

Quiz 1

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?

Quiz 2

The following Prolog-program implements, according to a text book, the typing-rules for the simply-typed lambda calculus. What is wrong with it?

$$\frac{x:A \in \Gamma}{\Gamma \vdash x:A} \quad \frac{\Gamma \vdash M:A \rightarrow B \quad \Gamma \vdash N:A}{\Gamma \vdash MN:B} \quad \frac{x:A, \Gamma \vdash M:B}{\Gamma \vdash \lambda x.M:A \rightarrow B}$$

```
type Gamma (var X) A :- member (pair X A) Gamma.  
type Gamma (app M N) B :- type Gamma M (arrow A B),  
                           type Gamma N A.  
type Gamma (lam X M) (arrow A B) :-  
                                   type (pair X A)::Gamma M B.  
  
member A A::Tail.  
member A B::Tail :- member A Tail.
```

Meta-Variables

Meta-variables work with possibly-capturing substitutions, e.g.

$$\frac{\text{app}(\text{fn } a.Y, X) \Downarrow V}{\text{let } a = X \text{ in } Y \Downarrow V} \text{ scheme}$$

$$\text{let } a = 1 \text{ in } a \Downarrow 1$$

Meta-Variables

Meta-variables work with possibly-capturing substitutions, e.g.

$$\frac{\text{app}(\text{fn } a.Y, X) \Downarrow V}{\text{let } a = X \text{ in } Y \Downarrow V} \quad \text{scheme}$$

$$\text{let } a = 1 \text{ in } a \Downarrow 1$$
$$[Y := a; X, V := 1]$$

correct instance

$$\frac{\text{app}(\text{fn } a.a, 1) \Downarrow 1}{\text{let } a = 1 \text{ in } a \Downarrow 1}$$

Meta-Variables

Meta-variables work with possibly-capturing substitutions, e.g.

$$\frac{\text{app}(\text{fn } a.Y, X) \Downarrow V}{\text{let } a = X \text{ in } Y \Downarrow V} \quad \text{scheme}$$

$$\text{let } a = 1 \text{ in } a \Downarrow 1 \quad =_{\alpha} \quad \text{let } b = 1 \text{ in } b \Downarrow 1 \\ [Y := a; X, V := 1]$$

correct instance

$$\frac{\text{app}(\text{fn } a.a, 1) \Downarrow 1}{\text{let } a = 1 \text{ in } a \Downarrow 1}$$

Meta-Variables

Meta-variables work with possibly-capturing substitutions, e.g.

$$\frac{\text{app}(\text{fn } a.Y, X) \Downarrow V}{\text{let } a = X \text{ in } Y \Downarrow V} \quad \text{scheme}$$

$$\text{let } a = 1 \text{ in } a \Downarrow 1 \quad =_{\alpha} \quad \text{let } b = 1 \text{ in } b \Downarrow 1$$
$$[Y := a; X, V := 1] \quad [Y := b; X, V := 1]$$

correct instance

$$\frac{\text{app}(\text{fn } a.a, 1) \Downarrow 1}{\text{let } a = 1 \text{ in } a \Downarrow 1}$$

incorrect instance

$$\frac{\text{app}(\text{fn } a.b, 1) \Downarrow 1}{\text{let } b = 1 \text{ in } b \Downarrow 1}$$

HOAS

This style of reasoning can be made precise by using higher-order abstract syntax (HOAS) and higher-order unification (capture-avoiding substitutions).

$$\frac{\text{app (fn } \lambda a.F(a)) X \Downarrow V}{\text{let } X \lambda a.F(a) \Downarrow V}$$

λ -calc.

