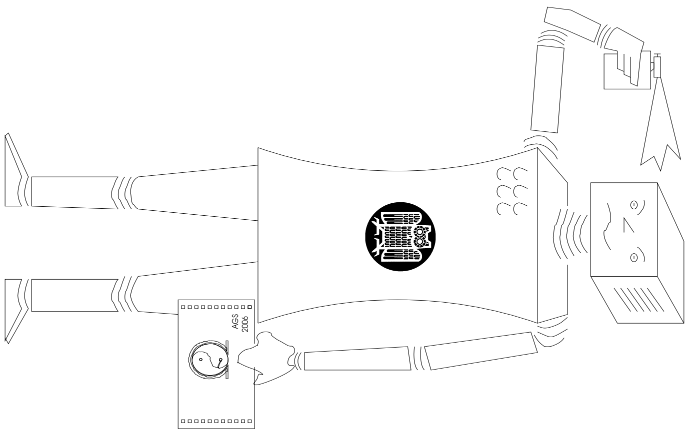
SEKI-Working-Paper ISSN 1860-5931

UNIVERSITÄT DES SAARLANDES
FACHRICHTUNG INFORMATIK
D–66123 SAARBRÜCKEN
GERMANY
WWW: http://www.ags.uni-sb.de/

Dependently Typed Set Theory

Chad E. Brown
FR Informatik, Saarland Univ.
cebrown@ags.uni-sb.de

SEKI-Working-Paper SWP–2006–03



This SEKI-Working-Paper was internally reviewed by:

Serge Autexier
FR Informatik, Universität des Saarlandes, D–66123 Saarbrücken, Germany
E-mail: autexier@ags.uni-sb.de
WWW: http://www-ags.dfki.uni-sb.de/~serge/

Editor of SEKI series:

Claus-Peter Wirth
Brandenburger Str. 42, D–65582 Diez, Germany
E-mail: wirth@logic.at
WWW: http://www.ags.uni-sb.de/~cp

Dependently Typed Set Theory

Chad E. Brown
FR Informatik, Saarland Univ.
cebrown@ags.uni-sb.de

Submitted October 30, 2006
Definitive Print Edition
December 1, 2006

Abstract

Dependently Typed Set Theory (DeTSeT) is untyped set theory encoded in a dependent type theory. The dependent type theory includes proof irrelevance and a special Σ-type between objects and proofs. This allows one to create a "class type" from any predicate on the untyped universe of sets. Given this rich type structure, one can represent partial functions as total functions on the appropriately restricted domain. Set theory can be encoded as a finite signature in the type theory. Furthermore, the signature is essentially second-order (in the λ-calculus sense). We develop the basic theory of DeTSeT and a signature for mathematics.

Keywords: Dependent Type Theory, Set Theory, Proof Irrelevance, Foundations of Mathematics

1 Introduction

In this report, we describe a Dependently Typed Set Theory (DeTSeT) as a foundation for formal mathematics. DeTSeT allows one to work with both an untyped universe of all objects and certain typed collections of these objects. Using the framework, one can mediate between untyped and typed objects. Intuitively, in DeTSeT, one has

$$Sets \subseteq Classes \subseteq Types.$$

Another important aspect of the framework is proof irrelevance. In logic, proofs are often important objects of study. In mathematics, proofs are not the objects of study, but rather a way of determining the "truth" of propositions about mathematical objects.

We give a theoretical account of DeTSeT with typing judgments and denotational semantics. We also give a syntax-directed algorithmic typing judgment.

Currently DeTSeT is implemented in the system Scunak [9, 7]. Soon it will also be implemented as the new logical foundation for the system Ωmega [32, 33].

As a foundation for formal mathematics, DeTSeT could be used to represent a formal library of mathematics as envisioned in the QED Manifesto [28]. There are many motivations for having such a library and these motivations are discussed in the QED Manifesto. Building a formal library (i.e., formalizing mathematics) is a time consuming process. Ideally, one could imagine a program which scans a mathematics book and converts this scanned version into a formal version for the library. We are far from this ideal world, but it is clear that for this to ever be possible, the formal version must correspond closely to the informal version. Attempts to force higher-order concepts into a first-order representation make clear the need for a formal language other than pure first-order logic. Likewise, attempts to treat concepts such as generic sets, partial functions, and algebraic concepts in simple type theory indicate that simple type theory is also not ideal. We introduce DeTSeT as a new form of set theory which is completely formal and yet allows one to represent informal mathematical concepts in a more natural way than previous logical systems.

1.1 What is Set Theory?

A model of set theory is a collection V along with a binary relation $\in$. The elements of V are meant to represent sets. Hence each member of V should be uniquely determined by its elements. That is, for all $X, Y \in V$, if $\forall z \in V.z \in X \equiv z \in Y$, then $X = Y$. Often one restricts to models in which the members of V actually are sets and $\in$ is the usual membership relation. For this to make sense, for every element x of every set $X \in V$, we must have $x \in X$. That is, for every $X \in V$, we have $X \subseteq V$. In other words, V is a *transitive* collection of sets.

A reasonably simple example of a set theoretic model V consists of the hereditarily finite sets. We inductively define V_n for natural numbers n by $V_0 := \emptyset$ and $V_{n+1} := \mathcal{P}(V_n)$ (the power set of V_n). For each n, V_n is transitive. (The base case is trivial. For the inductive case, note that $x \in X \in V_{n+1}$ implies $x \in V_n$ implies $x \subseteq V_n$ implies $x \in V_{n+1}$.) Hence

$$V_0 \subseteq V_1 \subseteq \cdots \subseteq V_n \subseteq \cdots$$

Finally, we let $V_\omega := \bigcup V_n$. Clearly, V_ω is transitive as well.

What other properties does V_ω satisfy? One can easily verify the following facts:

1. $\emptyset \in V_\omega$.
2. If $x, y \in V_\omega$, then $\{x, y\} \in V_\omega$.
3. If $X \in V_\omega$, then $\mathcal{P}(X) \in V_\omega$, i.e., V_ω is closed under power sets.
4. If $X \in V_\omega$, then $\bigcup X \in V_\omega$, i.e., V_ω is closed under unions of sets of sets.

These are some of the usual properties expected of a model of set theory. In particular, these correspond to some of the axioms of Zermelo set theory:

- *Extensionality:* Sets are determined by their elements.
- *Emptyset:* The empty set is a set.

- *Pairing:* If x and y are sets, then $\{x, y\}$ is a set.
- *Powerset:* If X is a set, then $\mathcal{P}(X)$ is a set.
- *Union:* If X is a set, then $\bigcup X$ is a set.
- *Separation:* If X is a set and ϕ is a *property*, then $\{x \in X | \phi(x)\}$.

The notion of *property* is imprecise in the separation axiom, as it was when Zermelo first formulated set theory [38]. The term "Zermelo set theory" has come to mean the interpretation in which $\phi(x)$ is a first-order proposition. This can be changed to obtain other set theories. For example, if one restricts $\phi(x)$ to be a first-order formula in which all quantifiers are bounded (i.e., every $\forall y$ occurs as $\forall y.y \in A \Rightarrow \cdots$ and every $\exists y$ occurs as $\exists y.y \in A \wedge \cdots$), then one obtains the separation axiom for "Mac Lane set theory".

The foundational significance of set theory is that the usual mathematical objects of interest (pairs, relations, functions, etc.) can be constructed as sets. Within the collection V_ω, one can do many of the usual constructions: the (Kurwatowski) ordered pair $\langle x, y\rangle$ can be defined as the set $\{\{x\}, \{x, y\}\}$. Binary relations are sets of pairs, and functions are functional relations. So, within the collection V_ω we already have many of the mathematical constructions of interest.

What is missing in V_ω? The answer to this question is simple: V_ω contains no infinite set. In particular, there is no set of natural numbers. The usual way of constructing the natural numbers in set theory is via finite ordinals. That is, $\emptyset$ corresponds to 0, and if the set N corresponds to the natural number n, then the set $N \cup \{N\}$ corresponds to the natural number $n + 1$. The set ω of natural numbers is the least set such that $\emptyset \in \omega$ and such that $N \in \omega$ implies $N \cup \{N\} \in \omega$. Note that $\emptyset \in V_\omega$ and for every $N \in V_\omega$, $N \cup \{N\} \in V_\omega$. Hence, the set ω is a subset of V_ω. However, ω is not a "set" relative to V_ω, i.e., $\omega \notin V_\omega$. While a finitist may be happy working with the collection V_ω, most mathematicians want to work with the set of natural numbers as well as the many objects one can construct from sets of natural numbers.

We can obtain an extended collection by continuing the construction using transfinite induction. We have already defined V_ω. We can define $V_{\omega+1}$ to be $\mathcal{P}(V_\omega)$. Note that $\omega \in V_{\omega+1}$, so we now have the first infinite set we want. So, $V_{\omega+1}$ satisfies another axiom of Zermelo set theory:

- *Infinity:* ω is a set.

However, $V_{\omega+1}$ is not closed under powersets. In particular, $\omega \in V_{\omega+1}$ but $\mathcal{P}(\omega) \notin V_{\omega+1}$.

In general, for any ordinal α, we define $V_{\alpha+1} := \mathcal{P}(V_\alpha)$ (where $\alpha + 1$ is the ordinal successor $\alpha \cup \{\alpha\}$). For limit ordinals γ (such as ω), we define $V_\gamma := \bigcup_{\alpha<\gamma} V_\alpha$. This yields the von Neumann hierarchy of sets. Let On denote the collection of all ordinals. (The collection On is not a set in Zermelo–Fraenkel set theory.) Let $V := \bigcup_{\alpha \in On} V_\alpha$.

The collection V is transitive, satisfies the axioms above, and contains the infinite set ω. To be precise, $V_{\omega+\omega}$ is already transitive and satisfies the axioms of extensionality, emptyset, pairing, powerset, union, separation, and infinity.

However, $V_{\omega+\omega}$ does not satisfy another closure condition corresponding to the Fraenkel (and Skolem) axiom of replacement. The axiom of replacement intuitively states that images of sets under "class functions" should be sets. Consider the function f sending each $n \in \omega$ to $\omega + n$. $\omega \in V_{\omega+\omega}$, but the image of ω under f – $\{\omega, \omega+1, \ldots, \omega+n, \ldots\}$ – is not in $V_{\omega+\omega}$.

- *Replacement:* If f is a class function and X is a set, then $\{f(x)|x \in X\}$ is a set.

In the usual first-order formulation of ZFC, replacement implies separation. In DeTSeT we will include a dependently typed form of separation which apparently cannot be so easily reduced to replacement.

Why is it problematic that $(\omega + \omega) \notin V_{\omega+\omega}$? One of the nice properties of ZFC is that every well-ordered set is isomorphic to an ordinal. That is, the ordinals provide canonical representations of well-ordered sets. This property fails in $V_{\omega+\omega}$. One can easily find a set X and well-ordering R in $V_{\omega+\omega}$ which is isomorphic to $\omega+\omega$ in the larger set theoretic universe. For example, consider the set $X := \{\langle 0, n\rangle | n \in \omega\} \cup \{\langle 1, n\rangle | n \in \omega\}$ and well-ordering R given by

$$\langle 0,0\rangle < \langle 0,1\rangle < \cdots < \langle 0,n\rangle < \cdots < \langle 1,0\rangle < \langle 1,1\rangle < \cdots < \langle 1,n\rangle < \cdots$$

Further axioms of ZFC include *foundation* (or *regularity*) and *choice* (giving ZFC — Zermelo–Fraenkel with Choice). The set theories omitting one or both of these axioms are also of theoretical interest.

The intuition for the foundation axiom is that the only sets are those which are constructed in the von Neumann hierarchy. Foundation is often formulated as the following elementary statement:

- *Foundation:* For every nonempty X, there is some $x \in X$ such that $x \cap X$ is empty.

The foundation axiom prevents cyclic sets, such as $\Omega = \{\Omega\}$.

The choice axiom can be formulated in many equivalent ways [30]. For example:

- *Choice:* If X is a set of nonempty sets, then there is a function $f : X \to \bigcup X$ such that $f(x) \in x$ for all $x \in X$.

- *Well-Ordering principle:* For every set X there is some binary relation which well-orders X. (To be definite, we are referring to a reflexive well-ordering.)

In order to describe either of these axioms, one needs to first construct pairs in order to assert the existence of a function or relation. By considering "downward closed" sets relative to an ordering, we can reformulate the Well-Ordering Principle without referring to relations.

- *Alternative Well-Ordering Principle:* For every set X there is a set C such that:

  1. C is a set of subsets of X.
  2. For all $x, y \in X$ if $\forall A \in C(x \in A \equiv y \in A)$, then $x = y$. That is, C contains enough sets to distinguish between different elements of X.

3. For all $A, B \in C$, either $A \subseteq B$ or $B \subseteq A$.
4. For all nonempty $Y \subseteq X$, there exist $A \in C$ and $y \in Y$ such that A and Y are disjoint, and for every $B \in C$ either $B \subseteq A$ or $y \in B$.

This alternative form of the well-ordering principle can be formulated without needing to construct intermediate concepts such as ordered pairs. To see that the well-ordering principle implies the alternative form, let C be the set of $Y \subseteq X$ such that $\forall y \in Y \forall x \in X.x \leq y \supset x \in Y$ where $\leq$ is the well-ordering of X (i.e., Y is downward-closed with respect to $\leq$). To see that the alternative form of the well-ordering principle implies the original form, one can use C to define a well-ordering by $x \leq y$ if $\forall Y \in C.y \in Y \supset x \in Y$.

1.2 What is Dependent Type Theory?

The usual method for making the axioms of set theory formal is to do so within first-order logic. DeTSeT is a form of set theory which formulates the axioms in a form of dependent type theory. This was done previously in Automath (earlier in [34] and more recently in [37]).

Generally speaking, a type theory includes terms and types and some notion of when a term has a type. In dependent type theory (at least in the form we consider it here), the types are closed under a Π-constructor, and the terms are closed under λ-abstraction and application. That is, we have types of the form $(\Pi x : A.B(x))$ where A is a type and $B(x)$ is a type depending on $x : A$. Semantically, this is the type of functions f with domain A such that for each $a \in A$, $f(a) \in B(a)$. Π-types are the dependently typed generalization of simple function types. Given certain types for variables and constants, some terms will be well-typed and some types will be well-formed. That is, we will have an inductive characterization of when a term M inhabits a type A in a context Γ relative to a signature Σ. Likewise, we will have an inductive characterization of when a type is "well-formed" in a context. More accurately, we will define when a type A is well-formed at level i with $i \in \{0, 1, 2\}$ and we will define when two terms M and N can be checked to be equal at type A. We will then say that a term M has type A when it can be checked to be equal to itself at type A.

There is always room for variation in the presentation of type theories. In presentations of "pure type systems" [3], terms and types are combined into a single universe of "pseudo-terms". In presentations of the particular type theory LF [27] there are three levels of "objects," "families," and "kinds." We will not strive for generality and will restrict ourselves to just what we need to define the particular dependent type theory which serves as the framework for the dependently typed set theory DeTSeT.

The particular dependent type theory we will use will be just strong enough to conveniently formulate the set theory described above. The dependent type theory is a modified version of the type theory LF (also called λP) as implemented in Twelf [27], which itself can essentially be traced back to the Automath family of type theories [10]. Simply put, the type theory can be obtained from LF via the following modifications:

1. Restrict LF to three primitive type families: `set`, `prop` and `pf` M where M should be of type `prop`. In LF, there are no assumed type families and the signature can

introduce arbitrary new type families. In our dependent type theory, we fix the type families at the beginning.

2. Restrict LF by restricting the complexity of types ($Type^0$, $Type^1$ and $Type^2$). This restricts the terms to be second-order λ-terms (with a further restriction on the occurrence of propositional variables). LF allows arbitrary higher-order types.

3. Extend LF to allow product types $\Sigma_{x:\mathtt{set}}(\mathtt{pf}\,(\phi x))$ for each term $\phi : \mathtt{set} \to \mathtt{prop}$. On the term level, we include the pairing constructor, first and second destructors and the corresponding reductions. LF does not include pairing or Σ-types, but only λ-abstraction, application, and Π-types.

4. Extend LF to include proof irrelevance, so that all terms of type $\mathtt{pf}\,M$ are equal (where $M : \mathtt{prop}$). LF does not contain any proof irrelevance. A more general way of incorporating proof irrelevance into an extension of LF is discussed in [29]. Proof irrelevance was also considered earlier as part of the Automath project.

The purpose of each modifications to LF is to allow a natural representation of mathematics, while keeping the language as restricted and simple as possible.

1.3 What is DeTSeT?

DeTSeT refers to the dependent type theory described above with a fixed set of logical constants appropriate for a ZFC style set theory. Propositions in DeTSeT correspond to terms of type `prop`. Proofs in DeTSeT correspond to terms of type $\mathtt{pf}\,M$ for some $M : \mathtt{prop}$. Sets (or "untyped objects") in DeTSeT correspond to terms of type `set`. Typed objects in DeTSeT correspond to terms of type $\mathtt{cl}\,P$ for some $P : \mathtt{set} \to \mathtt{prop}$. Many terms will actually be of a type such as

$$\Pi x^1 : A^1 \cdots \Pi x^n : A^n.B$$

where B is either `prop`, $\mathtt{pf}\,M$, `set`, or $\mathtt{cl}\,P$. We can describe these terms as "parameterized" propositions, proofs, sets, or typed objects.

The logical constants of DeTSeT correspond to the ZFC axioms described above, plus a description operator and a dependent set constructor (corresponding to a dependently typed form of separation).

2 An Algorithmic Formulation of DeTSeT

Let $\mathcal{V}$ be a countably infinite set of variables. (Note that variables are untyped.) Let $\mathcal{L}$ be a set (disjoint from $\mathcal{V}$) consisting of 46 *logical constants* named as follows:

`in`	`eq`	`eqE`	`not`
`imp`	`equiv`	`and`	`or`
`xmcases`	`notE`	`impI`	`impE`
`equivI1`	`equivI2`	`equivEimp1`	`equivEimp2`
`andEquiv`	`orEquiv`	`all`	`ex`
`allI`	`allE`	`exEquiv`	`exu`
`exuEquiv`	`setextAx`	`emptyset`	`emptysetAx`
`setadjoin`	`setadjoinAx`	`powerset`	`powersetAx`
`setunion`	`setunionAx`	`omega`	`omega0Ax`
`omegaSAx`	`omegaIndAx`	`replAx`	`foundationAx`
`wellorderingAx`	`descr`	`dsetconstr`	`dsetconstrI`
`dsetconstrEL`	`dsetconstrERa`		

Let $\mathcal{A}$, $\mathcal{A}^f$ and $\mathcal{A}^u$ be disjoint infinite sets, each disjoint from both $\mathcal{V}$ and $\mathcal{L}$. We call the members of $\mathcal{A}$ *abbreviations*, the members of $\mathcal{A}^f$ *folding rules* and the members of $\mathcal{A}^u$ *unfolding rules*. Taking the union of the disjoint sets $\mathcal{L}$, $\mathcal{A}$, $\mathcal{A}^f$ and $\mathcal{A}^u$, we obtain the set $\mathcal{C}$ of *constants*.

We use $a, b, f, g, a^0, a^1, \ldots$ to denote generic abbreviations, and use $c, c^0, c^1, \ldots$ to denote generic constants. For each abbreviation a, we choose a distinguished folding rule $(a)^f \in \mathcal{A}^f$ and unfolding rule $(a)^u \in \mathcal{A}^u$. We assume these are chosen distinctly, so that $(a)^f = (b)^f$ iff $a = b$ iff $(a)^u = (b)^u$.

Definition 2.1 The set of *terms* is inductively defined using the following grammar:

$$M :: \mathcal{C}|\mathcal{V}|(M\,M)|(\lambda\mathcal{V}M)|\langle M, M\rangle|\pi_1(M)|\pi_2(M)$$

A term of the form $(M\,N)$ corresponds intuitively to applying the term M to the term N. A term of the form $(\lambda x M)$ corresponds intuitively to the function returning the value M when given the input x. A pair $\langle M, P\rangle$ intuitively corresponds to a term M representing a set and a term P representing a proof that the set M satisfies some property (i.e., belongs to some class). The two projections $\pi_1(M)$ and $\pi_2(M)$ recover the set and proof, respectively, from a pair. We often omit parentheses which should surround application or λ-abstraction. Application associates to the left, as usual.

The set of *types* is inductively defined using the following grammar:

$$A :: \mathtt{set}|\mathtt{prop}|\mathtt{pf}\,M|\mathtt{cl}\,M|\Pi\mathcal{V} : A.A$$

The type `set` is the type of all sets (i.e., all the "objects" of our mathematical universe). The type `prop` is the type of propositions. A type of the form `pf` M is the type of proofs of M, where M should be of type `prop`. (The fact that M should be of type `prop` is not enforced in the definition of types, but in the judgment of validity, which we define later.)

A type of the form $\Pi x : A.B$ is the type of functions from A to B, where the value of the type B depends on the input to the function. As is common in dependent type theory, we use the notation $A \to B$ to stand for $\Pi x : A.B$ where x does not occur free in B. Finally, a type of the form $\mathtt{cl}\,\phi$ is the "class type" of ϕ, where ϕ should be of type $\mathtt{set} \to \mathtt{prop}$. A class type $\mathtt{cl}\,\phi$ corresponds to the pairs $\langle M, P\rangle$ where M is a set and P is a proof of (ϕM), i.e., a proof that M belongs to the class determined by ϕ.

The set of *signatures* is inductively defined as follows:

signatures $\quad (\Sigma) \quad \cdot | \Sigma, a : A = M$

where $a \in \mathcal{A}$. We assume each a is listed at most once in any signature Σ.

The set of *contexts* is inductively defined as follows:

contexts $\quad (\Gamma) \quad \cdot | \Gamma, x : A$

where $x \in \mathcal{V}$. We assume each variable x in a context Γ is listed only once.

We define the set of free variables $\mathbf{FV}(M)$ of a term M and the set free variables $\mathbf{FV}(A)$ of a type A in the usual way.

Since we use $\lambda x M$ instead of $\lambda x : AM$, the definition of terms is independent of the definition of types. We choose this representation because we only consider normal terms, for which we can reconstruct the type of x given the intended type of the term.

For terms and types, we have the usual notion of α-conversion for renaming of bound variables. For convenience, we assume α-equivalent terms as "equal" and assume there are no conflicts with bound variable names whenever convenient.

We let $[M/x]W$ denote the term (resp., type) obtained by substituting the term M for free occurrences of the variable x in the term (resp., type) W. This can be technically defined by induction on W. Similarly, we let $\theta(W)$ denote the result of simultaneously substituting finitely many variables as defined in the usual way.

We consider reduction relations induced by the following reductions, again, in the usual way:

β: $((\lambda x M)N) \to_\beta [N/x]M$

π_1: $\pi_1(\langle M, N\rangle) \to_{\pi_1} M$

π_2: $\pi_2(\langle M, N\rangle) \to_{\pi_2} N$

We say a term or type is β'-normal if there are no β-redexes, no π_1-redexes and no π_2-redexes.

Each of these reductions (along with any combinations of the reductions) is substitutive. That is, if θ is a substitution and W (for a term or type W) reduces to W', then $\theta(W)$ reduces to $\theta(W')$. A proof could follow the same outline as Propositions 2.22 and 2.24 in [12].

Note that $\beta\pi_1\pi_2$-reduction is confluent. We use $M^{\downarrow}$ to denote the $\beta\pi_1\pi_2$-normal form of M, which is unique if it exists. Of course, some terms have no normal form.

2.1 Types of Logical Constants

Each logical constant is associated with a type. We list the correspondence here, along with a short explanation for the type. We write $\mathcal{L}_{\mathcal{T}}(c)$ to denote the type of $c \in \mathcal{L}$ given in the list below.

First, we have logical constants corresponding to formulas in first-order logic with equality.

- $\boxed{\texttt{in} : \quad \texttt{set} \to \texttt{set} \to \texttt{prop}}$
  – the membership relation. We often write $X \in Y$ for the term $(\texttt{in}\, Y\, X)$, which has type `prop` if it is well-typed.

- $\boxed{\texttt{eq} : \quad \texttt{set} \to \texttt{set} \to \texttt{prop}}$
  – the equality relation. We often write $X = Y$ for the term $(\texttt{eq}\, X\, Y)$, which has type `prop` if it is well-typed.

- $\boxed{\texttt{not} : \quad \texttt{prop} \to \texttt{prop}}$
  – negation. We often write $\neg M$ for the term $(\texttt{not}\, M)$, which has type `prop` if it is well-typed.

- $\boxed{\texttt{imp} : \quad \texttt{prop} \to \texttt{prop} \to \texttt{prop}}$
  – implication. We may write $(M \supset N)$ for the term $(\texttt{imp}\, M\, N)$, which has type `prop` if it is well-typed.

- $\boxed{\texttt{equiv} : \quad \texttt{prop} \to \texttt{prop} \to \texttt{prop}}$
  – equivalence. We may write $(M \equiv N)$ for the term $(\texttt{equiv}\, M\, N)$, which has type `prop` if it is well-typed.

- $\boxed{\texttt{and} : \quad \texttt{prop} \to \texttt{prop} \to \texttt{prop}}$
  – conjunction. We may write $(M \wedge N)$ for the term $(\texttt{and}\, M\, N)$, which has type `prop` if it is well-typed.

- $\boxed{\texttt{or} : \quad \texttt{prop} \to \texttt{prop} \to \texttt{prop}}$
  – disjunction. We may write $(M \vee N)$ for the term $(\texttt{or}\, M\, N)$, which has type `prop` if it is well-typed.

- $\boxed{\texttt{all} : \quad (\texttt{set} \to \texttt{prop}) \to \texttt{prop}}$
  – universal quantification. We use $\forall x.\phi$ to denote the set $(\texttt{all}\, (\lambda x \phi))$.

- $\boxed{\texttt{ex} : \quad (\texttt{set} \to \texttt{prop}) \to \texttt{prop}}$
  – existential quantification. We use $\exists x.\phi$ to denote the set $(\texttt{ex}\, (\lambda x \phi))$.

- $\boxed{\texttt{exu} : \quad (\texttt{set} \to \texttt{prop}) \to \texttt{prop}}$
  – unique existential quantification. We use $\exists! x.\phi$ to denote the set $(\texttt{exu}\, (\lambda x \phi))$.

