Pure Type Systems and Equality Checking

Vincent Siles

INRIA - PPS - Ecole Polytechnique

April 15th, 2010

Definitions

- PTSs are a way to have general results over families of type systems (System **F**, Calculus of Constructions, Simply-Typed λ-Calculus,...).

Definitions

- PTSs are a way to have general results over families of type systems (System **F**, Calculus of Constructions, Simply-Typed λ-Calculus,...).
- Terms and Contexts:

$$\begin{array}{ccc} A, B, M, N & ::= & s \mid x \mid M\ N \mid \lambda x^A.M \mid \Pi x^A.B \text{ (or } A \to B) \\ \Gamma & ::= & [\,] \mid \Gamma, x : A \end{array}$$

Definitions

- PTSs are a way to have general results over families of type systems (System **F**, Calculus of Constructions, Simply-Typed λ-Calculus,...).
- Terms and Contexts:

$$\begin{array}{ccc} A, B, M, N & ::= & s \mid x \mid M\ N \mid \lambda x^A.M \mid \Pi x^A.B \text{ (or } A \to B) \\ \Gamma & ::= & [\,] \mid \Gamma, x : A \end{array}$$

- The validity of typing judgments relies on two sets:
  - *Ax* is used to type sorts .
  - *Rel* is used to type functions (or Π-types).

Definitions

- PTSs are a way to have general results over families of type systems (System **F**, Calculus of Constructions, Simply-Typed λ-Calculus,...).
- Terms and Contexts:

$$\begin{array}{ccc} A, B, M, N & ::= & s \mid x \mid M\ N \mid \lambda x^A.M \mid \Pi x^A.B \text{ (or } A \rightarrow B) \\ \Gamma & ::= & [\,] \mid \Gamma, x : A \end{array}$$

- The validity of typing judgments relies on two sets:
  - *Ax* is used to type sorts .
  - *Rel* is used to type functions (or Π-types).
- Reduction :

$$(\lambda x^A.M)\ N \ \stackrel{\beta}{\rightarrow}\ M[N/x] + \text{congruences}$$

PTS typing rules

$$\frac{}{\emptyset_{wf}} \qquad \frac{\Gamma \vdash A : s \quad x \notin Dom(\Gamma)}{(\Gamma, x : A)_{wf}} \qquad \frac{\Gamma_{wf} \quad (s,t) \in \mathcal{A}x}{\Gamma \vdash s : t} \qquad \frac{\Gamma_{wf} \quad \Gamma(x) = A}{\Gamma \vdash x : A}$$

$$\frac{\begin{array}{cc}\Gamma \vdash A : s & \Gamma, x : A \vdash B : t \\ (s,t,u) \in \mathcal{R}el & \Gamma, x : A \vdash M : B\end{array}}{\Gamma \vdash \lambda x^A.M : \Pi x^A.B}$$

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

$$\frac{\Gamma \vdash M : \Pi x^A.B \qquad \Gamma \vdash N : A}{\Gamma \vdash MN : B[x/N]} \qquad \frac{\Gamma \vdash M : A \qquad A \stackrel{\beta}{\equiv} B \qquad \Gamma \vdash B : s}{\Gamma \vdash M : B}$$

Some known Type Systems

- Simply-Typed λ-Calculus:
  $S = \{\star, \Box\} \quad Ax = \{(\star, \Box)\} \quad Rel = \{(\star, \star, \star)\}$

Some known Type Systems

- Simply-Typed λ-Calculus:
  $S = \{\star, \square\} \quad Ax = \{(\star, \square)\} \quad Rel = \{(\star, \star, \star)\}$
- System **F**:
  $S = \{\star, \square\} \quad Ax = \{(\star, \square)\} \quad Rel = \{(\star, \star, \star), (\square, \star, \star)\}$

Some known Type Systems

- Simply-Typed λ-Calculus:
  $S = \{\star, \Box\} \quad Ax = \{(\star, \Box)\} \quad Rel = \{(\star, \star, \star)\}$
- System **F**:
  $S = \{\star, \Box\} \quad Ax = \{(\star, \Box)\} \quad Rel = \{(\star, \star, \star), (\Box, \star, \star)\}$
- Calculus of Constructions:
  $S = \{Prop, Type\} \quad Ax = \{(Prop, Type)\}$
  $Rel = \{(s, Prop, Prop), (s, Type, Type)\}$

Some special classes of PTS

- Functional: If $(s,t) \in \mathcal{A}x$ and $(s,t') \in \mathcal{A}x$ then $t = t'$.
  If $(s,t,u) \in \mathcal{R}el$ and $(s,t,u') \in \mathcal{R}el$ then $u = u'$.

Some special classes of PTS

- Functional: If $(s,t) \in \mathcal{A}x$ and $(s,t') \in \mathcal{A}x$ then $t = t'$.
  If $(s,t,u) \in \mathcal{R}el$ and $(s,t,u') \in \mathcal{R}el$ then $u = u'$.
  Those PTS enjoy the *Uniqueness of Type* property:

If $\Gamma \vdash M : A$ and $\Gamma \vdash M : B$ then $A \overset{\beta}{\equiv} B$.

Some special classes of PTS

- Functional: If $(s,t) \in \mathcal{A}x$ and $(s,t') \in \mathcal{A}x$ then $t = t'$.
  If $(s,t,u) \in \mathcal{R}el$ and $(s,t,u') \in \mathcal{R}el$ then $u = u'$.
  Those PTS enjoy the *Uniqueness of Type* property:

If $\Gamma \vdash M : A$ and $\Gamma \vdash M : B$ then $A \overset{\beta}{\equiv} B$.

- Full: for all s, t, there is a u such that $(s,t,u) \in \mathcal{R}el$.

