

# Lecture 8: nominal unification

# Sample $\alpha$ Prolog code

```
id : name_type. (* variables *)
tm : type.      (* lambda terms *)
var : id -> tm.
app : tm -> tm -> tm.
lam : id\tm -> tm.
pred subst (id\tm) tm tm.
(* "subst (a\X) Y Z" holds if Z is the result of capture-avoiding substitution
   of Y for all free occurrences of var a in X *)
subst (a\var a) Y Y.
subst (a\X) Y X :- a # X.
subst (a\app X X') Y (app Z Z') :- subst (a\X) Y Z, subst (a\X') Y Z'.
subst (a\lam(b\X)) Y (lam (b\Z)) :- subst (a\X) Y Z, b # Y.

?- subst (b\lam(a\var b)) (var a) X. (* search for X satisfying X =  $\lambda a.b[a/b]$  *)
Yes. X = lam(a'\var a) (* X is  $\lambda a'.a$ , not  $\lambda a.a$  *)
```

As for Prolog, search for solutions to queries involves resolution (try to unify query with head of each clause), but using **nominal unification**, which solves  $\alpha$ -equivalence and freshness constraints.

$\Sigma(S)$  = raw terms over  $\Sigma$  of sort  $S$

$$\frac{a \in \mathbb{A}}{a \in \Sigma(N)} \quad \frac{t \in \Sigma(S) \quad \text{op} : S \rightarrow D}{\text{op } t \in \Sigma(D)} \quad \frac{}{() \in \Sigma(1)}$$
$$\frac{t_1 \in \Sigma(S_1) \quad t_2 \in \Sigma(S_2)}{t_1, t_2 \in \Sigma(S_1, S_2)} \quad \frac{a \in \mathbb{A} \quad t \in \Sigma(S)}{a . t \in \Sigma(N.S)}$$

Each  $\Sigma(S)$  is a nominal set once equipped with the obvious **Perm**  $\mathbb{A}$ -action—any finite set of atoms containing all those occurring in  $t$  supports  $t \in \Sigma(S)$ .

# Alpha-equivalence

$$=_{\alpha} \subseteq \Sigma(S) \times \Sigma(S)$$

$$\frac{a \in A}{a =_{\alpha} a}$$

$$\frac{t =_{\alpha} t'}{\text{op } t =_{\alpha} \text{op } t'}$$

$$\frac{}{() =_{\alpha} ()}$$

$$\frac{t_1 =_{\alpha} t'_1 \quad t_2 =_{\alpha} t'_2}{t_1, t_2 =_{\alpha} t'_1, t'_2}$$

$$\frac{(a_1 \ a) \cdot t_1 =_{\alpha} (a_2 \ a) \cdot t_2 \quad a \# (a_1, t_1, a_2, t_2)}{a_1 \cdot t_1 =_{\alpha} a_2 \cdot t_2}$$

# Examples of unification 'mod $\alpha$ '

over the nominal algebraic signature  $\Sigma$  for  $\lambda$ -calculus:  
name-sort **Var**, data-sort **Term**, operations

$V : \text{Var} \rightarrow \text{Term}$

$A : \text{Term}, \text{Term} \rightarrow \text{Term}$

$L : \text{Var} . \text{Term} \rightarrow \text{Term}$

**Ex. 1:** does there exist a  $t \in \Sigma(\text{Term})$  with

$$L(a . L(b . A(t, Vb))) =_{\alpha} L(b . L(a . A(Va, t)))$$

(where  $a \neq b$ )?

**Ex. 2:** do there exist  $t_1, t_2 \in \Sigma(\text{Term})$  with

$$L(a . L(b . A(Vb, t_1))) =_{\alpha} L(a . L(a . A(Va, t_2)))$$

(where  $a \neq b$ )?

# Examples of unification 'mod $\alpha$ '

over the nominal algebraic signature  $\Sigma$  for  $\lambda$ -calculus:  
name-sort **Var**, data-sort **Term**, operations

$V : \text{Var} \rightarrow \text{Term}$

$A : \text{Term}, \text{Term} \rightarrow \text{Term}$

$L : \text{Var} . \text{Term} \rightarrow \text{Term}$

**Ex. 1:** does there exist a  $\lambda$ -term  $e$  with

$$\lambda a. \lambda b. e b = \lambda b. \lambda a. a e$$

(where  $a \neq b$ )?

**Ex. 2:** do there exist  $\lambda$ -terms  $e_1, e_2$  with

$$\lambda a. \lambda b. b e_1 = \lambda a. \lambda a. a e_2$$

(where  $a \neq b$ )?

