Pure Type System conversion is always typable

Vincent Siles, Hugo Herbelin

Under consideration for publication in J. Functional Programming

Pure Type System conversion is always typable

Vincent Siles
Ecole Polytechnique / INRIA / Laboratoire PPS, Equipe πr^2
and Hugo Herbelin
INRIA / Laboratoire PPS, Equipe πr^2
(*e-mail:* vincent.siles@polytechnique.edu, hugo.herbelin@inria.fr)

Abstract

Pure Type Systems are usually described in two different ways, one that uses an external notion of computation like beta-reduction, and one that relies on a typed judgment of equality, directly in the typing system.

For a long time, the question was open to know whether both presentations described the same theory. A first step toward this equivalence has been made by Adams for a particular class of *Pure Type Systems* (PTS) called functional. Then, his result has been relaxed to all semi-full PTS in previous work. In this paper, we finally give a positive answer to the general issue, and prove that equivalence holds for any Pure Type System.

1 Introduction

Dependent type systems are used as a basis for both formalizing mathematics and building more expressive programming languages. Some popular implementations of those concepts are the proof systems *Coq*[1] - which is built on top of the *Calculus of Inductive Constructors* (Werner, 1994) - *Isabelle-HOL*[2] - which can be seen as an extension of Girard's system F_ω - and the dependently typed programming language *Agda 2* (Norell, 2007). A key ingredient of these systems is the presence of an internal notion of equality based on β-conversion or $\beta\eta$-conversion. However, two traditional presentations of this equality can be found in the literature. One way to express it is to rely on an untyped, external, conversion through a rule of the form:

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash B\ type}{\Gamma \vdash M : B}\ A =_\beta B$$

This rule is the one conventionally used to define e.g. the *Calculus of Inductive Constructions*. The equality is used as a toolbox that knows nothing about the typing validity of the terms it deals with: each conversion step is not checked to be well-typed, but along with *Confluence* and *Subject Reduction*, we can ensure that everything goes fine. A second approach embeds a notion of equality directly in the type system. So there are two kinds of

[1] http://coq.inria.fr/refman/
[2] http://www.cl.cam.ac.uk/research/hvg/Isabelle/

typing judgments: one to type terms, and one to type equalities. With this kind of approach, we can ensure that every conversion step is well-typed:

$$\frac{\Gamma \vdash_e M : A \qquad \Gamma \vdash_e A =_\beta B\ type}{\Gamma \vdash_e M : B}$$

Those systems are known as "type systems with judgmental equality". The equality knows some typing information, and needs to fulfill some typing constraints to hold, it is not an external tool anymore. This is the case of *Martin-Löf's Type Theory* (Martin-Löf, 1984; Nordstrom *et al.*, 1990) or *UTT* (Goguen, 1994) from which *Agda 2* is derived.

One way or the other, these presentations seem to express the same judgments, in two different ways, but surprisingly, showing the equivalence between those two definitions is difficult. Geuvers and Werner (1994) early noticed that being able to lift an untyped equality to a typed one, i.e. to turn a system with β-conversion into a system with judgmental equality requires to show *Subject Reduction* in the latter system:

$$\text{If } \Gamma \vdash_e M : A \text{ and } M \twoheadrightarrow_\beta N \text{ then } \Gamma \vdash_e M =_\beta N : A.$$

Subject Reduction requires the injectivity of dependent products $\Pi x^A.B$:

$$\text{If } \Gamma \vdash_e \Pi x^A.B =_\beta \Pi x^C.D\ type \text{ then } \Gamma \vdash_e A =_\beta C\ type \text{ and } \Gamma(x:A) \vdash_e B =_\beta D\ type.$$

This property itself relies on a notion of typed confluence which again involves *Subject Reduction*: we are facing a circular dependency.

Both presentations have their own purpose, but in two different directions. Usually the systems based on judgmental equality are better suited for theoretical considerations, like building models (Goguen, 1994; Abel *et al.*, 2007; Werner & Lee, 2010) or studying semantics and *Normalization by Evaluation* (Abel, 2010). On the other hand, the typing judgments are irrelevant for computation and with untyped conversion, one can concentrate on the purely computational content of conversion. Those systems are also better suited for type-checking and type-inference as developed in van Benthem Jutting (1993) with the definition of a *syntax directed* version of *Pure Type Systems*. However, there is still a missing link between both presentations to ensure that they are effectively describing the same theory.

Besides looking for a better understanding of the relations between typed and untyped equality, another motivation is to apply such an equivalence to the foundations of proof assistants. For instance, for *Coq*, the construction of a set-theoretical model (on which relies the consistency of some standard mathematical axioms) requires the use of a typed equality. However, the implementation relies on an untyped version of the same system. By achieving the equivalence between both presentations, we would be able to assert that a set-theoretical model, such as the one given by Werner and Lee, correctly applies to the actual implementation.

The first proofs of equivalence only concerned particular cases without aiming for a general statement, and were based on construction of models, one system at a time (Geuvers, 1993; Goguen, 1994; Abel *et al.*, 2007). However, this kind of approach does not scale

easily since it relies on the underlying model construction, which is closely linked to the structure of each particular system.

All the previous examples rely on several different concepts, like type dependency, but also subtyping or inductive types. *Pure Type Systems* are a framework which is at the core of the world of dependent types, with the (dependent) implication as only type constructor: most complex systems are built on top of a particular PTS by adding new kinds of type constructors (inductive types, intersection types, ...). As we will see in the following sections, the typed vs untyped equality issue also shows up in PTS. A few years ago, Adams (2006) found a *syntactical* criterion and proved that every *functional* Pure Type System is equivalent to its counterpart with judgmental equality. The authors also made a new step toward an extension of the result to all PTS by reusing Adams' criterion to prove that the equivalence also holds for any *semi-full* Pure Type System (Siles & Herbelin, 2010). The main idea of those proofs is to define an intermediate system called *Typed Parallel One Step Reduction* (or TPOSR) that embeds the idea of a typed equality, along with the idea of parallel reduction which is at the heart of the proof of *Confluence*.

In this paper, we shall prove that the equivalence holds for *any PTS*: every instance of Pure Type System, even a non-normalizing one, is equivalent to its judgmental equality counterpart. To do so, we extended Adams' TPOSR definition into a new system which enjoys the same properties about typing and reduction, while keeping the whole generality of PTS: *Pure Type System based on Annotated Typed Reduction* (PTS_{atr}).

The whole process that we are going to describe involves some quite complicated structures and large mutual inductive proofs, so everything stated in this paper has been formalized (using de Bruijn indices (1972)) in the proof assistant *Coq*. The whole development can be found at

`http://www.lix.polytechnique.fr/~vsiles/coq/PTSATR.html`.

By closing this open problem, we are one step closer to more complex typing systems, for example systems with subtyping like the *Extended Calculus Of Constructions* (Luo, 1989) and the *Calculus of Inductive Constructions*, or systems with more expressive conversion that consider η-expansion (as in Geuvers & Werner, 1994).

2 The meta-theory of PTS

In this section, we give the definitions of *Pure Type System* (PTS) and *Pure Type System with Judgmental Equality* (PTS_e), its "typed" counterpart. We also recall the main properties of these systems, and the main issues that one will face while trying to prove that both presentations are equivalent.

2.1 Terms and Untyped Reductions

The terms used in the following type systems are the usual λ-calculus terms *a la* Church - variable, abstraction and application - extended with two more constructions which are the entry points of types inside terms : Π-types and sorts.

Structure of terms and contexts

s : *Sorts*

x : *Vars*

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

$\Gamma ::= \emptyset \mid \Gamma(x:A)$

The Π construct will be used to type functions, and is usually noted $A \to B$ when B does not depend on its argument. If there is a dependency, we keep track of the binding variable x with this notation.

The set *Sorts* is the first parameter that defines an instance of PTS. Sorts are used to assert that a term can correctly be used in a typing position. We will see how it works in more detail after the introduction of the typing rules. The set of variables *Vars* is assumed to be infinite, and is common to all PTS. In the following, we consider s, s_i and t to be in *Sorts*, and x, y and z to be in *Vars*. A context is a list of terms labeled by distinct variables, e.g. $\Gamma \equiv (x_1 : A_1) \ldots (x_n : A_n)$, where all the x_i are distinct. Since we want to handle dependent types, the order inside the context matters: a x_i can only appear in A_j where $j > i$. $\Gamma(x) = A$ is a shortcut for $(x : A) \in \Gamma$ and $\emptyset$ denotes the empty context. The *domain* $Dom(\Gamma)$ of a context Γ is defined as the set of x_i such that $\Gamma(x_i)$ exists. The concatenation of two contexts whose domains are disjoint is written $\Gamma_1\Gamma_2$.

The term $\lambda x^A.M$ (resp. $\Pi x^A.B$) binds the variable x in M (resp. B) but not in A and the set of *free variables (fv)* is defined as usual according to those binding rules.

We use an external notion of substitution: $\_[\_/\_]$ is the function of substitution, and $M[N/x]$ stands for the term M where all the free variables x have been replaced by N, without any variable capture. We can extend the substitution to contexts (in this case, we consider that $x \notin Dom(\Gamma)$). $\Gamma[N/x]$ is recursively defined as :

1. $\emptyset[N/x] \triangleq \emptyset$
2. $(\Gamma(y:A))[N/x] \triangleq \Gamma[N/x](y : A[N/x])$

The notion of β-reduction ($\to_\beta$) is defined as the congruence closure of the relation $(\lambda x^A.M)N \to_\beta M[N/x]$ over the grammar of terms. The reflexive-transitive closure of $\to_\beta$ is written as $\twoheadrightarrow_\beta$, and its reflexive-symmetric-transitive closure as $=_\beta$. The notion of syntactic equality (up to α-conversion) is denoted as $\equiv$.

At this point, it is important to notice the order in which we can prove things: *Confluence* of the β-reduction can be established before even defining the typing system, it is only a property of the reduction. Using this, we can prove some useful properties of Π-types and sorts:

Lemma 2.1 (*Confluence and its consequences*)

- If $M \twoheadrightarrow_\beta N$ and $M \twoheadrightarrow_\beta P$ then there is Q such that $N \twoheadrightarrow_\beta Q$ and $P \twoheadrightarrow_\beta Q$.
- Π-*injectivity*: If $\Pi x^A.B =_\beta \Pi x^C.D$ then $A =_\beta C$ and $B =_\beta D$
- If $s =_\beta t$ then $s \equiv t$.

