

# Equivariant ZFA and the foundations of nominal techniques

Murdoch J. Gabbay

XIII Summer Workshop in Mathematics

[https://www.mat.unb.br/verao2021/workshop\\_en.html](https://www.mat.unb.br/verao2021/workshop_en.html)

10 February 2021

# Introduction

It's my pleasure to be here. Thank you for inviting me to speak.  
I'd like to talk about the shape of the universe.

# The cumulative hierarchy of sets ...

Cumulative hierarchy starts at 0 and iterates with transfinite powersets:

$$\begin{aligned}V_0 &= \emptyset \\V_{\alpha+1} &= V_\alpha \cup \text{pow}(V_\alpha) \\V_\lambda &= \bigcup_{\alpha < \lambda} \text{pow}(V_\alpha)\end{aligned}$$

An essence of its structure is represented by ordinals:

$$\begin{aligned}\emptyset = 0 &\in V_1 \\ \{0\} = 1 &\in V_2 \\ \{0, 1\} = 2 &\in V_3 \\ \{0, 1, \dots, i\} = i + 1 &\in V_{i+2} \\ &\dots \\ \{0, 1, \dots\} = \omega &\in V_{\omega+1} \quad \omega_1, \omega^\omega \dots \\ \{\alpha \mid \alpha < \beta\} = \beta &\in V_{\beta+1}\end{aligned}$$

## ... is a mathematical universe ...

This can all be represented in a cumulative hierarchy:

- ▶ Arithmetic.
- ▶ Complex numbers.
- ▶ Group theory.
- ▶ Vector spaces.
- ▶ Streams (infinite lists).
- ▶ (Co)Inductive datatypes.
- ▶ (Typed) Functional programming.
- ▶ Dependent types. Category theory. Mechanised mathematics. SAT solving. Topos theory. ...

What does 'represent' mean?

... in which we can represent maths ...

**Example:** If  $x$  and  $y$  are sets then  $(x, y)$  is represented as  $\{\{x\}, \{x, y\}\}$  (Kuratowski pairs).

**Example:** If  $X$  and  $Y$  are sets then  $f : X \rightarrow Y$  is represented as  $\{(x, f(x)) \mid x \in X\}$  (graph of a function).

In the past perhaps we learned this as undergraduates and ignored it, but computing is making the influence of the ideas more visible.

Modern computing is implemented pure maths (it's quite incredible, really). As implementations get more powerful,

- ▶ more mathematics gets turned into concrete software, and
- ▶ more mathematics is devoted to concrete verification / model-checking of system properties.

So things that were abstract, take form and start jumping around inside your computer!

... and thus the real world.

E.g. **Plato's ideals** (c. 350 BC) are a nice idea ('the ideal of a car').  
**Java object classes** (c. 1995 AD) are a concrete implementation ('Class Car has methods start, stop, steer, license number ...').

**Propositional, first-, and higher-order logic** are nice ideas.  
**Prolog, relational databases, SAT solvers, and Isabelle/HOL** are implementations.

**$\lambda$ -calculus, System F, and dependent types** are nice ideas.  
**Lisp, Haskell, OCaml, COQ, and Agda** are implementations.

These relate to the cumulative hierarchy like rivers relate to the summit of a mountain from which they flow: the rivers have their own identity, but the *mountain* sets them in motion and gives the initial force and direction.

Shift the mountain ... and the rivers shift too.

# So are there many possible universes?

Certainly.

It's common to tweak the cumulative hierarchy — usually with assumptions designed to make sure it's big enough. E.g.

- ▶ The Axiom of Choice = function-spaces are maximal.
- ▶ Inaccessible cardinal = the cumulative hierarchy contains a set which is itself a cumulative hierarchy. (Thus, the universe can represent a scale model of itself.)

COQ has 'universes', Isabelle has 'kinds' and 'theories'. Oracles may be admitted to inject extra power into a system. Etc.

This is common. But what is not generally questioned is the cumulative hierarchy itself.

## But is the cumulative hierarchy itself right?

- ▶ Yes, for the mathematics of the 20th century.
- ▶ Perhaps less so, for the mathematics of the 21st century.