Some special classes of PTS

- Functional: If $(s, t) \in \mathcal{A}x$ and $(s, t') \in \mathcal{A}x$ then $t = t'$.
  If $(s, t, u) \in \mathcal{R}el$ and $(s, t, u') \in \mathcal{R}el$ then $u = u'$.
  Those PTS enjoy the *Uniqueness of Type* property:

If $\Gamma \vdash M : A$ and $\Gamma \vdash M : B$ then $A \overset{\beta}{\equiv} B$.

- Full: for all s, t, there is a u such that $(s, t, u) \in \mathcal{R}el$.
  $\hookrightarrow$ In those PTS, “any” products is typable.

Some special classes of PTS

- Functional: If $(s, t) \in \mathcal{A}x$ and $(s, t') \in \mathcal{A}x$ then $t = t'$.
  If $(s, t, u) \in \mathcal{R}el$ and $(s, t, u') \in \mathcal{R}el$ then $u = u'$.
  Those PTS enjoy the *Uniqueness of Type* property:

If $\Gamma \vdash M : A$ and $\Gamma \vdash M : B$ then $A \overset{\beta}{\equiv} B$.

- Full: for all s, t, there is a u such that $(s, t, u) \in \mathcal{R}el$.
  $\hookrightarrow$ In those PTS, "any" products is typable.
- Semi-full PTS: If $(s, t, u) \in \mathcal{R}el$ then for all t', there is u' such that $(s, t', u') \in \mathcal{R}el$.

Some special classes of PTS

- Functional: If $(s,t) \in \mathcal{A}x$ and $(s,t') \in \mathcal{A}x$ then $t = t'$.
  If $(s,t,u) \in \mathcal{R}el$ and $(s,t,u') \in \mathcal{R}el$ then $u = u'$.
  Those PTS enjoy the *Uniqueness of Type* property:

If $\Gamma \vdash M : A$ and $\Gamma \vdash M : B$ then $A \overset{\beta}{\equiv} B$.

- Full: for all s, t, there is a u such that $(s,t,u) \in \mathcal{R}el$.
  $\hookrightarrow$ In those PTS, "any" products is typable.
- Semi-full PTS: If $(s,t,u) \in \mathcal{R}el$ then for all t', there is u' such that $(s,t',u') \in \mathcal{R}el$.
  $\hookrightarrow$ If the product $\Pi x^A.B$ is typable, then for any B' well-typed, $\Pi x^A.B'$ is also well-typed (or Π-functionality).

Facts about PTS

- Inversion lemmas :

e.g. if $\Gamma \vdash \lambda x^A.M : T$ then there are s, t, u and B such that

- $(s, t, u) \in \mathcal{R}el$, $T \overset{\beta}{\equiv} \Pi x^A.B$
- $\Gamma \vdash A : s$ and $\Gamma, x : A \vdash B : t$ and $\Gamma, x : A \vdash M : B$.

Facts about PTS

- Inversion lemmas :

e.g. if $\Gamma \vdash \lambda x^A.M : T$ then there are s, t, u and B such that

- $(s, t, u) \in \mathcal{R}el,\ T \overset{\beta}{\equiv} \Pi x^A.B$
- $\Gamma \vdash A : s$ and $\Gamma, x : A \vdash B : t$ and $\Gamma, x : A \vdash M : B$.

- Correctness of types :

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

Facts about PTS

- Inversion lemmas :

e.g. if $\Gamma \vdash \lambda x^A.M : T$ then there are s, t, u and B such that

- $(s,t,u) \in \mathcal{R}el$, $T \overset{\beta}{\equiv} \Pi x^A.B$
- $\Gamma \vdash A : s$ and $\Gamma, x : A \vdash B : t$ and $\Gamma, x : A \vdash M : B$.

- Correctness of types :

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

- Injectivity of Π-types:

If $\Pi x^A.B \overset{\beta}{\equiv} \Pi x^C.D$ then $A \overset{\beta}{\equiv} C$ and $B \overset{\beta}{\equiv} D$.

Facts about PTS

- Inversion lemmas :

e.g. if $\Gamma \vdash \lambda x^A.M : T$ then there are s, t, u and B such that

- $(s,t,u) \in \mathcal{R}el$, $T \stackrel{\beta}{\equiv} \Pi x^A.B$
- $\Gamma \vdash A : s$ and $\Gamma, x : A \vdash B : t$ and $\Gamma, x : A \vdash M : B$.

- Correctness of types :

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

- Injectivity of Π-types:

If $\Pi x^A.B \stackrel{\beta}{\equiv} \Pi x^C.D$ then $A \stackrel{\beta}{\equiv} C$ and $B \stackrel{\beta}{\equiv} D$.

- Subject Reduction:

If $\Gamma \vdash M : T$ and $M \stackrel{\beta}{\rightarrow} M'$ then $\Gamma \vdash M' : T$.

Shape of types in PTS

In 1993, Jutting did a deep study about the types of terms in PTS:

- Terms are classified in two families *Tv* and *Ts*:

Shape of types in PTS

In 1993, Jutting did a deep study about the types of terms in PTS:

- Terms are classified in two families Tv and Ts:
  - $\forall v \in V$ then $v \in Tv$
  - if $M \in Tv, MN \in Tv$ and $\lambda x^A.M \in Tv$

Shape of types in PTS

In 1993, Jutting did a deep study about the types of terms in PTS:

- Terms are classified in two families Tv and Ts:
  - $\forall v \in V$ then $v \in Tv$
  - if $M \in Tv, MN \in Tv$ and $\lambda x^A.M \in Tv$

  - $\forall s \in S, s \in Ts$
  - $\forall A, B, \Pi x^A.B \in Ts$
  - if $M \in Ts, MN \in Ts$ and $\lambda x^A.M \in Ts$

Shape of types in PTS

In 1993, Jutting did a deep study about the types of terms in PTS:

- Terms are classified in two families Tv and Ts:
  - $\forall v \in V$ then $v \in Tv$
  - if $M \in Tv$, $MN \in Tv$ and $\lambda x^A.M \in Tv$

  - $\forall s \in S, s \in Ts$
  - $\forall A, B, \Pi x^A.B \in Ts$
  - if $M \in Ts$, $MN \in Ts$ and $\lambda x^A.M \in Ts$

  - if $M \in Tv$,$\Gamma \vdash M : A$ and $\Gamma \vdash M : B$, then $A \overset{\beta}{\equiv} B$.
  - if $M \in Ts$,$\Gamma \vdash M : A$ and $\Gamma \vdash M : B$, then $A \overset{\beta}{\twoheadrightarrow} \Pi x_1^{U_1} \dots x_n^{U_n}.s$ and $B \overset{\beta}{\twoheadrightarrow} \Pi x_1^{U_1} \dots x_n^{U_n}.t$.

