(Higher-Order) Unification via λs_e-style of explicit substitution

Mauricio Ayala-Rincón
Departamento de Matemática
Universidade de Brasília
Brasília D. F., Brasil

Fairouz Kamareddine
Computer and Electrical Engineering
Heriot-Watt University
Edinburgh, Scotland

PPDP 2000 — September 21, 2000

Talk's Plan

1. HOU in explicit substitution calculi
2. Unification in the λs_e-style of explicit substitution
3. Checking arithmetic constraints (versus shifts and composition in $\lambda\sigma$)
4. A simple example
5. Related work
6. Future work and Conclusions

1. HOU in explicit substitution calculi

HOU $\left\{\begin{array}{c}\text{Given two simply-typed lambda terms } a \text{ and } b \\ \text{find a } \textit{substitution } \theta \text{ such that} \\ \theta(a) =_{\beta\eta} \theta(b)\end{array}\right.$

- HOU essential for generalizations of the Robinson's first-order resolution principle.
- HOU applied in
  - Automated (Higher order) reasoning
  - Higher order proof assistants
  - Higher order logic programming

Why *making substitutions explicit* is adequate for reasoning about HOU?

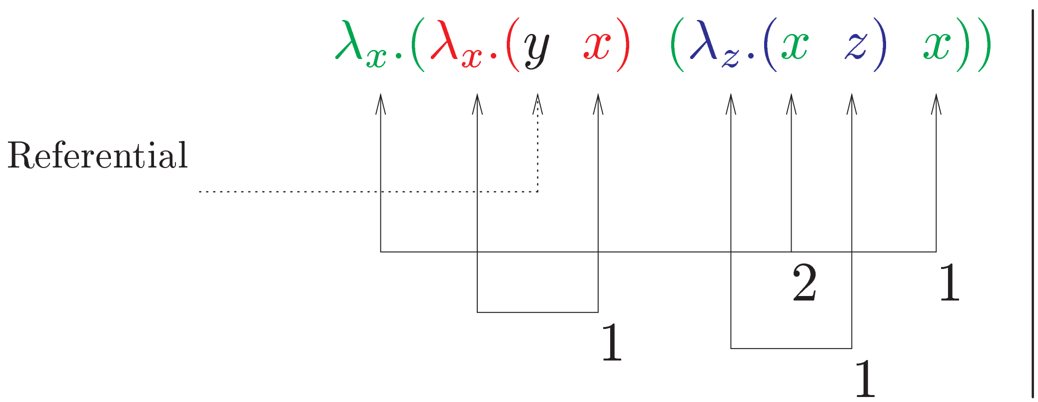
- Substitution is the key operation for HOU.

- *Implicitness* of substitution is the "Achilles heel" of the λ-calculus:
  - β-reduction is given via informal/implicit variable renaming

Implicit substitution does not provide any formal mechanism for analysing essential computational properties

  such as
  - time and
  - space complexity

• Terms in de Bruijn notation, $\Lambda_{dB}(\mathcal{X})$: $a ::= \mathbb{N} \mid \mathcal{X} \mid (a\ \ a) \mid \lambda.a$, where $\mathcal{X}$ meta-variables and $\mathbb{N}$ set of de Bruijn indices.



$$\rightarrow_\beta \ \lambda_x.(y\ \ (\lambda_z.(x\ \ z)\ \ x))$$
$$\rightarrow_\beta \ \lambda_x.(y\ \ (x\ \ x))$$

