

# (Higher-Order) Unification via $\lambda s_e$ -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  $\lambda_{s_e}$ -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 { Given two simply-typed lambda terms  $a$  and  $b$   
 find a *substitution*  $\theta$  such that  
 $\theta(a) =_{\beta\eta} \theta(b)$

- 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

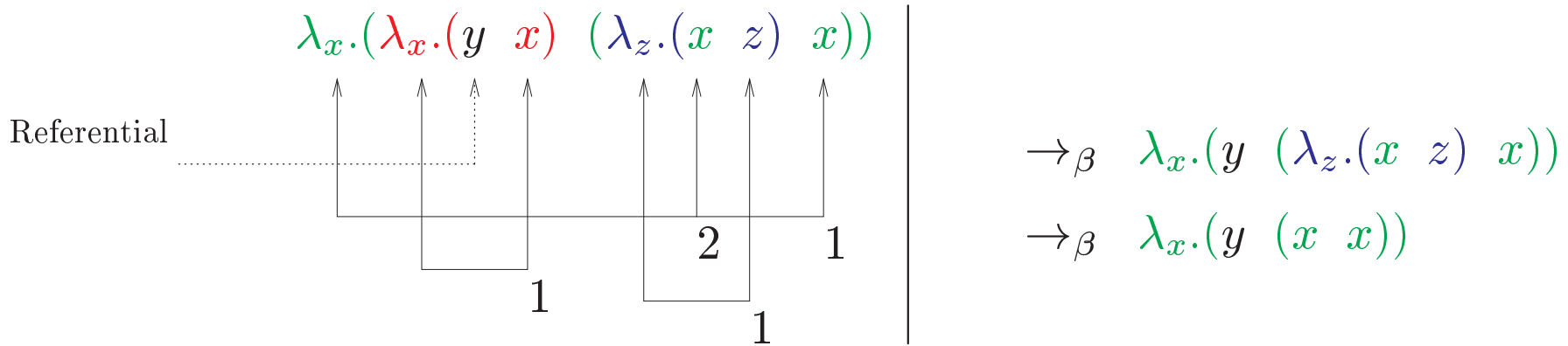
## Why *making substitutions explicit* is adequate for reasoning about HOU?

- Substitution is the key operation for HOU.
- *Implicitness* of substitution is the “Achilles heel” of the  $\lambda$ -calculus:
  - $\beta$ -reduction is given via informal/implicit variable renaming

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

such as  $\left\{ \begin{array}{l} - \text{time and} \\ - \text{space complexity} \end{array} \right.$

- 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.



For instance, for the referential  $x, y, z, \dots$ :

$$\lambda.(\lambda.(4 \ 1) \ (\lambda.(2 \ 1) \ 1))$$

$\beta$ -reduction:

$$\lambda.(\lambda.(4 \ 1) \ (\lambda.(2 \ 1) \ 1)) \rightarrow_{\beta} \lambda.(3 \ (\lambda.(2 \ 1) \ 1)) \rightarrow_{\beta} \lambda.(3 \ (1 \ 1))$$

