Freie Universität Berlin

Free Higher-Order Logic

Notion, Definition and Embedding in HOL

Master's Thesis

March 10, 2020

Name:	Irina Makarenko
Matriculation number:	4675619
E-mail address:	irina.makarenko@fu-berlin.de
Department:	Mathematics and Computer Science
Institute:	Institute of Computer Science
Study program:	Master of Science in Computer Science
First supervisor:	Prof. Dr. Christoph Benzmüller
Second supervisor:	Dr. Alexander Steen

Abstract

Free logics are a family of logics that are free of any existential assumptions. This family can roughly be divided into positive, negative, neutral and supervaluational free logic whose semantics differ in the way how nondenoting terms are treated. While there has been remarkable work done concerning the definition of free first-order logic, free higher-order logic has not been addressed thoroughly so far. The purpose of this thesis is, firstly, to give a notion and definition of free higher-order logic based on simple type theory and, secondly, to propose faithful shallow semantical embeddings of free higher-order logic into classical higher-order logic found on this definition. Such embeddings can then effectively be utilized to enable the application of powerful state-of-the-art higher-order interactive and automated theorem provers for the formalization and verification and also the further development of increasingly important free logical theories.

Statutory Declaration

I hereby declare that I have written the present thesis independently, without enlisting any external assistance, and only using the specified aids. I additionally assert that this thesis has not been part of another examination process.

Berlin, 10 March 2020 .....

Irina Makarenko

Contents

1 Introduction

Nonexistence of objects may take many forms. For instance, there is `Pegasus`, a fictitious divine horse from mythology. Though obviously not existent in the real world, everyone has a clear idea of its general appearance: white hair coat, four legs, and two wings. On the other hand, we have `Vetnas`, a distant, possibly habitable planet yet unknown to us. Since it has not been discovered or explored so far, it has no implicit properties and therefore likewise fails to refer to some concrete object in the actual world. Finding suitable interpretations for nonexistent objects in terms of their meaning and handling has a long philosophical tradition, for example with Łukasiewicz and Kleene. Most commonly these objects are interpreted as possible, unknown, undefined, half-true, irrelevant, or inconsistent (cf. Ciucci and Dubois 2013). Free logic is a logical framework that tries to accommodate all these different types of nonexistence. It tries to provide answers to questions including but not limited to: What status do nonexistent objects have? What do they refer to? Are all nonexistent objects the same? Can terms involving them have truth values? With respect of higher-order logic, reasoning with nonexistent objects leads inevitably to partiality of predicates and functions when arguments or results are nonexisting, and thus to our motivation.

1.1 Motivation

Partiality is a key challenge not only for philosophical and mathematical concerns but also for computational approaches to artificial intelligence and natural language.[1] For that reason, free logical theories are becoming more and more important and so is computer-assisted reasoning with them. However, most mature interactive and automated theorem provers are limited to classical logics as first- or higher-order logic where only total functions are supported natively. Instead of investing time and effort in the development of new automatic provers, a highly promising approach is providing a *shallow semantical embedding* (SSE), that is, a translation, of free logic into higher-order logic to profit from the strength of long-standing reasoning systems for establishing the correctness of theories in propositional and quantified free logic.[2] Very recently, Tiemens, Scott, Benzmüller, and Benda (2019) showed successfully how this approach works well in the context of partial functions within category theory and modeloids. But free logic has more than one facet and one use case, and hence it would be beneficial to generally investigate how the individual treatments of nondenoting terms in free logic can be formalized and embedded into higher-order logic. The aim of this thesis is to provide a way for universal reasoning within free logic by utilizing already existing automated proving tools. The first step towards this is to determine a syntax and semantics of *free higher-order logic* (FHOL) suitable for this purpose and to form faithful embeddings for each of free logic's main varieties from these definitions. Then, these embeddings are automated and tested by encoding them into an interactive proof assistant based on higher-order logic.

[1] See the publications of Feferman (1992), Gumb and Lambert (1997), and Gumb (2001) for rich discussions of possible applications of free logic to computation.

[2] Bove, Krauss, and Sozeau (2016) also explored ways in which partial functions can be represented in modern theorem provers, but were mainly interested in the partiality that arises from non well-founded recursion rather than the partiality which results from a function not being defined on some arguments.

Furthermore, although the primary tactic relied on in this thesis is the shallow semantical embedding approach, the specified embeddings will be compared against another definition of free higher-order logic after Bacon, Hawthorne, and Uzquiano (2016) to see if helpful conclusions for the intention of this thesis can be drawn from the comparison. Now, before we start our study with the theoretical basics as the formal language of *classical higher-order logic* (HOL), we shall introduce the proof assistant of our choice for the encoding, which is Isabelle/HOL.

1.2 Isabelle/HOL

Isabelle/HOL (cf. Nipkow, Paulson, and Wenzel 2002) is a leading interactive proof assistant based on polymorphic higher-order logic augmented with axiomatic type classes that offers a broad module system as well as Isar structured proofs[3] and an user interface with notation support. For interactive proof construction, various state-of-the-art first-order and higher-order automated theorem provers and model finders are combined within Isabelle/HOL. The integrated tools include, among others, the model finder Nitpick (cf. Blanchette and Nipkow 2010) and the meta-prover Sledgehammer (cf. Blanchette, Böhme, and Paulson 2013; Paulson and Blanchette 2015), which invokes third-party resolution provers and SMT solvers like the equational reasoner Simp (cf. Nipkow 1989), the classical reasoners Auto, Force, and Fast (cf. Paulson 1994) and the untyped tableau prover Blast (cf. Paulson 1999). Prominent also remotely reachable higher-order provers are Satallax (cf. Brown 2012) and Leo-III (cf. Steen and Benzmüller 2018a,b). Although there are viable alternatives with Coq and Lean, Isabelle/HOL is arguably the most advanced language for large scale formalizations effectively used in diverse abstract problem representations. With its overall good reputation, it qualifies as the best available tool for reaching the goal of this thesis (cf. Paulson and Blanchette 2015).

[3] Isar (cf. Wenzel 1999) is a proof language used in Isabelle/HOL.

2 Formal Language of Classical Higher-Order Logic

The *simple theory of types* (STT), referring to a logic presented in 1940 by Alonzo Church, is a theory that builds classical higher-order logic on top of the simply typed λ-calculus. Church's original definitions as generalized by Henkin (1950) to *extensional type theory* (ExTT), the logical basis of most automated theorem proving systems for higher-order logic (cf. Steen 2020: 8), are rephrased below with reference to Benzmüller and Andrews (2019).

2.1 Syntax

The main components of Church's type theory are types and terms; more precisely, typed terms. The set of types is freely generated from a set of two base types $\{o, i\}$ and the right associative function type constructor $\to$.[4] Intuitively, o is the type of standard truth values and i is the type of individuals.

Definition 1. The set $\mathcal{T}$ of simple types is given by the following abstract grammar:

$$\alpha, \beta := \quad o \mid i \mid (\alpha \to \beta)^5 .$$

Definition 2. The subset $\mathcal{T}_o \subset \mathcal{T}$ of simple types of type o is given by the following abstract grammar:

$$\beta := \quad o \mid (\alpha \to \beta)$$

with $\alpha \in \mathcal{T}$.

The nonempty subset $\mathcal{T}_i \subset \mathcal{T}$ of simple types of type i is defined accordingly. Obviously, $\mathcal{T}_o \cup \mathcal{T}_i = \mathcal{T}$.

Simply typed terms of higher-order logic are finite sequences of symbols and constructed by exploiting logical constants for equality, negation, disjunction, universal quantifier and definite description, the improper symbols $(,)$, . and λ together with variables denoted by lower case letters and nonlogical constants denoted by capital letters.[6]

[4] For the sake of completeness it shall be mentioned that there is no serious restriction to a two-valued base set (Benzmüller and Miller 2014). Extending the given definitions to a larger set of base types, to many sortedness, is straightforward (cf. Farmer 2008: 282).

[5] In his paper from 1940, Church used the more compact notation $\beta\alpha$ for $\alpha \to \beta$, which, since given in reverse order, associates to the left.

[6] The set of primitive logical constants could be a much smaller one, e.g., equality is known to be sufficient in order to define all logical constants of classical higher-order logic apart from the description operator (Henkin 1963, 1975; Benzmüller and Andrews 2019: 1.4, and further references therein).

Definition 3. Terms of HOL are defined based on the following formation rules:

$$\begin{aligned} s,t \;:=\;& P_\alpha \mid x_\alpha \mid ((=_{\alpha\to\alpha\to o}\, s_\alpha)_{\alpha\to o}\, t_\alpha)_o \mid (\neg_{o\to o}\, s_o)_o \mid ((\vee_{o\to o\to o} s_o)\, t_o)_o \mid \\ & (\forall_{(\alpha\to o)\to o}(\lambda x_\alpha.\, s_o)_{\alpha\to o})_o \mid (\iota_{(\alpha\to o)\to\alpha}(\lambda x_\alpha.\, s_o)_{\alpha\to o})_\alpha \mid \\ & (s_{\alpha\to\beta}\, t_\alpha)_\beta \mid (\lambda x_\alpha.\, s_\beta)_{\alpha\to\beta} \end{aligned}$$

with $\alpha, \beta \in \mathcal{T}$.

Additional logical constants can be introduced as follows:

$$\begin{aligned} \wedge_{o\to o\to o} &:= \lambda x_o.\, \lambda y_o.\, \neg\,(\neg x \vee \neg y) \\ \to_{o\to o\to o} &:= \lambda x_o.\, \lambda y_o.\, \neg x \vee y \\ \leftrightarrow_{o\to o\to o} &:= \lambda x_o.\, \lambda y_o.\, (x \to y) \wedge (y \to x) \\ \exists_{(\alpha\to o)\to o} &:= \lambda p_{\alpha\to o}.\, \neg\forall(\lambda x_\alpha.\, \neg(p\, x)) \\ \top_o &:= \lambda x_\alpha.\, x \;=\; \lambda x_\alpha.\, x \end{aligned}$$

with $\alpha \in \mathcal{T}$.

When equality is not considered as primitive in the language, it can be represented by Leibniz' formula encoding that two objects of type $\alpha \in \mathcal{T}$ are the same if they have the same properties (Benzmüller, Brown, and Kohlhase 2004):

$$=^L_{\alpha\to\alpha\to o} \;:=\; \lambda x_\alpha.\, \lambda y_\alpha.\; \forall p_{\alpha\to o}.\, p\, x \to p\, y\,.$$

The definite description $\iota(\lambda x_\alpha.\, s_o)$ denotes the unique object x of type $\alpha \in \mathcal{T}$ satisfying s_o if it exists and a canonical error value of type α otherwise (cf. Carpenter 1997: 77). It offers one the possibility to define an if-then-else operator as, e.g, Farmer (cf. 2008: 273) did:

$$ite_{o\to\alpha\to\alpha\to\alpha} \;:=\; \lambda s_o.\, \lambda x_\alpha.\, \lambda y_\alpha.\; \iota(\lambda z_\alpha.\, (s \to z = x) \wedge (\neg s \to z = y))\,.$$

The type of each term is given as a subscript and is an element of $\mathcal{T}$ unless oppositely stated. Type information may be omitted if clear from the context. Terms of type o are *formulas*, nonformula terms of type $\alpha \in \mathcal{T}_o$ are called *predicates*. Conversely, terms of type $\alpha \in \mathcal{T}_i$ except i itself serve as *functions*. Formulas whose leftmost nonparenthesis symbol is either equality or some nonlogical constant or variable are called *atomic formulas* (cf. Lipton and Nieva 2007: 3). V_α is the countably infinite set of all variable symbols of type α and C_α is the countable set of all nonlogical constant symbols of type α.

Definition 4. A variable x is *bound* in a term s if it occurs in the scope of a quantifier in s. x is *free* in s when it is not bound in s.

Any term with no free variable is a *closed term*. Any closed formula is a *sentence*.

Definition 5. The substitution of a term t_α for all free occurrences of variable x_α in a term s_β is denoted by $s\,[x \to t]$.

Substitutions are capture-avoiding, meaning that a term is α-converted whenever needed to prevent binding free variables. α-conversion as well as β- and η-conversions are defined as usual:

$$\begin{aligned} \lambda x_\alpha.\, s_\beta &= \lambda y_\alpha.\, (s\,[x \to y]) && \text{where } y \text{ does not occur free in } s && (\alpha)\\ (\lambda x_\alpha.\, s_\beta)\, t_\alpha &= s\,[x \to t] && && (\beta)\\ \lambda x_\alpha.\, (s_\beta\, x_\alpha) &= s && \text{where } x \text{ does not occur free in } s. && (\eta) \end{aligned}$$

For each binary operator *op* with prefix notation $((op\, s)\, t)$ we may fall back to its infix notation $(s\, op\, t)$ to improve readability. Likewise, the binder notation $\{\forall, \iota\}(x.\, s)$ may be used as shorthand for $\{\forall, \iota\}(\lambda x.\, s)$. In the remainder of this thesis, a matching pair of parentheses in a type or term may be dropped when they are not necessary, assuming that, in addition to the generally known rules, $(s\, t)$, the application, and $(\lambda x.\, s)$, function abstraction, are left and right associative, respectively, and that application has a smaller scope than abstraction.

2.2 Semantics

Semantics assigns meaning to syntactically complex terms of a language. Classical higher-order logic has a well-defined and thoroughly documented semantics (cf. Benzmüller, Brown, et al. 2004) which is summarized below.

Definition 6. A *frame* D is a set $\{D_\alpha : \alpha \in \mathcal{T}\}$ of nonempty sets (formally *domains*) D_α such that D_i is chosen freely, $D_o = \{\mathrm{T}, \mathrm{F}\}$ where $\mathrm{T} \neq \mathrm{F}$ and T represents truth and F represents falsehood, and $D_{\alpha\to\beta}$ is the set of all total functions from domain D_α to codomain D_β.

Definition 7. A *standard model* is a tuple $M = \langle D, I \rangle$ where D is a frame and I is a family of typed interpretation functions, i.e., $I = \{I_\alpha : \alpha \in \mathcal{T}\}$. Each *interpretation function* I_α maps constants of type α to appropriate elements of D_α.

The logical constants $=$, $\neg$, $\vee$, $\forall$ and ι always receive the same meaning of the standard kind by every interpretation:

$$\begin{aligned} I(=_{\alpha\to\alpha\to o}) &:= id^{7} && \in D_{\alpha\to\alpha\to o} && \text{s.t. for all } d, d' \in D_\alpha,\\ & && && \quad id(d, d') = \mathrm{T} \text{ iff } d \text{ is identical to } d'\\ I(\neg_{o\to o}) &:= not && \in D_{o\to o} && \text{s.t. } not(\mathrm{T}) = \mathrm{F} \text{ and } not(\mathrm{F}) = \mathrm{T}\\ I(\vee_{o\to o\to o}) &:= or && \in D_{o\to o\to o} && \text{s.t. } or(v_1, v_2) = \mathrm{T} \text{ iff } v_1 = \mathrm{T} \text{ or } v_2 = \mathrm{T} \end{aligned}$$

[7] Note that since equality is taken as primitive for all types, it is ensured that the corresponding domains contain the respective identity relations. Andrews (1972a) demonstrated the importance of including these relations (cf. Benzmüller, Brown, et al. 2004: 1046–1048).

$$I(\forall_{(\alpha\to o)\to o}) := allq \in D_{(\alpha\to o)\to o} \quad \text{s.t. for all } f \in D_{\alpha\to o},\ allq(f) = \mathrm{T} \text{ iff } f(d) = \mathrm{T} \text{ for all } d \in D_\alpha$$

$$I(\iota_{(\alpha\to o)\to\alpha}) := desc \in D_{(\alpha\to o)\to\alpha} \quad \text{s.t. for all } f \in D_{\alpha\to o},\ desc(f) = d \in D_\alpha \text{ if } f(d) = \mathrm{T} \text{ and for all } d' \in D_\alpha\text{: if } f(d') = \mathrm{T}, \text{ then } d' = d, \text{ otherwise } desc(f) = e, \text{ where } e \text{ is an arbitrary object in } D_\alpha$$

with $\alpha \in \mathcal{T}$.

Definition 8. The function $g_\alpha : V_\alpha \to D_\alpha$ is a *variable assignment* mapping variables in V_α to corresponding elements in D_α. Thus, $g = \{g_\alpha : \alpha \in \mathcal{T}\}$ is a family of typed variable assignments. $g\,[x \to d]$ denotes the assignment that is identical to g, except for variable x_α, which is now mapped to d_α:

$$g\,[x \to d]\,(x) = d$$
$$\text{and} \quad g\,[x \to d]\,(y) = g\,(y) \in D_\alpha \quad \text{for all } y \neq x\,.$$

The value $[\![\, s_\alpha \,]\!]^{M,g}$ of a HOL term s_α in a standard model M under assignment g is an object $d \in D_\alpha$ and defined in the following way:

$$[\![\, P_\alpha \,]\!]^{M,g} := I(P_\alpha)$$

$$[\![\, x_\alpha \,]\!]^{M,g} := g(x_\alpha)$$

$$[\![\, (s_{\alpha\to\beta}\, t_\alpha)_\beta \,]\!]^{M,g} := [\![\, s_{\alpha\to\beta} \,]\!]^{M,g}\,([\![\, t_\alpha \,]\!]^{M,g})$$

$$[\![\, (\lambda x_\alpha .\ s_\beta)_{\alpha\to\beta} \,]\!]^{M,g} := \text{the function } f \text{ from } D_\alpha \text{ into } D_\beta \text{ s.t. for all } d \in D_\alpha\text{: } f(d) = [\![\, s_\beta \,]\!]^{M,g[x\to d]}$$

with $\alpha, \beta \in \mathcal{T}$.

Definition 9. A formula s is *true* in a standard model M under assignment g if and only if $[\![\, s_o \,]\!]^{M,g} = \mathrm{T}$, also denoted by $M, g \vDash s$. A formula s is called *valid in* M, denoted by $M \vDash s$, if and only if $M, g \vDash s$ for all assignments g. Moreover, a formula s is called *(generally) valid*, denoted by $\vDash s_o$, if and only if s is valid in all standard models M.

By immediate consequence of Gödel's incompleteness theorem from 1931, Church's type theory with respect to the ordinary semantics based on standard models is incomplete.[8] However, Henkin (1950) introduced a broader notion of a model in which the function domains contain enough but not necessarily all functions:

[8] Farmer (2008) gave a detailed account of this proof.

Definition 10. In a standard model a domain $D_{\alpha\to\beta}$ is defined as the set of all total functions from D_α to D_β. In a *Henkin model* (or *general model*) $D_{\alpha\to\beta}$ is some nonempty set of total functions, $D_{\alpha\to\beta} \subseteq \{ f \mid f : D_\alpha \to D_\beta\}$, containing at least sufficiently many of them that the valuation function remains total.

Henkin's weaker form of the standard semantics is surprisingly strong: For his generalized semantics sound and complete proof calculi exist (Henkin 1950; Andrews 1972a,b). Any standard model is obviously also a Henkin model. Hence, any formula that is valid in all Henkin models must be valid in all standard models as well. Therefore, the semantics employed in this thesis are Henkin's general models. For truth, validity and general validity in a Henkin model, definition 9 is adapted in the obvious way.

3 Free Higher-Order Logic

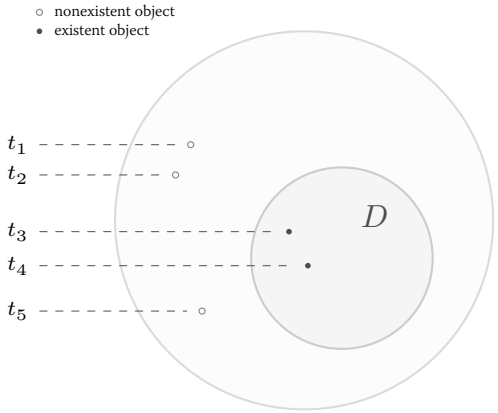

Figure 1: Schema of a domain where each term denotes either an existent or a nonexistent object

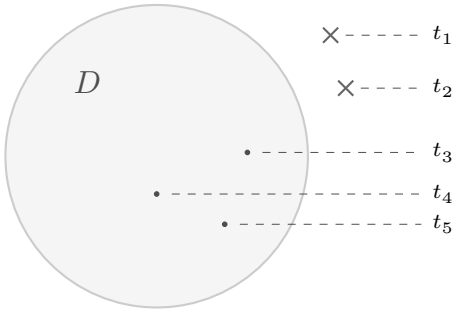

Figure 2: Schema of a domain where each term denotes either an existent object or not at all

Free logic, a term coined by Lambert (1960), refers to a family of logics that are free of existential presuppositions in general and with respect to the denotation of terms in specific. Terms of free logic may denote existent[9] objects, but are not necessarily required to do so. Quantification is treated exactly as in classical logic, which means that quantifiers range only over the existing objects.[10] As illustrated in fig. 1 and 2, terms fail to refer to objects in the range of the quantifiers either when they refer to objects outside the quantification domain D, or when they do not refer at all (cf. Lehmann 2002: 218–219). Both approaches have their raison d'être considering that there are indeed two different views on nondenoting terms: While a philosophical logician is more interested in describing relations between denoting and nondenoting terms and therefore prefers the approach seen in fig. 1, the mathematician's main intention is to reason about partial functions where nondenotation has no particular significance (cf. fig. 2). Despite general criticism[11], as the approach with two domains is capable of mimicking the other one[12], it will be pursued for the remainder of this thesis.

[9] In the thesis at hand, the terms existent/existing and defined are used interchangeably even though a differentiation is advisable. The same applies to the terms nonexistent/nonexisting and undefined.

[10] To say, that the quantifiers are treated exactly as in classical logic, means, essentially, that the universal quantifier $\forall$ reads: 'every existing object', and the existential quantifier $\exists$ reads: 'there exists an object'.

[11] The approach implies, in a way, a Meinong-inspired view, which was considered critical by, for example, Lambert (2001b: 262–263), Antonelli (2000: 278, 2007: 7), and Bacon (2013: 9). Paśniczek (2001) also questioned whether Meinongian logic can be free. However, a slightly modified approach, exactly as offered in this thesis, was supported by Bencivenga (1986: 415) and also Bacon (2013). Antonelli (2000, 2007) urged his proto-semantics instead.

[12] A partial function f can be simulated by a total function that coincides with f where f is defined and maps to some arbitrary nonexisting object otherwise. All nondenoting terms can be mapped to the very same object (e.g., as Scott [1967] proposed), but do not have to be.

Nonetheless, we will continue to use the term 'nondenoting', meaning 'denoting some nonexistent object'. The approach selected is also known as the *dual domain approach* where both domains are labeled. The domain of nonexisting objects is hereby called D, and the name of the domain of quantification changes to E. Interrelation of both domains can be described in two different ways: as a dual domain with two disjoint sets or as an inner-outer dual domain where D includes E. The first suggestion is exemplified in fig. 3 and traces back to Leblanc and Thomason (1968) as well as Meyer and Lambert (1968). Its variation, depicted in fig. 4, was proposed by Cocchiarella (1966) (cf. Morscher and Simons 2001: 25). Aiming for short, readable definitions the latter proposal fits best for our needs, which is why we choose it for the further course. Restating the upcoming definitions for dual domains with disjoint sets is a fairly easy thing to do and left to the ambitious reader.

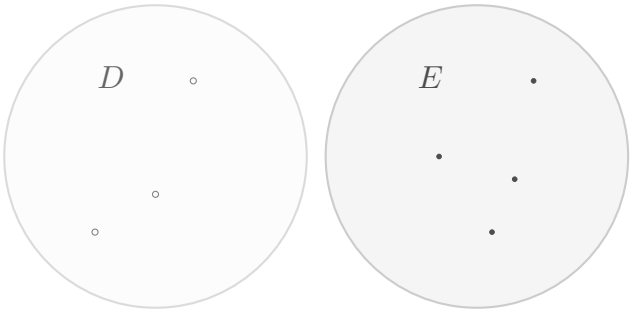

Figure 3: Example of the dual domain approach with two disjoint domains

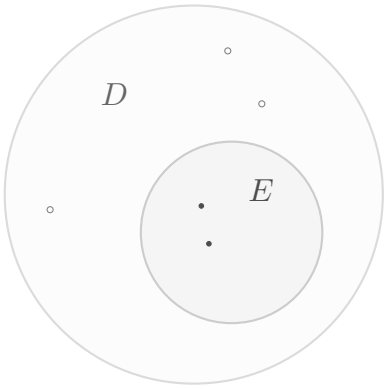

Figure 4: Example of the inner-outer dual domain approach

Most literature cited in this thesis deals with free first-order or propositional logic. In the following, the concepts and results therein are reproduced and transferred to free higher-order logic as accurately and precisely as possible. Throughout this section, a broad overview of all varieties of free logic and their individual characteristics revised to be of higher order is given. Upon that, we will explore how free logics and logics that have the property to be inclusive are connected. For motivational reasons, we shall begin with a detailed look at a recently published paper in which free higher-order logic plays a special role.

3.1 On Modelling Prior's Theorem

In their 2016 published paper 'Higher-Order Free Logic and the Prior-Kaplan Paradox' the authors Bacon, Hawthorne, and Uzquiano discovered that universal instantiation, or, better, the rejection of it, is key to blocking certain paradoxes inherent in higher-order logics. Logics without existential assumptions, free logics, just naturally reject the principle of universal instantiation. Basically, universal instantiation incorporates that from 'every x in E satisfies s' we may infer 'some y satisfies s'. This rule, written out as

$$\forall x.\, s \;\rightarrow\; s\,[x \rightarrow y], \tag{sUI}$$

valid in classical logic where $D = E$, is not valid in free logic. Consider, for example, the case that y does not denote an object in E. Then, even if every object in E satisfies s, $s\,[x \rightarrow y]$ could still be false. Related thereto is the principle of existential generalization, which is equally valid in classical logic but invalid in free logic. Existential generalization refers to the principle that, from a formula s containing y, we may infer, that there is some object in E that satisfies s:[13]

$$s\,[x \rightarrow y] \;\rightarrow\; \exists x_{\alpha}.\, s. \tag{sEG}$$

If y does not denote an object in the existence domain, then $s\,[x \rightarrow y]$ being true does not imply that there exists an object in E that satisfies s. In fact, sUI and sEG are not provable in free logic unless an explicit premise is added to make sure the term $s\,[x \rightarrow y]$ is satisfied by an existing object y and not just any object in D. Existence of y can be surveyed by, e.g., introducing a predicate $E!$. Thus, we can redefine universal instantiation and existential generalization to be valid in free logic as follows:

$$\forall x.\, s \wedge E!\, y \;\rightarrow\; s\,[x \rightarrow y], \tag{wUI}$$

$$s\,[x \rightarrow y] \wedge E!\, y \;\rightarrow\; \exists x.\, s \tag{wEG}$$

(Nolt 2018: 1.2).

Arthur Prior (1961), coinciding with David Kaplan (1995), showed that paradoxes can arise quickly in particular philosophical theories that include both sets and propositions. Bacon et al. (2016: 4) amended, that these paradoxes are derivable by accepting sUI and are invalid with respect to wUI. The family of paradoxes considered by Bacon et al. is represented by what we will call Prior's theorem in this thesis. Prior's theorem is an intensional version of the Liar paradox[14] and states:

$$Q\,\forall p.\, (Q\,p \rightarrow \neg p) \;\rightarrow\; \exists p.\, (Q\,p \wedge p) \wedge \exists p.\, (Q\,p \wedge \neg p).$$

Reading $Q\,p$ as, for instance, 'Kaplan believes that p', Prior's theorem says that if Kaplan believes that everything that he believes is false, then he believes something true and also believes something false (Bacon et al. 2016: 4–6) – a logical self-contradiction[15] that should be resolved, and will be as we will see in the further course.

[13] It is remarkable, that while the principle of universal instantiation is denied in all variants of free logic, sEG for some s, e.g., a formula with its leftmost symbol being a nonlogical constant, is valid in special settings called negative and neutral free logic (cf. Posy 2007: 647–648, see also Walters [2014: 25–26] for an excessive discussion). We will have a closer look at this in section 4.

[14] See, e.g., the notes of Beall, Glanzberg, and Ripley (2019) for this famous paradox.

[15] To be more precise, the theorem becomes a puzzle in the presence of 'opaque contexts' created by intensional attitudes like thinking, fearing, hoping, and so forth (Bacon 2019: 11). For further elaborations on this, see the reference.

For the rest of the section, we focus on the language fragment outlined by Bacon et al. (2016) aiming to invalidate Prior's theorem. The logic in question is a quantified propositional logic with logical constants $\top$, $\bot$, $=$, $\neg$, $\vee$, $\forall$ and $\Box$ in addition to an arbitrary unary logical connective Q. The modal operator $\Box$ stands for necessity, while $\Diamond$ as its counterpart expresses possibility. $\Diamond s$ abbreviates as $\neg\Box\neg s$. The semantics of interest are Kripke models (1963) which involve a set of elements, called worlds, W, and an accessibility relation $R \subseteq W \times W$ to interpret modalities, i.e., necessities and possibilities. Furthermore, we have a domain $D(w) \subseteq \mathcal{P}W$ – with $\mathcal{P}W$ being the powerset of W – defining the range of the quantifiers at each world, and an extension for Q at each $w \in W$, a function, written $|Q|$, which maps a world to a set of sets of worlds. Kripke models can be depicted as directed graphs. See fig. 5 for an example where $W = \{x, y\}$, $R = \{(x, x), (x, y)\}$, $|Q|(x) = \{\{x, y\}\}$ and $|Q|(y) = \{\{y\}, \varnothing\}$. In terms of Kripke semantics, the modal system S5 is characterized by models where the accessibility relation is an equivalence relation. In what follows we shall only consider S5 models.

