

Nominal Unification with Triangular Substitutions

Ramana Kumar Michael Norrish

11 July 2010



Outline

Triangular Substitutions

Nominal Terms

Nominal Unification

- Phase 1: Equality Constraints

- Phase 2: Freshness Constraints

- Example

Termination

Correctness

- Soundness

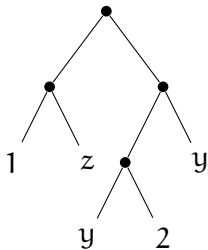
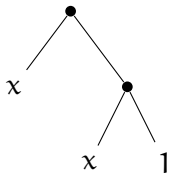
- Completeness

- Generality

Unifying Terms to Build a Triangular Substitution

Do these terms unify?

$$\langle x, \langle x, 1 \rangle \rangle =? \langle \langle 1, z \rangle, \langle \langle y, 2 \rangle, y \rangle \rangle$$



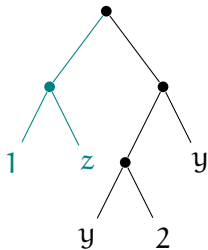
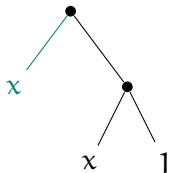
Accumulator:

\emptyset

Unifying Terms to Build a Triangular Substitution

Do these terms unify?

$x =? \langle 1, z \rangle$



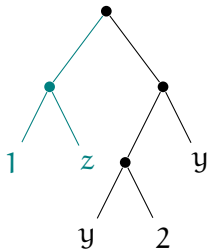
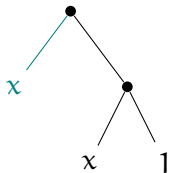
Accumulator:

\emptyset

Unifying Terms to Build a Triangular Substitution

Do these terms unify?

$x =? \langle 1, z \rangle$



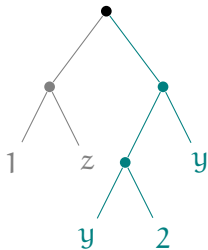
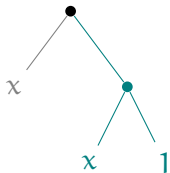
Accumulator:

$x \mapsto \langle 1, z \rangle, \emptyset$

Unifying Terms to Build a Triangular Substitution

Do these terms unify?

$$\langle x, 1 \rangle =? \langle \langle y, 2 \rangle, y \rangle$$



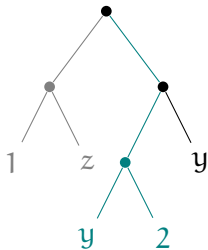
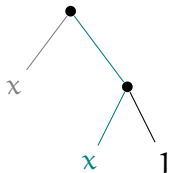
Accumulator:

$$x \mapsto \langle 1, z \rangle, \emptyset$$

Unifying Terms to Build a Triangular Substitution

Do these terms unify?

$x =? \langle y, 2 \rangle$



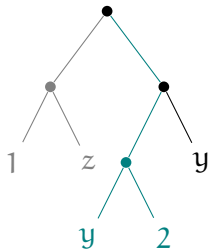
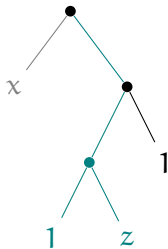
Accumulator:

$x \mapsto \langle 1, z \rangle, \emptyset$

Unifying Terms to Build a Triangular Substitution

Do these terms unify?

$$\langle 1, z \rangle =? \langle y, 2 \rangle$$



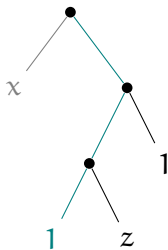
Accumulator:

$$x \mapsto \langle 1, z \rangle, \emptyset$$

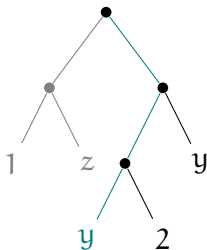
Unifying Terms to Build a Triangular Substitution

Do these terms unify?

$$\langle 1, z \rangle =? \langle y, 2 \rangle$$



Accumulator:

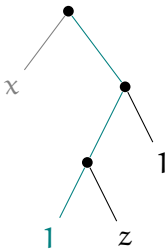


$x \mapsto \langle 1, z \rangle, \emptyset$

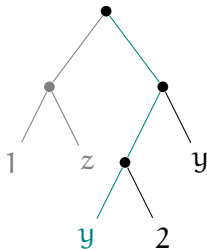
Unifying Terms to Build a Triangular Substitution

Do these terms unify?

