From Dependent Type Theory to Dependently Typed Higher Order Logic (Technical Report)

Luca Maio[1], Alexander Bentkamp[2], Jasmin Blanchette[1], and Sophie Tourret[3,4]

[1] Ludwig-Maximilians-Universität München, Munich, Germany
{luca.maio,jasmin.blanchette}@ifi.lmu.de
[2] Cryspen, Paris, France
alex@cryspen.com
[3] Inria, LORIA, CNRS, Université de Lorraine, Nancy, France
sophie.tourret@inria.fr
[4] Max-Planck-Institut für Informatik, Saarland Informatics Campus, Saarbrücken, Germany
sophie.tourret@mpi-inf.mpg.de

Abstract. There is an expressivity gap between proof assistants based on dependent type theory (DTT), such as Lean and Rocq, and automated theorem provers based on higher order logic (HOL). This gap has been reducedd with the introduction of dependently typed HOL (DHOL), along with automated theorem proving for DHOL. To reduce the gap even further, we introduce a translation from a fragment of DTT to polymorphic DHOL. Our translation is sound and lightweight, preserving constructs such as polymorphism and dependent types, and could be used in a hammer system.

1 Introduction

Proof assistants such as Lean [18], Matita [3] and Rocq [5] are based on rich dependent type theories (DTTs) [19], whereas most state-of-the-art automated theorem provers are based on first order logic (FOL) or higher order logic (HOL) [14]. This makes it difficult to develop hammer systems [7] for these proof assistants.

However, the expressivity gap between dependently typed proof assistants and automated provers has shrunk with the emergence of dependently typed HOL (DHOL) [25, 28], along with DHOL specific automated theorem proving techniques. In particular, the Lash [20] system is an extension of the HOL based Satallax [8] tableau prover to DHOL. Moreover, there exists a sound and complete translation from DHOL to HOL [25, 28], which can be used in a hammer.

Nevertheless, there is still a substantial gap between DTT (Sect. 2) and DHOL (Sect. 3). Both are based on λ-calculi and support types that depend on terms. Examples of dependent types include the integers modulo n and the

vectors of length n. A distinguishing feature of traditional DTT is its reliance on the Curry-Howard correspondence: propositions are represented by types, and proofs are represented by terms. Types themselves have types, which are called universes. Even universes can have types, forming a possibly infinite universe hierarchy. DTT is fundamentally intuitionistic but can be extended with classical axioms. The DTTs implemented by Lean, Matita and Rocq are intensional: they use a weak, computational notion of equality for checking types, ensuring that type checking is decidable.

By contrast, DHOL is an extension of classical HOL, which itself is a polymorphic generalisation of Church's simple theory of types [10]. DHOL has a distinguished type of booleans, and propositions are terms of that type. Types cannot have types, and proofs are not represented within the logic. DHOL is extensional: it uses standard equality for type checking, at the expense of decidability. Due to the lack of higher universes, DHOL is theoretically weaker than DTT with classical axioms, but there are reasons to believe that it is more amenable to automated reasoning.

To bridge the expressivity gap between the two logics and give proof assistant users access to automatic provers, we introduce a translation from a fragment of DTT to DHOL (Sect. 4). The fragment is restricted to three universes and supports only rank 1 (i.e. top level) polymorphism. Although it does not natively support advanced features such as inductive definitions [23] and quotient types [11], it still constitutes a large part of DTT as used in practice. Many DTT formalisations require no more than three universes, higher rank polymorphism is uncommon, and inductive definitions can be axiomatised [24].

Our translation is designed to preserve the structure of DTT expressions. In particular, it maps polymorphism to polymorphism and dependent types to dependent types. Based on experience with other translations between logics [6, 13, 17, 24], we hypothesise that such a lightweight translation will lead to more efficient automated proving.

Our translation is sound (Sect. 5). The detailed proofs of soundness and related lemmas can be found in our technical report [16]. We believe that our translation is also complete, but proving this is left for future work.

2 Dependent Type Theory

We describe a fragment of full DTT as presented by Carneiro [9]. We call it DTT3, or 'DTT with three universes'. We will later identify a fragment of DTT3 as the domain of our translation.

As a basis, we use a pure type system [4] with

$$\begin{aligned} \text{sorts}\ \ & \mathcal{S} = \{\mathsf{U}_0, \mathsf{U}_1, \mathsf{U}_2\} \\ \text{axioms}\ \ & \mathcal{A} = \{(\mathsf{U}_0 : \mathsf{U}_1), (\mathsf{U}_1 : \mathsf{U}_2)\} \\ \text{rules}\ \ & \mathcal{R} = \{(\mathsf{U}_i, \mathsf{U}_j, \mathsf{U}_{\mathrm{imax}(i,j)}) \mid 0 \leq i, j \leq 2\} \end{aligned}$$

where imax is defined as follows:

$$\mathrm{imax}(i,j) = \begin{cases} \max(i,j) & \text{if } j > 0 \\ 0 & \text{if } j = 0 \end{cases}$$

We will then add signatures, which are a device to enable global definitions for constants, and an axiomatisation of equality.

Table 1. Abbreviated notations

Abbreviation	Expansion
$\overrightarrow{e}$	$e_1, \ldots, e_n$
$\overrightarrow{e} : \overrightarrow{\alpha}$	$e_1 : \alpha_1, \ldots, e_n : \alpha_n$
$\overrightarrow{e} : \alpha$	$e_1 : \alpha, \ldots, e_n : \alpha$
$\forall_{\overrightarrow{x}:\overrightarrow{\alpha}}$	$\forall_{x_1:\alpha_1} \cdots \forall_{x_n:\alpha_n}$
$\lambda_{\overrightarrow{x}:\overrightarrow{\alpha}}$	$\lambda_{x_1:\alpha_1} \cdots \lambda_{x_n:\alpha_n}$

Table 1 summarizes abbreviations that we will use throughout. We use x, y, z to refer to variables, a, b for expressions without variables and c to refer to constants. Sometimes we write $\overrightarrow{x}^n$ instead of $\overrightarrow{x}$ to indicate the number of components. The abbreviations regarding $\forall$ or λ can include the cases where $n = 0$; a notation such as $\forall_{\overrightarrow{x}:\overrightarrow{\alpha}}\beta$ then denotes β. Moreover, we write $f\ \overrightarrow{e}$ to denote an iterated curried application.

The syntax of expressions and contexts is given by the following grammar:

$$\begin{aligned} \ell &::= 0 \mid 1 \mid 2 \mid \max(\ell,\ell) \mid \mathrm{imax}(\ell,\ell) \\ e &::= x \mid c \mid U_\ell \mid \forall_{x:e} e \mid e\ e \mid \lambda_{x:e} e \\ \Gamma &::= \circ \mid \Gamma, x : e \end{aligned}$$

The syntax of signatures $\mathcal{T}$ is given by the following grammar:

$$\mathcal{T} ::= . \mid \mathcal{T}, (c, e) \mid \mathcal{T}, (c, e, e)$$

Signatures formalise global definitions of constants c. Constants can be declared either only with a type α, which we write as (c, α), or additionally with a value e, which we write as (c, α, e).

The typing rules of DTT3 follow:

$$\boxed{\Gamma \vdash_{\mathcal{T}} e : \alpha}$$

$$\frac{\Gamma \vdash_{\mathcal{T}} \alpha : \mathsf{U}_\ell \quad \Gamma \vdash_{\mathcal{T}} e : \beta}{\Gamma, x : \alpha \vdash_{\mathcal{T}} e : \beta}\mathsf{Weak} \qquad \frac{\Gamma \vdash_{\mathcal{T}} \alpha : \mathsf{U}_\ell}{\Gamma, x : \alpha \vdash_{\mathcal{T}} x : \alpha}\mathsf{Assume}$$

$$\frac{}{\circ \vdash_{\mathcal{T}} \mathsf{U}_0 : \mathsf{U}_1}\mathsf{Univ0} \qquad \frac{}{\circ \vdash_{\mathcal{T}} \mathsf{U}_1 : \mathsf{U}_2}\mathsf{Univ1}$$

$$\frac{\Gamma \vdash_{\mathcal{T}} e_1 : \forall_{x:\alpha}\beta \quad \Gamma \vdash_{\mathcal{T}} e_2 : \alpha}{\Gamma \vdash_{\mathcal{T}} e_1\ e_2 : \beta[e_2/x]}\mathsf{Apply} \qquad \frac{\Gamma, x : \alpha \vdash_{\mathcal{T}} e : \beta}{\Gamma \vdash_{\mathcal{T}} \lambda_{x:\alpha} e : \forall_{x:\alpha}\beta}\lambda\mathsf{Form}$$

$$\frac{\Gamma \vdash_{\mathcal{T}} \alpha : \mathsf{U}_{\ell_1} \quad \Gamma, x : \alpha \vdash_{\mathcal{T}} \beta : \mathsf{U}_{\ell_2}}{\Gamma \vdash_{\mathcal{T}} \forall_{x:\alpha}\beta : \mathsf{U}_{\mathrm{imax}(\ell_1,\ell_2)}}\forall\mathsf{Form} \qquad \frac{\Gamma \vdash_{\mathcal{T}} e : \alpha \quad \Gamma \vdash_{\mathcal{T}} \alpha \equiv \beta}{\Gamma \vdash_{\mathcal{T}} e : \beta}\mathsf{Conv}$$

$$\frac{}{\circ \vdash_{\mathcal{T}} c : \mathcal{T}^{\mathsf{ty}}(c)}\mathsf{ConstTy}$$

where $\mathcal{T}^{\mathsf{ty}}(c)$ looks for an entry in $\mathcal{T}$ with c as the first component and returns the second component of the entry if it exists. Later we will also use $\mathcal{T}^{\mathsf{val}}(c)$ to refer to the same lookup but returning the third component if it exists.

We use $\equiv$ to denote definitional equality [9], including β-, δ- and η-conversion. We also assume proof irrelevance (PI) regarding $\mathbb{P}$, meaning the following holds:

$$\frac{\Gamma \vdash_{\mathcal{T}} p : \mathbb{P} \quad \Gamma \vdash_{\mathcal{T}} h : p \quad \Gamma \vdash_{\mathcal{T}} h' : p}{\Gamma \vdash_{\mathcal{T}} h \equiv h'}\mathsf{PI}$$

By $e_1[e_2/x]$ we denote the capture avoiding substitution of e_2 for x in e_1. If it is clear which variable is replaced, we simply write $e_1[e_2]$.

We sometimes write $\mathbb{P}$ for U_0. We also sometimes write $\alpha \to \beta$ for function types instead of $\forall_{x:\alpha}\beta$ when β does not depend on the argument x.

We add the following rules for the axiomatisation of propositional, or provable, equality following Angiuli and Gratzer [2]:

$$\frac{\Gamma \vdash_{\mathcal{T}} \alpha\ \mathsf{type} \quad \Gamma \vdash_{\mathcal{T}} a : \alpha \quad \Gamma \vdash_{\mathcal{T}} b : \alpha}{\Gamma \vdash_{\mathcal{T}} a =_\alpha b : \mathbb{P}}\mathsf{EqForm} \qquad \frac{\Gamma \vdash_{\mathcal{T}} \alpha\ \mathsf{type} \quad \Gamma \vdash_{\mathcal{T}} a : \alpha}{\Gamma \vdash_{\mathcal{T}} \mathsf{refl}_a : a =_\alpha a}\mathsf{EqIntro}$$

$$\frac{\Gamma \vdash_{\mathcal{T}} \alpha\ \mathsf{type} \quad \Gamma \vdash_{\mathcal{T}} a : \alpha \quad \Gamma \vdash_{\mathcal{T}} b : \alpha \quad \Gamma \vdash_{\mathcal{T}} p : a =_\alpha b \quad \Gamma, x : \alpha, y : \alpha, \pi : x =_\alpha y \vdash_{\mathcal{T}} Q : U_\ell \quad \Gamma, x : \alpha \vdash_{\mathcal{T}} R : Q[x, x, \mathsf{refl}_x]}{\Gamma \vdash_{\mathcal{T}} \mathcal{J}(R, p) : Q[a, b, p]}\mathsf{EqInd}$$

where $\mathcal{J}(R, \mathsf{refl}_a) \equiv R[a]$ for any $a : \alpha$. Using EqInd, we can then define

$$\mathsf{cast} : \forall_{\alpha,\beta:\mathsf{U}_\ell}\forall_{\pi:\alpha=_{\mathsf{U}_\ell}\beta}\alpha \to \beta.$$

We omit the type arguments of cast whenever they can be inferred from the context.

We define the following judgements for types, contexts and signatures:

$$\boxed{\Gamma \vdash \alpha\ \mathsf{type}} \qquad \frac{\Gamma \vdash_{\mathcal{T}} \alpha : U_\ell}{\Gamma \vdash_{\mathcal{T}} \alpha\ \mathsf{type}}$$

$$\boxed{\vdash_{\mathcal{T}} \Gamma\ \mathsf{ok}} \qquad \frac{\vdash \mathcal{T}\ \mathsf{sig}}{\vdash_{\mathcal{T}} \circ\ \mathsf{ok}} \qquad \frac{\vdash_{\mathcal{T}} \Gamma\ \mathsf{ok} \quad \Gamma \vdash_{\mathcal{T}} \alpha\ \mathsf{type}}{\vdash_{\mathcal{T}} \Gamma, x : \alpha\ \mathsf{ok}}$$

$$\boxed{\vdash \mathcal{T}\ \mathsf{sig}} \qquad \frac{}{\cdot\ \mathsf{sig}} \qquad \frac{\vdash \mathcal{T}\ \mathsf{sig} \quad \vdash_{\mathcal{T}} \alpha\ \mathsf{type}}{\vdash \mathcal{T}, (c, \alpha)\ \mathsf{sig}} \qquad \frac{\vdash \mathcal{T}\ \mathsf{sig} \quad \vdash_{\mathcal{T}} \alpha\ \mathsf{type} \quad \vdash_{\mathcal{T}} e : \alpha}{\vdash \mathcal{T}, (c, \alpha, e)\ \mathsf{sig}}$$

where c must be a fresh name and for technical reasons may not have a name of the form c_{val} for any name c.

DTT3 can easily be extended to the calculus of inductive constructions, which forms the basis of proof assistants such as Lean, Matita and Rocq.

3 Dependently Typed Higher Order Logic

We now present our translation's target language, polymorphic DHOL [25], a generalisation of DHOL [28]. Unless otherwise specified, by DHOL we mean the polymorphic variant. DHOL consists of a HOL basis, where the function type $A \to B$ is replaced by a dependent type $\Pi_{x:A} B$, where x may occur freely in B. Furthermore, there exists a sound and complete translation to HOL [25, 28], which enables automated theorem proving using existing efficient HOL provers.

We present the syntax and rules mainly following Ranalter et al. [25] and add some omitted rules following Rothgang et al. [28, 29]. We also add names to axioms and local assumptions following Rothgang et al. [28]. Finally, we index the ✓ construct used in the substitutions for formulae by the formula they replace. These changes are needed because we will want the translation function to be injective in the soundness proof.

