The Semantics of Nominal Logic Programs

James Cheney

ICLP 2006

August 19, 2006

Motivation

- *Nominal logic* [Pitts 2003] is a first-order axiomatization of names, name-binding, and alpha-equivalence

- Provides a logical foundation for logic programming with "concrete" names

- Much more convenient for prototyping type systems,

- "First-class" names, including nondeterministic fresh name generation, so sometimes more convenient than HO abstract syntax

Example

- A (very tired) example: typechecking.

$$\frac{x : T \in \Gamma}{\Gamma \vdash x : T} \qquad \frac{\Gamma \vdash e : T \to U \quad \Gamma \vdash f : T}{\Gamma \vdash e\ f : U} \qquad \frac{(x \notin \Gamma) \quad \Gamma, x : T \vdash e : U}{\Gamma \vdash \lambda x.e : T \to U}$$

```
tc(G,var(X),T)              :- mem((X,T),G).
tc(G,app(E,F),U)            :- tc(G,E,arr(T,U)), tc(G,F,T).
tc(G,lam(x\E),arr(T,U)) :- x # G, tc([(x,T)|G],E,U).
```

- Note that clauses and subgoals correspond exactly (read x # G as $x \notin \Gamma$)

Example

- Large-step semantics for ML-like references:

$$\frac{(a \in Lab)}{\langle M, a\rangle \rightarrow \langle M, a\rangle} \qquad \frac{\langle M, e_1\rangle \rightarrow \langle M', a\rangle \quad \langle M', e_2\rangle \rightarrow \langle M'', v\rangle}{\langle M, e_1 := e_2\rangle \rightarrow \langle M''[a := v], ()\rangle}$$

$$\frac{\langle M, e\rangle \rightarrow \langle M', a\rangle}{\langle M, !e\rangle \rightarrow \langle M', M'(a)\rangle} \qquad \frac{\langle M, e\rangle \rightarrow \langle M', v\rangle \quad (a \notin dom(M'))}{\langle M, \mathsf{ref}\ e\rangle \rightarrow \langle M'[a := v], a\rangle}$$

- Interesting part: last rule requires *fresh* label for new memory cell

Example

- Large-step semantics for ML-like references:

```
(M,lab(A))         `eval` (M,lab(A)).
(M,assign(E1,E2)) `eval` (M3,unit)              :- (M,E1) `eval` (M1,lab(A)),
                                                   (M1, E2) `eval` (M2,V),
                                                   update((A,V),M2,M3).
(M,deref(E))       `eval` (M',V)                 :- (M,E) `eval` (M',lab(A)),
                                                   mem((A,V),M').
(M,ref(E))         `eval` ([(a,V)|M'],lab(a)) :- (M,E) `eval` ,(M',V),
                                                   a#M'.
```

- Interesting part: in last rule, name a is constrained to be sufficiently fresh

Motivation (II)

- Previous papers have considered differing operational, proof-theoretic, and denotational semantics separately...

- This paper gives a *unified* presentation that ties them together

- **Main contribution:** Improved "uniform proof" semantics

Notation

a, b	$\in$	$\mathbb{A}$	Atoms/Names
f, g	$\in$	$FnSym$	Term symbols
X, Y	$\in$	Var	Variables
a, b, t, u	$::=$	$c \mid f(\vec{t}) \mid X$	First-order terms
	$\mid$	$\langle a \rangle t \mid (a\ b) \cdot t \mid \mathsf{a}$	Nominal terms
C	$::=$	$t \approx u \mid a \# t$	Equality, freshness
Σ	$::=$	$\cdot \mid \Sigma, X{:}\tau \mid \Sigma \# \mathsf{a}{:}\nu$	Contexts
∇	$::=$	$\cdot \mid \nabla, C$	Constraint sets

Note: Contexts $\Sigma \# \mathsf{a}$ have special meaning: name a cannot occur free in any variables in Σ.

