
7802       apply (metis that[THEN "&E"(1), THEN "&E"(2)])
7803       using "rule=E"[rotated, OF that[THEN "&E"(2)]]
7804       "ord=Eequiv:1"[THEN "→E", OF that[THEN "&E"(1), THEN "&E"(1)]]
7805       by fast
7806     }
7807   qed
7808 qed
7809
7810 AOT_theorem "ind-nec": <∀F ([F]x ≡ [F]y) → □∀F ([F]x ≡ [F]y)> (238)

```

```

7811 proof(rule "→I")
7812   AOT_assume <∀F ([F]x ≡ [F]y)>
7813   moreover AOT_have <[λx □∀F ([F]x ≡ [F]y)]↓> by "cqt:2[lambda]"
7814   ultimately AOT_have <[λx □∀F ([F]x ≡ [F]y)]x ≡ [λx □∀F ([F]x ≡ [F]y)]y>
7815   using "∀E" by blast
7816   moreover AOT_have <[λx □∀F ([F]x ≡ [F]y)]y>
7817   apply (rule "β←C"(1))
7818   apply "cqt:2[lambda]"
7819   apply (fact "cqt:2[const_var]"[axiom_inst])
7820   by (simp add: RN GEN "oth-class-taut:3:a")
7821   ultimately AOT_have <[λx □∀F ([F]x ≡ [F]y)]x> using "≡E" by blast
7822   AOT_thus <□∀F ([F]x ≡ [F]y)>
7823   using "β→C"(1) by blast
7824 qed
7825
7826 AOT_theorem "ord=E:1": <(O!x & O!y) → (∀F ([F]x ≡ [F]y) → x =E y)> (239.1)
7827 proof (rule "→I"; rule "→I")
7828   AOT_assume <∀F ([F]x ≡ [F]y)>
7829   AOT_hence <□∀F ([F]x ≡ [F]y)>
7830   using "ind-nec"[THEN "→E"] by blast
7831   moreover AOT_assume <O!x & O!y>
7832   ultimately AOT_have <O!x & O!y & □∀F ([F]x ≡ [F]y)>
7833   using "&I" by blast
7834   AOT_thus <x =E y> using "=E-simple:1"[THEN "≡E"(2)] by blast
7835 qed
7836
7837 AOT_theorem "ord=E:2": <(O!x & O!y) → (∀F ([F]x ≡ [F]y) → x = y)> (239.2)
7838 proof (rule "→I"; rule "→I")
7839   AOT_assume <O!x & O!y>
7840   moreover AOT_assume <∀F ([F]x ≡ [F]y)>
7841   ultimately AOT_have <x =E y>
7842   using "ord=E:1" "→E" by blast
7843   AOT_thus <x = y> using "=E-simple:2"[THEN "→E"] by blast
7844 qed
7845
7846 AOT_theorem "ord=E2:1": (240.1)
7847   <(O!x & O!y) → (x ≠ y ≡ [λz z =E x] ≠ [λz z =E y])>
7848 proof (rule "→I"; rule "≡I"; rule "→I";
7849   rule "≡dfI"[OF "=-infix"]; rule "raa-cor:2")
7850   AOT_assume 0: <O!x & O!y>
7851   AOT_assume <x ≠ y>
7852   AOT_hence 1: <¬(x = y)> using "≡dfE"[OF "=-infix"] by blast
7853   AOT_assume <[λz z =E x] = [λz z =E y]>
7854   moreover AOT_have <[λz z =E x]x>
7855   apply (rule "β←C"(1))
7856   apply "cqt:2[lambda]"
7857   apply (fact "cqt:2[const_var]"[axiom_inst])
7858   using "ord=Eequiv:1"[THEN "→E", OF 0[THEN "&E"(1)]] .
7859   ultimately AOT_have <[λz z =E y]x> using "rule=E" by fast
7860   AOT_hence <x =E y> using "β→C"(1) by blast
7861   AOT_hence <x = y> by (metis "=E-simple:2" "vdash-properties:6")
7862   AOT_thus <x = y & ¬(x = y)> using 1 "&I" by blast
7863 next
7864   AOT_assume <[λz z =E x] ≠ [λz z =E y]>
7865   AOT_hence 0: <¬([λz z =E x] = [λz z =E y])>
7866   using "≡dfE"[OF "=-infix"] by blast
7867   AOT_have <[λz z =E x]↓> by "cqt:2[lambda]"
7868   AOT_hence <[λz z =E x] = [λz z =E x]>
7869   by (metis "rule=I:1")
7870   moreover AOT_assume <x = y>
7871   ultimately AOT_have <[λz z =E x] = [λz z =E y]>
7872   using "rule=E" by fast
7873   AOT_thus <[λz z =E x] = [λz z =E y] & ¬([λz z =E x] = [λz z =E y])>

```



```

7874     using 0 "&I" by blast
7875 qed
7876
7877 AOT_theorem "ord=E2:2": (240.2)
7878   <(O!x & O!y) → (x ≠ y ≡ [λz z = x] ≠ [λz z = y])>
7879 proof (rule "→I"; rule "≡I"; rule "→I";
7880   rule "≡dfI"[OF "--infix"]; rule "raa-cor:2")
7881   AOT_assume 0: <O!x & O!y>
7882   AOT_assume <x ≠ y>
7883   AOT_hence 1: <¬(x = y)> using "≡dfE"[OF "--infix"] by blast
7884   AOT_assume <[λz z = x] = [λz z = y]>
7885   moreover AOT_have <[λz z = x]x>
7886     apply (rule "β←C"(1))
7887     apply (fact "ord==E=:2"[THEN "→E", OF 0[THEN "&E"(1)]])
7888     apply (fact "cqt:2[const_var]"[axiom_inst])
7889     by (simp add: "id-eq:1")
7890   ultimately AOT_have <[λz z = y]x> using "rule=E" by fast
7891   AOT_hence <x = y> using "β→C"(1) by blast
7892   AOT_thus <x = y & ¬(x = y)> using 1 "&I" by blast
7893 next
7894   AOT_assume 0: <O!x & O!y>
7895   AOT_assume <[λz z = x] ≠ [λz z = y]>
7896   AOT_hence 1: <¬([λz z = x] = [λz z = y])>
7897     using "≡dfE"[OF "--infix"] by blast
7898   AOT_have <[λz z = x]↓>
7899     by (fact "ord==E=:2"[THEN "→E", OF 0[THEN "&E"(1)]])
7900   AOT_hence <[λz z = x] = [λz z = x]>
7901     by (metis "rule=I:1")
7902   moreover AOT_assume <x = y>
7903   ultimately AOT_have <[λz z = x] = [λz z = y]>
7904     using "rule=E" by fast
7905   AOT_thus <[λz z = x] = [λz z = y] & ¬([λz z = x] = [λz z = y])>
7906     using 1 "&I" by blast
7907 qed
7908
7909 AOT_theorem ordnecfail: <O!x → □¬∃F x[F]> (241)
7910   by (meson "RM:1" "→I" nocoder[axiom_inst] "oa-facts:1" "→E")
7911
7912 AOT_theorem "ab-obey:1": <(A!x & A!y) → (∀F (x[F] ≡ y[F]) → x = y)> (242.1)
7913 proof (rule "→I"; rule "→I")
7914   AOT_assume 1: <A!x & A!y>
7915   AOT_assume <∀F (x[F] ≡ y[F])>
7916   AOT_hence <x[F] ≡ y[F]> for F using "∀E" by blast
7917   AOT_hence <□(x[F] ≡ y[F])> for F by (metis "en-eq:6[1]" "≡E"(1))
7918   AOT_hence <∀F □(x[F] ≡ y[F])> by (rule GEN)
7919   AOT_hence <□∀F (x[F] ≡ y[F])> by (rule BF[THEN "→E"])
7920   AOT_thus <x = y>
7921     using "≡dfI"[OF "identity:1", OF "∀I"(2)] 1 "&I" by blast
7922 qed
7923
7924 AOT_theorem "ab-obey:2": (242.2)
7925   <(∃F (x[F] & ¬y[F]) ∨ ∃F (y[F] & ¬x[F])) → x ≠ y>
7926 proof (rule "→I"; rule "≡dfI"[OF "--infix"]; rule "raa-cor:2")
7927   AOT_assume 1: <x = y>
7928   AOT_assume <∃F (x[F] & ¬y[F]) ∨ ∃F (y[F] & ¬x[F])>
7929   moreover {
7930     AOT_assume <∃F (x[F] & ¬y[F])>
7931     then AOT_obtain F where <x[F] & ¬y[F]>
7932       using "∃E"[rotated] by blast
7933     moreover AOT_have <y[F]>
7934       using calculation[THEN "&E"(1)] 1 "rule=E" by fast
7935     ultimately AOT_have <p & ¬p> for p
7936       by (metis "Conjunction Simplification"(2) "modus-tollens:2" "raa-cor:3")

```



```

7937 }
7938 moreover {
7939   AOT_assume <∃F (y[F] & ¬x[F])>
7940   then AOT_obtain F where <y[F] & ¬x[F]>
7941     using "∃E"[rotated] by blast
7942   moreover AOT_have <¬y[F]>
7943     using calculation[THEN "&E"(2)] 1 "rule=E" by fast
7944   ultimately AOT_have <p & ¬p> for p
7945     by (metis "Conjunction Simplification"(1) "modus-tollens:1" "raa-cor:3")
7946 }
7947 ultimately AOT_show <p & ¬p> for p
7948   by (metis "∨E"(3) "raa-cor:1")
7949 qed
7950
7951 AOT_theorem "encoders-are-abstract": <∃F x[F] → A!x> (243)
7952   by (meson "deduction-theorem" "≡E"(2) "modus-tollens:2" nocoder
7953     "oa-contingent:3" "vdash-properties:1[2]")
7954
7955 AOT_theorem "denote=:1": <∀H∃x x[H]> (244.1)
7956   by (rule GEN; rule "existence:2[1]"[THEN "≡dfE"]; "cqt:2")
7957
7958 AOT_theorem "denote=:2": <∀G∃x1...∃xn x1...xn[H]> (244.2)
7959   by (rule GEN; rule "existence:2"[THEN "≡dfE"]; "cqt:2")
7960
7961 AOT_theorem "denote=:2[2]": <∀G∃x1∃x2 x1x2[H]> (244.2)
7962   by (rule GEN; rule "existence:2[2]"[THEN "≡dfE"]; "cqt:2")
7963
7964 AOT_theorem "denote=:2[3]": <∀G∃x1∃x2∃x3 x1x2x3[H]> (244.2)
7965   by (rule GEN; rule "existence:2[3]"[THEN "≡dfE"]; "cqt:2")
7966
7967 AOT_theorem "denote=:2[4]": <∀G∃x1∃x2∃x3∃x4 x1x2x3x4[H]> (244.2)
7968   by (rule GEN; rule "existence:2[4]"[THEN "≡dfE"]; "cqt:2")
7969
7970 AOT_theorem "denote=:3": <∃x x[II] ≡ ∃H (H = II)> (244.3)
7971   using "existence:2[1]" "free-thms:1" "≡E"(2,5)
7972     "Commutativity of ≡" "≡Df" by blast
7973
7974 AOT_theorem "denote=:4": <(∃x1...∃xn x1...xn[II]) ≡ ∃H (H = II)> (244.4)
7975   using "existence:2" "free-thms:1" "≡E"(6) "≡Df" by blast
7976
7977 AOT_theorem "denote=:4[2]": <(∃x1∃x2 x1x2[II]) ≡ ∃H (H = II)> (244.4)
7978   using "existence:2[2]" "free-thms:1" "≡E"(6) "≡Df" by blast
7979
7980 AOT_theorem "denote=:4[3]": <(∃x1∃x2∃x3 x1x2x3[II]) ≡ ∃H (H = II)> (244.4)
7981   using "existence:2[3]" "free-thms:1" "≡E"(6) "≡Df" by blast
7982
7983 AOT_theorem "denote=:4[4]": <(∃x1∃x2∃x3∃x4 x1x2x3x4[II]) ≡ ∃H (H = II)> (244.4)
7984   using "existence:2[4]" "free-thms:1" "≡E"(6) "≡Df" by blast
7985
7986 AOT_theorem "A-objects!": <∃!x (A!x & ∀F (x[F] ≡ φ{F}))> (247)
7987 proof (rule "uniqueness:1"[THEN "≡dfI"])
7988   AOT_obtain a where a_prop: <A!a & ∀F (a[F] ≡ φ{F})>
7989     using "A-objects"[axiom_inst] "∃E"[rotated] by blast
7990   AOT_have <A!β & ∀F (β[F] ≡ φ{F}) → β = a> for β
7991   proof (rule "→I")
7992     AOT_assume β_prop: <A!β & ∀F (β[F] ≡ φ{F})>
7993     AOT_hence <β[F] ≡ φ{F}> for F
7994       using "∀E" "&E" by blast
7995     AOT_hence <β[F] ≡ a[F]> for F
7996       using a_prop[THEN "&E"(2)] "∀E" "≡E"(2,5)
7997       "Commutativity of ≡" by fast
7998     AOT_hence <∀F (β[F] ≡ a[F])> by (rule GEN)
7999     AOT_thus <β = a>

```

```

8000     using "ab-obey:1"[THEN "→E",
8001         OF "&I"[OF β_prop[THEN "&E"(1)], OF a_prop[THEN "&E"(1)]],
8002         THEN "→E"] by blast
8003 qed
8004 AOT_hence <∀β ((A!β & ∀F (β[F] ≡ φ{F})) → β = a)> by (rule GEN)
8005 AOT_thus <∃α ([A!]α & ∀F (α[F] ≡ φ{F}) &
8006         ∀β ([A!]β & ∀F (β[F] ≡ φ{F}) → β = α))>
8007     using "∃I" using a_prop "&I" by fast
8008 qed
8009
8010 AOT_theorem "obj-oth:1": <∃!x (A!x & ∀F (x[F] ≡ [F]y))> (248.1)
8011     using "A-objects!" by fast
8012
8013 AOT_theorem "obj-oth:2": <∃!x (A!x & ∀F (x[F] ≡ [F]y & [F]z))> (248.2)
8014     using "A-objects!" by fast
8015
8016 AOT_theorem "obj-oth:3": <∃!x (A!x & ∀F (x[F] ≡ [F]y ∨ [F]z))> (248.3)
8017     using "A-objects!" by fast
8018
8019 AOT_theorem "obj-oth:4": <∃!x (A!x & ∀F (x[F] ≡ □[F]y))> (248.4)
8020     using "A-objects!" by fast
8021
8022 AOT_theorem "obj-oth:5": <∃!x (A!x & ∀F (x[F] ≡ F = G))> (248.5)
8023     using "A-objects!" by fast
8024
8025 AOT_theorem "obj-oth:6": <∃!x (A!x & ∀F (x[F] ≡ □∀y([G]y → [F]y)))> (248.6)
8026     using "A-objects!" by fast
8027
8028 AOT_theorem "A-descriptions": <ιx (A!x & ∀F (x[F] ≡ φ{F}))↓> (249)
8029     by (rule "A-Exists:2"[THEN "≡E"(2)]; rule "RA[2]"; rule "A-objects!")
8030
8031 AOT_act_theorem "thm-can-terms2": (251)
8032     <y = ιx(A!x & ∀F (x[F] ≡ φ{F})) → (A!y & ∀F (y[F] ≡ φ{F}))>
8033     using "y-in:2" by blast
8034
8035 AOT_theorem "can-ab2": <y = ιx(A!x & ∀F (x[F] ≡ φ{F})) → A!y> (252)
8036 proof(rule "→I")
8037     AOT_assume <y = ιx(A!x & ∀F (x[F] ≡ φ{F}))>
8038     AOT_hence <A!y & ∀F (y[F] ≡ φ{F})>
8039         using "actual-desc:2"[THEN "→E"] by blast
8040     AOT_hence <A!y> by (metis "Act-Basic:2" "&E"(1) "≡E"(1))
8041     AOT_thus <A!y> by (metis "≡E"(2) "oa-facts:8")
8042 qed
8043
8044 AOT_act_theorem "desc-encode:1": <ιx(A!x & ∀F (x[F] ≡ φ{F}))[F] ≡ φ{F}> (253.1)
8045 proof -
8046     AOT_have <ιx(A!x & ∀F (x[F] ≡ φ{F}))↓>
8047         by (simp add: "A-descriptions")
8048     AOT_hence <A!ιx(A!x & ∀F (x[F] ≡ φ{F})) &
8049         ∀F(ιx(A!x & ∀F (x[F] ≡ φ{F}))[F] ≡ φ{F})>
8050         using "y-in:3"[THEN "→E"] by blast
8051     AOT_thus <ιx(A!x & ∀F (x[F] ≡ φ{F}))[F] ≡ φ{F}>
8052         using "&E" "∀E" by blast
8053 qed
8054
8055 AOT_act_theorem "desc-encode:2": <ιx(A!x & ∀F (x[F] ≡ φ{F}))[G] ≡ φ{G}> (253.2)
8056     using "desc-encode:1".
8057
8058 AOT_theorem "desc-nec-encode:1": (255.1)
8059     <ιx (A!x & ∀F (x[F] ≡ φ{F}))[F] ≡ Aφ{F}>
8060 proof -
8061     AOT_have 0: <ιx(A!x & ∀F (x[F] ≡ φ{F}))↓>
8062         by (simp add: "A-descriptions")

```

```

8063 AOT_hence <A(A!x(A!x & VF (x[F] ≡ φ{F})) &
8064     VF(Lx(A!x & VF (x[F] ≡ φ{F})) [F] ≡ φ{F}))>
8065     using "actual-desc:4"[THEN "→E"] by blast
8066 AOT_hence <AVF (Lx(A!x & VF (x[F] ≡ φ{F})) [F] ≡ φ{F})>
8067     using "Act-Basic:2" "&E"(2) "≡E"(1) by blast
8068 AOT_hence <VF A(Lx(A!x & VF (x[F] ≡ φ{F})) [F] ≡ φ{F})>
8069     using "≡E"(1) "logic-actual-nec:3" "vdash-properties:1[2]" by blast
8070 AOT_hence <A(Lx(A!x & VF (x[F] ≡ φ{F})) [F] ≡ φ{F})>
8071     using "VE" by blast
8072 AOT_hence <ALx(A!x & VF (x[F] ≡ φ{F})) [F] ≡ Aφ{F}>
8073     using "Act-Basic:5" "≡E"(1) by blast
8074 AOT_thus <Lx(A!x & VF (x[F] ≡ φ{F})) [F] ≡ Aφ{F}>
8075     using "en-eq:10[1]" [unvarify x1, OF 0] "≡E"(6) by blast
8076 qed
8077
8078 AOT_theorem "desc-nec-encode:2": (255.2)
8079 <Lx (A!x & VF (x[F] ≡ φ{F})) [G] ≡ Aφ{G}>
8080     using "desc-nec-encode:1".
8081
8082 AOT_theorem "Box-desc-encode:1": <□φ{G} → Lx(A!x & VF (x[F] ≡ φ{G})) [G]> (256.1)
8083     by (rule "→I"; rule "desc-nec-encode:2"[THEN "≡E"(2)])
8084     (meson "nec-imp-act" "vdash-properties:10")
8085
8086 AOT_theorem "Box-desc-encode:2": (256.2)
8087 <□φ{G} → □(Lx(A!x & VF (x[F] ≡ φ{G})) [G] ≡ φ{G})>
8088 proof(rule CP)
8089   AOT_assume <□φ{G}>
8090   AOT_hence <□□φ{G}> by (metis "S5Basic:6" "≡E"(1))
8091   moreover AOT_have <□□φ{G} → □(Lx(A!x & VF (x[F] ≡ φ{G})) [G] ≡ φ{G})>
8092   proof (rule RM; rule "→I")
8093     AOT_modally_strict {
8094       AOT_assume 1: <□φ{G}>
8095       AOT_hence <Lx(A!x & VF (x[F] ≡ φ{G})) [G]>
8096         using "Box-desc-encode:1" "→E" by blast
8097       moreover AOT_have <φ{G}>
8098         using 1 by (meson "qml:2"[axiom_inst] "→E")
8099       ultimately AOT_show <Lx(A!x & VF (x[F] ≡ φ{G})) [G] ≡ φ{G}>
8100         using "→I" "≡I" by simp
8101     }
8102   qed
8103   ultimately AOT_show <□(Lx(A!x & VF (x[F] ≡ φ{G})) [G] ≡ φ{G})>
8104     using "→E" by blast
8105 qed
8106
8107 definition rigid_condition where
8108   <rigid_condition φ ≡ ∀v . [v ⊨ ∀α (φ{α} → □φ{α})]>
8109 syntax rigid_condition :: <id_position ⇒ AOT_prop> ("RIGID'_CONDITION'(_)'")
8110
8111 AOT_theorem "strict-can:1[E]": (257.1)
8112     assumes <RIGID_CONDITION(φ)>
8113     shows <∀α (φ{α} → □φ{α})>
8114     using assms[unfolded rigid_condition_def] by auto
8115
8116 AOT_theorem "strict-can:1[I]": (257.1)
8117     assumes <⊢□ ∀α (φ{α} → □φ{α})>
8118     shows <RIGID_CONDITION(φ)>
8119     using assms rigid_condition_def by auto
8120
8121 AOT_theorem "box-phi-a:1": (258.1)
8122     assumes <RIGID_CONDITION(φ)>
8123     shows <(A!x & VF (x[F] ≡ φ{F})) → □(A!x & VF (x[F] ≡ φ{F}))>
8124 proof (rule "→I")
8125   AOT_assume a: <A!x & VF (x[F] ≡ φ{F})>

```

```

8126 AOT_hence b: <□A!x>
8127   by (metis "Conjunction Simplification"(1) "oa-facts:2" "→E")
8128 AOT_have <x[F] ≡ φ{F}> for F
8129   using a[THEN "&E"(2)] "∀E" by blast
8130 moreover AOT_have <□(x[F] → □x[F])> for F
8131   by (meson "pre-en-eq:1[1]" RN)
8132 moreover AOT_have <□(φ{F} → □φ{F})> for F
8133   using RN "strict-can:1[E]"[OF assms] "∀E" by blast
8134 ultimately AOT_have <□(x[F] ≡ φ{F})> for F
8135   using "sc-eq-box-box:5" "qml:2"[axiom_inst, THEN "→E"] "→E" "&I" by metis
8136 AOT_hence <∀F □(x[F] ≡ φ{F})> by (rule GEN)
8137 AOT_hence <□∀F (x[F] ≡ φ{F})> by (rule BF[THEN "→E"])
8138 AOT_thus <□([A!]x & ∀F (x[F] ≡ φ{F}))>
8139   using b "KBasic:3" "≡S"(1) "≡E"(2) by blast
8140 qed
8141
8142 AOT_theorem "box-phi-a:2": (258.2)
8143   assumes <RIGID_CONDITION(φ)>
8144   shows <y = ιx(A!x & ∀F (x[F] ≡ φ{F})) → (A!y & ∀F (y[F] ≡ φ{F}))>
8145 proof(rule "→I")
8146   AOT_assume <y = ιx(A!x & ∀F (x[F] ≡ φ{F}))>
8147   AOT_hence <A(A!y & ∀F (y[F] ≡ φ{F}))>
8148     using "actual-desc:2"[THEN "→E"] by fast
8149   AOT_hence abs: <AA!y> and <A∀F (y[F] ≡ φ{F})>
8150     using "Act-Basic:2" "&E" "≡E"(1) by blast+
8151   AOT_hence <∀F A(y[F] ≡ φ{F})>
8152     by (metis "≡E"(1) "logic-actual-nec:3" "vdash-properties:1[2]")
8153   AOT_hence <A(y[F] ≡ φ{F})> for F
8154     using "∀E" by blast
8155   AOT_hence <Ay[F] ≡ Aφ{F}> for F
8156     by (metis "Act-Basic:5" "≡E"(1))
8157   AOT_hence <y[F] ≡ φ{F}> for F
8158     using "sc-eq-fur:2"[THEN "→E"],
8159     OF "strict-can:1[E]"[OF assms,
8160       THEN "∀E"(2)[where β=F], THEN RN]]
8161     by (metis "en-eq:10[1]" "≡E"(6))
8162   AOT_hence <∀F (y[F] ≡ φ{F})> by (rule GEN)
8163   AOT_thus <[A!]y & ∀F (y[F] ≡ φ{F})>
8164     using abs "&I" "≡E"(2) "oa-facts:8" by blast
8165 qed
8166
8167 AOT_theorem "box-phi-a:3": (258.3)
8168   assumes <RIGID_CONDITION(φ)>
8169   shows <ιx(A!x & ∀F (x[F] ≡ φ{F})) [F] ≡ φ{F}>
8170   using "desc-nec-encode:2"
8171     "sc-eq-fur:2"[THEN "→E"],
8172     OF "strict-can:1[E]"[OF assms,
8173       THEN "∀E"(2)[where β=F], THEN RN]]
8174     "≡E"(5) by blast
8175
8176 AOT_define Null :: <τ ⇒ φ> ("Null'(_)")
8177   "df-null-uni:1": <Null(x) ≡df A!x & ¬∃F x[F]> (260.1)
8178
8179 AOT_define Universal :: <τ ⇒ φ> ("Universal'(_)")
8180   "df-null-uni:2": <Universal(x) ≡df A!x & ∀F x[F]> (260.2)
8181
8182 AOT_theorem "null-uni-uniq:1": <∃!x Null(x)> (261.1)
8183 proof (rule "uniqueness:1"[THEN "≡dfI"])
8184   AOT_obtain a where a_prop: <A!a & ∀F (a[F] ≡ ¬(F = F))>
8185     using "A-objects"[axiom_inst] "∃E"[rotated] by fast
8186   AOT_have a_null: <¬a[F]> for F
8187   proof (rule "raa-cor:2")
8188     AOT_assume <a[F]>

```

```

8189   AOT_hence < $\neg(F = F)$ > using a_prop[THEN "&E"(2)] " $\forall E$ " " $\equiv E$ " by blast
8190   AOT_hence < $F = F \ \& \ \neg(F = F)$ > by (metis "id-eq:1" "raa-cor:3")
8191   AOT_thus < $p \ \& \ \neg p$ > for p by (metis "raa-cor:1")
8192 qed
8193 AOT_have < $\text{Null}(a) \ \& \ \forall\beta (\text{Null}(\beta) \rightarrow \beta = a)$ >
8194 proof (rule "&I")
8195   AOT_have < $\neg\exists F a[F]$ >
8196     using a_null by (metis "instantiation" "reductio-aa:1")
8197   AOT_thus < $\text{Null}(a)$ >
8198     using "df-null-uni:1"[THEN " $\equiv_{df}I$ "] a_prop[THEN "&E"(1)] "&I" by metis
8199 next
8200 AOT_show < $\forall\beta (\text{Null}(\beta) \rightarrow \beta = a)$ >
8201 proof (rule GEN; rule " $\rightarrow I$ ")
8202   fix  $\beta$ 
8203   AOT_assume a: < $\text{Null}(\beta)$ >
8204   AOT_hence < $\neg\exists F \beta[F]$ >
8205     using "df-null-uni:1"[THEN " $\equiv_{df}E$ "] "&E" by blast
8206   AOT_hence  $\beta\_null$ : < $\neg\beta[F]$ > for F
8207     by (metis "existential:2[const_var]" "reductio-aa:1")
8208   AOT_have < $\forall F (\beta[F] \equiv a[F])$ >
8209     apply (rule GEN; rule " $\equiv I$ "; rule CP)
8210     using "raa-cor:3"  $\beta\_null$  a_null by blast+
8211   moreover AOT_have < $A!\beta$ >
8212     using a "df-null-uni:1"[THEN " $\equiv_{df}E$ "] "&E" by blast
8213   ultimately AOT_show < $\beta = a$ >
8214     using a_prop[THEN "&E"(1)] "ab-obey:1"[THEN " $\rightarrow E$ ", THEN " $\rightarrow E$ "]
8215     "&I" by blast
8216   qed
8217 qed
8218 AOT_thus < $\exists\alpha (\text{Null}(\alpha) \ \& \ \forall\beta (\text{Null}(\beta) \rightarrow \beta = \alpha))$ >
8219   using " $\exists I$ "(2) by fast
8220 qed
8221
8222 AOT_theorem "null-uni-uniq:2": < $\exists!x \text{Universal}(x)$ > (261.2)
8223 proof (rule "uniqueness:1"[THEN " $\equiv_{df}I$ "])
8224   AOT_obtain a where a_prop: < $A!a \ \& \ \forall F (a[F] \equiv F = F)$ >
8225     using "A-objects"[axiom_inst] " $\exists E$ "[rotated] by fast
8226   AOT_hence aF: < $a[F]$ > for F using "&E" " $\forall E$ " " $\equiv E$ " "id-eq:1" by fast
8227   AOT_hence < $\text{Universal}(a)$ >
8228     using "df-null-uni:2"[THEN " $\equiv_{df}I$ "] "&I" a_prop[THEN "&E"(1)] GEN by blast
8229   moreover AOT_have < $\forall\beta (\text{Universal}(\beta) \rightarrow \beta = a)$ >
8230 proof (rule GEN; rule " $\rightarrow I$ ")
8231   fix  $\beta$ 
8232   AOT_assume < $\text{Universal}(\beta)$ >
8233   AOT_hence abs_ $\beta$ : < $A!\beta$ > and < $\beta[F]$ > for F
8234     using "df-null-uni:2"[THEN " $\equiv_{df}E$ "] "&E" " $\forall E$ " by blast+
8235   AOT_hence < $\beta[F] \equiv a[F]$ > for F
8236     using aF by (metis "deduction-theorem" " $\equiv I$ ")
8237   AOT_hence < $\forall F (\beta[F] \equiv a[F])$ > by (rule GEN)
8238   AOT_thus < $\beta = a$ >
8239     using a_prop[THEN "&E"(1)] "ab-obey:1"[THEN " $\rightarrow E$ ", THEN " $\rightarrow E$ "]
8240     "&I" abs_ $\beta$  by blast
8241   qed
8242   ultimately AOT_show < $\exists\alpha (\text{Universal}(\alpha) \ \& \ \forall\beta (\text{Universal}(\beta) \rightarrow \beta = \alpha))$ >
8243     using "&I" " $\exists I$ " by fast
8244 qed
8245
8246 AOT_theorem "null-uni-uniq:3": < $\iota x \text{Null}(x)\downarrow$ > (261.3)
8247   using "A-Exists:2" "RA[2]" " $\equiv E$ "(2) "null-uni-uniq:1" by blast
8248
8249 AOT_theorem "null-uni-uniq:4": < $\iota x \text{Universal}(x)\downarrow$ > (261.4)
8250   using "A-Exists:2" "RA[2]" " $\equiv E$ "(2) "null-uni-uniq:2" by blast
8251

```

```

8252 AOT_define Null_object :: <κs> (<a0>)
8253   "df-null-uni-terms:1": <a0 =df λx Null(x)> (262.1)
8254
8255 AOT_define Universal_object :: <κs> (<av>)
8256   "df-null-uni-terms:2": <av =df λx Universal(x)> (262.2)
8257
8258 AOT_theorem "null-uni-facts:1": <Null(x) → □Null(x)> (263.1)
8259 proof (rule "→I")
8260   AOT_assume <Null(x)>
8261   AOT_hence x_abs: <A!x> and x_null: <¬∃F x[F]>
8262     using "df-null-uni:1"[THEN "≡dfE"] "&E" by blast+
8263   AOT_have <¬x[F]> for F using x_null
8264     using "existential:2[const_var]" "reductio-aa:1"
8265     by metis
8266   AOT_hence <□¬x[F]> for F by (metis "en-eq:7[1]" "≡E"(1))
8267   AOT_hence <∀F □¬x[F]> by (rule GEN)
8268   AOT_hence <□∀F ¬x[F]> by (rule BF[THEN "→E"])
8269   moreover AOT_have <□∀F ¬x[F] → □¬∃F x[F]>
8270     apply (rule RM)
8271     by (metis (full_types) "instantiation" "cqt:2[const_var]"[axiom_inst]
8272         "→I" "reductio-aa:1" "rule-ui:1")
8273   ultimately AOT_have <□¬∃F x[F]>
8274     by (metis "→E")
8275   moreover AOT_have <□A!x> using x_abs
8276     using "oa-facts:2" "vdash-properties:10" by blast
8277   ultimately AOT_have r: <□(A!x & ¬∃F x[F])>
8278     by (metis "KBasic:3" "&I" "≡E"(3) "raa-cor:3")
8279   AOT_show <□Null(x)>
8280     by (AOT_subst <Null(x)> <A!x & ¬∃F x[F]>)
8281     (auto simp: "df-null-uni:1" "≡Df" r)
8282 qed
8283
8284 AOT_theorem "null-uni-facts:2": <Universal(x) → □Universal(x)> (263.2)
8285 proof (rule "→I")
8286   AOT_assume <Universal(x)>
8287   AOT_hence x_abs: <A!x> and x_univ: <∀F x[F]>
8288     using "df-null-uni:2"[THEN "≡dfE"] "&E" by blast+
8289   AOT_have <x[F]> for F using x_univ "∨E" by blast
8290   AOT_hence <□x[F]> for F by (metis "en-eq:2[1]" "≡E"(1))
8291   AOT_hence <∀F □x[F]> by (rule GEN)
8292   AOT_hence <□∀F x[F]> by (rule BF[THEN "→E"])
8293   moreover AOT_have <□A!x> using x_abs
8294     using "oa-facts:2" "vdash-properties:10" by blast
8295   ultimately AOT_have r: <□(A!x & ∀F x[F])>
8296     by (metis "KBasic:3" "&I" "≡E"(3) "raa-cor:3")
8297   AOT_show <□Universal(x)>
8298     by (AOT_subst <Universal(x)> <A!x & ∀F x[F]>)
8299     (auto simp add: "df-null-uni:2" "≡Df" r)
8300 qed
8301
8302 AOT_theorem "null-uni-facts:3": <Null(a0)> (263.3)
8303   apply (rule "=dfI"(2)[OF "df-null-uni-terms:1"])
8304   apply (simp add: "null-uni-uniq:3")
8305   using "actual-desc:4"[THEN "→E", OF "null-uni-uniq:3"]
8306     "sc-eq-fur:2"[THEN "→E",
8307       OF "null-uni-facts:1"[unvarify x, THEN RN, OF "null-uni-uniq:3"],
8308       THEN "≡E"(1)]
8309   by blast
8310
8311 AOT_theorem "null-uni-facts:4": <Universal(av)> (263.4)
8312   apply (rule "=dfI"(2)[OF "df-null-uni-terms:2"])
8313   apply (simp add: "null-uni-uniq:4")
8314   using "actual-desc:4"[THEN "→E", OF "null-uni-uniq:4"]

```

```

8315     "sc-eq-fur:2"[THEN "→E",
8316         OF "null-uni-facts:2"[unvarify x, THEN RN, OF "null-uni-uniq:4"],
8317         THEN "≡E"(1)]
8318   by blast
8319
8320 AOT_theorem "null-uni-facts:5": <a0 ≠ av> (263.5)
8321 proof (rule "=dfI"(2)[OF "df-null-uni-terms:1", OF "null-uni-uniq:3"];
8322     rule "=dfI"(2)[OF "df-null-uni-terms:2", OF "null-uni-uniq:4"];
8323     rule "≡dfI"[OF "--infix"];
8324     rule "raa-cor:2")
8325 AOT_obtain x where nullx: <Null(x)>
8326   by (metis "instantiation" "df-null-uni-terms:1" "existential:1"
8327       "null-uni-facts:3" "null-uni-uniq:3" "rule-id-df:2:b[zero]")
8328 AOT_hence act_null: <ANull(x)>
8329   by (metis "nec-imp-act" "null-uni-facts:1" "→E")
8330 AOT_assume <ιx Null(x) = ιx Universal(x)>
8331 AOT_hence <A∀x(Null(x) ≡ Universal(x))>
8332   using "actual-desc:5"[THEN "→E"] by blast
8333 AOT_hence <∀x A(Null(x) ≡ Universal(x))>
8334   by (metis "≡E"(1) "logic-actual-nec:3" "vdash-properties:1[2]")
8335 AOT_hence <ANull(x) ≡ AUniversal(x)>
8336   using "Act-Basic:5" "≡E"(1) "rule-ui:3" by blast
8337 AOT_hence <AUniversal(x)> using act_null "≡E" by blast
8338 AOT_hence <Universal(x)>
8339   by (metis RN "≡E"(1) "null-uni-facts:2" "sc-eq-fur:2" "→E")
8340 AOT_hence <∀F x[F]> using "≡dfE"[OF "df-null-uni:2"] "&E" by metis
8341 moreover AOT_have <¬∃F x[F]>
8342   using nullx "≡dfE"[OF "df-null-uni:1"] "&E" by metis
8343 ultimately AOT_show <p & ¬p> for p
8344   by (metis "cqt-further:1" "raa-cor:3" "→E")
8345 qed
8346
8347 AOT_theorem "null-uni-facts:6": <a0 = ιx(A!x & ∀F (x[F] ≡ F ≠ F))> (263.6)
8348 proof (rule "ab-obey:1"[unvarify x y, THEN "→E", THEN "→E"])
8349   AOT_show <ιx([A!]x & ∀F (x[F] ≡ F ≠ F))↓>
8350     by (simp add: "A-descriptions")
8351 next
8352   AOT_show <a0↓>
8353     by (rule "=dfI"(2)[OF "df-null-uni-terms:1", OF "null-uni-uniq:3"])
8354       (simp add: "null-uni-uniq:3")
8355 next
8356   AOT_have <ιx([A!]x & ∀F (x[F] ≡ F ≠ F))↓>
8357     by (simp add: "A-descriptions")
8358   AOT_hence 1: <ιx([A!]x & ∀F (x[F] ≡ F ≠ F)) = ιx([A!]x & ∀F (x[F] ≡ F ≠ F))>
8359     using "rule=I:1" by blast
8360   AOT_show <[A!]a0 & [A!]ιx([A!]x & ∀F (x[F] ≡ F ≠ F))>
8361     apply (rule "=dfI"(2)[OF "df-null-uni-terms:1", OF "null-uni-uniq:3"];
8362           rule "&I")
8363     apply (meson "≡dfE" "Conjunction Simplification"(1)
8364             "df-null-uni:1" "df-null-uni-terms:1" "null-uni-facts:3"
8365             "null-uni-uniq:3" "rule-id-df:2:a[zero]" "→E")
8366     using "can-ab2"[unvarify y, OF "A-descriptions", THEN "→E", OF 1].
8367 next
8368   AOT_show <∀F (a0[F] ≡ ιx([A!]x & ∀F (x[F] ≡ F ≠ F))[F])>
8369   proof (rule GEN)
8370     fix F
8371     AOT_have <¬a0[F]>
8372       by (rule "=dfI"(2)[OF "df-null-uni-terms:1", OF "null-uni-uniq:3"])
8373         (metis (no_types, lifting) "≡dfE" "&E"(2) "∀I"(2) "∀E"(3) "∃I"(2)
8374             "df-null-uni:1" "df-null-uni-terms:1" "null-uni-facts:3"
8375             "raa-cor:2" "rule-id-df:2:a[zero]"
8376             "russell-axiom[enc,1].ψ_denotes_asm")
8377     moreover AOT_have <¬ιx([A!]x & ∀F (x[F] ≡ F ≠ F))[F]>

```



```

8378   proof(rule "raa-cor:2")
8379     AOT_assume 0: <math>\lambda x([A!]x \ \&\ \forall F(x[F] \equiv F \neq F)) [F]>
8380     AOT_hence <math>\mathcal{A}(F \neq F)>
8381       using "desc-nec-encode:2" [THEN "≡E"(1), OF 0] by blast
8382     moreover AOT_have <math>\neg \mathcal{A}(F \neq F)>
8383       using "≡dfE" "id-act:2" "id-eq:1" "≡E"(2)
8384         "=-infix" "raa-cor:3" by blast
8385     ultimately AOT_show <math>\mathcal{A}(F \neq F) \ \&\ \neg \mathcal{A}(F \neq F)> by (rule "&I")
8386   qed
8387   ultimately AOT_show <math>a_0[F] \equiv \lambda x([A!]x \ \&\ \forall F(x[F] \equiv F \neq F)) [F]>
8388     using "deduction-theorem" "≡I" "raa-cor:4" by blast
8389   qed
8390 qed
8391
8392 AOT_theorem "null-uni-facts:7": <math>a_v = \lambda x(A!x \ \&\ \forall F(x[F] \equiv F = F))> \tag{263.7}
8393 proof (rule "ab-obey:1" [unvarify x y, THEN "→E", THEN "→E"])
8394   AOT_show <math>\lambda x([A!]x \ \&\ \forall F(x[F] \equiv F = F)) \downarrow >
8395     by (simp add: "A-descriptions")
8396 next
8397   AOT_show <math>a_v \downarrow >
8398     by (rule "≡dfI"(2) [OF "df-null-uni-terms:2", OF "null-uni-uniq:4"])
8399       (simp add: "null-uni-uniq:4")
8400 next
8401   AOT_have <math>\lambda x([A!]x \ \&\ \forall F(x[F] \equiv F = F)) \downarrow >
8402     by (simp add: "A-descriptions")
8403   AOT_hence 1: <math>\lambda x([A!]x \ \&\ \forall F(x[F] \equiv F = F)) = \lambda x([A!]x \ \&\ \forall F(x[F] \equiv F = F))>
8404     using "rule=I:1" by blast
8405   AOT_show <math>[A!]a_v \ \&\ [A!] \lambda x([A!]x \ \&\ \forall F(x[F] \equiv F = F))>
8406     apply (rule "≡dfI"(2) [OF "df-null-uni-terms:2", OF "null-uni-uniq:4"];
8407           rule "&I")
8408     apply (meson "≡dfE" "Conjunction Simplification"(1) "df-null-uni:2"
8409              "df-null-uni-terms:2" "null-uni-facts:4" "null-uni-uniq:4"
8410              "rule-id-df:2:a[zero]" "→E")
8411     using "can-ab2" [unvarify y, OF "A-descriptions", THEN "→E", OF 1].
8412 next
8413   AOT_show <math>\forall F(a_v[F] \equiv \lambda x([A!]x \ \&\ \forall F(x[F] \equiv F = F)) [F])>
8414   proof (rule GEN)
8415     fix F
8416     AOT_have <math>a_v[F]>
8417       apply (rule "≡dfI"(2) [OF "df-null-uni-terms:2", OF "null-uni-uniq:4"])
8418       using "≡dfE" "&E"(2) "df-null-uni:2" "df-null-uni-terms:2"
8419         "null-uni-facts:4" "null-uni-uniq:4" "rule-id-df:2:a[zero]"
8420         "rule-ui:3" by blast
8421     moreover AOT_have <math>\lambda x([A!]x \ \&\ \forall F(x[F] \equiv F = F)) [F]>
8422       using "RA[2]" "desc-nec-encode:2" "id-eq:1" "≡E"(2) by fastforce
8423     ultimately AOT_show <math>a_v[F] \equiv \lambda x([A!]x \ \&\ \forall F(x[F] \equiv F = F)) [F]>
8424       using "deduction-theorem" "≡I" by simp
8425   qed
8426 qed
8427
8428 AOT_theorem "aclassical:1": \tag{265.1}
8429 <math>\forall R \exists x \exists y (A!x \ \&\ A!y \ \&\ x \neq y \ \&\ [\lambda z [R]zx] = [\lambda z [R]zy])>
8430 proof (rule GEN)
8431   fix R
8432   AOT_obtain a where a_prop:
8433     <math>A!a \ \&\ \forall F(a[F] \equiv \exists y(A!y \ \&\ F = [\lambda z [R]zy] \ \&\ \neg y[F]))>
8434     using "A-objects" [axiom_inst] "∃E" [rotated] by fast
8435   AOT_have a_enc: <math>a[\lambda z [R]za]>
8436   proof (rule "raa-cor:1")
8437     AOT_assume 0: <math>\neg a[\lambda z [R]za]>
8438     AOT_hence <math>\neg \exists y(A!y \ \&\ [\lambda z [R]za] = [\lambda z [R]zy] \ \&\ \neg [\lambda z [R]za])>
8439       by (rule a_prop [THEN "&E"(2), THEN "∀E"(1) [where  $\tau = \llbracket \lambda z [R]za \rrbracket$ ]],
8440         THEN "oth-class-taut:4:b" [THEN "≡E"(1)],

```



```

8441         THEN "≡E"(1), rotated])
8442     "cqt:2[lambda]"
8443 AOT_hence <∀y ¬(A!y & [λz [R]za] = [λz [R]zy] & ¬y[λz [R]za])>
8444     using "cqt-further:4" "vdash-properties:10" by blast
8445 AOT_hence <¬(A!a & [λz [R]za] = [λz [R]za] & ¬a[λz [R]za])>
8446     using "∀E" by blast
8447 AOT_hence <(A!a & [λz [R]za] = [λz [R]za]) → a[λz [R]za]>
8448     by (metis "&I" "deduction-theorem" "raa-cor:3")
8449 moreover AOT_have <[λz [R]za] = [λz [R]za]>
8450     by (rule "=I") "cqt:2[lambda]"
8451 ultimately AOT_have <a[λz [R]za]>
8452     using a_prop[THEN "&E"(1)] "→E" "&I" by blast
8453 AOT_thus <a[λz [R]za] & ¬a[λz [R]za]>
8454     using 0 "&I" by blast
8455 qed
8456 AOT_hence <∃y(A!y & [λz [R]za] = [λz [R]zy] & ¬y[λz [R]za])>
8457     by (rule a_prop[THEN "&E"(2), THEN "∀E"(1), THEN "≡E"(1), rotated])
8458     "cqt:2"
8459 then AOT_obtain b where b_prop:
8460     <A!b & [λz [R]za] = [λz [R]zb] & ¬b[λz [R]za]>
8461     using "∃E"[rotated] by blast
8462 AOT_have <a ≠ b>
8463     apply (rule "≡dfI"[OF "--infix"])
8464     using a_enc b_prop[THEN "&E"(2)]
8465     using "¬I" "rule=E" id_sym "≡E"(4) "oth-class-taut:3:a"
8466     "raa-cor:3" "reductio-aa:1" by fast
8467 AOT_hence <A!a & A!b & a ≠ b & [λz [R]za] = [λz [R]zb]>
8468     using b_prop "&E" a_prop "&I" by meson
8469 AOT_hence <∃y (A!a & A!y & a ≠ y & [λz [R]za] = [λz [R]zy])> by (rule "∃I")
8470 AOT_thus <∃x∃y (A!x & A!y & x ≠ y & [λz [R]zx] = [λz [R]zy])> by (rule "∃I")
8471 qed
8472
8473 AOT_theorem "aclassical:2": (265.2)
8474     <∀R∃x∃y(A!x & A!y & x ≠ y & [λz [R]xz] = [λz [R]yz])>
8475 proof(rule GEN)
8476     fix R
8477     AOT_obtain a where a_prop:
8478     <A!a & ∀F (a[F] ≡ ∃y(A!y & F = [λz [R]yz] & ¬y[F]))>
8479     using "A-objects"[axiom_inst] "∃E"[rotated] by fast
8480     AOT_have a_enc: <a[λz [R]az]>
8481     proof (rule "raa-cor:1")
8482         AOT_assume 0: <¬a[λz [R]az]>
8483         AOT_hence <¬∃y(A!y & [λz [R]az] = [λz [R]yz] & ¬y[λz [R]az])>
8484             by (rule a_prop[THEN "&E"(2), THEN "∀E"(1)[where τ="«[λz [R]az]»"],
8485                 THEN "oth-class-taut:4:b"[THEN "≡E"(1)],
8486                 THEN "≡E"(1), rotated])
8487         "cqt:2[lambda]"
8488         AOT_hence <∀y ¬(A!y & [λz [R]az] = [λz [R]yz] & ¬y[λz [R]az])>
8489             using "cqt-further:4" "vdash-properties:10" by blast
8490         AOT_hence <¬(A!a & [λz [R]az] = [λz [R]az] & ¬a[λz [R]az])>
8491             using "∀E" by blast
8492         AOT_hence <(A!a & [λz [R]az] = [λz [R]az]) → a[λz [R]az]>
8493             by (metis "&I" "deduction-theorem" "raa-cor:3")
8494         moreover AOT_have <[λz [R]az] = [λz [R]az]>
8495             by (rule "=I") "cqt:2[lambda]"
8496         ultimately AOT_have <a[λz [R]az]>
8497             using a_prop[THEN "&E"(1)] "→E" "&I" by blast
8498         AOT_thus <a[λz [R]az] & ¬a[λz [R]az]>
8499             using 0 "&I" by blast
8500     qed
8501     AOT_hence <∃y(A!y & [λz [R]az] = [λz [R]yz] & ¬y[λz [R]az])>
8502     by (rule a_prop[THEN "&E"(2), THEN "∀E"(1), THEN "≡E"(1), rotated])
8503     "cqt:2"

```

```

8504 then AOT_obtain b where b_prop:
8505   <A!b & [ $\lambda z$  [R]az] = [ $\lambda z$  [R]bz] &  $\neg b$ [ $\lambda z$  [R]az]>
8506   using "E"[rotated] by blast
8507 AOT_have <a  $\neq$  b>
8508   apply (rule "E"[OF "--infix"])
8509   using a_enc b_prop[THEN "&E"(2)]
8510   using " $\neg$ I" "rule=E" id_sym "E"(4) "oth-class-taut:3:a"
8511     "raa-cor:3" "reductio-aa:1" by fast
8512 AOT_hence <A!a & A!b & a  $\neq$  b & [ $\lambda z$  [R]az] = [ $\lambda z$  [R]bz]>
8513   using b_prop "&E" a_prop "&I" by meson
8514 AOT_hence < $\exists y$  (A!a & A!y & a  $\neq$  y & [ $\lambda z$  [R]az] = [ $\lambda z$  [R]yz])> by (rule "E")
8515 AOT_thus < $\exists x \exists y$  (A!x & A!y & x  $\neq$  y & [ $\lambda z$  [R]xz] = [ $\lambda z$  [R]yz])> by (rule "E")
8516 qed
8517
8518 AOT_theorem "aclassical:3": (265.3)
8519   < $\forall F \exists x \exists y$  (A!x & A!y & x  $\neq$  y & [ $\lambda$  [F]x] = [ $\lambda$  [F]y])>
8520 proof(rule GEN)
8521   fix R
8522   AOT_obtain a where a_prop:
8523     <A!a &  $\forall F$  (a[F]  $\equiv \exists y$  (A!y & F = [ $\lambda z$  [R]y] &  $\neg y$ [F]))>
8524     using "A-objects"[axiom_inst] "E"[rotated] by fast
8525   AOT_have den: <[ $\lambda z$  [R]a] $\downarrow$ > by "cqt:2[lambda]"
8526   AOT_have a_enc: <a[ $\lambda z$  [R]a]>
8527   proof (rule "raa-cor:1")
8528     AOT_assume 0: < $\neg a$ [ $\lambda z$  [R]a]>
8529     AOT_hence < $\neg \exists y$  (A!y & [ $\lambda z$  [R]a] = [ $\lambda z$  [R]y] &  $\neg y$ [ $\lambda z$  [R]a])>
8530       by (safe intro!: a_prop[THEN "&E"(2), THEN "VE"(1)[where  $\tau = \llcorner \lambda z$  [R]a \rceil]],
8531         THEN "oth-class-taut:4:b"[THEN "E"(1)],
8532         THEN "E"(1), rotated] "cqt:2")
8533     AOT_hence < $\forall y \neg$  (A!y & [ $\lambda z$  [R]a] = [ $\lambda z$  [R]y] &  $\neg y$ [ $\lambda z$  [R]a])>
8534       using "cqt-further:4" " $\rightarrow$ E" by blast
8535     AOT_hence < $\neg$  (A!a & [ $\lambda z$  [R]a] = [ $\lambda z$  [R]a] &  $\neg a$ [ $\lambda z$  [R]a])> using "VE" by blast
8536     AOT_hence <(A!a & [ $\lambda z$  [R]a] = [ $\lambda z$  [R]a])  $\rightarrow$  a[ $\lambda z$  [R]a]>
8537       by (metis "&I" "deduction-theorem" "raa-cor:3")
8538     AOT_hence <a[ $\lambda z$  [R]a]>
8539       using a_prop[THEN "&E"(1)] " $\rightarrow$ E" "&I"
8540       by (metis "rule=I:1" den)
8541     AOT_thus <a[ $\lambda z$  [R]a] &  $\neg a$ [ $\lambda z$  [R]a]> by (metis "0" "raa-cor:3")
8542   qed
8543   AOT_hence < $\exists y$  (A!y & [ $\lambda z$  [R]a] = [ $\lambda z$  [R]y] &  $\neg y$ [ $\lambda z$  [R]a])>
8544     by (rule a_prop[THEN "&E"(2), THEN "VE"(1), OF den, THEN "E"(1), rotated])
8545   then AOT_obtain b where b_prop: <A!b & [ $\lambda z$  [R]a] = [ $\lambda z$  [R]b] &  $\neg b$ [ $\lambda z$  [R]a]>
8546     using "E"[rotated] by blast
8547   AOT_have 1: <a  $\neq$  b>
8548     apply (rule "E"[OF "--infix"])
8549     using a_enc b_prop[THEN "&E"(2)]
8550     using " $\neg$ I" "rule=E" id_sym "E"(4) "oth-class-taut:3:a"
8551       "raa-cor:3" "reductio-aa:1" by fast
8552   AOT_have a: <[ $\lambda$  [R]a] = ([R]a)>
8553     apply (rule "lambda-predicates:3[zero]"[axiom_inst, unvarify p])
8554     by (meson "log-prop-prop:2")
8555   AOT_have b: <[ $\lambda$  [R]b] = ([R]b)>
8556     apply (rule "lambda-predicates:3[zero]"[axiom_inst, unvarify p])
8557     by (meson "log-prop-prop:2")
8558   AOT_have <[ $\lambda$  [R]a] = [ $\lambda$  [R]b]>
8559     apply (rule "rule=E"[rotated, OF a[THEN id_sym]])
8560     apply (rule "rule=E"[rotated, OF b[THEN id_sym]])
8561     apply (rule "identity:4"[THEN "E", OF "&I", rotated])
8562     using b_prop "&E" apply blast
8563     apply (safe intro!: "&I")
8564     by (simp add: "log-prop-prop:2")+
8565   AOT_hence <A!a & A!b & a  $\neq$  b & [ $\lambda$  [R]a] = [ $\lambda$  [R]b]>
8566     using 1 a_prop[THEN "&E"(1)] b_prop[THEN "&E"(1), THEN "&E"(1)]

```

```

8567     "&I" by auto
8568   AOT_hence < $\exists y (A!a \ \& \ A!y \ \& \ a \neq y \ \& \ [\lambda [R]a] = [\lambda [R]y])$ > by (rule "EI")
8569   AOT_thus < $\exists x \exists y (A!x \ \& \ A!y \ \& \ x \neq y \ \& \ [\lambda [R]x] = [\lambda [R]y])$ > by (rule "EI")
8570 qed
8571
8572 AOT_theorem aclassical2: < $\exists x \exists y (A!x \ \& \ A!y \ \& \ x \neq y \ \& \ \forall F ([F]x \equiv [F]y))$ > (266)
8573 proof -
8574   AOT_have < $\exists x \exists y ([A!]x \ \& \ [A!]y \ \& \ x \neq y \ \& \ [\lambda z [\lambda xy \ \forall F ([F]x \equiv [F]y)]zx] =$ 
8575      $[\lambda z [\lambda xy \ \forall F ([F]x \equiv [F]y)]zy])$ >
8576     by (rule "aclassical:1"[THEN "\&E"(1)[where  $\tau = \llbracket \lambda xy \ \forall F ([F]x \equiv [F]y) \rrbracket$ ]])
8577     "cqt:2"
8578   then AOT_obtain x where < $\exists y ([A!]x \ \& \ [A!]y \ \& \ x \neq y \ \& \ [\lambda z [\lambda xy \ \forall F ([F]x \equiv [F]y)]zx] =$ 
8579      $[\lambda z [\lambda xy \ \forall F ([F]x \equiv [F]y)]zy])$ >
8580     using "\&E"[rotated] by blast
8581   then AOT_obtain y where 0: < $\langle [A!]x \ \& \ [A!]y \ \& \ x \neq y \ \& \ [\lambda z [\lambda xy \ \forall F ([F]x \equiv [F]y)]zx] =$ 
8582      $[\lambda z [\lambda xy \ \forall F ([F]x \equiv [F]y)]zy] \rangle$ >
8583     using "\&E"[rotated] by blast
8584   AOT_have < $[\lambda z [\lambda xy \ \forall F ([F]x \equiv [F]y)]zx]x$ >
8585     by (auto intro!: "\beta\leftarrow C"(1) "cqt:2";
8586         simp add: "&I" "ex:1:a" prod_denotesI "rule-ui:3"
8587             "oth-class-taut:3:a" "universal-cor")
8588   AOT_hence < $[\lambda z [\lambda xy \ \forall F ([F]x \equiv [F]y)]zy]x$ >
8589     by (rule "rule=E"[rotated, OF 0[THEN "\&E"(2)]])
8590   AOT_hence < $[\lambda xy \ \forall F ([F]x \equiv [F]y)]xy$ >
8591     by (rule "\beta\rightarrow C"(1))
8592   AOT_hence < $\forall F ([F]x \equiv [F]y)$ >
8593     using "\beta\rightarrow C"(1) old.prod.case by fast
8594   AOT_hence < $[A!]x \ \& \ [A!]y \ \& \ x \neq y \ \& \ \forall F ([F]x \equiv [F]y)$ >
8595     using 0 "\&E" "\&I" by blast
8596   AOT_hence < $\exists y ([A!]x \ \& \ [A!]y \ \& \ x \neq y \ \& \ \forall F ([F]x \equiv [F]y))$ > by (rule "EI")
8597   AOT_thus < $\exists x \exists y ([A!]x \ \& \ [A!]y \ \& \ x \neq y \ \& \ \forall F ([F]x \equiv [F]y))$ > by (rule "EI"(2))
8600 qed
8601
8602
8603 AOT_theorem "kirchner-thm:1": (268.1)
8604 < $[\lambda x \ \varphi\{x}] \downarrow \equiv \Box \forall x \forall y (\forall F ([F]x \equiv [F]y) \rightarrow (\varphi\{x} \equiv \varphi\{y}))$ >
8605 proof(rule "\equiv I"; rule "\rightarrow I")
8606   AOT_assume < $[\lambda x \ \varphi\{x}] \downarrow$ >
8607   AOT_hence < $\Box [\lambda x \ \varphi\{x}] \downarrow$ > by (metis "exist-nec" "vdash-properties:10")
8608   moreover AOT_have < $\Box [\lambda x \ \varphi\{x}] \downarrow \rightarrow \Box \forall x \forall y (\forall F ([F]x \equiv [F]y) \rightarrow (\varphi\{x} \equiv \varphi\{y}))$ >
8609   proof (rule "RM:1"; rule "\rightarrow I"; rule GEN; rule GEN; rule "\rightarrow I")
8610     AOT_modally_strict {
8611       fix x y
8612       AOT_assume 0: < $[\lambda x \ \varphi\{x}] \downarrow$ >
8613       moreover AOT_assume < $\forall F ([F]x \equiv [F]y)$ >
8614       ultimately AOT_have < $[\lambda x \ \varphi\{x}]x \equiv [\lambda x \ \varphi\{x}]y$ >
8615         using "\&E" by blast
8616       AOT_thus < $\langle \varphi\{x} \equiv \varphi\{y} \rangle$ >
8617         using "beta-C-meta"[THEN "\rightarrow E", OF 0] "\equiv E"(6) by meson
8618     }
8619   qed
8620   ultimately AOT_show < $\Box \forall x \forall y (\forall F ([F]x \equiv [F]y) \rightarrow (\varphi\{x} \equiv \varphi\{y}))$ >
8621   using "\rightarrow E" by blast
8622 next
8623 AOT_have < $\Box \forall x \forall y (\forall F ([F]x \equiv [F]y) \rightarrow (\varphi\{x} \equiv \varphi\{y})) \rightarrow$ 
8624    $\Box \forall y (\exists x (\forall F ([F]x \equiv [F]y) \ \& \ \varphi\{x}) \equiv \varphi\{y})$ >
8625 proof(rule "RM:1"; rule "\rightarrow I"; rule GEN)
8626   AOT_modally_strict {
8627     AOT_assume < $\Box \forall x \forall y (\forall F ([F]x \equiv [F]y) \rightarrow (\varphi\{x} \equiv \varphi\{y}))$ >
8628     AOT_hence indisc: < $\langle \varphi\{x} \equiv \varphi\{y} \rangle$  if < $\forall F ([F]x \equiv [F]y)$ > for x y>
8629     using "\&E"(2) "\rightarrow E" that by blast

```

```

8630   AOT_show <( $\exists x(\forall F([F]x \equiv [F]y) \ \& \ \varphi\{x\}) \equiv \varphi\{y\}$ )> for y
8631   proof (rule "raa-cor:1")
8632     AOT_assume < $\neg(\exists x(\forall F([F]x \equiv [F]y) \ \& \ \varphi\{x\}) \equiv \varphi\{y\})$ >
8633     AOT_hence < $(\exists x(\forall F([F]x \equiv [F]y) \ \& \ \varphi\{x\}) \ \& \ \neg\varphi\{y\}) \vee$   

8634             < $(\neg(\exists x(\forall F([F]x \equiv [F]y) \ \& \ \varphi\{x\})) \ \& \ \varphi\{y\})$ >
8635     using "≡E"(1) "oth-class-taut:4:h" by blast
8636     moreover {
8637       AOT_assume 0: < $\exists x(\forall F([F]x \equiv [F]y) \ \& \ \varphi\{x\}) \ \& \ \neg\varphi\{y\}$ >
8638       AOT_obtain a where < $\forall F([F]a \equiv [F]y) \ \& \ \varphi\{a\}$ >
8639       using "∃E"[rotated, OF 0[THEN "&E"(1)]] by blast
8640       AOT_hence < $\varphi\{y\}$ >
8641       using indisc[THEN "≡E"(1)] "&E" by blast
8642       AOT_hence <p & ¬p> for p
8643       using 0[THEN "&E"(2)] "&I" "raa-cor:3" by blast
8644     }
8645     moreover {
8646       AOT_assume 0: < $(\neg(\exists x(\forall F([F]x \equiv [F]y) \ \& \ \varphi\{x\})) \ \& \ \varphi\{y\})$ >
8647       AOT_hence < $\forall x \neg(\forall F([F]x \equiv [F]y) \ \& \ \varphi\{x\})$ >
8648       using "&E"(1) "cqt-further:4" "→E" by blast
8649       AOT_hence < $\neg(\forall F([F]y \equiv [F]y) \ \& \ \varphi\{y\})$ >
8650       using "∇E" by blast
8651       AOT_hence < $\neg\forall F([F]y \equiv [F]y) \vee \neg\varphi\{y\}$ >
8652       using "≡E"(1) "oth-class-taut:5:c" by blast
8653       moreover AOT_have < $\forall F([F]y \equiv [F]y)$ >
8654       by (simp add: "oth-class-taut:3:a" "universal-cor")
8655       ultimately AOT_have < $\neg\varphi\{y\}$ > by (metis "¬I" "∇E"(2))
8656       AOT_hence <p & ¬p> for p
8657       using 0[THEN "&E"(2)] "&I" "raa-cor:3" by blast
8658     }
8659     ultimately AOT_show <p & ¬p> for p
8660     using "∇E"(3) "raa-cor:1" by blast
8661   qed
8662 }
8663 qed
8664 moreover AOT_assume < $\Box\forall x\forall y(\forall F([F]x \equiv [F]y) \rightarrow (\varphi\{x\} \equiv \varphi\{y\}))$ >
8665 ultimately AOT_have < $\Box\forall y(\exists x(\forall F([F]x \equiv [F]y) \ \& \ \varphi\{x\}) \equiv \varphi\{y\})$ >
8666 using "→E" by blast
8667 AOT_thus < $[\lambda x \ \varphi\{x\}]\downarrow$ >
8668 by (rule "safe-ext"[axiom_inst, THEN "→E", OF "&I", rotated]) "cqt:2"
8669 qed
8670
8671 AOT_theorem "kirchner-thm:2":
8672 < $[\lambda x_1 \dots x_n \ \varphi\{x_1 \dots x_n\}]\downarrow \equiv \Box\forall x_1 \dots \forall x_n \forall y_1 \dots \forall y_n$   

8673 < $(\forall F([F]x_1 \dots x_n \equiv [F]y_1 \dots y_n) \rightarrow (\varphi\{x_1 \dots x_n\} \equiv \varphi\{y_1 \dots y_n\}))$ >
8674 proof(rule "≡I"; rule "→I")
8675   AOT_assume < $[\lambda x_1 \dots x_n \ \varphi\{x_1 \dots x_n\}]\downarrow$ >
8676   AOT_hence < $\Box[\lambda x_1 \dots x_n \ \varphi\{x_1 \dots x_n\}]\downarrow$ > by (metis "exist-nec" "→E")
8677   moreover AOT_have < $\Box[\lambda x_1 \dots x_n \ \varphi\{x_1 \dots x_n\}]\downarrow \rightarrow \Box\forall x_1 \dots \forall x_n \forall y_1 \dots \forall y_n$   

8678 < $(\forall F([F]x_1 \dots x_n \equiv [F]y_1 \dots y_n) \rightarrow (\varphi\{x_1 \dots x_n\} \equiv \varphi\{y_1 \dots y_n\}))$ >
8679   proof (rule "RM:1"; rule "→I"; rule GEN; rule GEN; rule "→I")
8680     AOT_modally_strict {
8681       fix x1xn y1yn :: <'a AOT_var>
8682       AOT_assume 0: < $[\lambda x_1 \dots x_n \ \varphi\{x_1 \dots x_n\}]\downarrow$ >
8683       moreover AOT_assume < $\forall F([F]x_1 \dots x_n \equiv [F]y_1 \dots y_n)$ >
8684       ultimately AOT_have < $[\lambda x_1 \dots x_n \ \varphi\{x_1 \dots x_n\}]x_1 \dots x_n \equiv$   

8685 < $[\lambda x_1 \dots x_n \ \varphi\{x_1 \dots x_n\}]y_1 \dots y_n$ >
8686       using "∇E" by blast
8687       AOT_thus < $(\varphi\{x_1 \dots x_n\} \equiv \varphi\{y_1 \dots y_n\})$ >
8688       using "beta-C-meta"[THEN "→E", OF 0] "≡E"(6) by meson
8689     }
8690   qed
8691   ultimately AOT_show < $\Box\forall x_1 \dots \forall x_n \forall y_1 \dots \forall y_n$   

8692 < $\forall F([F]x_1 \dots x_n \equiv [F]y_1 \dots y_n) \rightarrow (\varphi\{x_1 \dots x_n\} \equiv \varphi\{y_1 \dots y_n\})$ >

```

(268.2)

```

8693   )>
8694   using "→E" by blast
8695 next
8696   AOT_have <
8697     □(∀x1...∀xn∀y1...∀yn
8698       (∀F([F]x1...xn ≡ [F]y1...yn) → (φ{x1...xn} ≡ φ{y1...yn})))
8699   → □∀y1...∀yn
8700     ((∃x1...∃xn(∀F([F]x1...xn ≡ [F]y1...yn) & φ{x1...xn})) ≡
8701     φ{y1...yn})>
8702 proof(rule "RM:1"; rule "→I"; rule GEN)
8703   AOT_modally_strict {
8704     AOT_assume <∀x1...∀xn∀y1...∀yn
8705       (∀F([F]x1...xn ≡ [F]y1...yn) → (φ{x1...xn} ≡ φ{y1...yn}))>
8706     AOT_hence indisc: <φ{x1...xn} ≡ φ{y1...yn}>
8707     if <∀F([F]x1...xn ≡ [F]y1...yn)> for x1xn y1yn
8708     using "∇E"(2) "→E" that by blast
8709     AOT_show <(∃x1...∃xn(∀F([F]x1...xn ≡ [F]y1...yn) & φ{x1...xn})) ≡
8710       φ{y1...yn}> for y1yn
8711   proof (rule "raa-cor:1")
8712     AOT_assume <¬((∃x1...∃xn(∀F([F]x1...xn ≡ [F]y1...yn) & φ{x1...xn})) ≡
8713       φ{y1...yn})>
8714     AOT_hence <((∃x1...∃xn(∀F([F]x1...xn ≡ [F]y1...yn)
8715       & φ{x1...xn}))
8716       & ¬φ{y1...yn}) ∨
8717       (¬(∃x1...∃xn(∀F([F]x1...xn ≡ [F]y1...yn) & φ{x1...xn}))
8718       & φ{y1...yn})>
8719     using "≡E"(1) "oth-class-taut:4:h" by blast
8720   moreover {
8721     AOT_assume 0: <(∃x1...∃xn(∀F([F]x1...xn ≡ [F]y1...yn) & φ{x1...xn}))
8722       & ¬φ{y1...yn}>
8723     AOT_obtain a1an where <∀F([F]a1...an ≡ [F]y1...yn) & φ{a1...an}>
8724     using "∃E"[rotated, OF 0[THEN "&E"(1)]] by blast
8725     AOT_hence <φ{y1...yn}>
8726     using indisc[THEN "≡E"(1)] "&E" by blast
8727     AOT_hence <p & ¬p> for p
8728     using 0[THEN "&E"(2)] "&I" "raa-cor:3" by blast
8729   }
8730   moreover {
8731     AOT_assume 0: <¬(∃x1...∃xn(∀F([F]x1...xn ≡ [F]y1...yn) & φ{x1...xn}))
8732       & φ{y1...yn}>
8733     AOT_hence <∀x1...∀xn¬(∀F([F]x1...xn ≡ [F]y1...yn) & φ{x1...xn})>
8734     using "&E"(1) "cqt-further:4" "→E" by blast
8735     AOT_hence <¬(∀F([F]y1...yn ≡ [F]y1...yn) & φ{y1...yn})>
8736     using "∇E" by blast
8737     AOT_hence <¬∀F([F]y1...yn ≡ [F]y1...yn) ∨ ¬φ{y1...yn}>
8738     using "≡E"(1) "oth-class-taut:5:c" by blast
8739     moreover AOT_have <∀F([F]y1...yn ≡ [F]y1...yn)>
8740     by (simp add: "oth-class-taut:3:a" "universal-cor")
8741     ultimately AOT_have <¬φ{y1...yn}>
8742     by (metis "¬¬I" "∇E"(2))
8743     AOT_hence <p & ¬p> for p
8744     using 0[THEN "&E"(2)] "&I" "raa-cor:3" by blast
8745   }
8746     ultimately AOT_show <p & ¬p> for p
8747     using "∇E"(3) "raa-cor:1" by blast
8748   qed
8749 }
8750 qed
8751 moreover AOT_assume <□∀x1...∀xn∀y1...∀yn
8752   (∀F([F]x1...xn ≡ [F]y1...yn) → (φ{x1...xn} ≡ φ{y1...yn}))>
8753 ultimately AOT_have <□∀y1...∀yn
8754   ((∃x1...∃xn(∀F([F]x1...xn ≡ [F]y1...yn) & φ{x1...xn})) ≡
8755   φ{y1...yn})>

```

```

8756   using "→E" by blast
8757   AOT_thus <[λx1...xn φ{x1...xn}]↓>
8758   by (rule "safe-ext"[axiom_inst, THEN "→E", OF "&I", rotated]) "cqt:2"
8759 qed
8760
8761 AOT_theorem "kirchner-thm-cor:1": (269.1)
8762 <[λx φ{x}]↓ → ∀x∀y(∀F([F]x ≡ [F]y) → □(φ{x} ≡ φ{y}))>
8763 proof(rule "→I"; rule GEN; rule GEN; rule "→I")
8764   fix x y
8765   AOT_assume <[λx φ{x}]↓>
8766   AOT_hence <□∀x∀y (∀F ([F]x ≡ [F]y) → (φ{x} ≡ φ{y}))>
8767   by (rule "kirchner-thm:1"[THEN "≡E"(1)])
8768   AOT_hence <∀x□∀y (∀F ([F]x ≡ [F]y) → (φ{x} ≡ φ{y}))>
8769   using CBF[THEN "→E"] by blast
8770   AOT_hence <□∀y (∀F ([F]x ≡ [F]y) → (φ{x} ≡ φ{y}))>
8771   using "∀E" by blast
8772   AOT_hence <∀y □(∀F ([F]x ≡ [F]y) → (φ{x} ≡ φ{y}))>
8773   using CBF[THEN "→E"] by blast
8774   AOT_hence <□(∀F ([F]x ≡ [F]y) → (φ{x} ≡ φ{y}))>
8775   using "∀E" by blast
8776   AOT_hence <□∀F ([F]x ≡ [F]y) → □(φ{x} ≡ φ{y})>
8777   using "qml:1"[axiom_inst] "vdash-properties:6" by blast
8778   moreover AOT_assume <∀F([F]x ≡ [F]y)>
8779   ultimately AOT_show <□(φ{x} ≡ φ{y})> using "→E" "ind-nec" by blast
8780 qed
8781
8782 AOT_theorem "kirchner-thm-cor:2": (269.2)
8783 <[λx1...xn φ{x1...xn}]↓ → ∀x1...∀xn∀y1...∀yn
8784 (∀F([F]x1...xn ≡ [F]y1...yn}) → □(φ{x1...xn} ≡ φ{y1...yn}))>
8785 proof(rule "→I"; rule GEN; rule GEN; rule "→I")
8786   fix x1xn y1yn
8787   AOT_assume <[λx1...xn φ{x1...xn}]↓>
8788   AOT_hence 0: <□∀x1...∀xn∀y1...∀yn
8789 (∀F ([F]x1...xn ≡ [F]y1...yn}) → (φ{x1...xn} ≡ φ{y1...yn}))>
8790   by (rule "kirchner-thm:2"[THEN "≡E"(1)])
8791   AOT_have <∀x1...∀xn∀y1...∀yn
8792 □(∀F ([F]x1...xn ≡ [F]y1...yn}) → (φ{x1...xn} ≡ φ{y1...yn}))>
8793 proof(rule GEN; rule GEN)
8794   fix x1xn y1yn
8795   AOT_show <□(∀F ([F]x1...xn ≡ [F]y1...yn}) → (φ{x1...xn} ≡ φ{y1...yn}))>
8796   apply (rule "RM:1"[THEN "→E", rotated, OF 0]; rule "→I")
8797   using "∀E" by blast
8798 qed
8799   AOT_hence <∀y1...∀yn □(∀F ([F]x1...xn ≡ [F]y1...yn}) →
8800 (φ{x1...xn} ≡ φ{y1...yn}))>
8801   using "∀E" by blast
8802   AOT_hence <□(∀F ([F]x1...xn ≡ [F]y1...yn}) → (φ{x1...xn} ≡ φ{y1...yn}))>
8803   using "∀E" by blast
8804   AOT_hence <□(∀F ([F]x1...xn ≡ [F]y1...yn}) → (φ{x1...xn} ≡ φ{y1...yn}))>
8805   using "∀E" by blast
8806   AOT_hence 0: <□∀F ([F]x1...xn ≡ [F]y1...yn}) → □(φ{x1...xn} ≡ φ{y1...yn}))>
8807   using "qml:1"[axiom_inst] "vdash-properties:6" by blast
8808   moreover AOT_assume <∀F([F]x1...xn ≡ [F]y1...yn)>
8809   moreover AOT_have <[λx1...xn □∀F ([F]x1...xn ≡ [F]y1...yn)]↓> by "cqt:2"
8810   ultimately AOT_have <[λx1...xn □∀F ([F]x1...xn ≡ [F]y1...yn)]x1...xn ≡
8811 [λx1...xn □∀F ([F]x1...xn ≡ [F]y1...yn)]y1...yn>
8812   using "∀E" by blast
8813   moreover AOT_have <[λx1...xn □∀F ([F]x1...xn ≡ [F]y1...yn)]y1...yn>
8814   apply (rule "β←C"(1))
8815   apply "cqt:2[lambda]"
8816   apply (fact "cqt:2[const_var]"[axiom_inst])
8817   by (simp add: RN GEN "oth-class-taut:3:a")
8818   ultimately AOT_have <[λx1...xn □∀F ([F]x1...xn ≡ [F]y1...yn)]x1...xn>

```

```

8819   using "≡E"(2) by blast
8820   AOT_hence <□∀F ([F]x1...xn ≡ [F]y1...yn)>
8821   using "β→C"(1) by blast
8822   AOT_thus <□(φ{x1...xn} ≡ φ{y1...yn})> using "→E" 0 by blast
8823 qed
8824
8825 subsection<Propositional Properties>
8826 text<\label{PLM: 9.12}>
8827
8828 AOT_define propositional :: <Π ⇒ φ> (<Propositional'('_)>)
8829   "prop-prop1": <Propositional([F]) ≡df ∃p(F = [λy p])> (270)
8830
8831 AOT_theorem "prop-prop2:1": <∀p [λy p]↓> (271.1)
8832   by (rule GEN) "cqt:2[lambda]"
8833
8834 AOT_theorem "prop-prop2:2": <[λν φ]↓> (271.2)
8835   by "cqt:2[lambda]"
8836
8837 AOT_theorem "prop-prop2:3": <F = [λy p] → □∀x([F]x ≡ p)> (271.3)
8838 proof (rule "→I")
8839   AOT_assume 0: <F = [λy p]>
8840   AOT_show <□∀x([F]x ≡ p)>
8841     by (rule "rule=E"[rotated, OF 0[symmetric]]);
8842       rule RN; rule GEN; rule "beta-C-meta"[THEN "→E"])
8843       "cqt:2[lambda]"
8844 qed
8845
8846 AOT_theorem "prop-prop2:4": <Propositional([F]) → □Propositional([F])> (271.4)
8847 proof(rule "→I")
8848   AOT_assume <Propositional([F])>
8849   AOT_hence <∃p(F = [λy p])>
8850     using "≡dfE"[OF "prop-prop1"] by blast
8851   then AOT_obtain p where <F = [λy p]>
8852     using "∃E"[rotated] by blast
8853   AOT_hence <□(F = [λy p])>
8854     using "id-nec:2" "modus-tollens:1" "raa-cor:3" by blast
8855   AOT_hence <∃p □(F = [λy p])>
8856     using "∃I" by fast
8857   AOT_hence 0: <□∃p (F = [λy p])>
8858     by (metis Buridan "vdash-properties:10")
8859   AOT_thus <□Propositional([F])>
8860     using "prop-prop1"[THEN "≡dfI"]
8861     by (AOT_subst <Propositional([F])> <∃p (F = [λy p])>) auto
8862 qed
8863
8864 AOT_define indiscriminate :: <Π ⇒ φ> ("Indiscriminate'('_)")
8865   "prop-indis": <Indiscriminate([F]) ≡df F↓ & □(∃x [F]x → ∀x [F]x)> (272)
8866
8867 AOT_theorem "prop-in-thm": <Propositional([II]) → Indiscriminate([II])> (273)
8868 proof(rule "→I")
8869   AOT_assume <Propositional([II])>
8870   AOT_hence <∃p II = [λy p]> using "≡dfE"[OF "prop-prop1"] by blast
8871   then AOT_obtain p where II_def: <II = [λy p]> using "∃E"[rotated] by blast
8872   AOT_show <Indiscriminate([II])>
8873   proof (rule "≡dfI"[OF "prop-indis"]; rule "&I")
8874     AOT_show <II↓>
8875     using II_def by (meson "t=t-proper:1" "vdash-properties:6")
8876   next
8877     AOT_show <□(∃x [II]x → ∀x [II]x)>
8878     proof (rule "rule=E"[rotated, OF II_def[symmetric]]);
8879       rule RN; rule "→I"; rule GEN)
8880     AOT_modally_strict {
8881       AOT_assume <∃x [λy p]x>

```



```

8882     then AOT_obtain a where <[ $\lambda y p$ ]a> using "E" [rotated] by blast
8883     AOT_hence 0: <p> by (metis " $\beta \rightarrow C$ "(1))
8884     AOT_show <[ $\lambda y p$ ]x> for x
8885         apply (rule " $\beta \leftarrow C$ "(1))
8886         apply "cqt:2[lambda]"
8887         apply (fact "cqt:2[const_var]" [axiom_inst])
8888         by (fact 0)
8889     }
8890     qed
8891   qed
8892 qed
8893
8894 AOT_theorem "prop-in-f:1": <Necessary([F])  $\rightarrow$  Indiscriminate([F])> (274.1)
8895 proof (rule " $\rightarrow I$ ")
8896   AOT_assume <Necessary([F])>
8897   AOT_hence 0: < $\Box \forall x_1 \dots \forall x_n [F] x_1 \dots x_n$ >
8898     using " $\equiv_{df} E$ " [OF "contingent-properties:1"] by blast
8899   AOT_show <Indiscriminate([F])>
8900     by (rule " $\equiv_{df} I$ " [OF "prop-indis"])
8901     (metis "0" "KBasic:1" "&I" "ex:1:a" "rule-ui:2[const_var]" " $\rightarrow E$ ")
8902 qed
8903
8904 AOT_theorem "prop-in-f:2": <Impossible([F])  $\rightarrow$  Indiscriminate([F])> (274.2)
8905 proof (rule " $\rightarrow I$ ")
8906   AOT_modally_strict {
8907     AOT_have < $\forall x \neg [F] x \rightarrow (\exists x [F] x \rightarrow \forall x [F] x)$ >
8908       by (metis "E" "cqt-orig:3" "Hypothetical Syllogism" " $\rightarrow I$ " "raa-cor:3")
8909   }
8910   AOT_hence 0: < $\Box \forall x \neg [F] x \rightarrow \Box (\exists x [F] x \rightarrow \forall x [F] x)$ >
8911     by (rule "RM:1")
8912   AOT_assume <Impossible([F])>
8913   AOT_hence < $\Box \forall x \neg [F] x$ >
8914     using " $\equiv_{df} E$ " [OF "contingent-properties:2"] "&E" by blast
8915   AOT_hence 1: < $\Box (\exists x [F] x \rightarrow \forall x [F] x)$ >
8916     using 0 " $\rightarrow E$ " by blast
8917   AOT_show <Indiscriminate([F])>
8918     by (rule " $\equiv_{df} I$ " [OF "prop-indis"]; rule "&I")
8919     (simp add: "ex:1:a" "rule-ui:2[const_var]" 1)+
8920 qed
8921
8922 AOT_theorem "prop-in-f:3:a": < $\neg$ Indiscriminate([E!])> (274.3.a)
8923 proof (rule "raa-cor:2")
8924   AOT_assume <Indiscriminate([E!])>
8925   AOT_hence 0: < $\Box (\exists x [E!] x \rightarrow \forall x [E!] x)$ >
8926     using " $\equiv_{df} E$ " [OF "prop-indis"] "&E" by blast
8927   AOT_hence < $\Diamond \exists x [E!] x \rightarrow \Diamond \forall x [E!] x$ >
8928     using "KBasic:13" "vdash-properties:10" by blast
8929   moreover AOT_have < $\Diamond \exists x [E!] x$ >
8930     by (simp add: "thm-cont-e:3")
8931   ultimately AOT_have < $\Diamond \forall x [E!] x$ >
8932     by (metis "vdash-properties:6")
8933   AOT_thus <p &  $\neg$ p> for p
8934     by (metis " $\equiv_{df} E$ " "conventions:5" "o-objects-exist:5" "reductio-aa:1")
8935 qed
8936
8937 AOT_theorem "prop-in-f:3:b": < $\neg$ Indiscriminate([E!]~)> (274.3.b)
8938 proof (rule "rule=E" [rotated, OF "rel-neg-T:2" [symmetric]];
8939   rule "raa-cor:2")
8940   AOT_assume <Indiscriminate([ $\lambda x \neg [E!] x$ ])>
8941   AOT_hence 0: < $\Box (\exists x [\lambda x \neg [E!] x] x \rightarrow \forall x [\lambda x \neg [E!] x] x)$ >
8942     using " $\equiv_{df} E$ " [OF "prop-indis"] "&E" by blast
8943   AOT_hence < $\Box \exists x [\lambda x \neg [E!] x] x \rightarrow \Box \forall x [\lambda x \neg [E!] x] x$ >
8944     using " $\rightarrow E$ " "qml:1" "vdash-properties:1[2]" by blast

```



```

8945 moreover AOT_have <□∃x [λx ¬[E!]x]x>
8946   apply (AOT_subst <[λx ¬E!x]x> <¬E!x> for: x)
8947   apply (rule "beta-C-meta"[THEN "→E"])
8948   apply "cqt:2"
8949   by (metis (full_types) "B◇" RN "T◇" "cqt-further:2"
8950       "o-objects-exist:5" "→E")
8951 ultimately AOT_have 1: <□∀x [λx ¬[E!]x]x>
8952   by (metis "vdash-properties:6")
8953 AOT_hence <□∀x ¬[E!]x>
8954   by (AOT_subst (reverse) <¬[E!]x> <[λx ¬[E!]x]x> for: x)
8955   (auto intro!: "cqt:2" "beta-C-meta"[THEN "→E"])
8956 AOT_hence <∀x □¬[E!]x> by (metis "CBF" "vdash-properties:10")
8957 moreover AOT_obtain a where abs_a: <O!a>
8958   using "∃E" "o-objects-exist:1" "qml:2"[axiom_inst] "→E" by blast
8959 ultimately AOT_have <□¬[E!]a> using "∀E" by blast
8960 AOT_hence 2: <¬◇[E!]a> by (metis "≡dfE" "conventions:5" "reductio-aa:1")
8961 AOT_have <A!a>
8962   apply (rule "=dfI"(2)[OF AOT_abstract])
8963   apply "cqt:2[lambda]"
8964   apply (rule "β←C"(1))
8965   apply "cqt:2[lambda]"
8966   using "cqt:2[const_var]"[axiom_inst] apply blast
8967   by (fact 2)
8968 AOT_thus <p & ¬p> for p using abs_a
8969   by (metis "≡E"(1) "oa-contingent:2" "reductio-aa:1")
8970 qed
8971
8972 AOT_theorem "prop-in-f:3:c": <¬Indiscriminate(O!)> (274.3.c)
8973 proof(rule "raa-cor:2")
8974   AOT_assume <Indiscriminate(O!)>
8975   AOT_hence 0: <□(∃x O!x → ∀x O!x)>
8976     using "≡dfE"[OF "prop-indis"] "&E" by blast
8977   AOT_hence <□∃x O!x → □∀x O!x>
8978     using "qml:1"[axiom_inst] "vdash-properties:6" by blast
8979   moreover AOT_have <□∃x O!x>
8980     using "o-objects-exist:1" by blast
8981   ultimately AOT_have <□∀x O!x>
8982     by (metis "vdash-properties:6")
8983   AOT_thus <p & ¬p> for p
8984     by (metis "o-objects-exist:3" "qml:2"[axiom_inst] "raa-cor:3" "→E")
8985 qed
8986
8987 AOT_theorem "prop-in-f:3:d": <¬Indiscriminate(A!)> (274.3.d)
8988 proof(rule "raa-cor:2")
8989   AOT_assume <Indiscriminate(A!)>
8990   AOT_hence 0: <□(∃x A!x → ∀x A!x)>
8991     using "≡dfE"[OF "prop-indis"] "&E" by blast
8992   AOT_hence <□∃x A!x → □∀x A!x>
8993     using "qml:1"[axiom_inst] "vdash-properties:6" by blast
8994   moreover AOT_have <□∃x A!x>
8995     using "o-objects-exist:2" by blast
8996   ultimately AOT_have <□∀x A!x>
8997     by (metis "vdash-properties:6")
8998   AOT_thus <p & ¬p> for p
8999     by (metis "o-objects-exist:4" "qml:2"[axiom_inst] "raa-cor:3" "→E")
9000 qed
9001
9002 AOT_theorem "prop-in-f:4:a": <¬Propositional(E!)> (274.4.a)
9003   using "modus-tollens:1" "prop-in-f:3:a" "prop-in-thm" by blast
9004
9005 AOT_theorem "prop-in-f:4:b": <¬Propositional(E!')> (274.4.b)
9006   using "modus-tollens:1" "prop-in-f:3:b" "prop-in-thm" by blast
9007

```

```

9008 AOT_theorem "prop-in-f:4:c": <¬Propositional(O!)> (274.4.c)
9009   using "modus-tollens:1" "prop-in-f:3:c" "prop-in-thm" by blast
9010
9011 AOT_theorem "prop-in-f:4:d": <¬Propositional(A!)> (274.4.d)
9012   using "modus-tollens:1" "prop-in-f:3:d" "prop-in-thm" by blast
9013
9014 AOT_theorem "prop-prop-nec:1": <◇∃p (F = [λy p]) → ∃p(F = [λy p])> (275.1)
9015 proof(rule "→I")
9016   AOT_assume <◇∃p (F = [λy p])>
9017   AOT_hence <∃p ◇(F = [λy p])>
9018     by (metis "BF◇" "→E")
9019   then AOT_obtain p where <◇(F = [λy p])>
9020     using "∃E"[rotated] by blast
9021   AOT_hence <F = [λy p]>
9022     by (metis "derived-S5-rules:2" emptyE "id-nec:2" "→E")
9023   AOT_thus <∃p(F = [λy p])> by (rule "∃I")
9024 qed
9025
9026 AOT_theorem "prop-prop-nec:2": <∀p (F ≠ [λy p]) → □∀p(F ≠ [λy p])> (275.2)
9027 proof(rule "→I")
9028   AOT_assume <∀p (F ≠ [λy p])>
9029   AOT_hence <(F ≠ [λy p])> for p
9030     using "∀E" by blast
9031   AOT_hence <□(F ≠ [λy p])> for p
9032     by (rule "id-nec2:2"[unvarify β, THEN "→E", rotated]) "cqt:2"
9033   AOT_hence <∀p □(F ≠ [λy p])> by (rule GEN)
9034   AOT_thus <□∀p (F ≠ [λy p])> using BF[THEN "→E"] by fast
9035 qed
9036
9037 AOT_theorem "prop-prop-nec:3": <∃p (F = [λy p]) → □∃p(F = [λy p])> (275.3)
9038 proof(rule "→I")
9039   AOT_assume <∃p (F = [λy p])>
9040   then AOT_obtain p where <(F = [λy p])> using "∃E"[rotated] by blast
9041   AOT_hence <□(F = [λy p])> by (metis "id-nec:2" "→E")
9042   AOT_hence <∃p□(F = [λy p])> by (rule "∃I")
9043   AOT_thus <□∃p(F = [λy p])> by (metis Buridan "→E")
9044 qed
9045
9046 AOT_theorem "prop-prop-nec:4": <◇∀p (F ≠ [λy p]) → ∀p(F ≠ [λy p])> (275.4)
9047 proof(rule "→I")
9048   AOT_assume <◇∀p (F ≠ [λy p])>
9049   AOT_hence <∀p ◇(F ≠ [λy p])> by (metis "Buridan◇" "→E")
9050   AOT_hence <◇(F ≠ [λy p])> for p
9051     using "∀E" by blast
9052   AOT_hence <F ≠ [λy p]> for p
9053     by (rule "id-nec2:3"[unvarify β, THEN "→E", rotated]) "cqt:2"
9054   AOT_thus <∀p (F ≠ [λy p])> by (rule GEN)
9055 qed
9056
9057 AOT_theorem "enc-prop-nec:1": (276.1)
9058   <◇∀F (x[F] → ∃p(F = [λy p])) → ∀F(x[F] → ∃p (F = [λy p]))>
9059 proof(rule "→I"; rule GEN; rule "→I")
9060   fix F
9061   AOT_assume <◇∀F (x[F] → ∃p(F = [λy p]))>
9062   AOT_hence <∀F ◇(x[F] → ∃p(F = [λy p]))>
9063     using "Buridan◇" "vdash-properties:10" by blast
9064   AOT_hence 0: <◇(x[F] → ∃p(F = [λy p]))> using "∀E" by blast
9065   AOT_assume <x[F]>
9066   AOT_hence <□x[F]> by (metis "en-eq:2[1]" "≡E"(1))
9067   AOT_hence <◇∃p(F = [λy p])>
9068     using 0 by (metis "KBasic2:4" "≡E"(1) "vdash-properties:10")
9069   AOT_thus <∃p(F = [λy p])>
9070   using "prop-prop-nec:1"[THEN "→E"] by blast

```

```
9071 qed
9072
9073 AOT_theorem "enc-prop-nec:2": (276.2)
9074 < $\forall F (x[F] \rightarrow \exists p (F = [\lambda y p])) \rightarrow \Box \forall F (x[F] \rightarrow \exists p (F = [\lambda y p]))$ >
9075 using "derived-S5-rules:1"[where  $\Gamma = \{\}$ ", simplified, OF "enc-prop-nec:1"]
9076 by blast
9077
9078 (*<*)
9079 end
9080 (*>*)
```

A.8. Basic Logical Objects

```

1  (*<*)
2  theory AOT_BasicLogicalObjects
3    imports AOT_PLM
4  begin
5  (*>*)
6
7  section<Basic Logical Objects>
8  (* Note: so far only the parts required for possible world theory are implemented *)
9
10 AOT_define TruthValueOf :: < $\tau \Rightarrow \varphi \Rightarrow \varphi$ > (<TruthValueOf'(_,_)>)
11   "tv-p": <TruthValueOf(x,p)  $\equiv_{df}$   $A!x \ \& \ \forall F (x[F] \equiv \exists q((q \equiv p) \ \& \ F = [\lambda y \ q]))$ > (281)
12
13 AOT_theorem "p-has-!tv:1": < $\exists x \ \text{TruthValueOf}(x,p)$ > (283.1)
14   using "tv-p"[THEN "≡Df"]
15   by (AOT_subst <TruthValueOf(x,p)>
16       < $A!x \ \& \ \forall F (x[F] \equiv \exists q((q \equiv p) \ \& \ F = [\lambda y \ q]))$ > for: x)
17   (simp add: "A-objects"[axiom_inst])
18
19
20 AOT_theorem "p-has-!tv:2": < $\exists!x \ \text{TruthValueOf}(x,p)$ > (283.2)
21   using "tv-p"[THEN "≡Df"]
22   by (AOT_subst <TruthValueOf(x,p)>
23       < $A!x \ \& \ \forall F (x[F] \equiv \exists q((q \equiv p) \ \& \ F = [\lambda y \ q]))$ > for: x)
24   (simp add: "A-objects!")
25
26
27 AOT_theorem "uni-tv": < $\iota x \ \text{TruthValueOf}(x,p) \downarrow$ > (284)
28   using "A-Exists:2" "RA[2]" "≡E"(2) "p-has-!tv:2" by blast
29
30 AOT_define TheTruthValueOf :: < $\varphi \Rightarrow \kappa_g$ > (<_> [100] 100)
31   "the-tv-p": < $op_{=df} \ \iota x \ \text{TruthValueOf}(x,p)$ > (285)
32
33 AOT_define PropEnc :: < $\tau \Rightarrow \varphi \Rightarrow \varphi$ > (infix1 < $\Sigma$ > 40)
34   "prop-enc": < $x\Sigma p \equiv_{df} x \downarrow \ \& \ x[\lambda y \ p]$ > (286)
35
36 AOT_theorem "tv-id:1": < $op = \ \iota x (A!x \ \& \ \forall F (x[F] \equiv \exists q((q \equiv p) \ \& \ F = [\lambda y \ q])))$ > (287.1)
37 proof -
38   AOT_have < $\Box \forall x(\text{TruthValueOf}(x,p) \equiv A!x \ \& \ \forall F (x[F] \equiv \exists q((q \equiv p) \ \& \ F = [\lambda y \ q])))$ >
39     by (rule RN; rule GEN; rule "tv-p"[THEN "≡Df"])
40   AOT_hence < $\iota x \ \text{TruthValueOf}(x,p) = \iota x (A!x \ \& \ \forall F (x[F] \equiv \exists q((q \equiv p) \ \& \ F = [\lambda y \ q])))$ >
41     using "equiv-desc-eq:3"[THEN "→E", OF "&I", OF "uni-tv"] by simp
42   thus ?thesis
43     using "=dfI"(1)[OF "the-tv-p", OF "uni-tv"] by fast
44 qed
45
46 AOT_theorem "tv-id:2": < $op\Sigma p$ > (287.2)
47 proof -
48   AOT_modally_strict {
49     AOT_have < $(p \equiv p) \ \& \ [\lambda y \ p] = [\lambda y \ p]$ >
50       by (auto simp: "prop-prop2:2" "rule=I:1" intro!: "≡I" "→I" "&I")
51     AOT_hence < $\exists q ((q \equiv p) \ \& \ [\lambda y \ p] = [\lambda y \ q])$ >
52       using "≡I" by fast
53   }
54   AOT_hence < $\mathcal{A}\exists q ((q \equiv p) \ \& \ [\lambda y \ p] = [\lambda y \ q])$ >
55     using "RA[2]" by blast
56   AOT_hence < $\iota x(A!x \ \& \ \forall F (x[F] \equiv \exists q ((q \equiv p) \ \& \ F = [\lambda y \ q])))[\lambda y \ p]$ >
57     by (safe intro!: "desc-nec-encode:1"[unvarify F, THEN "≡E"(2)] "cqt:2")
58   AOT_hence < $\iota x(A!x \ \& \ \forall F (x[F] \equiv \exists q ((q \equiv p) \ \& \ F = [\lambda y \ q])))\Sigma p$ >
59     by (safe intro!: "prop-enc"[THEN "=dfI"] "&I" "A-descriptions")
60   AOT_thus < $op\Sigma p$ >
61     by (rule "rule=E"[rotated, OF "tv-id:1"[symmetric]])

```

```

62 qed
63
64 (* TODO more theorems *)
65
66 AOT_theorem "TV-lem1:1": (292.1)
67   <p  $\equiv \forall F(\exists q (q \ \& \ F = [\lambda y \ q]) \equiv \exists q((q \equiv p) \ \& \ F = [\lambda y \ q]))$ >
68 proof(safe intro!: "≡I" "→I" GEN)
69   fix F
70   AOT_assume < $\exists q (q \ \& \ F = [\lambda y \ q])$ >
71   then AOT_obtain q where <q & F = [λy q]> using "∃E"[rotated] by blast
72   moreover AOT_assume p
73   ultimately AOT_have <(q ≡ p) & F = [λy q]>
74     by (metis "&I" "&E"(1) "&E"(2) "deduction-theorem" "≡I")
75   AOT_thus < $\exists q ((q \equiv p) \ \& \ F = [\lambda y \ q])$ > by (rule "∃I")
76 next
77   fix F
78   AOT_assume < $\exists q ((q \equiv p) \ \& \ F = [\lambda y \ q])$ >
79   then AOT_obtain q where <(q ≡ p) & F = [λy q]> using "∃E"[rotated] by blast
80   moreover AOT_assume p
81   ultimately AOT_have <q & F = [λy q]>
82     by (metis "&I" "&E"(1) "&E"(2) "≡E"(2))
83   AOT_thus < $\exists q (q \ \& \ F = [\lambda y \ q])$ > by (rule "∃I")
84 next
85   AOT_assume < $\forall F (\exists q (q \ \& \ F = [\lambda y \ q]) \equiv \exists q ((q \equiv p) \ \& \ F = [\lambda y \ q]))$ >
86   AOT_hence < $\exists q (q \ \& \ [\lambda y \ p] = [\lambda y \ q]) \equiv \exists q ((q \equiv p) \ \& \ [\lambda y \ p] = [\lambda y \ q])$ >
87     using "∀E"(1)[rotated, OF "prop-prop2:2"] by blast
88   moreover AOT_have < $\exists q ((q \equiv p) \ \& \ [\lambda y \ p] = [\lambda y \ q])$ >
89     by (rule "∃I"(2)[where β=p])
90     (simp add: "rule=I:1" "&I" "oth-class-taut:3:a" "prop-prop2:2")
91   ultimately AOT_have < $\exists q (q \ \& \ [\lambda y \ p] = [\lambda y \ q])$ > using "≡E"(2) by blast
92   then AOT_obtain q where <q & [λy p] = [λy q]> using "∃E"[rotated] by blast
93   AOT_thus <p>
94     using "rule=E" "&E"(1) "&E"(2) id_sym "≡E"(2) "p-identity-thm2:3" by fast
95 qed
96
97 AOT_theorem "TV-lem1:2": (292.2)
98   < $\neg p \equiv \forall F(\exists q (\neg q \ \& \ F = [\lambda y \ q]) \equiv \exists q((q \equiv p) \ \& \ F = [\lambda y \ q]))$ >
99 proof(safe intro!: "≡I" "→I" GEN)
100   fix F
101   AOT_assume < $\exists q (\neg q \ \& \ F = [\lambda y \ q])$ >
102   then AOT_obtain q where < $\neg q \ \& \ F = [\lambda y \ q]$ > using "∃E"[rotated] by blast
103   moreover AOT_assume < $\neg p$ >
104   ultimately AOT_have <(q ≡ p) & F = [λy q]>
105     by (metis "&I" "&E"(1) "&E"(2) "deduction-theorem" "≡I" "raa-cor:3")
106   AOT_thus < $\exists q ((q \equiv p) \ \& \ F = [\lambda y \ q])$ > by (rule "∃I")
107 next
108   fix F
109   AOT_assume < $\exists q ((q \equiv p) \ \& \ F = [\lambda y \ q])$ >
110   then AOT_obtain q where <(q ≡ p) & F = [λy q]> using "∃E"[rotated] by blast
111   moreover AOT_assume < $\neg p$ >
112   ultimately AOT_have < $\neg q \ \& \ F = [\lambda y \ q]$ >
113     by (metis "&I" "&E"(1) "&E"(2) "≡E"(1) "raa-cor:3")
114   AOT_thus < $\exists q (\neg q \ \& \ F = [\lambda y \ q])$ > by (rule "∃I")
115 next
116   AOT_assume < $\forall F (\exists q (\neg q \ \& \ F = [\lambda y \ q]) \equiv \exists q ((q \equiv p) \ \& \ F = [\lambda y \ q]))$ >
117   AOT_hence < $\exists q (\neg q \ \& \ [\lambda y \ p] = [\lambda y \ q]) \equiv \exists q ((q \equiv p) \ \& \ [\lambda y \ p] = [\lambda y \ q])$ >
118     using "∀E"(1)[rotated, OF "prop-prop2:2"] by blast
119   moreover AOT_have < $\exists q ((q \equiv p) \ \& \ [\lambda y \ p] = [\lambda y \ q])$ >
120     by (rule "∃I"(2)[where β=p])
121     (simp add: "rule=I:1" "&I" "oth-class-taut:3:a" "prop-prop2:2")
122   ultimately AOT_have < $\exists q (\neg q \ \& \ [\lambda y \ p] = [\lambda y \ q])$ > using "≡E"(2) by blast
123   then AOT_obtain q where < $\neg q \ \& \ [\lambda y \ p] = [\lambda y \ q]$ > using "∃E"[rotated] by blast
124   AOT_thus < $\neg p$ >

```

```

125     using "rule=E" "&E"(1) "&E"(2) id_sym "≡E"(2) "p-identity-thm2:3" by fast
126 qed
127
128
129 AOT_define TruthValue :: <τ ⇒ φ> (<TruthValue'('_)>)
130   "T-value": <TruthValue(x) ≡df ∃p (TruthValueOf(x,p))>
131
132 (* TODO more theorems *)
133
134 AOT_act_theorem "T-lem:1": <TruthValueOf(op, p)>
135 proof -
136   AOT_have ϑ: <op = ιx TruthValueOf(x, p)>
137     using "rule-id-df:1" "the-tv-p" "uni-tv" by blast
138   moreover AOT_have <op↓>
139     using "t=t-proper:1" calculation "vdash-properties:10" by blast
140   ultimately show ?thesis by (metis "rule=E" id_sym "vdash-properties:10" "y-in:3")
141 qed
142
143 AOT_act_theorem "T-lem:2": <∀F (op[F] ≡ ∃q((q ≡ p) & F = [λy q]))>
144   using "T-lem:1"[THEN "tv-p"[THEN "≡dfE"], THEN "&E"(2)].
145
146 AOT_act_theorem "T-lem:3": <opΣr ≡ (r ≡ p)>
147 proof -
148   AOT_have ϑ: <op[λy r] ≡ ∃q ((q ≡ p) & [λy r] = [λy q])>
149     using "T-lem:2"[THEN "∀E"(1), OF "prop-prop2:2"].
150   show ?thesis
151   proof(rule "≡I"; rule "→I")
152     AOT_assume <opΣr>
153     AOT_hence <op[λy r]> by (metis "≡dfE" "&E"(2) "prop-enc")
154     AOT_hence <∃q ((q ≡ p) & [λy r] = [λy q])> using ϑ "≡E"(1) by blast
155     then AOT_obtain q where <(q ≡ p) & [λy r] = [λy q]> using "∃E"[rotated] by blast
156     moreover AOT_have <r = q> using calculation
157       using "&E"(2) "≡E"(2) "p-identity-thm2:3" by blast
158     ultimately AOT_show <r ≡ p>
159       by (metis "rule=E" "&E"(1) "≡E"(6) "oth-class-taut:3:a")
160   next
161     AOT_assume <r ≡ p>
162     moreover AOT_have <[λy r] = [λy r]>
163       by (simp add: "rule=I:1" "prop-prop2:2")
164     ultimately AOT_have <(r ≡ p) & [λy r] = [λy r]> using "&I" by blast
165     AOT_hence <∃q ((q ≡ p) & [λy r] = [λy q])> by (rule "∃I"(2)[where β=r])
166     AOT_hence <op[λy r]> using ϑ "≡E"(2) by blast
167     AOT_thus <opΣr>
168       by (metis "≡dfI" "&I" "prop-enc" "russell-axiom[enc,1].ψ_denotes_asm")
169   qed
170 qed
171
172 AOT_act_theorem "T-lem:4": <TruthValueOf(x, p) ≡ x = op>
173 proof -
174   AOT_have <∀x (x = ιx TruthValueOf(x, p) ≡ ∀z (TruthValueOf(z, p) ≡ z = x))>
175     by (simp add: "fund-cont-desc" GEN)
176   moreover AOT_have <op↓>
177     using "≡dfE" "tv-id:2" "&E"(1) "prop-enc" by blast
178   ultimately AOT_have
179     <(op = ιx TruthValueOf(x, p)) ≡ ∀z (TruthValueOf(z, p) ≡ z = op)>
180     using "∀E"(1) by blast
181   AOT_hence <∀z (TruthValueOf(z, p) ≡ z = op)>
182     using "≡E"(1) "rule-id-df:1" "the-tv-p" "uni-tv" by blast
183   AOT_thus <TruthValueOf(x, p) ≡ x = op> using "∀E"(2) by blast
184 qed
185
186
187 (* TODO more theorems *)

```

```

188
189 AOT_theorem "TV-lem2:1": (295.1)
190   <(A!x & ∀F (x[F] ≡ ∃q (q & F = [λy q]))) → TruthValue(x)>
191 proof(safe intro!: "→I" "T-value"[THEN "≡df I"] "tv-p"[THEN "≡df I"]
192       "∃I"(1)[rotated, OF "log-prop-prop:2"])
193   AOT_assume <[A!]x & ∀F (x[F] ≡ ∃q (q & F = [λy q]))>
194   AOT_thus <[A!]x & ∀F (x[F] ≡ ∃q ((q ≡ (∀p (p → p))) & F = [λy q]))>
195     apply (AOT_subst <∃q ((q ≡ (∀p (p → p))) & F = [λy q])>
196           <∃q (q & F = [λy q])> for: F :: <<κ>>)
197     apply (AOT_subst <q ≡ ∀p (p → p)> <q> for: q)
198     apply (metis (no_types, lifting) "→I" "≡I" "≡E"(2) GEN)
199     by (auto simp: "cqt-further:7")
200 qed
201
202 AOT_theorem "TV-lem2:2": (295.2)
203   <(A!x & ∀F (x[F] ≡ ∃q (¬q & F = [λy q]))> → TruthValue(x)>
204 proof(safe intro!: "→I" "T-value"[THEN "≡df I"] "tv-p"[THEN "≡df I"]
205       "∃I"(1)[rotated, OF "log-prop-prop:2"])
206   AOT_assume <[A!]x & ∀F (x[F] ≡ ∃q (¬q & F = [λy q]))>
207   AOT_thus <[A!]x & ∀F (x[F] ≡ ∃q ((q ≡ (∃p (p & ¬p))) & F = [λy q]))>
208     apply (AOT_subst <∃q ((q ≡ (∃p (p & ¬p))) & F = [λy q])>
209           <∃q (¬q & F = [λy q])> for: F :: <<κ>>)
210     apply (AOT_subst <q ≡ ∃p (p & ¬p)> <¬q> for: q)
211     apply (metis (no_types, lifting)
212           "→I" "≡E" "≡E"(1) "≡I" "raa-cor:1" "raa-cor:3")
213     by (auto simp add: "cqt-further:7")
214 qed
215
216 AOT_define TheTrue :: κs (<⊤>)
217   "the-true:1": <⊤ =df λx (A!x & ∀F (x[F] ≡ ∃p (p & F = [λy p])))> (296.1)
218 AOT_define TheFalse :: κs (<⊥>)
219   "the-true:2": <⊥ =df λx (A!x & ∀F (x[F] ≡ ∃p (¬p & F = [λy p])))> (296.2)
220
221
222 AOT_theorem "the-true:3": <⊤ ≠ ⊥> (296.3)
223 proof(safe intro!: "ab-obey:2"[unvarify x y, THEN "→E", rotated 2, OF "∀I"(1)]
224       "∃I"(1)[where τ=«[λx ∀q(q → q)]»] "&I" "prop-prop2:2")
225   AOT_have false_def: <⊥ = λx (A!x & ∀F (x[F] ≡ ∃p (¬p & F = [λy p])))>
226     by (simp add: "A-descriptions" "rule-id-df:1[zero]" "the-true:2")
227   moreover AOT_show false_den: <⊥↓>
228     by (meson "→E" "t=t-proper:1" "A-descriptions"
229           "rule-id-df:1[zero]" "the-true:2")
230   ultimately AOT_have false_prop: <⊥(A!⊥ & ∀F (⊥[F] ≡ ∃p (¬p & F = [λy p])))>
231     using "nec-hintikka-scheme"[unvarify x, THEN "≡E"(1), THEN "&E"(1)] by blast
232   AOT_hence <⊥∀F (⊥[F] ≡ ∃p (¬p & F = [λy p]))>
233     using "Act-Basic:2" "&E"(2) "≡E"(1) by blast
234   AOT_hence <⊥∀F ⊥(⊥[F] ≡ ∃p (¬p & F = [λy p]))>
235     using "≡E"(1) "logic-actual-nec:3"[axiom_inst] by blast
236   AOT_hence false_enc_cond:
237     <⊥(⊥[λx ∀q(q → q)] ≡ ∃p (¬p & [λx ∀q(q → q)] = [λy p]))>
238     using "∀E"(1)[rotated, OF "prop-prop2:2"] by blast
239
240   AOT_have true_def: <⊤ = λx (A!x & ∀F (x[F] ≡ ∃p (p & F = [λy p])))>
241     by (simp add: "A-descriptions" "rule-id-df:1[zero]" "the-true:1")
242   moreover AOT_show true_den: <⊤↓>
243     by (meson "t=t-proper:1" "A-descriptions" "rule-id-df:1[zero]" "the-true:1" "→E")
244   ultimately AOT_have true_prop: <⊤(A!⊤ & ∀F (⊤[F] ≡ ∃p (p & F = [λy p])))>
245     using "nec-hintikka-scheme"[unvarify x, THEN "≡E"(1), THEN "&E"(1)] by blast
246   AOT_hence <⊤∀F (⊤[F] ≡ ∃p (p & F = [λy p]))>
247     using "Act-Basic:2" "&E"(2) "≡E"(1) by blast
248   AOT_hence <⊤∀F ⊥(⊤[F] ≡ ∃p (p & F = [λy p]))>
249     using "≡E"(1) "logic-actual-nec:3"[axiom_inst] by blast
250   AOT_hence <⊤(⊤[λx ∀q(q → q)] ≡ ∃p (p & [λx ∀q(q → q)] = [λy p]))>

```

