No need to be implicit!

Thiago Felicissimo

February 2022

Abstract

We consider a variant of Pure Type Systems in which the parameters of the dependent product type are made explicit. This variant, which we call Explicitly-Typed Pure Type Systems, is then shown equivalent to the traditional definition. This result is of particular interest for doing encodings of Pure Type Systems in logical frameworks, as in these settings all the type parameters of constructors and eliminators are represented explicitly in the syntax, and thus need to be provided.

1 Introduction

In the literature, it is most common to write $\lambda x : A.M$ for abstraction and MN for application. However, if we remind ourselves that the dependent product can be seen as a (negative) inductive type, we realize that this regular notation actually uses implicit arguments. For instance, while the first projection of a pair is normally written as $\pi^1(p)$, if we look at the full type of π^1 (for instance in Coq), we see that this is an implicit notation for $\pi^1(A, B, p)$, where $A \times B$ is the type of p.

Likewise, if we wanted to write the dependent product type with explicit paremeters, then the full form of abstraction and application would be $\lambda(A, {}_{[x]}B, {}_{[x]}M)$ and $@(A, {}_{[x]}B, M, N)$. Furthermore, if $- \times -$ is universe-polymorphic, we should also write $\pi^1_{s_A,s_B}(A, B, p)$ to explicit the sort parameters. Because in PTSs the dependent product type is used with different sorts, it is also a universe-polymorphic definition, thus we should also write $\lambda_{s_A,s_B}(A, {}_{[x]}B, {}_{[x]}M)$, $@_{s_A,s_B}(A, {}_{[x]}B, M, N)$ and $\Pi_{s_A,s_B}(A, {}_{[x]}B)$.

A natural question we can then ask ourselves is how this implicit information changes the theory, and a good setting to address this question in a general way is within Pure Type Systems (PTSs). Such an equivalence would not only be interesting from a theoretical perspective, but also for practical reasons, for instance when representing PTSs in logical frameworks, such as in [3]. In this work, we show that those two settings are indeed equivalent when restraining to functional PTSs, which make the quasi totality of PTSs used in practice.

Related works

This is not the first work to consider variants of PTSs in which some of the type parameters are made explicit in the syntax. In [6] they consider as an auxiliary system a presentation of PTSs with typed conversion and in which only applications have their parameters explicit. For instance, they represent $@_{s_A,s_B}(A, {}_{[x]}B, M, N)$ as $M_{\Pi x:A.B}N$, but $\lambda_{s_A,s_B}(A, {}_{[x]}B, {}_{[x]}M)$ is represented as usual, as $\lambda x : A.M$.

They then show that this version is equivalent to the standard one. In our work, we restrict

our investigation to the case of functional PTSs, but we explicit all the parameters, even on abstractions, and also the sorts. However, their proofs were of great inspiration to our ones.

In [5] they consider a version in which the parameters A, B are explicit everywhere, and in particular B is also explicit in abstractions. However, s_A, s_B remain implicit — for instance, they represent $\lambda_{s_A,s_B}(A, [x]B, M)$ as $\lambda_{x:A.B}M$. Then they also show that this presentation is similar to the traditional one, under the assumption of strong normalization. Differently from their work, here we do not assume this hypothesis, and we also explicit the sort parameters.

In [2], to show the existence of η-long forms for PTSs in the lambda cube, they consider a notion of marked terms, in which they mark each variable, abstraction and application with their domain types. They then show that such presentation is equivalent for PTSs in the cube.

Our point of view is quite different, as in our presentation we don't annotate terms with their types, but instead constructors and eliminators with their parameters. For instance, they annotate variables with their types, which we do not do as variables do not take parameters. Moreover, this means that some parameters are still implicit. For instance, if $M : \Pi x : A.B$, then an application is represented in their system as $(MN)^{B\{N/x\}}$, but from $B\{N/x\}$ we cannot extract neither A or B.

2 Pure Type Systems

$$\frac{}{- \text{ well-formed}}\ \textsc{Empty} \qquad \frac{\Gamma \vdash A : s \qquad x \notin \Gamma}{\Gamma, x : A \text{ well-formed}}\ \textsc{Decl}$$

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : s \qquad A \equiv B}{\Gamma \vdash M : B}\ \textsc{Conv} \qquad \frac{\Gamma \text{ well-formed} \qquad x : A \in \Gamma}{\Gamma \vdash x : A}\ \textsc{Var}$$

$$\frac{\Gamma \text{ well-formed} \qquad (s_1, s_2) \in \mathcal{A}}{\Gamma \vdash s_1 : s_2}\ \textsc{Sort} \qquad \frac{\Gamma \vdash N : A \qquad \Gamma \vdash M : \Pi x : A.B}{\Gamma \vdash MN : B\{N/x\}}\ \textsc{App}$$

$$\frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2 \qquad (s_1, s_2, s_3) \in \mathcal{R}}{\Gamma \vdash \Pi x : A.B : s_3}\ \textsc{Prod}$$

$$\frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2 \qquad \Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x : A.M : \Pi x : A.B}\ \textsc{Abs}$$

Typing rules for Pure Type Systems (PTS)

For completeness purposes, we start by recalling the traditional definition[1] of Pure Type Systems (PTS)[1]. A PTS is parameterized by a sort specification $(\mathcal{S}, \mathcal{A}, \mathcal{R})$, where $\mathcal{S}$ is a set, $\mathcal{A} \subseteq \mathcal{S}^2$ and $\mathcal{R} \subseteq \mathcal{S}^3$. Terms are defined by the following syntax.

$$A, B, M, N ::= x \mid s \mid \Pi x : A.B \mid \lambda x : A.B \mid MN$$

[1]There are actually two traditional definitions of Pure Type Systems: one with a primitive notion of well-formed contexts, and one with a weakening rule. We chose here to use the first one, thought both are known to be equivalent.

