Automated theorem proving for dependent typed theories via a translation to higher-order logic

Colin Rothgang

Supervised by
Prof. Dr. Christoph Benzmüller
Prof. Dr. Alexander Steen

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science
(Mathematics)

at
Freie Universität Berlin

January 31, 2023

Automated theorem proving for dependent typed theories via a translation to higher-order logic

Acknowledgments

I am very grateful to Christoph Benzmüller and Alexander Steen for their support, guidance and feedback in my research. Moreover, I want to express my gratitude to Florian Rabe for his support, feedback and advice throughout my thesis work as well as several earlier reserach projects in which I learned a lot that helped me write this thesis.

I also want to thank David Fuenmayor and anyone else who participated in the weekly seminars and shared their thoughts during my preliminary presentations about my thesis. I want to thank Daniel Kirchner and Chad Brown for their input and intuition regarding some questions that came up in my work.

Finally, I would like to extend my deepest gratitude to my friends and family for their support throughout my studies.

Contents

Abstract

Higher-order logic (HOL) offers a simple syntax and semantics for representing and reasoning about typed mathematical concepts. There are many state-of-the-art automated theorem provers for HOL. But the type system of HOL lacks advanced features where types may depend on terms. This is useful as many mathematical notions are inherently dependent typed. Dependent type theory offers such a rich type system but has rather substantial conceptual differences to HOL and isn't supported as well by automated theorem provers. We introduce a dependently typed extension DHOLP of HOL that supports dependent types while retaining the style and conceptual framework of HOL. Moreover, we describe one translation from DHOLP and a second one from one of its fragments to HOL and prove the soundness and completeness of the translations. We implement both translations within the MMT system — a system for formalizing mathematics — where they are combined with a HOL prover into an automated theorem prover for DHOLP. Finally, we formalize basic set theory notions in DHOLP and outline how such a formalization can be combined with a recent translation of the language of the Mizar proof assistant to MMT to obtain an automated theorem prover for problems in the language of the Mizar proof assistant.

1 Introduction and Related Work

Motivation and goal of our work In recent times computer-aided mathematics is emerging as a new way for building and communicating mathematical results and proofs. Computer-aided mathematics promises to help with various challenges faced by the traditional approach, like mistakes in complicated proofs, finding existing results and explaining proofs to humans. The traditional approach relies on informal definitions and arguments meant to communicate to and convince other researchers in the field. In contrast, computer-aided mathematics can support human mathematicians by

1. ensuring that mathematical proofs are actually correct,
2. building tools that help explain arguments at the desired level of details,
3. building search engines for formalized mathematical results,
4. organizing complicated proofs, by "bookkeeping" what is known so far and what still needs to be proven.

Tools for formalizing mathematics (such as theorem provers and proof assistants) can also help in other related areas, for example in verifying the correctness of computer programs or (more rarely) checking formal arguments in other fields ranging from physics to philosophy.

However, formalizing mathematics in computer-readable systems creates many challenges as well. In particular, formalizing mathematical proofs in proof assistants requires (manually or automatically) proving many details (of larger proofs) that would be immediately obvious to human researchers working in the field. Such details are not interesting to prove, so we don't want to force mathematicians to spend time on them, although we do want to make sure that they can actually be proven. Often, a mathematical notion can be formalized in several ways which are different to a proof assistant (but not to a human). This leads to situations in which existing results are present only in a form which makes them not directly applicable, requiring proving even more "obvious details". Those issues are likely to only get worse with further adoption of computer-aided mathematics.

In order to address these challenges and aid the adoption of computer-aided mathematics we need better tools for proof assistants that help mathematicians with proving relatively easy lemmas. Ideally we want hammers — tools in a proof assistant that can prove simpler theorems at the push of a button, without any human assistance.

Hammers consist of mainly two parts: an automated theorem prover (ATP) system and a translation from the language of the proof assistant to a logic supported by the ATP system. The ATP can be developed as part of the hammer or, more commonly, an existing ATP system can be used.

Currently, hammers usually translate conjectures into problems written in first-order logic — although recently higher-order logic is becoming a more attractive translation target due to improving ATP systems supporting it — and then try one or several ATP systems on the problem. Afterwards the hammer might or might not attempt to reconstruct the proof, i.e. translate the

proof found by the ATP system back into a proof in the language of the proof assistant. Such proof reconstruction can increase the confidence in the correctness of the results of the hammers and are especially useful when the translation itself isn't sound and complete (meaning it may be possible to prove translations of conjectures that aren't provable originally or vice versa).

Particularly for proof assistants [27, 20, 17] based on higher-order logic (HOL) [9, 18, 1], but also for the proof assistant Mizar [23] based on set theory, good hammer tools [28, 22, 3] are available [5]. Considering the many existing mature ATP systems for HOL [2, 37, 7, 31, 41] itself, this is hardly surprising.

However, higher-order logic is inconvenient for formalizing some advanced concepts (for instance categories) — and dependent types are required to express some mathematical concepts directly, at the cost of making types, type-checking and consequently proof automation more difficult. [1] Dependent types are used in dependent typed logics or more commonly in dependent type theory (DTT) to express such concepts. There exist many proof assistants for various versions of dependent type theory (DTT) [10, 24, 14, 29]. But, contrary to the situation for HOL, there are effectively no existing ATP systems for DTT (or dependent typed logics) — although there are some hammer tools (most notably Coqhammer [12]) for proof assistants using dependent type theory. The difficulty in developing automated theorem provers for logics or type theories with dependent types is mainly due to typing and type equality becoming undecidable, as they require deciding equalities of terms.

Our goal is to work towards bridging this gap between higher-order logic for which many mature ATP systems exist and dependent typed logic which is more useful for formalizing mathematics.

We are thus interested in sound and complete translations from dependent typed logics into HOL, which we can use to build hammers for those logics. A translation being sound and complete, means that all information present in the problem is encoded by the translation and a statement is provable iff its translation is. It is important that the translations are sound and complete, in order to ensure that hammers using them can find proofs for provable conjectures — and only for those. For that reason completeness is the more important property, especially completeness w.r.t. those kinds of statements that we want to prove with a hammer. This also means that we don't need to reconstruct proofs to ensure their correctness. Such translations can be considered as "embeddings" of the source logic into the target logic.

There are two kinds of embeddings. In *shallow embeddings*, types are translated to types that correspond to them as closely as possible. In *deep embeddings* all term and type information is represented on term level. Shallow embeddings are more "natural" than deep embeddings and HOL provers are more likely to be able to use their translation results efficiently. Deep embeddings on the other hand are easier to construct but harder to efficiently use with an ATP system

[1] We are not claiming as Kevin Buzzard conjectured on his blog that such concepts cannot be expressed using different mechanisms in simple typed logics. We are merely saying that otherwise a different mechanism (for instance locales as used in a recent formalization of schemes in Isabelle/Hol [6]) is required. Which mechanism works best long-term (and for which mathematical domains) is hard to say, not enough mathematics has been formalized to really judge this. We are just assuming that dependent types are useful for formalizing mathematics and it is therefore interesting to study proof automation for logics with dependent types.

to build a hammer. We are interested in a shallow embedding. In particular, we want to translate equalities to equalities in HOL, with some additional typing conditions ("relativizations") added.

To enable this we want to translate from a dependently typed logic that — while extending the type system of HOL to make it more expressive — retains the style and conceptual framework of HOL as much as possible. Therefore, we will use judgments (especially validity) to represent statements in the logic, as it is also done also in HOL and the dependent types are only used to express inherently dependently typed mathematical concept (for example in category theory or set theory, as exemplified later in the thesis). This is different from how mathematics is typically represented in proof assistants like Coq, Lean or Agda, which represent propositions as types (via the Curry-Howard correspondence) instead of as formulae as done in HOL and its extensions that we consider. The Curry-Howard correspondence is the observation that truth, falsehood and the logical connectives implication, conjunction and disjunction correspond to empty resp. non-empty types, function types, product and sum types respectively. Then a proof of a formula corresponds to running a function of the corresponding type. Our choice to focus on HOL style problems not using the Curry-Howard correspondence, is also motivated by the desire to make the source logic more intuitive to use for mathematicians, who are usually not familiar with (dependent) type theory but probably are at least somewhat familiar with logic.

We will therefore define a dependently typed (conservative) extension DHOLP of HOL which adds two kinds of dependent types — namely dependent function types (Π-types) and predicate subtypes — to the language and has modified inference rules compared to HOL. Then we consider translations into HOL and study their soundness and completeness properties. Our goal is to find at least one sound translation from DHOLP to HOL which is at the same type complete at least w.r.t. the validity judgment.

Our work is additionally motivated by a concrete practical goal. We want to develop a hammer for the MMT [30] system — a formal system for formalizing mathematical knowledge — and make this hammer as useful as possible. Since dependent types often occur in theories in the MMT system, we want the hammer to also support dependent types. Consequently, we want to implement our translation and build a hammer for MMT theories based on DHOLP.

Related work In the following, we will mention some of the more similar previous work and discuss how this work differs from ours.

Bart and Melham [4] develop a translation of Martin-Löf-style dependent type theory into Gordon's HOL interactive theorem prover [17]. This work differs from ours in various respects. Their work focuses on dependent type theory in the propositions-as-types style, in particular using identity types to represent equalities, whereas we builds DHOLP as an extension of HOL with classical booleans and an equality predicate. Many of the subtleties we encounter are unique to our approach. Furthermore, they target an interactive prover; whereas our approach targets an automated theorem prover.

Another interesting example is the translation used in Coqhammer [12], a hammer tool for the Coq [10] proof assistant. Coqhammer is based on a translation [11] of the calculus of inductive construction [26] (a much more expressive type system than the one we consider) into first-order

logic. This translation is neither sound nor complete in general (meaning not all first-order logic proofs found by the hammer correspond to proofs in Coq and not all first-order translations of provable conjectures are provable). This is due to the translation loosing some information. This is partially intentional as it simplifies the translated problems and in some cases actually improves the chances of being able to lift proofs back to the calculus of inductive constructions. In practice, the translation is "sufficiently" sound and complete to allow lifting many proofs from first-order logic to Coq: more than 40% of all theorems in the Coq standard library [12] can be proven by the hammer in push-button mode.

In this work, we are investigating somewhat simpler translations and therefore it is reasonable to aim for a translation that is sound and complete. If the goal is to also reconstruct proofs however, it might make pay off to aim for a simpler but incomplete translation (unlike ours).

Sojakova and Rabe [35] describe a sound and complete translation of dependently typed first-order logic into simple typed first-order logic. The sound– and correctness proof is based on model-theory and cannot be easily generalized to our setting as many of the main difficulties we encounter don't arise in a first-order setting (such as quantification or equality over function types). It is also not clear how the approach would deal with the non-surjectivity of the translation in higher-order setting. Furthermore, we prefer giving a completeness proof that allows to "lift" proofs from simple typed higher-order logic to dependently typed higher-order logic, allowing the reconstruction of proofs in the source logic.

Contribution We start with higher-order logic and only replace its function types with Π-types, a function type in which the return type may depend on the value of the argument to the function, yielding dependently typed higher-order logic (DHOL). As it turns out, this change already makes finding a sound and complete shallow embedding very difficult, the resulting translation is slightly non-intuitive and proving its correctness is quite challenging. However, the translation and the proof of soundness and completeness generalize: adding predicate subtypes — subtypes specified by a predicate that its members need to satisfy — to the logic requires only a few and rather straightforward changes to the translation and the proofs. The resulting logic is dependently typed higher-order logic with predicate subtypes (DHOLP). We actually present two slightly different translations — one from DHOLP into HOL and one from a fragment of DHOLP into the corresponding fragment of HOL. For the completeness of the translation we only consider completeness w.r.t. validity, i.e. whether formulae with provably valid translations are provably valid as this is the property we actually need in practice. For soundness, we need to restrict the theory we consider. We show the soundness of both translations, the completeness (w.r.t. validity) of the second translation and the completeness (w.r.t. validity) of the first translation if we consider the intuitionistic fragments — meaning we replace one inference rule by a somewhat weaker one — of the source and target logics. Considering the intuitionistic fragments of the logics is motivated by the observation that the translation is otherwise known to be incomplete w.r.t. typing and type-equality, as the translation is loosing information. However we also explain why the translation preserves all information on term level and why the completeness of the translation (w.r.t. validity) from DHOLP into HOL can be done analogously.

Besides these theoretical contributions, we also implement both translations to obtain a hammer (using either translation) for theorems inside MMT [30] theories based on DHOLP. As a HOL prover to use for the translated problems we pick LEO-III [37]. Since LEO-III is designed for proving theorems in simple typed HOL (among other things like modal logics), it is already quite good at proving simple typed theorems. LEO-III is designed with support for various kinds of logics in mind and its ecosystem contains a logic-embedding tool [36] to embed various logics into logics supported by LEO-III. The tool takes input in a language that can also express DHOL (although this isn't intended or supported by ATP systems). We extend the implementation of this tool so that it can also translate DHOL problems into HOL problems which we can pass on into HOL provers.

Furthermore, we continue the work [43] of Wolff, who implemented an extension for MMT using LEO-III as an external prover for content based on first-order and higher-order logic. Building on his work we extend proving support to MMT content based on DHOLP, extending the hammer tool for MMT theories based on HOL to one for MMT theories based on DHOLP. In fact, all the examples given in this thesis have been formalized in MMT, assuring us of the correctness of our analysis of the examples and testing the implementation.

Finally, we demonstrate how DHOLP can be used to formalize set theory. Taking advantage of a recently developed importer [32] from the proof assistant Mizar, which is based on set theory, into MMT it becomes possible to use our MMT hammer to prove theorems from the "Mizar mathematical library" — a library of mathematical results formalized in Mizar.

Structure of the thesis In Section 2, we give some historical background for higher-order logic, explain some basic notions and briefly describe the LEO-III prover and the MMT system. We also define the version of higher-order logic we want to use and talk about its properties.

In Section 3, we introduce DHOL and its intuitionistic fragment DIHOL, their translations into HOL and its intuitionistic fragment IHOL respectively and explain the choice of logic and translation. We also discuss the main difficulty for proving completeness of the translation and describe our approach for overcoming this obstacle. Soundness and completeness of the translation follow from the soundness and completeness of the translation from DHOLP into HOL.

In Section 4, we introduce DHOLP and its intuitionistic fragment DIHOLP and their translations into HOL and IHOL respectively. We prove the soundness of the translations. Then we discuss how we can modify proofs in IHOL into simpler proofs, for which it is easier to construct a corresponding proof in DIHOLP. Finally, we prove that we can lift such proofs in IHOL back to DIHOLP and conclude the completeness of the translation. We argue why this proof generalizes to the same translation from DHOLP into HOL and therefore conjecture the completeness of that translation. Using analogous arguments we also show the completeness of a second similar translation of the fragment of DHOLP in which we disallow bool-valued function types (of positive arity) into the target logic HOL.

In Section 5, we discuss higher-order logic with external propositions (a logic similar to the fragment of HOL in which we disallow quantification over the type bool and also disallow bool-valued function types) and its extensions with dependent types and how to create a similar trans-

lation with analogous soundness and completeness proofs. We summarize our soundness and completeness results in Table 2.

In Section 6, we discuss the implementation of the translation in the logic embedding tool and the MMT system and how both can be used for theorem proving. We demonstrate the implementation of the translations on an example theory formalizing set theory and discuss how to formalize the foundations of the Mizar language and use this with a recent importer into MMT for proving theorems in Mizar.

In Section 7, we discuss an extension of DHOLP with parametric predicates and how we can translate it into HOL. Finally, in Section 8 we conclude and discuss potential future work.

2 Preliminaries

2.1 History of classical logic

In this subsection, we will give a very brief introduction into the development of first-order and higher order logic. This section can serve as a starting point for learning more about logic and can explain better the larger area in which this work is embedded. Since we formally introduce our logics in a self-contained fashion, it is not necessary to read this subsection in order to understand the remaining thesis.

First-order logic and in particular second-order logic were first formally introduced by Frege [15] in 1879. Later on the discovery of Russel's paradox [34, 33] threatened the development of logic and motivated Russel and Whitehead to invent higher-order type theory in their books *Principia mathematica* [42] as a solution to the paradox (some people therefore credit Russel with discovering higher-order logic). Their main idea was to separate out ordinary sets from dangerously large types in order to prevent paradoxes like Russel's paradox. Afterwards, they went on to develop much of known mathematics at the point in their type theory. At this point both Russel and Frege had discovered higher-order logic as part of their work, without giving it many considerations as a system on its own. This changed with Hilbert who inspired by *Principia mathematica* formally introduced modern first-order logic and higher-order logic in his lectures and in 1928 published them with Ackermann in [13]. They also posed the question of the completeness of higher-order logic, i.e. whether it is possible prove all true formulae in the logic. Up until the incompleteness theorem's by Gödel [16], the significance of the difference between first-order and second-order (or higher-order) logic was not realized by most logicians. After the incompleteness theorems it was clear that first-order logic was sound and complete, while higher-order logic wasn't. In 1936 Church introduced a model for computation, called the λ-calculus [8] and in 1940 he published a version of higher-order logic [9] based on it. In 1950 Henkin came up with alternative semantics (called Henkin semantics) for higher-order logic [21]. In higher-order logic with Henkin semantics quantifiers over a function type $A \to B$ are taken to range only over some fixed subset of the set of functions from $A \to B$. With these semantics higher-order logic is equivalent to typed first-order logic and in particular it is complete [21]. As is usually done in recent times for theorem proving in higher-order logic, we will consider higher-order logic with Henkin semantics.

Automated theorem proving in higher-order logic also has a long history [39]. One of the first successful ATPs for HOL is the TPS prover [2]. TPS is based on a version of higher-order logic called Q_0 [1] which also serves as an inspiration for (and has a fragment equivalent to) our definition of HOL. Other early automated theorem prover systems for HOL include Isabelle [27] (which can be used as an ATP system instead of as a proof assistant) and LEO with its successors LEO-II and LEO-III [37]. However, in the last 15 years or so more powerful ATPs for HOL — in particular Satallax [7] and Zipperposition [41] — have been developed as can be seen in their performance in the CASC competition [2]. Due to this somewhat recent development HOL provers are becoming an attractive translation target for development of hammer tools.

2.2 Classical and intuitionistic higher order logic

We start by formalizing higher-order logic HOL and its intuitionistic fragment IHOL. They serve both as the starting point to define DHOL and DHOLP and as the target of our translation from DHOL (and DHOLP) to HOL.

To define a logic we need to define the following three ingredients:

1. an object language used by the logic to express content, in our case this language will be given by a (context-free) grammar
2. judgements which use objects in the object language to form statements
3. axiom (schemata) and inference rules to derive new statements from known ones

We will discuss the three ingredients of the logics defined in this thesis in this order.

HOL uses the following grammar:

T	$::=$	$\circ \mid T,\ Dec$	theories
Dec	$::=$	$a : \mathsf{tp} \mid c : A \mid F$	declarations
Γ	$::=$	$. \mid \Gamma, x : A \mid \Gamma, F$	contexts
A, B	$::=$	$a \mid A \to B \mid \mathsf{bool}$	types
s, t, c, f, F, G	$::=$	$c \mid x \mid f\ t \mid \lambda x : A.\ t \mid s =_A t \mid F \Rightarrow G$	terms

Here the meta-variables on the right can be created using the productions (separated by $\mid$) on the left. For instance,

$$A, B ::= A \to B,$$

means if A, B are types then we can also create the new type $A \to B$. We use the symbols $\circ$ and $.$ to refer to the empty theory and the empty context, respectively. We use $x : A$ in Γ or F in Γ to denote that the variable $x : A$ or the assumption F occurs in Γ. We use $\Gamma, x : A$ to denote the context Γ with the variable declaration $x : A$ appended (and wlog. (renaming) we assume that x previously didn't occur in Γ). Similarly, we use Γ, F to denote the context Γ with the additional assumption F.

[2] https://www.tptp.org/CASC/

As usual we can assume that HOL supports all the usual logical connectives true, false, $\neg$, $\wedge$, $\vee$ and quantifier $\forall$ and $\exists$, either as additional primitives defined in terms of the others or as additions to the grammar. If the connectives are added to the grammar, further inference rules need to be added to the calculus to make sure we can prove the same theorems.

We can define them similarly (except for the existential quantifier and disjunction which are defined by Andrews using negation and the universal quantifier resp. conjunction) to how they are defined by Andrews in [1]. We modify the definitions for the existential quantifier and disjunction, as the definition by Andrews don't work in the intuitionistic fragment (as the equivalence proof requires double negation elimination).

$$
\begin{aligned}
\mathsf{true} &:= \lambda x : \mathsf{bool}.\ x =_{\mathsf{bool}} \lambda x : \mathsf{bool}.\ x \\
\mathsf{false} &:= \lambda x : \mathsf{bool}.\ x =_{\mathsf{bool}} \lambda x : \mathsf{bool}.\ \mathsf{true} \\
\neg &:= \lambda x : \mathsf{bool}.\ x =_{\mathsf{bool}} \mathsf{false} \\
\forall x : A.\ F &:= \lambda x : A.\ F =_{\mathsf{bool}} \lambda x : A.\ \mathsf{true} \\
\wedge &:= \lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ \forall r : \mathsf{bool} \to \mathsf{bool} \to \mathsf{bool}.\ r\ x\ y =_{\mathsf{bool}} r\ \mathsf{true}\ \mathsf{true} \\
\vee &:= \lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ \forall z : \mathsf{bool}.\ (x \Rightarrow z) \Rightarrow (y \Rightarrow z) \Rightarrow z \\
\exists x : A.F &:= \forall y : \mathsf{bool}.\ (\forall x : A.\ F \Rightarrow y) \Rightarrow y
\end{aligned}
$$

We could also define $\Rightarrow$ by $\Rightarrow := \lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ (x \wedge y) =_{\mathsf{bool}} x$. However, we single out $\Rightarrow$ in the grammar because it will subtly be affected when going to DHOL later, so giving inference rules for it simplifies the soundness and completeness proofs.

Our grammar uses contexts Γ that are a list of typed variables $x : A$ and assumptions F. That makes stating the rules below more convenient. However, HOL contexts (in contrast to DHOL contexts below) can always be normalized into a set of variable declarations followed by a set of assumptions because the well-formedness of a type A can never depend on a variable or an assumption.

We will furthermore allow defined declarations in the theory, but will consider them as meta-level abbreviations, not actual declarations. So they are not part of the grammar or language, instead they are merely syntactic sugar for their definiens. This is useful to abbreviate some terms that occur many times within a theory. The logical connectives defined above are also realized in exactly this way.

The type and proof system for HOL uses the following judgments. Note that the equality of types is a separate judgment because it cannot be expressed in the syntax, whereas the equality of terms is a special case of the validity judgment — since equalities are ordinary formulae — we only list equality here, because in other versions of HOL equality may be a separate judgement.

Name	Judgment	Intuition
theories	$\vdash\ T\ \mathsf{Thy}$	T is well-formed theory
contexts	$\vdash_T\ \Gamma\ \mathsf{Ctx}$	Γ is well-formed context
types	$\Gamma \vdash_T\ A\ \mathsf{tp}$	A is well-formed type
typing	$\Gamma \vdash_T\ t : A$	well-formed term t has well-formed type A
validity	$\Gamma \vdash_T\ F$	well-formed boolean F is provable
equality of terms	$\Gamma \vdash_T\ t =_A t'$	special case of validity
equality of types	$\Gamma \vdash_T\ A \equiv B$	well-formed types A and B are provably equal

All judgments except for $\vdash\ T\ \mathsf{Thy}$ are relative to a theory T. If we use $\vdash$ instead of $\vdash_T$ for such a judgment, we refer to the judgement relative to the empty theory $\circ$ and we use $\vdash_T\ F$ to refer to the judgement $.\ \vdash_T\ F$ in the empty context. The calculus has the rules given in Figure 1. We assume that names in a theory or a context are unique without making that explicit in the rules. Furthermore, we use $E[{}^{x}/_{t}]$ to denotes the capture-avoiding (meaning that we only replace x in the scope of a λ-function or quantifier binding a variable x) substitution of the variable x with the term t within expression E.

The rules for IHOL are the same as for HOL, except that we replace rule (boolExt) with rule (propExt) (which otherwise follows from rule (boolExt) as shown in Lemma 1).

Rule (boolExt) states that if the application of a predicate on bool holds for both true and for false, then the predicate holds on an arbitrary boolean variable and is therefore trivially true. This rule allows us to proof a formula by case distinction on the value of a boolean variable and by rule (∀E) (proven in Lemma 1) also by case distinction on the value of arbitrary boolean subterms. This implies that all booleans are equal to either true or equal to false. In contrast, in IHOL it is not provable whether there are booleans non equal to true or false.

Some terminology The following terminology will be used for all the logics in this thesis.

Definition 1. We will call a theory T *well-formed* whenever we can show $\vdash\ T\ \mathsf{Thy}$. We will say a context Γ is *well-formed* (relative to T), iff we can show $\vdash\ \Gamma\ \mathsf{Ctx}$ ($\vdash_T\ \Gamma\ \mathsf{Ctx}$). We will say that a type A is *well-formed* in context Γ (relative to T) if we can show $\Gamma \vdash\ A\ \mathsf{tp}$ ($\Gamma \vdash_T\ A\ \mathsf{tp}$). We will say that two types (terms) are the same or identical, whenever they are the (syntactically) same type (term). We will say that two types A, A' are equal in context Γ (relative to T) whenever we can show $\vdash\ A \equiv A'$ ($\Gamma \vdash_T\ A \equiv A'$). We will say that a term t is *well-typed* (or well-formed) in a context Γ (and relative to T) iff we have $\Gamma \vdash\ t : A$ ($\Gamma \vdash_T\ t : A$) for some well-formed type A. In that case, we will also say that t *has type* A and that t is *of type* A. Similarly, we will say that two terms t, t' of type A are *equal* in context Γ (relative to T) whenever we can show $\Gamma \vdash\ t =_A t'$ ($\Gamma \vdash_T\ t =_A t'$). We will call a term a *formula* if it has type bool. Finally, we will say that a formula F is derivable in context Γ (relative to T) if $\Gamma \vdash\ F$ ($\Gamma \vdash_T\ F$).

2.2.1 Comparison to other HOL versions

Formalization in a logical framework Our definition of HOL differs from other definitions of HOL primarily by the explicit definition of the grammar. In particular, our grammar explic-

Theories and contexts:

$$\frac{}{\vdash \circ\ \mathsf{Thy}}\,\text{thyEmpty} \qquad \frac{\vdash T\ \mathsf{Thy}}{\vdash T,\ a\ \mathsf{tp}\ \mathsf{Thy}}\,\text{thyType} \qquad \frac{\vdash_T A\ \mathsf{tp}}{\vdash T,\ c : A\ \mathsf{Thy}}\,\text{thyConst} \qquad \frac{\vdash_T F : \mathsf{bool}}{\vdash T,\ F\ \mathsf{Thy}}\,\text{thyAxiom}$$

$$\frac{\vdash T\ \mathsf{Thy}}{\vdash_T .\ \mathsf{Ctx}}\,\text{ctxEmpty} \qquad \frac{\Gamma \vdash_T A\ \mathsf{tp}}{\vdash_T \Gamma,\ x : A\ \mathsf{Ctx}}\,\text{ctxVar} \qquad \frac{\Gamma \vdash_T F : \mathsf{bool}}{\vdash_T \Gamma,\ F\ \mathsf{Ctx}}\,\text{ctxAssume}$$

Lookup in theory and context:

$$\frac{a : \mathsf{tp}\ \text{in}\ T \quad \vdash_T \Gamma\ \mathsf{Ctx}}{\Gamma \vdash_T a\ \mathsf{tp}}\,\text{type} \qquad \frac{c : a\ \text{in}\ T \quad \vdash_T \Gamma\ \mathsf{Ctx}}{\Gamma \vdash_T c : A}\,\text{const} \qquad \frac{F\ \text{in}\ T \quad \vdash_T \Gamma\ \mathsf{Ctx}}{\Gamma \vdash_T F}\,\text{axiom}$$

$$\frac{x : A\ \text{in}\ \Gamma \quad \vdash_T \Gamma\ \mathsf{Ctx}}{\Gamma \vdash_T x : A}\,\text{var} \qquad \frac{F\ \text{in}\ \Gamma \quad \vdash_T \Gamma\ \mathsf{Ctx}}{\Gamma \vdash_T F}\,\text{assume}$$

Well-formedness and equality of types:

$$\frac{\Gamma \vdash_T A\ \mathsf{tp} \quad \Gamma \vdash_T B\ \mathsf{tp}}{\Gamma \vdash_T A \to B\ \mathsf{tp}}\,\text{arrow} \qquad \frac{\Gamma \vdash_T a\ \mathsf{tp}}{\Gamma \vdash_T a \equiv a}\,\text{congBase} \qquad \frac{\Gamma \vdash_T A \equiv A' \quad \Gamma \vdash_T B \equiv B'}{\Gamma \vdash_T A \to B \equiv A' \to B'}\,\text{cong}\to$$

Typing and equality of terms:

$$\frac{\Gamma \vdash_T s : A \quad \Gamma \vdash_T t : A}{\Gamma \vdash_T s =_A t : \mathsf{bool}}\,=\text{type} \qquad \frac{\Gamma,\ x : A \vdash_T t : B}{\Gamma \vdash_T (\lambda x : A.\ t) : A \to B}\,\text{lambda} \qquad \frac{\Gamma \vdash_T f : A \to B \quad \Gamma \vdash_T t : A}{\Gamma \vdash_T f\ t : B}\,\text{appl}$$

$$\frac{\Gamma \vdash_T A \equiv A' \quad \Gamma,\ x : A \vdash_T t =_B t'}{\Gamma \vdash_T \lambda x : A.\ t =_{A \to B} \lambda x : A'.\ t'}\,\text{cong}\lambda\ (\text{xi}) \qquad \frac{\Gamma \vdash_T t =_A t' \quad \Gamma \vdash_T f =_{A \to B} f'}{\Gamma \vdash_T f\ t =_B f'\ t'}\,\text{congAppl}$$

$$\frac{\Gamma \vdash_T t : A}{\Gamma \vdash_T t =_A t}\,\text{refl} \qquad \frac{\Gamma \vdash_T t =_A s}{\Gamma \vdash_T s =_A t}\,\text{sym} \qquad \frac{\Gamma \vdash_T (\lambda x : A.\ s)\ t : B}{\Gamma \vdash_T (\lambda x : A.\ s)\ t =_B s[x/t]}\,\text{beta} \qquad \frac{\Gamma \vdash_T t : A \to B \quad x\ \text{not in}\ \Gamma}{\Gamma \vdash_T t =_{A \to B} \lambda x : A.\ t\ x}\,\text{eta}$$

Typing, introduction and elimination rules for implication:

$$\frac{\Gamma \vdash_T F : \mathsf{bool} \quad \Gamma \vdash_T G : \mathsf{bool}}{\Gamma \vdash_T F \Rightarrow G : \mathsf{bool}}\,\Rightarrow\text{type} \qquad \frac{\Gamma \vdash_T F : \mathsf{bool} \quad \Gamma,\ F \vdash_T G}{\Gamma \vdash_T F \Rightarrow G}\,\Rightarrow\text{I} \qquad \frac{\Gamma \vdash_T F \Rightarrow G \quad \Gamma \vdash_T F}{\Gamma \vdash_T G}\,\Rightarrow\text{E}$$

Congruence for validity and boolean extensionality:

$$\frac{\Gamma \vdash_T F =_{\mathsf{bool}} F' \quad \Gamma \vdash_T F'}{\Gamma \vdash_T F}\,\text{cong}\vdash \qquad \frac{\Gamma \vdash_T p\ \mathsf{true} \quad \Gamma \vdash_T p\ \mathsf{false}}{\Gamma \vdash_T \forall x : \mathsf{bool}.\ p\ x}\,\text{boolExt}$$

Figure 1: HOL Rules

itly mentions theories which can declare additional declarations and axioms for a problem and grammars which declare the free variables that may occur in statements in that context and may give further assumptions for statements. In other definitions of logics, theories would usually not be part of the grammar and contexts might not be treated explicitely. For our work, it is necessary to make them explicit as we want to:

- reason about the differences and translations from extensions of HOL (and its fragments)
- implement those logics in a logical framework

A logical framework uses type-theory to formalize and reason about logics. For our implementation, a logical framework based on LF [19] in MMT [30] is used to define HOL and its extensions. This logical framework itself uses type theory and the Curry-Howard correspondence to formalize logics. For example, the validity judgment is represented as a function from formulae to types (we can think of it as giving us the type of validity proofs of the formula). Thus, a translation between logics in the framework is a function on the language of the framework.

Empty Types (I)HOL is traditionally built with the assumption that all types are non-empty. We can express this assumption in the axiom schema:

$$\exists x : A.\mathsf{true},$$

for all well-formed types A.

For DHOLP (and its fragments), on the other hand, this assumption is usually unsuitable because dependent types are commonly used together with empty types.

For the soundness results (soundness and completeness are formally defined in Definition 6), it is inconsequential whether the non-emptiness assumption is added to (I)HOL or not.

For the completeness result we have to assume that for each declaration

$$a : \Pi x_1 : A_1. \ \ldots \Pi x_n : A_n. \ \mathsf{tp}$$

in the theory T, there exists a term t of type $a\ t_1\ \ldots\ t_n$ for some t_i of type A_i respectively, i.e.:

$$\vdash_T \exists t_1 : A_i. \ldots \exists t_n : A_n[{}^{x_1}/_{t_1}] \ldots [{}^{x_{n-1}}/_{t_{n-1}}].\exists x : a\ t_1\ \ldots\ t_n.\mathsf{true}, \qquad \text{(nonEmpty')}$$

for

$$a : \Pi x_1 : A_1. \ \ldots \Pi x_n : A_n. \ \mathsf{tp}$$

in the theory T. This assumption is reasonable in DHOLP (and its fragments) and translates to the assumption that for each declaration $a : \mathsf{tp}$ in the translation $\overline{T}$ of the theory, there exists a well-typed term t_A in DHOLP s.t. its translation $\overline{t_A}$ has type $\overline{A}$ in HOL, i.e.:

$$\vdash_{\overline{T}} \overline{t_A} : \overline{A} \qquad \text{(nonEmpty)}$$

Since we are only really interested in translations of well-typed terms in HOL, it is reasonable to assume that each declared type should contain such a term.

Since these assumptions are both reasonable and obviously correspond to each other we will not discuss them much in the soundness and completeness proofs.

Choice of inference rules In our formulation we try to pick the minimal and simplest rules (without giving a definition to $\Rightarrow$) required to yield HOL with the usual properties, as this choice simplifies with the sound– and completeness proofs. In particular, we don't have a rewrite rule and need to conclude it from (cong$\vdash$) and the congruence rules. As rule (cong$\vdash$) allows us a valid formula on the right of an equality by the formula on the left of it but not vice versa. Thus, we can only use rule (beta) for beta expansions and not for beta reductions. Similarly, we cannot proof symmetry of term equality and have to assume it as an extra rule (at least on the type bool). Alternatively (and equivalently) we can add a modified version of rule (beta) (at least on the type bool) in which we flip the equlity in the conclusion or add a modified version of rule (cong$\vdash$) in which we flip the equality in the assumption of the rule. We also pick largely the same rules as present in the already existing formalization of HOL in MMT.

To ensure our formalization of HOL is equivalent to classical HOL, we will show the equivalence of our validity rules with e.g. the rules of a fragment of Q_0. This is shown in Lemma 5, after we prove some useful properties of our formalization of HOL.

Quantifier and implication as part of the grammar We include implication in the HOL and DHOL grammars (rather than defining it in terms of $=_{\mathsf{bool}}, \wedge$ and λ-functions), since we need dependent-implication in DHOL — meaning the term on the right of an implication needs to be well-typed only if the term on the left of it holds. Thus we need to make implication part of the grammr for DHOL. To simplify reasoning about the translation, we thus also include it in the HOL grammar. We don't include quantifiers in the grammar, since the rules for the quantifiers stay the same and we want to translate them to an expression equal to the translations of their definiens.

Comparison to HOL as implemented in MMT Our definition of HOL is equivalent to its definition in the MMT system. The syntax and grammar differ slightly, due to it being defined in a logical framework in such a way as to share as much of its definitions with related logics as possible. The inference rules used in the implementation of HOL in MMT include the rules given in this thesis, but also include some additional rules that can be proven from them (for instance the implementation of HOL in MMT includes the rules (propExt) and (extensionality) proven in Lemma 1).

Writing proofs in HOL We will usually write proofs over this (or similar calculi) in natural deduction style. In them, we proof new statements line by line, starting with (some of) our assumptions and ending in the claim. For each line we will write on the side by which rule it is concluded and in which previous lines in the proof the assumptions of that rule are given. To shorten proofs we sometimes write a rule and how an assumption of that rule is derived, instead of proving that assumption in a separate step. We may also not give justifications for assumptions to rules that are trivial to prove (often this will be the case for typing assumptions or equalities that can be derived from reflexivity or symmetry of equality).

Throughout this paper we will ignore contexthood assumptions of rules in proofs, as they can be checked easily and furthermore the only contexts that typically occur in proofs are extensions of

the context of the claim we want to proof.

Some consequence of the inference rules: The following lemma collects a few routine meta-theorems that we make use of later on:

Lemma 1. *Given the inference rules for HOL or IHOL (cfg. Figure 1), the following rules are admissible:*

$$\frac{\vdash_T \Gamma\ \mathsf{Ctx}}{\vdash\ T\ \mathsf{Thy}}\,ctxThy \qquad \frac{\Gamma \vdash_T A\ \mathsf{tp}}{\vdash_T \Gamma\ \mathsf{Ctx}}\,tpCtx \qquad \frac{\Gamma \vdash_T t : A}{\Gamma \vdash_T A\ \mathsf{tp}}\,typingTp \qquad \frac{\Gamma \vdash_T F}{\Gamma \vdash_T F : \mathsf{bool}}\,validTyping$$

$$\frac{\Gamma \vdash_T A \equiv A'}{\Gamma \vdash_T A' \equiv A}\equiv sym \qquad \frac{\Gamma \vdash_T A \equiv A' \quad \Gamma \vdash_T A' \equiv A''}{\Gamma \vdash_T A \equiv A''}\equiv trans$$

$$\frac{\Gamma \vdash_T s =_A t}{\Gamma \vdash_T s : A}\,eqTyping \qquad \frac{\Gamma \vdash_T F \Rightarrow G}{\Gamma \vdash_T F : \mathsf{bool}}\,implTypingL \qquad \frac{\Gamma \vdash_T F \Rightarrow G}{\Gamma \vdash_T G : \mathsf{bool}}\,implTypingR$$

$$\frac{\Gamma \vdash_T s : A \quad \Gamma \vdash_T s : A'}{\Gamma \vdash_T A \equiv A'}\,typesUnique \qquad \frac{\Gamma \vdash_T f\ t : B \quad \Gamma \vdash_T f : A \to B}{\Gamma \vdash_T t : A}\,typingWf$$

$$\frac{\Gamma \vdash_T t : A \quad \Gamma \vdash_T f\ t : B}{\Gamma \vdash_T f : A \to B}\,applType \qquad \frac{\Gamma,\ x : B \vdash_T s : A \quad \Gamma \vdash_T t : B}{\Gamma \vdash_T s[^x/_t] : A}\,rewriteTyping$$

$$\frac{\Gamma \vdash_T F : \mathsf{bool} \quad \Gamma \vdash_T G}{\Gamma,\ F \vdash_T G}\,monotonic\vdash \qquad \frac{\Gamma \vdash_T A\ \mathsf{tp} \quad \Gamma \vdash_T J \qquad \textit{for any statement } \vdash_T J}{\Gamma,\ x : A \vdash_T J}\,var\vdash$$

$$\frac{\Gamma,\ x : A \vdash_T F : \mathsf{bool}}{\Gamma \vdash_T \forall x : A.\ F : \mathsf{bool}}\forall type \qquad \frac{\Gamma,\ x : A \vdash_T F}{\Gamma \vdash_T \forall x : A.\ F}\forall I \qquad \frac{\Gamma \vdash_T \forall x : A.\ F \quad \Gamma \vdash_T t : A}{\Gamma \vdash_T F[^x/_t]}\forall E$$

$$\frac{\Gamma\ \mathsf{Ctx} \quad F\ in\ \Gamma}{\Gamma \vdash_T F : \mathsf{bool}}\,assTyping \qquad \frac{\Gamma \vdash_T t =_A t' \quad \Gamma \vdash_T A \equiv A' \quad \Gamma \vdash_T t : A'}{\Gamma \vdash_T t' : A'}\,cong:$$

$$\frac{\Gamma,\ F \vdash_T G \quad \Gamma,\ G \vdash_T F}{\Gamma \vdash_T F =_{\mathsf{bool}} G}\,propExt \qquad \frac{\Gamma,\ x : A \vdash_T f\ x =_B f'\ x}{\Gamma \vdash_T f =_{A\to B} f'}\,extensionality$$

$$\frac{\Gamma \vdash_T s =_A t \quad \Gamma \vdash_T t =_A u}{\Gamma \vdash_T s =_A u}\,trans \qquad \frac{\Gamma \vdash_T s =_A s' \quad \Gamma \vdash_T t =_A t'}{\Gamma \vdash_T (s =_A t) =_{\mathsf{bool}} (s' =_A t')} = cong$$

$$\frac{\Gamma \vdash_T A \equiv A' \quad \Gamma,\ x : A \vdash_T F =_{\mathsf{bool}} F'}{\Gamma \vdash_T \forall x : A.F =_{\mathsf{bool}} \forall x : A'.F'}\forall cong \qquad \frac{\Gamma \vdash_T F =_{\mathsf{bool}} F' \quad \Gamma \vdash_T G =_{\mathsf{bool}} G'}{\Gamma \vdash_T F \Rightarrow G =_{\mathsf{bool}} F' \Rightarrow G'}\Rightarrow cong$$

$$\frac{\Gamma \vdash_T F =_{\mathsf{bool}} F' \quad \Gamma \vdash_T F}{\Gamma \vdash_T F'}\vdash cong \qquad \frac{\Gamma \vdash_T F[^x/_t] \quad \Gamma \vdash_T t =_A t' \quad \Gamma,\ x : A \vdash_T F : \mathsf{bool}}{\Gamma \vdash_T F[^x/_{t'}]}\,rewrite$$

Furthermore we don't use rule (boolExt) in the proof of this lemma, except to derive rule (propExt) which is anyways assumed in IHOL, so our proof will show that these rules are admissible both in HOL and in IHOL.

This is proven in Appendix A.

Lemma 2. *If F is a well-typed IHOL (or HOL) formula we can show $\Gamma \vdash_T F =_{\mathsf{bool}} \mathsf{true}$ iff $\Gamma \vdash_T F$.*

Proof. We start with the ⇒ direction:

$\Gamma \vdash_T \lambda x : \mathsf{bool}.\ x =_{\mathsf{bool}\to\mathsf{bool}} \lambda x : \mathsf{bool}.\ x$	(refl),(lambda),(var)	(1)
$\Gamma \vdash_T \mathsf{true}$	definition of true,(1)	(2)
$\Gamma \vdash_T F =_{\mathsf{bool}} \mathsf{true}$	by assumption	(3)
$\Gamma \vdash_T F$	(cong$\vdash$),(3),(2)	(4)

Now the ⇐ direction:

$\Gamma \vdash_T F$	by assumption	(5)
$\Gamma \vdash_T F : \mathsf{bool}$	by assumption	(6)
$\Gamma \vdash_T \mathsf{true} : \mathsf{bool}$	by definition	(7)
$\Gamma,\ \mathsf{true} \vdash_T F$	(monotonic$\vdash$),(5)	(8)
$\Gamma,\ F \vdash_T \mathsf{true}$	(monotonic$\vdash$),(2)	(9)
$\Gamma \vdash_T F =_{\mathsf{bool}} \mathsf{true}$	(propExt),(6),(7),(9),(8)	(10)

□

Remark 3. In the following, we will omit derivations for typing assumptions of rules, whenever they are provable using the typing rules (=type), (lambda), (appl), (⇒type) and (var), as proving those is uninteresting and straightforward.

We will also often omit derivations for assumptions of rules, which are derivable using the rules (assume), (refl), (sym), (axiom) and (assume).

Furthermore, we will allow using previous statements for assumptions of rules even if their context is a subset of the context of the assumption of the rule. This will save us from having to invoke the rules (monotonic$\vdash$) and (var$\vdash$) separately to show the assumption of the rule.

Considering the rather lengthy and unreadable proof of Lemma 1 (in Appendix A), it becomes clear that these conventions allow us to shorten proofs significantly, while improving their readability.

Remark 4. In the following we will also use the following rules about the Boolean connectives, which can be easily derived from their definitions and the already shown rules:

$$\frac{\Gamma \vdash_T F \quad \Gamma \vdash_T G}{\Gamma \vdash_T F \wedge G}\wedge\mathrm{I} \qquad \frac{\Gamma \vdash_T F}{\Gamma \vdash_T F \vee G}\vee\mathrm{Il} \qquad \frac{\Gamma \vdash_T G}{\Gamma \vdash_T F \vee G}\vee\mathrm{Ir}$$

$$\frac{\Gamma \vdash_T F \wedge G}{\Gamma \vdash_T F}\wedge\mathrm{El} \qquad \frac{\Gamma \vdash_T F \wedge G}{\Gamma \vdash_T G}\wedge\mathrm{Er} \qquad \frac{\Gamma \vdash_T t : A \quad \Gamma \vdash_T F[{}^{x}/_{t}]}{\Gamma \vdash_T \exists x : A.F}\exists\mathrm{I}$$

As an example we can show rule (∧Er) as follows:

$\Gamma \vdash_T F \wedge G$	by assumption	(11)

$\Gamma \vdash_T \forall r : \mathsf{bool} \to \mathsf{bool} \to \mathsf{bool}.\ r\ F\ G =_{\mathsf{bool}} r\ \mathsf{true}\ \mathsf{true}$	definition,(11)	(12)
$\Gamma \vdash_T \lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ y : \mathsf{bool} \to \mathsf{bool} \to \mathsf{bool}$	(lambda),(lambda),(var)	(13)
$\Gamma \vdash_T (\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ y)\ F\ G =_{\mathsf{bool}}$ $(\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ y)\ \mathsf{true}\ \mathsf{true}$	(∀E),(12),(13)	(14)
$\Gamma \vdash_T (\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ y)\ F\ G =_{\mathsf{bool}} \mathsf{true}$	(rewrite),(rewrite),(14),(beta),(beta)	(15)
$\Gamma \vdash_T G =_{\mathsf{bool}} \mathsf{true}$	(rewrite),(rewrite),(15),(beta),(beta)	(16)
$\Gamma \vdash_T G$	Lemma 2,(16)	

Definition 2. For $F = (\lambda x : A.\ t)\ s$ being the application of a λ-function $\lambda x : A.\ t$ to an argument s, we will call $t[^x/_s]$ the beta reduced version of F and denote it by F^β. This notation is occasionally useful as it allows us to avoid writing out some lengthy formulae. Similarly for $F = \lambda x : A.\ f\ x$ being a λ-function applying a function to its bound variable, we will call the function f the eta reduced version of F and denote it by F^η. Both notations are taken to denote the identity function on terms of other shape. The composition of the functions $\cdot^\beta$ and $\cdot^\eta$ is denoted by $\cdot^{\beta\eta}$ (note that by Church-Rosser theorem the order in which the reductions are applied doesn't matter).

