An adequacy theorem for partial type theory

Thierry Coquand and Simon Huber

Introduction

This paper has two main contributions. The first one is to present a domain model of dependent type theory where a type is interpreted as a finitary projection on one "universal" domain. We believe this model to be quite natural and canonical, since it can be presented as a simple *decidable* typing system on finite elements[1]. While this model is based on an "univerval" domain, two convertible terms have the *same* semantics[2], like for the set theoretic model [3]. The second contribution is to show, using this model, purely syntactical properties of dependent type theory. In particular, we can show that dependent product is one-to-one for conversion in a *weak* metatheory[3]. Furthermore, the technique that is used is similar to the use of "inclusive predicates", fundamental in domain theory [13, 16]. Another technical advantage of our approach is that we don't need to use contexts as Kripke worlds as in previous arguments [7, 2]. We also establish, in a weak metatheory, that two convertible terms in type theory (maybe *partial* [12, 15]), have the same Böhm tree.

1 Domain and finite elements

We shall use the following Scott domain, least solution of a recursive domain equation (see [19, 20] for a lively description of Scott domains and solutions to domain equations):

$$D \ = \ [D \to D] + \Pi \ D \ [D \to D] + \mathsf{N} + \mathsf{0} + \mathsf{S} \ D + \mathsf{U}_i$$

In this equation, $+$ denotes the coalesced sum [19] and $i = 0, 1, 2, \dots$

We write $a, b, u, v, \dots$ for the elements of this domain. We define $u(v)$ for u and v in D as follows: it is the application of u to v if u belongs to $D \to D$ and it is $\perp$ otherwise.

A fundamental result of domain theory is that the finite/compact elements of this domain can be described in a purely syntactical way, and both the *order* and the *compatibility* relations on these finite elements are *decidable* [17, 19, 20]. It also has been noticed [17] that this domain is *coherent* in the sense that a finite set is compatible (i.e. has a least upper bound) if, and only if, it is pairwise compatible.

Here is a syntactical description of the finite elements

- $\perp$ or
- U_i, N or $\mathsf{0}$ or
- $\mathsf{S}\ u$ where u is finite
- $\Pi\ a\ f$ where a is finite and f is a finite function or
- a finite function

and a finite function is of the form $\perp$ or $u_1 \mapsto v_1, \dots, u_n \mapsto v_n$ (with $n \geqslant 1$ and all u_i,v_i finite) such that whenever u_i and u_j are compatible then so are v_i and v_j.

The order relation on finite elements can then be described by the rules

[1]Finitary projections have already been used to model dependent type theory, e.g. in [6], but the observation that it can be presented as a decidable typing system on finite elements seems to be new.

[2]This is to be contrasted with an "untyped" semantics, like the one used in [1] and where one needs to quotient by an extra partial equivalence relation.

[3]This is to be contrasted with existing proofs [2, 7] which so far require strong logical principles, contrary to what is expected for proving a purely syntactical property. The references [4, 18] are in a weak metatheory but do not cover η-conversion.

- $\bot \leqslant u$,
- $\mathsf{S}\ u \leqslant \mathsf{S}\ v$ if $u \leqslant v$,
- $\Pi\ a\ f \leqslant \Pi\ b\ g$ if $a \leqslant b$ and $f \leqslant g$, and
- $(u_1 \mapsto v_1, \dots, u_n \mapsto v_n) \leqslant (a_1 \mapsto b_1, \dots, a_m \mapsto b_m)$ if $v_i \leqslant \vee\{b_j \mid a_j \leqslant u_i\}$ for all i.

A *finitary projection* [19, 20] of a Scott domain E is a map $p : E \to E$ such that $p \circ p = p$ and $p\ a \leqslant a$ and the image of p, which is also the set of fixed-points of p, is a Scott domain. If $p\ u = u$ and $p\ v = v$ and u, v are compatible then $p\ (u \vee v) = u \vee v$ since both u and v are $\leqslant p\ (u \vee v)$. A finitary projection is thus completely determined by a set of finite elements which is closed by compatible sups. If F, E are two Scott domains, we write $F \lhd E$, F is a *subdomain* of E, to mean that F is the image of a finitary projection of E. Equivalently F is the set of directed sups of a given subset of finite elements of E which is closed by compatible binary sups, and this set is exactly the set of finite elements of F. A fundamental result [19] is that the poset of finitary projections of a Scott domain E is itself a Scott domain, which is a subdomain of $E \to E$.

The domain D can be seen as the limit of a chain of domains $D_n \lhd D_{n+1} \lhd D$ with $D_0 = \bot$ and

$$D_{n+1}\ =\ [D_n \to D_n] + \Pi\ D_n\ [D_n \to D_n] + \mathsf{N} + \mathsf{0} + \mathsf{S}\ D_n + \mathsf{U}_i$$

The domain D_n is the image of a finitary projection p_n of D. We can define $rk(u)$ the *rank* of a finite element u as the least n such that $p_n\ u = u$ [5]. If $rk(f) = n + 1$ we have $f = p_n \circ f \circ p_n$.

If $rk(f) = n + 1$ then we have $f\ u = p_n\ (f\ u)$ and so, since $f\ u$ is finite, $rk(f\ u) < rk(f)$ for any u in D.

In general there are different possible ways to write a finite function f as a least upper bound of step functions. For instance, we have $(\bot \mapsto \mathsf{N}) = (\mathsf{U}_3 \mapsto \mathsf{N}, \bot \mapsto \mathsf{N})$. We say that a description $f = (u_1 \mapsto v_1, \dots, u_n \mapsto v_n)$ is *minimal* if we cannot remove some $u_i \mapsto v_i$ in this description. An important property is the following.

Lemma 1.1 *If $f = (u_1 \mapsto v_1, \dots, u_n \mapsto v_n)$ is minimal, we have $f\ u < f\ u_i$ whenever $u < u_i$.*

Proof. If we have $u < u_i$ and $f\ u_i = f\ u$, then $\vee\{v_j \mid u_j < u_i\} \geqslant f\ u = f\ u_i$ and we can remove $u_i \mapsto v_i$ from the given description of f. □

Lemma 1.2 *If $f = (u_1 \mapsto v_1, \dots, u_n \mapsto v_n)$ is minimal and $f = f \circ p$ where p is a finitary projection, then $p\ u_i = u_i$ for all i.*

Proof. We have $f\ u_i = f\ (p\ u_i)$ and so we cannot have $p\ u_i < u_i$ since the description is minimal, and $p\ u_i = u_i$. □

Corollary 1.3 *If $f = (u_1 \mapsto v_1, \dots, u_n \mapsto v_n)$ is minimal, then we have $rk(u_i) < rk(f)$ for all i.*

Proof. If $rk(f) = n + 1$, we have $f = f \circ p_n$ and so $p_n\ u_i = u_i$ and $rk(u_i) \leqslant n$ for all i. □

We define the relation $lv(a, n)$ inductively. We have $lv(\bot, 0)$ and $lv(\Pi\ a\ f, n)$ if $lv(a, n)$ and $lv(f, n)$ and $lv(\mathsf{U}_i, n)$ if $i \leqslant n$ and finally $lv((u_1 \mapsto v_1, \dots, u_p \mapsto v_p), n)$ if $lv(u_i, n)$ and $lv(v_i, n)$ for all i.

It is then direct that $lv(a \vee b, n)$ if we have both $lv(a, n)$ and $lv(b, n)$ and a and b are compatible.

Using the following Lemma, we can compute the *level* $lv(a)$, least n such that $lv(a, n)$ as a function of a, defining $lv(f)$ as the maximum of $lv(u_i), lv(f\ u_i)$ for $f = (u_1 \mapsto v_1, \dots, u_n \mapsto v_n)$ minimal.

Lemma 1.4 *If we have two minimal descriptions of the same function $f = (u_1 \mapsto v_1, \dots, u_n \mapsto v_n) = (a_1 \mapsto b_1, \dots, a_m \mapsto b_m)$, then $u_i = \vee\{a_j \mid a_j \leqslant u_i\}$ and $f\ u_i = \vee\{f\ a_j \mid a_j \leqslant u_i\}$.*

Finally we define the *complexity* of a finite element a as the pair $lv(a), rk(a)$ with the lexicographic ordering.