Next we have logical constants corresponding to deduction rules for classical first-order logic with equality. The rules for implication and universal quantification represent standard natural deduction rules. The `xmcases` constant is a cases rule relying on excluded middle for arbitrary formulas, thus guaranteeing the logic is classical. Equivalence is taken as primitive with corresponding natural deduction rules. Conjunction, disjunction, existential quantification and unique existential quantification are also taken as primitive, but with axioms which essentially define each in terms of implication and universal quantification.

- $\boxed{\texttt{eqE} : \Pi x{:}\texttt{set}.\ \ \Pi y{:}\texttt{set}.\ \ \Pi P{:}(\texttt{set} \to \texttt{prop}).\ \ (\texttt{pf}\ (x = y)) \to (\texttt{pf}\ (P\,x)) \to \texttt{pf}\ (P\,y)}$
  – a rule for replacing equal objects with equal objects in a context. Intuitively, if $(\texttt{eqE}\,x\,y\,P\,u\,v)$ is well-typed, then it is a proof of $(P\,y)$ whenever u is a proof of $x = y$ and v is a proof of $(P\,x)$.

- $\boxed{\texttt{xmcases} : \Pi M{:}\texttt{prop}.\ \ \Pi N{:}\texttt{prop}.\ \ ((\texttt{pf}\ M) \to \texttt{pf}\ N) \to ((\texttt{pf}\ \neg M) \to \texttt{pf}\ N) \to \texttt{pf}\ N}$
  If we can prove N from M and we can prove N from $\neg M$, then we have a proof of N.

- $\boxed{\texttt{notE} : \Pi M{:}\texttt{prop}.\ \ \Pi N{:}\texttt{prop}.\ \ (\texttt{pf}\ M) \to (\texttt{pf}\ \neg M) \to \texttt{pf}\ N}$
  If we prove both M and $\neg M$, then we have a proof of N.

- $\boxed{\texttt{impI} : \Pi M{:}\texttt{prop}.\ \ \Pi N{:}\texttt{prop}.\ \ ((\texttt{pf}\ M) \to \texttt{pf}\ N) \to \texttt{pf}\ (M \supset N)}$
  If we prove N from M, then we have a proof of $M \supset N$.

- $\boxed{\texttt{impE} : \Pi M{:}\texttt{prop}.\ \ \Pi N{:}\texttt{prop}.\ \ (\texttt{pf}\ (M \supset N)) \to (\texttt{pf}\ M) \to \texttt{pf}\ N}$
  If we prove $M \supset N$ and M, then we have a proof of N.

- $\boxed{\texttt{equivI1} : \Pi M{:}\texttt{prop}.\ \ \Pi N{:}\texttt{prop}.\ \ (\texttt{pf}\ M) \to (\texttt{pf}\ N) \to \texttt{pf}\ (M \equiv N)}$
  If we prove M and N, then we have a proof of $M \equiv N$.

- $\boxed{\texttt{equivI2} : \Pi M{:}\texttt{prop}.\ \ \Pi N{:}\texttt{prop}.\ \ (\texttt{pf}\ \neg M) \to (\texttt{pf}\ \neg N) \to \texttt{pf}\ (M \equiv N)}$
  If we prove $\neg M$ and $\neg N$, then we have a proof of $M \equiv N$.

- $\boxed{\texttt{equivEimp1} : \Pi M{:}\texttt{prop}.\ \ \Pi N{:}\texttt{prop}.\ \ (\texttt{pf}\ (M \equiv N)) \to (\texttt{pf}\ M) \to \texttt{pf}\ N}$
  If we prove $M \equiv N$ and M, then we have a proof of N.

- $\boxed{\texttt{equivEimp2} : \Pi M{:}\texttt{prop}.\ \ \Pi N{:}\texttt{prop}.\ \ (\texttt{pf}\ (M \equiv N)) \to (\texttt{pf}\ N) \to \texttt{pf}\ M}$
  If we prove $M \equiv N$ and N, then we have a proof of M.

- $\boxed{\texttt{andEquiv} : \Pi M{:}\texttt{prop}.\ \ \Pi N{:}\texttt{prop}.\ \ \texttt{pf}\ ((M \wedge N) \equiv \neg(M \supset \neg N))}$
  This axiom "defines" $\wedge$ in terms of $\neg$ and $\supset$.

- $\boxed{\texttt{orEquiv} : \Pi M{:}\texttt{prop}.\ \ \Pi N{:}\texttt{prop}.\ \ \texttt{pf}\ ((M \vee N) \equiv (\neg M \supset N))}$
  This axiom "defines" $\vee$ in terms of $\neg$ and $\supset$.

- $\boxed{\texttt{allI} : \Pi P{:}(\texttt{set} \to \texttt{prop}).\ \ (\Pi x{:}\texttt{set}.\ \ \texttt{pf}\ (Px)) \to \texttt{pf}\ (\forall x.Px)}$
  If we have a proof of Px for a generic x, then we have a proof of $\forall x.Px$.

- $\boxed{\texttt{allE} : \Pi P{:}(\texttt{set} \to \texttt{prop}).\ \ (\texttt{pf}\ (\forall x.Px)) \to \Pi x{:}\texttt{set}.\ \ \texttt{pf}\ (Px)}$
  If we have a proof of $\forall x.Px$, then we have a proof of Px where x is any set.

- $\boxed{\texttt{exEquiv}: \quad \Pi P{:}(\texttt{set} \to \texttt{prop}). \quad \texttt{pf}\,((\exists x.Px) \equiv \neg(\forall x.\neg(Px)))}$
This axiom "defines" existential quantification (`ex`).

- $\boxed{\texttt{exuEquiv}: \quad \begin{array}{l} \Pi P{:}(\texttt{set} \to \texttt{prop}). \\ \texttt{pf}\,((\exists! x.Px) \equiv (\exists x.(Px) \wedge (\forall y.(Py) \supset (x = y)))) \end{array}}$
This axiom "defines" unique existential quantification (`exu`).

Now we turn to the set theory itself. We start with extensionality. For most of the axioms of ZFC, we can give constructors along with axioms characterizing the constructed sets. One variation from ZFC is that we take `setadjoin` (intuitively taking two sets x and y to the set $\{x\} \cup y$) to be a primitive constructor instead of taking unordered pairing (taking x and y to $\{x, y\}$) as primitive. There are two reasons for this choice. First, one can easily describe finitely enumerated sets $\{a_1, \ldots, a_n\}$ in terms of `setadjoin` and the empty set. Second, the ordinal successor of α can be represented as $(\texttt{setadjoin}\,\alpha\,\alpha)$, thus making the formulation of the axiom of infinity simpler (in terms of primitives).

- $\boxed{\texttt{setextAx}: \quad \texttt{pf}\,(\forall x.\forall y.(\forall z.(z \in x) \equiv (z \in y)) \supset (x = y))}$
– Set extensionality. Two sets x and y are equal if they contain the same elements.

- $\boxed{\texttt{emptyset}: \quad \texttt{set}}$
– the empty set. We use $\emptyset$ to denote the term `emptyset`.

- $\boxed{\texttt{emptysetAx}: \quad \texttt{pf}\,(\forall x.\neg(x \in \emptyset))}$
– nothing is in the empty set.

- $\boxed{\texttt{setadjoin}: \quad \texttt{set} \to \texttt{set} \to \texttt{set}}$
– adjoining a set to another set. $(\texttt{setadjoin}\,x\,y)$ intuitively represents $\{x\} \cup y$. We will use $\{M_1, \ldots, M_n\}$ where $M_1, \ldots, M_n$ are terms as shorthand for the term $(\texttt{setadjoin}\,M_1\,(\texttt{setadjoin} \cdots (\texttt{setadjoin}\,M_n\,\texttt{emptyset}) \cdots))$.

- $\boxed{\texttt{setadjoinAx}: \quad \texttt{pf}\,(\forall x.\forall y.\forall z.(z \in (\texttt{setadjoin}\,x\,y)) \equiv ((z = x) \vee (z \in y)))}$
– axiom characterizing `setadjoin`.

- $\boxed{\texttt{powerset}: \quad \texttt{set} \to \texttt{set}}$
– the power set constructor. We use $\mathcal{P}(X)$ for the term $(\texttt{powerset}\,X)$.

- $\boxed{\texttt{powersetAx}: \quad \texttt{pf}\,(\forall x.\forall y.(y \in \mathcal{P}(x)) \equiv (\forall z.(z \in y) \supset (z \in x)))}$
– axiom characterizing the power set.

- $\boxed{\texttt{setunion}: \quad \texttt{set} \to \texttt{set}}$
– union of a collection of sets. We use $\bigcup X$ for the term $(\texttt{setunion}\,X)$.

- $\boxed{\texttt{setunionAx}: \quad \texttt{pf}\,(\forall x.\forall y.(y \in \bigcup x) \equiv (\exists z.(y \in z) \wedge (z \in x)))}$
– axiom characterizing $\bigcup$.

- $\boxed{\texttt{omega}: \quad \texttt{set}}$
– the first infinite ordinal. We use ω to denote the term `omega`.

- $\boxed{\texttt{omega0Ax}: \quad \texttt{pf}\,(\emptyset \in \omega)}$
  – the empty set (the zero ordinal) is in ω.

- $\boxed{\texttt{omegaSAx}: \quad \texttt{pf}\,(\forall x.(x \in \omega) \supset ((\texttt{setadjoin}\,x\,x) \in \omega))}$
  – ω is closed under ordinal successor.

- $\boxed{\texttt{omegaIndAx}: \quad \begin{array}{r} \texttt{pf}\,(\forall x.((\emptyset \in x) \wedge (\forall y.((y \in \omega) \wedge (y \in x)) \supset ((\texttt{setadjoin}\,y\,y) \in x))) \\ \supset (\forall y.(y \in \omega) \supset (y \in x))) \end{array}}$
  – ω is the least set containing $\emptyset$ (the zero ordinal) and closed under ordinal successor. This is the induction principle for the natural numbers.

The remaining axioms do not correspond directly to set constructors in DeTSeT.

- $\boxed{\texttt{replAx}: \quad \begin{array}{r} \Pi R\!:\!(\texttt{set} \to \texttt{set} \to \texttt{prop}). \quad \texttt{pf}\,(\forall A.(\forall y.(y \in A) \supset (\exists! z.Ryz)) \\ \supset (\exists B.\forall y.(y \in B) \equiv (\exists x.(x \in A) \wedge (Rxy)))) \end{array}}$
  – Replacement Axiom.

- $\boxed{\texttt{foundationAx}: \quad \texttt{pf}\,(\forall x.(\exists y.y \in x) \supset (\exists y.(y \in x) \wedge \neg(\exists z.(z \in y) \wedge (z \in x))))}$
  – Foundation Axiom. If x is nonempty, then there is a $y \in x$ such that y and x are disjoint.

- $$\boxed{\begin{array}{l} \texttt{wellorderingAx}: \texttt{pf}\,(\forall X.\exists C.(((\forall A.(A \in C) \supset (\forall x.(x \in A) \supset (x \in X))) \\ \wedge(\forall x.\forall y.(((x \in X) \wedge (y \in X)) \\ \qquad \supset ((\forall A.(A \in C) \supset ((x \in A) \equiv (y \in A))) \supset (x = y))))) \\ \wedge(\forall A.\forall B.((A \in C) \wedge (B \in C)) \\ \qquad \supset ((\forall x.(x \in A) \supset (x \in B)) \vee (\forall x.(x \in B) \supset (x \in A))))) \\ \wedge(\forall Y.((\forall y.(y \in Y) \supset (y \in X)) \wedge (\exists y.y \in Y)) \\ \qquad \supset (\exists A.\exists y.(((A \in C) \wedge (y \in Y)) \\ \qquad\quad \wedge\neg(\exists z.(z \in A) \wedge (z \in Y))) \\ \qquad\quad \wedge(\forall B.(B \in C) \supset ((\forall x.(x \in B) \supset (x \in A)) \vee (y \in B)))))) \end{array}}$$
  – our formulation of the *Alternative Well-Ordering Principle* described earlier. The axiom is equivalent to the Axiom of Choice.

The logical constants thus far are sufficient for encoding any ZFC theorem and proof. However, the purpose of DeTSeT is to allow more convenient representations of theorems and proofs than first-order ZFC. For this reason we add two more concepts: a description operator and a dependent set constructor. (We conjecture these new concepts do not change the expressive power or deductive strength of the theory over ZFC.)

- $\boxed{\texttt{descr}: \quad \Pi P\!:\!(\texttt{set} \to \texttt{prop}). \quad (\texttt{pf}\,(\exists! x.Px)) \to \texttt{cl}\,P}$
  – description operator. Given a predicate P and a proof that exactly one set satisfies P, return an object in the class determined by P. Note that the description operator is not defined on predicates for which there is no proof that a unique element satisfies P.

- $\boxed{\texttt{dsetconstr} : \quad \Pi A\!:\!\texttt{set}. \quad ((\texttt{cl}\,(\texttt{in}A)) \to \texttt{prop}) \to \texttt{set}}$
  – a dependent set constructor. Given a set A and a proposition $\phi(x)$ depending on a member x of A, $(\texttt{dsetconstr}\,A\,(\lambda x(\phi(x))))$ represents the set $\{x \in A|\phi(x)\}$. In fact, we use $\{x \in A|\phi\}$ to denote the term $(\texttt{dsetconstr}\,A\,(\lambda x\phi))$.

- $\boxed{\texttt{dsetconstrI} : \begin{array}{r} \Pi A\!:\!\texttt{set}. \quad \Pi P\!:\!((\texttt{cl}\,(\texttt{in}A)) \to \texttt{prop}). \\ \Pi a\!:\!(\texttt{cl}\,(\texttt{in}A)). \quad (\texttt{pf}\,(P\,a)) \to \texttt{pf}\,(\pi_1(a) \in \{x \in A|P\,x\}) \end{array}}$
  – an introduction rule for the dependent set constructor.

- $\boxed{\texttt{dsetconstrEL} : \begin{array}{r} \Pi A\!:\!\texttt{set}. \quad \Pi P\!:\!((\texttt{cl}\,(\texttt{in}A)) \to \texttt{prop}). \quad \Pi y\!:\!\texttt{set}. \\ (\texttt{pf}\,(y \in \{x \in A|P\,x\})) \to \texttt{pf}\,(y \in A) \end{array}}$
  – a left elimination rule for the dependent set constructor.

- $\boxed{\texttt{dsetconstrERa} : \begin{array}{r} \Pi A\!:\!\texttt{set}. \quad \Pi P\!:\!((\texttt{cl}\,(\texttt{in}A)) \to \texttt{prop}). \quad \Pi a\!:\!(\texttt{cl}\,(\texttt{in}A)). \\ (\texttt{pf}\,(\pi_1(a) \in \{x \in A|P\,x\})) \to \texttt{pf}\,(P\,a) \end{array}}$
  – a right elimination rule for the dependent set constructor.

2.2 Eta and Surjective Pairing

Two forms of reduction we do not build into the notion of reduction are η and π:

η: $(\lambda x(Mx)) \to_\eta M$ if x is not free in M.

π: $\langle \pi_1(M), \pi_2(M) \rangle \to_\pi M$

Instead, there will be some rules in the typing judgment where we either want to η-expand or π-expand on the fly. We introduce some notation to facilitate this.

- If M is a term of the form $(\lambda x N)$, then let $\mathbf{x}_\lambda^M$ denote x and $\mathbf{B}_\lambda^M$ denote N. If M is any other term, then let $\mathbf{x}_\lambda^M$ be a variable not occurring in M and $\mathbf{B}_\lambda^M$ denote $(M\mathbf{x}_\lambda^M)$. Note that in the first case, $(\lambda \mathbf{x}_\lambda^M \mathbf{B}_\lambda^M)$ is identical to M. In the second case, $(\lambda \mathbf{x}_\lambda^M \mathbf{B}_\lambda^M)$ η-reduces to M.

- If M is a term of the form $\langle N, P \rangle$, then let $\mathbf{fst}^M$ denote N and $\mathbf{snd}^M$ denote P. If M is any other term, then let $\mathbf{fst}^M$ denote $\pi_1(M)$ and $\mathbf{snd}^M$ denote $\pi_2(M)$. In the first case, $\langle \mathbf{fst}^M, \mathbf{snd}^M \rangle$ is identical to M. In the second case, $\langle \mathbf{fst}^M, \mathbf{snd}^M \rangle$ π-reduces to M.

2.3 Types for Folding and Unfolding Abbreviations

When an abbreviation $a : A = M$ is declared in a signature, we need to be able to fold and unfold occurrences a to the definition M. We could include in the notion of reduction with rules such as

$$a \rightarrow M$$

to expand a by its definition M in place. Such rules are often used to give a notion of δ-reduction. One can combine δ-reduction with $\beta\pi_1\pi_2$-reduction to give a more general notion of structural equality on terms than $\beta\pi_1\pi_2$-reduction alone provides. The advantage of δ-reduction is that this makes type checking more powerful. Unfortunately, it also means that to check if two terms are equal one may need to expand all abbreviations in the two terms and reduce to a normal form. When there are too many nested abbreviations (a situation unavoidable in mathematics), this is an unreasonable operation.

The alternative we take is the following. Whenever $a : A = M$ occurs in a signature, one is allowed to use the fact that a has type A and, when appropriate, use the constant $(a)^f$ to fold M to be a and the constant $(a)^u$ to unfold a to be M. To make this precise, we define a partial function $\mathit{Leib}(A, M, N)$ which takes a type A and two terms M and N and, if defined, returns a type B. This function is defined by induction on the type A as follows:

- $\mathit{Leib}((\Pi x{:}B.\, C), M, N) :=$

  $$(\Pi x{:}B. \quad \mathit{Leib}(C, [x/\mathbf{x}_\lambda^M]\mathbf{B}_\lambda^M, [x/\mathbf{x}_\lambda^N]\mathbf{B}_\lambda^N))$$

  when $\mathit{Leib}(C, [x/\mathbf{x}_\lambda^M]\mathbf{B}_\lambda^M, [x/\mathbf{x}_\lambda^N]\mathbf{B}_\lambda^N)$ is defined.
- $\mathit{Leib}(\mathtt{prop}, M, N) := \mathtt{pf}\, M \rightarrow \mathtt{pf}\, N$.
- $\mathit{Leib}(\mathtt{set}, M, N) := (\Pi P{:}(\mathtt{set} \rightarrow \mathtt{prop}).\quad \mathtt{pf}\,(P\, M) \rightarrow \mathtt{pf}\,(P\, N))$.
- $\mathit{Leib}(\mathtt{cl}\, \phi, M, N) := (\Pi P{:}(\mathtt{set} \rightarrow \mathtt{prop}).\quad \mathtt{pf}\,(P\, \pi_1(M)) \rightarrow \mathtt{pf}\,(P\, \pi_1(N)))$.

Note that there is no case corresponding to when A is $\mathtt{pf}\, P$. Thus, the function is partial, and in particular is not defined for types of the form $\Pi x^1{:}A^1 \cdots \Pi x^n{:}A^n.\, \mathtt{pf}\, P$. One cannot fold or unfold abbreviations of such a type, but due to proof irrelevance, there is no reason to do so. Any two terms of a type of the form $\Pi x^1 : A^1 \cdots \Pi x^n : A^n.\mathtt{pf}\, P$ will already be equal (in terms of the typing judgments).

Intuitively, $\mathit{Leib}(A, M, N)$ represents Leibniz equality of A and is of the form $\Pi x^1 : A^1. \cdots \Pi x^n : A^n.\mathtt{set}$ or $\Pi x^1 : A^1. \cdots \Pi x^n : A^n.\mathtt{cl}\, P$. If A is of the form $\Pi x^1 : A^1. \cdots \Pi x^n : A^n.\mathtt{prop}$, then $\mathit{Leib}(A, M, N)$ corresponds to implication (and so one obtains equivalence by combining $\mathit{Leib}(A, M, N)$ with $\mathit{Leib}(A, N, M)$.

Now, given a definition $a : A = M$ in a signature M, if $\mathit{Leib}(A, a, M)$ and $\mathit{Leib}(A, M, a)$ are defined (either both are defined or both are undefined), then the folding constant $(a)^f$ has type $\mathit{Leib}(A, M, a)$ and the unfolding constant $(a)^u$ has type $\mathit{Leib}(A, a, M)$. See the rules xaf and xau in Figure 1.

2.4 Typing Judgments

We next introduce judgments relating terms and types in contexts.

- $\vdash \Sigma$ `sig` means Σ is a valid signature.
- $\vdash_\Sigma \Gamma$ `ctx` means Γ is a valid context.
- $\Gamma \vdash_\Sigma A : Type^i$ means the type A is a valid type at level $i \in \{0, 1, 2\}$.
- $\Gamma \vdash_\Sigma M \sim N \uparrow A$ means we can check M and N are equal as members of the type A.
- $\Gamma \vdash_\Sigma M \sim N \downarrow A$ means we can extract a type A such that M and N are equal in A.

We have listed the judgments in the order above because the order seems a natural one for explanation. In terms of dependencies, they are listed in the wrong order. In order to know if a signature Σ or a context Γ is valid, we must check if types are valid. In order to check if a type such as $\texttt{pf}\, M$ is valid in a context Γ, we must check if M has type `prop` (i.e., if M equals itself at type `prop`) in context Γ. One way to check if a term P has a type $\texttt{pf}\, M$ is to extract a type $\texttt{pf}\, N$ and check if M and N are equal at type `prop`.

Accordingly, we begin by giving the algorithmic rules for the last two judgments for determining if two terms are congruent relative to a type, as well as rules for determining if A is a valid type. These rules are given in Figure 1. The rules for validity of signatures and contexts are given in Figure 2. In the rule *coercepf* (the rule which builds in proof irrelevance), we make use of the normal forms of the terms Q and R. If either of these terms has no normal form, the rule does not apply. (That is, an implicit premiss of the rule is that Q and R have a normal form.) We conjecture that if the signature Σ is valid, the context Γ is valid, and $\Gamma \vdash M \sim M \downarrow \texttt{pf}\, Q$ is derivable, then Q has a normal form. If this is true, then the implicit premiss would also be redundant. For the record, here is the general conjecture:

Conjecture 1: If $\vdash \Sigma$ sig, $\vdash_\Sigma \Gamma$ ctx, and $\Gamma \vdash_\Sigma M \sim N \downarrow A$ hold, then the type A has a normal form.

For convenience, we refer to two special cases of judgments using shorthand.

- $\Gamma \vdash M \uparrow A$ stands for $\Gamma \vdash M \sim M \uparrow A$, intuitively meaning we can check the term M has the type A.
- $\Gamma \vdash M \downarrow A$ stands for $\Gamma \vdash M \sim M \downarrow A$, intuitively meaning we can extract the type A for M.

$$\frac{x : A \in \Gamma}{\Gamma \vdash_\Sigma x \sim x \downarrow A}\,xv \qquad \frac{c \in \mathcal{L}}{\Gamma \vdash_\Sigma c \sim c \downarrow \mathcal{L}_{\mathcal{T}}(c)}\,xs \qquad \frac{a : A = M \in \Sigma}{\Gamma \vdash_\Sigma a \sim a \downarrow A}\,xa$$

$$\frac{a : A = M \in \Sigma \quad Leib(A, M, a) = B}{\Gamma \vdash_\Sigma (a)^f \sim (a)^f \downarrow B}\,xaf \qquad \frac{a : A = M \in \Sigma \quad Leib(A, a, M) = B}{\Gamma \vdash_\Sigma (a)^u \sim (a)^u \downarrow B}\,xau$$

$$\frac{\Gamma \vdash_\Sigma M \sim P \downarrow (\Pi x{:}A.B) \quad \Gamma \vdash_\Sigma N \sim Q \uparrow A}{\Gamma \vdash_\Sigma (MN) \sim (PQ) \downarrow ([N/x]B)}\,xa$$

$$\frac{\Gamma \vdash_\Sigma M \sim N \downarrow \mathtt{cl}\,\phi}{\Gamma \vdash_\Sigma \pi_1(M) \sim \pi_1(N) \downarrow \mathtt{set}}\,xpi1 \qquad \frac{\Gamma \vdash_\Sigma M \sim N \downarrow \mathtt{cl}\,\phi}{\Gamma \vdash_\Sigma \pi_2(M) \sim \pi_2(N) \downarrow \mathtt{pf}\,(\phi\,\pi_1(M))}\,xpi2$$

$$\frac{\Gamma, z : A \vdash_\Sigma [z/\mathbf{x}_\lambda^M]\mathbf{B}_\lambda^M \sim [z/\mathbf{x}_\lambda^N]\mathbf{B}_\lambda^N \uparrow [z/x]B \quad z \in \mathcal{V} \text{ fresh}}{\Gamma \vdash_\Sigma M \sim N \uparrow (\Pi x{:}A.B)}\,c\lambda^z$$

$$\frac{\Gamma \vdash_\Sigma \mathbf{fst}^M \sim \mathbf{fst}^N \uparrow \mathtt{set} \quad \Gamma \vdash_\Sigma \mathbf{snd}^M \sim \mathbf{snd}^N \uparrow \mathtt{pf}\,(\phi\,\mathbf{fst}^M)}{\Gamma \vdash_\Sigma M \sim N \uparrow \mathtt{cl}\,\phi}\,cp$$

$$\frac{\Gamma \vdash_\Sigma M \sim N \downarrow B \quad B \in \{\mathtt{set}, \mathtt{prop}\}}{\Gamma \vdash_\Sigma M \sim N \uparrow B}\,coerce$$

$$\frac{\begin{array}{cc} \Gamma \vdash_\Sigma M \sim M \downarrow \mathtt{pf}\,Q & \Gamma \vdash_\Sigma Q^\downarrow \sim P \uparrow \mathtt{prop} \\ \Gamma \vdash_\Sigma N \sim N \downarrow \mathtt{pf}\,R & \Gamma \vdash_\Sigma R^\downarrow \sim P \uparrow \mathtt{prop} \end{array}}{\Gamma \vdash_\Sigma M \sim N \uparrow \mathtt{pf}\,P}\,coercepf$$

$$\frac{\Gamma \vdash_\Sigma A : Type^i \quad \Gamma, z : A \vdash_\Sigma [z/x]B : Type^j \quad \begin{array}{c} i \in \{0,1\}, j \in \{0,1,2\} \\ z \in \mathcal{V} \text{ fresh} \end{array}}{\Gamma \vdash_\Sigma (\Pi x{:}A.B) : Type^{\max(i+1,j)}}\,vt\Pi$$

$$\frac{}{\Gamma \vdash_\Sigma \mathtt{set}{:}Type^0}\,vto \qquad \frac{}{\Gamma \vdash_\Sigma \mathtt{prop} : Type^1}\,vtp \qquad \frac{\Gamma \vdash_\Sigma M \sim M \uparrow \mathtt{prop}}{\Gamma \vdash_\Sigma \mathtt{pf}\,M : Type^0}\,vtpf$$

$$\frac{\Gamma \vdash_\Sigma M \sim M \uparrow (\mathtt{set} \to \mathtt{prop})}{\Gamma \vdash_\Sigma \mathtt{cl}\,M : Type^0}\,vtcl \qquad \frac{\Gamma \vdash_\Sigma A : Type^i \quad i \in \{0,1\}}{\Gamma \vdash_\Sigma A : Type^{i+1}}\,vtu$$

Figure 1: Rules for Algorithmic Typing Judgments

$$\frac{}{\vdash \cdot \text{ sig}}\ \mathit{emptysig}$$

$$\frac{\vdash \Sigma \text{ sig} \quad \cdot \vdash_\Sigma A : \mathit{Type}^2 \quad \cdot \vdash_\Sigma M \sim M \uparrow A \quad a \in \mathcal{A} \setminus \mathit{dom}(\Sigma)}{\vdash \Sigma, a : A = M \text{ sig}}\ \mathit{conssig}$$

$$\frac{}{\vdash_\Sigma \cdot \text{ ctx}}\ \mathit{emptyctx}$$

$$\frac{\vdash_\Sigma \Gamma \text{ ctx} \quad \Gamma \vdash_\Sigma A : \mathit{Type}^1 \quad x \in \mathcal{V} \setminus \mathit{dom}(\Gamma)}{\vdash_\Sigma \Gamma, x : A \text{ ctx}}\ \mathit{consctx}$$

Figure 2: Rules for Valid Signatures and Contexts

3 Developing a Signature for Basic Mathematics

Starting from the logical constants given above, one can construct a signature of abbreviations which includes the basic inference rules one wants as well as the usual objects of mathematical interest: ordered pairs, Cartesian products, and functions. Note that in particular one can use these abbreviations to encode simply typed λ-calculus within DeTSeT.