$1 \stackrel{?}{=} y$



Accumulator:

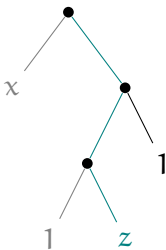


$y \mapsto 1, x \mapsto \langle 1, z \rangle, \emptyset$

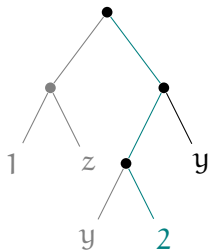
Unifying Terms to Build a Triangular Substitution

Do these terms unify?

$z \stackrel{?}{=} 2$



Accumulator:

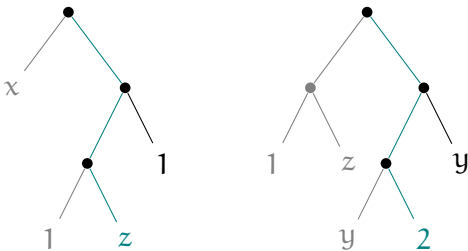


$y \mapsto 1, x \mapsto \langle 1, z \rangle, \emptyset$

Unifying Terms to Build a Triangular Substitution

Do these terms unify?

$z \stackrel{?}{=} 2$

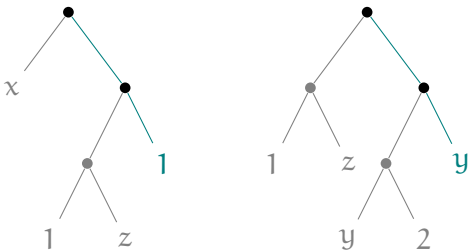


Accumulator: $z \mapsto 2, y \mapsto 1, x \mapsto \langle 1, z \rangle, \emptyset$

Unifying Terms to Build a Triangular Substitution

Do these terms unify?

$1 \stackrel{?}{=} y$

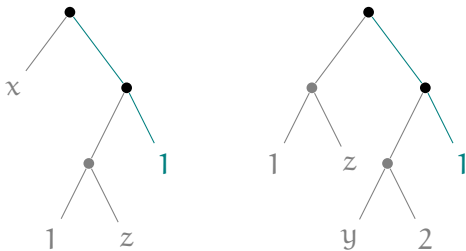


Accumulator: $z \mapsto 2, y \mapsto 1, x \mapsto \langle 1, z \rangle, \emptyset$

Unifying Terms to Build a Triangular Substitution

Do these terms unify?

$1 \stackrel{?}{=} 1$

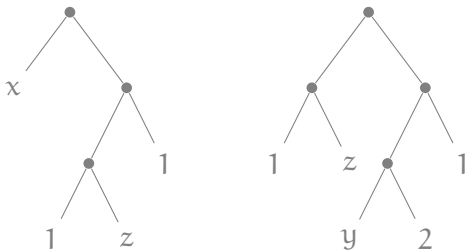


Accumulator: $z \mapsto 2, y \mapsto 1, x \mapsto \langle 1, z \rangle, \emptyset$

Unifying Terms to Build a Triangular Substitution

Do these terms unify?

Yes.



Accumulator: $z \mapsto 2, y \mapsto 1, x \mapsto \langle 1, z \rangle, \emptyset$

Triangular Substitutions

Unifying $\langle\langle x, 2 \rangle, x\rangle$ and $\langle y, \langle 1, z \rangle\rangle$

- Triangular: $x \mapsto \langle 1, z \rangle, y \mapsto \langle x, 2 \rangle$
- Idempotent: $x \mapsto \langle 1, z \rangle, y \mapsto \langle\langle 1, z \rangle, 2\rangle$

Composing with the result of unifying z and 3

- Triangular: $z \mapsto 3, x \mapsto \langle 1, z \rangle, y \mapsto \langle x, 2 \rangle$
- Idempotent: $x \mapsto \langle 1, 3 \rangle, y \mapsto \langle\langle 1, 3 \rangle, 2\rangle, z \mapsto 3$

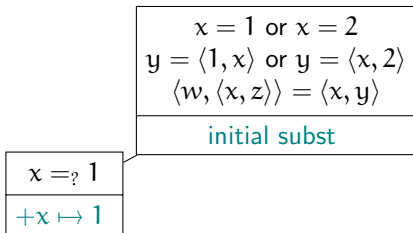
Advantages:

persistence, memory sharing, no copying, can be much smaller.

