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 towards 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 PTSs in previous work. In this paper, we finally give a positive answer to the general question, 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 conversion" rule of the form:

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

Untyped conversion is the equality conventionally used to define e.g. the *Calculus of Inductive Constructions*. The equality is a black box that knows nothing about the typing validity of the terms it deals with: each conversion step is not checked to be well-typed and it is only a posteriori that we know that for two convertible well-typed terms, there is a path exclusively made of well-typed terms that connects them (see Corollary 2.9). A second approach embeds a notion of equality directly in the type system. So there are two

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

kinds of typing judgments: one to type terms, and one to type equalities. With this kind of approach, we enforce 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) from which *Agda 2* is derived, or *UTT* (Goguen, 1994).

Surprisingly, showing the equivalence between those two definitions is difficult. Translating a judgmental equality into an untyped one is simple, but the reverse translation is significantly more difficult. Geuvers (1993) 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. Because they carry more typing information, the systems based on judgmental equality are convenient for building models (Goguen, 1994; Abel *et al.*, 2007; Abel, 2010; Werner & Lee, 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 *et al.*, 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.

Among type systems, the class of Pure Type Systems (or PTSs) that Berardi (1990) and Terlouw (1989) independently introduced as a generalization of Barendregt's λ-cube (Barendregt, 1991) is a framework based on untyped conversion 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 or concepts (inductive types, intersection types, subtyping, ...).

A few years ago, Adams (2006) showed that building models was not necessary to connect PTSs and their counterpart with judgmental equality (also knows as *semantical* PTS (Geuvers, 1993), or PTS_e): he proved by purely *syntactical* means[3] that every functional Pure Type System is equivalent to its variant with judgmental equality. The authors also made a new step toward an extension of the result to all PTSs by reusing Adams' technique 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 combines the idea of a typed equality 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 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 PTSs: *Pure Type System based on Annotated Typed Reduction* (PTS_{atr}).

PTS_{atr} can be seen as an operational presentation of PTS_e with enough typing information embedded in terms so that the main meta-theoretical properties of PTSs hold, starting with Π-injectivity. That Π-injectivity holds is not obvious and a by-product of our approach is that only a non-uniformly typed form of Π-injectivity holds. This weak Π-injectivity is however enough to get *Church-Rosser* and *Subject Reduction* and this is shown in Section 3. The equivalence comes then from the ability to annotate any derivation in PTSs or PTS_e so that it holds in PTS_{atr}. We show how do to that for PTSs in Section 4.

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 in (Siles, 2010).

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* and *Pure Type System with Judgmental Equality*, its "typed" counterpart. We also recall the main properties of these

[3] Formalizable in primitive recursive arithmetic.

systems, and the main issues that one faces 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 is used to type functions, and is usually denoted $A \to B$ 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 PTSs. 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 shorthand 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: $M[N/x]$ 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 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 a well-established literature on PTSs and we only recall the main ideas of those systems. The reader interested in more details is invited to look for instance at (Geuvers & Nederhof, 1991; Barendregt, 1992; Geuvers, 1993).

The generic nature of PTSs arise in the typing rules for sorts and Π-types. The set of *axioms* $\mathscr{A} \subset (Sorts \times Sorts)$ is used to type sorts: $(s,t) \in \mathscr{A}$ means that the sort s can be typed by the sort t. The set of rules $\mathscr{R} \subset (Sorts \times Sorts \times Sorts)$ is used to check the well-formedness of Π-types.

In this paper, we describe a variant of PTSs (which is known to be equivalent to their usual description, see (Pollack, 1994) or the proof provided in the Coq formalization) which uses a notion of "well-formed contexts". The typing rules for PTSs 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.

In this paper, we refer to some subclasses of PTSs:

Functional, Full and semi-Full PTS

- A PTS is functional if:
  1. for all s,t,t', if $(s,t) \in \mathscr{A}$ and $(s,t') \in \mathscr{A}$ then $t \equiv t'$.
  2. for all s,t,u,u', if $(s,t,u) \in \mathscr{R}$ and $(s,t,u') \in \mathscr{R}$ then $u \equiv u'$.
- A PTS is semi-full[4] if $(s,t,u) \in \mathscr{R}$ implies that for all t', there is u' such that $(s,t',u') \in \mathscr{R}$.
- A PTS is full if for any s,t, there is u such that $(s,t,u) \in \mathscr{R}$.
  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 PTSs. They are the basic meta-theory that we need to prove the interesting theorems.

[4] The notion of semi-full is due to Pollack, see (van Benthem Jutting *et al.*, 1993).

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

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

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

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

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

Fig. 1. Typing Rules for PTS

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

Theorem 2.5 (*Generation*)

1. If $\Gamma \vdash s : T$ then there is t such that $(s,t) \in \mathscr{A}$ 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 $\Gamma \vdash A : s_1$, $\Gamma(x:A) \vdash B : s_2$, $(s_1,s_2,s_3) \in \mathscr{R}$ 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 $\Gamma \vdash A : s_1$, $\Gamma(x:A) \vdash B : s_2$, $\Gamma(x:A) \vdash M : B$, $(s_1,s_2,s_3) \in \mathscr{R}$ 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 PTSs, we need to distinguish between the two conclusions: nothing ensures that all sorts are well-typed.

The notion of β-conversion can easily be 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 PTSs, 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, 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*)
If $\Gamma \vdash M : A$, $\Gamma \vdash B : s$ and $A =_\beta B$, then there is a sequence $(C_1, s_1), \ldots, (C_p, s_p)$ such that $A \equiv C_1$, $B \equiv C_p$, $\Gamma \vdash C_i : s_i$ and $C_i \to_\beta C_{i+1}$ or $C_{i+1} \to_\beta C_i$.

