HEREDITARILY FINITE SETS AND PROOFGOLD'S CONSENSUS ALGORITHM

CHAD E. BROWN

Abstract. We describe the theory of hereditarily finite sets built into Proofgold. This is the theory in which pseudorandomly generated conjectures are made as part of the consensus algorithm. The generated conjectures can take a number of different forms and we discuss each possible form.

1. Introduction

Proofgold[1] is a peer to peer cryptocurrency making multiple uses of formal logic. One of the use cases is the publication of a theory (e.g., the theory of higher-order abstract syntax [6] or the Mizar style theory of sets [5]) and then developing that theory by publishing documents with definitions, conjectures and proofs. The blockchain records the theories and their state of development (e.g., which theorems have been proven and when). The idea for such a blockchain has been described and motivated in earlier work [30, 26] and is not the focus of this work. Here we focus on a different use case: using theorem proving as part of the consensus algorithm.

The Proofgold consensus algorithm is a combination of proof of burn (realized by the burning of litecoins) and proof of stake. Effectively the more Proofgold bars a node has staking, the fewer litecoins need to be burned in order to create a new block. Proof of stake has unfortunate centralizing effects since new block rewards go to those with preexisting stake, effectively increasing their stake.[2] Proofgold attempts to alleviate this centralizing effect by splitting the block reward into two parts: half of the reward goes to the staker and half of the reward becomes a bounty on a pseudorandomly generated conjecture. When someone resolves this conjecture (either by proving it or proving its negation), then they can claim that half of the reward. This may happen much later than the creation of the block.[3] This can be seen as a form of *delayed proof of work*. When someone does the proof of work in the future, they obtain stake in the system that they can use to become a staker.

Date: August 31, 2020.

Czech Technical University in Prague.

[1] proofgold.org

[2] If all bars were being used for staking, the proportions would in principle remain the same. In practice this does not happen for several reasons. One reason is that larger stakers are less likely to have their blocks orphaned, so that larger stakers have an advantange exceeding their proportion of stake.

[3] For example, as of August 2020, over 2000 blocks have been created, leading to over 50,000 bars being placed on over 2000 pseudorandom conjectures. Of these, only 6 have been resolved leading to 150 of these bars being claimed.

The idea of placing a bounty on pseudorandomly generated conjectures is open to several criticisms. One problem is that the conjecture might be independent. That is, it might be that neither the conjecture nor its negation is provable. As a simple example, in a system with no axioms the statement $\forall xy.x = y$ is neither provable nor is its negation provable. Proofgold's solution to the problem is to use a relatively strong theory so that the conjectures are unlikely to be independent.[4] Specifically the built in theory Proofgold uses for its consensus algorithm is a higher-order theory of hereditarily finite sets (HF).

Another problem is that pseudorandomly generated conjectures might not be interesting. This problem in itself may not be too serious, as there is a case to be made that the work done as part of a proof of work system should *not* be interesting, as this creates external incentives. On the other hand, even if the conjectures are not interesting, it is likely that interesting results would be proven as lemmas from which the uninteresting conjecture can be resolved. For example, it is not particularly interesting that 57 has no integer square root, but the process of proving it might involve lemmas about integer squares that could be reused.

Even such uninteresting conjectures such as whether 57 has an integer square root is unlikely to be generated if we start with a traditional axiomatic set theory. In traditional set theories, few primitives are required (sometimes only membership itself, leaving the basic operations as implicitly given by existential axioms). If a theory followed this approach, then a "random" conjecture would only be able to mention finite ordinals if the conjecture essentially defined finite ordinals in the hypothesis. This would be unlikely in practice, at least without explicitly making such conjectures more likely to be generated. Proofgold makes these conjectures more likely to be generated by including many primitive typed constants (over 100) that talk about a variety of mathematical objects, including sets, pairs, functions, finite ordinals, equipotence and loops. The drawback to having a large theory with many primitives and many axioms is that the theory takes longer to describe. We make an attempt to describe it here, but do not dwell on many aspects that may deserve more discussion.

In Section 2 we describe the underlying framework of intuitionistic higher-order logic used by Proofgold for all theories. In Section 3 we start describing HF by giving what could be called the "proper" primitives and axioms, of which there are few. The remaining primitives have a corresponding axiom giving a defining equation. However, these are not definitions. They are primitives that could have been definitions in an alternative formulation. The fact that they are primitives make them available to be used in the pseudorandomly generated conjectures. Section 4 gives several primitives allowing us to estimate the size of a given set. Section 5 gives several primitives giving properties of binary relations. Section 6 gives primitives for set theoretic operations beyond those given with proper axioms. Section 7 gives a $\in$-recursion operator (following [4]) and gives a number of definitions using the recursion operator. Section 8

[4]To estimate the probability of a randomly generated conjecture being independent would require placing some probability distribution on conjectures corresponding to the generation process. We do not attempt to do this here. Over time empirical estimates should emerge for how likely a pseudorandomly generated Proofgold conjecture is to be independent.

defines disjoint unions (which also doubles as a notion of ordered pairs). Using disjoint unions and power sets, a primitive is given in Section 9 that allows us to give practically sized terms for sets of specific large cardinalities. Section 10 gives primitives for representing functions, sets of functions, tuples, and similar constructions. Section 11 numbers gives primitives for working with Conway's surreal numbers [12] (which in the context of HF are the dyadic rationals). Primitives for loops and notations related to The AIM Conjecture [17] are given in Section 12. Primitives for a representation of untyped combinatory logic are given in Section 13. Finally in Section 14 we discuss the classes of pseudorandomly generated conjectures that are part of Proofgold's consensus algorithm and conclude in Section 15.

2. Intuitionistic Higher-Order Logic

We begin by describing the framework underlying all Proofgold theories: intuitionistic higher-order logic. The types are simple types and the terms are simply typed λ-terms in the style of Church [11]. The proof system is a natural deduction system [23] that admits proof terms in the usual Curry-Howard-de Bruijn style [15, 9, 25].

For the sake of clarity we begin with a careful description of the set of types, the family of typed terms, the notions of free variables and substitutions, the capture avoiding substitution operation, α-conversion and $\beta\eta$-reduction. This material is standard and can be skipped by a reader familiar with these notions.

For simplicity, we assume one uninterpreted base type ι of individuals (although technically Proofgold allows theories to use multiple uninterpreted base types, with a technical limit of 65536). In addition we have a special base type o of propositions. All other types are function types of the form $(\alpha\beta)$ of functions from α to β. Such function types are often written as $(\alpha \to \beta)$. We omit the arrow since we have no other kinds of compound types. When parentheses are omitted they should be replaced to the right. That is, $\iota\iota o$ is the type $(\iota(\iota o))$. Let $\mathcal{T}$ denote the set of types.

Assume we have countably many variables at each type. Let $\mathcal{V}_\alpha$ be the set of variables of type α. A signature $\mathcal{S}$ is a typed family $(\mathcal{S}_\alpha)_{\alpha\in\mathcal{T}}$ of sets of constants. A signature is finite if $\bigcup_{\alpha\in\mathcal{T}} \mathcal{S}_\alpha$ is finite. (In practice, all Proofgold signatures will be finite.) We also assume there are no conflicts: no variable is also a constant and $\mathcal{V}_\alpha \cap \mathcal{V}_\beta = \emptyset = \mathcal{S}_\alpha \cap \mathcal{S}_\beta$ when $\alpha \neq \beta$.

We now define a family $(\Lambda_\alpha)_{\alpha\in\mathcal{T}}$ of terms recursively, where $s \in \Lambda_\alpha$ means s is a term of type α.

- (Variables) If $x \in \mathcal{V}_\alpha$, then $x \in \Lambda_\alpha$.
- (Constants) If $c \in \mathcal{S}_\alpha$, then $c \in \Lambda_\alpha$.
- (Application) If $s \in \Lambda_{\alpha\beta}$ and $t \in \Lambda_\alpha$, then $(st) \in \Lambda_\beta$.
- (Abstraction) If $x \in \mathcal{V}_\alpha$ and $t \in \Lambda_\beta$, then $(\lambda x.t) \in \Lambda_{\alpha\beta}$.
- (Implication) If $s \in \Lambda_o$ and $t \in \Lambda_o$, then $(s \to t) \in \Lambda_o$.
- (Universal Quantification) If $x \in \mathcal{V}_\alpha$ and $t \in \Lambda_o$, then $(\forall x.t) \in \Lambda_o$.

The λ and $\forall$ are called *binders* and they bind the variables that follow. We sometimes explicitly give the type of variables with the binding construct in order to indicate the type, e.g., $(\lambda x : \alpha.t)$ and $(\forall y : \beta.t)$ to indicate $x \in \mathcal{V}_\alpha$ and $y \in \mathcal{V}_\beta$. If the type is omitted and no other information is given, the reader can assume the variable has type

ι. When several binders occur in sequence and all the bound variables have the same type, we may write the binder only once. That is, $(\lambda x_1 \cdots x_n.t)$ means $(\lambda x_1. \cdots .\lambda x_n.t)$ and $(\forall x_1 \cdots x_n.t)$ means $(\forall x_1. \cdots .\forall x_n.t)$.

Parentheses are often omitted, with the convention that application associates to the left, e.g., stu means $(st)u$, and implication associates to the right, e.g., $s \to t \to u$ means $s \to (t \to u)$. We assume the scope of bound variables is as far to the right as possible consistent with parentheses, e.g., $\forall p : o.p \to q$ means $\forall p : o.(p \to q)$.

It is straightforward to define the set $\mathcal{F}(t)$ of free variables of a term as follows:

- $\mathcal{F}(x) = \{x\}$
- $\mathcal{F}(c) = \emptyset$
- $\mathcal{F}(st) = \mathcal{F}(s) \cup \mathcal{F}(t)$
- $\mathcal{F}(\lambda x.t) = \mathcal{F}(t) \setminus \{x\}$
- $\mathcal{F}(s \to t) = \mathcal{F}(s) \cup \mathcal{F}(t)$
- $\mathcal{F}(\forall x.t) = \mathcal{F}(t) \setminus \{x\}$

We say x is *free in* t if $x \in \mathcal{F}(t)$. A term t is called *closed* if $\mathcal{F}(t) = \emptyset$. A *proposition* is a term of type o and a *sentence* is a closed term of type o.

