Generating Custom Set Theories with Non-Set Structured Objects[⋆]

Ciarán Dunne, J. B. Wells, Fairouz Kamareddine

Heriot-Watt University

2021-07-26 T 14:34

Abstract. Set theory has long been viewed as a foundation of mathematics, is pervasive in mathematical culture, and is explicitly used by much written mathematics. Because arrangements of sets can represent a vast multitude of mathematical objects, in most set theories every object is a set. This causes confusion and adds difficulties to formalising mathematics in set theory. We wish to have set theory's features while also having many mathematical objects not be sets. A *generalized set theory* (GST) is a theory that has *pure sets* and may also have non-sets that can have internal structure and *impure sets* that mix sets and non-sets. This paper provides a GST-building framework. We show example GSTs that have sets and also (1) non-set ordered pairs, (2) non-set natural numbers, (3) a non-set exception object that can not be inside another object, and (4) modular combinations of these features. We show how to axiomatize GSTs and how to build models for GSTs in other GSTs.

1 Introduction

1.1 Set Theory as a Foundation of Mathematics

Set theories like Zermelo-Fraenkel (ZF), and closely related set theories like ZFC and Tarski-Grothendieck (TG), play many important roles in mathematics. ZF's axioms allow expressing a vast number of mathematical concepts. For most of the last century most mathematicians have accepted theories like ZF as suitable foundations of mathematics. ZF's axioms have been rigorously evaluated for roughly a century and have no known inconsistencies. Mathematical theories are often assessed against the standard of whether models can be constructed for them in theories like ZF (what Maddy [14] calls *risk assessment*). Much of mathematical notation and reasoning is rooted in set theory. A significant amount of mathematics has been formalised in set theory and computer-verified using proof assistants like Isabelle/ZF [18,9], Mizar [2], and Metamath [15].

[⋆] The final authenticated publication from Springer is available online at `https://doi.org/10.1007/978-3-030-81097-9_19`. This document differs from the Springer publication by the following corrections in figure 4, all related to the **Exception** feature: (1) in $\mathsf{ExOutside}$, a $\forall_i$ was corrected to be a $\forall_i^{\neq\bullet}$ and two $\vee$'s were corrected to be $\wedge$'s; (2) in ZFE_i and ZF_i^+, the set Base_i was decomposed and the Init portion of it was removed from the scope of the substitution replacing $\forall_i$ by $\forall_i^{\neq\bullet}$.

Mathematics varies in the kind and degree of assumptions made of the underlying foundation. Some mathematics explicitly specifies a set-theoretic or type-theoretic foundation and some does not. Set theories like ZF are usually stated in first-order logic (FOL), but are sometimes stated in higher-order logic (HOL) or given as theories embedded in a dependent type system. In some mathematics, functions are sets of ordered pairs while in other mathematics functions are not even sets. There is variation in how undefined terms are treated [6,19]. When viewing ZF as the underlying foundation, it is assumed that high-level mathematics has meaningful translations into ZF, or that ZF can be safely modified to accommodate the user's needs.

1.2 Representation Overlap in Set-Theoretic Formalisation

Translating human-written mathematics into ZF has complications. Every object in ZF's domain of discourse is a pure set, so objects of human-written text must be represented as pure sets. Objects that the mathematician views as distinct can have the same ZF representation. For example, consider formalising a function $g : (\mathbb{N}^2 \cup \mathcal{P}(\mathbb{N})) \to \{0, 1\}$ such that $g(\langle 0, 1\rangle) = 0$ and $g(\{1, 2\}) = 1$. Let $(\,\cdot\,)^*$ be the translation of human-written mathematical objects into the domain of ZF. Typically, $\mathbb{N}$ is represented using the von Neumann ordinals, so $0^* := \emptyset$ and $(k + 1)^* := k^* \cup \{k^*\}$. Also, ordered pairs are usually represented using Kuratowski's encoding where $\langle a, b\rangle^* := \{\{a^*\}, \{a^*, b^*\}\}$. Furthermore, sets of the human-written text usually get the naïve translation $\{x_1, \ldots, x_n\}^* = \{x_1^*, \ldots, x_n^*\}$. Using these representations, the ordered pair $\langle 0, 1\rangle$ and the set $\{1, 2\}$ are represented in ZF by *the same pure set*: $\{\{\emptyset\}, \{\emptyset, \{\emptyset\}\}\}$. Thus, a naïve translation of the definition of g will require that $0^* = g(\langle 0, 1\rangle^*) = g(\{1, 2\}^*) = 1^*$ and there will be no value for g satisfying its specification, i.e., the naïvely translated definition will fail to define anything. A standard set-theoretic solution is to not use $\mathbb{N}^2 \cup \mathcal{P}(\mathbb{N})$ as the domain of g but instead to use $(\{0\} \times \mathbb{N}^2) \cup (\{1\} \times \mathcal{P}(\mathbb{N}))$, i.e., tag every member of $\mathbb{N}^2$ with 0 and every member of $\mathcal{P}(\mathbb{N})$ with 1. This works because $\{0\} \times \mathbb{N}^2$ and $\{1\} \times \mathcal{P}(\mathbb{N})$ are disjoint. This requires complete foresight of the objects to be used, obscures the mathematics under a layer of tagging and untagging, and increases the costs of formalisation.

Furthermore, sometimes the mathematics needs a class (sometimes a proper class) of objects that are distinct from *all* sets, adding complication. And sometimes the objects that must be distinct from all sets can contain sets. The proper set-theoretic solution is to build a hierarchy in ZF to represent both the sets and the non-set objects of the human-written mathematics using a construction similar to the von Neumann cumulative hierarchy. An example of doing this is the set theory ZFP [5] which has proper classes of both sets and non-set ordered pairs. A model of ZFP can be built in ZF using tagged ZF sets to represent both ZFP sets and ZFP non-set ordered pairs.

Proper classes and tagging both involve awkward reasoning. Definitions, lemmas, and proofs quickly become messy. The user must redefine and reprove operations and relations, leaving them with duplicate symbols and concepts (e.g., the power set operation of ZF vs. the analogous operation on the tagged sets within ZF that represent the sets of the human-written mathematics). How to build these models is not obvious to most mathematicians.

1.3 Type Theory as an Alternative

Type theories typically avoid representation overlap by preventing operations that mix types. Operating on multiple types is done via sum types or inductive datatypes, and something equivalent to tagging and untagging happens in a type theory's underlying model theory, but the user is shielded from most details.

Unfortunately, formalising mathematics in type theory is not always the best option. Removing set-theoretic dependencies can transform a text in ways that take it far from the author's conception. As mathematics gets more complex, the typing combinations push the limits of human cognition. Type error messages can be beyond human comprehension. The typing rules of proof assistants can differ in significant ways from the human-readable documentation, and immense expertise in the implementation can be needed. Typing constraints can add proving obligations that are not relevant to the mathematics being formalised. Formalising mathematics in type theories can require awkward and expensive workarounds that sometimes seem infeasible. To address these issues, more sophisticated type systems are developed that can require more expertise to comprehend. Finally, some type-theoretic provers focus on constructive, non-classical reasoning, but much mathematics is non-constructive, and constructive reasoning can be an unnecessary burden.

1.4 An Arena for Custom Set Theories

We seek to retain the useful qualities of set theory whilst being able to have mathematical objects that are genuinely not sets. Some set theories (e.g., ZFA, KPU) have non-set objects called *urelements* which contain no set members (but are not the empty set) and can belong to sets. Typically urelements have no internal structure (an exception is Aczel's GST [1]). To avoid confusion with typical structure-free urelements, we use the phrase *non-set object* for members of a domain that are not sets. A set is *pure* iff all of its members are pure sets; other sets are *impure*. A *generalized set theory* (GST) is a theory that has pure sets and may also have non-sets that can have internal structure and impure sets. ZFP (mentioned above) is a GST with non-sets with internal structure.

If a set theory S can be shown to be consistent relative to ZF and S is a better match for some mathematicians' needs, it is reasonable that they use S instead of ZF as a foundation. So we ask: Is it possible to give each mathematician a foundation that matches their intuition and in which their mathematics is formally true in the original human-written form rather than only becoming formally true after substantial effort and transformation? With this aim in mind, we propose what we call an *arena* in which multiple different GSTs (including ZF) can co-exist and a toolkit to support showing relative consistency results. Our plan and its fulfillment in later sections of this paper is as follows.

We begin in section 2 with a logical framework that supplies features needed for an arena for set theories. Our design is inspired by systems such as the combination of Isabelle/Pure with Isabelle/FOL that underlies Isabelle/ZF, but we have deliberately used a bare minimum of features. To cleanly support multiple GSTs simultaneously, we have a countably infinite set of *domain* types which are base types of individuals. We have function types to support definitions and set theory axiom schemas. In any given derivation, one domain type is designated as the *founder* domain type. We allow ∀-introduction only at the founder domain type. We allow ∀-elimination at the founder domain type and all non-domain types, and forbid ∀-elimination at all non-founder domain types. Reasoning in a derivation about a domain d_k other than the derivation's founder domain d_i is intended to work via a connection from d_k to a model for d_k in another domain; there should be a chain of connections from domains to their models which terminates in the founder domain. We supply axioms for using (eliminating) equality at all types but we only introduce equality at domain types and do so via domain-specific axioms like ZF's Axiom of Extensionality.

Section 3 axiomatizes example GSTs with non-set ordered pairs, non-set natural numbers, a non-set exception element that can not be inside any other object, and the combination of all of these features. This section gives a generalised specification of Zermelo-Fraenkel set theory (GZF) as a feature of GSTs. The GZF specification differs from ZF by (1) not expecting everything to be a set and (2) not specifying well-foundedness because this is handled by our toolkit for combining features to build a GST. The GZF specification is also used as a template where the ∀-quantifier may be replaced by a quantifier restricted to a model constructed within a domain.

Section 4 provides a toolkit for building and reasoning about models of GSTs. The user can build models of GSTs within any GST to verify the consistency of a GST or to explore what models are possible and what axiomatizations might be possible for those models. The main parameter of our model-building machinery is a constant called $\mathsf{Ops}_{i,j}$ which the user axiomatizes to specify the operations used to build in d_i the tiers of a cumulative model intended for use as domain d_j. Models are defined using transfinite recursion, the user-supplied axioms for $\mathsf{Ops}_{i,j}$, and tagging machinery. We show how a user can axiomatize $\mathsf{Ops}_{i,j}$ to yield a model satisfying GZF.

Section 5 defines how to connect to domain d_j a model built in domain d_i intended for domain d_j. Connection is achieved by axiomatizing an isomorphism between the model and the domain in the style of Gordon/HOL type definitions.

Section 6 builds models for the example GSTs given in section 3 and connects these GSTs to their models as part of showing consistency.

