Equality is typable in Semi-Full Pure Type Systems

Vincent Siles - Hugo Herbelin

INRIA - PPS - Ecole Polytechnique

LICS 2010

Road Map

1. A brief history of PTS
2. Equivalence between all presentations
3. Partial Solution with Adams' TPOSR

First steps

- Pure Type Systems were introduced by Berardi and Terlouw, inspired by Barendregt's λ-cube (1992).

First steps

- Pure Type Systems were introduced by Berardi and Terlouw, inspired by Barendregt's λ-cube (1992).
- It is a general framework to have results over a large family of type systems (e.g. STλC, System-$\mathbf{F}/\mathbf{F}_\omega$, CoC, System-$\mathbf{U}$, LF, $\mathit{Type} : \mathit{Type}$, $\ldots$).

First steps

- Pure Type Systems were introduced by Berardi and Terlouw, inspired by Barendregt's λ-cube (1992).
- It is a general framework to have results over a large family of type systems (e.g. STλC, System-$\mathbf{F}/\mathbf{F}_\omega$, CoC, System-$\mathbf{U}$, LF, $Type : Type$, $\ldots$).
- Terms and Contexts:

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

First steps

- Pure Type Systems were introduced by Berardi and Terlouw, inspired by Barendregt's λ-cube (1992).
- It is a general framework to have results over a large family of type systems (e.g. STλC, System-$\mathbf{F}/\mathbf{F}_\omega$, CoC, System-$\mathbf{U}$, LF, $\mathit{Type} : \mathit{Type}$, ...).
- Terms and Contexts:

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

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

First steps

- Pure Type Systems were introduced by Berardi and Terlouw, inspired by Barendregt's λ-cube (1992).
- It is a general framework to have results over a large family of type systems (e.g. STλC, System-$\mathbf{F}/\mathbf{F}_\omega$, CoC, System-$\mathbf{U}$, LF, $\mathit{Type} : \mathit{Type}$, ...).
- Terms and Contexts:

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

- 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

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

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

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

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

Facts about PTS

Generation:

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 $\Gamma, x : A \vdash M : B$.

Facts about PTS

Generation:

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 $\Gamma, x : A \vdash M : B$.

Type Correctness

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

Facts about PTS

Generation:

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 $\Gamma, x : A \vdash M : B$.

Type Correctness

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

Subject Reduction

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

Needs injectivity of Π-types: If $\Pi x^A.B \overset{\beta}{\equiv} \Pi x^C.D$ then $A \overset{\beta}{\equiv} C$ and $B \overset{\beta}{\equiv} D$.
(Easy by confluence of β-reduction).

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 $u = u'$.

Uniqueness of Types

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 $u = u'$.

Uniqueness of Types

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 $u = u'$.

Uniqueness of Types

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" product 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 $u = u'$.

Uniqueness of Types

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" product 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 $u = u'$.

Uniqueness of Types

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" product is typable.
- Semi-full PTS: If $(s,t,u) \in \mathcal{R}el$ then for all t', there is u' such that $(s,t',u') \in \mathcal{R}el$.

Functionality of products

If $\Gamma \vdash \Pi x^A.B : u$ and $\Gamma(x : A) \vdash B' : t'$, there is u' such that $\Gamma \vdash \Pi x^A.B' : u'$.

Shape of types in PTS

In 1993, Jutting studied the types of terms in PTS:
Terms are classified in two families Tv and Ts:

Shape of types in PTS

In 1993, Jutting studied the types of terms in PTS:
Terms are classified in two families Tv and Ts:

$$\begin{cases} v \in V & \longrightarrow & v \in Tv \\ M \in Tv & \longrightarrow & MN, \lambda x^A.M \in Tv \end{cases}$$