```

251   using "VE"(1)[rotated, OF "prop-prop2:2"] by blast
252   moreover AOT_have <math>\mathcal{A}\exists p(p \ \& \ [\lambda x \ \forall q(q \ \rightarrow \ q)] = [\lambda y \ p])>
253     by (safe intro!: "nec-imp-act"[THEN "\rightarrow E"] RN "\exists I"(1)[where \tau="<math>\forall q(q \ \rightarrow \ q)>" "&I"
254         GEN "\rightarrow I" "log-prop-prop:2" "rule=I:1" "prop-prop2:2"])
255   ultimately AOT_have <math>\mathcal{A}(\top[\lambda x \ \forall q(q \ \rightarrow \ q)])>
256     using "Act-Basic:5" "\equiv"(1,2) by blast
257   AOT_thus <math>\top[\lambda x \ \forall q(q \ \rightarrow \ q)]>
258     using "en-eq:10[1]"[unvarify x1 F, THEN "\equiv"(1)] true_den "prop-prop2:2" by blast
259
260   AOT_show <math>\neg \perp[\lambda x \ \forall q(q \ \rightarrow \ q)]>
261   proof(rule "raa-cor:2")
262     AOT_assume <math>\perp[\lambda x \ \forall q(q \ \rightarrow \ q)]>
263     AOT_hence <math>\mathcal{A}\perp[\lambda x \ \forall q(q \ \rightarrow \ q)]>
264       using "en-eq:10[1]"[unvarify x1 F, THEN "\equiv"(2)]
265         false_den "prop-prop2:2" by blast
266     AOT_hence <math>\mathcal{A}\exists p(\neg p \ \& \ [\lambda x \ \forall q(q \ \rightarrow \ q)] = [\lambda y \ p])>
267       using false_enc_cond "Act-Basic:5" "\equiv"(1) by blast
268     AOT_hence <math>\exists p \ \mathcal{A}(\neg p \ \& \ [\lambda x \ \forall q(q \ \rightarrow \ q)] = [\lambda y \ p])>
269       using "Act-Basic:10" "\equiv"(1) by blast
270     then AOT_obtain p where p_prop: <math>\mathcal{A}(\neg p \ \& \ [\lambda x \ \forall q(q \ \rightarrow \ q)] = [\lambda y \ p])>
271       using "\exists E"[rotated] by blast
272     AOT_hence <math>\mathcal{A}[\lambda x \ \forall q(q \ \rightarrow \ q)] = [\lambda y \ p]>
273       by (metis "Act-Basic:2" "\&E"(2) "\equiv"(1))
274     AOT_hence <math>[\lambda x \ \forall q(q \ \rightarrow \ q)] = [\lambda y \ p]>
275       using "id-act:1"[unvarify \alpha \ \beta, THEN "\equiv"(2)] "prop-prop2:2" by blast
276     AOT_hence <math>(\forall q(q \ \rightarrow \ q)) = p>
277       using "p-identity-thm2:3"[unvarify p, THEN "\equiv"(2)]
278         "log-prop-prop:2" by blast
279     moreover AOT_have <math>\mathcal{A}\neg p> using p_prop
280       using "Act-Basic:2" "\&E"(1) "\equiv"(1) by blast
281     ultimately AOT_have <math>\mathcal{A}\neg \forall q(q \ \rightarrow \ q)>
282       by (metis "Act-Sub:1" "\equiv"(1,2) "raa-cor:3" "rule=E")
283     moreover AOT_have <math>\neg \mathcal{A}\neg \forall q(q \ \rightarrow \ q)>
284       by (meson "Act-Sub:1" "RA[2]" "if-p-then-p" "\equiv"(1) "universal-cor")
285     ultimately AOT_show <math>\mathcal{A}\neg \forall q(q \ \rightarrow \ q) \ \& \ \neg \mathcal{A}\neg \forall q(q \ \rightarrow \ q)>
286       using "&I" by blast
287   qed
288   qed
289
290   AOT_act_theorem "T-T-value:1": <math>\langle \text{TruthValue}(\top) \rangle \tag{297.1}
291   proof -
292     AOT_have true_def: <math>\langle \top = \iota x \ (A!x \ \& \ \forall F \ (x[F] \equiv \exists p(p \ \& \ F = [\lambda y \ p]))) \rangle
293       by (simp add: "A-descriptions" "rule-id-df:1[zero]" "the-true:1")
294     AOT_hence true_den: <math>\langle \top \downarrow \rangle
295       using "t=t-proper:1" "vdash-properties:6" by blast
296     AOT_show <math>\langle \text{TruthValue}(\top) \rangle
297       using "y-in:2"[unvarify z, OF true_den, THEN "\rightarrow E", OF true_def]
298         "TV-lem2:1"[unvarify x, OF true_den, THEN "\rightarrow E"] by blast
299   qed
300
301   AOT_act_theorem "T-T-value:2": <math>\langle \text{TruthValue}(\perp) \rangle \tag{297.2}
302   proof -
303     AOT_have false_def: <math>\langle \perp = \iota x \ (A!x \ \& \ \forall F \ (x[F] \equiv \exists p(\neg p \ \& \ F = [\lambda y \ p]))) \rangle
304       by (simp add: "A-descriptions" "rule-id-df:1[zero]" "the-true:2")
305     AOT_hence false_den: <math>\langle \perp \downarrow \rangle
306       using "t=t-proper:1" "vdash-properties:6" by blast
307     AOT_show <math>\langle \text{TruthValue}(\perp) \rangle
308       using "y-in:2"[unvarify z, OF false_den, THEN "\rightarrow E", OF false_def]
309         "TV-lem2:2"[unvarify x, OF false_den, THEN "\rightarrow E"] by blast
310   qed
311
312   AOT_theorem "two-T": <math>\langle \exists x \exists y (\text{TruthValue}(x) \ \& \ \text{TruthValue}(y) \ \& \ x \neq y \ \& \tag{298}
313     \forall z (\text{TruthValue}(z) \ \rightarrow \ z = x \ \vee \ z = y)) \rangle

```



```

314 proof -
315   AOT_obtain a where a_prop: <A!a & ∀F (a[F] ≡ ∃p (p & F = [λy p]))>
316   using "A-objects"[axiom_inst] "∃E"[rotated] by fast
317   AOT_obtain b where b_prop: <A!b & ∀F (b[F] ≡ ∃p (¬p & F = [λy p]))>
318   using "A-objects"[axiom_inst] "∃E"[rotated] by fast
319   AOT_obtain p where p: p
320   by (metis "log-prop-prop:2" "raa-cor:3" "rule-ui:1" "universal-cor")
321   show ?thesis
322   proof (rule "∃I"(2)[where β=a]; rule "∃I"(2)[where β=b];
323     safe intro!: "&I" GEN "→I")
324     AOT_show <TruthValue(a)>
325     using "TV-lem2:1" a_prop "vdash-properties:10" by blast
326   next
327     AOT_show <TruthValue(b)>
328     using "TV-lem2:2" b_prop "vdash-properties:10" by blast
329   next
330     AOT_show <a ≠ b>
331     proof (rule "ab-obey:2"[THEN "→E", OF "∀I"(1)])
332       AOT_show <∃F (a[F] & ¬b[F])>
333       proof (rule "∃I"(1)[where τ="«[λy p]»"]; rule "&I" "prop-prop2:2")
334         AOT_show <a[λy p]>
335         by (safe intro!: "∃I"(2)[where β=p] "&I" p "rule=I:1"[OF "prop-prop2:2"]
336           a_prop[THEN "&E"(2), THEN "∀E"(1), THEN "≡E"(2), OF "prop-prop2:2"])
337       next
338         AOT_show <¬b[λy p]>
339         proof (rule "raa-cor:2")
340           AOT_assume <b[λy p]>
341           AOT_hence <∃q (¬q & [λy p] = [λy q])>
342           using "∀E"(1)[rotated, OF "prop-prop2:2", THEN "≡E"(1)]
343           b_prop[THEN "&E"(2)] by fast
344           then AOT_obtain q where <¬q & [λy p] = [λy q]>
345           using "∃E"[rotated] by blast
346           AOT_hence <¬p>
347           by (metis "rule=E" "&E"(1) "&E"(2) "deduction-theorem" "≡I"
348             "≡E"(2) "p-identity-thm2:3" "raa-cor:3")
349           AOT_thus <p & ¬p> using p "&I" by blast
350         qed
351       qed
352     qed
353   next
354   fix z
355   AOT_assume <TruthValue(z)>
356   AOT_hence <∃p (TruthValueOf(z, p))>
357   by (metis "≡dfE" "T-value")
358   then AOT_obtain p where <TruthValueOf(z, p)> using "∃E"[rotated] by blast
359   AOT_hence z_prop: <A!z & ∀F (z[F] ≡ ∃q ((q ≡ p) & F = [λy q]))>
360   using "≡dfE" "tv-p" by blast
361   {
362     AOT_assume p: <p>
363     AOT_have <z = a>
364     proof (rule "ab-obey:1"[THEN "→E", THEN "→E", OF "&I",
365       OF z_prop[THEN "&E"(1)], OF a_prop[THEN "&E"(1)]];
366       rule GEN)
367     fix G
368     AOT_have <z[G] ≡ ∃q ((q ≡ p) & G = [λy q])>
369     using z_prop[THEN "&E"(2)] "∀E"(2) by blast
370     also AOT_have <∃q ((q ≡ p) & G = [λy q]) ≡ ∃q (q & G = [λy q])>
371     using "TV-lem1:1"[THEN "≡E"(1), OF p, THEN "∀E"(2)[where β=G], symmetric].
372     also AOT_have <... ≡ a[G]>
373     using a_prop[THEN "&E"(2), THEN "∀E"(2)[where β=G], symmetric].
374     finally AOT_show <z[G] ≡ a[G]>.
375   }
376   qed
377   AOT_hence <z = a ∨ z = b> by (rule "∀I")

```

```

377 }
378 moreover {
379   AOT_assume notp: <¬p>
380   AOT_have <z = b>
381   proof(rule "ab-obey:1"[THEN "→E", THEN "→E", OF "&I",
382     OF z_prop[THEN "&E"(1)], OF b_prop[THEN "&E"(1)]];
383     rule GEN)
384     fix G
385     AOT_have <z[G] ≡ ∃q ((q ≡ p) & G = [λy q])>
386     using z_prop[THEN "&E"(2)] "∀E"(2) by blast
387     also AOT_have <∃q ((q ≡ p) & G = [λy q]) ≡ ∃q (¬q & G = [λy q])>
388     using "TV-lem1:2"[THEN "≡E"(1), OF notp, THEN "∀E"(2), symmetric].
389     also AOT_have <... ≡ b[G]>
390     using b_prop[THEN "&E"(2), THEN "∀E"(2), symmetric].
391     finally AOT_show <z[G] ≡ b[G]>.
392   qed
393   AOT_hence <z = a ∨ z = b> by (rule "∨I")
394 }
395 ultimately AOT_show <z = a ∨ z = b>
396 by (metis "reductio-aa:1")
397 qed
398 qed
399
400 AOT_act_theorem "valueof-facts:1": <TruthValueOf(x, p) → (p ≡ x = ⊤)> (299.1)
401 proof(safe intro!: "→I" dest!: "tv-p"[THEN "≡dfE"])
402   AOT_assume ϕ: <[A!]x & ∀F (x[F] ≡ ∃q ((q ≡ p) & F = [λy q]))>
403   AOT_have a: <A!⊤>
404     using "∃E" "T-T-value:1" "T-value" "&E"(1) "≡dfE" "tv-p" by blast
405   AOT_have true_def: <⊤ = λx (A!x & ∀F (x[F] ≡ ∃p(p & F = [λy p])))>
406     by (simp add: "A-descriptions" "rule-id-df:1[zero]" "the-true:1")
407   AOT_hence true_den: <⊤↓>
408     using "t=t-proper:1" "vdash-properties:6" by blast
409   AOT_have b: <∀F (⊤[F] ≡ ∃q (q & F = [λy q]))>
410     using "y-in:2"[unvarify z, OF true_den, THEN "→E", OF true_def] "&E" by blast
411   AOT_show <p ≡ x = ⊤>
412   proof(safe intro!: "≡I" "→I")
413     AOT_assume p
414     AOT_hence <∀F (∃q (q & F = [λy q]) ≡ ∃q ((q ≡ p) & F = [λy q]))>
415     using "TV-lem1:1"[THEN "≡E"(1)] by blast
416     AOT_hence <∀F(⊤[F] ≡ ∃q ((q ≡ p) & F = [λy q]))>
417     using b "cqt-basic:10"[THEN "→E", OF "&I", OF b] by fast
418     AOT_hence c: <∀F(∃q((q ≡ p) & F = [λy q]) ≡ ⊤[F])>
419     using "cqt-basic:11"[THEN "≡E"(1)] by fast
420     AOT_hence <∀F (x[F] ≡ ⊤[F])>
421     using "cqt-basic:10"[THEN "→E", OF "&I", OF ϕ[THEN "&E"(2)]] by fast
422     AOT_thus <x = ⊤>
423     by (rule "ab-obey:1"[unvarify y, OF true_den, THEN "→E", THEN "→E",
424       OF "&I", OF ϕ[THEN "&E"(1)], OF a])
425   next
426     AOT_assume <x = ⊤>
427     AOT_hence d: <∀F (⊤[F] ≡ ∃q ((q ≡ p) & F = [λy q]))>
428     using "rule=E" ϕ[THEN "&E"(2)] by fast
429     AOT_have <∀F (∃q (q & F = [λy q]) ≡ ∃q ((q ≡ p) & F = [λy q]))>
430     using "cqt-basic:10"[THEN "→E", OF "&I",
431       OF b[THEN "cqt-basic:11"[THEN "≡E"(1)]], OF d].
432     AOT_thus p using "TV-lem1:1"[THEN "≡E"(2)] by blast
433   qed
434 qed
435
436 AOT_act_theorem "valueof-facts:2": <TruthValueOf(x, p) → (¬p ≡ x = ⊥)> (299.2)
437 proof(safe intro!: "→I" dest!: "tv-p"[THEN "≡dfE"])
438   AOT_assume ϕ: <[A!]x & ∀F (x[F] ≡ ∃q ((q ≡ p) & F = [λy q]))>
439   AOT_have a: <A!⊥>

```

```

440   using "≡E" "T-T-value:2" "T-value" "&E"(1) "≡dfE" "tv-p" by blast
441 AOT_have false_def: <⊥ = ιx (A!x & ∀F (x[F] ≡ ∃p(¬p & F = [λy p])))>
442   by (simp add: "A-descriptions" "rule-id-df:1[zero]" "the-true:2")
443 AOT_hence false_den: <⊥↓>
444   using "t=t-proper:1" "vdash-properties:6" by blast
445 AOT_have b: <∀F (⊥[F] ≡ ∃q (¬q & F = [λy q]))>
446   using "y-in:2"[unvarify z, OF false_den, THEN "→E", OF false_def] "&E" by blast
447 AOT_show <¬p ≡ x = ⊥>
448 proof(safe intro!: "≡I" "→I")
449   AOT_assume <¬p>
450   AOT_hence <∀F (∃q (¬q & F = [λy q]) ≡ ∃q ((q ≡ p) & F = [λy q]))>
451     using "TV-lem1:2"[THEN "≡E"(1)] by blast
452   AOT_hence <∀F(⊥[F] ≡ ∃q ((q ≡ p) & F = [λy q]))>
453     using b "cqt-basic:10"[THEN "→E", OF "&I", OF b] by fast
454   AOT_hence c: <∀F(∃q((q ≡ p) & F = [λy q]) ≡ ⊥[F])>
455     using "cqt-basic:11"[THEN "≡E"(1)] by fast
456   AOT_hence <∀F (x[F] ≡ ⊥[F])>
457     using "cqt-basic:10"[THEN "→E", OF "&I", OF ∅[THEN "&E"(2)]] by fast
458   AOT_thus <x = ⊥>
459     by (rule "ab-obey:1"[unvarify y, OF false_den, THEN "→E", THEN "→E",
460       OF "&I", OF ∅[THEN "&E"(1)], OF a])
461 next
462   AOT_assume <x = ⊥>
463   AOT_hence d: <∀F (⊥[F] ≡ ∃q ((q ≡ p) & F = [λy q]))>
464     using "rule=E" ∅[THEN "&E"(2)] by fast
465   AOT_have <∀F (∃q (¬q & F = [λy q]) ≡ ∃q ((q ≡ p) & F = [λy q]))>
466     using "cqt-basic:10"[THEN "→E", OF "&I",
467       OF b[THEN "cqt-basic:11"[THEN "≡E"(1)], OF d].
468   AOT_thus <¬p> using "TV-lem1:2"[THEN "≡E"(2)] by blast
469 qed
470 qed
471
472 AOT_act_theorem "q-True:1": <p ≡ (op = ⊤)> (300.1)
473   apply (rule "valueof-facts:1"[unvarify x, THEN "→E", rotated, OF "T-lem:1"])
474   using "≡dfE" "tv-id:2" "&E"(1) "prop-enc" by blast
475
476 AOT_act_theorem "q-True:2": <¬p ≡ (op = ⊥)> (300.2)
477   apply (rule "valueof-facts:2"[unvarify x, THEN "→E", rotated, OF "T-lem:1"])
478   using "≡dfE" "tv-id:2" "&E"(1) "prop-enc" by blast
479
480 AOT_act_theorem "q-True:3": <p ≡ ⊤Σp> (300.3)
481 proof(safe intro!: "≡I" "→I")
482   AOT_assume p
483   AOT_hence <op = ⊤> by (metis "≡E"(1) "q-True:1")
484   moreover AOT_have <opΣp>
485     by (simp add: "tv-id:2")
486   ultimately AOT_show <⊤Σp>
487     using "rule=E" "T-lem:4" by fast
488 next
489   AOT_have true_def: <⊤ = ιx (A!x & ∀F (x[F] ≡ ∃p(p & F = [λy p])))>
490     by (simp add: "A-descriptions" "rule-id-df:1[zero]" "the-true:1")
491   AOT_hence true_den: <⊤↓>
492     using "t=t-proper:1" "vdash-properties:6" by blast
493   AOT_have b: <∀F (⊤[F] ≡ ∃q (q & F = [λy q]))>
494     using "y-in:2"[unvarify z, OF true_den, THEN "→E", OF true_def] "&E" by blast
495
496   AOT_assume <⊤Σp>
497   AOT_hence <⊤[λy p]> by (metis "≡dfE" "&E"(2) "prop-enc")
498   AOT_hence <∃q (q & [λy p] = [λy q])>
499     using b[THEN "∀E"(1), OF "prop-prop2:2", THEN "≡E"(1)] by blast
500   then AOT_obtain q where <q & [λy p] = [λy q]> using "∃E"[rotated] by blast
501   AOT_thus <p>
502     using "rule=E" "&E"(1) "&E"(2) id_sym "≡E"(2) "p-identity-thm2:3" by fast

```

```

503 qed
504
505
506 AOT_act_theorem "q-True:5": < $\neg p \equiv \perp \Sigma p$ > (300.5)
507 proof(safe intro!: "≡I" "→I")
508   AOT_assume < $\neg p$ >
509   AOT_hence < $\circ p = \perp$ > by (metis "≡E"(1) "q-True:2")
510   moreover AOT_have < $\circ p \Sigma p$ >
511     by (simp add: "tv-id:2")
512   ultimately AOT_show < $\perp \Sigma p$ >
513     using "rule=E" "T-lem:4" by fast
514 next
515   AOT_have false_def: < $\perp = \iota x (A!x \ \& \ \forall F (x[F] \equiv \exists p (\neg p \ \& \ F = [\lambda y p])))$ >
516     by (simp add: "A-descriptions" "rule-id-df:1[zero]" "the-true:2")
517   AOT_hence false_den: < $\perp \downarrow$ >
518     using "t=t-proper:1" "vdash-properties:6" by blast
519   AOT_have b: < $\forall F (\perp [F] \equiv \exists q (\neg q \ \& \ F = [\lambda y q]))$ >
520     using "y-in:2"[unvarify z, OF false_den, THEN "→E", OF false_def] "&E" by blast
521
522   AOT_assume < $\perp \Sigma p$ >
523   AOT_hence < $\perp [\lambda y p]$ > by (metis "≡dfE" "&E"(2) "prop-enc")
524   AOT_hence < $\exists q (\neg q \ \& \ [\lambda y p] = [\lambda y q])$ >
525     using b[THEN "∀E"(1), OF "prop-prop2:2", THEN "≡E"(1)] by blast
526   then AOT_obtain q where < $\neg q \ \& \ [\lambda y p] = [\lambda y q]$ > using "∃E"[rotated] by blast
527   AOT_thus < $\neg p$ >
528     using "rule=E" "&E"(1) "&E"(2) id_sym "≡E"(2) "p-identity-thm2:3" by fast
529 qed
530
531 AOT_act_theorem "q-True:4": < $p \equiv \neg(\perp \Sigma p)$ > (300.4)
532   using "q-True:5"
533   by (metis "deduction-theorem" "≡I" "≡E"(2) "≡E"(4) "raa-cor:3")
534
535 AOT_act_theorem "q-True:6": < $\neg p \equiv \neg(\top \Sigma p)$ > (300.6)
536   using "≡E"(1) "oth-class-taut:4:b" "q-True:3" by blast
537
538 AOT_define ExtensionOf :: < $\tau \Rightarrow \varphi \Rightarrow \varphi$ > (<ExtensionOf'(_,_)>)
539   "exten-p": <ExtensionOf(x,p) ≡df A!x & (301)
540      $\forall F (x[F] \rightarrow \text{Propositional}([F])) \ \&$ 
541      $\forall q ((x \Sigma q) \equiv (q \equiv p))$ >
542
543 AOT_theorem "extof-e": <ExtensionOf(x, p) ≡ TruthValueOf(x, p)> (302)
544 proof (safe intro!: "≡I" "→I" "tv-p"[THEN "≡dfI"] "exten-p"[THEN "≡dfE"]
545   dest!: "tv-p"[THEN "≡dfE"] "exten-p"[THEN "≡dfE"]
546   AOT_assume 1: <[A!]x &  $\forall F (x[F] \rightarrow \text{Propositional}([F])) \ \& \ \forall q (x \Sigma q \equiv (q \equiv p))$ >
547   AOT_hence  $\vartheta$ : <[A!]x &  $\forall F (x[F] \rightarrow \exists q (F = [\lambda y q])) \ \& \ \forall q (x \Sigma q \equiv (q \equiv p))$ >
548     by (AOT_subst < $\exists q (F = [\lambda y q])$ > <Propositional([F])> for: F :: << $\kappa$ >>)
549     (auto simp add: "df-rules-formulas[3]" "df-rules-formulas[4]"
550       "≡I" "prop-prop1")
551   AOT_show <[A!]x &  $\forall F (x[F] \equiv \exists q ((q \equiv p) \ \& \ F = [\lambda y q]))$ >
552   proof(safe intro!: "&I" GEN 1[THEN "&E"(1), THEN "&E"(1)] "≡I" "→I")
553     fix F
554     AOT_assume 0: <x[F]>
555     AOT_hence < $\exists q (F = [\lambda y q])$ >
556       using  $\vartheta$ [THEN "&E"(1), THEN "&E"(2)] "∀E"(2) "→E" by blast
557     then AOT_obtain q_prop: <F = [\lambda y q]> using "∃E"[rotated] by blast
558     AOT_hence <x[\lambda y q]> using 0 "rule=E" by blast
559     AOT_hence <x\Sigma q> by (metis "≡dfI" "&I" "ex:1:a" "prop-enc" "rule-ui:3")
560     AOT_hence <q ≡ p> using  $\vartheta$ [THEN "&E"(2)] "∀E"(2) "≡E"(1) by blast
561     AOT_hence <(q ≡ p) & F = [\lambda y q]> using q_prop "&I" by blast
562     AOT_thus < $\exists q ((q \equiv p) \ \& \ F = [\lambda y q])$ > by (rule "∃I")
563   next
564     fix F
565     AOT_assume < $\exists q ((q \equiv p) \ \& \ F = [\lambda y q])$ >

```

```

566   then AOT_obtain q where q_prop: <(q ≡ p) & F = [λy q]>
567     using "∃E"[rotated] by blast
568   AOT_hence <xΣq> using ∅[THEN "&E"(2)] "∀E"(2) "&E" "≡E"(2) by blast
569   AOT_hence <x[λy q]> by (metis "≡dfE" "&E"(2) "prop-enc")
570   AOT_thus <x[F]> using q_prop[THEN "&E"(2), symmetric] "rule=E" by blast
571   qed
572 next
573 AOT_assume 0: <[A!]x & ∀F (x[F] ≡ ∃q ((q ≡ p) & F = [λy q]))>
574 AOT_show <[A!]x & ∀F (x[F] → Propositional([F])) & ∀q (x Σ q ≡ (q ≡ p))>
575 proof(safe intro!: "&I" 0[THEN "&E"(1)] GEN "→I")
576   fix F
577   AOT_assume <x[F]>
578   AOT_hence <∃q ((q ≡ p) & F = [λy q])>
579     using 0[THEN "&E"(2)] "∀E"(2) "≡E"(1) by blast
580   then AOT_obtain q where <(q ≡ p) & F = [λy q]>
581     using "∃E"[rotated] by blast
582   AOT_hence <F = [λy q]> using "&E"(2) by blast
583   AOT_hence <∃q F = [λy q]> by (rule "∃I")
584   AOT_thus <Propositional([F])> by (metis "≡dfI" "prop-prop1")
585 next
586 AOT_show <xΣr ≡ (r ≡ p)> for r
587 proof(rule "≡I"; rule "→I")
588   AOT_assume <xΣr>
589   AOT_hence <x[λy r]> by (metis "≡dfE" "&E"(2) "prop-enc")
590   AOT_hence <∃q ((q ≡ p) & [λy r] = [λy q])>
591     using 0[THEN "&E"(2), THEN "∀E"(1), OF "prop-prop2:2", THEN "≡E"(1)] by blast
592   then AOT_obtain q where <(q ≡ p) & [λy r] = [λy q]>
593     using "∃E"[rotated] by blast
594   AOT_thus <r ≡ p>
595     by (metis "rule=E" "&E"(1,2) id_sym "≡E"(2) "Commutativity of ≡"
596           "p-identity-thm2:3")
597 next
598   AOT_assume <r ≡ p>
599   AOT_hence <(r ≡ p) & [λy r] = [λy r]>
600     by (metis "rule=I:1" "≡S"(1) "≡E"(2) "Commutativity of &" "prop-prop2:2")
601   AOT_hence <∃q ((q ≡ p) & [λy r] = [λy q])> by (rule "∃I")
602   AOT_hence <x[λy r]>
603     using 0[THEN "&E"(2), THEN "∀E"(1), OF "prop-prop2:2", THEN "≡E"(2)] by blast
604   AOT_thus <xΣr> by (metis "≡dfI" "&I" "ex:1:a" "prop-enc" "rule-ui:3")
605   qed
606   qed
607   qed
608
609 AOT_theorem "ext-p-tv:1": <∃!x ExtensionOf(x, p)> (303.1)
610   by (AOT_subst <ExtensionOf(x, p)> <TruthValueOf(x, p)> for: x)
611     (auto simp: "extof-e" "p-has-!tv:2")
612
613 AOT_theorem "ext-p-tv:2": <ιx(ExtensionOf(x, p))↓> (303.2)
614   using "A-Exists:2" "RA[2]" "ext-p-tv:1" "≡E"(2) by blast
615
616 AOT_theorem "ext-p-tv:3": <ιx(ExtensionOf(x, p)) = op> (303.3)
617 proof -
618   AOT_have 0: <⊆∀x(ExtensionOf(x, p) ≡ TruthValueOf(x,p))>
619     by (rule "RA[2]"; rule GEN; rule "extof-e")
620   AOT_have 1: <op = ιx TruthValueOf(x,p)>
621     using "rule-id-df:1" "the-tv-p" "uni-tv" by blast
622   show ?thesis
623     apply (rule "equiv-desc-eq:1"[THEN "→E", OF 0, THEN "∀E"(1)[where τ=<<op>>],
624           THEN "≡E"(2), symmetric])
625     using "1" "t=t-proper:1" "vdash-properties:10" apply blast
626     by (fact 1)
627   qed

```

628 (***<***)end(***>***)

629

A.9. Restricted Variables

```

1  (*<*)
2  theory AOT_RestrictedVariables
3    imports AOT_PLM
4    keywords "AOT_register_rigid_restricted_type" :: thy_goal
5           and "AOT_register_restricted_type" :: thy_goal
6  begin
7  (*>*)
8
9  section<Restricted Variables>
10
11 locale AOT_restriction_condition =
12   fixes  $\psi$  :: <'a::AOT_Term_id_2  $\Rightarrow$  o>
13   assumes "res-var:2"[AOT]: <[v  $\models \exists \alpha \psi\{\alpha\}$ > (330.2)
14   assumes "res-var:3"[AOT]: <[v  $\models \psi\{\tau\} \rightarrow \tau\downarrow$ > (330.3)
15
16 ML<
17 fun register_restricted_type (name:string, restriction:string) thy =
18 let
19 val ctxt = thy
20 val ctxt = setupStrictWorld ctxt
21 val trm = Syntax.check_term ctxt (AOT_read_term @<nonterminal  $\varphi$ '> ctxt restriction)
22 val free = case (Term.add_frees trm []) of [f] => f |
23   _ => raise Term.TERM ("Expected single free variable.", [trm])
24 val trm = Term.absfree free trm
25 val localeTerm = Const (const_name<AOT_restriction_condition>, dummyT) $ trm
26 val localeTerm = Syntax.check_term ctxt localeTerm
27 fun after_qed thms thy = let
28 val st = Interpretation.global_interpretation
29   (((@<locale AOT_restriction_condition>, ((name, true),
30     (Expression.Named [(" $\psi$ ", trm)], []))), [], [])) [] thy
31 val st = Proof.refine_insert (flat thms) st
32 val thy = Proof.global_immediate_proof st
33
34 val thy = Local_Theory.background_theory
35   (AOT_Constraints.map (Symtab.update
36     (name, (term_of (snd free), term_of (snd free)))) thy
37 val thy = Local_Theory.background_theory
38   (AOT_Restriction.map (Symtab.update
39     (name, (trm, Const (const_name<AOT_term_of_var>, dummyT)))) thy
40
41 in thy end
42 in
43 Proof.theorem NONE after_qed [[(HOLogic.mk_Trueprop localeTerm, [])] ctxt
44 end
45
46 val _ =
47   Outer_Syntax.command
48     command_keyword<AOT_register_restricted_type>
49     "Register a restricted type."
50     ((Parse.short_ident -| Parse.$$$ ":" - Parse.term) »
51     (Toplevel.local_theory_to_proof NONE NONE o register_restricted_type));
52 >
53
54 locale AOT_rigid_restriction_condition = AOT_restriction_condition +
55   assumes rigid[AOT]: <[v  $\models \forall \alpha (\psi\{\alpha\} \rightarrow \Box \psi\{\alpha\})$ >
56 begin
57 lemma rigid_condition[AOT]: <[v  $\models \Box (\psi\{\alpha\} \rightarrow \Box \psi\{\alpha\})$ >
58   using rigid[THEN " $\forall E$ "(2)] RN by simp
59 lemma type_set_nonempty[AOT_no_atp, no_atp]: < $\exists x . x \in \{ \alpha . [w_0 \models \psi\{\alpha\}] \}$ >
60   by (metis "instantiation" mem_Collect_eq "res-var:2")
61 end

```

```

62
63 locale AOT_restricted_type = AOT_rigid_restriction_condition +
64   fixes Rep and Abs
65   assumes AOT_restricted_type_definition[AOT_no_atp]:
66     <type_definition Rep Abs {  $\alpha$  . [ $w_0 \models \psi\{\alpha\}$ ]}>
67 begin
68
69 AOT_theorem restricted_var_condition: < $\psi\{\llbracket\text{AOT\_term\_of\_var (Rep } \alpha)\rrbracket\}$ >
70 proof -
71   interpret type_definition Rep Abs "{  $\alpha$  . [ $w_0 \models \psi\{\alpha\}$ ]"
72   using AOT_restricted_type_definition.
73   AOT_actually {
74     AOT_have <<AOT_term_of_var (Rep  $\alpha$ )>> and < $\psi\{\llbracket\text{AOT\_term\_of\_var (Rep } \alpha)\rrbracket\}$ >
75     using AOT_sem_imp Rep "res-var:3" by auto
76   }
77   moreover AOT_actually {
78     AOT_have < $\psi\{\alpha\} \rightarrow \Box\psi\{\alpha\}$ > for  $\alpha$ 
79     using AOT_sem_box rigid_condition by presburger
80     AOT_hence < $\psi\{\tau\} \rightarrow \Box\psi\{\tau\}$ > if < $\tau \downarrow$ > for  $\tau$ 
81     by (metis AOT_model.AOT_term_of_var_cases AOT_sem_denotes that)
82   }
83   ultimately AOT_show < $\psi\{\llbracket\text{AOT\_term\_of\_var (Rep } \alpha)\rrbracket\}$ >
84   using AOT_sem_box AOT_sem_imp by blast
85 qed
86 lemmas " $\psi$ " = restricted_var_condition
87
88 AOT_theorem GEN: assumes <for arbitrary  $\alpha$ :  $\varphi\{\llbracket\text{AOT\_term\_of\_var (Rep } \alpha)\rrbracket\}$ >
89   shows < $\forall \alpha (\psi\{\alpha\} \rightarrow \varphi\{\alpha\})$ >
90 proof(rule GEN; rule " $\rightarrow$ I")
91   interpret type_definition Rep Abs "{  $\alpha$  . [ $w_0 \models \psi\{\alpha\}$ ]"
92   using AOT_restricted_type_definition.
93   fix  $\alpha$ 
94   AOT_assume < $\psi\{\alpha\}$ >
95   AOT_hence < $\mathcal{A}\psi\{\alpha\}$ >
96   by (metis AOT_model_axiom_def AOT_sem_box AOT_sem_imp act_closure rigid_condition)
97   hence 0: < $w_0 \models \psi\{\alpha\}$ > by (metis AOT_sem_act)
98   {
99     fix  $\tau$ 
100     assume  $\alpha\_def$ : < $\alpha = \text{Rep } \tau$ >
101     AOT_have < $\varphi\{\alpha\}$ >
102     unfolding  $\alpha\_def$ 
103     using assms by blast
104   }
105   AOT_thus < $\varphi\{\alpha\}$ >
106   using Rep_cases[simplified, OF 0]
107   by blast
108 qed
109 lemmas " $\forall$ I" = GEN
110
111 end
112
113
114 lemma AOT_restricted_type_intro[AOT_no_atp, no_atp]:
115   assumes <type_definition Rep Abs {  $\alpha$  . [ $w_0 \models \psi\{\alpha\}$ ]}>
116   and <AOT_rigid_restriction_condition  $\psi$ >
117   shows <AOT_restricted_type  $\psi$  Rep Abs>
118   by (auto intro!: assms AOT_restricted_type_axioms.intro AOT_restricted_type.intro)
119
120
121
122 ML<
123 fun register_rigid_restricted_type (name:string, restriction:string) thy =
124 let

```



```

125 val ctxt = thy
126 val ctxt = setupStrictWorld ctxt
127 val trm = Syntax.check_term ctxt (AOT_read_term @{nonterminal  $\varphi$ '} ctxt restriction)
128 val free = case (Term.add_frees trm []) of [f] => f
129 | _ => raise Term.TERM ("Expected single free variable.", [trm])
130 val trm = Term.absfree free trm
131 val localeTerm = HOLogic.mk_Trueprop
132   (Const (const_name<AOT_rigid_restriction_condition>, dummyT) $ trm)
133 val localeTerm = Syntax.check_prop ctxt localeTerm
134 val int_bnd = Binding.concealed (Binding.qualify true "internal" (Binding.name name))
135 val bnds = {Rep_name = Binding.qualify true name (Binding.name "Rep"),
136            Abs_name = Binding.qualify true "Abs" int_bnd,
137            type_definition_name = Binding.qualify true "type_definition" int_bnd}
138
139 fun after_qed witts thy = let
140   val thms = (map (Element.conclude_witness ctxt) (flat witts))
141
142   val typeset = HOLogic.mk_Collect (" $\alpha$ ", dummyT,
143     const<AOT_model_valid_in> $ const<w0> $
144     (trm $ (Const (const_name<AOT_term_of_var>, dummyT) $ Bound 0)))
145   val typeset = Syntax.check_term thy typeset
146   val nonempty_thm = Drule.OF
147     (@{thm AOT_rigid_restriction_condition.type_set_nonempty}, thms)
148
149   val ((_,st),thy) = Typedef.add_typedef {overloaded=true}
150     (Binding.name name, [], Mixfix.NoSyn) typeset (SOME bnds)
151     (fn ctxt => (Tactic.resolve_tac ctxt ([nonempty_thm] 1)) thy)
152   val ({rep_type = _, abs_type = _, Rep_name = Rep_name, Abs_name = Abs_name,
153       axiom_name = _},
154       {inhabited = _, type_definition = type_definition, Rep = _,
155        Rep_inverse = _, Abs_inverse = _, Rep_inject = _, Abs_inject = _,
156        Rep_cases = _, Abs_cases = _, Rep_induct = _, Abs_induct = _}) = st
157
158   val locale_thm = Drule.OF (@{thm AOT_restricted_type_intro}, type_definition::thms)
159
160   val st = Interpretation.global_interpretation (([(@{locale AOT_restricted_type},
161     ((name, true), (Expression.Named [
162       (" $\psi$ ", trm),
163       ("Rep", Const (Rep_name, dummyT)),
164       ("Abs", Const (Abs_name, dummyT))], [])))
165     ], [])) [] thy
166
167   val st = Proof.refine_insert [locale_thm] st
168   val thy = Proof.global_immediate_proof st
169
170   val thy = Local_Theory.background_theory (AOT_Constraints.map (
171     Syntab.update (name, (term_of (snd free), term_of (snd free)))) thy
172   val thy = Local_Theory.background_theory (AOT_Restriction.map (
173     Syntab.update (name, (trm, Const (Rep_name, dummyT)))) thy
174
175   in thy end
176 in
177 Element.witness_proof after_qed [[localeTerm]] thy
178 end
179
180 val _ =
181   Outer_Syntax.command
182     command_keyword<AOT_register_rigid_restricted_type>
183     "Register a restricted type."
184     (((Parse.short_ident -| Parse.$$$ ":" ) - Parse.term) »
185     (Toplevel.local_theory_to_proof NONE NONE o register_rigid_restricted_type));
186 >
187

```

```

188 (* Generalized mechanism for "AOT_restricted_type.∀I" followed by ∀E *)
189 ML<
190 fun get_instantiated_allI ctxt varname thm = let
191   val trm = Thm.concl_of thm
192   val trm = case trm of (@{const Trueprop} $ (@{const AOT_model_valid_in} $ _ $ x)) => x
193                 | _ => raise Term.TERM ("Expected simple theorem.", [trm])
194   fun extractVars (Const (const_name<AOT_term_of_var>, t) $ (Const rep $ Var v)) =
195       (if fst (fst v) = fst varname
196        then [Const (const_name<AOT_term_of_var>, t) $ (Const rep $ Var v)]
197        else []) (* TODO: care about the index *)
198   | extractVars (t1 $ t2) = extractVars t1 @ extractVars t2
199   | extractVars (Abs (_, _, t)) = extractVars t
200   | extractVars _ = []
201   val vars = extractVars trm
202   val vartrm = hd vars
203   val vars = fold Term.add_vars vars []
204   val var = hd vars
205   val trmty = (case vartrm of (Const (_, Type ("fun", [_ , ty])) $ _) => ty
206                             | _ => raise Match)
207   val varty = snd var
208   val tyname = fst (Term.dest_Type varty)
209   val b = tyname^".∀I" (* TODO: better way to find the theorem *)
210   val thms = fst (Context.map_proof_result (fn ctxt => (Attrib.eval_thms ctxt
211     [(Facts.Named ((b,Position.none),NONE),[]]), ctxt)) ctxt)
212   val allthm = (case thms of (thm::_) => thm
213                       | _ => raise Fail "Unknown restricted type.")
214   val trm = Abs (Term.string_of_vname (fst var), trmty, Term.abstract_over (vartrm, trm))
215   val trm = Thm.ctrm_of (Context.proof_of ctxt) trm
216   val phi = hd (Term.add_vars (Thm.prop_of allthm) [])
217   val allthm = Drule.instantiate_normalize (TVars.empty, Vars.make [(phi,trm)]) allthm
218   in
219   allthm
220   end
221   >
222
223 (* TODO: unconstraining multiple variables does not work yet *)
224 attribute_setup "unconstrain" =
225   <Scan.lift (Scan.repeat1 Args.var) >> (fn args => Thm.rule_attribute []
226     (fn ctxt => fn thm =>
227       let
228         val thm = fold (fn arg => fn thm => thm RS get_instantiated_allI ctxt arg thm)
229                       args thm
230         val thm = fold (fn _ => fn thm => thm RS @{thm "∀E"(2)}) args thm
231         in
232         thm
233         end))>
234   "Generalize a statement about restricted variables to a statement about
235   unrestricted variables with explicit restriction condition."
236
237
238
239 context AOT_restricted_type
240 begin
241
242 AOT_theorem "rule-ui":
243   assumes <∀α(ψ{α} → φ{α})>
244   shows <φ{«AOT_term_of_var (Rep α)»}>
245 proof -
246   AOT_have <φ{α}> if <ψ{α}> for α using assms[THEN "∀E"(2), THEN "→E"] that by blast
247   moreover AOT_have <ψ{«AOT_term_of_var (Rep α)»}>
248     by (auto simp: ψ)
249   ultimately show ?thesis by blast
250 qed

```

(93)

```

251 lemmas "∀E" = "rule-ui"
252
253 AOT_theorem "instantiation": (102)
254   assumes <for arbitrary  $\beta$ :  $\varphi\{\llbracket\text{AOT\_term\_of\_var (Rep } \beta)\rrbracket\} \vdash \chi$ > and < $\exists\alpha (\psi\{\alpha\} \ \& \ \varphi\{\alpha\})$ >
255   shows < $\chi$ >
256 proof -
257   AOT_have < $\varphi\{\llbracket\text{AOT\_term\_of\_var (Rep } \alpha)\rrbracket\} \rightarrow \chi$ > for  $\alpha$ 
258     using assms(1)
259     by (simp add: "deduction-theorem")
260   AOT_hence 0: < $\forall\alpha (\psi\{\alpha\} \rightarrow (\varphi\{\alpha\} \rightarrow \chi))$ >
261     using GEN by simp
262   moreover AOT_obtain  $\alpha$  where < $\psi\{\alpha\} \ \& \ \varphi\{\alpha\}$ > using assms(2) "∃E"[rotated] by blast
263   ultimately AOT_show < $\chi$ > using "AOT_PLM.∀E"(2) [THEN "→E", THEN "→E"] "&E" by fast
264 qed
265 lemmas "∃E" = "instantiation"
266
267 AOT_theorem existential: assumes < $\varphi\{\llbracket\text{AOT\_term\_of\_var (Rep } \beta)\rrbracket\}$ > (101)
268   shows < $\exists \alpha (\psi\{\alpha\} \ \& \ \varphi\{\alpha\})$ >
269   by (meson AOT_restricted_type. $\psi$  AOT_restricted_type_axioms assms
270       "&I" "existential:2[const_var]")
271 lemmas "∃I" = existential
272 end
273
274
275 context AOT_rigid_restriction_condition
276 begin
277
278 AOT_theorem "res-var-bound-reas[1]": (334)
279   < $\forall\alpha(\psi\{\alpha\} \rightarrow \forall\beta \varphi\{\alpha, \beta\}) \equiv \forall\beta\forall\alpha (\psi\{\alpha\} \rightarrow \varphi\{\alpha, \beta\})$ >
280 proof(safe intro!: "≡I" "→I" GEN)
281   fix  $\beta \ \alpha$ 
282   AOT_assume < $\forall\alpha (\psi\{\alpha\} \rightarrow \forall\beta \varphi\{\alpha, \beta\})$ >
283   AOT_hence < $\psi\{\alpha\} \rightarrow \forall\beta \varphi\{\alpha, \beta\}$ > using "∀E"(2) by blast
284   moreover AOT_assume < $\psi\{\alpha\}$ >
285   ultimately AOT_have < $\forall\beta \varphi\{\alpha, \beta\}$ > using "→E" by blast
286   AOT_thus < $\varphi\{\alpha, \beta\}$ > using "∀E"(2) by blast
287 next
288   fix  $\alpha \ \beta$ 
289   AOT_assume < $\forall\beta\forall\alpha(\psi\{\alpha\} \rightarrow \varphi\{\alpha, \beta\})$ >
290   AOT_hence < $\forall\alpha(\psi\{\alpha\} \rightarrow \varphi\{\alpha, \beta\})$ > using "∀E"(2) by blast
291   AOT_hence < $\psi\{\alpha\} \rightarrow \varphi\{\alpha, \beta\}$ > using "∀E"(2) by blast
292   moreover AOT_assume < $\psi\{\alpha\}$ >
293   ultimately AOT_show < $\varphi\{\alpha, \beta\}$ > using "→E" by blast
294 qed
295
296 AOT_theorem "res-var-bound-reas[BF]": (334)
297   < $\forall\alpha(\psi\{\alpha\} \rightarrow \Box\varphi\{\alpha\}) \rightarrow \Box\forall\alpha(\psi\{\alpha\} \rightarrow \varphi\{\alpha\})$ >
298 proof(safe intro!: "→I")
299   AOT_assume < $\forall\alpha(\psi\{\alpha\} \rightarrow \Box\varphi\{\alpha\})$ >
300   AOT_hence < $\psi\{\alpha\} \rightarrow \Box\varphi\{\alpha\}$ > for  $\alpha$ 
301     using "∀E"(2) by blast
302   AOT_hence < $\Box(\psi\{\alpha\} \rightarrow \varphi\{\alpha\})$ > for  $\alpha$ 
303     by (metis "sc-eq-box-box:6" rigid_condition "vdash-properties:6")
304   AOT_hence < $\forall\alpha \Box(\psi\{\alpha\} \rightarrow \varphi\{\alpha\})$ >
305     by (rule GEN)
306   AOT_thus < $\Box\forall\alpha (\psi\{\alpha\} \rightarrow \varphi\{\alpha\})$ >
307     by (metis "BF" "vdash-properties:6")
308 qed
309
310 AOT_theorem "res-var-bound-reas[CBF]": (334)
311   < $\Box\forall\alpha(\psi\{\alpha\} \rightarrow \varphi\{\alpha\}) \rightarrow \forall\alpha(\psi\{\alpha\} \rightarrow \Box\varphi\{\alpha\})$ >
312 proof(safe intro!: "→I" GEN)
313   fix  $\alpha$ 

```

```

314 AOT_assume <□∀α (ψ{α} → φ{α})>
315 AOT_hence <∀α □(ψ{α} → φ{α})>
316   by (metis "CBF" "vdash-properties:6")
317 AOT_hence 1: <□(ψ{α} → φ{α})>
318   using "∇E"(2) by blast
319 AOT_assume <ψ{α}>
320 AOT_hence <□ψ{α}>
321   by (metis "B◇" "T◇" rigid_condition "vdash-properties:6")
322 AOT_thus <□φ{α}>
323   using 1 "qml:1"[axiom_inst, THEN "→E", THEN "→E"] by blast
324 qed
325
326 AOT_theorem "res-var-bound-reas[2]":
327 <∀α (ψ{α} →  $\mathcal{A}\varphi\{\alpha\}$ ) →  $\mathcal{A}\forall\alpha$  (ψ{α} → φ{α})> (334)
328 proof(safe intro!: "→I")
329   AOT_assume <∀α (ψ{α} →  $\mathcal{A}\varphi\{\alpha\}$ )>
330   AOT_hence <ψ{α} →  $\mathcal{A}\varphi\{\alpha\}$ > for α
331     using "∇E"(2) by blast
332   AOT_hence < $\mathcal{A}(\psi\{\alpha\} \rightarrow \varphi\{\alpha\})$ > for α
333     by (metis "sc-eq-box-box:7" rigid_condition "vdash-properties:6")
334   AOT_hence <∀α  $\mathcal{A}(\psi\{\alpha\} \rightarrow \varphi\{\alpha\})$ >
335     by (rule GEN)
336   AOT_thus < $\mathcal{A}\forall\alpha(\psi\{\alpha\} \rightarrow \varphi\{\alpha\})$ >
337     by (metis "≡E"(2) "logic-actual-nec:3"[axiom_inst])
338 qed
339
340
341 AOT_theorem "res-var-bound-reas[3]":
342 < $\mathcal{A}\forall\alpha$  (ψ{α} → φ{α}) → ∀α (ψ{α} →  $\mathcal{A}\varphi\{\alpha\}$ )> (334)
343 proof(safe intro!: "→I" GEN)
344   fix α
345   AOT_assume < $\mathcal{A}\forall\alpha$  (ψ{α} → φ{α})>
346   AOT_hence <∀α  $\mathcal{A}(\psi\{\alpha\} \rightarrow \varphi\{\alpha\})$ >
347     by (metis "≡E"(1) "logic-actual-nec:3"[axiom_inst])
348   AOT_hence 1: < $\mathcal{A}(\psi\{\alpha\} \rightarrow \varphi\{\alpha\})$ > by (metis "rule-ui:3")
349   AOT_assume <ψ{α}>
350   AOT_hence < $\mathcal{A}\psi\{\alpha\}$ >
351     by (metis "nec-imp-act" "qml:2"[axiom_inst] rigid_condition "→E")
352   AOT_thus < $\mathcal{A}\varphi\{\alpha\}$ >
353     using 1 by (metis "act-cond" "→E")
354 qed
355
356 AOT_theorem "res-var-bound-reas[Buridan]":
357 <∃α (ψ{α} & □φ{α}) → □∃α (ψ{α} & φ{α})> (334)
358 proof (rule "→I")
359   AOT_assume <∃α (ψ{α} & □φ{α})>
360   then AOT_obtain α where <ψ{α} & □φ{α}>
361     using "∃E"[rotated] by blast
362   AOT_hence <□(ψ{α} & φ{α})>
363     by (metis "KBasic:11" "KBasic:3" "T◇" "&I" "&E"(1) "&E"(2)
364         "≡E"(2) "reductio-aa:1" rigid_condition "vdash-properties:6")
365   AOT_hence <∃α □(ψ{α} & φ{α})>
366     by (rule "∃I")
367   AOT_thus <□∃α (ψ{α} & φ{α})>
368     by (rule Buridan[THEN "→E"])
369 qed
370
371 AOT_theorem "res-var-bound-reas[BF◇]":
372 <◇∃α (ψ{α} & φ{α}) → ∃α (ψ{α} & ◇φ{α})> (334)
373 proof(rule "→I")
374   AOT_assume <◇∃α (ψ{α} & φ{α})>
375   AOT_hence <∃α ◇(ψ{α} & φ{α})>
376     using "BF◇"[THEN "→E"] by blast

```

```

377   then AOT_obtain  $\alpha$  where  $\langle \diamond(\psi\{a\} \ \& \ \varphi\{a\}) \rangle$ 
378     using "E" [rotated] by blast
379   AOT_hence  $\langle \diamond\psi\{a\} \ \& \ \diamond\varphi\{a\} \rangle$ 
380     using "KBasic2:3" "&E" " $\rightarrow$ E" by blast+
381   moreover AOT_have  $\langle \psi\{a\} \rangle$ 
382     using calculation rigid_condition by (metis "B $\diamond$ " "K $\diamond$ " " $\rightarrow$ E")
383   ultimately AOT_have  $\langle \psi\{a\} \ \& \ \diamond\varphi\{a\} \rangle$ 
384     using "&I" by blast
385   AOT_thus  $\langle \exists\alpha (\psi\{a\} \ \& \ \diamond\varphi\{a\}) \rangle$ 
386     by (rule "EI")
387 qed
388
389 AOT_theorem "res-var-bound-reas[CBF $\diamond$ ]": (334)
390    $\langle \exists\alpha (\psi\{a\} \ \& \ \diamond\varphi\{a\}) \rightarrow \diamond\exists\alpha (\psi\{a\} \ \& \ \varphi\{a\}) \rangle$ 
391 proof(rule " $\rightarrow$ I")
392   AOT_assume  $\langle \exists\alpha (\psi\{a\} \ \& \ \diamond\varphi\{a\}) \rangle$ 
393   then AOT_obtain  $\alpha$  where  $\langle \psi\{a\} \ \& \ \diamond\varphi\{a\} \rangle$ 
394     using "E" [rotated] by blast
395   AOT_hence  $\langle \Box\psi\{a\} \ \& \ \diamond\varphi\{a\} \rangle$ 
396     using rigid_condition [THEN "qml:2" [axiom_inst, THEN " $\rightarrow$ E"], THEN " $\rightarrow$ E"] "&E" by blast+
397   AOT_hence  $\langle \diamond(\psi\{a\} \ \& \ \varphi\{a\}) \rangle$ 
398     by (metis "KBasic:16" "con-dis-taut:5" " $\rightarrow$ E")
399   AOT_hence  $\langle \exists\alpha \diamond(\psi\{a\} \ \& \ \varphi\{a\}) \rangle$ 
400     by (rule "EI")
401   AOT_thus  $\langle \diamond\exists\alpha (\psi\{a\} \ \& \ \varphi\{a\}) \rangle$ 
402     using "CBF $\diamond$ " [THEN " $\rightarrow$ E"] by fast
403 qed
404
405 AOT_theorem "res-var-bound-reas[A-Exists:1]": (334)
406    $\langle \mathcal{A}\exists!\alpha (\psi\{a\} \ \& \ \varphi\{a\}) \equiv \exists!\alpha (\psi\{a\} \ \& \ \mathcal{A}\varphi\{a\}) \rangle$ 
407 proof(safe intro!: "EI" " $\rightarrow$ I")
408   AOT_assume  $\langle \mathcal{A}\exists!\alpha (\psi\{a\} \ \& \ \varphi\{a\}) \rangle$ 
409   AOT_hence  $\langle \exists!\alpha \mathcal{A}(\psi\{a\} \ \& \ \varphi\{a\}) \rangle$ 
410     using "A-Exists:1" [THEN "E"(1)] by blast
411   AOT_hence  $\langle \exists!\alpha (\mathcal{A}\psi\{a\} \ \& \ \mathcal{A}\varphi\{a\}) \rangle$ 
412     apply (AOT_subst  $\langle \mathcal{A}\psi\{a\} \ \& \ \mathcal{A}\varphi\{a\} \rangle \langle \mathcal{A}(\psi\{a\} \ \& \ \varphi\{a\}) \rangle$  for:  $\alpha$ )
413     apply (meson "Act-Basic:2" "intro-elim:3:f" "oth-class-taut:3:a")
414     by simp
415   AOT_thus  $\langle \exists!\alpha (\psi\{a\} \ \& \ \mathcal{A}\varphi\{a\}) \rangle$ 
416     apply (AOT_subst  $\langle \psi\{a\} \ \& \ \mathcal{A}\varphi\{a\} \rangle \langle \mathcal{A}\psi\{a\} \ \& \ \mathcal{A}\varphi\{a\} \rangle$  for:  $\alpha$ )
417     using "Commutativity of  $\equiv$ " "intro-elim:3:b" "sc-eq-fur:2"
418     " $\rightarrow$ E" rigid_condition by blast
419 next
420   AOT_assume  $\langle \exists!\alpha (\psi\{a\} \ \& \ \mathcal{A}\varphi\{a\}) \rangle$ 
421   AOT_hence  $\langle \exists!\alpha (\mathcal{A}\psi\{a\} \ \& \ \mathcal{A}\varphi\{a\}) \rangle$ 
422     apply (AOT_subst  $\langle \mathcal{A}\psi\{a\} \ \& \ \mathcal{A}\varphi\{a\} \rangle \langle \psi\{a\} \ \& \ \varphi\{a\} \rangle$  for:  $\alpha$ )
423     apply (meson "sc-eq-fur:2" " $\rightarrow$ E" rigid_condition)
424     by simp
425   AOT_hence  $\langle \exists!\alpha \mathcal{A}(\psi\{a\} \ \& \ \varphi\{a\}) \rangle$ 
426     apply (AOT_subst  $\langle \mathcal{A}(\psi\{a\} \ \& \ \varphi\{a\}) \rangle \langle \mathcal{A}\psi\{a\} \ \& \ \mathcal{A}\varphi\{a\} \rangle$  for:  $\alpha$ )
427     using "Act-Basic:2" apply presburger
428     by simp
429   AOT_thus  $\langle \mathcal{A}\exists!\alpha (\psi\{a\} \ \& \ \varphi\{a\}) \rangle$ 
430     by (metis "A-Exists:1" "intro-elim:3:b")
431 qed
432
433 end
434
435 (*<*)
436 end
437 (*>*)

```

A.10. Extended Relation Comprehension

```

1  theory AOT_ExtendedRelationComprehension
2    imports AOT_RestrictedVariables
3  begin
4
5  section<Extended Relation Comprehension>
6
7  text<This theory depends on choosing extended models.>
8  interpretation AOT_ExtendedModel by (standard; auto)
9
10 text<Auxiliary lemma: negations of denoting relations denote.>
11 AOT_theorem negation_denotes: <[ $\lambda x \varphi\{x\} \downarrow \rightarrow [\lambda x \neg\varphi\{x\} \downarrow$ >
12 proof(rule "→I")
13   AOT_assume 0: <[ $\lambda x \varphi\{x\} \downarrow$ >
14   AOT_show <[ $\lambda x \neg\varphi\{x\} \downarrow$ >
15   proof (rule "safe-ext"[axiom_inst, THEN "→E", OF "&I"])
16     AOT_show <[ $\lambda x \neg[\lambda x \varphi\{x\}]x \downarrow$ > by "cqt:2"
17   next
18     AOT_have < $\Box[\lambda x \varphi\{x\} \downarrow$ >
19     using 0 "exist-nec"[THEN "→E"] by blast
20     moreover AOT_have < $\Box[\lambda x \varphi\{x\} \downarrow \rightarrow \Box\forall x (\neg[\lambda x \varphi\{x\}]x \equiv \neg\varphi\{x\})$ >
21     by(rule RM; safe intro!: GEN "≡I" "→I" " $\beta\rightarrow C$ "(2) " $\beta\leftarrow C$ "(2) "cqt:2")
22     ultimately AOT_show < $\Box\forall x (\neg[\lambda x \varphi\{x\}]x \equiv \neg\varphi\{x\})$ >
23     using "→E" by blast
24   qed
25 qed
26
27 text<Auxiliary lemma: conjunctions of denoting relations denote.>
28 AOT_theorem conjunction_denotes: <[ $\lambda x \varphi\{x\} \downarrow \ \& \ [\lambda x \psi\{x\} \downarrow \rightarrow [\lambda x \varphi\{x\} \ \& \ \psi\{x\} \downarrow$ >
29 proof(rule "→I")
30   AOT_assume 0: <[ $\lambda x \varphi\{x\} \downarrow \ \& \ [\lambda x \psi\{x\} \downarrow$ >
31   AOT_show <[ $\lambda x \varphi\{x\} \ \& \ \psi\{x\} \downarrow$ >
32   proof (rule "safe-ext"[axiom_inst, THEN "→E", OF "&I"])
33     AOT_show <[ $\lambda x [\lambda x \varphi\{x\}]x \ \& \ [\lambda x \psi\{x\}]x \downarrow$ > by "cqt:2"
34   next
35     AOT_have < $\Box([\lambda x \varphi\{x\} \downarrow \ \& \ [\lambda x \psi\{x\} \downarrow)$ >
36     using 0 "exist-nec"[THEN "→E"] "&E"
37     "KBasic:3" "df-simplify:2" "intro-elim:3:b" by blast
38     moreover AOT_have
39     < $\Box([\lambda x \varphi\{x\} \downarrow \ \& \ [\lambda x \psi\{x\} \downarrow) \rightarrow \Box\forall x ([\lambda x \varphi\{x\}]x \ \& \ [\lambda x \psi\{x\}]x \equiv \varphi\{x\} \ \& \ \psi\{x\})$ >
40     by(rule RM; auto intro!: GEN "≡I" "→I" "cqt:2" "&I"
41         intro: " $\beta\leftarrow C$ "
42         dest: "&E" " $\beta\rightarrow C$ ")
43     ultimately AOT_show < $\Box\forall x ([\lambda x \varphi\{x\}]x \ \& \ [\lambda x \psi\{x\}]x \equiv \varphi\{x\} \ \& \ \psi\{x\})$ >
44     using "→E" by blast
45   qed
46 qed
47
48 AOT_register_rigid_restricted_type
49 Ordinary: <O! $\kappa$ >
50 proof
51   AOT_modally_strict {
52     AOT_show < $\exists x O!x$ >
53     by (meson "B $\diamond$ " "T $\diamond$ " "o-objects-exist:1" "→E")
54   }
55 next
56   AOT_modally_strict {
57     AOT_show <O! $\kappa \rightarrow \kappa \downarrow$ > for  $\kappa$ 
58     by (simp add: "→I" "cqt:5:a[1]"[axiom_inst, THEN "→E", THEN "&E"(2)])
59   }
60 next
61   AOT_modally_strict {

```

```

62   AOT_show <∀α(0!α → □0!α)>
63     by (simp add: GEN "oa-facts:1")
64   }
65 qed
66
67 AOT_register_variable_names
68   Ordinary: u v r t s
69
70 text<In PLM this is defined in the Natural Numbers chapter,
71     but since it is helpful for stating the comprehension principles,
72     we already define it here.>
73 AOT_define eqE :: <τ ⇒ τ ⇒ φ> (infixl <≡E> 50)
74   eqE: <F ≡E G ≡df F↓ & G↓ & ∀u ([F]u ≡ [G]u)> (738)
75
76 text<Derive existence claims about relations from the axioms.>
77 AOT_theorem denotes_all: <[λx ∀G (□G ≡E F → x[G])↓>
78   and denotes_all_neg: <[λx ∀G (□G ≡E F → ¬x[G])↓>
79 proof -
80   AOT_have Aux: <∀F (□F ≡E G → (x[F] ≡ x[G])), ¬(x[G] ≡ y[G])
81     ⊢□ ∃F([F]x & ¬[F]y)> for x y G
82   proof -
83     AOT_modally_strict {
84       AOT_assume 0: <∀F (□F ≡E G → (x[F] ≡ x[G]))>
85       AOT_assume G_prop: <¬(x[G] ≡ y[G])>
86       {
87         AOT_assume <¬∃F([F]x & ¬[F]y)>
88         AOT_hence 0: <∀F ¬([F]x & ¬[F]y)>
89         by (metis "cqt-further:4" "→E")
90         AOT_have <∀F ([F]x ≡ [F]y)>
91         proof (rule GEN; rule "≡I"; rule "→I")
92           fix F
93           AOT_assume <[F]x>
94           moreover AOT_have <¬([F]x & ¬[F]y)>
95             using 0[THEN "∀E"(2)] by blast
96           ultimately AOT_show <[F]y>
97             by (metis "&I" "raa-cor:1")
98         next
99         fix F
100        AOT_assume <[F]y>
101        AOT_hence <¬[λx ¬[F]x]y>
102        by (metis "¬¬I" "β→C"(2))
103        moreover AOT_have <¬([λx ¬[F]x]x & ¬[λx ¬[F]x]y)>
104        apply (rule 0[THEN "∀E"(1)]) by "cqt:2[lambda]"
105        ultimately AOT_have 1: <¬[λx ¬[F]x]x>
106        by (metis "&I" "raa-cor:3")
107        {
108          AOT_assume <¬[F]x>
109          AOT_hence <[λx ¬[F]x]x>
110          by (auto intro!: "β←C"(1) "cqt:2")
111          AOT_hence <p & ¬p> for p
112          using 1 by (metis "raa-cor:3")
113        }
114        AOT_thus <[F]x> by (metis "raa-cor:1")
115      qed
116      AOT_hence <□∀F ([F]x ≡ [F]y)>
117      using "ind-nec"[THEN "→E"] by blast
118      AOT_hence <∀F □([F]x ≡ [F]y)>
119      by (metis "CBF" "→E")
120    } note indistI = this
121    {
122      AOT_assume G_prop: <x[G] & ¬y[G]>
123      AOT_hence Ax: <A!x>
124      using "&E"(1) "∃I"(2) "→E" "encoders-are-abstract" by blast

```

```

125
126 {
127   AOT_assume Ay: <A!y>
128   {
129     fix F
130     {
131       AOT_assume <∀u□([F]u ≡ [G]u)>
132       AOT_hence <□∀u([F]u ≡ [G]u)>
133         using "Ordinary.res-var-bound-reas[BF]"[THEN "→E"] by simp
134       AOT_hence <□F ≡E G>
135         by (AOT_subst <F ≡E G> <∀u ([F]u ≡ [G]u)>)
136           (auto intro!: "eqE"[THEN "≡Df", THEN "≡S"(1), OF "&I"] "cqt:2")
137       AOT_hence <x[F] ≡ x[G]>
138         using 0[THEN "∀E"(2)] "≡E" "→E" by meson
139       AOT_hence <x[F]>
140         using G_prop "&E" "≡E" by blast
141     }
142     AOT_hence <∀u□([F]u ≡ [G]u) → x[F]>
143       by (rule "→I")
144   }
145   AOT_hence xprop: <∀F(∀u□([F]u ≡ [G]u) → x[F])>
146     by (rule GEN)
147   moreover AOT_have yprop: <¬∀F(∀u□([F]u ≡ [G]u) → y[F])>
148   proof (rule "raa-cor:2")
149     AOT_assume <∀F(∀u□([F]u ≡ [G]u) → y[F])>
150     AOT_hence <∀F(□∀u([F]u ≡ [G]u) → y[F])>
151       apply (AOT_subst <□∀u([F]u ≡ [G]u)> <∀u□([F]u ≡ [G]u)> for: F)
152       using "Ordinary.res-var-bound-reas[BF]"
153         "Ordinary.res-var-bound-reas[CBF]"
154         "intro-elim:2" apply presburger
155     by simp
156     AOT_hence A: <∀F(□F ≡E G → y[F])>
157       by (AOT_subst <F ≡E G> <∀u ([F]u ≡ [G]u)> for: F)
158         (auto intro!: "eqE"[THEN "≡Df", THEN "≡S"(1), OF "&I"] "cqt:2")
159     moreover AOT_have <□G ≡E G>
160       by (auto intro!: "eqE"[THEN "≡dfI"] "cqt:2" RN "&I" GEN "→I" "≡I")
161     ultimately AOT_have <y[G]> using "∀E"(2) "→E" by blast
162     AOT_thus <p & ¬p> for p using G_prop "&E" by (metis "raa-cor:3")
163   qed
164   AOT_have <∃F([F]x & ¬[F]y)>
165   proof(rule "raa-cor:1")
166     AOT_assume <¬∃F([F]x & ¬[F]y)>
167     AOT_hence indist: <∀F □([F]x ≡ [F]y)> using indistI by blast
168     AOT_have <∀F(∀u□([F]u ≡ [G]u) → y[F])>
169       using indistinguishable_ord_enc_all[axiom_inst, THEN "→E", OF "&I",
170         OF "&I", OF "&I", OF "cqt:2[const_var]"[axiom_inst],
171         OF Ax, OF Ay, OF indist, THEN "≡E"(1), OF xprop].
172     AOT_thus <∀F(∀u□([F]u ≡ [G]u) → y[F]) & ¬∀F(∀u□([F]u ≡ [G]u) → y[F])>
173       using yprop "&I" by blast
174   qed
175 }
176 moreover {
177   AOT_assume notAy: <¬A!y>
178   AOT_have <∃F([F]x & ¬[F]y)>
179     apply (rule "∃I"(1)[where τ=«A!»])
180     using Ax notAy "&I" apply blast
181     by (simp add: "oa-exist:2")
182 }
183 ultimately AOT_have <∃F([F]x & ¬[F]y)>
184   by (metis "raa-cor:1")
185 }
186 moreover {
187   AOT_assume G_prop: <¬x[G] & y[G]>

```



```

188 AOT_hence Ay: <A!y>
189   by (meson "&E"(2) "encoders-are-abstract" "existential:2[const_var]" "→E")
190 AOT_hence not0y: <¬0!y>
191   using "≡E"(1) "oa-contingent:3" by blast
192 {
193   AOT_assume Ax: <A!x>
194   {
195     fix F
196     {
197       AOT_assume <□∀u([F]u ≡ [G]u)>
198       AOT_hence <□F ≡E G>
199         by (AOT_subst <F ≡E G> <∀u([F]u ≡ [G]u)>)
200           (auto intro!: "eqE"[THEN "≡Df", THEN "≡S"(1), OF "&I"] "cqt:2")
201       AOT_hence <x[F] ≡ x[G]>
202         using 0[THEN "∀E"(2)] "≡E" "→E" by meson
203       AOT_hence <¬x[F]>
204         using G_prop "&E" "≡E" by blast
205     }
206     AOT_hence <□∀u([F]u ≡ [G]u) → ¬x[F]>
207       by (rule "→I")
208   }
209   AOT_hence x_prop: <∀F(□∀u([F]u ≡ [G]u) → ¬x[F])>
210     by (rule GEN)
211   AOT_have x_prop: <¬∃F(∀u□([F]u ≡ [G]u) & x[F])>
212   proof (rule "raa-cor:2")
213     AOT_assume <∃F(∀u □([F]u ≡ [G]u) & x[F])>
214     then AOT_obtain F where F_prop: <∀u □([F]u ≡ [G]u) & x[F]>
215       using "∃E"[rotated] by blast
216     AOT_have <□([F]u ≡ [G]u)> for u
217       using F_prop[THEN "&E"(1), THEN "Ordinary.∀E"].
218     AOT_hence <∀u □([F]u ≡ [G]u)>
219       by (rule Ordinary.GEN)
220     AOT_hence <□∀u([F]u ≡ [G]u)>
221       by (metis "Ordinary.res-var-bound-reas[BF]" "→E")
222     AOT_hence <¬x[F]>
223       using x_prop[THEN "∀E"(2), THEN "→E"] by blast
224     AOT_thus <p & ¬p> for p
225       using F_prop[THEN "&E"(2)] by (metis "raa-cor:3")
226   qed
227   AOT_have y_prop: <∃F(∀u □([F]u ≡ [G]u) & y[F])>
228   proof (rule "raa-cor:1")
229     AOT_assume <¬∃F (∀u □([F]u ≡ [G]u) & y[F])>
230     AOT_hence 0: <∀F ¬(∀u □([F]u ≡ [G]u) & y[F])>
231       using "cqt-further:4"[THEN "→E"] by blast
232     {
233       fix F
234       {
235         AOT_assume <∀u □([F]u ≡ [G]u)>
236         AOT_hence <¬y[F]>
237           using 0[THEN "∀E"(2)] "&I" "raa-cor:1" by meson
238       }
239       AOT_hence <(∀u □([F]u ≡ [G]u) → ¬y[F])>
240         by (rule "→I")
241     }
242     AOT_hence A: <∀F(∀u □([F]u ≡ [G]u) → ¬y[F])>
243       by (rule GEN)
244     moreover AOT_have <∀u □([G]u ≡ [G]u)>
245       by (simp add: RN "oth-class-taut:3:a" "universal-cor" "→I")
246     ultimately AOT_have <¬y[G]>
247       using "∀E"(2) "→E" by blast
248     AOT_thus <p & ¬p> for p
249       using G_prop "&E" by (metis "raa-cor:3")
250   qed

```

```

251   AOT_have <∃F([F]x & ¬[F]y)>
252   proof(rule "raa-cor:1")
253     AOT_assume <¬∃F([F]x & ¬[F]y)>
254     AOT_hence indist: <∀F □([F]x ≡ [F]y)>
255     using indistI by blast
256     AOT_thus <∃F(∀u □([F]u ≡ [G]u) & x[F]) & ¬∃F(∀u □([F]u ≡ [G]u) & x[F])>
257     using indistinguishable_ord_enc_ex[axiom_inst, THEN "→E", OF "&I",
258       OF "&I", OF "&I", OF "cqt:2[const_var]"[axiom_inst],
259       OF Ax, OF Ay, OF indist, THEN "≡E"(2), OF y_prop]
260     x_prop "&I" by blast
261   qed
262 }
263 moreover {
264   AOT_assume notAx: <¬A!x>
265   AOT_hence 0x: <0!x>
266   using "VE"(3) "oa-exist:3" by blast
267   AOT_have <∃F([F]x & ¬[F]y)>
268   apply (rule "EI"(1)[where τ=<<0!>>])
269   using 0x not0y "&I" apply blast
270   by (simp add: "oa-exist:1")
271 }
272 ultimately AOT_have <∃F([F]x & ¬[F]y)>
273   by (metis "raa-cor:1")
274 }
275 ultimately AOT_show <∃F([F]x & ¬[F]y)>
276   using G_prop by (metis "&I" "→I" "≡I" "raa-cor:1")
277 }
278 qed
279
280 AOT_modally_strict {
281   fix x y
282   AOT_assume indist: <∀F ([F]x ≡ [F]y)>
283   AOT_hence nec_indist: <□∀F ([F]x ≡ [F]y)>
284   using "ind-nec" "vdash-properties:10" by blast
285   AOT_hence indist_nec: <∀F □([F]x ≡ [F]y)>
286   using "CBF" "vdash-properties:10" by blast
287   AOT_assume 0: <∀G (□G ≡E F → x[G])>
288   AOT_hence 1: <∀G (□∀u ([G]u ≡ [F]u) → x[G])>
289   by (AOT_subst (reverse) <∀u ([G]u ≡ [F]u)> <G ≡E F> for: G)
290   (auto intro!: "eqE"[THEN "≡df", THEN "≡S"(1), OF "&I"] "cqt:2")
291   AOT_have <x[F]>
292   by (safe intro!: 1[THEN "VE"(2), THEN "→E"] GEN "→I" RN "≡I")
293   AOT_have <∀G (□G ≡E F → y[G])>
294   proof(rule "raa-cor:1")
295     AOT_assume <¬∀G (□G ≡E F → y[G])>
296     AOT_hence <∃G ¬(□G ≡E F → y[G])>
297     using "cqt-further:2" "→E" by blast
298     then AOT_obtain G where G_prop: <¬(□G ≡E F → y[G])>
299     using "∃E"[rotated] by blast
300     AOT_hence 1: <□G ≡E F & ¬y[G]>
301     by (metis "≡E"(1) "oth-class-taut:1:b")
302     AOT_have xG: <x[G]>
303     using 0[THEN "VE"(2), THEN "→E"] 1[THEN "&E"(1)] by blast
304     AOT_hence <x[G] & ¬y[G]>
305     using 1[THEN "&E"(2)] "&I" by blast
306     AOT_hence B: <¬(x[G] ≡ y[G])>
307     using "&E"(2) "≡E"(1) "reductio-aa:1" xG by blast
308   {
309     fix H
310     {
311       AOT_assume <□H ≡E G>
312       AOT_hence <□(H ≡E G & G ≡E F)>
313       using 1 by (metis "KBasic:3" "con-dis-i-e:1" "con-dis-i-e:2:a")

```

```

314         "intro-elim:3:b")
315 moreover AOT_have <□(H ≡E G & G ≡E F) → □(H ≡E F)>
316 proof(rule RM)
317   AOT_modally_strict {
318     AOT_show <H ≡E G & G ≡E F → H ≡E F>
319     proof (safe intro!: "→I" "eqE"[THEN "≡dfI"] "&I" "cqt:2" Ordinary.GEN)
320       fix u
321       AOT_assume <H ≡E G & G ≡E F>
322       AOT_hence <∀u ([H]u ≡ [G]u) and <∀u ([G]u ≡ [F]u)>
323         using "eqE"[THEN "≡dfE"] "&E" by blast+
324       AOT_thus <[H]u ≡ [F]u>
325         by (auto dest!: "Ordinary.∀E" dest: "≡E")
326     qed
327   }
328   qed
329   ultimately AOT_have <□(H ≡E F)>
330     using "→E" by blast
331   AOT_hence <x[H]>
332     using 0[THEN "∀E"(2)] "→E" by blast
333   AOT_hence <x[H] ≡ x[G]>
334     using xG "≡I" "→I" by blast
335 }
336 AOT_hence <□H ≡E G → (x[H] ≡ x[G])> by (rule "→I")
337 }
338 AOT_hence A: <∀H(□H ≡E G → (x[H] ≡ x[G]))>
339   by (rule GEN)
340 then AOT_obtain F where F_prop: <[F]x & ¬[F]y>
341   using Aux[OF A, OF B] "∃E"[rotated] by blast
342 moreover AOT_have <[F]y>
343   using indist[THEN "∀E"(2), THEN "≡E"(1), OF F_prop[THEN "&E"(1)]] .
344 AOT_thus <p & ¬p> for p
345   using F_prop[THEN "&E"(2)] by (metis "raa-cor:3")
346 qed
347 } note 0 = this
348 AOT_modally_strict {
349   fix x y
350   AOT_assume <∀F ([F]x ≡ [F]y)>
351   moreover AOT_have <∀F ([F]y ≡ [F]x)>
352     by (metis calculation "cqt-basic:11" "≡E"(2))
353   ultimately AOT_have <∀G (□G ≡E F → x[G]) ≡ ∀G (□G ≡E F → y[G])>
354     using 0 "≡I" "→I" by auto
355 } note 1 = this
356 AOT_show <[λx ∀G (□G ≡E F → x[G])]>
357   by (safe intro!: RN GEN "→I" 1 "kirchner-thm:2"[THEN "≡E"(2)])
358
359 AOT_modally_strict {
360   fix x y
361   AOT_assume indist: <∀F ([F]x ≡ [F]y)>
362   AOT_hence nec_indist: <□∀F ([F]x ≡ [F]y)>
363     using "ind-nec" "vdash-properties:10" by blast
364   AOT_hence indist_nec: <∀F □([F]x ≡ [F]y)>
365     using "CBF" "vdash-properties:10" by blast
366   AOT_assume 0: <∀G (□G ≡E F → ¬x[G])>
367   AOT_hence 1: <∀G (□∀u ([G]u ≡ [F]u) → ¬x[G])>
368     by (AOT_subst (reverse) <∀u ([G]u ≡ [F]u)> <G ≡E F> for: G)
369     (auto intro!: "eqE"[THEN "≡df", THEN "≡S"(1), OF "&I"] "cqt:2")
370   AOT_have <¬x[F]>
371     by (safe intro!: 1[THEN "∀E"(2), THEN "→E"] GEN "→I" RN "≡I")
372   AOT_have <∀G (□G ≡E F → ¬y[G])>
373   proof(rule "raa-cor:1")
374     AOT_assume <¬∀G (□G ≡E F → ¬y[G])>
375     AOT_hence <∃G ¬(□G ≡E F → ¬y[G])>
376     using "cqt-further:2" "→E" by blast

```

```

377   then AOT_obtain G where G_prop: <¬(□G ≡E F → ¬y[G])>
378     using "∃E"[rotated] by blast
379   AOT_hence 1: <□G ≡E F & ¬¬y[G]>
380     by (metis "≡E"(1) "oth-class-taut:1:b")
381   AOT_hence yG: <y[G]>
382     using G_prop "→I" "raa-cor:3" by blast
383   moreover AOT_hence 12: <¬x[G]>
384     using 0[THEN "∀E"(2), THEN "→E"] 1[THEN "&E"(1)] by blast
385   ultimately AOT_have <¬x[G] & y[G]>
386     using "&I" by blast
387   AOT_hence B: <¬(x[G] ≡ y[G])>
388     by (metis "12" "≡E"(3) "raa-cor:3" yG)
389   {
390     fix H
391     {
392       AOT_assume 3: <□H ≡E G>
393       AOT_hence <□(H ≡E G & G ≡E F)>
394         using 1
395         by (metis "KBasic:3" "con-dis-i-e:1" "→I" "intro-elim:3:b"
396             "reductio-aa:1" G_prop)
397       moreover AOT_have <□(H ≡E G & G ≡E F) → □(H ≡E F)>
398       proof (rule RM)
399         AOT_modally_strict {
400           AOT_show <H ≡E G & G ≡E F → H ≡E F>
401           proof (safe intro!: "→I" "eqE"[THEN "≡dfI"] "&I" "cqt:2" Ordinary.GEN)
402             fix u
403             AOT_assume <H ≡E G & G ≡E F>
404             AOT_hence <∀u ([H]u ≡ [G]u)> and <∀u ([G]u ≡ [F]u)>
405               using "eqE"[THEN "≡dfE"] "&E" by blast+
406             AOT_thus <[H]u ≡ [F]u>
407               by (auto dest!: "Ordinary.∀E" dest: "≡E")
408           qed
409         }
410       qed
411       ultimately AOT_have <□(H ≡E F)>
412         using "→E" by blast
413       AOT_hence <¬x[H]>
414         using 0[THEN "∀E"(2)] "→E" by blast
415       AOT_hence <x[H] ≡ x[G]>
416         using 12 "≡I" "→I" by (metis "raa-cor:3")
417     }
418     AOT_hence <□H ≡E G → (x[H] ≡ x[G])>
419       by (rule "→I")
420   }
421   AOT_hence A: <∀H(□H ≡E G → (x[H] ≡ x[G]))>
422     by (rule GEN)
423   then AOT_obtain F where F_prop: <[F]x & ¬[F]y>
424     using Aux[OF A, OF B] "∃E"[rotated] by blast
425   moreover AOT_have <[F]y>
426     using indist[THEN "∀E"(2), THEN "≡E"(1), OF F_prop[THEN "&E"(1)]] .
427   AOT_thus <p & ¬p> for p
428     using F_prop[THEN "&E"(2)] by (metis "raa-cor:3")
429   qed
430 } note 0 = this
431 AOT_modally_strict {
432   fix x y
433   AOT_assume <∀F ([F]x ≡ [F]y)>
434   moreover AOT_have <∀F ([F]y ≡ [F]x)>
435     by (metis calculation "cqt-basic:11" "≡E"(2))
436   ultimately AOT_have <∀G (□G ≡E F → ¬x[G]) ≡ ∀G (□G ≡E F → ¬y[G])>
437     using 0 "≡I" "→I" by auto
438 } note 1 = this
439 AOT_show <[λx ∀G (□G ≡E F → ¬x[G])]↓>

```

```

440   by (safe intro!: RN GEN "→I" 1 "kirchner-thm:2"[THEN "≡E"(2)])
441 qed
442
443 text<Reformulate the existence claims in terms of their negations.>
444
445 AOT_theorem denotes_ex: <[λx ∃G (□G ≡E F & x[G])]>↓>
446 proof (rule "safe-ext"[axiom_inst, THEN "→E", OF "&I"])
447   AOT_show <[λx ¬∀G (□G ≡E F → ¬x[G])]>↓>
448   using denotes_all_neg[THEN negation_denotes[THEN "→E"]].
449 next
450   AOT_show <□∀x (¬∀G (□G ≡E F → ¬x[G]) ≡ ∃G (□G ≡E F & x[G]))>
451   by (AOT_subst <□G ≡E F & x[G]> <¬(□G ≡E F → ¬x[G])> for: G x)
452     (auto simp: "conventions:1" "rule-eq-df:1"
453       intro: "oth-class-taut:4:b"[THEN "≡E"(2)]
454       "intro-elim:3:f"[OF "cqt-further:3", OF "oth-class-taut:3:b"]
455       intro!: RN GEN)
456 qed
457
458 AOT_theorem denotes_ex_neg: <[λx ∃G (□G ≡E F & ¬x[G])]>↓>
459 proof (rule "safe-ext"[axiom_inst, THEN "→E", OF "&I"])
460   AOT_show <[λx ¬∀G (□G ≡E F → x[G])]>↓>
461   using denotes_all[THEN negation_denotes[THEN "→E"]].
462 next
463   AOT_show <□∀x (¬∀G (□G ≡E F → x[G]) ≡ ∃G (□G ≡E F & ¬x[G]))>
464   by (AOT_subst (reverse) <□G ≡E F & ¬x[G]> <¬(□G ≡E F → x[G])> for: G x)
465     (auto simp: "oth-class-taut:1:b"
466       intro: "oth-class-taut:4:b"[THEN "≡E"(2)]
467       "intro-elim:3:f"[OF "cqt-further:3", OF "oth-class-taut:3:b"]
468       intro!: RN GEN)
469 qed
470
471 text<Derive comprehension principles.>
472
473 AOT_theorem Comprehension_1:
474   shows <□∀F∀G(□G ≡E F → (φ{F} ≡ φ{G})) → [λx ∃F (φ{F} & x[F])]>↓>
475 proof(rule "→I")
476   AOT_assume asm: <□∀F∀G(□G ≡E F → (φ{F} ≡ φ{G}))>
477   AOT_modally_strict {
478     fix x y
479     AOT_assume 0: <∀F∀G (□G ≡E F → (φ{F} ≡ φ{G}))>
480     AOT_assume indist: <∀F ([F]x ≡ [F]y)>
481     AOT_assume x_prop: <∃F (φ{F} & x[F])>
482     then AOT_obtain F where F_prop: <φ{F} & x[F]>
483     using "∃E"[rotated] by blast
484     AOT_hence <□F ≡E F & x[F]>
485     by (auto intro!: RN eqE[THEN "≡defI"] "&I" "cqt:2" GEN "≡I" "→I" dest: "&E")
486     AOT_hence <∃G(□G ≡E F & x[G])>
487     by (rule "∃I")
488     AOT_hence <[λx ∃G(□G ≡E F & x[G])>x>
489     by (safe intro!: "β←C" denotes_ex "cqt:2")
490     AOT_hence <[λx ∃G(□G ≡E F & x[G])>y>
491     using indist[THEN "∀E"(1), OF denotes_ex, THEN "≡E"(1)] by blast
492     AOT_hence <∃G(□G ≡E F & y[G])>
493     using "β→C" by blast
494     then AOT_obtain G where <□G ≡E F & y[G]>
495     using "∃E"[rotated] by blast
496     AOT_hence <φ{G} & y[G]>
497     using 0[THEN "∀E"(2), THEN "∀E"(2), THEN "→E", THEN "≡E"(1)]
498     F_prop[THEN "&E"(1)] "&E" "&I" by blast
499     AOT_hence <∃F (φ{F} & y[F])>
500     by (rule "∃I")
501   } note 1 = this
502   AOT_modally_strict {

```

```

503   AOT_assume 0: <∀F∀G (□G ≡E F → (φ{F} ≡ φ{G}))>
504   {
505     fix x y
506     {
507       AOT_assume <∀F ([F]x ≡ [F]y)>
508       moreover AOT_have <∀F ([F]y ≡ [F]x)>
509         by (metis calculation "cqt-basic:11" "≡E"(1))
510       ultimately AOT_have <∃F (φ{F} & x[F]) ≡ ∃F (φ{F} & y[F])>
511         using 0 1[OF 0] "≡I" "→I" by simp
512     }
513     AOT_hence <∀F ([F]x ≡ [F]y) → (∃F (φ{F} & x[F]) ≡ ∃F (φ{F} & y[F]))>
514       using "→I" by blast
515   }
516   AOT_hence <∀x∀y(∀F ([F]x ≡ [F]y) → (∃F (φ{F} & x[F]) ≡ ∃F (φ{F} & y[F])))>
517     by (auto intro!: GEN)
518 } note 1 = this
519 AOT_hence <⊢□ ∀F∀G (□G ≡E F → (φ{F} ≡ φ{G})) →
520   ∀x∀y(∀F ([F]x ≡ [F]y) → (∃F (φ{F} & x[F]) ≡ ∃F (φ{F} & y[F])))>
521   by (rule "→I")
522 AOT_hence <□∀F∀G (□G ≡E F → (φ{F} ≡ φ{G})) →
523   □∀x∀y(∀F ([F]x ≡ [F]y) → (∃F (φ{F} & x[F]) ≡ ∃F (φ{F} & y[F])))>
524   by (rule RM)
525 AOT_hence <□∀x∀y(∀F ([F]x ≡ [F]y) → (∃F (φ{F} & x[F]) ≡ ∃F (φ{F} & y[F])))>
526   using "→E" assm by blast
527 AOT_thus <[λx ∃F (φ{F} & x[F])]↓>
528   by (safe intro!: "kirchner-thm:2"[THEN "≡E"(2)])
529 qed
530
531 AOT_theorem Comprehension_2:
532   shows <□∀F∀G(□G ≡E F → (φ{F} ≡ φ{G})) → [λx ∃F (φ{F} & ¬x[F])]↓>
533   proof(rule "→I")
534     AOT_assume assm: <□∀F∀G(□G ≡E F → (φ{F} ≡ φ{G}))>
535     AOT_modally_strict {
536       fix x y
537       AOT_assume 0: <∀F∀G (□G ≡E F → (φ{F} ≡ φ{G}))>
538       AOT_assume indist: <∀F ([F]x ≡ [F]y)>
539       AOT_assume x_prop: <∃F (φ{F} & ¬x[F])>
540       then AOT_obtain F where F_prop: <φ{F} & ¬x[F]>
541         using "∃E"[rotated] by blast
542       AOT_hence <□F ≡E F & ¬x[F]>
543         by (auto intro!: RN eqE[THEN "≡dfI"] "&I" "cqt:2" GEN "≡I" "→I" dest: "&E")
544       AOT_hence <∃G(□G ≡E F & ¬x[G])>
545         by (rule "∃I")
546       AOT_hence <[λx ∃G(□G ≡E F & ¬x[G])]x>
547         by (safe intro!: "β←C" denotes_ex_neg "cqt:2")
548       AOT_hence <[λx ∃G(□G ≡E F & ¬x[G])]y>
549         using indist[THEN "∀E"(1), OF denotes_ex_neg, THEN "≡E"(1)] by blast
550       AOT_hence <∃G(□G ≡E F & ¬y[G])>
551         using "β→C" by blast
552       then AOT_obtain G where <□G ≡E F & ¬y[G]>
553         using "∃E"[rotated] by blast
554       AOT_hence <φ{G} & ¬y[G]>
555         using 0[THEN "∀E"(2), THEN "∀E"(2), THEN "→E", THEN "≡E"(1)]
556         F_prop[THEN "&E"(1)] "&E" "&I" by blast
557       AOT_hence <∃F (φ{F} & ¬y[F])>
558         by (rule "∃I")
559     } note 1 = this
560     AOT_modally_strict {
561       AOT_assume 0: <∀F∀G (□G ≡E F → (φ{F} ≡ φ{G}))>
562       {
563         fix x y
564         {
565           AOT_assume <∀F ([F]x ≡ [F]y)>

```

```

566     moreover AOT_have <∀F ([F]y ≡ [F]x)>
567     by (metis calculation "cqt-basic:11" "≡E"(1))
568     ultimately AOT_have <∃F (φ{F} & ¬x[F]) ≡ ∃F (φ{F} & ¬y[F])>
569     using 0 1[OF 0] "≡I" "→I" by simp
570   }
571   AOT_hence <∀F ([F]x ≡ [F]y) → (∃F (φ{F} & ¬x[F]) ≡ ∃F (φ{F} & ¬y[F]))>
572   using "→I" by blast
573 }
574 AOT_hence <∀x∀y(∀F ([F]x ≡ [F]y) → (∃F (φ{F} & ¬x[F]) ≡ ∃F (φ{F} & ¬y[F])))>
575 by (auto intro!: GEN)
576 } note 1 = this
577 AOT_hence <⊢□ ∀FVG (□G ≡E F → (φ{F} ≡ φ{G})) →
578   ∀x∀y(∀F ([F]x ≡ [F]y) → (∃F (φ{F} & ¬x[F]) ≡ ∃F (φ{F} & ¬y[F])))>
579 by (rule "→I")
580 AOT_hence <□∀FVG (□G ≡E F → (φ{F} ≡ φ{G})) →
581   □∀x∀y(∀F ([F]x ≡ [F]y) → (∃F (φ{F} & ¬x[F]) ≡ ∃F (φ{F} & ¬y[F])))>
582 by (rule RM)
583 AOT_hence <□∀x∀y(∀F ([F]x ≡ [F]y) → (∃F (φ{F} & ¬x[F]) ≡ ∃F (φ{F} & ¬y[F])))>
584 using "→E" assm by blast
585 AOT_thus <[λx ∃F (φ{F} & ¬x[F])]↓>
586 by (safe intro!: "kirchner-thm:2"[THEN "≡E"(2)])
587 qed
588
589 text<Derived variants of the comprehension principles above.>
590
591 AOT_theorem Comprehension_1':
592 shows <□∀FVG(□G ≡E F → (φ{F} ≡ φ{G})) → [λx ∀F (x[F] → φ{F})]↓>
593 proof(rule "→I")
594   AOT_assume <□∀FVG(□G ≡E F → (φ{F} ≡ φ{G}))>
595   AOT_hence 0: <□∀FVG(□G ≡E F → (¬φ{F} ≡ ¬φ{G}))>
596   by (AOT_subst (reverse) <¬φ{F} ≡ ¬φ{G}> <φ{F} ≡ φ{G}> for: F G)
597   (auto simp: "oth-class-taut:4:b")
598   AOT_show <[λx ∀F (x[F] → φ{F})]↓>
599   proof(rule "safe-ext"[axiom_inst, THEN "→E", OF "&I"])
600     AOT_show <[λx ¬∃F(¬φ{F} & x[F])]↓>
601     using Comprehension_1[THEN "→E", OF 0, THEN negation_denotes[THEN "→E"]].
602   next
603     AOT_show <□∀x (¬∃F (¬φ{F} & x[F]) ≡ ∀F (x[F] → φ{F}))>
604     by (AOT_subst (reverse) <¬φ{F} & x[F]> <¬(x[F] → φ{F})> for: F x)
605     (auto simp: "oth-class-taut:1:b"[THEN "intro-elim:3:e",
606       OF "oth-class-taut:2:a"]
607       intro: "intro-elim:3:f"[OF "cqt-further:3", OF "oth-class-taut:3:a",
608         symmetric]
609       intro!: RN GEN)
610   qed
611 qed
612
613 AOT_theorem Comprehension_2':
614 shows <□∀FVG(□G ≡E F → (φ{F} ≡ φ{G})) → [λx ∀F (φ{F} → x[F])]↓>
615 proof(rule "→I")
616   AOT_assume 0: <□∀FVG(□G ≡E F → (φ{F} ≡ φ{G}))>
617   AOT_show <[λx ∀F (φ{F} → x[F])]↓>
618   proof(rule "safe-ext"[axiom_inst, THEN "→E", OF "&I"])
619     AOT_show <[λx ¬∃F(φ{F} & ¬x[F])]↓>
620     using Comprehension_2[THEN "→E", OF 0, THEN negation_denotes[THEN "→E"]].
621   next
622     AOT_show <□∀x (¬∃F (φ{F} & ¬x[F]) ≡ ∀F (φ{F} → x[F]))>
623     by (AOT_subst (reverse) <φ{F} & ¬x[F]> <¬(φ{F} → x[F])> for: F x)
624     (auto simp: "oth-class-taut:1:b"
625       intro: "intro-elim:3:f"[OF "cqt-further:3", OF "oth-class-taut:3:a",
626         symmetric]
627       intro!: RN GEN)
628   qed

```

```

629 qed
630
631 text<Derive a combined comprehension principles.>
632
633 AOT_theorem Comprehension_3:
634   <math display="block">\Box \forall F \forall G (\Box \equiv_E F \rightarrow (\varphi\{F\} \equiv \varphi\{G\})) \rightarrow [\lambda x \forall F (x[F] \equiv \varphi\{F\})] \downarrow \rangle
635 proof(rule "→I")
636   AOT_assume 0: <math display="block">\Box \forall F \forall G (\Box \equiv_E F \rightarrow (\varphi\{F\} \equiv \varphi\{G\})) \rangle
637   AOT_show <math display="block">[\lambda x \forall F (x[F] \equiv \varphi\{F\})] \downarrow \rangle
638   proof(rule "safe-ext"[axiom_inst, THEN "→E", OF "&I"])
639     AOT_show <math display="block">[\lambda x \forall F (x[F] \rightarrow \varphi\{F\}) \& \forall F (\varphi\{F\} \rightarrow x[F])] \downarrow \rangle
640     by (safe intro!: conjunction_denotes[THEN "→E", OF "&I"]
641         Comprehension_1'[THEN "→E"]
642         Comprehension_2'[THEN "→E"] 0)
643   next
644     AOT_show <math display="block">\Box \forall x (\forall F (x[F] \rightarrow \varphi\{F\}) \& \forall F (\varphi\{F\} \rightarrow x[F]) \equiv \forall F (x[F] \equiv \varphi\{F\})) \rangle
645     by (auto intro!: RN GEN "≡I" "→I" "&I" dest: "&E" "∀E"(2) "→E" "≡E"(1,2))
646   qed
647 qed
648
649 notepad
650 begin
651 text<Verify that the original axioms are equivalent to @{thm denotes_ex}
652   and @{thm denotes_ex_neg}.>
653 AOT_modally_strict {
654   fix x y H
655   AOT_have <math display="block">\Box x \& \Box y \& \forall F \Box ([F]x \equiv [F]y) \rightarrow
656   (\forall G (\forall z (\Box z \rightarrow \Box ([G]z \equiv [H]z)) \rightarrow x[G]) \equiv
657   \forall G (\forall z (\Box z \rightarrow \Box ([G]z \equiv [H]z)) \rightarrow y[G])) \rangle
658   proof(rule "→I")
659     {
660       fix x y
661       AOT_assume <math display="block">\Box x \rangle
662       AOT_assume <math display="block">\Box y \rangle
663       AOT_assume indist: <math display="block">\forall F \Box ([F]x \equiv [F]y) \rangle
664       AOT_assume <math display="block">\forall G (\forall u \Box ([G]u \equiv [H]u) \rightarrow x[G]) \rangle
665       AOT_hence <math display="block">\forall G (\Box \forall u ([G]u \equiv [H]u) \rightarrow x[G]) \rangle
666       using "Ordinary.res-var-bound-reas[BF]" "Ordinary.res-var-bound-reas[CBF]"
667         "intro-elim:2"
668       by (AOT_subst <math display="block">\Box \forall u ([G]u \equiv [H]u) \rangle \langle \forall u \Box ([G]u \equiv [H]u) \rangle \text{for: } G) \text{ auto}
669       AOT_hence <math display="block">\forall G (\Box \equiv_E H \rightarrow x[G]) \rangle
670       by (AOT_subst <math display="block">G \equiv_E H \rangle \langle \forall u ([G]u \equiv [H]u) \rangle \text{for: } G)
671         (safe intro!: "eqE"[THEN "≡Df", THEN "≡S"(1), OF "&I"] "cqt:2")
672       AOT_hence <math display="block">\neg \exists G (\Box \equiv_E H \& \neg x[G]) \rangle
673       by (AOT_subst (reverse) <math display="block">(\Box \equiv_E H \& \neg x[G]) \rangle \langle \neg (\Box \equiv_E H \rightarrow x[G]) \rangle \text{for: } G)
674         (auto simp: "oth-class-taut:1:b" "cqt-further:3"[THEN "≡E"(1)])
675       AOT_hence <math display="block">\neg [\lambda x \exists G (\Box \equiv_E H \& \neg x[G])] x \rangle
676       by (auto intro: "β→C")
677       AOT_hence <math display="block">\neg [\lambda x \exists G (\Box \equiv_E H \& \neg x[G])] y \rangle
678       using indist[THEN "∀E"(1), OF denotes_ex_neg,
679         THEN "qml:2"[axiom_inst, THEN "→E"],
680         THEN "≡E"(3)] by blast
681       AOT_hence <math display="block">\neg \exists G (\Box \equiv_E H \& \neg y[G]) \rangle
682       by (safe intro!: "β←C" denotes_ex_neg "cqt:2")
683       AOT_hence <math display="block">\forall G \neg (\Box \equiv_E H \& \neg y[G]) \rangle
684       using "cqt-further:4"[THEN "→E"] by blast
685       AOT_hence <math display="block">\forall G (\Box \equiv_E H \rightarrow y[G]) \rangle
686       by (AOT_subst <math display="block">\Box \equiv_E H \rightarrow y[G] \rangle \langle \neg (\Box \equiv_E H \& \neg y[G]) \rangle \text{for: } G)
687         (auto simp: "oth-class-taut:1:a")
688       AOT_hence <math display="block">\forall G (\Box \forall u ([G]u \equiv [H]u) \rightarrow y[G]) \rangle
689       by (AOT_subst (reverse) <math display="block">\forall u ([G]u \equiv [H]u) \rangle \langle G \equiv_E H \rangle \text{for: } G)
690         (safe intro!: "eqE"[THEN "≡Df", THEN "≡S"(1), OF "&I"] "cqt:2")
691       AOT_hence <math display="block">\forall G (\forall u \Box ([G]u \equiv [H]u) \rightarrow y[G]) \rangle

```



```

692     using "Ordinary.res-var-bound-reas[BF]" "Ordinary.res-var-bound-reas[CBF]"
693     "intro-elim:2"
694     by (AOT_subst <math>\forall u \square([G]u \equiv [H]u)> <math>\square \forall u ([G]u \equiv [H]u)> \text{for: } G \text{ auto}
695 } \text{ note } 0 = \text{this}
696 AOT_assume <math>\langle A!x \ \& \ A!y \ \& \ \forall F \square([F]x \equiv [F]y)>
697 AOT_hence <math>\langle A!x \rangle \ \text{and} \ \langle A!y \rangle \ \text{and} \ \langle \forall F \square([F]x \equiv [F]y) \rangle
698     using "&E" by blast+
699 moreover AOT_have <math>\langle \forall F \square([F]y \equiv [F]x) \rangle
700     using calculation(3)
701     apply (safe intro!: CBF[THEN "\to E"] dest!: BF[THEN "\to E"])
702     using "RM:3" "cqt-basic:11" "intro-elim:3:b" by fast
703 ultimately AOT_show <math>\langle \forall G (\forall u \square([G]u \equiv [H]u) \to x[G]) \equiv
704     \forall G (\forall u \square([G]u \equiv [H]u) \to y[G]) \rangle
705     using 0 by (auto intro!: "\equiv I" "\to I")
706 qed
707
708 AOT_have <math>\langle A!x \ \& \ A!y \ \& \ \forall F \square([F]x \equiv [F]y) \to
709 (\exists G (\forall z (0!z \to \square([G]z \equiv [H]z)) \ \& \ x[G]) \equiv \exists G (\forall z (0!z \to \square([G]z \equiv [H]z)) \ \& \ y[G])) \rangle
710 proof(rule "\to I")
711 {
712     fix x y
713     AOT_assume <math>\langle A!x \rangle
714     AOT_assume <math>\langle A!y \rangle
715     AOT_assume indist: <math>\langle \forall F \square([F]x \equiv [F]y) \rangle
716     AOT_assume x_prop: <math>\langle \exists G (\forall u \square([G]u \equiv [H]u) \ \& \ x[G]) \rangle
717     AOT_hence <math>\langle \exists G (\square \forall u ([G]u \equiv [H]u) \ \& \ x[G]) \rangle
718         using "Ordinary.res-var-bound-reas[BF]" "Ordinary.res-var-bound-reas[CBF]"
719         "intro-elim:2"
720         by (AOT_subst <math>\langle \square \forall u ([G]u \equiv [H]u) \rangle \langle \forall u \square([G]u \equiv [H]u) \rangle \text{for: } G \text{ auto}
721     AOT_hence <math>\langle \exists G (\square G \equiv_E H \ \& \ x[G]) \rangle
722         by (AOT_subst <math>\langle G \equiv_E H \rangle \langle \forall u ([G]u \equiv [H]u) \rangle \text{for: } G)
723         (safe intro!: "eqE"[THEN "\equiv Df", THEN "\equiv S"(1), OF "&I"] "cqt:2")
724     AOT_hence <math>\langle [\lambda x \exists G (\square G \equiv_E H \ \& \ x[G])]x \rangle
725         by (safe intro!: "\beta \leftarrow C" denotes_ex "cqt:2")
726     AOT_hence <math>\langle [\lambda x \exists G (\square G \equiv_E H \ \& \ x[G])]y \rangle
727         using indist[THEN "\forall E"(1), OF denotes_ex,
728             THEN "qml:2"[axiom_inst, THEN "\to E"],
729             THEN "\equiv E"(1)] by blast
730     AOT_hence <math>\langle \exists G (\square G \equiv_E H \ \& \ y[G]) \rangle
731         by (rule "\beta \to C")
732     AOT_hence <math>\langle \exists G (\square \forall u ([G]u \equiv [H]u) \ \& \ y[G]) \rangle
733         by (AOT_subst (reverse) <math>\langle \forall u ([G]u \equiv [H]u) \rangle \langle G \equiv_E H \rangle \text{for: } G)
734         (safe intro!: "eqE"[THEN "\equiv Df", THEN "\equiv S"(1), OF "&I"] "cqt:2")
735     AOT_hence <math>\langle \exists G (\forall u \square([G]u \equiv [H]u) \ \& \ y[G]) \rangle
736         using "Ordinary.res-var-bound-reas[BF]"
737         "Ordinary.res-var-bound-reas[CBF]"
738         "intro-elim:2"
739         by (AOT_subst <math>\langle \forall u \square([G]u \equiv [H]u) \rangle \langle \square \forall u ([G]u \equiv [H]u) \rangle \text{for: } G \text{ auto}
740 } \text{ note } 0 = \text{this}
741 AOT_assume <math>\langle A!x \ \& \ A!y \ \& \ \forall F \square([F]x \equiv [F]y) \rangle
742 AOT_hence <math>\langle A!x \rangle \ \text{and} \ \langle A!y \rangle \ \text{and} \ \langle \forall F \square([F]x \equiv [F]y) \rangle
743     using "&E" by blast+
744 moreover AOT_have <math>\langle \forall F \square([F]y \equiv [F]x) \rangle
745     using calculation(3)
746     apply (safe intro!: CBF[THEN "\to E"] dest!: BF[THEN "\to E"])
747     using "RM:3" "cqt-basic:11" "intro-elim:3:b" by fast
748 ultimately AOT_show <math>\langle \exists G (\forall u \square([G]u \equiv [H]u) \ \& \ x[G]) \equiv
749     \exists G (\forall u \square([G]u \equiv [H]u) \ \& \ y[G]) \rangle
750     using 0 by (auto intro!: "\equiv I" "\to I")
751 qed
752 }
753 end
754 end

```

A.11. Possible Worlds

```

1  (*<*)
2  theory AOT_PossibleWorlds
3    imports AOT_PLM AOT_BasicLogicalObjects AOT_RestrictedVariables
4  begin
5  (*>*)
6
7  section<Possible Worlds>
8
9  AOT_define Situation :: < $\tau \Rightarrow \varphi$ > (<Situation'('_)>)
10   situations: <Situation(x)  $\equiv_{df}$   $A!x \ \& \ \forall F (x[F] \rightarrow \text{Propositional}([F]))$ > (456)
11
12 AOT_theorem "T-sit": <TruthValue(x)  $\rightarrow$  Situation(x)> (457)
13 proof(rule " $\rightarrow$ I")
14   AOT_assume <TruthValue(x)>
15   AOT_hence < $\exists p \ \text{TruthValueOf}(x,p)$ >
16     using "T-value"[THEN " $\equiv_{df}E$ "] by blast
17   then AOT_obtain p where <TruthValueOf(x,p)> using " $\exists E$ "[rotated] by blast
18   AOT_hence  $\vartheta$ : < $A!x \ \& \ \forall F (x[F] \equiv \exists q((q \equiv p) \ \& \ F = [\lambda y \ q]))$ >
19     using "tv-p"[THEN " $\equiv_{df}E$ "] by blast
20   AOT_show <Situation(x)>
21   proof(rule situations[THEN " $\equiv_{df}I$ "]; safe intro!: "&I" GEN " $\rightarrow$ I"  $\vartheta$ [THEN "&E"(1)])
22     fix F
23     AOT_assume <x[F]>
24     AOT_hence < $\exists q((q \equiv p) \ \& \ F = [\lambda y \ q])$ >
25       using  $\vartheta$ [THEN "&E"(2), THEN " $\forall E$ "(2)[where  $\beta=F$ ], THEN " $\equiv E$ "(1)] by argo
26     then AOT_obtain q where < $(q \equiv p) \ \& \ F = [\lambda y \ q]$ > using " $\exists E$ "[rotated] by blast
27     AOT_hence < $\exists p \ F = [\lambda y \ p]$ > using "&E"(2) " $\exists I$ "(2) by metis
28     AOT_thus <Propositional([F])>
29       by (metis " $\equiv_{df}I$ " "prop-prop1")
30   qed
31 qed
32
33 AOT_theorem "possit-sit:1": <Situation(x)  $\equiv \Box$ Situation(x)> (458.1)
34 proof(rule " $\equiv I$ "; rule " $\rightarrow$ I")
35   AOT_assume <Situation(x)>
36   AOT_hence 0: < $A!x \ \& \ \forall F (x[F] \rightarrow \text{Propositional}([F]))$ >
37     using situations[THEN " $\equiv_{df}E$ "] by blast
38   AOT_have 1: < $\Box(A!x \ \& \ \forall F (x[F] \rightarrow \text{Propositional}([F])))$ >
39   proof(rule "KBasic:3"[THEN " $\equiv E$ "(2)]; rule "&I")
40     AOT_show < $\Box A!x$ > using 0[THEN "&E"(1)] by (metis "oa-facts:2"[THEN " $\rightarrow E$ "])
41   next
42     AOT_have < $\forall F (x[F] \rightarrow \text{Propositional}([F])) \rightarrow \Box \forall F (x[F] \rightarrow \text{Propositional}([F]))$ >
43       by (AOT_subst <Propositional([F])> < $\exists p (F = [\lambda y \ p])$ > for: F :: << $\kappa$ >>)
44       (auto simp: "prop-prop1" " $\equiv Df$ " "enc-prop-nec:2")
45     AOT_thus < $\Box \forall F (x[F] \rightarrow \text{Propositional}([F]))$ >
46       using 0[THEN "&E"(2)] " $\rightarrow E$ " by blast
47   qed
48   AOT_show < $\Box$ Situation(x)>
49   by (AOT_subst <Situation(x)> < $A!x \ \& \ \forall F (x[F] \rightarrow \text{Propositional}([F]))$ >)
50     (auto simp: 1 " $\equiv Df$ " situations)
51 next
52   AOT_show <Situation(x)> if < $\Box$ Situation(x)>
53   using "qml:2"[axiom_inst, THEN " $\rightarrow E$ ", OF that].
54 qed
55
56 AOT_theorem "possit-sit:2": < $\Diamond$ Situation(x)  $\equiv$  Situation(x)> (458.2)
57   using "possit-sit:1"
58   by (metis "RE $\Diamond$ " "S5Basic:2" " $\equiv E$ "(1) " $\equiv E$ "(5) "Commutativity of  $\equiv$ ")
59
60 AOT_theorem "possit-sit:3": < $\Diamond$ Situation(x)  $\equiv \Box$ Situation(x)> (458.3)
61   using "possit-sit:1" "possit-sit:2" by (meson " $\equiv E$ "(5))

```

