Investigation on the typing of equality in type systems

PhD Defense

Vincent Siles

under the supervision of B. Barras and H. Herbelin

Typical - $\pi.r^2$ Team
Ecole Polytechnique - INRIA - PPS

Nov 25th, 2010

What is a type ?

A type is a way to give a (more or less) detailed description of a whole family of things:

What is a type ?

A type is a way to give a (more or less) detailed description of a whole family of things:

What is a type ?

A type is a way to give a (more or less) detailed description of a whole family of things:

cute animal

vegetable

fruit

What is a type ?

A type is a way to give a (more or less) detailed description of a whole family of things:

cute animal

vegetable

fruit

With types, we can state general properties about these families like "vegetables are good for health".

Types in sciences

Types can also be used to describe mathematical objects or data-structures, and these information can be helpful to avoid mistakes in programs:

Types in sciences

Types can also be used to describe mathematical objects or data-structures, and these information can be helpful to avoid mistakes in programs:

- π is a real number, 1664 is a natural number ($\mathbb{N}$), $[1, 2, 3, 5, 7]$ is a list of natural numbers, ...

Types in sciences

Types can also be used to describe mathematical objects or data-structures, and these information can be helpful to avoid mistakes in programs:

- π is a real number, 1664 is a natural number ($\mathbb{N}$), $[1, 2, 3, 5, 7]$ is a list of natural numbers, ...
- $\texttt{plus} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is a function.

Types in sciences

Types can also be used to describe mathematical objects or data-structures, and these information can be helpful to avoid mistakes in programs:

- π is a real number, 1664 is a natural number ($\mathbb{N}$), $[1, 2, 3, 5, 7]$ is a list of natural numbers, ...
- `plus` : $\mathbb{N} \times \mathbb{N} \to \mathbb{N}$ is a function.
- `plus`$(\pi, 1664)$ is an *ill-typed* program since `plus` is expecting two natural numbers, while π is not one.

Proofs and Types (1)

- Types can be used to ensure the correctness of programs:

  "If `length` computes the length of a list, then we know that, if `l` is a valid list, then `length(l)` is valid (and has to be a natural number)."

Proofs and Types (1)

- Types can be used to ensure the correctness of programs:

  "If `length` computes the length of a list, then we know that, if `l` is a valid list, then `length(l)` is valid (and has to be a natural number)."

- This preservation of validity looks like a *proof step*:

  "If $A \Rightarrow B$ and A is valid, then B is valid."

Proofs and Types (1)

- Types can be used to ensure the correctness of programs:

  "If `length` computes the length of a list, then we know that, if `l` is a valid list, then `length(l)` is valid (and has to be a natural number)."
- This preservation of validity looks like a *proof step*:

  "If $A \Rightarrow B$ and A is valid, then B is valid."
- If *when it's snowing, I'm cold* and *it's snowing*, then we can conclude that *I'm cold*.

Proof and Types (2)

- There is a deeper connexion between proofs and programs, known as the *Curry-Howard* correspondence:

Proof		Program
Proposition	$\leftrightarrow$	Type
Proof	$\leftrightarrow$	Program

Proof and Types (2)

- There is a deeper connexion between proofs and programs, known as the *Curry-Howard* correspondence:

Proof		Program
Proposition	$\leftrightarrow$	Type
Proof	$\leftrightarrow$	Program

- This correspondence leads to the development of *proof assistants* such as *ACL2, Agda, Coq, HOL, Matita, PVS*, ...

Proof and Types (2)

- There is a deeper connexion between proofs and programs, known as the *Curry-Howard* correspondence:

Proof		Program
Proposition	$\leftrightarrow$	Type
Proof	$\leftrightarrow$	Program

- This correspondence leads to the development of *proof assistants* such as *ACL2, Agda, Coq, HOL, Matita, PVS*, ...

All of these proof assistants are based on particular *type theories* which ensure their correctness.

1. Pure Type Systems
2. Equivalence and Typed Reduction
3. Further extensions and conclusion

Simply Typed λ-Calculus

$$
\begin{array}{rcl}
M, N & ::= & x \mid \lambda x^A.M \mid M\ N \\
A, B & ::= & a \mid A \to B \\
\Gamma & ::= & \emptyset \mid \Gamma, x : A
\end{array}
$$

Simply Typed λ-Calculus

$$
\begin{array}{rcl}
M, N & ::= & x \mid \lambda x^A.M \mid M\ N \\
A, B & ::= & a \mid A \to B \\
\Gamma & ::= & \emptyset \mid \Gamma, x : A
\end{array}
$$

$$
\frac{\Gamma(x) = A}{\Gamma \vdash x : A} \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x^A.M : A \to B} \qquad \frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B}
$$

Simply Typed λ-Calculus

$$
\begin{array}{rcl}
M, N & ::= & x \mid \lambda x^A.M \mid M\ N \\
A, B & ::= & a \mid A \rightarrow B \\
\Gamma & ::= & \emptyset \mid \Gamma, x : A
\end{array}
$$

$$
\frac{\Gamma(x) = A}{\Gamma \vdash x : A} \qquad \frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x^A.M : A \rightarrow B} \qquad \frac{\Gamma \vdash M : A \rightarrow B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B}
$$

- STLC was presented by Church in the 1930's.
- Since then, several extensions have been studied: *"Type:Type", System-***F***, Calculus of Constructions . . .*
- Each extension has more expressive power than the previous one (polynomials, second order arithmetic . . .)
- All these type systems share a common core, but their meta-theory were studied one at a time.

What is a dependent type

To make a dependent version of `list`, we can add the *length* to the type:

- e.g. $\texttt{l} \equiv [1, 3, 5, 7]$ is a list of natural number, of length 4. Its type is `list 4`.

What is a dependent type

To make a dependent version of `list`, we can add the *length* to the type:

- e.g. $\texttt{l} \equiv [1, 3, 5, 7]$ is a list of natural number, of length 4. Its type is `list 4`.
- concatenation of two non-dependent lists is of type

$$\texttt{list} \rightarrow \texttt{list} \rightarrow \texttt{list}.$$

What is a dependent type

To make a dependent version of `list`, we can add the *length* to the type:

- e.g. $\mathtt{l} \equiv [1, 3, 5, 7]$ is a list of natural number, of length 4. Its type is `list 4`.
- concatenation of two non-dependent lists is of type
$$\mathtt{list} \rightarrow \mathtt{list} \rightarrow \mathtt{list}.$$
- the dependent version (`concat`) is of type
$\Pi n^{nat}.\Pi m^{nat}.\quad \mathtt{list}\ n \rightarrow \mathtt{list}\ m \rightarrow \mathtt{list}\ (n+m)$

$\Pi x^{A}.B$ is called a *dependent product*.

