Principal Typings for Explicit Substitution*

Daniel Lima Ventura, Mauricio Ayala-Rincón
Departamento de Matemática, Universidade de Brasília
Brasília D.F., Brazil
Fairouz Kamareddine
ULTRA Group, Heriot-Watt University
Edinburgh, Scotland

Computability in Europe - CiE 2008
June 18, 2008, Athens

* Supported by the Brazilian Research Foundation (CNPq) and Coordination of Higher Education (CAPES)

Universidade de Brasília

Outline

Principal Typing

Let $A \vdash M : \tau$ be a type judgement in some type system $\mathcal{S}$

- $\Theta = \langle A, \tau \rangle$ is a typing of M in $\mathcal{S}$ ($\mathcal{S} \Vdash M : \Theta$).

Principal Typing

Let $A \vdash M : \tau$ be a type judgement in some type system $\mathcal{S}$

- $\Theta = \langle A, \tau \rangle$ is a typing of M in $\mathcal{S}$ ($\mathcal{S} \Vdash M : \Theta$).
- Θ is a **principal typing** (PT) of M if $\mathcal{S} \Vdash M : \Theta$ and Θ "represents" any other possible typing of M.

Principal Typing

Let $A \vdash M : \tau$ be a type judgement in some type system $\mathcal{S}$

- $\Theta = \langle A, \tau \rangle$ is a typing of M in $\mathcal{S}$ ($\mathcal{S} \Vdash M : \Theta$).
- Θ is a **principal typing** (PT) of M if $\mathcal{S} \Vdash M : \Theta$ and Θ "represents" any other possible typing of M.
- PT property allows *compositional* type inference

Principal Typing versus Principal Type [Jim96]

Given term M and context A, τ is a **principal type** of M if it represents any other possible type of M in A.

Principal Typing versus Principal Type [Jim96]

Given term M and context A, τ is a **principal type** of M if it represents any other possible type of M in A.

Principal Type: $A \vdash M :?$

Principal Typing versus Principal Type [Jim96]

Given term M and context A, τ is a **principal type** of M if it represents any other possible type of M in A.

Principal Type: $A \vdash M :?$

Example

$y{:}\alpha{\to}\alpha \vdash \lambda_x.(y\ x) :?$

Principal Typing versus Principal Type [Jim96]

Given term M and context A, τ is a **principal type** of M if it represents any other possible type of M in A.

Principal Type: $A \vdash M :?$

Example

$y{:}\alpha{\rightarrow}\alpha \vdash \lambda_x.(y\ x) :?$
Answer: $\alpha{\rightarrow}\alpha$

Principal Typing versus Principal Type [Jim96]

Given term M and context A, τ is a **principal type** of M if it represents any other possible type of M in A.

Principal Type: $A \vdash M :?$

Example

$y{:}\alpha{\rightarrow}\alpha \vdash \lambda_x.(y\ x) :?$
Answer: $\alpha{\rightarrow}\alpha$

Example

$y{:}\alpha{\rightarrow}\beta \vdash \lambda_x.(y\ x) :?$

Principal Typing versus Principal Type [Jim96]

Given term M and context A, τ is a **principal type** of M if it represents any other possible type of M in A.

Principal Type: $A \vdash M :?$

Example

$y{:}\alpha{\rightarrow}\alpha \vdash \lambda_x.(y\ x) :?$
Answer: $\alpha{\rightarrow}\alpha$

Example

$y{:}\alpha{\rightarrow}\beta \vdash \lambda_x.(y\ x) :?$
Answer: $\alpha{\rightarrow}\beta$

Principal Typing versus Principal Type

Principal Typing: $? \vdash M :?$
Are there typings for M? If so which one is principal?

Principal Typing versus Principal Type

Principal Typing: $? \vdash M :?$
Are there typings for M? If so which one is principal?

Example

$? \vdash \lambda_x.(y\ x) :?$

Principal Typing versus Principal Type

Principal Typing: $? \vdash M :?$
Are there typings for M? If so which one is principal?