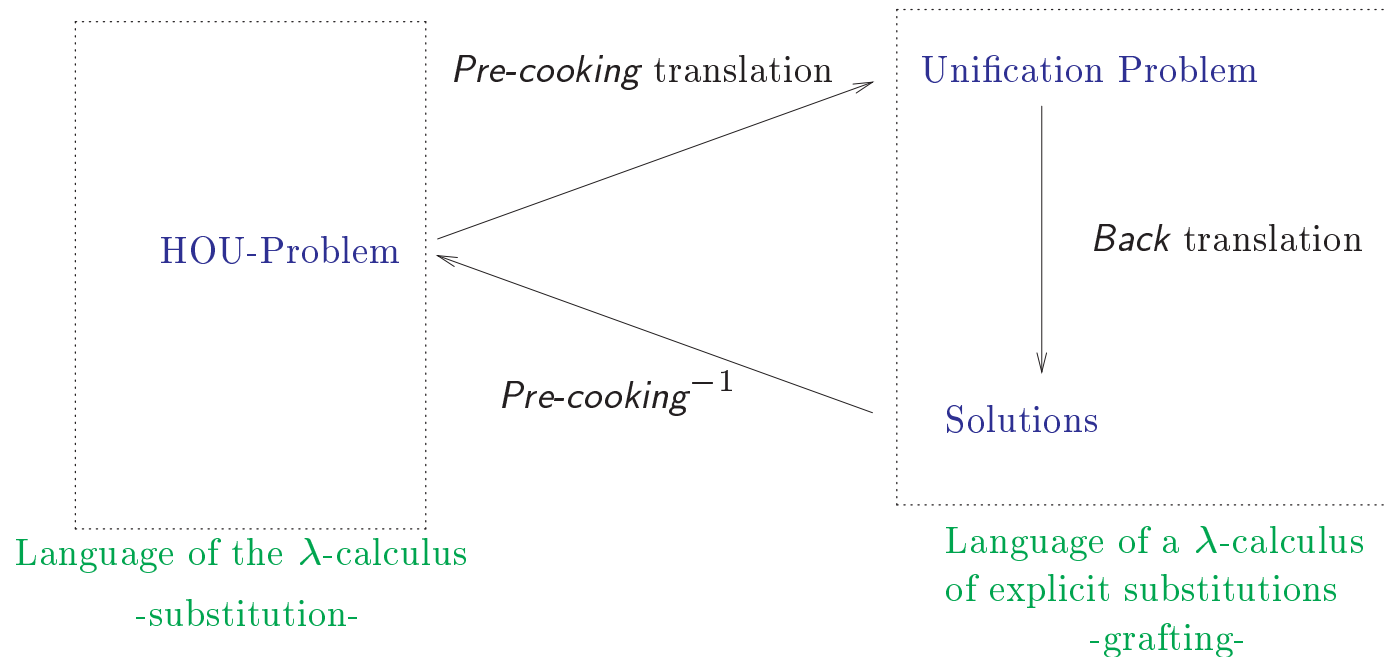
- Higher order *substitution*:

$$\{X/1\}(\lambda.(1 \ X) \ X) = (\lambda.(1 \ 2) \ 1)$$

substitution	$\neq$	<i>grafting</i>
$\{X/a\}(\lambda.X)$		$(\lambda.X)\{X/a\}$
$\parallel$		$\parallel$
$\lambda.\{X/a^+\}X$		$\lambda.X\{X/a\}$
$\parallel$		$\parallel$
$\lambda.\underbrace{a^+}_{\text{lift}}$	$\neq$	$\lambda.a$

$$\beta\text{-reduction}$$

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



- Introduced by G. Dowek, T. Hardin and C. Kirchner using the  $\lambda\sigma$ -calculus.
- Subsumes Huet's HOU method.

## 2. Unification in the $\lambda_{se}$ -style of explicit substitution

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

- A  $\lambda_{se}$ -unification problem  $P$  is: 
$$\left\{ \begin{array}{l} \forall_{j \in J} \underbrace{\exists \vec{w}_j \bigwedge_{i \in I_j} s_i =_{\lambda_{se}}^? t_i}_{\text{unification system}} \end{array} \right.$$

