Tipando Cálculos de Substituições Explícitas

Mauricio Ayala-Rincón

Grupo de Teoria da Computação - GTC/UnB

Departamento de Matemática, Universidade de Brasília

Segundo Seminário Informal (, mas formal!) do GTC

Brasília, 28 de outubro de 2004

GTC/UnB: www.mat.unb.br/∼ayala/TCgroup

Summary

1. The Simply Typed λ-calculus
2. The Curry-Howard Isomorphism
3. The λ-calculus à la de Bruijn
4. Explicit Substitutions calculi: λs_e
5. Typing Explicit substitutions calculi
6. Type inference for the λs_e
7. Conclusions

1. The Simply Typed λ-calculus

- λ-terms, Λ: $a ::= (a \ \ a) \mid \lambda.a$

Type annotations:

$$\lambda x{:}A.M$$

Beta and Eta rules: $\begin{cases} (\lambda x : A.M \ \ N) \longrightarrow^{\beta} M\{N/x\} \\ \lambda x : A.(M \ \ x) \longrightarrow^{\eta} M, \text{ if } x \notin \mathcal{F}V(M) \end{cases}$

1. The Simply Typed λ-calculus

Typing judgment: $\Gamma \vdash M : A \equiv$ "M has type A in the *context* Γ"

A context is a list $x_1{:}A_1, \ldots, x_n{:}A_n$ of variable declarations.

$$\frac{x \notin \Gamma}{x{:}A, \Gamma \vdash x : A}\ \text{(Start)} \qquad \frac{x \notin \Gamma \quad \Gamma \vdash M : B}{x{:}A, \Gamma \vdash M : B}\ \text{(Weak)}$$

$$\frac{x{:}A, \Gamma \vdash M : B}{\Gamma \vdash \lambda x{:}A.M : A \to B}\ \text{(Abs)} \qquad \frac{\Gamma \vdash M : A \to B \quad \Gamma \vdash N : A}{\Gamma \vdash (M\ \ N) : B}\ \text{(Appl)}$$

Figure 1: The simply-typed λ-calculus

1. The Simply Typed λ-calculus

Relevant problems/notions in type theory:

- Type checking: Given M and A determine whether there exists Γ such that $\Gamma \vdash M : A$.

- Type inference: given M determine Γ and A such that $\Gamma \vdash M : A$.

- Type inhabitation: The type A is *inhabited* in Γ if and only if there exists a λ-term M such that $\Gamma \vdash M : A$.

2. The Curry-Howard Isomorphism

$\boxed{\text{Type theory}}$ versus $\boxed{\text{intuitionistic logic}}$

Typing rules of the simply-typed λ-calculus correspond one to one to deduction rules of minimal intuitionistic logic: typing rules are logical rules decorated with typed λ-terms.

A judgment $\Omega \vdash_I A$ denotes that A is a logical consequence of Ω.

$$\frac{}{\Omega, A \vdash_I A}\ \text{(Axiom)} \qquad \frac{\Omega, A \vdash_I B}{\Omega \vdash_I A \to B}\ \text{(Intro)} \qquad \frac{\Omega \vdash_I A \to B \quad \Omega \vdash_I A}{\Omega \vdash_I B}\ \text{(Elim)}$$

A formula A is a *tautology* if and only if the judgment $\vdash_I A$ is provable.

2. The Curry-Howard Isomorphism

Example $A \rightarrow ((A \rightarrow B) \rightarrow B)$ is a tautology:

$$\dfrac{\dfrac{\dfrac{\dfrac{}{A, A\rightarrow B \vdash_I A\rightarrow B}\text{(Axiom)} \quad \dfrac{}{A, A\rightarrow B \vdash_I A}\text{(Axiom)}}{A, A\rightarrow B \vdash_I B}\text{(Elim)}}{A \vdash_I (A\rightarrow B)\rightarrow B}\text{(Intro)}}{\vdash_I A\rightarrow((A\rightarrow B)\rightarrow A)}\text{(Intro)}$$

For this example:

$$\vdash \lambda x{:}A.\lambda y{:}A\rightarrow B.(y\ \ x) : A\rightarrow((A\rightarrow B)\rightarrow A)$$

2. The Curry-Howard Isomorphism

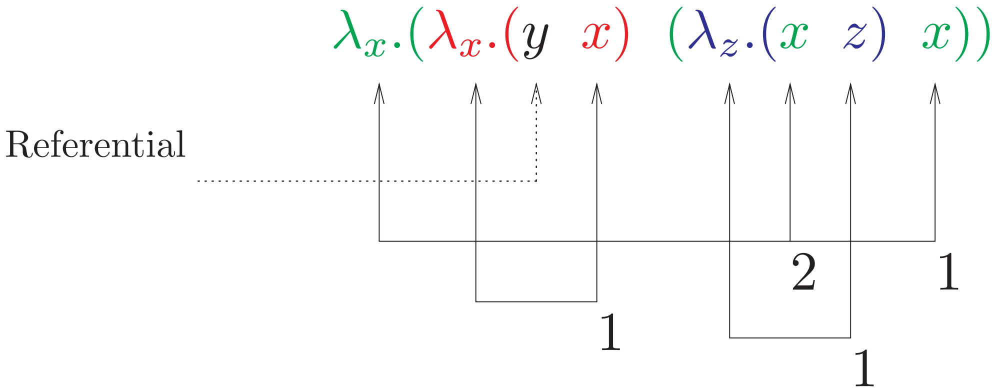
Curry-Howard isomorphism: $\Omega \vdash_I A$ is provable in the minimal intuitionistic logic if and only if $\Gamma \vdash M : A$ is a valid typing judgment in the simply-typed λ-calculus, where Γ is a list of variable declaration of propositions, seen as types, in Ω. The term M is a λ-term that represents the proof derivation.