Pure Type Systems

Pure Type Systems have been built to *unify* all these different presentations in a single system:

Pure Type Systems

Pure Type Systems have been built to *unify* all these different presentations in a single system:

$$\begin{array}{rcl} M, N, A, B & ::= & x \mid \lambda x^A.M \mid M\ N \mid \Pi x^A.B^\dagger \mid s \\ \Gamma & ::= & \emptyset \mid \Gamma, x : A \end{array}$$

- PTSs are an abstraction of Barendregt's λ-cube, presented independently by Berardi and Terlouw.

---

†We write $A \to B$ when B does not depend on the input.

Pure Type Systems

Pure Type Systems have been built to *unify* all these different presentations in a single system:

$$\begin{array}{rcl} M, N, A, B & ::= & x \mid \lambda x^A.M \mid M\ N \mid \Pi x^A.B^\dagger \mid s \\ \Gamma & ::= & \emptyset \mid \Gamma, x : A \end{array}$$

- PTSs are an abstraction of Barendregt's λ-cube, presented independently by Berardi and Terlouw.
- To be able to deal with all the different type systems, PTSs have parameters that describe which type is valid: $\mathit{Sorts}, \mathcal{A}x$ and $\mathcal{R}el$.

† We write $A \rightarrow B$ when B does not depend on the input.

Some typing rules for PTSs

STLC		PTS
$$\frac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x^A.M : A \to B}$$	$\Rightarrow$	$$\frac{\Gamma \vdash \Pi x^A.B : s \qquad \Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x^A.M : \Pi x^A.B}$$
$$\frac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B}$$	$\Rightarrow$	$$\frac{\Gamma \vdash M : \Pi x^A.B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B[N/x]}$$

Some typing rules for PTSs

STLC		PTS
$\dfrac{\Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x^A.M : A \to B}$	$\Rightarrow$	$\dfrac{\Gamma \vdash \Pi x^A.B : s \qquad \Gamma, x : A \vdash M : B}{\Gamma \vdash \lambda x^A.M : \Pi x^A.B}$
$\dfrac{\Gamma \vdash M : A \to B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B}$	$\Rightarrow$	$\dfrac{\Gamma \vdash M : \Pi x^A.B \qquad \Gamma \vdash N : A}{\Gamma \vdash M\ N : B[N/x]}$

$$\dfrac{\Gamma \vdash M : A \qquad A =_\beta B \qquad \Gamma \vdash B : s}{\Gamma \vdash M : B}\ \text{CONV}$$

Why a conversion rule ?

- Let's consider the type of lists of $\mathbb{N}$ of length `n`: `list n`.

Why a conversion rule ?

- Let's consider the type of lists of $\mathbb{N}$ of length `n`: `list n`.
- What is the result and the type of `concat l l` ?
  $\Pi n^{nat}.\Pi m^{nat}.\quad \texttt{list}\ n \to \texttt{list}\ m \to \texttt{list}\ (n+m)$

Why a conversion rule ?

- Let's consider the type of lists of $\mathbb{N}$ of length `n`: `list n`.
- What is the result and the type of `concat l l` ?
  $\Pi n^{nat}.\Pi m^{nat}.\quad \texttt{list}\ n \rightarrow \texttt{list}\ m \rightarrow \texttt{list}\ (n+m)$
- $\texttt{concat 4 4 l l} =_\beta [1,3,5,7,1,3,5,7]\ :\ \texttt{list}\ (4+4)$

Why a conversion rule ?

- Let's consider the type of lists of $\mathbb{N}$ of length `n`: `list n`.
- What is the result and the type of `concat l l` ?
  $\Pi n^{nat}.\Pi m^{nat}.\quad \texttt{list}\ n \rightarrow \texttt{list}\ m \rightarrow \texttt{list}\ (n+m)$
- `concat 4 4 l l` $=_\beta [1, 3, 5, 7, 1, 3, 5, 7]$ `:` `list` (4+4)

The conversion rule is here to *compute* at the level of types and change `list` (4+4) into `list` 8.

Facts about PTSs

For example:

Type Correctness

If $\Gamma \vdash M : T$ then there is $s \in \mathit{Sorts}$ such that $T \equiv s$ or $\Gamma \vdash T : s$.

Facts about PTSs

For example:

Type Correctness

If $\Gamma \vdash M : T$ then there is $s \in Sorts$ such that $T \equiv s$ or $\Gamma \vdash T : s$.

A more complex one:

Subject Reduction

If $\Gamma \vdash M : T$ and $M \to_\beta M'$ then $\Gamma \vdash M' : T$.

Needs Injectivity of Π-types: If $\Pi x^A.B =_\beta \Pi x^C.D$ then $A =_\beta C$ and $B =_\beta D$. (Easy by confluence of β-reduction)

Untyped conversion considered harmful ?

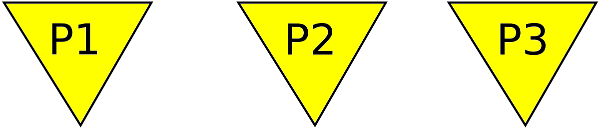
What if the path between A and B is “ill-typed” ?

$$\frac{}{(\lambda x^{A'}.P)\ Q =_\beta P[Q/x]} \qquad \frac{\Gamma \vdash M : A \qquad A =_\beta B \qquad \Gamma \vdash B : s}{\Gamma \vdash M : B}$$

Untyped conversion considered harmful ?

What if the path between *A* and *B* is "ill-typed" ?

Let's consider *P* to be the following proof of $\Gamma \vdash M : B$.



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

Untyped conversion considered harmful ?

What if the path between *A* and *B* is “ill-typed” ?

By Confluence:

$$\frac{\overset{\text{P1}}{\Gamma \vdash M : A} \quad \overset{\text{P'2}}{A \twoheadrightarrow_\beta C} \quad \overset{\text{P"2}}{B \twoheadrightarrow_\beta C} \quad \overset{\text{P3}}{\Gamma \vdash B : s}}{\Gamma \vdash M : B}$$

Untyped conversion considered harmful ?