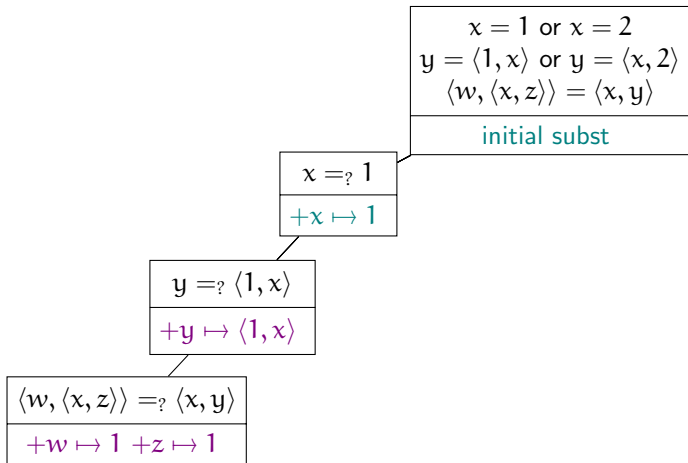
Backtracking with Triangular Substitutions

$x = 1 \text{ or } x = 2$ $y = \langle 1, x \rangle \text{ or } y = \langle x, 2 \rangle$ $\langle w, \langle x, z \rangle \rangle = \langle x, y \rangle$
initial subst

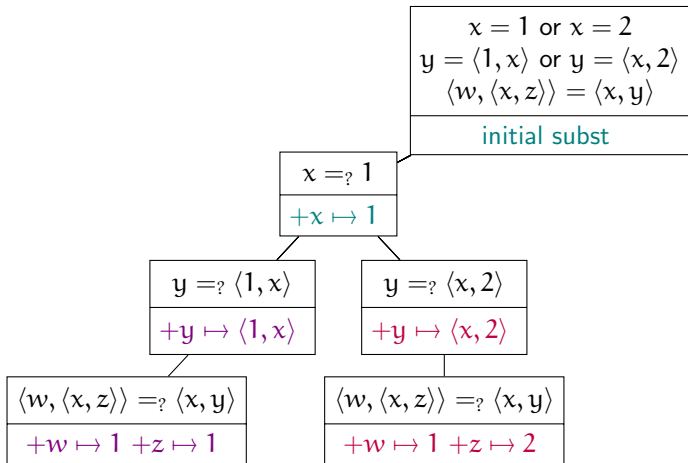
Backtracking with Triangular Substitutions



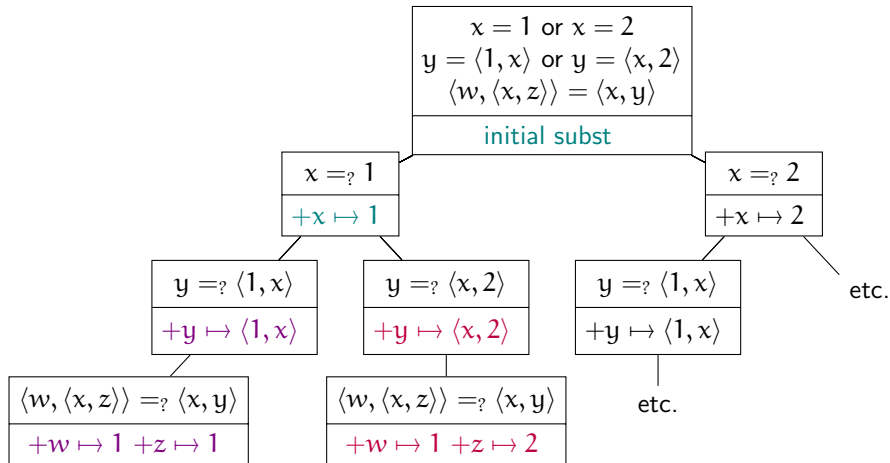
Backtracking with Triangular Substitutions



Backtracking with Triangular Substitutions



Backtracking with Triangular Substitutions



Applying Triangular Substitutions

Let $\sigma = \{z \mapsto 3, y \mapsto \langle z, 1 \rangle, x \mapsto y\}$, $t = x$.

