Make it so: Imperatival foundations for mathematics

Neil Barton[1], Ethan Russo[2], and Chris Scambler[3]

3 March 2023

[1]IFIKK, Universitetet i Oslo.

[2]New York University

[3]All Souls College

Introduction: The Technical Program of Procedural Postulationism

Procedural Postulationism: the existence of objects in and truth of claims about a mathematical domain are secured by postulating instructions or procedures for the creation of that domain

For example: "Introduce a number with no successors! Then introduce a successor for that number!"

On the Procedural Postulationist account, we might gain knowledge of the mathematical domain by somehow deriving its truths from the rules of its construction

Introduction (cont.)

To derive truths from the rules of construction, we need, at minimum:

1. A language for expressing imperatival rules;
2. A logic laying out what a given imperative creates and makes true

Our aims:

- Set up a language and logic
- Specify suitable imperatives from which we can derive PA_2 and ZFC_2
- Assess, based on our attempt at the above, some of the philosophical prospects of Procedural Postulationism

Imperatival Logic

The Language of Postulation

We will frame our arguments in positive, free HOL with base types e, t, ι.

e is the type of nominal terms, t the type of declarative sentences, and ι is the type of *imperatives*

When σ and τ are types and $\tau \neq e$, then $\sigma \rightarrow \tau$ is a type

When τ is a type, $[\tau]$ is a type (intuitively, the type of pluralities of type-τ things)

When $t : \tau$ and $x : \sigma$, $\lambda x.t(x) : \sigma \rightarrow \tau$

When $t_1 : \sigma \rightarrow \tau$ and $t_2 : \sigma$, $t_1(t_2) : \tau$

Convention: $F, G, H\ldots$ as variables for intensional properties (type $\sigma \rightarrow t$), and X, Y, Z for pluralities (type $[\sigma]$)

We have axioms that existence is closed under application, and that, roughly, a plurality exists when and only when all its members do

Postulational Vocabulary

In addition to the usual logical constants, we have special constants $\ulcorner !^{\sigma} \urcorner$, $\ulcorner ; \urcorner$, $\ulcorner \rightarrow \urcorner$, and $\ulcorner \forall^{\sigma}_{i} \urcorner$.

- $!F$ is an imperative when $F : \sigma \rightarrow t$, ("Make an F!")
- $i; j$ is an imperative when $i, j : \iota$, ("Do i, and then j!")
- $p \rightarrow i : \iota$ is an imperative when $p : t$ and $i : \iota$, ("If p, do i!")
- $\forall^{\sigma}\Psi$ is an imperative when $\Psi : \sigma \rightarrow \iota$ ("Do Ψ to everything!")

The Language of Postulation: Postulational Modalities

How do we express that something is so as the result of an imperative?

Postulational modalities $\langle i \rangle$ and $[i]$ for an imperative i:

- $[i]\phi$ iff, no matter how i is carried out, ϕ is true
- $\langle i \rangle\phi$ iff, for some way of carrying out i, ϕ is true

An example: Let $i =$ Let there be a table!

Then: $[i]$ There is a table, $\langle i \rangle$ There is a dinner table, $\langle i \rangle$ There is a coffee table

We also help ourselves to a general $\Box$ with **S5** modal logic (and without either Barcan Formula)

General Principles

What principles should govern these postulational modalities?

We take all postulational modalities to be governed by the K axiom, Converse Barcan, and the rule of Necessitation.