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

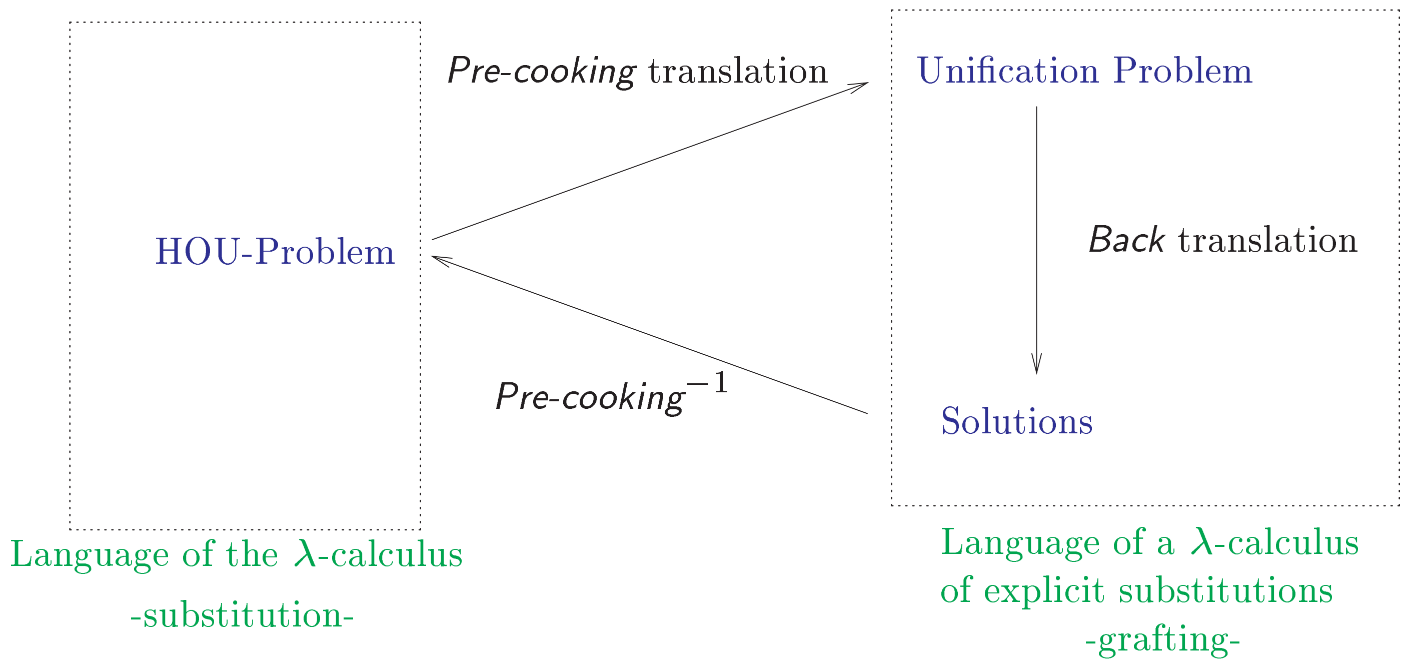
β-reduction: $\lambda.(\lambda.(4\ \ 1)\ \ (\lambda.(2\ \ 1)\ \ 1)) \rightarrow_\beta \lambda.(3\ \ (\lambda.(2\ \ 1)\ \ 1)) \rightarrow_\beta \lambda.(3\ \ (1\ \ 1))$

- Higher order *substitution*: $\{X/1\}(\lambda.(1\ \ X)\ \ X) = (\lambda.(1\ \ 2)\ \ 1)$

substitution	$\neq$	*grafting*
$\{X/a\}(\lambda.X)$		$(\lambda.X)\{X/a\}$
$\Vert$		$\Vert$
$\lambda.\{X/a^+\}X$		$\lambda.X\{X/a\}$
$\Vert$		$\Vert$
$\lambda.\underbrace{a^+}_{\text{lift}}$	$\neq$	$\lambda.a$

β-reduction

$$(\lambda.a\ \ b) \to \{1/b\}a$$



- Introduced by G. Dowek, T. Hardin and C. Kirchner using the $\lambda\sigma$-calculus.

- Subsumes Huet's HOU method.

2. Unification in the λs_e-style of explicit substitution

• Terms in λs_e: $a ::= \mathcal{X} \mid \mathbb{N} \mid (a\ \ a) \mid \lambda.a \mid a\sigma^j a \mid \varphi^i_k a$, for $j,\, i \geq 1\,,\ \ k \geq 0$ where $\mathcal{X}$ *meta-variables* and $\mathbb{N}$ set of de Bruijn indices.