The syntax of DHOL theories, contexts, substitutions, types and terms is given by the following grammar:

$$
\begin{aligned}
\text{theories } T &::= . \mid T, a : \Pi_{\overrightarrow{\alpha}} \Pi_{\overrightarrow{x:A}} : \mathsf{type} \mid T, c : \Pi_{\overrightarrow{\alpha}} A \mid T, c \rhd \Pi_{\overrightarrow{\alpha}} F \\
\text{contexts } \Gamma &::= \circ \mid \Gamma, \alpha : \mathsf{type} \mid \Gamma, x : A \mid \Gamma, x \rhd F \\
\text{substitutions } \gamma &::= \bullet \mid \gamma, A \mid \gamma, t \mid \gamma, \checkmark_F \\
\text{types } A, B &::= a\ \overrightarrow{A}\ \overrightarrow{t} \mid \alpha \mid \Pi_{x:A} B \mid o \\
\text{terms } t, u &::= c\ \overrightarrow{A} \mid x \mid \lambda_{x:A} t \mid t\ u \mid t \Rightarrow u \mid t =_A u
\end{aligned}
$$

The type o denotes the booleans, and terms $F : o$ are also called formulae (or propositions). We call (polymorphic) formulae declared in theories, written $c \rhd \Pi_{\overrightarrow{\alpha}} F$, global assumptions, and we call (monomorphic) formulae declared in contexts, written $x \rhd F$, local assumptions. We use the shorthand notations $\Pi_{\overrightarrow{x:A}} B$ and $\Pi_{\overrightarrow{\alpha}} B$ to mean $\Pi_{x_1:A_1} \ldots \Pi_{x_n:A_n} B$ and $\Pi_{\alpha_1} \ldots \Pi_{\alpha_n} B$ respectively, and, as in DTT3, $n = 0$ is included in these, in which case they are just B. Theories denote global declarations similar to signatures in DTT3. For substitutions, $\checkmark_F$ is the "term" that may be substituted for a local assumption $x \rhd F$ if and only if F is provable. Note that type is not itself a type. We write $t_1[t_2/x]$ and $t_1[t_2]$ to denote capture avoiding substitution. Again we sometimes write $A \to B$ instead of $\Pi_{x:A} B$ when B does not depend on x.

Finally, $F \Rightarrow G$ is a dependent implication. This means that to type-check $\Gamma \vdash_T F \Rightarrow G : o$, we need to type-check $\Gamma \vdash_T F : o$ as usual but may assume F as true when type checking G, so we need to prove $\Gamma, x \rhd F \vdash_T G : o$. This makes it possible to state formulae such as $\Gamma \vdash a =_A b \Rightarrow f\ a =_{B[a]} f\ b$ for some type A, a type family B and a dependent function $f : \Pi_{x:A} B$, since the truth of $a =_A b$ may be assumed during type checking. DHOL being an extensional logic, it has only one kind of equality instead of distinguishing between definitional and propositional equality as is done in DTT3. As a result, under the assumption that $a =_A b$, the types $B[a]$ and $B[b]$ can be unified directly, without needing to add constructs such as DTT3's cast. On the other hand, this means that type checking in DHOL is undecidable. This is considered

tolerable because the primary use case of DHOL, automated theorem proving, is undecidable regardless.

We define the remaining logical connectives and quantifiers following Andrews [1] and Rothgang et al. [28]:

$$\begin{aligned}
\top &:= (\lambda_{x:o} x) =_{o\to o} (\lambda_{x:o} x) \\
\bot &:= (\lambda_{x:o} x) =_{o\to o} (\lambda_{x:o} \top) \\
\neg F &:= F =_o \bot \\
\forall_{x:A} F &:= \lambda_{x:A} F =_{A\to o} \lambda_{x:A} \top \\
\exists_{x:A} F &:= \neg \forall_{x:A} \neg F \\
F \wedge G &:= \neg(F \Rightarrow \neg G) \\
F \vee G &:= \neg F \Rightarrow G
\end{aligned}$$

where F and G are formulae.

The inference rules for well formed theories T are as follows:

$$\boxed{\vdash T\ \mathsf{Thy}} \qquad \frac{}{\vdash .\ \mathsf{Thy}}\,\mathsf{ThyEmpty} \qquad \frac{\vdash T\ \mathsf{Thy} \qquad a \notin T \qquad \vdash_T \Delta\ \mathsf{Ctx} \qquad \Delta = \overrightarrow{\alpha} : \mathsf{type}, \overrightarrow{x} : \overrightarrow{A}}{\vdash T, a : \Pi_\Delta : \mathsf{type}\ \mathsf{Thy}}\,\mathsf{ThyTp}$$

$$\frac{\vdash T\ \mathsf{Thy} \qquad c \notin T \qquad \Delta \vdash_T A : \mathsf{type} \qquad \Delta = \overrightarrow{\alpha} : \mathsf{type}}{\vdash T, c : \Pi_\Delta A\ \mathsf{Thy}}\,\mathsf{ThyCon}$$

$$\frac{\vdash T\ \mathsf{Thy} \qquad c \notin T \qquad \Delta \vdash_T F : o \qquad \Delta = \overrightarrow{\alpha} : \mathsf{type}}{\vdash T, c \rhd \Pi_\Delta F\ \mathsf{Thy}}\,\mathsf{ThyAsn}$$

The rules for contexts are given on the left hand side below, and those for substitutions for them are given on the right hand side:

$$\boxed{\vdash_T \Delta\ \mathsf{Ctx}} \qquad\qquad \boxed{\Gamma \vdash_T \Delta \leftarrow \delta}$$

$$\frac{\vdash T\ \mathsf{Thy}}{\vdash_T \circ\ \mathsf{Ctx}}\,\mathsf{CtxEmpty} \qquad \frac{\vdash_T \Gamma\ \mathsf{Ctx}}{\Gamma \vdash_T \circ \leftarrow \bullet}\,\mathsf{SubEmpty}$$

$$\frac{\vdash_T \Delta\ \mathsf{Ctx}}{\vdash_T \Delta, \alpha : \mathsf{type}\ \mathsf{Ctx}}\,\mathsf{CtxTp} \qquad \frac{\Gamma \vdash_T \Delta \leftarrow \delta \qquad \Gamma \vdash_T A : \mathsf{type}}{\Gamma \vdash_T \Delta, \alpha : \mathsf{type} \leftarrow \delta, A}\,\mathsf{SubTp}$$

$$\frac{\vdash_T \Delta\ \mathsf{Ctx} \qquad \Delta \vdash_T A : \mathsf{type}}{\vdash_T \Delta, x : A\ \mathsf{Ctx}}\,\mathsf{CtxVar} \qquad \frac{\Gamma \vdash_T \Delta \leftarrow \delta \qquad \Delta \vdash_T A : \mathsf{type} \qquad \Gamma \vdash_T t : A[\delta]}{\Gamma \vdash_T \Delta, x : A \leftarrow \delta, t}\,\mathsf{SubVar}$$

$$\frac{\vdash_T \Delta\ \mathsf{Ctx} \qquad \Delta \vdash_T F : o}{\vdash_T \Delta, x \rhd F\ \mathsf{Ctx}}\,\mathsf{CtxAsn} \qquad \frac{\Gamma \vdash_T \Delta \leftarrow \delta \qquad \Delta \vdash_T F : o \qquad \Gamma \vdash_T F[\delta]}{\Gamma \vdash_T \Delta, x \rhd F \leftarrow \delta, \checkmark_F}\,\mathsf{SubAsn}$$

The rules for looking up a type, term or assumption in a theory are given on the left hand side, and the corresponding rules for lookup in a context are given on the right hand side:

$$\boxed{\Gamma \vdash_T A : \mathsf{type}} \qquad \frac{\vdash_T \Gamma\ \mathsf{Ctx} \qquad \Gamma \vdash_T \Delta \leftarrow \delta \qquad a : \Pi_\Delta : \mathsf{type}\ \text{in}\ T}{\Gamma \vdash_T a\ \delta : \mathsf{type}}\mathsf{TpSym} \qquad \frac{\vdash_T \Gamma\ \mathsf{Ctx} \qquad \alpha : \mathsf{type}\ \text{in}\ \Gamma}{\Gamma \vdash_T \alpha : \mathsf{type}}\mathsf{TpVar}$$

$$\boxed{\Gamma \vdash_T t : A} \qquad \frac{\vdash_T \Gamma\ \mathsf{Ctx} \qquad \Gamma \vdash_T \Delta \leftarrow \delta \qquad c : \Pi_\Delta B\ \text{in}\ T}{\Gamma \vdash_T c\ \delta : B[\delta]}\mathsf{TermSym} \qquad \frac{\vdash_T \Gamma\ \mathsf{Ctx} \qquad x : A\ \text{in}\ \Gamma}{\Gamma \vdash_T x : A}\mathsf{TermVar}$$

$$\boxed{\Gamma \vdash_T F} \qquad \frac{\vdash_T \Gamma\ \mathsf{Ctx} \qquad \Gamma \vdash_T \Delta \leftarrow \delta \qquad c \rhd \Pi_\Delta F\ \text{in}\ T}{\Gamma \vdash_T F[\delta]}\mathsf{ValidSym} \qquad \frac{\vdash_T \Gamma\ \mathsf{Ctx} \qquad x \rhd F\ \text{in}\ \Gamma}{\Gamma \vdash_T F}\mathsf{ValidVar}$$

The rules for the formation and the terms of booleans are as follows:

$$\frac{\vdash_T \Gamma\ \mathsf{Ctx}}{\Gamma \vdash_T o : \mathsf{type}}\mathsf{TpBool} \qquad \frac{\Gamma \vdash_T t : A \qquad \Gamma \vdash_T u : A}{\Gamma \vdash_T t =_A u : o}\mathsf{TermEq}$$

$$\frac{\Gamma \vdash_T F : o \qquad \Gamma, x \rhd F \vdash_T G : o}{\Gamma \vdash_T F \Rightarrow G : o}\mathsf{TermImpl}$$

Type formation, λ-abstraction and application are defined as follows:

$$\frac{\Gamma \vdash_T A : \mathsf{type} \qquad \Gamma, x : A \vdash_T B : \mathsf{type}}{\Gamma \vdash_T \Pi_{x:A} B : \mathsf{type}}\mathsf{TpPi} \qquad \frac{\Gamma \vdash_T A : \mathsf{type} \qquad \Gamma, x : A \vdash_T t : B}{\Gamma \vdash_T \lambda_{x:A} t : \Pi_{x:A} B}\mathsf{TermLambda}$$

$$\frac{\Gamma \vdash_T t : \Pi_{x:A} B \qquad \Gamma \vdash_T u : A}{\Gamma \vdash_T t\, u : B[u]}\mathsf{TermApply}$$

The type equality judgement is defined by the following rules:

$$\boxed{\Gamma \vdash_T A \equiv A'} \qquad \frac{\vdash_T \Gamma\ \mathsf{Ctx} \qquad \alpha : \mathsf{type}\ \text{in}\ \Gamma}{\Gamma \vdash_T \alpha \equiv \alpha}\mathsf{TpEqVar}$$

$$\frac{\vdash_T \Gamma\ \mathsf{Ctx} \qquad \Gamma \vdash_T \delta \equiv_\Delta \delta' \qquad a : \Pi_\Delta \mathsf{type}\ \text{in}\ T}{\Gamma \vdash_T a\ \delta \equiv a\ \delta'}\mathsf{TpEqSym} \qquad \frac{\vdash_T \Gamma\ \mathsf{Ctx}}{\Gamma \vdash_T o \equiv o}\mathsf{TpEqBool}$$

$$\frac{\Gamma \vdash_T A \equiv A' \qquad \Gamma, x : A \vdash_T B \equiv B'}{\Gamma \vdash_T \Pi_{x:A} B \equiv \Pi_{x:A'} B'}\mathsf{TpEqPi} \qquad \frac{\Gamma \vdash_T t : A \qquad \Gamma \vdash_T A \equiv A'}{\Gamma \vdash_T t : A'}\mathsf{TermConvert}$$

where $\Gamma \vdash_T \delta \equiv_\Delta \delta'$ abbreviates the component-wise provable equality of two substitutions for Δ.

The following rules were omitted by Ranalter et al. [25], so we adapt them from Rothgang et al. [28, 29]:

$$\frac{\Gamma \vdash_T \lambda_{x:A} t : \Pi_{x:A} B \qquad \Gamma \vdash_T u : A}{\Gamma \vdash_T (\lambda_{x:A} t)\ u =_{B[u]} t[u]}\mathsf{TermBeta} \qquad \frac{\Gamma \vdash_T t : \Pi_{x:A} B \qquad x \text{ not in } \Gamma}{\Gamma \vdash_T t =_{\Pi_{x:A} B} \lambda_{x:A} t\ x}\mathsf{TermEta}$$

$$\frac{\Gamma \vdash_T A \equiv A' \qquad \Gamma, x : A \vdash_T t =_B t'}{\Gamma \vdash_T \lambda_{x:A} t =_{\Pi_{x:A} B} \lambda_{x:A'} t'}\mathsf{CongLambda} \qquad \frac{\Gamma \vdash_T t =_A t' \qquad \Gamma \vdash_T f =_{\Pi_{x:A} B} f'}{\Gamma \vdash_T f\ t =_{B[t]} f'\ t'}\mathsf{CongApply}$$

$$\frac{\Gamma \vdash_T t : A}{\Gamma \vdash_T t =_A t}\mathsf{Refl} \qquad \frac{\Gamma \vdash_T t =_A s}{\Gamma \vdash_T s =_A t}\mathsf{Sym} \qquad \frac{\Gamma \vdash_T F =_o F' \qquad \Gamma \vdash_T F'}{\Gamma \vdash_T F}\mathsf{CongValid}$$

$$\frac{\Gamma, x \rhd F \vdash_T G}{\Gamma \vdash_T F \Rightarrow G}\mathsf{ImplIntro} \qquad \frac{\Gamma \vdash_T F \Rightarrow G \qquad \Gamma \vdash_T F}{\Gamma \vdash_T G}\mathsf{MP}$$

Finally, it is possible to add the following rule for boolean extensionality:

$$\frac{\Gamma \vdash_T t : \Pi_{x:o} o \qquad \Gamma \vdash_T t\ \bot \qquad \Gamma \vdash_T t\ \top}{\Gamma, x : o \vdash_T t\ x}\mathsf{BoolExt}$$

If we add this rule, then a suitable rule for the law of the excluded middle, as well as suitable axiomatisations of the empty type, the singleton type and the co-product, are needed in DTT3 to prove the our translation's soundness.

Full DTT is strictly stronger than DHOL. This is due to the former's infinite universe hierarchy, which allows for the same strength as set theory with inaccessible cardinals [31]. By contrast, DHOL has the same strength as HOL by the existence of a sound and complete translation. HOL, in turn, is subsumed by the calculus of constructions, which can be expressed in set theory without additional cardinals [31].

We believe that DHOL lends itself well to automated theorem proving, because of its extensionality and because it resembles HOL. This might enable a term rewriting based proof strategy such as superposition, which is highly successful in FOL and HOL, to be implemented in DHOL and achieve high success rates.

4 The Translation

Here we present our translation from DTT3 to DHOL. The main difficulty in defining it is to map the expressions of DTT3 to either DHOL types or terms depending on the situation. We achieve this through the definition of several mutually recursive translation functions.

Due to the limitations of the polymorphism in DHOL to rank 1 polymorphism and DHOL's lack of a type corresponding to U_1, which in DTT3 could occur anywhere (e.g. as a type argument), we also restrict the translation's domain.

Definition 1 (Translatability). Given a DTT3 signature $\mathcal{T}$ and a DTT3 context Γ, we call a DTT3 expression e $(\mathcal{T}, \Gamma)$-*translatable* if all the following hold:

1. the universes U_1 and U_2 do not occur in e,
2. all subexpressions of the form $\lambda_{x:\alpha} e_1$, have type $\forall_{x:\alpha}\beta$ with $\Gamma, \Delta, x : \alpha \vdash_{\mathcal{T}} \beta \not: \mathsf{U}_2$,
3. for all subexpressions $c\ \overrightarrow{e_1}^n$, where c is a constant with $\Gamma, \Delta \vdash_{\mathcal{T}} c : \forall_{\overrightarrow{x}^n:\overrightarrow{\alpha}^n}\beta$, we have $\Gamma, \Delta \vdash_{\mathcal{T}} c\ \overrightarrow{e_1}^n : \beta[\overrightarrow{e_1}^n]$ and β is not a function type,

where Δ consists of the binders which occur prior to the subexpression of e.

We define *translatability* of a DTT3 signature $\mathcal{T}$ inductively:

1. . is translatable,
2. $\mathcal{T}, (c, \alpha)$ is translatable, if $\mathcal{T}$ is translatable and α is $(\mathcal{T}, \circ)$-translatable or if α is of the form $\forall_{\overrightarrow{x}:\mathsf{U}_1}\forall_{\overrightarrow{y}:\overrightarrow{\alpha}}\beta$, all α_i are $(\mathcal{T}, \overrightarrow{x} : \mathsf{U}_1, \overrightarrow{y}^{i-1} : \overrightarrow{\alpha}^{i-1})$-translatable and β is $(\mathcal{T}, \overrightarrow{x} : \mathsf{U}_1, \overrightarrow{y} : \overrightarrow{\alpha})$-translatable or $\beta = \mathsf{U}_1$,
3. $\mathcal{T}, (c, \alpha, e)$ is translatable, if $\mathcal{T}, (c, \alpha)$ is translatable and e is $(\mathcal{T}, \circ)$-translatable.

