A THEORY SUPPORTING HIGHER-ORDER ABSTRACT SYNTAX

CHAD E. BROWN

Abstract. We describe an intuitionistic extensional higher-order theory supporting reasoning about syntax. Binders are represented using higher-order abstract syntax. The theory has no standard set theoretic model, but we give a nonstandard model using actions by a monoid of substitutions. The theory has been published in the Proofgold blockchain. Conjectures in this theory can be used to request a piece of syntax satisfying a certain property. We give examples in which two untyped λ-terms are requested via conjectures.

1. Introduction

We present a theory in a simply typed intuitionistic extensional higher-order logic [6] supporting higher-order abstract syntax [13]. The theory has an M-set model where M is the monoid of substitutions (up to σ-equality) [1, 7]. M-sets as models for simple type theory were described in [3]. That taking M to be the monoid of substitutions gives a model for classical nonextensional higher-order logic has been known since then, but unpublished. One can find some information in [20].

The M-set model is essentially a special case of a presheaf category. Hofmann also considered presheaf semantics for higher-order abstract syntax [11]. We leave the clarification of the precise relationship between Hofmann's presheaf models and the M-set model described here to future work.

The higher-order theory is given in Section 2. A representation of untyped λ-calculus in the theory is described in Section 3. In Section 4 we describe the published formal counterpart in the Proofgold blockchain. An M-set model of the theory is given in Section 5 along with a proof of a basic soundness result. Related work is considered in Section 6 and we conclude in Section 7.

2. A Higher-Order Theory of Syntax

We begin by defining a set $\mathcal{T}$ of *simple types*:

- ι : This is a base type which will (eventually) be interpreted as untyped λ-terms (in de Bruijn representation).
- o : This is a base type of propositions.
- $\alpha\beta$: This is the type of functions from α to β.

We next define a family of simply typed terms. For each type $\alpha \in \mathcal{T}$, let $\mathcal{V}_\alpha$ be a countably infinite set of variables of type α. We define a set Λ_α of *terms of type* α as follows:

Date: August 19, 2020.

Czech Technical University in Prague.

$$\frac{}{\Gamma \vdash s}\ s \in \mathcal{A} \qquad \frac{}{\Gamma \vdash s}\ s \in \Gamma \qquad \frac{\Gamma \vdash s}{\Gamma \vdash t}\ s{\approx}t \qquad \frac{\Gamma, s \vdash t}{\Gamma \vdash s \to t} \qquad \frac{\Gamma \vdash s \to t \qquad \Gamma \vdash s}{\Gamma \vdash t}$$

$$\frac{\Gamma \vdash s}{\Gamma \vdash \forall x.s}\ x \in \mathcal{V}_\alpha \setminus \mathcal{F}\Gamma \qquad \frac{\Gamma \vdash \forall x.s}{\Gamma \vdash s^x_t}\ x \in \mathcal{V}_\alpha, t \in \Lambda_\alpha$$

$$\frac{\Gamma \vdash sx = tx}{\Gamma \vdash s = t}\ x \in \mathcal{V}_\alpha \setminus (\mathcal{F}\Gamma \cup \mathcal{F}s \cup \mathcal{F}t) \text{ AND } s, t \in \Lambda_{\alpha\beta}$$

FIGURE 1. Proof Calculus for Intuitionistionistic HOL

- For each variable x of type α, $x \in \Lambda_\alpha$.
- $\mathsf{B} \in \Lambda_{(\iota\iota)\iota}$. (This is a constant which binds an object variable.)
- $\mathsf{P} \in \Lambda_{\iota\iota\iota}$. (This is a constant which forms a pair from two pieces of syntax.)
- If $s \in \Lambda_{\alpha\beta}$ and $t \in \Lambda_\alpha$, then $(st) \in \Lambda_\beta$.
- If x is a variable of type α and $s \in \Lambda_\beta$, then $(\lambda x.s) \in \Lambda_{\alpha\beta}$.
- If $s, t \in \Lambda_o$, then $(s \to t) \in \Lambda_o$.
- If x is a variable of type α and $s \in \Lambda_o$, then $(\forall x.s) \in \Lambda_o$.

We use common conventions to omit parentheses. We sometimes include annotations on λ and $\forall$ bound variables (e.g., $\lambda x : \alpha.s$ and $\forall y : \beta.s$) to indicate the type of the variable. We define $\mathcal{F}s$ to be the free variables of s and for sets A of terms we define $\mathcal{F}A$ to be $\bigcup_{s \in A} \mathcal{F}s$. We assume a capture avoiding substition s^x_t is defined. Terms of type o are called *propositions*. A *sentence* is a proposition with no free variables.

The only built-in logical connective is implication ($\to$) and the only built-in quantifier is the universal quantifier ($\forall$). In the context of higher-order logic it is well-known how to define the remaining logical constructs in a way that respects their intuitionistic meaning. In each case we use an impredicative definition that traces its roots to Russell [15] and Prawitz [14]. We define $\bot$ to be the proposition $\forall p : o.p$ where x is a variable of type o. We write $\neg s$ for $s \to \bot$. We define $\wedge$ to be $\lambda qr : o.\forall p : o.(q \to r \to p) \to p$ and write $s \wedge t$ for $(\wedge s)t$. We define $\vee$ to be $\lambda qr : o.\forall p : o.(q \to p) \to (r \to p) \to p$ and write $s \vee t$ for $(\vee s)t$. For each type α we use $\exists x : \alpha.s$ as notation for $\forall p : o.(\forall x : \alpha.s \to p) \to p$ where p is not x and is not free in s. For equality we write $s = t$ (where s and t are type α) as notation for $\forall p : \alpha\alpha o.pst \to pts$ where p is neither free in s nor t. This is a modification of Leibniz equality which we will call *symmetric Leibniz equality*. We write $s \neq t$ to mean $(s = t) \to \bot$.

We define a $(\beta\eta)$ conversion relation on terms of the same type in two steps. We first define $s{\Rightarrow_1}t$ if s β or η reduces to t in one step. We then define $s{\approx}t$ to be the reflexive, symmetric, transitive closure of $\Rightarrow_1$.

Let $\mathcal{A}$ be a set of sentences intended to be axioms of a theory. A natural deduction system for intuitionistic higher-order logic with functional extensionality and axioms $\mathcal{A}$ is given by Figure 1. In particular the rules define when $\Gamma \vdash s$ holds where Γ is a

finite set of propositions and s is a proposition. Aside from the treatment of functional extensionality, this is the same as the natural deduction calculus described in [4].[1]

For our theory of syntax, we include four axioms in $\mathcal{A}$:

- Pairing is injective: $\forall xyzw : \iota.\mathsf{P}xy = \mathsf{P}zw \to x = z \wedge y = w$.
- Binding is injective: $\forall fg : \iota\iota.\mathsf{B}f = \mathsf{B}g \to f = g$.
- Binding and pairing give distinct values: $\forall xy : \iota.\forall f : \iota\iota.\mathsf{P}xy \neq \mathsf{B}f$.
- Propositional extensionality: $\forall pq : o.(p \to q) \to (q \to p) \to p = q$.

It is not clear if this logic has a model at all. Since pairs are different from binders, there is no trivial model with ι interpreted as a singleton. However, the axiom that B is injective means B must be an injection from the function type $\iota\iota$ to ι. Taken together, we conclude there is no standard set-theoretic model since this would contradict a form of Cantor's Theorem. We will give the intended semantics as an M-set model in Section 5.

3. Representing Untyped Lambda Calculus

We can embed many syntactic constructs into the theory by building on top of the basic pairing and binding operators. For example, we could embed untyped λ-calculus by taking P to represent application and B to represent λ-abstraction. Instead of adopting this simple approach, we will use tagged pairs when representing application and λ-abstraction, so that there will still be infinitely many pieces of syntax that do not represent untyped λ-terms. To do this we will need one tag, so let us define nil to be $\mathsf{B}(\lambda x.x)$. Now we can define $\mathsf{A} : \iota\iota\iota$ to be $\lambda xy : \iota.\mathsf{P}\ \mathsf{nil}\ (\mathsf{P}\ x\ y)$ and define $\mathsf{L} : (\iota\iota)\iota$ to be $\lambda f : \iota\iota.\mathsf{P}\ \mathsf{nil}\ (\mathsf{B}\ f)$. It is easy to prove A and L are both injective and give distinct values.

We can now impredicatively define the set of untyped λ-terms relative to a set $\mathcal{G}$ (intended to be the set of possible free variables) as follows. Let us write $(\mathcal{G}, x)$ for the term $\lambda y : \iota.\mathcal{G}\ y \vee y = x$. Here $\mathcal{G}$ has type ιo while x and y have type ι (and are different). We will define $\mathsf{Ter} : (\iota o)\iota o$ so that Ter is the least relation satisfying three conditions:

- $\forall \mathcal{G} : \iota o.\forall y : \iota.\mathcal{G}y \to \mathsf{Ter}\ \mathcal{G}\ y$,
- $\forall \mathcal{G} : \iota o.\forall f : \iota\iota.(\forall x : \iota.\mathsf{Ter}\ (\mathcal{G}, x)\ (fx)) \to \mathsf{Ter}\ \mathcal{G}\ (\mathsf{L}\ f)$ and
- $\forall \mathcal{G} : \iota o.\forall yz : \iota.\mathsf{Ter}\ \mathcal{G}\ y \to \mathsf{Ter}\ \mathcal{G}\ z \to \mathsf{Ter}\ \mathcal{G}\ (\mathsf{A}\ y\ z)$.