2 Concrete description of the typing relation on finite elements

We now describe a *type system* on finite elements.

$$\frac{}{\bot : \bot} \qquad \frac{}{\bot : \mathsf{U}_j} \qquad \frac{}{\bot : \mathsf{N}}$$

$$\frac{}{\mathsf{U}_i : \mathsf{U}_j}\, i < j \qquad \frac{}{\mathsf{N} : \mathsf{U}_j} \qquad \frac{}{\mathsf{0} : \mathsf{N}} \qquad \frac{u : \mathsf{N}}{\mathsf{S}\ u : \mathsf{N}}$$

$$\frac{a : \mathsf{U}_j \quad u_1 : a \quad t_1 : \mathsf{U}_j \ \ldots \ u_n : a \quad t_n : \mathsf{U}_j}{\Pi\ a\ (u_1 \mapsto t_1, \ldots, u_n \mapsto t_n) : \mathsf{U}_j}$$

$$\frac{u_1 : a \quad v_1 : f(u_1) \ \ldots \ u_n : a \quad v_n : f(u_n)}{(u_1 \mapsto v_1, \ldots, u_n \mapsto v_n) : \Pi\ a\ f}$$

Note that, with these rules, if $a : \mathsf{U}_k$, then a is strictly simpler than U_k since there is a description of a which does not contain any U_i with $i \geqslant k$.

Lemma 2.1 *If $u : a$ and $a \leqslant b$, then $u : b$. If $u : a$, $v : a$, and u and v are compatible, then $u \vee v : a$.*

Proof. The first statement is by induction on the derivation of $u : a$. For the second statement, we look at the case where $a = \mathsf{U}_k$ and $u = \Pi\ b\ (u_1 \mapsto t_1, \ldots, u_n \mapsto t_n)$ and $v = \Pi\ b'\ (v_1 \mapsto l_1, \ldots, v_m \mapsto l_m)$. By induction, we have $b \vee b' : \mathsf{U}_k$. Also $u_i : b$ and hence $u_i : b \vee b'$ by the first statement and similarly $v_j : b \vee b'$. The other cases are similar. □

Corollary 2.2 *If $w : \Pi\ a\ f$ and $u : a$, then $w(u) : f(u)$.*

Proof. We can write $w = (u_1 \mapsto v_1, \ldots, u_n \mapsto v_n)$ with $v_i : f(u_i)$. We then have $w(u) = \vee\{v_i \mid u_i \leqslant u\} : f(u)$ by Lemma 2.1. □

Lemma 2.3 *If $\Pi\ a\ f : \mathsf{U}_k$ and $f = (u_1 \mapsto t_1, \ldots, u_n \mapsto t_n)$ is minimal, then $u_i : a$ and $f(u_i) : \mathsf{U}_k$.*

Proof. We have $f = (u_1 \mapsto t_1, \ldots, u_n \mapsto t_n) = (v_1 \mapsto l_1, \ldots, v_m \mapsto l_m)$ with $v_j : a$ and $l_j : \mathsf{U}_k$. Since $f(u_i) = \vee\{l_j \mid v_j \leqslant u_i\}$ we have $f(u_i) : \mathsf{U}_k$ by Lemma 2.1. Also $u_i = \vee\{v_j \mid v_j \leqslant u_i\}$ and so $u_i : a$ by Lemma 2.1. □

Lemma 2.4 *If $w : \Pi\ a\ f$ and $w = (u_1 \mapsto t_1, \ldots, u_n \mapsto t_n)$ is minimal, then $u_i : a$ and $w(u_i) : f(u_i)$.*

Proof. We have $w = (u_1 \mapsto t_1, \ldots, u_n \mapsto t_n) = (v_1 \mapsto l_1, \ldots, v_m \mapsto l_m)$ with $v_j : a$ and $l_j : f(v_j)$. It follows from Lemma 1.3 that we have $u_i = \vee v_j \mid v_j \leqslant u_i$ and so $u_i : a$ by Lemma 2.1. Using Corollary 2.2, we get $w(u_i) : f(u_i)$. □

Corollary 2.5 *The relation $u : a$ is decidable.*

Proof. By induction on $rk(a)$ and $rk(u)$. □

The following Lemma will be useful when connecting syntax and semantics.

Lemma 2.6 *If $w : \Pi\ b\ f$ and $b \leqslant a$, then we any $u : a$ there exists $v : b$ such that $v \leqslant u$ and $w(u) = w(v)$.*

Proof. We write $w = (u_1 \mapsto l_1, \ldots, u_n \mapsto l_n)$ with $u_i : b$ and $l_i : f(u_i)$. We then have $w(u) = w(v)$ for $v = \vee\{u_i \mid u_i \leqslant u\}$ and $v : b$ by Lemma 2.1. □

We now introduce the predicate a type by the rules.

$$\frac{}{\bot\ \mathsf{type}} \qquad \frac{}{\mathsf{U}_i\ \mathsf{type}} \qquad \frac{}{\mathsf{N}\ \mathsf{type}} \qquad \frac{a\ \mathsf{type} \quad u_1 : a \quad t_1\ \mathsf{type} \ \ldots \ u_n : a \quad t_n\ \mathsf{type}}{\Pi\ a\ (u_1 \mapsto t_1, \ldots, u_n \mapsto t_n)\ \mathsf{type}}$$

Lemma 2.7 *If $a : \mathsf{U}_j$, then a type. If a type, b type, and a, b are compatible, then $a \vee b$ type. If $\Pi\ a\ (u_1 \mapsto t_1, \ldots, u_n \mapsto t_n)$ type and $u_1 \mapsto t_1, \ldots, u_n \mapsto t_n$ is a minimal description, then $u_i : a$ and t_i type.*

Proof. The first statement is by induction on the derivation of $a : \mathsf{U}_j$. The second statement is proved as in Lemma 2.1, and the last statement as in the proof of Lemma 2.3. $\square$

Corollary 2.8 *The predicate a type is decidable.*

Given a finite element a, the set of finite elements u such that $u : a$ is closed by compatible binary sups by Lemma 2.1. Hence it defines a finitary projection $p\ a$. Similarly the set of finite elements a such that a type defines a finitary projection p_{type}. We write $\mathsf{Type} \lhd D$ the corresponding subdomain.

By Lemma 2.1, we have $p\ a \leqslant p\ b$ if $a \leqslant b$ and we can hence define the finitary projection $p\ a$ for an *arbitrary* element a of D, not necessarily finite, as the directed sup of all $p\ a_0$ for $a_0 \leqslant a$ finite, in the Scott domain of finitary projections of D. We write $El\ a \lhd D$ the image of $p\ a$.

We have $El\ \mathsf{U}_i \lhd El\ \mathsf{U}_{i+1}$ and $El\ \mathsf{U}_i \lhd \mathsf{Type}$.

Let us write $a \to b$ for $\Pi\ a\ (\bot \mapsto b)$. The domain $El\ \mathsf{N} \lhd D$ is exactly the domain of "lazy" natural numbers, that is elements of the form $\mathsf{S}^k\ 0$ or $\mathsf{S}^k \bot$. The poset of finite elements w such that $w : \mathsf{N} \to \mathsf{N}$ is exactly the poset of finite element of the domain of continuous functions $El\ \mathsf{N} \to El\ \mathsf{N}$.

Proposition 2.9 *We have*

$$\begin{array}{lcll}
p\ \mathsf{N}\ 0 & = & 0 & \\
p\ \mathsf{N}\ (\mathsf{S}\ u) & = & \mathsf{S}\ (p\ \mathsf{N}\ u) & \\
p\ (\Pi\ a\ f)\ w & = & x \mapsto p\ (f(p\ a\ x))\ (w(p\ a\ x)) & \\
p\ \mathsf{U}_j\ \mathsf{N} & = & \mathsf{N} & \\
p\ \mathsf{U}_j\ (\Pi\ a\ f) & = & \Pi\ (p\ \mathsf{U}_j\ a)\ ((p\ \mathsf{U}_j) \circ\ f\ \circ (p\ a)) & \\
p\ \mathsf{U}_j\ \mathsf{U}_i & = & \mathsf{U}_i & \text{if} \quad i < j
\end{array}$$

and $p\ a\ b = \bot$ in all other cases. We also have

$$\begin{array}{lcl}
p_{\mathsf{type}}\ \mathsf{N} & = & \mathsf{N} \\
p_{\mathsf{type}}\ (\Pi\ a\ f) & = & \Pi\ (p_{\mathsf{type}}\ a)\ (p_{\mathsf{type}} \circ\ f\ \circ (p\ a)) \\
p_{\mathsf{type}}\ \mathsf{U}_i & = & \mathsf{U}_i
\end{array}$$

and $p_{\mathsf{type}}\ b = \bot$ in all other cases.

