Efficiency of a good but not linear nominal unification algorithm

Weixi Ma, Jeremy G. Siek, David Thrane Christiansen, Daniel P. Friedman

June 25, 2018

Nominal Unification [Urban et al., 2004]

$$\lambda\, a.\lambda\, b.X\ Y \stackrel{?}{=} \lambda\, a.\lambda\, a.a\, Z$$

Unifier:

$\{X/b,\ Y/(a \leftrightarrow b) \cdot Z\}$ and $\{b \# Z\}$

Nominal Unification [Urban et al., 2004]

- The fastest existing nominal unification algorithms are $O(n^2)$.
- One n is the size of the term.
- The other n is the number of binders, e.g., $X \stackrel{?}{=} (a_0 \leftrightarrow a'_0, \ldots, a_n \leftrightarrow a'_n)a$.
- Lists cannot be replaced with structures of better lookup efficiency, because of $\pi \cdot (\pi' \cdot X) \equiv (\pi @ \pi') \cdot X$.

It's straightforward to check α-equivalence

using de Bruijn numbers [de Bruijn, 1972]

$$\lambda\, a.\lambda\, b.a \overset{\alpha}{=} \lambda\, c.\lambda\, d.c$$

In de Bruijn numbers:

$$\lambda.\, \lambda\, .1 \overset{\alpha}{=} \lambda\, .\lambda\, .1$$

De Bruijn numbers should coexist with names

A *scope* is a list of names.
A *static closure* consists of a term and its enclosing scope.

term: $\lambda\, a.\lambda\, b.a$
closure: $\langle (b\, a); a \rangle$

Free and Bound

$$\frac{a \notin \Phi}{\Phi \vdash \texttt{Fr}\ a}\ [\textsc{Free}]$$

$$\frac{\begin{array}{c}(\texttt{name}{\rightarrow}\texttt{idx}\ \Phi\ a) = i \\ (\texttt{idx}{\rightarrow}\texttt{name}\ \Phi\ i) = a\end{array}}{\Phi \vdash \texttt{Bd}\ a\ i}\ [\textsc{Bound}]$$

α-equivalence between name-closures

$$\frac{a_1 = a_2 \quad \Phi_1 \vdash \texttt{Fr}\ a_1 \quad \Phi_2 \vdash \texttt{Fr}\ a_2}{\langle \Phi_1; a_1 \rangle \approx \langle \Phi_2; a_2 \rangle}\ [\textsc{Same-Free}]$$

$$\frac{i_1 = i_2 \quad \Phi_1 \vdash \texttt{Bd}\ a_1\ i_1 \quad \Phi_2 \vdash \texttt{Bd}\ a_2\ i_2}{\langle \Phi_1; a_1 \rangle \approx \langle \Phi_2; a_2 \rangle}\ [\textsc{Same-Bound}]$$

Unifying name-closures

$$\lambda a.\lambda b.a \stackrel{?}{=} \lambda c.\lambda d.c$$

$$\sigma \vdash \langle (b\,a); a \rangle \stackrel{\alpha}{=} \langle (d\,c); c \rangle \Rightarrow \sigma$$

Unifying name-closure versus var-closure

$$\lambda\, a.\lambda\, b.X \stackrel{?}{=} \lambda\, c.\lambda\, d.c$$

$$\sigma \vdash \langle (b\, a); X\rangle \stackrel{\alpha}{=} \langle (d\, c); c\rangle \Rightarrow \{X/a\} \cup \sigma$$

Unifying name-closure versus var-closure

another example

$$\lambda\, a.\lambda\, a.X \overset{?}{=} \lambda\, c.\lambda\, d.c$$

$$\sigma \vdash \langle (a\, a); X\rangle \overset{\alpha}{=} \langle (d\, c); c\rangle \Rightarrow \bot$$

Unifying var-closures

$$\lambda a.\lambda b.X \stackrel{?}{=} \lambda c.\lambda d.Y$$

$$\sigma \vdash \langle (b\, a); X \rangle \stackrel{\alpha}{=} \langle (d\, c); Y \rangle \Rightarrow ?$$

σ is not enough.