Ground swapping

The result of applying a swapping (b b′) to a ground term is:

$$\begin{array}{rcl}(\mathsf{b}\ \mathsf{b}')\cdot\mathsf{a} & = & (\mathsf{b}\ \mathsf{b}')(\mathsf{a})\\ (\mathsf{b}\ \mathsf{b}')\cdot c & = & c\\ (\mathsf{b}\ \mathsf{b}')\cdot f(\vec{t}) & = & f((\mathsf{b}\ \mathsf{b}')\cdot t_1,\ldots,(\mathsf{b}\ \mathsf{b}')\cdot t_n)\\ (\mathsf{b}\ \mathsf{b}')\cdot\langle\mathsf{a}\rangle t & = & \langle(\mathsf{b}\ \mathsf{b}')\cdot\mathsf{a}\rangle(\mathsf{b}\ \mathsf{b}')\cdot t\end{array}$$

where

$$(\mathsf{b}\ \mathsf{b}')(\mathsf{a}) = \begin{cases}\mathsf{b} & (\mathsf{a}=\mathsf{b}')\\ \mathsf{b}' & (\mathsf{a}=\mathsf{b})\\ \mathsf{a} & (\mathsf{a}\neq\mathsf{b}\neq\mathsf{b}')\end{cases}$$

Note: In case of abstraction, no α-renaming is needed; swapping is intrinsically capture-avoiding!

Ground freshness theory

$$\frac{(\mathsf{a} \neq \mathsf{b})}{\mathsf{a} \# \mathsf{b}}$$ Different names fresh

$$\frac{}{\mathsf{a} \# c}$$ Anything fresh for constant

$$\frac{\mathsf{a} \# t_1 \quad \cdots \quad \mathsf{a} \# t_n}{\mathsf{a} \# f(\vec{t})}$$ Freshness ignores function symbols

$$\frac{(\mathsf{a} \neq \mathsf{b}) \quad \mathsf{a} \# t}{\mathsf{a} \# \langle \mathsf{b} \rangle t}$$ Fresh if fresh for body

$$\frac{}{\mathsf{a} \# \langle \mathsf{a} \rangle t}$$ Fresh if bound

Ground equational theory

$$\frac{}{\mathsf{a} \approx \mathsf{a}}$$

$$\frac{}{c \approx c}$$

$$\frac{t_1 \approx u_1 \quad \cdots \quad t_n \approx u_n}{f(\vec{t}) \approx f(\vec{u})}$$

$$\frac{t \approx u}{\langle \mathsf{a} \rangle t \approx \langle \mathsf{a} \rangle u}$$

Standard equational rules

$$\frac{(\mathsf{a} \neq \mathsf{b}) \quad \mathsf{a} \# u \quad t \approx (\mathsf{a}\ \mathsf{b}) \cdot u}{\langle \mathsf{a} \rangle t \approx \langle \mathsf{b} \rangle u}$$

α-equivalence for abstractions

Don't worry if that went by a little fast.

The constraint theory is largely irrelevant to the rest of the talk.

The И-quantifier

- The semantics of the И-quantifier on ground formulas ϕ is as follows

$$\models \text{Иa}.\phi \iff \models (\mathsf{a}\ \mathsf{b}) \cdot \phi \text{ for some } \mathsf{b} \notin supp(\text{Иa}.\phi)$$

  More generally, if $\mathsf{a} \notin FN(\Sigma)$,

$$\Sigma : \nabla \models \text{Иa}.\phi \iff \Sigma \# \mathsf{a} : \nabla \models \phi$$

- Example:

$$\models \text{Иa.Иb.a} \# \mathsf{b} \qquad \models \forall X.\text{Иa.a} \# X \qquad \not\models \text{Иa}.\forall X.\mathsf{a} \# X$$

Nominal logic goals and programs

- Goal formulae and program clauses are of the form

$$
\begin{array}{rcl}
G & ::= & A \mid C \mid \top \mid G \wedge G' \mid G \vee G' \mid \exists X.G \mid \text{И}\mathsf{a}.G \\
D & ::= & A \mid \top \mid D \wedge D \mid G \supset D \mid \forall X.D \mid \text{И}\mathsf{a}.D
\end{array}
$$

- Note: We interpret

$$A :- B_1, \ldots, B_n \text{ as } \text{И}\vec{\mathsf{a}}.\forall \vec{X}.B_1 \wedge \cdots \wedge B_n \supset A$$

where $\vec{\mathsf{a}} = FN(A, \vec{B})$ and $\vec{X} = FV(A, \vec{B})$.