HOAS

This style of reasoning can be made precise by using higher-order abstract syntax (HOAS) and higher-order unification (capture-avoiding substitutions).

$$\frac{\text{app (fn } F) X \Downarrow V}{\text{let } X F \Downarrow V} \quad \lambda\text{-calc.}$$

HOAS

This style of reasoning can be made precise by using higher-order abstract syntax (HOAS) and higher-order unification (capture-avoiding substitutions).

$$\frac{\text{app (fn } F) X \Downarrow V}{\text{let } X F \Downarrow V} \quad \lambda\text{-calc.}$$

$$\text{let } 1 \lambda a.a \Downarrow 1$$

$$\text{let } 1 \lambda b.b \Downarrow 1$$

$$\frac{\text{app (fn } \lambda a.a) 1 \Downarrow 1}{\text{let } 1 \lambda a.a \Downarrow 1}$$

$$\frac{\text{app (fn } \lambda b.b) 1 \Downarrow 1}{\text{let } 1 \lambda b.b \Downarrow 1}$$

Drawbacks

- We targeted α , but have to deal with β (or Miller's β_0 , at least) as well.
- Unification theory is not simple.
- Informal practice suggests that leaving name dependencies implicit can be convenient.
- Combining HOAS and structural induction can be a nightmare.

Drawbacks

- We targeted α , but have to deal with β (or Miller's β_0 , at least) as well.
- Unification theory is not simple.
- Informal practice suggests that leaving name dependencies implicit can be convenient.
- Combining HOAS and structural induction can be a nightmare.

Do we have to put up with this?

Drawbacks

- We targeted α , but have to deal with β (or Miller's β_0 , at least) as well.
- Unification theory is not simple.
- Informal practice suggests that leaving name dependencies implicit can be convenient.
- Combining HOAS and structural induction can be a nightmare.

Do we have to put up with this? **No!**

Swappings

Problem: substitution does not respect α -equivalence, e.g.

fn a.b

fn c.b

Swappings

Problem: substitution does not respect α -equivalence, e.g.

$$\begin{array}{ll} [b := a] \text{fn } a.b & [b := a] \text{fn } c.b \\ = \text{fn } a.a & = \text{fn } c.a \end{array}$$

Swappings

Problem: substitution does not respect α -equivalence, e.g.

$$\begin{array}{ll} [b := a] \text{fn } a.b & [b := a] \text{fn } c.b \\ = \text{fn } a.a & = \text{fn } c.a \end{array}$$

Traditional Solution: replace $[b := a]t$ by a more complicated, 'capture-avoiding' form of substitution.

Swappings

Problem: substitution does not respect α -equivalence, e.g.

$$\begin{array}{ll} (b\ a) \cdot \text{fn } a.b & (b\ a) \cdot \text{fn } c.b \\ = \text{fn } b.a & = \text{fn } c.a \end{array}$$

Nice Alternative: use a less complicated operation for renaming.

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

Swappings

Problem: substitution does not respect α -equivalence, e.g.

$$\begin{array}{ll} (b\ a) \cdot \text{fn } a.b & (b\ a) \cdot \text{fn } c.b \\ = \text{fn } b.a & = \text{fn } c.a \end{array}$$

Nice Alternative: use a less complicated operation for renaming.

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

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

Terms

- $\langle \rangle$ Units
- $\langle t, t' \rangle$ Pairs
- $F t$ Fun. Symb's

Terms

- $\langle \rangle$ Units
- $\langle t, t' \rangle$ Pairs
- $F t$ Fun. Symb's
- a Atoms
- $a.t$ Abstractions

bindable names
(of object-level
variables etc.)

generic binder:

$\lceil \lambda a.a \rceil \mapsto \text{fn } a.a$

constructions like
 $\text{fn } X.X$ are not
allowed

Terms

- $\langle \rangle$ Units
- $\langle t, t' \rangle$ Pairs
- $F t$ Fun. Symb's
- a Atoms
- $a.t$ Abstractions
- $\pi \cdot X$ Suspensions

π is an explicit permutation, which is a list of swappings $(a_1 b_1) \dots (a_n b_n)$, waiting to be applied to the term that is substituted for X

X is a variable standing for an unknown term

Permutations

A permutation applied to a term:

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

Permutations

A permutation applied to a term:

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

Permutations

A permutation applied to a term:

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

Permutations

A permutation applied to a term:

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

Permutations on atoms are bijections!

$$\pi \cdot a = b \quad \text{iff} \quad a = (\pi^{-1}) \cdot b$$

Freshness Relation

We will identify

$$\text{fn } a.X \approx \text{fn } b.(a b).X$$

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 Relation

We will identify

$$\text{fn } a.X \approx \text{fn } b.(a\ b).X$$

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

explicit permutation —
waits to be applied to
the term that is substituted
for X

Freshness Relation

We will identify

$$\text{fn } a.X \approx \text{fn } b.(a b).X$$

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 Relation

We will identify

$$\text{fn } a.X \approx \text{fn } b.(a b).X$$

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 $(a b).X$ by X .