Why do we want a typed equality ?

- In the conversion rules the intermediate steps are not checked.

$$\frac{\Gamma \vdash M : A \qquad A \overset{\beta}{\equiv} B \qquad \Gamma \vdash B : s}{\Gamma \vdash M : B}$$

Why do we want a typed equality ?

- In the conversion rules the intermediate steps are not checked.

$$\frac{\Gamma \vdash M : A \qquad A \overset{\beta}{\equiv} B \qquad \Gamma \vdash B : s}{\Gamma \vdash M : B}$$

- β-equality is all about *program computation*, where types are useless.

Why do we want a typed equality ?

- In the conversion rules the intermediate steps are not checked.

$$\frac{\Gamma \vdash M : A \qquad A \overset{\beta}{\equiv} B \qquad \Gamma \vdash B : s}{\Gamma \vdash M : B}$$

- β-equality is all about *program computation*, where types are useless.
- Other kind of equalities may depend on types (η-expansion, external axioms).

Why do we want a typed equality ?

- In the conversion rules the intermediate steps are not checked.

$$\frac{\Gamma \vdash M : A \qquad A \overset{\beta}{\equiv} B \qquad \Gamma \vdash B : s}{\Gamma \vdash M : B}$$

- β-equality is all about *program computation*, where types are useless.
- Other kind of equalities may depend on types (η-expansion, external axioms).
- So, what if we check each conversion step during conversion ?

Why do we want a typed equality ?

- In the conversion rules the intermediate steps are not checked.

$$\frac{\Gamma \vdash M : A \qquad A \overset{\beta}{\equiv} B \qquad \Gamma \vdash B : s}{\Gamma \vdash M : B}$$

- β-equality is all about *program computation*, where types are useless.
- Other kind of equalities may depend on types (η-expansion, external axioms).
- So, what if we check each conversion step during conversion ?

$\hookrightarrow$ all this lead to the definition of PTS *with Judgmental Equality*.

PTSe typing rules (1)

$$\frac{}{\emptyset_{wf_e}} \qquad \frac{\Gamma \vdash_e A : s \quad x \notin Dom(\Gamma)}{(\Gamma, x : A)_{wf_e}} \qquad \frac{\Gamma_{wf_e} \quad (s,t) \in \mathcal{A}x}{\Gamma \vdash_e s : t} \qquad \frac{\Gamma_{wf_e} \quad \Gamma(x) = A}{\Gamma \vdash_e x : A}$$

$$\frac{\begin{array}{cc}\Gamma \vdash_e A : s & \Gamma, x : A \vdash_e B : t \\ (s,t,u) \in \mathcal{R}el & \Gamma, x : A \vdash_e M : B\end{array}}{\Gamma \vdash_e \lambda x^A.M : \Pi x^A.B}$$

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

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

PTSe typing rules (1)

$$\frac{}{\emptyset_{wf_e}} \qquad \frac{\Gamma \vdash_e A : s \quad x \notin Dom(\Gamma)}{(\Gamma, x : A)_{wf_e}} \qquad \frac{\Gamma_{wf_e} \quad (s,t) \in \mathcal{A}x}{\Gamma \vdash_e s : t} \qquad \frac{\Gamma_{wf_e} \quad \Gamma(x) = A}{\Gamma \vdash_e x : A}$$

$$\frac{\begin{array}{cc} \Gamma \vdash_e A : s & \Gamma, x : A \vdash_e B : t \\ (s,t,u) \in \mathcal{R}el & \Gamma, x : A \vdash_e M : B \end{array}}{\Gamma \vdash_e \lambda x^A.M : \Pi x^A.B}$$

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

$$\frac{\Gamma \vdash_e M : \Pi x^A.B \qquad \Gamma \vdash_e N : A}{\Gamma \vdash_e MN : B[x/N]} \qquad \frac{\Gamma \vdash_e M : A \qquad \boxed{\Gamma \vdash_e A = B : s}}{\Gamma \vdash_e M : B}$$

PTSe typing rules (2)

$$\frac{\Gamma_{wf_e} \qquad (s,t) \in \mathcal{A}x}{\Gamma \vdash_e s = s : t} \qquad \frac{\Gamma_{wf_e} \qquad \Gamma(x) = A}{\Gamma \vdash_e x = x : A}$$

$$\frac{\Gamma \vdash_e M = M' : \Pi x^A.B \qquad \Gamma \vdash_e N = N' : A}{\Gamma \vdash_e MN = M'N' : B[x/N]}$$

$$\frac{\Gamma \vdash_e A = A' : s \qquad \Gamma, x : A \vdash_e B = B' : t \qquad (s,t,u) \in \mathcal{R}el}{\Gamma \vdash_e \Pi x^A.B = \Pi x^{A'}.B' : u}$$

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

PTSe typing rules (3)

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

$$\frac{\Gamma \vdash_e M : A}{\Gamma \vdash_e M = M : A} \qquad \frac{\Gamma \vdash_e M = N : A}{\Gamma \vdash_e N = M : A} \qquad \frac{\Gamma \vdash_e M = N : A \quad \Gamma \vdash_e N = P : A}{\Gamma \vdash_e M = P : A}$$

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

The Big Question

Are both systems the same ?