One-step Homomorphic lift of map from variables to terms. (Unused. Result would be y .)

Full Repeated one-step. $\sigma \triangleleft t = \langle 3, 1 \rangle$.

Walk Repeats only on top-level variables.
 $\text{walk } \sigma t = \langle z, 1 \rangle$.

Well-formedness

Applying $\{z \mapsto x, y \mapsto z, x \mapsto y\}$ to x loops indefinitely.

Disallow substitutions with loops:

- Non-looping substitutions are *well-formed* (wfs).
- $\{z \mapsto \langle x, 3 \rangle, y \mapsto z, x \mapsto y\}$ is not well-formed.
- $\vdash \text{wfs } \sigma \iff \exists n. \sigma^n$ is idempotent.

Outline

Triangular Substitutions

Nominal Terms

Nominal Unification

Phase 1: Equality Constraints

Phase 2: Freshness Constraints

Example

Termination

Correctness

Soundness

Completeness

Generality

Nominal Terms

Terms t form an **algebraic** type of five constructors:

Nom Object-level names/atoms $a_1, a_2 \dots$

Const Arbitrary non-name data $c_1, c_2 \dots$

Susp A suspension: a meta-variable coupled with a permutation (over noms). $\pi_1 \cdot V_1, \pi_2 \cdot V_2 \dots$

Tie A binding: a name (bound variable) coupled with a term (the body). $[a_1]t_1, [a_2]t_2 \dots$

Pair A pair of terms. $\langle t_{11}, t_{12} \rangle, \langle t_{21}, t_{22} \rangle \dots$

We **don't** quotient by α -equivalence on ties.

Freshness Constraints

A pair $(a \# V)$

meaning: V should not be bound to a term with a free.

Example

Q. $[a_1]V_1 =? [a_2]V_2$

A. Yes: $\sigma = \{V_1 \mapsto (a_1 a_2) \cdot V_2\}$, $\nabla = \{(a_1 \# V_2)\}$.

V_1	V_2	Verdict
a_3	a_3	✓
a_1	a_2	✓
a_2	a_1	✗

Freshness and Equivalence Judgements

Define $\nabla \vdash a \# t$ and $\nabla \vdash t_1 \approx t_2$.

Interesting rules

$$\frac{}{\nabla \vdash a \# [a]t} \quad \frac{(\pi^{-1}(a) \# V) \in \nabla}{\nabla \vdash a \# \pi \cdot V}$$

$$\frac{\nabla \vdash t_1 \approx (a_1 a_2) \cdot t_2 \quad \nabla \vdash a_1 \# t_2}{\nabla \vdash [a_1]t_1 \approx [a_2]t_2}$$

Outline

Triangular Substitutions

Nominal Terms

Nominal Unification

Phase 1: Equality Constraints

Phase 2: Freshness Constraints

Example

Termination

Correctness

Soundness

Completeness

Generality

Nominal Unification

Nominal unification works in two phases.

The first phase is similar to first-order unification.

The second phase checks freshness constraints.

Unification Algorithm 1

```
unify1 ( $\sigma, \nabla$ )  $t_1$   $t_2$  =  
  case (walk  $\sigma$   $t_1$ , walk  $\sigma$   $t_2$ ) of  
    ( $a_1$ ,  $a_2$ )  $\rightarrow$  if  $a_1 = a_2$  then ( $\sigma, \nabla$ ) else fail  
    ( $c_1$ ,  $c_2$ )  $\rightarrow$  if  $c_1 = c_2$  then ( $\sigma, \nabla$ ) else fail  
    ( $\pi_1 \cdot V_1$ ,  $\pi_2 \cdot V_2$ )  $\rightarrow$   
      if  $V_1 = V_2$  then  
        unify_eq_vars (dis_set  $\pi_1$   $\pi_2$ )  $V_1$  ( $\sigma, \nabla$ )  
      else  
        add_bdg  $\pi_1$   $V_1$  ( $\pi_2 \cdot V_2$ ) ( $\sigma, \nabla$ )  
    ( $\pi_1 \cdot V_1$ ,  $t_2$ )  $\rightarrow$  add_bdg  $\pi_1$   $V_1$   $t_2$  ( $\sigma, \nabla$ )  
    ( $t_1$ ,  $\pi_2 \cdot V_2$ )  $\rightarrow$  add_bdg  $\pi_2$   $V_2$   $t_1$  ( $\sigma, \nabla$ )  
    ...
```

(add_bdg does an *occurs check*.)