Example

$? \vdash \lambda_x.(y\ x) :?$
Possible typings: $\langle y{:}\alpha{\to}\beta, \alpha{\to}\beta\rangle$;
$\langle y{:}\alpha{\to}\alpha, \alpha{\to}\alpha\rangle$; and many more

Principal Typing versus Principal Type

Principal Typing: $? \vdash M :?$
Are there typings for M? If so which one is principal?

Example

$? \vdash \lambda_x.(y\ x) :?$
Possible typings: $\langle y{:}\alpha{\rightarrow}\beta, \alpha{\rightarrow}\beta\rangle$;
$\langle y{:}\alpha{\rightarrow}\alpha, \alpha{\rightarrow}\alpha\rangle$; and many more
Principal typing: $\langle y{:}\alpha{\rightarrow}\beta, \alpha{\rightarrow}\beta\rangle$

Principal Typing versus Principal Type

Principal Typing: $? \vdash M :?$
Are there typings for M? If so which one is principal?

Example

$? \vdash \lambda_x.(y\ x) :?$
Possible typings: $\langle y{:}\alpha{\rightarrow}\beta, \alpha{\rightarrow}\beta\rangle$;
$\langle y{:}\alpha{\rightarrow}\alpha, \alpha{\rightarrow}\alpha\rangle$; and many more
Principal typing: $\langle y{:}\alpha{\rightarrow}\beta, \alpha{\rightarrow}\beta\rangle$

Example

$? \vdash \lambda_x.(y\ (y\ x)) :?$

Principal Typing versus Principal Type

Principal Typing: $? \vdash M :?$
Are there typings for M? If so which one is principal?

Example

$? \vdash \lambda_x.(y\ x) :?$
Possible typings: $\langle y{:}\alpha{\rightarrow}\beta, \alpha{\rightarrow}\beta\rangle$;
$\langle y{:}\alpha{\rightarrow}\alpha, \alpha{\rightarrow}\alpha\rangle$; and many more
Principal typing: $\langle y{:}\alpha{\rightarrow}\beta, \alpha{\rightarrow}\beta\rangle$

Example

$? \vdash \lambda_x.(y\ (y\ x)) :?$
Possible typings: $\langle y{:}\alpha{\rightarrow}\alpha, \alpha{\rightarrow}\alpha\rangle$;
$\langle y{:}(\alpha{\rightarrow}\beta){\rightarrow}\alpha{\rightarrow}\beta, (\alpha{\rightarrow}\beta){\rightarrow}\alpha{\rightarrow}\beta\rangle$; and many more

Principal Typing versus Principal Type

Principal Typing: $? \vdash M :?$
Are there typings for M? If so which one is principal?

Example

$? \vdash \lambda_x.(y\ x) :?$
Possible typings: $\langle y{:}\alpha\to\beta, \alpha\to\beta\rangle$;
$\langle y{:}\alpha\to\alpha, \alpha\to\alpha\rangle$; and many more
Principal typing: $\langle y{:}\alpha\to\beta, \alpha\to\beta\rangle$

Example

$? \vdash \lambda_x.(y\ (y\ x)) :?$
Possible typings: $\langle y{:}\alpha\to\alpha, \alpha\to\alpha\rangle$;
$\langle y{:}(\alpha\to\beta)\to\alpha\to\beta, (\alpha\to\beta)\to\alpha\to\beta\rangle$; and many more
Principal typing: $\langle y{:}\alpha\to\alpha, \alpha\to\alpha\rangle$

Principal Typing versus Principal Type

	Principal Type	Principal Typing
STLC	✓ [Hi97]	✓ [Wells02]
Hindley/Milner	✓ [DM82]	X [Wells02]
System F	?	X [Wells02]
System $\mathbb{I}$	✓ [KW04]	✓ [KW04]

Principal Typing

Definition

For some typing Θ in $\mathcal{S}$ let $\mathit{Terms}_{\mathcal{S}}(\Theta)=\{M \,|\, \mathcal{S} \Vdash M{:}\Theta\}$.