What if the path between *A* and *B* is “ill-typed” ?

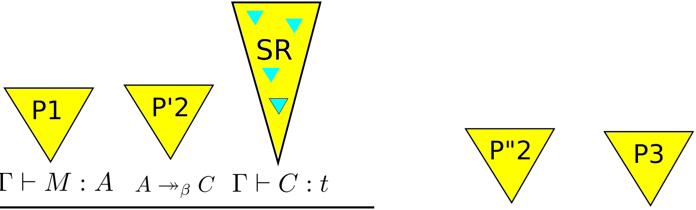
By Type Correctness and Subject Reduction:

$$\dfrac{\dfrac{\overset{\text{P1}}{\Gamma \vdash M : A} \quad \overset{\text{P'2}}{A \twoheadrightarrow_\beta C} \quad \overset{\text{SR}}{\Gamma \vdash C : t}}{\Gamma \vdash M : C} \qquad \overset{\text{P"2}}{B \twoheadrightarrow_\beta C} \quad \overset{\text{P3}}{\Gamma \vdash B : s}}{\Gamma \vdash M : B}$$

Untyped conversion considered harmful ?

What if the path between *A* and *B* is “ill-typed” ?

But Subject Reduction introduces new harmful conversions:



$$\dfrac{\dfrac{\Gamma \vdash M : A \quad A \twoheadrightarrow_\beta C \quad \Gamma \vdash C : t}{\Gamma \vdash M : C} \qquad B \twoheadrightarrow_\beta C \quad \Gamma \vdash B : s}{\Gamma \vdash M : B}$$

PTS with typed equality (PTSe)

To avoid this possibility, we can *type* every step of the conversion. They are called *semantical PTSs* or *PTSs with typed equality* [Geuvers93]. We have now two typing judgments:

PTS with typed equality (PTSe)

To avoid this possibility, we can *type* every step of the conversion. They are called *semantical PTSs* or *PTSs with typed equality* [Geuvers93]. We have now two typing judgments:

- One for terms: $\Gamma \vdash_e M : T$

PTS with typed equality (PTSe)

To avoid this possibility, we can *type* every step of the conversion. They are called *semantical PTSs* or *PTSs with typed equality* [Geuvers93]. We have now two typing judgments:

- One for terms: $\Gamma \vdash_e M : T$
- One for equalities: $\Gamma \vdash_e M =_\beta N : T$

PTS with typed equality (PTSe)

To avoid this possibility, we can *type* every step of the conversion. They are called *semantical PTSs* or *PTSs with typed equality* [Geuvers93]. We have now two typing judgments:

- One for terms: $\Gamma \vdash_e M : T$
- One for equalities: $\Gamma \vdash_e M =_\beta N : T$

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

PTS with typed equality (PTSe)

To avoid this possibility, we can *type* every step of the conversion. They are called *semantical PTSs* or *PTSs with typed equality* [Geuvers93]. We have now two typing judgments:

- One for terms: $\Gamma \vdash_e M : T$
- One for equalities: $\Gamma \vdash_e M =_\beta N : T$

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

PTS with typed equality

Untyped β-equality is quite "small":

$$\frac{}{(\lambda x^A.M)\ N =_\beta M[N/x]}$$

$$\frac{A =_\beta A' \qquad M =_\beta M'}{\lambda x^A.M =_\beta \lambda x^{A'}.M'}$$

PTS with typed equality

Typed β-equality is notably "bigger":

$$\frac{\begin{array}{c}\Gamma, x : A \vdash_e M : B \qquad \Gamma \vdash_e N : A \\ \Gamma \vdash_e A : s \qquad \Gamma, x : A \vdash_e B : t \qquad (s, t, u) \in \mathcal{R}el\end{array}}{\Gamma \vdash_e (\lambda x^A.M)N =_\beta M[N/x] : B[N/x]}$$

$$\frac{\begin{array}{c}\Gamma \vdash_e A =_\beta A' : s \qquad \Gamma, x : A \vdash_e M =_\beta M' : B \\ \Gamma, x : A \vdash_e B : t \qquad (s, t, u) \in \mathcal{R}el\end{array}}{\Gamma \vdash_e \lambda x^A.M =_\beta \lambda x^{A'}.M' : \Pi x^A.B}$$

Facts about PTSe

Almost all the properties that we know about PTSs are easily proved valid for PTSe, and there are new ones:

Facts about PTSe

Almost all the properties that we know about PTSs are easily proved valid for PTSe, and there are new ones:

Left-hand/Right-hand reflexivity

If $\Gamma \vdash_e M =_\beta N : T$ then $\Gamma \vdash_e M : T$ and $\Gamma \vdash_e N : T$.

Facts about PTSe

Almost all the properties that we know about PTSs are easily proved valid for PTSe, and there are new ones:

Left-hand/Right-hand reflexivity

If $\Gamma \vdash_e M =_\beta N : T$ then $\Gamma \vdash_e M : T$ and $\Gamma \vdash_e N : T$.

However, Subject Reduction is really troublesome to prove:

Typed Subject Reduction:

If $\Gamma \vdash_e M : T$ and $M \to_\beta N$, then $\Gamma \vdash_e M =_\beta N : T$.

To prove this as we did for PTSs, we need Π-Injectivity for typed equality judgments, which is a really difficult question for PTSe since it relies on (typed) property of *Confluence*, which relies on *Subject Reduction*,

Facts about PTSe

Almost all the properties that we know about PTSs are easily proved valid for PTSe, and there are new ones:

Left-hand/Right-hand reflexivity

If $\Gamma \vdash_e M =_\beta N : T$ then $\Gamma \vdash_e M : T$ and $\Gamma \vdash_e N : T$.

However, Subject Reduction is really troublesome to prove:

Typed Subject Reduction:

If $\Gamma \vdash_e M : T$ and $M \to_\beta N$, then $\Gamma \vdash_e M =_\beta N : T$.

To prove this as we did for PTSs, we need Π-Injectivity for typed equality judgments, which is a really difficult question for PTSe since it relies on (typed) property of *Confluence*, which relies on *Subject Reduction*, which relies on Π-Injectivity,

Facts about PTSe

Almost all the properties that we know about PTSs are easily proved valid for PTSe, and there are new ones:

Left-hand/Right-hand reflexivity

If $\Gamma \vdash_e M =_\beta N : T$ then $\Gamma \vdash_e M : T$ and $\Gamma \vdash_e N : T$.