We also require $\Box\phi \supset [i]\phi$

We don't require $\langle i\rangle\top$ for every i

There is thus a distinction between *executable* commands ($\langle i\rangle\top$) and *non-executable* ($\neg\langle i\rangle\top$)

Executable commands can be carried out, while there is no way of carrying out non-executable commands (e.g., $!x.x \neq x$)

Principles for Specific Command Types

There are some principles we can give for imperatives of particular forms:

- **Eco1** $\neg\exists F \supset \exists X(\forall y Xy \wedge [!F]\exists x(Fx \wedge \forall y(\neg Xy \supset y = x))$
- **Eco2** $\exists F \supset (\phi \equiv [!F]\phi)$
  From these two, follows **Make**: $[!F]\exists F$
- **Then** $[i; j]\phi \equiv [i][j]\phi$
- **If1** $p \supset ([p \rightarrow i]\phi \equiv [i]\phi)$
- **If2** $\neg p \supset ([p \rightarrow i]\phi \equiv \phi)$

Universal Commands

The intuition: the genie executes $\forall x i(x)$ by splitting into many copies (one for each x), and having each copy perform $i(x)$ for some x. The overall result of the command is the “union” or “pooling together” of the results for each $i(x)$.

- **All1** $\exists x[i(x)]Eo \supset [\forall x i(x)]Eo$
- **All2** $\langle \forall x i(x)\rangle Eo \supset \exists x\langle i(x)\rangle Eo$
- **All3** $\forall x \text{😴}(i(x)) \supset \text{😴}(\forall x i(x))$
  Where $\text{😴}(i)$ is defined as: $\forall p([i]p \equiv p)$

Executability again

We do not see any good way of telling on something like syntatic grounds whether an imperative will be executable or not.

Note, however, that specific principles allow us to relate the executability of complex commands to that of their parts:

- $\langle i; j\rangle\top \equiv \langle i\rangle\langle j\rangle\top$
- $\langle p \rightarrow i\rangle\top \equiv (p \supset \langle i\rangle\top)$

We cannot, however, derive:

- $\langle\forall x i\rangle\top \equiv \forall x\langle i\rangle\top$

Determinism

Lastly, and perhaps most controversially, we adopt:

- D $\langle i \rangle \phi \supset [i]\phi$

This principle is clearly false for commands like “Build a chair!” and even mathematical commands like “Extend this line!”

It’s possible to take D as a stipulation: it restricts the class of commands that we are interested in to those that are deterministic

Defining Indefinite Iteration

Indefinite Iteration

Fine's postulational language contains has an important addition we lack: an indefinite iteration operator $*$

*i** means to do *i*, and then again, and again, and again... through not only finite iterations, but also transfinite iterations

Such power is necessary for the case of Set Theory, where our command will essentially be "For every plurality, create a set of that plurality! And again! And again! And again! ..."

Defining Indefinite Iteration

With our higher-order resources, it's possible to define a term that certainly acts a lot like indefinite iteration

Using the ancestral, we can define a term X applying exactly the finite iterations of i: i, $i; i$, $i; i; i$...

But how can we represent the "limit" of all these iterations $i; i; i; i \ldots$

Our suggestion is we take the imperative $i^\omega = \forall j(Xj \rightarrow j)$ ("Do all the finite iterations!")

Anything some finite iteration of i makes, i^ω makes. And anything i^ω makes, some finite iteration of i makes

Defining Indefinite Iteration (cont.)

Generalizing, we characterize not only the finite iterations, but all iterations of i as the least plurality I_i closed under these conditions:

- i is an iteration of i
- If j is an iteration of i, then $j; i$ is an iteration of i
- If $X \subseteq I_i$, then $\forall j(Xj \to j)$ is an iteration of i

We set $i^* = \forall j(I_i(j) \to j)$

Note this definition provides for a form of induction on iterations

A nice result: Fixed Point Theorem for i^*

We say i is essentially creative just in case: necessarily, if it does something, then it creates something

Plausibly, all the postulationist's commands are essentially creative. But in fact we can prove (by induction on terms) that all the imperatives we use are essentially creative

For any essentially creative i, after you do i^*, i does nothing ($[i^*]$😴(i))

Proof. We consider only the case of executable i^*. Since i is essentially creative, we only have to show that you don't get anything new by doing i after i^*: that is, you don't get anything from $i^*; i$ which you don't get from i^*.