• A λs_e-**unification problem** P is: $$\left\{ \bigvee_{j\in J} \underbrace{\exists \vec{w}_j \bigwedge_{i\in I_j} s_i =^?_{\lambda s_e} t_i}_{\text{unification system}} \right.$$

• A **unifier** of $\underbrace{\exists \vec{w} \bigwedge_{i\in I} s_i =^?_{\lambda s_e} t_i}_{\text{unification system}}$ is a **grafting** σ such that $\boxed{\exists \vec{w} \bigwedge_{i\in I} s_i\sigma = t_i\sigma}$

Example : $(\lambda.(\lambda.(X\ \ 2)\ \ 1)\ \ Y) \ =^?_{\lambda s_e}\ (\lambda.(Z\ \ 1)\ \ U)$

$\downarrow\ X, Z : A \to A;\ Y, U : A$

Normalize $((X\sigma^2 Y)\sigma^1(\varphi^1_0 Y)\ \ \varphi^1_0 Y) \ =^?_{\lambda s_e}\ (Z\sigma^1 U\ \ \varphi^1_0 U)$

$\downarrow$

Dec-App $(X\sigma^2 Y)\sigma^1(\varphi^1_0 Y) =^?_{\lambda s_e} Z\sigma^1 U \ \wedge\ \varphi^1_0 Y =^?_{\lambda s_e} \varphi^1_0 U$

$\downarrow$

Dec-φ $(X\sigma^2 Y)\sigma^1(\varphi^1_0 Y) =^?_{\lambda s_e} Z\sigma^1 U \ \wedge\ Y =^?_{\lambda s_e} U$

$\downarrow$

Replace $(X\sigma^2 Y)\sigma^1(\varphi^1_0 Y) =^?_{\lambda s_e} Z\sigma^1 Y \ \wedge\ Y =^?_{\lambda s_e} U$

$\downarrow_*$

Exp-λ + Replace $((\lambda.X')\sigma^2 Y)\sigma^1(\varphi^1_0 Y) =^?_{\lambda s_e} (\lambda.Z')\sigma^1 Y \ \wedge\ \left\{\begin{array}{l} Y =^?_{\lambda s_e} U \\ X =^?_{\lambda s_e} \lambda.X' \\ Z =^?_{\lambda s_e} \lambda.Z' \end{array}\right.$

$\downarrow_*$

Normalize + Dec-λ $(X'\sigma^3 Y)\sigma^2(\varphi^1_0 Y) =^?_{\lambda s_e} Z'\sigma^2 Y \ \wedge\ \left\{\begin{array}{l} Y =^?_{\lambda s_e} U \\ X =^?_{\lambda s_e} \lambda.X' \\ Z =^?_{\lambda s_e} \lambda.Z' \end{array}\right.$

- *Solved* equations:
- *Flex-Flex* equations:

$$\left.\begin{array}{l}\left\{\begin{array}{l} Y =^?_{\lambda s_e} U \\ X =^?_{\lambda s_e} \lambda.X' \\ Z =^?_{\lambda s_e} \lambda.Z' \end{array}\right. \\ (X'\sigma^3 Y)\sigma^2(\varphi^1_0 Y) =^?_{\lambda s_e} Z'\sigma^2 Y \end{array}\right\} \textit{Solved Forms}$$

• Solutions: $\{Y/X_1, U/X_1\} \bigcup$ solutions for X and Z given by the *Flex-Flex* equation.

Take, for instance, $\{Y/X_1, U/X_1\} \bigcup \{X/\lambda.\mathtt{n+1}, Z/\lambda.\mathtt{n}\}$ with $n > 2$:

$\underline{(\lambda.(\lambda.(\lambda.\mathtt{n+1}\ \ \mathtt{2})\ \ \mathtt{1})\ \ X_1)} \to_\beta (\lambda.(\lambda.\mathtt{n}\ \ \mathtt{2})\ \ X_1) \to_\beta (\lambda.\mathtt{n-1}\ \ X_1) \to_\beta \underline{\mathtt{n-2}}$

and

$\underline{(\lambda.(\lambda.\mathtt{n}\ \ \mathtt{1})\ \ X_1)} \to_\beta (\lambda.\mathtt{n-1}\ \ X_1) \to_\beta \underline{\mathtt{n-2}}$

- Correctness: If P reduces to P' then every unifier of P' is a unifier of P.
- Completeness: If P reduces to P' then every unifier of P is a unifier of P'.