- Type inference: given M determine Γ and A such that $\Gamma \vdash M : A$ corresponds to correctness proofs of programs

- Type inhabitation: The type A is *inhabited* in Γ if and only if there exists a λ-term M such that $\Gamma \vdash M : A$
  corresponds to extracting programs from proofs

3. The λ-calculus à la de Bruijn

- Terms in de Bruijn notation, Λ_{dB}: $a ::= \mathbb{N} \mid (a \;\; a) \mid \lambda.a$, where $\mathbb{N}$ is the set of de Bruijn indices.



$$\to_\beta \;\; \lambda_x.(y \;\; (\lambda_z.(x \;\; z) \;\; x))$$

$$\to_\beta \;\; \lambda_x.(y \;\; (x \;\; x))$$

For instance, for the referential $x, y, z, \ldots$: $\boxed{\lambda.(\lambda.(4 \;\; 1) \;\; (\lambda.(2 \;\; 1) \;\; 1))}$

3. The λ-calculus à la de Bruijn

β-reduction: $\boxed{\lambda.(\lambda.(4\ \ 1)\ \ (\lambda.(2\ \ 1)\ \ 1)) \to_\beta \lambda.(3\ \ (\lambda.(2\ \ 1)\ \ 1)) \to_\beta \lambda.(3\ \ (1\ \ 1))}$

3. The λ-calculus à la de Bruijn

In the de Bruijn setting of the simply typed λ-calculus, a context Γ is a list of types $A_1 \ldots . A_n$ where A_i is the type of the free-variable represented by the index $\underline{i}$.

$$\frac{1 \leq i \leq n}{A_1.A_2.\ldots.A_n \vdash \underline{i} : A_i}\ \text{(Var)} \qquad \frac{A.\Gamma \vdash M : B}{\Gamma \vdash \lambda_A.M : A{\rightarrow}B}\ \text{(Abs)}$$

$$\frac{\Gamma \vdash M : A{\rightarrow}B \qquad \Gamma \vdash N : A}{\Gamma \vdash (M\ \ N) : B}\ \text{(Appl)}$$

Figure 2: The simply-typed λ-calculus for Λ_{dB}-terms

4. Explicit Substitutions calculi: λs_e

Implicitness of substitution is the *Achilles heel* of the λ-calculus: the substitution involved in β-reductions does not belong in the calculus, but rather in an informal meta-level.

Definition 1. [$\boldsymbol{\lambda s_e}$-calculus] *The λs_e-calculus is given by the rewrite system in Fig. 3 and the grammar*

$$M, N ::= \underline{n} \mid (M \;\; N) \mid \lambda M \mid M \sigma^j N \mid \varphi^i_k M \quad \textit{for} \;\; n,\, j,\, i \geq 1 \;\; \textit{and} \;\; k \geq 0.$$

The calculus of substitutions associated with the λs_e-calculus, namely s_e, is the rewriting system generated by the set of rules $s_e = \lambda s_e - \{\sigma\text{-}generation, Eta\}$.