Conversion is defined by the context closure of the rule

$$(\lambda x : A.M)N \hookrightarrow M\{N/x\}$$

for any terms A, M, N, and where $M\{N/x\}$ denotes the capture-avoiding substitution of x by N in M. Typing is defined by the previously presented rules. A PTS is said to be *functional* if the relations $\mathcal{A}, \mathcal{R}$ are functional, that is, they define partial functions $\mathcal{S} \dashrightarrow \mathcal{S}$ and $\mathcal{S} \dashrightarrow \mathcal{S} \times \mathcal{S}$ respectively.

3 Explicitly-Typed Pure Type Systems

We define the syntax of Explicitly-Typed Pure Type Systems by the following grammar, whereas x ranges in an infinite set of variables $\mathcal{V}$ and s, s_1, s_2 ranges in the set of sorts $\mathcal{S}$.

$$\begin{aligned} A, B, M, N ::= \ & | \ x \\ & | \ s \\ & | \ \Pi_{s_1,s_2}(A, [x]B) \\ & | \ \lambda_{s_1,s_2}(A, [x]B, [x]M) \\ & | \ @_{s_1,s_2}(A, [x]B, M, N) \end{aligned}$$

As discussed, the dependent product type and its constructor (abstraction) and eliminator (application) are explicitly-typed, and thus take the sort parameters s_1, s_2 and the regular parameters A, B. of the dependent product that is being constructed (with abstraction) or eliminated (with application). Note that in terms such as $\Pi_{s_1,s_2}(A, [x]B)$ the sorts s_1, s_2 are not terms, but indices of Π. Therefore, we have a symbol Π_{s_1,s_2} for each pair $s_1, s_2 \in \mathcal{S}$.

We now need to chose how to define β-reduction in this system. One could be tempted to take the rule

$$@_{s_1,s_2}(A, [x]B, \lambda_{s_1,s_2}(A, [x]B, [x]M), N) \hookrightarrow_\beta M\{N/x\}\,,$$

which requires the parameters A, B in λ and $@$ to be syntactically the same. However, as this rule is non-left linear, it is non-confluent in untyped terms, which leads to a much less behaved metatheory. Therefore we will prefer to define β-reduction with the linearized version of the rule, which is

$$@_{s_1,s_2}(A, [x]B, \lambda_{s_1,s_2}(A', [x]B', [x]M), N) \hookrightarrow_\beta M\{N/x\}\,.$$

Note that, when this rule is used with well-typed terms, the typing constraints ensure $A \equiv A'$ and $B \equiv B'$. Moreover, because s_1, s_2 are not terms but indices, this rule is indeed left-linear — actually, there is not only one β rule, but one for each pair $s_1, s_2 \in \mathcal{S}^2$.

We now present the typing rules of the system. Note that in the rules Abs and App we prefer to explicitly require that all the subterms appearing in the conclusion are well-typed, even

thought this would be ensured by inversion of the typing rules.

$$\frac{}{- \text{ well-formed}}\ \textsc{Empty} \qquad \frac{\Gamma \vdash A : s \qquad x \notin \Gamma}{\Gamma, x : A \text{ well-formed}}\ \textsc{Decl-ctx}$$

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash B : s \qquad A \equiv B}{\Gamma \vdash M : B}\ \textsc{Conv}$$

$$\frac{\Gamma \text{ well-formed} \qquad x : A \in \Gamma}{\Gamma \vdash x : A}\ \textsc{Var}$$

$$\frac{\Gamma \text{ well-formed} \qquad (s_1, s_2) \in \mathcal{A}}{\Gamma \vdash s_1 : s_2}\ \textsc{Sort}$$

$$\frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2 \qquad (s_1, s_2, s_3) \in \mathcal{R}}{\Gamma \vdash \Pi_{s_1,s_2}(A, [x]B) : s_3}\ \textsc{Prod}$$

$$\frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2 \qquad (s_1, s_2, s_3) \in \mathcal{R} \qquad \Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda_{s_1,s_2}(A, [x]B, [x]M) : \Pi_{s_1,s_2}(A, [x]B)}\ \textsc{Abs}$$

$$\frac{\Gamma \vdash A : s_1 \qquad \Gamma, x : A \vdash B : s_2 \qquad (s_1, s_2, s_3) \in \mathcal{R} \qquad \Gamma \vdash N : A \qquad \Gamma \vdash M : \Pi_{s_1,s_2}(A, [x]B)}{\Gamma \vdash @_{s_1,s_2}(A, [x]B, M, N) : B\{N/x\}}\ \textsc{App}$$

Typing rules for Explicitly-Typed Pure Type Systems (EPTS)

4 Metatheory of EPTSs

We now show that EPTSs satisfy the usual metaproperties of PTSs. We only sketch the proofs, as they follow the same idea as the standard ones for PTSs. Therefore this serves more as a sanity check.

In the following, we write $\Gamma \sqsubseteq \Gamma'$ when Γ is a subsequence of Γ', that is, Γ' extends Γ while preserving the order among the elements of Γ.

Proposition 4.1 (Basic properties) *EPTSs satisfy the following basic properties.*

1. *Confluence: If* $N_1 \longleftarrow^* M \longhookrightarrow^* N_2$ *then there is* M' *such that* $N_1 \longhookrightarrow^* M' \longleftarrow^* N_2$.
2. *Weakening: If* Γ' well-formed *and* $\Gamma \sqsubseteq \Gamma'$*, we have*
   - Γ well-formed $\Rightarrow \Gamma'$ well-formed
   - $\Gamma \vdash M : A \Rightarrow \Gamma' \vdash M : A$
3. *Well-formedeness of contexts: If* $\Gamma \vdash M : A$ *then* Γ well-formed *and forall* $x : A \in \Gamma$, $\Gamma \vdash A : s$.
4. *Conversion in context: If* $A \equiv A'$ *and* $\Gamma \vdash A' : s$ *then*
   - $\Gamma, x : A, \Gamma'$ well-formed $\Rightarrow \Gamma, x : A', \Gamma'$ well-formed
   - $\Gamma, x : A, \Gamma' \vdash M : B \Rightarrow \Gamma, x : A', \Gamma' \vdash M : B$

Proof. The untyped terms together with β form a Combinatory Reduction System which is orthogonal, and thus confluent (Corollary 13.6 of [4]). For the three other properties, they follow from a simple induction, with 3 using 2 and 4 using 3. ■

As usual, EPTSs satisfy inversion of typing.

