Equivariance: the name of the true monster in your induction

Murdoch J. Gabbay

LAIV seminar

15 June 2023

Intro

Many thanks to the LAIV organisers for inviting me to speak.

Following on from John Power's talk, I'll survey nominal algebra, its axiomatisations of substitution and first-order logic, and indicate why that's the easy bit.

Then I'll talk about the hard bit — which is equivariance.

This talk may be opinionated. I mean this in a positive way: I have made mistakes and fallen into traps. Heed this history, lest you repeat it.

Permutation sets and equivariance

Fix a countable set of atoms $\mathbb{A}$ and write $\mathrm{Perm} = \mathbb{A} \cong \mathbb{A}$ for bijections on atoms.

A permutation set $(X, \cdot)$ is

- an underlying set X, and
- a permutation group action $\pi \cdot x$.[1]

A function $f : \prod_i (X_i, \cdot) \to (Y, \cdot)$ or predicate $\phi : \prod_i (X_i, \cdot) \to \mathbb{B}$ is equivariant when it commutes with the permutation action:

$$f(\pi \cdot x_1, \ldots, \pi \cdot x_n) = \pi \cdot f(x_1, \ldots, x_n)$$
$$\phi(\pi \cdot x_1, \ldots, \pi \cdot x_n) \Leftrightarrow \phi(x_1, \ldots, x_n).$$

(The ϕ-case is a special case of the f-case, taking a standard trivial action $\pi \cdot \top = \top$ and $\pi \cdot \bot = \bot$ in $\mathbb{B}$. The f-case is a special case of the ϕ-case, if we specify f by its predicate-graph.)

[1] Group action: $\pi \cdot \pi' \cdot x = (\pi \circ \pi') \cdot x$ and $\mathrm{id} \cdot x = x$.

Permutation sets and equivariance

Most functions and predicates that you can specify are equivariant — though you must permute over *all parameters*! For example:

- Equality is equivariant.
  $\pi{\cdot}x = \pi{\cdot}y$ iff $x = y$.
- 'Size of a term' is equivariant, where $\pi{\cdot}n = n \in \mathbb{N}$ and atoms serve as variable symbols:

$$size(\lambda a.ab) = size(\lambda c.cg).$$

- 'free variables of' and 'variables mentioned in' are equivariant.
- 'Is an element of' is equivariant, where the standard action on sets is pointwise as $\pi{\cdot}X = \{\pi{\cdot}x \mid x \in X\}$:

$$x \in X \quad \Leftrightarrow \quad \pi{\cdot}x \in \pi{\cdot}X.$$

Fixing permutations

Given $S \subseteq_{fin} \mathbb{A}$, write $fix(S)$ for the set of π that fix every $a \in S$:

$$fix(S) = \{\pi \in \mathrm{Perm} \mid \forall a \in S.\pi(a) = a\}.$$

Example:

- id the identity permutation is in $fix(S)$ for every $S \subseteq_{fin} \mathbb{A}$.
- $(a\,b)$ the swapping $(a \mapsto b, b \mapsto a)$ is in $fix(S)$ iff $a, b \notin S$.
- Note: $(a\,b) \notin fix(\{a, b\})$! This is because $(a\,b)(a) \neq a$ and $(a\,b)(b) \neq b$.

Nominal sets

If $(X, \cdot)$ is a permutation set and $x \in X$ then say S supports x, written $S\$x$, when $\pi \in fix(S)$ implies $\pi \cdot x = x$.

Lemma (equivariance of \$): $S\$x$ if and only if $\pi \cdot S\$\pi \cdot x$.

Example:

- If $a \in A$ then $S\$a$ if and only if $a \in S$.
- If $A \subseteq_{fin} \mathbb{A}$ then $S\$A$ precisely when $A \subseteq S$.
- If $A \subseteq_{fin} \mathbb{A}$ then $S\$(\mathbb{A} \setminus A)$ precisely when $A \subseteq S$.

A nominal set is a permutation set whose every $x \in X$ has some finite supporting set.

(Lemma: If $x \in X$ has some finite supporting set, then it has a least one, $\mathrm{supp}(x)$*.)*

Write $a\#x$, and call a fresh for x, when there exists $S \subseteq_{fin} \mathbb{A}$ such that $S\$x$ and $a \notin S$.

A permutation set is nominal when $\forall x.\exists a.a\#x$.

Substitution in nominal algebra

Assume an X and a substitution $\mathsf{X} \times \mathbb{A} \times \mathsf{X} \to \mathsf{X}$ such that:

$$\begin{array}{ll} a\#z \Rightarrow & z[a{\mapsto}x] = z \\ b\#z \Rightarrow & z[a{\mapsto}x] = ((b\,a){\cdot}z)[b{\mapsto}x] \\ b\#x \Rightarrow & z[a{\mapsto}x][b{\mapsto}y] = z[b{\mapsto}y][a{\mapsto}x[b{\mapsto}y]] \end{array}$$

This axiomatises substitution in nominal algebra.