Freshness Assumptions

Our equality is **not** just

$t \approx t'$ α -equivalence

Freshness Assumptions

but judgements

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

where

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

is a finite set of **freshness assumptions**.

Freshness Assumptions

but judgements

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

where

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

is a finite set of **freshness assumptions**.

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

Freshness Assumptions

but judgements

$\nabla \vdash t \approx t'$ α -equivalence

$\nabla \vdash a \# t$ freshness

where

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

is a finite set of **freshness assumptions**.

$$\begin{array}{l} \{b \# X\} \vdash b \# a.X \\ \{\} \vdash a \# a.X \end{array}$$

Rules for Equivalence

Excerpt
(i.e. only the interesting rules)

Rules for Equivalence

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

$$\frac{a \neq b \quad \nabla \vdash t \approx (a \ b) \cdot t' \quad \nabla \vdash a \# t'}{\nabla \vdash a.t \approx b.t'}$$

Rules for Equivalence

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

Rules for Equivalence

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

for example

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

Rules for Freshness

Excerpt
(i.e. only the interesting rules)

Rules for Freshness

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

$$\frac{}{\nabla \vdash a \# a.t}$$

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

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

\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 $\nabla \vdash \pi \cdot t \approx \pi \cdot t'$
- $\nabla \vdash a \# t$ then $\nabla \vdash \pi \cdot a \# \pi \cdot t$

\approx is an Equivalence

Theorem: \approx is an equivalence relation.

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

\approx is an Equivalence

Theorem: \approx is an equivalence relation.

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

Comparison with $=_{\alpha}$

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 with $=_{\alpha}$

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:

Theorem: $t =_{\alpha} t'$ iff $\emptyset \vdash t \approx t'$
 $a \notin FA(t)$ iff $\emptyset \vdash a \# t$

Comparison with $=_{\alpha}$

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

$\emptyset \vdash a.X \approx b.X$ ($a \neq b$)

Comparison with $=_{\alpha}$

That is a crucial point: if we had

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

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.

Substitutions

- $\sigma(a.t) \stackrel{\text{def}}{=} a.\sigma(t)$
- $\sigma(\pi.X) \stackrel{\text{def}}{=} \begin{cases} \pi \cdot \sigma(X) & \text{if } \sigma(X) \neq X \\ \pi \cdot X & \text{o'wise do nothing} \end{cases}$

Substitutions

- $\sigma(a.t) \stackrel{\text{def}}{=} a.\sigma(t)$
- $\sigma(\pi.X) \stackrel{\text{def}}{=} \begin{cases} \pi \cdot \sigma(X) & \text{if } \sigma(X) \neq X \\ \pi.X & \text{o'wise do nothing} \end{cases}$

for example

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

Substitutions

- $\sigma(a.t) \stackrel{\text{def}}{=} a.\sigma(t)$
- $\sigma(\pi.X) \stackrel{\text{def}}{=} \begin{cases} \pi \cdot \sigma(X) & \text{if } \sigma(X) \neq X \\ \pi.X & \text{o'wise do nothing} \end{cases}$

for example

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

Substitutions

- $\sigma(a.t) \stackrel{\text{def}}{=} a.\sigma(t)$
- $\sigma(\pi.X) \stackrel{\text{def}}{=} \begin{cases} \pi \cdot \sigma(X) & \text{if } \sigma(X) \neq X \\ \pi.X & \text{o'wise do nothing} \end{cases}$

for example

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

Substitutions

- $\sigma(a.t) \stackrel{\text{def}}{=} a.\sigma(t)$
- $\sigma(\pi.X) \stackrel{\text{def}}{=} \begin{cases} \pi \cdot \sigma(X) & \text{if } \sigma(X) \neq X \\ \pi.X & \text{o'wise do nothing} \end{cases}$

for example

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

Substitutions

- $\sigma(a.t) \stackrel{\text{def}}{=} a.\sigma(t)$
- $\sigma(\pi.X) \stackrel{\text{def}}{=} \begin{cases} \pi \cdot \sigma(X) & \text{if } \sigma(X) \neq X \\ \pi \cdot X & \text{o'wise do nothing} \end{cases}$
- if $\nabla \vdash t \approx t'$ and $\nabla' \vdash \sigma(\nabla)$
then $\nabla' \vdash \sigma(t) \approx \sigma(t')$

