Equality is typable in Semi-Full Pure Type Systems

Vincent Siles
Ecole Polytechnique / INRIA / Laboratoire PPS
Equipe πr^2
23 avenue d'Italie, 75013 Paris, France
vincent.siles@lix.polytechnique.fr

Hugo Herbelin
INRIA / Laboratoire PPS
Equipe πr^2
23 avenue d'Italie, 75013 Paris, France
hugo.herbelin@inria.fr

***Abstract*—There are two usual ways to describe equality in a dependent typing system, one that uses an external notion of computation like beta-reduction, and one that introduces a typed judgement of beta-equality directly in the typing system.**

After being an open problem for some time, the general equivalence between both approaches has been solved by Adams for a class of pure type systems (PTSs) called functional. In this paper, we relax the functionality constraint and prove the equivalence for all semi-full PTSs by combining the ideas of Adams with a study of the general shape of types in PTSs.

As one application, an extension of this result to systems with sub-typing would be a first step toward bringing closer the theory behind a proof assistant such as Coq to its implementation.

I. Introduction

There are two traditional ways to specify a dependent type system. The first one relies on an external notion of equality, like β-conversion or $\beta\eta$-conversion through a conversion rule:

$$\frac{\Gamma \vdash M : A \qquad \Gamma \vdash B\ type \qquad A \equiv B}{\Gamma \vdash M : B}\ \text{CONV}$$

It is the case of systems like the *Extended Calculus of Constructions* (ECC [1]) or the *Calculus of Inductive Constructions* [2] on which the proof-system *Coq* [3] is based on. Each conversion step is not checked to be well-typed, but along with *Confluence* and *Subject Reduction*, we can ensure that everything goes fine.

The second one embeds a notion of equality directly in the type system. So there are two types of judgement: one to type terms, and one to type equalities. With this kind of approach, we can ensure that every conversion step is well-typed:

$$\frac{\Gamma \vdash_e M : A \qquad \Gamma \vdash_e A = B}{\Gamma \vdash_e M : B}\ \text{CONV}$$

Those systems are known as "type systems with judgmental equality". This is the case of *UTT* [4] or *Martin-Löf's Type Theory and Logical Framework* [5] from which the dependently typed programming language *Agda 2* [6] is derived.

Surprisingly, showing the equivalence between those two definitions is difficult. Geuvers and Werner [7] 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:

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

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

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

This property relies on a notion of confluence which will involve *Subject Reduction*: we are facing a circular dependency.

Usually the systems based on judgmental equality are better suited for theoretical considerations, like building models [8]. On the other hand, in systems with untyped conversion, we can concentrate on the purely computational content of conversion and get rid of the types which are a priori useless to compute (e.g. see [9]). However, in both cases, they seem to describe the same theory and we would like to be sure that it is effectively the case.

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, in *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 presentation, we would be able to prove the theory equal to its implementation.

For some times, the only way to do it was based on normalization procedures [10], [4], one system at once. This approach does not scale easily since it relies on the construction of a model. Recently, Adams [11] found a *syntactical* criterion and proved that every functional Pure Type System (PTS) is equivalent to its counterpart with judgmental equality. To do so, he defined an intermediate

system called *Typed Parallel One-Step Reduction* (TPOSR). This system embeds the idea of the judgmental systems with a typed notion of equality, along with the idea of parallel reduction which is at the heart of the proof of confluence.

An interesting class of systems for which we would have this equivalence is the ones with sub-typing, which are not functional. We managed to get rid of the functionality by considering instead another class of PTS called *full*[1].

More precisely, in this paper, we shall prove that every *semi-full* Pure Type System is equivalent to its judgmental equality counterpart by combining the approach used by Adams with a study of the general shape of types in PTSs. By switching from functional to semi-full, we are one step closer to adapt this result to full systems like *ECC* or *CIC*, which are underlying systems of *Coq*.

II. The meta-theory of PTS

In this section, we will give a general description of PTSs and the main results that we will need on *Pure Type Systems with Judgmental Equality* (PTSe).

A. Terms and Untyped Reductions

Definition *Structure of terms and contexts*

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

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

We consider two separate sets *Sorts* and *Vars*, where the latter is infinite. In the following, we will 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. $\Gamma = (x_1 : A_1) \dots (x_n : A_n)$, where all the x_i are distinct, and x_i can only appear in $A_j (j > i)$. $\Gamma(x) = A$ is a shortcut for $(x : A) \in \Gamma$. $\emptyset$ denotes the empty context, the set of x_i such that $\Gamma(x_i)$ exists is called the *domain* of Γ, or $Dom(\Gamma)$ and the concatenation of two contexts whose domains are disjoint is written $\Gamma_1\Gamma_2$.

The term $\lambda x^A.M$ (resp. $\Pi x^A.B$) bounds 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 $M[x/N]$ 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 $x \notin Dom(\Gamma)$). $\Gamma[x/N]$ is recursively defined as :

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

[1]full means that we are allowed to build any dependent product

Later on, we will need to use *telescopes* for λ and Π-types, defined as follows:

$$\begin{array}{rcl} \lambda\langle\rangle.M & ::= & M \\ \Pi\langle\rangle.M & ::= & M \\ \lambda\langle(x : A)\Delta\rangle.M & ::= & \lambda x^A(\lambda\Delta).M \\ \Pi\langle(x : A)\Delta\rangle.M & ::= & \Pi x^A(\Pi\Delta).M \end{array}$$

The notion of β-reduction ($\rightarrow_\beta$) is defined as the congruence closure of the relation $(\lambda x^A.M)N \rightarrow_\beta M[x/N]$ over the grammar of terms. $\twoheadrightarrow_\beta$ stands for the reflexive transitive closure of $\rightarrow_\beta$ and $\equiv_\beta$ for its reflexive-symmetric-transitive closure.

