Third-Order Matching via Explicit Substitutions

F. L. C. de Moura, M. Ayala-Rincón and F. Kamareddine

IV Seminário do GTC/UnB

07-08 de dezembro de 2006

Outline

Motivation

$\lambda\sigma$-Böhm Trees

From Matching to Interpolation Problems

Conclusions and Future Work

Outline

Motivation

$\lambda\sigma$-Böhm Trees

From Matching to Interpolation Problems

Conclusions and Future Work

Motivation

- Unification is a basic operation extensively used in Mathematics and Computer Science: it is of general use to describe computation as well as deduction.
- The unification problem can be state as follows:
  - Given two terms t and s, there exists a substitution σ such that $s\sigma = t\sigma$?
- Higher-Order Unification (HOU) is unification in the simply typed λ-calculus:
  - Given two λ-terms u and v of the same type, there exists a substitution γ such that $u\gamma =_{\beta\eta} v\gamma$?

Motivation

- Unification is a basic operation extensively used in Mathematics and Computer Science: it is of general use to describe computation as well as deduction.
- The unification problem can be state as follows:
  - Given two terms t and s, there exists a substitution σ such that $s\sigma = t\sigma$?
- Higher-Order Unification (HOU) is unification in the simply typed λ-calculus:
  - Given two λ-terms u and v of the same type, there exists a substitution γ such that $u\gamma =_{\beta\eta} v\gamma$?

Motivation

- Unification is a basic operation extensively used in Mathematics and Computer Science: it is of general use to describe computation as well as deduction.
- The unification problem can be state as follows:
  - Given two terms t and s, there exists a substitution σ such that $s\sigma = t\sigma$?
- Higher-Order Unification (HOU) is unification in the simply typed λ-calculus:
  - Given two λ-terms u and v of the same type, there exists a substitution γ such that $u\gamma =_{\beta\eta} v\gamma$?

Motivation

- Higher-Order Matching (HOM) is a particular case of HOU: It consists in determining whether a term is an instance of another term in the simply typed λ-calculus:
  - Given two λ-terms u and v of the same type, there exists a substitution γ such that $u\gamma =_{\beta\eta} v$?
- Decidability of HOM was conjectured for more than 30 years . . . and it was proved decidable recently (2006) by C. Stirling from University of Edinburgh!
- The proof of Stirling uses game-theoretic arguments.
- Applications of HOM include proof-checking, semi-automated theorem proving and program transformation among others.

Motivation

- Higher-Order Matching (HOM) is a particular case of HOU: It consists in determining whether a term is an instance of another term in the simply typed λ-calculus:
  - Given two λ-terms u and v of the same type, there exists a substitution γ such that $u\gamma =_{\beta\eta} v$?
- Decidability of HOM was conjectured for more than 30 years ... and it was proved decidable recently (2006) by C. Stirling from University of Edinburgh!
- The proof of Stirling uses game-theoretic arguments.
- Applications of HOM include proof-checking, semi-automated theorem proving and program transformation among others.

Motivation

- Higher-Order Matching (HOM) is a particular case of HOU: It consists in determining whether a term is an instance of another term in the simply typed λ-calculus:
  - Given two λ-terms u and v of the same type, there exists a substitution γ such that $u\gamma =_{\beta\eta} v$?
- Decidability of HOM was conjectured for more than 30 years ... and it was proved decidable recently (2006) by C. Stirling from University of Edinburgh!
- The proof of Stirling uses game-theoretic arguments.
- Applications of HOM include proof-checking, semi-automated theorem proving and program transformation among others.

Motivation

- Higher-Order Matching (HOM) is a particular case of HOU: It consists in determining whether a term is an instance of another term in the simply typed λ-calculus:
  - Given two λ-terms u and v of the same type, there exists a substitution γ such that $u\gamma =_{\beta\eta} v$?
- Decidability of HOM was conjectured for more than 30 years …and it was proved decidable recently (2006) by C. Stirling from University of Edinburgh!
- The proof of Stirling uses game-theoretic arguments.
- Applications of HOM include proof-checking, semi-automated theorem proving and program transformation among others.