Proof
Let us suppose we have $\Gamma \vdash M : A$, $\Gamma \vdash B : s$ and $A =_\beta B$. By *Confluence*, there is C such that $A \twoheadrightarrow_\beta C \;{}_\beta\!\!\twoheadleftarrow B$. By *Type Correctness*, there is t such that $\Gamma \vdash A : t$, or $A \equiv t$:

1. In the first case, by *Subject Reduction*, we know that any term that appears in the reduction $A \to_\beta A_1 \to_\beta \ldots \to_\beta A_k \to_\beta C$ is typed by t, and any term that appears in the reduction $B \to_\beta B_1 \to_\beta \ldots \to_\beta B_l \to_\beta C$ is typed by s. So we can take the sequence $(A,t), (A_1,t), \ldots, (A_k,t), (C,t), (B_l,s), \ldots, (B_1,s), (B,s)$.
2. In the second case, $B =_\beta t$ and by *Confluence*, $B \to_\beta B_1 \to_\beta \ldots \to_\beta B_p \to_\beta t$. *Subject Reduction* implies that $\Gamma \vdash t : s$. So this time, we can choose the sequence $(A,s), (B_p,s), \ldots, (B_1,s), (B,s)$.

□

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 required to be well-typed. With those judgments, we no longer need to rely on *Confluence* and *Subject Reduction* to ensure that CONV is sound. The typing rules for PTS_e are given in Fig. 2. The first thing we can prove (by direct induction) about this system is that equality enjoys reflexivity:

Lemma 2.10 (*Equality Reflexivity in PTS_e*)
If $\Gamma \vdash_e M : T$ then $\Gamma \vdash_e M =_\beta M : T$.

We can prove by the same arguments that some properties of PTSs also hold for PTS_e, namely *Weakening*, *Substitution* (with similar statements) and *Context Conversion*:

Lemma 2.11 (*Weakening in PTS_e*)
1. If $\Gamma_1\Gamma_2 \vdash_e M : B$, $\Gamma_1 \vdash_e A : s$ and $x \notin \mathit{Dom}(\Gamma_1\Gamma_2)$ then $\Gamma_1(x : A)\Gamma_2 \vdash_e M : B$.
2. If $\Gamma_1\Gamma_2 \vdash_e M =_\beta N : B$, $\Gamma_1 \vdash_e A : s$ and $x \notin \mathit{Dom}(\Gamma_1\Gamma_2)$ then $\Gamma_1(x : A)\Gamma_2 \vdash_e M =_\beta N : B$.
3. If $\Gamma_1\Gamma_2\ _{wf}$, $\Gamma_1 \vdash_e A : s$ and $x \notin \mathit{Dom}(\Gamma_1\Gamma_2)$ then $\Gamma_1(x : A)\Gamma_2\ _{wf}$.

Lemma 2.12 (*Substitution in PTS_e*)
1. If $\Gamma_1(x : A)\Gamma_2 \vdash_e M : B$ and $\Gamma_1 \vdash_e P : A$ then $\Gamma_1\Gamma_2[P/x] \vdash_e M[P/x] : B[P/x]$.
2. If $\Gamma_1(x : A)\Gamma_2 \vdash_e M =_\beta N : B$ and $\Gamma_1 \vdash_e P : A$ then $\Gamma_1\Gamma_2[P/x] \vdash_e M[P/x] =_\beta N[P/x] : B[P/x]$.
3. If $\Gamma_1(x : A)\Gamma_2\ _{wf}$ and $\Gamma_1 \vdash_e P : A$ then $\Gamma_1\Gamma_2[P/x]_{wf}$.

Lemma 2.13 (*Context Conversion in PTS_e*)
- If $\Gamma_1(x : A)\Gamma_2 \vdash_e M : T$ and $\Gamma_1 \vdash_e A =_\beta B : s$ then $\Gamma_1(x : B)\Gamma_2 \vdash_e M : T$.
- If $\Gamma_1(x : A)\Gamma_2 \vdash_e M =_\beta N : T$ and $\Gamma_1 \vdash_e A =_\beta B : s$ then $\Gamma_1(x : B)\Gamma_2 \vdash_e M =_\beta N : T$.
- If $\Gamma_1(x : A)\Gamma_2\ _{wf}$ and $\Gamma_1 \vdash_e A =_\beta B : s$ then $\Gamma_1(x : B)\Gamma_2\ _{wf}$.

Later on, we will need another variant of the substitution lemma, to prove that we can safely perform parallel substitution in PTS_e:

Lemma 2.14 (*Parralel Substitution in PTS_e*)
1. If $\Gamma_1(x : A)\Gamma_2 \vdash_e M : B$ and $\Gamma_1 \vdash_e P =_\beta P' : A$ then $\Gamma_1\Gamma_2[P/x] \vdash_e M[P/x] =_\beta M[P'/x] : B[P/x]$.
2. If $\Gamma_1(x : A)\Gamma_2 \vdash_e M =_\beta N : B$ and $\Gamma_1 \vdash_e P =_\beta P' : A$ then $\Gamma_1\Gamma_2[P/x] \vdash_e M[P/x] =_\beta N[P'/x] : B[P/x]$.

Proof
The proof of the first point is straightforward by induction on the shape of the typing judgment $\Gamma_1(x : A)\Gamma_2 \vdash_e M : B$, using the previous *Substitution* lemma. The proof of the latter is a trivial combination of TRANS, *Substitution* and the first point. □

We can add to the list the following reflexivity properties (also known as *Equation Validity*) which need to be proved along with *Type Correctness*:

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

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

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

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

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

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

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

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

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

Fig. 2. Typing Rules for PTS_e

Lemma 2.15 (*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 $s \in Sorts$ such that $T \equiv s$ or $\Gamma \vdash_e T : s$.
- If $\Gamma \vdash_e M =_\beta N : A$, then $\Gamma \vdash_e M : A$.
- If $\Gamma \vdash_e M =_\beta N : A$, then $\Gamma \vdash_e N : A$.