1.5 Related Work

Isabelle/ZF [18] is an embedding of first-order logic and ZF in Isabelle/Pure, a simply-typed intuitionistic higher-order logic [17]. Isabelle/ZF's base library primarily proves theorems about set theory (functions, ordinals, recursion). IsarMathLib [9] is a library of mathematics in areas such as abstract algebra,

$$\begin{array}{rll} \delta \mathrel{\dot\epsilon} \mathsf{Domain} & ::= \mathsf{d}_1 \mid \mathsf{d}_2 \mid \mathsf{d}_3 \mid \cdots & a,b,p,q,x,y,z \mathrel{\dot\epsilon} \mathsf{Var} \ ::= \mathsf{v}_1 \mid \mathsf{v}_2 \mid \mathsf{v}_3 \mid \cdots \\ \sigma,\tau \mathrel{\dot\epsilon} \mathsf{Type} & ::= \star \mid \delta \mid \sigma \Rightarrow \tau & c \mathrel{\dot\epsilon} \mathsf{Const} \ ::= \rightarrow \mid \forall_\tau \mid \cdots \\ i,j \mathrel{\dot\epsilon} \mathbb{N} & & \nu \mathrel{\dot\epsilon} \mathsf{Var} \cup \mathsf{Const} \end{array}$$

$$A,\ldots,Z \mathrel{\dot\epsilon} \mathsf{Term} ::= x \mid c \mid B\,C \mid \lambda\, x : \tau \,.\, B \quad (\text{if } \mathsf{vtyp}(x) \equiv \tau)$$

$$\frac{}{x :: \mathsf{vtyp}(x)} \qquad \frac{}{c :: \mathsf{ctyp}(c)} \qquad \frac{B :: \sigma \quad \mathsf{vtyp}(x) \equiv \tau}{(\lambda\, x : \tau \,.\, B) :: \tau \Rightarrow \sigma} \qquad \frac{B :: \tau \Rightarrow \sigma \quad C :: \tau}{(B\,C) :: \sigma}$$

Fig. 1. Syntax and typing rules

analysis, and topology that is formalised in Isabelle/ZF. Mizar [2] provides a language for proving theorems in TG. A notable feature of Mizar is "weak typing" which gives some of the advantages of types. Metamath/ZFC [15] develops ZFC in a minimal framework without much proof automation.

Many have sought a middle ground between set theory and type theory. Krauss [10] worked on adding "soft types" to Isabelle/ZF, and this proposal was later developed into Isabelle/Set [11], an axiomatisation of TG. Brown [4] developed extended first-order logic (EFOL), which extends FOL with some higher-order convenience. The Egal prover [3] axiomatizes TG within EFOL. HOLZF [16] axiomatizes in Isabelle/HOL a type `ZF` of the pure sets of ZF and supports conversion between ZF sets and HOL sets of ZF sets.

Aczel and Lunnon [1] worked on GSTs (and coined the phrase "GST"). It appears that their systems assume the Anti-Foundation axiom instead of ZF's Axiom of Foundation. They discuss model building but identify no axioms.

Kunčar and Popescu [8,12,13] developed and proved soundness of methods for connecting an entire abstract type τ to a subset of a concrete representation type τ' given by a predicate on τ'; our approach in section 5 has a very similar essential core. Under the slogan "little theories", Farmer et al. [7] developed in the IMPS prover flexible meta-level methods for automatically generating and using theory interpretations for connecting abstract theories to concrete theories; here the emphasis is more on using the abstract theories to prove things in the concrete theories and less on using a trusted believed-to-be-consistent concrete theory to prove consistency of the abstract theory.

2 Logical Framework

2.1 Syntax

Figure 1 defines the meta-level sets Domain, Type, Var, and Const. Each type d_i is a *domain* (of FOL individuals). The *function type* constructor $\Rightarrow$ is right associative, i.e., $(\tau_1 \Rightarrow \tau_2 \Rightarrow \tau_3) \equiv (\tau_1 \Rightarrow (\tau_2 \Rightarrow \tau_3))$. The fixed *variable type* function vtyp maps every $x \in \mathsf{Var}$ to some $\sigma \in \mathsf{Type}$. For each $\tau \in \mathsf{Type}$ there are infinitely many variables $y \in \mathsf{Var}$ such that $\mathsf{vtyp}(y) \equiv \tau$. The fixed *constant type* function ctyp maps every member of Const to some $\sigma \in \mathsf{Type}$ and it holds that

(HYP) $\{\varphi\} \vdash_i \varphi$

(IMPI) If $\mathit{\Gamma} \vdash_i \psi$, then $\mathit{\Gamma} - \varphi \vdash_i \varphi \to \psi$

(IMPE) If $\mathit{\Gamma} \vdash_i \varphi \to \psi$ and $\mathit{\Gamma}' \vdash_i \varphi$, then $\mathit{\Gamma} \cup \mathit{\Gamma}' \vdash_i \psi$

(ALLI$_i$) If $\mathit{\Gamma} \vdash_i \varphi$, $x :: \mathsf{d}_i$, and $x \notin \mathsf{FV}[\mathit{\Gamma}]$, then $\mathit{\Gamma} \vdash_i \forall_i\,(\lambda\, x : \mathsf{d}_i \,.\, \varphi)$

(ALLE$_i$) If $\mathit{\Gamma} \vdash_i \forall_\tau P$, and $B :: \tau$, and $\forall\, j \neq i \,.\, \tau \neq \mathsf{d}_j$, then $\mathit{\Gamma} \vdash_i P\,B$

$$\begin{aligned}\mathsf{Init} := &\left\{\begin{matrix}\forall\, p \,.\, \forall_\tau\, x, y \,.\, x = y \to (p\,x \leftrightarrow p\,y), \\ (\neq_\tau) = (\lambda\, x, y \,.\, \neg\,(x = y))\end{matrix} \;\middle|\; \tau \,\epsilon\, \mathsf{Type}\right\} \\ &\cup \{\forall_\star\, p, q \,.\, (\neg\, p \to \neg\, q) \to q \to p, \quad \forall_\star\, p, q \,.\, (p \leftrightarrow q) \to p \to q, \\ &\quad \forall_\star\, p, q \,.\, (p \leftrightarrow q) \to q \to p, \quad \forall_\star\, p, q \,.\, (p \to q) \to (q \to p) \to (p \leftrightarrow q), \\ &\quad \wedge = (\lambda\, p, q \,.\, \neg\,(p \to \neg\, q)),\ \vee = (\lambda\, p, q \,.\, \neg\, p \to q)\}\end{aligned}$$

$$\begin{aligned}\mathsf{FOLQuants}_i := \{\, &\exists_i = (\lambda\, p \,.\, \neg\,(\forall_i\,(\lambda\, x \,.\, \neg\,(p\,x)))),\ \ \forall_i[\,\cdot\,] = (\lambda\, p, q \,.\, \forall_i\, x \,.\, p\,x \to q\,x), \\ &\exists_i^{\leq 1} = (\lambda\, p \,.\, \forall_i\, y, z \,.\, p\,y \wedge p\,z \to y = z), \exists_i[\,\cdot\,] = (\lambda\, p, q \,.\, \exists_i\, x \,.\, p\,x \wedge q\,x)\}\end{aligned}$$

Fig. 2. Inference rules, initial theory, and simple definitions for quantifiers

$\mathsf{ctyp}(\to) \equiv \star \Rightarrow \star \Rightarrow \star$ and for every $\tau \,\epsilon\, \mathsf{Type}$ that $\mathsf{ctyp}(\forall_\tau) \equiv (\tau \Rightarrow \star) \Rightarrow \star$ and $\mathsf{ctyp}(=_\tau) \equiv \mathsf{ctyp}(\neq_\tau) \equiv \tau \Rightarrow \tau \Rightarrow \star$. For any $i \,\epsilon\, \mathbb{N}$, we abbreviate $\forall_{\mathsf{d}_i}$ as $\forall_i$. Fixed meta-level names for the other *constants* in Const and further details of ctyp will be revealed incrementally. Subscripts i and i, j on the meta-level names of constants are used to indicate a constant is relevant to domain d_i or both domains d_i and d_j; these subscripts are often light grey to help the reader not be distracted by them. Notation of the form $\mathcal{C} :\equiv (\xi_1 :: \tau_1, \ldots, \xi_n :: \tau_n)$ asserts for each $i \,\epsilon\, \{1, \ldots, n\}$ that $\xi_i \,\epsilon\, \mathsf{Const}$ (so the meta-metavariable ξ_i could have been written c_i) and $\mathsf{ctyp}(\xi_i) \equiv \tau_i$ and $\mathcal{C} \equiv \{\xi_1, \ldots, \xi_n\}$.

The rules in figure 1 define the meta-level set Term. As is standard for a λ-calculus, each *abstraction* $\lambda\, x : \sigma \,.\, C$ *binds* the variable x and this is the only way variables can be bound. We identify terms modulo α-equivalence. We then define the *free variable* function FV so that $\mathsf{FV}(B)$ is the set of variables free in the β-normal-form of B. We then further identify terms modulo β-equivalence and lift FV accordingly. Substitution $B[\nu{:=}C]$ is defined as usual. Constants can not be bound by λ.

Figure 1 defines the typing relation :: between Term and Type. Inside a term expression $B :: \tau$ we allow omitting the type σ that is part of the name of an occurrence of $\forall_\sigma$, $=_\sigma$, or $\neq_\sigma$, or that is part of an abstraction $\lambda x : \sigma \,.\, C$, provided that σ can be uniquely determined by the other type information in or about B including what is known about the types of constants.

We say that a term B is a *formula* iff $B :: \star$. Let φ, ψ, γ range over formulas, and let $\mathit{\Phi}, \mathit{\Psi}, \mathit{\Gamma}$ range over sets of formulas. Let $\mathit{\Gamma} + \varphi$ denote $\mathit{\Gamma} \cup \{\varphi\}$ and let $\mathit{\Gamma} - \varphi$ denote $\mathit{\Gamma} \setminus \{\varphi\}$. Let $\mathsf{FV}[\mathit{\Gamma}]$ be the union of all $\mathsf{FV}(\varphi)$ for each $\varphi \,\epsilon\, \mathit{\Gamma}$. Let $\mathit{\Gamma}[\nu := B]$ be the set of all $\varphi[\nu := B]$ for each $\varphi \,\epsilon\, \mathit{\Gamma}$.

2.2 Propositional and First-Order Logic

A *sequent* is a syntactic object $\Gamma \vdash_i \varphi$ with *founder domain* d_i. Figure 2 give inference rules that define the entailment relation $\vdash_i$. We write $\Gamma \vdash_i \Psi$ iff $\Gamma \vdash_i \varphi$ for every $\varphi \in \Psi$. Note that $\vdash_i$ can only do $\forall$-introduction for $\forall_i$ (which abbreviates $\forall_{\mathsf{d}_i}$) and cannot do $\forall$-elimination for $\forall_j$ where $i \neq j$. We will later supply simple definitions for $\forall_j$ where $i \neq j$ that make these rules admissible:

$$(\textsc{allI}_{i,j})\ \frac{\Gamma \vdash_i \varphi \quad x :: \mathsf{d}_j \quad x \notin \mathsf{FV}[\Gamma]}{\Gamma \vdash_i \forall_j\,(\lambda\, x : \mathsf{d}_j \,.\, \varphi)} \qquad (\textsc{allE}_{i,j})\ \frac{\Gamma \vdash_i \forall_j\, P \quad B :: \mathsf{d}_j}{\Gamma \vdash_i P\, B}$$

We write $\Gamma \vdash_i (\textsc{allI}_{i,j}), (\textsc{allE}_{i,j})$ iff both $(\textsc{allI}_{i,j})$ and $(\textsc{allE}_{i,j})$ are admissible using Γ. The rule $(\textsc{allE}_i)$ allows us to eliminate universal quantifications at d_i and any non-domain type, which supports simple definitions.