The issue is lack of **symmetry**. Specifically, there are no nontrivial  $\in$ -endomorphisms ( $f$  such that  $f(X) = \{f(x) \mid x \in X\}$ ).

It's a rigid structure.<sup>1</sup>

Another way to see this is that — assuming AC — everything can be encoded as an ordinal. This is usually considered a feature, but from a 21st century perspective I'd argue that it's a bug.

To see why, let's look at some examples of 21st century data:

---

<sup>1</sup>C.f. Ehrenfurcht-Mostowski models. But that's out of scope here.  8/41



# Symmetries

- ▶ Computer memory is addressed using **pointers**  $p$ .
- ▶ Communication channels and file I/O are addressed using **handles**  $h$ .
- ▶ Formal syntax (like computer code) addresses unknown values using variable symbols like  $x$ ,  $y$ , and  $z$  in  $x + y$ , or  $\forall x.\phi(x)$ , or  $\int f(x)dx$  or  $\sum_x f(x)$ .  
Note that variable symbols are not themselves values of the domain they represent: ' $x = 1$ ' does not literally mean that the symbol ' $x$ ' is the ordinal 1!
- ▶ Fermions. Electrons  $e$  are fermions.
- ▶ Locations (e.g. on a UTxO blockchain, or a game in the sense of game theory) link a structure together; one block/node connects to another block/node by pointing to a location on that node, by name.

# Symmetries

Memory pointers, I/O handles, variable symbols, fermions, and locations have a distinctive structure: they are **atomic**, **interchangeable**, and **generative** (you can always generate a new one).

- ▶ If I give you an electron you can't meaningfully say 'I don't like this one; I want another'.
- ▶ If you ask me for a fresh variable symbol, I can't say 'I ran out'.

# Symmetries

Here's how the symmetry manifests itself:

- ▶ If we run a computation  $f$  with some choice of atoms  $a, b, c, d$  to obtain a result  $f(a, b, c, d)$ , then
- ▶ we know we could run the same computation with  $\pi(a), \pi(b), \pi(c), \pi(d)$ , and the result should depend symmetrically on the permutation  $\pi$  according to **equivariance**

$$\pi \cdot f(a, b, c, d) = f(\pi(a), \pi(b), \pi(c), \pi(d)).$$

This is a symmetry interacting with our universe, where  $f$  represents some computation or logic on the universe, and equivariance asserts that it is symmetric under permuting atoms.

This can be **extremely useful** and it is **not directly representable** in the cumulative hierarchy.

# Capturing this behaviour

So we have a class of data with the following properties:

1. **Symmetry.**

The universe is symmetric up to permuting the underlying datatype of pointers / channels / variable symbols / locations.

2. **Generativity.**

There is some notion of 'for a fresh  $x$ ,  $\phi$ '.

The cumulative hierarchy is particularly bad at modelling both:

- ▶ Everything's basically an ordinal and ordinals are totally ordered, and
- ▶ we can exhaust any set  $X$  just by writing down ' $X$ '.

The costs of this are real.

# The costs of asymmetry

Concrete example: consider an automaton that at each time step generates a fresh nonce (nonce = atom). We don't care **which** nonce it generates, so long as it's fresh.

Intuitively this automaton is finite, with just one action: 'make fresh nonce'.

Formally, the state space is huge. If we model nonces using  $\omega = \mathbb{N}$  (which is really all we can do) then our automaton has  $\omega$ -infinitely many branches at each step: e.g.  $i, i + 1, i + 2, \dots$

This discrepancy is easily resolved by observing that the formal state space is **finite, up to a symmetry** of equivariantly permuting the choice of nonce.

# The costs of asymmetry

The cumulative hierarchy gives no primitive representation of this symmetry.

This is a conceptual barrier: it provides no guidance on what a model of a finite-up-to-symmetry model should look like (get many ad hoc solutions).

It is also a practical barrier: a verification of our automaton's behaviour must carry **equivariance proofs**, that prove explicitly that symmetries are respected.

# The costs of asymmetry

Unfortunately **equivariance proofs tend to be expensive**.

They must be executed explicitly for every step and every operation. Empirically, they tend to resist automation — they are simple, but like a tangled ball of string is simple.

This can easily cause a quadratic blow-up in proof size.