Proof

We need to prove these three propositions simultaneously for three main reasons:

1. to prove *Type Correctness*, we need the *Right-Hand reflexivity* for the CONV rule.
2. to prove both reflexivity statements, we need *Type Correctness* for the APP-EQ rule.
3. because of the SYM rule, we need to prove both reflexivity statements 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 additional work. The proof is also done by induction, but *Context Conversion* is used in the rules involving λ-abstractions and Π-types, and the *Substitution* lemmas are used to type the right part of BETA. The proof of *Type Correctness* also follows directly from the mutual induction hypothesis. □

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.

Our final goal is to prove the equivalence between PTS and PTS_e:

Theorem 2.16 (*Equivalence betwwen PTS and PTS_e*)
- $\Gamma \vdash M : T$ iff $\Gamma \vdash_e M : T$
- $\Gamma \vdash M : T$, $\Gamma \vdash N : T$, and $M =_\beta N$ iff $\Gamma \vdash_e M =_\beta N : T$

With the few results we listed for PTS_e, we can already prove half of this equivalence:

Theorem 2.17 (*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 main idea of the proof is to remove the typing information from the typed equalities. The proof is straightforward by mutual induction on the typing judgments of PTS_e. *Context Conversion* (in PTSs) is also requiered for the second conclusion. □

2.3 Subject Reduction and Equivalence

We previously saw that *Subject Reduction* and Π-*injectivity* were two important properties of PTSs: *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 required to be well-typed. If it is the case, we would be able to prove that both presentation are in fact two different ways to describe the same theory.

Theorem 2.18 (*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 PTSs, 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. Here are two examples of injectivity:

- We can completely getting rid of the types (as we did for PTSs):
  If $\Gamma \vdash_e \Pi x^A B =_\beta \Pi x^C.D : u$, then $A =_\beta C$ and $B =_\beta D$.
- We can also try to keep as much typing information as we can:
  If $\Gamma \vdash_e \Pi x^A.B =_\beta \Pi x^C.D : u$ then $\Gamma \vdash_e A =_\beta C : s$ and $\Gamma(x:A) \vdash_e B =_\beta D : t$ for some $s,t \in \mathit{Sorts}$ such that $(s,t,u) \in \mathscr{R}$.

With the first solution, we lack too much type information to build the typed equality needed by *Subject Reduction*. The second one is used by Adams to prove the equivalence in the functional case. However, this statement is wrong in the general case (this proof can also be found in the Coq formalization):

Lemma 2.19 (*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 $s,t \in \mathit{Sorts}$ such that $(s,t,u) \in \mathscr{R}$.

Proof
We are going to build a counterexample by selecting the right sets for *Sorts*, $\mathscr{A}$ and $\mathscr{R}$. Let us assume that previous statement of strong injectivity holds for all PTS$_e$, including the following one:

- $\mathit{Sorts} \equiv \{u, v, v', w, w'\}$
- $\mathscr{A} \equiv \{(u,v), (u,v'), (v,w), (v',w')\}$
- $\mathscr{R} \equiv \{(w,w,w), (w',w',w'), (v,v,u), (v',v',u)\}$

Let us define two terms $D_1 \equiv (\lambda x^v.u)\ u$ and $D_2 \equiv (\lambda x^{v'}.u)\ u$.

1. $\emptyset \vdash_e D_1 : v$ and if $\emptyset \vdash_e D_1 : T$ then $T =_\beta v$.
   This is a consequence of our choices for the sets $\mathscr{A}$ and $\mathscr{R}$: to type the abstraction $\lambda x^v.u$, we need to find a rule $(a,b,c) \in \mathscr{R}$ and a type A such that $\emptyset \vdash_e v : a$, $(x:v) \vdash_e u : A$ and $(x:v) \vdash_e A : b$. The first typing judgment implies that $a \equiv w$, and the only rule involving w is (w,w,w), so $b \equiv c \equiv w$. This also implies that the only choice for A is v. Therefore, the abstraction has only one type, $v \to v$, and T has to be equal to $v[u/x] \equiv v$.
2. For the same reason, $\emptyset \vdash_e D_2 : v'$ and if $\emptyset \vdash_e D_2 : T$ then $T =_\beta v'$.
3. with both results and the fact that $\emptyset \vdash_e u : v$ and $\emptyset \vdash_e u : v'$, we can prove $\emptyset \vdash_e D_1 =_\beta u : v$ and $\emptyset \vdash_e D_2 =_\beta u : v'$.
4. The correct choice of rules in $\mathscr{R}$ leads to $\emptyset \vdash_e \Pi x^{D_1}.u =_\beta \Pi x^u.u : u$ and $\emptyset \vdash_e \Pi x^u.u =_\beta \Pi x^{D_2}.u : u$, so by transitivity: $\emptyset \vdash_e \Pi x^{D_1}.u =_\beta \Pi x^{D_2}.u : u$.
5. Since we supposed strong-injectivity, either $\emptyset \vdash_e D_1 =_\beta D_2 : v$ or $\emptyset \vdash_e D_1 =_\beta D_2 : v'$.
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 prove *Subject Reduction*, we need a weaker form of Π-injectivity. In the next sections, we give the description of a correct injectivity statement, but we are not able to prove it before proving *Subject Reduction*. This is the reason why we postpone this discussion to

Section 4.