```

62
63 AOT_theorem "possit-sit:4": <ASituation(x) ≡ Situation(x)> (458.4)
64   by (meson "Act-Basic:5" "Act-Sub:2" "RA[2]" "≡E"(1) "≡E"(6) "possit-sit:2")
65
66 AOT_theorem "possit-sit:5": <Situation(op)> (458.5)
67 proof (safe intro!: situations[THEN "≡dfI"] "&I" GEN "→I" "prop-prop1"[THEN "≡dfI"])
68   AOT_have <∃F op[F]>
69     using "tv-id:2"[THEN "prop-enc"[THEN "≡dfE"], THEN "&E"(2)]
70     "existential:1" "prop-prop2:2" by blast
71   AOT_thus <A!op>
72     by (safe intro!: "encoders-are-abstract"[unvarify x, THEN "→E"]
73         "t=t-proper:2"[THEN "→E", OF "ext-p-tv:3"])
74 next
75   fix F
76   AOT_assume <op[F]>
77   AOT_hence <∃x(A!x & ∀F (x[F] ≡ ∃q ((q ≡ p) & F = [λy q])))>[F]
78     using "tv-id:1" "rule=E" by fast
79   AOT_hence <A∃q ((q ≡ p) & F = [λy q])>
80     using "≡E"(1) "desc-nec-encode:1" by fast
81   AOT_hence <∃q A((q ≡ p) & F = [λy q])>
82     by (metis "Act-Basic:10" "≡E"(1))
83   then AOT_obtain q where <A((q ≡ p) & F = [λy q])> using "∃E"[rotated] by blast
84   AOT_hence <AF = [λy q]> by (metis "Act-Basic:2" "con-dis-i-e:2:b" "intro-elim:3:a")
85   AOT_hence <F = [λy q]>
86     using "id-act:1"[unvarify β, THEN "≡E"(2)] by (metis "prop-prop2:2")
87   AOT_thus <∃p F = [λy p]>
88     using "∃I" by fast
89 qed
90
91 AOT_theorem "possit-sit:6": <Situation(T)> (458.6)
92 proof -
93   AOT_have true_def: <⊢□ T = ∃x (A!x & ∀F (x[F] ≡ ∃p(p & F = [λy p])))>
94     by (simp add: "A-descriptions" "rule-id-df:1[zero]" "the-true:1")
95   AOT_hence true_den: <⊢□ T↓>
96     using "t=t-proper:1" "vdash-properties:6" by blast
97   AOT_have <ATruthValue(T)>
98     using "actual-desc:2"[unvarify x, OF true_den, THEN "→E", OF true_def]
99     using "TV-lem2:1"[unvarify x, OF true_den, THEN "RA[2]",
100        THEN "act-cond"[THEN "→E"], THEN "→E"]
101     by blast
102   AOT_hence <ASituation(T)>
103     using "T-sit"[unvarify x, OF true_den, THEN "RA[2]",
104        THEN "act-cond"[THEN "→E"], THEN "→E"] by blast
105   AOT_thus <Situation(T)>
106     using "possit-sit:4"[unvarify x, OF true_den, THEN "≡E"(1)] by blast
107 qed
108
109 AOT_theorem "possit-sit:7": <Situation(⊥)> (458.7)
110 proof -
111   AOT_have true_def: <⊢□ ⊥ = ∃x (A!x & ∀F (x[F] ≡ ∃p(¬p & F = [λy p])))>
112     by (simp add: "A-descriptions" "rule-id-df:1[zero]" "the-true:2")
113   AOT_hence true_den: <⊢□ ⊥↓>
114     using "t=t-proper:1" "vdash-properties:6" by blast
115   AOT_have <ATruthValue(⊥)>
116     using "actual-desc:2"[unvarify x, OF true_den, THEN "→E", OF true_def]
117     using "TV-lem2:2"[unvarify x, OF true_den, THEN "RA[2]",
118        THEN "act-cond"[THEN "→E"], THEN "→E"]
119     by blast
120   AOT_hence <ASituation(⊥)>
121     using "T-sit"[unvarify x, OF true_den, THEN "RA[2]",
122        THEN "act-cond"[THEN "→E"], THEN "→E"] by blast
123   AOT_thus <Situation(⊥)>
124     using "possit-sit:4"[unvarify x, OF true_den, THEN "≡E"(1)] by blast

```

```

125 qed
126
127 AOT_register_rigid_restricted_type
128   Situation: <Situation( $\kappa$ )>
129 proof
130   AOT_modally_strict {
131     fix p
132     AOT_obtain x where <TruthValueOf(x,p)>
133       by (metis "instantiation" "p-has-!tv:1")
134     AOT_hence < $\exists p$  TruthValueOf(x,p)> by (rule " $\exists$ I")
135     AOT_hence <TruthValue(x)> by (metis " $\equiv_{df}$ I" "T-value")
136     AOT_hence <Situation(x)> using "T-sit"[THEN " $\rightarrow$ E"] by blast
137     AOT_thus < $\exists x$  Situation(x)> by (rule " $\exists$ I")
138   }
139 next
140   AOT_modally_strict {
141     AOT_show <Situation( $\kappa$ )  $\rightarrow \kappa \downarrow$ > for  $\kappa$ 
142     proof (rule " $\rightarrow$ I")
143       AOT_assume <Situation( $\kappa$ )>
144       AOT_hence <A! $\kappa$ > by (metis " $\equiv_{df}$ E" "&E"(1) situations)
145       AOT_thus < $\kappa \downarrow$ > by (metis "russell-axiom[exe,1]. $\psi$ -denotes_asm")
146     qed
147   }
148 next
149   AOT_modally_strict {
150     AOT_show < $\forall \alpha$ (Situation( $\alpha$ )  $\rightarrow \Box$ Situation( $\alpha$ ))>
151     using "possit-sit:1"[THEN "conventions:3"[THEN " $\equiv_{df}$ E"],
152           THEN "&E"(1)] GEN by fast
153   }
154 qed
155
156 AOT_register_variable_names
157   Situation: s
158
159 AOT_define TruthInSituation :: < $\tau \Rightarrow \varphi \Rightarrow \varphi$ > (" $\_ \models \_$ ") [100, 40] 100
160   "true-in-s": <s  $\models$  p  $\equiv_{df}$  s $\Sigma$ p> (459)
161
162 notepad
163 begin
164   (* Verify precedence. *)
165   fix x p q
166   have <<x  $\models$  p  $\rightarrow$  q> = <<(x  $\models$  p)  $\rightarrow$  q>>
167     by simp
168   have <<x  $\models$  p & q> = <<(x  $\models$  p) & q>>
169     by simp
170   have <<x  $\models$   $\neg$ p> = <x  $\models$  ( $\neg$ p)>>
171     by simp
172   have <<x  $\models$   $\Box$ p> = <x  $\models$  ( $\Box$ p)>>
173     by simp
174   have <<x  $\models$   $\mathcal{A}$ p> = <x  $\models$  ( $\mathcal{A}$ p)>>
175     by simp
176   have << $\Box$ x  $\models$  p> = < $\Box$ (x  $\models$  p)>>
177     by simp
178   have << $\neg$ x  $\models$  p> = < $\neg$ (x  $\models$  p)>>
179     by simp
180 end
181
182
183 AOT_theorem lem1: <Situation(x)  $\rightarrow$  (x  $\models$  p  $\equiv$  x[ $\lambda y$  p])> (460)
184 proof (rule " $\rightarrow$ I"; rule " $\equiv$ I"; rule " $\rightarrow$ I")
185   AOT_assume <Situation(x)>
186   AOT_assume <x  $\models$  p>
187   AOT_hence <x $\Sigma$ p>

```

```

188   using "true-in-s"[THEN "≡dfE"] "&E" by blast
189   AOT_thus <x[λy p]> using "prop-enc"[THEN "≡dfE"] "&E" by blast
190 next
191   AOT_assume 1: <Situation(x)>
192   AOT_assume <x[λy p]>
193   AOT_hence <xΣp>
194   using "prop-enc"[THEN "≡dfI", OF "&I", OF "cqt:2"(1)] by blast
195   AOT_thus <x ⊢ p>
196   using "true-in-s"[THEN "≡dfI"] 1 "&I" by blast
197 qed
198
199 AOT_theorem "lem2:1": <s ⊢ p ≡ □s ⊢ p> (462.1)
200 proof -
201   AOT_have sit: <Situation(s)>
202   by (simp add: Situation.ψ)
203   AOT_have <s ⊢ p ≡ s[λy p]>
204   using lem1[THEN "→E", OF sit] by blast
205   also AOT_have <... ≡ □s[λy p]>
206   by (rule "en-eq:2[1]"[unvarify F]) "cqt:2[lambda]"
207   also AOT_have <... ≡ □s ⊢ p>
208   using lem1[THEN RM, THEN "→E", OF "possit-sit:1"[THEN "≡E"(1), OF sit]]
209   by (metis "KBasic:6" "≡E"(2) "Commutativity of ≡" "→E")
210   finally show ?thesis.
211 qed
212
213 AOT_theorem "lem2:2": <◇s ⊢ p ≡ s ⊢ p> (462.2)
214 proof -
215   AOT_have <□(s ⊢ p → □s ⊢ p)>
216   using "possit-sit:1"[THEN "≡E"(1), OF Situation.ψ]
217   "lem2:1"[THEN "conventions:3"[THEN "≡dfE", THEN "&E"(1)]]
218   RM[OF "→I", THEN "→E"] by blast
219   thus ?thesis by (metis "B◇" "S5Basic:13" "T◇" "≡I" "≡E"(1) "→E")
220 qed
221
222 AOT_theorem "lem2:3": <◇s ⊢ p ≡ □s ⊢ p> (462.3)
223 using "lem2:1" "lem2:2" by (metis "≡E"(5))
224
225 AOT_theorem "lem2:4": <A(s ⊢ p) ≡ s ⊢ p> (462.4)
226 proof -
227   AOT_have <□(s ⊢ p → □s ⊢ p)>
228   using "possit-sit:1"[THEN "≡E"(1), OF Situation.ψ]
229   "lem2:1"[THEN "conventions:3"[THEN "≡dfE", THEN "&E"(1)]]
230   RM[OF "→I", THEN "→E"] by blast
231   thus ?thesis
232   using "sc-eq-fur:2"[THEN "→E"] by blast
233 qed
234
235 AOT_theorem "lem2:5": <¬s ⊢ p ≡ □¬s ⊢ p> (462.5)
236 by (metis "KBasic2:1" "contraposition:1[2]" "→I" "≡I" "≡E"(3) "≡E"(4) "lem2:2")
237
238 AOT_theorem "sit-identity": <s = s' ≡ ∀p(s ⊢ p ≡ s' ⊢ p)> (463)
239 proof(rule "≡I"; rule "→I")
240   AOT_assume <s = s'>
241   moreover AOT_have <∀p(s ⊢ p ≡ s' ⊢ p)>
242   by (simp add: "oth-class-taut:3:a" "universal-cor")
243   ultimately AOT_show <∀p(s ⊢ p ≡ s' ⊢ p)>
244   using "rule=E" by fast
245 next
246   AOT_assume a: <∀p (s ⊢ p ≡ s' ⊢ p)>
247   AOT_show <s = s'>
248   proof(safe intro!: "ab-obey:1"[THEN "→E", THEN "→E"] "&I" GEN "≡I" "→I")
249     AOT_show <A!s> using Situation.ψ "≡dfE" "&E"(1) situations by blast
250   next

```

```

251   AOT_show <A!s'> using Situation.ψ "≡dfE" "&E"(1) situations by blast
252 next
253   fix F
254   AOT_assume 0: <s[F]>
255   AOT_hence <∃p (F = [λy p])>
256     using Situation.ψ[THEN situations[THEN "≡dfE"], THEN "&E"(2),
257       THEN "∀E"(2)[where β=F], THEN "→E"]
258     "prop-prop1"[THEN "≡dfE"] by blast
259   then AOT_obtain p where F_def: <F = [λy p]>
260     using "∃E" by metis
261   AOT_hence <s[λy p]>
262     using 0 "rule=E" by blast
263   AOT_hence <s ⊨ p>
264     using lem1[THEN "→E", OF Situation.ψ, THEN "≡E"(2)] by blast
265   AOT_hence <s' ⊨ p>
266     using a[THEN "∀E"(2)[where β=p], THEN "≡E"(1)] by blast
267   AOT_hence <s'[λy p]>
268     using lem1[THEN "→E", OF Situation.ψ, THEN "≡E"(1)] by blast
269   AOT_thus <s'[F]>
270     using F_def[symmetric] "rule=E" by blast
271 next
272   fix F
273   AOT_assume 0: <s'[F]>
274   AOT_hence <∃p (F = [λy p])>
275     using Situation.ψ[THEN situations[THEN "≡dfE"], THEN "&E"(2),
276       THEN "∀E"(2)[where β=F], THEN "→E"]
277     "prop-prop1"[THEN "≡dfE"] by blast
278   then AOT_obtain p where F_def: <F = [λy p]>
279     using "∃E" by metis
280   AOT_hence <s'[λy p]>
281     using 0 "rule=E" by blast
282   AOT_hence <s' ⊨ p>
283     using lem1[THEN "→E", OF Situation.ψ, THEN "≡E"(2)] by blast
284   AOT_hence <s ⊨ p>
285     using a[THEN "∀E"(2)[where β=p], THEN "≡E"(2)] by blast
286   AOT_hence <s[λy p]>
287     using lem1[THEN "→E", OF Situation.ψ, THEN "≡E"(1)] by blast
288   AOT_thus <s[F]>
289     using F_def[symmetric] "rule=E" by blast
290 qed
291 qed
292
293 AOT_define PartOfSituation :: <τ ⇒ τ ⇒ φ> (infixl <≤> 80)
294   "sit-part-whole": <s ≤ s' ≡df ∀p (s ⊨ p → s' ⊨ p)> (464)
295
296 AOT_theorem "part:1": <s ≤ s> (465.1)
297   by (rule "sit-part-whole"[THEN "≡dfI"])
298     (safe intro!: "&I" Situation.ψ GEN "→I")
299
300 AOT_theorem "part:2": <s ≤ s' & s ≠ s' → ¬(s' ≤ s)> (465.2)
301 proof(rule "→I"; frule "&E"(1); drule "&E"(2); rule "raa-cor:2")
302   AOT_assume 0: <s ≤ s'>
303   AOT_hence a: <s ⊨ p → s' ⊨ p> for p
304     using "∀E"(2) "sit-part-whole"[THEN "≡dfE"] "&E" by blast
305   AOT_assume <s' ≤ s>
306   AOT_hence b: <s' ⊨ p → s ⊨ p> for p
307     using "∀E"(2) "sit-part-whole"[THEN "≡dfE"] "&E" by blast
308   AOT_have <∀p (s ⊨ p ≡ s' ⊨ p)>
309     using a b by (simp add: "≡I" "universal-cor")
310   AOT_hence 1: <s = s'>
311     using "sit-identity"[THEN "≡E"(2)] by metis
312   AOT_assume <s ≠ s'>
313   AOT_hence <¬(s = s')>

```

```

314   by (metis "≡dfE" "--infix")
315   AOT_thus <s = s' & ¬(s = s')>
316   using 1 "&I" by blast
317 qed
318
319 AOT_theorem "part:3": <s ≤ s' & s' ≤ s" → s ≤ s"> (465.3)
320 proof(rule "→I"; frule "&E"(1); drule "&E"(2);
321   safe intro!: "&I" GEN "→I" "sit-part-whole"[THEN "≡dfI"] Situation.ψ)
322   fix p
323   AOT_assume <s ⊨ p>
324   moreover AOT_assume <s ≤ s'>
325   ultimately AOT_have <s' ⊨ p>
326     using "sit-part-whole"[THEN "≡dfE", THEN "&E"(2),
327       THEN "∀E"(2)[where β=p], THEN "→E"] by blast
328   moreover AOT_assume <s' ≤ s">
329   ultimately AOT_show <s" ⊨ p>
330     using "sit-part-whole"[THEN "≡dfE", THEN "&E"(2),
331       THEN "∀E"(2)[where β=p], THEN "→E"] by blast
332 qed
333
334 AOT_theorem "sit-identity2:1": <s = s' ≡ s ≤ s' & s' ≤ s> (466.1)
335 proof (safe intro!: "≡I" "&I" "→I")
336   AOT_show <s ≤ s'> if <s = s'>
337   using "rule=E" "part:1" that by blast
338 next
339   AOT_show <s' ≤ s> if <s = s'>
340   using "rule=E" "part:1" that[symmetric] by blast
341 next
342   AOT_assume <s ≤ s' & s' ≤ s>
343   AOT_thus <s = s'> using "part:2"[THEN "→E", OF "&I"]
344   by (metis "≡dfI" "&E"(1) "&E"(2) "--infix" "raa-cor:3")
345 qed
346
347 AOT_theorem "sit-identity2:2": <s = s' ≡ ∀s" (s" ≤ s ≡ s" ≤ s')> (466.2)
348 proof(safe intro!: "≡I" "→I" Situation.GEN "sit-identity"[THEN "≡E"(2)]
349   GEN[where 'a=0])
350   AOT_show <s" ≤ s'> if <s" ≤ s> and <s = s'> for s"
351   using "rule=E" that by blast
352 next
353   AOT_show <s" ≤ s> if <s" ≤ s'> and <s = s'> for s"
354   using "rule=E" id_sym that by blast
355 next
356   AOT_show <s' ⊨ p> if <s ⊨ p> and <∀s" (s" ≤ s ≡ s" ≤ s')> for p
357   using "sit-part-whole"[THEN "≡dfE", THEN "&E"(2),
358     OF that(2)[THEN "Situation.∀E", THEN "≡E"(1), OF "part:1"],
359     THEN "∀E"(2), THEN "→E", OF that(1)].
360 next
361   AOT_show <s ⊨ p> if <s' ⊨ p> and <∀s" (s" ≤ s ≡ s" ≤ s')> for p
362   using "sit-part-whole"[THEN "≡dfE", THEN "&E"(2),
363     OF that(2)[THEN "Situation.∀E", THEN "≡E"(2), OF "part:1"],
364     THEN "∀E"(2), THEN "→E", OF that(1)].
365 qed
366
367 AOT_define Persistent :: <φ ⇒ φ> (<Persistent'(_)>)
368   persistent: <Persistent(p) ≡df ∀s (s ⊨ p → ∀s' (s ≤ s' → s' ⊨ p))> (467)
369
370 AOT_theorem "pers-prop": <∀p Persistent(p)> (468)
371   by (safe intro!: GEN[where 'a=0] Situation.GEN persistent[THEN "≡dfI"] "→I")
372   (simp add: "sit-part-whole"[THEN "≡dfE", THEN "&E"(2), THEN "∀E"(2), THEN "→E"])
373
374 AOT_define NullSituation :: <τ ⇒ φ> (<NullSituation'(_)>)
375   "df-null-trivial:1": <NullSituation(s) ≡df ¬∃p s ⊨ p> (469.1)
376

```

```

377 AOT_define TrivialSituation :: < $\tau \Rightarrow \varphi$ > (<TrivialSituation'('_)>)
378   "df-null-trivial:2": <TrivialSituation(s)  $\equiv_{df} \forall p \ s \models p$ > (469.2)
379
380 AOT_theorem "thm-null-trivial:1": < $\exists!x \ \text{NullSituation}(x)$ > (470.1)
381 proof (AOT_subst <NullSituation(x)> <A!x &  $\forall F \ (x[F] \equiv F \neq F)$ > for: x)
382   AOT_modally_strict {
383     AOT_show <NullSituation(x)  $\equiv A!x \ \& \ \forall F \ (x[F] \equiv F \neq F)$ > for x
384     proof (safe intro!: "≡I" "→I" "df-null-trivial:1"[THEN "≡dfI"])
385       dest!: "df-null-trivial:1"[THEN "≡dfE"]
386     AOT_assume 0: <Situation(x) &  $\neg \exists p \ x \models p$ >
387     AOT_have 1: <A!x>
388       using 0[THEN "&E"(1), THEN situations[THEN "≡dfE"], THEN "&E"(1)].
389     AOT_have 2: <x[F]  $\rightarrow \exists p \ F = [\lambda y \ p]$ > for F
390       using 0[THEN "&E"(1), THEN situations[THEN "≡dfE"],
391         THEN "&E"(2), THEN "∀E"(2)]
392       by (metis "≡dfE" "→I" "prop-prop1" "→E")
393     AOT_show <A!x &  $\forall F \ (x[F] \equiv F \neq F)$ >
394     proof (safe intro!: "&I" 1 GEN "≡I" "→I")
395       fix F
396       AOT_assume <x[F]>
397       moreover AOT_obtain p where <F =  $[\lambda y \ p]$ >
398         using calculation 2[THEN "→E"] "∃E"[rotated] by blast
399       ultimately AOT_have <x[ $\lambda y \ p$ ]>
400         by (metis "rule=E")
401       AOT_hence <x  $\models p$ >
402         using lem1[THEN "→E", OF 0[THEN "&E"(1)], THEN "≡E"(2)] by blast
403       AOT_hence < $\exists p \ (x \models p)$ >
404         by (rule "∃I")
405       AOT_thus <F  $\neq F$ >
406         using 0[THEN "&E"(2)] "raa-cor:1" "&I" by blast
407     next
408     fix F :: << $\kappa$ > AOT_var>
409     AOT_assume <F  $\neq F$ >
410     AOT_hence < $\neg(F = F)$ > by (metis "≡dfE" "=-infix")
411     moreover AOT_have <F = F>
412       by (simp add: "id-eq:1")
413     ultimately AOT_show <x[F]> using "&I" "raa-cor:1" by blast
414   qed
415 next
416 AOT_assume 0: <A!x &  $\forall F \ (x[F] \equiv F \neq F)$ >
417 AOT_hence <x[F]  $\equiv F \neq F$ > for F
418   using "∀E" "&E" by blast
419 AOT_hence 1: < $\neg x[F]$ > for F
420   using "≡dfE" "id-eq:1" "=-infix" "reductio-aa:1" "≡E"(1) by blast
421 AOT_show <Situation(x) &  $\neg \exists p \ x \models p$ >
422 proof (safe intro!: "&I" situations[THEN "≡dfI"] 0[THEN "&E"(1)] GEN "→I")
423   AOT_show <Propositional([F])> if <x[F]> for F
424   using that 1 "&I" "raa-cor:1" by fast
425 next
426 AOT_show < $\neg \exists p \ x \models p$ >
427 proof (rule "raa-cor:2")
428   AOT_assume < $\exists p \ x \models p$ >
429   then AOT_obtain p where <x  $\models p$ > using "∃E"[rotated] by blast
430   AOT_hence <x[ $\lambda y \ p$ ]>
431     using "≡dfE" "&E"(1) "≡E"(1) lem1 "modus-tollens:1"
432     "raa-cor:3" "true-in-s" by fast
433   moreover AOT_have < $\neg x[\lambda y \ p]$ >
434     by (rule 1[unvarify F]) "cqt:2[lambda]"
435   ultimately AOT_show <p &  $\neg p$ > for p using "&I" "raa-cor:1" by blast
436   qed
437   qed
438   qed
439 }

```



```

440 next
441   AOT_show <∃!x ([A!]x & ∀F (x[F] ≡ F ≠ F))>
442   by (simp add: "A-objects!")
443 qed
444
445
446 AOT_theorem "thm-null-trivial:2": <∃!x TrivialSituation(x)> (470.2)
447 proof (AOT_subst <TrivialSituation(x)> <A!x & ∀F (x[F] ≡ ∃p F = [λy p])> for: x)
448   AOT_modally_strict {
449     AOT_show <TrivialSituation(x) ≡ A!x & ∀F (x[F] ≡ ∃p F = [λy p])> for x
450     proof (safe intro!: "≡I" "→I" "df-null-trivial:2"[THEN "≡dfI"]
451           dest!: "df-null-trivial:2"[THEN "≡dfE"])
452       AOT_assume 0: <Situation(x) & ∀p x ⊨ p>
453       AOT_have 1: <A!x>
454         using 0[THEN "&E"(1), THEN situations[THEN "≡dfE"], THEN "&E"(1)].
455       AOT_have 2: <x[F] → ∃p F = [λy p]> for F
456         using 0[THEN "&E"(1), THEN situations[THEN "≡dfE"],
457               THEN "&E"(2), THEN "∀E"(2)]
458         by (metis "≡dfE" "deduction-theorem" "prop-prop1" "→E")
459       AOT_show <A!x & ∀F (x[F] ≡ ∃p F = [λy p])>
460       proof (safe intro!: "&I" 1 GEN "≡I" "→I" 2)
461         fix F
462         AOT_assume <∃p F = [λy p]>
463         then AOT_obtain p where <F = [λy p]>
464           using "∃E"[rotated] by blast
465         moreover AOT_have <x ⊨ p>
466           using 0[THEN "&E"(2)] "∀E" by blast
467         ultimately AOT_show <x[F]>
468           by (metis 0 "rule=E" "&E"(1) id_sym "≡E"(2) lem1
469               "Commutativity of ≡" "→E")
470       qed
471     qed
472   next
473     AOT_assume 0: <A!x & ∀F (x[F] ≡ ∃p F = [λy p])>
474     AOT_hence 1: <x[F] ≡ ∃p F = [λy p]> for F
475     using "∀E" "&E" by blast
476     AOT_have 2: <Situation(x)>
477     proof (safe intro!: "&I" situations[THEN "≡dfI"] 0[THEN "&E"(1)] GEN "→I")
478       AOT_show <Propositional([F])> if <x[F]> for F
479       using 1[THEN "≡E"(1), OF that]
480       by (metis "≡dfI" "prop-prop1")
481     qed
482     AOT_show <Situation(x) & ∀p (x ⊨ p)>
483     proof (safe intro!: "&I" 2 0[THEN "&E"(1)] GEN "→I")
484       AOT_have <x[λy p] ≡ ∃q [λy p] = [λy q]> for p
485       by (rule 1[unvarify F, where τ="«[λy p]»"]) "cqt:2[lambda]"
486       moreover AOT_have <∃q [λy p] = [λy q]> for p
487       by (rule "∃I"(2)[where β=p])
488       (simp add: "rule=I:1" "prop-prop2:2")
489       ultimately AOT_have <x[λy p]> for p by (metis "≡E"(2))
490       AOT_thus <x ⊨ p> for p
491       by (metis "2" "≡E"(2) lem1 "→E")
492     qed
493   qed
494 next
495   AOT_show <∃!x ([A!]x & ∀F (x[F] ≡ ∃p F = [λy p]))>
496   by (simp add: "A-objects!")
497 qed
498
499 AOT_theorem "thm-null-trivial:3": <∃x NullSituation(x)↓> (470.3)
500 by (meson "A-Exists:2" "RA[2]" "≡E"(2) "thm-null-trivial:1")
501
502 AOT_theorem "thm-null-trivial:4": <∃x TrivialSituation(x)↓> (470.4)

```

```

503   using "A-Exists:2" "RA[2]" "≡E"(2) "thm-null-trivial:2" by blast
504
505 AOT_define TheNullSituation :: <κs> (<s0>)
506   "df-the-null-sit:1": <s0 =df λx NullSituation(x)> (471.1)
507
508 AOT_define TheTrivialSituation :: <κs> (<sv>)
509   "df-the-null-sit:2": <sv =df λx TrivialSituation(x)> (471.2)
510
511 AOT_theorem "null-triv-sc:1": <NullSituation(x) → □NullSituation(x)> (472.1)
512 proof(safe intro!: "→I" dest!: "df-null-trivial:1"[THEN "≡dfE"];
513   frule "&E"(1); drule "&E"(2))
514   AOT_assume 1: <¬∃p (x ⊨ p)>
515   AOT_assume 0: <Situation(x)>
516   AOT_hence <□Situation(x)> by (metis "≡E"(1) "possit-sit:1")
517   moreover AOT_have <□¬∃p (x ⊨ p)>
518   proof(rule "raa-cor:1")
519     AOT_assume <¬□¬∃p (x ⊨ p)>
520     AOT_hence <◇∃p (x ⊨ p)>
521     by (metis "≡dfI" "conventions:5")
522     AOT_hence <∃p ◇(x ⊨ p)> by (metis "BF◇" "→E")
523     then AOT_obtain p where <◇(x ⊨ p)> using "∃E"[rotated] by blast
524     AOT_hence <x ⊨ p>
525     by (metis "≡E"(1) "lem2:2"[unconstrain s, THEN "→E", OF 0])
526     AOT_hence <∃p x ⊨ p> using "∃I" by fast
527     AOT_thus <∃p x ⊨ p & ¬∃p x ⊨ p> using 1 "&I" by blast
528   qed
529   ultimately AOT_have 2: <□(Situation(x) & ¬∃p x ⊨ p)>
530   by (metis "KBasic:3" "&I" "≡E"(2))
531   AOT_show <□NullSituation(x)>
532   by (AOT_subst <NullSituation(x)> <Situation(x) & ¬∃p x ⊨ p>)
533   (auto simp: "df-null-trivial:1" "≡Df" 2)
534 qed
535
536
537 AOT_theorem "null-triv-sc:2": <TrivialSituation(x) → □TrivialSituation(x)> (472.2)
538 proof(safe intro!: "→I" dest!: "df-null-trivial:2"[THEN "≡dfE"];
539   frule "&E"(1); drule "&E"(2))
540   AOT_assume 0: <Situation(x)>
541   AOT_hence 1: <□Situation(x)> by (metis "≡E"(1) "possit-sit:1")
542   AOT_assume <∀p x ⊨ p>
543   AOT_hence <x ⊨ p> for p
544     using "∀E" by blast
545   AOT_hence <□x ⊨ p> for p
546     using 0 "≡E"(1) "lem2:1"[unconstrain s, THEN "→E"] by blast
547   AOT_hence <∀p □x ⊨ p>
548     by (rule GEN)
549   AOT_hence <□∀p x ⊨ p>
550     by (rule BF[THEN "→E"])
551   AOT_hence 2: <□(Situation(x) & ∀p x ⊨ p)>
552     using 1 by (metis "KBasic:3" "&I" "≡E"(2))
553   AOT_show <□TrivialSituation(x)>
554     by (AOT_subst <TrivialSituation(x)> <Situation(x) & ∀p x ⊨ p>)
555     (auto simp: "df-null-trivial:2" "≡Df" 2)
556 qed
557
558 AOT_theorem "null-triv-sc:3": <NullSituation(s0)> (472.3)
559   by (safe intro!: "df-the-null-sit:1"[THEN "≡dfI"(2)] "thm-null-trivial:3"
560     "rule=I:1"[OF "thm-null-trivial:3"]
561     "!box-desc:2"[THEN "→E", THEN "→E", rotated, OF "thm-null-trivial:1",
562       OF "∀I", OF "null-triv-sc:1", THEN "∀E"(1), THEN "→E"])
563
564 AOT_theorem "null-triv-sc:4": <TrivialSituation(sv)> (472.4)
565   by (safe intro!: "df-the-null-sit:2"[THEN "≡dfI"(2)] "thm-null-trivial:4"

```

```

566     "rule=I:1"[OF "thm-null-trivial:4"]
567     "!box-desc:2"[THEN "→E", THEN "→E", rotated, OF "thm-null-trivial:2",
568         OF "∀I", OF "null-triv-sc:2", THEN "∀E"(1), THEN "→E"]
569
570 AOT_theorem "null-triv-facts:1": <NullSituation(x) ≡ Null(x)> (473.1)
571 proof (safe intro!: "≡I" "→I" "df-null-uni:1"[THEN "≡dfI"]
572     "df-null-trivial:1"[THEN "≡dfI"]
573     dest!: "df-null-uni:1"[THEN "≡dfE"] "df-null-trivial:1"[THEN "≡dfE"]])
574 AOT_assume 0: <Situation(x) & ¬∃p x ⊨ p>
575 AOT_have 1: <x[F] → ∃p F = [λy p]> for F
576     using 0[THEN "&E"(1), THEN situations[THEN "≡dfE"], THEN "&E"(2), THEN "∀E"(2)]
577     by (metis "≡dfE" "deduction-theorem" "prop-prop1" "→E")
578 AOT_show <A!x & ¬∃F x[F]>
579 proof (safe intro!: "&I" 0[THEN "&E"(1), THEN situations[THEN "≡dfE"],
580     THEN "&E"(1)];
581     rule "raa-cor:2")
582 AOT_assume <∃F x[F]>
583 then AOT_obtain F where F_prop: <x[F]>
584     using "∃E"[rotated] by blast
585 AOT_hence <∃p F = [λy p]>
586     using 1[THEN "→E"] by blast
587 then AOT_obtain p where <F = [λy p]>
588     using "∃E"[rotated] by blast
589 AOT_hence <x[λy p]>
590     by (metis "rule=E" F_prop)
591 AOT_hence <x ⊨ p>
592     using lem1[THEN "→E", OF 0[THEN "&E"(1)], THEN "≡E"(2)] by blast
593 AOT_hence <∃p x ⊨ p>
594     by (rule "∃I")
595 AOT_thus <∃p x ⊨ p & ¬∃p x ⊨ p>
596     using 0[THEN "&E"(2)] "&I" by blast
597 qed
598 next
599 AOT_assume 0: <A!x & ¬∃F x[F]>
600 AOT_have <Situation(x)>
601     apply (rule situations[THEN "≡dfI", OF "&I", OF 0[THEN "&E"(1)]]; rule GEN)
602     using 0[THEN "&E"(2)] by (metis "→I" "existential:2[const_var]" "raa-cor:3")
603 moreover AOT_have <¬∃p x ⊨ p>
604 proof (rule "raa-cor:2")
605     AOT_assume <∃p x ⊨ p>
606     then AOT_obtain p where <x ⊨ p> by (metis "instantiation")
607     AOT_hence <x[λy p]> by (metis "≡dfE" "&E"(2) "prop-enc" "true-in-s")
608     AOT_hence <∃F x[F]> by (rule "∃I") "cqt:2[lambda]"
609     AOT_thus <∃F x[F] & ¬∃F x[F]> using 0[THEN "&E"(2)] "&I" by blast
610 qed
611 ultimately AOT_show <Situation(x) & ¬∃p x ⊨ p> using "&I" by blast
612 qed
613
614 AOT_theorem "null-triv-facts:2": <sθ = aθ> (473.2)
615 apply (rule "=dfI"(2)[OF "df-the-null-sit:1"])
616 apply (fact "thm-null-trivial:3")
617 apply (rule "=dfI"(2)[OF "df-null-uni-terms:1"])
618 apply (fact "null-uni-uniq:3")
619 apply (rule "equiv-desc-eq:3"[THEN "→E"])
620 apply (rule "&I")
621 apply (fact "thm-null-trivial:3")
622 by (rule RN; rule GEN; rule "null-triv-facts:1")
623
624 AOT_theorem "null-triv-facts:3": <sv ≠ av> (473.3)
625 proof(rule "=-infix"[THEN "≡dfI"])
626 AOT_have <Universal(av)>
627     by (simp add: "null-uni-facts:4")
628 AOT_hence 0: <av[A!]>

```

```

629     using "df-null-uni:2"[THEN "≡dfE" "&E" "∀E"(1)
630     by (metis "cqt:5:a" "vdash-properties:10" "vdash-properties:1[2]")
631 moreover AOT_have 1: <¬sv[A!]>
632 proof(rule "raa-cor:2")
633   AOT_have <Situation(sv)>
634   using "≡dfE" "&E"(1) "df-null-trivial:2" "null-triv-sc:4" by blast
635   AOT_hence <∀F (sv[F] → Propositional([F]))>
636   by (metis "≡dfE" "&E"(2) situations)
637   moreover AOT_assume <sv[A!]>
638   ultimately AOT_have <Propositional(A!)>
639   using "∀E"(1)[rotated, OF "oa-exist:2"] "→E" by blast
640   AOT_thus <Propositional(A!) & ¬Propositional(A!)>
641   using "prop-in-f:4:d" "&I" by blast
642 qed
643 AOT_show <¬(sv = av)>
644 proof (rule "raa-cor:2")
645   AOT_assume <sv = av>
646   AOT_hence <sv[A!]> using 0 "rule=E" id_sym by fast
647   AOT_thus <sv[A!] & ¬sv[A!]> using 1 "&I" by blast
648 qed
649 qed
650
651 definition ConditionOnPropositionalProperties :: <(<κ> ⇒ o) ⇒ bool> where
652   "cond-prop": <ConditionOnPropositionalProperties ≡ λ φ . ∀ v .
653     [v ⊨ ∀F (φ{F} → Propositional([F]))]>
654
655 syntax ConditionOnPropositionalProperties :: <id_position ⇒ AOT_prop>
656   ("CONDITION'_ON'_PROPOSITIONAL'_PROPERTIES'('_')")
657
658 AOT_theorem "cond-prop[E]":
659   assumes <CONDITION_ON_PROPOSITIONAL_PROPERTIES(φ)>
660   shows <∀F (φ{F} → Propositional([F]))>
661   using assms[unfolded "cond-prop"] by auto
662
663 AOT_theorem "cond-prop[I]":
664   assumes <⊢□ ∀F (φ{F} → Propositional([F]))>
665   shows <CONDITION_ON_PROPOSITIONAL_PROPERTIES(φ)>
666   using assms "cond-prop" by metis
667
668 AOT_theorem "pre-comp-sit":
669   assumes <CONDITION_ON_PROPOSITIONAL_PROPERTIES(φ)>
670   shows <(Situation(x) & ∀F (x[F] ≡ φ{F})) ≡ (A!x & ∀F (x[F] ≡ φ{F}))>
671 proof(rule "≡I"; rule "→I")
672   AOT_assume <Situation(x) & ∀F (x[F] ≡ φ{F})>
673   AOT_thus <A!x & ∀F (x[F] ≡ φ{F})>
674   using "&E" situations[THEN "≡dfE" "&I" by blast
675 next
676   AOT_assume 0: <A!x & ∀F (x[F] ≡ φ{F})>
677   AOT_show <Situation(x) & ∀F (x[F] ≡ φ{F})>
678   proof (safe intro!: situations[THEN "≡dfI" "&I"])
679     AOT_show <A!x> using 0[THEN "&E"(1)].
680   next
681     AOT_show <∀F (x[F] → Propositional([F]))>
682     proof(rule GEN; rule "→I")
683       fix F
684       AOT_assume <x[F]>
685       AOT_hence <φ{F}>
686       using 0[THEN "&E"(2)] "∀E" "≡E" by blast
687       AOT_thus <Propositional([F])>
688       using "cond-prop[E]" [OF assms] "∀E" "→E" by blast
689     qed
690   next
691     AOT_show <∀F (x[F] ≡ φ{F})> using 0 "&E" by blast

```

```

692   qed
693 qed
694
695 AOT_theorem "comp-sit:1": (476.1)
696   assumes <CONDITION_ON_PROPOSITIONAL_PROPERTIES( $\varphi$ )>
697   shows < $\exists s \forall F (s[F] \equiv \varphi\{F\})$ >
698   by (AOT_subst <Situation(x) &  $\forall F (x[F] \equiv \varphi\{F\})$ > <A!x &  $\forall F (x[F] \equiv \varphi\{F\})$ > for: x)
699     (auto simp: "pre-comp-sit"[OF assms] "A-objects"[where  $\varphi=\varphi$ , axiom_inst])
700
701 AOT_theorem "comp-sit:2": (476.2)
702   assumes <CONDITION_ON_PROPOSITIONAL_PROPERTIES( $\varphi$ )>
703   shows < $\exists! s \forall F (s[F] \equiv \varphi\{F\})$ >
704   by (AOT_subst <Situation(x) &  $\forall F (x[F] \equiv \varphi\{F\})$ > <A!x &  $\forall F (x[F] \equiv \varphi\{F\})$ > for: x)
705     (auto simp: assms "pre-comp-sit" "pre-comp-sit"[OF assms] "A-objects!")
706
707 AOT_theorem "can-sit-desc:1": (477.1)
708   assumes <CONDITION_ON_PROPOSITIONAL_PROPERTIES( $\varphi$ )>
709   shows < $\iota s (\forall F (s[F] \equiv \varphi\{F\})) \downarrow$ >
710   using "comp-sit:2"[OF assms] "A-Exists:2" "RA[2]" " $\equiv E$ "(2) by blast
711
712 AOT_theorem "can-sit-desc:2": (477.2)
713   assumes <CONDITION_ON_PROPOSITIONAL_PROPERTIES( $\varphi$ )>
714   shows < $\iota s (\forall F (s[F] \equiv \varphi\{F\})) = \iota x (A!x \ \& \ \forall F (x[F] \equiv \varphi\{F\}))$ >
715   by (auto intro!: "equiv-desc-eq:2"[THEN " $\rightarrow E$ ", OF "&I",
716     OF "can-sit-desc:1"[OF assms]]
717     "RA[2]" GEN "pre-comp-sit"[OF assms])
718
719 AOT_theorem "strict-sit": (478)
720   assumes <RIGID_CONDITION( $\varphi$ )>
721   and <CONDITION_ON_PROPOSITIONAL_PROPERTIES( $\varphi$ )>
722   shows < $y = \iota s (\forall F (s[F] \equiv \varphi\{F\})) \rightarrow \forall F (y[F] \equiv \varphi\{F\})$ >
723   using "rule=E"[rotated, OF "can-sit-desc:2"[OF assms(2), symmetric]]
724     "box-phi-a:2"[OF assms(1)] " $\rightarrow E$ " " $\rightarrow I$ " "&E" by fast
725
726 (* TODO: exercise (479) sit-lit *)
727
728 AOT_define actual :: < $\tau \Rightarrow \varphi$ > (<Actual'(_')>) (481)
729   <Actual(s)  $\equiv_{df} \forall p (s \models p \rightarrow p)$ >
730
731 AOT_theorem "act-and-not-pos": < $\exists s (Actual(s) \ \& \ \Diamond \neg Actual(s))$ > (482)
732 proof -
733   AOT_obtain q1 where q1_prop: <q1 &  $\Diamond \neg q_1$ >
734   by (metis " $\equiv_{df} E$ " "instantiation" "cont-tf:1" "cont-tf-thm:1")
735   AOT_have < $\exists s (\forall F (s[F] \equiv F = [\lambda y \ q_1]))$ >
736   proof (safe intro!: "comp-sit:1" "cond-prop[I]" GEN " $\rightarrow I$ ")
737     AOT_modally_strict {
738       AOT_show <Propositional([F])> if <F =  $[\lambda y \ q_1]$ > for F
739       using " $\equiv_{df} I$ " "existential:2[const_var]" "prop-prop1" that by fastforce
740     }
741   qed
742   then AOT_obtain s1 where s1_prop: < $\forall F (s_1[F] \equiv F = [\lambda y \ q_1])$ >
743   using "Situation. $\exists E$ "[rotated] by meson
744   AOT_have <Actual(s1)>
745   proof (safe intro!: actual[THEN " $\equiv_{df} I$ " "&I" GEN " $\rightarrow I$ " s1_prop Situation. $\psi$ ]
746     fix p
747     AOT_assume <s1  $\models p$ >
748     AOT_hence <s1  $[\lambda y \ p]$ >
749     by (metis " $\equiv_{df} E$ " "&E"(2) "prop-enc" "true-in-s")
750     AOT_hence < $[\lambda y \ p] = [\lambda y \ q_1]$ >
751     by (rule s1_prop[THEN " $\forall E$ "(1), THEN " $\equiv E$ "(1), rotated]) "cqt:2[lambda]"
752     AOT_hence <p = q1> by (metis " $\equiv E$ "(2) "p-identity-thm2:3")
753     AOT_thus <p> using q1_prop[THEN "&E"(1)] "rule=E" id_sym by fast
754   qed

```

```

755 moreover AOT_have < $\diamond \neg \text{Actual}(s_1)$ >
756 proof(rule "raa-cor:1"; drule "KBasic:12"[THEN "≡E"(2)])
757   AOT_assume < $\Box \text{Actual}(s_1)$ >
758   AOT_hence < $\Box(\text{Situation}(s_1) \ \& \ \forall p (s_1 \models p \rightarrow p))$ >
759     using actual[THEN "≡df", THEN "conventions:3"[THEN "≡dfE"],
760       THEN "&E"(1), THEN RM, THEN "→E"] by blast
761   AOT_hence < $\Box \forall p (s_1 \models p \rightarrow p)$ >
762     by (metis "RM:1" "Conjunction Simplification"(2) "→E")
763   AOT_hence < $\forall p \Box (s_1 \models p \rightarrow p)$ >
764     by (metis "CBF" "vdash-properties:10")
765   AOT_hence < $\Box (s_1 \models q_1 \rightarrow q_1)$ >
766     using "∀E" by blast
767   AOT_hence < $\Box s_1 \models q_1 \rightarrow \Box q_1$ >
768     by (metis "→E" "qml:1" "vdash-properties:1[2]")
769   moreover AOT_have < $s_1 \models q_1$ >
770     using s_prop[THEN "∀E"(1), THEN "≡E"(2),
771       THEN lem1[THEN "→E", OF Situation.ψ, THEN "≡E"(2)]]
772       "rule=I:1" "prop-prop2:2" by blast
773   ultimately AOT_have < $\Box q_1$ >
774     using "≡dfE" "&E"(1) "≡E"(1) "lem2:1" "true-in-s" "→E" by fast
775   AOT_thus < $\diamond \neg q_1 \ \& \ \neg \diamond \neg q_1$ >
776     using "KBasic:12"[THEN "≡E"(1)] q1_prop[THEN "&E"(2)] "&I" by blast
777 qed
778 ultimately AOT_have < $(\text{Actual}(s_1) \ \& \ \diamond \neg \text{Actual}(s_1))$ >
779   using s_prop "&I" by blast
780 thus ?thesis
781   by (rule "Situation.∃I")
782 qed
783
784 AOT_theorem "actual-s:1": < $\exists s \text{Actual}(s)$ > (484.1)
785 proof -
786   AOT_obtain s where < $(\text{Actual}(s) \ \& \ \diamond \neg \text{Actual}(s))$ >
787     using "act-and-not-pos" "Situation.∃E"[rotated] by meson
788   AOT_hence < $\text{Actual}(s)$ > using "&E" "&I" by metis
789   thus ?thesis by (rule "Situation.∃I")
790 qed
791
792 AOT_theorem "actual-s:2": < $\exists s \neg \text{Actual}(s)$ > (484.2)
793 proof(rule "∃I"(1)[where τ=< $s_v$ >]; (rule "&I")?)
794   AOT_show < $\text{Situation}(s_v)$ >
795     using "≡dfE" "&E"(1) "df-null-trivial:2" "null-triv-sc:4" by blast
796 next
797   AOT_show < $\neg \text{Actual}(s_v)$ >
798   proof(rule "raa-cor:2")
799     AOT_assume 0: < $\text{Actual}(s_v)$ >
800     AOT_obtain p1 where notp1: < $\neg p_1$ >
801       by (metis "∃E" "∃I"(1) "log-prop-prop:2" "non-contradiction")
802     AOT_have < $s_v \models p_1$ >
803       using "null-triv-sc:4"[THEN "≡dfE"[OF "df-null-trivial:2"], THEN "&E"(2)]
804         "∀E" by blast
805     AOT_hence < $p_1$ >
806       using 0[THEN actual[THEN "≡dfE"], THEN "&E"(2), THEN "∀E"(2), THEN "→E"]
807       by blast
808     AOT_thus < $p \ \& \ \neg p$ > for p using notp1 by (metis "raa-cor:3")
809   qed
810 next
811   AOT_show < $s_v \downarrow$ >
812     using "df-the-null-sit:2" "rule-id-df:2:b[zero]" "thm-null-trivial:4" by blast
813 qed
814
815 AOT_theorem "actual-s:3": < $\exists p \forall s (\text{Actual}(s) \rightarrow \neg s \models p)$ > (484.3)
816 proof -
817   AOT_obtain p1 where notp1: < $\neg p_1$ >

```

```

818   by (metis "∃E" "∃I"(1) "log-prop-prop:2" "non-contradiction")
819 AOT_have <∀s (Actual(s) → ¬(s ⊨ p1))>
820 proof (rule Situation.GEN; rule "→I"; rule "raa-cor:2")
821   fix s
822   AOT_assume <Actual(s)>
823   moreover AOT_assume <s ⊨ p1>
824   ultimately AOT_have <p1>
825     using actual[THEN "≡dfE", THEN "&E"(2), THEN "∀E"(2), THEN "→E"] by blast
826   AOT_thus <p1 & ¬p1>
827     using notp1 "&I" by simp
828   qed
829   thus ?thesis by (rule "∃I")
830 qed
831
832 AOT_theorem comp: (485)
833   <∃s (s' ≤ s & s" ≤ s & ∀s'' (s' ≤ s'' & s" ≤ s'' → s ≤ s''))>
834 proof -
835   have cond_prop: <ConditionOnPropositionalProperties (λ Π . «s'[II] ∨ s"[II]»)>
836   proof (safe intro!: "cond-prop[I]" GEN "oth-class-taut:8:c"[THEN "→E", THEN "→E"];
837     rule "→I")
838     AOT_modally_strict {
839       fix F
840       AOT_have <Situation(s')>
841         by (simp add: Situation.restricted_var_condition)
842       AOT_hence <s'[F] → Propositional([F])>
843         using "situations"[THEN "≡dfE", THEN "&E"(2), THEN "∀E"(2)] by blast
844       moreover AOT_assume <s'[F]>
845       ultimately AOT_show <Propositional([F])>
846         using "→E" by blast
847     }
848   next
849     AOT_modally_strict {
850       fix F
851       AOT_have <Situation(s'')>
852         by (simp add: Situation.restricted_var_condition)
853       AOT_hence <s"[F] → Propositional([F])>
854         using "situations"[THEN "≡dfE", THEN "&E"(2), THEN "∀E"(2)] by blast
855       moreover AOT_assume <s"[F]>
856       ultimately AOT_show <Propositional([F])>
857         using "→E" by blast
858     }
859   qed
860   AOT_obtain s3 where ϑ: <∀F (s3[F] ≡ s'[F] ∨ s"[F])>
861     using "comp-sit:1"[OF cond_prop] "Situation.∃E"[rotated] by meson
862   AOT_have <s' ≤ s3 & s" ≤ s3 & ∀s'' (s' ≤ s'' & s" ≤ s'' → s3 ≤ s'')>
863   proof (safe intro!: "&I" "≡dfI"[OF "true-in-s"] "≡dfI"[OF "prop-enc"]
864     "Situation.GEN" "GEN"[where 'a=o] "→I"
865     "sit-part-whole"[THEN "≡dfI"]
866     Situation.ψ "cqt:2[const_var]"[axiom_inst])
867     fix p
868     AOT_assume <s' ⊨ p>
869     AOT_hence <s'[λx p]>
870       by (metis "&E"(2) "prop-enc" "≡dfE" "true-in-s")
871     AOT_thus <s3[λx p]>
872       using ϑ[THEN "∀E"(1),OF "prop-prop2:2", THEN "≡E"(2), OF "∀I"(1)] by blast
873   next
874     fix p
875     AOT_assume <s" ⊨ p>
876     AOT_hence <s"[λx p]>
877       by (metis "&E"(2) "prop-enc" "≡dfE" "true-in-s")
878     AOT_thus <s3[λx p]>
879       using ϑ[THEN "∀E"(1),OF "prop-prop2:2", THEN "≡E"(2), OF "∀I"(2)] by blast
880   next

```



```

881   fix s p
882   AOT_assume 0: <s'  $\sqsubseteq$  s & s"  $\sqsubseteq$  s>
883   AOT_assume <s3  $\models$  p>
884   AOT_hence <s3[ $\lambda$ x p]>
885     by (metis "&E"(2) "prop-enc" " $\equiv_{df}E$ " "true-in-s")
886   AOT_hence <s' [ $\lambda$ x p]  $\vee$  s" [ $\lambda$ x p]>
887     using  $\vartheta$ [THEN " $\forall E$ "(1),OF "prop-prop2:2", THEN " $\equiv E$ "(1)] by blast
888   moreover {
889     AOT_assume <s' [ $\lambda$ x p]>
890     AOT_hence <s'  $\models$  p>
891     by (safe intro!: "prop-enc"[THEN " $\equiv_{df}I$ "] "true-in-s"[THEN " $\equiv_{df}I$ "] "&I"
892         Situation. $\psi$  "cqt:2[const_var]"[axiom_inst])
893     moreover AOT_have <s'  $\models$  p  $\rightarrow$  s  $\models$  p>
894       using "sit-part-whole"[THEN " $\equiv_{df}E$ ", THEN "&E"(2)] 0[THEN "&E"(1)]
895         " $\forall E$ "(2) by blast
896     ultimately AOT_have <s  $\models$  p>
897       using " $\rightarrow E$ " by blast
898     AOT_hence <s [ $\lambda$ x p]>
899       using "true-in-s"[THEN " $\equiv_{df}E$ "] "prop-enc"[THEN " $\equiv_{df}E$ "] "&E" by blast
900   }
901   moreover {
902     AOT_assume <s" [ $\lambda$ x p]>
903     AOT_hence <s"  $\models$  p>
904     by (safe intro!: "prop-enc"[THEN " $\equiv_{df}I$ "] "true-in-s"[THEN " $\equiv_{df}I$ "] "&I"
905         Situation. $\psi$  "cqt:2[const_var]"[axiom_inst])
906     moreover AOT_have <s"  $\models$  p  $\rightarrow$  s  $\models$  p>
907       using "sit-part-whole"[THEN " $\equiv_{df}E$ ", THEN "&E"(2)] 0[THEN "&E"(2)]
908         " $\forall E$ "(2) by blast
909     ultimately AOT_have <s  $\models$  p>
910       using " $\rightarrow E$ " by blast
911     AOT_hence <s [ $\lambda$ x p]>
912       using "true-in-s"[THEN " $\equiv_{df}E$ "] "prop-enc"[THEN " $\equiv_{df}E$ "] "&E" by blast
913   }
914   ultimately AOT_show <s [ $\lambda$ x p]>
915     by (metis " $\forall E$ "(1) " $\rightarrow I$ ")
916   qed
917   thus ?thesis
918     using "Situation. $\exists I$ " by fast
919   qed
920
921   AOT_theorem "act-sit:1": <Actual(s)  $\rightarrow$  (s  $\models$  p  $\rightarrow$  [ $\lambda$ y p]s)> (486.1)
922   proof (safe intro!: " $\rightarrow I$ ")
923     AOT_assume <Actual(s)>
924     AOT_hence p if <s  $\models$  p>
925       using actual[THEN " $\equiv_{df}E$ ", THEN "&E"(2), THEN " $\forall E$ "(2), THEN " $\rightarrow E$ "] that by blast
926     moreover AOT_assume <s  $\models$  p>
927     ultimately AOT_have p by blast
928     AOT_thus <[ $\lambda$ y p]s>
929     by (safe intro!: " $\beta \leftarrow C$ "(1) "cqt:2")
930   qed
931
932   AOT_theorem "act-sit:2": (486.2)
933     <(Actual(s') & Actual(s"))  $\rightarrow$   $\exists$ x (Actual(x) & s'  $\sqsubseteq$  x & s"  $\sqsubseteq$  x)>
934   proof(rule " $\rightarrow I$ "; frule "&E"(1); drule "&E"(2))
935     AOT_assume act_s': <Actual(s')>
936     AOT_assume act_s": <Actual(s")>
937     have "cond-prop": <ConditionOnPropositionalProperties (474)
938         ( $\lambda$   $\Pi$  .  $\llbracket \exists$ p ( $\Pi = [\lambda$ y p] & (s'  $\models$  p  $\vee$  s"  $\models$  p)) $\rrbracket$ >>
939   proof (safe intro!: "cond-prop[I]" " $\forall I$ " " $\rightarrow I$ " "prop-prop1"[THEN " $\equiv_{df}I$ "])
940     AOT_modally_strict {
941       fix  $\beta$ 
942       AOT_assume < $\exists$ p ( $\beta = [\lambda$ y p] & (s'  $\models$  p  $\vee$  s"  $\models$  p))>
943       then AOT_obtain p where < $\beta = [\lambda$ y p]> using " $\exists E$ "[rotated] "&E" by blast

```



```

944   AOT_thus <∃p β = [λy p] > by (rule "∃I")
945 }
946 qed
947 have rigid: <rigid_condition (λ Π . «∃p (Π = [λy p] & (s' ⊨ p ∨ s" ⊨ p))»)>
948 proof(safe intro!: "strict-can:1[I]" "→I" GEN)
949   AOT_modally_strict {
950     fix F
951     AOT_assume <∃p (F = [λy p] & (s' ⊨ p ∨ s" ⊨ p))>
952     then AOT_obtain p1 where p1_prop: <F = [λy p1] & (s' ⊨ p1 ∨ s" ⊨ p1)>
953       using "∃E"[rotated] by blast
954     AOT_hence <□(F = [λy p1])>
955       using "&E"(1) "id-nec:2" "vdash-properties:10" by blast
956     moreover AOT_have <□(s' ⊨ p1 ∨ s" ⊨ p1)>
957     proof(rule "∨E"; (rule "→I"; rule "KBasic:15"[THEN "→E"])?)
958       AOT_show <s' ⊨ p1 ∨ s" ⊨ p1> using p1_prop "&E" by blast
959     next
960       AOT_show <□s' ⊨ p1 ∨ □s" ⊨ p1> if <s' ⊨ p1>
961         apply (rule "∨I"(1))
962         using "≡defE" "&E"(1) "≡E"(1) "lem2:1" that "true-in-s" by blast
963     next
964       AOT_show <□s' ⊨ p1 ∨ □s" ⊨ p1> if <s" ⊨ p1>
965         apply (rule "∨I"(2))
966         using "≡defE" "&E"(1) "≡E"(1) "lem2:1" that "true-in-s" by blast
967     qed
968     ultimately AOT_have <□(F = [λy p1] & (s' ⊨ p1 ∨ s" ⊨ p1))>
969       by (metis "KBasic:3" "&I" "≡E"(2))
970     AOT_hence <∃p □(F = [λy p] & (s' ⊨ p ∨ s" ⊨ p))> by (rule "∃I")
971     AOT_thus <□∃p (F = [λy p] & (s' ⊨ p ∨ s" ⊨ p))>
972       using Buridan[THEN "→E"] by fast
973   }
974 qed
975
976 AOT_have desc_den: <ιs(∀F (s[F] ≡ ∃p (F = [λy p] & (s' ⊨ p ∨ s" ⊨ p))))>↓
977   by (rule "can-sit-desc:1"[OF "cond-prop"])
978 AOT_obtain x0
979   where x0_prop1: <x0 = ιs(∀F (s[F] ≡ ∃p (F = [λy p] & (s' ⊨ p ∨ s" ⊨ p))))>
980   by (metis (no_types, lifting) "∃E" "rule=I:1" desc_den "∃I"(1) id_sym)
981 AOT_hence x0_sit: <Situation(x0)>
982   using "actual-desc:3"[THEN "→E"] "Act-Basic:2" "&E"(1) "≡E"(1)
983     "possit-sit:4" by blast
984
985 AOT_have 1: <∀F (x0[F] ≡ ∃p (F = [λy p] & (s' ⊨ p ∨ s" ⊨ p)))>
986   using "strict-sit"[OF rigid, OF "cond-prop", THEN "→E", OF x0_prop1].
987 AOT_have 2: <(x0 ⊨ p) ≡ (s' ⊨ p ∨ s" ⊨ p)> for p
988 proof (rule "≡I"; rule "→I")
989   AOT_assume <x0 ⊨ p>
990   AOT_hence <x0[λy p]> using lem1[THEN "→E", OF x0_sit, THEN "≡E"(1)] by blast
991   then AOT_obtain q where <[λy p] = [λy q] & (s' ⊨ q ∨ s" ⊨ q)>
992     using 1[THEN "∨E"(1)[where τ="«[λy p]»"], OF "prop-prop2:2", THEN "≡E"(1)]
993     "∃E"[rotated] by blast
994   AOT_thus <s' ⊨ p ∨ s" ⊨ p>
995     by (metis "rule=E" "&E"(1) "&E"(2) "∨I"(1) "∨I"(2)
996         "∨E"(1) "deduction-theorem" id_sym "≡E"(2) "p-identity-thm2:3")
997 next
998   AOT_assume <s' ⊨ p ∨ s" ⊨ p>
999   AOT_hence <[λy p] = [λy p] & (s' ⊨ p ∨ s" ⊨ p)>
1000     by (metis "rule=I:1" "&I" "prop-prop2:2")
1001   AOT_hence <∃q ([λy p] = [λy q] & (s' ⊨ q ∨ s" ⊨ q))>
1002     by (rule "∃I")
1003   AOT_hence <x0[λy p]>
1004     using 1[THEN "∨E"(1), OF "prop-prop2:2", THEN "≡E"(2)] by blast
1005   AOT_thus <x0 ⊨ p>
1006     by (metis "≡defI" "&I" "ex:1:a" "prop-enc" "rule-ui:2[const_var]"

```

```

1007         x0_sit "true-in-s")
1008 qed
1009
1010 AOT_have <Actual(x0) & s' ≤ x0 & s" ≤ x0>
1011 proof(safe intro!: "→I" "&I" "∃I"(1) actual[THEN "≡dfI"] x0_sit GEN
1012         "sit-part-whole"[THEN "≡dfI"])
1013   fix p
1014   AOT_assume <x0 ⊨ p>
1015   AOT_hence <s' ⊨ p ∨ s" ⊨ p>
1016     using 2 "≡E"(1) by metis
1017   AOT_thus <p>
1018     using act_s' act_s"
1019       actual[THEN "≡dfE", THEN "&E"(2), THEN "∀E"(2), THEN "→E"]
1020     by (metis "∀E"(3) "reductio-aa:1")
1021 next
1022   AOT_show <x0 ⊨ p> if <s' ⊨ p> for p
1023     using 2[THEN "≡E"(2), OF "∀I"(1), OF that].
1024 next
1025   AOT_show <x0 ⊨ p> if <s" ⊨ p> for p
1026     using 2[THEN "≡E"(2), OF "∀I"(2), OF that].
1027 next
1028   AOT_show <Situation(s')>
1029     using act_s'[THEN actual[THEN "≡dfE"]] "&E" by blast
1030 next
1031   AOT_show <Situation(s)">
1032     using act_s"[THEN actual[THEN "≡dfE"]] "&E" by blast
1033 qed
1034 AOT_thus <∃x (Actual(x) & s' ≤ x & s" ≤ x)>
1035   by (rule "∃I")
1036 qed
1037
1038 AOT_define Consistent :: <τ ⇒ φ> (<Consistent'(_)>)
1039   cons: <Consistent(s) ≡df ¬∃p (s ⊨ p & s ⊨ ¬p)> (487)
1040
1041 AOT_theorem "sit-cons": <Actual(s) → Consistent(s)> (489)
1042 proof(safe intro!: "→I" cons[THEN "≡dfI"] "&I" Situation.ψ
1043         dest!: actual[THEN "≡dfE"]; frule "&E"(1); drule "&E"(2))
1044   AOT_assume 0: <∀p (s ⊨ p → p)>
1045   AOT_show <¬∃p (s ⊨ p & s ⊨ ¬p)>
1046   proof (rule "raa-cor:2")
1047     AOT_assume <∃p (s ⊨ p & s ⊨ ¬p)>
1048     then AOT_obtain p where <s ⊨ p & s ⊨ ¬p>
1049       using "∃E"[rotated] by blast
1050     AOT_hence <p & ¬p>
1051       using 0[THEN "∀E"(1)[where τ=«¬p»], THEN "→E"], OF "log-prop-prop:2"]
1052       0[THEN "∀E"(2)[where β=p], THEN "→E"] "&E" "&I" by blast
1053     AOT_thus <p & ¬p> for p by (metis "raa-cor:1")
1054   qed
1055 qed
1056
1057 AOT_theorem "cons-rigid:1": <¬Consistent(s) ≡ □¬Consistent(s)> (490.1)
1058 proof (rule "≡I"; rule "→I")
1059   AOT_assume <¬Consistent(s)>
1060   AOT_hence <∃p (s ⊨ p & s ⊨ ¬p)>
1061     using cons[THEN "≡dfI", OF "&I", OF Situation.ψ]
1062     by (metis "raa-cor:3")
1063   then AOT_obtain p where p_prop: <s ⊨ p & s ⊨ ¬p>
1064     using "∃E"[rotated] by blast
1065   AOT_hence <□s ⊨ p>
1066     using "&E"(1) "≡E"(1) "lem2:1" by blast
1067   moreover AOT_have <□s ⊨ ¬p>
1068     using p_prop "T◇" "&E" "≡E"(1)
1069     "modus-tollens:1" "raa-cor:3" "lem2:3"[unvarify p]

```

```

1070     "log-prop-prop:2" by metis
1071 ultimately AOT_have <□(s ⊨ p & s ⊨ ¬p)>
1072   by (metis "KBasic:3" "&I" "≡E"(2))
1073 AOT_hence <∃p □(s ⊨ p & s ⊨ ¬p)>
1074   by (rule "∃I")
1075 AOT_hence <□∃p(s ⊨ p & s ⊨ ¬p)>
1076   by (metis Buridan "vdash-properties:10")
1077 AOT_thus <□¬Consistent(s)>
1078   apply (rule "qml:1"[axiom_inst, THEN "→E", THEN "→E", rotated])
1079   apply (rule RN)
1080   using "≡dfE" "&E"(2) cons "deduction-theorem" "raa-cor:3" by blast
1081 next
1082   AOT_assume <□¬Consistent(s)>
1083   AOT_thus <¬Consistent(s)> using "qml:2"[axiom_inst, THEN "→E"] by auto
1084 qed
1085
1086 AOT_theorem "cons-rigid:2": <◇Consistent(x) ≡ Consistent(x)> (490.2)
1087 proof(rule "≡I"; rule "→I")
1088   AOT_assume 0: <◇Consistent(x)>
1089   AOT_have <◇(Situation(x) & ¬∃p (x ⊨ p & x ⊨ ¬p))>
1090     apply (AOT_subst <Situation(x) & ¬∃p (x ⊨ p & x ⊨ ¬p)> <Consistent(x)>)
1091     using cons "≡E"(2) "Commutativity of ≡" "≡Df" apply blast
1092     by (simp add: 0)
1093   AOT_hence <◇Situation(x)> and 1: <◇¬∃p (x ⊨ p & x ⊨ ¬p)>
1094     using "RM◇" "Conjunction Simplification"(1) "Conjunction Simplification"(2)
1095     "modus-tollens:1" "raa-cor:3" by blast+
1096   AOT_hence 2: <Situation(x)> by (metis "≡E"(1) "possit-sit:2")
1097   AOT_have 3: <¬□∃p (x ⊨ p & x ⊨ ¬p)>
1098     using 2 using 1 "KBasic:11" "≡E"(2) by blast
1099   AOT_show <Consistent(x)>
1100   proof (rule "raa-cor:1")
1101     AOT_assume <¬Consistent(x)>
1102     AOT_hence <∃p (x ⊨ p & x ⊨ ¬p)>
1103       using 0 "≡dfE" "conventions:5" 2 "cons-rigid:1"[unconstrain s, THEN "→E"]
1104       "modus-tollens:1" "raa-cor:3" "≡E"(4) by meson
1105     then AOT_obtain p where <x ⊨ p> and 4: <x ⊨ ¬p>
1106       using "∃E"[rotated] "&E" by blast
1107     AOT_hence <□x ⊨ p>
1108       by (metis "2" "≡E"(1) "lem2:1"[unconstrain s, THEN "→E"])
1109     moreover AOT_have <□x ⊨ ¬p>
1110       using 4 "lem2:1"[unconstrain s, unvarify p, THEN "→E"]
1111       by (metis 2 "≡E"(1) "log-prop-prop:2")
1112     ultimately AOT_have <□(x ⊨ p & x ⊨ ¬p)>
1113       by (metis "KBasic:3" "&I" "≡E"(3) "raa-cor:3")
1114     AOT_hence <∃p □(x ⊨ p & x ⊨ ¬p)>
1115       by (metis "existential:1" "log-prop-prop:2")
1116     AOT_hence <□∃p (x ⊨ p & x ⊨ ¬p)>
1117       by (metis Buridan "vdash-properties:10")
1118     AOT_thus <p & ¬p> for p
1119       using 3 "&I" by (metis "raa-cor:3")
1120   qed
1121 next
1122   AOT_show <◇Consistent(x)> if <Consistent(x)>
1123     using "T◇" that "vdash-properties:10" by blast
1124 qed
1125
1126 AOT_define possible :: <τ ⇒ φ> (<Possible'('_)>)
1127   pos: <Possible(s) ≡df ◇Actual(s)> (491)
1128
1129 AOT_theorem "sit-pos:1": <Actual(s) → Possible(s)> (492.1)
1130   apply(rule "→I"; rule pos[THEN "≡dfI"]; rule "&I")
1131   apply (meson "≡dfE" actual "&E"(1))
1132   using "T◇" "vdash-properties:10" by blast

```

```

1133
1134 AOT_theorem "sit-pos:2": <∃p ((s ⊨ p) & ¬◇p) → ¬Possible(s)> (492.2)
1135 proof(rule "→I")
1136   AOT_assume <∃p ((s ⊨ p) & ¬◇p)>
1137   then AOT_obtain p where a: <(s ⊨ p) & ¬◇p>
1138     using "∃E"[rotated] by blast
1139   AOT_hence <□(s ⊨ p)>
1140     using "&E" by (metis "T◇" "≡E"(1) "lem2:3" "vdash-properties:10")
1141   moreover AOT_have <□¬p>
1142     using a[THEN "&E"(2)] by (metis "KBasic2:1" "≡E"(2))
1143   ultimately AOT_have <□(s ⊨ p & ¬p)>
1144     by (metis "KBasic:3" "&I" "≡E"(3) "raa-cor:3")
1145   AOT_hence <∃p □(s ⊨ p & ¬p)>
1146     by (rule "∃I")
1147   AOT_hence 1: <□∃q (s ⊨ q & ¬q)>
1148     by (metis Buridan "vdash-properties:10")
1149   AOT_have <□¬∀q (s ⊨ q → q)>
1150     apply (AOT_subst <s ⊨ q → q> <¬(s ⊨ q & ¬q)> for: q)
1151     apply (simp add: "oth-class-taut:1:a")
1152     apply (AOT_subst <¬∀q ¬(s ⊨ q & ¬q)> <∃q (s ⊨ q & ¬q)>)
1153     by (auto simp: "conventions:4" "df-rules-formulas[3]" "df-rules-formulas[4]" "≡I" 1)
1154   AOT_hence 0: <¬◇∀q (s ⊨ q → q)>
1155     by (metis "≡dfE" "conventions:5" "raa-cor:3")
1156   AOT_show <¬Possible(s)>
1157     apply (AOT_subst <Possible(s)> <Situation(s) & ◇Actual(s)>)
1158     apply (simp add: pos "≡Df")
1159     apply (AOT_subst <Actual(s)> <Situation(s) & ∀q (s ⊨ q → q)>)
1160     using actual "≡Df" apply presburger
1161     by (metis "0" "KBasic2:3" "&E"(2) "raa-cor:3" "vdash-properties:10")
1162 qed
1163
1164 AOT_theorem "pos-cons-sit:1": <Possible(s) → Consistent(s)> (493.1)
1165   by (auto simp: "sit-cons"[THEN "RM◇", THEN "→E",
1166     THEN "cons-rigid:2"[THEN "≡E"(1)]]
1167     intro!: "→I" dest!: pos[THEN "≡dfE"] "&E"(2))
1168
1169 AOT_theorem "pos-cons-sit:2": <∃s (Consistent(s) & ¬Possible(s))> (493.2)
1170 proof -
1171   AOT_obtain q1 where <q1 & ◇¬q1>
1172     using "≡dfE" "instantiation" "cont-tf:1" "cont-tf-thm:1" by blast
1173   have "cond-prop": <ConditionOnPropositionalProperties
1174     (λ Π . «Π = [λy q1 & ¬q1]]> (474)
1175     by (auto intro!: "cond-prop[I]" GEN "→I" "prop-prop1"[THEN "≡dfI"]
1176       "∃I"(1)[where τ=<<q1 & ¬q1>>, rotated, OF "log-prop-prop:2"]])
1177   have rigid: <rigid-condition (λ Π . «Π = [λy q1 & ¬q1]]>
1178     by (auto intro!: "strict-can:1[I]" GEN "→I" simp: "id-nec:2"[THEN "→E"])
1179
1180   AOT_obtain x where x_prop: <x = ιs (∀F (s[F] ≡ F = [λy q1 & ¬q1]))>
1181     using "ex:1:b"[THEN "∀E"(1), OF "can-sit-desc:1", OF "cond-prop"]
1182     "∃E"[rotated] by blast
1183   AOT_hence 0: <¬A(Situation(x) & ∀F (x[F] ≡ F = [λy q1 & ¬q1]))>
1184     using "→E" "actual-desc:2" by blast
1185   AOT_hence <¬A(Situation(x))> by (metis "Act-Basic:2" "&E"(1) "≡E"(1))
1186   AOT_hence s_sit: <Situation(x)> by (metis "≡E"(1) "possit-sit:4")
1187   AOT_have s_enc_prop: <∀F (x[F] ≡ F = [λy q1 & ¬q1])>
1188     using "strict-sit"[OF rigid, OF "cond-prop", THEN "→E", OF x_prop].
1189   AOT_hence <x[λy q1 & ¬q1]>
1190     using "∀E"(1)[rotated, OF "prop-prop2:2"]
1191     "rule=I:1"[OF "prop-prop2:2"] "≡E" by blast
1192   AOT_hence <x ⊨ (q1 & ¬q1)>
1193     using lem1[THEN "→E", OF s_sit, unvarify p, THEN "≡E"(2), OF "log-prop-prop:2"]
1194     by blast
1195   AOT_hence <□(x ⊨ (q1 & ¬q1))>

```

```

1196   using "lem2:1"[unconstrain s, THEN "→E", OF s_sit, unvarify p,
1197           OF "log-prop-prop:2", THEN "≡E"(1)] by blast
1198 moreover AOT_have <□(x ⊨ (q1 & ¬q1) → ¬Actual(x))>
1199 proof(rule RN; rule "→I"; rule "raa-cor:2")
1200   AOT_modally_strict {
1201     AOT_assume <Actual(x)>
1202     AOT_hence <∀p (x ⊨ p → p)>
1203       using actual[THEN "≡dfE", THEN "&E"(2)] by blast
1204     moreover AOT_assume <x ⊨ (q1 & ¬q1)>
1205     ultimately AOT_show <q1 & ¬q1>
1206       using "∀E"(1)[rotated, OF "log-prop-prop:2"] "→E" by metis
1207   }
1208 qed
1209 ultimately AOT_have nec_not_actual_s: <□¬Actual(x)>
1210   using "qml:1"[axiom_inst, THEN "→E", THEN "→E"] by blast
1211 AOT_have 1: <¬∃p (x ⊨ p & x ⊨ ¬p)>
1212 proof (rule "raa-cor:2")
1213   AOT_assume <∃p (x ⊨ p & x ⊨ ¬p)>
1214   then AOT_obtain p where <x ⊨ p & x ⊨ ¬p>
1215     using "∃E"[rotated] by blast
1216   AOT_hence <x[λy p] & x[λy ¬p]>
1217     using lem1[unvarify p, THEN "→E", OF "log-prop-prop:2",
1218             OF s_sit, THEN "≡E"(1)] "&I" "&E" by metis
1219   AOT_hence <[λy p] = [λy q1 & ¬q1] & [λy ¬p] = [λy q1 & ¬q1] >
1220     by (auto intro!: "prop-prop2:2" s_enc_prop[THEN "∀E"(1), THEN "≡E"(1)]
1221         elim: "&E")
1222   AOT_hence i: <[λy p] = [λy ¬p] > by (metis "rule=E" id_sym)
1223   {
1224     AOT_assume 0: <p>
1225     AOT_have <[λy p]x> for x
1226       by (auto intro!: "β←C"(1) "cqt:2" 0)
1227     AOT_hence <[λy ¬p]x> for x using i "rule=E" by fast
1228     AOT_hence <¬p>
1229       using "β→C"(1) by auto
1230   }
1231 moreover {
1232   AOT_assume 0: <¬p>
1233   AOT_have <[λy ¬p]x> for x
1234     by (auto intro!: "β←C"(1) "cqt:2" 0)
1235   AOT_hence <[λy p]x> for x using i[symmetric] "rule=E" by fast
1236   AOT_hence <p>
1237     using "β→C"(1) by auto
1238 }
1239 ultimately AOT_show <p & ¬p> for p by (metis "raa-cor:1" "raa-cor:3")
1240 qed
1241 AOT_have 2: <¬Possible(x)>
1242 proof(rule "raa-cor:2")
1243   AOT_assume <Possible(x)>
1244   AOT_hence <◇Actual(x)>
1245     by (metis "≡dfE" "&E"(2) pos)
1246   moreover AOT_have <¬◇Actual(x)> using nec_not_actual_s
1247     using "≡dfE" "conventions:5" "reductio-aa:2" by blast
1248   ultimately AOT_show <◇Actual(x) & ¬◇Actual(x)> by (rule "&I")
1249 qed
1250 show ?thesis
1251   by(rule "∃I"(2)[where β=x]; safe intro!: "&I" 2 s_sit cons[THEN "≡dfI"] 1)
1252 qed
1253
1254 AOT_theorem "sit-classical:1": <∀p (s ⊨ p ≡ p) → ∀q(s ⊨ ¬q ≡ ¬s ⊨ q)> (494.1)
1255 proof(rule "→I"; rule GEN)
1256   fix q
1257   AOT_assume <∀p (s ⊨ p ≡ p)>
1258   AOT_hence <s ⊨ q ≡ q & s ⊨ ¬q ≡ ¬q>

```

```

1259     using "VE"(1)[rotated, OF "log-prop-prop:2"] by blast+
1260 AOT_thus <s ⊢ ¬q ≡ ¬s ⊢ q>
1261     by (metis "deduction-theorem" "≡I" "≡E"(1) "≡E"(2) "≡E"(4))
1262 qed
1263
1264 AOT_theorem "sit-classical:2": (494.2)
1265   <∀p (s ⊢ p ≡ p) → ∀q∀r((s ⊢ (q → r)) ≡ (s ⊢ q → s ⊢ r))>
1266 proof(rule "→I"; rule GEN; rule GEN)
1267   fix q r
1268   AOT_assume <∀p (s ⊢ p ≡ p)>
1269   AOT_hence ϑ: <s ⊢ q ≡ q> and ξ: <s ⊢ r ≡ r> and ζ: <(s ⊢ (q → r)) ≡ (q → r)>
1270     using "VE"(1)[rotated, OF "log-prop-prop:2"] by blast+
1271   AOT_show <(s ⊢ (q → r)) ≡ (s ⊢ q → s ⊢ r)>
1272   proof (safe intro!: "≡I" "→I")
1273     AOT_assume <s ⊢ (q → r)>
1274     moreover AOT_assume <s ⊢ q>
1275     ultimately AOT_show <s ⊢ r>
1276     using ϑ ξ ζ by (metis "≡E"(1) "≡E"(2) "vdash-properties:10")
1277   next
1278     AOT_assume <s ⊢ q → s ⊢ r>
1279     AOT_thus <s ⊢ (q → r)>
1280     using ϑ ξ ζ by (metis "deduction-theorem" "≡E"(1) "≡E"(2) "→E")
1281   qed
1282 qed
1283
1284 AOT_theorem "sit-classical:3": (494.3)
1285   <∀p (s ⊢ p ≡ p) → ((s ⊢ ∀α φ{α}) ≡ ∀α s ⊢ φ{α})>
1286 proof (rule "→I")
1287   AOT_assume <∀p (s ⊢ p ≡ p)>
1288   AOT_hence ϑ: <s ⊢ φ{α} ≡ φ{α}> and ξ: <s ⊢ ∀α φ{α} ≡ ∀α φ{α}> for α
1289     using "VE"(1)[rotated, OF "log-prop-prop:2"] by blast+
1290   AOT_show <s ⊢ ∀α φ{α} ≡ ∀α s ⊢ φ{α}>
1291   proof (safe intro!: "≡I" "→I" GEN)
1292     fix α
1293     AOT_assume <s ⊢ ∀α φ{α}>
1294     AOT_hence <φ{α}> using ξ "VE"(2) "≡E"(1) by blast
1295     AOT_thus <s ⊢ φ{α}> using ϑ "≡E"(2) by blast
1296   next
1297     AOT_assume <∀α s ⊢ φ{α}>
1298     AOT_hence <s ⊢ φ{α}> for α using "VE"(2) by blast
1299     AOT_hence <φ{α}> for α using ϑ "≡E"(1) by blast
1300     AOT_hence <∀α φ{α}> by (rule GEN)
1301     AOT_thus <s ⊢ ∀α φ{α}> using ξ "≡E"(2) by blast
1302   qed
1303 qed
1304
1305 AOT_theorem "sit-classical:4": <∀p (s ⊢ p ≡ p) → ∀q (s ⊢ □q → □s ⊢ q)> (494.4)
1306 proof(rule "→I"; rule GEN; rule "→I")
1307   fix q
1308   AOT_assume <∀p (s ⊢ p ≡ p)>
1309   AOT_hence ϑ: <s ⊢ q ≡ q> and ξ: <s ⊢ □q ≡ □q>
1310     using "VE"(1)[rotated, OF "log-prop-prop:2"] by blast+
1311   AOT_assume <s ⊢ □q>
1312   AOT_hence <□q> using ξ "≡E"(1) by blast
1313   AOT_hence <q> using "qml:2"[axiom_inst, THEN "→E"] by blast
1314   AOT_hence <s ⊢ q> using ϑ "≡E"(2) by blast
1315   AOT_thus <□s ⊢ q> using "≡defE" "&E"(1) "≡E"(1) "lem2:1" "true-in-s" by blast
1316 qed
1317
1318 AOT_theorem "sit-classical:5": (494.5)
1319   <∀p (s ⊢ p ≡ p) → ∃q(□(s ⊢ q) & ¬(s ⊢ □q))>
1320 proof (rule "→I")
1321   AOT_obtain r where A: <r> and <◇¬r>

```

```

1322   by (metis "&E"(1) "&E"(2) "≡dfE" "instantiation" "cont-tf:1" "cont-tf-thm:1")
1323 AOT_hence B: <¬□r>
1324   using "KBasic:11" "≡E"(2) by blast
1325 moreover AOT_assume asm: <∀ p (s ⊨ p ≡ p)>
1326 AOT_hence <s ⊨ r>
1327   using "∀E"(2) A "≡E"(2) by blast
1328 AOT_hence 1: <□s ⊨ r>
1329   using "≡dfE" "&E"(1) "≡E"(1) "lem2:1" "true-in-s" by blast
1330 AOT_have <s ⊨ ¬□r>
1331   using asm[THEN "∀E"(1)[rotated, OF "log-prop-prop:2"], THEN "≡E"(2)] B by blast
1332 AOT_hence <¬s ⊨ □r>
1333   using "sit-classical:1"[THEN "→E", OF asm,
1334     THEN "∀E"(1)[rotated, OF "log-prop-prop:2"], THEN "≡E"(1)] by blast
1335 AOT_hence <□s ⊨ r & ¬s ⊨ □r>
1336   using 1 "&I" by blast
1337 AOT_thus <∃r (□s ⊨ r & ¬s ⊨ □r)>
1338   by (rule "∃I")
1339 qed
1340
1341 AOT_theorem "sit-classical:6":
1342   <∃s ∀p (s ⊨ p ≡ p)>
1343 proof -
1344   have "cond-prop": <ConditionOnPropositionalProperties
1345     (λ Π . «∃q (q & Π = [λy q])»)>
1346 proof (safe intro!: "cond-prop[I]" GEN "→I")
1347   fix F
1348   AOT_modally_strict {
1349     AOT_assume <∃q (q & F = [λy q])>
1350     then AOT_obtain q where <q & F = [λy q]>
1351       using "∃E"[rotated] by blast
1352     AOT_hence <F = [λy q]>
1353       using "&E" by blast
1354     AOT_hence <∃q F = [λy q]>
1355       by (rule "∃I")
1356     AOT_thus <Propositional([F])>
1357       by (metis "≡dfI" "prop-prop1")
1358   }
1359 qed
1360 AOT_have <∃s ∀F (s[F] ≡ ∃q (q & F = [λy q]))>
1361   using "comp-sit:1"[OF "cond-prop"].
1362 then AOT_obtain s0 where s0_prop: <∀F (s0[F] ≡ ∃q (q & F = [λy q]))>
1363   using "Situation.∃E"[rotated] by meson
1364 AOT_have <∀p (s0 ⊨ p ≡ p)>
1365 proof(safe intro!: GEN "≡I" "→I")
1366   fix p
1367   AOT_assume <s0 ⊨ p>
1368   AOT_hence <s0[λy p]>
1369     using lem1[THEN "→E", OF Situation.ψ, THEN "≡E"(1)] by blast
1370   AOT_hence <∃q (q & [λy p] = [λy q])>
1371     using s0_prop[THEN "∀E"(1)[rotated, OF "prop-prop2:2"], THEN "≡E"(1)] by blast
1372   then AOT_obtain q1 where q1_prop: <q1 & [λy p] = [λy q1]>
1373     using "∃E"[rotated] by blast
1374   AOT_hence <p = q1>
1375     by (metis "&E"(2) "≡E"(2) "p-identity-thm2:3")
1376   AOT_thus <p>
1377     using q1_prop[THEN "&E"(1)] "rule=E" id_sym by fast
1378 next
1379   fix p
1380   AOT_assume <p>
1381   moreover AOT_have <[λy p] = [λy p]>
1382     by (simp add: "rule=I:1"[OF "prop-prop2:2"])
1383   ultimately AOT_have <p & [λy p] = [λy p]>
1384     using "&I" by blast

```



```

1385   AOT_hence < $\exists q (q \ \& \ [\lambda y \ p] = [\lambda y \ q])$ >
1386     by (rule "EI")
1387   AOT_hence < $s_0 \ [\lambda y \ p]$ >
1388     using s0_prop[THEN "VE"(1)[rotated, OF "prop-prop2:2"], THEN "E"(2)] by blast
1389   AOT_thus < $s_0 \models p$ >
1390     using lem1[THEN "E", OF Situation. $\psi$ , THEN "E"(2)] by blast
1391   qed
1392   AOT_hence < $\forall p (s_0 \models p \equiv p)$ >
1393     using "&I" by blast
1394   AOT_thus < $\exists s \ \forall p (s \models p \equiv p)$ >
1395     by (rule "Situation.EI")
1396   qed
1397
1398   AOT_define PossibleWorld :: < $\tau \Rightarrow \varphi$ > (<PossibleWorld'('_)>)
1399     "world:1": <PossibleWorld(x)  $\equiv_{df}$  Situation(x) &  $\diamond \forall p (x \models p \equiv p)$ > (496.1)
1400
1401   AOT_theorem "world:2": < $\exists x \text{ PossibleWorld}(x)$ > (496.2)
1402   proof -
1403     AOT_obtain s where s_prop: < $\forall p (s \models p \equiv p)$ >
1404       using "sit-classical:6" "Situation.EI"[rotated] by meson
1405     AOT_have < $\forall p (s \models p \equiv p)$ >
1406     proof (safe intro!: GEN "EI" "E")
1407       fix p
1408       AOT_assume < $s \models p$ >
1409       AOT_thus <p>
1410         using s_prop[THEN "VE"(2), THEN "E"(1)] by blast
1411     next
1412       fix p
1413       AOT_assume <p>
1414       AOT_thus < $s \models p$ >
1415         using s_prop[THEN "VE"(2), THEN "E"(2)] by blast
1416     qed
1417     AOT_hence < $\diamond \forall p (s \models p \equiv p)$ >
1418       by (metis "T" $\diamond$ [THEN "E"])
1419     AOT_hence < $\diamond \forall p (s \models p \equiv p)$ >
1420       using s_prop "&I" by blast
1421     AOT_hence <PossibleWorld(s)>
1422       using "world:1"[THEN "E"] Situation. $\psi$  "&I" by blast
1423     AOT_thus < $\exists x \text{ PossibleWorld}(x)$ >
1424       by (rule "EI")
1425   qed
1426
1427   AOT_theorem "world:3": <PossibleWorld( $\kappa$ )  $\rightarrow \kappa \downarrow$ > (496.3)
1428   proof (rule "E")
1429     AOT_assume <PossibleWorld( $\kappa$ )>
1430     AOT_hence <Situation( $\kappa$ )>
1431       using "world:1"[THEN "E"] "&E" by blast
1432     AOT_hence <A! $\kappa$ >
1433       by (metis "E" "&E"(1) situations)
1434     AOT_thus < $\kappa \downarrow$ >
1435       by (metis "russell-axiom[exe,1]. $\psi$ _denotes_asm")
1436   qed
1437
1438   AOT_theorem "rigid-pw:1": <PossibleWorld(x)  $\equiv \square \text{PossibleWorld}(x)$ > (497.1)
1439   proof (safe intro!: "EI" "E")
1440     AOT_assume <PossibleWorld(x)>
1441     AOT_hence <Situation(x) &  $\diamond \forall p (x \models p \equiv p)$ >
1442       using "world:1"[THEN "E"] by blast
1443     AOT_hence < $\square \text{Situation}(x) \ \& \ \square \diamond \forall p (x \models p \equiv p)$ >
1444       by (metis "S5Basic:1" "&I" "&E"(1) "&E"(2) "E"(1) "possit-sit:1")
1445     AOT_hence 0: < $\square (\text{Situation}(x) \ \& \ \diamond \forall p (x \models p \equiv p))$ >
1446       by (metis "KBasic:3" "E"(2))
1447     AOT_show < $\square \text{PossibleWorld}(x)$ >

```



```

1448     by (AOT_subst <PossibleWorld(x)> <Situation(x) &  $\diamond \forall p(x \models p \equiv p)$ >)
1449     (auto simp: "≡Df" "world:1" 0)
1450 next
1451   AOT_show <PossibleWorld(x)> if < $\Box$ PossibleWorld(x)>
1452     using that "qml:2"[axiom_inst, THEN "→E"] by blast
1453 qed
1454
1455 AOT_theorem "rigid-pw:2": < $\diamond$ PossibleWorld(x)  $\equiv$  PossibleWorld(x)> (497.2)
1456   using "rigid-pw:1"
1457   by (meson "RE $\diamond$ " "S5Basic:2" "≡E"(2) "≡E"(6) "Commutativity of  $\equiv$ ")
1458
1459 AOT_theorem "rigid-pw:3": < $\diamond$ PossibleWorld(x)  $\equiv$   $\Box$ PossibleWorld(x)> (497.3)
1460   using "rigid-pw:1" "rigid-pw:2" by (meson "≡E"(5))
1461
1462 AOT_theorem "rigid-pw:4": < $\mathcal{A}$ PossibleWorld(x)  $\equiv$  PossibleWorld(x)> (497.4)
1463   by (metis "Act-Sub:3" "→I" "≡I" "≡E"(6) "nec-imp-act" "rigid-pw:1" "rigid-pw:2")
1464
1465 AOT_register_rigid_restricted_type
1466   PossibleWorld: <PossibleWorld( $\kappa$ )>
1467 proof
1468   AOT_modally_strict {
1469     AOT_show < $\exists x$  PossibleWorld(x)> using "world:2".
1470   }
1471 next
1472   AOT_modally_strict {
1473     AOT_show <PossibleWorld( $\kappa$ )  $\rightarrow$   $\kappa \downarrow$ > for  $\kappa$  using "world:3".
1474   }
1475 next
1476   AOT_modally_strict {
1477     AOT_show < $\forall \alpha$ (PossibleWorld( $\alpha$ )  $\rightarrow$   $\Box$ PossibleWorld( $\alpha$ ))>
1478     by (meson GEN "→I" "≡E"(1) "rigid-pw:1")
1479   }
1480 qed
1481 AOT_register_variable_names
1482   PossibleWorld: w
1483
1484 AOT_theorem "world-pos": <Possible(w)> (500)
1485 proof (safe intro!: "≡dfE"[OF "world:1", OF PossibleWorld. $\psi$ , THEN "&E"(1)]
1486   pos[THEN "≡dfI"] "&I" )
1487   AOT_have < $\diamond \forall p(w \models p \equiv p)$ >
1488     using "world:1"[THEN "≡dfE", OF PossibleWorld. $\psi$ , THEN "&E"(2)].
1489   AOT_hence < $\diamond \forall p(w \models p \rightarrow p)$ >
1490   proof (rule "RM $\diamond$ "[THEN "→E", rotated]; safe intro!: "→I" GEN)
1491     AOT_modally_strict {
1492       fix p
1493       AOT_assume < $\forall p(w \models p \equiv p)$ >
1494       AOT_hence < $w \models p \equiv p$ > using "VE"(2) by blast
1495       moreover AOT_assume < $w \models p$ >
1496       ultimately AOT_show p using "≡E"(1) by blast
1497     }
1498   qed
1499   AOT_hence 0: < $\diamond$ (Situation(w) &  $\forall p(w \models p \rightarrow p)$ )>
1500     using "world:1"[THEN "≡dfE", OF PossibleWorld. $\psi$ , THEN "&E"(1),
1501     THEN "possit-sit:1"[THEN "≡E"(1)]]
1502   by (metis "KBasic:16" "&I" "vdash-properties:10")
1503   AOT_show < $\diamond$ Actual(w)>
1504     by (AOT_subst <Actual(w)> <Situation(w) &  $\forall p(w \models p \rightarrow p)$ >)
1505     (auto simp: actual "≡Df" 0)
1506 qed
1507
1508 AOT_theorem "world-cons:1": <Consistent(w)> (501.1)
1509   using "world-pos"
1510   using "pos-cons-sit:1"[unconstrain s, THEN "→E", THEN "→E"]

```

```

1511   by (meson "≡dfE" "&E"(1) pos)
1512
1513 AOT_theorem "world-cons:2": <¬TrivialSituation(w)> (501.2)
1514 proof(rule "raa-cor:2")
1515   AOT_assume <TrivialSituation(w)>
1516   AOT_hence <Situation(w) & ∀p w ⊨ p>
1517     using "df-null-trivial:2"[THEN "≡dfE"] by blast
1518   AOT_hence 0: <□w ⊨ (∃p (p & ¬p))>
1519     using "&E"
1520     by (metis "Buridan◇" "T◇" "&E"(2) "≡E"(1) "lem2:3"[unconstrain s, THEN "→E"]
1521         "log-prop-prop:2" "rule-ui:1" "universal-cor" "→E")
1522   AOT_have <◇∀p (w ⊨ p ≡ p)>
1523     using PossibleWorld.ψ "world:1"[THEN "≡dfE", THEN "&E"(2)] by metis
1524   AOT_hence <∀p ◇(w ⊨ p ≡ p)>
1525     using "Buridan◇"[THEN "→E"] by blast
1526   AOT_hence <◇(w ⊨ (∃p (p & ¬p)) ≡ (∃p (p & ¬p)))>
1527     by (metis "log-prop-prop:2" "rule-ui:1")
1528   AOT_hence <◇(w ⊨ (∃p (p & ¬p)) → (∃p (p & ¬p)))>
1529     using "RM◇"[THEN "→E"] "→I" "≡E"(1) by meson
1530   AOT_hence <◇(∃p (p & ¬p))> using 0
1531     by (metis "KBasic2:4" "≡E"(1) "→E")
1532   moreover AOT_have <¬◇(∃p (p & ¬p))>
1533     by (metis "instantiation" "KBasic2:1" RN "≡E"(1) "raa-cor:2")
1534   ultimately AOT_show <◇(∃p (p & ¬p)) & ¬◇(∃p (p & ¬p))>
1535     using "&I" by blast
1536 qed
1537
1538 AOT_theorem "rigid-truth-at:1": <w ⊨ p ≡ □w ⊨ p> (502.1)
1539   using "lem2:1"[unconstrain s, THEN "→E",
1540     OF PossibleWorld.ψ[THEN "world:1"[THEN "≡dfE"], THEN "&E"(1)]] .
1541
1542 AOT_theorem "rigid-truth-at:2": <◇w ⊨ p ≡ w ⊨ p> (502.2)
1543   using "lem2:2"[unconstrain s, THEN "→E",
1544     OF PossibleWorld.ψ[THEN "world:1"[THEN "≡dfE"], THEN "&E"(1)]] .
1545
1546 AOT_theorem "rigid-truth-at:3": <◇w ⊨ p ≡ □w ⊨ p> (502.3)
1547   using "lem2:3"[unconstrain s, THEN "→E",
1548     OF PossibleWorld.ψ[THEN "world:1"[THEN "≡dfE"], THEN "&E"(1)]] .
1549
1550 AOT_theorem "rigid-truth-at:4": <Aw ⊨ p ≡ w ⊨ p> (502.4)
1551   using "lem2:4"[unconstrain s, THEN "→E",
1552     OF PossibleWorld.ψ[THEN "world:1"[THEN "≡dfE"], THEN "&E"(1)]] .
1553
1554 AOT_theorem "rigid-truth-at:5": <¬w ⊨ p ≡ □¬w ⊨ p> (502.5)
1555   using "lem2:5"[unconstrain s, THEN "→E",
1556     OF PossibleWorld.ψ[THEN "world:1"[THEN "≡dfE"], THEN "&E"(1)]] .
1557
1558 AOT_define Maximal :: <τ ⇒ φ> (<Maximal'('_)>)
1559   max: <Maximal(s) ≡df ∀p (s ⊨ p ∨ s ⊨ ¬p)> (503)
1560
1561 AOT_theorem "world-max": <Maximal(w)> (504)
1562 proof(safe intro!: PossibleWorld.ψ[THEN "≡dfE"[OF "world:1"], THEN "&E"(1)]
1563     GEN "≡dfI"[OF max] "&I" )
1564   fix q
1565   AOT_have <◇(w ⊨ q ∨ w ⊨ ¬q)>
1566   proof(rule "RM◇"[THEN "→E"]; (rule "→I")?)
1567     AOT_modally_strict {
1568       AOT_assume <∀p (w ⊨ p ≡ p)>
1569       AOT_hence <w ⊨ q ≡ q> and <w ⊨ ¬q ≡ ¬q>
1570         using "∀E"(1)[rotated, OF "log-prop-prop:2"] by blast+
1571       AOT_thus <w ⊨ q ∨ w ⊨ ¬q>
1572         by (metis "∨I"(1) "∨I"(2) "≡E"(3) "reductio-aa:1")
1573     }

```

```

1574 next
1575   AOT_show <◇∀p (w ⊨ p ≡ p)>
1576   using PossibleWorld.ψ[THEN "≡dfE"[OF "world:1"], THEN "&E"(2)].
1577 qed
1578 AOT_hence <◇w ⊨ q ∨ ◇w ⊨ ¬q>
1579 using "KBasic2:2"[THEN "≡E"(1)] by blast
1580 AOT_thus <w ⊨ q ∨ w ⊨ ¬q>
1581 using "lem2:2"[unconstrain s, THEN "→E", unvarify p,
1582   OF PossibleWorld.ψ[THEN "≡dfE"[OF "world:1"], THEN "&E"(1)],
1583   THEN "≡E"(1), OF "log-prop-prop:2"]
1584 by (metis "∀I"(1) "∀I"(2) "∀E"(3) "raa-cor:2")
1585 qed
1586
1587 AOT_theorem "world=maxpos:1": <Maximal(x) → □Maximal(x)> (505.1)
1588 proof (AOT_subst <Maximal(x)> <Situation(x) & ∀p (x ⊨ p ∨ x ⊨ ¬p)>);
1589   safe intro!: max "≡Df" "→I"; frule "&E"(1); drule "&E"(2)
1590   AOT_assume sit_x: <Situation(x)>
1591   AOT_hence nec_sit_x: <□Situation(x)>
1592   by (metis "≡E"(1) "possit-sit:1")
1593   AOT_assume <∀p (x ⊨ p ∨ x ⊨ ¬p)>
1594   AOT_hence <x ⊨ p ∨ x ⊨ ¬p> for p
1595   using "∀E"(1)[rotated, OF "log-prop-prop:2"] by blast
1596   AOT_hence <□x ⊨ p ∨ □x ⊨ ¬p> for p
1597   using "lem2:1"[unconstrain s, THEN "→E", OF sit_x, unvarify p,
1598     OF "log-prop-prop:2", THEN "≡E"(1)]
1599   by (metis "∀I"(1) "∀I"(2) "∀E"(2) "raa-cor:1")
1600   AOT_hence <□(x ⊨ p ∨ x ⊨ ¬p)> for p
1601   by (metis "KBasic:15" "→E")
1602   AOT_hence <∀p □(x ⊨ p ∨ x ⊨ ¬p)>
1603   by (rule GEN)
1604   AOT_hence <□∀p (x ⊨ p ∨ x ⊨ ¬p)>
1605   by (rule BF[THEN "→E"])
1606   AOT_thus <□(Situation(x) & ∀p (x ⊨ p ∨ x ⊨ ¬p))>
1607   using nec_sit_x by (metis "KBasic:3" "&I" "≡E"(2))
1608 qed
1609
1610 AOT_theorem "world=maxpos:2": <PossibleWorld(x) ≡ Maximal(x) & Possible(x)> (505.2)
1611 proof(safe intro!: "≡I" "→I" "&I" "world-pos"[unconstrain w, THEN "→E"]
1612   "world-max"[unconstrain w, THEN "→E"]);
1613   frule "&E"(2); drule "&E"(1)
1614   AOT_assume pos_x: <Possible(x)>
1615   AOT_have <◇(Situation(x) & ∀p(x ⊨ p → p))>
1616   apply (AOT_subst (reverse) <Situation(x) & ∀p(x ⊨ p → p)> <Actual(x)>)
1617   using actual "≡Df" apply presburger
1618   using "≡dfE" "&E"(2) pos pos_x by blast
1619   AOT_hence 0: <◇∀p(x ⊨ p → p)>
1620   by (metis "KBasic2:3" "&E"(2) "vdash-properties:6")
1621   AOT_assume max_x: <Maximal(x)>
1622   AOT_hence sit_x: <Situation(x)> by (metis "≡dfE" max_x "&E"(1) max)
1623   AOT_have <□Maximal(x)> using "world=maxpos:1"[THEN "→E", OF max_x] by simp
1624   moreover AOT_have <□Maximal(x) → □(∀p(x ⊨ p → p) → ∀p (x ⊨ p ≡ p))>
1625   proof(safe intro!: "→I" RM GEN)
1626     AOT_modally_strict {
1627       fix p
1628       AOT_assume 0: <Maximal(x)>
1629       AOT_assume 1: <∀p (x ⊨ p → p)>
1630       AOT_show <x ⊨ p ≡ p>
1631       proof(safe intro!: "≡I" "→I" 1[THEN "∀E"(2), THEN "→E"]; rule "raa-cor:1")
1632         AOT_assume <¬x ⊨ p>
1633         AOT_hence <x ⊨ ¬p>
1634         using 0[THEN "≡dfE"[OF max], THEN "&E"(2), THEN "∀E"(2)]
1635         1 by (metis "∀E"(2))
1636       AOT_hence <¬p>

```

```

1637         using 1[THEN "\VE"(1), OF "log-prop-prop:2", THEN "\toE"] by blast
1638     moreover AOT_assume p
1639     ultimately AOT_show <p & \to p> using "&I" by blast
1640 qed
1641 }
1642 qed
1643 ultimately AOT_have <\(\forall p(x \models p \to p) \to \forall p (x \models p \equiv p))>
1644     using "\toE" by blast
1645 AOT_hence <\(\forall p(x \models p \to p) \to \Diamond \forall p(x \models p \equiv p))>
1646     by (metis "KBasic:13"[THEN "\toE"])
1647 AOT_hence <\(\forall p(x \models p \equiv p))>
1648     using 0 "\toE" by blast
1649 AOT_thus <PossibleWorld(x)>
1650     using "\equivI"[OF "world:1", OF "&I", OF sit_x] by blast
1651 qed
1652
1653 AOT_define NecImpl :: <\(\varphi \Rightarrow \varphi \Rightarrow \varphi) (infixl <\Rightarrow> 26)>
1654     "nec-impl-p:1": <p \Rightarrow q \equiv_{df} \Box(p \to q)>
1655 AOT_define NecEquiv :: <\(\varphi \Rightarrow \varphi \Rightarrow \varphi) (infixl <\Leftrightarrow> 21)>
1656     "nec-impl-p:2": <p \Leftrightarrow q \equiv_{df} (p \Rightarrow q) & (q \Rightarrow p)>
1657
1658 AOT_theorem "nec-equiv-nec-im": <p \Leftrightarrow q \equiv \Box(p \equiv q)>
1659 proof(safe intro!: "\equivI" "\toI")
1660   AOT_assume <p \Leftrightarrow q>
1661   AOT_hence <(p \Rightarrow q)> and <(q \Rightarrow p)>
1662     using "nec-impl-p:2"[THEN "\equivE"] "&E" by blast+
1663   AOT_hence <\Box(p \to q)> and <\Box(q \to p)>
1664     using "nec-impl-p:1"[THEN "\equivE"] by blast+
1665   AOT_thus <\Box(p \equiv q)> by (metis "KBasic:4" "&I" "\equivE"(2))
1666 next
1667   AOT_assume <\Box(p \equiv q)>
1668   AOT_hence <\Box(p \to q)> and <\Box(q \to p)>
1669     using "KBasic:4" "&E" "\equivE"(1) by blast+
1670   AOT_hence <(p \Rightarrow q)> and <(q \Rightarrow p)>
1671     using "nec-impl-p:1"[THEN "\equivI"] by blast+
1672   AOT_thus <p \Leftrightarrow q>
1673     using "nec-impl-p:2"[THEN "\equivI"] "&I" by blast
1674 qed
1675
1676 (* TODO: PLM: discuss these; still not in PLM *)
1677 AOT_theorem world_closed_lem_1_a:
1678   <(s \models (\varphi & \psi)) \to (\forall p (s \models p \equiv p) \to (s \models \varphi & s \models \psi))>
1679 proof(safe intro!: "\toI")
1680   AOT_assume <\forall p (s \models p \equiv p)>
1681   AOT_hence <s \models (\varphi & \psi) \equiv (\varphi & \psi)> and <s \models \varphi \equiv \varphi> and <s \models \psi \equiv \psi>
1682     using "\VE"(1)[rotated, OF "log-prop-prop:2"] by blast+
1683   moreover AOT_assume <s \models (\varphi & \psi)>
1684   ultimately AOT_show <s \models \varphi & s \models \psi>
1685     by (metis "&I" "&E"(1) "&E"(2) "\equivE"(1) "\equivE"(2))
1686 qed
1687
1688 AOT_theorem world_closed_lem_1_b:
1689   <(s \models \varphi & (\varphi \to q)) \to (\forall p (s \models p \equiv p) \to s \models q)>
1690 proof(safe intro!: "\toI")
1691   AOT_assume <\forall p (s \models p \equiv p)>
1692   AOT_hence <s \models \varphi \equiv \varphi> for \varphi
1693     using "\VE"(1)[rotated, OF "log-prop-prop:2"] by blast
1694   moreover AOT_assume <s \models \varphi & (\varphi \to q)>
1695   ultimately AOT_show <s \models q>
1696     by (metis "&E"(1) "&E"(2) "\equivE"(1) "\equivE"(2) "\toE")
1697 qed
1698
1699 AOT_theorem world_closed_lem_1_c:

```

```

1700   <(s ⊨ φ & s ⊨ (φ → ψ)) → (∀p (s ⊨ p ≡ p) → s ⊨ ψ)>
1701 proof(safe intro!: "→I")
1702   AOT_assume <∀ p (s ⊨ p ≡ p)>
1703   AOT_hence <s ⊨ φ ≡ φ> for φ
1704   using "∀E"(1)[rotated, OF "log-prop-prop:2"] by blast
1705   moreover AOT_assume <s ⊨ φ & s ⊨ (φ → ψ)>
1706   ultimately AOT_show <s ⊨ ψ>
1707   by (metis "&E"(1) "&E"(2) "≡E"(1) "≡E"(2) "→E")
1708 qed
1709
1710 AOT_theorem "world-closed-lem:1[0]":
1711   <q → (∀p (s ⊨ p ≡ p) → s ⊨ q)>
1712   by (meson "→I" "≡E"(2) "log-prop-prop:2" "rule-ui:1")
1713
1714 AOT_theorem "world-closed-lem:1[1]":
1715   <s ⊨ p1 & (p1 → q) → (∀p (s ⊨ p ≡ p) → s ⊨ q)>
1716   using world_closed_lem_1_b.
1717
1718 AOT_theorem "world-closed-lem:1[2]":
1719   <s ⊨ p1 & s ⊨ p2 & ((p1 & p2) → q) → (∀p (s ⊨ p ≡ p) → s ⊨ q)>
1720   using world_closed_lem_1_b world_closed_lem_1_a
1721   by (metis (full_types) "&I" "&E" "→I" "→E")
1722
1723 AOT_theorem "world-closed-lem:1[3]":
1724   <s ⊨ p1 & s ⊨ p2 & s ⊨ p3 & ((p1 & p2 & p3) → q) → (∀p (s ⊨ p ≡ p) → s ⊨ q)>
1725   using world_closed_lem_1_b world_closed_lem_1_a
1726   by (metis (full_types) "&I" "&E" "→I" "→E")
1727
1728 AOT_theorem "world-closed-lem:1[4]":
1729   <s ⊨ p1 & s ⊨ p2 & s ⊨ p3 & s ⊨ p4 & ((p1 & p2 & p3 & p4) → q) →
1730   (∀p (s ⊨ p ≡ p) → s ⊨ q)>
1731   using world_closed_lem_1_b world_closed_lem_1_a
1732   by (metis (full_types) "&I" "&E" "→I" "→E")
1733
1734 AOT_theorem "coherent:1": <w ⊨ ¬p ≡ ¬w ⊨ p>
1735 proof(safe intro!: "≡I" "→I")
1736   AOT_assume 1: <w ⊨ ¬p>
1737   AOT_show <¬w ⊨ p>
1738   proof(rule "raa-cor:2")
1739     AOT_assume <w ⊨ p>
1740     AOT_hence <w ⊨ p & w ⊨ ¬p> using 1 "&I" by blast
1741     AOT_hence <∃q (w ⊨ q & w ⊨ ¬q)> by (rule "∃I")
1742     moreover AOT_have <¬∃q (w ⊨ q & w ⊨ ¬q)>
1743     using "world-cons:1"[THEN "≡defE"[OF cons], THEN "&E"(2)].
1744     ultimately AOT_show <∃q (w ⊨ q & w ⊨ ¬q) & ¬∃q (w ⊨ q & w ⊨ ¬q)>
1745     using "&I" by blast
1746   qed
1747 next
1748   AOT_assume <¬w ⊨ p>
1749   AOT_thus <w ⊨ ¬p>
1750   using "world-max"[THEN "≡defE"[OF max], THEN "&E"(2)]
1751   by (metis "∀E"(2) "log-prop-prop:2" "rule-ui:1")
1752 qed
1753
1754 AOT_theorem "coherent:2": <w ⊨ p ≡ ¬w ⊨ ¬p>
1755   by (metis "coherent:1" "deduction-theorem" "≡I" "≡E"(1) "≡E"(2) "raa-cor:3")
1756
1757 AOT_theorem "act-world:1": <∃w ∀p (w ⊨ p ≡ p)>
1758 proof -
1759   AOT_obtain s where s_prop: <∀p (s ⊨ p ≡ p)>
1760   using "sit-classical:6" "Situation.∃E"[rotated] by meson
1761   AOT_hence <◇∀p (s ⊨ p ≡ p)>
1762   by (metis "T◇" "vdash-properties:10")

```