Shape of types in PTS

In 1993, Jutting studied the types of terms in PTS:
Terms are classified in two families Tv and Ts:

$$\left\{ \begin{array}{ccc} v \in V & \longrightarrow & v \in Tv \\ M \in Tv & \longrightarrow & MN, \lambda x^A.M \in Tv \end{array} \right.$$

$$\left\{ \begin{array}{ccc} s \in S & \longrightarrow & s \in Ts \\ & \longrightarrow & \Pi x^A.B \in Ts \\ M \in Ts & \longrightarrow & MN, \lambda x^A.M \in Ts \end{array} \right.$$

Shape of types in PTS

In 1993, Jutting studied the types of terms in PTS:
Terms are classified in two families Tv and Ts:

$$\begin{cases} v \in V & \longrightarrow & v \in Tv \\ M \in Tv & \longrightarrow & MN, \lambda x^A.M \in Tv \end{cases}$$

$$\begin{cases} s \in S & \longrightarrow & s \in Ts \\ & \longrightarrow & \Pi x^A.B \in Ts \\ M \in Ts & \longrightarrow & MN, \lambda x^A.M \in Ts \end{cases}$$

- 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 $A \overset{\beta}{\twoheadrightarrow} \Pi x_1^{U_1} ... x_n^{U_n}.s$ and $B \overset{\beta}{\twoheadrightarrow} \Pi x_1^{U_1} ... x_n^{U_n}.t$.

Could we be able to type the equality ?

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

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

Could we be able to type the 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.

Could we be able to type the equality ?

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

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

- β-equality is all about *program computation*, where types are useless.
- However, set-theoretical models need a typed equality.

Could we be able to type the 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.
- However, set-theoretical models need a typed equality.
- Other kind of equalities may depend on types (η-expansion, external axioms).

Could we be able to type the 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.
- However, set-theoretical models need a typed equality.
- Other kind of equalities may depend on types (η-expansion, external axioms).
- So, what if we could ensure that each conversion step is intrisincally well-typed ?

Could we be able to type the 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.
- However, set-theoretical models need a typed equality.
- Other kind of equalities may depend on types (η-expansion, external axioms).
- So, what if we could ensure that each conversion step is intrinsincally well-typed ?

$\hookrightarrow$ all this lead to the definition of PTS *with Judgmental Equality* (aka a semantical version of PTS, mostly inspired by [Martin-Löf 84]).

PTSe typing rules (1)

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

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

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

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

PTSe typing rules (1)

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

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

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

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

PTSe typing rules (2)

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

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

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

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

PTSe typing rules (3)

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

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

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

Are both systems the same ?

Proof of the equivalence

We prove by mutual induction that

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

Proof of the equivalence

We prove by mutual induction that

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

$\Rightarrow$ trivial, we just "lose" some information.

Proof of the equivalence

We prove by mutual induction that

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

$\Rightarrow$ trivial, we just "lose" some information.
$\Leftarrow$ we need to find a way to type all the intermediate steps.

Proof of the equivalence

We prove by mutual induction that

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

$\Rightarrow$ trivial, we just “lose” some information.
$\Leftarrow$ 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 ?

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

$M \twoheadrightarrow P \twoheadleftarrow N$

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

How do we do this ?

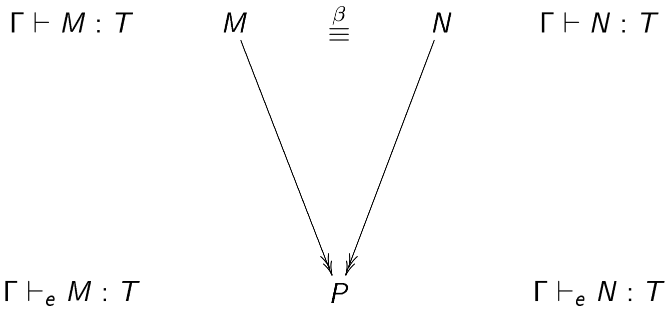
$$\Gamma \vdash M : T \qquad M \overset{\beta}{\equiv} N \qquad \Gamma \vdash N : T$$

$$M \twoheadrightarrow P \twoheadleftarrow N$$

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

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

How do we do this ?



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

How do we do this ?

