Flexible Foundations of Mathematics

Numerous Numerosity Focus Workshop

Ciarán M. Dunne, joint work with J. B. Wells and Fairouz Kamareddine

June 1, 2021

Numbers in the Foundations of Mathematics

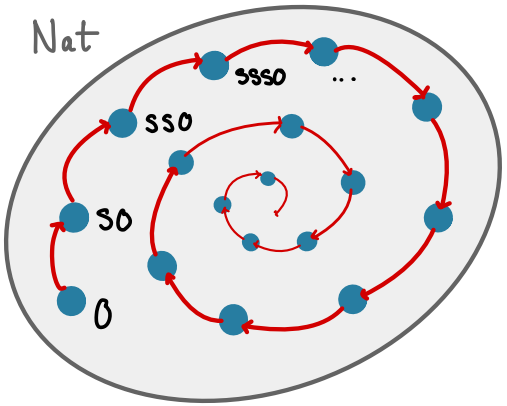
- A foundation of mathematics is a small formal system that attempts to model the practice of mathematicians.

Numbers in the Foundations of Mathematics

- A foundation of mathematics is a small formal system that attempts to model the practice of mathematicians.
- A foundation must provide explanations of common mathematical concepts.

Numbers in the Foundations of Mathematics

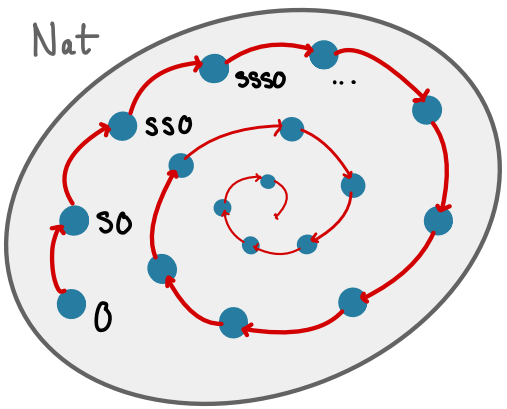
- A foundation of mathematics is a small formal system that attempts to model the practice of mathematicians.
- A foundation must provide explanations of common mathematical concepts.
- The work of Peano and Dedekind gave a formalization of the natural numbers by characterising their structure with axioms for symbols ($\mathbf{0}$ :: Nat, $\mathbf{S}$ :: Nat $\Rightarrow$ Nat)



Numbers in the Foundations of Mathematics

- A foundation of mathematics is a small formal system that attempts to model the practice of mathematicians.
- A foundation must provide explanations of common mathematical concepts.
- The work of Peano and Dedekind gave a formalization of the natural numbers by characterising their structure with axioms for symbols ($\mathbf{0}$ :: Nat, $\mathbf{S}$ :: Nat $\Rightarrow$ Nat)

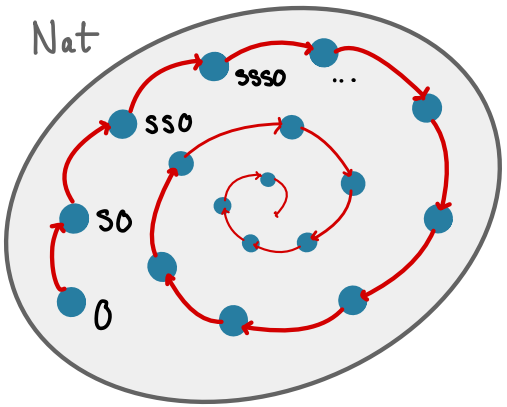
- $\forall m, n \,.\, m = n \leftrightarrow \mathbf{S}\, m = \mathbf{S}\, n$



Numbers in the Foundations of Mathematics

- A foundation of mathematics is a small formal system that attempts to model the practice of mathematicians.
- A foundation must provide explanations of common mathematical concepts.
- The work of Peano and Dedekind gave a formalization of the natural numbers by characterising their structure with axioms for symbols ($\mathbf{0}$ :: Nat, $\mathbf{S}$ :: Nat $\Rightarrow$ Nat)

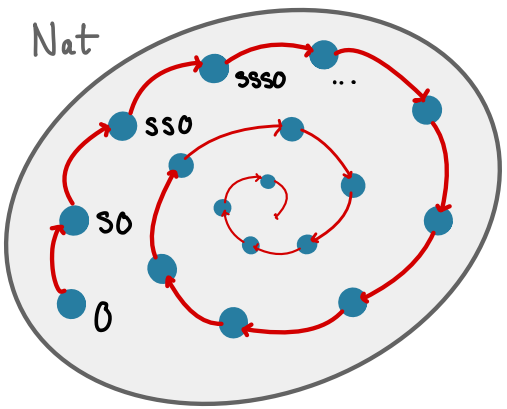
- $\forall m, n \,.\, m = n \leftrightarrow \mathbf{S}\, m = \mathbf{S}\, n$
- $\forall n \,.\, \mathbf{S}\, n \neq \mathbf{0}$