A *substitution* θ is a mapping such that $\mathsf{dom}(\theta) \subseteq \bigcup_{\alpha \in \mathcal{T}} \mathcal{V}_\alpha$ and $\theta(x) \in \Lambda_\alpha$ for all $x \in \mathsf{dom}(\theta)$. We write θ^x_t for the substitution such that $\mathsf{dom}(\theta^x_t) = \mathsf{dom}(\theta) \cup \{x\}$, $\theta^x_t(x) = t$ and $\theta^x_t(y) = \theta(y)$ for $y \in \mathsf{dom}(\theta) \setminus \{x\}$.

For each substitution θ there is a substituion operation $\hat{\theta}$ mapping Λ_α to Λ_α (for each type α). The substitution operation must avoid capturing bound variables. This can be accomplished by renaming bound variables when necessary.

- $\hat{\theta}x = \theta(x)$ if $x \in \mathsf{dom}(\theta)$.
- $\hat{\theta}x = x$ if $x \notin \mathsf{dom}(\theta)$.
- $\hat{\theta}c = c$
- $\hat{\theta}(st) = (\hat{\theta}s\ \hat{\theta}t)$.
- $\hat{\theta}(\lambda x.t) = (\lambda y.\widehat{\theta^x_y}t)$ where y is a variable (with the same type as x) such that $y \notin \mathcal{F}(\theta(z))$ for all $z \in \mathsf{dom}(\theta) \cap \mathcal{F}(\lambda x.t)$. We assume y is x if x already has this property.
- $\hat{\theta}(s \to t) = (\hat{\theta}s \to \hat{\theta}t)$.
- $\hat{\theta}(\forall x.t) = (\forall y.\widehat{\theta^x_y}t)$ where y is a variable (with the same type as x) such that $y \notin \mathcal{F}(\theta(z))$ for all $z \in \mathsf{dom}(\theta) \cap \mathcal{F}(\lambda x.t)$. We again assume y is x if x already has this property.

The most common case of substitution sends one variable $x \in \mathcal{V}_\alpha$ to a term $t \in \Lambda_\alpha$. Using our notation above we can write this substitution as $\emptyset^x_t$. We write s^x_t as shorthand for $\widehat{\emptyset^x_t}s$. Simply stated, s^x_t denotes the result of substituting t for all free occurrences of x (while avoiding capture).

We next define when two terms s and t of the same type are α-*convertible*. Informally, this means s and t are the same up to the names of bound variables. It is technically easier to recursively define a 4-ary relation between two terms (of the same type) and two substitutions θ and ψ. We write this relation as $s \sim^\theta_\psi t$ and define it as the least relation satisfying the following conditions:

$$\frac{}{\Gamma \vdash s}\ s \in \mathcal{A} \qquad \frac{}{\Gamma \vdash s}\ s \in \Gamma \qquad \frac{\Gamma \vdash s}{\Gamma \vdash t}\ s{\approx}t \qquad \frac{\Gamma, s \vdash t}{\Gamma \vdash s \to t} \qquad \frac{\Gamma \vdash s \to t \qquad \Gamma \vdash s}{\Gamma \vdash t}$$

$$\frac{\Gamma \vdash s}{\Gamma \vdash \forall x.s}\ x \in \mathcal{V}_\alpha \setminus \mathcal{F}\Gamma \qquad \frac{\Gamma \vdash \forall x.s}{\Gamma \vdash s^x_t}\ x \in \mathcal{V}_\alpha, t \in \Lambda_\alpha$$

$$\frac{}{\Gamma \vdash \forall f g : \alpha\beta.(\forall x : \alpha.fx = gx) \to f = g}\ f, g \text{ DISTINCT}$$

FIGURE 1. Natural Deduction Calculus for Intuitionistionistic HOL

- $x \sim^\theta_\psi y$ if $\hat{\theta}x = y$ and $\hat{\psi}y = x$.
- $c \sim^\theta_\psi c$.
- $(s_1 t_1) \sim^\theta_\psi (s_2 t_2)$ if $s_1 \sim^\theta_\psi s_2$ and $t_1 \sim^\theta_\psi t_2$.
- $(\lambda x.s) \sim^\theta_\psi (\lambda y.t)$ if $s \sim^{\theta^x_y}_{\psi^y_x} t$.
- $(s_1 \to t_1) \sim^\theta_\psi (s_2 \to t_2)$ if $s_1 \sim^\theta_\psi s_2$ and $t_1 \sim^\theta_\psi t_2$.
- $(\forall x.s) \sim^\theta_\psi (\forall y.t)$ if $s \sim^{\theta^x_y}_{\psi^y_x} t$.

We then say s and t are *α-convertible* if $s \sim^\emptyset_\emptyset t$. From now on, we simply say terms are the same if they are α-convertible.[5]

A *β-redex* is a term of the form $(\lambda x.s)t$ and its *β-reduct* is s^x_t. An *η-redex* is a term of the form $(\lambda x.tx)$ where $x \notin \mathcal{F}(t)$ and its *η-reduct* is t. A term is $\beta\eta$-normal if it contains no β-redex and no η-redex. It is well known that $\beta\eta$ reduction on simply typed terms terminates and is confluent, so that reduction gives a unique normal form [14]. We write $s{\approx}t$ when s and t have the same $\beta\eta$-normal form.

Before moving on to natural deduction proofs and proof terms, we introduce two new notations for specific kinds of terms. For a type α and two terms $s, t \in \Lambda_\alpha$, we write $s = t$ as notation for the term $\forall p.pst \to pts$ where $p \in \mathcal{V}_{\alpha\alpha o}$ is chosen such that $p \notin \mathcal{F}(s) \cup \mathcal{F}(t)$. We call this term *symmetric Leibniz equality*.

Given a type α, a variable $x \in \mathcal{V}_\alpha$ and a proposition t, we write $\exists x.t$ as notation for the term $\forall p.(\forall x.t \to p) \to p$ where $p \in \mathcal{V}_o$ is chosen such that $p \notin \mathcal{F}(t)$. We adopt the same notational conventions for $\exists$ as for the binders $\forall$ and λ.

Let $\mathcal{A}$ be a set of sentences we call *axioms*. The natural deduction proof system in Figure 1 defines when $\Gamma \vdash t$ holds for a finite set Γ of propositions and a proposition t. Most rules are what one expects from a natural deduction system: a hypothesis rule giving $\Gamma \vdash s$ when $s \in \Gamma$ and introduction and elimination rules for $\to$ and $\forall$. The exceptions are the following:

- We include an axiom rule: $\Gamma \vdash s$ if $s \in \mathcal{A}$.
- We include a conversion rule so that provability respects $\approx$.

[5]In the Proofgold implementation, α-convertible terms are equal, as de Bruijn indices are used [10].

$$\frac{}{\Gamma \vdash \mathsf{Known}_s : s}\ s \in \mathcal{A} \qquad \frac{}{\Gamma \vdash u : s}\ u : s \in \Gamma \qquad \frac{\Gamma \vdash \mathcal{D} : s}{\Gamma \vdash \mathcal{D} : t}\ s{\approx}t$$

$$\frac{\Gamma, u : s \vdash \mathcal{D} : t}{\Gamma \vdash (\lambda u : s.\mathcal{D}) : s \to t} \qquad \frac{\Gamma \vdash \mathcal{D} : s \to t \qquad \Gamma \vdash \mathcal{E} : s}{\Gamma \vdash (\mathcal{D}\mathcal{E}) : t}$$

$$\frac{\Gamma \vdash \mathcal{D} : s}{\Gamma \vdash (\lambda x.\mathcal{D}) : \forall x.s}\ x \in \mathcal{V}_\alpha \setminus \mathcal{F}\Gamma \qquad \frac{\Gamma \vdash \mathcal{D} : \forall x.s}{\Gamma \vdash (\mathcal{D}t) : s^x_t}\ x \in \mathcal{V}_\alpha, t \in \Lambda_\alpha$$

$$\frac{}{\Gamma \vdash \mathsf{Ext}_{\alpha,\beta} : (\forall fg : \alpha\beta.(\forall x : \alpha.fx = gx) \to f = g)}\ f, g \text{ DISTINCT}$$

FIGURE 2. Natural Deduction Calculus with Proof Terms

- We include a functional extensionality rule so that two terms of function type $\alpha\beta$ can be proven equal by proving they give the same results when applied to arbitrary arguments.[6]

We briefly outline proof terms. Let $\mathcal{H}$ be a countably infinite set of *hypothesis variables* and assume these do not conflict with our previous objects (e.g., variables, constants or terms in general). Let $\mathcal{P}$ be the set of *proof terms* given inductively as follows:

- If $u \in \mathcal{H}$, then $u \in \mathcal{P}$.
- If s is a sentence, then $\mathsf{Known}_s \in \mathcal{P}$.
- If $\mathcal{D}, \mathcal{E} \in \mathcal{P}$, then $(\mathcal{D}\mathcal{E}) \in \mathcal{P}$.
- If $\mathcal{D} \in \mathcal{P}$ and $s \in \Lambda_\alpha$, then $(\mathcal{D}s) \in \mathcal{P}$.
- If $u \in \mathcal{H}$, $s \in \Lambda_o$ and $\mathcal{D} \in \mathcal{P}$, then $(\lambda u : s.\mathcal{D}) \in \mathcal{P}$.
- If $x \in \mathcal{V}_\alpha$ and $\mathcal{D} \in \mathcal{P}$, then $(\lambda x.\mathcal{D}) \in \mathcal{P}$.
- If $\alpha, \beta \in \mathcal{T}$, then $\mathsf{Ext}_{\alpha,\beta} \in \mathcal{P}$.

Not all proof terms will correspond to proofs of propositions, but all proofs can be assigned corresponding proof terms that allow for easy checking of proofs. Let us now use Γ for sets of pairs of the form $u : s$ where $u \in \mathcal{H}$ and $s \in \Lambda_o$. That is, instead of having a finite set of hypotheses, we will have a finite set of hypotheses with labels. For the calculus with proof terms we define when $\Gamma \vdash \mathcal{D} : t$ holds for such a Γ, a proof term $\mathcal{D}$ and a proposition t. The rules with proof terms are given in Figure 2.