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

$M \twoheadrightarrow P \twoheadleftarrow N$

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

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

The need of Subject Reduction

As pointed out in [Geuvers-Werner 94], 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

As pointed out in [Geuvers-Werner 94], 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 for typed equality judgments, which is a really difficult question for PTSe since it relies on (typed) property of *Confluence*,

The need of Subject Reduction

As pointed out in [Geuvers-Werner 94], 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 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*,

The need of Subject Reduction

As pointed out in [Geuvers-Werner 94], 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 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,

The need of Subject Reduction

As pointed out in [Geuvers-Werner 94], 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 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 ...

Current status of the equivalence

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

Current status of the equivalence

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

Current status of the equivalence

- for functional PTS : [Adams 06] “Pure Type Systems with Judgmental Equality”.
- for semi-full and full PTS : [Herbelin-Siles 10] “Equality is typable in Semi-Full Pure Type Systems”.
- But the question is ~~still open~~ finally solved for any kind of PTS! (Herbelin-Siles, submitted at JFP).

Adams' approach

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

Adams' approach

- In order to break the loop, Adams only considered the *functional* PTS and 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' approach

- In order to break the loop, Adams only considered the *functional* PTS and 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' approach

- In order to break the loop, Adams only considered the *functional* PTS and 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' approach

- In order to break the loop, Adams only considered the *functional* PTS and 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 *injectivity of* Π*-types*.

Adams' approach

- In order to break the loop, Adams only considered the *functional* PTS and 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 *injectivity of* Π*-types*.
  - Prove that TPOSR has *Subject-Reduction*.

Adams' approach

- In order to break the loop, Adams only considered the *functional* PTS and 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 *injectivity of* Π*-types*.
  - Prove that TPOSR has *Subject-Reduction*.
  - Prove that TPOSR is equivalent to PTS and PTSe.

TPOSR typing rules (1)

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

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

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

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

TPOSR typing rules (1)

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

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

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

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

TPOSR typing rules (2)

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

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

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

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

TPOSR typing rules (2)

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

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

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

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

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

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

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 beta and app rules:

when applying an induction hypothesis, both contexts need to be syntactically the same.

Where is the trap ?

- Input:
  $\Gamma \vdash M_{(x)\ B} N \triangleright M'_{(x)\ B'} N' : B[N/x]$
  $\Gamma \vdash M_{(x)\ B} N \triangleright M''_{(x)\ B''} N'' : B[N/x]$

Where is the trap ?

- Input:
  $\Gamma \vdash M_{(x)\ B} N \rhd M'_{(x)\ B'} N' : B[N/x]$
  $\Gamma \vdash M_{(x)\ B} N \rhd M''_{(x)\ B''} N'' : B[N/x]$
- Induction Hypothesis over B: If $\Gamma(x : A) \vdash B \rhd B' : T$ and $\Gamma(x : A) \vdash B \rhd B'' : T'$ then there is ...

Where is the trap ?

- Input:
  $\Gamma \vdash M_{(x)\ B} N \rhd M'_{(x)\ B'} N' : B[N/x]$
  $\Gamma \vdash M_{(x)\ B} N \rhd M''_{(x)\ B''} N'' : B[N/x]$
- Induction Hypothesis over B: If $\Gamma(x : A) \vdash B \rhd B' : T$ and $\Gamma(x : A) \vdash B \rhd B'' : T'$ then there is ...
- Information about B: $\Gamma(x : A) \vdash B \rhd B' : t$ and $\Gamma(x : C) \vdash B \rhd B'' : t'$

Where is the trap ?

- Input:
  $\Gamma \vdash M_{(x)\ B} N \rhd M'_{(x)\ B'} N' : B[N/x]$
  $\Gamma \vdash M_{(x)\ B} N \rhd M''_{(x)\ B''} N'' : B[N/x]$
- Induction Hypothesis over B: If $\Gamma(x : A) \vdash B \rhd B' : T$ and $\Gamma(x : A) \vdash B \rhd B'' : T'$ then there is ...
- Information about B: $\Gamma(x : A) \vdash B \rhd B' : t$ and $\Gamma(x : C) \vdash B \rhd B'' : t'$
- Information about A and C (by induction): there is N_0 such that $\Gamma \vdash N' \rhd N_0 : A$ and $\Gamma \vdash N'' \rhd N_0 : C$ (resp M_0).