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. With it, we can prove some useful tools like Π-injectivity or Sort Uniqueness:

Lemma II.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 $\Pi x^A.B \equiv_\beta \Pi x^C.D$ then $A \equiv_\beta C$ and $B \equiv_\beta D$*
- *Sort Uniqueness: If $s \equiv_\beta t$ then $s = t$.*

B. Presentation of Pure Type System

1) Pure Type System: A PTS is a generic framework first presented by Berardi [12] and Terlouw to study a family of type systems all at once. Popular type systems like *simply typed lambda calculus*, *System F* or *CoC* are part of this family. There is plenty of literature on the subject [13] so we will only recall the main ideas of those systems.

To make the system generic enough, we will abstract the typing of 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 $Rel \subset (Sorts \times Sorts \times Sorts)$ will be used to check the good formation of Π-types. The typing rules for PTS are given in Fig. 1.

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.

Theorem II.2. Subject Reduction
If $\Gamma \vdash M : A$ and $M \twoheadrightarrow_\beta N$, then $\Gamma \vdash N : A$.

Proof: The proof can be found in [13]. We just want to put forward that it relies on *Confluence*, more precisely on the Π-injectivity of β-reduction. ∎

In this paper, we will later consider the sub-class of *semi-full* PTS:

Definition *Full and semi-full PTS*

- A PTS is full if for any s, t, there is u such that $(s, t, u) \in Rel$.

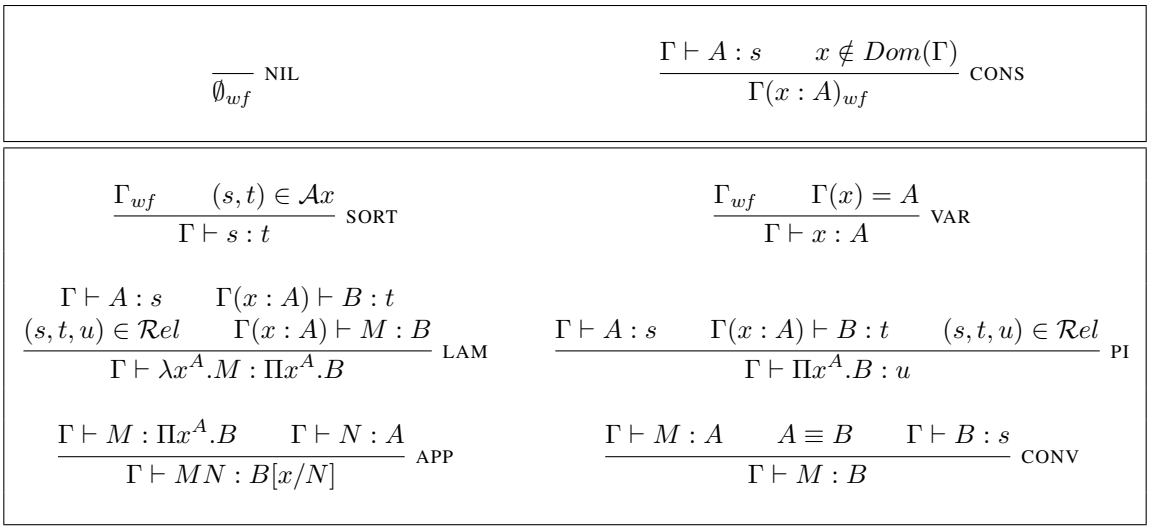

Figure 1. Typing Rules for PTS

- A PTS is semi-full if $(s,t,u) \in Rel$ enforces that for all t', there is u' such that $(s,t',u') \in Rel$.

Obviously, a full PTS is also semi-full.

2) Pure Type System with Judgmental Equality: There is another way to express PTSs by defining an internal notion of equality: Pure Type System with Judgmental Equality (PTSe), where every conversion step is well-typed. A complete presentation of PTSe can be found in [11]. For the rest of this paper, we will follow Adams' method and focus on the equivalence between PTS and TPOSR.

One result we need about PTSe is that every valid judgement in PTSe is valid in PTS:

Theorem II.3. From PTSe to PTS

1) *If* $\Gamma \vdash_e M : A$ *then* $\Gamma \vdash M : A$.
2) *If* $\Gamma \vdash_e M = N : A$ *then* $\Gamma \vdash M : A$, $\Gamma \vdash N : A$ *and* $M \equiv_\beta N$.

Proof: This theorem is valid for all PTS without restrictions. The proof is a simple induction and relies on properties of PTS, nothing is needed from PTSe at this point apart from their definition. ■

III. Basic meta-theory of TPOSR

A. Definition of TPOSR

In his approach to prove the equivalence for the functional PTSs, Adams defined a system called *Typed Parallel One Step Reduction* that inherits from the usual parallel reduction presentation, but in a typed way. The terms used in TPOSR are a little more informative than the terms used for PTS: the co-domain of the function is added as an annotation (with its binding variable) to all applications, to help proving the *Church-Rosser* property. Adams gives a detailed study about the necessity of this annotation in the third section of [11].

