Proposal for a second Examples Class (on 2nd 1/2 of Example Sheet):

THURSDAY 13 MARCH (6:00 - 17:00 SWOI

Lecture 8: nominal unification

Sample *a*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\Xar a) Y Y.
subst (a\X) Y X :- a # X.
subst (a\Xar y 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 α -equivalence and freshness constraints.

Examples of unification 'mod α '

over the nominal algebraic signature Σ for λ -calculus: name-sort Var, data-sort Term, operations V: Var \rightarrow Term

- A: Term, Term \rightarrow Term
- $\texttt{L}:\texttt{Var}.\texttt{Term}\to\texttt{Term}$

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

Ex. 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 $a \neq b$)? Alpha-equivalence = $_{\alpha} \subseteq \Sigma(S) \times \Sigma(S)$

$$\frac{a \in \mathbb{A}}{a =_{\alpha} a} \qquad \frac{t =_{\alpha} t'}{\operatorname{op} t =_{\alpha} \operatorname{op} t'} \qquad \overline{() =_{\alpha} ()}$$
$$\frac{t_1 =_{\alpha} t'_1 \qquad t_2 =_{\alpha} t'_2}{t_1, t_2 =_{\alpha} t'_1, t'_2}$$
$$\underbrace{(a_1 a) \cdot t_1 =_{\alpha} (a_2 a) \cdot t_2 \qquad a \# (a_1, t_1, a_2, t_2)}{a_1 \cdot t_1 =_{\alpha} a_2 \cdot t_2}$$

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

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

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

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

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)

> $t =_{\alpha} \forall b$ and $\forall a =_{\alpha} t$ where d # c, d, c # (a, b, t)

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

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

> $t =_{\alpha} \forall b$ and $\forall a =_{\alpha} t$ where d # c, d, c # (a, b, t)

$$V b =_{\alpha} V c$$

b = a

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

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

$$L(b \cdot A(\forall b, t_1)) =_{\alpha} L(a \cdot A(\forall a, t_2))$$
$$A(\forall c, (b c) \cdot t_1)) =_{\alpha} A(\forall c, (a c) \cdot t_2))$$
where $c \# (a, b, t_1, t_2)$

 $L(b \cdot A(\forall b, t_1)) =_{\alpha} L(a \cdot A(\forall a, t_2))$ $A(\forall c, (b c) \cdot t_1)) =_{\alpha} A(\forall c, (a c) \cdot t_2))$ where $c \# (a, b, t_1, t_2)$ $V c = \forall c \text{ and } (b c) \cdot t_2 = (a c) \cdot t_2$

$$\begin{array}{l} \forall c =_{\alpha} \forall c \text{ and } (b c) \cdot t_1 =_{\alpha} (a c) \cdot t_2 \\ \text{where } c \ \# (a, b, t_1, t_2) \end{array}$$

 $L(b \cdot A(\forall b, t_1)) =_{\alpha} L(a \cdot A(\forall a, t_2))$ $A(\forall c, (b c) \cdot t_1)) =_{\alpha} A(\forall c, (a c) \cdot t_2))$ where $c \# (a, b, t_1, t_2)$ $\forall c =_{\alpha} \forall c \text{ and } (b c) \cdot t_1 =_{\alpha} (a c) \cdot t_2$ 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 $c \# (a, b, (a b)(b c) \cdot t_2, t_2)$

> $L(b.A(Vb,t_1)) =_{\alpha} L(a.A(Va,t_2))$ $A(Vc, (bc) \cdot t_1)) =_{\alpha} A(Vc, (ac) \cdot t_2))$ where $c \# (a, b, t_1, t_2)$ $\forall c =_{\alpha} \forall c \text{ and } (b c) \cdot t_1 =_{\alpha} (a c) \cdot t_2$ 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 $b \# t_2$

> $L(b.A(Vb,t_1)) =_{\alpha} L(a.A(Va,t_2))$ $A(Vc, (bc) \cdot t_1)) =_{\alpha} A(Vc, (ac) \cdot t_2))$ where $c \# (a, b, t_1, t_2)$ $\forall c =_{\alpha} \forall c \text{ and } (b c) \cdot t_1 =_{\alpha} (a c) \cdot t_2$ 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 $b \# t_2$ $t_1 = (a b) \cdot t_2$, for any t_2 with $b \# t_2$

Examples of unification 'mod α '

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.

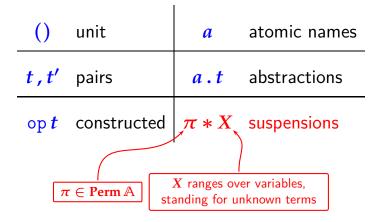
First, need to extend the synax of terms over a nominal signature with variables...