Figure 2 defines the *initial theory* Init that defines the other logic operators ($\neg$, $\leftrightarrow$, $\wedge$, $\vee$), and proves their usual introduction and elimination rules, establishes classical logic, and implements equality. A *simple definition* is a formula of the form $c =_\tau B$. The first axiom in Init allows *eliminating* equalities at all types, but we only *introduce* equalities via domain-specific axioms at domain types. The constants $=_\tau$, $\wedge$, $\vee$, $\leftrightarrow$, and $\rightarrow$ are all binary infix operators, listed in descending order of precedence. If c is infix, an application $(c\,X)\,Y$ may be written $X\ c\ Y$. If $B_1, \ldots, B_n, C$ are terms and $\sim$ is a binary infix operator, then we may write $B_1, \ldots, B_n \sim C$ for $B_1 \sim C \wedge \cdots \wedge B_n \sim C$. Negation ($\neg$) and function application take precedence over infix operators, e.g., $F\,x =_\tau G\,x$ is $(F\,x) =_\tau (G\,x)$.

If Q is a constant for a quantifier, then $Q\,(\lambda\, x : \tau \,.\, \varphi)$ may be written $Q\, x \,.\, \varphi$. The notation $Q\, x_1, \ldots, x_n \,.\, \varphi$ abbreviates the nested applications of quantifiers and abstractions $Q\,(\lambda x_1 : \tau \,.\, \cdots\, Q\,(\lambda x_n : \tau \,.\, \varphi))$. Quantification has lower precedence than all other logical constants. Thus, $\forall_0\, x \,.\, \varphi \rightarrow \psi$ is $\forall_0\, x \,.\, (\varphi \rightarrow \psi)$.

From each constant $\forall_i$ that represents a universal quantifier at type d_i, the set $\mathsf{FOLQuants}_i$ of simple definitions in figure 2 defines *existential* ($\exists$), *at-most-one* ($\exists^{\leq 1}$), and *bounded* (also called *restricted*) quantification ($\forall[\cdot], \exists[\cdot]$). Formulas of the form $(\forall[\cdot]\,P)\,Q$ and $(\exists[\cdot]\,P)\,Q$ may be written as $\forall[P]\,x \,.\, Q\,x$ and $\exists[P]\,x \,.\, Q\,x$ respectively. If $\sim$ is a binary infix operator, we may write $\forall\, x \sim B \,.\, \varphi$ and $\exists\, x \sim B \,.\, \varphi$ for $\forall[\lambda\, y \,.\, y \sim B]\, x \,.\, \varphi$ and $\exists[\lambda\, y \,.\, y \sim B]\, x \,.\, \varphi$ respectively, where y is fresh. If $\Gamma \vdash_i (\textsc{allI}_{i,j}), (\textsc{allE}_{i,j})$ and $\Gamma \vdash_i \mathsf{Init} \cup \mathsf{FOLQuants}_j$, then each quantifier satisfies its usual introduction and elimination rules on d_j.

3 Example Axiomatizations of Generalized Set Theories

This section axiomatizes five example GSTs. We define four example modular *features* that each characterise a kind of mathematical object. So the reader does not mix them up, we index features by odd numbers and later in section 6 we index example domains by even numbers. Feature k in domain d_i is given by (1) a signature of constants sig_i^k, (2) a set of formulas theory_i^k that characterizes the constants in sig_i^k, (3) an unary predicate iden_i^k that identifies objects added by the feature, and (4) a binary predicate child_i^k that declares *internal* structure.

$$
\begin{aligned}
\mathsf{GZFConsts}_i &:\equiv (\in_i :: \mathsf{d}_i \Rightarrow \mathsf{d}_i \Rightarrow \star,\ \emptyset_i :: \mathsf{d}_i,\ \mathsf{Set}_i :: \mathsf{d}_i \Rightarrow \star,\ \textstyle\bigcup_i :: \mathsf{d}_i \Rightarrow \mathsf{d}_i,\\
&\qquad \subseteq_i :: \mathsf{d}_i \Rightarrow \mathsf{d}_i \Rightarrow \star,\ \mathcal{P}_i :: \mathsf{d}_i \Rightarrow \mathsf{d}_i,\ \mathsf{succ}_i :: \mathsf{d}_i \Rightarrow \mathsf{d}_i,\\
&\qquad \mathsf{Inf}_i :: \mathsf{d}_i,\ \mathcal{R}_i :: (\mathsf{d}_i \Rightarrow \mathsf{d}_i \Rightarrow \star) \Rightarrow \mathsf{d}_i \Rightarrow \mathsf{d}_i)\\
\mathsf{GZF}_i &:= \{\, (\text{EMP}_i)\ \forall_i\, a \,.\, a \notin_i \emptyset_i,\ \ (\text{SET}_i)\ \forall_i\, x \,.\, (\mathsf{Set}_i\ x) \leftrightarrow (x = \emptyset_i \vee \exists_i\, y \,.\, y \in_i x),\\
&\qquad (\text{SUB}_i)\ \ \subseteq_i\ = (\lambda\, x, y \,.\, \mathsf{Set}_i\ x \wedge \mathsf{Set}_i\ y \wedge \forall_i\ a \in_i x \,.\, a \in_i y),\\
&\qquad (\text{EXT}_i)\ \ \forall_i[\mathsf{Set}_i]\ x, y \,.\, (\forall_i\ a \,.\, a \in_i x \leftrightarrow a \in_i y) \to x = y,\\
&\qquad (\text{UNI}_i)\ \ \forall_i[\mathsf{Set}_i]\ x \,.\, \mathsf{Set}_i\ (\textstyle\bigcup_i\ x)\ \wedge\ \forall_i\ a \,.\, a \in_i (\textstyle\bigcup_i\ x) \leftrightarrow (\exists_i\ z \in_i x \,.\, a \in_i z),\\
&\qquad (\text{POW}_i)\ \ \forall_i[\mathsf{Set}_i]\ x \,.\, \forall_i\ z \,.\, z \in_i (\mathcal{P}_i\ x) \leftrightarrow z \subseteq_i x,\\
&\qquad (\text{SUC}_i)\ \ \forall_i[\mathsf{Set}_i]\ x \,.\, \forall_i\ a \,.\, a \in_i (\mathsf{succ}_i\ x) \leftrightarrow (a \in_i x \vee a = x),\\
&\qquad (\text{INF}_i)\ \ \emptyset_i \in_i \mathsf{Inf}_i \wedge \forall_i\ x \in_i \mathsf{Inf}_i \,.\, (\mathsf{succ}_i\ x) \in_i \mathsf{Inf}_i,\\
&\qquad (\text{RPL}_i)\ \ \forall_{\mathsf{d}_i \Rightarrow \mathsf{d}_i \Rightarrow \star}\ p \,.\, \forall_i[\mathsf{Set}_i]\ x \,.\, (\forall_i\ a \in_i x \,.\, \exists_i^{\leq 1}\ b \,.\, p\, a\, b)\\
&\qquad\qquad \to (\mathsf{Set}_i\ (\mathcal{R}_i\ p\, x) \wedge \forall_i\ b \,.\, b \in_i (\mathcal{R}_i\ p\, x) \leftrightarrow \exists_i\ a \in_i x \,.\, p\, a\, b) \,\}\\
\mathsf{PConsts}_i &:\equiv (\mathsf{pair}_i :: \mathsf{d}_i \Rightarrow \mathsf{d}_i \Rightarrow \mathsf{d}_i, \mathsf{Pair}_i :: \mathsf{d}_i \Rightarrow \star) \qquad (X, Y)_i := \mathsf{pair}_i\ X\, Y\\
\mathsf{PTheory}_i &:= \{\, \forall_i\ a, b, x, y \,.\, (a, b)_i = (x, y)_i \leftrightarrow (a = x \wedge b = y),\\
&\qquad \forall_i\ p \,.\, \mathsf{Pair}_i\ p \leftrightarrow \exists_i\ x, y \,.\, p = (x, y)_i \,\}\\
\mathsf{NConsts}_i &:\equiv (\mathbf{0}_i :: \mathsf{d}_i,\ \mathbf{S}_i :: \mathsf{d}_i \Rightarrow \mathsf{d}_i,\ \mathsf{Nat}_i :: \mathsf{d}_i \Rightarrow \star)\\
\mathsf{NTheory}_i &:= \{\, \mathsf{Nat}_i\ \mathbf{0}_i, \quad \mathbf{0}_i = \mathbf{0}_i, \quad \forall_i[\mathsf{Nat}_i]\ x \,.\, \mathsf{Nat}_i\ (\mathbf{S}_i\ x),\\
&\qquad \forall_i[\mathsf{Nat}_i]\ x, y \,.\, x = y \leftrightarrow \mathbf{S}_i\ x = \mathbf{S}_i\ y,\\
&\qquad \forall_i[\mathsf{Nat}_i]\ x \,.\, \mathbf{S}_i\ x \neq \mathbf{0}_i,\\
&\qquad \forall_{\mathsf{d}_i \Rightarrow \star}\ p \,.\, p\, \mathbf{0}_i \to (\forall_i[\mathsf{Nat}_i]\ x \,.\, p\, x \to p\, (\mathbf{S}_i\ x)) \to \forall_i[\mathsf{Nat}_i]\ y \,.\, p\, y \,\}\\
\mathsf{EConsts}_i &:\equiv (\bullet_i :: \mathsf{d}_i,\ \imath_i :: (\mathsf{d}_i \Rightarrow \star) \Rightarrow \mathsf{d}_i)\\
\mathsf{ETheory}_i &:= \{\, \exists!_i = (\lambda\, p \,.\, \exists_i\ x \,.\, p\, x \wedge \exists_i^{\leq 1}\ x \,.\, p\, x),\\
&\qquad \forall_{\mathsf{d}_i \Rightarrow \star}\ p \,.\, (\exists!_i\ x \,.\, p\, x) \to (\forall_i\ y \,.\, p\, y \leftrightarrow y = (\imath_i\ z \,.\, p\, z)),\\
&\qquad \forall_{\mathsf{d}_i \Rightarrow \star}\ p \,.\, \neg\, (\exists!_i\ x \,.\, p\, x) \to (\imath_i\ z \,.\, p\, z) = \bullet_i \,\}
\end{aligned}
$$

Fig. 3. Signatures and theories for the **Set**, **Pair**, **Nat**, and **Exception** features

The **Set** feature provides sets. Figure 3 defines constants $\mathsf{GZFConsts}_i$ and formulas GZF_i. The feature's theory, signature, identification predicate, and structure predicate are given by $\mathsf{sig}_i^1 \equiv \mathsf{GZFConsts}_i$, and $\mathsf{theory}_i^1 \equiv \mathsf{GZF}_i$, and $\mathsf{iden}_i^1 \equiv \mathsf{Set}_i$, and $\mathsf{child}_i^1 \equiv \in_i$. The axioms in GZF_i allow non-sets. The Foundation axiom is missing from GZF_i and will be supplied when features are combined.