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



<b>Example :</b>	$(\lambda.(\lambda.(X \ 2) \ 1) \ Y)$	$=_{\lambda_{se}}^?$	$(\lambda.(Z \ 1) \ U)$
<i>Normalize</i>	$((X \sigma^2 Y) \sigma^1 (\varphi_0^1 Y) \ \varphi_0^1 Y)$	$\downarrow$	$X, Z : A \rightarrow A; Y, U : A$
<i>Dec-App</i>	$(X \sigma^2 Y) \sigma^1 (\varphi_0^1 Y) =_{\lambda_{se}}^? Z \sigma^1 U$	$=_{\lambda_{se}}^?$	$(Z \sigma^1 U \ \varphi_0^1 U)$
<i>Dec-<math>\varphi</math></i>	$(X \sigma^2 Y) \sigma^1 (\varphi_0^1 Y) =_{\lambda_{se}}^? Z \sigma^1 U$	$\downarrow$	$\varphi_0^1 Y =_{\lambda_{se}}^? \varphi_0^1 U$
<i>Replace</i>	$(X \sigma^2 Y) \sigma^1 (\varphi_0^1 Y) =_{\lambda_{se}}^? Z \sigma^1 Y$	$\downarrow$	$Y =_{\lambda_{se}}^? U$
<i>Exp-<math>\lambda</math> + Replace</i>	$((\lambda.X') \sigma^2 Y) \sigma^1 (\varphi_0^1 Y) =_{\lambda_{se}}^? (\lambda.Z') \sigma^1 Y$	$\downarrow^*$	$\wedge \left\{ \begin{array}{l} Y =_{\lambda_{se}}^? U \\ X =_{\lambda_{se}}^? \lambda.X' \\ Z =_{\lambda_{se}}^? \lambda.Z' \end{array} \right.$
<i>Normalize + Dec-<math>\lambda</math></i>	$(X' \sigma^3 Y) \sigma^2 (\varphi_0^1 Y) =_{\lambda_{se}}^? Z' \sigma^2 Y$	$\downarrow^*$	$\wedge \left\{ \begin{array}{l} Y =_{\lambda_{se}}^? U \\ X =_{\lambda_{se}}^? \lambda.X' \\ Z =_{\lambda_{se}}^? \lambda.Z' \end{array} \right.$

- *Solved* equations:  $\left. \begin{array}{l} Y =_{\lambda_{se}}^? U \\ X =_{\lambda_{se}}^? \lambda.X' \\ Z =_{\lambda_{se}}^? \lambda.Z' \end{array} \right\} \textit{Solved Forms}$
- *Flex-Flex* equations:  $(X'\sigma^3Y)\sigma^2(\varphi_0^1Y) =_{\lambda_{se}}^? Z'\sigma^2Y$

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

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

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

and

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

- 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  $\lambda s_e$ -unification rules are correct and complete.

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

$\lambda s_e$ -calculus and  $\lambda$ -calculus  $\rightarrow$   $\left. \begin{array}{l} \textit{Term} \\ \textit{Substitution} \end{array} \right\}$  objects  $\lambda\sigma$ -calculus

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

$\lambda\sigma$  uses only 1, “shift” and “composition”:  $n \equiv 1 \underbrace{[\uparrow \circ \dots \circ \uparrow]}_{n-1}$

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

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

$$\wedge \left\{ \begin{array}{l} P \\ X[a_1 \dots a_p. \uparrow^n] =_{\lambda\sigma}^? (m \ b_1 \dots b_q) \\ \bigvee_{r \in R_p \cup R_i} \exists H_1 \dots H_k, X =_{\lambda\sigma}^? (r \ H_1 \dots H_k) \end{array} \right.$$

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

*Exp-App*  $\lambda_{se}$ -unification rule

$$P \wedge \psi_{i_p}^{j_p} \dots \psi_{i_1}^{j_1}(X, a_1, \dots, a_p) =_{\lambda_{se}}^? (m \ b_1 \dots b_q) \rightarrow$$

$$\wedge \left\{ \begin{array}{l} P \\ \psi_{i_p}^{j_p} \dots \psi_{i_1}^{j_1}(X, a_1, \dots, a_p) =_{\lambda_{se}}^? (m \ b_1 \dots b_q) \\ \bigvee_{r \in R_p \cup R_i} \exists H_1, \dots, H_k, X =_{\lambda_{se}}^? (r \ H_1 \dots H_k) \end{array} \right.$$

$\psi_{i_p}^{j_p} \dots \psi_{i_1}^{j_1}(X, a_1, \dots, a_p)$  skeleton of a  $\lambda_{se}$ -normal term;  $X$  atomic and not solved;  $\Gamma_{H_i} = \Gamma_X$ ,  $R_p \subseteq \{i_1, \dots, i_p\}$  of superscripts of the  $\sigma$  operator such that  $(r \ H_1 \dots H_k)$  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