Where is the trap ?

- Input:
  $\Gamma \vdash M_{(x)\ B} N \rhd M'_{(x)\ B'} N' : B[N/x]$
  $\Gamma \vdash M_{(x)\ B} N \rhd M''_{(x)\ B''} N'' : B[N/x]$
- Induction Hypothesis over B: If $\Gamma(x : A) \vdash B \rhd B' : T$ and $\Gamma(x : A) \vdash B \rhd B'' : T'$ then there is ...
- Information about B: $\Gamma(x : A) \vdash B \rhd B' : t$ and $\Gamma(x : C) \vdash B \rhd B'' : t'$
- Information about A and C (by induction): there is N_0 such that $\Gamma \vdash N' \rhd N_0 : A$ and $\Gamma \vdash N'' \rhd N_0 : C$ (resp M_0).

$\hookrightarrow$ 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 $\Gamma \vdash A \equiv C$ quite easily.

Functional vs Semi-Full

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

Shape of Types in semi-full 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 $\Gamma \vdash B \equiv \Pi x_1^{U_1} ... x_n^{U_n}.t$

Functional vs Semi-Full

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

Shape of Types in semi-full 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 $\Gamma \vdash B \equiv \Pi x_1^{U_1} ... x_n^{U_n}.t$

By applying it to M_0 and N_0, we can show that $s = t$ by removing the annotations and using untyped *Confluence* of usual β-reduction.

Functional vs Semi-Full

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

Shape of Types in semi-full TPOSR

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

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

By applying it to M_0 and N_0, we can show that $s = t$ by removing the annotations and using untyped *Confluence* of usual β-reduction.

We can now finish to prove *Church-Rosser*, *injectivity of* Πs and *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 there are Γ^*, M^* and T^* such that $\Gamma^* \vdash M^* \rhd M^* : T^*$, $|\Gamma^*| = \Gamma$, $|M^*| = M$ 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 there are Γ^*, M^* and T^* such that $\Gamma^* \vdash M^* \rhd M^* : T^*$, $|\Gamma^*| = \Gamma$, $|M^*| = M$ and $|T^*| = T$.

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

Erased Conversion

- If $\Gamma \vdash A \rhd ? : s$, $\Gamma \vdash B \rhd ? : t$ and $|A| = |B|$, then $\Gamma \vdash A \equiv B$.
- 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

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

Shape of Terms

Shape of Types in TPOSR

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

What does it mean to be typed by a telescope ?

Shape of Terms

Shape of Types in TPOSR

$\ldots$ 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 mean to be typed by a telescope ?

(Simple) Shape of Terms in TPOSR

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

- either $\Gamma \vdash A \equiv B$
- or $\Gamma \vdash M \rhd K : A$ and $\Gamma \vdash M \rhd K : B$ where K is a sort, a product $\Pi x^U.V$ or an abstraction $\lambda x^U.V$.

Shape of Terms

Shape of Types in TPOSR

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

What does it mean to be typed by a telescope ?

(Simple) Shape of Terms in TPOSR

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

- either $\Gamma \vdash A \equiv B$
- or $\Gamma \vdash M \rhd K : A$ and $\Gamma \vdash M \rhd K : B$ where K is a sort, a product $\Pi x^U.V$ or an abstraction $\lambda x^U.V$.

In the previous problematic case, M_0 is typed by a Π-type, so K can not be a sort, nor a Π-types, so we just created a β-redex whose reduction will erase the annotation.

Consequences of the equivalence

Equivalence PTS / PTSe

- $\Gamma \vdash M : T$ iff $\Gamma \vdash_e M : T$.
- $\Gamma \vdash M : T$ $\Gamma \vdash N : T$ and $M \stackrel{\beta}{\equiv} N$ iff $\Gamma \vdash_e M = N : T$.