(unify_eq_vars adds freshness constraints.)

Unification Algorithm 2

```
unify1 ( $\sigma, \nabla$ )  $t_1 t_2 =$   
  case (walk  $\sigma t_1$ , walk  $\sigma t_2$ ) of  
    ...  
    ( $[a_1]t_1$ ,  $[a_2]t_2$ )  $\rightarrow$   
      if  $a_1 = a_2$  then unify1 ( $\sigma, \nabla$ )  $t_1 t_2$   
      else do  
         $\nabla' \leftarrow$  term_fcs  $a_1$  ( $\sigma \triangleleft t_2$ );  
        unify1 ( $\sigma, \nabla \cup \nabla'$ )  $t_1 ((a_1 a_2) \cdot t_2)$   
  
    ( $\langle t_{11}, t_{12} \rangle$ ,  $\langle t_{21}, t_{22} \rangle$ )  $\rightarrow$  do  
      ( $\sigma', \nabla'$ )  $\leftarrow$  unify1 ( $\sigma, \nabla$ )  $t_{11} t_{21}$ ;  
      unify1 ( $\sigma', \nabla'$ )  $t_{12} t_{22}$   
  
    _  $\rightarrow$  fail
```

(term_fcs $a t$ generates the minimal set of freshness constraints to make a fresh for t .)

Verifying Freshness Constraints

`unify2` $\sigma \nabla$ generates a minimal ∇' , such that for each $(a \# V)$ in ∇ we have

$$\nabla' \vdash a \# (\sigma \triangleleft V)$$

`unify2` might:

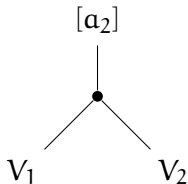
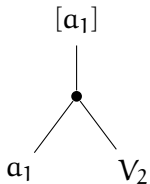
- add constraints $(\sigma = \{V \mapsto \langle V_2, V_3 \rangle\}, \nabla = (a \# V))$
- remove constraints $(\sigma = \{V \mapsto a_1\}, \nabla = \{(a_2 \# V)\})$
- fail $(\sigma = \{V \mapsto a_1\}, \nabla = \{(a_1 \# V)\})$

Putting the Two Phases Together

```
unify ( $\sigma, \nabla$ )  $t_1 t_2 = \mathbf{do}$   
  ( $\sigma', \nabla'$ )  $\leftarrow$  unify1 ( $\sigma, \nabla$ )  $t_1 t_2$ ;  
   $\nabla'' \leftarrow$  unify2  $\sigma' \nabla'$ ;  
  ( $\sigma', \nabla''$ )
```

Unification Example

$$[a_1]\langle a_1, V_2 \rangle =? [a_2]\langle V_1, V_2 \rangle$$



Substitution σ :

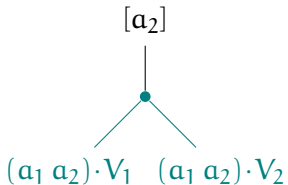
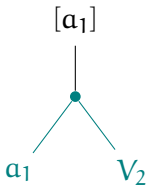
\emptyset

Freshnesses ∇ :

\emptyset

Unification Example

$$\langle \mathbf{a}_1, \mathbf{V}_2 \rangle =? (\mathbf{a}_1 \ \mathbf{a}_2) \cdot \langle \mathbf{V}_1, \mathbf{V}_2 \rangle$$



Substitution σ :

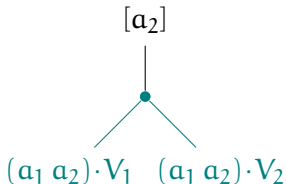
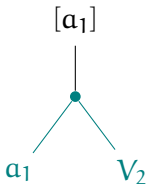
\emptyset

Freshnesses ∇ :

