Nominal techniques: past and future

Murdoch J. Gabbay

Thanks to the organisers,
. . . and to the prize selection committee!

11 July 2019

Introduction

It's my pleasure to be here. Thank you from both me and Andrew for the award.

I've lived long enough in the Real World to appreciate it.

Nominal Techniques Summer School: slice of research

3rd School on Foundations of Programming and Software Systems

(FoPSS 2019, co-located with HIGHLIGHTS 2019)

Warsaw, 10–15 September 2019

`https://www.mimuw.edu.pl/~fopss19/`

Johannes Borgström, *Nominal Process Calculi and Modal Logics*
Bartek Klin, *Basic Nominal Techniques*
Murdoch Gabbay, *Advanced Nominal Techniques*
Andrew Pitts, *Nominal Sets and Functional Programming*
Maribel Fernández, *Nominal Rewriting and Unification*
Mikołaj Bojańczyk, *Computation Theory with Atoms*
Sławomir Lasota, *Computation Theory with Atoms II*
Andrzej Murawski, *Nominal Game Semantics*

Introduction

Nominal techniques go back to two papers with Andrew Pitts:

- *A New Approach to Abstract Syntax with Variable Binding* in Formal Aspects of Computing, 2001.
- *A new approach to abstract syntax involving binders* in LICS 1999 (test-of-time award, 2019).

I know of two talks in this very ICALP conference using nominal techniques:

- *A Kleene Theorem for Nominal Automata* by Paul Brunet and Alexandra Silva.
- *Varieties of Data Languages* by Henning Urbat and Stefan Milius.

There's clearly real uptake here. Great. So what's it all about?

So what's it all about?

There are several good sources for technical information on nominal sets and their applications, including:

- A book by Andrew Pitts.
- A (draft) book by Mikołaj Bojańczyk.
- A survey article by myself.

So what's it all about?

Now, I might launch into a technical definition of nominal sets. Your eyes would glaze, I would be frustrated, and twenty minutes would pass.

So let's not do that.

I might give examples of nominal structure:

- syntax of the untyped λ-calculus,
- Schanuel topos,
- orbit-finite sets,
- function spaces, abstraction sets, etc.

But let's not do that, either.

So what's it all about?

I could say that nominal sets are an abstraction of syntax: elements can contain names and have a permutation action and *finite support* axiom modelling (abstractly) the 'free variables of' function.

True . . . but misleading:

- there are 'nominal' constructions with infinite support or no support at all, and
- ones with finite support but which are non-syntactic.

'Generalisaation of syntax-with-binding' is an important motivating example, but one which has tended to obscure the bigger picture.

Let's try to say something different, arguably controversial, and which is harder to find plainly stated in the literature:

Names are ZFA atoms

I put it to you that the innovation of the two papers above specifically, and of nominal techniques in general, is to say:

Names are a datatype

— specifically: the set of atoms in set theory with atoms (ZFA).

Let me elaborate . . .

A picture of names

Recall the cumulative hierarchy:

$$V_0 = \varnothing \quad V_{\alpha+1} = V_\alpha \cup \mathit{pow}(V_\alpha) \quad V_\lambda = \bigcup_{\alpha<\lambda} \mathit{pow}(V_\alpha)$$

This is what you've been told the universe is. Names do not exist in this universe. *Everything's a set!*

But in Computer Science, it's not true that everything's a set. Names naturally arise everywhere, and it seems to me that our universe looks more like this:

$$V_0 = \mathbb{A} = \{\mathbf{a}, \mathbf{b}, \mathbf{c}, \dots\} \quad V_{\alpha+1} = V_\alpha \cup \mathit{pow}(V_\alpha) \quad V_\lambda = \bigcup_{\alpha<\lambda} \mathit{pow}(V_\alpha)$$

Note: names $a, b, c, \dots$ have an independent existence.

A picture of names (typed version)

Similarly if you think in terms of types:

$$\alpha ::= o \mid \iota \mid \alpha \to \alpha$$
$$\alpha ::= o \mid \iota \mid \mathbb{A} \mid \alpha \to \alpha$$

$$V_0 = \varnothing \quad V_{\alpha+1} = V_\alpha \cup \mathit{pow}(V_\alpha) \quad V_\lambda = \bigcup_{\alpha<\lambda} \mathit{pow}(V_\alpha)$$
$$V_0 = \mathbb{A} \quad V_{\alpha+1} = V_\alpha \cup \mathit{pow}(V_\alpha) \quad V_\lambda = \bigcup_{\alpha<\lambda} \mathit{pow}(V_\alpha)$$

There are very few native users of universes that *aren't* a cumulative hierarchy starting from a base type that looks like $\varnothing$ or $\mathbb{N}$.

NFU and ZFA are examples of universes that differ from & extend the above.

A paper on this

I expand on these observations in a recent paper:

- *Equivariant ZFA and the foundations of nominal techniques*, in press.
  (Available from `http://www.gabbay.org.uk/papers.html`.)

The paper argues for Zermelo-Fraenkel sets with Atoms and Choice (ZFAC) as a foundation for nominal techniques — and not Fraenkel-Mostowski set theory.

In particular, nominal techniques are perfectly consistent with the Axiom of Choice. See paper above.

The future

People are starting to 'get' nominal techniques. That's wonderful; I am delighted to see how these ideas are being extended.

Let's talk about the medium-term future.

I dream of ZFA(C) / nominal sets taught in foundational courses, instead of ZF(C) / sets. Perfectly possible now.