To prove the full equivalence between untyped conversion and judgmental equality, we define an auxiliary type presentation PTS_{atr}, with judgments of the form $\Gamma \vdash M \rhd N : A$. The intended meaning is that M of type A can do a parallel reduction step to N. PTS_{atr} also has more informative terms so we can directly prove properties like *Confluence*, *Weak* Π-*injectivity* and *Subject-Reduction*. There is an erasure function $|\ |$ from the annotated terms of PTS_{atr} to original PTS and PTS_e terms. The outline of the equivalence is the following:

1. If $\Gamma \vdash M \rhd N : A$ then $|\Gamma| \vdash |M| : |A|$ and $\Gamma \vdash |N| : |A|$,
2. If $\Gamma \vdash M \rhd N : A$, then $|\Gamma| \vdash_e |M| =_\beta |N| : |A|$,
3. If $\Gamma \vdash M : A$, then there are Γ^+, M^+ and A^+ such that $\Gamma^+ \vdash M^+ \rhd M^+ : A^+$ and $|\Gamma^+| \equiv \Gamma$, $|M^+| \equiv M$ and $|A^+| \equiv A$.

The properties combined show that a PTS can be embedded into a PTS_e, using PTS_{atr} as an intermediate step.

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 PTSs 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 PTSs and PTS_e has been given by Adams (2006) for the subclass of *functional* PTSs, a result that has been later extended to the subclasses of *semi-full* and *full* PTSs 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 PTSs and PTS_e.
2. It has to satisfy 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.

Since we are dealing with a typed equality, we need to build a typed version of *Church-Rosser*. 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. In order to prove the *Church-Rosser* property, Adams decided to annotate applications by their co-domain, and to restrict to functional PTSs 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, but this is not enough for the general framework.

To extend Adams method to the class of all PTSs and PTS_e, we add 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 PTSs 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. In the following, we give a detailed description of the systems, its properties, and of 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 PTSs are defined in the same way for PTS_{atr}, with their natural adaptation to the annotated applications. To avoid confusion between the reductions, we write $\rightarrow_p$ for untyped parallel reduction in PTS_{atr} (we allow reduction in the annotations) and $\twoheadrightarrow$ for its transitive closure (since PTS_{atr} is a parallel system, using a one-step parallel reduction is 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 inductively removing the additional typing information within the applications.

The typing rules of PTS_{atr} are presented in Fig. 3. As a shortcut, we use the notation $\Gamma \vdash M \rhd N : A,B$ for "$\Gamma \vdash M \rhd N : A$ and $\Gamma \vdash M \rhd N : B$".

The $\rhd^+$ (resp. $\cong_\beta$) relation can be read as the transitive (resp. transitive-symmetric) closure of the $\rhd$ relation. 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 need this to be able to state the *Generation Lemmas* correctly, since we do not have the *Uniqueness of Types* in the general case. To avoid confusion in further development, here is a reminder of the several variants of β-equality we are dealing with:

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 meaning of the BETA rule is to ensure that there is a conversion path from the annotation A of the λ-abstraction, 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). As Adams pointed out for TPOSR, having A instead of A' would break the linearity of the left-hand side of the rule:

$$\frac{}{\emptyset_{wf}}\ \textsc{empty} \qquad \frac{\Gamma \vdash A \rhd B : s \qquad x \notin Dom(\Gamma)}{\Gamma(x:A)_{wf}}\ \textsc{extend}$$

$$\frac{\Gamma_{wf} \qquad (s,t) \in \mathscr{A}}{\Gamma \vdash s \rhd s : t}\ \textsc{sort} \qquad \frac{\Gamma_{wf} \qquad \Gamma(x) = A}{\Gamma \vdash x \rhd x : A}\ \textsc{var}$$

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

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

$$\frac{\Gamma \vdash A \rhd A : s_1 \qquad \Gamma \vdash A' \rhd A' : s_1 \qquad \Gamma \vdash A_0 \rhd^+ A : s_1 \qquad \Gamma \vdash A_0 \rhd^+ A' : s_1 \qquad (s_1,s_2,s_3) \in \mathscr{R} \qquad \Gamma(x:A) \vdash B \rhd B : s_2 \qquad \Gamma(x:A) \vdash M \rhd M' : B \qquad \Gamma \vdash N \rhd N' : A}{\Gamma \vdash (\lambda x^A.M)_{\Pi x:A'.B} N \rhd M'[N'/x] : B[N/x]}\ \textsc{beta}$$

$$\frac{\Gamma \vdash M \rhd N : A \qquad \Gamma \vdash A \rhd B : s}{\Gamma \vdash M \rhd N : B}\ \textsc{red} \qquad \frac{\Gamma \vdash M \rhd N : A \qquad \Gamma \vdash B \rhd A : s}{\Gamma \vdash M \rhd N : B}\ \textsc{exp}$$

$$\frac{\Gamma \vdash M \rhd N : A}{\Gamma \vdash M \rhd^+ N : A}\ \textsc{reds-intro} \qquad \frac{\Gamma \vdash M \rhd^+ N : A \qquad \Gamma \vdash N \rhd^+ P : A}{\Gamma \vdash M \rhd^+ P : A}\ \textsc{reds-trans}$$

$$\frac{\Gamma \vdash A \rhd B : s}{\Gamma \vdash A \cong_\beta B}\ \textsc{eq-intro} \qquad \frac{\Gamma \vdash B \rhd A : s}{\Gamma \vdash A \cong_\beta B}\ \textsc{eq-intro2} \qquad \frac{\Gamma \vdash A \cong_\beta B \qquad \Gamma \vdash B \cong_\beta C}{\Gamma \vdash A \cong_\beta C}\ \textsc{trans}$$

Fig. 3. Typing Rules and Type Equality for PTS_{atr}

a β-redex would only be able to reduce if both annotations are syntactically equal, which may not be the case (especially during the proof of the *Church-Rosser* property). To get over this limitation, we require that both annotations must be convertible, and the path between them has to be typed by the same sort.