```

1763   AOT_hence <PossibleWorld(s)>
1764     using "world:1"[THEN "≡dfI"] Situation.ψ "&I" by blast
1765   AOT_hence <PossibleWorld(s) & ∀p (s ⊨ p ≡ p)>
1766     using "&I" s_prop by blast
1767   thus ?thesis by (rule "∃I")
1768 qed
1769
1770 AOT_theorem "act-world:2": <∃!w Actual(w)> (514.2)
1771 proof -
1772   AOT_obtain w where w_prop: <∀p (w ⊨ p ≡ p)>
1773     using "act-world:1" "PossibleWorld.∃E"[rotated] by meson
1774   AOT_have sit_s: <Situation(w)>
1775     using PossibleWorld.ψ "world:1"[THEN "≡dfE", THEN "&E"(1)] by blast
1776   show ?thesis
1777   proof (safe intro!: "uniqueness:1"[THEN "≡dfI"] "∃I"(2) "&I" GEN "→I"
1778         PossibleWorld.ψ actual[THEN "≡dfI"] sit_s
1779         "sit-identity"[unconstrain s, unconstrain s', THEN "→E",
1780           THEN "→E", THEN "≡E"(2)] "≡I"
1781         w_prop[THEN "∀E"(2), THEN "≡E"(1)])
1782     AOT_show <PossibleWorld(w)> using PossibleWorld.ψ.
1783   next
1784     AOT_show <Situation(w)>
1785       by (simp add: sit_s)
1786   next
1787     fix y p
1788     AOT_assume w_asm: <PossibleWorld(y) & Actual(y)>
1789     AOT_assume <w ⊨ p>
1790     AOT_hence p: <p>
1791       using w_prop[THEN "∀E"(2), THEN "≡E"(1)] by blast
1792     AOT_show <y ⊨ p>
1793     proof(rule "raa-cor:1")
1794       AOT_assume <¬y ⊨ p>
1795       AOT_hence <y ⊨ ¬p>
1796         by (metis "coherent:1"[unconstrain w, THEN "→E"] "&E"(1) "≡E"(2) w_asm)
1797       AOT_hence <¬p>
1798         using w_asm[THEN "&E"(2), THEN actual[THEN "≡dfE"], THEN "&E"(2),
1799           THEN "∀E"(1), rotated, OF "log-prop-prop:2"]
1800         "→E" by blast
1801       AOT_thus <p & ¬p> using p "&I" by blast
1802     qed
1803   next
1804     AOT_show <w ⊨ p> if <y ⊨ p> and <PossibleWorld(y) & Actual(y)> for p y
1805       using that(2)[THEN "&E"(2), THEN actual[THEN "≡dfE"], THEN "&E"(2),
1806         THEN "∀E"(2), THEN "→E", OF that(1)]
1807       w_prop[THEN "∀E"(2), THEN "≡E"(2)] by blast
1808   next
1809     AOT_show <Situation(y)> if <PossibleWorld(y) & Actual(y)> for y
1810       using that[THEN "&E"(1)] "world:1"[THEN "≡dfE", THEN "&E"(1)] by blast
1811   next
1812     AOT_show <Situation(w)>
1813       using sit_s by blast
1814   qed(simp)
1815 qed
1816
1817 AOT_theorem "pre-walpha": <⊥w Actual(w)↓> (516)
1818   using "A-Exists:2" "RA[2]" "act-world:2" "≡E"(2) by blast
1819
1820 AOT_define TheActualWorld :: <κs> (<wα>) (517)
1821   "w-alpha": <wα =df ⊥w Actual(w)>
1822
1823 (* TODO: not in PLM *)
1824 AOT_theorem true_in_truth_act_true: <⊤ ⊨ p ≡  $\mathcal{A}p$ >
1825 proof(safe intro!: "≡I" "→I")

```

```

1826 AOT_have true_def: < $\vdash_{\square} \top = \iota x (A!x \ \& \ \forall F (x[F] \equiv \exists p(p \ \& \ F = [\lambda y \ p])))$ >
1827   by (simp add: "A-descriptions" "rule-id-df:1[zero]" "the-true:1")
1828 AOT_hence true_den: < $\vdash_{\square} \top \downarrow$ >
1829   using "t=t-proper:1" "vdash-properties:6" by blast
1830 {
1831   AOT_assume < $\top \models p$ >
1832   AOT_hence < $\top \models [\lambda y \ p]$ >
1833     by (metis "≡dfE" "con-dis-i-e:2:b" "prop-enc" "true-in-s")
1834   AOT_hence < $\iota x (A!x \ \& \ \forall F (x[F] \equiv \exists q (q \ \& \ F = [\lambda y \ q]))) [\lambda y \ p]$ >
1835     using "rule=E" true_def true_den by fast
1836   AOT_hence < $\mathcal{A}\exists q (q \ \& \ [\lambda y \ p] = [\lambda y \ q])$ >
1837     using "≡E"(1) "desc-nec-encode:1"[unvarify F] "prop-prop2:2" by fast
1838   AOT_hence < $\exists q \ \mathcal{A}(q \ \& \ [\lambda y \ p] = [\lambda y \ q])$ >
1839     by (metis "Act-Basic:10" "≡E"(1))
1840   then AOT_obtain q where < $\mathcal{A}(q \ \& \ [\lambda y \ p] = [\lambda y \ q])$ >
1841     using "∃E"[rotated] by blast
1842   AOT_hence actq: < $\mathcal{A}q$ > and < $\mathcal{A}[\lambda y \ p] = [\lambda y \ q]$ >
1843     using "Act-Basic:2" "intro-elim:3:a" "&E" by blast+
1844   AOT_hence < $[\lambda y \ p] = [\lambda y \ q]$ >
1845     using "id-act:1"[unvarify  $\alpha \ \beta$ , THEN "≡E"(2)] "prop-prop2:2" by blast
1846   AOT_hence < $p = q$ >
1847     by (metis "intro-elim:3:b" "p-identity-thm2:3")
1848   AOT_thus < $\mathcal{A}p$ >
1849     using actq "rule=E" id_sym by blast
1850 }
1851 {
1852   AOT_assume < $\mathcal{A}p$ >
1853   AOT_hence < $\mathcal{A}(p \ \& \ [\lambda y \ p] = [\lambda y \ p])$ >
1854     by (auto intro!: "Act-Basic:2"[THEN "≡E"(2)] "&I"
1855         intro: "RA[2]" "=I"(1)[OF "prop-prop2:2"])
1856   AOT_hence < $\exists q \ \mathcal{A}(q \ \& \ [\lambda y \ p] = [\lambda y \ q])$ >
1857     using "∃I" by fast
1858   AOT_hence < $\mathcal{A}\exists q (q \ \& \ [\lambda y \ p] = [\lambda y \ q])$ >
1859     by (metis "Act-Basic:10" "≡E"(2))
1860   AOT_hence < $\iota x (A!x \ \& \ \forall F (x[F] \equiv \exists q (q \ \& \ F = [\lambda y \ q]))) [\lambda y \ p]$ >
1861     using "≡E"(2) "desc-nec-encode:1"[unvarify F] "prop-prop2:2" by fast
1862   AOT_hence < $\top \models [\lambda y \ p]$ >
1863     using "rule=E" true_def true_den id_sym by fast
1864   AOT_thus < $\top \models p$ >
1865     by (safe intro!: "true-in-s"[THEN "≡dfI"] "&I" "possit-sit:6"
1866         "prop-enc"[THEN "≡dfI"] true_den)
1867 }
1868 qed
1869
1870 AOT_theorem "T-world": < $\top = w_{\alpha}$ > (518)
1871 proof -
1872   AOT_have true_den: < $\vdash_{\square} \top \downarrow$ >
1873     using "Situation.res-var:3" "possit-sit:6" "→E" by blast
1874   AOT_have < $\mathcal{A}\forall p (\top \models p \rightarrow p)$ >
1875   proof (safe intro!: "logic-actual-nec:3"[axiom_inst, THEN "≡E"(2)] GEN
1876         "logic-actual-nec:2"[axiom_inst, THEN "≡E"(2)] "→I")
1877     fix p
1878     AOT_assume < $\mathcal{A}\top \models p$ >
1879     AOT_hence < $\top \models p$ >
1880       using "lem2:4"[unconstrain s, unvarify  $\beta$ , OF true_den,
1881           THEN "→E", OF "possit-sit:6"] "≡E"(1) by blast
1882     AOT_thus < $\mathcal{A}p$ > using true_in_truth_act_true "≡E"(1) by blast
1883   qed
1884   moreover AOT_have < $\mathcal{A}(\text{Situation}(\kappa) \ \& \ \forall p (\kappa \models p \rightarrow p)) \rightarrow \mathcal{A}\text{Actual}(\kappa)$ > for  $\kappa$ 
1885     using actual[THEN "≡Df", THEN "conventions:3"[THEN "≡dfE", THEN "&E"(2)],
1886         THEN "RA[2]", THEN "act-cond"[THEN "→E"]].
1887   ultimately AOT_have act_act_true: < $\mathcal{A}\text{Actual}(\top)$ >
1888     using "possit-sit:4"[unvarify x, OF true_den, THEN "≡E"(2), OF "possit-sit:6"]

```



```

1889     "Act-Basic:2"[THEN "≡E"(2), OF "&I"] "→E" by blast
1890 AOT_hence <◇Actual(T)> by (metis "Act-Sub:3" "vdash-properties:10")
1891 AOT_hence <Possible(T)>
1892   by (safe intro!: pos[THEN "≡dfI"] "&I" "possit-sit:6")
1893 moreover AOT_have <Maximal(T)>
1894 proof (safe intro!: max[THEN "≡dfI"] "&I" "possit-sit:6" GEN)
1895   fix p
1896   AOT_have <Ap ∨ A¬p>
1897     by (simp add: "Act-Basic:1")
1898   moreover AOT_have <T ⊨ p> if <Ap>
1899     using that true_in_truth_act_true[THEN "≡E"(2)] by blast
1900   moreover AOT_have <T ⊨ ¬p> if <A¬p>
1901     using that true_in_truth_act_true[unvarify p, THEN "≡E"(2)]
1902     "log-prop-prop:2" by blast
1903   ultimately AOT_show <T ⊨ p ∨ T ⊨ ¬p>
1904     using "∨I"(3) "→I" by blast
1905 qed
1906 ultimately AOT_have <PossibleWorld(T)>
1907   by (safe intro!: "world=maxpos:2"[unvarify x, OF true_den, THEN "≡E"(2)] "&I")
1908 AOT_hence <APossibleWorld(T)>
1909   using "rigid-pw:4"[unvarify x, OF true_den, THEN "≡E"(2)] by blast
1910 AOT_hence 1: <A(PossibleWorld(T) & Actual(T))>
1911   using act_act_true "Act-Basic:2" "df-simplify:2" "intro-elim:3:b" by blast
1912 AOT_have <wα = ℓw(Actual(w))>
1913   using "rule-id-df:1[zero]"[OF "w-alpha", OF "pre-walpha"] by simp
1914 moreover AOT_have w_act_den: <wα↓>
1915   using calculation "t=t-proper:1" "→E" by blast
1916 ultimately AOT_have <∀z (A(PossibleWorld(z) & Actual(z)) → z = wα)>
1917   using "nec-hintikka-scheme"[unvarify x] "≡E"(1) "&E" by blast
1918 AOT_thus <T = wα>
1919   using "∇E"(1)[rotated, OF true_den] 1 "→E" by blast
1920 qed
1921
1922 AOT_act_theorem "truth-at-alpha:1": <p ≡ wα = ℓx (ExtensionOf(x, p))> (519.1)
1923   by (metis "rule=E" "T-world" "deduction-theorem" "ext-p-tv:3" id_sym "≡I"
1924     "≡E"(1) "≡E"(2) "q-True:1")
1925
1926 AOT_act_theorem "truth-at-alpha:2": <p ≡ wα ⊨ p> (519.2)
1927 proof -
1928   AOT_have <PossibleWorld(wα)>
1929     using "&E"(1) "pre-walpha" "rule-id-df:2:b[zero]" "→E"
1930     "w-alpha" "y-in:3" by blast
1931   AOT_hence sit_w_alpha: <Situation(wα)>
1932     by (metis "≡dfE" "&E"(1) "world:1")
1933   AOT_have w_alpha_den: <wα↓>
1934     using "pre-walpha" "rule-id-df:2:b[zero]" "w-alpha" by blast
1935   AOT_have <p ≡ TΣp>
1936     using "q-True:3" by force
1937   moreover AOT_have <T = wα>
1938     using "T-world" by auto
1939   ultimately AOT_have <p ≡ wαΣp>
1940     using "rule=E" by fast
1941   moreover AOT_have <wα Σ p ≡ wα ⊨ p>
1942     using lem1[unvarify x, OF w_alpha_den, THEN "→E", OF sit_w_alpha]
1943     using "≡S"(1) "≡E"(1) "Commutativity of ≡" "≡Df" sit_w_alpha "true-in-s" by blast
1944   ultimately AOT_show <p ≡ wα ⊨ p>
1945     by (metis "≡E"(5))
1946 qed
1947
1948 AOT_theorem "alpha-world:1": <PossibleWorld(wα)> (520.1)
1949 proof -
1950   AOT_have 0: <wα = ℓw Actual(w)>
1951     using "pre-walpha" "rule-id-df:1[zero]" "w-alpha" by blast

```



```

1952 AOT_hence walpha_den: <wα↓>
1953   by (metis "t=t-proper:1" "vdash-properties:6")
1954 AOT_have <A(PossibleWorld(wα) & Actual(wαα)>
1957   by (metis "Act-Basic:2" "&E"(1) "≡E"(1))
1958 AOT_thus <PossibleWorld(wα)>
1959   using "rigid-pw:4"[unvarify x, OF walpha_den, THEN "≡E"(1)]
1960   by blast
1961 qed
1962
1963 AOT_theorem "alpha-world:2": <Maximal(wα)> (520.2)
1964 proof -
1965   AOT_have <wα↓>
1966     using "pre-walphi" "rule-id-df:2:b[zero]" "w-alpha" by blast
1967   then AOT_obtain x where x_def: <x = wα>
1968     by (metis "instantiation" "rule=I:1" "existential:1" id_sym)
1969   AOT_hence <PossibleWorld(x)> using "alpha-world:1" "rule=E" id_sym by fast
1970   AOT_hence <Maximal(x)> by (metis "world-max"[unconstrain w, THEN "→E"])
1971   AOT_thus <Maximal(wα)> using x_def "rule=E" by blast
1972 qed
1973
1974 AOT_theorem "t-at-alpha-strict": <wα ⊨ p ≡ Ap> (521)
1975 proof -
1976   AOT_have 0: <wα = lw Actual(w)>
1977     using "pre-walphi" "rule-id-df:1[zero]" "w-alpha" by blast
1978   AOT_hence walphi_den: <wα↓>
1979     by (metis "t=t-proper:1" "vdash-properties:6")
1980   AOT_have 1: <A(PossibleWorld(wα) & Actual(wαα)>
1983     by (meson "≡dfE" "alpha-world:2" "&E"(1) max)
1984   {
1985     fix p
1986     AOT_have 2: <Situation(x) → (Ax ⊨ p ≡ x ⊨ p)> for x
1987       using "lem2:4"[unconstrain s] by blast
1988     AOT_assume <wα ⊨ p>
1989     AOT_hence ϑ: <Awα ⊨ p>
1990       using 2[unvarify x, OF walphi_den, THEN "→E", OF walphi_sit, THEN "≡E"(2)]
1991       by argo
1992     AOT_have 3: <AActual(wα)>
1993       using "1" "Act-Basic:2" "&E"(2) "≡E"(1) by blast
1994     AOT_have <A(Situation(wα) & ∀q(wα ⊨ q → q))>
1995       apply (AOT_subst (reverse) <Situation(wα) & ∀q(wα ⊨ q → q)> <Actual(wα)>)
1996       using actual "≡Df" apply blast
1997       by (fact 3)
1998     AOT_hence <A∀q(wα ⊨ q → q)> by (metis "Act-Basic:2" "&E"(2) "≡E"(1))
1999     AOT_hence <∀q A(wα ⊨ q → q)>
2000       using "logic-actual-nec:3"[axiom_inst, THEN "≡E"(1)] by blast
2001     AOT_hence <A(wα ⊨ p → p)> using "VE"(2) by blast
2002     AOT_hence <A(wα ⊨ p) → Ap> by (metis "act-cond" "vdash-properties:10")
2003     AOT_hence <Ap> using ϑ "→E" by blast
2004   }
2005   AOT_hence 2: <wα ⊨ p → Ap> for p by (rule "→I")
2006   AOT_have walphi_sit: <Situation(wα)>
2007     using "≡dfE" "alpha-world:2" "&E"(1) max by blast
2008   show ?thesis
2009   proof (safe intro!: "≡I" "→I" 2)
2010     AOT_assume actp: <Ap>
2011     AOT_show <wα ⊨ p>
2012     proof (rule "raa-cor:1")
2013       AOT_assume <¬wα ⊨ p>
2014       AOT_hence <wα ⊨ ¬p>

```

```

2015     using "alpha-world:2"[THEN max[THEN "≡dfE"], THEN "&E"(2),
2016     THEN "∀E"(1), OF "log-prop-prop:2"]
2017     by (metis "∀E"(2))
2018 AOT_hence <A¬p>
2019     using 2[unvarify p, OF "log-prop-prop:2", THEN "→E"] by blast
2020 AOT_hence <¬Ap> by (metis "¬¬I" "Act-Sub:1" "≡E"(4))
2021 AOT_thus <Ap & ¬Ap> using actp "&I" by blast
2022 qed
2023 qed
2024 qed
2025
2026 AOT_act_theorem "not-act": <w ≠ wα → ¬Actual(w)> (522)
2027 proof (rule "→I"; rule "raa-cor:2")
2028   AOT_assume <w ≠ wα>
2029   AOT_hence 0: <¬(w = wα)> by (metis "≡dfE" "--infix")
2030   AOT_have walpha_den: <wα↓>
2031     using "pre-walphi" "rule-id-df:2:b[zero]" "w-alpha" by blast
2032   AOT_have walphi_sit: <Situation(wα)>
2033     using "≡dfE" "alpha-world:2" "&E"(1) max by blast
2034   AOT_assume act_w: <Actual(w)>
2035   AOT_hence w_sit: <Situation(w)> by (metis "≡dfE" actual "&E"(1))
2036   AOT_have sid: <Situation(x') → (w = x' ≡ ∀p (w ⊨ p ≡ x' ⊨ p))> for x'
2037     using "sit-identity"[unconstrain s', unconstrain s, THEN "→E", OF w_sit]
2038     by blast
2039   AOT_have <w = wα>
2040   proof(safe intro!: GEN sid[unvarify x', OF walphi_den, THEN "→E",
2041     OF walphi_sit, THEN "≡E"(2)] "≡I" "→I")
2042     fix p
2043     AOT_assume <w ⊨ p>
2044     AOT_hence <p>
2045       using actual[THEN "≡dfE", OF act_w, THEN "&E"(2), THEN "∀E"(2), THEN "→E"]
2046       by blast
2047     AOT_hence <Ap>
2048       by (metis "RA[1]")
2049     AOT_thus <wα ⊨ p>
2050       using "t-at-alpha-strict"[THEN "≡E"(2)] by blast
2051   next
2052     fix p
2053     AOT_assume <wα ⊨ p>
2054     AOT_hence <Ap>
2055       using "t-at-alpha-strict"[THEN "≡E"(1)] by blast
2056     AOT_hence p: <p>
2057       using "logic-actual"[act_axiom_inst, THEN "→E"] by blast
2058     AOT_show <w ⊨ p>
2059     proof(rule "raa-cor:1")
2060       AOT_assume <¬w ⊨ p>
2061       AOT_hence <w ⊨ ¬p>
2062         by (metis "coherent:1" "≡E"(2))
2063       AOT_hence <¬p>
2064         using actual[THEN "≡dfE", OF act_w, THEN "&E"(2), THEN "∀E"(1),
2065           OF "log-prop-prop:2", THEN "→E"] by blast
2066       AOT_thus <p & ¬p> using p "&I" by blast
2067     qed
2068   qed
2069   AOT_thus <w = wα & ¬(w = wα)> using 0 "&I" by blast
2070 qed
2071
2072 AOT_act_theorem "w-alpha-part": <Actual(s) ≡ s ≤ wα> (523)
2073 proof(safe intro!: "≡I" "→I" "sit-part-whole"[THEN "≡dfI"] "&I" GEN
2074   dest!: "sit-part-whole"[THEN "≡dfE"])
2075   AOT_show <Situation(s)> if <Actual(s)>
2076   using "≡dfE" actual "&E"(1) that by blast
2077 next

```

```

2078   AOT_show <Situation(wα)>
2079     using "≡dfE" "alpha-world:2" "&E"(1) max by blast
2080 next
2081   fix p
2082   AOT_assume <Actual(s)>
2083   moreover AOT_assume <s ⊨ p>
2084   ultimately AOT_have <p>
2085     using actual[THEN "≡dfE", THEN "&E"(2), THEN "∀E"(2), THEN "→E"] by blast
2086   AOT_thus <wα ⊨ p>
2087     by (metis "≡E"(1) "truth-at-alpha:2")
2088 next
2089   AOT_assume 0: <Situation(s) & Situation(wα) & ∀p (s ⊨ p → wα ⊨ p)>
2090   AOT_hence <s ⊨ p → wα ⊨ p> for p
2091     using "&E" "∀E"(2) by blast
2092   AOT_hence <s ⊨ p → p> for p
2093     by (metis "→I" "≡E"(2) "truth-at-alpha:2" "→E")
2094   AOT_hence <∀p (s ⊨ p → p)> by (rule GEN)
2095   AOT_thus <Actual(s)>
2096     using actual[THEN "≡dfI", OF "&I", OF 0[THEN "&E"(1), THEN "&E"(1)]] by blast
2097 qed
2098
2099 AOT_act_theorem "act-world2:1": <wα ⊨ p ≡ [λy p]wα> (524.1)
2100   apply (AOT_subst <[λy p]wα> p)
2101   apply (rule "beta-C-meta"[THEN "→E", OF "prop-prop2:2", unvarify ν1νn])
2102   using "pre-walpha" "rule-id-df:2:b[zero]" "w-alpha" apply blast
2103   using "≡E"(2) "Commutativity of ≡" "truth-at-alpha:2" by blast
2104
2105 AOT_act_theorem "act-world2:2": <p ≡ wα ⊨ [λy p]wα> (524.2)
2106 proof -
2107   AOT_have <p ≡ [λy p]wα>
2108     apply (rule "beta-C-meta"[THEN "→E", OF "prop-prop2:2",
2109       unvarify ν1νn, symmetric])
2110     using "pre-walpha" "rule-id-df:2:b[zero]" "w-alpha" by blast
2111   also AOT_have <... ≡ wα ⊨ [λy p]wα>
2112     by (meson "log-prop-prop:2" "rule-ui:1" "truth-at-alpha:2" "universal-cor")
2113   finally show ?thesis.
2114 qed
2115
2116 AOT_theorem "fund-lem:1": <◇p → ◇∃w (w ⊨ p)> (525.1)
2117 proof (rule "RM◇"; rule "→I"; rule "raa-cor:1")
2118   AOT_modally_strict {
2119     AOT_obtain w where w_prop: <∀q (w ⊨ q ≡ q)>
2120       using "act-world:1" "PossibleWorld.∃E"[rotated] by meson
2121     AOT_assume p: <p>
2122     AOT_assume 0: <¬∃w (w ⊨ p)>
2123     AOT_have <∀w ¬(w ⊨ p)>
2124       apply (AOT_subst <PossibleWorld(x) → ¬x ⊨ p>
2125         <¬(PossibleWorld(x) & x ⊨ p)> for: x)
2126       apply (metis "&I" "&E"(1) "&E"(2) "→I" "≡I" "modus-tollens:2")
2127       using "0" "cqt-further:4" "vdash-properties:10" by blast
2128     AOT_hence <¬(w ⊨ p)>
2129       using PossibleWorld.ψ "rule-ui:3" "→E" by blast
2130     AOT_hence <¬p>
2131       using w_prop[THEN "∀E"(2), THEN "≡E"(2)]
2132       by (metis "raa-cor:3")
2133     AOT_thus <p & ¬p>
2134       using p "&I" by blast
2135   }
2136 qed
2137
2138 AOT_theorem "fund-lem:2": <◇∃w (w ⊨ p) → ∃w (w ⊨ p)> (525.2)
2139 proof (rule "→I")
2140   AOT_assume <◇∃w (w ⊨ p)>

```

```

2141 AOT_hence < $\exists w \diamond(w \models p)$ >
2142   using "PossibleWorld.res-var-bound-reas[BF $\diamond$ ]"[THEN " $\rightarrow$ E"] by auto
2143 then AOT_obtain w where < $\diamond(w \models p)$ >
2144   using "PossibleWorld. $\exists$ E"[rotated] by meson
2145 moreover AOT_have <Situation(w)>
2146   by (metis " $\equiv_{df}$ E" "&E"(1) pos "world-pos")
2147 ultimately AOT_have < $w \models p$ >
2148   using "lem2:2"[unconstrain s, THEN " $\rightarrow$ E"] " $\equiv$ E" by blast
2149 AOT_thus < $\exists w w \models p$ >
2150   by (rule "PossibleWorld. $\exists$ I")
2151 qed
2152
2153 AOT_theorem "fund-lem:3": < $p \rightarrow \forall s(\forall q (s \models q \equiv q) \rightarrow s \models p)$ > (525.3)
2154 proof(safe intro!: " $\rightarrow$ I" Situation.GEN)
2155   fix s
2156   AOT_assume <p>
2157   moreover AOT_assume < $\forall q (s \models q \equiv q)$ >
2158   ultimately AOT_show < $s \models p$ >
2159   using " $\forall$ E"(2) " $\equiv$ E"(2) by blast
2160 qed
2161
2162 AOT_theorem "fund-lem:4": < $\Box p \rightarrow \Box \forall s(\forall q (s \models q \equiv q) \rightarrow s \models p)$ > (525.4)
2163   using "fund-lem:3" by (rule RM)
2164
2165 AOT_theorem "fund-lem:5": < $\Box \forall s \varphi\{s\} \rightarrow \forall s \Box \varphi\{s\}$ > (525.5)
2166 proof(safe intro!: " $\rightarrow$ I" Situation.GEN)
2167   fix s
2168   AOT_assume < $\Box \forall s \varphi\{s\}$ >
2169   AOT_hence < $\forall s \Box \varphi\{s\}$ >
2170   using "Situation.res-var-bound-reas[CBF]"[THEN " $\rightarrow$ E"] by blast
2171   AOT_thus < $\Box \varphi\{s\}$ >
2172   using "Situation. $\forall$ E" by fast
2173 qed
2174
2175 text<Note: not explicit in PLM.>
2176 AOT_theorem "fund-lem:5[world]": < $\Box \forall w \varphi\{w\} \rightarrow \forall w \Box \varphi\{w\}$ > (525.5)
2177 proof(safe intro!: " $\rightarrow$ I" PossibleWorld.GEN)
2178   fix w
2179   AOT_assume < $\Box \forall w \varphi\{w\}$ >
2180   AOT_hence < $\forall w \Box \varphi\{w\}$ >
2181   using "PossibleWorld.res-var-bound-reas[CBF]"[THEN " $\rightarrow$ E"] by blast
2182   AOT_thus < $\Box \varphi\{w\}$ >
2183   using "PossibleWorld. $\forall$ E" by fast
2184 qed
2185
2186 AOT_theorem "fund-lem:6": < $\forall w w \models p \rightarrow \Box \forall w w \models p$ > (525.6)
2187 proof(rule " $\rightarrow$ I")
2188   AOT_assume < $\forall w (w \models p)$ >
2189   AOT_hence 1: <PossibleWorld(w)  $\rightarrow (w \models p)$ > for w
2190   using " $\forall$ E"(2) by blast
2191   AOT_show < $\Box \forall w w \models p$ >
2192   proof(rule "raa-cor:1")
2193     AOT_assume < $\neg \Box \forall w w \models p$ >
2194     AOT_hence < $\diamond \neg \forall w w \models p$ >
2195     by (metis "KBasic:11" " $\equiv$ E"(1))
2196     AOT_hence < $\diamond \exists x (\neg(\text{PossibleWorld}(x) \rightarrow x \models p))$ >
2197     apply (rule "RM $\diamond$ "[THEN " $\rightarrow$ E", rotated])
2198     by (simp add: "cqt-further:2")
2199     AOT_hence < $\exists x \diamond (\neg(\text{PossibleWorld}(x) \rightarrow x \models p))$ >
2200     by (metis "BF $\diamond$ " "vdash-properties:10")
2201     then AOT_obtain x where x_prop: < $\diamond \neg(\text{PossibleWorld}(x) \rightarrow x \models p)$ >
2202     using " $\exists$ E"[rotated] by blast
2203     AOT_have < $\diamond(\text{PossibleWorld}(x) \ \& \ \neg x \models p)$ >

```

```

2204     apply (AOT_subst <PossibleWorld(x) & ¬x ⊨ p>
2205             <¬(PossibleWorld(x) → x ⊨ p)>)
2206     apply (meson "≡E"(6) "oth-class-taut:1:b" "oth-class-taut:3:a")
2207     by (fact x_prop)
2208 AOT_hence 2: <◇PossibleWorld(x) & ◇¬x ⊨ p>
2209     by (metis "KBasic2:3" "vdash-properties:10")
2210 AOT_hence <PossibleWorld(x)>
2211     using "&E"(1) "≡E"(1) "rigid-pw:2" by blast
2212 AOT_hence <□(x ⊨ p)>
2213     using 2[THEN "&E"(2)] 1[unconstrain w, THEN "→E"] "→E"
2214         "rigid-truth-at:1"[unconstrain w, THEN "→E"]
2215     by (metis "≡E"(1))
2216 moreover AOT_have <¬□(x ⊨ p)>
2217     using 2[THEN "&E"(2)] by (metis "¬¬I" "KBasic:12" "≡E"(4))
2218 ultimately AOT_show <p & ¬p> for p
2219     by (metis "raa-cor:3")
2220 qed
2221 qed
2222
2223 AOT_theorem "fund-lem:7": <□∀w(w ⊨ p) → □p> (525.7)
2224 proof (rule RM; rule "→I")
2225   AOT_modally_strict {
2226     AOT_obtain w where w_prop: <∀p (w ⊨ p ≡ p)>
2227     using "act-world:1" "PossibleWorld.∃E"[rotated] by meson
2228     AOT_assume <∀w (w ⊨ p)>
2229     AOT_hence <w ⊨ p>
2230     using "PossibleWorld.∀E" by fast
2231     AOT_thus <p>
2232     using w_prop[THEN "∀E"(2), THEN "≡E"(1)] by blast
2233   }
2234 qed
2235
2236 AOT_theorem "fund:1": <◇p ≡ ∃w w ⊨ p> (526.1)
2237 proof (rule "≡I"; rule "→I")
2238   AOT_assume <◇p>
2239   AOT_thus <∃w w ⊨ p>
2240   by (metis "fund-lem:1" "fund-lem:2" "→E")
2241 next
2242   AOT_assume <∃w w ⊨ p>
2243   then AOT_obtain w where w_prop: <w ⊨ p>
2244   using "PossibleWorld.∃E"[rotated] by meson
2245   AOT_hence <◇∀p (w ⊨ p ≡ p)>
2246   using "world:1"[THEN "≡dfE", THEN "&E"(2)] PossibleWorld.ψ "&E" by blast
2247   AOT_hence <∀p ◇(w ⊨ p ≡ p)>
2248   by (metis "Buridan◇" "→E")
2249   AOT_hence 1: <◇(w ⊨ p ≡ p)>
2250   by (metis "log-prop-prop:2" "rule-ui:1")
2251   AOT_have <◇((w ⊨ p → p) & (p → w ⊨ p))>
2252   apply (AOT_subst <(w ⊨ p → p) & (p → w ⊨ p)> <w ⊨ p ≡ p>)
2253   apply (meson "conventions:3" "≡E"(6) "oth-class-taut:3:a" "≡Df")
2254   by (fact 1)
2255   AOT_hence <◇(w ⊨ p → p)>
2256   by (metis "RM◇" "Conjunction Simplification"(1) "→E")
2257   moreover AOT_have <□(w ⊨ p)>
2258   using w_prop by (metis "≡E"(1) "rigid-truth-at:1")
2259   ultimately AOT_show <◇p>
2260   by (metis "KBasic2:4" "≡E"(1) "→E")
2261 qed
2262
2263 AOT_theorem "fund:2": <□p ≡ ∀w (w ⊨ p)> (526.2)
2264 proof -
2265   AOT_have 0: <∀w (w ⊨ ¬p ≡ ¬w ⊨ p)>
2266   apply (rule PossibleWorld.GEN)

```

```

2267   using "coherent:1" by blast
2268 AOT_have < $\Diamond\neg p \equiv \exists w (w \models \neg p)$ >
2269   using "fund:1"[unvarify p, OF "log-prop-prop:2"] by blast
2270 also AOT_have < $\dots \equiv \exists w \neg(w \models p)$ >
2271 proof(safe intro!: "≡I" "→I")
2272   AOT_assume < $\exists w w \models \neg p$ >
2273   then AOT_obtain w where w_prop: < $w \models \neg p$ >
2274   using "PossibleWorld.∃E"[rotated] by meson
2275   AOT_hence < $\neg w \models p$ >
2276   using 0[THEN "PossibleWorld.∀E", THEN "≡E"(1)] "&E" by blast
2277   AOT_thus < $\exists w \neg w \models p$ >
2278   by (rule "PossibleWorld.∃I")
2279 next
2280   AOT_assume < $\exists w \neg w \models p$ >
2281   then AOT_obtain w where w_prop: < $\neg w \models p$ >
2282   using "PossibleWorld.∃E"[rotated] by meson
2283   AOT_hence < $w \models \neg p$ >
2284   using 0[THEN "∀E"(2), THEN "→E", THEN "≡E"(1)] "&E"
2285   by (metis "coherent:1" "≡E"(2))
2286   AOT_thus < $\exists w w \models \neg p$ >
2287   by (rule "PossibleWorld.∃I")
2288 qed
2289 finally AOT_have < $\neg\Diamond\neg p \equiv \neg\exists w \neg w \models p$ >
2290   by (meson "≡E"(1) "oth-class-taut:4:b")
2291 AOT_hence < $\Box p \equiv \neg\exists w \neg w \models p$ >
2292   by (metis "KBasic:12" "≡E"(5))
2293 also AOT_have < $\dots \equiv \forall w w \models p$ >
2294 proof(safe intro!: "≡I" "→I")
2295   AOT_assume < $\neg\exists w \neg w \models p$ >
2296   AOT_hence 0: < $\forall x (\neg(\text{PossibleWorld}(x) \ \& \ \neg x \models p))$ >
2297   by (metis "cqt-further:4" "→E")
2298   AOT_show < $\forall w w \models p$ >
2299   apply (AOT_subst <PossibleWorld(x) → x ≡ p>
2300         < $\neg(\text{PossibleWorld}(x) \ \& \ \neg x \models p)$ > for: x)
2301   using "oth-class-taut:1:a" apply presburger
2302   by (fact 0)
2303 next
2304   AOT_assume 0: < $\forall w w \models p$ >
2305   AOT_have < $\forall x (\neg(\text{PossibleWorld}(x) \ \& \ \neg x \models p))$ >
2306   by (AOT_subst (reverse) < $\neg(\text{PossibleWorld}(x) \ \& \ \neg x \models p)$ >
2307         <PossibleWorld(x) → x ≡ p> for: x)
2308   (auto simp: "oth-class-taut:1:a" 0)
2309   AOT_thus < $\neg\exists w \neg w \models p$ >
2310   by (metis "∃E" "raa-cor:3" "rule-ui:3")
2311 qed
2312 finally AOT_show < $\Box p \equiv \forall w w \models p$ >.
2313 qed
2314
2315 AOT_theorem "fund:3": < $\neg\Diamond p \equiv \neg\exists w w \models p$ > (526.3)
2316   by (metis (full_types) "contraposition:1[1]" "→I" "fund:1" "≡I" "≡E"(1,2))
2317
2318 AOT_theorem "fund:4": < $\neg\Box p \equiv \exists w \neg w \models p$ > (526.4)
2319   apply (AOT_subst < $\exists w \neg w \models p$ > < $\neg \forall w w \models p$ >)
2320   apply (AOT_subst <PossibleWorld(x) → x ≡ p>
2321         < $\neg(\text{PossibleWorld}(x) \ \& \ \neg x \models p)$ > for: x)
2322   by (auto simp add: "oth-class-taut:1:a" "conventions:4" "≡Df" RN
2323         "fund:2" "rule-sub-lem:1:a")
2324
2325 AOT_theorem "nec-dia-w:1": < $\Box p \equiv \exists w w \models \Box p$ > (527.1)
2326 proof -
2327   AOT_have < $\Box p \equiv \Diamond\Box p$ >
2328   using "S5Basic:2" by blast
2329 also AOT_have < $\dots \equiv \exists w w \models \Box p$ >

```

```

2330     using "fund:1"[unvarify p, OF "log-prop-prop:2"] by blast
2331     finally show ?thesis.
2332 qed
2333
2334 AOT_theorem "nec-dia-w:2": <□p ≡ ∀w w ⊨ □p> (527.2)
2335 proof -
2336   AOT_have <□p ≡ □□p>
2337     using 4 "qml:2"[axiom_inst] "≡I" by blast
2338   also AOT_have <... ≡ ∀w w ⊨ □p>
2339     using "fund:2"[unvarify p, OF "log-prop-prop:2"] by blast
2340   finally show ?thesis.
2341 qed
2342
2343 AOT_theorem "nec-dia-w:3": <◇p ≡ ∃w w ⊨ ◇p> (527.3)
2344 proof -
2345   AOT_have <◇p ≡ ◇◇p>
2346     by (simp add: "4◇" "T◇" "≡I")
2347   also AOT_have <... ≡ ∃w w ⊨ ◇p>
2348     using "fund:1"[unvarify p, OF "log-prop-prop:2"] by blast
2349   finally show ?thesis.
2350 qed
2351
2352 AOT_theorem "nec-dia-w:4": <◇p ≡ ∀w w ⊨ ◇p> (527.4)
2353 proof -
2354   AOT_have <◇p ≡ □◇p>
2355     by (simp add: "S5Basic:1")
2356   also AOT_have <... ≡ ∀w w ⊨ ◇p>
2357     using "fund:2"[unvarify p, OF "log-prop-prop:2"] by blast
2358   finally show ?thesis.
2359 qed
2360
2361 AOT_theorem "conj-dist-w:1": <w ⊨ (p & q) ≡ ((w ⊨ p) & (w ⊨ q))> (528.1)
2362 proof(safe intro!: "≡I" "→I")
2363   AOT_assume <w ⊨ (p & q)>
2364   AOT_hence 0: <□w ⊨ (p & q)>
2365     using "rigid-truth-at:1"[unvarify p, THEN "≡E"(1), OF "log-prop-prop:2"]
2366     by blast
2367   AOT_modally_strict {
2368     AOT_have <∀p (w ⊨ p ≡ p) → ((w ⊨ (φ & ψ)) → (w ⊨ φ & w ⊨ ψ))> for w φ ψ
2369     proof(safe intro!: "→I")
2370       AOT_assume <∀ p (w ⊨ p ≡ p)>
2371       AOT_hence <w ⊨ (φ & ψ) ≡ (φ & ψ)> and <w ⊨ φ ≡ φ> and <w ⊨ ψ ≡ ψ>
2372         using "∀E"(1)[rotated, OF "log-prop-prop:2"] by blast+
2373       moreover AOT_assume <w ⊨ (φ & ψ)>
2374       ultimately AOT_show <w ⊨ φ & w ⊨ ψ>
2375         by (metis "&I" "&E"(1) "&E"(2) "≡E"(1) "≡E"(2))
2376     qed
2377   }
2378   AOT_hence <◇∀p (w ⊨ p ≡ p) → ◇(w ⊨ (φ & ψ) → w ⊨ φ & w ⊨ ψ)> for w φ ψ
2379     by (rule "RM◇")
2380   moreover AOT_have pos: <◇∀p (w ⊨ p ≡ p)> (491)
2381     using "world:1"[THEN "≡dfE", OF PossibleWorld.ψ] "&E" by blast
2382   ultimately AOT_have <◇(w ⊨ (p & q) → w ⊨ p & w ⊨ q)> using "→E" by blast
2383   AOT_hence <◇(w ⊨ p) & ◇(w ⊨ q)>
2384     by (metis 0 "KBasic2:3" "KBasic2:4" "≡E"(1) "vdash-properties:10")
2385   AOT_thus <w ⊨ p & w ⊨ q>
2386     using "rigid-truth-at:2"[unvarify p, THEN "≡E"(1), OF "log-prop-prop:2"]
2387     "&E" "&I" by meson
2388 next
2389   AOT_assume <w ⊨ p & w ⊨ q>
2390   AOT_hence <□w ⊨ p & □w ⊨ q>
2391     using "rigid-truth-at:1"[unvarify p, THEN "≡E"(1), OF "log-prop-prop:2"]
2392     "&E" "&I" by blast

```



```

2393 AOT_hence 0: <□(w ⊨ p & w ⊨ q)>
2394   by (metis "KBasic:3" "≡E"(2))
2395 AOT_modally_strict {
2396   AOT_have <∀p (w ⊨ p ≡ p) → ((w ⊨ φ & w ⊨ ψ) → (w ⊨ (φ & ψ)))> for w φ ψ
2397   proof(safe intro!: "→I")
2398     AOT_assume <∀ p (w ⊨ p ≡ p)>
2399     AOT_hence <w ⊨ (φ & ψ) ≡ (φ & ψ)> and <w ⊨ φ ≡ φ> and <w ⊨ ψ ≡ ψ>
2400     using "∀E"(1)[rotated, OF "log-prop-prop:2"] by blast+
2401     moreover AOT_assume <w ⊨ φ & w ⊨ ψ>
2402     ultimately AOT_show <w ⊨ (φ & ψ)>
2403     by (metis "&I" "&E"(1) "&E"(2) "≡E"(1) "≡E"(2))
2404   qed
2405 }
2406 AOT_hence <◇∀p (w ⊨ p ≡ p) → ◇((w ⊨ φ & w ⊨ ψ) → w ⊨ (φ & ψ))> for w φ ψ
2407   by (rule "RM◇")
2408 moreover AOT_have pos: <◇∀p (w ⊨ p ≡ p)> (491)
2409   using "world:1"[THEN "≡defE", OF PossibleWorld.ψ] "&E" by blast
2410 ultimately AOT_have <◇((w ⊨ p & w ⊨ q) → w ⊨ (p & q))>
2411   using "→E" by blast
2412 AOT_hence <◇(w ⊨ (p & q))>
2413   by (metis 0 "KBasic2:4" "≡E"(1) "vdash-properties:10")
2414 AOT_thus <w ⊨ (p & q)>
2415   using "rigid-truth-at:2"[unvarify p, THEN "≡E"(1), OF "log-prop-prop:2"]
2416   by blast
2417 qed
2418
2419 AOT_theorem "conj-dist-w:2": <w ⊨ (p → q) ≡ ((w ⊨ p) → (w ⊨ q))> (528.2)
2420 proof(safe intro!: "≡I" "→I")
2421   AOT_assume <w ⊨ (p → q)>
2422   AOT_hence 0: <□w ⊨ (p → q)>
2423     using "rigid-truth-at:1"[unvarify p, THEN "≡E"(1), OF "log-prop-prop:2"]
2424     by blast
2425   AOT_assume <w ⊨ p>
2426   AOT_hence 1: <□w ⊨ p>
2427     by (metis "T◇" "≡E"(1) "rigid-truth-at:3" "→E")
2428   AOT_modally_strict {
2429     AOT_have <∀p (w ⊨ p ≡ p) → ((w ⊨ (φ → ψ)) → (w ⊨ φ → w ⊨ ψ))> for w φ ψ
2430     proof(safe intro!: "→I")
2431       AOT_assume <∀ p (w ⊨ p ≡ p)>
2432       AOT_hence <w ⊨ (φ → ψ) ≡ (φ → ψ)> and <w ⊨ φ ≡ φ> and <w ⊨ ψ ≡ ψ>
2433       using "∀E"(1)[rotated, OF "log-prop-prop:2"] by blast+
2434       moreover AOT_assume <w ⊨ (φ → ψ)>
2435       moreover AOT_assume <w ⊨ φ>
2436       ultimately AOT_show <w ⊨ ψ>
2437       by (metis "≡E"(1) "≡E"(2) "→E")
2438     qed
2439   }
2440 AOT_hence <◇∀p (w ⊨ p ≡ p) → ◇(w ⊨ (φ → ψ) → (w ⊨ φ → w ⊨ ψ))> for w φ ψ
2441   by (rule "RM◇")
2442 moreover AOT_have pos: <◇∀p (w ⊨ p ≡ p)> (491)
2443   using "world:1"[THEN "≡defE", OF PossibleWorld.ψ] "&E" by blast
2444 ultimately AOT_have <◇(w ⊨ (p → q) → (w ⊨ p → w ⊨ q))>
2445   using "→E" by blast
2446 AOT_hence <◇(w ⊨ p → w ⊨ q)>
2447   by (metis 0 "KBasic2:4" "≡E"(1) "→E")
2448 AOT_hence <◇w ⊨ q>
2449   by (metis 1 "KBasic2:4" "≡E"(1) "→E")
2450 AOT_thus <w ⊨ q>
2451   using "rigid-truth-at:2"[unvarify p, THEN "≡E"(1), OF "log-prop-prop:2"]
2452   "&E" "&I" by meson
2453 next
2454   AOT_assume <w ⊨ p → w ⊨ q>
2455   AOT_hence <¬(w ⊨ p) ∨ w ⊨ q>

```



```

2456   by (metis "VI"(1) "VI"(2) "reductio-aa:1" "→E")
2457 AOT_hence <w ⊨ ¬p ∨ w ⊨ q>
2458   by (metis "coherent:1" "VI"(1) "VI"(2) "VE"(2) "≡E"(2) "reductio-aa:1")
2459 AOT_hence 0: <□(w ⊨ ¬p ∨ w ⊨ q)>
2460   using "rigid-truth-at:1"[unvarify p, THEN "≡E"(1), OF "log-prop-prop:2"]
2461   by (metis "KBasic:15" "VI"(1) "VI"(2) "VE"(2) "reductio-aa:1" "→E")
2462 AOT_modally_strict {
2463   AOT_have <∀p (w ⊨ p ≡ p) → ((w ⊨ ¬φ ∨ w ⊨ ψ) → (w ⊨ (φ → ψ)))> for w φ ψ
2464   proof(safe intro!: "→I")
2465     AOT_assume <∀ p (w ⊨ p ≡ p)>
2466     moreover AOT_assume <w ⊨ ¬φ ∨ w ⊨ ψ>
2467     ultimately AOT_show <w ⊨ (φ → ψ)>
2468       by (metis "VE"(2) "→I" "≡E"(1) "≡E"(2) "log-prop-prop:2"
2469           "reductio-aa:1" "rule-ui:1")
2470   qed
2471 }
2472 AOT_hence <◇∀p (w ⊨ p ≡ p) → ◇((w ⊨ ¬φ ∨ w ⊨ ψ) → w ⊨ (φ → ψ))> for w φ ψ
2473   by (rule "RM◇")
2474 moreover AOT_have pos: <◇∀p (w ⊨ p ≡ p)>
2475   using "world:1"[THEN "≡dfE", OF PossibleWorld.ψ] "&E" by blast
2476 ultimately AOT_have <◇((w ⊨ ¬p ∨ w ⊨ q) → w ⊨ (p → q))>
2477   using "→E" by blast
2478 AOT_hence <◇(w ⊨ (p → q))>
2479   by (metis 0 "KBasic2:4" "≡E"(1) "→E")
2480 AOT_thus <w ⊨ (p → q)>
2481   using "rigid-truth-at:2"[unvarify p, THEN "≡E"(1), OF "log-prop-prop:2"]
2482   by blast
2483 qed
2484
2485 AOT_theorem "conj-dist-w:3": <w ⊨ (p ∨ q) ≡ ((w ⊨ p) ∨ (w ⊨ q))>
2486 proof(safe intro!: "≡I" "→I")
2487   AOT_assume <w ⊨ (p ∨ q)>
2488   AOT_hence 0: <□w ⊨ (p ∨ q)>
2489     using "rigid-truth-at:1"[unvarify p, THEN "≡E"(1), OF "log-prop-prop:2"]
2490     by blast
2491 AOT_modally_strict {
2492   AOT_have <∀p (w ⊨ p ≡ p) → ((w ⊨ (φ ∨ ψ)) → (w ⊨ φ ∨ w ⊨ ψ))> for w φ ψ
2493   proof(safe intro!: "→I")
2494     AOT_assume <∀ p (w ⊨ p ≡ p)>
2495     AOT_hence <w ⊨ (φ ∨ ψ) ≡ (φ ∨ ψ)> and <w ⊨ φ ≡ φ> and <w ⊨ ψ ≡ ψ>
2496       using "VE"(1)[rotated, OF "log-prop-prop:2"] by blast+
2497     moreover AOT_assume <w ⊨ (φ ∨ ψ)>
2498     ultimately AOT_show <w ⊨ φ ∨ w ⊨ ψ>
2499       by (metis "VI"(1) "VI"(2) "VE"(3) "≡E"(1) "≡E"(2) "reductio-aa:1")
2500   qed
2501 }
2502 AOT_hence <◇∀p (w ⊨ p ≡ p) → ◇(w ⊨ (φ ∨ ψ) → (w ⊨ φ ∨ w ⊨ ψ))> for w φ ψ
2503   by (rule "RM◇")
2504 moreover AOT_have pos: <◇∀p (w ⊨ p ≡ p)>
2505   using "world:1"[THEN "≡dfE", OF PossibleWorld.ψ] "&E" by blast
2506 ultimately AOT_have <◇(w ⊨ (p ∨ q) → (w ⊨ p ∨ w ⊨ q))> using "→E" by blast
2507 AOT_hence <◇(w ⊨ p ∨ w ⊨ q)>
2508   by (metis 0 "KBasic2:4" "≡E"(1) "vdash-properties:10")
2509 AOT_hence <◇w ⊨ p ∨ ◇w ⊨ q>
2510   using "KBasic2:2"[THEN "≡E"(1)] by blast
2511 AOT_thus <w ⊨ p ∨ w ⊨ q>
2512   using "rigid-truth-at:2"[unvarify p, THEN "≡E"(1), OF "log-prop-prop:2"]
2513   by (metis "VI"(1) "VI"(2) "VE"(2) "reductio-aa:1")
2514 next
2515 AOT_assume <w ⊨ p ∨ w ⊨ q>
2516 AOT_hence 0: <□(w ⊨ p ∨ w ⊨ q)>
2517   using "rigid-truth-at:1"[unvarify p, THEN "≡E"(1), OF "log-prop-prop:2"]
2518   by (metis "KBasic:15" "VI"(1) "VI"(2) "VE"(2) "reductio-aa:1" "→E")

```

```

2519 AOT_modally_strict {
2520   AOT_have <math>\forall p (w \models p \equiv p) \rightarrow ((w \models \varphi \vee w \models \psi) \rightarrow (w \models (\varphi \vee \psi)))> for w \varphi \psi
2521   proof(safe intro!: "\rightarrow I")
2522     AOT_assume <math>\forall p (w \models p \equiv p)>
2523     moreover AOT_assume <math>\langle w \models \varphi \vee w \models \psi \rangle
2524     ultimately AOT_show <math>\langle w \models (\varphi \vee \psi) \rangle
2525     by (metis "\forall I"(1) "\forall I"(2) "\vee E"(2) "\equiv E"(1) "\equiv E"(2)
2526         "log-prop-prop:2" "reductio-aa:1" "rule-ui:1")
2527   qed
2528 }
2529 AOT_hence <math>\langle \Diamond \forall p (w \models p \equiv p) \rightarrow \Diamond ((w \models \varphi \vee w \models \psi) \rightarrow w \models (\varphi \vee \psi)) \rangle for w \varphi \psi
2530   by (rule "RM\Diamond")
2531 moreover AOT_have pos: <math>\langle \Diamond \forall p (w \models p \equiv p) \rangle \tag{491}
2532   using "world:1"[THEN "\equiv_{df}E", OF PossibleWorld.\psi] "&E" by blast
2533 ultimately AOT_have <math>\langle \Diamond ((w \models p \vee w \models q) \rightarrow w \models (p \vee q)) \rangle
2534   using "\rightarrow E" by blast
2535 AOT_hence <math>\langle \Diamond (w \models (p \vee q)) \rangle
2536   by (metis 0 "KBasic2:4" "\equiv E"(1) "\rightarrow E")
2537 AOT_thus <math>\langle w \models (p \vee q) \rangle
2538   using "rigid-truth-at:2"[unvarify p, THEN "\equiv E"(1), OF "log-prop-prop:2"]
2539   by blast
2540 qed
2541
2542 AOT_theorem "conj-dist-w:4": <math>\langle w \models (p \equiv q) \equiv ((w \models p) \equiv (w \models q)) \rangle \tag{528.4}
2543 proof(rule "\equiv I"; rule "\rightarrow I")
2544   AOT_assume <math>\langle w \models (p \equiv q) \rangle
2545   AOT_hence 0: <math>\langle \Box w \models (p \equiv q) \rangle
2546     using "rigid-truth-at:1"[unvarify p, THEN "\equiv E"(1), OF "log-prop-prop:2"]
2547     by blast
2548   AOT_modally_strict {
2549     AOT_have <math>\langle \forall p (w \models p \equiv p) \rightarrow ((w \models (\varphi \equiv \psi)) \rightarrow (w \models \varphi \equiv w \models \psi)) \rangle for w \varphi \psi
2550     proof(safe intro!: "\rightarrow I")
2551       AOT_assume <math>\langle \forall p (w \models p \equiv p) \rangle
2552       AOT_hence <math>\langle w \models (\varphi \equiv \psi) \equiv (\varphi \equiv \psi) \rangle and <math>\langle w \models \varphi \equiv \varphi \rangle and <math>\langle w \models \psi \equiv \psi \rangle
2553         using "\vee E"(1)[rotated, OF "log-prop-prop:2"] by blast+
2554       moreover AOT_assume <math>\langle w \models (\varphi \equiv \psi) \rangle
2555       ultimately AOT_show <math>\langle w \models \varphi \equiv w \models \psi \rangle
2556       by (metis "\equiv E"(2) "\equiv E"(5) "Commutativity of \equiv")
2557     qed
2558   }
2559   AOT_hence <math>\langle \Diamond \forall p (w \models p \equiv p) \rightarrow \Diamond (w \models (\varphi \equiv \psi) \rightarrow (w \models \varphi \equiv w \models \psi)) \rangle for w \varphi \psi
2560     by (rule "RM\Diamond")
2561 moreover AOT_have pos: <math>\langle \Diamond \forall p (w \models p \equiv p) \rangle \tag{491}
2562   using "world:1"[THEN "\equiv_{df}E", OF PossibleWorld.\psi] "&E" by blast
2563 ultimately AOT_have <math>\langle \Diamond (w \models (p \equiv q) \rightarrow (w \models p \equiv w \models q)) \rangle
2564   using "\rightarrow E" by blast
2565 AOT_hence 1: <math>\langle \Diamond (w \models p \equiv w \models q) \rangle
2566   by (metis 0 "KBasic2:4" "\equiv E"(1) "vdash-properties:10")
2567 AOT_have <math>\langle \Diamond ((w \models p \rightarrow w \models q) \& (w \models q \rightarrow w \models p)) \rangle
2568   apply (AOT_subst <math>\langle (w \models p \rightarrow w \models q) \& (w \models q \rightarrow w \models p) \rangle \langle w \models p \equiv w \models q \rangle)
2569   apply (meson "\equiv_{df}E" "conventions:3" "\rightarrow I" "df-rules-formulas[4]" "\equiv I")
2570   by (fact 1)
2571 AOT_hence 2: <math>\langle \Diamond (w \models p \rightarrow w \models q) \& \Diamond (w \models q \rightarrow w \models p) \rangle
2572   by (metis "KBasic2:3" "vdash-properties:10")
2573 AOT_have <math>\langle \Diamond (\neg w \models p \vee w \models q) \rangle and <math>\langle \Diamond (\neg w \models q \vee w \models p) \rangle
2574   apply (AOT_subst (reverse) <math>\langle \neg w \models p \vee w \models q \rangle \langle w \models p \rightarrow w \models q \rangle)
2575   apply (simp add: "oth-class-taut:1:c")
2576   apply (fact 2[THEN "&E"(1)])
2577   apply (AOT_subst (reverse) <math>\langle \neg w \models q \vee w \models p \rangle \langle w \models q \rightarrow w \models p \rangle)
2578   apply (simp add: "oth-class-taut:1:c")
2579   by (fact 2[THEN "&E"(2)])
2580 AOT_hence <math>\langle \Diamond (\neg w \models p) \vee \Diamond w \models q \rangle and <math>\langle \Diamond \neg w \models q \vee \Diamond w \models p \rangle
2581   using "KBasic2:2" "\equiv E"(1) by blast+

```

```

2582 AOT_hence <¬□w ⊨ p ∨ ◇w ⊨ q> and <¬□w ⊨ q ∨ ◇w ⊨ p>
2583   by (metis "KBasic:11" "∀I"(1) "∀I"(2) "∀E"(2) "≡E"(2) "raa-cor:1")+
2584 AOT_thus <w ⊨ p ≡ w ⊨ q>
2585   using "rigid-truth-at:2"[unvarify p, THEN "≡E"(1), OF "log-prop-prop:2"]
2586   by (metis "¬¬I" "T◇" "∀E"(2) "→I" "≡I" "≡E"(1) "rigid-truth-at:3")
2587 next
2588 AOT_have <□PossibleWorld(w)>
2589   using "≡E"(1) "rigid-pw:1" PossibleWorld.ψ by blast
2590 moreover {
2591   fix p
2592   AOT_modally_strict {
2593     AOT_have <PossibleWorld(w) → (w ⊨ p → □w ⊨ p)>
2594     using "rigid-truth-at:1" "→I"
2595     by (metis "≡E"(1))
2596   }
2597   AOT_hence <□PossibleWorld(w) → □(w ⊨ p → □w ⊨ p)>
2598   by (rule RM)
2599 }
2600 ultimately AOT_have 1: <□(w ⊨ p → □w ⊨ p)> for p
2601   by (metis "→E")
2602 AOT_assume <w ⊨ p ≡ w ⊨ q>
2603 AOT_hence 0: <□(w ⊨ p ≡ w ⊨ q)>
2604   using "sc-eq-box-box:5"[THEN "→E", THEN "qml:2"[axiom_inst, THEN "→E"],
2605     THEN "→E", OF "&I"]
2606   by (metis "1")
2607 AOT_modally_strict {
2608   AOT_have <∀p (w ⊨ p ≡ p) → ((w ⊨ φ ≡ w ⊨ ψ) → (w ⊨ (φ ≡ ψ)))> for w φ ψ
2609   proof(safe intro!: "→I")
2610     AOT_assume <∀ p (w ⊨ p ≡ p)>
2611     moreover AOT_assume <w ⊨ φ ≡ w ⊨ ψ>
2612     ultimately AOT_show <w ⊨ (φ ≡ ψ)>
2613     by (metis "≡E"(2) "≡E"(6) "log-prop-prop:2" "rule-ui:1")
2614   qed
2615 }
2616 AOT_hence <◇∀p (w ⊨ p ≡ p) → ◇((w ⊨ φ ≡ w ⊨ ψ) → w ⊨ (φ ≡ ψ))> for w φ ψ
2617   by (rule "RM◇")
2618 moreover AOT_have pos: <◇∀p (w ⊨ p ≡ p)>
2619   using "world:1"[THEN "≡dfE", OF PossibleWorld.ψ] "&E" by blast
2620 ultimately AOT_have <◇((w ⊨ p ≡ w ⊨ q) → w ⊨ (p ≡ q))>
2621   using "→E" by blast
2622 AOT_hence <◇(w ⊨ (p ≡ q))>
2623   by (metis 0 "KBasic2:4" "≡E"(1) "→E")
2624 AOT_thus <w ⊨ (p ≡ q)>
2625   using "rigid-truth-at:2"[unvarify p, THEN "≡E"(1), OF "log-prop-prop:2"]
2626   by blast
2627 qed
2628
2629 AOT_theorem "conj-dist-w:5": <w ⊨ (∀α φ{α}) ≡ (∀ α (w ⊨ φ{α}))>
2630 proof(safe intro!: "≡I" "→I" GEN)
2631   AOT_assume <w ⊨ (∀α φ{α})>
2632   AOT_hence 0: <□w ⊨ (∀α φ{α})>
2633     using "rigid-truth-at:1"[unvarify p, THEN "≡E"(1), OF "log-prop-prop:2"]
2634     by blast
2635   AOT_modally_strict {
2636     AOT_have <∀p (w ⊨ p ≡ p) → ((w ⊨ (∀α φ{α})) → (∀α w ⊨ φ{α}))> for w
2637     proof(safe intro!: "→I" GEN)
2638       AOT_assume <∀p (w ⊨ p ≡ p)>
2639       moreover AOT_assume <w ⊨ (∀α φ{α})>
2640       ultimately AOT_show <w ⊨ φ{α}> for α
2641       by (metis "≡E"(1) "≡E"(2) "log-prop-prop:2" "rule-ui:1" "rule-ui:3")
2642     qed
2643   }
2644   AOT_hence <◇∀p (w ⊨ p ≡ p) → ◇(w ⊨ (∀α φ{α}) → (∀α w ⊨ φ{α}))> for w

```

```

2645   by (rule "RM◇")
2646   moreover AOT_have pos: <◇∀p (w ⊨ p ≡ p)> (491)
2647   using "world:1"[THEN "≡dfE", OF PossibleWorld.ψ] "&E" by blast
2648   ultimately AOT_have <◇(w ⊨ (∀α φ{α}) → (∀α w ⊨ φ{α}))> using "→E" by blast
2649   AOT_hence <◇(∀α w ⊨ φ{α})>
2650   by (metis 0 "KBasic2:4" "≡E"(1) "→E")
2651   AOT_hence <∀α ◇w ⊨ φ{α}>
2652   by (metis "Buridan◇" "→E")
2653   AOT_thus <w ⊨ φ{α}> for α
2654   using "rigid-truth-at:2"[unvarify p, THEN "≡E"(1), OF "log-prop-prop:2"]
2655   "∀E"(2) by blast
2656 next
2657   AOT_assume <∀α w ⊨ φ{α}>
2658   AOT_hence <w ⊨ φ{α}> for α using "∀E"(2) by blast
2659   AOT_hence <□w ⊨ φ{α}> for α
2660   using "rigid-truth-at:1"[unvarify p, THEN "≡E"(1), OF "log-prop-prop:2"]
2661   "&E" "&I" by blast
2662   AOT_hence <∀α □w ⊨ φ{α}> by (rule GEN)
2663   AOT_hence 0: <□∀α w ⊨ φ{α}> by (rule BF[THEN "→E"])
2664   AOT_modally_strict {
2665     AOT_have <∀p (w ⊨ p ≡ p) → ((∀α w ⊨ φ{α}) → (w ⊨ (∀α φ{α})))> for w
2666     proof(safe intro!: "→I")
2667       AOT_assume <∀ p (w ⊨ p ≡ p)>
2668       moreover AOT_assume <∀α w ⊨ φ{α}>
2669       ultimately AOT_show <w ⊨ (∀α φ{α})>
2670       by (metis "≡E"(1) "≡E"(2) "log-prop-prop:2" "rule-ui:1"
2671           "rule-ui:3" "universal-cor")
2672   qed
2673 }
2674   AOT_hence <◇∀p (w ⊨ p ≡ p) → ◇((∀α w ⊨ φ{α}) → w ⊨ (∀α φ{α}))> for w
2675   by (rule "RM◇")
2676   moreover AOT_have pos: <◇∀p (w ⊨ p ≡ p)> (491)
2677   using "world:1"[THEN "≡dfE", OF PossibleWorld.ψ] "&E" by blast
2678   ultimately AOT_have <◇((∀α w ⊨ φ{α}) → w ⊨ (∀α φ{α}))>
2679   using "→E" by blast
2680   AOT_hence <◇(w ⊨ (∀α φ{α}))>
2681   by (metis 0 "KBasic2:4" "≡E"(1) "→E")
2682   AOT_thus <w ⊨ (∀α φ{α})>
2683   using "rigid-truth-at:2"[unvarify p, THEN "≡E"(1), OF "log-prop-prop:2"]
2684   by blast
2685 qed
2686
2687 AOT_theorem "conj-dist-w:6": <w ⊨ (∃α φ{α}) ≡ (∃ α (w ⊨ φ{α}))> (528.6)
2688 proof(safe intro!: "≡I" "→I" GEN)
2689   AOT_assume <w ⊨ (∃α φ{α})>
2690   AOT_hence 0: <□w ⊨ (∃α φ{α})>
2691   using "rigid-truth-at:1"[unvarify p, THEN "≡E"(1), OF "log-prop-prop:2"]
2692   by blast
2693   AOT_modally_strict {
2694     AOT_have <∀p (w ⊨ p ≡ p) → ((w ⊨ (∃α φ{α})) → (∃α w ⊨ φ{α}))> for w
2695     proof(safe intro!: "→I" GEN)
2696       AOT_assume <∀p (w ⊨ p ≡ p)>
2697       moreover AOT_assume <w ⊨ (∃α φ{α})>
2698       ultimately AOT_show <∃ α (w ⊨ φ{α})>
2699       by (metis "≡E" "≡I"(2) "≡E"(1,2) "log-prop-prop:2" "rule-ui:1")
2700   qed
2701 }
2702   AOT_hence <◇∀p (w ⊨ p ≡ p) → ◇(w ⊨ (∃α φ{α}) → (∃α w ⊨ φ{α}))> for w
2703   by (rule "RM◇")
2704   moreover AOT_have pos: <◇∀p (w ⊨ p ≡ p)> (491)
2705   using "world:1"[THEN "≡dfE", OF PossibleWorld.ψ] "&E" by blast
2706   ultimately AOT_have <◇(w ⊨ (∃α φ{α}) → (∃α w ⊨ φ{α}))> using "→E" by blast
2707   AOT_hence <◇(∃α w ⊨ φ{α})>

```

```

2708   by (metis 0 "KBasic2:4" "≡E"(1) "→E")
2709 AOT_hence <∃α ◇w ⊢ φ{α}>
2710   by (metis "BF◇" "→E")
2711 then AOT_obtain α where <◇w ⊢ φ{α}>
2712   using "∃E"[rotated] by blast
2713 AOT_hence <w ⊢ φ{α}>
2714   using "rigid-truth-at:2"[unvarify p, THEN "≡E"(1), OF "log-prop-prop:2"] by blast
2715 AOT_thus <∃ α w ⊢ φ{α}> by (rule "∃I")
2716 next
2717 AOT_assume <∃α w ⊢ φ{α}>
2718 then AOT_obtain α where <w ⊢ φ{α}> using "∃E"[rotated] by blast
2719 AOT_hence <□w ⊢ φ{α}>
2720   using "rigid-truth-at:1"[unvarify p, THEN "≡E"(1), OF "log-prop-prop:2"]
2721     "&E" "&I" by blast
2722 AOT_hence <∃α □w ⊢ φ{α}>
2723   by (rule "∃I")
2724 AOT_hence 0: <□∃α w ⊢ φ{α}>
2725   by (metis Buridan "→E")
2726 AOT_modally_strict {
2727   AOT_have <∀p (w ⊢ p ≡ p) → ((∃α w ⊢ φ{α}) → (w ⊢ (∃α φ{α})))> for w
2728   proof (safe intro!: "→I")
2729     AOT_assume <∀ p (w ⊢ p ≡ p)>
2730     moreover AOT_assume <∃α w ⊢ φ{α}>
2731     then AOT_obtain α where <w ⊢ φ{α}>
2732       using "∃E"[rotated] by blast
2733     ultimately AOT_show <w ⊢ (∃α φ{α})>
2734       by (metis "∃I"(2) "≡E"(1,2) "log-prop-prop:2" "rule-ui:1")
2735   qed
2736 }
2737 AOT_hence <◇∀p (w ⊢ p ≡ p) → ◇((∃α w ⊢ φ{α}) → w ⊢ (∃α φ{α})))> for w
2738   by (rule "RM◇")
2739 moreover AOT_have pos: <◇∀p (w ⊢ p ≡ p)>
2740   using "world:1"[THEN "≡dfE", OF PossibleWorld.ψ] "&E" by blast
2741 ultimately AOT_have <◇((∃α w ⊢ φ{α}) → w ⊢ (∃α φ{α})))>
2742   using "→E" by blast
2743 AOT_hence <◇(w ⊢ (∃α φ{α})))>
2744   by (metis 0 "KBasic2:4" "≡E"(1) "→E")
2745 AOT_thus <w ⊢ (∃α φ{α})>
2746   using "rigid-truth-at:2"[unvarify p, THEN "≡E"(1), OF "log-prop-prop:2"]
2747   by blast
2748 qed
2749
2750 AOT_theorem "conj-dist-w:7": <(w ⊢ □p) → □w ⊢ p>
2751 proof (rule "→I")
2752   AOT_assume <w ⊢ □p>
2753   AOT_hence <∃w w ⊢ □p> by (rule "PossibleWorld.∃I")
2754   AOT_hence <◇□p> using "fund:1"[unvarify p, OF "log-prop-prop:2", THEN "≡E"(2)]
2755     by blast
2756   AOT_hence <□p>
2757     by (metis "5◇" "→E")
2758   AOT_hence 1: <□□p>
2759     by (metis "S5Basic:6" "≡E"(1))
2760   AOT_have <□∀w w ⊢ p>
2761     by (AOT_subst (reverse) <∀w w ⊢ p> <□p>)
2762     (auto simp add: "fund:2" 1)
2763   AOT_hence <∀w □w ⊢ p>
2764     using "fund-lem:5[world]"[THEN "→E"] by simp
2765   AOT_thus <□w ⊢ p>
2766     using "→E" "PossibleWorld.∀E" by fast
2767   qed
2768
2769 AOT_theorem "conj-dist-w:8": <∃w∃p((□w ⊢ p) & ¬w ⊢ □p)>
2770 proof -

```

(491)

(528.7)

(528.8)

```

2771 AOT_obtain r where A: r and <math>\Diamond \neg r</math>
2772   by (metis "&E"(1) "&E"(2) "≡dfE" "∃E" "cont-tf:1" "cont-tf-thm:1")
2773 AOT_hence B: <math>\neg \Box r</math>
2774   by (metis "KBasic:11" "≡E"(2))
2775 AOT_have <math>\Diamond r</math>
2776   using A "T $\Diamond$ "[THEN "→E"] by simp
2777 AOT_hence <math>\exists w w \models r</math>
2778   using "fund:1"[THEN "≡E"(1)] by blast
2779 then AOT_obtain w where w: <math>\langle w \models r \rangle</math>
2780   using "PossibleWorld.∃E"[rotated] by meson
2781 AOT_hence <math>\Box w \models r</math>
2782   by (metis "T $\Diamond$ " "≡E"(1) "rigid-truth-at:3" "vdash-properties:10")
2783 moreover AOT_have <math>\neg w \models \Box r</math>
2784 proof(rule "raa-cor:2")
2785   AOT_assume <math>\langle w \models \Box r \rangle</math>
2786   AOT_hence <math>\exists w w \models \Box r</math>
2787     by (rule "PossibleWorld.∃I")
2788   AOT_hence <math>\Box r</math>
2789     by (metis "≡E"(2) "nec-dia-w:1")
2790   AOT_thus <math>\langle \Box r \ \& \ \neg \Box r \rangle</math>
2791     using B "&I" by blast
2792 qed
2793 ultimately AOT_have <math>\langle \Box w \models r \ \& \ \neg w \models \Box r \rangle</math>
2794   by (rule "&I")
2795 AOT_hence <math>\langle \exists p (\Box w \models p \ \& \ \neg w \models \Box p) \rangle</math>
2796   by (rule "∃I")
2797 thus ?thesis
2798   by (rule "PossibleWorld.∃I")
2799 qed
2800
2801 AOT_theorem "conj-dist-w:9": <math>\langle (\Diamond w \models p) \rightarrow w \models \Diamond p \rangle</math> (528.9)
2802 proof(rule "→I"; rule "raa-cor:1")
2803   AOT_assume <math>\langle \Diamond w \models p \rangle</math>
2804   AOT_hence 0: <math>\langle w \models p \rangle</math>
2805     by (metis "≡E"(1) "rigid-truth-at:2")
2806   AOT_assume <math>\langle \neg w \models \Diamond p \rangle</math>
2807   AOT_hence 1: <math>\langle w \models \neg \Diamond p \rangle</math>
2808     using "coherent:1"[unvarify p, THEN "≡E"(2), OF "log-prop-prop:2"] by blast
2809   moreover AOT_have <math>\langle w \models (\neg \Diamond p \rightarrow \neg p) \rangle</math>
2810     using "T $\Diamond$ "[THEN "contraposition:1[1]", THEN RN]
2811       "fund:2"[unvarify p, OF "log-prop-prop:2", THEN "≡E"(1), THEN "∀E"(2),
2812         THEN "→E", rotated, OF PossibleWorld.ψ]
2813     by blast
2814   ultimately AOT_have <math>\langle w \models \neg p \rangle</math>
2815     using "conj-dist-w:2"[unvarify p q, OF "log-prop-prop:2", OF "log-prop-prop:2",
2816       THEN "≡E"(1), THEN "→E"]
2817     by blast
2818   AOT_hence <math>\langle w \models p \ \& \ w \models \neg p \rangle</math> using 0 "&I" by blast
2819   AOT_thus <math>\langle p \ \& \ \neg p \rangle</math>
2820     by (metis "coherent:1" "Conjunction Simplification"(1,2) "≡E"(4)
2821       "modus-tollens:1" "raa-cor:3")
2822 qed
2823
2824 AOT_theorem "conj-dist-w:10": <math>\langle \exists w \exists p ((w \models \Diamond p) \ \& \ \neg \Diamond w \models p) \rangle</math> (528.10)
2825 proof -
2826   AOT_obtain w where w: <math>\langle \forall p (w \models p \equiv p) \rangle</math>
2827     using "act-world:1" "PossibleWorld.∃E"[rotated] by meson
2828   AOT_obtain r where <math>\langle \neg r \rangle</math> and <math>\langle \Diamond r \rangle</math>
2829     using "cont-tf-thm:2" "cont-tf:2"[THEN "≡dfE"] "&E" "∃E"[rotated] by metis
2830   AOT_hence <math>\langle w \models \neg r \rangle</math> and 0: <math>\langle w \models \Diamond r \rangle</math>
2831     using w[THEN "∀E"(1), OF "log-prop-prop:2", THEN "≡E"(2)] by blast+
2832   AOT_hence <math>\langle \neg w \models r \rangle</math> using "coherent:1"[THEN "≡E"(1)] by blast
2833   AOT_hence <math>\langle \neg \Diamond w \models r \rangle</math> by (metis "≡E"(4) "rigid-truth-at:2")

```

```

2834 AOT_hence <w ⊨ ◇r & ¬◇w ⊨ r> using 0 "&I" by blast
2835 AOT_hence <∃p (w ⊨ ◇p & ¬◇w ⊨ p)> by (rule "∃I")
2836 thus ?thesis by (rule "PossibleWorld.∃I")
2837 qed
2838
2839 AOT_theorem "two-worlds-exist:1": <∃p(ContingentlyTrue(p)) → ∃w (¬Actual(w))> (530.1)
2840 proof(rule "→I")
2841 AOT_assume <∃p ContingentlyTrue(p)>
2842 then AOT_obtain p where <ContingentlyTrue(p)>
2843 using "∃E"[rotated] by blast
2844 AOT_hence p: <p & ◇¬p>
2845 by (metis "≡dfE" "cont-tf:1")
2846 AOT_hence <∃w w ⊨ ¬p>
2847 using "fund:1"[unvarify p, OF "log-prop-prop:2", THEN "≡E"(1)] "&E" by blast
2848 then AOT_obtain w where w: <w ⊨ ¬p>
2849 using "PossibleWorld.∃E"[rotated] by meson
2850 AOT_have <¬Actual(w)>
2851 proof(rule "raa-cor:2")
2852 AOT_assume <Actual(w)>
2853 AOT_hence <w ⊨ p>
2854 using p[THEN "&E"(1)] actual[THEN "≡dfE", THEN "&E"(2)]
2855 by (metis "log-prop-prop:2" "raa-cor:3" "rule-ui:1" "→E" w)
2856 moreover AOT_have <¬(w ⊨ p)>
2857 by (metis "coherent:1" "≡E"(4) "reductio-aa:2" w)
2858 ultimately AOT_show <w ⊨ p & ¬(w ⊨ p)>
2859 using "&I" by blast
2860 qed
2861 AOT_thus <∃w ¬Actual(w)>
2862 by (rule "PossibleWorld.∃I")
2863 qed
2864
2865
2866 AOT_theorem "two-worlds-exist:2": <∃p(ContingentlyFalse(p)) → ∃w (¬Actual(w))> (530.2)
2867 proof(rule "→I")
2868 AOT_assume <∃p ContingentlyFalse(p)>
2869 then AOT_obtain p where <ContingentlyFalse(p)>
2870 using "∃E"[rotated] by blast
2871 AOT_hence p: <¬p & ◇p>
2872 by (metis "≡dfE" "cont-tf:2")
2873 AOT_hence <∃w w ⊨ p>
2874 using "fund:1"[unvarify p, OF "log-prop-prop:2", THEN "≡E"(1)] "&E" by blast
2875 then AOT_obtain w where w: <w ⊨ p>
2876 using "PossibleWorld.∃E"[rotated] by meson
2877 moreover AOT_have <¬Actual(w)>
2878 proof(rule "raa-cor:2")
2879 AOT_assume <Actual(w)>
2880 AOT_hence <w ⊨ ¬p>
2881 using p[THEN "&E"(1)] actual[THEN "≡dfE", THEN "&E"(2)]
2882 by (metis "log-prop-prop:2" "raa-cor:3" "rule-ui:1" "→E" w)
2883 moreover AOT_have <¬(w ⊨ p)>
2884 using calculation by (metis "coherent:1" "≡E"(4) "reductio-aa:2")
2885 AOT_thus <w ⊨ p & ¬(w ⊨ p)>
2886 using "&I" w by metis
2887 qed
2888 AOT_thus <∃w ¬Actual(w)>
2889 by (rule "PossibleWorld.∃I")
2890 qed
2891
2892 AOT_theorem "two-worlds-exist:3": <∃w ¬Actual(w)> (530.3)
2893 using "cont-tf-thm:1" "two-worlds-exist:1" "→E" by blast
2894
2895 AOT_theorem "two-worlds-exist:4": <∃w∃w'(w ≠ w')> (530.4)
2896 proof -

```



```

2897 AOT_obtain w where w: <Actual(w)>
2898   using "act-world:2"[THEN "uniqueness:1"[THEN "≡dfE"],
2899     THEN "cqt-further:5"[THEN "→E"]]
2900   "PossibleWorld.∃E"[rotated] "&E"
2901   by blast
2902 moreover AOT_obtain w' where w': <¬Actual(w')>
2903   using "two-worlds-exist:3" "PossibleWorld.∃E"[rotated] by meson
2904 AOT_have <¬(w = w')>
2905 proof(rule "raa-cor:2")
2906   AOT_assume <w = w'>
2907   AOT_thus <p & ¬p> for p
2908     using w w' "&E" by (metis "rule=E" "raa-cor:3")
2909 qed
2910 AOT_hence <w ≠ w'>
2911   by (metis "≡dfI" "=-infix")
2912 AOT_hence <∃w' w ≠ w'>
2913   by (rule "PossibleWorld.∃I")
2914 thus ?thesis
2915   by (rule "PossibleWorld.∃I")
2916 qed
2917
2918 (* TODO: more theorems *)
2919
2920 AOT_theorem "w-rel:1": <[λx φ{x}]↓ → [λx w | = φ{x}]↓> (552.1)
2921 proof(rule "→I")
2922   AOT_assume <[λx φ{x}]↓>
2923   AOT_hence <□[λx φ{x}]↓>
2924     by (metis "exist-nec" "→E")
2925 moreover AOT_have
2926   <□[λx φ{x}]↓ → □∀x∀y(∀F([F]x ≡ [F]y) → ((w | = φ{x}) ≡ ( w | = φ{y})))>
2927 proof (rule RM; rule "→I"; rule GEN; rule GEN; rule "→I")
2928   AOT_modally_strict {
2929     fix x y
2930     AOT_assume <[λx φ{x}]↓>
2931     AOT_hence <∀x∀y (∀F ([F]x ≡ [F]y) → □(φ{x} ≡ φ{y}))>
2932       using "&E" "kirchner-thm-cor:1"[THEN "→E"] by blast
2933     AOT_hence <∀F ([F]x ≡ [F]y) → □(φ{x} ≡ φ{y})>
2934       using "∀E"(2) by blast
2935     moreover AOT_assume <∀F ([F]x ≡ [F]y)>
2936     ultimately AOT_have <□(φ{x} ≡ φ{y})>
2937       using "→E" by blast
2938     AOT_hence <∀w (w | = (φ{x} ≡ φ{y}))>
2939       using "fund:2"[unvarify p, OF "log-prop-prop:2", THEN "≡E"(1)] by blast
2940     AOT_hence <w | = (φ{x} ≡ φ{y})>
2941       using "∀E"(2) using PossibleWorld.ψ "→E" by blast
2942     AOT_thus <(w | = φ{x}) ≡ (w | = φ{y})>
2943       using "conj-dist-w:4"[unvarify p q, OF "log-prop-prop:2",
2944         OF "log-prop-prop:2", THEN "≡E"(1)] by blast
2945   }
2946 qed
2947 ultimately AOT_have <□∀x∀y(∀F([F]x ≡ [F]y) → ((w | = φ{x}) ≡ ( w | = φ{y})))>
2948   using "→E" by blast
2949 AOT_thus <[λx w | = φ{x}]↓>
2950   using "kirchner-thm:1"[THEN "≡E"(2)] by fast
2951 qed
2952
2953 AOT_theorem "w-rel:2": <[λx1...xn φ{x1...xn}]↓ → [λx1...xn w | = φ{x1...xn}]↓> (552.2)
2954 proof(rule "→I")
2955   AOT_assume <[λx1...xn φ{x1...xn}]↓>
2956   AOT_hence <□[λx1...xn φ{x1...xn}]↓>
2957     by (metis "exist-nec" "→E")
2958 moreover AOT_have <□[λx1...xn φ{x1...xn}]↓ → □∀x1...∀xn∀y1...∀yn(
2959   ∀F([F]x1...xn ≡ [F]y1...yn) → ((w | = φ{x1...xn}) ≡ ( w | = φ{y1...yn})))>

```



```

2960 proof (rule RM; rule "→I"; rule GEN; rule GEN; rule "→I")
2961   AOT_modally_strict {
2962     fix x1xn y1yn
2963     AOT_assume <[λx1...xn φ{x1...xn}]↓>
2964     AOT_hence <∀x1...∀xn∀y1...∀yn (
2965       ∀F ([F]x1...xn ≡ [F]y1...yn) → □(φ{x1...xn} ≡ φ{y1...yn}))>
2966       using "&E" "kirchner-thm-cor:2"[THEN "→E"] by blast
2967     AOT_hence <∀F ([F]x1...xn ≡ [F]y1...yn) → □(φ{x1...xn} ≡ φ{y1...yn}))>
2968       using "∀E"(2) by blast
2969     moreover AOT_assume <∀F ([F]x1...xn ≡ [F]y1...yn)>
2970     ultimately AOT_have <□(φ{x1...xn} ≡ φ{y1...yn})>
2971       using "→E" by blast
2972     AOT_hence <∀w (w ⊨ (φ{x1...xn} ≡ φ{y1...yn}))>
2973       using "fund:2"[unvarify p, OF "log-prop-prop:2", THEN "≡E"(1)] by blast
2974     AOT_hence <w ⊨ (φ{x1...xn} ≡ φ{y1...yn})>
2975       using "∀E"(2) using PossibleWorld.ψ "→E" by blast
2976     AOT_thus <(w ⊨ φ{x1...xn}) ≡ (w ⊨ φ{y1...yn})>
2977       using "conj-dist-w:4"[unvarify p q, OF "log-prop-prop:2",
2978         OF "log-prop-prop:2", THEN "≡E"(1)] by blast
2979   }
2980 qed
2981 ultimately AOT_have <□∀x1...∀xn∀y1...∀yn(
2982   ∀F([F]x1...xn ≡ [F]y1...yn) → ((w ⊨ φ{x1...xn}) ≡ (w ⊨ φ{y1...yn})))>
2983   using "→E" by blast
2984 AOT_thus <[λx1...xn w ⊨ φ{x1...xn}]↓>
2985   using "kirchner-thm:2"[THEN "≡E"(2)] by fast
2986 qed
2987
2988 AOT_theorem "w-rel:3": <[λx1...xn w ⊨ [F]x1...xn]↓> (552.3)
2989   by (rule "w-rel:2"[THEN "→E"]) "cqt:2[lambda]"
2990
2991 AOT_define WorldIndexedRelation :: <Π ⇒ τ ⇒ Π> (<_>)
2992   "w-index": <[F]w =df [λx1...xn w ⊨ [F]x1...xn]> (553)
2993
2994 AOT_define Rigid :: <τ ⇒ φ> (<Rigid'(_)>)
2995   "df-rigid-rel:1": (554.1)
2996 <Rigid(F) ≡df F↓ & □∀x1...∀xn([F]x1...xn → □[F]x1...xn)>
2997
2998 AOT_define Rigidifies :: <τ ⇒ τ ⇒ φ> (<Rigidifies'(_,&_)>)
2999   "df-rigid-rel:2": (554.2)
3000 <Rigidifies(F, G) ≡df Rigid(F) & ∀x1...∀xn([F]x1...xn ≡ [G]x1...xn)>
3001
3002 AOT_theorem "rigid-der:1": <[F]wx1...xn ≡ w ⊨ [F]x1...xn> (556.1)
3003   apply (rule "rule-id-df:2:b[2]"[where τ="λ (Π, κ). «[Π]κ»" and
3004     σ="λ(Π, κ). «[λx1...xn κ ⊨ [Π]x1...xn]»",
3005     simplified, OF "w-index"])
3006   apply (fact "w-rel:3")
3007   apply (rule "beta-C-meta"[THEN "→E"])
3008   by (fact "w-rel:3")
3009
3010 AOT_theorem "rigid-der:2": <Rigid([G]w)> (556.2)
3011 proof(safe intro!: "≡dfI"[OF "df-rigid-rel:1"] "&I")
3012   AOT_show <[G]w↓>
3013     by (rule "rule-id-df:2:b[2]"[where τ="λ (Π, κ). «[Π]κ»" and
3014       σ="λ(Π, κ). «[λx1...xn κ ⊨ [Π]x1...xn]»",
3015       simplified, OF "w-index"])
3016     (fact "w-rel:3")+
3017 next
3018 AOT_have <□∀x1...∀xn ([G]wx1...xn → □[G]wx1...xn)>
3019 proof(rule RN; safe intro!: "→I" GEN)
3020   AOT_modally_strict {
3021     AOT_have assms: <PossibleWorld(w)> using PossibleWorld.ψ.
3022     AOT_hence nec_pw_w: <□PossibleWorld(w)>

```

```

3023     using "≡E"(1) "rigid-pw:1" by blast
3024   fix x1xn
3025   AOT_assume <[[G]w]x1...xn>
3026   AOT_hence <[λx1...xn w ⊢ [G]x1...xn]x1...xn>
3027     using "rule-id-df:2:a[2]" [where τ="λ (Π, κ). «[[Π]κ»" and
3028       σ="λ(Π, κ). «[λx1...xn κ ⊢ [[Π]x1...xn]]»",
3029       simplified, OF "w-index", OF "w-rel:3"]
3030
3031   by fast
3032   AOT_hence <w ⊢ [G]x1...xn>
3033     by (metis "β→C"(1))
3034   AOT_hence <□w ⊢ [G]x1...xn>
3035     using "rigid-truth-at:1"[unvarify p, OF "log-prop-prop:2", THEN "≡E"(1)]
3036     by blast
3037   moreover AOT_have <□w ⊢ [G]x1...xn → □[λx1...xn w ⊢ [G]x1...xn]x1...xn>
3038     proof (rule RM; rule "→I")
3039       AOT_modally_strict {
3040         AOT_assume <w ⊢ [G]x1...xn>
3041         AOT_thus <[λx1...xn w ⊢ [G]x1...xn]x1...xn>
3042           by (auto intro!: "β←C"(1) simp: "w-rel:3" "cqt:2")
3043       }
3044     qed
3045     ultimately AOT_have 1: <□[λx1...xn w ⊢ [G]x1...xn]x1...xn>
3046       using "→E" by blast
3047   AOT_show <□[[G]w]x1...xn>
3048     by (rule "rule-id-df:2:b[2]" [where τ="λ (Π, κ). «[[Π]κ»" and
3049       σ="λ(Π, κ). «[λx1...xn κ ⊢ [[Π]x1...xn]]»",
3050       simplified, OF "w-index"])
3051     (auto simp: 1 "w-rel:3")
3052   }
3053   qed
3054   AOT_thus <□∀x1...xn ([[G]w]x1...xn → □[[G]w]x1...xn)>
3055     using "→E" by blast
3056   qed
3057   AOT_theorem "rigid-der:3": <∃F Rigidifies(F, G)> (556.3)
3058   proof -
3059     AOT_obtain w where w: <∀p (w ⊢ p ≡ p)>
3060       using "act-world:1" "PossibleWorld.∃E"[rotated] by meson
3061     show ?thesis
3062     proof (rule "∃I"(1)[where τ=«[[G]w»]])
3063       AOT_show <Rigidifies([G]w, [G])>
3064       proof (safe intro!: "≡dfI"[OF "df-rigid-rel:2"] "&I" GEN)
3065         AOT_show <Rigid([G]w)>
3066         using "rigid-der:2" by blast
3067       next
3068         fix x1xn
3069         AOT_have <[[G]w]x1...xn ≡ [λx1...xn w ⊢ [G]x1...xn]x1...xn>
3070         proof (rule "≡I"; rule "→I")
3071           AOT_assume <[[G]w]x1...xn>
3072           AOT_thus <[λx1...xn w ⊢ [G]x1...xn]x1...xn>
3073             by (rule "rule-id-df:2:a[2]"
3074               [where τ="λ (Π, κ). «[[Π]κ»" and
3075                 σ="λ(Π, κ). «[λx1...xn κ ⊢ [[Π]x1...xn]]»",
3076                 simplified, OF "w-index", OF "w-rel:3"])
3077           next
3078             AOT_assume <[λx1...xn w ⊢ [G]x1...xn]x1...xn>
3079             AOT_thus <[[G]w]x1...xn>
3080             by (rule "rule-id-df:2:b[2]"
3081               [where τ="λ (Π, κ). «[[Π]κ»" and
3082                 σ="λ(Π, κ). «[λx1...xn κ ⊢ [[Π]x1...xn]]»",
3083                 simplified, OF "w-index", OF "w-rel:3"])
3084           qed
3085         also AOT_have <... ≡ w ⊢ [G]x1...xn>

```

```

3086     by (rule "beta-C-meta"[THEN "→E"])
3087     (fact "w-rel:3")
3088     also AOT_have <... ≡ [G]x1...xn>
3089     using w[THEN "∀E"(1), OF "log-prop-prop:2"] by blast
3090     finally AOT_show <[[G]w]x1...xn ≡ [G]x1...xn>.
3091 qed
3092 next
3093 AOT_show <[G]w↓>
3094   by (rule "rule-id-df:2:b[2]"[where τ="λ (Π, κ). «[Π]κ»"
3095       and σ="λ(Π, κ). «[λx1...xn κ ⊨ [Π]x1...xn]»",
3096       simplified, OF "w-index"])
3097   (auto simp: "w-rel:3")
3098 qed
3099 qed
3100
3101 AOT_theorem "rigid-rel-thms:1": (557.1)
3102   <□(∀x1...∀xn ([F]x1...xn → □[F]x1...xn) ≡ ∀x1...∀xn (◇[F]x1...xn → □[F]x1...xn)>
3103 proof(safe intro!: "≡I" "→I" GEN)
3104   fix x1xn
3105   AOT_assume <□∀x1...∀xn ([F]x1...xn → □[F]x1...xn)>
3106   AOT_hence <∀x1...∀xn □([F]x1...xn → □[F]x1...xn)>
3107   by (metis "→E" GEN RM "cqt-orig:3")
3108   AOT_hence <□([F]x1...xn → □[F]x1...xn)>
3109   using "∀E"(2) by blast
3110   AOT_hence <◇[F]x1...xn → □[F]x1...xn>
3111   by (metis "≡E"(1) "sc-eq-box-box:1")
3112   moreover AOT_assume <◇[F]x1...xn>
3113   ultimately AOT_show <□[F]x1...xn>
3114   using "→E" by blast
3115 next
3116   AOT_assume <∀x1...∀xn (◇[F]x1...xn → □[F]x1...xn)>
3117   AOT_hence <◇[F]x1...xn → □[F]x1...xn> for x1xn
3118   using "∀E"(2) by blast
3119   AOT_hence <□([F]x1...xn → □[F]x1...xn)> for x1xn
3120   by (metis "≡E"(2) "sc-eq-box-box:1")
3121   AOT_hence 0: <∀x1...∀xn □([F]x1...xn → □[F]x1...xn)>
3122   by (rule GEN)
3123   AOT_thus <□(∀x1...∀xn ([F]x1...xn → □[F]x1...xn)>
3124   using "BF" "vdash-properties:10" by blast
3125 qed
3126
3127 AOT_theorem "rigid-rel-thms:2": (557.2)
3128   <□(∀x1...∀xn ([F]x1...xn → □[F]x1...xn) ≡ ∀x1...∀xn (□[F]x1...xn ∨ □¬[F]x1...xn)>
3129 proof(safe intro!: "≡I" "→I")
3130   AOT_assume <□(∀x1...∀xn ([F]x1...xn → □[F]x1...xn)>
3131   AOT_hence 0: <∀x1...∀xn □([F]x1...xn → □[F]x1...xn)>
3132   using CBF[THEN "→E"] by blast
3133   AOT_show <∀x1...∀xn (□[F]x1...xn ∨ □¬[F]x1...xn)>
3134 proof(rule GEN)
3135   fix x1xn
3136   AOT_have 1: <□([F]x1...xn → □[F]x1...xn)>
3137   using 0[THEN "∀E"(2)].
3138   AOT_hence 2: <◇[F]x1...xn → [F]x1...xn>
3139   using "B◇" "Hypothetical Syllogism" "K◇" "vdash-properties:10" by blast
3140   AOT_have <[F]x1...xn ∨ ¬[F]x1...xn>
3141   using "exc-mid".
3142   moreover {
3143     AOT_assume <[F]x1...xn>
3144     AOT_hence <□[F]x1...xn>
3145     using 1[THEN "qml:2"[axiom_inst, THEN "→E"], THEN "→E"] by blast
3146   }
3147   moreover {
3148     AOT_assume 3: <¬[F]x1...xn>

```

```

3149     AOT_have < $\Box \neg [F]_{x_1 \dots x_n}$ >
3150     proof(rule "raa-cor:1")
3151       AOT_assume < $\neg \Box \neg [F]_{x_1 \dots x_n}$ >
3152       AOT_hence < $\Diamond [F]_{x_1 \dots x_n}$ >
3153         by (AOT_subst_def "conventions:5")
3154       AOT_hence < $[F]_{x_1 \dots x_n}$ > using 2[THEN " $\rightarrow E$ "] by blast
3155       AOT_thus < $[F]_{x_1 \dots x_n} \ \& \ \neg [F]_{x_1 \dots x_n}$ >
3156         using 3 "&I" by blast
3157     qed
3158   }
3159   ultimately AOT_show < $\Box [F]_{x_1 \dots x_n} \vee \Box \neg [F]_{x_1 \dots x_n}$ >
3160     by (metis "\I"(1,2) "raa-cor:1")
3161   qed
3162 next
3163 AOT_assume 0: < $\forall x_1 \dots \forall x_n (\Box [F]_{x_1 \dots x_n} \vee \Box \neg [F]_{x_1 \dots x_n})$ >
3164 {
3165   fix  $x_1 x_n$ 
3166   AOT_have < $\Box [F]_{x_1 \dots x_n} \vee \Box \neg [F]_{x_1 \dots x_n}$ > using 0[THEN "\E"(2)] by blast
3167   moreover {
3168     AOT_assume < $\Box [F]_{x_1 \dots x_n}$ >
3169     AOT_hence < $\Box \Box [F]_{x_1 \dots x_n}$ >
3170       using "S5Basic:6"[THEN "\E"(1)] by blast
3171     AOT_hence < $\Box ([F]_{x_1 \dots x_n} \rightarrow \Box [F]_{x_1 \dots x_n})$ >
3172       using "KBasic:1"[THEN "\rightarrow E"] by blast
3173   }
3174   moreover {
3175     AOT_assume < $\Box \neg [F]_{x_1 \dots x_n}$ >
3176     AOT_hence < $\Box ([F]_{x_1 \dots x_n} \rightarrow \Box [F]_{x_1 \dots x_n})$ >
3177       using "KBasic:2"[THEN "\rightarrow E"] by blast
3178   }
3179   ultimately AOT_have < $\Box ([F]_{x_1 \dots x_n} \rightarrow \Box [F]_{x_1 \dots x_n})$ >
3180     using "con-dis-i-e:4:b" "raa-cor:1" by blast
3181 }
3182 AOT_hence < $\forall x_1 \dots \forall x_n \Box ([F]_{x_1 \dots x_n} \rightarrow \Box [F]_{x_1 \dots x_n})$ >
3183   by (rule GEN)
3184 AOT_thus < $\Box (\forall x_1 \dots \forall x_n ([F]_{x_1 \dots x_n} \rightarrow \Box [F]_{x_1 \dots x_n}))$ >
3185   using BF[THEN "\rightarrow E"] by fast
3186 qed
3187
3188 AOT_theorem "rigid-rel-thms:3": < $\text{Rigid}(F) \equiv \forall x_1 \dots \forall x_n (\Box [F]_{x_1 \dots x_n} \vee \Box \neg [F]_{x_1 \dots x_n})$ > (557.3)
3189   by (AOT_subst_thm "df-rigid-rel:1"[THEN "\equiv Df", THEN "\equiv S"(1), OF "cqt:2"(1)];
3190       AOT_subst_thm "rigid-rel-thms:2")
3191       (simp add: "oth-class-taut:3:a")
3192
3193 (*<*)
3194 end
3195 (*>*)
3196

```

A.12. Natural Numbers

```

1  (*<*)
2  theory AOT_NaturalNumbers
3    imports AOT_PossibleWorlds AOT_ExtendedRelationComprehension
4    abbrevs one-to-one = <1-1>
5           and onto = <onto>
6  begin
7  (*>*)
8
9  section<Natural Numbers>
10
11 AOT_define CorrelatesOneToOne :: < $\tau \Rightarrow \tau \Rightarrow \tau \Rightarrow \varphi$ > (<_ |: _ 1-1 $\longleftrightarrow$  _>)
12   "1-1-cor": <R |: F 1-1 $\longleftrightarrow$  G  $\equiv_{df}$  R $\downarrow$  & F $\downarrow$  & G $\downarrow$  &
13               $\forall x$  ([F]x  $\rightarrow$   $\exists!y$ ([G]y & [R]xy)) &
14               $\forall y$  ([G]y  $\rightarrow$   $\exists!x$ ([F]x & [R]xy))>
15
16 AOT_define MapsTo :: < $\tau \Rightarrow \tau \Rightarrow \tau \Rightarrow \varphi$ > (<_ |: _  $\longrightarrow$  _>)
17   "fFG:1": <R |: F  $\longrightarrow$  G  $\equiv_{df}$  R $\downarrow$  & F $\downarrow$  & G $\downarrow$  &  $\forall x$  ([F]x  $\rightarrow$   $\exists!y$ ([G]y & [R]xy))>
18
19 AOT_define MapsToOneToOne :: < $\tau \Rightarrow \tau \Rightarrow \tau \Rightarrow \varphi$ > (<_ |: _ 1-1 $\longrightarrow$  _>)
20   "fFG:2": <R |: F 1-1 $\longrightarrow$  G  $\equiv_{df}$ 
21             R |: F  $\longrightarrow$  G &  $\forall x \forall y \forall z$  (([F]x & [F]y & [G]z)  $\rightarrow$  ([R]xz & [R]yz  $\rightarrow$  x = y))>
22
23 AOT_define MapsOnto :: < $\tau \Rightarrow \tau \Rightarrow \tau \Rightarrow \varphi$ > (<_ |: _  $\longrightarrow_{onto}$  _>)
24   "fFG:3": <R |: F  $\longrightarrow_{onto}$  G  $\equiv_{df}$  R |: F  $\longrightarrow$  G &  $\forall y$  ([G]y  $\rightarrow$   $\exists x$ ([F]x & [R]xy))>
25
26 AOT_define MapsOneToOneOnto :: < $\tau \Rightarrow \tau \Rightarrow \tau \Rightarrow \varphi$ > (<_ |: _ 1-1 $\longrightarrow_{onto}$  _>)
27   "fFG:4": <R |: F 1-1 $\longrightarrow_{onto}$  G  $\equiv_{df}$  R |: F 1-1 $\longrightarrow$  G & R |: F  $\longrightarrow_{onto}$  G>
28
29 AOT_theorem "eq-1-1": <R |: F 1-1 $\longleftrightarrow$  G  $\equiv$  R |: F 1-1 $\longrightarrow_{onto}$  G>
30 proof(rule "≡I"; rule "→I")
31   AOT_assume <R |: F 1-1 $\longleftrightarrow$  G>
32   AOT_hence A: < $\forall x$  ([F]x  $\rightarrow$   $\exists!y$ ([G]y & [R]xy))>
33             and B: < $\forall y$  ([G]y  $\rightarrow$   $\exists!x$ ([F]x & [R]xy))>
34   using "≡dfE"[OF "1-1-cor"] "&E" by blast+
35   AOT_have C: <R |: F  $\longrightarrow$  G>
36   proof (rule "≡dfI"[OF "fFG:1"]; rule "&I")
37     AOT_show <R $\downarrow$  & F $\downarrow$  & G $\downarrow$ >
38     using "cqt:2[const_var]"[axiom_inst] "&I" by metis
39   next
40     AOT_show < $\forall x$  ([F]x  $\rightarrow$   $\exists!y$ ([G]y & [R]xy))> by (rule A)
41   qed
42   AOT_show <R |: F 1-1 $\longrightarrow_{onto}$  G>
43   proof (rule "≡dfI"[OF "fFG:4"]; rule "&I")
44     AOT_show <R |: F 1-1 $\longrightarrow$  G>
45     proof (rule "≡dfI"[OF "fFG:2"]; rule "&I")
46       AOT_show <R |: F  $\longrightarrow$  G> using C.
47     next
48       AOT_show < $\forall x \forall y \forall z$  ([F]x & [F]y & [G]z  $\rightarrow$  ([R]xz & [R]yz  $\rightarrow$  x = y))>
49     proof(rule GEN; rule GEN; rule GEN; rule "→I"; rule "→I")
50       fix x y z
51       AOT_assume 1: <[F]x & [F]y & [G]z>
52       moreover AOT_assume 2: <[R]xz & [R]yz>
53       ultimately AOT_have 3: < $\exists!x$  ([F]x & [R]xz)>
54       using B "&E" "∃E" "→E" by fast
55       AOT_show <x = y>
56       by (rule "uni-most"[THEN "→E", OF 3, THEN "∃E"(2)[where β=x],
57           THEN "∃E"(2)[where β=y], THEN "→E"])
58       (metis "&I" "&E" 1 2)
59     qed
60   qed
61 next

```

```

62   AOT_show <R | : F  $\longrightarrow_{\text{onto}}$  G>
63   proof (rule "≡dfI"[OF "fFG:3"]; rule "&I")
64     AOT_show <R | : F  $\longrightarrow$  G> using C.
65   next
66     AOT_show <∀y ([G]y  $\rightarrow$  ∃x ([F]x & [R]xy))>
67     proof(rule GEN; rule " $\rightarrow$ I")
68       fix y
69       AOT_assume <[G]y>
70       AOT_hence <∃!x ([F]x & [R]xy)>
71         using B[THEN "∀E"(2), THEN " $\rightarrow$ E"] by blast
72       AOT_hence <∃x ([F]x & [R]xy & ∀β (([F]β & [R]βy)  $\rightarrow$  β = x))>
73         using "uniqueness:1"[THEN "≡dfE"] by blast
74       then AOT_obtain x where <[F]x & [R]xy>
75         using "∃E"[rotated] "&E" by blast
76       AOT_thus <∃x ([F]x & [R]xy)> by (rule "∃I")
77     qed
78   qed
79   qed
80   next
81   AOT_assume <R | : F  $\xrightarrow{1-1}_{\text{onto}}$  G>
82   AOT_hence <R | : F  $\xrightarrow{1-1}$  G> and <R | : F  $\longrightarrow_{\text{onto}}$  G>
83     using "≡dfE"[OF "fFG:4"] "&E" by blast+
84   AOT_hence C: <R | : F  $\longrightarrow$  G>
85     and D: <∀x∀y∀z ([F]x & [F]y & [G]z  $\rightarrow$  ([R]xz & [R]yz  $\rightarrow$  x = y))>
86     and E: <∀y ([G]y  $\rightarrow$  ∃x ([F]x & [R]xy))>
87     using "≡dfE"[OF "fFG:2"] "≡dfE"[OF "fFG:3"] "&E" by blast+
88   AOT_show <R | : F  $\xrightarrow{1-1}$  G>
89   proof(rule "1-1-cor"[THEN "≡dfI"]; safe intro!: "&I" "cqt:2[const_var]"[axiom_inst])
90     AOT_show <∀x ([F]x  $\rightarrow$  ∃!y ([G]y & [R]xy))>
91     using "≡dfE"[OF "fFG:1", OF C] "&E" by blast
92   next
93     AOT_show <∀y ([G]y  $\rightarrow$  ∃!x ([F]x & [R]xy))>
94     proof (rule "GEN"; rule " $\rightarrow$ I")
95       fix y
96       AOT_assume 0: <[G]y>
97       AOT_hence <∃x ([F]x & [R]xy)>
98         using E "∀E" " $\rightarrow$ E" by fast
99       then AOT_obtain a where a_prop: <[F]a & [R]ay>
100         using "∃E"[rotated] by blast
101       moreover AOT_have <∀z ([F]z & [R]zy  $\rightarrow$  z = a)>
102       proof (rule GEN; rule " $\rightarrow$ I")
103         fix z
104         AOT_assume <[F]z & [R]zy>
105         AOT_thus <z = a>
106           using D[THEN "∀E"(2)[where β=z], THEN "∀E"(2)[where β=a],
107             THEN "∀E"(2)[where β=y], THEN " $\rightarrow$ E", THEN " $\rightarrow$ E"]
108           a_prop 0 "&E" "&I" by metis
109       qed
110     ultimately AOT_have <∃x ([F]x & [R]xy & ∀z ([F]z & [R]zy  $\rightarrow$  z = x))>
111     using "&I" "∃I"(2) by fast
112     AOT_thus <∃!x ([F]x & [R]xy)>
113     using "uniqueness:1"[THEN "≡dfI"] by fast
114   qed
115   qed
116   qed
117
118   text<We have already introduced the restricted type of Ordinary objects in the
119     Extended Relation Comprehension theory. However, make sure all variable names
120     are defined as expected (avoiding conflicts with situations
121     of possible world theory).>
122   AOT_register_variable_names
123     Ordinary: u v r t s
124

```

```

125 AOT_theorem "equi:1": <∃!u φ{u} ≡ ∃u (φ{u} & ∀v (φ{v} → v =E u))> (729.1)
126 proof(rule "≡I"; rule "→I")
127   AOT_assume <∃!u φ{u}>
128   AOT_hence <∃!x (0!x & φ{x})>.
129   AOT_hence <∃x (0!x & φ{x} & ∀β (0!β & φ{β} → β = x))>
130     using "uniqueness:1"[THEN "≡dfE"] by blast
131   then AOT_obtain x where x_prop: <0!x & φ{x} & ∀β (0!β & φ{β} → β = x)>
132     using "∃E"[rotated] by blast
133   {
134     fix β
135     AOT_assume beta_ord: <0!β>
136     moreover AOT_assume <φ{β}>
137     ultimately AOT_have <β = x>
138       using x_prop[THEN "&E"(2), THEN "∀E"(2)[where β=β]] "&I" "→E" by blast
139     AOT_hence <β =E x>
140       using "ord=E:1"[THEN "→E", OF "∀I"(1)[OF beta_ord],
141         THEN "qml:2"[axiom_inst, THEN "→E"],
142         THEN "≡E"(1)]
143     by blast
144   }
145   AOT_hence <(0!β → (φ{β} → β =E x))> for β
146     using "→I" by blast
147   AOT_hence <∀β(0!β → (φ{β} → β =E x))>
148     by (rule GEN)
149   AOT_hence <0!x & φ{x} & ∀y (0!y → (φ{y} → y =E x))>
150     using x_prop[THEN "&E"(1)] "&I" by blast
151   AOT_hence <0!x & (φ{x} & ∀y (0!y → (φ{y} → y =E x)))>
152     using "&E" "&I" by meson
153   AOT_thus <∃u (φ{u} & ∀v (φ{v} → v =E u))>
154     using "∃I" by fast
155 next
156   AOT_assume <∃u (φ{u} & ∀v (φ{v} → v =E u))>
157   AOT_hence <∃x (0!x & (φ{x} & ∀y (0!y → (φ{y} → y =E x))))>
158     by blast
159   then AOT_obtain x where x_prop: <0!x & (φ{x} & ∀y (0!y → (φ{y} → y =E x)))>
160     using "∃E"[rotated] by blast
161   AOT_have <∀y ([0!]y & φ{y} → y = x)>
162   proof(rule GEN; rule "→I")
163     fix y
164     AOT_assume <0!y & φ{y}>
165     AOT_hence <y =E x>
166       using x_prop[THEN "&E"(2), THEN "&E"(2), THEN "∀E"(2)[where β=y]]
167       "→E" "&E" by blast
168     AOT_thus <y = x>
169       using "ord=E:1"[THEN "→E", OF "∀I"(2)[OF x_prop[THEN "&E"(1)]],
170       THEN "qml:2"[axiom_inst, THEN "→E"], THEN "≡E"(2)] by blast
171   qed
172   AOT_hence <[0!]x & φ{x} & ∀y ([0!]y & φ{y} → y = x)>
173     using x_prop "&E" "&I" by meson
174   AOT_hence <∃x ([0!]x & φ{x} & ∀y ([0!]y & φ{y} → y = x))>
175     by (rule "∃I")
176   AOT_hence <∃!x (0!x & φ{x})>
177     by (rule "uniqueness:1"[THEN "≡dfI"])
178   AOT_thus <∃!u φ{u}>.
179 qed
180
181 AOT_define CorrelatesEOneToOne :: <τ ⇒ τ ⇒ φ> (<_ | : _ 1-1↔E _>)
182 "equi:2": <R | : F 1-1↔E G ≡df R↓ & F↓ & G↓ &
183           ∀u ([F]u → ∃!v([G]v & [R]uv)) &
184           ∀v ([G]v → ∃!u([F]u & [R]uv))> (729.2)
185
186 AOT_define EquinumerousE :: <τ ⇒ τ ⇒ φ> (infixl "≈E" 50)
187 "equi:3": <F ≈E G ≡df ∃R (R | : F 1-1↔E G)> (729.3)

```