Proof. Let $q\ a$ be the function defined by these recursive equations. We show by induction on the complexity of a finite that we have $q\ a\ u = u$, for u finite, if, and only if, $u : a$. This is clear if $a = \mathsf{N}$. If $a = \Pi\ b\ f$ and $u : a$, then using Lemma 2.4, we can write $u = (u_1 \mapsto v_1, \dots, u_n \mapsto v_n)$ with $u_i : a$ and $v_i : f(u_i)$. We then have $u(x) = u(q\ b\ x) : f\ (q\ b\ x)$ and so $u(x) = (q\ (\Pi\ b\ f)\ u)(x)$ for any x and so $u = q\ a\ u$. Conversely, if $u = q\ a\ u$, we have $u = u \circ (q\ b)$, and if $u = (u_1 \mapsto v_1, \dots, u_n \mapsto v_n)$ is a minimal description of u, we have $q\ b\ u_i = u_i$ by Lemma 1.2. So $u_i : b$ by induction. We then have $v_i = q\ (f(u_i))\ v_i$ and so $v_i : f(u_i)$ by induction.

Finally we prove $q\ \mathsf{U}_k\ a = a$ if, and only if, $a : \mathsf{U}_k$ by induction on the complexity of a finite. We cover the case $a = \Pi\ b\ f$ where $u_1 \mapsto l_1, \dots, u_n \mapsto l_n$ is a minimal description of f.

If $a : \mathsf{U}_k$, then $b : \mathsf{U}_k$ and so $q\ \mathsf{U}_k\ b = b$ by induction and $u_i : b$ and $l_i : \mathsf{U}_k$. Since b is strictly less complex than U_k, we have by induction $q\ b\ u_i = u_i$ and $q\ \mathsf{U}_k\ f(u_i) = f(u_i)$. It follows that we have $q\ \mathsf{U}_k\ a = a$.

Conversely, if $q\ \mathsf{U}_k\ a = a$ then $q\ \mathsf{U}_k\ b = b$ and so $b : \mathsf{U}_k$ by induction and we have $(q\ \mathsf{U}_k) \circ f \circ (q\ b) = f$. It follows that we have $f(u_i) = q\ \mathsf{U}_k\ (f(q\ b\ u_i))$ for all i and we have $q\ b\ u_i = u_i$ by Lemma 1.2. So $u_i : b$ since b is simpler than U_k. We then get $f(u_i) = q\ \mathsf{U}_k\ f(u_i)$ and so $f(u_i) : \mathsf{U}_k$ by induction. $\square$

We can now consider the continuous families of domains $El\ a$ and $El\ a \to \mathsf{Type}$ indexed over a in Type. We can form their products and get a continuous family $El\ a \times (El\ a \to \mathsf{Type})$ indexed over a in Type. We can consider the sum of this family, which is itself a Scott domain [8]

$$E = \Sigma(a \in \mathsf{Type})\ (El\ a \times (El\ a \to \mathsf{Type}))$$

and we have an evaluation function $E \to \mathsf{Type}, (a, v, f) \longmapsto f(v)$. This evaluation function is continuous. So, if we have $w_0 \leqslant f(v)$ in Type then we can find $a_0 \leqslant a$ finite in Type, and $u_0 \leqslant u$ finite in $El\ a_0$ and $f_0 \leqslant f$ finite in $El\ a_0 \to \mathsf{Type}$ such that $w_0 \leqslant f(v)$. This remark will be used in a crucial way in connecting syntax and semantics of type theory.

$$[\![x]\!]\rho = \rho(x) \qquad [\![M\ N]\!]\rho = [\![M]\!]\rho([\![N]\!]\rho) \qquad [\![\mathsf{N}]\!]\rho = \mathsf{N} \qquad [\![\mathsf{U}_i]\!]\rho = \mathsf{U}_i \qquad [\![\mathsf{0}]\!]\rho = \mathsf{0} \qquad [\![\mathsf{S}\ M]\!]\rho = \mathsf{S}\ ([\![M]\!]\rho)$$

$$[\![\lambda(x : A)M]\!]\rho \ = \ u \mapsto [\![M]\!](\rho, x : [\![A]\!]\rho = u)$$
$$[\![\Pi(x : A)B]\!]\rho \ = \ \Pi\ ([\![A]\!]\rho)\ (u \mapsto [\![B]\!](\rho, x : [\![A]\!]\rho = u))$$

Figure 1: Denotational semantics of type theory

3 Syntax and semantics of type theory

The syntax of type theory is defined as follows.

$$M, N, A, B \ ::= \ x \mid \lambda(x : A)M \mid M\ N \mid \Pi(x : A)B \mid \mathsf{N} \mid \mathsf{U}_i \mid \mathsf{0} \mid \mathsf{S}\ M \mid \mathsf{rec}(\lambda x\ A, M, M)$$

We write $F(x/M)$ the substitution of M for x in F. We may write simply $F(M)$ is x is clear from the context.

The semantics can be defined at this purely untyped syntactic level, exactly like for the set theoretic semantics presented in [3]. This semantics is described in Figure 1 where we define $\rho, x : a = u$ to be the update of ρ with the assignment $x = p\ a\ u$.

The semantics of rec is the usual lazy semantics of primitive recursion. We define $\mathsf{rec}(d_0, d_1)$ in $D \to D$ as the least function such that, for finite u

$$\mathsf{rec}(d_0, d_1)\ \bot \ = \ \bot \qquad \mathsf{rec}(d_0, d_1)\ \mathsf{0} \ = \ d_0 \qquad \mathsf{rec}(d_0, d_1)\ (\mathsf{S}\ u) \ = \ d_1(u)(\mathsf{rec}(d_0, d_1)\ u)$$

and then $[\![\mathsf{rec}(\lambda.T, M_0, M_1)]\!]\rho = \mathsf{rec}([\![M_0]\!]\rho, [\![M_1]\!]\rho)$.

The typing and conversion rules are in the appendix. There are two judgments for types, of the form A type and A conv A', and two judgments for elements, of the form $M : A$ and M conv $M' : A$. Such a judgment is stated in a *context*, which is a list of typing declarations $x : A$. As in [10], we may not write the context explicitly.

We say that ρ *fits* Γ if for all $x : A$ in Γ we have $[\![A]\!]\rho$ in Type and $\rho(x) : [\![A]\!]\rho$.

Theorem 3.1 *If ρ fits Γ, then $\Gamma \vdash A$ type implies $[\![A]\!]\rho \in \mathsf{Type}$ and $\Gamma \vdash A$ conv A' implies $[\![A]\!]\rho = [\![A']\!]\rho$. If $\Gamma \vdash M : A$, then $[\![M]\!]\rho \in El\ ([\![A]\!]\rho)$ and if $\Gamma \vdash M$ conv $M' : A$, then $[\![M]\!]\rho = [\![M']\!]\rho$.*

Proof. Direct by induction on the derivation of the judgment $\Gamma \vdash A$ type or $\Gamma \vdash A$ conv A' or $\Gamma \vdash M : A$ or $\Gamma \vdash M$ conv $M' : A$. $\square$

Note that the use of finitary projections takes care of η-conversion in the semantics. For instance, we have $[\![\lambda(x : \mathsf{N} \to \mathsf{N})x]\!] = [\![\lambda(x : \mathsf{N} \to \mathsf{N})\lambda(y : \mathsf{N})x\ y]\!]$. Indeed, both are equal to the function $u \mapsto v \mapsto p\ \mathsf{N}\ (p\ \mathsf{N}\ v)$.

The main difference with the semantics suggested in [12] and in [15] is that abstraction is not interpreted as a constructor. This is crucial in order to validate the rule of η-conversion that N conv $N' : \Pi(x : A)B$ as soon as $N\ x$ conv $N'\ x : B\ (x : A)$. If we represent abstraction by a constructor, we would have $w = \lambda(\bot) \neq \bot = w'$ but also $w(u) = \bot = w'(u)$ for any u in D, and so the rule for η-conversion cannot be valid in this case.