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. 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 would not be used elsewhere.

We do not directly have a symmetry statement for $\cong_\beta$ equality in order to have more control over the equality, but this rule is straightforward to prove by induction:

Lemma 3.1 (*Symmetry for* $\cong_\beta$)
If $\Gamma \vdash A \cong_\beta B$ then $\Gamma \vdash B \cong_\beta A$.

3.2 General properties of PTS$_{atr}$

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

Lemma 3.2 (*Weakening*)
1. If $\Gamma_1\Gamma_2 \vdash M \rhd N : B$, $\Gamma_1 \vdash A \rhd A' : s$ and $x \notin Dom(\Gamma_1\Gamma_2)$ then $\Gamma_1(x:A)\Gamma_2 \vdash M \rhd N : B$.
2. If $\Gamma_1\Gamma_2 \vdash M \rhd^+ N : B$, $\Gamma_1 \vdash A \rhd A' : s$ and $x \notin Dom(\Gamma_1\Gamma_2)$ then $\Gamma_1(x:A)\Gamma_2 \vdash M \rhd^+ N : B$.
3. If $\Gamma_1\Gamma_2\ _{wf}$, $\Gamma_1 \vdash A \rhd A' : s$ and $x \notin Dom(\Gamma_1\Gamma_2)$ then $\Gamma_1(x:A)\Gamma_2\ _{wf}$.

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 $\Gamma' \vdash A \cong_\beta B$.

The following lemmas are still proved by mutual induction, but they have to be proved in this order since they also rely on the lemma just before them.

Lemma 3.4 (*Left-Hand Typability*)
If $\Gamma \vdash M \rhd N : A$ or $\Gamma \vdash M \rhd^+ N : A$, then $\Gamma \vdash M \rhd M : A$.

Lemma 3.5 (*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 $\Gamma_1 \vdash P \rhd P' : A$ then $\Gamma_1\Gamma_2[P/x]_{wf}$.

Lemma 3.6 (*Right-Hand Typability*)
1. If $\Gamma \vdash M \rhd N : A$ or $\Gamma \vdash M \rhd^+ N : A$, then $\Gamma \vdash N \rhd N : A$.
2. 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 PTSs. By adding both annotations, we do not have to "guess" the domain and co-domain of an application anymore.

Lemma 3.7 (*Generation*)

1. If $\Gamma \vdash s \rhd N : T$ then $N \equiv s$ and there is t such that $(s,t) \in \mathscr{A}$ 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 $N \equiv \Pi x^{A'}.B'$, $(s_1,s_2,s_3) \in \mathscr{R}$, $\Gamma \vdash A \rhd A' : s_1$, $\Gamma(x:A) \vdash B \rhd B' : s_2$ 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, s_1, s_2, s_3 such that $N \equiv \lambda x^{A'}.M'$, $(s_1,s_2,s_3) \in \mathscr{R}$, $\Gamma \vdash A \rhd A' : s_1$, $\Gamma(x:A) \vdash B \rhd B : s_2$, $\Gamma(x:A) \vdash M \rhd M' : B$ 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 $(s_1,s_2,s_3) \in \mathscr{R}$, $\Gamma \vdash A \rhd A' : s_1$, $\Gamma(x:A) \vdash B \rhd B' : t_2$, $\Gamma \vdash Q \rhd Q' : A$, $\Gamma \vdash T \cong_\beta B[Q/x]$ 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'.

Proof
As for PTSs, the proof is done by induction on the shape of the typing judgment. □

One of the key-points 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 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. To prove these properties about multi-step congruence, Adams used the *Type Uniqueness* property thanks to its functional setting. To prove those multi-step congruence results for PTS_{atr}, we need to find something new. A particular example of what we need arise in the multi-step congruence case of application, where we need to check that terms are typed by the triple of sorts in $\mathscr{R}$. For example, we know that $\Gamma \vdash A \rhd A : s$ 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.

The following theorem is a sufficient tool to achieve this task:

Theorem 3.8 (*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 on the first judgment and *Generation* on the second one, there are no difficult cases since we have the co-domain annotations on the applications. The second part of the conclusion is proved by symmetry. □

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 3.9 (*Exchange of Types in multi-step reduction*)
If $\Gamma \vdash M \rhd^+ N : A$ and $\Gamma \vdash M \rhd M : B$, 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 3.10 (*Multi-step Congruences and Generations*)

- Congruences:
  - If $\Gamma \vdash A \rhd^+ A' : s_1$, $\Gamma(x:A) \vdash B \rhd^+ B' : s_2$ and $(s_1,s_2,s_3) \in \mathscr{R}$, then $\Gamma \vdash \Pi x^A.B \rhd^+ \Pi x^{A'}.B' : s_3$.
  - If $\Gamma \vdash A \rhd^+ A' : s_1$, $\Gamma(x:A) \vdash M \rhd^+ M' : B$, $\Gamma(x:A) \vdash B \rhd B : s_2$ and $(s_1,s_2,s_3) \in \mathscr{R}$, 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 $(s_1,s_2,s_3) \in \mathscr{R}$, $N \equiv \Pi x^{A'}.B'$, $\Gamma \vdash A \rhd^+ A' : s_1$, $\Gamma(x:A) \vdash B \rhd^+ B' : s_2$ 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 $(s_1,s_2,s_3) \in \mathscr{R}$, $N \equiv \lambda x^{A'}.M'$, $\Gamma \vdash A \rhd^+ A' : s_1$, $\Gamma(x:A) \vdash M \rhd^+ M' : B$, $\Gamma(x:A) \vdash B \rhd B : s_2$ and $\Gamma \vdash T \cong_\beta \Pi x^A.B$.
  - If $\Gamma \vdash s \rhd^+ N : T$, then there is t such that $N \equiv s$, $(s,t) \in \mathscr{A}$, and $\Gamma \vdash T \cong_\beta t$ or $T \equiv t$.

