Quiz

Assuming that a and b are distinct variables, is it possible to find λ-terms M_1 to M_7 that make the following pairs α-equivalent?

- $\lambda a.\lambda b.(M_1\, b)$ and $\lambda b.\lambda a.(a\, M_1)$
- $\lambda a.\lambda b.(M_2\, b)$ and $\lambda b.\lambda a.(a\, M_3)$
- $\lambda a.\lambda b.(b\, M_4)$ and $\lambda b.\lambda a.(a\, M_5)$
- $\lambda a.\lambda b.(b\, M_6)$ and $\lambda a.\lambda a.(a\, M_7)$

If there is one solution for a pair, can you describe all its solutions?

Nominal Unification
Hitting a Sweet Spot

Christian Urban

initial spark from Roy Dyckhoff in November 2001
joint work with Andy Pitts and Jamie Gabbay

One Motivation

One Motivation

Typing implemented in Prolog (from a textbook)

One Motivation

Typing implemented in Prolog (from a textbook)

```
type (Gamma, var(X), T) :- member (X,T) Gamma.

type (Gamma, app(M, N), T') :-
            type (Gamma, M, arrow(T, T')),
            type (Gamma, N, T).

type (Gamma, lam(X, M), arrow(T, T')) :-
            type ((X, T)::Gamma, M, T').

member X X::Tail.
member X Y::Tail :- member X Tail.
```

One Motivation

The problem is that $\lambda x.\lambda x.(x\ x)$ will have the types

$$T \to (T \to S) \to S \text{ and}$$
$$(T \to S) \to T \to S$$

Typi… …ook)

type… …amma.

type…

type (Gamma, N, T).

type (Gamma, lam(X, M), arrow(T, T')) :-
type ((X, T)::Gamma, M, T').

member X X::Tail.
member X Y::Tail :- member X Tail.

Higher-Order Unification

State of the art at the time:

- Lambda Prolog with full Higher-Order Unification
  (no mgus, undecidable, modulo $\alpha\beta$)

- Higher-Order Pattern Unification
  (has mgus, decidable, some restrictions, modulo $\alpha\beta_0$)

Underlying Ideas

- Unification (only) up to α

Underlying Ideas

- Unification (only) up to α
- Swappings / Permutations

$$\lambda a.b \qquad\qquad \lambda c.b$$

Underlying Ideas

- Unification (only) up to α
- Swappings / Permutations

$$[b := a]\ \lambda a.b = \lambda a.a \qquad [b := a]\ \lambda c.b = \lambda c.a$$

Underlying Ideas

- Unification (only) up to α
- Swappings / Permutations

$$(a\,b)\bullet\lambda a.b = \lambda b.a \qquad\qquad (a\,b)\bullet\lambda c.b = \lambda c.a$$

$(a\,b)\bullet t \stackrel{\text{def}}{=}$ swap **all** occurrences of b and a in t

Underlying Ideas

- Unification (only) up to α
- Swappings / Permutations

$$(a\,b) \cdot \lambda a.b = \lambda b.a \qquad (a\,b) \cdot \lambda c.b = \lambda c.a$$

$(a\,b) \bullet t \stackrel{\text{def}}{=}$ swap **all** occurrences of b and a in t

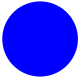
Unlike for $[b := a] \bullet (-)$, for $(a\,b) \bullet (-)$ we do have if $t =_\alpha t'$ then $\pi \bullet t =_\alpha \pi \bullet t'$.

Underlying Ideas

- Unification (only) up to α
- Swappings / Permutations
- Variables (or holes)

Underlying Ideas

- Unification (only) up to α
- Swappings / Permutations
- Variables (or holes)

$\lambda xs.$ 

Underlying Ideas

- Unification (only) up to α
- Swappings / Permutations
- Variables (or holes)

$$\lambda xs.\left(\bullet\ ys\right)$$

ys are the parameters the hole can depend on

Underlying Ideas

- Unification (only) up to α
- Swappings / Permutations
- Variables (or holes)

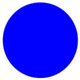
$$\lambda xs. \left(\bullet \; ys \right)$$

ys are the parameters the hole can depend on, but then you need β_0-reduction

$$(\lambda x.t)y \longrightarrow_{\beta_0} t[x := y]$$

Underlying Ideas

- Unification (only) up to α
- Swappings / Permutations
- Variables (or holes)

$\lambda xs.$ 

we will record the information about which parameters a hole **cannot** depend on

Terms

- $\langle\rangle$ Units
- $\langle t, t' \rangle$ Pairs
- $F\, t$ Funct.

Terms

- $\langle\rangle$ Units
- $\langle t, t' \rangle$ Pairs
- $F\,t$ Funct.
- a Atoms

Atoms are constants (infinitely many of them)

Terms

- $\langle\rangle$ Units
- $\langle t, t' \rangle$ Pairs
- $F\, t$ Funct.
- a Atoms
- $a.t$ Abstractions

$\ulcorner \lambda a.a \urcorner \mapsto \text{fn}\ a.a$
constructions like $\text{fn}\ X.X$ are not allowed

Terms

- $\langle\rangle$ Units
- $\langle t, t'\rangle$ Pairs
- $F\, t$ Funct.
- a Atoms
- $a.t$ Abstractions
- $\pi{\cdot}X$ Suspensions

X is a variable standing for a term
π is an explicit permutation $(a_1\, b_1)\ldots(a_n\, b_n)$, waiting to be applied to the term that is substituted for X

Permutations

a permutation applied to a term

- $$[]\bullet c \stackrel{\text{def}}{=} c$$
- $$(a\,b)::\pi\bullet c \stackrel{\text{def}}{=} \begin{cases} a & \text{if } \pi\bullet c = b \\ b & \text{if } \pi\bullet c = a \\ \pi\bullet c & \text{otherwise} \end{cases}$$