Numbers in the Foundations of Mathematics

- A foundation of mathematics is a small formal system that attempts to model the practice of mathematicians.
- A foundation must provide explanations of common mathematical concepts.
- The work of Peano and Dedekind gave a formalization of the natural numbers by characterising their structure with axioms for symbols ($\mathbf{0}$ :: Nat, $\mathbf{S}$ :: Nat $\Rightarrow$ Nat)

- $\forall m, n \,.\, m = n \leftrightarrow \mathbf{S}\, m = \mathbf{S}\, n$
- $\forall n \,.\, \mathbf{S}\, n \neq \mathbf{0}$
- $\forall P \,.\, P\, \mathbf{0} \wedge (\forall n \,.\, P\, n \rightarrow P\, (\mathbf{S}\, n)) \rightarrow (\forall n \,.\, P\, n)$



Numbers in the Foundations of Mathematics

- But the natural numbers do not exist in a vacuum!
  In modern math, they stand in relation to other objects:

Numbers in the Foundations of Mathematics

- But the natural numbers do not exist in a vacuum!
  In modern math, they stand in relation to other objects:
  - sets of numbers: $\mathbb{N}$, $\{\, n \in \mathbb{N} \mid \exists k \,.\, n = 2k \,\}$, $\mathbb{N} \subseteq \mathbb{R} \subseteq \mathbb{C}$

Numbers in the Foundations of Mathematics

- But the natural numbers do not exist in a vacuum!
  In modern math, they stand in relation to other objects:
  - sets of numbers: $\mathbb{N}$, $\{\, n \in \mathbb{N} \mid \exists k \,.\, n = 2k \,\}$, $\mathbb{N} \subseteq \mathbb{R} \subseteq \mathbb{C}$
  - inside other structures: $(2, 3)$, $f : \mathbb{N} \to \mathbb{N}$, $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$

Numbers in the Foundations of Mathematics

- But the natural numbers do not exist in a vacuum!
  In modern math, they stand in relation to other objects:
  - sets of numbers: $\mathbb{N}$, $\{\, n \in \mathbb{N} \mid \exists k \,.\, n = 2k \,\}$, $\mathbb{N} \subseteq \mathbb{R} \subseteq \mathbb{C}$
  - inside other structures: $(2, 3)$, $f : \mathbb{N} \to \mathbb{N}$, $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$
- We need theories with powerful concepts capable of representing a wide range of mathematical objects.

Numbers in the Foundations of Mathematics

- But the natural numbers do not exist in a vacuum!
  In modern math, they stand in relation to other objects:
  - sets of numbers: $\mathbb{N}$, $\{\, n \in \mathbb{N} \mid \exists k \,.\, n = 2k \,\}$, $\mathbb{N} \subseteq \mathbb{R} \subseteq \mathbb{C}$
  - inside other structures: $(2, 3)$, $f : \mathbb{N} \to \mathbb{N}$, $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$
- We need theories with powerful concepts capable of representing a wide range of mathematical objects.
  - In a set theory, natural numbers can be represented as a sequence of sets called the von Neumann natural numbers:
    $0 := \emptyset$, $1 := \{0\}$, $2 := \{0, 1\}$, $3 := \{0, 1, 2\}$, $\ldots$

Numbers in the Foundations of Mathematics

- But the natural numbers do not exist in a vacuum!
  In modern math, they stand in relation to other objects:
  - sets of numbers: $\mathbb{N}$, $\{\, n \in \mathbb{N} \mid \exists k \,.\, n = 2k \,\}$, $\mathbb{N} \subseteq \mathbb{R} \subseteq \mathbb{C}$
  - inside other structures: $(2, 3)$, $f : \mathbb{N} \to \mathbb{N}$, $\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$
- We need theories with powerful concepts capable of representing a wide range of mathematical objects.
  - In a set theory, natural numbers can be represented as a sequence of sets called the von Neumann natural numbers:
    $0 := \emptyset$, $1 := \{0\}$, $2 := \{0, 1\}$, $3 := \{0, 1, 2\}$, $\ldots$
  - In a type theory, natural numbers can be represented by an inductive data type:
    $\mathsf{Nat}\ n := \mathsf{zero}$
    $\quad\quad\quad\quad\quad\quad \mid\ \mathsf{succ}\, n$

Foundations in Proof Assistants

- As well as being of philosophical interest, foundational theories have practical applications in mathematical software, particularly in digital proof assistants.