Without giving an entire such signature, we list some of the abbreviations along with their types. In some cases we will also include the term defining the abbreviation. Since many terms (especially those corresponding to proofs) will be large and difficult to understand when written out in full detail, we will adopt some notation conventions.

First, we can give a modified form of the right elimination rule for the dependent set constructor `dsetconstr`. The logical constant corresponding to the right elimination rule is `dsetconstrERa`. Suppose A has type `set`, P has type $((\mathtt{cl}\,(\mathtt{in}\,A)) \rightarrow \mathtt{prop})$ and a has type $(\mathtt{cl}\,(\mathtt{in}\,A))$. Then $(\mathtt{dsetconstrERa}\,A\,P\,a)$ is a term which expects a proof of $\pi_1(a) \in \{x \in A | P\,x\}$ and returns a proof of $(P\,a)$. Suppose y is of type `set` and u is of type $\mathtt{pf}\,(y \in \{x \in A | P\,x\})$. Can we prove $(P\,y)$ holds? Strictly speaking, this term is ill-typed since P expects a term of type $\mathtt{cl}\,(\mathtt{in}\,A)$ and y is of type `set`. However, we can use a proof (any proof) of $y \in A$ to form a term, essentially equal to y, of type $\mathtt{cl}\,(\mathtt{in}\,A)$. Note that $(\mathtt{dsetconstrEL}\,A\,P\,y\,u)$ is of type $\mathtt{pf}\,(y \in A)$. So, the pair $\langle y, (\mathtt{dsetconstrEL}\,A\,P\,y\,u)\rangle$ has type $\mathtt{cl}\,(\mathtt{in}\,A)$. Now, we can form the term

$$(\mathtt{dsetconstrERa}\,A\,P\,\langle y, (\mathtt{dsetconstrEL}\,A\,P\,y\,u)\rangle u)$$

which has type $\mathtt{pf}\,(P\,\langle y, (\mathtt{dsetconstrEL}\,A\,P\,y\,u))$. This allows us to apply the right elimination rule to an arbitrary y which we do not know a priori is a member of the bounding set A. We will abbreviate this rule using the name `dsetconstrER` which has type

$$\begin{aligned}\Pi A\!:\!\mathtt{set}.\Pi P\!:\!((\mathtt{cl}\,(\mathtt{in}A)) \rightarrow \mathtt{prop}).\Pi y\!:\!\mathtt{set}.(\mathtt{pf}\,(y \in \{x \in A | P\,x\}))\\ \rightarrow \mathtt{pf}\,(P\,\langle y, (\mathtt{dsetconstrEL}\,A\,P\,y\,u))\end{aligned}$$

and is defined by the term

$$\lambda A \lambda P \lambda y \lambda u (\mathtt{dsetconstrERa}\,A\,P\,\langle y, (\mathtt{dsetconstrEL}\,A\,P\,y\,u)\rangle u)$$

While this proof term is of a reasonable size, and can be understood given the explanation above, more complex proof terms will be large and difficult to understand. A way to avoid this problem, is to use a temporary symbol to denote subterms (except variables) which have the type $\mathtt{pf}\,M$ for some M. Each such subterm can be thought of as a "proof step" or "proof line." For example, the subterm $(\mathtt{dsetconstrEL}\,A\,P\,y\,u)$ above has type $\mathtt{pf}\,(y \in A)$ in the appropriate context Γ. We can use the symbol **L1** to denote the proof term and then rewrite the definition of `dsetconstrER` as

$$\lambda A \lambda P \lambda y \lambda u (\mathtt{dsetconstrERa}\,A\,P\,\langle y, \mathbf{L1}\rangle u)$$

This notation has the advantage that it hides the details of proofs (which are unimportant in type-checking because of proof irrelevance). As we introduce this temporary symbol **L1**, we should explicitly write what the term **L1** denotes. We will call these temporary symbols (which will be named $\mathbf{L}i$ for some i) "proof steps" and list them after the term is given. In order to represent the term as a whole, it is enough to know what term each proof step represents. However, to understand the term, it is helpful to also give the *type* of each proof step. Hence we will give each proof step in the form

$\mathbf{L}i$ $\quad \mathtt{pf}\, M$ $\qquad\qquad$ by:P

where P is the term $\mathbf{L}i$ represents and $\mathtt{pf}\, M$ is the type of P in the appropriate context. When the proof steps are listed in order, they obtain the appearance of a linearized natural deduction proof. We leave the context implicit, but as a further aid to readability, we sometimes include lines of the form

Let x have type A

when x is a bound variable of type A and

$\mathcal{D}$ Assume M

when $\mathcal{D}$ is a bound variable of type $\mathtt{pf}\, M$. To be precise, we include these lines whenever the corresponding bound variable is in the context of *some* but not *all* proof steps. The "Let" or "Assume" statement will precede the first proof step which assumes the variable is in context, but it is the task of the reader to note when the variable has been discharged from the context. Again, no context or type information is necessary to uniquely reconstruct the term as a whole. All one needs is the unique term associated with each proof step $\mathbf{L}i$.

Using a proof step representation, we can write the abbreviation `dsetconstrER` as follows

$\mathtt{dsetconstrER} : \quad \Pi A\!:\!\mathtt{set}. \quad \Pi\phi\!:\!(\mathtt{cl}\,(\mathtt{in}\, A) \to \mathtt{prop}). \quad \Pi x\!:\!\mathtt{set}.$
$\Pi\mathcal{D}\!:\!\mathtt{pf}\,(x \in \{a \in A \,|\, \phi\, a\}). \quad \mathtt{pf}\,(\phi\,\langle\, x, \mathbf{L1}\,\rangle)$
$= \lambda A\, \lambda\phi\, \lambda x\, \lambda\mathcal{D}\, \mathtt{dsetconstrERa}\, A\, \phi\, \langle\, x, \mathbf{L1}\,\rangle\, \mathcal{D}$
using proof step
$\mathbf{L1}$: $\quad \mathtt{pf}\,(x \in A)$ $\qquad\qquad$ by: $\mathtt{dsetconstrEL}\, A\, \phi\, x\, \mathcal{D}$

Note that the type of ϕ in `dsetconstrER` is $\mathtt{cl}\,(\mathtt{in}\, A) \to \mathtt{prop}$. One important notational simplification is simply to write any class type $\mathtt{cl}\, P$ as P. Given this simplification, we can write the type of ϕ as $(\mathtt{in}\, A) \to \mathtt{prop}$. A further simplification is to omit the "`in`" in class types induced by sets (as is the case here). That is, we can write the type of ϕ as $A \to \mathtt{prop}$. Hence we can write the abbreviation `dsetconstrER` as follows:

$\mathtt{dsetconstrER} : \quad \Pi A\!:\!\mathtt{set}. \quad \Pi\phi\!:\!(A \to \mathtt{prop}). \quad \Pi x\!:\!\mathtt{set}.$
$\Pi\mathcal{D}\!:\!\mathtt{pf}\,(x \in \{a \in A \,|\, \phi\, a\}). \quad \mathtt{pf}\,(\phi\,\langle\, x, \mathbf{L1}\,\rangle)$
$= \lambda A\, \lambda\phi\, \lambda x\, \lambda\mathcal{D}\, \mathtt{dsetconstrERa}\, A\, \phi\, \langle\, x, \mathbf{L1}\,\rangle\, \mathcal{D}$
using proof step
$\mathbf{L1}$: $\quad \mathtt{pf}\,(x \in A)$ $\qquad\qquad$ by: $\mathtt{dsetconstrEL}\, A\, \phi\, x\, \mathcal{D}$

In addition to the usual quantifiers, we can define dependent bounded quantifiers `dall` and `dex` along with inference rules. Intuitively, $(\mathtt{dall}\, A\, (\lambda x \phi))$ means $\forall x \in A.\phi(x)$. The reason for saying these are *dependent* quantifiers is because within the body of the binder one can use the (proof of the) fact that the bound variable is in the bounding set. For example, if P represents the set of positive real numbers, then one can write $\exists x \in P.(\frac{x^2-1}{x} = 0)$ where one needs to know $x \neq 0$ (which should follow from $x \in P$) to form the proposition $(\frac{x^2-1}{x} = 0)$.

$\texttt{dall}: \quad \Pi A{:}\texttt{set}. \quad (A \to \texttt{prop}) \to \texttt{prop} = \lambda A\, \lambda\phi\, \{a \in A \mid \phi\, a\} = A$

$\texttt{dex}: \quad \Pi A{:}\texttt{set}. \quad (A \to \texttt{prop}) \to \texttt{prop} = \lambda A\, \lambda\phi\, \neg\{a \in A \mid \phi\, a\} = \emptyset$

Note that the term defining dall is

$$\lambda A \lambda \phi ((\texttt{dsetconstr}\, A\, (\lambda x(\phi\, x))) = A)$$

That is, $(\texttt{dall}\, A\, \phi)$ is defined to be the proposition $\{x \in A | \phi(x)\} = A$. One might try to define $(\texttt{dall}\, A\, \phi)$ as $\forall x.x \in A \supset (\phi\, x)$, except that this is ill-typed (since x would have type set).

From now on, we will use $(\forall x \in A.\phi)$ to denote a term of the form $(\texttt{dall}\, A\, (\lambda x \phi))$ and use $(\exists x \in A.\phi)$ to denote a term of the form $(\texttt{dex}\, A\, (\lambda x \phi))$.

One can also define dependent versions of conjunction (dand) and implication (dimp). This allows one to represent a proposition such as $A \wedge B$ where one needs to know A is true in order to construct B. For example, consider the proposition $(x > 0) \wedge (\frac{x^2-1}{x} = 0)$. In order to facilitate these abbreviations, we first define prop2set, which embeds propositions into sets by taking a proposition ϕ to the set $\{\emptyset\}$ if ϕ is true and to $\emptyset$ if ϕ is false. We also prove a rule prop2setE which says the proposition is true if the corresponding set has an element.

$\texttt{prop2set}: \quad \texttt{prop} \to \texttt{set} = \lambda\phi\, \{a \in \mathcal{P}\,\emptyset \mid \phi\}$

$\texttt{prop2setE}: \quad \Pi\phi{:}\texttt{prop}. \quad \Pi x{:}\texttt{set}. \quad \texttt{pf}\, (x \in (\texttt{prop2set}\, \phi)) \to \texttt{pf}\, \phi$
$= \lambda\phi\, \lambda x\, \lambda\mathcal{D}\, \texttt{dsetconstrER}\, (\mathcal{P}\,\emptyset)\, (\lambda a\, \phi)\, x\, \mathbf{L1}$
using proof step
$\mathbf{L1}: \quad \texttt{pf}\, (x \in \{a \in \mathcal{P}\,\emptyset \mid \phi\}) \quad$ by: $\quad (\texttt{prop2set})^u\, \phi\, (\lambda A\, x \in A)\, \mathcal{D}$

Now we can define dand. Note that the definition of dand makes use of prop2setE in a proof term.

$\texttt{dand}: \quad \Pi\phi{:}\texttt{prop}. \quad (\texttt{pf}\, \phi \to \texttt{prop}) \to \texttt{prop} = \lambda\phi\, \lambda\psi\, \exists a \in (\texttt{prop2set}\, \phi).\, \psi\, \mathbf{L2}$
using proof steps
$\mathbf{L1}: \quad \texttt{pf}\, (\pi_1(a) \in (\texttt{prop2set}\, \phi)) \quad$ by: $\quad \pi_2(a)$
$\mathbf{L2}: \quad \texttt{pf}\, \phi \quad$ by: $\quad \texttt{prop2setE}\, \phi\, \pi_1(a)\, \mathbf{L1}$

We now simplify the notation by omitting all occurrences of π_1 and π_2. It is left to the reader to recognize when one of these is omitted. The idea is that if a term has a class type, but is used as if it has type set, then a π_1 has been omitted. If a term has a class type, but is used as if it has a proof type, then a π_2 has been omitted. We can now present dand as follows:

$\texttt{dand}: \quad \Pi\phi{:}\texttt{prop}. \quad (\texttt{pf}\, \phi \to \texttt{prop}) \to \texttt{prop} = \lambda\phi\, \lambda\psi\, \exists a \in (\texttt{prop2set}\, \phi).\, \psi\, \mathbf{L2}$
using proof steps
$\mathbf{L1}: \quad \texttt{pf}\, (a \in (\texttt{prop2set}\, \phi)) \quad$ by: $\quad a$
$\mathbf{L2}: \quad \texttt{pf}\, \phi \quad$ by: $\quad \texttt{prop2setE}\, \phi\, a\, \mathbf{L1}$

We define dependent implication similarly:

$\texttt{dimp}: \quad \Pi\phi{:}\texttt{prop}. \quad (\texttt{pf}\, \phi \to \texttt{prop}) \to \texttt{prop} = \lambda\phi\, \lambda\psi\, \forall a \in (\texttt{prop2set}\, \phi).\, \psi\, \mathbf{L2}$
using proof steps

$\mathbf{L1}$: $\mathtt{pf}\,(a \in (\mathtt{prop2set}\,\phi))$ by: a
$\mathbf{L2}$: $\mathtt{pf}\,\phi$ by: $\mathtt{prop2setE}\,\phi\,a\,\mathbf{L1}$

We can define `false`.

$\mathtt{false} : \quad \mathtt{prop} = \emptyset \in \emptyset$

We use $\bot$ to denote the term `false`.

The following are rules for eliminating the empty set, eliminating false, and introducing negation.

$\mathtt{emptysetE} : \quad \Pi x{:}\mathtt{set}.\ \ \mathtt{pf}\,(x \in \emptyset) \to \Pi\phi{:}\mathtt{prop}.\ \ \mathtt{pf}\,\phi$
$= \lambda x\,\lambda\mathcal{D}\,\lambda\phi\,\mathtt{notE}\,(x \in \emptyset)\,\phi\,\mathcal{D}\,\mathbf{L2}$
using proof steps
$\mathbf{L1}$: $\mathtt{pf}\,(\forall y.\,\neg y \in \emptyset)$ by: $\mathtt{emptysetAx}$
$\mathbf{L2}$: $\mathtt{pf}\,(\neg x \in \emptyset)$ by: $\mathtt{allE}\,(\lambda y\,\neg y \in \emptyset)\,\mathbf{L1}\,x$

$\mathtt{falseE} : \quad \Pi\phi{:}\mathtt{prop}.\ \ \mathtt{pf}\,\bot \to \mathtt{pf}\,\phi = \lambda\phi\,\lambda\mathcal{D}\,\mathtt{emptysetE}\,\emptyset\,\mathbf{L1}\,\phi$
using proof step
$\mathbf{L1}$: $\mathtt{pf}\,(\emptyset \in \emptyset)$ by: $(\mathtt{false})^u\,\mathcal{D}$

$\mathtt{notI} : \quad \Pi\phi{:}\mathtt{prop}.\ \ (\mathtt{pf}\,\phi \to \mathtt{pf}\,\bot) \to \mathtt{pf}\,(\neg\phi)$
$= \lambda\phi\,\lambda\mathcal{D}\,\mathtt{xmcases}\,\phi\,(\neg\phi)\,(\lambda\mathcal{E}\,\mathbf{L2})\,(\lambda\mathcal{F}\,\mathcal{F})$
using proof steps
$\mathbf{L1}$: $\mathtt{pf}\,\bot$ by: $\mathcal{D}\,\mathcal{E}$
$\mathbf{L2}$: $\mathtt{pf}\,(\neg\phi)$ by: $\mathtt{falseE}\,(\neg\phi)\,\mathbf{L1}$

We can also derive a rule for proof by contradiction:

$\mathtt{contradiction} : \quad \Pi\phi{:}\mathtt{prop}.\ \ (\mathtt{pf}\,(\neg\phi) \to \mathtt{pf}\,\bot) \to \mathtt{pf}\,\phi$
$= \lambda\phi\,\lambda\mathcal{D}\,\mathtt{xmcases}\,\phi\,\phi\,(\lambda\mathcal{E}\,\mathcal{E})\,(\lambda\mathcal{F}\,\mathbf{L2})$
using proof steps
$\mathbf{L1}$: $\mathtt{pf}\,\bot$ by: $\mathcal{D}\,\mathcal{F}$
$\mathbf{L2}$: $\mathtt{pf}\,\phi$ by: $\mathtt{falseE}\,\phi\,\mathbf{L1}$

The way to construct proof terms should be clear from the examples above. Below we list a number of other logical rules which can be derived. For the sake of brevity, we omit the proof terms and give only the types.

$\mathtt{dnegE} : \quad \Pi\phi{:}\mathtt{prop}.\ \ \mathtt{pf}\,(\neg\neg\phi) \to \mathtt{pf}\,\phi$

$\mathtt{dnegI} : \quad \Pi\phi{:}\mathtt{prop}.\ \ \mathtt{pf}\,\phi \to \mathtt{pf}\,(\neg\neg\phi)$

$\mathtt{contrapositive1} : \quad \Pi\phi{:}\mathtt{prop}.\ \ \Pi\psi{:}\mathtt{prop}.\ \ (\mathtt{pf}\,\phi \to \mathtt{pf}\,\psi) \to \mathtt{pf}\,(\neg\psi) \to \mathtt{pf}\,(\neg\phi)$

$\mathtt{contrapositive2} : \quad \Pi\phi{:}\mathtt{prop}.\ \ \Pi\psi{:}\mathtt{prop}.\ \ (\mathtt{pf}\,(\neg\phi) \to \mathtt{pf}\,\psi) \to \mathtt{pf}\,(\neg\psi) \to \mathtt{pf}\,\phi$

$\mathtt{contrapositive3} : \quad \Pi\phi{:}\mathtt{prop}.\ \ \Pi\psi{:}\mathtt{prop}.\ \ (\mathtt{pf}\,(\neg\phi) \to \mathtt{pf}\,(\neg\psi)) \to \mathtt{pf}\,\psi \to \mathtt{pf}\,\phi$

$\mathtt{contrapositive4} : \quad \Pi\phi{:}\mathtt{prop}.\ \ \Pi\psi{:}\mathtt{prop}.\ \ (\mathtt{pf}\,\phi \to \mathtt{pf}\,(\neg\psi)) \to \mathtt{pf}\,\psi \to \mathtt{pf}\,(\neg\phi)$

$\texttt{andI}: \quad \Pi\phi{:}\texttt{prop}.\quad \Pi\psi{:}\texttt{prop}.\quad \texttt{pf}\,\phi \to \texttt{pf}\,\psi \to \texttt{pf}\,(\phi \wedge \psi)$

$\texttt{andEL}: \quad \Pi\phi{:}\texttt{prop}.\quad \Pi\psi{:}\texttt{prop}.\quad \texttt{pf}\,(\phi \wedge \psi) \to \texttt{pf}\,\phi$

$\texttt{andER}: \quad \Pi\phi{:}\texttt{prop}.\quad \Pi\psi{:}\texttt{prop}.\quad \texttt{pf}\,(\phi \wedge \psi) \to \texttt{pf}\,\psi$

$\texttt{orIL}: \quad \Pi\phi{:}\texttt{prop}.\quad \Pi\psi{:}\texttt{prop}.\quad \texttt{pf}\,\phi \to \texttt{pf}\,(\phi \vee \psi)$

$\texttt{orIR}: \quad \Pi\phi{:}\texttt{prop}.\quad \Pi\psi{:}\texttt{prop}.\quad \texttt{pf}\,\psi \to \texttt{pf}\,(\phi \vee \psi)$

$\texttt{orE}: \quad \Pi\phi{:}\texttt{prop}.\quad \Pi\psi{:}\texttt{prop}.\quad \texttt{pf}\,(\phi \vee \psi) \to \Pi\chi{:}\texttt{prop}.\quad (\texttt{pf}\,\phi \to \texttt{pf}\,\chi)$
$\to (\texttt{pf}\,\psi \to \texttt{pf}\,\chi) \to \texttt{pf}\,\chi$

$\texttt{orIDemorgan}: \quad \Pi\phi{:}\texttt{prop}.\quad \Pi\psi{:}\texttt{prop}.\quad (\texttt{pf}\,(\neg\phi) \to \texttt{pf}\,(\neg\psi) \to \texttt{pf}\,\bot) \to \texttt{pf}\,(\phi \vee \psi)$

$\texttt{exE}: \quad \Pi\phi{:}(\texttt{set} \to \texttt{prop}).\quad \texttt{pf}\,(\exists x.\,\phi\,x) \to \Pi\psi{:}\texttt{prop}.\quad (\Pi x{:}\texttt{set}.\quad \texttt{pf}\,(\phi\,x) \to \texttt{pf}\,\psi)$
$\to \texttt{pf}\,\psi$

$\texttt{exI}: \quad \Pi\phi{:}(\texttt{set} \to \texttt{prop}).\quad \Pi x{:}\texttt{set}.\quad \texttt{pf}\,(\phi\,x) \to \texttt{pf}\,(\exists y.\,\phi\,y)$

$\texttt{equivI}: \quad \Pi\phi{:}\texttt{prop}.\quad \Pi\psi{:}\texttt{prop}.\quad (\texttt{pf}\,\phi \to \texttt{pf}\,\psi) \to (\texttt{pf}\,\psi \to \texttt{pf}\,\phi) \to \texttt{pf}\,(\phi \equiv \psi)$

$\texttt{equivE}: \quad \Pi\phi{:}\texttt{prop}.\quad \Pi\psi{:}\texttt{prop}.\quad \Pi\chi{:}\texttt{prop}.\quad \texttt{pf}\,(\phi \equiv \psi) \to (\texttt{pf}\,\phi \to \texttt{pf}\,\psi \to \texttt{pf}\,\chi)$
$\to (\texttt{pf}\,(\neg\phi) \to \texttt{pf}\,(\neg\psi) \to \texttt{pf}\,\chi) \to \texttt{pf}\,\chi$

We derive natural deduction rules corresponding to the set theory axioms, as well as reflexivity of equality: `eqI`. We define `true` and give an introduction rule `trueI` as well. We use $\top$ to denote the term `true`.

$\texttt{setext}: \quad \Pi A{:}\texttt{set}.\quad \Pi B{:}\texttt{set}.\quad (\Pi x{:}\texttt{set}.\quad \texttt{pf}\,(x \in A) \to \texttt{pf}\,(x \in B)) \to$
$(\Pi x{:}\texttt{set}.\quad \texttt{pf}\,(x \in B) \to \texttt{pf}\,(x \in A)) \to \texttt{pf}\,(A = B)$

$\texttt{eqI}: \quad \Pi A{:}\texttt{set}.\quad \texttt{pf}\,(A = A)$

$\texttt{setadjoinIL}: \quad \Pi x{:}\texttt{set}.\quad \Pi A{:}\texttt{set}.\quad \texttt{pf}\,(x \in (\texttt{setadjoin}\,x\,A))$

$\texttt{setadjoinIR}: \quad \Pi x{:}\texttt{set}.\quad \Pi A{:}\texttt{set}.\quad \Pi y{:}\texttt{set}.\quad \texttt{pf}\,(y \in A) \to \texttt{pf}\,(y \in (\texttt{setadjoin}\,x\,A))$

$\texttt{setadjoinE}: \quad \Pi x{:}\texttt{set}.\quad \Pi A{:}\texttt{set}.\quad \Pi y{:}\texttt{set}.\quad \texttt{pf}\,(y \in (\texttt{setadjoin}\,x\,A)) \to \Pi\phi{:}\texttt{prop}.$
$(\texttt{pf}\,(y = x) \to \texttt{pf}\,\phi) \to (\texttt{pf}\,(y \in A) \to \texttt{pf}\,\phi) \to \texttt{pf}\,\phi$

$\texttt{true}: \quad \texttt{prop} = \emptyset \in \{\emptyset\}$

$\texttt{trueI}: \quad \texttt{pf}\,\top$

$\texttt{powersetI}: \quad \Pi A{:}\texttt{set}.\Pi X{:}\texttt{set}.(\Pi x{:}\texttt{set}.\texttt{pf}\,(x \in X) \to \texttt{pf}\,(x \in A)) \to \texttt{pf}\,(X \in (\mathcal{P}\,A))$

$\texttt{powersetE}: \quad \Pi A{:}\texttt{set}.\Pi X{:}\texttt{set}.\Pi x{:}\texttt{set}.\texttt{pf}\,(X \in (\mathcal{P}\,A)) \to \texttt{pf}\,(x \in X) \to \texttt{pf}\,(x \in A)$

$\texttt{setunionI}: \quad \Pi A{:}\texttt{set}.\Pi x{:}\texttt{set}.\Pi X{:}\texttt{set}.\texttt{pf}\,(x \in X) \to \texttt{pf}\,(X \in A) \to \texttt{pf}\,(x \in (\bigcup A))$

$\texttt{setunionE}: \quad \Pi A{:}\texttt{set}.\quad \Pi x{:}\texttt{set}.\quad \texttt{pf}\,(x \in (\bigcup A)) \to \Pi\phi{:}\texttt{prop}.$
$(\Pi X{:}\texttt{set}.\quad \texttt{pf}\,(x \in X) \to \texttt{pf}\,(X \in A) \to \texttt{pf}\,\phi) \to \texttt{pf}\,\phi$

Equality is symmetric and transitive. We also include a few variations on equality rules.