Definition *Structure of Annotated Terms*

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

All the other notions (context, substitution and untyped reduction) described for PTS are defined in the same way for TPOSR with the natural adaptation to the annotated applications. We define an erasure procedure $|\ |$ by induction on the structure of terms that maps TPOSR terms to PTS ones by inductively removing the additional typing information within the applications.

The typing rules of TPOSR are presented in Fig. 2. Sometimes we will write $\Gamma \vdash M \rhd N : A, B$ as a shortcut for $\Gamma \vdash M \rhd N : A$ and $\Gamma \vdash M \rhd N : B$, or $\Gamma \vdash M \rhd ? : A$ as a shortcut for "there is some N such that $\Gamma \vdash M \rhd N : A$".

Then we define the transitive and reflexive-symmetric-transitive closures of TPOSR in Fig. 3. It is interesting to notice that we can keep the same type at every step of a reduction sequence (as we will see further on), but not in an expansion sequence: without the functionality, we also lose the fact that a type only lives in a unique sort, so we cannot enforce to keep the same sort along a conversion sequence that links two types. But this will not be a problem since we are only interested in *equality at the type level*, so we need to check that every step is a well-formed type but we do not keep track of its sort.

B. General properties of TPOSR

Without any additional property like functional or semi-full, we can start to prove some properties of TPOSR:

(empty) $$\frac{}{\emptyset_{wf}}$$

(extend) $$\frac{\Gamma \vdash A \rhd A' : s}{\Gamma(x : A)_{wf}} \quad x \notin Dom(\Gamma)$$

(sort) $$\frac{\Gamma_{wf}}{\Gamma \vdash s \rhd s : t} \quad (s,t) \in \mathcal{A}x$$

(var) $$\frac{\Gamma_{wf}}{\Gamma \vdash x \rhd x : A} \quad \Gamma(x) = A$$

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

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

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

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

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

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

Figure 2. Typing Rules for the TPOSR system

$$\frac{\Gamma \vdash M \rhd N : A}{\Gamma \vdash M \rhd^+ N : A} \qquad \frac{\Gamma \vdash M \rhd^+ N : A \qquad \Gamma \vdash N \rhd^+ P : A}{\Gamma \vdash M \rhd^+ P : A}$$

$$\frac{\Gamma \vdash A \rhd B : s}{\Gamma \vdash A \equiv B} \qquad \frac{\Gamma \vdash B \equiv A}{\Gamma \vdash A \equiv B} \qquad \frac{\Gamma \vdash A \equiv B \qquad \Gamma \vdash B \equiv C}{\Gamma \vdash A \equiv C}$$

Figure 3. Multi step Reduction and Equality in TPOSR

Lemma III.1. 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* $\Gamma_1\Gamma_2\ {}_{wf}$, $\Gamma_1 \vdash A \rhd A' : s$ *and* $x \notin Dom(\Gamma_1\Gamma_2)$ *then* $\Gamma_1(x : A)\Gamma_2\ {}_{wf}$.

Lemma III.2. Substitution

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

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

Definition *Context Conversion*

- $\emptyset \equiv \emptyset$.
- If $\Gamma \equiv \Gamma'$, $\Gamma \vdash A \equiv B$ and $x \notin Dom(\Gamma) \bigcup Dom(\Gamma')$, then $\Gamma(x : A) \equiv \Gamma'(x : B)$.

Lemma III.3. Conversion in Context

- *If* $\Gamma \vdash M \rhd N : A$ *and* $\Gamma \equiv \Gamma'$ *then* $\Gamma' \vdash M \rhd N : A$.
- *If* $\Gamma \vdash M \rhd^+ N : A$ *and* $\Gamma \equiv \Gamma'$ *then* $\Gamma' \vdash M \rhd^+ N : A$.
- *If* $\Gamma \vdash A \equiv B$ *and* $\Gamma \equiv \Gamma'$ *then* $\Gamma' \vdash A \equiv B$.

Lemma III.4. Reflexivity

1) *If* $\Gamma \vdash M \rhd N : A$, *then* $\Gamma \vdash M \rhd M : A$.
2) *If* $\Gamma \vdash M \rhd N : A$, *then* $\Gamma \vdash N \rhd N : A$.

The following lemma is often called the *Generation* lemma, or the *Inversion* lemma. It allows us to know the general shape of a reduction based on the shape of the term that is reduced:

