Nominal: an Overview

Murdoch J Gabbay

UPM, Madrid, Spain 30/3/2006

Thanks to Julio Mariño and Angel Herrenz

Purpose of this talk . . .

. . . is to give some idea of what I'm doing, without going into technical detail. I will follow a more-or-less chronological framework.

Thesis

Invented Fraenkel-Mostowski set theory (FM sets, aka Nominal Sets), the И quantifier, and inductive-datatypes-with-binding.

Also implemented FM sets in Isabelle and designed one version of what later became FreshML.

FM sets

FM sets is a variant of Zermelo-Fraenkel set theory (ZF sets) with atoms (urelemente).

ZF sets is the dominant notion of set, used in foundations of mathematics (apologies to Quine's New Foundations!).

FM sets

ZF sets with atoms (ZFA) admits sets ‘from outside’, such as ‘the set of greeks’ or ‘the set of mortals’ without insisting these be modelled *a partir de* the empty set.

E.g. not all greeks have to look like $\{\{\},\{\{\}\},\{\{\},\{\{\}\}\}\}$; we admit $\{$Socrates$\}$ where Socrates is an urelement.

These atoms are collected and form a set of atoms $\mathbb{A}$.

Fraenkel-Mostowski sets. . .

. . . enriches ZFA with an axiom saying there are infinitely many atoms, and with the ‘fresh axiom’

$$\forall z.\ \text{И}a.\ \text{И}b.\ (b\ a) \cdot z = z$$

Here a is an atom and z is any set.

$\text{И}a.\ \phi(a)$ means:

- Perhaps $\neg\phi(a)$ for some finite set S.
- However, $\phi(a)$ holds for all atoms $a \notin S$.

The fresh axiom

I'll say what swapping is in a moment.

$\text{И}a.\ \text{И}b.\ (b\ a)\cdot z$ unpacks as

$$\exists S_a, S_b.\ S_a \text{ finite} \wedge S_b \text{ finite} \wedge \forall a \in \mathbb{A} \setminus S_a.\ \forall b \in \mathbb{A} \setminus S_b.\ (b\ a)\cdot z = z.$$

Since $S_a \vee S_b$ is finite, this simplifies to:

$$\exists S.\ S \text{ finite} \wedge \forall a, b \in (\mathbb{A} \setminus S).\ (b\ a)\cdot z = z$$

Swapping

In FM a z is either an atom, or a set $\{z' \mid z' \in z\}$. So...

$$(b\ a) \cdot a = b$$
$$(b\ a) \cdot b = a$$
$$(b\ a) \cdot c = c$$
$$(b\ a) \cdot z = \{(b\ a) \cdot z' \mid z' \in z\}.$$

As a picture this is very easy to understand: swapping swaps atoms in sets, wherever (and however deep) they may appear in the term.

Swapping is bijective, so if $(b\ a) \cdot z \neq z$ it must be that z mentions b in some 'distinguished' way in its structure. The finiteness axiom generalises 'finite variable support' to sets.

Why is this useful?

Aside from the beauty of this idea . . .

Note that $(b\ a) \cdot x = (b\ a) \cdot y$ if and only if $x = y$.

Note that $(b\ a) \cdot x \in (b\ a) \cdot y$ if and only if $x \in y$.

Note that $(b\ a) \cdot \mathbb{A} = \mathbb{A}$.

Recall that FM sets is merely a theory of first-order logic in the language $(=: 2,\ \in: 2,\ \mathbb{A} : 0)$.

So we have the principle of equivariance:

$$\Phi(x_1, \ldots, x_n) \Leftrightarrow \Phi((b\ a) \cdot x_1, \ldots, (b\ a) \cdot x_n).$$

Problem with inductive principles

This solved a big problem in the formal (e.g. mechanised, in Isabelle) theory of inductive datatypes.

Suppose you have some inductive hypothesis
$\Phi(z) = \forall b, a, x.\ \phi(b, z, a, x)$ where ϕ is

$$a \in \mathbb{A} \wedge b \in \mathbb{A} \wedge x \in \Lambda \wedge b \notin fv(z) \wedge b \notin fv(x) \Rightarrow b \notin fv(z[a \mapsto x]).$$

Here Λ is some sets-based implementation of a datatype such as

$$t ::= a \mid tt \mid \lambda a.t.$$

Now you want to prove $\Phi(z)$ implies $\Phi(\lambda b.z)$.

Problem with inductive principles

Unfortunately $b \in x$ so your definition of substitution tells you

$$(\lambda b.z)[a\mapsto x] = \lambda b'.((b'b)\cdot z)[a\mapsto x]$$

for some fixed but arbitrary $b' \notin fv(z,x) \cup \{a,b\}$.

($(b'\ b)\cdot z$ equals $z[b\mapsto b']$, but is more useful, see below.)

So you have $\Phi(z)$ — not $\Phi((b'\ b)\cdot z)$.

¡Leces!

Ah — but you do have $\Phi((b'\ b)\cdot z)$, because of equivariance.

¡Muy bien!

Jamie $\mapsto$ Doctor Jamie.

Abstraction sets

Another thing that came out of FM sets was a NEW model of abstraction, which is to α-equivalence as functional abstraction is to β-equivalence.

Just as a set Y^X is populated by graphs of functions from elements of X to elements of Y, so . . .