$$
\begin{array}{lcll}
(\lambda M \;\; N) & \longrightarrow & M\,\sigma^1\,N & (\sigma\text{-generation}) \\
(\lambda M)\,\sigma^i N & \longrightarrow & \lambda(M\,\sigma^{i+1}\,N) & (\sigma\text{-}\lambda\text{-transition}) \\
(M_1 \;\; M_2)\,\sigma^i N & \longrightarrow & ((M_1\,\sigma^i N) \;\; (M_2\,\sigma^i N)) & (\sigma\text{-app-transition}) \\
\underline{n}\,\sigma^i N & \longrightarrow & \begin{cases} \underline{n-1} & \text{if } \; n > i \\ \varphi_0^i\,N & \text{if } \; n = i \\ \underline{n} & \text{if } \; n < i \end{cases} & (\sigma\text{-destruction}) \\
\varphi_k^i(\lambda M) & \longrightarrow & \lambda(\varphi_{k+1}^i\,M) & (\varphi\text{-}\lambda\text{-transition}) \\
\varphi_k^i(M_1 \;\; M_2) & \longrightarrow & ((\varphi_k^i\,M_1) \;\; (\varphi_k^i\,M_2)) & (\varphi\text{-app-transition}) \\
\varphi_k^i\,\underline{n} & \longrightarrow & \begin{cases} \underline{n+i-1} & \text{if } \; n > k \\ \underline{n} & \text{if } \; n \leq k \end{cases} & (\varphi\text{-destruction}) \\
(M_1\,\sigma^i M_2)\,\sigma^j\,N & \longrightarrow & (M_1\,\sigma^{j+1}\,N)\;\sigma^i\,(M_2\,\sigma^{j-i+1}\,N) \quad \text{if } \; i \leq j & (\sigma\text{-}\sigma\text{-transition}) \\
(\varphi_k^i\,M)\,\sigma^j\,N & \longrightarrow & \varphi_k^{i-1}\,M \quad \text{if } \; k < j < k+i & (\sigma\text{-}\varphi\text{-transition 1}) \\
(\varphi_k^i\,M)\,\sigma^j\,N & \longrightarrow & \varphi_k^i(M\,\sigma^{j-i+1}\,N) \quad \text{if } \; k+i \leq j & (\sigma\text{-}\varphi\text{-transition 2}) \\
\varphi_k^i(M\,\sigma^j\,N) & \longrightarrow & (\varphi_{k+1}^i\,M)\,\sigma^j\,(\varphi_{k+1-j}^i\,N) \quad \text{if } \; j \leq k+1 & (\varphi\text{-}\sigma\text{-transition}) \\
\varphi_k^i\,(\varphi_l^j\,M) & \longrightarrow & \varphi_l^j\,(\varphi_{k+1-j}^i\,M) \quad \text{if } \; l+j \leq k & (\varphi\text{-}\varphi\text{-transition 1}) \\
\varphi_k^i\,(\varphi_l^j\,M) & \longrightarrow & \varphi_l^{j+i-1}\,M \quad \text{if } \; l \leq k < l+j & (\varphi\text{-}\varphi\text{-transition 2}) \\
\lambda(M \;\; \underline{1}) & \longrightarrow & N \quad \text{if } \; M =_{s_e} \varphi_0^2 N & (\text{Eta})
\end{array}
$$

Figure 3: Rewriting system of the λs_e-calculus

4. Explicit Substitutions calculi: λs_e

λs_e simula a β-contração:

$$(((\lambda(\lambda(\lambda(\lambda((\underline{4}\;\underline{2})\;((\underline{3}\;\underline{2})\;\underline{1}))))))\;(\lambda(\lambda(\underline{2}\;\underline{1}))))\;(\lambda(\lambda(\underline{2}\;\underline{1})))) \to_\beta ((\lambda(\lambda(\lambda(((\lambda(\lambda(\underline{2}\;\underline{1})))\;\underline{2})\;((\underline{3}\;\underline{2})\;\underline{1})))))\;(\lambda(\lambda(\underline{2}\;\underline{1}))))$$

Utilizando o sistema SUBSEXPL:

$$\underline{(((\lambda(\lambda(\lambda(\lambda((\underline{4}\;\underline{2})\;((\underline{3}\;\underline{2})\;\underline{1}))))))\;(\lambda(\lambda(\underline{2}\;\underline{1}))))\;(\lambda(\lambda(\underline{2}\;\underline{1}))))} \to_{\sigma\text{- gen}}$$

$$(\underline{((\lambda(\lambda(\lambda((\underline{4}\;\underline{2})\;((\underline{3}\;\underline{2})\;\underline{1})))))\sigma^1(\lambda(\lambda(\underline{2}\;\underline{1}))))}\;(\lambda(\lambda(\underline{2}\;\underline{1})))) \to_{\sigma\lambda}$$

$$((\lambda\underline{((\lambda(\lambda((\underline{4}\;\underline{2})\;((\underline{3}\;\underline{2})\;\underline{1}))))\sigma^2(\lambda(\lambda(\underline{2}\;\underline{1}))))})\;(\lambda(\lambda(\underline{2}\;\underline{1})))) \to_{\sigma\lambda}$$

$$((\lambda(\lambda\underline{((\lambda((\underline{4}\;\underline{2})\;((\underline{3}\;\underline{2})\;\underline{1})))\sigma^3(\lambda(\lambda(\underline{2}\;\underline{1}))))}))\;(\lambda(\lambda(\underline{2}\;\underline{1})))) \to_{\sigma\lambda}$$

$$((\lambda(\lambda(\lambda\underline{(((\underline{4}\;\underline{2})\;((\underline{3}\;\underline{2})\;\underline{1}))\sigma^4(\lambda(\lambda(\underline{2}\;\underline{1}))))})))\;(\lambda(\lambda(\underline{2}\;\underline{1})))) \to_{\sigma\text{- app}}$$

$$((\lambda(\lambda(\lambda(\underline{((\underline{4}\;\underline{2})\sigma^4(\lambda(\lambda(\underline{2}\;\underline{1}))))}\;(((\underline{3}\;\underline{2})\;\underline{1})\sigma^4(\lambda(\lambda(\underline{2}\;\underline{1})))))))))\;(\lambda(\lambda(\underline{2}\;\underline{1})))) \to_{\sigma\text{- app}}$$

$$((\lambda(\lambda(\lambda(((\underline{4}\sigma^4(\lambda(\lambda(\underline{2}\;\underline{1}))))\;(\underline{2}\sigma^4(\lambda(\lambda(\underline{2}\;\underline{1})))))\;\underline{(((\underline{3}\;\underline{2})\;\underline{1})\sigma^4(\lambda(\lambda(\underline{2}\;\underline{1}))))})))))\;(\lambda(\lambda(\underline{2}\;\underline{1})))) \to_{\sigma\text{- app}}$$