2.2 Presentation of Pure Type Systems

2.2.1 Pure Type System

A PTS is a generic framework first presented by Berardi (1988) and Terlouw to study a family of type systems all at once. Popular type systems like *Simply Typed Lambda Calculus*, *System F* or *Calculus of Constructions (CoC)* are part of this family. There is plenty of literature on the subject (Barendregt *et al.*, 1992) so we only recall the main ideas of those systems.

The abstract nature of PTS arise in the typing rules for sorts and Π-types. The set $Ax \subset (Sorts \times Sorts)$ is used to type sorts: $(s,t) \in Ax$ means that the sort s can be typed by the sort t. The set $Rel \subset (Sorts \times Sorts \times Sorts)$ is used to check the well-formedness of Π-types.
The typing rules for PTS are given in Fig. 1. Intuitively, $\Gamma \vdash M : T$ can be read as "the term M has type T in the context Γ", and $\Gamma \vdash A : s$ as "A is a valid type in Γ".

As we can see, the CONV rule relies on the external notion of β-conversion, so we do not check that every step of the conversion is well-typed. However, it is easy to prove *Confluence* and *Subject Reduction*, two properties which ensure that everything goes well.

In this paper, we will later refer to some subclasses of PTS:

Functional, Full and semi-Full PTS

- A PTS is functional if:
  1. for all s,t,t', $(s,t) \in \mathscr{A}x$ and $(s,t') \in \mathscr{A}x$ forces $t \equiv t'$.
  2. for all s,t,u,u', $(s,t,u) \in \mathscr{R}el$ and $(s,t,u') \in \mathscr{R}el$ forces $u \equiv u'$.
- A PTS is semi-full if $(s,t,u) \in Rel$ enforces that for all t', there is u' such that $(s,t',u') \in Rel$.
- A PTS is full if for any s,t, there is u such that $(s,t,u) \in Rel$.
  Obviously, a full PTS is also semi-full.

Lemma 2.2 (*Type Uniqueness for functional PTS*)
In any *functional* PTS, if $\Gamma \vdash M : T$ and $\Gamma \vdash M : T'$ then $T =_\beta T'$.

The following properties hold for all PTS. Even if they are quite technical, they are the basic meta-theory that we need to prove the interesting theorems.

Lemma 2.3 (*Weakening*)
1. If $\Gamma_1\Gamma_2 \vdash M : B$, $\Gamma_1 \vdash A : s$ and $x \notin Dom(\Gamma_1\Gamma_2)$ then $\Gamma_1(x:A)\Gamma_2 \vdash M : B$.
2. If $\Gamma_1\Gamma_2\ _{wf}$, $\Gamma_1 \vdash A : s$ and $x \notin Dom(\Gamma_1\Gamma_2)$ then $\Gamma_1(x:A)\Gamma_2\ _{wf}$.

Lemma 2.4 (*Substitution*)
1. If $\Gamma_1(x:A)\Gamma_2 \vdash M : B$ and $\Gamma_1 \vdash P : A$ then $\Gamma_1\Gamma_2[P/x] \vdash M[P/x] : B[P/x]$.
2. If $\Gamma_1(x:A)\Gamma_2\ _{wf}$ and $\Gamma_1 \vdash P : A$ then $\Gamma_1\Gamma_2[P/x]_{wf}$.

While proving facts about PTS, we will often need to compute some typing information about the subterms of one judgment. To do this, we will frequently use the *Generation* (or *Inversion*) property:

Theorem 2.5 (*Generation*)

$$\frac{}{\emptyset_{wf}}\ \text{NIL} \qquad \frac{\Gamma \vdash A : s \qquad x \notin Dom(\Gamma)}{\Gamma(x:A)_{wf}}\ \text{CONS}$$

$$\frac{\Gamma_{wf} \qquad (s,t) \in \mathscr{A}x}{\Gamma \vdash s : t}\ \text{SORT} \qquad \frac{\Gamma_{wf} \qquad \Gamma(x) = A}{\Gamma \vdash x : A}\ \text{VAR}$$

$$\frac{\Gamma \vdash A : s \qquad \Gamma(x:A) \vdash B : t \qquad (s,t,u) \in \mathscr{R}el \qquad \Gamma(x:A) \vdash M : B}{\Gamma \vdash \lambda x^A.M : \Pi x^A.B}\ \text{LAM} \qquad \frac{\Gamma \vdash A : s \qquad \Gamma(x:A) \vdash B : t \qquad (s,t,u) \in \mathscr{R}el}{\Gamma \vdash \Pi x^A.B : u}\ \text{PI}$$

$$\frac{\Gamma \vdash M : \Pi x^A.B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B[N/x]}\ \text{APP} \qquad \frac{\Gamma \vdash M : A \qquad A =_\beta B \qquad \Gamma \vdash B : s}{\Gamma \vdash M : B}\ \text{CONV}$$

Fig. 1. Typing Rules for PTS

1. If $\Gamma \vdash s : T$ then there is t such that $(s,t) \in \mathscr{A}x$ and $T =_\beta t$.
2. If $\Gamma \vdash x : A$ then there is B such that $\Gamma(x) = B$ and $A =_\beta B$.
3. If $\Gamma \vdash \Pi x^A.B : T$ then there are s_1, s_2, s_3 such that $\Gamma \vdash A : s_1$, $\Gamma(x : A) \vdash B : s_2$, $(s_1, s_2, s_3) \in \mathscr{R}el$ and $T =_\beta s_3$.
4. If $\Gamma \vdash \lambda x^A.M : T$ then there are s_1, s_2, s_3 and B such that $\Gamma \vdash A : s_1$, $\Gamma(x : A) \vdash B : s_2$, $\Gamma(x : A) \vdash M : B$, $(s_1, s_2, s_3) \in \mathscr{R}el$ and $T =_\beta \Pi x^A.B$.
5. If $\Gamma \vdash M\ N : T$ then there are A and B such that $\Gamma \vdash M : \Pi x^A.B$, $\Gamma \vdash N : A$ and $T =_\beta B[N/x]$.

Lemma 2.6 (*Type Correctness*)
If $\Gamma \vdash M : T$, then there is s such that $T \equiv s$ or $\Gamma \vdash T : s$.

Since we want the full generality of PTS, we need to distinguish between the two conclusions: nothing ensures that all sorts are well-typed. Such sorts do not appear in left position of any pair $(s,t) \in Ax$, and they are called *top sorts*.

The notion of β-conversion can easily extended to context since they are ordered lists of terms:

Context Conversion

- $\emptyset =_\beta \emptyset$.
- If $\Gamma =_\beta \Gamma'$, $A =_\beta B$ and $x \notin Dom(\Gamma)$, then $\Gamma(x : A) =_\beta \Gamma'(x : B)$.

Lemma 2.7 (*Context Conversion in Judgments*)

If $\Gamma \vdash M : A$, $\Gamma =_\beta \Gamma'$ and Γ'_{wf} then $\Gamma' \vdash M : A$.

With all those tools, we can now prove the main property of PTS, which states that computation preserves typing:

Theorem 2.8 (*Subject Reduction*)
If $\Gamma \vdash M : A$ and $M \to_\beta N$, then $\Gamma \vdash N : A$.

Proof
The proof can be found in (Barendregt *et al.*, 1992). We just want to put forward that it relies on *Confluence*, more precisely on the Π-*injectivity* of β-reduction. □

Now that we have *Subject Reduction*, we can prove that any use of the CONV rule is sound, even if the conversion path uses ill-typed terms. If this is the case, we can find another path only made of well-typed terms.

Corollary 2.9 (*Using* CONV *is always sound*)
Any use of CONV can be broken into single reduction and expansion steps between well-typed terms only.

Proof
Let us suppose we have $\Gamma \vdash M : T$, $\Gamma \vdash T' : s$ and $T =_\beta T'$. By *Confluence*, there is T_0 such that $T \twoheadrightarrow_\beta T_0 \ {}_\beta\!\twoheadleftarrow T'$. By *Type Correctness*, there is t such that $\Gamma \vdash T : t$, or $T \equiv t$:

1. In the first case, by *Subject Reduction*, we know that any term that appears in the reduction from T to T_0 is typed by t, and any term that appears in the reduction from T' to T_0 is typed by s. So we have a path from T to T' exactly made of well-typed terms.
2. In the second case, $T' =_\beta t$ and by *Confluence*, $T' \twoheadrightarrow_\beta t$. *Subject Reduction* enforces $\Gamma \vdash t : s$. So this time also, the path from $T (\equiv t)$ and T' is exactly made of well-typed terms.

□

It is here interesting to see that in the first case, the path between T and T' is well-typed by sorts, but nothing guarantees that we can have the same sort in both branches. If we wanted to do so, we would need to be in a functional PTS.

2.2.2 Pure Type System with Judgmental Equality

There is another variant of the presentation of Pure Type System, by defining an internal notion of equality: Pure Type System with Judgmental Equality, where every conversion step is checked to be well-typed. With those judgments, we no longer need to rely on *Confluence* and *Subject Reduction* to ensure that the conversion sequences all involve well-typed terms. The typing rules for PTS_e are given in Fig. 2.