Using these conventions we can show the equivalence of HOL with a fragment of Q_0:

Lemma 5. *The rules for HOL as given in Figure 1 imply the axiom schemata and the rule of Q_0 (except for the fifth axiom schema the "axiom of description" which characterizes the description operator present in Q_0 but not in our formalization of HOL). Conversely, we can derive the validity rules for HOL (except for the lookup rules (axiom) and (assume) which relate provability with axioms and assumptions in context and theory and thus have no analogue in Q_0) from the axiom schemata and rules of Q_0.*

This is proven in Appendix B.

This lemma tells us that our formalization of HOL is in fact equivalent to classical HOL, so it makes sense to use it as a translation target.

2.2.2 TPTP

TPTP (Thousands of Problems for Theorem Provers) [38] is a library of test problems for automated theorem provers (ATPs) as well as a collection of languages to be used for input for ATP systems. Basically all major ATPs support input via TPTP languages. For our purposes, we are especially interested in the language TPTP THF [40] which roughly corresponds to HOL.

2.3 The LEO-III prover

Leo-III is a an automated theorem prover for (parametric) HOL and modal logics based on a resolution calculus with paramodulation at its core. LEO-III is also designed to cooperate with external provers or solvers which can be tasked with subproblems simultaneously.

Like all major provers LEO-III can take TPTP problems as input, it support all the main TPTP languages including TPTP THF. LEO-III (like MMT) is implemented in the Scala language, which simplifies its use within MMT and might facilitate a better integration in the future.

When LEO-III is called on a problem file (written in some TPTP language) it will first parse the problem, typecheck the TPTP formulae, remove redundant ones and possibly eliminate some formulae that are very unlikely to help at proving the conjecture.

The conjecture is then negated and LEO-III will try to prove a contradiction assuming this negation as well as the previous formulae.

For this a resolution calculus is used: LEO-III applies inference rules to derive further formulae from already existing formulae. The derived formulae are unified (i.e. rewritten into a more useful form in which it is also easier to check if such formulae are equal) and it is checked if a contradiction is found. These steps are repeated until a contradiction is found or the prover gives up. It is worth noticing that various of these steps only work in classical HOL not in its intuitionistic fragment IHOL, already the approach of negating the conjecture and trying to derive a contradiction doesn't work for proving a conjecture in IHOL.

2.4 The MMT system

MMT [30] is a foundation independent formal language, an implementation of this language and a LF based logical framework using this language to formalize formal content (mathematical or otherwise). As a language is specifies basic concepts like theories, (optionally typed or defined) declarations, terms, types and theory morphisms. MMT theories have a meta theory which specifies allowed syntax for basic concepts (like sets, types, equality, logical connectives, ...) and potentially provides semantics for them. We can think of the meta-theory as specifying the language the theory is written in (in the same way the ZFC axioms are written in first-order logic). MMT heavily relies on the concept of theory morphisms, which can be used to translate content in MMT theories along these morphisms. Building on the MMT language and system LATIN has been created, an archive in which various logics, type theories, set theories and their connections (expressed as theory-morphisms) have been formalized. In particular, classical higher-order logic has already been formalized in LATIN and we add further theories formalizing also the other logics mentioned in this thesis.

The MMT system includes a parser for the MMT language, a type-checker and simplifier, but limited proving support. It is implemented in a modular fashion that makes it easy to extend these components and by adding inference rules to the typechecker or simplifier or even to extend MMT with a new hammer tool.

Very recently, Wolff [43] provided a translation from MMT theories based on first– and higher-order logic to appropriate TPTP languages and build an automated theorem prover that uses this translation and calls LEO-III on the translated conjectures. Our implementation builds on (and generalizes) this architecture. We improve on his work mainly by adding formalizations of DHOL and DHOLP to MMT and implementing their translations into HOL. This way we extend the hammer for HOL problems in MMT to a hammer also for DHOL and DHOLP problems.

3 Translating dependently typed (I)HOL into simple typed (I)HOL

3.1 Dependently typed higher-order logic

3.1.1 Language

To obtain DHOL from HOL, only a few surgical changes are needed. The grammar is as follows where unchanged parts are shaded out:

T	$::=$	$* \mid T, Dec$	theories
Dec	$::=$	$c : A \mid a : (\Pi x : A.)^* \ \mathsf{tp} \mid F$	declarations
Γ	$::=$	$. \mid \Gamma, x : A \mid \Gamma, F$	contexts
A, B	$::=$	$a\ t_1 \dots t_n \mid \Pi x : A.\ B \mid \mathsf{bool}$	types
s, t, c, f, F, G	$::=$	$c \mid x \mid f\ t \mid \lambda x : A.\ t \mid s =_A t \mid F \Rightarrow G$	terms

Concretely, type constants a can take term arguments and consequently function types may depend on them. Moreover, while not apparent from the grammar, the list of variables in a DHOL context must not be reordered anymore: variables can occur in the types of later variables, and the well-formedness of types and assumptions may depend on previous assumptions.

The grammar for DIHOL is the same as for DHOL. It will be important to look at the DHOL fragment in which we disallow variables of function types (of positive arity) with return type bool in lambdas, context variables and Πs. We will say that *quantification*s over such types are disallowed. This fragment will be called *DHOL without boolean-valued variables* abbreviated as DHOL*. Disallowing quantifications over bool-valued function types in HOL (here we disallow such types on the left of an $\to$ rather than as the type of a variable bound by a Π) yields a logic which we will denote by HOL*.

Alternatively, we can look at HOL with external propositions (abbreviated HOLE), i.e. use a grammar like:

T	$::=$	$* \mid T, Dec$	theories
Dec	$::=$	$c : E \mid a : (\Pi x : A.)^* \ \mathsf{tp} \mid F$	declarations
Γ	$::=$	$. \mid \Gamma, x : A \mid \Gamma, F$	contexts
A, B	$::=$	$a\ t_1 \dots t_n \mid \Pi x : A.\ B$	internal types
E	$::=$	$A \mid \mathsf{bool}$	types
s, t, f, F, G	$::=$	$c \mid x \mid f\ t \mid \lambda x : A.\ t \mid s =_A t \mid F \Rightarrow G$	terms

Then bool would not be a tp, but instead be type. For our purposes this works similarly as DHOL* as formalized above, except that we additionally disallow quantifications on the type bool and bool-valued function types. HOLE is actually commonly used as a basis in dependent type theory (e.g. in Agda, Lean and Coq we have external proposition).

Definition 3. Let T be a DHOL (or DIHOL) theory and a a declaration in it. If a is of the form $a : \mathsf{tp}$ or $a : \Pi x_1 : A_1. \dots \Pi x_n : A_n.\ \mathsf{tp}$ we will call a a *simple type constructor* or *dependent type constructor* respectively. In both cases we will say a is a *type constructor*.

Any declaration in the theory that is not an axiom or a type constructor will be called a *term declaration*. Given a term declaration $c : \Pi y_1 : T_1. \dots \Pi y_s : T_s.\ B$ with either $B = \mathsf{bool}$ or

$B = b\ z_1\ \ldots\ z_t$ (for some type constructor b (and $t = 0$ is allowed here)) we will call B the *return type* of c. In the first case we will call c a *boolean-valued function* or a $s+1$-ary predicate (in case $s = 0$ we may also just call it a predicate), in the second case we will say that c is a *constructor* of the type constructor a. We will use the same terminology also for variables in contexts.

We will use the same terminology also for DHOLP and its fragments.

3.1.2 Inference rules

The calculus for DHOL (DIHOL) has the same judgments as the one for HOL (IHOL), so we only replace the Axiom nonEmpty by the Axiom nonEmpty' and make minor changes to some inference rules as shown in figure 2.

$$\frac{\vdash_T x_1 : A_1, \ldots, x_n : A_n\ \mathsf{Ctx}}{\vdash T,\ a : \Pi x_1 : A_1.\ \ldots \Pi x_n : A_n.\ \mathsf{tp}\ \mathsf{Thy}}\text{thyType'}$$

$$\frac{a : \Pi x_1 : A_1.\ \ldots \Pi x_n : A_n.\ \mathsf{tp}\ \text{in}\ T \quad \Gamma \vdash_T t_1 : A_1\ \ \ldots\ \ \Gamma \vdash_T t_n : A_n[{}^{x_1}/_{t_1}] \ldots [{}^{x_{n-1}}/_{t_{n-1}}]\ \ \vdash_T \Gamma\ \mathsf{Ctx}}{\Gamma \vdash_T a\ t_1\ \ldots\ t_n\ \mathsf{tp}}\text{type'}$$

$$\frac{\Gamma \vdash_T A\ \mathsf{tp} \quad \Gamma \vdash_T B\ \mathsf{tp}}{\Gamma \vdash_T \Pi x : A.\ B\ \mathsf{tp}}\text{pi} \qquad \frac{\Gamma \vdash_T A \equiv A' \quad \Gamma, x : A \vdash_T B \equiv B'}{\Gamma \vdash_T \Pi x : A.\ B \equiv \Pi x : A'.\ B'}\text{congΠ}$$

$$\frac{a : \Pi x_1 : A_1.\ \ldots \Pi x_n : A_n.\ \mathsf{tp}\ \text{in}\ T \quad \Gamma \vdash_T s_1 =_{A_1} t_1\ \ \ldots\ \Gamma \vdash_T s_n =_{A_n[{}^{x_1}/_{t_1}]\ldots[{}^{x_{n-1}}/_{t_{n-1}}]} t_n\ \ \vdash_T \Gamma\ \mathsf{Ctx}}{\Gamma \vdash_T a\ s_1\ \ldots\ s_n \equiv a\ t_1\ \ldots t_n}\text{congBase'}$$

$$\frac{\Gamma,\ x : A \vdash_T t : B}{\Gamma \vdash_T (\lambda x : A.\ t) : \Pi x : A.\ B}\text{lambda'} \qquad \frac{\Gamma \vdash_T f : \Pi x : A.\ B \quad \Gamma \vdash_T t : A}{\Gamma \vdash_T f\ t : B[{}^{x}/_{t}]}\text{appl'}$$

$$\frac{\Gamma \vdash_T t =_A t' \quad \Gamma \vdash_T A \equiv A' \quad \Gamma \vdash_T t : A'}{\Gamma \vdash_T t' : A'}\text{cong:} \qquad \frac{\Gamma \vdash_T F : \mathsf{bool} \quad \Gamma,\ F \vdash_T G : \mathsf{bool}}{\Gamma \vdash_T F \Rightarrow G : \mathsf{bool}}\Rightarrow\text{type'}$$

$$\frac{\Gamma \vdash_T t =_A t' \quad \Gamma \vdash_T f =_{\Pi x:A.\ B} f'}{\Gamma \vdash_T f\ t =_B f'\ t'}\text{congAppl'} \qquad \frac{\Gamma \vdash_T A \equiv A' \ \Gamma,\ x : A \vdash_T t =_B t'}{\Gamma \vdash_T \lambda x : A.\ t =_{\Pi x:A.\ B} \lambda x : A'.\ t'}\text{congλ'}$$

$$\frac{\Gamma \vdash_T t : \Pi x : A.\ B}{\Gamma \vdash_T t =_{\Pi x:A.\ B} \lambda x : A.\ t\ x}\text{etaPi}$$

Figure 2: Modified Rules in DHOL and DIHOL

Routinely, we replace all rules for → types with the corresponding rules for Π-types, e.g., the rule (cong→) with the rule (congΠ), or the rule (arrow) with the rule (pi). Similarly, all rules for base types a need to be extended to handle arguments.

More subtly, types may now depend on arguments making type equality non-trivial, thus we need to explicitly add rule (cong:) (in HOL we can prove this rule). Furthermore, we replace rule (congBase) with rule (congBase') and rule (type) with rule (type'), as the former rules only covers the case of a simple type constructor whereas the modified rules also cover the case of a dependent type constructor.

Rule (congBase') is the rule that makes type equality undecidable in DHOL as deciding its applicability now requires deciding a term equality. Then, rule (cong:) makes also typing undecidable as deciding its applicability requires deciding type equality. This is illustrated in Example 1:

Example 1 (Undecidability example for rule (congBase'):). Let φ denote any undecidable formula in DHOL and consider the theory T:

$$a : \Pi x : \mathsf{bool}.\ \mathsf{tp}$$
$$t : a\ \varphi$$

The type-equality statement $\vdash_T a\ \mathsf{true} \equiv a\ \varphi$ and thus (by the rules (cong:) and (typesUnique)) the typing statement $\vdash_T t =_{a\ \mathsf{true}} t : \mathsf{bool}$ holds exactly if $\vdash_T \varphi =_{\mathsf{bool}} \mathsf{true}$ or equivalently if φ provably valid. Since φ is undecidable, both statements are undecidable.

Finally, we replace rule (⇒type) with rule (⇒type') for dependent implication: this allows the well-formedness of consequent to depend of the validity of the antecedent. The proof rules for implications are unchanged.

This dependence is illustrated in the following Example 2. The conjecture would not have been well-typed with the rule (⇒type) instead of (⇒type'). This assumption is required in order to invoke rule (congBase') showing that the types $\mathtt{mor}\ x\ x$ and $\mathtt{mor}\ y\ y$ of the terms on both sides of the equality $\mathtt{id}_x =_{\mathtt{mor}\ x\ x} \mathtt{id}_y$ are actually equal. By rule (cong:) it follows that $\mathtt{id}_y$ then also has type $\mathtt{mor}\ x\ x$ and the equality is well-typed.

Example 2 (Category Theory). The theory of a category contains the type constants *obj* for objects and $\mathtt{mor}\ a\ b$ for morphisms, the constants *id* and comp for identity and composition, and the axioms for neutrality and associativity. We omit the latter for brevity. We also use → to denote Π-types in which we don't want to name the argument.

$$\begin{aligned}
\mathtt{obj} &: \mathsf{tp}\\
\mathtt{mor} &: \Pi x : \mathtt{obj}.\ \Pi y : \mathtt{obj}.\ \mathsf{tp}\\
\mathtt{id} &: \Pi a : \mathtt{obj}.\ \mathtt{mor}\ a\ a\\
\mathtt{comp} &: \Pi a : \mathtt{obj}.\ \Pi b : \mathtt{obj}.\ \Pi c : \mathtt{obj}.\ \mathtt{mor}\ a\ b \to \mathtt{mor}\ b\ c \to \mathtt{mor}\ a\ c\\
\forall x : \mathtt{obj}.\ \forall y : \mathtt{obj}.\ \forall m : \mathtt{mor}\ x\ y.\ & \ m \circ \mathtt{id}_x =_{\mathtt{mor}\ x\ y} m\\
\forall x : \mathtt{obj}.\ \forall y : \mathtt{obj}.\ \forall m : \mathtt{mor}\ x\ y.\ & \ \mathtt{id}_y \circ m =_{\mathtt{mor}\ x\ y} m
\end{aligned}$$

Here, to enhance readability, we use the notations $\mathtt{id}_x$ for $\mathtt{id}\ x$ and $h \circ g$ for $\mathtt{comp}\ \_\ \_\ \_\ g\ h$ where the _ denote omitted arguments of type obj.

We can then formalize the property of the pair (f, F) being a covariant endo-functor using a predicate:

$$\begin{aligned}
\mathtt{endoFunctorial} : &\Pi f : \mathtt{obj} \to \mathtt{obj}.\\
&\Pi F : \big(\Pi a : \mathtt{obj}.\ \Pi b : \mathtt{obj}.\ \mathtt{mor}\ a\ b \to \mathtt{mor}\ (f\ a)\ (f\ b)\big).\ \mathsf{bool}
\end{aligned}$$

with a defining axiom

$$\begin{aligned}&\forall f : \mathtt{obj} \to \mathtt{obj}.\, \forall F : \big(\Pi a : \mathtt{obj}.\, \Pi b : \mathtt{obj}.\, \mathtt{mor}\ a\ b \to \mathtt{mor}\ (f\ a)\ (f\ b)\big).\\ &\quad \mathtt{endoFunctorial}\ f\ F =_{\mathsf{bool}}\\ &\qquad \forall x : \mathtt{obj}.\, F\ x\ x\ \mathtt{id}_x =_{\mathtt{mor}\ (f\ x)\ (f\ x)} \mathtt{id}_{f\ x} \wedge\\ &\qquad \forall x : \mathtt{obj}.\, \forall y : \mathtt{obj}.\, \forall z : \mathtt{obj}.\, \forall g : \mathtt{mor}\ x\ y.\, \forall h : \mathtt{mor}\ y\ z.\\ &\qquad\quad (F\ y\ z\ h) \circ (F\ x\ y\ g) =_{\mathtt{mor}\ (f\ x)\ (f\ z)} F\ x\ z\ (h \circ g)\end{aligned}$$

Finally we consider the conjecture

$$\vdash_T \forall x : \mathtt{obj}.\, \forall y : \mathtt{obj}.\, x =_{\mathtt{obj}} y \Rightarrow \mathtt{id}_x =_{\mathtt{mor}\ x\ x} \mathtt{id}_y$$

This is a straightforward theorem stating that equal objects have equal identity morphisms. This theorem is well-formed only because of the typing rule for dependent implication.

Observe that D(I)HOL is a conservative extension of (I)HOL via the usual abbreviation of $A \to B$ for $\Pi x : A.\ B$ if x does not occur free in B. In fact, we can recover (I)HOL as the fragment of D(I)HOL in which theories don't declare any dependent type constructors (since rule (type') can be used only for those to show typehood of a dependent type).

3.2 Translating DHOL* into HOL* and DIHOL into IHOL

In this section, we define a translation function Ψ (denoted by $\bar{\cdot}$) that maps DHOL-syntax to HOL-syntax and a slightly different translation that maps DHOL*-syntax to HOL-syntax. The intuition behind the translations is to erase type dependencies by translating dependent types $a\ t_1\ \ldots,\ t_n$ into simple types a and replacing Π by $\to$.

In this subsection, we will explain the idea for the translations, the subsequent two subsections will define the translations formally. We will discuss the first slightly simpler translation first and then describe how the second translation differs from it.

We choose for our first translation the simplest (shallow) translation for which we can actually hope to proof its soundness and completeness. In particular, the translation maps D(I)HOL types to (I)HOL types corresponding to the original types as closely as possible.

This is useful, to make the translation result simpler, more readable and most importantly simpler to use for existing ATPs. In particular, it is important that equalities are translated to equalities, as HOL provers are optimized to efficiently work with them.

The translation follows the same ideas as the translations in [11, 4, 35]: erase dependencies, add $n+1$-ary predicates for n-ary function symbols, then use them to relativize variables and equalities.

This idea to compensate for the non-injectivity on type level of the translation by relativizing formulas was first introduced by Oberschelp in [25], who also introduced the word relativization for it.

The intended invariants of the translation are as follows:

DIHOL	IHOL
theory	theory with additional typing functions and axioms
context Γ	context $\overline{\Gamma}$ with additional assumptions $A^?$ for each variable $x : A$ in Γ
type A	type $\overline{A}$ and typing function $A^? : \overline{A} \to \mathsf{bool}$
term $t : A$	term $\bar{t} : \overline{A}$ satisfying $A^? \bar{t}$

Table 1: Summary of intended invariants for (all) our translations

The translation is defined formally in Definition 1.

Translation 1 generates an additional $n+1$-ary typing predicate $a^?$ for each n-ary (dependent) type declaration a. It also generates a typing axiom for each constructor c (of A) and a typing assumption for each variable x of type A. To relativize Equalities are relativized by the translation: for equalities over non-function types we do this by adding the typing axioms for the terms on both sides of the equality (see IT24). For equalities over function types we need to translate differently (see IT23), taking advantage of functional extensionality (extensionality), otherwise the translation will be neither sound nor complete (as discussed in Subsubsection 3.2.3).

If we use this translation to translate into HOL (rather than to IHOL), the typing predicate for the type bool becomes trivial (by rule (boolExt)). It follows that function types with (eventual) return type bool also have trivial typing predicates, so the their applications are all true and hence contain no information at all. This leads to typing and type-equality becoming incomplete for bool-valued function types (of positive arity) as can be seen from Example 3:

Example 3. Consider the DHOL theory T:

$$a : \Pi x : \mathsf{bool}.\ \mathsf{tp}$$
$$c : a\ \mathsf{false}$$

and the false judgment $\vdash_T c : a\ \mathsf{true}$. Since the typing predicate $a^?$ is trivial we can prove $a^?\ \mathsf{true}\ c$. This violates completeness of Translation 1 w.r.t. typing.

This may not be a serious problem in practice as we don't use the typing predicates for type-checking anyways and the translation remains sound and complete wr.t. validity. Nevertheless, it motivates studying Translation 2 — a more faitful translation — which is defined in Definition 2.

To address the issue, Translation 2 will not use a typing predicate $\mathsf{bool}^?$, but instead use a meta-level abbreviation $\mathsf{bool}^?\ t$ by induction on the shape of $t^{\beta\eta}$. As this abbreviation cannot be expressed as the application of a predicate, rule (boolExt) cannot be used to show that its triviality.

Unfortunately, we cannot define this abbreviation for function applications of type bool, if the function being applied is a variable (otherwise we yield a circular definition). Therefore, we define Translation 2 as a translation from DHOL* into HOL* (and hence also to HOL). Using this abbreviation instead of the typing predicate $\mathsf{bool}^?$ in the definition of soundness and completeness, the soundness and completeness proofs are essentially unaffected. We dont know if this translation is actually complete w.r.t. typing.

Inspecting the definition of the abbreviation $\mathsf{bool}^?$, we notice that we can actually generalize a version of it to arbitrary types. Since equalities and implications are always of type bool, we only have to consider the case of atomic terms and function applications. Since we no longer need to worry about rule (boolExt) making the predicate trivial, we can define them as actual predicates in DHOL rather than meta-level abbreviations (although their translations via the Translation 2 may use the abbreviation $\mathsf{bool}^?$ making their translations meta-level abbreviations). We will call them the replacement predicates as their translations will be used to replace the typing predicates in parts of the completeness proof. They are defined as follows:

Definition 4. Let Γ be a fixed DHOL context, T a DHOL theory and φ a formula in Γ relative to T. Let t be a subterm of φ of type well-formed type A. Assume that $A = \mathsf{bool}$ and $t^{\beta\eta} = p\ t_1\ \dots\ t_q$ (where $q > 0$ and p not a function application) or that $A = a\ t_1\ \dots\ t_n$ (here $n = 0$ is allowed) for a type constructor $a : \Pi x_1 : A_1.\ \dots \Pi x_n : A_n.\ \mathsf{tp}$ (if $A = \mathsf{bool}$ we take $\mathsf{bool} : \mathsf{tp}$ to be the type constructor a). Then we define the *replacement predicate* p_A of type $\Pi x : A.\ \mathsf{bool}$ by:

$$\begin{aligned}
p_A := \lambda t : A.\ \big(& (\exists r_{1,1} : T_{1,1}. \dots \exists r_{1,p_1} : T_{1,p_1}. t_1 =_{A_1} f_1(r_{1,1}, \dots, r_{1,p_1}) \\
& \quad \wedge \dots \wedge t_n =_{A_n} f_n(r_{1,1}, \dots, r_{1,p_1}) \wedge t =_A c_1\ r_{1,1}\ \dots\ r_{1,p_1}) \\
& \vee (\exists r_{2,1} : T_{2,1}. \dots \exists r_{2,p_2} : T_{2,p_2}. t_1 =_{A_1} f_1(r_{2,1}, \dots, r_{2,p_2}) \\
& \quad \wedge \dots \wedge t_n =_{A_n} f_n(r_{2,1}, \dots, r_{2,p_2}) \wedge t =_A c_2\ r_{2,1}\ \dots\ r_{2,p_2}) \\
& \quad \vdots \\
& \vee (\exists r_{m,1} : T_{m,1}. \dots \exists r_{m,p_m} : T_{m,p_m}. t_1 =_{A_1} f_1(r_{m,1}, \dots, r_{m,p_m}) \\
& \quad \wedge \dots \wedge t_n =_{A_n} f_n(r_{m,1}, \dots, r_{m,p_m}) \wedge t =_A c_m\ r_{m,1}\ \dots\ r_{m,p_m}) \\
& \quad \vee t =_A v_1 \\
& \quad \vdots \\
& \quad \vee t =_A v_{q(t)} \big)
\end{aligned}$$

where

$$\begin{aligned}
& c_1 : \Pi r_{1,1} : T_{1,1}.\ \dots \Pi r_{1,p_1} : T_{1,p_1}.\ a\ f_{1,1}(r_{1,1}, \dots, r_{1,p_1})\ \dots\ f_{n,1}(r_{1,1}, \dots, r_{1,p_1}), \\
& c_2 : \Pi r_{2,1} : T_{2,1}.\ \dots \Pi r_{2,p_2} : T_{2,p_2}.\ a\ f_{1,2}(r_{2,1}, \dots, r_{2,p_2})\ \dots\ f_{n,2}(r_{2,1}, \dots, r_{2,p_2}), \\
& \vdots \\
& c_m : \Pi r_{m,1} : T_{m,1}.\ \dots \Pi r_{m,p_m} : T_{m,p_m}.\ a\ f_{1,m}(r_{m,1}, \dots, r_{m,p_m})\ \dots\ f_{n,m}(r_{m,1}, \dots, r_{m,p_m})
\end{aligned}$$

are the constructors of a in theory T and context Γ and $v_1, \dots, v_{q(x)}$ are the variables of type A in the scope of t (meaning the bound and free variables at the position in which t occurs in φ).

On bool (and with t not beta-eta reducible to a function application) and on Π-types (and later predicate-subtypes) we define the p_A by:

$$p_{\Pi x:A.\ B}\ f \quad := \forall x : A.\ p_A\ x \Rightarrow p_B\ (f\ x) \tag{R1}$$

$$p_{\mathsf{bool}}\ (t_1 \Rightarrow t_2) \quad := p_{\mathsf{bool}}\ t_1 \wedge p_{\mathsf{bool}}\ t_2 \tag{R2}$$

$$p_{\mathsf{bool}}\,(s =_A t) \quad :=p_A\, s \wedge p_A\, t \qquad A \text{ not function type} \qquad \text{(R3)}$$
$$p_{\mathsf{bool}}\, t \quad :=p_{\mathsf{bool}}\, t^{\beta\eta} \qquad t \text{ is beta or eta reduceable} \qquad \text{(R4)}$$
$$p_{\mathsf{bool}}\, x \quad :=\mathsf{true} x \text{ variable} \qquad \text{(R5)}$$
$$p_{\mathsf{bool}}\, c \quad :=\mathsf{true} \qquad \text{else} \qquad \text{(R6)}$$

These replacement predicates will play a very important role in the completeness proof.

We will see later, in IHOL (and hence also in HOL) the typing predicates are provably equal to the translation of the corresponding replacement predicates.

We will first prove soundness and completeness (w.r.t. validity) for Translation 1, then the sound– and completeness for Translation 2 will be mostly analogous.

3.2.1 Translation 1 from DIHOL into IHOL

Translation definition 1 (Translation from DIHOL to IHOL)**.** We define a translation from DIHOL to IHOL syntax by induction on the Grammar.

The cases for theories and contexts are:

$$\overline{\circ} := \circ,\ \mathsf{bool}^?\ :\ \mathsf{bool} \to \mathsf{bool},\ \mathsf{bool}^?\ \mathsf{true},\ \mathsf{bool}^?\ \mathsf{false} \qquad \text{(IT1)}$$
$$\overline{T,\ a\ :\ \Pi x_1\ :A_1.\ \ldots \Pi x_n\ :A_n.\ \mathsf{tp}} := \overline{T},\ a\ :\ \mathsf{tp},\ a^?\ :\ \overline{A_1} \to \ldots \to \overline{A_n} \to a \to \mathsf{bool} \qquad \text{(IT2)}$$
$$\overline{T,\ c\ :\ A} := \overline{T},\ c\ :\ \overline{A},\ A^?\ c \qquad \text{(IT3)}$$
$$\overline{T,\ F} := \overline{T},\ \overline{F} \qquad \text{(IT4)}$$

$$\cdot := \cdot \qquad \text{(IT5)}$$
$$\overline{\Gamma, x\ :\ A} := \overline{\Gamma},\ x\ :\ \overline{A},\ A^?\ x \qquad \text{(IT6)}$$
$$\overline{\Gamma, F} := \overline{\Gamma},\ \overline{F} \qquad \text{(IT7)}$$

The cases of $\overline{A}$ and $A^?\ t$ for types A and terms $t\ :\ \overline{A}$ are:

$$\overline{(a\ t_1\ \ldots\ t_n)} := a \qquad \text{(IT8)}$$
$$(a\ t_1\ \ldots\ t_n)^?\ t := a^?\ \overline{t_1}\ \ldots\ \overline{t_n}\ t \qquad \text{(IT9)}$$
$$\overline{\Pi x\ :\ A.\ B} := \overline{A} \to \overline{B} \qquad \text{(IT10)}$$
$$(\Pi x\ :\ A.\ B)^?\ f\ := \forall x\ :\ \overline{A}.\ A^?\ x \Rightarrow B^?\ \left(f\ x\right) \qquad \text{(IT11)}$$
$$\overline{\mathsf{bool}} := \mathsf{bool} \qquad \text{(IT12)}$$

For $t^{\beta\eta}$ not the application of a function with (eventual) return type bool to all its arguments, we treat $\mathsf{bool}^?\ t$ not as a function application of $\mathsf{bool}^?$ to t, but instead as an abbreviation for:

$$\mathsf{bool}^?\ (t_1 \Rightarrow t_2) \quad :=\mathsf{bool}^?\ t_1 \wedge \mathsf{bool}^?\ t_2 \qquad \text{(IT13)}$$
$$\mathsf{bool}^?\ (s =_A t) \quad :=A^?\ s \wedge A^?\ t \qquad A \text{ not function type} \qquad \text{(IT14)}$$

$$\mathsf{bool}^?\ t := \mathsf{bool}^?\ t^{\beta\eta} \quad t \text{ is beta or eta reduceable} \tag{IT15}$$
$$\mathsf{bool}^?\ x := \mathsf{true} \quad x \text{ variable} \tag{IT16}$$
$$\mathsf{bool}^?\ c := \mathsf{true} \quad x \text{ constant} \tag{IT17}$$

The cases for terms are:

$$\overline{c} := c \tag{IT18}$$
$$\overline{x} := x \tag{IT19}$$
$$\overline{\lambda x : A.\ t} := \lambda x : \overline{A}.\ \overline{t} \tag{IT20}$$
$$\overline{f\ t} := \overline{f}\ \overline{t} \tag{IT21}$$
$$\overline{F \Rightarrow G} := \overline{F} \Rightarrow \overline{G} \tag{IT22}$$
$$\overline{f =_{\Pi x:A.\ B} g} := \forall x : \overline{A}.\ A^?\ x \Rightarrow \mathsf{Relat}_B[\overline{f}\ x =_{\overline{B}} \overline{g}\ x] \tag{IT23}$$
$$\overline{s =_A t} := \overline{s} =_{\overline{A}} \overline{t} \wedge A^?\ \overline{s} \wedge A^?\ \overline{t} \quad A \text{ not function type} \tag{IT24}$$

Here the abbreviation $\mathsf{Relat}_A[x =_{\overline{A}} x']$ is used to denote the *relativization* of the equality $x =_{\overline{A}} x'$. We define the relativization of an equality by induction on the type A:

$$\mathsf{Relat}_A[t =_{\overline{A}} t'] := \mathsf{R}[t] =_{\overline{A}} \mathsf{R}[t]' \wedge A^?\ \mathsf{R}[t] \wedge A^?\ \mathsf{R}[t'] \quad A \text{ not function type} \tag{IT25}$$
$$\mathsf{Relat}_{\Pi x:A.\ B}[f =_{\overline{\Pi x:A.\ B}} f'] := \forall x : \overline{A}.\ A^?\ x \Rightarrow \mathsf{Relat}_B[f\ x =_{\overline{B}} f\ x] \quad \text{else} \tag{IT26}$$

And the abbreviation $\mathsf{R}[s]$ denoting the *relativization* of the term s is defined by:

$$\mathsf{R}[s] := s \quad \text{if } s = \mathsf{R}[t] \tag{IT27}$$

And for s not of the form $\mathsf{R}[t]$:

$$\mathsf{R}[s =_A t] := \mathsf{Relat}_A[\overline{s} =_{\overline{A}} \overline{t}] \tag{IT28}$$
$$\mathsf{R}[f\ t] := \mathsf{R}[f]\ \mathsf{R}[t] \tag{IT29}$$
$$\mathsf{R}[F \Rightarrow G] := \mathsf{R}[F] \Rightarrow \mathsf{R}[G] \tag{IT30}$$
$$\mathsf{R}[\lambda x : A.\ s] := \lambda x : A.\ \mathsf{R}[s] \tag{IT31}$$
$$\mathsf{R}[x] := x \tag{IT32}$$
$$\mathsf{R}[c] := c \tag{IT33}$$

The relativization of a term is well-defined, since its definition recurses only into the definition of the relativization of an equality over a type of smaller arity (for equalities over types of positive arity) and into relativizations of subterms (for all other terms).

3.2.2 Translation 2 from DHOL into HOL

Translation definition 2 (Translation from DHOL* into HOL*)**.** We define this translation from DHOL* into HOL* syntax by induction on the Grammar. The cases in the definition are mostly

the same as in Translation 1. The only difference between this translation and Translation 1 is that we introduce no typing predicate $\mathsf{bool}^?$ but instead define $\mathsf{bool}^?$ of a function applications as a meta-level abbreviation.

For $t^{\beta\eta}$ not the application of a predicate to all its arguments we define $\mathsf{bool}^?\ t$ as in Translation 1. If $t^{\beta\eta}$ is the application $t' := p\ t_1\ \ldots\ t_n$ of a predicate p (from the theory or context) with (eventual) return type bool to all its arguments $t_1, \ldots, t_{p_i}$, it follows that $p =: c_i$ must be a variable or a constructor in the context, scope or in the theory respectively or that t' is the application of a λ-function to an argument that is ill-typed or of a type not equal to the type quantified over in the λ-function (otherwise t' beta reducible). In the latter case, the term t' itself is ill-typed in HOL and we define $\mathsf{bool}^?\ t'$ to be false.

Otherwise, we define the meta-level abbreviation $\mathsf{bool}^?\ t := \mathsf{bool}^?\ t' = \mathsf{bool}^?\ \left(p\ t_1\ \ldots\ t_n\right)$ by:

$$\mathsf{bool}^?\ c_i\ t_1\ \ldots\ t_{p_i} :=T_{i,1}{}^?\ t_1 \wedge \ldots \wedge \left(T_{i,p_i}[{}^{r_{i,1}}/_{t_1}] \ldots [{}^{t_{i,p_i-1}}/_{t_{p_i-1}}]\right)^?\ t_{p_i}, \tag{17}$$

where

$$\begin{aligned}
&c_1 : \Pi r_{1,1} : T_{1,1}.\ \ldots \Pi r_{1,p_1} : T_{1,p_1}.\ \mathsf{bool},\\
&c_2 : \Pi r_{2,1} : T_{2,1}.\ \ldots \Pi r_{2,p_2} : T_{2,p_2}.\ \mathsf{bool},\\
&\vdots\\
&c_m : \Pi r_{m,1} : T_{m,1}.\ \ldots \Pi r_{m,p_m} : T_{m,p_m}.\ \mathsf{bool}
\end{aligned}$$

are the constructors (and variables) of return type bool in the DHOL* theory, context and scope. We will sometimes also denote this abbreviation $\mathsf{bool}^?$ by p_{bool} if we are comparing the two translations and the abbreviation might otherwise get confused with the typing predicate.

A complete self-contained definition of Translation 2 is given in Appendix C.

3.2.3 Non-injectivity of the translation and spurious terms

Inspecting the Translation 1, we notice two interesting cases, namely the translation of equalities over function types and the treatment of the typing predicate over type bool.

The case IT23 needs to be considered separately (from case IT23), to ensure the soundness of the translation, otherwise the rule (congλ') makes the translation unsound. When proving soundness of the translation, we can easily treat the inductive step case for the rule using the translation by case IT23, but there is no obvious way to prove the step when translating according to IT24. This unsoundness can be shown by considering the counterexample:

Example 4. Consider the theory S:

$$b : \mathsf{tp} \tag{1}$$
$$c_1 : b \tag{2}$$
$$c_2 : b \tag{3}$$

$$\left(c_1 =_b c_2\right) =_{\mathsf{bool}} \mathsf{false} \tag{4}$$
$$a : \Pi x : b.\, \mathsf{tp} \tag{5}$$
$$\forall\, x : a\, c_1.\, \mathsf{false} \tag{6}$$
$$d : a\, c_2 \tag{7}$$

and the conjecture (for some well-formed context Γ)

$$\Gamma \vdash_S \lambda x : a\, c_1.\, c_1 =_{\Pi x : a\ c_1.\ b} \lambda x : a\, c_1.\, c_2.$$

We can prove this conjecture in DIHOL as follows:

$$\Gamma,\ x : a\, c_1 \vdash_S \mathsf{false} \qquad (\forall\mathrm{E}),(6),(\mathrm{var}) \tag{8}$$
$$\Gamma,\ x : a\, c_1 \vdash_S c_1 =_b c_2 \qquad (\mathrm{cong}\vdash),(4),(8) \tag{9}$$
$$\Gamma \vdash_S \lambda x : a\, c_1.\, c_1 =_{\Pi x : a\ c_1.\ b} \lambda x : a\, c_1.\, c_2 \qquad (\mathrm{cong}\lambda),(9) \tag{10}$$

However, the translation of the conjecture according to case T24 is:

$$\overline{\Gamma} \vdash_{\overline{S}} \lambda x : a.\, c_1 =_{a\to b} \lambda x : a.\, c_2 \wedge \Pi x : a\, c_1.\, b^? \ \lambda x : a.\, c_1 \wedge \Pi x : a\, c_1.\, b^? \ \lambda x : a.\, c_2$$

This translated conjecture is not provable. Already the statement of validity of its first conjunct

$$\overline{\Gamma} \vdash_{\overline{S}} \lambda x : a.\, c_1 =_{a\to b} \lambda x : a.\, c_2 \tag{11}$$

is a contradiction. This can be seen as follows:

$$\overline{\Gamma} \vdash_{\overline{S}} \left(\lambda x : a.\, c_1\right)\ d =_b \left(\lambda x : a.\, c_2\right)\ d \qquad (\mathrm{congAppl}),(11),(\mathrm{refl}) \tag{12}$$
$$\overline{\Gamma} \vdash_{\overline{S}} \left(\lambda x : a.\, c_1\right)\ d =_b c_2 \qquad (\mathrm{rewrite}),(12),(\mathrm{beta}) \tag{13}$$
$$\overline{\Gamma} \vdash_{\overline{S}} c_1 =_b c_2 \qquad (\mathrm{rewrite}),(13),(\mathrm{beta})$$

This contradicts (4) which (by definition of $\neg$) beta reduces to the negation of this equality.

Using the simpler translation of equality in Translation 1 also makes the translation incomplete, as seen from the conjecture:

$$\left(\forall\, p : a\, s \to \mathsf{bool}.\, p =_{a\ s\to\mathsf{bool}} \lambda x : a\, s.\, \mathsf{true}\right) =_{\mathsf{bool}} \left(\forall\, p : a\, t \to \mathsf{bool}.\, p =_{a\ t\to\mathsf{bool}} \lambda x : a\, t.\, \mathsf{true}\right),$$

for $a\ s$ empty and $a\ t$ non-empty.

In the following, we will give a necessary (injectivity) condition for the completeness (w.r.t. validity) of the translation and show why Translation 1 satisfies it.

Since the translation looses type information to dependency-erasure we can write down conjectures that are not even well-typed in DIHOL, but whose translation is well-typed and in fact provable. Perhaps the simplest such example is the DIHOL theory T:

$$b : \mathsf{tp}, c_1 : b, c_2 : b, a : \Pi x : b.\, \mathsf{tp}$$

with the conjecture

$$\vdash_T \lambda x : a\ c_1.\ x =_{a\ c_1} \lambda x : a\ c_2.\ x$$

This conjecture is not well-typed (as $c_1 =_b c_2$ not provable). Its translation however becomes:

$$\vdash_{\overline{T}} \forall\ x : \overline{a}.\ a^?\ \overline{c_1}\ x \Rightarrow \mathsf{Relat}_{a\ c_1}[x =_{\overline{a\ c_1}} x],$$

which is follows by the rules (refl), (assume), ($\wedge$I), ($\Rightarrow$I) and ($\forall$I).

The problem is that terms of different DIHOL types can have equal images in IHOL of same type. Thus the translation function is not complete if we allow for ill-typed DIHOL terms that are translated to well-typed IHOL terms.

Following this observation, we introduce the notions of spurious terms and types:

Definition 5. Let t be an ill-typed DIHOL (or DIHOLP) term with well-typed image $\bar{t}$ in IHOL. In this case we will say that $\bar{t}$ is a *spurious* term. A term $\bar{s}$ in IHOL that is the image of a well-typed term s, will be called *proper*. A term *tm* in IHOL that is not the image of any (well-typed or not) term is said to be *improper*. This is visualized in Figure 3 below.

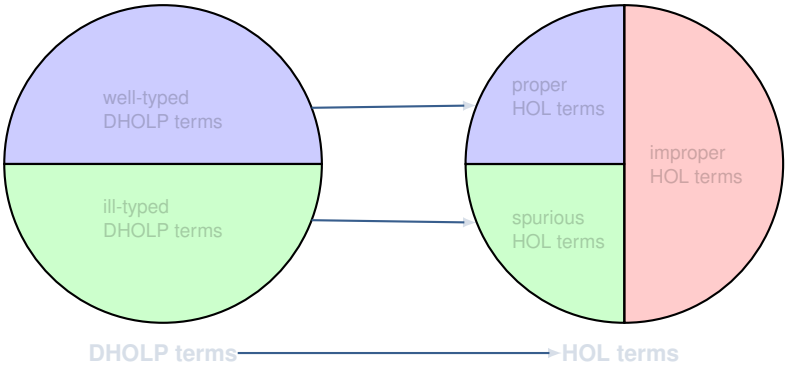

Figure 3: Diagram about spurious and improper terms in HOL

Similarly, given an ill-formed DIHOL type A with a well-formed image $\overline{A}$ in IHOL, we will say that $\overline{A}$ is a spurious type. A well-formed IHOL type that is not spurious will be called proper.

Observe, that proper terms in IHOL are the image of well-typed DIHOL (or DIHOLP) terms. Well-formed DIHOL (or DIHOLP) terms have – by definition – well-formed types, the soundness of the translation w.r.t. typing will therefore imply that proper terms have proper types.

For example, relative to the translation of above theory T the term $\lambda x : a.\ x$ is proper, the term $\lambda x : a.\ x =_{a\to a} \lambda x : a.\ x$ is improper and the translated conjecture is spurious.

If the translation is not (term-wise) injective, we can create a counterexample to the correctness of the translation by choosing terms s, t of types A and A' respectively s.t. $\bar{s} = \bar{t}$ and $\overline{A} \equiv \overline{A'}$ but not $A \equiv A'$ and considering the conjecture that the equality $s =_A t$ is valid (we have already seen an example for this situation above). This conjecture is not be provable in DIHOL, but its translation is spurious but well-typed and provable.

To avoid such issues coming from spurious formulas, it suffices to restrict to well-formed (and thus well-typed) DIHOL theories and conjectures. Then any equality will necessarily have terms of equal DIHOL type on the left and the right side. Now we only get injectivity issues if there are different DIHOL terms of same DIHOL type with same image in IHOL.

Considering either only identical terms or also just provably equal terms as the same we get different injectivity notions. Ultimately, we need that not provably equal terms of same DIHOL type have images that are not provably equal, a property that we call *value-wise injectivity*. If the translation is *value-wise injective* then we know that if equalities are provable in the image they are also provable in DIHOL. In fact, lemma 2 implies that *value-wise injectivity* is equivalent to the completeness of the translation w.r.t. validity.

Since it is very difficult to create equality proofs in DIHOL out of equality proofs of the images in IHOL, it will be useful to also consider other injectivity notions that are easier to prove to be satisfied. The other obvious injectivity notion is *term-wise injectivity*, indicating whether the translation maps non-identical DIHOL terms of equal type to non-identical terms. Term-wise injectivity can be easily shown using induction on the term productions in DIHOL:

Lemma 6. *Let Δ be a DIHOL context and let Γ denote its translation. Given two DIHOL terms s, t of type A and assuming s and t are not identical, it follows that $\overline{s}$ and $\overline{t}$ are not identical.*

This is proven in appendix D. We can show term-wise injectivity of Translation 2 in exactly the same way.

Remark 7. Later in the thesis we extend both translations to source logics which additionally feature predicate subtypes. However, since no additional terms are added, this lemma will hold for those translations as well.

3.3 Soundness and completeness of translation from DIHOL into IHOL

3.3.1 Soundness of the translation

Definition 6 (Soundness and completeness of a translation between logics)**.** We will say that a translation from a logic L to a logic L' is *sound*, if the following property holds: Given a conjecture $\Gamma \vdash_T J$ relative to a theory T in L, if the conjecture $\Gamma \vdash_T J$ is provable in L, then its translation is also provable in L'. For the logics in this thesis, we will additionally require that the invariants given in Table 1 hold.

We will say that a translation from logic L to logic L' is *complete*, if the following property holds: Take a **well-formed** conjecture $\Gamma \vdash_T J$ relative to a **well-formed** theory T in L. If the translation of the conjecture (and the corresponding invariants) are provable in L', then the conjecture is also provable in L.

The motivation for requiring the well-formedness (and hence also well-typedness) of the theory and the conjecture was already discussed in Section 3.2.3: If we don't assume well-typedness of the theory and conjecture, the translation will definitely not be complete.

The words soundness and completeness are sometimes used with reversed meanings (soundness

meaning what we call completeness and vice versa). We will use the concepts soundness/completeness from the perspective of the translation: Soundness carries provable DHOL statements to provable HOL statements, and completeness the other way. That corresponds to thinking of our translation as assigning semantics to DHOL by interpreting it in HOL. Consequently, from the perspective of theorem proving, completeness is the critical property: it ensures that proving $\overline{F}$ in HOL implies the validity of F in DHOL.

As usual soundness is the relatively easier property to establish, in our case it can be shown directly via induction on the derivations in D(I)HOL.

The more difficult completeness property will be discussed in the later Section 3.3.3.