Permutations

a permutation applied to a term

- $$[] \bullet c \stackrel{\text{def}}{=} c$$
- $$(a\,b) :: \pi \bullet c \stackrel{\text{def}}{=} \begin{cases} a & \text{if } \pi \bullet c = b \\ b & \text{if } \pi \bullet c = a \\ \pi \bullet c & \text{otherwise} \end{cases}$$
- $$\pi \bullet a.t \stackrel{\text{def}}{=} \pi \bullet a.\pi \bullet t$$

Permutations

a permutation applied to a term

- $[]\bullet c \stackrel{\text{def}}{=} c$
- $(a\,b)::\pi\bullet c \stackrel{\text{def}}{=} \begin{cases} a & \text{if } \pi\bullet c = b \\ b & \text{if } \pi\bullet c = a \\ \pi\bullet c & \text{otherwise} \end{cases}$
- $\pi\bullet a.t \stackrel{\text{def}}{=} \pi\bullet a.\pi\bullet t$
- $\pi\bullet\pi'\bullet X \stackrel{\text{def}}{=} (\pi @ \pi')\bullet X$

Freshness Constraints

Recall $\lambda a.\bullet$

Freshness Constraints

Recall $\lambda a.$●

We therefore will identify

$$\mathrm{fn}\; a.X \;\approx\; \mathrm{fn}\; b.(a\,b)\bullet X$$

provided that 'b is fresh for X — (b # X)', i.e., does not occur freely in any ground term that might be substituted for X.

Freshness Constraints

Recall $\lambda a.$●

We therefore will identify

$$\mathsf{fn}\ a.X \;\approx\; \mathsf{fn}\ b.(a\ b)\bullet X$$

provided that 'b is fresh for X — ($b \# X$)', i.e., does not occur freely in any ground term that might be substituted for X.

If we know more about X, e.g., if we knew that $a \# X$ and $b \# X$, then we can replace $(a\ b)\bullet X$ by X.

Equivalence Judgements

Our equality is **not** just

$$t \approx t' \quad \text{α-equivalence}$$

Equivalence Judgements

but judgements

$$\nabla \vdash t \approx t' \quad \alpha\text{-equivalence}$$

where

$$\nabla = \{a_1 \# X_1, \ldots, a_n \# X_n\}$$

is a finite set of freshness assumptions.

Equivalence Judgements

but judgements

$$\nabla \vdash t \approx t' \quad \alpha\text{-equivalence}$$

where

$$\nabla = \{a_1 \# X_1, \ldots, a_n \# X_n\}$$

is a finite set of freshness assumptions.

$$\{a \# X, b \# X\} \vdash \text{fn}\, a.X \approx \text{fn}\, b.X$$

Equivalence Judgements

but judgements

$$\nabla \vdash t \approx t' \quad \alpha\text{-equivalence}$$

$$\nabla \vdash a \# t \quad \text{freshness}$$

where

$$\nabla = \{a_1 \# X_1, \ldots, a_n \# X_n\}$$

is a finite set of freshness assumptions.

$$\{a \# X, b \# X\} \vdash \mathrm{fn}\, a.X \approx \mathrm{fn}\, b.X$$

Rules for Equivalence

Excerpt
(i.e. only the interesting rules)

Rules for Equivalence

$$\frac{}{\nabla \vdash a \approx a}$$

$$\frac{\nabla \vdash t \approx t'}{\nabla \vdash a.t \approx a.t'}$$

$$\frac{a \neq b \qquad \nabla \vdash t \approx (a\,b){\bullet}t' \qquad \nabla \vdash a \# t'}{\nabla \vdash a.t \approx b.t'}$$

Rules for Equivalence

$$\frac{(a \mathrel{\#} X) \in \nabla \quad \text{for all } a \text{ with } \pi \cdot a \neq \pi' \cdot a}{\nabla \vdash \pi \cdot X \approx \pi' \cdot X}$$

Rules for Equivalence

$$\frac{\begin{array}{c}(a \;\#\; X) \in \nabla \\ \text{for all } a \text{ with } \pi\bullet a \neq \pi'\bullet a\end{array}}{\nabla \vdash \pi\bullet X \approx \pi'\bullet X}$$

for example

$$\{a \;\#\; X, b \;\#\; X\} \vdash X \approx (a\,b)\bullet X$$

Rules for Equivalence

$$\frac{(a \mathrel{\#} X) \in \nabla \quad \text{for all } a \text{ with } \pi \bullet a \neq \pi' \bullet a}{\nabla \vdash \pi \bullet X \approx \pi' \bullet X}$$

for example

$$\{a \# X, c \# X\} \vdash (a\,c)(a\,b) \bullet X \approx (b\,c) \bullet X$$

because $(a\,c)(a\,b)$: $a \mapsto b$, $b \mapsto c$, $c \mapsto a$ $\quad (b\,c)$: $a \mapsto a$, $b \mapsto c$, $c \mapsto b$

disagree at a and c.

Rules for Freshness

Excerpt
(i.e. only the interesting rules)

Rules for Freshness

$$\frac{a \neq b}{\nabla \vdash a \# b}$$

$$\frac{}{\nabla \vdash a \# a.t} \qquad \frac{a \neq b \qquad \nabla \vdash a \# t}{\nabla \vdash a \# b.t}$$

$$\frac{(\pi^{-1} \bullet a \# X) \in \nabla}{\nabla \vdash a \# \pi \bullet X}$$

$\approx$ is an Equivalence

Theorem: $\approx$ is an equivalence relation.