We can prove that some properties of PTS also hold for PTS_e, namely *Weakening, Substitution* and *Context Conversion*. We can add to the list the following reflexivity properties (also known as *Equation Validity*) which need to be proved along with *Type Correctness*:

Lemma 2.10 (*Type Correctness and, Left-Hand / Right-Hand reflexivity of* PTS_e)
- If $\Gamma \vdash_e M : T$ or $\Gamma \vdash_e M = N : T$, then there is $s \in \mathcal{S}orts$ such that $T \equiv s$ or $\Gamma \vdash_e T : s$.
- If $\Gamma \vdash_e M =_\beta N : A$, then $\Gamma \vdash_e M : A$.

$$\frac{}{\emptyset_{wf}}\ \text{NIL} \qquad \frac{\Gamma \vdash_e A : s \qquad x \notin Dom(\Gamma)}{\Gamma(x:A)_{wf}}\ \text{CONS}$$

$$\frac{\Gamma_{wf} \qquad (s,t) \in \mathscr{A}x}{\Gamma \vdash_e s : t}\ \text{SORT} \qquad \frac{(s_1,s_2,s_3) \in \mathscr{R}el \qquad \Gamma \vdash_e A : s_1 \qquad \Gamma(x:A) \vdash_e B : s_2}{\Gamma \vdash_e \Pi x^A.B : s_3}\ \text{PI}$$

$$\frac{\Gamma_{wf} \qquad (s,t) \in \mathscr{A}x}{\Gamma \vdash_e s =_\beta s : t}\ \text{SORT-EQ} \qquad \frac{(s_1,s_2,s_3) \in \mathscr{R}el \qquad \Gamma \vdash_e A =_\beta A' : s_1 \qquad \Gamma(x:A) \vdash_e B =_\beta B' : s_2}{\Gamma \vdash_e \Pi x^A.B =_\beta \Pi x^{A'}.B' : s_3}\ \text{PI-EQ}$$

$$\frac{\Gamma_{wf} \qquad \Gamma(x) = A}{\Gamma \vdash_e x : A}\ \text{VAR} \qquad \frac{\Gamma \vdash_e A : s_1 \qquad \Gamma(x:A) \vdash_e B : s_2 \qquad (s_1,s_2,s_3) \in \mathscr{R}el \qquad \Gamma(x:A) \vdash_e M : B}{\Gamma \vdash_e \lambda x^A.M : \Pi x^A.B}\ \text{LAM}$$

$$\frac{\Gamma_{wf} \qquad \Gamma(x) = A}{\Gamma \vdash_e x =_\beta x : A}\ \text{VAR-EQ} \qquad \frac{\Gamma \vdash_e A =_\beta A' : s_1 \qquad \Gamma(x:A) \vdash_e B : s_2 \qquad (s_1,s_2,s_3) \in \mathscr{R}el \qquad \Gamma(x:A) \vdash_e M =_\beta M' : B}{\Gamma \vdash_e \lambda x^A.M =_\beta \lambda x^{A'}.M' : \Pi x^A.B}\ \text{LAM-EQ}$$

$$\frac{\Gamma \vdash_e M : A \qquad \Gamma \vdash_e A =_\beta B : s}{\Gamma \vdash_e M : B}\ \text{CONV} \qquad \frac{\Gamma \vdash_e M : \Pi x^A.B \qquad \Gamma \vdash_e N : A}{\Gamma \vdash_e MN : B[N/x]}\ \text{APP}$$

$$\frac{\Gamma \vdash_e M =_\beta N : A \qquad \Gamma \vdash_e A =_\beta B : s}{\Gamma \vdash_e M =_\beta N : B}\ \text{CONV-EQ} \qquad \frac{\Gamma \vdash_e M =_\beta M' : \Pi x^A.B \qquad \Gamma \vdash_e N =_\beta N' : A}{\Gamma \vdash_e MN =_\beta M'N' : B[N/x]}\ \text{APP-EQ}$$

$$\frac{\Gamma \vdash_e M : A}{\Gamma \vdash_e M =_\beta M : A}\ \text{REFL} \qquad \frac{(s_1,s_2,s_3) \in \mathscr{R}el \qquad \Gamma \vdash_e A : s_1 \qquad \Gamma(x:A) \vdash_e B : s_2 \qquad \Gamma \vdash_e N : A \qquad \Gamma(x:A) \vdash_e M : B}{\Gamma \vdash_e (\lambda x^A.M)N =_\beta M[N/x] : B[N/x]}\ \text{BETA}$$

$$\frac{\Gamma \vdash_e N =_\beta M : A}{\Gamma \vdash_e M =_\beta N : A}\ \text{SYM} \qquad \frac{\Gamma \vdash_e M =_\beta N : A \qquad \Gamma \vdash_e N =_\beta P : A}{\Gamma \vdash_e M =_\beta P : A}\ \text{TRANS}$$

Fig. 2. Typing Rules for PTS_e

- If $\Gamma \vdash_e M =_\beta N : A$, then $\Gamma \vdash_e N : A$.

Proof

We need to prove all these propositions at once for three main reasons:

1. to prove *Type Correctness*, we need the *Right-Hand reflexivity* for the CONV rule.
2. to prove both reflexivity statement, we need *Type Correctness* for the APP-EQ rule.

3. because of the SYM rule, we need to prove both reflexivity statement at once.

Then, *Left-Hand reflexivity* is simply done by induction: all the premises of the typing rules of PTS_e have been chosen to correctly type the left hand-side of the equality in the current context. However, the *Right-Hand reflexivity* needs a little more work: the proof rely on the *Substitution Lemma* (to type the right part of BETA), *Left Reflexivity* and *Context Conversion*.

It is interesting to notice that we could have removed the dependency on *Type Correctness* just by adding more typing information (like the fact that A and B are also well-typed, with the correct sorts) to the premises of APP-EQ. □

With these few results, we can prove half of the equivalence we are looking for:

Theorem 2.11 (*From PTS_e to PTS*)
1. If $\Gamma \vdash_e M : A$ then $\Gamma \vdash M : A$.
2. If $\Gamma \vdash_e M =_\beta N : A$ then $\Gamma \vdash M : A$, $\Gamma \vdash N : A$ and $M =_\beta N$.

Proof
The proof is a simple induction and relies on properties of PTS: we just "forget" some typing information when dealing with the typed equalities. □

2.3 Subject Reduction and Equivalence

We previously saw that *Subject Reduction* and Π-*injectivity* were two important properties of PTS: *Subject Reduction* allows us to freely compute without having to check that typing is preserved at every reduction step, and Π-*injectivity* is a crucial step to prove the latter. With the basic meta-theory for PTS_e at hand, we can now try to check if both properties also holds when the equality is checked to be well-typed. If it is the case, we would be able to prove that both presentation are in fact two different way to describe the same theory.

Theorem 2.12 (*Subject Reduction*)
If $\Gamma \vdash_e M : T$ and $M \to_\beta N$ then $\Gamma \vdash_e M =_\beta N : T$.

To prove this property for PTS_e, we can try the same approach that was used for PTS, but this requires to have the Π-*injectivity* for PTS_e. Since we are using a typed equality, we can express this injectivity in several ways, for example by completely getting rid of the types (as we did for PTS), or instead by trying to keep as much typing information as we can.

With the first solution, we lack too much type information to build the typed equality needed by *Subject Reduction*. For the second one, we need to find the correct statement for the injectivity. After proving the equivalence between *functional* PTS and PTS_e, Adams did manage to prove a strong version of injectivity, but was unsuccessful at doing it in the general case. In fact, this statement is wrong in the general case :

Lemma 2.13 (*Strong* Π-*injectivity does not hold for all PTS_e*)
The following statement does not hold for all PTS_e:

If $\Gamma \vdash_e \Pi x^A.B =_\beta \Pi x^C.D : u$, then $\Gamma \vdash_e A =_\beta C : s$, $\Gamma(x : A) \vdash_e B =_\beta D : t$ for some $s,t \in \mathcal{S}orts$ such that $(s,t,u) \in \mathcal{R}el$.

Proof
We are going to build a counterexample by selecting the right sets for *Sorts*, *Ax* and *Rel*. Let us assume that strong injectivity (1) holds for all PTS_e, including the following one:

- $Sorts \equiv \{u, v, v', w, w'\}$
- $Ax \equiv \{(u, v), (u, v'), (v, w), (v', w')\}$
- $Rel \equiv \{(w, w, w), (w', w', w'), (v, v, u), (v', v', u)\}$

Let us define two terms $D1 \equiv (\lambda x^{v}.u)\ u$ and $D2 \equiv (\lambda x^{v'}.u)\ u$.

1. $\emptyset \vdash_e D1 : v$ and $\emptyset \vdash_e D1 : T$ forces $T =_\beta v$.
2. $\emptyset \vdash_e D2 : v'$ and $\emptyset \vdash_e D2 : T$ forces $T =_\beta v'$.
3. with both results and the fact that $\emptyset \vdash_e u : v$ and $\emptyset \vdash_e u : v'$, we can prove $\emptyset \vdash_e D1 =_\beta u : v$ and $\emptyset \vdash_e D2 =_\beta u : v'$.
4. The correct choice of rules in *Rel* leads to $\emptyset \vdash_e \Pi x^{D1}.u =_\beta \Pi x^{u}.u : u$ and $\emptyset \vdash_e \Pi x^{u}.u : u =_\beta \Pi x^{D2}.u : u$, so by transitivity: $\emptyset \vdash_e \Pi x^{D1}.u =_\beta \Pi x^{D2}.u : u$.
5. Since we supposed (1), either $\emptyset \vdash_e D1 =_\beta D2 : v$ or $\emptyset \vdash_e D1 =_\beta D2 : v'$.
6. In both case, one of the reflexivity lemmas and the first two items force $v =_\beta v'$ which is impossible by *Confluence* (cf Lemma 2.1).

□

To directly prove *Subject Reduction*, we need to find the correct injectivity statement that will give enough typing information to build the equality, but not too much so that it is still provable in all cases. In the next sections, we will see a statement that enjoys both properties, but we are not able to prove it directly from the lemmas we have right now, so we will come back to it later.

Renouncing to prove the Π-injectivity we need directly from within PTS_e, one may want to translate PTS_e judgments in PTS ones to use their properties, but again one is stuck: even if the translation from PTS_e to PTS is almost trivial, the translation back from PTS to PTS_e relies itself on *Subject Reduction in* PTS_e.

At this stage, we do not have enough available material to prove *Subject Reduction* for PTS_e. We will come back to this proof after achieving the equivalence between PTS and PTS_e. In the next section, we explain our new approach to prove the general equivalence, mostly influenced by Adam's TPOSR system. However, even if our new system is very similar to TPOSR, the ways to build its meta-theory have major differences.

3 Basic meta-theory of PTS_{atr}

3.1 Definition of PTS_{atr}

Let us go back to the question of lifting a typing judgment from PTS to PTS_e. To do so, we need to be able to lift a conversion $A =_\beta B$ into a typed equality judgment $\Gamma \vdash_e A =_\beta B$ and as said above, we would like to have *Subject Reduction* for PTS_e which itself requires the injectivity of Π-types.

A first proof of equivalence between PTS and PTS_e has been made by Adams (2006) for the subclass of *functional* PTS, a result that has been later extended to the subclasses of

semi-full and *full* PTS by the authors (Siles & Herbelin, 2010). As expected, the key step of these proofs is to build an intermediate system with two major properties:

1. It has to be equivalent to both PTS and PTS_e.
2. It has to verify the *Church-Rosser* property.

With such a system, we can prove that it enjoys Π-*injectivity* and *Subject Reduction*, and finally translate both properties into PTS_e.

