Names, computations, logics: the hole story

Murdoch J. Gabbay, Heriot-Watt University, Scotland

http://www.gabbay.org.uk

ICMS, Edinburgh, UK
Sunday, 27 May 2007

Thanks

It’s a pleasure to be here.

Thanks to the organisers for a lovely dinner last night.

Not maths

I want to talk about some ideas I've been working on.

An overview.

Working alone is crap. I'm a research tart. I'll sleep on a maths problem with anybody.

If you see something you like, let's do it.

Timeline

Fraenkel-Mostowski sets.

The anti-Dale is born!

As James Cheney said this morning, names are real semantic entities.

Names (free variables, in a sense) are the basis of the FM sets universe; $V_0 = \mathbb{A}$, $V_1 = \mathcal{P}V_0$, ... Furthermore you can implement binding by a concrete equivalence class of sets: names are not created by binding, binding emerges from names (and concrete sets structure).

$$[a](a,b) = \{(a,(a,b)),\ (c,(c,b)),\ (d,(d,b)),\ \ldots\}.$$

Significantly 'different' in the sense that atoms a, freshness $a\#x$, and abstraction assume a mathematical foundational status.

Timeline

Nominal terms and unification.

α-equivalence in the presence of meta-variables.

$$b\#X \vdash [b]((b\ a) \cdot X) = [a]X.$$

You still have unification and it's more like first-order unification than higher-order unification.

Note the capturing substitution. Also significantly 'different'; much maths is committed to capture-avoiding substitution.

Timeline

Nominal rewriting.

$(\lambda[a]X)Y \rightarrow X[a\mapsto Y]$.

Sugar for $(\lambda[a]X)Y \rightarrow \mathsf{sub}([a]X, Y)$. sub has its own rewrite rules.

There is still the capturing substitution. But notice something else; λ is a lambda-abstraction (we have β-reduction), but it is also just another term-former.

The reason this can happen is that, in some sense, atoms exist in the denotation for λ to bind.

What denotation? FM sets of course, but more refined semantics are possible. More on that later.

Timeline

Nominal algebra. (Just ‘undirected nominal rewriting’.)

Axiomatisation of substitution. A notion of substitution independent of β-equivalence (but which does many of the same things).

Axioms of substitution

$$
\begin{array}{rrrcl}
(\mathbf{var}{\mapsto}) & & \vdash & a[a{\mapsto}T] & = & T \\
(\#{\mapsto}) & a\#X & \vdash & X[a{\mapsto}T] & = & X \\
(\mathbf{f}{\mapsto}) & & \vdash & \mathsf{f}(X_1,\ldots,X_n)[a{\mapsto}T] & = & \mathsf{f}(X_1[a{\mapsto}T],\ldots,X_n[a{\mapsto}T]) \\
(\mathbf{abs}{\mapsto}) & b\#T & \vdash & ([b]U)[a{\mapsto}T] & = & [b](U[a{\mapsto}T]) \\
(\mathbf{ren}{\mapsto}) & b\#X & \vdash & X[a{\mapsto}b] & = & (b\ a)\cdot X
\end{array}
$$

Possibly useful as a model for non-standard programming constructs, e.g. calculi of pattern-matching, or pointers.

Timeline

Substitution action on FM sets.

The FM sets universe is itself a model of substitution. That's incredible.

Slogan: a variable is a name with a substitution action — denotationally.

FM atoms are more than a denotational model of names; they are also (with the substitution action) a denotational model of variables. I'll say that again:

- (Analogy:) 'Function' can be viewed as 'graph' and we can build a model in sets using Collection and Pairset and stuff.
- 'Variable' can be viewed as 'name with a substitution action' and we can build a a model in (FM) sets.

Substituting atoms in small sets

If Z is finite just set $Z[a \mapsto x] = \{z[a \mapsto x] \mid z \in Z\}$. Easy.

But this fails for $\mathbb{A}$, because $\mathbb{A}[a \mapsto b] = \mathbb{A} \setminus \{a\}$ violates $(\# \mapsto)$.

Likewise if $(\mathbb{A} \setminus \{a\})[a \mapsto b] = \mathbb{A} \setminus \{a\}$ then $a\#(\mathbb{A} \setminus \{a\})[a \mapsto b]$ fails. But we expect

$$\mathsf{supp}(z[a \mapsto x]) \subseteq \mathsf{supp}(z)\{a \mapsto \mathsf{supp}(x)\}$$

where

$$S\{a \mapsto T\} = S \quad \text{if } a \notin S \text{ and}$$
$$S\{a \mapsto T\} = (S \setminus \{a\}) \cup T \quad \text{otherwise.}$$

Problem: how do we substitute for the a that is not there?

The key idea

Suppose $A \subseteq \mathbb{A}$ is finite. Write

$$\mathsf{fix}(A) = \{\pi \mid \forall a \in A.\pi(a) = a\}.$$

$\mathsf{fix}(A)$ is the set of permutations π that fix A pointwise.

Write

$$z\|_A = \{\pi z \mid \pi \in \mathsf{fix}(A)\}.$$

For example $\mathbb{A} \setminus \{a\} = b\|_{\{a\}}$.

The key idea

$\mathbb{A} \setminus \{a\} = b\|_{\{a\}}$.

Define

$$(z\|_S)[a\mapsto x] = (z[a\mapsto x])\|_{S\{a\mapsto \mathsf{supp}(x)\}}$$

subject to a bundle of capture-avoidance conditions.