Technically, the impredicative definition of Ter is given as

$$\begin{aligned}\mathsf{Ter} \quad := \quad & \lambda \mathcal{G} : \iota o.\lambda x : \iota.\forall p : (\iota o)\iota o. \\ & (\forall \mathcal{G} : \iota o.\forall y : \iota.\mathcal{G}y \to p\ \mathcal{G}\ y) \\ & \to (\forall \mathcal{G} : \iota o.\forall f : \iota\iota.(\forall x : \iota.p\ (\mathcal{G}, x)\ (fx)) \to p\ \mathcal{G}\ (\mathsf{L}\ f)) \\ & \to (\forall \mathcal{G} : \iota o.\forall yz : \iota.p\ \mathcal{G}\ y \to p\ \mathcal{G}\ z \to p\ \mathcal{G}\ (\mathsf{A}\ y\ z)) \\ & \to p\ \mathcal{G}\ x.\end{aligned}$$

[1] Adding Curry-Howard style checkable proof terms to such a calculus is well-understood and we do not dwell on this here [17]. Proofs published in Proofgold documents are given by such proof terms.

We can similarly define one-step β-reduction (relative to a set of variables) as follows:

$$\begin{aligned}\mathsf{Beta}_1 \;:=\; & \lambda\mathcal{G} : \iota o.\lambda xy : \iota.\forall r : (\iota o)\iota\iota o.\\ & (\forall\mathcal{G} : \iota o.\forall f : \iota\iota.\forall z.(\forall x.\mathsf{Ter}\ (\mathcal{G}, x)\ (f x)) \to \mathsf{Ter}\ \mathcal{G} z \to r\ \mathcal{G}\ (\mathsf{A}\ (\mathsf{L}\ f)\ z)\ (f z))\\ & \to (\forall\mathcal{G} : \iota o.\forall fg : \iota\iota.(\forall z.r\ (\mathcal{G}, z)\ (f z)(g z)) \to r\ \mathcal{G}\ (\mathsf{L}\ f)\ (\mathsf{L}\ g))\\ & \to (\forall\mathcal{G} : \iota o.\forall xyz.r\ \mathcal{G}\ x\ z \to \mathsf{Ter}\ \mathcal{G} y \to r\ \mathcal{G}\ (\mathsf{A}\ x\ y)\ (\mathsf{A}\ z\ y))\\ & \to (\forall\mathcal{G} : \iota o.\forall xyz.r\ \mathcal{G}\ y\ z \to \mathsf{Ter}\ \mathcal{G} x \to r\ \mathcal{G}\ (\mathsf{A}\ x\ y)\ (\mathsf{A}\ x\ z))\\ & \to r\ \mathcal{G}\ x\ y.\end{aligned}$$

We can then define $\mathsf{BetaE}\ \mathcal{G}$ to be the least equivalence relation (relative to the domain $\mathsf{Ter}\ \mathcal{G}$) containing $\mathsf{Beta}_1\ \mathcal{G}$. We omit the details here.

These definitions give us sufficient material to make conjectures that ask for certain kinds of untyped λ-terms. Let $\emptyset$ be notation for the term $\lambda x : \iota.\bot$ (representing the empty set of variables). Consider the following sentences:

$$\exists F : \iota.\mathsf{Ter}\ \emptyset\ F \wedge \forall x : \iota.\mathsf{BetaE}\ (\emptyset, x)\ (\mathsf{A}\ F\ x)\ x \tag{1}$$

$$\exists Y : \iota.\mathsf{Ter}\ \emptyset\ Y \wedge \forall f : \iota.\mathsf{BetaE}\ (\emptyset, f)\ (\mathsf{A}\ Y\ f)\ (\mathsf{A}\ f\ (\mathsf{A}\ Y\ f)) \tag{2}$$

Sentence (1) asserts the existence of an identity combinator while sentence (2) asserts the existence of a fixed point combinator. In order to prove each sentence a combinator with the right property must be given as a witness and then be proven to have the property.[2] In the next section we describe how these sentences were published as conjectures (with bounties) in the Proofgold network. The solutions were then published as two theorems (with proofs). The solutions contain the witnesses: $\mathsf{L}(\lambda x.x)$ for (1) and the famous Y-combinator

$$\mathsf{L}(\lambda f.\mathsf{A}(\mathsf{L}(\lambda x.\mathsf{A}\ f\ (\mathsf{A}\ x\ x)))(\mathsf{L}(\lambda x.\mathsf{A}\ f\ (\mathsf{A}\ x\ x))))$$

for (2).

4. Theory of Syntax in Proofgold

Proofgold[3] is a cryptocurrency network with support for formal logic and mathematics. At the core of Proofgold is a proof checker for intuitionistic higher-order logic with functional extensionality. On top of this users can publish *theories*. A theory consists of a finite number of primitive constants along with their types and a finite number of sentences as axioms. A theory is uniquely identified by its 256-bit identifier given by the Merkle root of the theory (seen as a tree). After a theory has been published, documents can be published in the theory. Documents can define new objects (using primitives or previously defined objects), prove new theorems and make new conjectures. In this section we briefly describe the publication of the theory of syntax from the previous section, the publication of documents giving the conjectures (1) and (2), and the publication of a document proving the conjectures.

When a theory is published the axioms are associated with public keys which are marked as the *owners* of the propositions. Likewise when a document proves a theorem

[2]Note that in a classical calculus, it would be sufficient to prove such an existential statement by proving it is impossible for a witness not to exist.

[3]https://prfgld.github.io

within a theory, a public key (associated with publisher of the document) is associated with the proven proposition. These are the only ways propositions can have declared owners. As a consequence it is possible to determine if a proposition is known (either as an axiom or as a previously proven theorem) by checking if it has an owner. This method of keeping up with proven propositions by associating them with public keys was described in the Qeditas white paper [19] and was part of the Qeditas codebase.[4] Ownership of propositions also gives a way of redeeming bounties by proving conjectures. A bounty can be placed on an unproven proposition where this bounty can only be spent by the owner of a proposition (or the owner of the negated proposition). By publishing a document resolving the conjecture, the bounty proposition (or its negation) will become owned by public keys associated with the publisher of the document. After this the bounty can be claimed.

4.1. **The Proofgold Theory of Syntax.** We start by giving the theory in the form it was given to the Proofgold node software.[5] This consists of a file specifying the typed constants and axioms. Such a theory file that begins with the following line:

```
Theory
```

We next give a local name (e.g., `syn`) for the first base type:

```
Base syn
```

In principle a theory may use finitely many base types, and each would be given a local name in order by using multiple `Base` declarations.

We next declare the pairing operator to have type $\iota\iota\iota$ and binding operator to have type $(\iota\iota)\iota$. The syntax for types, terms and proofs consists of a prefix notation which we leave the reader to tease out of the following examples.

```
Prim pair : TpArr syn TpArr syn syn
Prim bind : TpArr TpArr syn syn syn
```

We next make four axiom declarations. We first declare pairing to be injective. The basic Proofgold syntax has special notation for equations (using `Eq` followed by the type), but does not have a special notation for conjunction. We could explicitly define conjunction and then use it here, but instead we expand away conjunction leaving the statement of the axiom in the form:

$$\forall xyzw : \iota.\mathsf{P}xy = \mathsf{P}zw \to \forall p : o.(x = z \to y = w \to p) \to p.$$

```
Axiom pair_inj : All x syn All y syn All z syn All w syn
 Imp Eq syn Ap Ap pair x y Ap Ap pair z w
 All p Prop Imp Imp Eq syn x z Imp Eq syn y w p p
```

We next declare the axiom stating binding is injective.

```
Axiom bind_inj : All f TpArr syn syn All g TpArr syn syn
 Imp Eq syn Ap bind f Ap bind g Eq TpArr syn syn f g
```

[4] A large part of Proofgold's code was inherited from the open source Qeditas project. More information about Qeditas is at https://iohk.io/en/projects/qeditas/.

[5] http://grid01.ciirc.cvut.cz/~chad/hoas/HOASthy.pfg

We do not have a special notation for negations of equations and we do not explicitly define $\bot$. Instead we give the axiom stating pairing and binding give distinct values in the following form:

$$\forall xy : \iota.\forall f : \iota\iota.\mathsf{P}xy = \mathsf{B}f \to \forall p : o.p.$$

```
Axiom pair_not_bind : All x syn All y syn All f TpArr syn syn
 Imp Eq syn Ap Ap pair x y Ap bind f All p Prop p
```

Finally we declare propositional extensionality as an axiom.

```
Axiom prop_ext : All p Prop All q Prop
 Imp Imp p q Imp Imp q p Eq Prop p q
```

Publishing a theory or document requires first publishing a commitment (after which the information cannot be changed), waiting at least 12 blocks (roughly 12 hours) and then publishing the theory or document. This commitment process was also described in the Qeditas white paper [19], although "commitments" were called "intentions" there. For this theory the commitment transaction[6] was published August 3, 2020, and the theory itself was published[7] was published August 4, 2020. The Merkle root identifying the theory is

```
513140056e2032628f48d11e221efe29892e9a03a661d3b691793524a5176ede
```

4.2. **Making the Proofgold Conjectures.** In order to conjecture (1) and (2) we first need to make formal counterparts of definitions from the previous section. These definitions were made in two documents within the syntax theory. Each such document file begins with the line:

```
Document 513140056e2032628f48d11e221efe29892e9a03a661d3b691793524a5176ede
```

This identifies that the file is a document and identifies the theory in which it should be interpreted. The rest of the file gives various declarations that either give local names, declare names for known objects or propositions, make (possibly new) definitions, give conjectures and prove (possibly new) theorems. Each declaration in the document file uses a prefix syntax like the one we saw in the theory file. Instead of showing such a document file, we instead show the relevant declarations in a Megalodon file. Megalodon is an interactive theorem prover (the successor to the Egal prover described in [4]) that can produce Proofgold readable files.