Proposition 4.2 (Inversion) *If* $\Gamma \vdash M : C$ *then*

- *If* $M = x$, *then*
  - Γ well-formed *with a smaller derivation tree*
  - *there is* x *with* $x : A \in \Gamma$ *and* $C \equiv A$
- *If* $M = s$, *then there is* s' *with* $(s, s') \in \mathcal{A}$ *and* $C \equiv s'$
- *If* $M = \Pi_{s_1,s_2}(A, [x]B)$ *then*
  - $\Gamma \vdash A : s_1$ *with a smaller derivation tree*
  - $\Gamma, x : A \vdash B : s_2$ *with a smaller derivation tree*
  - *there is* s_3 *with* $(s_1, s_2, s_3) \in \mathcal{R}$ *and* $C \equiv s_3$
- *If* $M = \lambda_{s_1,s_2}(A, [x]B, [x]N)$ *then*
  - $\Gamma \vdash A : s_1$ *with a smaller derivation tree*
  - $\Gamma, x : A \vdash B : s_2$ *with a smaller derivation tree*
  - *there is* s_3 *with* $(s_1, s_2, s_3) \in \mathcal{R}$
  - $\Gamma, x : A \vdash N : B$ *with a smaller derivation tree*
  - $C \equiv \Pi_{s_1,s_2}(A, [x]B)$
- *If* $M = @_{s_1,s_2}(A, [x]B, N_1, N_2)$ *then*
  - $\Gamma \vdash A : s_1$ *with a smaller derivation tree*
  - $\Gamma, x : A \vdash B : s_2$ *with a smaller derivation tree*
  - *there is* s_3 *with* $(s_1, s_2, s_3) \in \mathcal{R}$
  - $\Gamma \vdash N_1 : A$ *with a smaller derivation tree*
  - $\Gamma \vdash N_2 : \Pi_{s_1,s_2}(A, [x]B)$ *with a smaller derivation tree*
  - $C \equiv B\{N_2/x\}$

Recall that a specification $(\mathcal{S}, \mathcal{A}, \mathcal{R})$ is said to be functional when the relations $\mathcal{A}$ and $\mathcal{R}$ are functional.

Proposition 4.3 (Uniqueness of types) *For functional EPTSs,* $\Gamma \vdash M : A$ *and* $\Gamma \vdash M : B$ *imply* $A \equiv B$.

Proof. By induction on $\Gamma \vdash M : A$, and using inversion on $\Gamma \vdash M : B$. The interesting cases are for rules Sort and Prod, where we use functionality of $\mathcal{A}, \mathcal{R}$ to conclude. ■

This also implies that types in functional EPTSs have a unique sort.

Corollary 4.1 (Uniqueness of sorts) *For functional EPTSs,* $\Gamma \vdash A : s$ *and* $\Gamma \vdash A : s'$ *then* $s = s'$.

Before showing subject reduction we start as usual with a substitution lemma.

Lemma 4.1 (Substitution lemma) *Let* $\Gamma \vdash N : A$. *We have*

- $\Gamma, x : A, \Gamma'$ well-formed $\Rightarrow \Gamma, \Gamma'\{N/x\}$ well-formed
- $\Gamma, x : A, \Gamma' \vdash M : B \Rightarrow \Gamma, \Gamma'\{N/x\} \vdash M\{N/x\} : B\{N/x\}$

Proof. By induction on the judgment derivation. For the case Conv we use that $A_1 \equiv A_2$ implies $A_1\{N/x\} \equiv A_2\{N/x\}$ and for the case App we use $B\{N/x\}\{M_1\{N/x\}/y\} = B\{M_1/y\}\{N/x\}$. ■

Proposition 4.4 (Well-sortness) *If* $\Gamma \vdash M : A$ *then* $\Gamma \vdash A : s$ *for some sort, or* A *is a top sort.*

Proof. By induction on $\Gamma \vdash M : A$. The only non-trivial case is with App, where we use the substitution lemma. ■

We can finally show that EPTSs satisfy subject reduction, as expected.

Theorem 4.1 (Subject reduction) *If* $\Gamma \vdash M : A$ *and* $M \hookrightarrow_\beta M'$ *then* $\Gamma \vdash M' : A$.

Proof. By induction on the rewrite context and using inversion. The only interesting non-base case is when the reduction takes place to the right of an application, in which we use the fact that $N \hookrightarrow N'$ implies $B\{N/x\} \hookrightarrow^* B\{N'/x\}$ and the rule Conv. For the base case, we use both the substitution lemma and the fact that $\Pi(A, {}_{[x]}B) \equiv \Pi(A', {}_{[x]}B')$ implies $A \equiv A'$ and $B \equiv B'$, which follows from confluence. ■

5 EPTS and PTS are equivalent

In this section we finally show the main result of this article: PTSs and EPTSs are indeed equivalent. As previously mentioned, we restrict our study to the case of functional PTSs.

In the first part we will show that any EPTS judgment $\Gamma \vdash_{EPTS} M : A$ can be erased into a regular PTS judgment. We call this direction *soundness*, as it means intuitively that an EPTS do not type more terms then its corresponding PTS.

We will then show the other direction and prove that for any PTS judgment $\Gamma \vdash_{PTS} M : A$ we can fill in the missing parameters and get Γ', M', A' such that $\Gamma' \vdash_{EPTS} M' : A'$. We call this direction *completeness*, as it means intuitively that an EPTS fully captures the typing relation of its corresponding PTS.

Finally, we will put all this together to show that, by quocienting out from the EPTS syntax the parameters normally left implicit, we have a bijection between the EPTS terms living in a type A and the PTS terms living in the erasure of A.