Foundations in Proof Assistants

- As well as being of philosophical interest, foundational theories have practical applications in mathematical software, particularly in digital proof assistants.
- Aim: provide software to help mathematicians prove theorems.

Foundations in Proof Assistants

- As well as being of philosophical interest, foundational theories have practical applications in mathematical software, particularly in digital proof assistants.
- Aim: provide software to help mathematicians prove theorems.
- Still unclear what the best foundation is for achieving this aim.

Foundations in Proof Assistants

- As well as being of philosophical interest, foundational theories have practical applications in mathematical software, particularly in digital proof assistants.
- Aim: provide software to help mathematicians prove theorems.
- Still unclear what the best foundation is for achieving this aim.
    - Provers that use a type-theoretic foundation typically have better automation, but the formulation of original text is quite different to how a mathematician might write it.
      (Coq, Isabelle/HOL, Lean)

Foundations in Proof Assistants

- As well as being of philosophical interest, foundational theories have practical applications in mathematical software, particularly in digital proof assistants.
- Aim: provide software to help mathematicians prove theorems.
- Still unclear what the best foundation is for achieving this aim.
  - Provers that use a type-theoretic foundation typically have better automation, but the formulation of original text is quite different to how a mathematician might write it.
    (Coq, Isabelle/HOL, Lean)
  - A set-theoretic foundation is what mathematicians would "say" is their foundation, but existing implementation require lots of human intervention.
    (Mizar, Metamath, Isabelle/ZF)

Accidental Theorems & Representation Overlap

- Implementations of set theories use a small number of "types" (i.e. a type of objects, a type of propositions) but all of the objects of the theory are of type "set".

Accidental Theorems & Representation Overlap

- Implementations of set theories use a small number of "types" (i.e. a type of objects, a type of propositions) but all of the objects of the theory are of type "set".
- As a result, these implementations suffer from accidental theorems, because everything is a set.
  - e.g. it is possible to ask "is $42 \in \pi$?"

Accidental Theorems & Representation Overlap

- Implementations of set theories use a small number of "types" (i.e. a type of objects, a type of propositions) but all of the objects of the theory are of type "set".
- As a result, these implementations suffer from accidental theorems, because everything is a set.
  - e.g. it is possible to ask "is $42 \in \pi$?"
- Implementations of type theories encourage users to split up concepts into as many types as possible, to take advantage of type-based automation techniques.
  - questions like "is $42 \in \pi$?" and operations like $\mathcal{P}(2)$ are forbidden

Accidental Theorems & Representation Overlap

- Implementations of set theories use a small number of "types" (i.e. a type of objects, a type of propositions) but all of the objects of the theory are of type "set".
- As a result, these implementations suffer from accidental theorems, because everything is a set.
  - e.g. it is possible to ask "is $42 \in \pi$?"
- Implementations of type theories encourage users to split up concepts into as many types as possible, to take advantage of type-based automation techniques.
  - questions like "is $42 \in \pi$?" and operations like $\mathcal{P}(2)$ are forbidden
- This creates unnecessary theorems, which must be proved to satisfy the needs of the type system.

Research Aims

- Both of these approaches are reasonable and useful.

Research Aims

- Both of these approaches are reasonable and useful.
- Is it possible to have more flexibility?
  - design a set-theoretic foundation with mathematical objects that are genuinely not sets?
  - portion this universe into types for automation
  - answer "is $42 \in \pi$" as simply false, handle errors like $\mathcal{P}(2)$ with "exception objects".

Research Aims

- Both of these approaches are reasonable and useful.
- Is it possible to have more flexibility?
  - design a set-theoretic foundation with mathematical objects that are genuinely not sets?
  - portion this universe into types for automation
  - answer "is $42 \in \pi$" as simply false, handle errors like $\mathcal{P}(2)$ with "exception objects".
- Our research aims to test whether this hypothesis is workable/practical/sensible/reasonable.

Research Aims

- Both of these approaches are reasonable and useful.
- Is it possible to have more flexibility?
  - design a set-theoretic foundation with mathematical objects that are genuinely not sets?
  - portion this universe into types for automation
  - answer "is $42 \in \pi$" as simply false, handle errors like $\mathcal{P}(2)$ with "exception objects".
- Our research aims to test whether this hypothesis is workable/practical/sensible/reasonable.
- Within this talk, I aim to show how to use our framework to generate a set theory with non-set natural numbers.