Easy part of the equivalence

We prove by mutual induction that

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

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

Hard Part

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 $M \overset{\beta}{\equiv} N$ then $\Gamma \vdash_e M = N : T$.
- If Γ_{wf} then Γ_{wf_e}.

Hard Part

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 $M \stackrel{\beta}{\equiv} N$ then $\Gamma \vdash_e M = N : T$.
- If Γ_{wf} then Γ_{wf_e}.

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

Hard Part

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 $M \overset{\beta}{\equiv} N$ then $\Gamma \vdash_e M = N : T$.
- 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 ?

$$\Gamma \vdash M : T \qquad M \qquad \stackrel{\beta}{\equiv} \qquad N \qquad \Gamma \vdash N : T$$

How do we do this ?

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

How do we do this ?

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

- P is welltyped in PTS by *Subject Reduction*.

How do we do this ?

$$
\begin{tikzcd}
\Gamma \vdash M : T & M \arrow[dr, two heads] & \overset{\beta}{\equiv} & N \arrow[dl, two heads] & \Gamma \vdash N : T \\
\Gamma \vdash_e M : T & & P & & \Gamma \vdash_e N : T
\end{tikzcd}
$$

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

How do we do this ?

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

- P is welltyped in PTS by *Subject Reduction.*
- Is P welltyped in PTSe ?
- How do we type $M = P$ and $N = P$ in PTSe ?

The need of Subject Reduction

To do so, we need to prove that PTSe have the *Subject Reduction* property:

Subject Reduction:

If $\Gamma \vdash_e M : T$ and $M \overset{\beta}{\twoheadrightarrow} N$, then $\Gamma \vdash_e M = N : T$.

The need of Subject Reduction

To do so, we need to prove that PTSe have the *Subject Reduction* property:

Subject Reduction:

If $\Gamma \vdash_e M : T$ and $M \overset{\beta}{\twoheadrightarrow} N$, then $\Gamma \vdash_e M = N : T$.

But to prove this, we need Π-injectivity, which is still an open question for PTSe since it relies on *Confluency*,

The need of Subject Reduction

To do so, we need to prove that PTSe have the *Subject Reduction* property:

Subject Reduction:

If $\Gamma \vdash_e M : T$ and $M \overset{\beta}{\twoheadrightarrow} N$, then $\Gamma \vdash_e M = N : T$.

But to prove this, we need Π-injectivity, which is still an open question for PTSe since it relies on *Confluency*, which relies on *Subject Reduction*,

The need of Subject Reduction

To do so, we need to prove that PTSe have the *Subject Reduction* property:

Subject Reduction:

If $\Gamma \vdash_e M : T$ and $M \overset{\beta}{\twoheadrightarrow} N$, then $\Gamma \vdash_e M = N : T$.

But to prove this, we need Π-injectivity, which is still an open question for PTSe since it relies on *Confluency*, which relies on *Subject Reduction*, which relies on Π-injectivity,

The need of Subject Reduction

To do so, we need to prove that PTSe have the *Subject Reduction* property:

Subject Reduction:

If $\Gamma \vdash_e M : T$ and $M \overset{\beta}{\twoheadrightarrow} N$, then $\Gamma \vdash_e M = N : T$.

But to prove this, we need Π-injectivity, which is still an open question for PTSe since it relies on *Confluency*, which relies on *Subject Reduction*, which relies on Π-injectivity, which relies on ...

Current status of the equivalence

We only have some partials results:

- for functional PTS : R. Adams [06] “Pure Type Systems with Judgmental Equality”.

Current status of the equivalence

We only have some partials results:

- for functional PTS : R. Adams [06] “Pure Type Systems with Judgmental Equality”.
- for semi-full and full PTS : V. Siles and H. Herbelin [10] “Equality is typable in Semi-Full Pure Type Systems”.

Current status of the equivalence

We only have some partials results:

- for functional PTS : R. Adams [06] “Pure Type Systems with Judgmental Equality”.
- for semi-full and full PTS : V. Siles and H. Herbelin [10] “Equality is typable in Semi-Full Pure Type Systems”.
- But the question is still open for general PTS !

Adams' approch

- In order to break the loop, Adams defined a typed version of the usual parallel β-reduction, called *Typed Parallel One Step Reduction* (TPOSR).

Adams' approch

- In order to break the loop, Adams defined a typed version of the usual parallel β-reduction, called *Typed Parallel One Step Reduction* (TPOSR).
- His goal was to prove the *Diamond Property* for TPOSR, which leads to the addition of annotations on applications.

Adams' approch

- In order to break the loop, Adams defined a typed version of the usual parallel β-reduction, called *Typed Parallel One Step Reduction* (TPOSR).
- His goal was to prove the *Diamond Property* for TPOSR, which leads to the addition of annotations on applications.
- The main scheme is:

Adams' approch

- In order to break the loop, Adams defined a typed version of the usual parallel β-reduction, called *Typed Parallel One Step Reduction* (TPOSR).
- His goal was to prove the *Diamond Property* for TPOSR, which leads to the addition of annotations on applications.
- The main scheme is:
  - Prove that TPOSR is *Church-Rosser*.

Adams' approch

- In order to break the loop, Adams defined a typed version of the usual parallel β-reduction, called *Typed Parallel One Step Reduction* (TPOSR).
- His goal was to prove the *Diamond Property* for TPOSR, which leads to the addition of annotations on applications.
- The main scheme is:
  - Prove that TPOSR is *Church-Rosser*.
  - Prove that TPOSR has *Subject-Reduction*.

