Nominal techniques: logique, algèbre et calcul!

Murdoch J. Gabbay

Thanks to Delia Kesner and DIGITEO

November 16, 2010

Introduction

I am giving this talk at PPS—second only to Edinburgh as a centre of excellence in theoretical computer science.

It's a privilege to be here.

Nominal techniques were introduced by myself and Andrew Pitts in a 1999 LICS paper. The canonical journal reference is now "A New Approach to Abstract Syntax with Variable Binding", Formal Aspects of Computing, 13(3-5): 341-363 (2002).

An accessible introduction to nominal sets is Andrew Pitts's 2003 "Nominal Logic" paper in I&C (actually a first-order axiomatisation). Let me also advertise my survey paper "Foundations of nominal techniques" to be published in the Bulletin of Symbolic Logic (available on my webpage).

Logique, Algèbre et Calcul

Those concentrate on mathematical support for inductive syntax with binding; something that e.g. Christian Urban and others have implemented brilliantly.

But that's not really Logique, Algèbre et Calcul.

The next step was taken by myself and Maribel Fernández when we developed a rewriting framework based on nominal techniques, which I will only sketch.

The deeper point is that our bread-and-butter is specifying and reasoning about binders. It's part of the background furniture, so to speak.

Nominal techniques don't matter

When I stand here as a 'Mr Nominal Techniques' and say that I study names and binding this is true, but misleading.

We all do. Most interesting logics and calculi in computer science have binding: e.g. λ-calculus, first- and higher-order logic, set theory, and the π-calculus. Furthermore, binders are core aspects of the design.

(Combinators are an example of an interesting logic/calculus that does not have binding.)

Nominal techniques don't matter

You do not call somebody using Z specification is a set theorist; you do not call somebody using Isabelle a type-theorist.

In the same way, 'nominal' techniques should fade into the background.

Through nominal techniques we are approaching the foundations of computer science.

The goal is to create logical and semantic tools to give meaning and structure to complex applications. This is as it should be.

What logics, algebras, and calculi are available to specify logic, algebra, and calculus?

Your choices are broadly speaking:

- First-order logic/rewriting. Terms cannot bind. Dommage.
- Higher-order logic/rewriting. Terms can bind (thanks to λ) but this identifies binding constructs with computation constructs, which we may not want to do.

Quine famously called higher-order logic 'set theory in disguise'.

Nominal logics, algebras, and calculi

First-order logic is too weak; higher-order logic is already set-theory.

In a sequence of collaborative papers I have constructed:

- Nominal rewriting.
- Nominal algebra.
- Permissive nominal logic (PNL).

I invite you to consider these as the correct generalisations of the first-order case.

I also invite you to consider that you already use nominal rewriting, algebra, and PNL. You just may not realise it.

Informal specifications

Here are some informal specifications:

- If $x \notin fv(t)$ then $\lambda x.(tx) \to t$.
- If $x \notin fv(\Gamma)$ then $\Gamma \vdash \phi$ implies $\Gamma \vdash \forall x.\phi$.
- If $y \notin fv(t)$ then $\int t\,dx = \int t[y/x]\,dy$.
- If for every ϕ, $\phi[0/z]$ and $\forall z.\phi \Rightarrow \phi[z{+}1/z]$—then $\forall x.\phi$.

Here are their nominal formal specifications:

More formal nominal specifications

Suppose two sets of atoms $\mathbb{A}^<$ and $\mathbb{A}^>$ and write $\mathbb{A} = \mathbb{A}^< \cup \mathbb{A}^>$. Let permission sets $pmss(X)$ be sets of atoms differing finitely from $\mathbb{A}^<$.

- Nominal rewriting.
  If $a \notin pmss(X)$ then $\lambda a.(Xa) \to X$.
- Permissive-nominal logic.
  If $a \notin pmss(\Gamma)$ then $\Gamma \vdash \phi \Rightarrow \Gamma \vdash \forall a.\phi$.
- Nominal algebra.
  If $b \notin pmss(X)$ then $\int X\, da = \int (b\ a) \cdot X\, db$.
- Permissive-nominal logic.
  $\forall X.\big(X[c{:=}0] \wedge (\forall z.X \Rightarrow X[c{:=}c{+}1]) \Rightarrow \forall c.X\big)$.

Terms and propositions

Terms r and propositions ϕ are defined as follows:

$$r ::= a \mid (r, \ldots, r) \mid \mathsf{f}(\mathsf{r}) \mid [\mathsf{a}]\mathsf{r} \mid \pi{\cdot}\mathsf{X}$$
$$\phi ::= \bot \mid \phi \Rightarrow \psi \mid \mathsf{P}(\mathsf{r}) \mid \forall \mathsf{X}.\phi$$

This extends first-order terms: $[a]r$ is 'bind a in r'; $\pi{\cdot}X$ is 'α-convert names according to the permutation π, in X'.

The informal specification "If $x \notin fv(t)$ then $\lambda x.(tx) \to t$" becomes more formally

$$\forall X.\lambda([a]\mathsf{app}(\mathsf{X}, \mathsf{a})) \to \mathsf{X}$$

where X ranges over things that cannot have a in their support.

Some example theories: Substitution

These axiomatisations are finite and first-order (no infinite axiom-schemes).

$r[a \mapsto t]$ is sugar for sub([a]r, t).