Motivation

- Higher-Order Matching (HOM) is a particular case of HOU: It consists in determining whether a term is an instance of another term in the simply typed λ-calculus:
  - Given two λ-terms u and v of the same type, there exists a substitution γ such that $u\gamma =_{\beta\eta} v$?
- Decidability of HOM was conjectured for more than 30 years ... and it was proved decidable recently (2006) by C. Stirling from University of Edinburgh!
- The proof of Stirling uses game-theoretic arguments.
- Applications of HOM include proof-checking, semi-automated theorem proving and program transformation among others.

Motivation

- The adequate formalism to reason about systems based on the λ-calculus is known as *explicit substitutions*.
- In fact, the λ-calculus cannot be implemented directly because its substitution operation is a meta-operation:
  - $$(\lambda_x.M)\ N \to_\beta M[N/x]$$
- Explicit substitutions calculi extend the language of the λ-calculus with new operators that simulate the substitution operation.

Motivation

- The adequate formalism to reason about systems based on the λ-calculus is known as *explicit substitutions*.
- In fact, the λ-calculus cannot be implemented directly because its substitution operation is a meta-operation:
  - $$(\lambda_x.M)\ N \to_\beta M[N/x]$$
- Explicit substitutions calculi extend the language of the λ-calculus with new operators that simulate the substitution operation.

Motivation

- The adequate formalism to reason about systems based on the λ-calculus is known as *explicit substitutions*.
- In fact, the λ-calculus cannot be implemented directly because its substitution operation is a meta-operation:
  - $$(\lambda_x.M)\ N \rightarrow_\beta M[N/x]$$
- Explicit substitutions calculi extend the language of the λ-calculus with new operators that simulate the substitution operation.

The $\lambda\sigma$-calculus

The $\lambda\sigma$-calculus was developed by [ACCL91], and its grammar is given by:

Terms and Substitutions

$a ::= \ \underline{1} \mid \lambda.a \mid (a\ a) \mid a[s]$.
$s ::= \ id \mid \uparrow \mid a \cdot s \mid s \circ s$.

- Main properties of the typed $\lambda\sigma$-calculus:
  - Confluent;
  - Weak normalising.

The $\lambda\sigma$-calculus

The $\lambda\sigma$-calculus was developed by [ACCL91], and its grammar is given by:

Terms and Substitutions

$a ::= \ \underline{1} \mid \lambda.a \mid (a\ a) \mid a[s].$
$s ::= \ id \mid \uparrow \mid a \cdot s \mid s \circ s.$

- Main properties of the typed $\lambda\sigma$-calculus:
  - Confluent;
  - Weak normalising.

The $\lambda\sigma$-calculus

The $\lambda\sigma$-calculus was developed by [ACCL91], and its grammar is given by:

Terms and Substitutions

$a ::= \ \underline{1} \mid \lambda.a \mid (a\ a) \mid a[s].$
$s ::= \ id \mid \uparrow \mid a \cdot s \mid s \circ s.$

- Main properties of the typed $\lambda\sigma$-calculus:
  - Confluent;
  - Weak normalising.

Shortcuts:

- $\underline{n} = \underline{1}[\uparrow^{n-1}].$
- $\uparrow^n = \underbrace{\uparrow \circ (\uparrow \circ \ldots \circ \uparrow)}_{n \text{ times}} = \underline{n+1} \cdot \underline{n+2} \cdot \ldots$

The $\lambda\sigma$-calculus

(Beta)	$(\lambda_A.a)\ b$	$\longrightarrow$	$a[b \cdot id]$
(App)	$(a\ b)[s]$	$\longrightarrow$	$a[s]\ b[s]$
(Abs)	$(\lambda_A.a)[s]$	$\longrightarrow$	$\lambda_A.a[\underline{1} \cdot (s \circ \uparrow)]$
(Clos)	$(a[s])[t]$	$\longrightarrow$	$a[s \circ t]$
(VarCons)	$\underline{1}[a \cdot s]$	$\longrightarrow$	a
(Id)	$a[id]$	$\longrightarrow$	a
(Assoc)	$(s \circ t) \circ u$	$\longrightarrow$	$s \circ (t \circ u)$
(Map)	$(a \cdot s) \circ t$	$\longrightarrow$	$a[t] \cdot (s \circ t)$
(IdL)	$id \circ s$	$\longrightarrow$	s
(IdR)	$s \circ id$	$\longrightarrow$	s
(ShiftCons)	$\uparrow \circ (a \cdot s)$	$\longrightarrow$	s
(VarShift)	$\underline{1} \cdot \uparrow$	$\longrightarrow$	id
(SCons)	$\underline{1}[s] \cdot (\uparrow \circ s)$	$\longrightarrow$	s

Introduction

- In this work we show how Dowek's third-order matching procedure can be adapted to achieve decidability of the third-order matching problem in the language of the $\lambda\sigma$-calculus.