Adams' approch

- In order to break the loop, Adams defined a typed version of the usual parallel β-reduction, called *Typed Parallel One Step Reduction* (TPOSR).
- His goal was to prove the *Diamond Property* for TPOSR, which leads to the addition of annotations on applications.
- The main scheme is:
  - Prove that TPOSR is *Church-Rosser*.
  - Prove that TPOSR has *Subject-Reduction*.
  - Prove that TPOSR is equivalent to PTS and PTSe.

TPOSR typing rules (1)

$$\frac{}{\emptyset_{wf}} \qquad \frac{\Gamma \vdash A \rhd A' : s \quad x \notin Dom(\Gamma)}{(\Gamma, x : A)_{wf}} \qquad \frac{\Gamma_{wf} \quad (s,t) \in \mathcal{A}x}{\Gamma \vdash s \rhd s : t} \qquad \frac{\Gamma_{wf} \quad \Gamma(x) = A}{\Gamma \vdash x \rhd x : A}$$

$$\frac{\Gamma \vdash A \rhd A' : s \qquad \Gamma, x : A \vdash B \rhd B' : t \qquad (s,t,u) \in \mathcal{R}el}{\Gamma \vdash \Pi x^{A}.B \rhd \Pi x^{A'}.B' : u}$$

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

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

TPOSR typing rules (1)

$$\frac{}{\emptyset_{wf}} \qquad \frac{\Gamma \vdash A \rhd A' : s \quad x \notin Dom(\Gamma)}{(\Gamma, x : A)_{wf}} \qquad \frac{\Gamma_{wf} \quad (s,t) \in \mathcal{A}x}{\Gamma \vdash s \rhd s : t} \qquad \frac{\Gamma_{wf} \quad \Gamma(x) = A}{\Gamma \vdash x \rhd x : A}$$

$$\frac{\Gamma \vdash A \rhd A' : s \qquad \Gamma, x : A \vdash B \rhd B' : t \qquad (s,t,u) \in \mathcal{R}el}{\Gamma \vdash \Pi x^{A}.B \rhd \Pi x^{A'}.B' : u}$$

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

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

TPOSR typing rules (2)

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

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

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

$$\frac{\Gamma \vdash M \rhd N : s}{\Gamma \vdash M \equiv N} \qquad \frac{\Gamma \vdash M \equiv N}{\Gamma \vdash N \equiv M} \qquad \frac{\Gamma \vdash M \equiv N \qquad \Gamma \vdash N \equiv P}{\Gamma \vdash M \equiv P}$$

TPOSR typing rules (2)

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

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

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

$$\frac{\Gamma \vdash M \rhd N : s}{\Gamma \vdash M \equiv N} \qquad \frac{\Gamma \vdash M \equiv N}{\Gamma \vdash N \equiv M} \qquad \frac{\Gamma \vdash M \equiv N \qquad \Gamma \vdash N \equiv P}{\Gamma \vdash M \equiv P}$$

We do not keep track of the sort (it requires *Type Uniqueness*).

From TPOSR to PTS and PTSe

Let's consider the $|\ |$ function that removes all annotations on applications, we can easily prove the following lemmas:

From TPOSR to PTS

If $\Gamma \vdash M \rhd N : T$ then $|\Gamma| \vdash |M| : |T|$ $|\Gamma| \vdash |N| : |T|$ and $|M| \stackrel{\beta_{//}}{\longrightarrow} |N|$.

From TPOSR to PTSe

If $\Gamma \vdash M \rhd N : T$ then $|\Gamma| \vdash |M| = |N| : |T|$.

From TPOSR to PTS and PTSe

Let's consider the $|\ |$ function that removes all annotations on applications, we can easily prove the following lemmas:

From TPOSR to PTS

If $\Gamma \vdash M \rhd N : T$ then $|\Gamma| \vdash |M| : |T|$ $|\Gamma| \vdash |N| : |T|$ and $|M| \stackrel{\beta_{//}}{\longrightarrow} |N|$.

From TPOSR to PTSe

If $\Gamma \vdash M \rhd N : T$ then $|\Gamma| \vdash |M| = |N| : |T|$.

As easy as before by induction, we just remove some information in the derivations.

First step: Church-Rosser

To prove the TPOSR is *Church-Rosser*, we will prove that the *Diamond Property* holds for TPOSR.

First step: Church-Rosser

To prove the TPOSR is *Church-Rosser*, we will prove that the *Diamond Property* holds for TPOSR.

Diamond Property

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

The main issues are the critical pairs involving the application rules: we are unable to apply some induction hypothesis

First step: Church-Rosser

To prove the TPOSR is *Church-Rosser*, we will prove that the *Diamond Property* holds for TPOSR.

Diamond Property

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

The main issues are the critical pairs involving the application rules: we are unable to apply some induction hypothesis

- the induction hypothesis over B requires a context "$\Gamma, x : A$"
- we only have an hypothesis "$\Gamma, x : C \vdash B \rhd B' : s$"
- but we have some informations that may link A to C...

First step: Church-Rosser

To prove the TPOSR is *Church-Rosser*, we will prove that the *Diamond Property* holds for TPOSR.

Diamond Property

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

The main issues are the critical pairs involving the application rules: we are unable to apply some induction hypothesis

- the induction hypothesis over B requires a context "$\Gamma, x : A$"
- we only have an hypothesis "$\Gamma, x : C \vdash B \rhd B' : s$"
- but we have some informations that may link A to C...

$\hookrightarrow$ So we need a way to equal A and C.

Functional vs Semi-Full

- For any functional TPOSR system, *Uniqueness of Types* holds, so we can prove that $\Gamma \vdash A \equiv C$.

Functional vs Semi-Full

- For any functional TPOSR system, *Uniqueness of Types* holds, so we can prove that $\Gamma \vdash A \equiv C$.
- The *Shape of Types* property of PTS can be extended to any semi-full TPOSR (we need the functionality of Π to prove it).