However, Subject Reduction is really troublesome to prove:

Typed Subject Reduction:

If $\Gamma \vdash_e M : T$ and $M \rightarrow_\beta N$, then $\Gamma \vdash_e M =_\beta N : T$.

To prove this as we did for PTSs, we need Π-Injectivity for typed equality judgments, which is a really difficult question for PTSe since it relies on (typed) property of *Confluence*, which relies on *Subject Reduction*, which relies on Π-Injectivity, which relies on ...

Strong Π-Injectivity : a counterexample

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

Strong Π-Injectivity : a counterexample

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

By using the identity function as a *coercion*, one can restrict the different types of a term. Let's consider a particular PTS with (u, v) and (u, v') in the definition of its axioms:

1. u can be typed by v or v': $\emptyset \vdash u : v$ and $\emptyset \vdash u : v'$.

Strong Π-Injectivity : a counterexample

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

By using the identity function as a *coercion*, one can restrict the different types of a term. Let's consider a particular PTS with (u, v) and (u, v') in the definition of its axioms:

1. u can be typed by v or v': $\emptyset \vdash u : v$ and $\emptyset \vdash u : v'$.
2. $M_1 \equiv id_v\ u$ can only be typed by v: If $\emptyset \vdash M_1 : T$, then $T =_\beta v$.

Strong Π-Injectivity : a counterexample

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

By using the identity function as a *coercion*, one can restrict the different types of a term. Let's consider a particular PTS with (u, v) and (u, v') in the definition of its axioms:

1. u can be typed by v or v': $\emptyset \vdash u : v$ and $\emptyset \vdash u : v'$.
2. $M_1 \equiv id_v\ u$ can only be typed by v: If $\emptyset \vdash M_1 : T$, then $T =_\beta v$.
3. $M_2 \equiv id_{v'}\ u$ can only be typed by v': If $\emptyset \vdash M_2 : T$, then $T =_\beta v'$.

Strong Π-Injectivity : a counterexample

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

By using the identity function as a *coercion*, one can restrict the different types of a term. Let's consider a particular PTS with (u, v) and (u, v') in the definition of its axioms:

1. u can be typed by v or v': $\emptyset \vdash u : v$ and $\emptyset \vdash u : v'$.
2. $M_1 \equiv id_v\ u$ can only be typed by v: If $\emptyset \vdash M_1 : T$, then $T =_\beta v$.
3. $M_2 \equiv id_{v'}\ u$ can only be typed by v': If $\emptyset \vdash M_2 : T$, then $T =_\beta v'$.
4. $\emptyset \vdash_e \Pi x^{M_1}.u =_\beta \Pi x^{M_2}.u : w$

Strong Π-Injectivity : a counterexample

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

By using the identity function as a *coercion*, one can restrict the different types of a term. Let's consider a particular PTS with (u, v) and (u, v') in the definition of its axioms:

1. u can be typed by v or v': $\emptyset \vdash u : v$ and $\emptyset \vdash u : v'$.
2. $M_1 \equiv id_v\ u$ can only be typed by v: If $\emptyset \vdash M_1 : T$, then $T =_\beta v$.
3. $M_2 \equiv id_{v'}\ u$ can only be typed by v': If $\emptyset \vdash M_2 : T$, then $T =_\beta v'$.
4. $\emptyset \vdash_e \Pi x^{M_1}.u =_\beta \Pi x^{M_2}.u : w$

By injectivity, we would be able to get $\emptyset \vdash_e M_1 =_\beta M_2 : s$ for some s, which is impossible.

Another kind of equality

Some presentations of programming languages and type theories based on the work of Martin-Löf are using another form of equality:

Another kind of equality

Some presentations of programming languages and type theories based on the work of Martin-Löf are using another form of equality:

- The equality for terms is the same : $\Gamma \vdash_e M =_\beta N : T$.

Another kind of equality

Some presentations of programming languages and type theories based on the work of Martin-Löf are using another form of equality:

- The equality for terms is the same : $\Gamma \vdash_e M =_\beta N : T$.
- The equality for types is weaker : $\Gamma \vdash_e A =_\beta B$.

Another kind of equality

Some presentations of programming languages and type theories based on the work of Martin-Löf are using another form of equality:

- The equality for terms is the same : $\Gamma \vdash_e M =_\beta N : T$.
- The equality for types is weaker : $\Gamma \vdash_e A =_\beta B$.

If we assume that this equality enjoys Π-injectivity, then it is enough to prove *Subject Reduction* for PTSe. Sadly, there is no known proof of that at the moment.

Is there another way to prove it ?

Another way to prove Subject Reduction for PTSe would be to use the Subject Reduction we have for PTSs. We need to prove some kind of equivalence between both systems.

Is there another way to prove it ?

Another way to prove Subject Reduction for PTSe would be to use the Subject Reduction we have for PTSs. We need to prove some kind of equivalence between both systems.

A more practical reason why we are looking for this equivalence is about *proof assistants*. Usually, the implementation is done with an untyped equality, whereas the consistency proof is done with a typed equality. Such an equivalence would bring closer the implementation from its theory.

The Big Question

Are PTSs and PTSe the same systems ?

[Geuvers93]

Easy part of the equivalence

We prove by mutual induction that

- If $\Gamma \vdash_e M : T$ then $\Gamma \vdash M : T$.
- If $\Gamma \vdash_e M =_\beta N : T$ then $\Gamma \vdash M : T$, $\Gamma \vdash N : T$ and $M =_\beta N$.
- If Γ_{wf_e} then Γ_{wf}.

Easy part of the equivalence

We prove by mutual induction that

- If $\Gamma \vdash_e M : T$ then $\Gamma \vdash M : T$.
- If $\Gamma \vdash_e M =_\beta N : T$ then $\Gamma \vdash M : T$, $\Gamma \vdash N : T$ and $M =_\beta N$.
- If Γ_{wf_e} then Γ_{wf}.

Here we just “lose” some information, nothing complicated.