General Scheme

λ-calculus

Matching problem

$a \ll^{?} b$

$\downarrow$

Interpolation problem Φ

$(X\ a_1 \ldots a_p) \ll^{?} b'$

There exists a solution to Φ whose depth depends on the depth

of the Böhm tree of b.

$\lambda\sigma$-calculus

Matching problem

$a \ll^{?}_{\lambda\sigma} b$

$\downarrow$

Interpolation problem Φ

$(X[a_1 \cdot \ldots \cdot a_p \cdot \uparrow^n]\ b_1 \ldots b_q) \ll^{?}_{\lambda\sigma} b'$

There exists a solution to Φ whose depth depends on the depth

of the $\lambda\sigma$-Böhm tree of b.

Outline

Motivation

$\lambda\sigma$-Böhm Trees

From Matching to Interpolation Problems

Conclusions and Future Work

$\lambda\sigma$-Böhm Tree

Definition

A $\lambda\sigma$-Böhm tree is a tree whose nodes are labelled with pairs $\langle l, v\rangle$ such that l is a positive integer and v is a well typed $\lambda\sigma$-term.

Definition ($\lambda\sigma$-Böhm tree of a $\lambda\sigma$-nf)

Let $a = \lambda_{A_1} \ldots \lambda_{A_k}.(h\ b_1 \ldots b_m)$ be a term in $\lambda\sigma$-nf. The $\lambda\sigma$-Böhm tree of a is recursively defined as the tree whose root is labelled with the pair $\langle k, h\rangle$ and whose sons are the $\lambda\sigma$-Böhm trees of:

1. $b_1, \ldots, b_m$, if h is a de Bruijn index;
2. $a_1, \ldots, a_p, b_1, \ldots, b_m$, if h is a meta-variable of the form $X[a_1 \cdot \ldots \cdot a_p \cdot \uparrow^n]$, where $a_1 \cdot \ldots \cdot a_p \cdot \uparrow^n$ is a substitution in $\lambda\sigma$-nf.

$\lambda\sigma$-Böhm Tree

Definition

A $\lambda\sigma$-Böhm tree is a tree whose nodes are labelled with pairs $\langle l, v\rangle$ such that l is a positive integer and v is a well typed $\lambda\sigma$-term.

Definition ($\lambda\sigma$-Böhm tree of a $\lambda\sigma$-nf)

Let $a = \lambda_{A_1} \dots \lambda_{A_k}.(h\ b_1 \dots b_m)$ be a term in $\lambda\sigma$-nf. The $\lambda\sigma$-Böhm tree of a is recursively defined as the tree whose root is labelled with the pair $\langle k, h\rangle$ and whose sons are the $\lambda\sigma$-Böhm trees of:

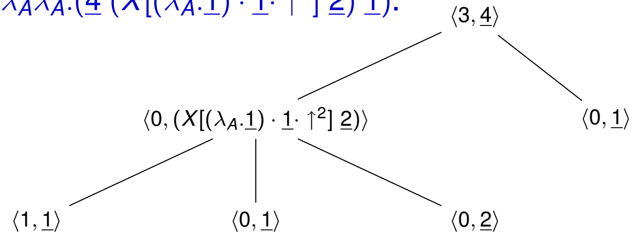
1. $b_1, \dots, b_m$, if h is a de Bruijn index;
2. $a_1, \dots, a_p, b_1, \dots, b_m$, if h is a meta-variable of the form $X[a_1 \cdot \dots \cdot a_p \cdot \uparrow^n]$, where $a_1 \cdot \dots \cdot a_p \cdot \uparrow^n$ is a substitution in $\lambda\sigma$-nf.

Example

- We write $|a|$ to denote the depth of the $\lambda\sigma$-Böhm tree of the $\lambda\sigma$-nf of the term a.

Example

$a = \lambda_A \lambda_A \lambda_A.(\underline{4}\ (X[(\lambda_A.\underline{1}) \cdot \underline{1} \cdot \uparrow^2]\ \underline{2})\ \underline{1})$.