$$((\lambda(\lambda(\lambda(((\underline{4}\sigma^4(\lambda(\lambda(\underline{2}\ \underline{1})))) (\underline{2}\sigma^4(\lambda(\lambda(\underline{2}\ \underline{1}))))) (\underline{((\underline{3}\ \underline{2})\sigma^4(\lambda(\lambda(\underline{2}\ \underline{1}))))}\ (\underline{1}\sigma^4(\lambda(\lambda(\underline{2}\ \underline{1}))))))))) (\lambda(\lambda(\underline{2}\ \underline{1})))) \to_{\sigma\text{- app}}$$

$$((\lambda(\lambda(\lambda((\underline{(\underline{4}\sigma^4(\lambda(\lambda(\underline{2}\ \underline{1}))))}\ (\underline{2}\sigma^4(\lambda(\lambda(\underline{2}\ \underline{1}))))) (((\underline{3}\sigma^4(\lambda(\lambda(\underline{2}\ \underline{1})))) (\underline{2}\sigma^4(\lambda(\lambda(\underline{2}\ \underline{1}))))) (\underline{1}\sigma^4(\lambda(\lambda(\underline{2}\ \underline{1}))))))))) (\lambda(\lambda(\underline{2}\ \underline{1})))) \to_{\sigma\text{- des}}$$

$$((\lambda(\lambda(\lambda((\varphi^4_0((\lambda(\lambda(\underline{2}\ \underline{1})))) \underline{(\underline{2}\sigma^4(\lambda(\lambda(\underline{2}\ \underline{1}))))}) (((\underline{3}\sigma^4(\lambda(\lambda(\underline{2}\ \underline{1})))) (\underline{2}\sigma^4(\lambda(\lambda(\underline{2}\ \underline{1}))))) (\underline{1}\sigma^4(\lambda(\lambda(\underline{2}\ \underline{1}))))))))) (\lambda(\lambda(\underline{2}\ \underline{1})))) \to_{\sigma\text{- des}}$$

$$((\lambda(\lambda(\lambda((\varphi^4_0((\lambda(\lambda(\underline{2}\ \underline{1})))) \underline{2}) ((\underline{(\underline{3}\sigma^4(\lambda(\lambda(\underline{2}\ \underline{1}))))}\ (\underline{2}\sigma^4(\lambda(\lambda(\underline{2}\ \underline{1}))))) (\underline{1}\sigma^4(\lambda(\lambda(\underline{2}\ \underline{1}))))))))) (\lambda(\lambda(\underline{2}\ \underline{1})))) \to_{\sigma\text{- des}}$$

$$((\lambda(\lambda(\lambda((\varphi^4_0((\lambda(\lambda(\underline{2}\ \underline{1})))) \underline{2}) ((\underline{3}\ \underline{(\underline{2}\sigma^4(\lambda(\lambda(\underline{2}\ \underline{1}))))}) (\underline{1}\sigma^4(\lambda(\lambda(\underline{2}\ \underline{1}))))))))) (\lambda(\lambda(\underline{2}\ \underline{1})))) \to_{\sigma\text{- des}}$$

$$((\lambda(\lambda(\lambda((\varphi^4_0((\lambda(\lambda(\underline{2}\ \underline{1})))) \underline{2}) ((\underline{3}\ \underline{2})\ \underline{(\underline{1}\sigma^4(\lambda(\lambda(\underline{2}\ \underline{1}))))}))))) (\lambda(\lambda(\underline{2}\ \underline{1})))) \to_{\sigma\text{- des}}$$

$$((\lambda(\lambda(\lambda((\underline{\varphi^4_0((\lambda(\lambda(\underline{2}\ \underline{1}))))}\ \underline{2}) ((\underline{3}\ \underline{2})\ \underline{1}))))) (\lambda(\lambda(\underline{2}\ \underline{1})))) \to_{\varphi\lambda}$$

$$((\lambda(\lambda(\lambda(((\lambda\underline{\varphi^4_1((\lambda(\underline{2}\ \underline{1}))))}\ \underline{2}) ((\underline{3}\ \underline{2})\ \underline{1}))))) (\lambda(\lambda(\underline{2}\ \underline{1})))) \to_{\varphi\lambda}$$

$$((\lambda(\lambda(\lambda(((\lambda(\lambda\underline{\varphi^4_2((\underline{2}\ \underline{1}))}))\ \underline{2}) ((\underline{3}\ \underline{2})\ \underline{1}))))) (\lambda(\lambda(\underline{2}\ \underline{1})))) \to_{\varphi\text{- app}}$$

$$((\lambda(\lambda(\lambda(((\lambda(\lambda(\underline{\varphi^4_2(\underline{2})}\ \varphi^4_2(\underline{1}))))\ \underline{2}) ((\underline{3}\ \underline{2})\ \underline{1}))))) (\lambda(\lambda(\underline{2}\ \underline{1})))) \to_{\varphi\text{- des}}$$

$$((\lambda(\lambda(\lambda(((\lambda(\lambda(\underline{2}\ \underline{\varphi^4_2(\underline{1})})))\ \underline{2}) ((\underline{3}\ \underline{2})\ \underline{1}))))) (\lambda(\lambda(\underline{2}\ \underline{1})))) \to_{\varphi\text{- des}}$$

$$((\lambda(\lambda(\lambda(((\lambda(\lambda(\underline{2}\ \underline{1})))\ \underline{2}) ((\underline{3}\ \underline{2})\ \underline{1}))))) (\lambda(\lambda(\underline{2}\ \underline{1}))))$$