```

188
189 text<Note: not explicitly in PLM.>
190 AOT_theorem eq_den_1: <math display="block">\Pi \downarrow \text{ if } \langle \Pi \approx_E \Pi' \rangle>
191 proof -
192   AOT_have <math display="block">\exists R (R \mid: \Pi \xrightarrow{1-1}_E \Pi') \rangle
193     using "equi:3"[THEN "≡dfE"] that by blast
194   then AOT_obtain R where <math display="block">\langle R \mid: \Pi \xrightarrow{1-1}_E \Pi' \rangle
195     using "∃E"[rotated] by blast
196   AOT_thus <math display="block">\Pi \downarrow \rangle
197     using "equi:2"[THEN "≡dfE"] "&E" by blast
198 qed
199
200 text<Note: not explicitly in PLM.>
201 AOT_theorem eq_den_2: <math display="block">\Pi' \downarrow \text{ if } \langle \Pi \approx_E \Pi' \rangle>
202 proof -
203   AOT_have <math display="block">\exists R (R \mid: \Pi \xrightarrow{1-1}_E \Pi') \rangle
204     using "equi:3"[THEN "≡dfE"] that by blast
205   then AOT_obtain R where <math display="block">\langle R \mid: \Pi \xrightarrow{1-1}_E \Pi' \rangle
206     using "∃E"[rotated] by blast
207   AOT_thus <math display="block">\Pi' \downarrow \rangle
208     using "equi:2"[THEN "≡dfE"] "&E" by blast+
209 qed
210
211 AOT_theorem "eq-part:1": <math display="block">\langle F \approx_E F \rangle (730.1)
212 proof (safe intro!: "&I" GEN "→I" "cqt:2[const_var]"[axiom_inst]
213   "≡dfI"[OF "equi:3"] "≡dfI"[OF "equi:2"] "∃I"(1))
214   fix x
215   AOT_assume 1: <math display="block">\langle 0!x \rangle
216   AOT_assume 2: <math display="block">\langle [F]x \rangle
217   AOT_show <math display="block">\langle \exists!v ([F]v \ \& \ x =_E v) \rangle
218   proof(rule "equi:1"[THEN "≡E"(2)];
219     rule "∃I"(2)[where β=x];
220     safe dest!: "&E"(2)
221     intro!: "&I" "→I" 1 2 Ordinary.GEN "ord=Eequiv:1"[THEN "→E", OF 1])
222     AOT_show <math display="block">\langle v =_E x \rangle \text{ if } \langle x =_E v \rangle \text{ for } v
223     by (metis that "ord=Eequiv:2"[THEN "→E"])
224   qed
225 next
226   fix y
227   AOT_assume 1: <math display="block">\langle 0!y \rangle
228   AOT_assume 2: <math display="block">\langle [F]y \rangle
229   AOT_show <math display="block">\langle \exists!u ([F]u \ \& \ u =_E y) \rangle
230     by (safe dest!: "&E"(2)
231       intro!: "equi:1"[THEN "≡E"(2)] "∃I"(2)[where β=y]
232         "&I" "→I" 1 2 GEN "ord=Eequiv:1"[THEN "→E", OF 1])
233   qed(auto simp: "=E[denotes]")
234
235
236 AOT_theorem "eq-part:2": <math display="block">\langle F \approx_E G \rightarrow G \approx_E F \rangle (730.2)
237 proof (rule "→I")
238   AOT_assume <math display="block">\langle F \approx_E G \rangle
239   AOT_hence <math display="block">\exists R R \mid: F \xrightarrow{1-1}_E G \rangle
240     using "equi:3"[THEN "≡dfE"] by blast
241   then AOT_obtain R where <math display="block">\langle R \mid: F \xrightarrow{1-1}_E G \rangle
242     using "∃E"[rotated] by blast
243   AOT_hence 0: <math display="block">\langle R \downarrow \ \& \ F \downarrow \ \& \ G \downarrow \ \& \ \forall u ([F]u \rightarrow \exists!v ([G]v \ \& \ [R]uv)) \ \& \ \forall v ([G]v \rightarrow \exists!u ([F]u \ \& \ [R]uv)) \rangle
244     using "equi:2"[THEN "≡dfE"] by blast
245
246   AOT_have <math display="block">\langle [\lambda xy [R]yx] \downarrow \ \& \ G \downarrow \ \& \ F \downarrow \ \& \ \forall u ([G]u \rightarrow \exists!v ([F]v \ \& \ [\lambda xy [R]yx]uv)) \ \& \ \forall v ([F]v \rightarrow \exists!u ([G]u \ \& \ [\lambda xy [R]yx]uv)) \rangle
247     proof (AOT_subst <math display="block">\langle [\lambda xy [R]yx]yx \rangle \langle [R]xy \rangle \text{ for: } x \ y;
248       (safe intro!: "&I" "cqt:2[const_var]"[axiom_inst] 0[THEN "&E"(2)]

```



```

251         O[THEN "&E"(1), THEN "&E"(2)]; "cqt:2[lambda]"?)
252 AOT_modally_strict {
253   AOT_have <[λxy [R]yx]xy> if <[R]yx> for y x
254   by (auto intro!: "β←C"(1) "cqt:2"
255       simp: "&I" "ex:1:a" prod_denotesI "rule-ui:3" that)
256   moreover AOT_have <[R]yx> if <[λxy [R]yx]xy> for y x
257   using "β→C"(1)[where φ="λ(x,y). _ (x,y)" and κ1κn="(_,_)"]
258   simplified, OF that, simplified].
259   ultimately AOT_show <[λxy [R]yx]αβ ≡ [R]βα> for α β
260   by (metis "deduction-theorem" "≡I")
261 }
262 qed
263 AOT_hence <[λxy [R]yx] | : G 1-1↔E F>
264 using "equi:2"[THEN "≡dfI"] by blast
265 AOT_hence <∃R R | : G 1-1↔E F>
266 by (rule "∃I"(1)) "cqt:2[lambda]"
267 AOT_thus <G ≈E F>
268 using "equi:3"[THEN "≡dfI"] by blast
269 qed
270
271 text<Note: not explicitly in PLM.>
272 AOT_theorem "eq-part:2[terms]": <Π ≈E Π' → Π' ≈E Π> (730.2)
273 using "eq-part:2"[unvarify F G] eq_den_1 eq_den_2 "→I" by meson
274 declare "eq-part:2[terms]"[THEN "→E", sym]
275
276 AOT_theorem "eq-part:3": <(F ≈E G & G ≈E H) → F ≈E H> (730.3)
277 proof (rule "→I")
278   AOT_assume <F ≈E G & G ≈E H>
279   then AOT_obtain R1 and R2 where
280     <R1 | : F 1-1↔E G>
281     and <R2 | : G 1-1↔E H>
282   using "equi:3"[THEN "≡dfE"] "&E" "∃E"[rotated] by metis
283   AOT_hence ∂: <∀u ([F]u → ∃!v([G]v & [R1]uv)) & ∀v ([G]v → ∃!u([F]u & [R1]uv))>
284     and ξ: <∀u ([G]u → ∃!v([H]v & [R2]uv)) & ∀v ([H]v → ∃!u([G]u & [R2]uv))>
285   using "equi:2"[THEN "≡dfE", THEN "&E"(2)]
286     "equi:2"[THEN "≡dfE", THEN "&E"(1), THEN "&E"(2)]
287     "&I" by blast+
288   AOT_have <∃R R = [λxy O!x & O!y & ∃v ([G]v & [R1]xv & [R2]vy)]>
289   by (rule "free-thms:3[lambda]") cqt_2_lambda_inst_prover
290   then AOT_obtain R where R_def: <R = [λxy O!x & O!y & ∃v ([G]v & [R1]xv & [R2]vy)]>
291   using "∃E"[rotated] by blast
292   AOT_have 1: <∃!v ([H]v & [R]uv)> if a: <[O!]u> and b: <[F]u> for u
293   proof (rule "≡E"(2)[OF "equi:1"])
294     AOT_obtain b where
295       b_prop: <[O!]b & ([G]b & [R1]ub & ∀v ([G]v & [R1]uv → v =E b))>
296     using ∂[THEN "&E"(1), THEN "∇E"(2), THEN "→E", THEN "→E",
297         OF a b, THEN "≡E"(1)[OF "equi:1"]]
298     "∃E"[rotated] by blast
299     AOT_obtain c where
300       c_prop: "[O!]c & ([H]c & [R2]bc & ∀v ([H]v & [R2]bv → v =E c))"
301     using ξ[THEN "&E"(1), THEN "∇E"(2)[where β=b], THEN "→E",
302         OF b_prop[THEN "&E"(1)], THEN "→E",
303         OF b_prop[THEN "&E"(2), THEN "&E"(1), THEN "&E"(1)],
304         THEN "≡E"(1)[OF "equi:1"]]
305     "∃E"[rotated] by blast
306     AOT_show <∃v ([H]v & [R]uv & ∀v' ([H]v' & [R]uv' → v' =E v))>
307   proof (safe intro!: "&I" GEN "→I" "∃I"(2)[where β=c])
308     AOT_show <O!c> using c_prop "&E" by blast
309   next
310     AOT_show <[H]c> using c_prop "&E" by blast
311   next
312     AOT_have 0: <[O!]u & [O!]c & ∃v ([G]v & [R1]uv & [R2]vc)>
313     by (safe intro!: "&I" a c_prop[THEN "&E"(1)] "∃I"(2)[where β=b]

```

```

314             b_prop[THEN "&E"(1)] b_prop[THEN "&E"(2), THEN "&E"(1)]
315             c_prop[THEN "&E"(2), THEN "&E"(1), THEN "&E"(2)]]
316   AOT_show <[R]uc>
317     by (auto intro: "rule=E"[rotated, OF R_def[symmetric]]
318         intro!: " $\beta \leftarrow C$ "(1) "cqt:2"
319         simp: "&I" "ex:1:a" prod_denotesI "rule-ui:3" 0)
320   next
321     fix x
322     AOT_assume ordx: <O!x>
323     AOT_assume <[H]x & [R]ux>
324     AOT_hence hx: <[H]x> and <[R]ux> using "&E" by blast+
325     AOT_hence <[ $\lambda xy$  O!x & O!y &  $\exists v$  ([G]v & [R1]xv & [R2]vy)]ux>
326       using "rule=E"[rotated, OF R_def] by fast
327     AOT_hence <O!u & O!x &  $\exists v$  ([G]v & [R1]uv & [R2]vx)>
328       by (rule " $\beta \rightarrow C$ "(1)[where  $\varphi = \lambda(\kappa, \kappa'). \_ \kappa \kappa'$ " and  $\kappa_1 \kappa_n = (\_, \_)$ ", simplified])
329     then AOT_obtain z where z_prop: <O!z & ([G]z & [R1]uz & [R2]zx)>
330       using "&E" " $\exists E$ "[rotated] by blast
331     AOT_hence <z =E b>
332       using b_prop[THEN "&E"(2), THEN "&E"(2), THEN " $\forall E$ "(2)[where  $\beta = z$ ]]
333       using "&E" " $\rightarrow E$ " by metis
334     AOT_hence <z = b>
335       by (metis "=E-simple:2"[THEN " $\rightarrow E$ "])
336     AOT_hence <[R2]bx>
337       using z_prop[THEN "&E"(2), THEN "&E"(2)] "rule=E" by fast
338     AOT_thus <x =E c>
339       using c_prop[THEN "&E"(2), THEN "&E"(2), THEN " $\forall E$ "(2)[where  $\beta = x$ ],
340                 THEN " $\rightarrow E$ ", THEN " $\rightarrow E$ ", OF ordx]
341       hx "&I" by blast
342   qed
343   qed
344   AOT_have 2: < $\exists ! u$  (([F]u & [R]uv))> if a: <O!v> and b: <[H]v> for v
345   proof (rule " $\equiv E$ "(2)[OF "equi:1"])
346     AOT_obtain b where
347       b_prop: <O!b & ([G]b & [R2]bv &  $\forall u$  ([G]u & [R2]uv  $\rightarrow$  u =E b))>
348       using  $\xi$ [THEN "&E"(2), THEN " $\forall E$ "(2), THEN " $\rightarrow E$ ", THEN " $\rightarrow E$ ",
349             OF a b, THEN " $\equiv E$ "(1)[OF "equi:1"]]
350       " $\exists E$ "[rotated] by blast
351     AOT_obtain c where
352       c_prop: "[O!c & ([F]c & [R1]cb &  $\forall v$  ([F]v & [R1]vb  $\rightarrow$  v =E c))"
353       using  $\vartheta$ [THEN "&E"(2), THEN " $\forall E$ "(2)[where  $\beta = b$ ], THEN " $\rightarrow E$ ",
354             OF b_prop[THEN "&E"(1)], THEN " $\rightarrow E$ ",
355             OF b_prop[THEN "&E"(2), THEN "&E"(1), THEN "&E"(1)],
356             THEN " $\equiv E$ "(1)[OF "equi:1"]]
357       " $\exists E$ "[rotated] by blast
358     AOT_show < $\exists u$  ([F]u & [R]uv &  $\forall v'$  ([F]v' & [R]v'v  $\rightarrow$  v' =E u))>
359     proof (safe intro!: "&I" GEN " $\rightarrow I$ " " $\exists I$ "(2)[where  $\beta = c$ ])
360       AOT_show <O!c> using c_prop "&E" by blast
361     next
362       AOT_show <[F]c> using c_prop "&E" by blast
363     next
364     AOT_have <O!c & O!v &  $\exists u$  ([G]u & [R1]cu & [R2]uv)>
365       by (safe intro!: "&I" a " $\exists I$ "(2)[where  $\beta = b$ ])
366       c_prop[THEN "&E"(1)] b_prop[THEN "&E"(1)]
367       b_prop[THEN "&E"(2), THEN "&E"(1), THEN "&E"(1)]
368       b_prop[THEN "&E"(2), THEN "&E"(1), THEN "&E"(2)]
369       c_prop[THEN "&E"(2), THEN "&E"(1), THEN "&E"(2)]]
370     AOT_thus <[R]cv>
371       by (auto intro: "rule=E"[rotated, OF R_def[symmetric]]
372           intro!: " $\beta \leftarrow C$ "(1) "cqt:2"
373           simp: "&I" "ex:1:a" prod_denotesI "rule-ui:3")
374   next
375     fix x
376     AOT_assume ordx: <O!x>

```

```

377   AOT_assume <[F]x & [R]xv>
378   AOT_hence hx: <[F]x> and <[R]xv> using "&E" by blast+
379   AOT_hence <[λxy 0!x & 0!y & ∃v ([G]v & [R1]xv & [R2]vy)]xv>
380     using "rule=E"[rotated, OF R_def] by fast
381   AOT_hence <0!x & 0!v & ∃u ([G]u & [R1]xu & [R2]uv)>
382     by (rule "β→C"(1)[where φ="λ(κ,κ'). _ κ κ'" and κ1κn="(_,_)", simplified])
383   then AOT_obtain z where z_prop: <0!z & ([G]z & [R1]xz & [R2]zv)>
384     using "&E" "∃E"[rotated] by blast
385   AOT_hence <z =E b>
386     using b_prop[THEN "&E"(2), THEN "&E"(2), THEN "∀E"(2)[where β=z]]
387     using "&E" "→E" "&I" by metis
388   AOT_hence <z = b>
389     by (metis "=E-simple:2"[THEN "→E"])
390   AOT_hence <[R1]xb>
391     using z_prop[THEN "&E"(2), THEN "&E"(1), THEN "&E"(2)] "rule=E" by fast
392   AOT_thus <x =E c>
393     using c_prop[THEN "&E"(2), THEN "&E"(2), THEN "∀E"(2)[where β=x],
394               THEN "→E", THEN "→E", OF ordx]
395     hx "&I" by blast
396   qed
397   qed
398   AOT_show <F ≈E H>
399     apply (rule "equi:3"[THEN "≡dfI"])
400     apply (rule "∃I"(2)[where β=R])
401     by (auto intro!: 1 2 "equi:2"[THEN "≡dfI"] "&I" "cqt:2[const_var]"[axiom_inst]
402       Ordinary.GEN "→I" Ordinary.ψ)
403   qed
404
405   text<Note: not explicitly in PLM.>
406   AOT_theorem "eq-part:3[terms]": <Π ≈E Π'> if <Π ≈E Π'> and <Π' ≈E Π'> (730.3)
407     using "eq-part:3"[unvarify F G H, THEN "→E"] eq_den_1 eq_den_2 "→I" "&I"
408     by (metis that(1) that(2))
409   declare "eq-part:3[terms]"[trans]
410
411   AOT_theorem "eq-part:4": <F ≈E G ≡ ∀H (H ≈E F ≡ H ≈E G)> (730.4)
412   proof(rule "≡I"; rule "→I")
413     AOT_assume 0: <F ≈E G>
414     AOT_hence 1: <G ≈E F> using "eq-part:2"[THEN "→E"] by blast
415     AOT_show <∀H (H ≈E F ≡ H ≈E G)>
416     proof (rule GEN; rule "≡I"; rule "→I")
417       AOT_show <H ≈E G> if <H ≈E F> for H using 0
418       by (meson "&I" "eq-part:3" that "vdash-properties:6")
419     next
420       AOT_show <H ≈E F> if <H ≈E G> for H using 1
421       by (metis "&I" "eq-part:3" that "vdash-properties:6")
422     qed
423   next
424     AOT_assume <∀H (H ≈E F ≡ H ≈E G)>
425     AOT_hence <F ≈E F ≡ F ≈E G> using "∀E" by blast
426     AOT_thus <F ≈E G> using "eq-part:1" "≡E" by blast
427   qed
428
429   AOT_define MapsE :: <τ ⇒ τ ⇒ τ ⇒ φ> ("_ | _ →E _")
430     "equi-rem:1": (731.1)
431     <R |: F →E G ≡df R↓ & F↓ & G↓ & ∀u ([F]u → ∃!v ([G]v & [R]uv))>
432
433   AOT_define MapsEOneToOne :: <τ ⇒ τ ⇒ τ ⇒ φ> ("_ | _ 1-1 →E _")
434     "equi-rem:2": (731.2)
435     <R |: F 1-1 →E G ≡df
436     R |: F →E G & ∀t∀u∀v (([F]t & [F]u & [G]v) → ([R]tv & [R]uv → t =E u))>
437
438   AOT_define MapsEOnto :: <τ ⇒ τ ⇒ τ ⇒ φ> ("_ | _ →ontoE _")
439     "equi-rem:3": (731.3)

```

```

440   <R | : F →ontoE G ≡df R | : F → E G & ∀v ([G]v → ∃u ([F]u & [R]uv))>
441
442 AOT_define MapsEOneToOneOnto :: <τ ⇒ τ ⇒ τ ⇒ φ> ("_ | : _ 1-1→ontoE _")
443 "equi-rem:4":
444   <R | : F 1-1→ontoE G ≡df R | : F 1-1→ E G & R | : F →ontoE G >
445
446 AOT_theorem "equi-rem-thm":
447   <R | : F 1-1↔E G ≡ R | : F 1-1→ontoE G >
448 proof -
449   AOT_have <R | : F 1-1↔E G ≡ R | : [λx O!x & [F]x] 1-1↔ [λx O!x & [G]x]>
450 proof (safe intro!: "≡I" "→I" "&I")
451   AOT_assume <R | : F 1-1↔E G >
452   AOT_hence <∀u ([F]u → ∃!v ([G]v & [R]uv))>
453     and <∀v ([G]v → ∃!u ([F]u & [R]uv))>
454     using "equi:2"[THEN "≡dfE"] "&E" by blast+
455   AOT_hence a: <([F]u → ∃!v ([G]v & [R]uv))>
456     and b: <([G]v → ∃!u ([F]u & [R]uv))> for u v
457     using "Ordinary.∇E" by fast+
458   AOT_have <([λx [O!]x & [F]x]x → ∃!y ([λx [O!]x & [G]x]y & [R]xy))> for x
459     apply (AOT_subst <[λx [O!]x & [F]x]x <[O!]x & [F]x>)
460     apply (rule "beta-C-meta"[THEN "→E"])
461     apply "cqt:2[lambda]"
462     apply (AOT_subst <[λx [O!]x & [G]x]x <[O!]x & [G]x> for: x)
463     apply (rule "beta-C-meta"[THEN "→E"])
464     apply "cqt:2[lambda]"
465     apply (AOT_subst <O!y & [G]y & [R]xy <O!y & ([G]y & [R]xy)> for: y)
466     apply (meson "≡E"(6) "Associativity of &" "oth-class-taut:3:a")
467     apply (rule "→I") apply (frule "&E"(1)) apply (drule "&E"(2))
468     by (fact a[unconstrain u, THEN "→E", THEN "→E", of x])
469   AOT_hence A: <∀x ([λx [O!]x & [F]x]x → ∃!y ([λx [O!]x & [G]x]y & [R]xy))>
470     by (rule GEN)
471   AOT_have <([λx [O!]x & [G]x]y → ∃!x ([λx [O!]x & [F]x]x & [R]xy))> for y
472     apply (AOT_subst <[λx [O!]x & [G]x]y <[O!]y & [G]y>)
473     apply (rule "beta-C-meta"[THEN "→E"])
474     apply "cqt:2[lambda]"
475     apply (AOT_subst <[λx [O!]x & [F]x]x <[O!]x & [F]x> for: x)
476     apply (rule "beta-C-meta"[THEN "→E"])
477     apply "cqt:2[lambda]"
478     apply (AOT_subst <O!x & [F]x & [R]xy <O!x & ([F]x & [R]xy)> for: x)
479     apply (meson "≡E"(6) "Associativity of &" "oth-class-taut:3:a")
480     apply (rule "→I") apply (frule "&E"(1)) apply (drule "&E"(2))
481     by (fact b[unconstrain v, THEN "→E", THEN "→E", of y])
482   AOT_hence B: <∀y ([λx [O!]x & [G]x]y → ∃!x ([λx [O!]x & [F]x]x & [R]xy))>
483     by (rule GEN)
484   AOT_show <R | : [λx [O!]x & [F]x] 1-1↔ [λx [O!]x & [G]x]>
485     by (safe intro!: "1-1-cor"[THEN "≡dfI"] "&I"
486         "cqt:2[const_var]"[axiom_inst] A B)
487     "cqt:2[lambda]"
488 next
489   AOT_assume <R | : [λx [O!]x & [F]x] 1-1↔ [λx [O!]x & [G]x]>
490   AOT_hence a: <([λx [O!]x & [F]x]x → ∃!y ([λx [O!]x & [G]x]y & [R]xy))> and
491     b: <([λx [O!]x & [G]x]y → ∃!x ([λx [O!]x & [F]x]x & [R]xy))> for x y
492     using "1-1-cor"[THEN "≡dfE"] "&E" "∇E"(2) by blast+
493   AOT_have <[F]u → ∃!v ([G]v & [R]uv)> for u
494   proof (safe intro!: "→I")
495     AOT_assume fu: <[F]u>
496     AOT_have 0: <[λx [O!]x & [F]x]u>
497       by (auto intro!: "β←C"(1) "cqt:2" "cqt:2[const_var]"[axiom_inst]
498           Ordinary.ψ fu "&I")
499     AOT_show <∃!v ([G]v & [R]uv)>
500       apply (AOT_subst <[O!]x & ([G]x & [R]ux)>
501           <([O!]x & [G]x) & [R]ux> for: x)
502       apply (simp add: "Associativity of &")

```

```

503     apply (AOT_subst (reverse) <[O!]x & [G]x>
504             <[λx [O!]x & [G]x]x> for: x)
505     apply (rule "beta-C-meta"[THEN "→E"])
506     apply "cqt:2[lambda]"
507     using a[THEN "→E", OF 0] by blast
508 qed
509 AOT_hence A: <∀u ([F]u → ∃!v ([G]v & [R]uv))>
510   by (rule Ordinary.GEN)
511 AOT_have <[G]v → ∃!u ([F]u & [R]uv)> for v
512 proof (safe intro!: "→I")
513   AOT_assume gu: <[G]v>
514   AOT_have 0: <[λx [O!]x & [G]x]v>
515     by (auto intro!: "β←C"(1) "cqt:2" "cqt:2[const_var]"[axiom_inst]
516         Ordinary.ψ gu "&I")
517   AOT_show <∃!u ([F]u & [R]uv)>
518     apply (AOT_subst <[O!]x & ([F]x & [R]xv)> <([O!]x & [F]x) & [R]xv> for: x)
519     apply (simp add: "Associativity of &")
520     apply (AOT_subst (reverse) <[O!]x & [F]x><[λx [O!]x & [F]x]x> for: x)
521     apply (rule "beta-C-meta"[THEN "→E"])
522     apply "cqt:2[lambda]"
523     using b[THEN "→E", OF 0] by blast
524 qed
525 AOT_hence B: <∀v ([G]v → ∃!u ([F]u & [R]uv))> by (rule Ordinary.GEN)
526 AOT_show <R | : F1-1↔E G>
527   by (safe intro!: "equi:2"[THEN "≡dfI"] "&I" A B "cqt:2[const_var]"[axiom_inst])
528 qed
529 also AOT_have <... ≡ R | : F1-1→ontoE G>
530 proof(safe intro!: "≡I" "→I" "&I")
531   AOT_assume <R | : [λx [O!]x & [F]x] 1-1↔ [λx [O!]x & [G]x>
532   AOT_hence a: <([λx [O!]x & [F]x]x → ∃!y ([λx [O!]x & [G]x]y & [R]xy))> and
533     b: <([λx [O!]x & [G]x]y → ∃!x ([λx [O!]x & [F]x]x & [R]xy))> for x y
534   using "1-1-cor"[THEN "≡dfE"] "&E" "∀E"(2) by blast+
535 AOT_show <R | : F1-1→ontoE G>
536 proof (safe intro!: "equi-rem:4"[THEN "≡dfI"] "&I" "equi-rem:3"[THEN "≡dfI"]
537     "equi-rem:2"[THEN "≡dfI"] "equi-rem:1"[THEN "≡dfI"]
538     "cqt:2[const_var]"[axiom_inst] Ordinary.GEN "→I")
539   fix u
540   AOT_assume fu: <[F]u>
541   AOT_have 0: <[λx [O!]x & [F]x]u>
542     by (auto intro!: "β←C"(1) "cqt:2" "cqt:2[const_var]"[axiom_inst]
543         Ordinary.ψ fu "&I")
544   AOT_hence 1: <∃!y ([λx [O!]x & [G]x]y & [R]uy)>
545     using a[THEN "→E"] by blast
546   AOT_show <∃!v ([G]v & [R]uv)>
547     apply (AOT_subst <[O!]x & ([G]x & [R]ux)> <([O!]x & [G]x) & [R]ux> for: x)
548     apply (simp add: "Associativity of &")
549     apply (AOT_subst (reverse) <[O!]x & [G]x> <[λx [O!]x & [G]x]x> for: x)
550     apply (rule "beta-C-meta"[THEN "→E"])
551     apply "cqt:2[lambda]"
552     by (fact 1)
553 next
554   fix t u v
555   AOT_assume <[F]t & [F]u & [G]v> and rtv_tuv: <[R]tv & [R]uv>
556   AOT_hence oft: <[λx [O!]x & [F]x]t> and
557     ofu: <[λx [O!]x & [F]x]u> and
558     ogv: <[λx [O!]x & [G]x]v>
559   by (auto intro!: "β←C"(1) "cqt:2" "&I"
560       simp: Ordinary.ψ dest: "&E")
561   AOT_hence <∃!x ([λx [O!]x & [F]x]x & [R]xv)>
562     using b[THEN "→E"] by blast
563   then AOT_obtain a where
564     a_prop: <[λx [O!]x & [F]x]a & [R]av &
565             ∀x (([λx [O!]x & [F]x]x & [R]xv) → x = a)>

```

```

566     using "uniqueness:1"[THEN "≡dfE"] "∃E"[rotated] by blast
567 AOT_hence ua: <u = a>
568     using ofu rtv_tuv[THEN "&E"(2)] "∀E"(2) "→E" "&I" "&E"(2) by blast
569 moreover AOT_have ta: <t = a>
570     using a_prop oft rtv_tuv[THEN "&E"(1)] "∀E"(2) "→E" "&I" "&E"(2) by blast
571 ultimately AOT_have <t = u> by (metis "rule=E" id_sym)
572 AOT_thus <t =E u>
573     using "rule=E" id_sym "ord=Eequiv:1" Ordinary.ψ ta ua "→E" by fast
574 next
575 fix u
576 AOT_assume <[F]u>
577 AOT_hence <[λx 0!x & [F]x]u>
578     by (auto intro!: "β←C"(1) "cqt:2" "&I"
579         simp: "cqt:2[const_var]"[axiom_inst] Ordinary.ψ)
580 AOT_hence <∃!y ([λx [0!]x & [G]x]y & [R]uy)>
581     using a[THEN "→E"] by blast
582 then AOT_obtain a where
583     a_prop: <[λx [0!]x & [G]x]a & [R]ua &
584             ∀x (([λx [0!]x & [G]x]x & [R]ux) → x = a)>
585     using "uniqueness:1"[THEN "≡dfE"] "∃E"[rotated] by blast
586 AOT_have <0!a & [G]a>
587     by (rule "β→C"(1)) (auto simp: a_prop[THEN "&E"(1), THEN "&E"(1)])
588 AOT_hence <0!a> and <[G]a> using "&E" by blast+
589 moreover AOT_have <∀v ([G]v & [R]uv → v =E a)>
590 proof(safe intro!: Ordinary.GEN "→I"; frule "&E"(1); drule "&E"(2))
591   fix v
592   AOT_assume <[G]v> and ruv: <[R]uv>
593   AOT_hence <[λx [0!]x & [G]x]v>
594     by (auto intro!: "β←C"(1) "cqt:2" "&I" simp: Ordinary.ψ)
595   AOT_hence <v = a>
596     using a_prop[THEN "&E"(2), THEN "∀E"(2), THEN "→E", OF "&I"] ruv by blast
597   AOT_thus <v =E a>
598     using "rule=E" "ord=Eequiv:1" Ordinary.ψ "→E" by fast
599 qed
600 ultimately AOT_have <0!a & ([G]a & [R]ua & ∀v' ([G]v' & [R]uv' → v' =E a))>
601     using "∃I" "&I" a_prop[THEN "&E"(1), THEN "&E"(2)] by simp
602 AOT_hence <∃v ([G]v & [R]uv & ∀v' ([G]v' & [R]uv' → v' =E v))>
603     by (rule "∃I")
604 AOT_thus <∃!v ([G]v & [R]uv)>
605     by (rule "equi:1"[THEN "≡E"(2)])
606 next
607 fix v
608 AOT_assume <[G]v>
609 AOT_hence <[λx 0!x & [G]x]v>
610     by (auto intro!: "β←C"(1) "cqt:2" "&I" Ordinary.ψ)
611 AOT_hence <∃!x ([λx [0!]x & [F]x]x & [R]xv)>
612     using b[THEN "→E"] by blast
613 then AOT_obtain a where
614     a_prop: <[λx [0!]x & [F]x]a & [R]av &
615             ∀y ([λx [0!]x & [F]x]y & [R]yv → y = a)>
616     using "uniqueness:1"[THEN "≡dfE", THEN "∃E"[rotated]] by blast
617 AOT_have <0!a & [F]a>
618     by (rule "β→C"(1)) (auto simp: a_prop[THEN "&E"(1), THEN "&E"(1)])
619 AOT_hence <0!a & ([F]a & [R]av)>
620     using a_prop[THEN "&E"(1), THEN "&E"(2)] "&E" "&I" by metis
621 AOT_thus <∃u ([F]u & [R]uv)>
622     by (rule "∃I")
623 qed
624 next
625 AOT_assume <R | : F1-1→ontoE G>
626 AOT_hence 1: <R | : F1-1→E G>
627     and 2: <R | : F→ontoE G>
628     using "equi-rem:4"[THEN "≡dfE"] "&E" by blast+

```

```

629 AOT_hence 3: <R |: F →E G>
630   and A: <∀t ∀u ∀v ([F]t & [F]u & [G]v → ([R]tv & [R]uv → t =E u))>
631   using "equi-rem:2"[THEN "≡dfE", OF 1] "&E" by blast+
632 AOT_hence B: <∀u ([F]u → ∃!v ([G]v & [R]uv))>
633   using "equi-rem:1"[THEN "≡dfE"] "&E" by blast
634 AOT_have C: <∀v ([G]v → ∃u ([F]u & [R]uv))>
635   using "equi-rem:3"[THEN "≡dfE", OF 2] "&E" by blast
636 AOT_show <R |: [λx [O!]x & [F]x] 1-1 ↔ [λx [O!]x & [G]x]>
637 proof (rule "1-1-cor"[THEN "≡dfI"];
638   safe intro!: "&I" "cqt:2" GEN "→I")
639   fix x
640   AOT_assume 1: <[λx [O!]x & [F]x]x>
641   AOT_have <O!x & [F]x>
642     by (rule "β→C"(1)) (auto simp: 1)
643   AOT_hence <∃!v ([G]v & [R]xv)>
644     using B[THEN "∀E"(2), THEN "→E", THEN "→E"] "&E" by blast
645   then AOT_obtain y where
646     y_prop: <O!y & ([G]y & [R]xy & ∀u ([G]u & [R]xu → u =E y))>
647     using "equi:1"[THEN "≡E"(1)] "∃E"[rotated] by fastforce
648   AOT_hence <[λx O!x & [G]x]y>
649     by (auto intro!: "β←C"(1) "cqt:2" "&I" dest: "&E")
650   moreover AOT_have <∀z ([λx O!x & [G]x]z & [R]xz → z = y)>
651   proof (safe intro!: GEN "→I"; frule "&E"(1); drule "&E"(2))
652     fix z
653     AOT_assume 1: <[λx [O!]x & [G]x]z>
654     AOT_have 2: <O!z & [G]z>
655       by (rule "β→C"(1)) (auto simp: 1)
656     moreover AOT_assume <[R]xz>
657     ultimately AOT_have <z =E y>
658       using y_prop[THEN "&E"(2), THEN "&E"(2), THEN "∀E"(2),
659         THEN "→E", THEN "→E", rotated, OF "&I"] "&E"
660     by blast
661     AOT_thus <z = y>
662       using 2[THEN "&E"(1)] by (metis "=E-simple:2" "→E")
663   qed
664   ultimately AOT_have <[λx O!x & [G]x]y & [R]xy &
665     ∀z ([λx O!x & [G]x]z & [R]xz → z = y)>
666     using y_prop[THEN "&E"(2), THEN "&E"(1), THEN "&E"(2)] "&I" by auto
667   AOT_hence <∃y ([λx O!x & [G]x]y & [R]xy &
668     ∀z ([λx O!x & [G]x]z & [R]xz → z = y))>
669     by (rule "∃I")
670   AOT_thus <∃!y ([λx [O!]x & [G]x]y & [R]xy)>
671     using "uniqueness:1"[THEN "≡dfI"] by fast
672 next
673   fix y
674   AOT_assume 1: <[λx [O!]x & [G]x]y>
675   AOT_have oy_gy: <O!y & [G]y>
676     by (rule "β→C"(1)) (auto simp: 1)
677   AOT_hence <∃u ([F]u & [R]uy)>
678     using C[THEN "∀E"(2), THEN "→E", THEN "→E"] "&E" by blast
679   then AOT_obtain x where x_prop: <O!x & ([F]x & [R]xy)>
680     using "∃E"[rotated] by blast
681   AOT_hence ofx: <[λx O!x & [F]x]x>
682     by (auto intro!: "β←C"(1) "cqt:2" "&I" dest: "&E")
683   AOT_have <∃α ([λx [O!]x & [F]x]α & [R]αy &
684     ∀β ([λx [O!]x & [F]x]β & [R]βy → β = α))>
685     proof (safe intro!: "∃I"(2)[where β=x] "&I" GEN "→I")
686       AOT_show <[λx O!x & [F]x]x> using ofx.
687     next
688       AOT_show <[R]xy> using x_prop[THEN "&E"(2), THEN "&E"(2)].
689     next
690       fix z
691       AOT_assume 1: <[λx [O!]x & [F]x]z & [R]zy>

```



```

692   AOT_have oz_fz: <O!z & [F]z>
693   by (rule "β→C"(1)) (auto simp: 1[THEN "&E"(1)])
694   AOT_have <z =E x>
695   using A[THEN "∀E"(2)[where β=z], THEN "→E", THEN "∀E"(2)[where β=x],
696     THEN "→E", THEN "∀E"(2)[where β=y], THEN "→E",
697     THEN "→E", THEN "→E", OF oz_fz[THEN "&E"(1)],
698     OF x_prop[THEN "&E"(1)], OF oy_gy[THEN "&E"(1)], OF "&I", OF "&I",
699     OF oz_fz[THEN "&E"(2)], OF x_prop[THEN "&E"(2), THEN "&E"(1)],
700     OF oy_gy[THEN "&E"(2)], OF "&I", OF 1[THEN "&E"(2)],
701     OF x_prop[THEN "&E"(2), THEN "&E"(2)]]].
702   AOT_thus <z =E x>
703   by (metis "=E-simple:2" "vdash-properties:10")
704   qed
705   AOT_thus <∃!x ([λx [O!]x & [F]x]x & [R]xy)>
706   by (rule "uniqueness:1"[THEN "≡dfI"])
707   qed
708   qed
709   finally show ?thesis.
710   qed
711
712   AOT_theorem "empty-approx:1": <(¬∃u [F]u & ¬∃v [H]v) → F ≈E H> (733.1)
713   proof(rule "→I"; frule "&E"(1); drule "&E"(2))
714     AOT_assume 0: <¬∃u [F]u> and 1: <¬∃v [H]v>
715     AOT_have <∀u ([F]u → ∃!v ([H]v & [R]uv))> for R
716     proof(rule Ordinary.GEN; rule "→I"; rule "raa-cor:1")
717       fix u
718       AOT_assume <[F]u>
719       AOT_hence <∃u [F]u> using "Ordinary.∃I" "&I" by fast
720       AOT_thus <∃u [F]u & ¬∃u [F]u> using "&I" 0 by blast
721       qed
722     moreover AOT_have <∀v ([H]v → ∃!u ([F]u & [R]uv))> for R
723     proof(rule Ordinary.GEN; rule "→I"; rule "raa-cor:1")
724       fix v
725       AOT_assume <[H]v>
726       AOT_hence <∃v [H]v> using "Ordinary.∃I" "&I" by fast
727       AOT_thus <∃v [H]v & ¬∃v [H]v> using 1 "&I" by blast
728       qed
729     ultimately AOT_have <R |: F1-1↔E H> for R
730     apply (safe intro!: "equi:2"[THEN "≡dfI"] "&I" GEN "cqt:2[const_var]"[axiom_inst])
731     using "∀E" by blast+
732     AOT_hence <∃R R |: F1-1↔E H> by (rule "∃I")
733     AOT_thus <F ≈E H>
734     by (rule "equi:3"[THEN "≡dfI"])
735   qed
736
737   AOT_theorem "empty-approx:2": <(∃u [F]u & ¬∃v [H]v) → ¬(F ≈E H)> (733.2)
738   proof(rule "→I"; frule "&E"(1); drule "&E"(2); rule "raa-cor:2")
739     AOT_assume 1: <∃u [F]u> and 2: <¬∃v [H]v>
740     AOT_obtain b where b_prop: <O!b & [F]b>
741     using 1 "∃E"[rotated] by blast
742     AOT_assume <F ≈E H>
743     AOT_hence <∃R R |: F1-1↔E H>
744     by (rule "equi:3"[THEN "≡dfE"])
745     then AOT_obtain R where <R |: F1-1↔E H>
746     using "∃E"[rotated] by blast
747     AOT_hence ϑ: <∀u ([F]u → ∃!v ([H]v & [R]uv))>
748     using "equi:2"[THEN "≡dfE"] "&E" by blast+
749     AOT_have <∃!v ([H]v & [R]bv)> for u
750     using ϑ[THEN "∀E"(2)[where β=b], THEN "→E", THEN "→E",
751       OF b_prop[THEN "&E"(1)], OF b_prop[THEN "&E"(2)]]].
752     AOT_hence <∃v ([H]v & [R]bv & ∀u ([H]u & [R]bu → u =E v))>
753     by (rule "equi:1"[THEN "≡E"(1)])
754     then AOT_obtain x where <O!x & ([H]x & [R]bx & ∀u ([H]u & [R]bu → u =E x))>

```



```

755   using "∃E"[rotated] by blast
756   AOT_hence <O!x & [H]x> using "&E" "&I" by blast
757   AOT_hence <∃v [H]v> by (rule "∃I")
758   AOT_thus <∃v [H]v & ¬∃v [H]v> using 2 "&I" by blast
759 qed
760
761
762 AOT_define FminusU :: <Π ⇒ τ ⇒ Π> ("_^-")
763   "F-u": <[F]-x =df [λz [F]z & z ≠E x]>
764
765 text<Note: not explicitly in PLM.>
766 AOT_theorem "F-u[den]": <[F]-x↓>
767   by (rule "=dfI"(1)[OF "F-u", where τ1τn="(_,_)"] , simplified]; "cqt:2[lambda]"
768 AOT_theorem "F-u[equiv]": <[[F]-x]y ≡ ([F]y & y ≠E x)>
769   by (auto intro: "F-u"[THEN "=dfI"(1), where τ1τn="(_,_)"] , simplified)
770       intro!: "cqt:2" "beta-C-cor:2"[THEN "→E", THEN "∀E"(2)])
771
772 AOT_theorem eqP': <F ≈E G & [F]u & [G]v → [F]-u ≈E [G]-v>
773 proof (rule "→I"; frule "&E"(2); drule "&E"(1); frule "&E"(2); drule "&E"(1))
774   AOT_assume <F ≈E G>
775   AOT_hence <∃R R | : F 1-1↔E G>
776     using "equi:3"[THEN "≡dfE"] by blast
777   then AOT_obtain R where R_prop: <R | : F 1-1↔E G>
778     using "∃E"[rotated] by blast
779   AOT_hence A: <∀u ([F]u → ∃!v ([G]v & [R]uv))>
780     and B: <∀v ([G]v → ∃!u ([F]u & [R]uv))>
781     using "equi:2"[THEN "≡dfE"] "&E" by blast+
782   AOT_have <R | : F 1-1→ontoE G>
783     using "equi-rem-thm"[THEN "≡E"(1), OF R_prop].
784   AOT_hence <R | : F 1-1→E G & R | : F →ontoE G>
785     using "equi-rem:4"[THEN "≡dfE"] by blast
786   AOT_hence C: <∀t∀u∀v (([F]t & [F]u & [G]v) → ([R]tv & [R]uv → t =E u))>
787     using "equi-rem:2"[THEN "≡dfE"] "&E" by blast
788   AOT_assume fu: <[F]u>
789   AOT_assume gv: <[G]v>
790   AOT_have <[λz [II]z & z ≠E κ]↓> for II κ
791     by "cqt:2[lambda]"
792   note II_minus_κI = "rule-id-df:2:b[2]"[
793     where τ=<(λ(II, κ). «[II]-κ»)>, simplified, OF "F-u", simplified, OF this]
794   and II_minus_κE = "rule-id-df:2:a[2]"[
795     where τ=<(λ(II, κ). «[II]-κ»)>, simplified, OF "F-u", simplified, OF this]
796   AOT_have II_minus_κ_den: <[II]-κ↓> for II κ
797     by (rule II_minus_κI) "cqt:2[lambda]"
798   {
799     fix R
800     AOT_assume R_prop: <R | : F 1-1↔E G>
801     AOT_hence A: <∀u ([F]u → ∃!v ([G]v & [R]uv))>
802       and B: <∀v ([G]v → ∃!u ([F]u & [R]uv))>
803       using "equi:2"[THEN "≡dfE"] "&E" by blast+
804     AOT_have <R | : F 1-1→ontoE G>
805       using "equi-rem-thm"[THEN "≡E"(1), OF R_prop].
806     AOT_hence <R | : F 1-1→E G & R | : F →ontoE G>
807       using "equi-rem:4"[THEN "≡dfE"] by blast
808     AOT_hence C: <∀t∀u∀v (([F]t & [F]u & [G]v) → ([R]tv & [R]uv → t =E u))>
809       using "equi-rem:2"[THEN "≡dfE"] "&E" by blast
810
811     AOT_assume Ruv: <[R]uv>
812     AOT_have <R | : [F]-u 1-1↔E [G]-v>
813     proof(safe intro!: "equi:2"[THEN "≡dfI"] "&I" "cqt:2[const_var]"[axiom_inst]
814       II_minus_κ_den Ordinary.GEN "→I")
815       fix u'
816       AOT_assume <[[F]-u]u'>
817       AOT_hence 0: <[λz [F]z & z ≠E u]u'>

```

```

818     using  $\Pi_{\text{minus\_}\kappa E}$  by fast
819 AOT_have 0: <[F]u' & u'  $\neq_E$  u>
820   by (rule " $\beta \rightarrow C$ "(1)[where  $\kappa_1 \kappa_n = \text{"AOT\_term\_of\_var (Ordinary.Rep u')"$ ]) (fact 0)
821 AOT_have < $\exists!v$  ([G]v & [R]u'v)>
822   using A[THEN "Ordinary. $\forall E$ "[where  $\alpha = u'$ ], THEN " $\rightarrow E$ ", OF 0[THEN "&E"(1)]]].
823 then AOT_obtain v' where
824   v'_prop: <[G]v' & [R]u'v' &  $\forall t$  ([G]t & [R]u't  $\rightarrow t =_E v'$ )>
825   using "equi:1"[THEN " $\equiv E$ "(1)] "Ordinary. $\exists E$ "[rotated] by fastforce
826
827 AOT_show < $\exists!v'$  ([[G] $^{-v}$ ]v' & [R]u'v')>
828 proof (safe intro!: "equi:1"[THEN " $\equiv E$ "(2)] "Ordinary. $\exists I$ "[where  $\beta = v'$ ]
829       "&I" Ordinary.GEN " $\rightarrow I$ ")
830   AOT_show <[[G] $^{-v}$ ]v'>
831   proof (rule  $\Pi_{\text{minus\_}\kappa I}$ ;
832         safe intro!: " $\beta \leftarrow C$ "(1) "cqt:2" "&I" "thm-neg=E"[THEN " $\equiv E$ "(2)])
833     AOT_show <[G]v'> using v'_prop "&E" by blast
834   next
835     AOT_show < $\neg v' =_E v$ >
836     proof (rule "raa-cor:2")
837       AOT_assume <v'  $=_E v$ >
838       AOT_hence <v' = v> by (metis "=E-simple:2" " $\rightarrow E$ ")
839       AOT_hence Ruv': <[R]uv'> using "rule=E" Ruv id_sym by fast
840       AOT_have <u'  $=_E u$ >
841         by (rule C[THEN "Ordinary. $\forall E$ ", THEN "Ordinary. $\forall E$ ",
842             THEN "Ordinary. $\forall E$ "[where  $\alpha = v'$ ], THEN " $\rightarrow E$ ", THEN " $\rightarrow E$ "])
843         (safe intro!: "&I" 0[THEN "&E"(1)] fu
844             v'_prop[THEN "&E"(1), THEN "&E"(1)]
845             Ruv' v'_prop[THEN "&E"(1), THEN "&E"(2)])
846       moreover AOT_have < $\neg(u' =_E u)$ >
847         using "0" "&E"(2) " $\equiv E$ "(1) "thm-neg=E" by blast
848       ultimately AOT_show <u'  $=_E u$  &  $\neg u' =_E u$ > using "&I" by blast
849     qed
850   qed
851   next
852     AOT_show <[R]u'v'> using v'_prop "&E" by blast
853   next
854     fix t
855     AOT_assume t_prop: <[[G] $^{-v}$ ]t & [R]u't>
856     AOT_have gt_t_noteq_v: <[G]t & t  $\neq_E v$ >
857     apply (rule " $\beta \rightarrow C$ "(1)[where  $\kappa_1 \kappa_n = \text{"AOT\_term\_of\_var (Ordinary.Rep t)"}$ ])
858     apply (rule  $\Pi_{\text{minus\_}\kappa E}$ )
859     by (fact t_prop[THEN "&E"(1)])
860     AOT_show <t  $=_E v'$ >
861     using v'_prop[THEN "&E"(2), THEN "Ordinary. $\forall E$ ", THEN " $\rightarrow E$ ",
862         OF "&I", OF gt_t_noteq_v[THEN "&E"(1)],
863         OF t_prop[THEN "&E"(2)]]].
864   qed
865   next
866     fix v'
867     AOT_assume G_minus_v_v': <[[G] $^{-v}$ ]v'>
868     AOT_have gt_t_noteq_v: <[G]v' & v'  $\neq_E v$ >
869     apply (rule " $\beta \rightarrow C$ "(1)[where  $\kappa_1 \kappa_n = \text{"AOT\_term\_of\_var (Ordinary.Rep v')"$ ])
870     apply (rule  $\Pi_{\text{minus\_}\kappa E}$ )
871     by (fact G_minus_v_v')
872     AOT_have < $\exists!u$ ([F]u & [R]uv')>
873     using B[THEN "Ordinary. $\forall E$ ", THEN " $\rightarrow E$ ", OF gt_t_noteq_v[THEN "&E"(1)]]].
874   then AOT_obtain u' where
875     u'_prop: <[F]u' & [R]u'v' &  $\forall t$  ([F]t & [R]tv'  $\rightarrow t =_E u'$ )>
876     using "equi:1"[THEN " $\equiv E$ "(1)] "Ordinary. $\exists E$ "[rotated] by fastforce
877   AOT_show < $\exists!u'$  ([[F] $^{-u}$ ]u' & [R]u'v')>
878   proof (safe intro!: "equi:1"[THEN " $\equiv E$ "(2)] "Ordinary. $\exists I$ "[where  $\beta = u'$ ] "&I"
879         u'_prop[THEN "&E"(1), THEN "&E"(2)] Ordinary.GEN " $\rightarrow I$ ")
880     AOT_show <[[F] $^{-u}$ ]u'>

```

```

881 proof (rule  $\Pi$ _minus_κI;
882   safe intro!: " $\beta \leftarrow C$ "(1) "cqt:2" "&I" "thm-neg=E"[THEN " $\equiv E$ "(2)]
883   u'_prop[THEN "&E"(1), THEN "&E"(1)]; rule "raa-cor:2")
884 AOT_assume u'_eq_u: <u' =E u>
885 AOT_hence <u' = u>
886   using "=E-simple:2" "vdash-properties:10" by blast
887 AOT_hence Ru'v: <[R]u'v> using "rule=E" Ruv id_sym by fast
888 AOT_have <v' ≠E v>
889   using "&E"(2) gt_t_noteq_v by blast
890 AOT_hence v'_noteq_v: <¬(v' =E v)> by (metis " $\equiv E$ "(1) "thm-neg=E")
891 AOT_have <∃u ([G]u & [R]u'u & ∀v ([G]v & [R]u'v → v =E u))>
892   using A[THEN "Ordinary.∀E", THEN "→E",
893     OF u'_prop[THEN "&E"(1), THEN "&E"(1)],
894     THEN "equi:1"[THEN " $\equiv E$ "(1)]]].
895 then AOT_obtain t where
896   t_prop: <[G]t & [R]u't & ∀v ([G]v & [R]u'v → v =E t)>
897   using "Ordinary.∃E"[rotated] by meson
898 AOT_have <v =E t> if <[G]v> and <[R]u'v> for v
899   using t_prop[THEN "&E"(2), THEN "Ordinary.∀E", THEN "→E",
900     OF "&I", OF that].
901 AOT_hence <v' =E t> and <v =E t>
902   by (auto simp: gt_t_noteq_v[THEN "&E"(1)] Ru'v gv
903     u'_prop[THEN "&E"(1), THEN "&E"(2)])
904 AOT_hence <v' =E v>
905   using "rule=E" "=E-simple:2" id_sym "→E" by fast
906 AOT_thus <v' =E v & ¬v' =E v>
907   using v'_noteq_v "&I" by blast
908 qed
909 next
910 fix t
911 AOT_assume 0: <[[F]-u]t & [R]tv'>
912 moreover AOT_have <[F]t & t ≠E u>
913   apply (rule " $\beta \rightarrow C$ "(1)[where κ1κn="AOT_term_of_var (Ordinary.Rep t)"])
914   apply (rule  $\Pi$ _minus_κE)
915   by (fact 0[THEN "&E"(1)])
916 ultimately AOT_show <t =E u'>
917   using u'_prop[THEN "&E"(2), THEN "Ordinary.∀E", THEN "→E", OF "&I"]
918   "&E" by blast
919 qed
920 qed
921 AOT_hence <∃R R |: [F]-u 1-1↔E [G]-v>
922   by (rule "∃I")
923 } note 1 = this
924 moreover {
925   AOT_assume not_Ruv: <¬[R]uv>
926   AOT_have <∃!v ([G]v & [R]uv)>
927     using A[THEN "Ordinary.∀E", THEN "→E", OF fu].
928 then AOT_obtain b where
929   b_prop: <0!b & ([G]b & [R]ub & ∀t([G]t & [R]ut → t =E b))>
930   using "equi:1"[THEN " $\equiv E$ "(1)] "∃E"[rotated] by fastforce
931 AOT_hence ob: <0!b> and gb: <[G]b> and Rub: <[R]ub>
932   using "&E" by blast+
933 AOT_have <0!t → ([G]t & [R]ut → t =E b)> for t
934   using b_prop "&E"(2) "∀E"(2) by blast
935 AOT_hence b_unique: <t =E b> if <0!t> and <[G]t> and <[R]ut> for t
936   by (metis Adjunction "modus-tollens:1" "reductio-aa:1" that)
937 AOT_have not_v_eq_b: <¬(v =E b)>
938 proof(rule "raa-cor:2")
939   AOT_assume <v =E b>
940   AOT_hence 0: <v = b>
941     by (metis "=E-simple:2" "→E")
942   AOT_have <[R]uv>
943     using b_prop[THEN "&E"(2), THEN "&E"(1), THEN "&E"(2)]

```

```

944         "rule=E"[rotated, OF 0[symmetric]] by fast
945     AOT_thus <[R]uv & ¬[R]uv>
946         using not_Ruv "&I" by blast
947     qed
948     AOT_have not_b_eq_v: <¬(b =E v)>
949         using "modus-tollens:1" not_v_eq_b "ord=Eequiv:2" by blast
950     AOT_have <∃!u ([F]u & [R]uv)>
951         using B[THEN "Ordinary.∀E", THEN "→E", OF gv].
952     then AOT_obtain a where
953     a_prop: <O!a & ([F]a & [R]av & ∀t([F]t & [R]tv → t =E a))>
954         using "equi:1"[THEN "≡E"(1)] "∃E"[rotated] by fastforce
955     AOT_hence Oa: <O!a> and fa: <[F]a> and Rav: <[R]av>
956         using "&E" by blast+
957     AOT_have <O!t → ([F]t & [R]tv → t =E a)> for t
958         using a_prop "&E" "∀E"(2) by blast
959     AOT_hence a_unique: <t =E a> if <O!t> and <[F]t> and <[R]tv> for t
960         by (metis Adjunction "modus-tollens:1" "reductio-aa:1" that)
961     AOT_have not_u_eq_a: <¬(u =E a)>
962     proof(rule "raa-cor:2")
963         AOT_assume <u =E a>
964         AOT_hence O: <u = a>
965             by (metis "=E-simple:2" "→E")
966         AOT_have <[R]uv>
967             using a_prop[THEN "&E"(2), THEN "&E"(1), THEN "&E"(2)]
968                 "rule=E"[rotated, OF 0[symmetric]] by fast
969         AOT_thus <[R]uv & ¬[R]uv>
970             using not_Ruv "&I" by blast
971     qed
972     AOT_have not_a_eq_u: <¬(a =E u)>
973         using "modus-tollens:1" not_u_eq_a "ord=Eequiv:2" by blast
974     let ?R = <<<[λu'v' (u' ≠E u & v' ≠E v & [R]u'v') ∨
975             (u' =E a & v' =E b) ∨
976             (u' =E u & v' =E v)]>>>
977     AOT_have <[<?R>]↓> by "cqt:2[lambda]"
978     AOT_hence <∃ β β = [<?R>]>
979         using "free-thms:1" "≡E"(1) by fast
980     then AOT_obtain R1 where R1_def: <R1 = [<?R>]>
981         using "∃E"[rotated] by blast
982     AOT_have Rxy1: <[R]xy> if <[R1]xy> and <x ≠E u> and <x ≠E a> for x y
983     proof -
984         AOT_have O: <[<?R>]xy>
985             by (rule "rule=E"[rotated, OF R1_def]) (fact that(1))
986         AOT_have <(x ≠E u & y ≠E v & [R]xy) ∨ (x =E a & y =E b) ∨ (x =E u & y =E v)>
987             using "β→C"(1)[OF O] by simp
988         AOT_hence <x ≠E u & y ≠E v & [R]xy> using that(2,3)
989             by (metis "∀E"(3) "Conjunction Simplification"(1) "≡E"(1)
990                 "modus-tollens:1" "thm-neg=E")
991         AOT_thus <[R]xy> using "&E" by blast+
992     qed
993     AOT_have Rxy2: <[R]xy> if <[R1]xy> and <y ≠E v> and <y ≠E b> for x y
994     proof -
995         AOT_have O: <[<?R>]xy>
996             by (rule "rule=E"[rotated, OF R1_def]) (fact that(1))
997         AOT_have <(x ≠E u & y ≠E v & [R]xy) ∨ (x =E a & y =E b) ∨ (x =E u & y =E v)>
998             using "β→C"(1)[OF O] by simp
999         AOT_hence <x ≠E u & y ≠E v & [R]xy>
1000             using that(2,3)
1001             by (metis "∀E"(3) "Conjunction Simplification"(2) "≡E"(1)
1002                 "modus-tollens:1" "thm-neg=E")
1003         AOT_thus <[R]xy> using "&E" by blast+
1004     qed
1005     AOT_have R1xy: <[R1]xy> if <[R]xy> and <x ≠E u> and <y ≠E v> for x y
1006         by (rule "rule=E"[rotated, OF R1_def[symmetric]])

```

```

1007     (auto intro!: " $\beta \leftarrow C$ "(1) "cqt:2"
1008         simp: "&I" "ex:1:a" prod_denotesI "rule-ui:3" that " $\forall I$ "(1))
1009 AOT_have R1ab: <[R1]ab>
1010   apply (rule "rule=E"[rotated, OF R1_def[symmetric]])
1011   apply (safe intro!: " $\beta \leftarrow C$ "(1) "cqt:2" prod_denotesI "&I")
1012   by (meson a_prop b_prop "&I" "&E"(1) " $\forall I$ "(1) " $\forall I$ "(2) "ord=Eequiv:1" " $\rightarrow E$ ")
1013 AOT_have R1uv: <[R1]uv>
1014   apply (rule "rule=E"[rotated, OF R1_def[symmetric]])
1015   apply (safe intro!: " $\beta \leftarrow C$ "(1) "cqt:2" prod_denotesI "&I")
1016   by (meson "&I" " $\forall I$ "(2) "ord=Eequiv:1" Ordinary. $\psi$  " $\rightarrow E$ ")
1017 moreover AOT_have <R1 | : F1-1 $\longleftrightarrow_E$  G>
1018 proof (safe intro!: "equi:2"[THEN " $\equiv_{df} I$ "] "&I" "cqt:2" Ordinary.GEN " $\rightarrow I$ ")
1019   fix u'
1020   AOT_assume fu': <[F]u'>
1021   {
1022     AOT_assume not_u'_eq_u: < $\neg(u' =_E u)$ > and not_u'_eq_a: < $\neg(u' =_E a)$ >
1023     AOT_hence u'_noteq_u: <u'  $\neq_E$  u> and u'_noteq_a: <u'  $\neq_E$  a>
1024     by (metis " $\equiv E$ "(2) "thm-neg=E")+
1025     AOT_have < $\exists! v$  ([G]v & [R]u'v)>
1026     using A[THEN "Ordinary. $\forall E$ ", THEN " $\rightarrow E$ ", OF fu'].
1027     AOT_hence < $\exists v$  ([G]v & [R]u'v &  $\forall t$  ([G]t & [R]u't  $\rightarrow t =_E v$ )>>
1028     using "equi:1"[THEN " $\equiv E$ "(1)] by simp
1029     then AOT_obtain v' where
1030       v'_prop: <[G]v' & [R]u'v' &  $\forall t$  ([G]t & [R]u't  $\rightarrow t =_E v'$ )>
1031       using "Ordinary. $\exists E$ "[rotated] by meson
1032     AOT_hence gv': <[G]v'> and Ru'v': <[R]u'v'>
1033     using "&E" by blast+
1034     AOT_have not_v'_eq_v: < $\neg v' =_E v$ >
1035     proof (rule "raa-cor:2")
1036       AOT_assume <v' =E v>
1037       AOT_hence <v' = v>
1038       by (metis "E-simple:2" " $\rightarrow E$ ")
1039       AOT_hence Ru'v: <[R]u'v>
1040       using "rule=E" Ru'v' by fast
1041       AOT_have <u' =E a>
1042       using a_unique[OF Ordinary. $\psi$ , OF fu', OF Ru'v].
1043       AOT_thus <u' =E a &  $\neg u' =_E a$ >
1044       using not_u'_eq_a "&I" by blast
1045     qed
1046     AOT_hence v'_noteq_v: <v'  $\neq_E$  v>
1047     using " $\equiv E$ "(2) "thm-neg=E" by blast
1048     AOT_have < $\forall t$  ([G]t & [R]u't  $\rightarrow t =_E v'$ )>
1049     using v'_prop "&E" by blast
1050     AOT_hence <[G]t & [R]u't  $\rightarrow t =_E v'$ > for t
1051     using "Ordinary. $\forall E$ " by meson
1052     AOT_hence v'_unique: <t =E v'> if <[G]t> and <[R]u't> for t
1053     by (metis "&I" that " $\rightarrow E$ ")
1054
1055     AOT_have <[G]v' & [R1]u'v' &  $\forall t$  ([G]t & [R1]u't  $\rightarrow t =_E v'$ )>
1056     proof (safe intro!: "&I" gv' R1xy Ru'v' u'_noteq_u u'_noteq_a " $\rightarrow I$ "
1057         Ordinary.GEN "thm-neg=E"[THEN " $\equiv E$ "(2)] not_v'_eq_v)
1058       fix t
1059       AOT_assume 1: <[G]t & [R1]u't>
1060       AOT_have <[R]u't>
1061       using Rxy1[OF 1[THEN "&E"(2)], OF u'_noteq_u, OF u'_noteq_a].
1062       AOT_thus <t =E v'>
1063       using v'_unique 1[THEN "&E"(1)] by blast
1064     qed
1065     AOT_hence < $\exists v$  ([G]v & [R1]u'v &  $\forall t$  ([G]t & [R1]u't  $\rightarrow t =_E v$ )>>
1066     by (rule "Ordinary. $\exists I$ ")
1067     AOT_hence < $\exists! v$  ([G]v & [R1]u'v)>
1068     by (rule "equi:1"[THEN " $\equiv E$ "(2)])
1069   }

```

```

1070 moreover {
1071   AOT_assume 0: <u' =E u>
1072   AOT_hence u'_eq_u: <u' = u>
1073     using "=E-simple:2" "→E" by blast
1074   AOT_have <∃!v ([G]v & [R1]u'v)>
1075   proof (safe intro!: "equi:1"[THEN "≡E"(2)] "Ordinary.∃I"[where β=v]
1076     "&I" Ordinary.GEN "→I" gv)
1077     AOT_show <[R1]u'v>
1078       apply (rule "rule=E"[rotated, OF R1_def[symmetric]])
1079       apply (safe intro!: "β←C"(1) "cqt:2" "&I" prod_denotesI)
1080       by (safe intro!: "∀I"(2) "&I" 0 "ord=Eequiv:1"[THEN "→E", OF Ordinary.ψ])
1081   next
1082   fix v'
1083   AOT_assume <[G]v' & [R1]u'v'>
1084   AOT_hence 0: <[R1]uv'>
1085     using "rule=E"[rotated, OF u'_eq_u] "&E"(2) by fast
1086   AOT_have 1: <[R]uv'>
1087     by (rule "rule=E"[rotated, OF R1_def]) (fact 0)
1088   AOT_have 2: <(u ≠E u & v' ≠E v & [R]uv') ∨
1089     (u =E a & v' =E b) ∨
1090     (u =E u & v' =E v)>
1091     using "β→C"(1)[OF 1] by simp
1092   AOT_have <¬u ≠E u>
1093     using "≡E"(4) "modus-tollens:1" "ord=Eequiv:1" Ordinary.ψ
1094     "reductio-aa:2" "thm-neg=E" by blast
1095   AOT_hence <¬((u ≠E u & v' ≠E v & [R]uv') ∨ (u =E a & v' =E b))>
1096     using not_u_eq_a
1097     by (metis "√E"(2) "Conjunction Simplification"(1)
1098       "modus-tollens:1" "reductio-aa:1")
1099   AOT_hence <(u =E u & v' =E v)>
1100     using 2 by (metis "√E"(2))
1101   AOT_thus <v' =E v>
1102     using "&E" by blast
1103   qed
1104 }
1105 moreover {
1106   AOT_assume 0: <u' =E a>
1107   AOT_hence u'_eq_a: <u' = a>
1108     using "=E-simple:2" "→E" by blast
1109   AOT_have <∃!v ([G]v & [R1]u'v)>
1110   proof (safe intro!: "equi:1"[THEN "≡E"(2)] "∃I"(2)[where β=b] "&I"
1111     Ordinary.GEN "→I" b_prop[THEN "&E"(1)]
1112     b_prop[THEN "&E"(2), THEN "&E"(1), THEN "&E"(1)])
1113     AOT_show <[R1]u'b>
1114       apply (rule "rule=E"[rotated, OF R1_def[symmetric]])
1115       apply (safe intro!: "β←C"(1) "cqt:2" "&I" prod_denotesI)
1116       apply (rule "∀I"(1); rule "∀I"(2); rule "&I")
1117       apply (fact 0)
1118       using b_prop "&E"(1) "ord=Eequiv:1" "→E" by blast
1119   next
1120   fix v'
1121   AOT_assume gv'_R1u'v': <[G]v' & [R1]u'v'>
1122   AOT_hence 0: <[R1]av'>
1123     using u'_eq_a by (meson "rule=E" "&E"(2))
1124   AOT_have 1: <[R]av'>
1125     by (rule "rule=E"[rotated, OF R1_def]) (fact 0)
1126   AOT_have <(a ≠E u & v' ≠E v & [R]av') ∨
1127     (a =E a & v' =E b) ∨
1128     (a =E u & v' =E v)>
1129     using "β→C"(1)[OF 1] by simp
1130   moreover {
1131     AOT_assume 0: <a ≠E u & v' ≠E v & [R]av'>
1132     AOT_have <∃!v ([G]v & [R]u'v)>

```

```

1133     using A[THEN "Ordinary.VE", THEN "→E", OF fu'].
1134 AOT_hence <∃!v ([G]v & [R]av)>
1135     using u'_eq_a "rule=E" by fast
1136 AOT_hence <∃v ([G]v & [R]av & ∀t ([G]t & [R]at → t =E v))>
1137     using "equi:1"[THEN "≡E"(1)] by fast
1138 then AOT_obtain s where
1139     s_prop: <[G]s & [R]as & ∀t ([G]t & [R]at → t =E s)>
1140     using "Ordinary.∃E"[rotated] by meson
1141 AOT_have <v' =E s>
1142     using s_prop[THEN "&E"(2), THEN "Ordinary.VE"]
1143     gv'_R1u'v'[THEN "&E"(1)] 0[THEN "&E"(2)]
1144     by (metis "&I" "vdash-properties:10")
1145 moreover AOT_have <v =E s>
1146     using s_prop[THEN "&E"(2), THEN "Ordinary.VE"] gv Rav
1147     by (metis "&I" "→E")
1148 ultimately AOT_have <v' =E v>
1149     by (metis "&I" "ord=Eequiv:2" "ord=Eequiv:3" "→E")
1150 moreover AOT_have <¬(v' =E v)>
1151     using 0[THEN "&E"(1), THEN "&E"(2)]
1152     by (metis "≡E"(1) "thm-neg=E")
1153 ultimately AOT_have <v' =E b>
1154     by (metis "raa-cor:3")
1155 }
1156 moreover {
1157     AOT_assume <a =E u & v' =E v>
1158     AOT_hence <v' =E b>
1159     by (metis "&E"(1) not_a_eq_u "reductio-aa:1")
1160 }
1161 ultimately AOT_show <v' =E b>
1162     by (metis "&E"(2) "VE"(3) "reductio-aa:1")
1163 qed
1164 }
1165 ultimately AOT_show <∃!v ([G]v & [R1]u'v)>
1166     by (metis "raa-cor:1")
1167 next
1168 fix v'
1169 AOT_assume gv': <[G]v'>
1170 {
1171     AOT_assume not_v'_eq_v: <¬(v' =E v)>
1172     and not_v'_eq_b: <¬(v' =E b)>
1173     AOT_hence v'_noteq_v: <v' ≠E v>
1174     and v'_noteq_b: <v' ≠E b>
1175     by (metis "≡E"(2) "thm-neg=E")+
1176     AOT_have <∃!u ([F]u & [R]uv')>
1177     using B[THEN "Ordinary.VE", THEN "→E", OF gv'].
1178     AOT_hence <∃u ([F]u & [R]uv' & ∀t ([F]t & [R]tv' → t =E u))>
1179     using "equi:1"[THEN "≡E"(1)] by simp
1180 then AOT_obtain u' where
1181     u'_prop: <[F]u' & [R]u'v' & ∀t ([F]t & [R]tv' → t =E u')>
1182     using "Ordinary.∃E"[rotated] by meson
1183     AOT_hence fu': <[F]u'> and Ru'v': <[R]u'v'>
1184     using "&E" by blast+
1185     AOT_have not_u'_eq_u: <¬u' =E u>
1186     proof (rule "raa-cor:2")
1187         AOT_assume <u' =E u>
1188         AOT_hence <u' = u>
1189         by (metis "=E-simple:2" "→E")
1190         AOT_hence Ruv': <[R]uv'>
1191         using "rule=E" Ru'v' by fast
1192         AOT_have <v' =E b>
1193         using b_unique[OF Ordinary.ψ, OF gv', OF Ruv'].
1194         AOT_thus <v' =E b & ¬v' =E b>
1195         using not_v'_eq_b "&I" by blast

```



```

1196 qed
1197 AOT_hence u'_noteq_u: <u' ≠E u>
1198   using "≡E"(2) "thm-neg=E" by blast
1199 AOT_have <∀t ([F]t & [R]tv' → t =E u')>
1200   using u'_prop "&E" by blast
1201 AOT_hence <[F]t & [R]tv' → t =E u'> for t
1202   using "Ordinary.VE" by meson
1203 AOT_hence u'_unique: <t =E u'> if <[F]t> and <[R]tv'> for t
1204   by (metis "&I" that "→E")
1205
1206 AOT_have <[F]u' & [R1]u'v' & ∀t ([F]t & [R1]tv' → t =E u')>
1207 proof (safe intro!: "&I" gv' R1xy Ru'v' u'_noteq_u Ordinary.GEN "→I"
1208   "thm-neg=E"[THEN "≡E"(2)] not_v'_eq_v fu')
1209   fix t
1210   AOT_assume 1: <[F]t & [R1]tv'>
1211   AOT_have <[R]tv'>
1212     using Rxy2[OF 1[THEN "&E"(2)], OF v'_noteq_v, OF v'_noteq_b].
1213   AOT_thus <t =E u'>
1214     using u'_unique 1[THEN "&E"(1)] by blast
1215 qed
1216 AOT_hence <∃u ([F]u & [R1]uv' & ∀t ([F]t & [R1]tv' → t =E u))>
1217   by (rule "Ordinary.∃I")
1218 AOT_hence <∃!u ([F]u & [R1]uv')>
1219   by (rule "equi:1"[THEN "≡E"(2)])
1220 }
1221 moreover {
1222   AOT_assume 0: <v' =E v>
1223   AOT_hence u'_eq_u: <v' = v>
1224     using "=E-simple:2" "→E" by blast
1225   AOT_have <∃!u ([F]u & [R1]uv')>
1226   proof (safe intro!: "equi:1"[THEN "≡E"(2)] "Ordinary.∃I"[where β=u]
1227     "&I" Ordinary.GEN "→I" fu)
1228     AOT_show <[R1]uv'>
1229       by (rule "rule=E"[rotated, OF R1_def[symmetric]])
1230       (safe intro!: "β←C"(1) "cqt:2" "&I" prod_denotesI Ordinary.ψ
1231         "∨I"(2) 0 "ord=Eequiv:1"[THEN "→E"])
1232   next
1233     fix u'
1234     AOT_assume <[F]u' & [R1]u'v'>
1235     AOT_hence 0: <[R1]u'v'>
1236       using "rule=E"[rotated, OF u'_eq_u] "&E"(2) by fast
1237     AOT_have 1: <[R]u'v'>
1238       by (rule "rule=E"[rotated, OF R1_def]) (fact 0)
1239     AOT_have 2: <(u' ≠E u & v ≠E v & [R]u'v) ∨
1240       (u' =E a & v =E b) ∨
1241       (u' =E u & v =E v)>
1242       using "β→C"(1)[OF 1, simplified] by simp
1243     AOT_have <¬v ≠E v>
1244       using "≡E"(4) "modus-tollens:1" "ord=Eequiv:1" Ordinary.ψ
1245       "reductio-aa:2" "thm-neg=E" by blast
1246     AOT_hence <¬((u' ≠E u & v ≠E v & [R]u'v) ∨ (u' =E a & v =E b))>
1247       by (metis "&E"(1) "&E"(2) "∨E"(3) not_v_eq_b "raa-cor:3")
1248     AOT_hence <(u' =E u & v =E v)>
1249       using 2 by (metis "∨E"(2))
1250     AOT_thus <u' =E u>
1251       using "&E" by blast
1252   qed
1253 }
1254 moreover {
1255   AOT_assume 0: <v' =E b>
1256   AOT_hence v'_eq_b: <v' = b>
1257     using "=E-simple:2" "→E" by blast
1258   AOT_have <∃!u ([F]u & [R1]uv')>

```



```

1259   proof (safe intro!: "equi:1"[THEN "≡E"(2)] "∃I"(2)[where β=a] "&I"
1260         Ordinary.GEN "→I" b_prop[THEN "&E"(1)] Oa fa
1261         b_prop[THEN "&E"(2), THEN "&E"(1), THEN "&E"(1)])
1262   AOT_show <[R1]av'>
1263     apply (rule "rule=E"[rotated, OF R1_def[symmetric]])
1264     apply (safe intro!: "β←C"(1) "cqt:2" "&I" prod_denotesI)
1265     apply (rule "∀I"(1); rule "∀I"(2); rule "&I")
1266     using Oa "ord=Eequiv:1" "→E" apply blast
1267     using "0" by blast
1268   next
1269   fix u'
1270   AOT_assume fu'_Riu'v': <[F]u' & [R1]u'v'>
1271   AOT_hence 0: <[R1]u'b>
1272     using v'_eq_b by (meson "rule=E" "&E"(2))
1273   AOT_have 1: <[<?R>]u'b>
1274     by (rule "rule=E"[rotated, OF R1_def]) (fact 0)
1275   AOT_have <(u' ≠E u & b ≠E v & [R]u'b) ∨
1276           (u' =E a & b =E b) ∨
1277           (u' =E u & b =E v)>
1278     using "β→C"(1)[OF 1, simplified] by simp
1279   moreover {
1280     AOT_assume 0: <u' ≠E u & b ≠E v & [R]u'b>
1281     AOT_have <∃!u ([F]u & [R]uv')>
1282       using B[THEN "Ordinary.∀E", THEN "→E", OF gv'].
1283     AOT_hence <∃!u ([F]u & [R]ub)>
1284       using v'_eq_b "rule=E" by fast
1285     AOT_hence <∃u ([F]u & [R]ub & ∀t ([F]t & [R]tb → t =E u))>
1286       using "equi:1"[THEN "≡E"(1)] by fast
1287     then AOT_obtain s where
1288       s_prop: <[F]s & [R]sb & ∀t ([F]t & [R]tb → t =E s)>
1289       using "Ordinary.∃E"[rotated] by meson
1290     AOT_have <u' =E s>
1291       using s_prop[THEN "&E"(2), THEN "Ordinary.∀E"]
1292       fu'_Riu'v'[THEN "&E"(1)] 0[THEN "&E"(2)]
1293       by (metis "&I" "→E")
1294     moreover AOT_have <u =E s>
1295       using s_prop[THEN "&E"(2), THEN "Ordinary.∀E"] fu Rub
1296       by (metis "&I" "→E")
1297     ultimately AOT_have <u' =E u>
1298       by (metis "&I" "ord=Eequiv:2" "ord=Eequiv:3" "→E")
1299     moreover AOT_have <¬(u' =E u)>
1300       using 0[THEN "&E"(1), THEN "&E"(1)] by (metis "≡E"(1) "thm-neg=E")
1301     ultimately AOT_have <u' =E a>
1302       by (metis "raa-cor:3")
1303   }
1304   moreover {
1305     AOT_assume <u' =E u & b =E v>
1306     AOT_hence <u' =E a>
1307       by (metis "&E"(2) not_b_eq_v "reductio-aa:1")
1308   }
1309   ultimately AOT_show <u' =E a>
1310     by (metis "&E"(1) "∀E"(3) "reductio-aa:1")
1311   qed
1312 }
1313 ultimately AOT_show <∃!u ([F]u & [R1]uv')>
1314   by (metis "raa-cor:1")
1315 qed
1316 ultimately AOT_have <∃R R |: [F]-u1-1↔E [G]-v>
1317   using 1 by blast
1318 }
1319 ultimately AOT_have <∃R R |: [F]-u1-1↔E [G]-v>
1320   using R_prop by (metis "reductio-aa:2")
1321 AOT_thus <[F]-u ≈E [G]-v>

```

```

1322   by (rule "equi:3"[THEN "≡dfI"])
1323 qed
1324
1325
1326 AOT_theorem "P'-eq": <[F]-u ≈E [G]-v & [F]u & [G]v → F ≈E G > (736)
1327 proof(safe intro!: "→I"; frule "&E"(1); drule "&E"(2);
1328   frule "&E"(1); drule "&E"(2))
1329 AOT_have <[λz [Π]z & z ≠E κ]↓> for Π κ by "cqt:2[lambda]"
1330 note Π_minus_κI = "rule-id-df:2:b[2]" [
1331   where τ=<(λ(Π, κ). «[Π]-κ»)>, simplified, OF "F-u", simplified, OF this]
1332 and Π_minus_κE = "rule-id-df:2:a[2]" [
1333   where τ=<(λ(Π, κ). «[Π]-κ»)>, simplified, OF "F-u", simplified, OF this]
1334 AOT_have Π_minus_κ_den: <[Π]-κ↓> for Π κ
1335   by (rule Π_minus_κI) "cqt:2[lambda]" +
1336
1337 AOT_have Π_minus_κE1: <[Π]κ' >
1338   and Π_minus_κE2: <κ' ≠E κ> if <[[Π]-κ]κ' > for Π κ κ'
1339 proof -
1340   AOT_have <[λz [Π]z & z ≠E κ]κ' >
1341     using Π_minus_κE that by fast
1342   AOT_hence <[Π]κ' & κ' ≠E κ >
1343     by (rule "β→C"(1))
1344   AOT_thus <[Π]κ' > and <κ' ≠E κ >
1345     using "&E" by blast+
1346 qed
1347 AOT_have Π_minus_κI': <[[Π]-κ]κ' > if <[Π]κ' > and <κ' ≠E κ > for Π κ κ'
1348 proof -
1349   AOT_have κ'_den: <κ'↓>
1350     by (metis "russell-axiom[exe,1].ψ_denotes_asm" that(1))
1351   AOT_have <[λz [Π]z & z ≠E κ]κ' >
1352     by (safe intro!: "β←C"(1) "cqt:2" κ'_den "&I" that)
1353   AOT_thus <[[Π]-κ]κ' >
1354     using Π_minus_κI by fast
1355 qed
1356
1357 AOT_assume Gv: <[G]v >
1358 AOT_assume Fu: <[F]u >
1359 AOT_assume <[F]-u ≈E [G]-v >
1360 AOT_hence <∃R R | : [F]-u 1-1↔E [G]-v >
1361   using "equi:3"[THEN "≡dfE"] by blast
1362 then AOT_obtain R where R_prop: <R | : [F]-u 1-1↔E [G]-v >
1363   using "∃E"[rotated] by blast
1364 AOT_hence Fact1: <∀r ([F]-u)r → ∃!s ([G]-v)s & [R]rs >>
1365   and Fact1': <∀s ([G]-v)s → ∃!r ([F]-u)r & [R]rs >>
1366   using "equi:2"[THEN "≡dfE"] "&E" by blast+
1367 AOT_have <R | : [F]-u 1-1→ontoE [G]-v >
1368   using "equi-rem-thm"[unvarify F G, OF Π_minus_κ_den, OF Π_minus_κ_den,
1369     THEN "≡E"(1), OF R_prop].
1370 AOT_hence <R | : [F]-u 1-1→E [G]-v & R | : [F]-u →ontoE [G]-v >
1371   using "equi-rem:4"[THEN "≡dfE"] by blast
1372 AOT_hence Fact2:
1373   <∀r∀s∀t (([F]-u)r & ([F]-u)s & ([G]-v)t → ([R]rt & [R]st → r =E s)) >
1374   using "equi-rem:2"[THEN "≡dfE"] "&E" by blast
1375
1376 let ?R = <<[λxy (([F]-u)x & ([G]-v)y & [R]xy) ∨ (x =E u & y =E v) >>
1377 AOT_have R_den: <<?R>↓> by "cqt:2[lambda]"
1378
1379 AOT_show <F ≈E G >
1380 proof(safe intro!: "equi:3"[THEN "≡dfI"] "∃I"(1)[where τ="?R"] R_den
1381   "equi:2"[THEN "≡dfI"] "&I" "cqt:2" Ordinary.GEN "→I")
1382   fix r
1383   AOT_assume Fr: <[F]r >
1384   {

```

```

1385 AOT_assume not_r_eq_u: <¬(r =E u)>
1386 AOT_hence r_noteq_u: <r ≠E u>
1387   using "≡E"(2) "thm-neg=E" by blast
1388 AOT_have <[[F]-ur>
1389   by(rule Π_minus_κI; safe intro!: "β←C"(1) "cqt:2" "&I" Fr r_noteq_u)
1390 AOT_hence <∃!s ([[G]-vs & [R]rs)>
1391   using Fact1[THEN "∀E"(2)] "→E" Ordinary.ψ by blast
1392 AOT_hence <∃s ([[G]-vs & [R]rs & ∀t ([[G]-vt & [R]rt → t =E s))>
1393   using "equi:1"[THEN "≡E"(1)] by simp
1394 then AOT_obtain s where s_prop: <[[G]-vs & [R]rs & ∀t ([[G]-vt & [R]rt → t =E s)>
1395   using "Ordinary.∃E"[rotated] by meson
1396 AOT_hence G_minus_v_s: <[[G]-vs> and Rrs: <[R]rs>
1397   using "&E" by blast+
1398 AOT_have s_unique: <t =E s> if <[[G]-vt> and <[R]rt> for t
1399   using s_prop[THEN "&E"(2), THEN "Ordinary.∀E", THEN "→E", OF "&I", OF that].
1400 AOT_have Gs: <[G]s>
1401   using Π_minus_κE1[OF G_minus_v_s].
1402 AOT_have s_noteq_v: <s ≠E v>
1403   using Π_minus_κE2[OF G_minus_v_s].
1404 AOT_have <∃s ([[G]s & [⟨?R⟩]rs & (∀t ([[G]t & [⟨?R⟩]rt → t =E s)))]>
1405 proof(safe intro!: "Ordinary.∃I"[where β=s] "&I" Gs Ordinary.GEN "→I")
1406   AOT_show <[⟨?R⟩]rs>
1407     by (auto intro!: "β←C"(1) "cqt:2" "&I" "∨I"(1) Π_minus_κI' Fr Gs
1408         s_noteq_v Rrs r_noteq_u
1409         simp: "&I" "ex:1:a" prod_denotesI "rule-ui:3")
1410 next
1411   fix t
1412   AOT_assume 0: <[G]t & [⟨?R⟩]rt>
1413   AOT_hence <([[F]-ur & [[G]-vt & [R]rt) ∨ (r =E u & t =E v)>
1414     using "β→C"(1)[OF 0[THEN "&E"(2)], simplified] by blast
1415   AOT_hence 1: <[[F]-ur & [[G]-vt & [R]rt>
1416     using not_r_eq_u by (metis "&E"(1) "∀E"(3) "reductio-aa:1")
1417   AOT_show <t =E s> using s_unique 1 "&E" by blast
1418 qed
1419 }
1420 moreover {
1421   AOT_assume r_eq_u: <r =E u>
1422   AOT_have <∃s ([[G]s & [⟨?R⟩]rs & (∀t ([[G]t & [⟨?R⟩]rt → t =E s)))]>
1423   proof(safe intro!: "Ordinary.∃I"[where β=v] "&I" Gv Ordinary.GEN "→I")
1424     AOT_show <[⟨?R⟩]rv>
1425       by (auto intro!: "β←C"(1) "cqt:2" "&I" "∨I"(2) Π_minus_κI' Fr r_eq_u
1426           "ord=Eequiv:1"[THEN "→E"] Ordinary.ψ
1427           simp: "&I" "ex:1:a" prod_denotesI "rule-ui:3")
1428 next
1429   fix t
1430   AOT_assume 0: <[G]t & [⟨?R⟩]rt>
1431   AOT_hence <([[F]-ur & [[G]-vt & [R]rt) ∨ (r =E u & t =E v)>
1432     using "β→C"(1)[OF 0[THEN "&E"(2)], simplified] by blast
1433   AOT_hence <r =E u & t =E v>
1434     using r_eq_u Π_minus_κE2
1435     by (metis "&E"(1) "∀E"(2) "≡E"(1) "reductio-aa:1" "thm-neg=E")
1436   AOT_thus <t =E v> using "&E" by blast
1437 qed
1438 }
1439 ultimately AOT_show <∃!s ([[G]s & [⟨?R⟩]rs)>
1440   using "reductio-aa:2" "equi:1"[THEN "≡E"(2)] by fast
1441 next
1442   fix s
1443   AOT_assume Gs: <[G]s>
1444
1445   {
1446     AOT_assume not_s_eq_v: <¬(s =E v)>
1447     AOT_hence s_noteq_v: <s ≠E v>

```

```

1448     using "≡E"(2) "thm-neg=E" by blast
1449 AOT_have <[[G]-vs>
1450   by (rule Π_minus_κI; auto intro!: "β←C"(1) "cqt:2" "&I" Gs s_noteq_v)
1451 AOT_hence <∃!r ([[F]-ur & [R]rs)>
1452   using Fact1'[THEN "Ordinary.∀E"] "→E" by blast
1453 AOT_hence <∃r ([[F]-ur & [R]rs & ∀t ([[F]-ut & [R]ts → t =E r))>
1454   using "equi:1"[THEN "≡E"(1)] by simp
1455 then AOT_obtain r where
1456   r_prop: <[[F]-ur & [R]rs & ∀t ([[F]-ut & [R]ts → t =E r)>
1457   using "Ordinary.∃E"[rotated] by meson
1458 AOT_hence F_minus_u_r: <[[F]-ur> and Rrs: <[R]rs>
1459   using "&E" by blast+
1460 AOT_have r_unique: <t =E r> if <[[F]-ut> and <[R]ts> for t
1461   using r_prop[THEN "&E"(2), THEN "Ordinary.∀E",
1462     THEN "→E", OF "&I", OF that].
1463 AOT_have Fr: <[F]r>
1464   using Π_minus_κE1[OF F_minus_u_r].
1465 AOT_have r_noteq_u: <r ≠E u>
1466   using Π_minus_κE2[OF F_minus_u_r].
1467 AOT_have <∃r ([F]r & [«?R»]rs & (∀t ([F]t & [«?R»]ts → t =E r)))>
1468 proof(safe intro!: "Ordinary.∃I"[where β=r] "&I" Fr Ordinary.GEN "→I")
1469   AOT_show <[«?R»]rs>
1470     by (auto intro!: "β←C"(1) "cqt:2" "&I" "∨I"(1) Π_minus_κI' Fr
1471       Gs s_noteq_v Rrs r_noteq_u
1472       simp: "&I" "ex:1:a" prod_denotesI "rule-ui:3")
1473 next
1474   fix t
1475   AOT_assume 0: <[F]t & [«?R»]ts>
1476   AOT_hence <([[F]-ut & [[G]-vs & [R]ts) ∨ (t =E u & s =E v)>
1477     using "β→C"(1)[OF 0[THEN "&E"(2)], simplified] by blast
1478   AOT_hence 1: <[[F]-ut & [[G]-vs & [R]ts>
1479     using not_s_eq_v by (metis "&E"(2) "∨E"(3) "reductio-aa:1")
1480   AOT_show <t =E r> using r_unique 1 "&E" by blast
1481 qed
1482 }
1483 moreover {
1484   AOT_assume s_eq_v: <s =E v>
1485   AOT_have <∃r ([F]r & [«?R»]rs & (∀t ([F]t & [«?R»]ts → t =E r)))>
1486   proof(safe intro!: "Ordinary.∃I"[where β=u] "&I" Fu Ordinary.GEN "→I")
1487     AOT_show <[«?R»]us>
1488       by (auto intro!: "β←C"(1) "cqt:2" "&I" prod_denotesI "∨I"(2)
1489         Π_minus_κI' Gs s_eq_v Ordinary.ψ
1490         "ord=Eequiv:1"[THEN "→E"])
1491 next
1492   fix t
1493   AOT_assume 0: <[F]t & [«?R»]ts>
1494   AOT_hence 1: <([[F]-ut & [[G]-vs & [R]ts) ∨ (t =E u & s =E v)>
1495     using "β→C"(1)[OF 0[THEN "&E"(2)], simplified] by blast
1496   moreover AOT_have <¬([[F]-ut & [[G]-vs & [R]ts)>
1497   proof (rule "raa-cor:2")
1498     AOT_assume <([[F]-ut & [[G]-vs & [R]ts)>
1499     AOT_hence <[[G]-vs> using "&E" by blast
1500     AOT_thus <s =E v & ¬(s =E v)>
1501     by (metis Π_minus_κE2 "≡E"(4) "reductio-aa:1" s_eq_v "thm-neg=E")
1502   qed
1503   ultimately AOT_have <t =E u & s =E v>
1504     by (metis "∨E"(2))
1505   AOT_thus <t =E u> using "&E" by blast
1506   qed
1507 }
1508 ultimately AOT_show <∃!r ([F]r & [«?R»]rs)>
1509   using "≡E"(2) "equi:1" "reductio-aa:2" by fast
1510 qed

```

```

1511 qed
1512
1513
1514 AOT_theorem "approx-cont:1": <math>\exists F \exists G \Diamond (F \approx_E G \ \&\ \Diamond \neg F \approx_E G)> \tag{737.1}
1515 proof -
1516   let ?P = <math>\langle\langle [\lambda x \ E!x \ \&\ \neg \mathcal{A}E!x] \rangle\rangle>
1517   AOT_have <math>\langle \Diamond q_0 \ \&\ \Diamond \neg q_0 \rangle> \text{ by (metis } q_0\_prop)
1518   AOT_hence 1: <math>\langle \Diamond \exists x (E!x \ \&\ \neg \mathcal{A}E!x) \ \&\ \Diamond \neg \exists x (E!x \ \&\ \neg \mathcal{A}E!x) \rangle>
1519     by (rule } q_0\_def[THEN "=_{df}E"(2), rotated])
1520     (simp add: "log-prop-prop:2")
1521   AOT_have  $\vartheta$ : <math>\langle \Diamond \exists x [\langle\langle ?P \rangle\rangle]x \ \&\ \Diamond \neg \exists x [\langle\langle ?P \rangle\rangle]x \rangle>
1522     apply (AOT_subst <math>\langle [\langle\langle ?P \rangle\rangle]x \rangle \langle E!x \ \&\ \neg \mathcal{A}E!x \rangle \text{ for: } x)
1523     apply (rule "beta-C-meta"[THEN "\to E"]; "cqt:2[lambda]")
1524     by (fact 1)
1525   show ?thesis
1526   proof (rule "\exists I"(1))+
1527     AOT_have <math>\langle \Diamond [L]^- \approx_E [\langle\langle ?P \rangle\rangle] \ \&\ \Diamond \neg [L]^- \approx_E [\langle\langle ?P \rangle\rangle] \rangle>
1528     proof (rule "&I"; rule "RM\Diamond"[THEN "\to E"]; (rule "\to I")?)
1529       AOT_modally_strict {
1530         AOT_assume A: <math>\langle \neg \exists x [\langle\langle ?P \rangle\rangle]x \rangle>
1531         AOT_show <math>\langle [L]^- \approx_E [\langle\langle ?P \rangle\rangle] \rangle>
1532         proof (safe intro!: "empty-approx:1"[unvarify F H, THEN "\to E"]
1533           "rel-neg-T:3" "&I")
1534           AOT_show <math>\langle [\langle\langle ?P \rangle\rangle] \downarrow \rangle> \text{ by "cqt:2[lambda]"}
1535         next
1536           AOT_show <math>\langle \neg \exists u [L]^- u \rangle>
1537           proof (rule "raa-cor:2")
1538             AOT_assume <math>\langle \exists u [L]^- u \rangle>
1539             then AOT_obtain u where <math>\langle [L]^- u \rangle>
1540               using "Ordinary.\exists E"[rotated] by blast
1541             moreover AOT_have <math>\langle \neg [L]^- u \rangle>
1542               using "thm-noncont-e-e:2"[THEN "contingent-properties:2"[THEN "\equiv_{df}E"],
1543                 THEN "&E"(2)]
1544               by (metis "qml:2"[axiom_inst] "rule-ui:3" "\to E")
1545             ultimately AOT_show <math>\langle p \ \&\ \neg p \rangle \text{ for } p
1546               by (metis "raa-cor:3")
1547           qed
1548         next
1549           AOT_show <math>\langle \neg \exists v [\langle\langle ?P \rangle\rangle]v \rangle>
1550           proof (rule "raa-cor:2")
1551             AOT_assume <math>\langle \exists v [\langle\langle ?P \rangle\rangle]v \rangle>
1552             then AOT_obtain u where <math>\langle [\langle\langle ?P \rangle\rangle]u \rangle>
1553               using "Ordinary.\exists E"[rotated] by blast
1554             AOT_hence <math>\langle [\langle\langle ?P \rangle\rangle]u \rangle>
1555               using "&E" by blast
1556             AOT_hence <math>\langle \exists x [\langle\langle ?P \rangle\rangle]x \rangle>
1557               by (rule "\exists I")
1558             AOT_thus <math>\langle \exists x [\langle\langle ?P \rangle\rangle]x \ \&\ \neg \exists x [\langle\langle ?P \rangle\rangle]x \rangle>
1559               using A "&I" by blast
1560           qed
1561         qed
1562       }
1563     next
1564     AOT_show <math>\langle \Diamond \neg \exists x [\langle\langle ?P \rangle\rangle]x \rangle>
1565     using  $\vartheta$  "&E" by blast
1566   next
1567   AOT_modally_strict {
1568     AOT_assume A: <math>\langle \exists x [\langle\langle ?P \rangle\rangle]x \rangle>
1569     AOT_have B: <math>\langle \neg [\langle\langle ?P \rangle\rangle] \approx_E [L]^- \rangle>
1570     proof (safe intro!: "empty-approx:2"[unvarify F H, THEN "\to E"]
1571       "rel-neg-T:3" "&I")
1572       AOT_show <math>\langle [\langle\langle ?P \rangle\rangle] \downarrow \rangle>
1573       by "cqt:2[lambda]"

```

```

1574     next
1575     AOT_obtain x where Px: <[<<?P>>]x>
1576     using A "∃E" by blast
1577     AOT_hence <E!x & ¬A!x>
1578     by (rule "β→C"(1))
1579     AOT_hence 1: <◇E!x>
1580     by (metis "T◇" "&E"(1) "vdash-properties:10")
1581     AOT_have <[λx ◇E!x]x>
1582     by (auto intro!: "β←C"(1) "cqt:2" 1)
1583     AOT_hence <O!x>
1584     by (rule AOT_ordinary[THEN "=dfI"(2), rotated]) "cqt:2[lambda]"
1585     AOT_hence <O!x & [<<?P>>]x>
1586     using Px "&I" by blast
1587     AOT_thus <∃u [<<?P>>]u>
1588     by (rule "∃I")
1589     next
1590     AOT_show <¬∃u [L]u>
1591     proof (rule "raa-cor:2")
1592     AOT_assume <∃u [L]u>
1593     then AOT_obtain u where <[L]u>
1594     using "Ordinary.∃E"[rotated] by blast
1595     moreover AOT_have <¬[L]u>
1596     using "thm-noncont-e-e:2"[THEN "contingent-properties:2"[THEN "≡dfE"]]
1597     by (metis "qml:2"[axiom_inst] "rule-ui:3" "→E" "&E"(2))
1598     ultimately AOT_show <p & ¬p> for p
1599     by (metis "raa-cor:3")
1600     qed
1601     qed
1602     AOT_show <¬[L] ≈E [<<?P>>]>
1603     proof (rule "raa-cor:2")
1604     AOT_assume <[L] ≈E [<<?P>>]>
1605     AOT_hence <[<<?P>>] ≈E [L]>
1606     apply (rule "eq-part:2"[unvarify F G, THEN "→E", rotated 2])
1607     apply "cqt:2[lambda]"
1608     by (simp add: "rel-neg-T:3")
1609     AOT_thus <[<<?P>>] ≈E [L] & ¬[<<?P>>] ≈E [L]>
1610     using B "&I" by blast
1611     qed
1612   }
1613   next
1614     AOT_show <◇∃x [<<?P>>]x>
1615     using ∅ "&E" by blast
1616     qed
1617     AOT_thus <◇([L] ≈E [<<?P>>] & ◇¬[L] ≈E [<<?P>>])>
1618     using "S5Basic:11" "≡E"(2) by blast
1619   next
1620     AOT_show <[λx [E!]x & ¬A[E!]x]↓>
1621     by "cqt:2"
1622   next
1623     AOT_show <[L]↓>
1624     by (simp add: "rel-neg-T:3")
1625   qed
1626   qed
1627
1628
1629   AOT_theorem "approx-cont:2":
1630     <∃F∃G ◇([λz A[F]z] ≈E G & ◇¬[λz A[F]z] ≈E G)>
1631   proof -
1632     let ?P = <<[λx E!x & ¬A!x]>>
1633     AOT_have <◇q0 & ◇¬q0> by (metis q0_prop)
1634     AOT_hence 1: <◇∃x (E!x & ¬A!x) & ◇¬∃x (E!x & ¬A!x)>
1635     by (rule q0_def[THEN "=dfE"(2), rotated])
1636     (simp add: "log-prop-prop:2")

```

(737.2)

```

1637 AOT_have  $\vartheta$ :  $\langle \diamond \exists x [\langle \langle ?P \rangle] x \ \& \ \diamond \neg \exists x [\langle \langle ?P \rangle] x] \rangle$ 
1638   apply (AOT_subst  $\langle [\langle \langle ?P \rangle] x \rangle \langle E!x \ \& \ \neg \mathcal{A}E!x \rangle$  for: x)
1639   apply (rule "beta-C-meta"[THEN " $\rightarrow E$ "]; "cqt:2")
1640   by (fact 1)
1641 show ?thesis
1642 proof (rule " $\exists I$ "(1))+
1643   AOT_have  $\langle \diamond [\lambda z \mathcal{A}[L^-]z] \approx_E [\langle \langle ?P \rangle] \ \& \ \diamond \neg [\lambda z \mathcal{A}[L^-]z] \approx_E [\langle \langle ?P \rangle] \rangle$ 
1644   proof (rule "&I"; rule "RM $\diamond$ "[THEN " $\rightarrow E$ "]; (rule " $\rightarrow I$ ")?)
1645     AOT_modally_strict {
1646       AOT_assume A:  $\langle \neg \exists x [\langle \langle ?P \rangle] x \rangle$ 
1647       AOT_show  $\langle [\lambda z \mathcal{A}[L^-]z] \approx_E [\langle \langle ?P \rangle] \rangle$ 
1648       proof (safe intro!: "empty-approx:1"[unvarify F H, THEN " $\rightarrow E$ "]
1649         "rel-neg-T:3" "&I")
1650         AOT_show  $\langle [\langle \langle ?P \rangle] \downarrow \rangle$  by "cqt:2"
1651     next
1652       AOT_show  $\langle \neg \exists u [\lambda z \mathcal{A}[L^-]z] u \rangle$ 
1653       proof (rule "raa-cor:2")
1654         AOT_assume  $\langle \exists u [\lambda z \mathcal{A}[L^-]z] u \rangle$ 
1655         then AOT_obtain u where  $\langle [\lambda z \mathcal{A}[L^-]z] u \rangle$ 
1656         using "Ordinary. $\exists E$ "[rotated] by blast
1657         AOT_hence  $\langle \mathcal{A}[L^-]u \rangle$ 
1658         using " $\beta \rightarrow C$ "(1) "&E" by blast
1659         moreover AOT_have  $\langle \Box \neg [L^-]u \rangle$ 
1660         using "thm-noncont-e-e:2"[THEN "contingent-properties:2"[THEN " $\equiv_{df} E$ "]]
1661         by (metis RN "qml:2"[axiom_inst] "rule-ui:3" " $\rightarrow E$ " "&E"(2))
1662         ultimately AOT_show  $\langle p \ \& \ \neg p \rangle$  for p
1663         by (metis "Act-Sub:3" "KBasic2:1" " $\equiv E$ "(1) "raa-cor:3" " $\rightarrow E$ ")
1664       qed
1665     next
1666       AOT_show  $\langle \neg \exists v [\langle \langle ?P \rangle] v \rangle$ 
1667       proof (rule "raa-cor:2")
1668         AOT_assume  $\langle \exists v [\langle \langle ?P \rangle] v \rangle$ 
1669         then AOT_obtain u where  $\langle [\langle \langle ?P \rangle] u \rangle$ 
1670         using "Ordinary. $\exists E$ "[rotated] by blast
1671         AOT_hence  $\langle [\langle \langle ?P \rangle] u \rangle$ 
1672         using "&E" by blast
1673         AOT_hence  $\langle \exists x [\langle \langle ?P \rangle] x \rangle$ 
1674         by (rule " $\exists I$ ")
1675         AOT_thus  $\langle \exists x [\langle \langle ?P \rangle] x \ \& \ \neg \exists x [\langle \langle ?P \rangle] x \rangle$ 
1676         using A "&I" by blast
1677       qed
1678     next
1679       AOT_show  $\langle [\lambda z \mathcal{A}[L^-]z] \downarrow \rangle$  by "cqt:2"
1680   qed
1681 }
1682 next
1683 AOT_show  $\langle \diamond \neg \exists x [\langle \langle ?P \rangle] x \rangle$  using  $\vartheta$  "&E" by blast
1684 next
1685 AOT_modally_strict {
1686   AOT_assume A:  $\langle \exists x [\langle \langle ?P \rangle] x \rangle$ 
1687   AOT_have B:  $\langle \neg [\langle \langle ?P \rangle] \approx_E [\lambda z \mathcal{A}[L^-]z] \rangle$ 
1688   proof (safe intro!: "empty-approx:2"[unvarify F H, THEN " $\rightarrow E$ "]
1689     "rel-neg-T:3" "&I")
1690     AOT_show  $\langle [\langle \langle ?P \rangle] \downarrow \rangle$  by "cqt:2"
1691   next
1692   AOT_obtain x where Px:  $\langle [\langle \langle ?P \rangle] x \rangle$ 
1693     using A " $\exists E$ " by blast
1694   AOT_hence  $\langle E!x \ \& \ \neg \mathcal{A}E!x \rangle$ 
1695     by (rule " $\beta \rightarrow C$ "(1))
1696   AOT_hence  $\langle \diamond E!x \rangle$ 
1697     by (metis "T $\diamond$ " "&E"(1) " $\rightarrow E$ ")
1698   AOT_hence  $\langle [\lambda x \diamond E!x] x \rangle$ 
1699     by (auto intro!: " $\beta \leftarrow C$ "(1) "cqt:2")

```

```

1700     AOT_hence <O!x>
1701     by (rule AOT_ordinary[THEN "=dfI"(2), rotated]) "cqt:2"
1702     AOT_hence <O!x & [«?P»]x>
1703     using Px "&I" by blast
1704     AOT_thus <∃u [«?P»]u>
1705     by (rule "∃I")
1706   next
1707     AOT_show <¬∃u [λz  $\mathcal{A}[L^-]z$ ]u>
1708     proof (rule "raa-cor:2")
1709       AOT_assume <∃u [λz  $\mathcal{A}[L^-]z$ ]u>
1710       then AOT_obtain u where <[λz  $\mathcal{A}[L^-]z$ ]u>
1711         using "Ordinary.∃E"[rotated] by blast
1712       AOT_hence < $\mathcal{A}[L^-]u$ >
1713         using " $\beta \rightarrow C$ "(1) "&E" by blast
1714       moreover AOT_have < $\Box \neg [L^-]u$ >
1715         using "thm-noncont-e-e:2"[THEN "contingent-properties:2"[THEN " $\equiv_{df}E$ "]]
1716         by (metis RN "qml:2"[axiom_inst] "rule-ui:3" " $\rightarrow E$ " "&E"(2))
1717       ultimately AOT_show <p & ¬p> for p
1718         by (metis "Act-Sub:3" "KBasic2:1" " $\equiv E$ "(1) "raa-cor:3" " $\rightarrow E$ ")
1719     qed
1720   next
1721     AOT_show <[λz  $\mathcal{A}[L^-]z$ ]↓> by "cqt:2"
1722   qed
1723   AOT_show <¬[λz  $\mathcal{A}[L^-]z$ ]  $\approx_E$  [«?P»]>
1724   proof (rule "raa-cor:2")
1725     AOT_assume <[λz  $\mathcal{A}[L^-]z$ ]  $\approx_E$  [«?P»]>
1726     AOT_hence <[«?P»]  $\approx_E$  [λz  $\mathcal{A}[L^-]z$ ]>
1727     by (rule "eq-part:2"[unvarify F G, THEN " $\rightarrow E$ ", rotated 2])
1728     "cqt:2"+
1729     AOT_thus <[«?P»]  $\approx_E$  [λz  $\mathcal{A}[L^-]z$ ] & ¬[«?P»]  $\approx_E$  [λz  $\mathcal{A}[L^-]z$ ]>
1730     using B "&I" by blast
1731   qed
1732 }
1733 next
1734   AOT_show <◇∃x [«?P»]x>
1735   using ∅ "&E" by blast
1736   qed
1737   AOT_thus <◇([λz  $\mathcal{A}[L^-]z$ ]  $\approx_E$  [«?P»]) & ◇¬[λz  $\mathcal{A}[L^-]z$ ]  $\approx_E$  [«?P»]>
1738   using "S5Basic:11" " $\equiv E$ "(2) by blast
1739 next
1740   AOT_show <[λx [E!]x & ¬ $\mathcal{A}[E!]x$ ]↓> by "cqt:2"
1741 next
1742   AOT_show <[L]^-↓>
1743   by (simp add: "rel-neg-T:3")
1744   qed
1745   qed
1746
1747   notepad
1748   begin
1749     text<We already have defined being equivalent on the ordinary objects in the
1750     Extended Relation Comprehension theory.>
1751     AOT_have <F  $\equiv_E$  G  $\equiv_{df}$  F↓ & G↓ & ∀u ([F]u  $\equiv$  [G]u)> for F G
1752     using eqE by blast
1753   end
1754
1755   AOT_theorem "apE-eqE:1": <F  $\equiv_E$  G  $\rightarrow$  F  $\approx_E$  G>
1756   proof(rule " $\rightarrow I$ ")
1757     AOT_assume 0: <F  $\equiv_E$  G>
1758     AOT_have <∃R R | : F  $\xrightarrow{1-1} \leftarrow_E$  G>
1759     proof (safe intro!: "∃I"(1)[where  $\tau = \ll (=E) \gg$ ] "equi:2"[THEN " $\equiv_{df}I$ " "&I"
1760     "=E[denotes]" "cqt:2[const_var]"[axiom_inst] Ordinary.GEN
1761     " $\rightarrow I$ " "equi:1"[THEN " $\equiv E$ "(2)])
1762     fix u

```