Outline

Motivation

$\lambda\sigma$-Böhm Trees

From Matching to Interpolation Problems

Conclusions and Future Work

Interpolation Equation

Definition

An *interpolation equation* in the $\lambda\sigma$-calculus is an equation of the form $(X[a_1 \cdot \ldots \cdot a_p \cdot \uparrow^n]\ b_1 \ldots b_q) \ll^?_{\lambda\sigma} b$, where:

- X is a meta-variable;
- the terms $a_1, \ldots, a_p, b_1, \ldots, b_q, b$ are ground.

Interpolation Equation

Definition

Let $a \ll^?_{\lambda\sigma} b$ be a matching equation and σ be a ground solution to this equation, i.e., the $\lambda\sigma$-normal form of $a\sigma$ is $\beta\eta$-equivalent to b. We define the interpolation problem $\Phi(a \ll^?_{\lambda\sigma} b, \sigma)$ inductively over the number of occurrences of a as follows:

- If $a = \lambda_A.c$ then b is also an abstraction of the form $\lambda_A.d$ and then σ is also a solution to $c \ll^?_{\lambda\sigma} d$ and we let $\Phi(a \ll^?_{\lambda\sigma} b, \sigma) = \Phi(c \ll^?_{\lambda\sigma} d, \sigma)$.
- If $a = (\underline{k}\ c_1 \ldots c_m)$ then b is also of the form $(\underline{k}\ d_1 \ldots d_m)$ because $a \ll^?_{\lambda\sigma} b$ is solvable and we let $\Phi(a \ll^?_{\lambda\sigma} b, \sigma) = \bigcup_i \Phi(c_i \ll^?_{\lambda\sigma} d_i, \sigma)$.

Interpolation Equation

Definition (cont.)

- If $a = (X[a_1 \cdot \ldots \cdot a_p \cdot \uparrow^n] \; b_1 \ldots b_q)$ then we let

$$\Phi(a \ll^?_{\lambda\sigma} b, \sigma) = \{(X[a_1 \cdot (\diamond) \cdot a_p \cdot \uparrow^n] \; b_1 \; (\diamond) \; b_q) \ll^?_{\lambda\sigma} b\} \bigcup_i H_i$$

$$H_i \begin{cases} \Phi(a_i \ll^?_{\lambda\sigma} a_i\sigma, \sigma) & \text{if } a_i \text{ is a flexible term and} \\ & \diamond \text{ occurs in the } \lambda\sigma\text{-normal form of} \\ & (X[a_1 \cdot \ldots \cdot a_{i-1} \cdot \diamond \cdot a_{i+1} \cdot \ldots \cdot a_p \cdot \uparrow^n] \; b_1 \cdot \ldots \cdot b_q)\sigma; \\ \Phi(b_i \ll^?_{\lambda\sigma} b_i\sigma, \sigma) & \text{if } b_i \text{ is a flexible term and} \\ & \diamond \text{ occurs in the } \lambda\sigma\text{-normal form of} \\ & (X[a_1 \cdot \ldots \cdot a_p \cdot \uparrow^n] \; b_1 \ldots b_{i-1} \diamond b_{i+1} \ldots b_q)\sigma; \\ \emptyset & \text{otherwise.} \end{cases}$$

Example

Example

Let

- A be an atomic type and $\Gamma = A \to A \cdot A \cdot nil$;
- matching equation: $X[\lambda_A.(\underline{2}\ \underline{1}) \cdot Y \cdot \uparrow] \ll^?_{\lambda\sigma} (\underline{1}\ \underline{2})$.
- solutions: $\sigma_1 = \{X \mapsto (\underline{1}\ \underline{3})\}$, $\sigma_2 = \{X \mapsto \underline{2}, Y \mapsto (\underline{1}\ \underline{2})\}$

Interpolation equation associated to:

$\Rightarrow$ $\sigma_1 : X[\lambda_A.(\underline{2}\ \underline{1}) \cdot \diamond \cdot \uparrow] \ll^?_{\lambda\sigma} (\underline{1}\ \underline{2})$.