- Example:

$\text{И}\mathsf{a}.\forall G, E, T.\mathsf{a} \# G \wedge tc([(\mathsf{a}, T)|G], E, U) \supset tc(G, \lambda(\langle \mathsf{a} \rangle E), arr(T, U))$

Denotational semantics

- Consider Herbrand (term) models only; a model is (essentially) a set S of atomic formulas.

- Given program clause D, define one-step deduction operator T_D thusly:

$$
\begin{array}{rcl}
T_{\top}(S) & = & S \\
T_A(S) & = & S \cup A \\
T_{D_1 \wedge D_2}(S) & = & T_{D_1}(S) \cup T_{D_2}(S) \\
T_{G \supset D}(S) & = & \begin{cases} T_D(S) & \text{if } S \vDash G \\ S & \text{otherwise} \end{cases} \\
T_{\forall X:\sigma.D}(S) & = & \bigcup_{t:\sigma} T_{D[t/X]}(S) \\
T_{\text{И}\mathsf{a}:\nu.D}(S) & = & \bigcup_{\mathsf{b}:\nu \notin FN(\text{И}\mathsf{a}.D)} T_{(\mathsf{a}\ \mathsf{b})\cdot D}(S)
\end{array}
$$

Uniform/focused proofs

- Define a proof theory that captures *uniform* (goal-directed) and *atomic* (program clause-directed) proofs

- $\Sigma : \Delta ; \nabla \Longrightarrow G$: given program Δ, constraint ∇ implies G.

- $\Sigma : \Delta ; \nabla \xrightarrow{D} A$: given program Δ, constraint ∇ and program clause D immediately imply A. ("Focused" proofs)

- Quantifier rules use constraints rather than substitutions.

Goal-directed proofs

$$\frac{\Sigma : \nabla \models C}{\Sigma : \Delta; \nabla \Longrightarrow C}\, con \qquad \frac{}{\Sigma : \Delta; \nabla \Longrightarrow \top}\, \top R$$

$$\frac{\Sigma : \Delta; \nabla \Longrightarrow G_1 \quad \Sigma : \Delta; \nabla \Longrightarrow G_2}{\Sigma : \Delta; \nabla \Longrightarrow G_1 \wedge G_2}\, \wedge R$$

$$\frac{\Sigma : \Delta; \nabla \Longrightarrow G_i}{\Sigma : \Delta; \nabla \Longrightarrow G_1 \vee G_2}\, \vee R_i$$

$$\frac{\Sigma : \nabla \models \exists X.C \quad \Sigma, X : \Delta; \nabla, C \Longrightarrow G}{\Sigma : \Delta; \nabla \Longrightarrow \exists X{:}\sigma.G}\, \exists R$$

$$\frac{\Sigma : \nabla \models \text{И}\mathsf{a}.C \quad \Sigma \# \mathsf{a} : \Delta; \nabla, C \Longrightarrow G}{\Sigma : \Delta; \nabla \Longrightarrow \text{И}\mathsf{a}{:}\nu.G}\, \text{И}R$$

$$\frac{\Sigma : \Delta; \nabla \xrightarrow{D} A \quad D \in \Delta}{\Sigma : \Delta; \nabla \Longrightarrow A}\, sel$$

Atomic focused proofs

$$\frac{\Sigma : \nabla \vDash A' \sim A}{\Sigma : \Delta ; \nabla \xrightarrow{A'} A}\, hyp \qquad \frac{\Sigma : \Delta ; \nabla \xrightarrow{D_i} A}{\Sigma : \Delta ; \nabla \xrightarrow{D_1 \wedge D_2} A}\, \wedge L_i$$

$$\frac{\Sigma : \Delta ; \nabla \xrightarrow{D} A \qquad \Sigma : \Delta ; \nabla \Longrightarrow G}{\Sigma : \Delta ; \nabla \xrightarrow{G \supset D} A}\, \supset L$$

$$\frac{\Sigma : \nabla \vDash \exists X.C \qquad \Sigma, X : \Delta ; \nabla, C \xrightarrow{D} A}{\Sigma : \Delta ; \nabla \xrightarrow{\forall X{:}\sigma.D} A}\, \forall L$$

$$\frac{\Sigma : \nabla \vDash И\mathsf{a}.C \qquad \Sigma \# \mathsf{a} : \Delta ; \nabla, C \xrightarrow{D} A}{\Sigma : \Delta ; \nabla \xrightarrow{И\mathsf{a}{:}\nu.D} A}\, ИL$$

Comments

- Most connective rules standard.