Consequences of the equivalence

Equivalence PTS / PTSe

- $\Gamma \vdash M : T$ iff $\Gamma \vdash_e M : T$.
- $\Gamma \vdash M : T$ $\Gamma \vdash N : T$ and $M \overset{\beta}{\equiv} N$ iff $\Gamma \vdash_e M = N : T$.

What about *Subject Reduction* for PTSe?

Consequences of the equivalence

Equivalence PTS / PTSe

- $\Gamma \vdash M : T$ iff $\Gamma \vdash_e M : T$.
- $\Gamma \vdash M : T$ $\Gamma \vdash N : T$ and $M \overset{\beta}{\equiv} N$ iff $\Gamma \vdash_e M = N : T$.

What about *Subject Reduction* for PTSe?

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

- By equivalence, $\Gamma \vdash M : T$.
- By *Subject Reduction* for PTS, $\Gamma \vdash N : T$.
- So by equivalence, $\Gamma \vdash_e M = N : T$.

Possible Extensions of the proof (1)

The system can be enhanced by changing the conversion rule, with η for example.

Possible Extensions of the proof (1)

The system can be enhanced by changing the conversion rule, with η for example.

- Adding η to the conversion is as hard as always : *Strengthening* and *Subject Reduction* (even untyped) still depend on one another, *Confluence* is only true on well-typed terms...
- Possible solutions: adding *Strengthening* as a primitive rule, restrict to normalizing systems, using a weaker form of *Confluence*...

Possible Extensions of the proof (2)

We can also consider adding *subtyping*:

Possible Extensions of the proof (2)

We can also consider adding *subtyping*:

- Using this approach, we are unable to prove the *Shape of Types* property for the *Extended Calculus of Constructions* (ECC).
- But with our approach to prove the general case of any PTS, we were able to prove that "TPOSR_{ECC}" enjoys Π-injectivity and *Subject Reduction*.
- However, even with the general framework, the *Validity of Annotations* do not scale to subtyping (*Erased Conversion* is wrong).

Conclusion: Where are we ?

What do we have so far:

+ The whole proof is formalized in *Coq*.

Conclusion: Where are we ?

What do we have so far:

+ The whole proof is formalized in *Coq*.
+ A new proof of *Church-Rosser* and *Validity of Annotations* for all semi-full and full TPOSR.

Conclusion: Where are we ?

What do we have so far:

+ The whole proof is formalized in *Coq*.
+ A new proof of *Church-Rosser* and *Validity of Annotations* for all semi-full and full TPOSR.
+ This proof can be extended to prove this equivalence for all PTS.

Conclusion: Where are we ?

What do we have so far:

+ The whole proof is formalized in *Coq*.
+ A new proof of *Church-Rosser* and *Validity of Annotations* for all semi-full and full TPOSR.
+ This proof can be extended to prove this equivalence for all PTS.
+ implementation *a la* PTS and model construction *a la* PTSe are now linked: an extension with subtyping would be useful for meta-theory of proof assistant.

Conclusion: Where are we ?

What do we have so far:

+ The whole proof is formalized in *Coq*.
+ A new proof of *Church-Rosser* and *Validity of Annotations* for all semi-full and full TPOSR.
+ This proof can be extended to prove this equivalence for all PTS.
+ implementation *a la* PTS and model construction *a la* PTSe are now linked: an extension with subtyping would be useful for meta-theory of proof assistant.

- Dealing with η-conversion is still complicated

Conclusion: Where are we ?

What do we have so far:

+ The whole proof is formalized in *Coq*.
+ A new proof of *Church-Rosser* and *Validity of Annotations* for all semi-full and full TPOSR.
+ This proof can be extended to prove this equivalence for all PTS.
+ implementation *a la* PTS and model construction *a la* PTSe are now linked: an extension with subtyping would be useful for meta-theory of proof assistant.

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

Thank you for your attention

Any questions ?

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