Experiments with Choice
in Dependently-Typed Higher-Order Logic

Daniel Ranalter[1], Chad E. Brown[2], and Cezary Kaliszyk[1]

[1] University of Innsbruck, Innsbruck, Austria
`{daniel.ranalter,cezary.kaliszyk}@uibk.ac.at`
[2] Czech Technical University in Prague, Prague, Czech Republic

Abstract

Recently an extension to higher-order logic — called DHOL — was introduced, enriching the language with dependent types, and creating a powerful extensional type theory. In this paper we propose two ways how choice can be added to DHOL. We extend the DHOL term structure by Hilbert's indefinite choice operator ϵ, define a translation of the choice terms to HOL choice that extends the existing translation and show that the extension of the translation is complete and give an argument for soundness. We finally evaluate the extended translation on a set of dependent HOL problems that require choice.

1 Introduction

Dependently-typed higher-order logic (DHOL), introduced by Rothgang et al. [9], is an extension of classical higher-order logic as originally presented by Church [5], albeit with a few modifications. DHOL turns the simply-typed lambda calculus with a base type for Booleans and an equality predicate for every type into a dependently-typed and extensional type theory. This makes it possible to define, for example, fixed-length lists or the type of finite sets, by replacing the simple function type $A \to B$ with its dependent version $\Pi x : A.B$. While the extensionality comes at the cost of undecidable type-checking, initial experiments by Niederhauser et al. [7] suggest that on the other side of this trade-off lies the ability to find proofs of otherwise unobtainable problems.

One of the changes to Church's original formulation is the omission of the choice operator. Choice is an important component of several higher-order interactive theorem provers such as Isabelle/HOL [8] or HOL Light [6] and therefore interesting for automated reasoning — for example illustrated by the successes of Satallax [1, 3]. This suggests that DHOL-capable theorem provers like Lash [4, 7] will eventually too benefit from the availability of choice.

Contributions: We provide the necessary groundwork for such future work by extending DHOL with an indefinite choice operator ε and the corresponding additions to the translation. We match the completeness results for the translation as given by Rothgang et al. [9] and give arguments for the soundness. We implement the erasure(s) in an extended version of Lash, introduce 34 DHOL problems with choice, and evaluate type-checking and proving on these problems. The remainder of the paper is structured as follows: Section 2 sets the stage for the extension, introducing HOL and DHOL as well as the erasure between them. Following that, in Section 3, we present the extension to the language and translation and subsequently prove it to be complete. We introduce some DHOL problems and evaluate the translation in Section 4.

2 Preliminaries

First, we present a formulation of higher-order logic (HOL), closely following the formulation given in [9] and [7] to facilitate a simple extension to DHOL in the next subsection. Figure 1

$$
\begin{array}{llll}
T, U & ::= & \circ \mid T, a : tp \mid T, x : A \mid T, t & \text{theories} \\
\Gamma, \Delta & ::= & \cdot \mid \Gamma, x : A \mid \Gamma, s & \text{context} \\
A, B & ::= & a \mid A \to B \mid o & \text{types} \\
t, u, v & ::= & x \mid \lambda x : A.t \mid tu \mid \bot \mid t \Rightarrow u \mid t =_A u \mid \forall x : A.t & \text{theories}
\end{array}
$$

Figure 1: HOL Syntax

gives the grammar of the HOL syntax.

A theory T is a concatenation of the empty theory $\circ$ and any number of (simple) base type declarations $a : tp$, typed variable and/or constant declarations $x : A$, and axioms. A context Γ similarly concatenates the empty context $\cdot$ with typed variables, typed constant declarations, and axioms, but misses base type declarations.

Aside from base types, there is no distinction in how theories and contexts are handled. The difference is mainly semantic: Theories *declare* the constants, types, and axioms that hold while contexts *assume* that there are some variables of a certain type, or that some proposition is true. As such, during proof search — for example in [7] — the theory is static, while there might be changes to the context.

Types are either the built-in boolean base type o, base types a as defined in the theory or function types $A \to B$. We will use upper-case letters $A, B, ...$ to denote type variables. Lastly, terms are variables/constants x, λ-abstractions $\lambda x : A.t$, applications tu or the built-in boolean constant $\bot$ denoting the false proposition extended with the boolean connectives $t \Rightarrow u$, $t =_A u$ and $\forall x : A.t$. We will use the lowercase letters $t, u, v, ...$ as term variables. We write $\neg t$ for $t \Rightarrow \bot$, $t \neq_A u$ for $\neg(t =_A u)$, $\top$ for $\neg\bot$ as well as $t \wedge u$ and $t \vee u$ for $\neg(t \Rightarrow \neg u)$ and $\neg t \Rightarrow u$ respectively. Lastly, $\exists x : A.t$ will stand for $\neg\forall x : A.\neg t$.

Application associates to the left i.e. $(tu)v \mathrel{\hat{=}} tuv$, and we drop the subscript in $t =_A u$ if it is clear from the context or irrelevant. We write $t[x_1/u_1, ..., x_n/u_n]$ for the simultaneous capture-avoiding substitution of variables x_i with terms t_i.