# The costs

There is also a moral cost here.

We observed a general phenomenon. As mathematicians, we should be eager to provide a clean abstraction to model it.



# A mathematical universe with atoms

I propose EZFA(C): equivariant Zermelo-Fraenkel set theory with Atoms (and Choice).

See [Equivariant ZFA and the foundations of nominal techniques](#):

- ▶ <https://doi.org/10.1093/logcom/exz015>
- ▶ <https://arxiv.org/abs/1801.09443>

# A mathematical universe **with atoms**

The EZFAC cumulative hierarchy looks like this:

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

Note  $V_0 = \{a, b, c, \dots\} \neq \emptyset$ . Our universe starts from atoms, instead of from  $\emptyset$ .

This hierarchy supports a permutation action, defined by a trivial  $\in$ -induction:

$$\begin{aligned}\pi \cdot a &= \pi(a) \\ \pi \cdot X &= \{\pi \cdot x \mid x \in X\}\end{aligned}$$

Now we have plenty of non-trivial  $\in$ -automorphisms. Our universe has symmetry!

## EZFAC (simple types version)

In case you think in terms of types, here's the universe without and with atoms:

$$\begin{aligned}\alpha &::= o \mid \iota \mid \alpha \rightarrow \alpha \\ \alpha &::= o \mid \iota \mid \mathbb{A} \mid \alpha \rightarrow \alpha\end{aligned}$$

The permutation automorphism becomes the **conjugation action**:

$$\begin{aligned}\pi \cdot a &= \pi(a) & a &: \mathbb{A} \\ \pi \cdot x &= x & x &: o, x : \iota \\ (\pi \cdot f)(x) &= \pi \cdot (f(\pi^{-1} \cdot x)) & f &: \alpha \rightarrow \beta\end{aligned}$$

Note again this universe has non-trivial automorphisms and has symmetry.

This looks just like ZFA (Zermelo-Fraenkel sets with atoms; or a simply-typed version thereof) — and it is.

But there's more going on here than that, because:

1. EZFA(C) is an axiom theory consisting of ZFA + an equivariance axiom-scheme that for any  $f$  and any permutation  $\pi$  of  $\mathbb{A}$ :

$$f(\pi \cdot x_1, \dots, \pi \cdot x_n) = \pi \cdot f(x_1, \dots, x_n).$$

In ZFA, equivariance is **provable** per  $f$ , at a **proof-cost** which increases with the complexity of  $f$ .

In EZFA, The cost of equivariance is constant: direct appeal to equivariance axiom-scheme for  $f$ .

It sounds obvious, but really it's not:

**Claim 1:** A theorem-prover based on ZF(C), or even on ZFA(C), is **not practical** for representing generative name-symmetries, because the cost of equivariance proofs tends to dominate.

**Claim 2:** A theorem-prover based on EZFA(C) will avoid this.

2. We explicitly use atoms in this universe to represent name/nonce/pointer-like data.

While simple — Need a datatype of names? Use  $\mathbb{A}$ ! — this is a significant departure from standard practice, which typically boils down to  $\mathbb{N}$  + explicit symmetry proofs.

We build nominal constructs in this universe, including equivariance,  $\mathbb{U}$ , and  $\#$ .

These are embedded in the EZFAC universe but their definitions, usage, and relationship with the broader EZFAC universe, are far from obvious.

Think of ordinals: they're embedded in the ZF universe and the definitions are simple enough ... but ordinal theory is unintuitive to most humans, and requires study!

3. We can assume the Axiom of Choice (and inaccessible cardinals, etc, if we want), and we note this is consistent with equivariance.

For, suppose  $f : \text{pow}^*(X) \rightarrow X$  is a choice function on  $X$ . Then  $\pi \cdot f : \text{pow}^*(\pi \cdot X) \rightarrow \pi \cdot X$  is a choice function on  $\pi \cdot X$ .

While simple to prove, it is not widely appreciated that we can have consistently combine symmetries **and** Choice.

# Tooling

Tooling is underdeveloped. There is no programming language or theorem-prover that admits a datatype of EZFAC-style atoms as a first-class entity.

It's a chicken-and-egg problem but still the lack is surprising, because generative name-symmetry turns up **all over** computer science. IMO there's a straight line leading back to the lack of automorphisms of the standard cumulative hierarchy.