$\Rightarrow$ $\sigma_2 : X[\lambda_A.(\underline{2}\ \underline{1}) \cdot (\underline{1}\ \underline{2}) \cdot \uparrow] \ll^?_{\lambda\sigma} (\underline{1}\ \underline{2}) \wedge Y \ll^?_{\lambda\sigma} (\underline{1}\ \underline{2})$.

General Idea

Let $(X[a_1 \cdot \ldots \cdot a_p \cdot \uparrow^n]\ b_1 \ldots b_q) \ll^?_{\lambda\sigma} b$ be an interpolation equation and σ is a solution to this equation with $X\sigma = \lambda_{B_1} \ldots \lambda_{B_q}.t$ then we have that

$$|t[b_q \cdot \ldots \cdot b_1 \cdot a_1 \cdot \ldots \cdot a_p \cdot \uparrow^n]| = |b|.$$

If it is always the case that $|t| \leq |t[b_q \cdot \ldots \cdot b_1 \cdot a_1 \cdot \ldots \cdot a_p \cdot \uparrow^n]|$, i.e.,

$$|t| \leq |b| \tag{1}$$

then an enumeration of the terms t satisfying (1) would give a decision procedure for third-order matching. Unfortunately, this is not always the case: to solve this problem we show that if a matching problem is solvable then there is a solution which is limited by a number that only depends on the initial problem.

General Idea

Let $(X[a_1 \cdot \ldots \cdot a_p \cdot \uparrow^n]\ b_1 \ldots b_q) \ll^?_{\lambda\sigma} b$ be an interpolation equation and σ is a solution to this equation with $X\sigma = \lambda_{B_1} \ldots \lambda_{B_q}.t$ then we have that

$$|t[b_q \cdot \ldots \cdot b_1 \cdot a_1 \cdot \ldots \cdot a_p \cdot \uparrow^n]| = |b|.$$

If it is always the case that $|t| \leq |t[b_q \cdot \ldots \cdot b_1 \cdot a_1 \cdot \ldots \cdot a_p \cdot \uparrow^n]|$, i.e.,

$$|t| \leq |b| \qquad (1)$$

then an enumeration of the terms t satisfying (1) would give a decision procedure for third-order matching. Unfortunately, this is not always the case: to solve this problem we show that if a matching problem is solvable then there is a solution which is limited by a number that only depends on the initial problem.

General Idea

Let $(X[a_1 \cdot \ldots \cdot a_p \cdot \uparrow^n]\ b_1 \ldots b_q) \ll^?_{\lambda\sigma} b$ be an interpolation equation and σ is a solution to this equation with $X\sigma = \lambda_{B_1} \ldots \lambda_{B_q}.t$ then we have that

$$|t[b_q \cdot \ldots \cdot b_1 \cdot a_1 \cdot \ldots \cdot a_p \cdot \uparrow^n]| = |b|.$$

If it is always the case that $|t| \leq |t[b_q \cdot \ldots \cdot b_1 \cdot a_1 \cdot \ldots \cdot a_p \cdot \uparrow^n]|$, i.e.,

$$|t| \leq |b| \qquad (1)$$

then an enumeration of the terms t satisfying (1) would give a decision procedure for third-order matching. Unfortunately, this is not always the case: to solve this problem we show that if a matching problem is solvable then there is a solution which is limited by a number that only depends on the initial problem.

General Idea

Let $(X[a_1 \cdot \ldots \cdot a_p \cdot \uparrow^n]\ b_1 \ldots b_q) \ll^?_{\lambda\sigma} b$ be an interpolation equation and σ is a solution to this equation with $X\sigma = \lambda_{B_1} \ldots \lambda_{B_q}.t$ then we have that

$$|t[b_q \cdot \ldots \cdot b_1 \cdot a_1 \cdot \ldots \cdot a_p \cdot \uparrow^n]| = |b|.$$

If it is always the case that $|t| \leq |t[b_q \cdot \ldots \cdot b_1 \cdot a_1 \cdot \ldots \cdot a_p \cdot \uparrow^n]|$, i.e.,

$$|t| \leq |b| \tag{1}$$

then an enumeration of the terms t satisfying (1) would give a decision procedure for third-order matching. Unfortunately, this is not always the case: to solve this problem we show that if a matching problem is solvable then there is a solution which is limited by a number that only depends on the initial problem.