(Reflexivity)	$\nabla \vdash t \approx t$
(Symmetry)	if $\nabla \vdash t_1 \approx t_2$ then $\nabla \vdash t_2 \approx t_1$
(Transitivity)	if $\nabla \vdash t_1 \approx t_2$ and $\nabla \vdash t_2 \approx t_3$ then $\nabla \vdash t_1 \approx t_3$

$\approx$ is an Equivalence

Theorem: $\approx$ is an equivalence relation.

- $\nabla \vdash t \approx t'$ then $\nabla \vdash \pi{\bullet}t \approx \pi{\bullet}t'$
- $\nabla \vdash a \# t$ then $\nabla \vdash \pi{\bullet}a \# \pi{\bullet}t$

$\approx$ is an Equivalence

Theorem: $\approx$ is an equivalence relation.

- $\nabla \vdash t \approx t'$ then $\nabla \vdash \pi{\bullet}t \approx \pi{\bullet}t'$
- $\nabla \vdash a \# t$ then $\nabla \vdash \pi{\bullet}a \# \pi{\bullet}t$
- $\nabla \vdash t \approx \pi{\bullet}t'$ then $\nabla \vdash (\pi^{-1}){\bullet}t \approx t'$
- $\nabla \vdash a \# \pi{\bullet}t$ then $\nabla \vdash (\pi^{-1}){\bullet}a \# t$

$\approx$ is an Equivalence

Theorem: $\approx$ is an equivalence relation.

- $\nabla \vdash t \approx t'$ then $\nabla \vdash \pi{\bullet}t \approx \pi{\bullet}t'$
- $\nabla \vdash a \# t$ then $\nabla \vdash \pi{\bullet}a \# \pi{\bullet}t$
- $\nabla \vdash t \approx \pi{\bullet}t'$ then $\nabla \vdash (\pi^{-1}){\bullet}t \approx t'$
- $\nabla \vdash a \# \pi{\bullet}t$ then $\nabla \vdash (\pi^{-1}){\bullet}a \# t$
- $\nabla \vdash a \# t$ and $\nabla \vdash t \approx t'$ then $\nabla \vdash a \# t'$

Comparison $=_\alpha$

Traditionally $=_\alpha$ is defined as

> least congruence which identifies $a.t$ with $b.[a := b]t$ provided b is not free in t

where $[a := b]t$ replaces all free occurrences of a by b in t.

Comparison $=_\alpha$

Traditionally $=_\alpha$ is defined as

> least congruence which identifies $a.t$ with $b.[a := b]t$ provided b is not free in t

where $[a := b]t$ replaces all free occurrences of a by b in t.

For ground terms:

> <u>Theorem:</u> $t =_\alpha t'$ iff $\varnothing \vdash t \approx t'$
>
> $a \notin FA(t)$ iff $\varnothing \vdash a \# t$

Comparison $=_\alpha$

Traditionally $=_\alpha$ is defined as

> least congruence which identifies $a.t$ with $b.[a := b]t$ provided b is not free in t

where $[a := b]t$ replaces all free occurrences of a by b in t.

In general $=_\alpha$ and $\approx$ are distinct!

> $a.X =_\alpha b.X$ but not
>
> $\varnothing \vdash a.X \approx b.X \quad (a \neq b)$

Comparison $=_\alpha$

That is a crucial point: if we had

$$\varnothing \vdash a.X \approx b.X\,,$$

then applying $[X := a]$, $[X := b]$, ... give two terms that are **not** α-equivalent.

The freshness constraints $a \# X$ and $b \# X$ rule out the problematic substitutions. Therefore

$$\{a \# X, b \# X\} \vdash a.X \approx b.X$$

does hold.

Substitution

- $$\sigma(a.t) \stackrel{\text{def}}{=} a.\sigma(t)$$
- $$\sigma(\pi \bullet X) \stackrel{\text{def}}{=} \begin{cases} \pi \bullet \sigma(X) & \text{if } \sigma(X) \neq X \\ \pi \bullet X & \text{otherwise} \end{cases}$$

Substitution

- $\sigma(a.t) \stackrel{\text{def}}{=} a.\sigma(t)$
- $\sigma(\pi{\bullet}X) \stackrel{\text{def}}{=} \begin{cases} \pi \bullet \sigma(X) & \text{if } \sigma(X) \neq X \\ \pi{\bullet}X & \text{otherwise} \end{cases}$

for example

$$a.(a\,b){\bullet}X \;\; [X := \langle b, Y\rangle]$$

Substitution

- $$\sigma(a.t) \stackrel{\text{def}}{=} a.\sigma(t)$$
- $$\sigma(\pi\bullet X) \stackrel{\text{def}}{=} \begin{cases} \pi \bullet \sigma(X) & \text{if } \sigma(X) \neq X \\ \pi\bullet X & \text{otherwise} \end{cases}$$

for example

$$\underline{a.(a\,b)\bullet X \;\; [X := \langle b, Y\rangle]}$$
$$\Rightarrow\; a.\underline{(a\,b)\bullet X[X := \langle b, Y\rangle]}$$

Substitution

- $\sigma(a.t) \stackrel{\text{def}}{=} a.\sigma(t)$
- $\sigma(\pi \cdot X) \stackrel{\text{def}}{=} \begin{cases} \pi \cdot \sigma(X) & \text{if } \sigma(X) \neq X \\ \pi \cdot X & \text{otherwise} \end{cases}$

for example

$$
\begin{aligned}
& a.(a\,b)\cdot X\ \ [X := \langle b, Y\rangle] \\
\Rightarrow\ & a.\underline{(a\,b)\cdot X[X := \langle b, Y\rangle]} \\
\Rightarrow\ & a.(a\,b)\cdot\underline{\langle b, Y\rangle}
\end{aligned}
$$