5.1 Soundness

As we will see, showing soundness is quite easy. We start by defining the erasure function from an EPTS to its corresponding PTS.

$$
\begin{aligned}
&|x| = x \\
&|s| = s \\
&|\Pi_{s_1,s_2}(A, [x]B)| = \Pi x : |A|.|B| \\
&|\lambda_{s_1,s_2}(A, [x]B, [x]M)| = \lambda x : |A|.|M| \\
&|@_{s_1,s_2}(A, [x]B, M, N)| = |M||N|
\end{aligned}
$$

Lemma 5.1 *The erasure satisfies the following basic properties.*

1. *Compositionality: For all M, N, we have $|M|\{|N|/x\} = |M\{N/x\}|$*
2. *Preservation of computation: For all M, N, if $M \hookrightarrow N$ then $|M| \hookrightarrow |N|$*
3. *Preservation of conversion: For all M, N, if $M \equiv N$ then $|M| \equiv |N|$*

Proof. 1. By induction in M.

2. By induction on the rewrite context, using part 1 for the base case.
3. By induction on $\equiv$, and using part 2. ■

Theorem 5.1 (Soundness) *We have the following*

- *If Γ well-formed$_{EPTS}$ then $|\Gamma|$ well-formed$_{PTS}$*
- *If $\Gamma \vdash_{EPTS} M : A$ then $|\Gamma| \vdash_{PTS} |M| : |A|$*

Proof. By induction on the judgment tree. The only non-trivial cases are App and Conv. For the case App we use the fact that $|B|\{|N|/x\} = |B\{N/x\}|$. For the case Conv we use the fact that $A \equiv B$ implies $|A| \equiv |B|$. ■

5.2 Completeness

To show completeness, we could try to define a function $[-] : \Lambda_{PTS} \to \Lambda_{EPTS}$ which fills in the missing parameters for each term $M \in \Lambda_{PTS}$. Because the parameters cannot be purely inferred from the syntax, this function should also depend on the typing context Γ in which M is well-typed.

We then could try to define $[-]_\Gamma$ by induction on M. For instance, we could try to define $[\lambda x : A.M]_\Gamma$ as $\lambda_{s_1,s_2}([A]_\Gamma, [x][B]_{\Gamma,x:A}, [M]_{\Gamma,x:A})$, where s_1, s_2, B are such that $\Gamma \vdash_{PTS} A : s_1$, $\Gamma, x : A \vdash_{PTS} B : s_2$ and $\Gamma, x : A \vdash_{PTS} M : B$. However, this definition is not well-founded, because we apply $[-]$ to B, which was not in the original term and thus can be arbitrarily large.

Instead of applying $[-]$ to B we could try taking a $B' \in \Lambda_{EPTS}$ satisfying $[\Gamma], x : [A]_\Gamma \vdash [M]_{\Gamma,x:A} : B'$, but at this stage there is no reason for us to say that such a B' can exists. Of

course, when we will have proven completeness we will know that there is indeed such a B', but we still do not know it yet.

Therefore, instead of defining a function $[-] : \Lambda_{PTS} \to \Lambda_{EPTS}$ we will use the erasure $|-|$ to prove completeness. We will proceed as it follows. First we will prove that for any $M, N \in \Lambda_{EPTS}$ well-typed with $|M| = |N|$, we have $M \equiv N$. Then we will use this to show that $|-|$ reflects conversion for well-typed terms: we have $|M| \equiv |N| \Rightarrow M \equiv N$. We will then use this to show completeness, by establishing that if $\Gamma \vdash_{PTS} M : A$ we can find Γ', M', A' with $|\Gamma'| = \Gamma, |M'| = M$ and $|A'| = A$ such that $\Gamma' \vdash_{EPTS} M' : A'$.

5.2.1 Preservation of computation

As we discussed, we will start by showing that for M, N well-typed, if $|M| = |N|$ then $M \equiv N$. However, we can actually be more precise and say that the conversion steps in M and N are only performed on the implicit parameter positions. Therefore, we start by defining this notion precisely.

Definition 5.1 (Implicit parameter step) *We write $M \hookrightarrow_I N$ when a reduction step is performed in one of the parameters that is usually implicit. More precisely, we should have a context $C(-)$, terms M', N' with $M' \hookrightarrow N'$ and be in one of the following cases.*

- $M = C(\lambda_{s_1,s_2}(A, [x]M', [x]P)) \hookrightarrow C(\lambda_{s_1,s_2}(A, [x]N', [x]P)) = N$
- $M = C(@_{s_1,s_2}(A, [x]M', P, P')) \hookrightarrow C(@_{s_1,s_2}(A, [x]N', P, P')) = N$
- $M = C(@_{s_1,s_2}(M', [x]B, P, P')) \hookrightarrow C(@_{s_1,s_2}(N', [x]B, P, P')) = N$

We then write $\hookrightarrow_I^*$ for the reflexive and transitive closure of $\hookrightarrow_I$, and we write $\equiv_I$ for the equivalence relation generated by it.

We will also need the following auxiliary lemma, which can be easily shown using subject reduction and conversion in context.

Lemma 5.2 *If $\Gamma, x : A, \Gamma' \vdash M : B$ and $A \hookrightarrow^* A'$ then $\Gamma, x : A', \Gamma' \vdash M : B$.*

We can now proceed with the proof of the following proposition. In the following, we write $\Gamma \vdash M$ *typed* when there is A such that $\Gamma \vdash M : A$. Likewise, we write $\Gamma \vdash A$ *type* if there is M such that $\Gamma \vdash M : A$.

Proposition 5.1 *If $\Gamma \vdash_{EPTS} M, M'$ typed and $|M| = |M'|$ then there is M'' with $M \hookrightarrow_I^* M'' \hookleftarrow_I^* M'$ with $|M''| = |M| = |M'|$.*

Proof. By induction on M, the base cases being trivial.