```

1763   AOT_assume Fu: <[F]u>
1764   AOT_hence Gu: <[G]u>
1765     using "≡dfE"[OF eqE, OF 0, THEN "&E"(2),
1766           THEN "Ordinary.∀E"[where α=u], THEN "≡E"(1)]
1767     Ordinary.ψ Fu by blast
1768   AOT_show <∃v ([G]v & u =E v & ∀v' ([G]v' & u =E v' → v' =E v))>
1769     by (safe intro!: "Ordinary.∃I"[where β=u] "&I" GEN "→I" Ordinary.ψ Gu
1770         "ord=Eequiv:1"[THEN "→E", OF Ordinary.ψ]
1771         "ord=Eequiv:2"[THEN "→E"] dest!: "&E"(2))
1772 next
1773   fix v
1774   AOT_assume Gv: <[G]v>
1775   AOT_hence Fv: <[F]v>
1776     using "≡dfE"[OF eqE, OF 0, THEN "&E"(2),
1777           THEN "Ordinary.∀E"[where α=v], THEN "≡E"(2)]
1778     Ordinary.ψ Gv by blast
1779   AOT_show <∃u ([F]u & u =E v & ∀v' ([F]v' & v' =E v → v' =E u))>
1780     by (safe intro!: "Ordinary.∃I"[where β=v] "&I" GEN "→I" Ordinary.ψ Fv
1781         "ord=Eequiv:1"[THEN "→E", OF Ordinary.ψ]
1782         "ord=Eequiv:2"[THEN "→E"] dest!: "&E"(2))
1783 qed
1784 AOT_thus <F ≈E G>
1785   by (rule "equi:3"[THEN "≡dfI"])
1786 qed
1787
1788 AOT_theorem "apE-eqE:2": <(F ≈E G & G ≡E H) → F ≈E H> (739.2)
1789 proof(rule "→I")
1790   AOT_assume <F ≈E G & G ≡E H>
1791   AOT_hence <F ≈E G> and <G ≈E H>
1792     using "apE-eqE:1"[THEN "→E"] "&E" by blast+
1793   AOT_thus <F ≈E H>
1794     by (metis Adjunction "eq-part:3" "vdash-properties:10")
1795 qed
1796
1797
1798 AOT_act_theorem "eq-part-act:1": <[λz A[F]z] ≡E F> (740.1)
1799 proof (safe intro!: eqE[THEN "≡dfI"] "&I" "cqt:2" Ordinary.GEN "→I")
1800   fix u
1801   AOT_have <[λz A[F]z]u ≡ A[F]u>
1802     by (rule "beta-C-meta"[THEN "→E"]) "cqt:2[lambda]"
1803   also AOT_have <... ≡ [F]u>
1804     using "act-conj-act:4" "logic-actual"[act_axiom_inst, THEN "→E"] by blast
1805   finally AOT_show <[λz A[F]z]u ≡ [F]u>.
1806 qed
1807
1808 AOT_act_theorem "eq-part-act:2": <[λz A[F]z] ≈E F> (740.2)
1809   by (safe intro!: "apE-eqE:1"[unvarify F, THEN "→E"] "eq-part-act:1") "cqt:2"
1810
1811
1812 AOT_theorem "actuallyF:1": <A(F ≈E [λz A[F]z])> (741.1)
1813 proof -
1814   AOT_have 1: <A([F]x ≡ A[F]x)> for x
1815     by (meson "Act-Basic:5" "act-conj-act:4" "≡E"(2) "Commutativity of ≡")
1816   AOT_have <A([F]x ≡ [λz A[F]z]x)> for x
1817     apply (AOT_subst <[λz A[F]z]x> <A[F]x>)
1818     apply (rule "beta-C-meta"[THEN "→E"])
1819     apply "cqt:2[lambda]"
1820     by (fact 1)
1821   AOT_hence <0!x → A([F]x ≡ [λz A[F]z]x)> for x
1822     by (metis "→I")
1823   AOT_hence <∀u A([F]u ≡ [λz A[F]z]u)>
1824     using "∀I" by fast
1825   AOT_hence 1: <A∀u ([F]u ≡ [λz A[F]z]u)>

```

```

1826   by (metis "Ordinary.res-var-bound-reas[2]" "→E")
1827 AOT_modally_strict {
1828   AOT_have <[λz  $\mathcal{A}[F]z$ ]↓> by "cqt:2"
1829 } note 2 = this
1830 AOT_have < $\mathcal{A}(F \equiv_E [\lambda z \mathcal{A}[F]z])$ >
1831   apply (AOT_subst < $F \equiv_E [\lambda z \mathcal{A}[F]z]$ > <∀u ([F]u ≡ [λz  $\mathcal{A}[F]z$ ]u)>>
1832   using eqE[THEN "≡df", THEN "≡S"(1), OF "&I",
1833     OF "cqt:2[const_var]"[axiom_inst], OF 2]
1834   by (auto simp: 1)
1835 moreover AOT_have < $\mathcal{A}(F \equiv_E [\lambda z \mathcal{A}[F]z] \rightarrow F \approx_E [\lambda z \mathcal{A}[F]z])$ >
1836   using "apE-eqE:1"[unvarify G, THEN "RA[2]", OF 2] by metis
1837 ultimately AOT_show < $\mathcal{A}F \approx_E [\lambda z \mathcal{A}[F]z]$ >
1838   by (metis "act-cond" "→E")
1839 qed
1840
1841 AOT_theorem "actuallyF:2": <Rigid([λz  $\mathcal{A}[F]z$ )]> (741.2)
1842 proof (safe intro!: GEN "→I" "df-rigid-rel:1"[THEN "≡dfI"] "&I")
1843   AOT_show <[λz  $\mathcal{A}[F]z$ ]↓> by "cqt:2"
1844 next
1845   AOT_show <□∀x ([λz  $\mathcal{A}[F]z$ ]x → □[λz  $\mathcal{A}[F]z$ ]x)>
1846   proof (rule RN; rule GEN; rule "→I")
1847     AOT_modally_strict {
1848       fix x
1849       AOT_assume <[λz  $\mathcal{A}[F]z$ ]x>
1850       AOT_hence < $\mathcal{A}[F]x$ >
1851       by (rule "β→C"(1))
1852       AOT_hence 1: <□ $\mathcal{A}[F]x$ > by (metis "Act-Basic:6" "≡E"(1))
1853       AOT_show <□[λz  $\mathcal{A}[F]z$ ]x>
1854       apply (AOT_subst <[λz  $\mathcal{A}[F]z$ ]x> < $\mathcal{A}[F]x$ >)
1855       apply (rule "beta-C-meta"[THEN "→E"])
1856       apply "cqt:2[lambda]"
1857       by (fact 1)
1858     }
1859   qed
1860 qed
1861
1862 AOT_theorem "approx-nec:1": <Rigid(F) →  $F \approx_E [\lambda z \mathcal{A}[F]z]$ > (742.1)
1863 proof (rule "→I")
1864   AOT_assume <Rigid([F])>
1865   AOT_hence A: <□∀x ([F]x → □[F]x)>
1866   using "df-rigid-rel:1"[THEN "≡dfE", THEN "&E"(2)] by blast
1867   AOT_hence 0: <∀x □([F]x → □[F]x)>
1868   using CBF[THEN "→E"] by blast
1869   AOT_hence 1: <∀x ([F]x → □[F]x)>
1870   using A "qml:2"[axiom_inst, THEN "→E"] by blast
1871   AOT_have act_F_den: <[λz  $\mathcal{A}[F]z$ ]↓>
1872   by "cqt:2"
1873   AOT_show < $F \approx_E [\lambda z \mathcal{A}[F]z]$ >
1874   proof (safe intro!: "apE-eqE:1"[unvarify G, THEN "→E"] eqE[THEN "≡dfI"] "&I"
1875     "cqt:2" act_F_den Ordinary.GEN "→I" "≡I")
1876     fix u
1877     AOT_assume <[F]u>
1878     AOT_hence <□[F]u>
1879     using 1[THEN "∀E"(2), THEN "→E"] by blast
1880     AOT_hence act_F_u: < $\mathcal{A}[F]u$ >
1881     by (metis "nec-imp-act" "→E")
1882     AOT_show <[λz  $\mathcal{A}[F]z$ ]u>
1883     by (auto intro!: "β←C"(1) "cqt:2" act_F_u)
1884   next
1885     fix u
1886     AOT_assume <[λz  $\mathcal{A}[F]z$ ]u>
1887     AOT_hence < $\mathcal{A}[F]u$ >
1888     by (rule "β→C"(1))

```

```

1889   AOT_thus <[F]u>
1890     using 0[THEN "∀E"(2)]
1891     by (metis "≡E"(1) "sc-eq-fur:2" "→E")
1892   qed
1893 qed
1894
1895
1896 AOT_theorem "approx-nec:2": (742.2)
1897   <F ≈E G ≡ ∀H ([λz A[H]z] ≈E F ≡ [λz A[H]z] ≈E G)>
1898 proof(rule "≡I"; rule "→I")
1899   AOT_assume 0: <F ≈E G>
1900   AOT_assume 0: <F ≈E G>
1901   AOT_hence <∀H (H ≈E F ≡ H ≈E G)>
1902     using "eq-part:4"[THEN "≡E"(1), OF 0] by blast
1903   AOT_have <[λz A[H]z] ≈E F ≡ [λz A[H]z] ≈E G> for H
1904     by (rule "∀E"(1)[OF "eq-part:4"[THEN "≡E"(1), OF 0]]) "cqt:2"
1905   AOT_thus <∀H ([λz A[H]z] ≈E F ≡ [λz A[H]z] ≈E G)>
1906     by (rule GEN)
1907 next
1908   AOT_assume 0: <∀H ([λz A[H]z] ≈E F ≡ [λz A[H]z] ≈E G)>
1909   AOT_obtain H where <Rigidifies(H,F)>
1910     using "rigid-der:3" "∃E" by metis
1911   AOT_hence H: <Rigid(H) & ∀x ([H]x ≡ [F]x)>
1912     using "df-rigid-rel:2"[THEN "≡dfE"] by blast
1913   AOT_have H_rigid: <□∀x ([H]x → □[H]x)>
1914     using H[THEN "&E"(1), THEN "df-rigid-rel:1"[THEN "≡dfE"], THEN "&E"(2)].
1915   AOT_hence <∀x □([H]x → □[H]x)>
1916     using "CBF" "vdash-properties:10" by blast
1917   AOT_hence <□([H]x → □[H]x)> for x using "∀E"(2) by blast
1918   AOT_hence rigid: <[H]x ≡ A[H]x> for x
1919     by (metis "≡E"(6) "oth-class-taut:3:a" "sc-eq-fur:2" "→E")
1920   AOT_have <H ≡E F>
1921 proof (safe intro!: eqE[THEN "≡dfI"] "&I" "cqt:2" Ordinary.GEN "→I")
1922   AOT_show <[H]u ≡ [F]u> for u using H[THEN "&E"(2)] "∀E"(2) by fast
1923 qed
1924   AOT_hence <H ≈E F>
1925     by (rule "apE-eqE:2"[THEN "→E", OF "&I", rotated])
1926     (simp add: "eq-part:1")
1927   AOT_hence F_approx_H: <F ≈E H>
1928     by (metis "eq-part:2" "→E")
1929   moreover AOT_have H_eq_act_H: <H ≡E [λz A[H]z]>
1930 proof (safe intro!: eqE[THEN "≡dfI"] "&I" "cqt:2" Ordinary.GEN "→I")
1931   AOT_show <[H]u ≡ [λz A[H]z]u> for u
1932     apply (AOT_subst <[λz A[H]z]u> <A[H]u>)
1933     apply (rule "beta-C-meta"[THEN "→E"])
1934     apply "cqt:2[lambda]"
1935     using rigid by blast
1936 qed
1937   AOT_have a: <F ≈E [λz A[H]z]>
1938     apply (rule "apE-eqE:2"[unvarify H, THEN "→E"])
1939     apply "cqt:2[lambda]"
1940     using F_approx_H H_eq_act_H "&I" by blast
1941   AOT_hence <[λz A[H]z] ≈E F>
1942     apply (rule "eq-part:2"[unvarify G, THEN "→E", rotated])
1943     by "cqt:2[lambda]"
1944   AOT_hence b: <[λz A[H]z] ≈E G>
1945     by (rule 0[THEN "∀E"(1), THEN "≡E"(1), rotated]) "cqt:2"
1946   AOT_show <F ≈E G>
1947     by (rule "eq-part:3"[unvarify G, THEN "→E", rotated, OF "&I", OF a, OF b])
1948     "cqt:2"
1949 qed
1950
1951 AOT_theorem "approx-nec:3": (742.3)

```

```

1952   <(Rigid(F) & Rigid(G)) → □(F ≈E G → □F ≈E G)>
1953 proof (rule "→I")
1954   AOT_assume <Rigid(F) & Rigid(G)>
1955   AOT_hence <□∀x([F]x → □[F]x) and □∀x([G]x → □[G]x)>
1956     using "df-rigid-rel:1"[THEN "≡dfE", THEN "&E"(2)] "&E" by blast+
1957   AOT_hence <□(□∀x([F]x → □[F]x) & □∀x([G]x → □[G]x))>
1958     using "KBasic:3" "4" "&I" "≡E"(2) "vdash-properties:10" by meson
1959   moreover AOT_have <□(□∀x([F]x → □[F]x) & □∀x([G]x → □[G]x)) →
1960     □(F ≈E G → □F ≈E G)>
1961   proof(rule RM; rule "→I"; rule "→I")
1962     AOT_modally_strict {
1963       AOT_assume <□∀x([F]x → □[F]x) & □∀x([G]x → □[G]x)>
1964       AOT_hence <□∀x([F]x → □[F]x)> and <□∀x([G]x → □[G]x)>
1965         using "&E" by blast+
1966       AOT_hence <∀x□([F]x → □[F]x) and ∀x□([G]x → □[G]x)>
1967         using CBF[THEN "→E"] by blast+
1968       AOT_hence F_nec: <□([F]x → □[F]x)>
1969         and G_nec: <□([G]x → □[G]x)> for x
1970         using "∀E"(2) by blast+
1971       AOT_assume <F ≈E G>
1972       AOT_hence <∃R R | : F 1-1 ↔E G>
1973         by (metis "≡dfE" "equi:3")
1974       then AOT_obtain R where <R | : F 1-1 ↔E G>
1975         using "∃E"[rotated] by blast
1976       AOT_hence C1: <∀u ([F]u → ∃!v ([G]v & [R]uv))>
1977         and C2: <∀v ([G]v → ∃!u ([F]u & [R]uv))>
1978         using "equi:2"[THEN "≡dfE"] "&E" by blast+
1979       AOT_obtain R' where <Rigidifies(R', R)>
1980         using "rigid-der:3" "∃E"[rotated] by blast
1981       AOT_hence 1: <Rigid(R') & ∀x1...∀xn ([R']x1...xn ≡ [R]x1...xn)>
1982         using "df-rigid-rel:2"[THEN "≡dfE"] by blast
1983       AOT_hence <□∀x1...∀xn ([R']x1...xn → □[R']x1...xn)>
1984         using "df-rigid-rel:1"[THEN "≡dfE"] "&E" by blast
1985       AOT_hence <∀x1...∀xn (◇[R']x1...xn → □[R']x1...xn)>
1986         using "≡E"(1) "rigid-rel-thms:1" by blast
1987       AOT_hence D: <∀x1∀x2 (◇[R']x1x2 → □[R']x1x2)>
1988         using tuple_forall[THEN "≡dfE"] by blast
1989       AOT_have E: <∀x1∀x2 ([R']x1x2 ≡ [R]x1x2)>
1990         using tuple_forall[THEN "≡dfE", OF 1[THEN "&E"(2)]] by blast
1991       AOT_have <∀u □([F]u → ∃!v ([G]v & [R']uv))>
1992         and <∀v □([G]v → ∃!u ([F]u & [R']uv))>
1993       proof (safe intro!: Ordinary.GEN "→I")
1994         fix u
1995         AOT_show <□([F]u → ∃!v ([G]v & [R']uv))>
1996         proof (rule "raa-cor:1")
1997           AOT_assume <¬□([F]u → ∃!v ([G]v & [R']uv))>
1998           AOT_hence 1: <◇¬([F]u → ∃!v ([G]v & [R']uv))>
1999             using "KBasic:11" "≡E"(1) by blast
2000           AOT_have <◇([F]u & ¬∃!v ([G]v & [R']uv))>
2001             apply (AOT_subst <[F]u & ¬∃!v ([G]v & [R']uv)>
2002               <¬([F]u → ∃!v ([G]v & [R']uv))>)
2003             apply (meson "≡E"(6) "oth-class-taut:1:b" "oth-class-taut:3:a")
2004             by (fact 1)
2005           AOT_hence A: <◇[F]u & ◇¬∃!v ([G]v & [R']uv)>
2006             using "KBasic2:3" "→E" by blast
2007           AOT_hence <□[F]u>
2008             using F_nec "&E"(1) "≡E"(1) "sc-eq-box-box:1" "→E" by blast
2009           AOT_hence <[F]u>
2010             by (metis "qml:2"[axiom_inst] "→E")
2011           AOT_hence <∃!v ([G]v & [R]uv)>
2012             using C1[THEN "Ordinary.∀E", THEN "→E"] by blast
2013           AOT_hence <∃v ([G]v & [R]uv & ∀v' ([G]v' & [R]uv' → v' =E v))>
2014             using "equi:1"[THEN "≡E"(1)] by auto

```

```

2015 then AOT_obtain a where
2016   a_prop: <O!a & ([G]a & [R]ua & ∀v' ([G]v' & [R]uv' → v' =E a))>
2017   using "∃E"[rotated] by blast
2018 AOT_have <∃v □([G]v & [R']uv & ∀v' ([G]v' & [R']uv' → v' =E v))>
2019 proof(safe intro!: "∃I"(2)[where β=a] "&I" a_prop[THEN "&E"(1)]
2020         "KBasic:3"[THEN "≡E"(2)])
2021   AOT_show <□[G]a>
2022     using a_prop[THEN "&E"(2), THEN "&E"(1), THEN "&E"(1)]
2023     by (metis G_nec "qml:2"[axiom_inst] "→E")
2024 next
2025 AOT_show <□[R']ua>
2026   using D[THEN "∀E"(2), THEN "∀E"(2), THEN "→E"]
2027     E[THEN "∀E"(2), THEN "∀E"(2), THEN "≡E"(2),
2028       OF a_prop[THEN "&E"(2), THEN "&E"(1), THEN "&E"(2)]]
2029   by (metis "T◇" "→E")
2030 next
2031 AOT_have <∀v' □([G]v' & [R']uv' → v' =E a)>
2032 proof (rule Ordinary.GEN; rule "raa-cor:1")
2033   fix v'
2034   AOT_assume <¬□([G]v' & [R']uv' → v' =E a)>
2035   AOT_hence <◇¬([G]v' & [R']uv' → v' =E a)>
2036     by (metis "KBasic:11" "≡E"(1))
2037   AOT_hence <◇([G]v' & [R']uv' & ¬v' =E a)>
2038     by (AOT_subst <[G]v' & [R']uv' & ¬v' =E a>
2039         <¬([G]v' & [R']uv' → v' =E a)>)
2040     (meson "≡E"(6) "oth-class-taut:1:b" "oth-class-taut:3:a")
2041   AOT_hence 1: <◇[G]v'> and 2: <◇[R']uv'> and 3: <◇¬v' =E a>
2042     using "KBasic2:3"[THEN "→E", THEN "&E"(1)]
2043         "KBasic2:3"[THEN "→E", THEN "&E"(2)] by blast+
2044   AOT_have Gv': <[G]v'> using G_nec 1
2045     by (meson "B◇" "KBasic:13" "→E")
2046   AOT_have <□[R']uv'>
2047     using 2 D[THEN "∀E"(2), THEN "∀E"(2), THEN "→E"] by blast
2048   AOT_hence R'uv': <[R']uv'>
2049     by (metis "B◇" "T◇" "→E")
2050   AOT_hence <[R]uv'>
2051     using E[THEN "∀E"(2), THEN "∀E"(2), THEN "≡E"(1)] by blast
2052   AOT_hence <v' =E a>
2053     using a_prop[THEN "&E"(2), THEN "&E"(2), THEN "Ordinary.∀E",
2054               THEN "→E", OF "&I", OF Gv'] by blast
2055   AOT_hence <□(v' =E a)>
2056     by (metis "id-nec3:1" "≡E"(4) "raa-cor:3")
2057   moreover AOT_have <¬□(v' =E a)>
2058     using 3 "KBasic:11" "≡E"(2) by blast
2059   ultimately AOT_show <□(v' =E a) & ¬□(v' =E a)>
2060     using "&I" by blast
2061 qed
2062 AOT_thus <□∀v' ([G]v' & [R']uv' → v' =E a)>
2063   using "Ordinary.res-var-bound-reas[BF]" "→E" by fast
2064 qed
2065 AOT_hence <□∃v ([G]v & [R']uv & ∀v' ([G]v' & [R']uv' → v' =E v))>
2066   using "Ordinary.res-var-bound-reas[Buridan]" "→E" by fast
2067 AOT_hence <□∃!v ([G]v & [R']uv)>
2068   by (AOT_subst_thm "equi:1")
2069 moreover AOT_have <¬□∃!v ([G]v & [R']uv)>
2070   using A[THEN "&E"(2)] "KBasic:11"[THEN "≡E"(2)] by blast
2071 ultimately AOT_show <□∃!v ([G]v & [R']uv) & ¬□∃!v ([G]v & [R']uv)>
2072   by (rule "&I")
2073 qed
2074 next
2075 fix v
2076 AOT_show <□([G]v → ∃!u ([F]u & [R']uv))>
2077 proof (rule "raa-cor:1")

```

```

2078 AOT_assume <¬□([G]v → ∃!u ([F]u & [R']uv))>
2079 AOT_hence 1: <◇¬([G]v → ∃!u ([F]u & [R']uv))>
2080   using "KBasic:11" "≡E"(1) by blast
2081 AOT_hence <◇([G]v & ¬∃!u ([F]u & [R']uv))>
2082   by (AOT_subst <[G]v & ¬∃!u ([F]u & [R']uv)>
2083       <¬([G]v → ∃!u ([F]u & [R']uv))>)
2084   (meson "≡E"(6) "oth-class-taut:1:b" "oth-class-taut:3:a")
2085 AOT_hence A: <◇[G]v & ◇¬∃!u ([F]u & [R']uv)>
2086   using "KBasic2:3" "→E" by blast
2087 AOT_hence <□[G]v>
2088   using G_nec "&E"(1) "≡E"(1) "sc-eq-box-box:1" "→E" by blast
2089 AOT_hence <[G]v> by (metis "qml:2"[axiom_inst] "→E")
2090 AOT_hence <∃!u ([F]u & [R]uv)>
2091   using C2[THEN "Ordinary.∀E", THEN "→E"] by blast
2092 AOT_hence <∃u ([F]u & [R]uv & ∀u' ([F]u' & [R]u'v → u' =E u))>
2093   using "equi:1"[THEN "≡E"(1)] by auto
2094 then AOT_obtain a where
2095   a_prop: <0!a & ([F]a & [R]av & ∀u' ([F]u' & [R]u'v → u' =E a))>
2096   using "≡E"[rotated] by blast
2097 AOT_have <∃u □([F]u & [R']uv & ∀u' ([F]u' & [R']u'v → u' =E u))>
2098 proof(safe intro!: "∃I"(2)[where β=a] "&I" a_prop[THEN "&E"(1)]
2099       "KBasic:3"[THEN "≡E"(2)])
2100   AOT_show <□[F]a>
2101     using a_prop[THEN "&E"(2), THEN "&E"(1), THEN "&E"(1)]
2102     by (metis F_nec "qml:2"[axiom_inst] "→E")
2103 next
2104   AOT_show <□[R']av>
2105     using D[THEN "∀E"(2), THEN "∀E"(2), THEN "→E"]
2106     E[THEN "∀E"(2), THEN "∀E"(2), THEN "≡E"(2),
2107       OF a_prop[THEN "&E"(2), THEN "&E"(1), THEN "&E"(2)]]
2108     by (metis "T◇" "→E")
2109 next
2110   AOT_have <∀u' □([F]u' & [R']u'v → u' =E a)>
2111 proof (rule Ordinary.GEN; rule "raa-cor:1")
2112   fix u'
2113   AOT_assume <¬□([F]u' & [R']u'v → u' =E a)>
2114   AOT_hence <◇¬([F]u' & [R']u'v → u' =E a)>
2115     by (metis "KBasic:11" "≡E"(1))
2116   AOT_hence <◇([F]u' & [R']u'v & ¬u' =E a)>
2117     by (AOT_subst <[F]u' & [R']u'v & ¬u' =E a>
2118         <¬([F]u' & [R']u'v → u' =E a)>)
2119     (meson "≡E"(6) "oth-class-taut:1:b" "oth-class-taut:3:a")
2120   AOT_hence 1: <◇[F]u'> and 2: <◇[R']u'v> and 3: <◇¬u' =E a>
2121     using "KBasic2:3"[THEN "→E", THEN "&E"(1)]
2122     "KBasic2:3"[THEN "→E", THEN "&E"(2)] by blast+
2123   AOT_have Fu': <[F]u'> using F_nec 1
2124     by (meson "B◇" "KBasic:13" "→E")
2125   AOT_have <□[R']u'v>
2126     using 2 D[THEN "∀E"(2), THEN "∀E"(2), THEN "→E"] by blast
2127   AOT_hence R'u'v: <[R']u'v>
2128     by (metis "B◇" "T◇" "→E")
2129   AOT_hence <[R]u'v>
2130     using E[THEN "∀E"(2), THEN "∀E"(2), THEN "≡E"(1)] by blast
2131   AOT_hence <u' =E a>
2132     using a_prop[THEN "&E"(2), THEN "&E"(2), THEN "Ordinary.∀E",
2133       THEN "→E", OF "&I", OF Fu'] by blast
2134   AOT_hence <□(u' =E a)>
2135     by (metis "id-nec3:1" "≡E"(4) "raa-cor:3")
2136 moreover AOT_have <¬□(u' =E a)>
2137   using 3 "KBasic:11" "≡E"(2) by blast
2138 ultimately AOT_show <□(u' =E a) & ¬□(u' =E a)>
2139   using "&I" by blast
2140 qed

```

```

2141     AOT_thus <□∀u' ([F]u' & [R']u'v → u' =E a)>
2142         using "Ordinary.res-var-bound-reas[BF]" "→E" by fast
2143 qed
2144 AOT_hence 1: <□∃u ([F]u & [R']uv & ∀u' ([F]u' & [R']u'v → u' =E u))>
2145     using "Ordinary.res-var-bound-reas[Buridan]" "→E" by fast
2146 AOT_hence <□∃!u ([F]u & [R']uv)>
2147     by (AOT_subst_thm "equi:1")
2148 moreover AOT_have <¬□∃!u ([F]u & [R']uv)>
2149     using A[THEN "&E"(2)] "KBasic:11"[THEN "≡E"(2)] by blast
2150 ultimately AOT_show <□∃!u ([F]u & [R']uv) & ¬□∃!u ([F]u & [R']uv)>
2151     by (rule "&I")
2152 qed
2153 qed
2154 AOT_hence <□∀u ([F]u → ∃!v ([G]v & [R']uv))>
2155     and <□∀v ([G]v → ∃!u ([F]u & [R']uv))>
2156     using "Ordinary.res-var-bound-reas[BF]"[THEN "→E"] by auto
2157 moreover AOT_have <□[R']↓> and <□[F]↓> and <□[G]↓>
2158     by (simp_all add: "ex:2:a")
2159 ultimately AOT_have <□([R']↓ & [F]↓ & [G]↓ & ∀u ([F]u → ∃!v ([G]v & [R']uv)) &
2160     ∀v ([G]v → ∃!u ([F]u & [R']uv))>
2161     using "KBasic:3" "&I" "≡E"(2) by meson
2162 AOT_hence <□R' | : F1-1 ↔E G>
2163     by (AOT_subst_def "equi:2")
2164 AOT_hence <∃R □R | : F1-1 ↔E G>
2165     by (rule "∃I"(2))
2166 AOT_hence <□∃R R | : F1-1 ↔E G>
2167     by (metis Buridan "→E")
2168 AOT_thus <□F ≈E G>
2169     by (AOT_subst_def "equi:3")
2170 }
2171 qed
2172 ultimately AOT_show <□(F ≈E G → □F ≈E G)>
2173     using "→E" by blast
2174 qed
2175
2176
2177 AOT_define numbers :: <τ ⇒ τ ⇒ φ> (<Numbers'(_,_)>) (744)
2178     <Numbers(x,G) ≡df A!x & G↓ & ∀F(x[F] ≡ [λz A[F]z] ≈E G)>
2179
2180 AOT_theorem "numbers[den]": (744)
2181     <Π↓ → (Numbers(κ, Π) ≡ A!κ & ∀F(κ[F] ≡ [λz A[F]z] ≈E Π))>
2182     apply (safe intro!: numbers[THEN "≡dfI"] "&I" "≡I" "→I" "cqt:2"
2183         dest!: numbers[THEN "≡dfE"])
2184     using "&E" by blast+
2185
2186 AOT_theorem "num-tran:1": (745.1)
2187     <G ≈E H → (Numbers(x, G) ≡ Numbers(x, H))>
2188     proof (safe intro!: "→I" "≡I")
2189         AOT_assume 0: <G ≈E H>
2190         AOT_assume <Numbers(x, G)>
2191         AOT_hence Ax: <A!x> and ∅: <∀F (x[F] ≡ [λz A[F]z] ≈E G)>
2192             using numbers[THEN "≡dfE"] "&E" by blast+
2193         AOT_show <Numbers(x, H)>
2194         proof (safe intro!: numbers[THEN "≡dfI"] "&I" Ax "cqt:2" GEN)
2195             fix F
2196             AOT_have <x[F] ≡ [λz A[F]z] ≈E G>
2197                 using ∅[THEN "∀E"(2)].
2198             also AOT_have <... ≡ [λz A[F]z] ≈E H>
2199                 using 0 "approx-nec:2"[THEN "≡E"(1), THEN "∀E"(2)] by metis
2200             finally AOT_show <x[F] ≡ [λz A[F]z] ≈E H>.
2201         qed
2202     next
2203     AOT_assume <G ≈E H>

```



```

2204 AOT_hence 0: <H  $\approx_E$  G>
2205   by (metis "eq-part:2" " $\rightarrow E$ ")
2206 AOT_assume <Numbers(x, H)>
2207 AOT_hence Ax: <A!x> and  $\vartheta$ : < $\forall F (x[F] \equiv [\lambda z \mathcal{A}[F]z] \approx_E H)$ >
2208   using numbers[THEN " $\equiv_{df} E$ "] "&E" by blast+
2209 AOT_show <Numbers(x, G)>
2210 proof(safe intro!: numbers[THEN " $\equiv_{df} I$ "] "&I" Ax "cqt:2" GEN)
2211   fix F
2212   AOT_have <x[F]  $\equiv [\lambda z \mathcal{A}[F]z] \approx_E H$ >
2213     using  $\vartheta$ [THEN " $\forall E$ "](2)].
2214   also AOT_have <...  $\equiv [\lambda z \mathcal{A}[F]z] \approx_E G$ >
2215     using 0 "approx-nec:2"[THEN " $\equiv E$ "](1), THEN " $\forall E$ "](2)] by metis
2216   finally AOT_show <x[F]  $\equiv [\lambda z \mathcal{A}[F]z] \approx_E G$ >.
2217 qed
2218 qed
2219
2220 AOT_theorem "num-tran:2": (745.2)
2221   <(Numbers(x, G) & Numbers(x, H))  $\rightarrow G \approx_E H$ >
2222 proof (rule " $\rightarrow I$ "; frule "&E"(1); drule "&E"(2))
2223   AOT_assume <Numbers(x, G)>
2224   AOT_hence < $\forall F (x[F] \equiv [\lambda z \mathcal{A}[F]z] \approx_E G)$ >
2225     using numbers[THEN " $\equiv_{df} E$ "] "&E" by blast
2226   AOT_hence 1: <x[F]  $\equiv [\lambda z \mathcal{A}[F]z] \approx_E G$ > for F
2227     using " $\forall E$ "](2) by blast
2228   AOT_assume <Numbers(x, H)>
2229   AOT_hence < $\forall F (x[F] \equiv [\lambda z \mathcal{A}[F]z] \approx_E H)$ >
2230     using numbers[THEN " $\equiv_{df} E$ "] "&E" by blast
2231   AOT_hence <x[F]  $\equiv [\lambda z \mathcal{A}[F]z] \approx_E H$ > for F
2232     using " $\forall E$ "](2) by blast
2233   AOT_hence < $[\lambda z \mathcal{A}[F]z] \approx_E G \equiv [\lambda z \mathcal{A}[F]z] \approx_E H$ > for F
2234     by (metis "1" " $\equiv E$ ")(6))
2235   AOT_thus <G  $\approx_E H$ >
2236     using "approx-nec:2"[THEN " $\equiv E$ "](2), OF GEN] by blast
2237 qed
2238
2239 AOT_theorem "num-tran:3": (745.3)
2240   <G  $\equiv_E H \rightarrow (Numbers(x, G) \equiv Numbers(x, H))$ >
2241   using "apE-eqE:1" "Hypothetical Syllogism" "num-tran:1" by blast
2242
2243 AOT_theorem "pre-Hume": (746)
2244   <(Numbers(x, G) & Numbers(y, H))  $\rightarrow (x = y \equiv G \approx_E H)$ >
2245 proof(safe intro!: " $\rightarrow I$ " " $\equiv I$ "; frule "&E"(1); drule "&E"(2))
2246   AOT_assume <Numbers(x, G)>
2247   moreover AOT_assume <x = y>
2248   ultimately AOT_have <Numbers(y, G)> by (rule "rule=E")
2249   moreover AOT_assume <Numbers(y, H)>
2250   ultimately AOT_show <G  $\approx_E H$ > using "num-tran:2" " $\rightarrow E$ " "&I" by blast
2251 next
2252   AOT_assume <Numbers(x, G)>
2253   AOT_hence Ax: <A!x> and xF: < $\forall F (x[F] \equiv [\lambda z \mathcal{A}[F]z] \approx_E G)$ >
2254     using numbers[THEN " $\equiv_{df} E$ "] "&E" by blast+
2255   AOT_assume <Numbers(y, H)>
2256   AOT_hence Ay: <A!y> and yF: < $\forall F (y[F] \equiv [\lambda z \mathcal{A}[F]z] \approx_E H)$ >
2257     using numbers[THEN " $\equiv_{df} E$ "] "&E" by blast+
2258   AOT_assume G_approx_H: <G  $\approx_E H$ >
2259   AOT_show <x = y>
2260 proof(rule "ab-obey:1"[THEN " $\rightarrow E$ ", THEN " $\rightarrow E$ ", OF "&I", OF Ax, OF Ay]; rule GEN)
2261   fix F
2262   AOT_have <x[F]  $\equiv [\lambda z \mathcal{A}[F]z] \approx_E G$ >
2263     using xF[THEN " $\forall E$ "](2)].
2264   also AOT_have <...  $\equiv [\lambda z \mathcal{A}[F]z] \approx_E H$ >
2265     using "approx-nec:2"[THEN " $\equiv E$ "](1), OF G_approx_H, THEN " $\forall E$ "](2)].
2266   also AOT_have <...  $\equiv y[F]$ >

```



```

2267     using yF[THEN "∀E"(2), symmetric].
2268     finally AOT_show <x[F] ≡ y[F]>.
2269   qed
2270 qed
2271
2272 AOT_theorem "two-num-not":
2273   <∃u∃v(u ≠ v) → ∃x∃G∃H(Numbers(x,G) & Numbers(x, H) & ¬G ≡E H)>
2274 proof (rule "→I")
2275   AOT_have eqE_den: <[λx x =E y]↓> for y by "cqt:2"
2276   AOT_assume <∃u∃v(u ≠ v)>
2277   then AOT_obtain c where Oc: <0!c> and <∃v (c ≠ v)>
2278     using "&E" "∃E"[rotated] by blast
2279   then AOT_obtain d where Od: <0!d> and c_noteq_d: <c ≠ d>
2280     using "&E" "∃E"[rotated] by blast
2281   AOT_hence c_noteqE_d: <c ≠E d>
2282     using "=E-simple:2"[THEN "→E"] "=E-simple:2" "≡E"(2) "modus-tollens:1"
2283     "=-infix" "≡dfE" "thm-neg=E" by fast
2284   AOT_hence not_c_eqE_d: <¬c =E d>
2285     using "≡E"(1) "thm-neg=E" by blast
2286   AOT_have <∃x (A!x & ∀F (x[F] ≡ [λz A[F]z] ≈E [λx x =E c]))>
2287     by (simp add: "A-objects"[axiom_inst])
2288   then AOT_obtain a where a_prop: <A!a & ∀F (a[F] ≡ [λz A[F]z] ≈E [λx x =E c])>
2289     using "∃E"[rotated] by blast
2290   AOT_have <∃x (A!x & ∀F (x[F] ≡ [λz A[F]z] ≈E [λx x =E d]))>
2291     by (simp add: "A-objects" "vdash-properties:1[2]")
2292   then AOT_obtain b where b_prop: <A!b & ∀F (b[F] ≡ [λz A[F]z] ≈E [λx x =E d])>
2293     using "∃E"[rotated] by blast
2294   AOT_have num_a_eq_c: <Numbers(a, [λx x =E c])>
2295     by (safe intro!: numbers[THEN "≡dfI"] "&I" a_prop[THEN "&E"(1)]
2296         a_prop[THEN "&E"(2)]) "cqt:2"
2297   moreover AOT_have num_b_eq_d: <Numbers(b, [λx x =E d])>
2298     by (safe intro!: numbers[THEN "≡dfI"] "&I" b_prop[THEN "&E"(1)]
2299         b_prop[THEN "&E"(2)]) "cqt:2"
2300   moreover AOT_have <[λx x =E c] ≈E [λx x =E d]>
2301   proof (rule "equi:3"[THEN "≡dfI"])
2302     let ?R = <<<[λxy (x =E c & y =E d)]>>
2303     AOT_have Rcd: <[<<?R>>]cd>
2304       by (auto intro!: "β←C"(1) "cqt:2" "&I" prod_denotesI
2305           "ord=Eequiv:1"[THEN "→E"] Od Oc)
2306     AOT_show <∃R R |: [λx x =E c] 1-1↔E [λx x =E d]>
2307     proof (safe intro!: "∃I"(1)[where τ=<?R>] "equi:2"[THEN "≡dfI"] "&I"
2308         eqE_den Ordinary.GEN "→I")
2309       AOT_show <<<?R>>↓> by "cqt:2"
2310   next
2311     fix u
2312     AOT_assume <[λx x =E c]u>
2313     AOT_hence <u =E c>
2314       by (metis "β→C"(1))
2315     AOT_hence u_is_c: <u = c>
2316       by (metis "=E-simple:2" "→E")
2317     AOT_show <∃!v ([λx x =E d]v & [<<?R>>]uv)>
2318     proof (safe intro!: "equi:1"[THEN "≡E"(2)] "∃I"(2)[where β=d] "&I"
2319         Od Ordinary.GEN "→I")
2320       AOT_show <[λx x =E d]d>
2321       by (auto intro!: "β←C"(1) "cqt:2" "ord=Eequiv:1"[THEN "→E", OF Od])
2322   next
2323     AOT_show <[<<?R>>]ud>
2324     using u_is_c[symmetric] Rcd "rule=E" by fast
2325   next
2326     fix v
2327     AOT_assume <[λx x =E d]v & [<<?R>>]uv>
2328     AOT_thus <v =E d>
2329     by (metis "β→C"(1) "&E"(1))

```

```

2330     qed
2331 next
2332   fix v
2333   AOT_assume <[ $\lambda x x =_E d$ ]v>
2334   AOT_hence <v =E d>
2335     by (metis " $\beta \rightarrow C$ "(1))
2336   AOT_hence v_is_d: <v = d>
2337     by (metis "=E-simple:2" " $\rightarrow E$ ")
2338   AOT_show < $\exists! u$  ([ $\lambda x x =_E c$ ]u & [«?R»]uv)>
2339   proof (safe intro!: "equiv:1"[THEN " $\equiv E$ "(2)] " $\exists I$ "(2)[where  $\beta=c$ ] "&I"
2340     Oc Ordinary.GEN " $\rightarrow I$ ")
2341     AOT_show <[ $\lambda x x =_E c$ ]c>
2342     by (auto intro!: " $\beta \leftarrow C$ "(1) "cqt:2" "ord=Eequiv:1"[THEN " $\rightarrow E$ ", OF Oc])
2343   next
2344     AOT_show <[«?R»]cv>
2345     using v_is_d[symmetric] Rcd "rule=E" by fast
2346   next
2347     fix u
2348     AOT_assume <[ $\lambda x x =_E c$ ]u & [«?R»]uv>
2349     AOT_thus <u =E c>
2350     by (metis " $\beta \rightarrow C$ "(1) "&E"(1))
2351   qed
2352 next
2353   AOT_show <<«?R» $\downarrow$ >
2354   by "cqt:2"
2355 qed
2356 qed
2357 ultimately AOT_have <a = b>
2358   using "pre-Hume"[unvarify G H, OF eqE_den, OF eqE_den, THEN " $\rightarrow E$ ",
2359     OF "&I", THEN " $\equiv E$ "(2)] by blast
2360 AOT_hence num_a_eq_d: <Numbers(a, [ $\lambda x x =_E d$ ])>
2361   using num_b_eq_d "rule=E" id_sym by fast
2362 AOT_have not_equiv: < $\neg$ [ $\lambda x x =_E c$ ]  $\equiv_E$  [ $\lambda x x =_E d$ ]>
2363 proof (rule "raa-cor:2")
2364   AOT_assume <[ $\lambda x x =_E c$ ]  $\equiv_E$  [ $\lambda x x =_E d$ ]>
2365   AOT_hence <[ $\lambda x x =_E c$ ]c  $\equiv$  [ $\lambda x x =_E d$ ]c>
2366     using eqE[THEN " $\equiv_{df} E$ ", THEN "&E"(2), THEN " $\forall E$ "(2), THEN " $\rightarrow E$ "] Oc by blast
2367   moreover AOT_have <[ $\lambda x x =_E c$ ]c>
2368     by (auto intro!: " $\beta \leftarrow C$ "(1) "cqt:2" "ord=Eequiv:1"[THEN " $\rightarrow E$ ", OF Oc])
2369   ultimately AOT_have <[ $\lambda x x =_E d$ ]c>
2370     using " $\equiv E$ "(1) by blast
2371   AOT_hence <c =E d>
2372     by (rule " $\beta \rightarrow C$ "(1))
2373   AOT_thus <c =E d &  $\neg$ c =E d>
2374     using not_c_eqE_d "&I" by blast
2375 qed
2376 AOT_show < $\exists x \exists G \exists H$  (Numbers(x,G) & Numbers(x,H) &  $\neg G \equiv_E H$ )>
2377   apply (rule " $\exists I$ "(2)[where  $\beta=a$ ])
2378   apply (rule " $\exists I$ "(1)[where  $\tau = \llcorner [\lambda x x =_E c] \gg$ ])
2379   apply (rule " $\exists I$ "(1)[where  $\tau = \llcorner [\lambda x x =_E d] \gg$ ])
2380   by (safe intro!: eqE_den "&I" num_a_eq_c num_a_eq_d not_equiv)
2381 qed
2382
2383 AOT_theorem "num:1": < $\exists x$  Numbers(x,G)> (749.1)
2384   by (AOT_subst <Numbers(x,G)> <[A!]x &  $\forall F$  (x[F]  $\equiv$  [ $\lambda z \mathcal{A}[F]z \approx_E G$ )> for: x)
2385   (auto simp: "numbers[den]"[THEN " $\rightarrow E$ ", OF "cqt:2[const_var]"[axiom_inst]]
2386     "A-objects"[axiom_inst])
2387
2388 AOT_theorem "num:2": < $\exists! x$  Numbers(x,G)> (749.2)
2389   by (AOT_subst <Numbers(x,G)> <[A!]x &  $\forall F$  (x[F]  $\equiv$  [ $\lambda z \mathcal{A}[F]z \approx_E G$ )> for: x)
2390   (auto simp: "numbers[den]"[THEN " $\rightarrow E$ ", OF "cqt:2[const_var]"[axiom_inst]]
2391     "A-objects!")
2392
2393

```

```

2393 AOT_theorem "num-cont:1": (750.1)
2394   <∃x∃G(Numbers(x, G) & ¬□Numbers(x, G))>
2395 proof -
2396   AOT_have <∃F∃G (◇([λz A[F]z] ≈E G & ◇¬[λz A[F]z] ≈E G))>
2397     using "approx-cont:2".
2398   then AOT_obtain F where <∃G (◇([λz A[F]z] ≈E G & ◇¬[λz A[F]z] ≈E G))>
2399     using "∃E"[rotated] by blast
2400   then AOT_obtain G where <◇([λz A[F]z] ≈E G & ◇¬[λz A[F]z] ≈E G)>
2401     using "∃E"[rotated] by blast
2402   AOT_hence ∅: <◇[λz A[F]z] ≈E G> and ζ: <◇¬[λz A[F]z] ≈E G>
2403     using "KBasic2:3"[THEN "→E"] "&E" "4◇"[THEN "→E"] by blast+
2404   AOT_obtain a where <Numbers(a, G)>
2405     using "num:1" "∃E"[rotated] by blast
2406   moreover AOT_have <¬□Numbers(a, G)>
2407   proof (rule "raa-cor:2")
2408     AOT_assume <□Numbers(a, G)>
2409     AOT_hence <□([A!]a & G↓ & ∀F (a[F] ≡ [λz A[F]z] ≈E G))>
2410       by (AOT_subst_def (reverse) numbers)
2411     AOT_hence <□A!a> and <□∀F (a[F] ≡ [λz A[F]z] ≈E G)>
2412       using "KBasic:3"[THEN "≡E"(1)] "&E" by blast+
2413     AOT_hence <∀F □(a[F] ≡ [λz A[F]z] ≈E G)>
2414       using CBF[THEN "→E"] by blast
2415     AOT_hence <□(a[F] ≡ [λz A[F]z] ≈E G)>
2416       using "∀E"(2) by blast
2417     AOT_hence A: <□(a[F] → [λz A[F]z] ≈E G)>
2418       and B: <□([λz A[F]z] ≈E G → a[F])>
2419       using "KBasic:4"[THEN "≡E"(1)] "&E" by blast+
2420     AOT_have <□(¬[λz A[F]z] ≈E G → ¬a[F])>
2421       apply (AOT_subst <¬[λz A[F]z] ≈E G → ¬a[F]> <a[F] → [λz A[F]z] ≈E G>)
2422       using "≡I" "useful-tautologies:4" "useful-tautologies:5" apply presburger
2423       by (fact A)
2424     AOT_hence <◇¬a[F]>
2425       by (metis "KBasic:13" ζ "→E")
2426     AOT_hence <¬a[F]>
2427       by (metis "KBasic:11" "en-eq:2[1]" "≡E"(2) "≡E"(4))
2428     AOT_hence <¬◇a[F]>
2429       by (metis "en-eq:3[1]" "≡E"(4))
2430     moreover AOT_have <◇a[F]>
2431       by (meson B ∅ "KBasic:13" "→E")
2432     ultimately AOT_show <◇a[F] & ¬◇a[F]>
2433       using "&I" by blast
2434   qed
2435
2436   ultimately AOT_have <Numbers(a, G) & ¬□Numbers(a, G)>
2437     using "&I" by blast
2438   AOT_hence <∃G (Numbers(a, G) & ¬□Numbers(a, G))>
2439     by (rule "∃I")
2440   AOT_thus <∃x∃G (Numbers(x, G) & ¬□Numbers(x, G))>
2441     by (rule "∃I")
2442   qed
2443
2444 AOT_theorem "num-cont:2": (750.2)
2445   <Rigid(G) → □∀x(Numbers(x, G) → □Numbers(x, G))>
2446   proof(rule "→I")
2447     AOT_assume <Rigid(G)>
2448     AOT_hence <□∀z([G]z → □[G]z)>
2449       using "df-rigid-rel:1"[THEN "≡dfE", THEN "&E"(2)] by blast
2450     AOT_hence <□□∀z([G]z → □[G]z)> by (metis "S5Basic:6" "≡E"(1))
2451     moreover AOT_have <□□∀z([G]z → □[G]z) → □∀x(Numbers(x, G) → □Numbers(x, G))>
2452     proof(rule RM; safe intro!: "→I" GEN)
2453       AOT_modally_strict {
2454         AOT_have act_den: <[λz A[F]z]↓> for F by "cqt:2[lambda]"
2455         fix x

```

```

2456   AOT_assume G_nec: <□∀z([G]z → □[G]z)>
2457   AOT_hence G_rigid: <Rigid(G)>
2458     using "df-rigid-rel:1"[THEN "≡dfI", OF "&I"] "cqt:2"
2459     by blast
2460   AOT_assume <Numbers(x, G)>
2461   AOT_hence <[A!]x & G↓ & ∀F (x[F] ≡ [λz A[F]z] ≈E G)>
2462     using numbers[THEN "≡dfE"] by blast
2463   AOT_hence Ax: <[A!]x> and <∀F (x[F] ≡ [λz A[F]z] ≈E G)>
2464     using "&E" by blast+
2465   AOT_hence <x[F] ≡ [λz A[F]z] ≈E G> for F
2466     using "∀E"(2) by blast
2467   moreover AOT_have <□([λz A[F]z] ≈E G → □[λz A[F]z] ≈E G)> for F
2468     using "approx-nec:3"[unvarify F, OF act_den, THEN "→E", OF "&I",
2469       OF "actuallyF:2", OF G_rigid].
2470   moreover AOT_have <□(x[F] → □x[F])> for F
2471     by (simp add: RN "pre-en-eq:1[1]")
2472   ultimately AOT_have <□(x[F] ≡ [λz A[F]z] ≈E G)> for F
2473     using "sc-eq-box-box:5" "→E" "qml:2"[axiom_inst] "&I" by meson
2474   AOT_hence <∀F □(x[F] ≡ [λz A[F]z] ≈E G)>
2475     by (rule "∀I")
2476   AOT_hence 1: <□∀F (x[F] ≡ [λz A[F]z] ≈E G)>
2477     using BF[THEN "→E"] by fast
2478   AOT_have <□G↓>
2479     by (simp add: "ex:2:a")
2480   moreover AOT_have <□[A!]x>
2481     using Ax "oa-facts:2" "→E" by blast
2482   ultimately AOT_have <□(A!x & G↓)>
2483     by (metis "KBasic:3" "&I" "≡E"(2))
2484   AOT_hence <□(A!x & G↓ & ∀F (x[F] ≡ [λz A[F]z] ≈E G))>
2485     using 1 "KBasic:3" "&I" "≡E"(2) by fast
2486   AOT_thus <□Numbers(x, G)>
2487     by (AOT_subst_def numbers)
2488 }
2489 qed
2490 ultimately AOT_show <□∀x(Numbers(x,G) → □Numbers(x,G))>
2491   using "→E" by blast
2492 qed
2493
2494 AOT_theorem "num-cont:3":
2495   <□∀x(Numbers(x, [λz A[G]z]) → □Numbers(x, [λz A[G]z]))>
2496   by (rule "num-cont:2"[unvarify G, THEN "→E"];
2497     ("cqt:2[lambda]" | rule "actuallyF:2"))
2498
2499 AOT_theorem "num-uniq": <ιx Numbers(x, G)↓>
2500   using "≡E"(2) "A-Exists:2" "RA[2]" "num:2" by blast
2501
2502 AOT_define num :: <τ ⇒ κs> (<#_> [100] 100)
2503   "num-def:1": <#G =df ιx Numbers(x, G)>
2504
2505 AOT_theorem "num-def:2": <#G↓>
2506   using "num-def:1"[THEN "≡dfI"(1)] "num-uniq" by simp
2507
2508 AOT_theorem "num-can:1":
2509   <#G = ιx(A!x & ∀F (x[F] ≡ [λz A[F]z] ≈E G))>
2510   proof -
2511     AOT_have <□∀x(Numbers(x,G) ≡ [A!]x & ∀F (x[F] ≡ [λz A[F]z] ≈E G))>
2512       by (safe intro!: RN GEN "numbers[den]"[THEN "→E"] "cqt:2")
2513     AOT_hence <ιx Numbers(x, G) = ιx([A!]x & ∀F (x[F] ≡ [λz A[F]z] ≈E G))>
2514       using "num-uniq" "equiv-desc-eq:3"[THEN "→E", OF "&I"] by auto
2515     thus ?thesis
2516       by (rule "≡dfI"(1)[OF "num-def:1", OF "num-uniq"])
2517   qed
2518
2519

```

```

2519 AOT_theorem "num-can:2": <#G =  $\lambda x(A!x \ \& \ \forall F(x[F] \equiv F \approx_E G))$ > (753.2)
2520 proof (rule id_trans[OF "num-can:1"]; rule "equiv-desc-eq:2"[THEN " $\rightarrow E$ "];
2521   safe intro!: "&I" "A-descriptions" GEN "Act-Basic:5"[THEN " $\equiv E$ "](2))
2522   "logic-actual-nec:3"[axiom_inst, THEN " $\equiv E$ "](2)])
2523 AOT_have act_den: < $\vdash_{\square} [\lambda z \mathcal{A}[F]z] \downarrow$ > for F
2524   by "cqt:2"
2525 AOT_have "eq-part:3[terms]": < $\vdash_{\square} F \approx_E G \ \& \ F \approx_E H \rightarrow G \approx_E H$ > for F G H (730.3)
2526   by (metis "&I" "eq-part:2" "eq-part:3" " $\rightarrow I$ " "&E" " $\rightarrow E$ ")
2527 fix x
2528 {
2529   fix F
2530   AOT_have < $\mathcal{A}(F \approx_E [\lambda z \mathcal{A}[F]z])$ >
2531     by (simp add: "actuallyF:1")
2532   moreover AOT_have < $\mathcal{A}((F \approx_E [\lambda z \mathcal{A}[F]z]) \rightarrow ([\lambda z \mathcal{A}[F]z] \approx_E G \equiv F \approx_E G))$ >
2533     by (auto intro!: "RA[2]" " $\rightarrow I$ " " $\equiv I$ "
2534       simp: "eq-part:3"[unvarify G, OF act_den, THEN " $\rightarrow E$ ", OF "&I"]
2535         "eq-part:3[terms]"[unvarify G, OF act_den, THEN " $\rightarrow E$ ", OF "&I"]])
2536   ultimately AOT_have < $\mathcal{A}([\lambda z \mathcal{A}[F]z] \approx_E G \equiv F \approx_E G)$ >
2537     using "logic-actual-nec:2"[axiom_inst, THEN " $\equiv E$ "](1), THEN " $\rightarrow E$ "] by blast
2538
2539   AOT_hence < $\mathcal{A}([\lambda z \mathcal{A}[F]z] \approx_E G \equiv \mathcal{A}F \approx_E G)$ >
2540     by (metis "Act-Basic:5" " $\equiv E$ ")(1)
2541   AOT_hence 0: < $(\mathcal{A}x[F] \equiv \mathcal{A}[\lambda z \mathcal{A}[F]z] \approx_E G) \equiv (\mathcal{A}x[F] \equiv \mathcal{A}F \approx_E G)$ >
2542     by (auto intro!: " $\equiv I$ " " $\rightarrow I$ " elim: " $\equiv E$ ")
2543   AOT_have < $\mathcal{A}(x[F] \equiv [\lambda z \mathcal{A}[F]z] \approx_E G) \equiv (\mathcal{A}x[F] \equiv \mathcal{A}[\lambda z \mathcal{A}[F]z] \approx_E G)$ >
2544     by (simp add: "Act-Basic:5")
2545   also AOT_have < $\dots \equiv (\mathcal{A}x[F] \equiv \mathcal{A}F \approx_E G)$ > using 0.
2546   also AOT_have < $\dots \equiv \mathcal{A}((x[F] \equiv F \approx_E G))$ >
2547     by (meson "Act-Basic:5" " $\equiv E$ ")(6) "oth-class-taut:3:a")
2548   finally AOT_have 0: < $\mathcal{A}(x[F] \equiv [\lambda z \mathcal{A}[F]z] \approx_E G) \equiv \mathcal{A}((x[F] \equiv F \approx_E G))$ >.
2549 } note 0 = this
2550 AOT_have < $\mathcal{A}\forall F(x[F] \equiv [\lambda z \mathcal{A}[F]z] \approx_E G) \equiv \forall F \mathcal{A}(x[F] \equiv [\lambda z \mathcal{A}[F]z] \approx_E G)$ >
2551   using "logic-actual-nec:3" "vdash-properties:1[2]" by blast
2552 also AOT_have < $\dots \equiv \forall F \mathcal{A}((x[F] \equiv F \approx_E G))$ >
2553   apply (safe intro!: " $\equiv I$ " " $\rightarrow I$ " GEN)
2554   using 0 " $\equiv E$ ">(1) " $\equiv E$ ">(2) "rule-ui:3" by blast+
2555 also AOT_have < $\dots \equiv \mathcal{A}(\forall F(x[F] \equiv F \approx_E G))$ >
2556   using " $\equiv E$ ">(6) "logic-actual-nec:3"[axiom_inst] "oth-class-taut:3:a" by fast
2557 finally AOT_have 0: < $\mathcal{A}\forall F(x[F] \equiv [\lambda z \mathcal{A}[F]z] \approx_E G) \equiv \mathcal{A}(\forall F(x[F] \equiv F \approx_E G))$ >.
2558 AOT_have < $\mathcal{A}([A!]x \ \& \ \forall F(x[F] \equiv [\lambda z \mathcal{A}[F]z] \approx_E G)) \equiv$ 
2559   ( $\mathcal{A}!x \ \& \ \mathcal{A}\forall F(x[F] \equiv [\lambda z \mathcal{A}[F]z] \approx_E G))$ >
2560   by (simp add: "Act-Basic:2")
2561 also AOT_have < $\dots \equiv \mathcal{A}[A!]x \ \& \ \mathcal{A}(\forall F(x[F] \equiv F \approx_E G))$ >
2562   using 0 "oth-class-taut:4:f" " $\rightarrow E$ " by blast
2563 also AOT_have < $\dots \equiv \mathcal{A}(A!x \ \& \ \forall F(x[F] \equiv F \approx_E G))$ >
2564   using "Act-Basic:2" " $\equiv E$ ">(6) "oth-class-taut:3:a" by blast
2565 finally AOT_show < $\mathcal{A}([A!]x \ \& \ \forall F(x[F] \equiv [\lambda z \mathcal{A}[F]z] \approx_E G)) \equiv$ 
2566    $\mathcal{A}([A!]x \ \& \ \forall F(x[F] \equiv F \approx_E G))$ >.
2567 qed
2568
2569 AOT_define NaturalCardinal :: < $\tau \Rightarrow \varphi$ > (<NaturalCardinal'('_)'>)
2570   card: <NaturalCardinal(x)  $\equiv_{df} \exists G(x = \#G)$ > (755)
2571
2572 AOT_theorem "natcard-nec": <NaturalCardinal(x)  $\rightarrow \square$ NaturalCardinal(x)> (756)
2573 proof(rule " $\rightarrow I$ ")
2574   AOT_assume <NaturalCardinal(x)>
2575   AOT_hence < $\exists G(x = \#G)$ > using card[THEN " $\equiv_{df} E$ "] by blast
2576   then AOT_obtain G where <x = #G> using " $\exists E$ "[rotated] by blast
2577   AOT_hence < $\square x = \#G$ > by (metis "id-nec:2" " $\rightarrow E$ ")
2578   AOT_hence < $\exists G \square x = \#G$ > by (rule " $\exists I$ ")
2579   AOT_hence < $\square \exists G x = \#G$ > by (metis Buridan " $\rightarrow E$ ")
2580   AOT_thus < $\square$ NaturalCardinal(x)>
2581   by (AOT_subst_def card)

```

```

2582 qed
2583
2584 AOT_act_theorem "hume:1": <Numbers(#G, G)> (757.1)
2585   apply (rule "=dfI"(1)[OF "num-def:1"])
2586   apply (simp add: "num-uniq")
2587   using "num-uniq" "vdash-properties:10" "y-in:3" by blast
2588
2589 AOT_act_theorem "hume:2": <#F = #G  $\equiv$  F  $\approx_E$  G> (757.2)
2590   by (safe intro!: "pre-Hume"[unvarify x y, OF "num-def:2",
2591     OF "num-def:2", THEN " $\rightarrow$ E"] "&I" "hume:1")
2592
2593 AOT_act_theorem "hume:3": <#F = #G  $\equiv$   $\exists$ R (R | : F  $\xrightarrow{1-1}$  onto E G)> (757.3)
2594   using "equi-rem-thm"
2595   apply (AOT_subst (reverse) <R | : F  $\xrightarrow{1-1}$  onto E G>
2596     <R | : F  $\xleftarrow{1-1}$  E G> for: R :: << $\kappa \times \kappa$ >>)
2597   using "equi:3" "hume:2" " $\equiv$ E"(5) " $\equiv$ Df" by blast
2598
2599 AOT_act_theorem "hume:4": <F  $\equiv_E$  G  $\rightarrow$  #F = #G> (757.4)
2600   by (metis "apE-eqE:1" "deduction-theorem" "hume:2" " $\equiv$ E"(2) " $\rightarrow$ E")
2601
2602 AOT_theorem "hume-strict:1": (758.1)
2603   < $\exists$ x (Numbers(x, F) & Numbers(x, G))  $\equiv$  F  $\approx_E$  G>
2604 proof(safe intro!: " $\equiv$ I" " $\rightarrow$ I")
2605   AOT_assume < $\exists$ x (Numbers(x, F) & Numbers(x, G))>
2606   then AOT_obtain a where <Numbers(a, F) & Numbers(a, G)>
2607     using " $\exists$ E"[rotated] by blast
2608   AOT_thus <F  $\approx_E$  G>
2609     using "num-tran:2" " $\rightarrow$ E" by blast
2610 next
2611   AOT_assume 0: <F  $\approx_E$  G>
2612   moreover AOT_obtain b where num_b_F: <Numbers(b, F)>
2613     by (metis "instantiation" "num:1")
2614   moreover AOT_have num_b_G: <Numbers(b, G)>
2615     using calculation "num-tran:1"[THEN " $\rightarrow$ E", THEN " $\equiv$ E"(1)] by blast
2616   ultimately AOT_have <Numbers(b, F) & Numbers(b, G)>
2617     by (safe intro!: "&I")
2618   AOT_thus < $\exists$ x (Numbers(x, F) & Numbers(x, G))>
2619     by (rule " $\exists$ I")
2620 qed
2621
2622 AOT_theorem "hume-strict:2": (758.2)
2623   < $\exists$ x $\exists$ y (Numbers(x, F) &
2624      $\forall$ z(Numbers(z,F)  $\rightarrow$  z = x) &
2625     Numbers(y, G) &
2626      $\forall$ z (Numbers(z, G)  $\rightarrow$  z = y) &
2627     x = y)  $\equiv$ 
2628     F  $\approx_E$  G>
2629 proof(safe intro!: " $\equiv$ I" " $\rightarrow$ I")
2630   AOT_assume < $\exists$ x $\exists$ y (Numbers(x, F) &  $\forall$ z(Numbers(z,F)  $\rightarrow$  z = x) &
2631     Numbers(y, G) &  $\forall$ z (Numbers(z, G)  $\rightarrow$  z = y) & x = y)>
2632   then AOT_obtain x where
2633     < $\exists$ y (Numbers(x, F) &  $\forall$ z(Numbers(z,F)  $\rightarrow$  z = x) & Numbers(y, G) &
2634      $\forall$ z (Numbers(z, G)  $\rightarrow$  z = y) & x = y)>
2635     using " $\exists$ E"[rotated] by blast
2636   then AOT_obtain y where
2637     <Numbers(x, F) &  $\forall$ z(Numbers(z,F)  $\rightarrow$  z = x) & Numbers(y, G) &
2638      $\forall$ z (Numbers(z, G)  $\rightarrow$  z = y) & x = y>
2639     using " $\exists$ E"[rotated] by blast
2640   AOT_hence <Numbers(x, F)> and <Numbers(y,G)> and <x = y>
2641     using "&E" by blast+
2642   AOT_hence <Numbers(y, F) & Numbers(y, G)>
2643     using "&I" "rule=E" by fast
2644   AOT_hence < $\exists$ y (Numbers(y, F) & Numbers(y, G))>

```

```

2645   by (rule "∃I")
2646   AOT_thus <F ≈E G>
2647   using "hume-strict:1"[THEN "≡E"(1)] by blast
2648 next
2649   AOT_assume <F ≈E G>
2650   AOT_hence <∃x (Numbers(x, F) & Numbers(x, G))>
2651   using "hume-strict:1"[THEN "≡E"(2)] by blast
2652   then AOT_obtain x where <Numbers(x, F) & Numbers(x, G)>
2653   using "∃E"[rotated] by blast
2654   moreover AOT_have <∀z (Numbers(z, F) → z = x)>
2655   and <∀z (Numbers(z, G) → z = x)>
2656   using calculation
2657   by (auto intro!: GEN "→I" "pre-Hume"[THEN "→E", OF "&I", THEN "≡E"(2),
2658   rotated 2, OF "eq-part:1"] dest: "&E")
2659   ultimately AOT_have <Numbers(x, F) & ∀z(Numbers(z,F) → z = x) &
2660   Numbers(x, G) & ∀z (Numbers(z, G) → z = x) & x = x>
2661   by (auto intro!: "&I" "id-eq:1" dest: "&E")
2662   AOT_thus <∃x∃y (Numbers(x, F) & ∀z(Numbers(z,F) → z = x) & Numbers(y, G) &
2663   ∀z (Numbers(z, G) → z = y) & x = y)>
2664   by (auto intro!: "∃I")
2665 qed
2666
2667 AOT_theorem unotEu: <¬∃y[λx 0!x & x ≠E x]y> (759)
2668 proof(rule "raa-cor:2")
2669   AOT_assume <∃y[λx 0!x & x ≠E x]y>
2670   then AOT_obtain y where <[λx 0!x & x ≠E x]y>
2671   using "∃E"[rotated] by blast
2672   AOT_hence 0: <0!y & y ≠E y>
2673   by (rule "β→C"(1))
2674   AOT_hence <¬(y =E y)>
2675   using "&E"(2) "≡E"(1) "thm-neg=E" by blast
2676   moreover AOT_have <y =E y>
2677   by (metis 0[THEN "&E"(1)] "ord=Eequiv:1" "→E")
2678   ultimately AOT_show <p & ¬p> for p
2679   by (metis "raa-cor:3")
2680 qed
2681
2682 AOT_define zero :: <κβ> (<0>)
2683 "zero:1": <0 =df #[λx 0!x & x ≠E x]> (760.1)
2684
2685 AOT_theorem "zero:2": <0↓> (760.2)
2686 by (rule "=dfI"(2)[OF "zero:1"]; rule "num-def:2"[unvarify G]; "cqt:2")
2687
2688 AOT_theorem "zero-card": <NaturalCardinal(0)> (761)
2689 apply (rule "=dfI"(2)[OF "zero:1"])
2690 apply (rule "num-def:2"[unvarify G]; "cqt:2")
2691 apply (rule card[THEN "≡dfI"])
2692 apply (rule "∃I"(1)[where τ=<<[λx [0!]x & x ≠E x]>>])
2693 apply (rule "rule=I:1"; rule "num-def:2"[unvarify G]; "cqt:2")
2694 by "cqt:2"
2695
2696 AOT_theorem "eq-num:1": (762.1)
2697 <ℳNumbers(x, G) ≡ Numbers(x, [λz ℳ[G]z])>
2698 proof -
2699   AOT_have act_den: <⊢□ [λz ℳ[F]z]↓> for F by "cqt:2"
2700   AOT_have <□(∃x(Numbers(x, G) & Numbers(x, [λz ℳ[G]z])) ≡ G ≈E [λz ℳ[G]z])>
2701   using "hume-strict:1"[unvarify G, OF act_den, THEN RN].
2702   AOT_hence <ℳ(∃x(Numbers(x, G) & Numbers(x, [λz ℳ[G]z])) ≡ G ≈E [λz ℳ[G]z])>
2703   using "nec-imp-act"[THEN "→E"] by fast
2704   AOT_hence <ℳ(∃x(Numbers(x, G) & Numbers(x, [λz ℳ[G]z])))>
2705   using "actuallyF:1" "Act-Basic:5" "≡E"(1) "≡E"(2) by fast
2706   AOT_hence <∃x ℳ((Numbers(x, G) & Numbers(x, [λz ℳ[G]z])))>
2707   by (metis "Act-Basic:10" "intro-elim:3:a")

```



```

2708 then AOT_obtain a where <A(Numbers(a, G) & Numbers(a, [λz A[G]z]))>
2709   using "∃E"[rotated] by blast
2710 AOT_hence act_a_num_G: <ANumbers(a, G)>
2711   and act_a_num_actG: <ANumbers(a, [λz A[G]z])>
2712   using "Act-Basic:2" "&E" "≡E"(1) by blast+
2713 AOT_hence num_a_act_g: <Numbers(a, [λz A[G]z])>
2714   using "num-cont:2"[unvarify G, OF act_den, THEN "→E", OF "actuallyF:2",
2715     THEN CBF[THEN "→E"], THEN "∀E"(2)]
2716   by (metis "≡E"(1) "sc-eq-fur:2" "vdash-properties:6")
2717 AOT_have 0: <⊢□ Numbers(x, G) & Numbers(y, G) → x = y> for y
2718   using "pre-Hume"[THEN "→E", THEN "≡E"(2), rotated, OF "eq-part:1"]
2719   "→I" by blast
2720 show ?thesis
2721 proof(safe intro!: "≡I" "→I")
2722   AOT_assume <ANumbers(x, G)>
2723   AOT_hence <Ax = a>
2724     using 0[THEN "RA[2]", THEN "act-cond"[THEN "→E"], THEN "→E",
2725       OF "Act-Basic:2"[THEN "≡E"(2)], OF "&I"]
2726     act_a_num_G by blast
2727   AOT_hence <x = a> by (metis "id-act:1" "≡E"(2))
2728   AOT_hence <a = x> using id_sym by auto
2729   AOT_thus <Numbers(x, [λz A[G]z])>
2730     using "rule=E" num_a_act_g by fast
2731 next
2732   AOT_assume <Numbers(x, [λz A[G]z])>
2733   AOT_hence <a = x>
2734     using "pre-Hume"[unvarify G H, THEN "→E", OF act_den, OF act_den, OF "&I",
2735       OF num_a_act_g, THEN "≡E"(2)]
2736     "eq-part:1"[unvarify F, OF act_den] by blast
2737   AOT_thus <ANumbers(x, G)>
2738     using act_a_num_G "rule=E" by fast
2739 qed
2740 qed
2741
2742 AOT_theorem "eq-num:2": <Numbers(x, [λz A[G]z]) ≡ x = #G> (762.2)
2743 proof -
2744   AOT_have 0: <⊢□ x = λx Numbers(x, G) ≡ ∀y (Numbers(y, [λz A[G]z]) ≡ y = x)> for x
2745     by (AOT_subst (reverse) <Numbers(x, [λz A[G]z])> <ANumbers(x, G)> for: x)
2746     (auto simp: "eq-num:1" descriptions[axiom_inst])
2747   AOT_have <#G = λx Numbers(x, G) ≡ ∀y (Numbers(y, [λz A[G]z]) ≡ y = #G)>
2748     using 0[unvarify x, OF "num-def:2"].
2749   moreover AOT_have <#G = λx Numbers(x, G)>
2750     using "num-def:1" "num-uniq" "rule-id-df:1" by blast
2751   ultimately AOT_have <∀y (Numbers(y, [λz A[G]z]) ≡ y = #G)>
2752     using "≡E" by blast
2753   thus ?thesis using "∀E"(2) by blast
2754 qed
2755
2756 AOT_theorem "eq-num:3": <Numbers(#G, [λy A[G]y])> (762.3)
2757 proof -
2758   AOT_have <#G = #G>
2759     by (simp add: "rule=I:1" "num-def:2")
2760   thus ?thesis
2761     using "eq-num:2"[unvarify x, OF "num-def:2", THEN "≡E"(2)] by blast
2762 qed
2763
2764 AOT_theorem "eq-num:4": (762.4)
2765   <A!#G & ∀F (#G[F] ≡ [λz A[F]z] ≈E [λz A[G]z])>
2766   by (auto intro!: "&I" "eq-num:3"[THEN numbers[THEN "≡dfE"],
2767     THEN "&E"(1), THEN "&E"(1)]
2768     "eq-num:3"[THEN numbers[THEN "≡dfE"], THEN "&E"(2)])
2769
2770 AOT_theorem "eq-num:5": <#G[G]> (762.5)

```



```

2771   by (auto intro!: "eq-num:4"[THEN "&E"(2), THEN "∀E"(2), THEN "≡E"(2)]
2772       "eq-part:1"[unvarify F] simp: "cqt:2")
2773
2774 AOT_theorem "eq-num:6": <Numbers(x, G) → NaturalCardinal(x)> (762.6)
2775 proof(rule "→I")
2776   AOT_have act_den: <⊢□ [λz A[F]z]↓> for F
2777   by "cqt:2"
2778   AOT_obtain F where <Rigidifies(F, G)>
2779   by (metis "instantiation" "rigid-der:3")
2780   AOT_hence ∅: <Rigid(F)> and <∀x([F]x ≡ [G]x)>
2781   using "df-rigid-rel:2"[THEN "≡dfE", THEN "&E"(2)]
2782   "df-rigid-rel:2"[THEN "≡dfE", THEN "&E"(1)]
2783   by blast+
2784   AOT_hence <F ≡E G>
2785   by (auto intro!: eqE[THEN "≡dfI"] "&I" "cqt:2" GEN "→I" elim: "∀E"(2))
2786   moreover AOT_assume <Numbers(x, G)>
2787   ultimately AOT_have <Numbers(x, F)>
2788   using "num-tran:3"[THEN "→E", THEN "≡E"(2)] by blast
2789   moreover AOT_have <F ≈E [λz A[F]z]>
2790   using ∅ "approx-nec:1" "→E" by blast
2791   ultimately AOT_have <Numbers(x, [λz A[F]z])>
2792   using "num-tran:1"[unvarify H, OF act_den, THEN "→E", THEN "≡E"(1)] by blast
2793   AOT_hence <x = #F>
2794   using "eq-num:2"[THEN "≡E"(1)] by blast
2795   AOT_hence <∃F x = #F>
2796   by (rule "∃I")
2797   AOT_thus <NaturalCardinal(x)>
2798   using card[THEN "≡dfI"] by blast
2799 qed
2800
2801 AOT_theorem "eq-df-num": <∃G (x = #G) ≡ ∃G (Numbers(x,G))> (763)
2802 proof(safe intro!: "≡I" "→I")
2803   AOT_assume <∃G (x = #G)>
2804   then AOT_obtain P where <x = #P>
2805   using "∃E"[rotated] by blast
2806   AOT_hence <Numbers(x, [λz A[P]z])>
2807   using "eq-num:2"[THEN "≡E"(2)] by blast
2808   moreover AOT_have <[λz A[P]z]↓> by "cqt:2"
2809   ultimately AOT_show <∃G(Numbers(x,G))> by (rule "∃I")
2810 next
2811   AOT_assume <∃G (Numbers(x,G))>
2812   then AOT_obtain Q where <Numbers(x,Q)>
2813   using "∃E"[rotated] by blast
2814   AOT_hence <NaturalCardinal(x)>
2815   using "eq-num:6"[THEN "→E"] by blast
2816   AOT_thus <∃G (x = #G)>
2817   using card[THEN "≡dfE"] by blast
2818 qed
2819
2820 AOT_theorem "card-en": <NaturalCardinal(x) → ∀F(x[F] ≡ x = #F)> (764)
2821 proof(rule "→I"; rule GEN)
2822   AOT_have act_den: <⊢□ [λz A[F]z]↓> for F by "cqt:2"
2823   fix F
2824   AOT_assume <NaturalCardinal(x)>
2825   AOT_hence <∃F x = #F>
2826   using card[THEN "≡dfE"] by blast
2827   then AOT_obtain P where x_def: <x = #P>
2828   using "∃E"[rotated] by blast
2829   AOT_hence num_x_act_P: <Numbers(x, [λz A[P]z])>
2830   using "eq-num:2"[THEN "≡E"(2)] by blast
2831   AOT_have <#P[F] ≡ [λz A[F]z] ≈E [λz A[P]z]>
2832   using "eq-num:4"[THEN "&E"(2), THEN "∀E"(2)] by blast
2833   AOT_hence <x[F] ≡ [λz A[F]z] ≈E [λz A[P]z]>

```

```

2834   using x_def[symmetric] "rule=E" by fast
2835   also AOT_have <...  $\equiv$  Numbers(x, [ $\lambda z \mathcal{A}[F]z$ ])>
2836   using "num-tran:1"[unvarify G H, OF act_den, OF act_den]
2837   using "num-tran:2"[unvarify G H, OF act_den, OF act_den]
2838   by (metis "&I" "deduction-theorem" " $\equiv$ I" " $\equiv$ E"(2) num_x_act_P)
2839   also AOT_have <...  $\equiv$  x = #F>
2840   using "eq-num:2" by blast
2841   finally AOT_show <x[F]  $\equiv$  x = #F>.
2842 qed
2843
2844 AOT_theorem "OF:1": < $\neg \exists u [F]u \equiv$  Numbers(0, F)> (765.1)
2845 proof -
2846   AOT_have unotEu_act_ord: < $\neg \exists v [\lambda x 0!x \ \& \ \mathcal{A}x \neq_E x]v$ >
2847   proof(rule "raa-cor:2")
2848     AOT_assume < $\exists v [\lambda x 0!x \ \& \ \mathcal{A}x \neq_E x]v$ >
2849     then AOT_obtain y where <[ $\lambda x 0!x \ \& \ \mathcal{A}x \neq_E x]y$ >
2850     using " $\exists$ E"[rotated] "&E" by blast
2851     AOT_hence 0: <0!y &  $\mathcal{A}y \neq_E y$ >
2852     by (rule " $\beta \rightarrow C$ "(1))
2853     AOT_have < $\mathcal{A}\neg(y =_E y)$ >
2854     apply (AOT_subst < $\neg(y =_E y)$ > <y  $\neq_E$  y>)
2855     apply (meson " $\equiv$ E"(2) "Commutativity of  $\equiv$ " "thm-neg=E")
2856     by (fact 0[THEN "&E"(2)])
2857     AOT_hence < $\neg(y =_E y)$ >
2858     by (metis " $\neg\neg$ I" "Act-Sub:1" "id-act2:1" " $\equiv$ E"(4))
2859     moreover AOT_have <y =E y>
2860     by (metis 0[THEN "&E"(1)] "ord=Eequiv:1" " $\rightarrow$ E")
2861     ultimately AOT_show <p &  $\neg$ p> for p
2862     by (metis "raa-cor:3")
2863   qed
2864   AOT_have <Numbers(0, [ $\lambda y \mathcal{A}[\lambda x 0!x \ \& \ x \neq_E x]y$ ])>
2865   apply (rule "=dfI"(2)[OF "zero:1"])
2866   apply (rule "num-def:2"[unvarify G]; "cqt:2")
2867   apply (rule "eq-num:3"[unvarify G])
2868   by "cqt:2[lambda]"
2869   AOT_hence numbers0: <Numbers(0, [ $\lambda x [0!]x \ \& \ \mathcal{A}x \neq_E x$ ])>
2870   proof (rule "num-tran:3"[unvarify x G H, THEN " $\rightarrow$ E", THEN " $\equiv$ E"(1), rotated 4])
2871     AOT_show <[ $\lambda y \mathcal{A}[\lambda x 0!x \ \& \ x \neq_E x]y \equiv_E [\lambda x [0!]x \ \& \ \mathcal{A}x \neq_E x]$ >
2872     proof (safe intro!: eqE[THEN "=dfI"] "&I" Ordinary.GEN " $\rightarrow$ I" "cqt:2")
2873       fix u
2874       AOT_have <[ $\lambda y \mathcal{A}[\lambda x 0!x \ \& \ x \neq_E x]y]u \equiv \mathcal{A}[\lambda x 0!x \ \& \ x \neq_E x]u$ >
2875       by (rule "beta-C-meta"[THEN " $\rightarrow$ E"]; "cqt:2[lambda]")
2876       also AOT_have <...  $\equiv \mathcal{A}(0!u \ \& \ u \neq_E u)$ >
2877       apply (AOT_subst <[ $\lambda x 0!x \ \& \ x \neq_E x]u$ > <0!u & u  $\neq_E$  u>)
2878       apply (rule "beta-C-meta"[THEN " $\rightarrow$ E"]; "cqt:2[lambda]")
2879       by (simp add: "oth-class-taut:3:a")
2880       also AOT_have <...  $\equiv (\mathcal{A}0!u \ \& \ \mathcal{A}u \neq_E u)$ >
2881       by (simp add: "Act-Basic:2")
2882       also AOT_have <...  $\equiv (0!u \ \& \ \mathcal{A}u \neq_E u)$ >
2883       by (metis Ordinary. $\psi$  "&I" "&E"(2) " $\rightarrow$ I" " $\equiv$ I" " $\equiv$ E"(1) "oa-facts:7")
2884       also AOT_have <...  $\equiv [\lambda x [0!]x \ \& \ \mathcal{A}x \neq_E x]u$ >
2885       by (rule "beta-C-meta"[THEN " $\rightarrow$ E", symmetric]; "cqt:2[lambda]")
2886       finally AOT_show <[ $\lambda y \mathcal{A}[\lambda x 0!x \ \& \ x \neq_E x]y]u \equiv [\lambda x [0!]x \ \& \ \mathcal{A}x \neq_E x]u$ >.
2887     qed
2888   qed(fact "zero:2" | "cqt:2")+
2889   show ?thesis
2890   proof(safe intro!: " $\equiv$ I" " $\rightarrow$ I")
2891     AOT_assume < $\neg \exists u [F]u$ >
2892     moreover AOT_have < $\neg \exists v [\lambda x [0!]x \ \& \ \mathcal{A}x \neq_E x]v$ >
2893     using unotEu_act_ord.
2894     ultimately AOT_have 0: <F  $\approx_E [\lambda x [0!]x \ \& \ \mathcal{A}x \neq_E x]$ >
2895     by (rule "empty-approx:1"[unvarify H, THEN " $\rightarrow$ E", rotated, OF "&I"]) "cqt:2"
2896     AOT_thus <Numbers(0, F)>

```

```

2897     by (rule "num-tran:1"[unvarify x H, THEN "→E",
2898         THEN "≡E"(2), rotated, rotated])
2899     (fact "zero:2" numbers0 | "cqt:2[lambda]")+
2900 next
2901   AOT_assume <Numbers(0, F)>
2902   AOT_hence 1: <F ≈E [λx [0!]x & Ax ≠E x]>
2903     by (rule "num-tran:2"[unvarify x H, THEN "→E", rotated 2, OF "&I"])
2904     (fact numbers0 "zero:2" | "cqt:2[lambda]")+
2905   AOT_show <¬∃u [F]u>
2906   proof(rule "raa-cor:2")
2907     AOT_have 0: <[λx [0!]x & Ax ≠E x]↓> by "cqt:2[lambda]"
2908     AOT_assume <∃u [F]u>
2909     AOT_hence <¬(F ≈E [λx [0!]x & Ax ≠E x])>
2910       by (rule "empty-approx:2"[unvarify H, OF 0, THEN "→E", OF "&I"])
2911       (rule unotEu_act_ord)
2912     AOT_thus <F ≈E [λx [0!]x & Ax ≠E x] & ¬(F ≈E [λx [0!]x & Ax ≠E x])>
2913       using 1 "&I" by blast
2914   qed
2915   qed
2916   qed
2917
2918 AOT_theorem "OF:2": <¬∃u A[F]u ≡ #F = 0> (765.2)
2919 proof(rule "≡I"; rule "→I")
2920   AOT_assume 0: <¬∃u A[F]u>
2921   AOT_have <¬∃u [λz A[F]z]u>
2922   proof(rule "raa-cor:2")
2923     AOT_assume <∃u [λz A[F]z]u>
2924     then AOT_obtain u where <[λz A[F]z]u>
2925       using "Ordinary.∃E"[rotated] by blast
2926     AOT_hence <A[F]u>
2927       by (metis "betaC:1:a")
2928     AOT_hence <∃u A[F]u>
2929       by (rule "Ordinary.∃I")
2930     AOT_thus <∃u A[F]u & ¬∃u A[F]u>
2931       using 0 "&I" by blast
2932   qed
2933   AOT_hence <Numbers(0, [λz A[F]z])>
2934     by (safe intro!: "OF:1"[unvarify F, THEN "≡E"(1)]) "cqt:2"
2935   AOT_hence <0 = #F>
2936     by (rule "eq-num:2"[unvarify x, OF "zero:2", THEN "≡E"(1)])
2937   AOT_thus <#F = 0> using id_sym by blast
2938 next
2939   AOT_assume <#F = 0>
2940   AOT_hence <0 = #F> using id_sym by blast
2941   AOT_hence <Numbers(0, [λz A[F]z])>
2942     by (rule "eq-num:2"[unvarify x, OF "zero:2", THEN "≡E"(2)])
2943   AOT_hence 0: <¬∃u [λz A[F]z]u>
2944     by (safe intro!: "OF:1"[unvarify F, THEN "≡E"(2)]) "cqt:2"
2945   AOT_show <¬∃u A[F]u>
2946   proof(rule "raa-cor:2")
2947     AOT_assume <∃u A[F]u>
2948     then AOT_obtain u where <A[F]u>
2949       using "Ordinary.∃E"[rotated] by meson
2950     AOT_hence <[λz A[F]z]u>
2951       by (auto intro!: "β←C" "cqt:2")
2952     AOT_hence <∃u [λz A[F]z]u>
2953       using "Ordinary.∃I" by blast
2954     AOT_thus <∃u [λz A[F]z]u & ¬∃u [λz A[F]z]u>
2955       using "&I" 0 by blast
2956   qed
2957   qed
2958
2959 AOT_theorem "OF:3": <□¬∃u [F]u → #F = 0> (765.3)

```

```

2960 proof(rule "→I")
2961   AOT_assume <□¬∃u [F]u>
2962   AOT_hence 0: <¬◇∃u [F]u>
2963     using "KBasic2:1" "≡E"(1) by blast
2964   AOT_have <¬∃u [λz A[F]z]u>
2965   proof(rule "raa-cor:2")
2966     AOT_assume <∃u [λz A[F]z]u>
2967     then AOT_obtain u where <[λz A[F]z]u>
2968       using "Ordinary.∃E"[rotated] by blast
2969     AOT_hence <A[F]u>
2970       by (metis "betaC:1:a")
2971     AOT_hence <◇[F]u>
2972       by (metis "Act-Sub:3" "→E")
2973     AOT_hence <∃u ◇[F]u>
2974       by (rule "Ordinary.∃I")
2975     AOT_hence <◇∃u [F]u>
2976       using "Ordinary.res-var-bound-reas[CBF◇]"[THEN "→E"] by blast
2977     AOT_thus <◇∃u [F]u & ¬◇∃u [F]u>
2978       using 0 "&I" by blast
2979   qed
2980   AOT_hence <Numbers(0,[λz A[F]z])>
2981     by (safe intro!: "OF:1"[unvarify F, THEN "≡E"(1)]) "cqt:2"
2982   AOT_hence <0 = #F>
2983     by (rule "eq-num:2"[unvarify x, OF "zero:2", THEN "≡E"(1)])
2984   AOT_thus <#F = 0> using id_sym by blast
2985 qed
2986
2987 AOT_theorem "OF:4": <w ⊨ ¬∃u [F]u ≡ #[F]w = 0> (765.4)
2988 proof (rule "rule-id-df:2:b"[OF "w-index", where τ1τn="(_,_)"] simplified)
2989   AOT_show <[λx1...xn w ⊨ [F]x1...xn]↓>
2990     by (simp add: "w-rel:3")
2991 next
2992   AOT_show <w ⊨ ¬∃u [F]u ≡ #[λx w ⊨ [F]x] = 0>
2993   proof (rule "≡I"; rule "→I")
2994     AOT_assume <w ⊨ ¬∃u [F]u>
2995     AOT_hence 0: <¬w ⊨ ∃u [F]u>
2996       using "coherent:1"[unvarify p, OF "log-prop-prop:2", THEN "≡E"(1)] by blast
2997     AOT_have <¬∃u A[λx w ⊨ [F]x]u>
2998     proof(rule "raa-cor:2")
2999       AOT_assume <∃u A[λx w ⊨ [F]x]u>
3000       then AOT_obtain u where <A[λx w ⊨ [F]x]u>
3001         using "Ordinary.∃E"[rotated] by meson
3002       AOT_hence <Aw ⊨ [F]u>
3003         by (AOT_subst (reverse) <w ⊨ [F]u> <[λx w ⊨ [F]x]u>;
3004           safe intro!: "beta-C-meta"[THEN "→E"] "w-rel:1"[THEN "→E"])
3005         "cqt:2"
3006       AOT_hence 1: <w ⊨ [F]u>
3007         using "rigid-truth-at:4"[unvarify p, OF "log-prop-prop:2", THEN "≡E"(1)]
3008         by blast
3009       AOT_have <□([F]u → ∃u [F]u)>
3010         using "Ordinary.∃I" "→I" RN by simp
3011       AOT_hence <w ⊨ ([F]u → ∃u [F]u)>
3012         using "fund:2"[unvarify p, OF "log-prop-prop:2", THEN "≡E"(1)]
3013         "PossibleWorld.∀E" by fast
3014       AOT_hence <w ⊨ ∃u [F]u>
3015         using 1 "conj-dist-w:2"[unvarify p q, OF "log-prop-prop:2",
3016           OF "log-prop-prop:2", THEN "≡E"(1),
3017           THEN "→E"] by blast
3018       AOT_thus <w ⊨ ∃u [F]u & ¬w ⊨ ∃u [F]u>
3019         using 0 "&I" by blast
3020     qed
3021   AOT_thus <#[λx w ⊨ [F]x] = 0>
3022     by (safe intro!: "OF:2"[unvarify F, THEN "≡E"(1)] "w-rel:1"[THEN "→E"])

```

```

3023     "cqt:2"
3024 next
3025 AOT_assume <#[ $\lambda x w \models [F]x = 0$ >
3026 AOT_hence 0: < $\neg \exists u \mathcal{A}[\lambda x w \models [F]x]u$ >
3027   by (safe intro!: "OF:2"[unvarify F, THEN "≡E"(2)] "w-rel:1"[THEN "→E"])
3028     "cqt:2"
3029 AOT_have < $\neg w \models \exists u [F]u$ >
3030 proof (rule "raa-cor:2")
3031   AOT_assume < $w \models \exists u [F]u$ >
3032   AOT_hence < $\exists x w \models (0!x \ \& \ [F]x)$ >
3033     using "conj-dist-w:6"[THEN "≡E"(1)] by fast
3034   then AOT_obtain x where < $w \models (0!x \ \& \ [F]x)$ >
3035     using "∃E"[rotated] by blast
3036   AOT_hence < $w \models 0!x$ > and Fx_in_w: < $w \models [F]x$ >
3037     using "conj-dist-w:1"[unvarify p q] "≡E"(1) "log-prop-prop:2"
3038       "&E" by blast+
3039   AOT_hence < $\langle 0!x \rangle$ >
3040     using "fund:1"[unvarify p, OF "log-prop-prop:2", THEN "≡E"(2)]
3041       "PossibleWorld.∃I" by simp
3042   AOT_hence ord_x: < $0!x$ >
3043     using "oa-facts:3"[THEN "→E"] by blast
3044   AOT_have < $\mathcal{A}w \models [F]x$ >
3045     using "rigid-truth-at:4"[unvarify p, OF "log-prop-prop:2", THEN "≡E"(2)]
3046       Fx_in_w by blast
3047   AOT_hence < $\mathcal{A}[\lambda x w \models [F]x]x$ >
3048     by (AOT_subst < $[\lambda x w \models [F]x]x$ > < $w \models [F]x$ >;
3049       safe intro!: "beta-C-meta"[THEN "→E"] "w-rel:1"[THEN "→E"]) "cqt:2"
3050   AOT_hence < $0!x \ \& \ \mathcal{A}[\lambda x w \models [F]x]x$ >
3051     using ord_x "&I" by blast
3052   AOT_hence < $\exists x (0!x \ \& \ \mathcal{A}[\lambda x w \models [F]x]x)$ >
3053     using "∃I" by fast
3054   AOT_thus < $\exists u (\mathcal{A}[\lambda x w \models [F]x]u) \ \& \ \neg \exists u \mathcal{A}[\lambda x w \models [F]x]u$ >
3055     using 0 "&I" by blast
3056   qed
3057   AOT_thus < $w \models \neg \exists u [F]u$ >
3058     using "coherent:1"[unvarify p, OF "log-prop-prop:2", THEN "≡E"(2)] by blast
3059   qed
3060 qed
3061
3062 AOT_act_theorem "zero=:1": (766.1)
3063   <NaturalCardinal(x) →  $\forall F (x[F] \equiv \text{Numbers}(x, F))$ >
3064 proof(safe intro!: "→I" GEN)
3065   fix F
3066   AOT_assume <NaturalCardinal(x)>
3067   AOT_hence < $\forall F (x[F] \equiv x = \#F)$ >
3068     by (metis "card-en" "→E")
3069   AOT_hence 1: < $x[F] \equiv x = \#F$ >
3070     using "∀E"(2) by blast
3071   AOT_have 2: < $x[F] \equiv x = \iota y(\text{Numbers}(y, F))$ >
3072     by (rule "num-def:1"[THEN "=defE"(1)])
3073     (auto simp: 1 "num-uniq")
3074   AOT_have < $x = \iota y(\text{Numbers}(y, F)) \rightarrow \text{Numbers}(x, F)$ >
3075     using "y-in:1" by blast
3076   moreover AOT_have < $\text{Numbers}(x, F) \rightarrow x = \iota y(\text{Numbers}(y, F))$ >
3077   proof(rule "→I")
3078     AOT_assume 1: <Numbers(x, F)>
3079     moreover AOT_obtain z where z_prop: < $\forall y (\text{Numbers}(y, F) \rightarrow y = z)$ >
3080       using "num:2"[THEN "uniqueness:1"[THEN "=defE"]] "∃E"[rotated] "&E" by blast
3081     ultimately AOT_have < $x = z$ >
3082       using "∀E"(2) "→E" by blast
3083     AOT_hence < $\forall y (\text{Numbers}(y, F) \rightarrow y = x)$ >
3084       using z_prop "rule=E" id_sym by fast
3085     AOT_thus < $x = \iota y(\text{Numbers}(y, F))$ >

```

```

3086     by (rule hintikka[THEN "≡E"(2), OF "&I", rotated])
3087     (fact 1)
3088   qed
3089   ultimately AOT_have <x =  $\iota y(\text{Numbers}(y, F)) \equiv \text{Numbers}(x, F)$ >
3090     by (metis "≡I")
3091   AOT_thus <x[F]  $\equiv \text{Numbers}(x, F)$ >
3092     using 2 by (metis "≡E"(5))
3093   qed
3094
3095   AOT_act_theorem "zero=:2": <0[F]  $\equiv \neg\exists u[F]u$ > (766.2)
3096   proof -
3097     AOT_have <0[F]  $\equiv \text{Numbers}(0, F)$ >
3098       using "zero=:1"[unvarify x, OF "zero:2", THEN " $\rightarrow$ E",
3099         OF "zero-card", THEN " $\forall$ E"(2)].
3100     also AOT_have <...  $\equiv \neg\exists u[F]u$ >
3101       using "OF:1"[symmetric].
3102     finally show ?thesis.
3103   qed
3104
3105   AOT_act_theorem "zero=:3": < $\neg\exists u[F]u \equiv \#F = 0$ > (766.3)
3106   proof -
3107     AOT_have < $\neg\exists u[F]u \equiv 0[F]$ > using "zero=:2"[symmetric].
3108     also AOT_have <...  $\equiv 0 = \#F$ >
3109       using "card-en"[unvarify x, OF "zero:2", THEN " $\rightarrow$ E",
3110         OF "zero-card", THEN " $\forall$ E"(2)].
3111     also AOT_have <...  $\equiv \#F = 0$ >
3112       by (simp add: "deduction-theorem" id_sym "≡I")
3113     finally show ?thesis.
3114   qed
3115
3116   AOT_define Hereditary :: < $\tau \Rightarrow \tau \Rightarrow \varphi$ > (<Hereditary'(_,_)>)
3117     "hered:1": (767.1)
3118     <Hereditary(F, R)  $\equiv_{df} R\downarrow \ \& \ F\downarrow \ \& \ \forall x\forall y([R]xy \rightarrow ([F]x \rightarrow [F]y))$ >
3119
3120   AOT_theorem "hered:2": (767.2)
3121     < $[\lambda xy \ \forall F((\forall z([R]xz \rightarrow [F]z) \ \& \ \text{Hereditary}(F,R)) \rightarrow [F]y)]\downarrow$ >
3122     by "cqt:2[lambda]"
3123
3124   AOT_define StrongAncestral :: < $\tau \Rightarrow \Pi$ > (<_*>)
3125     "ances-df": (768)
3126     < $R^* =_{df} [\lambda xy \ \forall F((\forall z([R]xz \rightarrow [F]z) \ \& \ \text{Hereditary}(F,R)) \rightarrow [F]y)]$ >
3127
3128   AOT_theorem "ances": (769)
3129     < $[R^*]xy \equiv \forall F((\forall z([R]xz \rightarrow [F]z) \ \& \ \text{Hereditary}(F,R)) \rightarrow [F]y)$ >
3130     apply (rule "=dfI"(1)[OF "ances-df"])
3131     apply "cqt:2[lambda]"
3132     apply (rule "beta-C-meta"[THEN " $\rightarrow$ E", OF "hered:2", unvarify  $\nu_1\nu_n$ ,
3133       where  $\tau = \langle \_, \_ \rangle$ , simplified])
3134     by (simp add: "&I" "ex:1:a" prod_denotesI "rule-ui:3")
3135
3136   AOT_theorem "anc-her:1": (770.1)
3137     < $[R]xy \rightarrow [R^*]xy$ >
3138   proof (safe intro!: " $\rightarrow$ I" ances[THEN "≡E"(2)] GEN)
3139     fix F
3140     AOT_assume < $\forall z([R]xz \rightarrow [F]z) \ \& \ \text{Hereditary}(F, R)$ >
3141     AOT_hence < $[R]xy \rightarrow [F]y$ >
3142       using " $\forall$ E"(2) "&E" by blast
3143     moreover AOT_assume < $[R]xy$ >
3144     ultimately AOT_show < $[F]y$ >
3145       using " $\rightarrow$ E" by blast
3146   qed
3147
3148   AOT_theorem "anc-her:2": (770.2)

```

```

3149   <([R*]xy & ∀z([R]xz → [F]z) & Hereditary(F,R)) → [F]y>
3150 proof(rule "→I"; (frule "&E"(1); drule "&E"(2))+)
3151   AOT_assume <[R*]xy>
3152   AOT_hence <(∀z([R]xz → [F]z) & Hereditary(F,R)) → [F]y>
3153     using ances[THEN "≡E"(1)] "∀E"(2) by blast
3154   moreover AOT_assume <∀z([R]xz → [F]z)>
3155   moreover AOT_assume <Hereditary(F,R)>
3156   ultimately AOT_show <[F]y>
3157     using "→E" "&I" by blast
3158 qed
3159
3160 AOT_theorem "anc-her:3": (770.3)
3161   <([F]x & [R*]xy & Hereditary(F, R)) → [F]y>
3162 proof(rule "→I"; (frule "&E"(1); drule "&E"(2))+)
3163   AOT_assume 1: <[F]x>
3164   AOT_assume 2: <Hereditary(F, R)>
3165   AOT_hence 3: <∀x ∀y ([R]xy → ([F]x → [F]y))>
3166     using "hered:1"[THEN "≡dfE"] "&E" by blast
3167   AOT_have <∀z ([R]xz → [F]z)>
3168   proof (rule GEN; rule "→I")
3169     fix z
3170     AOT_assume <[R]xz>
3171     moreover AOT_have <[R]xz → ([F]x → [F]z)>
3172       using 3 "∀E"(2) by blast
3173     ultimately AOT_show <[F]z>
3174       using 1 "→E" by blast
3175   qed
3176   moreover AOT_assume <[R*]xy>
3177   ultimately AOT_show <[F]y>
3178     by (auto intro!: 2 "anc-her:2"[THEN "→E"] "&I")
3179 qed
3180
3181 AOT_theorem "anc-her:4": <([R]xy & [R*]yz) → [R*]xz> (770.4)
3182 proof(rule "→I"; frule "&E"(1); drule "&E"(2))
3183   AOT_assume 0: <[R*]yz> and 1: <[R]xy>
3184   AOT_show <[R*]xz>
3185   proof(safe intro!: ances[THEN "≡E"(2)] GEN "&I" "→I";
3186     frule "&E"(1); drule "&E"(2))
3187     fix F
3188     AOT_assume <∀z ([R]xz → [F]z)>
3189     AOT_hence 1: <[F]y>
3190       using 1 "∀E"(2) "→E" by blast
3191     AOT_assume 2: <Hereditary(F,R)>
3192     AOT_show <[F]z>
3193       by (rule "anc-her:3"[THEN "→E"]; auto intro!: "&I" 1 2 0)
3194   qed
3195 qed
3196
3197 AOT_theorem "anc-her:5": <[R*]xy → ∃z [R]zy> (770.5)
3198 proof (rule "→I")
3199   AOT_have 0: <[λy ∃x [R]xy]↓> by "cqt:2"
3200   AOT_assume 1: <[R*]xy>
3201   AOT_have <[λy∃x [R]xy]y>
3202   proof(rule "anc-her:2"[unvarify F, OF 0, THEN "→E"];
3203     safe intro!: "&I" GEN "→I" "hered:1"[THEN "≡dfI"] "cqt:2" 0)
3204     AOT_show <[R*]xy> using 1.
3205   next
3206     fix z
3207     AOT_assume <[R]xz>
3208     AOT_hence <∃x [R]xz> by (rule "∃I")
3209     AOT_thus <[λy∃x [R]xy]z>
3210       by (auto intro!: "β←C"(1) "cqt:2")
3211   next

```

```

3212   fix x y
3213   AOT_assume <[R]xy>
3214   AOT_hence <∃x [R]xy> by (rule "∃I")
3215   AOT_thus <[λy ∃x [R]xy]y>
3216     by (auto intro!: "β←C"(1) "cqt:2")
3217   qed
3218   AOT_thus <∃z [R]zy>
3219     by (rule "β→C"(1))
3220   qed
3221
3222   AOT_theorem "anc-her:6": <<([R*]xy & [R*]yz) → [R*]xz> (770.6)
3223   proof (rule "→I"; frule "&E"(1); drule "&E"(2))
3224     AOT_assume <[R*]xy>
3225     AOT_hence ϑ: <∀z ([R]xz → [F]z) & Hereditary(F,R) → [F]y> for F
3226       using "∀E"(2) ances[THEN "≡E"(1)] by blast
3227     AOT_assume <[R*]yz>
3228     AOT_hence ξ: <∀z ([R]yz → [F]z) & Hereditary(F,R) → [F]z> for F
3229       using "∀E"(2) ances[THEN "≡E"(1)] by blast
3230     AOT_show <[R*]xz>
3231     proof (rule ances[THEN "≡E"(2)]; safe intro!: GEN "→I")
3232       fix F
3233       AOT_assume ζ: <∀z ([R]xz → [F]z) & Hereditary(F,R)>
3234       AOT_show <[F]z>
3235       proof (rule ξ[THEN "→E", OF "&I"])
3236         AOT_show <Hereditary(F,R)>
3237           using ζ[THEN "&E"(2)].
3238       next
3239         AOT_show <∀z ([R]yz → [F]z)>
3240         proof(rule GEN; rule "→I")
3241           fix z
3242           AOT_assume <[R]yz>
3243           moreover AOT_have <[F]y>
3244             using ϑ[THEN "→E", OF ζ].
3245           ultimately AOT_show <[F]z>
3246             using ζ[THEN "&E"(2), THEN "hered:1"[THEN "≡dfE"],
3247               THEN "&E"(2), THEN "∀E"(2), THEN "∀E"(2),
3248               THEN "→E", THEN "→E"]
3249           by blast
3250         qed
3251       qed
3252     qed
3253   qed
3254
3255   AOT_define OneToOne :: <τ ⇒ φ> (<1-1'('_)>)
3256     "df-1-1:1": <1-1(R) ≡df R↓ & ∀x∀y∀z([R]xz & [R]yz → x = y)> (772.1)
3257
3258   AOT_define RigidOneToOne :: <τ ⇒ φ> (<Rigid1-1'('_)>)
3259     "df-1-1:2": <Rigid1-1(R) ≡df 1-1(R) & Rigid(R)> (772.2)
3260
3261   AOT_theorem "df-1-1:3": <Rigid1-1(R) → □1-1(R)> (772.3)
3262   proof(rule "→I")
3263     AOT_assume <Rigid1-1(R)>
3264     AOT_hence <1-1(R)> and RigidR: <Rigid(R)>
3265       using "df-1-1:2"[THEN "≡dfE"] "&E" by blast+
3266     AOT_hence 1: <[R]xz & [R]yz → x = y> for x y z
3267       using "df-1-1:1"[THEN "≡dfE"] "&E"(2) "∀E"(2) by blast
3268     AOT_have 1: <[R]xz & [R]yz → □x = y> for x y z
3269       by (AOT_subst (reverse) <□x = y> <x = y>)
3270         (auto simp: 1 "id-nec:2" "≡I" "qml:2"[axiom_inst])
3271     AOT_have <□∀x1...∀xn ([R]x1...xn → □[R]x1...xn)>
3272       using "df-rigid-rel:1"[THEN "≡dfE", OF RigidR] "&E" by blast
3273     AOT_hence <∀x1...∀xn □([R]x1...xn → □[R]x1...xn)>
3274     using "CBF"[THEN "→E"] by fast

```



```

3275 AOT_hence < $\forall x_1 \forall x_2 \square([R]x_1x_2 \rightarrow \square([R]x_1x_2))$ >
3276   using tuple_forall[THEN " $\equiv_{df}E$ "] by blast
3277 AOT_hence < $\square([R]xy \rightarrow \square([R]xy))$ > for x y
3278   using "VE"(2) by blast
3279 AOT_hence < $\square((\square([R]xz \rightarrow \square([R]xz)) \& ([R]yz \rightarrow \square([R]yz)))$ > for x y z
3280   by (metis "KBasic:3" "&I" " $\equiv E$ "(3) "raa-cor:3")
3281 moreover AOT_have < $\square((\square([R]xz \rightarrow \square([R]xz)) \& ([R]yz \rightarrow \square([R]yz))) \rightarrow$ 
3282    $\square((\square([R]xz \& [R]yz) \rightarrow \square([R]xz \& [R]yz)))$ > for x y z
3283   by (rule RM) (metis " $\rightarrow I$ " "KBasic:3" "&I" "&E"(1) "&E"(2) " $\equiv E$ "(2) " $\rightarrow E$ ")
3284 ultimately AOT_have 2: < $\square((\square([R]xz \& [R]yz) \rightarrow \square([R]xz \& [R]yz)))$ > for x y z
3285   using " $\rightarrow E$ " by blast
3286 AOT_hence 3: < $\square([R]xz \& [R]yz \rightarrow x = y)$ > for x y z
3287   using "sc-eq-box-box:6"[THEN " $\rightarrow E$ ", THEN " $\rightarrow E$ ", OF 2, OF 1] by blast
3288 AOT_hence 4: < $\square \forall x \forall y \forall z ([R]xz \& [R]yz \rightarrow x = y)$ >
3289   by (safe intro!: GEN BF[THEN " $\rightarrow E$ "] 3)
3290 AOT_thus < $\square 1-1(R)$ >
3291   by (AOT_subst_thm "df-1-1:1"[THEN " $\equiv_{df}$ ", THEN " $\equiv S$ "(1),
3292     OF "cqt:2[const_var]"[axiom_inst]])
3293 qed
3294
3295 AOT_theorem "df-1-1:4": < $\forall R(\text{Rigid}_{1-1}(R) \rightarrow \square \text{Rigid}_{1-1}(R))$ > (772.4)
3296 proof(rule GEN;rule " $\rightarrow I$ ")
3297 AOT_modally_strict {
3298   fix R
3299     AOT_assume 0: < $\text{Rigid}_{1-1}(R)$ >
3300     AOT_hence 1: < $R \downarrow$ >
3301       by (meson " $\equiv_{df}E$ " "&E"(1) "df-1-1:1" "df-1-1:2")
3302     AOT_hence 2: < $\square R \downarrow$ >
3303       using "exist-nec" " $\rightarrow E$ " by blast
3304     AOT_have 4: < $\square 1-1(R)$ >
3305       using "df-1-1:3"[unvarify R, OF 1, THEN " $\rightarrow E$ ", OF 0].
3306     AOT_have < $\text{Rigid}(R)$ >
3307       using 0 " $\equiv_{df}E$ "[OF "df-1-1:2"] "&E" by blast
3308     AOT_hence < $\square \forall x_1 \dots \forall x_n ([R]x_1 \dots x_n \rightarrow \square [R]x_1 \dots x_n)$ >
3309       using "df-rigid-rel:1"[THEN " $\equiv_{df}E$ "] "&E" by blast
3310     AOT_hence < $\square \square \forall x_1 \dots \forall x_n ([R]x_1 \dots x_n \rightarrow \square [R]x_1 \dots x_n)$ >
3311       by (metis "S5Basic:6" " $\equiv E$ "(1))
3312     AOT_hence < $\square \text{Rigid}(R)$ >
3313       apply (AOT_subst_def "df-rigid-rel:1")
3314       using 2 "KBasic:3" " $\equiv S$ "(2) " $\equiv E$ "(2) by blast
3315     AOT_thus < $\square \text{Rigid}_{1-1}(R)$ >
3316       apply (AOT_subst_def "df-1-1:2")
3317       using 4 "KBasic:3" " $\equiv S$ "(2) " $\equiv E$ "(2) by blast
3318 }
3319 qed
3320
3321 AOT_define InDomainOf :: < $\tau \Rightarrow \tau \Rightarrow \varphi$ > (< $\text{InDomainOf}'(\_,\_)$ >)
3322 "df-1-1:5": < $\text{InDomainOf}(x, R) \equiv_{df} \exists y [R]xy$ > (772.5)
3323
3324 AOT_register_rigid_restricted_type
3325 RigidOneToOneRelation: < $\text{Rigid}_{1-1}(II)$ >
3326 proof
3327   AOT_modally_strict {
3328     AOT_show < $\exists \alpha \text{Rigid}_{1-1}(\alpha)$ >
3329     proof (rule " $\exists I$ "(1)[where  $\tau = \langle \langle (=E) \rangle \rangle$ ])
3330       AOT_show < $\text{Rigid}_{1-1}(\langle (=E) \rangle)$ >
3331       proof (safe intro!: "df-1-1:2"[THEN " $\equiv_{df}I$ "] "&I" "df-1-1:1"[THEN " $\equiv_{df}I$ "]
3332         GEN " $\rightarrow I$ " "df-rigid-rel:1"[THEN " $\equiv_{df}I$ "] "=E[denotes]")
3333         fix x y z
3334         AOT_assume < $x =_E z \& y =_E z$ >
3335         AOT_thus < $x = y$ >
3336         by (metis "rule=E" "&E"(1) "Conjunction Simplification"(2)
3337           "=E-simple:2" id_sym " $\rightarrow E$ ")
3338     qed
3339   }

```

