Relating Nominal and Higher-order Abstract Syntax Specifications

Andrew Gacek

INRIA Saclay - Île-de-France
& LIX/École polytechnique

PPDP'10
July 26–28, 2010
Hagenberg, Austria

Relating the nominal and HOAS worlds

Many approaches to formalizing systems with binding structure

Nominal: names, name-abstraction, freshness, И-quantifier
HOAS: λ-terms, raising, ∇-quantifier

Some convergence: Nominal vs higher-order pattern unification
[Cheney 2005, Levy and Villaret 2008]

Difficulties: αProlog vs λProlog

Current work: a translation from αProlog to $\mathcal{G}^-$
and from λProlog to $\mathcal{G}^-$

αProlog example

Encoding of λ-terms

$$\lambda x.\lambda y.x\ y \quad \rightsquigarrow \quad lam(\langle a\rangle lam(\langle b\rangle app(var(a), var(b))))$$

Type checking for λ-terms

$$tc(G, var(X), A) :- lookup(X, A, G)$$
$$tc(G, app(M, N), B) :- \exists A.tc(G, M, arr(A, B)) \wedge tc(G, N, A)$$
$$tc(G, lam(\langle x\rangle E), arr(A, B)) :- x\#G \wedge tc(bind(x, A, G), E, B)$$

$$\text{И}x.\forall G.\forall E.\forall A.\forall B.$$
$$tc(G, lam(\langle x\rangle E), arr(A, B)) :-$$
$$x\#G \wedge tc(bind(x, A, G), E, B)$$

αProlog basics

Syntax

$$t, u ::= a \mid X \mid f(\vec{t}) \mid (a\ b) \cdot t \mid \langle a \rangle t$$

$$G ::= \top \mid p(\vec{t}) \mid a \# t \mid t \approx u \mid G \wedge G' \mid G \vee G' \mid \exists X. G \mid \text{И} a. G$$

$$D ::= \text{И} \vec{a}. \forall \vec{X}. [p(\vec{t}) :- G]$$

Notions

Swapping: $(a\ b) \cdot (\langle a \rangle b) = \langle b \rangle a$

Freshness: $a \# \langle a \rangle t$

α-Equivalence: $\langle a \rangle a \approx \langle b \rangle b$

Variable capture: $\text{И} a. \exists X. \langle a \rangle X \approx \langle b \rangle b$ has solution $X \mapsto a$

αProlog rules

$$\frac{}{\Delta \Longrightarrow \top}\ \text{TRUE} \qquad \frac{\models a\#t}{\Delta \Longrightarrow a\#t}\ \text{FRESH} \qquad \frac{\models t \approx u}{\Delta \Longrightarrow t \approx u}\ \text{EQUAL}$$

$$\frac{\Delta \Longrightarrow G_1 \quad \Delta \Longrightarrow G_2}{\Delta \Longrightarrow G_1 \wedge G_2}\ \text{AND} \qquad \frac{\Delta \Longrightarrow G_i}{\Delta \Longrightarrow G_1 \vee G_2}\ \text{OR}$$

$$\frac{\Delta \Longrightarrow G[t/X]}{\Delta \Longrightarrow \exists X.G}\ \text{EXISTS} \qquad \frac{\Delta \Longrightarrow G}{\Delta \Longrightarrow \text{И}a.G}\ \text{NEW}$$

$$\frac{\Delta \Longrightarrow \pi.(G\theta)}{\Delta \Longrightarrow p(\vec{t})}\ \text{BACKCHAIN}$$

Where $\text{И}\vec{a}.\forall\vec{X}.[p(\vec{u}) :- G] \in \Delta$ and π is a permutation and θ is a substitution for $\vec{X}$ such that $\vec{t} \approx \pi.(\vec{u}\theta)$.