The **Pair** feature adds non-set ordered pairs. Figure 3 defines constants $\mathsf{PConsts}_i$ and formulas $\mathsf{PTheory}_i$. We define $\mathsf{sig}_i^3 \equiv \mathsf{PConsts}_i$, and $\mathsf{theory}_i^3 \equiv \mathsf{PTheory}_i$, and $\mathsf{iden}_i^3 \equiv \mathsf{Pair}_i$, and $\mathsf{child}_i^3 \equiv (\lambda\, x, p \,.\, \exists_i\ y \,.\, p =_{\mathsf{d}_i} (x, y)_i \vee p =_{\mathsf{d}_i} (y, x)_i)$. The axioms include the standard *characteristic property of ordered pairs*.

The **Nat** feature adds non-set natural numbers obeying Peano Arithmetic. Figure 3 defines constants $\mathsf{NConsts}_i$ and formulas $\mathsf{NTheory}_i$. We define $\mathsf{sig}_i^5 \equiv \mathsf{NConsts}_i$, and $\mathsf{theory}_i^5 \equiv \mathsf{NTheory}_i$, and $\mathsf{iden}_i^5 \equiv \mathsf{Nat}$, and leave child_i^5 undefined.

The **Exception** feature adds a non-set exception element $\bullet_i$ and a definite description operator $\imath_i$ that uses $\bullet_i$ as its default. Figure 3 defines constants $\mathsf{EConsts}_i$ and formulas $\mathsf{ETheory}_i$. We define $\mathsf{sig}_i^7 \equiv \mathsf{EConsts}_i$, and $\mathsf{theory}_i^7 \equiv \mathsf{ETheory}_i$, and $\mathsf{iden}_i^7 \equiv (\lambda\, x \,.\, x =_{\mathsf{d}_i} \bullet_i)$, and we leave child_i^7 undefined. The only object this feature adds is $\bullet_i$, which has no internal structure.

To combine features to make a GST, figure 4 defines formulas that state that a combination of features is well behaved. The formula $\mathsf{Iden}(k_1, \ldots, k_n)$ states

$$\begin{aligned}
\mathsf{Iden}_i(k_1,\ldots,k_n) &:= \forall_i\, x\,.\,\mathsf{iden}_i^{k_1}\, x \vee \cdots \vee \mathsf{iden}_i^{k_n}\, x \\
\mathsf{distinct}_i(k,l) &:= \forall_i\, x\,.\,\neg\,\mathsf{iden}_i^{k}\, x \vee \neg\,\mathsf{iden}_i^{l}\, x \\
\mathsf{AllDistinct}_i(k_1,\ldots,k_n) &:= \mathsf{distinct}_i(k_1,k_2) \wedge \cdots \wedge \mathsf{distinct}_i(k_1,k_n) \\
&\quad\ \wedge \cdots \wedge \mathsf{distinct}_i(k_{n-1},k_n) \\
\mathsf{WF}_i(k_1,\ldots,k_n) &:= \forall_{\mathsf{d}_i \Rightarrow \star}\, p\,.\,(\forall_i\, x\,.\,\neg\, p\, x) \\
&\quad\ \vee (\exists_i [p]\, a\,.\,\neg\,\exists_i [p]\, b\,.\,\mathsf{child}_i^{k_1}\, b\, a \wedge \cdots \wedge \mathsf{child}_i^{k_n}\, b\, a) \\
\mathsf{ExOutside}_i(k_1,\ldots,k_n) &:= \forall^{\neq\bullet}{}_i\, x\,.\,\neg\,\mathsf{child}_i^{k_1}\, \bullet_i\, x \wedge \cdots \wedge \neg\,\mathsf{child}_i^{k_n}\, \bullet_i\, x
\end{aligned}$$

$$\begin{aligned}
\mathsf{Base}_i &:= \mathsf{Init} \cup \mathsf{FOLQuants}_i \cup \mathsf{GZF}_i \\
\mathsf{ZF}_i &:= \mathsf{Base}_i + \mathsf{Iden}_i(1) + \mathsf{WF}_i(1) \\
\mathsf{ZFP}_i &:= \mathsf{Base}_i \cup \mathsf{PTheory}_i + \mathsf{Iden}_i(1,3) + \mathsf{AllDistinct}_i(1,3) + \mathsf{WF}_i(1,3) \\
\mathsf{ZF}\mathbb{N}_i &:= \mathsf{Base}_i \cup \mathsf{NTheory}_i + \mathsf{Iden}_i(1,5) + \mathsf{AllDistinct}_i(1,5) + \mathsf{WF}_i(1) \\
\mathsf{ZFE}_i &:= \{\forall_i^{\neq\bullet} =_{(\mathsf{d}_i \Rightarrow \star)\Rightarrow\star} (\lambda\, p\,.\,\forall_i\, x\,.\, x \neq_{\mathsf{d}_i} \bullet_i \to p\, x)\} \\
&\quad \cup\ \mathsf{Init} \cup (\mathsf{FOLQuants}_i \cup \mathsf{GZF}_i \cup \mathsf{ETheory}_i)[\forall_i := \forall_i^{\neq\bullet}] \\
&\quad +\ \mathsf{Iden}_i(1,7) + \mathsf{AllDistinct}_i(1,7) + \mathsf{WF}_i(1) + \mathsf{ExOutside}_i(1) \\
\mathsf{ZF}_i^{+} &:= \{\forall_i^{\neq\bullet} =_{(\mathsf{d}_i \Rightarrow \star)\Rightarrow\star} (\lambda\, p\,.\,\forall_i\, x\,.\, x \neq_{\mathsf{d}_i} \bullet_i \to p\, x)\} \cup \mathsf{Init} \\
&\quad \cup\ (\mathsf{FOLQuants}_i \cup \mathsf{GZF}_i \cup \mathsf{PTheory}_i \cup \mathsf{NTheory}_i \cup \mathsf{ETheory}_i)[\forall_i := \forall_i^{\neq\bullet}] \\
&\quad +\ \mathsf{Iden}(1,3,5,7) + \mathsf{AllDistinct}_i(1,3,5,7) + \mathsf{WF}_i(1,5) + \mathsf{ExOutside}_i(1,5)
\end{aligned}$$

Fig. 4. Operations for combining features, and axiomatisations of various GSTs

that every object in d_i belongs to at least one of the features k_1, …, k_n, while the formula $\mathsf{AllDistinct}_i(k_1,\ldots,k_n)$ states that every such object belongs to exactly one such feature. The formula $\mathsf{WF}_i(k_1,\ldots,k_n)$ asserts the well-foundedness of the union of the internal structure relations given by $\mathsf{child}_i^{k_1}$, …, $\mathsf{child}_i^{k_n}$. The formula $\mathsf{ExOutside}_i(k_1,\ldots,k_n)$ states that the exception element $\bullet_i$ is not a direct immediate child of any objects belonging to the features k_1, …, k_n.

We define **ZF** in domain d_i via the axioms ZF_i in figure 4 as a GST that uses just the **Set** feature. Let PureZF_i be a traditional formulation of ZF obtained by replacing all bounded $\forall_i[\mathsf{Set}_i]$ quantifiers in GZF_i with unbounded $\forall_i$ quantifiers and adding the Axiom of Foundation. Because $\mathsf{Iden}_i(1)$ allows us to prove $\forall_i\, x\,.\,\mathsf{Set}_i\, x$, it follows that $\mathsf{ZF}_i \vdash_i \mathsf{PureZF}_i$ and also that $\mathsf{PureZF}_i \vdash_i \mathsf{ZF}_i$.

We define **ZFP** in d_i via the axioms ZFP_i as a GST with non-set ordered pairs that combines the **Set** and **Pair** features. Note that the non-set ordered pairs of **ZFP** do not have any extraneous properties.

We define **ZF**$\mathbb{N}$ in d_i via the axioms $\mathsf{ZF}\mathbb{N}_i$ as a GST with non-set natural numbers that combines the **Set** and **Nat** features. Because $\mathsf{NTheory}_i$ only provides a predicate symbol Nat_i, the user of $\mathsf{ZF}\mathbb{N}_i$ will want a set $\mathbb{N}$ containing exactly all the objects that satisfy Nat_i (i.e., the non-set natural numbers), and this can be done via the axiom (RPL$_i$) and the von Neumann natural numbers.

We define **ZFE** in d_i via the axioms ZFE_i as a GST with a non-set exception element that is excluded from the domain of quantifiers and is not contained in any set. It is intended that a ZFE user does not directly use the (ALLI) and (ALLE) rules, but instead uses a different quantifier $\forall_i^{\neq\bullet}$ (and other quantifiers derived from it) that excludes the exception element. Note that all occurrences of $\forall_i$ are replaced by $\forall_i^{\neq\bullet}$ in the formulas GZF_i and $\mathsf{FOLQuants}_i$.

We define **ZF$^+$** in d_i via the axioms ZF_i^+ as a GST that combines all four example features. Note that this uses the same $\forall_i^{\neq\bullet}$ quantifier as ZFE.

Remember the example specification from section 1 of a function $g : (\mathbb{N}^2 \cup \mathcal{P}(\mathbb{N})) \to \{0,1\}$ such that $g(\langle 0,1\rangle) = 0$ and $g(\{1,2\}) = 1$. How can g be handled in our five example GSTs? Assume we use non-set natural numbers if we have the **Nat** feature (ZF$\mathbb{N}$, ZF$^+$) and otherwise we use the von Neumann naturals, and similarly we use non-set ordered pairs if we have the **Pair** feature (ZFP, ZF$^+$) and otherwise we use Kuratowski pairs. Represent g as the least set such that $\langle x, y\rangle \in g$ whenever input x should map to output y. In ZF, g is not a function because $\langle 0,1\rangle = \{1,2\}$ and the set-function application binary infix operator ${}^{\backprime}{}_i$ can not make both $g\,{}^{\backprime}{}_i\,\langle 0,1\rangle = 0$ and $g\,{}^{\backprime}{}_i\,\{1,2\} = 1$ true. Also, depending on how we "define" the "function" g, we might prove incorrect results or even make our entire system inconsistent. In ZFP, ZF$\mathbb{N}$, and ZF$^+$ it holds that $\langle 0,1\rangle \neq \{1,2\}$, so g is a function and we are happy. In ZFE, g is not a function but the **Exception** feature makes some failure-handling options a bit easier. One option uses the definite description operator ι_i in defining the set-function application operator ${}^{\backprime}{}_i$ to be $(\lambda\, x, y \,.\, \iota_i\, z \,.\, \langle y, z\rangle \in_i x)$, which makes $g\,{}^{\backprime}{}_i\, x = \bullet_i$ if g is not functional at x. Another option is taking a predicate gSpec specifying a function with the desired input/output behavior for g and then defining g as $(\iota_i\, z \,.\, \mathsf{gSpec}\, z)$, which would evaluate to $\bullet_i$. The exception object $\bullet_i$ is useful in these cases because it can not accidentally get embedded inside larger results and can not equal a value tested by the $\forall_i^{\neq\bullet}$ quantifier.