Proof
These proofs are done in the same way as their PTSs' counterpart, by induction on the length of the $\rhd^+$ reduction, along with *Exchange of Types*. □

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

Lemma 3.11 (*Type Correctness*)
If $\Gamma \vdash M \rhd N : A$, then there is $s \in \mathit{Sorts}$ such as either: $A \equiv s$ or $\Gamma \vdash A \rhd A : s$.

Proof
The proof is the same as for PTSs, by induction on the typing judgment. □

Theorem 3.12 (*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

As we did for the translation from PTS_e into PTSs, we want to strip a PTS_{atr} judgment from its annotation in the application, to get a valid judgment in PTSs. The first point is a consequence of the second and Theorem 2.17. The latter follows the same pattern as the proof of Theorem 2.17, by induction on the typing judgment, with some use of *Context Conversion* of PTS_e for LAM, PI, BETA and APP, and *Parallel Substitution* for BETA.

Since PTS_{atr} is a parallel system, and PTS_e is not, it is mandatory for the *Parallel Substitution* lemma to be provable in the latter. □

Corollary 3.13 (*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

Using Theorem 3.12, we can prove that $\Gamma \vdash M \cong_\beta N$ implies $|M| =_\beta |N|$ (by induction on the length of the conversion path). Let us consider a judgment of the form $\Gamma \vdash \Pi x^A.B \cong_\beta s$. Then by translating it into a PTS equality, we end up having $\Pi x^{|A|}.|B| =_\beta s$. Since β-conversion is confluent (Lemma 2.1), there is a term T such that $\Pi x^{|A|}.|B| \twoheadrightarrow_\beta T$ and $s \twoheadrightarrow_\beta T$. However, this implies that T has to be a Π-type *and at the same time* a sort, which is impossible. □

At this point we need to recall what we said about the order we used to prove things in PTSs. 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 PTSs 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 3.14 (*Diamond Property*)

If $\Gamma \vdash M \rhd N : A$ and $\Gamma \vdash M \rhd P : B$, then there is Q such that

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

It is to prove the *Diamond Property* property that the annotation is important. Indeed, to make the proof goes through, we need to satisfy the following constraints:

1. because the resulting type of an application in the APP and BETA rules is only an instance $B[N/x]$ of the original co-domain B present in the premises of the rule, some information needs to be kept to match both co-domains involved in the APP/APP, BETA/APP and APP/BETA cases;
2. because reduction steps can occur in the occurrence of A in both $\lambda x^A.M$ and $\Pi x^A.B$, the induction hypotheses over the domain of types do not always match the context of the hypothesis we actually have.

Adams solved the first problem by adding the co-domain as an annotation of application and he solved the second problem by requiring *Uniqueness of Typing* which comes from the functionality. In (Siles & Herbelin, 2010), we reused Adams' idea for solving the first problem and used instead a property on the shape of types (which is called *Typing Lemma* in (van Benthem Jutting, 1993)) to solve the second problem. To address the full generality of PTSs, our solution to the second problem is to add the domain as an extra annotation of application.

Adding the domain as an annotation raises new problems in the design of the BETA rule (Figure 3). We can not require A and A' to be syntactically the same in the rule BETA because A and A' are liable to be reduced in different directions and their syntactic equivalence would not be preserved as an invariant. We can not take them unrelated neither, nor can we take them $\cong_\beta$-convertible. Indeed, we need to enforce that each conversion step stays in the same sort, much like the equality judgments for PTS_e, and for that purpose, it happens that ensuring the existence of a common ancestor A_0 for the reduction is a sufficient condition.

Proof

The proof is done by induction on the first judgment and *Generation* on the second one. We only describe the BETA/APP. The APP/APP and APP/BETA are done in a similar way, and all other cases are straightforward.

The two judgments are

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

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

where[5]

$$\begin{array}{l|l} \Gamma \vdash A_0 \rhd^+ A : s_1 & \Gamma \vdash A \rhd C : t_1 \\ \Gamma \vdash A_0 \rhd^+ : A' : s_1 & \Gamma(x:A') \vdash B \rhd B'' : t_2 \\ \Gamma(x:A) \vdash B \rhd B' : s_2 & \Gamma(x:A) \vdash M \rhd M'' : D \\ \Gamma(x:A) \vdash M \rhd M' : B & \Gamma \vdash N \rhd N'' : A' \\ \Gamma \vdash N \rhd N' : A & \Gamma \vdash \Pi x^A.D \cong_\beta \Pi x^{A'}.B \\ & \Gamma \vdash A' \rhd C' : u_1 \end{array}$$

By induction (and *Context Conversion* for B), we can close the diamonds for M, N and B: there are M_0, N_0 and B_0 such that

- $\Gamma(x:A) \vdash M' \rhd M_0 : B, D$ and $\Gamma(x:A) \vdash M'' \rhd M_0 : B, D$

[5] To keep the proof readable, we do not keep track of all the $\mathscr{R}$ involved.