In practice proof terms of the form Known_s can be used for any previously proven theorem as well as axioms. Also, in the implementation the subscript s in Known_s is only the Merkle root of the sentence s, not the sentence itself.

A Proofgold theory is specified by giving a finite signature $\mathcal{S}$ of typed constants and a finite set $\mathcal{A}$ of axioms. In the next section we begin the description of the HF theory used for the Proofgold consensus algorithm. For other examples of Proofgold theories, see [6, 5].

[6]Functional extensionality can be considered optional and is only included here because it is included in the Proofgold implementation.

3. Proper Axioms of HF

There are only six constants that do not have a defining equation as an axiom:

- $\varepsilon : (\iota o)\iota$ (a "choice" operator)
- $\in: \iota\iota o$ (set membership)
- $\emptyset : \iota$ (the empty set, also the ordinal 0)
- $\bigcup : \iota\iota$ (the union operator)
- $\wp : \iota\iota$ (the power set operator)
- $\mathsf{Repl} : \iota(\iota\iota)\iota$ (the replacement operator)

We write $\in$ in infix, i.e., $s \in t$ means $\in st$. We will also write $\forall x \in s.t$ as shorthand for $\forall x.x \in s \to t$. Furthermore, we write $\{t|x \in s\}$ as notation for the term $\mathsf{Repl}\ s\ (\lambda x.t)$.

For each of the constants above there is at least one axiom giving a property the constant must satisfy. In most cases we will need additional logical connectives to state the axiom. For the case of ε we can already state the axiom.

Axiom 3.1. $\forall P : \iota o.\forall x.P\ x \to P\ (\varepsilon\ P)$.

This has the appearance of a form of the axiom of choice. In fact it will be much weaker in the context of the full theory. Suppose we have a unique existence operator (as will be given below) and the axiom above were replaced with the weaker form saying ε is a description operator:

$$\forall P : \iota o.(\exists! x.P\ x) \to P\ (\varepsilon\ P).$$

Since the hereditarily finite sets can be well-ordered, a choice operator ε' satisfying Axiom 3.1 could be defined from the description operator and the well-ordering.

Let $\subseteq$ be a constant of type $\iota\iota o$ with the following definitional axiom:

Axiom 3.2. $(\subseteq) = (\lambda XY.\forall x \in X.x \in Y)$.

As with $\in$, we will generally write $\subseteq$ in infix.

We now begin including logical constants and their defining equations. The definitions trace their roots to Russell [24] and Prawitz [23]. Let $\bot$ and $\top$ be constants of type o. These have the following definitional axioms.

Axiom 3.3. $\bot = (\forall p : o.p)$.

Axiom 3.4. $\top = (\forall p : o.p \to p)$.

Let $\neg$ be a constant of type oo with the following definitional axiom:

Axiom 3.5. $\neg = (\lambda A : o.A \to \bot)$.

Now that we have negation we will write $s \neq t$ as notation for $\neg(s = t)$. We could also give similar notations $\notin$ and $\not\subseteq$. Instead the HF theory uses two more constants $\notin$ and $\not\subseteq$ with the following definitional axioms:

Axiom 3.6. $(\notin) = (\lambda xy.\neg\ (x \in y))$.

Axiom 3.7. $(\not\subseteq) = (\lambda XY.\neg\ (X \subseteq\ Y))$.

We write $\notin$ and $\not\subseteq$ in infix.

Let $\wedge$, $\vee$ and $\leftrightarrow$ be constants of type ooo with the following definitional axioms:

Axiom 3.8. $(\wedge) = (\lambda AB : o.\forall p : o.(A \to B \to p) \to p)$.

Axiom 3.9. $(\vee) = (\lambda AB : o.\forall p : o.(A \to p) \to (B \to p) \to p)$.

Axiom 3.10. $(\leftrightarrow) = (\lambda AB : o.(A \to B) \wedge (B \to A))$.

We will write $\wedge$, $\vee$ and $\leftrightarrow$ in infix, with $\wedge$ and $\vee$ being left associative. We follow the common convention that $\wedge$ binds more tightly than $\vee$ which binds more tightly than $\leftrightarrow$. We will also write $\exists x \in s.t$ as shorthand for $\exists x.x \in s \wedge t$.

We can now state the remaining proper axioms.

The following axiom ensures the theory is classical.

Axiom 3.11. $\forall p : o.\neg\neg p \to p$.

The next axiom is a form of propositional extensionality, ensuring that two propositions are equal if they are equivalent.

Axiom 3.12. $\forall AB : o.(A \leftrightarrow B) \to A = B$.

The first proper set theory axiom is set extensionality. Two sets are equal if they are subsets of each other.

Axiom 3.13. $\forall XY.X \subseteq Y \to Y \subseteq X \to X = Y$.

The empty set axiom ensures there are no members of the empty set.

Axiom 3.14. $\neg\exists x.x \in \emptyset$.

The axiom for union characterizes when sets are elements of $\bigcup X$ in the expected way.

Axiom 3.15. $\forall Xx.x \in \bigcup X \leftrightarrow \exists Y.x \in Y \wedge Y \in X$.

The axiom for power sets states that $\wp X$ contains precisely the subsets of X.

Axiom 3.16. $\forall XY.Y \in \wp X \leftrightarrow Y \subseteq X$.

The axiom for Repl characterizes membership in $\{F\ x|x \in X\}$.

Axiom 3.17. $\forall X.\forall F : \iota\iota.\forall y.y \in \{F\ x|x \in X\} \leftrightarrow \exists x.x \in X \wedge y = F\ x$.

The next axiom essentially states ι is the least (transitive) collection containing $\emptyset$ and closed under the set theoretic operations above. This effectively states ι consts of (at most) the hereditarily finite sets.

Axiom 3.18.

$$\begin{gathered}\forall p : \iota o.(\forall X.p\ X \to \forall x \in X.p\ x) \to \\ p\ \emptyset \\ \to (\forall X.p\ X \to p\ (\bigcup X)) \\ \to (\forall X.p\ X \to p\ (\wp X)) \\ \to (\forall X.p\ X \to \forall F : \iota\iota.(\forall x \in X.p\ (F\ x)) \to p\ \{F\ x|x \in X\}) \\ \to \forall x.p\ x\end{gathered}$$

The final proper axiom is $\in$-induction. This is very likely to follow from the previous axiom, but is included here since it is technically included as an axiom of Proofgold's formulation of the HF theory.

Axiom 3.19. $\forall p : \iota o.(\forall X.(\forall x \in X.p\ x) \to p\ X) \to \forall X.p\ X$.

We finish the section with four more constants with definitional axioms that do not fit naturally into later sections.

Let $\mathsf{exactly1of2}$ (exclusive or) be a constant of type ooo with the definitional axiom:

Axiom 3.20. $\mathsf{exactly1of2} = (\lambda AB : o.A \land \neg B \lor \neg A \land B)$.

Let $\mathsf{exactly1of3}$ be a constant of type $oooo$ with the following definitional axiom:

Axiom 3.21. $\mathsf{exactly1of3} = (\lambda ABC : o.\mathsf{exactly1of2}\ A\ B \land \neg C \lor \neg A \land \neg B \land C)$.

We could have $\exists!$ as a binder notation (for variables of general types) that is expanded in terms of $\forall$ and $\to$ like we have done for $\exists$. Instead HF includes a constant for unique existence specifically at the base type ι. Let $\mathsf{exu\_i}$ be a constant of type $(\iota o)o$ with the following definitional axiom:

Axiom 3.22. $\mathsf{exu\_i} = (\lambda P : \iota o.(\exists x.P\ x) \land (\forall xy.P\ x \to P\ y \to x = y))$.

Finally we give a constant for conditionals (in the form of if-then-else). Let If be a constant of type $o\iota\iota\iota$ with the following definitional axiom:

Axiom 3.23. $\mathsf{If} = (\lambda P : o.\lambda xy.\varepsilon\ (\lambda z.P \land z = x \lor \neg P \land z = y))$.

Note that If is specifically an if-then-else constructor at type ι.

4. Basic Cardinality

We next describe predicates that allow us to express that a given set has at least or exactly a number of elements. Let $\mathsf{atleast2}$, $\mathsf{atleast3}$, $\mathsf{atleast4}$, $\mathsf{atleast5}$, $\mathsf{atleast6}$, $\mathsf{exactly2}$, $\mathsf{exactly3}$, $\mathsf{exactly4}$ and $\mathsf{exactly5}$ be constants of type ιo. Each constant has a definitional axiom, given below. Note that we purposefully use $\neg(A \subseteq B)$ instead of $(A \not\subseteq B)$ since the two variables are syntactically different. In particular $\neg(A \subseteq B)$ mentions two different constants $\neg$ and $\subseteq$ where $(A \not\subseteq B)$ only mentions one: $\not\subseteq$.

Axiom 4.1. $\mathsf{atleast2} = (\lambda X.\exists y.y \in X \land \neg(X \subseteq \wp y))$.

Axiom 4.2. $\mathsf{atleast3} = (\lambda X.\exists Y.Y \subseteq X \land (\neg(X \subseteq Y) \land \mathsf{atleast2}\ Y))$.

Axiom 4.3. $\mathsf{atleast4} = (\lambda X.\exists Y.Y \subseteq X \land (\neg(X \subseteq Y) \land \mathsf{atleast3}\ Y))$.

Axiom 4.4. $\mathsf{atleast5} = (\lambda X.\exists Y.Y \subseteq X \land (\neg(X \subseteq Y) \land \mathsf{atleast4}\ Y))$.

Axiom 4.5. $\mathsf{atleast6} = (\lambda X.\exists Y.Y \subseteq X \land (\neg(X \subseteq Y) \land \mathsf{atleast5}\ Y))$.

Axiom 4.6. $\mathsf{exactly2} = (\lambda X.\mathsf{atleast2}\ X \land \neg\mathsf{atleast3}\ X)$.