$\texttt{symeq}: \quad \Pi x{:}\texttt{set}. \quad \Pi y{:}\texttt{set}. \quad \texttt{pf}\,(x = y) \;\rightarrow\; \texttt{pf}\,(y = x)$

$\texttt{transeq}: \quad \Pi x{:}\texttt{set}. \quad \Pi y{:}\texttt{set}. \quad \Pi z{:}\texttt{set}. \quad \texttt{pf}\,(x = y) \;\rightarrow\; \texttt{pf}\,(y = z) \;\rightarrow\; \texttt{pf}\,(x = z)$

$\texttt{symtrans1eq}: \quad \Pi x{:}\texttt{set}. \quad \Pi y{:}\texttt{set}. \quad \Pi z{:}\texttt{set}. \quad \texttt{pf}\,(x = y) \;\rightarrow\; \texttt{pf}\,(z = y) \;\rightarrow\; \texttt{pf}\,(x = z)$

$\texttt{eqE2}: \quad \Pi x{:}\texttt{set}. \quad \Pi y{:}\texttt{set}. \quad \Pi \phi{:}(\texttt{set} \rightarrow \texttt{prop}). \quad \texttt{pf}\,(x = y) \;\rightarrow\; \texttt{pf}\,(\phi\, y) \;\rightarrow\; \texttt{pf}\,(\phi\, x)$

$\texttt{uniqinunit}: \quad \Pi x{:}\texttt{set}. \quad \Pi y{:}\texttt{set}. \quad \texttt{pf}\,(x \in \{y\}) \;\rightarrow\; \texttt{pf}\,(x = y)$

$\texttt{eqinunit}: \quad \Pi x{:}\texttt{set}. \quad \Pi y{:}\texttt{set}. \quad \texttt{pf}\,(x = y) \;\rightarrow\; \texttt{pf}\,(x \in \{y\})$

$\texttt{symtrans2eq}: \quad \Pi x{:}\texttt{set}. \quad \Pi y{:}\texttt{set}. \quad \Pi z{:}\texttt{set}. \quad \texttt{pf}\,(x = y) \;\rightarrow\; \texttt{pf}\,(z = y) \;\rightarrow\; \texttt{pf}\,(z = x)$

$\texttt{boxeq}: \quad \Pi x^1{:}\texttt{set}. \quad \Pi y^1{:}\texttt{set}. \quad \Pi x^2{:}\texttt{set}. \quad \Pi y^2{:}\texttt{set}. \quad \texttt{pf}\,(x^1 = y^1) \;\rightarrow\; \texttt{pf}\,(x^2 = y^2)$
$\rightarrow\; \texttt{pf}\,(y^1 = y^2) \;\rightarrow\; \texttt{pf}\,(x^1 = x^2)$

The following are rules for dependent quantifiers and dependent propositional connectives.

$\texttt{dallI}: \quad \Pi A{:}\texttt{set}. \quad \Pi \phi{:}(A \rightarrow \texttt{prop}). \quad (\Pi a{:}A. \quad \texttt{pf}\,(\phi\, a)) \;\rightarrow\; \texttt{pf}\,(\forall a \in A.\, \phi\, a)$

$\texttt{dallE}: \quad \Pi A{:}\texttt{set}. \quad \Pi \phi{:}(A \rightarrow \texttt{prop}). \quad \texttt{pf}\,(\forall a \in A.\, \phi\, a) \;\rightarrow\; \Pi a{:}A. \quad \texttt{pf}\,(\phi\, a)$

$\texttt{dexI}: \quad \Pi A{:}\texttt{set}. \quad \Pi \phi{:}(A \rightarrow \texttt{prop}). \quad \Pi a{:}A. \quad \texttt{pf}\,(\phi\, a) \;\rightarrow\; \texttt{pf}\,(\exists b \in A.\, \phi\, b)$

$\texttt{dexE}: \quad \Pi A{:}\texttt{set}. \quad \Pi \phi{:}(A \rightarrow \texttt{prop}). \quad \texttt{pf}\,(\exists a \in A.\, \phi\, a) \;\rightarrow\; \Pi \psi{:}\texttt{prop}\,.$
$(\Pi a{:}A. \quad \texttt{pf}\,(\phi\, a) \;\rightarrow\; \texttt{pf}\,\psi) \;\rightarrow\; \texttt{pf}\,\psi$

$\texttt{dandI}: \quad \Pi \phi{:}\texttt{prop}. \quad \Pi \psi{:}(\texttt{pf}\,\phi \;\rightarrow\; \texttt{prop}). \quad \Pi \mathcal{D}{:}\texttt{pf}\,\phi. \quad \texttt{pf}\,(\psi\, \mathcal{D}) \;\rightarrow\; \texttt{pf}\,(\texttt{dand}\,\phi\,\psi)$

$\texttt{dandEL}: \quad \Pi \phi{:}\texttt{prop}. \quad \Pi \psi{:}(\texttt{pf}\,\phi \;\rightarrow\; \texttt{prop}). \quad \texttt{pf}\,(\texttt{dand}\,\phi\,\psi) \;\rightarrow\; \texttt{pf}\,\phi$

$\texttt{dandER}: \quad \Pi \phi{:}\texttt{prop}. \quad \Pi \psi{:}(\texttt{pf}\,\phi \;\rightarrow\; \texttt{prop}). \quad \Pi \mathcal{D}{:}\texttt{pf}\,(\texttt{dand}\,\phi\,\psi). \quad \texttt{pf}\,(\psi\, \mathbf{L1})$
using proof step
$\mathbf{L1}: \quad \texttt{pf}\,\phi$ by: $\texttt{dandEL}\,\phi\,\psi\,\mathcal{D}$

$\texttt{dimpI}: \quad \Pi \phi{:}\texttt{prop}. \quad \Pi \psi{:}(\texttt{pf}\,\phi \;\rightarrow\; \texttt{prop}). \quad (\Pi \mathcal{D}{:}\texttt{pf}\,\phi. \quad \texttt{pf}\,(\psi\, \mathcal{D})) \;\rightarrow\; \texttt{pf}\,(\texttt{dimp}\,\phi\,\psi)$

$\texttt{dimpE}: \quad \Pi \phi{:}\texttt{prop}. \quad \Pi \psi{:}(\texttt{pf}\,\phi \;\rightarrow\; \texttt{prop}). \quad \texttt{pf}\,(\texttt{dimp}\,\phi\,\psi) \;\rightarrow\; \Pi \mathcal{D}{:}\texttt{pf}\,\phi. \quad \texttt{pf}\,(\psi\, \mathcal{D})$

The following rules are useful for special cases, such as when a negated proposition is assumed. Some of the rules are classical in nature.

$\texttt{vacuousImpI}: \quad \Pi \phi{:}\texttt{prop}. \quad \Pi \psi{:}\texttt{prop}. \quad \texttt{pf}\,(\neg\phi) \;\rightarrow\; \texttt{pf}\,(\phi \;\supset\; \psi)$

$\texttt{trivialImpI}: \quad \Pi \phi{:}\texttt{prop}. \quad \Pi \psi{:}\texttt{prop}. \quad \texttt{pf}\,\psi \;\rightarrow\; \texttt{pf}\,(\phi \;\supset\; \psi)$

$\texttt{excludedmiddle}: \quad \Pi \phi{:}\texttt{prop}. \quad \texttt{pf}\,(\phi \;\vee\; (\neg\phi))$

$\texttt{notimpE}: \quad \Pi \phi{:}\texttt{prop}. \quad \Pi \psi{:}\texttt{prop}. \quad \texttt{pf}\,(\neg(\phi \;\supset\; \psi)) \;\rightarrow\; \texttt{pf}\,(\phi \;\wedge\; (\neg\psi))$

$\texttt{notimpE1} : \quad \Pi\phi{:}\texttt{prop}. \quad \Pi\psi{:}\texttt{prop}. \quad \texttt{pf}\,(\neg(\phi \supset \psi)) \rightarrow \texttt{pf}\,\phi$

$\texttt{notimpE2} : \quad \Pi\phi{:}\texttt{prop}. \quad \Pi\psi{:}\texttt{prop}. \quad \texttt{pf}\,(\neg(\phi \supset \psi)) \rightarrow \texttt{pf}\,(\neg\psi)$

$\texttt{notorE} : \quad \Pi\phi{:}\texttt{prop}. \quad \Pi\psi{:}\texttt{prop}. \quad \texttt{pf}\,(\neg(\phi \vee \psi)) \rightarrow \texttt{pf}\,((\neg\phi) \wedge (\neg\psi))$

$\texttt{notorE1} : \quad \Pi\phi{:}\texttt{prop}. \quad \Pi\psi{:}\texttt{prop}. \quad \texttt{pf}\,(\neg(\phi \vee \psi)) \rightarrow \texttt{pf}\,(\neg\phi)$

$\texttt{notorE2} : \quad \Pi\phi{:}\texttt{prop}. \quad \Pi\psi{:}\texttt{prop}. \quad \texttt{pf}\,(\neg(\phi \vee \psi)) \rightarrow \texttt{pf}\,(\neg\psi)$

$\texttt{notandE} : \quad \Pi\phi{:}\texttt{prop}. \quad \Pi\psi{:}\texttt{prop}. \quad \texttt{pf}\,(\neg(\phi \wedge \psi)) \rightarrow \texttt{pf}\,((\neg\phi) \vee (\neg\psi))$

$\texttt{notexE} : \quad \Pi\phi{:}(\texttt{set} \rightarrow \texttt{prop}). \quad \texttt{pf}\,(\neg\exists A.\,\phi\,A) \rightarrow \texttt{pf}\,(\forall A.\,\neg\phi\,A)$

$\texttt{notdexE} : \quad \Pi A{:}\texttt{set}.\Pi\phi{:}(A \rightarrow \texttt{prop}).\texttt{pf}\,(\neg\exists a \in A.\,\phi\,a) \rightarrow \texttt{pf}\,(\forall a \in A.\,\neg\phi\,a)$

$\texttt{notallE} : \quad \Pi\phi{:}(\texttt{set} \rightarrow \texttt{prop}). \quad \texttt{pf}\,(\neg\forall A.\,\phi\,A) \rightarrow \texttt{pf}\,(\exists A.\,\neg\phi\,A)$

$\texttt{notdallE} : \quad \Pi A{:}\texttt{set}. \quad \Pi\phi{:}(A \rightarrow \texttt{prop}).\texttt{pf}\,(\neg\forall a \in A.\,\phi\,a) \rightarrow \texttt{pf}\,(\exists a \in A.\,\neg\phi\,a)$

We now give more rules for equivalence.

$\texttt{reflequiv} : \quad \Pi\phi{:}\texttt{prop}. \quad \texttt{pf}\,(\phi \equiv \phi)$

$\texttt{symequiv} : \quad \Pi\phi{:}\texttt{prop}. \quad \Pi\psi{:}\texttt{prop}. \quad \texttt{pf}\,(\phi \equiv \psi) \rightarrow \texttt{pf}\,(\psi \equiv \phi)$

$\texttt{transequiv} : \quad \Pi\phi{:}\texttt{prop}. \quad \Pi\psi{:}\texttt{prop}. \quad \Pi\chi{:}\texttt{prop}. \quad \texttt{pf}\,(\phi \equiv \psi) \rightarrow \texttt{pf}\,(\psi \equiv \chi)$
$\rightarrow \texttt{pf}\,(\phi \equiv \chi)$

$\texttt{symtrans1equiv} : \quad \Pi\phi{:}\texttt{prop}. \quad \Pi\psi{:}\texttt{prop}. \quad \Pi\chi{:}\texttt{prop}. \quad \texttt{pf}\,(\phi \equiv \psi) \rightarrow \texttt{pf}\,(\chi \equiv \psi)$
$\rightarrow \texttt{pf}\,(\phi \equiv \chi)$

$\texttt{boxequiv} : \quad \Pi\phi^1{:}\texttt{prop}. \quad \Pi\psi^1{:}\texttt{prop}. \quad \Pi\phi^2{:}\texttt{prop}. \quad \Pi\psi^2{:}\texttt{prop}. \quad \texttt{pf}\,(\phi^1 \equiv \psi^1)$
$\rightarrow \texttt{pf}\,(\phi^2 \equiv \psi^2) \rightarrow \texttt{pf}\,(\psi^1 \equiv \psi^2) \rightarrow \texttt{pf}\,(\phi^1 \equiv \phi^2)$

The following are modified versions of the rule corresponding to the `eqE` logical constant, which corresponds to the Leibniz property of equality. The version `eqCE` here (and the symmetric version `eqCE2`) allow one to replace equal terms of a *class type* (as opposed to type `set`) within a term. Note that the types of a and b below (written as ϕ) are $\texttt{cl}\,\phi$ and the type of ψ (written as $\phi \rightarrow \texttt{prop}$) is $\texttt{cl}\,\phi \rightarrow \texttt{prop}$. (We conjecture that it would not be possible to define a term of the types of `eqCE` or `eqCE2` using the signature of logical constants used in DeTSeT if proof irrelevance were not part of the type theory. That is, proof irrelevance seems to be vital for reducing "typed Leibniz equality" in the form of `eqCE` to "untyped Leibniz equality" in the form of `eqE`.)

$\texttt{eqCE} : \quad \Pi\phi{:}(\texttt{set} \rightarrow \texttt{prop}). \quad \Pi a{:}\phi. \quad \Pi b{:}\phi. \quad \Pi\psi{:}(\phi \rightarrow \texttt{prop}).\,\texttt{pf}\,(a = b)$
$\rightarrow \texttt{pf}\,(\psi\,a) \rightarrow \texttt{pf}\,(\psi\,b)$

$\texttt{eqCE2} : \quad \Pi\phi{:}(\texttt{set} \rightarrow \texttt{prop}). \quad \Pi a{:}\phi. \quad \Pi b{:}\phi. \quad \Pi\psi{:}(\phi \rightarrow \texttt{prop}). \quad \texttt{pf}\,(a = b)$
$\rightarrow \texttt{pf}\,(\psi\,b) \rightarrow \texttt{pf}\,(\psi\,a)$

The following is a definition of the subset relation:

$\texttt{subset}: \quad \texttt{set} \to \texttt{set} \to \texttt{prop} = \lambda A\, \lambda B\, \forall a \in A.\, a \in B$

We use $M \subseteq N$ to denote a term of the form $(\texttt{subset}\, M\, N)$. We next give rules relating to subset and powerset.

$\texttt{subsetI1}: \quad \Pi A{:}\texttt{set}.\ \ \Pi B{:}\texttt{set}.\ \ (\Pi a{:}A.\ \ \texttt{pf}\,(a \in B)) \to \texttt{pf}\,(A \subseteq B)$

$\texttt{subsetI2}: \quad \Pi A{:}\texttt{set}.\Pi B{:}\texttt{set}.(\Pi x{:}\texttt{set}.\texttt{pf}\,(x \in A) \to \texttt{pf}\,(x \in B)) \to \texttt{pf}\,(A \subseteq B)$

$\texttt{subsetE}: \quad \Pi A{:}\texttt{set}.\ \ \Pi B{:}\texttt{set}.\ \ \Pi x{:}\texttt{set}.\ \ \texttt{pf}\,(A \subseteq B) \to \texttt{pf}\,(x \in A) \to \texttt{pf}\,(x \in B)$

$\texttt{subset2powerset}: \quad \Pi A{:}\texttt{set}.\ \ \Pi B{:}\texttt{set}.\ \ \texttt{pf}\,(A \subseteq B) \to \texttt{pf}\,(A \in (\mathcal{P}\, B))$

$\texttt{setextsub}: \quad \Pi A{:}\texttt{set}.\ \ \Pi B{:}\texttt{set}.\ \ \texttt{pf}\,(A \subseteq B) \to \texttt{pf}\,(B \subseteq A) \to \texttt{pf}\,(A = B)$

$\texttt{powersetI1}: \quad \Pi A{:}\texttt{set}.\ \ \Pi B{:}\texttt{set}.\ \ \texttt{pf}\,(B \subseteq A) \to \texttt{pf}\,(B \in (\mathcal{P}\, A))$

$\texttt{powersetsubset}: \quad \Pi A{:}\texttt{set}.\ \ \Pi B{:}\texttt{set}.\ \ \texttt{pf}\,(A \subseteq B) \to \texttt{pf}\,((\mathcal{P}\, A) \subseteq (\mathcal{P}\, B))$

The following defines binary union:

$\texttt{binunion}: \quad \texttt{set} \to \texttt{set} \to \texttt{set} = \lambda A\, \lambda B \bigcup \{A, B\}$

Recall $\{A, B\}$ is shorthand for the term $(\texttt{setadjoin}\, A\, (\texttt{setadjoin}\, B\, \texttt{emptyset}))$ as described in Section 2.1. From now on, we will use $A \cup B$ to denote the term $(\texttt{binunion}\, A\, B)$. The following abbreviations correspond to deduction rules for binary unions:

$\texttt{binunionIL}: \quad \Pi A{:}\texttt{set}.\ \ \Pi B{:}\texttt{set}.\ \ \Pi x{:}\texttt{set}.\ \ \texttt{pf}\,(x \in A) \to \texttt{pf}\,(x \in (A \cup B))$

$\texttt{binunionIR}: \quad \Pi A{:}\texttt{set}.\ \ \Pi B{:}\texttt{set}.\ \ \Pi x{:}\texttt{set}.\ \ \texttt{pf}\,(x \in B) \to \texttt{pf}\,(x \in (A \cup B))$

$\texttt{binunionEcases}: \quad \Pi A{:}\texttt{set}.\ \ \Pi B{:}\texttt{set}.\ \ \Pi x{:}\texttt{set}.\ \ \Pi \phi{:}\texttt{prop}.\ \ \texttt{pf}\,(x \in (A \cup B))$
$\to (\texttt{pf}\,(x \in A) \to \texttt{pf}\,\phi) \to (\texttt{pf}\,(x \in B) \to \texttt{pf}\,\phi) \to \texttt{pf}\,\phi$

$\texttt{binunionE}: \quad \Pi A{:}\texttt{set}.\Pi B{:}\texttt{set}.\Pi x{:}\texttt{set}.\texttt{pf}\,(x \in (A \cup B)) \to \texttt{pf}\,((x \in A) \vee (x \in B))$

$\texttt{binunionLsub}: \quad \Pi A{:}\texttt{set}.\ \ \Pi B{:}\texttt{set}.\ \ \texttt{pf}\,(A \subseteq (A \cup B))$

$\texttt{binunionRsub}: \quad \Pi A{:}\texttt{set}.\ \ \Pi B{:}\texttt{set}.\ \ \texttt{pf}\,(B \subseteq (A \cup B))$

We next define binary intersection and give derived rules.

$\texttt{binintersect}: \quad \texttt{set} \to \texttt{set} \to \texttt{set} = \lambda A\, \lambda B\, \{a \in A \,|\, a \in B\}$

We use $(M \cap N)$ to denote $(\texttt{binintersect}\, M\, N)$.

$\texttt{binintersectI}: \quad \Pi A{:}\texttt{set}.\Pi B{:}\texttt{set}.\Pi x{:}\texttt{set}.\texttt{pf}\,(x \in A) \to \texttt{pf}\,(x \in B)$
$\to \texttt{pf}\,(x \in (A \cap B))$

$\texttt{binintersectEL}: \quad \Pi A{:}\texttt{set}.\ \ \Pi B{:}\texttt{set}.\ \ \Pi x{:}\texttt{set}.\ \ \texttt{pf}\,(x \in (A \cap B)) \to \texttt{pf}\,(x \in A)$

$\texttt{binintersectER}: \quad \Pi A{:}\texttt{set}.\ \ \Pi B{:}\texttt{set}.\ \ \Pi x{:}\texttt{set}.\ \ \texttt{pf}\,(x \in (A \cap B)) \to \texttt{pf}\,(x \in B)$

Defining set difference is also easy.

$\texttt{setminus}: \quad \texttt{set} \to \texttt{set} \to \texttt{set} = \lambda A\,\lambda B\,\{a \in A \mid \neg a \in B\}$

We now turn to pairs, Cartesian products and functions. This is enough to interpret simply typed λ-calculus and so should provide the basis for importing the libraries of theorem provers such as Isabelle-HOL [26], HOL-light [15] and HOL4 into DeTSeT.

$\texttt{iskpair}: \quad \texttt{set} \to \texttt{prop}$
$= \lambda u\, \exists a \in \bigcup u.\, \exists b \in \bigcup u.\, u = \{\{a\}, \{a, b\}\}$

$\texttt{kpairiskpair}: \quad \Pi x{:}\texttt{set}.\quad \Pi y{:}\texttt{set}.\quad \texttt{pf}\,(\texttt{iskpair}\,\{\{x\}, \{x, y\}\})$

$\texttt{kpair}: \quad \texttt{set} \to \texttt{set} \to \texttt{iskpair} = \lambda x\,\lambda y\,\langle\,\{\{x\}, \{x, y\}\}, \texttt{kpairiskpair}\,x\,y\,\rangle$

We will use $\langle\langle M, N\rangle\rangle$ to denote the term $(\texttt{kpair}\,M\,N)$.

$\texttt{cartprod}: \quad \texttt{set} \to \texttt{set} \to \texttt{set}$

We use $A \times B$ to denote the term $(\texttt{cartprod}\,A\,B)$.

$\texttt{cartprodpairin}: \quad \Pi A{:}\texttt{set}.\quad \Pi B{:}\texttt{set}.\quad \Pi a{:}A.\quad \Pi b{:}B.\quad \texttt{pf}\,(\langle\langle\, a, b\,\rangle\rangle \in (A \times B))$

$\texttt{cartprodpair}: \quad \Pi A{:}\texttt{set}.\quad \Pi B{:}\texttt{set}.\quad A \to B \to (A \times B)$

$\texttt{cartprodfst}: \quad \Pi A{:}\texttt{set}.\quad \Pi B{:}\texttt{set}.\quad (A \times B) \to A$

$\texttt{cartprodsnd}: \quad \Pi A{:}\texttt{set}.\quad \Pi B{:}\texttt{set}.\quad (A \times B) \to B$

$\texttt{cartprodfstpairEq}: \quad \Pi A{:}\texttt{set}.\quad \Pi B{:}\texttt{set}.\quad \Pi a{:}A.\quad \Pi b{:}B\,.$
$\texttt{pf}\,(\texttt{cartprodfst}\,A\,B\,(\texttt{cartprodpair}\,A\,B\,a\,b) = a)$

$\texttt{cartprodsndpairEq}: \quad \Pi A{:}\texttt{set}.\quad \Pi B{:}\texttt{set}.\quad \Pi a{:}A.\quad \Pi b{:}B\,.$
$\texttt{pf}\,(\texttt{cartprodsnd}\,A\,B\,(\texttt{cartprodpair}\,A\,B\,a\,b) = b)$

$\texttt{cartprodpairsurjEq}: \quad \Pi A{:}\texttt{set}.\quad \Pi B{:}\texttt{set}.\quad \Pi a{:}(A \times B)\,.$
$\texttt{pf}\,(\texttt{cartprodpair}\,A\,B\,(\texttt{cartprodfst}\,A\,B\,a)\,(\texttt{cartprodsnd}\,A\,B\,a) = a)$

$\texttt{breln}: \quad \texttt{set} \to \texttt{set} \to \texttt{set} \to \texttt{prop}$

$\texttt{dpsetconstr}: \quad \Pi A{:}\texttt{set}.\quad \Pi B{:}\texttt{set}.\quad (A \to B \to \texttt{prop}) \to \texttt{set}$

We use the notation $\{\langle\langle x, y\rangle\rangle \in A \times B | \phi\}$ to denote the term $(\texttt{dpsetconstr}\,A\,B\,(\lambda x \lambda y \phi))$.

$\texttt{funcSet}: \quad \texttt{set} \to \texttt{set} \to \texttt{set}$

$\texttt{ap2}: \quad \Pi A{:}\texttt{set}.\quad \Pi B{:}\texttt{set}.\quad (\texttt{funcSet}\,A\,B) \to A \to B$

$\texttt{lam2}: \quad \Pi A{:}\texttt{set}.\quad \Pi B{:}\texttt{set}.\quad (A \to B) \to (\texttt{funcSet}\,A\,B)$

$\texttt{funcext2}: \quad \Pi A{:}\texttt{set}.\quad \Pi B{:}\texttt{set}.\quad \Pi f{:}(\texttt{funcSet}\,A\,B).\quad \Pi g{:}(\texttt{funcSet}\,A\,B)\,.$
$(\Pi a{:}A.\quad \texttt{pf}\,(\texttt{ap2}\,A\,B\,f\,a = \texttt{ap2}\,A\,B\,g\,a)) \to \texttt{pf}\,(f = g)$

$\texttt{beta2}: \quad \Pi A{:}\texttt{set}.\Pi B{:}\texttt{set}.\Pi f{:}(A \to B).\Pi a{:}A.$
$\texttt{pf}\,((\texttt{ap2}\,A\,B\,(\texttt{lam2}\,A\,B\,f)\,a) = (f\,a))$

$\texttt{eta2}: \quad \Pi A\!:\!\texttt{set}. \quad \Pi B\!:\!\texttt{set}. \quad \Pi f\!:\!(\texttt{funcSet}\,A\,B). \quad \texttt{pf}\,((\texttt{lam2}\,A\,B\,(\texttt{ap2}\,A\,B\,f)) = f)$

An **if** operator is useful for defining objects or functions that depend on conditions.

