Formalizing Nominal Unification

Washington L. R. de Carvalho II

Mauricio Ayala-Rincón   Maribel Fernández

Grupo de Teoria da Computação - GTC

Departmento de Ciência da Computação - CIC

Universidade de Brasília - UnB

5th February, 2015

OUTLINE

Motivation

Problem Definition

Work Done so far

Conclusion and future work

Motivation

- The Nominal Logic applications seams to be closer of the *pencil and paper* experience than the nameless approaches (like *de Bruijn* indexes)
- There has been a lot of effort for building nominal logic frameworks:
  - *logic programming*: α-Karen [2], α-Prolog [7, 16], αML [9], Ocaml [4], CαML [12], Fresh-Ocaml [13], FreshML [14]
  - *theorem proving*: Coq [1], Maude [4], HOL4 [8], Isabelle/HOL [15, 17, 18] and PVS (*in progress*)
- We are presenting a, also *in progress*, contribution that is a formalization (in Coq) of the nominal unification algorithm.
- It had already done just in Maude, HOL4 and Isabelle/HOL

Atoms, variables, function symbols, swappings, permutations and freshness contexts

- $\mathbf{\Sigma} := \begin{cases} \mathcal{A}: & a, b, c, ... \\ \mathcal{X}: & X, Y, Z, ... \\ \mathcal{S}: & f, g, h, ... \end{cases}$
- $\mathbf{\Pi} := \{$the set of *finite permutations* over $\mathcal{A}\}$
- $\pi \in \mathbf{\Pi}$ are expressed as lists of *swappings*: $[(a_1\ b_1), ..., (a_n, b_n)]$
- Roughly speaking freshness constraints $(a \# X)$ express the idea that the atom a doesn't occur unbounded in X
- *Freshness contexts* are represented by the set $\bigtriangledown \subset \mathcal{A} \times \mathcal{X}$

Atoms, variables, permutations, and freshness contexts

Basic definitions in Coq

`Inductive` **Atom** : Set := atom : **nat** $\to$ **Atom**.

`Inductive` **Var** : Set := var : **nat** $\to$ **Var**.

`Definition` Perm := **list** (**Atom** $\times$ **Atom**).

`Definition` Context := set (**Atom** $\times$ **Var**).

Nominal terms

$t, t_1, t_2 ::= \langle\rangle \mid a \mid [a]t \mid \langle t_1, t_2\rangle \mid f\ t \mid \pi.X$

```
Inductive term : Set :=
  | Ut : term
  | At : Atom → term
  | Ab : Atom → term → term
  | Pr : term → term → term
  | Fc : nat → term → term
  | Su : Perm → Var → term
.
```

π-action

$$
\begin{aligned}
\pi \cdot \langle\rangle &:= \langle\rangle \\
\pi \cdot \langle t_1, t_2\rangle &:= \langle \pi \cdot t_1, \pi \cdot t_2\rangle \\
\pi \cdot (f\ t) &:= f\ (\pi \cdot t) \\
\pi \cdot [a]t &:= [\pi \cdot a](\pi \cdot t) \\
\pi \cdot (\pi'.X) &:= (\pi' \oplus \pi).X
\end{aligned}
$$

$$
\begin{aligned}
[] \cdot a &:= a \\
((a\,b) :: \pi) \cdot c &:= \begin{cases} a \text{ if } & \pi \cdot c = b \\ b \text{ if } & \pi \cdot c = a \end{cases}
\end{aligned}
$$

#-relation

$$\frac{}{\nabla \vdash a \# \langle\rangle}\text{[\#-unit]}$$

$$\frac{a \neq b}{\nabla \vdash a \# b}\text{[\#-atom]}$$

$$\frac{\nabla \vdash a \# t}{\nabla \vdash a \# (f\ t)}\text{[\#-func]}$$

$$\frac{}{\nabla \vdash a \# [a]t}\text{[\#-abs}_1\text{]}$$

$$\frac{\nabla \vdash a \# t_1 \quad \nabla \vdash a \# t_2}{\nabla \vdash a \# \langle t_1, t_2\rangle}\text{[\#-pair]}$$

$$\frac{a \neq b \quad \nabla \vdash a \# t}{\nabla \vdash a \# [b]t}\text{[\#-abs}_2\text{]}$$

$$\frac{(\pi^{-1} \cdot a, X) \in \nabla}{\nabla \vdash a \# \pi.X}\text{[\#-susp]}$$