Theorem 8 (Soundness)**.** *We have*

$$\vdash T\ \mathsf{Thy} \quad \textit{implies} \quad \vdash \overline{T}\ \mathsf{Thy} \tag{1}$$

$$\vdash_T \Gamma\ \mathsf{Ctx} \quad \textit{implies} \quad \vdash_{\overline{T}} \overline{\Gamma}\ \mathsf{Ctx} \tag{2}$$

$$\Gamma \vdash_T A : \mathsf{tp} \quad \textit{implies} \quad \overline{\Gamma} \vdash_{\overline{T}} \overline{A} : \mathsf{tp} \quad \textit{and}\ \overline{\Gamma} \vdash_{\overline{T}} A^? : \overline{A} \to \mathsf{bool} \tag{3}$$

$$\Gamma \vdash_T A \equiv B \quad \textit{implies} \quad \overline{\Gamma} \vdash_{\overline{T}} \overline{A} \equiv \overline{B} \quad \textit{and}\ \overline{\Gamma} \vdash_{\overline{T}} A^? =_{\overline{A} \to \mathsf{bool}} B^? \tag{4}$$

$$\Gamma \vdash_T t : A \quad \textit{implies} \quad \overline{\Gamma} \vdash_{\overline{T}} \bar{t} : \overline{A} \quad \textit{and}\ \Gamma \vdash_{\overline{T}} A^?\, \bar{t} \tag{5}$$

$$\Gamma \vdash_T F \quad \textit{implies} \quad \overline{\Gamma} \vdash_{\overline{T}} \overline{F} \tag{6}$$

In the special case of term equality $=_A$ we strengthen the claim to:

$$\Gamma \vdash_T t =_A t' \quad \textit{implies} \quad \overline{\Gamma} \vdash_{\overline{T}} \bar{t} =_{\overline{A}} \overline{t'} \wedge A^?\, \bar{t} \wedge A^?\, \overline{t'} \tag{7}$$

Additionally the substitution lemma holds, i.e.,

$$\Gamma, x : A \vdash t : B \quad \textit{and}\ \Gamma \vdash u : A \quad \textit{implies} \quad \overline{\Gamma} \vdash \overline{t[x/u]} =_{\overline{B}} \bar{t}[x/\bar{u}] \tag{8}$$

$$\Gamma, x : A \vdash B\ \mathsf{tp} \quad \textit{and}\ \Gamma \vdash u : A \quad \textit{implies} \quad \overline{\Gamma} \vdash \overline{B[x/u]} \equiv \overline{B}[x/\bar{u}] \tag{9}$$

Corollary 9. *Analogously (with just a case (treated in Appendix G) for rule (boolExt) instead of the case for rule (propExt)), it follows that Translation 1 from DHOL into HOL is sound.*

Theorem 10. *Similarly Translation 2 from DHOL* into HOL* is sound.*

Theorem 8 and Theorem 10 are special cases of the soundness theorems of Translation 1 from DIHOLP into IHOL (resp. Translation 2 from DHOLP* into HOL) proven in Theorem 14 (resp. Corollary 16).

3.3.2 The problem of spurious terms

Knowing the term-wise injectivity of the translation is helpful for show the completeness of the translation. Using it we can try to inductively "lift" proofs from IHOL to DIHOL.

However, there is a critical issue we still have to address for this approach, namely the translation being non-surjective. Specifically, in IHOL we have

- spurious terms
- unrelativized equalities
- typing predicates and their applications outside of relativations
- unrelativized context variables

The issue with the non-surjectivity is that proofs in IHOL can contain terms not in the image (of well-typed terms) of the translation. Since such terms don't have any corresponding (well-typed) term in DIHOL, there is no directly corresponding proof in DIHOL. In that sense, there are "more proofs" in IHOL than in DIHOL (although of course there are countably infinitely many proofs in both logics).

It will be useful to distinguish between two different kinds of improper terms.

Definition 7. An improper term is called *almost proper* iff its relativization isn't spurious and contains no spurious subterms, otherwise it is said to be *unnormalizably spurious*. We will consider proper terms to be *almost proper* as well.

As will be described in section 4.3.3 we can transform derivations in IHOL in a way that removes unnormalizably spurious terms from the derivation, yielding a derivation in which all occurring terms are almost proper.

Regarding typing predicates outside of relativizations of equalities: We will prove that the typing predicates are equivalent to the translations of the corresponding replacement predicates, so we can consider the replacement predicates as the quasi-preimage of the typing predicate.

Regarding unrelativized equalities: After the removing unnormalizably spurious terms (and replacing typing predicates by translations of replacement predicates outside of relativizations) in a derivation it is clear that unrelativized equalities have relativizations that are proper terms. For this reason unrelativized equalities don't really affect the completeness proof. We simply consider the preimage of a relativization as the quasi-preimage of the equality.

Regarding unrelativized context variables: As it turns out, they don't really affect the completeness proof. We can simply add the missing typing assumptions for them when constructing a DIHOL proof from an IHOL proof.

3.3.3 Completeness of the translation

Theorem 11 (Completeness of the Translation 1)**.** *Assume a well-formed DIHOL theory T and conjecture $\Gamma \vdash_T F$. Then it follows:*

$$\overline{\Gamma} \vdash_{\overline{T}} \overline{F} \quad \textit{implies} \quad \Gamma \vdash_T F \tag{10}$$

In particular, this includes the special case of term equality $=_{\overline{A}}$:

$$\overline{\Gamma} \vdash \mathsf{Relat}_A[\bar{t} =_{\overline{A}} \overline{t'}] \quad \textit{implies} \quad \Gamma \vdash t =_A t' \tag{11}$$

This is a special case of the completeness proof for Translation 1 from DIHOLP to IHOL given in Theorem 25.

Similarly, also Translation 2 from DHOL* into HOL* and HOL is complete (this is a special case of Corollary 28 resp. Corollary 30).

3.4 Typechecking DHOL* theories

Since we cannot directly use the translation for typechecking DHOL* theories, we need a different method for typechecking. This can be done by using the usual typing rules of DHOL and generating proof obligations for term equalities for the cases of the rules (cong:) and (cong-Base'). This is described in more detail in subsection 4.4.

4 Dependently typed higher-order logic with predicate subtypes

We can extend HOL, DHOL*, IHOL,DHOL or DIHOL with predicate subtypes in a similar way to how we added dependent function types. We will abbreviate DHOL*, DIHOLand DHOL with predicate subtypes as DHOLP*, DIHOLP and DHOLP respectively. The type $A|_p$ denotes the subtype of A defined by the predicate $p : A \to \mathsf{bool}$. HOL with predicate subtypes shares some common properties with DHOL: in both cases, terms can occur in types, type equality and typing becomes undecidable, well-typedness of a declarations may depend on assumptions and axioms, and dependent implication and conjunction are needed. Yet neither dependent function types nor predicate subtypes can be defined in terms of the other.

To extend D(I)HOL with predicate subtypes, we extend its grammar to the following grammar:

T	$::=$	$* \mid T, Dec$	theories
Dec	$::=$	$c : A \mid a : (\Pi x : A.)^* \ \mathsf{tp} \mid F$	declarations
Γ	$::=$	$. \mid \Gamma, x : A \mid \Gamma, F$	contexts
A, B	$::=$	$a\, t_1 \dots t_n \mid \Pi x : A.\ B \mid \mathsf{bool} \mid A\vert_p$	types
s, t, f, F, G	$::=$	$c \mid x \mid f\ t \mid \lambda x : A.\ t \mid s =_A t \mid F \Rightarrow G$	terms

By disallowing variables in contexts, Π-types or λ-functions of (subtypes of) bool-valued function types (of positive arity) and (subtypes of) $\mathsf{bool}|_p$-valued function types for predicates $p : \Pi x : \mathsf{bool}.\ \mathsf{bool}$, we yield the allowed DHOLP* syntax.

No new term constructors are needed.

The assumption nonEmpty' and the inference rules of D(I)HOL (DHOL*) remain, we only add the additional inference rules (for any of the three logics) for predicate subtypes:

$$\frac{\Gamma \vdash_T p : \Pi x : A.\ \mathsf{bool}}{\Gamma \vdash_T A|_p\ \mathsf{tp}}\,|_p\mathsf{tp} \qquad \frac{\Gamma \vdash_T A \equiv A' \quad \Gamma \vdash_T p =_{\Pi x:A.\ \mathsf{bool}} p'}{\Gamma \vdash_T A|_p \equiv A'|_{p'}}\,|_p \equiv$$

$$\frac{\Gamma \vdash_T A \equiv A' \quad \Gamma \vdash_T p =_{\Pi x:A.\ \mathsf{bool}} \lambda x : A.\ \mathsf{true}}{\Gamma \vdash_T A|_p \equiv A'}\,|_p \mathrm{trivL} \qquad \frac{\Gamma \vdash_T A \equiv A' \quad \Gamma \vdash_T p =_{\Pi x:A.\ \mathsf{bool}} \lambda x : A.\ \mathsf{true}}{\Gamma \vdash_T A' \equiv A|_p}\,|_p \mathrm{trivR}$$

$$\frac{\Gamma \vdash_T t : A \quad \Gamma \vdash_T p\ t}{\Gamma \vdash_T t : A|_p}|_p\mathrm{I} \qquad \frac{\Gamma \vdash_T t : A|_p}{\Gamma \vdash_T t : A}|_p\mathrm{E}: \qquad \frac{\Gamma \vdash_T t : A|_p}{\Gamma \vdash_T p\ t}|_p\mathrm{E}p$$

Remark 12. Observe that many of the rules derived for HOL in Lemma 1 still hold in DHOLP and its fragments. In particular, the rules (ctxThy), (tpCtx), (typingTp) and (validTyping) can be proven by the same method. Also the rules (monotonic$\vdash$), (var$\vdash$), (rewrite) and the introduction and elimination rules for the quantifier and defined logical connectives can be derived in D(I)HOLP with the same proofs.

We extend the definition of the replacement predicates to DHOLP and its fragments by defining

$$p_{A|_p}\ t := p\ t \wedge p_A\ t \quad . \tag{R7}$$

Example 5 (Set theory). This theory formalizes set theory, specifically sets, elementhood, subsets, functions between sets, function application and restriction of functions to subsets.

For better readability we use the infix notations $\in, \subset, @, \cdot|_\cdot$ for `of`, `subs`, `@` and `funcRestr`. Furthermore, we allow for defined type declarations and term constructors. We treat the defined declarations as mere abbreviations that are expanded before the translation is applied.

$$\begin{aligned}
\mathsf{set} &: \mathsf{tp} \\
\texttt{of} &: \Pi a : \mathsf{set}.\ \Pi x : \mathsf{set}.\ \mathsf{bool} \\
\texttt{subs} &: \Pi a : \mathsf{set}.\ \Pi s : \mathsf{set}.\ \mathsf{bool} \\
&\quad \forall a : \mathsf{set}.\ \forall s : \mathsf{set}.\ s \subset a =_{\mathsf{bool}} (\forall y : \mathsf{set}.\ y \in s \Rightarrow y \in a) \\
\mathsf{Class} &= \lambda p : (\Pi y : \mathsf{set}.\ \mathsf{bool}).\ \mathsf{set}|_{\lambda s : \mathsf{set}.\ p\ s} \\
\mathsf{Elem} &= \lambda X : \mathsf{set}.\ \mathsf{Class}\ (\lambda x : \mathsf{set}.\ x \in X) \\
\mathsf{Subset} &= \lambda X : \mathsf{set}.\ \mathsf{Class}\ (\lambda x : \mathsf{set}.\ x \subset X) \\
&: \Pi a : \mathsf{set}.\ \Pi b : \mathsf{set}.\ \mathsf{set} \\
@ &: \Pi a : \mathsf{set}.\ \Pi b : \mathsf{set}.\ \Pi f : \mathsf{Elem}\ (\texttt{func}\ a\ b).\ \Pi x : \mathsf{Elem}\ (a).\ \mathsf{Elem}\ (b) \\
&\quad \forall a : \mathsf{set}.\ \forall b : \mathsf{set}.\ \forall f : \mathsf{Elem}\ (\texttt{func}\ a\ b).\ \forall g : \mathsf{Elem}\ (\texttt{func}\ a\ b). \\
&\qquad \forall x : \mathsf{Elem}\ (a).\ f @ x =_{\mathsf{Elem}\ (b)} g @ x \Rightarrow f =_{\mathsf{Elem}\ (\texttt{func}\ a\ b)} g \\
\texttt{funcRestr} &: \Pi a : \mathsf{set}.\ \Pi b : \mathsf{set}.\ \Pi f : \mathsf{Elem}\ (\texttt{func}\ a\ b).\ \Pi s : \mathsf{Subset}\ (a).\ \texttt{func}\ s\ b \\
&\quad \forall a : \mathsf{set}.\ \forall b : \mathsf{set}.\ \forall f : \mathsf{Elem}\ (\texttt{func}\ a\ b).\ \forall s : \mathsf{Subset}\ (a). \\
&\qquad \forall x : \mathsf{Elem}\ (s).\ f\ @\ x =_{\mathsf{Elem}\ (y)} \left(f|_s\right)\ @\ x
\end{aligned}$$

In this theory we can formalize the conjecture:

$$\begin{aligned}
&\forall a : \mathsf{set}.\ \forall d : \mathsf{set}.\ \forall b : \mathsf{Subset}\ (a).\ \forall s : \mathsf{Subset}\ (b). \\
&\qquad \forall f : \mathsf{Elem}\ (\texttt{func}\ a\ d).\ \left(f|_b\right)\Big|_s =_{\mathsf{Elem}\ (\texttt{func}\ s\ d)} f|_s
\end{aligned}$$

Since both theory and conjecture use subtyping, this cannot be expressed (this way) in HOL.

4.1 Translating DIHOLP into IHOL and DHOLP* into HOL*

Definition 8 (Translation)**.** We define two translations from DHOLP (DHOLP*) into HOL (HOL*) syntax by induction on the Grammar. The translations use either of Translation 1 or Translation 2 (denoted by $\bar{\cdot}$ in the definiens) internally (which is why we obtain two translations here as well). Using the notation $\bar{\cdot}$ for either Translation 1 or Translation 2 we can extend this translation to DHOLP resp. DHOLP* by induction on the grammar:

The cases for theories and contexts are:

$$\overline{\mathsf{o}} := \overline{\mathsf{o}} \tag{PT1}$$

$$\overline{T,\ a : \Pi x_1 : A_1.\ \ldots \Pi x_n : A_n.\ \mathsf{tp}} := \overline{T},\ a : \mathsf{tp},\ a^? : \overline{A_1} \to \ldots \to \overline{A_n} \to a \to \mathsf{bool} \tag{PT2}$$

$$\overline{T,\ c : A} := \overline{T},\ c : \overline{A},\ A^?\ c \tag{PT3}$$

$$\overline{T,\ F} := \overline{T},\ \overline{F} \tag{PT4}$$

$$\bar{\cdot} := . \tag{PT5}$$

$$\overline{\Gamma,\ x : C} := \overline{\Gamma},\ x : \overline{A},\ C^?\ x \tag{PT6}$$

$$\overline{\Gamma,\ F} := \overline{\Gamma},\ \overline{F} \tag{PT7}$$

The case of $\overline{A}$ and $A^?$ for types A are:

$$\overline{A|_p} := \overline{A} \tag{PT8}$$

$$\left(A|_p\right)^? := \lambda t : \overline{A}.\ p\ t \wedge A^?\ t \tag{PT9}$$

$$\overline{\Pi x : A.\ B} := \Pi x : \overline{A}.\ \overline{B} \tag{PT10}$$

$$(\Pi x : A.\ B)^? := \lambda f : \overline{\Pi x : A.\ B}.\ \forall x : \overline{A}.\ A^?\ x \Rightarrow B^?\ \left(f\ x\right) \tag{PT11}$$

$$\overline{A} := \overline{A} \qquad A \text{ is DHOL type} \tag{PT12}$$

The cases for terms are:

$$\overline{c} := c \tag{PT13}$$

$$\overline{x} := x \tag{PT14}$$

$$\overline{\lambda x : A.\ t} := \lambda x : \overline{A}.\ \overline{t} \tag{PT15}$$

$$\overline{f\ t} := \overline{f}\ \overline{t} \tag{PT16}$$

$$\overline{F \Rightarrow G} := \overline{F} \Rightarrow \overline{G} \tag{PT17}$$

$$\overline{s =_{A|_p} t} := \overline{p}\ \overline{s} \wedge \overline{p}\ \overline{t} \wedge \overline{s =_A t} \tag{PT18}$$

$$\overline{f =_{\Pi y : A.\ B} g} := \forall x : \overline{A}.\ A^?\ x \Rightarrow \mathsf{Relat}_B[\overline{f}\ x =_{\overline{B}} \overline{g}\ x] \tag{PT19}$$

$$\overline{s =_A t} := s =_A t \wedge A^?\ s \wedge A^? t \qquad \text{else} \tag{PT20}$$

Here the notation $\mathsf{Relat}_T[x =_{\overline{T}} x']$ is defined inductively (on T) by:

$$\mathsf{Relat}_A[s =_{\overline{A}} t] := \mathsf{R}[s] =_{\overline{A}} \mathsf{R}[t] \wedge A^? \, \mathsf{R}[s] \wedge A^? \, \mathsf{R}[t] \quad \text{if } A \text{ not function type} \tag{PT21}$$

$$\mathsf{Relat}_{A|_p}[\overline{s} =_{\overline{A|_p}} \overline{t}] := p \, \mathsf{R}[s] \wedge p \, \mathsf{R}[t] \wedge \mathsf{Relat}_A[\overline{s} =_{\overline{A}} \overline{t}] \tag{PT22}$$

$$\mathsf{Relat}_{\Pi x:A.\ B}[f =_{\overline{\Pi x:A.\ B}} f'] := \forall x : \overline{A}.\ A^? \, x \Rightarrow \mathsf{Relat}_B[f \ x =_{\overline{B}} f \ x] \quad \text{else} \tag{PT23}$$

And the notation $\mathsf{R}[tm]$ is defined inductively on tm by:

$$\mathsf{R}[s] := s \quad \text{for } s = \mathsf{R}[t] \tag{PT24}$$

$$\mathsf{R}[s =_{\overline{A}} t] := \mathsf{Relat}_A[s =_{\overline{A}} t] \tag{PT25}$$

$$\mathsf{R}[f \ t] := \mathsf{R}[f] \, \mathsf{R}[t] \tag{PT26}$$

$$\mathsf{R}[F \Rightarrow G] := \mathsf{R}[F] \Rightarrow \mathsf{R}[G] \tag{PT27}$$

$$\mathsf{R}[\lambda x : \overline{A}.\ s] := \lambda x : \overline{A}.\ \mathsf{R}[s] \tag{PT28}$$

$$\mathsf{R}[\lambda x : \overline{A}.\ s] := \lambda x : \overline{A}.\ \mathsf{R}[s] \tag{PT29}$$

$$\mathsf{R}[x] := x \tag{PT30}$$

$$\mathsf{R}[c] := c \tag{PT31}$$

Translating example 5 to HOL yields:

Example 6 (Translation of set theory example).

$$\begin{aligned}
\mathsf{bool}^? &: \mathsf{bool} \to \mathsf{bool} \\
&\quad \mathsf{bool}^? \ \mathsf{true} \\
&\quad \mathsf{bool}^? \ \mathsf{false} \\
\mathsf{set} &: \mathsf{tp} \\
\mathsf{set}^? &: \mathsf{set} \to \mathsf{bool} \\
\mathtt{of} &: \mathsf{set} \to \mathsf{set} \to \mathsf{bool} \\
&\quad \forall a : \overline{\mathsf{set}}.\ \mathsf{set}^? \ a \Rightarrow \forall x : \overline{\mathsf{set}}.\ \mathsf{set}^? \ x \Rightarrow \mathsf{bool}^? \ (x \in a) \\
\mathtt{subs} &: \mathsf{set} \to \mathsf{set} \to \mathsf{bool} \\
&\quad \forall a : \overline{\mathsf{set}}.\ \mathsf{set}^? \ a \Rightarrow \forall s : \overline{\mathsf{set}}.\ \mathsf{set}^? \ s \Rightarrow \mathsf{bool}^? \ (s \subset a) \\
&\quad \forall a : \overline{\mathsf{set}}.\ \mathsf{set}^? \ a \Rightarrow \forall s : \overline{\mathsf{set}}.\ \mathsf{set}^? \ s \Rightarrow \\
&\qquad (s \subset a) =_{\mathsf{bool}} \left(\forall y : \overline{\mathsf{set}}.\ \mathsf{set}^? \ y \Rightarrow y \in s \Rightarrow y \in a\right) \\
&\qquad\quad /\backslash \ \mathsf{bool}^? \ (s \subset a) \ /\backslash \ \mathsf{bool}^? \ s \wedge \mathsf{bool}^? \ a \\
\mathtt{func} &: \mathsf{set} \to \mathsf{set} \to \mathsf{set} \\
&\quad \forall a : \overline{\mathsf{set}}.\ \mathsf{set}^? \ a \Rightarrow \forall b : \overline{\mathsf{set}}.\ \mathsf{set}^? \ b \Rightarrow \mathsf{set}^? \ (\mathtt{func} \ a \ b) \\
@ &: \mathsf{set} \to \mathsf{set} \to \mathsf{set} \to \mathsf{set} \to \mathsf{set}
\end{aligned}$$

$$\forall a : \overline{\mathsf{set}}.\ \mathsf{set}^? a \Rightarrow \forall b : \overline{\mathsf{set}}.\ \mathsf{set}^? b \Rightarrow \forall f : \mathsf{set}.\ f \in \mathtt{func}\ a\ b \wedge \mathsf{set}^? f \Rightarrow$$
$$\forall x : \mathsf{set}.\ x \in a \wedge \mathsf{set}^? x \Rightarrow f@x \in b \wedge \mathsf{set}^? f@x$$
$$\forall a : \overline{\mathsf{set}}.\ \mathsf{set}^? a \Rightarrow \forall b : \overline{\mathsf{set}}.\ \mathsf{set}^? b \Rightarrow \forall f : \mathsf{set}.\ f \in \mathtt{func}\ a\ b \wedge \mathsf{set}^? f \Rightarrow$$
$$\forall g : \mathsf{set}.\ g \in \mathtt{func}\ a\ b \wedge \mathsf{set}^? g \Rightarrow \forall x : \mathsf{set}.\ x \in a \wedge \mathsf{set}^? x \Rightarrow$$
$$(f@x) =_{\mathsf{set}} (g@x) \wedge (f@x) \in b \wedge \mathsf{set}^? (f@x) \wedge (g@x) \in b \wedge \mathsf{set}^? (f@x) \Rightarrow$$
$$f =_{\mathsf{set}} g \wedge f \in \mathtt{func}\ a\ b \wedge \mathsf{set}^? f \wedge g \in \mathtt{func}\ a\ b \wedge \mathsf{set}^? f$$

$\mathtt{funcRestr} : \mathsf{set} \to \mathsf{set} \to \mathsf{set} \to \mathsf{set} \to \mathsf{set}$

$$\forall a : \overline{\mathsf{set}}.\ \mathsf{set}^? a \Rightarrow \forall b : \overline{\mathsf{set}}.\ \mathsf{set}^? b \Rightarrow \forall f : \mathsf{set}.\ f \in \mathtt{func}\ a\ b \wedge \mathsf{set}^? f \Rightarrow$$
$$\forall s : \mathsf{set}.\ s \subset a \wedge \mathsf{set}^? s \Rightarrow \mathsf{set}^? \left(f|_s\right)$$
$$\forall a : \overline{\mathsf{set}}.\ \mathsf{set}^? a \Rightarrow \forall b : \overline{\mathsf{set}}.\ \mathsf{set}^? b \Rightarrow \forall f : \mathsf{set}.\ f \in \mathtt{func}\ a\ b \wedge \mathsf{set}^? f \Rightarrow$$
$$\forall s : \mathsf{set}.\ s \subset a \wedge \mathsf{set}^? s \Rightarrow \forall x : \mathsf{set}.\ x \in s \wedge \mathsf{set}^? x \Rightarrow$$
$$f@x =_{\mathsf{set}} \left(f|_s\right)@x \wedge f@x \in b \wedge \mathsf{set}^? f@x \wedge \left(f|_s\right)@x \in b \wedge \mathsf{set}^? f@x$$

The conjecture is translated to:

$$\forall a : \overline{\mathsf{set}}.\ \mathsf{set}^? a \Rightarrow \forall d : \overline{\mathsf{set}}.\ \mathsf{set}^? d \Rightarrow \forall b : \mathsf{set}.\ b \subset a \wedge \mathsf{set}^? b \Rightarrow$$
$$\forall s : \mathsf{set}.\ s \subset b \wedge \mathsf{set}^? s \Rightarrow \forall f : \mathsf{set}.\ f \in \mathtt{func}\ a\ d \wedge \mathsf{set}^? f \Rightarrow$$
$$\left(f|_b\right)\Big|_s =_{\mathsf{set}} f|_s \wedge \mathsf{set}^? f|_s$$
$$\wedge \left(f|_b\right)\Big|_s \in (\mathtt{func}\ s\ b) \wedge \mathsf{set}^? \left(f|_b\right)\Big|_s \wedge f|_s \in (\mathtt{func}\ s\ b) \wedge \mathsf{set}^? f|_s$$

4.2 Soundness of the translations

We will start by proving the soundness of the Translation 1 from DIHOLP into IHOL, then we will explain how the soundness proof for Translation 1 from DHOLP into HOL and for Translation 2 from DHOLP* into HOL will differ.

To distinguish statements in IHOL and in DIHOLP we will in the following denote the turnstyle symbols $\vdash$ and $\vdash_{\overline{T}}$ in IHOL by $\vdash^H$ and $\vdash^H_{\overline{T}}$

As usual soundness is the relatively easier property to establish, in our case it can be shown directly via induction on the derivations in DIHOLP.

The more difficult completeness property will be discussed in the subsequent Subsection 4.3.

The following lemma will be useful for the soundness proof:

Lemma 13. *Given DIHOLP terms s, t of type C which is well-formed in context Γ, we have $\overline{\Gamma} \vdash_{\overline{T}} \overline{s} =_{\overline{C}} \overline{t} \wedge C^? \overline{s} \wedge C^? \overline{t}$ implies $\overline{\Gamma} \vdash_{\overline{T}} \overline{s =_C t}$.*

This is proven in Appendix E.

We can now directly prove the soundness of Translation 1 from DIHOLP into IHOL.

Theorem 14 (Soundness)**.** *We have*

$$\vdash T\ \mathsf{Thy} \quad \textit{implies} \quad \vdash^H \overline{T}\ \mathsf{Thy} \tag{1}$$

$$\vdash_T \Gamma\ \mathsf{Ctx} \quad \textit{implies} \quad \vdash^H_{\overline{T}} \overline{\Gamma}\ \mathsf{Ctx} \tag{2}$$

$$\Gamma \vdash_T A : \mathsf{tp} \quad \textit{implies} \quad \overline{\Gamma} \vdash^H_{\overline{T}} \overline{A} : \mathsf{tp} \quad \textit{and}\ \overline{\Gamma} \vdash^H_{\overline{T}} A^? : \overline{A} \to \mathsf{bool} \tag{3}$$

$$\Gamma \vdash_T A \equiv B \quad \textit{implies} \quad \overline{\Gamma} \vdash^H_{\overline{T}} \overline{A} \equiv \overline{B} \quad \textit{and}\ \overline{\Gamma} \vdash^H_{\overline{T}} A^? =_{\overline{A}\to\mathsf{bool}} B^? \tag{4}$$

$$\Gamma \vdash_T t : A \quad \textit{implies} \quad \overline{\Gamma} \vdash^H_{\overline{T}} \bar{t} : \overline{A} \quad \textit{and}\ \Gamma \vdash^H_{\overline{T}} A^?\ \bar{t} \tag{5}$$

If the theory does not contain constants of types of arity at least 3, we additionally have:

$$\Gamma \vdash_T F \quad \textit{implies} \quad \overline{\Gamma} \vdash^H_{\overline{T}} \overline{F} \tag{6}$$

In the special case of term equality $=_A$ we strengthen this claim to:

$$\Gamma \vdash_T t =_A t' \quad \textit{implies} \quad \overline{\Gamma} \vdash^H_{\overline{T}} \overline{t =_A t'} \quad \textit{and}\ \overline{\Gamma} \vdash^H_{\overline{T}} A^?\ \bar{t}\ \textit{and}\ \overline{\Gamma} \vdash^H_{\overline{T}} A^?\ \overline{t'} \tag{7}$$

Additionally the substitution lemma holds, i.e.,

$$\Gamma,\ x : A \vdash t : B \quad \textit{and}\ \Gamma \vdash u : A \quad \textit{implies} \quad \overline{\Gamma} \vdash^H_{\overline{T}} \overline{t[x/u]} =_{\overline{B}} \bar{t}[x/\bar{u}] \tag{8}$$

$$\Gamma,\ x : A \vdash B\ \mathsf{tp} \quad \textit{and}\ \Gamma \vdash u : B \quad \textit{implies} \quad \overline{\Gamma} \vdash^H_{\overline{T}} \overline{B[x/u]} \equiv \overline{B}[x/\bar{u}] \tag{9}$$

The proof can be found in Section F in the appendix.

Corollary 15. *Translation 1 from DHOLP into HOL is sound.*

This is proven in Appendix G.

The soundness of the translation from DHOLP* into HOL* follows analogously:

Corollary 16. *Translation 2 from DHOLP* into HOL is sound.*

Proof. The proof is analogous to the soundness proof for the Translation 1 from DHOLP into HOL. The only difference is in having to prove $p_{bool}\ t$ instead of $\mathsf{bool}^?\ t$ in some easy cases. □

4.3 Completeness of the translations

Again we will first discuss Translation 1 from DIHOLP into IHOL then from DHOLP into HOL and finally discuss how the completeness proof for Translation 2 differs.

The idea for showing completeness of Translation 1 from DIHOLP to IHOL is to explicitly describe how to lift IHOL derivations to DIHOLP derivations. We construct a lift of a given IHOL derivation of a translated validity conjecture to a DIHOLP derivation in several steps:

1. show that typing predicates are equivalent to the translations of replacement predicates and a replacement predicate p_A holding on a term t implies the typing statement $t : A$ (this is not actually necessary for our proof, but a good sanity check),
2. define the quasi-preimages of improper IHOL terms as follows:
   - replacing typing predicates outside of relativizations with the translation of the corresponding replacement predicate,
   - to obtain quasi-preimage for unrelativized equalities in the derivation, consider their relativizations instead,
   - take the preimage of the resulting proper term,
3. describe proof transformation removing unnormalizably spurious terms from derivations,
4. construct a lifting of the proof to DIHOLP by induction on the validity inference rules in IHOL: assuming the quasi-preimages of the validity assumptions of a validity rule hold in DIHOLP, show that the quasi-preimage of the conclusion holds in DIHOLP.

This shows that Translation 1 from DIHOLP into IHOL is complete w.r.t. validity. We do not know whether Translation 1 from DIHOLP into IHOL is complete w.r.t. the typing and type-equality judgments (but we know that Translation 1 from DHOLP into HOL is not).

Finally, we discuss how we can analogously proof the completeness w.r.t. validity of Translation 1 from DHOLP into HOL and Translation 2 from DHOLP* to into HOL.

4.3.1 Replacing typing predicates with translations of replacement predicates

In this section, we will show that replacing the typing predicates with the corresponding replacement predicates will not affect the derivability of formulae in IHOL. We will also show that if a replacement predicate p_A holds on a term $t : A$ in DIHOLP, then t has type A. Therefore, it makes sense to use the replacement predicates to find quasi-preimages for the typing predicates.

Definition 9. Given a valid DIHOLP context Γ and well-typed IHOL formula φ over Γ, we denote by $\mathsf{RP}[\varphi]$ the formula obtained by replacing typing predicates $A^?$ in φ outside of relativizations by the corresponding replacement predicates p_A. More formally we define:

$$\mathsf{RP}[\mathsf{R}[s]] := \mathsf{R}[\mathsf{RP}[s]] \tag{PT32}$$
$$\mathsf{RP}\left[A^?\right] := p_A \tag{PT33}$$
$$\mathsf{RP}[c] := c \tag{PT34}$$
$$\mathsf{RP}[x] := x \tag{PT35}$$
$$\mathsf{RP}[f\ t] := \mathsf{RP}[f]\ \mathsf{RP}[t] \tag{PT36}$$
$$\mathsf{RP}[\lambda x : C.\ t] := \lambda x : C.\ \mathsf{RP}[t] \tag{PT37}$$
$$\mathsf{RP}\left[s =_A t\right] := \mathsf{RP}[s] =_A \mathsf{RP}[t] \tag{PT38}$$
$$\mathsf{RP}[s \Rightarrow t] := \mathsf{RP}[s] \Rightarrow \mathsf{RP}[t] \tag{PT39}$$

Remark 17. It is clear from (PT32) that applying $\mathsf{RP}[\cdot]$ to a relativized almost-proper term yields a relativized almost-proper term. Furthermore, it is clear from (PT33) that the result of applying $\mathsf{RP}[\cdot]$ to an almost-proper term yields a term with no typing predicates except for those required to relativize equalities. Using induction on the shape of a given almost proper term, we can conclude that for an almost proper term t the term $\mathsf{RP}[\mathsf{R}[t]]$ is always a proper term.

Regarding the equivalence of typing predicates with the translations of the corresponding replacement predicates:

Lemma 18. *Let T be a well-formed DIHOLP theory and φ a well-formed conjecture in the well-formed context Γ relative to T using the validity judgement.*

Given a proper IHOL term $t := \overline{tm}$ with tm of DIHOLP type A over $\overline{\Gamma}$ and a proper DIHOLP type B with $\overline{A} \equiv \overline{B}$, it follows that $\overline{\Gamma},\ B^?\ t \vdash_T \overline{p_B\ t}$.

This is proven in Appendix I.

With the same assumption on theories as in the validity case of the soundness theorem, we can also prove the converse to Lemma 18:

Lemma 19. *Let φ be a formula in DIHOLP context Γ and t an occurence of a subterm of φ. Denote the free and bound variables of type A in the scope of t by $v_1, \ldots, v_{q(\Gamma)}$ and $v_{q(\Gamma)+1}, \ldots, v_{r(t)}$. Then $\overline{\Gamma, v_{q(\Gamma)+1} : A, \ldots, v_{r(t)} : A} \vdash t : \overline{A}$ implies $\overline{\Gamma, v_{q(\Gamma)+1} : A, \ldots, v_{r(t)} : A},\ a^?\ t \vdash \overline{p_a\ t}$.*

This is proven in Appendix H.

By rule (propExt), the two lemmas jointly imply that applications of typing predicates are equal to the corresponding applications of translations of replacement predicates.

We want to check that if a replacement predicate $p_A\ t$ holds on a term in DIHOLP, it follows that the term is of the type of the replacement predicate.

Lemma 20. *Given a well-formed DIHOLP theory T and well-formed DIHOLP context Γ relative to T, let A be a well-formed type and t any well-typed term.*

If we have $\Gamma \vdash_T p_A\ t$, then $\Gamma \vdash_T t : A$.

This is proven in Appendix J.

Combining Lemma 18-18 yields that the typing predicates (or their replacements) correctly encode the typing information. If we can show that we can lift validity proofs for (possibly ill-typed) applications of typing predicates, completeness follows (although this is difficult to prove (and doesn't even hold when translating to HOL rather than IHOL)). This could be used to build a deep-embedding of DHOLP into HOL by disregarding types in HOL and only considering the typing predicates. But since we want a shallow-embedding, we instead use the replacement predicates as a tool to show the completeness of our shallow embeddings.

4.3.2 Constructing quasi-preimages of statements in IHOL derivations

Definition 10. Assume a well-formed DIHOLP theory $\vdash T\ \mathsf{Thy}$.

We say that an IHOL context Δ is *proper* (relative to $\overline{T}$), iff there is a well-formed IHOL context Θ (relative to $\overline{T}$), s.t. there exists a well-formed DIHOLP context Γ (relative to T) with $\overline{\Gamma} = \Theta$ and Θ can be obtained from Δ by adding well-typed typing assumptions. In this case, Γ is called a *quasi-preimage* of Δ. Inspecting the translation, it becomes clear that Γ is uniquely determined by the choices of the preimages of the types of variables without a typing assumption in Δ.

Given a proper IHOL context Δ and a well-typed IHOL formula φ over Δ, we say that φ is *quasi-proper* iff $\mathsf{RP}[\mathsf{R}[\varphi]] = \overline{F}$ with $\Gamma \vdash_T F : \mathsf{bool}$ and Γ is a quasi-preimage of Δ. In that case, we call F a *quasi-preimage* of φ.

Finally, we call a validity judgement $\Delta \vdash^H_{\overline{T}} \varphi$ in IHOL *proper* iff

1. Δ is proper,
2. φ is quasi-proper in context Δ

In this case, we will call $\overline{\Gamma} \vdash^H_{\overline{T}} \overline{F}$ a relativization of $\Delta \vdash^H_{\overline{T}} \varphi$ and $\Gamma \vdash_T F$ a *quasi-preimage* of the statement $\Delta \vdash^H_{\overline{T}} \varphi$, where Γ is a quasi-preimage of Δ and F a quasi-preimage of φ.

4.3.3 Removing spurious terms from IHOL proofs

Assuming a valid IHOL derivation, we can make all terms in the derivation almost proper using the following proof transformation:

Definition 11. A *statement transformation* in a given logic is a map that maps statements in the logic to statements in the logic.

Definition 12. A *macro-step M for* a statement transformation T *replacing* a step S in a derivation is a sequence of steps $S_1, \dots, S_n$ (called *micro-steps* of M) s.t. the assumptions of the S_i that are not concluded by S_j with $j < i$ are results of applying T to assumptions of step S and furthermore the conclusion of step S_n is the result of applying T to the conclusion of S. The assumptions of those S_j that are not concluded by previous micro-steps of M are called the *assumptions of macro-step M* and the conclusion of the last micro-step S_n of M is called the *conclusion of macro-step M*.

Definition 13. A *P-normalizing statement transformation* $\mathsf{sRed}(\cdot)$ is defined to be the transformation that replaces terms in statements (including their contexts) as described below. The definition of the transformation of a term will additionally depend on a DIHOLP-type A (called the *preimage type*) for each term t. We will write those types as indices to the IHOL terms, so for instance t_A would indicate an IHOL term t of type $\overline{A}$ and preimage type A.

These indices are used to effectively associate to each term a type of a possible quasi-preimage, which is useful as for λ-functions there are quasi-preimages of potentially many different types. We require that for an indexed term t_A, term t has type $\overline{A}$ and that for almost proper terms t_A with unique quasi-preimage the quasi-preimage has type A. To make this transformation more useful we also assume an arbitrary meta-level predicate P on terms which is satisfied by the terms in the final conclusion (including its context) of the derivation and which additionally satisfies:

$$P\ (s =_A t) \qquad =P\ s \wedge P\ t \tag{P1}$$

$$P\ (F \Rightarrow G) \quad =P\ F \wedge P\ G \tag{P2}$$
$$P\ (\lambda x : A.\ f\ x) \quad =P\ f \tag{P3}$$
$$P\ (\lambda x : A.\ s)\,t \quad \Longrightarrow P\ s[{}^{x}/_{t}] \tag{P4}$$
$$P\ \mathsf{sRed}\left(s_B\right)[{}^{x_A}/_{\mathsf{sRed}\left(t_A\right)}] \quad \text{if } s,t \text{ almost proper} \tag{P5}$$

The transformation will then beta and eta reduce terms not satisfying P. For this we need that beta reductions of the transformation of terms satisfy P and that P behaves naturally with respect to the productions of the grammar. For example P could be the property of an almost proper term of not being beta or eta reducible.

As we are assuming a valid derivation, we will only define this transformation on well-typed IHOL terms. We can then define the transformation of t_A (denoted by $\mathsf{sRed}\left(t_A\right)$) by induction on the shape of t_A as follows:

$$\mathsf{sRed}\left(t_A\right) \quad :=t_A \quad \text{if } t_A \text{ satisfies } P \text{ and } t \text{ has quasi-preimage of type } A \tag{SR1}$$
$$\mathsf{sRed}\left(f_{\Pi x:A.\ B}\ t_A\right) :=\mathsf{sRed}\left(f_{\Pi x:A.\ B}\right)\ \mathsf{sRed}\left(t_A\right) \quad \text{if } f_{\Pi x:A.\ B}\ t_A \text{ satisfies } P \text{ or isn't beta reducible} \tag{SR2}$$

In the following cases, we assume that the term t_A in $\mathsf{sRed}\,(\cdot)$ on the left of $:=$ is doesn't satisfy P or isn't almost proper with a quasi-preimage of type A:

$$\mathsf{sRed}\left(t_A\right) \quad :=\mathsf{sRed}\left(t^{\beta\eta}_{A}\right) \quad t \text{ is beta or eta reducible} \tag{SR3}$$
$$\mathsf{sRed}\left(s_A =_{\overline{A}} t_{A'}\right) \quad :=\mathsf{sRed}\left(s_A\right) =_{\overline{A}} \mathsf{sRed}\left(t_{A'}\right) \tag{SR4}$$
$$\mathsf{sRed}\left(F_{\mathsf{bool}} \Rightarrow G_{\mathsf{bool}}\right) \quad :=\mathsf{sRed}\left(F_{\mathsf{bool}}\right) \Rightarrow \mathsf{sRed}\left(G_{\mathsf{bool}}\right) \tag{SR5}$$
$$\mathsf{sRed}\left(\lambda x : A.\ s_B\right) \quad :=\lambda x : A.\ \mathsf{sRed}\left(s_B\right) \tag{SR6}$$
$$\mathsf{sRed}\left(f_{\Pi x:A.\ B}\ t_{A'}\right) \quad :=t_{\overline{B}} \tag{SR7}$$

Lemma 21. *Assume a well-typed DIHOLP theory T and a conjecture $\Gamma \vdash_T \varphi$ with Γ well-formed and φ well-typed. Assume a valid IHOL derivation of $\overline{\Gamma} \vdash^{H}_{\overline{T}} \overline{\varphi}$. Assume all terms in the theory T and conjecture satisfy property P. Then, there we can index the terms in the derivation s.t. any steps S in the derivation can be replaced by a macro-step for the P-normalizing statement transformation replacing step S s.t. after replacing all steps by their macro-steps:*

- *the resulting derivation is valid,*
- *all terms occuring in the derivation are almost proper,*
- *all terms occuring in the assumptions and conclusions of macro-steps that don't satisfy property P are beta and eta reduced.*

This is proven in Appendix K.

Using the lemma for P being true on all terms, we yield the corollary:

Corollary 22. *Let T be a well-formed DIHOLP theory and φ a well-formed conjecture in the well-formed context Γ relative to T using the validity judgement.*

Consider a valid derivation in IHOL of $\overline{\Gamma} \vdash^H_{\overline{T}} \overline{\varphi}$*. Then, there exists another derivation of this statement for which all terms occurring in the derivation are almost proper terms.*

4.3.4 Completeness of Translation 1 from DIHOLP into IHOL

Lemma 23. *Assume a well-formed DIHOLP theory* $\vdash\ T\ \mathsf{Thy}$ *and a context* Γ *that is well-formed relative to* T*. Assume that* $\overline{\Gamma} \vdash^H_{\overline{T}} \mathsf{Relat}_B[\overline{s} =_{\overline{B}} \overline{t}]$ *holds. It follows that* $\overline{\Gamma} \vdash^H_{\overline{T}} B^?\ \overline{s}$ *and* $\overline{\Gamma} \vdash^H_{\overline{T}} B^?\ \overline{t}$ *both hold.*

This is proven in Appendix L.

Lemma 24. *Assume that* $\Gamma \vdash_T \varphi : \mathsf{bool}$ *is derivable for* $\varphi = s =_A t$ *or* $\varphi = F \Rightarrow G$ *respectively in DIHOLP relative to a well-formed theory in a well-formed context. Then it follows that* $\Gamma \vdash_T s : A$ *and* $\Gamma \vdash_T t : A$ *or* $\Gamma \vdash_T F : \mathsf{bool}$ *and* $\Gamma \vdash_T G : \mathsf{bool}$ *respectively hold.*

Proof. Observe that for φ a possibly ill-typed formula in DIHOLP, $\varphi : \mathsf{bool}|_p$ is only derivable by rule ($|_p$I) for some $p : \Pi x : \mathsf{bool}.\ \mathsf{bool}$ and only if $\varphi : \mathsf{bool}$ is already known.

Continuing by the same inductive arguments as in the (joined) derivation of the rules (eqTyping), (implTypingL) and (implTypingR) but additionally repeating the above argument in the induction whenever necessary yields the claim. □

Theorem 25 (Completeness of Translation 1 from DIHOLP into IHOL)**.** *Assume a well-formed DIHOLP theory* $\vdash\ T\ \mathsf{Thy}$ *and conjecture* φ *in a context* Γ *that is well-formed relative to* T*.*

$$\overline{\Gamma} \vdash^H_{\overline{T}} \varphi \quad \textit{implies} \quad \Gamma \vdash F \quad \textit{for}\ \mathsf{RP}[\mathsf{R}[\varphi]] = \overline{F} \tag{10}$$

In the special case of $\varphi = \overline{F}$ *a proper term, this is exactly the claim of completeness of Translation 1 from DIHOLP into IHOL. In particular, this includes the case of term equality* $=_{\overline{A}}$*:*

$$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{t} =_{\overline{A}} \overline{t'} \wedge A^?\ \overline{t} \wedge A^?\ \overline{t'} \quad \textit{implies} \quad \Gamma \vdash_T t =_A t' \tag{11}$$

This is shown in Appendix M.

Inspecting the overall completeness proof for the first translation from DIHOLP into IHOL, we observe that it doesn't actually use the fact that we are considering intuitionistic logics. In fact, maybe somewhat surprisingly, the Translation 1 is complete also from DHOLP into HOL. The typing predicate $\mathsf{bool}^?$ becomes provably trivial in HOL, so the translation seemingly looses information, but the relativization of equalities over function types seems to prevent this from making the translation incomplete. ¨

For example, if we consider the correct conjecture $\neg\forall p : a\ \mathsf{true} \to \mathsf{bool}.\ \forall q : a\ \mathsf{true} \to \mathsf{bool}.\ p =_{a\to\mathsf{bool}} q$ for some empty type $a\ \mathsf{true}$, the relativization of the $\forall$s will be trivial, but the

inner equality is translated to $\forall\, x\ :\ \overline{(a\ T)}.\ (a\ T)^?\ x \Rightarrow p\ x\ =_{\mathsf{bool}}\ q\ x$ which is a trivially true statement since a true is an empty type and thence $(a\ T)^?$ never holds.