$M = \Pi_{s_1,s_2}(A, [x]B)$: Then $M' = \Pi_{s_1',s_2'}(A', [x]B')$ with $|A| = |A'|$ and $|B| = |B'|$. By inversion we have

$$\Gamma \vdash A, A' \text{ typed} \tag{1}$$
$$\Gamma, x : A \vdash B \text{ typed} \tag{2}$$
$$\Gamma, x : A' \vdash B' \text{ typed} \tag{3}$$

Using (1) and $|A| = |A'|$ we apply the IH to we deduce that there is A'' satisfying $A \hookrightarrow_I^* A'' \hookleftarrow_I^* A'$ (implying $s_1 = s_1'$) and $|A''| = |A| = |A'|$. From this and Lemma 5.2 applied to (2) and (3), we get $\Gamma, x : A'' \vdash B, B'$ *typed*, and thus using $|B| = |B'|$ we can apply the IH to get a B'' satisfying $B \hookrightarrow_I^* B'' \hookleftarrow_I^* B'$ (implying $s_2 = s_2'$) and $|B''| = |B| = |B'|$. Finally, now we can take $M'' = \Pi_{s_1,s_2}(A'', [x]B'')$ and thus we have $M \hookrightarrow_I^* M'' \hookleftarrow_I^* M'$ and $|M''| = |M| = |M'|$.

$\mathbf{M = \lambda_{s_1,s_2}(A, [x]B, [x]N)}$: Then $M' = \lambda_{s_1',s_2'}(A', [x]B', [x]N')$ with $|A| = |A'|$ and $|N| = |N'|$. By inversion we have

$$\Gamma \vdash A, A' \textit{ typed} \tag{1}$$
$$\Gamma, x : A \vdash N : B \tag{2}$$
$$\Gamma, x : A' \vdash N' : B' \tag{3}$$

Using (1) and $|A| = |A'|$ we apply the IH to we deduce that there is A'' satisfying $A \hookrightarrow_I^* A'' \hookleftarrow_I^* A'$ (implying $s_1 = s_1'$) and $|A''| = |A| = |A'|$. From this and Lemma 5.2 applied to (2) and (3) we get

$$\Gamma, x : A'' \vdash N : B \tag{4}$$
$$\Gamma, x : A'' \vdash N' : B' \tag{5}$$

and thus using the IH and $|N| = |N'|$ we get a N'' with $N \hookrightarrow_I^* N'' \hookleftarrow_I^* N'$ and $|N''| = |N| = |N'|$. By subject reduction applied to (4) and (5) we also have $\Gamma, x : A'' \vdash N'' : B, B'$, and thus by Proposition 4.3 we have $B \equiv B'$. By confluence, there is B'' with $B \hookrightarrow^* B'' \hookleftarrow^* B'$ (which implies $s_2 = s_2'$). Finally, now we can take $\lambda_{s_1,s_2}(A'', [x]B'', [x]N'')$ and we have $M \hookrightarrow_I^* M'' \hookleftarrow_I^* M'$ and $|M''| = |M| = |M'|$.

$\mathbf{M = @_{s_1,s_2}(A, [x]B, N, P)}$: Then $M' = @_{s_1',s_2'}(A', [x]B', N', P')$ with $|N| = |N'|$ and $|P| = |P'|$. By inversion we have

$$\Gamma \vdash N : \Pi_{s_1,s_2}(A, [x]B) \tag{1}$$
$$\Gamma \vdash N' : \Pi_{s_1',s_2'}(A', [x]B') \tag{2}$$
$$\Gamma \vdash P : A \tag{3}$$
$$\Gamma \vdash P' : A' \tag{4}$$

Using this and $|P| = |P'|, |N| = |N'|$ we apply the IH to get a P'' satisfying $P \hookrightarrow_I^* P'' \hookleftarrow_I^* P'$ and $|P''| = |P| = |P'|$ and a N'' satisfying $N \hookrightarrow_I^* N'' \hookleftarrow_I^* N'$ and $|N''| = |N| = |N'|$. By subject reduction we then have $\Gamma \vdash N'' : \Pi_{s_1,s_2}(A, [x]B), \Pi_{s_1',s_2'}(A', [x]B')$. Thus, by Proposition 4.3 and confluence we have A'', B'' with $A \hookrightarrow^* A'' \hookleftarrow^* A'$ (implying $s_1 = s_1'$), and $B \hookrightarrow^* B'' \hookleftarrow^* B'$ (implying $s_2 = s_2'$). Now we can take $M'' = @_{s_1,s_2}(A'', [x]B'', N'', P'')$ and we have $M \hookrightarrow_I^* M'' \hookleftarrow_I^* M'$ and $|M''| = |M| = |M'|$. ■

Proposition 5.2 *If $|A| \hookrightarrow B$ then there is B' with $A \hookrightarrow B'$ and $|B'| = B$.*

Proof. By induction on the rewrite context, using compositionality of $|-|$ for the base case. ■

By iterating this proposition, we get the following corollary.

Corollary 5.1 *If* $|A| \hookrightarrow^* B$ *then there is* B' *with* $A \hookrightarrow^* B'$ *and* $|B'| = B$.

Finally, now we can show that the conversion between well-typed terms in a PTS is indeed reflected into its corresponding EPTS.

Proposition 5.3 (Reflection of conversion) *If* $\Gamma \vdash_{EPTS} M, N$ *typed with* $|M| \equiv |N|$ *then* $M \equiv N$.

Proof. By confluence we have P with $|M| \hookrightarrow^* P \hookleftarrow^* |N|$. By the previous result, there are M', N' st $M \hookrightarrow^* M'$ and $N \hookrightarrow^* N'$ with $|M'| = P = |N'|$. By subject reduction $\Gamma \vdash M', N'$ *typed*, and thus we can apply Proposition 5.1 to get a P' satisfying $M' \hookrightarrow_l^* P' \hookleftarrow_l^* N'$, and thus we deduce $M \equiv N$. ■

5.2.2 Completeness

We are now ready to show our main result: if $M : A$ holds in a PTS, then we can fill in M, A the missing parameters to get M', A' such that $M' : A'$ holds in the corresponding EPTS. The intuition behind the proof is that, by analyzing the judgment tree of $\Gamma \vdash_{PTS} M : A$ we can obtain candidates for the type parameters we will insert in M.