5. Typing Explicit substitutions calculi

Rewriting rules of the λs_e-calculi are modified adding typing information:

$$
\begin{array}{llll}
(\lambda_A.M \;\; N) & \longrightarrow & M\,\sigma^1\,N & (\sigma\text{-generation}) \\
(\lambda_A.M)\,\sigma^i N & \longrightarrow & \lambda_A.(M\,\sigma^{i+1}\,N) & (\sigma\text{-}\lambda\text{-transition}) \\
\varphi^i_k(\lambda_A.M) & \longrightarrow & \lambda_A.(\varphi^i_{k+1}\,M) & (\varphi\text{-}\lambda\text{-transition}) \\
\lambda_A.(M \;\; \underline{1}) & \longrightarrow & N \quad \text{if} \;\; M =_{s_e} \varphi^2_0 N & (\text{Eta})
\end{array}
$$

5. Typing Explicit substitutions calculi

$$\frac{}{A.\Gamma \vdash \underline{1} : A} \text{ (Var)} \qquad \frac{\Gamma \vdash \underline{n} : B}{A.\Gamma \vdash \underline{n+1} : B} \text{ (Varn)}$$

$$\frac{A.\Gamma \vdash N : B}{\Gamma \vdash \lambda_A.N : A \rightarrow B} \text{ (Lambda)} \qquad \frac{\Gamma \vdash N : A \rightarrow B \quad \Gamma \vdash M : A}{\Gamma \vdash (N \;\; M) : B} \text{ (App)}$$

$$\frac{\Gamma_{\geq i} \vdash N : B \quad \Gamma_{<i}.B.\Gamma_{\geq i} \vdash M : A}{\Gamma \vdash M\, \sigma^i N : A} \text{ (Sigma)} \qquad \frac{\Gamma_{<k}.\Gamma_{>k+i} \vdash M : A}{\Gamma \vdash \varphi^i_k M : A} \text{ (Phi)}$$

Figure 4: Typing rules for the λs_e-calculus

Let Γ be a context of the form $A_1.A_2 \ldots A_n.\Delta$. We use the notation $\Gamma_{\leq k}$ and $\Gamma_{\geq k}$ for denoting the contexts $A_1 \ldots A_k$ and $A_k \ldots A_n.\Delta$, respectively. This notation is extended for "$<$" and "$>$" in the obvious manner.

5. Typing Explicit substitutions calculi

Example Type inference for the term $\lambda_{A\to B}.\lambda_{B\to C}.\lambda_A.(\underline{2}\ \ (\underline{3}\ \ \underline{1}))$ in λs_e.

Let $\Gamma = A.B{\to}C.A{\to}B$.

$$\dfrac{}{(1)\ \Gamma \vdash \underline{1} : A}\,(\mathsf{Var}) \qquad \dfrac{\dfrac{}{B{\to}C.A{\to}B \vdash \underline{1} : B{\to}C}\,(\mathsf{Var})}{(2)\ \Gamma \vdash \underline{2} : B{\to}C}\,(\mathsf{Varn}) \qquad \dfrac{\dfrac{\dfrac{}{A{\to}B \vdash \underline{1} : A{\to}B}\,(\mathsf{Var})}{B{\to}C.A{\to}B \vdash \underline{2} : A{\to}B}\,(\mathsf{Varn})}{(3)\ \Gamma \vdash \underline{3} : A{\to}B}\,(\mathsf{Varn})$$

$$\dfrac{(2) \qquad \dfrac{(3) \quad (1)}{\Gamma \vdash (\underline{3}\ \ \underline{1}) : B}\,(\mathsf{App})}{\Gamma \vdash (\underline{2}\ \ (\underline{3}\ \ \underline{1})) : C}\,(\mathsf{App})$$

5. Typing Explicit substitutions calculi

Example Continuation.

$$
\cfrac{\cfrac{\cfrac{\Gamma \vdash (\underline{2}\ \ (\underline{3}\ \ \underline{1})) : C}{B{\rightarrow}C . A{\rightarrow}B \vdash \lambda_A.(\underline{2}\ \ (\underline{3}\ \ \underline{1})) : A{\rightarrow}C}\ \text{(Lambda)}}{A{\rightarrow}B \vdash \lambda_{B\rightarrow C}.\lambda_A.(\underline{2}\ \ (\underline{3}\ \ \underline{1})) : (B{\rightarrow}C){\rightarrow}(A{\rightarrow}C)}\ \text{(Lambda)}}{\vdash \lambda_{A\rightarrow B}.\lambda_{B\rightarrow C}.\lambda_A.(\underline{2}\ \ (\underline{3}\ \ \underline{1})) : (A{\rightarrow}B){\rightarrow}((B{\rightarrow}C){\rightarrow}(A{\rightarrow}C))}\ \text{(Lambda)}
$$