But since i^* is an iteration of i, $i^*; i$ is, and so anything you get from $i^*; i$ you get from i^* by All1.

Imperatival Mathematics

Strategy

Our general strategy (following Fine's suggestion) is

1. Expand the language of our imperatival logic with **postulational predicates**
2. Lay down **postulational constraints**
3. Give **postulates** in the postulational predicates
4. Use the logic to prove that, if you do the postulates, you get the mathematical domain you want.

Arithmetic: Language and Postulational Constraints

We expand our language by new predicates N and S.

Successor Uniqueness	$Sxy \wedge Sxz \supset y = z$
Predecessor Uniqueness	$Syx \wedge Szx \supset y = z$
Number rigidity	$Nx \supset \Box Nx$
Succession is Numerical	$Sxy \supset Nx \wedge Ny$
Predecessor Rigidity	$\forall y(Syx \supset \Box Fy) \equiv \Box\forall y(Syx \supset Fy))$

Arithmetic Postulates

Let ζ be the command $!x.Nx$...

and let σ be the command $\forall x(Nx \rightarrow !y.Sxy)$...

and let Num be the command $\zeta ; \sigma^*$.

Arithmetic: Main result

Main theorem

If there are no numbers then $[Num]PA_2^N$.

- Case 1: $\neg\langle Num\rangle\top$
- Case 2:
    - For zero, $[\zeta]\exists xNx \wedge \forall y\neg S(y,x)$...
    - For successor, use fixed point lemma...
    - For induction, use induction!

Corollary

If there are no numbers and *Num* is executable then $\Diamond PA_2^N$.

Set theory: Language and Constraints

We expand our language by new predicates S and $\in$.

We write $Set(y, X)$ for $\forall z(z \in y \equiv Xz)$.

Extensionality	$\forall F \forall x \forall y(Set(x, F) \wedge Set(y, F) \supset x = y)$
(Plurality-Uniqueness)	$(\forall F \forall G(Set(x, F) \wedge Set(x, G) \supset F = G))$
Sethood Rigidity	$Sx \supset \Box Sx$
Only sets have members	$x \in y \supset Sy$
Membership Rigidity	$\forall x(\forall y(y \in x \supset \Box Fy) \equiv \Box \forall y(y \in x \supset Fy))$

Russell's Paradox

Let $Pow := \forall X ! y . Set(y, X)$.

Imperatival Russell's Paradox

Pow^* is not executable.

Say i is a necessary difference maker iff $\Box\neg$😴(i). Then:

Generalized RP

No essentially creative NDM has executable indefinite iteration.

Hedges and the While-do lemma

Given an imperative *i*, a command of the a form

$$(p \to i)^*$$

has the force of ‘while *p*, do *i*’.

We can think of it as a ‘hedged’ iteration of *i*.

While-do lemma

If *i* is EC NDM, then $(p \to i)^*$ is executable iff $\langle (p \to i)^* \rangle \neg p$.

Set theory: Main result

Definition: h_{zfc}

Let h_{zfc} be the proposition that the (von neumann ordinals) are either

- finite, or
- bounded, or
- singular, or
- **weak**.

Main result

Main result

Let η be $h_{zfc} \rightarrow Pow$, and let $hSet := \eta^*$.

If there are no sets, $[hSet]ZFC_2^S$.

Corollary

If there are no sets, and *hSet* is executable, then $\Diamond ZFC_2^S$.

Consistency

Consistency

Fine says:

> *Now what is remarkable is that, once consistency claims are formulated in this way, it is possible to provide convincing demonstrations of their truth that are purely modal in character and that make no appeal either to models or proofs or to any other kind of abstract object.*
> *(Fine, 'Our Knowledge of Mathematical Objects', p. 98)*

Let's explore this idea in light of our current proposal.