The system uses the following judgments: We write $\vdash^{\mathsf{s}} T \; thy$ and $\vdash^{\mathsf{s}}_T \Gamma \; ctx$ for well-formed theories and contexts respectively. The s in the superscript makes clear we are talking about simple types. $\Gamma \vdash^{\mathsf{s}}_T A \; tp$ and $\Gamma \vdash^{\mathsf{s}}_T t : A$ establish that A is a well-formed type and the well-formed term t has type a. A special case of this judgment is $\Gamma \vdash^{\mathsf{s}}_T u$. In this case, the well-formed term u is of type o and also provable from T and Γ. Lastly, we write $\Gamma \vdash^{\mathsf{s}}_T A \equiv B$ to say that types A and B are judgmentally equal. This is a trivial statement in HOL but will become much harder once we introduce dependent types in the next section. Note that the T in the subscript of the turnstile is only ever absent when the statement talks about the well-formedness of the theory. As such we will drop it in the remainder, as it is fairly clear from the context whether or not it is technically required.

2.1　DHOL

We now make two minute changes to the previously defined syntax to allow for dependent types: First, we replace the function type $A \to B$ with its dependent version $\Pi x : A.B(x)$. Now the second type B may depend on a term of type A. In the case where x does not appear free in B, we will stick to the arrow notation. The second change introduces dependent base types. A user-defined base type can now take any number of arguments of other — already declared — types. This base type can then be instantiated into a concrete type with terms of the fitting types. See Figure 2 for the exact changes to the grammar.

We call base types with an arity of 0 *simple base types* in reference to non-dependently typed

$$
\begin{array}{llll}
T, U & ::= & \circ \mid T, a : (\Pi x : A.)^* tp \mid T, x : A \mid T, t & \text{theories} \\
\Gamma, \Delta & ::= & \cdot \mid \Gamma, x : A \mid \Gamma, s & \text{context} \\
A, B & ::= & at_1...t_n \mid \Pi x : A.B \mid o & \text{types} \\
t, u, v & ::= & x \mid \lambda x : A.t \mid tu \mid \bot \mid t \Rightarrow u \mid t =_A u \mid \forall x : A.t & \text{theories}
\end{array}
$$

Figure 2: DHOL Syntax (changes are highlighted)

HOL, as the fragment of DHOL where all base types have arity 0 is just HOL. The judgments stay the same, but we now write $\vdash^{\mathsf{d}}$ to differentiate between HOL and DHOL. It is easy to see that it is now required for theories and contexts to be ordered, as the well-typedness of terms down the line may depend on the assumptions and assertions given previously.

More importantly, type-equality judgments $\Gamma \vdash^{\mathsf{d}} A \equiv B$ are now significantly harder. Consider a type $A := at_1...t_n$ and a type $A' := at'_1...t'_n$. To establish that $\Gamma \vdash^{\mathsf{d}} A \equiv A'$ it is necessary to show that for all i s.t. $1 \leqslant i \leqslant n$ $\Gamma \vdash^{\mathsf{d}} t_i =_{A_i[x_1/t_1,...,x_{i-1}/t_{i-1}]} t'_i$ is provable, i.e. DHOL is an extensional type theory and as such type checking is undecidable.

2.1.1 Erasure

As DHOL is rather new, there is not much native ATP support for the logic yet. However, the increased expressivity in itself is already valuable. Rothgang et al. introduced a sound and complete translation from DHOL to HOL, thereby providing automation support by serving as an interface between the DHOL language and any higher-order capable theorem prover. We give an overview of the translation and refer to the original paper [9] for the details and proofs.

The basic idea of the translation $\overline{\square}$ is to reduce dependent types to their simple components and to encode the lost information in a partial equivalence relation (PER). The distinguishing feature of a PER is that it does not require reflexivity. Elements of a DHOL-type $x : A$ are just those elements for which a PER A^* is indeed reflexive, i.e. $\Gamma \vdash^{\mathsf{d}} t : A$ holds if and only if $\overline{\Gamma} \vdash^{\mathsf{s}} A^* \, \bar{t} \, \bar{t}$ and $\overline{\Gamma} \vdash^{\mathsf{s}} \bar{t} : \overline{A}$.

Formally, the translation from DHOL to HOL is defined inductively on the grammar:

$$
\begin{array}{rcl}
\overline{\circ} & = & \circ \\
\overline{T, U} & = & \overline{T}, \overline{U} \\
\overline{o} & = & o \\
\overline{\Pi x : A.B} & = & \overline{A} \to \overline{B} \\
\overline{\lambda x : A.t} & = & \lambda x : \overline{A}. \, \bar{t} \\
\overline{\bot} & = & \bot \\
\overline{t \Rightarrow u} & = & \bar{t} \Rightarrow \overline{u} \\
\overline{\forall x : A.t} & = & \forall x : \overline{A}. \, A^* \, x \, x \Rightarrow \bar{t}
\end{array}
\qquad
\begin{array}{rcl}
\overline{\cdot} & = & \cdot \\
\overline{\Gamma, \Delta} & = & \overline{\Gamma}, \overline{\Delta} \\
\overline{at_1 \dots t_n} & = & a \\
\overline{x} & = & x \\
\overline{t \, u} & = & \bar{t} \, \overline{u} \\
\overline{\neg t} & = & \neg \bar{t} \\
\overline{t =_A u} & = & A^* \, \bar{t} \, \overline{u} \\
\overline{x : A} & = & x : \overline{A}, A^* \, x \, x
\end{array}
$$