\emptyset

Unification Example

$$\langle a_1, V_2 \rangle =? (a_1 a_2) \cdot \langle V_1, V_2 \rangle$$



Substitution σ :

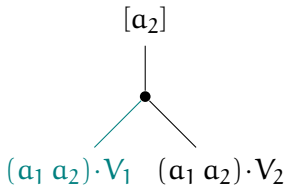
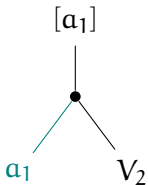
Freshnesses ∇ :

\emptyset

$(a_1 \# V_1), (a_1 \# V_2), \emptyset$

Unification Example

$$a_1 =? (a_1 a_2) \cdot V_1$$



Substitution σ :

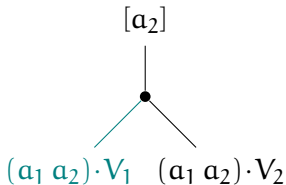
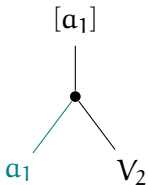
Freshnesses ∇ :

\emptyset

$(a_1 \# V_1), (a_1 \# V_2), \emptyset$

Unification Example

$$a_1 =? (a_1 a_2) \cdot V_1$$



Substitution σ :

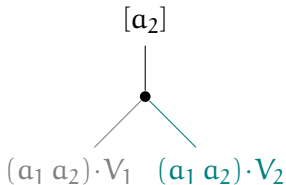
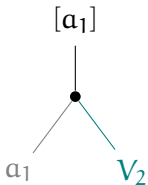
Freshnesses ∇ :

$V_1 \mapsto a_2, \emptyset$

$(a_1 \# V_1), (a_1 \# V_2), \emptyset$

Unification Example

$$V_2 =? (a_1 a_2) \cdot V_2$$



Substitution σ :

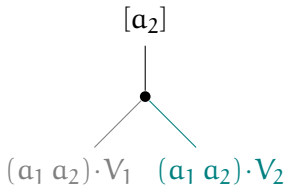
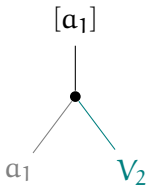
Freshnesses ∇ :

$V_1 \mapsto a_2, \emptyset$

$(a_1 \# V_1), (a_1 \# V_2), \emptyset$

Unification Example

$$V_2 =? (a_1 a_2) \cdot V_2$$

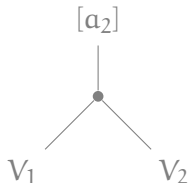
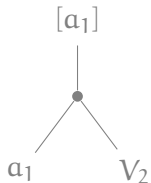


Substitution σ : $V_1 \mapsto a_2, \emptyset$

Freshnesses ∇ : $(a_2 \# V_2), (a_1 \# V_1), (a_1 \# V_2), \emptyset$

Unification Example

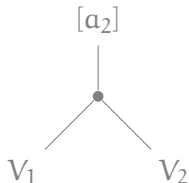
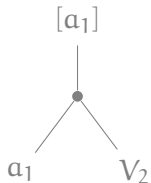
Verify $(a_2 \# (\sigma \triangleleft V_2))$, $(a_1 \# (\sigma \triangleleft V_1))$, $(a_1 \# (\sigma \triangleleft V_2))$.



Substitution σ : $V_1 \mapsto a_2, \emptyset$
Freshnesses ∇ : $(a_2 \# V_2), (a_1 \# V_1), (a_1 \# V_2), \emptyset$

Unification Example

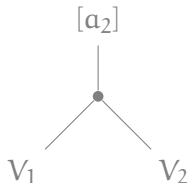
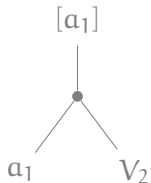
Verify $(a_2 \# V_2)$, $(a_1 \# a_2)$, $(a_1 \# V_2)$.



Substitution σ : $V_1 \mapsto a_2, \emptyset$
Freshnesses ∇ : $(a_2 \# V_2), (a_1 \# V_1), (a_1 \# V_2), \emptyset$

Unification Example

Verify $(a_2 \# V_2)$, $(a_1 \# a_2)$, $(a_1 \# V_2)$.