Given a DTT3 signature $\mathcal{T}$, we define $\mathcal{T}$-*translatability* of a DTT3 context Γ inductively:

1. $\circ$ is $\mathcal{T}$-translatable,
2. $\Gamma, x : \alpha$ is $\mathcal{T}$-translatable, if Γ is $\mathcal{T}$-translatable and either α is $(\mathcal{T}, \Gamma)$-translatable or $\alpha = \mathsf{U}_1$.

We omit $\mathcal{T}$ and Γ if they are clear from the context. This fragment of DTT3 may seem restrictive, but it is sufficient for many applications of DTT. In particular, higher rank polymorphism and higher universe levels are not needed to formalise many problems that do not relate to category theory or higher algebra.

The translation functions for DTT3 expressions inspect them without β-, δ-, or η-normalisation. The function definition mapping to DHOL terms includes cases for $\top, \bot, \wedge, \vee, \neg$ and $\exists$, which are not built into DTT3, and map these to the corresponding DHOL constructs. Our hope is that this will facilitate more efficient theorem proving on the resulting DHOL formulae.

Definition 2. Let Γ be a DTT3 context and $\mathcal{T}$ be a DTT3 signature. Then:

- $[\![\cdot]\!]^{\Gamma;\mathcal{T}}_{\text{exp}}$ maps translatable DTT3 expressions to DHOL types or terms,
- $[\![\cdot]\!]^{\Gamma;\mathcal{T}}_{\text{ty}}$ maps translatable DTT3 expressions α where $\Gamma \vdash_{\mathcal{T}} \alpha : \forall_{\overrightarrow{x}:\overrightarrow{\alpha}}\mathsf{U}_1$ to DHOL types,
- $[\![\cdot]\!]^{\Gamma;\mathcal{T}}_{\text{ter}}$ maps translatable DTT3 expressions e where $\Gamma \vdash_{\mathcal{T}} e : \forall_{\overrightarrow{x}:\overrightarrow{\beta}}\alpha$ and $\Gamma \vdash_{\mathcal{T}} \alpha : \mathsf{U}_1$ to DHOL terms,
- $[\![\cdot]\!]_{\text{sig}}$ maps translatable DTT3 signatures to DHOL theories and
- $[\![\cdot]\!]^{\mathcal{T}}_{\text{ctx}}$ maps translatable DTT3 contexts to DHOL contexts.

Moreover, we introduce the function rmc, which takes a translatable DTT3 expression and if it is of the form $\mathsf{cast}\ p\ e$ it replaces it by e. Finally, we let $[\![\cdot]\!]^{\Gamma;\mathcal{T}}_{\text{ty0}} = [\![\cdot]\!]^{\Gamma;\mathcal{T}}_{\text{ty}} \circ \mathsf{rmc}$ and $[\![\cdot]\!]^{\Gamma;\mathcal{T}}_{\text{ter0}} = [\![\cdot]\!]^{\Gamma;\mathcal{T}}_{\text{ter}} \circ \mathsf{rmc}$.

$$[\![e]\!]^{\Gamma;\mathcal{T}}_{\text{exp}} = \begin{cases} [\![e_1]\!]^{\Gamma;\mathcal{T}}_{\text{exp}}\ [\![e_2]\!]^{\Gamma;\mathcal{T}}_{\text{exp}} & \text{if } e = e_1\ e_2 \text{ and } e_1 \neq \mathsf{cast}\ p \\ & \text{and not } \Gamma \vdash_{\mathcal{T}} e : \alpha \text{ with } \Gamma \vdash_{\mathcal{T}} \alpha : \mathbb{P} \\ [\![e]\!]^{\Gamma;\mathcal{T}}_{\text{ty0}} & \text{if } \Gamma \vdash_{\mathcal{T}} e : \forall_{\overrightarrow{x}:\overrightarrow{\alpha}} \mathsf{U}_1 \\ & \text{and } e = \mathsf{cast}\ p\ e_1 \text{ or } e \text{ is not an application} \\ [\![e]\!]^{\Gamma;\mathcal{T}}_{\text{ter0}} & \text{if } \Gamma \vdash_{\mathcal{T}} e : \forall_{\overrightarrow{x}:\overrightarrow{\beta}} \alpha \text{ with } \Gamma \vdash_{\mathcal{T}} \alpha : \mathsf{U}_1 \\ & \text{and } e = \mathsf{cast}\ p\ e_1 \text{ or } e \text{ is not an application} \\ \checkmark_{[\![\alpha]\!]^{\Gamma;\mathcal{T}}_{\text{ter0}}} & \text{if } \Gamma \vdash_{\mathcal{T}} e : \alpha \text{ with } \Gamma \vdash_{\mathcal{T}} \alpha : \mathbb{P} \end{cases}$$

$$[\![\alpha]\!]^{\Gamma;\mathcal{T}}_{\text{ty}} = \begin{cases} c & \text{if } \alpha = c \text{ is a constant} \\ x & \text{if } \alpha = x \text{ is a variable} \\ o & \text{if } \alpha = \mathbb{P} \\ [\![\alpha]\!]^{\Gamma;\mathcal{T}}_{\text{exp}} & \text{if } \alpha = \alpha_1\ \alpha_2 \\ \Pi_{x:[\![\alpha_1]\!]^{\Gamma;\mathcal{T}}_{\text{ty0}}} [\![\alpha_2]\!]^{\Gamma,x:\alpha_1;\mathcal{T}}_{\text{ty0}} & \text{if } \alpha = \forall_{x:\alpha_1} \alpha_2 \end{cases}$$

$$[\![e]\!]^{\Gamma;\mathcal{T}}_{\text{ter}} = \begin{cases} c & \text{if } e = c \text{ is a constant} \\ x & \text{if } e = x \text{ is a variable} \\ \lambda_{x:[\![\alpha]\!]^{\Gamma;\mathcal{T}}_{\text{ty0}}}.[\![e_1]\!]^{\Gamma,x:\alpha;\mathcal{T}}_{\text{exp}} & \text{if } e = \lambda_{x:\alpha}.e_1 \\ [\![e]\!]^{\Gamma;\mathcal{T}}_{\text{exp}} & \text{if } e = e_1\ e_2 \\ \top & \text{if } e = \top \\ \bot & \text{if } e = \bot \\ [\![e_1]\!]^{\Gamma;\mathcal{T}}_{\text{ter0}} \diamond [\![e_2]\!]^{\Gamma;\mathcal{T}}_{\text{ter0}} & \text{if } e = e_1 \diamond e_2 \text{ with } \diamond \in \{\wedge, \vee\} \\ \neg [\![e_1]\!]^{\Gamma;\mathcal{T}}_{\text{ter0}} & \text{if } e = \neg e_1 \\ \exists_{x:[\![\alpha]\!]^{\Gamma;\mathcal{T}}_{\text{ty0}}} [\![e_1]\!]^{\Gamma,x:\alpha;\mathcal{T}}_{\text{ter0}} & \text{if } e = \exists_{x:\alpha} e_1 \\ \forall_{x:[\![\alpha]\!]^{\Gamma;\mathcal{T}}_{\text{ty0}}} [\![e_1]\!]^{\Gamma,x:\alpha;\mathcal{T}}_{\text{ter0}} & \text{if } e = \forall_{x:\alpha} e_1 \text{ with } \Gamma \vdash_{\mathcal{T}} \alpha : \beta \text{ and } \beta \neq \mathbb{P} \\ [\![\alpha]\!]^{\Gamma;\mathcal{T}}_{\text{ter0}} \Rightarrow [\![e_2]\!]^{\Gamma;\mathcal{T}}_{\text{ter0}} & \text{if } e = \forall_{x:\alpha} e_1 \text{ with } \Gamma \vdash_{\mathcal{T}} \alpha : \mathbb{P} \\ [\![e_1]\!]^{\Gamma;\mathcal{T}}_{\text{ter0}} =_{[\![\alpha]\!]^{\Gamma;\mathcal{T}}_{\text{ty0}}} [\![e_2]\!]^{\Gamma;\mathcal{T}}_{\text{ter0}} & \text{if } e = (e_1 =_\alpha e_2) \end{cases}$$

$$[\![\Gamma]\!]^{\mathcal{T}}_{\text{ctx}} = \begin{cases} \circ & \text{if } \Gamma = \circ \\ [\![\Gamma']\!]^{\mathcal{T}}_{\text{ctx}}, x : \mathsf{type} & \text{if } \Gamma = \Gamma', x : \mathsf{U}_1 \\ [\![\Gamma']\!]^{\mathcal{T}}_{\text{ctx}}, x : o & \text{if } \Gamma = \Gamma', x : \mathbb{P} \\ [\![\Gamma']\!]^{\mathcal{T}}_{\text{ctx}}, x \rhd [\![e]\!]^{\Gamma';\mathcal{T}}_{\text{ter}} & \text{if } \Gamma = \Gamma', x : e \text{ with } e : \mathbb{P} \\ [\![\Gamma']\!]^{\mathcal{T}}_{\text{ctx}}, x : [\![e]\!]^{\Gamma';\mathcal{T}}_{\text{ty}} & \text{if } \Gamma = \Gamma', x : e \text{ with } e \neq \mathbb{P} \text{ and } e : \mathsf{U}_1 \end{cases}$$

$$[\![\mathcal{T}]\!]_{\text{sig}} = \begin{cases} . & \text{if } \mathcal{T} = . \\ [\![\mathcal{T}', (c, \alpha)]\!]_{\text{sig}}, c_{\mathsf{val}} \rhd c =_{[\![\alpha]\!]^{\Gamma;\mathcal{T}}_{\text{ty}}} [\![e]\!]^{\Gamma;\mathcal{T}}_{\text{ter}} & \text{if } \mathcal{T} = \mathcal{T}', (c, \alpha, e) \\ [\![\mathcal{T}']\!]_{\text{sig}}, c \rhd \Pi_{\overrightarrow{x}^n} [\![e]\!]^{\Gamma, \overrightarrow{x}^n:U_1;\mathcal{T}}_{\text{ter}} & \text{if } \mathcal{T} = \mathcal{T}', (c, \forall_{\overrightarrow{x}^n:\mathsf{U}_1} e) \\ & \text{with } e : \mathbb{P} \\ [\![\mathcal{T}']\!]_{\text{sig}}, c : \Pi_{\overrightarrow{x}^n} \Pi_\Delta : \mathsf{type} & \\ \text{if } \mathcal{T} = \mathcal{T}', (c, \forall_{\overrightarrow{x}^n:\mathsf{U}_1} \forall_{\overrightarrow{y}^m:\overrightarrow{\alpha}^m} \mathsf{U}_1) & \\ [\![\mathcal{T}']\!]_{\text{sig}}, c : \Pi_{\overrightarrow{x}^n} [\![\alpha]\!]^{\Gamma, \overrightarrow{x}^n:U_1;\mathcal{T}}_{\text{ty}} & \text{if } \mathcal{T} = \mathcal{T}', (c, \forall_{\overrightarrow{x}^n:\mathsf{U}_1} \alpha) \\ & \text{with } \alpha : \mathsf{U}_1 \end{cases}$$

where $\Delta = y_1 : [\![\alpha_1]\!]^{\Gamma, \overrightarrow{x}^n:U_1;\mathcal{T}}_{\text{ty}}, y_2 : [\![\alpha_2]\!]^{\Gamma, \overrightarrow{x}^n:U_1, y_1:\alpha_1;\mathcal{T}}_{\text{ty}}, \ldots,$
$y_m : [\![\alpha_m]\!]^{\Gamma, \overrightarrow{x}^n:U_1, \overrightarrow{y}^{m-1}:\overrightarrow{\alpha}^{m-1};\mathcal{T}}_{\text{ty}}$

The annotations on some of the translation functions with a context and signature are necessary for the definition, but we omit them when they can easily be inferred.

Example 1. An axiomatisation of polymorphic length indexed vectors as a signature in DTT3 might look like this:

$$\mathcal{T} = (\mathbb{N}, \mathsf{U}_1),\ (0, \mathbb{N}),\ (\mathsf{suc}, \forall_{n:\mathbb{N}}\mathbb{N}),\ (\mathsf{Vec}, \forall_{\alpha:\mathsf{U}_1}\forall_{n:\mathbb{N}}\mathsf{U}_1),\ (\mathsf{nil}, \forall_{\alpha:\mathsf{U}_1}\mathsf{Vec}\ \alpha\ 0),\\ (\mathsf{cons}, \forall_{\alpha:\mathsf{U}_1}\forall_{n:\mathbb{N}}\forall_{v:\mathsf{Vec}\,\alpha\,n}\forall_{x:\alpha}\mathsf{Vec}\ \alpha\ (\mathsf{suc}\ n))$$

The translation of this signature is the following DHOL theory:

$$[\![\mathcal{T}]\!]_{\mathrm{sig}} = \mathbb{N} : \mathsf{type},\ 0 : \mathbb{N},\ \mathsf{suc} : \Pi_{n:\mathbb{N}}\mathbb{N},\ \mathsf{Vec} : \Pi_\alpha\Pi_{n:\mathbb{N}} : \mathsf{type},\ \mathsf{nil} : \Pi_\alpha\mathsf{Vec}\ \alpha\ 0,\\ \mathsf{cons} : \Pi_\alpha\Pi_{n:\mathbb{N}}\Pi_{v:\mathsf{Vec}\,\alpha\,n}\Pi_{x:\alpha}\mathsf{Vec}\ \alpha\ (\mathsf{suc}\ n)$$

We can state the following formula in DTT3:

$$\alpha : \mathsf{U}_1 \vdash_{\mathcal{T}} \forall_{x:\alpha}\forall_{y:\alpha}\forall_{n:\mathbb{N}}\forall_{v:\mathsf{Vec}\,\alpha\,n}\forall_{p\,:\,\mathsf{cons}\,\alpha\,n\,v\,x =_{\mathsf{Vec}\,\alpha\,(\mathsf{suc}\,n)}\,\mathsf{cons}\,\alpha\,n\,v\,y} x =_\alpha y : \mathbb{P}$$

We can then translate the context, the signature, the proposition and its type, using $[\![\cdot]\!]_{\mathrm{exp}}$ for the last two, as follows:

$$\alpha : \mathsf{type} \vdash_{[\![\mathcal{T}]\!]_{\mathrm{sig}}}\\ \forall_{x:\alpha}\forall_{y:\alpha}\forall_{n:\mathbb{N}}\forall_{v:\mathsf{Vec}\,\alpha\,n}\mathsf{cons}\ \alpha\ n\ v\ x =_{\mathsf{Vec}\,\alpha\,(\mathsf{suc}\,n)} \mathsf{cons}\ \alpha\ n\ v\ y \Rightarrow x =_\alpha y : o$$

Notably, the translation maps the $\mathbb{P}$-valued dependent function types of DTT3 to the $\forall$ quantifier in DHOL, which is defined in terms of equality and implication, instead of the DHOL dependent function type. Furthermore, since the last dependent function quantifies over a proposition, DHOL's dependent implication is used.