```

3338     next
3339     AOT_have < $\forall x \forall y \Box(x =_E y \rightarrow \Box x =_E y)$ >
3340     proof(rule GEN; rule GEN)
3341     AOT_show < $\Box(x =_E y \rightarrow \Box x =_E y)$ > for x y
3342     by (meson RN "deduction-theorem" "id-nec3:1" " $\equiv E$ "(1))
3343     qed
3344     AOT_hence < $\forall x_1 \dots \forall x_n \Box([\Box]_{x_1 \dots x_n} \rightarrow \Box[\Box]_{x_1 \dots x_n})$ >
3345     by (rule tuple_forall[THEN " $\equiv_{df} I$ "])
3346     AOT_thus < $\Box \forall x_1 \dots \forall x_n ([\Box]_{x_1 \dots x_n} \rightarrow \Box[\Box]_{x_1 \dots x_n})$ >
3347     using BF[THEN " $\rightarrow E$ "] by fast
3348     qed
3349     qed(fact " $=E$ [denotes]")
3350   }
3351 next
3352 AOT_modally_strict {
3353   AOT_show < $Rigid_{1-1}(\Pi) \rightarrow \Pi \downarrow$ > for  $\Pi$ 
3354   proof(rule " $\rightarrow I$ ")
3355     AOT_assume < $Rigid_{1-1}(\Pi)$ >
3356     AOT_hence < $1-1(\Pi)$ >
3357     using "df-1-1:2"[THEN " $\equiv_{df} E$ "] "&E" by blast
3358     AOT_thus < $\Pi \downarrow$ >
3359     using "df-1-1:1"[THEN " $\equiv_{df} E$ "] "&E" by blast
3360   qed
3361 }
3362 next
3363 AOT_modally_strict {
3364   AOT_show < $\forall F(Rigid_{1-1}(F) \rightarrow \Box Rigid_{1-1}(F))$ >
3365   by (safe intro!: GEN "df-1-1:4"[THEN " $\forall E$ "(2)])
3366 }
3367 qed
3368 AOT_register_variable_names
3369 RigidOneToOneRelation:  $\mathcal{R} \mathcal{S}$ 
3370
3371 AOT_define IdentityRestrictedToDomain :: < $\tau \Rightarrow \Pi$ > (<'(=)')>
3372 "id-d-R": < $(=_{\mathcal{R}}) =_{df} [\lambda xy \exists z ([\mathcal{R}]xz \ \& \ [\mathcal{R}]yz)]$ > (773)
3373
3374 syntax "_AOT_id_d_R_infix" :: < $\tau \Rightarrow \tau \Rightarrow \tau \Rightarrow \varphi$ > ("( _ = / _ )" [50, 51, 51] 50)
3375 translations
3376 "_AOT_id_d_R_infix  $\kappa \ \Pi \ \kappa'$ " ==
3377 "CONST AOT_exe (CONST IdentityRestrictedToDomain  $\Pi$ ) ( $\kappa, \kappa'$ )"
3378
3379 AOT_theorem "id-R-thm:1": < $x =_{\mathcal{R}} y \equiv \exists z ([\mathcal{R}]xz \ \& \ [\mathcal{R}]yz)$ > (774.1)
3380 proof -
3381   AOT_have 0: < $[\lambda xy \exists z ([\mathcal{R}]xz \ \& \ [\mathcal{R}]yz)] \downarrow$ > by "cqt:2"
3382   show ?thesis
3383     apply (rule " $=_{df} I$ "(1)[OF "id-d-R"])
3384     apply (fact 0)
3385     apply (rule "beta-C-meta"[THEN " $\rightarrow E$ ", OF 0, unvarify  $\nu_1 \nu_n$ ,
3386       where  $\tau = \langle \_, \_ \rangle$ , simplified])
3387     by (simp add: "&I" "ex:1:a" prod_denotesI "rule-ui:3")
3388   qed
3389
3390 AOT_theorem "id-R-thm:2": (774.2)
3391 < $x =_{\mathcal{R}} y \rightarrow (\text{InDomainOf}(x, \mathcal{R}) \ \& \ \text{InDomainOf}(y, \mathcal{R}))$ >
3392 proof(rule " $\rightarrow I$ ")
3393   AOT_assume < $x =_{\mathcal{R}} y$ >
3394   AOT_hence < $\exists z ([\mathcal{R}]xz \ \& \ [\mathcal{R}]yz)$ >
3395     using "id-R-thm:1"[THEN " $\equiv E$ "(1)] by simp
3396   then AOT_obtain z where z_prop: < $[\mathcal{R}]xz \ \& \ [\mathcal{R}]yz$ >
3397     using " $\exists E$ "[rotated] by blast
3398   AOT_show < $\text{InDomainOf}(x, \mathcal{R}) \ \& \ \text{InDomainOf}(y, \mathcal{R})$ >
3399   proof (safe intro!: "&I" "df-1-1:5"[THEN " $\equiv_{df} I$ "])
3400     AOT_show < $\exists y [\mathcal{R}]xy$ >

```

```

3401     using z_prop[THEN "&E"(1)] "∃I" by fast
3402 next
3403   AOT_show <∃z [ℛ]yz>
3404     using z_prop[THEN "&E"(2)] "∃I" by fast
3405   qed
3406 qed
3407
3408 AOT_theorem "id-R-thm:3": <x =ℛ y → x = y> (774.3)
3409 proof(rule "→I")
3410   AOT_assume <x =ℛ y>
3411   AOT_hence <∃z ([ℛ]xz & [ℛ]yz)>
3412     using "id-R-thm:1"[THEN "≡E"(1)] by simp
3413   then AOT_obtain z where z_prop: <[ℛ]xz & [ℛ]yz>
3414     using "∃E"[rotated] by blast
3415   AOT_thus <x = y>
3416     using "df-1-1:3"[THEN "→E", OF RigidOneToOneRelation.ψ,
3417           THEN "qml:2"[axiom_inst, THEN "→E"],
3418           THEN "≡dfE"[OF "df-1-1:1"], THEN "&E"(2),
3419           THEN "∀E"(2), THEN "∀E"(2),
3420           THEN "∀E"(2), THEN "→E"]
3421     by blast
3422 qed
3423
3424 AOT_theorem "id-R-thm:4": (774.4)
3425   <(InDomainOf(x, ℛ) ∨ InDomainOf(y, ℛ)) → (x =ℛ y ≡ x = y)>
3426 proof (rule "→I")
3427   AOT_assume <InDomainOf(x, ℛ) ∨ InDomainOf(y, ℛ)>
3428   moreover {
3429     AOT_assume <InDomainOf(x, ℛ)>
3430     AOT_hence <∃z [ℛ]xz>
3431       by (metis "≡dfE" "df-1-1:5")
3432     then AOT_obtain z where z_prop: <[ℛ]xz>
3433       using "∃E"[rotated] by blast
3434     AOT_have <x =ℛ y ≡ x = y>
3435     proof(safe intro!: "≡I" "→I" "id-R-thm:3"[THEN "→E"])
3436       AOT_assume <x = y>
3437       AOT_hence <[ℛ]yz>
3438         using z_prop "rule=E" by fast
3439       AOT_hence <[ℛ]xz & [ℛ]yz>
3440         using z_prop "&I" by blast
3441       AOT_hence <∃z ([ℛ]xz & [ℛ]yz)>
3442         by (rule "∃I")
3443       AOT_thus <x =ℛ y>
3444         using "id-R-thm:1" "≡E"(2) by blast
3445     qed
3446   }
3447   moreover {
3448     AOT_assume <InDomainOf(y, ℛ)>
3449     AOT_hence <∃z [ℛ]yz>
3450       by (metis "≡dfE" "df-1-1:5")
3451     then AOT_obtain z where z_prop: <[ℛ]yz>
3452       using "∃E"[rotated] by blast
3453     AOT_have <x =ℛ y ≡ x = y>
3454     proof(safe intro!: "≡I" "→I" "id-R-thm:3"[THEN "→E"])
3455       AOT_assume <x = y>
3456       AOT_hence <[ℛ]xz>
3457         using z_prop "rule=E" id_sym by fast
3458       AOT_hence <[ℛ]xz & [ℛ]yz>
3459         using z_prop "&I" by blast
3460       AOT_hence <∃z ([ℛ]xz & [ℛ]yz)>
3461         by (rule "∃I")
3462       AOT_thus <x =ℛ y>
3463         using "id-R-thm:1" "≡E"(2) by blast

```

```

3464   qed
3465 }
3466 ultimately AOT_show <x =R y ≡ x = y>
3467   by (metis "∀E"(2) "raa-cor:1")
3468 qed
3469
3470 AOT_theorem "id-R-thm:5": <InDomainOf(x, R) → x =R x> (774.5)
3471 proof (rule "→I")
3472   AOT_assume <InDomainOf(x, R)>
3473   AOT_hence <∃z [R]xz>
3474     by (metis "≡dfE" "df-1-1:5")
3475   then AOT_obtain z where z_prop: <[R]xz>
3476     using "∃E"[rotated] by blast
3477   AOT_hence <[R]xz & [R]xz>
3478     using "&I" by blast
3479   AOT_hence <∃z ([R]xz & [R]xz)>
3480     using "∃I" by fast
3481   AOT_thus <x =R x>
3482     using "id-R-thm:1" "≡E"(2) by blast
3483 qed
3484
3485 AOT_theorem "id-R-thm:6": <x =R y → y =R x> (774.6)
3486 proof(rule "→I")
3487   AOT_assume 0: <x =R y>
3488   AOT_hence 1: <InDomainOf(x, R) & InDomainOf(y, R)>
3489     using "id-R-thm:2"[THEN "→E"] by blast
3490   AOT_hence <x =R y ≡ x = y>
3491     using "id-R-thm:4"[THEN "→E", OF "∀I"(1)] "&E" by blast
3492   AOT_hence <x = y>
3493     using 0 by (metis "≡E"(1))
3494   AOT_hence <y = x>
3495     using id_sym by blast
3496   moreover AOT_have <y =R x ≡ y = x>
3497     using "id-R-thm:4"[THEN "→E", OF "∀I"(2)] 1 "&E" by blast
3498   ultimately AOT_show <y =R x>
3499     by (metis "≡E"(2))
3500 qed
3501
3502 AOT_theorem "id-R-thm:7": <x =R y & y =R z → x =R z> (774.7)
3503 proof (rule "→I"; frule "&E"(1); drule "&E"(2))
3504   AOT_assume 0: <x =R y>
3505   AOT_hence 1: <InDomainOf(x, R) & InDomainOf(y, R)>
3506     using "id-R-thm:2"[THEN "→E"] by blast
3507   AOT_hence <x =R y ≡ x = y>
3508     using "id-R-thm:4"[THEN "→E", OF "∀I"(1)] "&E" by blast
3509   AOT_hence x_eq_y: <x = y>
3510     using 0 by (metis "≡E"(1))
3511   AOT_assume 2: <y =R z>
3512   AOT_hence 3: <InDomainOf(y, R) & InDomainOf(z, R)>
3513     using "id-R-thm:2"[THEN "→E"] by blast
3514   AOT_hence <y =R z ≡ y = z>
3515     using "id-R-thm:4"[THEN "→E", OF "∀I"(1)] "&E" by blast
3516   AOT_hence <y = z>
3517     using 2 by (metis "≡E"(1))
3518   AOT_hence x_eq_z: <x = z>
3519     using x_eq_y id_trans by blast
3520   AOT_have <InDomainOf(x, R) & InDomainOf(z, R)>
3521     using 1 3 "&I" "&E" by meson
3522   AOT_hence <x =R z ≡ x = z>
3523     using "id-R-thm:4"[THEN "→E", OF "∀I"(1)] "&E" by blast
3524   AOT_thus <x =R z>
3525     using x_eq_z "≡E"(2) by blast
3526 qed

```

```

3527
3528 AOT_define WeakAncestral :: <Π ⇒ Π> (<_+>)
3529   "w-ances-df": <[ℛ]+ =df [λxy [ℛ]*xy ∨ x =ℛ y]> (775)
3530
3531 AOT_theorem "w-ances-df[den1]": <[λxy [Π]*xy ∨ x =Π y]↓> (775)
3532   by "cqt:2"
3533 AOT_theorem "w-ances-df[den2]": <[Π]+↓> (775)
3534   using "w-ances-df[den1]" "=dfI"(1)[OF "w-ances-df"] by blast
3535
3536 AOT_theorem "w-ances": <[ℛ]+xy ≡ ([ℛ]*xy ∨ x =ℛ y)> (776)
3537 proof -
3538   AOT_have 0: <[λxy [ℛ]*xy ∨ x =ℛ y]↓>
3539     by "cqt:2"
3540   AOT_have 1: <<<(AOT_term_of_var x,AOT_term_of_var y)⟩↓>
3541     by (simp add: "&I" "ex:1:a" prod_denotesI "rule-ui:3")
3542   have 2: <<[λμ1...μn [ℛ]*μ1...μn ∨ [(=ℛ)]μ1...μn]xy> =
3543     <<[λxy [ℛ]*xy ∨ [(=ℛ)]xy]xy>>
3544     by (simp add: cond_case_prod_eta)
3545   show ?thesis
3546     apply (rule "=dfI"(1)[OF "w-ances-df"])
3547     apply (fact "w-ances-df[den1]")
3548     using "beta-C-meta"[THEN "→E", OF 0, unvarify ν1νn,
3549           where τ=<(_,_)>, simplified, OF 1] 2 by simp
3550 qed
3551
3552 AOT_theorem "w-ances-her:1": <[ℛ]xy → [ℛ]+xy>
3553 proof(rule "→I")
3554   AOT_assume <[ℛ]xy>
3555   AOT_hence <[ℛ]+xy>
3556     using "anc-her:1"[THEN "→E"] by blast
3557   AOT_thus <[ℛ]+xy>
3558     using "w-ances"[THEN "≡E"(2)] "∀I" by blast
3559 qed
3560
3561 AOT_theorem "w-ances-her:2":
3562   <[F]x & [ℛ]+xy & Hereditary(F, ℛ) → [F]y>
3563 proof(rule "→I"; (frule "&E"(1); drule "&E"(2))+)
3564   AOT_assume 0: <[F]x>
3565   AOT_assume 1: <Hereditary(F, ℛ)>
3566   AOT_assume <[ℛ]+xy>
3567   AOT_hence <[ℛ]*xy ∨ x =ℛ y>
3568     using "w-ances"[THEN "≡E"(1)] by simp
3569   moreover {
3570     AOT_assume <[ℛ]*xy>
3571     AOT_hence <[F]y>
3572     using "anc-her:3"[THEN "→E", OF "&I", OF "&I"] 0 1 by blast
3573   }
3574   moreover {
3575     AOT_assume <x =ℛ y>
3576     AOT_hence <x = y>
3577     using "id-R-thm:3"[THEN "→E"] by blast
3578     AOT_hence <[F]y>
3579     using 0 "rule=E" by blast
3580   }
3581   ultimately AOT_show <[F]y>
3582     by (metis "∀E"(3) "raa-cor:1")
3583 qed
3584
3585 AOT_theorem "w-ances-her:3": <([ℛ]+xy & [ℛ]yz) → [ℛ]*xz>
3586 proof(rule "→I"; frule "&E"(1); drule "&E"(2))
3587   AOT_assume <[ℛ]+xy>
3588   moreover AOT_assume Ryz: <[ℛ]yz>
3589   ultimately AOT_have <[ℛ]*xy ∨ x =ℛ y>

```

```

3590     using "w-ances"[THEN "≡E"(1)] by metis
3591 moreover {
3592   AOT_assume R_star_xy: <[ $\mathcal{R}$ ]*xy>
3593   AOT_have <[ $\mathcal{R}$ ]*xz>
3594   proof (safe intro!: ances[THEN "≡E"(2)] "→I" GEN)
3595     fix F
3596     AOT_assume 0: <∀z ([ $\mathcal{R}$ ]xz → [F]z) & Hereditary(F,  $\mathcal{R}$ )>
3597     AOT_hence <[F]y>
3598     using R_star_xy ances[THEN "≡E"(1), OF R_star_xy,
3599       THEN "∀E"(2), THEN "→E"] by blast
3600     AOT_thus <[F]z>
3601     using "hered:1"[THEN "≡dfE", OF 0[THEN "&E"(2)], THEN "&E"(2)]
3602       "∀E"(2) "→E" Ryz by blast
3603   qed
3604 }
3605 moreover {
3606   AOT_assume <x = $\mathcal{R}$  y>
3607   AOT_hence <x = y>
3608   using "id-R-thm:3"[THEN "→E"] by blast
3609   AOT_hence <[ $\mathcal{R}$ ]xz>
3610   using Ryz "rule=E" id_sym by fast
3611   AOT_hence <[ $\mathcal{R}$ ]*xz>
3612   by (metis "anc-her:1"[THEN "→E"])
3613 }
3614 ultimately AOT_show <[ $\mathcal{R}$ ]*xz>
3615 by (metis "∀E"(3) "raa-cor:1")
3616 qed
3617
3618 AOT_theorem "w-ances-her:4": <([ $\mathcal{R}$ ]*xy & [ $\mathcal{R}$ ]yz) → [ $\mathcal{R}$ ]*xz>
3619 proof(rule "→I"; frule "&E"(1); drule "&E"(2))
3620   AOT_assume <[ $\mathcal{R}$ ]*xy>
3621   AOT_hence <[ $\mathcal{R}$ ]*xy ∨ x = $\mathcal{R}$  y>
3622   using "∀I" by blast
3623   AOT_hence <[ $\mathcal{R}$ ]*xy>
3624   using "w-ances"[THEN "≡E"(2)] by blast
3625   moreover AOT_assume <[ $\mathcal{R}$ ]yz>
3626   ultimately AOT_have <[ $\mathcal{R}$ ]*xz>
3627   using "w-ances-her:3"[THEN "→E", OF "&I"] by simp
3628   AOT_hence <[ $\mathcal{R}$ ]*xz ∨ x = $\mathcal{R}$  z>
3629   using "∀I" by blast
3630   AOT_thus <[ $\mathcal{R}$ ]*xz>
3631   using "w-ances"[THEN "≡E"(2)] by blast
3632 qed
3633
3634 AOT_theorem "w-ances-her:5": <([ $\mathcal{R}$ ]xy & [ $\mathcal{R}$ ]*yz) → [ $\mathcal{R}$ ]*xz>
3635 proof(rule "→I"; frule "&E"(1); drule "&E"(2))
3636   AOT_assume 0: <[ $\mathcal{R}$ ]xy>
3637   AOT_assume <[ $\mathcal{R}$ ]*yz>
3638   AOT_hence <[ $\mathcal{R}$ ]*yz ∨ y = $\mathcal{R}$  z>
3639   by (metis "≡E"(1) "w-ances")
3640   moreover {
3641     AOT_assume <[ $\mathcal{R}$ ]*yz>
3642     AOT_hence <[ $\mathcal{R}$ ]*xz>
3643     using 0 by (metis "anc-her:4" Adjunction "→E")
3644   }
3645   moreover {
3646     AOT_assume <y = $\mathcal{R}$  z>
3647     AOT_hence <y = z>
3648     by (metis "id-R-thm:3" "→E")
3649     AOT_hence <[ $\mathcal{R}$ ]xz>
3650     using 0 "rule=E" by fast
3651     AOT_hence <[ $\mathcal{R}$ ]*xz>
3652     by (metis "anc-her:1" "→E")

```

```

3653   }
3654   ultimately AOT_show <[ $\mathcal{R}$ ]+xz> by (metis "\E"(2) "reductio-aa:1")
3655 qed
3656
3657 AOT_theorem "w-ances-her:6": <([ $\mathcal{R}$ ]+xy & [ $\mathcal{R}$ ]+yz)  $\rightarrow$  [ $\mathcal{R}$ ]+xz>
3658 proof(rule "\I"; frule "&E"(1); drule "&E"(2))
3659   AOT_assume 0: <[ $\mathcal{R}$ ]+xy>
3660   AOT_hence 1: <[ $\mathcal{R}$ ]+xy  $\vee$  x = $\mathcal{R}$  y>
3661     by (metis "\E"(1) "w-ances")
3662   AOT_assume 2: <[ $\mathcal{R}$ ]+yz>
3663   {
3664     AOT_assume <x = $\mathcal{R}$  y>
3665     AOT_hence <x = y>
3666       by (metis "id-R-thm:3" "\E")
3667     AOT_hence <[ $\mathcal{R}$ ]+xz>
3668       using 2 "rule=E" id_sym by fast
3669   }
3670   moreover {
3671     AOT_assume < $\neg$ (x = $\mathcal{R}$  y)>
3672     AOT_hence 3: <[ $\mathcal{R}$ ]+xy>
3673       using 1 by (metis "\E"(3))
3674     AOT_have <[ $\mathcal{R}$ ]+yz  $\vee$  y = $\mathcal{R}$  z>
3675       using 2 by (metis "\E"(1) "w-ances")
3676     moreover {
3677       AOT_assume <[ $\mathcal{R}$ ]+yz>
3678       AOT_hence <[ $\mathcal{R}$ ]+xz>
3679         using 3 by (metis "anc-her:6" Adjunction "\E")
3680       AOT_hence <[ $\mathcal{R}$ ]+xz>
3681         by (metis "\I"(1) "\E"(2) "w-ances")
3682     }
3683     moreover {
3684       AOT_assume <y = $\mathcal{R}$  z>
3685       AOT_hence <y = z>
3686         by (metis "id-R-thm:3" "\E")
3687       AOT_hence <[ $\mathcal{R}$ ]+xz>
3688         using 0 "rule=E" id_sym by fast
3689     }
3690     ultimately AOT_have <[ $\mathcal{R}$ ]+xz>
3691       by (metis "\E"(3) "reductio-aa:1")
3692   }
3693   ultimately AOT_show <[ $\mathcal{R}$ ]+xz>
3694     by (metis "reductio-aa:1")
3695 qed
3696
3697 AOT_theorem "w-ances-her:7": <[ $\mathcal{R}$ ]+xy  $\rightarrow$   $\exists$ z([ $\mathcal{R}$ ]+xz & [ $\mathcal{R}$ ]zy)>
3698 proof(rule "\I")
3699   AOT_assume 0: <[ $\mathcal{R}$ ]+xy>
3700   AOT_have 1: < $\forall$ z ([ $\mathcal{R}$ ]xz  $\rightarrow$  [ $\Pi$ ]z) & Hereditary( $\Pi$ ,  $\mathcal{R}$ )  $\rightarrow$  [ $\Pi$ ]y> if < $\Pi \downarrow$ > for  $\Pi$ 
3701     using ances[THEN "\E"(1), THEN "\E"(1), OF 0] that by blast
3702   AOT_have <[ $\lambda$ y  $\exists$ z([ $\mathcal{R}$ ]+xz & [ $\mathcal{R}$ ]zy)]y>
3703   proof (rule 1[THEN "\E"]; "cqt:2[lamba]";
3704     safe intro!: "&I" GEN "\I" "hered:1"[THEN "\EdfI"] "cqt:2")
3705     fix z
3706     AOT_assume 0: <[ $\mathcal{R}$ ]xz>
3707     AOT_hence < $\exists$ z [ $\mathcal{R}$ ]xz> by (rule "\E")
3708     AOT_hence <InDomainOf(x,  $\mathcal{R}$ )> by (metis "\EdfI" "df-1-1:5")
3709     AOT_hence <x = $\mathcal{R}$  x> by (metis "id-R-thm:5" "\E")
3710     AOT_hence <[ $\mathcal{R}$ ]+xx> by (metis "\I"(2) "\E"(2) "w-ances")
3711     AOT_hence <[ $\mathcal{R}$ ]+xx & [ $\mathcal{R}$ ]xz> using 0 "&I" by blast
3712     AOT_hence < $\exists$ y ([ $\mathcal{R}$ ]+xy & [ $\mathcal{R}$ ]yz)> by (rule "\E")
3713     AOT_thus <[ $\lambda$ y  $\exists$ z ([ $\mathcal{R}$ ]+xz & [ $\mathcal{R}$ ]zy)]z>
3714       by (auto intro!: "\beta $\leftarrow$ C"(1) "cqt:2")
3715   next

```

```

3716   fix x' y
3717   AOT_assume Rx'y: <[ $\mathcal{R}$ ]x'y>
3718   AOT_assume <[ $\lambda y \exists z ([\mathcal{R}]^+xz \ \& \ [\mathcal{R}]zy)]x'$ >
3719   AOT_hence < $\exists z ([\mathcal{R}]^+xz \ \& \ [\mathcal{R}]zx')$ >
3720     using " $\beta \rightarrow C$ "(1) by blast
3721   then AOT_obtain c where c_prop: <[ $\mathcal{R}$ ]xc & [ $\mathcal{R}$ ]cx'>
3722     using " $\exists E$ "[rotated] by blast
3723   AOT_hence <[ $\mathcal{R}$ ]xx'>
3724     by (meson Rx'y "anc-her:1" "anc-her:6" Adjunction " $\rightarrow E$ " "w-ances-her:3")
3725   AOT_hence <[ $\mathcal{R}$ ]xx'  $\vee$  x = $_R$  x'> by (rule " $\vee I$ ")
3726   AOT_hence <[ $\mathcal{R}$ ]xx'> by (metis " $\equiv E$ "(2) "w-ances")
3727   AOT_hence <[ $\mathcal{R}$ ]xx' & [ $\mathcal{R}$ ]x'y> using Rx'y by (metis "&I")
3728   AOT_hence < $\exists z ([\mathcal{R}]^+xz \ \& \ [\mathcal{R}]zy)$ > by (rule " $\exists I$ ")
3729   AOT_thus <[ $\lambda y \exists z ([\mathcal{R}]^+xz \ \& \ [\mathcal{R}]zy)]y$ >
3730     by (auto intro!: " $\beta \leftarrow C$ "(1) "cqt:2")
3731   qed
3732   AOT_thus < $\exists z ([\mathcal{R}]^+xz \ \& \ [\mathcal{R}]zy)$ >
3733     using " $\beta \rightarrow C$ "(1) by fast
3734   qed
3735
3736   AOT_theorem "1-1-R:1": <([ $\mathcal{R}$ ]xy & [ $\mathcal{R}$ ]zy)  $\rightarrow$  [ $\mathcal{R}$ ]zx> (778.1)
3737   proof(rule " $\rightarrow I$ "; frule "&E"(1); drule "&E"(2))
3738     AOT_assume <[ $\mathcal{R}$ ]zy>
3739     AOT_hence < $\exists x ([\mathcal{R}]^+zx \ \& \ [\mathcal{R}]xy)$ >
3740       using "w-ances-her:7"[THEN " $\rightarrow E$ "] by simp
3741     then AOT_obtain a where a_prop: <[ $\mathcal{R}$ ]za & [ $\mathcal{R}$ ]ay>
3742       using " $\exists E$ "[rotated] by blast
3743     moreover AOT_assume <[ $\mathcal{R}$ ]xy>
3744     ultimately AOT_have <x = a>
3745       using "df-1-1:2"[THEN " $\equiv_{df} E$ ", OF RigidOneToOneRelation. $\psi$ , THEN "&E"(1),
3746         THEN " $\equiv_{df} E$ "[OF "df-1-1:1"], THEN "&E"(2), THEN " $\forall E$ "(2),
3747         THEN " $\forall E$ "(2), THEN " $\forall E$ "(2), THEN " $\rightarrow E$ ", OF "&I"]
3748       "&E" by blast
3749     AOT_thus <[ $\mathcal{R}$ ]zx>
3750       using a_prop[THEN "&E"(1)] "rule=E" id_sym by fast
3751   qed
3752
3753   AOT_theorem "1-1-R:2": <[ $\mathcal{R}$ ]xy  $\rightarrow$  ( $\neg$ [ $\mathcal{R}$ ]xx  $\rightarrow$   $\neg$ [ $\mathcal{R}$ ]yy)> (778.2)
3754   proof(rule " $\rightarrow I$ "; rule "useful-tautologies:5"[THEN " $\rightarrow E$ "]; rule " $\rightarrow I$ ")
3755     AOT_assume 0: <[ $\mathcal{R}$ ]xy>
3756     moreover AOT_assume <[ $\mathcal{R}$ ]yy>
3757     ultimately AOT_have <[ $\mathcal{R}$ ]yx>
3758       using "1-1-R:1"[THEN " $\rightarrow E$ ", OF "&I"] by blast
3759     AOT_thus <[ $\mathcal{R}$ ]xx>
3760       using 0 by (metis "&I" " $\rightarrow E$ " "w-ances-her:5")
3761   qed
3762
3763   AOT_theorem "1-1-R:3": < $\neg$ [ $\mathcal{R}$ ]xx  $\rightarrow$  ([ $\mathcal{R}$ ]xy  $\rightarrow$   $\neg$ [ $\mathcal{R}$ ]yy)> (778.3)
3764   proof(safe intro!: " $\rightarrow I$ ")
3765     AOT_have 0: <[ $\lambda z \neg$ [ $\mathcal{R}$ ]zz] $\downarrow$ > by "cqt:2"
3766     AOT_assume 1: < $\neg$ [ $\mathcal{R}$ ]xx>
3767     AOT_assume 2: <[ $\mathcal{R}$ ]xy>
3768     AOT_have <[ $\lambda z \neg$ [ $\mathcal{R}$ ]zz]y>
3769     proof(rule "w-ances-her:2"[unvarify F, OF 0, THEN " $\rightarrow E$ "];
3770       safe intro!: "&I" "hered:1"[THEN " $\equiv_{df} I$ "] "cqt:2" GEN " $\rightarrow I$ ")
3771       AOT_show <[ $\lambda z \neg$ [ $\mathcal{R}$ ]zz]x>
3772         by (auto intro!: " $\beta \leftarrow C$ "(1) "cqt:2" simp: 1)
3773     next
3774     AOT_show <[ $\mathcal{R}$ ]xy> by (fact 2)
3775     next
3776     fix x y
3777     AOT_assume <[ $\lambda z \neg$ [ $\mathcal{R}$ ]zz]x>
3778     AOT_hence < $\neg$ [ $\mathcal{R}$ ]xx> by (rule " $\beta \rightarrow C$ "(1))

```



```

3779   moreover AOT_assume <[ $\mathcal{R}$ ]xy>
3780   ultimately AOT_have < $\neg$ [ $\mathcal{R}$ ]*yy>
3781     using "1-1-R:2"[THEN " $\rightarrow$ E", THEN " $\rightarrow$ E"] by blast
3782   AOT_thus <[ $\lambda$ z  $\neg$ [ $\mathcal{R}$ ]*zz]y>
3783     by (auto intro!: " $\beta \leftarrow C$ "(1) "cqt:2")
3784   qed
3785   AOT_thus < $\neg$ [ $\mathcal{R}$ ]*yy>
3786     using " $\beta \rightarrow C$ "(1) by blast
3787 qed
3788
3789 AOT_theorem "1-1-R:4": <[ $\mathcal{R}$ ]*xy  $\rightarrow$  InDomainOf(x,  $\mathcal{R}$ )> (778.4)
3790 proof(rule " $\rightarrow$ I"; rule "df-1-1:5"[THEN " $\equiv_{df} I$ "])
3791   AOT_assume 1: <[ $\mathcal{R}$ ]*xy>
3792   AOT_have <[ $\lambda$ z [ $\mathcal{R}$ ]*xz  $\rightarrow$   $\exists$ y [ $\mathcal{R}$ ]xy]y>
3793   proof (safe intro!: "anc-her:2"[unvarify F, THEN " $\rightarrow$ E"];
3794     safe intro!: "cqt:2" "&I" GEN " $\rightarrow$ I" "hered:1"[THEN " $\equiv_{df} I$ "])
3795     AOT_show <[ $\mathcal{R}$ ]*xy> by (fact 1)
3796   next
3797     fix z
3798     AOT_assume <[ $\mathcal{R}$ ]xz>
3799     AOT_thus <[ $\lambda$ z [ $\mathcal{R}$ ]*xz  $\rightarrow$   $\exists$ y [ $\mathcal{R}$ ]xy]z>
3800       by (safe intro!: " $\beta \leftarrow C$ "(1) "cqt:2")
3801         (meson " $\rightarrow$ I" "existential:2[const_var]")
3802   next
3803     fix x' y
3804     AOT_assume Rx'y: <[ $\mathcal{R}$ ]x'y>
3805     AOT_assume <[ $\lambda$ z [ $\mathcal{R}$ ]*xz  $\rightarrow$   $\exists$ y [ $\mathcal{R}$ ]xy]x'>
3806     AOT_hence 0: <[ $\mathcal{R}$ ]*xx'  $\rightarrow$   $\exists$ y [ $\mathcal{R}$ ]xy> by (rule " $\beta \rightarrow C$ "(1))
3807     AOT_have 1: <[ $\mathcal{R}$ ]*xy  $\rightarrow$   $\exists$ y [ $\mathcal{R}$ ]xy>
3808     proof(rule " $\rightarrow$ I")
3809       AOT_assume <[ $\mathcal{R}$ ]*xy>
3810       AOT_hence <[ $\mathcal{R}$ ]*xx'> by (metis Rx'y "&I" "1-1-R:1" " $\rightarrow$ E")
3811       AOT_hence <[ $\mathcal{R}$ ]*xx'  $\vee$  x = $\mathcal{R}$  x'> by (metis " $\equiv E$ "(1) "w-ances")
3812       moreover {
3813         AOT_assume <[ $\mathcal{R}$ ]*xx'>
3814         AOT_hence < $\exists$ y [ $\mathcal{R}$ ]xy> using 0 by (metis " $\rightarrow$ E")
3815       }
3816       moreover {
3817         AOT_assume <x = $\mathcal{R}$  x'>
3818         AOT_hence <x = x'> by (metis "id-R-thm:3" " $\rightarrow$ E")
3819         AOT_hence <[ $\mathcal{R}$ ]xy> using Rx'y "rule=E" id_sym by fast
3820         AOT_hence < $\exists$ y [ $\mathcal{R}$ ]xy> by (rule " $\exists I$ ")
3821       }
3822       ultimately AOT_show < $\exists$ y [ $\mathcal{R}$ ]xy>
3823         by (metis " $\vee E$ "(3) "reductio-aa:1")
3824     qed
3825     AOT_show <[ $\lambda$ z [ $\mathcal{R}$ ]*xz  $\rightarrow$   $\exists$ y [ $\mathcal{R}$ ]xy]y>
3826       by (auto intro!: " $\beta \leftarrow C$ "(1) "cqt:2" 1)
3827   qed
3828   AOT_hence <[ $\mathcal{R}$ ]*xy  $\rightarrow$   $\exists$ y [ $\mathcal{R}$ ]xy> by (rule " $\beta \rightarrow C$ "(1))
3829   AOT_thus < $\exists$ y [ $\mathcal{R}$ ]xy> using 1 " $\rightarrow$ E" by blast
3830 qed
3831
3832 AOT_theorem "1-1-R:5": <[ $\mathcal{R}$ ]*xy  $\rightarrow$  InDomainOf(x,  $\mathcal{R}$ )> (778.5)
3833 proof (rule " $\rightarrow$ I")
3834   AOT_assume <[ $\mathcal{R}$ ]*xy>
3835   AOT_hence <[ $\mathcal{R}$ ]*xy  $\vee$  x = $\mathcal{R}$  y>
3836     by (metis " $\equiv E$ "(1) "w-ances")
3837   moreover {
3838     AOT_assume <[ $\mathcal{R}$ ]*xy>
3839     AOT_hence <InDomainOf(x,  $\mathcal{R}$ )>
3840       using "1-1-R:4" " $\rightarrow$ E" by blast
3841   }

```

```

3842   moreover {
3843     AOT_assume <x = $\mathcal{R}$  y>
3844     AOT_hence <InDomainOf(x, $\mathcal{R}$ )>
3845     by (metis "Conjunction Simplification"(1) "id-R-thm:2" " $\rightarrow$ E")
3846   }
3847   ultimately AOT_show <InDomainOf(x, $\mathcal{R}$ )>
3848   by (metis " $\vee$ E"(3) "reductio-aa:1")
3849 qed
3850
3851 AOT_theorem "pre-ind":
3852   <([F]z &  $\forall x \forall y$ (([ $\mathcal{R}$ ]+zx & [ $\mathcal{R}$ ]+zy)  $\rightarrow$  ([ $\mathcal{R}$ ]xy  $\rightarrow$  ([F]x  $\rightarrow$  [F]y)))  $\rightarrow$ 
3853    $\forall x$  ([ $\mathcal{R}$ ]+zx  $\rightarrow$  [F]x)>
3854 proof(safe intro!: " $\rightarrow$ I" GEN)
3855   AOT_have den: < $\lambda y$  [F]y & [ $\mathcal{R}$ ]+zy $\downarrow$ > by "cqt:2"
3856   fix x
3857   AOT_assume  $\vartheta$ : <[F]z &  $\forall x \forall y$ (([ $\mathcal{R}$ ]+zx & [ $\mathcal{R}$ ]+zy)  $\rightarrow$  ([ $\mathcal{R}$ ]xy  $\rightarrow$  ([F]x  $\rightarrow$  [F]y)))>
3858   AOT_assume 0: <[ $\mathcal{R}$ ]+zx>
3859
3860   AOT_have < $\lambda y$  [F]y & [ $\mathcal{R}$ ]+zy>x>
3861   proof (rule "w-ances-her:2"[unverify F, OF den, THEN " $\rightarrow$ E"]; safe intro!: "&I")
3862     AOT_show < $\lambda y$  [F]y & [ $\mathcal{R}$ ]+zy>x>
3863     proof (safe intro!: " $\beta \leftarrow$ C"(1) "cqt:2" "&I")
3864       AOT_show <[F]z> using  $\vartheta$  "&E" by blast
3865     next
3866       AOT_show <[ $\mathcal{R}$ ]+zz>
3867       by (rule "w-ances"[THEN " $\equiv$ E"(2), OF " $\vee$ I"(2)])
3868       (meson "0" "id-R-thm:5" "1-1-R:5" " $\rightarrow$ E")
3869     qed
3870   next
3871   AOT_show <[ $\mathcal{R}$ ]+zx> by (fact 0)
3872 next
3873 AOT_show <Hereditary( $\lambda y$  [F]y & [ $\mathcal{R}$ ]+zy, $\mathcal{R}$ )>
3874 proof (safe intro!: "hered:1"[THEN " $\equiv_{df}$ I"] "&I" "cqt:2" GEN " $\rightarrow$ I")
3875   fix x' y
3876   AOT_assume 1: <[ $\mathcal{R}$ ]+x'y>
3877   AOT_assume < $\lambda y$  [F]y & [ $\mathcal{R}$ ]+zy>x'>
3878   AOT_hence 2: <[F]x' & [ $\mathcal{R}$ ]+zx'> by (rule " $\beta \rightarrow$ C"(1))
3879   AOT_have <[ $\mathcal{R}$ ]+zy> using 1 2[THEN "&E"(2)]
3880   by (metis Adjunction "modus-tollens:1" "reductio-aa:1" "w-ances-her:3")
3881   AOT_hence 3: <[ $\mathcal{R}$ ]+zy> by (metis " $\vee$ I"(1) " $\equiv$ E"(2) "w-ances")
3882   AOT_show < $\lambda y$  [F]y & [ $\mathcal{R}$ ]+zy>y>
3883   proof (safe intro!: " $\beta \leftarrow$ C"(1) "cqt:2" "&I" 3)
3884     AOT_show <[F]y>
3885     proof (rule  $\vartheta$ [THEN "&E"(2), THEN " $\vee$ E"(2), THEN " $\vee$ E"(2),
3886       THEN " $\rightarrow$ E", THEN " $\rightarrow$ E", THEN " $\rightarrow$ E"]])
3887       AOT_show <[ $\mathcal{R}$ ]+zx' & [ $\mathcal{R}$ ]+zy>
3888       using 2 3 "&E" "&I" by blast
3889     next
3890     AOT_show <[ $\mathcal{R}$ ]+x'y> by (fact 1)
3891     next
3892     AOT_show <[F]x'> using 2 "&E" by blast
3893   qed
3894   qed
3895   qed
3896   qed
3897   AOT_thus <[F]x> using " $\beta \rightarrow$ C"(1) "&E"(1) by fast
3898 qed
3899
3900 text<The following is not part of PLM, but a theorem of AOT.
3901   It states that the predecessor relation coexists with numbering a property.
3902   We will use this fact to derive the predecessor axiom, which asserts that the
3903   predecessor relation denotes, from the fact that our models validate that
3904   numbering a property denotes.>

```

```

3905 AOT_theorem pred_coex:
3906   <[λxy ∃F∃u ([F]u & Numbers(y,F) & Numbers(x,[F]-u)]↓ ≡ ∀F ([λx Numbers(x,F)]↓)>
3907 proof(safe intro!: "≡I" "→I" GEN)
3908   fix F
3909   let ?P = <<[λxy ∃F∃u ([F]u & Numbers(y,F) & Numbers(x,[F]-u)]>>
3910   AOT_assume <[<?P>]↓>
3911   AOT_hence <□[<?P>]↓>
3912     using "exist-nec" "→E" by blast
3913   moreover AOT_have
3914     <□[<?P>]↓ → □(∀x∀y(∀F([F]x ≡ [F]y) → (Numbers(x,F) ≡ Numbers(y,F))))>
3915 proof(rule RM; safe intro!: "→I" GEN)
3916   AOT_modally_strict {
3917     fix x y
3918     AOT_assume pred_den: <[<?P>]↓>
3919     AOT_hence pred_equiv:
3920       <[<?P>]xy ≡ ∃F∃u ([F]u & Numbers(y,F) & Numbers(x,[F]-u)]> for x y
3921       by (safe intro!: "beta-C-meta"[unvarify ν1νn, where τ=<(<,>), THEN "→E",
3922         rotated, OF pred_den, simplified]
3923         tuple_denotes[THEN "≡dfI"] "&I" "cqt:2")
3924     text<We show as a subproof that any natural cardinal that is not zero
3925       has a predecessor.>
3926     AOT_have CardinalPredecessor:
3927       <∃y [<?P>]yx> if card_x: <NaturalCardinal(x)> and x_nonzero: <x ≠ 0> for x
3928     proof -
3929       AOT_have <∃G x = #G>
3930         using card[THEN "≡dfE", OF card_x].
3931       AOT_hence <∃G Numbers(x,G)>
3932         using "eq-df-num"[THEN "≡E"(1)] by blast
3933       then AOT_obtain G' where numxG': <Numbers(x,G')>
3934         using "∃E"[rotated] by blast
3935       AOT_obtain G where <Rigidifies(G,G')>
3936         using "rigid-der:3" "∃E"[rotated] by blast
3937
3938       AOT_hence H: <Rigid(G) & ∀x ([G]x ≡ [G']x)>
3939         using "df-rigid-rel:2"[THEN "≡dfE"] by blast
3940       AOT_have H_rigid: <□∀x ([G]x → □[G]x)>
3941         using H[THEN "&E"(1), THEN "df-rigid-rel:1"[THEN "≡dfE"], THEN "&E"(2)].
3942       AOT_hence <∀x □([G]x → □[G]x)>
3943         using "CBF" "→E" by blast
3944       AOT_hence R: <□([G]x → □[G]x)> for x using "∀E"(2) by blast
3945       AOT_hence rigid: <[G]x ≡  $\mathcal{A}$ [G]x> for x
3946         by (metis "≡E"(6) "oth-class-taut:3:a" "sc-eq-fur:2" "→E")
3947       AOT_have <G ≡E G'>
3948       proof (safe intro!: eqE[THEN "≡dfI"] "&I" "cqt:2" GEN "→I")
3949         AOT_show <[G]x ≡ [G']x> for x using H[THEN "&E"(2)] "∀E"(2) by fast
3950       qed
3951       AOT_hence <G ≡E G'>
3952         by (rule "apE-eqE:2"[THEN "→E", OF "&I", rotated])
3953         (simp add: "eq-part:1")
3954       AOT_hence numxG: <Numbers(x,G)>
3955         using "num-tran:1"[THEN "→E", THEN "≡E"(2)] numxG' by blast
3956
3957     {
3958       AOT_assume <¬∃y(y ≠ x & [<?P>]yx)>
3959       AOT_hence <∀y ¬(y ≠ x & [<?P>]yx)>
3960         using "cqt-further:4" "→E" by blast
3961       AOT_hence <¬(y ≠ x & [<?P>]yx)> for y
3962         using "∀E"(2) by blast
3963       AOT_hence 0: <¬y ≠ x ∨ ¬[<?P>]yx> for y
3964         using "¬→E" "intro-elim:3:c" "oth-class-taut:5:a" by blast
3965       {
3966         fix y
3967         AOT_assume <[<?P>]yx>

```

```

3968     AOT_hence < $\neg y \neq x$ >
3969         using 0 " $\neg\neg I$ " "con-dis-i-e:4:c" by blast
3970     AOT_hence < $y = x$ >
3971         using "--infix" " $\equiv_{df} I$ " "raa-cor:4" by blast
3972 } note Pxy_imp_eq = this
3973 AOT_have <[«?P»]xx>
3974 proof(rule "raa-cor:1")
3975     AOT_assume notPxx: < $\neg$ [«?P»]xx>
3976     AOT_hence < $\neg\exists F\exists u([F]u \ \& \ \text{Numbers}(x,F) \ \& \ \text{Numbers}(x,[F]^{-u}))$ >
3977         using pred_equiv "intro-elim:3:c" by blast
3978     AOT_hence < $\forall F \neg\exists u([F]u \ \& \ \text{Numbers}(x,F) \ \& \ \text{Numbers}(x,[F]^{-u}))$ >
3979         using "cqt-further:4"[THEN " $\rightarrow E$ "] by blast
3980     AOT_hence < $\neg\exists u([F]u \ \& \ \text{Numbers}(x,F) \ \& \ \text{Numbers}(x,[F]^{-u}))$ > for F
3981         using " $\forall E$ "(2) by blast
3982     AOT_hence < $\forall y \neg(O!y \ \& \ ([F]y \ \& \ \text{Numbers}(x,F) \ \& \ \text{Numbers}(x,[F]^{-y})))$ > for F
3983         using "cqt-further:4"[THEN " $\rightarrow E$ "] by blast
3984     AOT_hence 0: < $\neg(O!u \ \& \ ([F]u \ \& \ \text{Numbers}(x,F) \ \& \ \text{Numbers}(x,[F]^{-u})))$ > for F u
3985         using " $\forall E$ "(2) by blast
3986     AOT_have < $\Box\neg\exists u [G]u$ >
3987     proof(rule "raa-cor:1")
3988         AOT_assume < $\neg\Box\neg\exists u [G]u$ >
3989         AOT_hence < $\Diamond\exists u [G]u$ >
3990             using " $\equiv_{df} I$ " "conventions:5" by blast
3991         AOT_hence < $\exists u \Diamond[G]u$ >
3992             by (metis "Ordinary.res-var-bound-reas[BF $\Diamond$ ]"[THEN " $\rightarrow E$ "])
3993         then AOT_obtain u where posGu: < $\Diamond[G]u$ >
3994             using "Ordinary. $\exists E$ "[rotated] by meson
3995         AOT_hence Gu: <[G]u>
3996             by (meson "B $\Diamond$ " "K $\Diamond$ " " $\rightarrow E$ " R)
3997         AOT_have < $\neg([G]u \ \& \ \text{Numbers}(x,G) \ \& \ \text{Numbers}(x,[G]^{-u}))$ >
3998             using 0 Ordinary. $\psi$ 
3999             by (metis "con-dis-i-e:1" "raa-cor:1")
4000         AOT_hence notnumx: < $\neg\text{Numbers}(x,[G]^{-u})$ >
4001             using Gu numxG "con-dis-i-e:1" "raa-cor:5" by metis
4002         AOT_obtain y where numy: < $\text{Numbers}(y,[G]^{-u})$ >
4003             using "num:1"[unvarify G, OF "F-u[den]" " $\exists E$ "[rotated] by blast
4004         AOT_hence <[G]u & Numbers(x,G) & Numbers(y,[G]^{-u})>
4005             using Gu numxG "&I" by blast
4006         AOT_hence < $\exists u ([G]u \ \& \ \text{Numbers}(x,G) \ \& \ \text{Numbers}(y,[G]^{-u}))$ >
4007             by (rule "Ordinary. $\exists I$ ")
4008         AOT_hence < $\exists G\exists u ([G]u \ \& \ \text{Numbers}(x,G) \ \& \ \text{Numbers}(y,[G]^{-u}))$ >
4009             by (rule " $\exists I$ ")
4010         AOT_hence <[«?P»]yx>
4011             using pred_equiv[THEN " $\equiv E$ "(2)] by blast
4012         AOT_hence < $y = x$ > using Pxy_imp_eq by blast
4013         AOT_hence < $\text{Numbers}(x,[G]^{-u})$ >
4014             using numy "rule=E" by fast
4015         AOT_thus <p &  $\neg$ p> for p using notnumx "reductio-aa:1" by blast
4016     qed
4017     AOT_hence < $\neg\exists u [G]u$ >
4018         using "qml:2"[axiom_inst, THEN " $\rightarrow E$ "] by blast
4019     AOT_hence numOG: < $\text{Numbers}(0, G)$ >
4020         using "OF:1"[THEN " $\equiv E$ "(1)] by blast
4021     AOT_hence <x = 0>
4022         using "pre-Hume"[unvarify x, THEN " $\rightarrow E$ ", OF "zero:2", OF "&I",
4023             THEN " $\equiv E$ "(2), OF numOG, OF numxG, OF "eq-part:1"]
4024         id_sym by blast
4025     moreover AOT_have < $\neg x = 0$ >
4026         using x_nonzero
4027         using "--infix" " $\equiv_{df} E$ " by blast
4028     ultimately AOT_show <p &  $\neg$ p> for p using "reductio-aa:1" by blast
4029     qed
4030 }

```

```

4031 AOT_hence <[<?P>>]xx ∨ ∃y (y ≠ x & [<?P>>]yx)>
4032   using "con-dis-i-e:3:a" "con-dis-i-e:3:b" "raa-cor:1" by blast
4033 moreover {
4034   AOT_assume <[<?P>>]xx>
4035   AOT_hence <∃y [<?P>>]yx>
4036     by (rule "∃I")
4037 }
4038 moreover {
4039   AOT_assume <∃y (y ≠ x & [<?P>>]yx)>
4040   then AOT_obtain y where <y ≠ x & [<?P>>]yx>
4041     using "∃E"[rotated] by blast
4042   AOT_hence <[<?P>>]yx>
4043     using "&E" by blast
4044   AOT_hence <∃y [<?P>>]yx>
4045     by (rule "∃I")
4046 }
4047 ultimately AOT_show <∃y [<?P>>]yx>
4048   using "∨E"(1) "→I" by blast
4049 qed
4050
4051 text<Given above lemma, we can show that if one of two indistinguishable objects
4052   numbers a property, the other one numbers this property as well.>
4053 AOT_assume indist: <∀F([F]x ≡ [F]y)>
4054 AOT_assume numxF: <Numbers(x,F)>
4055 AOT_hence 0: <NaturalCardinal(x)>
4056   by (metis "eq-num:6" "vdash-properties:10")
4057 text<We show by case distinction that x equals y.
4058   As first case we consider x to be non-zero.>
4059 {
4060   AOT_assume <¬(x = 0)>
4061   AOT_hence <x ≠ 0>
4062     by (metis "--infix" "≡_drI")
4063   AOT_hence <∃y [<?P>>]yx>
4064     using CardinalPredecessor 0 by blast
4065   then AOT_obtain z where Pxz: <[<?P>>]zx>
4066     using "∃E"[rotated] by blast
4067   AOT_hence <[λy [<?P>>]zy]x>
4068     by (safe intro!: "β←C" "cqt:2")
4069   AOT_hence <[λy [<?P>>]zy]y>
4070     by (safe intro!: indist[THEN "∨E"(1), THEN "≡E"(1)] "cqt:2")
4071   AOT_hence Pyz: <[<?P>>]zy>
4072     using "β→C"(1) by blast
4073   AOT_hence <∃F∃u ([F]u & Numbers(y,F) & Numbers(z,[F]-u))>
4074     using Pyz pred_equiv[THEN "≡E"(1)] by blast
4075   then AOT_obtain F1 where <∃u ([F1]u & Numbers(y,F1) & Numbers(z,[F1]-u))>
4076     using "∃E"[rotated] by blast
4077   then AOT_obtain u where u_prop: <[F1]u & Numbers(y,F1) & Numbers(z,[F1]-u)>
4078     using "Ordinary.∃E"[rotated] by meson
4079   AOT_have <∃F∃u ([F]u & Numbers(x,F) & Numbers(z,[F]-u))>
4080     using Pxz pred_equiv[THEN "≡E"(1)] by blast
4081   then AOT_obtain F2 where <∃u ([F2]u & Numbers(x,F2) & Numbers(z,[F2]-u))>
4082     using "∃E"[rotated] by blast
4083   then AOT_obtain v where v_prop: <[F2]v & Numbers(x,F2) & Numbers(z,[F2]-v)>
4084     using "Ordinary.∃E"[rotated] by meson
4085   AOT_have <[F2]-v ≈E [F1]-u>
4086     using "hume-strict:1"[unvarify F G, THEN "≡E"(1), OF "F-u[den]",
4087       OF "F-u[den]", OF "∃I"(2)[where β=z], OF "&I"]
4088       v_prop u_prop "&E" by blast
4089   AOT_hence <F2 ≈E F1>
4090     using "P'-eq"[THEN "→E", OF "&I", OF "&I"]
4091       u_prop v_prop "&E" by meson
4092   AOT_hence <x = y>
4093     using "pre-Hume"[THEN "→E", THEN "≡E"(2), OF "&I"]

```

```

4094         v_prop u_prop "&E" by blast
4095     }
4096     text<The second case handles x being equal to zero.>
4097     moreover {
4098         fix u
4099         AOT_assume x_is_zero: <x = 0>
4100         moreover AOT_have <Numbers(0, [λz z =E u]-u)>
4101         proof (safe intro!: "OF:1"[unvarify F, THEN "≡E"(1)] "cqt:2" "raa-cor:2"
4102             "F-u[den]"[unvarify F])
4103             AOT_assume <∃v [[λz z =E u]-u]v>
4104             then AOT_obtain v where <[[λz z =E u]-u]v>
4105                 using "Ordinary.∃E"[rotated] by meson
4106             AOT_hence <[λz z =E u]v & v ≠E u>
4107                 by (auto intro: "F-u"[THEN "=dfE"(1), where τ1τn="(_,_)"] simplified)
4108                     intro!: "cqt:2" "F-u[equiv]"[unvarify F, THEN "≡E"(1)]
4109                         "F-u[den]"[unvarify F])
4110             AOT_thus <p & ¬p> for p
4111                 using "β→C" "thm-neg=E"[THEN "≡E"(1)] "&E" "&I"
4112                         "raa-cor:3" by fast
4113         qed
4114         ultimately AOT_have 0: <Numbers(x, [λz z =E u]-u)>
4115             using "rule=E" id_sym by fast
4116         AOT_have <∃y Numbers(y, [λz z =E u])>
4117             by (safe intro!: "num:1"[unvarify G] "cqt:2")
4118         then AOT_obtain z where <Numbers(z, [λz z =E u])>
4119             using "∃E" by metis
4120         moreover AOT_have <[λz z =E u]u>
4121             by (safe intro!: "β←C" "cqt:2" "ord=Eequiv:1"[THEN "→E"] Ordinary.ψ)
4122         ultimately AOT_have
4123             1: <[λz z =E u]u & Numbers(z, [λz z =E u]) & Numbers(x, [λz z =E u]-u)>
4124             using 0 "&I" by auto
4125         AOT_hence <∃v([λz z =E u]v & Numbers(z, [λz z =E u]) & Numbers(x, [λz z =E u]-v)>
4126             by (rule "Ordinary.∃I")
4127         AOT_hence <∃F∃u([F]u & Numbers(z, [F]) & Numbers(x, [F]-u)>
4128             by (rule "∃I"; "cqt:2")
4129         AOT_hence Px1: <[«?P»]xz>
4130             using "beta-C-cor:2"[THEN "→E", OF pred_den,
4131                 THEN tuple_forall[THEN "≡dfE"], THEN "∀E"(2),
4132                 THEN "∀E"(2), THEN "≡E"(2)] by simp
4133         AOT_hence <[λy [«?P»]yz]x>
4134             by (safe intro!: "β←C" "cqt:2")
4135         AOT_hence <[λy [«?P»]yz]y>
4136             by (safe intro!: indist[THEN "∀E"(1), THEN "≡E"(1)] "cqt:2")
4137         AOT_hence Py1: <[«?P»]yz>
4138             using "β→C" by blast
4139         AOT_hence <∃F∃u([F]u & Numbers(z, [F]) & Numbers(y, [F]-u)>
4140             using "β→C" by fast
4141         then AOT_obtain G where <∃u([G]u & Numbers(z, [G]) & Numbers(y, [G]-u)>
4142             using "∃E"[rotated] by blast
4143         then AOT_obtain v where 2: <[G]v & Numbers(z, [G]) & Numbers(y, [G]-v)>
4144             using "Ordinary.∃E"[rotated] by meson
4145         with 1 2 AOT_have <[λz z =E u] ≈E G>
4146             by (auto intro!: "hume-strict:1"[unvarify F, THEN "≡E"(1), rotated,
4147                 OF "∃I"(2)[where β=z], OF "&I"] "cqt:2"
4148                 dest: "&E")
4149         AOT_hence 3: <[λz z =E u]-u ≈E [G]-v>
4150             using 1 2
4151             by (safe_step intro!: "eqP"[unvarify F, THEN "→E"])
4152                 (auto dest: "&E" intro!: "cqt:2" "&I")
4153         with 1 2 AOT_have <x = y>
4154             by (auto intro!: "pre-Hume"[unvarify G H, THEN "→E",
4155                 THEN "≡E"(2), rotated 3, OF 3]
4156                 "F-u[den]"[unvarify F] "cqt:2" "&I")

```

```

4157         dest: "&E")
4158     }
4159     ultimately AOT_have <x = y>
4160     using "VE"(1) "→I" "reductio-aa:1" by blast
4161     text<Now since x numbers F, so does y.>
4162     AOT_hence <Numbers(y,F)>
4163     using numxF "rule=E" by fast
4164 } note 0 = this
4165 text<The only thing left is to generalize this result to a biconditional.>
4166 AOT_modally_strict {
4167     fix x y
4168     AOT_assume <[«?P»]↓>
4169     moreover AOT_assume <∀F([F]x ≡ [F]y)>
4170     moreover AOT_have <∀F([F]y ≡ [F]x)>
4171     by (metis "cqt-basic:11" "intro-elim:3:a" calculation(2))
4172     ultimately AOT_show <Numbers(x,F) ≡ Numbers(y,F)>
4173     using 0 "≡I" "→I" by auto
4174 }
4175 qed
4176 ultimately AOT_show <[λx Numbers(x,F)]↓>
4177 using "kirchner-thm:1"[THEN "≡E"(2)] "→E" by fast
4178 next
4179 text<The converse can be shown by coexistence.>
4180 AOT_assume <∀F [λx Numbers(x,F)]↓>
4181 AOT_hence <[λx Numbers(x,F)]↓> for F
4182 using "VE"(2) by blast
4183 AOT_hence <□[λx Numbers(x,F)]↓> for F
4184 using "exist-nec"[THEN "→E"] by blast
4185 AOT_hence <∀F □[λx Numbers(x,F)]↓>
4186 by (rule GEN)
4187 AOT_hence <□∀F [λx Numbers(x,F)]↓>
4188 using BF[THEN "→E"] by fast
4189 moreover AOT_have
4190 <□∀F [λx Numbers(x,F)]↓ →
4191   □∀x ∀y (∃F ∃u ([F]u & [λz Numbers(z,F)]y & [λz Numbers(z,[F]-u)]x) ≡
4192     ∃F ∃u ([F]u & Numbers(y,F) & Numbers(x,[F]-u)))>
4193 proof(rule RM; safe intro!: "→I" GEN)
4194   AOT_modally_strict {
4195     fix x y
4196     AOT_assume 0: <∀F [λx Numbers(x,F)]↓>
4197     AOT_show <∃F ∃u ([F]u & [λz Numbers(z,F)]y & [λz Numbers(z,[F]-u)]x) ≡
4198       ∃F ∃u ([F]u & Numbers(y,F) & Numbers(x,[F]-u))>
4199     proof(safe intro!: "≡I" "→I")
4200       AOT_assume <∃F ∃u ([F]u & [λz Numbers(z,F)]y & [λz Numbers(z,[F]-u)]x)>
4201       then AOT_obtain F where
4202         <∃u ([F]u & [λz Numbers(z,F)]y & [λz Numbers(z,[F]-u)]x)>
4203         using "∃E"[rotated] by blast
4204       then AOT_obtain u where <[F]u & [λz Numbers(z,F)]y & [λz Numbers(z,[F]-u)]x>
4205         using "Ordinary.∃E"[rotated] by meson
4206       AOT_hence <[F]u & Numbers(y,F) & Numbers(x,[F]-u)>
4207         by (auto intro!: "&I" "dest: "&E" "β→C")
4208       AOT_thus <∃F ∃u ([F]u & Numbers(y,F) & Numbers(x,[F]-u))>
4209         using "∃I" "Ordinary.∃I" by fast
4210     next
4211     AOT_assume <∃F ∃u ([F]u & Numbers(y,F) & Numbers(x,[F]-u))>
4212     then AOT_obtain F where <∃u ([F]u & Numbers(y,F) & Numbers(x,[F]-u))>
4213     using "∃E"[rotated] by blast
4214     then AOT_obtain u where <[F]u & Numbers(y,F) & Numbers(x,[F]-u)>
4215     using "Ordinary.∃E"[rotated] by meson
4216     AOT_hence <[F]u & [λz Numbers(z,F)]y & [λz Numbers(z,[F]-u)]x>
4217     by (auto intro!: "&I" "β←C" 0[THEN "VE"(1)] "F-u[den]"
4218       dest: "&E" intro: "cqt:2")
4219     AOT_hence <∃u([F]u & [λz Numbers(z,F)]y & [λz Numbers(z,[F]-u)]x)>

```



```

4220     by (rule "Ordinary.∃I")
4221     AOT_thus <∃F∃u ([F]u & [λz Numbers(z,F)]y & [λz Numbers(z,[F]-u)]x)>
4222     by (rule "∃I")
4223   qed
4224 }
4225 qed
4226 ultimately AOT_have
4227 <□∀x ∀y (∃F ∃u ([F]u & [λz Numbers(z,F)]y & [λz Numbers(z,[F]-u)]x) ≡
4228   ∃F ∃u ([F]u & Numbers(y,F) & Numbers(x,[F]-u))>
4229   using "→E" by blast
4230 AOT_thus <[λxy ∃F ∃u ([F]u & Numbers(y,F) & Numbers(x,[F]-u)]↓>
4231   by (rule "safe-ext[2]"[axiom_inst, THEN "→E", OF "&I", rotated]) "cqt:2"
4232 qed
4233
4234 text<The following is not part of PLM, but a consequence of extended relation
4235   comprehension and can be used to @{emph <derive>} the predecessor axiom.>
4236 AOT_theorem numbers_prop_den: <[λx Numbers(x,G)]↓>
4237 proof (rule "safe-ext"[axiom_inst, THEN "→E", OF "&I"])
4238   AOT_show <[λx A!x & [λx ∀F (x[F] ≡ [λz A[F]z] ≈E G)]x]↓>
4239     by "cqt:2"
4240 next
4241   AOT_have 0: <⊢□ [λx ∀F (x[F] ≡ [λz A[F]z] ≈E G)]↓>
4242   proof (safe intro!: Comprehension_3[THEN "→E"] "→I" RN GEN)
4243     AOT_modally_strict {
4244       fix F H
4245       AOT_assume <□H ≡E F>
4246       AOT_hence <□∀u ([H]u ≡ [F]u)>
4247         by (AOT_subst (reverse) <∀u ([H]u ≡ [F]u)> <H ≡E F>)
4248           (safe intro!: "eqE"[THEN "≡df", THEN "≡S"(1), OF "&I"] "cqt:2")
4249       AOT_hence <∀u □([H]u ≡ [F]u)>
4250         by (metis "Ordinary.res-var-bound-reas[CBF]" "→E")
4251       AOT_hence <□([H]u ≡ [F]u)> for u
4252         using "Ordinary.∀E" by fast
4253       AOT_hence <A([H]u ≡ [F]u)> for u
4254         by (metis "nec-imp-act" "→E")
4255       AOT_hence <A([F]u ≡ [H]u)> for u
4256         by (metis "Act-Basic:5" "Commutativity of ≡" "intro-elim:3:b")
4257       AOT_hence <[λz A[F]z] ≡E [λz A[H]z]>
4258         by (safe intro!: "eqE"[THEN "≡dfI"] "&I" "cqt:2" Ordinary.GEN;
4259           AOT_subst <[λz A[F]z]u> <A[F]u> for: u F)
4260         (auto intro!: "beta-C-meta"[THEN "→E"] "cqt:2"
4261           "Act-Basic:5"[THEN "≡E"(1)])
4262       AOT_hence <[λz A[F]z] ≈E [λz A[H]z]>
4263         by (safe intro!: "apE-eqE:1"[unvarify F G, THEN "→E"] "cqt:2")
4264       AOT_thus <[λz A[F]z] ≈E G ≡ [λz A[H]z] ≈E G>
4265         using "≡I" "eq-part:2[terms]" "eq-part:3[terms]" "→E" "→I"
4266         by metis
4267     }
4268   qed
4269   AOT_show <□∀x (A!x & [λx ∀F (x[F] ≡ [λz A[F]z] ≈E G)]x ≡ Numbers(x,G))>
4270   proof (safe intro!: RN GEN)
4271     AOT_modally_strict {
4272       fix x
4273       AOT_show <A!x & [λx ∀F (x[F] ≡ [λz A[F]z] ≈E G)]x ≡ Numbers(x,G)>
4274         by (AOT_subst_def numbers; AOT_subst_thm "beta-C-meta"[THEN "→E", OF 0])
4275         (auto intro!: "beta-C-meta"[THEN "→E", OF 0] "≡I" "→I" "&I" "cqt:2"
4276           dest: "&E")
4277     }
4278   qed
4279 qed
4280
4281 text<The two theorems above allow us to derive
4282   the predecessor axiom of PLM as theorem.>

```



```

4283
4284 AOT_theorem pred: <[ $\lambda xy \exists F \exists u ([F]u \ \& \ \text{Numbers}(y,F) \ \& \ \text{Numbers}(x, [F]^{-u})) \downarrow$ > (782)
4285   using pred_coex numbers_prop_den[" $\forall I$ " G] "≡E" by blast
4286
4287 AOT_define Predecessor :: <[I] <[P]>
4288   "pred-thm:1": (783.1)
4289   <[P] =df [ $\lambda xy \exists F \exists u ([F]u \ \& \ \text{Numbers}(y,F) \ \& \ \text{Numbers}(x, [F]^{-u}))$ >
4290
4291 AOT_theorem "pred-thm:2": <[P]↓> (783.2)
4292   using pred "pred-thm:1" "rule-id-df:2:b[zero]" by blast
4293
4294 AOT_theorem "pred-thm:3": (783.3)
4295   <[[P]xy ≡  $\exists F \exists u ([F]u \ \& \ \text{Numbers}(y,F) \ \& \ \text{Numbers}(x, [F]^{-u}))$ >
4296     by (auto intro!: "beta-C-meta"[unvarify  $\nu_1 \nu_n$ , where  $\tau = \langle \_ , \_ \rangle$ , THEN " $\rightarrow E$ ",
4297         rotated, OF pred, simplified]
4298         tuple_denotes[THEN "dfI"] "&I" "cqt:2" pred
4299         intro: "dfI"(2)[OF "pred-thm:1"])
4300
4301 AOT_theorem "pred-1-1:1": <[[P]xy  $\rightarrow \square$ [[P]xy> (784.1)
4302 proof(rule " $\rightarrow I$ ")
4303   AOT_assume <[[P]xy>
4304   AOT_hence < $\exists F \exists u ([F]u \ \& \ \text{Numbers}(y,F) \ \& \ \text{Numbers}(x, [F]^{-u}))$ >
4305     using "≡E"(1) "pred-thm:3" by fast
4306   then AOT_obtain F where < $\exists u ([F]u \ \& \ \text{Numbers}(y,F) \ \& \ \text{Numbers}(x, [F]^{-u}))$ >
4307     using "∃E"[rotated] by blast
4308   then AOT_obtain u where props: <[F]u  $\& \ \text{Numbers}(y,F) \ \& \ \text{Numbers}(x, [F]^{-u})$ >
4309     using "Ordinary.∃E"[rotated] by meson
4310   AOT_obtain G where Ridigifies_G_F: <Rigidifies(G, F)>
4311     by (metis "instantiation" "rigid-der:3")
4312   AOT_hence  $\xi$ : < $\square \forall x ([G]x \rightarrow \square [G]x)$ > and  $\zeta$ : < $\forall x ([G]x \equiv [F]x)$ >
4313     using "df-rigid-rel:2"[THEN "dfE", THEN "&E"(1),
4314         THEN "dfE"[OF "df-rigid-rel:1"], THEN "&E"(2)]
4315     "df-rigid-rel:2"[THEN "dfE", THEN "&E"(2)] by blast+
4316
4317 AOT_have rigid_num_nec: <Numbers(x,F)  $\& \ \text{Rigidifies}(G,F) \rightarrow \square$ Numbers(x,G)>
4318   for x G F
4319 proof(rule " $\rightarrow I$ "; frule "&E"(1); drule "&E"(2))
4320   fix G F x
4321   AOT_assume Numbers_xF: <Numbers(x,F)>
4322   AOT_assume <Rigidifies(G,F)>
4323   AOT_hence  $\xi$ : <Rigid(G)> and  $\zeta$ : < $\forall x ([G]x \equiv [F]x)$ >
4324     using "df-rigid-rel:2"[THEN "dfE"] "&E" by blast+
4325   AOT_thus < $\square$ Numbers(x,G)>
4326   proof (safe intro!:
4327     "num-cont:2"[THEN " $\rightarrow E$ ", OF  $\xi$ , THEN "qml:2"[axiom_inst, THEN " $\rightarrow E$ "],
4328     THEN " $\forall E$ "(2), THEN " $\rightarrow E$ "]
4329     "num-tran:3"[THEN " $\rightarrow E$ ", THEN "≡E"(1), rotated, OF Numbers_xF]
4330     eqE[THEN "dfI"]
4331     "&I" "cqt:2[const_var]"[axiom_inst] Ordinary.GEN " $\rightarrow I$ ")
4332     AOT_show <[F]u ≡ [G]u> for u
4333       using  $\zeta$ [THEN " $\forall E$ "(2)] by (metis "≡E"(6) "oth-class-taut:3:a")
4334   qed
4335   qed
4336   AOT_have < $\square$ Numbers(y,G)>
4337     using rigid_num_nec[THEN " $\rightarrow E$ ", OF "&I", OF props[THEN "&E"(1), THEN "&E"(2)],
4338     OF Ridigifies_G_F].
4339   moreover {
4340     AOT_have <Rigidifies([G]-u, [F]-u)>
4341     proof (safe intro!: "df-rigid-rel:1"[THEN "dfI"] "df-rigid-rel:2"[THEN "dfI"]
4342         "&I" "F-u[den]" GEN "≡I" " $\rightarrow I$ ")
4343       AOT_have < $\square \forall x ([G]x \rightarrow \square [G]x) \rightarrow \square \forall x ([G]^{-u}x \rightarrow \square [[G]^{-u}]x)$ >
4344       proof (rule RM; safe intro!: " $\rightarrow I$ " GEN)
4345         AOT_modally_strict {

```

```

4346     fix x
4347     AOT_assume 0: <∀x([G]x → □[G]x)>
4348     AOT_assume 1: <[[G]-u]x>
4349     AOT_have <[λx [G]x & x ≠E u]x>
4350     apply (rule "F-u"[THEN "=dfE"(1), where τ1τn"(_,_)"], simplified])
4351     apply "cqt:2[lambda]"
4352     by (fact 1)
4353     AOT_hence <[G]x & x ≠E u>
4354     by (rule "β→C"(1))
4355     AOT_hence 2: <□[G]x> and 3: <□x ≠E u>
4356     using "&E" 0[THEN "∀E"(2), THEN "→E"] "id-nec4:1" "≡E"(1) by blast+
4357     AOT_show <□[[G]-u]x>
4358     apply (AOT_subst <[[G]-u]x> <[G]x & x ≠E u>)
4359     apply (rule "F-u"[THEN "=dfI"(1), where τ1τn"(_,_)"], simplified])
4360     apply "cqt:2[lambda]"
4361     apply (rule "beta-C-meta"[THEN "→E"])
4362     apply "cqt:2[lambda]"
4363     using 2 3 "KBasic:3" "≡S"(2) "≡E"(2) by blast
4364   }
4365   qed
4366   AOT_thus <□∀x([[G]-u]x → □[[G]-u]x)> using ξ "→E" by blast
4367 next
4368   fix x
4369   AOT_assume <[[G]-u]x>
4370   AOT_hence <[λx [G]x & x ≠E u]x>
4371   by (auto intro: "F-u"[THEN "=dfE"(1), where τ1τn"(_,_)"], simplified])
4372   intro!: "cqt:2")
4373   AOT_hence <[G]x & x ≠E u>
4374   by (rule "β→C"(1))
4375   AOT_hence <[F]x & x ≠E u>
4376   using ζ "&I" "&E"(1) "&E"(2) "≡E"(1) "rule-ui:3" by blast
4377   AOT_hence <[λx [F]x & x ≠E u]x>
4378   by (auto intro!: "β←C"(1) "cqt:2")
4379   AOT_thus <[[F]-u]x>
4380   by (auto intro: "F-u"[THEN "=dfI"(1), where τ1τn"(_,_)"], simplified])
4381   intro!: "cqt:2")
4382 next
4383   fix x
4384   AOT_assume <[[F]-u]x>
4385   AOT_hence <[λx [F]x & x ≠E u]x>
4386   by (auto intro: "F-u"[THEN "=dfE"(1), where τ1τn"(_,_)"], simplified])
4387   intro!: "cqt:2")
4388   AOT_hence <[F]x & x ≠E u>
4389   by (rule "β→C"(1))
4390   AOT_hence <[G]x & x ≠E u>
4391   using ζ "&I" "&E"(1) "&E"(2) "≡E"(2) "rule-ui:3" by blast
4392   AOT_hence <[λx [G]x & x ≠E u]x>
4393   by (auto intro!: "β←C"(1) "cqt:2")
4394   AOT_thus <[[G]-u]x>
4395   by (auto intro: "F-u"[THEN "=dfI"(1), where τ1τn"(_,_)"], simplified])
4396   intro!: "cqt:2")
4397   qed
4398   AOT_hence <□Numbers(x, [G]-u)>
4399   using rigid_num_nec[unvarify F G, OF "F-u[den]", OF "F-u[den]", THEN "→E",
4400     OF "&I", OF props[THEN "&E"(2)]] by blast
4401 }
4402 moreover AOT_have <□[G]u>
4403   using props[THEN "&E"(1), THEN "&E"(1), THEN ζ[THEN "∀E"(2), THEN "≡E"(2)]]
4404   ξ[THEN "qml:2"[axiom_inst, THEN "→E"], THEN "∀E"(2), THEN "→E"]
4405   by blast
4406 ultimately AOT_have <□([G]u & Numbers(y, G) & Numbers(x, [G]-u)>
4407   by (metis "KBasic:3" "&I" "≡E"(2))
4408 AOT_hence <∃u (□([G]u & Numbers(y, G) & Numbers(x, [G]-u)))>

```

```

4409   by (rule "Ordinary.∃I")
4410 AOT_hence <□∃u ([G]u & Numbers(y,G) & Numbers(x,[G]-u)>
4411   using "Ordinary.res-var-bound-reas[Buridan]" "→E" by fast
4412 AOT_hence <∃F □∃u ([F]u & Numbers(y,F) & Numbers(x,[F]-u)>
4413   by (rule "∃I")
4414 AOT_hence 0: <□∃F∃u ([F]u & Numbers(y,F) & Numbers(x,[F]-u)>
4415   using Buridan "vdash-properties:10" by fast
4416 AOT_show <□[P]xy>
4417   by (AOT_subst <[P]xy> <∃F∃u ([F]u & Numbers(y,F) & Numbers(x,[F]-u)>;
4418     simp add: "pred-thm:3" 0)
4419 qed
4420
4421 AOT_theorem "pred-1-1:2": <Rigid(ℙ)> (784.2)
4422   by (safe intro!: "df-rigid-rel:1"[THEN "≡dfI"] "pred-thm:2" "&I"
4423     RN tuple_forall[THEN "≡dfI"];
4424     safe intro!: GEN "pred-1-1:1")
4425
4426 AOT_theorem "pred-1-1:3": <1-1(ℙ)> (784.3)
4427 proof (safe intro!: "df-1-1:1"[THEN "≡dfI"] "pred-thm:2" "&I" GEN "→I";
4428   frule "&E"(1); drule "&E"(2))
4429   fix x y z
4430   AOT_assume <[P]xz>
4431   AOT_hence <∃F∃u ([F]u & Numbers(z,F) & Numbers(x,[F]-u)>
4432     using "pred-thm:3"[THEN "≡E"(1)] by blast
4433   then AOT_obtain F where <∃u ([F]u & Numbers(z,F) & Numbers(x,[F]-u)>
4434     using "∃E"[rotated] by blast
4435   then AOT_obtain u where u_prop: <[F]u & Numbers(z,F) & Numbers(x,[F]-u)>
4436     using "Ordinary.∃E"[rotated] by meson
4437   AOT_assume <[P]yz>
4438   AOT_hence <∃F∃u ([F]u & Numbers(z,F) & Numbers(y,[F]-u)>
4439     using "pred-thm:3"[THEN "≡E"(1)] by blast
4440   then AOT_obtain G where <∃u ([G]u & Numbers(z,G) & Numbers(y,[G]-u)>
4441     using "∃E"[rotated] by blast
4442   then AOT_obtain v where v_prop: <[G]v & Numbers(z,G) & Numbers(y,[G]-v)>
4443     using "Ordinary.∃E"[rotated] by meson
4444   AOT_show <x = y>
4445   proof (rule "pre-Hume"[unvarify G H, OF "F-u[den]", OF "F-u[den]",
4446     THEN "→E", OF "&I", THEN "≡E"(2)])
4447     AOT_show <Numbers(x, [F]-u)>
4448     using u_prop "&E" by blast
4449   next
4450     AOT_show <Numbers(y, [G]-v)>
4451     using v_prop "&E" by blast
4452   next
4453     AOT_have <F ≈E G>
4454     using u_prop[THEN "&E"(1), THEN "&E"(2)]
4455     using v_prop[THEN "&E"(1), THEN "&E"(2)]
4456     using "num-tran:2"[THEN "→E", OF "&I"] by blast
4457     AOT_thus <[F]-u ≈E [G]-v>
4458     using u_prop[THEN "&E"(1), THEN "&E"(1)]
4459     using v_prop[THEN "&E"(1), THEN "&E"(1)]
4460     using eqP'[THEN "→E", OF "&I", OF "&I"]
4461     by blast
4462   qed
4463 qed
4464
4465 AOT_theorem "pred-1-1:4": <Rigid1-1(ℙ)> (784.4)
4466   by (meson "≡dfI" "&I" "df-1-1:2" "pred-1-1:2" "pred-1-1:3")
4467
4468 AOT_theorem "assume-anc:1": (785.1)
4469   <[P]* = [λxy ∀F((∀z([P]xz → [F]z) & Hereditary(F,ℙ)) → [F]y)]>
4470   apply (rule "≡dfI"(1)[OF "ances-df"])
4471   apply "cqt:2[lambda]"

```

```

4472   apply (rule "=I"(1))
4473   by "cqt:2[lambda]"
4474
4475 AOT_theorem "assume-anc:2": <[P*]↓> (785.2)
4476   using "t=t-proper:1" "assume-anc:1" "vdash-properties:10" by blast
4477
4478 AOT_theorem "assume-anc:3": (785.3)
4479   <[P*]xy ≡ ∀F((∀z([P]xz → [F]z) & ∀x'∀y'([P]x'y' → ([F]x' → [F]y')))) → [F]y)>
4480 proof -
4481   AOT_have prod_den: <⊢□ «(AOT_term_of_var x1,AOT_term_of_var x2)»↓>
4482     for x1 x2 :: <κ AOT_var>
4483     by (simp add: "&I" "ex:1:a" prod_denotesI "rule-ui:3")
4484   AOT_have den: <[λxy ∀F((∀z([P]xz → [F]z) & Hereditary(F,P)) → [F]y)]↓>
4485     by "cqt:2[lambda]"
4486   AOT_have 1: <[P*]xy ≡ ∀F((∀z([P]xz → [F]z) & Hereditary(F,P)) → [F]y)>
4487     apply (rule "rule=E"[rotated, OF "assume-anc:1"[symmetric]])
4488     by (rule "beta-C-meta"[unvarify ν1νn, OF prod_den, THEN "→E",
4489         simplified, OF den, simplified])
4490   show ?thesis
4491     apply (AOT_subst (reverse) <∀x'∀y' ([P]x'y' → ([F]x' → [F]y'))>
4492         <Hereditary(F,P)> for: F :: <<κ>>)
4493     using "hered:1"[THEN "≡Df", THEN "≡S"(1), OF "&I", OF "pred-thm:2",
4494         OF "cqt:2[const_var]"[axiom_inst]] apply blast
4495     by (fact 1)
4496 qed
4497
4498 AOT_theorem "no-pred-0:1": <¬∃x [P]x 0> (786.1)
4499 proof(rule "raa-cor:2")
4500   AOT_assume <∃x [P]x 0>
4501   then AOT_obtain a where <[P]a 0>
4502     using "∃E"[rotated] by blast
4503   AOT_hence <∃F∃u ([F]u & Numbers(0, F) & Numbers(a, [F]-u)>
4504     using "pred-thm:3"[unvarify y, OF "zero:2", THEN "≡E"(1)] by blast
4505   then AOT_obtain F where <∃u ([F]u & Numbers(0, F) & Numbers(a, [F]-u)>
4506     using "∃E"[rotated] by blast
4507   then AOT_obtain u where <[F]u & Numbers(0, F) & Numbers(a, [F]-u)>
4508     using "Ordinary.∃E"[rotated] by meson
4509   AOT_hence <[F]u> and num0_F: <Numbers(0, F)>
4510     using "&E" "&I" by blast+
4511   AOT_hence <∃u [F]u>
4512     using "Ordinary.∃I" by fast
4513   moreover AOT_have <¬∃u [F]u>
4514     using num0_F "≡E"(2) "OF:1" by blast
4515   ultimately AOT_show <p & ¬p> for p
4516     by (metis "raa-cor:3")
4517 qed
4518
4519 AOT_theorem "no-pred-0:2": <¬∃x [P*]x 0> (786.2)
4520 proof(rule "raa-cor:2")
4521   AOT_assume <∃x [P*]x 0>
4522   then AOT_obtain a where <[P*]a 0>
4523     using "∃E"[rotated] by blast
4524   AOT_hence <∃z [P]z 0>
4525     using "anc-her:5"[unvarify R y, OF "zero:2",
4526         OF "pred-thm:2", THEN "→E"] by auto
4527   AOT_thus <∃z [P]z 0 & ¬∃z [P]z 0>
4528     by (metis "no-pred-0:1" "raa-cor:3")
4529 qed
4530
4531 AOT_theorem "no-pred-0:3": <¬[P*]0 0> (786.3)
4532   by (metis "existential:1" "no-pred-0:2" "reductio-aa:1" "zero:2")
4533
4534 AOT_theorem "assume1:1": <(=P) = [λxy ∃z ([P]xz & [P]yz)]> (787.1)

```

```

4535 apply (rule "=dfI"(1)[OF "id-d-R"])
4536   apply "cqt:2[lambda]"
4537 apply (rule "=I"(1))
4538 by "cqt:2[lambda]"
4539
4540 AOT_theorem "assume1:2": <x =P y ≡ ∃z ([P]xz & [P]yz)> (787.2)
4541 proof (rule "rule=E"[rotated, OF "assume1:1"[symmetric]])
4542   AOT_have prod_den: <⊢□ «(AOT_term_of_var x1, AOT_term_of_var x2)»>↓>
4543     for x1 x2 :: <κ AOT_var>
4544     by (simp add: "&I" "ex:1:a" prod_denotesI "rule-ui:3")
4545   AOT_have 1: <[λxy ∃z ([P]xz & [P]yz)]>↓>
4546     by "cqt:2"
4547   AOT_show <[λxy ∃z ([P]xz & [P]yz)]xy ≡ ∃z ([P]xz & [P]yz)>
4548     using "beta-C-meta"[THEN "→E", OF 1, unvarify ν1νn,
4549       OF prod_den, simplified] by blast
4550 qed
4551
4552 AOT_theorem "assume1:3": <[P]+ = [λxy [P]*xy ∨ x =P y]> (787.3)
4553   apply (rule "=dfI"(1)[OF "w-ances-df"])
4554   apply (simp add: "w-ances-df[den1]")
4555   apply (rule "rule=E"[rotated, OF "assume1:1"[symmetric]])
4556   apply (rule "=dfI"(1)[OF "id-d-R"])
4557   apply "cqt:2[lambda]"
4558   apply (rule "=I"(1))
4559   by "cqt:2[lambda]"
4560
4561 AOT_theorem "assume1:4": <[P]+> (787.4)
4562   using "w-ances-df[den2]".
4563
4564 AOT_theorem "assume1:5": <[P]+xy ≡ [P]*xy ∨ x =P y> (787.5)
4565 proof -
4566   AOT_have 0: <[λxy [P]*xy ∨ x =P y]>↓> by "cqt:2"
4567   AOT_have prod_den: <⊢□ «(AOT_term_of_var x1, AOT_term_of_var x2)»>↓>
4568     for x1 x2 :: <κ AOT_var>
4569     by (simp add: "&I" "ex:1:a" prod_denotesI "rule-ui:3")
4570   show ?thesis
4571     apply (rule "rule=E"[rotated, OF "assume1:3"[symmetric]])
4572     using "beta-C-meta"[THEN "→E", OF 0, unvarify ν1νn, OF prod_den, simplified]
4573     by (simp add: cond_case_prod_eta)
4574 qed
4575
4576 AOT_define NaturalNumber :: <τ> (<N>)
4577   "nnumber:1": <N =df [λx [P]+0x> (788.1)
4578
4579 AOT_theorem "nnumber:2": <N>↓> (788.2)
4580   by (rule "=dfI"(2)[OF "nnumber:1"; "cqt:2[lambda]"])
4581
4582 AOT_theorem "nnumber:3": <[N]x ≡ [P]+0x> (788.3)
4583   apply (rule "=dfI"(2)[OF "nnumber:1"])
4584   apply "cqt:2[lambda]"
4585   apply (rule "beta-C-meta"[THEN "→E"])
4586   by "cqt:2[lambda]"
4587
4588 AOT_theorem "0-n": <[N]0> (789)
4589 proof (safe intro!: "nnumber:3"[unvarify x, OF "zero:2", THEN "≡E"(2)])
4590   "assume1:5"[unvarify x y, OF "zero:2", OF "zero:2", THEN "≡E"(2)]
4591   "∀I"(2) "assume1:2"[unvarify x y, OF "zero:2", OF "zero:2", THEN "≡E"(2))]
4592   fix u
4593   AOT_have den: <[λx 0!x & x =E u]>↓> by "cqt:2[lambda]"
4594   AOT_obtain a where a_prop: <Numbers(a, [λx 0!x & x =E u])>
4595     using "num:1"[unvarify G, OF den] "∃E"[rotated] by blast
4596   AOT_have <[P]0a>
4597   proof (safe intro!: "pred-thm:3"[unvarify x, OF "zero:2", THEN "≡E"(2)])

```

```

4598           "∃I"(1)[where τ=«[λx 0!x & x =E u]»>>]
4599           "Ordinary.∃I"[where β=u] "&I" den
4600           "OF:1"[unvarify F, OF "F-u[den]", unvarify F,
4601           OF den, THEN "≡E"(1))]
4602   AOT_show <[λx [0!]x & x =E u]>
4603     by (auto intro!: "β←C"(1) "cqt:2" "&I" "ord=Eequiv:1"[THEN "→E"]
4604         Ordinary.ψ)
4605   next
4606     AOT_show <Numbers(a,[λx [0!]x & x =E u])>
4607       using a_prop.
4608   next
4609     AOT_show <¬∃v [[λx [0!]x & x =E u]u]v>
4610     proof(rule "raa-cor:2")
4611       AOT_assume <∃v [[λx [0!]x & x =E u]u]v>
4612       then AOT_obtain v where <[[λx [0!]x & x =E u]u]v>
4613         using "Ordinary.∃E"[rotated] "&E" by blast
4614       AOT_hence <[λz [λx [0!]x & x =E u]z & z ≠E u]v>
4615         apply (rule "F-u"[THEN "=dfE"(1), where τ1τn="(_,_)"] simplified, rotated])
4616         by "cqt:2[lamba]"
4617       AOT_hence <[λx [0!]x & x =E u]v & v ≠E u>
4618         by (rule "β→C"(1))
4619       AOT_hence <v =E u> and <v ≠E u>
4620         using "β→C"(1) "&E" by blast+
4621       AOT_hence <v =E u & ¬(v =E u)>
4622         by (metis "≡E"(4) "reductio-aa:1" "thm-neg=E")
4623       AOT_thus <p & ¬p> for p
4624         by (metis "raa-cor:1")
4625     qed
4626   qed
4627   AOT_thus <∃z ([P]0z & [P]0z)>
4628     by (safe intro!: "&I" "∃I"(2)[where β=a])
4629   qed
4630
4631   AOT_theorem "mod-col-num:1": <[N]x → □[N]x> (790.1)
4632   proof(rule "→I")
4633     AOT_have nec0N: <[λx □[N]x]0>
4634       by (auto intro!: "β←C"(1) "cqt:2" simp: "zero:2" RN "0-n")
4635     AOT_have 1: <[λx □[N]x]0 &
4636       ∀x∀y ([P]+0x & [P]+10y → ([P]xy → ([λx □[N]x]x → [λx □[N]x]y))) →
4637       ∀x ([P]+0x → [λx □[N]x]x)>
4638       by (auto intro!: "cqt:2"
4639           intro: "pre-ind"[unconstrain  $\mathcal{R}$ , unvarify  $\beta$ , OF "pred-thm:2",
4640               THEN "→E", OF "pred-1-1:4", unvarify z, OF "zero:2",
4641               unvarify F])
4642     AOT_have <∀x ([P]+0x → [λx □[N]x]x)>
4643     proof (rule 1[THEN "→E"]; safe intro!: "&I" GEN "→I" nec0N;
4644           frule "&E"(1); drule "&E"(2))
4645       fix x y
4646       AOT_assume <[P]xy>
4647       AOT_hence 0: <□[P]xy>
4648         by (metis "pred-1-1:1" "→E")
4649       AOT_assume <[λx □[N]x]x>
4650       AOT_hence <□[N]x>
4651         by (rule "β→C"(1))
4652       AOT_hence <□([P]xy & [N]x)>
4653         by (metis "0" "KBasic:3" Adjunction "≡E"(2) "→E")
4654     moreover AOT_have <□([P]xy & [N]x) → □[N]y>
4655     proof (rule RM; rule "→I"; frule "&E"(1); drule "&E"(2))
4656       AOT_modally_strict {
4657         AOT_assume 0: <[P]xy>
4658         AOT_assume <[N]x>
4659         AOT_hence 1: <[P]+0x>
4660         by (metis "≡E"(1) "nnumber:3")

```

```

4661   AOT_show <[N]y>
4662     apply (rule "nnumber:3"[THEN "≡E"(2)])
4663     apply (rule "assume1:5"[unvarify x, OF "zero:2", THEN "≡E"(2)])
4664     apply (rule "∀I"(1))
4665     apply (rule "w-ances-her:3"[unconstrain  $\mathcal{R}$ , unvarify  $\beta$ , OF "pred-thm:2",
4666                                     THEN "→E", OF "pred-1-1:4", unvarify x,
4667                                     OF "zero:2", THEN "→E"])
4668     apply (rule "&I")
4669     apply (fact 1)
4670     by (fact 0)
4671   }
4672   qed
4673   ultimately AOT_have <□[N]y>
4674     by (metis "→E")
4675   AOT_thus <[λx □[N]x]y>
4676     by (auto intro!: "β←C"(1) "cqt:2")
4677   qed
4678   AOT_hence 0: <[[P]+0x → [λx □[N]x]x>
4679     using "∀E"(2) by blast
4680   AOT_assume <[N]x>
4681   AOT_hence <[[P]+0x>
4682     by (metis "≡E"(1) "nnumber:3")
4683   AOT_hence <[λx □[N]x]x>
4684     using 0[THEN "→E"] by blast
4685   AOT_thus <□[N]x>
4686     by (rule "β→C"(1))
4687   qed
4688
4689   AOT_theorem "mod-col-num:2": <Rigid(N)> (790.2)
4690     by (safe intro!: "df-rigid-rel:1"[THEN "≡dfI"] "&I" RN GEN
4691         "mod-col-num:1" "nnumber:2")
4692
4693   AOT_register_rigid_restricted_type
4694     Number: <[N]κ>
4695   proof
4696     AOT_modally_strict {
4697       AOT_show <∃x [N]x>
4698         by (rule "∃I"(1)[where τ=<<0>>]; simp add: "0-n" "zero:2")
4699     }
4700   next
4701     AOT_modally_strict {
4702       AOT_show <[N]κ → κ↓> for κ
4703         by (simp add: "→I" "cqt:5:a[1]"[axiom_inst, THEN "→E", THEN "&E"(2)])
4704     }
4705   next
4706     AOT_modally_strict {
4707       AOT_show <∀x([N]x → □[N]x)>
4708         by (simp add: GEN "mod-col-num:1")
4709     }
4710   qed
4711   AOT_register_variable_names
4712     Number: m n k i j
4713
4714   AOT_theorem "0-pred": <¬∃n [P]n 0> (791)
4715   proof (rule "raa-cor:2")
4716     AOT_assume <∃n [P]n 0>
4717     then AOT_obtain n where <[P]n 0>
4718       using "Number.∃E"[rotated] by meson
4719     AOT_hence <∃x [P]x 0>
4720       using "&E" "∃I" by fast
4721     AOT_thus <∃x [P]x 0 & ¬∃x [P]x 0>
4722       using "no-pred-0:1" "&I" by auto
4723   qed

```



```

4724
4725 AOT_theorem "no-same-succ": (792)
4726   <∀n∀m∀k ([P]nk & [P]mk → n = m)>
4727 proof(safe intro!: Number.GEN "→I")
4728   fix n m k
4729   AOT_assume <[P]nk & [P]mk>
4730   AOT_thus <n = m>
4731     by (safe intro!: "cqt:2[const_var]"[axiom_inst] "df-1-1:3"[
4732       unvarify R, OF "pred-thm:2",
4733       THEN "→E", OF "pred-1-1:4", THEN "qml:2"[axiom_inst, THEN "→E"],
4734       THEN "≡dfE"[OF "df-1-1:1"], THEN "&E"(2), THEN "∀E"(1), THEN "∀E"(1),
4735       THEN "∀E"(1)[where τ=<AOT_term_of_var (Number.Rep k)>], THEN "→E"])
4736 qed
4737
4738 AOT_theorem induction: (793)
4739   <∀F([F]0 & ∀n∀m([P]nm → ([F]n → [F]m)) → ∀n[F]n)>
4740 proof (safe intro!: GEN[where 'a=<<κ>>] Number.GEN "&I" "→I";
4741   frule "&E"(1); drule "&E"(2))
4742   fix F n
4743   AOT_assume FO: <[F]0>
4744   AOT_assume O: <∀n∀m([P]nm → ([F]n → [F]m))>
4745   {
4746     fix x y
4747     AOT_assume <[[P]+0x & [[P]+0y>
4748     AOT_hence <[N]x> and <[N]y>
4749     using "&E" "≡E"(2) "nnumber:3" by blast+
4750     moreover AOT_assume <[P]xy>
4751     moreover AOT_assume <[F]x>
4752     ultimately AOT_have <[F]y>
4753     using O[THEN "∀E"(2), THEN "→E", THEN "∀E"(2), THEN "→E",
4754       THEN "→E", THEN "→E"] by blast
4755   } note 1 = this
4756   AOT_have O: <[[P]+0n>
4757   by (metis "≡E"(1) "nnumber:3" Number.ψ)
4758   AOT_show <[F]n>
4759   apply (rule "pre-ind"[unconstrain  $\mathcal{R}$ , unvarify  $\beta$ , THEN "→E", OF "pred-thm:2",
4760     OF "pred-1-1:4", unvarify z, OF "zero:2", THEN "→E",
4761     THEN "∀E"(2), THEN "→E"]);
4762   safe intro!: O "&I" GEN "→I" FO)
4763   using 1 by blast
4764 qed
4765
4766 AOT_theorem "suc-num:1": <[P]nx → [N]x> (794.1)
4767 proof(rule "→I")
4768   AOT_have <[[P]+0 n>
4769   by (meson Number.ψ "≡E"(1) "nnumber:3")
4770   moreover AOT_assume <[P]nx>
4771   ultimately AOT_have <[[P]*0 x>
4772   using "w-ances-her:3"[unconstrain  $\mathcal{R}$ , unvarify  $\beta$ , OF "pred-thm:2", THEN "→E",
4773     OF "pred-1-1:4", unvarify x, OF "zero:2",
4774     THEN "→E", OF "&I"]
4775   by blast
4776   AOT_hence <[[P]+0 x>
4777   using "assume1:5"[unvarify x, OF "zero:2", THEN "≡E"(2), OF "∀I"(1)]
4778   by blast
4779   AOT_thus <[N]x>
4780   by (metis "≡E"(2) "nnumber:3")
4781 qed
4782
4783 AOT_theorem "suc-num:2": <[[P]*nx → [N]x> (794.2)
4784 proof(rule "→I")
4785   AOT_have <[[P]+0 n>
4786   using Number.ψ "≡E"(1) "nnumber:3" by blast

```



```

4787 AOT_assume <[[P]*]n x>
4788 AOT_hence <∀F (∀z ([P]nz → [F]z) & ∀x'∀y' ([P]x'y' → ([F]x' → [F]y')) → [F]x)>
4789   using "assume-anc:3"[THEN "≡E"(1)] by blast
4790 AOT_hence ϑ: <∀z ([P]nz → [N]z) & ∀x'∀y' ([P]x'y' → ([N]x' → [N]y')) → [N]x>
4791   using "∀E"(1) "nnumber:2" by blast
4792 AOT_show <[N]x>
4793 proof (safe intro!: ϑ[THEN "→E"] GEN "→I" "&I")
4794   AOT_show <[N]z> if <[P]nz> for z
4795     using Number.ψ "suc-num:1" that "→E" by blast
4796 next
4797   AOT_show <[N]y> if <[P]xy> and <[N]x> for x y
4798     using "suc-num:1"[unconstrain n, THEN "→E"] that "→E" by blast
4799 qed
4800 qed
4801
4802 AOT_theorem "suc-num:3": <[[P]+nx → [N]x> (794.3)
4803 proof (rule "→I")
4804   AOT_assume <[[P]+nx>
4805   AOT_hence <[[P]*nx ∨ n =P x>
4806     by (metis "assume1:5" "≡E"(1))
4807   moreover {
4808     AOT_assume <[[P]*nx>
4809     AOT_hence <[N]x>
4810       by (metis "suc-num:2" "→E")
4811   }
4812   moreover {
4813     AOT_assume <n =P x>
4814     AOT_hence <n = x>
4815       using "id-R-thm:3"[unconstrain  $\mathcal{R}$ , unvarify  $\beta$ , OF "pred-thm:2",
4816         THEN "→E", OF "pred-1-1:4", THEN "→E"] by blast
4817     AOT_hence <[N]x>
4818       by (metis "rule=E" Number.ψ)
4819   }
4820   ultimately AOT_show <[N]x>
4821     by (metis "∀E"(3) "reductio-aa:1")
4822 qed
4823
4824 AOT_theorem "pred-num": <[[P]xn → [N]x> (795)
4825 proof (rule "→I")
4826   AOT_assume 0: <[[P]xn>
4827   AOT_have <[[P]+0 n>
4828     using Number.ψ "≡E"(1) "nnumber:3" by blast
4829   AOT_hence <[[P]*]0 n ∨ 0 =P n>
4830     using "assume1:5"[unvarify x, OF "zero:2"] by (metis "≡E"(1))
4831   moreover {
4832     AOT_assume <0 =P n>
4833     AOT_hence <∃z ([P]0z & [P]nz)>
4834       using "assume1:2"[unvarify x, OF "zero:2", THEN "≡E"(1)] by blast
4835     then AOT_obtain a where <[P]0a & [P]na> using "∃E"[rotated] by blast
4836     AOT_hence <0 = n>
4837       using "pred-1-1:3"[THEN "df-1-1:1"[THEN "≡dfE"], THEN "&E"(2),
4838         THEN "∀E"(1), OF "zero:2", THEN "∀E"(2),
4839         THEN "∀E"(2), THEN "→E"] by blast
4840     AOT_hence <[[P]x 0>
4841       using 0 "rule=E" id_sym by fast
4842     AOT_hence <∃x [[P]x 0>
4843       by (rule "∃I")
4844     AOT_hence <∃x [[P]x 0 & ¬∃x [[P]x 0>
4845       by (metis "no-pred-0:1" "raa-cor:3")
4846   }
4847   ultimately AOT_have <[[P]*]0n>
4848     by (metis "∀E"(3) "raa-cor:1")
4849   AOT_hence <∃z ([P]+0z & [P]zn)>

```

```

4850     using "w-ances-her:7"[unconstrain  $\mathcal{R}$ , unvarify  $\beta$ , OF "pred-thm:2",
4851           THEN " $\rightarrow$ E", OF "pred-1-1:4", unvarify x,
4852           OF "zero:2", THEN " $\rightarrow$ E"] by blast
4853 then AOT_obtain b where b_prop: <[[ $\mathbb{P}$ ]+]0b & [[ $\mathbb{P}$ ]bn>
4854     using " $\exists$ E"[rotated] by blast
4855 AOT_hence <[ $\mathbb{N}$ ]b>
4856     by (metis "&E"(1) " $\equiv$ E"(2) "nnumber:3")
4857 moreover AOT_have <x = b>
4858     using "pred-1-1:3"[THEN "df-1-1:1"[THEN " $\equiv_{df}$ E"], THEN "&E"(2),
4859           THEN " $\forall$ E"(2), THEN " $\forall$ E"(2), THEN " $\forall$ E"(2), THEN " $\rightarrow$ E",
4860           OF "&I", OF 0, OF b_prop[THEN "&E"(2)]]].
4861 ultimately AOT_show <[ $\mathbb{N}$ ]x>
4862     using "rule=E" id_sym by fast
4863 qed
4864
4865 AOT_theorem "nat-card": <[ $\mathbb{N}$ ]x  $\rightarrow$  NaturalCardinal(x)> (796)
4866 proof(rule " $\rightarrow$ I")
4867   AOT_assume <[ $\mathbb{N}$ ]x>
4868   AOT_hence <[[ $\mathbb{P}$ ]+]0x>
4869     by (metis " $\equiv$ E"(1) "nnumber:3")
4870   AOT_hence <[[ $\mathbb{P}$ ]*]0x  $\vee$  0 = $\mathbb{P}$  x>
4871     using "assume1:5"[unvarify x, OF "zero:2", THEN " $\equiv$ E"(1)] by blast
4872   moreover {
4873     AOT_assume <[[ $\mathbb{P}$ ]*]0x>
4874     then AOT_obtain a where <[[ $\mathbb{P}$ ]ax>
4875       using "anc-her:5"[unvarify R x, OF "zero:2", OF "pred-thm:2", THEN " $\rightarrow$ E"]
4876       " $\exists$ E"[rotated] by blast
4877     AOT_hence < $\exists$ F $\exists$ u ([F]u & Numbers(x,F) & Numbers(a, [F]-u)>
4878       using "pred-thm:3"[THEN " $\equiv$ E"(1)] by blast
4879     then AOT_obtain F where < $\exists$ u ([F]u & Numbers(x,F) & Numbers(a, [F]-u)>
4880       using " $\exists$ E"[rotated] by blast
4881     then AOT_obtain u where <[F]u & Numbers(x,F) & Numbers(a, [F]-u)>
4882       using "Ordinary. $\exists$ E"[rotated] by meson
4883     AOT_hence <NaturalCardinal(x)>
4884       using "eq-num:6"[THEN " $\rightarrow$ E"] "&E" by blast
4885   }
4886   moreover {
4887     AOT_assume <0 = $\mathbb{P}$  x>
4888     AOT_hence <0 = x>
4889       using "id-R-thm:3"[unconstrain  $\mathcal{R}$ , unvarify  $\beta$ , OF "pred-thm:2",
4890           THEN " $\rightarrow$ E", OF "pred-1-1:4", unvarify x,
4891           OF "zero:2", THEN " $\rightarrow$ E"] by blast
4892     AOT_hence <NaturalCardinal(x)>
4893       by (metis "rule=E" "zero-card")
4894   }
4895   ultimately AOT_show <NaturalCardinal(x)>
4896     by (metis " $\forall$ E"(2) "raa-cor:1")
4897 qed
4898
4899 AOT_theorem "pred-func:1": <[[ $\mathbb{P}$ ]xy & [[ $\mathbb{P}$ ]xz  $\rightarrow$  y = z]> (797.1)
4900 proof (rule " $\rightarrow$ I"; frule "&E"(1); drule "&E"(2))
4901   AOT_assume <[[ $\mathbb{P}$ ]xy>
4902   AOT_hence < $\exists$ F $\exists$ u ([F]u & Numbers(y,F) & Numbers(x, [F]-u)>
4903     using "pred-thm:3"[THEN " $\equiv$ E"(1)] by blast
4904   then AOT_obtain F where < $\exists$ u ([F]u & Numbers(y,F) & Numbers(x, [F]-u)>
4905     using " $\exists$ E"[rotated] by blast
4906   then AOT_obtain a where
4907     Oa: <0!a>
4908     and a_prop: <[F]a & Numbers(y,F) & Numbers(x, [F]-a)>
4909     using " $\exists$ E"[rotated] "&E" by blast
4910   AOT_assume <[[ $\mathbb{P}$ ]xz>
4911   AOT_hence < $\exists$ F $\exists$ u ([F]u & Numbers(z,F) & Numbers(x, [F]-u)>
4912     using "pred-thm:3"[THEN " $\equiv$ E"(1)] by blast

```

```

4913 then AOT_obtain G where <∃u ([G]u & Numbers(z,G) & Numbers(x,[G]-u)>
4914   using "∃E"[rotated] by blast
4915 then AOT_obtain b where Ob: <O!b>
4916   and b_prop: <[G]b & Numbers(z,G) & Numbers(x,[G]-b)>
4917   using "∃E"[rotated] "&E" by blast
4918 AOT_have <[F]-a ≈E [G]-b>
4919   using "num-tran:2"[unvarify G H, OF "F-u[den]", OF "F-u[den]",
4920     THEN "→E", OF "&I", OF a_prop[THEN "&E"(2)],
4921     OF b_prop[THEN "&E"(2)]]].
4922 AOT_hence <F ≈E G>
4923   using "P'-eq"[unconstrain u, THEN "→E", OF Oa, unconstrain v, THEN "→E",
4924     OF Ob, THEN "→E", OF "&I", OF "&I"]
4925     a_prop[THEN "&E"(1), THEN "&E"(1)]
4926     b_prop[THEN "&E"(1), THEN "&E"(1)] by blast
4927 AOT_thus <y = z>
4928   using "pre-Hume"[THEN "→E", THEN "≡E"(2), OF "&I",
4929     OF a_prop[THEN "&E"(1), THEN "&E"(2)],
4930     OF b_prop[THEN "&E"(1), THEN "&E"(2)]]
4931 by blast
4932 qed
4933
4934 AOT_theorem "pred-func:2": <[P]nm & [P]nk → m = k> (797.2)
4935   using "pred-func:1".
4936
4937 AOT_theorem being_number_of_den: <[λx x = #G]↓>
4938 proof (rule "safe-ext"[axiom_inst, THEN "→E"]; safe intro!: "&I" GEN RN)
4939   AOT_show <[λx Numbers(x,[λz A[G]z])↓>
4940     by (rule numbers_prop_den[unvarify G]) "cqt:2[lambda]"
4941 next
4942   AOT_modally_strict {
4943     AOT_show <Numbers(x,[λz A[G]z]) ≡ x = #G> for x
4944     using "eq-num:2".
4945   }
4946 qed
4947
4948 axiomatization ω_nat :: <ω ⇒ nat> where ω_nat: <surj ω_nat>
4949 text<Unfortunately, since the axiom requires the type @{typ ω}
4950   to have an infinite domain, @{command nitpick} can only find a potential model
4951   and no genuine model.
4952   However, since we could trivially choose @{typ ω} as a copy of @{typ nat},
4953   we can still be assured that above axiom is consistent.>
4954 lemma <True> nitpick[satisfy, user_axioms, card nat=1, expect = potential] ..
4955
4956 AOT_axiom "modal-axiom": (798)
4957   <∃x([N]x & x = #G) → ◇∃y([E!]y & ∀u (A[G]u → u ≠E y))>
4958 proof(rule AOT_model_axiomI) AOT_modally_strict {
4959   text<The actual extension on the ordinary objects of a property is the
4960     set of ordinary urelements that exemplifies the property in the
4961     designated actual world.>
4962   define act_ωext :: <<κ> ⇒ ω set> where
4963     <act_ωext ≡ λ Π . {x :: ω . [w0 ⊨ [Π]«ωκ x»]}>
4964   text<Encoding a property with infinite actual extension on the ordinary objects
4965     denotes a property by extended relation comprehension.>
4966   AOT_have enc_finite_act_ωext_den:
4967     <⊢□ [λx ∃F(¬«εo w. finite (act_ωext F)» & x[F])↓>
4968 proof(safe intro!: Comprehension_1[THEN "→E"] RN GEN "→I")
4969   AOT_modally_strict {
4970     fix F G
4971     AOT_assume <□G ≡E F>
4972     AOT_hence <AG ≡E F>
4973     using "nec-imp-act"[THEN "→E"] by blast
4974     AOT_hence <A(G↓ & F↓ & ∀u([G]u ≡ [F]u))>
4975     by (AOT_subst_def (reverse) eqE)

```

```

4976   hence <[w0 ⊨ [G]«ωκ x»] = [w0 ⊨ [F]«ωκ x»]> for x
4977     by (auto dest!: "∀E"(1) "→E"
4978         simp: AOT_model_denotes_κ_def AOT_sem_denotes AOT_sem_conj
4979              AOT_model_ωκ_ordinary AOT_sem_act AOT_sem_equiv)
4980   AOT_thus <¬«ε0 w. finite (act_wext (AOT_term_of_var F))» ≡
4981           ¬«ε0 w. finite (act_wext (AOT_term_of_var G))»>
4982     by (simp add: AOT_sem_not AOT_sem_equiv act_wext_def
4983              AOT_model_proposition_choice_simp)
4984   }
4985 qed
4986 text<By coexistence, encoding only properties with finite actual extension
4987     on the ordinary objects denotes.>
4988 AOT_have <[λx ∀F(x[F] → «ε0 w. finite (act_wext F))]]↓>
4989 proof(rule "safe-ext"[axiom_inst, THEN "→E"]; safe intro!: "&I" RN GEN)
4990   AOT_show <[λx ¬[λx ∃F(¬«ε0 w. finite (act_wext F)) & x[F]])x]↓>
4991     by "cqt:2"
4992 next
4993   AOT_modally_strict {
4994     fix x
4995     AOT_show <¬[λx ∃F (¬«ε0 w. finite (act_wext F)) & x[F]])x ≡
4996             ∀F(x[F] → «ε0 w. finite (act_wext F))»>
4997       by (AOT_subst <[λx ∃F (¬«ε0 w. finite (act_wext F)) & x[F]])x>
4998           <∃F (¬«ε0 w. finite (act_wext F)) & x[F]]>;
4999         (rule "beta-C-meta"[THEN "→E"])?
5000     (auto simp: enc_finite_act_wext_den AOT_sem_equiv AOT_sem_not
5001              AOT_sem_forall AOT_sem_imp AOT_sem_conj AOT_sem_exists)
5002   }
5003 qed
5004 text<We show by induction that any property encoded by a natural number
5005     has a finite actual extension on the ordinary objects.>
5006 AOT_hence <[λx ∀F(x[F] → «ε0 w. finite (act_wext F))]]n> for n
5007 proof(rule induction[THEN "∀E"(1), THEN "→E", THEN "Number.∀E"];
5008        safe intro!: "&I" "Number.GEN" "β←C" "zero:2" "→I" "cqt:2"
5009        dest!: "β→C")
5010   AOT_show <∀F(0[F] → «ε0 w. finite (act_wext F))»>
5011 proof(safe intro!: GEN "→I")
5012   fix F
5013   AOT_assume <0[F]>
5014   AOT_actually {
5015     AOT_hence <¬∃u [F]u>
5016       using "zero:2" "intro-elim:3:a" AOT_sem_enc_nec by blast
5017     AOT_hence <∀x ¬(0!x & [F]x)>
5018       using "cqt-further:4" "vdash-properties:10" by blast
5019     hence <¬([w0 ⊨ [F]«ωκ x»])> for x
5020       by (auto dest!: "∀E"(1)[where τ=ωκ x]
5021           simp: AOT_sem_not AOT_sem_conj AOT_model_ωκ_ordinary
5022              "russell-axiom[exe,1].ψ_denotes_asm")
5023   }
5024   AOT_thus <«ε0 w. finite (act_wext (AOT_term_of_var F))»>
5025     by (auto simp: AOT_model_proposition_choice_simp act_wext_def)
5026 qed
5027 next
5028   fix n m
5029   AOT_assume <[P]nm>
5030   AOT_hence <∃F∃u ([F]u & Numbers(m,F) & Numbers(n, [F]-u)>
5031     using "pred-thm:3"[THEN "≡E"(1)] by blast
5032   then AOT_obtain G where <∃u ([G]u & Numbers(m,G) & Numbers(n, [G]-u)>
5033     using "∃E"[rotated] by blast
5034   then AOT_obtain u where 0: <[G]u & Numbers(m,G) & Numbers(n, [G]-u)>
5035     using "Ordinary.∃E"[rotated] by meson
5036
5037   AOT_assume n_prop: <∀F(n[F] → «ε0 w. finite (act_wext F))»>
5038   AOT_show <∀F(m[F] → «ε0 w. finite (act_wext F))»>

```