Theorem [Correctness and Completeness]

The λs_e-unification rules are correct and complete.

3. Checking arithmetic constraints (versus shifts and composition in $\lambda\sigma$)

λs_e-calculus and λ-calculus $\rightarrow$ *Term* / *Substitution* } objects $\lambda\sigma$-calculus

λs_e uses all de Bruijn indices: $\mathbb{N}$

$\lambda\sigma$ uses only $\mathtt{1}$, "shift" and "composition": $\mathtt{n} \equiv \mathtt{1}[\underbrace{\uparrow \circ \cdots \circ \uparrow}_{n-1}]$

Exp-App $\lambda\sigma$-unification rule

$$P \wedge X[a_1 \ldots a_p . \uparrow^n] =^?_{\lambda\sigma} (\mathtt{m}\, b_1 \ldots b_q) \quad \rightarrow$$

$$\bigwedge \begin{cases} P \\ X[a_1 \ldots a_p . \uparrow^n] =^?_{\lambda\sigma} (\mathtt{m}\, b_1 \ldots b_q) \\ \bigvee_{r \in R_p \cup R_i} \exists H_1 \ldots H_k, X =^?_{\lambda\sigma} (\mathtt{r}\ H_1 \ldots H_k) \end{cases}$$

X not solved and atomic; $H_1, \ldots, H_k$ variables of appropriate types; $\Gamma_{H_i} = \Gamma_X$, $R_p \subseteq \{1, \ldots, p\}$ such that $(\mathtt{r}\ H_1 \ldots H_k)$ has the right type, $R_i =$ if $m \geq n+1$ then $\{m - n + p\}$ else $\emptyset$

Exp-App λs_e-unification rule

$$P \wedge \psi^{j_p}_{i_p} \ldots \psi^{j_1}_{i_1}(X, a_1, \ldots, a_p) =^?_{\lambda s_e} (\mathtt{m}\, b_1 \ldots b_q) \quad \rightarrow$$
$$\bigwedge \begin{cases} P \\ \psi^{j_p}_{i_p} \ldots \psi^{j_1}_{i_1}(X, a_1, \ldots, a_p) =^?_{\lambda s_e} (\mathtt{m}\, b_1 \ldots b_q) \\ \bigvee_{r \in R_p \cup R_i} \exists H_1, \ldots, H_k, X =^?_{\lambda s_e} (\mathtt{r}\ H_1 \ldots H_k) \end{cases}$$

$\psi^{j_p}_{i_p} \ldots \psi^{j_1}_{i_1}(X, a_1, \ldots, a_p)$ skeleton of a λs_e-normal term; X atomic and not solved; $\Gamma_{H_i} = \Gamma_X$, $R_p \subseteq \{i_1, \ldots, i_p\}$ of superscripts of the σ operator such that $(\mathtt{r}\ H_1 \ldots H_k)$ has the right type, $R_i = \bigcup_{k=0}^{p}$ if $i_k \geq m + p - k - \sum_{l=k+1}^{p} j_l > i_{k+1}$ then $\{m + p - k - \sum_{l=k+1}^{p} j_l\}$ else $\emptyset$, where $i_0 = \infty, i_{p+1} = 0$

In the $\lambda\sigma$-calculus

$X[a_1 \ldots a_p . \uparrow^n] =^?_{\lambda\sigma} (\mathtt{m}\, b_1 \ldots b_q)$ has solutions of the form:

$$\left(\mathtt{1}[\underbrace{\uparrow \circ \cdots \circ \uparrow}_{r-1}] \quad \underbrace{H_1 \quad \ldots \quad H_k}_{\text{of appropriate type}} \right)$$

$$\mathtt{1}[\underbrace{\uparrow \circ \cdots \circ \uparrow}_{r-1}] \quad [a_1 \ldots a_p . \uparrow^n] = \begin{cases} a_i, \text{ if } 1 \leq r = i \leq p \\ \mathtt{1}[\underbrace{\uparrow \circ \cdots \circ \uparrow}_{r-1-p}] \quad [\underbrace{\uparrow \circ \cdots \circ \uparrow}_{n}] \quad otherwise. \end{cases}$$