(Story of <https://arxiv.org/abs/2003.14271>.)

Note: it is possible to add name-like symmetries to Haskell.

## This talk in context

There's a reasonable amount of research into nominal techniques.

In Brazil, I note there is excellent work on nominal rewriting in particular — that's the theory of rewriting, enriched with explicit name and binding constructs based on what (for the purposes of this talk) I can call an EZFAC model.

There's nominal automata, nominal algebra . . . and much more.



# This talk in context

I myself would be interested in (non-exhaustive list):

- ▶ **Practice.** Improving the tooling.
- ▶ **Theory.** Developing nominal algebra, because I think it's a very beautiful.

My point for this talk is a high-level observation:

1. This is based on something rather deep: namely, a re-thinking of mathematical foundations.
2. What nominal foundations really are is — two decades after the original publications — still something we can think about deeply and critically.

Thank you for listening.

## Appendix

...just in case somebody asks  
or I want to refer to it ...

# Background info

Nominal techniques go back to two papers:

- ▶ **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).

# Sources

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.

# Nominal sets definition

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

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

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

Example:

- ▶  $\mathbb{A}$  is a  $\Sigma_{\mathbb{A}}$ -set where  $\pi \cdot a = \pi(a)$ .
- ▶ Syntax of  $\lambda$ -terms over atoms as variable symbols is a  $\Sigma_{\mathbb{A}}$ -set where  $\pi$  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  $\text{power}(X)$  the powerset of  $X$ , by the **pointwise action**

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

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

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

## Nominal sets definition

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

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

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

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

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

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



## 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  $\text{supp}(x) \subseteq_{fin} \mathbb{A}$ .

Examples:

- ▶  $\mathbb{A}$  is a nominal set. If  $a \in \mathbb{A}$  then  $\{a\} \nsubseteq_{fin} \mathbb{A}$ .
- ▶ Finite sets of atoms are a nominal set. If  $A \subseteq_{fin} \mathbb{A}$  then  $A \nsubseteq_{fin} \mathbb{A}$ .
- ▶ Cofinite sets of atoms (complements of finite sets of atoms) are a nominal set. If  $A \subseteq_{fin} \mathbb{A}$  then  $A \nsubseteq_{fin} \mathbb{A} \setminus A$ .
- ▶ 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  $\alpha$ -equivalence is a nominal set (where variables symbols are atoms)  $[t]_{\alpha}$ .  
 $fV(t) \nsubseteq_{fin} [t]_{\alpha}$  (in words: an  $\alpha$ -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  $\alpha$ -equivalence then  $a \#[t]_\alpha$  when  $a \notin \text{fv}(t)$  ( $a$  is not free in  $t$ ).

# Boolean algebras in a nominal context

Consider a Boolean algebra  $\mathfrak{B}$  in nominal sets (i.e. a Boolean algebra whose underlying set happens to also be a nominal set).

To what extent might  $\mathfrak{B}$  **already** be a model of first-order logic?

Note that if  $x \in \mathfrak{B}$  then we can define:

$$\forall a.x = \bigvee \{x' \leq x \mid a \# x'\},$$

assuming this exists.

$\forall a.x$  is the greatest thing below  $x$  for which  $a$  is fresh.

## Quantifiers in nominal BAs

If  $x \in \mathfrak{B}$  then  $\forall a.x$  is the greatest  $a$ -fresh lower bound for  $x$ .

To see why this is a natural definition, rewrite in sequent style:

$$\frac{x' \leq x \quad a \# x'}{x' \leq \forall a.x}$$

Compare with the forall-right intro-rule:

$$\frac{\Gamma \vdash \phi \quad (a \notin \text{fv}(\Gamma))}{\Gamma \vdash \forall a.\phi.}$$

Look familiar?

## Substitution / $\sigma$ -action

We can build a simple account of FOL just with what we already have, but it is better if we bring in substitution.

FOL and  $\lambda$ -calculus have two name-related structures:

- ▶ binders  $\forall$  and  $\lambda$ , and
- ▶ **substitution**  $[a \mapsto u]$ , which I will call a  **$\sigma$ -action**.