Lemma III.5. Generation

1) *If* $\Gamma \vdash s \rhd N : T$ *then* $N = s$ *and there is* t *such that* $(s, t) \in Ax$ *and either* $T = t$ *or* $\Gamma \vdash T \equiv t$.
2) *If* $\Gamma \vdash x \rhd N : T$ *then* $N = x$ *and there is* A *such that* $\Gamma(x) = A$ *and* $\Gamma \vdash T \equiv A$.
3) *If* $\Gamma \vdash \Pi x^A.B \rhd N : T$ *then there are* A', B', s, t, u *such that* $N = \Pi x^{A'}.B'$, $(s, t, u) \in Rel$, $\Gamma \vdash A \rhd A' : s$, $\Gamma(x : A) \vdash B \rhd B' : t$ *and either* $T = u$ *or* $\Gamma \vdash T \equiv u$.
4) *If* $\Gamma \vdash \lambda x^A.M \rhd N : T$ *then there are* A', M', B, B', s, t, u *such that* $N = \lambda x^{A'}.M'$, $(s, t, u) \in Rel$, $\Gamma \vdash A \rhd A' : s$, $\Gamma(x : A) \vdash B \rhd B' : t$, $\Gamma(x : A) \vdash M \rhd M' : B$ *and* $\Gamma \vdash T \equiv \Pi x^A.B$.
5) *If* $\Gamma \vdash P_{(x)B}Q \rhd N : T$ *then there are* A, A', B', Q', s, t, u *such that* $(s, t, u) \in Rel$, $\Gamma \vdash A \rhd A' : s$, $\Gamma(x : A) \vdash B \rhd B' : t$, $\Gamma \vdash Q \rhd Q' : A$, $\Gamma \vdash T \equiv B[x/Q]$ *and*
   - *either* $\Gamma \vdash P \rhd P' : \Pi x^A.B$ *and* $N = P'_{(x)B'}Q'$ *for some* P'
   - *or* $P = \lambda x^A.R$, $\Gamma(x : A) \vdash R \rhd R' : B$ *and* $N = R'[x/Q']$ *for some* R, R'.

The next theorem is the main reason why we can keep track of the type while doing several consecutive reductions, but not while doing expansions:

Lemma III.6. Type Exchange

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 annotation on the applications. ∎

This property allows us to prove that the following transitivity lemma for $\rhd^+$ is admissible:

$$\frac{\Gamma \vdash M \rhd^+ N : A \qquad \Gamma \vdash N \rhd^+ P : B}{\Gamma \vdash M \rhd^+ P : A}$$

It will also be used in the proof of *Church-Rosser* to avoid using the type uniqueness property at some minor stage of the proof.

Lemma III.7. Type Correctness

If $\Gamma \vdash M \rhd N : A$, *then there is* $s \in Sorts$ *such as either:* $A = s$ *or* $\Gamma \vdash A \rhd A' : s$ *for some* A'.

Theorem III.8. From TPOSR to PTS and PTSe

1) *If* $\Gamma \vdash M \rhd N : A$ *then* $|\Gamma| \vdash |M| : |A|$, $|\Gamma| \vdash |N| : |A|$ *and* $|M| \equiv_\beta |N|$.
2) *If* $\Gamma \vdash M \rhd N : A$ *then* $|\Gamma| \vdash_e |M| : |A|$, $|\Gamma| \vdash_e |N| : |A|$ *and* $|\Gamma| \vdash_e |M| = |N| : |A|$.

Lemma III.9. Sort and Π-types incompatibility

It is impossible to prove that $\Gamma \vdash \Pi x^A.B \equiv s$ *for any* Γ, A, B, s.

Proof: The proof relies on a translation of the equality judgement $\Gamma \vdash \Pi x^A.B \equiv s$ in the PTSs by erasure of the annotation with Theorem III.8.1. The confluence of β-reduction forbids that $\Pi x^{|A|}.|B| \equiv s$ 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 TPOSR. The reason is that, in a typed framework like PTSe or TPOSR, the *Confluence* and *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*. The next two sections will describe how we get rid of this circular dependency in the functional and semi-full cases.

IV. The *Church-Rosser* Property in TPOSR

The next step in the meta-theory is to prove the *Church-Rosser* property by proving that TPOSR enjoys the *Diamond Property*:

Theorem IV.1. Diamond Property

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

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

We are trying to close the classic *Church-Rosser* diamond diagram in a typed way. Almost all the cases are simply done by induction, but the possible cases involving an application constructor $(app)/(app)$, $(app)/(beta)$ or $(beta)/(app)$ resist in two ways:

1) some types involved in the conclusion of those judgements are substituted (e.g. $D[x/N]$), so we do not have the complete typing information for D. That is why the annotation is so useful: it lets us use an induction hypothesis over D.
2) the annotation only gives us information about the codomain of the "function", nothing about the domain.

The second problem is a serious one since it forbids us to use one of our induction hypothesis: by doing the proof by induction, we require the context to be the same in both branches of the theorem. For example in the (app)/(app) case, we are in the following situation (after using some generation lemmas):

$M = U_{(x)B}V \quad N = U'_{(x)B'}V' \quad P = U''_{(x)B''}V''$

$\Gamma \vdash U \rhd U' : \Pi x^A.B \qquad \Gamma \vdash V \rhd V' : A$

$\Gamma \vdash U \rhd U'' : \Pi x^C.B \qquad \Gamma \vdash V \rhd V'' : C$

$\Gamma(x : A) \vdash B \rhd B' : s \qquad \Gamma(x : C) \vdash B \rhd B'' : t$

The induction hypothesis gives us two more judgments:

- there is U_0 such that $\Gamma \vdash U' \rhd U_0 : \Pi x^A.B, \Pi x^C.B$ and $\Gamma \vdash U'' \rhd U_0 : \Pi x^A.B, \Pi x^C.B$.
- there is V_0 such that $\Gamma \vdash V' \rhd V_0 : A, C$ and $\Gamma \vdash V'' \rhd V_0 : A, C$.