$$
\begin{array}{rcl}
\overline{a : \Pi x_1 : A_1. \, \cdots \, \Pi x_n : A_n. \, \mathsf{tp}} & = & a : \mathsf{tp} \\
& & a^* : \overline{A_1} \to \cdots \to \overline{A_n} \to a \to a \to o \\
& & \text{with } \forall x_1 : \overline{A_1}. \, \dots \forall x_n : \overline{A_n}. \, \forall u, v : a. \, a^* \, x_1 \, \dots \, x_n \, u \, v \Rightarrow u =_a v \\
o^* \, t \, u & = & t =_o u \\
(at_1 \dots t_n)^* \, u \, v & = & a^* \, \overline{t_1} \, \dots \, \overline{t_n} \, u \, v \\
(\Pi x : A.B)^* \, t \, u & = & \forall x, y : \overline{A}. \, A^* \, x \, y \Rightarrow B^* \, (t \, x) \, (u \, y)
\end{array}
$$

3   Choice

In order to allow choice in the input problems as well as in the inference rules, we extend the HOL and DHOL term languages with $(\varepsilon x : A.t)$. In HOL, the semantics of choice are more straightforward, since all types need to be non-empty. This is not the case in DHOL, which is why we propose two variants of typing rules for choice terms: the stronger ε_1 rule that requires the existence of an element satisfying the predicate and the weaker ε_2 rule that only relies on the type being inhabited. They are presented in Figures 3 and 4 respectively. These are in addition to the DHOL rules ([9]).

The HOL rules are extended with choice in a straightforward way since simple types are assumed nonempty. See Figure 5. Note the difference in formulation: $\mathrm{DHOL}_{\varepsilon 1}$ has a stricter type formation rule — requiring the ATP system to exhibit the existence of an element x such that t is true — while $\mathrm{DHOL}_{\varepsilon 2}$ only requires that the dependent type A is non-empty. The corresponding erasures respect this definition in that the stronger erasure assumes the provability of $\exists x : A.t$ while the second and weaker erasure allows for a case distinction. The actual proof rule in either case requires the existence of a valid element of course.

Now that we can form well-typed terms in DHOL including choice and have specified what is provable, we can we extend the erasure [9] to achieve automated proof support. A first attempt might be to erase $(\varepsilon x : A.t)$ to the simply typed term $(\varepsilon x : \overline{A}.\overline{t})$. This erasure would make the translation "incomplete." In particular, we could have $\Gamma \vdash^{\mathsf{d}}_{T} (\varepsilon x : A.t) : A$ but not have $\overline{\Gamma} \vdash^{\mathsf{s}}_{\overline{T}} A^* (\varepsilon x : \overline{A}.\overline{t}) (\varepsilon x : \overline{A}.\overline{t})$. As a simple example, consider the DHOL theory T given by $a : \Pi x : o.tp, c : a \perp$ and the DHOL term $(\varepsilon x : a \perp.\top)$. Clearly we have $\cdot \vdash^{\mathsf{d}}_{T} (\varepsilon x : a \perp.\top) : a \perp$ in both the strong and (hence) weak senses. However, $\cdot \nvdash^{\mathsf{s}}_{\overline{T}} a^* \perp (\varepsilon x : a.\top) (\varepsilon x : a.\top)$.

Consequently, we must somehow include information about A^* in the erasure of $\varepsilon x : A.t$ (just as is done when erasing quantifiers). In the case of the ε-operator, we need a different erasure depending on whether we use the strong typing rule or the weak typing rule. We will continue to use $\bar{t}$ for the strong erasure and introduce the variant $\hat{t}$ for the weak erasure. Likewise, we will continue to use the notation A^* for the strong erasure of a type to a PER and introduce

$$\frac{\Gamma, x : A \vdash^{\mathsf{d}\varepsilon_1} t : o \quad \Gamma, \forall x : A.\neg t \vdash^{\mathsf{d}\varepsilon_1} \perp}{\Gamma \vdash^{\mathsf{d}\varepsilon_1} (\varepsilon x : A.t) : A} \varepsilon_1 \mathrm{type} \qquad \frac{\Gamma, x : A \vdash^{\mathsf{d}\varepsilon_1} t : o \quad \Gamma, \forall x : A.\neg t \vdash^{\mathsf{d}\varepsilon_1} \perp}{\Gamma \vdash^{\mathsf{d}\varepsilon_1} t[x/(\varepsilon x : A.t)]} \varepsilon_1$$