## $\sigma$ -algebra

A **termlike sigma-algebra** is a nominal set  $T$  along with operations

- ▶  $\partial : \mathbb{A} \rightarrow T$  and
- ▶  $T \times \mathbb{A} \times T \rightarrow T$  written  $x[a \mapsto u]$ ,

satisfying the following equations:

$$\begin{aligned}(\sigma\#) \quad a\#x &\Rightarrow x[a \mapsto u] = x \\(\sigma\alpha) \quad b\#x &\Rightarrow x[a \mapsto u] = ((b \ a) \cdot x)[b \mapsto u] \\(\sigma\sigma) \quad a\#v &\Rightarrow x[a \mapsto u][b \mapsto v] = x[b \mapsto v][a \mapsto u[b \mapsto v]] \\(\sigma\text{id}) &x[a \mapsto \partial a] = x \\(\sigma\mathbf{a}) &\partial a[a \mapsto u] = u\end{aligned}$$

This is a nominal algebra distillation of the essential features of a substitution action.

A general  **$\sigma$ -algebra** is a nominal set  $X$  along with a  $\sigma$ -action  $X \times \mathbb{A} \times T \rightarrow X$  over some termlike  $\sigma$ -algebra.

# $\sigma$ -algebra

We can fine-tune these axioms:

$$\begin{aligned}(\sigma\#) \quad a\#x &\Rightarrow x[a\mapsto u] = x \\(\sigma\alpha) \quad b\#x &\Rightarrow x[a\mapsto u] = ((b\ a)\cdot x)[b\mapsto u] \\(\sigma\sigma) \quad a\#v &\Rightarrow x[a\mapsto u][b\mapsto v] = x[b\mapsto v][a\mapsto u[b\mapsto v]] \\(\sigma\mathbf{id}) & \quad x[a\mapsto \partial a] = x \\(\sigma\mathbf{a}) & \quad \partial a[a\mapsto u] = u\end{aligned}$$

- ▶ Drop  $(\sigma\alpha)$  to obtain a kind of ‘fusion’ operation: without  $(\sigma\alpha)$ ,  $[a\mapsto u]$  binds  $u$  to  $a$  but leaves the ‘port’  $a$  open for further bindings.
- ▶ Drop  $(\sigma\#)$  to permit communication on any port.
- ▶ Drop  $\partial$ ,  $(\sigma\mathbf{a})$ , and  $(\sigma\mathbf{id})$  if e.g. we only want to substitute for closed terms.

## Dualise to $\sigma^*$ -algebra

Consider a  $\sigma$ -algebra  $X$  over a termlike  $\sigma$ -algebra  $T$ . What structure does the powerset  $power(X)$  have?

We obtain an algebra dual  $\sigma$ -algebras; I call this **amgis**-algebra in my papers; here I will call it a  $\sigma^*$ -algebra:

$$\begin{aligned}(\sigma^*\sigma) \quad a \# v &\Rightarrow X[b \mapsto v]^* [a \mapsto u]^* = X[a \mapsto u [b \mapsto v]]^* [b \mapsto v]^* \\(\sigma^*\mathbf{id}) \quad &X[a \mapsto \partial a]^* = X\end{aligned}$$



## Dualise to $\sigma^*$ -algebra

Note that we consider the full powerset  $power(X)$ , not the finitely-supported powerset  $power_{fs}(X)$ .

This is important: intuitively  $X \in power(X)$  may be a **point** (e.g. a maximally consistent theory). It need not be finitely supported.

More precisely:

- ▶ in **classical** theories (like FOL)  $X$  will definitely not have finite support;
- ▶ in **intuitionistic** theories (like  $\lambda$ -calculus)  $X$  may have finite support, or it may not.

In summary:

- ▶ If  $X$  is a  $\sigma$ -algebra with  $(\sigma\#)$ ,  $(\sigma\alpha)$ ,  $(\sigma\sigma)$ ,  $(\sigma\mathbf{id})$ , and  $(\sigma\mathbf{a})$ , then
- ▶  $power(X)$  the set of all subsets of  $X$  is naturally a  $\sigma^*$ -algebra with  $(\sigma^*\sigma)$  and  $(\sigma^*\mathbf{id})$ .