6. Type inference for the λs_e

Type variables τ_i and context variables γ_i, $i \in \mathbb{N}$.

$$\underbrace{\underset{\text{input}}{\boxed{\lambda\text{-term}\, M_\tau^\gamma}} \quad \leadsto \quad \underset{\text{output}}{\boxed{1^{st}\text{-order unification problem on } \tau_i' s \text{ and } \gamma' s}}}_{\text{Type inference algorithm}}$$

6. Type inference for the λs_e

λs_e-terms are *decorated* with new type and context variables.

Example

$$\underbrace{\boxed{\lambda_A.\lambda_B.\lambda_C.(\underline{2}\ \ (\underline{3}\ \ \underline{1}))}}_{M} \rightsquigarrow \underbrace{\boxed{(\lambda_A.(\lambda_B.(\lambda_C.(\underline{2}^{\gamma_1}_{\tau_1}\ \ (\underline{3}^{\gamma_2}_{\tau_2}\ \ \underline{1}^{\gamma_3}_{\tau_3})^{\gamma_4}_{\tau_4})^{\gamma_5}_{\tau_5})^{\gamma_6}_{\tau_6})^{\gamma_7}_{\tau_7})^{\gamma_8}_{\tau_8}}}_{M'}$$

where τ_i and γ_i, $i = 1, ..., 8$ are new mutually different type and context variables.

6. Type inference for the λs_e

The kernel of the algorith is given by the set of *transformation rules* in Table 1.

Applying the transformations rules we obtain a sequence:

$$\boxed{\langle R_0, \emptyset\rangle \leadsto \langle R_1, E_1\rangle \leadsto \cdots}$$

where the R's and E's are sets of decorated terms equations on type and context variables.

The application starts from $\langle R_0, \emptyset\rangle$, where R_0 is the set of all decorated subterms of M'.

Table 1: Transformation rules for type inference in the λs_e-calculus

(Var)	$\langle R \cup \{\underline{1}_\tau^\gamma\}, E\rangle$	$\rightarrow$	$\langle R, E \cup \{\gamma = \tau.\gamma'\}\rangle$, where γ' is a fresh context variable;
(Varn)	$\langle R \cup \{\underline{n}_\tau^\gamma\}, E\rangle$	$\rightarrow$	$\langle R, E \cup \{\gamma = \tau_1'\ldots\tau_{n-1}'.\tau.\gamma'\}\rangle$, where γ' and $\tau_1', \ldots, \tau_{n-1}'$ are fresh context and type variables;
(Lambda)	$\langle R \cup \{(\lambda_A.M_{\tau_1}^{\gamma_1})_{\tau_2}^{\gamma_2}\}, E\rangle$	$\rightarrow$	$\langle R, E \cup \{\tau_2 = A\rightarrow\tau_1, \gamma_1 = A.\gamma_2\}\rangle$;
(App)	$\langle R \cup \{(M_{\tau_1}^{\gamma_1}\ N_{\tau_2}^{\gamma_2})_{\tau_3}^{\gamma_3}\}, E\rangle$	$\rightarrow$	$\langle R, E \cup \{\gamma_1 = \gamma_2, \gamma_2 = \gamma_3, \tau_1 = \tau_2\rightarrow\tau_3\}\rangle$;
(Sigma)	$\langle R \cup \{(M_{\tau_1}^{\gamma_1}\sigma^i N_{\tau_2}^{\gamma_2})_{\tau_3}^{\gamma_3}\}, E\rangle$	$\rightarrow$	$\langle R, E \cup \{\tau_1 = \tau_3, \gamma_1 = \tau_1'\ldots\tau_{i-1}'.\tau_2.\gamma_2, \gamma_3 = \tau_1'\ldots\tau_{i-1}'.\gamma_2\}\rangle$, where $\tau_1', \ldots, \tau_{i-1}'$ are fresh type variables and in the case that $i = 1$ the sequence $\tau_1'\ldots\tau_{i-1}'$ is empty;
(Phi)	$\langle R \cup \{(\varphi_k^i M_{\tau_1}^{\gamma_1})_{\tau_2}^{\gamma_2}\}, E\rangle$	$\rightarrow$	$\langle R, E \cup \{\tau_1 = \tau_2, \gamma_2 = \tau_1'\ldots\tau_{k+i-1}'.\gamma', \gamma_1 = \tau_1'\ldots\tau_{k-1}'.\gamma'\}\rangle$, where γ' and $\tau_1', \ldots, \tau_{k+i-1}'$ are fresh context and type variables and in the case that $k \leq 1$ respectively $k = 0$ and $i = 1$ the sequences $\tau_1'\ldots\tau_{k-1}'$ respectively $\tau_1'\ldots\tau_{k+i-1}'$ are empty;
(Meta)	$\langle R \cup \{X_\tau^\gamma\}, E\rangle$	$\rightarrow$	$\langle R, E \cup \{\gamma = \Gamma_X, \tau = A_X\}\rangle$, where $\Gamma_X \vdash X : A_X$;

6. Type inference for the λs_e

Transformation rules are built according to the typing rules of the λs_e-calculus.

$$\boxed{\langle R_0, \emptyset \rangle \leadsto^f \langle \emptyset, E_f \rangle}$$

The size of the set of decorated subterms R_i decreases by one.
Consequently f corresponds exactly to the number of subterms in M.