Figure 3: Choice Rules for $\mathrm{DHOL}_{\varepsilon 1}$

$$\frac{\Gamma, x : A \vdash^{\mathsf{d}\varepsilon_2} t : o \quad \Gamma, \forall x : A.\perp \vdash^{\mathsf{d}\varepsilon_2} \perp}{\Gamma \vdash^{\mathsf{d}\varepsilon_2} (\varepsilon x : A.t) : A} \varepsilon_2 \mathrm{type} \qquad \frac{\Gamma, x : A \vdash^{\mathsf{d}\varepsilon_2} t : o \quad \Gamma, \forall x : A.\neg t \vdash^{\mathsf{d}\varepsilon_2} \perp}{\Gamma \vdash^{\mathsf{d}\varepsilon_2} t[x/(\varepsilon x : A.t)]} \varepsilon_2$$

Figure 4: Choice Rules for $\mathrm{DHOL}_{\varepsilon 2}$

$$\frac{\Gamma, x : A \vdash^{\mathsf{s}\varepsilon} t : o}{\Gamma \vdash^{\mathsf{s}\varepsilon} (\varepsilon x : A.t) : A} \mathrm{choice\ type} \qquad \frac{\Gamma, x : A \vdash^{\mathsf{s}\varepsilon} t : o \quad \Gamma, \forall x : A.\neg t \vdash^{\mathsf{s}\varepsilon} \perp}{\Gamma \vdash^{\mathsf{s}\varepsilon} t[x/(\varepsilon x : A.t)]} \mathrm{choice}$$

Figure 5: Choice Rules for $\mathrm{HOL}_{\varepsilon}$

the notation $A^{\hat{*}}$ for the weak erasure of a type to a PER. The two erasures only differ in their treatment of ε. The A^* and $A^{\hat{*}}$ differ only in the sense of which erasure is used on the terms of a dependent type. That is, $(a\ t_1 \cdots t_n)^*$ is $a^*\ \overline{t_1} \cdots \overline{t_n}$ while $(a\ t_1 \cdots t_n)^{\hat{*}}$ is $a^*\ \hat{t_1} \cdots \hat{t_n}$. All that remains is to specify the values of $\overline{\varepsilon x : A.t}$ and $\widehat{\varepsilon x : A.t}$. For the strong erasure we simply take $\overline{\varepsilon x : A.t} = \varepsilon x : \overline{A}.A^*\ x\ x \wedge \overline{t}$. For the weak erasure we need to consider two cases: when there is a witness satisfying the predicate and when there is not. If there is a witness, we will use the simply typed ε just as in the strong erasure. However, when there is no witness, we will still need to choose something of type $\overline{A}$ that will be related to itself by A^*. If we had an if-then-else constructor in HOL, we could write $\widehat{\varepsilon x : A.t}$ as

$$\text{if } (\exists x : \hat{A}.A^{\hat{*}}\ x\ x \wedge \hat{t}) \text{ then } (\varepsilon x : \hat{A}.A^{\hat{*}}\ x\ x\ \wedge\ \hat{t}) \text{ else } (\varepsilon x : \hat{A}.A^{\hat{*}}\ x\ x).$$

As is well-known, if-then-else can be defined using ε as $\lambda pxy.\varepsilon z.p \wedge z = x \vee \neg p \wedge z = y$. We remain in the HOL language we have already introduced, and define $\widehat{\varepsilon x : A.t}$ to be

$$\varepsilon z : \hat{A}. \quad (\exists x : \hat{A}.A^{\hat{*}}\ x\ x \wedge \hat{t}) \wedge z = (\varepsilon x : \hat{A}.A^{\hat{*}}\ x\ x\ \wedge\ \hat{t})$$
$$\vee \neg(\exists x : \hat{A}.A^{\hat{*}}\ x\ x \wedge \hat{t}) \wedge z = (\varepsilon x : \hat{A}.A^{\hat{*}}\ x\ x).$$

3.1   Completeness

We will use $\widetilde{\square}$ to denote either weak or strong erasure, corresponding to the typing rule used. Using induction on the structure of the natural deduction rules, we extend the completeness proof given in [9] to get the following theorem:

Theorem. *For either variant — $DHOL_{\varepsilon 1}$ or $DHOL_{\varepsilon 2}$ — we retain that if $\Gamma \vdash^{\mathsf{d}} t : A$ then $\widetilde{\Gamma} \vdash^{\mathsf{s}} \widetilde{t} : \widetilde{A}$ and $\widetilde{\Gamma} \vdash^{\mathsf{s}} A^*\ \widetilde{t}\ \widetilde{t}$. Also, if $\Gamma \vdash^{\mathsf{d}} t$ then $\widetilde{\Gamma} \vdash^{\mathsf{s}} \widetilde{t}$.*

3.1.1   Proof of completeness for ε_1 and strong erasure

From the first assumption $\Gamma, x : A \vdash^{\mathsf{d}} t : o$ we get the induction hypothesis $\overline{\Gamma}, x : \overline{A}, A^*\ x\ x \vdash^{\mathsf{s}} \overline{t} : o$. With the $\mathtt{assume}$ rule, we can establish $\overline{\Gamma}, x : \overline{A}, A^*\ x\ x \vdash^{\mathsf{s}} A^*\ x\ x$. Due to well-typedness, we know this is of type o and as HOL types cannot depend on propositions in the context, we may drop the $A^*\ x\ x$ from the context in both hypotheses. Using $\wedge$-type on the hypotheses and a simple application of the definition of the strong erasure yields the first goal $\overline{\Gamma} \vdash^{\mathsf{s}} \overline{(\varepsilon x : A.t)} : \overline{A}$. Note, that the first assumption of all $\mathtt{choice\ type}$ rules is the same, so this reasoning holds for either case.