Axiom 4.7. $\mathsf{exactly3} = (\lambda X.\mathsf{atleast3}\ X \land \neg\mathsf{atleast4}\ X)$.

Axiom 4.8. $\mathsf{exactly4} = (\lambda X.\mathsf{atleast4}\ X \land \neg\mathsf{atleast5}\ X)$.

Axiom 4.9. $\mathsf{exactly5} = (\lambda X.\mathsf{atleast5}\ X \wedge \neg \mathsf{atleast6}\ X)$.

In order to generalize beyond the first few cardinalities, we define a notion of injectivity and bijectivity to define when one set has at least as many elements (or the same number of elements) as another set. Let inj and bij be constants of type $\iota\iota(\iota\iota)o$. Let atleastp and equip be constants of type $\iota\iota o$. Note that the notion of injection and bijection here is for meta-level functions of type $\iota\iota$, not functions encoded as sets in some way. These four constants each have a definitional axiom, given below.

Axiom 4.10. $\mathsf{inj} = (\lambda XY.\lambda f : \iota\iota.(\forall x \in X.fx \in Y) \wedge (\forall xy \in X.fx = fy \to x = y))$.

Axiom 4.11. $\mathsf{bij} = (\lambda XY.\lambda f : \iota\iota.\mathsf{inj}\ X\ Y\ f \wedge (\forall y \in Y.\exists x \in X.fx = y))$.

Axiom 4.12. $\mathsf{atleastp} = (\lambda XY.\exists f : \iota\iota.\mathsf{inj}\ X\ Y\ f)$.

Axiom 4.13. $\mathsf{equip} = (\lambda XY.\exists f : \iota\iota.\mathsf{bij}\ X\ Y\ f)$.

5. Properties of Binary Relations

In this section we consider a number of properties of (meta-level) binary relations. Each of these could be given for relations over a general type α, but as Proofgold has no support for polymorphism, only the type ι is considered. This is emphasized by the suffix of the name of each constant. Let reflexive_i, irreflexive_i, symmetric_i, antisymmetric_i, transitive_i, eqreln_i, per_i, linear_i, trichotomous_or_i, partialorder_i, totalorder_i, strictpartialorder_i and stricttotalorder_i be constants of type $(\iota\iota o)o$. We expect the name of each constant gives an indication of what the constant is intended to mean, and omit further explanation. Each of these has a definitional axiom, given below.

Axiom 5.1. $\mathsf{reflexive\_i} = (\lambda R : \iota\iota o.\forall x.R\ x\ x)$.

Axiom 5.2. $\mathsf{irreflexive\_i} = (\lambda R : \iota\iota o.\forall x.\neg\ (R\ x\ x))$.

Axiom 5.3. $\mathsf{symmetric\_i} = (\lambda R : \iota\iota o.\forall xy.R\ x\ y \to R\ y\ x)$.

Axiom 5.4. $\mathsf{antisymmetric\_i} = (\lambda R : \iota\iota o.\forall xy.R\ x\ y \to R\ y\ x \to x = y)$.

Axiom 5.5. $\mathsf{transitive\_i} = (\lambda R : \iota\iota o.\forall xyz.R\ x\ y \to R\ y\ z \to R\ x\ z)$.

Axiom 5.6. $\mathsf{eqreln\_i} = (\lambda R : \iota\iota o.\mathsf{reflexive\_i}\ R \wedge \mathsf{symmetric\_i}\ R \wedge \mathsf{transitive\_i}\ R)$.

Axiom 5.7. $\mathsf{per\_i} = (\lambda R : \iota\iota o.\mathsf{symmetric\_i}\ R \wedge \mathsf{transitive\_i}\ R)$.

Axiom 5.8. $\mathsf{linear\_i} = (\lambda R : \iota\iota o.\forall xy.R\ x\ y \vee R\ y\ x)$.

Axiom 5.9. $\mathsf{trichotomous\_or\_i} = (\lambda R : \iota\iota o.\forall xy.R\ x\ y \vee x = y \vee R\ y\ x)$.

Axiom 5.10.

$$\mathsf{partialorder\_i} = (\lambda R : \iota\iota o.\mathsf{reflexive\_i}\ R \wedge \mathsf{antisymmetric\_i}\ R \wedge \mathsf{transitive\_i}\ R).$$

Axiom 5.11. $\mathsf{totalorder\_i} = (\lambda R : \iota\iota o.\mathsf{partialorder\_i}\ R \wedge \mathsf{linear\_i}\ R)$.

Axiom 5.12. $\mathsf{strictpartialorder\_i} = (\lambda R : \iota\iota o.\mathsf{irreflexive\_i}\ R \wedge \mathsf{transitive\_i}\ R)$.

Axiom 5.13.

$$\mathsf{stricttotalorder\_i} = (\lambda R : \iota\iota o.\mathsf{strictpartialorder\_i}\ R \wedge \mathsf{trichotomous\_or\_i}\ R).$$

6. Set Operations

We next define a number of new set theoretic operations that can be constructed from the basic ones given in Section 3.

As pointed out by Paulson [22] following Suppes [27] it is possible to define unordered pairs using replacement and a set with (at least) two elements, e.g., $\wp(\wp\emptyset)$. Let UPair be a constant of type $\iota\iota\iota$ with the following definitional axiom:

Axiom 6.1. $\mathsf{UPair} = (\lambda xy.\{\mathsf{If}\ (\emptyset \in z)\ x\ y | z \in \wp(\wp\emptyset)\})$.

We write $\{s, t\}$ as notation for $\mathsf{UPair}\ s\ t$.

It is now trivial to obtain a singleton operation. Let Sing be a constant of type $\iota\iota$ with the following definitional axiom:

Axiom 6.2. $\mathsf{Sing} = (\lambda x.\{x, x\})$.

We write $\{s\}$ as notation for $\mathsf{Sing}\ s$.

In order to be able to generally have notation $\{s_1, \ldots, s_n\}$ for $n > 2$ we need a way to adjoin an element to a set. To obtain this we will include a binary union operation and use this to define the adjoin operation. Let $\mathsf{binunion}$ and $\mathsf{SetAdjoin}$ be constants of type $\iota\iota\iota$. The definining axiom for $\mathsf{binunion}$ is as follows:

Axiom 6.3. $\mathsf{binunion} = (\lambda XY.\bigcup\{X, Y\})$.

We write $s \cup t$ as notation for $\mathsf{binunion}\ s\ t$. The defining axiom for $\mathsf{SetAdjoin}$ is as follows:

Axiom 6.4. $\mathsf{SetAdjoin} = (\lambda Xy.X \cup \{y\})$.

When $n > 2$ we write $\{s_1, \ldots, s_n\}$ for

$$(\mathsf{SetAdjoin} \cdots (\mathsf{SetAdjoin}\ \{s_1, s_2\}\ s_3) \cdots\ s_n).$$

In addition to arbitrary unions given by $\bigcup$ and binary unions given by $\cup$, we have unions of families of sets. Let $\mathsf{famunion}$ be a constant of type $\iota(\iota\iota)\iota$ with the following defininitional axiom:

Axiom 6.5. $\mathsf{famunion} = (\lambda X.\lambda Y : \iota\iota.\bigcup\{Y\ x | x \in X\})$.

We write $\bigcup_{x \in s} t$ as notation for $\mathsf{famunion}\ s\ (\lambda x.t)$.

The constant Repl gives us a way to interpret the notation $\{t | x \in s\}$. A more common notation for sets is $\{x \in s | t\}$, i.e., the set of all members of s satisfying t. This corresponds to Zermelo's Separation Axiom [31]. Let Sep be a term of type $\iota(\iota o)\iota$. We will write $\{x \in s | t\}$ as notation for $\mathsf{Sep}\ s\ (\lambda x.t)$. The definitional axiom for Sep will make use of replacement and make two uses if the If operator. Informally, given a set $X : \iota$ and a property $P : \iota o$ either $\exists x \in X.Px$ holds or it does not. If it does not hold, then $\mathsf{Sep}\ X\ P$ can be $\emptyset$. If it does hold, then $\varepsilon(\lambda x.x \in X \wedge Px)$ yields a "default" element of X satisfying Px. We can then use replacement over X with a function that behaves like the identity for elements satisfying P and returns the default element otherwise. The formal definitional axiom looks as follows:

Axiom 6.6.

$$\mathsf{Sep} = (\lambda X.\lambda P : \iota o.\mathsf{If}\ (\exists x \in X.P\ x)\ \{\mathsf{If}\ (P\ x)\ x\ (\varepsilon\ (\lambda y.y \in X \wedge P\ y)) | x \in X\}\ \emptyset).$$

We can now combine the replacement and separation operator into one operator giving a way to interpret notation of the form $\{t|x \in s_1, s_2\}$ giving the set of all elements of the form t where x (usually free in t) is an element of s_1 and satisfying the property s_2. Formally $\{t|x \in s_1, s_2\}$ is notation for $\mathsf{ReplSep}\ s_1\ (\lambda x.s_2)\ (\lambda x.t)$ where $\mathsf{ReplSep}$ is a constant of type $\iota(\iota o)(\iota\iota)\iota$ with the following definitional axiom:

Axiom 6.7. $\mathsf{ReplSep} = (\lambda X.\lambda P : \iota o.\lambda F : \iota\iota.\{F\ x|x \in \{x \in X|P\ x\}\})$.

In addition to binary unions, we include binary intersections and set difference. Let $\mathsf{binintersect}$ and $\mathsf{setminus}$ be constants of type $\iota\iota\iota$. The definitional axioms make obvious uses of separation.

Axiom 6.8. $\mathsf{binintersect} = (\lambda XY.\{x \in X|x \in Y\})$.

Axiom 6.9. $\mathsf{setminus} = (\lambda XY.\{x \in X|x \notin Y\})$.

We end the section considering (finite) ordinals, giving us a way to interpret natural numbers as sets. Let $\mathsf{ordsucc}$ be a constant of type $\iota\iota$ with the following definitional axiom:

Axiom 6.10. $\mathsf{ordsucc} = (\lambda X.X \cup \{X\})$.