Difficult part of the equivalence

The other way around needs a way to “type” a β-equivalence into a judgmental equality:

- If $\Gamma \vdash M : T$ then $\Gamma \vdash_e M : T$.
- If $\Gamma \vdash M : T$, $\Gamma \vdash N : T$ and $M =_\beta N$ then $\Gamma \vdash_e M =_\beta N : T$.
- If Γ_{wf} then Γ_{wf_e}.

Difficult part of the equivalence

The other way around needs a way to “type” a β-equivalence into a judgmental equality:

- If $\Gamma \vdash M : T$ then $\Gamma \vdash_e M : T$.
- If $\Gamma \vdash M : T$, $\Gamma \vdash N : T$ and $M =_\beta N$ then $\Gamma \vdash_e M =_\beta N : T$.
- If Γ_{wf} then Γ_{wf_e}.

Here, we need to find a way to type all the intermediate steps.

Difficult part of the equivalence

The other way around needs a way to “type” a β-equivalence into a judgmental equality:

- If $\Gamma \vdash M : T$ then $\Gamma \vdash_e M : T$.
- If $\Gamma \vdash M : T$, $\Gamma \vdash N : T$ and $M =_\beta N$ then $\Gamma \vdash_e M =_\beta N : T$.
- If Γ_{wf} then Γ_{wf_e}.

Here, we need to find a way to type all the intermediate steps.

But can we ?

How do we do this ?

$\Gamma \vdash M : T \qquad M \qquad =_\beta \qquad N \qquad \Gamma \vdash N : T$

How do we do this ?

$$
\begin{tikzcd}
\Gamma \vdash M : T & M \arrow[dr, twoheadrightarrow] & =_\beta & N \arrow[dl, twoheadrightarrow] & \Gamma \vdash N : T \\
\Gamma \vdash_e M : T & & P & & \Gamma \vdash_e N : T
\end{tikzcd}
$$

How do we do this ?

$$
\begin{tikzcd}
\Gamma \vdash M : T & M \arrow[dr, twoheadrightarrow] & =_\beta & N \arrow[dl, twoheadrightarrow] & \Gamma \vdash N : T \\
\Gamma \vdash_e M : T & & P & & \Gamma \vdash_e N : T
\end{tikzcd}
$$

- *P* is well-typed in PTS by *Subject Reduction.*

How do we do this ?

$$\begin{tikzcd}
\Gamma \vdash M : T & M \arrow[dr, twoheadrightarrow] & =_\beta & N \arrow[dl, twoheadrightarrow] & \Gamma \vdash N : T \\
\Gamma \vdash_e M : T & & P & & \Gamma \vdash_e N : T
\end{tikzcd}$$

- P is well-typed in PTS by *Subject Reduction.*
- Is P well-typed in PTSe ?

How do we do this ?

$$\Gamma \vdash M : T \qquad\qquad \Gamma \vdash N : T$$

$$\begin{tikzcd}
M \arrow[dr, twoheadrightarrow] & =_\beta & N \arrow[dl, twoheadrightarrow] \\
 & P &
\end{tikzcd}$$

$$\Gamma \vdash_e M : T \qquad\qquad \Gamma \vdash_e N : T$$

- P is well-typed in PTS by *Subject Reduction.*
- Is P well-typed in PTSe ?
- How do we type $M =_\beta P$ and $N =_\beta P$ in PTSe ?

Some partial solutions

- Early attempts to prove such an equivalence did not aim at the whole generality of PTSs, and were based on the construction of a model [Geuvers93,Goguen94].

Some partial solutions

- Early attempts to prove such an equivalence did not aim at the whole generality of PTSs, and were based on the construction of a model [Geuvers93,Goguen94].
- A first *syntactical* criterion was shown for a subclass of PTSs [Adams06] called *functional* PTSs, by adding annotations inside the syntax of terms.

Some partial solutions

- Early attempts to prove such an equivalence did not aim at the whole generality of PTSs, and were based on the construction of a model [Geuvers93,Goguen94].
- A first *syntactical* criterion was shown for a subclass of PTSs [Adams06] called *functional* PTSs, by adding annotations inside the syntax of terms.
- By using the same intermediate system, Herbelin and I extended this result to other subclasses of PTSs called *semi-full* and *full*.

Extension of the TPOSR solution

Adams introduced an additional annotation inside the applications:

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

Extension of the TPOSR solution

Adams introduced an additional annotation inside the applications:

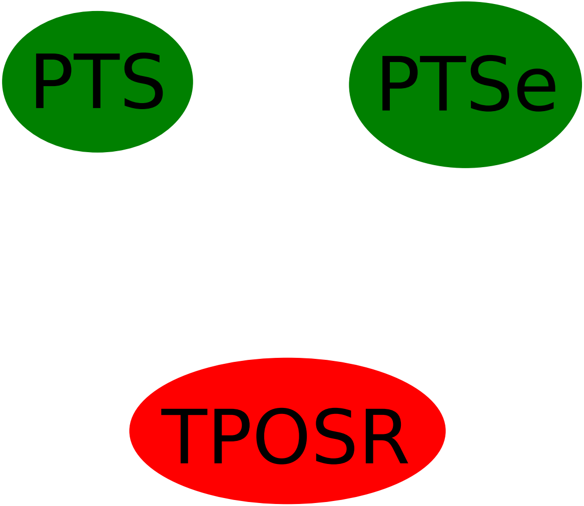
$$M, N, A, B \quad ::= \quad x \mid \lambda x^A.M \mid M_{(x)B}\ N \mid \Pi x^A.B \mid s$$

Also, its system called *Typed Parallel One Step Reduction* is no longer based on equality but on *reduction*:

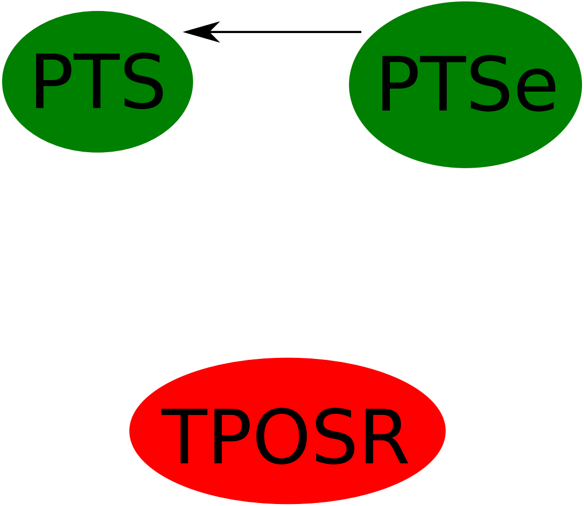
$$\frac{\Gamma \vdash M \rhd N : A \qquad \Gamma \vdash A \cong B : s}{\Gamma \vdash M \rhd N : B}$$

$$\frac{\begin{array}{c}\Gamma \vdash A \rhd A' : s \qquad \Gamma, x : A \vdash B \rhd B' : t \\ \Gamma, x : A \vdash M \rhd M' : B \qquad \Gamma \vdash N \rhd N' : A \qquad (s, t, u) \in \mathcal{R}el\end{array}}{\Gamma \vdash (\lambda x^A.M)_{(x)B} N \rhd M'[N'/x] : B[N/x]}$$