$\texttt{if}: \quad \Pi A\!:\!\texttt{set}. \quad \texttt{prop} \to A \to A \to A$

$\texttt{iftrue}: \quad \Pi A\!:\!\texttt{set}. \quad \Pi \phi\!:\!\texttt{prop}. \quad \Pi a\!:\!A. \quad \Pi b\!:\!A. \quad \texttt{pf}\,\phi \to \texttt{pf}\,(\texttt{if}\,A\,\phi\,a\,b = a)$

$\texttt{iffalse}: \quad \Pi A\!:\!\texttt{set}. \quad \Pi \phi\!:\!\texttt{prop}. \quad \Pi a\!:\!A. \quad \Pi b\!:\!A. \quad \texttt{pf}\,(\neg\phi) \to \texttt{pf}\,(\texttt{if}\,A\,\phi\,a\,b = b)$

$\texttt{iftrueorfalse}: \quad \Pi A\!:\!\texttt{set}. \quad \Pi \phi\!:\!\texttt{prop}. \quad \Pi a\!:\!A. \quad \Pi b\!:\!A. \quad \texttt{pf}\,(\texttt{if}\,A\,\phi\,a\,b \in \{a, b\})$

4 The Distributivity Example

An example considered several times within the Ωmega group is the distributivity property of binary union and intersection $A \cap (B \cup C)) = (A \cap B) \cup (A \cap C)$. (This is part of Theorem 1.1.4(d) in [4]. The proof in [4] was studied in [1, 9].) We formalize the problem as the type of the abbreviation `bs114d` and give a proof as a term of this type in Figure 3.

5 Wizard of Oz DiaLog Examples

5.1 Formalizing the First DiaLog Wizard of Oz Experiment: Typed Sets

The following abbreviations concern typed sets and are relevant to the first Wizard of Oz experiment in the DiaLog project [5]. We begin by defining "typed" versions of set operations such as binary intersection, powerset, and complement *relative* to a set U. That is, we define the operations *restricted* to subsets of a given set. Since the power set of U is the set of subsets of U, we can use this set as the class type of subsets of U. First, we can prove that if X and Y are subsets of U, then so is $X \cap Y$:

$$\texttt{binintersectT\_lem} : \quad \Pi U\!:\!\texttt{set}. \quad \Pi A\!:\!(\mathcal{P}\,U). \quad \Pi B\!:\!(\mathcal{P}\,U). \quad \texttt{pf}\,((A \cap B) \in (\mathcal{P}\,U))$$

Next we can *use* this fact to define the "typed" binary intersection operator which carries the information that the inputs and outputs are subsets of U:

$$\texttt{binintersectT} : \quad \Pi U\!:\!\texttt{set}. \quad (\mathcal{P}\,U) \to (\mathcal{P}\,U) \to (\mathcal{P}\,U)$$
$$= \lambda U\,\lambda A\,\lambda B\,\langle\, A \cap B, \texttt{binintersectT\_lem}\,U\,A\,B\,\rangle$$

We use $A \hat{\cap} B$ to denote the term $(\texttt{binintersectT}\,U\,A\,B)$ (where U is a fixed bound variable). We could overload the symbol $\cap$ to be typed intersection if the two arguments have type $\texttt{cl}\,(\texttt{in}\,U)$ and untyped intersection otherwise, but we choose to avoid overloading symbols for the sake of clarity.

A similar process defines a "typed" binary union operator.

$$\texttt{binunionT\_lem} : \quad \Pi U\!:\!\texttt{set}. \quad \Pi A\!:\!(\mathcal{P}\,U). \quad \Pi B\!:\!(\mathcal{P}\,U). \quad \texttt{pf}\,((A \cup B) \in (\mathcal{P}\,U))$$

$$\texttt{binunionT} : \quad \Pi U\!:\!\texttt{set}. \quad (\mathcal{P}\,U) \to (\mathcal{P}\,U) \to (\mathcal{P}\,U)$$
$$= \lambda U\,\lambda A\,\lambda B\,\langle\, A \cup B, \texttt{binunionT\_lem}\,U\,A\,B\,\rangle$$

We use $A \hat{\cup} B$ to denote the term $(\texttt{binunionT}\,U\,A\,B)$.

We define a "typed" powerset operator (note that the output type is one "untyped" powerset higher than the input type).

$$\texttt{powersetT\_lem} : \quad \Pi U\!:\!\texttt{set}. \quad \Pi A\!:\!(\mathcal{P}\,U). \quad \texttt{pf}\,((\mathcal{P}\,A) \in (\mathcal{P}\,\mathcal{P}\,U))$$

$$\texttt{powersetT} : \quad \Pi U\!:\!\texttt{set}. \quad (\mathcal{P}\,U) \to (\mathcal{P}\,\mathcal{P}\,U)$$
$$= \lambda U\,\lambda A\,\langle\, \mathcal{P}\,A, \texttt{powersetT\_lem}\,U\,A\,\rangle$$

We use $\hat{\mathcal{P}} A$ to denote the term $(\texttt{powersetT}\,U\,A)$.

$\texttt{bs114d}: \quad \Pi A\!:\!\texttt{set}.\Pi B\!:\!\texttt{set}.\Pi C\!:\!\texttt{set}.\texttt{pf}\,((A \cap (B \cup C)) = ((A \cap B) \cup (A \cap C)))$
$= \lambda A\, \lambda B\, \lambda C\, \texttt{setextsub}\,(A \cap (B \cup C))\,((A \cap B) \cup (A \cap C))\,\mathbf{L10}\,\mathbf{L21}$
using proof steps

Let a have type $(A \cap (B \cup C))$

L1 : $\texttt{pf}\,(a \in (A \cap (B \cup C)))$ by: a
L2 : $\texttt{pf}\,(a \in (B \cup C))$ by: $\texttt{binintersectER}\,A\,(B \cup C)\,a\,\mathbf{L1}$
$\mathcal{D}$ Assume $a \in B$
L3 : $\texttt{pf}\,(a \in A)$ by: $\texttt{binintersectEL}\,A\,(B \cup C)\,a\,\mathbf{L1}$
L4 : $\texttt{pf}\,(a \in (A \cap B))$ by: $\texttt{binintersectI}\,A\,B\,a\,\mathbf{L3}\,\mathcal{D}$
L5 : $\texttt{pf}\,(a \in ((A \cap B) \cup (A \cap C)))$ by: $\texttt{binunionIL}\,(A \cap B)\,(A \cap C)\,a\,\mathbf{L4}$
$\mathcal{E}$ Assume $a \in C$
L6 : $\texttt{pf}\,(a \in A)$ by: $\texttt{binintersectEL}\,A\,(B \cup C)\,a\,\mathbf{L1}$
L7 : $\texttt{pf}\,(a \in (A \cap C))$ by: $\texttt{binintersectI}\,A\,C\,a\,\mathbf{L6}\,\mathcal{E}$
L8 : $\texttt{pf}\,(a \in ((A \cap B) \cup (A \cap C)))$ by: $\texttt{binunionIR}\,(A \cap B)\,(A \cap C)\,a\,\mathbf{L7}$
L9 : $\texttt{pf}\,(a \in ((A \cap B) \cup (A \cap C)))$
by: $\texttt{binunionEcases}\,B\,C\,a\,(a \in ((A \cap B) \cup (A \cap C)))\,\mathbf{L2}\,(\lambda \mathcal{D}\,\mathbf{L5})\,(\lambda \mathcal{E}\,\mathbf{L8})$
L10 : $\texttt{pf}\,((A \cap (B \cup C)) \subseteq ((A \cap B) \cup (A \cap C)))$
by: $\texttt{subsetI1}\,(A \cap (B \cup C))\,((A \cap B) \cup (A \cap C))\,(\lambda a\,\mathbf{L9})$

Let b have type $((A \cap B) \cup (A \cap C))$

L11 : $\texttt{pf}\,(b \in ((A \cap B) \cup (A \cap C)))$ by: b
$\mathcal{F}$ Assume $b \in (A \cap B)$
L12 : $\texttt{pf}\,(b \in A)$ by: $\texttt{binintersectEL}\,A\,B\,b\,\mathcal{F}$
L13 : $\texttt{pf}\,(b \in B)$ by: $\texttt{binintersectER}\,A\,B\,b\,\mathcal{F}$
L14 : $\texttt{pf}\,(b \in (B \cup C))$ by: $\texttt{binunionIL}\,B\,C\,b\,\mathbf{L13}$
L15 : $\texttt{pf}\,(b \in (A \cap (B \cup C)))$ by: $\texttt{binintersectI}\,A\,(B \cup C)\,b\,\mathbf{L12}\,\mathbf{L14}$
$\mathcal{G}$ Assume $b \in (A \cap C)$
L16 : $\texttt{pf}\,(b \in A)$ by: $\texttt{binintersectEL}\,A\,C\,b\,\mathcal{G}$
L17 : $\texttt{pf}\,(b \in C)$ by: $\texttt{binintersectER}\,A\,C\,b\,\mathcal{G}$
L18 : $\texttt{pf}\,(b \in (B \cup C))$ by: $\texttt{binunionIR}\,B\,C\,b\,\mathbf{L17}$
L19 : $\texttt{pf}\,(b \in (A \cap (B \cup C)))$ by: $\texttt{binintersectI}\,A\,(B \cup C)\,b\,\mathbf{L16}\,\mathbf{L18}$
L20 : $\texttt{pf}\,(b \in (A \cap (B \cup C)))$
by: $\texttt{binunionEcases}\,(A \cap B)\,(A \cap C)\,b\,(b \in (A \cap (B \cup C)))$
$\mathbf{L11}\,(\lambda \mathcal{F}\,\mathbf{L15})\,(\lambda \mathcal{G}\,\mathbf{L19})$
L21 : $\texttt{pf}\,(((A \cap B) \cup (A \cap C)) \subseteq (A \cap (B \cup C)))$
by: $\texttt{subsetI1}\,((A \cap B) \cup (A \cap C))\,(A \cap (B \cup C))\,(\lambda b\,\mathbf{L20})$

Figure 3: Proof of Distributivity Property

We next define a *typed* version of set difference (setminusT) and a *typed* version of complement (complementT).

$\texttt{setminusT\_lem}: \quad \Pi U\!:\!\texttt{set}. \quad \Pi A\!:\!(\mathcal{P}\,U). \quad \Pi B\!:\!(\mathcal{P}\,U). \quad \texttt{pf}\,((\texttt{setminus}\,A\,B) \in (\mathcal{P}\,U))$

$\texttt{setminusT}: \quad \Pi U\!:\!\texttt{set}. \quad (\mathcal{P}\,U) \to (\mathcal{P}\,U) \to (\mathcal{P}\,U)$
$= \lambda U\,\lambda A\,\lambda B\,\langle\,\texttt{setminus}\,A\,B, \texttt{setminusT\_lem}\,U\,A\,B\,\rangle$

$\texttt{complementT\_lem}: \quad \Pi U\!:\!\texttt{set}. \quad \Pi A\!:\!(\mathcal{P}\,U). \quad \texttt{pf}\,((\texttt{setminus}\,U\,A) \in (\mathcal{P}\,U))$

$\texttt{complementT}: \quad \Pi U\!:\!\texttt{set}. \quad (\mathcal{P}\,U) \to (\mathcal{P}\,U)$
$= \lambda U\,\lambda A\,\langle\,\texttt{setminus}\,U\,A, \texttt{complementT\_lem}\,U\,A\,\rangle$

We use $\hat{\mathcal{K}}A$ to denote $(\texttt{complementT}\,U\,A)$.

We next give proof principles (omitting definitions, i.e., proof terms) for "typed sets":

$\texttt{setextT}: \quad \Pi U\!:\!\texttt{set}. \quad \Pi A\!:\!(\mathcal{P}\,U). \quad \Pi B\!:\!(\mathcal{P}\,U). \quad (\Pi x\!:\!U. \quad \texttt{pf}\,(x \in A) \to \texttt{pf}\,(x \in B))$
$\to (\Pi x\!:\!U. \quad \texttt{pf}\,(x \in B) \to \texttt{pf}\,(x \in A)) \to \texttt{pf}\,(A = B)$

$\texttt{subsetTI}: \quad \Pi U\!:\!\texttt{set}. \quad \Pi A\!:\!(\mathcal{P}\,U). \quad \Pi B\!:\!(\mathcal{P}\,U). \quad (\Pi x\!:\!U. \quad \texttt{pf}\,(x \in A) \to \texttt{pf}\,(x \in B))$
$\to \texttt{pf}\,(A \subseteq B)$

$\texttt{powersetTI}: \quad \Pi U\!:\!\texttt{set}. \quad \Pi A\!:\!(\mathcal{P}\,U). \quad \Pi B\!:\!(\mathcal{P}\,U). \quad \texttt{pf}\,(A \subseteq B) \to \texttt{pf}\,(A \in \hat{\mathcal{P}}\,B)$

$\texttt{powersetTE}: \quad \Pi U\!:\!\texttt{set}. \quad \Pi A\!:\!(\mathcal{P}\,U). \quad \Pi B\!:\!(\mathcal{P}\,U). \quad \texttt{pf}\,(A \in \hat{\mathcal{P}}\,B) \to \texttt{pf}\,(A \subseteq B)$

$\texttt{powersetTI1}: \quad \Pi U\!:\!\texttt{set}. \quad \Pi A\!:\!(\mathcal{P}\,U). \quad \Pi B\!:\!(\mathcal{P}\,U). \quad (\Pi x\!:\!U. \quad \texttt{pf}\,(x \in A)$
$\to \texttt{pf}\,(x \in B)) \to \texttt{pf}\,(A \in \hat{\mathcal{P}}\,B)$

$\texttt{powersetTE1}: \quad \Pi U\!:\!\texttt{set}. \quad \Pi A\!:\!(\mathcal{P}\,U). \quad \Pi B\!:\!(\mathcal{P}\,U). \quad \Pi x\!:\!U. \quad \texttt{pf}\,(A \in \hat{\mathcal{P}}\,B)$
$\to \texttt{pf}\,(x \in A) \to \texttt{pf}\,(x \in B)$

$\texttt{complementTI}: \quad \Pi U\!:\!\texttt{set}. \quad \Pi A\!:\!(\mathcal{P}\,U). \quad \Pi x\!:\!U. \quad \texttt{pf}\,(\neg x \in A) \to \texttt{pf}\,(x \in \hat{\mathcal{K}}\,A)$

$\texttt{complementTE}: \quad \Pi U\!:\!\texttt{set}. \quad \Pi A\!:\!(\mathcal{P}\,U). \quad \Pi x\!:\!U. \quad \texttt{pf}\,(x \in \hat{\mathcal{K}}\,A) \to \texttt{pf}\,(\neg x \in A)$

$\texttt{complementTI1}: \quad \Pi U\!:\!\texttt{set}. \quad \Pi A\!:\!(\mathcal{P}\,U). \quad \Pi x\!:\!U. \quad \texttt{pf}\,(x \in A) \to \texttt{pf}\,(\neg x \in \hat{\mathcal{K}}\,A)$

$\texttt{complementTE1}: \quad \Pi U\!:\!\texttt{set}. \quad \Pi A\!:\!(\mathcal{P}\,U). \quad \Pi x\!:\!U. \quad \texttt{pf}\,(\neg x \in \hat{\mathcal{K}}\,A) \to \texttt{pf}\,(x \in A)$

$\texttt{binintersectTEL}: \quad \Pi U\!:\!\texttt{set}. \quad \Pi A\!:\!(\mathcal{P}\,U). \quad \Pi B\!:\!(\mathcal{P}\,U). \quad \Pi x\!:\!U. \quad \texttt{pf}\,(x \in A \mathbin{\hat{\cap}} B)$
$\to \texttt{pf}\,(x \in A)$

$\texttt{binintersectTER}: \quad \Pi U\!:\!\texttt{set}. \quad \Pi A\!:\!(\mathcal{P}\,U). \quad \Pi B\!:\!(\mathcal{P}\,U). \quad \Pi x\!:\!U. \quad \texttt{pf}\,(x \in A \mathbin{\hat{\cap}} B)$
$\to \texttt{pf}\,(x \in B)$

$\texttt{binintersectTI}: \quad \Pi U\!:\!\texttt{set}. \quad \Pi A\!:\!(\mathcal{P}\,U). \quad \Pi B\!:\!(\mathcal{P}\,U). \quad \Pi x\!:\!U.$
$\texttt{pf}\,(x \in A) \to \texttt{pf}\,(x \in B) \to \texttt{pf}\,(x \in A \mathbin{\hat{\cap}} B)$

$\texttt{binintersectTELcontra}: \quad \Pi U\!:\!\texttt{set}. \quad \Pi A\!:\!(\mathcal{P}\,U). \quad \Pi B\!:\!(\mathcal{P}\,U). \quad \Pi x\!:\!U.$
$\texttt{pf}\,(\neg x \in A) \to \texttt{pf}\,(\neg x \in A \mathbin{\hat{\cap}} B)$

$\texttt{binintersectTERcontra}: \quad \Pi U{:}\texttt{set}. \quad \Pi A{:}(\mathcal{P}\,U). \quad \Pi B{:}(\mathcal{P}\,U). \quad \Pi x{:}U\,.$
$\texttt{pf}\,(\neg x \in B) \rightarrow \texttt{pf}\,(\neg x \in A \hat{\cap} B)$

$\texttt{contrasubsetT}: \quad \Pi U{:}\texttt{set}. \quad \Pi A{:}(\mathcal{P}\,U). \quad \Pi B{:}(\mathcal{P}\,U). \quad \Pi x{:}U. \quad \texttt{pf}\,(A \subseteq \hat{\mathcal{K}}\,B)$
$\rightarrow \texttt{pf}\,(x \in B) \rightarrow \texttt{pf}\,(\neg x \in A)$

$\texttt{contrasubsetT1}: \quad \Pi U{:}\texttt{set}. \quad \Pi A{:}(\mathcal{P}\,U). \quad \Pi B{:}(\mathcal{P}\,U). \quad \Pi x{:}U. \quad \texttt{pf}\,(A \subseteq B) \rightarrow$
$\texttt{pf}\,(\neg x \in B) \rightarrow \texttt{pf}\,(\neg x \in A)$

$\texttt{contrasubsetT2}: \quad \Pi U{:}\texttt{set}. \quad \Pi A{:}(\mathcal{P}\,U). \quad \Pi B{:}(\mathcal{P}\,U). \quad \texttt{pf}\,(A \subseteq B) \rightarrow \texttt{pf}\,(\hat{\mathcal{K}}\,B \subseteq \hat{\mathcal{K}}\,A)$

$\texttt{contrasubsetT3}: \quad \Pi U{:}\texttt{set}. \quad \Pi A{:}(\mathcal{P}\,U). \quad \Pi B{:}(\mathcal{P}\,U). \quad \texttt{pf}\,(\hat{\mathcal{K}}\,B \subseteq \hat{\mathcal{K}}\,A) \rightarrow \texttt{pf}\,(A \subseteq B)$

$\texttt{doubleComplementI1}: \quad \Pi U{:}\texttt{set}. \quad \Pi A{:}(\mathcal{P}\,U). \quad \Pi x{:}U. \quad \texttt{pf}\,(x \in A) \rightarrow \texttt{pf}\,(x \in \hat{\mathcal{K}}\,(\hat{\mathcal{K}}\,A))$

$\texttt{doubleComplementE1}: \quad \Pi U{:}\texttt{set}. \quad \Pi A{:}(\mathcal{P}\,U). \quad \Pi x{:}U. \quad \texttt{pf}\,(x \in \hat{\mathcal{K}}\,(\hat{\mathcal{K}}\,A)) \rightarrow \texttt{pf}\,(x \in A)$

$\texttt{doubleComplementSub1}: \quad \Pi U{:}\texttt{set}. \quad \Pi A{:}(\mathcal{P}\,U). \quad \texttt{pf}\,(A \subseteq \hat{\mathcal{K}}\,(\hat{\mathcal{K}}\,A))$

$\texttt{doubleComplementSub2}: \quad \Pi U{:}\texttt{set}. \quad \Pi A{:}(\mathcal{P}\,U). \quad \texttt{pf}\,(\hat{\mathcal{K}}\,(\hat{\mathcal{K}}\,A) \subseteq A)$

$\texttt{doubleComplementEq}: \quad \Pi U{:}\texttt{set}. \quad \Pi A{:}(\mathcal{P}\,U). \quad \texttt{pf}\,(A = \hat{\mathcal{K}}\,(\hat{\mathcal{K}}\,A))$

$\texttt{binintersectTSub1}: \quad \Pi U{:}\texttt{set}. \quad \Pi A{:}(\mathcal{P}\,U). \quad \Pi B{:}(\mathcal{P}\,U). \quad \texttt{pf}\,(A \hat{\cap} B \subseteq A)$

$\texttt{binintersectTSub2}: \quad \Pi U{:}\texttt{set}. \quad \Pi A{:}(\mathcal{P}\,U). \quad \Pi B{:}(\mathcal{P}\,U). \quad \texttt{pf}\,(A \hat{\cap} B \subseteq B)$

$\texttt{complementTnotintersectT}: \quad \Pi U{:}\texttt{set}. \quad \Pi A{:}(\mathcal{P}\,U). \quad \Pi B{:}(\mathcal{P}\,U). \quad \Pi x{:}U\,.$
$\texttt{pf}\,(x \in \hat{\mathcal{K}}\,A) \rightarrow \texttt{pf}\,(\neg x \in A \hat{\cap} B)$

$\texttt{complementImpComplementIntersect}: \quad \Pi U{:}\texttt{set}. \quad \Pi A{:}(\mathcal{P}\,U). \quad \Pi B{:}(\mathcal{P}\,U). \quad \Pi x{:}U\,.$
$\texttt{pf}\,(x \in \hat{\mathcal{K}}\,A) \rightarrow \texttt{pf}\,(x \in \hat{\mathcal{K}}\,(A \hat{\cap} B))$

$\texttt{complementSubsetComplementIntersect}: \quad \Pi U{:}\texttt{set}. \quad \Pi A{:}(\mathcal{P}\,U). \quad \Pi B{:}(\mathcal{P}\,U)\,.$
$\texttt{pf}\,(\hat{\mathcal{K}}\,A \subseteq \hat{\mathcal{K}}\,(A \hat{\cap} B))$

$\texttt{complementInPowersetComplementIntersect}: \quad \Pi U{:}\texttt{set}. \quad \Pi A{:}(\mathcal{P}\,U). \quad \Pi B{:}(\mathcal{P}\,U)\,.$
$\texttt{pf}\,(\hat{\mathcal{K}}\,A \in \hat{\mathcal{P}}\,(\hat{\mathcal{K}}\,(A \hat{\cap} B)))$

$\texttt{contraSubsetComplement}: \quad \Pi U{:}\texttt{set}. \quad \Pi A{:}(\mathcal{P}\,U). \quad \Pi B{:}(\mathcal{P}\,U)\,.$
$\texttt{pf}\,(A \subseteq \hat{\mathcal{K}}\,B) \rightarrow \Pi x{:}U. \quad \texttt{pf}\,(x \in B) \rightarrow \texttt{pf}\,(x \in \hat{\mathcal{K}}\,A)$

$\texttt{complementTcontraSubset}: \quad \Pi U{:}\texttt{set}.\Pi A{:}(\mathcal{P}\,U).\Pi B{:}(\mathcal{P}\,U).\texttt{pf}\,(A \subseteq \hat{\mathcal{K}}\,B)$
$\rightarrow \texttt{pf}\,(B \subseteq \hat{\mathcal{K}}\,A)$

$\texttt{binunionTIL}: \quad \Pi U{:}\texttt{set}. \quad \Pi A{:}(\mathcal{P}\,U). \quad \Pi B{:}(\mathcal{P}\,U). \quad \Pi x{:}U. \quad \texttt{pf}\,(x \in A)$
$\rightarrow \texttt{pf}\,(x \in A \hat{\cup} B)$

$\texttt{binunionTIR}: \quad \Pi U{:}\texttt{set}. \quad \Pi A{:}(\mathcal{P}\,U). \quad \Pi B{:}(\mathcal{P}\,U). \quad \Pi x{:}U. \quad \texttt{pf}\,(x \in B)$
$\rightarrow \texttt{pf}\,(x \in A \hat{\cup} B)$

$\texttt{binunionTILcontra}: \quad \Pi U{:}\texttt{set}.\Pi A{:}(\mathcal{P}\,U).\Pi B{:}(\mathcal{P}\,U).\Pi x{:}U.\texttt{pf}\,(\neg x \in A\hat{\cup}B)$
$\rightarrow \texttt{pf}\,(\neg x \in A)$

$\texttt{binunionTIRcontra}:\quad \Pi U\!:\!\texttt{set}.\Pi A\!:\!(\mathcal{P}\,U).\Pi B\!:\!(\mathcal{P}\,U).\Pi x\!:\!U.\texttt{pf}\,(\neg x \in A \hat{\cup} B)$
$\rightarrow \texttt{pf}\,(\neg x \in B)$

$\texttt{binunionTSub1}:\quad \Pi U\!:\!\texttt{set}.\quad \Pi A\!:\!(\mathcal{P}\,U).\quad \Pi B\!:\!(\mathcal{P}\,U).\quad \texttt{pf}\,(A \subseteq A \hat{\cup} B)$

$\texttt{binunionTSub2}:\quad \Pi U\!:\!\texttt{set}.\quad \Pi A\!:\!(\mathcal{P}\,U).\quad \Pi B\!:\!(\mathcal{P}\,U).\quad \texttt{pf}\,(B \subseteq A \hat{\cup} B)$

$\texttt{inIntersectImpInUnion}:\quad \Pi U\!:\!\texttt{set}.\quad \Pi A\!:\!(\mathcal{P}\,U).\quad \Pi B\!:\!(\mathcal{P}\,U).\quad \Pi C\!:\!(\mathcal{P}\,U).\quad \Pi x\!:\!U\,.$
$\texttt{pf}\,(x \in A \hat{\cap} B) \;\rightarrow\; \texttt{pf}\,(x \in A \hat{\cup} C)$

$\texttt{inIntersectImpInUnion2}:\quad \Pi U\!:\!\texttt{set}.\quad \Pi A\!:\!(\mathcal{P}\,U).\quad \Pi B\!:\!(\mathcal{P}\,U).\quad \Pi C\!:\!(\mathcal{P}\,U).\quad \Pi x\!:\!U\,.$
$\texttt{pf}\,(x \in A \hat{\cap} B) \;\rightarrow\; \texttt{pf}\,(x \in B \hat{\cup} C)$

$\texttt{inIntersectImpInIntersectUnions}:\quad \Pi U\!:\!\texttt{set}.\Pi A\!:\!(\mathcal{P}\,U).\Pi B\!:\!(\mathcal{P}\,U).\Pi C\!:\!(\mathcal{P}\,U).$
$\Pi x\!:\!U.\quad \texttt{pf}\,(x \in A \hat{\cap} B) \;\rightarrow\; \texttt{pf}\,(x \in (A \hat{\cup} C) \hat{\cap} (B \hat{\cup} C))$

$\texttt{intersectInPowersetIntersectUnions}:\quad \Pi U\!:\!\texttt{set}.\Pi A\!:\!(\mathcal{P}\,U).\Pi B\!:\!(\mathcal{P}\,U).\Pi C\!:\!(\mathcal{P}\,U).$
$\texttt{pf}\,(A \hat{\cap} B \in \hat{\mathcal{P}}\,((A \hat{\cup} C) \hat{\cap} (B \hat{\cup} C)))$

$\texttt{inComplementUnionImpNotIn1}:\quad \Pi U\!:\!\texttt{set}.\quad \Pi A\!:\!(\mathcal{P}\,U).\quad \Pi B\!:\!(\mathcal{P}\,U).\quad \Pi x\!:\!U\,.$
$\texttt{pf}\,(x \in \hat{\mathcal{K}}\,(A \hat{\cup} B)) \;\rightarrow\; \texttt{pf}\,(\neg x \in A)$

$\texttt{inComplementUnionImpInComplement1}:\quad \Pi U\!:\!\texttt{set}.\quad \Pi A\!:\!(\mathcal{P}\,U).\quad \Pi B\!:\!(\mathcal{P}\,U).\quad \Pi x\!:\!U\,.$
$\texttt{pf}\,(x \in \hat{\mathcal{K}}\,(A \hat{\cup} B)) \;\rightarrow\; \texttt{pf}\,(x \in \hat{\mathcal{K}}\,A)$

$\texttt{binunionTE1}:\quad \Pi U\!:\!\texttt{set}.\quad \Pi A\!:\!(\mathcal{P}\,U).\quad \Pi B\!:\!(\mathcal{P}\,U).\quad \Pi x\!:\!U.\quad \texttt{pf}\,(x \in A \hat{\cup} B) \;\rightarrow$
$\texttt{pf}\,((x \in A) \vee (x \in B))$

$\texttt{binunionTE}:\quad \Pi U\!:\!\texttt{set}.\quad \Pi A\!:\!(\mathcal{P}\,U).\quad \Pi B\!:\!(\mathcal{P}\,U).\quad \Pi \phi\!:\!\texttt{prop}.\quad \Pi x\!:\!U.\quad \texttt{pf}\,(x \in A \hat{\cup} B)$
$\rightarrow\; (\texttt{pf}\,(x \in A) \;\rightarrow\; \texttt{pf}\,\phi) \;\rightarrow\; (\texttt{pf}\,(x \in B) \;\rightarrow\; \texttt{pf}\,\phi) \;\rightarrow\; \texttt{pf}\,\phi$

$\texttt{binunionTEcontra}:\quad \Pi U\!:\!\texttt{set}.\quad \Pi A\!:\!(\mathcal{P}\,U).\quad \Pi B\!:\!(\mathcal{P}\,U).\quad \Pi x\!:\!U\,.$
$\texttt{pf}\,(\neg x \in A) \;\rightarrow\; \texttt{pf}\,(\neg x \in B) \;\rightarrow\; \texttt{pf}\,(\neg x \in A \hat{\cup} B)$

The following are versions of the DeMorgan's Law.