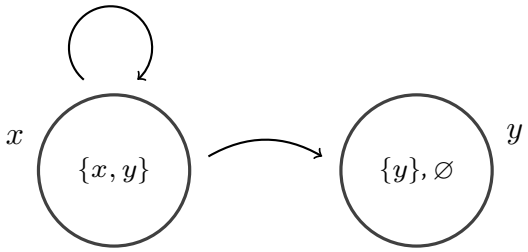

Figure 5: Example of a Kripke model

To understand why this language has a free logical flavour, one should be aware of the following distinction: An arbitrary set of worlds – an *intension* – is called a *proposition* at some world w if it is in the so-called existence domain $D(w)$ of that world.[16] The extension of Q at w maps each world to the set of intensions that are Q at that world, but not all of these intensions have to actually `exist` at w.

Let v be a variable assignment that assigns sets of worlds to propositional variables. Then, each formula of the language can be associated with a set of worlds in the following way:

$$[\![x]\!]_v := \begin{cases} v(x) & \text{if } x \text{ is a propositional variable} \\ [\![x]\!]_v & \text{if } x \text{ is an atomic sentence letter} \end{cases}$$

$$[\![s = t]\!]_v := \begin{cases} W & \text{if } [\![s]\!]_v = [\![t]\!]_v \\ \varnothing & \text{else} \end{cases}$$

[16] Identifying propositions with sets of worlds goes back to Stalnaker (1976, 1984) and Lewis (1986). Starr (2019:2.1, with regards to Lewis [1973]) and Kratzer (2012) contributed a very similar Kripkean semantics, and von Fintel and Heim (2011) published nicely written lecture notes on intensional semantics based on the λ-calculus. For additional information, the reader should refer to these publications.

$$
\begin{aligned}
[\![\neg s]\!]_v &:= W \setminus [\![s]\!]_v \\
[\![s \vee t]\!]_v &:= [\![s]\!]_v \cup [\![t]\!]_v \\
[\![\Box s]\!]_v &:= \{w \in W \mid R(w) \subseteq [\![s]\!]_v\} \\
[\![Q\, s]\!]_v &:= \{w \in W \mid [\![s]\!]_v \in |Q|(w)\} \\
[\![\forall x.\, s]\!]_v &:= \{w \in W \mid w \in [\![s]\!]_u, \forall u \text{ such that } u[x]v \text{ and } u(x) \in D(w)\},
\end{aligned}
$$

where $R(w)$ denotes the set of worlds accessible to w, so to say $\{x \mid R\, w\, x\}$, and $u[x]v$ means that the variable assignments u and v agree apart from, possibly, variable x. This concludes the language definition.

Exactly this semantic theory enabled Bacon et al. (2016) to conceive countermodels for Prior's theorem eventually showing that free logic is very much a fruitful framework for avoiding certain paradoxes worth investigating. One point, that is outstanding, is that the here advised free higher-order logic is based on the first-order language of set theory (Bacon et al. 2016: 11). Bacon et al. thus break with the long tradition of providing model theories for higher-order languages in a higher-order meta-logic as, e.g., HOL. For obvious reasons, including, inter alia, legibility, we will rely on the traditional approach for the general definition of free logic sought after in this thesis. Nonetheless, since set theory can easily be modelled within higher-order logic, both approaches are suitable for embedding them into Isabelle/HOL, which is after all the objective of this work. Therefore, in the following sections, we will give the general syntax and semantics of free higher-order logic in the meta-logic HOL before continuing with the corresponding embeddings and the computer-aided verification of Bacon, Hawthorne, and Uzquiano's results.

3.2 Formal Language of Free Higher-Order Logic

This section means to construe a free higher-order logic while keeping in mind the intention of an embedding and the preceding definitions of classical higher-order logic. Schütte (1960) and Farmer (1990, 1993, 2004) addressed the issue of how to properly define free higher-order logic founded on simple types very early. In that matter, Farmer (1990: 1271–1275) worked out eight major approaches for dealing with partial functions in quantificational logic and decided to focus on the one that uses a total valuation function for formulas and a partial valuation function for nonformula terms.[17] This strategy requires only few changes to the framework of classical higher-order logic and yet still handles nondenoting terms in a manner that closely corresponds to mathematical practice, which was exactly his driving force. Since our motivation is almost the same, we will inherit this approach. However, there will be a significant difference: Formulas and nonformula terms are *both* evaluated totally. But, of course, there will be some special mechanism allowing the early termination of an application, so to say an artificial partial application, whenever needed or required by the underlying free logic. This brings us even closer to a classical logic framework whilst not adding any

[17] Explicitly this tactic, in turn, was also advocated by Schock (1968) and Beeson (1985).

serious disadvantages. Our adapted approach allows for partial functions without actually incorporating partial valuation functions. Apart from that, the subsequent definitions also conveniently cover all the different and individual variations of free logic by requiring only minimal changes to the basic definitions. We shall now proceed with a detailed report of the syntax and semantics sketched here.

3.2.1 Syntax

All the free higher-order logics to be considered will share much of their syntactical structures, which are covered in this section. Except for terms, all definitions and terminology correspond to those presented in section 2.1 for classical higher-order logic. Some of them are repeated here to make the definitions in the following subsections easier to understand.

Definition 11. The set $\mathcal{T}$ of simple types is given by the following abstract grammar:

$$\alpha, \beta \;:=\; o \mid i \mid (\alpha \to \beta)\,.$$

Definition 12. The subset $\mathcal{T}_o \subset \mathcal{T}$ of simple types of type o is given by the following abstract grammar:

$$\beta \;:=\; o \mid (\alpha \to \beta)$$

with $\alpha \in \mathcal{T}$.

The nonempty subset $\mathcal{T}_i \subset \mathcal{T}$ of simple types of type i is defined analogously.

Simply typed terms of free higher-order logic essentially have the same structure as terms of classical higher-order logic but are additionally allowed to contain the nonlogical constant symbol $E!$.

Definition 13. Terms of FHOL are defined based on the following formation rules:

$$\begin{aligned} s, t \;:=\;& P_\alpha \mid x_\alpha \mid (E!_{\alpha\to o}\, s_\alpha)_o \mid ((=_{\alpha\to\alpha\to o}\, s_\alpha)_{\alpha\to o}\, t_\alpha)_o \mid \\ & (\neg_{o\to o}\, s_o)_o \mid ((\vee_{o\to o\to o} s_o)\, t_o)_o \mid (\forall_{(\alpha\to o)\to o}(\lambda x_\alpha .\, s_o)_{\alpha\to o})_o \mid \\ & (\iota_{(\alpha\to o)\to\alpha}(\lambda x_\alpha .\, s_o)_{\alpha\to o})_\alpha \mid (s_{\alpha\to\beta}\, t_\alpha)_\beta \mid (\lambda x_\alpha .\, s_\beta)_{\alpha\to\beta} \end{aligned}$$

with $\alpha, \beta \in \mathcal{T}$.

As Hintikka (1959) showed, the primitive unary existence predicate can alternatively be expressed as

$$E!_{\alpha\to o} \;:=\; \lambda x_\alpha .\, \exists y_\alpha .\, y = x\,,$$

where the existential quantifier iterates only over existent objects (cf. Lambert 1991b: 8). Note that by investigation of Meyer, Bencivenga, and Lambert (1982) either equality or a designated existence predicate has to be primitive in the language because otherwise no other

formula of FHOL could play the role of the term $E!\,s$. Rami (2014: 504n1) also stressed the logical relevance of such a predicate. Even though quantification in free logic is traditionally limited to existing objects, one could clearly introduce more quantifiers, e.g. $\forall^{+}$, that iterate over both existent and nonexistent objects. For the sake of readability, however, this was not applied.[18] Moreover, not only quantifiers have existential import: Definite descriptions of free logic denote a unique object satisfying some property if and only if it exists *and* is defined (Lambert 1972: 186; Bencivenga 1986: 417).

Lastly, it may be worth pointing out that the abbreviations for the logical constants as the existential quantifier, etc. also stay exactly the same as provided for HOL.

3.2.2 Semantics

The following proposal of semantics for free higher-order logic connects two well-conceived concepts which were worked out by Benzmüller and Scott (2019), based on Scott's work from 1967, and Farmer (1990, 1993, 2004). First, the general definitions that can be applied to all considered semantics are given, then we focus on the specific parts that are individual to each semantic variant that is discussed.

Definition 14. A *frame* D is a set $\{D_\alpha : \alpha \in \mathcal{T}\}$ of nonempty sets (formally *domains*) D_α such that D_i is chosen freely, $D_o = \{\mathrm{T}, \mathrm{F}\}$ where $\mathrm{T} \neq \mathrm{F}$ and T represents truth and F represents falsehood, and $D_{\alpha\to\beta}$ is the set of all total functions from domain D_α to codomain D_β.

Definition 15. A *subframe* E is a set $\{E_\alpha : \alpha \in \mathcal{T}\}$ of possibly empty sets (formally *domains*) D_α such that $E_\alpha \subset D_\alpha$ for each $\alpha \in \mathcal{T}_i$ and $E_\alpha = D_\alpha$ for each $\alpha \in \mathcal{T}_o$.[19]

We assume, inspired by Farmer (1990, 1993: 1277)[20], that $\bot_\alpha \in D_\alpha \setminus E_\alpha$ for all $\alpha \in \mathcal{T}_i$ with

$$\bot_{\alpha\to\beta}(d) \;:=\; \bot_\beta \quad \text{for all } d \in D_\alpha.$$

Furthermore, each set D_α with $\alpha \in \mathcal{T}_o$ contains the element F_α defined inductively by

$$\begin{aligned} \mathrm{F}_o &:= \mathrm{F} \\ \mathrm{F}_{\alpha\to\beta}(d) &:= \mathrm{F}_\beta \quad \text{for all } d \in D_\alpha. \end{aligned}$$

The purpose of these nonlogical constants is to propagate the nondetonation or falsehood of a term up through all terms containing it, wherein $\bot_i$ symbolizes 'the undefinedness'. Their intended use will be explained in the further course of this section. Exemplary diagrams of some of the domains can be found in fig. 6.

[18] Also because Scott (1970: 146–147), Lambert (1983: 122), and Antonelli (2000: 278), amongst others, had doubts about the metaphysical relevance of a 'Meinongian interpretation of outer domains' where quantification over all objects is justified.

[19] A perfectly reasonable alternative would be to restrict nondetonation to the domain of individuals, i.e., to define $E_i \subset D_i$ and for all $\alpha \neq i$, $E_\alpha = D_\alpha$ (cf. Barba Escribá 2001: 128). Though, unfortunately, this complicates the definition of strict functions.

[20] Farmer insisted, corresponding to mathematical practice, on a partial valuation for functions but a total valuation for formulas. Therefore he introduced F_α only. For this thesis, his idea is taken further to obtain total valuations for both.

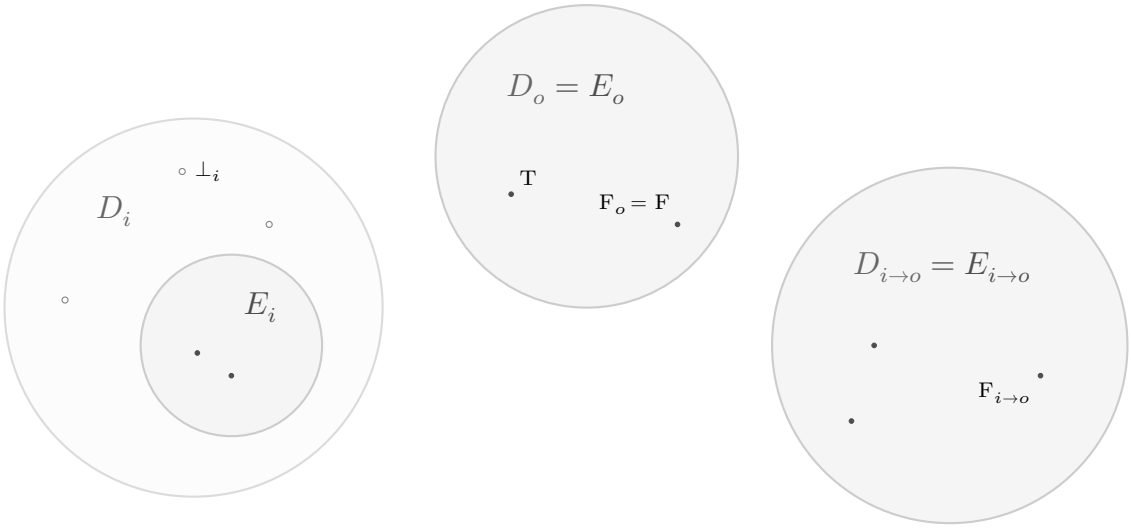

Figure 6: Schematics of domains D_i, D_o and $D_{i\to o}$

Definition 16. A *standard model* is a triple $M = \langle D, E, I \rangle$ where D is a frame, E is a subframe and I is a family of typed interpretation functions, i.e., $I = \{I_\alpha : \alpha \in \mathcal{T}\}$. Each *interpretation function* I_α maps constants of type α to appropriate elements of D_α.

The partiality characteristic for free logic is implemented by a trick that exploits the constant $\perp_\alpha$, which enables the functions in each domain $D_{\alpha\to\beta}$ to remain total. Hence, the generalization of standard models to Henkin models is equally applicable to free higher-order logic (cf. Leblanc 1980: 127, referring to Schütte [1960]).[21]

The interpretation of the logical constants, as well as the valuation function, depends on the free logical theory one wants to establish. Over the course of the years, many efforts have been made for a new conception of truth involving nondenoting terms. These efforts ultimately led to three fundamentally different theories, which are briefly summarized here:

- Nondenoting terms fail to correspond to a fact, and thus all atomic formulas containing them are false.
- Nondenoting terms are not well enough defined that atomic formulas involving them can receive any truth value.
- Atomic formulas with nondenoting terms can indeed evaluate to true as those terms are not truly nondenoting. They refer to objects that have a special form of existence, which actually is some kind of fact.

In light of these theories, logicians agreed upon distinguishing three types of free semantics: negative, positive, and neutral (cf. Bencivenga 1986; Lehmann 2002). For a comprehensive

[21] As shown by Farmer, it is possible to give a Henkin-style completeness proof for free higher-order logic with both partial and total functions (Farmer 1990: 1282–1286). He supplied his own notion of general models for this. Schütte (1960: 323–325) also came up with such a proof.

discussion of the philosophical backgrounds of each of them, refer to Bacon (2013). In what follows next, we shall only be concerned with the technical formalization of these semantics.

3.2.2.1 Positive Semantics

Positive semantics seems to be the most prevalent semantics for free logic – Scott (1967), Lambert (1963, 1964, 1967) and also Lambert and van Fraassen (1972) are just a few of the various famous names that have shaped this semantics. *Free higher-order logic with positive semantics* (PFHOL) allows atomic formulas with terms that do not denote to be true.[22] For example, $hasLegs(Pegasus)$ is regarded as a valid formula since the denotation of $Pegasus$ is a mythological creature which is usually depicted in the form of a winged horse (with legs). In this sense, the nonlogical constant *E!* and the logical constants $=$, $\neg$, $\vee$, $\forall$ and ι are interpreted as given below.

$$
\begin{array}{lllll}
I(E!_{\alpha\to o}) & := & ex & \in E_{\alpha\to o} & \text{s.t. for all } d \in D_\alpha, \\
 & & & & \quad ex(d) = \mathrm{T} \text{ iff } d \in E_\alpha \\
I(=_{\alpha\to\alpha\to o}) & := & id & \in E_{\alpha\to\alpha\to o} & \text{s.t. for all } d, d' \in D_\alpha, \\
 & & & & \quad id(d, d') = \mathrm{T} \text{ iff } d \text{ is identical to } d' \\
I(\neg_{o\to o}) & := & not & \in E_{o\to o} & \text{s.t. } not(\mathrm{T}) = \mathrm{F} \text{ and } not(\mathrm{F}) = \mathrm{T} \\
I(\vee_{o\to o\to o}) & := & or & \in E_{o\to o\to o} & \text{s.t. } or(v_1, v_2) = \mathrm{T} \text{ iff } v_1 = \mathrm{T} \text{ or } v_2 = \mathrm{T} \\
I(\forall_{(\alpha\to o)\to o}) & := & allq & \in E_{(\alpha\to o)\to o} & \text{s.t. for all } f \in D_{\alpha\to o}, \\
 & & & & \quad allq(f) = \mathrm{T} \text{ iff } f(d) = \mathrm{T} \text{ for all } d \in E_\alpha \\
I(\iota_{(\alpha\to o)\to\alpha}) & := & desc & \in E_{(\alpha\to o)\to\alpha} & \text{s.t. for all } f \in D_{\alpha\to o}, \\
 & & & & \quad desc(f) = d \in E_\alpha \text{ if } f(d) = \mathrm{T} \text{ and} \\
 & & & & \quad \text{for all } d' \in E_\alpha\text{: if } f(d') = \mathrm{T}, \text{ then } d' = d, \\
 & & & & \quad \text{otherwise } desc(f) = \perp_\alpha \text{ if } \alpha \in \mathcal{T}_i \text{ and} \\
 & & & & \quad desc(f) = \mathrm{F}_\alpha \text{ if } \alpha \in \mathcal{T}_o
\end{array}
$$

with $\alpha \in \mathcal{T}$.

The value $[\![s_\alpha]\!]^{M,g}$ of each PFHOL term s_α in a model M under the variable assignment g is an object $d \in D_\alpha$ and is evaluated as follows:

$$
\begin{array}{lll}
[\![P_\alpha]\!]^{M,g} & := & I(P_\alpha) \\
[\![x_\alpha]\!]^{M,g} & := & g(x_\alpha)
\end{array}
$$

[22] Lambert (1991a: 344, 1997: 62) and many more used the term positive as a synonym for not necessarily false, as is the case here. However, other ways of interpreting positive semantics can be found in the literature, e.g., Bencivenga (1986: 397) referred to a semantics where *each* atomic formula containing a nondenoting term is true.

$$[\![(s_{\alpha\to\beta}\, t_\alpha)_\beta]\!]^{M,g} := [\![s_{\alpha\to\beta}]\!]^{M,g} ([\![t_\alpha]\!]^{M,g})$$

$$[\![(\lambda x_\alpha .\, s_\beta)_{\alpha\to\beta}]\!]^{M,g} := \text{the function } f \text{ from } D_\alpha \text{ into } D_\beta \text{ s.t. for all } d \in D_\alpha\text{: } f(d) = [\![s_\beta]\!]^{M,g[x\to d]}$$

with $\alpha, \beta \in \mathcal{T}$.

The application is hereby defined in a nonstrict manner. A strict function application would be defined like this:

$$[\![(s_{\alpha\to_i\beta}\, t_\alpha)_\beta]\!]^{M,g} := \begin{cases} [\![s_{\alpha\to_i\beta}]\!]^{M,g} ([\![t_\alpha]\!]^{M,g}) & \text{if } [\![t_\alpha]\!]^{M,g} \in E_\alpha \text{ [23]} \\ \bot_\beta & \text{else} \end{cases}$$

with $(\alpha \to_i \beta) \in \mathcal{T}_i$.

A strictly applied function results in undefined if one of its arguments is undefined. In simple type theory, arguments are typically processed one after another. To be able to pass the undefined state of a once applied argument through any other possibly following arguments, the constant $\bot_\alpha$ was introduced for each relevant domain D_α. $\bot_{\alpha\to\beta}$ maps any argument of type α to $\bot_\beta$ until $\bot_i$ appears. This way, undefinedness is preserved until the evaluation of the application has reached its end.[24] Predicates, on the other hand, need no such special treatment. In positive free logic, atomic formulas denote truth or falsehood even if one of the arguments is undefined.[25] A sligthly different view will be considered in the next section.

3.2.2.2 Negative Semantics

Schock (1964, 1968), Scales (1969), and Burge (1974) were the first who motivated and refined a negative account of free logic. In *free higher-order logic with negative semantics* (NgFHOL), atomic formulas which contain nondenoting terms are all denied. So, a formula like $hasLegs(Pegasus)$ is certainly invalid. To achieve this result, the interpretations of the usual nonlogical and logical constants remain identical to the ones for positive semantics and the evaluation of a NgFHOL term s_α in a model M under the variable assignment g, so to say $[\![s_\alpha]\!]^{M,g}$, is defined as follows:

[23] Farmer (1990) also checked the function itself for existence. But since the distinction between existing and nonexisting functions – in contrast to existing and nonexisting individuals – is unusual and not well-defined, this was left out. After all, Farmer himself dropped it later on, too (cf. Farmer 1993; Farmer and Guttman 2000).

[24] Note that restraining applications like this could lead to malformed valuations, i.e., valuations might not receive their actually intended meaning. For instance, the ite operator must be handled separately in cases where the then- or else-parts are meant to be undefined.

[25] An obvious extension of positive semantics is to add a nonlogical constant $\bot_\alpha$ to each domain $\alpha \in \mathcal{T}_o$, so that predicates can likewise be affected by strictness. But one must bear in mind that by doing so, two-valued logic is abandoned. See the explanations related to neutral semantics in section 3.2.2.3 for more information on this.

$$
\begin{aligned}
[\![P_\alpha]\!]^{M,g} &:= I(P_\alpha) \\
[\![x_\alpha]\!]^{M,g} &:= g(x_\alpha) \\
[\![(s_{\alpha \to_i \beta}\, t_\alpha)_\beta]\!]^{M,g} &:= [\![s_{\alpha \to_i \beta}]\!]^{M,g} ([\![t_\alpha]\!]^{M,g}) && (1) \\
[\![(s_{\alpha \to_o \beta}\, t_\alpha)_\beta]\!]^{M,g} &:= \begin{cases} [\![s_{\alpha \to_o \beta}]\!]^{M,g} ([\![t_\alpha]\!]^{M,g}) & \text{if } [\![t_\alpha]\!]^{M,g} \in E_\alpha \\ \mathrm{F}_\beta & \text{else} \end{cases} && (2) \\
[\![(\lambda x_\alpha .\, s_\beta)_{\alpha \to \beta}]\!]^{M,g} &:= \text{the function } f \text{ from } D_\alpha \text{ into } D_\beta \\
&\quad \text{s.t. for all } d \in D_\alpha\text{: } f(d) = [\![s_\beta]\!]^{M,g[x \to d]}
\end{aligned}
$$

with $\alpha, \beta \in \mathcal{T}$, $(\alpha \to_i \beta) \in \mathcal{T}_i$, $(\alpha \to_o \beta) \in \mathcal{T}_o$.

Case (1) describes the function application, which is executed nonstrictly again.[26] Conversely, (2) defines how arguments are applied to predicates. When applying an undefined object to a predicate, the predicate needs to become false. Similarly to the approach seen in section 3.2.2.1, the nonlogical constant F_α in each domain D_α with $\alpha \in \mathcal{T}_o$ helps to propagate falsehood through all possibly following arguments after the undefined one. Besides ordinary predicates, this directly affects the evaluation of terms containing the logical constant for equality. For such an atomic formula to become true, both objects compared need to be defined. If one of them is not, then, as stipulated in free logic with negative semantics, the predicate application as in (2) will cause the corresponding formula to evaluate to false.[27] As a side effect of this constraint, the formula $E!\, t$ may syntactically also be abbreviated by $((= t)\, t)$ when using negative semantics (Burge 1974: 323).

3.2.2.3 Neutral Semantics

The former setup has a not unimportant downside: its bivalence. Positive and negative free logics require atomic formulas containing terms that do not refer to anything existing to be either true or false, even in cases where no decision can be made. The term $hasLegs(Pegasus)$ can readily be regarded as true, yet, since the denotation of $Pegasus$ is a not truly existing horse-demigod, it most probably has legs, but it is not entirely impossible that it does not. In fact, nobody knows. Much harder is $isSixFeetTall(Pegasus)$. For the size of Pegasus being purely speculative, the formula is neither true nor false, but simply indeterminate – a so-called 'truth value gap' as Bencivenga (1986: 395) named them. Assigning no truth value to such atomic formulas seems to be the best way to deal with the situation. But this inevitably means moving away from a bivalent to a trivalent logic, a logic with three different truth values. And, this brings up a new challenge: How do such logics spell out the truth-conditions of complex formulas with truth-valueless terms? Is $hasLegs(Pegasus) \wedge hasLegs(Horse)$ a truth?

[26] To define the function application in a strict way, adopt the technique presented in section 3.2.2.1.

[27] Some authors call free semantics having this restriction strongly negative semantics (Posy 2007: 643). Their understanding of a 'pure' negative semantics allows formulas like $((= s)\, t)$ to be true even if the terms are nondenoting (cf. Sainsbury 2005: 66).

What about $hasLegs(Pegasus) \vee \neg\, hasLegs(Pegasus)$? According to, e.g., Lehmann (1994, 2001, 2002), a formula should lack truth value if it encloses any truth-valueless subformula. A free logic that follows this line is called a neutral free logic, and in our case *free higher-order logic with neutral semantics* (NtFHOL). To meet the formal requirements of such a new kind of semantics, we need to update the definition of a subframe formerly given in section 3.2.2:

Definition 17. A *subframe* E is a set $\{E_\alpha : \alpha \in \mathcal{T}\}$ of possibly empty sets (formally *domains*) D_α such that $E_\alpha \subset D_\alpha$ for each $\alpha \in \mathcal{T}$.

We discard all values F_α and instead, in addition to $\bot_\alpha$ for each domain D_α with $\alpha \in \mathcal{T}_i$, include $\bot_\alpha \in D_\alpha \setminus E_\alpha$ for all $\alpha \in \mathcal{T}_o$ where again

$$\bot_{\alpha\to\beta}(d) \;:=\; \bot_\beta \quad \text{for all } d \in D_\alpha.$$

Note that this implies, for neutral free logic, that $D_o = \{\mathrm{T}, \mathrm{F}, \bot\}$ in each model. Figure 7 shows the new structure of some of the altered domains.

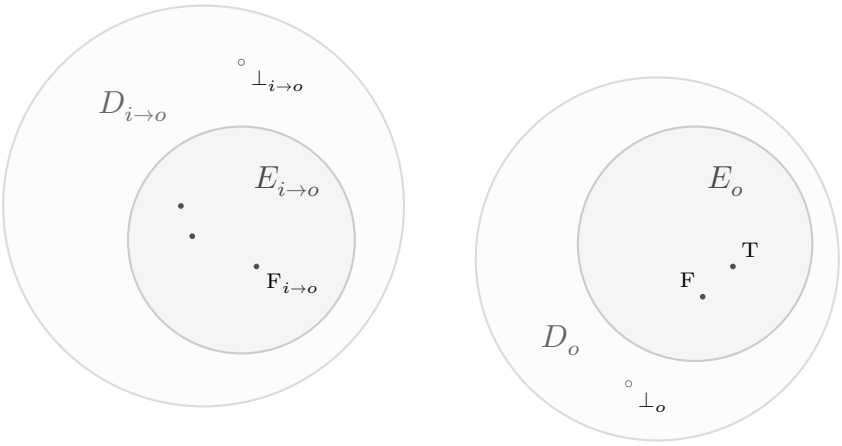

Figure 7: Schematics of domains $D_{i\to o}$ and D_o extended for neutral semantics

The forthcoming neutral free semantics follows Lehmann's (2002) representation except for the interpretation of the quantifier $\forall$. Lehmann and also Smiley (1960) defined the universal quantifier – unlike the other logical constants – to evaluate bivalently, i.e., quantified formulas are either true or false, but never without truth value. McKinsey (2020) expressed concerns[28] and suggested a different handling: Motivated by treating $E!\,x$ exactly as $\exists y.\, y = x$ if x does not denote, namely as truth-valueless, he required for the evaluation of $\forall x.\, s$ that all constants and free variables other than x in s must be existent (cf. McKinsey 2020: 81). If not,

[28] Lehmann (1994) took $\exists$ as primitive and interpreted it in the standard way, the universal quantifier was introduced via abbreviation. However, in the presence of nondenoting terms, $\exists x.\, s$ and $\neg\forall x.\, \neg s$ ceased to be equivalent. Later, Lehmann (2002: 234–235) treated both individually, though this alone could not eliminate all problems related to this approach (cf. McKinsey 2020: 57–60).

s is truth-valueless and hence the whole quantified formula, too. In order to provide a neutral semantics as advised, the nonlogical constant $E!$ and the logical constants $=$, $\neg$ and $\vee$ are interpreted as before, but $\forall$ obtains the following nonstandard interpretation:

$$I(\forall_{(\alpha\to o)\to o}) := allq \in E_{(\alpha\to o)\to o} \quad \text{s.t. for all } f \in D_{\alpha\to o},$$

$$allq(f) = \begin{cases} \mathrm{T} & \text{if } f(d) = \mathrm{T} \text{ for all } d \in E_\alpha \\ \mathrm{F} & \text{if } f(d) = \mathrm{F} \\ & \text{for at least one } d \in E_\alpha \\ \bot_o & \text{else} \end{cases}$$

with $\alpha \in \mathcal{T}$.

Additionally, taking advantage of the extension of the domains, the interpretation of the definite description operator ι can now be adapted as shown below.

$$I(\iota_{(\alpha\to o)\to\alpha}) := desc \in E_{(\alpha\to o)\to\alpha} \quad \text{s.t. for all } f \in D_{\alpha\to o},$$

$$desc(f) = \begin{cases} d \in E_\alpha & \text{if } f(d) = \mathrm{T} \text{ and} \\ & \text{for all } d' \in E_\alpha\text{:} \\ & \text{if } f(d') = \mathrm{T}, \\ & \text{then } d' = d \\ \bot_\alpha & \text{else} \end{cases}$$

with $\alpha \in \mathcal{T}$.