Example 2. This example demonstrates how casts are removed and how this interacts with provability. We reuse the length indexed vectors from above and assume a further axiomatisation of $+ : \mathbb{N} \to \mathbb{N} \to \mathbb{N}$, which we write infix as usual, as well as of $\mathsf{append} : \forall_{\alpha:\mathsf{U}_1}\forall_{m:\mathbb{N}}\forall_{n:\mathbb{N}}\forall_{v:\mathsf{Vec}\,\alpha\,m}\forall_{w:\mathsf{Vec}\,\alpha\,n}\mathsf{Vec}\ \alpha\ (m+n)$ in $\mathcal{T}$. In DTT3 the following is ill-typed:

$$\alpha : \mathsf{U}_1 \vdash_{\mathcal{T}} \forall_{m:\mathbb{N}}\forall_{n:\mathbb{N}}\forall_{v:\mathsf{Vec}\,\alpha\,m}\forall_{w:\mathsf{Vec}\,\alpha\,n}\\ \mathsf{append}\ \alpha\ m\ n\ v\ w =_{\mathsf{Vec}\,\alpha\,(m+n)} \mathsf{append}\ \alpha\ n\ m\ w\ v : \mathbb{P}$$

This is because $m + n$ and $n + m$ is propositionally equal but not definitionally equal. Therefore the types $\mathsf{Vec}\ \alpha\ (m + n)$ and $\mathsf{Vec}\ \alpha\ (n + m)$ are also not definitionally equal, which is required to even express propositional equality of the above vectors. If, however, we replace the right hand side of the equation by $\alpha : \mathsf{U}_1 \vdash_{\mathcal{T}} \mathsf{cast}\ p\ (\mathsf{append}\ \alpha\ n\ m\ w\ v)$, where p is an easy-to-construct proof term of type $\mathsf{Vec}\ \alpha\ (n + m) =_{\mathsf{U}_1} \mathsf{Vec}\ \alpha\ (m + n)$, this type-checks.

By contrast, the translated version of the above statement type-checks without issue, since DHOL is extensional. Specifically, while type checking, the rule TpEqSym would be used, where one of the conditions would be that $\Gamma \vdash_{[\![\mathcal{T}]\!]_{\mathrm{sig}}} m + n =_{\mathbb{N}} n + m$, which is provable. For this reason, occurrences of cast are removed during the translation.

5 Soundness

A key theoretical property of our translation is that it reflects judgements: if some DHOL terms, types, theories or contexts satisfy a DHOL judgement, then there exist corresponding translatable DTT3 expressions, signatures and contexts of which the DHOL counterparts are translations and for which a corresponding DTT3 judgement holds. This entails soundness, meaning that if a translated DTT3 expression is provable in DHOL, the original expression is provable in DTT3 (i.e. a proof term exists).

To prove reflection of judgements, we first show a certain kind of injectivity of our translation. This is needed to identify DTT3 constructs that according to the induction hypothesis in the proof of the theorem translate to the same corresponding DHOL construct but originate from different applications of the induction hypothesis.

Lemma 1 (Injectivity). *The translation functions* $[\![\cdot]\!]_{\text{ty}}$, $[\![\cdot]\!]_{\text{ter}}$, $[\![\cdot]\!]_{\text{exp}}$, $[\![\cdot]\!]_{\text{sig}}$ *and* $[\![\cdot]\!]_{\text{ctx}}$ *are all injective on the translatable fragment of DTT3, where additionally there are (possibly trivial)* cast*s in front of every subexpression. For DTT3 we use definitional equality* $\equiv$*; for DHOL we use syntactic equality* $=$*.*

Proof. We prove injectivity by mutual structural induction on the inputs, separating the cases by function. Since the DTT3 expressions have casts in front of every subexpression, which are stripped by the translation functions, two such DTT3 expressions that translate to the same DHOL type or term are definitionally equal if the remaining non-cast components are. For this the proofs which are the first argument of the casts are always definitionally equal, since either are proofs of the same proposition, so PI applies.

$[\![\cdot]\!]_{\text{ty}}$*:* Let α, β be translatable DTT3 expressions of type $\forall_{\vec{x}:\vec{\alpha}} \mathsf{U}_1$. We show that if $[\![\alpha]\!]_{\text{ty}} = [\![\beta]\!]_{\text{ty}}$ then $\alpha \equiv \beta$:

If $\alpha = x$ or $\alpha = c$, the translation function is the identity, therefore the conclusion follows immediately.

If $\alpha = \mathbb{P}$, then $[\![\alpha]\!]_{\text{ty}} = o = [\![\beta]\!]_{\text{ty}}$. This immediately implies that $\beta = \mathbb{P}$, since $\mathbb{P}$ by induction is the only pre-image of o under $[\![\cdot]\!]_{\text{ty}}$.

If $\alpha = \alpha_1\ \alpha_2$, we have $[\![\alpha]\!]_{\text{ty}} = [\![\alpha]\!]_{\text{exp}} = [\![\beta]\!]_{\text{exp}}$, which again by induction has only one pre-image and we conclude $\alpha \equiv \beta$ using the induction hypothesis regarding $[\![\cdot]\!]_{\text{exp}}$.

If $\alpha = \forall_{x:\alpha_1} \alpha_2$, then $[\![\alpha]\!]_{\text{ty}} = \Pi_{x:[\![\alpha_1]\!]_{\text{ty}}} [\![\alpha_2]\!]_{\text{ty}} = [\![\beta]\!]_{\text{ty}}$. Again by induction this only has one pre-image, therefore $\beta = \forall_{x:\alpha_1} \alpha_2$.

The case for $\alpha = \lambda_{x:\alpha_1} \alpha_2$ is completely analogous.

Expressions of the form $\alpha_1 =_{\alpha_2} \alpha_3$ are by definition of type $\mathbb{P}$ and can therefore not occur here.

$[\![\cdot]\!]_{\text{ter}}$*:* Let e, d be translatable DTT3 expressions of type $\forall_{\vec{x}:\vec{\beta}} \alpha$, where $\alpha : \mathsf{U}_1$. We show that if $[\![e]\!]_{\text{ter}} = [\![d]\!]_{\text{ter}}$ then $e \equiv d$:

Cases $e = x, e = c, e = e_1\ e_2$ are completely analogous to $[\![\cdot]\!]_{\text{ty}}$.

If $e = \exists_{x:\alpha} e_1$, then $[\![e]\!]_{\text{ter}} = \exists_{x:[\![\alpha]\!]_{\text{ty}}} [\![e_1]\!]_{\text{ter}}$. We use the induction hypothesis of the mutual induction regarding $[\![\cdot]\!]_{\text{ty}}$, which gets us that $[\![d]\!]_{\text{ter}} = \exists_{x:[\![\alpha]\!]_{\text{ty}}} [\![e_1]\!]_{\text{ter}}$. We conclude $e \equiv d$ from the definition of $[\![\cdot]\!]_{\text{ter}}$.

If $e = \forall_{x:\alpha} e_1$, we need to perform a case distinction on whether $\alpha : \mathbb{P}$ or not. If $\alpha : \mathbb{P}$ we have $[\![e]\!]_{\text{ter}} = [\![\alpha]\!]_{\text{ter}} \Rightarrow [\![e_1]\!]_{\text{ter}}$. We use the induction hypothesis, which gets us that $[\![d]\!]_{\text{ter}} = [\![\alpha]\!]_{\text{ter}} \Rightarrow [\![e_1]\!]_{\text{ter}}$. We conclude $e \equiv d$ from the definition of $[\![\cdot]\!]_{\text{ter}}$. If not, we have $[\![e]\!]_{\text{ter}} = \forall_{x:[\![\alpha]\!]_{\text{ty}}} [\![e_1]\!]_{\text{ter}}$ and the rest is analogous to the $\exists$ case.

For $e = (e_1 =_\alpha e_2)$ and $e = \lambda_{x:\alpha} e_1$ we employ the induction hypothesis of the mutual induction similarly.

Finally, $e = e_1 \wedge e_2, e = e_1 \vee e_2, e = \neg e_1$ all follow analogously to $e = (e_1 =_\alpha e_2)$.

$[\![\cdot]\!]_{\text{exp}}$*:* In the case of e being of the form $\mathsf{cast}\ p\ e_1$, or it is not an application, this immediately follows from the injectivity of $[\![\cdot]\!]_{\text{ty}}$ and $[\![\cdot]\!]_{\text{ter}}$ as well as IP for the ✓ case. Otherwise, if e is an application without top-level cast, we invoke the induction hypothesis as before.

$[\![\cdot]\!]_{\text{sig}}$*:* Let $\mathcal{T}_1, \mathcal{T}_2$ be well formed DTT3 signatures. We show that if $[\![\mathcal{T}_1]\!]_{\text{sig}} = [\![\mathcal{T}_2]\!]_{\text{sig}}$ then $\mathcal{T}_1 \equiv \mathcal{T}_2$:

If $\mathcal{T}_1 = .$, then $[\![\mathcal{T}_1]\!]_{\text{sig}} = . = [\![\mathcal{T}_2]\!]_{\text{sig}}$ and trivially $\mathcal{T}_1 = \circ$.

If $\mathcal{T}_1 = \mathcal{T}_1', (c, \forall_{\overrightarrow{x:\mathsf{U}_1}^n} e)$ we need to do a case distinction on e. First if $e : \mathbb{P}$, then $[\![\mathcal{T}_1]\!]_{\text{sig}} = [\![\mathcal{T}_1']\!]_{\text{sig}}, c \rhd \Pi_{\overrightarrow{x}^n} [\![e]\!]_{\text{ter}}$. Using the induction hypothesis regarding $[\![\cdot]\!]_{\text{ter}}$, we get that this also equals $[\![\mathcal{T}_2]\!]_{\text{sig}}$. From the definition of $[\![\cdot]\!]_{\text{sig}}$, since c does not have the val subscript, by induction there is only one possible pre-image, so we conclude that $\mathcal{T}_1 \equiv \mathcal{T}_2$.

Next, if $e : \mathsf{U}_1$, then $[\![\mathcal{T}_1]\!]_{\text{sig}} = [\![\mathcal{T}_1']\!]_{\text{sig}}, c : \Pi_{\overrightarrow{x}^n} [\![e]\!]_{\text{ty}}$. Analogously to the previous case we use the induction hypothesis, regarding $[\![\cdot]\!]_{\text{ty}}$ this time, and then by induction there is only one pre-image, meaning $\mathcal{T}_1 \equiv \mathcal{T}_2$.

Finally, if e had neither type U_1 nor type $\mathbb{P}$, in our fragment it must have the form $\forall_{\overrightarrow{x:\alpha}^m \mathsf{U}_1}$, in which case $[\![\mathcal{T}_1]\!]_{\text{sig}} = [\![\mathcal{T}_1']\!]_{\text{sig}}, c : \Pi_{\overrightarrow{x}^n} \Pi_{\overrightarrow{x:[\![\alpha]\!]_{\text{ty}}}^m} : \mathsf{type}$. Using the induction hypothesis regarding $[\![\cdot]\!]_{\text{ty}}$ again and the definition of $[\![\cdot]\!]_{\text{sig}}$, we again conclude $\mathcal{T}_1 \equiv \mathcal{T}_2$.

If $T_1 = \mathcal{T}_1', (c, \alpha, e)$ than $[\![\mathcal{T}_1]\!]_{\text{sig}} = [\![\mathcal{T}_1', (c, \alpha)]\!]_{\text{sig}}, c_{\mathsf{val}} \rhd c =_{[\![\alpha]\!]_{\text{ty}}} [\![e]\!]_{\text{ter}}$. Using the induction hypotheses regarding $[\![\cdot]\!]_{\text{ty}}$ and $[\![\cdot]\!]_{\text{ter}}$, we get that this also equals $[\![\mathcal{T}_2]\!]_{\text{sig}}$. From the definition of $[\![\cdot]\!]_{\text{sig}}$, since c does have the val subscript, by induction there is only one possible pre-image, so we conclude that $\mathcal{T}_1 \equiv \mathcal{T}_2$.

$[\![\cdot]\!]_{\text{ctx}}$*:* Let Γ, Δ be well formed DTT3 contexts. We show that if $[\![\Gamma]\!]_{\text{ctx}} = [\![\Delta]\!]_{\text{ctx}}$ then $\Gamma \equiv \Delta$:

If $\Gamma = \circ$ we are again done analogously to before.

If $\Gamma = \Gamma', x : \mathsf{U}_1$, then $[\![\Gamma]\!]_{\text{ctx}} = [\![\Gamma']\!]_{\text{ctx}}, x : \mathsf{type} = [\![\Delta]\!]_{\text{ctx}}$. There is exactly one pre-image and we conclude $\Gamma \equiv \Delta$.

If $\Gamma = \Gamma', x : \mathbb{P}$, then $[\![\Gamma]\!]_{\text{ctx}} = [\![\Gamma']\!]_{\text{ctx}}, x : o = [\![\Delta]\!]_{\text{ctx}}$. Again from the definition of $[\![\cdot]\!]_{\text{ctx}}$ we conclude $\Gamma \equiv \Delta$.

If $\Gamma = \Gamma', x : e$, where e is neither $\mathbb{P}$ nor U_1, we need to do a case distinction on the type of e. It must either be $\mathbb{P}$ or U_1.

If $e : \mathbb{P}$, then $[\![\Gamma]\!]_{\text{ctx}} = [\![\Gamma']\!]_{\text{ctx}}, x \rhd [\![e]\!]_{\text{ter}}$. Using the mutual induction hypothesis regarding $[\![\cdot]\!]_{\text{ter}}$ we get that it is also equal to $[\![\Delta]\!]_{\text{ctx}}$. With the definition of $[\![\cdot]\!]_{\text{ctx}}$ we again conclude $\Gamma \equiv \Delta$.

If finally $e : \mathsf{U}_1$, then $[\![\Gamma]\!]_{\text{ctx}} = [\![\Gamma']\!]_{\text{ctx}}, x : [\![e]\!]_{\text{ty}}$. Analogous to the previous case, using the induction hypothesis regarding $[\![\cdot]\!]_{\text{ty}}$ and the definition of $[\![\cdot]\!]_{\text{ctx}}$ we conclude $\Gamma \equiv \Delta$. □

We also need to prove that $[\![\cdot]\!]_{\text{exp}}$ is compatible with substitution, meaning that substituting in the DTT3 expression and then translating leads to the same result as translating the expression that is substituted into as well as the expressions used in the substitution individually first, and performing the DHOL substitution after.

Lemma 2 (Substitution). *Let t be a translatable DTT3 expression, and let $\overrightarrow{e}$ be a list of translatable DTT3 expressions. Then $[\![t[\overrightarrow{e}]]\!]_{\text{exp}} = [\![t]\!]_{\text{exp}}[\overrightarrow{[\![e]\!]_{\text{exp}}}]$.*

Proof. The proof is by induction on t:

$t = x_i$*:* Independent of the type of x_i, $[\![x_i]\!]_{\text{exp}} = x_i$. Therefore $[\![x_i[\overrightarrow{e}]]\!]_{\text{exp}} = [\![e_i]\!]_{\text{exp}} = x_i[[\![e_i]\!]_{\text{exp}}] = [\![x_i]\!]_{\text{exp}}[[\![e_i]\!]_{\text{exp}}] = [\![x_i]\!]_{\text{exp}}[\overrightarrow{[\![e]\!]_{\text{exp}}}]$.

$t = \mathbb{P}$*:* Then $[\![t]\!]_{\text{exp}} = o$ and $t[\overrightarrow{e}] = t$, meaning $[\![t[\overrightarrow{e}]]\!]_{\text{exp}} = [\![t]\!]_{\text{exp}} = o = o[\overrightarrow{[\![e]\!]_{\text{exp}}}] = [\![t]\!]_{\text{exp}}[\overrightarrow{[\![e]\!]_{\text{exp}}}]$.

$t = c$*:* Analogous to previous case.

$t = u_1\ u_2$*:* Then $[\![t]\!]_{\text{exp}} = [\![u_1]\!]_{\text{exp}}\ [\![u_2]\!]_{\text{exp}}$ and also $t[\overrightarrow{e}] = u_1[\overrightarrow{e}]\ u_2[\overrightarrow{e}]$. So we get $[\![t[\overrightarrow{e}]]\!]_{\text{exp}} = [\![u_1[\overrightarrow{e}]]\!]_{\text{exp}}\ [\![u_2[\overrightarrow{e}]]\!]_{\text{exp}} \overset{I.H.}{=} [\![u_1]\!]_{\text{exp}}[\overrightarrow{[\![e]\!]_{\text{exp}}}]\ [\![u_2]\!]_{\text{exp}}[\overrightarrow{[\![e]\!]_{\text{exp}}}] = [\![t]\!]_{\text{exp}}[\overrightarrow{[\![e]\!]_{\text{exp}}}]$.