Next, we consider the second assumption $\Gamma, \forall x : A.\neg t \vdash^{\mathsf{d}} \bot$ with the corresponding induction hypothesis $\overline{\Gamma}, \forall x : \overline{A}.A^*\ x\ x \Rightarrow \neg \overline{t} \vdash^{\mathsf{s}} \bot$. By $\Rightarrow_i$ and $\neg\neg_i$ we arrive at $\overline{\Gamma} \vdash^{\mathsf{s}} \neg\forall x : \overline{A}.\neg\neg(A^*\ x\ x \Rightarrow \neg \overline{t})$ which is equivalent to $\overline{\Gamma} \vdash^{\mathsf{s}} (\forall x : \overline{A}.\neg(A^*\ x\ x \wedge \overline{t})) \Rightarrow \bot$. We now put it back into the context and use the $\mathtt{simple\ choice}$ rule to conclude $\overline{\Gamma} \vdash^{\mathsf{s}} (A^*\ x\ x \wedge \overline{t})[x/(\varepsilon x : \overline{A}.A^*\ x\ x \wedge \overline{t})]$. The definition of the strong erasure applied to the term in the substitution and subsequent substituting of the x now yields our second goal and gives us the third one for free: $\overline{\Gamma} \vdash^{\mathsf{s}} A^*\ \overline{(\varepsilon x : A.t)}\ \overline{(\varepsilon x : A.t)} \wedge \overline{t}[x/\ \overline{(\varepsilon x : A.t)}]$. The derivation of $\overline{\Gamma} \vdash^{\mathsf{s}} \overline{t}[x/(\varepsilon x : A.t)]$ is valid because the $\mathtt{choice}$ rule in the ε_1 case uses the same premises as the $\mathtt{choice\ type}$ rule.

3.1.2   Proof of completeness for ε_2 and weak erasure

We now show the same results for the weak typing variant. Since $\widehat{(\varepsilon x : A.t)}$ is defined using choice to implement if-then-else, it is easy to prove that $\Gamma \vdash^{\mathsf{s}} \widehat{(\varepsilon x : A.t)} = (\varepsilon x : \hat{A}.A^{\hat{*}}\ x\ x\ \wedge\ \hat{t})$ if

$\Gamma \vdash^{\mathsf{s}} \exists x : \hat{A}.A^{\hat{*}} \ x \ x \wedge \hat{t}$. Likewise, if $\Gamma \vdash^{\mathsf{s}} \neg\exists x : \hat{A}.A^{\hat{*}} \ x \ x \wedge \hat{t}$, then $\Gamma \vdash^{\mathsf{s}} \widehat{(\varepsilon x : A.t)} = (\varepsilon x : \hat{A}.A^{\hat{*}} \ x \ x)$. We will use both facts implicitly below to rewrite $\widehat{(\varepsilon x : A.t)}$ according to one of these equations.

For both the ε_2-$\texttt{type}$ rule and the ε_2 rule the inductive hypothesis of the first premise allows us to infer $\hat{\Gamma} \vdash^{\mathsf{s}} \widehat{(\varepsilon x : A.t)} : \hat{A}$ as in the corresponding proof for the strong erasure.

We first show completeness for the ε_2 rule. The inductive hypothesis for the second premise gives $\hat{\Gamma}, \forall x : \hat{A}.A^{\hat{*}} \ x \ x \Rightarrow \neg\hat{t} \vdash^{\mathsf{s}} \bot$. From this we can derive $\hat{\Gamma} \vdash^{\mathsf{s}} \exists x : \hat{A}.A^{\hat{*}} \ x \ x \wedge \hat{t}$ and $\hat{\Gamma}, \forall x : \hat{A}.\neg(A^{\hat{*}} \ x \ x \wedge \hat{t}) \vdash^{\mathsf{s}} \bot$. We need to prove $\hat{\Gamma} \vdash^{\mathsf{s}} \hat{t}[x/\widehat{(\varepsilon x : A.t)}]$. It suffices to prove $\hat{\Gamma} \vdash^{\mathsf{s}} (A^{\hat{*}} \ x \ x \wedge \hat{t})[x/(\varepsilon x : \hat{A}.A^{\hat{*}} \ x \ x \wedge \hat{t})]$ which follows from the $\texttt{simple choice}$ rule.