4 Connecting syntax and semantics, first version

We write $M \to M'$ for weak-head reduction. This is defined at a purely syntactical level. The rules are the following.

$$\frac{}{(\lambda(x : A)N)\ M \to N(x/M)}$$

$$\frac{N \to N'}{N\ M \to N'\ M}$$

$$\frac{}{\mathsf{rec}(\lambda x\ T, M_0, M_1)\ \mathsf{0} \to M_0}$$

$$\frac{}{\mathsf{rec}(\lambda x\ T, M_0, M_1)\ (\mathsf{S}\ N) \to M_1\ N\ (\mathsf{rec}(\lambda x\ T, M_0, M_1)\ N)}$$

$$\frac{N \to N'}{\mathsf{rec}(\lambda x\ T, M_0, M_1)\ N \to \mathsf{rec}(\lambda x\ T, M_0, M_1)\ N'}$$

We write $M \to_A M'$ for $M \to M'$ and $M\ \mathsf{conv}\ M' : A$ and write $M \to_A^* M'$ for the corresponding transitive reflexive closure. We write $A \to_{\mathsf{type}} A'$ to mean that $A \to A'$ and $A\ \mathsf{conv}\ A'$, and we write $A \to_{\mathsf{type}}^* A'$ the corresponding transitive reflexive closure. These relations are similar to the relations used in [7, 2].

In this section, we will consider only *closed* terms. The relation $A\ \mathsf{conv}\ B$ defines an equivalence relation on the set of terms A such that $A\ \mathsf{type}$. If $A\ \mathsf{type}$, then, similarly, the relation $M\ \mathsf{conv}\ N : A$ defines an equivalence relation on the set of terms M satisfying $M : A$.

The main goal of this section is to analyze relations refining these predicates. We define $A\ \mathsf{type} \mid a$ and $A\ \mathsf{conv}\ B \mid a$ for $a \leqslant [\![A]\!]$ in Type and, if we have $A\ \mathsf{type} \mid a$, we define $M : A \mid u : a$ and $M\ \mathsf{conv}\ M' : A \mid u : a$ for $u \leqslant [\![M]\!]$ in $El\ a$. The relation $A\ \mathsf{conv}\ B \mid a$ will be an equivalence relation on the set of terms satisfying the predicate $\mid a$, while, if $A\ \mathsf{type} \mid a$, the relation $M\ \mathsf{conv}\ M' : A \mid u : a$ will be an equivalence relation on the set of terms M such that $M : A \mid u : a$. These relations are defined first for a finite element, by *induction* on a in Type. At the same time we define relations $A\ \mathsf{type} \mid_i a$ and $A\ \mathsf{conv}\ A' \mid_i a$ for $i = 0, 1, \ldots$. The definition is as follows.

1. If $a = \bot$, then $A\ \mathsf{type} \mid a$ means $A\ \mathsf{type}$, and $A\ \mathsf{conv}\ A' \mid a$ means $A\ \mathsf{conv}\ A'$. In this case $u : a$ is only satisfied for $u = \bot$, and $M : A \mid u : a$ means $M : A$ and $M\ \mathsf{conv}\ M' \mid u : a$ means $M\ \mathsf{conv}\ M' : A$.

2. If $a = \bot$, then $A\ \mathsf{type} \mid_i a$ means $A : \mathsf{U}_i$, and $A\ \mathsf{conv}\ A' \mid_i a$ means $A\ \mathsf{conv}\ A' : \mathsf{U}_i$. In this case $u : a$ is only satisfied for $u = \bot$, and $M : A \mid u : a$ means $M : A$ and $M\ \mathsf{conv}\ M' \mid u : a$ means $M\ \mathsf{conv}\ M' : A$.

3. If $a = \mathsf{U}_j$, then $A\ \mathsf{type} \mid a$ means $A \to_{\mathsf{type}}^* \mathsf{U}_j$, and $A\ \mathsf{conv}\ A' \mid a$ is always satisfied (given $A'\ \mathsf{type} \mid a$). In this case and $M : A \mid u : a$ means $M \mid_j u$ and $M\ \mathsf{conv}\ M' \mid u : a$ means $M\ \mathsf{conv}\ M' \mid_j u$.

4. If $a = \mathsf{U}_j$ and $i > j$, then $A\ \mathsf{type} \mid_i a$ means $A \to_{\mathsf{U}_i}^* \mathsf{U}_j$, and $A\ \mathsf{conv}\ A' \mid_i a$ is always satisfied (given $A'\ \mathsf{type} \mid_i a$).

5. If $a = \Pi\ b\ f$, then $A\ \mathsf{type} \mid a$ means $A \to_{\mathsf{type}}^* \Pi(x : B)F$ for some B and F such that $B\ \mathsf{type} \mid b$ and

   (a) $N : B \mid v : b$ implies $F(N)\ \mathsf{type} \mid f(v)$, and

   (b) $N\ \mathsf{conv}\ N' : B \mid v : b$ implies $F(N)\ \mathsf{conv}\ F(N') \mid f(v)$.

   If also $A'\ \mathsf{type} \mid a$, then $A' \to_{\mathsf{type}}^* \Pi(x : B')F'$ and $A\ \mathsf{conv}\ A' \mid a$ means that $B\ \mathsf{conv}\ B' \mid b$ and $N : B \mid v : b$ implies $F(N)\ \mathsf{conv}\ F(N) \mid f(v)$.

   In case $A\ \mathsf{type} \mid a$, $M : A \mid u : a$ means

   (a) $N : B \mid v : b$ implies $M\ N : F(N) \mid u(v) : f(v)$, and

   (b) $N\ \mathsf{conv}\ N' : B \mid v : b$ implies $M\ N\ \mathsf{conv}\ M\ N' : F(N) \mid u(v) : f(v)$.

   If also $M' : A \mid u : a$, then $M\ \mathsf{conv}\ M' : A \mid u : a$ means that $N : B \mid v : b$ implies $M\ N\ \mathsf{conv}\ M'\ N : F(N) \mid u(v) : f(v)$.

6. If $a = \Pi\ b\ f$, then $A\ \mathsf{type} \mid_i a$ means $A \to_{\mathsf{U}_i}^* \Pi(x : B)F$ for some B and F such that $B\ \mathsf{type} \mid_i b$ and

   (a) $N : B \mid v : b$ implies $F(N)\ \mathsf{type} \mid_i f(v)$, and

   (b) $N\ \mathsf{conv}\ N' : B \mid v : b$ implies $F(N)\ \mathsf{conv}\ F(N') \mid_i f(v)$.

   If also $A'\ \mathsf{type} \mid_i a$, then $A' \to_{\mathsf{U}_i}^* \Pi(x : B')F'$ and $A\ \mathsf{conv}\ A' \mid_i a$ means that $B\ \mathsf{conv}\ B' \mid_i b$ and $N : B \mid v : b$ implies $F(N)\ \mathsf{conv}\ F(N) \mid_i f(v)$.

7. If $a = \mathsf{N}$, then A $\mathsf{type} \mid a$ means $A \rightarrow^*_{\mathsf{type}} \mathsf{N}$, and if also A' $\mathsf{type} \mid a$, then A conv $A' \mid a$ is always satisfied. In this case $M : A \mid u : a$ is defined by induction on u. If $u = \bot$, it means $M : A$. If $u = \mathsf{0}$, it means $M \rightarrow^*_{\mathsf{N}} \mathsf{0}$. If $u = \mathsf{S}\ v$, it means $M \rightarrow^*_{\mathsf{N}} \mathsf{S}\ N$ and $N : A \mid v : a$. Finally, M conv $M' : A \mid \bot : a$ means M conv $M' : A$, and M conv $M' : A \mid \mathsf{0} : a$ is always satisfied, and M conv $M' : A \mid \mathsf{S}\ v : a$ means $M' \rightarrow^*_{\mathsf{N}} \mathsf{S}\ N'$ with N conv $N' : \mathsf{N} \mid v : a$.

8. If $a = \mathsf{N}$, then A $\mathsf{type} \mid_i a$ means $A \rightarrow^*_{\mathsf{U}_i} \mathsf{N}$, and if also A' $\mathsf{type} \mid_i a$, then A conv $A' \mid_i a$ is always satisfied.