In the λs_e-calculus

$\psi_{k_p}^{j_p} \ldots \psi_{k_1}^{j_1}(X, a_1, \ldots, a_p) =^?_{\lambda s_e} (\mathtt{m}\ b_1 \ldots b_q)$ solutions of the form:

$$\left(\mathtt{n} \ \underbrace{H_1 \ \ \ldots \ \ H_k}_{\text{of appropriate type}} \right)$$

such that for some i, $\left[\begin{array}{c} k_{i+1} < n \leq k_i \\ \text{and} \\ n - (p - i) + \sum_{r=i+1}^{p} j_r = m \end{array} \right]$

4. Translations between the pure λ-calculus and the λs_e-calculus

- A unifier of $\lambda.X =_{\beta\eta} \lambda.a$ is not a $\{X/b\}$ such that $b =_{\beta\eta} a$:

$$\{X/b\}(\lambda.X) = \lambda.(\{X/b^+\}X) = \lambda.(X\{X/b^+\}) = \lambda.b^+$$

- The **pre-cooking** of a λ-term in de Bruijn notation into the λs_e-calculus is defined by $\boldsymbol{a_{pc}} = PC(a, 0)$ where $PC(a, n)$ is defined by:

1. $PC(\lambda_B.a, n) = \lambda_B.PC(a, n+1)$
2. $PC((a\ \ b), n) = (PC(a, n)\ \ PC(b, n))$
3. $PC(\mathtt{k}, n) = \mathtt{k}$
4. $PC(X, n) = \begin{cases} \text{if } n = 0 \text{ then } X \\ \text{else } \varphi_0^{n+1} X \end{cases}$

Proposition[Semantics of pre-cooking]

$$\underbrace{(\{X_1/b_1, \ldots, X_p/b_p\}(a))_{pc}}_{\text{Substitution}} = \underbrace{a_{pc}\{X_1/b_{1_{pc}}, \ldots, X_p/b_{p_{pc}}\}}_{\text{Grafting}}$$

Proposition[Correspondence between solutions]

$$\exists N_1,\ldots,N_p \quad \underbrace{\{X_1/N_1,\ldots,X_p/N_p\}}_{\text{substitution}}(a) \quad =_{\beta\eta} \quad \underbrace{\{X_1/N_1,\ldots,X_p/N_p\}}_{\text{substitution}}(b)$$

$$\Longleftrightarrow$$

$$\exists M_1,\ldots,M_p \quad a_{pc}\underbrace{\{X_1/M_1,\ldots,X_p/M_p\}}_{\text{grafting}} \quad =_{\lambda s_e} \quad b_{pc}\underbrace{\{X_1/M_1,\ldots,X_p/M_p\}}_{\text{grafting}}$$

5. A simple example

Problem: $\lambda.(X\ \ 2) =^?_{\beta\eta} \lambda.2, \quad 2 : A, \quad X : A \to A$