$\approx$-relation

$$\frac{}{\triangledown \vdash \langle\rangle \approx \langle\rangle}[\approx\text{-unit}]$$

$$\frac{\triangledown \vdash t_1 \approx t_1' \quad \triangledown \vdash t_2 \approx t_2'}{\triangledown \vdash \langle t_1, t_2\rangle \approx \langle t_1', t_2'\rangle}[\approx\text{-pair}]$$

$$\frac{}{\triangledown \vdash a \approx a}[\approx\text{-atom}]$$

$$\frac{a \neq b \quad \triangledown \vdash t \approx (a\,b)\cdot t' \quad \triangledown \vdash a \# t'}{\triangledown \vdash [a]t \approx [b]t'}[\approx\text{-abs}_2]$$

$$\frac{\triangledown \vdash t \approx t'}{\triangledown \vdash f\ t \approx f\ t'}[\approx\text{-func}]$$

$$\frac{\forall a \in ds(\pi, \pi'),\ (a, X) \in \triangledown}{\triangledown \vdash \pi.X \approx \pi'.X}[\approx\text{-susp}]$$

$$\frac{\triangledown \vdash t \approx t'}{\triangledown \vdash [a]t \approx [a]t'}[\approx\text{-abs}_1]$$

$$ds(\pi, \pi') := \{a \mid \pi \cdot a \neq \pi' \cdot a\}$$

Unification problems

Equational and freshness problems

- $t \approx_? t' \iff \exists(\nabla, \sigma), \nabla \vdash \sigma(t) \approx \sigma(t')$?
- $a \#_? t \iff \exists(\nabla, \sigma), \nabla \vdash a \# \sigma(t)$?

- Reasoning by simplification $\begin{cases} \approx_? \text{-rules} \\ \#_?\text{-rules} \end{cases}$

$\approx_?$-rules

[$\approx_?$-unit]	$\{\langle\rangle \approx_? \langle\rangle\} \uplus P$	$\stackrel{\epsilon}{\Longrightarrow}$	P
[$\approx_?$-pair]	$\{\langle t_1, t_2\rangle \approx_? \langle t_1', t_2'\rangle\} \uplus P$	$\stackrel{\epsilon}{\Longrightarrow}$	$\{t_1 \approx_? t_1', t_2 \approx_? t_2'\} \bigcup P$
[$\approx_?$-func]	$\{f\ t \approx_? f\ t'\} \uplus P$	$\stackrel{\epsilon}{\Longrightarrow}$	$\{t \approx_? t'\} \bigcup P$
[$\approx_?$-abs$_1$]	$\{[a]t \approx_? [a]t'\} \uplus P$	$\stackrel{\epsilon}{\Longrightarrow}$	$\{t \approx_? t'\} \bigcup P$
[$\approx_?$-abs$_2$]	$\{[a]t \approx_? [b]t'\} \uplus P$	$\underset{a \neq b}{\stackrel{\epsilon}{\Longrightarrow}}$	$\{t \approx_? (a\,b) \cdot t', a \#_? t'\} \bigcup P$
[$\approx_?$-atom]	$\{a \approx_? a\} \uplus P$	$\stackrel{\epsilon}{\Longrightarrow}$	P
[$\approx_?$-susp]	$\{\pi.X \approx_? \pi'.X\} \uplus P$	$\stackrel{\epsilon}{\Longrightarrow}$	$\{a \#_? X \mid a \in ds(\pi, \pi')\} \bigcup P$
[$\approx_?$-var$_1$]	$\{t \approx_? \pi.X\} \uplus P$	$\underset{X \notin t}{\xrightarrow{\sigma_k := [X \leftarrow \pi^{-1} \cdot t]}}$	$\sigma_k P$
[$\approx_?$-var$_2$]	$\{\pi.X \approx_? t\} \uplus P$	$\underset{X \notin t}{\xrightarrow{\sigma_k := [X \leftarrow \pi^{-1} \cdot t]}}$	$\sigma_k P$

$\#_?$-rules

[$\#_?$-unit]	$\{a \#_? \langle\rangle\} \uplus P$	$\overset{\emptyset}{\Longrightarrow}$	P
[$\#_?$-pair]	$\{a \#_? \langle t_1, t_2\rangle\} \uplus P$	$\overset{\emptyset}{\Longrightarrow}$	$\{a \#_? t_1, a \#_? t_2\} \bigcup P$
[$\#_?$-func]	$\{a \#_? (f\ t)\} \uplus P$	$\overset{\emptyset}{\Longrightarrow}$	$\{a \#_? t\} \bigcup P$
[$\#_?$-abs$_1$]	$\{a \#_? [a]t\} \uplus P$	$\overset{\emptyset}{\Longrightarrow}$	P
[$\#_?$-abs$_2$]	$\{a \#_? [b]t\} \uplus P$	$\underset{a \neq b}{\overset{\emptyset}{\Longrightarrow}}$	$\{a \#_? t\} \bigcup P$
[$\#_?$-atom]	$\{a \#_? b\} \uplus P$	$\underset{a \neq b}{\overset{\emptyset}{\Longrightarrow}}$	P
[$\#_?$-susp]	$\{a \#_? \pi.X\} \uplus P$	$\xrightarrow{\nabla_k := \{\pi^{-1} \cdot a \# X\}}$	P