In order to get a common reduct for B' and B'', we need to apply the induction hypothesis, but it requires the second context to be syntactically equal to $\Gamma(x : A)$. If we could have $\Gamma \vdash A \equiv C$, we would be able to apply Lemma III.3 to make both context match each other and apply the induction hypothesis.

A. *Proof of* Church-Rosser *in the functional case*

By assuming functionality, we get the *Uniqueness of Types* property at hand:

Lemma IV.2. Uniqueness of Types

If $\Gamma \vdash M \rhd ? : A$ *and* $\Gamma \vdash M \rhd ? : B$, *then* $A = B$ *or* $\Gamma \vdash A \equiv B$.

By applying Lemma III.4.2 to the judgments on V_0, we have that $\Gamma \vdash V_0 \rhd ? : A, C$ which gives us $\Gamma \vdash A \equiv C$ by *Uniqueness of Types*. We can now close all the critical pairs by applying the correct induction hypothesis.

B. *Proof of* Church-Rosser *in the semi-full case*

This time, we do not have the *Uniqueness of Types*. We decided to make a deeper study of the types in TPOSR in order to identify if there was not another way to enforce the equality of types that we needed to conclude the proof of the *Diamond Property*.

In [14], Jutting described a general "shape of types" of PTSs:

Definition *Terms classification in PTS*

- Every x is in Tv.
- If M is in Tv, then MN and $\lambda x^A.M$ are also in Tv.
- Every s or $\Pi x^A.B$ is in Ts.
- If M is in Ts, then MN and $\lambda x^A.M$ are also in Ts.

Theorem IV.3. Shape of types in PTS

$\forall M \in Tv$, *if* $\Gamma \vdash M : A$ *and* $\Gamma \vdash M : B$, *then* $A \equiv B$.

$\forall M \in Ts$, *if* $\Gamma \vdash M : A$ *and* $\Gamma \vdash M : B$, *then there are* Δ, s, t *such that* $A \twoheadrightarrow_\beta \Pi\Delta.s$ *and* $B \twoheadrightarrow_\beta \Pi\Delta.t$.

He showed that even if a PTS does not have the *Uniqueness of Types* property, it enjoys a weaker form of equality in the sense that two types of a same term may only differ by the last sort of their telescope form. This is really close to what we needed to conclude.

It is easy to adapt the definitions of Tv and Ts to the TPOSR framework. However the shape of types slightly changes and is expressed as:

Theorem IV.4. Shape of types in TPOSR

If $\Gamma \vdash M \rhd ? : A$ *and* $\Gamma \vdash M \rhd ? : B$, *then:*

- *either* $\Gamma \vdash A \equiv B$
- *or there are* Δ, s, t *such that:*
  - *either* $\Gamma \vdash A \equiv \Pi\Delta.s$ *or* $A = s$ *and* $\Delta = \langle\rangle$
  - *and either* $\Gamma \vdash B \equiv \Pi\Delta.t$ *or* $B = t$ *and* $\Delta = \langle\rangle$

The additional $\Delta = \langle\rangle$ conclusion when A or B is a sort comes from the fact that our typed equality is reflexive only on well-typed terms, and a sort (e.g. the third sort of a triplet of Rel) may not be well-typed. But if A is a sort, we still want B's telescope to be empty, even if it is convertible to a sort.

Proof: The proof is almost trivial by induction on the first judgement and generation on the second judgement, but in the (lam) case, we will need to build a Π-type equality from another equality:

Wrong Lemma. (naïve) Π-functionality

If $\Gamma(x : A) \vdash B \equiv B'$ *then* $\Gamma \vdash \Pi x^A.B \equiv \Pi x^A.B'$.

This is where semi-full is needed: we know in the proof of this theorem that the first term $\Pi x^A.B$ is well-typed, but we cannot enforce that every Π-type that appears in the conversion sequence will be well-formed. Indeed, since we cannot keep track of the sorts in the sequence, we know that every term between B and B' is well typed by a sort, but we cannot be sure that every resulting Π-type will have a valid triplet in Rel. This is why we need the semi-full hypothesis: we need to be sure that if the first Π-type is well-typed, then all the other that may appear in the sequence are also well-typed. A correct statement is:

Lemma IV.5. Π-functionality

If $\Gamma(x : A) \vdash B \equiv B'$, $\Gamma \vdash A \rhd ? : s$ *and* $(s, t, u) \in Rel$, *then* $\Gamma \vdash \Pi x^A.B \equiv \Pi x^A.B'$.

This lemma is trivial by induction on the length of the sequence, as soon as we are semi-full. ∎

You can notice that this property is somehow weaker: since we do not have yet the Π-injectivity in TPOSR, we had to trade $\twoheadrightarrow$ for $\equiv$ and we do not have the clear separation between Tv and Ts anymore. However this is enough to prove the *Diamond Property*. Now that we have more information about the shape of types, we can go back to the main proof.

By applying Theorem IV.4 to the two premises about U $\Gamma \vdash U \rhd U' : \Pi x^A.B$ and $\Gamma \vdash U \rhd U'' : \Pi x^C, B$, we get:

- either $\Gamma \vdash \Pi x^A.B \equiv \Pi x^C.B$
- or $\Gamma \vdash \Pi x^A.B \equiv \Pi\Delta.s$ and $\Gamma \vdash \Pi x^C.B \equiv \Pi\Delta.t$.