Substitution σ :

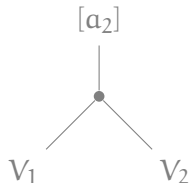
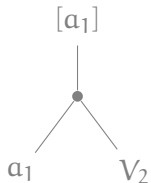
Freshnesses ∇ : $(a_2 \# V_2)$,

$V_1 \mapsto a_2, \emptyset$

$(a_1 \# V_2), \emptyset$

Unification Example

Done.



Substitution σ :

Freshnesses ∇ : $(a_2 \# V_2),$

$V_1 \mapsto a_2, \emptyset$

$(a_1 \# V_2), \emptyset$

Outline

Triangular Substitutions

Nominal Terms

Nominal Unification

- Phase 1: Equality Constraints

- Phase 2: Freshness Constraints

- Example

Termination

Correctness

- Soundness

- Completeness

- Generality

Nested Recursion Under walk

```
unify1 ( $\sigma, \nabla$ )  $t_1$   $t_2$  =  
  case (walk  $\sigma$   $t_1$ , walk  $\sigma$   $t_2$ ) of  
    ...  
    ( $\langle t_{11}, t_{12} \rangle$ ,  $\langle t_{21}, t_{22} \rangle$ )  $\rightarrow$   
      do ( $\sigma', \nabla'$ )  $\leftarrow$  unify1 ( $\sigma, \nabla$ )  $t_{11}$   $t_{21}$ ;  
        unify1 ( $\sigma', \nabla'$ )  $t_{12}$   $t_{22}$   
    ...
```

Need a well-founded relation R such that:

- $R(((\sigma, \nabla), t_{11}, t_{21}), ((\sigma, \nabla), t_1, t_2))$ and
- $R(((\sigma', \nabla'), t_{12}, t_{22}), ((\sigma, \nabla), t_1, t_2))$ (ugh!)
- where $\text{walk } \sigma t_1 = \langle t_{11}, t_{12} \rangle$ and $\text{walk } \sigma t_2 = \langle t_{21}, t_{22} \rangle$.

Termination Relation for unify1

Our \mathcal{R} , which ignores the freshness environment (∇):

$$\begin{aligned} \text{UTR}((\sigma', t_{12}, t_{22}), (\sigma, t_1, t_2)) &\iff \\ &\text{wfs } \sigma' \quad (\implies \text{wfs } \sigma) \\ \wedge \quad \sigma &\sqsubseteq \sigma' \\ \wedge \quad \mathcal{V}(\sigma', t_{12}, t_{22}) &\subseteq \mathcal{V}(\sigma, t_1, t_2) \\ \wedge \quad |\sigma' \triangleleft t_{12}| &< |\sigma' \triangleleft t_1| \end{aligned}$$

This is **not** a lexicographic combination.

Termination 1: UTR is Well-Founded

Our \mathcal{R} , which ignores the freshness environment (∇):

$$\begin{aligned} \text{UTR}((\sigma', t_{12}, t_{22}), (\sigma, t_1, t_2)) &\iff \\ &\text{wfs } \sigma' \quad (\implies \text{wfs } \sigma) \\ \wedge \quad \sigma &\sqsubseteq \sigma' \\ \wedge \quad \mathcal{V}(\sigma', t_{12}, t_{22}) &\subseteq \mathcal{V}(\sigma, t_1, t_2) \\ \wedge \quad |\sigma' \triangleleft t_{12}| &< |\sigma' \triangleleft t_1| \end{aligned}$$

UTR sequence: $\dots, (\sigma'', t_{122}, t_{222}), (\sigma', t_{12}, t_{22}), (\sigma, t_1, t_2)$

- $\mathcal{V}(\sigma, t_1, t_2)$ reaches a fixpoint: finite subsets.
- σ' reaches a fixpoint: new bindings drawn from fixed set.
- Decreasing sequence of $|\sigma' \triangleleft t_1|$ must stop.

Termination 2: UTR Between Recursive Calls

Need to show:

- $\text{UTR}((\sigma, t_{11}, t_{21}), (\sigma, t_1, t_2))$ and
- $\text{UTR}((\text{????}, t_{12}, t_{22}), (\sigma, t_1, t_2))$
- where $\text{walk } \sigma t_1 = \langle t_{11}, t_{12} \rangle$ and $\text{walk } \sigma t_2 = \langle t_{21}, t_{22} \rangle$.