This injectivity is a direct conclusion of the *Church-Rosser* property. But since we are dealing with a typed equality, we need to build a typed version of this property. The usual way to prove it for β-reduction is to define a parallel reduction that enjoys the *Diamond Property*, and whose transitive-closure is the same closure as β-reduction. So Adams defined a typed version of this parallel reduction called *Type Parallel One Step Reduction* to prove his result. However, the proof of the *Church-Rosser* property for TPOSR is not so trivial to do: as we will see in more details later, additional typing information are required to conclude the proof. Adams decided to annotate applications by their co-domain, and to restrict to functional PTS so his system would also enjoy the *Uniqueness of Types*. We used the same annotation system to show that the *Church-Rosser* property also holds for semi-full and full systems. However, to be able to prove the *Church-Rosser* property in the general framework, this was not enough.

To overcome this limitation to restricted versions of PTS, we extended Adams' system by adding a second annotation to the applications. In his paper, he rejected this solution because it introduces a new constraint one has to check when one wants to reduce a β-redex, and he did not investigate how to handle this additional complication. Such methods have already been tried to prove normalization results for PTS in (Melliès & Werner, 1997) and for correctness and completeness results in (Streicher, 1991), but we had to adapt it without any normalization requirement.

All of this has led us to define a variant of TPOSR that we call *Pure Type System based on Annotated Typed Reduction*. This system is built on a trade-off : this additional annotation allows us to get more information from our typing judgments, but it adds new constraints in the typed reduction that we will have to face. We will now see in details how it is defined and what are the difficulties introduced by this new annotation.

Structure of Annotated Terms

$A,B,M,N ::= s \mid x \mid M_{\Pi x^A.B}N \mid \lambda x^A.M \mid \Pi x^A.B$

All the other notions (context, substitution and untyped reduction) described for the terms of PTS are defined in the same way for PTS_{atr}, with their natural adaptation to the annotated applications. To avoid confusion between the reductions, we will write $\rightarrow_p$ for untyped parallel reduction in PTS_{atr} and $\twoheadrightarrow$ for its transitive closure (since PTS_{atr} is a parallel system, using a one-step parallel reduction will be easier, but its closure is still the same as the usual one-step β-reduction). We define an erasure procedure $|\ |$ by induction on the structure of terms that maps annotated PTS_{atr} terms to non-annotated PTS ones, by

$$\frac{}{\emptyset_{wf}}\ \text{EMPTY} \qquad \frac{\Gamma \vdash A \rhd ? : s \qquad x \notin Dom(\Gamma)}{\Gamma(x:A)_{wf}}\ \text{EXTEND}$$

$$\frac{\Gamma_{wf} \qquad (s,t) \in \mathscr{A}x}{\Gamma \vdash s \rhd s : t}\ \text{SORT} \qquad \frac{\Gamma_{wf} \qquad \Gamma(x) = A}{\Gamma \vdash x \rhd x : A}\ \text{VAR}$$

$$\frac{(s_1,s_2,s_3) \in \mathscr{R}el \qquad \Gamma \vdash A \rhd A' : s_1 \qquad \Gamma(x:A) \vdash B \rhd B' : s_2}{\Gamma \vdash \Pi x^A.B \rhd \Pi x^{A'}.B' : s_3}\ \text{PROD}$$

$$\frac{\Gamma \vdash A \rhd A' : s_1 \qquad (s_1,s_2,s_3) \in \mathscr{R}el \qquad \Gamma(x:A) \vdash B \rhd ? : s_2 \qquad \Gamma(x:A) \vdash M \rhd M' : B}{\Gamma \vdash \lambda x^A.M \rhd \lambda x^{A'}.M' : \Pi x^A.B}\ \text{LAM}$$

$$\frac{(s_1,s_2,s_3) \in \mathscr{R}el \qquad \Gamma \vdash A \rhd A' : s_1 \qquad \Gamma(x:A) \vdash B \rhd B' : s_2 \qquad \Gamma \vdash M \rhd M' : \Pi x^A.B \qquad \Gamma \vdash N \rhd N' : A}{\Gamma \vdash M_{\Pi x^A.B} N \rhd M'_{\Pi x^{A'}.B'} N' : B[N/x]}\ \text{APP}$$

$$\frac{\begin{array}{c}\Gamma \vdash A \rhd ? : s_1 \qquad \Gamma \vdash A' \rhd ? : s_1 \\ \Gamma \vdash A_0 \rhd^+ A : s_1 \qquad \Gamma \vdash A_0 \rhd^+ A' : s_1 \qquad (s_1,s_2,s_3) \in \mathscr{R}el \\ \Gamma(x:A) \vdash B \rhd B' : s_2 \qquad \Gamma(x:A) \vdash M \rhd M' : B \qquad \Gamma \vdash N \rhd N' : A\end{array}}{\Gamma \vdash (\lambda x^A.M)_{\Pi x^{A'}.B} N \rhd M'[N'/x] : B[N/x]}\ \text{BETA}$$

$$\frac{\Gamma \vdash M \rhd N : A \qquad \Gamma \vdash A \rhd B : s}{\Gamma \vdash M \rhd N : B}\ \text{RED} \qquad \frac{\Gamma \vdash M \rhd N : A \qquad \Gamma \vdash B \rhd A : s}{\Gamma \vdash M \rhd N : B}\ \text{EXP}$$

$$\frac{\Gamma \vdash M \rhd N : A}{\Gamma \vdash M \rhd^+ N : A}\ \text{REDS-INTRO} \qquad \frac{\Gamma \vdash M \rhd^+ N : A \qquad \Gamma \vdash N \rhd^+ P : A}{\Gamma \vdash M \rhd^+ P : A}\ \text{REDS-TRANS}$$

Fig. 3. Typing Rules for the PTS_{atr} system

inductively removing the additional typing information within the applications.

The typing rules of PTS_{atr} are presented in Fig. 3. As a shortcut, we will use the notations $\Gamma \vdash M \rhd N : A,B$ for "$\Gamma \vdash M \rhd N : A$ and $\Gamma \vdash M \rhd N : B$", and $\Gamma \vdash M \rhd ? : A$ for "there is some N such that $\Gamma \vdash M \rhd N : A$".

The transitive-closure of $\rhd$ is written as $\rhd^+$, and the transitive-symmetric closure of $\rhd$ as $\cong_\beta$, restricted to terms typed by sorts. We will not need a full notion of equality since this new judgment already embeds a notion of reduction. The $\cong_\beta$ judgment has to be understood as an equality at "the level of types", where we do not demand to keep the same sort at every transitivity step. We will need this to be able to state the *Generation Lemmas* correctly, since we do not have the *Uniqueness of Types* in the general case.

$$\frac{\Gamma \vdash A \rhd B : s}{\Gamma \vdash A \cong_\beta B}\ \text{EQ-INTRO} \qquad \frac{\Gamma \vdash B \cong_\beta A}{\Gamma \vdash A \cong_\beta B}\ \text{SYM} \qquad \frac{\Gamma \vdash A \cong_\beta B \qquad \Gamma \vdash B \cong_\beta C}{\Gamma \vdash A \cong_\beta C}\ \text{TRANS}$$

Fig. 4. Type Equality in PTS_{atr}

So far, we are juggling with a few variants of β-equality, so we will now recall all our notations as a remainder to avoid confusion:

Notation	Terms	Systems	Meaning
$M \equiv N$	all	all	syntactic (α-conversion)
$M =_\beta N$	non-annotated	PTS	β-conversion
$\Gamma \vdash_e M =_\beta N : T$	non-annotated	PTS_e	β-conversion with typing constraints
$\Gamma \vdash M \cong_\beta N$	annotated	PTS_{atr}	β-conversion with typing constraints

The BETA rule may seem complicated at first, but its meaning is to ensure that there is a conversion path from the annotation of the λ case A, to the annotation of the application A', where each step is *typed by the sort* s_1 (which is the first sort of the triple). The equality $\cong_\beta$ ensures that each step is typed by a sort, but does not guarantee that each step use the same one, so we can not use it directly. And using another equality where we ensure that each step lives in the same type (much like PTS_e equality) did not help at all in the following proofs. That is the reason why we stated the system with this "common expanded form" rather than with another new judgment that will not be used elsewhere.

3.2 General properties of PTS_{atr}

From now on, we consider the general case of PTS, without any restrictions: we can start to prove some properties of PTS_{atr} (by mutual induction over $\rhd$ and $\rhd^+$ at once):

Lemma 3.1 (*Weakening*)

1. If $\Gamma_1\Gamma_2 \vdash M \rhd N : B, \Gamma_1 \vdash A \rhd ? : s$ and $x \notin Dom(\Gamma_1\Gamma_2)$ then $\Gamma_1(x:A)\Gamma_2 \vdash M \rhd N : B$.
2. If $\Gamma_1\Gamma_2 \vdash M \rhd^+ N : B, \Gamma_1 \vdash A \rhd ? : s$ and $x \notin Dom(\Gamma_1\Gamma_2)$ then $\Gamma_1(x:A)\Gamma_2 \vdash M \rhd^+ N : B$.
3. If $\Gamma_1\Gamma_2\ _{wf}, \Gamma_1 \vdash A \rhd ? : s$ and $x \notin Dom(\Gamma_1\Gamma_2)$ then $\Gamma_1(x:A)\Gamma_2\ _{wf}$.

Lemma 3.2 (*Parallel Substitution*)

1. If $\Gamma_1(x:A)\Gamma_2 \vdash M \rhd N : B$ and $\Gamma_1 \vdash P \rhd P' : A$ then $\Gamma_1\Gamma_2[P/x] \vdash M[P/x] \rhd N[P'/x] : B[P/x]$.
2. If $\Gamma_1(x:A)\Gamma_2 \vdash M \rhd^+ N : B$ and $\Gamma_1 \vdash P \rhd P' : A$ then $\Gamma_1\Gamma_2[P/x] \vdash M[P/x] \rhd^+ N[P'/x] : B[P/x]$.
3. If $\Gamma_1(x:A)\Gamma_2\ _{wf}$ and $\Gamma_1 \vdash P \rhd ? : A$ then $\Gamma_1\Gamma_2[P/x]_{wf}$.

We extend the notion of equality on terms to equality on contexts, which are nothing but ordered lists of terms:

Context Conversion

- $\emptyset \cong_\beta \emptyset$.

- If $\Gamma \cong_\beta \Gamma'$, $\Gamma \vdash A \cong_\beta B$ and $x \notin Dom(\Gamma)$, then $\Gamma(x:A) \cong_\beta \Gamma'(x:B)$.