$\mathcal{G}^-$ example

Encoding of λ-terms

$$\lambda x.\lambda y.x\ y \quad \leadsto \quad lam\ (\lambda x.lam\ (\lambda y.app\ (var\ x)\ (var\ y)))$$

Type checking for λ-terms

$$tc\ G\ (var\ X)\ A \triangleq lookup\ X\ A\ G$$

$$tc\ G\ (app\ M\ N)\ B \triangleq \exists A.tc\ G\ M\ (arr\ A\ B) \wedge tc\ G\ N\ A$$

$$tc\ G\ (lam\ \lambda x.E\ x)\ (arr\ A\ B) \triangleq \nabla x.tc\ (bind\ x\ A\ G)\ (E\ x)\ B$$

$\mathcal{G}^-$ basics

Syntax

$$
\begin{aligned}
t, u &::= x \mid c \mid a \mid (t\,u) \mid \lambda x.t \\
B, C &::= \top \mid p\,\vec{t} \mid t = u \mid B \wedge C \mid B \vee C \mid \exists x.B \mid \nabla z.B \\
D &::= \forall \vec{x}.[(\nabla \vec{z}.p\,\vec{u}) \triangleq B]
\end{aligned}
$$

Notions

Equality is λ-conversion: $\lambda a.a = \lambda b.b$, $(\lambda x.t)\,u = t[u/x]$

Capture-avoiding substitution: $\exists X.\lambda a.X = \lambda b.b$ has no solution.

$\mathcal{G}^-$ rules

$$\frac{}{\longrightarrow \top}\ \top\mathcal{R} \qquad\qquad \frac{}{\longrightarrow t = t}\ =\mathcal{R}$$

$$\frac{\longrightarrow B_1 \quad \longrightarrow B_2}{\longrightarrow B_1 \wedge B_2}\ \wedge\mathcal{R} \qquad\qquad \frac{\longrightarrow B_i}{\longrightarrow B_1 \vee B_2}\ \vee\mathcal{R}$$

$$\frac{\longrightarrow B[t/x]}{\longrightarrow \exists x.B}\ \exists\mathcal{R} \qquad\qquad \frac{\longrightarrow B[a/x]}{\longrightarrow \nabla x.B}\ \nabla\mathcal{R},\ a \notin \mathrm{supp}(B)$$

$$\frac{\longrightarrow B\theta}{\longrightarrow p\ \vec{t}}\ def\mathcal{R}$$

Where $\forall \vec{x}.[(\nabla \vec{z}.p\ \vec{u}) \triangleq B] \in \mathcal{D}$ and θ is a substitution for $\vec{z}$ and $\vec{x}$ such that each $z_i\theta$ is a unique nominal constant, $\mathrm{supp}(\vec{x}\theta) \cap \{\vec{z}\theta\} = \emptyset$, and $\vec{t} = \vec{u}\theta$.

A Naive Translation

$$\langle\cdot\rangle\cdot \;\rightsquigarrow\; \lambda \qquad \text{И} \;\rightsquigarrow\; \nabla \qquad \approx \;\rightsquigarrow\; =$$

Problem

$$\text{И}a.\exists X.\langle a\rangle X \approx \langle b\rangle b \;\rightsquigarrow\; \nabla a.\exists X.\lambda a.X = \lambda b.b$$

The first has solution $X \mapsto a$, the second has no solution

Solution

Use raising to explicitly encode dependencies:

$$\text{И}a.\exists X.\langle a\rangle X \approx \langle b\rangle b \;\rightsquigarrow\; \exists X.\lambda a.X\ a = \lambda b.b$$

Now the second formula has solution $X \mapsto \lambda y.y$

Freshness

There is no direct analog of $a \# t$ in $\mathcal{G}^-$

But we can define it:

$$\forall E.(\nabla x.\mathit{fresh}\ x\ E) \triangleq \top$$

x is quantified inside the scope of E so no substitution for E can contain the value of x

Translation for terms

$$\phi(a) = a \qquad \phi(f(\vec{t})) = f\ \overrightarrow{\phi(t)} \qquad \phi(\langle a \rangle t) = \lambda a.\phi(t)$$

$$\phi(X\,\vec{a}) = X\,\vec{a} \qquad \phi((a\ b) \cdot t) = (a\ b) \cdot \phi(t)$$