Interpolation Equation

Theorem

Let a be a $\lambda\sigma$-nf, $a_1, \ldots, a_p$ be $\lambda\sigma$-normal terms of at most second-order, $n \geq 0$ and $a[a_1 \cdot \ldots \cdot a_p \cdot \uparrow^n]$ be a well typed term. If for all $1 \leq i \leq p$, the term a_i is relevant in all its arguments and $|a_i| \neq 0$ then

$$|a| \leq |a[a_1 \cdot \ldots \cdot a_p \cdot \uparrow^n]|.$$

Depth of the solutions

Example

Let

- A be an atomic type and $\Gamma = A \rightarrow A \cdot A \cdot nil$;
- $A \rightarrow A \rightarrow A \cdot A \cdot nil \vdash X : A$

$\Rightarrow$ Interpolation equation well typed in Γ:

$$X[\lambda_A \lambda_A.(\underline{3}\ \underline{2}) \cdot \uparrow] \ll^{?}_{\lambda\sigma} (\underline{1}\ \underline{2})$$

Solutions:

- $\sigma_1 = \{X \mapsto (\underline{1}\ \underline{2}\ \underline{2})\}$
- $\sigma_2 = \{X \mapsto (\underline{1}\ \underline{2}\ (\underline{1}\ \underline{2}\ \underline{2}))\}$
- $\sigma_3 = \{X \mapsto (\underline{1}\ \underline{2}\ (\underline{1}\ \underline{2}\ (\underline{1}\ \underline{2}\ \underline{2})))\}$
- $\sigma_4 = \{X \mapsto (\underline{1}\ \underline{2}\ (\underline{1}\ \underline{2}\ (\underline{1}\ \underline{2}\ (\underline{1}\ \underline{2}\ \underline{2}))))\} \ldots$

Occurrence accessible w.r.t. an equation

Definition

Consider a matching equation of the form $(X[a_1 \cdot \ldots \cdot a_p \cdot \uparrow^n] \; b_1 \ldots b_q) \ll^?_{\lambda\sigma} b$ and the term $t = \lambda_{C_1} \ldots \lambda_{C_q}.u$ with the same type and context of X. The set of occurrences in the $\lambda\sigma$-Böhm tree of t *accessible* with respect to the equation

$$(X[a_1 \cdot \ldots \cdot a_p \cdot \uparrow^n] \; b_1 \ldots b_q) \ll^?_{\lambda\sigma} b$$

is inductively defined as:

- the root of the $\lambda\sigma$-Böhm tree of t is accessible.

Occurrence accessible w.r.t. an equation

Definition (cont.)

- if α is an accessible occurrence labelled with a de Bruijn index $\underline{j}$ with $1 \leq j \leq p + q$ and d_j is relevant in its r-th argument then the occurrence $\alpha\langle r\rangle$ is accessible, where:
$$d_j = \begin{cases} a_j & \text{if } q < j \leq p + q; \\ b_{q-i+1} & \text{if } 1 \leq j \leq q. \end{cases}$$
- if α is an accessible occurrence labelled with a de Bruijn index greater than $p + q$ or with a meta-variable then all the sons of α are accessible.

Accessible solution built from a solution

Definition

Φ be an interpolation problem and let σ be a ground solution to this problem. For each meta-variable X occurring in the equations of Φ consider the $\lambda\sigma$-term t such that $\{X \mapsto t\} \subseteq \sigma$. In the $\lambda\sigma$-Böhm tree of t, we prune all occurrences non accessible (that are not leaves) with respect to the equations of Φ in which X has an occurrence and put $\lambda\sigma$-Böhm trees of ground terms of depth 0 of the expected type as leaves. Call t' the term whose $\lambda\sigma$-Böhm tree is obtained in this way and $\widehat{\sigma}$ the resulting substitution.

Example

Example

Consider again the example where $\Gamma = A \to A \cdot A \cdot nil$, $A \to A \to A \cdot A \cdot nil \vdash X : A$ and

$$X[\lambda_A \lambda_A.(\underline{3}\ \underline{2}) \cdot \uparrow] \ll^?_{\lambda\sigma} (\underline{1}\ \underline{2}) \tag{2}$$

The grafting $\sigma = \{X \mapsto (\underline{1}\ \underline{2}\ a)\}$ is a solution to this equation where a is any $\lambda\sigma$-term of type A that is well typed in context $A \cdot A \cdot \Gamma$. In fact, the occurrence a in the term $(\underline{1}\ \underline{2}\ a)$ is not accessible w.r.t. equation (2).