Difficulty:

- The ????.

Solution:

- See the paper!

Outline

Triangular Substitutions

Nominal Terms

Nominal Unification

Phase 1: Equality Constraints

Phase 2: Freshness Constraints

Example

Termination

Correctness

Soundness

Completeness

Generality

Correctness

If `unify` produces a result, it is a nominal unifier (soundness).

If the arguments are unifiable, `unify` succeeds (completeness).

If `unify` produces a result, it is most-general (generality).

Correctness in an Accumulator-Passing World

What does “soundness” (say) mean for

$$\text{unify } (\sigma, \nabla) t_1 t_2 = (\sigma', \nabla')$$

For the substitutions:

$$\text{unify } (\sigma, \nabla) t_1 t_2 \simeq \text{unify } (\emptyset, \nabla) (\sigma \triangleleft t_1) (\sigma \triangleleft t_2)$$

The freshness environment (∇) imposes additional (potentially unsatisfiable) requirements.

Soundness

$$\vdash \text{unify } (\sigma, \nabla) t_1 t_2 = (\sigma', \nabla') \implies \\ \nabla' \vdash \sigma' \triangleleft (\sigma \triangleleft t_1) \approx \sigma' \triangleleft (\sigma \triangleleft t_2)$$

Alternatively: we know that $\sigma \sqsubseteq \sigma'$, so conclusion can be

$$\nabla' \vdash \sigma' \triangleleft t_1 \approx \sigma' \triangleleft t_2$$

Completeness

$$\vdash \nabla_u \vdash \sigma_u \triangleleft (\sigma \triangleleft t_1) \approx \sigma_u \triangleleft (\sigma \triangleleft t_2) \implies \\ \exists \sigma'. \forall \nabla.$$

$$\nabla \text{ consistent with } \sigma' \implies$$

$$\exists \nabla'. \text{unify} (\sigma, \nabla) t_1 t_2 = (\sigma', \nabla')$$

Can simplify, by taking (universally quantified) $\nabla = \emptyset$, concluding:

$$\exists \sigma' \nabla'. \text{unify} (\sigma, \emptyset) t_1 t_2 = (\sigma', \nabla')$$

Or even:

$$\exists \sigma' \nabla'. \text{unify} (\emptyset, \emptyset) (\sigma \triangleleft t_1) (\sigma \triangleleft t_2) = (\sigma', \nabla')$$

Generality

Generality of unify's returned substitution:

$$\begin{aligned} &\vdash \text{unify}(\sigma, \nabla) t_1 t_2 = (\sigma', \nabla') \wedge \\ &\quad \nabla_u \vdash \sigma_u \triangleleft (\sigma \triangleleft t_1) \approx \sigma_u \triangleleft (\sigma \triangleleft t_2) \implies \\ &\quad \exists \sigma_w. \forall t. \\ &\quad \quad \nabla_u \vdash \sigma_w \triangleleft (\sigma' \triangleleft (\sigma \triangleleft t)) \approx \sigma_u \triangleleft (\sigma \triangleleft t) \end{aligned}$$

The witness σ_w can be σ_u .

We also know that $\sigma' \triangleleft (\sigma \triangleleft t) = \sigma' \triangleleft t$, since $\sigma \sqsubseteq \sigma'$.

Generality

Generality of unify's returned freshness environment:

$$\begin{aligned} &\vdash \text{unify } (\sigma, \nabla) t_1 t_2 = (\sigma', \nabla') \wedge \\ &\quad \nabla_u \vdash \sigma_u \triangleleft (\sigma \triangleleft t_1) \approx \sigma_u \triangleleft (\sigma \triangleleft t_2) \implies \\ &\quad \forall (\mathbf{a} \# V) \in \nabla'. \\ &\quad (\mathbf{a} \# V) \text{ is due to something in } \nabla, \text{ or} \\ &\quad \nabla_u \vdash \mathbf{a} \# (\sigma_u \triangleleft V) \end{aligned}$$

(“due to something” is long when captured formally.)

Conclusion

Formal treatment of triangular substitutions

- well-formedness condition, different forms of application.

Nominal unification in accumulator-passing style

- implementable, efficient.

Termination

- unusual termination relation, same as first-order case.

Correctness

- soundness, completeness, generality, for accumulator-style.