One technical difficulty is that, when applying the induction hypothesis to two judgments $\Gamma \vdash M : A$ and $\Gamma \vdash N : B$ with the same contexts, we get contexts Γ_1, Γ_2 where the inserted parameters need not be syntactically the same, but only convertible. Therefore, the following proposition will be of great help to switch between Γ_1 and Γ_2.

Proposition 5.4 (Context exchange) *If* $|\Gamma_1| \equiv |\Gamma_2|$, $\Gamma_1 \vdash_{EPTS} M : A$ *and* Γ_2 well-formed *then* $\Gamma_2 \vdash_{EPTS} M : A$.

Proof. We write Γ_1, Γ_2 as $\Gamma_1 = \Gamma, \Delta_1$ and $\Gamma_2 = \Gamma, \Delta_2$ and we prove the result on the size of Δ_1 (which is the same as the one of Δ_2).

For the base case this is trivial. For the induction step, we write $\Delta_i = x : B_i, \Delta_i'$ and thus our hypothesis becomes

$$\Gamma, x : B_1, \Delta_1' \vdash_{EPTS} M : A \tag{5}$$

From $|\Gamma_1| \equiv |\Gamma_2|$ we get $|B_1| \equiv |B_2|$ and because Γ_1 and Γ_2 are well-formed, we get $\Gamma \vdash_{EPTS} B_1 : s_1$ and $\Gamma \vdash_{EPTS} B_2 : s_2$. Hence, by Proposition 5.3 we deduce $B_1 \equiv B_2$. By conversion in context, we get

$$\Gamma, x : B_2, \Delta_1' \vdash_{EPTS} M : A \tag{6}$$

We can now apply the induction hypothesis to conclude. ■

We now proceed with the proof of completeness.

Theorem 5.2 (Completeness) *We have*

- *If* Γ well-formed *in PTS then there is* Γ' *st* Γ' well-formed *in EPTS with* $|\Gamma'| = \Gamma$.

- *If* $\Gamma \vdash_{PTS} M : A$ *then there are* Γ', M', A' *st* $\Gamma' \vdash_{EPTS} M' : A'$ *with* $|\Gamma'| = \Gamma$, $|M'| = M$ *and* $|A'| = A$.

Proof. By induction on the derivation. The case Empty is trivial.

Decl: The last rule of the derivation is

$$\frac{\Gamma \vdash_{PTS} A : s \qquad x \notin \Gamma}{\Gamma, x : A \text{ well-formed}} \text{ Decl}$$

By IH there are Γ', A', s' st $\Gamma' \vdash_{EPTS} A' : s'$ with $|X'| = X$ for $X = \Gamma, A, s$. For s, this implies $s' = s$. We can thus apply the rule Decl-ctx to get

$$\frac{\Gamma' \vdash_{EPTS} A' : s \qquad x \notin \Gamma}{\Gamma', x : A' \text{ well-formed}} \text{ Decl}$$

and we have indeed $|\Gamma', x : A'| = \Gamma, x : A$.

Var: The last rule of the derivation is

$$\frac{\Gamma \text{ well-formed} \qquad x : A \in \Gamma}{\Gamma \vdash_{PTS} x : A} \text{ Var}$$

By IH there is Γ' st $|\Gamma'| = \Gamma$ and Γ' well-formed in EPTS. Moreover, as $x : A \in \Gamma$, then $x : A' \in \Gamma'$ with $|A'| = A$. Therefore, we can apply the rule Var to obtain

$$\frac{\Gamma' \text{ well-formed} \qquad x : A' \in \Gamma'}{\Gamma' \vdash_{EPTS} x : A'} \text{ Var}$$

and we indeed have $|\Gamma' \vdash x : A'| = \Gamma \vdash x : A$.

Sort: Similar to previous case.

Conv: The last rule of the derivation is

$$\frac{\Gamma \vdash_{PTS} M : A \qquad \Gamma \vdash_{PTS} B : s \qquad A \equiv B}{\Gamma \vdash_{PTS} M : B} \text{ Conv}$$

By induction hypothesis we have $\Gamma', \Gamma'', M', A', B', s'$ st

$$\Gamma' \vdash_{EPTS} M' : A' \tag{1}$$
$$\Gamma'' \vdash_{EPTS} B' : s' \tag{2}$$

with $|X''| = |X'| = X$[2]. For s this gives $s' = s$. As $|\Gamma'| = |\Gamma''|$, we get

$$\Gamma' \vdash_{EPTS} B' : s \tag{3}$$

We also have that either $\Gamma' \vdash A' : s_A$ or A' is a top sort. However, if A' is a top sort then $A' = B'$ and this contradicts $\Gamma' \vdash B' : s$, thus $A' : s_A$. Therefore, from $|A'| = A \equiv B = |B'|$ we can apply Proposition 5.3 to deduce $A' \equiv B'$. We can now apply Conv to get

[2]We write X for any of the $\Gamma', \Gamma'', M', \ldots$

$$\frac{\Gamma' \vdash_{EPTS} M' : A' \qquad \Gamma' \vdash_{EPTS} B' : s \qquad A' \equiv B'}{\Gamma' \vdash_{EPTS} M' : B'} \text{Conv}$$

and we indeed have $|\Gamma' \vdash M' : B'| = \Gamma \vdash M : B$.