Lemma 3.3 (*Conversion in Context*)
- If $\Gamma \vdash M \rhd N : A$ and $\Gamma \cong_\beta \Gamma'$ then $\Gamma' \vdash M \rhd N : A$.
- If $\Gamma \vdash M \rhd^+ N : A$ and $\Gamma \cong_\beta \Gamma'$ then $\Gamma' \vdash M \rhd^+ N : A$.
- If $\Gamma \vdash A \cong_\beta B$ and $\Gamma \cong_\beta \Gamma'$ then $\Gamma' \vdash A \cong_\beta B$.

Lemma 3.4 (*Left-Hand and Right-Hand Typability*)
1. If $\Gamma \vdash M \rhd N : A$ or $\Gamma \vdash M \rhd^+ N : A$, then $\Gamma \vdash M \rhd M : A$.
2. If $\Gamma \vdash M \rhd N : A$ or $\Gamma \vdash M \rhd^+ N : A$, then $\Gamma \vdash N \rhd N : A$.
3. If $\Gamma \vdash A \cong_\beta B$, then $\Gamma \vdash A \rhd A : s$ and $\Gamma \vdash B \rhd B : t$ for some sorts s and t.

The following lemma is an adapted version of the *Generation Lemma* introduced for PTS. By adding both annotations, we do not have to "guess" the domain and co-domain of an application anymore.

Lemma 3.5 (*Generation*)
1. If $\Gamma \vdash s \rhd N : T$ then $N \equiv s$ and there is t such that $(s,t) \in Ax$ and either $T \equiv t$ or $\Gamma \vdash T \cong_\beta t$.
2. If $\Gamma \vdash x \rhd N : T$ then $N \equiv x$ and there is A such that $\Gamma(x) = A$ and $\Gamma \vdash T \cong_\beta A$.
3. If $\Gamma \vdash \Pi x^A.B \rhd N : T$ then there are A', B', s_1, s_2, s_3 such that $N \equiv \Pi x^{A'}.B'$, $(s_1,s_2,s_3) \in Rel$, $\Gamma \vdash A \rhd A' : s_1$, $\Gamma(x:A) \vdash B \rhd B' : s_2$ and either $T \equiv s_3$ or $\Gamma \vdash T \cong_\beta s_3$.
4. If $\Gamma \vdash \lambda x^A.M \rhd N : T$ then there are $A', M', B, B', s_1, s_2, s_3$ such that $N \equiv \lambda x^{A'}.M'$, $(s_1,s_2,s_3) \in Rel$, $\Gamma \vdash A \rhd A' : s_1$, $\Gamma(x:A) \vdash B \rhd B' : s_2$, $\Gamma(x:A) \vdash M \rhd M' : B$ and $\Gamma \vdash T \cong_\beta \Pi x^A.B$.
5. If $\Gamma \vdash P_{\Pi x^U.B} Q \rhd N : T$ then there are $A, A', B', Q', s_1, s_2, s_3$ such that $(s_1,s_2,s_3) \in Rel$, $\Gamma \vdash A \rhd A' : s_1$, $\Gamma(x:A) \vdash B \rhd B' : t_2$, $\Gamma \vdash Q \rhd Q' : A$, $\Gamma \vdash T \cong_\beta B[Q/x]$ and
   - either (APP case) $U \equiv A$, $\Gamma \vdash P \rhd P' : \Pi x^A.B$ and $N \equiv P'_{\Pi x^{A'}.B'} Q'$ for some P'
   - or (BETA case) $U \equiv A''$, $P \equiv \lambda x^A.R$, $\Gamma(x:A) \vdash R \rhd R' : B$, $N \equiv R'[Q'/x]$, $\Gamma \vdash A_0 \rhd^+ A'' : s_1$ and $\Gamma \vdash A_0 \rhd^+ A : s_1$ for some A_0, A'', R, R'.

One of the key-point to prove the *Church-Rosser* property for β-reduction (more exactly, to prove that the usual reduction and the parallel one have the same transitive closure) is that β enjoys some nice multi-step congruence properties like:

- If $A \twoheadrightarrow_\beta B$ and $C \twoheadrightarrow_\beta D$, then $\Pi x^A.C \twoheadrightarrow_\beta \Pi x^B.D$
- If $A \twoheadrightarrow_\beta B$ and $M \twoheadrightarrow_\beta N$, then $\lambda x^A.M \twoheadrightarrow_\beta \lambda x^B.N$
- ...

However, to have the same properties in PTS_{atr}, that is with type restrictions to fulfill, those lemmas can be hard to prove, especially for the application case. By only considering the functional case, which enjoys *Type Uniqueness*, Adams got rid of this trouble and managed to prove those extensions to TPOSR quite easily. Without this uniqueness property, we need another way to be able to find the right typing information.

To prove those multi-step congruence results for PTS_{atr}, we need to check that some terms are typed by the correct sorts (for example in the application case, we need to check that terms are typed by the triple of sorts in *Rel*). One practical case is when we know that

$\Gamma \vdash A \rhd ? : s$ and $\Gamma \vdash A \rhd^+ A' : t$, but we need the latter statement typed by s. With *Type Uniqueness*, we would be able to prove that $s \equiv t$, but this is not true in the general case. What we would like to do it to keep the reduction skeleton of the second statement and use it with the types of the first judgment.

According to us, the following theorem is the best tool to achieve this task:

Theorem 3.6 (*Exchange of Types*)
If $\Gamma \vdash M \rhd N : A$ and $\Gamma \vdash M \rhd P : B$, then $\Gamma \vdash M \rhd N : B$ and $\Gamma \vdash M \rhd P : A$.

Proof
By induction, there are no difficult cases since we have the co-domain annotations on the applications. □

The heart of this theorem is to keep the reduction structure of a derivation and allowing to change the type annotations inside, if we have a witness that these annotations are correct. We can directly extend this result to multi-step reduction:

Corollary 3.7 (*Exchange of Types in multi-step reduction*)
If $\Gamma \vdash M \rhd^+ N : A$ and $\Gamma \vdash M \rhd ? : B$, then $\Gamma \vdash M \rhd^+ N : B$.

It allows us to prove that the following transitivity rule for $\rhd^+$ is admissible:

$$\frac{\Gamma \vdash M \rhd^+ N : A \qquad \Gamma \vdash N \rhd^+ P : B}{\Gamma \vdash M \rhd^+ P : A}\ \text{REDS-TRANS-ALT}$$

This is the key lemma to prove our multi-step congruence lemma for PTS_{atr}:

Lemma 3.8 (*Multi-step Congruences and Generations*)

- Congruences:
  - If $\Gamma \vdash A \rhd^+ A' : s_1$, $\Gamma(x : A) \vdash B \rhd^+ B' : s_2$ and $(s_1, s_2, s_3) \in \mathcal{R}el$, then $\Gamma \vdash \Pi x^A.B \rhd^+ \Pi x^{A'}.B' : s_3$.
  - If $\Gamma \vdash A \rhd^+ A' : s_1$, $\Gamma(x : A) \vdash M \rhd^+ M' : B$, $\Gamma(x : A) \vdash B \rhd ? : s_2$ and $(s_1, s_2, s_3) \in \mathcal{R}el$, then $\Gamma \vdash \lambda x^A.M \rhd^+ \lambda x^{A'}, M' : \Pi x^A.B$.
  - If $\Gamma \vdash A \rhd^+ A' : s$, $\Gamma(x : A) \vdash B \rhd^+ B' : t$, $\Gamma \vdash M \rhd^+ M' : \Pi x^A.B$, and $\Gamma \vdash N \rhd^+ N' : A$, then $\Gamma \vdash M_{\Pi x^A.B} N \rhd^+ M'_{\Pi x^{A'}.B'} N' : B[N/x]$.
- (Multi-step) Generation:
  - If $\Gamma \vdash \Pi x^A.B \rhd^+ N : T$ then there are A', B', s_1, s_2, s_3 such that $(s_1, s_2, s_3) \in \mathcal{R}el$, $N \equiv \Pi x^{A'}.B'$, $\Gamma \vdash A \rhd^+ A' : s_1$, $\Gamma(x : A) \vdash B \rhd^+ B' : s_2$ and $\Gamma \vdash T \cong_\beta s_3$ or $T \equiv s_3$.
  - If $\Gamma \vdash \lambda x^A.M \rhd^+ N : T$ then there are A', M', B, s_1, s_2, s_3 such that $(s_1, s_2, s_3) \in \mathcal{R}el$, $N \equiv \lambda x^{A'}.M'$, $\Gamma \vdash A \rhd^+ A' : s_1$, $\Gamma(x : A) \vdash M \rhd^+ M' : B$, $\Gamma(x : A) \vdash B \rhd ? : s_2$ and $\Gamma \vdash T \cong_\beta \Pi x^A.B$.
  - If $\Gamma \vdash s \rhd^+ N : T$, then there is t such that $N \equiv s$, $(s, t) \in \mathcal{A}x$, and $\Gamma \vdash T \cong_\beta t$ or $T \equiv t$.

This exchange of types will also be used in the proof of the *Church-Rosser* property to avoid building the right sets of sorts in $\mathcal{R}el$ at some minor stage of the proof. However, we will use it extensively while proving that well-typed terms in PTS can be correctly annotated into well-typed annotated terms in PTS_{atr}.

Lemma 3.9 (*Type Correctness*)
If $\Gamma \vdash M \rhd N : A$, then there is $s \in \mathit{Sorts}$ such as either: $A \equiv s$ or $\Gamma \vdash A \rhd ? : s$.

Theorem 3.10 (*From PTS$_{atr}$ to PTS and PTS$_e$*)
1. If $\Gamma \vdash M \rhd N : A$ then $|\Gamma| \vdash |M| : |A|$,
   $|\Gamma| \vdash |N| : |A|$ and $|M| =_\beta |N|$.
2. If $\Gamma \vdash M \rhd N : A$ then $|\Gamma| \vdash_e |M| : |A|$,
   $|\Gamma| \vdash_e |N| : |A|$ and $|\Gamma| \vdash_e |M| =_\beta |N| : |A|$.

Proof
This proof is much like the translation from PTS$_e$ to PTS: we have more typing informations in PTS$_{atr}$ than in PTS or PTS$_e$, so we just need to remove the additional annotations. Since $\rhd$ has been designed to mimic the parallel reduction for β, it is quite easy to show that erased terms are still connected by typed or untyped β-conversion. □

Corollary 3.11 (*Sort and* Π*-types incompatibility*)
It is impossible to prove that $\Gamma \vdash \Pi x^A.B \cong_\beta s$ for any Γ, A, B, s.