Substitutions

- $\sigma(a.t) \stackrel{\text{def}}{=} a.\sigma(t)$
- $\sigma(\pi.X) \stackrel{\text{def}}{=} \begin{cases} \pi \cdot \sigma(X) & \text{if } \sigma(X) \neq X \\ \pi.X & \text{o'wise do nothing} \end{cases}$
- 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

$(a \# X) \in \nabla$

Substitutions

- $\sigma(a.t) \stackrel{\text{def}}{=} a.\sigma(t)$
- $\sigma(\pi.X) \stackrel{\text{def}}{=} \begin{cases} \pi \cdot \sigma(X) & \text{if } \sigma(X) \neq X \\ \pi \cdot X & \text{o'wise do nothing} \end{cases}$
- if $\nabla \vdash t \approx t'$ and $\nabla' \vdash \sigma(\nabla)$
then $\nabla' \vdash \sigma(t) \approx \sigma(t')$
- $\sigma(\pi.t) = \pi \cdot \sigma(t)$

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

Freshness Problems

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

$$t \approx? (a a') \cdot t' \quad \text{and} \quad a \#? t'$$

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 P , decides whether or not it has a solution (σ, ∇) , and returns a most general one if it does.

Existence of MGUs

Theorem: there is an algorithm which, given a nominal unification problem P , decides whether or not it has a solution (σ, ∇) , and returns a **most general** one if it does.



straightforward definition:
"iff there exists a τ such that ..."

Existence of MGUs

Theorem: there is an algorithm which, given a nominal unification problem P , decides whether or not it has a solution (σ, ∇) , 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

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

$$\xRightarrow{\varepsilon} b.\langle M_1, b \rangle \approx? (a\ b) \cdot a.\langle a, M_1 \rangle, \quad a \# a.\langle a, M_1 \rangle$$

Answers to the Quiz

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

$$\xRightarrow{\varepsilon} b.\langle M_1, b \rangle \approx? b.\langle b, (a b) \cdot M_1 \rangle, a \# a.\langle a, M_1 \rangle$$

Answers to the Quiz

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

$$\xRightarrow{\varepsilon} b.\langle M_1, b \rangle \approx? b.\langle b, (a b) \cdot M_1 \rangle, a \# a.\langle a, M_1 \rangle$$

$$\xRightarrow{\varepsilon} \langle M_1, b \rangle \approx? \langle b, (a b) \cdot M_1 \rangle, a \#? a.\langle a, M_1 \rangle$$

Answers to the Quiz

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

$$\xRightarrow{\varepsilon} b.\langle M_1, b \rangle \approx? b.\langle b, (a b) \cdot M_1 \rangle, a \# a.\langle a, M_1 \rangle$$

$$\xRightarrow{\varepsilon} \langle M_1, b \rangle \approx? \langle b, (a b) \cdot M_1 \rangle, a \#? a.\langle a, M_1 \rangle$$

$$\xRightarrow{\varepsilon} M_1 \approx? b, b \approx? (a b) \cdot M_1, a \#? a.\langle a, M_1 \rangle$$

Answers to the Quiz

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

$$\stackrel{\varepsilon}{\implies} b.\langle M_1, b \rangle \approx? b.\langle b, (a b) \cdot M_1 \rangle, a \# a.\langle a, M_1 \rangle$$

$$\stackrel{\varepsilon}{\implies} \langle M_1, b \rangle \approx? \langle b, (a b) \cdot M_1 \rangle, a \#? a.\langle a, M_1 \rangle$$

$$\stackrel{\varepsilon}{\implies} M_1 \approx? b, b \approx? (a b) \cdot M_1, a \#? a.\langle a, M_1 \rangle$$

$$\stackrel{[M_1 := b]}{\implies} b \approx? (a b) \cdot b, a \# a.\langle a, b \rangle$$