Substitution

- $$\sigma(a.t) \stackrel{\text{def}}{=} a.\sigma(t)$$
- $$\sigma(\pi \bullet X) \stackrel{\text{def}}{=} \begin{cases} \pi \bullet \sigma(X) & \text{if } \sigma(X) \neq X \\ \pi \bullet X & \text{otherwise} \end{cases}$$

for example

$$\begin{aligned} & a.(a\,b)\bullet X \;\; [X := \langle b, Y \rangle] \\ \Rightarrow\; & a.(a\,b)\bullet X[X := \langle b, Y \rangle] \\ \Rightarrow\; & a.\underline{(a\,b)}\bullet \langle b, Y \rangle \\ \Rightarrow\; & a.\langle a, (a\,b)\bullet Y \rangle \end{aligned}$$

Substitution

- $\sigma(a.t) \stackrel{\text{def}}{=} a.\sigma(t)$
- $\sigma(\pi \cdot X) \stackrel{\text{def}}{=} \begin{cases} \pi \cdot \sigma(X) & \text{if } \sigma(X) \neq X \\ \pi \cdot X & \text{otherwise} \end{cases}$
- if $\nabla \vdash t \approx t'$ and $\nabla' \vdash \sigma(\nabla)$
  then $\nabla' \vdash \sigma(t) \approx \sigma(t')$

Substitution

- $\sigma(a.t) \stackrel{\text{def}}{=} a.\sigma(t)$

- $\sigma(\pi \cdot X) \stackrel{\text{def}}{=} \begin{cases} \pi \cdot \sigma(X) & \text{if } \sigma(X) \neq X \\ \pi \cdot X & \text{otherwise} \end{cases}$

- if $\nabla \vdash t \approx t'$ and $\nabla' \vdash \sigma(\nabla)$
  then $\nabla' \vdash \sigma(t) \approx \sigma(t')$

this means
$\nabla' \vdash a \# \sigma(X)$
holds for all
$(a \# X) \in \nabla$

Substitution

- $$\sigma(a.t) \stackrel{\text{def}}{=} a.\sigma(t)$$
- $$\sigma(\pi \cdot X) \stackrel{\text{def}}{=} \begin{cases} \pi \cdot \sigma(X) & \text{if } \sigma(X) \neq X \\ \pi \cdot X & \text{otherwise} \end{cases}$$
- if $\nabla \vdash t \approx t'$ and $\nabla' \vdash \sigma(\nabla)$
  then $\nabla' \vdash \sigma(t) \approx \sigma(t')$

Substitution

- $$\sigma(a.t) \stackrel{\text{def}}{=} a.\sigma(t)$$
- $$\sigma(\pi{\bullet}X) \stackrel{\text{def}}{=} \begin{cases} \pi \bullet \sigma(X) & \text{if } \sigma(X) \neq X \\ \pi{\bullet}X & \text{otherwise} \end{cases}$$
- if $\nabla \vdash t \approx t'$ and $\nabla' \vdash \sigma(\nabla)$
  then $\nabla' \vdash \sigma(t) \approx \sigma(t')$
- $$\sigma(\pi{\bullet}t) = \pi{\bullet}\sigma(t)$$

Equational Problems

An equational problem

$$t \approx ? \ t'$$

is solved by

- a substitution σ (terms for variables)
- **and** a set of freshness assumptions ∇

so that $\nabla \vdash \sigma(t) \approx \sigma(t')$.

Unifying equations may entail solving freshness problems.

E.g. assuming that $a \neq a'$, then

$$a.t \approx ? \; a'.t'$$

can only be solved if

$$t \approx ? \; (a\,a')\bullet t' \quad \text{and} \quad a \;\#?\; t'$$

can be solved.

Freshness Problems

A freshness problem

$$a \;\#?\; t$$

is solved by

- a substitution σ
- and a set of freshness assumptions ∇

so that $\nabla \vdash a \;\#\; \sigma(t)$.

Existence of MGUs

Theorem: There is an algorithm which, given a nominal unification problem $\boldsymbol{P}$, decides whether or not it has a solution $(\boldsymbol{\sigma}, \boldsymbol{\nabla})$, and returns a most general one if it does.

Existence of MGUs

Theorem: There is an algorithm which, given a nominal unification problem $\boldsymbol{P}$, decides whether or not it has a solution $(\boldsymbol{\sigma}, \boldsymbol{\nabla})$, and returns a most general one if it does.

most general:
straightforward definition
"iff there exists a $\boldsymbol{\tau}$ such that ..."

Existence of MGUs

Theorem: There is an algorithm which, given a nominal unification problem $\boldsymbol{P}$, decides whether or not it has a solution $(\boldsymbol{\sigma}, \boldsymbol{\nabla})$, and returns a most general one if it does.

Proof: one can reduce all the equations to 'solved form' first (creating a substitution), and then solve the freshness problems (easy).

Remember the Quiz?

Assuming that a and b are distinct variables, is it possible to find λ-terms M_1 to M_7 that make the following pairs α-equivalent?

- $\lambda a.\lambda b.(M_1\, b)$ and $\lambda b.\lambda a.(a\, M_1)$
- $\lambda a.\lambda b.(M_2\, b)$ and $\lambda b.\lambda a.(a\, M_3)$
- $\lambda a.\lambda b.(b\, M_4)$ and $\lambda b.\lambda a.(a\, M_5)$
- $\lambda a.\lambda b.(b\, M_6)$ and $\lambda a.\lambda a.(a\, M_7)$

If there is one solution for a pair, can you describe all its solutions?