4 Model Building Kit

This section defines tools for building within GZF-domains models of GSTs with the **Set** feature that can be specified to support additional features.

4.1 Set Theory Tools

We define three variants of *set comprehension* notation. If $a, b \notin \mathsf{FV}(P) \cup \mathsf{FV}(X)$, we write $\{\, b \mid \exists_i a {\in_i} X \,.\, P\, a\, b \,\}$ for $\mathcal{R}_i\, P\, X$, and $\{\, a {\in_i} X \mid P\, a \,\}$ for $\mathcal{R}_i\, (\lambda a, b \,.\, a =_{\mathsf{d}_i} b \wedge a \in_i X \wedge P\, a)\, X$. If $F :: \mathsf{d}_i \Rightarrow \mathsf{d}_i$ and $x, y \notin \mathsf{FV}(B) \cup \mathsf{FV}(F)$, we write $\{\, F\, x \mid x \in_i B,\, P\, x \,\}$ for $\{\, y \mid \exists_i\, x \in_i B \,.\, P\, x \wedge y =_{\mathsf{d}_i} F\, x \,\}$.

Figure 5 defines the set $\mathsf{ZFUtils}_i$ of simple definitions for operators including those related to ordered pairs, ordinals, and tagging. The operators π_i^1 and π_i^2, called the *left* and *right projections* (resp.), are defined such that if X and Y are sets, then $\langle X, Y\rangle_i =_{\mathsf{d}_i} \langle \pi_i^1\, \langle X, Y\rangle_i, \pi_i^2\, \langle X, Y\rangle_i \rangle_i$. A set X is *transitive* iff every set member of X is also a subset of X. A set X is an *ordinal* iff it is a transitive set whose set members are all transitive sets. We say that X is a *limit ordinal* iff $\mathsf{Limit}_i\, X$. The constant ω_i is defined as the intersection of all subsets of Inf_i that are limit ordinals. Thus, ω_i is the smallest limit ordinal.

$$
\begin{array}{l}
\{X,Y\}_i := \mathsf{upair}_i\, X\, Y, \qquad \{X\}_i := \{X,X\}_i, \qquad \langle X,Y\rangle_i := \mathsf{kpair}_i\, X\, Y, \\
0_i := \emptyset_i, \quad 1_i := \mathsf{succ}_i\, 0_i, \quad 2_i := \mathsf{succ}_i\, 1_i \quad 3_i := \mathsf{succ}_i\, 2_i, \quad \ldots \\
\mathsf{ZFUtils}_i := \{ \bigcap_i = (\lambda\, x\,.\, \{\, y \in_i \bigcup_i\, x \mid \forall_i\, a \in_i x\,.\, y \in_i a\,\}), \\
\qquad \phi_i = (\lambda\, x, y, a, b\,.\, (a =_{\mathsf{d}_i} \emptyset_i \land b =_{\mathsf{d}_i} x) \lor (a =_{\mathsf{d}_i} \mathcal{P}_i\, \emptyset_i \land b =_{\mathsf{d}_i} y)), \\
\qquad \mathsf{upair}_i = (\lambda\, x, y\,.\, \mathcal{R}_i\, (\phi_i\, x\, y)\, (\mathcal{P}_i\, (\mathcal{P}_i\, \emptyset_i))), \\
\qquad \mathsf{kpair}_i = (\lambda\, x, y\,.\, \{\{x,y\}_i, \{x\}_i\}_i), \\
\qquad \pi^1_i = (\lambda\, p\,.\, \bigcup_i \bigcap_i p), \quad \pi^2_i = (\lambda\, p\,.\, \bigcup_i \{\, x \in_i \bigcup_i\, p \mid x \neq \pi^1_i\, p\,\}), \\
\qquad \times_i = (\lambda\, x, y\,.\, \bigcup_i \{\, z \mid \exists_i\, a \in_i x\,.\, z = \{\, p \mid \exists_i\, b \in_i y\,.\, p = \langle a, b\rangle_i\,\}\,\}), \\
\qquad \cup_i = (\lambda\, x, y\,.\, \bigcup_i\, \{x,y\}_i), \quad \mathsf{Tr}_i = (\lambda\, x\,.\, \mathsf{Set}_i\, X \land \forall_i\, y \in_i X\,.\, y \subseteq_i X), \\
\qquad \mathsf{Ord}_i = (\lambda\, x\,.\, \mathsf{Tr}_i\, x \land (\forall_i\, y \in_i x\,.\, \mathsf{Tr}_i\, y)), \\
\qquad <_i = (\lambda\, x, y\,.\, x \in_i y \land \mathsf{Ord}_i\, y), \\
\qquad \mathsf{Limit}_i = (\lambda\, x\,.\, \mathsf{Ord}_i\, x \land (0_i <_i x) \land (\forall_i\, y <_i x\,.\, \mathsf{succ}_i\, y <_i x)), \\
\qquad \omega_i = \bigcap_i \{\, x \in_i \mathcal{P}_i\, \mathsf{Inf}_i \mid \mathsf{Limit}_i\, x\,\}, \\
\qquad \mathsf{TagSetMems}_i = (\lambda\, a, x\,.\, \{a\}_i \times_i x), \quad \mathsf{TagOf}_i = \pi^1_i, \\
\qquad \biguplus_i = \lambda\, y\,.\, \bigcup_i\, \{\, \mathsf{TagSetMems}_i\, b\, (y\, b) \mid b \in_i \omega_i\,\}, \\
\qquad \mathsf{Part}_i = (\lambda\, a, x\,.\, \{\, y \in_i x \mid \mathsf{TagOf}_i\, y = a\,\}), \\
\qquad -_i = (\lambda\, x, y\,.\, \{\, a \in_i x \mid a \notin_i y\,\}), \\
\qquad \mathsf{OrdRec}_i =_{(\mathsf{d}_i \Rightarrow \mathsf{d}_i \Rightarrow \mathsf{d}_i) \Rightarrow \mathsf{d}_i \Rightarrow \mathsf{d}_i \Rightarrow \mathsf{d}_i} T_i\, \}
\end{array}
$$

Fig. 5. Set theoretic utilities

If X is a set and A is an object, then $\mathsf{TagSetMems}_i\, A\, X$ is the set whose set members are exactly all ordered pairs $\langle A, Y\rangle_i$ where Y is a set member of X. If $X =_{\mathsf{d}_i} \langle A, Y\rangle_i$ for some A and Y, then $\mathsf{TagOf}_i\, X =_{\mathsf{d}_i} A$. We say that X is *tagged with* A or A*-tagged* iff $\mathsf{TagOf}_i\, X =_{\mathsf{d}_i} A$.

We now describe operators that use tagging to build disjoint unions and extract partitions from disjoint unions. Let S be a term such that $\Gamma \vdash_i A \in_i \omega_i \to \mathsf{Set}_i\, (S\, A)$, i.e., S has type $\mathsf{d}_i \Rightarrow \mathsf{d}_i$ and represents a sequence of sets indexed by von Neumann natural numbers. Then $\biguplus_i S$ is a set called the *disjoint union* of S, which is the result of tagging the members of each set in the sequence S with the set's index and collecting all the tagged objects. Hence $X \in_i \biguplus_i S$ iff $X =_{\mathsf{d}_i} \langle A, Y\rangle_i$ where $Y \in_i S\, A$ for some ordinal A. If X is a set containing objects with many different tags, then $\mathsf{Part}_i\, A\, X$ gives a set whose members are exactly all of the members of X tagged with A.

For any GZF-domain, we conjecture the existence of a term T_i such that the simple definition $\mathsf{OrdRec}_i =_\tau T_i$ defines OrdRec to do transfinite recursion on the ordinals.[1] The characterisation of OrdRec_i in figure 6 is equivalent to such a definition, where $A :: \mathsf{d}_i$ and $F :: \mathsf{d}_i \Rightarrow \mathsf{d}_i \Rightarrow \mathsf{d}_i$ is such that $\Gamma \vdash_i \forall_i [\mathsf{Ord}_i]\, b\,.\, \forall_i [\mathsf{Set}_i]\, x\,.\, \mathsf{Set}\, (F\, b\, x)$. The set A is used for the zero case, F is used for the successor case, and unions are taken at limit ordinals.

[1] Our belief is based on tracing the expansion of uses of `transrec3` in Isabelle/ZF.

$$\begin{array}{rl} (\mathsf{OrdRec}_i\, F\, A\, 0_i) & = A \\ \forall_i[\mathsf{Ord}_i]\, b\,.\ (\mathsf{OrdRec}_i\, F\, A\, (\mathsf{succ}_i\, b)) & = F\, (\mathsf{succ}_i\, b)\, (\mathsf{OrdRec}_i\, F\, A\, b) \\ \forall_i[\mathsf{Limit}_i]\, z\,.\ (\mathsf{OrdRec}_i\, F\, A\, z) & = \bigcup_i \{\, \mathsf{OrdRec}_i\, F\, A\, b \mid b \in_i z \,\} \end{array}$$

$$\begin{array}{l} \mathsf{Model}_{i,j} := \{\, \mathsf{Tier}_{i,j} = \mathsf{OrdRec}_i\, (\lambda\, z, x\,.\, x \cup_i \biguplus_i (\lambda\, y\,.\, \mathsf{Ops}_{i,j}\, y\, z\, (x -_i \mathsf{Ignored}_{i,j}))) \\ \qquad\qquad\qquad\qquad\qquad\qquad (\biguplus_i (\lambda\, y\,.\, \mathsf{Ops}_{i,j}\, y\, 0_i\, \emptyset_i)), \\ \qquad\qquad \mathsf{inModel}_{i,j} = (\lambda\, x\,.\, \exists_i[\mathsf{Ord}_i]\, a\,.\, x \in_i (\mathsf{Tier}_{i,j}\, a)), \\ \qquad\qquad \overline{\forall}_{i,j} = (\lambda\, p\,.\, \forall_i[\mathsf{inModel}_{i,j}]\, x\,.\, p\, x) \,\} \end{array}$$

Fig. 6. Recursion equations, and simple definitions for building a model for d_j in d_i

4.2 Model Framework

The constant $\mathsf{Ops}_{i,j}$ acts as a table of operations used for building in d_i the tiers of a model for d_j. The constant $\mathsf{Ignored}_{i,j}$ is a set of objects which are not to be used in building further objects. The user must axiomatize both of these constants. For this to work, if A and B are ordinals, then $\mathsf{Ops}_{i,j}\, A\, B$ must be an operator which returns a set when given a set. We call the A-indexed aspect of $\mathsf{Ops}_{i,j}$ the *slot* A. Each slot is used for a different kind of mathematical object, e.g., set, non-set ordered pair, non-set natural number, etc. When building a model, $\mathsf{Ops}_{i,j}\, A\, B$ is given the previous model tier minus the ignored objects and returns a set of objects, each of which is then tagged by A before being added to the next tier.