Proof
The proof relies on a translation of the equality judgment $\Gamma \vdash \Pi x^A.B =_\beta s$ in the PTS by erasure of the annotations with the first part of Theorem 3.10. The confluence of β-reduction forbids that $\Pi x^{|A|}.|B| =_\beta s$ in any way. □

At this point we need to recall what we said about the order we used to prove things in PTS. We did not present any kind of confluence for PTS$_{atr}$. The reason is that, in a typed framework like PTS$_e$ or PTS$_{atr}$, the *Confluence* and the *Church-Rosser* properties are a blocking step. Since they mix together typing and reduction, it is difficult to find a proof without involving the *Subject Reduction* of the system, and the proof of this theorem involves already knowing the Π-injectivity property (as required for PTS in the previous section) which comes from *Confluence*.

3.3 *The* Church-Rosser *Property in PTS$_{atr}$*

The next step in the meta-theory is to prove the *Church-Rosser* property by proving that PTS$_{atr}$ enjoys the *Diamond Property*:

Theorem 3.12 (*Diamond Property*)
If $\Gamma \vdash M \rhd N : A$ and $\Gamma \vdash M \rhd P : B$, then there is Q such that

$$\Gamma \vdash N \rhd Q : A, B \qquad \Gamma \vdash N \rhd Q : A, B$$
$$\Gamma \vdash P \rhd Q : A, B \qquad \Gamma \vdash P \rhd Q : A, B$$

We are trying to close the classic *Church-Rosser* diamond diagram in a typed way. In all previous attempts (Adams, 2006; Siles & Herbelin, 2010), the main issue was to be able to close the cases involving an application constructor: APP/APP, APP/BETA and BETA/APP. We lacked information about the co-domain (say D) of the application:

1. some types involved in the conclusion of those judgments are substituted (e.g. $D[N/x]$), so we do not have the complete typing information for D.
2. some induction hypotheses over the co-domain of types do not always reflect the context of the hypothesis we actually have.

The first problem is "easily" solved by adding the D as an annotation. But it is this additional annotation that makes the second problem arise: it forbids us to use one of our induction hypothesis. During the proof, the induction hypothesis requires the context to be the same in both branches of the theorem and for the APP case, we needed to prove that it was actually the case.

In his proof, Adams relies on the *Uniqueness of Typing* which comes from the functionality, and in the semi-full case, we relied on the *Shape of Types* (Siles & Herbelin, 2010) to make the contexts match. To get rid of both constraints over the PTS, we will use here the new annotation in applications, that will *force* the context to match: now in the APP/APP case, co-domain contexts are syntactically the same, and in the other cases (where we will β-reduce), we have enough typing information to type the resulting substitution.

We will not give more details about the second issue here since we no longer face it (explanations and a concrete example can be found in Siles & Herbelin (2010). However, since it is the first time that the new annotation comes in handy, we can now explain our choices for it.

The annotation is here to have a full remainder of the function space: if $\lambda x^A.M$ of type $\Pi x^A.B$ is applied to N of type A, we want to have both A and B available while looking at the β-redex. Our first attempt was to put *syntactically* the same A in the annotation, and thus allowing the reduction of the redex only if the annotation matches exactly the domain of the function. But this approach failed, and made us realized that we need to annotate by any A' convertible to A. However, this notion of conversion has to be more strict than our $\cong_\beta$ judgment: we need to enforce that each conversion step stays in the same sort, much like the equality judgments for PTS_e.

We could have used two different notions of conversion, one that cares about the type, and one that only cares about the types being sorts, but the first one was only needed for this new annotation, and as soon as we proved *Confluence*, we will always break it into two multi-step reductions. Instead, we tried to find another "self-contained" notion of strict conversion, with the judgments we built so far around $\rhd$, $\rhd^+$ and $\cong_\beta$. Having a common expanded term satisfied all our requirements:

- all the steps between the domain and the annotation are well-typed, by the very same sort.
- we do not have to introduce a new kind of judgment.
- it behaves nicely in the proof of the *Church-Rosser* property.

The main reason why this choice behaves so nicely is that PTS_{atr} is a *reduction* system: it is directed. The domain and the annotation are always reduced in the same direction. Informally, if $A_0 \rhd^+ A$ and $A_0 \rhd^+ A'$, since both A and A' can only be reduced, we just have to append the new reductions steps to the sequences starting from A_0. In short, we never need to "guess" what will be the common expanded term.

Doing so, the proof of the *Diamond Property* becomes quite straightforward by induction, since we pushed all the issues inside the new annotation. However, those issues did not disappear: we will face them when we will prove that the annotations are correct.

3.4 Consequences of the Church-Rosser property

With the *Church-Rosser* property, we can finally settle with all the missing pieces of theory that we do not know how to prove directly in a typed framework:

Lemma 3.13 (*Confluence*)
If $\Gamma \vdash A \cong_\beta B$, there are C, s, t such that $\Gamma \vdash A \rhd^+ C : s$ and $\Gamma \vdash B \rhd^+ C : t$.

Lemma 3.14 (*Weak* Π*-injectivity for PTS*$_{atr}$)
If $\Gamma \vdash \Pi x^A.B \cong_\beta \Pi x^C.D$ then $\Gamma \vdash A \cong_\beta C$ and $\Gamma(x:A) \vdash B \cong_\beta D$.

Since strong injectivity does not hold for PTS$_{atr}$ (the same counterexample we used for PTS$_e$ also works here), we stated a weaker form of injectivity. However, this Π-*injectivity* for $\cong_\beta$ along with the *Exchange of Types* properties are enough for the rest of the development.

Theorem 3.15 (*Subject Reduction*)
If $\Gamma \vdash M \rhd ? : A$ and $M \to_p N$ then $\Gamma \vdash M \rhd^+ N : A$.

Proof
This is the first place where we will encounter the difficulties that we postponed in the proof of *Church-Rosser* property It is interesting to notice that we did not manage to prove the conclusion of *Subject Reduction* as a one step PTS$_{atr}$ reduction: all the conversion we have to do to make the annotations in the application match forced us to have a multi-step version of the conclusion. However, this will not be a problem for the following proofs.

The proof is done by induction on $M \to_p N$: as usual, most cases are trivial. In the case of application congruence, some type conversions are required, but everything is directly available. However, if M is a β-redex which is reduced, we need to show that we have the right to do this reduction according to the typing rule BETA. We will make an extensive use of *Confluence* and *Exchange of Types* to show that everything is fine.

The situation is the following: we want to prove that

$$\Gamma \vdash (\lambda x^A.M_{\Pi x^{A'}.B'})N \rhd^+ M'[N'/x] : B'[N/x]$$

knowing that the β-redex is well-typed. By inversion, we have two choices: either the redex is typed as an application, or it is typed with the BETA rule. In the second case, we directly have all the information to conclude. However, in the first case, we need to use the *Generation* Lemma to retrieve typing information from the application and from the λ-term. We get the following judgments:

- $\Gamma \vdash A \rhd ? : s_1$, $\Gamma(x:A) \vdash M \rhd ? : B$ and $\Gamma(x:A) \vdash B \rhd ? : s_2$ where $(s_1, s_2, s_3) \in \mathcal{R}el$.
- $\Gamma \vdash A' \rhd ? : t_1$, $\Gamma(x:A') \vdash B' \rhd ? : t_2$ where $(t_1, t_2, t_3) \in \mathcal{R}el$.
- $\Gamma \vdash N \rhd ? : A'$ and $\Gamma \vdash \Pi x^A.B \cong_\beta \Pi x^{A'}.B'$.

Using Π-*injectivity*, we can show that $\Gamma \vdash A \cong_\beta A'$, but as we said before, this is not enough to trigger the reduction of the redex, since A and A' are not typed by the same sort, and we do not know any common expanded form for them. However, by *Confluence*, we can find common reduced terms for A and A' , and also for B and B':

- $\Gamma \vdash A \rhd^+ A_0 : s$ and $\Gamma \vdash A' \rhd^+ A_0 : t$.

- $\Gamma \vdash B \rhd^+ B_0 : s'$ and $\Gamma(x:A) \vdash B' \rhd^+ B_O : t'$.

Using the *Exchange of Types*, we can replace s by s_1, t by t_1, s' by s_2 and t' by t_2. Doing so, we can prove that $\Gamma \vdash \lambda x^A.M_{\Pi x^{A'}.B'} N \rhd^+ \lambda x^A.M_{\Pi x^{A_0}.B_0} N : B'[N/x]$. With this new redex, we have everything at hand to fire the reduction and prove that

$\Gamma \vdash \lambda x^A.M_{\Pi x^{A_0}.B_0} N \rhd M[N/x] : B_0[N/x]$.

With (REDS-TRANS-ALT), and the *Substitution* Lemma, we can now glue both reductions and conclude the final case of *Subject Reduction*. □

4 Equivalence of PTS$_{atr}$ and PTS

4.1 Confluence of the annotation process

The last step to prove the equivalence is to prove the correctness of annotations, i.e. to prove that every judgment $\Gamma \vdash M : T$ can be annotated into a valid PTS$_{atr}$ derivation $\Gamma^+ \vdash M^+ \rhd M^+ : T^+$ where $|\Gamma^+| \equiv \Gamma$, $|M^+| \equiv M$ and $|T^+| \equiv T$.

To do so, we need to show some basic properties of the annotation process. Since there are several ways to annotate a term, we will face some difficult situations while performing induction. Let us take a simple example with the construction of Π-types with the PI rule:

$$\frac{\Gamma \vdash A : s_1 \qquad \Gamma(x:A) \vdash B : s_2 \qquad (s_1,s_2,s_3) \in \mathscr{R}el}{\Gamma \vdash \Pi x^A.B : s_3}\ \text{PI}$$

By induction, we get that $\Gamma_1 \vdash A_1 \rhd A_1 : s_1$ and $\Gamma_2(x:A_2) \vdash B_2 \rhd B_2 : s_2$ with the equalities $|\Gamma_1| \equiv |\Gamma_2| = \Gamma$, $|B_2| \equiv B$ and $|A_1| \equiv |A_2| = A$. To build a Π-type from those two judgments, we need to relate Γ_1 to Γ_2 and A_1 to A_2 *in PTS$_{atr}$*. More precisely, we need to show that if two annotated types come from the same non-annotated term, and if they are well-typed in PTS$_{atr}$, they are equivalent in PTS$_{atr}$. With such a property, we would be able to state a similar lemma for contexts and prove that our annotation procedure is correct.

