

Nominal techniques: evolving the atom

A statement of position

Murdoch J. Gabbay, Heriot-Watt University, Scotland

STP, St Andrews, Scotland

13 October 2006

Thanks to Phil Trinder and Rob Rothenberg

Complexity

Much computing theory is driven by a need to control **complexity** in software design.

Here **compositionality** is a great help:

- Got n big problems of size u ?
- Turn them into $k.n$ problems of size u/k .

Say for our application a problem of size u is $o(u^2)$ hard to solve. Then we make the problem easier by a factor of

$$\frac{n.u^2}{k.n.(u/k)^2} = k.$$

λ -calculus composition

In order to split up a problem we must express it more-or-less formally.

A dominant semantics is the λ -calculus. The **only** way to build a big programs out of smaller ones in the λ -calculus is function application fx .

Therefore in this world

composition = **function application**.

We can **imagine** other compositions, e.g. contexts $C[t]$. However $C[-]$ is no λ -term, context application is no term-former — and denotational theories provide no native support.

Yet we often have cause to imagine compositions other than functional application. For example:

Composition problems: goto

`goto` jumps to labelled code.

Not directly compatible with function composition!

(Dijkstra says: use `while...do...` and `repeat...until.`)

(Parigot then Wadler says: this is classical logic. Btw I wrote a paper on that, on 'restart')

Composition problems: global state

Values are associated to variables by a heap.

Any part of a program can instantaneously change the heap, affecting evaluation program-wide.

Really bad for composition, and certainly incompatible with raw λ -calculus.

(Wadler says: use monads. True, but this only formalises the inherent sequentiality of global state.)

Composition problems: Montague semantics

Montague gave semantics to natural language ('dog', 'loves') as λ -terms.

This semantics gets into serious difficulties with 'he', because it may refer to a variable bound by an existential in a previous sentence, as in

A man was in the park.	$(\exists x.\text{man}(x) \wedge \text{inpark}(x)) \wedge$
A policeman saw him.	$\exists y.\text{policeman}(y) \wedge \text{saw}(y, x)$

Montague semantics for natural language get **bigger**, the **smaller** the fragments; the semantics of 'he' is really hard.

(Kempson et. al.: Dynamic syntax. Others: continuations, ...)

Naming problems: GPH

Glasgow Parallel Haskell (GPH) has `seq` and `par` to influence evaluation order.

$$x \text{ seq } (y \text{ par } z) = (x \text{ seq } y) \text{ par } (x \text{ seq } z)$$

$$x \text{ seq } (y \text{ seq } z) = (x \text{ seq } y) \text{ seq } z$$

They act on the **name** of a variable, rather than its value. If they acted on the value, there'd be no point in using them, would there?

Naming problems: dynamic binding

In dynamic binding code moves from scope to scope, as in

$$\lambda x.t \mid \lambda y.u \rightarrow \lambda x.u \mid \lambda y.t.$$

(Here \mid is some parallel composition, à la π -calculus.)

For example http port 80 always binds to a webserver — but **which** webserver.

In all cases the difficulty is with compositionality and names and movement of code across name-binders.

λ -calculus contexts no-no

λ -terms can emulate capturing substitution by type-lifting.

$\lambda x.f x$ emulates $\lambda x.-$.

Here f emulates $-$, but f has higher type than $-$.

$\lambda x.\lambda y.f x y$ emulates $\lambda x.\lambda y.-$. Here f has even higher type.

- It is not possible to substitute in a capturing manner parametrically over contexts.
- It is not possible to emulate $-$ uniformly over all contexts.
- It is not possible to write $C[t] \mid D[u] \rightarrow C[u] \mid D[t]$.

λ -calculus names no-no

λ -terms cannot reference variables by name.

For example in HOL $x \neq y$ means the **value** of x is not equal to the **value** of y .

It is simply impossible to access the names of x and y .

Pointers

Pointers are great. We can compare the values associated to pointers, or the pointers themselves:

$$!p \neq !q \quad p \neq q.$$

We can pass them through arbitrary contexts:

$$C[p] \mid D[q] \rightarrow C[q] \mid D[p].$$

Pointers have disadvantages:

- **Really** difficult mathematical theory.
- Not a priori abstractable, e.g. as $\lambda p.t$.

Nominal terms

Nominal terms are:

- New.
- (Partly) my work.
- The technology I am advocating.

They have some features of pointers, including liberal notions of compositionality — but they are mathematically better-behaved, like the λ -calculus.

Nominal terms

$$t ::= a \mid X \mid [a]t \mid f(t, \dots, t).$$

a is an **atom**, X an **unknown**, $[a]t$ an **abstraction**, and $f(t, \dots, t)$ a term-former applied to terms.

α -equivalence built into equality gives abstractions meaning, e.g.

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

Compositionality

- Substitution is capturing and this is parametric:

$$([a]X)[a/X] = [a]a \quad ([a][b]X)[a/X] = [a][b]a.$$

- Forms of functional application exist that capture in their argument, permitting function application to do more with names:

$$(\lambda X.(\lambda a.X))a \rightarrow (\lambda a.X)[a/X] = \lambda a.a.$$

- Names can be compared for equality as names:

$$a \neq b \text{ is true.}$$

Semantics

So we can develop logics, calculi, programming languages, and semantics, which directly represent things that we **really want to do** — and research is demonstrating that **really interesting** maths emerges.

- Axiomatisation of substitution: $a\#P \vdash P[a \mapsto Q] = P$.

Murdoch J. Gabbay, Aad Mathijssen

Capture-avoiding substitution as a nominal algebra, ICTAC'2006

- Logic with explicit meta-variables: $\forall[a]P$.

Murdoch J. Gabbay, Aad Mathijssen

One-and-a-halfth-order-logic, PPDP'2006

Semantics

- Fraenkel-Mostowski sets is a model of substitution: $z[a \mapsto x]$.
Murdoch J. Gabbay, Samuel Rota Buló, Andrea Marin
Almostfinishedwritingit, a good journal.
- Category of substitution is cartesian-closed: $f[a \mapsto g]$.
Murdoch J. Gabbay, Samuel Rota Buló, Andrea Marin
A nominal semantics for simple types, submitted.

There are lots more papers to be written.