The first document[8] contains definitions for $\bot$, conjunction, disjunction, $\emptyset$ (at type ιo) and the adjoin operation we informally wrote as $(\mathcal{G}, x)$ in Section 3. Coq-style infix notations for conjunction and disjunction are declared. There are also other definitions and several theorems we omit here.[9]

```
Definition False : prop := forall p:prop, p.
...
Definition and : prop -> prop -> prop := fun A B:prop =>
  forall p:prop, (A -> B -> p) -> p.
```

[6] Txid: 32c23f32db657c22beca0b7819391a5e3c99a0ae4c579f4c364cfc8053744e9e

[7] Txid: 01b3eec88f5241523510883707e6bcb29d2cb3d28dbbfc11b1396ebc5ec48c09

[8] http://grid01.ciirc.cvut.cz/~chad/hoas/PfgHOASDoc1.mg

[9] The Megalodon syntax, like the Egal syntax before, is very inspired by Coq.

```
(* Unicode /\ "2227" *)
Infix /\ 780 left := and.
...
Definition or : prop -> prop -> prop := fun A B:prop =>
  forall p:prop, (A -> p) -> (B -> p) -> p.

(* Unicode \/ "2228" *)
Infix \/ 785 left := or.
...
Definition emp : set -> prop := fun x => False.
...
Definition adj : (set -> prop) -> set -> set -> prop
  := fun G x y => G y \/ y = x.
```

After this first document was written, Megalodon produced a Proofgold file which was then published into the Proofgold blockchain. After publication of the first document a second document[10] was created to define the embedding of untyped λ-calculus in the theory and make the relevant conjectures. The definitions and previously proven theorems from the first document were included in a preamble file for Megalodon so that these were available in the second document. When Megalodon produced the Proofgold version of the second document, it determined which parts of the preamble needed to be included in the second document to make it self-contained.

The second document defined nil, A and L as follows:

```
Definition nil : set := bind (fun x => x).
...
Definition ap : set -> set -> set := fun x y => pair nil (pair x y).
Definition lam : (set -> set) -> set := fun f => pair nil (bind f).
```

Then the definition of Ter was given impredicatively:

```
Definition ulamp : (set -> prop) -> set -> prop := fun G x =>
  forall p:(set -> prop) -> set -> prop,
       (forall G:set -> prop, forall y, G y -> p G y)
    -> (forall G:set -> prop, forall f:set -> set,
          (forall x, p (adj G x) (f x)) -> p G (lam f))
    -> (forall G:set -> prop, forall y z,
          p G y -> p G z -> p G (ap y z))
    -> p G x.
```

The definition of one-step β-reduction was also defined impredicatively:

```
Definition beta1 : (set -> prop) -> set -> set -> prop
  := fun G x y =>
  forall r:(set -> prop) -> set -> set -> prop,
       (forall G:set -> prop, forall f:set -> set, forall z,
         (forall x, ulamp (adj G x) (f x)) -> ulamp G z
```

[10] http://grid01.ciirc.cvut.cz/~chad/hoas/PfgHOASDoc2.mg

```
                -> r G (ap (lam f) z) (f z))
    -> (forall G:set -> prop, forall f g:set -> set,
         (forall z, r (adj G z) (f z) (g z)) -> r G (lam f) (lam g))
    -> (forall G:set -> prop, forall x y z,
                      r G x z -> ulamp G y -> r G (ap x y) (ap z y))
    -> (forall G:set -> prop, forall x y z,
                      r G y z -> ulamp G x -> r G (ap x y) (ap x z))
    -> r G x y.
```

A general notion of the equivalence relation closure of a relation (relative to a domain predicate) was defined and this was used to define β-equivalence.

```
Definition eqclos : (set -> prop) -> (set -> set -> prop)
   -> set -> set -> prop := fun dom q x y =>
   forall r:set -> set -> prop,
        (forall x y, q x y -> r x y)
     -> (forall x, dom x -> r x x)
     -> (forall x y, r x y -> r y x)
     -> (forall x y z, r x y -> r y z -> r x z)
     -> r x y.
...
Definition betaeq : (set -> prop) -> set -> set -> prop
    := fun G => eqclos (ulamp G) (beta1 G).
```

This is sufficient to write the sentences corresponding to (1) and (2). In Megalodon we make these conjectures by declaring them as theorems, but omitting the proof using the `Admitted` keyword. Such "theorems" will translate to conjectures in Proofgold documents.

```
Theorem ulam_id_ex : exists F, ulamp emp F
       /\ forall x, betaeq (adj emp x) (ap F x) x.
Admitted.

Theorem ulam_fp_ex : exists Y, ulamp emp Y
       /\ forall f, betaeq (adj emp f) (ap Y f) (ap f (ap Y f)).
Admitted.
```

In the corresponding Proofgold document file, the two conjectures are given as follows:

```
Conj ulam_id_ex : Ex x1 syn Ap Ap and ...
Conj ulam_fp_ex : Ex x1 syn Ap Ap and ...
```

Bounties were added to the file by hand as follows:[11]

```
Bounty ulam_id_ex 1 NoTimeout
Bounty ulam_fp_ex 2 NoTimeout
```

When the second document was published, the transaction publishing the transaction also placed bounties of 1 bar and 2 bars (respectively) on the conjectures. These bounties could be redeemed by proving the conjectures, as we did in the third document.

[11]There is something untrue here. In truth both bounties were accidentally placed on the first conjecture. For the purposes of explanation, we pretend this mistake was not made.

4.3. **Resolving the Proofgold Conjectures.** The third document[12] proved theorems corresponding to the two conjectures. We show parts of the proofs here with some commentary, but note that they make use of a number of theorems proven in the first two documents which we will not describe in detail.

We first prove that the identity term represented by $\mathsf{L}(\lambda x.x)$ is in $\mathsf{Ter}\ \mathcal{G}$ for every $\mathcal{G}$. The proof makes use of three previously proven results:

- `adjI2`: $\forall \mathcal{G} : \iota o.\forall x : \iota.(G, x)\ x$.
- `ulamp_var`: $\forall \mathcal{G} : \iota o.\forall x : \iota.\mathcal{G}\ x \to \mathsf{Ter}\ \mathcal{G}\ x$.
- `ulamp_lam`: $\forall \mathcal{G} : \iota o.\forall f : \iota\iota.(\forall x.\mathsf{Ter}\ (\mathcal{G}, x)\ (fx)) \to \mathsf{Ter}\ \mathcal{G}(\mathsf{L}\ f)$.

We begin by stating the theorem.

```
Theorem ulamp_id : forall G:set -> prop, ulamp G (lam (fun x => x)).
```

We start the proof by letting an arbitrary but fixed $\mathcal{G}$ be given.

```
let G.
```

We now apply `ulamp_lam` leaving us to prove $\forall x.\mathsf{Ter}\ (\mathcal{G}, x)\ (\lambda x.x)$.

```
apply ulamp_lam.
```

We let x be given and note that we need to prove $\mathsf{Ter}\ (\mathcal{G}, x)\ (\lambda x.x)$.

```
let x. prove ulamp (adj G x) x.
```

The proof is completed by applying `ulamp_var` and `adjI2`.

```
apply ulamp_var. apply adjI2.
Qed.
```

Next we prove the theorem corresponding to the first conjecture.

```
Theorem ulam_id_ex : exists F, ulamp emp F
      /\ forall x, betaeq (adj emp x) (ap F x) x.
```

Since the statement is existential we start using the `witness` tactic giving the obvious witness term $\mathsf{L}(\lambda x.x)$.

```
witness (lam (fun x => x)).
```

We now need to prove a conjunction. Applying a previously proven theorem `andI` allows us to split this into two subgoals. The first subgoal is immediately proven using `ulamp_id`, the theorem we just proved.

```
apply andI.
- prove ulamp emp (lam (fun x => x)). apply ulamp_id.
```

Proving the second subgoal requires proving $\forall x.\mathsf{BetaE}\ (\emptyset, x)\ (\mathsf{A}\ (\mathsf{L}\ (\lambda x.x))\ x)\ x$. We let x be given and state what we need to prove.

```
- let x.
  prove betaeq (adj emp x) (ap (lam (fun x => x)) x) x.
```

A previously proven theorem (`betaeq_beta`) allows us to conclude BetaE holds if the left hand side is a β-redex with the right hand side as its reduct.

```
  apply betaeq_beta (adj emp x) (fun x => x) x.
```

[12] http://grid01.ciirc.cvut.cz/~chad/hoas/PfgHOASDoc3.mg

This still leaves two subgoals verifying that we have well-formed λ-terms which are easily proven.

```
  + prove forall y, ulamp (adj (adj emp x) y) y.
    let y. apply ulamp_var. apply adjI2.
  + prove ulamp (adj emp x) x. apply ulamp_var. apply adjI2.
Qed.
```