We can now use any natural number as notation for a given term in the obvious way: n is notation for

$$\underbrace{\mathsf{ordsucc}\ (\mathsf{ordsucc} \cdots (\mathsf{ordsucc}}_{n}\ \emptyset) \cdots).$$

This unary representation is sometimes used in pseudorandomly generated Proofgold conjectures, but only for relatively small numbers. For larger numbers a term corresponding to a set with the given cardinality is used. This representation will be described in Section 9.

Let $\mathsf{nat\_p}$ be a constant of type ιo. The definitional axiom for $\mathsf{nat\_p}$ specifies that $\mathsf{nat\_p}$ is the least predicate including 0 and closed under $\mathsf{ordsucc}$.

Axiom 6.11. $\mathsf{nat\_p} = (\lambda x.\forall p : \iota o.p\ 0 \to (\forall n.p\ n \to p\ (\mathsf{ordsucc}\ n)) \to p\ x)$.

In general an ordinal is a set that is well-ordered by $\in$. An easy (classical) way to characterize ordinals is as transitive sets whose elements are all transitive. Let $\mathsf{TransSet}$ and $\mathsf{ordinal}$ be constants of type ιo with the following definitional axioms:

Axiom 6.12. $\mathsf{TransSet} = (\lambda U.\forall X \in U.X \subseteq U)$.

Axiom 6.13. $\mathsf{ordinal} = (\lambda X.\mathsf{TransSet}\ X \land \forall x \in X.\mathsf{TransSet}\ x)$.

Since HF only contains hereditarily finite sets, all the ordinals in this theory are natural numbers. This would still need to be formally proven within Proofgold. Once it has proven, extensionality principles can strengthen the result to obtain $\mathsf{nat\_p} = \mathsf{ordinal}$. This will have the effect of allowing people to interchange occurrences of $\mathsf{nat\_p}$ and $\mathsf{ordinal}$ in the pseudorandomly generated propositions.

7. Recursion

In this section we give a $\in$-recursion operator and show several applications of the operator. More information about the construction can be found in [4] and some discussion of its use is in [7].

Let $\mathsf{In\_rec}$ be a constant of type $(\iota(\iota\iota)\iota)\iota\iota$. Our goes is to give a definitional axiom for $\mathsf{In\_rec}$ so that the identity $\mathsf{In\_rec}\ F\ X = F\ X\ (\mathsf{In\_rec}\ F\ X)$ will follow from a condition on F (that $F\ X\ g$ only depends on the value of g on members of X). The definition of $\mathsf{In\_rec}$ will make use of a separate constant describing the graph of the function. Let $\mathsf{In\_rec\_G}$ be a constant of type $(\iota(\iota\iota)\iota)\iota\iota o$. The definitional axiom for $\mathsf{In\_rec\_G}$ states that $\mathsf{In\_rec\_G}$ is the least relation satisfying the appropriate closure condition corresponding to the desired identity above.

Axiom 7.1.

$$\begin{array}{c}\mathsf{In\_rec\_G} = (\lambda F : \iota(\iota\iota)\iota.\lambda XY.\forall R : \iota\iota o.\\ (\forall Z.\forall f : \iota\iota.(\forall z \in Z.R\ z\ (f\ z)) \to R\ Z\ (F\ Z\ f))\\ \to R\ X\ Y).\end{array}$$

We can now give the definitional axiom for $\mathsf{In\_rec}$ simply by using ε to lift $\mathsf{In\_rec\_G}$ from being a relation to being a function.

Axiom 7.2. $\mathsf{In\_rec} = (\lambda F : \iota(\iota\iota)\iota.\lambda X.\varepsilon\ (\lambda Y.\mathsf{In\_rec\_G}\ F\ X\ Y))$.

We next use the $\in$-recursion operator to define a more specific primitive recursion operator on finite ordinals. Let $\mathsf{nat\_primrec}$ be a constant of type $\iota(\iota\iota\iota)\iota\iota$ with the following definitional axiom:

Axiom 7.3.

$$\begin{array}{c}\mathsf{nat\_primrec} = (\lambda n.\lambda g : \iota\iota\iota.\\ \mathsf{In\_rec}\ (\lambda X.\lambda f : \iota\iota.\mathsf{If}\ ((\bigcup X)\ \in\ X)\ (g\ (\bigcup X)\ (f\ (\bigcup X)))\ n))\end{array}.$$

To better understand this axiom, suppose X is a finite ordinal. If X is 0, then obviously $\bigcup X \notin X$. If X is $\mathsf{ordsucc}\ Y$ for a natural number Y, then $\bigcup X$ is Y (the predecessor of X).

Using this primitive recursion operator we can define addition and multiplication on the natural numbers. Let $\mathsf{add\_nat}$ and $\mathsf{mul\_nat}$ be constants of type $\iota\iota\iota$ with the following definitional axioms:

Axiom 7.4. $\mathsf{add\_nat} = (\lambda mn.\mathsf{nat\_primrec}\ m\ (\lambda kr.\mathsf{ordsucc}\ r)\ n)$.

Axiom 7.5. $\mathsf{mul\_nat} = (\lambda mn.\mathsf{nat\_primrec}\ 0\ (\lambda kr.\mathsf{add\_nat}\ m\ r)\ n)$.

We can also use the $\in$-recursion operator to define the von Neumann hierarchy (as in [7]). Let $\mathsf{V\_}$ be a constant of type $\iota\iota$ with the following definitional axiom:

Axiom 7.6. $\mathsf{V\_} = (\mathsf{In\_rec}\ (\lambda X.\lambda Y : \iota\iota.\bigcup_{x\in X}\wp(Y\ x)))$.

As a final application of $\in$-recursion, we give ways of tagging and untagging sets. This is similar to material in [4]. Let $\mathsf{Inj1}$, $\mathsf{Inj0}$ and Unj be constants of type $\iota\iota$. We will define $\mathsf{Inj1}$ and $\mathsf{Inj0}$ so that they are injective and always give distinct values. We will define Unj so that it is a one-sided inverse of both $\mathsf{Inj1}$ and $\mathsf{Inj0}$. The idea is to define $\mathsf{Inj1}$ by $\in$-recursion to be

$$\mathsf{Inj1}\ X = \{0\} \cup \{\mathsf{Inj1}\ x | x \in X\}.$$

We can then define $\mathsf{Inj0}$ directly by

$$\mathsf{Inj0}\ X = \{\mathsf{Inj1}\ x | x \in X\}.$$

Intuitively $\mathsf{Inj1}$ adds copies of 0 recursively through the iterative construction of its input. We can "undo" this construction by recursively removing these copies by defining Unj so that

$$\mathsf{Unj}\ X = \{\mathsf{Unj}\ x | x \in X \setminus \{0\}\}.$$

The three definitial axioms are given below.

Axiom 7.7. $\mathsf{Inj1} = (\mathsf{In\_rec}\ (\lambda X.\lambda Y : \iota\iota.\{0\} \cup \{Y\ \ x | x \in X\}))$.

Axiom 7.8. $\mathsf{Inj0} = (\lambda X.\{\mathsf{Inj1}\ x | x \in X\})$.

Axiom 7.9. $\mathsf{Unj} = (\mathsf{In\_rec}\ (\lambda X.\lambda Y : \iota\iota.\{Y\ \ z | z \in X \setminus \{0\}\}))$.

8. Disjoint Unions

Tagged sets can be used to definte disjoint unions (sums) of sets. Given sets X and Y the copies $\{\mathsf{Inj0}\ x | x \in X\}$ and $\{\mathsf{Inj1}\ y | y \in Y\}$ are disjoint. This justifies the definitional axiom below for the constant setsum of type $\iota\iota\iota$.

Axiom 8.1. $\mathsf{setsum} = (\lambda XY.\{\mathsf{Inj0}\ x | x \in X\} \cup \{\mathsf{Inj1}\ y | y \in Y\})$.

We will write $\uplus$ as a left associative infix operator corresponding to applying term setsum.

Let X and Y be sets and f and g be (meta-level) functions (of type $\iota\iota$). We can combine the functions to give a function from $X \uplus Y$ that behaves like f on X and g on Y. Let $\mathsf{combine\_funcs}$ be a constant of type $\iota\iota(\iota\iota)(\iota\iota)\iota\iota$ with the following definitional axiom:

Axiom 8.2. $\mathsf{combine\_funcs} = (\lambda XYfgz.\mathsf{If}\ \ (z = \mathsf{Inj0}\ \ (\mathsf{Unj}\ z))\ \ (f\ \ (\mathsf{Unj}\ z))\ \ (g\ \ (\mathsf{Unj}\ z)))$.

Clearly from $X \uplus Y$ we can recover X and Y, so we can view disjoint unions as an implementation of ordered pairs. (Ordered pairs of classes were represented as disjoint unions by Morse [20].) To support this view of ordered pairs, we give constants for the two projections. Let $\mathsf{proj0}$ and $\mathsf{proj1}$ be constants of type $\iota\iota$ with the following two definitional axioms:

Axiom 8.3. $\mathsf{proj0} = (\lambda Z.\{\mathsf{Unj}\ \ z | z \in Z, \exists x.\mathsf{Inj0}\ \ x = z\})$.

Axiom 8.4. $\mathsf{proj1} = (\lambda Z.\{\mathsf{Unj}\ \ z | z \in Z, \exists y.\mathsf{Inj1}\ \ y = z\})$.

9. Binary Representation of Natural Numbers

In this section we introduce one new constant, binrep of type $\iota\iota\iota$. The purpose of this constant is to provide support for a binary representation of natural numbers (different from their representation as finite ordinals). The definitional axiom for binrep is as follows:

Axiom 9.1. $\mathsf{binrep} = (\lambda XY.X \uplus \wp Y)$.

Let us (temporarily) write $|X|$ for the (finite) cardinality of a set X. It is clear that $|\wp Y|$ is $2^{|Y|}$. Hence $|\mathsf{binrep}\ X\ Y| = |X| + 2^{|Y|}$. The psuedorandomly generated conjectures generated by Proofgold often make use of a binary representation of natural numbers. Let us define this as a function B taking natural numbers to closed terms. The intention is that $B(n)$ is a term that is interpreted as a set with cardinality n.