In the completeness proof we define replacement predicates and show the equivalence of their translations with the typing predicates. However being a predicate over the same type, the replacement predicate p_{bool} (and its translation) can be proven to be trivial by rule (boolExt) just like the typing predicate $\mathsf{bool}^?$ can. So the typing– and translated replacement predicates are still equivalent. Furthermore, if the replacement predicate p_{bool} holds on a term t the application must be well-typed so $t\ :\ \mathsf{bool}$ must be provable. Finally, the proof of completeness w.r.t. validity doesn't really depend on any properties of the typing predicates (other than their equivalence with the replacement predicates which still holds), so Translation 1 is actually complete w.r.t. validity also from DHOLP into HOL. As with the Translation 1 from DIHOLP into IHOL, we cannot easily prove the completeness w.r.t. typing and type equality (since we have to assume that the theory and conjecture are well-typed, this wouldn't be very useful anyways). But when translating into HOL we can actually show that the translation is **not** complete w.r.t. typing and type-equality for bool-valued function types. The fact that Translation 1 is not complete w.r.t. typing for bool-valued function types can be seen directly from Example 3. Similarly for different bool-valued function types of same image Translation 1 will create equivalent typing predicates, so Translation 1 is also not complete w.r.t. type-equality.

However, unlike the proof of completeness from DIHOLP into IHOL, we have not checked the arguments in the completeness proof (for validity) in sufficient detail to consider this proven, so we will only conjecture the following:

Conjecture 26. *Translation 1 from DHOLP into HOL is complete w.r.t. validity.*

4.3.5 Completeness of Translation 2 from DHOLP* into HOL* and HOL

The proof of completeness w.r.t. validity will be almost completely analogous to the proof for the completeness of the first translation from DIHOLP into IHOL– we just get one additional easy case for the inference rule (boolExt) in the proof by induction on the validity rules of Theorem 25. As a sanity check we verify that Lemma 20 also hold for the quasi-preimages of $\mathsf{bool}^?\ t$ for t a function application:

Lemma 27. *Given a well-formed DHOLP* theory T and well-formed DHOLP* context Γ relative to T, let A be a well-formed type and F any almost proper formula.*

Let φ denote the quasi-preimage of $\mathsf{bool}^?\ \overline{F}$. If $\Gamma \vdash_T \varphi$ is derivable in DHOLP, then $\Gamma \vdash_T F\ :\ A$ is also derivable.*

This is proven in Appendix N.

Corollary 28. *The Translation 2 from DHOLP* into HOL* is also complete w.r.t. validity.*

Proof. This can be proven analogously to the proof of Theorem 25. □

Lemma 29. *HOL is a conservative extension of HOL*.*

This is proven in Appendix O using proof transformations.

Corollary 30. *Combining Corollary 28 and the above Lemma 29, it follows that the translation from DHOLP* into HOL is also complete w.r.t. the validity judgement.*

4.4 Typechecking DHOLP* theories

Since we cannot use the translation for checking typing and type-equality judgements over DI-HOLP and DHOLP* theories (we need to assume well-formed theories in the completeness theorem), we will once again describe how we can check them nonetheless using the translation. Typing judgements are decided based on the rules (lambda'), (appl'), (⇒type'), (var), (const), (=type), (cong:), (∀type), ($|_p$I), ($|_p$E:). Checking the assumptions of these rules is easy if we can check type equality judgements (in case of rule (cong:)) and validity judgements (in case of rule ($|_p$I)).

Type-equality judgements are decided based on the rules (congBase'), (congΠ), ($|_p \equiv$). Checking the assumptions of the rules is easy if we can decide term equalities (needed for (congBase')).

As shown in the subsections 4.2 and 4.3, Translation 1 of DHOLP into HOL is both sound and complete. Consequently, we can use the translation and a HOL prover to decide validity judgements in DHOLP.

There is one caveat we need to think about: It is important to check declarations in the theory and context assumptions in order. This way previous declarations of the theory and assumptions in the context (only on those we are allowed to depend on) are known to be well-typed, so the theory and context they form are well-formed. Furthermore for typechecking a term t it is important to first typecheck the quantifier in whose scope t occurs, to first typecheck the subterms of t and to first check typing against the ambient type in case of a predicate subtype before checking if the predicate holds on t. Then, the proof obligations generated for the prover will be well-typed conjectures relative to a well-formed theory and in a well-formed context and we can apply our completeness results.

5 Other translations and summary of soundness and completeness results

Translating from classical to intuitionistic logic For this we modify case IT16 in the definition of Translation 1 to:

$$\mathsf{bool}^? \; x \qquad :=x =_{\mathsf{bool}} \mathsf{true} \vee x =_{\mathsf{bool}} \mathsf{false} \tag{IT34}$$

We observe that the rule (boolExt) is equivalent to the following one:

$$\frac{\vdash_T \Gamma \; \mathsf{Ctx}}{\Gamma \vdash_T \forall x : \mathsf{bool}.\; x =_{\mathsf{bool}} \mathsf{true} \vee x =_{\mathsf{bool}} \mathsf{false}} \text{boolExt'}$$

Rule (boolExt) implies rule (boolExt') as can be seen by inserting $\lambda x : \mathsf{bool}.\ x =_{\mathsf{bool}} \mathsf{true} \vee x =_{\mathsf{bool}} \mathsf{false}$ for p in (boolExt).

Rule (boolExt') also implies rule (boolExt), which can be seen as follows: Assume that $\Gamma \vdash_T p\ \mathsf{true}$ and $\Gamma \vdash_T p\ \mathsf{false}$. Then $\vdash_T \Gamma\ \mathsf{Ctx}$ follows and hence we have $\Gamma,\ x : \mathsf{bool} \vdash_T \lambda x : \mathsf{bool}.\ x =_{\mathsf{bool}} \mathsf{true} \vee x =_{\mathsf{bool}} \mathsf{false}$. Observe that assuming $x =_{\mathsf{bool}} \mathsf{true}$ applying rule (rewrite) yields $p\ x$ and analogously for $x =_{\mathsf{bool}} \mathsf{false}$ applying the rule yields $p\ x$. Thus $\Gamma,\ x : \mathsf{bool} \vdash_T x =_{\mathsf{bool}} \mathsf{true} \Rightarrow p\ x$ and $\Gamma,\ x : \mathsf{bool} \vdash_T x =_{\mathsf{bool}} \mathsf{false} \Rightarrow p\ x$ hold. By definition of $\vee$ and using rule ($\Rightarrow$E) it follows that $\Gamma,\ x : \mathsf{bool} \vdash_T p\ x$, so rule ($\forall$I) implies the desired conclusion of $\Gamma \vdash_T \forall x : \mathsf{bool}.\ p\ x$.

Secondly, we observe that the translation of the formula in the conclusion of rule (boolExt') is translated to a trivially true formula by the modified version of Translation 1. We have:

$$\begin{aligned}
&\overline{\forall x : \mathsf{bool}.\ x =_{\mathsf{bool}} \mathsf{true} \vee x =_{\mathsf{bool}} \mathsf{false}} \\
=&\forall x : \overline{\mathsf{bool}}.\ \mathsf{bool}^? x \Rightarrow \left(x =_{\mathsf{bool}} \mathsf{true} \wedge \left(x =_{\mathsf{bool}} \mathsf{true} \vee x =_{\mathsf{bool}} \mathsf{false}\right) \wedge \mathsf{bool}^? \mathsf{true}\right) \\
&\vee \left(x =_{\mathsf{bool}} \mathsf{false} \wedge \left(x =_{\mathsf{bool}} \mathsf{true} \vee x =_{\mathsf{bool}} \mathsf{false} \wedge \mathsf{bool}^? \mathsf{false}\right)\right).
\end{aligned}$$

Since $\mathsf{bool}^? \mathsf{true}$ and $\mathsf{bool}^? \mathsf{false}$ both hold and by definition we have $\mathsf{bool}^? x = x =_{\mathsf{bool}} \mathsf{true} \vee x =_{\mathsf{bool}} \mathsf{false}$, this can be further simplified to

$$\forall x : \mathsf{bool}.\ x =_{\mathsf{bool}} \mathsf{true} \vee x =_{\mathsf{bool}} \mathsf{false} \Rightarrow x =_{\mathsf{bool}} \mathsf{true} \vee x =_{\mathsf{bool}} \mathsf{false},$$

which is trivially true.

Therefore soundness is not affected by adding the rule (boolExt) to DIHOL or DIHOLP (but it is not clear if the modified case (IT16) makes the translation unsound). Completeness is also not affected by adding an additional rule to the source logic and is probably also not affected by the modified translation as the additional information it provides is hopefully compensated for by the additional rule (boolExt) in the source logic. Thus it would be interesting to study the soundness and completeness of this translation from DHOLP to IHOL.

Alternatively, we can also translate normally with Translation 1 from DHOLP to IHOL, this translation will be unsound but complete (by Conjecture 26).

Classical logic with external propositions instead of bool Instead of adding Π-types and predicate subtypes to HOL*, we can instead add Π-types and predicate subtypes to HOLE and translate from the resulting logic (denoted by DHOLPE) to HOL. Since it is not possible to quantify over prop, the translating to HOL is complete (and also sound) for the same reason as the translation for DHOLP*.

Summary of soundness and completeness results The overall soundness (with the assumption on the theory for soundness w.r.t. validity given in Theorem 14) and completeness (w.r.t. validity) results are summarized in Table 2:

[3] Since we have the additional inference rule (boolExt) in the classical logic.

	Translation from				
Translation to	DIHOL	DIHOLP	DHOLP	DHOLP*	DHOLPE
IHOL via Translation 1	sound by Theorem 8, complete by Theorem 11	sound by Theorem 14, complete by Theorem 25	unsound[3], probably complete by Conjecture 26	unsound[3], complete by Theorem 25	-
HOL via Translation 1	sound by Theorem 8, incomplete[3]	sound by Corollary 9, incomplete[3]	sound by Corollary 15, probably complete by Conjecture 26	sound by Corollary 16, probably complete by Conjecture 26	-
HOL via Translation 2	-	-	-	sound by Corollary 16, complete by Corollary 30	-
HOL* via Translation 1	-	-	-	sound by Corollary 16, maybe complete	-
HOL* via Translation 2	-	-	-	sound by Corollary 16, complete by Corollary 28	-
HOLE	-	-	-	-	sound by Corollary 16, complete by Corollary 28

Table 2: Summary of sound– and completeness results for different translations

6 TPTP encoding and implementation of prover

In this section we discuss our implementation of Translation 1 from DHOL to HOL as a preprocessor for HOL ATPs. We also discuss the implementation of both translation inside the MMT system and how they are used in a hammer for DHOLP based theories in MMT. We start with the discussion of the preprocessor for the LEO-III prover.

We choose the LEO-III system because it is well-developed and there is an already existing preprocessor infrastructure as well as an additional logic embedding tool [36], which can be extended to implement Translation 1 from DHOL or HOL.

For this it is helpful that TPTP already supports DHOL* syntax although — to our knowledge — no ATP system has made use of it so far. Concretely, the type $\Pi x : A.\ B$ is encoded as `!>[X:A]B` and a base type $a\ t_1 \dots t_n$ as `a @ t1 ... @ tn`. Taking advantage of this, it was straightforward to extend the logic embedding tool to DHOL. Since we cannot currently express DHOLP in TPTP (it seems a future TPTP language is needed here), we unfortunately cannot currently implement the translation from DHOLP.

The logic embedding tool now accepts TPTP problems using the DHOL logic and translates them to TPTP problems using the HOL logic in the language TPTP THF. LEO-III (or any other HOL prover) can then be run on the translated problem.

Furthermore, we developed a bridge between the MMT system [30] and LEO-III. This is implemented as an exporter from MMT theories using DHOLP or DHOLP* as meta-language to TPTP problems. This exporter is used by MMT to translate proof obligations generated by MMT's internal provers which then runs the LEO-III prover on those problems, yielding a hammer for DHOLP in MMT. Given an implementation of a translation to TPTP THF problems, this implementation extends also to other logics, furthermore it is trivial to change the implementation to use other HOL provers as well.

We have added the type equality rules (congΠ) and ($|_p \equiv$) to MMT's formalizations of DHOL* resp. DHOLP*, so MMT's type-checker can take advantage of this prover for type-checking theories based on those logics.

The implementation of the translations can be found in this folder. The formalization of DHOLP and related logics in MMT can be found in this file. Finally the example theories in DHOL and DHOLP given throughout this thesis (and some more) are formalized in MMT in this folder.

This way, we build two different systems for different purposes:

- An MMT-based system that uses Leo as a backend. This system provides a type-checker, IDE, and knowledge management system for DHOLP using our extension as a hammer tool. The type-checkers call LEO-III to discharge proof obligations in a way that is transparent to the user.
- A LEO-III-based system that provides a general purpose DHOL* ATP that accepts input in TPTP encoding. This acts as a preprocessor for Leo, which translates the DHOL* TPTP problem into a HOL* TPTP problem on which LEO-III (or any other HOL prover for that matter) can be called. In the future it would be nice to better integrate this implementation with LEO-III to also type-check the problem in a similar way as done with the MMT based system. Once the TPTP syntax supports predicate subtypes, this can be further generalized to DHOLP* TPTP problems.

As an example of how the TPTP translation looks like, consider the translation of example 6 to IHOL. This translation is generated as an intermediate step in the implementation of the MMT based prover for DHOLP, but immediately translated further into HOL TPTP.

The resulting IHOL TPTP problem is given in example 7 below: Translating example 6 to HOL TPTP yields:

Example 7 (Translation of set theory example to HOL TPTP using Translation 1).

$thf(\mathsf{bool}^?\_type,\ type,\ \mathsf{bool}^? : \$o > \$o).$

$thf(ax1,\ axiom,\ \mathsf{bool}^?(\$true)).$

$thf(ax2,\ axiom,\ \mathsf{bool}^?(\$false)).$

$thf(\mathsf{set}\_type,\ type,\ \mathsf{set} : \$tType).$

$thf(\text{set}^?\_type,\ type,\ \text{set}^? : \text{set} > \$o).$

$thf(\texttt{of}\_type,\ type,\ \texttt{of} : \text{set} > \text{set} > \$o).$

$thf(ax3,\ axiom,\ ! > [a : \text{set}] :\ \text{set}^?\ (a) \Rightarrow\ ! > [x : \text{set}] :\ \text{set}^?\ (x) \Rightarrow\ \text{bool}^?\,(\texttt{of}(a,\ s))).$

$thf(\texttt{subs}\_type,\ type,\ \texttt{subs} : \text{set} > \text{set} > \$o).$

$thf(ax4,\ axiom,\ ! > [a : \text{set}] :\ \text{set}^?\ (a) \Rightarrow\ ! > [s : \text{set}] :\ \text{set}^?\ (s) \Rightarrow\ \text{bool}^?\,(\texttt{subs}(a,\ s))).$

$$thf(ax5,\ axiom,\ ! > [a : \text{set}] :\ \text{set}^?\ (a) \Rightarrow\ ! > [s : \text{set}] :\ \text{set}^?\ (s) \Rightarrow$$
$$(s \subset a) = \left(! > [y : \text{set}] :\ \text{set}^?\ (y) \Rightarrow\ \texttt{of}\,(s,\ y) \Rightarrow \texttt{of}\,(a,\ y)\right)$$
$$/\backslash\ \text{bool}^?\,(s \subset a)\ /\backslash\ \text{bool}^?\ s\ /\backslash\ \text{bool}^?\ a\ ).$$

$thf(\texttt{func}\_type,\ type,\ \texttt{func} : \text{set} > \text{set} > \text{set}).$

$thf(ax6,\ axiom,\ ! > [a : \text{set}] :\ \text{set}^?\ (a) \Rightarrow\ ! > [b : \text{set}] :\ \text{set}^?\ (b) \Rightarrow\ \text{set}^?\,(\texttt{func}\ (a,\ b))).$

$thf(\texttt{appl}\_type,\ type,\ \texttt{appl} : \text{set} > \text{set} > \text{Elem} > \text{Elem} > \text{Elem}).$

$$thf(ax7,\ axiom,\ ! > [a : \text{set}] :\ \text{set}^?\ (a) \Rightarrow\ ! > [b : \text{set}] :\ \text{set}^?\ (b) \Rightarrow$$
$$! > [f : \text{set}] :\ \texttt{of}\,(\texttt{func}\ (a,\ b),\ f)\ /\backslash\ \text{set}^?\,(f)\ =>\ ! > [x : \text{set}] :\ \texttt{of}\,(a,\ x)\ /\backslash\ \text{set}^?\,(x)\ =>$$
$$\texttt{of}\,(b,\ \texttt{appl}\ (f,\ x,\ a,\ b))\ /\backslash\ \text{set}^?\,(\texttt{appl}\ (f,\ x,\ a,\ b))).$$

$$thf(ax8,\ axiom,\ ! > [a : \text{set}] :\ \text{set}^?\ (a) \Rightarrow\ ! > [b : \text{set}] :\ \text{set}^?\ (b) \Rightarrow$$
$$! > [f : \text{set}] :\ \texttt{of}\,(\texttt{func}\ (a,\ b),\ f)\ /\backslash\ \text{set}^?\,(f)\ =>$$
$$! > [g : \text{set}] :\ \texttt{of}\,(\texttt{func}\ (a,\ b),\ g)\ /\backslash\ \text{set}^?\,(g)\ =>\ ! > [x : \text{set}] :\ \texttt{of}\,(a,\ x)\ /\backslash\ \text{set}^?\,(x)\ =>$$
$$\texttt{appl}\ (f,\ x,\ a,\ b) = \texttt{appl}\ (g,\ x,\ a,\ b)\ /\backslash\ \texttt{of}\,(b,\ \texttt{appl}\ (f,\ x,\ a,\ b))\ /\backslash\ \text{set}^?\ (\texttt{appl}\ (f,\ x,\ a,\ b))$$
$$/\backslash\ \texttt{of}\,(b,\ \texttt{appl}\ (g,\ x,\ a,\ b))\ /\backslash\ \text{set}^?\ (\texttt{appl}\ (g,\ x,\ a,\ b)) \Rightarrow$$
$$f = g\ /\backslash\ \texttt{of}\,(\texttt{func}\ (a,\ b),\ f)\ /\backslash\ \text{set}^?\ (f)\ /\backslash\ \texttt{of}\,(\texttt{func}\ (a,\ b),\ g)\ /\backslash\ \text{set}^?\ (g)).$$

$thf(\texttt{funcRestr}\_type,\ type,\ \texttt{funcRestr} : \text{set} > \text{set} > \text{set} > \text{set} > \text{set}).$

$$thf(ax9,\ axiom,\ ! > [a : \text{set}] :\ \text{set}^?\ (a) \Rightarrow\ ! > [b : \text{set}] :\ \text{set}^?\ (b) \Rightarrow$$
$$! > [f : \text{set}] :\ \texttt{of}\,((\texttt{func}\ (a,\ b)),\ f)\ /\backslash\ \text{set}^?\,(f)\ =>$$
$$! > [s : \text{set}] :\ \texttt{subs}\,(s,\ a)\ /\backslash\ \text{set}^?\,(s)\ =>\ \text{set}^?\,(\texttt{funcRestr}\,(f,\ s,\ a,\ b))).$$

$$thf(ax10,\ axiom,\ ! > [a : \text{set}] :\ \text{set}^?\ (a) \Rightarrow\ ! > [b : \text{set}] :\ \text{set}^?\ (b) \Rightarrow$$
$$! > [f : \text{set}] :\ \texttt{of}\,((\texttt{func}\ (a,\ b)),\ f)\ /\backslash\ \text{set}^?\,(f)\ =>$$
$$! > [s : \text{set}] :\ \texttt{subs}\,(s,\ a)\ /\backslash\ \text{set}^?\,(s)\ =>\ ! > [x : \text{set}] :\ \texttt{of}\,(s,\ x)\ /\backslash\ \text{set}^?\,(x)\ =>$$
$$\texttt{appl}\ (f,\ x,\ a,\ b) = \texttt{appl}\ ((\texttt{funcRestr}\,(f,\ s,\ a,\ b)),\ x,\ a,\ b)$$
$$/\backslash\ \texttt{of}\,(b,\ \texttt{appl}\ (f,\ x,\ a,\ b))\ /\backslash\ \text{set}^?\ (\texttt{appl}\ (f,\ x,\ a,\ b))$$
$$/\backslash\ \texttt{of}\,(b,\ \texttt{appl}\ ((\texttt{funcRestr}\,(f,\ s,\ a,\ b)),\ x,\ a,\ b))\ /\backslash$$
$$\text{set}^?\ (\texttt{appl}\ ((\texttt{funcRestr}\,(f,\ s,\ a,\ b)),\ x,\ a,\ b))).$$

The conjecture is translated to:

$$thf(conj,\ conjecture,\ ! > [a : \text{set}] :\ \text{set}^?\ (a) \Rightarrow\ ! > [d : \text{set}] :\ \text{set}^?\ (d) \Rightarrow$$
$$! > [b : \text{set}] :\ \texttt{subs}\,(b,\ a)\ /\backslash\ \text{set}^?\,(b)\ =>\ ! > [s : \text{set}] :\ \texttt{subs}\,(s,\ b)\ /\backslash\ \text{set}^?\,(s)\ =>$$
$$! > [f : \text{set}] :\ \texttt{of}\,(\texttt{func}\ (a,\ b),\ f)\ /\backslash\ \text{set}^?\,(f)\ =>$$
$$\texttt{funcRestr}\,((\texttt{funcRestr}\,(f,\ b,\ a,\ b)),\ s,\ a,\ b) = \texttt{funcRestr}\,(f,\ s,\ a,\ b)$$

$$/\backslash\,\mathsf{Elem}^{?}\ (\mathtt{func}\ s\ d)\ (\mathtt{funcRestr}\,((\mathtt{funcRestr}\,(f,\ b,\ a,\ b)),\ s,\ a,\ b))$$
$$/\backslash\,\mathsf{Elem}^{?}\ (\mathtt{func}\ s\ d)\ (\mathtt{funcRestr}\,(f,\ s,\ a,\ b))\ ).$$

This is also the result yielded by translating the theory in example 5 using the DHOLP to TPTP exporter in MMT. Note however that the MMT based prover doesn't directly use this translation, but instead only translates those proof obligations MMT cannot solve itself to HOL TPTP.

Example use case for MMT implementation: Mizar proofs As illustrated in the above example 5, DHOLP is quite suitable to express set theory, using predicate subtypes of a set type to represent types and an elementhood predicate to represent typing. The Mizar system is based on set theory and can be formalized in DHOLP similarly to how set theory is formalized in example 5. Recently, [32] the entire Mizar mathematical library has been imported to a formalization in MMT based on soft-typed first-order logic. After replacing the formalization with an equivalent one based on DHOLP (or a translation of the previous formalization into DHOLP), we can use the MMT implementation of our translation to prove Mizar theorems in MMT.

7 Extending DHOLP with parametric predicates

To obtain DPHOL with parametric predicates, we extend the DPHOL grammar as follows:

T	$::=$	$* \mid T,\ Dec$	theories
Dec	$::=$	$c : A \mid a : \left(\Pi x : A.\ \right)^{*}\ \mathsf{tp} \mid F \mid d : \left[D_1, \ldots, D_m\right]\ \left(\Pi x : A.\ \right)^{*}\ \mathsf{bool}$	declarations
Γ	$::=$	$. \mid \Gamma, x : A \mid \Gamma, F \mid \Gamma, D : \mathsf{tp}$	contexts
A, B	$::=$	$a\ t_1 \ldots t_n \mid \Pi x : A.\ B \mid \mathsf{bool} \mid A\vert_p$	types
s, t, c, f, F, G	$::=$	$c \mid x \mid f\ t \mid \lambda x : B.\ t \mid s =_B t \mid F \Rightarrow G \mid d\ D_1\ \ldots\ D_m\ t_1 \ldots t_n$	terms

The additional inference rules for DPHOL with parametric predicates are:

$$\frac{\vdash_T D_1 : \mathsf{tp}, \ldots, D_m : \mathsf{tp}, x_1 : A_1, \ldots, x_n : A_n\ \mathsf{Ctx}}{\vdash T,\ d : \left[D_1, \ldots, D_m\right]\ \Pi x_1 : A_1.\ \ldots \Pi x_n : A_n.\ \mathsf{tp}\ \ \mathsf{Thy}}\ \text{thyPPred} \qquad \frac{\vdash_T \Gamma\ \mathsf{Ctx}}{\vdash_T \Gamma,\ D : \mathsf{tp}\ \mathsf{Ctx}}\ \text{ctxTp} \qquad \frac{D : \mathsf{tp}\ \text{in}\ \Gamma}{\Gamma \vdash_T D\ \mathsf{tp}}\ \text{TVar}$$

$$\frac{\begin{array}{c} d : \left[D_1, \ldots, D_m\right]\ \Pi x_1 : A_1.\ \ldots \Pi x_n : A_n.\ \mathsf{tp}\ \ \text{in}\ T \\ \Gamma,\ D_1 : \mathsf{tp},\ \ldots,\ D_m : \mathsf{tp} \vdash_T t_1 : A_1 \\ \vdots \\ \Gamma,\ D_1 : \mathsf{tp},\ \ldots,\ D_m : \mathsf{tp},\ x_1 : A_1,\ \ldots,\ x_{n-1} : A_{n-1}[x_1/t_1] \ldots [x_{n-2}/t_{n-2}] \vdash_T t_n : A_n[x_1/t_1] \ldots [x_{n-1}/t_{n-1}] \end{array}}{\Gamma \vdash_T d\ D_1 \ldots\ D_m\ t_1\ \ldots\ t_n : \mathsf{bool}}\ \text{PPredTp}$$

and using the notation $\Delta := \Gamma,\ D_1 : \mathsf{tp}, \ldots, D_m : \mathsf{tp}, D'_1 : \mathsf{tp}, \ldots, D'_p : \mathsf{tp}$:

$$\frac{\begin{array}{c} d : \left[D_1, \ldots, D_m\right]\ \Pi x_1 : A_1.\ \ldots \Pi x_n : A_n.\ \mathsf{tp}\ \ \text{in}\ T \quad d' : \left[D'_1, \ldots, D'_p\right]\ \Pi x_1 : A'_1.\ \ldots \Pi x_q : A'_n.\ \mathsf{tp}\ \ \text{in}\ T \\ \Delta \vdash_T A_1 \equiv A'_1 \quad \ldots \quad \Delta,\ x_1 : A_1,\ \ldots,\ x_{n-1} : A_{n-1}[x_1/t_1] \ldots [x_{n-2}/t_{n-2}] \vdash_T A_n[x_1/t_1] \ldots [x_{n-1}/t_{n-1}] \equiv A'_n[x_1/t_1] \ldots [x_{n-1}/t_{n-1}] \\ \Delta \vdash_T s_1 =_{A_1} t'_1 \quad \ldots \quad \Delta \vdash_T s_n =_{A_n[x_1/t_1] \ldots [x_{n-1}/t_{n-1}]} t'_n \end{array}}{\Gamma \vdash_T d\ D_1 \ldots\ D_m\ s_1\ \ldots\ s_n =_{\mathsf{bool}} d'\ D'_1 \ldots\ D'_p\ t_1\ \ldots\ t_n}\ \text{congPPred}$$

Translation to DPHOL

The translation for parametric predicate declaration is similar to the translation of equality for non-function types.

The translation of DPHOL with parametric predicates to DPHOL is given by induction on the grammar. The interesting cases (that differ from the cases in the definition of Translation 1) are:

$$\overline{T,\ d : \left[D_1, \ldots, D_m\right]\ \Pi x_1 : A_1.\ \ldots \Pi x_n : A_n.\ \mathsf{bool}} := \overline{T},\ d : \Pi x_1 : A_1.\ \ldots \Pi x_n : A_n.\ \mathsf{bool} \tag{PT40}$$

$$\overline{\Gamma,\ D} := \overline{\Gamma} \tag{PT41}$$

We need to define the abbreviation $\mathsf{bool}^?\ t$ for t being the application of a parametric predicate:

$$\mathsf{bool}^?\ \overline{d\ D_1\ \ldots\ D_n\ t_1\ \ldots\ t_n} := d^?\ \overline{D_1}\ \ldots\ \overline{D_m}\ \overline{t_1}\ \ldots\ \overline{t_n} \tag{PT42}$$

This uses the abbreviation $d^?\ \overline{D_1\ \ldots\ D_m\ t_1\ \ldots\ t_n}$, which is defined for a parametric predicate declaration $d : \left[D_1, \ldots, d_m\right]\ \Pi x_1 : A_1.\ \ldots \Pi x_n : A_n.\ \mathsf{bool}$ by:

$$d^?\ \overline{D_1\ \ldots\ D_m\ t_1\ \ldots\ t_n} := {A_1}^?\ \overline{t_1} \wedge \ldots \wedge {A_n}^?\ \overline{t_n}$$

The cases for terms are:

$$\begin{aligned}\overline{d\ D_1\ \ldots\ D_n\ t_1\ \ldots\ t_n} := {A_1}^?\ t_1 \Rightarrow \ldots \Rightarrow \left(A_n[{}^{x_1}/_{t_1}] \ldots [{}^{x_{n-1}}/_{t_{n-1}}]\right)^?\ t_n \Rightarrow \left(d\ \overline{t_1}\ \ldots\ \overline{t_n}\right)\\ \wedge\ d^?\ \overline{D_1}\ \ldots\ \overline{D_m}\ \overline{t_1}\ \ldots\ \overline{t_n}\end{aligned} \tag{PT43}$$

$$\bar{t} := t \qquad \text{if } t \text{ a DPHOL term} \tag{PT44}$$

8 Conclusion and Future Work

We have investigated ways to extend the applicability of HOL provers by translating dependently typed problems to equivalent problems in higher-order logic. We have defined DHOLP as an extension of HOL with classical booleans and equality (unlike other dependent type theories which use propositions-as-types) to facilitate the use of the translation for automated theorem proving. We have focused on two additional typing features, Π-types and predicate subtypes yielding the logic DHOLP and given a translation of DHOLP into HOL. A further extension of DHOLP with parametric predicates and of our translation for it is given.

This approach has several drawbacks, including undecidable type-checking (arguably an acceptable price in practice) and problems translating dependent types with eventual return type bool. Furthermore, we cannot prove the soundness w.r.t. validity if the theory contains (undefined) constants of types of the form $\Pi x : \Pi y : B.\ C.\ D$. Since we mostly care about theories with defined constants, this assumption is relatively harmless, furthermore being a soundness issue this doesn't restrict the applicability of the translation for use in a hammer. The issue regarding the bool-valued function types doesn't affect the soundness of our translation or the completeness of the translation restricted to the intuitionistic fragment of the source. Somewhat surprisingly, it also doesn't affect the completeness of the translation w.r.t. validity. Since we can decide typing and type-equality statements using the soundness and completeness of the translation w.r.t. validity, this is exactly what we need in practice.

Nevertheless, we have described a second translation which is complete also w.r.t. typing and type equality. However, this requires disallowing quantification over bool-valued function-types in the source logic, yielding a (sound with same restriction and) complete translation from this fragment DHOLP* into the corresponding fragment HOL* of HOL. We have proven that HOL is a conservative extension of HOL*, which implies that also the translation to HOL is complete. As our completeness proof(s) are constructive, they also provide a way to reconstruct proofs in DHOLP (although in practice, a simpler and more practical method for reconstructing proofs should be used).

We have implemented both translations as extensions to MMT and integrated them into a prover infrastructure that cooperates with the LEO-III prover, yielding a hammer for MMT theories based on DHOLP. This way we have archived our practical goal: developing a hammer for MMT theories based on DHOLP. However, at the moment we do not have enough example theories in MMT based on DHOLP to seriously evaluate the effectiveness of the hammer.

We have shown how set theory can be formalized within DHOLP and discussed how we can modify the formalization of the foundation of the Mizar system in MMT to be based on it, allowing the use of our prover implementation for theorems imported from the Mizar mathematical library. This would also produce many examples that can be used to test and evaluate the effectiveness of the hammer implementation.

These results are interesting for several reasons:

1. Automated theorem provers have been extended to support more and more expressive input logics, mirroring the development of new TPTP dialects for more expressive logics (e.g. TPTP TH1 corresponding to higher-order logic with rank-1 polymorphism). Our results are a further step towards extending the supported input logics of ATPs.
2. The methods we developed for the completeness proof (considering termwise-injectivity, the proof transformations in (I)HOL and the replacement predicates and their use in defining quasi-preimages of terms) can be reused for similar translations. Furthermmore, we provide criteria (like injectivity for terms of same type) and intuition for the completeness of such translations. We also give some counterexamples for more naive translations that can benefit the development of other translations. Furthermore, the implementation in MMT can be easily extended with other translations yielding support for other logics. Our work therefore constitutes a case study of how we can extend HOL provers to more expressive logics based on translations into HOL.

It would be natural to study how our work can be further generalized. Especially, Σ-types are an interesting extension to consider. Since we already support Π-types and predicate subtypes we can almost define them (to do this we need to add a description operator in DHOLP and translate it to an operator that additionally takes a typing predicate as an argument in HOL), so we expect that DHOLP and its translations can be extended with Σ-types without major problems.

References

[1] ANDREWS, P. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*. Academic Press, 1986.

[2] ANDREWS, P., BISHOP, M., ISSAR, S., NESMITH, D., PFENNING, F., AND XI, H. TPS: A Theorem-Proving System for Classical Type Theory. *Journal of Automated Reasoning 16*, 3 (1996), 321–353.

[3] BANCEREK, G., BYLIŃSKI, C., GRABOWSKI, A., KORNIŁOWICZ, A., R. MATUSZEWSKI, A. N., PĄK, K., AND URBAN, J. Mizar: State-of-the-art and beyond. In *Intelligent Computer Mathematics* (Cham, 2015), M. Kerber, J. Carette, C. Kaliszyk, F. Rabe, and V. Sorge, Eds., Springer International Publishing, p. 261–279.

[4] BART, J., AND MELHAM, T. Translating dependent type theory into higher order logic. In *Typed Lambda Calculi and Applications* (1993), M. Bezem and J. F. Groote, Eds., Springer Berlin, Heidelberg, p. 209–229.

[5] BLANCHETTE, J., KALISZYK, C., PAULSON, L., AND URBAN, J. Hammering towards qed. *Journal of Formalized Reasoning 9*, 1 (2016), 101–148.

[6] BORDG, A., PAULSON, L., AND LI, W. Simple type theory is not too simple: Grothendieck's schemes without dependent types. *Experimental Mathematics 31*, 2 (2022), 364–382.

[7] BROWN, C. E. Satallax: An automatic higher-order prover. In *Automated Reasoning* (2012), B. Gramlich, D. Miller, and U. Sattler, Eds., Springer Berlin, Heidelberg, p. 111–117.

[8] CHURCH, A. An Unsolvable Problem of Elementary Number Theory. *American Journal of Mathematics 58* (1936).

[9] CHURCH, A. A Formulation of the Simple Theory of Types. *Journal of Symbolic Logic 5*, 1 (1940), 56–68.

[10] COQ DEVELOPMENT TEAM. The Coq Proof Assistant: Reference Manual. Tech. rep., INRIA, 2015.

[11] CZAJKA, L. A Shallow Embedding of Pure Type Systems into First-Order Logic. In *22nd International Conference on Types for Proofs and Programs (TYPES 2016)* (Dagstuhl, Germany, 2018), S. Ghilezan, H. Geuvers, and J. Ivetić, Eds., vol. 97 of *Leibniz International Proceedings in Informatics (LIPIcs)*, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, p. 9:1–9:39.

[12] CZAJKA, L., AND KALISZYK, C. Hammer for coq: Automation for dependent type theory. *Journal of Automated Reasoning 61*, 1-4 (2018), 423–453.

[13] D. HILBERT, W. A. *Grundzüge der theoretischen Logik*. Springer, Berlin, Heidelberg, 1928.

[14] DE MOURA, L., KONG, S., AVIGAD, J., VAN DOORN, F., AND VON RAUMER, J. The Lean Theorem Prover (System Description). In *Automated Deduction* (Cham, 2015), A. Felty and A. Middeldorp, Eds., Springer International Publishing, p. 378–388.

[15] FREGE, G. *Begriffsschrift: eine der arithmetischen nachgebildete Formelsprache des reinen Denkens*. Verlag von Louis Nebert, Halle, 1879.

[16] GÖDEL, K. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme. *Monatshefte für Mathematik und Physik 38* (1931), 173–198. English title: On Formally Undecidable Propositions Of Principia Mathematica And Related Systems.

[17] GORDON, M. HOL: A Proof Generating System for Higher-Order Logic. In *VLSI Specification, Verification and Synthesis*, G. Birtwistle and P. Subrahmanyam, Eds. Kluwer-Academic Publishers, 1988, p. 73–128.

[18] GORDON, M., AND PITTS, A. The HOL Logic. In *Introduction to HOL, Part III*, M. Gordon and T. Melham, Eds. Cambridge University Press, 1993, p. 191–232.

[19] HARPER, R., HONSELL, F., AND PLOTKIN, G. A framework for defining logics. *Journal of the Association for Computing Machinery 40*, 1 (1993), 143–184.

[20] HARRISON, J. HOL Light: A Tutorial Introduction. In *Proceedings of the First International Conference on Formal Methods in Computer-Aided Design* (1996), Springer Berlin, Heidelberg, p. 265–269.

[21] HENKIN, L. Completeness in the Theory of Types. *Journal of Symbolic Logic 15*, 2 (1950), 81–91.

[22] KALISZYK, C., AND URBAN, J. HOL(y)hammer: Online ATP service for HOL light. *Mathematics in Computer Science 9*, 1 (2015), 5–22.

[23] KALISZYK, C., AND URBAN, J. MizAR 40 for Mizar 40. *Journal of Automated Reasoning 55* (2015), 245–256.

[24] NORELL, U. The Agda WiKi, 2005. http://wiki.portal.chalmers.se/agda.

[25] OBERSCHELP, A. Untersuchungen zur mehrsortigen quantorenlogik. *Mathematische Annalen 145* (1962), 297–333.

[26] PAULIN-MOHRING, C. Introduction to the Calculus of Inductive Constructions. In *All about Proofs, Proofs for All*, B. W. Paleo and D. Delahaye, Eds., vol. 55 of *Studies in Logic (Mathematical logic and foundations)*. College Publications, Jan. 2015.

[27] PAULSON, L. *Isabelle: A Generic Theorem Prover*, vol. 828 of *Lecture Notes in Computer Science*. Springer Berlin, Heidelberg, 1994.

[28] PAULSON, L. C., AND BLANCHETTE, J. C. Three years of experience with sledgehammer, a practical link between automatic and interactive theorem provers. In *International Workshop on the Implementation of Logics* (2010), G. Sutcliffe, E. Ternovska, and S. Schulz, Eds.

[29] PFENNING, F., AND SCHÜRMANN, C. System description: Twelf — a meta-logical framework for deductive systems. In *Automated Deduction* (1999), H. Ganzinger, Ed., p. 202–206.

[30] RABE, F. MMT: The Meta Meta Tool. see https://kwarc.info/people/frabe/Research/rabe_mmtsys_20.pdf, 2020.

[31] RIAZANOV, A., AND VORONKOV, A. The design and implementation of Vampire. *AI Communications 15* (2002), 91–110.

[32] ROTHGANG, C., KORNILOWICZ, A., AND RABE, F. A New Export of the Mizar Mathematical Library. In *Intelligent Computer Mathematics* (Cham, 2021), F. Kamareddine and C. Sacerdoti Coen, Eds., Springer International Publishing, p. 205–210.

[33] RUSSELL, B. *Principles of Mathematics*. W. W. Norton & Company, Berlin, 1903.

[34] RUSSLL, B. Recent work in the philosophy of mathematics. *International Monthly* (1901).

[35] SOJAKOVA, K., AND RABE, F. *Translating a Dependently-Typed Logic to First-Order Logic*. Springer Berlin, Heidelberg, 2009, p. 326–341.

[36] STEEN, A. An extensible logic embedding tool for lightweight non-classical reasoning, 2022. arXiv:2203.12352.

[37] STEEN, A., AND BENZMÜLLER, C. Extensional Higher-Order Paramodulation in Leo-III. *Journal of Automated Reasoning 65*, 6 (2021), 775–807.

[38] SUTCLIFFE, G. The TPTP Problem Library and Associated Infrastructure: The FOF and CNF Parts, v3.5.0. *Journal of Automated Reasoning 43*, 4 (2009), 337–362.

[39] SUTCLIFFE, G., BENZMÜLLER, C., BROWN, C., AND THEISS, F. Progress in the development of automated theorem proving for higher-order logic. In *Automated Deduction – CADE-22* (2009), R. Schmidt, Ed., Springer Berlin, Heidelberg, p. 116–130.

[40] SUTCLIFFE, G., AND BENZMÜLLER, C. Automated reasoning in higher-order logic using the tptp thf infrastructure. *Journal of Formalized Reasoning 3* (2010).

[41] VUKMIROVIĆ, P., BENTKAMP, A., BLANCHETTE, J., CRUANES, S., NUMMELIN, V., AND TOURRET, S. Making higher-order superposition work. In *Automated Deduction – CADE 28* (Cham, 2021), A. Platzer and G. Sutcliffe, Eds., Springer International Publishing, p. 415–432.

[42] WHITEHEAD, A., AND RUSSELL, B. *Principia Mathematica*. Cambridge University Press, 1913.

[43] WOLFF, L. Theorem proving in the MMT system, 2022. The Bachelor's thesis is available at https://gl.kwarc.info/supervision/BSc-archive/-/blob/master/2022/wolff_luca.pdf.

Appendix

A Proof of lemma deriving further inference rules for HOL

Proof of Lemma 1:

Regarding (ctxThy), (tpCtx), (typingTp) and (validTyping): We can show these rules easily by induction on the inference rules of HOL. In each step of the induction we show that the assumptions of the rule and these four rules holding on the assumptions of the rule implies that these four rules also hold for the conclusion of the rule. In case of a validity rule this means that we need to show that the conclusion is well-typed, in case of a typing rule we need to show that the type of the term in the typing statement in the conclusion is well-formed, in case of a well-formedness rule for types we need to show that the context of the conclusion is well-formed and in case of a well-formedness rule for contexts we need to show that the theory relative to which the conclusion is stated is well-formed. These properties hold by construction of the inference rules (each inference rules makes whatever assumptions are necessary to guarantee that the inductive step in this proof works).

Regarding ($\equiv$sym) and ($\equiv$trans): We can show both on induction on the two rules that allow showing type equality, namely (congBase) and (cong$\to$). The rule (congBase) only allows showing type equality for identical types (so symmetry is trivial and transitivity reduces to one of the two assumption of the transitivity rule). The rule (cong$\to$) only allows showing type equality for $\to$-types $A \to B$ and $A' \to B'$ based on equal types $A \equiv A'$ and $B \equiv B'$. Now, if the assumption of ($\equiv$sym) is shown using rule (cong$\to$), by induction hypothesis we yield that the type equalities $A \equiv A'$ and $B \equiv B'$ satisfy symmetry and using (cong$\to$) with the swapped type equalities yields $A' \to B' \equiv A \to B$ as desired. Similarly, if $A \to B \equiv A' \to B'$ and $A' \to B' \equiv A'' \to B''$ are both shown using the rule (cong$\to$) we have $A \equiv A'$ and $A' \equiv A''$ and hence by induction hypothesis $A \equiv A''$ and $B \equiv B''$, so rule (cong$\to$) implies $A \to B \equiv A'' \to B''$ as desired.

Regarding (eqTyping), (implTypingL) and (implTypingR): By assumption we have $\Gamma \vdash_T s =_A t$ resp. $\Gamma \vdash_T F \Rightarrow G$. By the rules (ctxThy), (tpCtx), (typingTp) and (validTyping) it follows that the theory T and context Γ must be well-formed. Since the well-formedness of a context with an additional assumption or of a theory with an additional axiom can only be shown by rule (ctxAssume) and (thyAxiom) respectively it follows that axioms in T and assumptions in Γ must be well-typed and that the proof of the axiom or assumption being well-typed must not use that assumption or axiom. We can then prove by induction on derivations that whenever $\Gamma \vdash_T s =_A t$ is concluded in a step then the assumption of the step imply that $\Gamma \vdash_T s : A$ and $\Gamma \vdash_T t : A$ are derivable and whenever $\Gamma \vdash_T F \Rightarrow G$ is concluded in a step $\Gamma \vdash_T F : \mathsf{bool}$ and $\Gamma \vdash_T G : \mathsf{bool}$ are derivable from the assumptions of the step and furthermore whenever we conclude a validity statement, the formula is well-typed and in all conclusions in the derivation all types and contexts are well-formed.

The the claim follows directly from using what we just proved for the step showing the assumption $\Gamma \vdash_T s =_A t$ resp. $\Gamma \vdash_T F \Rightarrow G$.

Regarding (cong:): This follows from the fact that provably equal types in HOL are necessarily identical, so whenever we have $A \equiv A'$ and $t : A$, we already have that $t : A'$ holds by assumption. We can show that provably equal types are identical by induction on the two type-equality rules in HOL namely (congBase) and (cong→).

Regarding (typesUnique): There is no rule allowing to show that a well-typed boolean term has any other type than bool (since we only allow validity but not type equality or typing assumptions). Hence C, C' are not bool and s not of type bool. Continue by induction on the shape of s.

Depending on the shape of s at most one of the two assumption (wlog. (renaming) assume that it is $\Gamma \vdash_T s : C$) can be concluded using one of the rules (const) or (var). The other must be shown using on of the rules (lambda), (appl) (since names of context variables and constants are mutually distinct).

In the case that rule (lambda) was used to show $s : C'$ (with $C' = A \to B'$ and $s = \lambda x : A.\ t$) it follows from the shape of s that also $C = A \to B$ for some type B with $\Gamma,\ x : A \vdash_T t : B$ (as also $s : C$ must be proven using rule (lambda)). By the induction hypothesis it follows that then $\Gamma,\ x : A \vdash_T B \equiv B'$. Since equal types in HOL must be identical (see proof of rule (cong:)), it follows that $\Gamma \vdash_T B \equiv B'$. Rule (cong→) then implies the claim of $\Gamma \vdash_T C \equiv C'$.

In the case that rule (appl) was used to show $s : C'$ (with $C' = B'$ and $s = f\ t$ and $\Gamma \vdash_T f : A' \to B'$ and $\Gamma \vdash_T t : A'$) it follows from the shape of s that also $C = B$ for some type B with $\Gamma \vdash_T f : A \to B$ and $\Gamma \vdash_T t : A$. By the induction hypothesis (applied to f and t) it follows that $\Gamma \vdash_T A \to B \equiv A' \to B'$ and $\Gamma \vdash_T A \equiv A'$. Since equal types in HOL are identical it follows that also $\Gamma \vdash_T B \equiv B'$ holds, so by assumption the conclusion holds.

Regarding (typingWf): The first assumption can only be proven using rule (appl). In the first case the conclusion is an assumption of the rule used to prove $\Gamma \vdash_T f\ t : B$ and must therefore hold.

Regarding (monotonic⊢) and (var⊢): The idea for showing this is observing that adding additional variables or assumptions to the contexts occuring in a derivation will always result in another (equally valid) derivation. This can be shown by induction on derivations. If in a derivation some rule is used to conclude $\Gamma \vdash_T F$ and adding additional variables and assumptions to Γ as well as the contexts of all judgements appearing in those derivations leaves those derivations valid, then we need to show that also the assumptions of the rule are still satisfied. Since we have valid derivations for all assumptions that are judgements themselves, we only need to check assumptions of rules that are not judgements. The only assumptions of this kind that appear in inference rules are assumptions that some variable or context assumption is con-

tained in a context. Adding further assumptions or variables will not make this false if is was true initially.