However, we have to recall that what we call here types are just terms typed by a sort, and their typing judgment may use β-redexes, which will involve "non-types". So we will state a more general lemma about the conversion of different annotated versions of a same PTS term.

Lemma 4.1 (*Erased Confluence*)
If $|M| \equiv |N|$, $\Gamma \vdash M \rhd ? : A$ and $\Gamma \vdash N \rhd ? : B$, then there is R such that
$\Gamma \vdash M \rhd^+ R : A$ and $\Gamma \vdash N \rhd^+ R : B$.

Proof
The proof is done by induction on M, the only difficult part is again the application case:

$$M \equiv P_{\Pi x^{A_0}.D} Q,\ N \equiv P'_{\Pi x^{A'_0}.D'} Q' \qquad |P| \equiv |P'|,\ |Q| \equiv |Q'|$$

By generation, we get that P, P', Q and Q' are well-typed, so by induction, there are P_0, Q_0 such that:

$$\Gamma \vdash P \rhd^+ P_0 : \Pi x^C.D \qquad \Gamma \vdash Q \rhd^+ Q_0 : C$$
$$\Gamma \vdash P' \rhd^+ P_0 : \Pi x^{C'}.D' \qquad \Gamma \vdash Q' \rhd^+ Q_0 : C'$$

and some additional information relating A_0 and A'_0 to C and C' depending on the way M was typed (BETA or APP).

In the functional case (where only one annotation is needed), this is quite trivial : thanks to the *Uniqueness of Types* applied to P_0 and Π-injectivity we get that $\Gamma(x : C) \vdash D \cong_\beta D'$. By *Confluence*, we get a common reduct D_0 for D and D', so the common reduct of M and N is $P_0\ {}_{D_0} Q_0$.

We need to be a little more subtle here: for the semi-full case (see Siles & Herbelin, 2010), we showed that terms can be classified in two families whose types have very particular shapes. Fortunately, the full generality of this classification is not needed here:

Lemma 4.2 (*Weak shape of type*)
If $\Gamma \vdash M \rhd ? : A$ and $\Gamma \vdash M \rhd ? : B$, then:

- either $\Gamma \vdash A \cong_\beta B$
- or we are in the following cases:
  1. there are U and V such that $\Gamma \vdash M \rhd \lambda x^U.V : A$ and $\Gamma \vdash M \rhd \lambda x^U.V : B$.
  2. there is s such that $\Gamma \vdash M \rhd s : A$ and $\Gamma \vdash M \rhd s : B$.
  3. there is U and V such that $\Gamma \vdash M \rhd \Pi x^U.V : A$ and $\Gamma \vdash M \rhd \Pi x^U.V : B$.

The proof of this lemma is quite trivial by induction, and relies on the fact that we have the annotation of co-domains at hand.

From now on, we will mainly focus on P_0: we can apply the previous lemma to it and, for the first part of the conclusion, conclude almost like the functional case. By generation, we also got a way to prove that $\Gamma \vdash A_0 \cong_\beta A'_0$, depending on the constructor used. By *Confluence*, we can get a common reduct A'', and use $P_0\ {}_{\Pi x^{A''}.D_0} Q_0$ to close the lemma.

If we are in the second part of the conclusion, the only relevant case is the first one: since P_0 is typed by a Π-types, it can not reduce itself to a sort or another Π-type. The reason is because with the *Generation* lemma, we know that the type of a sort or a Π-type is always convertible to a sort. If they could be typed by a Π-type, we would end up having a judgment of the form $\Gamma \vdash \Pi x^A.B \cong_\beta s$ which is impossible due to Corollary 3.11.

In the last remaining case, there are U and V such that:

- $\Gamma \vdash P_0 \rhd \lambda x^U.V : \Pi x^C.D$
- $\Gamma \vdash P_0 \rhd \lambda x^U.V : \Pi x^{C'}.D'$

We just created a β-redex since P_0 is going to be applied, so this time, the common reduced term will be the result of the β-reduction initiated by P_0 instead of just a simple application.

Actually, we still need to show that we are allowed to reduce this redex, just as we needed to show it for *Subject Reduction*: this is the second place where we are facing quite technical points because of the new annotations. There are four different cases to handle here, depending on how M and M' are originally typed (by BETA or APP), but each can be closed by extensive use of *Confluence* and *Exchange of Types*. The main idea behind each case is the same, and follows this scheme:

$$\begin{array}{lll}
\Gamma \vdash P_{\Pi x^U.D} Q & \rhd^+ P_{0\ \Pi x^U.D} Q & : D[Q/x] \\
& \rhd^+ (\lambda x^U.V)_{\Pi x^U.D} Q & : D[Q/x] \\
& \rhd^+ V[Q/x] & : D[Q/x] \\
& \rhd^+ V[Q_0/x] & : D[Q/x] \\
\Gamma \vdash P'_{\Pi x^U.D'} Q' & \rhd^+ P_{0\ \Pi x^U.D'} Q' & : D'[Q'/x] \\
& \rhd^+ (\lambda x^U.V)_{\Pi x^U.D'} Q' & : D'[Q'/x] \\
& \rhd^+ V[Q'/x] & : D'[Q'/x] \\
& \rhd^+ V[Q_0/x] & : D'[Q'/x]
\end{array}$$

In the end, we manage to find a common reduct in each type without having to find a common reduct for the annotations, which concludes the proof of this lemma. □

4.2 Consequences of the Erased Confluence

With the general statement for all terms, we can now show what we needed about types and contexts:

Lemma 4.3 (*Erased Conversion*)

1. If $|A| \equiv |B|$, $\Gamma \vdash A \rhd ? : s$ and $\Gamma \vdash B \rhd ? : t$ then $\Gamma \vdash A \cong_\beta B$.
2. If $|\Gamma_1| \equiv |\Gamma_2|$ and $\Gamma_1 \vdash M \rhd N : A$, then $\Gamma_2 \vdash M \rhd N : A$.

Proof

The first statement directly follows from Lemma 4.1. The second is a consequence of the first one, by simple induction on the length of Γ_1. □

Now let us go back to the annotation of Π-types. With Lemma 4.3, we can derive the fact that $\Gamma_1 \vdash A_1 \cong_\beta A_2$ and $\Gamma_1 \equiv \Gamma_2$. By context conversion, we can exchange the contexts and we end up proving that $\Gamma_1 (x : A_1) \vdash B_2 \rhd B_2 : s_2$, and so we can finally build the annotated judgment $\Gamma_1 \vdash \Pi x^{A_1}.B_2 \rhd \Pi x^{A_1}.B_2 : s_3$, with $|\Gamma_1| \equiv \Gamma$, $|A_1| \equiv A$ and $|B_2| \equiv B$.

By doing the same process for each constructor, we can now conclude the last missing piece of the whole equivalence process:

Theorem 4.4 (*From PTS to PTS_{atr}*)
If $\Gamma \vdash M : T$, then there are Γ^+, M^+, T^+ such that $\Gamma^+ \vdash M^+ \rhd M^+ : T^+$, $|\Gamma^+| \equiv \Gamma$, $|M^+| \equiv M$ and $|T^+| \equiv T$.

Proof

Since we have managed to prove *Subject Reduction* and Lemma 4.3, the proof is almost the same as for Adams' TPOSR. A few type exchanges are needed in the BETA case but it does not involve complicated nor technical things. □

Finally, all of this leads us to state that:

Theorem 4.5 (*Equivalence of PTS and PTS_e*)

1. $\Gamma \vdash M : T$ iff $\Gamma \vdash_e M : T$.
2. $\Gamma \vdash_e M =_\beta N : T$ iff $\Gamma \vdash M : T$, $\Gamma \vdash N : T$ and $M =_\beta N$.

Proof

This is just a combination of all the previous theorems:

- If $\Gamma \vdash_e M : T$, then by Theorem 2.11, we have $\Gamma \vdash M : T$.
- If $\Gamma \vdash M : T$, by Theorem 4.4 we know that $\Gamma^+ \vdash M^+ \rhd M^+ : T^+$ with $|\Gamma^+| \equiv \Gamma$, $|M^+| \equiv M$ and $|T^+| \equiv T$. By Theorem 3.10, $|\Gamma^+| \vdash_e |M^+| : |T^+|$ which is equal to $\Gamma \vdash_e M : T$.
- If $\Gamma \vdash_e M =_\beta N : T$, so we conclude by Theorem 2.11.
- If $\Gamma \vdash M : T$, $\Gamma \vdash N : T$ and $M =_\beta N$, by *Confluence*, there is P such that $M \twoheadrightarrow_\beta P$ and $N \twoheadrightarrow_\beta P$. By Theorem 4.4, there are Γ^+, M^+, T^+ such that $|\Gamma^+| \equiv \Gamma$, $|M^+| \equiv M$, $|T^+| \equiv T$ and $\Gamma^+ \vdash M^+ \rhd M^+ : T^+$. Let us consider P^+ such that $|P^+| \equiv P$ and $M^+ \twoheadrightarrow P^+$ (such a term always exists, the proof is a simple induction on the structure of M).

$$\begin{array}{lll} & \Gamma^+ \vdash M^+ \rhd M^+ : T^+ & \\ \Rightarrow & \Gamma^+ \vdash M^+ \rhd^+ P^+ : T^+ & \text{(Subject Reduction)} \\ \Rightarrow & \Gamma \vdash_e M =_\beta P : T & \text{(Theorem 3.10 and TRANS)} \end{array}$$

We do the same to conclude that $\Gamma \vdash_e N =_\beta P : T$, so by SYM and TRANS, we finally have $\Gamma \vdash_e M =_\beta N : T$.

□

4.3 Consequences of the equivalence

Now that we have a way to go from PTS to PTS_e (and the other way around), we can go back to the proof of *Subject Reduction* for PTS_e.

Theorem 4.6 (*Subject Reduction for PTS_e*)
If $\Gamma \vdash_e M : T$ and $M \to_\beta N$ then $\Gamma \vdash_e M =_\beta N : T$.