Research Aims

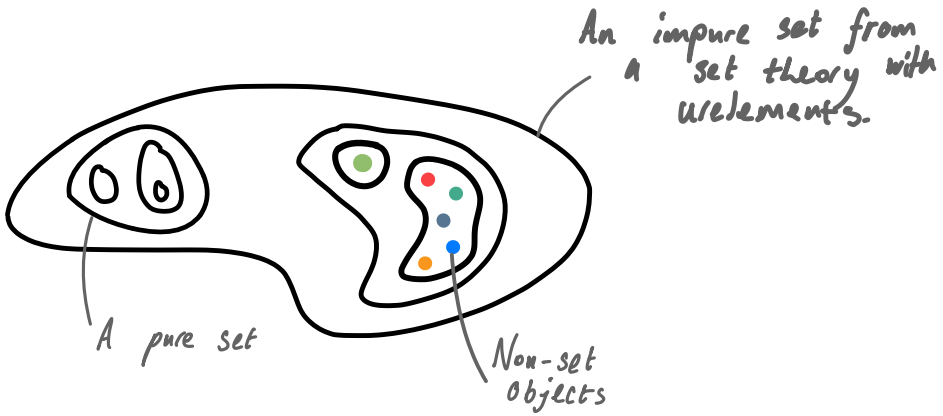
- Both of these approaches are reasonable and useful.
- Is it possible to have more flexibility?
  - design a set-theoretic foundation with mathematical objects that are genuinely not sets?
  - portion this universe into types for automation
  - answer "is $42 \in \pi$" as simply false, handle errors like $\mathcal{P}(2)$ with "exception objects".
- Our research aims to test whether this hypothesis is workable/practical/sensible/reasonable.
- Within this talk, I aim to show how to use our framework to generate a set theory with non-set natural numbers.
- We have also built set theories with non-set ordered pairs, an exception element, and modular combinations of these features.

Axiomatic Set Theory: A Recap

- Traditionally, axiomatic set theories are collections of (first-order or second-order) axioms which characterise a binary relation $\in$ over a domain of individuals.
- The individuals are called sets, and the relation is called set membership — $x \in y$ reads as "x is in y".
- The most popular systems are Zermelo-Fraenkel set theory (ZF), and its variants that add extra axioms (ZFC, TG, etc.)
- The axioms of ZF tell us:
  - when sets are equal (extensionality)
  - combinatorial operations (union, power set, replacement)
  - there is an infinite set (infinity)
  - no infinitely descending chains of sets (foundation)

Generalised Set Theories

- A Generalised Set Theory (GST) (Aczel, 1996) is a theory with:
  - pure sets, which contain only sets "all the way down" to $\emptyset$,
  - non-set objects with no set members,
  - impure sets which may contain a mixture of pure sets, non-set objects, and other impure sets.
- There are simple instances of GSTs with non-set objects called urelements or atoms.
- These objects typically have no "structure", other than existing in a certain multitude.



GST Features

- We generate GSTs by combining user-specified features.
- Each feature is a record consisting of:
  - a signature of symbols for predicates & operators
  - a theory of formulas which specify those symbols
  - a unary predicate from the signature that identifies members, used for combining features
- We generate ZF by using only the $\mathbf{Set}$ feature.
- We combine the $\mathbf{Set}$ and $\mathbf{Nat}$ features to generate a GST with sets and non-set natural numbers — called ZF$\mathbb{N}$.
- We have worked out the details of a $\mathbf{Pair}$ feature and a $\mathbf{Exception}$ feature.

The Set and Nat features

$$
\begin{aligned}
\mathbf{Set} := [\quad \mathsf{sig} &:= \{\in, \mathsf{Set}, \emptyset, \cup, \subseteq, \mathcal{P}, \mathsf{succ}, \mathsf{Inf}, \mathcal{R}\} \\
\mathsf{theory} &:= \{\, \forall a \,.\, a \notin \emptyset, \\
&\quad \forall x \,.\, \mathsf{Set}\, x \leftrightarrow (x = \emptyset \vee \exists y \,.\, y \in x), \\
&\quad x \subseteq y \leftrightarrow (\mathsf{Set}\, x \wedge \mathsf{Set}\, y \wedge \forall a \in x \,.\, a \in y), \\
&\quad \forall [\mathsf{Set}]\, x, y \,.\, (\forall a \,.\, a \in x \leftrightarrow a \in y) \rightarrow x = y, \\
&\quad \forall [\mathsf{Set}]\, x \,.\, \mathsf{Set}\, (\textstyle\bigcup x) \wedge (\forall a \,.\, a \in \bigcup x \leftrightarrow \exists z \in x \,.\, a \in z), \\
&\quad \forall [\mathsf{Set}]\, x \,.\, \forall z \,.\, z \in \mathcal{P}\, x \leftrightarrow z \subseteq x, \\
&\quad \forall [\mathsf{Set}]\, x \,.\, \forall a \,.\, a \in (\mathsf{succ}\, x) \leftrightarrow (a \in x \vee a = x), \\
&\quad \emptyset \in \mathsf{Inf} \wedge \forall x \in \mathsf{Inf} \,.\, \mathsf{succ}\, x \in \mathsf{Inf}, \\
&\quad \forall P \,.\, \forall [\mathsf{Set}]\, x \,.\, (\forall a \in x \,.\, \exists^{\leq 1} b \,.\, P(a, b)) \\
&\quad \rightarrow \mathsf{Set}\, (\mathcal{R}(P, x)) \wedge \forall b \,.\, b \in \mathcal{R}(P, x) \leftrightarrow \exists a \in x \,.\, P(a, b) \,\} \\
\mathsf{iden} &:= \mathsf{Set} \quad]
\end{aligned}
$$