The value $[\![s_\alpha]\!]^{M,g}$ of each NtFHOL term s_α in a model M under the variable assignment g is an object $d \in D_\alpha$ and is evaluated as follows:

$$[\![P_\alpha]\!]^{M,g} := I(P_\alpha)$$

$$[\![x_\alpha]\!]^{M,g} := g(x_\alpha)$$

$$[\![(s_{\alpha\to_i\beta}\, t_\alpha)_\beta]\!]^{M,g} := [\![s_{\alpha\to_i\beta}]\!]^{M,g}([\![t_\alpha]\!]^{M,g})$$

$$[\![(s_{\alpha\to_o\beta}\, t_\alpha)_\beta]\!]^{M,g} := \begin{cases} [\![s_{\alpha\to_o\beta}]\!]^{M,g}([\![t_\alpha]\!]^{M,g}) & \text{if } [\![t_\alpha]\!]^{M,g} \in E_\alpha \\ \bot_\beta & \text{else} \end{cases} \tag{3}$$

$$[\![(\lambda x_\alpha .\, s_\beta)_{\alpha\to\beta}]\!]^{M,g} := \text{the function } f \text{ from } D_\alpha \text{ into } D_\beta \text{ s.t. for all } d \in D_\alpha\text{: } f(d) = [\![s_\beta]\!]^{M,g[x\to d]}$$

with $\alpha, \beta \in \mathcal{T}$, $(\alpha \to_i \beta) \in \mathcal{T}_i$, $(\alpha \to_o \beta) \in \mathcal{T}_o$.

As it was the case for equality in negative semantics, case (3) ensures that, in agreement with the general perception of neutral free logic, terms matching $((= s)\, t)$, $(\neg s)$, $((\vee s)\, t)$ or $E!\, s$ are evaluated to truth-valueless if at least one of s or t equals to $\bot_\alpha$ or some other undefined

object. The same holds for atomic formulas containing predicates.

Logics of this kind are thought to be very weak. If a formula includes any nondenoting term, it immediately lacks truth value and is therefore meaningless. Due to the change of domain D_o even the definition of a Henkin model is no longer applicable and hence completeness unclear. Many logical truths, indeed, remain at least weakly valid, i.e., not false on all models, which admits defining weak validity as a substitute for general validity (cf. Lehmann 2001: 153). But such a logic will still be weaker than classical logic and bivalent free logics. For this reason, most logicians rejected this strategy or tried to improve it, from which the method of supervaluations arose.

3.2.2.4 Supervaluational Semantics

Free logic with supervaluational semantics was originally developed by van Fraassen (1966a,b) and subsequently reviewed and sharpened by Skryms (1968), Meyer and Lambert (1968), Bencivenga (1981) and Woodruff (1984).[29] It strives to accommodate the trivalent nature of neutral free logic while still preserving the principles of classical reasoning. Even though the term $isSixFeetTall(Pegasus)$ is truth-valueless, it is still safe to assume the validity of $isSixFeetTall(Pegasus) \vee \neg isSixFeetTall(Pegasus)$. The general approach to the aforesaid semantic behavior involves that a partial valuation function is extended to a total one by arbitrarily assigning denotation to terms which previously were nondenoting. The combination of all such possible extensions, together with the information gathered by the partial function, will ultimately determine a so-called supervalue for all formulas of the language. Supervaluations are often considered as an alternative to positive free logic with dual domains since a single domain is equipped with an outer domain (cf. Bencivenga 1984: 1; Yeakel 2015: 87). However, in the current context, they will be defined already starting from inner-outer dual domain semantics. Instead of handing out denotation to nondenoting terms as suggested by Bencivenga, we change the denotative meaning of terms that map to the nonexistent objects $\perp_\alpha$ and leave all other denotations as they are. We will follow Bencivenga's (1981, 1986) work closely but have to modify some of his definitions for our purpose. For defining *free higher-order logic with supervaluational semantics* (SFHOL), we start by describing a primary auxiliary valuation function. The nonlogical constant *E!* and the logical constants $=$, $\neg$, $\vee$, $\forall$ and ι have the same interpretation as they have in neutral free logic, which means that in particular the connectives have the standard one. Then, the value $[\![s_\alpha]\!]^{M,g}_n$ of each SFHOL term s_α in a model M under the variable assignment g is an object $d \in D_\alpha$ and is evaluated as follows:

$$
\begin{aligned}
[\![P_\alpha]\!]^{M,g}_n &:= I(P_\alpha) \\
[\![x_\alpha]\!]^{M,g}_n &:= g(x_\alpha) \\
[\![(E!_{\alpha\to o}\, s_\alpha)_o]\!]^{M,g}_n &:= \begin{cases} \mathrm{T} & \text{if } [\![s_\alpha]\!]^{M,g}_n \in E_\alpha \\ \mathrm{F} & \text{else} \end{cases}
\end{aligned}
$$

[29] For spirited comments on this journey, consult the writing of Bencivenga (1986).

$$[\![((=_{\alpha\to\alpha\to o}\, s_\alpha)_{\alpha\to o}\, t_\alpha)_o]\!]^{M,g}_n := \begin{cases} \mathrm{T} & \text{if } [\![s_\alpha]\!]^{M,g}_n, [\![t_\alpha]\!]^{M,g}_n \in E_\alpha \text{ and} \\ & [\![s_\alpha]\!]^{M,g}_n = [\![t_\alpha]\!]^{M,g}_n \\ \mathrm{F} & \text{if } [\![s_\alpha]\!]^{M,g}_n, [\![t_\alpha]\!]^{M,g}_n \in E_\alpha \text{ and} \\ & [\![s_\alpha]\!]^{M,g}_n \neq [\![t_\alpha]\!]^{M,g}_n \\ & \text{or if } [\![s_\alpha]\!]^{M,g}_n \in E_\alpha \text{ and } [\![t_\alpha]\!]^{M,g}_n \notin E_\alpha \\ & \text{or if } [\![s_\alpha]\!]^{M,g}_n \notin E_\alpha \text{ and } [\![t_\alpha]\!]^{M,g}_n \in E_\alpha \\ \bot_o & \text{else} \end{cases}$$

$$[\![((\vee_{o\to o\to o} s_o)_{o\to o}\, t_o)_o]\!]^{M,g}_n := \begin{cases} \mathrm{T} & \text{if } [\![s_\alpha]\!]^{M,g}_n = \mathrm{T} \text{ or } [\![t_\alpha]\!]^{M,g}_n = \mathrm{T} \\ \mathrm{F} & \text{if } [\![s_\alpha]\!]^{M,g}_n = [\![t_\alpha]\!]^{M,g}_n = \mathrm{F} \\ \bot_o & \text{else} \end{cases}$$

$$[\![(s_{\alpha\to_i\beta}\, t_\alpha)_\beta]\!]^{M,g}_n := [\![s_{\alpha\to_i\beta}]\!]^{M,g}_n ([\![t_\alpha]\!]^{M,g}_n)$$

$$[\![(s_{\alpha\to_o\beta}\, t_\alpha)_\beta]\!]^{M,g}_n := \begin{cases} [\![s_{\alpha\to_o\beta}]\!]^{M,g}_n ([\![t_\alpha]\!]^{M,g}_n) & \text{if } [\![t_\alpha]\!]^{M,g}_n \in E_\alpha \\ \bot_\beta & \text{else} \end{cases}$$

$$[\![(\lambda x_\alpha .\, s_\beta)_{\alpha\to\beta}]\!]^{M,g}_n := \text{the function } f \text{ from } D_\alpha \text{ into } D_\beta \text{ s.t. for all } d \in D_\alpha\text{: } f(d) = [\![s_\beta]\!]^{M,g[x\to d]}_n$$

with $\alpha, \beta \in \mathcal{T}$, $(\alpha \to_i \beta) \in \mathcal{T}_i$, $(\alpha \to_o \beta) \in \mathcal{T}_o$.

It is of special interest here, that this valuation function is more extensive than the others presented in this thesis. Bencivenga rejected Lehmann's (2002) evaluation of neutral free logical terms for several reasons and interpreted equality, disjunction, and the existence predicate in his own way loosely following the truth tables of Kleene's (1952) strong logic of indeterminacy. From his perspective, it makes more sense if, for example, for equality and disjunction, both objects do not necessarily have to exist for the formula to obtain a defined truth value. It also goes well for him that $E!\, s$ evaluates to F if s is nondenoting instead of receiving no truth value. To reflect this view, the valuation function above has been complemented by extra cases for terms with $E!$, $=$ and $\vee$ allowing the function to evaluate exactly as Bencivenga meant it to. Nevertheless, it should be mentioned that Bencivenga (1981: 34–35) stressed, no matter how the connectives are set up, one will always be able to attain comparable results as he did by carefully adjusting the definitions.

Before we turn to the second auxiliary valuation function, we must specify what a completion is.

Definition 18. $M^{\bullet} = \langle D, E^{\bullet}, I^{\bullet} \rangle$ is a *completion* of a model $M = \langle D, E, I \rangle$ if and only if $E^{\bullet}_{\alpha} = D_{\alpha} \setminus \{\bot_{\alpha}\}$ [30] for each $\alpha \in \mathcal{T}$ and for all $P_{\alpha} \in C_{\alpha}, \alpha \in \mathcal{T}$[31]: $I^{\bullet}(P_{\alpha}) = I(P_{\alpha})$ whenever $I(P_{\alpha}) \in E^{\bullet}_{\alpha}$, and otherwise, with d_{α} being an arbitrary object in $E^{\bullet}_{\alpha}$, $I^{\bullet}(P_{\alpha}) = d_{\alpha}$. Further, for all $P_{\alpha\to\beta} \in C_{\alpha\to\beta}$, $\alpha, \beta \in \mathcal{T}$: $I^{\bullet}(P_{\alpha\to\beta})\, x_{\alpha} = I(P_{\alpha\to\beta})\, x_{\alpha}$ whenever $I(P_{\alpha\to\beta})\, x_{\alpha} \in E^{\bullet}_{\beta}$ and otherwise, with d_{β} being an arbitrary object in $E^{\bullet}_{\beta}$, $I^{\bullet}(P_{\alpha\to\beta})\, x_{\alpha} = d_{\beta}$. Moreover, $g^{\bullet}$ is the *completion* of a variable assignment g if and only if for all $x_{\alpha} \in V_{\alpha}$, $\alpha \in \mathcal{T}$: $g^{\bullet}_{\alpha}(x_{\alpha}) = g(x_{\alpha})$ whenever $g(x_{\alpha}) \in E^{\bullet}_{\alpha}$, and otherwise, with d_{α} being some arbitrary object in $E^{\bullet}_{\alpha}$, $g^{\bullet}(x_{\alpha}) = d_{\alpha}$.

Informally, a completion is the transformation of a free model into a classical one, easily confirmed by the fact that, in the completed model, terms denote objects from the inner domain only.[32] The secondary auxiliary valuation function is defined on the grounds of primary auxiliary evaluations and completions such that the value $[\![s_{\alpha}]\!]^{M,M^{\bullet},g,g^{\bullet}}_{s}$ of a SFHOL term s_{α} relative to a model M and its completion $M^{\bullet}$ under a variable assignment g and its completion $g^{\bullet}$ is an object $d \in E_{\alpha}$ for $\alpha = o$ or an object $d \in D_{\alpha}$ for all $\alpha \neq o$ evaluated as given below.[33]

$$[\![(\neg_{o\to o} s_o)_o]\!]^{M,M^{\bullet},g,g^{\bullet}}_{s} := \mathrm{T} \text{ iff } [\![s_o]\!]^{M,M^{\bullet},g,g^{\bullet}}_{s} = \mathrm{F}$$

$$[\![((\vee_{o\to o\to o} s_o)_{o\to o}\, t_o)_o]\!]^{M,M^{\bullet},g,g^{\bullet}}_{s} := \mathrm{T} \text{ iff } [\![s_o]\!]^{M,M^{\bullet},g,g^{\bullet}}_{s} = \mathrm{T} \text{ or } [\![t_o]\!]^{M,M^{\bullet},g,g^{\bullet}}_{s} = \mathrm{T}$$

$$[\![(\forall_{(\alpha\to o)\to o}(\lambda x_{\alpha}.\, s_o)_{\alpha\to o})_o]\!]^{M,M^{\bullet},g,g^{\bullet}}_{s} := \mathrm{T} \text{ iff for all } d \in E_{\alpha}\text{: } [\![s_o]\!]^{M,M^{\bullet},g[x\to d],g^{\bullet}[x\to d]}_{s} = \mathrm{T}$$

$$[\![(\iota_{(\alpha\to o)\to\alpha}(\lambda x_{\alpha}.\, s_o)_{\alpha\to o})_{\alpha}]\!]^{M,M^{\bullet},g,g^{\bullet}}_{s} := \begin{cases} d \in E_{\alpha} & \text{if } [\![s_o]\!]^{M,M^{\bullet},g[x\to d],g^{\bullet}[x\to d]}_{s} = \mathrm{T} \text{ and} \\ & \text{for all } d' \in E_{\alpha}\text{:} \\ & \text{if } [\![s_o]\!]^{M,M^{\bullet},g[x\to d'],g^{\bullet}[x\to d']}_{s} = \mathrm{T}, \\ & \text{then } d' = d \\ \bot_{\alpha} & \text{else if } \alpha \neq o \\ \mathrm{F} & \text{else} \end{cases}$$

$$[\![(\lambda x_{\alpha}.\, s_{\beta})_{\alpha\to\beta}]\!]^{M,M^{\bullet},g,g^{\bullet}}_{s} := \text{the function } f \text{ from } D_{\alpha} \text{ into } D_{\beta} \text{ s.t. for all } d \in D_{\alpha}\text{: } f(d) = [\![s_{\beta}]\!]^{M,M^{\bullet},g[x\to d],g^{\bullet}[x\to d]}_{s}$$

$$[\![s_{\alpha}]\!]^{M,M^{\bullet},g,g^{\bullet}}_{s} := \begin{cases} [\![s_{\alpha}]\!]^{M,g}_{n} & \text{if } [\![s_{\alpha}]\!]^{M,g}_{n} \in E_{\alpha} \\ [\![s_{\alpha}]\!]^{M^{\bullet},g^{\bullet}}_{n} & \text{else} \end{cases}$$

with $\alpha \in \mathcal{T}$.

[30] Similar concepts for combining inner-outer dual domains with supervaluations are found in Read (1995: 141–142), Barba Escribá (2001: 129) and Nolt (2018: 3.4). In essence, nonexistent objects are made to exist.

[31] According to Bencivenga (1981: 35–36), it is also sufficient to reassign the constants (and free variables) occurring in the formula under evaluation. But both options are 'likely to give the same results'.

[32] In terms of dual domain semantics, it might be better to call this process conversion and the respective altered model a converted model, but we rather retain the terminology used by Bencivenga.

[33] Note that the last pattern is a general one so that all terms not matched before will be matched at this point.

What is happening here is that a term of SFHOL is recursively evaluated by the secondary auxiliary valuation function until a certain subterm is reached. If this subterm receives a defined value by the primary auxiliary valuation function, it keeps this value. If not, it is evaluated once again using the primary auxiliary valuation function along with the provided completions. Hence, since constants and free variables always denote something existent in any completion, every secondary auxiliary evaluation applied to a formula yields a defined truth value.

Now, we are ready to establish the notion of a supervaluation. The value $[\![s_\alpha]\!]^{M,g}$ of a SFHOL term s_α in a model M under the variable assignment g is an object $d \in D_\alpha$ and defined as shown beneath.

$$[\![s_o]\!]^{M,g} \quad := \quad \begin{cases} \mathrm{T} & \text{if } [\![s_o]\!]_s^{M,M^\bullet,g,g^\bullet} = \mathrm{T} \text{ for every } M^\bullet, g^\bullet \\ \mathrm{F} & \text{if } [\![s_o]\!]_s^{M,M^\bullet,g,g^\bullet} = \mathrm{F} \text{ for every } M^\bullet, g^\bullet \\ \perp_{\mathrm{o}} & \text{else} \end{cases}$$

$$[\![s_\alpha]\!]^{M,g} \quad := \quad [\![s_\alpha]\!]_n^{M,g}$$

with $\alpha \in \mathcal{T}$, $\alpha \neq o$.

Obviously, supervaluations only pertain to formulas while nonformula terms continue to be evaluated as in ordinary neutral free logic. Any formula of SFHOL is *supertrue* if it is true in a standard model M under assignment g and all completions $M^\bullet$ and $g^\bullet$. It is *superfalse* if it is false in a standard model M under assignment g and all completions $M^\bullet$ and $g^\bullet$. Otherwise, it is *supervalueless*. Thus, a supervaluation is still a partial valuation with some formulas having no defined supervalue. Upon these definitions, it becomes clear that some classical validities are redeemed, which was exactly the motivation. For instance, a formula as $x = x$ turns out supertrue, even when x is nondenoting, due to it being true for all objects in a completed model. To get a better understanding of the evaluation mechanism, consider the following example brought up by Bencivenga (1986: 408): $[\![\forall x.\, Px \to Pa]\!]_s^{M,M^\bullet,g,g^\bullet}$. Let $[\![\forall x.\, Px]\!]_n^{M,g}$ evaluate to T (which is in E_o), and $[\![Pa]\!]_n^{M,g}$, assuming a is nondenoting, to truth-valueless (which is not in E_o). Now, say $[\![Pa]\!]_n^{M^\bullet,g^\bullet} = \mathrm{F}$. Cleary, $[\![\forall x.\, Px]\!]_n^{M^\bullet,g^\bullet}$ would also evaluate to false in the same model under the same variable assignment. Yet, in our case, for finishing the evaluation of the exemplary formula, we need to inspect, informally speaking, $[\![\forall x.\, Px]\!]_n^{M,g} \to [\![Pa]\!]_n^{M^\bullet,g^\bullet}$, which equals to $\mathrm{T} \to \mathrm{F}$, and that, in turn, is false and hence strong universal instantiation invalid. The example $[\![Px \vee \neg Px]\!]_s^{M,M^\bullet,g,g^\bullet}$ can be evaluated similarly and is left to the reader as an exercise.

However intuitively attractive supervaluations might seem, supervaluational validity as established by Bencivenga is neither compact nor recursively axiomatizable and therefore not strongly complete as shown by Woodruff (1984) and approved by Bencivenga (1984: 1). More than that, what Bencivenga (1986: 404) called the 'counterfactual theory of truth' faced a lot of criticism in recent years: Lambert (2001b: 262) worried that supervaluations 'will strike many as intolerably complex', Nolt (2018: 3.4) described supervaluations as merely 'built from completions that are in effect positive dual-domain models' and Lehmann (2002: 233) pointed out: "If supervaluations make sense in free logic, I believe we do not yet know why".

3.3 Inclusive Logic

Classical logic carries existential import because its quantification domain D is nonempty: There is always something in D for terms to denote. Logics in which this requirement is relinquished such that the domain of quantification may be empty are called *inclusive*.[34] Logics that are both free and inclusive are called *universally free* (cf. Lehmann 2002: 197). Emphasis should be placed here, that, contrary to what is often claimed, inclusive logics need not necessarily be free, nor need free logics necessarily be inclusive. In classical logic, some existential claims as $\exists x.\, x = x$ are logical truths. In an inclusive logic, no existentially quantified formula can be generally valid because the existence of anything is not guaranteed. Moreover, all universally quantified formulas are vacuously true in the empty domain. Hence, universal logics invalidate the principle of universal instantiation, i.e.,

$$\forall x.\, s \;\rightarrow\; s\,[x \rightarrow y], \tag{sUI}$$

which is valid not only in classical logic but also in some other logics that are not inclusive. In any inclusive logic, the empty domain is a counterexample to all such inferences.

Attentive readers will by now have noticed that we defined semantics for a universally free logic throughout section 3.2. The reason for this will – if not already obvious anyway – become clear as we proceed. For a free logic without inclusiveness, definition 15 concerning free models has to be modified appropriately, i.e., domain E_α needs to be nonempty for all $\alpha \in \mathcal{T}$.

[34] Historically Quine (1954) penned the term, though Jaśkowski (1934) was the first to formalize a system that is today known as inclusive logic – twenty-five years before free logic made an appearance (cf. Bencivenga 1986: 380).

4 Shallow Semantical Embedding

For reducing automation of any free logic to reasoning within a classical higher-order logic framework, also advertised as a 'universal reasoning framework' by Benzmüller (2017, 2019), we exploit a well known meta-logical approach called shallow semantical embedding, abbreviated as SSE. This technique has quite far-reaching roots (cf. Gabbay 1996; Ohlbach, Nonnengart, de Rijke, and Gabbay 2001), though in recent years especially Benzmüller has achieved great results with it.[35] Apart from the herein presented ones for free logics, SSEs are also available for other nonclassical logics as modal logics, intuitionistic logics, many-valued logics, dyadic deontic logics, and counting.

When shallowly embedding a source language into a target language, the semantics of the source language is mapped to the corresponding syntax of the target language. In other words, a shallow embedding is just a translation between languages.[36] In the SSEs for Isabelle/HOL shown in this section, the translation of a source language, in our case FHOL, into HOL is itself formalized in the meta-logic HOL[37]. These translations target only the semantical differences between the two languages, the model-theoretical constituents they have in common remain untouched. E.g., while the domain of individuals is shared as it is, the equational theory defining the mapping of FHOL into HOL explicitly models those existential peculiarities of free semantics that are not natively supported by classical logic.

As we shall see, shallow semantical embeddings convince with their sheer simplicity and flexibility. The implementation of any translation is highly readable and traceable, and individual properties of nonclassical logics or specific models can easily be embedded or axiomatized. Moreover, higher-order provers need not to be adjusted at any time and can be used out of the box serving properly encoded, translated HOL input files. For all these reasons, the SSE approach is well suited for automated and interactive reasoning within theories of free higher-order logic and hence chosen for the purpose of this thesis.

In what follows, all the shallow semantical embeddings of free higher-order logic into classical higher-order logic developed for this study are, variant by variant, explained theoretically, proven where appropriate, and then encoded into Isabelle/HOL.[38]

4.1 Embedding of PFHOL into HOL

To provide a shallow embedding of PFHOL into HOL, the syntax and semantics of classical higher-order logic as presented in section 2 have to be enriched with an additional nonlogical

[35] The potential of the shallow semantical embedding approach has been demonstrated in numerous experiments relevant to Gödel's ontological argument by Benzmüller and Woltzenlogel Paleo (2013, 2014, 2016).

[36] A *deep embedding*, in contrast, represents the syntax of the source language within the target language by utilizing some kind of inductive data structure. Then, instances of the data structure are traversed and translated into terms of the target language. See, for example, the conference poster of Villadsen, Schlichtkrull, and Hess (2015) to get an impression of the general idea.

[37] If one is to be precise, Isabelle/HOL's meta-logic is only a well-chosen fragment of higher-order logic (Grundy and Newey 1998: 126).

[38] All encodings have been written and tested in Isabelle/HOL's June 2019 version. The .thy files can also be accessed via `https://github.com/stilleben/Free-Higher-Order-Logic`.

constant, a unary predicate $E_{\alpha\to o}$, which artificially enables one to distinguish between existing and nonexisting objects in the domain D_α. Besides, we include an error value e_α in each domain D_α with $\alpha \in \mathcal{T}$, which is meant to be the object that is returned by the definite description of type $(\alpha \to o) \to \alpha$ if no such object exists. We redefine the interpretation of ι thusly as follows:

$$\begin{array}{lcll} I(\iota_{(\alpha\to o)\to\alpha}) & := & desc \in D_{(\alpha\to o)\to\alpha} & \text{s.t. for all } f \in D_{\alpha\to o}, \\ & & & desc(f) = d \in D_\alpha \text{ if } f(d) = \mathrm{T} \text{ and} \\ & & & \text{for all } d' \in D_\alpha\text{: if } f(d') = \mathrm{T}, \text{ then } d = d', \\ & & & \text{otherwise } desc(f) = e_\alpha \end{array}$$

with $\alpha \in \mathcal{T}$.

Obviously, $(E_{\alpha\to o}\, e_\alpha)_o = \mathrm{F}$ for each $\alpha \in \mathcal{T}_i$, and for all $\alpha \in \mathcal{T}_o$: $(\forall x_\alpha.\, (E_{\alpha\to o}\, x_\alpha)_o)_o = \mathrm{T}$. Then, a HOL term $[\, s_\alpha \,]$ is assigned to each PFHOL term s_α according to the following translation function:[39]

$$\begin{array}{lcl}
[\, P_\alpha \,] & = & P_\alpha \\
[\, x_\alpha \,] & = & x_\alpha \\
[\, (E!_{\alpha\to o}\, s_\alpha)_o \,] & = & (E_{\alpha\to o}\, [\, s_\alpha \,])_o \\
[\, ((=^{F}_{\alpha\to\alpha\to o}\, s_\alpha)_{\alpha\to o}\, t_\alpha)_o \,] & = & ((=^{H}_{\alpha\to\alpha\to o}\, [\, s_\alpha \,])_{\alpha\to o}\, [\, t_\alpha \,])_o \\
[\, (\neg^{F}_{o\to o}\, s_o)_o \,] & = & (\neg^{H}_{o\to o}\, [\, s_o \,])_o \\
[\, ((\wedge^{F}_{o\to o\to o}\, s_o)_{o\to o}\, t_o)_o \,] & = & ((\wedge^{H}_{o\to o\to o}\, [\, s_o \,])_{o\to o}\, [\, t_o \,])_o \\
[\, (\forall^{F}_{(\alpha\to o)\to o}(\lambda x_\alpha.\, s_o)_{\alpha\to o})_o \,] & = & (\forall^{H}_{(\alpha\to o)\to o}(\lambda x_\alpha.\, ((E_{\alpha\to o}\, x_\alpha)_o \to^{H}_{o\to o\to o} [\, s_o \,])_o)_{\alpha\to o})_o \\
[\, (\iota^{F}_{(\alpha\to o)\to\alpha}(\lambda x_\alpha.\, s_o)_{\alpha\to o})_\alpha \,] & = & (\iota^{H}_{(\alpha\to o)\to\alpha}(\lambda x_\alpha.\, ((E_{\alpha\to o}\, x_\alpha)_o \wedge^{H}_{o\to o\to o} [\, s_o \,])_o)_{\alpha\to o})_\alpha \\
[\, (s_{\alpha\to\beta}\, t_\alpha)_\beta \,] & = & ([\, s_{\alpha\to\beta} \,]\, [\, t_\alpha \,])_\beta \\
[\, (\lambda x_\alpha.\, s_\beta)_{\alpha\to\beta} \,] & = & (\lambda x_\alpha.\, [\, s_\beta \,])_{\alpha\to\beta}
\end{array}$$

with $\alpha, \beta \in \mathcal{T}$.

Note that in the embeddings and wherever it is necessary to differentiate them, operators of HOL are annotated with a superscript ^{H}, while operators of any kind of FHOL are annotated with ^{F}.

The main trick of this translation is that the existential import of the quantifier and description operator is secured by cleverly exploiting the additional predicate $E_{\alpha\to o}$ as a guard. When mapping definite descriptions, $[\, (\iota^{F}_{(\alpha\to o)\to\alpha}(\lambda x_\alpha.\, s_o)_{\alpha\to o})_\alpha \,]$ could also be translated into

[39] A very similar translation, even though for first-order logic, was provided and proved to be sound and complete by Meyer and Lambert (1968). Benzmüller and Scott (2016, 2019) also came up with such a translation.

$$
\begin{array}{l}
(ite^{H}_{o\to\alpha\to\alpha\to\alpha} \\
\quad (\exists^{H}_{(\alpha\to o)\to o}(\lambda x_\alpha . \\
\quad\quad (((E_{\alpha\to o}\, x_\alpha)_o \wedge^{H}_{o\to o\to o} [\, s_o \,])_o \\
\quad\quad\quad \wedge^{H}_{o\to o\to o} (\forall^{H}_{(\alpha\to o)\to o}(\lambda y_\alpha . \\
\quad\quad\quad\quad\quad (((E_{\alpha\to o}\, y_\alpha)_o \to^{H}_{o\to o\to o} [\, s_o \,])_o \\
\quad\quad\quad\quad\quad\quad \to^{H}_{o\to o\to o} \\
\quad\quad\quad\quad\quad\quad (y_\alpha =^{H}_{\alpha\to\alpha\to o} x_\alpha)_o)_o)_{\alpha\to o})_o)_o)_{\alpha\to o})_o \\
\quad (\iota^{H}_{(\alpha\to o)\to\alpha}(\lambda x_\alpha .\, ((E_{\alpha\to o}\, x_\alpha)_o \wedge^{H}_{o\to o\to o} [\, s_o \,])_o)_{\alpha\to o})_\alpha \\
\quad e_\alpha)_\alpha
\end{array}
$$

using the if-then-else operator ite to ensure that the classical description definitely returns the error value e_α in case of no such existing object. But due to our previously done redefinition of classical definite description, this is not really necessary for this embedding. Furthermore, it is noteworthy that any term $\exists^{F} x.\, s$ is translated into $\neg^{H}\forall^{H} x.\, E\, x \to^{H} \neg^{H} s$, which is the same as $\exists^{H} x.\, E\, x \wedge^{H} s$ (cf. Meyer and Lambert 1968: 11n9).

The faithfulness of the translation will be proved in the next section.

4.1.1 Proof of Faithfulness

The two theorems stated below are already known to hold for HOL with Henkin semantics through the work of Henkin (1950) and Andrews (1972a,b).

Theorem 1 (Soundness). If $\vdash_{\text{HOL}} s_o$, then $\vDash_{\text{HOL}} s_o$.

Theorem 2 (Completeness). If $\vDash_{\text{HOL}} s_o$, then $\vdash_{\text{HOL}} s_o$.