Answers to the Quiz

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

$$\xRightarrow{\varepsilon} b.\langle M_1, b \rangle \approx? b.\langle b, (a b) \cdot M_1 \rangle, a \# a.\langle a, M_1 \rangle$$

$$\xRightarrow{\varepsilon} \langle M_1, b \rangle \approx? \langle b, (a b) \cdot M_1 \rangle, a \#? a.\langle a, M_1 \rangle$$

$$\xRightarrow{\varepsilon} M_1 \approx? b, b \approx? (a b) \cdot M_1, a \#? a.\langle a, M_1 \rangle$$

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

\implies FAIL

Answers to the Quiz

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

$$\xRightarrow{\varepsilon} b.\langle M_1, b \rangle \approx? b.\langle b, (a b) \cdot M_1 \rangle, a \# a.\langle a, M_1 \rangle$$

$$\xRightarrow{\varepsilon} \langle M_1, b \rangle \approx? \langle b, (a b) \cdot M_1 \rangle, a \#? a.\langle a, M_1 \rangle$$

$$\xRightarrow{\varepsilon} M_1 \approx? b, b \approx? (a b) \cdot M_1, a \#? a.\langle a, M_1 \rangle$$

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

\implies 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

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

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

Answers to the Quiz

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

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

$$\xRightarrow{\varepsilon} \langle b, M_6 \rangle \approx? \langle b, (b a) \cdot M_7 \rangle, \quad b \# \langle a, M_7 \rangle$$

Answers to the Quiz

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

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

$$\xRightarrow{\varepsilon} \langle b, M_6 \rangle \approx? \langle b, (b a) \cdot M_7 \rangle, b \# \langle a, M_7 \rangle$$

$$\xRightarrow{\varepsilon} b \approx? b, M_6 \approx? (b a) \cdot M_7, b \# \langle a, M_7 \rangle$$

Answers to the Quiz

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

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

$$\xRightarrow{\varepsilon} \langle b, M_6 \rangle \approx? \langle b, (b a) \cdot M_7 \rangle, b \# \langle a, M_7 \rangle$$

$$\xRightarrow{\varepsilon} b \approx? b, M_6 \approx? (b a) \cdot M_7, b \# \langle a, M_7 \rangle$$

$$\xRightarrow{\varepsilon} M_6 \approx? (b a) \cdot M_7, b \# \langle a, M_7 \rangle$$

Answers to the Quiz

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

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

$$\xRightarrow{\varepsilon} \langle b, M_6 \rangle \approx? \langle b, (b a) \cdot M_7 \rangle, b \# \langle a, M_7 \rangle$$

$$\xRightarrow{\varepsilon} b \approx? b, M_6 \approx? (b a) \cdot M_7, b \# \langle a, M_7 \rangle$$

$$\xRightarrow{\varepsilon} M_6 \approx? (b a) \cdot M_7, b \# \langle a, M_7 \rangle$$

$$\xRightarrow{[M_6 := (b a) \cdot M_7]} b \#? \langle a, M_7 \rangle$$

Answers to the Quiz

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

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

$$\xRightarrow{\varepsilon} \langle b, M_6 \rangle \approx? \langle b, (b a) \cdot M_7 \rangle, b \# \langle a, M_7 \rangle$$

$$\xRightarrow{\varepsilon} b \approx? b, M_6 \approx? (b a) \cdot M_7, b \# \langle a, M_7 \rangle$$

$$\xRightarrow{\varepsilon} M_6 \approx? (b a) \cdot M_7, b \# \langle a, M_7 \rangle$$

$$\xRightarrow{[M_6 := (b a) \cdot M_7]} b \#? \langle a, M_7 \rangle$$

$$\xRightarrow{\emptyset} b \# a, b \# M_7$$

Answers to the Quiz

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

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

$$\xRightarrow{\varepsilon} \langle b, M_6 \rangle \approx? \langle b, (b a) \cdot M_7 \rangle, b \# \langle a, M_7 \rangle$$