**Ex. 1:** does there exist  $t \in \Sigma(\text{Term})$  with

$$L(a.L(b.A(t, \forall b))) =_{\alpha} L(b.L(a.A(\forall a, t)))$$

(where  $a \neq b$ )?

**Ex. 1:** does there exist  $t \in \Sigma(\text{Term})$  with

$$L(a.L(b.A(t, \forall b))) =_{\alpha} L(b.L(a.A(\forall a, t)))$$

(where  $a \neq b$ )?

$$L(b.A((a\ c) \cdot t, \forall b)) =_{\alpha} L(a.A(\forall a, (b\ c) \cdot t))$$

where  $c \# (a, b, t)$



**Ex. 1:** does there exist  $t \in \Sigma(\text{Term})$  with

$$L(a.L(b.A(t, \forall b))) =_{\alpha} L(b.L(a.A(\forall a, t)))$$

(where  $a \neq b$ )?

$$L(b.A((a\ c) \cdot t, \forall b)) =_{\alpha} L(a.A(\forall a, (b\ c) \cdot t))$$

where  $c \# (a, b, t)$

$$A((b\ d)(a\ c) \cdot t, \forall d) =_{\alpha} A(\forall d, (a\ d)(b\ c) \cdot t)$$

where  $d \# c, d, c \# (a, b, t)$

**Ex. 1:** does there exist  $t \in \Sigma(\text{Term})$  with

$$L(a.L(b.A(t, \forall b))) =_{\alpha} L(b.L(a.A(\forall a, t)))$$

(where  $a \neq b$ )?

$$L(b.A((a\ c) \cdot t, \forall b)) =_{\alpha} L(a.A(\forall a, (b\ c) \cdot t))$$

where  $c \# (a, b, t)$

$$A((b\ d)(a\ c) \cdot t, \forall d) =_{\alpha} A(\forall d, (a\ d)(b\ c) \cdot t)$$

where  $d \# c, d, c \# (a, b, t)$

$$(b\ d)(a\ c) \cdot t =_{\alpha} \forall d \text{ and } \forall d =_{\alpha} (a\ d)(b\ c) \cdot t$$

where  $d \# c, d, c \# (a, b, t)$

**Ex. 1:** does there exist  $t \in \Sigma(\text{Term})$  with

$$L(a.L(b.A(t, \forall b))) =_{\alpha} L(b.L(a.A(\forall a, t)))$$

(where  $a \neq b$ )?

$$L(b.A((a\ c) \cdot t, \forall b)) =_{\alpha} L(a.A(\forall a, (b\ c) \cdot t))$$

where  $c \# (a, b, t)$

$$A((b\ d)(a\ c) \cdot t, \forall d) =_{\alpha} A(\forall d, (a\ d)(b\ c) \cdot t)$$

where  $d \# c, d, c \# (a, b, t)$

$$(b\ d)(a\ c) \cdot t =_{\alpha} \forall d \text{ and } \forall d =_{\alpha} (a\ d)(b\ c) \cdot t$$

where  $d \# c, d, c \# (a, b, t)$

$$t =_{\alpha} \forall b \text{ and } \forall a =_{\alpha} t$$

where  $d \# c, d, c \# (a, b, t)$

**Ex. 1:** does there exist  $t \in \Sigma(\text{Term})$  with

$$L(a . L(b . A(t, \forall b))) =_{\alpha} L(b . L(a . A(\forall a, t)))$$

(where  $a \neq b$ )?

$$L(b . A((a c) \cdot t, \forall b)) =_{\alpha} L(a . A(\forall a, (b c) \cdot t))$$

where  $c \# (a, b, t)$

$$A((b d)(a c) \cdot t, \forall d) =_{\alpha} A(\forall d, (a d)(b c) \cdot t)$$

where  $d \# c, d, c \# (a, b, t)$

$$(b d)(a c) \cdot t =_{\alpha} \forall d \text{ and } \forall d =_{\alpha} (a d)(b c) \cdot t$$

where  $d \# c, d, c \# (a, b, t)$

$$t =_{\alpha} \forall b \text{ and } \forall a =_{\alpha} t$$

where  $d \# c, d, c \# (a, b, t)$

$$\forall b =_{\alpha} \forall a$$

$$b = a$$

contradicting  $a \neq b$  — so no such  $t$  can exist.

**Ex. 2:** do there exist  $t_1, t_2 \in \Sigma(\text{Term})$  with

$$L(a.L(b.A(Vb, t_1))) =_{\alpha} L(a.L(a.A(Va, t_2)))$$

(where  $a \neq b$ )?