Nominal algebra is a (possibly multi-sorted) logic of equality over nominal sets. It permits axioms of the form 'freshness assumptions $\Rightarrow s = t$', as above.

- In Nominal algebra (pdf link) and Nominal Algebra and the HSP Theorem (pdf link) we show how this is a natural equality logic over nominal sets.
- We prove the axioms above *do* axiomatise substitution: see several papers, including those above and here.

So, substitution is an algebra; like groups and rings. It gets better ...

Nominal algebra (FOL)

. . . Assume the axioms of Boolean Algebra ($\bot$, $\wedge$, $\neg$). Write $v \leq v'$ for $v \wedge v' = v$. Now add $\forall$, with axioms:

$$\begin{aligned} a\#v &\Rightarrow \forall a.v = v \\ b\#v &\Rightarrow \forall a.v = \forall b.(b\,a)\cdot v \\ &\quad\ \forall a.v \leq v \end{aligned}$$

Slogan: FOL = BOOL + $\forall$.

This *does* algebraise FOL: we can prove soundness and completeness; build notions of (ultra)filter; and prove a Stone-style representation result — just as for Boolean algebras (see Semantics out of context; pdf link).

Similarly for the λ-calculus.

Why is this not better-known?

Why is this not better-known?

Because:

1. I suck at publicity and . . .
2. . . . it's not fashionable.
   Categories are modern, and what people expect to see. Universal algebra is old. Nobody cares about a new universal algebra, or about a Stone duality result for FOL (though I rate it as amongst my strongest work, if I do say so myself).
3. But there's something else.
   The above isn't actually the hard bit.

Not that it was easy, but it elides a core difficulty that's embedded already in the phrase 'permutation set'.

Inductive syntax + binding $\Longrightarrow$ renaming

A canonical problem is to reconcile inductive datatypes of syntax, with binding. What's the core problem here?

E.g. if we want to inductively push a substitution under a binder, we may need to RENAME. Naively we'd use a substitution $[a \mapsto a']$:

$$(\forall a.\phi)[b \mapsto s] \to \forall a'.(\phi[a \mapsto a'][b \mapsto s]).$$

That's awful and wrong; go there at your own risk. Fiore, Plotkin and Turi would use a name-for-name substitution:

$$(\forall a.\phi)[b \mapsto s] \to \forall a'.(\phi\langle a \mapsto a' \rangle[b \mapsto s]).$$

Nominal techniques would use a permutation:

$$(\forall a.\phi)[b \mapsto s] \to \forall a'.((a'\ a)\cdot\phi)[b \mapsto s]).$$

Now $\phi' \in \{\phi[a \mapsto a'], \phi\langle a \mapsto a' \rangle, (a'\ a)\cdot\phi\}$ is not a subterm of $\forall a.\phi$, so you lose IH. How to deal with this?

Nominal abstractions

It's easy to axiomatise abstraction $[a]x$ in nominal algebra. Define a relation on pairs

$$b\#x \Rightarrow (a, x) \sim_\alpha (b,\ (b\,a)\cdot x)$$

and quotient

$$[a]x = [(a, x)]_{\sim_\alpha} \in [\mathbb{A}]\mathsf{X} = (\mathbb{A} \times \mathsf{X})/\sim_\alpha .$$

$[\mathbb{A}]$- has excellent properties, e.g. it commutes with $+$, $\times$, and $\to$.[2]

We can now build an inductive datatype of syntax-with-binding in a (by now) familiar way. For example:

$$\Lambda = \mathbb{A} + \Lambda \times \Lambda + [\mathbb{A}]\Lambda.$$

But this just squeezes the problem elsewhere ...

[2] When I showed Andrew $[\mathbb{A}](\mathsf{X} \to \mathsf{Y}) \cong [\mathbb{A}]\mathsf{X} \to [\mathbb{A}]\mathsf{Y}$ he said to go home and find the error in my working.

Induction

. . . Suppose we want to define an inductive function $F^z : \Lambda \to \mathsf{Y}$. (Here, z is a parameter of other stuff we needed to build F.)

As usual, we must provide functions:

$$\begin{aligned} f^z_{\mathbb{A}} &: \quad \mathbb{A} \to \mathsf{Y} \\ f^z_{app} &: \Lambda \times \Lambda \to \mathsf{Y} \\ f^z_{lam} &: \quad [\mathbb{A}]\Lambda \to \mathsf{Y} \end{aligned}$$

Nominal abstractions

To provide $f^z_{lam} : [\mathbb{A}]\Lambda \to \mathsf{Y}$ we must provide

$$f'_z : \mathbb{A} \times \Lambda \to \mathsf{Y},$$

and then prove that f'_z respects $\sim_\alpha$ for all names not in some $S \subseteq_{fin} \mathbb{A}$ such that $S\$z$. This is an equivariance argument:

$$f'_z(\pi(a), \pi{\cdot}x) = \pi{\cdot}f'_z(a, x). \quad \text{for all} \quad \pi \in \mathit{fix}(S).$$

(Or $f' : \mathsf{Z}\times\mathbb{A}\times\Lambda \to \mathsf{Y}$ *and prove* $f'(\pi{\cdot}z, \pi(a), \pi{\cdot}x) = \pi{\cdot}f'(z, a, x)$ *for all* π*.)*

This equivariance proof may trigger an inductive argument on x, and — if we're unlucky — also on z.