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

$\frac{a \in \mathbb{A}}{a \in \Sigma(\mathbb{N})}$	$\frac{t\in\Sigma(S)}{\operatorname{op} t\in\Sigma}$	-	$\overline{()\in\Sigma(1)}$
$\frac{t_1 \in \Sigma(\mathbf{S}_1)}{t_1, t_2 \in}$	$\frac{t_2 \in \boldsymbol{\Sigma}(\boldsymbol{\mathrm{S}}_2)}{\boldsymbol{\Sigma}(\boldsymbol{\mathrm{S}}_1,\boldsymbol{\mathrm{S}}_2)}$		$\frac{t \in \Sigma(S)}{\Sigma(N.S)}$

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

Open nominal terms

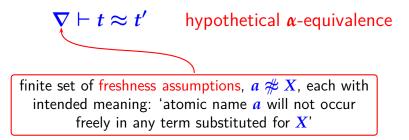


E.g. L(a.A(Vc, (ac) * X))

Equality of open terms is not just

$$t =_{\alpha} t' \qquad \alpha$$
-equivalence

Equality is in general hypothetical



Intended meaning:

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

Equality is in general hypothetical

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

Examples of valid judgements:

 $\{b \not \not \approx X\} \vdash a \, . \, X \approx b \, . \, ((a \ b) \ast X)$ $\{a \not \approx X, b \not \approx X\} \vdash a \, . \, X \approx b \, . \, X$

Also need freshness judgements

 $\nabla \vdash a \not\approx t$

Intended meaning:

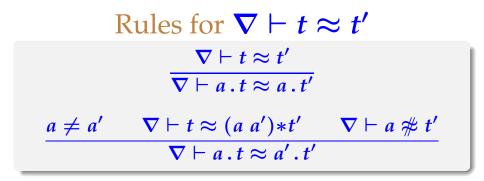
'any closing substitution satisfying ∇ makes t not contain the atom a freely'

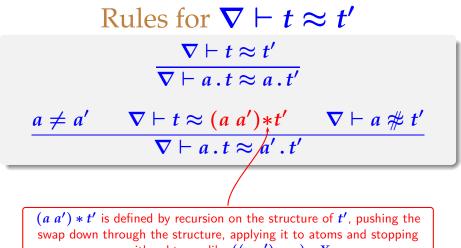
Examples of valid judgements:

 $\{b \not \# X\} \vdash b \not \# a . X \\ \{ \} \vdash a \not \# a . X$

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

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





with subterms like $((a \ a') \circ \pi) * X$

Rules for $\nabla \vdash t \approx t'$ $\frac{(a \notin X) \in \nabla \text{ for all } a \text{ with } \pi(a) \neq \pi'(a)}{\nabla \vdash \pi * X \approx \pi' * X}$

E.g.

$$\{a \not \# X, c \not \# X\} \vdash (a c)(a b) * X \approx (b c) * X$$

because

$$(a c)(a b): a \mapsto b$$
 $(b c): a \mapsto a$
 $b \mapsto c$ $b \mapsto c$
 $c \mapsto a$ $c \mapsto b$

disagree at *a* and *c*.

Rules for $\nabla \vdash a \not\cong t$ (Excerpt) $\frac{a \neq a'}{\nabla \vdash a \not\cong a'}$ $\frac{a \neq a' \quad \nabla \vdash a \not \ll t}{\nabla \vdash a \not \ll a' \cdot t}$ $\nabla \vdash a \not\approx a.t$ $(\pi^{-1} a \not\cong X) \in \mathbf{\nabla}$ $\nabla \vdash a \not\# \pi * X$

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

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

Furthermore

 $\emptyset \vdash a \not \approx t$ is valid iff $a \notin fn(t)$.

Substitution

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

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

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

 $\sigma \left(L(a.(a b) * X) \right) = L(a.(a b) * A(V b, Y))$ = L(a.A(V a, (a b) * Y))

Equational & freshness problems

An equational problem $t_? \approx_? t'$ is solved by

- a substitution σ , plus
- \blacktriangleright a set of freshness assumptions abla

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

Equational & freshness problems

An equational problem $t \ge t'$ is solved by

- a substitution σ , plus
- \blacktriangleright a set of freshness assumptions abla

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

Solving equations may entail solving freshness problems. E.g. assuming that $a \neq a'$, then $L(a \cdot t) \ge L(a' \cdot t')$ can only be solved if

$$t_{?} \approx_{?} (a a') * t'$$
 and $a \not\cong_{?} t'$

can be solved.