$t = \lambda_{x:\alpha} u$*:* This means $t \not: \mathsf{U}_1$, so $[\![t]\!]_{\text{exp}} = [\![t]\!]_{\text{ter}}$. Further $t[\overrightarrow{e}] = \lambda_{x:\alpha[\overrightarrow{e}]u[\overrightarrow{e}]}$. So we get $[\![t[\overrightarrow{e}]]\!]_{\text{ter}} = \lambda_{x:[\![\alpha[\overrightarrow{e}]]\!]_{\text{ty}}}[\![u[\overrightarrow{e}]]\!]_{\text{exp}} \overset{I.H.}{=} \lambda_{x:[\![\alpha]\!]_{\text{ty}}[\overrightarrow{[\![e]\!]_{\text{exp}}}]}[\![u]\!]_{\text{exp}}[\overrightarrow{[\![e]\!]_{\text{exp}}}] = [\![t]\!]_{\text{ter}}[\overrightarrow{[\![e]\!]_{\text{exp}}}]$. Since α must be of type U_1, it is equivalent to use $[\![\cdot]\!]_{\text{ty}}$ instead of $[\![\cdot]\!]_{\text{exp}}$.

$t = \forall_{x:\alpha_1} u$*:* There are two cases to consider: $t : \mathbb{P}$ and $t : \mathsf{U}_1$. If $t : \mathbb{P}$, then $[\![t]\!]_{\text{exp}} = [\![t]\!]_{\text{ter}} = \forall_{x:[\![\alpha]\!]_{\text{ty}}}[\![u]\!]_{\text{ter}}$ with $\alpha : \mathsf{U}_1$ and $u : \mathbb{P}$. Also $t[\overrightarrow{e}] = \forall_{x:\alpha[\overrightarrow{e}]} u[\overrightarrow{e}]$. So we get $[\![t[\overrightarrow{e}]]\!]_{\text{exp}} = [\![\forall_{x:\alpha[\overrightarrow{e}]} u[\overrightarrow{e}]]\!]_{\text{exp}} = \forall_{x:[\![\alpha[\overrightarrow{e}]]\!]_{\text{ty}}}[\![u[\overrightarrow{e}]]\!]_{\text{ter}} \overset{I.H.}{=} \forall_{x:[\![\alpha]\!]_{\text{ty}}[\overrightarrow{[\![e]\!]_{\text{exp}}}]}[\![u]\!]_{\text{ter}}[\overrightarrow{[\![e]\!]_{\text{exp}}}] = [\![t]\!]_{\text{exp}}[\overrightarrow{[\![e]\!]_{\text{exp}}}]$. If $t : \mathsf{U}_1$, then $[\![t]\!]_{\text{exp}} = [\![t]\!]_{\text{ty}} = \Pi_{x:[\![\alpha]\!]_{\text{ty}}}[\![u]\!]_{\text{ty}}$ with $\alpha : \mathsf{U}_1$ and $u : \mathsf{U}_1$. Also $t[\overrightarrow{e}] = \Pi_{x:\alpha[\overrightarrow{e}]} u[\overrightarrow{e}]$. So we get $[\![t[\overrightarrow{e}]]\!]_{\text{exp}} = [\![\Pi_{x:\alpha[\overrightarrow{e}]} u[\overrightarrow{e}]]\!]_{\text{exp}} = \Pi_{x:[\![\alpha[\overrightarrow{e}]]\!]_{\text{ty}}}[\![u[\overrightarrow{e}]]\!]_{\text{ty}} \overset{I.H.}{=} \Pi_{x:[\![\alpha]\!]_{\text{ty}}[\overrightarrow{[\![e]\!]_{\text{exp}}}]}[\![u]\!]_{\text{ty}}[\overrightarrow{[\![e]\!]_{\text{exp}}}] = [\![t]\!]_{\text{exp}}[\overrightarrow{[\![e]\!]_{\text{exp}}}]$ □

Table 2. Correspondence between DHOL and DTT3 judgements

DHOL	DTT3
$\Gamma \vdash^{\mathrm{DHOL}}_{T} F$	There exists p such that $\Gamma' \vdash^{\mathrm{DTT3}}_{\mathcal{T}} p : e$ and $[\![\Gamma']\!]^{\mathcal{T}}_{\mathrm{ctx}} = \Gamma, [\![e]\!]^{\Gamma';\mathcal{T}}_{\mathrm{ter}} = F, [\![\mathcal{T}]\!]_{\mathrm{sig}} = T$
$\Gamma \vdash^{\mathrm{DHOL}}_{T} t : A$	$\Gamma' \vdash^{\mathrm{DTT3}}_{\mathcal{T}} e : \alpha$ and $[\![\Gamma']\!]^{\mathcal{T}}_{\mathrm{ctx}} = \Gamma, [\![e]\!]^{\Gamma';\mathcal{T}}_{\mathrm{ter}} = t, [\![\alpha]\!]^{\Gamma';\mathcal{T}}_{\mathrm{ty}} = A, [\![\mathcal{T}]\!]_{\mathrm{sig}} = T$
$\vdash^{\mathrm{DHOL}} T$ Thy	$\vdash^{\mathrm{DTT3}} \mathcal{T}$ sig and $[\![\mathcal{T}]\!]_{\mathrm{sig}} = T$
$\vdash^{\mathrm{DHOL}}_{T} \Gamma$ Ctx	$\vdash^{\mathrm{DTT3}}_{\mathcal{T}} \Gamma'$ ok and $[\![\Gamma']\!]^{\mathcal{T}}_{\mathrm{ctx}} = \Gamma, [\![\mathcal{T}]\!]_{\mathrm{sig}} = T$
$\Gamma \vdash^{\mathrm{DHOL}}_{T} \Delta \leftarrow \delta$	$\Gamma' \vdash^{\mathrm{DTT3}}_{\mathcal{T}} \delta'_i : \alpha[\overrightarrow{\delta'}^{i-1}]$ when $\Delta'_i = x : \alpha[\overrightarrow{\delta'}^{i-1}]$ for $1 \leq i \leq \lvert\delta\rvert$ and $[\![\Gamma']\!]^{\mathcal{T}}_{\mathrm{ctx}} = \Gamma, [\![\Delta']\!]^{\mathcal{T}}_{\mathrm{ctx}} = \Delta, [\![\delta'_i]\!]^{\Gamma';\mathcal{T}}_{\mathrm{exp}} = \delta_i, [\![\mathcal{T}]\!]_{\mathrm{sig}} = T$
$\Gamma \vdash^{\mathrm{DHOL}}_{T} A :$ type	$\Gamma' \vdash^{\mathrm{DTT3}}_{\mathcal{T}} \alpha : \mathsf{U}_1$ and $[\![\Gamma']\!]^{\mathcal{T}}_{\mathrm{ctx}} = \Gamma, [\![\alpha]\!]^{\Gamma';\mathcal{T}}_{\mathrm{ty}} = A, [\![\mathcal{T}]\!]_{\mathrm{sig}} = T$
$\Gamma \vdash^{\mathrm{DHOL}}_{T} A \equiv B$	There exists p such that $\Gamma' \vdash^{\mathrm{DTT3}}_{\mathcal{T}} p : \alpha =_{\mathsf{U}_1} \beta$ and $[\![\Gamma']\!]^{\mathcal{T}}_{\mathrm{ctx}} = \Gamma, [\![\alpha]\!]^{\Gamma';\mathcal{T}}_{\mathrm{ty}} = A, [\![\beta]\!]^{\Gamma';\mathcal{T}}_{\mathrm{ty}} = B, [\![\mathcal{T}]\!]_{\mathrm{sig}} = T$

Next we state the reflection of judgements theorem. In general, we try to match the judgements of DHOL with those of DTT3. The cases where they notably differ are the ones for provability of a formula F and for the well-formedness of a substitution δ. For the former, we require the existence of a term of the corresponding proposition $[\![F]\!]_{\mathrm{ter}}$ in DTT3. For the latter, since DTT3 does not have explicit substitutions, we spell out the condition each of the components of δ needs to fulfil individually. This works since δ can simply be viewed as the list $\overrightarrow{\delta}^{n}$ of terms, types and $\checkmark_F$'s for some DHOL context Δ with corresponding types, type variables and local assumptions $\overrightarrow{\Delta}^{n}$ for some n.

Theorem 1 (Reflection of Judgements). *Let Γ, Δ be a DHOL context, A, B be DHOL types, t be a DHOL term, F be a DHOL formula, δ be a DHOL substitution and T be a DHOL theory. In Table 2, if the left hand side holds, then there exist translatable DTT3 contexts Γ', Δ', translatable DTT3 expressions $e, \alpha, \beta, \delta'_i$ and a translatable DTT3 signature $\mathcal{T}$ such that the right hand side also holds.*

We actually prove a slightly stronger version, where all DTT3 expressions have casts in front of every subexpression. This is necessary to fulfil the requirements of Lemma 1. The stronger condition can easily be fulfilled by adding casts that use a proof of reflexivity in front of subexpressions that do not have casts. We omit trivial casts for readability.

Since Lemma 1 only provides definitional equality of DTT3 constructs, we need to further invoke the compatibility of DTT3 judgements with definitional equality, i.e. subject reduction, meaning if some DTT3 judgement holds and there exist definitionally equal constructs, the same judgement also holds for these. This holds in the generalisation of DTT3 given by Carneiro [9], so it should hold for DTT3 as well.

For example regarding the typing judgement we have

$$\frac{\vdash \mathcal{T} \equiv \mathcal{T}' \qquad \vdash_{\mathcal{T}} \Gamma \equiv \Gamma' \qquad \Gamma \vdash_{\mathcal{T}} e \equiv e' \qquad \Gamma \vdash_{\mathcal{T}} \alpha \equiv \alpha' \qquad \vdash \mathcal{T}\ \mathsf{sig} \qquad \vdash_{\mathcal{T}} \Gamma\ \mathsf{ok} \qquad \Gamma \vdash_{\mathcal{T}} \alpha\ \mathsf{type} \qquad \Gamma \vdash_{\mathcal{T}} e : \alpha}{\Gamma' \vdash_{\mathcal{T}'} e' : \alpha'}$$

Similar statements for all other judgements hold.

Proof. We will use induction on the DHOL proof of our assumption.

Case ThyEmpty: Follows trivially from the first case of the sig judgement.

Case ThyTp: The assumptions of the rule are

$$(1) \vdash^{\mathrm{DHOL}} T\ \mathsf{Thy},\ (2)\ a \notin T,\ (3) \vdash^{\mathrm{DHOL}}_{T} \Delta\ \mathsf{Ctx},\ (4)\ \Delta = \overrightarrow{\alpha} : \mathsf{type}, \overrightarrow{x} : \overrightarrow{A}$$

The conclusion is (5) $\vdash^{\mathrm{DHOL}} T, a : \Pi_{\Delta} : \mathsf{type}\ \mathsf{Thy}$. The induction hypothesis allows us to use the assumptions (1) and (3) to derive that there exists a DTT3 signature $\mathcal{T}$ and a DTT3 context Δ', which have casts in front of every subexpression, such that $(1') \vdash^{\mathrm{DTT3}} \mathcal{T}\ \mathsf{sig}$ where $[\![\mathcal{T}]\!]_{\mathrm{sig}} = T$ and $(3') \vdash^{\mathrm{DTT3}}_{\mathcal{T}} \Delta'\ \mathsf{ok}$ where $[\![\mathcal{T}]\!]_{\mathrm{sig}} = T$, $[\![\Delta']\!]_{\mathrm{ctx}} = \Delta$. Note that we implicitly use the injectivity of the translation function stated in Lemma 2 to equate the $\mathcal{T}$ in $(1')$ and $(3')$. To prove this case of the theorem, we now need to show given $(1'), (2), (3'), (4)$ that there exist a DTT3 signature $\mathcal{T}'$ such that if (5) holds, it holds that $(5') \vdash^{\mathrm{DTT3}} \mathcal{T}'\ \mathsf{sig}$ with $[\![\mathcal{T}']\!]_{\mathrm{sig}} = T, a : \Pi_{\Delta} : \mathsf{type}$. We therefore assume (5) and show $(5')$. We choose $\mathcal{T}' = \mathcal{T}, (a, \forall_{\Delta'} \mathsf{U}_1)$, which translates to the theory $[\![\mathcal{T}]\!]_{\mathrm{sig}}, a : \Pi_{[\![\Delta']\!]_{\mathrm{ctx}}} : \mathsf{type}$, which is equal to $T, a : \Pi_{\Delta} : \mathsf{type}$ due to the equalities associated with $(1')$ and $(3')$. It remains to prove $\vdash^{\mathrm{DTT3}} \mathcal{T}, (a, \forall_{\Delta'} \mathsf{U}_1)\ \mathsf{sig}$. For this it suffices to show that $\vdash^{\mathrm{DTT3}}_{\mathcal{T}} \forall_{\Delta'} \mathsf{U}_1\ \mathsf{type}$. From the definition of the translation function, (4) and the equalities associated with $(3')$ we know that $\Delta' = \overrightarrow{\alpha} : \mathsf{U}_1, \overrightarrow{x} : \overrightarrow{\beta}$, where $[\![\beta_i]\!]_{\mathrm{ty}} = A_i$. To show our goal we use $\forall\mathsf{Form}$. It therefore suffices that $\overrightarrow{\alpha} : \mathsf{U}_1, \overrightarrow{x}^{i-1} : \overrightarrow{\beta}^{i-1} \vdash^{\mathrm{DTT3}}_{\mathcal{T}} \beta_i\ \mathsf{type}$ for all β_i. This then follows from $(3')$, which concludes the case.

In the following cases we will only state the results of applying the induction hypothesis to the DHOL preconditions (here, $(1'), (3')$), possible leftover assumptions (here, $(2), (4)$), the collection of equalities associated with these and state directly what we must prove (here, $(5')$), without necessarily mentioning the DHOL conclusion directly.

Case ThyCon: From the induction hypothesis, we get that $(1) \vdash \mathcal{T}\ \mathsf{sig}$ and $(2)\Delta' \vdash_{\mathcal{T}} \beta : \mathsf{U}_1$ with $[\![\mathcal{T}]\!]_{\mathrm{sig}} = T, [\![\Delta']\!]_{\mathrm{ctx}} = \Delta$ and $[\![\beta]\!]_{\mathrm{ty}} = A$. We further get the assumptions $c \notin T$ and $(3)\Delta = \overrightarrow{\alpha} : \mathsf{type}$. We need to show that $\vdash \mathcal{T}'\ \mathsf{sig}$, where $[\![\mathcal{T}']\!]_{\mathrm{sig}} = T, c : \Pi_{\Delta} A$. We choose $\mathcal{T}' = \mathcal{T}, (c, \forall_{\Delta'} \beta)$. $[\![\mathcal{T}']\!]_{\mathrm{sig}}$ computes as desired. It suffices to show $\vdash_{\mathcal{T}} \forall_{\Delta'} \beta\ \mathsf{type}$. We use $\forall\mathsf{Form}$, so we need to show $\Delta' \vdash_{\mathcal{T}} \beta\ \mathsf{type}$, which follows from (2), and $\overrightarrow{x}^{i-1} : \overrightarrow{\alpha'}^{i-1} \vdash_{\mathcal{T}} \alpha'_i\ \mathsf{type}$ for all α'_i in Δ'. We know $\Delta' = \overrightarrow{\alpha'} : \mathsf{U}_1$, due to (3) and the definition of the translation function. From this the desired result follows immediately.