This proves that (monotonic$\vdash$) and (var$\vdash$) hold.

Regarding (rewriteTyping): This can be seen by applying the substitution $\cdot[^x/_t]$ to the terms in the derivation (but not the variable declaration x in the context itself) of $\Gamma,\ x : B \vdash_T s : A$, using the fact that $\Gamma \vdash_T t : B$ holds instead of rule (var) (that may be used to conclude $\Gamma,\ x : B \vdash_T x : B$ in the original declaration). This leads to a derivation of the conclusion $\Gamma,\ x : A \vdash_T s[^x/_t] : A$. Since x doesn't appear in the derivation, removing the variable declaration $x : A$ from the context leads to a valid derivation of $\Gamma \vdash_T s[^x/_t] : A$ as desired.

Regarding (assTyping):

$\vdash \Gamma\ \mathsf{Ctx}$	by assumption	(12)
$\vdash F$ in Γ	by assumption	(13)
$\Gamma \vdash F$	(assume),(13),(12)	(14)
$\Gamma \vdash F : \mathsf{bool}$	(validTyping),(14)	(15)

Regarding ($\forall$type):

$\Gamma,\ x : A \vdash_T F : \mathsf{bool}$	by assumption	(16)
$\Gamma \vdash_T \lambda x : A.\ F : A \to \mathsf{bool}$	(lambda),(16)	(17)
$\Gamma,\ x : A \vdash_T \mathsf{true} : \mathsf{bool}$	by definition	(18)
$\Gamma \vdash_T \lambda x : A.\ \mathsf{true} : A \to \mathsf{bool}$	(lambda),(18)	(19)
$\Gamma \vdash_T \lambda x : A.\ F =_{\mathsf{bool}} \lambda x : A.\ \mathsf{true} : \mathsf{bool}$	(=type),(17),(19)	(20)
$\Gamma \vdash_T \forall\, x : A.\ F : \mathsf{bool}$	by definition,(20)	(21)

Regarding ($\forall$E):

$\Gamma \vdash_T \forall\, x : A.\ F$	by assumption	(22)
$\Gamma \vdash_T t : A$	by assumption	(23)
$\Gamma \vdash_T \lambda x : A.\ F =_{A \to \mathsf{bool}} \lambda x : A.\ \mathsf{true}$	by definition,(22)	(24)
$\Gamma \vdash_T t =_A t$	(refl),(23)	(25)
$\Gamma \vdash_T \big(\lambda x : A.\ F\big)\ t =_{\mathsf{bool}} \big(\lambda x : A.\ \mathsf{true}\big)\ t$	(congAppl),(24),(25)	(26)
$\Gamma \vdash_T \big(\lambda x : A.\ \mathsf{true}\big)\ t =_{\mathsf{bool}} \big(\lambda x : A.\ F\big)\ t$	(sym),(26)	(27)
$\Gamma \vdash_T \big(\lambda x : A.\ F\big)\ t : \mathsf{bool}$	(eqTyping),(26)	(28)
$\Gamma \vdash_T \big(\lambda x : A.\ \mathsf{true}\big)\ t : \mathsf{bool}$	(eqTyping),(27)	(29)
$\Gamma \vdash_T \big(\lambda x : A.\ F\big)\ t =_{\mathsf{bool}} F[^x/_t]$	(beta),(univE6)	(30)
$\Gamma \vdash_T \big(\lambda x : A.\ \mathsf{true}\big)\ t =_{\mathsf{bool}} \mathsf{true}$	(beta),(univE6)	(31)

$\Gamma \vdash_T F[^x/_t] =_{\mathsf{bool}} \left(\lambda x : A.\ \mathsf{true}\right) t$ (rewrite),(26),(30) (32)
$\Gamma \vdash_T F[^x/_t] =_{\mathsf{bool}} \mathsf{true}$ (rewrite),(32),(31) (33)
$\Gamma \vdash_T \mathsf{true}$ (refl),definition (34)
$\Gamma \vdash_T F[^x/_t]$ (cong$\vdash$),(31),(34) (35)

Regarding (propExt):

$\Gamma, F \vdash_T G$ by assumption (36)
$\Gamma, F \vdash_T G : \mathsf{bool}$ (validTyping),(36) (37)
$\vdash_T \Gamma,\ F\ \mathsf{Ctx}$ (tpCtx),(46) (38)
$\Gamma \vdash_T F : \mathsf{bool}$ (assTyping),(38) (39)
$\Gamma, G \vdash_T F$ by assumption (40)
$\Gamma, G \vdash_T F : \mathsf{bool}$ (validTyping),(36) (41)
$\vdash_T \Gamma,\ G\ \mathsf{Ctx}$ (tpCtx),(47) (42)
$\Gamma \vdash_T G : \mathsf{bool}$ (assTyping),(42) (43)
$\Gamma \vdash_T \mathsf{true} : \mathsf{bool}$ by definition (44)
$\Gamma \vdash_T \mathsf{false} : \mathsf{bool}$ by definition (45)
$\Gamma \vdash_T F \Rightarrow G$ ($\Rightarrow$I), (39),(36) (46)
$\Gamma \vdash_T G \Rightarrow F$ ($\Rightarrow$I), (43),(40) (47)
$\Gamma \vdash_T \mathsf{true} =_{\mathsf{bool}} \mathsf{true}$ (refl),(44) (48)
$\Gamma \vdash_T \mathsf{true} \Rightarrow \mathsf{true} : \mathsf{bool}$ ($\Rightarrow$type),(44),(44) (49)
$\Gamma,\ \mathsf{true} \Rightarrow \mathsf{true} \vdash_T \mathsf{true} =_{\mathsf{bool}} \mathsf{true}$ (monotonic$\vdash$),(49), (48) (50)
$\Gamma \vdash_T (\mathsf{true} \Rightarrow \mathsf{true}) \Rightarrow \mathsf{true} =_{\mathsf{bool}} \mathsf{true}$ ($\Rightarrow$I),(49), (50) (51)
$\Gamma,\ \mathsf{true} \Rightarrow \mathsf{true} \vdash_T (\mathsf{true} \Rightarrow \mathsf{true}) \Rightarrow \mathsf{true} =_{\mathsf{bool}} \mathsf{true}$ (monotonic$\vdash$),(49),(51) (52)
$\Gamma \vdash_T (\mathsf{true} \Rightarrow \mathsf{true}) \Rightarrow ((\mathsf{true} \Rightarrow \mathsf{true}) \Rightarrow \mathsf{true} =_{\mathsf{bool}} \mathsf{true})$ ($\Rightarrow$I),(49), (52) (53)

Analogously we can show:

$\Gamma \vdash_T (\mathsf{false} \Rightarrow \mathsf{false}) \Rightarrow ((\mathsf{false} \Rightarrow \mathsf{false}) \Rightarrow \mathsf{false} =_{\mathsf{bool}} \mathsf{false})$ analogous to (53) (54)
$\Gamma \vdash_T \mathsf{true} \Rightarrow \mathsf{false} : \mathsf{bool}$ ($\Rightarrow$type),(44),(45) (55)
$\vdash_T \Gamma,\ \mathsf{true} \Rightarrow \mathsf{false}\ \mathsf{Ctx}$ (ctxAssume),(55) (56)
$\Gamma, \mathsf{true} \Rightarrow \mathsf{false} \vdash_T \mathsf{true} \Rightarrow \mathsf{false} : \mathsf{bool}$ ($\Rightarrow$type),(monotonic$\vdash$),(55),(44),(monotonic$\vdash$),(55),(45) (57)
$\Gamma \vdash_T \mathsf{false} \Rightarrow \mathsf{true} : \mathsf{bool}$ ($\Rightarrow$type),(45),(44) (58)
$\Gamma, \mathsf{false} \Rightarrow \mathsf{true} \vdash_T \mathsf{false} \Rightarrow \mathsf{true} : \mathsf{bool}$ ($\Rightarrow$type),(monotonic$\vdash$),(55),(45),(monotonic$\vdash$),(55),(44) (59)
$\Gamma \vdash_T \mathsf{true}$ (refl),(44) (60)
$\Gamma, \mathsf{true} \Rightarrow \mathsf{false} \vdash_T \mathsf{true} \Rightarrow \mathsf{false}$ (assume),(56) (61)
$\Gamma, \mathsf{true} \Rightarrow \mathsf{false} \vdash_T \mathsf{false}$ ($\Rightarrow$E),(61),(60) (62)
$\Gamma, \mathsf{true} \Rightarrow \mathsf{false} \vdash_T \mathsf{true}$ (monotonic$\vdash$),(55),(60) (63)
$\Gamma, \mathsf{true} \Rightarrow \mathsf{false},\ y : \mathsf{bool} \vdash_T y : \mathsf{bool}$ (var),(ctxVar),(56) (64)

$\Gamma, \mathsf{true} \Rightarrow \mathsf{false} \vdash_T \lambda y : \mathsf{bool}.\ y : \mathsf{bool} \to \mathsf{bool}$ (lambda),(64) (65)

$\Gamma, \mathsf{true} \Rightarrow \mathsf{false} \vdash_T \mathsf{true} : \mathsf{bool}$ by definition (66)

$\Gamma, \mathsf{true} \Rightarrow \mathsf{false} \vdash_T \mathsf{false} : \mathsf{bool}$ by definition (67)

$\Gamma, \mathsf{true} \Rightarrow \mathsf{false} \vdash_T \mathsf{false} =_{\mathsf{bool}} \mathsf{true} : \mathsf{bool}$ (=type),(67),(66) (68)

$\Gamma, \mathsf{true} \Rightarrow \mathsf{false} \vdash_T (\lambda y : \mathsf{bool}.\ y)\ \mathsf{false} =_{\mathsf{bool}} \mathsf{true} : \mathsf{bool}$ (appl),(65),(68) (69)

$\Gamma, \mathsf{true} \Rightarrow \mathsf{false} \vdash_T \mathsf{true} =_{\mathsf{bool}} \mathsf{false} : \mathsf{bool}$ (=type),(66),(67) (70)

$\Gamma, \mathsf{true} \Rightarrow \mathsf{false} \vdash_T (\lambda y : \mathsf{bool}.\ y)\ \mathsf{true} =_{\mathsf{bool}} \mathsf{false} : \mathsf{bool}$ (appl),(65),(70) (71)

$\Gamma, \mathsf{true} \Rightarrow \mathsf{false} \vdash_T (\lambda y : \mathsf{bool}.\ y)\ \mathsf{true} : \mathsf{bool}$ (appl),(65),(66) (72)

$\Gamma, \mathsf{true} \Rightarrow \mathsf{false} \vdash_T (\lambda y : \mathsf{bool}.\ y)\ \mathsf{false} : \mathsf{bool}$ (appl),(65),(67) (73)

$\Gamma, \mathsf{true} \Rightarrow \mathsf{false} \vdash_T (\lambda x : \mathsf{bool}.\ x)\ \mathsf{false} =_{\mathsf{bool}} \mathsf{false}$ (beta),(73) (74)

$\Gamma, \mathsf{true} \Rightarrow \mathsf{false} \vdash_T (\lambda y : \mathsf{bool}.\ y)\ \mathsf{false}$ (cong$\vdash$),(74),(62) (75)

$\Gamma, \mathsf{true} \Rightarrow \mathsf{false} \vdash_T (\lambda x : \mathsf{bool}.\ x)\ \mathsf{true} =_{\mathsf{bool}} \mathsf{true}$ (beta),(72) (76)

$\Gamma, \mathsf{true} \Rightarrow \mathsf{false} \vdash_T (\lambda y : \mathsf{bool}.\ y)\ \mathsf{true}$ (cong$\vdash$),(76),(63) (77)

$\Gamma, \mathsf{true} \Rightarrow \mathsf{false} \vdash_T \forall x : \mathsf{bool}.\ (\lambda y : \mathsf{bool}.\ y)\ x$ (boolExt),(75),(77) (78)

$\Gamma, \mathsf{true} \Rightarrow \mathsf{false} \vdash_T (\lambda y : \mathsf{bool}.\ y)\ (\mathsf{true} =_{\mathsf{bool}} \mathsf{false})$ ($\forall$E),(78),(70) (79)

$\Gamma, \mathsf{true} \Rightarrow \mathsf{false} \vdash_T (\lambda y : \mathsf{bool}.\ y)\ (\mathsf{false} =_{\mathsf{bool}} \mathsf{true})$ ($\forall$E),(78),(68) (80)

$\Gamma, \mathsf{true} \Rightarrow \mathsf{false} \vdash_T (\lambda y : \mathsf{bool}.\ y)\ (\mathsf{true} =_{\mathsf{bool}} \mathsf{false}) =_{\mathsf{bool}} (\mathsf{true} =_{\mathsf{bool}} \mathsf{false})$ (beta),(71) (81)

$\Gamma, \mathsf{true} \Rightarrow \mathsf{false} \vdash_T (\lambda y : \mathsf{bool}.\ y)\ (\mathsf{false} =_{\mathsf{bool}} \mathsf{true}) =_{\mathsf{bool}} (\mathsf{false} =_{\mathsf{bool}} \mathsf{true})$ (beta),(69) (82)

$\Gamma, \mathsf{true} \Rightarrow \mathsf{false} \vdash_T (\mathsf{true} =_{\mathsf{bool}} \mathsf{false}) =_{\mathsf{bool}} (\lambda y : \mathsf{bool}.\ y)\ (\mathsf{true} =_{\mathsf{bool}} \mathsf{false})$ (sym),(81) (83)

$\Gamma, \mathsf{true} \Rightarrow \mathsf{false} \vdash_T (\mathsf{false} =_{\mathsf{bool}} \mathsf{true}) =_{\mathsf{bool}} (\lambda y : \mathsf{bool}.\ y)\ (\mathsf{false} =_{\mathsf{bool}} \mathsf{true})$ (sym),(82) (84)

$\Gamma, \mathsf{true} \Rightarrow \mathsf{false} \vdash_T \mathsf{true} =_{\mathsf{bool}} \mathsf{false}$ (cong$\vdash$),(83),(79) (85)

$\Gamma, \mathsf{true} \Rightarrow \mathsf{false} \vdash_T \mathsf{false} =_{\mathsf{bool}} \mathsf{true}$ (cong$\vdash$),(84),(80) (86)

$\Gamma, \mathsf{true} \Rightarrow \mathsf{false}, \mathsf{false} \Rightarrow \mathsf{true} \vdash_T \mathsf{true} =_{\mathsf{bool}} \mathsf{false}$ (monotonic$\vdash$),(59),(85) (87)

$\Gamma, \mathsf{true} \Rightarrow \mathsf{false} \vdash_T (\mathsf{false} \Rightarrow \mathsf{true}) \Rightarrow (\mathsf{true} =_{\mathsf{bool}} \mathsf{false})$ ($\Rightarrow$I),(57),(87) (88)

$\Gamma \vdash_T (\mathsf{true} \Rightarrow \mathsf{false}) \Rightarrow (\mathsf{false} =_{\mathsf{bool}} \mathsf{true})$ ($\Rightarrow$I),(58),(86) (89)

$\Gamma \vdash_T (\mathsf{true} \Rightarrow \mathsf{false}) \Rightarrow ((\mathsf{false} \Rightarrow \mathsf{true}) \Rightarrow \mathsf{true} =_{\mathsf{bool}} \mathsf{false})$ ($\Rightarrow$I),(55),(88) (90)

$\Gamma, \mathsf{false} \Rightarrow \mathsf{true} \vdash_T (\mathsf{true} \Rightarrow \mathsf{false}) \Rightarrow (\mathsf{true} =_{\mathsf{bool}} \mathsf{false})$ (monotonic$\vdash$),(59),(89) (91)

$\Gamma \vdash_T (\mathsf{false} \Rightarrow \mathsf{true}) \Rightarrow ((\mathsf{true} \Rightarrow \mathsf{false}) \Rightarrow \mathsf{false} =_{\mathsf{bool}} \mathsf{true})$ ($\Rightarrow$I),(59),(91) (92)

$\Gamma \vdash_T \forall y : \mathsf{bool}.\ (\mathsf{true} \Rightarrow y) \Rightarrow ((y \Rightarrow \mathsf{true}) \Rightarrow \mathsf{true} =_{\mathsf{bool}} y)$ (boolExt),(53),(cong$\vdash$),(beta),(90) (93)

$\Gamma \vdash_T \forall y : \mathsf{bool}.\ (\mathsf{false} \Rightarrow y) \Rightarrow ((y \Rightarrow \mathsf{false}) \Rightarrow \mathsf{false} =_{\mathsf{bool}} y)$ (boolExt),(92),(cong$\vdash$),(beta),(54) (94)

Showing that these terms are all well-typed:

$\Gamma, x : \mathsf{bool}, y : \mathsf{bool} \vdash_T x : \mathsf{bool}$ (var),(ctxAssume),(ctxAssume) (95)

$\Gamma, x : \mathsf{bool}, y : \mathsf{bool} \vdash_T y : \mathsf{bool}$ (var),(ctxAssume),(ctxAssume) (96)

$\Gamma, x : \mathsf{bool}, y : \mathsf{bool} \vdash_T (x \Rightarrow y) : \mathsf{bool}$ (⇒type),(95),(96) (97)

$\Gamma, x : \mathsf{bool}, y : \mathsf{bool} \vdash_T (y \Rightarrow x) : \mathsf{bool}$ (⇒type),(96),(95) (98)

$\Gamma, x : \mathsf{bool}, y : \mathsf{bool} \vdash_T (x =_{\mathsf{bool}} y) : \mathsf{bool}$ (=type),(95),(96) (99)

$\Gamma, x : \mathsf{bool}, y : \mathsf{bool} \vdash_T (y \Rightarrow x) \Rightarrow (x =_{\mathsf{bool}} y) : \mathsf{bool}$ (⇒type),(98),(99) (100)

$\Gamma, x : \mathsf{bool}, y : \mathsf{bool} \vdash_T (x \Rightarrow y) \Rightarrow ((y \Rightarrow x) \Rightarrow x =_{\mathsf{bool}} y) : \mathsf{bool}$ (⇒type),(97),(100) (101)

$\Gamma, x : \mathsf{bool} \vdash_T \forall y : \mathsf{bool}. (x \Rightarrow y) \Rightarrow ((y \Rightarrow x) \Rightarrow x =_{\mathsf{bool}} y) : \mathsf{bool}$ (∀type),(101) (102)

$\Gamma \vdash_T \lambda x : \mathsf{bool}. \forall y : \mathsf{bool}. (x \Rightarrow y) \Rightarrow ((y \Rightarrow x) \Rightarrow x =_{\mathsf{bool}} y) : \mathsf{bool} \to \mathsf{bool}$ (lambda),(102) (103)

$\Gamma \vdash_T (\lambda x : \mathsf{bool}. \forall y : \mathsf{bool}. (x \Rightarrow y) \Rightarrow ((y \Rightarrow x) \Rightarrow x =_{\mathsf{bool}} y))\ \mathsf{true} : \mathsf{bool}$ (appl),(103),(44) (104)

$\Gamma \vdash_T (\lambda x : \mathsf{bool}. \forall y : \mathsf{bool}. (x \Rightarrow y) \Rightarrow ((y \Rightarrow x) \Rightarrow x =_{\mathsf{bool}} y))\ \mathsf{false} : \mathsf{bool}$ (appl),(103),(45) (105)

Beta reduce and conclude claim:

$\Gamma \vdash_T (\lambda x : \mathsf{bool}. \forall y : \mathsf{bool}. (x \Rightarrow y) \Rightarrow ((y \Rightarrow x) \Rightarrow x =_{\mathsf{bool}} y))\ \mathsf{true}$
$=_{\mathsf{bool}} \forall y : \mathsf{bool}. (\mathsf{true} \Rightarrow y) \Rightarrow ((y \Rightarrow \mathsf{true}) \Rightarrow \mathsf{true} =_{\mathsf{bool}} y)$ (beta),(104) (106)

$\Gamma \vdash_T (\lambda x : \mathsf{bool}. \forall y : \mathsf{bool}. (x \Rightarrow y) \Rightarrow ((y \Rightarrow x) \Rightarrow x =_{\mathsf{bool}} y))\ \mathsf{false}$
$=_{\mathsf{bool}} \forall y : \mathsf{bool}. (\mathsf{false} \Rightarrow y) \Rightarrow ((y \Rightarrow \mathsf{false}) \Rightarrow \mathsf{false} =_{\mathsf{bool}} y)$ (beta),(105) (107)

$\Gamma \vdash_T (\lambda x : \mathsf{bool}. \forall y : \mathsf{bool}. (x \Rightarrow y) \Rightarrow ((y \Rightarrow x) \Rightarrow x =_{\mathsf{bool}} y))\ \mathsf{true}$ (cong$\vdash$),(106),(93) (108)

$\Gamma \vdash_T (\lambda x : \mathsf{bool}. \forall y : \mathsf{bool}. (x \Rightarrow y) \Rightarrow ((y \Rightarrow x) \Rightarrow x =_{\mathsf{bool}} y))\ \mathsf{false}$ (cong$\vdash$),(107),(94) (109)

$\Gamma \vdash_T \forall x : \mathsf{bool}. (\lambda x : \mathsf{bool}. \forall y : \mathsf{bool}. (x \Rightarrow y) \Rightarrow ((y \Rightarrow x) \Rightarrow x =_{\mathsf{bool}} y))\ x$ (boolExt),(108),(109) (110)

$\Gamma \vdash_T \lambda x : \mathsf{bool}. \forall y : \mathsf{bool}. (x \Rightarrow y) \Rightarrow ((y \Rightarrow x) \Rightarrow x =_{\mathsf{bool}} y)\ F$ (∀E),(110),(39) (111)

$\Gamma \vdash_T \lambda x : \mathsf{bool}. \forall y : \mathsf{bool}. (x \Rightarrow y) \Rightarrow ((y \Rightarrow x) \Rightarrow x =_{\mathsf{bool}} y)\ F : \mathsf{bool}$ (validTyping),(111) (112)

$\Gamma \vdash_T \forall y : \mathsf{bool}. (F \Rightarrow y) \Rightarrow ((y \Rightarrow F) \Rightarrow F =_{\mathsf{bool}} y) =_{\mathsf{bool}}$
$\lambda x : \mathsf{bool}. \forall y : \mathsf{bool}. (x \Rightarrow y) \Rightarrow ((y \Rightarrow x) \Rightarrow x =_{\mathsf{bool}} y)\ F$ (beta),(112) (113)

$\Gamma \vdash_T \forall y : \mathsf{bool}. (F \Rightarrow y) \Rightarrow ((y \Rightarrow F) \Rightarrow F =_{\mathsf{bool}} y)$ (cong$\vdash$),(113),(111) (114)

$\Gamma \vdash_T (F \Rightarrow G) \Rightarrow ((G \Rightarrow F) \Rightarrow F =_{\mathsf{bool}} G$ (∀E),(114),(43) (115)

$\Gamma \vdash_T (G \Rightarrow F) \Rightarrow F =_{\mathsf{bool}} G$ (⇒E), (115), (46) (116)

$\Gamma \vdash_T F =_{\mathsf{bool}} G$ (⇒E), (116), (47) (117)

Regarding (∀I): Note that while this proof uses Lemma 2, this is not a problem since we only use those of the rules in this lemma that are shown in the above cases which don't assume (∀I) or Lemma 2.

$\Gamma, x : A \vdash_T F$ by assumption (118)

$\Gamma, x : A \vdash_T F =_{\mathsf{bool}} \mathsf{true}$ lemma 2,(118) (119)

$\Gamma \vdash_T A \equiv A$ (congBase) (120)

$\Gamma \vdash_T \lambda x : A. F =_{A \to \mathsf{bool}} \lambda x : A. \mathsf{true}$ (congλ),(119),(120) (121)

$\Gamma \vdash_T \forall x : A. F$ by definition,(121) (122)

Regarding (applType):

$\Gamma \vdash_T t : A$ by assumption (123)

$\Gamma \vdash_T f\ t : B$ by assumption (124)

Since typing of a function application can only be proven by rule (appl), the assumptions of the rule must be satisfied, so we yield:

$\Gamma \vdash_T f : A' \to B$ see above (125)
$\Gamma \vdash_T t : A'$ see above (126)
$\Gamma \vdash_T A \equiv A'$ (typesUnique),(126) (127)
$\Gamma \vdash_T A' \equiv A$ ($\equiv$sym),(127) (128)
$\Gamma \vdash_T A' \to B \equiv A \to B$ (cong$\to$),(128) (129)
$\Gamma \vdash_T f =_{A' \to B} f$ (refl),(125) (130)
$\Gamma \vdash_T f : A \to B$ (cong:),(130),(129),(125) (131)

Regarding (trans):

$\Gamma \vdash_T s =_A t$ by assumption (132)
$\Gamma \vdash_T s =_A t : \mathsf{bool}$ (validTyping),(132) (133)
$\Gamma \vdash_T t =_A u$ by assumption (134)
$\Gamma, s =_A t \vdash_T s =_A t$ (assume),(ctxAssume),(133) (135)
$\Gamma, s =_A t, t =_A t \vdash_T s =_A t$ (monotonic$\vdash$),(133), (135) (136)
$\Gamma \vdash_T t =_A s$ (sym),(132) (137)
$\Gamma \vdash_T t : A$ (eqTyping),(137) (138)
$\Gamma \vdash_T A\ \mathsf{tp}$ (typingTp),(138) (139)
$\Gamma \vdash_T u =_A t$ (sym),(134) (140)
$\Gamma \vdash_T u : A$ (eqTyping),(140) (141)
$\Gamma \vdash_T t =_A t : \mathsf{bool}$ (=type),(138),(138) (142)
$\Gamma, s =_A t \vdash_T t =_A t : \mathsf{bool}$ (monotonic$\vdash$),(133), (142) (143)
$\Gamma \vdash_T s : A$ (eqTyping),(132) (144)
$\Gamma, s =_A t \vdash_T t =_A t \Rightarrow s =_A t$ ($\Rightarrow$I),(143) (136) (145)
$\Gamma \vdash_T s =_A t \Rightarrow t =_A t \Rightarrow s =_A t$ ($\Rightarrow$I),(133), (145) (146)
$\Gamma, z : A \vdash_T s : A$ (var$\vdash$),(139),(144) (147)
$\vdash_T \Gamma, z : A\ \mathsf{Ctx}$ (ctxVar),(tpCtx),(144),(139) (148)
$\Gamma, z : A \vdash_T t : A$ (var$\vdash$),(139),(138) (149)
$\Gamma, z : A \vdash_T z : A$ (var),(148) (150)
$\Gamma, z : A \vdash_T t =_A z : \mathsf{bool}$ ($\Rightarrow$I),(149),(150) (151)
$\Gamma, z : A \vdash_T s =_A z : \mathsf{bool}$ ($\Rightarrow$I),(147),(150) (152)
$\Gamma, z : A \vdash_T (t =_A z) \Rightarrow (s =_A z) : \mathsf{bool}$ ($\Rightarrow$I),(151),(152) (153)
$\Gamma, z : A \vdash_T s =_A t : \mathsf{bool}$ ($\Rightarrow$I),(147),(149) (154)
$\Gamma, z : A \vdash_T (s =_A t) \Rightarrow ((t =_A z) \Rightarrow (s =_A z)) : \mathsf{bool}$ ($\Rightarrow$type),(154),(153) (155)
$\Gamma \vdash_T \lambda z : A.\ s =_A t \Rightarrow t =_A z \Rightarrow s =_A z : A \to \mathsf{bool}$ (lambda),(155) (156)
$\Gamma \vdash_T (\lambda z : A.\ s =_A t \Rightarrow t =_A z \Rightarrow s =_A z)\ t : \mathsf{bool}$ (appl),(156),(138) (157)

$\Gamma \vdash_T (\lambda z : A.\ s =_A t \Rightarrow t =_A z \Rightarrow s =_A z)\ t$

$\quad =_{\mathsf{bool}} s =_A t \Rightarrow t =_A t \Rightarrow s =_A t$ (beta),(157) (158)

$\Gamma \vdash_T (\lambda z : A.\ s =_A t \Rightarrow t =_A z \Rightarrow s =_A z)\ t$ (cong$\vdash$), (146), (158) (159)

$\Gamma \vdash_T (\lambda z : A.\ s =_A t \Rightarrow t =_A z \Rightarrow s =_A z)\ t$

$\quad =_{\mathsf{bool}} (\lambda z : A.\ s =_A t \Rightarrow t =_A z \Rightarrow s =_A z)\ u$ (congAppl), (132) (160)

$\Gamma \vdash_T (\lambda z : A.\ s =_A t \Rightarrow t =_A z \Rightarrow s =_A z)\ u$

$\quad =_{\mathsf{bool}} (\lambda z : A.\ s =_A t \Rightarrow t =_A z \Rightarrow s =_A z)\ t$ (sym), (160) (161)

$\Gamma \vdash_T (\lambda z : A.\ s =_A t \Rightarrow t =_A z \Rightarrow s =_A z)\ u$ (cong$\vdash$), (161), (159) (162)

$\Gamma \vdash_T (\lambda z : A.\ s =_A t \Rightarrow t =_A z \Rightarrow s =_A z)\ t : \mathsf{bool}$ (appl),(156),(141) (163)

$\Gamma \vdash_T (\lambda z : A.\ s =_A t \Rightarrow t =_A z \Rightarrow s =_A z)\ u$

$\quad =_{\mathsf{bool}} s =_A t \Rightarrow t =_A u \Rightarrow s =_A u$ (beta),(163) (164)

$\Gamma \vdash_T s =_A t \Rightarrow t =_A u \Rightarrow s =_A u$ (cong$\vdash$),(164), (162) (165)

$\Gamma \vdash_T t =_A u \Rightarrow s =_A u$ ($\Rightarrow$E),(165),(132) (166)

$\Gamma \vdash_T s =_A u$ ($\Rightarrow$E),(166),(134) (167)

Regarding (extensionality):

$\Gamma, x : A \vdash_T f\ x =_B f'\ x$ by assumption (168)

$\Gamma, x : A \vdash_T f\ x : B$ (eqTyping),(168) (169)

$\Gamma, x : A \vdash_T f'\ x =_B f\ x$ (sym),(168) (170)

$\Gamma, x : A \vdash_T f'\ x : B$ (eqTyping),(170) (171)

$\Gamma, x : A \vdash_T x : A$ (var),(tpCtx),(169) (172)

$\Gamma, x : A \vdash_T f : A \to B$ (applType),(172),(169) (173)

$\Gamma, x : A \vdash_T f' : A \to B$ (applType),(172),(171) (174)

$\Gamma \vdash_T f =_{A \to B} \lambda x : A.\ f\ x$ (eta),(173) (175)

$\Gamma \vdash_T \lambda x : A.\ f\ x =_{A \to B} \lambda x : A.\ f'\ x$ (congλ), (168) (176)

$\Gamma \vdash_T f =_{A \to B} \lambda x : A.\ f'\ x$ (trans), (175), (176) (177)

$\Gamma \vdash_T f' =_{A \to B} \lambda x : A.\ f'\ x$ (eta),(174) (178)

$\Gamma \vdash_T \lambda x : A.\ f'\ x =_{A \to B} f'$ (sym),(178) (179)

$\Gamma \vdash_T f =_{A \to B} f'$ (trans), (177), (179) (180)

Regarding ($\forall$cong):

$\Gamma \vdash_T A \equiv A'$ by assumption (181)

By induction on derivations, it follows that A, A' well-typed

$\Gamma, x : A \vdash_T F =_{\mathsf{bool}} F'$ by assumption (182)

$\Gamma \vdash_T \forall x : A.\ F : \mathsf{bool}$ (=type),(lambda),definition,(lambda),(eqTyping),(182) (183)

$\Gamma,\ \forall x : A.\ F \vdash_T \forall x : A.\ F$ (assume),(ctxAssume),(183) (184)

$\Gamma,\ \forall x : A.\ F,\ y : A' \vdash_T \forall x : A.\ F$ (var$\vdash$),(ctxVar),(184) (185)

$\Gamma,\ \forall x : A.\ F,\ y : A',\ x' : A \vdash_T \forall x : A.\ F$ (var$\vdash$),(ctxVar),(185) (186)

$$\Gamma,\ \forall x : A.\ F,\ y : A',\ x' : A \vdash_T x' : A \quad \text{(var),(ctxVar)} \quad (187)$$
$$\Gamma,\ \forall x : A.\ F,\ y : A',\ x : A \vdash_T F \quad \text{(}\forall\text{E),(186),(187)} \quad (188)$$
$$\Gamma,\ \forall x : A.\ F, x : A \vdash_T F =_{\mathsf{bool}} F' \quad \text{(monotonic}\vdash\text{),(182)} \quad (189)$$
$$\Gamma,\ \forall x : A.\ F,\ y : A',\ x : A \vdash_T F =_{\mathsf{bool}} F' \quad \text{(var}\vdash\text{),(tpCtx),(187),(189)} \quad (190)$$
$$\Gamma,\ \forall x : A.\ F,\ y : A',\ x : A \vdash_T F' \quad \text{(cong}\vdash\text{),(190),(188)} \quad (191)$$
$$\Gamma,\ \forall x : A.\ F,\ y : A' \vdash_T \forall x : A.\ F' \quad \text{(}\forall\text{I),(191)} \quad (192)$$

Now we will prove that $\Gamma,\ \forall x : A.\ F,\ y : A' \vdash_T y : A$

$$\Gamma,\ \forall x : A.\ F,\ y : A' \vdash_T y : A' \quad \text{(var),(ctxVar)} \quad (193)$$
$$\Gamma,\ \forall x : A.\ F,\ y : A' \vdash_T y =_{A'} y \quad \text{(refl),(193)} \quad (194)$$
$$\Gamma,\ \forall x : A.\ F,\ y : A' \vdash_T A \equiv A' \quad \text{(var}\vdash\text{),(monotonic}\vdash\text{),(ctxAssume),(183),(181)} \quad (195)$$
$$\Gamma,\ \forall x : A.\ F,\ y : A' \vdash_T A' \equiv A \quad (\equiv\text{sym}) \quad (196)$$
$$\Gamma,\ \forall x : A.\ F,\ y : A' \vdash_T y : A \quad \text{(cong:),(194),(196),(193)} \quad (197)$$

Substitute y for x in (192) and move y from context into a $\forall$ binder.

$$\Gamma,\ \forall x : A.\ F,\ y : A' \vdash_T F'[{}^{x}/_{y}] \quad \text{(}\forall\text{E),(192),(197)} \quad (198)$$
$$\Gamma,\ \forall x : A.\ F, x : A' \vdash_T F' \quad \text{renaming } y \text{ to } x \quad (199)$$
$$\Gamma,\ \forall x : A.\ F, x : A' \vdash_T x : A' \quad \text{(var),(tpCtx),(197)} \quad (200)$$
$$\Gamma,\ \forall x : A.\ F \vdash_T \forall x : A'.\ F' \quad \text{(}\forall\text{I),(199),(200)} \quad (201)$$

Since term and type equality are both symmetric (and we can use the same trick as above to show that variables of type A' are also of type A and vice versa), we can prove the following formula analogously:

$$\Gamma,\ \forall x : A'.\ F' \vdash_T \forall x : A.\ F \quad \text{analogously} \quad (202)$$
$$\Gamma \vdash_T \forall x : A.\ F =_{\mathsf{bool}} \forall x : A.\ F' \quad \text{(propExt),(201),(202)} \quad (203)$$

Regarding (⇒cong):

$$\Gamma \vdash_T F =_{\mathsf{bool}} F' \quad \text{by assumption} \quad (204)$$
$$\Gamma \vdash_T G =_{\mathsf{bool}} G' \quad \text{by assumption} \quad (205)$$

Introduce lambda function in two variables $\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ x \Rightarrow y$ show rule for it using (congAppl) and use (beta) and (trans) to conclude the same about ⇒:

$$\Gamma \vdash_T (\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ x \Rightarrow y) =_{\mathsf{bool}} (\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ x \Rightarrow y) \quad \text{(refl),(lambda),(lambda),(}\Rightarrow\text{type),(var),(var)} \quad (206)$$
$$\Gamma \vdash_T (\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ x \Rightarrow y)\ F =_{\mathsf{bool}} (\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ x \Rightarrow y)\ F' \quad \text{(congAppl),(206),(204)} \quad (207)$$
$$\Gamma \vdash_T (\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ x \Rightarrow y)\ F\ G =_{\mathsf{bool}} (\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ x \Rightarrow y)\ F'\ G' \quad \text{(congAppl),(207),(205)} \quad (208)$$

Now we prove that $\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ x \Rightarrow y$ and its applications are well-typed to allow using the rule (beta) for it:

$$\Gamma, x : \mathsf{bool}, y : \mathsf{bool} \vdash_T x : \mathsf{bool} \quad \text{(var),(ctxVar),(ctxVar)} \quad (209)$$

$\Gamma, x : \mathsf{bool}, y : \mathsf{bool} \vdash_T y : \mathsf{bool}$ (var),(ctxVar),(ctxVar) (210)

$\Gamma, x : \mathsf{bool}, y : \mathsf{bool} \vdash_T x \Rightarrow y : \mathsf{bool}$ (⇒type),(209),(210) (211)

$\Gamma, x : \mathsf{bool} \vdash_T \lambda y : \mathsf{bool}.\ (x \Rightarrow y) : \mathsf{bool} \to \mathsf{bool}$ (lambda),(211) (212)

$\Gamma \vdash_T \lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ x \Rightarrow y : \mathsf{bool} \to \mathsf{bool} \to \mathsf{bool}$ (lambda),(212) (213)

$\Gamma \vdash_T F : \mathsf{bool}$ (eqTyping),(204) (214)

$\Gamma \vdash_T G : \mathsf{bool}$ (eqTyping),(205) (215)

$\Gamma \vdash_T F' =_{\mathsf{bool}} F$ (sym),(204) (216)

$\Gamma \vdash_T G' =_{\mathsf{bool}} G$ (sym),(205) (217)

$\Gamma \vdash_T F' : \mathsf{bool}$ (eqTyping),(216) (218)

$\Gamma \vdash_T G' : \mathsf{bool}$ (eqTyping),(217) (219)

$\Gamma \vdash_T (\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ x \Rightarrow y)\ F : \mathsf{bool} \to \mathsf{bool}$ (appl),(213),(214) (220)

$\Gamma \vdash_T (\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ x \Rightarrow y)\ F' : \mathsf{bool} \to \mathsf{bool}$ (appl),(213),(218) (221)

$\Gamma \vdash_T (\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ x \Rightarrow y)\ F\ G : \mathsf{bool}$ (appl),(220),(215) (222)

$\Gamma \vdash_T (\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ x \Rightarrow y)\ F'\ G' : \mathsf{bool}$ (appl),(221),(219) (223)

Now we can use rule (beta) to show that the applications of $\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ x \Rightarrow y$ to F, G and f', G' respectively are equal to their (beta) reduced versions:

$\Gamma \vdash_T (\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ x \Rightarrow y)\ F =_{\mathsf{bool}\to\mathsf{bool}} (\lambda y : \mathsf{bool}.\ (F \Rightarrow y))$ (beta),(220) (224)

$\Gamma \vdash_T (\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ x \Rightarrow y)\ F' =_{\mathsf{bool}\to\mathsf{bool}} (\lambda y : \mathsf{bool}.\ (F' \Rightarrow y))$ (beta),(221) (225)

And again:

$\Gamma \vdash_T \mathsf{bool} \to \mathsf{bool} \equiv \mathsf{bool} \to \mathsf{bool}$ (congBase) (226)

$\Gamma \vdash_T \lambda y : \mathsf{bool}.\ (F \Rightarrow y) : \mathsf{bool} \to \mathsf{bool}$ (cong:),(224),(226),(220) (227)

$\Gamma \vdash_T \lambda y : \mathsf{bool}.\ (F' \Rightarrow y) : \mathsf{bool} \to \mathsf{bool}$ (cong:),(225),(226),(221) (228)

$\Gamma \vdash_T \lambda y : \mathsf{bool}.\ (F \Rightarrow y)\ G : \mathsf{bool}$ (appl),(227),(215) (229)

$\Gamma \vdash_T \lambda y : \mathsf{bool}.\ (F' \Rightarrow y)\ G' : \mathsf{bool}$ (appl),(228),(219) (230)

$\Gamma \vdash_T (\lambda y : \mathsf{bool}.\ (F \Rightarrow y))\ G =_{\mathsf{bool}} ((F \Rightarrow G))$ (beta),(222) (231)

$\Gamma \vdash_T (\lambda y : \mathsf{bool}.\ (F' \Rightarrow y))\ G' =_{\mathsf{bool}} ((F' \Rightarrow G'))$ (beta),(223) (232)

$\Gamma \vdash_T G =_{\mathsf{bool}} G$ (refl),(215) (233)

$\Gamma \vdash_T G' =_{\mathsf{bool}} G'$ (refl),(219) (234)

$\Gamma \vdash_T (\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ x \Rightarrow y)\ F\ G =_{\mathsf{bool}} \lambda y : \mathsf{bool}.\ (F \Rightarrow y)\ G$ (congAppl),(233),(224) (235)

$\Gamma \vdash_T (\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ x \Rightarrow y)\ F'\ G' =_{\mathsf{bool}} \lambda y : \mathsf{bool}.\ (F' \Rightarrow y)\ G'$ (congAppl),(234),(225) (236)

$\Gamma \vdash_T (\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ x \Rightarrow y)\ F\ G =_{\mathsf{bool}} (F \Rightarrow G)$ (trans),(235),(231) (237)

$\Gamma \vdash_T (\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ x \Rightarrow y)\ F'\ G' =_{\mathsf{bool}} (F' \Rightarrow G')$ (trans),(236),(232) (238)

$\Gamma \vdash_T (F \Rightarrow G) =_{\mathsf{bool}} (\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ x \Rightarrow y)\ F\ G$ (sym),(237) (239)

$\Gamma \vdash_T (F \Rightarrow G) =_{\mathsf{bool}} (\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ x \Rightarrow y)\ F'\ G'$ (trans),(239),(208) (240)

$\Gamma \vdash_T (F \Rightarrow F') =_{\mathsf{bool}} (G \Rightarrow G')$ (trans),(240),(238) (241)

Regarding ($\vdash$ cong):

$\Gamma \vdash_T F =_{\mathsf{bool}} F'$	by assumption	(242)
$\Gamma \vdash_T F$	by assumption	(243)
$\Gamma \vdash_T F' =_{\mathsf{bool}} F$	(sym),(242)	(244)
$\Gamma \vdash_T F'$	(cong$\vdash$),(244),(243)	(245)

Regarding (rewrite):

$\Gamma \vdash_T F[x/t]$	by assumption	(246)
$\Gamma \vdash_T t =_A t'$	by assumtion	(247)
$\Gamma, y : A \vdash_T F[x/y] : \mathsf{bool}$	by assumption	(248)
$\Gamma \vdash_T \lambda y : A.\ F[x/y] : A \to \mathsf{bool}$	(lambda),(248)	(249)
$\Gamma \vdash_T t : A$	(eqTyping),(247)	(250)
$\Gamma \vdash_T (\lambda y : A.\ F[x/y])\ t : \mathsf{bool}$	(appl),(249),(250)	(251)
$\Gamma \vdash_T (\lambda y : A.\ F[x/y])\ t =_{\mathsf{bool}} F[x/t]$	(beta),(251)	(252)
$\Gamma \vdash_T (\lambda y : A.\ F[x/y])\ t$	(cong$\vdash$),(252),(246)	(253)
$\Gamma \vdash_T \lambda y : A.\ F[x/y] =_{A \to \mathsf{bool}} \lambda y : A.\ F[x/y]$	(refl),(249)	(254)
$\Gamma \vdash_T (\lambda y : A.\ F[x/y])\ t =_{\mathsf{bool}} (\lambda y : A.\ F[x/y])\ t'$	(congAppl),(247),(254)	(255)
$\Gamma \vdash_T (\lambda y : A.\ F[x/y])\ t'$	($\vdash$ cong),(255),(253)	(256)
$\Gamma \vdash_T t' : A$	(eqTyping),(sym),(247)	(257)
$\Gamma \vdash_T (\lambda y : A.\ F[x/y])\ t' : \mathsf{bool}$	(appl),(249),(257)	(258)
$\Gamma \vdash_T (\lambda y : A.\ F[x/y])\ t' =_{\mathsf{bool}} F[x/t']$	(beta),(258)	(259)
$\Gamma \vdash_T F[x/t']$	($\vdash$ cong),(259),(256)	(260)

□

B Proof of lemma about equivalence of HOL with Q_0

Proof of Lemma 5. Firstly, let us quickly talk about the inference rules for the other judgments (not validity) and why those rules are reasonable.

However, since usually theories and context and their well-formedness are not formalized (because it is relatively obvious how they should be formalized) we will not discuss the rules for well-formedness of types and theories. Similarly the lookup rules for contexts and theories are obvious and result from us defining the logic more formally than usually done.

The type-equality and typing rules are also not very interesting:

- the type equality rules allow showing equality of types exactly for identical types (this can be shown, once again, by induction on the two type-equality rules)

- the typing rules (=type), (lambda), (appl) and (⇒type) all allow showing a typing statement for a term assuming typing statements for its subterms

The interesting rules are the validity rules and in order to ensure equivalence with classical HOL we mainly have to ensure they are equivalent to e.g. the validity rules (and axiom schemata) of Q_0 (except for the one about the description operator). This focus is further motivated by the fact that we are interested in problems formalized in HOL style, which means that the conjectures we are interested in are validity statements.

Translated into our syntax, the first four of the five axiom schemata of Q_0 are: [4]

$$\begin{aligned}
&p\ \mathsf{true} \wedge p\ \mathsf{false} && =_{\mathsf{bool}} \forall x : \mathsf{bool}.\ p\ x && \text{for } p : \mathsf{bool} \to \mathsf{bool} && \text{(Q1)}\\
&t =_A t' && \Rightarrow f\ t =_B f\ t' && \text{for } f : A \to B, t, t' : A && \text{(Q2)}\\
&f =_{A\to B} g && =_{\mathsf{bool}} \forall x : A.\ f\ x =_B g\ x && \text{for } f, g : A \to B && \text{(Q3)}\\
& && (\lambda x : A.\ s)\ t =_B s[^{x}/_{t}] && \text{for } s : B \text{ and } t : A && \text{(Q4)}
\end{aligned}$$

and additionally we have one inference rule:

$$\frac{\vdash t =_A t' \qquad \vdash F}{F'}\mathrm{R} \quad \text{if } F' \text{ obtained from } F \text{ by replacing a single occurence of } t \text{ by } t' \text{ in } F \tag{261}$$

Let Q_0F denote the fragment of Q_0 without the description operator and without the fifth axiom schema (about the description operator). This fragment has exactly the above four axiom schemata and the above inference rule (R). We claim that in a context without assumptions and relative to a theory without axioms HOL is equivalent to Q_0F. To show this, we need to derive the axiom schemata and inference rule of Q_0F in HOL and conversely show the validity rules of HOL (except for the lookup rules (assume) and (axiom)) in Q_0F. Thus it follows that Q_0F can prove the same validity statements as HOL relative to a theory without axioms and in a context without assumptions.