Example

$$lam(\langle a \rangle lam(\langle b \rangle app(X, (a\ b) \cdot X)))$$

$$\rightsquigarrow$$

$$lam\ (\lambda a.lam\ (\lambda b.app\ (X\ a\ b)\ (X\ b\ a))))$$

Translation for goals and clauses

$$\begin{aligned}
\phi_{\vec{a}}(\top) &= \top \\
\phi_{\vec{a}}(p\,\vec{t}) &= \nabla\vec{a}.p\,\overrightarrow{\phi(t)} \\
\phi_{\vec{a}}(a\#t) &= \nabla\vec{a}.fresh\,\phi(a)\,\phi(t) \\
\phi_{\vec{a}}(t \approx u) &= \nabla\vec{a}.(\phi(t) = \phi(u)) \\
\phi_{\vec{a}}(G_1 \wedge G_2) &= \phi_{\vec{a}}(G_1) \wedge \phi_{\vec{a}}(G_2) \\
\phi_{\vec{a}}(G_1 \vee G_2) &= \phi_{\vec{a}}(G_1) \vee \phi_{\vec{a}}(G_2) \\
\phi_{\vec{a}}(\exists X.G) &= \exists X.\phi_{\vec{a}}(G[X\,\vec{a}/X]) \\
\phi_{\vec{a}}(\text{И}b.G) &= \phi_{\vec{a}b}(G)
\end{aligned}$$

$$\phi\left(\text{И}\vec{a}.\forall\vec{X}.[p(\vec{t}) :- G]\right) = \forall\vec{X}.[(\nabla\vec{a}.p\,\overrightarrow{\phi(t\sigma)}) \triangleq \phi_{\vec{a}}(G\sigma))]$$

$$\text{where } \sigma = \{X\,\vec{a}/X \mid X \in \vec{X}\}$$

Correctness of the translation

Theorem (Soundness)

If $\Delta \Longrightarrow G$ *then* $\longrightarrow \phi(G)$ *with the definitions* $\phi(\Delta)$

Theorem (Completeness)

If $\longrightarrow \phi(G)$ *with the definitions* $\phi(\Delta)$ *then* $\Delta \Longrightarrow G$

Type checking example

$$tc(G, var(X), A) \;{:}{-}\; lookup(X, A, G)$$

$$tc(G, app(M, N), B) \;{:}{-}\; \exists A.tc(G, M, arr(A, B)) \wedge tc(G, N, A)$$

$$tc(G, lam(\langle x \rangle E), arr(A, B)) \;{:}{-}\; x \# G \wedge tc(bind(x, A, G), E, B)$$

$$tc\ G\ (var\ X)\ A \triangleq lookup\ X\ A\ G$$

$$tc\ G\ (app\ M\ N)\ B \triangleq \exists A.tc\ G\ M\ (arr\ A\ B) \wedge tc\ G\ N\ A$$

$$\begin{aligned}&(\nabla x.tc\ (G\,x)\ (lam\ \lambda x.E\,x)\ (arr\ (A\,x)\ (B\,x))) \triangleq \\ &\qquad (\nabla x.fresh\ x\ (G\,x)) \wedge \\ &\qquad (\nabla x.tc\ (bind\ x\ (A\,x)\ (G\,x))\ (E\,x)\ (B\,x))\end{aligned}$$

Simplifications

$$\begin{aligned}(\nabla x.tc\ (G\,x)\ (lam\ \lambda x.E\,x)\ (arr\ (A\,x)\ (B\,x))) \triangleq \\ (\nabla x.fresh\ x\ (G\,x)) \wedge \\ (\nabla x.tc\ (bind\ x\ (A\,x)\ (G\,x))\ (E\,x)\ (B\,x))\end{aligned}$$

1. Statically solve freshness constraint:

$$\begin{aligned}(\nabla x.tc\ G\ (lam\ \lambda x.E\,x)\ (arr\ (A\,x)\ (B\,x))) \triangleq \\ \nabla x.tc\ (bind\ x\ (A\,x)\ G)\ (E\,x)\ (B\,x)\end{aligned}$$