Answers to the Quiz

$\lambda a.\lambda b.(M_1\, b)$ and $\lambda b.\lambda a.(a\, M_1)$

Answers to the Quiz

$$a.b.\langle M_1, b\rangle \;\approx ?\; b.a.\langle a, M_1\rangle$$

Answers to the Quiz

$$a.b.\langle M_1, b\rangle \;\approx?\; b.a.\langle a, M_1\rangle$$

$$\stackrel{\varepsilon}{\Longrightarrow}\quad b.\langle M_1, b\rangle \approx? (a\ b)\bullet a.\langle a, M_1\rangle\ ,\ a\ \#?\ a.\langle a, M_1\rangle$$

Answers to the Quiz

$$\boldsymbol{a.b.\langle M_1, b\rangle} \approx ? \boldsymbol{b.a.\langle a, M_1\rangle}$$

$$\stackrel{\varepsilon}{\Longrightarrow} \quad \boldsymbol{b.\langle M_1, b\rangle} \approx ? \boldsymbol{b.\langle b, (a\,b)\bullet M_1\rangle}\ ,\ \boldsymbol{a} \# ? \boldsymbol{a.\langle a, M_1\rangle}$$

Answers to the Quiz

$$a.b.\langle M_1, b\rangle \;\approx?\; b.a.\langle a, M_1\rangle$$

$$\overset{\varepsilon}{\Longrightarrow} \quad b.\langle M_1, b\rangle \approx? \, b.\langle b, (a\,b)\bullet M_1\rangle \;,\; a \,\#?\, a.\langle a, M_1\rangle$$

$$\overset{\varepsilon}{\Longrightarrow} \quad \langle M_1, b\rangle \approx? \, \langle b, (a\,b)\bullet M_1\rangle \;,\; a \,\#?\, a.\langle a, M_1\rangle$$

Answers to the Quiz

$$a.b.\langle M_1, b\rangle \;\approx?\; b.a.\langle a, M_1\rangle$$

$$\stackrel{\varepsilon}{\Longrightarrow}\quad b.\langle M_1, b\rangle \approx? \; b.\langle b, (a\,b)\bullet M_1\rangle \;,\; a \;\#?\; a.\langle a, M_1\rangle$$

$$\stackrel{\varepsilon}{\Longrightarrow}\quad \langle M_1, b\rangle \approx? \; \langle b, (a\,b)\bullet M_1\rangle \;,\; a \;\#?\; a.\langle a, M_1\rangle$$

$$\stackrel{\varepsilon}{\Longrightarrow}\quad M_1 \approx? \; b \;,\; b \approx? \; (a\,b)\bullet M_1 \;,\; a \;\#?\; a.\langle a, M_1\rangle$$

Answers to the Quiz

$$a.b.\langle M_1, b\rangle \ \approx ?\ \ b.a.\langle a, M_1\rangle$$

$$\overset{\varepsilon}{\Longrightarrow}\ \ b.\langle M_1, b\rangle \approx ?\ b.\langle b, (a\,b)\bullet M_1\rangle\ ,\ a \ \#?\ a.\langle a, M_1\rangle$$

$$\overset{\varepsilon}{\Longrightarrow}\ \ \langle M_1, b\rangle \approx ?\ \langle b, (a\,b)\bullet M_1\rangle\ ,\ a \ \#?\ a.\langle a, M_1\rangle$$

$$\overset{\varepsilon}{\Longrightarrow}\ \ M_1 \approx ?\ b\ ,\ b \approx ?\ (a\,b)\bullet M_1\ ,\ a \ \#?\ a.\langle a, M_1\rangle$$

$$\overset{[M_1:=b]}{\Longrightarrow}\ \ b \approx ?\ (a\,b)\bullet b\ ,\ a \ \#?\ a.\langle a, b\rangle$$

Answers to the Quiz

$$a.b.\langle M_1, b\rangle \ \approx? \ b.a.\langle a, M_1\rangle$$

$$\overset{\varepsilon}{\Longrightarrow} \quad b.\langle M_1, b\rangle \approx? b.\langle b, (a\,b)\bullet M_1\rangle \ , \ a \ \#? \ a.\langle a, M_1\rangle$$

$$\overset{\varepsilon}{\Longrightarrow} \quad \langle M_1, b\rangle \approx? \langle b, (a\,b)\bullet M_1\rangle \ , \ a \ \#? \ a.\langle a, M_1\rangle$$

$$\overset{\varepsilon}{\Longrightarrow} \quad M_1 \approx? b \ , \ b \approx? (a\,b)\bullet M_1 \ , \ a \ \#? \ a.\langle a, M_1\rangle$$

$$\overset{[M_1:=b]}{\Longrightarrow} \quad b \approx? a \ , \ a \ \#? \ a.\langle a, b\rangle$$