$\vdash_{\text{HOL}} s_o$ expresses that s_o is provable in HOL. Thus, to demonstrate that the embedding in section 4.1 is sound and complete, i.e., faithful, with respect to Henkin-style general model semantics, it suffices to show that $\vDash_{\text{PFHOL}} s_o$ if and only if $\vDash_{\text{HOL}} [\, s_o \,]$. For this proof, we first need to elaborate how to transform a PFHOL model M into a HOL model M', and a PFHOL variable assignment g into a HOL variable assignment g'. We assume, that $D'_\alpha = D_\alpha$ and $C'_\alpha = C_\alpha$ for all $\alpha \in \mathcal{T}$, and set $e_\alpha = \bot_\alpha$ for each $\alpha \in \mathcal{T}_i$ and $e_\alpha = \mathrm{F}_\alpha$ for each $\alpha \in \mathcal{T}_o$. Then, $M = \langle D, E, I \rangle$ equals to the model $M' = \langle D', I' \rangle$ where I' is a family of interpretation functions that assigns the standard interpretation to all logical constants of HOL. For all other constants $P_\alpha \in C'_\alpha$: $I'(P_\alpha) = I(P_\alpha)$. Additionally, the nonlogical constant $E_{\alpha\to o}$ is interpreted as follows:

$$
\begin{array}{lll}
I'(E_{\alpha\to o}) & := \quad ex \ \in D'_{\alpha\to o} & \text{s.t. for all } d \in D'_\alpha, \\
 & & \quad ex(d) = \mathrm{T} \text{ iff } d \in E_\alpha
\end{array}
$$

with $\alpha \in \mathcal{T}$.

We further assume an identical denumerable list of individual variables, so to say, $V_\alpha^{'} = V_\alpha$ for all $\alpha \in \mathcal{T}$. Then, $g_\alpha^{'} : V_\alpha^{'} \to D_\alpha^{'}$ for all $\alpha \in \mathcal{T}$ is defined such that

$$g_\alpha^{'}(x_\alpha) \;=\; g(x_\alpha) \quad \text{for all } x_\alpha \in V_\alpha^{'}\,.$$

It is easy to see that both M and $M^{'}$ are Henkin models.

Lemma 1. For all PFHOL models M and PFHOL variable assignments g,

$$[\![\, s_\alpha \,]\!]^{M,g} \;=\; [\![\, [\, s_\alpha \,] \,]\!]^{M^{'},g^{'}}\,.$$

The lengthy proof of this lemma will not be given here, but can be found in appendix A.1.

Theorem 3. $\models_{\text{PFHOL}} s_o$ if and only if $\models_{\text{HOL}} [\, s_o \,]$.

Proof.

($\rightarrow$) The proof is by contraposition:

Assume $\not\models_{\text{PFHOL}} s_o$. Then, there exists at least one combination of a PFHOL model M and a variable assignment g so that $[\![\, s_o \,]\!]^{M,g} = \mathrm{F}$. By lemma 1, $[\![\, s_o \,]\!]^{M,g} = [\![\, [\, s_o \,] \,]\!]^{M^{'},g^{'}} = \mathrm{F}$. Hence, $\not\models_{\text{HOL}} [\, s_o \,]$.

($\leftarrow$) Analogous to above by contraposition and lemma 1. ∎

Therefore, the embedding of PFHOL into HOL is sound and complete.

4.1.2 Encoding into Isabelle/HOL

We will now encode the embedding explained in section 4.1 into Isabelle/HOL. Isabelle/HOL, as already mentioned, uses the meta-language HOL for the representation of classical higher-order logic formulas. The general syntax and semantics of Isabelle/HOL can be studied in the tutorial by Nipkow, Paulson, and Wenzel (2019a), and is therefore omitted here. In the following, we will introduce, in close reference to the general idea of the embedding presented before, constants, axiomatizations and definitions which combined result in a full theory for Isabelle/HOL. When defining a constant or an operator, the type of it is given as a so-called signature for which we need to declare a second – besides `bool` – base type `i` for individuals.

```
typedecl i
```

Next, we define an existence predicate `E` for each of the base and compound types. The single quote in `'a` indicates that this is a type variable, meaning, that the definition given hereupon is polymorphic.[40] The prefix 'f' in this and all upcoming definitions stands for 'free'.

[40] Use of polymorphism has advantages and disadvantages, but the provided encoding can easily be rewritten to work with specific types.

```
consts fExistence :: "'a ⇒ bool" ("E")
```

Then, we introduce a new constant **e** for every type and, accordingly to the definitions in section 4.1, let **e** of type `i` be nonexistent and **e** of type `bool` be `False`. Furthermore, `True` and `False` are set to be existent.

```
consts fUndef :: "'a" ("e")
axiomatization where fUndefIAxiom: "¬E (e::i)"
axiomatization where fFalsehoodBAxiom: "(e::bool) = False"
axiomatization where fTrueAxiom: "E True"
axiomatization where fFalseAxiom: "E False"
```

The embedding of the logical connectives is rather uncomplicated and performed beneath. The new embedded operators of FHOL are identified by using bold-face fonts, the native HOL operators are lighter.

```
definition fIdentity :: "'a ⇒ 'a ⇒ bool" (infixr "=" 56)
    where "φ = ψ ≡ φ = ψ"

definition fNot :: "bool ⇒ bool" ("¬_" [52]53)
    where "¬φ ≡ ¬φ"
definition fOr :: "bool ⇒ bool ⇒ bool" (infixr "V" 51)
    where "φ V ψ ≡ φ ∨ ψ"
```

The numbers, e.g., in `(infixr "=" 56)` or `(binder "∀" [8]9)` as seen below, help avoiding brackets in formulas by specifying structural priorities.

Now, for embedding the existential import of the universal quantifier, we utilize the existence predicate `E` of the respective type exactly as discussed in section 4.1. Isabelle/HOL supports the introduction of syntactic sugar for binding notations, which we adopt in the following definition in order to benefit from the more familiar notation $\forall x.\, Px$ instead of writing $\forall(\lambda x.\, Px)$ or $\forall P$.

```
definition fForall :: "('a ⇒ bool) ⇒ bool" ("∀")
    where "∀Φ ≡ ∀x. E x ⟶ Φ x"
definition fForallBinder:: "('a ⇒ bool) ⇒ bool" (binder "∀" [8]9)
    where "∀x. φ x ≡ ∀φ"
```

Note that the set the $\forall$ quantifier ranges over could also be empty and the embedded logic would therefore be universally free unless the following axiomatization is added:

```
axiomatization where fNonemptyDomains: "∃x. E x"
```

For the encoding of the FHOL operator ι, we rely on Isabelle/HOL's own definite description operator `THE`. Unlike the formal embedding shown in section 4.1, here we must specify the exact object that will be returned if there is no unique object that has the desired property. We use Isabelle/HOL's if-then-else operator for this:

```
definition fThat :: "('a ⇒ bool) ⇒ 'a" ("I")
  where "IΦ ≡ if ∃x. E x ∧ Φ x ∧ (∀y. (E y ∧ Φ y) ⟶ (y = x))
              then THE x. E x ∧ Φ x
              else e"
definition fThatBinder:: "('a ⇒ bool) ⇒ 'a" (binder "I" [8]9)
  where "Ix. φ x ≡ Iφ"
```

Analogous to the precedent case, we introduced binder notation for **I**. The embedding of further free logical constants is presented below.

```
definition fAnd :: "bool ⇒ bool ⇒ bool" (infixr "∧" 52)
  where "φ ∧ ψ ≡ ¬(¬φ ∨ ¬ψ)"
definition fImp :: "bool ⇒ bool ⇒ bool" (infixr "⟶" 49)
  where "φ ⟶ ψ ≡ ¬φ ∨ ψ"
definition fEquiv :: "bool ⇒ bool ⇒ bool" (infixr "↔" 50)
  where "φ ↔ ψ ≡ φ ⟶ ψ ∧ ψ ⟶ φ"

definition fExists :: "('a ⇒ bool) ⇒ bool" ("∃")
  where "∃Φ ≡ ¬(∀(λy. ¬(Φ y)))"
definition fExistsBinder :: "('a ⇒ bool) ⇒ bool" (binder "∃" [8]9)
  where "∃x. φ x ≡ ∃φ"
```

In the remainder of this section, we will test our encoding to see if the embedding works as expected. For this, we will run the Isabelle/HOL tool Sledgehammer on simple formulas known to hold in positive free logic. First, we analyze the formula $x = x$ in two different ways, one time x is arbitrary and the other time it is an object that does not exist. As Sledgehammer found out, both variants are valid:

```
consts fIndividual1 :: "i" ("i₁")
axiomatization where fUndefIndividual1Axiom: "¬(E i₁)"

lemma "x = x" unfolding Defs[41] by auto
lemma "i₁ = i₁" by (simp add: fIdentity_def)
```

For the next tests, we will investigate the principles of universal instantiation and existential generalization. As expected, the Isabelle/HOL built-in model finder Nitpick was able to

[41] `unfolding Defs` is a mechanism for passing on our definitions as necessary facts to Sledgehammer.

find countermodels for sUI, strong universal instantiation, and for sEG, strong existential generalization:

```
lemma "(∀x. P x) ⟶ P x"
  nitpick [user_axioms=true, show_all, format=2, card i=2]
  oops
```

```
Nitpick found a counterexample for card 'a = 3 and card i = 3:[42]

  Free variables:
    P = (λx. _)(a₁ := False, a₂ := False, a₃ := True)
    x = a₂
  Constants:
    E = (λx. _)(a₁ := False, a₂ := False, a₃ := True)
    E = (λx. _)(i₁ := False, i₂ := False, i₃ := False)
    e = i₁
    e = False
```

```
lemma "P x ⟶ (∃x. P x)"
  nitpick [user_axioms=true, show_all, format=2]
  oops
```

```
Nitpick found a counterexample for card 'a = 4 and card i = 4:

  Free variables:
    P = (λx. _)(a₁ := False, a₂ := False, a₃ := True, a₄ := True)
    x = a₄
  Constants:
    E = (λx. _)(a₁ := False, a₂ := False, a₃ := False, a₄ := False)
    E = (λx. _)(i₁ := False, i₂ := False, i₃ := False, i₄ := False)
    e = i₃
    e = False
```

Each of the two countermodels can easily be seen as correct. On the contrary, wUI and wEG, weak universal instantiation and weak existential generalization, are both valid:

```
lemma "((∀x. (P x)) ∧ (E x)) ⟶ (P x)"
  by (metis fAnd_def fForallBinder_def fForall_def fImp_def fNot_def
    fOr_def)
```

[42] Seeing such verbose countermodels is an undesired side effect of Isabelle/HOL's polymorphism. Nonetheless, the countermodels are still perfectly fine.

```
lemma "((P x) ∧ (E x)) → (∃x. P x)"
  unfolding Defs
  by blast
```

Hence, the embedding seems to embody all fundamentals of positive free logic. The complete embedding can be found in the appendix, in B.2. A nonpolymorphic version is also provided in B.3.

4.2 Embedding of NgFHOL into HOL

For the embedding of negative semantics, the same enriched syntax and semantics of HOL is used as for the embedding shown in section 4.1 for positive free logic. In particular, we once again make use of the unary existence predicate $E_{\alpha\to o}$. A HOL term $[\,s_\alpha\,]$ is associated with a NgFHOL term s_α in the following way:

$$
\begin{array}{lcl}
[\,P_\alpha\,] & = & P_\alpha \\
[\,x_\alpha\,] & = & x_\alpha \\
[\,(E!_{\alpha\to o}\, s_\alpha)_o\,] & = & (E_{\alpha\to o}\,[\,s_\alpha\,])_o \\
[\,((=^{F}_{\alpha\to\alpha\to o}\, s_\alpha)_{\alpha\to o}\, t_\alpha)_o\,] & = & ((\wedge^{H}_{o\to o\to o}\,((\wedge^{H}_{o\to o\to o}\,(E_{\alpha\to o}[\,s_\alpha\,])_o)_{o\to o} \\
 & & \qquad (E_{\alpha\to o}[\,t_\alpha\,])_o)_o)_{o\to o} \\
 & & \quad ((=^{H}_{\alpha\to\alpha\to o}\,[\,s_\alpha\,])_{\alpha\to o}\,[\,t_\alpha\,])_o)_o \\
[\,(\neg^{F}_{o\to o}\, s_o)_o\,] & = & (\neg^{H}_{o\to o}\,[\,s_o\,])_o \\
[\,((\wedge^{F}_{o\to o\to o}\, s_o)_{o\to o}\, t_o)_o\,] & = & ((\wedge^{H}_{o\to o\to o}\,[\,s_o\,])_{o\to o}\,[\,t_o\,])_o \\
[\,(\forall^{F}_{(\alpha\to o)\to o}(\lambda x_\alpha.\, s_o)_{\alpha\to o})_o\,] & = & (\forall^{H}_{(\alpha\to o)\to o}(\lambda x_\alpha.\,((E_{\alpha\to o}\, x_\alpha)_o \to^{H}_{o\to o\to o}\,[\,s_o\,])_o)_{\alpha\to o})_o \\
[\,(\iota^{F}_{(\alpha\to o)\to\alpha}(\lambda x_\alpha.\, s_o)_{\alpha\to o})_\alpha\,] & = & (\iota^{H}_{(\alpha\to o)\to\alpha}(\lambda x_\alpha.\,((E_{\alpha\to o}\, x_\alpha)_o \wedge^{H}_{o\to o\to o}\,[\,s_o\,])_o)_{\alpha\to o})_\alpha \\
[\,(s_{\alpha\to_i\beta}\, t_\alpha)_\beta\,] & = & ([\,s_{\alpha\to_i\beta}\,]\,[\,t_\alpha\,])_\beta \\
[\,(s_{\alpha\to_o\beta}\, t_\alpha)_\beta\,] & = & (ite^{H}_{o\to\beta\to\beta\to\beta}\,(E_{\alpha\to o}\, t_\alpha)_o\,([\,s_{\alpha\to_o\beta}\,]\,[\,t_\alpha\,])_\beta\, e_\beta)_\beta \\
[\,(\lambda x_\alpha.\, s_\beta)_{\alpha\to\beta}\,] & = & (\lambda x_\alpha.\,[\,s_\beta\,])_{\alpha\to\beta}
\end{array}
$$

with $\alpha, \beta \in \mathcal{T}$, $(\alpha \to_i \beta) \in \mathcal{T}_i$, $(\alpha \to_o \beta) \in \mathcal{T}_o$.

The biggest contrast to the previous embedding is that terms of type $\alpha \in \mathcal{T}_o$ are evaluated differently, which influences the application of arguments within formulas. But since $D_o = E_o$, most connectives and quantifiers, and by definition also $E!_{\alpha\to o}$, need not to be treated individually in this mapping. Only the translation of terms with equality has to be fulfilled carefully with additional conjunctions to conform to the doctrines of negative semantics. The correct evaluation of predicates, in turn, is ensured by implementing the application with a

proper utilization of the ite operator.

4.2.1 Proof of Faithfulness

To proof that the embedding in section 4.2 is sound and complete, it suffices, similar to the proof of faithfulness shown in section 4.1.1, to show that $\models_{\text{NgFHOL}} s_o$ if and only if $\models_{\text{HOL}} [\, s_o \,]$. The transformation of a NgFHOL model M and a NgFHOL variable assignment g into a HOL model $M^{'}$ and a HOL variable assignment $g^{'}$ is done in the same way as described in section 4.1.1 for a PFHOL model. Anew we start with the following lemma:

Lemma 2. For all NgFHOL models M and NgFHOL variable assignments g,

$$[\![\, s_\alpha \,]\!]^{M,g} \;=\; [\![\, [\, s_\alpha \,] \,]\!]^{M^{'},g^{'}}.$$

For the proof, the reader is again referred to the appendix.

Theorem 4. $\models_{\text{NgFHOL}} s_o$ if and only if $\models_{\text{HOL}} [\, s_o \,]$.

Proof.

The proof is exactly the same as for theorem 3. ∎

Soundness and completeness of the embedding of NgFHOL into HOL ensue.

4.2.2 Encoding into Isabelle/HOL

For the Isabelle/HOL encoding of the embedding of NgFHOL into HOL, we once more begin with a general setup as follows:

```
typedecl i
consts fExistence :: "'a ⇒ bool" ("E")

consts fUndef :: "'a" ("e")
axiomatization where fUndefIAxiom: "¬E (e::i)"
axiomatization where fFalsehoodBAxiom: "(e::bool) = False"
axiomatization where fTrueAxiom: "E True"
axiomatization where fFalseAxiom: "E False"
```

Equality cannot be translated in a trivial way, because, as explained in section 4.2, we must ensure that the two objects in comparison are not only equal, but also existent. Otherwise, the FHOL formula cannot be true. We implement it like that:

```
definition fIdentity :: "'a ⇒ 'a ⇒ bool" (infixr "=" 56)
    where "φ = ψ ≡ E φ ∧ E ψ ∧ (φ = ψ)"
```

The remaining embedding is identical to the one the reader has seen in section 4.1.2 for PFHOL, so it will only be reproduced here, but left uncommented. The axiomatization for the domain of quantification to be nonempty is again optional and omitted here.

```
definition fNot :: "bool ⇒ bool" ("¬_" [52]53)
    where "¬φ ≡ ¬φ"
definition fOr :: "bool ⇒ bool ⇒ bool" (infixr "V" 51)
    where "φ V ψ ≡ φ ∨ ψ"

definition fAnd :: "bool ⇒ bool ⇒ bool" (infixr "Λ" 52)
  where "φ Λ ψ ≡ ¬(¬φ V ¬ψ)"
definition fImp :: "bool ⇒ bool ⇒ bool" (infixr "→" 49)
  where "φ → ψ ≡ ¬φ V ψ"
definition fEquiv :: "bool ⇒ bool ⇒ bool" (infixr "↔" 50)
  where "φ ↔ ψ ≡ φ → ψ Λ ψ → φ"

definition fForall :: "('a ⇒ bool) ⇒ bool" ("∀")
  where "∀Φ ≡ ∀x. E x ⟶ Φ x"
definition fForallBinder:: "('a ⇒ bool) ⇒ bool" (binder "∀" [8]9)
  where "∀x. φ x ≡ ∀φ"

definition fExists :: "('a ⇒ bool) ⇒ bool" ("∃")
  where "∃Φ ≡ ¬(∀(λy. ¬(Φ y)))"
definition fExistsBinder :: "('a ⇒ bool) ⇒ bool" (binder "∃" [8]9)
  where "∃x. φ x ≡ ∃φ"

definition fThat :: "('a ⇒ bool) ⇒ 'a" ("I")
  where "IΦ ≡ if ∃x. E x ∧ Φ x ∧ (∀y. (E y ∧ Φ y) ⟶ (y = x))
              then THE x. E x ∧ Φ x
              else e"
definition fThatBinder:: "('a ⇒ bool) ⇒ 'a" (binder "I" [8]9)
  where "Ix. φ x ≡ Iφ"
```

However, predicates need special treatment in negative free logic. Since it is not possible to directly redefine applications in Isabelle/HOL in an easy way, we have to resort to a little trick: We create a dummy prefix operator to use with a predicate for forcing the negative evaluation of it. Two of such operators for exemplary predicates of a specific type, one for a unary predicate and one for a secondary predicate, are given below.

```
definition fPredicate1 :: "('a ⇒ bool) ⇒ 'a ⇒ bool" ("ⁿ")
  where "ⁿP x ≡ E x ∧ P x"
definition fPredicate2 :: "('a ⇒ 'b ⇒ bool) ⇒ 'a ⇒ 'b ⇒ bool" ("ⁿⁿ")
  where "ⁿⁿP x y ≡ E x ∧ E y ∧ P x y"
```

Of course, for each type of predicate in a from NgFHOL translated HOL term the corresponding operator must be introduced as shown above and remembered to be applied for the proper translation of a term.

For the last part of this section, we will analyze some simple formulas of negative free higher-order logic and thereby also see how predicates are translated accurately. First, formulas containing equality, i.e., $x = x$, are addressed. The following examples perform exactly as expected for a negative free logic:

```
lemma "x = x"
  nitpick [user_axioms=true, show_all, format=2]
  oops
```

```
Nitpick found a counterexample for card 'a = 3 and card i = 3.
```

```
consts fIndividual1 :: "i" ("i₁")
axiomatization where fUndefIndividual1Axiom: "¬(E i₁)"

lemma "i₁ = i₁"
  nitpick [user_axioms=true, show_all, format=2]
  oops
```

```
Nitpick found a counterexample for card i = 1.
```

```
consts fIndividual1 :: "i" ("i₁")
axiomatization where fUndefIndividual1Axiom: "¬(E i₁)"

lemma "¬(i₁ = i₁)"
  unfolding Defs using fNot_def fUndefIndividual1Axiom
  by auto
```

Negative semantics has the special property that it rejects universal instantiation, but recovers existential generalization. This can easily be deduced from the fact that Px can never be true if x is nondenoting. And indeed, Nitpick was able to find a countermodel for sUI while wUI and sEG are both valid:

```
lemma "(∀x. ⁿP x) ⟶ ⁿP x"
  nitpick [user_axioms=true, show_all, format=2]
  oops
```

```
Nitpick found a counterexample for card 'a = 3 and card i = 3:

  Free variables:
    P = (λx. _)(a₁ := False, a₂ := False, a₃ := True)
    x = a₃
  Constants:
    E = (λx. _)(a₁ := False, a₂ := False, a₃ := False)
    E = (λx. _)(i₁ := False, i₂ := False, i₃ := False)
    E = (λx. _)(True := True, False := True)
    e = i₃
    e = False
```

```
lemma "((∀x. (ⁿP x)) ∧ (E x)) ⟶ (ⁿP x)"
  by (metis fAnd_def fForallBinder_def fForall_def fImp_def fNot_def
    fOr_def)
```

```
lemma "ⁿP x ⟶ (∃x. ⁿP x)"
  unfolding Defs
  by auto
```

sEG is not only limited to predicates. The experiments below show the validity of some other instantiations of the principle.

```
lemma "(x = y) ⟶ (∃x. (x = y))"
  unfolding Defs
  by blast
```

```
lemma "(x ∨ y) ⟶ (∃x. (x ∨ y))"
  unfolding Defs using fTrueAxiom
  by auto
```

Obviously, `True` must be existing for the proof of the latter.

Appendix B.4 contains the here presented complete polymorphic encoding of the embedding, appendix B.5 the nonpolymorphic version of it.

4.3 On Embedding NtFHOL into HOL

Unfortunately, an embedding similarly intuitive as the preceding ones is not possible for free logic with neutral semantics. HOL defined on Henkin models has the axiom of Boolean extensionality built into it (cf. Benzmüller, Brown, et al. 2004), which does not allow for any

more truth values than one for truth and falsehood. Since it's crucial for neutral free logic to employ a third truth value, we are stuck here.

However, it is possible to encode some many-valued logics in Isabelle/HOL by pursuing a somewhat different strategy. More precisely, Steen and Benzmüller (2016) successfully embedded the sixteen-valued language SIXTEEN (cf. Shramko and Wansing 2012) into HOL. Their embedding rests on a set-theoretical approach, but the language considered is only a propositional logic without quantification and hence not useful for deriving an embedding of NtFHOL. Nonetheless, a similar approach could lead to the encoding of quantified trivalent logics as neutral free higher-order logic. Though to this date, the author is not aware of any literature covering quantificational many-valued logic ready to be embedded into classical first-order or higher-order logic, and developing one is not trivial at all. Because of these reasons, embedding free logic with neutral semantics could not be done so far.

4.4 On Embedding SFHOL into HOL

The method of supervaluations as described in section 3.2.2.4 involves extending the information available about nonexisting objects in all possible ways, the notion of supertruth is defined upon that. But supertruth is evaluated by quantifying over completions of models and variable assignments. It probably comes as no surprise that these quantifications cannot be shallowly embedded straight away. HOL does not provide a device for such unconventional evaluations. So, as for neutral free logics, since not only a quantified trivalent logic must be embedded, but also a new kind of quantification is needed, a shallow embedding of SFHOL cannot be accomplished in a trivial way. Yet Barba Escribá (2001) had a rather ingenious idea: He noticed modal flavor[43] within Bencivenga's semantics and proposed an embedding of supervaluational free logic into a free version of modal logic S4.1.[44] And, which is why this is mentioned here, the latter one can smoothly be embedded into HOL (cf. Benzmüller and Woltzenlogel Paleo 2015). Anyhow, there is still one problem with Barba Escribá's proposal: The free modal semantics he postulated is bivalent, and supervaluational semantics is obviously not (Lehmann 2002: 233). Barba Escribá nonetheless established the following theorem: A SFHOL formula s is supervaluationally valid if and only if its translation into free modal logic is valid in the class of all S4.1 models. In the following, we will investigate this discrepancy by encoding the suggested embedding.

Barba Escribá's target language is a negative free modal logic in which terms involving equality can be true even if they contain nondenoting terms. The underlying modal logic is S4.1, meaning that, with Kripke's (1963) possible world semantics, the accessibility relation is reflexive and transitive, and McKinsey's axiom $\Box\Diamond P \rightarrow \Diamond\Box P$ holds. Furthermore, we assume nested domains, i.e., if some world v is reachable from w, then the existence domain of v is bigger than or equal to the existence domain of w (cf. Barba Escribá 2001: 131–132). For the translation of a supervaluational term into this free modal language, in our context *free higher-order modal logic* (FHOML), only the evaluation of predicates needs to be changed. A SFHOL

[43] Bacigalupo (2017: 132n5) also saw evidence that Bencivenga wanted his models to be interpreted modally.

[44] Barba (1989) presented similar results before, but back then he imposed very restrictive conditions on the applied model theory by introducing so-called K-models that fixed an actual world. The approach taken here is a much more general one.

formula Px is true if and only if the FHOML formula $(E!\,x \to Px) \wedge (\neg E!\,x \to \Box\Diamond Px)$ is true. The motivation behind this idea is simple: If the argument to some atomic formula containing a predicate fails to exist at the actual world, then, for the formula to be true, for all worlds reachable from the actual world there must exist a world reachable from that world where the argument exists and the formula is true. In short, there must be some reachable world where $E!\,s$ and Ps are both true (if not already true in the actual world). For the embedding of this translation, we first have to encode FHOML into HOL. Benzmüller and Woltzenlogel Paleo (2015) explored a solution for embedding *higher-order modal logic* (HOML) into HOL by lifting HOML terms into equivalent world-dependent HOL terms. Following their suggestion, we introduce for the embedding of FHOML into HOL, in addition to the base type for the individuals, `i`, a new type for possible worlds, `w`:

```
typedecl i
typedecl w
type_synonym μ = "w ⇒ bool"
```

Then, FHOML terms can be identified with certain Isabelle/HOL terms of type `w ⇒ bool` such that these terms are evaluated dependent on the world under consideration. The type `w ⇒ bool` is abbreviated in the encoding as μ. In the upcoming definitions, the prefix 'fm' stands for 'free modal'. Note that this time the definitions are nonpolymorphic.

```
definition fmIdentity :: "i ⇒ i ⇒ μ" (infixr "=fm" 56)
  where "φ =fm ψ ≡ λw. φ = ψ"

definition fmNot :: "μ ⇒ μ" ("¬fm_" [52]53)
  where "¬fmφ ≡ λw. ¬(φ w)"
definition fmOr :: "μ ⇒ μ ⇒ μ" (infixr "Vfm" 51)
  where "φ Vfm ψ ≡ λw. φ w ∨ ψ w"

definition fmAnd :: "μ ⇒ μ ⇒ μ" (infixr "∧fm" 52)
  where "φ ∧fm ψ ≡ ¬fm(¬fmφ Vfm ¬fmψ)"
definition fmImp :: "μ ⇒ μ ⇒ μ" (infixr "⟶fm" 49)
  where "φ ⟶fm ψ ≡ ¬fmφ Vfm ψ"
definition fmEquiv :: "μ ⇒ μ ⇒ μ" (infixr "↔fm" 50)
  where "φ ↔fm ψ ≡ φ ⟶fm ψ ∧fm ψ ⟶fm φ"
```

For defining the universal and existential quantifiers, we first give a nonpolymorphic existence predicate and use it as a guard as usual.

```
consts fExistenceI :: "i ⇒ μ" ("Ei")

definition fmForallI :: "(i ⇒ μ) ⇒ μ" ("∀ifm")
  where "∀ifmΦ ≡ λw. ∀x. Ei x w ⟶ Φ x w"
```

```
definition fmForallIBinder:: "(i ⇒ μ) ⇒ μ" (binder "∀ⁱ_fm" [8]9)
  where "∀ⁱ_fmx. φ x ≡ ∀ⁱ_fmφ"

definition fmExistsI :: "(i ⇒ μ) ⇒ μ" ("∃ⁱ_fm")
  where "∃ⁱ_fmΦ ≡ ¬_fm(∀ⁱ_fm(λy. ¬_fm(Φ y)))"
definition fmExistsIBinder :: "(i ⇒ μ) ⇒ μ" (binder "∃ⁱ_fm" [8]9)
  where "∃ⁱ_fmx. φ x ≡ ∃ⁱ_fmφ"
```

Predicates are, as mentioned previously, evaluated as in negative free logic. We therefore provide a dummy prefix operator below, as explained in section 4.2.2.

```
definition fmPredicateI :: "(i ⇒ μ) ⇒ i ⇒ μ" ("^fm")
  where "^fmP x ≡ λw. Eⁱ x w ∧ P x w"
```

Next, we have to define the accessibility relation `r` and impose conditions on it like reflexivity, transitivity, and an equivalent to McKinsey's axiom (cf. Barba Escribá 2001: 131).

```
consts r :: "w ⇒ w ⇒ bool" (infixr "r" 53)

abbreviation reflexive :: "bool"
  where "reflexive ≡ ∀x. x r x"
abbreviation transitive :: "bool"
  where "transitive ≡ ∀x y z. (x r y) ∧ (y r z) ⟶ (x r z)"
abbreviation mcKinseysAxiom :: "bool"
  where "mcKinseysAxiom ≡ ∀x. ∃y. (x r y) ∧ (∀z. (y r z) ⟶ y = z)"
axiomatization where S41:
  "reflexive ∧ transitive ∧ mcKinseysAxiom"
```

The nested domains property is also introduced:

```
axiomatization where nestedDomains:
  "∀x y. x r y ⟶ (∀z. Eⁱ z x ⟶ Eⁱ z y)"
```

To finish up the embedding of FHOML, these are the definitions of the modal operators □ and ◇:

```
definition fmBox :: "μ ⇒ μ" ("□_" [52]53)
  where "□φ ≡ λw. ∀v. w r v ⟶ φ v"

definition fmDia :: "μ ⇒ μ" ("◇_" [52]53)
  where "◇φ ≡ ¬_fm(□(¬_fmφ))"
```

We are now ready to embed SFHOL. Since the translation mainly effects predicates, we will give the other definitions without comments. The prefix 's' stands for 'supervaluational'.

```
definition sIdentity :: "i ⇒ i ⇒ μ" (infixr "=ₛ" 56)
 where φ" =ₛ ψ ≡ φ =fm ψ"

definition sNot :: "μ ⇒ μ" ("¬ₛ_" [52]53)
  where "¬ₛφ ≡ ¬fmφ"
definition sOr :: "μ ⇒ μ ⇒ μ" (infixr "Vₛ" 51)
  where "φ Vₛ ψ ≡ φ Vfm ψ"

definition sAnd :: "μ ⇒ μ ⇒ μ" (infixr "Λₛ" 52)
  where "φ Λₛ ψ ≡ ¬ₛ(¬ₛφ Vₛ ¬ₛψ)"
definition sImp :: "μ ⇒ μ ⇒ μ" (infixr "⟶ₛ" 49)
  where "φ ⟶ₛ ψ ≡ ¬ₛφ Vₛ ψ"
definition sEquiv :: "μ ⇒ μ ⇒ μ" (infixr "↔ₛ" 50)
  where "φ ↔ₛ ψ ≡ φ ⟶ₛ ψ Λₛ ψ ⟶ₛ φ"

definition sForallI :: "(i ⇒ μ) ⇒ μ" ("∀ⁱₛ")
  where "∀ⁱₛΦ ≡ ∀ⁱfmx. Φ x"
definition sForallIBinder:: "(i ⇒ μ) ⇒ μ" (binder "∀ⁱₛ" [8]9)
  where "∀ⁱₛx. φ x ≡ ∀ⁱₛφ"

definition sExistsI :: "(i ⇒ μ) ⇒ μ" ("∃ⁱₛ")
  where "∃ⁱₛΦ ≡ ¬ₛ(∀ⁱₛ(λy. ¬ₛ(Φ y)))"
definition sExistsIBinder :: "(i ⇒ μ) ⇒ μ" (binder "∃ⁱₛ" [8]9)
  where "∃ⁱₛx. φ x ≡ ∃ⁱₛφ"
```

Predicates are translated exactly as described in the introduction to this section:

```
definition sPredicateI :: "(i ⇒ μ) ⇒ i ⇒ μ" ("ˢ")
  where "ˢP x ≡ (Eⁱ x ⟶fm (ᶠᵐP x))
                Λfm (¬fm(Eⁱ x) ⟶fm (□(◇(ᶠᵐP x))))"
```

According to Barba Escribá (2001: 135), we need to ensure that the formula is valid in the class of all S4.1 models. For validity in a model class, the respective formulas have to be true in all worlds (cf. Gasquet, Herzig, Said, and Schwarzentruber 2013: 44–46). Hence, we implement validity as shown beneath.

```
definition sValid :: "μ ⇒ bool" ("⌊_⌋ₛ" [7]8)
  where "⌊φ⌋ₛ ≡ ∀w. φ w"
```

For testing that we indeed have a free logic, we check if the principles of universal instantiation and existential generalization fail to hold. As can be seen hereupon, Nitpick was able to find countermodels for both principles whereas their weaker analogs could be proved.

```
lemma "⌊(∀ˢⁱx. ˢP x) →ₛ (ˢP x)⌋ₛ"
  nitpick [user_axioms=true, show_all, format=2]
  oops
```

```
Nitpick found a counterexample for card i = 2 and card w = 1.
```

```
lemma "⌊(ˢP x) →ₛ (∃ˢⁱx. ˢP x)⌋ₛ"
  nitpick [user_axioms=true, show_all, format=2]
  oops
```

```
Nitpick found a counterexample for card i = 2 and card w = 1.
```

```
lemma "⌊((∀ˢⁱx. (ˢP x)) ∧ₛ (Eⁱ x)) →ₛ (ˢP x)⌋ₛ"
  unfolding Defs
  by blast

lemma "⌊((ˢP x) ∧ₛ (Eⁱ x)) →ₛ (∃ˢⁱx. ˢP x)⌋ₛ"
  unfolding Defs
  by blast
```

At this point, it is important to highlight that, although we started from a negative free logic, it seems that we ended up with a positive one. Existential generalization is a principle that holds in free logic with negative semantics, and since it doesn't hold here, we must have a positive setting.