Functional vs Semi-Full

- For any functional TPOSR system, *Uniqueness of Types* holds, so we can prove that $\Gamma \vdash A \equiv C$.
- The *Shape of Types* property of PTS can be extended to any semi-full TPOSR (we need the functionality of Π to prove it).

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 $\Gamma \vdash A \equiv \Pi x_1^{U_1} ... x_n^{U_n}.s$ and $\Gamma \vdash B \equiv \Pi x_1^{U_1} ... x_n^{U_n}.t$

Back to the Untyped World

- Goal: Prove $\Gamma \vdash A \equiv C$ valid.

Back to the Untyped World

- Goal: Prove $\Gamma \vdash A \equiv C$ valid.
- Useful hypothesis:
  - $\Gamma \vdash N' \triangleright N''' : A$
  - $\Gamma \vdash N'' \triangleright N''' : C$
  - $\Gamma \vdash \Pi x^{A}.B \equiv \Pi x^{C}.B$

Back to the Untyped World

- Goal: Prove $\Gamma \vdash A \equiv C$ valid.
- Useful hypothesis:
  - $\Gamma \vdash N' \triangleright N''' : A$
  - $\Gamma \vdash N'' \triangleright N''' : C$
  - $\Gamma \vdash \Pi x^A.B \equiv \Pi x^C.B$

By applying the previous lemma to N''':

Back to the Untyped World

- Goal: Prove $\Gamma \vdash A \equiv C$ valid.
- Useful hypothesis:
  - $\Gamma \vdash N' \rhd N''' : A$
  - $\Gamma \vdash N'' \rhd N''' : C$
  - $\Gamma \vdash \Pi x^A.B \equiv \Pi x^C.B$

By applying the previous lemma to N''':

- first case : $\Gamma \vdash A \equiv C$
- second case : A and C only differ by their last sort s and t

Back to the Untyped World, second case

If we erase all the equalities we have so far, by *untyped Confluence* we can conclude that:

- $\Pi x^{|A|}.|B| \overset{\beta}{\equiv} \Pi x^{|C|}.|B|$.

Back to the Untyped World, second case

If we erase all the equalities we have so far, by *untyped Confluence* we can conclude that:

- $\Pi x^{|A|}.|B| \overset{\beta}{\equiv} \Pi x^{|C|}.|B|$.
- $\Longrightarrow |A| \overset{\beta}{\equiv} |C|$ by *untyped* Π*-injectivity*.

Back to the Untyped World, second case

If we erase all the equalities we have so far, by *untyped Confluence* we can conclude that:

- $\Pi x^{|A|}.|B| \stackrel{\beta}{\equiv} \Pi x^{|C|}.|B|$.
- $\Longrightarrow |A| \stackrel{\beta}{\equiv} |C|$ by *untyped* Π*-injectivity*.
- $\Longrightarrow \Pi x_1^{|U_1|}...x_n^{|U_n|}.s \stackrel{\beta}{\equiv} \Pi x_1^{|U_1|}...x_n^{|U_n|}.t$ by transitivity.

Back to the Untyped World, second case

If we erase all the equalities we have so far, by *untyped Confluence* we can conclude that:

- $\Pi x^{|A|}.|B| \overset{\beta}{\equiv} \Pi x^{|C|}.|B|$.
- $\Longrightarrow |A| \overset{\beta}{\equiv} |C|$ by *untyped* Π*-injectivity*.
- $\Longrightarrow \Pi x_1^{|U_1|} \dots x_n^{|U_n|}.s \overset{\beta}{\equiv} \Pi x_1^{|U_1|} \dots x_n^{|U_n|}.t$ by transitivity.
- $\Longrightarrow s = t$ by *untyped Confluence*.

Back to the Untyped World, second case

If we erase all the equalities we have so far, by *untyped Confluence* we can conclude that:

- $\Pi x^{|A|}.|B| \overset{\beta}{\equiv} \Pi x^{|C|}.|B|$.
- $\Longrightarrow |A| \overset{\beta}{\equiv} |C|$ by *untyped* Π*-injectivity*.
- $\Longrightarrow \Pi x_1^{|U_1|} \ldots x_n^{|U_n|}.s \overset{\beta}{\equiv} \Pi x_1^{|U_1|} \ldots x_n^{|U_n|}.t$ by transitivity.
- $\Longrightarrow s = t$ by *untyped Confluence*.

by transitivity, we finally have $\Gamma \vdash A \equiv C$.

Back to the Untyped World, second case

If we erase all the equalities we have so far, by *untyped Confluence* we can conclude that:

- $\Pi x^{|A|}.|B| \overset{\beta}{\equiv} \Pi x^{|C|}.|B|$.
- $\Longrightarrow |A| \overset{\beta}{\equiv} |C|$ by *untyped* Π*-injectivity*.
- $\Longrightarrow \Pi x_1^{|U_1|} \dots x_n^{|U_n|}.s \overset{\beta}{\equiv} \Pi x_1^{|U_1|} \dots x_n^{|U_n|}.t$ by transitivity.
- $\Longrightarrow s = t$ by *untyped Confluence*.

by transitivity, we finally have $\Gamma \vdash A \equiv C$.

With this, we can now finish to prove everything up to *Subject Reduction*

Validity of Annotations

To close the equivalence, we need to prove that the additional annotations on applications did not change the typing system, that is:

Validity of Annotations

If $\Gamma \vdash M : T$, then $\Gamma^* \vdash M^* \rhd M^* : T^*$
(for all Γ^*, M^*, T^* such than $|\Gamma^*| = \Gamma$, $|M^*| = M$ and $|T^*| = T$).

Validity of Annotations

To close the equivalence, we need to prove that the additional annotations on applications did not change the typing system, that is:

Validity of Annotations

If $\Gamma \vdash M : T$, then $\Gamma^* \vdash M^* \rhd M^* : T^*$
(for all Γ^*, M^*, T^* such than $|\Gamma^*| = \Gamma$, $|M^*| = M$ and $|T^*| = T$).