Principal Typing

Definition

For some typing Θ in $\mathcal{S}$ let $\mathit{Terms}_{\mathcal{S}}(\Theta)=\{M \,|\, \mathcal{S} \Vdash M{:}\Theta\}$.

Definition (Typing's Partial Order)

Let $\Theta \leq_{\mathcal{S}} \Theta'$ iff $\mathit{Terms}_{\mathcal{S}}(\Theta) \subseteq \mathit{Terms}_{\mathcal{S}}(\Theta')$

Principal Typing

Definition

For some typing Θ in $\mathcal{S}$ let $\mathit{Terms}_{\mathcal{S}}(\Theta)=\{M \mid \mathcal{S} \Vdash M{:}\Theta\}$.

Definition (Typing's Partial Order)

Let $\Theta \leq_{\mathcal{S}} \Theta'$ iff $\mathit{Terms}_{\mathcal{S}}(\Theta) \subseteq \mathit{Terms}_{\mathcal{S}}(\Theta')$

Example

Let $\Theta_1 = \langle y{:}\alpha{\rightarrow}\beta, \alpha{\rightarrow}\beta\rangle$ and $\Theta_2 = \langle y{:}\alpha{\rightarrow}\alpha, \alpha{\rightarrow}\alpha\rangle$.
Moreover, $\lambda_x.(y\ x)$ is in $\mathit{Terms}(\Theta_1) \cap \mathit{Terms}(\Theta_2)$.
And $\lambda_x.x$ is in $\mathit{Terms}(\Theta_2) \setminus \mathit{Terms}(\Theta_1)$.

Principal Typing

Definition

For some typing Θ in $\mathcal{S}$ let $\mathit{Terms}_{\mathcal{S}}(\Theta)=\{M \mid \mathcal{S} \Vdash M{:}\Theta\}$.

Definition (Typing's Partial Order)

Let $\Theta \leq_{\mathcal{S}} \Theta'$ iff $\mathit{Terms}_{\mathcal{S}}(\Theta) \subseteq \mathit{Terms}_{\mathcal{S}}(\Theta')$

Example

Let $\Theta_1 = \langle y{:}\alpha{\rightarrow}\beta, \alpha{\rightarrow}\beta\rangle$ and $\Theta_2 = \langle y{:}\alpha{\rightarrow}\alpha, \alpha{\rightarrow}\alpha\rangle$.
Moreover, $\lambda_x.(y\ x)$ is in $\mathit{Terms}(\Theta_1) \cap \mathit{Terms}(\Theta_2)$.
And $\lambda_x.x$ is in $\mathit{Terms}(\Theta_2) \setminus \mathit{Terms}(\Theta_1)$.
Therefore, $\Theta_1 \leq_{\mathcal{S}} \Theta_2$.

Principal Typing

Definition (General PT [Wells02])

A typing Θ in system S is principal for some term M iff $\mathcal{S} \Vdash M{:}\Theta$ and for all Θ', $\mathcal{S} \Vdash M{:}\Theta'$ implies $\Theta \leq_{\mathcal{S}} \Theta'$.

Simple Type System

Definition (Simple Types and Contexts)

Types $\tau ::= \alpha \,|\, \tau \to \tau$ **Contexts** $A ::= nil \,|\, \tau.A$

Simple Type System

Definition (Simple Types and Contexts)

Types $\tau ::= \alpha \,|\, \tau \to \tau$ **Contexts** $A ::= nil \,|\, \tau.A$

- $|A|$: A's length.
- For $A = \tau_1.\tau_2.\cdots.\tau_m$:
  $A_{\leq n} = \tau_1.\cdots.\tau_n$
  $A_{\geq n} = \tau_n.\cdots.\tau_m.nil$
- $A_{<n}$ and $A_{>n}$ defined similarly

The System $TA_{\lambda dB}$

Syntax

Terms $M ::= \underline{n} \mid (M\ M) \mid \lambda.M$

Typing Rules

$$\tau.A \vdash \underline{1} : \tau \ (Var) \qquad \frac{A \vdash \underline{n} : \tau}{\sigma.A \vdash \underline{n+1} : \tau} \ (Varn)$$

$$\frac{\sigma.A \vdash M : \tau}{A \vdash \lambda.M : \sigma \to \tau} \ (Lambda) \qquad \frac{A \vdash M : \sigma \to \tau \quad A \vdash N : \sigma}{A \vdash (M\ N) : \tau} \ (App)$$