Accessible solution: $\widehat{\sigma} = \{X \mapsto (\underline{1}\ \underline{2}\ \diamond)\}$

Depth of the solutions

Example

Let

- A be an atomic type and $\Gamma = A \cdot A \cdot nil$;
- $\Gamma \vdash X : (A \to A) \to A$

$\Rightarrow$ Third-order matching problem well typed in Γ:

$$X(\lambda_A.\underline{1}) \ll^?_{\lambda\sigma} \underline{2}$$

Solutions:

- $\sigma_1 = \{X \mapsto \lambda_{A\to A}.\underline{3}\}$
- $\sigma_2 = \{X \mapsto \lambda_{A\to A}.(\underline{1}\ \underline{3})\}$
- $\sigma_3 = \{X \mapsto \lambda_{A\to A}.(\underline{1}(\underline{1}\ \underline{3}))\} \ldots$

Compact Solution

Theorem

Let

- $X[a_1 \cdot \ldots \cdot a_p \cdot \uparrow^n] \ll^?_{\lambda\sigma} b$ *be an interpolation equation;*
- $\widehat{\sigma} = t$ *be an accessible solution to this equation.*

If α *is an occurrence in the* $\lambda\sigma$*-Böhm tree of* t *that contains more than* $|b| + 1$ *free occurrences of the de Bruijn index* $\underline{j}$ $(1 \leq j \leq p)$ *in its path, then the* $(|b| + 2)$*-th occurrence labelled with* $\underline{j}$ *is accessible w.r.t. this equation, the term* a_j *is a projection, i.e., there exists an integer* $1 \leq r \leq p$ *such that* $a_j = \lambda_{B_1} \ldots \lambda_{B_p}.\underline{r}$.

Compact Accessible Solution

Definition

Let

- Φ: interpolation problem;
- $\widehat{\sigma}$ be an accessible solution;
- *h* be the maximum depth in the $\lambda\sigma$-Böhm tree of the right-hand side of the equations of Φ.

The grafting $\widehat{\sigma}$ is a *compact accessible solution built from an accessible solution* to Φ if, for all meta-variable X occurring in Φ, the term $t = \widehat{\sigma}X$ is such that there is no path in the $\lambda\sigma$-Böhm tree of t containing more than $h + 1$ occurrences labelled with the de Bruijn index $\underline{j}$ $(1 \leq j \leq p)$.

Remark

Remark

If there exists a path in the $\lambda\sigma$-Böhm tree of t that has more than $h+1$ free occurrences of the de Bruijn index $\underline{j}$ $(1 \leq j \leq p)$ then the compact accessible solution is built as follows: if these occurrences are accessible w.r.t. the equation $X[a_1 \cdot \ldots \cdot a_p \cdot \uparrow^n]$, we have that a_j is a projection of the form $\lambda_{B_1} \ldots \lambda_{B_p}.\underline{r}$. In this case, we replace all these occurrences of $\underline{j}$ by $\lambda_{B_1} \ldots \lambda_{B_p}.\underline{r}$. The compact accessible solution built from the accessible solution σ is denoted by σ'.

Compact accessible solution

Theorem

Let Φ be a third-order interpolation problem, σ be a solution to Φ, $\widehat{\sigma}$ be the accessible solution built from σ and σ' be the compact accessible solution built from $\widehat{\sigma}$. If h is the maximum depth in the $\lambda\sigma$-Böhm tree of the right-hand side of the equations of Φ, then for every meta-variable X of arity n, the depth of the $\lambda\sigma$-Böhm tree of $\sigma' X$ is less than or equal to $(n+1)(h+1)-1$.

Decision procedure

Theorem

The class of third-order matching problems in the $\lambda\sigma$-calculus is decidable.

Proof.