As a helper function, we will first define $B_i(n)$ as follows:

- $B_i(0) := \emptyset$
- $B_i(1) := \wp(B_0(i))$
- $B_i(2n+1) := \mathsf{binrep}\ (B_{i+1}(n))\ (B_0(i))$ for $n > 0$.
- $B_i(2n) := B_{i+1}(n)$ for $n > 0$.

We then define $B(n) := B_0(n)$.

We leave it to the reader to check that $B(n)$ corresponds to a set of cardinality n. Many pseudorandomly generated Proofgold conjectures will likely require lemmas allowing one to infer $B(n)$ has cardinality n.

10. Functions, Dependent Sums and Dependent Products

If we consider x to be the ordered pair of x and y, then we have the material to represent functions as sets. We use the Aczel trace representation [2, 29, 18] of functions instead of the more common graph representation. Let lam be a constant of type $\iota(\iota\iota)\iota$ with the following definitional axiom:

Axiom 10.1. $\mathsf{lam} = (\lambda X.\lambda f : \iota\iota.\bigcup_{x\in X} \{x \uplus y | y \in f\ x\})$.

We will use the notation $\lambda x \in s.t$ for the term $\mathsf{lam}\ s\ (\lambda x.t)$.

Note that $\lambda x \in s.t$ also represents the dependent sum $\Sigma x \in s.t$, since it consists of the pairs $x \uplus y$ where $x \in s$ and $y \in t$ (where x may be free in t). Hence if $x \notin \mathcal{F}(t)$, then $\lambda x \in s.t$ corresponds to the Cartesian product of s and t. Instead of simply using notation for this special case, an extra constant is used. Let setprod be a constant of type $\iota\iota\iota$ with the following definitional axiom:

Axiom 10.2. $\mathsf{setprod} = (\lambda XY.(\lambda x \in X.Y))$.

We use $\times$ as a left associative infix operator corresponding to applying term setprod.

Since we have a representation of functions as sets, we will also include a constant for applying a function to an argument. Letap be a constant of type $\iota\iota\iota$ with the following definitional axiom:

Axiom 10.3. $\mathsf{ap} = (\lambda fx.\{\mathsf{proj1}\ z | z \in f, \exists y.z = x \uplus y\})$.

We will in practice omit ap. That is, if s, t have type ι, then we write st to mean $\mathsf{ap}\ s\ t$. This is possible without misinterpretation since st would be ill-typed without considering it notation for some other kind of term.

It is not difficult to prove that a β-law holds when the argument is in the domain. That is,

$$\forall X.\forall f : \iota\iota.\forall x \in X.(\lambda x \in X.fx)\ x = fx.$$

In addition, if the argument is not in the domain, then the application operator returns the empty set.

$$\forall X.\forall f : \iota\iota.\forall x.x \notin X \to (\lambda x \in X.fx)\ x = \emptyset.$$

Applying a set X to 0 or 1 turns out to be the same as applying the functions $\mathsf{proj0}$ or $\mathsf{proj1}$ to X. Due to this coincidence we can define a predicate recognizing disjoint unions (i.e., ordered pairs) making use of application. Let $\mathsf{setsum\_p}$ be a constant of type ιo with the following definitional axiom:

Axiom 10.4. $\mathsf{setsum\_p} = (\lambda Z.(Z\ 0 \uplus Z\ 1) = Z)$.

We already have a constant equip that determines if two sets are equipotent (have the same cardinality). By making use of $\uplus$ and $\times$ we can modify equip to test of two sets X and Y have the same cardinality modulo the cardinality of a third set M. Let $\mathsf{equip\_mod}$ be a constant of type $\iota\iota o$ with the following definitional axiom:

Axiom 10.5.

$$\begin{aligned}\mathsf{equip\_mod} = (\lambda XYM.\exists ZV. \quad & \mathsf{equip}\ (X \uplus Z)\ Y \wedge \mathsf{equip}\ (V \times Z)\ M \\ & \vee \mathsf{equip}\ (Y \uplus Z)\ X \wedge \mathsf{equip}\ (V \times Z)\ M).\end{aligned}$$

For a finite ordinal n, we consider an n-tuple to be a function (encoded as a set) with domain n. Let $\mathsf{tuple\_p}$ be a constant of type $\iota\iota o$ with the following definitional axiom:

Axiom 10.6. $\mathsf{tuple\_p} = (\lambda nZ.\forall z \in Z.(\exists i \in n.\exists x.z = i \uplus x))$.

If n is a finite ordinal and Z is a set, then $\mathsf{tuple\_p}\ n\ Z$ means Z is an n-tuple. For each $i \in n$, $Z\ i$ (i.e., $\mathsf{ap}\ Z\ i$) gives the i^{th} component of the n-tuple. Note that 2-tuples are the same as ordered pairs.

We next represent the dependent set of functions (as sets) determined by a domain set X and a family Y (of type $\iota\iota$) of codomain sets. Let Pi be a constant of type $\iota(\iota\iota)\iota$ with the following definitional axiom:

Axiom 10.7. $\mathsf{Pi} = (\lambda X.\lambda Y : \iota\iota.\{f \in \wp(\lambda x \in X.\bigcup(Y\ x)) | \forall x \in X.f\ x \in Y\ x\})$.

We write $\Pi x \in s.t$ as notation for $\mathsf{Pi}\ s\ (\lambda x.t)$. The special case when $x \notin \mathcal{F}(t)$ yields exponents of sets (via simple function spaces). Let setexp be a constant of type $\iota\iota\iota$ with the following definitional axiom:

Axiom 10.8. $\mathsf{setexp} = (\lambda XY.\Pi x \in Y.X)$.

Before ending the section we introduce three more constants that make it easier to work with sets of pairs and functions taking two arguments.

Let $\mathsf{Sep2}$ be a constant of type $\iota(\iota\iota)(\iota\iota o)\iota$. We can use $\mathsf{Sep2}$ to seperate pairs from a set X and a family Y satisfying a (meta-level) relation R. The definitional axiom for $\mathsf{Sep2}$ follows.

Axiom 10.9. $\mathsf{Sep2} = (\lambda X.\lambda Y : \iota\iota.\lambda R : \iota\iota o.\{z \in (\lambda x \in X.Y\ x)|R\ (z\ 0)\ (z\ 1)\})$.

Let set_of_pairs be a constant of type ιo. This predicate simply recognizes if a set only contains pairs and has the following definitional axiom:

Axiom 10.10. $\mathsf{set\_of\_pairs} = (\lambda X.\forall x \in X.\exists yz.x = (\lambda i \in 2.\mathsf{If}\ (i = 0)\ y\ z))$.

Finally let lam2 be a constant of type $\iota(\iota\iota)(\iota\iota\iota)\iota$. The purpose of lam2 is to give a representation as a set of a meta-level binary function f when its first argument is restricted to a set X and its second argument is restricted to $Y\ x$ (where $x \in X$ is the first argument). The representation is accomplished by Currying and using the lam operator twice as is shown in the following definitional axiom:

Axiom 10.11. $\mathsf{lam2} = (\lambda X.\lambda Y : \iota\iota.\lambda f : \iota\iota\iota.\lambda x \in X.\lambda y \in Y\ x.f\ x\ y)$.

11. Surreal Numbers

In this section we describe a number of constants related to Conway's surreal numbers [12] and their definitional axioms. These will give us the natural numbers yet again, but the surreal version of each natural number n will be the finite ordinal n, so the representation is not new. This will extend the representation to include negative natural numbers (so we have the integers) and the dyadic rational numbers. Since every set in ι is hereditarily finite, we do not obtain, e.g., the real numbers. However, real numbers can be obtained by going higher in the type hierarchy and properly generalizing the definitions at higher types. We will not follow elaborate on this here.

This is not the first formal version of surreal numbers. Mamane [19] formalized surreal numbers in Coq and Obua [21] formalized Conway games and surreal numbers in an extension of Isabelle/HOL called Isabelle/HOLZF.

There are actually two representations of surreal numbers considered. We call these the *external* view and the *internal* view. The external view considers a surreal number to be an ordinal α and a predicate P.[7] The internal view gives a set representation that remembers the α and which members of α satisfy P. As a consequence of these two views there will generally be two constants (of different types) for each concept.

We consider the external view first. Let PNoEq_ be a constant of type $\iota(\iota o)(\iota o)o$. The meaning of PNoEq_ $\alpha\ P\ Q$ is that P and Q agree on α. This is a way of saying α and P specify the same surreal number as α and Q. The definitional axiom follows.

Axiom 11.1. $\mathsf{PNoEq\_} = (\lambda\alpha.\lambda PQ : \iota o.\forall\beta \in \alpha.P\ \beta \leftrightarrow Q\ \beta)$.

Let PNoLt_ be a constant of type $\iota(\iota o)(\iota o)o$. The meaning of PNoLt_ $\alpha\ P\ Q$ is that the surreal number specified α and P is "less than" the surreal number as α and Q. The definitional axiom follows.

Axiom 11.2. $\mathsf{PNoLt\_} = (\lambda\alpha.\lambda PQ : \iota o.\exists\beta \in \alpha.\mathsf{PNoEq\_}\ \beta\ P\ Q \wedge \neg P\ \beta \wedge Q\ \beta)$.