Equational & freshness problems

An equational problem $t \ge t'$ is solved by

- a substitution σ , plus
- \blacktriangleright a set of freshness assumptions abla
- so that $\nabla \vdash \sigma t \approx \sigma t'$.
- A freshness problem $a \not\approx_{?} t$ is solved by
 - a substitution σ , plus
 - \blacktriangleright a set of freshness assumptions abla

so that $\nabla \vdash a \not\approx \sigma t$.

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 (σ, ∇) , 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 (σ, ∇) , 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)))} $(\stackrel{ia}{\rightarrow})^3 \{b.A(Vb,X) \ge a.A(Va,Y)\}$ id {A($\forall b, X$) $_{?}\approx_{?}$ A($\forall b, (b, a) * Y$), $b \not\approx_{?}$ A($\forall a, Y$) $(\stackrel{id}{\rightarrow})^3 \{X_2 \approx_2 (b a) * Y, b \not\cong_2 \mathbb{A}(\mathbb{V} a, Y)\}$ $\overset{[X:=(b a)*Y]}{\rightarrow} \{b \not\cong_{2} A(\forall a, Y)\}$ $\stackrel{\varnothing}{\rightarrow} \{b \not \#_{?} \forall a, b \not \#_{?} Y\}$ $(\stackrel{\oslash}{\rightarrow})^2 \{b \not\approx_2 Y\}$ $\{(b \not \not \approx Y)\}$

{L(a.L(b.A($\forall b$, X))) $_{?}\approx_{?}$ L(a.L(a.A($\forall a$, Y)))} $(\stackrel{id}{\rightarrow})^3 \{b.A(\forall b, X) \ge a.A(\forall a, Y)\}$ id {A($\forall b, X$) $_{?} \approx_{?} A(\forall b, (b a) * Y), b #, A(\forall a, Y)$ } $(\stackrel{id}{\rightarrow})^3 \{X_2 \approx_2 (b a) * Y, b \not\cong_2 \mathbb{A}(\mathbb{V} a, Y)\}$ $\stackrel{[X:=(b a)*Y]}{\to} \{b \not\cong, A(\forall a, Y)\}$ $\stackrel{\varnothing}{\rightarrow} \{b \not \#_{?} \forall a, b \not \#_{?} Y\}$ $(\stackrel{\oslash}{\rightarrow})^2 \{b \not\approx_2 Y\}$ $\{(b \not \cong Y)\}$

most general solution = $[X := (b \ a) * Y], \{(b \not\cong Y)\}$

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 (σ, ∇) , 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(a₁,..., a_n) rather than NOMU's variables that are fresh for names ({a₁,..., a_n} [#] X). (Higher-order patterns form a subset of Church's simply typed λ-calculus.)

Other applications of nominal sets

Computational logic

- Higher-order logic: Urban & Berghofer's Nominal package for the interactive theorem-prover Isabelle/HOL.
- Equational logic: rewriting for nominal terms [Fernandez+Gabbay+Calves+···]

Other applications of nominal sets

Computational logic

- Higher-order logic: Urban & Berghofer's Nominal package for the interactive theorem-prover Isabelle/HOL.
- Equational logic: rewriting for nominal terms [Fernandez+Gabbay+Calves+•••]

Automata theory & verification

- HD-automata [Montanari el al]
- fresh-register automata [Tzevelekos]
- orbit-finite computation theory [Bojańczyk et al]

Other applications of nominal sets

► Homotopy Type Theory (HoTT)

Cubical sets [Bezem-Coquand-Huber] model of Voevodsky's axiom of univalence makes use of nominal sets equipped with an operation of substitution $x \mapsto x(i|a)$ where $i \in \{0, 1\}$.

- names are names of directions (cartesian axes)
 - (so e.g., if an object has support $\{a, b, c\}$ it is 3-dimensional)
- freshness (a # x) = degeneracy (x(i|a) = x)
- identity types are modelled by name-abstraction: $\langle a \rangle x$ is a proof that x(0/a) is equal to x(1/a).

HoTT and univalence is about (computable) *mathematical foundations* (a topic no longer very popular with mathematicians). That's where the mathematics of nominal sets came from...

Impact can take a long time

The mathematics behind nominal sets goes back a long way...



Abraham Fraenkel, Der Begriff "definit" und die Unabhängigkeit des Auswahlsaxioms, Sitzungsberichte der Preussischen Akademie der Wissenschaften, Physikalisch-mathematische Klasse (1922), 253–257.



Andrzej Mostowski, Uber die Unabhängigkeit des Wohlordnungssatzes vom Ordnungsprinzip, Fundamenta Mathematicae 32 (1939), 201–252.

Impact can take a long time

The mathematics behind nominal sets goes back a long way...

 \ldots and it's still too early to tell what will be the impact of the applications of it to CS developed over the last 15 years.