Let us look at one example: A $\mathsf{type} \mid a$, for $a = \Pi\ \mathsf{N}\ (\mathsf{0} \mapsto \mathsf{N},\ \mathsf{S}\ \mathsf{0} \mapsto \mathsf{U}_3)$, means $A \rightarrow^*_{\mathsf{type}} \Pi(x : B)F$ with B $\mathsf{type} \mid \mathsf{N}$, that is $B \rightarrow^*_{\mathsf{type}} \mathsf{N}$, and

- if $M \rightarrow^*_{\mathsf{N}} \mathsf{0}$, then $F(M)\mathsf{type} \mid \mathsf{N}$ that is $F(M) \rightarrow^*_{\mathsf{type}} \mathsf{N}$, and
- if $M \rightarrow^*_{\mathsf{N}} \mathsf{S}\ \mathsf{0}$ then $F(M) \mid \mathsf{U}_3$ that is $F(M) \rightarrow^*_{\mathsf{type}} \mathsf{U}_3$.

Lemma 4.1 *The following properties hold.*

1. *If $J \mid a$ and $a' \leqslant a$ in* Type, *then $J \mid a'$ where J is A* type *or A* conv *B.*
2. *If $J \mid_i a$ and $a' \leqslant a$ in El* U_i, *then $J \mid_i a'$ where J is A* type *or A* conv *B.*
3. *If A* $\mathsf{type} \mid a$ *and $a' \leqslant a$ in* Type *and $J \mid u' : a'$, then $J \mid u' : a$ where J is $M : A$ or M* conv *$M' : A$.*
4. *Finally, if A* $\mathsf{type} \mid a$ *and $a' \leqslant a$ in* Type *and $J \mid u : a$ and $u' : a'$ and $u' \leqslant u$, then $J \mid u' : a'$ where J is $M : A$ or M* conv *$M' : A$.*

Proof. We prove simultaneously the assertions by induction on a in Type. We explain two representative cases.

In case $a = \Pi\ b\ f$ and $a' = \Pi\ b'\ f' \leqslant a$ and A $\mathsf{type} \mid a$, we assume A $\mathsf{type} \mid a$ and we show A $\mathsf{type} \mid a'$. We first have $A \rightarrow^*_{\mathsf{type}} \Pi(x : B)F$ and B $\mathsf{type} \mid b$ and $N : B \mid v : b$ implies $F(N)$ $\mathsf{type} \mid f(v)$ and N conv $N' : B \mid v : b$ implies $F(N)$ conv $F(N') \mid f(v)$. By induction we have B $\mathsf{type} \mid b'$. Also if $N : B \mid v : b'$ then $N : B \mid v : b$ by induction and so $F(N)$ $\mathsf{type} \mid f(v)$ and so $F(N)$ $\mathsf{type} \mid f'(v)$ by induction. Similarly, N conv $N'B \mid v : b'$ implies $F(N)$ conv $F(N') \mid f(v)$ and so $F(N)$ conv $F(N') \mid f'(v)$ by induction.

If $a = \Pi\ b\ f$ and $a' = \Pi\ b'\ f' \leqslant a$ and A $\mathsf{type} \mid a$ and $M : A \mid u : a'$, then we claim that $M : A \mid u : a$. We know $A \rightarrow^*_{\mathsf{type}} \Pi(x : B)F$ with B $\mathsf{type} \mid b$. We have to show that $N : B \mid v : b$ implies $M\ N : F(N) \mid u(v) : f(v)$. By induction, we know that if $v' \leqslant v$ and v' in *El* b' then $N : B \mid v' : b'$. Since $M : A \mid u : a'$ we have $M\ N : F(N) \mid u(v') : f'(v')$. Since $u : a'$ we have $v' \leqslant v$ in *El* b' such that $u(v) = u(v')$ by Lemma 2.6. For this v' we have $M\ N : F(N) \mid u(v) : f'(v')$ and then $M\ N : F(N) \mid u(v) : f(v)$ by induction, since $F(N)$ $\mathsf{type} \mid f(v)$ and $f'(v') \leqslant f(v)$. We prove similarly that N conv $P : B \mid v : b$ implies $M\ N$ conv $M\ P : F(N) \mid u(v) : f(v)$ □

We use this result to extend the relation A $\mathsf{type} \mid a$ for a arbitrary (possibly infinite) in Type: we define it to mean A $\mathsf{type} \mid a_0$ for *all finite* $a_0 \leqslant a$ in Type.

The relations $J \mid u : a$ where J is $M : A$ or M conv $M' : A$ for u arbitrary in *El* a is then defined as follows: for *all* $u_0 \leqslant u$ finite in *El* a there *exists* $a_0 \leqslant a$ finite in Type such that $J \mid u_0 : a_0$. Note that if we have $J \mid u_0 : a_0$ then we also have $J \mid u_0 : a_1$ for any finite a_1 such that $a_0 \leqslant a_1 \leqslant a$ in Type by Lemma 4.1.

Proposition 4.2 *We have A* $\mathsf{type} \mid \Pi\ b\ f$ *if, and only if, $A \rightarrow^*_{\mathsf{type}} \Pi(x : B)F$ with B* $\mathsf{type} \mid b$ *and $N : B \mid v : b$ implies $F(N)$* $\mathsf{type} \mid f(v)$ *and N* conv *$N' : B \mid v : b$ implies $F(N)$* conv *$F(N') \mid f(v)$.*

Proof. We assume A $\mathsf{type} \mid \Pi\ b\ f$. This means A $\mathsf{type} \mid \Pi\ b_0\ f_0$ for any finite $\Pi\ b_0\ f_0 \leqslant \Pi\ b\ f$ in Type, and, in particular, we have A $\mathsf{type} \mid \Pi\ b_0\ f_0$ for $b_0 = \bot$ and $f_0 = \bot$. This implies $A \rightarrow^*_{\mathsf{type}} \Pi(x : B)F$ for some B type and F type $(x : B)$.

If $b_0 \leqslant b$ in Type is finite we have A $\mathsf{type} \mid \Pi\ b_0\ \bot$ and so B $\mathsf{type} \mid b_0$. So we have B $\mathsf{type} \mid b$.

For $N : B \mid v : b$ we show $F(N)$ $\mathsf{type} \mid f(v)$ by showing that $F(N)$ $\mathsf{type} \mid w_0$ for any $w_0 \leqslant f(v)$ finite in Type. By the remark on continuity of evaluation at the end of Section 2, we can find $b_0 \leqslant b$ finite in Type and $v_0 \leqslant v$ finite in $El\ b_0$ and $f_0 \leqslant f$ finite in $El\ b_0 \to \mathsf{Type}$ such that $w_0 \leqslant f_0(v_0)$ in Type. We then have A $\mathsf{type} \mid \Pi\ b_0\ f_0$ and $N : B \mid v_0 : b_1$ for some $b_1 \leqslant b$ finite in Type. We can assume $b_0 \leqslant b_1$, maybe changing b_1 to $b_0 \vee b_1$, and using Lemma 4.1.3. By Lemma 4.1.4, we also have $N : B \mid v_0 : b_0$ and hence $F(N) \mid f_0(v_0)$ as needed to be shown.

The last assertion about conversion has a similar proof.

Conversely assume $A \to^*_{\mathsf{type}} \Pi(x : B)F$ with B $\mathsf{type} \mid b$ and $N : B \mid v : b$ implies $F(N) \mid f(v)$ and N conv $N' : B \mid v : b$ implies $F(N)$ conv $F(N') \mid f(v)$. We show that A $\mathsf{type} \mid \Pi\ b_0\ f_0$ for all finite $\Pi\ b_0\ f_0 \leqslant \Pi\ b\ f$ in Type. We have B $\mathsf{type} \mid b_0$ by definition, and if $N : B \mid v : b_0$ with v in $El\ b_0$ finite, then $N : B \mid v : b$ and so $F(N)$ $\mathsf{type} \mid f(v)$ and hence $F(N)$ $\mathsf{type} \mid f_0(v)$. We prove similarly that N conv $N' : B \mid v : b_0$ with v finite implies $F(N)$ conv $F(N') \mid f_0(v)$. □

Proposition 4.3 *Given A $\mathsf{type} \mid \Pi\ b\ f$ and $A \to^*_{\mathsf{type}} \Pi(x : B)F$, we have $M : A \mid w : \Pi\ b\ f$ if, and only if, $N : B \mid v : b$ implies $M\ N : F(N) \mid w(v) : f(v)$ and N conv $N' : B \mid v : b$ implies $M\ N$ conv $M\ N' : F(N) \mid w(v) : f(v)$.*

Proof. Similar to the proof of Proposition 4.2. □

The two last propositions hold by definition if $\Pi\ b\ f$ is a *finite* element of Type. Note that we could not have used these propositions directly on general, maybe infinite, element as a definition of A $\mathsf{type} \mid \Pi\ b\ f$ since it might be that $f(v)$ is as complex as $\Pi\ b\ f$. The method we have used instead was thus first to define the relation A $\mathsf{type} \mid \Pi\ b\ f$ for $\Pi\ b\ f$ *finite*, and then extend this relation by "continuity" on general elements. This is similar to the use of "inclusive predicates" [13, 16], fundamental in denotational semantics.