To investigate this embedding further, we will have a closer look at some special formula, namely $Px \vee \neg Px$, where x is nonexisting. It is provable, and the model Nitpick returned for it is the following:

```
consts sIndividual1 :: "i" ("i₁")
axiomatization where sUndefIndividual1Axiom: "∃w. ¬(E¹ i₁ w)"

lemma "(⌊(ˢP i₁) ∨ₛ (¬ₛ(ˢP i₁))⌋ₛ)"
   unfolding Defs
   nitpick [satisfy, user_axioms=true, show_all, format=2]
   by blast
```

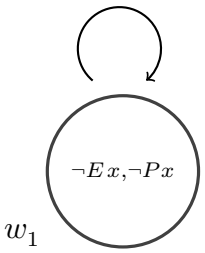

Figure 8: Example for a supervaluational free model with one world

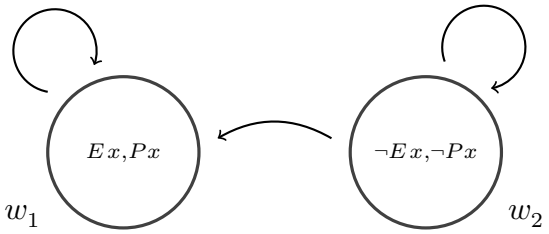

Figure 9: Example for a supervaluational free model with two worlds

```
Nitpick found a model for card i = 1 and card w = 1:

  Free variable:
    P = (λx. _)((i₁, w₁) := False)
  Skolem constants:
    λw. v = (λx. _)(w₁ := w₁)
    λw v. v = (λx. _)((w₁, w₁) := w₁)
  Constants:
    Eⁱ = (λx. _)((i₁, w₁) := False)
    i₁ = i₁
    (r) = (λx. _)((w₁, w₁) := True)
```

Nitpick found a model with one world and one individual, see fig. 8. This model is indeed reasonable, since $E!\,x \to \neg Px$ is true in FHOML because $\neg E!\,x$ is true and $\neg E!\,s \to \Box\Diamond\neg Ps$ is true because $\neg E!\,x$ and $\neg Px$ are both true in w_1, and w_1 is the only world reachable from w_1. Therefore, the SFHOL formula $\neg Px$ is true, and hence the formula $Px \vee \neg Px$, too. But, since x is nonexisting, the FHOML formula $\neg Px$ should technically be truth-valueless in w_1, and not true or false. So this is not really a model we would expect for a true supervaluational semantics. When asking for a model with two worlds, Nitpick showed the following one.

```
consts sIndividual1 :: "i" ("i₁")
axiomatization where sUndefIndividual1Axiom: "∃w. ¬(E¹ i₁ w)"

lemma "(⌊(ˢP i₁) ∨ₛ (¬ₛ(ˢP i₁))⌋ₛ)"
   unfolding Defs
   nitpick [satisfy, user_axioms=true, show_all, format=2, card i=1,
     card w=2]
   by blast
```

```
Nitpick found a model for card i = 1 and card w = 2:

  Free variable:
    P = (λx. _)((i₁, w₁) := True, (i₁, w₂) := False)
  Skolem constants:
    λw. v = (λx. _)(w₁ := w₁, w₂ := w₂)
    λw v. v = (λx. _)
                ((w₁, w₁) := w₂, (w₁, w₂) := w₂, (w₂, w₁) := w₁, (w₂, w₂) := w₁)
  Constants:
    Eⁱ = (λx. _)((i₁, w₁) := True, (i₁, w₂) := False)
    i₁ = i₁
    (r) =
      (λx. _)
      ((w₁, w₁) := True, (w₁, w₂) := False, (w₂, w₁) := True, (w₂, w₂) := True)
```

See fig. 9 for an illustration of this model. Once again it can easily be seen that the SFHOL formula $Px \vee \neg Px$ is true for both worlds. And this model now has a structure that is typical for a modally interpreted supervaluational semantics: Reachable worlds increase information of other worlds. World w_1 incorporates more information than world w_2. Thus, the approach seems to actually work somehow. However, it is not yet fully refined and important technical details have not been thought through to the end.[45] Without a neutral free logic as basis, we just end up with a positive free logic with unnecessary exaggerated evaluations. But, as also acknowledged by Lambert (2001a: 239–240), it is apparently a very promising starting point towards a shallow semantical embedding of supervaluational free higher-order logic.

[45] Lehmann (2002: 233) made similiar critical remarks, but referred to Barba's earlier approach. The critisim is nevertheless also applicable here.

5 Experimental Application to Prior's Theorem

Exploring how some disturbing results in higher-order modal logic revealed through Prior (1961), Kaplan (1995), and others can be avoided, Bacon et al. (2016) ended up arguing for free logic. Within this section, we want to reconstruct some of the results established by Bacon et al. by encoding them into Isabelle/HOL, and thereby also see how well interactive and automated theorem provers operate in this context. First of all, we analyze Prior's theorem based on the language and model theory these authors provided and verify two of the propositions given in their paper. Then, we try to obtain comparable results with the embeddings developed in this thesis for positive and negative free higher-order logic.

5.1 Implementing the Set-Theoretical Approach

As expounded in section 3.1, the semantics given by Bacon et al. (2016) relies on set theory where solely worlds are taken as primitive. In order to verify the results of the paper using their models, we first have to implement some basic set-theoretical concepts in Isabelle/HOL before being able to implement their language.[46] The central idea here is that a set can be represented by a characteristic function, a predicate that maps elements of the same type to truth values. An element is in a set if and only if the predicate representing the set maps the element in question to truth, i.e., $x \in S$ if and only if $S\,x$ (Benzmüller and Andrews 2019). Therefore, the set-theoretical concepts membership, union, set difference and set inclusion for sets containing elements of type w (for worlds) can be formalized in HOL as seen hereafter (cf. Benzmüller and Miller 2014: 9).

$$
\begin{aligned}
\in_{w\to(w\to o)\to o} &:= \lambda X_w.\, \lambda C_{w\to o}.\, C\,X \\
\cup_{(w\to o)\to(w\to o)\to(w\to o)} &:= \lambda A_{w\to o}.\, \lambda B_{w\to o}.\, \lambda X_w.\, A\,X \vee B\,X \\
\setminus_{(w\to o)\to(w\to o)\to(w\to o)} &:= \lambda A_{w\to o}.\, \lambda B_{w\to o}.\, \lambda X_w.\, A\,X \wedge \neg(B\,X) \\
\subseteq_{(w\to o)\to(w\to o)\to o} &:= \lambda A_{w\to o}.\, \lambda B_{w\to o}.\, \forall(\lambda X_w.\, A\,X \to B\,X)\,.
\end{aligned}
$$

For providing an equivalent formalization of these concepts in Isabelle/HOL, we introduce a new base type `w` and abbreviate the predicate that represents the set of elements of type `w` with `wSet`:

```
typedecl w
type_synonym wSet = "w ⇒ bool"
type_synonym wSetSet = "(w ⇒ bool) ⇒ bool"
```

[46] Isabelle/HOL provides a massive built-in set theory for import, which could be used instead (cf. Nipkow, Paulson, and Wenzel 2019b: 122–156). But to keep things as transparent as possible, we chose to implement it on our own.

The predicate `wSetSet` represents a set of sets of elements of type `w` to model subsets of the powerset of worlds. Furthermore, we define two different types of memberships, one for testing if a world is in the set `wSet` and one for testing if a set `wSet` is a member of the set of sets `wSetSet`. We will distinguish them using special subscripts as displayed below.

```
definition wSetMember :: "w ⇒ wSet ⇒ bool" (infixr "∈{}" 53)
  where "x ∈{} S ≡ S x"
definition wSetSetMember :: "wSet ⇒ wSetSet ⇒ bool" (infixr "∈{{}}" 53)
  where "x ∈{{}} S ≡ S x"
```

Union, set difference and set inclusion are accordingly to the initial considerations defined like that:

```
definition wSetUnion :: "wSet ⇒ wSet ⇒ wSet" (infixr "∪" 50)
  where "A ∪ B ≡ λx. A x ∨ B x"
definition wSetOther :: "wSet ⇒ wSet ⇒ wSet" (infixr "-" 50)
  where "A - B ≡ λx. A x ∧ ¬(B x)"
definition wSetSubsetEq :: "wSet ⇒ wSet ⇒ bool" (infixr "⊆" 50)
  where "A ⊆ B ≡ ∀x. A x ⟶ B x"
```

We further introduce three fairly handy constants: `W`, corresponding to Bacon, Hawthorne, and Uzquiano's set W, is the set containing all worlds, and **{}** and **{{}}** are the empty sets of type `wSet` and `wSetSet`, respectively. The individual characteristics of these constants are ensured by the underneath given axiomatizations.

```
consts W :: "wSet"
axiomatization where defW: "∀x. x ∈{} W"

consts emptySet :: "wSet" ("{}")
axiomatization where defEmptySet: "∀x. ¬(x ∈{} {})"

consts emptySetSet :: "wSetSet" ("{{}}")
axiomatization where defEmptySetSet: "∀x. ¬(x ∈{{}} {{}})"
```

Since Bacon et al. (2016: 10) are concerned exclusively with Kripke-style models that validate S5, we need to introduce another Isabelle/HOL constant symbolizing their accessibility relation R and constrain this relation to be an equivalence relation, i.e., reflexive, symmetric and transitive, and, moreover, also universal. We will again achieve this through an axiomatization:

```
consts r :: "w ⇒ w ⇒ bool" (infixr "r" 53)
```

```
abbreviation reflexive :: "bool"
  where "reflexive ≡ ∀x. x r x"
abbreviation symmetric :: "bool"
  where "symmetric ≡ ∀x y. x r y ⟶ y r x"
abbreviation transitive :: "bool"
  where "transitive ≡ ∀x y z. (x r y) ∧ (y r z) ⟶ (x r z)"
abbreviation universal :: "bool"
  where "universal ≡ ∀x y. x r y"

axiomatization where S5:
  "reflexive ∧ symmetric ∧ transitive ∧ universal"
```

Based on this newly defined relation `r`, the set $R(w) = \{x \mid R\,w\,x\}$ can be represented as the hereunder given abbreviation.

```
abbreviation R :: "w ⇒ wSet" ("R_"[52]53)
  where "R w ≡ λx. w r x"
```

Recall, that the existence domain $D(w)$ is a set of sets of worlds, a set of propositions, existing at world w. Intuitively, this coincides with the following function mapping worlds to sets of sets of worlds.

```
consts fmExistenceDomains :: "w ⇒ wSetSet" ("D")
```

Now we provided all auxiliary definitions needed to transfer the evaluation of formulas into Isabelle/HOL. To make the forthcoming definitions more comprehensible, the formal evaluation function of the language under consideration shall be repeated here.

$$
\begin{aligned}
[\![\, x \,]\!]_v &:= \begin{cases} v(x) & \text{if } x \text{ is a propositional variable} \\ [\![\, x \,]\!]_v & \text{if } x \text{ is an atomic sentence letter} \end{cases} \\
[\![\, s = t \,]\!]_v &:= \begin{cases} W & \text{if } [\![\, s \,]\!]_v = [\![\, t \,]\!]_v \\ \varnothing & \text{else} \end{cases} \\
[\![\, \neg s \,]\!]_v &:= W \setminus [\![\, s \,]\!]_v \\
[\![\, s \vee t \,]\!]_v &:= [\![\, s \,]\!]_v \cup [\![\, t \,]\!]_v \\
[\![\, \Box\, s \,]\!]_v &:= \{w \in W \mid R(w) \subseteq [\![\, s \,]\!]_v\} \\
[\![\, Q\, s \,]\!]_v &:= \{w \in W \mid [\![\, s \,]\!]_v \in |Q|(w)\} \\
[\![\, \forall x.\, s \,]\!]_v &:= \{w \in W \mid w \in [\![\, s \,]\!]_u, \forall u \text{ such that } u[x]v \text{ and } u(x) \in D(w)\}\,.
\end{aligned}
$$

Unless for the universal quantifier, the translation is almost one to one and given below.

```
definition fmIdentity :: "wSet ⇒ wSet ⇒ wSet" (infixr "=" 56)
  where "φ = ψ ≡ if (φ = ψ) then W else {}"

definition fmNot :: "wSet ⇒ wSet" ("¬_" [52]53)
  where "¬φ ≡ W - φ"
definition fmOr :: "wSet ⇒ wSet ⇒ wSet" (infixr "∨" 51)
  where "φ ∨ ψ ≡ φ ∪ ψ"

definition fmBox :: "wSet ⇒ wSet" ("□_" [52]53)
  where "□φ ≡ λw. (R w) ⊆ φ"

consts Qex :: "w ⇒ wSetSet" ("|Q|")
definition Qin :: "wSet ⇒ wSet" ("Q")
  where "Qφ ≡ λw. φ ∈{{}} |Q|(w)"
```

The quantified formula $\forall x.\, s$ is satisfied by variable assignment v if and only if s is satisfied by every assignment u such that u and v agree except, possibly, on which set of worlds they have assigned to x, and $u(x) \in D(w)$. This translates into the HOL formula $\forall x.\, s\, x$ by ensuring that for all $x \in D(w)$, $s\, x$ is satisfied. Hence, universal quantification and its binder notation have the following embedded form:

```
definition fmForall :: "(wSet ⇒ wSet) ⇒ wSet" ("∀")
  where "∀Φ ≡ λw. ∀x. x ∈{{}} (D w) ⟶ w ∈{} (Φ x)"
definition fmForallB:: "(wSet ⇒ wSet) ⇒ wSet" (binder "∀" [8]9)
  where "∀x. φ x ≡ ∀φ"
```

The remaining logical connectives and the existential quantifier are conventionally abbreviated and embedded as shown hereafter.

```
definition fmAnd :: "wSet ⇒ wSet ⇒ wSet" (infixr "∧" 52)
  where "φ ∧ ψ ≡ ¬(¬φ ∨ ¬ψ)"
definition fmImp :: "wSet ⇒ wSet ⇒ wSet" (infixr "⟶" 49)
  where "φ ⟶ ψ ≡ ¬φ ∨ ψ"
definition fmEquiv :: "wSet ⇒ wSet ⇒ wSet" (infixr "↔" 50)
  where "φ ↔ ψ ≡ (φ ⟶ ψ) ∧ (ψ ⟶ φ)"

definition fmDia :: "wSet ⇒ wSet" ("◇_" [52]53)
  where "◇φ ≡ ¬(□¬φ)"
```

```
definition fmExists :: "(wSet ⇒ wSet) ⇒ wSet" ("∃")
  where "∃Φ ≡ ¬(∀(λy. ¬(Φ y)))"
definition fmExistsB:: "(σ ⇒ σ) ⇒ σ" (binder "∃" [8]9)
  where "∃x. φ x ≡ ∃φ"
```

Bacon, Hawthorne, and Uzquiano's semantics identifies a term with the set of worlds where it is true, which means, that generally valid terms are associated with the set W (cf. Starr 2019: 2.1). Thus, we redefine the notion of validity as follows:

```
definition fmValid :: "wSet ⇒ bool" ("⌊_⌋" [7]8)
  where "⌊φ⌋ ≡ φ = W"
```

The embedding in its entirety, without explanatory notes, can be found in appendix B.1.

For the remainder of this section, the results gained by Bacon et al. (2016) will be discussed and verfied in Isabelle/HOL. But first, we recite Prior's theorem:

$$Q\,\forall p.\,(Q\,p \to \neg p) \;\to\; \exists p.\,(Q\,p \wedge p) \wedge \exists p.\,(Q\,p \wedge \neg p)\,.$$

Found on the model theory mainly presented in section 3.1 and encoded here, Bacon et al. (2016) laid out a number of claims that threaten a certain family of intensional paradoxes within the classical setting, a family, to which Prior's theorem also belongs. The following two propositions, among others, were the first claims they derived:[47]

Proposition 1. There exists a finite countermodel with possibly empty constant existence domains for Prior's theorem.

Proposition 2. There exists a finite countermodel with nonempty constant existence domains containing truth and falsehood for Prior's theorem.

For proposition 1, the countermodel given by the authors has these properties: $W = \{w\}$, $R = \{(w, w)\}$, $D(w) = \varnothing$ and $|Q|(w) = \{\varnothing, \{w\}\}$. It is illustrated in fig. 10. Note that the underlying logic is universal since $D(w)$ is empty. We transcribe Prior's theorem into Isabelle/HOL using our embedding and try to verify the countermodel from above by applying the model finder Nitpick:

[47] The propositions as stated here differ from the original ones given by Bacon et al. (2016). Instead of only providing countermodels, they fixed three more rather reasonable formulas related to Prior's and Kaplan's paradoxes and gave models for those. This was shortened in order not to go beyond the scope. Consider the referred literature for more details and a comprehensive study of these models.

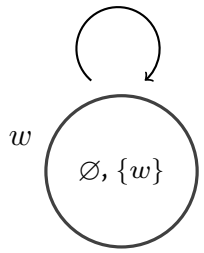

Figure 10: Illustration of a countermodel for Prior's theorem with one world

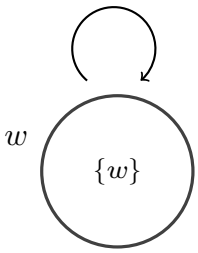

Figure 11: Illustration of another countermodel for Prior's theorem with one world

```
lemma "⌊(Q (∀p. (Q p → ¬p))) → ((∃p. Q p ∧ p) ∧ (∃p. Q p ∧ ¬p))⌋"
  unfolding Defs
  nitpick [user_axioms=true, format=2, show_all]
  oops
```

```
Nitpick found a counterexample for card w = 1:

  Constants:
    |Q| =
      (λx. _)
      ((w₁, (λx. _)(w₁ := True)) := True,
       (w₁, (λx. _)(w₁ := False)) := False)
    W = (λx. _)(w₁ := True)
    {} = (λx. _)(w₁ := False)
    {{}} = (λx. _)((λx. _)(w₁ := True) := False, (λx. _)(w₁ := False) := False)
    D =
      (λx. _)
      ((w₁, (λx. _)(w₁ := True)) := False,
       (w₁, (λx. _)(w₁ := False)) := False)
    R = (λx. _)((w₁, w₁) := True)
```

Nitpick found a countermodel with $W = \{w\}$, $R = \{(w, w)\}$ and $D(w) = \varnothing$, but $|Q|(w) = \{\{w\}\}$. This countermodel, also seen in fig. 11, is, although perfectly valid, not exactly the countermodel Bacon et al. (2016) had in mind. To reproduce their countermodel, we introduce two axiomatizations that explicitly define that (i) the existence domain of each world is empty and that (ii) W and $\varnothing$ are in the extension of Q at some world, and try it again:

```
axiomatization where Ax1: "∀x. (D x) = {{}}"
axiomatization where Ax2: "∃x. W ∈{{}} |Q|(x) ∧ {} ∈{{}} |Q|(x)"
```

```
lemma "⌊(Q (∀p. (Q p → ¬p))) → ((∃p. Q p ∧ p) ∧ (∃p. Q p ∧ ¬p))⌋"
  unfolding Defs
  nitpick [user_axioms=true, format=2, show_all]
  oops
```

```
Nitpick found a counterexample for card w = 1:

  Constants:
    |Q| =
      (λx. _)
      ((w₁, (λx. _)(w₁ := True)) := True,
       (w₁, (λx. _)(w₁ := False)) := True)
    W = (λx. _)(w₁ := True)
    {} = (λx. _)(w₁ := False)
    {{}} = (λx. _)((λx. _)(w₁ := True) := False, (λx. _)(w₁ := False) := False)
    D =
      (λx. _)
      ((w₁, (λx. _)(w₁ := True)) := False,
       (w₁, (λx. _)(w₁ := False)) := False)
    R = (λx. _)((w₁, w₁) := True)
```

This time, the resulting countermodel agrees with $W = \{w\}$, $R = \{(w, w)\}$, $D(w) = \varnothing$ and $|Q|(w) = \{\varnothing, \{w\}\}$. Therefore, Bacon, Hawthorne, and Uzquiano's countermodel for proposition 1 could be successfully verified.

Proposition 2 was proven by the following countermodel: Let $W = \{x, y, z\}$ and $D(x) = D(y) = D(z) = \{\{x, y, z\}, \varnothing\}$, i.e., only W and $\varnothing$ exist at each world. Then, we define $|Q|(x) = \{W\}$, $|Q|(y) = \{\varnothing\}$ and $|Q|(z) = \mathcal{P}W \setminus \{W, \varnothing\}$ with $\mathcal{P}W$ being the powerset of W. This countermodel is depicted in fig. 12. To find it via Nitpick, we again need to introduce multiple axiomatizations. First of all, we must fulfill the precondition that all existence domains have to be nonempty. Then, we willingly axiomatize that (i) all existence domains are equal, (ii) W and $\varnothing$ are in the existence domains of all worlds, and (iii) nothing else is. Further, (iv) there must exist a world where only W is in the extension of Q and (v) also a world where only $\varnothing$ is in the extension of Q. Additionally, (vi) there must be a world where anything else except W and $\varnothing$ is in the extension of Q. Let's see:

```
axiomatization where fmNonemptyExistenceDomains: "∀w. (D w) ≠ {{}}"

axiomatization where Ax1: "∀x y. (D x) = (D y)"
axiomatization where Ax2: "∀x. W ∈{{}} (D x) ∧ {} ∈{{}} (D x)"
axiomatization where Ax3:
  "∀x y. (y ∈{{}} (D x)) ⟶ ((y = {}) ∨ (y = W))"
axiomatization where Ax4:
  "∃x. W ∈{{}} |Q|(x) ∧ (∀y. y ∈{{}} |Q|(x) ⟶ y = W)"
```

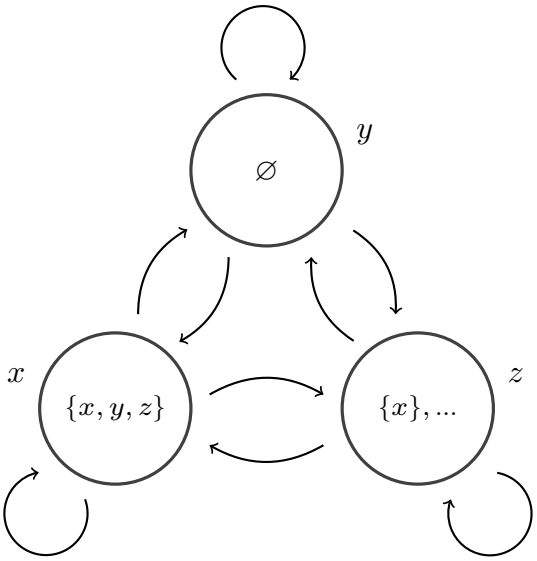

Figure 12: Illustration of a countermodel for Prior's theorem with three worlds

```
axiomatization where Ax5:
  "∃x. {} ∈{{}} |Q|(x) ∧ (∀y. y ∈{{}} |Q|(x) ⟶ y = {})"
axiomatization where Ax6:
  "∃x y. ((y ≠ {}) ∧ (y ≠ W)) ⟶ (y ∈{{}} |Q|(x))"

lemma "⌊(Q (∀p. (Q p → ¬p))) → ((∃p. Q p ∧ p) ∧ (∃p. Q p ∧ ¬p))⌋"
  unfolding Defs
  nitpick [user_axioms=true, format=2, show_all, card=3]
  oops
```

```
Nitpick found a counterexample for card w = 3:

  Constants:
    |Q| =
      (λx. _)
      ((w₁, (λx. _)(w₁ := True, w₂ := True, w₃ := True)) := True,
       (w₁, (λx. _)(w₁ := True, w₂ := True, w₃ := False)) := False,
       (w₁, (λx. _)(w₁ := True, w₂ := False, w₃ := True)) := False,
       (w₁, (λx. _)(w₁ := True, w₂ := False, w₃ := False)) := False,
       (w₁, (λx. _)(w₁ := False, w₂ := True, w₃ := True)) := False,
       (w₁, (λx. _)(w₁ := False, w₂ := True, w₃ := False)) := False,
       (w₁, (λx. _)(w₁ := False, w₂ := False, w₃ := True)) := False,
       (w₁, (λx. _)(w₁ := False, w₂ := False, w₃ := False)) := False,
```

```
(w₂, (λx. _)(w₁ := True, w₂ := True, w₃ := True)) := False,
(w₂, (λx. _)(w₁ := True, w₂ := True, w₃ := False)) := False,
(w₂, (λx. _)(w₁ := True, w₂ := False, w₃ := True)) := False,
(w₂, (λx. _)(w₁ := True, w₂ := False, w₃ := False)) := False,
(w₂, (λx. _)(w₁ := False, w₂ := True, w₃ := True)) := False,
(w₂, (λx. _)(w₁ := False, w₂ := True, w₃ := False)) := False,
(w₂, (λx. _)(w₁ := False, w₂ := False, w₃ := True)) := False,
(w₂, (λx. _)(w₁ := False, w₂ := False, w₃ := False)) := True,
(w₃, (λx. _)(w₁ := True, w₂ := True, w₃ := True)) := False,
(w₃, (λx. _)(w₁ := True, w₂ := True, w₃ := False)) := True,
(w₃, (λx. _)(w₁ := True, w₂ := False, w₃ := True)) := True,
(w₃, (λx. _)(w₁ := True, w₂ := False, w₃ := False)) := True,
(w₃, (λx. _)(w₁ := False, w₂ := True, w₃ := True)) := True,
(w₃, (λx. _)(w₁ := False, w₂ := True, w₃ := False)) := True,
(w₃, (λx. _)(w₁ := False, w₂ := False, w₃ := True)) := True,
(w₃, (λx. _)(w₁ := False, w₂ := False, w₃ := False)) := False)
…
```

The counterexample was truncated due to its length, the complete version can be found in appendix C.1. The short excerpt, however, is enough to see that, with $x = w_1$, $y = w_2$ and $z = w_3$, the model found by Nitpick is exactly the one the authors came up with. That is, both propositions claiming finite countermodels could be successfully verified with our encoded embedding. Unfortunately, all other even more reasonable countermodels provided by Bacon et al. (2016) are infinite ones, but any model finder available within Isabelle/HOL, including Nitpick, are finite model finders. To date, as far as the author is aware of, there are no infinite model finders available for Isabelle/HOL or other automated theorem proving systems. Therefore, the remaining propositions cannot yet be verified.