In both cases, by erasing the annotations of TPOSR with Theorem III.8 and using the Π-injectivity of PTSs, we get $|A| \equiv |C|$.

In the same way, if we apply Theorem IV.4 to V_0, we have either $\Gamma \vdash A \equiv C$ or both of $\Gamma \vdash A \equiv \Pi\Delta'.s'$ and $\Gamma \vdash C \equiv \Pi\Delta'.t'$. In the second case, we can again erase the annotations to translate the equality back to PTSs: $|A| \equiv |C|$ implies that $\Pi|\Delta'|.s' \equiv \Pi|\Delta'|.t'$. In the untyped setting, we can easily prove by *Confluence* that this configuration forces $s' = t'$. Back to TPOSR, we get that $\Gamma \vdash A \equiv \Pi\Delta'.s' = \Pi\Delta'.t' \equiv C$, so $\Gamma \vdash A \equiv C$.

In both cases, we conclude that $\Gamma \vdash A \equiv C$, so we can apply the same conclusion as in the previous section, and finish the proof that the *Diamond Property* is valid for semi-full TPOSR.

V. Consequences of *Church-Rosser*

With *Church-Rosser*, we can finally settle with all the missing pieces of theory that are really tedious to show in a typed framework:

Lemma V.1. Confluence

If $\Gamma \vdash A \equiv B$, there are C, s, t such that $\Gamma \vdash A \rhd^+ C : s$ and $\Gamma \vdash B \rhd^+ C : t$.

Lemma V.2. Π-injectivity

If $\Gamma \vdash \Pi x^A.B \equiv \Pi x^C.D$ then $\Gamma \vdash A \equiv C$ and $\Gamma(x : A) \vdash B \equiv D$.

Theorem V.3. Subject Reduction

If $\Gamma \vdash M \rhd ? : A$ and $M \twoheadrightarrow_\beta N$ then $\Gamma \vdash M \rhd N : A$.

The proof of Π-injectivity in TPOSR completely relies on *Church-Rosser*, not directly on the semi-full hypothesis. So the proof is exactly the same whether we are functional or semi-full. However, to prove *Subject Reduction*, we will need the Π-functionality property in the case where we reduce a β-redex, so semi-full is also required at this point.

Finally, the Π-injectivity property allows us to refine the shape of types to retrieve the separation between Tv and Ts. This gives us a hint for later: a well-typed term in Ts is nothing but some λ abstractions followed by a term typed by a sort, which has to be a sort or a Π-type.

Theorem V.4. Full Shape of types in TPOSR

1) *If $M \in Tv$, $\Gamma \vdash M \rhd ? : A$ and $\Gamma \vdash M \rhd ? : B$, then $\Gamma \vdash A \equiv B$.*
2) *If $M \in Ts$, $\Gamma \vdash M \rhd ? : A$ and $\Gamma \vdash M \rhd ? : B$, then there are Δ, s, t such that:*
   - *either $\Gamma \vdash A \equiv \Pi\Delta.s$ or $A = s$ and $\Delta = \langle\rangle$.*
   - *and either $\Gamma \vdash B \equiv \Pi\Delta.t$ or $B = t$ and $\Delta = \langle\rangle$.*

VI. Equivalence of TPOSR and PTS

The last step to prove the equivalence is to prove the correctness of annotations, i.e. to prove that every judgement $\Gamma \vdash M : T$ can be annotated into a well-typed TPOSR derivation $\Gamma^+ \vdash M^+ \rhd M^+ : T^+$ where $|\Gamma^+| = \Gamma$, $|M^+| = M$ and $|T^+| = T$.

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 doing induction. Let's take a simple example with the construction of Π-types with the PI rule:

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

By induction, we get that $\Gamma_1 \vdash A_1 \rhd A_1 : s$ and $\Gamma_2(x : A_2) \vdash B_2 \rhd B_2 : t$ with $|\Gamma_1| = |\Gamma_2| = \Gamma$ and $|A_1| = |A_2| = A$. 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 TPOSR*. 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 TPOSR, they are equivalent in TPOSR. Only with this, we will be able to state a similar lemma for context 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 judgement 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 the same PTS term.

Lemma VI.1. Erased Confluence

If $|M| = |N|$, $\Gamma \vdash M \rhd ? : A$ and $\Gamma \vdash N \rhd ? : B$, then there is R such that
$\Gamma \vdash M \rhd^+ R : A, B$ *and* $\Gamma \vdash N \rhd^+ R : A, B$.

The proof is done by induction on M, the only difficult part is again the application case: $M = P_{(x)D}Q,\ N = P'_{(x)D'}Q' \quad |P| = |P'|,\ |Q| = |Q'|$

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

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

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

A. Proof of Erased Confluence in the functional case

Again, thanks to the *Uniqueness of Types* and Π-injectivity we get that $\Gamma(x : C) \vdash D \equiv D'$. By *Confluence*, we get a common reduct D_0 for D and D', so the common reduct of M and N is $P_0\ {}_{D_0}Q_0$.