The final proof in the third document corresponded to the second conjecture. We briefly discuss it here.

```
Theorem ulam_fp_ex : exists Y, ulamp emp Y
     /\ forall f, betaeq (adj emp f) (ap Y f) (ap f (ap Y f)).
```

We want to give the Y combinator as the witness, but we will first use local abbreviations to make it easier to state the Y combinator and intermediate properties.

```
set W : set -> set := fun f => lam (fun x => ap f (ap x x)).
set Y := lam (fun f => (ap (W f) (W f))).
witness Y.
```

The fact that Y satisfies the conjunction follows from a number of intermediate results. First we prove the following claim:

```
claim L1: forall G:set -> prop, forall f x,
               ulamp (adj (adj G f) x) (ap f (ap x x)).
```

We omit the proof here. We next prove the following claim whose proof we also omit:

```
claim L2: forall G:set -> prop, forall f, ulamp (adj G f) (W f).
```

We then apply `andI` to split the proof into two subgoals. The first subgoal requires checking Y is an untyped λ-term (with no free variables) and is straightforward. The second subgoal requires proving that $\mathsf{A}\ Y\ f$ is β-equivalent to $\mathsf{A}\ f\ (\mathsf{A}\ Y\ f)$, with f as the only free variable. This is also not difficult, but requires using symmetry and transitivity of β-equivalence (previously proven theorems) to give a few intermediate forms and argue the β-equivalence at each step. We omit further details.

After the third document was published, the public key of the publisher became the owner of the proven propositions. Consequently, the 3 bars of bounties could be (and were) collected from the propositions.

5. A Model of the Theory

We now turn to the description of the intended model of the theory.

5.1. **Untyped Lambda Terms and the Monoid.** The set $\hat{\Lambda}$ of untyped λ terms are given by the grammar

$$s, t ::= n|(st)|(\lambda s)$$

where n ranges over natural numbers (de Bruijn indices [5]). Substitutions (σ, τ) are functions from natural numbers to $\hat{\Lambda}$. Let M be the set of all substititutions.

Let $\uparrow \in M$ be the substitution taking each natural number n to the de Bruijn index $n+1$. Given a term a and substitution σ there is a substitution operation giving a term $a\sigma$ defined as follows: $n\sigma$ is $\sigma(n)$, $(st)\sigma$ is $((s\sigma)(t\sigma))$ and $(\lambda s)\sigma$ is $(\lambda(s\sigma'))$ where $\sigma' \in M$ is given by $\sigma'(0) = 0$ and $\sigma'(n+1) = \sigma(n) \uparrow$.

The substitution ε which maps each n to the term n is called the *identity substitution*. Given $\sigma, \tau \in M$, we can compose σ and τ to form $\sigma\tau$ by taking each n to the term $\tau(\sigma n)$ (where the application of τ to σn is via the substitution operation). It is easy to see that this operation on substitutions is associative and that ε is a two-sided identity for the operation. Hence M is a monoid.

An M-set is a set A with an operation taking $a \in A$ and $\sigma \in M$ to an element $a\sigma \in A$. This operation is called the *action* of the M-set. The action must satisfy two properties: $a\varepsilon = a$ for all $a \in A$ and $(a\sigma)\tau = a(\sigma\tau)$ for all $a \in A$ and $\sigma, \tau \in M$. We will interpret simple types as M-sets following [3].

A very easy example of an M-set is $\hat{\Lambda}$ itself. We take the action to be the substitution operation. Indeed this will be the interpretation of ι.

5.2. **The Model.** We define an M-set $\mathcal{D}_\alpha$ for each $\alpha \in \mathcal{T}$. As already mentioned we define $\mathcal{D}_\iota$ to be the M-set of untyped λ-terms $\hat{\Lambda}$ with the action given by the substitution operation.

We have many options for interpreting the type o of propositions. In [3] we considered two possibilities giving classical (nonextensional) models: $\mathcal{D}_o$ could be the two element set with a trivial action or $\mathcal{D}_o$ could be the power set of M with an action taking $X\sigma$ to $\{\tau | \sigma\tau \in X\}$. Here we take $\mathcal{D}_o$ to be a Heyting algebra given by a subset of the power set of M. This Heyting algebra corresponds to how truth values are interpreted in a presheaf topos.

To be specific we take $\mathcal{D}_o$ to be the set of right ideals of M. A set $X \subseteq M$ is a *right ideal* if $\sigma\tau \in X$ whenever $\sigma \in X$ and $\tau \in M$. Note that two specific right ideals are the empty set and M. In fact, arbitrary intersections and arbitrary unions of right ideals are right ideals. This is enough to know the right ideals form a Heyting algebra. The action on $\mathcal{D}_o$ is given by taking $X\sigma$ to be $\{\tau | \sigma\tau \in X\}$. It is easy to see that $X\sigma$ is a right ideal: If $\sigma\tau \in X$ and $\mu \in M$, then $\sigma\tau\mu \in X$ and so $\tau\mu \in X\sigma$.

Finally we interpret function types. Assume $\mathcal{D}_\alpha$ and $\mathcal{D}_\beta$ are M-sets. We take $\mathcal{D}_{\alpha\beta}$ to be the M-set

$$\{f : M \times \mathcal{D}_\alpha \to \mathcal{D}_\beta | \forall \sigma\tau \in M. \forall a \in \mathcal{D}_\alpha. f(\sigma, a)\tau = f(\sigma\tau, a\tau)\}.$$

The action is given by taking $f\mu$ to be $f\mu(\sigma, a) = f(\mu\sigma, a)$.

We next need to interpret the simply typed terms to be elements of the corresponding M-set. Two specific terms we will need to interpret are the constants P and B.

Let $\hat{\mathsf{P}}$ be the function given by $\hat{\mathsf{P}}(\sigma, s)(\tau, t) = ((s\tau)t)$. It is easy to check $\hat{\mathsf{P}}$ is in $\mathcal{D}_{\iota\iota\iota}$ and $\hat{\mathsf{P}}\mu = \hat{\mathsf{P}}$.

We take $\hat{\mathsf{B}}$ to be the function taking $\hat{\mathsf{B}}(\sigma, f) = (\lambda f(\uparrow, 0))$ for each $\sigma \in M$ and $f \in \mathcal{D}_{\iota\iota}$. It is easy to check $\hat{\mathsf{B}} \in \mathcal{D}_{(\iota\iota)\iota}$ and $\hat{\mathsf{B}}\mu = \hat{\mathsf{B}}$.

In order to prove soundness for this interpretation of B we will need to know the function sending $f \in \mathcal{D}_{\iota\iota}$ to $\hat{\mathsf{B}}(\sigma, f)$ is injective. We prove this fact here.

Lemma 5.1. *Let $f, g \in \mathcal{D}_{\iota\iota}$ and $\sigma \in M$ be given. If $\hat{\mathsf{B}}(\sigma, f) = \hat{\mathsf{B}}(\sigma, g)$, then $f = g$.*

Proof. Suppose $\hat{\mathsf{B}}(\sigma, f) = \hat{\mathsf{B}}(\sigma, g)$. This means $(\lambda f(\uparrow, 0)) = (\lambda g(\uparrow, 0))$. This means $f(\uparrow, 0)$ and $g(\uparrow, 0)$ are the same untyped λ-term. We will prove $f = g$. Let $\sigma \in M$ and

$a \in \mathcal{D}_\iota$. Let $a :: \sigma$ be the substitution sending 0 to a and $n+1$ to $\sigma(n)$. Since $f, g \in \mathcal{D}_{\iota\iota}$ we know

$$f(\sigma, a) = f(\uparrow (a :: \sigma), 0(a :: \sigma)) = f(\uparrow, 0)(a :: \sigma) = g(\uparrow, 0)(a :: \sigma)$$
$$= g(\uparrow (a :: \sigma), 0(a :: \sigma)) = g(\sigma, a)$$

as desired. □

We also prove an injectivity style result for $\hat{\mathsf{P}}$.

Lemma 5.2. *Let $a, b, c, d \in \mathcal{D}_\iota$ and $\sigma, \tau \in M$ be given. If $\hat{\mathsf{P}}(\sigma, a)(\tau, b) = \hat{\mathsf{P}}(\sigma, c)(\tau, d)$, then $a\tau = c\tau$ and $b = d$.*

Proof. Assume $\hat{\mathsf{P}}(\sigma, a)(\tau, b) = \hat{\mathsf{P}}(\sigma, c)(\tau, d)$, i.e., $(a\tau)b$ and $(c\tau)d$ are the same untyped λ-term. Thus $a\tau = c\tau$ and $b = d$ as desired. □

We are now in a position to define the evaluation function for simply typed terms. Since we need to interpret variables, the evaluation function will depend on an assignment sending each variable $x \in \mathcal{V}_\alpha$ to an element of $\mathcal{D}_\alpha$. As in [3] the evaluation function also depends on an element of the monoid M. Given an assignment φ and substitution $\sigma \in M$, we let $\varphi\sigma$ be the assignment taking x to $\varphi(x)\sigma$ (i.e., we can act on assignments). Also, given an assignment φ, a variable $x \in \mathcal{V}_\alpha$ and an element $a \in \mathcal{D}_\alpha$, we let φ^x_a be the substitution which sends x to a and each other y to $\varphi(y)$.