Lemma 4.4 *The following properties hold.*

1. *If $A \to_{\mathsf{type}} A'$ and A' $\mathsf{type} \mid a$, then A $\mathsf{type} \mid a$ and A conv $A' \mid a$.*
2. *If A $\mathsf{type} \mid a$ and $M \to_A M'$ and $M' : A \mid u : a$, then $M : A \mid u : a$ and M conv $M' : A \mid u : a$.*

Proof. Both properties are shown by induction on a. The most interesting case is for the second assertion when $a = \Pi\ b\ f$. We then have $A \to^*_{\mathsf{type}} \Pi(x : B)F$ with B $\mathsf{type} \mid b$. If $N : B \mid v : b$, we have $M\ N \to_{F(N)} M'\ N$ and $M'\ N : F(N) \mid u(v) : f(v)$. By induction, we have $M\ N : F(N) \mid u(v) : f(v)$ and $M\ N$ conv $M'\ N : F(N) \mid u(v) : f(v)$. Similarly, if N conv $N' \mid v : b$, we get $M\ N'$ conv $M'\ N' : F(N) \mid u(v) : f(v)$. Since $M' : A \mid u : a$ we have $M'\ N$ conv $M'\ N' : F(N) \mid u(v) : f(v)$ and we get by transitivity and symmetry $M\ N$ conv $M\ N' : F(N) \mid u(v) : f(v)$. □

We write $\sigma : \Gamma \mid \rho$ to mean that we have $A\sigma$ $\mathsf{type} \mid [\![A]\!]\rho$ and $\sigma(x) : A\sigma \mid \rho(x) : [\![A]\!]\rho$ for all $x : A$ in Γ. Note that, in particular, this implies that ρ fits Γ.

Similarly, we write σ conv $\sigma' : \Gamma \mid \rho$ to mean that we have $A\sigma$ conv $A\sigma' \mid [\![A]\!]\rho$ and $\sigma(x)$ conv $\sigma'(x) : A\sigma \mid \rho(x) : [\![A]\!]\rho$ for all $x : A$ in Γ.

Theorem 4.5 *The following properties hold, given $\sigma : \Gamma \mid \rho$ and σ conv $\sigma' : \Gamma \mid \rho$.*

1. *If $\Gamma \vdash A$ type, then $A\sigma$ $\mathsf{type} \mid [\![A]\!]\rho$.*
2. *If $\Gamma \vdash M : A$, then $M\sigma : A\sigma \mid [\![M]\!]\rho : [\![A]\!]\rho$.*
3. *If $\Gamma \vdash A$ conv A', then $A\sigma$ conv $A'\sigma \mid [\![A]\!]\rho$.*
4. *If $\Gamma \vdash M$ conv $M' : A$, then $M\sigma$ conv $M'\sigma : A\sigma \mid [\![M]\!]\rho : [\![A]\!]\rho$.*
5. *If $\Gamma \vdash A$ type, then $A\sigma$ conv $A\sigma' \mid [\![A]\!]\rho$.*
6. *If $\Gamma \vdash M : A$, then $M\sigma$ conv $M\sigma' : A\sigma \mid [\![A]\!]\rho$.*

Proof. This follows from Propositions 4.2 and 4.3 and Lemma 4.4, and the fact that weak-head reduction is stable under substitution. □

Corollary 4.6 *If* $\mathsf{0}$ conv $M : \mathsf{N}$*, then* $M \to^*_{\mathsf{N}} \mathsf{0}$*. If* $\mathsf{S}\ M_0$ conv $M : \mathsf{N}$*, then* $M \to^*_{\mathsf{N}} \mathsf{S}\ M_1$ *with* M_0 conv $M_1 : \mathsf{N}$*. If* A_0 conv $\Pi(x : B_1)F_1$*, then* $A_0 \to^*_{\mathsf{type}} \Pi(x : B_0)F_0$ *with* B_0 conv B_1 *and* $N : B_0$ *implies* $F_0(N)$ conv $F_1(N)$*.*

Proof. For the first statement, we have $[\![M]\!] = [\![\mathsf{0}]\!] = \mathsf{0}$. Using the previous theorem, we get $M : \mathsf{N} \mid [\![M]\!] : \mathsf{N}$ that is $M : \mathsf{N} \mid \mathsf{0} : \mathsf{N}$, which means precisely $M \to^*_{\mathsf{N}} \mathsf{0}$. The proof of the second statement is similar.

If A_0 conv $\Pi(x : B_1)F_1$, then $[\![A_0]\!] = [\![\Pi(x : B_1)F_1]\!] = \Pi\ [\![B_1]\!]\ [\![\lambda(x : B_1)F_1]\!]$ and we have A_0 conv $\Pi(x : B_1)F_1 \mid [\![A_0]\!]$ by Theorem 4.5. It follows that $A_0 \to^*_{\mathsf{type}} \Pi(x : B_0)F_0$ and B_0 conv $B_1 \mid [\![B_0]\!]$ and $N : B_0 \mid v : [\![B_1]\!]$ implies $F_0(N)$ conv $F_1(N) \mid [\![F_0]\!](x = v)$. In particular, we have B_0 conv B_1 and, for $v = \bot$, we have $F_0(N)$ conv $F_1(N)$ if $N : B_0$. □

Note that we cannot conclude that dependent product is one-to-one for conversion yet, since in the last case we get only that $N : B_0$ implies $F_0(N)$ conv $F_1(N)$, for $N : B_0$ *closed*, which is not enough to conclude F_0 conv F_1 $(x : B_0)$. A simple modification of our argument will apply however, as we shall see in the next section.

5 Connecting syntax and semantics, second version

We fix a context $\Delta = x_1 : A_1, \dots, x_n : A_n$. Working in this context Δ corresponds to extend the type system with constants $c_i : T_i$ with $T_i = A_i(x_1/c_1, \dots, x_{i-1}/c_{i-1})$. We define the interpretation of these constants by taking $[\![c_i]\!] = \bot$.

We then have $c_1 : T_1 \mid [\![c_1]\!] : [\![T_1]\!]$, $c_2 : T_2 \mid [\![c_2]\!] : [\![T_2]\!], \dots$. All the reasoning of the previous section applies with this addition of constants c_i. We deduce in particular:

Proposition 5.1 *If* $\Delta \vdash A_0$ conv $\Pi(x : B_1)F_1$*, then* $A_0 \to^*_{\mathsf{type}} \Pi(x : B_0)F_0$ *with* $\Delta \vdash B_0$ conv B_1 *and* $\Delta \vdash N : B_0$ *implies* $\Delta \vdash F_0(N/x)$ conv $F_1(N/x)$*.*

Note that for this proposition the context Δ is *arbitrary.*

Corollary 5.2 *If* $\Delta \vdash A_0$ conv $\Pi(x : B_1)F_1$*, then* $A_0 \to^*_{\mathsf{type}} \Pi(x : B_0)F_0$ *with* $\Delta \vdash B_0$ conv B_1 *and* $\Delta, x : B_0 \vdash F_0$ conv F_1*.*

Proof. Since all judgments stay valid by extension of the context, we also then have $\Delta, x : B_0 \vdash A_0$ conv $\Pi(x : B_1)F_1$. We can then apply the previous proposition changing Δ to $\Delta, x : B_0$ and taking $u = x$. □

As in [2], an important application of the injectivity of dependent product for conversion is the following result.