$\texttt{demorgan2a1}:\quad \Pi U\!:\!\texttt{set}.\Pi A\!:\!(\mathcal{P}U).\Pi B\!:\!(\mathcal{P}U).\Pi x\!:\!U.\texttt{pf}\,(x \in \hat{\mathcal{K}}(A \hat{\cup} B)) \rightarrow \texttt{pf}\,(x \in \hat{\mathcal{K}}A)$

$\texttt{demorgan2a2}:\quad \Pi U\!:\!\texttt{set}.\Pi A\!:\!(\mathcal{P}U).\Pi B\!:\!(\mathcal{P}U).\Pi x\!:\!U.\texttt{pf}\,(x \in \hat{\mathcal{K}}(A \hat{\cup} B)) \rightarrow \texttt{pf}\,(x \in \hat{\mathcal{K}}B)$

$\texttt{demorgan1}:\quad \Pi U\!:\!\texttt{set}.\Pi A\!:\!(\mathcal{P}U).\Pi B\!:\!(\mathcal{P}U).\texttt{pf}\,(\hat{\mathcal{K}}(A \hat{\cap} B) = (\hat{\mathcal{K}}A) \hat{\cup} (\hat{\mathcal{K}}B))$

$\texttt{demorgan2}:\quad \Pi U\!:\!\texttt{set}.\Pi A\!:\!(\mathcal{P}U).\Pi B\!:\!(\mathcal{P}U).\texttt{pf}\,(\hat{\mathcal{K}}(A \hat{\cup} B) = (\hat{\mathcal{K}}A) \hat{\cap} (\hat{\mathcal{K}}B))$

The following rules are especially relevant to the problem `woz1_3`.

$\texttt{woz13rule0}:\quad \Pi U\!:\!\texttt{set}.\quad \Pi A\!:\!(\mathcal{P}\,U).\quad \Pi B\!:\!(\mathcal{P}\,U).\quad \Pi C\!:\!\texttt{set}.\quad \texttt{pf}\,(C \in A \hat{\cap} B)$
$\rightarrow\; \texttt{pf}\,(C \in U)$

$\texttt{woz13rule1}:\quad \Pi U\!:\!\texttt{set}.\quad \Pi A\!:\!(\mathcal{P}\,U).\quad \Pi B\!:\!(\mathcal{P}\,U).\quad \Pi C\!:\!(\mathcal{P}\,U).\quad \texttt{pf}\,(A \subseteq C)$
$\rightarrow\; \texttt{pf}\,(A \hat{\cap} B \subseteq C)$

$\texttt{woz1\_1}: \quad \Pi U{:}\texttt{set}. \quad \Pi A{:}(\mathcal{P}\,U). \quad \Pi B{:}(\mathcal{P}\,U). \quad \texttt{pf}\,(\hat{\mathcal{K}}\,A \in \hat{\mathcal{P}}\,(\hat{\mathcal{K}}\,(A \hat{\cap} B)))$
$= \lambda U\,\lambda A\,\lambda B\,\texttt{powersetTI1}\,U\,(\hat{\mathcal{K}}\,A)\,(\hat{\mathcal{K}}\,(A \hat{\cap} B))\,(\lambda x\,\lambda \mathcal{D}\,\mathbf{L3})$
using proof steps

L1 :	$\texttt{pf}\,(\neg x \in A)$	by: $\texttt{complementTE}\,U\,A\,x\,\mathcal{D}$
L2 :	$\texttt{pf}\,(\neg x \in A \hat{\cap} B)$	by: $\texttt{binintersectTELcontra}\,U\,A\,B\,x\,\mathbf{L1}$
L3 :	$\texttt{pf}\,(x \in \hat{\mathcal{K}}\,(A \hat{\cap} B))$	by: $\texttt{complementTI}\,U\,(A \hat{\cap} B)\,x\,\mathbf{L2}$

Figure 4: `woz1_1`

$\texttt{woz1\_2}: \quad \Pi U{:}\texttt{set}. \quad \Pi A{:}(\mathcal{P}\,U). \quad \Pi B{:}(\mathcal{P}\,U). \quad \Pi C{:}(\mathcal{P}\,U). \quad \Pi D{:}(\mathcal{P}\,U).$
$\texttt{pf}\,(\hat{\mathcal{K}}\,((A \hat{\cup} B) \hat{\cap} (C \hat{\cup} D)) = ((\hat{\mathcal{K}}\,A) \hat{\cap} (\hat{\mathcal{K}}\,B)) \hat{\cup} ((\hat{\mathcal{K}}\,C) \hat{\cap} (\hat{\mathcal{K}}\,D)))$
$= \lambda U\,\lambda A\,\lambda B\,\lambda C\,\lambda D\,\texttt{eqCE}\,(\texttt{in}\,(\mathcal{P}\,U))\,(\hat{\mathcal{K}}\,(C \hat{\cup} D))\,((\hat{\mathcal{K}}\,C) \hat{\cap} (\hat{\mathcal{K}}\,D))$
$(\lambda X\,\hat{\mathcal{K}}\,((A \hat{\cup} B) \hat{\cap} (C \hat{\cup} D)) = ((\hat{\mathcal{K}}\,A) \hat{\cap} (\hat{\mathcal{K}}\,B)) \hat{\cup} X)\,\mathbf{L1}\,\mathbf{L4}$
using proof steps

L1 :	$\texttt{pf}\,(\hat{\mathcal{K}}\,(C \hat{\cup} D) = (\hat{\mathcal{K}}\,C) \hat{\cap} (\hat{\mathcal{K}}\,D))$	by: $\texttt{demorgan2}\,U\,C\,D$
L2 :	$\texttt{pf}\,(\hat{\mathcal{K}}\,(A \hat{\cup} B) = (\hat{\mathcal{K}}\,A) \hat{\cap} (\hat{\mathcal{K}}\,B))$	by: $\texttt{demorgan2}\,U\,A\,B$
L3 :	$\texttt{pf}\,(\hat{\mathcal{K}}\,((A \hat{\cup} B) \hat{\cap} (C \hat{\cup} D)) = (\hat{\mathcal{K}}\,(A \hat{\cup} B)) \hat{\cup} (\hat{\mathcal{K}}\,(C \hat{\cup} D)))$	by: $\texttt{demorgan1}\,U\,(A \hat{\cup} B)\,(C \hat{\cup} D)$
L4 :	$\texttt{pf}\,(\hat{\mathcal{K}}\,((A \hat{\cup} B) \hat{\cap} (C \hat{\cup} D)) = ((\hat{\mathcal{K}}\,A) \hat{\cap} (\hat{\mathcal{K}}\,B)) \hat{\cup} (\hat{\mathcal{K}}\,(C \hat{\cup} D)))$	by: $\texttt{eqCE}\,(\texttt{in}\,(\mathcal{P}\,U))\,(\hat{\mathcal{K}}\,(A \hat{\cup} B))\,((\hat{\mathcal{K}}\,A) \hat{\cap} (\hat{\mathcal{K}}\,B))$ $(\lambda X\,\hat{\mathcal{K}}\,((A \hat{\cup} B) \hat{\cap} (C \hat{\cup} D)) = X \hat{\cup} (\hat{\mathcal{K}}\,(C \hat{\cup} D)))\,\mathbf{L2}\,\mathbf{L3}$

Figure 5: `woz1_2`

$\texttt{woz13rule2}: \quad \Pi U{:}\texttt{set}. \quad \Pi A{:}(\mathcal{P}\,U). \quad \Pi B{:}(\mathcal{P}\,U). \quad \Pi C{:}(\mathcal{P}\,U). \quad \texttt{pf}\,(B \subseteq C)$
$\to \texttt{pf}\,(A \hat{\cap} B \subseteq C)$

$\texttt{woz13rule3}: \quad \Pi U{:}\texttt{set}. \quad \Pi A{:}(\mathcal{P}\,U). \quad \Pi B{:}(\mathcal{P}\,U). \quad \Pi C{:}(\mathcal{P}\,U). \quad \texttt{pf}\,(A \subseteq B)$
$\to \texttt{pf}\,(A \subseteq C) \to \texttt{pf}\,(A \subseteq B \hat{\cap} C)$

$\texttt{woz13rule4}: \quad \Pi U{:}\texttt{set}.\Pi A{:}(\mathcal{P}\,U).\Pi B{:}(\mathcal{P}\,U).\Pi C{:}(\mathcal{P}\,U).\Pi D{:}(\mathcal{P}\,U). \quad \texttt{pf}\,(A \subseteq C)$
$\to \texttt{pf}\,(B \subseteq D) \to \texttt{pf}\,(A \hat{\cap} B \subseteq C \hat{\cap} D)$

Finally, we formulate abbreviations with types corresponding to the five problems in the Wizard of Oz experiment described in [5]. We also give sample proof terms. To analyze the proof attempts of students, alternative proofs will likely need to be considered. In fact, one likely needs to consider *partial* proof terms which could be represented by terms which have free variables of proof type which need to be instantiated with subproofs. These abbreviations are `woz1_1` (see Figure 4), `woz1_2` (see Figure 5), `woz1_3` (see Figure 6), `woz1_4` (see Figure 7), and `woz1_5` (see Figure 8).

$\texttt{woz1\_3}: \quad \Pi U\!:\!\texttt{set}. \quad \Pi A\!:\!(\mathcal{P}\,U). \quad \Pi B\!:\!(\mathcal{P}\,U). \quad \Pi C\!:\!(\mathcal{P}\,U)\,.$
$\texttt{pf}\,(A \hat{\cap} B \in \hat{\mathcal{P}}\,((A \hat{\cup} C) \hat{\cap} (B \hat{\cup} C)))$
$= \lambda U\,\lambda A\,\lambda B\,\lambda C\,\texttt{powersetTI}\,U\,(A \hat{\cap} B)\,((A \hat{\cup} C) \hat{\cap} (B \hat{\cup} C))\,\mathbf{L5}$
using proof steps

$\mathbf{L1}$:	$\texttt{pf}\,(A \subseteq A \hat{\cup} C)$	by: $\texttt{binunionTSub1}\,U\,A\,C$
$\mathbf{L2}$:	$\texttt{pf}\,(A \hat{\cap} B \subseteq A \hat{\cup} C)$	by: $\texttt{woz13rule1}\,U\,A\,B\,(A \hat{\cup} C)\,\mathbf{L1}$
$\mathbf{L3}$:	$\texttt{pf}\,(B \subseteq B \hat{\cup} C)$	by: $\texttt{binunionTSub1}\,U\,B\,C$
$\mathbf{L4}$:	$\texttt{pf}\,(A \hat{\cap} B \subseteq B \hat{\cup} C)$	by: $\texttt{woz13rule2}\,U\,A\,B\,(B \hat{\cup} C)\,\mathbf{L3}$
$\mathbf{L5}$:	$\texttt{pf}\,(A \hat{\cap} B \subseteq (A \hat{\cup} C) \hat{\cap} (B \hat{\cup} C))$	by: $\texttt{woz13rule3}\,U\,(A \hat{\cap} B)\,(A \hat{\cup} C)\,(B \hat{\cup} C)\,\mathbf{L2}\,\mathbf{L4}$

Figure 6: `woz1_3`

$\texttt{woz1\_4}: \quad \Pi U\!:\!\texttt{set}. \quad \Pi A\!:\!(\mathcal{P}\,U). \quad \Pi B\!:\!(\mathcal{P}\,U). \quad \texttt{pf}\,(A \subseteq \hat{\mathcal{K}}\,B) \to \texttt{pf}\,(B \subseteq \hat{\mathcal{K}}\,A)$
$= \lambda U\,\lambda A\,\lambda B\,\lambda \mathcal{D}\,\texttt{subsetTI}\,U\,B\,(\hat{\mathcal{K}}\,A)\,(\lambda x\,\lambda \mathcal{E}\,\mathbf{L2})$
using proof steps

$\mathbf{L1}$:	$\texttt{pf}\,(\neg x \in A)$	by: $\texttt{contrasubsetT}\,U\,A\,B\,x\,\mathcal{D}\,\mathcal{E}$
$\mathbf{L2}$:	$\texttt{pf}\,(x \in \hat{\mathcal{K}}\,A)$	by: $\texttt{complementTI}\,U\,A\,x\,\mathbf{L1}$

Figure 7: `woz1_4`

$\texttt{woz1\_5}: \quad \Pi U\!:\!\texttt{set}. \quad \Pi A\!:\!(\mathcal{P}\,U). \quad \Pi B\!:\!(\mathcal{P}\,U). \quad \texttt{pf}\,(\hat{\mathcal{K}}\,(A \hat{\cup} B) \in \hat{\mathcal{P}}\,(\hat{\mathcal{K}}\,A))$
$= \lambda U\,\lambda A\,\lambda B\,\texttt{powersetTI1}\,U\,(\hat{\mathcal{K}}\,(A \hat{\cup} B))\,(\hat{\mathcal{K}}\,A)\,(\lambda x\,\lambda \mathcal{D}\,\mathbf{L1})$
using proof step

$\mathbf{L1}$:	$\texttt{pf}\,(x \in \hat{\mathcal{K}}\,A)$	by: $\texttt{demorgan2a1}\,U\,A\,B\,x\,\mathcal{D}$

Figure 8: `woz1_5`

5.2 Formalizing the Second DiaLog Wizard of Oz Experiment: Relations

The following abbreviations concern relations and are relevant to the second Wizard of Oz experiment in the DiaLog project [6, 31].

We omit the definition of `breln1`, but the idea is that ($\texttt{breln1}\, M\, R$) holds iff $R \subseteq M \times M$ (i.e., R is a binary relation on M). We will use ($\texttt{breln1}\, M$) – which has type $\texttt{set} \rightarrow \texttt{prop}$ – as a class type.

$\texttt{breln1}: \quad \texttt{set} \rightarrow \texttt{set} \rightarrow \texttt{prop}$

The following correspond to derivable rules which are useful for this domain:

$\texttt{breln1all1}: \quad \Pi M{:}\texttt{set}. \quad \Pi R{:}(\texttt{breln1}\, M). \quad \Pi\phi{:}(\texttt{set} \rightarrow \texttt{prop})\,.$
$(\Pi a{:}M. \quad \Pi b{:}M. \quad \texttt{pf}\,(\langle\langle\, a,b\,\rangle\rangle \in R) \rightarrow \texttt{pf}\,(\phi\,\langle\langle\, a,b\,\rangle\rangle)) \rightarrow \texttt{pf}\,(\forall c \in R.\, \phi\, c)$

$\texttt{subbreln1}: \quad \Pi M{:}\texttt{set}. \quad \Pi R{:}(\texttt{breln1}\, M). \quad \Pi S{:}(\texttt{breln1}\, M)\,.$
$(\Pi a{:}M. \quad \Pi b{:}M. \quad \texttt{pf}\,(\langle\langle\, a,b\,\rangle\rangle \in R) \rightarrow \texttt{pf}\,(\langle\langle\, a,b\,\rangle\rangle \in S)) \rightarrow \texttt{pf}\,(R \subseteq S)$

$\texttt{eqbreln1}: \quad \Pi M{:}\texttt{set}. \quad \Pi R{:}(\texttt{breln1}\, M). \quad \Pi S{:}(\texttt{breln1}\, M)\,.$
$(\Pi a{:}M. \quad \Pi b{:}M. \quad \texttt{pf}\,(\langle\langle\, a,b\,\rangle\rangle \in R) \rightarrow \texttt{pf}\,(\langle\langle\, a,b\,\rangle\rangle \in S)) \rightarrow$
$(\Pi a{:}M. \quad \Pi b{:}M. \quad \texttt{pf}\,(\langle\langle\, a,b\,\rangle\rangle \in S) \rightarrow \texttt{pf}\,(\langle\langle\, a,b\,\rangle\rangle \in R)) \rightarrow \texttt{pf}\,(R = S)$

We define the inverse of a binary relation R on M using the "set of pairs constructor" `dpsetconstr`. We will do this in three steps: define the untyped set which is the inverse of R, prove this untyped set is a binary relation on M, pair the previous two abbreviations to have an inverse operator which expects a binary relation on M and returns a binary relation on M. Explicitly defining the untyped set is optional.

$\texttt{breln1invset}: \quad \Pi M{:}\texttt{set}. \quad (\texttt{breln1}\, M) \rightarrow \texttt{set}$
$= \lambda M\, \lambda R\, \{\langle\langle a,b\rangle\rangle \in\ M\ \times\ M \mid \langle\langle\, b,a\,\rangle\rangle \in R\,\}$

$\texttt{breln1invprop}: \quad \Pi M{:}\texttt{set}. \quad \Pi R{:}(\texttt{breln1}\, M). \quad \texttt{pf}\,(\texttt{breln1}\, M\,(\texttt{breln1invset}\, M\, R))$

$\texttt{breln1inv}: \quad \Pi M{:}\texttt{set}. \quad (\texttt{breln1}\, M) \rightarrow (\texttt{breln1}\, M)$
$= \lambda M\, \lambda R\, \langle\, \texttt{breln1invset}\, M\, R, \texttt{breln1invprop}\, M\, R\,\rangle$

We use R^{-1} to denote the term ($\texttt{breln1inv}\, M\, R$) where M is a fixed bound variable.

Now that we have defined the inverse operator, we can derive two proof rules. We can always use these two proof rules instead of expanding the definition of `breln1inv` or `breln1invset` above.

$\texttt{breln1invI}: \quad \Pi M{:}\texttt{set}. \quad \Pi R{:}(\texttt{breln1}\, M). \quad \Pi a{:}M. \quad \Pi b{:}M. \quad \texttt{pf}\,(\langle\langle\, a,b\,\rangle\rangle \in R)$
$\rightarrow \texttt{pf}\,(\langle\langle\, b\,,\, a\,\rangle\rangle \in R^{-1})$

$\texttt{breln1invE}: \quad \Pi M{:}\texttt{set}. \quad \Pi R{:}(\texttt{breln1}\, M). \quad \Pi a{:}M. \quad \Pi b{:}M. \quad \texttt{pf}\,(\langle\langle\, b,a\,\rangle\rangle \in R^{-1})$
$\rightarrow \texttt{pf}\,(\langle\langle\, a,b\,\rangle\rangle \in R)$

Following a similar outline, we define the composition of two relations on a set.

$\texttt{breln1compset}: \quad \Pi M{:}\texttt{set}. \quad (\texttt{breln1}\, M) \rightarrow (\texttt{breln1}\, M) \rightarrow \texttt{set}$
$= \lambda M\, \lambda R\, \lambda S\, \{\langle\langle a,b\rangle\rangle \in\ M\ \times\ M \mid \exists c \in M.\, (\langle\langle\, a,c\,\rangle\rangle \in R) \wedge (\langle\langle\, c,b\,\rangle\rangle \in S)\,\}$

$\texttt{breln1compprop}: \quad \Pi M\!:\!\texttt{set}.\Pi R\!:\!(\texttt{breln1}\,M).\Pi S\!:\!(\texttt{breln1}\,M).$
$\texttt{pf}\,(\texttt{breln1}\,M\,(\texttt{breln1compset}\,M\,R\,S))$

$\texttt{breln1comp}: \quad \Pi M\!:\!\texttt{set}. \quad (\texttt{breln1}\,M) \to (\texttt{breln1}\,M) \to (\texttt{breln1}\,M)$
$= \lambda M\,\lambda R\,\lambda S\,\langle\,\texttt{breln1compset}\,M\,R\,S, \texttt{breln1compprop}\,M\,R\,S\,\rangle$

We use $R \circ S$ to denote the term $(\texttt{breln1comp}\,M\,R\,S)$ where M is a fixed bound variable. Introduction and elimination rules for $\circ$ follow:

$\texttt{breln1compI}: \quad \Pi M\!:\!\texttt{set}. \quad \Pi R\!:\!(\texttt{breln1}\,M).\Pi S\!:\!(\texttt{breln1}\,M).\Pi a\!:\!M.\Pi b\!:\!M.\Pi c\!:\!M.$
$\texttt{pf}\,(\langle\langle\,a,c\,\rangle\rangle \in R) \to \texttt{pf}\,(\langle\langle\,c,b\,\rangle\rangle \in S) \to \texttt{pf}\,(\langle\langle\,a,b\,\rangle\rangle \in R \circ S)$

$\texttt{breln1compE}: \quad \Pi M\!:\!\texttt{set}. \quad \Pi R\!:\!(\texttt{breln1}\,M). \quad \Pi S\!:\!(\texttt{breln1}\,M). \quad \Pi a\!:\!M. \quad \Pi b\!:\!M\,.$
$\texttt{pf}\,(\langle\langle\,a,b\,\rangle\rangle \in R \circ S) \to \texttt{pf}\,(\exists c \in M.\,(\langle\langle\,a,c\,\rangle\rangle \in R) \wedge (\langle\langle\,c,b\,\rangle\rangle \in S))$

Finally, we define a notion of binary union on binary relations on M. This time we do not explicitly define the untyped set since in fact the untyped set is already defined by `binunion` (for which we use the infix notation $\cup$). Note that in the definition of `breln1union` below the term $(R \cup S)$ is shorthand for $(\texttt{binunion}\,R\,S)$ which in turn is shorthand for $(\texttt{binunion}\,\pi_1(R)\,\pi_1(S))$.