So now we have a *nested* argument; and the nesting can repeat. For toy examples (like Λ) this is manageable. For non-toy examples, it's not.

Equivariance is hard

Equivariance proofs always succeed: this is for deep foundational reasons, made formal in Equivariant ZFA and the foundations of nominal techniques.

But as noted in that paper, the proofs don't *scale*. We get swamped by equivariance proofs.

Nominal techniques — helpful for discussing names and binding! — reveal this issue, but it is inherent. This is not 'just' a nominal thing.

If you have binding then equivariance is there — just waiting to jump out and get you.

This is, I now believe, the true monster we must slay, and the reason why attempts to deal with names and binding encounter difficulties regardless of the specifics of the approach.

This is unwelcome news. How should we respond?

Five stages of grief

- **Denial.**
  I'm different. I don't have this problem. There is no problem.
- **Anger.**
  Gabbay's work is fine but he doesn't explain the connection to HOAS/Categories/de Bruijn/AC/(delete as appropriate).
- **Bargaining.**
  Let's can circumvent the issue using *this* clever trick. *Now, if I can fold in enough of the meta-logic, and ask the user to provide these simple lemmas, then surely we can find a reasonable way out.*
  It won't work. You can't bargain your way around equivariance.
- **Depression.**
  Give up. Write papers about something else.
- **Acceptance.**
  OK. This is a real issue. How do we deal with it?

How to accept, and live with equivariance

Technically, it's not actually hard. The barriers are more psychological than mathematical.

We need to admit the following:

1. Atoms (urelemente) are primitive.
   The cumulative hierarchy starts from $\mathbb{A}$; not from $\varnothing$. The hierarchy of simple types starts from $\mathbb{N}$ *and* $\mathbb{A}$; not just $\mathbb{N}$.
2. Once we admit atoms as primitive, then we can admit equivariance as an axiom.

$$\pi \cdot f(x_1, \ldots, x_n) = f(\pi \cdot x_1, \ldots, \pi \cdot x_n)$$

(for f an expression with no other parameters).
This is not something we prove *from* first principles; this *is* a first principle!

So we have a slogan: atoms are real, and equivariance is an axiom.

Atoms are real, and equivariance is an axiom

That was the hard bit (after that, nominal algebra is pretty easy).

It's hard because it goes against what we've educated to expect. For example:

- The ZFC cumulative hierarchy is the limit of $X \longmapsto X \cup \mathit{pow}(X)$, starting from $\varnothing$ (not $\mathbb{A}$).
- The simple types hierarchy has types $\alpha ::= \mathbb{N} \mid \alpha \to \alpha$ (not $\alpha ::= \mathbb{N} \mid \mathbb{A} \mid \alpha \to \alpha$).
- No theorem-prover or programming language in the wild[3] offers atoms and equivariant permutations as polymorphic primitives, even though there is no technical barrier to doing so.
- ...and so on.

[3] 'in the wild' = not one of the prototypes inspired directly by nominal ideas

The punchline

The maths we are brought up with comes with an embedded assumption that atoms are *not* primitive, and equivariance (if required) should be derived on instances as they arise.

This is an error.

Proving theorems is hard — but accepting a reality that is not as we expected, is much harder.

The 'difficulty' with names and binding is not really mathematical. It's a social problem, created by the use of foundations that do not assign atoms and equivariant permutations equal status with 'standard' data like numbers or functions.

What I do not claim:

- I *do not* claim that $\mathbb{A}$ is incompatible with Choice.
  On the contrary, some constructions, such as my ultrafilters for nominal FOL algebras, require Choice. They just generate non-finitely-supported sets.
  Note that you can have a finitely supported X such that all $x \in X$ are not finitely supported — take X to be the set of well-orderings on $\mathbb{A}$.
- I *do not* argue for One True Foundation — on the contrary, I argue for more flexibility, including that theorem-provers and programming languages could usefully admit atoms and their permutation as primitive, and equivariance as a primitive reasoning principle. (Best done as a package, possibly calling underlying API hooks provided by the language designers.)
- I *do* claim that this stuff could be more widely, e.g. by making names and binding available in implementations.

To learn more

- A paper that presents these ideas: https://doi.org/10.1093/logcom/exz015 (pdf link).
- A Haskell package: https://hackage.haskell.org/package/nom
- A prototype Agda library based partly on these ideas (joint work with Orestis Melkonian; he wrote the Agda code): https://omelkonian.github.io/data/publications/nominal-agda-short.pdf