- Quantifier rules use *constraints* rather than *substitutions*. More on this later.

- Atomic formula rule (hyp) uses relation $A \sim A'$ rather than $A \approx A'$. Technically,

$$\Sigma : \nabla \vDash A \sim A' \iff \exists \pi . \Sigma : \nabla \vDash \pi \cdot A \approx A'$$

  More on this later.

Residuated proofs

- Define a slight variant of proof theory that *computes* a sufficient constraint or goal
- $\Sigma : \Delta \Longrightarrow G \setminus C$: given program Δ, G reduces to residual constraint C
- $\Sigma : \Delta \xrightarrow{D} A \setminus G$: atomic formula A reduces against focused program clause D to subgoal G
- Rules not shown, straightforward.

Operational semantics

- Similar to [Darlington and Guo 1994]'s operational semantics

$$
\begin{array}{lll}
(B) & \Sigma\langle A, \Gamma \mid \nabla\rangle & \longrightarrow \quad \Sigma\langle G, \Gamma \mid \nabla\rangle \\
& (\text{if } \exists D \in \Delta.\Sigma : \Delta \xrightarrow{D} A \setminus G) & \\
(C) & \Sigma\langle C, \Gamma \mid \nabla\rangle & \longrightarrow \quad \Sigma\langle \Gamma \mid \nabla, C\rangle \\
& (\nabla, C \text{ consistent}) & \\
(\top) & \Sigma\langle \top, \Gamma \mid \nabla\rangle & \longrightarrow \quad \Sigma\langle \Gamma \mid \nabla\rangle \\
(\wedge) & \Sigma\langle G_1 \wedge G_2, \Gamma \mid \nabla\rangle & \longrightarrow \quad \Sigma\langle G_1, G_2, \Gamma \mid \nabla\rangle \\
(\vee_i) & \Sigma\langle G_1 \vee G_2, \Gamma \mid \nabla\rangle & \longrightarrow \quad \Sigma\langle G_i, \Gamma \mid \nabla\rangle \\
(\exists) & \Sigma\langle \exists X{:}\sigma.G, \Gamma \mid \nabla\rangle & \longrightarrow \quad \Sigma, X{:}\sigma\langle G, \Gamma \mid \nabla\rangle \\
(\text{И}) & \Sigma\langle \text{И}\mathsf{a}{:}\nu.G, \Gamma \mid \nabla\rangle & \longrightarrow \quad \Sigma\#\mathsf{a}{:}\nu\langle G, \Gamma \mid \nabla\rangle
\end{array}
$$

Most rules standard.

Key results

- Least Herbrand models of Δ and least fixed points of T_Δ exist and equal.

- Proof theoretic semantics sound and (weakly) complete wrt model theoretic semantics.

- Operational semantics sound and complete wrt proof theory.

- Spared details, outline in paper, full version forthcoming.

Freshness rule

- Previous proof theories for NL had a "freshness" rule.

$$\frac{\Sigma \# \mathsf{a} : \Gamma \Rightarrow \phi}{\Sigma : \Gamma \Rightarrow \phi} F \quad (\mathsf{a} \notin FN(\Sigma, \Gamma, \phi))$$

- Complicates the proof theory since not goal-directed & can't be permuted past $\exists R$. For example,

$$\cfrac{\cfrac{\cfrac{\cfrac{\vdots}{\mathsf{a}\#\mathsf{b} : \cdot \Rightarrow \mathsf{a} \# \mathsf{b}}}{\mathsf{a}\#\mathsf{b} : \cdot \Rightarrow \exists X.\mathsf{a} \# X} \exists R}{\mathsf{a} : \cdot \Rightarrow \exists X.\mathsf{a} \# X} F}{\cdot : \cdot \Rightarrow \text{И}\mathsf{a}.\exists X.\mathsf{a} \# X} \text{И}R$$