We will denote the evaluation function by $[\![s]\!]^\sigma_\varphi$ where $s \in \Lambda_\alpha$, $\sigma \in M$ and φ is an assignment. We define it by giving the following equations.

$$[\![x]\!]^\sigma_\varphi = \varphi x$$
$$[\![\mathsf{P}]\!]^\sigma_\varphi = \hat{\mathsf{P}}$$
$$[\![\mathsf{B}]\!]^\sigma_\varphi = \hat{\mathsf{B}}$$
$$[\![st]\!]^\sigma_\varphi = [\![s]\!]^\sigma_\varphi(\varepsilon, [\![t]\!]^\sigma_\varphi)$$
$$[\![\lambda x.s]\!]^\sigma_\varphi(\tau, a) = [\![s]\!]^{\sigma\tau}_{(\varphi\tau)^x_a}$$
$$[\![s \to t]\!]^\sigma_\varphi = \bigcup\{Z \in \mathcal{D}_o | \forall \mu \in M. \mu \in Z \to \varepsilon \in [\![s]\!]^{\sigma\mu}_{\varphi\mu} \to \varepsilon \in [\![t]\!]^{\sigma\mu}_{\varphi\mu}\}$$
$$[\![\forall x.s]\!]^\sigma_\varphi = \{\tau \in M | \forall \mu \in M. \forall a \in \mathcal{D}_\alpha. \varepsilon \in [\![s]\!]^{\sigma\tau\mu}_{(\varphi\tau\mu)^x_a}\} \text{ where } x \in \mathcal{V}_\alpha$$

We state a sequence of results with an indication of how to prove them. Proofs of analogous results (in a classical setting) can be found in Appendix A of [3].

Each of the next three lemmas is provable by induction on s.

Lemma 5.3. *If $\varphi(x) = \psi(x)$ for all $x \in \mathcal{F}s$, then $[\![s]\!]^\sigma_\varphi = [\![s]\!]^\sigma_\psi$.*

Lemma 5.4. $[\![s]\!]^\sigma_\varphi\tau = [\![s]\!]^{\sigma\tau}_{\varphi\tau}$.

Proof. The application and λ-abstraction cases follow from straightforward applications of the inductive hypotheses. Due to the formulations of interpretations of implication and universal quantifications, these cases do not require the use of the induction hypothesis. We only show the implication case. We need to prove $[\![s \to t]\!]^\sigma_\varphi\tau$ and $[\![s \to t]\!]^{\sigma\tau}_{\varphi\tau}$ are the same right ideals.

First assume $\mu \in [\![s \to t]\!]^\sigma_\varphi\tau$ and so $\tau\mu \in [\![s \to t]\!]^\sigma_\varphi$. There must be a right ideal Z such that $\tau\mu \in Z$ and $\forall \nu \in Z. \varepsilon \in [\![s]\!]^{\sigma\nu}_{\varphi\nu} \to \varepsilon \in [\![t]\!]^{\sigma\nu}_{\varphi\nu}$. We claim that $Z\tau$ is a right

ideal witnessing $\mu \in [\![s \to t]\!]^{\sigma\tau}_{\varphi\tau}$. Since $\tau\mu \in Z$ we know $\mu \in Z\tau$. Let $\nu \in Z\tau$ such that $\varepsilon \in [\![s]\!]^{\sigma\tau\nu}_{\varphi\tau\nu}$ be given. Since $\tau\nu \in Z$ we infer $\varepsilon \in [\![t]\!]^{\sigma\tau\nu}_{\varphi\tau\nu}$ as desired.

Next assume $\mu \in [\![s \to t]\!]^{\sigma\tau}_{\varphi\tau}$. There must be a right ideal Z such that $\mu \in Z$ and $\forall \nu \in Z.\varepsilon \in [\![s]\!]^{\sigma\tau\nu}_{\varphi\tau\nu} \to \varepsilon \in [\![t]\!]^{\sigma\tau\nu}_{\varphi\tau\nu}$. We need to prove $\mu \in [\![s \to t]\!]^{\sigma}_{\varphi}\tau$, i.e., $\tau\mu \in [\![s \to t]\!]^{\sigma}_{\varphi}$. Let Z' be $\{\xi \in M | \exists \nu \in Z.\xi = \tau\nu\}$. It is easy to check that Z' is a right ideal and $\tau\mu \in Z'$. Let $\xi \in Z'$ such that $\varepsilon \in [\![s]\!]^{\sigma\xi}_{\varphi\xi}$ be given. Let $\nu \in Z$ be such that $\xi = \tau\nu$ and so $\varepsilon \in [\![s]\!]^{\sigma\tau\nu}_{\varphi\tau\nu}$. We conclude $\varepsilon \in [\![t]\!]^{\sigma\tau\nu}_{\varphi\tau\nu}$ and so $\varepsilon \in [\![t]\!]^{\sigma\xi}_{\varphi\xi}$ as desired. □

Lemma 5.5. $[\![s^x_t]\!]^{\sigma}_{\varphi} = [\![s]\!]^{\sigma}_{\varphi^x_{[\![t]\!]^{\sigma}_{\varphi}}}$

Proof. We show the λ-abstraction case. The other cases are straightforward. Assume y is a variable of type α different from x and not free in t. We need to prove $[\![(\lambda y.s)^x_t]\!]^{\sigma}_{\varphi}$ and $[\![\lambda y.s]\!]^{\sigma}_{\varphi^x_{[\![t]\!]^{\sigma}_{\varphi}}}$ are the same functions. Let $\mu \in M$ and $a \in \mathcal{D}_\alpha$ be given. We begin by computing

$$[\![(\lambda y.s)^x_t]\!]^{\sigma}_{\varphi}(\mu, a) = [\![\lambda y.s^x_t]\!]^{\sigma}_{\varphi}(\mu, a) = [\![s^x_t]\!]^{\sigma\mu}_{(\varphi\mu)^y_a} = [\![s]\!]^{\sigma\mu}_{((\varphi\mu)^y_a)^x_{[\![t]\!]^{\sigma\mu}_{(\varphi\mu)^y_a}}}$$

where the last step is the result of the inductive hypothesis for s. Since y is not free in t, Lemma 5.3 allows us to infer that $[\![t]\!]^{\sigma\mu}_{(\varphi\mu)^y_a}$ equals $[\![t]\!]^{\sigma\mu}_{\varphi\mu}$. Hence we have

$$[\![(\lambda y.s)^x_t]\!]^{\sigma}_{\varphi}(\mu, a) = [\![s]\!]^{\sigma\mu}_{((\varphi\mu)^y_a)^x_{[\![t]\!]^{\sigma\mu}_{\varphi\mu}}}$$

Since y and x are distinct, the assignment $((\varphi\mu)^y_a)^x_{[\![t]\!]^{\sigma\mu}_{\varphi\mu}}$ can also be written as $((\varphi\mu)^x_{[\![t]\!]^{\sigma\mu}_{\varphi\mu}})^y_a$. Using Lemma 5.4 we can also write the assignment as $((\varphi^x_{[\![t]\!]^{\sigma}_{\varphi}})\mu)^y_a$. Hence we have

$$[\![(\lambda y.s)^x_t]\!]^{\sigma}_{\varphi}(\mu, a) = [\![s]\!]^{\sigma\mu}_{((\varphi^x_{[\![t]\!]^{\sigma}_{\varphi}})\mu)^y_a} = [\![\lambda y.s]\!]^{\sigma}_{\varphi^x_{[\![t]\!]^{\sigma}_{\varphi}}}(\mu, a)$$

as desired. □

The following results are easy to prove and justify soundness of conversion.

Lemma 5.6. $[\![(\lambda x.s)t]\!]^{\sigma}_{\varphi} = [\![s^x_t]\!]^{\sigma}_{\varphi}$

Lemma 5.7. *If $x \notin \mathcal{F}s$, then* $[\![\lambda x.sx]\!]^{\sigma}_{\varphi} = [\![s]\!]^{\sigma}_{\varphi}$

Lemma 5.8. *If $s \Rightarrow_1 t$, then* $[\![s]\!]^{\sigma}_{\varphi} = [\![t]\!]^{\sigma}_{\varphi}$

Lemma 5.9. *If $s \approx t$, then* $[\![s]\!]^{\sigma}_{\varphi} = [\![t]\!]^{\sigma}_{\varphi}$

Using Lemma 5.4 it becomes clear that we have the following alternative characterization of when a substitution is in the interpretation of an implication.

Lemma 5.10. *Let $s, t \in \Lambda_o$, $\sigma, \tau \in M$ and φ be an assignment. τ is in $[\![s \to t]\!]^{\sigma}_{\varphi}$ if and only if for all $\mu \in M$ $\tau\mu \in [\![s]\!]^{\sigma}_{\varphi}$ implies $\tau\mu \in [\![t]\!]^{\sigma}_{\varphi}$.*

Proof. Assume $\tau \in [\![s \to t]\!]^{\sigma}_{\varphi}$ and $\tau\mu \in [\![s]\!]^{\sigma}_{\varphi}$. Let Z be a right ideal such that $\tau \in Z$ and $\forall \nu \in Z.\varepsilon \in [\![s]\!]^{\sigma\nu}_{\varphi\nu} \to \varepsilon \in [\![t]\!]^{\sigma\nu}_{\varphi\nu}$. From $\tau\mu \in [\![s]\!]^{\sigma}_{\varphi}$ we obtain $\varepsilon \in [\![s]\!]^{\sigma\tau\mu}_{\varphi\tau\mu}$ by Lemma 5.4. Since $\tau \in Z$ and Z is a right ideal, we know $\tau\mu \in Z$ and so $\varepsilon \in [\![t]\!]^{\sigma\tau\mu}_{\varphi\tau\mu}$. Applying Lemma 5.4 again we have $\tau\mu \in [\![t]\!]^{\sigma}_{\varphi}$.