Answers to the Quiz

$$a.b.\langle M_1, b\rangle \;\approx?\; b.a.\langle a, M_1\rangle$$

$$\overset{\varepsilon}{\Longrightarrow}\; b.\langle M_1, b\rangle \approx? b.\langle b, (a\,b)\bullet M_1\rangle \;,\; a \,\#?\, a.\langle a, M_1\rangle$$

$$\overset{\varepsilon}{\Longrightarrow}\; \langle M_1, b\rangle \approx? \langle b, (a\,b)\bullet M_1\rangle \;,\; a \,\#?\, a.\langle a, M_1\rangle$$

$$\overset{\varepsilon}{\Longrightarrow}\; M_1 \approx? b \;,\; b \approx? (a\,b)\bullet M_1 \;,\; a \,\#?\, a.\langle a, M_1\rangle$$

$$\overset{[M_1:=b]}{\Longrightarrow}\; b \approx? a \;,\; a \,\#?\, a.\langle a, b\rangle$$

$$\Longrightarrow\; FAIL$$

Answers to the Quiz

$$a.b.\langle M_1, b\rangle \ \approx?\ \ b.a.\langle a, M_1\rangle$$

$$\stackrel{\varepsilon}{\Longrightarrow}\quad b.\langle M_1, b\rangle \approx? \ b.\langle b, (a\,b)\bullet M_1\rangle\ ,\ a \ \#? \ a.\langle a, M_1\rangle$$

$$\stackrel{\varepsilon}{\Longrightarrow}\quad \langle M_1, b\rangle \approx? \ \langle b, (a\,b)\bullet M_1\rangle\ ,\ a \ \#? \ a.\langle a, M_1\rangle$$

$$\stackrel{\varepsilon}{\Longrightarrow}\quad M_1 \approx? \ b\ ,\ b \approx? \ (a\,b)\bullet M_1\ ,\ a \ \#? \ a.\langle a, M_1\rangle$$

$$\stackrel{[M_1:=b]}{\Longrightarrow}\quad b \approx? \ a\ ,\ a \ \#? \ a.\langle a, b\rangle$$

$$\Longrightarrow\quad FAIL$$

$\lambda a.\lambda b.(M_1\ b) =_\alpha \lambda b.\lambda a.(a\ M_1)$ has no solution

Answers to the Quiz

$\lambda a.\lambda b.(b\, M_6)$ and $\lambda a.\lambda a.(a\, M_7)$

Answers to the Quiz

$a.b.\langle b, M_6\rangle \approx ? \; a.a.\langle a, M_7\rangle$

Answers to the Quiz

$$a.b.\langle b, M_6\rangle \;\approx?\; a.a.\langle a, M_7\rangle$$

$$\stackrel{\varepsilon}{\Longrightarrow}\; b.\langle b, M_6\rangle \approx? a.\langle a, M_7\rangle$$

Answers to the Quiz

$$a.b.\langle b, M_6\rangle \;\approx?\; a.a.\langle a, M_7\rangle$$

$$\stackrel{\varepsilon}{\Longrightarrow}\; b.\langle b, M_6\rangle \approx? \; a.\langle a, M_7\rangle$$

$$\stackrel{\varepsilon}{\Longrightarrow}\; \langle b, M_6\rangle \approx? \; \langle b, (b\, a)\bullet M_7\rangle \;,\; b \;\#?\; \langle a, M_7\rangle$$

Answers to the Quiz

$$a.b.\langle b, M_6\rangle \;\approx?\; a.a.\langle a, M_7\rangle$$

$$\overset{\varepsilon}{\Longrightarrow}\; b.\langle b, M_6\rangle \approx? a.\langle a, M_7\rangle$$

$$\overset{\varepsilon}{\Longrightarrow}\; \langle b, M_6\rangle \approx? \langle b, (b\,a)\bullet M_7\rangle \;,\; b \;\#?\; \langle a, M_7\rangle$$

$$\overset{\varepsilon}{\Longrightarrow}\; b \approx? b \;,\; M_6 \approx? (b\,a)\bullet M_7 \;,\; b \;\#?\; \langle a, M_7\rangle$$

Answers to the Quiz

$$a.b.\langle b, M_6\rangle \approx ?\ a.a.\langle a, M_7\rangle$$

$$\overset{\varepsilon}{\Longrightarrow}\ b.\langle b, M_6\rangle \approx ?\ a.\langle a, M_7\rangle$$

$$\overset{\varepsilon}{\Longrightarrow}\ \langle b, M_6\rangle \approx ?\ \langle b, (b\,a)\bullet M_7\rangle\ ,\ b \,\#?\ \langle a, M_7\rangle$$

$$\overset{\varepsilon}{\Longrightarrow}\ b \approx ?\ b\ ,\ M_6 \approx ?\ (b\,a)\bullet M_7\ ,\ b \,\#?\ \langle a, M_7\rangle$$

$$\overset{\varepsilon}{\Longrightarrow}\ M_6 \approx ?\ (b\,a)\bullet M_7\ ,\ b \,\#?\ \langle a, M_7\rangle$$

Answers to the Quiz

$$a.b.\langle b, M_6\rangle \approx ? \; a.a.\langle a, M_7\rangle$$

$$\overset{\varepsilon}{\Longrightarrow} b.\langle b, M_6\rangle \approx ? \; a.\langle a, M_7\rangle$$

$$\overset{\varepsilon}{\Longrightarrow} \langle b, M_6\rangle \approx ? \; \langle b, (b\,a)\bullet M_7\rangle \; , \; b \;\#? \; \langle a, M_7\rangle$$

$$\overset{\varepsilon}{\Longrightarrow} b \approx ? \; b \; , \; M_6 \approx ? \; (b\,a)\bullet M_7 \; , \; b \;\#? \; \langle a, M_7\rangle$$

$$\overset{\varepsilon}{\Longrightarrow} M_6 \approx ? \; (b\,a)\bullet M_7 \; , \; b \;\#? \; \langle a, M_7\rangle$$

$$\overset{[M_6:=(b\,a)\bullet M_7]}{\Longrightarrow} b \;\#? \; \langle a, M_7\rangle$$