**Ex. 2:** do there exist  $t_1, t_2 \in \Sigma(\text{Term})$  with

$$L(a . L(b . A(V b, t_1))) =_{\alpha} L(a . L(a . A(V a, t_2)))$$

(where  $a \neq b$ )?

$$L(b . A(V b, t_1)) =_{\alpha} L(a . A(V a, t_2))$$

**Ex. 2:** do there exist  $t_1, t_2 \in \Sigma(\text{Term})$  with

$$L(a . L(b . A(V b, t_1))) =_{\alpha} L(a . L(a . A(V a, t_2)))$$

(where  $a \neq b$ )?

$$L(b . A(V b, t_1)) =_{\alpha} L(a . A(V a, t_2))$$

$$A(V c, (b c) \cdot t_1) =_{\alpha} A(V c, (a c) \cdot t_2)$$

where  $c \# (a, b, t_1, t_2)$

**Ex. 2:** do there exist  $t_1, t_2 \in \Sigma(\text{Term})$  with  
 $L(a . L(b . A(V b, t_1))) =_{\alpha} L(a . L(a . A(V a, t_2)))$   
(where  $a \neq b$ )?

$$L(b . A(V b, t_1)) =_{\alpha} L(a . A(V a, t_2))$$

$$A(V c, (b c) \cdot t_1) =_{\alpha} A(V c, (a c) \cdot t_2)$$

where  $c \# (a, b, t_1, t_2)$

$$V c =_{\alpha} V c \text{ and } (b c) \cdot t_1 =_{\alpha} (a c) \cdot t_2$$

where  $c \# (a, b, t_1, t_2)$



**Ex. 2:** do there exist  $t_1, t_2 \in \Sigma(\text{Term})$  with  
 $L(a . L(b . A(V b, t_1))) =_{\alpha} L(a . L(a . A(V a, t_2)))$   
 (where  $a \neq b$ )?

$$L(b . A(V b, t_1)) =_{\alpha} L(a . A(V a, t_2))$$

$$A(V c, (b c) \cdot t_1) =_{\alpha} A(V c, (a c) \cdot t_2)$$

where  $c \# (a, b, t_1, t_2)$

$$V c =_{\alpha} V c \text{ and } (b c) \cdot t_1 =_{\alpha} (a c) \cdot t_2$$

where  $c \# (a, b, t_1, t_2)$

$$t_1 = (b c)(a c) \cdot t_2 [= (a b)(b c) \cdot t_2]$$

where  $c \# (a, b, (a b)(b c) \cdot t_2, t_2)$

**Ex. 2:** do there exist  $t_1, t_2 \in \Sigma(\text{Term})$  with  
 $L(a . L(b . A(V b, t_1))) =_{\alpha} L(a . L(a . A(V a, t_2)))$   
 (where  $a \neq b$ )?

$$L(b . A(V b, t_1)) =_{\alpha} L(a . A(V a, t_2))$$

$$A(V c, (b c) \cdot t_1) =_{\alpha} A(V c, (a c) \cdot t_2)$$

where  $c \# (a, b, t_1, t_2)$

$$V c =_{\alpha} V c \text{ and } (b c) \cdot t_1 =_{\alpha} (a c) \cdot t_2$$

where  $c \# (a, b, t_1, t_2)$

$$t_1 = (b c)(a c) \cdot t_2 [ = (a b)(b c) \cdot t_2 ]$$

where  $c \# (a, b, (a b)(b c) \cdot t_2, t_2)$

$$t_1 = (a b)(b c) \cdot t_2 [ = (a b) \cdot t_2 ]$$

where  $c \# (a, b, t_2)$  and  $b \# t_2$

**Ex. 2:** do there exist  $t_1, t_2 \in \Sigma(\text{Term})$  with  

$$L(a . L(b . A(V b, t_1))) =_{\alpha} L(a . L(a . A(V a, t_2)))$$
 (where  $a \neq b$ )?

$$L(b . A(V b, t_1)) =_{\alpha} L(a . A(V a, t_2))$$

$$A(V c, (b c) \cdot t_1) =_{\alpha} A(V c, (a c) \cdot t_2)$$

where  $c \# (a, b, t_1, t_2)$

$$V c =_{\alpha} V c \text{ and } (b c) \cdot t_1 =_{\alpha} (a c) \cdot t_2$$

where  $c \# (a, b, t_1, t_2)$

$$t_1 = (b c)(a c) \cdot t_2 [ = (a b)(b c) \cdot t_2 ]$$

where  $c \# (a, b, (a b)(b c) \cdot t_2, t_2)$

$$t_1 = (a b)(b c) \cdot t_2 [ = (a b) \cdot t_2 ]$$

where  $c \# (a, b, t_2)$  and  $b \# t_2$

$$t_1 = (a b) \cdot t_2, \text{ for any } t_2 \text{ with } b \# t_2$$

# Examples of unification 'mod $\alpha$ '

**Ex. 1:** does there exist a  $t \in \Sigma(\text{Term})$  with  
$$L(a.L(b.A(t, Vb))) =_{\alpha} L(b.L(a.A(Va, t)))$$
  
(where  $a \neq b$ )?