For each pair of domain types, $\mathsf{Model}_{i,j}$ in figure 6 is a set of simple definitions that builds a model in d_i for d_j and gives a membership predicate and a $\forall$-quantifier restricted to the model. The operator $\mathsf{Tier}_{i,j} :: \mathsf{d}_i \Rightarrow \mathsf{d}_i$ uses OrdRec_i to map d_i ordinals to model tiers. The formula $\mathsf{inModel}_{i,j}\, X$ holds if there exists an ordinal A such that $\Gamma \vdash_k X \in_i (\mathsf{Tier}_{i,j}\, A)$. The quantifier $\overline{\forall}_{i,j}$ allows quantification over the model by restricting $\forall_i$ to objects satisfying $\mathsf{inModel}_{i,j}$.

Figure 7 defines a function $\mathsf{trans}_{i,j}$ for translating formulas that speak about d_j to formulas that speak about the model in d_i for d_j. The function is defined recursively on terms mostly by translating constants to their "model versions". For example $\mathsf{trans}_{i,j}(\forall_j) \equiv \overline{\forall}_{i,j}$, and $\mathsf{trans}_{i,j}(\mathcal{P}_j) \equiv \overline{\mathcal{P}}_i$. Sets of formulas can also be translated. For example, we use $\mathsf{trans}_{i,j}(\mathsf{FOLQuants}_j)$ to generate extra quantifiers relativized to a model.

4.3 GZF Models

We now show how to configure the set slot of $\mathsf{Ops}_{i,j}$ to obtain a model satisfying GZF. We reserve slot 1 for sets. Each model tier must contain all subsets of all previous tiers, tagged with 1. Figure 7 defines the formula set $\mathsf{ZFOps}_{i,j}$ that specifies that $\mathsf{Ops}_{i,j}$ invokes the power set operator ($\mathcal{P}_i$) in slot 1 at each successor ordinal. The formulas in $\mathsf{ZFOps}_{i,j}$ allow proving that every 1-tagged set of model sets belongs to some model tier. A crucial fact used for demonstrating this is:

$$\Gamma \vdash_i \forall_i[\mathsf{Ord}]\, b\,.\, \{1_i\} \times_i (\mathcal{P}_i\, (\mathsf{Tier}_{i,j}\, b)) \subseteq_i \mathsf{Tier}_{i,j}\, (\mathsf{succ}_i\, b)$$

$$\mathsf{swap}_{i,j}(\star) \;:=\; \star \qquad \mathsf{swap}_{i,j}(\mathsf{d}_i) \;:=\; \mathsf{d}_j \qquad \mathsf{swap}_{i,j}(\mathsf{d}_j) \;:=\; \mathsf{d}_i$$
$$\mathsf{swap}_{i,j}(\sigma \Rightarrow \tau) \;:=\; \mathsf{swap}_{i,j}(\sigma) \Rightarrow \mathsf{swap}_{i,j}(\tau)$$

$$\begin{aligned}
\mathsf{trans}_{i,j}(x, m) &:= m(x)\\
\mathsf{trans}_{i,j}(\forall_{\mathsf{d}_j}, m) &:= \overline{\forall}_{i,j}\\
\mathsf{trans}_{i,j}(\heartsuit_j, m) &:= \overline{\heartsuit}_{i,j} \quad \text{if } \heartsuit \in \{\imath, \forall^{\neq\bullet}\}\\
\mathsf{trans}_{i,j}(\heartsuit_j, m) &:= \overline{\heartsuit}_i \quad \text{otherwise, e.g., } \heartsuit_j \equiv \mathsf{Set}_j\\
\mathsf{trans}_{i,j}(\forall_\tau, m) &:= \forall_{\mathsf{swap}_{i,j}(\tau)} \quad \text{if } \tau \neq \mathsf{d}_k \text{ for any } k \in \mathbb{N}\\
\mathsf{trans}_{i,j}(=_\tau, m) &:= (=_{\mathsf{swap}_{i,j}(\tau)})\\
\mathsf{trans}_{i,j}(\neq_\tau, m) &:= (\neq_{\mathsf{swap}_{i,j}(\tau)})\\
\mathsf{trans}_{i,j}(B\,C, m) &:= \mathsf{trans}_{i,j}(B, m)\,\mathsf{trans}_{i,j}(C, m)\\
\mathsf{trans}_{i,j}(\lambda\, x : \tau\,.\,B, m) &:= \lambda\, y : \mathsf{swap}_{i,j}(\tau)\,.\,\mathsf{trans}_{i,j}(B, m[x \mapsto y])\\
\mathsf{trans}_{i,j}(\varGamma) &:= \{\, \mathsf{trans}_{i,j}(\varphi, \varnothing) \mid \varphi \in \varGamma \,\}
\end{aligned}$$

$$\mathsf{ZFOps}_{i,j} := \{\, \mathsf{Ops}_{i,j}\, 1_i\, 0_i \;= (\lambda\, x\,.\,\emptyset_i), \quad \forall_i[\mathsf{Ord}_i]\, b\,.\,\mathsf{Ops}_{i,j}\, 1_i\, (\mathsf{succ}_i\, b) \;= \mathcal{P}_i \,\}$$

$$\begin{aligned}
\mathsf{ZFModelDefs}_i := \{\, &\overline{\emptyset}_i = \langle 1_i, \emptyset_i \rangle_i, \quad \overline{\mathsf{Set}}_i = (\lambda\, x\,.\,\mathsf{TagOf}_i\; x \;=_{\mathsf{d}_i}\; 1_i),\\
&\overline{\in}_i = (\lambda\, x, y\,.\,\overline{\mathsf{Set}}_i\; y \wedge x \in_i (\pi^2_i\; y)),\\
&\overline{\subseteq}_i = (\lambda\, x, y : \mathsf{d}_i\,.\,\overline{\mathsf{Set}}_i\; x \wedge \overline{\mathsf{Set}}_i\; y \wedge \forall_i\; a\, \overline{\in}_i\; x\,.\,a\, \overline{\in}_i\; y),\\
&\overline{\mathcal{P}}_i = (\lambda\, x\,.\,\langle 1_i, \mathsf{TagSetMems}_i\; 1_i\; (\mathcal{P}_i\; (\pi^2_i\; x))\rangle),\\
&\overline{\bigcup}_i = (\lambda\, x\,.\,\langle 1_i, \bigcup_i \{\, \pi^2_i\; y \mid y \in_i (\pi^2_i\; x) \,\}\rangle),\\
&\overline{\mathsf{succ}}_i = (\lambda\, x\,.\,\overline{\bigcup}_i\; \langle 1_i, \{x, \langle 1_i, \{x\}\rangle\}\rangle_i),\\
&\Theta_i = (\lambda\, a\,.\,\mathsf{OrdRec}_i\; (\lambda\, b, x\,.\,\{\, \overline{\mathsf{succ}}_i\; y \mid y \in_i x \,\})\; \{\langle 1_i, \emptyset_i\rangle_i\}_i\; a),\\
&\overline{\mathsf{Inf}}_i = \langle 1_i \Theta_i\; \omega_i, , \rangle_i\\
&\overline{\mathcal{R}}_i = (\lambda\, p, x\,.\,\langle 1_i, \mathcal{R}_i\; p\, (\pi^2_i\; x)\rangle_i) \,\}
\end{aligned}$$

$$\mathsf{BuildModel}_{i,j} := \mathsf{ZFUtils}_i \cup \mathsf{ZFOps}_{i,j} \cup \mathsf{Model}_{i,j} \cup \mathsf{trans}_{i,j}(\mathsf{FOLQuants}_j) \cup \mathsf{ZFModelDefs}_i$$

Fig. 7. Definition of $\mathsf{swap}_{i,j}$ on types and $\mathsf{trans}_{i,j}$ and formula sets for model building

Figure 7 defines $\mathsf{ZFModelDefs}_i$ as a set of simple definitions for each model constant in $\mathsf{trans}_{i,j}(\mathsf{GZFConsts}_j)$. Because the definitions in $\mathsf{ZFModelDefs}_i$ only make use of the set slot of the model, they can be shared amongst all models we build in d_i. The constants in $\mathsf{trans}_{i,j}(\mathsf{GZFConsts}_j)$ act on the "model sets", and have been shown to satisfy the formulas in $\mathsf{trans}_{i,j}(\mathsf{GZF}_j)$ when used in a model. Figure 7 also defines $\mathsf{BuildModel}_{i,j}$ as a set of simple definitions for (1) set theoretic utilities for model building, including ordinal recursion, (2) specifying slot 1 of $\mathsf{Ops}_{i,j}$ to invoke $(\mathcal{P}_i)$ at successor ordinals (3) building model tiers, checking model membership, quantifying over the model, (4) extra quantifiers relativized to the model, and (5) simple definitions for $\mathsf{trans}_{i,j}(\mathsf{GZFConsts}_j)$.

We say that $\varGamma$ builds a *GZF-model* in d_i for d_j iff $\varGamma \vdash_k \mathsf{trans}_{i,j}(\mathsf{GZF}_j)$. We have proved that if $\varGamma \vdash_k (\text{ALLI}_{k,i}), (\text{ALLE}_{k,i})$ and $\varGamma \vdash_k \mathsf{Base}_i \cup \mathsf{BuildModel}_{i,j}$, then $\varGamma$ builds a GZF-model in d_i for d_j.

$$\begin{aligned}\mathsf{Connection}_{i,j} := \{ & \overline{\forall}_{i,j}\, x \,.\, \mathsf{Rep}_{i,j}\, (\mathsf{Abs}_{i,j}\, x) = x, \quad \forall_j = \mathsf{swap}_{i,j}(\overline{\forall}_{i,j}),\\ & \forall_j\, y \,.\, \mathsf{Abs}_{i,j}\, (\mathsf{Rep}_{i,j}\, y) = y, \quad \forall_j\, y \,.\, \mathsf{inModel}_{i,j}\, (\mathsf{Rep}_{i,j}\, y) \}\end{aligned}$$

$$\mathsf{swap}_{i,j}(B) := \begin{cases} \mathsf{swap}_{i,j}(C)\, \mathsf{swap}_{i,j}(D) & \text{if } B :: \star,\ B = C\, D \\ B & \text{if } B :: \star,\ B \in \mathsf{Var} \cup \mathsf{Const} \\ \mathsf{Abs}_{i,j}\ B & \text{if } B :: \mathsf{d}_i \\ \mathsf{Rep}_{i,j}\ B & \text{if } B :: \mathsf{d}_j \\ (\lambda\, x : \mathsf{swap}_{i,j}(\sigma) \,.\, \mathsf{swap}_{i,j}(B\, (\mathsf{swap}_{i,j}(x)))) & \text{if } B :: \sigma \Rightarrow \tau \end{cases}$$

$$\mathsf{Delegate}_{i,j}(\mathcal{C}) := \{\, c =_\tau \mathsf{swap}_{i,j}(\mathsf{trans}_{i,j}(c)) \mid c \in \mathcal{C}, c :: \tau \,\}$$

$$\mathsf{AbsModel}_{i,j} := \mathsf{Connection}_{i,j} \cup \mathsf{FOLQuants}_j \cup \mathsf{Delegate}_{i,j}(\mathsf{GZFConsts}_j)$$

Fig. 8. Formulas axiomatising $\mathsf{Abs}_{i,j}$ and $\mathsf{Rep}_{i,j}$, definitions of $\mathsf{swap}_{i,j}$ on terms and $\mathsf{Delegate}_{i,j}$, and formulas for connecting a model built in d_i to d_j

5 Connecting Models to Domains

Section 3 showed how to axiomatize a GST in domain d_i directly using d_i as the founder domain. We now show how to combine an axiomatization Γ_i of a GST S_1 in domain d_i with model building definitions $\Psi_{i,j}$ to justify an axiomatization Γ_j of a GST S_2 in domain d_j so that $\Gamma_i \cup \Psi_{i,j} \vdash_i \Gamma_j$. This connects S_2 to a model for it built in S_1, which supports stating that S_2 is consistent if S_1 is.