In the next section, we will double-check these results using the embeddings for PFHOL and NgFHOL given in this thesis.

5.2 Formalization using the Embeddings of FHOL into HOL

Obviously, there is a significant difference between the logic developed by Bacon et al. (2016) and the definitions presented in the whole section 3.2 in regards to free higher-order logic. While the first allows an infinite number of truth values, the second one must get along with a maximum of three truth values. Nevertheless, we will try to work with the results of the previous section by transferring them as closely as possible to a more restricted free logical setting as we imposed it.

Applying Sledgehammer together with the embedding of PFHOL into HOL where the domain of quantification can be empty onto Prior's theorem, we end up with the following result:

```
axiomatization where fTrueAxiom: "E True"
axiomatization where fFalseAxiom: "E False"

lemma "(Q (∀p. (Q p ⟶ (¬p)))) ⟶ ((∃p. Q p ∧ p) ∧ (∃p. Q p ∧ (¬p)))"
  using Defs
  by (smt fFalseAxiom fTrueAxiom)
```

The theorem is valid. But as we can cleary see, the theorem is proved using the axioms `fTrueAxiom` and `fFalseAxiom` stipulating that both truth values are defined. We try it again without these:

```
lemma "(Q (∀p. (Q p ⟶ (¬p)))) ⟶ ((∃p. Q p ∧ p) ∧ (∃p. Q p ∧ (¬p)))"
  nitpick [user_axioms=true, show_all, format=2]
  oops
```

```
Nitpick found a counterexample for card i = 3:

  Free variable:
    Q = (λx. _)(True := True, False := True)
  Constants:
    E = (λx. _)(True := True, False := False)
    E = (λx. _)(i₁ := False, i₂ := False, i₃ := True)
    e = i₂
    e = False
```

This time Nitpick found a countermodel. Observe, that in this countermodel one of the two truth values is undefined, namly `False`. Enforcing both truth values to be undefined would essentially lead to the countermodel provided for proposition 1 by Bacon et al. (2016). However, on a metaphysical level, it is highly questionable to shift one or even both truth values into the undefined range. Bacon et al. (2016) themselves did not find this approach for overcoming the paradox very promising and have constructed other countermodels as a substitute, e.g., the one for proposition 2, which, unfortunately, cannot be reproduced with the eligible embeddings of this thesis. For these countermodels, at least three different truth values are required, and we have not yet been able to adequately embed trivalent or other many-valued free higher-order logics.

As a next step, we apply Sledgehammer onto Prior's theorem again, now together with the embedding of NgFHOL into HOL where again the quantification domain can be empty:

```
lemma "(ⁿQ (∀p. (ⁿQ p ⟶ (¬p)))) ⟶ ((∃p. ⁿQ p ∧ p) ∧ (∃p. ⁿQ p ∧ (¬p)))"
  unfolding Defs
  by smt
```

The theorem is once again valid, and in this case not even the axioms are necessary for the proof. One can show that Prior's theorem holds in a free logic with negative semantics without specifying if the truth values are defined or not.[48] Due to the nature of negative free logic this is not really surprising. But what is surprising is that although the principle of universal instantiation does not apply in negative semantics, Prior's theorem holds here, contradicting the hypothesis that sUI alone is key to such intentional paradoxes as supposed by Bacon et al. (2016: 4). However, there is a crucial difference between positive and negative free logic that could explain this and that was probably not considered by Bacon, Hawthorne, and Uzquiano. Strong existential generalization is not valid in positive semantics, but it is indeed valid in negative semantics. Therefore, one might have reason to think that the invalidity of the principle of existential generalization plays a greater role in this context than Bacon et al. (2016) may have thought about.

[48] From this we can definitely infer that the free logic of Bacon et al. (2016) is a positive one.

6 Discussion and Conclusion

After we initially set the fundamentals, free higher-order logic and its characteristics of non-existent objects and partial functions was represented in an adequately modified version of simple type theory. The semantics of choice in this case were the less common inner-outer dual domains, which indeed turned out to be a pleasant all-in-one solution for addressing all different nuances of free logic. A key point of this approach was that partiality was only simulated instead of inherently accomodating it, such that a classical environment valuable for embeddings into HOL could be maintained. The other generic properties of each free logical variant could also very well be reflected on STT with inner-outer dual domains eventually leading to all metaphysically interesting aspects being covered. The only exception was supervaluational semantics, which was quite hard to achieve. Bencivenga's original semantics had to undergo major reconstruction, especially with regard to completions, and the higher-order definitions are now somewhat more complex than they were before. Any negative implications for, e.g., soundness and completeness proofs cannot be excluded and have yet to be investigated. Nevertheless, at this stage, an at least formally acceptable solution ready to work on with has been found. Despite these small issues, the great advantage of the dual domain approach became apparent in the course of this thesis when the embeddings for positive and negative semantics were realized very easily. In this very final part, we will bring together all the findings of the previous sections and draw an overall conclusion. The discussion on the technical implementation of the embeddings is hereby followed by a reflection on the relevance of the topic in terms of actuality and practicability. In this context, related and further research work will be mentioned, as well as an outlook on the continuation of the project given.

As far as the technical implementation is concerned, it can now be said that it was certainly as simple as promised in the introduction to section 4. The embeddings suggested in this section were easily accomplished, uncomplicated to understand, and effortlessly implemented into Isabelle/HOL. So, SSEs in general, and concretely the one for positive semantics, come off well as expected. If one wants to name one shortcoming, then, that the shallow semantical embedding for negative semantics does not feel as natural as one is used to, for example, from the alike embedding of positive free logic, or even modal logic. The modal embedding of Benzmüller and Woltzenlogel Paleo (2015) appears lightweight and is hardly noticeable in practice when used for modal logical theories within Isabelle/HOL. That, unfortunately, cannot be said about the SSE of NgFHOL. In free logics, as opposed to modal logics, it is mainly the unconventional application handling that has to be taken into account in the embeddings. An embedding for negative semantics needs to make use of a workaround, an additional operator for predicates to be placed before any application of arguments so that the right applicational behavior is ensured. However, this approach is prone to errors, as the operator can easily be forgotten or overlooked and hence might cause wrongly evaluated terms. Applications cannot be directly controlled in Isabelle/HOL through a shallow embedding, so we have probably already chosen the best possible way to implement an encoding without switching to a deep embedding. On the other hand, this might be of less importance anyway. Although Scales (1969) and Feferman (1992) presented strong arguments for this family of

logics, due to a lack of applications, free logic with negative semantics received little support from Gumb and Lambert (1997) (cf. Lambert 2001b: 262), Bacon (2013: 5–6, 2019), and others. Bacon, in particular, rather argued at length, in detail and most convincingly in favor of a positive semantics, which, in turn, can be embedded very nicely. The research of Bacon et al. (2016) studied in this thesis is also based on positive free logic explicitly exploiting the refusal of strong existential generalization. Priest (2008: 467) was an equally keen advocate of positive semantics while McKinsey (2020), conversely, preferred neutral free logic for solid reasons. Equally popular are supervaluations, which were often named by many as the standard theory of vagueness (cf. Varzi 2007) and considered as a true alternative for classical logic (Paul n.d.). All these recent publications show that today free logic is more actual than ever, and the urge for automation, specifically with regard to positive and neutral semantics, is fully justified. It is therefore even more crucial that neutral free logic including supervaluational semantics could not yet be successfully embedded into HOL. The embedding approach of Barba Escribá (2001) encoded in section 4.4 is a good start, but still needs further development.

However, free logic is not only an interesting topic in itself. One thing that the reader might have noticed while reading this thesis is that free logic and modal logic highly intertwine. Already Kripke (1963) stressed that a 'correct' possible world semantics for quantified modal logic must support per-world individual domains over which the quantifiers range at those worlds, so-called varying domains (Braüner and Ghilardi 2007: 557–558), instead of only permitting a single fixed domain for all worlds. Nowadays it is generally accepted to build modal logic on top of free logic, rather than on a classical logic, whereas positive and negative semantics predominate (cf. Nolt 2018: 5.3). E.g., Garson (1991, 2001: 278), who chose a positive free logic for his modal system, persuasively argued that free semantics are the most adequate choice for modal logics with varying domains, because otherwise formulas like $Px \to \exists x.\, Px$ with the existential quantifier ranging over all objects in the domain can no longer be valid at certain worlds (cf. Lehmann 2001: 50–51; Schurz 2006: 469; McKinsey 2020: 68). Stalnaker (1994), on the contrary, showed that combining modal logic with a negative free logic would not be sufficient since some natural-looking formulas would undesirably fail to be validated in Kripke models (cf. Bacon 2013: 6). McKinsey (2020: 84–86) took a completely different approach and proposed a formal semantics for quantified modal logic that is based on a a free logic with neutral semantics.

The last related topic to be considered here are many-valued logics, in which particularly neutral free logic plays a major role. A Fregean conception of truth offers the possibility to distinguish between various truth values, not just two or three (Shramko and Wansing 2012: 41). Free logic with neutral semantics is a trivalent logic, but, of course, also logics with more than three truth values, finitely or infinitely many, are possible. While logics with infinitely many truth values have become known as fuzzy logics (cf. Cintula, Fermüller, and Noguera 2017), most many-valued logics usually have a fixed number of truth values as, for instance, four-valued logic or sixteen-valued logic (cf. Shramko and Wansing 2012). Numerous extensions of classical bivalent logic to many-valued logic have been studied to meet the demand for extra truth values, and an important driving force behind that was partiality as it is in free logic. In this process, supervaluations were thought-through as an alternative to many-valued logic by Urquhart (1986: 113), and Smith (2008: 87–93) investigated how a combination of a many-valued logic and supervaluational semantics could work. The key to

an improved definition of free higher-order logic in a neutral and supervaluational sense, that can neatly be embedded into HOL, is probably somewhere in between many-valued logic and set theory, similarly to Bacon, Hawthorne, and Uzquiano's well-performing approach[49] seen in section 5.1. Though a lot of work has already been done in this field, even more research in possible collaboration with automated theorem provers is needed to finally come up with a definition suitable for an embedding of NtFHOL and SFHOL into HOL.

The number of truth values is not the only variable in free logic. As indicated in the introduction, a much more obvious augmentation is opting for multiple states of existence. Not all nonexistent objects are equally nonexistent. This already starts with the fact that there is a difference between nonexistence and undefinedness. Reiterating the examples from the introduction, a fictional character might be nonexistent but still owns designated properties. Contrarily, some distinct, yet uncharted planet might have a certain probability of existence but is absolutely undefined. And between these examples there are countless shades of gradation. One inventive idea is that a free logical evaluation of terms, of predicates and functions, should take into consideration each object's state of nonexistence. This entails, for example, multiple domains and matching existence predicates per domain. See fig. 13 and 14 for an exemplary scheme of how two such outer domains might be combined. Clearly, more finely granulated truth values would complement this idea. Fourman and Scott (1979) already submitted a similar idea in another context. However, it remains an open problem to elaborate and refine such a very special semantical theory.

Figure 13: Example for two outer domains where one contains the other

Figure 14: Example for two overlapping outer domains

Nondenoting terms have always been an important subject for logic in general, and free logic in particular. With free logic spreading more and more into different areas of application like artificial intelligence, cognitive science, and linguistics, and connecting with modal and many-valued logics, new requirements and challenges are forming. Dealing with these problems

[49] Some critical voices, such as Bueno, Menzel, and Zalta (2014) and Merricks (2015), expressed the opinion that taking sets of propositions as worlds is not very promising. However, the idea could still somehow help develop such new theories.

might probably lead to new innovative semantics for free logic, which are surely easier to establish with the help of interactive and automated theorem provers. This thesis laid a first foundation of relevant embeddings on which one can and should now continue to build.

References

Andrews, Peter B. (1972a). General Models and Extensionality. In: *Journal of Symbolic Logic* 37.2, pp. 395–397.

– (1972b). General Models, Descriptions, and Choice in Type Theory. In: *Journal of Symbolic Logic* 37.2, pp. 385–394.

Antonelli, G. Aldo (2000). Proto-Semantics for Positive Free Logic. In: *Journal of Philosophical Logic* 29.3, pp. 277–294.

– (2007). Free Quantification and Logical Invariance. In: *Rivista di Estetica* 33.1, pp. 61–73.

Bacigalupo, Giuliano (2017). *A Study on Existence. Two Approaches and a Deflationist Compromise*. Cambridge Scholars Publishing.

Bacon, Andrew (2013). Quantificational Logic and Empty Names. In: *Philosophers' Imprint* 13.24.

– (2019). "Opacity and Paradox".

Bacon, Andrew, Hawthorne, John, and Uzquiano, Gabriel (2016). Higher-Order Free Logic and the Prior-Kaplan Paradox. In: *Canadian Journal of Philosophy* 46.4-5, pp. 493–541. doi: 10.1080/00455091.2016.1201387.

Barba Escribá, Juan L. (2001). "Supervaluational Free Logic and the Logic of Information Growth". In: *New Essays in Free Logic. In Honour of Karel Lambert*. Ed. by Edgar Morscher and Alexander Hieke. Vol. 23. Dordrecht: Springer Netherlands, pp. 127–146. doi: 10.1007/978-94-015-9761-6_7.

Barba, Juan L. (1989). A Modal Version of Free Logic. In: *Topoi* 8.2, pp. 131–135. doi: 10.1007/BF00141368.

Beall, J. C., Glanzberg, Michael, and Ripley, David (2019). "Liar Paradox". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Winter 2019. Metaphysics Research Lab, Stanford University.

Beeson, Michael J. (1985). *Foundations of Constructive Mathematics. Metamathematical Studies*. Ergebnisse der Mathematik und ihrer Grenzgebiete, 3. Folge, Band 6. Berlin, Heidelberg: Springer.

Bencivenga, Ermanno (1981). "Free Semantics". In: *Italian Studies in the Philosophy of Science*. Ed. by Maria Luisa Dalla Chiara. Vol. 47. Boston Studies in the Philosophy of Science. Dordrecht: Springer Netherlands, pp. 31–48.

– (1984). Supervaluations and Theories. In: *Grazer Philosophische Studien* 21.1, pp. 89–98.

– (1986). "Free Logics". In: *Handbook of Philosophical Logic. Volume III: Alternatives in Classical Logic*. Ed. by Dov M. Gabbay and Franz Günthner. Dordrecht: Springer Netherlands, pp. 373–426.

Benzmüller, Christoph (2017). *Universal Reasoning, Rational Argumentation and Human-Machine Interaction*. Tech. rep. CoRR. url: http://arxiv.org/abs/1703.09620.

– (2019). "Universal (Meta-)Logical Reasoning: Recent Successes".

Benzmüller, Christoph and Andrews, Peter B. (2019). "Church's Type Theory". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Summer 2019. Metaphysics Research Lab, Stanford University.

Benzmüller, Christoph, Brown, Chad E., and Kohlhase, Michael (2004). Higher-Order Semantics and Extensionality. In: *Journal of Symbolic Logic* 69.4, pp. 1027–1088.

Benzmüller, Christoph and Miller, Dale (2014). "Automation of Higher-Order Logic". In: *Computational Logic*. Ed. by Dov M. Gabbay, Jörg H. Siekmann, and John Woods. Vol. 9. Handbook of the History of Logic. North Holland: Elsevier, pp. 215–254.

Benzmüller, Christoph and Scott, Dana (2016). "Automating Free Logic in Isabelle/HOL". In: *Mathematical Software – ICMS 2016. 5th International Conference*. Ed. by Gert-Martin Greuel, Thorsten Koch, Peter Paule, and Andrew Sommese. Vol. 9725. Lecture Notes in Computer Science. Berlin, Germany: Springer Berlin, Heidelberg, pp. 43–50. doi: 10.1007/978-3-319-42432-3_6.

– (2019). Automating Free Logic in HOL, with an Experimental Application in Category Theory. In: *Journal of Automated Reasoning*, pp. 1–20.

Benzmüller, Christoph and Woltzenlogel Paleo, Bruno (2013). Gödel's God in Isabelle/HOL. In: *Archive of Formal Proofs*, pp. 1–5.

– (2014). "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, pp. 93–98. doi: 10.3233/978-1-61499-419-0-93.

– (2015). "Higher-Order Modal Logics: Automation and Applications". In: *Reasoning Web 2015*. Ed. by Adrian Paschke and Wolfgang Faber. Lecture Notes in Computer Science 9203. Berlin, Heidelberg: Springer, pp. 32–74. doi: 10.1007/978-3-319-21768-0_2. url: http://christoph-benzmueller.de/papers/C46.pdf.

– (2016). "The Inconsistency in Gödel's Ontological Argument: A Success Story for AI in Metaphysics". In: *IJCAI 2016*. Ed. by Subbarao Kambhampati. Vol. 1–3. AAAI Press, pp. 936–942.

Blanchette, Jasmin Christian, Böhme, Sascha, and Paulson, Lawrence C. (2013). Extending Sledgehammer with SMT Solvers. In: *Journal of Automated Reasoning* 51.1, pp. 109–128. doi: 10.1007/s10817-013-9278-5.

Blanchette, Jasmin Christian and Nipkow, Tobias (2010). "Nitpick: A Counterexample Generator for Higher-Order Logic Based on a Relational Model Finder". In: *First International Conference on Interactive Theorem Proving. Proceedings*. Ed. by Matt Kaufmann and Lawrence C. Paulson. Vol. 6172. Lecture Notes in Computer Science. Edinburgh, United Kingdom: Springer Berlin, Heidelberg, pp. 131–146. doi: 10.1007/978-3-642-14052-5_11.

Bove, Ana, Krauss, Alexander, and Sozeau, Matthieu (2016). Partiality and Recursion in Interactive Theorem Provers – An Overview. In: *Mathematical Structures in Computer Science* 26.1, pp. 38–88. doi: 10.1017/S0960129514000115.

Braüner, Torben and Ghilardi, Silvio (2007). "First-Order Modal Logic". In: *Handbook of Modal Logic*. Ed. by Patrick Blackburn, Johan Van Benthem, and Frank Wolter. Vol. 3. Studies in Logic and Practical Reasoning. Elsevier, pp. 549–620.

Brown, Chad E. (2012). "Satallax: An Automatic Higher-Order Prover". In: *Proceedings of the 6th International Joint Conference on Automated Reasoning*. IJCAR'12. Berlin, Heidelberg: Springer, pp. 111–117. doi: 10.1007/978-3-642-31365-3_11.

Bueno, Otávio, Menzel, Christopher, and Zalta, Edward N. (2014). Worlds and Propositions Set Free. In: *Erkenntnis* 79.4, pp. 797–820. doi: 10.1007/s10670-013-9565-x.

Burge, Tyler (1974). Truth and Singular Terms. In: *Noûs* 8.4, pp. 309–325. doi: 10.2307/2214437.

Carpenter, Bob (1997). *Type-Logical Semantics*. MIT Press.

Church, Alonzo (1940). A Formulation of the Simple Theory of Types. In: *Journal of Symbolic Logic* 5.2, pp. 56–68.

Cintula, Petr, Fermüller, Christian G., and Noguera, Carles (2017). "Fuzzy Logic". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Fall 2017. Metaphysics Research Lab, Stanford University.

Ciucci, Davide and Dubois, Didier (2013). A Map of Dependencies among Three-Valued Logics. In: *Information Sciences* 250, pp. 162–177. doi: 10.1016/j.ins.2013.06.040.

Cocchiarella, Nino B. (1966). (Abstract) A Logic of Actual and Possible Objects. In: *Journal of Symbolic Logic* 31.2, pp. 688–689.

Farmer, William M. (1990). A Partial Functions Version of Church's Simple Theory of Types. In: *Journal of Symbolic Logic* 55, pp. 1269–1291.

– (1993). A Simple Type Theory with Partial Functions and Subtypes. In: *Annals of Pure and Applied Logic* 64.3, pp. 211–240. doi: 10.1016/0168-0072(93)90144-3.

– (2004). "Formalizing Undefinedness Arising in Calculus". In: *Automated Reasoning. Second International Joint Conference*. Ed. by David Basin and Michaël Rusinowitch. Vol. 3097. Lecture Notes in Computer Science. Cork, Ireland: Springer Berlin, Heidelberg, pp. 475–489. doi: 10.1007/978-3-540-25984-8_35.

– (2008). The Seven Virtues of Simple Type Theory. In: *Journal of Applied Logic* 6.3, pp. 267–286. url: http://dblp.uni-trier.de/db/journals/japll/japll6.html#Farmer08.

Farmer, William M. and Guttman, Joshua D. (2000). A Set Theory with Support for Partial Functions. In: *Studia Logica* 66.1, pp. 59–78.

Feferman, Solomon (1992). "Logics for Termination and Correctness of Functional Programs". In: *Logic from Computer Science*. Ed. by Yiannis N. Moschovakis. New York, NY: Springer, pp. 95–127.

Fourman, Michael and Scott, Dana (1979). "Sheaves and Logic". In: *Applications of Sheaves*. Ed. by Michael Fourman, Christopher Mulvey, and Dana Scott. Vol. 753. Lecture Notes in Mathematics. Berlin, Heidelberg: Springer, pp. 302–401.

Gabbay, Dov M. (1996). *Labelled Deductive Systems*. Vol. 1. Oxford Logic Guides. Oxford: Clarendon Press.

Garson, James (1991). "Applications of Free Logic to Quantified Intensional Logic". In: *Philosophical Applications of Free Logic*. Ed. by Karel Lambert. Oxford University Press, pp. 110–142.

– (2001). "Quantification in Modal Logic". In: *Handbook of Philosophical Logic. Volume III: Alternatives in Classical Logic*. Ed. by Dov M. Gabbay and Franz Günthner. Dordrecht: Springer Netherlands, pp. 267–323. Repr. of "Quantification in Modal Logic". In: *Handbook of Philosophical Logic. Volume II: Extensions of Classical Logic*. Ed. by Dov M. Gabbay and Franz Günthner. Dordrecht: D. Reidel, 1984, pp. 249–307.

Gasquet, O., Herzig, A., Said, B., and Schwarzentruber, F. (2013). *Kripke's Worlds. An Introduction to Modal Logics via Tableaux*. Studies in Universal Logic. Basel: Springer.

Gödel, Kurt (1931). Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. In: *Monatshefte für Mathematik und Physik* 38.1, pp. 173–198.

Grundy, Jim and Newey, Malcolm C., eds. (1998). *Theorem Proving in Higher Order Logics: Emerging Trends. 11th International Conference, TPHOLs'98, Canberra, Australia, September 27 – October 1, 1998, Proceedings*. Vol. 1479. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer.

Gumb, Raymond D. (2001). "Free Logic in Program Specification and Verification". In: *New Essays in Free Logic. In Honour of Karel Lambert*. Ed. by Edgar Morscher and Alexander Hieke. Vol. 23. Dordrecht: Springer Netherlands, pp. 157–93. doi: 10.1007/978-94-015-9761-6_9.

Gumb, Raymond D. and Lambert, Karel (1997). Definitions in Nonstrict Positive Free Logic. In: *Modern Logic* 7.1, pp. 25–55.

Henkin, Leon (1950). Completeness in the Theory of Types. In: *Journal of Symbolic Logic* 15.2, pp. 81–91.

– (1963). A Theory of Propositional Types. In: *Fundamenta Mathematicae* 52, pp. 323–334.

– (1975). Identity as a Logical Primitive. In: *Philosophia* 5.1-2, pp. 31–45. doi: 10.1007/BF02380832.

Hintikka, Jaako (1959). Existential Presuppositions and Existential Commitments. In: *Journal of Philosophy* 56.3, pp. 125–137.

Jaśkowski, Stanisław (1934). On the Rules of Suppositions in Formal Logic. In: *Studia Logica* 1, pp. 5–32. Repr. in Storrs McCall, ed. *Polish Logic. 1920–1939*. Oxford University Press, 1967, pp. 232–258.

Kaplan, David (1995). "A Problem in Possible Worlds Semantics". In: *Modality, Morality and Belief: Essays in Honor of Ruth Barcan Marcus*. Ed. by Walter Sinnott-Armstrong, Diana Raffman, and Nicholas Asher. Cambridge University Press, pp. 41–52.

Kleene, Stephen Cole (1952). *Introduction to Metamathematics*. Wolters-Noordhoff.

Kratzer, Angelika (2012). *Modals and Conditionals. New and Revised Perspectives*. Oxford Studies in Theoretical Linguistics. Oxford: Oxford University Press.

Kripke, Saul A. (1963). Semantical Considerations on Modal Logic. In: *Acta Philosophica Fennica* 16, pp. 83–94.

Lambert, Karel (1960). The Definition of E! in Free Logic. In: *Abstracts: The International Congress for Logic, Methodology and Philosophy of Science*.

– (1963). Existential Import Revisited. In: *Notre Dame Journal of Formal Logic* 4.4, pp. 288–292. doi: 10.1305/ndjfl/1093957655.

– (1964). Notes on "E!" IV: A Reduction in Free Quantification Theory with Identity and Descriptions. In: *Philosophical Studies: An International Journal for Philosophy in the Analytic Tradition* 15.6, pp. 85–88.

– (1967). Free Logic and the Concept of Existence. In: *Notre Dame Journal of Formal Logic* 1-2.8, pp. 133–144. doi: 10.1305/ndjfl/1093956251.

– (1972). Notes on Free Description Theory: Some Philosophical Issues and Consequences. In: *Journal of Philosophical Logic* 1.2, pp. 184–191.

– (1983). *Meinong and the Principle of Independence. Its Place in Meinong's Theory of Objects and its Significance in Contemporary Philosophical Logic*. Modern European Philosophy. Cambridge: Cambridge University Pres.

– (1991a). "A Theory about Logical Theories of 'Expressions of the Form "The So and So", where "The" is in the Singular'". In: *Erkenntnis Orientated: A Centennial Volume for Rudolf*

Carnap and Hans Reichenbach. Ed. by Wolfgang Spohn. Dordrecht: Springer Netherlands, pp. 337–346. doi: 10.1007/978-94-011-3490-3_18.

Lambert, Karel (1991b). *Philosophical Applications of Free Logic*. Oxford University Press.

– (1997). *Free Logics: Their Foundations, Character and Some Applications Thereof*. Sankt Augustin: Academia Verlag.

– (2001a). "Comments". In: *New Essays in Free Logic. In Honour of Karel Lambert*. Ed. by Edgar Morscher and Alexander Hieke. Vol. 23. Dordrecht: Springer Netherlands, pp. 239–252. doi: 10.1007/978-94-015-9761-6_12.

– (2001b). From Predication to Programming. In: *Minds and Machines* 11.2, pp. 257–265.

Lambert, Karel and van Fraassen, Bas C. (1972). *Derivation and Counterexample: An Introduction to Philosophical Logic*. Dickenson Publishing Company.

Leblanc, Hugues (1980). *Existence, Truth, and Probability*. State University of New York Press.

Leblanc, Hugues and Thomason, Richmond H. (1968). Completeness Theorems for Some Presupposition-Free Logics. In: *Fundamenta Mathematicae* 62.2, pp. 125–164.

Lehmann, Scott (1994). Strict Fregean Free Logic. In: *Journal of Philosophical Logic* 23.3, pp. 307–336. doi: 10.1007/BF01048484.

– (2001). "'No Input, No Output' Logic". In: *New Essays in Free Logic. In Honour of Karel Lambert*. Ed. by Edgar Morscher and Alexander Hieke. Vol. 23. Dordrecht: Springer Netherlands, pp. 147–155. doi: 10.1007/978-94-015-9761-6_8.

– (2002). "More Free Logic". In: *Handbook of Philosophical Logic. Volume V*. Ed. by Dov M. Gabbay and Franz Günthner. Dordrecht: Springer Netherlands, pp. 197–259.

Lewis, David K. (1973). *Counterfactuals*. Oxford: Wiley-Blackwell.

– (1986). *On the Plurality of Worlds*. Oxford: Wiley-Blackwell.

Lipton, James and Nieva, Susana (2007). "Higher-Order Logic Programming Languages with Constraints: A Semantics". In: *Typed Lambda Calculi and Applications*. Ed. by Rocca Della and Ronchi Simona. Berlin, Heidelberg: Springer, pp. 272–289. doi: 10.1007/978-3-540-73228-0_20.

McKinsey, Michael (2020). *Consequences of Reference Failure*. Routledge Studies in Contemporary Philosophy. New York: Routledge.

Merricks, Trenton (2015). "Propositions Are Not Sets of Possible Worlds". In: *Propositions*. Oxford: Oxford University Press, pp. 82–120.

Meyer, Robert K., Bencivenga, Ermanno, and Lambert, Karel (1982). The Ineliminability of E! in Free Quantification Theory without Identity. In: *Journal of Philosophical Logic* 11.2, pp. 229–231.

Meyer, Robert K. and Lambert, Karel (1968). Universally Free Logic and Standard Quantification Theory. In: *Journal of Symbolic Logic* 33.1, pp. 8–26. doi: 10.2307/2270048.

Morscher, Edgar and Simons, Peter (2001). Free Logic: A Fifty-Year Past and an Open Future. In: *Applied Logic Series* 23. Ed. by Edgar Morscher and Alexander Hieke, pp. 1–34. doi: 10.1007/978-94-015-9761-6_1.

Nipkow, Tobias (1989). Equational Reasoning in Isabelle. In: *Science of Computer Programming* 12, pp. 123–149.