Let Ψ be a third-order matching problem in the $\lambda\sigma$-calculus. Enumerate all ground substitutions for the meta-variables occurring in the equations of the form $(X[a_1 \cdot \ldots \cdot a_p \cdot \uparrow^n]\ b_1 \ldots b_q) \ll^?_{\lambda\sigma} b$ of Ψ, such that the terms to be substituted for X have depth less than or equal to $(q+1)(h+1)-1$, where h is the depth of the $\lambda\sigma$-Böhm tree of b. If none of these substitutions is a solution Φ then Φ is not solvable. Otherwise, it is solvable. $\square$

Outline

Motivation

$\lambda\sigma$-Böhm Trees

From Matching to Interpolation Problems

Conclusions and Future Work

Conclusions and Future Work

In this work:

- We adapted the Dowek's decision procedure to third-order matching in the $\lambda\sigma$-calculus to prove decidability of third-order matching in the $\lambda\sigma$-calculus;
- We introduced the notion of $\lambda\sigma$-Böhm tree that is essential for establishing the decision procedure.
- We introduced the notion of interpolation problem for the $\lambda\sigma$-language.

Conclusions and Future Work

In this work:

- We adapted the Dowek's decision procedure to third-order matching in the $\lambda\sigma$-calculus to prove decidability of third-order matching in the $\lambda\sigma$-calculus;
- We introduced the notion of $\lambda\sigma$-Böhm tree that is essential for establishing the decision procedure.
- We introduced the notion of interpolation problem for the $\lambda\sigma$-language.

Conclusions and Future Work

In this work:

- We adapted the Dowek's decision procedure to third-order matching in the $\lambda\sigma$-calculus to prove decidability of third-order matching in the $\lambda\sigma$-calculus;
- We introduced the notion of $\lambda\sigma$-Böhm tree that is essential for establishing the decision procedure.
- We introduced the notion of interpolation problem for the $\lambda\sigma$-language.

Conclusions and Future Work

Future work:

- $\Rightarrow$ Investigate if explicit substitutions can give some insights on how to get better bounds for the depth of the $\lambda\sigma$-Böhm trees including higher-order cases.
- $\Rightarrow$ Implementation of this decision procedure to compare efficiency with other implementations.

Conclusions and Future Work

Future work:

- $\Rightarrow$ Investigate if explicit substitutions can give some insights on how to get better bounds for the depth of the $\lambda\sigma$-Böhm trees including higher-order cases.
- $\Rightarrow$ Implementation of this decision procedure to compare efficiency with other implementations.

Main References

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

M. Ayala-Rincón, F.L.C. de Moura, and F. Kamareddine.
Comparing and implementing calculi of explicit substitutions with eta-reduction.
Annals of Pure and Applied Logic, 134:5–41, 2005.

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

G. Dowek.
Third order matching is decidable.
Annals of Pure and Applied Logic, 69:135–155, 1994.

F.L.C. de Moura, F. Kamareddine, and M. Ayala-Rincón.
Second order matching via explicit substitutions.
In F. Baader and A. Voronkov, editors, *11th International Conference on Logic for Programming Artificial Intelligence and Reasoning (LPAR'04)*, volume 3452 of *LNAI*, pages 433–448. Springer-Verlag, 2005.

Definition (Free occurrence)

A free occurrence of a de Bruijn index $\underline{i}$ in the $\lambda\sigma$-term a is defined by:

1. If $a = X[s]$ then $\underline{i}$ does not occur free in a, where X is a meta-variable and s is any substitution.
2. If $a = \underline{i}$ then $\underline{i}$ occurs free in a.
3. If $a = \lambda_A.b$ and $\underline{i}$ occurs free in b then $\underline{i+1}$ occurs free in a.
4. If $a = (b\ c)$ and $\underline{i}$ occurs free in b or c (or in both) then $\underline{i}$ occurs free in a.

Definition (Free occurrence)

A free occurrence of a de Bruijn index $\underline{i}$ in the $\lambda\sigma$-term a is defined by:

1. If $a = X[s]$ then $\underline{i}$ does not occur free in a, where X is a meta-variable and s is any substitution.
2. If $a = \underline{i}$ then $\underline{i}$ occurs free in a.
3. If $a = \lambda_A.b$ and $\underline{i}$ occurs free in b then $\underline{i+1}$ occurs free in a.
4. If $a = (b\ c)$ and $\underline{i}$ occurs free in b or c (or in both) then $\underline{i}$ occurs free in a.

Definition (Relevant term)

A $\lambda\sigma$-term $a = \lambda_{A_1} \ldots \lambda_{A_k}.b$ is *relevant in its i-th* $(1 \leq i \leq k)$ *argument* if the index $\underline{k - i + 1}$ occurs free in b.