Semantic nominal terms

Murdoch J. Gabbay, www.gabbay.org.uk

Sunday 22 March 2009

Joint work with Dominic Mulligan

Thanks to the TAASN organisers

Semantics nominal terms

There are two distinct ideas in this paper:

1. Semantics for nominal terms unknowns.
2. Reconciling capturing and capture-avoiding substitution.

I'll motivate each, in turn.

I welcome questions.

Semantics for nominal terms

I'll assume you're familiar with nominal terms' syntax:

$$t ::= a \mid \pi \cdot X \mid [a]X \mid \mathsf{f}(t, \ldots, t)$$

- a is an atom (a level 1/object-level variable).
- X is an unknown (a level 2/meta-level variable).

I don't like to call X a 'meta-level variable' because it's not. It's a variable symbol in nominal terms' syntax.

Yet, a and X models object- and meta-level variables. For example, substitution for X captures in $[a]X$.

$$([a]X)[X := a] \equiv [a]a \quad \text{models} \quad \text{'set } s \text{ to be } x \text{ in } \lambda x.s\text{'}.$$

Nominal unknowns

So, are nominal unknowns variables? If so, there should be a valuation ρ mapping them to denotations. Thus:

- In 'capture-avoiding substitution as a nominal algebra' and 'the lambda-calculus is nominal algebraic' we prove ω-completeness. Unknowns map to terms.
- In nominal algebra and nominal equational logic, unknowns map to elements of nominal sets.

This is standard: '$[\![X]\!]_\rho = \rho(X)$'.

Nominal unknowns semantics

I've never been entirely happy with this. Why?

We have a model of α-equivalence given by my thesis, in FM set theory. Call it 'nominal abstract syntax'.

Nominal unknowns X have infinite support, in the sense that $\pi \cdot X \not\equiv X$ if π is not the identity.

They don't fit in that model! a maps to a, but there's no FM set we can point to and say 'this is X'.

Nominal unknowns semantics

I felt that we were betraying our own model. Having added atoms to our universe, do we have to introduce another level of variable (unknowns), that cannot be accommodated in it?

Or can we use our imagination a bit more, and come up with something else?

Nominal unknowns semantics

Here's another question:

Nominal abstract syntax is in some sense the canonical model for raw syntax up to α-equivalence. What is a canonical model of nominal terms?

'Raw syntax'	$\Leftrightarrow$	Nominal abstract syntax $\cong$ Raw syntax$/\alpha$
Nominal terms	$\Leftrightarrow$	Functions from valuations to denotational elements?

The creature at the bottom right is the 'odd one out'.

I think this is a right answer:

Nominal unknowns correspond with infinite lists of distinct atoms.

Identify X with an infinite list $(a, b, c, d, \ldots)$ (fancy name: an ω-tuple).

Why lists? Why not sets?

One property we want is completeness:

if $\pi \neq \pi'$ then $[\![\pi \cdot X]\!] \neq [\![\pi' \cdot X]\!]$.

We also want compositionality: $[\![\pi \cdot X]\!] = \pi \cdot [\![X]\!]$.

$\pi \cdot (a, b, c, d, \ldots) = (\pi(a), \pi(b), \pi(c), \pi(d), \ldots)$.

So we get completeness if we use lists. If we were to use sets, this would not happen: $(a\ b) \cdot \{a, b, c, d, \ldots\} = \{a, b, c, d, \ldots\}$.

Semantic nominal terms

Why infinite lists?

So that we can recover X from $[\![\pi \cdot X]\!]$, for any finitely-supported π. We examine the 'asymptotic behaviour' to derive $[\![X]\!]$.

$(a, b, c, d, \ldots)$ and $(\pi(a), \pi(b), \pi(c), \pi(d), \ldots)$ are 'eventually the same', for finitely-supported π.

Another nice feature of using lists is that we have infinitely many lists with the same support, but not related by a finitely-supported permutation. Consider

$$(a, b, c, d, \ldots) \qquad \text{and} \qquad (b, a, d, c, \ldots).$$

There is no π such that $\pi \cdot (a, b, c, d, \ldots) = (b, a, d, c, \ldots)$. This lets us model $X, Y, Z, \ldots$ as infinitely many semantic nominal unknowns, with the same support, if we want.

Atoms-abstraction

Atoms-abstraction works fine. We can define $[a](a, b, c, d, \ldots)$ and it has exactly the right behaviour.

As a nice bonus, not yet fully explored, we can also define $[X]t$ (abstraction over infinite lists of atoms).

In short, the answer ‘unknowns are infinite lists of distinct atoms’ gives level 2 variables the right behaviour and puts them in our sets model shoulder to shoulder with atoms.

Questions?

Next problem: capture-avoiding vs. capturing substitution

A funny thing happened to me on the way to the conference!

I substituted x to y in $\lambda y.x$ and I got $\lambda y'.y$.

I instantiated t to y in $\lambda y.t$ and I got $\lambda y.y$.

Why?

Why does y turn into y' in one case, and not in the other?

Yes, I know that one of them is capture-avoiding and the other is capturing. I know it models informal practice! But why? What's the underlying structure here?

Next problem: capture-avoiding vs. capturing substitution

One answer is:

There are two kinds of substitution:

- Capture-avoiding substitution (this is appropriate for level 1 binders interacting with level 1 substitutions).
- Capturing substitution (this is appropriate for level 1 binders interacting with level 2 substitutions).

See nominal terms, and also ‘hierarchical nominal rewriting’ and ‘the lambda-context calculus’ which extend this idea to ω many levels.

But now, we have explained level 2 variables as ‘level 1 variables + infinite lists’. So I’ll propose a different answer.