**Ex. 2:** do there exist  $t_1, t_2 \in \Sigma(\text{Term})$  with  
$$L(a.L(b.A(Vb, t_1))) =_{\alpha} L(a.L(a.A(Va, t_2)))$$
  
(where  $a \neq b$ )?

Can decide all such problems (over any nominal algebraic signature) using the **nominal unification algorithm** [Urban+AMP+Gabbay, TCS 323(2004)473–497]  $\triangleq$  **NOMU**.

First, need to extend the syntax of terms over a nominal signature with **variables**...

# Open nominal terms

$()$	unit	$a$	atomic names
$t, t'$	pairs	$a . t$	abstractions
$opt$	constructed	$\pi * X$	suspensions

$\pi \in \text{Perm } \mathbb{A}$

$X$  ranges over variables, standing for unknown terms

E.g.  $L(a . A(Vc, (a c) * X))$

# Equality & freshness


Equality of open terms is not just

$$t =_{\alpha} t' \quad \alpha\text{-equivalence}$$

# Equality & freshness

Equality is in general hypothetical

$\nabla \vdash t \approx t'$       hypothetical  $\alpha$ -equivalence



finite set of **freshness assumptions**,  $a \# X$ , each with intended meaning: 'atomic name  $a$  will not occur freely in any term substituted for  $X$ '

Intended meaning:

'any closing **substitution** (= replacement of variables by terms) satisfying  $\nabla$  makes  $t$  and  $t'$   $\alpha$ -equivalent'

# Equality & freshness

Equality is in general hypothetical

$\nabla \vdash t \approx t'$       hypothetical  $\alpha$ -equivalence

Examples of valid judgements:

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



# Equality & freshness

Also need freshness judgements

$$\nabla \vdash a \# t$$

Intended meaning:

'any closing substitution satisfying  $\nabla$  makes  $t$  not contain the atom  $a$  freely'

Examples of valid judgements:

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

# Rules for $\nabla \vdash t \approx t'$

Excerpt

(see NOMU, or NSB chapter 12, for full details)

# Rules for $\nabla \vdash t \approx t'$

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

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

# Rules for $\nabla \vdash t \approx t'$

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

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

$(a \ a') * t'$  is defined by recursion on the structure of  $t'$ , pushing the swap down through the structure, applying it to atoms and stopping with subterms like  $((a \ a') \circ \pi) * X$

## Rules for $\nabla \vdash t \approx t'$

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

E.g.

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

because

$$\begin{array}{ll} (a c)(a b) : & a \mapsto b \\ & b \mapsto c \\ & c \mapsto a \end{array} \quad \begin{array}{ll} (b c) : & a \mapsto a \\ & b \mapsto c \\ & c \mapsto b \end{array}$$

disagree at  $a$  and  $c$ .

# Rules for $\nabla \vdash a \not\approx t$

(Excerpt)

$$\frac{a \neq a'}{\nabla \vdash a \not\approx a'}$$

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

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

$$\frac{(\pi^{-1} a \not\approx X) \in \nabla}{\nabla \vdash a \not\approx \pi * X}$$

# Correctness

[NOMU, Proposition 2.16]

**Theorem.**  $\approx$  is an equivalence relation and agrees with  $=_\alpha$  on ground terms: if  $t$  and  $t'$  contain no variables then

$$\emptyset \vdash t \approx t' \text{ is valid iff } t =_\alpha t'.$$

Furthermore

$$\emptyset \vdash a \not\# t \text{ is valid iff } a \notin \text{fn}(t).$$

# Substitution

**Substitutions**  $\sigma$  are finite maps from variables to terms,  $[X_1 := t_1, \dots, X_n := t_n]$ .

**Applying a substitution to a term:**  $\sigma t$  = result of replacing variables in  $t$  with terms according to  $\sigma$ , carrying out any permutations of atomic names that are generated.