Unifying var-closures

$$\lambda\, a.\lambda\, b.X \stackrel{?}{=} \lambda\, c.\lambda\, d.Y$$

$$\sigma;\ \delta \vdash \langle (b\, a); X\rangle \stackrel{\alpha}{=} \langle (d\ c);\ Y\rangle \Rightarrow \sigma;\ \delta'$$

$$\delta' = \{\langle (b\, a); X\rangle \stackrel{\alpha}{=} \langle (d\ c);\ Y\rangle\} \cup \delta$$

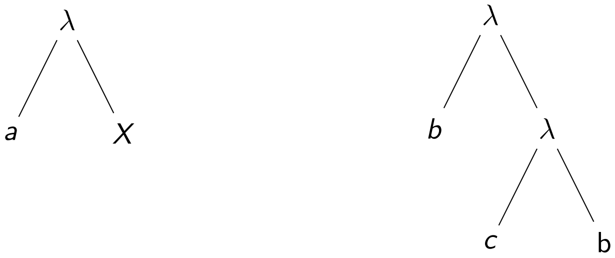
To sum up:

- De Bruijn numbers get along with names in closures.
- σ is a substitution.
- δ contains unsolved pairs of var-closures.
- We have solved the simple case where unification variables only associate with names.
- Next, let's look at $X \stackrel{?}{=} \lambda\, a.Y$ and $X \stackrel{?}{=} Y\, Z$.

To generalize the algorithm:

Need: a shared shape for two arbitrary terms

$$\lambda\, a.X \stackrel{?}{=} \lambda\, b.\lambda\, c.b$$



To generalize the algorithm:

Need: a shared shape for two arbitrary terms

$$\lambda\, a.X \stackrel{?}{=} \lambda\, b.\lambda\, c.b$$

Also, $\sigma = \{X/\lambda\, c'.Y\} \cup \sigma$

Something about Φ

$$\frac{\delta_0;\sigma_0 \vdash \langle a_1, \Phi_1; t_1\rangle \overset{\alpha}{=} \langle a_2, \Phi_2; t_2\rangle \Rightarrow \delta_1;\sigma_1}{\delta_0;\sigma_0 \vdash \langle \Phi_1; \lambda\, a_1.t_1\rangle \overset{\alpha}{=} \langle \Phi_2; \lambda\, a_2.t_2\rangle \Rightarrow \delta_1;\sigma_1}\ [\textsc{Abs}]$$

$$\frac{\begin{array}{c}\delta_0;\sigma_0 \vdash \langle \Phi_1; l_1\rangle \overset{\alpha}{=} \langle \Phi_2; l_2\rangle \Rightarrow \delta_0';\sigma_0' \\ \delta_0';\sigma_0' \vdash \langle \Phi_1; r_1\rangle \overset{\alpha}{=} \langle \Phi_2; r_2\rangle \Rightarrow \delta_1;\sigma_1\end{array}}{\delta_0;\sigma_0 \vdash \langle \Phi_1; l_1\ r_1\rangle \overset{\alpha}{=} \langle \Phi_2; l_2\ r_2\rangle \Rightarrow \delta_1;\sigma_1}\ [\textsc{App}]$$

Why is it linear?

- Every term is annotated with a scope, no swappings are needed ($\pi @ \pi'$ is gone).
- A scope, previously represented by a list, can be replaced with two hashtables.
- Persistent hashtables [Bagwell, 2001] provide efficient lookup and access, and more importantly, sharing.

Replacing lists with hashtables

$$\lambda\, a.\lambda\, b.\lambda\, a....$$

indices with list:

$(a\ b\ a)$

levels with two hashtables:

$$\begin{array}{lcl} a \mapsto 0 & \quad & 0 \mapsto a \\ b \mapsto 1 & & 1 \mapsto b \\ a \mapsto 2 & & 2 \mapsto a \end{array}$$

Conclusion

Nominal unification is linear!

Primary references

Phil Bagwell. Ideal Hash Trees. *Technical Report EPFL-REPORT-169879, Ecole polytechnique fédérale de Lausanne*, November 2001.

Nicolaas G. de Bruijn. Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. *Indagationes Mathematicae (Proceedings)*, 75(5):381–392, January 1972.

Christian Urban, Andrew M. Pitts, and Murdoch J. Gabbay. Nominal unification. *Theoretical Computer Science*, 323(1-3): 473–497, September 2004.

Thank you!