We can now generalize to surreal numbers specified using different ordinals. Let PNoLt and PNoLe be constants of type $\iota(\iota o)\iota(\iota o)o$. The meaning of PNoLt $\alpha\ P\ \beta\ Q$ is

[7] In this section, α and β are not types, but instead are variables of type ι intended to be (finite) ordinals. The same definitions would make sense if the ordinals were not finite.

that the surreal number specified by α and P is "less than" the surreal number specified by β and Q. The meaning of $\mathsf{PNoLe}\ \alpha\ P\ \beta\ Q$ is that the surreal number specified by α and P is "less than or equal to" the surreal number specified by β and Q. The definitional axioms follows:

Axiom 11.3.

$$\begin{aligned}\mathsf{PNoLt} = (\lambda\alpha.\lambda P : \iota o.\lambda\beta.\lambda Q : \iota o.&\mathsf{PNoLt\_}\ (\alpha \cap \beta)\ P\ Q\\ &\vee \alpha \in \beta \wedge \mathsf{PNoEq\_}\ \alpha\ P\ Q \wedge Q\ \alpha\\ &\vee \beta \in \alpha \wedge \mathsf{PNoEq\_}\ \beta\ P\ Q \wedge \neg P\ \beta).\end{aligned}$$

Axiom 11.4.

$$\mathsf{PNoLe} = (\lambda\alpha.\lambda P : \iota o.\lambda\beta.\lambda Q : \iota o.\mathsf{PNoLt}\ \alpha\ P\ \beta\ Q \vee \alpha = \beta \wedge \mathsf{PNoEq\_}\ \alpha\ P\ Q).$$

Conway **??** defines surreal numbers as pairs of sets of surreal numbers (those on the left and those on the right) with an ordering condition ensuring those on the left are less than those on the right. With our external view we can consider a collection of surreal numbers to be a value L of type $\iota(\iota o)o$ where $L\ \alpha\ P$ implies α and P represents a surreal number. In order to relate to Conway's definition it is useful to be able to take such collections and close them downwards (for the left) or upwards (for the right). For this purpose let $\mathsf{PNo\_downc}$ and $\mathsf{PNo\_upc}$ be constants of type $(\iota(\iota o)o)\iota(\iota o)o$ with the following definitional axioms:

Axiom 11.5.

$$\mathsf{PNo\_downc} = (\lambda L : \iota(\iota o)o.\lambda\alpha.\lambda p : \iota o.\exists\beta.\mathsf{ordinal}\ \beta \wedge \exists q : \iota o.L\ \beta\ q \wedge \mathsf{PNoLe}\ \alpha\ p\ \beta\ q).$$

Axiom 11.6.

$$\mathsf{PNo\_upc} = (\lambda R : \iota(\iota o)o.\lambda\alpha.\lambda p : \iota o.\exists\beta.\mathsf{ordinal}\ \beta \wedge \exists q : \iota o.R\ \beta\ q \wedge \mathsf{PNoLe}\ \beta\ q\ \alpha\ p).$$

We now pass to the internal view. Let us write β' for the set $\mathsf{SetAdjoin}\ \beta\ \{1\}$. Note that no ordinal contains $\{1\}$ as an element since $\{1\}$ is not transitive. Hence if β is an ordinal, then $\beta' \neq \beta$ and $\beta = \beta' \setminus \{\{1\}\}$. If α and P give the external view of a surreal number, then we include β in the internal view to record $\beta \in \alpha$ where $P\ \beta$ holds and include β' to record values $\beta \in \alpha$ where $P\ \beta$ does *not* hold. It is easy to see that if $P\ \beta$ holds for all $\beta \in \alpha$, then α itself will provide the internal view of the surreal number.

Let $\mathsf{SNoElts\_}$ be a constant of type $\iota\iota$ with the following definitional axiom:

Axiom 11.7. $\mathsf{SNoElts\_} = (\lambda\alpha.\alpha \cup \{\beta' | \beta \in \alpha\})$.

The purpose of $\mathsf{SNoElts\_}\ \alpha$ is to give a bounding set from which all elements of the internal view of the surreal number specified by α and P.

Let $\mathsf{SNo\_}$ be a constant of type $\iota\iota o$. The meaning of $\mathsf{SNo\_}\ \alpha$ (when α is an ordinal) is the set of all (internal views of) surreal numbers specified by α and P for some P. The definitional axiom follows.

Axiom 11.8. $\mathsf{SNo\_} = (\lambda\alpha x.x \subseteq \mathsf{SNoElts\_}\ \alpha \wedge \forall\beta \in \alpha.\mathsf{exactly1of2}\ (\beta' \in x)\ (\beta \in x))$.

Let PSNo be a constant of type $\iota(\iota o)\iota$ that is intended to coerce from the external view to the internal view. The definitional axiom follows.

Axiom 11.9. $\mathsf{PSNo} = (\lambda\alpha.\lambda p : \iota o.\{\beta \in \alpha | p\ \beta\} \cup \{\beta' | \beta \in \alpha, \neg p\ \beta\})$.

Let SNo be a constant of type ιo. The proposition $\mathsf{SNo}\ x$ should hold precisely when x is (the internal view of) a surreal number (in HF).

Axiom 11.10. $\mathsf{SNo} = (\lambda x.\exists\alpha.\mathsf{ordinal}\ \alpha \wedge \mathsf{SNo\_}\ \ \alpha\ x)$.

Each surreal number can be said to have a *level* (or *birthday*). This is obvious in the external view: if α and P specify a surreal number, then α is its level. In the internal view we can recover the level using the ε operator. (This could alternatively be done by collecting the ordinals β that either occur in the form β or in the modified form β'.) Let SNoLev be a constant of type $\iota\iota$ with the following definitional axiom:

Axiom 11.11. $\mathsf{SNoLev} = (\lambda x.\varepsilon\ (\lambda\alpha.\mathsf{ordinal}\ \alpha \wedge \mathsf{SNo\_}\ \ \alpha\ x))$.

Note that from the internal view x we can now recover the external view by taking α to be $\mathsf{SNoLev}\ x$ and taking P to be $\lambda\beta.\beta \in x$. We make use of this fact to internalize the relations PNoEq_, PNoLt and PNoLe. Let SNoEq_ be a constant of type $\iota\iota\iota o$ with the following definitional axiom:

Axiom 11.12. $\mathsf{SNoEq\_} = (\lambda\alpha x y.\mathsf{PNoEq\_}\ \ \alpha\ (\lambda\beta.\beta \in x)\ (\lambda\beta.\beta \in y))$.

Let SNoLt and SNoLe be constants of type $\iota\iota o$ with the following definitional axioms:

Axiom 11.13. $\mathsf{SNoLt} = (\lambda xy.\mathsf{PNoLt}\ (\mathsf{SNoLev}\ x)\ (\lambda\beta.\beta \in x)\ (\mathsf{SNoLev}\ y)\ (\lambda\beta.\beta \in y))$.

Axiom 11.14. $\mathsf{SNoLe} = (\lambda xy.\mathsf{PNoLe}\ (\mathsf{SNoLev}\ x)\ (\lambda\beta.\beta \in x)\ (\mathsf{SNoLev}\ y)\ (\lambda\beta.\beta \in y))$.

12. Loops and Inner Mappings

We next consider a few constants and definitional axioms about loops with a focus on The AIM Conjecture [17, 8].[8] To start with, let binop_on be a constant of type $\iota(\iota\iota\iota)o$. where $\mathsf{binop\_on}\ X\ f$ means f is a binary operation on X. The definitional axiom follows.

Axiom 12.1. $\mathsf{binop\_on} = (\lambda X.\lambda f : \iota\iota\iota.\forall xy \in X.fxy \in X)$.

Next let Loop be a constant of type $\iota(\iota\iota\iota)(\iota\iota\iota)(\iota\iota\iota)\iota o$. Here $\mathsf{Loop}\ X\ m\ b\ s\ e$ will mean that X is a loop with binary operations m (multiplication), b (left division) and s (right division) and identity element e. The definitional axiom follows.

Axiom 12.2.

$$\begin{gathered}\mathsf{Loop} = \\ (\lambda X.\lambda mbs : \iota\iota\iota.\lambda e. \\ \mathsf{binop\_on}\ X\ m \wedge \mathsf{binop\_on}\ X\ b \wedge \mathsf{binop\_on}\ X\ s \\ \wedge(\forall x \in X.(m\ e\ x = x \wedge m\ x\ e = x)) \\ \wedge(\forall xy \in X.(b\ x\ (m\ x\ y) = y \wedge m\ x\ (b\ x\ y) = y \\ \wedge s\ (m\ x\ y)\ y = x \wedge m\ (s\ x\ y)\ y = x))).\end{gathered}$$

[8]The AIM Conjecture is an open mathematics problem as of 2020.

We will next consider an extension of Loop that includes reference to several definable functions. Most of these definable functions are families of inner mappings. The exceptions are an associator function a and a commutator function K. Before giving the constant and its definitional axiom we give an informal description of each of the new functions.

Suppose Loop $X\ m\ b\ s\ e$ holds. Below let x, y, z, w, u, v range over elements of X. Following [17] let us write $(x \cdot y)$ for $m\ x\ y$, $(x\backslash y)$ for $b\ x\ y$ and (x/y) for $s\ x\ y$.

The commutator $K\ x\ y$ of x and y is $(y{\cdot}x)\backslash(x{\cdot}y)$. It is easy to see that $(y{\cdot}x){\cdot}(K\ x\ y) = x \cdot y$ and in particular $K\ x\ y = e$ if and only if $y \cdot x = x \cdot y$. The associator $a\ x\ y\ z$ of x, y and z is $(x{\cdot}(y{\cdot}z))\backslash((x{\cdot}y){\cdot}z)$. As with the commutator case $(x{\cdot}(y{\cdot}z)){\cdot}(a\ x\ y\ z) = (x{\cdot}y){\cdot}z$ and $a\ x\ y\ z = e$ if and only if $x \cdot (y \cdot z) = (x \cdot y) \cdot z$.

The remaining functions we consider will correspond to inner mappings. We first describe the three families of inner mappings T, L and R that are central to the discussion in [17]. These families of inner mappings are sufficient to generate the set of all inner mappings. For each $x \in X$, T_x is the inner mapping taking u to $x\backslash(u \cdot x)$. For each $x, y \in X$, $L_{x,y}$ is the inner mapping taking u to $(y \cdot x)\backslash(y \cdot (x \cdot u))$ and $R_{x,y}$ is the inner mapping taking u to $((u \cdot x) \cdot y)/(x \cdot y)$. For $x \in X$ there are four inner mappings I^1_x, J^1_x, I^2_x and J^2_x specified as follows:

- $I^1_x u = x \cdot (u \cdot (x\backslash e))$.
- $J^1_x u = ((e/x) \cdot u) \cdot x$.
- $I^2_x u = (x\backslash u) \cdot ((x\backslash e)\backslash e)$.
- $J^2_x u = (e/(e/x)) \cdot (u/x)$.