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

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

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

In the  $\lambda_{se}$ -calculus

$\psi_{k_p}^{j_p} \dots \psi_{k_1}^{j_1}(X, a_1, \dots, a_p) \stackrel{?}{=}_{\lambda_{se}} (m \ b_1 \dots b_q)$  solutions of the form:

$$\left( \mathbf{n} \quad \underbrace{H_1 \ \dots \ H_k}_{\text{of appropriate type}} \right)$$

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 $\lambda$ -calculus and the $\lambda_{s_e}$ -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  $\lambda$ -term in de Bruijn notation into the  $\lambda_{s_e}$ -calculus is defined by  $a_{pc} = PC(a, 0)$  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.  $PC(\mathbf{k}, n) = \mathbf{k}$
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]

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

**Proposition**[Correspondence between solutions]

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

$$\iff$$

$$\exists M_1, \dots, M_p \underbrace{a_{pc}\{X_1/M_1, \dots, X_p/M_p\}}_{\text{grafting}} =_{\lambda_{se}} \underbrace{b_{pc}\{X_1/M_1, \dots, X_p/M_p\}}_{\text{grafting}}$$

## 5. A simple example

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

$$\lambda.(\varphi_0^2(X) \ 2) =_{\lambda_{se}}^? \lambda.2$$

$$(\varphi_0^2(X) \ 2) =_{\lambda_{se}}^? 2$$

$$\exists Y (\varphi_0^2(X) \ 2) =_{\lambda_{se}}^? 2 \wedge X =_{\lambda_{se}}^? \lambda.Y$$

$$\exists Y (\varphi_0^2(\lambda.Y) \ 2) =_{\lambda_{se}}^? 2 \wedge X =_{\lambda_{se}}^? \lambda.Y$$

$$\exists Y (\varphi_1^2 Y) \sigma^1 2 =_{\lambda_{se}}^? 2 \wedge X =_{\lambda_{se}}^? \lambda.Y$$

$$(\exists Y (\varphi_1^2 Y) \sigma^1 2 =_{\lambda_{se}}^? 2 \wedge X =_{\lambda_{se}}^? \lambda.Y) \wedge (Y =_{\lambda_{se}}^? 1 \vee Y =_{\lambda_{se}}^? 2)$$

$$((\varphi_1^2 1) \sigma^1 2 =_{\lambda_{se}}^? 2 \wedge X =_{\lambda_{se}}^? \lambda.1) \vee ((\varphi_1^2 2) \sigma^1 2 =_{\lambda_{se}}^? 2 \wedge X =_{\lambda_{se}}^? \lambda.2)$$

$$(2 =_{\lambda_{se}}^? 2 \wedge X =_{\lambda_{se}}^? \lambda.1) \vee (2 =_{\lambda_{se}}^? 2 \wedge X =_{\lambda_{se}}^? \lambda.2)$$

$$(X =_{\lambda_{se}}^? \lambda.1) \vee (X =_{\lambda_{se}}^? \lambda.2)$$

$\rightarrow Dec-\lambda$

$\rightarrow Exp-\lambda$

$\rightarrow Replace$

$\rightarrow Normalize$

$\rightarrow Exp-app$

$\rightarrow Replace$

$\rightarrow Normalize$

$\equiv$

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

Solutions:  $\left\{ \begin{array}{l} \{X/\lambda.1\} \\ \{X/\lambda.2\} \end{array} \right.$

Note that we have:

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

and

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

## 6. Related work

Our development of the  $\lambda_{s_e}$ -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  $\lambda$ Prolog, Maude and ELAN.

Additional facts about the *back* transformation and practical considerations for an eventual implementation are available in Ayala-Rincón & Kamareddine “*On Applying  $\lambda_{s_e}$ -Style of Unification for Simply-Typed Higher Order Unification in the Pure  $\lambda$ -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  $\lambda_{s_e}$ -(HO)Unification strategies don't differ.
- Pre-cooking (and back) translations in  $\lambda\sigma$  and  $\lambda_{s_e}$  differ:
  - A simple selection of the scripts for the operators  $\varphi$  and  $\sigma$  in  $\lambda_{s_e}$  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.