$\texttt{breln1unionprop}: \quad \Pi M\!:\!\texttt{set}.\Pi R\!:\!(\texttt{breln1}\,M).\Pi S\!:\!(\texttt{breln1}\,M).\texttt{pf}\,(\texttt{breln1}\,M\,(R \cup S))$

$\texttt{breln1union}: \quad \Pi M\!:\!\texttt{set}. \quad (\texttt{breln1}\,M) \to (\texttt{breln1}\,M) \to (\texttt{breln1}\,M)$
$= \lambda M\,\lambda R\,\lambda S\,\langle\,R \cup S, \texttt{breln1unionprop}\,M\,R\,S\,\rangle$

Note that $X \cup Y$ is already used for untyped binary union $(\texttt{binunion}\,X\,Y)$ and $\hat{\cup}$ is already used for typed binary union $(\texttt{binunionT}\,U\,A\,B)$ relative to a fixed U. For binary union of binary relations on a fixed set M, we use $R \dot{\cup} S$ to denote $(\texttt{breln1union}\,M\,R\,S)$. Rules for $\dot{\cup}$ follow:

$\texttt{breln1unionIL}: \quad \Pi M\!:\!\texttt{set}. \quad \Pi R\!:\!(\texttt{breln1}\,M). \quad \Pi S\!:\!(\texttt{breln1}\,M). \quad \Pi a\!:\!M. \quad \Pi b\!:\!M\,.$
$\texttt{pf}\,(\langle\langle\,a,b\,\rangle\rangle \in R) \to \texttt{pf}\,(\langle\langle\,a,b\,\rangle\rangle \in R \dot{\cup} S)$

$\texttt{breln1unionIR}: \quad \Pi M\!:\!\texttt{set}. \quad \Pi R\!:\!(\texttt{breln1}\,M). \quad \Pi S\!:\!(\texttt{breln1}\,M). \quad \Pi a\!:\!M. \quad \Pi b\!:\!M\,.$
$\texttt{pf}\,(\langle\langle\,a,b\,\rangle\rangle \in S) \to \texttt{pf}\,(\langle\langle\,a,b\,\rangle\rangle \in R \dot{\cup} S)$

$\texttt{breln1unionI}: \quad \Pi M\!:\!\texttt{set}. \quad \Pi R\!:\!(\texttt{breln1}\,M). \quad \Pi S\!:\!(\texttt{breln1}\,M). \quad \Pi a\!:\!M. \quad \Pi b\!:\!M\,.$
$\texttt{pf}\,((\langle\langle\,a,b\,\rangle\rangle \in R) \vee (\langle\langle\,a,b\,\rangle\rangle \in S)) \to \texttt{pf}\,(\langle\langle\,a,b\,\rangle\rangle \in R \dot{\cup} S)$

$\texttt{breln1unionE}: \quad \Pi M\!:\!\texttt{set}. \quad \Pi R\!:\!(\texttt{breln1}\,M). \quad \Pi S\!:\!(\texttt{breln1}\,M). \quad \Pi a\!:\!M. \quad \Pi b\!:\!M\,.$
$\texttt{pf}\,(\langle\langle\,a,b\,\rangle\rangle \in R \dot{\cup} S) \to \texttt{pf}\,((\langle\langle\,a,b\,\rangle\rangle \in R) \vee (\langle\langle\,a,b\,\rangle\rangle \in S))$

$\texttt{breln1unionEcases}: \quad \Pi M\!:\!\texttt{set}.\Pi R\!:\!(\texttt{breln1}\,M).\Pi S\!:\!(\texttt{breln1}\,M).\Pi a\!:\!M.\Pi b\!:\!M.$
$\texttt{pf}\,(\langle\langle\,a,b\,\rangle\rangle \in R \dot{\cup} S) \to \Pi \phi\!:\!\texttt{prop}. \quad (\texttt{pf}\,(\langle\langle\,a,b\,\rangle\rangle \in R) \to \texttt{pf}\,\phi) \to$
$(\texttt{pf}\,(\langle\langle\,a,b\,\rangle\rangle \in S) \to \texttt{pf}\,\phi) \to \texttt{pf}\,\phi$

$\texttt{breln1unionCommutes}: \quad \Pi M\!:\!\texttt{set}.\Pi R\!:\!(\texttt{breln1}\,M).\Pi S\!:\!(\texttt{breln1}\,M).\texttt{pf}\,(R \dot{\cup} S = S \dot{\cup} R)$

We now give four abbreviations `woz2Ex` (see Figure 9), `woz2W` (see Figure 10), `woz2A` (see Figures 11 and 12), and `woz2B` (see Figure 13). The types of these abbreviations correspond to the problem in the Wizard of Oz experiment described in [31]. We include sample proof

$\texttt{woz2Ex}: \quad \Pi M\!:\!\texttt{set}. \quad \Pi R\!:\!(\texttt{breln1}\, M). \quad \texttt{pf}\,(R = (R^{-1})^{-1})$
$= \lambda M\, \lambda R\, \texttt{setextsub}\, R\, (R^{-1})^{-1}\, \mathbf{L3}\, \mathbf{L6}$
using proof steps

	Let a have type M	
	Let b have type M	
$\mathcal{D}$	Assume $\langle\langle a, b\rangle\rangle \in R$	
$\mathbf{L1}$:	$\texttt{pf}\,(\langle\langle b, a\rangle\rangle \in R^{-1})$	by: $\texttt{breln1invI}\, M\, R\, a\, b\, \mathcal{D}$
$\mathbf{L2}$:	$\texttt{pf}\,(\langle\langle a, b\rangle\rangle \in (R^{-1})^{-1})$	by: $\texttt{breln1invI}\, M\, R^{-1}\, b\, a\, \mathbf{L1}$
$\mathbf{L3}$:	$\texttt{pf}\,(R \subseteq (R^{-1})^{-1})$	by: $\texttt{subbreln1}\, M\, R\, ((R^{-1})^{-1})\, (\lambda a\, \lambda b\, \lambda \mathcal{D}\, \mathbf{L2})$
	Let a have type M	
	Let b have type M	
$\mathcal{E}$	Assume $\langle\langle a, b\rangle\rangle \in (R^{-1})^{-1}$	
$\mathbf{L4}$:	$\texttt{pf}\,(\langle\langle b, a\rangle\rangle \in R^{-1})$	by: $\texttt{breln1invE}\, M\, R^{-1}\, b\, a\, \mathcal{E}$
$\mathbf{L5}$:	$\texttt{pf}\,(\langle\langle a, b\rangle\rangle \in R)$	by: $\texttt{breln1invE}\, M\, R\, a\, b\, \mathbf{L4}$
$\mathbf{L6}$:	$\texttt{pf}\,((R^{-1})^{-1} \subseteq R)$	by: $\texttt{subbreln1}\, M\, ((R^{-1})^{-1})\, R\, (\lambda a\, \lambda b\, \lambda \mathcal{E}\, \mathbf{L5})$

Figure 9: `woz2Ex`

terms as well. As with the other Wizard of Oz experiment, one should consider other proof terms and partial proofs (proof terms with variables) to evaluate student responses.

$\texttt{woz2W}:\quad \Pi M\!:\!\texttt{set}.\Pi R\!:\!(\texttt{breln1}\,M).\Pi S\!:\!(\texttt{breln1}\,M).\texttt{pf}\,((R\circ S)^{-1}=S^{-1}\circ R^{-1})$
$=\lambda M\,\lambda R\,\lambda S\,\texttt{setextsub}\,(R\circ S)^{-1}\,(S^{-1}\circ R^{-1})\,\mathbf{L9}\,\mathbf{L18}$
using proof steps

	Let a have type M	
	Let b have type M	
$\mathcal{D}$	Assume $\langle\langle a, b\rangle\rangle \in (R\circ S)^{-1}$	
L1 :	$\texttt{pf}\,(\langle\langle b, a\rangle\rangle \in R\circ S)$	by: $\texttt{breln1invE}\,M\,(R\circ S)\,b\,a\,\mathcal{D}$
L2 :	$\texttt{pf}\,(\exists c\in M.\,(\langle\langle b, c\rangle\rangle \in R)\wedge(\langle\langle c, a\rangle\rangle \in S))$	by: $\texttt{breln1compE}\,M\,R\,S\,b\,a\,\mathbf{L1}$
	Let c have type M	
$\mathcal{E}$	Assume $(\langle\langle b, c\rangle\rangle \in R)\wedge(\langle\langle c, a\rangle\rangle \in S)$	
L3 :	$\texttt{pf}\,(\langle\langle c, a\rangle\rangle \in S)$	by: $\texttt{andER}\,(\langle\langle b, c\rangle\rangle \in R)\,(\langle\langle c, a\rangle\rangle \in S)\,\mathcal{E}$
L4 :	$\texttt{pf}\,(\langle\langle a, c\rangle\rangle \in S^{-1})$	by: $\texttt{breln1invI}\,M\,S\,c\,a\,\mathbf{L3}$
L5 :	$\texttt{pf}\,(\langle\langle b, c\rangle\rangle \in R)$	by: $\texttt{andEL}\,(\langle\langle b, c\rangle\rangle \in R)\,(\langle\langle c, a\rangle\rangle \in S)\,\mathcal{E}$
L6 :	$\texttt{pf}\,(\langle\langle c, b\rangle\rangle \in R^{-1})$	by: $\texttt{breln1invI}\,M\,R\,b\,c\,\mathbf{L5}$
L7 :	$\texttt{pf}\,(\langle\langle a, b\rangle\rangle \in S^{-1}\circ R^{-1})$	by: $\texttt{breln1compI}\,M\,S^{-1}\,R^{-1}\,a\,b\,c\,\mathbf{L4}\,\mathbf{L6}$
L8 :	$\texttt{pf}\,(\langle\langle a, b\rangle\rangle \in S^{-1}\circ R^{-1})$	by: $\texttt{dexE}\,M\,(\lambda c\,(\langle\langle b, c\rangle\rangle \in R)\wedge(\langle\langle c, a\rangle\rangle \in S))\,\mathbf{L2}$ $(\langle\langle a, b\rangle\rangle \in S^{-1}\circ R^{-1})\,(\lambda c\,\lambda\mathcal{E}\,\mathbf{L7})$
L9 :	$\texttt{pf}\,((R\circ S)^{-1}\subseteq S^{-1}\circ R^{-1})$	by: $\texttt{subbreln1}\,M\,((R\circ S)^{-1})\,(S^{-1}\circ R^{-1})\,(\lambda a\,\lambda b\,\lambda\mathcal{D}\,\mathbf{L8})$
	Let a have type M	
	Let b have type M	
$\mathcal{F}$	Assume $\langle\langle a, b\rangle\rangle \in S^{-1}\circ R^{-1}$	
L10 :	$\texttt{pf}\,(\exists c\in M.\,(\langle\langle a, c\rangle\rangle \in S^{-1})\wedge(\langle\langle c, b\rangle\rangle \in R^{-1}))$	by: $\texttt{breln1compE}\,M\,S^{-1}\,R^{-1}\,a\,b\,\mathcal{F}$
	Let c have type M	
$\mathcal{G}$	Assume $(\langle\langle a, c\rangle\rangle \in S^{-1})\wedge(\langle\langle c, b\rangle\rangle \in R^{-1})$	
L11 :	$\texttt{pf}\,(\langle\langle c, b\rangle\rangle \in R^{-1})$	by: $\texttt{andER}\,(\langle\langle a, c\rangle\rangle \in S^{-1})\,(\langle\langle c, b\rangle\rangle \in R^{-1})\,\mathcal{G}$
L12 :	$\texttt{pf}\,(\langle\langle b, c\rangle\rangle \in R)$	by: $\texttt{breln1invE}\,M\,R\,b\,c\,\mathbf{L11}$
L13 :	$\texttt{pf}\,(\langle\langle a, c\rangle\rangle \in S^{-1})$	by: $\texttt{andEL}\,(\langle\langle a, c\rangle\rangle \in S^{-1})\,(\langle\langle c, b\rangle\rangle \in R^{-1})\,\mathcal{G}$
L14 :	$\texttt{pf}\,(\langle\langle c, a\rangle\rangle \in S)$	by: $\texttt{breln1invE}\,M\,S\,c\,a\,\mathbf{L13}$
L15 :	$\texttt{pf}\,(\langle\langle b, a\rangle\rangle \in R\circ S)$	by: $\texttt{breln1compI}\,M\,R\,S\,b\,a\,c\,\mathbf{L12}\,\mathbf{L14}$
L16 :	$\texttt{pf}\,(\langle\langle a, b\rangle\rangle \in (R\circ S)^{-1})$	by: $\texttt{breln1invI}\,M\,(R\circ S)\,b\,a\,\mathbf{L15}$
L17 :	$\texttt{pf}\,(\langle\langle a, b\rangle\rangle \in (R\circ S)^{-1})$	by: $\texttt{dexE}\,M\,(\lambda c\,(\langle\langle a, c\rangle\rangle \in S^{-1})\wedge(\langle\langle c, b\rangle\rangle \in R^{-1}))\,\mathbf{L10}$ $(\langle\langle a, b\rangle\rangle \in (R\circ S)^{-1})\,(\lambda c\,\lambda\mathcal{G}\,\mathbf{L16})$
L18 :	$\texttt{pf}\,(S^{-1}\circ R^{-1}\subseteq (R\circ S)^{-1})$	by: $\texttt{subbreln1}\,M\,(S^{-1}\circ R^{-1})\,((R\circ S)^{-1})\,(\lambda a\,\lambda b\,\lambda\mathcal{F}\,\mathbf{L17})$

Figure 10: `woz2W`

$\texttt{woz2A}: \quad \Pi M{:}\texttt{set}. \quad \Pi R{:}(\texttt{breln1}\,M). \quad \Pi S{:}(\texttt{breln1}\,M). \quad \Pi T{:}(\texttt{breln1}\,M).$
$\texttt{pf}\,((R \dot\cup S) \circ T = (R \circ T) \dot\cup (S \circ T))$
$= \lambda M\,\lambda R\,\lambda S\,\lambda T\,\texttt{setextsub}\,(R \dot\cup S) \circ T\,(R \circ T) \dot\cup (S \circ T)\,\mathbf{L12}\,\mathbf{L27}$
using proof steps

	Let a have type M	
	Let b have type M	
$\mathcal{D}$	Assume $\langle\langle a, b\rangle\rangle \in (R \dot\cup S) \circ T$	
L1 :	$\texttt{pf}\,(\exists c \in M.\,(\langle\langle a, c\rangle\rangle \in R \dot\cup S \wedge \langle\langle c, b\rangle\rangle \in T))$	by: $\texttt{breln1compE}\,M\,(R \dot\cup S)\,T\,a\,b\,\mathcal{D}$
	Let c have type M	
$\mathcal{E}$	Assume $\langle\langle a, c\rangle\rangle \in R \dot\cup S \wedge \langle\langle c, b\rangle\rangle \in T$	
L2 :	$\texttt{pf}\,(\langle\langle a, c\rangle\rangle \in R \dot\cup S)$	by: $\texttt{andEL}\,(\langle\langle a, c\rangle\rangle \in R \dot\cup S)\,(\langle\langle c, b\rangle\rangle \in T)\,\mathcal{E}$
L3 :	$\texttt{pf}\,((\langle\langle a, c\rangle\rangle \in R) \vee (\langle\langle a, c\rangle\rangle \in S))$	by: $\texttt{breln1unionE}\,M\,R\,S\,a\,c\,\mathbf{L2}$
$\mathcal{F}$	Assume $\langle\langle a, c\rangle\rangle \in R$	
L4 :	$\texttt{pf}\,(\langle\langle c, b\rangle\rangle \in T)$	by: $\texttt{andER}\,(\langle\langle a, c\rangle\rangle \in R \dot\cup S)\,(\langle\langle c, b\rangle\rangle \in T)\,\mathcal{E}$
L5 :	$\texttt{pf}\,(\langle\langle a, b\rangle\rangle \in R \circ T)$	by: $\texttt{breln1compI}\,M\,R\,T\,a\,b\,c\,\mathcal{F}\,\mathbf{L4}$
L6 :	$\texttt{pf}\,(\langle\langle a, b\rangle\rangle \in (R \circ T) \dot\cup (S \circ T))$	by: $\texttt{breln1unionIL}\,M\,(R \circ T)\,(S \circ T)\,a\,b\,\mathbf{L5}$
$\mathcal{G}$	Assume $\langle\langle a, c\rangle\rangle \in S$	
L7 :	$\texttt{pf}\,(\langle\langle c, b\rangle\rangle \in T)$	by: $\texttt{andER}\,(\langle\langle a, c\rangle\rangle \in R \dot\cup S)\,(\langle\langle c, b\rangle\rangle \in T)\,\mathcal{E}$
L8 :	$\texttt{pf}\,(\langle\langle a, b\rangle\rangle \in S \circ T)$	by: $\texttt{breln1compI}\,M\,S\,T\,a\,b\,c\,\mathcal{G}\,\mathbf{L7}$
L9 :	$\texttt{pf}\,(\langle\langle a, b\rangle\rangle \in (R \circ T) \dot\cup (S \circ T))$	by: $\texttt{breln1unionIR}\,M\,(R \circ T)\,(S \circ T)\,a\,b\,\mathbf{L8}$
L10 :	$\texttt{pf}\,(\langle\langle a, b\rangle\rangle \in (R \circ T) \dot\cup (S \circ T))$	by: $\texttt{orE}\,(\langle\langle a, c\rangle\rangle \in R)\,(\langle\langle a, c\rangle\rangle \in S)\,\mathbf{L3}$ $(\langle\langle a, b\rangle\rangle \in (R \circ T) \dot\cup (S \circ T))\,(\lambda\mathcal{F}\,\mathbf{L6})\,(\lambda\mathcal{G}\,\mathbf{L9})$
L11 :	$\texttt{pf}\,(\langle\langle a, b\rangle\rangle \in (R \circ T) \dot\cup (S \circ T))$	by: $\texttt{dexE}\,M\,(\lambda c\,(\langle\langle a, c\rangle\rangle \in R \dot\cup S) \wedge (\langle\langle c, b\rangle\rangle \in T))\,\mathbf{L1}$ $(\langle\langle a, b\rangle\rangle \in (R \circ T) \dot\cup (S \circ T))\,(\lambda c\,\lambda\mathcal{E}\,\mathbf{L10})$
L12 :	$\texttt{pf}\,((R \dot\cup S) \circ T \subseteq (R \circ T) \dot\cup (S \circ T))$	by: $\texttt{subbreln1}\,M\,((R \dot\cup S) \circ T)\,((R \circ T) \dot\cup (S \circ T))\,(\lambda a\,\lambda b\,\lambda\mathcal{D}\,\mathbf{L11})$

Figure 11: `woz2A`, Part 1

	Let a have type M	
	Let b have type M	
$\mathcal{E}^0$	Assume $\langle\langle a, b\rangle\rangle \in (R \circ T) \dot\cup (S \circ T)$	
L13 :	$\mathtt{pf}\,((\langle\langle a, b\rangle\rangle \in R \circ T) \vee (\langle\langle a, b\rangle\rangle \in S \circ T))$	by: $\mathtt{breln1unionE}\, M\, (R \circ T)\, (S \circ T)\, a\, b\, \mathcal{E}^0$
$\mathcal{F}^0$	Assume $\langle\langle a, b\rangle\rangle \in R \circ T$	
L14 :	$\mathtt{pf}\,(\exists c \in M. (\langle\langle a, c\rangle\rangle \in R) \wedge (\langle\langle c, b\rangle\rangle \in T))$	by: $\mathtt{breln1compE}\, M\, R\, T\, a\, b\, \mathcal{F}^0$
	Let c have type M	
$\mathcal{G}^0$	Assume $(\langle\langle a, c\rangle\rangle \in R) \wedge (\langle\langle c, b\rangle\rangle \in T)$	
L15 :	$\mathtt{pf}\,(\langle\langle a, c\rangle\rangle \in R)$	by: $\mathtt{andEL}\, (\langle\langle a, c\rangle\rangle \in R)\, (\langle\langle c, b\rangle\rangle \in T)\, \mathcal{G}^0$
L16 :	$\mathtt{pf}\,(\langle\langle a, c\rangle\rangle \in R \dot\cup S)$	by: $\mathtt{breln1unionIL}\, M\, R\, S\, a\, c\, \mathbf{L15}$
L17 :	$\mathtt{pf}\,(\langle\langle c, b\rangle\rangle \in T)$	by: $\mathtt{andER}\, (\langle\langle a, c\rangle\rangle \in R)\, (\langle\langle c, b\rangle\rangle \in T)\, \mathcal{G}^0$
L18 :	$\mathtt{pf}\,(\langle\langle a, b\rangle\rangle \in (R \dot\cup S) \circ T)$	by: $\mathtt{breln1compI}\, M\, (R \dot\cup S)\, T\, a\, b\, c\, \mathbf{L16}\, \mathbf{L17}$
L19 :	$\mathtt{pf}\,(\langle\langle a, b\rangle\rangle \in (R \dot\cup S) \circ T)$	by: $\mathtt{dexE}\, M\, (\lambda c\, (\langle\langle a, c\rangle\rangle \in R) \wedge (\langle\langle c, b\rangle\rangle \in T))\, \mathbf{L14}$ $(\langle\langle a, b\rangle\rangle \in (R \dot\cup S) \circ T)\, (\lambda c\, \lambda \mathcal{G}^0\, \mathbf{L18})$
$\mathcal{D}^1$	Assume $\langle\langle a, b\rangle\rangle \in S \circ T$	
L20 :	$\mathtt{pf}\,(\exists c \in M. (\langle\langle a, c\rangle\rangle \in S) \wedge (\langle\langle c, b\rangle\rangle \in T))$	by: $\mathtt{breln1compE}\, M\, S\, T\, a\, b\, \mathcal{D}^1$
	Let c have type M	
$\mathcal{E}^1$	Assume $(\langle\langle a, c\rangle\rangle \in S) \wedge (\langle\langle c, b\rangle\rangle \in T)$	
L21 :	$\mathtt{pf}\,(\langle\langle a, c\rangle\rangle \in S)$	by: $\mathtt{andEL}\, (\langle\langle a, c\rangle\rangle \in S)\, (\langle\langle c, b\rangle\rangle \in T)\, \mathcal{E}^1$
L22 :	$\mathtt{pf}\,(\langle\langle a, c\rangle\rangle \in R \dot\cup S)$	by: $\mathtt{breln1unionIR}\, M\, R\, S\, a\, c\, \mathbf{L21}$
L23 :	$\mathtt{pf}\,(\langle\langle c, b\rangle\rangle \in T)$	by: $\mathtt{andER}\, (\langle\langle a, c\rangle\rangle \in S)\, (\langle\langle c, b\rangle\rangle \in T)\, \mathcal{E}^1$
L24 :	$\mathtt{pf}\,(\langle\langle a, b\rangle\rangle \in (R \dot\cup S) \circ T)$	by: $\mathtt{breln1compI}\, M\, (R \dot\cup S)\, T\, a\, b\, c\, \mathbf{L22}\, \mathbf{L23}$
L25 :	$\mathtt{pf}\,(\langle\langle a, b\rangle\rangle \in (R \dot\cup S) \circ T)$	by: $\mathtt{dexE}\, M\, (\lambda c\, (\langle\langle a, c\rangle\rangle \in S) \wedge (\langle\langle c, b\rangle\rangle \in T))\, \mathbf{L20}$ $(\langle\langle a, b\rangle\rangle \in (R \dot\cup S) \circ T)\, (\lambda c\, \lambda \mathcal{E}^1\, \mathbf{L24})$
L26 :	$\mathtt{pf}\,(\langle\langle a, b\rangle\rangle \in (R \dot\cup S) \circ T)$	by: $\mathtt{orE}\, (\langle\langle a, b\rangle\rangle \in R \circ T)\, (\langle\langle a, b\rangle\rangle \in S \circ T)\, \mathbf{L13}$ $(\langle\langle a, b\rangle\rangle \in (R \dot\cup S) \circ T)\, (\lambda \mathcal{F}^0\, \mathbf{L19})\, (\lambda \mathcal{D}^1\, \mathbf{L25})$
L27 :	$\mathtt{pf}\,((R \circ T) \dot\cup (S \circ T) \subseteq (R \dot\cup S) \circ T)$	by: $\mathtt{subbreln1}\, M\, ((R \circ T) \dot\cup (S \circ T))\, ((R \dot\cup S) \circ T)\, (\lambda a\, \lambda b\, \lambda \mathcal{E}^0\, \mathbf{L26})$

Figure 12: `woz2A`, Part 2

$\texttt{woz2B}: \quad \Pi M\!:\!\texttt{set}. \quad \Pi R\!:\!(\texttt{breln1}\,M). \quad \Pi S\!:\!(\texttt{breln1}\,M). \quad \Pi T\!:\!(\texttt{breln1}\,M)\,.$
$\texttt{pf}\,((R \dot\cup S) \circ T \;=\; ((T^{-1} \circ S^{-1})^{-1}) \dot\cup ((T^{-1} \circ R^{-1})^{-1}))$
$= \lambda M\, \lambda R\, \lambda S\, \lambda T\, \texttt{transeq}\,((R \dot\cup S) \circ T)\,((T^{-1} \circ R^{-1})^{-1} \dot\cup (T^{-1} \circ S^{-1})^{-1})$
$((T^{-1} \circ S^{-1})^{-1} \dot\cup (T^{-1} \circ R^{-1})^{-1})\,\mathbf{L11}\,\mathbf{L12}$
using proof steps

$\mathbf{L1}: \quad \texttt{pf}\,((T^{-1} \circ S^{-1})^{-1} = ((S^{-1})^{-1} \circ (T^{-1})^{-1})) \qquad$ by: $\texttt{woz2W}\,M\,T^{-1}\,S^{-1}$
$\mathbf{L2}: \quad \texttt{pf}\,((T^{-1} \circ R^{-1})^{-1} = ((R^{-1})^{-1} \circ (T^{-1})^{-1})) \qquad$ by: $\texttt{woz2W}\,M\,T^{-1}\,R^{-1}$
$\mathbf{L3}: \quad \texttt{pf}\,(T = (T^{-1})^{-1}) \qquad$ by: $\texttt{woz2Ex}\,M\,T$
$\mathbf{L4}: \quad \texttt{pf}\,(S = (S^{-1})^{-1}) \qquad$ by: $\texttt{woz2Ex}\,M\,S$
$\mathbf{L5}: \quad \texttt{pf}\,(R = (R^{-1})^{-1}) \qquad$ by: $\texttt{woz2Ex}\,M\,R$
$\mathbf{L6}: \quad \texttt{pf}\,((R \dot\cup S) \circ T = (R \circ T) \dot\cup (S \circ T)) \qquad$ by: $\texttt{woz2A}\,M\,R\,S\,T$
$\mathbf{L7}: \quad \texttt{pf}\,((R \dot\cup S) \circ T = ((R^{-1})^{-1} \circ T) \dot\cup (S \circ T))$
by: $\texttt{eqCE}\,(\texttt{breln1}\,M)\,R\,(R^{-1})^{-1}$
$(\lambda X\,(R \dot\cup S) \circ T = (X \circ T) \dot\cup (S \circ T))\,\mathbf{L5}\,\mathbf{L6}$
$\mathbf{L8}: \quad \texttt{pf}\,((R \dot\cup S) \circ T = ((R^{-1})^{-1} \circ T) \dot\cup ((S^{-1})^{-1} \circ T))$
by: $\texttt{eqCE}\,(\texttt{breln1}\,M)\,S\,(S^{-1})^{-1}$
$(\lambda X\,(R \dot\cup S) \circ T = ((R^{-1})^{-1} \circ T) \dot\cup (X \circ T))\,\mathbf{L4}\,\mathbf{L7}$
$\mathbf{L9}: \quad \texttt{pf}\,((R \dot\cup S) \circ T \;=\; ((R^{-1})^{-1} \circ (T^{-1})^{-1}) \dot\cup ((S^{-1})^{-1} \circ (T^{-1})^{-1}))$
by: $\texttt{eqCE}\,(\texttt{breln1}\,M)\,T\,(T^{-1})^{-1}$
$(\lambda X\,(R \dot\cup S) \circ T = ((R^{-1})^{-1} \circ X) \dot\cup ((S^{\,-1})^{-1} \circ X))\,\mathbf{L3}\,\mathbf{L8}$
$\mathbf{L10}: \quad \texttt{pf}\,((R \dot\cup S) \circ T = (T^{-1} \circ R^{-1})^{-1} \dot\cup ((S^{-1})^{-1} \circ (T^{-1})^{-1}))$
by: $\texttt{eqCE2}\,(\texttt{breln1}\,M)\,((T^{-1} \circ R^{-1})^{-1})\,((R^{-1})^{-1} \circ (T^{-1})^{-1})$
$(\lambda X\,(R \dot\cup S) \circ T = X \dot\cup ((S^{-1})^{-1} \circ (T^{-1})^{-1}))\,\mathbf{L2}\,\mathbf{L9}$
$\mathbf{L11}: \quad \texttt{pf}\,((R \dot\cup S) \circ T = (T^{-1} \circ R^{-1})^{-1} \dot\cup (T^{-1} \circ S^{-1})^{-1})$
by: $\texttt{eqCE2}\,(\texttt{breln1}\,M)\,(T^{-1} \circ S^{-1})^{-1}\,((S^{-1})^{-1} \circ (T^{-1})^{-1})$
$(\lambda X\,(R \dot\cup S) \circ T = (T^{-1} \circ R^{-1})^{-1} \dot\cup X)\,\mathbf{L1}\,\mathbf{L10}$
$\mathbf{L12}: \quad \texttt{pf}\,((T^{-1} \circ R^{-1})^{-1} \dot\cup (T^{-1} \circ S^{-1})^{-1} = (T^{-1} \circ S^{-1})^{-1} \dot\cup (T^{-1} \circ R^{-1})^{-1})$
by: $\texttt{breln1unionCommutes}\,M\,(T^{-1} \circ R^{-1})^{-1}\,(T^{-1} \circ S^{-1})^{-1}$

Figure 13: woz2B

6 Future Work

There are many avenues of research left for the future. We discuss several possibilities here.