Nipkow, Tobias, Paulson, Lawrence C., and Wenzel, Markus (2002). *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*. Vol. 2283. Lecture Notes in Computer Science. Updated in 2019a. Berlin, Heidelberg: Springer.

Nipkow, Tobias, Paulson, Lawrence C., and Wenzel, Markus (2019a). *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*. http://isabelle.in.tum.de/doc/tutorial.pdf. Last accessed on December 30, 2019.

– (2019b). *Isabelle/HOL – Higher-Order Logic*. https://isabelle.in.tum.de/dist/library/HOL/HOL/document.pdf, https://isabelle.in.tum.de/dist/library/HOL/HOL/Set.html. Last accessed on February 05, 2020.

Nolt, John (2018). "Free Logic". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Fall 2018. Metaphysics Research Lab, Stanford University.

Ohlbach, Hans Jürgen, Nonnengart, Andreas, de Rijke, Maarten, and Gabbay, Dov M. (2001). "Encoding Two-Valued Nonclassical Logics in Classical Logic". In: *Handbook of Automated Reasoning*. Ed. by Alan Robinson and Andrei Voronkov. Vol. 2. North Holland: Elsevier, pp. 1403–1486.

Paśniczek, Jacek (2001). "Can Meinongian Logic be Free?" In: *New Essays in Free Logic. In Honour of Karel Lambert*. Ed. by Edgar Morscher and Alexander Hieke. Vol. 23. Dordrecht: Springer Netherlands, pp. 227–236. doi: 10.1007/978-94-015-9761-6_11.

Paul, Bornali (n.d.). "Proposal of Replacing Classical Logic with Free Logic for Reasoning with Non-Referring Names in Ordinary Discourse".

Paulson, Lawrence C. (1994). *A Generic Theorem Prover*. Vol. 828. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 73–87. doi: 10.1007/BFb0030541.

– (1999). A Generic Tableau Prover and its Integration with Isabelle. In: *Journal of Universal Computer Science* 5.3, pp. 73–87.

Paulson, Lawrence C. and Blanchette, Jasmin Christian (2015). Three Years of Experience with Sledgehammer, a Practical Link between Automatic and Interactive Theorem Provers. In: *Proceedings of the 8th International Workshop on the Implementation of Logics*, pp. 131–146.

Posy, Carl J. (2007). "Free Logics". In: *The Many Valued and Nonmonotonic Turn in Logic*. Ed. by Dov M. Gabbay and John Woods. Vol. 8. Handbook of the History of Logic. North Holland: Elsevier, pp. 633–680. doi: 10.1016/S1874-5857(07)80013-6.

Priest, Graham (2008). *An Introduction to Non-Classical Logic. From If to Is*. Cambridge Introductions to Philosophy. Cambridge University Press.

Prior, Arthur N. (1961). On a Family of Paradoxes. In: *Notre Dame Journal of Formal Logic* 2.1, pp. 16–32.

Quine, Willard Van Orman (1954). Quantification and the Empty Domain. In: *Journal of Symbolic Logic* 19.3, pp. 177–179.

Rami, Dolf (2014). Existence as a Property of Individuals. In: *Erkenntnis* 79.S3, pp. 503–523. doi: 10.1007/s10670-013-9505-9.

Read, Stephen (1995). *Thinking About Logic*. Oxford University Press.

Sainsbury, Richard M. (2005). "Names in Free Logical Truth Theory". In: *Thought, Reference, and Experience: Themes from the Philosophy of Gareth Evans*. Ed. by José Luis Bermúdez. Oxford: Oxford University Press, pp. 66–83.

Scales, Ronald D. (1969). "Attribution and Existence". PhD thesis. Ann Arbor, Michigan: University of California, Irvine.

Schock, Rolf (1964). Contributions to Syntax, Semantics, and the Philosophy of Science. In: *Notre Dame Journal of Formal Logic* 5.4, pp. 241–289. doi: 10.1305/ndjfl/1093957975.

– (1968). *Logics Without Existence Assumptions*. Stockholm: Almqvist & Wiksell.

Schurz, Gerhard (2006). "Alethic Modal Logics and Semantics". In: *A Companion to Philosophical Logic*. Ed. by Dale Jacquette. Blackwell Publishing Ltd. Chap. 29, pp. 442–477. doi: 10.1002/9780470996751.ch30.

Schütte, Kurt (1960). Syntactical and Semantical Properties of Simple Type Theory. In: *Journal of Symbolic Logic* 25.4, pp. 305–326. doi: 10.2307/2963525.

Scott, Dana (1967). "Existence and Description in Formal Logic". In: *Bertrand Russell: Philosopher of the Century*. Ed. by Ralph Schoenman. Repr. in Lambert 1991b, pp. 28–48. Boston: Little, Brown & Company, pp. 181–200.

– (1970). "Advice on Modal Logic". In: *Philosophical Problems in Logic. Some Recent Developments*. Ed. by Karel Lambert. Dordrecht: Springer Netherlands, pp. 143–173. doi: 10.1007/978-94-010-3272-8_7.

Shramko, Yaroslav and Wansing, Heinrich (2012). *Truth and Falsehood. An Inquiry into Generalized Logical Values*. Vol. 36. Trends in Logic. Dordrecht: Springer Netherlands. doi: 10.1007/978-94-007-0907-2.

Skryms, Brian (1968). Supervaluations: Identity, Existence, and Individual Concepts. In: *Journal of Philosophy* 65.16, pp. 477–483.

Smiley, Timothy (1960). Sense without Denotation. In: *Analysis* 20.6, pp. 125–135.

Smith, Nicholas J. J. (2008). *Vagueness and Degrees of Truth*. Oxford: Oxford University Press.

Stalnaker, Robert C. (1976). Possible Worlds. In: *Noûs* 10.1, pp. 65–75.

– (1984). *Inquiry*. Cambridge University Press.

– (1994). "The Interaction of Modality with Quantification and Identity". In: *Modality, Morality and Belief. Essays in Honor of Ruth Barcan Marcus*. Ed. by W. Sinnott-Armstrong, D. Raffman, and N. Asher. Cambridge: Cambridge University Press, pp. 12–28.

Starr, William (2019). "Counterfactuals". In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Fall 2019. Metaphysics Research Lab, Stanford University.

Steen, Alexander (2020). Extensional Paramodulation for Higher-Order Logic and Its Effective Implementation Leo-III. In: *KI - Künstliche Intelligenz* 34, pp. 105–108. doi: 10.1007/s13218-019-00628-8.

Steen, Alexander and Benzmüller, Christoph (2016). Sweet SIXTEEN: Automation via Embedding into Classical Higher-Order Logic. In: *Logic and Logical Philosophy* 25.4, pp. 535–554. doi: 10.12775/LLP.2016.021.

– (2018a). "System Demonstration: The Higher-Order Prover Leo-III". In: *ARQNL 2018. Automated Reasoning in Quantified Non-Classical Logics*. Ed. by Christoph Benzmüller and Jens Otten. Vol. 2095. CEUR Workshop Proceedings, http://ceur-ws.org, pp. 79–85. url: http://ceur-ws.org/Vol-2095/paper5.pdf.

– (2018b). "The Higher-Order Prover Leo-III". In: *Automated Reasoning. IJCAR 2018*. Ed. by Didier Galmiche, Stephan Schulz, and Roberto Sebastiani. Vol. 10900. Lecture Notes in Computer Science. Springer, pp. 108–116. doi: 10.1007/978-3-319-94205-6_8.

Tiemens, Lucca, Scott, Dana, Benzmüller, Christoph, and Benda, Miroslav (2019). "Computer-supported Exploration of a Categorical Axiomatization of Modeloids".

Urquhart, Alasdair (1986). "Many-Valued Logic". In: *Handbook of Philosophical Logic. Volume III: Alternatives in Classical Logic*. Ed. by Dov M. Gabbay and Franz Günthner. Dordrecht: Springer Netherlands, pp. 71–116.

van Fraassen, Bas C. (1966a). Singular Terms, Truth-Value Gaps, and Free Logic. In: *Journal of Philosophy* 63.17, pp. 481–495.

– (1966b). The Completeness of Free Logic. In: *Mathematical Logic Quarterly* 12.1, pp. 219–234. doi: 10.1002/malq.19660120117.

Varzi, Achille C. (2007). Supervaluationism and Its Logics. In: *Mind* 463, pp. 633–675.

Villadsen, Jørgen, Schlichtkrull, Anders, and Hess, Andreas Viktor (2015). "Meta-Logical Reasoning in Higher-Order Logic". In: LOGICA 2015 - 29th Annual International Symposia Devoted to Logic. Conference date: 15-06-2015 through 19-06-2015. url: http://logika.flu.cas.cz/en/logica-2015/logica-20152.

von Fintel, Kai and Heim, Irene (2011). *Intensional Semantics*. http://web.mit.edu/fintel/fintel-heim-intensional.pdf. Last accessed on January 20, 2020.

Walters, Lee (2014). "Singular Thought and The Nonexistent". PhD thesis. University College London.

Wenzel, Markus (1999). "Isar — A Generic Interpretative Approach to Readable Formal Proof Documents". In: *Theorem Proving in Higher Order Logics*. Ed. by Yves Bertot, Gilles Dowek, Laurent Théry, André Hirschowitz, and Christine Paulin. Berlin, Heidelberg: Springer, pp. 167–183.

Woodruff, Peter W. (1984). On Supervaluations in Free Logic. In: *Journal of Symbolic Logic* 49.3, pp. 943–950. doi: 10.2307/2274148.

Yeakel, Daniel (2015). An Argument For A Neutral Free Logic. Paper 1177. In: *Wayne State University Dissertations*.

List of Figures

A Proofs

A.1 Proof of Lemma 1

Lemma 1. For all PFHOL models M and PFHOL variable assignments g,

$$[\![s_\alpha]\!]^{M,g} = [\![[s_\alpha]]\!]^{M^!,g^!}.$$

Proof. The proof is by induction on the structure of s_α.

For $s_\alpha = P_\alpha$ and $s_\alpha = x_\alpha$, the following holds:

$$[\![P_\alpha]\!]^{M,g} = I(P_\alpha) = I^!(P_\alpha) = I^!([P_\alpha]) = [\![[P_\alpha]]\!]^{M^!,g^!}$$

and

$$[\![x_\alpha]\!]^{M,g} = g(x_\alpha) = g^!(x_\alpha) = g^!([x_\alpha]) = [\![[x_\alpha]]\!]^{M^!,g^!}.$$

This completes the base case.

For $s_\alpha = (E!_{\alpha\to o}\, s_\alpha)_o$, we have:

$$[\![(E!_{\alpha\to o}\, s_\alpha)_o]\!]^{M,g} = [\![E!_{\alpha\to o}]\!]^{M,g}([\![s_\alpha]\!]^{M,g}) = I(E!_{\alpha\to o})\,([\![s_\alpha]\!]^{M,g}).$$

With $I(E!_{\alpha\to o})\, d = ex(d) = I^!(E_{\alpha\to o})\, d$ for all $d \in D_\alpha = D^!_\alpha$, it is:

$$\begin{aligned} I(E!_{\alpha\to o})\,([\![s_\alpha]\!]^{M,g}) &= I^!(E_{\alpha\to o})\,([\![[s_\alpha]]\!]^{M^!,g^!}) = [\![E_{\alpha\to o}]\!]^{M^!,g^!}([\![[s_\alpha]]\!]^{M^!,g^!}) \\ &= [\![(E_{\alpha\to o}\,[s_\alpha])_o]\!]^{M^!,g^!} = [\![[(E!_{\alpha\to o}\, s_\alpha)_o]]\!]^{M^!,g^!}. \end{aligned}$$

For $s_\alpha = (s_{\alpha\to\beta}\, t_\alpha)_\beta$, we have:

$$\begin{aligned} [\![(s_{\alpha\to\beta}\, t_\alpha)_\beta]\!]^{M,g} &= [\![s_{\alpha\to\beta}]\!]^{M,g}\,([\![t_\alpha]\!]^{M,g}) = [\![[s_{\alpha\to\beta}]]\!]^{M^!,g^!}([\![[t_\alpha]]\!]^{M^!,g^!}) \\ &= [\![([s_{\alpha\to\beta}]\,[t_\alpha])_\beta]\!]^{M^!,g^!} = [\![[(s_{\alpha\to\beta}\, t_\alpha)_\beta]]\!]^{M^!,g^!}. \end{aligned}$$

For $s_\alpha = (\lambda x_\alpha .\, s_\beta)_{\alpha\to\beta}$, we have:

Let $[\![(\lambda x_\alpha .\, s_\beta)_{\alpha\to\beta}]\!]^{M,g}$ be the function f from D_α into D_β s.t. for all $d \in D_\alpha$:

$$f(d) = [\![s_\beta]\!]^{M,g[x\to d]}.$$

By induction hypothesis and since $D_\alpha = D^!_\alpha$ for each $\alpha \in \mathcal{T}$, f equals to the function f' from $D^!_\alpha$ into $D^!_\beta$ s.t. for all $d \in D^!_\alpha$:

$$f'(d) = [\![[s_\beta]]\!]^{M^!,g^![x\to d]}.$$

Then, $f' = [\![(\lambda x_\alpha .\, [\, s_\beta \,])_{\alpha\to\beta}]\!]^{M,g'} = [\![[\, (\lambda x_\alpha .\, s_\beta \,])_{\alpha\to\beta} \,]\!]^{M,g'}$.

For $s_\alpha = ((=^{F}_{\alpha\to\alpha\to o}\, s_\alpha)_{\alpha\to o}\, t_\alpha)_o$, we have:

$$\begin{aligned} [\![((=^{F}_{\alpha\to\alpha\to o}\, s_\alpha)_{\alpha\to o}\, t_\alpha)_o]\!]^{M,g} &= ([\![=^{F}_{\alpha\to\alpha\to o}]\!]^{M,g}([\![s_\alpha]\!]^{M,g}))\,([\![t_\alpha]\!]^{M,g}) \\ &= (I(=^{F}_{\alpha\to\alpha\to o})\,([\![s_\alpha]\!]^{M,g}))\,([\![t_\alpha]\!]^{M,g}). \end{aligned}$$

Since $(I(=^{F}_{\alpha\to\alpha\to o})\, d_1)\, d_2 = id(d_1, d_2) = (I'(=^{H}_{\alpha\to\alpha\to o})\, d_1)\, d_2$ for all $d_1, d_2 \in D_\alpha = D'_\alpha$,

$$\begin{aligned} (I(=^{F}_{\alpha\to\alpha\to o})\,([\![s_\alpha]\!]^{M,g}))\,([\![t_\alpha]\!]^{M,g}) &= (I'(=^{H}_{\alpha\to\alpha\to o})\,([\![s_\alpha]\!]^{M,g}))\,([\![t_\alpha]\!]^{M,g}) \\ &= ([\![=^{H}_{\alpha\to\alpha\to o}]\!]^{M,g'}([\![[\, s_\alpha \,]]\!]^{M,g'}))\,([\![[\, t_\alpha \,]]\!]^{M,g'}) \\ &= [\![((=^{H}_{\alpha\to\alpha\to o}\, [\, s_\alpha \,])_{\alpha\to o}\, [\, t_\alpha \,])_o]\!]^{M,g'} \\ &= [\![[\, ((=^{F}_{\alpha\to\alpha\to o}\, s_\alpha)_{\alpha\to o}\, t_\alpha)_o \,]]\!]^{M,g'}. \end{aligned}$$

The cases $s_\alpha = (\neg^{F}_{o\to o}\, s_o)_o$ and $s_\alpha = ((\wedge^{F}_{o\to o\to o}\, s_o)_{o\to o}\, t_o)_o$ are similiar to the case above and left out.

For $s_\alpha = (\forall^{F}_{(\alpha\to o)\to o}(\lambda x_\alpha .\, s_o)_{\alpha\to o})_o$, we have:

$$\begin{aligned} [\![(\forall^{F}_{(\alpha\to o)\to o}(\lambda x_\alpha .\, s_o)_{\alpha\to o})_o]\!]^{M,g} &= [\![\forall^{F}_{(\alpha\to o)\to o}]\!]^{M,g}\,([\![(\lambda x_\alpha .\, s_o)_{\alpha\to o}]\!]^{M,g}) \\ &= I(\forall^{F}_{(\alpha\to o)\to o})\,([\![(\lambda x_\alpha .\, s_o)_{\alpha\to o}]\!]^{M,g}). \end{aligned}$$

Let $f = [\![(\lambda x_\alpha .\, s_o)_{\alpha\to o}]\!]^{M,g}$. Then, $I(\forall^{F}_{(\alpha\to o)\to o})\, f = \mathrm{T}$ if and only if for all $d \in E_\alpha$: $f(d) = \mathrm{T}$. Obviously, for the function $f' = [\![(\lambda x_\alpha .\, ((E!_{\alpha\to o}\, x_\alpha)_o \to^{F}_{o\to o\to o} s_o)_o)_{\alpha\to o}]\!]^{M,g}$, we equally have that $I'(\forall^{H}_{(\alpha\to o)\to o})\, f' = \mathrm{T}$ if and only if for all $d \in D_\alpha$: $f'(d) = \mathrm{T}$. Also, by induction hypothesis,

$$\begin{aligned} &[\![(\lambda x_\alpha .\, ((E!_{\alpha\to o}\, x_\alpha)_o \to^{F}_{o\to o\to o} s_o)_o)_{\alpha\to o}]\!]^{M,g} \\ &\quad = [\![[\, (\lambda x_\alpha .\, ((E!_{\alpha\to o}\, x_\alpha)_o \to^{F}_{o\to o\to o} s_o)_o)_{\alpha\to o} \,]]\!]^{M,g'} \\ &\quad = [\![(\lambda x_\alpha .\, ((E_{\alpha\to o}\, x_\alpha)_o \to^{H}_{o\to o\to o} [\, s_o \,])_o)_{\alpha\to o}]\!]^{M,g'}. \end{aligned}$$

Therefore,

$$\begin{aligned} &I(\forall^{F}_{(\alpha\to o)\to o})\,([\![(\lambda x_\alpha .\, s_o)_{\alpha\to o}]\!]^{M,g}) \\ &\quad = I'(\forall^{H}_{(\alpha\to o)\to o})\,([\![(\lambda x_\alpha .\, ((E_{\alpha\to o}\, x_\alpha)_o \to^{H}_{o\to o\to o} [\, s_o \,])_o)_{\alpha\to o}]\!]^{M,g'}) \\ &\quad = [\![\forall^{H}_{(\alpha\to o)\to o}]\!]^{M,g'}\,([\![(\lambda x_\alpha .\, ((E_{\alpha\to o}\, x_\alpha)_o \to^{H}_{o\to o\to o} [\, s_o \,])_o)_{\alpha\to o}]\!]^{M,g'}) \\ &\quad = [\![(\forall^{H}_{(\alpha\to o)\to o}(\lambda x_\alpha .\, ((E_{\alpha\to o}\, x_\alpha)_o \to^{H}_{o\to o\to o} [\, s_o \,])_o)_{\alpha\to o})_o]\!]^{M,g'} \\ &\quad = [\![[\, (\forall^{F}_{(\alpha\to o)\to o}(\lambda x_\alpha .\, s_o)_{\alpha\to o})_o \,]]\!]^{M,g'}. \end{aligned}$$

For $s_\alpha = (\iota^{F}_{(\alpha\to o)\to\alpha}(\lambda x_\alpha.\, s_o)_{\alpha\to o})_\alpha$, we have:

$$\begin{aligned}[\![\,(\iota^{F}_{(\alpha\to o)\to\alpha}(\lambda x_\alpha.\, s_o)_{\alpha\to o})_\alpha\,]\!]^{M,g} &= [\![\,\iota^{F}_{(\alpha\to o)\to\alpha}\,]\!]^{M,g}\,([\![\,(\lambda x_\alpha.\, s_o)_{\alpha\to o}\,]\!]^{M,g}) \\ &= I(\iota^{F}_{(\alpha\to o)\to\alpha})\,([\![\,(\lambda x_\alpha.\, s_o)_{\alpha\to o}\,]\!]^{M,g}).\end{aligned}$$

Now, we need to distinguish two cases.

For the first case, we have $I(\iota^{F}_{(\alpha\to o)\to\alpha})\,([\![\,(\lambda x_\alpha.\, s_o)_{\alpha\to o}\,]\!]^{M,g}) = d \in E_\alpha$.

Let $f = [\![\,(\lambda x_\alpha.\, s_o)_{\alpha\to o}\,]\!]^{M,g}$. Then, there is a $d \in E_\alpha$ s.t. $f(d) = \mathrm{T}$ and for all $d' \in E_\alpha$: if $f(d') = \mathrm{T}$, then $d' = d$. Obviously, with $f' = [\![\,(\lambda x_\alpha.\, ((E!_{\alpha\to o}\, x_\alpha)_o \wedge^{F}_{o\to o\to o}\, s_o)_o)_{\alpha\to o}\,]\!]^{M,g}$, we equally have that $I'(\iota^{H}_{(\alpha\to o)\to\alpha})\, f' = d$ for the very same d as above. Furthermore, by induction hypothesis,

$$\begin{aligned}&[\![\,(\lambda x_\alpha.\, ((E!_{\alpha\to o}\, x_\alpha)_o \wedge^{F}_{o\to o\to o}\, s_o)_o)_{\alpha\to o}\,]\!]^{M,g} \\ &\quad= [\![\,[\,(\lambda x_\alpha.\, ((E!_{\alpha\to o}\, x_\alpha)_o \wedge^{F}_{o\to o\to o}\, s_o)_o)_{\alpha\to o}\,]\,]\!]^{M;g'} \\ &\quad= [\![\,(\lambda x_\alpha.\, ((E_{\alpha\to o}\, x_\alpha)_o \wedge^{H}_{o\to o\to o}\, [\,s_o\,])_o)_{\alpha\to o}\,]\!]^{M;g'}.\end{aligned}$$

Therefore,

$$\begin{aligned}&I(\iota^{F}_{(\alpha\to o)\to\alpha})\,([\![\,(\lambda x_\alpha.\, s_o)_{\alpha\to o}\,]\!]^{M,g}) \\ &\qquad= d \\ &\qquad= I'(\iota^{H}_{(\alpha\to o)\to\alpha})\,([\![\,(\lambda x_\alpha.\, ((E_{\alpha\to o}\, x_\alpha)_o \wedge^{H} [\,s_o\,])_o)_{\alpha\to o}\,]\!]^{M;g'}) \\ &\qquad= [\![\,\iota^{H}_{(\alpha\to o)\to\alpha}\,]\!]^{M;g'}\,[\![\,(\lambda x_\alpha.\, ((E_{\alpha\to o}\, x_\alpha)_o \wedge^{H} [\,s_o\,])_o)_{\alpha\to o}\,]\!]^{M;g'} \\ &\qquad= [\![\,(\iota^{H}_{(\alpha\to o)\to\alpha}(\lambda x_\alpha.\, ((E_{\alpha\to o}\, x_\alpha)_o \wedge^{H} [\,s_o\,])_o)_{\alpha\to o})_\alpha\,]\!]^{M;g'} \\ &\qquad= [\![\,[\,(\iota^{F}_{(\alpha\to o)\to\alpha}(\lambda x_\alpha.\, s_o)_{\alpha\to o})_\alpha\,]\,]\!]^{M;g'}.\end{aligned}$$

For the second case, we have $I(\iota^{F}_{(\alpha\to o)\to\alpha})\,([\![\,(\lambda x_\alpha.\, s_o)_{\alpha\to o}\,]\!]^{M,g}) = \bot_\alpha$ if $\alpha \in \mathcal{T}_i$ and $I(\iota^{F}_{(\alpha\to o)\to\alpha})\,([\![\,(\lambda x_\alpha.\, s_o)_{\alpha\to o}\,]\!]^{M,g}) = \mathrm{F}_\alpha$ if $\alpha \in \mathcal{T}_o$. Then, similar to before,

$$\begin{aligned}\bot_\alpha &= e_\alpha = [\![\,[\,(\iota^{F}_{(\alpha\to o)\to\alpha}(\lambda x_\alpha.\, s_o)_{\alpha\to o})_\alpha\,]\,]\!]^{M;g'} \quad \text{if } \alpha \in \mathcal{T}_i, \\ \mathrm{F}_\alpha &= e_\alpha = [\![\,[\,(\iota^{F}_{(\alpha\to o)\to\alpha}(\lambda x_\alpha.\, s_o)_{\alpha\to o})_\alpha\,]\,]\!]^{M;g'} \quad \text{if } \alpha \in \mathcal{T}_o.\end{aligned}$$

This finishes the proof.

∎

A.2 Proof of Lemma 2

Lemma 2. For all NgFHOL models M and NgFHOL variable assignments g,

$$[\![s_\alpha]\!]^{M,g} = [\![[s_\alpha]]\!]^{M,g'}.$$

Proof. The proof is by induction on the structure of s_α.

For all cases except two, namely $((=^{F}_{\alpha\to\alpha\to o}\ s_\alpha)_{\alpha\to o}\ t_\alpha)_o$ and $(s_{\alpha\to\beta}\, t_\alpha)_\beta$, the proof steps are the same as in the proof of lemma 1.[50] We only cover the remaining cases and avoid repeating the rest.

For $s_\alpha = ((=^{F}_{\alpha\to\alpha\to o}\ s_\alpha)_{\alpha\to o}\ t_\alpha)_o$, we need to distinguish two more cases.

First, if $[\![s_\alpha]\!]^{M,g}, [\![t_\alpha]\!]^{M,g} \in E_\alpha$, we have:

$$\begin{aligned}[\![((=^{F}_{\alpha\to\alpha\to o}\ s_\alpha)_{\alpha\to o}\ t_\alpha)_o]\!]^{M,g} &= ([\![=^{F}_{\alpha\to\alpha\to o}]\!]^{M,g}([\![s_\alpha]\!]^{M,g}))\,([\![t_\alpha]\!]^{M,g}) \\ &= (I(=^{F}_{\alpha\to\alpha\to o})\,([\![s_\alpha]\!]^{M,g}))\,([\![t_\alpha]\!]^{M,g}).\end{aligned}$$

Since $(I(=^{F}_{\alpha\to\alpha\to o})\, d_1)\, d_2 = id(d_1, d_2) = (I'(=^{H}_{\alpha\to\alpha\to o})\, d_1)\, d_2$ for all $d_1, d_2 \in D_\alpha = D'_\alpha$,

$$\begin{aligned}(I(=^{F}_{\alpha\to\alpha\to o})\,([\![s_\alpha]\!]^{M,g}))\,([\![t_\alpha]\!]^{M,g}) &= (I'(=^{H}_{\alpha\to\alpha\to o})\,([\![s_\alpha]\!]^{M,g}))\,([\![t_\alpha]\!]^{M,g}) \\ &= ([\![=^{H}_{\alpha\to\alpha\to o}]\!]^{M,g'}([\![[s_\alpha]]\!]^{M,g'}))\,([\![[t_\alpha]]\!]^{M,g'}).\end{aligned}$$

Because of $[\![s_\alpha]\!]^{M,g} \in E_\alpha$, we know that $[\![(E!_{\alpha\to o} s_\alpha)_o]\!]^{M,g} = \mathrm{T}$. By induction hypothesis, $[\![[(E!_{\alpha\to o} s_\alpha)_o]]\!]^{M,g'} = \mathrm{T}$. The same holds for $[\![t_\alpha]\!]^{M,g}$. Thus, we can add the following conjunctions without changing the meaning of the formula:

$$\begin{aligned}&([\![=^{H}_{\alpha\to\alpha\to o}]\!]^{M,g'}([\![[s_\alpha]]\!]^{M,g'}))\,([\![[t_\alpha]]\!]^{M,g'}) \\ &\quad= (([\![\wedge^{H}_{o\to o\to o}]\!]^{M,g'}\,([\![\wedge^{H}_{o\to o\to o}]\!]^{M,g'}\,[\![[(E!_{\alpha\to o} s_\alpha)_o]]\!]^{M,g'})\,[\![[(E!_{\alpha\to o} t_\alpha)_o]]\!]^{M,g'})) \\ &\qquad(([\![=^{H}_{\alpha\to\alpha\to o}]\!]^{M,g'}\,[\![[s_\alpha]]\!]^{M,g'})\,[\![[t_\alpha]]\!]^{M,g'})) \\ &\quad= (([\![\wedge^{H}_{o\to o\to o}]\!]^{M,g'}\,(([\![\wedge^{H}_{o\to o\to o}]\!]^{M,g'}\,[\![(E_{\alpha\to o}[s_\alpha])_o]\!]^{M,g'})\,[\![(E_{\alpha\to o}[t_\alpha])_o]\!]^{M,g'})) \\ &\qquad(([\![=^{H}_{\alpha\to\alpha\to o}]\!]^{M,g'}\,[\![[s_\alpha]]\!]^{M,g'})\,[\![[t_\alpha]]\!]^{M,g'})) \\ &\quad= [\![((\wedge^{H}_{o\to o\to o}\,((\wedge^{H}_{o\to o\to o}\,(E_{\alpha\to o}[s_\alpha])_o)_{o\to o}(E_{\alpha\to o}[t_\alpha])_o)_o)_{o\to o} \\ &\qquad((=^{H}_{\alpha\to\alpha\to o}\,[s_\alpha])_{\alpha\to o}\,[t_\alpha])_o)_o]\!]^{M,g'} \\ &\quad= [\![[((=^{F}_{\alpha\to\alpha\to o}\ s_\alpha)_{\alpha\to o}\ t_\alpha)_o]]\!]^{M,g'}.\end{aligned}$$

[50] This can easily be seen since the interpretation functions are the same for both positive and negative semantics, and considering that $D_\alpha = E_\alpha$ for all $\alpha \in \mathcal{T}_o$ makes most case distinctions unnecessary.

