This talk is not a talk about Stone duality for first-order logic

Murdoch J. Gabbay

Thanks to the organisers

December 20, 2011

About variables

I submitted what I consider to be one of my more important recent papers to Howard's festschrift. It establishes an amazing result, which is part of a broader 'nominal' programme which I have been pursuing for ten years.

The paper is also technically complex—not only the material itself is hard, but also it builds on hard material.

Don't worry. I'm not going to try to run through this all in twenty minutes.

Let me try to locate this paper in a broader 'nominal' context and explain why the overall programme is worthwhile, and how this paper fits in.

On the structure of FOL

The paper is about Stone duality for first-order logic. This extends Stone duality for Boolean logic with quantification $\forall a$.

First-order logic (FOL) is a (the) basic logic of the foundations of computer science. (Second-order and higher-order logic are arguably already ‘set theory in disguise’.)

A typical FOL sentence (Mary loves everybody):

$$\forall a. Loves(Mary, a)$$

Let’s look at how we give meaning to FOL.

Meaning for FOL

Here is an overview of denotations for first-order logic:

Predicate	*Boolean Alg.*	*Usual sem. FO logic*	*Nominal semantics*
$\phi \wedge \psi$	$[\![\phi]\!] \cap [\![\psi]\!]$	$\lambda\varsigma.([\![\phi]\!]_\varsigma \cap [\![\psi]\!]_\varsigma)$	$[\![\phi]\!] \cap [\![\psi]\!]$
$\neg\phi$	$\mathcal{U}\setminus[\![\phi]\!]$	$\lambda\varsigma.(\mathcal{U}\setminus[\![\phi]\!]_\varsigma)$	$\mathcal{U}\setminus[\![\phi]\!]$
$\forall a.\phi$	n/a	$\lambda\varsigma.\bigcap_{x\in\mathcal{U}}[\![\phi]\!]_{\varsigma[a:=x]}$	$\bigcap_{x\in\mathcal{U}}[\![\phi]\!][a\mapsto x]$

$\mathcal{U}$ is some 'domain of points' (e.g. $\mathcal{U}$ could be 'the set of all people').

Boolean algebra does not have a universal quantification, whence the 'n/a'. FOL adds quantification, allowing us to talk parametrically over $\mathcal{U}$.

This audience will already know how philosophically, socially, and commercially important FOL is.

Meaning for FOL

FOL semantics usually interprets $\forall$ using valuations ς. Sets structure interacts indirectly with $\forall$:

$$\lambda\varsigma.\bigcap_{x\in\mathcal{U}}[\![\phi]\!]_{\varsigma[a:=x]}$$

Valuations make a variable into something that refers to something in the denotation, but is not itself in the denotation as primitive.

This is a limited and restricted view.

On valuations

Valuations exclude variables from the denotation. ‘a’ does not appear as primitive in $\mathcal{U}$; only $\varsigma(a)$ does.

This made sense and was sufficient once upon a time, but not now.

For instance:

On referents

We want to reason on open terms, e.g. type equality in dependent type theory under a λ, or code generation.

In PROLOG-like languages, whether something is a variable is important for control. The semantics of control operators are sensitive to variables, rather than what they point to.

Computer science is full of structures in which referring is itself important information: π-calculus, dynamic linking, sharing, memory location and memory allocation.

(I think there are application to model-checking, by the way.)

There is a mathematics of referents, which is slowly revealing itself.

On referents

The act of 'pointing to something else' is itself important information, above and beyond what is being pointed to.

I argue the following:

1. Variables and their properties are a special case of referents.
2. Referents are best understood in terms of denotational, not syntactic, behaviour.

Philosophically, I propose that referring is a denotational activity, not a syntactic one.

Practically, I propose that the theoretical tools which we inherited from the early 20th century are not completely sufficient for the requirements of modern computer science. The theory and practice do not match up. This has real, and measurable, effects on tool support.

This is not a problem; it is an opportunity.

On referents

One particular corollary of my proposal is that the 'x' and 'y' in '$x + y = 2$' are real objects in their own right. They are just as real as the '2' (a number) or indeed the '$=$' (a relation on numbers).

On referents

So what do x and y look like, concretely?

Well, suppose x and y are FOL variables, i.e. they have the power to point to things (but not to sets of things). Then my Festschrift submission gives one answer to that question.

Because variables are in the denotation, they exist in the denotation of predicates, along with a denotational analogue of substitution. So the meaning of ‘forall’ is just this:

$$\bigcap_{x \in \mathcal{U}} [\![\phi]\!][a \mapsto x]$$

Conclusions

Nominal techniques approach referents as a denotational phenomenon.

This requires us to re-think the foundations of mathematics. We have to re-create computer science from the ground up. This is an extremely exciting thing to have the opportunity to do.

In previous work I built the tools to specify and construct 'variable-like behaviour' for referents. These are things that can be renamed, substituted, and quantified.

In the Festschrift paper I apply this programme specifically to the case of first-order logic.

Conclusions

So much of computer science is really about open terms, not closed terms. So much is about how information is organised and refers to other information.

I say: this is best understood in denotational terms. Current denotations do not really allow this.

This has a knock-on effect, by the way: it can be difficult to design languages that reflect on references, because we don't have the denotations to support this kind of thinking.

In the Festschrift paper, I give a denotation to the particular kind of referent that inhabits first-order logic.

That's a substitutable, bindable, quantifiable, variable 'ranging over' some domain.

Future work

Now we know it can be done, we can try to do it again. For instance:

- λa from the λ-calculus,
- second-order logic, higher-order logic,
- the propositional quantifier $\bigwedge\alpha$ from System F, and
- generalised quantifiers.

Plus, of course, there are many many applications to referents outside of logic.