The key idea

$$\mathbb{A} \setminus \{a\} = b\|_{\{a\}}.$$

Then

$$\begin{aligned}(\mathbb{A} \setminus \{a\})[a\mapsto x] &= b[a\mapsto x]\|_{\mathsf{supp}(x)} \\ &= \mathbb{A}.\end{aligned}$$

Timeline

Lambda context calculus (LCC).

An idea I've been kicking around since my time in Cambridge. Nominal terms have meta-variables.

I say these meta-variables are not just a convenience. They are real — just like any other kind of variable.

A bit of motivation

$$\cfrac{\cfrac{\cfrac{A\Rightarrow B\Rightarrow C \quad [A]^i}{B\Rightarrow C} \qquad \cfrac{?}{B}}{C}}{A\Rightarrow C}\; i$$

$$\cfrac{\cfrac{\cfrac{A\Rightarrow B\Rightarrow C \quad [A]^i}{B\Rightarrow C} \qquad \cfrac{A\Rightarrow B \quad [A]^i}{B}}{C}}{A\Rightarrow C}\; i$$

Capturing substitution necessary for Curry-Howard with incomplete proofs.

(Example borrowed from [Jojgov, TYPES 2002]).

LCC syntax

$s, t ::= a_i \mid tt \mid \lambda a_i.t \mid t[a_i \mapsto t]$.

a_i has level i.

Define free variables as usual, e.g.

$$\mathsf{fv}(\lambda a_i.t) = \mathsf{fv}(t) \setminus \{a_i\}.$$

Let $\mathsf{level}(t)$ be the maximum level of any variable in t, free or bound.

Write $a_i \# S$ when $a_i \notin S$ and if $c_k \in S$ then $k \leq i$. So $a_i \# \{c_i\}$ and not $a_i \# \{b_j\}$ where $j > i$.

LCC reduction rules

(β)	$(\lambda a_i.s)t \longrightarrow s[a_i{\mapsto}t]$	
$(\sigma\mathbf{a})$	$a_i[a_i{\mapsto}t] \longrightarrow t$	
$(\sigma\mathsf{fv})$	$s[a_i{\mapsto}t] \longrightarrow s$	$a_i\#\mathsf{fv}(s)$
$(\sigma\mathbf{p})$	$(ss')[a_i{\mapsto}t] \longrightarrow (s[a_i{\mapsto}t])(s'[a_i{\mapsto}t])$	$\mathsf{level}(s, s', t) \leq i$
$(\sigma\sigma)$	$s[a_i{\mapsto}t][b_j{\mapsto}u] \longrightarrow s[b_j{\mapsto}u][a_i{\mapsto}t[b_j{\mapsto}u]]$	$i < j$
$(\sigma\lambda)$	$(\lambda a_i.s)[b_j{\mapsto}u] \longrightarrow \lambda a_i.(s[b_j{\mapsto}u])$	$i < j$
$(\sigma\lambda')$	$(\lambda a_i.s)[c_i{\mapsto}u] \longrightarrow \lambda a_i.(s[c_i{\mapsto}u])$	$a_i\#\mathsf{fv}(u)$

LCC

Build a model; whatever metavariables are, they should display these equalities.

Make a higher-order logic out of it; investigate derivation rules.

Use it to model stuff with links. For example, take $R = X[x{\mapsto}2][y{\mapsto}3]$ and reduce as follows:

$$
\begin{aligned}
(\lambda \mathcal{W}.\mathcal{W}[X{\mapsto}X[x{\mapsto}3]])\ R \quad &\xrightarrow{(\beta)} \quad \mathcal{W}[X{\mapsto}X[x{\mapsto}3]][\mathcal{W}{\mapsto}R] \\
&\xrightarrow{(\sigma\sigma),(\sigma\mathbf{a})}{}^{*} \quad R[X{\mapsto}X[x{\mapsto}3]] = X[x{\mapsto}2][y{\mapsto}3][X{\mapsto}X[x{\mapsto}3]] \\
&\xrightarrow{(\sigma\sigma)}{}^{*} \quad X[X{\mapsto}X[x{\mapsto}3]][x{\mapsto}2[X{\mapsto}X[x{\mapsto}3]]][y{\mapsto}3[X{\mapsto}X[x{\mapsto}3]]] \\
&\xrightarrow{(\sigma\mathbf{a}),(\sigma\mathbf{b})}{}^{*} \quad X[x{\mapsto}3][x{\mapsto}2][y{\mapsto}3].
\end{aligned}
$$

Other stuff

One-and-a-halfth order logic (stand by for two-and-a-halfth order logic; internalising the $a\#Z$, the $\vdash$, and the $\forall Z, X$ implicit in something like $a\#Z \vdash Z[a{\mapsto}X]$).

a-logic (predicate ‘isvar’ identifies a variable symbol in first-order logic; good for PROLOG).

Hierarchical nominal rewriting (hierarchy of variables; needs a model).

Types for nominal terms (semantics without denotations).

Slogans:

Variables are denotational entities.

Variables have internal structure. They are non-trivial mathematical entities.

There are many name-like entities out there; pointers, variables, patterns, context holes. There's no shortage of potential applications. Benton and Leperchey apply the FM/nominal 'package' to reason on pointers, and develop it further.

Nominal techniques are not solely atoms and inductive datatypes.

Names

Some referees seem to get very unhappy about this. I don't see why.

Gilles Dowek used a new truth-value in his talk. Nobody made a fuss:

- . . . but we already have two truth-values, why do we need a third?
- . . . hey, I can encode truth-values in numbers anyway!
- . . . it's all a special case of fuzzy logic!
- . . . truth is a purely meta-level assertion about denotation but has no denotation itself.
- . . . the author does not make clear in the paper what 'truth' is.

Let's do a mathematics of names.