PT in $TA_{\lambda dB}$

Definition (System Dependent PT in $TA_{\lambda dB}$)

In $TA_{\lambda dB}$, $\Theta = \langle A, \tau \rangle$ is *system dependent PT* for M iff $\mathrm{TA}_{\lambda dB} \Vdash M : \Theta$ and for any $\Theta' = \langle A', \tau' \rangle$, if $\mathrm{TA}_{\lambda dB} \Vdash M : \Theta'$, then there exists some type substitution s such that $s(A) = A'_{\leq |A|}.nil$ and $s(\tau) = \tau'$.

PT for $TA_{\lambda dB}$

Theorem (Correspondence for $TA_{\lambda dB}$)

In $TA_{\lambda dB}$, Θ is a system dependent PT for M iff Θ is a general PT for M.

PT for $TA_{\lambda dB}$

Theorem (Correspondence for $TA_{\lambda dB}$)

In $TA_{\lambda dB}$, Θ is a system dependent PT for M iff Θ is a general PT for M.

Theorem (PT for $TA_{\lambda dB}$)

$TA_{\lambda dB}$ satisfies the PT property.

Type Inference for $TA_{\lambda dB}$ [AyMu2000]

1st Given term M, let M' be its annoted version.
Ex: for $M = \lambda.(\underline{2}\ \ \underline{1})$, $M' = (\lambda.(\underline{2}\,{}^{A_1}_{\tau_1}\ \ \underline{1}\,{}^{A_2}_{\tau_2})^{A_3}_{\tau_3})^{A_4}_{\tau_4}$

Type Inference for $TA_{\lambda dB}$ [AyMu2000]

1st Given term M, let M' be its annoted version.
Ex: for $M = \lambda.(\underline{2}\ \ \underline{1})$, $M' = (\lambda.(\underline{2}\,{}^{A_1}_{\tau_1}\ \ \underline{1}\,{}^{A_2}_{\tau_2})^{A_3}_{\tau_3})^{A_4}_{\tau_4}$

2nd Let R_0 be the set of subterms of M'. Start using the type inference algorithm on $\langle\langle R_0, \varnothing \rangle\rangle$

Type Inference for $TA_{\lambda dB}$ [AyMu2000]

1st Given term M, let M' be its annoted version.
Ex: for $M = \lambda.(\underline{2}\ \ \underline{1})$, $M' = (\lambda.(\underline{2}\,^{A_1}_{\tau_1}\ \ \underline{1}\,^{A_2}_{\tau_2})^{A_3}_{\tau_3})^{A_4}_{\tau_4}$

2nd Let R_0 be the set of subterms of M'. Start using the type inference algorithm on $\langle\langle R_0, \varnothing\rangle\rangle$

3rd When $\langle\langle \varnothing, E\rangle\rangle$ is reached, E is the set of equations on type and context variables

Type Inference for $TA_{\lambda dB}$ [AyMu2000]

1st Given term M, let M' be its annoted version.
Ex: for $M = \lambda.(\underline{2}\ \ \underline{1})$, $M' = (\lambda.(\underline{2}\,{}^{A_1}_{\tau_1}\ \ \underline{1}\,{}^{A_2}_{\tau_2})^{A_3}_{\tau_3})^{A_4}_{\tau_4}$

2nd Let R_0 be the set of subterms of M'. Start using the type inference algorithm on $\langle\langle R_0, \varnothing \rangle\rangle$

3rd When $\langle\langle \varnothing, E \rangle\rangle$ is reached, E is the set of equations on type and context variables

4th Use a first order unification algorithm to obtain the m.g.u.