B. Proof of Erased Confluence in the semi-full case

We now show that we can adapt the proof of the functional case to the semi-full case by weakening the statement of the lemma:

Lemma VI.2. Erased Confluence in semi-full PTS

If $|M| = |N|$, $\Gamma \vdash M \rhd ? : A$ *and* $\Gamma \vdash N \rhd ? : B$ *, then there is* R *such that*
$\Gamma \vdash M \rhd^+ R : A$ *and* $\Gamma \vdash N \rhd^+ R : B$.

Without the *Uniqueness of Types*, we are not sure that all the correct annotations of an application are convertible, we can only attest that when it is not the case, they are convertible apart from their last sort. With this fact in mind, it is easy to realize that we will not be able to make the reduction sequences in both types, since the annotations that appear in the reduction of type A have no chance to match the types involved in the reduction of type B (and the other way around).

Another consequence is that we do not have the equality of the annotations $\Gamma(x : C) \vdash D \equiv D'$ anymore, we just know that $\Gamma \vdash P_0 \rhd ? : \Pi x^C, D, \Pi x^{C'}, D'$. But we can concentrate ourselves on the types of P_0.

If $P_0 \in Tv$, then we have $\Gamma \vdash \Pi x^C.D \equiv \Pi x^{C'}.D'$ which gives, thanks to the Π-injectivity, the equality $\Gamma(x : C) \vdash D \equiv D'$ and allows us to conclude in the same way than than we did for the functional case.

However, if $P_0 \in Ts$, it seems that we are stuck. In the proof of *Church-Rosser*, we only cared about the domains C and C', but here we need to find a common reduct for D and D'. The idea of translating to PTSs to justify that the a priori different sorts are in fact the same one won't work, because this time, it is the very last sort that bothers us, and even in the untyped setting they may be totally different. We need something else.

As we previously said, we noticed that terms in Ts have a very particular shape: they are build around sorts and Π-types. By erasing the annotations and using the translation from TPOSR to PTS, we can prove that neither a sort nor a Π-type can be typed by a Π-type. This means that all the applications hidden inside a Ts term are simply β-redexes, which leads to the following (simplified) definition:

Lemma VI.3. Shape of Ts' terms in TPOSR
If $M \in Ts$, $\Gamma \vdash M \rhd ? : A$ *and* $\Gamma \vdash M \rhd ? : B$, *then there are* Δ, K, s, t *such that:*

- $\Gamma \vdash M \rhd^+ \lambda\Delta.K : A$ *and* $\Gamma \vdash A \equiv \Pi\Delta.s$
- $\Gamma \vdash M \rhd^+ \lambda\Delta.K : B$ *and* $\Gamma \vdash B \equiv \Pi\Delta.t$

where K *is a sort or a* Π*-type.*

The real important fact here is that M reduces to the *exact same* telescope in *both* types and that the proof requires the framework to be semi-full for the same reason as Π-functionality. The full statement and the proof are quite technical, and are not the main point here. They can be found in the *Coq* formalization, see [15].

Back to the proof of Lemma. VI.2, since P_0's type is a Π-type, we are sure that the Δ involved in its shape is not empty. As a consequence, we can build a reduction from P_0 to a valid λ-abstraction whose domain can be either the type of Q or Q' thanks to the conclusion about the types of the telescope in Lemma VI.3 and the Π-injectivity.

We were not allowed to find a common reduct to D and D', but this is not the only solution anymore: the common reduct here will not be another simple application, but the result of the β-reduction initiated by the λ-abstraction reduced from P_0 applied to Q (resp. Q'). By generation, we know that $\Gamma \vdash A \equiv D[x/Q]$ and $\Gamma \vdash B \equiv D'[x/Q']$ so we can do the β-reduction in M and N:

$$
\begin{array}{lll}
\Gamma \vdash P_{(x)D}Q & \rhd^+ & P_0\ {}_{(x)D}Q : D[x/Q] \\
 & \rhd^+ & (\lambda x^C \lambda\Delta.K)_{(x)D}Q : D[x/Q] \\
 & \rhd^+ & \lambda\Delta[x/Q].K[x/Q] : D[x/Q] \\
 & \rhd^+ & \lambda\Delta[x/Q_0].K[x/Q_0] : D[x/Q] \\
\Gamma \vdash P'_{(x)D'}Q' & \rhd^+ & P_0\ {}_{(x)D'}Q' : D'[x/Q'] \\
 & \rhd^+ & (\lambda x^{C'} \lambda\Delta.K)_{(x)D'}Q' : D'[x/Q'] \\
 & \rhd^+ & \lambda\Delta[x/Q'].K[x/Q'] : D'[x/Q'] \\
 & \rhd^+ & \lambda\Delta[x/Q_0].K[x/Q_0] : D'[x/Q']
\end{array}
$$

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

C. Consequences of the Erased Confluence

With Lemma VI.2, we can show what we needed about types and contexts:

Lemma VI.4. Erased Conversion

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

We can now conclude the last missing piece of the whole equivalence process:

Theorem VI.5. From PTS to TPOSR

If $\Gamma \vdash M : T$, *then there are* Γ^+, M^+, T^+ *such that* $\Gamma^+ \vdash M^+ \rhd M^+ : T^+$, $|\Gamma^+| = \Gamma$, $|M^+| = M$ *and* $|T^+| = T$.