Start by assuming that $\Gamma \vdash_k \mathsf{BuildModel}_{i,j}$ and we will connect the model built in d_i to d_j so we can prove things about d_j using $\vdash_k$. Figure 8 defines the set $\mathsf{Connection}_{i,j}$ that axiomatizes that the operators $\mathsf{Abs}_{i,j} :: \mathsf{d}_i \Rightarrow \mathsf{d}_j$ and $\mathsf{Rep}_{i,j} :: \mathsf{d}_j \Rightarrow \mathsf{d}_i$ are an isomorphism between the objects satisfying $\mathsf{Tier}_{i,j}$ and d_j. Figure 8 defines the meta-level function $\mathsf{swap}_{i,j}$ that translates terms with types involving the abstract domain d_j to corresponding terms with types involving the representation domain d_i, and vice versa. We also define $\mathsf{Delegate}_{i,j}$ to generate simple definitions for a set of constants for use in d_j in terms of the translation of those constants to corresponding constants for use with the model in d_i. In particular, swapping $\overline{\forall}_{i,j}$ supplies a definition for $\forall_j$ such that $(\text{ALLI}_{i,j})$, $(\text{ALLE}_{i,j})$ are admissible with $\vdash_i$.

If $\Gamma \vdash_k \mathsf{Base}_i \cup \mathsf{BuildModel}_{i,j}$, then we can give simple definitions for $\mathsf{GZFConsts}_j$ using $\mathsf{Delegate}_{i,j}(\mathsf{GZFConsts}_j)$. Hence we define $\mathsf{AbsModel}_{i,j}$ in figure 8 as the set of formulas which (1) axiomatizes an isomorphism between members of d_i satisfying $\mathsf{Tier}_{i,j}$ and d_j and (2) gives simple definitions for quantifiers over d_j and $\mathsf{GZFConsts}_j$ by swapping their model versions in d_i. To prove that the swapped constants and quantifiers form a GZF-domain, we show that if $\Gamma \vdash_k$ $(\text{ALLI}_{k,i})$, $(\text{ALLE}_{k,i})$ and $\Gamma \vdash_k \mathsf{Base}_i \cup \mathsf{BuildModel}_{i,j} \cup \mathsf{AbsModel}_{i,j}$, then $\Gamma \vdash_k$ GZF_j. This is achieved by expanding the delegated definitions of $\mathsf{GZFConsts}_j$ in each formula of GZF_j. In practice, the instances of $\mathsf{Abs}_{i,j}$ and $\mathsf{Rep}_{i,j}$ in these formulas cancel each other out because the terms they are applied to always belong to the model. We are then left with exactly the formulas of $\mathsf{trans}_{i,j}(\mathsf{GZF}_j)$, which hold because $\Gamma \vdash_k \mathsf{BuildModel}_{i,j}$ can be shown to entail these formulas.

$$
\begin{aligned}
\mathsf{RestrictOps}_{i,j}(\beta_1,\ldots,\beta_n) &:= \forall_i[\mathsf{Ord}_i]\,\alpha\,.\,(\alpha \neq \beta_1 \wedge \ldots \wedge \alpha \neq \beta_n) \\
&\qquad \rightarrow \mathsf{Ops}_{i,j}\ \alpha \ = \ (\lambda\,\beta, x\,.\,\emptyset_i)
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{PairOps}_{i,j} &:= \{\,\mathsf{Ops}_{i,j}\ 3_i\ 0_i \ = (\lambda\,x\,.\,\emptyset_i), \\
&\qquad \forall_i[\mathsf{Ord}_i]\,\beta <_i 0_i\,.\,\mathsf{Ops}_{i,j}\ 3_i\ \beta \ = (\lambda\,x\,.\,x \times_i x)\,\} \\
\mathsf{NatOps}_{i,j} &:= \{\,\forall_i[\mathsf{Ord}_i]\,\beta <_i \omega_i\,.\,\mathsf{Ops}_{i,j}\ 5_i\ \beta \ = (\lambda\,x\,.\,\{\beta\}_i), \\
&\qquad \forall_i[\mathsf{Ord}_i]\,\omega_i <_i \beta\,.\,\mathsf{Ops}_{i,j}\ 5_i\ \beta \ = (\lambda\,x\,.\,\emptyset_i)\,\} \\
\mathsf{ExOps}_{i,j} &:= \{\,\mathsf{Ops}_{i,j}\ 7_i\ 0_i \ = (\lambda\,x\,.\,\{\emptyset_i\}_i), \\
&\qquad \forall_i[\mathsf{Ord}_i]\,\beta\,.\,\mathsf{Ops}_{i,j}\ 7_i\ \beta^{+_i} \ = (\lambda\,x\,.\,\emptyset_i)\,\}
\end{aligned}
$$

$$
\begin{aligned}
\mathsf{PairModelDefs}_i &:= \{\,\overline{\mathsf{pair}}_i = (\lambda\,x, y\,.\,\langle 3_i, \langle x, y\rangle_i\rangle_i), \\
&\qquad \overline{\mathsf{Pair}}_i = (\lambda\,p\,.\,\mathsf{TagOf}_i\ p \ = 3_i)\,\} \\
\mathsf{NatModelDefs}_i &:= \{\,\overline{\mathbf{0}}_i = \langle 5_i, 0_i\rangle_i \\
&\qquad \overline{\mathbf{S}}_i = (\lambda\,x\,.\,\langle 5_i, \mathsf{succ}_i\ (\pi^2_i\ x)\rangle_i), \\
&\qquad \overline{\mathsf{Nat}}_i = (\lambda\,n\,.\,\mathsf{TagOf}_i\ n \ = 5_i)\,\} \\
\mathsf{ExModelDefs}_{i,j} &:= \{\,\overline{\bullet}_i = \langle 7_i, \emptyset_i\rangle_i, \\
&\qquad \bar{\imath}_{i,j} = (\lambda\,p\,.\,\imath^{\mathsf{Set}}_i\ x\,.\,\mathsf{inModel}_{i,j}\ x \wedge p\,x), \\
&\qquad \overline{\forall}^{\neq\bullet}_{i,j} = (\lambda\,p\,.\,\overline{\forall}_{i,j}\ x \neq \overline{\bullet}\,.\,p\,x)\,\}
\end{aligned}
$$

Fig. 9. Specifications of $\mathsf{Ops}_{i,j}$ and simple definitions for model constants

6 Examples of Models of GSTs

We now build models for each of the GSTs shown in section 3. We use d_0 as our founder domain with ZF_0 as axioms.

We build a model of **ZF** in d_0 for d_2, then of **ZFP** in d_2 for d_4, then of **ZFN** in d_4 for d_6, then of **ZFE** in d_6 for d_8, and finally of $\mathbf{ZF}^+$ in d_0 for d_{10}. First we define a meta-level function in figure 9 for building formulas which restrict $\mathsf{Ops}_{i,j}$ to only invoke certain slots. We then define specifications of $\mathsf{Ops}_{i,j}$ in figure 9 for the **Set**, **Nat** and **Exception** features, and simple definitions for the model translations of each constant in their signatures. The sets of formulas $\mathit{\Psi}_{\overline{\mathsf{ZF}}}$, $\mathit{\Psi}_{\overline{\mathsf{ZFP}}}$, $\mathit{\Psi}_{\overline{\mathsf{ZF}\mathbb{N}}}$, $\mathit{\Psi}_{\overline{\mathsf{ZFE}}}$, and $\mathit{\Psi}_{\overline{\mathsf{ZF}^+}}$ in figure 10 build models according to these specifications, including the simple definitions for acting on these models. The case for ZFE and ZF^+ is again more complex, requiring generation of definitions for model quantifiers using $\overline{\forall}^{\neq\bullet}_{i,j}$. Finally, we define the sets of formulas $\mathit{\Psi}_{\mathsf{ZF}}$, $\mathit{\Psi}_{\mathsf{ZFP}}$, $\mathit{\Psi}_{\mathsf{ZF}\mathbb{N}}$, $\mathit{\Psi}_{\mathsf{ZFE}}$, and $\mathit{\Psi}_{\mathsf{ZF}^+}$ which connect each of the models to $\mathsf{d}_2, \mathsf{d}_4, \mathsf{d}_6, \mathsf{d}_8, \mathsf{d}_{10}$ respectively, and delegate the constants of each signature.