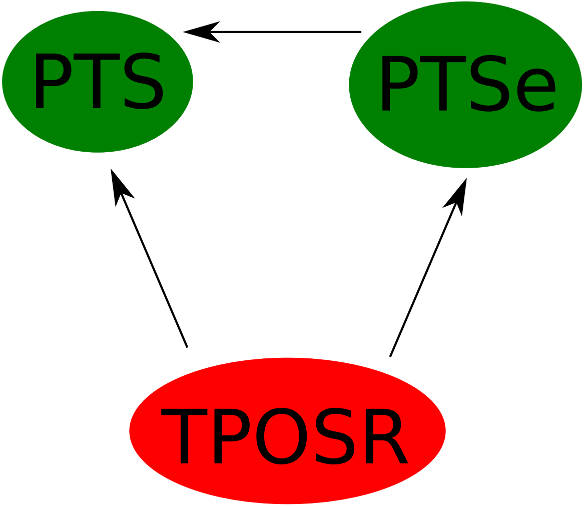
Summary of the proof



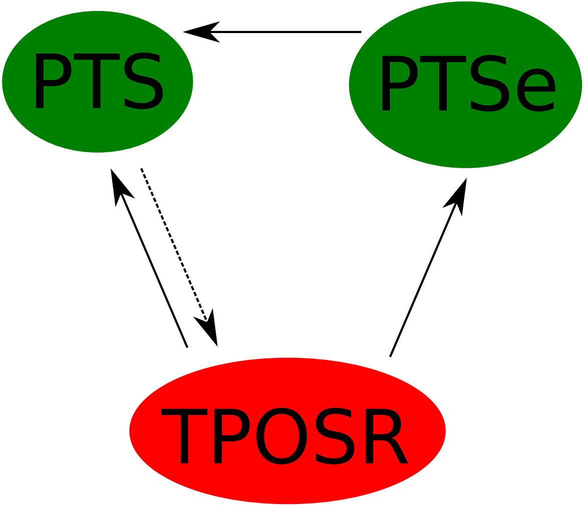
Summary of the proof



Summary of the proof



Summary of the proof



Road map to a proof of equivalence

The idea is to prove that:

- TPOSR's equality is *Confluent*.
- TPOSR's equality has *Injectivity of* Π*-types*.
- TPOSR has *Subject-Reduction*.
- TPOSR is equivalent to PTS and PTSe.

Road map to a proof of equivalence

The idea is to prove that:

- TPOSR's equality is *Confluent*.
- TPOSR's equality has *Injectivity of* Π*-types*.
- TPOSR has *Subject-Reduction*.
- TPOSR is equivalent to PTS and PTSe.

$\Uparrow$

tricky part

PTS with Annotated Typed Reduction

To achieve the equivalence for *all* PTSs, we need to improve TPOSR.
The idea came from [Streicher91], but was used for completeness results.

PTS with Annotated Typed Reduction

To achieve the equivalence for *all* PTSs, we need to improve TPOSR.
The idea came from [Streicher91], but was used for completeness results.

- Our idea is to extend the annotation on application: $M_{\Pi x^A.B}\ N$.

PTS with Annotated Typed Reduction

To achieve the equivalence for *all* PTSs, we need to improve TPOSR. The idea came from [Streicher91], but was used for completeness results.

- Our idea is to extend the annotation on application: $M_{\color{red}{\Pi x^A.B}}\ N$.
- And we have to change the typing rule to deal with this new annotation:

$$\frac{\Gamma \vdash M \rhd N : A \qquad \Gamma \vdash A \cong B : s}{\Gamma \vdash M \rhd N : B}$$

PTS with Annotated Typed Reduction

To achieve the equivalence for *all* PTSs, we need to improve TPOSR. The idea came from [Streicher91], but was used for completeness results.

- Our idea is to extend the annotation on application: $M_{\Pi x^A.B}\ N$.
- And we have to change the typing rule to deal with this new annotation:

$$\frac{\Gamma \vdash M \rhd N : A \qquad \Gamma \vdash A \cong B}{\Gamma \vdash M \rhd N : B}$$

PTS with Annotated Typed Reduction

To achieve the equivalence for *all* PTSs, we need to improve TPOSR. The idea came from [Streicher91], but was used for completeness results.

- Our idea is to extend the annotation on application: $M_{\Pi x^A . B}\ N$.
- And we have to change the typing rule to deal with this new annotation:

$$\frac{\Gamma \vdash M \rhd N : A \qquad \Gamma \vdash A \cong B}{\Gamma \vdash M \rhd N : B}$$

$$\frac{\begin{array}{cc} \ldots & \Gamma \vdash A \cong A' : s \\ \Gamma, x : A \vdash M \rhd M' : B & \Gamma \vdash N \rhd N' : A \end{array}}{\Gamma \vdash (\lambda x^A . M)_{\Pi x^{A'} . B} N \rhd M'[N'/x] : B[N/x]}$$

PTS with Annotated Typed Reduction

To achieve the equivalence for *all* PTSs, we need to improve TPOSR. The idea came from [Streicher91], but was used for completeness results.