$$
\begin{array}{lr}
\lambda.(\varphi_0^2(X)\ \ 2) =^?_{\lambda s_e} \lambda.2 & \to \textit{Dec-}\lambda \\
(\varphi_0^2(X)\ \ 2) =^?_{\lambda s_e} 2 & \to \textit{Exp-}\lambda \\
\exists Y\, (\varphi_0^2(X)\ \ 2) =^?_{\lambda s_e} 2 \wedge X =^?_{\lambda s_e} \lambda.Y & \to \textit{Replace} \\
\exists Y\, (\varphi_0^2(\lambda.Y)\ \ 2) =^?_{\lambda s_e} 2 \wedge X =^?_{\lambda s_e} \lambda.Y & \to \textit{Normalize} \\
\exists Y\, (\varphi_1^2 Y)\sigma^1 2 =^?_{\lambda s_e} 2 \wedge X =^?_{\lambda s_e} \lambda.Y & \to \textit{Exp-app} \\
(\exists Y\, (\varphi_1^2 Y)\sigma^1 2 =^?_{\lambda s_e} 2 \wedge X =^?_{\lambda s_e} \lambda.Y) \wedge (Y =^?_{\lambda s_e} 1 \vee Y =^?_{\lambda s_e} 2) & \to \textit{Replace} \\
((\varphi_1^2 1)\sigma^1 2 =^?_{\lambda s_e} 2 \wedge X =^?_{\lambda s_e} \lambda.1) \vee ((\varphi_1^2 2)\sigma^1 2 =^?_{\lambda s_e} 2 \wedge X =^?_{\lambda s_e} \lambda.2) & \to \textit{Normalize} \\
(2 =^?_{\lambda s_e} 2 \wedge X =^?_{\lambda s_e} \lambda.1) \vee (2 =^?_{\lambda s_e} 2 \wedge X =^?_{\lambda s_e} \lambda.2) & \equiv \\
(X =^?_{\lambda s_e} \lambda.1) \vee (X =^?_{\lambda s_e} \lambda.2) &
\end{array}
$$

Problem: $\boxed{\lambda.(X\ \ 2) =^?_{\beta\eta} \lambda.2, \quad 2 : A, \ \ X : A \to A}$

Solutions: $\begin{cases} \{X/\lambda.1\} \\ \{X/\lambda.2\} \end{cases}$

Note that we have:

$$\{X/\lambda.1\}(\lambda.(X\ \ 2)) = \lambda.(\{X/(\lambda.1)^+\}(X)\ \ 2) = \lambda.(\lambda.1^{+1}\ \ 2) = \lambda.(\lambda.1\ \ 2) =_\beta \lambda.2$$

and

$$\{X/\lambda.2\}(\lambda.(X\ \ 2)) = \lambda.(\{X/(\lambda.2)^+\}(X)\ \ 2) = \lambda.(\lambda.2^{+1}\ \ 2) = \lambda.(\lambda.3\ \ 2) =_\beta \lambda.2$$

6. Related work

Our development of the λs_e-HOU was based on the ones of Dowek, Hardin and Kirchner for the $\lambda\sigma$-calculus of explicit substitutions.

One of our motivations was, in the practical setting of HOU, to compare the advantages and disadvantages of the two styles of explicit substitutions. This provides objective facts about that interesting theoretical question.

We think that our method can be adapted for applications in/for systems as the λProlog, Maude and ELAN.

Additional facts about the *back* transformation and practical considerations for an eventual implementation are available in Ayala-Rincón & Kamareddine *"On Applying λs_e-Style of Unification for Simply-Typed Higher Order Unification in the Pure λ-Calculus"* at `http://www.cee.hw.ac.uk/ultra/pubs.html`.

7. Future work and Conclusions

To be done:
- Prototype implementation.
- Comparison with the *suspension* calculus.

- $\lambda\sigma$-(HO)Unification and λs_e-(HO)Unification strategies don't differ.
- Pre-cooking (and back) translations in $\lambda\sigma$ and λs_e differ:
  - A simple selection of the scripts for the operators φ and σ in λs_e corresponds to the manipulation of substitution objects in the $\lambda\sigma$-HOU approach.
  - Use of all de Bruijn indices makes our approach simpler.

References

G. Dowek, T. Hardin, and C. Kirchner. *Higher-order Unification via Explicit Substitutions*, Information and Computation, 157(1/2):183-235, 2000.

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*, LNCS, 1012:363-368, 1995.

G. Nadathur and D.S. Wilson. *A Notation for Lambda Terms A Generalization of Environments*, Theoretical Computer Science, 198:49-98, 1998.

G. Nadathur. *A Fine-Grained Notation for Lambda Terms and Its Use in Intensional Operations*, The Journal of Functional and Logic Programming, 1999(2):1-62, 1999.