The output $\boxed{E_f}$ is a first-order unification problem.

6. Type inference for the λs_e

$\boxed{\text{Any first-order unification algorithm is then applied to } E_f}$

- If the unification algorithm fails then our term is ill-typed.

- Otherwise, the resulting *mgu* gives a context Γ and a type A such that $\Gamma \vdash M : A$.

6. Type inference for the λs_e

Theorem 1. *The type inference algorithm for λs_e is correct and complete.*

Proof: Consequence of the correctness and completeness of first-order unification and of the correspondence between the typing rules and the transformation rules of the λs_e-calculus. $\diamond$

6. Type inference for the λs_e

Example

$$\underbrace{\boxed{\lambda_A.\lambda_B.\lambda_C.(\underline{2}\ \ (\underline{3}\ \ \underline{1}))}}_{M} \rightsquigarrow \underbrace{\boxed{(\lambda_A.(\lambda_B.(\lambda_C.(\underline{2}^{\gamma_1}_{\tau_1}\ \ (\underline{3}^{\gamma_2}_{\tau_2}\ \ \underline{1}^{\gamma_3}_{\tau_3})^{\gamma_4}_{\tau_4})^{\gamma_5}_{\tau_5})^{\gamma_6}_{\tau_6})^{\gamma_7}_{\tau_7})^{\gamma_8}_{\tau_8}}}_{M'}$$

Initial input for the transformation rules: $\langle R_0, \emptyset \rangle$, where

$$R_0 = \left\{ \begin{array}{l} \underline{2}^{\gamma_1}_{\tau_1},\ \ \underline{3}^{\gamma_2}_{\tau_2},\ \ \underline{1}^{\gamma_3}_{\tau_3},\ \ (\underline{3}^{\gamma_2}_{\tau_2}\ \ \underline{1}^{\gamma_3}_{\tau_3})^{\gamma_4}_{\tau_4},\ \ (\underline{2}^{\gamma_1}_{\tau_1}\ \ (\underline{3}^{\gamma_2}_{\tau_2}\ \ \underline{1}^{\gamma_3}_{\tau_3})^{\gamma_4}_{\tau_4})^{\gamma_5}_{\tau_5}, \\ (\lambda_C.(\underline{2}^{\gamma_1}_{\tau_1}\ \ (\underline{3}^{\gamma_2}_{\tau_2}\ \underline{1}^{\gamma_3}_{\tau_3})^{\gamma_4}_{\tau_4})^{\gamma_5}_{\tau_5})^{\gamma_6}_{\tau_6},\ \ (\lambda_B.(\lambda_C.(\underline{2}^{\gamma_1}_{\tau_1}\ \ (\underline{3}^{\gamma_2}_{\tau_2}\ \underline{1}^{\gamma_3}_{\tau_3})^{\gamma_4}_{\tau_4})^{\gamma_5}_{\tau_5})^{\gamma_6}_{\tau_6})^{\gamma_7}_{\tau_7}, \\ (\lambda_A.(\lambda_B.(\lambda_C.(\underline{2}^{\gamma_1}_{\tau_1}\ \ (\underline{3}^{\gamma_2}_{\tau_2}\ \ \underline{1}^{\gamma_3}_{\tau_3})^{\gamma_4}_{\tau_4})^{\gamma_5}_{\tau_5})^{\gamma_6}_{\tau_6})^{\gamma_7}_{\tau_7})^{\gamma_8}_{\tau_8}\} \end{array} \right\}$$

6. Type inference for the λs_e

Example Continuation.

$\langle R_0, \emptyset \rangle \quad \rightarrow_{Var}$

$\langle R_1 = R_0 \setminus \{\underline{1}^{\gamma_3}_{\tau_3}\}, E_1 = \{\gamma_3 = \tau_3.\gamma'_1\} \rangle \quad \rightarrow_{Varn}$

$\langle R_2 = R_1 \setminus \{\underline{2}^{\gamma_1}_{\tau_1}\}, E_2 = E_1 \cup \{\gamma_1 = \tau'_1.\tau_1.\gamma'_2\} \rangle \quad \rightarrow_{Varn}$

$\langle R_3 = R_2 \setminus \{\underline{3}^{\gamma_2}_{\tau_2}\}, E_3 = E_2 \cup \{\gamma_2 = \tau'_2.\tau'_3.\tau_2.\gamma'_3\} \rangle \quad \rightarrow_{App}$

$\langle R_4 = R_3 \setminus \{(\underline{3}^{\gamma_2}_{\tau_2} \ \underline{1}^{\gamma_3}_{\tau_3})^{\gamma_4}_{\tau_4}\}, E_4 = E_3 \cup \{\gamma_2 = \gamma_3, \gamma_3 = \gamma_4, \tau_2 = \tau_3 \rightarrow \tau_4\} \rangle \quad \rightarrow_{App}$