For the other direction suppose we know $\tau\mu \in [\![s]\!]^\sigma_\varphi$ implies $\tau\mu \in [\![t]\!]^\sigma_\varphi$ for all $\mu \in M$. Let Z be the right ideal given by $\{\tau\mu | \mu \in M\}$. It is straightforward to check Z witnesses $\tau \in [\![s \to t]\!]^\sigma_\varphi$. □

We next prove the following characterization of the interpretation of symmetric Leibniz equality.

Lemma 5.11. *Let $s, t \in \Lambda_\alpha$, $\sigma, \tau \in M$ and φ be an assignment. τ is in $[\![s = t]\!]^\sigma_\varphi$ if and only if $[\![s]\!]^\sigma_\varphi \tau = [\![t]\!]^\sigma_\varphi \tau$.*

Proof. For both directions fix p to be a variable of type $\alpha\alpha o$ free in neither s nor t. Recall that $s = t$ is $\forall p.pst \to pts$. For the first direction assume $\tau \in [\![s = t]\!]^\sigma_\varphi$. We need to prove $[\![s]\!]^\sigma_\varphi \tau = [\![t]\!]^\sigma_\varphi \tau$. Let $b \in \mathcal{D}_\alpha$ be $[\![t]\!]^\sigma_\varphi \tau$. Let $f : M \times \mathcal{D}_\alpha \to \mathcal{D}_o$ be defined by $f(\mu, a) = \{\nu \in M | a\nu = b\mu\nu\}$. It is clear that $f(\mu, a)$ is a right ideal and hence in $\mathcal{D}_o$. To verify $f(\mu, a)\xi = f(\mu\xi, a\xi)$ we note that $\nu \in f(\mu, a)\xi$ iff $\xi\nu \in f(\mu, a)$ iff $a\xi\nu = b\mu\xi\nu$ iff $\nu \in f(\mu\xi, a\xi)$. Hence $f \in \mathcal{D}_{\alpha o}$. Let $q : M \times \mathcal{D}_\alpha \to \mathcal{D}_{\alpha o}$ be defined by $q(\mu, a) = f\mu$. It is easy to see that $q \in \mathcal{D}_{\alpha\alpha o}$. From $\tau \in [\![s = t]\!]^\sigma_\varphi$ we infer $\varepsilon \in [\![pst \to pts]\!]^{\sigma\tau}_{(\varphi\tau)^p_q}$. Note that $[\![pst]\!]^{\sigma\tau}_{(\varphi\tau)^p_q}$ is $q(\varepsilon, [\![s]\!]^\sigma_\varphi \tau)(\varepsilon, b)$ (using Lemmas 5.4 and 5.3 and the definition of b). We compute

$$q(\varepsilon, [\![s]\!]^\sigma_\varphi \tau)(\varepsilon, b) = f(\varepsilon, b) = \{\nu \in M | b\nu = b\nu\}.$$

Hence $\varepsilon \in [\![pst]\!]^{\sigma\tau}_{(\varphi\tau)^p_q}$. By Lemma 5.10 we now know $\varepsilon \in [\![pts]\!]^{\sigma\tau}_{(\varphi\tau)^p_q}$. Note that $[\![pst]\!]^{\sigma\tau}_{(\varphi\tau)^p_q}$ is $q(\varepsilon, b)(\varepsilon, [\![s]\!]^\sigma_\varphi \tau)$. We compute

$$q(\varepsilon, b)(\varepsilon, [\![s]\!]^\sigma_\varphi \tau) = f(\varepsilon, [\![s]\!]^\sigma_\varphi \tau) = \{\nu \in M | [\![s]\!]^\sigma_\varphi \tau\nu = b\nu\}.$$

Since $\varepsilon \in q(\varepsilon, b)(\varepsilon, [\![s]\!]^\sigma_\varphi \tau)$ we conclude $[\![s]\!]^\sigma_\varphi \tau = b$ as desired.

For the other direction assume $[\![s]\!]^\sigma_\varphi \tau = [\![t]\!]^\sigma_\varphi \tau$. We need to prove $\tau \in [\![\forall p.pst \to pts]\!]^\sigma_\varphi$. Let $\mu \in M$ and $q \in \mathcal{D}_{\alpha\alpha o}$ be given. We need to prove $\varepsilon \in [\![pst \to pts]\!]^{\sigma\tau\mu}_{\varphi\tau\mu^p_q}$. We use Lemma 5.10 to prove this. Let $\nu \in M$ such that $\nu \in [\![pst]\!]^{\sigma\tau\mu}_{\varphi\tau\mu^p_q}$ be given. Using Lemma 5.3 we have

$$\nu \in q(\varepsilon, [\![s]\!]^{\sigma\tau\mu}_{\varphi\tau\mu})(\varepsilon, [\![t]\!]^{\sigma\tau\mu}_{\varphi\tau\mu}).$$

From $[\![s]\!]^\sigma_\varphi \tau = [\![t]\!]^\sigma_\varphi \tau$ and Lemma 5.4 we know $[\![s]\!]^{\sigma\tau\mu}_{\varphi\tau\mu} = [\![t]\!]^{\sigma\tau\mu}_{\varphi\tau\mu}$. Hence

$$\nu \in q(\varepsilon, [\![t]\!]^{\sigma\tau\mu}_{\varphi\tau\mu})(\varepsilon, [\![s]\!]^{\sigma\tau\mu}_{\varphi\tau\mu})$$

and so $\nu \in [\![pts]\!]^{\sigma\tau\mu}_{\varphi\tau\mu^p_q}$ as desired. □

Before moving on to the main soundness result, we first prove the four axioms are true in the model.

Lemma 5.12. $\varepsilon \in [\![\forall xyzw : \iota.\mathsf{P}xy = \mathsf{P}zw \to x = z \wedge y = w]\!]^\sigma_\varphi$.

Proof. Let $\mu_1, \mu_2, \mu_3, \mu_4 \in M$ and $a, b, c, d \in \mathcal{D}_\iota$ be given. Let ψ be

$$(((((((\varphi\mu_1)^x_a)\mu_2)^y_b)\mu_3)^z_c)\mu_4)^w_d$$

and note $\psi(x) = a\mu_2\mu_3\mu_4$, $\psi(y) = b\mu_3\mu_4$, $\psi(z) = c\mu_4$ and $\psi(w) = d$. We need to prove

$$\varepsilon \in [\![\mathsf{P}xy = \mathsf{P}zw \to x = z \wedge y = w]\!]^{\sigma\mu_1\mu_2\mu_3\mu_4}_\psi.$$

We will use Lemma 5.10. Let $\mu_5 \in M$ such that $\mu_5 \in [\![\mathsf{P}xy = \mathsf{P}zw]\!]_\psi^{\sigma\mu_1\mu_2\mu_3\mu_4}$ be given. By Lemma 5.11 we have $\hat{\mathsf{P}}(\varepsilon, a\mu_2\mu_3\mu_4)(\varepsilon, b\mu_3\mu_4)\mu_5 = \hat{\mathsf{P}}(\varepsilon, c\mu_4)(\varepsilon, d)\mu_5$ and so

$$\hat{\mathsf{P}}(\varepsilon, a\mu_2\mu_3\mu_4)(\mu_5, b\mu_3\mu_4\mu_5) = \hat{\mathsf{P}}(\varepsilon, c\mu_4)(\mu_5, d\mu_5).$$

By Lemma 5.2 we know $a\mu_2\mu_3\mu_4\mu_5 = c\mu_4\mu_5$ and $b\mu_3\mu_4\mu_5 = d\mu_5$. We need to prove $\mu_5 \in [\![x = z \wedge y = w]\!]_\psi^{\sigma\mu_1\mu_2\mu_3\mu_4}$, i.e., $\mu_5 \in [\![\forall p : o.(x = z \to y = w \to p) \to p]\!]_\psi^{\sigma\mu_1\mu_2\mu_3\mu_4}$. Let $\mu_6 \in M$ and $X \in \mathcal{D}_o$ be given. We need to prove

$$\varepsilon \in [\![(x = z \to y = w \to p) \to p]\!]_{(\psi\mu_5\mu_6)^p_X}^{\sigma\mu_1\mu_2\mu_3\mu_4\mu_5\mu_6}.$$

Let $\mu_7 \in M$ such that $\mu_7 \in [\![x = z \to y = w \to p]\!]_{(\psi\mu_5\mu_6)^p_X}^{\sigma\mu_1\mu_2\mu_3\mu_4\mu_5\mu_6}$ be given. We need to prove $\mu_7 \in [\![p]\!]_{(\psi\mu_5\mu_6)^p_X}^{\sigma\mu_1\mu_2\mu_3\mu_4\mu_5\mu_6}$, i.e., $\mu_7 \in X$. Since

$$(\psi\mu_5\mu_6)^p_X(x)\mu_7 = a\mu_2\mu_3\mu_4\mu_5\mu_6\mu_7 = c\mu_4\mu_5\mu_6\mu_7 = (\psi\mu_5\mu_6)^p_X(z)\mu_7$$

we know $\mu_7 \in [\![x = z]\!]_{(\psi\mu_5\mu_6)^p_X}^{\sigma\mu_1\mu_2\mu_3\mu_4\mu_5\mu_6}$. Since

$$(\psi\mu_5\mu_6)^p_X(y)\mu_7 = b\mu_3\mu_4\mu_5\mu_6\mu_7 = d\mu_5\mu_6\mu_7 = (\psi\mu_5\mu_6)^p_X(w)\mu_7$$

we know $\mu_7 \in [\![y = w]\!]_{(\psi\mu_5\mu_6)^p_X}^{\sigma\mu_1\mu_2\mu_3\mu_4\mu_5\mu_6}$. Hence $\mu_7 \in X$ as desired by applying Lemma 5.10 twice. □

Lemma 5.13. $\varepsilon \in [\![\forall fg : \iota\iota.\mathsf{B}f = \mathsf{B}g \to f = g]\!]_\varphi^\sigma$.