E.g. if  $\sigma = [X := A(Vb, Y)]$ , then

$$\begin{aligned}\sigma (L(a . (a b) * X)) &= L(a . (a b) * A(Vb, Y)) \\ &= L(a . A(Va, (a b) * Y))\end{aligned}$$



# Equational & freshness problems

An equational problem  $t \approx? t'$  is **solved** by

- ▶ a substitution  $\sigma$ , plus
- ▶ a set of freshness assumptions  $\nabla$

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

# Equational & freshness problems

An equational problem  $t \approx? t'$  is **solved** by

- ▶ a substitution  $\sigma$ , plus
- ▶ a set of freshness assumptions  $\nabla$

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

Solving equations may entail solving **freshness problems**.

E.g. assuming that  $a \neq a'$ , then  $L(a.t) \approx? L(a'.t')$  can only be solved if

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

can be solved.

# Equational & freshness problems

An equational problem  $t \approx t'$  is **solved** by

- ▶ a substitution  $\sigma$ , plus
- ▶ a set of freshness assumptions  $\nabla$

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

A freshness problem  $a \# t$  is **solved** by

- ▶ a substitution  $\sigma$ , plus
- ▶ a set of freshness assumptions  $\nabla$

so that  $\nabla \vdash a \# \sigma t$ .

# Existence of MGUs

**Theorem.** There is an algorithm which given any finite set  $P$  of equational and freshness problems (over any nominal algebraic signature), decides whether or not it has a solution  $(\sigma, \nabla)$ , and returns a **most general** one if it does.

straightforward definition, omitted



# Existence of MGUs

**Theorem.** There is an algorithm which given any finite set  $P$  of equational and freshness problems (over any nominal algebraic signature), decides whether or not it has a solution  $(\sigma, \nabla)$ , and returns a most general one if it does.

Algorithm first reduces all the equations to 'solved form' (creating a substitution), possibly generating extra freshness problems, and then solves all the freshness problems (easy).

(See [NOMU, Sect. 3].)

$$\{L(a.L(b.A(\forall b, X))) \approx L(a.L(a.A(\forall a, Y)))\}$$

$$\left(\xrightarrow{id}\right)^3 \{b.A(\forall b, X) \approx a.A(\forall a, Y)\}$$

$$\xrightarrow{id} \{A(\forall b, X) \approx A(\forall b, (b a) * Y), b \# A(\forall a, Y)\}$$

$$\left(\xrightarrow{id}\right)^3 \{X \approx (b a) * Y, b \# A(\forall a, Y)\}$$

$$[X := (b a) * Y] \xrightarrow{\quad} \{b \# A(\forall a, Y)\}$$

$$\xrightarrow{\emptyset} \{b \# \forall a, b \# Y\}$$

$$\left(\xrightarrow{\emptyset}\right)^2 \{b \# Y\}$$

$$\left(\xrightarrow{\{b \# Y\}}\right) \emptyset$$

$$\{L(a.L(b.A(Vb, X))) \approx L(a.L(a.A(Va, Y)))\}$$

$$\left(\frac{id}{\rightarrow}\right)^3 \{b.A(Vb, X) \approx a.A(Va, Y)\}$$

$$\xrightarrow{id} \{A(Vb, X) \approx A(Vb, (b a) * Y), b \# A(Va, Y)\}$$

$$\left(\frac{id}{\rightarrow}\right)^3 \{X \approx (b a) * Y, b \# A(Va, Y)\}$$

$$\left[\frac{X := (b a) * Y}{\rightarrow}\right] \{b \# A(Va, Y)\}$$

$$\xrightarrow{\emptyset} \{b \# Va, b \# Y\}$$

$$\left(\frac{\emptyset}{\rightarrow}\right)^2 \{b \# Y\}$$

$$\left[\frac{(b \# Y)}{\rightarrow}\right] \emptyset$$

most general solution =  $[X := (b a) * Y], \{(b \# Y)\}$

# Existence of MGUs

**Theorem.** There is an algorithm which given any finite set  $P$  of equational and freshness problems (over any nominal algebraic signature), decides whether or not it has a solution  $(\sigma, \nabla)$ , and returns a most general one if it does.

- ▶ Current best NOMU algorithm is quadratic [Levy & Villaret, Proc. RTA 2010].
- ▶ NOMU is (quadratically) inter-reducible with Dale Miller's **higher-order pattern unification**, which uses variables that depend on names  $X(a_1, \dots, a_n)$  rather than NOMU's variables that are fresh for names  $(\{a_1, \dots, a_n\} \# X)$ . (Higher-order patterns form a subset of Church's simply typed  $\lambda$ -calculus.)