A two phases algorithm

Phase 1

$$P_0 \xRightarrow{\epsilon,\ \sigma_1} P_1 \xRightarrow{\epsilon,\ \sigma_2} P_2 \xRightarrow{\cdots} P_n$$

If P_n contains any equational subproblems $\Rightarrow$ FAIL

Phase 2

$$P_n \xRightarrow{\emptyset,\ \nabla_1} P_{n+1} \xRightarrow{\emptyset,\ \nabla_2} P_{n+2} \xRightarrow{\cdots} P_{n+m}$$

$P_{n+m} \neq \emptyset \Rightarrow$ FAIL

If not fails, the output is

$$(\nabla, \sigma) \begin{cases} \nabla & := & \nabla_1 \cup \nabla_2 \cup \ldots \cup \nabla_i \\ \sigma & := & \sigma_1 \circ \sigma_2 \circ \ldots \circ \sigma_j \end{cases}$$

Fail example (extracted of [11])

$$\{[a][b]\langle X, b\rangle \approx_? [b][a]\langle a, X\rangle\} \xRightarrow[a \neq b]{\epsilon\ [\approx_?\text{-abs}_2]}$$

$$\{[b]\langle X, b\rangle \approx_? [b]\langle b, (a\,b).X\rangle, a \#_? ([a]\langle a, X\rangle)\} \xRightarrow{\epsilon\ [\approx_?\text{-abs}_1]}$$

$$\{\langle X, b\rangle \approx_? \langle b, (a\,b).X\rangle, a \#_? ([a]\langle a, X\rangle)\} \xRightarrow{\epsilon\ [\approx_?\text{-pair}]}$$

$$\{X \approx_? b, b \approx_? (a\,b).X, a \#_? ([a]\langle a, X\rangle)\} \xRightarrow[X \notin b]{\sigma_1 := [X \leftarrow b]}$$

$$\{b \approx_? a, a \#_? ([a]\langle a, b\rangle)\} \Longrightarrow \text{FAIL}$$

Phase 1 - success unification example (extracted of [11])

$$\{[a][b]\langle b, X\rangle \approx_? [a][a]\langle a, Y\rangle\} \xRightarrow{\epsilon\ [\approx_?\text{-abs}_1]}$$

$$\{[b]\langle b, X\rangle \approx_? [a]\langle a, Y\rangle\} \xRightarrow[a \neq b]{\epsilon\ [\approx_?\text{-abs}_2]}$$

$$\{\langle b, X\rangle \approx_? \langle b, (b\,a).Y\rangle, b \#_? \langle a, Y\rangle\} \xRightarrow{\epsilon\ [\approx_?\text{-pair}]}$$

$$\{b \approx_? b, X \approx_? (b\,a).Y, b \#_? \langle a, Y\rangle\} \xRightarrow{\epsilon\ [\approx_?\text{-atom}]}$$

$$\{X \approx_? (b\,a).Y, b \#_? \langle a, Y\rangle\} \xRightarrow[X \notin (b\,a).Y]{\sigma_1 := [X \leftarrow (b\,a).Y]}$$

$$\{b \#_? \langle a, Y\rangle\}$$

Phase 2 - success unification example

$$\{b \#_? \langle a, Y\rangle\} \xrightarrow[]{\emptyset \ [\#_?\text{-pair}]}$$

$$\{b \#_? a, b \#_? Y\} \xrightarrow[]{\emptyset \ [\#_?\text{-atom}]}$$

$$\{b \#_? Y\} \xrightarrow[]{\nabla_1 := \{b \# Y\}} \emptyset$$

$$(\nabla, \sigma) := (\{b \# Y\}, [X \leftarrow (b\, a).Y])$$

$$\{b \# Y\} \vdash [a][b]\langle b, (b\, a).Y\rangle \approx_? [a][a]\langle a, Y\rangle$$

In α-Prolog

- `test.apl`

```
id  :  name_type.
tm  :  type.

var :  id -> tm.
app :  tm -> tm -> tm.
lam :  id\tm -> tm.
```

1. `lam (a\lam (b\app X (var b))) = lam (b\lam (a\app (var a) X)).`
2. `lam (a\lam (b\app (var b) X)) = lam (a\lam (a\app (var a) Y)).`

According [17] we have

Termination

There is no infinite series of unification transitions

Correctness

Given a unification problem P:

1. if the algorithm fails on P, then P has no solution; and
2. if the algorithm succeeds on P, then the result it produces is an idempotent most general solution.

So far ...