Now, if either $[\![s_\alpha]\!]^{M,g} \notin E_\alpha$ or $[\![t_\alpha]\!]^{M,g} \notin E_\alpha$ (or both), we have:

$$\begin{aligned}[\![((=^{F}_{\alpha\to\alpha\to o}\ s_\alpha)_{\alpha\to o}\ t_\alpha)_o]\!]^{M,g} &= ([\![=^{F}_{\alpha\to\alpha\to o}]\!]^{M,g}([\![s_\alpha]\!]^{M,g}))\,([\![t_\alpha]\!]^{M,g}) \\ &= \mathrm{F}_o = \mathrm{F}.\end{aligned}$$

Without loss of generality, if $[\![s_\alpha]\!]^{M,g} \notin E_\alpha$, we know that $[\![(E!_{\alpha\to o} s_\alpha)_o]\!]^{M,g} = \mathrm{F}$. By induction hypothesis, $[\![[(E!_{\alpha\to o} s_\alpha)_o]]\!]^{M,g'} = \mathrm{F}$. Therefore, we can add again the same conjunctions as for the previous case:

$$\mathrm{F} = (([\![\wedge^{H}_{o\to o\to o}]\!]^{M,g'}\,([\![\wedge^{H}_{o\to o\to o}]\!]^{M,g'}\,[\![[(E!_{\alpha\to o} s_\alpha)_o]]\!]^{M,g'})\,[\![[(E!_{\alpha\to o} t_\alpha)_o]]\!]^{M,g'})),$$

and the remaining steps are the same as for the case with $[\![s_\alpha]\!]^{M,g}, [\![t_\alpha]\!]^{M,g} \in E_\alpha$.

For $s_\alpha = (s_{\alpha\to\beta}\, t_\alpha)_\beta$, we have another case distinction.

The first case is function application, i.e., $s_\alpha = (s_{\alpha\to_i\beta}\, t_\alpha)_\beta$:

$$\begin{aligned}[\![(s_{\alpha\to_i\beta}\, t_\alpha)_\beta]\!]^{M,g} &= [\![s_{\alpha\to_i\beta}]\!]^{M,g}\,([\![t_\alpha]\!]^{M,g}) = [\![[s_{\alpha\to_i\beta}]]\!]^{M,g'}\,([\![[t_\alpha]]\!]^{M,g'}) \\ &= [\![([s_{\alpha\to_i\beta}]\,[t_\alpha])_\beta]\!]^{M,g'} = [\![[(s_{\alpha\to_i\beta}\, t_\alpha)_\beta]]\!]^{M,g'}.\end{aligned}$$

The second case is the application for predicates, $s_\alpha = (s_{\alpha\to_o\beta}\, t_\alpha)_\beta$.

First, if $[\![t_\alpha]\!]^{M,g} \in E_\alpha$, we have:

$$\begin{aligned}[\![(s_{\alpha\to_o\beta}\, t_\alpha)_\beta]\!]^{M,g} &= [\![s_{\alpha\to_o\beta}]\!]^{M,g}\,([\![t_\alpha]\!]^{M,g}) = [\![[s_{\alpha\to_o\beta}]]\!]^{M,g'}\,([\![[t_\alpha]]\!]^{M,g'}) \\ &= [\![([s_{\alpha\to_o\beta}]\,[t_\alpha])_\beta]\!]^{M,g'} \\ &= [\![(ite^{H}_{o\to\beta\to\beta\to\beta}\,(E_{\alpha\to o}\, t_\alpha)_o\,([s_{\alpha\to_o\beta}]\,[t_\alpha])_\beta\, e_\beta)_\beta]\!]^{M,g'} \\ &= [\![[(s_{\alpha\to_o\beta}\, t_\alpha)_\beta]]\!]^{M,g'},\end{aligned}$$

because $(E_{\alpha\to o}\, t_\alpha)_o = \mathrm{T}$ if and only if $[\![t_\alpha]\!]^{M,g} \in E_\alpha$.

Then, if $[\![t_\alpha]\!]^{M,g} \notin E_\alpha$, we have:

$$\begin{aligned}[\![(s_{\alpha\to_o\beta}\, t_\alpha)_\beta]\!]^{M,g} &= \mathrm{F}_\beta = e_\beta \\ &= [\![(ite^{H}_{o\to\beta\to\beta\to\beta}\,(E_{\alpha\to o}\, t_\alpha)_o\,([s_{\alpha\to_o\beta}]\,[t_\alpha])_\beta\, e_\beta)_\beta]\!]^{M,g'} \\ &= [\![[(s_{\alpha\to_o\beta}\, t_\alpha)_\beta]]\!]^{M,g'},\end{aligned}$$

because $(E_{\alpha\to o}\, t_\alpha)_o = \mathrm{F}$ if and only if $[\![t_\alpha]\!]^{M,g} \notin E_\alpha$.

This finishes the proof.

■

B Embeddings in Isabelle/HOL

B.1 Complete Embedding of the Set-Theoretical Approach

```
typedecl w
type_synonym wSet = "w ⇒ bool"
type_synonym wSetSet = "(w ⇒ bool) ⇒ bool"

definition wSetMember :: "w ⇒ wSet ⇒ bool" (infixr "∈{}" 53)
  where "x ∈{} S ≡ S x"
definition wSetSetMember :: "wSet ⇒ wSetSet ⇒ bool" (infixr "∈{{}}" 53)
  where "x ∈{{}} S ≡ S x"

definition wSetUnion :: "wSet ⇒ wSet ⇒ wSet" (infixr "∪" 50)
  where "A ∪ B ≡ λx. A x ∨ B x"
definition wSetOther :: "wSet ⇒ wSet ⇒ wSet" (infixr "-" 50)
  where "A - B ≡ λx. A x ∧ ¬(B x)"
definition wSetSubsetEq :: "wSet ⇒ wSet ⇒ bool" (infixr "⊆" 50)
  where "A ⊆ B ≡ ∀x. A x ⟶ B x"

consts W :: "wSet"
axiomatization where defW: "∀x. x ∈{} W"

consts emptySet :: "wSet" ("{}")
axiomatization where defEmptySet: "∀x. ¬(x ∈{} {})"

consts emptySetSet :: "wSetSet" ("{{}}")
axiomatization where defEmptySetSet: "∀x. ¬(x ∈{{}} {{}})"

consts r :: "w ⇒ w ⇒ bool" (infixr "r" 53)

abbreviation reflexive :: "bool"
  where "reflexive ≡ ∀x. x r x"
abbreviation symmetric :: "bool"
  where "symmetric ≡ ∀x y. x r y ⟶ y r x"
abbreviation transitive :: "bool"
  where "transitive ≡ ∀x y z. (x r y) ∧ (y r z) ⟶ (x r z)"
abbreviation universal :: "bool"
  where "universal ≡ ∀x y. x r y"

axiomatization where S5:
  "reflexive ∧ symmetric ∧ transitive ∧ universal"

abbreviation R :: "w ⇒ wSet" ("R_"[52]53)
  where "R w ≡ λx. w r x"
```

```
consts fmExistenceDomains :: "w ⇒ wSetSet" ("D")

definition fmTop :: "σ" ("⊤")
  where "⊤ ≡ λw. W"
definition fmBot :: "σ" ("⊥")
  where "⊥ ≡ λw. {}"

definition fmIdentity :: "wSet ⇒ wSet ⇒ wSet" (infixr "=" 56)
  where "φ = ψ ≡ if (φ = ψ) then W else {}"

definition fmNot :: "wSet ⇒ wSet" ("¬_" [52]53)
  where "¬φ ≡ W - φ"
definition fmOr :: "wSet ⇒ wSet ⇒ wSet" (infixr "∨" 51)
  where "φ ∨ ψ ≡ φ ∪ ψ"

definition fmAnd :: "wSet ⇒ wSet ⇒ wSet" (infixr "∧" 52)
  where "φ ∧ ψ ≡ ¬(¬φ ∨ ¬ψ)"
definition fmImp :: "wSet ⇒ wSet ⇒ wSet" (infixr "→" 49)
  where "φ → ψ ≡ ¬φ ∨ ψ"
definition fmEquiv :: "wSet ⇒ wSet ⇒ wSet" (infixr "↔" 50)
  where "φ ↔ ψ ≡ (φ → ψ) ∧ (ψ → φ)"

definition fmBox :: "wSet ⇒ wSet" ("□_" [52]53)
  where "□φ ≡ λw. (R w) ⊆ φ"

definition fmDia :: "wSet ⇒ wSet" ("◇_" [52]53)
  where "◇φ ≡ ¬(□¬φ)"

consts Qex :: "w ⇒ wSetSet" ("|Q|")
definition Qin :: "wSet ⇒ wSet" ("Q")
  where "Qφ ≡ λw. φ ∈{{}} |Q|(w)"

definition fmForall :: "(wSet ⇒ wSet) ⇒ wSet" ("∀")
  where "∀Φ ≡ λw. ∀x. x ∈{{}} (D w) ⟶ w ∈{} (Φ x)"
definition fmForallB:: "(wSet ⇒ wSet) ⇒ wSet" (binder "∀" [8]9)
  where "∀x. φ x ≡ ∀φ"

definition fmExists :: "(wSet ⇒ wSet) ⇒ wSet" ("∃")
  where "∃Φ ≡ ¬(∀(λy. ¬(Φ y)))"
definition fmExistsB:: "(σ ⇒ σ) ⇒ σ" (binder "∃" [8]9)
  where "∃x. φ x ≡ ∃φ"

definition fmValid :: "wSet ⇒ bool" ("⌊_⌋" [7]8)
  where "⌊φ⌋ ≡ φ = W"
```

B.2 Complete Polymorphic Embedding of PFHOL into HOL

```
typedecl i

consts fExistence :: "'a ⇒ bool" ("E")

consts fUndef :: "'a" ("e")
axiomatization where fUndefIAxiom: "¬E (e::i)"
axiomatization where fFalsehoodBAxiom: "(e::bool) = False"
axiomatization where fTrueAxiom: "E True"
axiomatization where fFalseAxiom: "E False"

axiomatization where fNonemptyDomains: "∃x. E x"

definition fIdentity :: "'a ⇒ 'a ⇒ bool" (infixr "=" 56)
    where "φ = ψ ≡ φ = ψ"

definition fNot :: "bool ⇒ bool" ("¬_" [52]53)
    where "¬φ ≡ ¬φ"
definition fOr :: "bool ⇒ bool ⇒ bool" (infixr "∨" 51)
    where "φ ∨ ψ ≡ φ ∨ ψ"

definition fAnd :: "bool ⇒ bool ⇒ bool" (infixr "∧" 52)
  where "φ ∧ ψ ≡ ¬(¬φ ∨ ¬ψ)"
definition fImp :: "bool ⇒ bool ⇒ bool" (infixr "⟶" 49)
  where "φ ⟶ ψ ≡ ¬φ ∨ ψ"
definition fEquiv :: "bool ⇒ bool ⇒ bool" (infixr "↔" 50)
  where "φ ↔ ψ ≡ φ ⟶ ψ ∧ ψ ⟶ φ"

definition fForall :: "('a ⇒ bool) ⇒ bool" ("∀")
  where "∀Φ ≡ ∀x. E x ⟶ Φ x"
definition fForallBinder:: "('a ⇒ bool) ⇒ bool" (binder "∀" [8]9)
  where "∀x. φ x ≡ ∀φ"

definition fExists :: "('a ⇒ bool) ⇒ bool" ("∃")
  where "∃Φ ≡ ¬(∀(λy. ¬(Φ y)))"
definition fExistsBinder :: "('a ⇒ bool) ⇒ bool" (binder "∃" [8]9)
  where "∃x. φ x ≡ ∃φ"

definition fThat :: "('a ⇒ bool) ⇒ 'a" ("I")
  where "IΦ ≡ if ∃x. E x ∧ Φ x ∧ (∀y. (E y ∧ Φ y) ⟶ (y = x))
              then THE x. E x ∧ Φ x
              else e"
definition fThatBinder:: "('a ⇒ bool) ⇒ 'a" (binder "I" [8]9)
  where "Ix. φ x ≡ Iφ"
```

B.3 Complete Nonpolymorphic Embedding of PFHOL into HOL

```
typedecl i

consts fExistenceI :: "i ⇒ bool" ("E$^{i}$")
consts fExistenceP :: "(i ⇒ bool) ⇒ bool" ("E$^{p}$")

consts fUndefI :: "i" ("**e**$^{i}$")
axiomatization where fUndefIAxiom: "¬E$^{i}$ **e**$^{i}$"

consts fFalsehoodB :: "bool" ("**e**$^{b}$")
axiomatization where fFalsehoodBAxiom: "**e**$^{b}$ = False"
axiomatization where fTrueAxiom: "E True"
axiomatization where fFalseAxiom: "E False"

axiomatization where fnonemptyDomainI: "∃x. E$^{i}$ x"
axiomatization where fnonemptyDomainP: "∃x. E$^{p}$ x"

definition fIdentityI :: "i ⇒ i ⇒ bool" (infixr "=" 56)
  where "φ = ψ ≡ φ = ψ"

definition fNot :: "bool ⇒ bool" ("¬_" [52]53)
  where "¬φ ≡ ¬φ"
definition fOr (infixr "∨" 51)
  where "φ ∨ ψ ≡ φ ∨ ψ"

definition fAnd :: "bool ⇒ bool ⇒ bool" (infixr "∧" 52)
  where "φ ∧ ψ ≡ ¬(¬φ ∨ ¬ψ)"
definition fImp :: "bool ⇒ bool ⇒ bool" (infixr "⟶" 49)
  where "φ ⟶ ψ ≡ ¬φ ∨ ψ"
definition fEquiv :: "bool ⇒ bool ⇒ bool" (infixr "⟷" 50)
  where "φ ⟷ ψ ≡ φ ⟶ ψ ∧ ψ ⟶ φ"

definition fForallI :: "(i ⇒ bool) ⇒ bool" ("∀$^{i}$")
  where "∀$^{i}$Φ ≡ ∀x. E$^{i}$ x ⟶ Φ x"
definition fForallIBinder:: "(i ⇒ bool) ⇒ bool" (binder "∀$^{i}$" [8]9)
  where "∀$^{i}$x. φ x ≡ ∀$^{i}$φ"
definition fForallP :: "((i ⇒ bool) ⇒ bool) ⇒ bool" ("∀$^{p}$")
  where "∀$^{p}$Φ ≡ ∀x. E$^{p}$ x ⟶ Φ x"
definition fForallPBinder:: "((i ⇒ bool) ⇒ bool) ⇒ bool" (binder "∀$^{p}$" [8]9)
  where "∀$^{p}$x. φ x ≡ ∀$^{p}$φ"

definition fExistsI :: "(i ⇒ bool) ⇒ bool" ("∃$^{i}$")
  where "∃$^{i}$Φ ≡ ¬(∀$^{i}$(λy. ¬(Φ y)))"
definition fExistsIBinder :: "(i ⇒ bool) ⇒ bool" (binder "∃$^{i}$" [8]9)
  where "∃$^{i}$x. φ x ≡ ∃$^{i}$φ"
```

```
definition fExistsP :: "((i ⇒ bool) ⇒ bool) ⇒ bool" ("∃ᵖ")
  where "∃ᵖΦ ≡ ¬(∀ᵖ(λy. ¬(Φ y)))"
definition fExistsPBinder :: "((i ⇒ bool) ⇒ bool) ⇒ bool" (binder "∃ᵖ" [8]9)
  where "∃ᵖx. φ x ≡ ∃ᵖφ"

definition fThatI :: "(i ⇒ bool) ⇒ i" ("I")
  where "IΦ ≡ if ∃x. Eⁱ x ∧ Φ x ∧ (∀y. (Eⁱ y ∧ Φ y) ⟶ (y = x))
              then THE x. Eⁱ x ∧ Φ x
              else eⁱ"
definition fThatIBinder:: "(i ⇒ bool) ⇒ i" (binder "I" [8]9)
  where "Ix. φ x ≡ Iφ"
```

B.4 Complete Polymorphic Embedding of NgFHOL into HOL

```
typedecl i

consts fExistence :: "'a ⇒ bool" ("E")

consts fUndef :: "'a" ("e")
axiomatization where fUndefIAxiom: "¬E (e::i)"
axiomatization where fFalsehoodBAxiom: "(e::bool) = False"
axiomatization where fTrueAxiom: "E True"
axiomatization where fFalseAxiom: "E False"

axiomatization where fNonemptyDomains: "∃x. E x"

definition fIdentity :: "'a ⇒ 'a ⇒ bool" (infixr "=" 56)
    where "φ = ψ ≡ E φ ∧ E ψ ∧ (φ = ψ)"

definition fNot :: "bool ⇒ bool" ("¬_" [52]53)
    where "¬φ ≡ ¬φ"
definition fOr :: "bool ⇒ bool ⇒ bool" (infixr "∨" 51)
    where "φ ∨ ψ ≡ φ ∨ ψ"

definition fAnd :: "bool ⇒ bool ⇒ bool" (infixr "∧" 52)
  where "φ ∧ ψ ≡ ¬(¬φ ∨ ¬ψ)"
definition fImp :: "bool ⇒ bool ⇒ bool" (infixr "⟶" 49)
  where "φ ⟶ ψ ≡ ¬φ ∨ ψ"
definition fEquiv :: "bool ⇒ bool ⇒ bool" (infixr "↔" 50)
  where "φ ↔ ψ ≡ φ ⟶ ψ ∧ ψ ⟶ φ"

definition fForall :: "('a ⇒ bool) ⇒ bool" ("∀")
  where "∀Φ ≡ ∀x. E x ⟶ Φ x"
definition fForallBinder:: "('a ⇒ bool) ⇒ bool" (binder "∀" [8]9)
  where "∀x. φ x ≡ ∀φ"
```

```
definition fExists :: "('a ⇒ bool) ⇒ bool" ("∃")
  where "∃Φ ≡ ¬(∀(λy. ¬(Φ y)))"
definition fExistsBinder :: "('a ⇒ bool) ⇒ bool" (binder "∃" [8]9)
  where "∃x. φ x ≡ ∃φ"

definition fThat :: "('a ⇒ bool) ⇒ 'a" ("I")
  where "IΦ ≡ if ∃x. E x ∧ Φ x ∧ (∀y. (E y ∧ Φ y) ⟶ (y = x))
              then THE x. E x ∧ Φ x
              else e"
definition fThatBinder:: "('a ⇒ bool) ⇒ 'a" (binder "I" [8]9)
  where "Ix. φ x ≡ Iφ"

definition fPredicate1 :: "('a ⇒ bool) ⇒ 'a ⇒ bool" ("ⁿ")
  where "ⁿP x ≡ E x ∧ P x"
definition fPredicate2 :: "('a ⇒ 'b ⇒ bool) ⇒ 'a ⇒ 'b ⇒ bool" ("ⁿⁿ")
  where "ⁿⁿP x y ≡ E x ∧ E y ∧ P x y"
```

B.5 Complete Nonpolymorphic Embedding of NgFHOL into HOL

```
typedecl i

consts fExistenceI :: "i ⇒ bool" ("Eⁱ")
consts fExistenceP :: "(i ⇒ bool) ⇒ bool" ("Eᵖ")

consts fUndefI :: "i" ("eⁱ")
axiomatization where fUndefIAxiom: "¬Eⁱ eⁱ"

consts fFalsehoodB :: "bool" ("eᵇ")
axiomatization where fFalsehoodBAxiom: "eᵇ = False"
axiomatization where fTrueAxiom: "E True"
axiomatization where fFalseAxiom: "E False"

axiomatization where fnonemptyDomainI: "∃x. Eⁱ x"
axiomatization where fnonemptyDomainP: "∃x. Eᵖ x"

definition fIdentityI :: "i ⇒ i ⇒ bool" (infixr "=" 56)
  where "φ = ψ ≡ Eⁱ φ ∧ Eⁱ ψ ∧ (φ = ψ)"

definition fNot :: "bool ⇒ bool" ("¬_" [52]53)
  where "¬φ ≡ ¬φ"
definition fOr :: "bool ⇒ bool ⇒ bool" (infixr "V" 51)
  where "φ V ψ ≡ φ ∨ ψ"
```

```
definition fAnd :: "bool ⇒ bool ⇒ bool" (infixr "∧" 52)
  where "φ ∧ ψ ≡ ¬(¬φ ∨ ¬ψ)"
definition fImp :: "bool ⇒ bool ⇒ bool" (infixr "⟶" 49)
  where "φ ⟶ ψ ≡ ¬φ ∨ ψ"
definition fEquiv :: "bool ⇒ bool ⇒ bool" (infixr "↔" 50)
  where "φ ↔ ψ ≡ φ ⟶ ψ ∧ ψ ⟶ φ"

definition fForallI :: "(i ⇒ bool) ⇒ bool" ("∀ⁱ")
  where "∀ⁱΦ ≡ ∀x. Eⁱ x ⟶ Φ x"
definition fForallIBinder:: "(i ⇒ bool) ⇒ bool" (binder "∀ⁱ" [8]9)
  where "∀ⁱx. φ x ≡ ∀ⁱφ"
definition fForallP :: "((i ⇒ bool) ⇒ bool) ⇒ bool" ("∀ᵖ")
  where "∀ᵖΦ ≡ ∀x. Eᵖ x ⟶ Φ x"
definition fForallPBinder:: "((i ⇒ bool) ⇒ bool) ⇒ bool" (binder "∀ᵖ" [8]9)
  where "∀ᵖx. φ x ≡ ∀ᵖφ"

definition fExistsI :: "(i ⇒ bool) ⇒ bool" ("∃ⁱ")
  where "∃ⁱΦ ≡ ¬(∀ⁱ(λy. ¬(Φ y)))"
definition fExistsIBinder :: "(i ⇒ bool) ⇒ bool" (binder "∃ⁱ" [8]9)
  where "∃ⁱx. φ x ≡ ∃ⁱφ"
definition fExistsP :: "((i ⇒ bool) ⇒ bool) ⇒ bool" ("∃ᵖ")
  where "∃ᵖΦ ≡ ¬(∀ᵖ(λy. ¬(Φ y)))"
definition fExistsPBinder :: "((i ⇒ bool) ⇒ bool) ⇒ bool" (binder "∃ᵖ" [8]9)
  where "∃ᵖx. φ x ≡ ∃ᵖφ"

definition fThatI :: "(i ⇒ bool) ⇒ i" ("𝐈")
  where "𝐈Φ ≡ if ∃x. Eⁱ x ∧ Φ x ∧ (∀y. (Eⁱ y ∧ Φ y) ⟶ (y = x))
              then THE x. Eⁱ x ∧ Φ x
              else 𝐞ⁱ"
definition fThatIBinder:: "(i ⇒ bool) ⇒ i" (binder "𝐈" [8]9)
  where "𝐈x. φ x ≡ 𝐈φ"

definition fPredicateI1 :: "(i ⇒ bool) ⇒ i ⇒ bool" ("ⁿ")
  where "ⁿP x ≡ Eⁱ x ∧ P x"
definition fPredicateI2 :: "(i ⇒ i ⇒ bool) ⇒ i ⇒ i ⇒ bool" ("ⁿⁿ")
  where "ⁿⁿP x y ≡ Eⁱ x ∧ Eⁱ y ∧ P x y"
```

C Countermodels

C.1 Countermodel for Prior's Theorem with Three Worlds

```
axiomatization where fmNonemptyExistenceDomains: "∀w. (D w) ≠ {{}}"

axiomatization where Ax1: "∀x y. (D x) = (D y)"
axiomatization where Ax2: "∀x. W ∈{{}} (D x) ∧ {} ∈{{}} (D x)"
axiomatization where Ax3: "∀x y. (y ∈{{}} (D x)) ⟶ ((y = {}) ∨ (y = W))"
axiomatization where Ax4: "∃x. W ∈{{}} |Q|(x) ∧ (∀y. y ∈{{}} |Q|(x) ⟶ y = W)"
axiomatization where Ax5: "∃x. {} ∈{{}} |Q|(x) ∧ (∀y. y ∈{{}} |Q|(x) ⟶ y = {})"
axiomatization where Ax6: "∃x. ∀y. ((y ≠{}) ∧ (y ≠ W)) ⟶ (y ∈{{}} |Q|(x))"

lemma "⌊(Q (∀p. (Q p → ¬p))) → ((∃p. Q p ∧ p) ∧ (∃p. Q p ∧ ¬p))⌋"
  unfolding Defs
  nitpick [user_axioms=true, format=2, show_all, card=3]
  oops
```

```
Nitpick found a counterexample for card w = 3:

  Constants:
    |Q| =
      (λx. _)
      ((w₁, (λx. _)(w₁ := True, w₂ := True, w₃ := True)) := True,
       (w₁, (λx. _)(w₁ := True, w₂ := True, w₃ := False)) := False,
       (w₁, (λx. _)(w₁ := True, w₂ := False, w₃ := True)) := False,
       (w₁, (λx. _)(w₁ := True, w₂ := False, w₃ := False)) := False,
       (w₁, (λx. _)(w₁ := False, w₂ := True, w₃ := True)) := False,
       (w₁, (λx. _)(w₁ := False, w₂ := True, w₃ := False)) := False,
       (w₁, (λx. _)(w₁ := False, w₂ := False, w₃ := True)) := False,
       (w₁, (λx. _)(w₁ := False, w₂ := False, w₃ := False)) := False,
       (w₂, (λx. _)(w₁ := True, w₂ := True, w₃ := True)) := False,
       (w₂, (λx. _)(w₁ := True, w₂ := True, w₃ := False)) := False,
       (w₂, (λx. _)(w₁ := True, w₂ := False, w₃ := True)) := False,
       (w₂, (λx. _)(w₁ := True, w₂ := False, w₃ := False)) := False,
       (w₂, (λx. _)(w₁ := False, w₂ := True, w₃ := True)) := False,
       (w₂, (λx. _)(w₁ := False, w₂ := True, w₃ := False)) := False,
       (w₂, (λx. _)(w₁ := False, w₂ := False, w₃ := True)) := False,
       (w₂, (λx. _)(w₁ := False, w₂ := False, w₃ := False)) := True,
       (w₃, (λx. _)(w₁ := True, w₂ := True, w₃ := True)) := False,
       (w₃, (λx. _)(w₁ := True, w₂ := True, w₃ := False)) := True,
       (w₃, (λx. _)(w₁ := True, w₂ := False, w₃ := True)) := True,
       (w₃, (λx. _)(w₁ := True, w₂ := False, w₃ := False)) := True,
       (w₃, (λx. _)(w₁ := False, w₂ := True, w₃ := True)) := True,
```

```
   (w₃, (λx. _)(w₁ := False, w₂ := True, w₃ := False)) := True,
   (w₃, (λx. _)(w₁ := False, w₂ := False, w₃ := True)) := True,
   (w₃, (λx. _)(w₁ := False, w₂ := False, w₃ := False)) := False)
W = (λx. _)(w₁ := True, w₂ := True, w₃ := True)
{} = (λx. _)(w₁ := False, w₂ := False, w₃ := False)
{{}} =
  (λx. _)
  ((λx. _)(w₁ := True, w₂ := True, w₃ := True) := False,
   (λx. _)(w₁ := True, w₂ := True, w₃ := False) := False,
   (λx. _)(w₁ := True, w₂ := False, w₃ := True) := False,
   (λx. _)(w₁ := True, w₂ := False, w₃ := False) := False,
   (λx. _)(w₁ := False, w₂ := True, w₃ := True) := False,
   (λx. _)(w₁ := False, w₂ := True, w₃ := False) := False,
   (λx. _)(w₁ := False, w₂ := False, w₃ := True) := False,
   (λx. _)(w₁ := False, w₂ := False, w₃ := False) := False)
D = (λx. _)
  ((w₁, (λx. _)(w₁ := True, w₂ := True, w₃ := True)) := True,
   (w₁, (λx. _)(w₁ := True, w₂ := True, w₃ := False)) := False,
   (w₁, (λx. _)(w₁ := True, w₂ := False, w₃ := True)) := False,
   (w₁, (λx. _)(w₁ := True, w₂ := False, w₃ := False)) := False,
   (w₁, (λx. _)(w₁ := False, w₂ := True, w₃ := True)) := False,
   (w₁, (λx. _)(w₁ := False, w₂ := True, w₃ := False)) := False,
   (w₁, (λx. _)(w₁ := False, w₂ := False, w₃ := True)) := False,
   (w₁, (λx. _)(w₁ := False, w₂ := False, w₃ := False)) := True,
   (w₁, (λx. _)(w₁ := True, w₂ := True, w₃ := True)) := True,
   (w₂, (λx. _)(w₁ := True, w₂ := True, w₃ := False)) := False,
   (w₂, (λx. _)(w₁ := True, w₂ := False, w₃ := True)) := False,
   (w₂, (λx. _)(w₁ := True, w₂ := False, w₃ := False)) := False,
   (w₂, (λx. _)(w₁ := False, w₂ := True, w₃ := True)) := False,
   (w₂, (λx. _)(w₁ := False, w₂ := True, w₃ := False)) := False,
   (w₂, (λx. _)(w₁ := False, w₂ := False, w₃ := True)) := False,
   (w₂, (λx. _)(w₁ := False, w₂ := False, w₃ := False)) := True,
   (w₃, (λx. _)(w₁ := True, w₂ := True, w₃ := True)) := True,
   (w₃, (λx. _)(w₁ := True, w₂ := True, w₃ := False)) := False,
   (w₃, (λx. _)(w₁ := True, w₂ := False, w₃ := True)) := False,
   (w₃, (λx. _)(w₁ := True, w₂ := False, w₃ := False)) := False,
   (w₃, (λx. _)(w₁ := False, w₂ := True, w₃ := True)) := False,
   (w₃, (λx. _)(w₁ := False, w₂ := True, w₃ := False)) := False,
   (w₃, (λx. _)(w₁ := False, w₂ := False, w₃ := True)) := False,
   (w₃, (λx. _)(w₁ := False, w₂ := False, w₃ := False)) := True)
R = (λx. _)
    ((w₁, w₁) := True, (w₁, w₂) := True, (w₁, w₃) := True,
     (w₂, w₁) := True, (w₂, w₂) := True, (w₂, w₃) := True,
     (w₃, w₁) := True, (w₃, w₂) := True, (w₃, w₃) := True)
```