The Set and Nat features

$$
\begin{aligned}
\mathbf{Nat} := [\quad & \mathsf{sig} := \{\mathsf{Nat}, \mathbf{0}, \mathbf{S}\} \\
& \mathsf{theory} := \{\, \mathsf{Nat}\,\mathbf{0}, \quad \mathbf{0} = \mathbf{0}, \quad \forall[\mathsf{Nat}]\, n \,.\, \mathsf{Nat}\,(\mathbf{S}\, n), \\
& \qquad \forall[\mathsf{Nat}]\, m, n \,.\, m = n \leftrightarrow \mathbf{S}\, m = \mathbf{S}\, n, \\
& \qquad \forall[\mathsf{Nat}]\, n \,.\, \mathbf{S}\, n \neq \mathbf{0}, \\
& \qquad \forall P \,.\, P\,\mathbf{0} \to \forall[\mathsf{Nat}]\, m \,.\, (P\, m \to P\,(\mathbf{S}\, m)) \\
& \qquad\qquad \to \forall[\mathsf{Nat}]\, n \,.\, P\, n \quad \} \\
& \mathsf{iden} := \mathsf{Nat} \quad]
\end{aligned}
$$

Combining Features

- To combine features, we take the union of the theories, and generate two extra axioms to ensure all objects are identified by a feature, and that the objects of each feature are distinct.
- In this case:

$$\begin{aligned}\mathsf{ZF}\mathbb{N} := {} & \mathbf{Set}.\mathsf{theory} \cup \mathbf{Nat}.\mathsf{theory} \\ & \cup \{\forall x\,.\,\mathsf{Set}\,x \vee \mathsf{Nat}\,x\} \qquad \text{(either a Set or a Nat)} \\ & \cup \{\forall x\,.\,\neg\,\mathsf{Set}\,x \vee \neg\,\mathsf{Nat}\,x\} \quad \text{(but not both!)}\end{aligned}$$

- $\mathsf{ZF}\mathbb{N}$ can prove all of our usual theorems of set theory in the $\mathbf{Set}$ portion of the domain, and also theorems of arithmetic in the $\mathbf{Nat}$ portion!

Safety of Generated GSTs

- ZF is believed nearly everyone to be consistent.
  - i.e. there is no φ such that ZF $\vdash \varphi$ and ZF $\vdash \neg\varphi$
  - due to nearly a century of intense scrutiny
- Extra axioms could introduce contradictions!
- How do we know that ZF$\mathbb{N}$ is safe to use?
- If we can build a model of a GST Γ_1 in a consistent GST Γ_2, then Γ_1 is consistent.
- Our framework includes a model building kit which may be invoked within any GSTs with the $\mathbf{Set}$ feature.

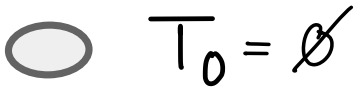
Modelling sets and non-sets

- To show consistency of ZF$\mathbb{N}$, we build a model in ZF?
- ZF$\mathbb{N}$ sets are modelled as ZF ordered pairs with 0 on left, and a set on the right.
  - $\overline{\mathsf{Set}}(x) := \exists x' \,.\, x = \langle 0, x' \rangle$,
  - $x \mathrel{\overline{\in}} y := \exists y' \,.\, y = \langle 0, y' \rangle \wedge x \in y'$
  - $\overline{\mathcal{P}}, \overline{\bigcup}, \overline{\emptyset}, \ldots$ defined similarly.
- ZF$\mathbb{N}$ natural numbers are modelled as ordered pairs with 1 on the left and a von Neumann natural number on the right.
  - $\overline{\mathsf{Nat}}(n) := \exists n' \,.\, n = \langle 1, n' \rangle$,
  - $\overline{\mathbf{0}} := \langle 1, \emptyset \rangle, \quad \overline{\mathbf{S}} \langle 1, n' \rangle := \langle 1, \mathsf{succ}\, n' \rangle$
- Because ordered pairs use the Kuratowski encoding, and the natural numbers use the von Neumann encoding:
  we only use sets to model the objects of ZF$\mathbb{N}$.