- $- \vdash - \approx -$ is an equivalence relation:
  1. reflexivity ✓
  2. transitivity ✓
  3. symmetry ✓
- Basic properties over $\sigma(t)$ ✓
- The existence of Most General Unifiers •

More about the $\approx$-equivalence proof

Proof script:

- Reflexivity is trivial
- In [15] Urban provided a simpler proof of transitivity
- We have symmetry as a consequence of reflexivity and transitivity

The key lemmas before transitivity are:

1. $\forall a \in ds(\pi, \pi'), \bigtriangledown \vdash a \# t$ iff $\bigtriangledown \vdash \pi \cdot t \approx \pi' \cdot t$ (72 lines)
2. If $\bigtriangledown \vdash t_1 \approx t_2$ and $\bigtriangledown \vdash t_2 \approx \pi \cdot t_2$ then $\bigtriangledown \vdash t_1 \approx \pi \cdot t_2$ (96 lines)

Finally we reach transitivity:

- If $\bigtriangledown \vdash t_1 \approx t_2$ and $\bigtriangledown \vdash t_2 \approx t_3$ then $\vdash t_1 \approx t_3$ (64 lines)

Conclusion

- A naive implementation of this algorithm is exponential
- There is a more general form, named *equivariant unification* [6], that is a **NP**-complete problem
- Independently, *Levy and Villaret* [10] and *Calvès and Fernández* [4, 5] presented algorithms to solve nominal unification in quadratic time and space [3]
- Once we have finished the formalization of the simpler one we'd need to think how to extended the code to contain the more efficient versions

References I

[1] Brian Aydemir, Aaron Bohannon, and Stephanie Weirich.
Nominal reasoning techniques in coq.
Electronic Notes in Theoretical Computer Science, 174(5):69–77, 2007.

[2] William E Byrd and Daniel P Friedman.
A fresh name in nominal logic programming.

[3] Christophe Calvès.
Unifying nominal unification.
In *Rewriting Techniques and Applications (RTA)*, volume 21, pages 143–157. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.

[4] Christophe Calves and Maribel Fernández.
Implementing nominal unification.
Electronic Notes in Theoretical Computer Science, 176(1):25–37, 2007.

References II

[5] Christophe Calvès and Maribel Fernández.
The first-order nominal link.
Springer, 2011.

[6] James Cheney.
Equivariant unification.
In *Term Rewriting and Applications*, pages 74–89. Springer, 2005.

[7] James Cheney and Christian Urban.
System description: Alpha-prolog, a fresh approach to logic programming modulo alpha-equivalence.
In *Proc. 17th Int. Workshop on Unification, UNIF*, volume 3, pages 15–19, 2003.

[8] Ramana Kumar and Michael Norrish.
(nominal) unification by recursive descent with triangular substitutions.
In *Interactive Theorem Proving*, pages 51–66. Springer, 2010.

References III

[9] Matthew R Lakin and Andrew M Pitts.
Resolving inductive definitions with binders in higher-order typed functional programming.
In *Programming Languages and Systems*, pages 47–61. Springer, 2009.

[10] Jordi Levy and Mateu Villaret i Ausellé.
An efficient nominal unification algorithm.
In *Leibniz International Proceedings in Informatics (LIPPICS): Proceedings of the 21st International Conference on Rewriting Techniques and Applications: RTA 2010: July 11-13 2010: Edinburgh, Scottland, UK, 2010, vol. 6, p. 209-226*. Dagstuhl Publishing, 2010.

[11] Andrew M Pitts.
Nominal Sets: Names and Symmetry in Computer Science, volume 57.
Cambridge University Press, 2013.

References IV

[12] François Pottier.
An overview of cαml.
Electronic Notes in Theoretical Computer Science, 148(2):27–52, 2006.

[13] Mark R Shinwell.
Fresh o'caml: nominal abstract syntax for the masses.
Electronic Notes in Theoretical Computer Science, 148(2):53–77, 2006.

[14] Mark R. Shinwell, Andrew M. Pitts, and Murdoch J. Gabbay.
Freshml: Programming with binders made simple.
SIGPLAN Not., 38(9):263–274, August 2003.

[15] Christian Urban.
Nominal unification revisited.
arXiv preprint arXiv:1012.4890, 2010.

References V

[16] Christian Urban and James Cheney.
Avoiding equivariance in alpha-prolog.
In *Typed Lambda Calculi and Applications*, pages 401–416.
Springer, 2005.

[17] Christian Urban, Andrew M Pitts, and Murdoch J Gabbay.
Nominal unification.
Theoretical Computer Science, 323(1):473–497, 2004.

[18] Christian Urban and Christine Tasson.
Nominal techniques in isabelle/hol.
In *Automated Deduction–CADE-20*, pages 38–53. Springer, 2005.

Thank You ...