Type Inference for $TA_{\lambda dB}$

(Var)	$\langle\langle R \cup \{\underline{1}^{A}_{\tau}\}, E\rangle\rangle \rightarrow \langle\langle R, E \cup \{A = \tau.A'\}\rangle\rangle$
(Varn)	$\langle\langle R \cup \{\underline{n}^{A}_{\tau}\}, E\rangle\rangle \rightarrow \langle\langle R, E \cup \{A = \tau'_1. \cdots .\tau'_{n-1}.\tau.A'\}\rangle\rangle$
(Lambda)	$\langle\langle R \cup \{(\lambda.M^{A_1}_{\tau_1})^{A_2}_{\tau_2}\}, E\rangle\rangle \rightarrow \langle\langle R, E \cup \{\tau_2 = \tau^* \rightarrow \tau_1, A_1 = \tau^*.A_2\}\rangle\rangle$
(App)	$\langle\langle R \cup \{(M^{A_1}_{\tau_1}\ N^{A_2}_{\tau_2})^{A_3}_{\tau_3}\}, E\rangle\rangle \rightarrow \langle\langle R, E \cup \{A_1 = A_2, A_2 = A_3, \tau_1 = \tau_2 \rightarrow \tau_3\}\rangle\rangle$

Obs: τ', τ^* and A' are fresh variables

The System $TA_{\lambda s_e}$ [KR97]

Syntax

Terms $\quad M ::= \underline{n} \mid (M\ M) \mid \lambda.M \mid M\,\sigma^i M \mid \varphi^j_k\, M$

Typing Rules

$$\frac{A_{\leq k}.A_{\geq k+i} \vdash M : \tau}{A \vdash \varphi^i_k\, M : \tau}\ (Phi) \qquad \frac{A_{\geq i} \vdash N : \sigma \quad A_{<i}.\sigma.A_{\geq i} \vdash M : \tau}{A \vdash M\,\sigma^i N : \tau}\ (Sigma)$$

PT in $TA_{\lambda s_e}$

Definition (System Dependent PT in $TA_{\lambda s_e}$)

In $TA_{\lambda s_e}$, $\Theta = \langle A, \tau \rangle$ is a *system dependent PT* for M iff $\mathrm{TA}_{\lambda s_e} \Vdash M : \Theta$ and for any typing $\Theta' = \langle A', \tau' \rangle$, if $\mathrm{TA}_{\lambda s_e} \Vdash M : \Theta'$, then there exists some type substitution s such that $s(A) = A'_{\leq |A|}.nil$ and $s(\tau) = \tau'$.

PT for $TA_{\lambda s_e}$

Theorem (Correspondence for $TA_{\lambda s_e}$)

In $TA_{\lambda s_e}$, Θ is a system dependent PT for M iff Θ is a general PT for M.

PT for $TA_{\lambda s_e}$

Theorem (Correspondence for $TA_{\lambda s_e}$)

In $TA_{\lambda s_e}$, Θ is a system dependent PT for M iff Θ is a general PT for M.

Theorem (PT for $TA_{\lambda s_e}$)

$TA_{\lambda s_e}$ satisfies the PT property.

PT for $TA_{\lambda s_e}$

Theorem (Correspondence for $TA_{\lambda s_e}$)

In $TA_{\lambda s_e}$, Θ is a system dependent PT for M iff Θ is a general PT for M.

Theorem (PT for $TA_{\lambda s_e}$)

$TA_{\lambda s_e}$ satisfies the PT property.

$$
\begin{array}{ll}
\text{(Sigma)} & \langle\!\langle R \cup \{(M^{A_1}_{\tau_1}\, \sigma^i\, N^{A_2}_{\tau_2})^{A_3}_{\tau_3}\}, E\rangle\!\rangle \to \\
& \qquad \langle\!\langle R, E \cup \{\tau_1{=}\tau_3, A_1{=}\tau'_1.\cdots.\tau'_{i-1}.\tau_2.A_2, A_3{=}\tau'_1.\cdots.\tau'_{i-1}.A_2\}\rangle\!\rangle \\
\text{(Phi)} & \langle\!\langle R \cup \{(\varphi^i_k\, M^{A_1}_{\tau_1})^{A_2}_{\tau_2}\}, E\rangle\!\rangle \to \\
& \qquad \langle\!\langle R, E \cup \{\tau_1 = \tau_2, A_2 = \tau'_1.\cdots.\tau'_{k+i-1}.A', A_1 = \tau'_1.\cdots.\tau'_k.A'\}\rangle\!\rangle
\end{array}
$$