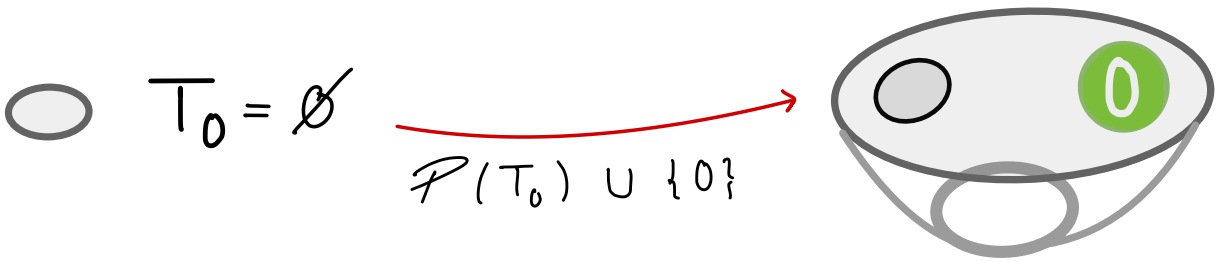
Building a Domain

- We need to build a domain for our model, and show that the relations and operations satisfy ZF$\mathbb{N}$'s axioms on this domain.
- But a domain of a set theory is too big to be a set!
- Instead, tiers of a model must be built by transfinite recursion.
- That is, for each ordinal α, we define a set T_α in terms of T_β, where $\beta < \alpha$.
- If α is an ordinal, then either:
  - $\alpha = \emptyset$, (written as 0)
  - $\alpha = \mathsf{succ}\,\beta$ for some ordinal β,
  - $\mathsf{Limit}(\alpha)$, i.e. $\forall \beta \in \alpha \,.\, \mathsf{succ}\,\beta \in \alpha$.