Since there are several ways to annotate a term, the induction can be quite tricky without the following lemma:

Erased Context Conversion

If $\Gamma_1 \vdash M \rhd N : A$, $|\Gamma_1| = |\Gamma_2|$ and $\Gamma_{2\ wf}$, then $\Gamma_2 \vdash M \rhd N : A$.

Erased Conversion: the second pitfall

To prove this conversion lemma, we need a more general lemma which is easily done for functional PTS, but strangely hard for semi-full:

Erased Conversion: the second pitfall

To prove this conversion lemma, we need a more general lemma which is easily done for functional PTS, but strangely hard for semi-full:

Erased Confluence

If $\Gamma \vdash M \rhd ? : S$, $\Gamma \vdash N \rhd ? : T$ and $|M| = |N|$, then there is P such that:

- $\Gamma \vdash M \rhd^+ P : S$
- $\Gamma \vdash N \rhd^+ P : T$

Erased Conversion: the second pitfall

To prove this conversion lemma, we need a more general lemma which is easily done for functional PTS, but strangely hard for semi-full:

Erased Confluence

If $\Gamma \vdash M \rhd ? : S$, $\Gamma \vdash N \rhd ? : T$ and $|M| = |N|$, then there is P such that:

- $\Gamma \vdash M \rhd^+ P : S$
- $\Gamma \vdash N \rhd^+ P : T$

By induction, all the cases are trivial but the application one

Erased Conversion: the second pitfall

To prove this conversion lemma, we need a more general lemma which is easily done for functional PTS, but strangely hard for semi-full:

Erased Confluence

If $\Gamma \vdash M \rhd ? : S$, $\Gamma \vdash N \rhd ? : T$ and $|M| = |N|$, then there is P such that:

- $\Gamma \vdash M \rhd^+ P : S$
- $\Gamma \vdash N \rhd^+ P : T$

By induction, all the cases are trivial but the application one

$$\begin{array}{ll}
|M| = |M'| & |N| = |N'| \\
\Gamma \vdash M \rhd^+ M_0 : \Pi x^A.B & \Gamma \vdash M' \rhd^+ M_0 : \Pi x^{A'}.B' \\
\Gamma \vdash N \rhd^+ N_0 : A & \Gamma \vdash N' \rhd^+ N_0 : A' \\
\Gamma \vdash M_{(x)B} N \rhd ? : B[x/N] & \Gamma \vdash M'_{(x)B'} N' \rhd ? : B'[x/N']
\end{array}$$

Shape of Terms

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 $\Gamma \vdash A \equiv \Pi x_1^{U_1} ... x_n^{U_n}.s$ and $\Gamma \vdash B \equiv \Pi x_1^{U_1} ... x_n^{U_n}.t$

What does it means to be typed by a telescope ?

Shape of Terms

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 $\Gamma \vdash A \equiv \Pi x_1^{U_1} ... x_n^{U_n}.s$ and $\Gamma \vdash B \equiv \Pi x_1^{U_1} ... x_n^{U_n}.t$

What does it means to be typed by a telescope ?

(Very Simplified) Shape of Terms in TPOSR in Ts

If $\Gamma \vdash M \rhd ? : \Pi x_1^{U_1} ... x_n^{U_n}.s$ then $\Gamma \vdash M \rhd^+ \lambda x_1^{U_1} ... x_n^{U_n}.P : \Pi x_1^{U_1} ... x_n^{U_n}.s$ and $\Gamma, x_1 : U_1, ..., x_n : U_n \vdash P \rhd P : s$

Shape of Terms

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 $\Gamma \vdash A \equiv \Pi x_1^{U_1} ... x_n^{U_n}.s$ and $\Gamma \vdash B \equiv \Pi x_1^{U_1} ... x_n^{U_n}.t$

What does it means to be typed by a telescope ?

(Very Simplified) Shape of Terms in TPOSR in Ts

If $\Gamma \vdash M \rhd ? : \Pi x_1^{U_1} ... x_n^{U_n}.s$ then $\Gamma \vdash M \rhd^+ \lambda x_1^{U_1} ... x_n^{U_n}.P : \Pi x_1^{U_1} ... x_n^{U_n}.s$ and $\Gamma, x_1 : U_1, ..., x_n : U_n \vdash P \rhd P : s$

By combining Shape of Types and Terms, we can prove that $n > 1$ in our problematic case, thus we can erase the troublesome annotation by performing a β-reduction step first.

Solution to the pitfall

$$
\begin{array}{llll}
\Gamma \vdash M_{(x)B} N & \rhd^+ & M_0\ {}_{(x)B} N & : B[x/N] \\
 & \rhd^+ & (\lambda x^C \lambda\Delta.K)_{(x)B} N & : B[x/N] \\
 & \rhd^+ & \lambda\Delta[x/N].K[x/N] & : B[x/N] \\
 & \rhd^+ & \lambda\Delta[x/N_0].K[x/N_0] & : B[x/N] \\
\Gamma \vdash M'_{(x)B'} N' & \rhd^+ & M_0\ {}_{(x)B'} N' & : B'[x/N'] \\
 & \rhd^+ & (\lambda x^{C'} \lambda\Delta.K)_{(x)B'} N' & : B'[x/N'] \\
 & \rhd^+ & \lambda\Delta[x/N'].K[x/N'] & : B'[x/N'] \\
 & \rhd^+ & \lambda\Delta[x/N_0].K[x/N_0] & : B'[x/N']
\end{array}
$$