Firstly, we prove the equivalence of the inference rules for ⇒ with the definition of ⇒ in Q_0:

Lemma 31. *Implication as formalized with the its inference rules is equivalent to implication defined by* $\Rightarrow := \lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ (x \wedge y) =_{\mathsf{bool}} x$ *as in* Q_0.

Proof of Lemma 31. Firstly, observe that by rule (appl) and the observation that the definien $\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ (x \wedge y) =_{\mathsf{bool}} x$ for ⇒ has type $\mathsf{bool} \to \mathsf{bool} \to \mathsf{bool}$ it follows that rule (⇒type) holds for the defined implication. Furthermore, if rule (⇒type) can be used to show that a term $F \Rightarrow G$ is well-typed then by assumption F and G are well-typed booleans and thus rule (appl) yields that the same conclusion for the defined implication without using rule (⇒type).

[4] In the original version of Q_0 introduced in [1] four separate schemata are used instead of the fourth one, but it is also proven in the same book that they are jointly equivalent to this axiom schema, which is more convenient for our purposes.

It remains to show that the rules (⇒I) and (⇒E) are jointly equivalent to the validity of the definition for ⇒ in Q_0. We can show that the defined ⇒ satisfies rule (⇒I) as follows:

$\Gamma,\ F \vdash_T G$	by assumption	(262)
$\Gamma,\ F \vdash_T F$	(assume)	(263)
$\Gamma,\ F \vdash_T F \wedge G$	(∧I),(262),(263)	(264)
$\Gamma,\ F \wedge G \vdash_T F$	(∧El)	(265)
$\Gamma \vdash_T F \wedge G =_{\mathsf{bool}} F$	(propExt),(264),(265)	(266)
$\Gamma \vdash_T \left(\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ x \wedge y =_{\mathsf{bool}} x\right)\ F$ $=_{\mathsf{bool}} \left(\lambda y : \mathsf{bool}.\ F \wedge y =_{\mathsf{bool}} F\right)$	(beta)	(267)
$\Gamma \vdash_T \left(\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ x \wedge y =_{\mathsf{bool}} x\right)\ F\ G$ $=_{\mathsf{bool}} \left(\lambda y : \mathsf{bool}.\ F \wedge y =_{\mathsf{bool}} F\right)\ G$	(congAppl),(267,(refl))	(268)
$\Gamma \vdash_T \left(\lambda y : \mathsf{bool}.\ F \wedge y =_{\mathsf{bool}} F\right)\ G$ $=_{\mathsf{bool}} \left(F \wedge G =_{\mathsf{bool}} F\right)$	(beta)	(269)
$\Gamma \vdash_T \left(\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ x \wedge y =_{\mathsf{bool}} x\right)\ F\ G$ $=_{\mathsf{bool}} \left(F \wedge G =_{\mathsf{bool}} F\right)$	(trans),(268),(269)	(270)
$\Gamma \vdash_T \left(\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ x \wedge y =_{\mathsf{bool}} x\right)\ F\ G$	(cong⊢),(270),(266)	(271)
$\Gamma \vdash_T F \Rightarrow G$	definition of ⇒,(271)	(272)

Secondly, we can show that the defined ⇒ satisfies rule (⇒E) by:

$\Gamma \vdash_T F \Rightarrow G$	by assumption	(273)
$\Gamma \vdash_T F$	by assumption	(274)
$\Gamma \vdash_T \left(\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ x \wedge y =_{\mathsf{bool}} x\right)\ F\ G$	definition of ⇒,(273)	(275)
$\Gamma \vdash_T F \wedge G =_{\mathsf{bool}} F$	(cong⊢),(270),(275)	(276)
$\Gamma \vdash_T F \wedge G$	(cong⊢),(276),(274)	(277)
$\Gamma \vdash_T G$	(∧Er),(277)	(278)

It remains to show that given $F, G : \mathsf{bool}$ and assuming the rules (⇒I) and (⇒E) it follows that $F \Rightarrow G =_{\mathsf{bool}} \left(\lambda x : \mathsf{bool}.\ \lambda y : \mathsf{bool}.\ x \wedge y =_{\mathsf{bool}} x\right)\ F\ G$ holds. By rule (trans) and equation (270), it suffices to show that $F \Rightarrow G =_{\mathsf{bool}} \left(F \wedge G =_{\mathsf{bool}} F\right)$. This can be shown by case distinctions using rule (boolExt). □

Secondly we observe that the different definitions of ∨ and ∃ are equivalent. The equivalence of the definitions for ∨ can be seen by case distinction on the arguments to the disjunction using rule (boolExt). The equivalence of the definitions for the existential quantifier can be shown using rule (propExt) and double negation elimination (which follows from (boolExt)).

We observe that the inference rule (R) of Q_0F is a special case of (rewrite) as proven in Lemma 1, which allows replacing arbitrarily many occurrences of a subterm t inside a formula F by a term t' equal to t. Let us now discuss how we can prove the axiom schemata of Q_0F in HOL.

We can prove the axiom schema Q1 in HOL as follows:

$$\Gamma,\ p\ \mathsf{true} \wedge p\ \mathsf{false} \vdash_T p\ \mathsf{true} \wedge p\ \mathsf{false} \quad \text{(assume)} \quad (279)$$
$$\Gamma,\ p\ \mathsf{true} \wedge p\ \mathsf{false} \vdash_T p\ \mathsf{true} \quad (\wedge\text{El}),(279) \quad (280)$$
$$\Gamma,\ p\ \mathsf{true} \wedge p\ \mathsf{false} \vdash_T p\ \mathsf{false} \quad (\wedge\text{Er}),(536) \quad (281)$$
$$\Gamma,\ p\ \mathsf{true} \wedge p\ \mathsf{false} \vdash_T \forall x : \mathsf{bool}.\ p\ x \quad (\text{boolExt}),(537),(538) \quad (282)$$
$$\Gamma,\ \forall x : \mathsf{bool}.\ p\ x \vdash_T \forall x : \mathsf{bool}.\ p\ x \quad \text{(assume)} \quad (283)$$
$$\Gamma,\ \forall x : \mathsf{bool}.\ p\ x \vdash_T p\ \mathsf{true} \quad (\forall\text{E}),(283) \quad (284)$$
$$\Gamma,\ \forall x : \mathsf{bool}.\ p\ x \vdash_T p\ \mathsf{false} \quad (\forall\text{E}),(283) \quad (285)$$
$$\Gamma,\ \forall x : \mathsf{bool}.\ p\ x \vdash_T p\ \mathsf{true} \wedge p\ \mathsf{false} \quad (\wedge\text{I}),(284),(285) \quad (286)$$
$$\Gamma \vdash_T p\ \mathsf{true} \wedge p\ \mathsf{false} =_{\mathsf{bool}} \forall x : \mathsf{bool}.\ p\ x \quad (\text{propExt}),(282),(286)$$

The axiom schema Q2 can be proven in HOL as follows:

$$\Gamma,\ t =_A t' \vdash_T t =_A t' \quad \text{(assume)} \quad (287)$$
$$\Gamma,\ t =_A t' \vdash_T f =_{A\to B} f \quad \text{(refl)} \quad (288)$$
$$\Gamma,\ t =_A t' \vdash_T f\ t =_B f\ t' \quad (\text{congAppl}),(287),(288) \quad (289)$$
$$\Gamma \vdash_T t =_A t' \Rightarrow f\ t =_B f\ t' \quad (\Rightarrow\text{I}),(289)$$

The axiom schema Q3 can be proven in HOL using rule (propExt) by:

$$\Gamma,\ f =_{A\to B} g,\ x : A \vdash_T f =_{A\to B} g \quad \text{(assume)} \quad (290)$$
$$\Gamma,\ f =_{A\to B} gx : A, \vdash_T x =_A x \quad \text{(refl)} \quad (291)$$
$$\Gamma,\ f =_{A\to B} g,\ x : A \vdash_T f\ x =_A g\ x \quad (\text{congAppl}),(290),(291) \quad (292)$$
$$\Gamma,\ f =_{A\to B} g \vdash_T \forall x : A.\ f\ x =_B g\ x \quad (\forall\text{I}),(292) \quad (293)$$
$$\Gamma,\ \forall x : A.\ f\ x =_B g\ x \vdash_T \forall x : A.\ f\ x =_B g\ x \quad \text{(assume)} \quad (294)$$
$$\Gamma,\ \forall x : A.\ f\ x =_B g\ x,\ y : A \vdash_T \forall x : A.\ f\ x =_B g\ x \quad \text{(assume)} \quad (295)$$
$$\Gamma,\ \forall x : A.\ f\ x =_B g\ x,\ y : A \vdash_T f\ y =_B g\ y \quad (\forall\text{E}),(295) \quad (296)$$
$$\Gamma,\ \forall x : A.\ f\ x =_B g\ x \vdash_T f =_{A\to B} g \quad (\text{extensionality}),(296) \quad (297)$$
$$\Gamma \vdash_T \left(f =_{A\to B} g\right) =_{\mathsf{bool}} \forall x : A.\ f\ x =_B g\ x \quad (\text{propExt}),(293),(297)$$

The axiom schema Q4 follows directly from rule (beta).

It remains to show that we can derive the validity rules of HOL (except for the lookup rules (axiom) and (assume)) from the axiom schemata and inference rule of Q_0F. For this we assume a context and theory without axioms or assumptions. Since variables in contexts in HOL in a validity statement only help for proving the validity of the formula in the statement if the variable occurs in it, we may assume that all variables in the context of a validity statement will occur in the formula whose validity is asserted in the statement. Thus the contexts, will consist of exactly the variable declarations for the free variables of the formula in the statement. Similarly for declarations in the theory. It thus makes sense to omit the context and theory from our statements as it is done in Q_0F.

Lemma 32. *A formula whose validity is shown using one of the inference rules (congλ), (congAppl), (refl), (sym), (beta), (eta), (cong$\vdash$) and (boolExt) in HOL can be derived in Q_0F from the assumptions of the respective rule.*

Proof of Lemma 32. Given the assumptions of rule (boolExt) we can apply rule ($\wedge$I) to conclude $p\ \mathsf{true} \wedge p\ \mathsf{false}$ and then apply rule (R) to conclude $\forall\, x : \mathsf{bool}.\ p\ x$ which is exactly the conclusion of rule (boolExt).

The rules (congλ), (congAppl), (sym) and (cong$\vdash$) all follow directly from rule (R).

The rule (beta) follows directly from axiom schema Q4 and the observation that $(\lambda x : A.\ s)\ t : B$ is only provable if $s : B$ and $t : A$. Thus the assumptions of the rule imply that the axiom schema is applicable, yielding the desired conclusion of $(\lambda x : A.\ s)\ t =_B s[^x/_t]$.

The rules (eta) and (refl) are both derived in Q_0F in [1]. □

□

C Self-contained definition of Translation 2 from DHOL* into HOL*

Translation definition 2 (Translation from DHOL* into HOL*)**.** We define Translation 2 from DHOL into HOL syntax by induction on the Grammar.

The cases for theories and contexts are:

$$\overline{\mathsf{o}} := \mathsf{o} \tag{T1}$$

$$\overline{T,\ a : \Pi x_1 : A_1.\ \ldots \Pi x_n : A_n.\ \mathsf{tp}} := \overline{T},\quad a : \mathsf{tp},\quad a^? : \overline{A_1} \to \ldots \to \overline{A_n} \to a \to \mathsf{bool} \tag{T2}$$

$$\overline{T,\ c : A} := \overline{T},\quad c : \overline{A},\quad A^?\ c \tag{T3}$$

$$\overline{T,\ F} := \overline{T},\ \overline{F} \tag{T4}$$

$$\cdot := \cdot \tag{T5}$$

$$\overline{\Gamma, x : A} := \overline{\Gamma},\ x : \overline{A}, A^?\ x \tag{T6}$$

$$\overline{\Gamma, F} := \overline{\Gamma},\ \overline{F} \tag{T7}$$

The case of $\overline{A}$ and $A^?\ t$ for types A and terms $t : \overline{A}$ are:

$$\overline{(a\ t_1\ \ldots\ t_n)} := a \tag{T8}$$

$$(a\ t_1\ \ldots\ t_n)^?\ t := a^?\ \overline{t_1}\ \ldots\ \overline{t_n}\ t \tag{T9}$$

$$\overline{\Pi x : A.\ B} := \overline{A} \to \overline{B} \tag{T10}$$

$$\forall\, x : \overline{A}.\ A^?\ x \Rightarrow B^?\ \left(f\ x\right) \tag{T11}$$

$$\overline{\mathsf{bool}} := \mathsf{bool} \tag{T12}$$

For $t^{\beta\eta}$ not the application of a function (from either context or theory) with (eventual) return type bool to all its arguments, we treat $\mathsf{bool}^?\ t$ as a meta-level abbreviation for:

$$\begin{aligned}
&\mathsf{bool}^?\ (t_1 \Rightarrow t_2) && :=\mathsf{bool}^?\ t_1 \wedge \mathsf{bool}^?\ t_2 && && \text{(T13)}\\
&\mathsf{bool}^?\ (s =_A t) && :=p_A\ s \wedge p_A\ t && A \text{ not function type} && \text{(T14)}\\
&\mathsf{bool}^?\ t && :=\mathsf{bool}^?\ t^{\beta\eta} && t \text{ is beta or eta reduceable} && \text{(T15)}\\
&\mathsf{bool}^?\ x && :=\mathsf{true} && x \text{ variable} && \text{(T16)}\\
&\mathsf{bool}^?\ c && :=\mathsf{true} && \text{else} && \text{(T17)}
\end{aligned}$$

For $t^{\beta\eta}$ being the application $t' := p\ t_1\ \ldots\ t_n$ of an n-ary predicate p (from the theory or context) to all its arguments $t_1, \ldots, t_{p_i}$, it follows that $p =: c_i$ must be a variable or a constructor in the context, scope or theory respectively or that t' is the application of a λ-function to an argument that doesn't have the type quantified over in the λ-function (otherwise t' beta reducible). In the latter case, the term t' itself is ill-typed in HOL and thus we define $\mathsf{bool}^?\ t'$ to be false.

Otherwise we define the meta-level abbreviation $\mathsf{bool}^?\ t' = \mathsf{bool}^?\ \left(p\ t_1\ \ldots\ t_n\right)$ as follows:

$$\mathsf{bool}^?\ c_i\ t_1\ \ldots\ t_{p_i} :=T_{i,1}{}^?\ t_1 \wedge \ldots \wedge \left(T_{i,p_i}[{}^{r_{i,1}}/_{t_1}] \ldots [{}^{t_{i,p_i-1}}/_{t_{p_i-1}}]\right)^?\ t_{p_i}, \tag{298}$$

where

$$\begin{aligned}
&c_1 : \Pi r_{1,1} : T_{1,1}.\ \ldots \Pi r_{1,p_1} : T_{1,p_1}.\ \mathsf{bool},\\
&c_2 : \Pi r_{2,1} : T_{2,1}.\ \ldots \Pi r_{2,p_2} : T_{2,p_2}.\ \mathsf{bool},\\
&\vdots\\
&c_m : \Pi r_{m,1} : T_{m,1}.\ \ldots \Pi r_{m,p_m} : T_{m,p_m}.\ \mathsf{bool}
\end{aligned}$$

are the constructors (and variables) of return type bool in the DHOL* theory, context and scope.

The cases for terms are:

$$\begin{aligned}
\overline{c} &:= c && && \text{(T18)}\\
\overline{x} &:= x && && \text{(T19)}\\
\overline{\lambda x : A.\ t} &:= \lambda x : \overline{A}.\ \overline{t} && && \text{(T20)}\\
\overline{f\ t} &:= \overline{f}\ \overline{t} && && \text{(T21)}\\
\overline{F \Rightarrow G} &:= \overline{F} \Rightarrow \overline{G} && && \text{(T22)}\\
\overline{f =_{\Pi x:A.\ B} g} &:= \forall x : \overline{A}.\ A^?\ x \Rightarrow \mathsf{Relat}_B[\overline{f}\ x =_{\overline{B}} \overline{g}\ x] && && \text{(T23)}\\
\overline{s =_A t} &:= \overline{s} =_{\overline{A}} \overline{t} \wedge A^?\ \overline{s} \wedge A^?\ \overline{t} && \text{otherwise} && \text{(T24)}
\end{aligned}$$

Here the notation $\mathsf{Relat}_A[x =_{\overline{A}} x']$ is used to denote the *relativization* of the equality $x =_{\overline{A}} x'$, i.e. the translation of $x =_A x'$ for x, x' being variables of type A in DHOL, rather than variables of type $\overline{A}$ in HOL. The definition is exactly the same as for Translation 1:

We define the relativization of an equality by induction on the type A:

$$\mathsf{Relat}_A[t =_{\overline{A}} t'] := \mathsf{R}[t] =_{\overline{A}} \mathsf{R}[t]' \wedge A^? \, \mathsf{R}[t] \wedge A^? \, \mathsf{R}[t'] \qquad A \text{ not function type} \tag{T25}$$

$$\mathsf{Relat}_{\Pi x:A.\ B}[f =_{\overline{\Pi x:A.\ B}} f'] := \forall x : \overline{A}.\ A^? \, x \Rightarrow \mathsf{Relat}_B[f\ x =_{\overline{B}} f\ x] \qquad \text{else} \tag{T26}$$

And the abbreviation $\mathsf{R}[s]$ denoting the *relativization* of the term s is defined by:

$$\mathsf{R}[s] := s \qquad \text{if } s = \mathsf{R}[t] \tag{T27}$$

And for s not of the form $\mathsf{R}[t]$:

$$\mathsf{R}[s =_A t] := \mathsf{Relat}_A[\overline{s} =_{\overline{A}} \overline{t}] \tag{T28}$$

$$\mathsf{R}[f\ t] := \mathsf{R}[f]\ \mathsf{R}[t] \tag{T29}$$

$$\mathsf{R}[F \Rightarrow G] := \mathsf{R}[F] \Rightarrow \mathsf{R}[G] \tag{T30}$$

$$\mathsf{R}[\lambda x : A.\ s] := \lambda x : A.\ \mathsf{R}[s] \tag{T31}$$

$$\mathsf{R}[x] := x \tag{T32}$$

$$\mathsf{R}[c] := c \tag{T33}$$

The relativization of a term is well-defined, since its definition recurses only into the definition of the relativization of an equality over a type of smaller arity (for equalities over types of positive arity) and into relativizations of subterms (for all other terms).

D Proof of termwise injectivity of Translation 1

Proof of Lemma 6. We prove this by induction on the arity of the types both equalities are over, in case both terms are equalities and by subinduction on the shape of the two terms otherwise. We observe that terms created using a different top-level production are non-identical and will remain that way in the translation. So we can go over the productions one by one and assuming term-wise injectivity for subterms show injectivity of applying them. Different constants are mapped to different constants and different variables to different variables, so in those cases there is nothing to prove. If two function applications or implications differ in DIHOL then one of the two pairs of corresponding arguments must differ as well. By induction hypothesis so will the images of the terms in that pair. Since function application and implication both commute with the translation, it follows that the images of the function applications or implications also differ. Since the translations of the terms on both sides of an equality also show up in the translation, the same argument also works for two equalities over the same type. Similarly for lambda functions of same type.

Consider now two equalities over different types that get identified by dependency-erasure.

In case of equalities over different non-function types, the typing predicate in the relativizations of the equalities will be different and ensure term-wise-injectivity. For equalities over different function types either the domain type or the codomain type must differ by rule (congΠ). If the

domain types differ then the typing predicates in the relativizations of the universal quantifier in the beginning of the translated equalities will be different. If the codomain types are different then the bodies of those relativized universally quantified formulae that are the translations of the equalities will be the translations of the equalities that we yield by applying the functions on both sides of the equalities to a freshly bound variable of the domain type. The translations of the equalities are only identical if those "inner equalities" are identical. Furthermore, the inner equalities are over types of smaller arity. The remaining claim therefore follows from the induction hypothesis. □

E Proof of Lemma 13 about relativizations of equalities

Proof of Lemma 13. If C is not a function type, there is nothing to prove. If C is a predicate subtype $C'|_p$ of a function type C' both the assumption and the conclusion of the lemma change by the additional conjuncts $\overline{p\ s}$ and $\overline{p\ t}$. By the introduction and elimination rules ((∧I), (∧El), (∧Er)) of conjunction it suffices to prove the claim for the equality $s =_{C'} t$ over C instead. So let us assume that $C := \Pi x : A.\ B$ and prove that the claim holds for arbitrary equalities in HOL by induction on the shape of A and B. If follows:

$$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{s} =_{\overline{A} \to \overline{B}} \overline{t} \quad \text{by assumption} \quad (299)$$

$$\overline{\Gamma} \vdash^H_{\overline{T}} (\Pi x : A.\ B)^? \ \overline{s} \quad \text{by assumption} \quad (300)$$

$$\overline{\Gamma} \vdash^H_{\overline{T}} (\Pi x : A.\ B)^? \ \overline{t} \quad \text{by assumption} \quad (301)$$

$$\overline{\Gamma} \vdash^H_{\overline{T}} \forall x : \overline{A}.\ A^? \ x \Rightarrow B^? \ (\overline{s}\ x) \quad \text{IT10,(300)} \quad (302)$$

$$\overline{\Gamma} \vdash^H_{\overline{T}} \forall x : \overline{A}.\ A^? \ x \Rightarrow B^? \ (\overline{t}\ x) \quad \text{IT10,(301)} \quad (303)$$

$$\overline{\Gamma},\ x : \overline{A},\ A^? \ x \vdash^H_{\overline{T}} \overline{s}\ x =_{\overline{B}} \overline{t}\ x \quad \text{(monotonic}\vdash\text{),(congAppl'),(299)} \quad (304)$$

$$\overline{\Gamma},\ x : \overline{A},\ A^? \ x \vdash^H_{\overline{T}} B^? \ (\overline{s}\ x) \quad \text{(∀E),(var}\vdash\text{),(302)} \quad (305)$$

$$\overline{\Gamma},\ x : \overline{A},\ A^? \ x \vdash^H_{\overline{T}} B^? \ (\overline{t}\ x) \quad \text{(∀E),(var}\vdash\text{),(303)} \quad (306)$$

$$\overline{\Gamma},\ x : \overline{A},\ A^? \ x \vdash^H_{\overline{T}} \mathsf{Relat}_B[\overline{s}\ x =_{\overline{B}} \overline{t}\ x] \wedge B^? \ (\overline{s}\ x) \wedge B^? \ (\overline{t}\ x) \quad (\wedge\text{I}),(304),(\wedge\text{I}),(302),(303) \quad (307)$$

$$\overline{\Gamma},\ x : \overline{A},\ A^? \ x \vdash^H_{\overline{T}} \mathsf{Relat}_B[\overline{s}\ x =_{\overline{B}} \overline{t}\ x] \quad \text{induction hypothesis,(307)} \quad (308)$$

$$\overline{\Gamma} \vdash^H_{\overline{T}} \forall x : \overline{A}.\ A^? \ x \Rightarrow \mathsf{Relat}_B[\overline{s}\ x =_{\overline{B}} \overline{t}\ x] \quad (\forall\text{I}),(\Rightarrow\text{I}),(308) \quad (309)$$

$$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{s =_{\Pi x : A.\ B} t} \quad \text{IT23,(309)}$$

□

F Proof of soundness of Translation 1 from DIHOLP into IHOL

Proof of substitution lemma and soundness. We will prove Theorem 14 by induction on derivations, i.e. by doing inductions over the rules that allow us to derive the formulae to the left of the implications in the lemma and show that we can then use the assumptions to those rules to derive the formulae on the right side.

The induction hypothesis then states that the statements (1)-(9) hold for the judgements occurring as assumptions in the rules.

For the case of each of these judgements, we will also assume soundness w.r.t. the previous judgements. For brevity's sake we will nevertheless write this proof as one big inductive proof rather than many separate ones.

Substitution lemma Since the translation of types commutes with the type productions of the grammar (9) is obvious.

We show (8) by induction on the grammar of DPIHOL. If x is not a free variable in t, then $\overline{t[x/u]} = \overline{t} = \overline{t}[x/\overline{u}]$ and the claim (8) follows by rule (refl). So assume that x is a free variable of t.

If t is a variable, then by assumption (that x is a free variable in t) it follows that $t = x$ and thus $\overline{t[x/u]} = \overline{u} = \overline{t}[x/\overline{u}]$ and the claim follows by rule (refl).

If t is a λ-term $\lambda y : A.\ s$, then by induction hypothesis we have $\overline{\Gamma},\ y : \overline{A} \vdash^H_{\overline{T}} \overline{s[x/u]} =_{\overline{A}} \overline{s}[x/\overline{u}]$, where A is the type of s. By rule (congλ), the claim of $\overline{\Gamma} \vdash^H_{\overline{T}} \overline{\lambda y : A.\ s[x/u]} =_{\overline{B}} \overline{\lambda y : A.\ s}[x/\overline{u}]$ follows.

If t is a function application $f\ s$, then by induction hypothesis we have $\overline{\Gamma} \vdash^H_{\overline{T}} \overline{s[x/u]} =_{\overline{A}} \overline{s}[x/\overline{u}]$ and $\overline{\Gamma} \vdash^H_{\overline{T}} \overline{f[x/u]} =_{\overline{A}\to\overline{B}} \overline{f}[x/\overline{u}]$, where T is the type of s. By rule (congAppl), the claim of $\overline{\Gamma} \vdash^H_{\overline{T}} \overline{(f\ s)[x/u]} =_{\overline{B}} \overline{f\ s}[x/\overline{u}]$ follows.

If t is an equality $s =_A s'$, then by induction hypothesis we have $\overline{\Gamma} \vdash^H_{\overline{T}} \overline{s[x/u]} =_{\overline{A}} \overline{s}[x/\overline{u}]$ and $\overline{\Gamma} \vdash^H_{\overline{T}} \overline{s'[x/u]} =_{\overline{A}} \overline{s'}[x/\overline{u}]$, where A is the type of s and s'. By rule ($=$cong), the claim of $\overline{\Gamma} \vdash^H_{\overline{T}} \overline{(s =_A s')[x/u]} =_{\mathsf{bool}} \left(\overline{s =_A s'}\right)[x/\overline{u}]$ follows.

Well-formedness of theories Well-formedness of DPIHOL theories can be shown using the rules (thyEmpty), (thyType'), (thyConst) and (thyAxiom):

(thyEmpty):

$$\vdash \circ\ \mathsf{Thy} \qquad \text{(thyEmpty)} \qquad (310)$$

$$\vdash^H \overline{\circ}\ \mathsf{Thy} \qquad \text{(thyEmpty)} \qquad (311)$$

$$\vdash^H \circ,\ \mathsf{bool}^?\ :\ \mathsf{bool} \to \mathsf{bool}\ \mathsf{Thy} \qquad \text{(thyConst)} \qquad (312)$$

$$\vdash^{H}_{\circ,\ \mathsf{bool}^?:\mathsf{bool}\to\mathsf{bool}} \mathsf{bool}^?\ \mathsf{true} \qquad \text{(appl)} \qquad (313)$$

$$\vdash^{H} \circ,\ \mathsf{bool}^?:\mathsf{bool}\to\mathsf{bool}\ \mathsf{Thy} \qquad \text{(thyAxiom),(313)} \qquad (314)$$

$$\vdash^{H}_{\circ,\ \mathsf{bool}^?:\mathsf{bool}\to\mathsf{bool},\ \mathsf{bool}^?\ \mathsf{true}} \mathsf{bool}^?\ \mathsf{false} \qquad \text{(appl)} \qquad (315)$$

$$\vdash^{H} \circ,\ \mathsf{bool}^?:\mathsf{bool}\to\mathsf{bool},\ \mathsf{bool}^?\ \mathsf{true},\ \mathsf{bool}^?\ \mathsf{false}\ \mathsf{Thy} \qquad \text{(thyAxiom),(315)} \qquad (316)$$

$$\vdash^{H} \overline{\circ}\ \mathsf{Thy} \qquad \text{PT1,(316)}$$

(thyType'):

$$\vdash_T x_1 : A_1, \ldots, x_n : A_n\ \mathsf{Ctx} \qquad \text{by assumption} \qquad (317)$$

$$\vdash^{H}_{\overline{T}} x_1 : \overline{A_1}, {A_1}^? x_1, \ldots, x_n : \overline{A_n}, {A_n}^? x_n\ \mathsf{Ctx} \qquad \text{induction hypothesis,(317)} \qquad (318)$$

$$\vdash^{H} \overline{T}\ \mathsf{Thy} \qquad \text{(ctxThy),(318)} \qquad (319)$$

$$\vdash^{H} \overline{T},\ a : \mathsf{tp}\ \mathsf{Thy} \qquad \text{(thyType),(319)} \qquad (320)$$

$$\vdash^{H} \overline{T,\ a : \Pi x_1 : A_1.\ \ldots \Pi x_n : A_n.\ \mathsf{tp}}\ \mathsf{Thy} \qquad \text{PT2,(320)} \qquad (321)$$

(thyConst):

$$\vdash_T A\ \mathsf{tp} \qquad \text{by assumption} \qquad (322)$$

$$\vdash^{H}_{\overline{T}} \overline{A}\ \mathsf{tp} \qquad \text{induction hypothesis,(322)} \qquad (323)$$

$$\vdash^{H} \overline{T},\ c : \overline{A}\ \mathsf{Thy} \qquad \text{(thyConst),(323)} \qquad (324)$$

$$\vdash^{H} \overline{T,\ c : A}\ \mathsf{Thy} \qquad \text{PT3,(324)} \qquad (325)$$

(thyAxiom):

$$\vdash_T F : \mathsf{bool} \qquad \text{by assumption} \qquad (326)$$

$$\vdash^{H}_{\overline{T}} \overline{F} : \mathsf{bool} \qquad \text{induction hypothesis,(326)} \qquad (327)$$

$$\vdash^{H} \overline{T},\ \overline{F}\ \mathsf{Thy} \qquad \text{(thyAxiom),(327)} \qquad (328)$$

$$\vdash^{H} \overline{T,\ F}\ \mathsf{Thy} \qquad \text{PT4,(328)} \qquad (329)$$

Well-formedness of contexts Well-formedness of contexts can be concluded using the rules (ctxEmpty), (ctxVar) and (ctxAssume):

(ctxEmpty):

$$\vdash T\ \mathsf{Thy} \qquad \text{by assumption} \qquad (330)$$

$$\vdash^{H} \overline{T}\ \mathsf{Thy} \qquad \text{induction hypothesis,(330)} \qquad (331)$$

$$\vdash^{H}_{\overline{T}} .\ \mathsf{Ctx} \qquad \text{(ctxEmpty),(331)} \qquad (332)$$

$$\vdash^{H}_{\overline{T}} :\ \mathsf{Ctx} \qquad \text{PT5,(332)} \qquad (333)$$

(ctxVar):

$\Gamma \vdash_T A\ \mathsf{tp}$	by assumption	(334)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{A}\ \mathsf{tp}$	induction hypothesis,(334)	(335)
$\overline{\Gamma} \vdash^H_{\overline{T}} A^?\ :\ \overline{A} \to \mathsf{bool}$	induction hypothesis,(334)	(336)
$\vdash^H_{\overline{T}} \overline{\Gamma},\ x\ :\ \overline{A}\ \mathsf{Ctx}$	(ctxVar),(335)	(337)
$\overline{\Gamma},\ x\ :\ \overline{A} \vdash^H_{\overline{T}} A^?\ :\ \overline{A} \to \mathsf{bool}$	(var$\vdash$),(335),(336)	(338)
$\overline{\Gamma},\ x\ :\ \overline{A} \vdash^H_{\overline{T}} A^?\ x\ :\ \mathsf{bool}$	(appl),(338),(var)	(339)
$\vdash^H_{\overline{T}} \overline{\Gamma},\ x\ :\ \overline{A},\ A^?\ x\ \mathsf{Ctx}$	(ctxAssume),(339)	(340)
$\vdash^H_{\overline{T}} \overline{\Gamma,\ x\ :\ A}\ \mathsf{Ctx}$	PT6,(340)	(341)

(ctxAssume):

$\Gamma \vdash_T F\ :\ \mathsf{bool}$	by assumption	(342)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{F}\ :\ \mathsf{bool}$	induction hypothesis,(342)	(343)
$\vdash^H_{\overline{T}} \overline{\Gamma},\ \overline{F}\ \mathsf{Ctx}$	(ctxAssume),(343)	(344)
$\vdash^H_{\overline{T}} \overline{\Gamma,\ F}\ \mathsf{Ctx}$	PT7,(344)	(345)

Well-formedness of types Well-formedness of types can be shown in DHOL using the rules (type'), (pi) and ($|_p\,\mathsf{tp}$):

(type'):

$a\ :\ \Pi x_1\ :\ A_1.\ \ldots\ \Pi x_n\ :\ A_n.\ \mathsf{tp}\ \mathsf{in}\ T$	by assumption	(346)
$\Gamma \vdash_T \Gamma\ \mathsf{Ctx}$	by assumption	(347)
$\vdash^H_{\overline{T}} \overline{\Gamma}\ \mathsf{Ctx}$	induction hypothesis,(347)	(348)
$a\ :\ \mathsf{tp}\ \mathsf{in}\ \overline{T}$	PT2,(346)	(349)
$a^?\ :\ a \to \mathsf{bool}\ \mathsf{in}\ \overline{T}$	PT2,(346)	(350)
$\overline{\Gamma} \vdash^H_{\overline{T}} a\ \mathsf{tp}$	(type),(349),(348)	(351)
$\overline{\Gamma} \vdash^H_{\overline{T}} a^?\ :\ a \to \mathsf{bool}$	(const),(350)	(352)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{a\ :\ \Pi x_1\ :\ A_1.\ \ldots\ \Pi x_n\ :\ A_n.\ \mathsf{tp}}$	PT2,(351)	(353)
$\overline{\Gamma} \vdash^H_{\overline{T}} a^?\ :\ \overline{a} \to \mathsf{bool}$	PT16,(352)	(354)

(pi):

$\Gamma \vdash_T A\ \mathsf{tp}$	by assumption	(355)
$\Gamma \vdash_T B\ \mathsf{tp}$	by assumption	(356)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{A}\ \mathsf{tp}$	induction hypothesis,(355)	(357)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{B}\ \mathsf{tp}$	induction hypothesis,(356)	(358)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{A} \to \overline{B}\ \mathsf{tp}$	(arrow),(357),(358)	(359)
$\overline{\Gamma} \vdash^H_{\overline{T}} A^? : \overline{A} \to \mathsf{bool}$	induction hypothesis,(355)	(360)
$\overline{\Gamma} \vdash^H_{\overline{T}} B^? : \overline{B} \to \mathsf{bool}$	induction hypothesis,(356)	(361)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{\Pi x : A.\ B}\ \mathsf{tp}$	PT10,(359)	(362)
$\overline{\Gamma} \vdash^H_{\overline{T}} (\Pi x : A.\ B)^? : (\overline{\Pi x : A.\ B}) \to \mathsf{bool}$	PT11,(360),(361)	(363)

($|_p\,\mathsf{tp}$):

$\Gamma \vdash_T A\ \mathsf{tp}$	by assumption	(364)
$\Gamma \vdash_T p : A \to \mathsf{bool}$	by assumption	(365)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{A}\ \mathsf{tp}$	induction hypothesis,(364)	(366)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{A|_p}\ \mathsf{tp}$	PT8,(366)	(367)
$\overline{\Gamma} \vdash^H_{\overline{T}} A^? : \overline{A} \to \mathsf{bool}$	induction hypothesis,(364)	(368)
$\overline{\Gamma} \vdash^H_{\overline{T}} \left(A|_p\right)^? : \overline{A} \to \mathsf{bool}$	PT9,(368)	(369)

Type-equality Type-equality can be shown using the rules ($|_p\ \equiv$), (congBase'), (congΠ), ($|_p$trivL) and ($|_p$trivR):

($|_p \equiv$):

$\Gamma \vdash_T A \equiv A'$	by assumption	(370)
$\Gamma \vdash_T p =_{\Pi x : A.\ \mathsf{bool}} p'$	by assumption	(371)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{A} \equiv \overline{A'}$	induction hypothesis,(370)	(372)
$\overline{\Gamma} \vdash^H_{\overline{T}} A^? =_{\overline{A} \to \mathsf{bool}} A'^?$	induction hypothesis,(370)	(373)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{p} =_{\overline{A} \to \mathsf{bool}} \overline{p'}$	induction hypothesis,(371)	(374)
$\overline{\Gamma} \vdash^H_{\overline{T}} (A|_p)^? =_{\overline{A} \to \mathsf{bool}} (A|_p)^?$	(refl),PT9,(eqTyping),(373)	(375)
$\overline{\Gamma} \vdash^H_{\overline{T}} (A|_p)^? =_{\overline{A} \to \mathsf{bool}} (A|_{p'})^?$	(rewrite),(375),(374)	(376)

$$\overline{\Gamma} \vdash^H_{\overline{T}} (A|_p)^? =_{\overline{A}\to\mathsf{bool}} (A'|_{p'})^? \quad \text{PT9,(rewrite),(376),(373)} \quad (377)$$

$$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{A|_p} \equiv \overline{A'|_{p'}} \quad \text{PT8,(372)} \quad (378)$$

(congBase'):

$$a : \Pi x_1 : A_1.\ \ldots \Pi x_n : A_n.\ \mathsf{tp} \text{ in } T \quad \text{by assumption} \quad (379)$$

$$\Gamma \vdash_T s_1 =_{A_1} t_1 \quad \text{by assumption} \quad (380)$$

$$\vdots$$

$$\Gamma \vdash_T s_n =_{A_n[^{x_1}/_{t_1}]\ldots[^{x_{n-1}}/_{t_{n-1}}]} t_n \quad \text{by assumption} \quad (381)$$

$$\vdash_T \Gamma\ \mathsf{Ctx} \quad \text{by assumption} \quad (382)$$

$$a : \mathsf{tp} \text{ in } \overline{T} \quad \text{PT2,(379)} \quad (383)$$

$$a^? : \overline{A_1} \to \ldots \to \overline{A_n} \to \overline{a} \to \mathsf{bool} \text{ in } \overline{T} \quad \text{PT2,(379)} \quad (384)$$

$$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{s_1} =_{\overline{A_1}} \overline{t_1} \quad \text{induction hypothesis,(380)} \quad (385)$$

$$\vdots$$

$$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{s_n} =_{\overline{A_n}} \overline{t_n} \quad \text{induction hypothesis,(381)} \quad (386)$$

$$\vdash^H_{\overline{T}} \overline{\Gamma}\ \mathsf{Ctx} \quad \text{induction hypothesis,(382)} \quad (387)$$

$$\overline{\Gamma} \vdash^H_{\overline{T}} a : \mathsf{tp} \quad \text{(type),(383),(387)} \quad (388)$$

$$\overline{\Gamma} \vdash^H_{\overline{T}} a \equiv a \quad \text{(congBase),(388)} \quad (389)$$

$$\overline{\Gamma} \vdash^H_{\overline{T}} a^? =_{\overline{A_1}\to\ldots\to\overline{A_n}\to\overline{a}\to\mathsf{bool}} a^? \quad \text{(refl),(const),(384),(387)} \quad (390)$$

$$\overline{\Gamma} \vdash^H_{\overline{T}} a^?\ \overline{s_1} =_{\overline{A_2}\to\ldots\to\overline{A_n}\to\overline{a}\to\mathsf{bool}} a^?\ \overline{t_1} \quad \text{(congAppl),(385),(390)} \quad (391)$$

$$\vdots$$

$$\overline{\Gamma} \vdash^H_{\overline{T}} a^?\ \overline{s_1}\ \ldots\ \overline{s_n} =_{\overline{a}\to\mathsf{bool}} a^?\ \overline{t_1}\ \ldots\ \overline{t_n} \quad \text{(congAppl),(386),previous line} \quad (392)$$

$$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{a\ s_1\ \ldots\ s_n} \equiv \overline{a\ t_1\ \ldots\ t_n} \quad \text{T8,PT12,(389)} \quad (393)$$

$$\overline{\Gamma} \vdash^H_{\overline{T}} (a\ s_1\ \ldots\ s_n)^? =_{\overline{a\ s_1\ \ldots\ s_n}} (a\ t_1\ \ldots\ t_n)^? \quad \text{PT12,(392)} \quad (394)$$

(congΠ):

$$\Gamma \vdash_T A \equiv A' \quad \text{by assumption} \quad (395)$$

$$\Gamma,\ x : A \vdash_T B \equiv B \quad \text{by assumption} \quad (396)$$

$$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{A} \equiv \overline{A'} \quad \text{induction hypothesis,(395)} \quad (397)$$

$$\overline{\Gamma} \vdash^H_{\overline{T}} A^? =_{\overline{A}\to\mathsf{bool}} A'^? \quad \text{induction hypothesis,(395)} \quad (398)$$

$$\overline{\Gamma},\ x : \overline{A},\ A^?\ x \vdash^H_{\overline{T}} \overline{B} \equiv \overline{B'} \quad \text{induction hypothesis,(396)} \quad (399)$$

$$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{B} \equiv \overline{B'} \quad \equiv \text{ context independent in HOL,(399)} \quad (400)$$

$$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{A} \to \overline{B} \equiv \overline{A'} \to \overline{B'} \qquad \text{(cong}\to\text{),(397),(400)} \qquad (401)$$

$$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{\Pi x : A.\ B} \equiv \overline{\Pi x : A'.\ B'} \qquad \text{PT10,(401)} \qquad (402)$$

$$\overline{\Gamma},\ x : \overline{A},\ A^?\ x \vdash^H_{\overline{T}} B^? =_{\overline{B}\to\mathsf{bool}} B'^? \qquad \text{induction hypothesis,(396)} \qquad (403)$$

$$\overline{\Gamma},\ f : \overline{A} \to \overline{B},\ x : \overline{A} \vdash^H_{\overline{T}} A^?\ x \Rightarrow B^?\ (f\ x)$$
$$=_{\mathsf{bool}} A'^?\ x \Rightarrow B^?\ (f\ x) \qquad \text{(rewrite),(refl),(398)} \qquad (404)$$

$$\overline{\Gamma},\ f : \overline{A} \to \overline{B},\ x : \overline{A} \vdash^H_{\overline{T}} A^?\ x \Rightarrow B^?\ (f\ x)$$
$$=_{\mathsf{bool}} A'^?\ x \Rightarrow B'^?\ (f\ x) \qquad \text{(rewrite),(404),(403)} \qquad (405)$$

$$\overline{\Gamma},\ f : \overline{A} \to \overline{B} \vdash^H_{\overline{T}} \forall\, x : \overline{A}.\ A^?\ x \Rightarrow (B^?(f\ x)) =_{\mathsf{bool}}$$
$$\forall\, x : \overline{A'}.\ A'^?\ x \Rightarrow (B'^?(f\ x))) \qquad (\forall\text{cong}),(397),(405) \qquad (406)$$

$$\overline{\Gamma} \vdash^H_{\overline{T}} (\Pi x : A.\ B)^? =_{\overline{A}\to\mathsf{bool}} \left(\Pi x : A'.\ B'\right)^? \qquad \text{PT15,(cong}\lambda\text{),(406)} \qquad (407)$$

($|_p$trivL):

$$\Gamma \vdash_T A \equiv A' \qquad \text{by assumption} \qquad (408)$$

$$\Gamma \vdash_T p =_{\Pi x : A.\ \mathsf{bool}} \lambda x : A.\ \mathsf{true} \qquad \text{by assumption} \qquad (409)$$

$$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{A} \equiv \overline{A'} \qquad \text{induction hypothesis,(408)} \qquad (410)$$

$$\overline{\Gamma} \vdash^H_{\overline{T}} A^? =_{\Pi x : \overline{A}.\ \mathsf{bool}} A'^? \qquad \text{induction hypothesis,(409)} \qquad (411)$$

$$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{p} =_{\Pi x : \overline{A}.\ \mathsf{bool}} \lambda x : \overline{A}.\ \mathsf{true} \qquad \text{induction hypothesis,(409)} \qquad (412)$$

$$\overline{\Gamma} x : \overline{A},\ \vdash^H_{\overline{T}} \overline{p}\ x =_{\mathsf{bool}} \mathsf{true} \qquad \text{(trans),(congAppl),(refl),(412),(beta)} \qquad (413)$$

$$\overline{\Gamma},\ x : \overline{A} \vdash^H_{\overline{T}} \overline{p}\ x \qquad \text{Lemma 2,(413)} \qquad (414)$$

$$\overline{\Gamma},\ x : \overline{A},\ \vdash^H_{\overline{T}} A^?\ x =_{\mathsf{bool}} A'^?\ x \qquad \text{(congAppl),(refl),(411)} \qquad (415)$$

$$\overline{\Gamma},\ x : \overline{A},\ A^?\ x \wedge \overline{p}\ x \vdash^H_{\overline{T}} A^?\ x \qquad (\wedge\text{El}),(\text{assume}) \qquad (416)$$

$$\overline{\Gamma},\ x : \overline{A},\ A^?\ x \wedge \overline{p}\ x \vdash^H_{\overline{T}} A'^?\ x \qquad (\vdash\text{cong}),(415),(416) \qquad (417)$$

$$\overline{\Gamma},\ x : \overline{A},\ A'^?\ x \vdash^H_{\overline{T}} A^?\ x \qquad (\text{cong}\vdash),(415),(\text{assume}) \qquad (418)$$

$$\overline{\Gamma},\ x : \overline{A},\ A'^?\ x \vdash^H_{\overline{T}} A^?\ x \wedge \overline{p}\ x \qquad (\wedge\text{I}),(418),(\text{monotonic}\vdash),(414) \qquad (419)$$

$$\overline{\Gamma},\ x : \overline{A} \vdash^H_{\overline{T}} A^?\ x \wedge \overline{p}\ x =_{\mathsf{bool}} A'^?\ x \qquad (\text{propExt}),(417),(419) \qquad (420)$$

$$\overline{\Gamma},\ x : \overline{A} \vdash^H_{\overline{T}} \left(\lambda y : \overline{A}.\ A^?\ y \wedge \overline{p}\ y\right)\ x =_{\mathsf{bool}} A'^?\ x \qquad (\text{rewrite}),(\text{beta}),(420) \qquad (421)$$

$$\overline{\Gamma} \vdash^H_{\overline{T}} \left(A|_p\right)^? =_{\Pi x : \overline{A}.\ \mathsf{bool}} A'^? \qquad \text{PT9,(extensionality),(421)}$$

($|_p$trivR): This case is analogous to the above case for rule ($|_p$trivL).