$\langle R_5 = R_4 \setminus \{(\underline{2}^{\gamma_1}_{\tau_1} \ (\underline{3}^{\gamma_2}_{\tau_2} \ \underline{1}^{\gamma_3}_{\tau_3})^{\gamma_4}_{\tau_4})^{\gamma_5}_{\tau_5}\}, E_5 = E_4 \cup \{\gamma_1 = \gamma_4, \gamma_4 = \gamma_5, \tau_1 = \tau_4 \rightarrow \tau_5\} \rangle \quad \rightarrow_{Lambda}$

$\langle R_6 = R_5 \setminus \{(\lambda_C.(\underline{2}^{\gamma_1}_{\tau_1} \ (\underline{3}^{\gamma_2}_{\tau_2} \ \underline{1}^{\gamma_3}_{\tau_3})^{\gamma_4}_{\tau_4})^{\gamma_5}_{\tau_5})^{\gamma_6}_{\tau_6}\}, E_6 = E_5 \cup \{\tau_6 = C \rightarrow \tau_5, \gamma_5 = C.\gamma_6\} \rangle \quad \rightarrow_{Lambda}$

$\langle R_7 = R_6 \setminus \{(\lambda_B.(\lambda_C.(\underline{2}^{\gamma_1}_{\tau_1} \ (\underline{3}^{\gamma_2}_{\tau_2} \ \underline{1}^{\gamma_3}_{\tau_3})^{\gamma_4}_{\tau_4})^{\gamma_5}_{\tau_5})^{\gamma_6}_{\tau_6})^{\gamma_7}_{\tau_7}\}, E_7 = E_6 \cup \{\tau_7 = B \rightarrow \tau_6, \gamma_6 = B.\gamma_7\} \rangle \quad \rightarrow_{Lambda}$

$\langle \emptyset = R_7 \setminus \{(\lambda_A.(\lambda_B.(\lambda_C.(\underline{2}^{\gamma_1}_{\tau_1} \ (\underline{3}^{\gamma_2}_{\tau_2} \ \underline{1}^{\gamma_3}_{\tau_3})^{\gamma_4}_{\tau_4})^{\gamma_5}_{\tau_5})^{\gamma_6}_{\tau_6})^{\gamma_7}_{\tau_7})^{\gamma_8}_{\tau_8}\}, E_8 = E_7 \cup \{\tau_8 = A \rightarrow \tau_7, \gamma_7 = A.\gamma_8\} \rangle$

6. Type inference for the λs_e

Exercise Continuation. Apply any first-order unification algorithm to the problem:

$$E_8 = \left\{ \begin{array}{l} \gamma_3 = \tau_3.\gamma_1', \\ \gamma_1 = \tau_1'.\tau_1.\gamma_2', \\ \gamma_2 = \tau_2'.\tau_3'.\tau_2.\gamma_3', \\ \gamma_2 = \gamma_3, \gamma_3 = \gamma_4, \tau_2 = \tau_3 \rightarrow \tau_4, \\ \gamma_1 = \gamma_4, \gamma_4 = \gamma_5, \tau_1 = \tau_4 \rightarrow \tau_5, \\ \tau_6 = C \rightarrow \tau_5, \gamma_5 = C.\gamma_6, \\ \tau_7 = B \rightarrow \tau_6, \gamma_6 = B.\gamma_7, \\ \tau_8 = A \rightarrow \tau_7, \gamma_7 = A.\gamma_8 \end{array} \right\}$$

And then resolve the bindings of the resulting unifier (if it exists) for giving appropriate contexts and types for the input λ-term.

7. Conclusions

- Explicit substitutions calculi close the gap between theory and practice.
- Building a "satisfactory" substitutions calculus is an open problem.
- Types are essential when thinking about formal frameworks for reasoning about implementation of *programming languages* and *automated deduction environments*.
- *Type checking* and *type inference* algorithms belong to the kernel of any practical computational environment.
- *Type inhabitation* methods are essential for extracting the computational information of correctness proofs of specifications.

*References

[1] M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Lévy. Explicit Substitutions. *J. of Func. Programming*, 1(4):375–416, 1991.

[2] M. Ayala-Rincón and F. Kamareddine. On Applying the λ_{se}-Style of Unification for Simply-Typed Higher Order Unification in the Pure lambda Calculus. *Matemática Contemporânea*, 24(1):1–22, 2003. WoLLIC'01 Special Issue, J. Baldwin, R. de Queiroz and E.H. Haeusler, Eds.

[3] P. Borovanský. Implementation of Higher-Order Unification Based on Calculus of Explicit Substitutions. In M. Bartošek, J. Staudek, and J. Wiedermann, editors, *Proceedings of the SOFSEM'95: Theory and Practice of Informatics*, volume 1012 of *LNCS*, pages 363–368. Springer Verlag, 1995.

[4] F. Kamareddine, A. Ríos, and J.B. Wells. Calculi of generalised β_e-reduction and explicit substitution: Type free and simply typed versions. *J. of Func. and Logic Programming*, 1998.