Solution to the pitfall

$$
\begin{array}{llll}
\Gamma \vdash M_{(x)B} N & \rhd^+ & M_0\ {}_{(x)B} N & : B[x/N] \\
& \rhd^+ & (\lambda x^C \lambda\Delta.K)_{(x)B} N & : B[x/N] \\
& \rhd^+ & \lambda\Delta[x/N].K[x/N] & : B[x/N] \\
& \rhd^+ & \lambda\Delta[x/N_0].K[x/N_0] & : B[x/N] \\
\Gamma \vdash M'_{(x)B'} N' & \rhd^+ & M_0\ {}_{(x)B'} N' & : B'[x/N'] \\
& \rhd^+ & (\lambda x^{C'} \lambda\Delta.K)_{(x)B'} N' & : B'[x/N'] \\
& \rhd^+ & \lambda\Delta[x/N'].K[x/N'] & : B'[x/N'] \\
& \rhd^+ & \lambda\Delta[x/N_0].K[x/N_0] & : B'[x/N']
\end{array}
$$

With *Subject Reduction* and *Validity of Annotations*, we are know able to prove that PTS $\Rightarrow$ TPOSR, and so:

$$\text{PTSe} \Rightarrow \text{PTS} \Rightarrow \text{TPOSR} \Rightarrow \text{PTSe}$$

Possible Extensions of the Proof

They are several ways to enhance the system:

- Change the conversion rule (with η for example).
- Extend the conversion rule with cumulativity : the road to subtyping.

Possible Extensions of the Proof

They are several ways to enhance the system:

- Change the conversion rule (with η for example).
- Extend the conversion rule with cumulativity : the road to subtyping.

Adding η to the conversion is as hard as always : *Strenghthening* and *Subject Reduction* (even untyped) still depend on one another, *Confluence* is only true on well-typed terms…

Possible Extensions of the Proof

They are several ways to enhance the system:

- Change the conversion rule (with η for example).
- Extend the conversion rule with cumulativity : the road to subtyping.

Adding η to the conversion is as hard as always : *Strenghthening* and *Subject Reduction* (even untyped) still depend on one another, *Confluence* is only true on well-typed terms...

Possible solutions: adding *Strenghthening* as a primitive rule as in *ICC*, restrict to normalizing systems, only add η-expansion...

Subtyping

Adding cumulativity for Π-types and sorts requires an odd lemma before being able to prove the *Shape of Types* property (so even far before Π-*injectivity*) which has resisted all attempts until now:

Subtyping

Adding cumulativity for Π-types and sorts requires an odd lemma before being able to prove the *Shape of Types* property (so even far before Π-*injectivity*) which has resisted all attempts until now:

If $\Gamma \vdash \Pi x_1^{U_1} \ldots x_n^{U_n}.s \equiv \Pi x_1^{V_1} \ldots x_n^{V_n}.s$ then for all t,
$\Gamma \vdash \Pi x_1^{U_1} \ldots x_n^{U_n}.t \equiv \Pi x_1^{V_1} \ldots x_n^{V_n}.t$.

Subtyping

Adding cumulativity for Π-types and sorts requires an odd lemma before being able to prove the *Shape of Types* property (so even far before Π-*injectivity*) which has resisted all attempts until now:

If $\Gamma \vdash \Pi x_1^{U_1} ... x_n^{U_n}.s \equiv \Pi x_1^{V_1} ... x_n^{V_n}.s$ then for all t,
$\Gamma \vdash \Pi x_1^{U_1} ... x_n^{U_n}.t \equiv \Pi x_1^{V_1} ... x_n^{V_n}.t$.

However, even if we manage to prove this, the approach used to proof *Validity of Annotations* do not scale to subtyping, so a new way to prove it still needs to be found.

Conclusion: Where are we ?

What do we have so far:

+ A more precise proof of *Church-Rosser* for TPOSR which works for all useful PTS.

Conclusion: Where are we ?

What do we have so far:

+ A more precise proof of *Church-Rosser* for TPOSR which works for all useful PTS.
+ A new proof of *Validity of Annotations* which settles the equivalence between PTS and PTSe for all useful PTS.

Conclusion: Where are we ?

What do we have so far:

+ A more precise proof of *Church-Rosser* for TPOSR which works for all useful PTS.
+ A new proof of *Validity of Annotations* which settles the equivalence between PTS and PTSe for all useful PTS.
+ At last a base system to start proving the equivalence between *Coq*'s implementation and some axioms-based models that requires a typed equality.

Conclusion: Where are we ?

What do we have so far:

+ A more precise proof of *Church-Rosser* for TPOSR which works for all useful PTS.
+ A new proof of *Validity of Annotations* which settles the equivalence between PTS and PTSe for all useful PTS.
+ At last a base system to start proving the equivalence between *Coq*'s implementation and some axioms-based models that requires a typed equality.

- Dealing with η-conversion is still the same nightmare

Conclusion: Where are we ?

What do we have so far:

+ A more precise proof of *Church-Rosser* for TPOSR which works for all useful PTS.
+ A new proof of *Validity of Annotations* which settles the equivalence between PTS and PTSe for all useful PTS.
+ At last a base system to start proving the equivalence between *Coq*'s implementation and some axioms-based models that requires a typed equality.

- Dealing with η-conversion is still the same nightmare
- Subtyping forces us to throw away the *Shape of Types* approach to *Validity of Annotations* and redo it from scratch.

Conclusion: Next step ?

What are the leads ?

- The issues to prove Church-Rosser arise because we want a single term to have multiple types, maybe we should use *Intersection Types* ?

Conclusion: Next step ?

What are the leads ?

- The issues to prove Church-Rosser arise because we want a single term to have multiple types, maybe we should use *Intersection Types* ?
- PTS are computation-friendly when PTSe are model-friendly, maybe something is missing and they are not the right way to think about syntax ?

Conclusion: Next step ?

What are the leads ?

- The issues to prove Church-Rosser arise because we want a single term to have multiple types, maybe we should use *Intersection Types* ?
- PTS are computation-friendly when PTSe are model-friendly, maybe something is missing and they are not the right way to think about syntax ?

Thank you for your time. Any questions ?