Typing Typing can be shown using the rules ($|_p$I), ($|_p$E:), (lambda'), (appl'), (⇒type'), (const), (cong:),(var), (=type):

($|_p$I):

$\Gamma \vdash_T t : A$	by assumption	(422)
$\Gamma \vdash_T p\ t$	by assumption	(423)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{t} : \overline{A}$	induction hypothesis,(422)	(424)
$\overline{\Gamma} \vdash^H_{\overline{T}} A^?\ \overline{t}$	induction hypothesis,(422)	(425)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{p}\ \overline{t}$	induction hypothesis,(423)	(426)
$\overline{\Gamma} \vdash^H_{\overline{T}} \left(A|_p\right)^?\ \overline{t}$	PT9,(∧I),(425),(426)	
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{t} : \overline{A|_p}$	PT8,(424)	

($|_p$E:):

$\Gamma \vdash_T t : A|_p$	by assumption	(427)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{t} : \overline{A|_p}$	induction hypothesis,(427)	(428)
$\overline{\Gamma} \vdash^H_{\overline{T}} \left(A|_p\right)^?\ \overline{t}$	induction hypothesis,(427)	(429)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{t} : \overline{A}$	PT8,(428)	
$\overline{\Gamma} \vdash^H_{\overline{T}} A^?\ \overline{t}$	PT9,(429)	

(lambda'):

$\Gamma,\ x : \overline{A} \vdash_T t : B$	by assumption	(430)
$\Gamma,\ x : \overline{A},\ A^?\ x \vdash^H_{\overline{T}} \overline{t} : \overline{B}$	induction hypothesis,PT6,(430)	(431)
$\Gamma,\ x : \overline{A},\ A^?\ x \vdash^H_{\overline{T}} B^?\ \overline{t}$	induction hypothesis,PT6,(430)	(432)
$\overline{\Gamma} \vdash^H_{\overline{T}} \forall\ x : \overline{A}.\ A^?\ x \Rightarrow B^?\ \overline{t}$	(⇒I),(∀I),(432)	(433)
$\Gamma,\ x : A \vdash^H_{\overline{T}} \overline{t} : \overline{B}$	typing is independent of assumptions,(431)	(434)
$\overline{\Gamma} \vdash^H_{\overline{T}} (\lambda x : \overline{A}.\ \overline{t}) : \overline{A} \to \overline{B}$	(lambda),(434)	(435)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{\lambda x : A.\ t} : \overline{\Pi x : A.\ B}$	PT15,PT10,(435)	
$\overline{\Gamma} \vdash^H_{\overline{T}} (\Pi x : A.\ B)^?\ \overline{\lambda x : A.\ t}$	PT11,(433)	

(appl'):

$\Gamma \vdash_T f : \Pi x : A.\ B$	by assumption	(436)
$\Gamma \vdash_T t : A$	by assumption	(437)

$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{f} : \overline{A} \to \overline{B}$	induction hypothesis,PT10,(436)	(438)
$\overline{\Gamma} \vdash^H_{\overline{T}} (\Pi x : A.\ B)^?\ \overline{f}$	induction hypothesis,PT10,(436)	(439)
$\overline{\Gamma} \vdash^H_{\overline{T}} \forall\, x : \overline{A}.\ A^?\ x \Rightarrow B^?\ (\overline{f}\ x)$	PT11,(439)	(440)
$\overline{\Gamma} \vdash^H_{\overline{T}} \bar{t} : \overline{A}$	induction hypothesis,(437)	(441)
$\overline{\Gamma} \vdash^H_{\overline{T}} A^?\ \bar{t}$	induction hypothesis,(437)	(442)
$\overline{\Gamma} \vdash^H_{\overline{T}} B^?\ (\overline{f}\ \bar{t})$	(∀E),(440),(442)	(443)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{f}\ \bar{t} : \overline{B}$	(appl),(438),(441)	(444)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{f\ t} : \overline{B}$	PT16,(444)	
$\overline{\Gamma} \vdash^H_{\overline{T}} B^?\ \overline{f\ t}$	PT16,(443)	

(⇒type'):

$\Gamma \vdash_T F : \mathsf{bool}$	by assumption	(445)
$\Gamma,\ F \vdash_T G : \mathsf{bool}$	by assumption	(446)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{F} : \mathsf{bool}$	induction hypothesis,(445)	(447)
$\overline{\Gamma} \vdash^H_{\overline{T}} \mathsf{bool}^?\ \overline{F}$	induction hypothesis,(445)	(448)
$\overline{\Gamma},\ \overline{F} \vdash^H_{\overline{T}} \overline{G} : \mathsf{bool}$	induction hypothesis,(446)	(449)
$\overline{\Gamma},\ \overline{F} \vdash^H_{\overline{T}} \mathsf{bool}^?\ \overline{G}$	induction hypothesis,(446)	(450)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{G} : \mathsf{bool}$	typing is independent of assumptions,(449)	(451)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{F} \Rightarrow (\mathsf{bool}^?\ \overline{G})$	(⇒I),(450)	(452)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{F} \Rightarrow \overline{G} : \mathsf{bool}$	(⇒type),(447),(451)	(453)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{F \Rightarrow G} : \mathsf{bool}$	PT17,(453)	
$\overline{\Gamma} \vdash^H_{\overline{T}} \mathsf{bool}^?\ \overline{F \Rightarrow G}$	IT13,PT17,(448),(452)	

(const):

$c : A \text{ in } T$	by assumption	(454)
$\vdash_T \Gamma\ \mathsf{Ctx}$	by assumption	(455)
$c : \overline{A} \text{ in } \overline{T}$	PT3,(454)	(456)
$A^?\ c \text{ in } \overline{T}$	PT3,(454)	(457)
$\vdash^H_{\overline{T}} \Gamma\ \mathsf{Ctx}$	induction hypothesis,(455)	(458)
$\overline{\Gamma} \vdash^H_{\overline{T}} c : \overline{A}$	(const),(456),(458)	(459)

$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{c} : \overline{A}$	PT13,(459)
$\overline{\Gamma} \vdash^H_{\overline{T}} A^? \; \overline{c}$	PT13,(457)

(var):

$x : A \text{ in } \Gamma$	by assumption	(460)
$\vdash_T \Gamma \; \mathsf{Ctx}$	by assumption	(461)
$x : \overline{A} \text{ in } \overline{\Gamma}$	PT6,(460)	(462)
$A^? \; x \text{ in } \overline{\Gamma}$	PT6,(460)	(463)
$\vdash_T \overline{\Gamma} \; \mathsf{Ctx}$	induction hypothesis,(461)	(464)
$\overline{\Gamma} \vdash^H_{\overline{T}} x : \overline{A}$	(var),(462),(464)	(465)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{x} : \overline{A}$	PT14,(465)	(466)
$\overline{\Gamma} \vdash^H_{\overline{T}} A^? \; \overline{x}$	PT14,(assume),(463),(464)	

(=type):

$\Gamma \vdash_T s : A$	by assumption	(467)
$\Gamma \vdash_T t : A$	by assumption	(468)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{s} : \overline{A}$	induction hypothesis,(467)	(469)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{t} : \overline{A}$	induction hypothesis,(468)	(470)
$\overline{\Gamma} \vdash^H_{\overline{T}} A^? \; \overline{s}$	induction hypothesis,(467)	(471)
$\overline{\Gamma} \vdash^H_{\overline{T}} A^? \; \overline{t}$	induction hypothesis,(468)	(472)
$\overline{\Gamma} \vdash^H_{\overline{T}} A^? \; \overline{s} \wedge A^? \; \overline{t}$	($\wedge$I),(471),(472)	(473)
$\overline{\Gamma} \vdash^H_{\overline{T}} A^? \; \overline{s} : \mathsf{bool}$	(validTyping),(471)	(474)
$\overline{\Gamma} \vdash^H_{\overline{T}} A^? \; \overline{t} : \mathsf{bool}$	(validTyping),(472)	(475)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{s} =_{\overline{A}} \overline{t} : \mathsf{bool}$	(=type),(469),(470)	(476)

We claim that we can then show

$$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{s =_A s} : \mathsf{bool}.$$

If A is not a function type, this follows directly from $\wedge$ being (by definition) of type $\mathsf{bool} \to \mathsf{bool} \to \mathsf{bool}$, rule (appl) and (474)-(476). Otherwise it follows by rule ($\forall$type), ($\Rightarrow$type), (appl), rule (eqTyping) and rule (appl) with (469), (470).

$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{s =_A t} : \mathsf{bool}$	see above explanation

$\overline{\Gamma} \vdash^{H}_{\overline{T}} \mathsf{bool}^{?}\ \overline{(s =_A t)}$ IT14,Lemma 13,(473)

(cong:):

$\Gamma \vdash_T t =_A t'$	by assumption	(477)
$\Gamma \vdash_T A \equiv A'$	by assumption	(478)
$\Gamma \vdash_T t : A$	by assumption	(479)
$\overline{\Gamma} \vdash^{H}_{\overline{T}} \overline{t =_A t'}$	induction hypothesis,(477)	(480)

We claim that from the prevous line (480), we can conclude:

$$\overline{\Gamma} \vdash^{H}_{\overline{T}} \overline{t'} : \overline{A}$$

In case A is not a function type, this follows using rule ($\wedge$El), rule (sym) and rule (eqTyping). If A is a function type we first show that the term on the right of the $\Rightarrow$ inside the scope of the universal quantifier is well typed using the rules rule ($\forall$E) (with a fresh variable), rule (validTyping) and rule (implTypingR). By the rules (sym), (eqTyping) and (applType) the desired claim follows.

$\overline{\Gamma} \vdash^{H}_{\overline{T}} \overline{t'} : \overline{A}$	see explanation
$\overline{\Gamma} \vdash^{H}_{\overline{T}} A^{?}\ \overline{t'}$	induction hypothesis,(477)

Term equality For term-equality and validity we need the assumption that the theory T doesn't contain constants of type with arity at least 3. This means that all terms in the DIHOLP derivation of a function type $\Pi x : \Pi y : C.\ D.\ B$ must be λ-functions. We then proceed by applying a simple proof transformation to the given DIHOLP derivation. Namely whenever for two λ-functions $f := \lambda x : \Pi y : C.\ D.\ s$ and $g := \lambda x : \Pi y : C.\ D.\ s'$ of type $\Pi x : \Pi y : C.\ D.\ B$ of arity n and two terms t, t' (which are not identical) of type $\Pi y : C.\ D$ rule (congAppl) is used to derive $f\ t =_B f'\ t'$, then we will replace this step by the following alternative steps: Use rule (congAppl) to conclude $\Gamma \vdash_T f\ t =_B f'\ t$. Use rule (beta) (and rule (sym)) to conclude $\Gamma \vdash_T f'\ t =_B s'[x/t]$ and $\Gamma \vdash_T s'[x/t'] =_B f'\ t'$. Derive that $\Gamma \vdash_T s'[x/t] =_B s'[x/t']$ using the congruence rules for the productions of the grammar (in case of equality ($=$cong), in case of $\Rightarrow$ rule ($\Rightarrow$cong), in case of function application (congAppl) and in case of λ-function rule (congλ)) and the fact that $\Gamma \vdash_T t =_{\Pi y:C.\ D} t'$. Conclude the desired statement of $\Gamma \vdash_T f\ t =_B f'\ t'$ by rule using (trans) (several times). Observe that iterating these replacements reduces the number of steps using rule (congAppl) to conclude a statement $\Gamma \vdash_T f\ t =_B f'\ t'$ for t, t' not identical and for f, f' having the highest arity of such terms in such steps. Thus repeating these replacements of steps will eventually result in a valid derivation that never uses rule (congAppl) to conclude a statement $\Gamma \vdash_T f\ t =_B f'\ t'$ for t, t' not identical and of function type. At this point, we can continue the proof of the theorem by induction on derivations again.

By rule (eqTyping), whenever we can show $\Gamma \vdash_T s =_T t$ in DPIHOL, we can also show $\Gamma \vdash_T s : T$ and $\Gamma \vdash_T t : T$. By induction hypothesis, this implies both $\overline{\Gamma} \vdash^H_{\overline{T}} T^? \, \overline{s}$ and $\overline{\Gamma} \vdash^H_{\overline{T}} T^? \, \overline{t}$.

By Lemma 13, it thus suffices to prove that $\Gamma \vdash_T s =_T t$ implies $\overline{\Gamma} \vdash^H_{\overline{T}} \overline{s} =_{\overline{T}} \overline{t}$.
If $T = \Pi x : A.\ B$ we can instead also show $\overline{\Gamma} \vdash^H_{\overline{T}} \forall x : \overline{A}.\ \forall x' : \overline{A}.\ \mathsf{Relat}_A[x =_{\overline{A}} x'] \Rightarrow \mathsf{Relat}_B[\overline{f}\ x =_{\overline{B}} \overline{g}\ x']$.

Term equality can be shown using the rules (congAppl'), (congλ'), (etaPi), (refl), (sym), (beta) and (propExt) in DPIHOL.

(congAppl') Firstly, let us consider the case of A not a function type:

$\Gamma \vdash_T t =_A t'$	by assumption	(481)
$\Gamma \vdash_T f =_{\Pi x:A.\ B} f'$	by assumption	(482)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{t} =_{\overline{A}} \overline{t'}$	($\wedge$El),induction hypothesis,(481)	(483)
$\overline{\Gamma} \vdash^H_{\overline{T}} A^? \, \overline{t}$	($\wedge$Er),($\wedge$El),induction hypothesis,(481)	(484)
$\overline{\Gamma} \vdash^H_{\overline{T}} \forall x : \overline{A}.\ A^? \, x \Rightarrow \overline{f}\ x =_{\overline{B}} \overline{f'}\ x$	induction hypothesis,PT10,(482)	(485)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{f}\ \overline{t} =_{\overline{\Pi x:A.\ B}} \overline{f'}\ \overline{t}$	($\Rightarrow$E),($\forall$E),(485),(484)	(486)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{f}\ \overline{t} =_{\overline{\Pi x:A.\ B}} \overline{f'}\ \overline{t'}$	(rewrite),(486),(483)	(487)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{f\ t} =_{\overline{\Pi x:A.\ B}} \overline{f'\ t'}$	PT16,(487)	(488)

Secondly consider the case of A being a function type. Without loss of generality we assume that A is not a predicate subtype of a function type as that only strengthens our assumptions without affecting what we need to prove). Thus A is of the form $\Pi x : C.\ D$. Since the theory doesn't contain constants of type $\Pi x : A.\ B$, it follows that f, f' are λ-functions. But such a step cannot appear in the derivation anymore (after the proof transformation discussed in the beginning of the paragraph about soundness of term equality). So this case cannot occur and there is nothing left to prove.

(congλ')

$\Gamma \vdash_T A \equiv A'$	by assumption	(489)
$\Gamma,\ x : A \vdash_T t =_B t'$	by assumption	(490)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{A} \equiv \overline{A'}$	induction hypothesis,(489)	(491)
$\overline{\Gamma},\ x : \overline{A},\ A^? \, x \vdash^H_{\overline{T}} \mathsf{Relat}_B[\overline{t} =_{\overline{B}} \overline{t'}]$	induction hypothesis,(490)	(492)
$\overline{\Gamma},\ x : \overline{A} \vdash^H_{\overline{T}} A^? \, x \Rightarrow \mathsf{Relat}_B[\overline{t} =_{\overline{B}} \overline{t'}]$	($\Rightarrow$I),(492)	(493)
$\overline{\Gamma} \vdash^H_{\overline{T}} \forall x : \overline{A}.\ A^? \, x \Rightarrow \mathsf{Relat}_B[\overline{t} =_{\overline{B}} \overline{t'}]$	(congλ'),Lemma 2,(493)	(494)

$$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{\lambda x : A.\ t =_{\Pi x : A.\ B} \lambda x : A'.\ t'} \qquad \text{PT20}$$

(etaPi)

$\Gamma \vdash_T t : \Pi x : A.\ B$	by assumption	(495)
$\overline{\Gamma} \vdash^H_{\overline{T}} \bar{t} : \overline{A} \to \overline{B}$	PT10,induction hypothesis,(495)	(496)
$\overline{\Gamma} \vdash^H_{\overline{T}} \bar{t} =_{\overline{A} \to \overline{B}} \lambda x : \overline{A}.\ \bar{t}\ x$	(eta),(496)	(497)
$\overline{\Gamma} \vdash^H_{\overline{T}} \bar{t} =_{\overline{\Pi x : A.\ B}} \overline{\lambda x : A.\ t\ x}$	PT15,PT10,(497)	

(refl)

$\Gamma \vdash_T t : A$	by assumption	(498)
$\overline{\Gamma} \vdash^H_{\overline{T}} \bar{t} : \overline{A}$	induction hypothesis,(498)	(499)
$\overline{\Gamma} \vdash^H_{\overline{T}} \bar{t} =_{\overline{A}} \bar{t}$	(refl),(499)	

(sym)

$\Gamma \vdash_T s =_A t$	by assumption	(500)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{s =_A t}$	induction hypothesis,(500)	(501)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{t =_A s}$	(rewrite),(501),(sym)	

(beta)

$\Gamma \vdash_T (\lambda x : A.\ s)\ t : B$	by assumption	(502)
$\overline{\Gamma} \vdash^H_{\overline{T}} (\lambda x : \overline{A}.\ \bar{s})\ \bar{t} : \overline{B}$	induction hypothesis,PT15,(502)	(503)
$\overline{\Gamma} \vdash^H_{\overline{T}} (\lambda x : \overline{A}.\ \bar{s})\ \bar{t} =_{\overline{B}} \bar{s}[x/t]$	(beta),(503)	(504)
$\overline{\Gamma} \vdash^H_{\overline{T}} (\lambda x : \overline{A}.\ \bar{s})\ \bar{t} =_{\overline{B}} \overline{s[x/t]}$	property (8),(504)	(505)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{(\lambda x : A.\ s)\ t} =_{\overline{B}} \overline{s[x/t]}$	PT15,PT16,(505)	(506)

(propExt)

$\Gamma,\ F \vdash_T G$	by assumption	(507)
$\Gamma,\ G \vdash_T F$	by assumption	(508)
$\overline{\Gamma},\ \overline{F} \vdash^H_{\overline{T}} \overline{G}$	induction hypothesis,PT16,(507)	(509)
$\overline{\Gamma},\ \overline{G} \vdash^H_{\overline{T}} \overline{F}$	induction hypothesis,PT16,(508)	(510)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{F} =_{\mathsf{bool}} \overline{G}$	(propExt),(509),(510)	(511)

Validity Validity can be shown using the rules ($|_p$Ep), (axiom), (assume), (cong$\vdash$), ($\Rightarrow$I) and ($\Rightarrow$E).

($|_p$Ep)

$\Gamma \vdash_T t : A\|_p$	by assumption	(512)
$\overline{\Gamma} \vdash^H_{\overline{T}} \left(A\|_p\right)^? \overline{t}$	induction hypothesis,(512)	(513)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{p}\,\overline{t}$	($\wedge$El),PT9,(513)	(514)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{p\,t}$	PT16,(514)	

(axiom)

F in T	by assumption	(515)
$\vdash_T \Gamma$ Ctx	by assumption	(516)
$\overline{F}$ in $\overline{T}$	PT4,(515	(517)
$\vdash^H_{\overline{T}} \overline{\Gamma}$ Ctx	induction hypothesis,516	(518)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{F}$	(axiom),(517),(518)	

(assume)

F in Γ	by assumption	(519)
$\vdash_T \Gamma$ Ctx	by assumption	(520)
$\overline{F}$ in $\overline{\Gamma}$	PT7,(519	(521)
$\vdash^H_{\overline{T}} \overline{\Gamma}$ Ctx	induction hypothesis,520	(522)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{F}$	(assume),(521),(522)	

(cong$\vdash$)

$\Gamma \vdash_T F =_{\mathsf{bool}} F'$	by assumption	(523)
$\Gamma \vdash_T F'$	by assumption	(524)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{F} =_{\mathsf{bool}} \overline{F'}$	($\wedge$El),induction hypothesis,(523)	(525)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{F'}$	induction hypothesis,(524	(526)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{F}$	(cong$\vdash$),(525),(526)	

(⇒I)

$\Gamma \vdash_T F : \mathsf{bool}$	by assumption	(527)
$\Gamma,\ F \vdash_T G$	by assumption	(528)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{F} : \mathsf{bool}$	induction hypothesis,(527)	(529)
$\overline{\Gamma},\ \overline{F} \vdash^H_{\overline{T}} \overline{G}$	induction hypothesis,PT7,(528)	(530)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{F} \Rightarrow \overline{G}$	(⇒I),(529),(530)	(531)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{F \Rightarrow G}$	PT17,(531)	

(⇒E)

$\Gamma \vdash_T F \Rightarrow G$	by assumption	(532)
$\Gamma \vdash_T F$	by assumption	(533)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{F} \Rightarrow \overline{G}$	induction hypothesis,PT17,(532)	(534)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{F}$	induction hypothesis,(533)	(535)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{G}$	(⇒E),(534),(535)	

□

Remark 33. Since the restriction on the theory for the soundness w.r.t. validity only allies to constants of type $\Pi x : \Pi y : C.\ D.\ B$ that are not defined as λ-functions, it would be interesting to consider this restriction semantically. We expect that models of theories will soundly translate to models of the translated theory (since in a model all constants are given definien). Therefore from a semantic viewpoint, the translation is sound without restrictions.

G Proof of soundness of Translation 1 from DHOLP into HOL

Proof of Corollary 15. The soundness proof is the same as in the intuitionistic case except that we need to consider the case of rule (boolExt) instead of rule (propExt). This case can be shown as follows:

$\Gamma \vdash_T p\ \mathsf{true}$	by assumption	(536)
$\Gamma \vdash_T p\ \mathsf{false}$	by assumption	(537)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{p}\ \mathsf{true}$	induction hypothesis,IT21,(536)	(538)
$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{p}\ \mathsf{false}$	induction hypothesis,IT21,(537)	(539)
$\overline{\Gamma} \vdash^H_{\overline{T}} \forall\, x : \mathsf{bool}.\ \overline{p}\ x$	(boolExt),(538),(539)	(540)

By definition, $\mathsf{bool}^?\ x$ holds for variables $x : \mathsf{bool}$, hence $\overline{\Gamma},\ x : \mathsf{bool} \vdash^H_{\overline{T}} \left(\mathsf{bool}^?\ x \Rightarrow \overline{p}\ x\right) =_{\mathsf{bool}} \overline{p}\ x$.

$$\overline{\Gamma} \vdash^H_{\overline{T}} \forall\, x : \mathsf{bool}.\ \left(\mathsf{bool}^?\ x \Rightarrow \overline{p}\ x\right) =_{\mathsf{bool}} \forall\, x : \mathsf{bool}.\ \overline{p}\ x \qquad (\forall\text{cong}),\ \text{above explanation} \quad (541)$$

$$\overline{\Gamma} \vdash^H_{\overline{T}} \forall\, x : \mathsf{bool}.\ \mathsf{bool}^?\ x \Rightarrow \overline{p}\ x \qquad (\text{cong}\vdash),(541),(540)$$

□

H Proof of Lemma 19 about replacement predicates

Proof of Lemma 19. To simplify the notations denote $\Delta := \overline{\Gamma, v_{q(\Gamma)+1} : A, \ldots, v_{r(t)} : A}$.

We want to show that the analogues of the typing axioms and assumptions for $A^?$ that are generated by the translation, will also hold for $\overline{p_A}$. Since nothing more is known about the $A^?$ and the typing and replacement predicates are defined analogously on Π-types and predicate subtypes, by induction on derivation it suffices to prove: $\Delta \vdash^H_{\overline{T}} A^? t$ implies $\Delta \vdash^H_{\overline{T}} \overline{p_A}\ t$ for base types (i.e. bool and types declared in the theory, but not function types or predicate subtypes).

We therefore need to show that $\Delta \vdash^H_{\overline{T}} A^? t$ implies $\Delta \vdash^H_{\overline{T}} \overline{p_A}\ t$ for base types. If t is a variable it is one of $v_1, \ldots, v_q$, say v_i. Hence $\overline{p_A\ t} = \overline{\mathsf{true}} = \mathsf{true}$ holds by definition and there is nothing to prove.

It remains to consider the case of t not a variable.

For non-variables the translation generates exactly the axioms $\forall\, r_{i,1} : \overline{T_{i,1}}.\ {T_{i,1}}^?\ r_{i,1} \Rightarrow \ldots \forall\, r_{i,1} : \overline{T_{i,1}}.\ {T_{i,1}}^?\ r_{i,1} \Rightarrow \left(a\ f_{1,i}(r_{i,1}, \ldots, r_{i,p_i})\ \ldots\ f_{n,i}(r_{i,1}, \ldots, r_{i,p_i})\right)^?\ c_i\ r_{i,1}\ \ldots\ r_{i,p_i}$ for each constructor $c_i : \Pi r_{i,1} : T_{i,1}.\ \ldots \Pi r_{i,p_i} : T_{i,p_i}.\ a\ f_{1,i}(r_{i,1}, \ldots, r_{i,p_i})\ \ldots\ f_{n,i}(r_{i,1}, \ldots, r_{i,p_i})$.

We therefore have to prove that if in the scope of t (i.e. in context Δ) we have

$$a\ f_{1,i}(r_{i,1}, \ldots, r_{i,p_i})\ \ldots\ f_{n,i}(r_{i,1}, \ldots, r_{i,p_i}) \equiv A$$

for $r_{i,1}, \ldots, r_{i,p_i}$ of type $T_{i,1}, \ldots, T_{i,p_i}$, then we have

$$\overline{p_A\ c_i\ r_{i,1}\ \ldots\ r_{i,p_i}}.$$

Then, the remaining claim follows by induction on derivations.

Since $c_i\ r_{i,1}\ \ldots\ r_{i,p_i}$ doesn't have a function type, it must be of the form $a\ t_1\ \ldots\ t_n$ (with $n = 0$ and $a = \mathsf{bool}$ allowed) it follows that the type equality

$$a\ f_{1,i}(r_{i,1}, \ldots, r_{i,p_i})\ \ldots\ f_{n,i}(r_{i,1}, \ldots, r_{i,p_i}) \equiv A,$$

can only be shown by rule (congBase') in DIHOLP. Hence,

$$\Gamma \vdash_T t_1 =_{T_{i,1}} f_{1,i}(r_{i,1}, \ldots, r_{i,p_i}), \ldots, \Gamma \vdash_T t_n =_{T_{i,p_i}} f_{n,i}(r_{i,1}, \ldots, r_{i,p_i})$$

must all hold in DIHOLP.

Since $\vee$ commutes with the translation, by the introduction rule of $\vee$ it follows that we need to show that the translation of one of the disjuncts of $\overline{p_A\ (c_i\ r_{i,1}\ \ldots\ r_{i,p_i})}$ holds. We will show this for the i-th conjunct namely for:
$\exists r_{i,1} : \overline{T_{i,1}}.{T_{i,1}}^?\ r_{i,1} \wedge \ldots \exists r_{i,p_i} : \overline{T_{i,p_i}}.{T_{i,p_i}}^?\ r_{i,p_i} \wedge (\mathsf{Relat}_{A_1}[\overline{t_1} =_{\overline{A_1}} \overline{f_{1,i}}(r_{i,1}, \ldots, r_{i,p_i})]) \wedge \ldots \wedge (\mathsf{Relat}_{A_n}[\overline{t_n} =_{\overline{A_n}} \overline{f_{n,i}}(r_{i,1}, \ldots, r_{i,p_i})]) \wedge (\mathsf{Relat}_A[(c_i\ r_{i,1}\ \ldots\ r_{i,p_i}) =_{\overline{A}} c_i\ r_{i,1}\ \ldots\ r_{i,p_i}])$.
Since the $r_{i,j}$ have type $T_{i,j}$ respectively in DIHOLP, by the soundness of the translation, it follows that ${T_{i,j}}^?\ \overline{r_{i,j}}$ for all i, j.

Furthermore by the soundness of the translation and the equalities between the t_j and the applications of the $f_{j,i}$ to the $r_{i,j}$, it follows that $\Delta \vdash^H_{\overline{T}} \overline{t_1 =_{T_{i,1}} f_{1,i}(r_{i,1}, \ldots, r_{i,p_i})}$, $\ldots$, $\Delta \vdash^H_{\overline{T}} \overline{t_n =_{T_{i,p_i}} f_{n,i}(r_{i,1}, \ldots, r_{i,p_i})}$ hold. This step requires the assumption that the theory contains no constants of types $\Pi x : \Pi y : C.\ D.\ B$.

By the soundness of the translation w.r.t. typing $\overline{c_i\ r_{i,1}\ \ldots\ r_{i,p_i}} : \overline{A}$ and thus by rule (refl):

$$\mathsf{Relat}_A[(c_i\ r_{i,1}\ \ldots\ r_{i,p_i}) =_{\overline{A}} c_i\ r_{i,1}\ \ldots\ r_{i,p_i}].$$

The claim thus follows by repeated application of the introduction rules for $\vee$ and the introduction rule for $\exists$ for the $r_{i,j}$. □

I Proof of Lemma 18 about replacement predicates

Proof of Lemma 18. Since the definitions of the replacement predicates and typing predicates for Π-types, predicates subtypes and booleans are completely analogous, we can by induction on the derivations reduce the remaining claim to the cases of a disjunct of $\overline{p_B\ t}$ (corresponding to a typing assumption/axiom of a non-function or predicate subtype) holding. In such a case, we have that B is not a function type or predicate subtype and t is not the application of a predicate to all its arguments.

If t is a variable of type other than bool, we have $\mathsf{Relat}_A[t =_{\overline{A}} v_i]$ for some variable v_i in the scope and the claim ${A}^?\ t$ is actually a conjunct of the assumption.

If t is a boolean variable, it follows that tm is a boolean variable, so $tm : \mathsf{bool}$ trivially holds.

It remains to consider the case of

$$tm = c\ d_1\ \ldots\ d_p$$

being the application of term constructor

$$c : \Pi r_1 : T_1.\ \ldots \Pi r_p : T_p.\ a\ f_1(r_1, \ldots, r_p)\ \ldots\ f_n(r_1, \ldots, r_p)$$

in the theory or scope and

$$B = a\ t_1\ \ldots\ t_n.$$

In this case, there is a typing axiom or assumption in $\overline{T}$ or $\overline{\Gamma}$ that (after reducing with rule (∀E) and (⇒E)) asserts the claim of ${B}^?\ t$. □

J Proof that replacement predicates holding implies typing

Proof of Lemma 20. The below DIHOLP analogue of rule (typingWf) can be derived similarly to how it is derived in IHOL:

$$\frac{\Gamma \vdash_T f\ t : D \quad \Gamma \vdash_T f : \Pi x : C.\ D}{\Gamma \vdash_T t : C}\text{typingWellformed'}$$

If B is a predicate subtype $B'|_p$, the definition at R7 implies that $p\ t$ must hold, so rule ($|_p$I) allow us to reduce the claim to the case of the simpler type B'.

Wlog. (rule (rewrite) guarantees the equivalence) assume that t is beta and eta reduced.

Continue by induction on the shape of t. If B is bool but t is not the application of a function of (eventual) return type bool to all its arguments, then t is a also not a variable or constant of type bool (or we would be in the case of an empty function application) and hence t is an equality or implication. The definition of p_{bool} combined with the induction hypothesis then ensures that all the subterms have the appropriate types and we can use one of the typing rules (=type) or (⇒type) to conclude $t : \mathsf{bool}$ as desired.

It follows that p_B is an actual predicate on type B i.e. $\Gamma \vdash_T p_B : \Pi x : B.\ \mathsf{bool}$. By rule (validTyping) (and Remark 12 about its proof in D(I)HOLP) and rule (typingWellformed') it follows that $\Gamma \vdash_T t : B$. □

K Proof of Lemma 21 about the P-normalizing proof transformation

Proof of Lemma 21. We will show this by induction on the inference rules.

Firstly, we observe that there are no dependent types in IHOL and the context and axioms contain no spurious subterms. Hence, well-formedness (of theories, contexts, types) and type-equality judgements are unaffected by the transformation. So there is nothing to prove for the well-formedness and type-equality rules.

Regarding the indices for the terms: For proper terms there is no choice here. We start indexing the term at the end of the derivation and go up line by line. Whenever we need to pick an index for a term we already encountered (in a later step) in the derivation we will pick the same index. Whenever we need to pick an index for a non-atomic term we pick indices for the atomic terms in the term and then choose the type of the quasi-preimage (by termwise-injectivity (see remark 7) this quasi-preimage is unique) for which the quasi-preimages of the subterm have the types they are indexed with. If we have to pick an index for a variable that is part of a λ-function we see if that λ-function is applied to an argument and if so we pick the index of that argument (this choice ensures that the reduction will do as little as necessary and is also consistent with the indexing of variables in proper terms). Otherwise, we pick the index arbitrarily (but satisfying the already stated requirements).

It remains to consider the typing and validity rules and to show that if the P-normalizing statement transformation of the assumptions hold, then the P-normalizing statement transformation of the conclusion holds:

(const): Since constants are proper terms satisfying predicate P, there is nothing to prove.

(var): Since context variables are proper terms satisfying predicate P, there is nothing to prove.

(=type):

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\,(s)_A : \overline{A} \qquad \text{by assumption} \qquad (542)$$

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\,(t)_A : \overline{A} \qquad \text{by assumption} \qquad (543)$$

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\,(s)_A =_{\overline{A}} \mathsf{sRed}\,(t)_A : \mathsf{bool} \qquad (=\text{type}),(542),(543) \qquad (544)$$

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\left(s_A =_{\overline{A}} t_A\right)_{\mathsf{bool}} : \mathsf{bool} \qquad \text{SR4},(544)$$

(lambda):

$$\Delta,\ x_A : \overline{A} \vdash^H_{\overline{T}} \mathsf{sRed}\left(t_B\right) : \overline{B} \qquad \text{by assumption} \qquad (545)$$

$$\Delta \vdash^H_{\overline{T}} \left(\lambda x_A : \overline{A}.\ \mathsf{sRed}\,(t)_B\right) : \overline{A} \to \overline{B} \qquad (\text{lambda}),(545) \qquad (546)$$

If $\mathsf{sRed}\,(t)_B$ isn't a function application $\mathsf{sRed}\,(f)_{\Pi y:A.\ B}\ x_A$ not satisfying P for which x doesn't appear in f:

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\left(\lambda x_A : \overline{A}.\ \mathsf{sRed}\,(t)_B\right) : \overline{A} \to \overline{B} \qquad \text{SR6},(546)$$

Else by (SR3) we have $\mathsf{sRed}\left(\lambda x_A : \overline{A}.\ \mathsf{sRed}\,(t)_B\right) = \mathsf{sRed}\,(f)_{\Pi y:A.\ B}$:

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\left(\lambda x_A : \overline{A}.\ \mathsf{sRed}\,(t)_B\right) : \overline{A} \to \overline{B} \qquad \text{SR3},(546)$$

(appl):

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\,(f)_{\Pi x:A.\ B} : \overline{A} \to \overline{B} \qquad \text{by assumption} \qquad (547)$$

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\,(t)_{A'} : \overline{A} \qquad \text{by assumption} \qquad (548)$$

Unless $\mathsf{sRed}\,(f)\ \mathsf{sRed}\,(t)$ beta reducible and either not satisfing P or not satisfying $A \equiv A$:

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\left(f_{\Pi x:A.\ B}\ t_{A'}\right) : \overline{B} \qquad \text{SR1},(\text{lambda}),(547),(548)$$

Otherwise if $\mathsf{sRed}\,(f)$ $\mathsf{sRed}\,(t)$ beta reducible with $\mathsf{sRed}\,(f)_{\Pi x:A.\ B} = \lambda x_A : \overline{A}.\ \mathsf{sRed}\,(s)_B$ and is either not satisfing P or not satisfying $A \equiv A$, it follows that

$$\mathsf{sRed}\left(f_{\Pi x:A.\ B}\ t_{A'}\right) = \mathsf{sRed}\left(\mathsf{sRed}\,(s)_B\,[{}^{x_A}/_{\mathsf{sRed}\,(t)_{A'}}]\right).$$

Observe that $\Delta,\ x : \overline{A} \vdash^H_{\overline{T}} \mathsf{sRed}\,(s) : \overline{B}$ must be derivable and thus:

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\,(s)_B\,[{}^{x_A}/_{\mathsf{sRed}\,(t)_{A'}}] : \overline{B} \qquad \text{(rewriteTyping),assumption,(548)}$$

Observe that by induction hypothesis the derivations of $\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\,(s)_B : \overline{B}$ and of $\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\,(t)_A : \overline{A}$ are almost proper with quasi-preimages for terms of indexed types. Consequently, the steps in the derivation obtained by plugging in the proof of rule (rewriteTyping) into this case will also be almost proper with quasi-preimages of indexed types (as the terms in it for corresponding steps have the same types and indices as in the derivation for $\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\,(s)_B : \overline{B}$ and steps in the derivation of $\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\,(t)_A : \overline{A}$ which occur in this derivation are unchanged). Therefore treating rule (rewriteTyping) like a primitive rule is harmless here (we can replace a step using the rule by the steps in the derivation of the rule). By (P5) and the fact that $\mathsf{sRed}\,(s)_B\,[{}^{x_A}/_{\mathsf{sRed}\,(t)_{A'}}]$ (and its subterms) are almost proper with quasi-preimages of the indexed types, it follows that P holds on $\mathsf{sRed}\,(s)_B\,[{}^{x_A}/_{\mathsf{sRed}\,(t)_{A'}}]$. Therefore by (SR1) we have $\mathsf{sRed}\left(\mathsf{sRed}\,(s)_B\,[{}^{x_A}/_{\mathsf{sRed}\,(t)_{A'}}]\right) = \mathsf{sRed}\,(s)_B\,[{}^{x_A}/_{\mathsf{sRed}\,(t)_{A'}}]$ and we are already done.

(⇒type):

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\,(F)_{\mathsf{bool}} : \mathsf{bool} \qquad \text{by assumption} \qquad (549)$$
$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\,(G)_{\mathsf{bool}} : \mathsf{bool} \qquad \text{by assumption} \qquad (550)$$
$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\,(F)_{\mathsf{bool}} \Rightarrow \mathsf{sRed}\,(G)_{\mathsf{bool}} : \mathsf{bool} \qquad (\Rightarrow\text{type}),(549),(550) \qquad (551)$$
$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\left(F_{\mathsf{bool}} \Rightarrow F_{\mathsf{bool}}\right) : \mathsf{bool} \qquad \text{SR5},(551)$$

(axiom): Since translations of axioms to IHOL are always proper terms, there is nothing to prove here.

(assume):

$$\mathsf{sRed}\,(F)_{\mathsf{bool}} \text{ in } \Delta \qquad \text{by assumption} \qquad (552)$$
$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\,(F)_{\mathsf{bool}} \qquad \text{(assume)},(552)$$

By assumption $\mathsf{sRed}\,(F)$ almost proper (with a quasi-preimage of type bool), so the conclusion of the rule is almost proper.

(congλ):

$$\Delta \vdash^H_{\overline{T}} A \equiv A' \quad \text{by assumption} \quad (553)$$

$$\Delta,\ x_A : \overline{A} \vdash^H_{\overline{T}} \mathsf{sRed}\left(t_B =_{\overline{B}} t'_B\right) \quad \text{by assumption} \quad (554)$$

$$\Delta,\ x_A : \overline{A} \vdash^H_{\overline{T}} \mathsf{sRed}\,(t)_B =_{\overline{B}} \mathsf{sRed}\left(t'\right)_B \quad \text{SR4,(554)} \quad (555)$$

$$\Delta \vdash^H_{\overline{T}} \lambda x_A : \overline{A}.\ \mathsf{sRed}\,(t)_B =_{\overline{A}\to\overline{B}} \lambda x_A : \overline{A}.\ \mathsf{sRed}\left(t'\right)_B \quad (\text{cong}\lambda),(553),(555) \quad (556)$$

By assumption $\mathsf{sRed}\,(t)_B =_{\overline{B}} \mathsf{sRed}\left(t'\right)_B$ almost proper with quasi-preimage consistent with type indices and $A \equiv A'$, thus also $\lambda x_A : \overline{A}.\ \mathsf{sRed}\,(t)_B =_{\overline{A}\to\overline{B}} \lambda x_A : \overline{A}.\ \mathsf{sRed}\left(t'\right)_B$ almost proper with quasi-preimage consistent with type indices. It follows that the same is true for the (repeated) beta and eta reductions of those λ-functions.
Using the rules (beta), (eta) and (trans), it follows that beta and eta reductions (due to terms not satisfying property P) will not affect the derivability (using only almost proper terms with quasi-preimages consistent with the type indices) of the conclusion. So wlog. assume that the two λ-functions do satisfy the property P:

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\left(\lambda x_A : \overline{A}.\ \mathsf{sRed}\,(t)_B =_{\overline{A}\to\overline{B}} \lambda x_A : \overline{A}.\ \mathsf{sRed}\left(t'\right)_B\right) \text{SR6,SR4,(556)}$$

(congAppl):

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\left(t_{A'} =_{\overline{A}} t'_{A'}\right) \quad \text{by assumption} \quad (557)$$

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\,(t)_{A'} =_{\overline{A}} \mathsf{sRed}\left(t'\right)_{A'} \quad \text{SR4557} \quad (558)$$

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\left(f_{\Pi x:A.\ B} =_{\overline{A}\to\overline{B}} f'_{\Pi x:A.\ B}\right) \quad \text{by assumption} \quad (559)$$

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\,(f)_{\Pi x:A.\ B} =_{\overline{A}\to\overline{B}} \mathsf{sRed}\left(f'\right)_{\Pi x:A.\ B} \quad \text{SR4,(559)} \quad (560)$$

Assume that $A \not\equiv A'$. Our choice of type indices then implies $\mathsf{sRed}\,(f)$ and $\mathsf{sRed}\left(f'\right)$ are not λ-functions. Consequently, the applications $\mathsf{sRed}\,(f)\ \mathsf{sRed}\,(s)$ and $\mathsf{sRed}\left(f'\right)\ \mathsf{sRed}\left(s'\right)$ are not beta or eta reducible. Thus, $\mathsf{sRed}\left(\mathsf{sRed}\,(f)_{\Pi x:A.\ B}\ \mathsf{sRed}\,(t)_{A'}\right) = t_{\overline{B}}$ and
$\mathsf{sRed}\left(\mathsf{sRed}\left(f'\right)_{\Pi x:A.\ B}\ \mathsf{sRed}\left(t'\right)_{A'}\right) = t_{\overline{B}}$ and we yield:

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\left(\mathsf{sRed}\,(f)_{\Pi x:A.\ B}\ \mathsf{sRed}\,(t)_{A'}\right) =_{\overline{B}} \mathsf{sRed}\left(\mathsf{sRed}\left(f'\right)_{\Pi x:A.\ B}\ \mathsf{sRed}\left(t'\right)_{A'}\right) \quad \text{(refl)}$$

Otherwise the applications $\mathsf{sRed}\,(f)_{\Pi x:A.\ B}\ \mathsf{sRed}\,(t)_{A'}$ and $\mathsf{sRed}\left(f'\right)_{\Pi x:A.\ B}\ \mathsf{sRed}\left(t'\right)_{A'}$ are almost proper with quasi-preimages consistent with type indices. It follows that the same is true for the (repeated) beta and eta reductions of those function applications.
Considering the rules (beta), (eta) and (trans), it follows that beta and eta reductions (due to terms not satisfying property P) will not affect the derivability (using only almost proper terms with quasi-preimages consistent with the type indices) of the conclusion. So wlog. we can assume that the two function applications both either are not beta reducible or do satisfy property P. It follows:

$$\mathsf{sRed}\left(\mathsf{sRed}\,(f)_{\Pi x:A.\ B}\ \mathsf{sRed}\,(t)_{A'}\right) = \mathsf{sRed}\,(f)_{\Pi x:A.\ B}\ \mathsf{sRed}\,(t)_{A'}$$

and

$$\mathsf{sRed}\left(\mathsf{sRed}\left(f'\right)_{\Pi x:A.\ B}\ \mathsf{sRed}\left(t'\right)_{A'}\right) = \mathsf{sRed}\left(f'\right)_{\Pi x:A.\ B}\ \mathsf{sRed}\left(t'\right)_{A'}$$

and thus:

$$\begin{aligned}
&\Delta \vdash^{H}_{\overline{T}} \mathsf{sRed}\left(\mathsf{sRed}\,(f)_{\Pi x:A.\ B}\ \mathsf{sRed}\,(t)_{A'}\right) =_{\overline{B}} \\
&\qquad \mathsf{sRed}\left(\mathsf{sRed}\left(f'\right)_{\Pi x:A.\ B}\ \mathsf{sRed}\left(t'\right)_{A'}\right) && \text{(congAppl),(558),(560)} && (561)\\
&\Delta \vdash^{H}_{\overline{T}} \mathsf{sRed}\left(\mathsf{sRed}\,(f)_{\Pi x:A.\ B}\ \mathsf{sRed}\,(t)_{A'} =_{\overline{B}} \mathsf{sRed}\left(f'\right)_{\Pi x:A.\ B}\ \mathsf{sRed}\left(t'\right)_{A'}\right) && \text{SR4,(561)}
\end{aligned}$$

(refl):

$$\begin{aligned}
&\Delta \vdash^{H}_{\overline{T}} \mathsf{sRed}\,(t)_A : \overline{A} && \text{by assumption} && (562)\\
&\Delta \vdash^{H}_{\overline{T}} \mathsf{sRed}\,(t)_A =_{\overline{A}} \mathsf{sRed}\,(t)_A && \text{(refl),(562)} && (563)\\
&\Delta \vdash^{H}_{\overline{T}} \mathsf{sRed}\left(\mathsf{sRed}\,(t)_A =_{\overline{A}} \mathsf{sRed}\,(t)_A\right) && \text{SR4,(563)}
\end{aligned}$$

(sym):

$$\begin{aligned}
&\Delta \vdash^{H}_{\overline{T}} \mathsf{sRed}\left(t_A =_{\overline{A}} s_A\right) && \text{by assumption} && (564)\\
&\Delta \vdash^{H}_{\overline{T}} \mathsf{sRed}\,(t)_A =_{\overline{A}} \mathsf{sRed}\,(s)_A && \text{SR4,(564)} && (565)\\
&\Delta \vdash^{H}_{\overline{T}} \mathsf{sRed}\,(s)_A =_{\overline{A}} \mathsf{sRed}\,(t)_A && \text{(sym),(565)} && (566)\\
&\Delta \vdash^{H}_{\overline{T}} \mathsf{sRed}\left(s_A =_{\overline{A}} t_A\right) && \text{SR4,(566)}
\end{aligned}$$

(beta):

$$\Delta \vdash^{H}_{\overline{T}} \mathsf{sRed}\left(\left(\lambda x_A : \overline{A}.\ s_B\right)\ t_A\right) : \overline{B} \qquad \text{by assumption} \qquad (567)$$

$\left(\lambda x_A : \overline{A}.\ s_B\right)\ t_A$ is by choice of indexing almost proper with quasi-preimage of type B. If it does satisfy P, it follows:

$$\Delta \vdash^{H}_{\overline{T}} \left(\lambda x_A : \overline{A}.\ \mathsf{sRed}\left(s_B\right)\right)\ \mathsf{sRed}\left(t_A\right) : \overline{B} \qquad \text{SR2,(567)} \qquad (568)$$

$$\Delta \vdash^H_{\overline{T}} \left(\lambda x_A : \overline{A}.\ \mathsf{sRed}\left(s_B\right)\right)\ \mathsf{sRed}\left(t_A\right) =_{\overline{B}} \mathsf{sRed}\left(s_B\right)[^{x_A}/_{\mathsf{sRed}(t_A)}] \quad \text{(beta),(568)} \quad (569)$$

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\left(\left(\lambda x_A : \overline{A}.\ \mathsf{sRed}\left(s_B\right)\right)\ \mathsf{sRed}\left(t_A\right) =_{\overline{B}} \mathsf{sRed}\left(s_B\right)[^{x_A}/_{\mathsf{sRed}(t_A)}]\right) \quad \text{SR4,SR1,P4,(569)}$$

Otherwise, we have $\mathsf{sRed}\left(\left(\lambda x_A : \overline{A}.\ \mathsf{sRed}\left(s_B\right)\right)\ \mathsf{sRed}\left(t_A\right)\right) = \mathsf{sRed}\left(\mathsf{sRed}\left(s_B\right)[^{x_A}/_{\mathsf{sRed}(t_A)}]\right)$ and thus:

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\left(\left(\lambda x_A : \overline{A}.\ \mathsf{sRed}\left(s_B\right)\right)\ \mathsf{sRed}\left(t_A\right) =_{\overline{B}} \mathsf{sRed}\left(s_B\right)[^{x_A}/_{\mathsf{sRed}(t_A)}]\right) \quad \text{(refl),SR4}$$

(eta):

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\,(t)_{\Pi x:A.\ B} : \overline{A} \to \overline{B} \quad \text{by assumption} \quad (570)$$

$$x \text{ not in } \Delta \quad \text{by assumption} \quad (571)$$

$\lambda x_A : \overline{A}.\ \mathsf{sRed}\,(t)_{\Pi x:A.\ B}\ x_A$ is by choice of indexing almost proper with quasi-preimage of type $\Pi x : A.\ B$. If it does satisfy P, it follows:

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\,(t)_{\Pi x:A.\ B} =_{\overline{A}\to\overline{B}} \lambda x_A : \overline{A}.\ \mathsf{sRed}\,(t)_{\Pi x:A.\ B}\ x_A \quad \text{(eta),(570),(571)} \quad (572)$$

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\left(\mathsf{sRed}\,(t)_{\Pi x:A.\ B} =_{\overline{A}\to\overline{B}} \lambda x_A : \overline{A}.\ \mathsf{sRed}\,(t)_{\Pi x:A.\ B}\ x_A\right) \quad \text{SR2,SR6,SR4,(572)} \quad (573)$$

Otherwise we have $\mathsf{sRed}\left(\lambda x_A : \overline{A}.\ \mathsf{sRed}\,(t)_{\Pi x:A.\ B}\ x_A\right) = \mathsf{sRed}\,(t)_{\Pi x:A.\ B}$ and thus:

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\left(\mathsf{sRed}\,(t)_{\Pi x:A.\ B} =_{\overline{A}\to\overline{B}} \lambda x_A : \overline{A}.\ \mathsf{sRed}\,(t)_{\Pi x:A.\ B}\ x_A\right) \quad \text{(refl),SR4}$$

(cong$\vdash$):

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\left(F_{\mathsf{bool}} =_{\mathsf{bool}} F'_{\mathsf{bool}}\right) \quad \text{by assumption} \quad (574)$$

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\left(F'\right)_{\mathsf{bool}} \quad \text{by assumption} \quad (575)$$

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\,(F)_{\mathsf{bool}} =_{\mathsf{bool}} \mathsf{sRed}\left(F'\right)_{\mathsf{bool}} \quad \text{SR4,(574)} \quad (576)$$

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\,(F)_{\mathsf{bool}} \quad \text{(cong}\vdash\text{),(576),(575)}$$

($\Rightarrow$I):

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\,(F)_{\mathsf{bool}} : \mathsf{bool} \quad \text{by assumption} \quad (577)$$

$$\Delta,\ \mathsf{sRed}\,(F)_{\mathsf{bool}} \vdash^H_{\overline{T}} \mathsf{sRed}\,(G)_{\mathsf{bool}} \quad \text{by assumption} \quad (578)$$

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\,(F)_{\mathsf{bool}} \Rightarrow \mathsf{sRed}\,(G)_{\mathsf{bool}} \quad (\Rightarrow\text{I}),(577),(578) \quad (579)$$

$$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\left(F_{\mathsf{bool}} \Rightarrow G_{\mathsf{bool}}\right) \quad \text{SR5,(579)}$$

(⇒E):

$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\left(F_{\mathsf{bool}} \Rightarrow G_{\mathsf{bool}}\right)$	by assumption	(580)
$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\,(F)_{\mathsf{bool}}$	by assumption	(581)
$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\,(F)_{\mathsf{bool}} \Rightarrow \mathsf{sRed}\,(G)_{\mathsf{bool}}$	SR5,(580)	(582)
$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\,(G)_{\mathsf{bool}}$	(⇒E),(582),(581)	

(propExt):

$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\,(F)_{\mathsf{bool}} : \mathsf{bool}$	by assumption	(583)
$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\,(G)_{\mathsf{bool}} : \mathsf{bool}$	by assumption	(584)
$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\,(F)_{\mathsf{bool}} =_{\mathsf{bool}} \mathsf{sRed}\,(G)_{\mathsf{bool}}$	(propExt),(583),(584)	(585)
$\Delta \vdash^H_{\overline{T}} \mathsf{sRed}\left(F_{\mathsf{bool}} =_{\mathsf{bool}} G_{\mathsf{bool}}\right)$	SR4,(585)	

□

L Proof that relativized equalities imply typing conditions

Proof of Lemma 23. If B is not a function type the claims are conjuncts of the assumption and follow immediately (by rule (∧El) and rule (∧Er)). Otherwise B is of the shape $\Pi x : C.\ D$ and the second claim is:

$$\forall\, x : \overline{C}.\ C^?\ x \Rightarrow \mathsf{Relat}_D[\overline{s}\ x =_{\overline{D}} \overline{t}\ x].$$

Let us compare this with the typing predicate $\left(\Pi x : C.\ D\right)^?\ y$ which is

$$\forall\, x : \overline{C}.\ C^?\ x \Rightarrow D^?\ y.$$

By the rules (∀cong) and (⇒cong) we can reduce the claim to the analogous claim for the type D, instead of B. Continuing by induction on the shape of D finishes the proof. □

M Proof of Theorem 25 about lifting IHOL to DIHOLP proofs

Proof. As shown in Corollary 22, we may assume that the derivation of $\overline{\Gamma} \vdash^H_{\overline{T}} \varphi$ contains only almost proper terms. Consequently, for any unrelativized equality $\overline{s} =_{\overline{A}} \overline{t}$ in the derivation, we know that also its relativized version $\mathsf{Relat}_A[\overline{s} =_{\overline{A}} \overline{t}]$ holds.