$$\xRightarrow{\varepsilon} b \approx? b, M_6 \approx? (b a) \cdot M_7, b \# \langle a, M_7 \rangle$$

$$\xRightarrow{\varepsilon} M_6 \approx? (b a) \cdot M_7, b \# \langle a, M_7 \rangle$$

$$\xRightarrow{[M_6 := (b a) \cdot M_7]} b \#? \langle a, M_7 \rangle$$

$$\xRightarrow{\emptyset} b \# a, b \# M_7$$

$$\xRightarrow{\emptyset} b \# M_7$$

Answers to the Quiz

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

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

$$\xRightarrow{\varepsilon} \langle b, M_6 \rangle \approx? \langle b, (b a) \cdot M_7 \rangle, b \# \langle a, M_7 \rangle$$

$$\xRightarrow{\varepsilon} b \approx? b, M_6 \approx? (b a) \cdot M_7, b \# \langle a, M_7 \rangle$$

$$\xRightarrow{\varepsilon} M_6 \approx? (b a) \cdot M_7, b \# \langle a, M_7 \rangle$$

$$\xRightarrow{[M_6 := (b a) \cdot M_7]} b \#? \langle a, M_7 \rangle$$

$$\xRightarrow{\emptyset} b \# a, b \# M_7$$

$$\xRightarrow{\emptyset} b \# M_7$$

$$\xRightarrow{\{b \# M_7\}} \emptyset$$

Answers to the Quiz

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

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

$$\xRightarrow{\varepsilon} \langle b, M_6 \rangle \approx? \langle b, (b a).M_7 \rangle, b \# \langle a, M_7 \rangle$$

$$\xRightarrow{\varepsilon} b \approx? b, M_6 \approx? (b a).M_7, b \# \langle a, M_7 \rangle$$

$$\xRightarrow{\varepsilon} M_6 \approx? (b a).M_7, b \# \langle a, M_7 \rangle$$

$$\xRightarrow{[M_6 := (b a).M_7]} b \# \langle a, M_7 \rangle$$

$$\xRightarrow{\emptyset} b \# a, b \# M_7$$

$$\xRightarrow{\emptyset} b \# M_7$$

$$\xRightarrow{\{b \# M_7\}} \emptyset$$

$$\lambda a. \lambda b. (b M_6) =_{\alpha} \lambda a. \lambda a. (a M_7)$$

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

Conclusion

- used a permutation operation for renamings (has much nicer properties)
- it is a completely first-order language
- computed with freshness assumptions; this allowed us to define \approx so that substitution respects α -equivalence
- verified everything in Isabelle

- is equivalent to higher-order pattern unification
- it is polynomial (linear?)

Is it Useful?

applications to logic programming (with J. Cheney /
D. Friedman et al.)

$$\frac{x:A \in \Gamma}{\Gamma \vdash x:A} \quad \frac{\Gamma \vdash M:A \rightarrow B \quad \Gamma \vdash N:A}{\Gamma \vdash M N:B} \quad \frac{x:A, \Gamma \vdash M:B}{\Gamma \vdash \lambda x.M:A \rightarrow B}$$

```
type Gamma (var X) A :- member (pair X A) Gamma.  
type Gamma (app M N) B :- type Gamma M (arrow A B),  
                           type Gamma N A.  
type Gamma (lam x.M) (arrow A B) / x#Gamma :-  
                           type (pair x A)::Gamma M B.  
  
member A A::Tail.  
member A B::Tail :- member A Tail.
```

Thank you very much!

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 σ with

- $\nabla_2 \vdash \sigma(\nabla_1)$
 - $\nabla_2 \vdash a \# \sigma(X)$ holds for all $(a \# X) \in \nabla_1$
- $\nabla_2 \vdash \sigma_2 \approx \sigma \circ \sigma_1$
 - $\nabla_2 \vdash \sigma_2(X) \approx \sigma(\sigma_1(X))$ holds for all $X \in \text{dom}(\sigma_2) \cup \text{dom}(\sigma \circ \sigma_1)$