Proof: Since we have managed to prove *Subject Reduction* and Lemma VI.4, the proof is strictly the same as in [11]. ∎

Finally, all of this leads us to state that:

Theorem VI.6. Equivalence of PTS and PTSe in the semi-full case

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

Proof: This is just a combination of the following lemmas:

- If $\Gamma \vdash_e M : T$, then by Theorem II.3, we have $\Gamma \vdash M : T$.
- If $\Gamma \vdash M : T$, by Theorem VI.5 we know that $\Gamma^+ \vdash M^+ \rhd M^+ : T^+$ with $|\Gamma^+| = \Gamma$, $|M^+| = M$ and $|T^+| = T$. By Theorem III.8, $|\Gamma^+| \vdash_e |M^+| : |T^+|$ which is equal to $\Gamma \vdash_e M : T$.
- If $\Gamma \vdash_e M = N : T$, so we conclude by Theorem II.3.
- If $\Gamma \vdash M : T$, $\Gamma \vdash N : T$ and $M \equiv_\beta N$, by *Confluence*, there is P such that $M \twoheadrightarrow_\beta P$ and $N \twoheadrightarrow_\beta P$. By Theorem VI.5, there are Γ', M', A' such that $|\Gamma'| = \Gamma$, $|M'| = M$, $|T'| = T$ and
  $\Gamma' \vdash M' \rhd M' : A'$
  $\Rightarrow \quad \Gamma' \vdash M' \rhd P' : A'$ (Subject Reduction)
  $\Rightarrow \quad \Gamma \vdash_e M = P : A$ (Theorem III.8 and $(trans)$)
  We do the same to conclude that $\Gamma \vdash_e N = P : A$, so by (sym) and $(trans)$, we finally have $\Gamma \vdash_e M = N : A$.

■

VII. Conclusion

We have proven that any semi-full PTS that uses an external notion of β-equality (PTS) is equivalent to its counterpart that uses a typing judgment to deal with β-equality (PTSe). Along with Adams' results for any functional PTS, we covert almost all known PTSs without having to rely on specific model-based proof of normalization.

The whole process described here is based on some technically complex lemmas, so everything has been formalized in the proof-assistant *Coq* [3], using de Bruijn indices [16] to handle variable bindings. The whole development can be found at [15].

The next step should be to adapt this approach to η-conversion, or to extended type systems with sub-typing. We are pretty confident in the first one since this will not change the shape of types in any way. However, the latter may need some more work to find the right shape of types once sub-typing is added. If it is the case, we would be able to apply this result to the *ECC* system.

However the question of a general equivalence for all PTS is still an open question since they may be some pathological PTS which are neither functional nor semi-full which are a counter-example.

Acknowledgements

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

References

[1] Z. Luo, "Ecc: an extended calculus of constructions," in *Proceedings of the Fourth Annual Symposium on Logic in computer science*. Piscataway, NJ, USA: IEEE Press, 1989, pp. 385–395.

[2] B. Werner, "Une théorie des constructions inductives," Ph.D. dissertation, Université Paris 7, 1994.

[3] Coq Development Team, "The coq proof assistant reference manual," http://coq.inria.fr/refman/.

[4] H. Goguen, "A typed operational semantics for type theory," Ph.D. dissertation, University of Edinburg, 1994.

[5] B. Nordstrom, K. Petersson, and J. M. Smith, *Programming in Martin-Löf's Type Theory: An Introduction*. Oxford University Press, USA, 1990.

[6] U. Norell, "Towards a practical programming language based on dependent type theory," Ph.D. dissertation, Department of Computer Science and Engineering, Chalmers University of Technology, SE-412 96 Göteborg, Sweden, September 2007.

[7] H. Geuvers and B. Werner, "On the church-rosser property for expressive type systems and its consequences for their metatheoretic study," in *LICS*, 1994, pp. 320–329.

[8] B. Werner and G. Lee, "A simple model of calculus of inductive constructions with judgemental equality," unpublished manuscript.

[9] B. Grégoire, "Compilation des termes de preuves: un (nouveau) mariage entre coq et ocaml." Thése de doctorat, spécialité informatique, Université Paris 7, école Polytechnique, France, December 2003. [Online]. Available: http://www-sop.inria.fr/everest/personnel/Benjamin.Gregoire/Publi/gregoire_these.ps.gz

[10] H. Geuvers, "Logics and type systems," Ph.D. dissertation, Katholieke Universiteit Nijmegen, 1993.

[11] R. Adams, "Pure type systems with judgemental equality," *J. Funct. Program.*, vol. 16, no. 2, pp. 219–246, 2006.

[12] S. Berardi, "Type dependence and constructive mathematics," Ph.D. dissertation, Mathematical Institute Torino, 1988.

[13] H. Barendregt, S. Abramsky, D. M. Gabbay, T. S. E. Maibaum, and H. P. Barendregt, "Lambda calculi with types," in *Handbook of Logic in Computer Science*. Oxford University Press, 1992, pp. 117–309.

[14] L. S. van Benthem Jutting, "Typing in pure type systems," *Inf. Comput.*, vol. 105, no. 1, pp. 30–41, 1993.

[15] V. Siles, "Formalization of equivalence between semi-full pts and ptse," http://www.lix.polytechnique.fr/~vsiles/coq/TPOSR.html.

[16] N. de Bruijn, "Lambda-calculus notation with nameless dummies: a tool for automatic formula manipulation with application to the Church-Rosser theorem," *Indag. Math.*, vol. 34, no. 5, pp. 381–392, 1972.