If such a relativized equality occurs in the assumption of a rule the induction hypothesis (assuming that the assumption is proper) yields $s =_A t$ in DIHOLP. By Lemma 24 it follows that also $s : A$ and $t : A$ are derivable in DIHOLP.

Without loss of generality (adding extra assumptions throughout the proof) we may assume that the context of the (final) conclusion is the translation of a DIHOLP context.

By remark 7, the translation is termwise-injective. Therefore, if we can lift a derivation of a proper validity statement to a derivation of a quasi-preimage DIHOLP then this lift is a derivation of is uniquely determined and we have created a derivation for the right conjecture in DIHOLP.

We prove the claim of $\Gamma \vdash F$ by induction on the validity rules of IHOL as follows:

Given a validity rule R with assumptions $A_1, \dots, A_n$, validity assumptions (assumptions that are validity judgements) $V_1, \dots, V_m$, non-judgement assumptions (meaning assumptions that something occurs in a context or theory) $N_1, \dots, N_p$ and conclusion C we will show the following:

Claim. *Assuming that the A_i and the N_j hold.*

1. *Assume that the conclusion C is proper with quasi-preimage C^{-1}. It follows that the contexts C_i of the A_i are proper and the quasi-preimages of the C_i are uniquely determined from the quasi-preimage C^{-1} of C.*
2. *Assume that whenever an A_i is proper its quasi-preimage (where we choose the same preimages for identical terms and types with several possible preimages) holds in DIHOLP and that the conclusion C is proper with quasi-preimage C^{-1}. Then, C^{-1} holds in DIHOLP.*

Let us start with the first part of this claim, namely that if C is proper (i.e. the formula in C^{-1} well-typed in the context of C^{-1}) then the C_i are proper and their quasi-preimages are uniquely determined from C^{-1}. Since all formulae appearing in the derivation are almost proper, this implies that the A_i themselves are proper and the contexts of their quasi-preimages fit together with the context of C^{-1}.

The translation clearly implies that if an N_j holds in IHOL, the corresponding non-judgement assumption ${N_j}^{-1}$ holds in DIHOLP (e.g. if $\overline{F}$ is an axiom in $\overline{T}$, then F must be an axiom in T).

Since the validity judgement being derived is proper, it follows from this first part of the claim that the validity assumptions of all validity rules in the derivation are proper and the contexts of the quasi-preimages of those assumptions are uniquely determined.

By induction on the validity rules, if given an arbitrary validity rule R whose assumptions hold and whose validity assumptions all satisfy a property P we can show that P holds on the conclusion of R, then all derivable validity judgments have property P. Since all the validity assumptions and conclusions of validity rules in the derivation are proper, the property of having a derivable quasi-preimage is such a property. By the above induction principle, it thus suffices to prove the second part of the claim for the validity rules in IHOL.

Validity can be shown using the rules (congλ), (eta), (congAppl), (beta), (refl), (sym), (assume), (axiom), (⇒I), (⇒E) and (propExt).

(congλ): Since the conclusion is proper, it follows that the relativization $\overline{\Gamma} \vdash^H_{\overline{T}} \forall\, x : \overline{A}.\ A^?\, x \Rightarrow \mathsf{Relat}_B[\bar{t}\ x =_{\overline{B}} \overline{t'}\ x]$ of the conclusion holds. Since the formula in this relativization is almost proper, it follows that $\Gamma \vdash_T \lambda x : A.\ t =_{\Pi x:A.\ B} \lambda x : A'.\ t'$ is well-typed.

By rule (eqTyping) and rule (sym) $\Gamma \vdash_T \lambda x : A'.\, t' : \Pi x : A.\, B$ holds in DIHOLP. This is only provable if $\Gamma \vdash_T A \equiv A'$. Consequently, the relativization $\overline{\Gamma},\ x : \overline{A},\ A^?\ x \vdash^H_{\overline{T}} \mathsf{Relat}_B[\bar{t}\ x =_{\overline{B}} \overline{t'}\ x]$ of the validity assumption is proper and thus the assumption must be as well. Clearly, $\Gamma,\ x : A \vdash_T t =_B t'$ is a quasi-preimage of the validity assumption. It follows:

$$\overline{\Gamma},\ x : \overline{A} \vdash^H_{\overline{T}} \bar{t} =_{\overline{B}} \overline{t'} \qquad \text{by assumption} \qquad (586)$$

$$\overline{\Gamma},\ x : \overline{A},\ A^?\ x \vdash^H_{\overline{T}} \bar{t} =_{\overline{B}} \overline{t'} \qquad (\text{monotonic}\vdash),(586) \qquad (587)$$

$$\Gamma,\ x : A \vdash_T t =_B t' \qquad \text{induction hypothesis},(587) \qquad (588)$$

$$\Gamma \vdash_T A \equiv A' \qquad \text{see above} \qquad (589)$$

$$\Gamma \vdash_T \lambda x : A.\, t =_{\Pi x : A.\ B} \lambda x : A'.\, t' \qquad (\text{cong}\lambda\text{'}),(589) \qquad (590)$$

(eta): Since the conclusion is proper, it follows that the relativization

$$\overline{\Gamma} \vdash^H_{\overline{T}} \mathsf{Relat}_{\Pi x : A.\ B}[\bar{t} =_{\overline{\Pi x : A.\ B}} \lambda x : \overline{A}\bar{t}\ x.\ ]$$

of the conclusion holds. The preimage of the formula in this relativization must therefore be a well-typed equality and hence

$$\Gamma \vdash_T t =_{\Pi x : A.\ B} \lambda x : A.\, t\ x : \mathsf{bool}.$$

It follows:

$$\Gamma \vdash_T t =_{\Pi x : A.\ B} \lambda x : A.\, t\ x : \mathsf{bool} \qquad \text{see above} \qquad (591)$$

$$\Gamma \vdash_T t =_{\Pi x : A.\ B} \lambda x : A.\, t\ x \qquad (\text{etaPi}),(591) \qquad (592)$$

(congAppl): Since the conclusion $\overline{\Gamma} \vdash^H_{\overline{T}} \overline{f}\ \bar{t} =_{\overline{B}} \overline{f'}\ \overline{t'}$ is proper, it follows that its relativization $\overline{\Gamma} \vdash^H_{\overline{T}} \mathsf{Relat}_B[\overline{f}\ \bar{t} =_{\overline{B}} \overline{f'}\ \overline{t'}]$ holds and the preimage $f\ t =_B f'\ t'$ of the formula in this relativization must be well-typed i.e. $\Gamma \vdash_T f\ t =_B f'\ t'$.

Since the validity assumption use the same context as the conclusion, it follows that they are both proper. As observed in the beginning of the proof if a proper assumption of a rule is an equality over a type $\overline{A}$, the induction hypothesis implies that the quasi-preimage of that assumption in which the equality is over type A must hold. Hence, we have both $\Gamma \vdash_T t =_A t'$ and $\Gamma \vdash_T f =_{\Pi x : A.\ B} f'$ in DIHOLP.

It follows:

$$\Gamma \vdash_T t =_{A'} t' \qquad \text{see above} \qquad (593)$$

$$\Gamma \vdash_T f =_{\Pi x : A'.\ B} f' \qquad \text{see above} \qquad (594)$$

$$\Gamma \vdash_T f\ t =_B f'\ t' \qquad (\text{congAppl}),(593),(594) \qquad (595)$$

This is what we had to show.

(cong$\vdash$): Since the conclusion $\overline{\Gamma} \vdash^H_{\overline{T}} \overline{F}$ is proper it follows that its quasi-preimage is well-typed i.e. $\Gamma \vdash_T F : \mathsf{bool}$. Since the validity assumptions use tha same context as the conclusion, which is proper, it follows that they are also proper. The first assumption $\overline{\Gamma} \vdash^H_{\overline{T}} \varphi =_{\mathsf{bool}} \overline{F}$ of the rule therefore implies that $\Gamma \vdash_T F' =_{\mathsf{bool}} F$, where F' denotes the preimage of $\mathsf{RP}[\mathsf{R}[\varphi]]$. In particular, it follows that the (validity) assumptions of the rule are proper. By the induction hypothesis applied to the second assumption, we get $\Gamma \vdash_T F'$. The desired result of $\Gamma \vdash_T F$ then follows from rule (cong$\vdash$).

(beta): Since the conclusion is proper, it follows that the relativization $\overline{\Gamma} \vdash^H_{\overline{T}} \mathsf{Relat}_B[\left(\lambda x : \overline{A}.\ s\right)\ t =_{\overline{B}} \overline{s}[^x/_t]]$ of the conclusion holds. The preimage of the formula in this relativization must therefore be a well-typed equality and hence

$$\Gamma \vdash_T \left(\lambda x : A.\ s\right)\ t =_{\Pi x:A.\ B}\ s[^x/_t] : \mathsf{bool}.$$

It follows:

$$\Gamma \vdash_T \left(\lambda x : A.\ s\right)\ t =_{\Pi x:A.\ B}\ s[^x/_t] : \mathsf{bool} \qquad \text{see above} \qquad (596)$$

$$\Gamma \vdash_T \left(\lambda x : A.\ s\right)\ t =_{\Pi x:A.\ B}\ s[^x/_t] \qquad \text{(beta),(596)} \qquad (597)$$

(refl): Since the conclusion $\overline{\Gamma} \vdash^H_{\overline{T}} \overline{t} =_{\overline{A}} \overline{t}$ of the rule is proper, it follows that its relativization $\overline{\Gamma} \vdash^H_{\overline{T}} \mathsf{Relat}_A[\overline{t} =_{\overline{A}} \overline{t}]$ must hold. Since the preimage of the formula in this relativization must be well-typed it follows that $\Gamma \vdash_T t =_A t : \mathsf{bool}$. Rule (eqTyping) thus implies that $\Gamma \vdash_T t : A$. The claim of $\Gamma \vdash_T t =_A t$, follows by rule (refl) and termwise injectivity of the translation (shown in Lemma 7).

(sym): Since the assumption $\overline{\Gamma} \vdash^H_{\overline{T}} \overline{s} =_{\overline{A}} \overline{t}$ of the rule is proper, so must be the assumption $\overline{\Gamma} \vdash^H_{\overline{T}} \overline{t} =_{\overline{A}} \overline{s}$. Obviously, $\Gamma \vdash_T t =_A s$ is the right quasi-preimage of the assumption. Thus:

$$\overline{\Gamma} \vdash^H_{\overline{T}} \overline{t} =_{\overline{A}} \overline{s} \qquad \text{by assumption} \qquad (598)$$

$$\Gamma \vdash_T t =_A s \qquad \text{induction hypothesis,(598)} \qquad (599)$$

$$\Gamma \vdash_T s =_A s \qquad \text{(sym),(599)} \qquad (600)$$

(assume): Since the conclusion $\overline{\Gamma} \vdash^H_{\overline{T}} \overline{F}$ is proper it follows that $\Gamma \vdash_T F : \mathsf{bool}$ and hence by rule (typingTp) followed by rule (tpCtx) it follows that $\vdash_T \Gamma\ \mathsf{Ctx}$. By definition of the translation function (particularly the cases PT7 and IT7) an assumption $\overline{F}$ occurs in $\overline{\Gamma}$ iff an axiom F occurs in T. Hence by rule (assume) the conclusion of $\Gamma \vdash_T F$ follows.

(axiom): Since the conclusion $\overline{\Gamma} \vdash^H_{\overline{T}} \overline{F}$ is proper it follows that $\Gamma \vdash_T F : \mathsf{bool}$ and hence by rule (typingTp) followed by rule (tpCtx) it follows that $\vdash_T \Gamma\ \mathsf{Ctx}$. By definition of the translation function (particularly the cases PT4 and IT4) an axiom $\overline{F}$ occurs in $\overline{T}$ iff an axiom F occurs in T. Hence by rule (axiom) the conclusion of $\Gamma \vdash_T F$ follows.

(⇒I): Since the conclusion $\overline{\Gamma} \vdash^H_{\overline{T}} \overline{F} \Rightarrow \overline{G}$ is proper it follows that $\Gamma \vdash_T F \Rightarrow G : \mathsf{bool}$.

Thus by Lemma 24 it follows that $\Gamma \vdash_T F : \mathsf{bool}$ and $\Gamma,\ F \vdash^H_{\overline{T}} G : \mathsf{bool}$ both hold. Consequently, the validity assumption $\overline{\Gamma},\ \overline{F} \vdash^H_{\overline{T}} \overline{G}$ is proper with preimage (and hence quasi-preimage) $\Gamma,\ F \vdash_T G$. By induction hypothesis this preimage of the validity assumption must hold. Rule (⇒I), then implies the desired result of $\Gamma \vdash_T F \Rightarrow G$.

(⇒E): Since the conclusion $\overline{\Gamma} \vdash^H_{\overline{T}} \overline{G}$ is proper it follows that $\Gamma \vdash_T G : \mathsf{bool}$. Since the formula $\mathsf{RP}\left[\mathsf{R}[F']\right]$ (where $\overline{\Gamma} \vdash^H_{\overline{T}} F'$ is the second validity assumption) must be proper, it follows that its preimage F is well-typed i.e. $\Gamma \vdash_T F : \mathsf{bool}$. By rule (⇒type') this implies that also $\Gamma \vdash_T F \Rightarrow G : \mathsf{bool}$. By Lemma 24 we yield both $\Gamma \vdash_T F : \mathsf{bool}$ and $\Gamma,\ F \vdash^H_{\overline{T}} G : \mathsf{bool}$. Thus, both validity assumptions are proper with the obvious quasi-preimages. By induction hypothesis, it follows that $\Gamma \vdash_T F$ and $\Gamma,\ F \vdash_T G$ both hold. Rule (⇒E) then allows us to conclude the claim of $\Gamma \vdash_T G$.

(propExt): Since by assumption the conclusion $\overline{\Gamma} \vdash^H_{\overline{T}} \overline{F} =_{\mathsf{bool}} \overline{G}$ is proper, it follows by rule (eqTyping) (and rule (sym)) that $\Gamma \vdash_T F : \mathsf{bool}$ and $\Gamma \vdash_T G : \mathsf{bool}$ and hence the assumptions are also proper. Thus:

$\overline{\Gamma},\ \overline{F} \vdash^H_{\overline{T}} \overline{G}$	by assumption	(601)
$\overline{\Gamma},\ \overline{G} \vdash^H_{\overline{T}} \overline{F}$	by assumption	(602)
$\Gamma,\ F \vdash_T G$	induction hypothesis,(601)	(603)
$\Gamma,\ G \vdash_T F$	induction hypothesis,(602)	(604)
$\Gamma \vdash_T F =_{\mathsf{bool}} G$	(propExt),(603),(604)	(605)

□

N Proof of Lemma 27 about Translation 2

Proof of Lemma 27. Since beta and eta reductions don't affect typing statements, we can wlog. assume that F is beta and eta reduced. If F is the application of a λ-function to an argument that is ill-typed or of wrong type, then by definition $\varphi = \mathsf{false}$ and there is nothing to prove. Otherwise, we know that F must be an equality, an implication, a constant or variable or the application $p\ t_1 \ldots\ t_n$ of a constant or variable p to (all its) arguments.

We proceed by induction on the shape of F. If F is an equality, then rule (=type) reduces the claim to the result of Lemma 20 (for arguments t_i not of type bool) and to the induction hypothesis for arguments of type bool. If F is an implication, then rule (⇒type) reduces the claim to the induction hypothesis. Finally if F is the application $p\ t_1 \ldots\ t_n$ of a constant or variable p to (all its) arguments, the abbreviation abbreviates $p_{A_1}\ t_1 \wedge \ldots \wedge p_{A_n}$, where the A_i are the types of the arguments in the Π-type of p. Using Lemma 20 for i with A_i not (a subtype of)

the type bool and the induction hypothesis for the remaining i we yield $\Gamma \vdash_T t_i : A_i$ for all i. The claim then follows by (repeated application of) rule (appl'). □

O Proof that HOL is a conservative extension of HOL*

Definition 14. We define the logic fragment of HOL without quantification of order n of bool-valued function types (denoted by HOLF^{n}n) as the fragment of HOL in which we explicitly disallow terms of type A for types A of the form $(((\ldots(C \to B_1) \to B_2) \to \ldots \to B_m)$ with C a bool-valued function type and $m > n$, except in context variables where we even disallow such types A for $m \geq n$.

Definition 15. We want to define a proof transformation that takes a valid HOLF^{n+1} derivation relative to a HOL* theory and of a HOL* conjecture and yields a valid HOL* derivation relative to the same theory of the same conjecture. Let n and the derivation in HOLF^{n+1} be given.

We define the *order reducing* statement transformation of order n (denoted by $\mathsf{ORed}^n(\cdot)$) to be the statement transformation in HOLF^{n+1} which replaces terms and types in statements as follows

$\mathsf{ORed}^n(t)$	$:= t$	t in HOLF^n	(OR1)
$\mathsf{ORed}^n(s =_A t)$	$:= \mathsf{ORed}^n(s) =_{\mathsf{ORed}^n(A)} \mathsf{ORed}^n(t)$		(OR2)
$\mathsf{ORed}^n(F \Rightarrow G)$	$:= \mathsf{ORed}^n(F) \Rightarrow \mathsf{ORed}^n(G)$		(OR3)
$\mathsf{ORed}^n((\lambda x : A.\ f)\ t)$	$:= \mathsf{ORed}^n(f)\,[x/\mathsf{ORed}^n(t)]$	$\lambda x : A.\ f$ not in HOLF^n	(OR4)
$\mathsf{ORed}^n(f\ t)$	$:= \mathsf{ORed}^n(f)\ \mathsf{ORed}^n(t)$	otherwise	(OR5)
$\mathsf{ORed}^n(\lambda x : A.\ f)$	$:= \mathsf{ORed}^n(f)$	$\lambda x : A.\ f$ not in HOLF^n	(OR6)
$\mathsf{ORed}^n(\lambda x : A.\ f)$	$:= \lambda x : \mathsf{ORed}^n(A).\ \mathsf{ORed}^n(f)$	otherwise	(OR7)
$\mathsf{ORed}^n(A)$	$:= A$	type A in HOLF^n	(OR8)
$\mathsf{ORed}^n((\ldots(C \to B_1) \to \ldots \to B_{n+1}))$	$:= B_{n+1}$	else	(OR9)

Secondly, we add a fresh free variable $x : A$ to contexts in the conclusions of rules whose order-reducing statement transformation has a bound variable x of type A for A not in HOLF^n. Whenever we add a context variable to one of the assumptions of a rule, we will add it also to contexts of the other assumptions (and their derivations). Observe that this still cannot lead to additional context variables for the context of the final conclusion in the entire derivation.

We now need to define a macro-step for each step S in the resulting derivation for the order-reducing statement transformation of order n replacing step S.

For the most part we can pick the macro-steps to be identical to the step they replace. However, we have to use rule (refl) instead of a congruence rule in some steps, remove some steps using rule (lambda) entirely and have to replace steps using rule (appl) by steps using rule (rewrite-Typing) (or rather the steps in the proof of the rule in our case as a macro-step) for function applications outside of HOLF^n. Finally, we need to replace steps using the rule (congAppl) for

applications outside of HOLF^n by a macro-step often consisting of several micro-steps whose micro-steps all use congruence rules. We don't care in which order those congruence rules are applied and will consider the resulting transformation to be an order-reducing proof transformation in any case. Observe that all those (macro-)steps will (have micro-steps that) lie (modulo the variable declarations in the context) inside HOLF^n.

The resulting proof transformation will be called the order-reducing proof transformation of order n. Notice that applying the order reducing proof transformation of order n to a derivation in HOLF^{n+1} for a theory and conjecture in HOL* will result in a derivation whose context assumptions, terms and types within assumptions and conclusions of steps all lie in HOLF^n. Since the final conclusion lies in HOL* and we cannot remove free variables outside of HOLF^n (without creating bound variables of same type) from contexts in the derivation, it follows that the resulting derivation lies in HOLF^n. The application on term and type-level of the order reducing proof transformation of order n, will be called the order reducing reduction of order n.

Lemma 34. *Given a valid HOLF^{n+1} derivation relative to a HOL* theory for a HOL* conjecture, there exists an order reducing proof transformation of order n whose application to the derivation is a valid derivation.*

Proof. We will prove this by induction on the inference rules of HOL.

If the proof transformation modifies types at all, it is my replacing types $A \to B$ by types B and the former can only be proven to be well-typed (by rule (arrow)) once the latter is already known to be well-typed. Therefore the cases of the rules for well-formedness of types and contexts are trivial (and the proof transformation doesn't affect the theories at all, so we don't have to prove anything about the well-formedness of theories).

Firstly, we observe that there are no dependent types in IHOL and the theory lies in HOL*. Furthermore all types are translated to either to themselves or to their codomain in case of certain function types. As the well-formedness of the original types imply the well-formedness of the later (and similarly for type equalities between two such function types), it follows that the well-formedness (of theories, contexts, types) and type-equality judgements are unaffected by the order reducing proof transformation of order n (or any other order). It remains to consider the typing and validity rules and to show that if the order reducing reductions of the assumptions hold, then the order reducing reduction of the conclusion holds:

(const): Since constants lie in HOL*, there is nothing to prove.

(var): Since the types of context variables are not modified by the proof transformation:

$$\vdash^H_T x : A \text{ in } \mathsf{ORed}^n(\Delta) \qquad \text{by assumption} \qquad (606)$$

$$\vdash^H_T \mathsf{ORed}^n(\Delta)\ \mathsf{Ctx} \qquad \text{by assumption} \qquad (607)$$

$$\Delta \vdash^H_T x : A \qquad \text{(var),(606),(607)} \qquad (608)$$

(=type):

$$\Delta \vdash^H_T \mathsf{ORed}^n(s) : \mathsf{ORed}^n(A) \quad \text{by assumption} \quad (609)$$

$$\Delta \vdash^H_T \mathsf{ORed}^n(t) : \mathsf{ORed}^n(A) \quad \text{by assumption} \quad (610)$$

$$\Delta \vdash^H_T \mathsf{ORed}^n(s) =_{\mathsf{ORed}^n(A)} \mathsf{ORed}^n(t) : \mathsf{bool} \quad (=\text{type}),(609),(610) \quad (611)$$

$$\Delta \vdash^H_T \mathsf{ORed}^n\left(s =_A t\right) : \mathsf{bool} \quad \text{OR2},(611) \quad (612)$$

(lambda):

$$\Delta,\ x : A \vdash^H_T \mathsf{ORed}^n(t) : \mathsf{ORed}^n(B) \quad \text{by assumption} \quad (613)$$

if $\mathsf{ORed}^n(A)$ in HOLF^{n-1}:

$$\Delta \vdash^H_T (\lambda x : A.\ \mathsf{ORed}^n(t)) : A \to \mathsf{ORed}^n(B) \quad (\text{lambda}),(613) \quad (614)$$

$$\Delta \vdash^H_T \mathsf{ORed}^n(\lambda x : A.\ \mathsf{ORed}^n(t) : \mathsf{ORed}^n(A \to B)) \quad \text{OR1,OR8},(614) \quad (615)$$

otherwise:

$$\Delta,\ x : A \vdash^H_T \mathsf{ORed}^n(\lambda x : A.\ \mathsf{ORed}^n(t)) : \mathsf{ORed}^n(A \to \mathsf{ORed}^n(B)) \quad \text{OR6,OR9},(613) \quad (616)$$

(appl):

$$\Delta \vdash^H_T \mathsf{ORed}^n(f) : \mathsf{ORed}^n(A \to B) \quad \text{by assumption} \quad (617)$$

$$\Delta \vdash^H_T \mathsf{ORed}^n(t) : \mathsf{ORed}^n(A) \quad \text{by assumption} \quad (618)$$

If $A \to B$ in HOLF^n:

$$\Delta \vdash^H_T \mathsf{ORed}^n(f)\ \mathsf{ORed}^n(t) : \mathsf{ORed}^n(B) \quad (\text{lambda}),\text{OR8},(617),(618) \quad (619)$$

$$\Delta \vdash^H_T \mathsf{ORed}^n(f\ t) : \mathsf{ORed}^n(B) \quad \text{OR5},(619) \quad (620)$$

Otherwise we have $f =: \lambda x : \mathsf{ORed}^n(A).\ \mathsf{ORed}^n(s)$, A not in HOLF^{n-1}, so by OR6, OR9 and OR8 we have $\mathsf{ORed}^n(f) = \mathsf{ORed}^n(s)$ and $\mathsf{ORed}^n(A \to B) = B = \mathsf{ORed}^n(B)$ and $\mathsf{ORed}^n(A) = A$:

$$\Delta \vdash^H_T \mathsf{ORed}^n(s) : \mathsf{ORed}^n(B) \quad \text{OR6,OR9,OR8},(617) \quad (621)$$

$$\Delta \vdash^H_T \mathsf{ORed}^n(s)\left[{}^{x}/_{\mathsf{ORed}^n(t) : \mathsf{ORed}^n(B)}\right] \quad (\text{rewriteTyping}),(621),(619) \quad (622)$$

Observe that by induction hypothesis the derivations of $\Delta \vdash^H_T \mathsf{ORed}^n(s) : \mathsf{ORed}^n(B)$ and of $\Delta \vdash^H_T \mathsf{ORed}^n(t) : \mathsf{ORed}^n(A)$ lie in HOLF^n. Consequently the steps in the derivation which we get when plugging in the proof of rule (rewriteTyping) in this case will also lie in HOLF^n(as the terms in it for corresponding steps have the same types as in the derivation for $\Delta \vdash^H_T \mathsf{ORed}^n(s) : \mathsf{ORed}^n(B)$ and steps in the derivation of $\Delta \vdash^H_T \mathsf{ORed}^n(t) : \mathsf{ORed}^n(A)$ which occur in this derivation are unchanged).

(⇒type): Observe that $\mathsf{ORed}^n(\mathsf{bool}) = \mathsf{bool}$.

$$\Delta \vdash^H_{\overline{T}} \mathsf{ORed}^n(F) : \mathsf{bool} \quad \text{by assumption} \quad (623)$$
$$\Delta \vdash^H_{\overline{T}} \mathsf{ORed}^n(G) : \mathsf{bool} \quad \text{by assumption} \quad (624)$$
$$\Delta \vdash^H_{\overline{T}} \mathsf{ORed}^n(F) \Rightarrow \mathsf{ORed}^n(G) : \mathsf{bool} \quad (\Rightarrow\text{type}),(623),(624) \quad (625)$$
$$\Delta \vdash^H_{\overline{T}} \mathsf{ORed}^n(F \Rightarrow F) : \mathsf{bool} \quad \text{OR3},(625) \quad (626)$$

(axiom): Since axioms always lie in HOL*, there is nothing to prove here.

(assume):

$$\mathsf{ORed}^n(F) \text{ in } \overline{T} \quad (627)$$
$$\Delta \vdash^H_{\overline{T}} \mathsf{ORed}^n(F) \quad (\text{assume}),(627) \quad (628)$$

(congλ):

$$\Delta \vdash^H_{\overline{T}} \mathsf{ORed}^n(A) \equiv \mathsf{ORed}^n(A') \quad \text{by assumption} \quad (629)$$
$$\Delta, x : A \vdash^H_{\overline{T}} \mathsf{ORed}^n(t =_B t') \quad \text{by assumption} \quad (630)$$
$$\Delta, x : A \vdash^H_{\overline{T}} \mathsf{ORed}^n(t) =_{\mathsf{ORed}^n(B)} \mathsf{ORed}^n(t') \quad \text{OR2},(630) \quad (631)$$

If A, A' in HOLF^{n-1}:

$$\Delta \vdash^H_{\overline{T}} \lambda x : \mathsf{ORed}^n(A).\, \mathsf{ORed}^n(t) =_{\mathsf{ORed}^n(A)\to\mathsf{ORed}^n(B)} \lambda x : \mathsf{ORed}^n(A).\, \mathsf{ORed}^n(t') \quad (\text{cong}\lambda),(629),(631) \quad (632)$$
$$\Delta \vdash^H_{\overline{T}} \mathsf{ORed}^n(\lambda x : A.\, t =_{A\to B} \lambda x : A.\, t') \quad \text{OR7,OR8,OR2, (632)} \quad (633)$$

Since $A \equiv A'$ we neither are in HOLF^{n-1} and thus the claim is exactly the assumption (630).

(congAppl):

$$\Delta \vdash^H_{\overline{T}} \mathsf{ORed}^n(t =_A t') \quad \text{by assumption} \quad (634)$$
$$\Delta \vdash^H_{\overline{T}} \mathsf{ORed}^n(t) =_{\mathsf{ORed}^n(A)} \mathsf{ORed}^n(t') \quad \text{OR2},(634) \quad (635)$$
$$\Delta \vdash^H_{\overline{T}} \mathsf{ORed}^n(f =_{A\to B} f') \quad \text{by assumption} \quad (636)$$
$$\Delta \vdash^H_{\overline{T}} \mathsf{ORed}^n(f) =_{\mathsf{ORed}^n(A\to B)} \mathsf{ORed}^n(f') \quad \text{OR2},(636) \quad (637)$$

If $A \to B$ in HOLF^n:

$$\Delta \vdash^H_{\overline{T}} \mathsf{ORed}^n(f)\ \mathsf{ORed}^n(t) =_{\mathsf{ORed}^n(B)} \mathsf{ORed}^n(f')\ \mathsf{ORed}^n(t') \quad \text{OR8},(\text{congAppl}),(635),(637) \quad (638)$$

$$\Delta \vdash^H_T \mathsf{ORed}^n\left(f\ t =_{A\to B} f'\ t'\right) \qquad \text{OR5,OR2,(638)} \qquad (639)$$

Otherwise f and f' are of the form $\lambda x : A.\ s$ and $\lambda x : A.\ s'$ and the context of the assumptions contains a variable $x : A$ at the very end. Let Δ' denote the context without this variable declaration. Then the second assumption simplifies to:

$$\Delta \vdash^H_T \mathsf{ORed}^n(s) =_{\mathsf{ORed}^n(B)} \mathsf{ORed}^n\left(s'\right) \qquad \text{OR6,OR9,(637)} \qquad (640)$$

We know that $\mathsf{ORed}^n(s), \mathsf{ORed}^n\left(s'\right), \mathsf{ORed}^n(B)$ in HOLF^n. Thus we can use the congruence rules for equality, λ-functions, function application and implication (namely (trans), (congλ), (congAppl), ($\Rightarrow$cong)) to conclude by induction that

$$\Delta' \vdash^H_T \mathsf{ORed}^n(s)\left[x/_{\mathsf{ORed}^n(t)}\right] =_B \mathsf{ORed}^n\left(s'\right)\left[x/_{\mathsf{ORed}^n(t)}\right]$$

and except possibly for free variable this derivation will lie in HOLF^n.

$$\Delta' \vdash^H_T \mathsf{ORed}^n(s)\left[x/_{\mathsf{ORed}^n(t)}\right] =_B \mathsf{ORed}^n\left(s'\right)\left[x/_{\mathsf{ORed}^n(t')}\right] \qquad \text{see above}$$

Since x doesn't appear in the substitutions, the context in the desired statement is Δ' and the next step concludes the desired statement:

$$\Delta' \vdash^H_T \mathsf{ORed}^n\left(\mathsf{ORed}^n(s)\left[x/_{\mathsf{ORed}^n(t)}\right] =_{\mathsf{ORed}^n(B)} \mathsf{ORed}^n\left(s'\right)\left[x/_{\mathsf{ORed}^n(t')}\right]\right) \qquad \text{OR2,(641)} \qquad (641)$$

(refl):

$$\Delta \vdash^H_T \mathsf{ORed}^n(t) : \mathsf{ORed}^n(A) \qquad \text{by assumption} \qquad (642)$$
$$\Delta \vdash^H_T \mathsf{ORed}^n(t) =_{\mathsf{ORed}^n(A)} \mathsf{ORed}^n(t) \qquad \text{(refl),(642)} \qquad (643)$$
$$\Delta \vdash^H_T \mathsf{ORed}^n\left(t =_A t\right) \qquad \text{OR2,(643)} \qquad (644)$$

(sym):

$$\Delta \vdash^H_T \mathsf{ORed}^n\left(t =_A s\right) \qquad \text{by assumption} \qquad (645)$$
$$\Delta \vdash^H_T \mathsf{ORed}^n(t) =_{\mathsf{ORed}^n(A)} \mathsf{ORed}^n(s) \qquad \text{OR2,(645)} \qquad (646)$$
$$\Delta \vdash^H_T \mathsf{ORed}^n(s) =_A \mathsf{ORed}^n(t) \qquad \text{(sym),(646)} \qquad (647)$$
$$\Delta \vdash^H_T \mathsf{ORed}^n\left(s =_A t\right) \qquad \text{OR2,(647)} \qquad (648)$$

(beta):

$$\Delta \vdash^H_T \mathsf{ORed}^n((\lambda x : A.\ s)\ t) : \mathsf{ORed}^n(B) \qquad \text{by assumption} \qquad (649)$$

If $(\lambda x : A.\ \mathsf{ORed}^n(s))\ \mathsf{ORed}^n(t)$ is in HOLF^n:

$$\Delta \vdash^H_T (\lambda x : A.\ \mathsf{ORed}^n(s))\ \mathsf{ORed}^n(t) : \mathsf{ORed}^n(B) \qquad \text{OR2,OR5,OR7,(649)} \qquad (650)$$
$$\Delta \vdash^H_T (\lambda x : A.\ \mathsf{ORed}^n(s))\ \mathsf{ORed}^n(t) =_B \mathsf{ORed}^n(s)\,[{}^x/_{\mathsf{ORed}^n(t)}] \qquad \text{(beta),(650)} \qquad (651)$$
$$\Delta \vdash^H_T \mathsf{ORed}^n\left((\lambda x : A.\ s)\ t =_A s[{}^x/_t]\right) \qquad \text{OR5,OR7,OR2,(651)} \qquad (652)$$

Otherwise we have

$$\mathsf{ORed}^n((\lambda x : A.\ s)\ t) = \mathsf{ORed}^n(s)\,[{}^x/_{\mathsf{ORed}^n(t)}] = \mathsf{ORed}^n(s[{}^x/_t])$$

and thus:

$$\Delta \vdash^H_T \mathsf{ORed}^n((\lambda x : A.\ s)\ t) =_A \mathsf{ORed}^n(s[{}^x/_t]) \qquad \text{(refl),(649)} \qquad (653)$$
$$\Delta \vdash^H_T \mathsf{ORed}^n\left((\lambda x : A.\ s)\ t =_A s[{}^x/_t]\right) \qquad \text{OR2,(653)} \qquad (654)$$

Note that in either case we only applied a single rule either rule (beta) or rule (refl) once, in the other of the above steps we are just rewriting statements using the definition of the order-reducing reduction.

(eta):

$$\Delta \vdash^H_T \mathsf{ORed}^n(t) : \mathsf{ORed}^n(A \to B) \qquad \text{by assumption} \qquad (655)$$
$$x \text{ not in } \Gamma \qquad \text{by assumption} \qquad (656)$$

If $A \to B$ in HOLF^n:

$$\Delta \vdash^H_T \mathsf{ORed}^n(t) =_{A\to B} \lambda x : A.\ \mathsf{ORed}^n(t)\ x \qquad \text{OR8,(eta),(655),(656)} \qquad (657)$$
$$\Delta \vdash^H_T \mathsf{ORed}^n\left(t =_{A\to B} \lambda x : A.\ t\ x\right) \qquad \text{OR7,OR2,(657)} \qquad (658)$$

Otherwise we have $\mathsf{ORed}^n(A \to B) = B = \mathsf{ORed}^n(B)$ and t is of the form $\lambda x : A.\ f$ with $\mathsf{ORed}^n(t) = \mathsf{ORed}^n(f)$ and thus:

$$\Delta \vdash^H_T \mathsf{ORed}^n(f) : \mathsf{ORed}^n(B) \qquad \text{OR6,OR9,OR8,(655)} \qquad (659)$$
$$\Delta \vdash^H_T \mathsf{ORed}^n(f) =_{\mathsf{ORed}^n(B)} \mathsf{ORed}^n(t) \qquad \text{(refl),(659)} \qquad (660)$$
$$\Delta \vdash^H_T \mathsf{ORed}^n(f) =_{\mathsf{ORed}^n(B)} \mathsf{ORed}^n(\lambda x : A.\ t\ x) \qquad \text{OR6,(660)} \qquad (661)$$
$$\Delta \vdash^H_T \mathsf{ORed}^n\left(t =_{A\to B} \lambda x : A.\ t\ x\right) \qquad \text{OR2,(661)} \qquad (662)$$

Just as for rule (beta), we used exactly one rule, either rule (eta) or rule (refl) exactly once.

Since $\mathsf{ORed}^n(\mathsf{bool}) = \mathsf{bool}$, we simply write bool instead of $\mathsf{ORed}^n(\mathsf{bool})$ in the cases for the remaining 4 rules:

(cong$\vdash$):

$$\Delta \vdash^{H}_{T} \mathsf{ORed}^{n}\left(F =_{\mathsf{bool}} F'\right) \quad \text{by assumption} \quad (663)$$
$$\Delta \vdash^{H}_{T} \mathsf{ORed}^{n}\left(F'\right) \quad \text{by assumption} \quad (664)$$
$$\Delta \vdash^{H}_{T} \mathsf{ORed}^{n}(F) =_{\mathsf{bool}} \mathsf{ORed}^{n}\left(F'\right) \quad \text{OR2,(663)} \quad (665)$$
$$\Delta \vdash^{H}_{T} \mathsf{ORed}^{n}(F) \quad \text{(cong}\vdash\text{),(665),(664)} \quad (666)$$

($\Rightarrow$I):

$$\Delta \vdash^{H}_{T} \mathsf{ORed}^{n}(F) : \mathsf{bool} \quad \text{by assumption} \quad (667)$$
$$\Delta, \mathsf{ORed}^{n}(F) \vdash^{H}_{T} \mathsf{ORed}^{n}(G) \quad \text{by assumption} \quad (668)$$
$$\Delta \vdash^{H}_{T} \mathsf{ORed}^{n}(F) \Rightarrow \mathsf{ORed}^{n}(G) \quad (\Rightarrow\text{I}),(667),(668) \quad (669)$$
$$\Delta \vdash^{H}_{T} \mathsf{ORed}^{n}(F \Rightarrow G) \quad \text{OR3,(669)} \quad (670)$$

($\Rightarrow$E):

$$\Delta \vdash^{H}_{T} \mathsf{ORed}^{n}(F \Rightarrow G) \quad \text{by assumption} \quad (671)$$
$$\Delta \vdash^{H}_{T} \mathsf{ORed}^{n}(F) \quad \text{by assumption} \quad (672)$$
$$\Delta \vdash^{H}_{T} \mathsf{ORed}^{n}(F) \Rightarrow \mathsf{ORed}^{n}(G) \quad \text{OR3,(671)} \quad (673)$$
$$\Delta \vdash^{H}_{T} \mathsf{ORed}^{n}(G) \quad (\Rightarrow\text{E}),(673),(672) \quad (674)$$

(boolExt):

$$\Delta \vdash^{H}_{T} \mathsf{ORed}^{n}(p\ \mathsf{true}) \quad \text{by assumption} \quad (675)$$
$$\Delta \vdash^{H}_{T} p\ \mathsf{true} \quad \text{OR1,(675)} \quad (676)$$
$$\Delta \vdash^{H}_{T} \mathsf{ORed}^{n}(p\ \mathsf{false}) \quad \text{by assumption} \quad (677)$$
$$\Delta \vdash^{H}_{T} p\ \mathsf{false} \quad \text{OR1,(677)} \quad (678)$$
$$\Delta \vdash^{H}_{T} \forall x : \mathsf{bool}.\ p\ x \quad \text{(boolExt),(676),(678)} \quad (679)$$
$$\Delta \vdash^{H}_{T} \mathsf{ORed}^{n}(\forall x : \mathsf{bool}.\ p\ x) \quad \text{OR1,(679)} \quad (680)$$

□

Proof of Lemma 29. This lemma follows directly from induction and Lemma 34. □

Remark 35. Lemma 34 is a very useful lemma and can also be used to show for instance that HOL is a conservative extension of first-order logic.