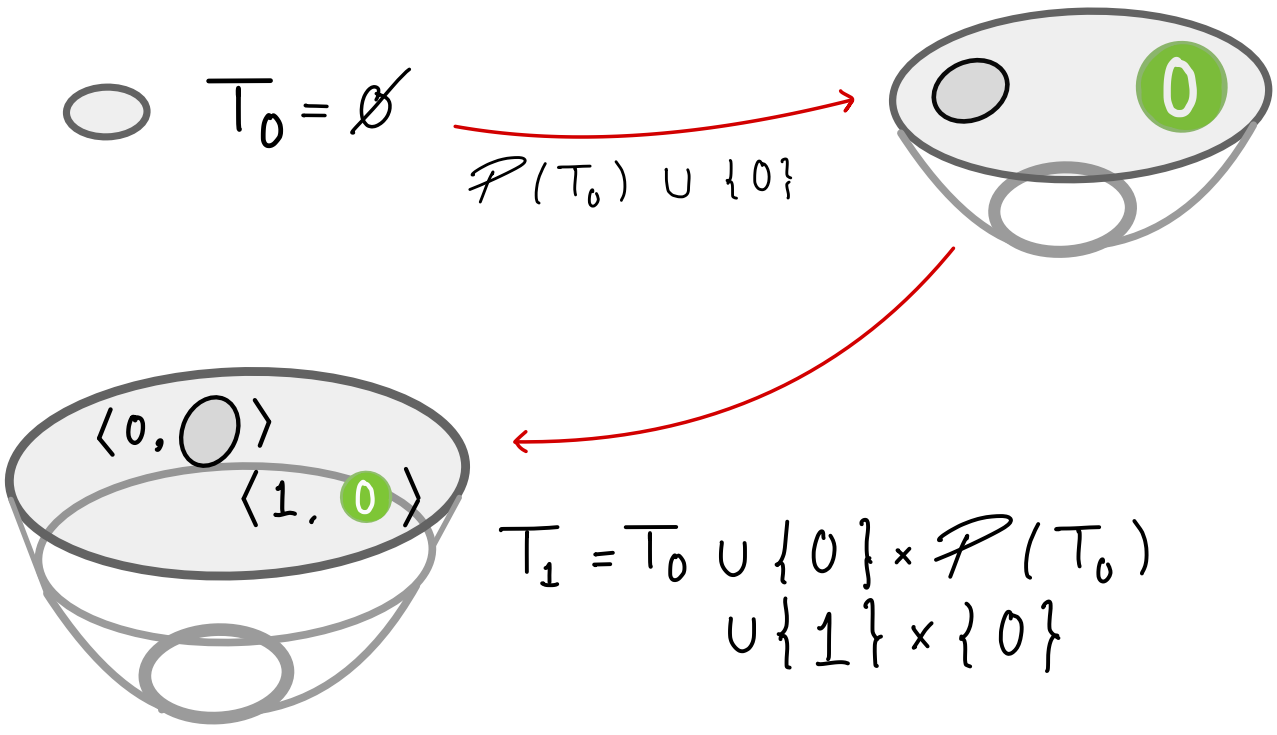
Building a Domain



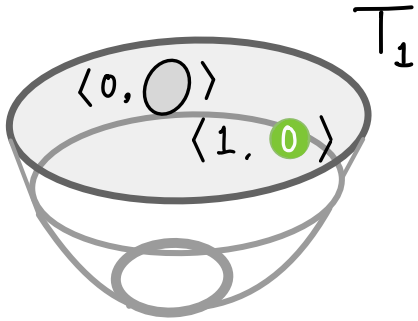
Building a Domain



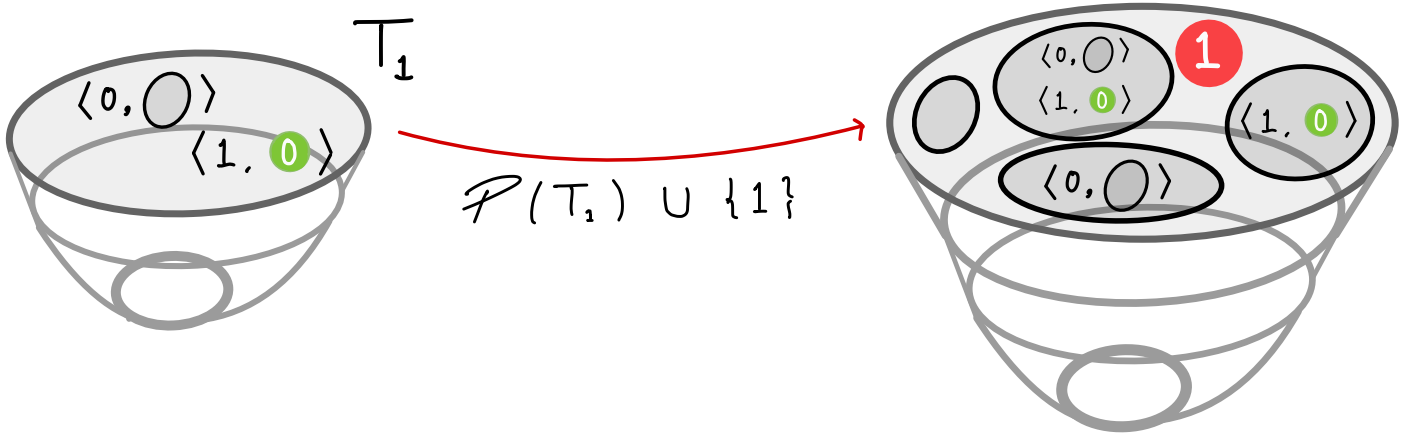
Building a Domain



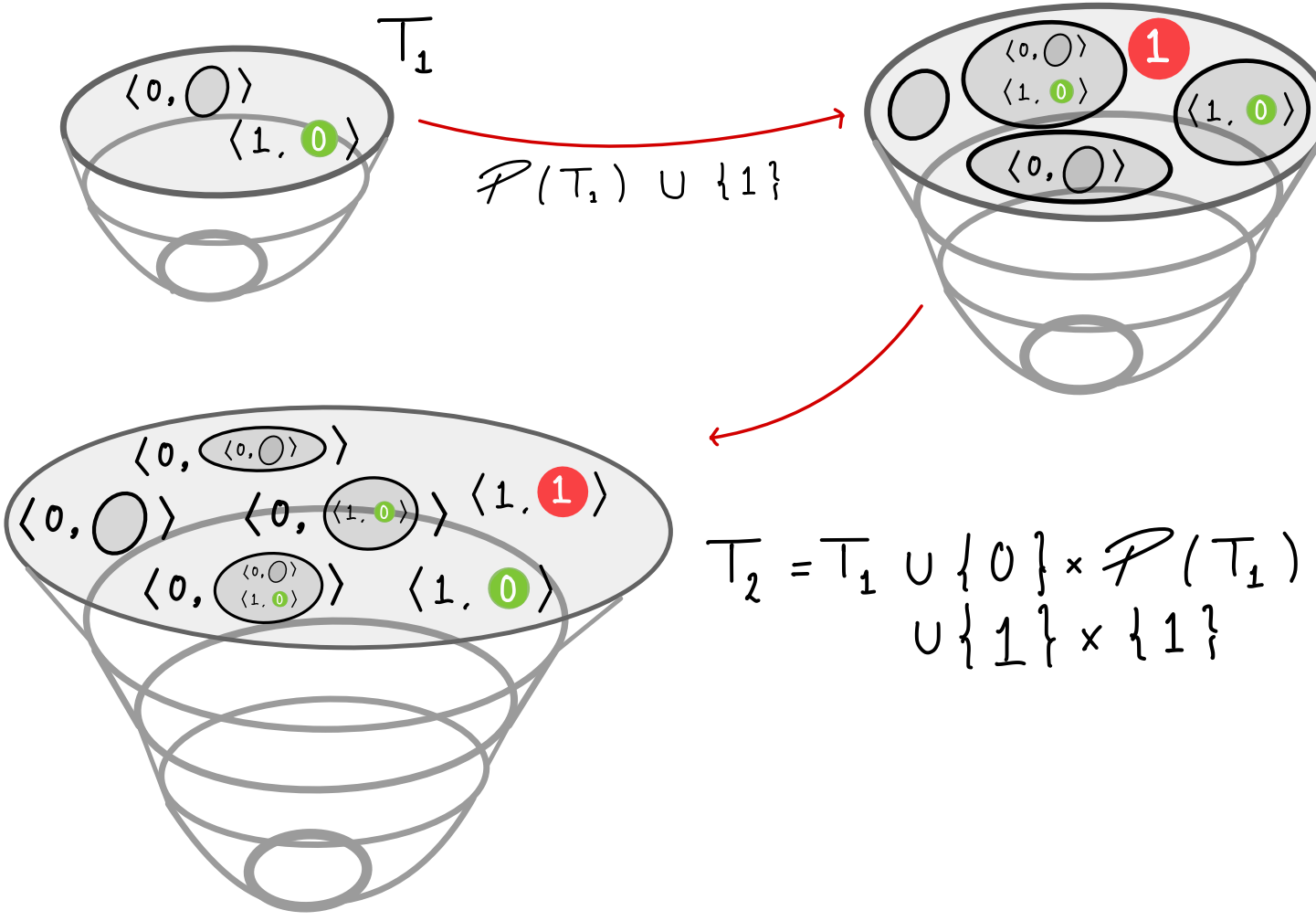
Building a Domain



Building a Domain



Building a Domain



Relativized Quantifiers

- A predicate $\mathbf{T}$ can be defined that asks for a given object x, "is there a tier of the model that x belongs to?" by:

$$\mathbf{T}\, x := \exists[\mathsf{Ord}]\, \alpha \,.\, x \in T_\alpha$$

- Then we define relativized quantifiers that quantify over the tiers of the model.

$$\overline{\forall} x \,.\, \varphi := \forall x \,.\, \mathbf{T}\, x \to \varphi$$
$$\overline{\exists} x \,.\, \varphi := \exists x \,.\, \mathbf{T}\, x \wedge \varphi$$

- If we translate the axioms of ZF$\mathbb{N}$ to formulas only using $\overline{\forall}, \overline{\exists}, \overline{\in}, \overline{\mathcal{P}}, \ldots, \overline{\mathsf{Nat}}, \overline{\mathbf{S}}, \overline{\mathbf{0}}$, we can prove each of these formulas.

Domain Types

- To support our consistency results, we need to be able to build multiple set theories within our framework.
- Set theories of our framework are built in domain types:

$$\mathcal{D} := \mathsf{d}_0 \mid \mathsf{d}_1 \mid \mathsf{d}_2 \mid \cdots$$

- One way of using a GST is by assuming a set of axioms for predicates and operators that act on a certain domain type.