Corollary 5.3 *If* A type *and* $A \to A'$ *then* A' type *and* A conv A'*. If* $M : A$ *and* $M \to M'$ *then* $M' : A$ *and* M conv $M' : A$*.*

6 Connecting syntax and semantics, third version

We refine the domain as follows

$$D \ = \ [D \to D] + \Pi\ D\ [D \to D] + \mathsf{U}_i + \mathsf{N} + \mathsf{0} + \mathsf{S}\ D + \mathsf{T}$$

and we add the following typing rules:

$$\frac{}{\mathsf{T}\ \mathsf{type}} \qquad \frac{}{\mathsf{T} : \mathsf{U}_i} \qquad \frac{}{\mathsf{T} : \mathsf{N}} \qquad \frac{}{\mathsf{T} : \mathsf{T}}$$

We extend the application operation $u(v)$ by taking $\mathsf{T}(v)$ to be T for any value T. An operational intuition about T is that it represents the semantics of an "exception". We also extend the definition of rec by $\mathsf{rec}(d_0, d_1)\ \mathsf{T} = \mathsf{T}$. We finally refine the definition of the projection function by adding the clauses

$$\begin{array}{lcl} p\ \mathsf{N}\ \mathsf{T} & = & \mathsf{T} \\ p\ \mathsf{U}_j\ \mathsf{T} & = & \mathsf{T} \\ p\ \mathsf{T}\ \mathsf{T} & = & \mathsf{T} \\ p_{\mathsf{type}}\ \mathsf{T} & = & \mathsf{T} \end{array}$$

We now introduce the special class of "neutral" terms

$$k, K \ ::= \ c_i \mid k\ N \mid \mathsf{rec}(\lambda x\ A, M, M)\ k$$

and the predicate $G(k)$ of "typable" neutral terms, which is defined by the following clauses, where we define at the same time the type function $\tau(k)$:

1. Any constant c_i is typable, $G(c_i)$, and $\tau(c_i) = T_i$ is the given type of c_i.
2. If $G(k)$ and $\tau(k) \rightarrow^*_{\mathsf{type}} \Pi(x : B)F$ and $N : B$, then $G(k\ N)$ and $\tau(k\ N) = F(N)$
3. If $G(k)$ and $\tau(k) \rightarrow^*_{\mathsf{type}} \mathsf{N}$ and $T\ \mathsf{type}\ (x : \mathsf{N})$ and $M_0 : T(\mathsf{0})$ and $M_1 : \Pi(x : \mathsf{N})(T \rightarrow T(\mathsf{S}\ x))$, then $G(\mathsf{rec}(\lambda x\ T, M_0, M_1)\ k)$ and $\tau(\mathsf{rec}(\lambda x\ T, M_0, M_1)\ k) = T(k)$

We define next an equivalence relation $Q(k, k')$ on elements satisfying G by the clauses:

1. $Q(c_i, c_i)$
2. If $Q(k, k')$ and $\tau(k) \rightarrow^*_{\mathsf{type}} \Pi(x : B)F$ and $N : B$ and $\tau(k') \rightarrow^*_{\mathsf{type}} \Pi(x : B')F'$ and $N' : B'$ and $B\ \mathsf{conv}\ B'$ and $F\ \mathsf{conv}\ F'\ (x : B)$, then $Q(k\ N, k'\ N')$.
3. If $Q(k, k')$ and $\tau(k) \rightarrow^*_{\mathsf{type}} \mathsf{N}$ and $\tau(k') \rightarrow^*_{\mathsf{type}} \mathsf{N}$ and $T\ \mathsf{conv}\ T'\ (x : \mathsf{N})$ and $M_0\ \mathsf{conv}\ M_0' : T(\mathsf{0})$ and $M_1\ \mathsf{conv}\ M_1' : \Pi(x : \mathsf{N})(T \rightarrow T(\mathsf{S}\ x))$, then $Q(\mathsf{rec}(\lambda x\ T, M_0, M_1)\ k, \mathsf{rec}(\lambda x\ T', M_0', M_1')\ k')$.

We refine then the definitions of $J \mid a$ and $J \mid u : a$ by the clauses:

1. $A\ \mathsf{type} \mid \mathsf{T}$ means that $A \rightarrow^*_{\mathsf{type}} K$ for some K
2. $A\ \mathsf{conv}\ A' \mid \mathsf{T}$ means that $A\ \mathsf{conv}\ A'$
3. $M : A \mid \mathsf{T} : a$, where a is T or U_i or N, means $M \rightarrow^*_A k$ for some k
4. $M\ \mathsf{conv}\ M' : A \mid \mathsf{T} : a$, where a is T or U_i or N, means $M\ \mathsf{conv}\ M' : A$

Lemma 6.1 *If $G(k)$ and $\tau(k)\ \mathsf{type} \mid a$, then $k : \tau(k) \mid p\ a\ \mathsf{T} : a$. If $Q(k, k')$ and $\tau(k)\ \mathsf{type} \mid a$, then $k\ \mathsf{conv}\ k' : \tau(k) \mid p\ a\ \mathsf{T} : a$.*

Proof. By induction on $a\ \mathsf{type}$. Let us for instance prove the first assertion in the case where $a = \Pi\ b\ f$. We have $\tau(k) \rightarrow^*_{\mathsf{type}} \Pi(x : B)F$ with $B\ \mathsf{type} \mid b$ and $N : B \mid v : b$ implies $F(N)\ \mathsf{type} \mid f(v)$ and $N\ \mathsf{conv}\ N' : B \mid v : b$ implies $F(N)\ \mathsf{conv}\ F(N') \mid f(v)$. It follows that $N : B \mid v : b$ implies $G(k\ N)$ and $\tau(k\ N) = F(N)$, so that $k\ N : F(N) \mid p\ (f(v))\ \mathsf{T} : f(v)$ by induction. Similarly we show that $N\ \mathsf{conv}\ N' \mid v : b$ implies $Q(k\ N, k\ N')$ and so $k\ N\ \mathsf{conv}\ k\ N' : F(N) \mid p\ (f(v))\ \mathsf{T} : f(v)$ by induction. $\square$

We explain now how the semantics of the constants $c_1 : T_1,\ c_2 : T_2, \ldots$. We take $[\![c_1]\!] = p\ [\![T_1]\!]\ \mathsf{T}$ and then $[\![c_2]\!] = p\ [\![T_2]\!]\ \mathsf{T}$ and so on. This is justified since T_1 does not refer to any constant c_i and T_2 refers at most to the constant c_1, and so on. It follows from the last lemma that we have $T_i\ \mathsf{type} \mid [\![T_i]\!]$ and $c_i : T_i \mid [\![c_i]\!] : [\![T_i]\!]$.

Theorem 4.5 holds then with this semantics, since it holds for the constants c_i.

We then have the following application, using as in the previous section the fact that the context Δ is arbitrary.

Theorem 6.2 *If $\Delta \vdash M\ \mathsf{conv}\ N : A$, then $M : A$ and $N : A$ have the same Böhm tree.*

Proof. Corollary 5.3 implies that a term is convertible to (and hence as the same semantics as) its weak head normal form. Theorem 4.5 shows then that, given any two convertible terms, if one has a weak head normal form, so does the other term and these weak head normal form have the same shape. $\square$

Conclusion

We have shown that constructors are one-to-one for dependent type theory with conversion as judgment *and* η-conversion in a weak metatheory, while all existing proofs [2] use strong logical principles. Our argument applies as well to *partial* type theory, where we may have non terminating computations. An example is given in the reference [12]: one introduces a new base type Ω, which is like the type of natural numbers N with $\mathsf{0}$ deleted, and an element $\omega : \Omega$ such that $\omega \ \mathsf{conv} \ \mathsf{S}\ \omega : \Omega$. The type Ω will be represented by a new finite element of the domain, while the element ω will be the least upper bound of the sequence $\bot,\ \mathsf{S}\ \bot,\ \mathsf{S}\ (\mathsf{S}\ \bot), \dots$

Using strong logical principles, it should be possible to define a semantical notion of totality on elements of the domain, and prove that a total element corresponds to a finite Böhm tree. If we are only interested by the evaluation of closed expressions, the techniques we have presented are enough to show canonicity of type theory extended with bar recursion, as in [9], but with η-conversion in the type system.

On the other hand it is not clear how to extend the present method to a type system with a type of all types. Do we still have adequacy in this case?