Case ThyAsn: From the induction hypothesis, we get that (1) $\vdash \mathcal{T}$ sig and (2)$\Delta' \vdash_{\mathcal{T}} e : \beta$ with $[\![\mathcal{T}]\!]_{\text{sig}} = T, [\![\Delta']\!]_{\text{ctx}} = \Delta, [\![e]\!]_{\text{ter}} = F$ and $[\![\beta]\!]_{\text{ty}} = o$. We further get the assumptions $c \notin T$ and (3)$\Delta = \overrightarrow{\alpha} : \mathsf{type}$. We need to show that $\vdash \mathcal{T}'$ sig, where $[\![\mathcal{T}']\!]_{\text{sig}} = T, c \rhd \Pi_{\Delta} F$. We choose $\mathcal{T}' = \mathcal{T}, (c, \forall_{\Delta'} e)$. $[\![\mathcal{T}']\!]_{\text{sig}}$ computes as desired. The rest of the proof is analogous to the previous case, apart from showing $\vdash_{\mathcal{T}} e$ type, which follows from (2) and from $\beta = \mathbb{P}$, due to $[\![\beta]\!]_{\text{ty}} = o$ and the definition of the translation function.

Case CtxEmpty: Follows trivially from the first case of the ok judgement.

Case CtxTp: From the induction hypothesis, we get $\vdash_{\mathcal{T}} \Delta'$ ok with $[\![\Delta']\!]_{\text{ctx}} = \Delta$ and $[\![\mathcal{T}]\!]_{\text{sig}} = T$. We need to show that $\vdash_{\mathcal{T}} \Delta''$ ok with $[\![\Delta'']\!]_{\text{ctx}} = \Delta, \alpha : \mathsf{type}$. We choose $\Delta'' = \Delta', \alpha : \mathsf{U}_1$. $[\![\Delta'']\!]_{\text{ctx}}$ computes as desired. We then only need to show that $\vdash_{\mathcal{T}} \mathsf{U}_1$ type. This is then immediate.

Case CtxVar: From the induction hypothesis, we get $\vdash_{\mathcal{T}} \Delta'$ ok and $\Delta' \vdash_{\mathcal{T}} \alpha : \mathsf{U}_1$ with $[\![\Delta']\!]_{\text{ctx}} = \Delta, [\![\alpha]\!]_{\text{ty}} = A$ and $[\![\mathcal{T}]\!]_{\text{sig}} = T$. We need to show that $\vdash_{\mathcal{T}} \Delta''$ ok with $[\![\Delta'']\!]_{\text{ctx}} = \Delta, x : A$. We choose $\Delta'' = \Delta', x : \alpha$. $[\![\Delta'']\!]_{\text{ctx}}$ computes as desired. Thus we need to show $\vdash_{\mathcal{T}} \Delta'$ ok and $\Delta' \vdash_{\mathcal{T}} \alpha$ type, which follows immediately from our assumptions.

Case CtxAsn: From the induction hypothesis, we get $\vdash_{\mathcal{T}} \Delta'$ ok and $\Delta' \vdash_{\mathcal{T}} e : \alpha$ with $[\![\Delta']\!]_{\text{ctx}} = \Delta, [\![e]\!]_{\text{ter}} = F, [\![\alpha]\!]_{\text{ty}} = o$ and $[\![\mathcal{T}]\!]_{\text{sig}} = T$. We need to show that $\vdash_{\mathcal{T}} \Delta''$ ok with $[\![\Delta'']\!]_{\text{ctx}} = \Delta', x \rhd F$. We choose $\Delta'' = \Delta', x : e$. $[\![\Delta'']\!]_{\text{ctx}}$ computes as desired. Thus we need to show $\vdash_{\mathcal{T}} \Delta'$ ok and $\Delta' \vdash_{\mathcal{T}} e$ type. The former we already have as an assumption, the latter holds due to $[\![\alpha]\!]_{\text{ty}} = o$, which means that $\alpha = \mathbb{P}$.

Case SubEmpty: Trivial.

Case SubTp: From the induction hypothesis, we get (1)$\Gamma' \vdash_{\mathcal{T}} \delta'_i : \beta[\overrightarrow{\delta'}^{i-1}]$ when $\Delta'_i = x : \beta[\overrightarrow{\delta'}^{i-1}]$ and (2)$\Gamma' \vdash_{\mathcal{T}} \gamma : \mathsf{U}_1$ where $[\![\Gamma']\!]_{\text{ctx}} = \Gamma, [\![\Delta']\!]_{\text{ctx}} = \Delta, [\![\delta'_i]\!]_{\text{exp}} = \delta_i, [\![\gamma]\!]_{\text{ty}} = A$ and $[\![\mathcal{T}]\!]_{\text{sig}} = T$. We need to show that $\Gamma' \vdash_{\mathcal{T}} \delta''_i : \beta[\overrightarrow{\delta''}^{i-1}]$ when $\Delta''_i = x : \beta[\overrightarrow{\delta''}^{i-1}]$ where $[\![\Delta'']\!]_{\text{ctx}} = \Delta, \alpha : \mathsf{type}$ and $[\![\delta''_i]\!]_{\text{exp}} = \delta_i$ for all $1 \leq i \leq |\delta|$ and $[\![\delta''_{|\delta|+1}]\!]_{\text{exp}} = A$. We choose $\delta''_i = \delta'_i$ for $1 \leq i \leq |\delta|$ and $\delta''_{|\delta|+1} = \gamma$ and $\Delta'' = \Delta', \alpha : \mathsf{U}_1$. $[\![\delta''_i]\!]_{\text{exp}}$ and $[\![\Delta'']\!]_{\text{ctx}}$ compute as desired. For all $1 \leq i \leq |\delta|$, our goal is equal to (1). For $i = |\delta| + 1$, we need to show $\Gamma' \vdash_{\mathcal{T}} \gamma : \mathsf{U}_1$, which is exactly (2).

Case SubVar: From the induction hypothesis, we get (1)$\Gamma' \vdash_{\mathcal{T}} \delta'_i : \alpha[\overrightarrow{\delta'}^{i-1}]$ when $\Delta'_i = x : \alpha[\overrightarrow{\delta'}^{i-1}]$ and (2)$\Delta' \vdash_{\mathcal{T}} \beta : \mathsf{U}_1$ and (3)$\Gamma' \vdash_{\mathcal{T}} e : \gamma$ with $[\![\Gamma']\!]_{\text{ctx}} = \Gamma, [\![\Delta']\!]_{\text{ctx}} = \Delta, [\![\delta'_i]\!]_{\text{exp}} = \delta_i, [\![\beta]\!]_{\text{ty}} = A, [\![\gamma]\!]_{\text{ty}} = A[\delta], [\![e]\!]_{\text{ter}} = t$ and $[\![\mathcal{T}]\!]_{\text{sig}} = T$. The goal and choice is analogous to the previous case, with the exception of $\delta''_{|\delta|+1} = e$ and $\Delta'' = \Delta', x : \beta$. $[\![\delta''_i]\!]_{\text{exp}}$ and $[\![\Delta'']\!]_{\text{ctx}}$ compute as desired. For

all $1 \leq i \leq |\delta|$, our goal is equal to (1). For $i = |\delta| + 1$, we need to show $\Gamma' \vdash_{\mathcal{T}} e : \beta[\overrightarrow{\delta''}^{(|\delta|+1)-1}]$, which is equal to $\Gamma' \vdash_{\mathcal{T}} e : \beta[\delta']$. Using the induction hypotheses and Lemma 2 this then follows immediately.

Case SubAsn: From the induction hypothesis, we get (1) $\Gamma' \vdash_{\mathcal{T}} \delta'_i : \alpha[\overrightarrow{\delta'}^{i-1}]$ when $\Delta'_i = x : \alpha[\overrightarrow{\delta'}^{i-1}]$ and (2) $\Delta' \vdash_{\mathcal{T}} e : \beta$ and there exists p such that (3) $\Gamma' \vdash_{\mathcal{T}} p : e'$ with $[\![\Gamma']\!]_{\text{ctx}} = \Gamma, [\![\Delta']\!]_{\text{ctx}} = \Delta, [\![\delta'_i]\!]_{\text{exp}} = \delta_i, [\![\beta]\!]_{\text{ty}} = o, [\![e]\!]_{\text{ter}} = F, [\![e']\!]_{\text{ter}} = F[\delta]$ and $[\![\mathcal{T}]\!]_{\text{sig}} = T$. We need to show that $\Gamma' \vdash_{\mathcal{T}} \delta''_i : \beta[\overrightarrow{\delta''}^{i-1}]$ when $\Delta''_i = x : \beta[\overrightarrow{\delta''}^{i-1}]$ where $[\![\Delta'']\!]_{\text{ctx}} = \Delta, \alpha : \mathsf{type}$ and $[\![\delta''_i]\!]_{\text{exp}} = \delta_i$ for all $1 \leq i \leq |\delta|$ and $[\![\delta''_{|\delta|+1}]\!]_{\text{exp}} = p$. We choose $\delta''_i = \delta'_i$ for $1 \leq i \leq |\delta|$ and $\delta''_{|\delta|+1} = p$ and $\Delta'' = \Delta', x : e$. $[\![\delta''_i]\!]_{\text{exp}} = \delta_i$, for $1 \leq i \leq |\delta|$, and $[\![\delta_{|\delta|+1}]\!]_{\text{exp}} = \checkmark_{[\![e']\!]_{\text{ter}}} = \checkmark_e$. Further $[\![\Delta'']\!]_{\text{ctx}} = [\![\Delta']\!]_{\text{ctx}}, x \rhd e = \Delta, x \rhd e$. For all $1 \leq i \leq |\delta|$, our goal is equal to (1). For $i = |\delta| + 1$ we need to show $\Gamma' \vdash_{\mathcal{T}} p : e[\overrightarrow{\delta''}^{(|\delta|+1)-1}]$. This is equivalent to $\Gamma' \vdash_{\mathcal{T}} p : e[\delta']$. Using the induction hypotheses and Lemma 2 this then follows immediately.

Case TpSym: From the induction hypothesis, we get $\vdash_{\mathcal{T}} \Gamma'$ ok and (2)$\Gamma' \vdash_{\mathcal{T}} \delta'_i : \alpha[\overrightarrow{\delta'}^{i-1}]$ when $\Delta'_i = x : \alpha[\overrightarrow{\delta'}^{i-1}]$ with $[\![\Gamma']\!]_{\text{ctx}} = \Gamma, [\![\Delta']\!]_{\text{ctx}} = \Delta, [\![\delta'_i]\!]_{\text{exp}} = \delta_i$ and $[\![\mathcal{T}]\!]_{\text{sig}} = T$. We also get the assumption that (3) $a : \Pi_\Delta : \mathsf{type}$ is in T. We need to show that $\Gamma' \vdash_{\mathcal{T}} \beta : \mathsf{U}_1$ where $[\![\beta]\!]_{\text{ty}} = a\ \delta$. We choose $\beta = a\ \delta'$. $[\![\beta]\!]_{\text{ty}}$ computes as desired. From the definition of the translation function and (3), we know that $(a, \forall_\Delta \mathsf{U}_1) \in \mathcal{T}$. This, in combination with (2), then yields the desired result using Apply and ConstTy.

Case TermSym: From the induction hypothesis, we get $\vdash_{\mathcal{T}} \Gamma'$ ok and (2)$\Gamma' \vdash_{\mathcal{T}} \delta'_i : \alpha[\overrightarrow{\delta'}^{i-1}]$ when $\Delta'_i = x : \alpha[\overrightarrow{\delta'}^{i-1}]$ with $[\![\Gamma']\!]_{\text{ctx}} = \Gamma, [\![\Delta']\!]_{\text{ctx}} = \Delta, [\![\delta'_i]\!]_{\text{exp}} = \delta_i$ and $[\![\mathcal{T}]\!]_{\text{sig}} = T$. We also get the assumption that (3) $c : \Pi_\Delta B$ is in T. We need to show that $\Gamma' \vdash_{\mathcal{T}} e : \beta$ where $[\![e]\!]_{\text{ter}} = c\ \delta$ and $[\![\beta]\!]_{\text{ty}} = B[\delta]$. From the definition of the translation function and (3), we know that $(c, \forall_\Delta \beta') \in \mathcal{T}$ where $[\![\beta']\!]_{\text{ty}} = B$. We therefore choose $e = c\ \delta'$ and $\beta = \beta'[\delta']$. $[\![\beta]\!]_{\text{ty}}$ and $[\![e]\!]_{\text{ter}}$ compute as desired using Lemma 2. The desired result follows with (2) using Apply and ConstTy.

Case ValidSym: From the induction hypothesis, we get $\vdash_{\mathcal{T}} \Gamma'$ ok and (2)$\Gamma' \vdash_{\mathcal{T}} \delta'_i : \alpha[\overrightarrow{\delta'}^{i-1}]$ when $\Delta'_i = x : \alpha[\overrightarrow{\delta'}^{i-1}]$ with $[\![\Gamma']\!]_{\text{ctx}} = \Gamma, [\![\Delta']\!]_{\text{ctx}} = \Delta, [\![\delta'_i]\!]_{\text{exp}} = \delta_i$ and $[\![\mathcal{T}]\!]_{\text{sig}} = T$. We also get the assumption that (3) $c \rhd \Pi_\Delta F$ is in T. We need to show that there exists p such that $\Gamma' \vdash_{\mathcal{T}} p : e$ and $[\![e]\!]_{\text{ter}} = F[\delta]$. From the definition of the translation function and (3), we know that $(c, \forall_\Delta e') \in \mathcal{T}$ where $[\![e']\!]_{\text{ter}} = F$. We therefore choose $e = e'[\delta']$ and $p = c\ \delta'$. $[\![e]\!]_{\text{ter}}$ computes as desired using Lemma 2. The desired result follows with (2) using Apply and ConstTy.

Case TpVar: From the induction hypothesis, we have $\vdash_{\mathcal{T}} \Gamma'$ ok where $[\![\Gamma']\!]_{\text{ctx}} = \Gamma$ and $[\![\mathcal{T}]\!]_{\text{sig}} = T$. We also get the assumption that (2) $\alpha : \mathsf{type} \in \Gamma$. From this and the definition of the translation function we therefore know that (3) $\alpha : \mathsf{U}_1 \in \Gamma'$. We need to show that $\Gamma' \vdash_{\mathcal{T}} \alpha : \mathsf{U}_1$. This follows from the Assume rule and (3).

Case TermVar: Analogous to TpVar.

Case ValidVar: From the induction hypothesis, we have $\vdash_{\mathcal{T}} \Gamma'$ ok where $[\![\Gamma']\!]_{\text{ctx}} = \Gamma$ and $[\![\mathcal{T}]\!]_{\text{sig}} = T$. We also get the assumption that (2) $x \rhd F \in \Gamma$. From this and the definition of the translation function we therefore know that (3) $x : e \in \Gamma'$ where $[\![e]\!]_{\text{ter}} = F$. We need to show that there exists p such that $\Gamma' \vdash_{\mathcal{T}} p : e'$ and $[\![e']\!]_{\text{ter}} = F$. We choose $e' = e$ and $p = x$. The result then follows from the Assume rule and (3).

Case TpBool: Follows trivially from the Univ rule.

Case TermEq: From the induction hypothesis, we get that $\Gamma' \vdash_{\mathcal{T}} t' : \alpha$ and $\Gamma' \vdash_{\mathcal{T}} u' : \alpha$ where $[\![\Gamma']\!]_{\text{ctx}} = \Gamma, [\![t']\!]_{\text{ter}} = t, [\![u']\!]_{\text{ter}} = u, [\![\alpha]\!]_{\text{ty}} = A$ and $[\![\mathcal{T}]\!]_{\text{sig}} = T$. We need to show that $\Gamma' \vdash_{\mathcal{T}} e : \beta$ where $[\![e]\!]_{\text{ter}} = (t =_A u)$ and $[\![\beta]\!]_{\text{ty}} = o$. We choose $e = (t' =_\alpha u')$ and $\beta = \mathbb{P}$. $[\![e]\!]_{\text{ter}}$ and $[\![\beta]\!]_{\text{ty}}$ compute as desired. The result then immediately follows from the induction hypotheses and EqForm.