We now show completeness of the ε_2-$\texttt{type}$ rule. In this case the induction hypothesis of the second premise yields $\hat{\Gamma}, \forall x : \hat{A}.A^{\hat{*}} \ x \ x \Rightarrow \bot \vdash^{\mathsf{s}} \bot$. We need to prove $\hat{\Gamma} \vdash^{\mathsf{s}} A^{\hat{*}} \ \widehat{(\varepsilon x : A.t)} \ \widehat{(\varepsilon x : A.t)}$. Unlike the previous argument, the inductive hypothesis is not sufficient to know the condition of the if-then-else is provable. From here we proceed with a case distinction along the condition. Since we are in a classical setting, proving $\hat{\Gamma}, \exists x : \hat{A}.A^{\hat{*}} \ x \ x \wedge \hat{t} \vdash^{\mathsf{s}} A^{\hat{*}} \ \widehat{(\varepsilon x : A.t)} \ \widehat{(\varepsilon x : A.t)}$ and $\hat{\Gamma}, \neg\exists x : \hat{A}.A^{\hat{*}} \ x \ x \wedge \hat{t} \vdash^{\mathsf{s}} A^{\hat{*}} \ \widehat{(\varepsilon x : A.t)} \ \widehat{(\varepsilon x : A.t)}$ suffice to prove $\hat{\Gamma} \vdash^{\mathsf{s}} A^{\hat{*}} \ \widehat{(\varepsilon x : A.t)} \ \widehat{(\varepsilon x : A.t)}$.

We begin with the first case in which we assume $\exists x : \hat{A}.A^{\hat{*}} \ x \ x \wedge \hat{t}$ in the context. Let Δ be $\hat{\Gamma}, \exists x : \hat{A}.A^{\hat{*}} \ x \ x \wedge \hat{t}$. Since $\Delta \vdash^{\mathsf{s}} \exists x : \hat{A}.A^{\hat{*}} \ x \ x \wedge \hat{t}$, it suffices to prove

$$\Delta \vdash^{\mathsf{s}} A^{\hat{*}} \ (\varepsilon x : \hat{A}.A^{\hat{*}} \ x \ x \ \wedge \ \hat{t}) \ (\varepsilon x : \hat{A}.A^{\hat{*}} \ x \ x \ \wedge \ \hat{t}).$$

By the $\texttt{simple choice}$ rule, it suffices to prove $\Delta, \forall x : \hat{A}.\neg(A^{\hat{*}} \ x \ x \wedge \hat{t}) \vdash \bot$. This is obvious since $\exists x : \hat{A}.A^{\hat{*}} \ x \ x \wedge \hat{t}$ is literally the same as $\neg\forall x : \hat{A}.\neg(A^{\hat{*}} \ x \ x \wedge \hat{t})$.

Finally, we consider the second case in which the negation of the condition. Let Δ be $\hat{\Gamma}, \neg\exists x : \hat{A}.A^{\hat{*}} \ x \ x \wedge \hat{t}$. Since $\Delta \vdash^{\mathsf{s}} \neg\exists x : \hat{A}.A^{\hat{*}} \ x \ x \wedge \hat{t}$, it suffices to prove

$$\Delta \vdash^{\mathsf{s}} A^{\hat{*}} \ (\varepsilon x : \hat{A}.A^{\hat{*}} \ x \ x) \ (\varepsilon x : \hat{A}.A^{\hat{*}} \ x \ x).$$

By the $\texttt{simple choice}$ rule, it suffices to prove $\Delta, \forall x : \hat{A}.\neg(A^{\hat{*}} \ x \ x) \vdash^{\mathsf{s}} \bot$. This is precisely the inductive hypothesis of the second premise.

3.1.3   Soundness

Theorem 2 of [9] gives a corresponding soundness result for the translation from DHOL to HOL. In particular, they prove $\Gamma \vdash^{\mathsf{d}}_T F$ holds whenever $\Gamma \vdash^{\mathsf{d}}_T F : o$ and $\overline{\Gamma} \vdash^{\mathsf{s}}_{\overline{T}} \overline{F}$. The proof of this result (found in the appendix of [9]) is significantly more involved than the completeness result. One complication is that the HOL proof of $\overline{\Gamma} \vdash^{\mathsf{s}}_{\overline{T}} \overline{F}$ may make use of terms t' that are not of the form $\overline{t}$ for a well-typed DHOL term t. (In [9] these are called "improper terms.") Semantics would provide an alternative strategy for proving soundness. One could argue that if $\Gamma \not\vdash^{\mathsf{d}}_T F$, then there is a DHOL model of T satisfying Γ but not F (via a hypothetical completeness result for DHOL models). Then one could argue that a DHOL model yields a Henkin model (for HOL) of $\overline{T}$ satisfying $\overline{\Gamma}$ but not $\overline{F}$. Such a Henkin model would violate soundness of HOL relative to Henkin models since we are assuming $\overline{\Gamma} \vdash_{\overline{T}} \overline{F}$. At the moment there is no accepted notion of DHOL model with soundness and completeness results (even without choice), and no relationship between DHOL models and Henkin models of HOL. Consequently, for the present paper we leave soundness (of both the weak and strong forms of DHOL with choice) as a conjecture and leave the sketch of a semantic argument as some indication why the soundness results should hold.

4   Experimental Problems and Results

We propose several problems in DHOL that include choice and experiment with the performance of the erasure on these problems. For the experiments, a modified version of Lash [4, 7] was run on the listed problems, once with each kind of erasure, for 90s.