Answers to the Quiz

$$a.b.\langle b, M_6\rangle \;\approx?\; a.a.\langle a, M_7\rangle$$

$$\overset{\varepsilon}{\Longrightarrow}\; b.\langle b, M_6\rangle \approx? \; a.\langle a, M_7\rangle$$

$$\overset{\varepsilon}{\Longrightarrow}\; \langle b, M_6\rangle \approx? \; \langle b, (b\,a)\bullet M_7\rangle \;,\; b \,\#?\, \langle a, M_7\rangle$$

$$\overset{\varepsilon}{\Longrightarrow}\; b \approx? \; b \;,\; M_6 \approx? \; (b\,a)\bullet M_7 \;,\; b \,\#?\, \langle a, M_7\rangle$$

$$\overset{\varepsilon}{\Longrightarrow}\; M_6 \approx? \; (b\,a)\bullet M_7 \;,\; b \,\#?\, \langle a, M_7\rangle$$

$$\overset{[M_6:=(b\,a)\bullet M_7]}{\Longrightarrow}\; b \,\#?\, \langle a, M_7\rangle$$

$$\overset{\varnothing}{\Longrightarrow}\; b \,\#?\, a \;,\; b \,\#?\, M_7$$

Answers to the Quiz

$$a.b.\langle b, M_6\rangle \approx ? \; a.a.\langle a, M_7\rangle$$

$$\overset{\varepsilon}{\Longrightarrow} b.\langle b, M_6\rangle \approx ? \; a.\langle a, M_7\rangle$$

$$\overset{\varepsilon}{\Longrightarrow} \langle b, M_6\rangle \approx ? \; \langle b, (b\,a)\bullet M_7\rangle \; , \; b \; \#? \; \langle a, M_7\rangle$$

$$\overset{\varepsilon}{\Longrightarrow} b \approx ? \; b \; , \; M_6 \approx ? \; (b\,a)\bullet M_7 \; , \; b \; \#? \; \langle a, M_7\rangle$$

$$\overset{\varepsilon}{\Longrightarrow} M_6 \approx ? \; (b\,a)\bullet M_7 \; , \; b \; \#? \; \langle a, M_7\rangle$$

$$\overset{[M_6:=(b\,a)\bullet M_7]}{\Longrightarrow} b \; \#? \; \langle a, M_7\rangle$$

$$\overset{\varnothing}{\Longrightarrow} b \; \#? \; a \; , \; b \; \#? \; M_7$$

$$\overset{\varnothing}{\Longrightarrow} b \; \#? \; M_7$$

Answers to the Quiz

$$a.b.\langle b, M_6\rangle \;\approx?\; a.a.\langle a, M_7\rangle$$

$$\overset{\varepsilon}{\Longrightarrow}\; b.\langle b, M_6\rangle \approx? a.\langle a, M_7\rangle$$

$$\overset{\varepsilon}{\Longrightarrow}\; \langle b, M_6\rangle \approx? \langle b, (b\,a)\bullet M_7\rangle \;,\; b \,\#?\, \langle a, M_7\rangle$$

$$\overset{\varepsilon}{\Longrightarrow}\; b \approx? b \;,\; M_6 \approx? (b\,a)\bullet M_7 \;,\; b \,\#?\, \langle a, M_7\rangle$$

$$\overset{\varepsilon}{\Longrightarrow}\; M_6 \approx? (b\,a)\bullet M_7 \;,\; b \,\#?\, \langle a, M_7\rangle$$

$$\overset{[M_6:=(b\,a)\bullet M_7]}{\Longrightarrow}\; b \,\#?\, \langle a, M$$

$$\overset{\varnothing}{\Longrightarrow}\; b \,\#?\, a \;,\; b \,\#?\,$$

$$\overset{\varnothing}{\Longrightarrow}\; b \,\#?\, M_7$$

$$\overset{\{b\#M_7\}}{\Longrightarrow}\; \varnothing$$

$$\lambda a.\lambda b.(b\,M_6) \;=_\alpha\; \lambda a.\lambda a.(a\,M_7)$$

we can take M_7 to be any λ-term that does not contain free occurrences of b, so long as we take M_6 to be the result of swapping all occurrences of b and a throughout M_7

Properties

- An interesting feature of nominal unification is that it does not need to create new atoms.

$$\{a.t \approx ?\ b.t'\} \cup P \stackrel{\varepsilon}{\Longrightarrow} \{t \approx ?\ (a\,b){\bullet}t',\, a\ \#?\ t'\} \cup P$$

Properties

- An interesting feature of nominal unification is that it does not need to create new atoms.

$$\{a.t \approx ? \; b.t'\} \cup P \stackrel{\varepsilon}{\Longrightarrow} \{t \approx ? \; (a\, b) \bullet t', a \; \# ? \; t'\} \cup P$$

- The alternative rule

$$\{a.t \approx ? \; b.t'\} \cup P \stackrel{\varepsilon}{\Longrightarrow} \{(a\, c) \bullet t \approx ? \; (b\, c) \bullet t', c \; \# ? \; t, c \; \# ? \; t'\} \cup P$$

leads to a more complicated notion of mgu.

Properties

- An interesting feature of nominal unification is that it does not need to create new atoms.

$$\{a.t \approx ?\ b.t'\} \cup P \stackrel{\varepsilon}{\Longrightarrow} \{t \approx ?\ (a\,b)\bullet t', a \,\#?\ t'\} \cup P$$

- The alternative rule

$$\{a.t \approx ?\ b.t'\} \cup P \stackrel{\varepsilon}{\Longrightarrow} \{(a\,c)\bullet t \approx ?\ (b\,c)\bullet t', c \,\#?\ t, c \,\#?\ t'\} \cup P$$

leads to a more complicated notion of mgu.

$\{a.X \approx ?\ b.Y\} \Rightarrow (\{a \# Y, c \# Y\}, [X := (a\,c)(b\,c)\bullet Y])$

Is it Useful?