Consistency

Let's start with the positives:

1. We can prove the consistency of axioms by assuming the executability of commands.
2. We think it's plausible that our imperativalist does get some epistemic gain regarding beliefs about consistency.
    - Apart from (controversial) enumerative induction arguments, why do we think arithmetic/set theory are consistent?
    - A natural answer: We can envisage "building" a model for the axioms using commands that are individually intuitively executable, and iterating them many times.
    - This is what imperatival logic makes formally precise.

However, we might want more, and to *prove* the consistency of Num/Set_h inductively.

Consistency

This is an itemization of a passage from ‘Our Knowledge of Mathematical Objects’:

- Say that a postulational rule i is strongly consistent, or conservative, if it is necessarily consistent ($\Box\langle i\rangle\top$), that is, consistent no matter what the domain.
- Then it should be clear that the simple rule ζ is conservative and that the simple rule $!y.(Ny \wedge Sxy)$ is conservative whatever the object x.
- It should also be clear that each of the operations for forming complex rules will preserve conservativity. For example, if i and j are conservative then so is $i; j$
- But it then follows that *Num* is consistent, as is any other rule that is formed from conservative simple rules by means of the operations for forming complex rules.

Upwards Transmission

- Whilst we can prove in imperatival logic that if *i* and *j* are executable then their sequential command *i*; *j* is executable...
- For many other commands (e.g. universal commands/indefinite iterations) this isn't clear.
- e.g. Fine seems to assume that we have the upwards transmission of executability for all complex commands (including indefinite iteration).

Upwards Transmission

- Since we've defined indefinite iteration in terms of the universal command, we can concentrate on the latter.
- In that context, we can characterize the upwards transmission of executability by appealing to the following principle:
- **Universal Upwards Transmission** (**UUT**): $\forall x \langle i \rangle \top \equiv \langle \forall x i \rangle \top$ (if you can do a command to each x separately, you can do said command to everything in parallel).
- If we add **UUT**, we get our consistency proofs in the manner Fine suggests.
- But is **UUT** plausible/true?

Upwards Transmission

- There are problems with **UUT**.
- **Problem 1:** Resource-scarcity counterexamples. Suppose you have a single handle *h*, and 9001 hammerheads. Consider the imperative ”Join *x* to *h*!”.
- **Problem 2:** Paradox. Suppose, as Fine wants to, that *Pow* is strongly consistent (i.e. necessarily executable).
- But then, there's an inductive argument using **UUT** to $\langle Pow^* \rangle \top$. Contradiction!

Upwards Transmission

- If we want the necessary executability of *Pow*, then **UUT** must fail.
- But given **UUT** fails, there must be some iterations such that each of them is individually doable, but the command to do all of them is not.
- But given that each is individually doable, what stops us from doing all of them (apart from the big stick)?
- What separates the bad pluralities of imperatives from the good ones?

Upwards Transmission

- **Question:** Are there consistent restrictions of **UUT** that are (1.) well-motivated, and (2.) sufficiently strong for proving executability?
- A natural move: Find a "nice" property/plurality P of imperatives, and assert that if every imperative with P is doable individually then they all are together.
- For example, we can define the finite version of **UUT** (call this **Finite-UUT**): "If i is an imperative, and all finite iterations of i are executable, then so is the command to do every finite iteration of i".
- The executability of *Num* follows from **Finite-UUT**.
- Two questions:
  1. How much of an advance is this? If we bolt on a principle that says we can do ω-length iterations, is that any better than just brute claiming that *Num* is executable?
  2. What about other restricted versions of **UUT** (e.g. for set theory)?

Upwards Transmission

- A different approach is to accept **UUT** and curtail what imperatives are executable.
- This opens the door to a kind of universe-imperativalist: You can execute *Pow* to obtain a maximal domain for set theory, whereupon *Pow* fails to be executable (i.e. we give up the necessary executability of *Pow*).
- But what stops us from always being able to do *Pow* (apart from the big stick)?
- Whichever way you go, there's a revenge problem lurking.

Thanks!

Thanks for listening!