Case TermImpl: From the induction hypothesis, we have (1) $\Gamma' \vdash_{\mathcal{T}} e : \alpha$ and (2) $\Gamma'' \vdash_{\mathcal{T}} u : \alpha$ where $[\![\Gamma']\!]_{\text{ctx}} = \Gamma, [\![\Gamma'']\!]_{\text{ctx}} = \Gamma, x \rhd F, [\![e]\!]_{\text{ter}} = F, [\![u]\!]_{\text{ter}} = G, [\![\alpha]\!]_{\text{ty}} = o$ and $[\![\mathcal{T}]\!]_{\text{sig}} = T$. From the definition of the translation function we further know that $\Gamma'' = \Gamma', x : e$ and $\alpha = \mathbb{P}$. We need to show that $\Gamma' \vdash_{\mathcal{T}} t : \beta$ where $[\![t]\!]_{\text{ter}} = F \Rightarrow G$ and $[\![\beta]\!]_{\text{ty}} = o$. We choose $t = \forall_{x:e} u$ and $\beta = \alpha$. $[\![t]\!]_{\text{ter}}$ and $[\![\beta]\!]_{\text{ty}}$ compute as desired. We prove $\Gamma' \vdash_{\mathcal{T}} \forall_{x:e} u : \mathbb{P}$. We use $\forall$Form, meaning it suffices to show $\Gamma' \vdash_{\mathcal{T}} e$ type and $\Gamma', x : e \vdash_{\mathcal{T}} u : \mathbb{P}$ due to the definition of imax. Using (1) and (2) respectively with the fact that $\alpha = \mathbb{P}$ these follow immediately.

Case TpPi: From the induction hypothesis, we have $\Gamma' \vdash_{\mathcal{T}} \alpha : \mathsf{U}_1$ and $\Gamma'' \vdash_{\mathcal{T}} \beta : \mathsf{U}_1$ where $[\![\Gamma']\!]_{\text{ctx}} = \Gamma, [\![\Gamma'']\!]_{\text{ctx}} = \Gamma, x : A, [\![\alpha]\!]_{\text{ty}} = A, [\![\beta]\!]_{\text{ty}} = B$ and $[\![\mathcal{T}]\!]_{\text{sig}} = T$. From the definition of the translation function we further know that $\Gamma'' = \Gamma', x : \alpha$. We need to show that $\Gamma' \vdash_{\mathcal{T}} \gamma : \mathsf{U}_1$ where $[\![\gamma]\!]_{\text{ty}} = \Pi_{x:A} B$. We choose $\gamma = \forall_{x:\alpha} \beta$. $[\![\gamma]\!]_{\text{ty}}$ computes as desired. Using $\forall$Form, it suffices to show that $\Gamma' \vdash_{\mathcal{T}} \alpha : \mathsf{U}_1$ and $\Gamma', x : \alpha \vdash_{\mathcal{T}} \beta : \mathsf{U}_1$, which is exactly the induction hypotheses.

Case TermLambda: From the induction hypothesis, we have $\Gamma' \vdash_{\mathcal{T}} \alpha : \mathsf{U}_1$ and (2)$\Gamma'' \vdash_{\mathcal{T}} t' : \beta$ where $[\![\Gamma']\!]_{\text{ctx}} = \Gamma, [\![\Gamma'']\!]_{\text{ctx}} = \Gamma, x : A, [\![\alpha]\!]_{\text{ty}} = A, [\![\beta]\!]_{\text{ty}} = B, [\![t']\!]_{\text{ter}} = t$ and $[\![\mathcal{T}]\!]_{\text{sig}} = T$. From the definition of the translation function we further know that $\Gamma'' = \Gamma', x : \alpha$. We need to show that $\Gamma' \vdash_{\mathcal{T}} e : \gamma$ where $[\![e]\!]_{\text{ter}} = \lambda_{x:A} t$ and $[\![\gamma]\!]_{\text{ty}} = \Pi_{x:A} B$. We choose $e = \lambda_{x:\alpha} t'$ and $\gamma = \forall_{x:\alpha} \beta$. Using λForm we need to show $\Gamma', x : \alpha \vdash_{\mathcal{T}} t' : \beta$, which is exactly (2).

Case TermApply: From the induction hypothesis, we have $\Gamma' \vdash_{\mathcal{T}} t' : \gamma$ and $\Gamma' \vdash_{\mathcal{T}} u' : \alpha$ where $[\![\Gamma']\!]_{\text{ctx}} = \Gamma, [\![\alpha]\!]_{\text{ty}} = A, [\![\gamma]\!]_{\text{ty}} = \Pi_{x:A} B, [\![t']\!]_{\text{ter}} = t, [\![u']\!]_{\text{ter}} = u$ and $[\![\mathcal{T}]\!]_{\text{sig}} = T$. From the definition of the translation function we also know that $\gamma = \forall_{x:\alpha} \beta$, where $[\![\beta]\!]_{\text{ty}} = B$. We need to show that $\Gamma' \vdash_{\mathcal{T}} e : \beta'$ where $[\![e]\!]_{\text{ter}} = t\ u, [\![\beta']\!]_{\text{ty}} = B[u]$. We choose $e = t'\ u'$ and $\beta' = \beta[u']$. $[\![\beta']\!]_{\text{ty}}$ and $[\![e]\!]_{\text{ter}}$ compute as desired using Lemma 2. Using Apply, it suffices to show $\Gamma' \vdash_{\mathcal{T}} t' : \forall_{x:\alpha} \beta$ and $\Gamma' \vdash_{\mathcal{T}} u' : \alpha$, which are exactly the induction hypotheses.

Case TpEqVar: Follows trivially from $\mathsf{EqIntro}$.

Case TpEqSym: From the induction hypothesis, we have $\vdash_{\mathcal{T}} \Gamma' \ \mathsf{ok}$ and that there exists p such that $\Gamma' \vdash_{\mathcal{T}} p : \gamma_i =_{\Delta'_i[\overrightarrow{\gamma}^{i-1}]} \gamma'_i$ since $\equiv_\Delta$ means the component-wise equality of the terms/types, where $[\![\Gamma']\!]_{\text{ctx}} = \Gamma, [\![\Delta']\!]_{\text{ctx}} = \Delta, [\![\gamma_i]\!]_{\text{exp}} = \delta_i, [\![\gamma'_i]\!]_{\text{exp}} = \delta'_i$ and $[\![\mathcal{T}]\!]_{\text{sig}} = T$. We also get the assumption that $(3)a : \Pi_\Delta$ type is in T. From the definition of the translation function we then know that $(a, \forall'_\Delta \mathsf{U}_1) \in \mathcal{T}$. We need to show that there exists p such that $\Gamma' \vdash_{\mathcal{T}} p' : \alpha =_{\mathsf{U}_1} \beta$ and $[\![\alpha]\!]_{\text{ty}} = a\ \delta$ and $[\![\beta]\!]_{\text{ty}} = a\ \delta'$. We choose $\alpha = a\ \gamma, \beta = a\ \gamma'$. $[\![\alpha]\!]_{\text{ty}}$ and $[\![\beta]\!]_{\text{ty}}$ compute as desired. Using the congruence rules about equality derivable in DTT3 and the induction hypotheses we can construct p'.

Case TpEqBool: Follows trivially from the introduction rule of equality.

Case TpEqPi: From the induction hypothesis, we have that there exists p_1 such that $\Gamma' \vdash_{\mathcal{T}} p_1 : \alpha =_{\mathsf{U}_1} \alpha'$ and that there exists p_2 such that $(2)\Gamma'' \vdash_{\mathcal{T}} p_2 : \beta =_{\mathsf{U}_1} \beta'$ where $[\![\Gamma']\!]_{\text{ctx}} = \Gamma, [\![\Gamma'']\!]_{\text{ctx}} = \Gamma, x : A, [\![\alpha]\!]_{\text{ty}} = A, [\![\alpha']\!]_{\text{ty}} = A', [\![\beta]\!]_{\text{ty}} = B, [\![\beta]\!]_{\text{ty}} = B'$ and $[\![\mathcal{T}]\!]_{\text{sig}} = T$. From the definition of the translation function we further know that $\Gamma'' = \Gamma', x : \alpha$. We need to show that there exists p such that $\Gamma' \vdash_{\mathcal{T}} p : \gamma =_{\mathsf{U}_1} \gamma'$ and $[\![\gamma]\!]_{\text{ty}} = \Pi_{x:A} B$ and $[\![\gamma']\!]_{\text{ty}} = \Pi_{x:A'} B$. We choose $\gamma = \forall_{x:\alpha} \beta$ and $\gamma' = \forall_{x:\alpha'} \beta'$. After this everything type-checks. $[\![\gamma]\!]_{\text{ty}}$ and $[\![\gamma']\!]_{\text{ty}}$ compute as desired. Using the common derivable rules for equality in DTT3 and the induction hypotheses we can construct p.

Case TermConvert: From the induction hypothesis, we have $\Gamma' \vdash_{\mathcal{T}} t' : \alpha$ and that there exists p such that $(2)\Gamma' \vdash_{\mathcal{T}} p : \alpha =_{\mathsf{U}_1} \alpha'$ where $[\![\Gamma']\!]_{\text{ctx}} = \Gamma, [\![\Delta']\!]_{\text{ctx}} = \Delta, [\![\alpha]\!]_{\text{ty}} = A, [\![\alpha']\!]_{\text{ty}} = A', [\![t']\!]_{\text{ter}} = t$ and $[\![\mathcal{T}]\!]_{\text{sig}} = T$. We need to show that $\Gamma' \vdash_{\mathcal{T}} e : \beta$ where $[\![e]\!]_{\text{ter}} = t$ and $[\![\beta]\!]_{\text{ty}} = A'$. We choose $e = t''$ and $\beta = \alpha'$. t'' is t' where the top-level cast is replaced by one that uses p. $[\![e]\!]_{\text{ter}}$ and $[\![\beta]\!]_{\text{ty}}$ compute as desired. Due to the replacement of the cast our goal follows immediately.

Case TermBeta: From the induction hypothesis, we get $\Gamma' \vdash_{\mathcal{T}} e : \gamma$ and $\Gamma' \vdash_{\mathcal{T}} u' : \alpha$ where $[\![\Gamma']\!]_{\text{ctx}} = \Gamma, [\![\gamma]\!]_{\text{ty}} = \Pi_{x:A} B, [\![\alpha]\!]_{\text{ty}} = A, [\![e]\!]_{\text{ter}} = \lambda_{x:A} t, [\![u']\!]_{\text{ter}} = u$ and $[\![\mathcal{T}]\!]_{\text{sig}} = T$. From the definition of the translation function we further know that $\gamma = \forall_{x:\alpha} \beta$ where $[\![\beta]\!]_{\text{ty}} = B$ and $e = \lambda_{x:\alpha} t'$ where $[\![t']\!]_{\text{ter}} = t$. We need to show that there exists p such that $\Gamma' \vdash_{\mathcal{T}} p : e'$ and $[\![e']\!]_{\text{ter}} = (\lambda_{x:A} t)\ u =_{B[u]} t[u]$. We choose $e' = (\lambda_{x:\alpha} t')\ u' =_{\beta[u']} t'[u']$ and $p = refl_{t'[u']}$. $[\![e']\!]_{\text{ter}}$ computes as desired using Lemma 2. This holds definitionally in DTT3 so it is proved by $refl$.

Case TermEta: From the induction hypothesis, we get $\Gamma' \vdash_{\mathcal{T}} t' : \gamma$ where $[\![\Gamma']\!]_{\text{ctx}} = \Gamma, [\![\gamma]\!]_{\text{ty}} = \Pi_{x:A} B, [\![t']\!]_{\text{ter}} = t$ and $[\![\mathcal{T}]\!]_{\text{sig}} = T$. From the definition of the translation function we further know that $\gamma = \forall_{x:\alpha} \beta$ where $[\![\alpha]\!]_{\text{ty}} = A$ and $[\![\beta]\!]_{\text{ty}} = B$. We additionally get the assumption that x does not occur in Γ. It therefore can also not occur in Γ' or we can perform α-renaming to achieve this. We need to show that there exists p such that $\Gamma' \vdash_{\mathcal{T}} p : e$ and

$[\![e]\!]_{\text{ter}} = t =_{\Pi_{x:A}B} \lambda_{x:A} t\ x$. We choose $e = t' =_\gamma \lambda_{x:\alpha} t'\ x$ and $p = refl_{t'}$. $[\![e]\!]_{\text{ter}}$ computes as desired. Again this equation holds definitionally in DTT3 so it is proved by $refl$.

Case CongApply: From the induction hypothesis, we get that there exists p_1 such that $\Gamma' \vdash_{\mathcal{T}} p_1 : e$ and that there exists p such that $\Gamma \vdash_{\mathcal{T}} p_2 : e'$ where $[\![\Gamma']\!]_{\text{ctx}} = \Gamma, [\![e]\!]_{\text{ter}} = t =_A t', [\![e']\!]_{\text{ter}} = f =_{\Pi_{x:A}B} f'$ and $[\![\mathcal{T}]\!]_{\text{sig}} = T$. From the definition of the translation function we further know that $e = (u =_\alpha u')$ and $e' = (g =_{\forall_{x:\alpha}\beta} g')$ where $[\![u]\!]_{\text{ter}} = t, [\![u']\!]_{\text{ter}} = t', [\![g]\!]_{\text{ter}} = f, [\![g']\!]_{\text{ter}} = f', [\![\alpha]\!]_{\text{ty}} = A$ and $[\![\beta]\!]_{\text{ty}} = B$. We need to show that there exists p such that $\Gamma' \vdash_{\mathcal{T}} p : e''$ and $[\![e'']\!]_{\text{ter}} = (f\ t =_{B[t]} f'\ t')$. We choose $e'' = (g\ u =_{\beta[u]} \mathsf{cast}\ p_1'\ (g'\ u'))$. The cast added around the right hand side of the equation is needed to make this statement type-check. The proof term $p_1' : \beta[u] = \beta[u']$ can easily be derived using p_1 and the rules for equality. Then $[\![e'']\!]_{\text{ter}}$ computes as desired using Lemma 2. Then using induction twice yields the desired p.

Case CongLambda: From the induction hypothesis, we get that there exists p_1 such that $\Gamma' \vdash_{\mathcal{T}} p_1 : \alpha =_{\mathsf{U}_1} \alpha'$ and that there exists p_2 such that $\Gamma'' \vdash_{\mathcal{T}} p_2 : e$ where $[\![\Gamma']\!]_{\text{ctx}} = \Gamma, [\![\Gamma'']\!]_{\text{ctx}} = \Gamma, x : A, [\![\alpha]\!]_{\text{ty}} = A, [\![\alpha']\!]_{\text{ty}} = A', [\![e]\!]_{\text{ter}} = t =_B t'$ and $[\![\mathcal{T}]\!]_{\text{sig}} = T$. From the definition of the translation function we further know that $e = u =_\beta u'$ and $\Gamma'' = \Gamma', x : \alpha$ where $[\![u]\!]_{\text{ter}} = t, [\![u']\!]_{\text{ter}} = t'$ and $[\![\beta]\!]_{\text{ty}} = B$. We need to show that there exists p such that $\Gamma' \vdash_{\mathcal{T}} p : e'$ and $[\![e']\!]_{\text{ter}} = \lambda_{x:A} t =_{\Pi_{x:A}B} \lambda_{x:A'} t'$. We choose $e' = \lambda_{x:\alpha} u =_{\forall_{x:\alpha}\beta} \lambda_{x:\alpha'} u'$. $[\![e]\!]_{\text{ter}}$ computes as desired. To make this type-check, all occurrences of x in u' must have their casts amended by p_1. Then using induction twice yields the desired p.

Case Refl: Follows trivially from the introduction rule of equality.

Case Sym: From the induction hypothesis, we get there exists p such that $\Gamma' \vdash_{\mathcal{T}} p : e$ where $[\![\Gamma']\!]_{\text{ctx}} = \Gamma, [\![e]\!]_{\text{ter}} = t =_A s$ and $[\![\mathcal{T}]\!]_{\text{sig}} = T$. From the definition of the translation function we further know that $e = t' =_\alpha s'$ with $[\![t']\!]_{\text{ter}} = t, [\![s']\!]_{\text{ter}} = s$ and $[\![\alpha]\!]_{\text{ty}} = A$ We need to show that there exists p' such that $\Gamma' \vdash_{\mathcal{T}} p' : e'$ and $[\![e']\!]_{\text{ter}} = s =_A t$. We choose $e' = t' =_\alpha s'$. $[\![e']\!]_{\text{ter}}$ computes as desired. Using induction on p we can construct the desired p'.