The first set of problems share definitions for the type of natural numbers nat with the corresponding constructors $0 : \mathsf{nat}$ and $\mathsf{s} : \mathsf{nat} \to \mathsf{nat}$, and the dependent-type of fixed-size finite sets $\Pi n : \mathsf{nat}.tp$ with the two constructors $\mathsf{fz} : \Pi n : \mathsf{nat}.\mathsf{fin}(\mathsf{s}\ n)$ and $\mathsf{fs} : \Pi n : \mathsf{nat}.\mathsf{fin}\ n \to \mathsf{fin}(\mathsf{s}\ n)$. We use numbers $1, 2, ...$ to mean $\mathsf{s}\ 0, \mathsf{s}\ (\mathsf{s}\ 0), ...$ Following is a short description of the different classes of problems:

- `choice_def1` has in addition a type for predicates p for elements of $\mathsf{fin}\ 2$ and an axiom that establishes the existence of an element $x : \mathsf{fin}\ 2$ which fulfills the predicate. The conjecture expresses the definition of the choice operator in this setting: $p(\varepsilon x : \mathsf{fin}\ 2.p\ x)$.
- `choice_def2` is a generalization of `choice_def` from $\mathsf{fin}\ 2$ to $\mathsf{fin}\ n$ where $n \in \mathbb{N}$.
- `choice_def3` finally generalizes the previous example to an arbitrary dependent type.
- `choice_eq1/2` establish that the ε-operator respects identity for the types $\mathsf{fin}\ 1$, where there is only one element to choose from, and $\mathsf{fin}\ 2$ respectively.
- `choice_nq` in turn, demonstrates that choice correctly chooses the other element in the $\mathsf{fin}\ 2$ type, or alternatively, that there are at least two elements in $\mathsf{fin}\ 2$.
- `no_fp_finN_reg` and `no_fp_finN_min` is a family of problems for $N \in \{0, ..., 9\}$. Each has a conjecture that says that the function $\lambda x : (\mathsf{fin}\ N).(\varepsilon y : (\mathsf{fin}\ N).x \neq y))$ has no fixed point. The variants with `_reg` have $\mathsf{fin}\ N$ behave like previously, while the `_min` versions have fewer axioms and result in types that are assumed to have at least N elements as opposed to exactly N elements. In case $N \geqslant 2$, it is clear that the term $\lambda x : (\mathsf{fin}\ N).(\varepsilon y : (\mathsf{fin}\ N).x \neq y))$ is well typed (of type $\mathsf{fin}\ N \to \mathsf{fin}\ N$) in both the strong and weak senses. Likewise it is provable that the function given by the term has no fixed points. When $N = 1$, the term $\lambda x : (\mathsf{fin}\ 1).(\varepsilon y : (\mathsf{fin}\ 1).x \neq y))$ is well-typed in the weak sense (since $\mathsf{fin}\ 1$ is provably nonempty) but is not well-typed in the strong sense (since $\mathsf{fin}\ 1$ does not provably have at least two elements). When $N = 0$, the situation depends on whether we have assumed $\mathsf{fin}\ 0$ has precisely 0 elements (a *provably* empty type) or at least 0 elements (a *possibly* empty type). If $\mathsf{fin}\ 0$ is provably empty, then the term $\lambda x : (\mathsf{fin}\ 0).(\varepsilon y : (\mathsf{fin}\ 0).x \neq y))$ is well-typed in both the strong and weak sense, simply because $\Gamma, x : \mathsf{fin}\ 0 \vdash^{\mathsf{d}} \bot$ allows us to prove the premises of the relevant rules. Likewise, if $\mathsf{fin}\ 0$ is provably empty, it is provable that the function has no fixed point since quantification over an empty type is vacuous. If $\mathsf{fin}\ 0$ is only possibly empty, then $\lambda x : (\mathsf{fin}\ 0).(\varepsilon y : (\mathsf{fin}\ 0).x \neq y))$ is not well-typed in the strong sense but is well-typed in the weak sense. The reason it is well-typed in the weak sense is that, although we cannot prove $\mathsf{fin}\ 0$ is nonempty, we do have $\Gamma, x : \mathsf{fin}\ 0, \forall x : \mathsf{fin}\ 0.\bot \vdash^{\mathsf{d}} \bot$.

Additionally, several examples of problems related to fixed-length lists have been considered. These extend the definitions of natural numbers and successor function by definitions for the fixed length list, which has the same type as fin, $\mathsf{nil} : \mathsf{list}\ 0$ and the constructor $\mathsf{cons} : \Pi n : \mathsf{nat}.\mathsf{nat} \to \mathsf{list}\ n \to \mathsf{list}\ (\mathsf{s}\ n)$.

- `list_empty` additionally introduces the predicate `empty` that takes lists of any length and two axioms that ensure that the predicate is only satisfied when the list is indeed of length 0. The conjecture then establishes that choosing an empty list with ε satisfies the predicate.
- `list_nonempty` includes the same additional definitions but asserts that choosing a list of length 1 does not satisfy the empty predicate.