. . . a set $[\mathbb{A}]X$ is populated by elements $[a]x$ for $a \in \mathbb{A}$ (atoms) and $x \in X$, defined by

$$[a]x = \{(b, (b\ a) \cdot x) \quad | \quad b\#x \vee b = a\}$$

Here $b\#x$ when $\text{И}b'.\ (b'\ b) \cdot x = x$ is a notion of 'fresh for'.

Fresh for

The this idea is not something I can do justice to in this talk.

Just a few examples:

- $b\#\mathbb{A}$ since $(b'\ b) \cdot \mathbb{A} = \mathbb{A}$, since swapping is bijective on atoms.
- $b\#\{a\}$ since for $S = \{a\}$ and $b' \not\in S$ we have $(b'\ b) \cdot \{a\} = \{a\}$.
- $\neg(a\#\{a\})$.
- $\neg(b\#\mathbb{A}\backslash\{b\})$ since $(b'\ b) \cdot (\mathbb{A}\backslash\{b\}) = \mathbb{A}\backslash\{b'\}$.

So 'fresh for' does not imply 'not set-included in'. Corresponds more to 'does not occur in any distinguished way in'.

The basic theorem of abstraction sets:

$$[a]x = \{(b, (b\ a) \cdot x) \quad | \quad b\#x \vee b = a\}$$

In fact, $b\#[a]x$ if and only if $b\#x$, and $a\#[a]x$.

This exactly replicates the behaviour of $b \notin fv(z)$, with $[a]x$ corresponding to a binder.

So we can build Λ more compactly as

$$t ::= a \quad | \quad tt \quad | \quad \lambda[a]t.$$

Giving not only equivariance, but a true inductively defined datatype up to binding.

Nominal terms

But let's talk about something else now. We have this semantic notion of abstraction. Let's define a language for talking about it:

$$t ::= a \quad | \quad \pi \cdot X \quad | \quad \mathsf{f}t \quad | \quad (t, t) \quad | \quad [a]t.$$

These are nominal terms. Note that they have a notion of abstraction $[a]t$, with semantics which are not functional.

Nominal terms

$$t ::= a \quad | \quad \pi \cdot X \quad | \quad \mathsf{f}t \quad | \quad (t, t) \quad | \quad [a]t.$$

$\mathsf{f}t$ is a term-former.

X is an 'unknown element'.

$\pi \cdot X$ has a moderating permutation; π is a (finite) permutation on atoms.

To express the capture-avoiding aspects of syntax with variable names (and its abstract nominal version) we introduce an intentional notion of freshness:

$$\frac{a\#t}{a\#\mathsf{f}t} \qquad \frac{a\#t \; a\#t'}{a\#(t,t')} \qquad \frac{a\#t}{a\#[b]t} \qquad \frac{}{a\#b} \qquad \frac{}{a\#[a]t} \qquad \frac{\pi^{-1}(a)\#X}{a\#\pi\cdot X}$$

Then the core equality of nominal terms, can be written as

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

Believe it or not, this simple equality abstracts α-equivalence. That is, the least congruence containing this equality (also instantiating X) is a reasonable generalisation of α-equivalence to a syntax with unknowns X, Y, Z.

The key point of nominal terms

The key point is not that nominal terms have abstraction (and α-equivalence), but that they have abstraction in the presence of a kind of unknown which can be substituted for in a capturing manner.

For example, we can set X to be $[a]a$ in the core equality, and we obtain

$$[b]b = [a]a$$

which is what you'd expect ($a\#[a]a$ and $b\#[b]b$).

Advantages of nominal terms

$[\mathbb{A}]X$ has the same cardinality as X. Note that Y^X does not have the same cardinality as Y or X (in general).

This leads to good computational properties. For example, unification of nominal terms is decidable (higher-order unification is not).

See work on Nominal Unification with Urban and Pitts — also α-prolog by Urban and Cheney.

Advantages of nominal terms

Yet rewriting of nominal terms is just as expressive as higher-order rewriting, because we can express β-reduction as

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

Here we assume term-formers λ, app and sub with sugar

$$\mathsf{app}(t, u) = tu \qquad \mathsf{sub}([a]u, t) = u[a \mapsto t].$$

There's a body of work on Nominal Rewriting, with Fernández.

Equalities on nominal terms

My last ‘chunk’ of work was with Mathijssen in TU/e.

We developed the abstract theory of equality on nominal terms. That is, we developed Nominal Algebra. Nominal terms let us talk about abstraction, you see.

For example, we wrote some axioms for substitution as an abstract algebraic operation. These axioms turned out to be beautiful and subtle, with a really quite difficult meta-theory. It was not easy (but we managed it!) to prove them sound and complete for the canonical term model.

I am exploring abstract non-syntactic models of the theory. It turns out that just the abstract models of nominal terms raise significant mathematical questions.

One-and-a-halfth-order logic

We used this to give an algebraic axiomatisation of first-order logic, which Mathijssen wanted for mCRL2, and to develop one-and-a-halfth-order-logic, which is a sequent system for first-order logic, with first-class predicate unknowns.

So we can prove $\forall[a]\phi \Rightarrow \phi[a \mapsto X]$ where ϕ is an 'unknown predicate' and X is an 'unknown term'.

Future work

Poernomo is interested in using this as a model for contexts and software components.

I want to develop nominal terms as a programming language and logic, and extend them with a hierarchy of unknowns.

I want to explore the semantics of nominal terms, also up to theories in nominal algebra, also computation, e.g. unification.

Conclusion

This is a brief non-detailed overview of a large and growing body of work, by myself and others.

I believe there is something genuinely new and unexpected behind all this. We are uncovering The Truth bit by bit, but there is lots more.