- $\Gamma \vdash N' \rhd N_0 : A, A'$ and $\Gamma \vdash N'' \rhd N_0 : A, A'$
- $\Gamma(x : A) \vdash B' \rhd B_0 : s_2, t_2$ and $\Gamma(x : A) \vdash B'' \rhd B_0 : s_2, t_2$

Our candidate to close the diamond is $M_0[N_0/x]$. To conclude, we need to prove that (1) $\Gamma \vdash M'[N'/x] \rhd M_0[N_0/x] : B[N/x]$ and (2) $\Gamma \vdash (\lambda x^C.M'')_{\Pi x:C'.B''}\ N'' \rhd M_0[N_0/x] : B[N/x]$.

Thanks to the *Substitution* lemma, $\Gamma \vdash B[N/x] \rhd B[N_0/x] : s_2, t_2$, so $\Gamma \vdash B[N/x] \cong_\beta B[N_0/x]$. So we can close (1) by converting $B[N_0/x]$ into $B[N/x]$ and applying the *Substitution* lemma once more.

To prove (2), we perform the same replacement, then we need to apply the BETA rule, and so we need to find a well-typed path from C to C'. Fortunately, we already have one, through A, A_0 and A'. However, we have a mix of s_1, t_1 and u_1 while we need the exact same sort along the path. This is where Theorem 3.8 is useful: we can rewrite the judgments into $\Gamma \vdash A \rhd C : s_1$ and $\Gamma \vdash A' \rhd C' : s_1$, which leads to $\Gamma \vdash A_0 \rhd^+ C : s_1$ and $\Gamma \vdash A_0 \rhd^+ C' : s_1$. We can now correctly apply the BETA rule. □

As a direct consequence (by induction of the structure of the $\rhd^+$ reductions) of the *Diamond Property*, we finally are able to prove the *Church-Rosser* property.

Theorem 3.15 (*Church-Rosser Property*)
If $\Gamma \vdash M \rhd^+ N : A$ and $\Gamma \vdash M \rhd^+ P : B$, then $\Gamma \vdash N \rhd^+ Q : A$ and $\Gamma \vdash P \rhd^+ A : B$.

3.4 Consequences of the Church-Rosser *property*

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

Lemma 3.16 (*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 3.17 (*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 $\Gamma(x : A) \vdash B \cong_\beta D$.

Proof
The two previous lemmas are proved in the exact same way as their PTS version:

- *Confluence* is proved by induction on the structure of the conversion path.
- *Weak* Π*-injectivity* is a direct consequence of *Confluence* and the fact that a Π-type can only reduce itself to another Π-type.

□

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 statement of Π*-injectivity* for $\cong_\beta$ along with the *Exchange of Types* property are powerful enough to prove *Subject Reduction*.

Theorem 3.18 (*Subject Reduction*)
If $\Gamma \vdash M \rhd M : A$ and $M \to_p N$ then $\Gamma \vdash M \rhd^+ N : A$.

Proof

The proof is done by induction on $M \to_p N$, where most cases are trivial but the case of parallel β-reduction. Whereas in the proof of the *Diamond Property*, we already had a well-typed path to use with the BETA rule, this time we need to build one.

We are in the following situation:

$$\frac{M \to_p M' \qquad N \to_p N'}{(\lambda x^A.M)_{\Pi x:C.D}\, N \to_p M'[N'/x]}$$

and $\Gamma \vdash (\lambda x^A.M)_{\Pi x:C.D}\, N \rhd (\lambda x^A.M)_{\Pi x:C.D}\, N : T$. By *Generation*, we have two possibilities: the typing judgment is either built from APP or from BETA. In both cases, we know that $\Gamma \vdash T \cong_\beta D[N/x]$ so we can replace T right now. In the latter case, we have every information at hand to prove that $\Gamma \vdash (\lambda x^A.M)_{\Pi x:C.D}\, N \rhd M'[N'/x] : D[N/x]$. The problem arises if we only have typing information coming from the APP rule:

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

Using Π-*injectivity*, we can show that $\Gamma \vdash A \cong_\beta C$, and *Confluence* gives us A_0 such that $\Gamma \vdash A \rhd^+ A_0 : s$ and $\Gamma \vdash C \rhd^+ A_0 : t$. The same argument is valid for B and D, so we have B_0 such that $\Gamma(x:A) \vdash B \rhd^+ B_0 : s'$ and $\Gamma \vdash D \rhd^+ B_0 : t'$.

Using Theorem 3.8, we can replace s by s_1, t by t_1, s' by s_2 and t' by t_2, which allows us to prove that

$$\Gamma \vdash (\lambda x^A.M)_{\Pi x:C.D} N \rhd^+ (\lambda x^A.M)_{\Pi x:A_0.B_0} N : D[N/x]$$

With this new redex, we can now use BETA on its right-hand side, proving that:

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

By induction, we have that $\Gamma(x:A) \vdash M \rhd^+ M' : B$ and $\Gamma \vdash N \rhd^+ N' : C$, so 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

Our 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 face some difficult situations while performing induction. Let us take a simple example with the construction of Π-types with the PI rule:

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

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 may involve "non-types". So we have to state a more general lemma about the conversion of different annotated versions of a same PTS term.

Lemma 4.1 (*Erased Confluence*)
If $|M| \equiv |N|$, $\Gamma \vdash M \rhd M : A$ and $\Gamma \vdash N \rhd N : B$, 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 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:

$$\begin{array}{ll} \Gamma \vdash P \rhd^+ P_0 : \Pi x^C.D & \Gamma \vdash Q \rhd^+ Q_0 : C \\ \Gamma \vdash P' \rhd^+ P_0 : \Pi x^{C'}.D' & \Gamma \vdash Q' \rhd^+ Q_0 : C' \end{array}$$

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 $\Gamma \vdash M \rhd N : A$ and $\Gamma \vdash M \rhd P : B$, 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.

We can apply the previous lemma to P_0 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 3.13.