(**sub**var)	$\forall X.$	$\mathsf{var}(\mathsf{a})[\mathsf{a} \mapsto \mathsf{X}]$	$\approx X$
(**sub**#)	$\forall X, Z.$	$Z[a \mapsto X]$	$\approx Z$
(**sub**succ)	$\forall X', X.$	$\mathsf{succ}(\mathsf{X}')[\mathsf{a} \mapsto \mathsf{X}]$	$\approx \mathsf{succ}(\mathsf{X}'[\mathsf{a} \mapsto \mathsf{X}])$
(**sub***op*)	$\forall X'', X', X.$	$(X'' \mathit{op} X')[a \mapsto X]$	$\approx (X''[a \mapsto X] \mathit{op} X'[a \mapsto X])$ $(\mathit{op} \in \{+, *, \dot{\Rightarrow}, \dot{\approx}\})$
(**sub**$\dot{\forall}$)	$\forall X, Z.$	$(\dot{\forall}([b]Z))[a \mapsto X]$	$\approx \dot{\forall}([b](Z[a \mapsto X]))$
(**subid**)	$\forall X.$	$X[a \mapsto \mathsf{var}(\mathsf{a})]$	$\approx X$

The permission set of X'', X', and X is equal to $\mathbb{A}^<$. The permission set of Z is equal to $(b\ a) \cdot \mathbb{A}^<$. $a \in \mathbb{A}^<$ and $b \notin \mathbb{A}^<$.

Some example theories: First-order logic

$$
\begin{array}{ll}
(\dot{\Rightarrow}) & \forall Z', Z.\ \epsilon(Z' \mathrel{\dot{\Rightarrow}} Z) \Leftrightarrow (\epsilon(Z') \Rightarrow \epsilon(Z)) \\
(\dot{\forall}) & \forall Z.\ \big(\epsilon(\dot{\forall}([a]Z)) \Leftrightarrow \forall X.\epsilon(Z[a{\mapsto}X])\big) \\
(\dot{\bot}) & \epsilon(\dot{\bot}) \Rightarrow \bot \\
(\dot{\approx}) & \forall X', X.\ X' \approx X \Rightarrow \epsilon(X' \mathrel{\dot{\approx}} X)
\end{array}
$$

Here Z' and Z have sort o and permission set $\mathbb{A}^<$; X' and X have sort ι and permission set $\mathbb{A}^<$; and $a \in \mathbb{A}^<$.

Some example theories: arithmetic

$$
\begin{array}{ll}
(\mathbf{PS0}) & \forall X.\ \mathsf{succ}(\mathsf{X}) \approx 0 \Rightarrow \bot \\
(\mathbf{PSS}) & \forall X', X.\ \mathsf{succ}(\mathsf{X}') \approx \mathsf{succ}(\mathsf{X}) \Rightarrow \mathsf{X}' \approx \mathsf{X} \\
(\mathbf{P{+}0}) & \forall X.\ X + 0 \approx X \\
(\mathbf{P{+}succ}) & \forall X', X.\ X' + \mathsf{succ}(\mathsf{X}) \approx \mathsf{succ}(\mathsf{X}') + \mathsf{X} \\
(\mathbf{P{*}0}) & \forall X.\ X * 0 \approx 0 \\
(\mathbf{P{*}succ}) & \forall X', X.\ X' * \mathsf{succ}(\mathsf{X}) \approx (\mathsf{X}' * \mathsf{X}) + \mathsf{X} \\
(\mathbf{PInd}) & \forall Z.\ (\epsilon(Z[a{\mapsto}0]) \Rightarrow \\
 & \qquad \big(\forall X.(\epsilon(Z[a{\mapsto}X]) \Rightarrow \epsilon(Z[a{\mapsto}\mathsf{succ}(\mathsf{X})]))\big) \Rightarrow \\
 & \qquad \forall X.\epsilon(Z[a{\mapsto}X]))
\end{array}
$$

All variables have permission set $\mathbb{A}^<$, and $a \in \mathbb{A}^<$.

Sequent derivation rules of PNL

$$\frac{}{\Phi,\ \phi \vdash \pi{\cdot}\phi,\ \Psi}\ (\mathbf{Ax}) \qquad \frac{}{\Phi,\ \bot \vdash \Psi}\ (\bot\mathbf{L})$$

$$\frac{\Phi \vdash \phi,\ \Psi \quad \Phi,\ \psi \vdash \Psi}{\Phi,\ \phi \Rightarrow \psi \vdash \Psi}\ (\Rightarrow\mathbf{L}) \qquad \frac{\Phi,\ \phi \vdash \psi,\ \Psi}{\Phi \vdash \phi \Rightarrow \psi,\ \Psi}\ (\Rightarrow\mathbf{R})$$

$$\frac{\Phi,\ \phi[X{:=}r] \vdash \Psi \quad (\mathit{fa}(r){\subseteq}\mathit{pmss}(X))}{\Phi,\ \forall X.\phi \vdash \Psi}\ (\forall\mathbf{L})$$

$$\frac{\Phi \vdash \phi,\ \Psi \quad (X{\notin}\mathit{fV}(\Phi,\Psi))}{\Phi \vdash \forall X.\phi,\ \Psi}\ (\forall\mathbf{R})$$

Conclusions

I've shown you some rules and shared with you some wishful thinking.

Let me recap:

First-order logic does not allow terms to bind. Higher-order logic is overly powerful and involves circularity since the device that manages binding—λ—also manages computation.

Conclusions

I have developed ‘nominal’ logics, algebras, and calculi. Most recently Permissive-Nominal Logic with Gilles Dowek. These let you write down informal specifications almost symbol-for-symbol.

Thus I claim that you all do nominal techniques already, but informally.

More generally, to engage with names and binders is to engage with theoretical computer science in the broadest sense. Nominal techniques are a tool for doing this.

Thus PNL and related systems will succeed when they fade into the background, for most people; of course they can be studied in its own right, just like any logic and semantics.