References

[1] A. Abel, Th. Coquand and P Dybjer. Normalization by evaluation for Martin-Löf type theory with typed equality judgements. Proceedings of LICS '07, 2007.

[2] A. Abel and G. Scherer. On Irrelevance and Algorithmic Equality in Predicative Type Theory. Logical Methods in Computer Science, 8(1):1-36, 2012.

[3] P. Aczel. On Relating Type Theories and Set Theories. Types for Proofs and Programs, LNCS 1657, pp 1-18, 1998.

[4] R. Adams. Pure Type Systems with Judgemental Equality. Journal of Functional. Programming 16 (2): 219-246, 2006.

[5] U. Berger Realisability for Induction and Coinduction with Applications to Constructive Analysis. J. UCS 16 (18), 2535-2555, 2010.

[6] L. Cardelli. A Polymorphic λ-calulus with Type:Type. SRC Research Report 10, 1986.

[7] Th. Coquand. An algorithm for testing conversion in type theory. In *Logical Frameworks*, p. 255-279, 1991.

[8] Th. Coquand, C. Gunter and G. Winskel. Domain theoretic models of polymorphisms. Information and Computation, Vol. 81, p. 123-167, 1989.

[9] Th. Coquand and A. Spiwack. A proof of strong normalisation using domain theory. Proceeding of LICS '06, Pages 307-316, 2006.

[10] P. Martin-Löf. Constructive Mathematics and Computer Programming. Studies in Logic and the Foundations of Mathematics, Volume 104, p. 153-175, 1982.

[11] P. Martin-Löf. Lecture notes on the domain interpretation of type theory. Workshop on semantics of programming languages, Chalmers University, Göteborg, August 1983.

[12] P. Martin-Löf. Unifying Scott's theory of domains for denotational semantics and intuitionistic type theory (Abstract). Atti del Congresso *Logica e Filosofia della Scienza* San Gimignano, December 1983. Vol. I.

[13] R. Milne. *The formal semantics of computer languages and their implemnetation.* PhD thesis, Oxford University Computing Laboratory, 1974.

[14] E. Palmgren, V. Stoltenberg-Hansen. Domain interpretations of Martin-Löf's partial type theory. Annals of Pure and Applied Logic, Volume 48, Issue 2, p. 135-196, 1990.

[15] E. Palmgren, V. Stoltenberg-Hansen. Remarks on Martin-Löf's partial type theory. BIT 32, p. 70-82, 1992.

[16] J. Reynolds. On The Relation Between Direct and Continuation Semantics. Proceedings of the Second Colloquim on Automata, Languages and Programming, p. 141-156, 1974.

[17] H. Schwichtenberg and S.S. Wainer. *Proofs and Computations (Perspectives in Logic).* Cambridge University Press, 2012.

[18] V. Siles and H. Herbelin. Equality is typable in Semi-Full Pure Type Systems. Proceedings of LICS '10, 2010.

[19] D. Scott. Lectures on a Mathematical Theory of Computation. Oxford University PRG Technical Monograph, 1981.

[20] D. Scott. Some ordered sets in computer science. NATO Advanced Study Institutes Series book series (ASIC, volume 83), p. 677-718, 1982.

Appendix: typing and conversion rules of type theory

Rules for contexts

$$\frac{\Gamma \vdash A\ \mathsf{type}}{\Gamma, x : A \vdash} \qquad \frac{}{() \vdash} \qquad \frac{\Gamma \vdash}{\Gamma \vdash x : A}\ (x{:}A\ \mathit{in}\ \Gamma)$$

Like in [10], we don't write explicitly the context in the next rules.

Typing rules

$$\frac{M : A \qquad A\ \mathsf{conv}\ B}{M : B}$$

$$\frac{A\ \mathsf{type} \qquad B\ \mathsf{type}\ (x : A)}{\Pi(x : A)B\ \mathsf{type}} \qquad \frac{A : \mathsf{U}_i \qquad B : \mathsf{U}_i\ (x : A)}{\Pi(x : A)B : \mathsf{U}_i}$$

$$\frac{A\ \mathsf{type} \qquad B\ \mathsf{type}\ (x : A) \qquad N : B\ (x : A)}{\lambda(x : A)N : \Pi(x : A)B} \qquad \frac{A\ \mathsf{type} \qquad B\ \mathsf{type}\ (x : A) \qquad N : \Pi(x : A)B \qquad M : A}{N\ M : B(x/M)}$$

$$\frac{A : \mathsf{U}_i}{A : \mathsf{U}_j}\, i < j \qquad \frac{}{\mathsf{U}_i : \mathsf{U}_j}\, i < j \qquad \frac{A : \mathsf{U}_i}{A\ \mathsf{type}}$$

Conversion rules

$$\frac{M\ \mathsf{conv}\ N : A \qquad A\ \mathsf{conv}\ B}{M\ \mathsf{conv}\ N : B} \qquad \frac{M\ \mathsf{conv}\ N : \mathsf{U}_i}{M\ \mathsf{conv}\ N : \mathsf{U}_j}\, i < j \qquad \frac{M\ \mathsf{conv}\ N : \mathsf{U}_i}{M\ \mathsf{conv}\ N}$$

$$\frac{A\ \mathsf{type}}{A\ \mathsf{conv}\ A} \qquad \frac{A\ \mathsf{conv}\ C \qquad B\ \mathsf{conv}\ C}{A\ \mathsf{conv}\ B} \qquad \frac{M : A}{M\ \mathsf{conv}\ M : A} \qquad \frac{M\ \mathsf{conv}\ P : A \qquad N\ \mathsf{conv}\ P : A}{M\ \mathsf{conv}\ N : A}$$

$$\frac{A_0\ \mathsf{conv}\ A_1 \qquad B_0\ \mathsf{conv}\ B_1\ (x : A_0)}{\Pi(x : A_0)B_0\ \mathsf{conv}\ \Pi(x : A_1)B_1} \qquad \frac{A_0\ \mathsf{conv}\ A_1 : \mathsf{U}_i \qquad B_0\ \mathsf{conv}\ B_1 : \mathsf{U}_i\ (x : A_0)}{\Pi(x : A_0)B_0\ \mathsf{conv}\ \Pi(x : A_1)B_1 : \mathsf{U}_i}$$

$$\frac{A\ \mathsf{type} \qquad B\ \mathsf{type}\ (x : A) \qquad N\ \mathsf{conv}\ N' : \Pi(x : A)B \qquad M : A}{N\ M\ \mathsf{conv}\ N'\ M : B(x/M)}$$

$$\frac{A\ \mathsf{type} \qquad B\ \mathsf{type}\ (x : A) \qquad N : \Pi(x : A)B \qquad M\ \mathsf{conv}\ M' : A}{N\ M\ \mathsf{conv}\ N\ M' : B(x/M)}$$

$$\frac{A\ \mathsf{type} \qquad B\ \mathsf{type}\ (x : A) \qquad N : B\ (x : A) \qquad M : A}{(\lambda(x : A)N)\ M\ \mathsf{conv}\ N(x/M) : B(x/M)}$$

$$\frac{A\ \mathsf{type} \qquad B\ \mathsf{type}\ (x : A) \qquad N : \Pi(x : A)B \qquad N' : \Pi(x : A)B \qquad N\ x\ \mathsf{conv}\ N'\ x : B\ (x : A)}{N\ \mathsf{conv}\ N' : \Pi(x : A)B}$$

Rules for natural numbers

$$\frac{}{\mathsf{N} : \mathsf{U}_0} \qquad \frac{}{\mathsf{0} : \mathsf{N}} \qquad \frac{M : \mathsf{N}}{\mathsf{S}\ M : \mathsf{N}}$$

$$\frac{T\ \mathsf{type}\ (x : \mathsf{N}) \qquad M_0 : T(\mathsf{0}) \qquad M_1 : \Pi(x : \mathsf{N})\ (T \to T(\mathsf{S}\ x))}{\mathsf{rec}(\lambda x\ T, M_0, M_1) : \Pi(x : \mathsf{N})T}$$

$$\mathsf{rec}(\lambda x\ T, M_0, M_1)\ \mathsf{0}\ \mathsf{conv}\ M_0 \qquad \mathsf{rec}(\lambda x\ T, M_0, M_1)\ (\mathsf{S}\ n)\ \mathsf{conv}\ M_1\ n\ (\mathsf{rec}(\lambda x\ T, M_0, M_1)\ n)$$