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 is 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*, as we did for *Subject Reduction*. 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 $|A| \equiv |B|$, $\Gamma \vdash A \rhd A : s$ and $\Gamma \vdash B \rhd B : t$ 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 $\Gamma_1 \cong_\beta \Gamma_2$. 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 similar to Adams' proof for TPOSR, with a few type exchanges in the BETA case. □

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 2.17, 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 3.12, $|\Gamma^+| \vdash_e |M^+| : |T^+|$ which is equal to $\Gamma \vdash_e M : T$.
- If $\Gamma \vdash_e M =_\beta N : T$, so we conclude by Theorem 2.17.
- 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).

$$\begin{array}{lll} & \Gamma^+ \vdash M^+ \rhd M^+ : T^+ & \\ \Rightarrow & \Gamma^+ \vdash M^+ \rhd^+ P^+ : T^+ & \text{(Subject Reduction)} \\ \Rightarrow & \Gamma \vdash_e M =_\beta P : T & \text{(Theorem 3.12 and TRANS)} \end{array}$$

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 Subject Reduction in PTS$_e$

Now that we have a way to go from PTSs 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 3.18, we can prove that $\Gamma^+ \vdash M^+ \rhd^+ N^+ : T^+$. By erasing the annotations using the last part of Theorem 3.12, we end up having $|\Gamma^+| \vdash_e |M^+| =_\beta |N^+| : |T^+|$ which is the exact result we wanted. □

We showed how to map PTS derivations to PTS$_{atr}$ derivations. We believe that the same could have been done directly from PTS$_e$ to PTS$_{atr}$. That would have provided with a direct way to transfer *Subject Reduction* in PTS$_{atr}$ to *Subject Reduction* in PTS$_e$ and the equivalence between PTSs and PTS$_e$ would then just have been a consequence of *Subject Reduction* in PTS$_e$.

4.4 Weak Π*-injectivity in PTS$_e$*

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 only state the ones that are really need here:

1. If $\Gamma \vdash_e \Pi x^A.B : T$ then there are s_1, s_2, s_3 such that $(s_1, s_2, s_3) \in \mathscr{R}$, $\Gamma \vdash_e A : s_1$, $\Gamma(x : A) \vdash_e B : s_2$, 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 $(s_1, s_2, s_3) \in \mathscr{R}$, $\Gamma \vdash_e A : s_1$, $\Gamma(x : A) \vdash_e M : B$, $\Gamma(x : A) \vdash_e B : s_2$ 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 $(s_1, s_2, s_3) \in \mathscr{R}$ and $(s_1', s_2', s_3') \in \mathscr{R}$.

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 PTSs. 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 prove that both presentations are describing the same theory, without having to rely on specific model-based proofs of normalization.

This result can also be seen as a completion of Adams' syntactic approach to the meta-theory of PTS_e. In particular, two main properties of PTSs based on judgmental equality can now be stated and proved in a precise way: *Subject Reduction* and *Weak -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 PTSs, 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.

Acknowledgments

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, Stéphane Lengrand, Paul-André Melliès and Randy Pollack for many stimulating discussions on the topic.

References

Abel, A. (2010). Towards Normalization by Evaluation for the $\beta\eta$-Calculus of Constructions. *Pages 224–239 of:* Blume, Matthias, Kobayashi, Naoki, & Vidal, Germán (eds), *Functional and Logic Programming, 10th International Symposium, FLOPS 2010, Sendai, Japan, April 19-21, 2010. proceedings*. Lecture Notes in Computer Science, vol. 6009. Springer-Verlag.

Abel, A., Coquand, T., & Dybjer, P. (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, R. (2006). Pure Type Tystems with Judgemental Equality. *Journal of Functional Programming*, **16**(2), 219–246.

Barendregt, H. (1991). Introduction to Generalized Type Systems. *Journal of Functional Programming*, **1**(2), 125–154.

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

Berardi, S. (1990). *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. *Indagtiones Mathematica*, **34**(5), 381–392.

Geuvers, H. (1993). *Logics and Type Systems*. Ph.D. thesis, Katholieke Universiteit Nijmegen.

Geuvers, H., & Nederhof, M-J. (1991). Modular Proof of Strong Normalization for the Calculus of Constructions. *Journal of Functional Programming*, **1**(2), 155–189.

Geuvers, H., & Werner, B. (1994). On the Church-Rosser Property for Expressive Type Systems and its Consequences for their Metatheoretic Study. *Pages 320–329 of: Logic In Computer Science*.

Goguen, H. (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, P. (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, B., Petersson, K., & Smith, J. M. (1990). *Programming in Martin-Löf's Type Theory: An introduction*. Oxford University Press, USA.

Norell, U. 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.

Pollack, Robert. (1994). *The theory of LEGO: A proof checker for the extended calculus of constructions*. Ph.D. thesis, Univ. of Edinburgh.

Siles, V. (2010). *Formalization of Equivalence between PTS and PTSe*. `http://www.lix.polytechnique.fr/~vsiles/coq/PTSATR.html`.

Siles, V., & Herbelin, H. (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, T. (1991). *Semantics of Type Theory: Correctness, Completeness, and Independence Results*. Cambridge, MA, USA: Birkhauser Boston Inc.

Terlouw, J. (1989). *Een nadere bewijstheoretische analyse van GSTTs (Manuscript, in Dutch)*. Tech. rept. University of Nijmegen, Netherlands.

van Benthem Jutting, L. S. (1993). Typing in Pure Type Systems. *Information and Computation*, **105**(1), 30–41.

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

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

Werner, B., & Lee, G. (2010). *A Proof-Irrelevant Model of CIC with Predicate Induction and Judgemental Equality*. Submitted to Logical Methods in Compute Science.