	ε_1 type-check	ε_1 prove	ε_2 type-check	ε_2 prove
choice_def1..def3	✓	✓	✓	✗
no_fp_fin{0,2..9}_reg	✓	✓	✓	✓
no_fp_fin{0,2..8}_min	✓	✓	✓	✓
no_fp_fin9_min	✓	✓	✓	✗
choice_eq1	✓	✓	✓	✗
choice_eq2	✓	✗	✓	✗
choice_nq	✗	✗	✗	✗
list_empty	✓	✓	✓	✗
list_nonempty	✗	✗	✗	✗
list_head	✓	✗	✓	✗

Table 1: Summary of the experimental results for the two typing/erasure variants.

- list_head introduces hd : Πn : nat.list (s n) → nat — the unfailing head function. The conjecture asserts that the first element of a list chosen such that the chosen list's first element is 0, is indeed 0.

4.1   Results

The results are presented in Table 1. Lash can easily type-check both versions of all choice_def problems. Proving the conjectures, however, only works well for the strictly typed version — requiring less than 10 seconds in all cases — but times out when using the weakly typed version. Similarly, choice_eq1 is type-checkable and provable with strong typing only. choice_eq2 still type-checks for both versions but is now provable by neither. Lash is not able to type-check or prove choice_nq for either of the choice typing variants. The dependent list experiments are the hardest: Lash can only prove the simplest of the three conjectures.

Lash does not type check the ill-typed problems among no_fp_fin* (included as a sanity test). Lash also does not report them as provable. With a 90s timeout Lash can type check and prove each of the problems for $n \geq 2$ with one exception: it does not prove no_fp_fin9_min.

We generated erased versions of the no_fp_fin9_min problem, and ran them on several other higher-order ATP systems. Only Leo-III [10] and Zipperposition [2] were supporting native choice in the TPTP THF language and succeeded. Leo-III was able to prove the conjecture in 38s using the strong erasure but timed out using the weak version. Zipperposition took 0.02s for the strong erasure, and 0.96s for the weak one, suggesting that the challenges posed by the weak typing are not Lash-specific.

Overall, surprisingly, type-checking performs comparably well for weak and strong choice. However, using the strong typing rules results in more than 50% more problems proved.

5   Conclusion

We have extended DHOL by dependent choice. Since types in DHOL can be empty, we proposed two ways to specify the rules: a strong typing rule that ensures that there exist witnesses and a weak choice typing rule that only checks that the underlying type is non-empty. These are accompanied by a proof rule and two erasure functions to enable automated reasoning. We also created a collection of 34 dependent HOL problems that use/require dependent choice (http://cl-informatik.uibk.ac.at/cek/choice.tgz). Our proofs and experiments show that stronger choice works well with DHOL. Future work includes implementation of native choice rules.

References

[1] Julian Backes and Chad E. Brown. Analytic tableaux for higher-order logic with choice. *Journal of Automated Reasoning*, 47:451–479, 2011.

[2] Alexander Bentkamp, Jasmin Blanchette, Sophie Tourret, and Petar Vukmirović. Superposition for higher-order logic. *Journal of Automated Reasoning*, 67:10, 2023.

[3] Chad E. Brown. Satallax: An automatic higher-order prover. In Bernhard Gramlich, Dale Miller, and Uli Sattler, editors, *Proc. 6th International Joint Conference on Automated Reasoning*, volume 7364 of *LNAI*, pages 111–117, 2012.

[4] Chad E. Brown and Cezary Kaliszyk. Lash 1.0 (system description). In Jasmin Blanchette, Laura Kovács, and Dirk Pattinson, editors, *Proc. 11th International Joint Conference on Automated Reasoning*, volume 13385 of *LNAI*, pages 350–358, 2022.

[5] Alonzo Church. A formulation of the simple theory of types. *Journal of Symbolic Logic*, 5(2):56–68, 1940.

[6] John Harrison. HOL Light: An overview. In Stefan Berghofer, Tobias Nipkow, Christian Urban, and Makarius Wenzel, editors, *Proc. 22nd International Conference on Theorem Proving in Higher Order Logics*, volume 5674 of *LNCS*, pages 60–66, 2009.

[7] Johannes Niederhauser, Chad E. Brown, and Cezary Kaliszyk. Tableaux for automated reasoning in dependently-typed higher-order logic. Submitted, http://cl-informatik.uibk.ac.at/cek/submitted/jncbck.pdf.

[8] Tobias Nipkow, Lawrence C. Paulson, and Markus Wenzel. *Isabelle/HOL — A Proof Assistant for Higher-Order Logic*, volume 2283 of *LNCS*. Springer Berlin Heidelberg, 2002.

[9] Colin Rothgang, Florian Rabe, and Christoph Benzmüller. Theorem proving in dependently-typed higher-order logic. In Brigitte Pientka and Cesare Tinelli, editors, *Proc. 29th International Conference on Automated Deduction*, volume 14132 of *LNAI*, pages 438–455, 2023.

[10] Alexander Steen and Christoph Benzmüller. Extensional higher-order paramodulation in Leo-III. *Journal of Automated Reasoning*, 65:775–807, 2021.