Prod: The last rule of the derivation is

$$\frac{\Gamma \vdash_{PTS} A : s_1 \qquad \Gamma, x : A \vdash_{PTS} B : s_2 \qquad (s_1, s_2, s_3) \in \mathcal{R}}{\Gamma \vdash_{PTS} \Pi x : A.B : s_3} \text{Prod}$$

By IH we get $\Gamma', \Gamma'', A', A'', s_1', B', s_2'$ st

$$\Gamma' \vdash_{EPTS} A' : s_1' \tag{1}$$
$$\Gamma'', x : A'' \vdash_{EPTS} B' : s_2' \tag{2}$$

with $|X''| = |X'| = X$. For s_i this gives $s_i' = s_i$. As $|\Gamma'| = |\Gamma''|$, we have

$$\Gamma'' \vdash_{EPTS} A' : s_1 \tag{3}$$

From (2) we also get $\Gamma'' \vdash A'' : s$ for some s, and thus by applying Proposition 5.1 to $|A'| = |A''|$ we get $A' \equiv A''$. We can then use Conv to deduce $\Gamma'' \vdash A'' : s_1$. Now we can use this to deduce

$$\frac{\Gamma'' \vdash_{EPTS} A'' : s_1 \qquad \Gamma'', x : A'' \vdash_{EPTS} B' : s_2 \qquad (s_1, s_2, s_3) \in \mathcal{R}}{\Gamma'' \vdash_{EPTS} \Pi_{s_1,s_2}(A'', [x]B') : s_3} \text{Prod}$$

and we indeed have $|\Gamma'' \vdash \Pi_{s_1,s_2}(A'', [x]B') : s_3| = \Gamma \vdash \Pi x : A.B : s_3$

Abs: The last rule of the derivation is

$$\frac{\Gamma \vdash_{PTS} \Pi x : A.B : s \qquad \Gamma, x : A \vdash_{PTS} N : B}{\Gamma \vdash_{PTS} \lambda x : A.N : \Pi x : A.B} \text{Abs}$$

By IH we have $\Gamma', \Gamma'', A', A'', N', B', B'', s'$ st

$$\Gamma' \vdash_{EPTS} \Pi_{s_1,s_2}(A', [x]B') : s' \tag{1}$$
$$\Gamma'', x : A'' \vdash_{EPTS} N' : B'' \tag{2}$$

with $|X''| = |X'| = X$. For s this gives $s = s'$. From $|\Gamma'| = |\Gamma''|$ we get

$$\Gamma'' \vdash_{EPTS} \Pi_{s_1,s_2}(A', [x]B') : s \tag{3}$$

By inversion we also deduce

$$(s_1, s_2, s) \in \mathcal{R} \tag{4}$$
$$\Gamma'' \vdash_{EPTS} A' : s_1 \tag{5}$$
$$\Gamma'', x : A' \vdash_{EPTS} B' : s_2 \tag{6}$$

From (2) we can get

$$\Gamma'' \vdash_{EPTS} A'' : s_{A''} \tag{7}$$
$$\Gamma'', x : A'' \vdash_{EPTS} B'' : s_{B''} \tag{8}$$

By applying Proposition 5.1 with $|A'| = |A''|$, we get $A' \equiv A''$. Then from conversion in context applied to $A' \equiv A''$, we get $\Gamma'', x : A'' \vdash B' : s_2$. Then we can use $|B| = |B'|$ and Proposition 5.1 to get $B \equiv B'$.

Using all this, we can apply conversion in context followed by Conv to (2), to get $\Gamma'', x : A' \vdash N' : B'$. Now we conclude with the following derivation.

$$\frac{\Gamma'' \vdash_{EPTS} A' : s_1 \qquad \Gamma'', x : A' \vdash_{EPTS} B' : s_2 \qquad (s_1, s_2, s) \in \mathcal{R} \qquad \Gamma'', x : A' \vdash_{EPTS} N' : B'}{\Gamma'' \vdash_{EPTS} \lambda_{s_1,s_2}(A', [x]B', [x]N') : \Pi_{s_1,s_2}(A', [x]B')} \text{Abs}$$

and we indeed have $|\Gamma'' \vdash \lambda_{s_1,s_2}(A', [x]B', [x]N') : \Pi_{s_1,s_2}(A', [x]B')| = \Gamma \vdash \lambda x : A.N : \Pi x : A.B$.

App: The last rule of the derivation is

$$\frac{\Gamma \vdash_{PTS} N_1 : \Pi x : A.B \qquad \Gamma \vdash_{PTS} N_2 : A}{\Gamma \vdash_{PTS} N_1 N_2 : B\{N_2/x\}} \text{Abs}$$

By the IH we have $\Gamma, \Gamma', A', A'', B', N_1', N_2'$ with

$$\Gamma' \vdash_{EPTS} N_2' : A' \tag{1}$$
$$\Gamma'' \vdash_{EPTS} N_1' : \Pi_{s_1,s_2}(A'', [x]B') \tag{2}$$

with $|X''| = |X'| = X$. By $|\Gamma'| = |\Gamma''|$ we get

$$\Gamma' \vdash_{EPTS} N_1' : \Pi_{s_1,s_2}(A'', [x]B') \tag{3}$$

We can then also get $\Gamma' \vdash_{EPTS} \Pi_{s_1,s_2}(A'', [x]B') : s_3$, to which we apply inversion and get

$$(s_1, s_2, s_3) \in \mathcal{R} \tag{4}$$
$$\Gamma' \vdash_{EPTS} A'' : s_1 \tag{5}$$
$$\Gamma', x : A'' \vdash_{EPTS} B' : s_2 \tag{6}$$

From $|A'| = |A''|$, we can apply Proposition 5.1 to get $A' \equiv A''$, from which we apply Conv with (5) and (1) to get $\Gamma' \vdash_{EPTS} N_2' : A''$. We can then finally conclude with

$$\frac{\Gamma' \vdash A'' : s_1 \qquad \Gamma', x : A'' \vdash B' : s_2 \qquad (s_1, s_2, s_3) \in \mathcal{R} \qquad \Gamma' \vdash N_1' : \Pi_{s_1,s_2}(A'', [x]B') \qquad \Gamma' \vdash N_2' : A''}{\Gamma' \vdash @_{s_1,s_2}(A'', [x]B', N_1', N_2') : B'\{N_2'/x\}} \text{App}$$

and we have indeed $|\Gamma' \vdash @_{s_1,s_2}(A'', [x]B', N_1', N_2') : B'\{N_2'/x\}| = \Gamma \vdash N_1 N_2 : B\{N_2/x\}$ ■

5.3 Syntactic correspondence

We now show a correspondence between terms in a PTS and in its corresponding EPTS. Given $\Gamma \vdash_{PTS} A$ *type*, we would like to take A', Γ' and establish a function mapping PTS terms of type A in Γ to EPTS terms of type A' in Γ', and then show it is a bijection up to some equivalence.