- Our idea is to extend the annotation on application: $M_{\Pi x^A.B}\ N$.
- And we have to change the typing rule to deal with this new annotation:

$$\frac{\Gamma \vdash M \rhd N : A \qquad \Gamma \vdash A \cong B}{\Gamma \vdash M \rhd N : B}$$

$$\frac{\begin{array}{c}\dots \qquad \Gamma \vdash A_0 \rhd^+ A : s \qquad \Gamma \vdash A_0 \rhd^+ A' : s \\ \Gamma, x : A \vdash M \rhd M' : B \qquad \Gamma \vdash N \rhd N' : A\end{array}}{\Gamma \vdash (\lambda x^{A}.M)_{\Pi x^{A'}.B} N \rhd M'[N'/x] : B[N/x]}$$

Typed Confluence and Injectivity

Diamond property for PTS_{atr}

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

Typed Confluence and Injectivity

Diamond property for PTS_{atr}

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

$$
\begin{tikzcd}
 & M \arrow[dl] \arrow[dr] & \\
N \arrow[dr, dotted] & & P \arrow[dl, dotted] \\
 & Q &
\end{tikzcd}
$$

Typed Confluence and Injectivity

Diamond property for PTS_{atr}

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

The proof is easier than the proof for TPOSR because of the additional annotations. However, these annotations will give us extra work in the following properties.

As a direct consequence:

Π-Injectivity for PTS_{atr}

If $\Gamma \vdash \Pi x^A.B \cong \Pi x^C.D$ then $\Gamma \vdash A \cong C$ and $\Gamma, x : A \vdash B \cong D$.

Typed Subject Reduction and annotations

As we said before, the key point of the equivalence is the *Subject Reduction* of the typed system:

Typed Subject Reduction and annotations

As we said before, the key point of the equivalence is the *Subject Reduction* of the typed system:

Subject Reduction for PTS_{atr}

If $\Gamma \vdash M \rhd P : T$ and $M \to_\beta N$ then $\Gamma \vdash M \rhd^+ N : T$.

Typed Subject Reduction and annotations

As we said before, the key point of the equivalence is the *Subject Reduction* of the typed system:

Subject Reduction for PTS_{atr}

If $\Gamma \vdash M \rhd P : T$ and $M \to_\beta N$ then $\Gamma \vdash M \rhd^+ N : T$.

The proof is almost the same as the usual one for PTSs. Some additional work is required for the β case: we need to provide the A_0 that links both annotations.

$$\frac{\dots \qquad \Gamma \vdash A_0 \rhd^+ A : s \qquad \Gamma \vdash A_0 \rhd^+ A' : s}{\Gamma \vdash (\lambda x^A.M)_{\Pi x^{A'}.B} N \rhd M'[N'/x] : B[N/x]}$$

Equivalence between PTSs and PTS$_{atr}$

- It is easy to translate a PTS$_{atr}$ judgment into PTSs (same kind of erasure than PTSe).

Equivalence between PTSs and PTS$_{atr}$

- It is easy to translate a PTS$_{atr}$ judgment into PTSs (same kind of erasure than PTSe).
- However, from usual PTSs, we need to compute the additional annotations needed by PTS$_{atr}$:

Equivalence between PTSs and PTS$_{atr}$

- It is easy to translate a PTS$_{atr}$ judgment into PTSs (same kind of erasure than PTSe).
- However, from usual PTSs, we need to compute the additional annotations needed by PTS$_{atr}$:

From PTS to PTS$_{atr}$

If $\Gamma \vdash M : T$, then there is Γ^*, M^* and T^* such that $\Gamma^* \vdash M^* \rhd M^* : T^*$, where $|\Gamma^*| \equiv \Gamma$, $|M^*| \equiv M$ and $|T^*| \equiv T$.

Equivalence between PTSs and PTS$_{atr}$

- It is easy to translate a PTS$_{atr}$ judgment into PTSs (same kind of erasure than PTSe).
- However, from usual PTSs, we need to compute the additional annotations needed by PTS$_{atr}$:

From PTS to PTS$_{atr}$

If $\Gamma \vdash M : T$, then there is Γ^*, M^* and T^* such that $\Gamma^* \vdash M^* \rhd M^* : T^*$, where $|\Gamma^*| \equiv \Gamma$, $|M^*| \equiv M$ and $|T^*| \equiv T$.

- Thanks to Subject Reduction of PTS$_{atr}$, the conversion rule is no longer a problem, but we still have to compute some valid Γ^*, M^* and T^*.

Annotations and Typing

The proof is done by induction:

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

Annotations and Typing

The proof is done by induction:

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

By induction, we have:

- Γ_1, A_1 such that $\Gamma_1 \vdash A_1 \rhd A_1 : s$, $|\Gamma_1| \equiv \Gamma$ and $|A_1| \equiv A$.

Annotations and Typing

The proof is done by induction:

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

By induction, we have:

- Γ_1, A_1 such that $\Gamma_1 \vdash A_1 \rhd A_1 : s$, $|\Gamma_1| \equiv \Gamma$ and $|A_1| \equiv A$.
- Γ_2, A_2 and B_2 such that $\Gamma_2, x : A_2 \vdash B_2 \rhd B_2 : t$, $|\Gamma_2| \equiv \Gamma$, $|A_2| \equiv A$ and $|B_2| \equiv B$.

Annotations and Typing

The proof is done by induction:

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

By induction, we have:

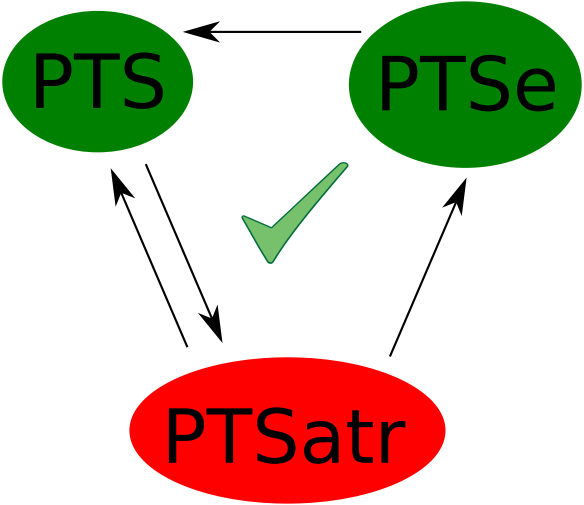
- Γ_1, A_1 such that $\Gamma_1 \vdash A_1 \rhd A_1 : s$, $|\Gamma_1| \equiv \Gamma$ and $|A_1| \equiv A$.
- Γ_2, A_2 and B_2 such that $\Gamma_2, x : A_2 \vdash B_2 \rhd B_2 : t$, $|\Gamma_2| \equiv \Gamma$, $|A_2| \equiv A$ and $|B_2| \equiv B$.
- We need a way to glue things together:

Erased Conversion

If $|A| \equiv |B|$, and if A and B are well-formed types in PTS_{atr}, then $\Gamma \vdash A \cong B$.