```

5039 proof(safe intro!: GEN "→I")
5040   fix F
5041   AOT_assume <m[F]>
5042   AOT_hence 1: <[λx  $\mathcal{A}[F]x$ ]  $\approx_E$  G>
5043     using 0[THEN "&E"(1), THEN "&E"(2), THEN numbers[THEN "≡dfE"],
5044       THEN "&E"(2), THEN "∀E"(2), THEN "≡E"(1)] by auto
5045   AOT_show <<εo w. finite (act_wext (AOT_term_of_var F))>>
5046   proof(rule "raa-cor:1")
5047     AOT_assume <¬<εo w. finite (act_wext (AOT_term_of_var F))>>
5048     hence inf: <infinite (act_wext (AOT_term_of_var F))>
5049     by (auto simp: AOT_sem_not AOT_model_proposition_choice_simp)
5050   then AOT_obtain v where act_F_v: < $\mathcal{A}[F]v$ >
5051     unfolding AOT_sem_act act_wext_def
5052     by (metis AOT_term_of_var_cases AOT_model_ωκ_ordinary
5053       AOT_model_denotes_κ_def Ordinary.Rep_cases κ.disc(7)
5054       mem_Collect_eq not_finite_existsD)
5055   AOT_hence <[λx  $\mathcal{A}[F]x$ ] v>
5056     by (safe intro!: "β←C" "cqt:2")
5057   AOT_hence <[λx  $\mathcal{A}[F]x$ ] -v  $\approx_E$  [G] -u>
5058     by (safe intro!: eqP'[unvarify F, THEN "→E"] "&I" "cqt:2" 1
5059       0[THEN "&E"(1), THEN "&E"(1)])
5060   moreover AOT_have <[λx  $\mathcal{A}[F]x$ ] -v  $\approx_E$  [λx  $\mathcal{A}[\lambda y [F]y \ \& \ y \neq_E \ v]x$ ]>
5061   proof(safe intro!: "apE-eqE:1"[unvarify F G, THEN "→E"] "cqt:2"
5062     "F-u[den]"[unvarify F] eqE[THEN "≡dfI"] "&I"
5063     Ordinary.GEN)
5064     fix u
5065     AOT_have <[λx [λx  $\mathcal{A}[F]x$ ]x & x ≠E v]u ≡ [λx  $\mathcal{A}[F]x$ ]u & u ≠E v>
5066       by (safe intro!: "beta-C-meta"[THEN "→E"] "cqt:2")
5067     also AOT_have <[λx  $\mathcal{A}[F]x$ ]u & u ≠E v ≡  $\mathcal{A}[F]u$  & u ≠E v>
5068       by (AOT_subst <[λx  $\mathcal{A}[F]x$ ]u> < $\mathcal{A}[F]u$ >)
5069       (safe intro!: "beta-C-meta"[THEN "→E"] "cqt:2"
5070         "oth-class-taut:3:a")
5071     also AOT_have < $\mathcal{A}[F]u$  & u ≠E v ≡  $\mathcal{A}([F]u \ \& \ u \neq_E \ v)$ >
5072       using "id-act2:2" AOT_sem_conj AOT_sem_equiv AOT_sem_act by auto
5073     also AOT_have < $\mathcal{A}([F]u \ \& \ u \neq_E \ v) \equiv \mathcal{A}[\lambda y [F]y \ \& \ y \neq_E \ v]u$ >
5074       by (AOT_subst <[λy [F]y & y ≠E v]u> <[F]u & u ≠E v>)
5075       (safe intro!: "beta-C-meta"[THEN "→E"] "cqt:2"
5076         "oth-class-taut:3:a")
5077     also AOT_have < $\mathcal{A}[\lambda y [F]y \ \& \ y \neq_E \ v]u \equiv [λx \mathcal{A}[\lambda y [F]y \ \& \ y \neq_E \ v]x]u$ >
5078       by (safe intro!: "beta-C-meta"[THEN "→E", symmetric] "cqt:2")
5079     finally AOT_show <[[λx  $\mathcal{A}[F]x$ ] -v]u ≡ [λx  $\mathcal{A}[\lambda y [F]y \ \& \ y \neq_E \ v]x$ ]u>
5080       by (auto intro!: "cqt:2"
5081         intro: "rule-id-df:2:b"[OF "F-u", where τ1τn=<(_,_)>, simplified])
5082   qed
5083   ultimately AOT_have <[λx  $\mathcal{A}[\lambda y [F]y \ \& \ y \neq_E \ v]x$ ]  $\approx_E$  [G] -u>
5084     using "eq-part:2[terms]" "eq-part:3[terms]" "→E" by blast
5085   AOT_hence <n[λy [F]y & y ≠E v]>
5086     by (safe intro!: 0[THEN "&E"(2), THEN numbers[THEN "≡dfE"],
5087       THEN "&E"(2), THEN "∀E"(1), THEN "≡E"(2)] "cqt:2")
5088   hence finite: <finite (act_wext <[λy [F]y & y ≠E v]>>>
5089     by (safe intro!: n_prop[THEN "∀E"(1), THEN "→E",
5090       simplified AOT_model_proposition_choice_simp]
5091       "cqt:2")
5092   obtain y where y_def: <ωκ y = AOT_term_of_var (Ordinary.Rep v)>
5093     by (metis AOT_model_ordinary_ωκ Ordinary.restricted_var_condition)
5094   AOT_actually {
5095     fix x
5096     AOT_assume <[λy [F]y & y ≠E v]ωκ x>>
5097     AOT_hence <[F]ωκ x>>
5098       by (auto dest!: "β→C" "&E"(1))
5099   }
5100   moreover AOT_actually {
5101     AOT_have <[F]ωκ y>>

```

```

5102     unfolding y_def using act_F_v AOT_sem_act by blast
5103   }
5104   moreover AOT_actually {
5105     fix x
5106     assume noteq: <x ≠ y>
5107     AOT_assume <[F] «ωκ x»>
5108     moreover AOT_have ωκ_x_den: <«ωκ x»↓>
5109     using AOT_sem_exe calculation by blast
5110     moreover {
5111       AOT_have <¬(«ωκ x» =E v)>
5112       proof(rule "raa-cor:2")
5113         AOT_assume <«ωκ x» =E v>
5114         AOT_hence <«ωκ x» = v>
5115           using "=E-simple:2"[unvarify x, THEN "→E", OF ωκ_x_den]
5116           by blast
5117         hence <ωκ x = ωκ y>
5118           unfolding y_def AOT_sem_eq
5119           by meson
5120         hence <x = y>
5121           by blast
5122         AOT_thus <p & ¬p> for p using noteq by blast
5123       qed
5124       AOT_hence <«ωκ x» ≠E v>
5125         by (safe intro!: "thm-neg=E"[unvarify x, THEN "≡E"(2)] ωκ_x_den)
5126     }
5127     ultimately AOT_have <[λy [F]y & y ≠E v]«ωκ x»>
5128       by (auto intro!: "β←C" "cqt:2" "&I")
5129   }
5130   ultimately have <(insert y (act_wext «[λy [F]y & y ≠E v]»)) =
5131     (act_wext (AOT_term_of_var F))>
5132     unfolding act_wext_def
5133     by auto
5134   hence <finite (act_wext (AOT_term_of_var F))>
5135     using finite finite.insertI by metis
5136   AOT_thus <p & ¬p> for p
5137     using inf by blast
5138   qed
5139   qed
5140   qed
5141   AOT_hence nat_enc_finite: <∀F (n[F] → «εo w. finite (act_wext F)»)> for n
5142     using "β→C"(1) by blast
5143
5144   text<The main proof can now generate a witness, since we required
5145     the domain of ordinary objects to be infinite.>
5146   AOT_show <∃x ([N]x & x = #G) → ◇∃y (E!y & ∀u (A[G]u → u ≠E y))>
5147   proof(safe intro!: "→I")
5148     AOT_assume <∃x ([N]x & x = #G)>
5149     then AOT_obtain n where <n = #G>
5150       using "Number.∃E"[rotated] by meson
5151     AOT_hence <Numbers(n, [λx A[G]x])>
5152       using "eq-num:3" "rule=E" id_sym by fast
5153     AOT_hence <n[G]>
5154       by (auto intro!: numbers[THEN "≡dfE", THEN "&E"(2),
5155         THEN "∀E"(2), THEN "≡E"(2)]
5156         "eq-part:1"[unvarify F] "cqt:2")
5157     AOT_hence <«εo w. finite (act_wext (AOT_term_of_var G))>
5158       using nat_enc_finite[THEN "∀E"(2), THEN "→E"] by blast
5159     hence finite: <finite (act_wext (AOT_term_of_var G))>
5160       by (auto simp: AOT_model_proposition_choice_simp)
5161     AOT_have <∃u ¬A[G]u>
5162     proof(rule "raa-cor:1")
5163       AOT_assume <¬∃u ¬A[G]u>
5164       AOT_hence <∀x ¬(0!x & ¬A[G]x)>

```

```

5165     by (metis "cqt-further:4" "→E")
5166 AOT_hence <A[G]x> if <O!x> for x
5167     using "∀E"(2) AOT_sem_conj AOT_sem_not that by blast
5168 hence <[w0 ⊨ [G]«ωκ x»]> for x
5169     by (metis AOT_term_of_var_cases AOT_model_ωκ_ordinary
5170         AOT_model_denotes_κ_def AOT_sem_act κ.disc(7))
5171 hence <(act_ωext (AOT_term_of_var G)) = UNIV>
5172     unfolding act_ωext_def by auto
5173 moreover have <infinite (UNIV::ω set)>
5174     by (metis ω_nat finite_imageI infinite_UNIV_char_0)
5175 ultimately have <infinite (act_ωext (AOT_term_of_var G))>
5176     by simp
5177 AOT_thus <p & ¬p> for p using finite by blast
5178 qed
5179 then AOT_obtain x where x_prop: <O!x & ¬A[G]x>
5180     using "∃E"[rotated] by blast
5181 AOT_hence <◇E!x>
5182     by (metis "betaC:1:a" "con-dis-i-e:2:a" AOT_sem_ordinary)
5183 moreover AOT_have <□∀u (A[G]u → u ≠E x)>
5184 proof(safe intro!: RN GEN "→I")
5185   AOT_modally_strict {
5186     fix y
5187     AOT_assume <O!y>
5188     AOT_assume 0: <A[G]y>
5189     AOT_show <y ≠E x>
5190     proof (safe intro!: "thm-neg=E"[THEN "≡E"(2)] "raa-cor:2")
5191       AOT_assume <y =E x>
5192       AOT_hence <y = x>
5193         by (metis "=E-simple:2" "vdash-properties:10")
5194       hence <y = x>
5195         by (simp add: AOT_sem_eq AOT_term_of_var_inject)
5196       AOT_hence <¬A[G]y>
5197         using x_prop "&E" AOT_sem_not AOT_sem_act by metis
5198       AOT_thus <A[G]y & ¬A[G]y>
5199         using 0 "&I" by blast
5200     qed
5201   }
5202 qed
5203 ultimately AOT_have <◇(∀u (A[G]u → u ≠E x) & E!x)>
5204     using "KBasic:16"[THEN "→E", OF "&I"] by blast
5205 AOT_hence <◇(E!x & ∀u (A[G]u → u ≠E x))>
5206     by (AOT_subst <E!x & ∀u (A[G]u → u ≠E x)> <∀u (A[G]u → u ≠E x) & E!x>)
5207     (auto simp: "oth-class-taut:2:a")
5208 AOT_hence <∃y ◇(E!y & ∀u (A[G]u → u ≠E y))>
5209     using "∃I" by fast
5210 AOT_thus <◇∃y (E!y & ∀u (A[G]u → u ≠E y))>
5211     using "CBF◇"[THEN "→E"] by fast
5212 qed
5213 } qed
5214
5215 AOT_theorem "modal-lemma":
5216   <◇∀u(A[G]u → u ≠E v) → ∀u(A[G]u → u ≠E v)>
5217 proof(safe intro!: "→I" Ordinary.GEN)
5218   AOT_modally_strict {
5219     fix u
5220     AOT_assume act_Gu: <A[G]u>
5221     AOT_have <∀u (A[G]u → u ≠E v) → u ≠E v>
5222     proof(rule "→I")
5223       AOT_assume <∀u (A[G]u → u ≠E v)>
5224       AOT_hence <A[G]u → u ≠E v>
5225         using "Ordinary.∀E" by fast
5226       AOT_thus <u ≠E v>
5227         using act_Gu "→E" by blast

```

(800)


```

5228   qed
5229 } note 0 = this
5230 AOT_have  $\vartheta$ :  $\langle \Box(\forall u (\mathcal{A}[G]u \rightarrow u \neq_E v) \rightarrow u \neq_E v) \rangle$  if  $\langle \Box \mathcal{A}[G]u \rangle$  for u
5231 proof -
5232   AOT_have  $\langle \Box \mathcal{A}[G]u \rightarrow \Box(\forall u (\mathcal{A}[G]u \rightarrow u \neq_E v) \rightarrow u \neq_E v) \rangle$ 
5233     apply (rule RM) using 0 "&E" " $\rightarrow$ I" by blast
5234     thus ?thesis using that " $\rightarrow$ E" by blast
5235   qed
5236   fix u
5237   AOT_assume 1:  $\langle \Diamond \forall u (\mathcal{A}[G]u \rightarrow u \neq_E v) \rangle$ 
5238   AOT_assume  $\langle \mathcal{A}[G]u \rangle$ 
5239   AOT_hence  $\langle \Box \mathcal{A}[G]u \rangle$ 
5240     by (metis "Act-Basic:6" " $\equiv$ E"(1))
5241   AOT_hence  $\langle \Box(\forall u (\mathcal{A}[G]u \rightarrow u \neq_E v) \rightarrow u \neq_E v) \rangle$ 
5242     using Ordinary. $\psi$   $\vartheta$  by blast
5243   AOT_hence  $\langle \Diamond u \neq_E v \rangle$ 
5244     using 1 "K $\Diamond$ "[THEN " $\rightarrow$ E", THEN " $\rightarrow$ E"] by blast
5245   AOT_thus  $\langle u \neq_E v \rangle$ 
5246     by (metis "id-nec4:2" " $\equiv$ E"(1))
5247   qed
5248
5249 AOT_theorem "th-succ":  $\langle \forall n \exists ! m [P]nm \rangle$  (801)
5250 proof(safe intro!: Number.GEN " $\rightarrow$ I" "uniqueness:1"[THEN " $\equiv_{df}$ I"])
5251   fix n
5252   AOT_have  $\langle \text{NaturalCardinal}(n) \rangle$ 
5253     by (metis "nat-card" Number. $\psi$  " $\rightarrow$ E")
5254   AOT_hence  $\langle \exists G(n = \#G) \rangle$ 
5255     by (metis " $\equiv_{df}$ E" card)
5256   then AOT_obtain G where n_num_G:  $\langle n = \#G \rangle$ 
5257     using " $\exists$ E"[rotated] by blast
5258   AOT_hence  $\langle \exists n (n = \#G) \rangle$ 
5259     by (rule "Number. $\exists$ I")
5260   AOT_hence  $\langle \Diamond \exists y ([E!]y \ \& \ \forall u (\mathcal{A}[G]u \rightarrow u \neq_E y)) \rangle$ 
5261     using "modal-axiom"[axiom_inst, THEN " $\rightarrow$ E"] by blast
5262   AOT_hence  $\langle \exists y \Diamond ([E!]y \ \& \ \forall u (\mathcal{A}[G]u \rightarrow u \neq_E y)) \rangle$ 
5263     using "BF $\Diamond$ "[THEN " $\rightarrow$ E"] by auto
5264   then AOT_obtain y where  $\langle \Diamond ([E!]y \ \& \ \forall u (\mathcal{A}[G]u \rightarrow u \neq_E y)) \rangle$ 
5265     using " $\exists$ E"[rotated] by blast
5266   AOT_hence  $\langle \Diamond E!y \rangle$  and 2:  $\langle \Diamond \forall u (\mathcal{A}[G]u \rightarrow u \neq_E y) \rangle$ 
5267     using "KBasic2:3" "&E" " $\rightarrow$ E" by blast+
5268   AOT_hence 0y:  $\langle 0!y \rangle$ 
5269     by (auto intro!: " $\beta \leftarrow C$ "(1) "cqt:2" intro: AOT_ordinary[THEN " $\equiv_{df}$ I"(2)])
5270   AOT_have 0:  $\langle \forall u (\mathcal{A}[G]u \rightarrow u \neq_E y) \rangle$ 
5271     using 2 "modal-lemma"[unconstrain v, THEN " $\rightarrow$ E", OF 0y, THEN " $\rightarrow$ E"] by simp
5272   AOT_have 1:  $\langle [\lambda x \mathcal{A}[G]x \vee x =_E y] \downarrow \rangle$ 
5273     by "cqt:2"
5274   AOT_obtain b where b_prop:  $\langle \text{Numbers}(b, [\lambda x \mathcal{A}[G]x \vee x =_E y]) \rangle$ 
5275     using "num:1"[unvarify G, OF 1] " $\exists$ E"[rotated] by blast
5276   AOT_have Pnb:  $\langle [P]nb \rangle$ 
5277   proof(safe intro!: "pred-thm:3"[THEN " $\equiv$ E"(2)]
5278     " $\exists$ I"(1)[where  $\tau = \ll [\lambda x \mathcal{A}[G]x \vee x =_E y] \gg$ ]
5279     1 " $\exists$ I"(2)[where  $\beta = y$ ] "&I" 0y b_prop)
5280     AOT_show  $\langle [\lambda x \mathcal{A}[G]x \vee x =_E y] y \rangle$ 
5281       by (auto intro!: " $\beta \leftarrow C$ "(1) "cqt:2" " $\vee$ I"(2)
5282         "ord=Eequiv:1"[THEN " $\rightarrow$ E", OF 0y])
5283   next
5284     AOT_have equinum:  $\langle [\lambda x \mathcal{A}[G]x \vee x =_E y]^{-y} \approx_E [\lambda x \mathcal{A}[G]x] \rangle$ 
5285     proof(rule "apE-eqE:1"[unvarify F G, THEN " $\rightarrow$ E"];
5286       ("cqt:2[lambda]" | rule "F-u[den]"[unvarify F]; "cqt:2[lambda]"?)
5287       AOT_show  $\langle [\lambda x \mathcal{A}[G]x \vee x =_E y]^{-y} \equiv_E [\lambda x \mathcal{A}[G]x] \rangle$ 
5288       proof (safe intro!: eqE[THEN " $\equiv_{df}$ I"] "&I" "F-u[den]"[unvarify F]
5289         Ordinary.GEN " $\rightarrow$ I"; "cqt:2"?)
5290     fix u

```



```

5291 AOT_have <[[ $\lambda x \mathcal{A}[G]x \vee [(=_{\mathbb{E}})]xy]^{-y}]u \equiv ([\lambda x \mathcal{A}[G]x \vee x =_{\mathbb{E}} y]u) \& u \neq_{\mathbb{E}} y$ >
5292   apply (rule "F-u"[THEN "=defI"(1)[where  $\tau_1\tau_n=<(\_,\_)>$ ], simplified]; "cqt:2"? )
5293   by (rule "beta-C-cor:2"[THEN " $\rightarrow E$ ", THEN " $\forall E$ "(2)]; "cqt:2")
5294 also AOT_have <...  $\equiv (\mathcal{A}[G]u \vee u =_{\mathbb{E}} y) \& u \neq_{\mathbb{E}} y$ >
5295   apply (AOT_subst <[[ $\lambda x \mathcal{A}[G]x \vee [(=_{\mathbb{E}})]xy]u$ ] < $\mathcal{A}[G]u \vee u =_{\mathbb{E}} y$ >)
5296   apply (rule "beta-C-cor:2"[THEN " $\rightarrow E$ ", THEN " $\forall E$ "(2)]; "cqt:2")
5297   using "oth-class-taut:3:a" by blast
5298 also AOT_have <...  $\equiv \mathcal{A}[G]u$ >
5299 proof(safe intro!: " $\equiv I$ " " $\rightarrow I$ ")
5300   AOT_assume <( $\mathcal{A}[G]u \vee u =_{\mathbb{E}} y$ ) & u  $\neq_{\mathbb{E}} y$ >
5301   AOT_thus < $\mathcal{A}[G]u$ >
5302     by (metis "&E"(1) "&E"(2) " $\forall E$ "(3) " $\equiv E$ "(1) "thm-neg=E")
5303 next
5304   AOT_assume < $\mathcal{A}[G]u$ >
5305   AOT_hence <u  $\neq_{\mathbb{E}} y$ > and < $\mathcal{A}[G]u \vee u =_{\mathbb{E}} y$ >
5306     using 0[THEN " $\forall E$ "(2), THEN " $\rightarrow E$ ", OF Ordinary. $\psi$ , THEN " $\rightarrow E$ "]
5307     " $\forall I$ " by blast+
5308   AOT_thus <( $\mathcal{A}[G]u \vee u =_{\mathbb{E}} y$ ) & u  $\neq_{\mathbb{E}} y$ >
5309     using "&I" by simp
5310 qed
5311 also AOT_have <...  $\equiv [\lambda x \mathcal{A}[G]x]u$ >
5312   by (rule "beta-C-cor:2"[THEN " $\rightarrow E$ ", THEN " $\forall E$ "(2), symmetric]; "cqt:2")
5313 finally AOT_show <[[ $\lambda x \mathcal{A}[G]x \vee [(=_{\mathbb{E}})]xy]^{-y}]u \equiv [\lambda x \mathcal{A}[G]x]u$ >.
5314 qed
5315 qed
5316 AOT_have 2: <[[ $\lambda x \mathcal{A}[G]x$ ] $\downarrow$ ] by "cqt:2[lambda]"
5317 AOT_show <Numbers(n, [[ $\lambda x \mathcal{A}[G]x \vee x =_{\mathbb{E}} y$ ] $^{-y}$ ])>
5318   using "num-tran:1"[unvarify G H, OF 2, OF "F-u[den]"[unvarify F, OF 1],
5319     THEN " $\rightarrow E$ ", OF equinum, THEN " $\equiv E$ "(2),
5320     OF "eq-num:2"[THEN " $\equiv E$ "(2), OF n_num_G]].
5321 qed
5322 AOT_show < $\exists \alpha ([\mathbb{N}] \alpha \& [\mathbb{P}] n \alpha \& \forall \beta ([\mathbb{N}] \beta \& [\mathbb{P}] n \beta \rightarrow \beta = \alpha)$ >
5323 proof(safe intro!: " $\exists I$ "(2)[where  $\beta=b$ ] "&I" Pnb " $\rightarrow I$ " GEN)
5324   AOT_show <[[ $\mathbb{N}] b$ ] using "suc-num:1"[THEN " $\rightarrow E$ ", OF Pnb].
5325 next
5326   fix y
5327   AOT_assume 0: <[[ $\mathbb{N}] y \& [\mathbb{P}] n y$ ]>
5328   AOT_show <y = b>
5329     apply (rule "pred-func:1"[THEN " $\rightarrow E$ "])
5330     using 0[THEN "&E"(2)] Pnb "&I" by blast
5331 qed
5332 qed
5333
5334 (* Note the use of a bold '.* *)
5335 AOT_define Successor :: < $\tau \Rightarrow \kappa_{\mathbb{E}}$ > <('_)'> [100] 100)
5336   "def-suc": <n' =def  $\iota m ([\mathbb{P}] nm)$ > (804)
5337
5338 text<Note: not explicitly in PLM>
5339 AOT_theorem "def-suc[den1]": < $\iota m ([\mathbb{P}] nm)$  $\downarrow$ > (804)
5340   using "A-Exists:2" "RA[2]" " $\equiv E$ "(2) "th-succ"[THEN "Number. $\forall E$ "] by blast
5341 text<Note: not explicitly in PLM>
5342 AOT_theorem "def-suc[den2]": shows <n' $\downarrow$ > (804)
5343   by (rule "def-suc"[THEN "=defI"(1)])
5344   (auto simp: "def-suc[den1]")
5345
5346 (* TODO: not in PLM *)
5347 AOT_theorem suc_eq_desc: <n' =  $\iota m ([\mathbb{P}] nm)$ >
5348   by (rule "def-suc"[THEN "=defI"(1)])
5349   (auto simp: "def-suc[den1]" "rule=I:1")
5350
5351 AOT_theorem "suc-fact": <n = m  $\rightarrow$  n' = m'> (805)
5352 proof (rule " $\rightarrow I$ ")
5353   AOT_assume 0: <n = m>

```

```

5354   AOT_show <n' = m'>
5355     apply (rule "rule=E"[rotated, OF 0])
5356     by (rule "=I"(1)[OF "def-suc[den2]"])
5357 qed
5358
5359 AOT_theorem "ind-gnd": <m = 0  $\vee$   $\exists n(m = n')$ > (806)
5360 proof -
5361   AOT_have <[[ $\mathbb{P}$ ]+]0m>
5362     using Number. $\psi$  " $\equiv$ E"(1) "nnumber:3" by blast
5363   AOT_hence <[[ $\mathbb{P}$ ]+]0m  $\vee$  0 = $\mathbb{P}$  m>
5364     using "assume1:5"[unvarify x, OF "zero:2", THEN " $\equiv$ E"(1)] by blast
5365   moreover {
5366     AOT_assume <[[ $\mathbb{P}$ ]+]0m>
5367     AOT_hence < $\exists z$  ([[ $\mathbb{P}$ ]+]0z & [ $\mathbb{P}$ ]z)m>
5368       using "w-ances-her:7"[unconstrain  $\mathcal{R}$ , unvarify  $\beta$  x, OF "zero:2",
5369         OF "pred-thm:2", THEN " $\rightarrow$ E", OF "pred-1-1:4",
5370         THEN " $\rightarrow$ E"]
5371     by blast
5372     then AOT_obtain z where  $\vartheta$ : <[[ $\mathbb{P}$ ]+]0z> and  $\xi$ : <[[ $\mathbb{P}$ ]z]m>
5373       using "&E" " $\exists$ E"[rotated] by blast
5374     AOT_have Nz: <[ $\mathbb{N}$ ]z>
5375       using  $\vartheta$  " $\equiv$ E"(2) "nnumber:3" by blast
5376     moreover AOT_have <m = z'>
5377     proof (rule "def-suc"[THEN "=dfI"(1)];
5378       safe intro!: "def-suc[den1]"[unconstrain n, THEN " $\rightarrow$ E", OF Nz]
5379         "nec-hintikka-scheme"[THEN " $\equiv$ E"(2)] "&I"
5380         GEN " $\rightarrow$ I" "Act-Basic:2"[THEN " $\equiv$ E"(2)])
5381       AOT_show < $\mathcal{A}$ [ $\mathbb{N}$ ]m> using Number. $\psi$ 
5382         by (meson "mod-col-num:1" "nec-imp-act" " $\rightarrow$ E")
5383     next
5384       AOT_show < $\mathcal{A}$ [ $\mathbb{P}$ ]z)m> using  $\xi$ 
5385         by (meson "nec-imp-act" "pred-1-1:1" " $\rightarrow$ E")
5386     next
5387     fix y
5388     AOT_assume < $\mathcal{A}$ ([ $\mathbb{N}$ ]y & [ $\mathbb{P}$ ]zy)>
5389     AOT_hence < $\mathcal{A}$ [ $\mathbb{N}$ ]y> and < $\mathcal{A}$ [ $\mathbb{P}$ ]zy>
5390       using "Act-Basic:2" "&E" " $\equiv$ E"(1) by blast+
5391     AOT_hence 0: <[[ $\mathbb{P}$ ]zy]>
5392       by (metis RN " $\equiv$ E"(1) "pred-1-1:1" "sc-eq-fur:2" " $\rightarrow$ E")
5393     AOT_thus <y = m>
5394       using "pred-func:1"[THEN " $\rightarrow$ E", OF "&I"]  $\xi$  by metis
5395   qed
5396   ultimately AOT_have <[ $\mathbb{N}$ ]z & m = z'>
5397     by (rule "&I")
5398   AOT_hence < $\exists n m = n'$ >
5399     by (rule " $\exists$ I")
5400   hence ?thesis
5401     by (rule " $\vee$ I")
5402 }
5403 moreover {
5404   AOT_assume <0 = $\mathbb{P}$  m>
5405   AOT_hence <0 = m>
5406     using "id-R-thm:3"[unconstrain  $\mathcal{R}$ , unvarify  $\beta$  x, OF "zero:2", OF "pred-thm:2",
5407     THEN " $\rightarrow$ E", OF "pred-1-1:4", THEN " $\rightarrow$ E"]
5408   by auto
5409   hence ?thesis using id_sym " $\vee$ I" by blast
5410 }
5411 ultimately show ?thesis
5412   by (metis " $\vee$ E"(2) "raa-cor:1")
5413 qed
5414
5415 AOT_theorem "suc-thm": <[[ $\mathbb{P}$ ]n n'> (807)
5416 proof -

```

```

5417 AOT_obtain x where m_is_n: <x = n'>
5418   using "free-thms:1"[THEN "≡E"(1), OF "def-suc[den2]"]
5419   using "∃E" by metis
5420 AOT_have <A([N]n' & [P]n n')>
5421   apply (rule "rule=E"[rotated, OF suc_eq_desc[symmetric]])
5422   apply (rule "actual-desc:4"[THEN "→E"])
5423   by (simp add: "def-suc[den1]")
5424 AOT_hence <A[N]n'> and <A[P]n n'>
5425   using "Act-Basic:2" "≡E"(1) "&E" by blast+
5426 AOT_hence <A[P]nx>
5427   using m_is_n[symmetric] "rule=E" by fast+
5428 AOT_hence <[P]nx>
5429   by (metis RN "≡E"(1) "pred-1-1:1" "sc-eq-fur:2" "→E")
5430 thus ?thesis
5431   using m_is_n "rule=E" by fast
5432 qed
5433
5434 AOT_define Numeral1 :: <κs> ("1")
5435   "numerals:1": <1 =df 0'>
5436
5437 AOT_theorem "prec-facts:1": <[P]0 1>
5438   by (auto intro: "numerals:1"[THEN "rule-id-df:2:b[zero]",
5439     OF "def-suc[den2]"[unconstrain n, unvarify β,
5440     OF "zero:2", THEN "→E", OF "0-n"]])
5441     "suc-thm"[unconstrain n, unvarify β, OF "zero:2",
5442     THEN "→E", OF "0-n"])
5443
5444 (* TODO: more theorems *)
5445
5446 (* Note: we forgo restricted variables for natural cardinals. *)
5447 AOT_define Finite :: <τ ⇒ φ> (<Finite'('_)>)
5448   "inf-card:1": <Finite(x) ≡df NaturalCardinal(x) & [N]x>
5449 AOT_define Infinite :: <τ ⇒ φ> (<Infinite'('_)>)
5450   "inf-card:2": <Infinite(x) ≡df NaturalCardinal(x) & ¬Finite(x)>
5451
5452 AOT_theorem "inf-card-exist:1": <NaturalCardinal(#0!)>
5453   by (safe intro!: card[THEN "≡dfI"] "∃I"(1)[where τ=<<0!>>] "=I"
5454     "num-def:2"[unvarify G] "oa-exist:1")
5455
5456 AOT_theorem "inf-card-exist:2": <Infinite(#0!)>
5457 proof (safe intro!: "inf-card:2"[THEN "≡dfI"] "&I" "inf-card-exist:1")
5458   AOT_show <¬Finite(#0!)>
5459 proof (rule "raa-cor:2")
5460   AOT_assume <Finite(#0!)>
5461   AOT_hence 0: <[N]#0!>
5462     using "inf-card:1"[THEN "≡dfE"] "&E"(2) by blast
5463   AOT_have <Numbers(#0!, [λz A0!z])>
5464     using "eq-num:3"[unvarify G, OF "oa-exist:1"].
5465   AOT_hence <#0! = #0!>
5466     using "eq-num:2"[unvarify x G, THEN "≡E"(1), OF "oa-exist:1",
5467     OF "num-def:2"[unvarify G], OF "oa-exist:1"]
5468   by blast
5469   AOT_hence <[N]#0! & #0! = #0!>
5470     using 0 "&I" by blast
5471   AOT_hence <∃x ([N]x & x = #0!)>
5472     using "num-def:2"[unvarify G, OF "oa-exist:1"] "∃I"(1) by fast
5473   AOT_hence <◇∃y ([E!]y & ∀u (A[0!]u → u ≠E y))>
5474     using "modal-axiom"[axiom_inst, unvarify G, THEN "→E", OF "oa-exist:1"] by blast
5475   AOT_hence <∃y ◇([E!]y & ∀u (A[0!]u → u ≠E y))>
5476     using "BF◇"[THEN "→E"] by blast
5477   then AOT_obtain b where <◇([E!]b & ∀u (A[0!]u → u ≠E b))>
5478     using "∃E"[rotated] by blast
5479   AOT_hence <◇[E!]b> and 2: <◇∀u (A[0!]u → u ≠E b)>

```

```

5480     using "KBasic2:3"[THEN "→E"] "&E" by blast+
5481 AOT_hence <[λx ◇[E!]x]b>
5482   by (auto intro!: "β←C"(1) "cqt:2")
5483 moreover AOT_have <0! = [λx ◇[E!]x]>
5484   by (rule "rule-id-df:1[zero]"[OF "oa:1"]) "cqt:2"
5485 ultimately AOT_have b_ord: <0!b>
5486   using "rule=E" id_sym by fast
5487 AOT_hence <A0!b>
5488   by (meson "≡E"(1) "oa-facts:7")
5489 moreover AOT_have 2: <∀u (A[0!]u → u ≠E b)>
5490   using "modal-lemma"[unvarify G, unconstrain v, OF "oa-exist:1",
5491     THEN "→E", OF b_ord, THEN "→E", OF 2].
5492 ultimately AOT_have <b ≠E b>
5493   using "Ordinary.∀E"[OF 2, unconstrain α, THEN "→E",
5494     OF b_ord, THEN "→E"] by blast
5495 AOT_hence <¬(b =E b)>
5496   by (metis "≡E"(1) "thm-neg=E")
5497 moreover AOT_have <b =E b>
5498   using "ord=Eequiv:1"[THEN "→E", OF b_ord].
5499 ultimately AOT_show <p & ¬p> for p
5500   by (metis "raa-cor:3")
5501 qed
5502 qed
5503
5504
5505
5506 (*<*)
5507 end
5508 (*>*)
5509

```

A.13. Additional Theorems

```

1  theory AOT_misc
2    imports AOT_NaturalNumbers
3  begin
4
5  AOT_theorem PossiblyNumbersEmptyPropertyImpliesZero:
6    <◇Numbers(x, [λz 0!z & z ≠E z]) → x = 0>
7  proof(rule "→I")
8    AOT_have <Rigid([λz 0!z & z ≠E z])>
9    proof (safe intro!: "df-rigid-rel:1"[THEN "≡dfI"] "&I" "cqt:2";
10         rule RN; safe intro!: GEN "→I")
11      AOT_modally_strict {
12        fix x
13        AOT_assume <[λz 0!z & z ≠E z]x>
14        AOT_hence <0!x & x ≠E x> by (rule "β→C")
15        moreover AOT_have <x =E x> using calculation[THEN "&E"(1)]
16          by (metis "ord=Eequiv:1" "vdash-properties:10")
17        ultimately AOT_have <x =E x & ¬x =E x>
18          by (metis "con-dis-i-e:1" "con-dis-i-e:2:b" "intro-elim:3:a" "thm-neg=E")
19        AOT_thus <□[λz 0!z & z ≠E z]x> using "raa-cor:1" by blast
20      }
21    qed
22    AOT_hence <□∀x (Numbers(x, [λz 0!z & z ≠E z]) → □Numbers(x, [λz 0!z & z ≠E z]))>
23      by (safe intro!: "num-cont:2"[unvarify G, THEN "→E"] "cqt:2")
24    AOT_hence <∀x □(Numbers(x, [λz 0!z & z ≠E z]) → □Numbers(x, [λz 0!z & z ≠E z]))>
25      using "BFs:2"[THEN "→E"] by blast
26    AOT_hence <□(Numbers(x, [λz 0!z & z ≠E z]) → □Numbers(x, [λz 0!z & z ≠E z]))>
27      using "∇E"(2) by auto
28    moreover AOT_assume <◇Numbers(x, [λz 0!z & z ≠E z])>
29    ultimately AOT_have <ℳNumbers(x, [λz 0!z & z ≠E z])>
30      using "sc-eq-box-box:1"[THEN "≡E"(1), THEN "→E", THEN "nec-imp-act"[THEN "→E"]]
31      by blast
32    AOT_hence <Numbers(x, [λz ℳ[λz 0!z & z ≠E z]z])>
33      by (safe intro!: "eq-num:1"[unvarify G, THEN "≡E"(1)] "cqt:2")
34    AOT_hence <x = #[λz 0!z & z ≠E z]>
35      by (safe intro!: "eq-num:2"[unvarify G, THEN "≡E"(1)] "cqt:2")
36    AOT_thus <x = 0>
37      using "cqt:2"(1) "rule-id-df:2:b[zero]" "rule=E" "zero:1" by blast
38  qed
39
40  AOT_define Numbers' :: <τ ⇒ τ ⇒ φ> (<Numbers''(_,_)>)
41    <Numbers'(x, G) ≡df A!x & G↓ & ∀F (x[F] ≡ F ≈E G)>
42  AOT_theorem Numbers'equiv: <Numbers'(x, G) ≡ A!x & ∀F (x[F] ≡ F ≈E G)>
43    by (AOT_subst_def Numbers')
44    (auto intro!: "≡I" "→I" "&I" "cqt:2" dest: "&E")
45
46  AOT_theorem Numbers'DistinctZeroes:
47    <∃x∃y (◇Numbers'(x, [λz 0!z & z ≠E z]) & ◇Numbers'(y, [λz 0!z & z ≠E z]) & x ≠ y)>
48  proof -
49    AOT_obtain w1 where <∃w w1 ≠ w>
50      using "two-worlds-exist:4" "PossibleWorld.∃E"[rotated] by fast
51    then AOT_obtain w2 where distinct_worlds: <w1 ≠ w2>
52      using "PossibleWorld.∃E"[rotated] by blast
53    AOT_obtain x where x_prop:
54      <A!x & ∀F (x[F] ≡ w1 ⊨ F ≈E [λz 0!z & z ≠E z])>
55      using "A-objects"[axiom_inst] "∃E"[rotated] by fast
56    moreover AOT_obtain y where y_prop:
57      <A!y & ∀F (y[F] ≡ w2 ⊨ F ≈E [λz 0!z & z ≠E z])>
58      using "A-objects"[axiom_inst] "∃E"[rotated] by fast
59    moreover {
60      fix x w
61      AOT_assume x_prop: <A!x & ∀F (x[F] ≡ w ⊨ F ≈E [λz 0!z & z ≠E z])>

```

```

62   AOT_have <∀F w ⊨ (x[F] ≡ F ≈E [λz 0!z & z ≠E z])>
63   proof (safe intro!: GEN "conj-dist-w:4"[unvarify p q, OF "log-prop-prop:2",
64         OF "log-prop-prop:2", THEN "≡E"(2)] "≡I" "→I")
65     fix F
66     AOT_assume <w ⊨ x[F]>
67     AOT_hence <◇x[F]>
68       using "fund:1"[unvarify p, OF "log-prop-prop:2", THEN "≡E"(2),
69         OF "PossibleWorld.∃I"] by blast
70     AOT_hence <x[F]>
71       by (metis "en-eq:3[1]" "intro-elim:3:a")
72     AOT_thus <w ⊨ (F ≈E [λz 0!z & z ≠E z])>
73       using x_prop[THEN "&E"(2), THEN "∀E"(2), THEN "≡E"(1)] by blast
74   next
75     fix F
76     AOT_assume <w ⊨ (F ≈E [λz 0!z & z ≠E z])>
77     AOT_hence <x[F]>
78       using x_prop[THEN "&E"(2), THEN "∀E"(2), THEN "≡E"(2)] by blast
79     AOT_hence <□x[F]>
80       using "pre-en-eq:1[1]"[THEN "→E"] by blast
81     AOT_thus <w ⊨ x[F]>
82       using "fund:2"[unvarify p, OF "log-prop-prop:2", THEN "≡E"(1)]
83         "PossibleWorld.∀E" by fast
84   qed
85   AOT_hence <w ⊨ ∀F (x[F] ≡ F ≈E [λz 0!z & z ≠E z])>
86     using "conj-dist-w:5"[THEN "≡E"(2)] by fast
87   moreover {
88     AOT_have <□[λz 0!z & z ≠E z]↓>
89       by (safe intro!: RN "cqt:2")
90     AOT_hence <w ⊨ [λz 0!z & z ≠E z]↓>
91       using "fund:2"[unvarify p, OF "log-prop-prop:2", THEN "≡E"(1),
92         THEN "PossibleWorld.∀E"] by blast
93   }
94   moreover {
95     AOT_have <□A!x>
96       using x_prop[THEN "&E"(1)] by (metis "oa-facts:2" "→E")
97     AOT_hence <w ⊨ A!x>
98       using "fund:2"[unvarify p, OF "log-prop-prop:2",
99         THEN "≡E"(1), THEN "PossibleWorld.∀E"] by blast
100  }
101  ultimately AOT_have <w ⊨ (A!x & [λz 0!z & z ≠E z]↓ &
102    ∀F (x[F] ≡ F ≈E [λz 0!z & z ≠E z]))>
103    using "conj-dist-w:1"[unvarify p q, OF "log-prop-prop:2",
104      OF "log-prop-prop:2", THEN "≡E"(2), OF "&I"] by auto
105  AOT_hence <∃w w ⊨ (A!x & [λz 0!z & z ≠E z]↓ &
106    ∀F (x[F] ≡ F ≈E [λz 0!z & z ≠E z]))>
107    using "PossibleWorld.∃I" by auto
108  AOT_hence <◇(A!x & [λz 0!z & z ≠E z]↓ & ∀F (x[F] ≡ F ≈E [λz 0!z & z ≠E z]))>
109    using "fund:1"[unvarify p, OF "log-prop-prop:2", THEN "≡E"(2)] by blast
110  AOT_hence <◇Numbers'(x, [λz 0!z & z ≠E z])>
111    by (AOT_subst_def Numbers')
112  }
113  ultimately AOT_have <◇Numbers'(x, [λz 0!z & z ≠E z])>
114    and <◇Numbers'(y, [λz 0!z & z ≠E z])>
115  by auto
116  moreover AOT_have <x ≠ y>
117  proof (rule "ab-obey:2"[THEN "→E"])
118    AOT_have <□¬∃u [λz 0!z & z ≠E z]u>
119    proof (safe intro!: RN "raa-cor:2")
120      AOT_modally_strict {
121        AOT_assume <∃u [λz 0!z & z ≠E z]u>
122        then AOT_obtain u where <[λz 0!z & z ≠E z]u>
123          using "Ordinary.∃E"[rotated] by blast
124        AOT_hence <0!u & u ≠E u>

```

```

125     by (rule "β→C")
126 AOT_hence <¬(u =E u)>
127     by (metis "con-dis-taut:2" "intro-elim:3:d" "modus-tollens:1"
128         "raa-cor:3" "thm-neg=E")
129 AOT_hence <u =E u & ¬u =E u>
130     by (metis "modus-tollens:1" "ord=Eequiv:1" "raa-cor:3" Ordinary.ψ)
131 AOT_thus <p & ¬p> for p
132     by (metis "raa-cor:1")
133 }
134 qed
135 AOT_hence nec_not_ex: <∀w w' | ¬∃u [λz 0!z & z ≠E z]u>
136     using "fund:2"[unvarify p, OF "log-prop-prop:2", THEN "≡E"(1)] by blast
137 AOT_have <□([λy p]x ≡ p)> for x p
138     by (safe intro!: RN "beta-C-meta"[THEN "→E"] "cqt:2")
139 AOT_hence <∀w w' | ([λy p]x ≡ p)> for x p
140     using "fund:2"[unvarify p, OF "log-prop-prop:2", THEN "≡E"(1)] by blast
141 AOT_hence world_prop_beta: <∀w (w | [λy p]x ≡ w | p)> for x p
142     using "conj-dist-w:4"[unvarify p, OF "log-prop-prop:2", THEN "≡E"(1)]
143         "PossibleWorld.∀E" "PossibleWorld.∀I" by meson
144
145 AOT_have <∃p (w1 | p & ¬w2 | p)>
146 proof(rule "raa-cor:1")
147     AOT_assume 0: <¬∃p (w1 | p & ¬w2 | p)>
148     AOT_have 1: <w1 | p → w2 | p> for p
149     proof(safe intro!: GEN "→I")
150         AOT_assume <w1 | p>
151         AOT_thus <w2 | p>
152         using 0 "con-dis-i-e:1" "∃I"(2) "raa-cor:4" by fast
153     qed
154     moreover AOT_have <w2 | p → w1 | p> for p
155     proof(safe intro!: GEN "→I")
156         AOT_assume <w2 | p>
157         AOT_hence <¬w2 | ¬p>
158         using "coherent:2" "intro-elim:3:a" by blast
159         AOT_hence <¬w1 | ¬p>
160         using 1["∀I" p, THEN "∀E"(1), OF "log-prop-prop:2"]
161         by (metis "modus-tollens:1")
162         AOT_thus <w1 | p>
163         using "coherent:1" "intro-elim:3:b" "reductio-aa:1" by blast
164     qed
165     ultimately AOT_have <w1 | p ≡ w2 | p> for p
166     by (metis "intro-elim:2")
167 AOT_hence <w1 = w2>
168     using "sit-identity"[unconstrain s, THEN "→E",
169         OF PossibleWorld.ψ[THEN "world:1"[THEN "≡dfE"], THEN "&E"(1)],
170         unconstrain s', THEN "→E",
171         OF PossibleWorld.ψ[THEN "world:1"[THEN "≡dfE"], THEN "&E"(1)],
172         THEN "≡E"(2)] GEN by fast
173 AOT_thus <w1 = w2 & ¬w1 = w2>
174     using "=-infix" "≡dfE" "con-dis-i-e:1" distinct_worlds by blast
175 qed
176 then AOT_obtain p where 0: <w1 | p & ¬w2 | p>
177     using "∃E"[rotated] by blast
178 AOT_have <y[λy p]>
179 proof (safe intro!: y_prop[THEN "&E"(2), THEN "∀E"(1), THEN "≡E"(2)] "cqt:2")
180     AOT_show <w2 | [λy p] ≈E [λz 0!z & z ≠E z]>
181     proof (safe intro!: "cqt:2" "empty-approx:1"[unvarify F H, THEN RN,
182         THEN "fund:2"[unvarify p, OF "log-prop-prop:2", THEN "≡E"(1)],
183         THEN "PossibleWorld.∀E",
184         THEN "conj-dist-w:2"[unvarify p q, OF "log-prop-prop:2",
185             OF "log-prop-prop:2", THEN "≡E"(1)],
186             THEN "→E"])
187         "conj-dist-w:1"[unvarify p q, OF "log-prop-prop:2",

```

```

188                                     OF "log-prop-prop:2", THEN "≡E"(2)] "&I")
189 AOT_have <¬w2 ⊨ ∃u [λy p]u>
190 proof (rule "raa-cor:2")
191   AOT_assume <w2 ⊨ ∃u [λy p]u>
192   AOT_hence <∃x w2 ⊨ (O!x & [λy p]x)>
193     by (metis "conj-dist-w:6" "intro-elim:3:a")
194   then AOT_obtain x where <w2 ⊨ (O!x & [λy p]x)>
195     using "∃E"[rotated] by blast
196   AOT_hence <w2 ⊨ [λy p]x>
197     using "conj-dist-w:1"[unvarify p q, OF "log-prop-prop:2",
198       OF "log-prop-prop:2", THEN "≡E"(1), THEN "&E"(2)] by blast
199   AOT_hence <w2 ⊨ p>
200     using world_prop_beta[THEN "PossibleWorld.∀E", THEN "≡E"(1)] by blast
201   AOT_thus <w2 ⊨ p & ¬w2 ⊨ p>
202     using 0[THEN "&E"(2)] "&I" by blast
203 qed
204 AOT_thus <w2 ⊨ ¬∃u [λy p]u>
205   by (safe intro!: "coherent:1"[unvarify p, OF "log-prop-prop:2",
206     THEN "≡E"(2)])
207 next
208   AOT_show <w2 ⊨ ¬∃v [λz O!z & z ≠E z]v>
209     using nec_not_ex[THEN "PossibleWorld.∀E"] by blast
210 qed
211 qed
212 moreover AOT_have <¬x[λy p]>
213 proof(rule "raa-cor:2")
214   AOT_assume <x[λy p]>
215   AOT_hence "w1 ⊨ [λy p] ≈E [λz O!z & z ≠E z]"
216     using x_prop[THEN "&E"(2), THEN "∀E"(1), THEN "≡E"(1)]
217       "prop-prop2:2" by blast
218   AOT_hence "¬w1 ⊨ ¬[λy p] ≈E [λz O!z & z ≠E z]"
219     using "coherent:2"[unvarify p, OF "log-prop-prop:2", THEN "≡E"(1)] by blast
220   moreover AOT_have "w1 ⊨ ¬([λy p] ≈E [λz O!z & z ≠E z])"
221   proof (safe intro!: "cqt:2" "empty-approx:2"[unvarify F H, THEN RN,
222     THEN "fund:2"[unvarify p, OF "log-prop-prop:2", THEN "≡E"(1)],
223     THEN "PossibleWorld.∀E",
224     THEN "conj-dist-w:2"[unvarify p q, OF "log-prop-prop:2",
225     OF "log-prop-prop:2", THEN "≡E"(1)], THEN "→E"]
226     "conj-dist-w:1"[unvarify p q, OF "log-prop-prop:2",
227     OF "log-prop-prop:2", THEN "≡E"(2)] "&I")
228   fix u
229   AOT_have <w1 ⊨ O!u>
230     using Ordinary.ψ[THEN RN,
231     THEN "fund:2"[unvarify p, OF "log-prop-prop:2", THEN "≡E"(1)],
232     THEN "PossibleWorld.∀E"] by simp
233   moreover AOT_have <w1 ⊨ [λy p]u>
234     by (safe intro!: world_prop_beta[THEN "PossibleWorld.∀E", THEN "≡E"(2)]
235     0[THEN "&E"(1)])
236   ultimately AOT_have <w1 ⊨ (O!u & [λy p]u)>
237     using "conj-dist-w:1"[unvarify p q, OF "log-prop-prop:2",
238     OF "log-prop-prop:2", THEN "≡E"(2),
239     OF "&I"] by blast
240   AOT_hence <∃x w1 ⊨ (O!x & [λy p]x)>
241     by (rule "∃I")
242   AOT_thus <w1 ⊨ ∃u [λy p]u>
243     by (metis "conj-dist-w:6" "intro-elim:3:b")
244 next
245   AOT_show <w1 ⊨ ¬∃v [λz O!z & z ≠E z]v>
246     using "PossibleWorld.∀E" nec_not_ex by fastforce
247 qed
248 ultimately AOT_show <p & ¬p> for p
249   using "raa-cor:3" by blast
250 qed

```



```

251   ultimately AOT_have <y[λy p] & ¬x[λy p]>
252   using "&I" by blast
253   AOT_hence <∃F (y[F] & ¬x[F])>
254   by (metis "existential:1" "prop-prop2:2")
255   AOT_thus <∃F (x[F] & ¬y[F]) ∨ ∃F (y[F] & ¬x[F])>
256   by (rule "∨I")
257   qed
258   ultimately AOT_have <⟨Numbers'(x, [λz 0!z & z ≠E z]) &
259   ⟨Numbers'(y, [λz 0!z & z ≠E z]) & x ≠ y⟩
260   using "&I" by blast
261   AOT_thus <∃x∃y (⟨Numbers'(x, [λz 0!z & z ≠E z]) &
262   ⟨Numbers'(y, [λz 0!z & z ≠E z]) & x ≠ y⟩)
263   using "∃I"(2)[where β=x] "∃I"(2)[where β=y] by auto
264   qed
265
266   AOT_theorem restricted_identity:
267   <x =R y ≡ (InDomainOf(x, R) & InDomainOf(y, R) & x = y)>
268   by (auto intro!: "≡I" "→I" "&I"
269       dest: "id-R-thm:2"[THEN "→E"] "&E"
270           "id-R-thm:3"[THEN "→E"]
271           "id-R-thm:4"[THEN "→E", OF "∨I"(1), THEN "≡E"(2)])
272
273   AOT_theorem induction': <∀F ([F]0 & ∀n([F]n → [F]n') → ∀n [F]n)>
274   proof(rule GEN; rule "→I")
275     fix F n
276     AOT_assume A: <[F]0 & ∀n([F]n → [F]n')>
277     AOT_have <∀n∀m([P]nm → ([F]n → [F]m))>
278     proof(safe intro!: "Number.GEN" "→I")
279       fix n m
280       AOT_assume <[P]nm>
281       moreover AOT_have <[P]n n'>
282       using "suc-thm".
283       ultimately AOT_have m_eq_suc_n: <m = n'>
284       using "pred-func:1"[unvarify z, OF "def-suc[den2]", THEN "→E", OF "&I"]
285       by blast
286       AOT_assume <[F]n>
287       AOT_hence <[F]n'>
288       using A[THEN "&E"(2), THEN "Number.∨E", THEN "→E"] by blast
289       AOT_thus <[F]m>
290       using m_eq_suc_n[symmetric] "rule=E" by fast
291     qed
292     AOT_thus <∀n [F]n>
293     using induction[THEN "∨E"(2), THEN "→E", OF "&I", OF A[THEN "&E"(1)]]
294     by simp
295   qed
296
297   AOT_define ExtensionOf :: <τ ⇒ Π ⇒ φ> (<ExtensionOf'(_,_)>)
298   "exten-property:1": <ExtensionOf(x, [G]) ≡df A!x & G↓ & ∀F(x[F] ≡ ∀z([F]z ≡ [G]z))> (307.1)
299
300   AOT_define OrdinaryExtensionOf :: <τ ⇒ Π ⇒ φ> (<OrdinaryExtensionOf'(_,_)>)
301   <OrdinaryExtensionOf(x, [G]) ≡df A!x & G↓ & ∀F(x[F] ≡ ∀z(0!z → ([F]z ≡ [G]z)))>
302
303   AOT_theorem BeingOrdinaryExtensionOfDenotes:
304   <[λx OrdinaryExtensionOf(x, [G])↓>
305   proof(rule "safe-ext"[axiom_inst, THEN "→E", OF "&I"])
306     AOT_show <[λx A!x & G↓ & [λx ∀F(x[F] ≡ ∀z(0!z → ([F]z ≡ [G]z)))]x↓>
307     by "cqt:2"
308   next
309     AOT_show <□∀x (A!x & G↓ & [λx ∀F (x[F] ≡ ∀z (0!z → ([F]z ≡ [G]z))))x ≡
310     OrdinaryExtensionOf(x, [G])>
311     proof(safe intro!: RN GEN)
312       AOT_modally_strict {
313         fix x

```

```

314   AOT_modally_strict {
315     AOT_have <[ $\lambda x \forall F (x[F] \equiv \forall z (O!z \rightarrow ([F]z \equiv [G]z)))$ ] $\downarrow$ >
316     proof (safe intro!: "Comprehension_3"[THEN " $\rightarrow$ E"] RN GEN
317           " $\rightarrow$ I" " $\equiv$ I" Ordinary.GEN)
318       AOT_modally_strict {
319         fix F H u
320         AOT_assume < $\Box H \equiv_E F$ >
321         AOT_hence < $\forall u([H]u \equiv [F]u)$ >
322           using eqE[THEN " $\equiv_{df} E$ ", THEN "&E"(2)] "qml:2"[axiom_inst, THEN " $\rightarrow$ E"]
323           by blast
324         AOT_hence 0: < $[H]u \equiv [F]u$ > using "Ordinary. $\forall E$ " by fast
325         {
326           AOT_assume < $\forall u([F]u \equiv [G]u)$ >
327           AOT_hence 1: < $[F]u \equiv [G]u$ > using "Ordinary. $\forall E$ " by fast
328           AOT_show < $[G]u$ > if < $[H]u$ > using 0 1 " $\equiv E$ "(1) that by blast
329           AOT_show < $[H]u$ > if < $[G]u$ > using 0 1 " $\equiv E$ "(2) that by blast
330         }
331         {
332           AOT_assume < $\forall u([H]u \equiv [G]u)$ >
333           AOT_hence 1: < $[H]u \equiv [G]u$ > using "Ordinary. $\forall E$ " by fast
334           AOT_show < $[G]u$ > if < $[F]u$ > using 0 1 " $\equiv E$ "(1,2) that by blast
335           AOT_show < $[F]u$ > if < $[G]u$ > using 0 1 " $\equiv E$ "(1,2) that by blast
336         }
337       }
338     qed
339   }
340   AOT_thus <( $A!x \ \& \ G\downarrow \ \& \ [\lambda x \forall F (x[F] \equiv \forall z (O!z \rightarrow ([F]z \equiv [G]z)))$ ] $x \equiv$ 
341     OrdinaryExtensionOf( $x, [G]$ )>
342     apply (AOT_subst_def OrdinaryExtensionOf)
343     apply (AOT_subst <[ $\lambda x \forall F (x[F] \equiv \forall z (O!z \rightarrow ([F]z \equiv [G]z)))$ ] $x$ >
344           < $\forall F (x[F] \equiv \forall z (O!z \rightarrow ([F]z \equiv [G]z)))$ >)
345     by (auto intro!: "beta-C-meta"[THEN " $\rightarrow$ E"] simp: "oth-class-taut:3:a")
346   }
347   qed
348   qed
349
350   text<Fragments of PLM's theory of Concepts.>
351
352   AOT_define FimpG :: < $\Pi \Rightarrow \Pi \Rightarrow \varphi$ > (infix1 < $\Leftrightarrow$ > 50)
353     "F-imp-G": < $[G] \Rightarrow [F] \equiv_{df} F\downarrow \ \& \ G\downarrow \ \& \ \Box \forall x ([G]x \rightarrow [F]x)$ > (432)
354
355   AOT_define concept :: < $\Pi$ > (<C!>)
356     concepts: < $C! \equiv_{df} A!$ > (593)
357
358   AOT_register_rigid_restricted_type
359     Concept: < $C!\kappa$ >
360   proof
361     AOT_modally_strict {
362       AOT_have < $\exists x A!x$ >
363         using "o-objects-exist:2" "qml:2"[axiom_inst] " $\rightarrow$ E" by blast
364       AOT_thus < $\exists x C!x$ >
365         using "rule-id-df:1[zero]"[OF concepts, OF "oa-exist:2"] "rule=E" id_sym
366         by fast
367     }
368   next
369     AOT_modally_strict {
370       AOT_show < $C!\kappa \rightarrow \kappa\downarrow$ > for  $\kappa$ 
371         using "cqt:5:a"[axiom_inst, THEN " $\rightarrow$ E", THEN "&E"(2)] " $\rightarrow$ I"
372         by blast
373     }
374   next
375     AOT_modally_strict {
376       AOT_have < $\forall x(A!x \rightarrow \Box A!x)$ >

```

```

377     by (simp add: "oa-facts:2" GEN)
378 AOT_thus <∀x(C!x → □C!x)>
379     using "rule-id-df:1[zero]"[OF concepts, OF "oa-exist:2"] "rule=E" id_sym
380     by fast
381 }
382 qed
383
384 AOT_register_variable_names
385     Concept: c d e
386
387 AOT_theorem "concept-comp:1": <∃x(C!x & ∀F(x[F] ≡ φ{F}))> (595.1)
388     using concepts[THEN "rule-id-df:1[zero]", OF "oa-exist:2", symmetric]
389     "A-objects"[axiom_inst]
390     "rule=E" by fast
391
392 AOT_theorem "concept-comp:2": <∃!x(C!x & ∀F(x[F] ≡ φ{F}))> (595.2)
393     using concepts[THEN "rule-id-df:1[zero]", OF "oa-exist:2", symmetric]
394     "A-objects!"
395     "rule=E" by fast
396
397 AOT_theorem "concept-comp:3": <ιx(C!x & ∀F(x[F] ≡ φ{F}))↓> (595.3)
398     using "concept-comp:2" "A-Exists:2"[THEN "≡E"(2)] "RA[2]" by blast
399
400 AOT_theorem "concept-comp:4": (595.4)
401     <ιx(C!x & ∀F(x[F] ≡ φ{F})) = ιx(A!x & ∀F(x[F] ≡ φ{F}))>
402     using "=I"(1)[OF "concept-comp:3"]
403     "rule=E"[rotated]
404     concepts[THEN "rule-id-df:1[zero]", OF "oa-exist:2"]
405     by fast
406
407 AOT_define conceptInclusion :: <τ ⇒ τ ⇒ φ> (infixl <=> 100)
408     "con:1": <c ≲ d ≡df ∀F(c[F] → d[F])> (605.1)
409
410
411 AOT_define conceptOf :: <τ ⇒ τ ⇒ φ> (<ConceptOf'(_,_)>)
412     "concept-of-G": <ConceptOf(c,G) ≡df G↓ & ∀F(c[F] ≡ [G] ⇒ [F])> (651)
413
414 AOT_theorem ConceptOfOrdinaryProperty: <([H] ⇒ O!) → [λx ConceptOf(x,H)]↓>
415 proof(rule "→I")
416     AOT_assume <[H] ⇒ O!>
417     AOT_hence <□∀x([H]x → O!x)>
418     using "F-imp-G"[THEN "≡dfE"] "&E" by blast
419     AOT_hence <□□∀x([H]x → O!x)>
420     using "S5Basic:6"[THEN "≡E"(1)] by blast
421     moreover AOT_have <□□∀x([H]x → O!x) →
422         □∀FVG(□(G ≡E F) → ([H] ⇒ [F] ≡ [H] ⇒ [G]))>
423     proof(rule RM; safe intro!: "→I" GEN "≡I")
424         AOT_modally_strict {
425             fix F G
426             AOT_assume 0: <□∀x([H]x → O!x)>
427             AOT_assume <□G ≡E F>
428             AOT_hence 1: <□∀u([G]u ≡ [F]u)>
429             by (AOT_subst_thm eqE[THEN "≡Df", THEN "≡S"(1), OF "&I",
430                 OF "cqt:2[const_var]"[axiom_inst],
431                 OF "cqt:2[const_var]"[axiom_inst], symmetric])
432         {
433             AOT_assume <[H] ⇒ [F]>
434             AOT_hence <□∀x([H]x → [F]x)>
435             using "F-imp-G"[THEN "≡dfE"] "&E" by blast
436             moreover AOT_modally_strict {
437                 AOT_assume <∀x([H]x → O!x)>
438                 moreover AOT_assume <∀u([G]u ≡ [F]u)>
439                 moreover AOT_assume <∀x([H]x → [F]x)>

```

```

440     ultimately AOT_have <[H]x → [G]x> for x
441     by (auto intro!: "→I" dest!: "∀E"(2) dest: "→E" "≡E")
442     AOT_hence <∀x([H]x → [G]x)>
443     by (rule GEN)
444   }
445   ultimately AOT_have <□∀x([H]x → [G]x)>
446   using "RN[prem]"[where
447     Γ="{«∀x([H]x → O!x)», «∀u([G]u ≡ [F]u)», «∀x([H]x → [F]x)»}"
448   using 0 1 by fast
449   AOT_thus <[H] ⇒ [G]>
450   by (AOT_subst_def "F-imp-G")
451     (safe intro!: "cqt:2" "&I")
452   }
453   {
454     AOT_assume <[H] ⇒ [G]>
455     AOT_hence <□∀x([H]x → [G]x)>
456     using "F-imp-G"[THEN "≡dfE"] "&E" by blast
457     moreover AOT_modally_strict {
458       AOT_assume <∀x([H]x → O!x)>
459       moreover AOT_assume <∀u([G]u ≡ [F]u)>
460       moreover AOT_assume <∀x([H]x → [G]x)>
461       ultimately AOT_have <[H]x → [F]x> for x
462       by (auto intro!: "→I" dest!: "∀E"(2) dest: "→E" "≡E")
463       AOT_hence <∀x([H]x → [F]x)>
464       by (rule GEN)
465     }
466     ultimately AOT_have <□∀x([H]x → [F]x)>
467     using "RN[prem]"[where
468       Γ="{«∀x([H]x → O!x)», «∀u([G]u ≡ [F]u)», «∀x([H]x → [G]x)»}"
469     using 0 1 by fast
470     AOT_thus <[H] ⇒ [F]>
471     by (AOT_subst_def "F-imp-G")
472       (safe intro!: "cqt:2" "&I")
473   }
474 }
475 qed
476 ultimately AOT_have <□∀F∀G(□(G ≡E F) → ([H] ⇒ [F] ≡ [H] ⇒ [G]))>
477 using "→E" by blast
478 AOT_hence 0: <[λx ∀F(x[F] ≡ ([H] ⇒ [F]))]↓>
479 using Comprehension_3[THEN "→E"] by blast
480 AOT_show <[λx ConceptOf(x,H)]↓>
481 proof (rule "safe-ext"[axiom_inst, THEN "→E", OF "&I"])
482   AOT_show <[λx C!x & [λx ∀F(x[F] ≡ ([H] ⇒ [F]))]x]↓> by "cqt:2"
483 next
484   AOT_show <□∀x (C!x & [λx ∀F (x[F] ≡ [H] ⇒ [F]))x ≡ ConceptOf(x,H))>
485   proof (rule "RN[prem]"[where Γ="{«[λx ∀F(x[F] ≡ ([H] ⇒ [F]))]↓»}", simplified])
486     AOT_modally_strict {
487       AOT_assume 0: <[λx ∀F (x[F] ≡ [H] ⇒ [F]))]↓>
488       AOT_show <∀x (C!x & [λx ∀F (x[F] ≡ [H] ⇒ [F]))x ≡ ConceptOf(x,H))>
489       proof(safe intro!: GEN "≡I" "→I" "&I")
490         fix x
491         AOT_assume <C!x & [λx ∀F (x[F] ≡ [H] ⇒ [F]))x>
492         AOT_thus <ConceptOf(x,H)>
493         by (AOT_subst_def "concept-of-G")
494           (auto intro!: "&I" "cqt:2" dest: "&E" "β→C")
495       next
496         fix x
497         AOT_assume <ConceptOf(x,H)>
498         AOT_hence <C!x & (H↓ & ∀F(x[F] ≡ [H] ⇒ [F]))>
499         by (AOT_subst_def (reverse) "concept-of-G")
500         AOT_thus <C!x> and <[λx ∀F(x[F] ≡ [H] ⇒ [F]))x>
501         by (auto intro!: "β←C" 0 "cqt:2" dest: "&E")
502     }
503   qed

```

```

503     }
504   next
505     AOT_show <□[λx ∀F(x[F] ≡ ([H] ⇒ [F]))]↓>
506     using "exist-nec"[THEN "→E"] 0 by blast
507   qed
508   qed
509 qed
510
511 AOT_theorem "con-exists:1": <∃c ConceptOf(c,G)> (652.1)
512 proof -
513   AOT_obtain c where <∀F (c[F] ≡ [G] ⇒ [F])>
514     using "concept-comp:1" "Concept.∃E"[rotated] by meson
515   AOT_hence <ConceptOf(c,G)>
516     by (auto intro!: "concept-of-G"[THEN "≡dfI"] "&I" "cqt:2" Concept.ψ)
517   thus ?thesis by (rule "Concept.∃I")
518 qed
519
520 AOT_theorem "con-exists:2": <∃!c ConceptOf(c,G)> (652.2)
521 proof -
522   AOT_have <∃!c ∀F (c[F] ≡ [G] ⇒ [F])>
523     using "concept-comp:2" by simp
524   moreover {
525     AOT_modally_strict {
526       fix x
527       AOT_assume <∀F (x[F] ≡ [G] ⇒ [F])>
528       moreover AOT_have <[G] ⇒ [G]>
529         by (safe intro!: "F-imp-G"[THEN "≡dfI"] "&I" "cqt:2" RN GEN "→I")
530       ultimately AOT_have <x[G]>
531         using "∀E"(2) "≡E" by blast
532       AOT_hence <A!x>
533         using "encoders-are-abstract"[THEN "→E", OF "∃I"(2)] by simp
534       AOT_hence <C!x>
535         using concepts[THEN "rule-id-df:1[zero]", OF "oa-exist:2", symmetric]
536           "rule=E"[rotated]
537       by fast
538     }
539   }
540   ultimately show ?thesis
541     by (AOT_subst <ConceptOf(c,G)> <∀F (c[F] ≡ [G] ⇒ [F])> for: c;
542       AOT_subst_def "concept-of-G")
543     (auto intro!: "≡I" "→I" "&I" "cqt:2" Concept.ψ dest: "&E")
544 qed
545
546 AOT_theorem "con-exists:3": <∃c ConceptOf(c,G)↓> (652.3)
547   by (safe intro!: "A-Exists:2"[THEN "≡E"(2)] "con-exists:2"[THEN "RA[2]"])
548
549
550 AOT_define theConceptOfG :: <τ ⇒ κg> (<c->)
551   "concept-G": <cG =df ∃c ConceptOf(c, G)> (653)
552
553 AOT_theorem "concept-G[den]": <cG↓> (653)
554   by (auto intro!: "rule-id-df:1"[OF "concept-G"]
555     "t=t-proper:1"[THEN "→E"]
556     "con-exists:3")
557
558
559 AOT_theorem "concept-G[concept]": <C!cG> (653)
560 proof -
561   AOT_have <A(C!cG & ConceptOf(cG, G))>
562     by (auto intro!: "actual-desc:2"[unvarify x, THEN "→E"]
563       "rule-id-df:1"[OF "concept-G"]
564       "concept-G[den]"
565       "con-exists:3")

```

```

566 AOT_hence <A!cG>
567   by (metis "Act-Basic:2" "con-dis-i-e:2:a" "intro-elim:3:a")
568 AOT_hence <A!cG>
569   using "rule-id-df:1[zero]"[OF concepts, OF "oa-exist:2"]
570   "rule=E" by fast
571 AOT_hence <A!cG>
572   using "oa-facts:8"[unvarify x, THEN "≡E"(2)] "concept-G[den]" by blast
573 thus ?thesis
574   using "rule-id-df:1[zero]"[OF concepts, OF "oa-exist:2", symmetric]
575   "rule=E" by fast
576 qed
577
578 AOT_theorem "conG-strict": <cG =  $\iota c \forall F(c[F] \equiv [G] \Rightarrow [F])$ > (654)
579 proof (rule "id-eq:3"[unvarify  $\alpha \beta \gamma$ , THEN " $\rightarrow$ E"])
580   AOT_have < $\Box \forall x (C!x \ \& \ \text{ConceptOf}(x,G) \equiv C!x \ \& \ \forall F (x[F] \equiv [G] \Rightarrow [F]))$ >
581     by (auto intro!: "concept-of-G"[THEN "≡dfI"] RN GEN "≡I" " $\rightarrow$ I" "&I" "cqt:2"
582         dest: "&E";
583         auto dest: " $\forall$ E"(2) "≡E"(1,2) dest!: "&E"(2) "concept-of-G"[THEN "≡dfE"])
584   AOT_thus <cG =  $\iota c \ \text{ConceptOf}(c, G) \ \& \ \iota c \ \text{ConceptOf}(c, G) = \iota c \forall F(c[F] \equiv [G] \Rightarrow [F])$ >
585     by (auto intro!: "&I" "rule-id-df:1"[OF "concept-G"] "con-exists:3"
586         "equiv-desc-eq:3"[THEN " $\rightarrow$ E"])
587 qed(auto simp: "concept-G[den]" "con-exists:3" "concept-comp:3")
588
589
590 AOT_theorem "conG-lemma:1": < $\forall F(c_G[F] \equiv [G] \Rightarrow [F])$ > (655.1)
591 proof(safe intro!: GEN "≡I" " $\rightarrow$ I")
592   fix F
593   AOT_have < $\mathcal{A} \forall F(c_G[F] \equiv [G] \Rightarrow [F])$ >
594     using "actual-desc:4"[THEN " $\rightarrow$ E", OF "concept-comp:3",
595         THEN "Act-Basic:2"[THEN "≡E"(1)],
596         THEN "&E"(2)]
597     "conG-strict"[symmetric] "rule=E" by fast
598   AOT_hence < $\mathcal{A}(c_G[F] \equiv [G] \Rightarrow [F])$ >
599     using "logic-actual-nec:3"[axiom_inst, THEN "≡E"(1)] " $\forall$ E"(2)
600     by blast
601   AOT_hence 0: < $\mathcal{A}c_G[F] \equiv \mathcal{A}[G] \Rightarrow [F]$ >
602     using "Act-Basic:5"[THEN "≡E"(1)] by blast
603   {
604     AOT_assume <cG[F]>
605     AOT_hence < $\mathcal{A}c_G[F]$ >
606       by(safe intro!: "en-eq:10[1]"[unvarify x1, THEN "≡E"(2)]
607           "concept-G[den]")
608     AOT_hence < $\mathcal{A}[G] \Rightarrow [F]$ >
609       using 0[THEN "≡E"(1)] by blast
610     AOT_hence < $\mathcal{A}(F \downarrow \ \& \ G \downarrow \ \& \ \Box \forall x([G]x \rightarrow [F]x))$ >
611       by (AOT_subst_def (reverse) "F-imp-G")
612     AOT_hence < $\mathcal{A} \Box \forall x([G]x \rightarrow [F]x)$ >
613       using "Act-Basic:2"[THEN "≡E"(1)] "&E" by blast
614     AOT_hence < $\Box \forall x([G]x \rightarrow [F]x)$ >
615       using "qml-act:2"[axiom_inst, THEN "≡E"(2)] by simp
616     AOT_thus <[G]  $\Rightarrow$  [F]>
617       by (AOT_subst_def "F-imp-G"; auto intro!: "&I" "cqt:2")
618   }
619   {
620     AOT_assume <[G]  $\Rightarrow$  [F]>
621     AOT_hence < $\Box \forall x([G]x \rightarrow [F]x)$ >
622       by (safe dest!: "F-imp-G"[THEN "≡dfE"] "&E"(2))
623     AOT_hence < $\mathcal{A} \Box \forall x([G]x \rightarrow [F]x)$ >
624       using "qml-act:2"[axiom_inst, THEN "≡E"(1)] by simp
625     AOT_hence < $\mathcal{A}(F \downarrow \ \& \ G \downarrow \ \& \ \Box \forall x([G]x \rightarrow [F]x))$ >
626       by (auto intro!: "Act-Basic:2"[THEN "≡E"(2)] "&I" "cqt:2"
627           intro: "RA[2]")
628     AOT_hence < $\mathcal{A}([G] \Rightarrow [F])$ >

```

```

629     by (AOT_subst_def "F-imp-G")
630 AOT_hence <AcG[F]>
631     using 0[THEN "≡E"(2)] by blast
632 AOT_thus <cG[F]>
633     by(safe intro!: "en-eq:10[1]"[unvarify x1, THEN "≡E"(1)]
634         "concept-G[den]")
635 }
636 qed
637
638 AOT_theorem conH_enc_ord:
639   <([H] ⇒ 0!) → □∀F ∀G (□G ≡E F → (cH[F] ≡ cH[G]))>
640 proof(rule "→I")
641   AOT_assume 0: <[H] ⇒ 0!>
642   AOT_have 0: <□([H] ⇒ 0!)>
643     apply (AOT_subst_def "F-imp-G")
644     using 0[THEN "≡dfE"[OF "F-imp-G"]]
645     by (auto intro!: "KBasic:3"[THEN "≡E"(2)] "&I" "exist-nec"[THEN "→E"]
646         dest: "&E" 4[THEN "→E"])
647 moreover AOT_have <□([H] ⇒ 0!) → □∀F ∀G (□G ≡E F → (cH[F] ≡ cH[G]))>
648 proof(rule RM; safe intro!: "→I" GEN)
649   AOT_modally_strict {
650     fix F G
651     AOT_assume <[H] ⇒ 0!>
652     AOT_hence 0: <□∀x ([H]x → 0!x)>
653       by (safe dest!: "F-imp-G"[THEN "≡dfE"] "&E"(2))
654     AOT_assume 1: <□G ≡E F>
655     AOT_assume <cH[F]>
656     AOT_hence <[H] ⇒ [F]>
657       using "conG-lemma:1"[THEN "∀E"(2), THEN "≡E"(1)] by simp
658     AOT_hence 2: <□∀x ([H]x → [F]x)>
659       by (safe dest!: "F-imp-G"[THEN "≡dfE"] "&E"(2))
660     AOT_modally_strict {
661       AOT_assume 0: <∀x ([H]x → 0!x)>
662       AOT_assume 1: <∀x ([H]x → [F]x)>
663       AOT_assume 2: <G ≡E F>
664       AOT_have <∀x ([H]x → [G]x)>
665       proof(safe intro!: GEN "→I")
666         fix x
667         AOT_assume <[H]x>
668         AOT_hence <0!x> and <[F]x>
669           using 0 1 "∀E"(2) "→E" by blast+
670         AOT_thus <[G]x>
671           using 2[THEN eqE[THEN "≡dfE"], THEN "&E"(2)]
672             "∀E"(2) "→E" "≡E"(2) calculation by blast
673       qed
674     }
675     AOT_hence <□∀x ([H]x → [G]x)>
676       using "RN[premise]"[where Γ=<{<∀x ([H]x → 0!x)>,
677         <∀x ([H]x → [F]x)>,
678         <G ≡E F}>, simplified] 0 1 2 by fast
679     AOT_hence <[H] ⇒ [G]>
680       by (safe intro!: "F-imp-G"[THEN "≡dfI"] "&I" "cqt:2")
681     AOT_hence <cH[G]>
682       using "conG-lemma:1"[THEN "∀E"(2), THEN "≡E"(2)] by simp
683   } note 0 = this
684   AOT_modally_strict {
685     fix F G
686     AOT_assume <[H] ⇒ 0!>
687     moreover AOT_assume <□G ≡E F>
688     moreover AOT_have <□F ≡E G>
689       by (AOT_subst <F ≡E G> <G ≡E F>)
690       (auto intro!: calculation(2)
691         eqE[THEN "≡dfI"]

```

```

692             "≡I" "→I" "&I" "cqt:2" Ordinary.GEN
693             dest!: eqE[THEN "≡dfE"] "&E"(2)
694             dest: "≡E"(1,2) "Ordinary.VE"
695             ultimately AOT_show <(cH[F] ≡ cH[G])>
696             using 0 "≡I" "→I" by auto
697         }
698     qed
699     ultimately AOT_show <□∀F ∀G (□G ≡E F → (cH[F] ≡ cH[G]))>
700     using "→E" by blast
701 qed
702
703 AOT_theorem concept_inclusion_denotes_1:
704   <([H] ⇒ 0!) → [λx cH ≲ x]↓>
705 proof(rule "→I")
706   AOT_assume 0: <[H] ⇒ 0!>
707   AOT_show <[λx cH ≲ x]↓>
708   proof(rule "safe-ext"[axiom_inst, THEN "→E", OF "&I"])
709     AOT_show <[λx C!x & ∀F(cH[F] → x[F])↓>
710     by (safe intro!: conjunction_denotes[THEN "→E", OF "&I"]
711         Comprehension_2'[THEN "→E"]
712         conH_enc_ord[THEN "→E", OF 0]) "cqt:2"
713   next
714     AOT_show <□∀x (C!x & ∀F (cH[F] → x[F]) ≡ cH ≲ x)>
715     by (safe intro!: RN GEN; AOT_subst_def "con:1")
716     (auto intro!: "≡I" "→I" "&I" "concept-G[concept]" dest: "&E")
717   qed
718 qed
719
720 AOT_theorem concept_inclusion_denotes_2:
721   <([H] ⇒ 0!) → [λx x ≲ cH]↓>
722 proof(rule "→I")
723   AOT_assume 0: <[H] ⇒ 0!>
724   AOT_show <[λx x ≲ cH]↓>
725   proof(rule "safe-ext"[axiom_inst, THEN "→E", OF "&I"])
726     AOT_show <[λx C!x & ∀F(x[F] → cH[F])↓>
727     by (safe intro!: conjunction_denotes[THEN "→E", OF "&I"]
728         Comprehension_1'[THEN "→E"]
729         conH_enc_ord[THEN "→E", OF 0]) "cqt:2"
730   next
731     AOT_show <□∀x (C!x & ∀F (x[F] → cH[F]) ≡ x ≲ cH)>
732     by (safe intro!: RN GEN; AOT_subst_def "con:1")
733     (auto intro!: "≡I" "→I" "&I" "concept-G[concept]" dest: "&E")
734   qed
735 qed
736
737 AOT_define ThickForm :: <τ ⇒ τ ⇒ φ> (<FormOf'(_,_)>)
738   "tform-of": <FormOf(x,G) ≡df A!x & G↓ & ∀F(x[F] ≡ [G] ⇒ [F])> (434)
739
740 AOT_theorem FormOfOrdinaryProperty: <([H] ⇒ 0!) → [λx FormOf(x,H)]↓>
741 proof(rule "→I")
742   AOT_assume 0: <[H] ⇒ [0!]>
743   AOT_show <[λx FormOf(x,H)]↓>
744   proof (rule "safe-ext"[axiom_inst, THEN "→E", OF "&I"])
745     AOT_show <[λx ConceptOf(x,H)]↓>
746     using 0 ConceptOfOrdinaryProperty[THEN "→E"] by blast
747     AOT_show <□∀x (ConceptOf(x,H) ≡ FormOf(x,H))>
748     proof(safe intro!: RN GEN)
749       AOT_modally_strict {
750         fix x
751         AOT_modally_strict {
752           AOT_have <A!x ≡ A!x>
753           by (simp add: "oth-class-taut:3:a")
754           AOT_hence <C!x ≡ A!x>

```



```

755         using "rule-id-df:1[zero]"[OF concepts, OF "oa-exist:2"]
756         "rule=E" id_sym by fast
757     }
758     AOT_thus <ConceptOf(x,H)  $\equiv$  FormOf(x,H)>
759     by (AOT_subst_def "tform-of";
760         AOT_subst_def "concept-of-G";
761         AOT_subst <C!x> <A!x>)
762     (auto intro!: " $\equiv$ I" " $\rightarrow$ I" "&I" dest: "&E")
763 }
764 qed
765 qed
766 qed
767
768 AOT_theorem equal_E_rigid_one_to_one: <Rigid1-1((=E))>
769 proof (safe intro!: "df-1-1:2"[THEN " $\equiv_{df}$ I"] "&I" "df-1-1:1"[THEN " $\equiv_{df}$ I"]
770       GEN " $\rightarrow$ I" "df-rigid-rel:1"[THEN " $\equiv_{df}$ I"] "=E[denotes]")
771   fix x y z
772   AOT_assume <x =E z & y =E z>
773   AOT_thus <x = y>
774   by (metis "rule=E" "&E"(1) "Conjunction Simplification"(2)
775       "=E-simple:2" id_sym " $\rightarrow$ E")
776 next
777   AOT_have < $\forall x \forall y \square(x =_E y \rightarrow \square x =_E y)$ >
778   proof(rule GEN; rule GEN)
779     AOT_show < $\square(x =_E y \rightarrow \square x =_E y)$ > for x y
780     by (meson RN "deduction-theorem" "id-nec3:1" " $\equiv$ E"(1))
781   qed
782   AOT_hence < $\forall x_1 \dots \forall x_n \square([\equiv_E]x_1 \dots x_n \rightarrow \square[\equiv_E]x_1 \dots x_n)$ >
783   by (rule tuple_forall[THEN " $\equiv_{df}$ I"])
784   AOT_thus < $\square \forall x_1 \dots \forall x_n ([\equiv_E]x_1 \dots x_n \rightarrow \square[\equiv_E]x_1 \dots x_n)$ >
785   using BF[THEN " $\rightarrow$ E"] by fast
786 qed
787
788 AOT_theorem equal_E_domain: <InDomainOf(x,(=E))  $\equiv$  0!x>
789 proof(safe intro!: " $\equiv$ I" " $\rightarrow$ I")
790   AOT_assume <InDomainOf(x,(=E))>
791   AOT_hence < $\exists y x =_E y$ >
792   by (metis " $\equiv_{df}$ E" "df-1-1:5")
793   then AOT_obtain y where <x =E y>
794   using " $\exists$ E"[rotated] by blast
795   AOT_thus <0!x>
796   using "=E-simple:1"[THEN " $\equiv$ E"(1)] "&E" by blast
797 next
798   AOT_assume <0!x>
799   AOT_hence <x =E x>
800   by (metis "ord=Eequiv:1"[THEN " $\rightarrow$ E"])
801   AOT_hence < $\exists y x =_E y$ >
802   using " $\exists$ I"(2) by fast
803   AOT_thus <InDomainOf(x,(=E))>
804   by (metis " $\equiv_{df}$ I" "df-1-1:5")
805 qed
806
807 AOT_theorem shared_urelement_projection_identity:
808   assumes < $\forall y [\lambda x (y[\lambda z [R]zx])]\downarrow$ >
809   shows < $\forall F([F]a \equiv [F]b) \rightarrow [\lambda z [R]za] = [\lambda z [R]zb]$ >
810 proof(rule " $\rightarrow$ I")
811   AOT_assume 0: < $\forall F([F]a \equiv [F]b)$ >
812   {
813     fix z
814     AOT_have < $[\lambda x (z[\lambda z [R]zx])]\downarrow$ >
815     using assms[THEN " $\forall$ E"(2)].
816     AOT_hence 1: < $\forall x \forall y (\forall F ([F]x \equiv [F]y) \rightarrow \square(z[\lambda z [R]zx] \equiv z[\lambda z [R]zy]))$ >
817     using "kirchner-thm-cor:1"[THEN " $\rightarrow$ E"]

```

```

818     by blast
819     AOT_have <□(z[λz [R]za] ≡ z[λz [R]zb])>
820     using 1[THEN "∀E"(2), THEN "∀E"(2), THEN "→E", OF 0] by blast
821 }
822 AOT_hence <∀z □(z[λz [R]za] ≡ z[λz [R]zb])>
823     by (rule GEN)
824 AOT_hence <□∀z(z[λz [R]za] ≡ z[λz [R]zb])>
825     by (rule BF[THEN "→E"])
826 AOT_thus <[λz [R]za] = [λz [R]zb]>
827     by (AOT_subst_def "identity:2")
828     (auto intro!: "&I" "cqt:2")
829 qed
830
831 AOT_theorem shared_urelement_exemplification_identity:
832     assumes <∀y [λx (y[λz [G]x])]>↓>
833     shows <∀F([F]a ≡ [F]b) → ([G]a) = ([G]b)>
834     proof(rule "→I")
835         AOT_assume 0: <∀F([F]a ≡ [F]b)>
836         {
837             fix z
838             AOT_have <[λx (z[λz [G]x])]>↓>
839             using assms[THEN "∀E"(2)].
840             AOT_hence 1: <∀x ∀y (∀F ([F]x ≡ [F]y) → □(z[λz [G]x] ≡ z[λz [G]y]))>
841             using "kirchner-thm-cor:1"[THEN "→E"]
842             by blast
843             AOT_have <□(z[λz [G]a] ≡ z[λz [G]b])>
844             using 1[THEN "∀E"(2), THEN "∀E"(2), THEN "→E", OF 0] by blast
845         }
846         AOT_hence <∀z □(z[λz [G]a] ≡ z[λz [G]b])>
847             by (rule GEN)
848         AOT_hence <□∀z(z[λz [G]a] ≡ z[λz [G]b])>
849             by (rule BF[THEN "→E"])
850         AOT_hence <[λz [G]a] = [λz [G]b]>
851             by (AOT_subst_def "identity:2")
852             (auto intro!: "&I" "cqt:2")
853         AOT_thus <([G]a) = ([G]b)>
854             by (safe intro!: "identity:4"[THEN "≡dfI"] "&I" "log-prop-prop:2")
855     qed
856
857 text<The assumptions of the theorems above are derivable, if the additional
858     introduction rules for the upcoming extension of @{thm "cqt:2[lambda]"}
859     are explicitly allowed (while they are currently not part of the
860     abstraction layer).>
861 notepad
862 begin
863     AOT_modally_strict {
864         AOT_have <∀R∀y [λx (y[λz [R]zx])]>↓>
865         by (safe intro!: GEN "cqt:2" AOT_instance_of_cqt_2_intro_next)
866         AOT_have <∀G∀y [λx (y[λz [G]x])]>↓>
867         by (safe intro!: GEN "cqt:2" AOT_instance_of_cqt_2_intro_next)
868     }
869 end
870
871 end
872

```

References

- [1] Jesse Alama, Paul E. Oppenheimer, and Edward N. Zalta. “Automating Leibniz’s Theory of Concepts”. In: *Automated Deduction - CADE-25 - 25th International Conference on Automated Deduction, Berlin, Germany, August 1-7, 2015, Proceedings*. Ed. by Amy P. Felty and Aart Middeldorp. Vol. 9195. Lecture Notes in Computer Science. Springer, 2015, pp. 73–97. DOI: 10.1007/978-3-319-21401-6.
- [2] Christoph Benz Müller. “Universal (Meta-)Logical Reasoning: Recent Successes”. In: *Science of Computer Programming* 172 (2019), pp. 48–62. DOI: 10.1016/j.scico.2018.10.008.
- [3] Christoph Benz Müller, Maximilian Claus, and Nik Sultana. “Systematic Verification of the Modal Logic Cube in Isabelle/HOL”. In: *PxTP 2015*. Ed. by Cezary Kaliszyk and Andrei Paskevich. Vol. 186. Berlin, Germany: EPTCS, 2015, pp. 27–41. DOI: 10.4204/EPTCS.186.5.
- [4] Christoph Benz Müller and David Fuenmayor. “Computer-supported Analysis of Positive Properties, Ultrafilters and Modal Collapse in Variants of Gödel’s Ontological Argument”. In: *Bulletin of the Section of Logic* 49.2 (2020), pp. 127–148. DOI: 10.18778/0138-0680.2020.08.
- [5] Christoph Benz Müller, Xavier Parent, and Leendert van der Torre. “Designing normative theories for ethical and legal reasoning: LogiKEy framework, methodology, and tool support”. In: *Artificial Intelligence* 287 (2020), p. 103348. ISSN: 0004-3702. DOI: 10.1016/j.artint.2020.103348.
- [6] Christoph Benz Müller and Lawrence Paulson. “Quantified Multimodal Logics in Simple Type Theory”. In: *Logica Universalis (Special Issue on Multimodal Logics)* 7.1 (2013), pp. 7–20. DOI: 10.1007/s11787-012-0052-y.
- [7] Christoph Benz Müller and Sebastian Reiche. *Modeling and Automating Public Announcement Logic with Relativized Common Knowledge as a Fragment of HOL in LogiKEy*. 2021. arXiv: 2111.01654 [cs.AI].
- [8] Christoph Benz Müller and Bruno Woltzenlogel Paleo. “Automating Gödel’s Ontological Proof of God’s Existence with Higher-order Automated Theorem Provers”. In: *ECAI 2014*. Ed. by Torsten Schaub, Gerhard Friedrich, and Barry O’Sullivan. Vol. 263. Frontiers in Artificial Intelligence and Applications. IOS Press, 2014, pp. 93–98. DOI: 10.3233/978-1-61499-419-0-93.
- [9] Christoph Benz Müller and Bruno Woltzenlogel Paleo. “Higher-Order Modal Logics: Automation and Applications”. In: July 2015. ISBN: 978-3-319-21767-3. DOI: 10.1007/978-3-319-21768-0_2.

- [10] Christoph Benzmüller et al. “LogiKEy workbench: Deontic logics, logic combinations and expressive ethical and legal reasoning (Isabelle/HOL dataset)”. In: *Data in Brief* 33 (2020), p. 106409. ISSN: 2352-3409. DOI: 10.1016/j.dib.2020.106409.
- [11] Patrick Blackburn and Johan van Benthem. “Modal logic: a semantic perspective”. In: *Handbook of Modal Logic*. Ed. by Patrick Blackburn, Johan Van Benthem, and Frank Wolter. Vol. 3. Studies in Logic and Practical Reasoning. Elsevier, 2007, pp. 1–84. DOI: 10.1016/S1570-2464(07)80004-8.
- [12] Jasmin Christian Blanchette and Tobias Nipkow. “Nitpick: A Counterexample Generator for Higher-Order Logic Based on a Relational Model Finder”. In: *Interactive Theorem Proving*. Ed. by Matt Kaufmann and Lawrence C. Paulson. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 131–146. ISBN: 978-3-642-14052-5. DOI: 10.1007/978-3-642-14052-5_11.
- [13] Sascha Böhme and Tjark Weber. “Fast LCF-Style Proof Reconstruction for Z3”. In: *Interactive Theorem Proving*. Ed. by Matt Kaufmann and Lawrence C. Paulson. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 179–194. ISBN: 978-3-642-14052-5. DOI: 10.1007/978-3-642-14052-5_14.
- [14] George Boolos. “The Consistency of Frege’s Foundations of Arithmetic”. In: *On Being and Saying: Essays in Honor of Richard Cartwright*. Ed. by J. Thomson. Cambridge: MIT Press, 1987, pp. 3–20.
- [15] Thomas Bouton et al. “veriT: An Open, Trustable and Efficient SMT-Solver”. In: *Automated Deduction – CADE-22*. Ed. by Renate A. Schmidt. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 151–156. ISBN: 978-3-642-02959-2. DOI: 10.1007/978-3-642-02959-2_12.
- [16] N.G de Bruijn. “Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem”. In: *Indagationes Mathematicae (Proceedings)* 75.5 (1972), pp. 381–392. ISSN: 1385-7258. DOI: 10.1016/1385-7258(72)90034-0.
- [17] Branden Fitelson and Edward N. Zalta. “Steps Toward a Computational Metaphysics”. In: *Journal Philosophical Logic* 36.2 (2007), pp. 227–247. DOI: 10.1007/s10992-006-9038-7.
- [18] Melvin Fitting. “A tableau system for propositional S5.” In: *Notre Dame Journal of Formal Logic* 18.2 (1977), pp. 292–294. DOI: 10.1305/ndjfl/1093887933.
- [19] Mathias Fleury and Hans-Jörg Schurr. “Reconstructing veriT Proofs in Isabelle/HOL”. In: *Electronic Proceedings in Theoretical Computer Science* 301 (Aug. 2019), pp. 36–50. ISSN: 2075-2180. DOI: 10.4204/eptcs.301.6.
- [20] David Fuenmayor and Christoph Benzmüller. “A Case Study On Computational Hermeneutics: E. J. Lowe’s Modal Ontological Argument”. In: *Beyond Faith and Rationality: Essays on Logic, Religion and Philosophy*. Ed. by Ricardo Silvestre et al. Vol. 34. Sophia Studies in Cross-cultural Philosophy of Traditions and Cultures. Springer Nature Switzerland AG, 2020. Chap. 12. DOI: 10.1007/978-3-030-43535-6_12.

-
- [21] Daniel Gallin. *Intensional and Higher-Order Modal Logic: With Applications to Montague Semantics*. North-Holland Mathematics Studies: Volume 19, 1975.
- [22] Jeremy Gibbons and Nicolas Wu. “Folding Domain-Specific Languages: Deep and Shallow Embeddings (Functional Pearl)”. In: *SIGPLAN Not.* 49.9 (Aug. 2014), pp. 339–347. ISSN: 0362-1340. DOI: 10.1145/2692915.2628138.
- [23] Patrick Grim and Daniel Singer. “Computational Philosophy”. In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Spring 2020. Metaphysics Research Lab, Stanford University, 2020. URL: <https://plato.stanford.edu/archives/spr2020/entries/computational-philosophy/> (visited on 11/20/2021).
- [24] John Harrison. “HOL Light: An Overview”. In: *Theorem Proving in Higher Order Logics*. Ed. by Stefan Berghofer et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 60–66. ISBN: 978-3-642-03359-9. DOI: 10.1007/978-3-642-03359-9_4.
- [25] Brian Huffman and Ondrej Kunar. “Lifting and Transfer: A Modular Design for Quotients in Isabelle/HOL”. In: *Certified Programs and Proofs*. Ed. by Georges Gonthier and Michael Norrish. Cham: Springer International Publishing, 2013, pp. 131–146. ISBN: 978-3-319-03545-1. DOI: 10.1007/978-3-319-03545-1_9.
- [26] Maciej Janowicz et al. “Application of Automated Theorem-Proving to Philosophical Thought: Spinoza’s Ethics”. In: *Information Science and Applications 2017*. Ed. by Kuinam Kim and Nikolai Joukov. Singapore: Springer Singapore, 2017, pp. 512–518. ISBN: 978-981-10-4154-9. DOI: 10.1007/978-981-10-4154-9_59.
- [27] Daniel Kirchner. *Embedding of Abstract Object Theory in Isabelle/HOL*. Full sources. URL: <https://github.com/ekpyron/AOT/tree/dissertation> (visited on 12/14/2021).
- [28] Daniel Kirchner. *Embedding of Abstract Object Theory in Isabelle/HOL*. Variant counting equivalence classes of objects. URL: <https://github.com/ekpyron/AOT/tree/classes> (visited on 12/14/2021).
- [29] Daniel Kirchner. “Representation and Partial Automation of the Principia Logico-Metaphysica in Isabelle/HOL”. In: *Archive of Formal Proofs* (Sept. 2017). Formal proof development. ISSN: 2150-914x. URL: <http://isa-afp.org/entries/PLM.html> (visited on 11/20/2021).
- [30] Daniel Kirchner, Christoph Benzmüller, and Edward N. Zalta. “Computer Science and Metaphysics: A Cross-Fertilization”. In: *Open Philosophy* 2.1 (2019), pp. 230–251. DOI: 10.1515/opphil-2019-0015.
- [31] Daniel Kirchner, Christoph Benzmüller, and Edward N. Zalta. “Mechanizing Principia Logico-Metaphysica in functional type-theory”. In: *The Review of Symbolic Logic* 13.1 (2020), pp. 206–218. DOI: 10.1017/S1755020319000297.
- [32] Saul A. Kripke. “Semantical Considerations on Modal Logic”. In: *Acta Philosophica Fennica* 16 (1963), pp. 83–94.

- [33] Hannes Leitgeb, Uri Nodelman, and Edward N. Zalta. *A Defense of Logicism*. URL: <http://mally.stanford.edu/papers/logicism.pdf> (visited on 12/14/2021). In preparation.
- [34] W. McCune. “Prover9 and Mace4”. 2005–2010. URL: <http://www.cs.unm.edu/~mccune/prover9/> (visited on 11/20/2021).
- [35] Elliott Mendelson. “The Propositional Calculus”. In: *Introduction to Mathematical Logic*. Boston, MA: Springer US, 1987, pp. 10–40. ISBN: 978-1-4615-7288-6. DOI: 10.1007/978-1-4615-7288-6_2.
- [36] Christopher Menzel. “The Proper Treatment of Predication in Fine-Grained Intensional Logic”. In: *Philosophical Perspectives* 7 (1993), pp. 61–87. ISSN: 15208583, 17582245. DOI: 10.2307/2214116.
- [37] Leonardo de Moura and Nikolaj Bjørner. “Z3: An Efficient SMT Solver”. In: *Tools and Algorithms for the Construction and Analysis of Systems*. Ed. by C. R. Ramakrishnan and Jakob Rehof. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 337–340. ISBN: 978-3-540-78800-3. DOI: 10.1007/978-3-540-78800-3_24.
- [38] Tobias Nipkow, Lawrence C. Paulson, and Makarius Wenzel. *A Proof Assistant for Higher-Order Logic*. URL: <https://isabelle.in.tum.de/website-Isabelle2021-1/dist/Isabelle2021-1/doc/tutorial.pdf> (visited on 12/14/2021).
- [39] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*. Vol. 2283. LNCS. Springer, 2002. DOI: 10.1007/3-540-45949-9.
- [40] Steven Obua. “Partizan Games in Isabelle/HOLZF”. In: *Theoretical Aspects of Computing - ICTAC 2006*. Ed. by Kamel Barkaoui, Ana Cavalcanti, and Antonio Cerone. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 272–286. ISBN: 978-3-540-48816-3. DOI: 10.1007/11921240_19.
- [41] Paul E. Oppenheimer and Edward N. Zalta. “A Computationally-Discovered Simplification of the Ontological Argument”. In: *Australasian Journal of Philosophy* 89.2 (2011), pp. 333–349. DOI: 10.1080/00048401003674482.
- [42] Paul E. Oppenheimer and Edward N. Zalta. “On the Logic of the Ontological Argument”. In: *Philosophical Perspectives* 5 (1991), pp. 509–529.
- [43] Paul E. Oppenheimer and Edward N. Zalta. “Relations Versus Functions at the Foundations of Logic: Type-Theoretic Considerations”. In: *Journal of Logic and Computation* 21 (2011), pp. 351–374. DOI: 10.1093/logcom/exq017.
- [44] Lawrence Paulson. “Three Years of Experience with Sledgehammer, a Practical Link between Automatic and Interactive Theorem Provers”. In: *PAAR-2010: Proceedings of the 2nd Workshop on Practical Aspects of Automated Reasoning*. Ed. by Renate A. Schmidt, Stephan Schulz, and Boris Konev. Vol. 9. EPiC Series in Computing. EasyChair, 2012, pp. 1–10. DOI: 10.29007/tnfd.
- [45] Lawrence C. Paulson. *Isabelle’s Logics*. URL: <https://isabelle.in.tum.de/website-Isabelle2021-1/dist/Isabelle2021-1/doc/logics.pdf> (visited on 12/14/2021).

-
- [46] Francis J. Pelletier and Edward N. Zalta. “How to Say Goodbye to the Third Man”. In: *Noûs* 34.2 (2000), pp. 165–202. DOI: 10.1111/0029-4624.00207.
- [47] William Joseph Rapaport. “Intentionality and the Structure of Existence”. PhD thesis. Indiana University, 1976. URL: <https://cse.buffalo.edu/~rapaport/Papers/rapaport1976-PhD-Diss.pdf> (visited on 11/20/2021).
- [48] Stephan Schulz, Simon Cruanes, and Petar Vukmirovi. “Faster, Higher, Stronger: E 2.3”. In: *Proc. of the 27th CADE, Natal, Brasil*. Ed. by Pascal Fontaine. LNAI 11716. Springer, 2019, pp. 495–507. DOI: 10.1007/978-3-030-29436-6_29.
- [49] Herbert A. Simon. *Models of My Life*. Cambridge: MIT Press, 1996. ISBN: 978-0-262-69185-7.
- [50] The Coq Development Team. *The Coq Proof Assistant*. Version 8.14. Oct. 2021. DOI: 10.5281/zenodo.5704840.
- [51] Christoph Weidenbach. “Combining Superposition, Sorts and Splitting”. In: *Handbook of Automated Reasoning*. Ed. by Alan Robinson and Andrei Voronkov. Handbook of Automated Reasoning. Amsterdam: North-Holland, 2001, pp. 1965–2013. ISBN: 978-0-444-50813-3. DOI: 10.1016/B978-044450813-3/50029-1.
- [52] Makarius Wenzel. “Interaction with Formal Mathematical Documents in Isabelle/PIDE”. In: *Intelligent Computer Mathematics*. Ed. by Cezary Kaliszyk et al. Cham: Springer International Publishing, 2019, pp. 1–15. ISBN: 978-3-030-23250-4. DOI: 10.1007/978-3-030-23250-4_1.
- [53] Makarius Wenzel. *The Isabelle System Manual*. URL: <https://isabelle.in.tum.de/website-Isabelle2021-1/dist/Isabelle2021-1/doc/system.pdf> (visited on 12/14/2021).
- [54] Makarius Wenzel. *The Isabelle/Isar Implementation*. URL: <https://isabelle.in.tum.de/website-Isabelle2021-1/dist/Isabelle2021-1/doc/implementation.pdf> (visited on 12/14/2021).
- [55] Makarius Wenzel. *The Isabelle/Isar Reference Manual*. URL: <https://isabelle.in.tum.de/website-Isabelle2021-1/dist/Isabelle2021-1/doc/isar-ref.pdf> (visited on 12/14/2021).
- [56] Edward N. Zalta. *Abstract Objects: An Introduction to Axiomatic Metaphysics*. Synthese Library. Springer, 1983. ISBN: 9789027714749. DOI: 10.1007/978-94-009-6980-3.
- [57] Edward N. Zalta. “Frege’s Theorem and Foundations for Arithmetic”. In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Fall 2021. Metaphysics Research Lab, Stanford University, 2021. URL: <https://plato.stanford.edu/archives/fall2021/entries/frege-theorem/> (visited on 11/29/2021).
- [58] Edward N. Zalta. *Further Explanation of the Objectives of the Theory*. URL: <http://mally.stanford.edu/objectives.html> (visited on 12/14/2021).
- [59] Edward N. Zalta. *Intensional Logic and the Metaphysics of Intentionality*. A Bradford book. MIT Press, 1988. ISBN: 9780262240277.

- [60] Edward N. Zalta. “Natural Numbers and Natural Cardinals as Abstract Objects: A Partial Reconstruction of Frege’s Grundgesetze in Object Theory”. In: *Journal of Philosophical Logic* 28.6 (1999), pp. 619–660. DOI: 10.1023/A:1004330128910.
- [61] Edward N. Zalta. “Neo-Logicism? An Ontological Reduction of Mathematics to Metaphysics”. In: *Erkenntnis* 53.1 (Sept. 2000), pp. 219–265. ISSN: 1572-8420. DOI: 10.1023/A:1005614102033.
- [62] Edward N. Zalta. *Principia Logico-Metaphysica*. [Draft/Excerpt; continuously updated current version]. URL: <http://mally.stanford.edu/principia.pdf> (visited on 11/20/2021).
- [63] Edward N. Zalta. *Principia Logico-Metaphysica*. [Draft/Excerpt; dated October 13, 2021]. URL: <http://mally.stanford.edu/principia-2021-10-13.pdf> (visited on 11/20/2021).
- [64] Edward N. Zalta. *Principia Logico-Metaphysica*. [Draft/Excerpt; dated October 28, 2016]. URL: <https://mally.stanford.edu/principia-2016-10-28.pdf> (visited on 11/20/2021).
- [65] Edward N. Zalta. “Twenty-Five Basic Theorems in Situation and World Theory”. In: *Journal of Philosophical Logic* 22.4 (1993), pp. 385–428. DOI: 10.1007/BF01052533.

Zusammenfassung der Ergebnisse

Wir präsentieren die Implementierung einer metaphysischen Grundlagentheorie in einem automatischen Theorembeweiser mit Hilfe einer Erweiterung des Prinzips von *shallow semantic embeddings* (SSEs) in klassischer Logik höherer Stufe. Insbesondere kommen wir zu folgenden Ergebnissen:

- SSEs sind skalierbar und können nicht nur für die Analyse einzelner Argumente verwendet werden, sondern können auch auf komplette metaphysische Theorien angewendet werden, und deren Axiome und Deduktionssysteme präzise darstellen.
- Eine solche Implementierung ist kein rein technisches Unterfangen, sondern kann zu einem fruchtbaren Austausch führen, der in unserem Fall einerseits zu signifikanten Verbesserungen der analysierten Theorie geführt hat und andererseits neues Licht auf die technischen Möglichkeiten und Einschränkungen von SSEs werfen konnte.
- Es ist nicht nur möglich, die Logik eines komplexen Zielsystems technisch zu reproduzieren, sondern auch eine nahezu transparente Darstellung von Syntax und Beweisführung im Zielsystem zu erreichen, was einen effizienten und einfachen Austausch von Ergebnissen zwischen traditioneller Beweisführung von Hand und der computerbasierten Implementierung ermöglicht.
- Die Automatisierungsverfahren von Isabelle/HOL bleiben dabei erhalten und können dazu verwendet werden, Beweise im Zielsystem zu konstruieren, die allein den Deduktionsregeln des Zielsystems folgen. Dadurch erreichen wir effektiv einen dedizierten Theorembeweiser für unser Zielsystem auf der Grundlage einer verifizierbar konsistenten metalogischen Konstruktion.
- Unser Zielsystem *Abstract Object Theory* (AOT) selbst kann seinem Anspruch gerecht werden, eine philosophisch fundierte Konstruktion und Analyse der natürlichen Zahlen bieten zu können. Insbesondere können wir bestätigen, dass Frege's Konstruktion der natürlichen Zahlen in AOT konsistent reproduziert werden kann. Darüber hinaus konnten wir signifikant zur Weiterentwicklung der Konstruktion beitragen und können zusätzliche Erkenntnisse über die benötigten Axiome und über mögliche Varianten der Konstruktion beisteuern.

Interessanterweise stützen unsere Ergebnisse einerseits die Verwendung von klassischer Logik höherer Stufe als universale Metalogik, nachdem wir demonstrieren konnten, dass mit Hilfe der SSE Methode selbst herausfordernde logische Fundamentaltheorien präzise einbettbar sind, während wir andererseits die Position unseres Zielsystems AOT als metaphysische Fundamentaltheorie stärken, nachdem wir bestätigen können, dass es eine philosophisch fundierte Konstruktion mathematischer Objekte erlaubt. In diesem Zusammenhang bilden die Implementierung und Analyse des vollen typen-theoretischen Systems höherer Ordnung von AOT mit Hilfe der SSE Methode, sowie die Analyse der relativen Stärke dieses Systems im Vergleich zu HOL und ZF faszinierende Themen für zukünftige Forschungsarbeit.