6.1 Equality Reasoning

At the object level, we have equality between sets (given by `eq`) and equivalence between propositions (given by `equiv`). There is no object-level notion of equality between function types or proof types. Indeed, due to proof irrelevance, there is no need to consider equality between terms of a proof type, since any two terms are always equal if they are the same proof type. By a "proof type" here, we mean any type of the form

$$\Pi x^1 : A^1 \cdots \Pi x^n : A^n \mathtt{pf}\, M.$$

We have already defined a partial function *Leib* which in a sense corresponds to equality at arbitrary types, except those which return a proof type. We can give an alternative partial function *Eq* as follows:

- $Eq((\Pi x : B.C), M, N) :=$
$$(\Pi x : B.Eq(C, [x/\mathbf{x}_\lambda^M]\mathbf{B}_\lambda^M, [x/\mathbf{x}_\lambda^N]\mathbf{B}_\lambda^N))$$
when $Eq(C, [x/\mathbf{x}_\lambda^M]\mathbf{B}_\lambda^M, [x/\mathbf{x}_\lambda^N]\mathbf{B}_\lambda^N)$ is defined.
- $Eq(\mathtt{prop}, M, N) := \mathtt{pf}\,(M \equiv N)$.
- $Eq(\mathtt{set}, M, N) := (\Pi P : (\mathtt{set} \to \mathtt{prop}).\mathtt{pf}\,(M = N)$
- $Eq(\mathtt{cl}\,\phi, M, N) := (\Pi P : (\mathtt{set} \to \mathtt{prop}).\mathtt{pf}\,(\pi_1(M) = \pi_1(N))$.

One can show by induction that for every non-proof type A, $Eq(A, M, N)$ is inhabited iff $Leib(A, M, N)$ is inhabited.

When replacing equals for equals within subterms, we need to know that if F and G are equal at type $(\Pi x : AB)$ and M and N are equal at type A, then (FM) and (GN) are equal at type $[M/x]B$ (or $[N/x]B$). Making sense of this can be tricky, since the types $[M/x]B$ and $[N/x]B$ are different, but should also be, in some sense, "the same". The notion of "sameness" here should correspond to a form of isomorphism. That is, the existence of terms (in context) $f : [M/x]B \to [N/x]B$ and $g : [N/x]B \to [M/x]B$ such that the terms $f \circ g$ and $g \circ f$ are "equal" to the identity function at the appropriate types.

Assuming we can prove the existence of terms showing such types are isomorphic, we could define "congruence" types as follows:

- $Cong^1((\Pi x : B.C), M, N) :=$
$$(\Pi x : B.\Pi y : B.\Pi u : Eq(B, x, y).Cong^1(C, [x/\mathbf{x}_\lambda^M]\mathbf{B}_\lambda^M, f([y/\mathbf{x}_\lambda^N]\mathbf{B}_\lambda^N)))$$
when $Cong^1(B, x, y)$ and $Cong^1(C, [x/\mathbf{x}_\lambda^M]\mathbf{B}_\lambda^M, [y/\mathbf{x}_\lambda^N]\mathbf{B}_\lambda^N)$ are defined and where $f : [y/x]C \to C$ is a term induced by the isomorphism of $[y/x]C$ and C (in a context with x, y and u).

- $Cong^1((\Pi x\!:\!B.C), M, N) :=$

$$(\Pi x\!:\!B.\Pi y\!:\!B.Cong^1(C, [x/\mathbf{x}_\lambda^M]\mathbf{B}_\lambda^M, f([y/\mathbf{x}_\lambda^N]\mathbf{B}_\lambda^N)))$$

  when $Cong^1(C, [x/\mathbf{x}_\lambda^M]\mathbf{B}_\lambda^M, f([y/\mathbf{x}_\lambda^N]\mathbf{B}_\lambda^N))$ is defined but $Cong^1(B, x, y)$ is not defined (i.e., B is a proof type) and where $f : [y/x]C \to C$ is a term induced by the isomorphism of $[y/x]C$ and C.

- $Cong^1(\mathtt{prop}, M, N) := \mathtt{pf}\ (M \equiv N)$.

- $Cong^1(\mathtt{set}, M, N) := \mathtt{pf}\ (M = N)$

- $Cong^1(\mathtt{cl}\,\phi, M, N) := \mathtt{pf}\ (\pi_1(M) = \pi_1(N)$.

The expected results are the following:

1. For each type A with $\Gamma \vdash A : Type^i$ with $i \in \{0, 1\}$, if $Cong^1(A, x, x)$ is defined, then there is a term C_A such that $\Gamma, x : A \vdash C_A \uparrow Cong^1(A, x, x)$.

2. For each logical constant c of type A, if $Cong^1(A, c, c)$ is defined (i.e., A is not a proof type), then $Cong^1(A, c, c)$ can be shown to be inhabited.

3. $Cong^1(A, M, M)$ is inhabited (at least in some signature extended with new abbreviations of proof type) whenever $\cdot \vdash_\Sigma M \sim M \uparrow A$ and A is not a proof type.

4. If $Eq(A, M, M)$, then $Cong^1(A, M, M)$.

There are alternative (not necessarily equivalent) possible definitions of congruence types. For example, the following definition of $Cong^2(A, B, M, N)$ avoids isomorphism of types by carrying two types as arguments. The idea is it is inhabited when M has type A, N has type B, A and B are isomorphic, and, up to isomorphism, M and N are equal.

- $Cong^2((\Pi x\!:\!B.C), (\Pi y\!:\!B'.C'), M, N) :=$

$$(\Pi x\!:\!B.\Pi y\!:\!B'.\Pi u\!:\!Cong^2(B, B', x, y).Cong^2(C, C', [x/\mathbf{x}_\lambda^M]\mathbf{B}_\lambda^M, f([y/\mathbf{x}_\lambda^N]\mathbf{B}_\lambda^N)))$$

  when $Cong^2(B, B', x, y)$ and $Cong^2(C, C', [x/\mathbf{x}_\lambda^M]\mathbf{B}_\lambda^M, [y/\mathbf{x}_\lambda^N]\mathbf{B}_\lambda^N)$ are defined.

- $Cong^2((\Pi x\!:\!B.C), (\Pi y\!:\!B'.C'), M, N) :=$

$$(\Pi x\!:\!B.\Pi y\!:\!B'.Cong^2(C, C', [x/\mathbf{x}_\lambda^M]\mathbf{B}_\lambda^M, [y/\mathbf{x}_\lambda^N]\mathbf{B}_\lambda^N))$$

  when $Cong^2(C, [x/\mathbf{x}_\lambda^M]\mathbf{B}_\lambda^M, f([y/\mathbf{x}_\lambda^N]\mathbf{B}_\lambda^N))$ is defined but $Cong^2(B, x, y)$ is not defined.

- $Cong^2(\mathtt{prop}, \mathtt{prop}, M, N) := \mathtt{pf}\ (M \equiv N)$.

- $Cong^2(\mathtt{set}, \mathtt{set}, M, N) := \mathtt{pf}\ (M = N)$

- $Cong^2(\mathtt{cl}\,\phi, \mathtt{cl}\,\psi, M, N) := \mathtt{pf}\ (\pi_1(M) = \pi_1(N)$.

Note that $Cong^2(A, B, M, N)$ is only defined when A and B have similar structure and neither are proof types. What is the "best" definition of congruence types is questionable and left open.

One can use congruence types to replace equals by equals within terms and underneath binders. For instance, one could define judgments $\Gamma \vdash^{\mathcal{E}} M = N : A$ for whether two terms are equal up to a given set $\mathcal{E}$ of equations in context, then prove that if the judgment holds and A is not a proof type, then $Cong^1(A, M, N)$ (or $Cong^2(A, A, M, N)$) is inhabited. At type `prop`, this means we can prove equivalence of M and N. If we know M and we can prove $M \equiv N$, then we can derive N. The details are left for future work.

6.2 Computations

One would like to use programs such as computer algebra systems to determine the truth of purely computational propositions and then build a DeTSeT proof term. This could especially be useful when working with, e.g., the natural numbers, real numbers, etc.

6.3 Inductive Definitions

One can encode inductively defined sets, relations and functions in ZFC (and so in DeTSeT). We sketch how such an encoding can be done by example.

Suppose S is intended to be the set inductively defined to contain an element *nil* and to be closed under a binary constructor *cons*. We can define a proposition $\phi(x, y)$ to be

$$\forall z.z \in y \equiv ((z = \emptyset) \vee (\exists a^1, a^2 \in x \exists b^1, b^2.\phi(a^1, b^1) \wedge \phi(a^2, b^2) \wedge \exists v \in b^1 \exists w \in b^2.z = \langle v, w \rangle))$$

Using extensionality we know for all x there is at most one y such that $\phi(x, y)$ holds. Note $\phi(\emptyset, \{\emptyset\})$ holds. For each finite ordinal $n \in \omega$, assume $\phi(i, S_i)$ holds for $i \in n$. Then $\phi(n, S_n)$ holds where

$$S_n := \{\emptyset\} \cup \{\langle v, w \rangle | v, w \in S_0 \cup \cdots \cup S_{n-1}\}$$

One can prove by induction on ω that for all $n \in \omega$, S_n is the unique set such that $\phi(n, S_n)$. Using replacement, $\{S_0, S_1, \ldots, S_n, \ldots\}$ is a set.[1] We finally define S to be $\bigcup\{S_0, S_1, \ldots, S_n, \ldots\}$ and define $nil := \emptyset \in S$ and $cons(v, w) := \langle v, w \rangle \in S$ for $v, w \in S$. One can prove the usual induction principle for S using this definition.

Using similar techniques one can define recursive functions on an inductively defined domain such as S.

Direct support for specifying inductive definitions which fall back on the construction above would be useful. For example, one could specify S as

```
inductive S {
  nil:S.
  cons:S -> S -> S.
}
```

[1] This is an important application of the axiom of replacement.

which should define $S : \texttt{set}$, $nil : S$ and $cons : S \to S \to S$ and prove the important induction principles (hiding the definitions and proofs from the user).

6.4 Importing Libraries: MetaMath, Isabelle-HOL/ZF, HOL-light, HOL4, Mizar

One of the most important immediate concerns is the building of a large DeTSeT library by importing the libraries of existing systems.

Translating the library of MetaMath [22] should be relatively easy since MetaMath is fully formal and is based on ZFC. Note that MetaMath includes a construction of the real numbers. One should also be able to translate the Isabelle-ZF library in a fairly direct manner.

Mizar [2] has a very large library, but does not use a formal notion of a proof object. The Mizar checker reads text files (Mizar articles) and accepts or rejects these files based on criteria native to the code of Mizar. One could imagine writing an interpreter for Mizar articles which generates corresponding DeTSeT abbreviations. Mizar is based on an extension of ZFC (Tarski-Grothendieck set theory, which assumes every set is a member of a universe modelling ZFC). This extension will likely only prevent small parts of the Mizar library from being directly translated into DeTSeT.

Finally, using the encoding of simply typed λ-calculus in DeTSeT, a translation from Isabelle-HOL and the various members of the HOL family (HOL-light and HOL4) should be possible. There has already been work building proof terms for these libraries which allows the libraries of one system to be imported into the others (see for example [25, 21]).

6.5 Modules

We have described here how to encode mathematical information in a single, large signature of abbreviations. One would like to have a module system which allows several people to independently develop parts of the overall signature without concern about interference from other people developing a different part of the signature.

6.6 Encoding Texts

The library should be further extended by encoding mathematics from textbooks. Of particular interest are portions of Number Theory [14], Real Analysis [11, 4], Algebra [17], Topology [24], and Category Theory [19].

6.7 Natural Input and Output

Much work could be done with respect to input and output of DeTSeT terms and types. As an example, one could consider the input specification language PAM which is used in the system Scunak [8, 7]. Scunak was also able to interpret a LaTeX proof from a textbook as a specification language [9]. Optimally, one would like to be able to scan a mathematical text and obtain a corresponding signature of abbreviations (concepts and proofs). Certainly there must be interesting intermediate stages which could be studied at the moment. One possibility would be to allow proofs to be specified in scripts or articles similar to those used by Mizar [2].

In terms of output, the hope is that one could indicate which parts of a signature one wishes to output and a system could use this to write an article or even a book. The output representation could be either a typesetting language like LaTeX, a markup language like OmDoc [18] which could be used by ActiveMath [23], or a document language like PLATO [36, 20] which could be used to mediate between systems and editors.

6.8 Unification and Matching

One can define an erasure from the dependent types of DeTSeT to simple types. Using this erasure and by Currying away the product types corresponding to class types, one can easily adapt Huet's algorithm for higher-order pre-unification [16] to DeTSeT. However, it is probably worthwhile to develop algorithms for unification and matching directly for DeTSeT.

6.9 Proof Search

We need methods for finding proofs or filling gaps in proofs. This boils down to the following problem: Given a valid signature Σ and a valid context Γ and a type such that $\Gamma \vdash_\Sigma A : Type^i$ (for $i \in \{0, 1, 2\}$), find a term M such that $\Gamma \vdash_\Sigma M \sim M \uparrow A$.

A special case would involve creating an efficient algorithm for answering the question of whether there exists a "one-step" M such that $\Gamma \vdash_\Sigma M \sim M \uparrow A$. One would need to be precise about what "one-step" means. One easy possibility can be seen by considering imitations and projections from Huet's pre-unification algorithm [16]. Each imitation or projection partially instantiating a variable of a proof type can be considered as essentially "one-step" in partially filling a proof. A "one-step" completion of a proof (in a context Γ) could be an imitation or a projection where any newly introduced existential variables can be filled using variables from the context Γ. More than likely, however, one would also like to consider certain equality steps as "one-step" even if these do not correspond simply to an imitation or projection.

The idea of what "one-step" means is related to the issue of *granularity* (as studied in [31]) which is being studied in the tutoring context as part of the DiaLog project [5, 6]. This also relates to checking textbook proofs [9]. A step filling a gap in a proof would be at the appropriate level of granularity for a given student (or reader) if the type corresponding to the gap could be filled in "one-step" where "one-step" is defined relative to the student (reader) model. The head of an imitation step could in principle be any logical constant or abbreviation from Σ, but could also be restricted to a subset of constants and abbreviations. Thus, such a student (reader) model could be defined, at least in part, by giving a subset of abbreviations from Σ that the student (reader) is expected to "know."

6.10 Compact Representation

In the formulation of DeTSeT given here, terms which correspond to proof terms are β-normal, even though by proof irrelevance it is not necessary to compute β-normal forms to check if two proof terms are equal. (Any two proof terms of the same type are equal, regardless of their normal forms.) We could avoid this by distinguishing in the syntax of terms between proof terms and non-proof terms. This should make some proof terms smaller since we could share subproofs within a term more easily. (At the moment one can only share subproofs by making abbreviations.) One could also form more compact representation by allowing implicit arguments (as in Twelf [27]) or by extending the term structure to include admissible rules (such as replacing equals for equals). In any case, one must prove that the new term representation can be mapped to the original representation in a type preserving way.

6.11 Theory and Semantics

We have given an algorithmic formulation of DeTSeT. In the future, a theoretical formulation should be given. In such a formulation, the λ on terms should probably carry the type so that each term has at most one type. One could hopefully prove that the algorithmic formulation and theoretical formulation are equivalent (in an appropriate sense).

One should be able to define a semantics and prove a soundness and (hopefully) completeness results relating the theoretical formulation of the type theory with the given semantics. Such a semantics could be set theoretic (using ideas from Henkin models), category theoretic (using, for example, either slice categories or fibrations) or perhaps by using pers over models of simple type theory.

6.12 Relationship to first-order ZFC

The motivation for DeTSeT was to give a usable formulation of ZFC. The fact that DeTSeT is at least as expressive and proof-theoretically strong seems obvious from the formulation given here. We conjecture that DeTSeT is actually equivalent to ZFC in the following senses:

- There is a translation taking each M such that $\cdot \vdash_\Sigma M \sim M \uparrow \texttt{prop}$ to a first-order proposition M'.
- For every M such that $\cdot \vdash_\Sigma M \sim M \uparrow \texttt{prop}$, there is a term P such that $\cdot \vdash_\Sigma P \sim P \uparrow \texttt{pf}\ M$ iff M' is provable in ZFC.
- For every proposition N in the language of first-order set theory there is a DeTSeT term M such that $\cdot \vdash_\Sigma M \sim M \uparrow \texttt{prop}$ and such that $N \equiv M'$ is provable in ZFC.

This would also imply ZFC and DeTSeT have the same consistency strength.

6.13 Choice

The formulation of the axiom of choice is a bit awkward. An easier formulation would involve a choice operator (similar to those commonly used in HOL). However, this would seem to increase the proof strength of the theory beyond ZFC. The problem is that one could use the choice operator within the definition of a set, which does not seem to be possible in ZFC. The status of choice operators in DeTSeT (or extensions of DeTSeT) and the relation to ZFC should be clarified.

7 Conclusion

We have described DeTSeT: a form of ZFC set theory encoded in a dependent type theory with proof irrelevance. DeTSeT has all the power of first-order ZFC and so can be used as a foundation for mathematics. Unlike first-order ZFC, one has the ability to directly construct sets such as

$$\{x \in A|\phi(x)\}$$

as formal terms, even when the property ϕ only makes sense for elements of A. In DeTSeT, sets (and classes) can be treated as types. Also, one can form functional relations using an "object-level" λ-abstraction operator. Thus, DeTSeT combines the advantage of having a single universe of all sets (as in first-order set theory) with the expressive power of logical systems based on typed λ-calculi. We hope to demonstrate the power and naturality of DeTSeT in the future by building a large library of formal mathematics as a DeTSeT signature.

References

[1] Serge Autexier and Armin Fiedler. Textbook proofs meet formal logic - the problem of underspecification and granularity. In Michael Kohlhase, editor, *MKM*, volume 3863 of *Lecture Notes in Computer Science*, pages 96–110. Springer, 2005.

[2] Grzegorz Bancerek and Piotr Rudnicki. A Compendium of Continuous Lattices in MIZAR. *J. Autom. Reason.*, 29(3-4):189–224, 2002.

[3] Henk Barendregt. Lambda calculi with types. In S. Abramsky, Dov M. Gabbay, and T.S.E. Maibaum, editors, *Handbook of Logic in Computer Science*, pages 117–309. Clarendon Press, 1992.

[4] R.G. Bartle and D.R. Sherbert. *Introduction to Real Analysis*. John Wiley & Sons, New York, 1982.

[5] Christoph Benzmüller, Armin Fiedler, Malte Gabsdil, Helmut Horacek, Ivana Kruijff-Korbayová, Manfred Pinkal, Jörg Siekmann, Dimitra Tsovaltzi, Bao Quoc Vo, and Magdalena Wolska. A wizard of oz experiment for tutorial dialogues in mathematics. In *Proceedings of AI in Education (AIED 2003) Workshop on Advanced Technologies for Mathematics Education*, Sydney, Australia, 2003.

[6] Christoph Benzmüller, Helmut Horacek, Henri Lesourd, Ivana Kruijff-Korbayova, Marvin Schiller, and Magdalena Wolska. A corpus of tutorial dialogs on theorem proving; the influence of the presentation of the study-material. In *Proceedings of International Conference on Language Resources and Evaluation (LREC 2006)*, Genova, Italy, 2006. ELDA. To appear.

[7] Chad E. Brown. Combining Type Theory and Untyped Set Theory. In Furbach and Shankar [13], pages 205–219.

[8] Chad E. Brown. Encoding functional relations in Scunak. In *LFMTP'2006*, September 2006.

[9] Chad E. Brown. Verifying and Invalidating Textbook Proofs using Scunak. In *Mathematical Knowledge Management, MKM 2006*, pages 110–123, Wokingham, England, 2006.

[10] N .G. de Bruijn. A survey of the project AUTOMATH. In J. P. Seldin and J. R. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism*, pages 579–606. Academic Press, 1980.

[11] J. Dieudonne. *Foundations of Modern Analysis*, volume 1. Academic Press, New York, 1960.

[12] Conal M. Elliot. *Extensions and Applications of Higher-order Unification.* PhD thesis, Carnegie Mellon University, 1990.

[13] Ulrich Furbach and Natarajan Shankar, editors. *Automated Reasoning, Third International Joint Conference, IJCAR 2006*, volume 4130 of *Lecture Notes in Artificial Intelligence*, Seattle, Washington, 2006. Springer-Verlag.

[14] Godfrey H. Hardy and E. M. Wright. *An introduction to the theory of numbers.* Oxford University Press, 5th edition, 1979.

[15] John Harrison. HOL light: A tutorial introduction. In Mandayam Srivas and Albert Camilleri, editors, *Proceedings of the First International Conference on Formal Methods in Computer-Aided Design (FMCAD'96)*, volume 1166 of *Lecture Notes in Computer Science*, pages 265–269. Springer-Verlag, 1996.

[16] Gérard P. Huet. A unification algorithm for typed λ-calculus. *Theoretical Computer Science*, 1:27–57, 1975.

[17] Thomas W. Hungerford. *Algebra.* Springer-Verlag New York, Inc., 1974.

[18] Michael Kohlhase. Omdoc: Towards an internet standard for the administration, distribution, and teaching of mathematical knowledge. In John A. Campbell and Eugenio Roanes-Lozano, editors, *Artificial Intelligence and Symbolic Computation: International Conference AISC 2000*, volume 1930 of *Lecture Notes in Artificial Intelligence*, pages 32–52. Springer-Verlag, 2001.

[19] Saunders MacLane. *Categories for the Working Mathematician.* Springer-Verlag, second edition, 1998.

[20] Christoph Benzmüller Marc Wagner, Serge Autexier. Plato: A mediator between text-editors and proof assistance systems. In Christoph Benzmüller Serge Autexier, editor, *7th Workshop on User Interfaces for Theorem Provers (UITP'06)*, ENTCS. Elsevier, august 2006.

[21] Sean McLaughlin. An Interpretation of Isabelle/HOL in HOL Light. In Furbach and Shankar [13], pages 192–204.

[22] Norman Megill. Metamath Home Page. `http://au.metamath.org/index.html`.

[23] Erica Melis. ActiveMath Home Page. `http://www.activemath.org/`.

[24] James R. Munkres. *Topology: A First Course*. Prentice-Hall, 1975.

[25] Steven Obua and Sebastian Skalberg. Importing HOL into Isabelle/HOL. In Furbach and Shankar [13], pages 298–302.

[26] Lawrence C. Paulson. *Isabelle: A Generic Theorem Prover*, volume 828 of *Lecture Notes in Computer Science*. Springer Verlag, 1994.

[27] Frank Pfenning and Carsten Schürmann. System Description: Twelf–A Meta-Logical Framework for Deductive Systems. In Harald Ganzinger, editor, *Proceedings of the 16th International Conference on Automated Deduction*, volume 1632 of *Lecture Notes in Artificial Intelligence*, pages 202–206, Trento, Italy, 1999. Springer-Verlag.

[28] The QED manifesto. In Alan Bundy, editor, *Proceedings of the 12th International Conference on Automated Deduction*, volume 814 of *Lecture Notes in Artificial Intelligence*, pages 238–251, Nancy, France, 1994. Springer-Verlag.

[29] Jason Reed. Proof irrelevance and strict definitions in a logical framework. Technical Report 02-153, School of Computer Science, Carnegie Mellon University, 2002.

[30] Herman Rubin and Jean E. Rubin. *Equivalents of the Axiom of Choice, II*. North-Holland, 1985.

[31] Marvin Schiller. Mechanizing Proof Step Evaluation for Mathematics Tutoring - the Case of Granularity. Master's thesis, Universität des Saarlandes, 2005.

[32] Jörg Siekmann, Christoph Benzmüller, Vladimir Brezhnev, Lassaad Cheikhrouhou, Armin Fiedler, Andreas Franke, Helmut Horacek, Michael Kohlhase, Andreas Meier, Erica Melis, Markus Moschner, Immanuel Normann, Martin Pollet, Volker Sorge, Carsten Ullrich, Claus-Peter Wirth, and Jürgen Zimmer. Proof development with Ωmega. In Andrei Voronkov, editor, *Proceedings of the 18th International Conference on Automated Deduction*, volume 2392 of *Lecture Notes in Artificial Intelligence*, pages 144–149, Copenhagen, Denmark, 2002. Springer-Verlag.

[33] Jörg Siekmann, Christoph Benzmüller, Armin Fiedler, Andreas Meier, Immanuel Normann, and Martin Pollet. Proof development in OMEGA: The irrationality of square root of 2. In Fairouz Kamareddine, editor, *Thirty Five Years of Automating Mathematics*, Kluwer Applied Logic series (28), pages 271–314. Kluwer Academic Publishers, 2003. ISBN 1-4020-1656-5.

[34] D.T. van Daalen. Verzamelingstheorie, de axioma's van Zermelo-Fraenkel. Technical report, Department of Mathematics, Eindhoven University of Technology, September 1970. Internal Report, Dutch.

[35] Jean van Heijenoort. *From Frege to Gödel. A Source Book in Mathematical Logic 1879–1931.* Harvard University Press, Cambridge, Massachusetts, 1967.

[36] Marc Wagner. Mediation between text-editors and proof assistance systems. Diploma thesis, Saarland University, Saarbrücken, Germany, 2006.

[37] Freek Wiedijk. Is ZF a hack? Comparing the complexity of some (formalist interpretations of) foundational systems for mathematics. *Journal of Applied Logic*, 2005. to appear.

[38] Ernst Zermelo. Untersuchungen über die Grundlagen der Mengenlehre I. *Mathematische Annalen*, 65:261–281, 1908. English translation, "Investigations in the foundations of set theory" in [35], pages 199–215.

Index