The proof of this lemma is very technical, and the most difficult proof of this thesis.

Complete Equivalence



Consequences of the equivalence

Complete Equivalence:

$$\begin{cases} \Gamma \vdash_e M : T & \textit{iff} \quad \Gamma \vdash M : T \\ \Gamma \vdash_e M =_\beta N : T & \textit{iff} \quad \Gamma \vdash M : T, \Gamma \vdash N : T \text{ and } M =_\beta N \\ \Gamma_{wf} & \textit{iff} \quad \Gamma_{wf_e} \end{cases}$$

Consequences of the equivalence

Complete Equivalence:

$$\begin{cases} \Gamma \vdash_e M : T & \textit{iff} \quad \Gamma \vdash M : T \\ \Gamma \vdash_e M =_\beta N : T & \textit{iff} \quad \Gamma \vdash M : T, \Gamma \vdash N : T \text{ and } M =_\beta N \\ \Gamma_{wf} & \textit{iff} \quad \Gamma_{wf_e} \end{cases}$$

Proving Subject Reduction for PTSe is now trivial:

- If $\Gamma \vdash_e M : T$ and $M \to_\beta N$, then $\Gamma \vdash M : T$.
- By Subject Reduction in PTS, $\Gamma \vdash N : T$, so $\Gamma \vdash_e N : T$.
- Once again, by equivalence, $\Gamma \vdash_e M =_\beta N : T$.

Consequences of the equivalence

Complete Equivalence:

$$\begin{cases} \Gamma \vdash_e M : T & \textit{iff} \quad \Gamma \vdash M : T \\ \Gamma \vdash_e M =_\beta N : T & \textit{iff} \quad \Gamma \vdash M : T, \Gamma \vdash N : T \text{ and } M =_\beta N \\ \Gamma_{wf} & \textit{iff} \quad \Gamma_{wf_e} \end{cases}$$

Proving Subject Reduction for PTSe is now trivial:

- If $\Gamma \vdash_e M : T$ and $M \to_\beta N$, then $\Gamma \vdash M : T$.
- By Subject Reduction in PTS, $\Gamma \vdash N : T$, so $\Gamma \vdash_e N : T$.
- Once again, by equivalence, $\Gamma \vdash_e M =_\beta N : T$.

Corollary: Weak Π-Injectivity

If $\Gamma \vdash_e \Pi x^A.B =_\beta \Pi x^C.D$ then $\Gamma \vdash_e A =_\beta C$ and $\Gamma, x : A \vdash_e B =_\beta D$.

By the way

Proved in Coq

Expansion Postponement: still stuck

Expansion Postponement is another hard problem regarding PTSs [Pollack92]. The idea is once again to modify the conversion rule to suit our needs.

Expansion Postponement: still stuck

Expansion Postponement is another hard problem regarding PTSs [Pollack92]. The idea is once again to modify the conversion rule to suit our needs.

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

$$\Updownarrow$$

$$\frac{\Gamma \vdash M : A \qquad A \to_\beta B}{\Gamma \vdash M : B} \quad \frac{\Gamma \vdash M : A \qquad B \to_\beta A \qquad \Gamma \vdash B : s}{\Gamma \vdash M : B}$$

Expansion Postponement: still stuck

Expansion Postponement is another hard problem regarding PTSs [Pollack92]. The idea is once again to modify the conversion rule to suit our needs.

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

$$\Updownarrow$$

$$\frac{\Gamma \vdash M : A \qquad A \to_\beta B}{\Gamma \vdash M : B} \quad \frac{\Gamma \vdash M : A \qquad B \to_\beta A \qquad \Gamma \vdash B : s}{\Gamma \vdash M : B}$$

Following ideas from [Lengrand06], we tried to switch to a view of PTSs based on *Sequent Calculus*, but failed at finding a final answer to this problem.

To infinity . . . and beyond !

In order to apply such a result to full scale type systems like the ones behind proof assistants, we need to extend the theory.

- Adding inductive types to have more usable datatypes.

To infinity . . . and beyond !

In order to apply such a result to full scale type systems like the ones behind proof assistants, we need to extend the theory.

- Adding inductive types to have more usable datatypes.
- Trying to add η-expansion to the conversion.

To infinity . . . and beyond !

In order to apply such a result to full scale type systems like the ones behind proof assistants, we need to extend the theory.

- Adding inductive types to have more usable datatypes.
- Trying to add η-expansion to the conversion.
- Adding universes *à la* Martin-Löf: towards CC_ω and *CIC*.

The Calculus of Constructions with Universes: CC_ω

We tried several different extensions in order to deal with subtyping in the CC_ω type system [Miquel01] but we didn't yet find a proper solution.

The Calculus of Constructions with Universes: CC_ω

We tried several different extensions in order to deal with subtyping in the CC_ω type system [Miquel01] but we didn't yet find a proper solution.

The two main issues we faced were quite different:

- By keeping the same annotation process, the *Erased Confluence* lemma is no longer true, so we can't annotate a PTS into PTS_{atr}.

The Calculus of Constructions with Universes: CC_ω

We tried several different extensions in order to deal with subtyping in the CC_ω type system [Miquel01] but we didn't yet find a proper solution.

The two main issues we faced were quite different:

- By keeping the same annotation process, the *Erased Confluence* lemma is no longer true, so we can't annotate a PTS into PTS_{atr}.
- All the attempts to fix the way we deal with the annotations have broken the *Church-Rosser* lemma.

We still need to find the right way to deal with universes and subtyping.

Conclusion

Implementation contributions:

- Extension of the formalization started by Barras of the meta theory of PTSs and PTSe by adding the full meta theory of PTS_{atr} and the complete proof of equivalence.
- Formalization of a part of [Lengrand06] about PTSs in *Sequent Calculus* with some extensions.

Theoretical contributions:

- A new system with typed reduction that enjoys all the good properties of usual PTSs, without relying on normalization.
- The right notion of equality at the level of types for PTSe, which enjoys the *Injectivity of* Π*-types*.
- A final answer to the link between PTS and PTSe: they are completely equivalent.

We finally have a **unified** theory of Pure Type Systems.