2. Use *subordination* to elimination vacuous raisings:

$$\begin{aligned}(\nabla x.tc\ G\ (lam\ \lambda x.E\,x)\ (arr\ A\ B)) \triangleq \\ \nabla x.tc\ (bind\ x\ A\ G)\ (E\,x)\ B\end{aligned}$$

3. Remove vacuous ∇s:

$$tc\ G\ (lam\ \lambda x.E\,x)\ (arr\ A\ B) \triangleq \nabla x.tc\ (bind\ x\ A\ G)\ (E\,x)\ B$$

Extending the translation

αProlog allows arbitrary abstraction and swapping:

$$\langle u \rangle t \qquad\qquad (u_1\ u_2) \cdot t$$

$$\begin{aligned} t' \approx \langle u \rangle t &\quad\rightsquigarrow\quad \mathit{abst}\ u\ t\ t' \\ t' \approx (u_1\ u_2) \cdot t &\quad\rightsquigarrow\quad \mathit{swap}\ u_1\ u_2\ t\ t' \end{aligned}$$

$$\forall E.(\nabla x.\mathit{abst}\ x\ (E\ x)\ (\lambda x.E\ x)) \triangleq \top$$

$$\forall E.(\nabla x, y.\mathit{swap}\ x\ y\ (E\ x\ y)\ (E\ y\ x)) \triangleq \top$$

$$\forall E.(\nabla x.\mathit{swap}\ x\ x\ (E\ x)\ (E\ x)) \triangleq \top$$

Going fully higher-order

Encoding of λ-terms

$$\lambda x.\lambda y.x\ y \quad \leadsto \quad \mathit{lam}\ \lambda x.\mathit{lam}\ \lambda y.\mathit{app}\ x\ y$$

Type checking for λ-terms in λProlog

$$\mathit{tc}\ (\mathit{app}\ M\ N)\ B :- \mathit{tc}\ M\ (\mathit{arr}\ A\ B) \wedge \mathit{tc}\ N\ A$$
$$\mathit{tc}\ (\mathit{lam}\ \lambda x.R\,x)\ (\mathit{arr}\ A\ B) :- \forall x.\mathit{tc}\ x\ A \Rightarrow \mathit{tc}\ (R\,x)\ B$$

Free lemma

If $\Delta \vdash \mathit{tc}\ (\mathit{lam}\ \lambda x.R\,x)\ (\mathit{arr}\ A\ B)$ and $\Delta \vdash \mathit{tc}\ N\ A$ then $\Delta \vdash \mathit{tc}\ (R\,N)\ B$

λProlog in $\mathcal{G}^-$

$$seq\, L\, \top \triangleq \top$$

$$seq\, L\, (B \wedge C) \triangleq seq\, L\, B \wedge seq\, L\, C$$

$$seq\, L\, (A \Rightarrow B) \triangleq seq\, (A :: L)\, B$$

$$seq\, L\, (\forall x. B\, x) \triangleq \nabla x. seq\, L\, (B\, x)$$

$$seq\, L\, \langle A \rangle \triangleq member\, A\, L$$

$$seq\, L\, \langle A \rangle \triangleq \exists B. prog\, A\, B \wedge seq\, L\, B$$

$$prog\, (tc\, (app\, M\, N)\, B)$$
$$(\langle tc\, M\, (arr\, A\, B)\rangle \wedge \langle tc\, N\, A\rangle) \triangleq \top$$

$$prog\, (tc\, (lam\, \lambda x. R\, x)\, (arr\, A\, B))$$
$$(\forall x. tc\, x\, A \Rightarrow \langle tc\, (R\, x)\, B\rangle) \triangleq \top$$

Future Work

- Reverse translation [Cheney 2005]
- Mixing $\mathcal{G}^-$ and λProlog specifications
- Reasoning about αProlog via $\mathcal{G}$
- Identifying special subclasses of αProlog specifications
- Unification, specification, reasoning