The System $TA_{\lambda\sigma}$[ACCL91]

Syntax

Terms $M ::= \underline{1} \,|\, (M\ M) \,|\, \lambda.M \,|\, M[S]$ Substitution $S ::= id \,|\, \uparrow \,|\, M.S \,|\, S \circ S$

Typing rules

Terms

$$\tau.A \vdash \underline{1} : \tau \ (var) \qquad \frac{\sigma.A \vdash M : \tau}{A \vdash \lambda.M : \sigma \to \tau} \ (lambda)$$

$$\frac{A \vdash M : \sigma \to \tau \quad A \vdash N : \sigma}{A \vdash (M\ N) : \tau} \ (app) \qquad \frac{A \vdash S \rhd A' \quad A' \vdash M : \tau}{A \vdash M[S] : \tau} \ (clos)$$

Substitutions

$$A \vdash id \rhd A \ (id) \qquad \tau.A \vdash \uparrow \rhd A \ (shift)$$

$$\frac{A \vdash M : \tau \quad A \vdash S \rhd A'}{A \vdash M.S \rhd \tau.A'} \ (cons) \qquad \frac{A \vdash S'' \rhd A'' \quad A'' \vdash S' \rhd A'}{A \vdash S' \circ S'' \rhd A'} \ (comp)$$

PT in $TA_{\lambda\sigma}$

Definition (System Dependent PT in $TA_{\lambda\sigma}$)

In $TA_{\lambda\sigma}$, $\Theta = \langle A, \mathbb{T}\rangle$ is a *system dependent PT* for M iff $\mathrm{TA}_{\lambda\sigma} \Vdash M : \Theta$ and for any typing $\Theta' = \langle A', \mathbb{T}'\rangle$, if $\mathrm{TA}_{\lambda\sigma} \Vdash M : \Theta'$, then there exists some type substitution s such that $s(A) = A'_{\leq |A|}.nil$ and: if $\mathbb{T}$ is a type then $s(\mathbb{T}) = \mathbb{T}'$, otherwise $s(\mathbb{T}) = \mathbb{T}'_{\leq |\mathbb{T}|}.nil$.

PT for $TA_{\lambda\sigma}$

Theorem (Correspondence for $TA_{\lambda\sigma}$)

In $TA_{\lambda\sigma}$, Θ is a system dependent PT for M iff Θ is a general PT for M.

PT for $TA_{\lambda\sigma}$

Theorem (Correspondence for $TA_{\lambda\sigma}$)

In $TA_{\lambda\sigma}$, Θ is a system dependent PT for M iff Θ is a general PT for M.

Theorem (PT for $TA_{\lambda\sigma}$)

$TA_{\lambda\sigma}$ satisfies the PT property.

Type Inference for $TA_{\lambda\sigma}$ [Bo95]