Proof. Let $\mu_1, \mu_2 \in M$ and $h, k \in \mathcal{D}_{\iota\iota}$ be given. We need to prove

$$\varepsilon \in [\![\mathsf{B}f = \mathsf{B}g \to f = g]\!]_{(((\varphi\mu_1)^f_h)\mu_2)^g_k}^{\sigma\mu_1\mu_2}.$$

We will use Lemma 5.10. Let $\mu_3 \in M$ such that $\mu_3 \in [\![\mathsf{B}f = \mathsf{B}g]\!]_{(((\varphi\mu_1)^f_h)\mu_2)^g_k}^{\sigma\mu_1\mu_2}$ be given. We need to prove $\mu_3 \in [\![f = g]\!]_{(((\varphi\mu_1)^f_h)\mu_2)^g_k}^{\sigma\mu_1\mu_2}$. Let ψ be $(((\varphi\mu_1)^f_h)\mu_2)^g_k$ and note that $\psi(f) = h\mu_2$ and $\psi(g) = k$. Using Lemma 5.11 we know $\hat{\mathsf{B}}(\varepsilon, h\mu_2)\mu_3 = \hat{\mathsf{B}}(\varepsilon, k)\mu_3$ and that it is enough to prove $h\mu_2\mu_3 = k\mu_3$. Since $\hat{\mathsf{B}}(\varepsilon, h\mu_2)\mu_3 = \hat{\mathsf{B}}(\varepsilon, k)\mu_3$, we know $\hat{\mathsf{B}}(\mu_3, h\mu_2\mu_3) = \hat{\mathsf{B}}(\mu_3, k\mu_3)$ and so $h\mu_2\mu_3 = k\mu_3$ by Lemma 5.1. □

Lemma 5.14. $\varepsilon \in [\![\forall xy : \iota.\forall f : \iota\iota.\mathsf{P}xy \neq \mathsf{B}f]\!]_\varphi^\sigma$.

Proof. Let $\mu_1, \mu_2, \mu_3 \in M$, $a, b \in \mathcal{D}_\iota$ and $h \in \mathcal{D}_{\iota\iota}$ be given. Let ψ be $(((((\varphi\mu_1)^x_a)\mu_2)^y_b)\mu_3)^f_h$ and note $\psi(x) = a\mu_2\mu_3$, $\psi(y) = b\mu_3$ and $\psi(f) = h$. We need to prove

$$\varepsilon \in [\![\mathsf{P}xy = \mathsf{B}f \to \bot]\!]_\psi^{\sigma\mu_1\mu_2\mu_3}.$$

We will use Lemma 5.10. Let $\mu_4 \in M$ such that $\mu_4 \in [\![\mathsf{P}xy = \mathsf{B}f]\!]_\psi^{\sigma\mu_1\mu_2\mu_3}$ be given. We will prove a contradiction. By Lemma 5.11 we must have $[\![\mathsf{P}xy]\!]_\psi^{\sigma\mu_1\mu_2\mu_3}\mu_4 = [\![\mathsf{B}f]\!]_\psi^{\sigma\mu_1\mu_2\mu_3}\mu_4$. Hence $\hat{\mathsf{P}}(\varepsilon, a\mu_2\mu_3)(\varepsilon, b\mu_3)\mu_4 = \hat{\mathsf{B}}(\varepsilon, h)\mu_4$. Thus the untyped λ-term $((a\mu_2\mu_3)(b\mu_3))\mu_4$ must equal $(\lambda h(\uparrow, 0))\mu_4$. This is impossible since one is an application and the other is a λ-abstraction. □

Lemma 5.15. $\varepsilon \in [\![\forall pq : o.(p \to q) \to (q \to p) \to p = q]\!]_\varphi^\sigma$.

Proof. Let $\mu_1, \mu_2 \in M$ and $X, Y \in \mathcal{D}_o$ be given. Let ψ be $(((\varphi\mu_1)^p_X)\mu_2)^q_Y$ and note that $\psi(p) = X\mu_2$ and $\psi(q) = Y$. We need to prove

$$\varepsilon \in [\![(p \to q) \to (q \to p) \to p = q]\!]^{\sigma\mu_1\mu_2}_{\psi}.$$

Let $\mu_3 \in M$ such that $\mu_3 \in [\![p \to q]\!]^{\sigma\mu_1\mu_2}_{\psi}$ be given. We need to prove

$$\mu_3 \in [\![(q \to p) \to p = q]\!]^{\sigma\mu_1\mu_2}_{\psi}.$$

Let $\mu_4 \in M$ such that $\mu_3\mu_4 \in [\![q \to p]\!]^{\sigma\mu_1\mu_2}_{\psi}$ be given. We need to prove

$$\mu_3\mu_4 \in [\![p = q]\!]^{\sigma\mu_1\mu_2}_{\psi}.$$

By Lemma 5.11 it is enough to prove $X\mu_2\mu_3\mu_4 = Y\mu_3\mu_4$. Suppose $\tau \in X\mu_2\mu_3\mu_4$. Then $\mu_3\mu_4\tau \in X\mu_2 = [\![p]\!]^{\sigma\mu_1\mu_2}_{\psi}$. Since $\mu_3 \in [\![p \to q]\!]^{\sigma\mu_1\mu_2}_{\psi}$ we know $\mu_3\mu_4\tau \in [\![q]\!]^{\sigma\mu_1\mu_2}_{\psi} = Y$ by Lemma 5.10. Hence $\tau \in Y\mu_3\mu_4$ and so $X\mu_2\mu_3\mu_4 \subseteq Y\mu_3\mu_4$. It remains to prove the reverse inclusion.

Suppose $\tau \in Y\mu_3\mu_4$. Then $\mu_3\mu_4\tau \in [\![q]\!]^{\sigma\mu_1\mu_2}_{\psi}$. Since $\mu_3\mu_4 \in [\![q \to p]\!]^{\sigma\mu_1\mu_2}_{\psi}$ we know $\mu_3\mu_4\tau \in [\![p]\!]^{\sigma\mu_1\mu_2}_{\psi} = X\mu_2$. Hence $\tau \in X\mu_2\mu_3\mu_4$ and we are done. □

The main result is soundness.

Theorem 5.1. *If* $\Gamma \vdash s$ *and* $\tau \in [\![u]\!]^{\sigma}_{\varphi}$ *for every* $u \in \Gamma$, *then* $\tau \in [\![s]\!]^{\sigma}_{\varphi}$.

Proof. We prove this by induction on the derivation of $\Gamma \vdash s$. In each case we assume we have a σ, τ and φ such that $\tau \in [\![u]\!]^{\sigma}_{\varphi}$ for every $u \in \Gamma$. The inductive hypothesis can be applied to the premises of the rule (possibly changing σ, τ and φ) and we must prove $\tau \in [\![s]\!]^{\sigma}_{\varphi}$.

Soundness of the axiom rule follows from Lemmas 5.12, 5.13, 5.14 and 5.15. Soundness of the hypothesis rule is clear. Soundness of the conversion rule follows from Lemma 5.9. Lemma 5.10 can be used to easily prove soundness for the implication introduction and elimination rules.

For the introduction rule for the universal quantifier suppose x is a variable of type α not free in Γ. We must prove $\tau \in [\![\forall x.s]\!]^{\sigma}_{\varphi}$. By definition this means we need to prove $\varepsilon \in [\![s]\!]^{\sigma\tau\mu}_{(\varphi\tau\mu)^x_a}$ for all $\mu \in M$ and $a \in \mathcal{D}_\alpha$. Let $\mu \in M$ and $a \in \mathcal{D}_\alpha$ be given. We will apply the inductive hypothesis with $\sigma\tau\mu$, ε and $(\varphi\tau\mu)^x_a$. Since $\tau \in [\![u]\!]^{\sigma}_{\varphi}$ we also know $\tau\mu \in [\![u]\!]^{\sigma}_{\varphi}$ for each $u \in \Gamma$. Using Lemma 5.4 we know $\varepsilon \in [\![u]\!]^{\sigma\tau\mu}_{\varphi\tau\mu}$ for each $u \in \Gamma$. Since x is not free in Γ we also know $\varepsilon \in [\![u]\!]^{\sigma\tau\mu}_{(\varphi\tau\mu)^x_a}$ for each $u \in \Gamma$. Hence the inductive hypothesis applies and we have $\varepsilon \in [\![s]\!]^{\sigma\tau\mu}_{(\varphi\tau\mu)^x_a}$ as desired.