We now briefly explain how to prove that $\mathit{\Psi}_{\mathsf{ZF}} \vdash_0 \mathsf{ZF}_2$, $\mathit{\Psi}_{\mathsf{ZFP}} \vdash_0 \mathsf{ZFP}_4$, $\mathit{\Psi}_{\mathsf{ZF}\mathbb{N}} \vdash_0 \mathsf{ZF}\mathbb{N}_6$, $\mathit{\Psi}_{\mathsf{ZFE}} \vdash_0 \mathsf{ZFE}_8$, and $\mathit{\Psi}^+_{\mathsf{ZF}} \vdash_0 \mathsf{ZF}^+_8$. Because $\mathit{\Psi}_{\overline{\mathsf{ZF}}} \vdash_0$ $(\textsc{AllI}_{0,0})$, $(\textsc{AllE}_{0,0})$ and $\mathit{\Psi}_{\overline{\mathsf{ZF}}} \vdash_0 \mathsf{BuildModel}_{0,2}$, we have that $\mathit{\Psi}_{\overline{\mathsf{ZF}}} \vdash_0 \mathsf{trans}_{0,2}(\mathsf{GZF}_2)$. Then because $\mathit{\Psi}_{\mathsf{ZF}} \vdash_0 \mathsf{AbsModel}_{0,2}$, we have that $\mathit{\Psi}_{\mathsf{ZF}} \vdash_0 \mathsf{Base}_0$ and $\mathit{\Psi}_{\mathsf{ZF}} \vdash_0$ $(\textsc{AllI}_{0,2})$. The same argument can be repeated for the other instances of $\mathit{\Psi}$, with the exception of $\mathit{\Psi}_{\mathsf{ZFE}}$ and $\mathit{\Psi}_{\mathsf{ZF}^+}$ for which we are required to show $\mathit{\Psi}_{\mathsf{ZFE}} \vdash_0 \mathsf{Base}_8[\forall_8 := \forall^{\neq\bullet}_8]$ and

$$
\begin{aligned}
\mathit{\Psi}_{\overline{\mathsf{ZF}}} &:= \mathsf{ZF}_0 \cup \mathsf{BuildModel}_{0,2} + \mathsf{RestrictOps}_{0,2}(1) + \mathsf{Ignored}_{0,2} = \emptyset_0 \\
\mathit{\Psi}_{\overline{\mathsf{ZFP}}} &:= \mathit{\Psi}_{\mathsf{ZF}} \cup \mathsf{BuildModel}_{2,4} \cup \mathsf{PairOps}_{2,4} \cup \mathsf{PairModelDefs}_2 \\
&\quad + \mathsf{RestrictOps}_{2,4}(1,3) + \mathsf{Ignored}_{2,4} = \emptyset_2 \\
\mathit{\Psi}_{\overline{\mathsf{ZF}\mathbb{N}}} &:= \mathit{\Psi}_{\mathsf{ZFP}} \cup \mathsf{BuildModel}_{4,6} \cup \mathsf{NatOps}_{4,6} \cup \mathsf{NatModelDefs}_4 \\
&\quad + \mathsf{RestrictOps}_{4,6}(1,5) + \mathsf{Ignored}_{4,6} = \emptyset_4 \\
\mathit{\Psi}_{\overline{\mathsf{ZFE}}} &:= \mathit{\Psi}_{\mathsf{ZFE}} \cup \mathsf{BuildModel}_{6,8}[\overline{\forall}_{6,8} := \overline{\forall}^{\neq\bullet}_{6,8}] \cup \mathsf{ExOps}_{6,8} \cup \mathsf{ExModelDefs}_6 \\
&\quad + \mathsf{RestrictOps}_{6,8}(1,7) + \mathsf{Ignored}_{6,8} = \{\overline{\bullet}_6\}_6 \\
\mathit{\Psi}_{\overline{\mathsf{ZF}+}} &:= \mathsf{ZF}_0 \cup \mathsf{BuildModel}_{0,10}[\overline{\forall}_{0,10} := \overline{\forall}^{\neq\bullet}_{0,10}] \\
&\quad \cup \mathsf{PairOps}_{0,10} \cup \mathsf{NatOps}_{0,10} \cup \mathsf{ExOps}_{0,10} \\
&\quad \cup \mathsf{PairModelDefs}_{10} \cup \mathsf{NatModelDefs}_{10} \cup \mathsf{ExModelDefs}_{10} \\
&\quad + \mathsf{RestrictOps}_{0,10}(1,7) + \mathsf{Ignored}_{0,10} = \{\overline{\bullet}_0\}_0
\end{aligned}
$$

$$
\begin{aligned}
\mathit{\Psi}_{\mathsf{ZF}} &:= \mathit{\Psi}_{\overline{\mathsf{ZF}}} \cup \mathsf{AbsModel}_{0,2} \\
\mathit{\Psi}_{\mathsf{ZFP}} &:= \mathit{\Psi}_{\overline{\mathsf{ZFP}}} \cup \mathsf{AbsModel}_{2,4} \cup \mathsf{Delegate}_{2,4}(\mathsf{PConsts}_4) \\
\mathit{\Psi}_{\mathsf{ZF}\mathbb{N}} &:= \mathit{\Psi}_{\overline{\mathsf{ZF}\mathbb{N}}} \cup \mathsf{AbsModel}_{4,6} \cup \mathsf{Delegate}_{2,4}(\mathsf{NConsts}_6) \\
\mathit{\Psi}_{\mathsf{ZFE}} &:= \mathit{\Psi}_{\overline{\mathsf{ZFE}}} \cup \mathsf{Connection}_{6,8} \cup \mathsf{Delegate}_{6,8}(\mathsf{GZFConsts}_8 \cup \mathsf{EConsts}_8) \\
&\quad \cup \{\forall^{\neq\bullet}_8 =_{(\mathsf{d}_8 \Rightarrow \star) \Rightarrow \star} \mathsf{swap}_{6,8}(\overline{\forall}^{\neq\bullet}_{6,8})\} \cup \mathsf{FOLQuants}_8[\forall_8 := \forall^{\neq\bullet}_8] \\
\mathit{\Psi}_{\mathsf{ZF}+} &:= \mathit{\Psi}_{\overline{\mathsf{ZF}+}} \cup \mathsf{Connection}_{0,10} \\
&\quad \cup \mathsf{Delegate}_{0,10}(\mathsf{GZFConsts}_{10} \cup \mathsf{PConsts}_{10} \cup \mathsf{NConsts}_{10} \cup \mathsf{EConsts}_{10}) \\
&\quad \cup \{\forall^{\neq\bullet}_{10} =_{(\mathsf{d}_{10} \Rightarrow \star) \Rightarrow \star} \mathsf{swap}_{0,10}(\overline{\forall}^{\neq\bullet}_{0,10})\} \cup \mathsf{FOLQuants}_{10}[\forall_{10} := \forall^{\neq\bullet}_{10}]
\end{aligned}
$$

Fig. 10. Sets of formulas for building and abstracting models for GSTs

$\mathit{\Psi}_{\mathsf{ZF}+} \vdash_0 \mathsf{Base}_{10}[\forall_{10} := \forall^{\neq\bullet}_{10}]$. With some work, we can also show:

$$
\begin{aligned}
&\mathit{\Psi}_{\overline{\mathsf{ZFP}}} \vdash_0 \mathsf{trans}_{2,4}(\mathsf{PTheory}_4), \qquad \mathit{\Psi}_{\overline{\mathsf{ZF}\mathbb{N}}} \vdash_0 \mathsf{trans}_{4,6}(\mathsf{NTheory}_4), \\
&\mathit{\Psi}_{\overline{\mathsf{ZFE}}} \vdash_0 \mathsf{trans}_{6,8}(\mathsf{ETheory}_4), \\
&\mathit{\Psi}_{\overline{\mathsf{ZF}+}} \vdash_0 \mathsf{trans}_{8,10}(\mathsf{PTheory}_{10} \cup \mathsf{NTheory}_{10} \cup \mathsf{ETheory}_{10})
\end{aligned}
$$

The translations of $\mathsf{AllDistinct}$ and WF formulas are easy to prove from the structure of the model. After this, we have that $\mathit{\Psi}_{\mathsf{ZF}} \vdash_0 \mathsf{ZF}_2$, $\mathit{\Psi}_{\mathsf{ZFP}} \vdash_0 \mathsf{ZFP}_4$, $\mathit{\Psi}_{\mathsf{ZF}\mathbb{N}} \vdash_0 \mathsf{ZF}\mathbb{N}_6$, $\mathit{\Psi}_{\mathsf{ZFE}} \vdash_0 \mathsf{ZFE}_8$, and $\mathit{\Psi}^{+}_{\mathsf{ZF}} \vdash_0 \mathsf{ZF}^{+}_{10}$.

We now argue that the reasoning above can be completed to conclude the consistency of ZF_2, ZFP_4, $\mathsf{ZF}\mathbb{N}_6$, ZFE_8, and ZF^{+}_{10}. We begin with belief in the consistency of first-order logic and ZF, which are embedded in our system as Base_0. We now discuss why we believe consistency is preserved by our methods of extending Base_0 to $\mathit{\Psi}_{\mathsf{ZF}}$, $\mathit{\Psi}_{\mathsf{ZF}}$ to $\mathit{\Psi}_{\mathsf{ZFP}}$, and so on. Most of the extensions are done by adding simple definitions, which preserve consistency. We have not yet written the term T_i in the simple definition for OrdRec_i, but we believe this can be done because Isabelle/ZF does it. Our specifications of $\mathsf{Ops}_{i,j}$ and $\mathsf{RestrictOps}_{i,j}$ are currently not simple definitions, but we believe we know how to reformulate them as simple definitions. The axiomatizations of $\mathsf{Abs}_{i,j}$ and $\mathsf{Rep}_{i,j}$ are not simple definitions, but this technique is widely used in Isabelle/HOL and has been argued to preserve consistency by Kunčar and Popescu [13].

7 Conclusion and Future Work

This paper presented methods for generating custom set theories intended to be more suitable for the formalisation of mathematics by being closer to mathematical practice. Our logical framework and toolkit supports reasoning about axiomatizations and models for a variety of GSTs. We show how to define ZF as a GST and give four examples of how to extend ZF with non-set features. We show how to use a GST via an axiomatization and also how to use it via a connection to a model.

Toward an Isabelle Implementation. We aim to mechanize the results of this paper in Isabelle/Pure using locales and overloading with type classes. This includes adapting the development of transfinite ordinal recursion in the Isabelle/ZF library to our setting.

Toward User-Friendly GST Specification and Use. We aim that users should be able to construct a structure and specify some properties of the structure and request a fresh copy of it and the system should be able to generate a new GST domain where that structure exists as non-set objects with no other properties than those specified. We also aim that users should be able to specify identifications (e.g., quotienting) and then have a GST generated where those identifications are true. Ideally, there will be support for doing this locally within part of a formal development and the user should not need to be aware that they are temporarily operating in a new GST.

References

1. P. Aczel. Generalised set theory. In *Logic, Language and Computation*, vol. 1 of *CSLI Lecture Notes*, 1996.
2. G. Bancerek, C. Byliński, A. Grabowski, A. Korniłowicz, R. Matuszewski, A. Naumowicz, K. Pąk, J. Urban. Mizar: State-of-the-art and beyond. In *Intelligent Computer Mathematics*, LNCS. Springer, 2015.
3. C. E. Brown, K. Pak. A tale of two set theories. In *Intelligent Computer Mathematics*, LNCS. Springer, 2019.
4. C. E. Brown, G. Smolka. Extended first-order logic. In *Theorem Proving in Higher Order Logics*. Springer, 2009.
5. C. Dunne, J. B. Wells, F. Kamareddine. Adding an abstraction barrier to ZF set theory. In *Intelligent Computer Mathematics*, vol. 12236 of *LNCS*. Springer, 2020.
6. W. M. Farmer. Formalizing undefinedness arising in calculus. In *International Joint Conference on Automated Reasoning*. Springer, 2004.
7. W. M. Farmer, J. D. Guttman, F. J. Thayer. Little theories. In *Automated Deduction: CADE-11*, vol. 607 of *LNCS*. Springer-Verlag, 1992.
8. B. Huffman, O. Kunčar. Lifting and transfer: A modular design for quotients in Isabelle/HOL. In *Certified Programs and Proofs*, vol. 8307 of *LNCS*. Springer, 2013.