To my mind, a major outstanding issue is *tooling*: Specifically:

- Support for nominal datatypes in programming languages (Haskell, OCaml, Python, ...).
- Support for nominal techniques in theorem-provers (Isabelle, COQ, Agda, ...)

Adding nominal support is more than programming a datatype of names and permutations.

The future

Adding support for nominal constructs requires more than only programming up a datatype of names and permutations.

The issue is to extend the language or theorem prover — which, I guarantee, is implicitly based on a ZF-style model — with relevant ZFA structure, including at higher types.

This is a research problem. Get in touch if interested.

Thank you for listening.

Appendix: definitions

. . . just in case somebody asks

or I want to refer to it . . .

Nominal sets definition

Fix a countably infinite set of atoms $\mathbb{A}$. Write $\Sigma_{\mathbb{A}}$ for the symmetry group of $\mathbb{A}$ (bijections π on $\mathbb{A}$; also called *atoms-permutations*).

A $\Sigma_{\mathbb{A}}$-set X is:

- An underlying set X, with
- a group action $\Sigma_{\mathbb{A}} \times X \to X$.

Example:

- $\mathbb{A}$ is a $\Sigma_{\mathbb{A}}$-set where $\pi \cdot a = \pi(a)$.
- Syntax of λ-terms over atoms as variable symbols is a $\Sigma_{\mathbb{A}}$-set where π acts pointwise on the atoms in a term. So $\pi \cdot (\lambda a.\lambda b.ab) = \lambda \pi(a).\lambda \pi(b).\pi(a)\pi(b)$.

Nominal sets definition

Note that if X is a $\Sigma_{\mathbb{A}}$-set then so is $power(X)$ the powerset of X, by the pointwise action

$$\pi \cdot X = \{\pi \cdot x \mid x \in X\} \quad \text{where} \quad X \in power(X).$$

Example if $a, b, c \in \mathbb{A}$ then:

- $\pi \cdot \{a, b, c\} = \{\pi(a), \pi(b), \pi(c)\}$.
- $\pi \cdot (\mathbb{A} \setminus \{a, b, c\}) = \mathbb{A} \setminus \{\pi(a), \pi(b), \pi(c)\}$.
- $\pi \cdot \big\{\{a, b, c\}, \mathbb{A} \setminus \{a, b, c\}\big\} = \big\{\{\pi(a), \pi(b), \pi(c)\}, \mathbb{A} \setminus \{\pi(a), \pi(b), \pi(c)\}\big\}$.

Nominal sets definition

If X is a $\Sigma_{\mathbb{A}}$-set and $x \in \mathsf{X}$ and $X \subseteq_{fin} \mathsf{X}$ then define:

$$\begin{aligned} \mathit{fix}(x) &= \{\pi{\in}\Sigma_{\mathbb{A}} \mid \pi{\cdot}x = x\} \\ \mathit{stab}(X) &= \textstyle\bigcap_{x\in X} \mathit{fix}(x) = \{\pi{\in}\Sigma_{\mathbb{A}} \mid \forall x{\in}X.\pi{\cdot}x = x\} \end{aligned}$$

Easy to check that if $A \subseteq_{fin} \mathbb{A}$ (finite subset) then:

$$\mathit{stab}(A) = \{\pi{\in}\Sigma_{\mathbb{A}} \mid \forall a{\in}A.\pi(a) = a\}.$$

Define $A\$x$ and say $A \subseteq \mathbb{A}$ supports $x \in \mathsf{X}$ by:

$$\begin{array}{lll} A\$x & \text{when} & \mathit{stab}(A) \subseteq \mathit{fix}(x) \\ & \Leftrightarrow & \forall\pi{\in}\Sigma_{\mathbb{A}}.\big(\forall a{\in}A.\pi(a) = a\big) \Rightarrow \pi{\cdot}x = x. \end{array}$$

Nominal sets definition

A nominal set X is a $\Sigma_{\mathbb{A}}$-set such that every $x \in X$ has a unique least finite supporting set $\mathrm{supp}(x) \subseteq_{fin} \mathbb{A}$.

Examples:

- $\mathbb{A}$ is a nominal set. If $a \in \mathbb{A}$ then $\{a\}\$a$.
- Finite sets of atoms are a nominal set. If $A \subseteq_{fin} \mathbb{A}$ then $A\$A$.
- Cofinite sets of atoms (complements of finite sets of atoms) are a nominal set. If $A \subseteq_{fin} \mathbb{A}$ then $A\$\mathbb{A}\backslash A$.
- Syntax is a nominal set (where variables symbols are atoms). t is supported by the atoms mentioned in t (free or bound).
- Syntax quotiented by α-equivalence is a nominal set (where variables symbols are atoms) $[t]_\alpha$.
  $fv(t)\$[t]_\alpha$ (in words: an α-equivalence class $[t]_\alpha$ is supported by $fv(t)$ the free variables of t).
- If X is a nominal set then $power_{fs}(X)$ the set of $X \subseteq X$ with finite support under the permutation action, is a nominal set.

Freshness

If $a \in \mathbb{A}$ define $a\#x$ and say a is fresh for x by:

$$a\#x \quad \text{when} \quad a \notin \text{supp}(x).$$

- If X is finite sets of atoms then $a\#X$ when $a \notin X$.
- If X is cofinite sets of atoms then $a\#X$ when $a \in X$.
- If X is syntax quotiented by α-equivalence then $a\#[t]_\alpha$ when $a \notin fv(t)$ (a is not free in t).

This concludes the definitions.