We now consider the elimination rule for the universal quantifier. The inductive hypothesis gives $\tau \in [\![\forall x.s]\!]^{\sigma}_{\varphi}$. Hence $\varepsilon \in [\![s]\!]^{\sigma\tau\mu}_{(\varphi\tau\mu)^x_a}$ for every $\mu \in M$ and $a \in \mathcal{D}_\alpha$. In particular, $\varepsilon \in [\![s]\!]^{\sigma\tau\varepsilon}_{(\varphi\tau\varepsilon)^x_{[\![t]\!]^{\sigma\tau}_{\varphi\tau}}}$. That is, $\varepsilon \in [\![s]\!]^{\sigma\tau}_{(\varphi\tau)^x_{[\![t]\!]^{\sigma\tau}_{\varphi\tau}}}$. Applying Lemma 5.4 and Lemma 5.5 we have

$$[\![s]\!]^{\sigma\tau}_{(\varphi\tau)^x_{[\![t]\!]^{\sigma\tau}_{\varphi\tau}}} = [\![s]\!]^{\sigma\tau}_{(\varphi^x_{[\![t]\!]^{\sigma}_{\varphi}})\tau} = [\![s]\!]^{\sigma}_{(\varphi^x_{[\![t]\!]^{\sigma}_{\varphi}})}\tau = [\![s^x_t]\!]^{\sigma}_{\varphi}\tau.$$

Hence $\varepsilon \in [\![s^x_t]\!]^{\sigma}_{\varphi}\tau$. That is, $\tau \in [\![s^x_t]\!]^{\sigma}_{\varphi}$ as desired.

We finally consider the functional extensionality rule. Let s and t have type $\alpha\beta$ and let x be a variable of type α not free in Γ, s or t. We need to prove $\tau \in [\![s = t]\!]^{\sigma}_{\varphi}$.

Applying Lemma 5.11 we need to prove the two functions $[\![s]\!]^{\sigma}_{\varphi}\tau$ and $[\![t]\!]^{\sigma}_{\varphi}\tau$ are equal. Let $\mu \in M$ and $a \in \mathcal{D}_{\alpha}$ be given. Recalling the action on functions, we compute

$$([\![s]\!]^{\sigma}_{\varphi}\tau)(\mu, a) = ([\![s]\!]^{\sigma}_{\varphi}\tau\mu)(\varepsilon, a) = [\![s]\!]^{\sigma\tau\mu}_{\varphi\tau\mu}(\varepsilon, a) = [\![s]\!]^{\sigma\tau\mu}_{(\varphi\tau\mu)^x_a}(\varepsilon, a)$$

and

$$([\![t]\!]^{\sigma}_{\varphi}\tau)(\mu, a) = ([\![t]\!]^{\sigma}_{\varphi}\tau\mu)(\varepsilon, a) = [\![t]\!]^{\sigma\tau\mu}_{\varphi\tau\mu}(\varepsilon, a) = [\![t]\!]^{\sigma\tau\mu}_{(\varphi\tau\mu)^x_a}(\varepsilon, a).$$

It remains to prove the identity

$$[\![s]\!]^{\sigma\tau\mu}_{(\varphi\tau\mu)^x_a}(\varepsilon, a) = [\![t]\!]^{\sigma\tau\mu}_{(\varphi\tau\mu)^x_a}(\varepsilon, a).$$

Note that $\tau\mu \in [\![u]\!]^{\sigma}_{\varphi}$ for every $u \in \Gamma$. Hence $\varepsilon \in [\![u]\!]^{\sigma\tau\mu}_{\varphi\tau\mu}$ for every $u \in \Gamma$. Furthermore, since x is not free in Γ, we have $\varepsilon \in [\![u]\!]^{\sigma\tau\mu}_{(\varphi\tau\mu)^x_a}$ for every $u \in \Gamma$. Applying the inductive hypothesis with $\sigma\tau\mu$, ε and $(\varphi\tau\mu)^x_a$ we have $\varepsilon \in [\![sx = tx]\!]^{\sigma\tau\mu}_{(\varphi\tau\mu)^x_a}$. Using Lemma 5.11 we conclude $[\![sx]\!]^{\sigma\tau\mu}_{(\varphi\tau\mu)^x_a} = [\![tx]\!]^{\sigma\tau\mu}_{(\varphi\tau\mu)^x_a}$. Hence $[\![s]\!]^{\sigma\tau\mu}_{(\varphi\tau\mu)^x_a}(\varepsilon, a) = [\![t]\!]^{\sigma\tau\mu}_{(\varphi\tau\mu)^x_a}(\varepsilon, a)$ as desired. □

6. Related Work

In addition to the presheaf semantics of Hofmann mentioned earlier [11], work on permutation models should be mentioned. Gabbay and Pitts used Fraenkel-Mostowski permutation models to model logics capable of abstract reasoning about fresh names [10]. The action given by the permutation group on a set of atoms in [10] arguably plays a similar role to the action given by the monoid of substitutions here. Gabbay and Mathijssen provide a nominal axiomatisation of the lambda-calculus in [8, 9]. Similar permutation based nominal techniques were used by Urban to formalize properties of syntax with binders in the Isabelle theorem prover [18].

7. Conclusion

We have given an intuitionistic extensional higher-order theory supporting reasoning syntax using higher-order abstract syntax. The theory has a counterpart in the Proofgold network. As a simple test case we have published definitions of untyped λ-terms and β-equivalence, conjectures about the existence of certain combinators, and proofs resolving these conjectures.

Acknowledgment

This work has been supported by the European Research Council (ERC) Consolidator grant nr. 649043 *AI4REASON*. An earlier version of this work was formalized in Coq [12] using the Autosubst package [16] while part of Professor Gert Smolka's Programming Systems Lab at Saarland University.

References

[1] Abadi, M., Cardelli, L., Curien, P.L., Lévy, J.J.: Explicit substitutions. Journal of Functional Programming **1**(4), 375–416 (1991)
[2] Barendregt, H.: The Lambda Calculus: its Syntax and Semantics. North-Holland, Amsterdam, revised edn. (1984)
[3] Brown, C.E.: M-set models. In: Reasoning in Simple Type Theory: Festschrift in Honor of Peter B. Andrews on His 70th Birthday. College Publications (2008)

[4] Brown, C.E., Pąk, K.: A tale of two set theories. In: Kaliszyk, C., Brady, E.C., Kohlhase, A., Coen, C.S. (eds.) Intelligent Computer Mathematics - 12th International Conference, CICM 2019, Prague, Czech Republic, July 8-12, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11617, pp. 44–60. Springer (2019)
[5] de Bruijn, N.: Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. Indagationes Mathematicae (Proceedings) **34**(5), 381–392 (1972)
[6] Church, A.: A formulation of the simple theory of types. The Journal of Symbolic Logic **5**, 56–68 (1940)
[7] Dowek, G., Hardin, T., Kirchner, C.: Higher order unification via explicit substitutions. Information and Computation **157**(1–2), 183 – 235 (2000)
[8] Gabbay, M.J., Mathijssen, A.: The lambda-calculus is nominal algebraic. In: Benzmüller, C., Brown, C., Siekmann, J., Statman, R. (eds.) Reasoning in simple type theory: Festschrift in Honour of Peter B. Andrews on his 70th Birthday. Studies in Logic and the Foundations of Mathematics, IFCoLog (2008)
[9] Gabbay, M.J., Mathijssen, A.: A nominal axiomatisation of the lambda-calculus. Journal of Logic and Computation **20**(2), 501–531 (2010)
[10] Gabbay, M.J., Pitts, A.M.: A New Approach to Abstract Syntax Involving Binders. In: Proceedings of the 14th Annual Symposium on Logic in Computer Science (LICS 1999). pp. 214–224. IEEE Computer Society Press (1999)
[11] Hofmann, M.: Semantical analysis of higher-order abstract syntax. In: Proceedings of the 14th Annual IEEE Symposium on Logic in Computer Science. pp. 204–213. LICS '99, IEEE Computer Society, Washington, DC, USA (1999)
[12] The Coq development team: The Coq proof assistant reference manual. LogiCal Project (2020), http://coq.inria.fr, version 8.12
[13] Pfenning, F., Elliot, C.: Higher-order abstract syntax. SIGPLAN Notices **23**(7), 199–208 (Jun 1988)
[14] Prawitz, D.: Natural deduction: a proof-theoretical study. Dover (2006)
[15] Russell, B.: The Principles of Mathematics. Cambridge University Press (1903)
[16] Schäfer, S., Tebbi, T., Smolka, G.: Autosubst: Reasoning with de bruijn terms and parallel substitutions. In: Zhang, X., Urban, C. (eds.) Interactive Theorem Proving - 6th International Conference, ITP 2015, Nanjing, China, August 24-27, 2015. Logical Notes in Artifical Intelligence, Springer-Verlag (Aug 2015)
[17] Sørensen, M., Urzyczyn, P.: Lectures on the Curry-Howard Isomorphism. Rapport (Københavns universitet. Datalogisk institut), Datalogisk Institut, Københavns Universitet (1998)
[18] Urban, C.: Nominal techniques in isabelle/hol. Journal of Automated Reasoning **40**(4), 327–356 (2008)
[19] White, B.: Qeditas: A formal library as a bitcoin spin-off (2016), http://qeditas.org/docs/qeditas.pdf
[20] Zhang, X.: Using LEO-II to Prove Properties of an Explicit Substitution M-set Model. Bachelor's thesis, Saarland University (2008)