Yes. αProlog by James Cheney (main developer)

```
type (Gamma, var(X), T) :- member (X,T) Gamma.

type (Gamma, app(M, N), T') :-
                type (Gamma, M, arrow(T, T')),
                type (Gamma, N, T).

type (Gamma, lam(x.M), arrow(T, T')) / x # Gamma :-
                type ((x, T)::Gamma, M, T').

member X X::Tail.
member X Y::Tail :- member X Tail.
```

One problem: If we ask whether

?- type ([(x, T')], lam(x.Var(x)), T)

is typable, we expect an answer for T.

type (Gamma, lam(x.M), arrow(T, T')) / x # Gamma :-
type ((x, T)::Gamma, M, T').

member X X::Tail.
member X Y::Tail :- member X Tail.

One problem: If we ask whether

?- type ([(x, T')], lam(x.Var(x)), T)

is typable, we expect an answer for T.

Solution: Before back-chaining freshen all variables and atoms in a program (clause).

type (Gamma, lam(x.M), arrow(T, T')) / x # Gamma :-
type ((x, T)::Gamma, M, T').

member X X::Tail.
member X Y::Tail :- member X Tail.

Equivariant Unification

James Cheney proposed

$$t \approx ? \; t' \stackrel{\nabla,\sigma,\pi}{\Longrightarrow} \nabla \vdash \sigma(t) \approx \pi \bullet \sigma(t')$$

Equivariant Unification

James Cheney proposed

$$t \approx ? \; t' \stackrel{\nabla,\sigma,\pi}{\Longrightarrow} \nabla \vdash \sigma(t) \approx \pi \bullet \sigma(t')$$

But he also showed this problem is undecidable in general. :(

Taking Atoms as Variables

Instead of $a.X$, have $A.X$.

Taking Atoms as Variables

Instead of $a.X$, have $A.X$.

Unfortunately this breaks the mgu-property:

$$a.Z \approx ?\ X.Y.v(a)$$

can be solved by

$$[X := a, Z := Y.v(a)] \text{ and}$$
$$[Y := a, Z := Y.v(Y)]$$

HOPU vs. NOMU

- James Cheney showed

$$HOPU \Rightarrow NOMU$$

- Jordi Levy and Mateu Villaret established

$$HOPU \Leftarrow NOMU$$

The translations ‘explode’ the problems quadratically.

From: Zhenyu Qian <zhqian@microsoft.com>
To: Christian Urban <urbanc@in.tum.de>
Subject: RE: Linear Higher-Order Pattern Unification
Date: Mon, 14 Apr 2008 09:56:47 +0800

Hi Christian,

Thanks for your interests and asking. I know that that paper is complex. As I told Tobias when we met last time, I have raised the question to myself many times whether the proof could have some flaws, and so making it through a theorem prover would definitely bring piece to my mind (no matter what the result would be). The only problem for me is the time.

…

Thanks/Zhenyu

Complexity

- Christiopher Calves and Maribel Fernandez showed first that it is polynomial and then also quadratic
- Jordi Levy and Mateu Villaret showed that it is quadratic by a translation into a subset of NOMU and using ideas from Martelli/Montenari.

Conclusion

- Nominal Unification is a completely first-order language, but implements unification modulo α. (verification...Ramana Kumar and Michael Norrish)

Conclusion

- Nominal Unification is a completely first-order language, but implements unification modulo α. (verification...Ramana Kumar and Michael Norrish)
- NOMU has been applied in term-rewriting and logic programming. (Maribel Fernandez et al has a KB-completion procedure.) I hope it will also be used in typing systems.

Conclusion

- Nominal Unification is a completely first-order language, but implements unification modulo α. (verification...Ramana Kumar and Michael Norrish)
- NOMU has been applied in term-rewriting and logic programming. (Maribel Fernandez et al has a KB-completion procedure.) I hope it will also be used in typing systems.
- NOMU and HOPU are 'equivalent' (it took a long time and considerable research to find this out).

Conclusion

- Nominal Unification is a completely first-order language, but implements unification modulo α. (verification...Ramana Kumar and Michael Norrish)
- NOMU has been applied in term-rewriting and logic programming. (Maribel Fernandez et al has a KB-completion procedure.) I hope it will also be used in typing systems.
- NOMU and HOPU are 'equivalent' (it took a long time and considerable research to find this out).
- The question about complexity is still an ongoing story.

Thank you very much!

Questions?

Most General Unifiers

Definition: For a unification problem P, a solution (σ_1, ∇_1) is more general than another solution (σ_2, ∇_2), iff there exists a substitution τ with

- $\nabla_2 \vdash \tau(\nabla_1)$
- $\nabla_2 \vdash \sigma_2 \approx \tau \circ \sigma_1$

Most General Unifiers

Definition: For a unification problem P, a solution (σ_1, ∇_1) is more general than another solution (σ_2, ∇_2), iff there exists a substitution τ with

- $\nabla_2 \vdash \tau(\nabla_1)$
- $\nabla_2 \vdash \sigma_2 \approx \tau \circ \sigma_1$

$\nabla_2 \vdash a \# \sigma(X)$ holds for all $(a \# X) \in \nabla_1$

Most General Unifiers

Definition: For a unification problem P, a solution (σ_1, ∇_1) is more general than another solution (σ_2, ∇_2), iff there exists a substitution τ with

- $\nabla_2 \vdash \tau(\nabla_1)$
- $\nabla_2 \vdash \sigma_2 \approx \tau \circ \sigma_1$

$\nabla_2 \vdash \sigma_2(X) \approx \sigma(\sigma_1(X))$ holds for all $X \in \mathrm{dom}(\sigma_2) \cup \mathrm{dom}(\sigma \circ \sigma_1)$