(Var)	$\langle\langle R \cup \{\underline{1}^{A}_{\tau}\}, E\rangle\rangle$	$\rightarrow \langle\langle R, E \cup \{A = \tau.A'\}\rangle\rangle$
(Lambda)	$\langle\langle R \cup \{(\lambda.M^{A_1}_{\tau_1})^{A_2}_{\tau_2}\}, E\rangle\rangle$	$\rightarrow \langle\langle R, E \cup \{\tau_2 = \tau^* \rightarrow \tau_1, A_1 = \tau^*.A_2\}\rangle\rangle$
(App)	$\langle\langle R \cup \{(M^{A_1}_{\tau_1}\ N^{A_2}_{\tau_2})^{A_3}_{\tau_3}\}, E\rangle\rangle$	$\rightarrow \langle\langle R, E \cup \{A_1 = A_2, A_2 = A_3, \tau_1 = \tau_2 \rightarrow \tau_3\}\rangle\rangle$
(Clos)	$\langle\langle R \cup \{(M^{A_1}_{\tau_1}[S^{A_2}_{A_3}])^{A_4}_{\tau_2}\}, E\rangle\rangle$	$\rightarrow \langle\langle R, E \cup \{A_1 = A_3, A_2 = A_4, \tau_1 = \tau_2\}\rangle\rangle$
(Id)	$\langle\langle R \cup \{id^{A_1}_{A_2}\}, E\rangle\rangle$	$\rightarrow \langle\langle R, E \cup \{A_1 = A_2\}\rangle\rangle$
(Shift)	$\langle\langle R \cup \{\uparrow^{A_1}_{A_2}\}, E\rangle\rangle$	$\rightarrow \langle\langle R, E \cup \{A_1 = \tau'.A_2\}\rangle\rangle$
(Cons)	$\langle\langle R \cup \{(M^{A_1}_{\tau_1}.S^{A_2}_{A_3})^{A_4}_{A_5}\}, E\rangle\rangle$	$\rightarrow \langle\langle R, E \cup \{A_1 = A_2, A_2 = A_4, A_5 = \tau_1.A_3\}\rangle\rangle$
(Comp)	$\langle\langle R \cup \{(S^{A_1}_{A_2} \circ T^{A_3}_{A_4})^{A_5}_{A_6}\}, E\rangle\rangle$	$\rightarrow \langle\langle R, E \cup \{A_1 = A_4, A_2 = A_6, A_3 = A_5\}\rangle\rangle$

Conclusions

- PT is not a trivial property.
- A system dependent definition of PT for the λ-calculus in de Bruijn notation was proposed and proved correct and $TA_{\lambda dB}$ was proved to satisfy the PT property.
- More importantly, dependent definitions of PT for λs_e and $\lambda\sigma$ were proposed and proved correct and both type systems were proved to satisfy the PT property.
- In all type systems cosidered the PT property was constructively proved based on the existence of type inference algorithms.
- Since $TA_{\lambda s_e}$ and $TA_{\lambda\sigma}$ respectively involve the treatment of a built-in arithmetic theory and *substitution* objects, establishing the PT property was not trivial.

M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy.
Explicit Substitutions.
Journal of Functional Programming, 1(4):375–416, 1991. Cambridge University Press.

M. Ayala-Rincón and C. Muñoz.
Explicit Substitutions and All That.
Revista Colombiana de Computación, 1(1):47–71, 2000.

P. Borovanský.
Implementation of Higher-Order Unification Based on Calculus of Explicit Substitutions.
LNCS: Proceedings of the 22nd Seminar on Current Trends in Theory and Practice of Informatics (SOFSEM'95), 1012:363–368, 1995. Springer Verlag.

N.G. de Bruijn.
Lambda-Calculus Notation with Nameless Dummies, a Tool for Automatic Formula Manipulation, with Application to the Church-Rosser Theorem.
Indagationes Mathematicae, 34(5):381–392, 1972.

L. Damas and R. Milner.
Principal Type-Schemes for Functional Programs.
ACM Symposium on Principles of Programming Languages (POPL'82), 207–212, 1982. ACM Press.

J. R. Hindley.
Basic Simple Type Theory.
Cambridge Tracts in Theoretical Computer Science, 42, 1997. Cambridge University Press.

T. Jim.
What are principal typings and what are they good for?
ACM Symposium on Principles of Programming Languages (POPL'96), 42–53, 1996. ACM Press.

A.J. Kfoury and J.B. Wells
Principality and type inference for intersection types using expansion variables,
Theoretical Computer Science, 311(1-3):1–70, 2004. Elsevier.

F. Kamareddine and A. Ríos.
Extending a λ-calculus with explicit substitution which preserves strong normalisation into a confluent calculus on open terms,
Journal of Functional Programming, 7:395–420, 1997. Cambridge University Press.

J.B. Wells
The essence of principal typings,
LNCS: Proceedings of the 29th International Colloquium on Automata, Languages and Programming, 2380:913–925, 2002. Springer-Verlag.