- For example, we can begin by axiomatising ZF in d_0, and build our model of ZF$\mathbb{N}$ in here.

Connecting a Model to a Domain

- We use a technique common in higher-order logic that allows presenting a type of abstract objects as a portion of an existing type.

Connecting a Model to a Domain

- We use a technique common in higher-order logic that allows presenting a type of abstract objects as a portion of an existing type.
- Rather than axiomatising ZF$\mathbb{N}$, it can be defined in d_1 by abstracting the model in d_0.

Chaining Domains and Models

- Models can be built in any GST with the $\mathbf{Set}$ feature.
- Any model built is a model of a GST with the $\mathbf{Set}$ feature.
- It is possible to chain the process of model building and connection to create interesting arenas of GSTs, in which the 'rules' of the mathematical universe are different in each.
- So far, our kit has been used to build models of GSTs with:
  - non-set natural numbers — ZF$\mathbb{N}$
  - non-set ordered pairs — ZFP
  - a non-set exception element $\bullet$ — ZFE
  - modular combinations of these features — ZF$^+$

The Future of GSTs

- We are excited to continue our research on GSTs, and understand better their applications in digital proof assistants.
- User-friendly GST specification and use:
  - no model building
  - GST blindness
- Isabelle — a generic theorem prover
  - implement in Isabelle to mechanize current results
  - write code to automate model building
  - write code to reuse theorems