Previous solution

- Previous solution [Gabbay & C 2004]: Change definition of uniform proof

- "Bake in" applications of freshness rule to $\exists R$

$$\frac{\Sigma \# \vec{\mathsf{a}} \vdash t : \tau \quad \Sigma \# \vec{\mathsf{a}} : \Gamma \Rightarrow G[t/X]}{\Sigma : \Gamma \Rightarrow \exists X^{\tau}.G} \exists R^*$$

- Messy (so hard to analyze), worse, unclear how to implement!

New solution

- Insight: $\exists X.G$ may hold only for X mentioning new names, but *we don't need to know them in the proof*

- New solution: Use constraints instead of substitutions in quantifier rules

$$\frac{\Sigma : \nabla \vDash \exists X.C \quad \Sigma, X : \Delta; \nabla, C \Longrightarrow G}{\Sigma : \Delta; \nabla \Longrightarrow \exists X.G} \exists R$$

- This pushes freshness reasoning into constraint solving; proof search reduces to constraint solving in a "goal-directed" way

New solution

- Using constraint-based rules, can for example derive

$$\dfrac{\cdot : \top \vDash \text{И}\mathsf{a}.\top \qquad \dfrac{\mathsf{a} : \top \vDash \exists X.\mathsf{a} \mathrel{\#} X \qquad \overline{\mathsf{a} : \mathsf{a} \mathrel{\#} X \Rightarrow \mathsf{a} \mathrel{\#} X}\ hyp}{\mathsf{a} : \top \Rightarrow \exists X.\mathsf{a} \mathrel{\#} X}\ \exists R}{\cdot : \top \Rightarrow \text{И}\mathsf{a}.\exists X.\mathsf{a} \mathrel{\#} X}\ \text{И}R$$

since $\vDash \exists X.\mathsf{a} \mathrel{\#} X$ holds.

- Such constraint-based quantifier rules were introduced earlier to define uniform proofs for CLP [Darlington and Guo 1994, Leach et al. 2001].

An application

- We used the cleaner proof-theoretic semantics to prove the correctness of program rewriting rules such as

$$G \supset \forall X.D \rightsquigarrow \forall X.(G \supset D) \quad (X \notin FV(G))$$

$$G \supset \text{И}\mathsf{a}.D \rightsquigarrow \text{И}\mathsf{a}.(G \supset D) \quad (\mathsf{a} \notin supp(G))$$

- These can be used to "elaborate" all program clauses to the form

$$\text{И}\vec{\mathsf{a}}\forall\vec{X}.G \supset A$$

Another application

- Resolution based on equality (rather than $\sim$) sometimes makes constraint solving more tractable

$$\frac{\Sigma : \nabla \vDash A \approx A'}{\Sigma : \Delta ; \nabla \xrightarrow{A} A'}\, hyp_{\approx}$$

- Showed that $\approx$-resolution is complete for "И-clause-free" programs (in which И only appears in *goal subformulas*)

- Simple proof transformation argument (compares favorably with previous work [Urban and C 2005])

Related work

- Higher-order LP and uniform proofs [Miller et al. 1991]
- Constraint LP semantics
  - [Jaffar et al. 1998]: denotational and operational
  - [Darlington and Guo 1994, Leach, Nieva, Rodrigues-Artalejo 2001]: proof-theoretic and operational
- Miller's L_λ language
  - Seems related to И-clause-free fragment of NomLP

Future work

- Mode checking, additional optimizations
- Generalize semantics to arbitrary (nominal) constraint domains
- Incorporate nominal constraint solving into existing CLP system?
- Relate to L_λ?

Conclusions

- Nominal logic programming is a conceptually simple extension to plain FO (C)LP supporting name-binding

- This work consolidates and improves prior treatments of its semantics

  - Key issues: rules for quantifiers, freshness

- Provides a solid foundation for verifying program transformations, interpretation, compilation.