By Theorem 5.2 we know that for each $M \in \Lambda_{PTS}$ satisfying $\Gamma \vdash_{PTS} M : A$ there are Γ', M', A' st $\Gamma' \vdash_{EPTS} M' : A'$ and $|\Gamma'| = \Gamma$, $|A'| = A$ and $|M'| = M$. Let $\psi_{\Gamma,A}$ be a function choosing for any such M some M' satisfying this property.

Our first issue to solve is that Theorem 5.2 gives for each M a different Γ', A' with $\Gamma' \vdash_{EPTS} \psi_{\Gamma,A}(M) : A'$. The following result solves this problem.

Proposition 5.5 *Let $\Gamma \vdash_{PTS} A$ type. There exists $\Gamma', A' \in \Lambda_{EPTS}$ st $|\Gamma'| = \Gamma, |A'| = A$ and for all M with $\Gamma \vdash_{PTS} M : A$, $\Gamma' \vdash_{EPTS} \psi_{\Gamma,A}(M) : A'$.*

Proof. As $\Gamma \vdash_{PTS} A$ *type*, by definition there is some M with $\Gamma \vdash_{PTS} M : A$. By definition of ψ, there are Γ', A' with $|\Gamma'| = \Gamma, |A'| = A$ and $\Gamma' \vdash_{EPTS} \psi_{\Gamma,A}(M) : A'$.

Now let N be any term st $\Gamma \vdash_{PTS} N : A$. By the definition of ψ once again, there are Γ'', A'' with $|\Gamma''| = \Gamma, |A''| = A$ and $\Gamma'' \vdash_{EPTS} \psi_{\Gamma,A}(N) : A''$. As $|\Gamma'| = |\Gamma''|$, by Proposition 5.4 we have $\Gamma' \vdash_{EPTS} \psi_{\Gamma,A}(N) : A''$. As $|A'| = |A''|$, by Proposition 5.1 we have $A' \equiv A''$, thus by Conv we have $\Gamma' \vdash_{EPTS} \psi_{\Gamma,A}(N) : A'$. ■

It is now left to show that $\psi_{\Gamma,A}$ forms a bijection with $|-|$ up to some equivalence. Note that by its very definition we have $|\psi_{\Gamma,A}(M)| = M$. We can also show the complementary direction using our previously defined notion of $\equiv_I$.

Proposition 5.6 *Let $\Gamma \vdash_{EPTS} M : A$. We have $\psi_{|\Gamma|,|A|}(|M|) \equiv_I M$.*

Proof. If $\Gamma \vdash_{EPTS} M : A$, then by Theorem 5.1 we have $|\Gamma| \vdash_{PTS} |M| : |A|$, and thus $\psi_{|\Gamma|,|A|}$ is well-defined in $|M|$. Then, by definition of ψ, we have $\Gamma' \vdash_{EPTS} \psi_{|\Gamma|,|A|}(|M|) : A'$, where $|\Gamma'| = |\Gamma|$, $|\psi_{|\Gamma|,|A|}(|M|)| = |M|$ and $|A'| = |A|$. Finally, $|\psi_{|\Gamma|,|A|}(|M|)| = |M|$ implies $\psi_{|\Gamma|,|A|}(|M|) \equiv_I M$ by Proposition 5.1. ■

With all these results, we can now show our bijection.

Theorem 5.3 *Let $\Gamma \vdash_{PTS} A$ type. There are Γ', A' with $|\Gamma'| = \Gamma, |A'| = A$ such that we have a bijection*

$$\Lambda(\Gamma \vdash_{PTS} \_ : A) \simeq \Lambda(\Gamma' \vdash_{EPTS} \_ : A')/\equiv_I$$

given by $\psi_{\Gamma,A}$ and $|-|$.

Proof. Given $\Gamma \vdash_{PTS} A$ *type*, we apply Proposition 5.5 to get Γ', A' satisfying $|\Gamma'| = \Gamma, |A'| = A$. By Proposition 5.5 also, $\psi_{\Gamma,A}$ is a function $\Lambda(\Gamma \vdash_{PTS} \_ : A) \to \Lambda(\Gamma' \vdash_{EPTS} \_ : A')$, and thus in particular a function $\Lambda(\Gamma \vdash_{PTS} \_ : A) \to \Lambda(\Gamma' \vdash_{EPTS} \_ : A')/\equiv_I$.

By Theorem 5.1, $|-|$ is a function $\Lambda(\Gamma' \vdash_{EPTS} \_ : A') \to \Lambda(\Gamma \vdash_{PTS} \_ : A)$. Because for $M \equiv_I N$ we can show $|M| = |N|$, this restricts to a function $\Lambda(\Gamma' \vdash_{EPTS} \_ : A')/\equiv_I \to \Lambda(\Gamma \vdash_{PTS} \_ : A)$.

Moreover, we have $|\psi_{\Gamma,A}(M)| = M$ by definition and $\psi_{\Gamma,A}(|M|) = \psi_{|\Gamma'|,|A'|}(|M|) \equiv_I M$ by Proposition 5.6, establishing the bijection. ■

References

[1] H. P. Barendregt. *Lambda Calculi with Types*, page 117–309. Oxford University Press, Inc., USA, 1993.

[2] G. Dowek and B. Werner. On the definition of the eta-long normal form in type systems of the cube. In *Informal Proceedings of the Workshop on Types for Proofs and Programs*, 1993.

[3] T. Felicissimo. Adequate and computational encodings in the logical framework Dedukti. In preparaton, 2022.

[4] J. W. Klop, V. van Oostrom, and F. van Raamsdonk. Combinatory reduction systems: introduction and survey. *Theoretical Computer Science*, 121(1):279–308, 1993.

[5] P.-A. Melliès and B. Werner. A generic normalisation proof for pure type systems. In E. Giménez and C. Paulin-Mohring, editors, *Types for Proofs and Programs*, pages 254–276, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[6] V. Siles and H. Herbelin. Pure type system conversion is always typable. *Journal of Functional Programming*, 22:153 – 180, 2012.