Case CongValid: From the induction hypothesis, we get that there exists p_1 such that $\Gamma' \vdash_{\mathcal{T}} p_1 : e$ and that there exists p_2 such that $\Gamma' \vdash_{\mathcal{T}} p_2 : e'$ where $[\![\Gamma']\!]_{\text{ctx}} = \Gamma, [\![e]\!]_{\text{ter}} = F =_o F', [\![e']\!]_{\text{ter}} = F'$ and $[\![\mathcal{T}]\!]_{\text{sig}} = T$. From the definition of the translation function we further know that $e = t =_{\mathbb{P}} e'$ where $[\![t]\!]_{\text{ter}} = F$. We need to show that there exists p such that $\Gamma' \vdash_{\mathcal{T}} p : t$. We can construct p by changing the cast around p_2 to one that casts along p_1.

Case ImplIntro: From the induction hypothesis, we get that there exists p such that $\Gamma' \vdash_{\mathcal{T}} p : e$ where $[\![\Gamma']\!]_{\text{ctx}} = \Gamma, x \rhd F, [\![e]\!]_{\text{ter}} = G$ and $[\![\mathcal{T}]\!]_{\text{sig}} = T$. From the definition of the translation function we further know that $\Gamma' = \Gamma'', x : e'$ where $[\![\Gamma'']\!]_{\text{ctx}} = \Gamma$ and $[\![e']\!]_{\text{ter}} = F$. We need to show that there exists p' such that

$\Gamma'' \vdash_{\mathcal{T}} p' : u$ and $[\![u]\!]_{\text{ter}} = F \Rightarrow G$. We choose $u = \forall_{x:e'} e$ and $p' = \lambda_{x:e'} p$. $[\![u]\!]_{\text{ter}}$ computes as desired. Using $\lambda\mathsf{Form}$ with the induction hypothesis immediately yields the result.

Case MP: From the induction hypothesis, we get that there exists p_1 such that $\Gamma' \vdash_{\mathcal{T}} p_1 : e$ and that there exists p_2 such that $\Gamma' \vdash_{\mathcal{T}} p_2 : e'$ where $[\![\Gamma']\!]_{\text{ctx}} = \Gamma, [\![e]\!]_{\text{ter}} = F \Rightarrow G, [\![e']\!]_{\text{ter}} = F$ and $[\![\mathcal{T}]\!]_{\text{sig}} = T$. From the definition of the translation function we further know that $e = \forall_{x:e'} e''$ where $[\![e'']\!]_{\text{ter}} = G$. We need to show that there exists p such that $\Gamma' \vdash_{\mathcal{T}} p : e''$. We choose $p = p_1\ p_2$. Using Apply with the induction hypotheses immediately yields the result.

Case BoolExt: From the induction hypothesis, we get $\Gamma' t' : \alpha$, that there exists p_1 such that $\Gamma' \vdash_{\mathcal{T}} p_1 : e$ and that there exists p_2 such that $\Gamma' \vdash_{\mathcal{T}} p_2 : e'$ where $[\![\Gamma']\!]_{\text{ctx}} = \Gamma, [\![t']\!]_{\text{ter}} = t, [\![e]\!]_{\text{ter}} = t\ \bot, [\![e']\!]_{\text{ter}} = t\ \top, [\![\alpha]\!]_{\text{ty}} = \Pi_{x:o} o$ and $[\![\mathcal{T}]\!]_{\text{sig}} = T$. From the definition of the translation function we further know that $\alpha = \forall_{x:\mathbb{P}} \mathbb{P}$, that $e = t'\ \bot$ and that $e' = t'\ \top$. As stated earlier for this case we assume axiomatisations of a constant $\vdash_{\mathcal{T}} \mathsf{lem} : \forall_{e:\mathbb{P}} e \vee (\neg e)$, the empty type $\bot$, the singleton type $\top$, the co-product $\vee$ and negation $\neg$. We need to prove that there exists p such that $\Gamma'' \vdash_{\mathcal{T}} p : e''$, where $[\![\Gamma'']\!]_{\text{ctx}} = \Gamma, x : o, [\![e'']\!]_{\text{ter}} = t\ x$. We choose $\Gamma'' = \Gamma', x : \mathbb{P}$, $e'' = t'\ x$. Then $[\![\Gamma'']\!]_{\text{ctx}}$ and $[\![e'']\!]_{\text{ter}}$ compute as desired. To acquire p, we use the induction principle associated with $\vee$ on $\mathsf{lem}\ x$ with e and e', which yields the desired result. □

We do not prove the cases regarding $\top$, $\bot$, $\wedge$, $\vee$, $\neg$ and $\exists$, since they depend on some design decisions when axiomatising their counterparts in DTT. For commonly used definitions, however, these easily follow in the same style as the other cases.

From Theorem 1, we easily derive soundness:

Theorem 2 (Soundness). *Let $\mathcal{T}$ be a translatable DTT3 signature, let Γ be a translatable DTT3 context, let e be a translatable DTT3 expression such that $\Gamma \vdash_{\mathcal{T}} e : \mathbb{P}$. If $[\![\Gamma]\!]_{\text{ctx}} \vdash^{\text{DHOL}}_{[\![\mathcal{T}]\!]_{\text{sig}}} [\![e]\!]_{\text{ter}}$, then there exists p such that $\Gamma \vdash^{\text{DTT3}}_{\mathcal{T}} p : e$.*

Proof. Consequence of the first non-header row of Table 2 and injectivity together with subject reduction. □

6 Related Work

Our work extends the recent research on DHOL. Rothgang et al. [28] introduced the original monomorphic DHOL, including a sound and complete translation to monomorphic HOL. DHOL is reminiscent of the formalisms underlying the PVS [22] and Nuprl [12] proof assistants. In a follow-up unpublished manuscript, Ranalter et al. [25] generalised the logic and the translation to support rank 1 polymorphism. This generalised logic is the target of our own translation.

Rothgang and Rabe [29] also extended monomorphic DHOL with subtyping and adapted the translation to HOL accordingly. Moreover, Ranalter et al. [26]

described an extension of monomorphic DHOL with Hilbert's choice operator and adapted the translation to HOL. On the theorem proving front, Niederhauser et al. [20] designed a sound and complete tableau style proof calculus for monomorphic DHOL and implemented it in the Lash prover. Finally, Ranalter et al. [27] extended the TPTP (Thousands of Problems for Theorem Provers) syntax and infrastructure with support for polymorphic DHOL.

Apart from the aforementioned translations from variants of DHOL to HOL, translations from DTTs are also relevant to our work. Jacobs and Melham [15] encoded Martin-Löf type theory into polymorphic HOL. In the context of hammers, Czajka and Kaliszyk [13] and Qian et al. [24] developed pragmatic translations from DTT to FOL and HOL, respectively. Oury [21], Winterhalter et al. [32] and Vaishnav [30] all translated extensional DTTs to intensional DTTs. These three approaches therefore take an inverse direction to our work, which translates from an intensional logic to an extensional logic.

7 Conclusion

We presented a translation from a fragment of DTT to polymorphic DHOL. The translation is sound and lightweight. The translatable fragment constitutes a large part of DTT as used in practice. The translation could be used to integrate DHOL based provers into proof assistants based on DTT as part of a hammer.

There are several avenues for future work. First, we should try to prove completeness. Second, we want to extend the translation to support more advanced features of DTT (e.g. inductive definitions and quotient types). Third, we would like to formalise the soundness argument using a proof assistant. Finally, to reach users we should implement the translation in a hammer system.

Acknowledgements. We thank Massin Guerdi for invaluable discussions on the definitions and theorems. We thank Mark Summerfield for helpful textual suggestions.

Blanchette's research was co-funded by the European Union (ERC, Nekoka, 101083038). Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council. Neither the European Union nor the granting authority can be held responsible for them.

References

1. Andrews, P.: An Introduction to Mathematical Logic and Type Theory, Applied Logic Series, vol. 27. Springer Dordrecht (2002)
2. Angiuli, C., Gratzer, D.: Principles of dependent type theory (2025), to appear, Cambridge University Press
3. Asperti, A., Ricciotti, W., Coen, C.S., Tassi, E.: The matita interactive theorem prover. In: Bjørner, N.S., Sofronie-Stokkermans, V. (eds.) CADE-23. LNCS, vol. 6803, pp. 64–69. Springer (2011). https://doi.org/10.1007/978-3-642-22438-6_7

4. Barendregt, H., Dekkers, W., Statman, R.: Lambda Calculus with Types. Perspectives in Logic, Cambridge University Press (2013)
5. Bertot, Y., Castéran, P.: Interactive Theorem Proving and Program Development – Coq'Art: The Calculus of Inductive Constructions. Texts in Theoretical Computer Science. An EATCS Series, Springer (2004). https://doi.org/10.1007/978-3-662-07964-5
6. Blanchette, J.C., Böhme, S., Popescu, A., Smallbone, N.: Encoding monomorphic and polymorphic types. Log. Methods Comput. Sci. **12**(4) (2016). https://doi.org/10.2168/LMCS-12(4:13)2016
7. Blanchette, J.C., Kaliszyk, C., Paulson, L.C., Urban, J.: Hammering towards QED. J. Formaliz. Reason. **9**(1), 101–148 (2016). https://doi.org/10.6092/ISSN.1972-5787/4593
8. Brown, C.E.: Satallax: An automatic higher-order prover. In: Gramlich, B., Miller, D., Sattler, U. (eds.) IJCAR 2012. LNCS, vol. 7364, pp. 111–117. Springer (2012). https://doi.org/10.1007/978-3-642-31365-3_11
9. Carneiro, M.: The Type Theory of Lean. Master's thesis, Carnegie Mellon University (2019), https://github.com/digama0/lean-type-theory/releases/tag/v1.0
10. Church, A.: A formulation of the simple theory of types. J. Symb. Log. **5**(2), 56–68 (1940). https://doi.org/10.2307/2266170
11. Cohen, C.: Pragmatic quotient types in coq. In: Blazy, S., Paulin-Mohring, C., Pichardie, D. (eds.) Interactive Theorem Proving - 4th International Conference, ITP 2013, Rennes, France, July 22-26, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7998, pp. 213–228. Springer (2013). https://doi.org/10.1007/978-3-642-39634-2_17, https://doi.org/10.1007/978-3-642-39634-2_17
12. Constable, R.L., Allen, S.F., Bromley, M., Cleaveland, R., Cremer, J.F., Harper, R., Howe, D.J., Knoblock, T.B., Mendler, N.P., Panangaden, P., Sasaki, J.T., Smith, S.F.: Implementing Mathematics with the Nuprl Proof Development System. Prentice Hall (1986), http://dl.acm.org/citation.cfm?id=10510
13. Czajka, L., Kaliszyk, C.: Hammer for Coq: Automation for dependent type theory. J. Autom. Reason. **61**(1-4), 423–453 (2018). https://doi.org/10.1007/S10817-018-9458-4
14. Gordon, M.J.C.: Introduction to the HOL system. In: Archer, M., Joyce, J.J., Levitt, K.N., Windley, P.J. (eds.) 1991 International Workshop on the HOL Theorem Proving System and Its Applications. pp. 2–3. IEEE Computer Society (1991)
15. Jacobs, B., Melham, T.F.: Translating dependent type theory into higher order logic. In: Bezem, M., Groote, J.F. (eds.) TLCA '93. LNCS, vol. 664, pp. 209–229. Springer (1993). https://doi.org/10.1007/BFB0037108
16. Maio, L., Bentkamp, A., Blanchette, J., Tourret, S.: From dependent type theory to dependently typed higher order logic (technical report) (2026), https://nekoka-project.github.io/pubs/dtt-to-dhol-report.pdf
17. Meng, J., Paulson, L.C.: Translating higher-order clauses to first-order clauses. J. Autom. Reason. **40**(1), 35–60 (2008). https://doi.org/10.1007/S10817-007-9085-Y
18. de Moura, L., Ullrich, S.: The Lean 4 theorem prover and programming language. In: Platzer, A., Sutcliffe, G. (eds.) CADE-28. LNCS, vol. 12699, pp. 625–635. Springer (2021). https://doi.org/10.1007/978-3-030-79876-5_37
19. Nederpelt, R., Geuvers, H.: Type Theory and Formal Proof: An Introduction. Cambridge University Press (2014)
20. Niederhauser, J., Brown, C.E., Kaliszyk, C.: Tableaux for automated reasoning in dependently-typed higher-order logic. In: Benzmüller, C., Heule, M.J.H., Schmidt, R.A. (eds.) IJCAR 2024, Part I. LNCS, vol. 14739, pp. 86–104. Springer (2024). https://doi.org/10.1007/978-3-031-63498-7_6

21. Oury, N.: Extensionality in the calculus of constructions. In: Hurd, J., Melham, T.F. (eds.) TPHOLs 2005. LNCS, vol. 3603, pp. 278–293. Springer (2005). https://doi.org/10.1007/11541868_18
22. Owre, S., Rajan, S., Rushby, J.M., Shankar, N., Srivas, M.K.: PVS: Combining specification, proof checking, and model checking. In: Alur, R., Henzinger, T.A. (eds.) CAV '96. LNCS, vol. 1102, pp. 411–414. Springer (1996). https://doi.org/10.1007/3-540-61474-5_91
23. Pfenning, F., Paulin-Mohring, C.: Inductively defined types in the calculus of constructions. In: Main, M.G., Melton, A., Mislove, M.W., Schmidt, D.A. (eds.) MFPS 1989. LNCS, vol. 442, pp. 209–228. Springer (1989). https://doi.org/10.1007/BFB0040259
24. Qian, Y., Clune, J., Barrett, C.W., Avigad, J.: Lean-auto: An interface between Lean 4 and automated theorem provers. CoRR **abs/2505.14929** (2025). https://doi.org/10.48550/ARXIV.2505.14929
25. Ranalter, D., Rabe, F., Kaliszyk, C.: Polymorphic theorem proving for DHOL (2025), unpublished manuscript
26. Ranalter, D., Brown, C., Kaliszyk, C.: Experiments with choice in dependently-typed higher-order logic. In: Bjørner, N., Heule, M., Voronkov, A. (eds.) Proceedings of 25th Conference on Logic for Programming, Artificial Intelligence and Reasoning. EPiC Series in Computing, vol. 100, pp. 311–320. EasyChair (2024). https://doi.org/10.29007/2v8h, /publications/paper/lRJ1
27. Ranalter, D., Kaliszyk, C., Rabe, F., Sutcliffe, G.: The dependently typed higher-order form for the TPTP world. In: Thiemann, R., Weidenbach, C. (eds.) FroCoS 2025. LNCS, vol. 15979, pp. 287–305. Springer (2025). https://doi.org/10.1007/978-3-032-04167-8_16
28. Rothgang, C., Rabe, F., Benzmüller, C.: Dependently-typed higher-order logic. ACM Trans. Comput. Log. (2025), to appear
29. Rothgang, C., Rabe, F.: Subtyping in dependently-typed higher-order logic. In: Thiemann, R., Weidenbach, C. (eds.) FroCoS 2025. LNCS, vol. 15979, pp. 98–114. Springer (2025). https://doi.org/10.1007/978-3-032-04167-8_6
30. Vaishnav, R.: Lean4less: Eliminating definitional equalities from lean via an extensional-to-intensional translation. In: Liu, Z., Saoud, A., Wehrheim, H. (eds.) Theoretical Aspects of Computing – ICTAC 2025. pp. 202–219. Springer Nature Switzerland, Cham (2026)
31. Werner, B.: Sets in types, types in sets. In: Abadi, M., Ito, T. (eds.) Theoretical Aspects of Computer Software. pp. 530–546. Springer Berlin Heidelberg, Berlin, Heidelberg (1997)
32. Winterhalter, T., Sozeau, M., Tabareau, N.: Eliminating reflection from type theory. In: Proceedings of the 8th ACM SIGPLAN International Conference on Certified Programs and Proofs. p. 91–103. CPP 2019, Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3293880.3294095, https://doi.org/10.1145/3293880.3294095