My answer will also remove the side-condition in nominal terms ‘$\Delta \vdash \nabla\sigma$’, i.e. ‘the substitution has to respect the freshness context’.

Permissive nominal terms

Suppose $S \subseteq \mathbb{A}$ is a set of atoms. Suppose π is a permutation.

Specify π/S by:

- If $a \in S$ then $(\pi/S)(a) = \pi(a)$.
- If $a \notin S$ and $\pi^{-1}(a) \notin S$ then $(\pi/S)(a) = a$.
- If $a \notin S$ and $\pi^{-1}(a) \in S$ then $(\pi/S)(a) = a_n$, where $a_n \in S$ and there exist $a_1, \ldots, a_{n-1}$ such that $\pi(a) = a_1$ and $a_i \notin S$ and $a_{i+1} = \pi(a_i)$ for $i < n$.

Substitution

Suppose

$$\pi = (a\ b\ c\ d\ e)(f\ g)$$

(π maps a to b to c to d to e to a, and f to g to f).

$$\pi/\{a\} = (a\ b) \quad \pi/\{a,b\} = (a\ b\ c)$$
$$\pi/\{a,c\} = (a\ b\ c\ d) \quad \pi/\{a,f\} = (a\ b)(f\ g).$$

Note that $\pi|_S = (\pi/S)|_S$. (-$|_S$ is 'restrict domain to S'.)

π/S is the permutation such that $\{a \mid (\pi/S)(a) \neq a\}$ is minimal, such that $(\pi/S)(a) = \pi(a)$ for all $a \in S$.

It's the least permutation agreeing with π on S.

Substitution

Substitution can be defined by:

$$a\sigma = a \qquad (\pi \cdot X)\sigma = (\pi / \mathsf{supp}(X)) \cdot \sigma(X)$$
$$\mathsf{f}(r_1, \ldots, r_n)\sigma = \mathsf{f}(r_1\sigma, \ldots, r_n\sigma)$$
$$([a]r)\sigma = [b](b\ a) \cdot (r\sigma) \quad (b\#r,\ b\#r\sigma)$$

We choose any fresh b (it doesn't matter).

Actually, this is a small lie, but it's more convincing than the correct definition.

Now, you're all going to demand to see it, right?

Substitution

Write $\mathsf{Orb}(X)$ for $\{\pi \cdot X \mid \text{all } \pi\}$; call this the (permutation) orbit of X.

A substitution is a function from semantic unknowns to terms such that for each orbit $\mathsf{Orb}(X)$ there exists some representative $X \in \mathsf{Orb}(X)$ such that $\sigma(\pi \cdot X) = (\pi/\mathsf{supp}(X)) \cdot \sigma(X)$ for all π.

$\sigma, \sigma', \sigma'', \ldots$ will range over substitutions.

Write $[X \mapsto t]$ for the function mapping $\pi \cdot X$ to $(\pi/\mathsf{supp}(X)) \cdot t$, and all other Y to Y.

Substitution

Now define a substitution action by:

$$a\sigma = a \qquad X\sigma = \sigma(X) \qquad \mathsf{f}(r_1, \ldots, r_n)\sigma = \mathsf{f}(r_1\sigma, \ldots, r_n\sigma)$$

$$([a]r)\sigma = [b](b\ a) \cdot (r\sigma) \quad (b\#r,\ b\#r\sigma)$$

Substitution

A problem with semantic nominal terms is that, if we stick with the strictly capturing substitution, then we can only admit σ such that $\mathsf{supp}(\sigma(X)) \subseteq \mathsf{supp}(X)$ always. This corresponds with the condition '$\Delta \vdash \nabla\sigma$' from nominal terms' theory.

The capture-avoiding substitution above removes that condition. So now X represents any term; X can be substituted for any term.

Substitution

$\mathsf{supp}(X)$ represents the namespace of ‘known atoms’, for which substitution is capturing. $\mathbb{A} \setminus \mathsf{supp}(X)$ represents the namespace of ‘atoms generated fresh later’, for which substitution is capture-avoiding.

If you base nominal rewriting on semantic nominal terms using this substitution action, then $X \longrightarrow X$ represents any (trivial) rewrite, and not just rewrites for terms with free atoms in $\mathsf{supp}(X)$.

Some examples

Suppose that $a, b \in \mathsf{supp}(X)$ and $c, d \notin \mathsf{supp}(X)$. Then

$$\begin{aligned} X[X \mapsto a] &= a \\ ([a]X)[X \mapsto a] &= [b]b = [a]a \\ ([c]X)[X \mapsto c] &= [d]c \\ ([a][c]X)[X \mapsto (a,c)] &= [a][d](a,c). \end{aligned}$$

See how a does not avoid capture, and c does? Remember my question 'Why'? Answer: because $a \in \mathsf{supp}(X)$ and $c \notin \mathsf{supp}(X)$.

This substitution action can be applied to single atoms rather than infinite lists. Then, it specialises to capture-avoiding substitution. Morally, the only 'capturing' substitution possible in this case, is $[a := a]$.

Conclusions

I have proposed:

- Nominal terms unknowns can be understood as ‘atoms + infinite lists’.
- Nominal terms style capturing substitution, and first-order syntax style capture-avoiding substitution, are compatible, and I’ve shown how to do it.

This all has a simple concrete model in a cumulative hierarchy with atoms.

Further work

Explore $[X]t$ — model of 'α-abstraction by meta-variables'.

Admit lists of atoms with finitely many non-atomic elements. For example, $(a, 2, 3, d, e, f, \ldots)$. Nice model of unknowns with substitutions.

Use X to model 'holes' in things like: incomplete derivations, manipulating trees with hidden labels, π-calculus contexts, and so on.