Proof
By using the first part of Theorem 4.5 and Theorem 4.4, there are Γ^+, M^+ and T^+ such that $\Gamma^+ \vdash M^+ \rhd M^+ : T^+$ and $|\Gamma^+| \equiv \Gamma$, $|M^+| \equiv M$ and $|T^+| \equiv T$. Let us consider N^+ such that $|N^+| \equiv N$ and $M^+ \to_p N^+$. With such a term, and using Theorem 3.15, we can prove that $\Gamma^+ \vdash M^+ \rhd^+ N^+ : T^+$. By erasing the annotations using the last part of Theorem 3.10, we end up having $|\Gamma^+| \vdash_e |M^+| =_\beta |N^+| : |T^+|$ which is the exact result we wanted. □

The last missing piece of our development is to find the correct statement for injectivity of products in PTS_e. *Subject Reduction* for PTS_{atr} relied on the *weak* Π-*injectivity* for $\cong_\beta$ and we choose such an equality to be able to state the *Generation* lemmas for PTS_{atr}. Since PTS_{atr} is "enhanced" version of PTS_e with additional annotations, that may be the correct presentation we were looking for:

Weak PTS_e equality

$$\frac{\Gamma \vdash_e A =_\beta B : s}{\Gamma \vdash_e A =_\beta B} \qquad \frac{\Gamma \vdash_e B =_\beta A}{\Gamma \vdash_e A =_\beta B} \qquad \frac{\Gamma \vdash_e A =_\beta B \qquad \Gamma \vdash_e B =_\beta C}{\Gamma \vdash_e A =_\beta C}$$

This weaker form of equality enjoys some nice properties:

- If $\Gamma \vdash_e A =_\beta B$, then there are s and t such that $\Gamma \vdash_e A : s$ and $\Gamma \vdash_e B : t$.

- If $\Gamma \vdash_e A =_\beta B$, then $A =_\beta B$.
- This equality is compatible with conversion in PTS_e context: if $\Gamma_1 \vdash_e A =_\beta B$ and $\Gamma_1(x:A)\Gamma_2 \vdash_e M : T$, then $\Gamma_1(x:B)\Gamma_2 \vdash_e M : T$.

All those properties are directly consequences of the usual equality for PTS_e.

With this equality, we can directly state some generation lemmas for PTS_e without relying on the equivalence:

Lemma 4.7 (*Generation Lemmas for PTS$_e$*)
Those properties are much like PTS_{atr}'s one, so we will only state the one that we will really need here:

1. If $\Gamma \vdash_e \Pi x^A.B : T$ then there are s_1, s_2, s_3 such that $(s_1, s_2, s_3) \in \mathcal{R}el$, $\Gamma \vdash_e A : s_1$, $\Gamma(x:A) \vdash_e B : s_2$, and $T \equiv s_3$ or $\Gamma \vdash_e T =_\beta s_3$.
2. If $\Gamma \vdash_e \lambda x^A.M : T$ then there are s_1, s_2, s_3 and B such that $(s_1, s_2, s_3) \in \mathcal{R}el$, $\Gamma \vdash_e A : s_1$, $\Gamma(x:A) \vdash_e M : B$, $\Gamma(x:A) \vdash_e B : s_2$ and $\Gamma \vdash_e T =_\beta \Pi x^A.B$.
3. If $\Gamma \vdash_e M\ N : T$ then there are A and B such that $\Gamma \vdash_e M : \Pi x^A.B$, $\Gamma \vdash_e N : A$ and $\Gamma \vdash_e T =_\beta B[N/x]$.

Now that we have the *Generation Lemmas* and *Subject Reduction*, we can prove what we consider to be the *correct* statement for injectivity of products in PTS_e.

Corollary 4.8 (*Weak* Π*-injectivity for PTS$_e$*)
If $\Gamma \vdash_e \Pi x^A.B =_\beta \Pi x^C.D$ then $\Gamma \vdash_e A =_\beta C$ and $\Gamma(x:A) \vdash_e B =_\beta D$.

Proof
By using the properties of weak equality that we just stated, there are s_3 and s_3' such that $\Gamma \vdash \Pi x^A.B : s_3$, $\Gamma \vdash \Pi x^C.D : s_3'$, and $\Pi x^A.B =_\beta \Pi x^C.D$. By Π-*injectivity* and *Confluence* for the usual untyped β, and *Generation* for PTS_e, we get:

- $A \twoheadrightarrow_\beta U\ {}_\beta\!\!\twoheadleftarrow C$ and $B \twoheadrightarrow_\beta V\ {}_\beta\!\!\twoheadleftarrow D$
- $\Gamma \vdash A : s_1$, $\Gamma \vdash C : s_1'$, $\Gamma(x:A) \vdash B : s_2$ and $\Gamma(x:C) \vdash D : s_2'$ for s_1, s_1', s_2, s_2' such that $(s_1, s_2, s_3) \in \mathcal{R}el$ and $(s_1', s_2', s_3') \in \mathcal{R}el$.

By using *Subject Reduction for PTS$_e$*, we get that $\Gamma \vdash_e A =_\beta U : s_1$, $\Gamma \vdash_e C =_\beta U : s_1'$, $\Gamma(x:A) \vdash_e B =_\beta V : s_2$ and $\Gamma(x:C) \vdash_e D =_\beta V : s_2'$. It is now easy to glue everything together to obtain $\Gamma \vdash_e A =_\beta C$ and $\Gamma(x:A) \vdash_e B =_\beta D$. □

This proof of injectivity holds for *any* PTS_e, even the non-functional ones or the ones that do not enjoy normalization. Another test that validate we did the right choice, is that if we consider this property for granted, we can make a direct proof of *Subject Reduction for PTS$_e$* by adapting the well-known proof for PTS. However we do not have any proof of this weak injectivity that do not use *Subject Reduction*, which makes us think that the *correct framework* to deal with judgmental equality is PTS_{atr}, and not PTS_e.

5 Conclusion

Pure Type Systems are a general framework at the core of dependently typed theories. Until now, there were two main presentations, with or without typed equality judgments.

With this new result, we finally managed to prove that both presentations are describing the same theory, without having to rely on specific model-based proofs of normalization.

This result also gives us a way to correctly state and prove two main properties of PTS based on judgmental equality: *Subject Reduction* and *Weak* Π*-injectivity*. Regarding the strong version of injectivity, we provide a counterexample for the general case of PTS_e, but we know it is true in the functional case since Adams proved it (2006).

Now that we know how to deal with any kind of PTS, we will be able to focus on extending the typing system, with subtyping for example, and looking toward proving the same equivalence for the *Extended Calculus of Constructions*, or even for the *Calculus of Inductive Constructors*. On the other hand, we can also try to change the conversion rule, by adding η-expansion for example. This would provide an interesting framework to deal with normalization by evaluation, or to improve unification of proof assistants by adding techniques based on η-expansion, like pattern-unification.

Acknowledgements

We are particularly grateful to Bruno Barras for guiding us through our experiments on the problem solved in this paper. We also thank Andreas Abel, Paul-André Melliès and Stéphane Lengrand for many stimulating discussions on the topic.

References

Abel, Andreas. (2010). Towards Normalization by Evaluation for the Calculus of Constructions. *Pages 224–239 of: FLOPS*.

Abel, Andreas, Coquand, Thierry, & Dybjer, Peter. (2007). Normalization by Evaluation for Martin-Löf Type Theory with typed equality judgements. *Pages 3–12 of: 22nd IEEE Symposium on Logic in Computer Science (LICS 2007), 10-12 July 2007, Wroclaw, Poland, proceedings*. IEEE Computer Society Press.

Adams, Robin. (2006). Pure Type Tystems with Judgemental Equality. *J. Funct. Program.*, **16**(2), 219–246.

Barendregt, H., Abramsky, S., Gabbay, D. M., Maibaum, T. S. E., & Barendregt, H. P. (1992). Lambda Calculi with Types. *Pages 117–309 of: Handbook of Logic in Computer Science*. Oxford University Press.

Berardi, Stefano. (1988). *Type dependence and constructive mathematics*. Ph.D. thesis, Mathematical Institute Torino.

Coq Development Team. *The Coq proof assistant reference manual*. http://coq.inria.fr/refman/.

de Bruijn, N.G. (1972). Lambda-calculus notation with nameless dummies: a tool for automatic formula manipulation with application to the Church-Rosser theorem. *Indag. math.*, **34**(5), 381–392.

Geuvers, Herman. (1993). *Logics and type systems*. Ph.D. thesis, Katholieke Universiteit Nijmegen.

Geuvers, Herman, & Werner, Benjamin. (1994). On the Church-Rosser property for expressive type systems and its consequences for their metatheoretic study. *Pages 320–329 of: LICS*.

Goguen, Healfdene. (1994). *A typed operational semantics for type theory*. Ph.D. thesis, University of Edinburgh.

Luo, Z. (1989). ECC: an extended calculus of constructions. *Pages 385–395 of: Proceedings of the fourth annual symposium on logic in computer science*. Piscataway, NJ, USA: IEEE Press.

Martin-Löf, Per. (1984). *Intuitionistic type theory*. Bibliopolis.

Melliès, P.-A., & Werner, B. (1997). A generic normalization proof for Pure Type Systems. Paulin-Mohring, C., & Gimenez, E. (eds), *TYPES'96*. LNCS, Springer-Verlag.

Nordstrom, Bengt, Petersson, Kent, & Smith, Jan M. (1990). *Programming in Martin-Löf's Type Theory: An introduction*. Oxford University Press, USA.

Norell, Ulf. 2007 (September). *Towards a practical programming language based on dependent type theory*. Ph.D. thesis, Department of Computer Science and Engineering, Chalmers University of Technology, SE-412 96 Göteborg, Sweden.

Siles, Vincent. *Formalization of equivalence between PTS and PTSe*. `http://www.lix.polytechnique.fr/~vsiles/coq/PTSATR.html`.

Siles, Vincent, & Herbelin, Hugo. (2010). Equality is typable in semi-full Pure Type Systems. *Proceedings, 25th annual IEEE symposium on Login in Computer Science (LICS '10), Edinburgh, UK, 11-14 July 2010*. IEEE Compuer Society Press.

Streicher, Thomas. (1991). *Semantics of type theory: correctness, completeness, and independence results*. Cambridge, MA, USA: Birkhauser Boston Inc.

van Benthem Jutting, L. S., McKinna, James, & Pollack, Robert. (1993). Checking Algorithms for Pure Type Systems. *Pages 19–61 of: TYPES*.

Werner, Benjamin. (1994). *Une théorie des Constructions Inductives*. Ph.D. thesis, Université Paris 7.

Werner, Benjamin, & Lee, Gyesik. (2010). *A proof-irrelevant model of CIC with predicate induction and judgemental equality*. Submitted to LMCS.