In the definitional axioms below we will write $T\ x\ u$ for $T_x u$, $L\ x\ y\ u$ for $L_{x,y} u$, $R\ x\ y\ u$ for $R_{x,y} u$, $I^1\ x\ u$ for $I^1_x u$, $J^1\ x\ u$ for $J^1_x u$, $I^2\ x\ u$ for $I^2_x u$ and $J^2\ x\ u$ for $J^2_x u$.

Let Loop_with_defs be a constant of type

$$\iota(\iota\iota\iota)(\iota\iota\iota)(\iota\iota\iota)\iota(\iota\iota\iota)(\iota\iota\iota\iota) \to (\iota\iota\iota)(\iota\iota\iota\iota) \to (\iota\iota\iota\iota) \to (\iota\iota\iota)(\iota\iota\iota)(\iota\iota\iota)(\iota\iota\iota)o.$$

The meaning of Loop_with_defs $X\ m\ b\ s\ e\ K\ a\ T\ L\ R\ I^1\ J^1\ I^2\ J^2$ is that Loop $X\ m\ b\ s\ e$ holds and that K, a and T, L, R, I^1, J^1, I^2 and J^2 satisfy the equations given above.
Constant. The name

Axiom 12.3.

$$\begin{gathered}
\mathsf{Loop_with_defs} = \\
(\lambda X.\lambda mbs : \iota\iota\iota.\lambda e.\lambda K : \iota\iota\iota.\lambda a : \iota\iota\iota\iota.\lambda T : \iota\iota\iota.\lambda LR : \iota\iota\iota\iota.\lambda I^1 J^1 I^2 J^2 : \iota\iota\iota. \\
\mathsf{Loop}\ X\ m\ b\ s\ e \\
\wedge(\forall xy \in X.K\ x\ y = b\ (m\ y\ x)\ (m\ x\ y)) \\
\wedge(\forall xyz \in X.a\ x\ y\ z = b\ (m\ x\ (m\ y\ z))\ (m\ (m\ x\ y)\ z)) \\
\wedge(\forall xu \in X.T\ x\ u = b\ x\ (m\ u\ x) \\
\wedge I^1\ x\ u = m\ x\ (m\ u\ (b\ x\ e)) \\
\wedge J^1\ x\ u = m\ (m\ (s\ e\ x)\ u)\ x \\
\wedge I^2\ x\ u = m\ (b\ x\ u)\ (b\ (b\ x\ e)\ e) \\
\wedge J^2\ x\ u = m\ (s\ e\ (s\ e\ x))\ (s\ u\ x)) \\
\wedge(\forall xyu \in X.L\ x\ y\ u = b\ (m\ y\ x)\ (m\ y\ (m\ x\ u)) \\
\wedge R\ x\ y\ u = s\ (m\ (m\ u\ x)\ y)\ (m\ x\ y))).
\end{gathered}$$

Suppose $\mathsf{Loop\_with\_defs}\ X\ m\ b\ s\ e\ K\ a\ T\ L\ R\ I^1\ J^1\ I^2\ J^2$ holds. As described in [17] we know a loop is AIM (i.e., the inner mappings form an abelian group) if the following six equations hold:

- $T_x(T_y u) = T_y(T_x u)$ for $x, y, u \in X$.
- $T_x(L_{y,z} u) = L_{y,z}(T_x u)$ for $x, y, z, u \in X$.
- $T_x(R_{y,z} u) = R_{y,z}(T_x u)$ for $x, y, z, u \in X$.
- $L_{x,y}(L_{z,w} u) = L_{z,w}(L_{x,y} u)$ for $x, y, z, w, u \in X$.
- $L_{x,y}(R_{z,w} u) = R_{z,w}(L_{x,y} u)$ for $x, y, z, w, u \in X$.
- $R_{x,y}(R_{z,w} u) = R_{z,w}(R_{x,y} u)$ for $x, y, z, w, u \in X$.

We claim that The AIM Conjecture holds if the following two identities hold in every AIM loop (where x, y, z, w, u range over elements of the loop):[9]

(1) $K\ ((L_{x,y} u \backslash e) \cdot u)\ w = e$
(2) $a\ w\ ((e/u) \cdot R_{x,y} u)\ z = e$

Assuming this is correct, then there are two possible kinds of counterexamples to The AIM Conjecture. The first kind of counterexample contains a violation of the first identity and the second kind of counterexample contains a violation of the second identity. This is the motivation for the last two constants and definitional axioms of this section.

Let $\mathsf{Loop\_with\_defs\_cex1}$ and $\mathsf{Loop\_with\_defs\_cex2}$ be constants of type

$$\iota(\iota\iota\iota)(\iota\iota\iota)(\iota\iota\iota)\iota(\iota\iota\iota)(\iota\iota\iota\iota)(\iota\iota\iota)(\iota\iota\iota\iota)(\iota\iota\iota\iota)(\iota\iota\iota)(\iota\iota\iota)(\iota\iota\iota)(\iota\iota\iota)o$$

with the following two definitional axioms:

Axiom 12.4.

$$\begin{gathered}\mathsf{Loop\_with\_defs\_cex1} = \\ (\lambda X.\lambda mbs : \iota\iota\iota.\lambda e.\lambda K : \iota\iota\iota.\lambda a : \iota\iota\iota\iota.\lambda T : \iota\iota\iota.\lambda LR : \iota\iota\iota\iota.\lambda I^1 J^1 I^2 J^2 : \iota\iota\iota. \\ \mathsf{Loop\_with\_defs}\ X\ m\ b\ s\ e\ K\ a\ T\ L\ R\ I^1\ J^1\ I^2\ J^2 \\ \wedge \exists uxyw \in X. \neg (K\ (m\ (b\ (L\ x\ y\ u)\ e)\ u)\ w = e)).\end{gathered}$$

Axiom 12.5.

$$\begin{gathered}\mathsf{Loop\_with\_defs\_cex2} = \\ (\lambda X.\lambda mbs : \iota\iota\iota.\lambda e.\lambda K : \iota\iota\iota.\lambda a : \iota\iota\iota\iota.\lambda T : \iota\iota\iota.\lambda LR : \iota\iota\iota\iota.\lambda I^1 J^1 I^2 J^2 : \iota\iota\iota. \\ \mathsf{Loop\_with\_defs}\ X\ m\ b\ s\ e\ K\ a\ T\ L\ R\ I^1\ J^1\ I^2\ J^2 \\ \wedge \exists uxyzw \in X. \neg (a\ w\ (m\ (s\ e\ u)\ (R\ x\ y\ u))\ z = e)).\end{gathered}$$

The meaning of $\mathsf{Loop\_with\_defs\_cex1}\ X\ m\ b\ s\ e\ K\ a\ T\ L\ R\ I^1\ J^1\ I^2\ J^2$ is that $\mathsf{Loop\_with\_defs}\ X\ m\ b\ s\ e\ K\ a\ T\ L\ R\ I^1\ J^1\ I^2\ J^2$ holds and the loop has a counterexample of the first kind. Likewise $\mathsf{Loop\_with\_defs\_cex2}\ X\ m\ b\ s\ e\ K\ a\ T\ L\ R\ I^1\ J^1\ I^2\ J^2$ means $\mathsf{Loop\_with\_defs}\ X\ m\ b\ s\ e\ K\ a\ T\ L\ R\ I^1\ J^1\ I^2\ J^2$ holds and the loop has a counterexample of the second kind. Such loops do exist. The only way the counterexample would actually be a counterexample to The AIM Conjecture is if the loop were AIM. We can now state The AIM Conjecture (for finite loops) as two conjectures in

[9] A justification for this is beyond the scope of this work.

the HF theory:

$$\begin{array}{c}\forall X.\forall mbs:\iota\iota.\forall e.\forall K:\iota\iota.\forall a:\iota\iota\iota.\forall T:\iota\iota.\forall LR:\iota\iota\iota.\forall I^1J^1I^2J^2:\iota\iota.\\ \textsf{Loop\_with\_defs\_cex1}\ X\ m\ b\ s\ e\ K\ a\ T\ L\ R\ I^1\ J^1\ I^2\ J^2\\ \to(\forall xyu\in X.T\ x\ (T\ y\ u)=T\ y\ (T\ x\ u))\\ \to(\forall xyzu\in X.T\ x\ (L\ y\ z\ u)=L\ y\ z\ (T\ x\ u))\\ \to(\forall xyzu\in X.T\ x\ (R\ y\ z\ u)=R\ y\ z\ (T\ x\ u))\\ \to(\forall xyzwu\in X.L\ x\ y\ (L\ z\ w\ u)=L\ z\ w\ (L\ x\ y\ u))\\ \to(\forall xyzwu\in X.L\ x\ y\ (R\ z\ w\ u)=R\ z\ w\ (L\ x\ y\ u))\\ \to(\forall xyzwu\in X.R\ x\ y\ (R\ z\ w\ u)=R\ z\ w\ (R\ x\ y\ u))\\ \to\bot\end{array}$$

and

$$\begin{array}{c}\forall X.\forall mbs:\iota\iota.\forall e.\forall K:\iota\iota.\forall a:\iota\iota\iota.\forall T:\iota\iota.\forall LR:\iota\iota\iota.\forall I^1J^1I^2J^2:\iota\iota.\\ \textsf{Loop\_with\_defs\_cex2}\ X\ m\ b\ s\ e\ K\ a\ T\ L\ R\ I^1\ J^1\ I^2\ J^2\\ \to(\forall xyu\in X.T\ x\ (T\ y\ u)=T\ y\ (T\ x\ u))\\ \to(\forall xyzu\in X.T\ x\ (L\ y\ z\ u)=L\ y\ z\ (T\ x\ u))\\ \to(\forall xyzu\in X.T\ x\ (R\ y\ z\ u)=R\ y\ z\ (T\ x\ u))\\ \to(\forall xyzwu\in X.L\ x\ y\ (L\ z\ w\ u)=L\ z\ w\ (L\ x\ y\ u))\\ \to(\forall xyzwu\in X.L\ x\ y\ (R\ z\ w\ u)=R\ z\ w\ (L\ x\ y\ u))\\ \to(\forall xyzwu\in X.R\ x\ y\ (R\ z\ w\ u)=R\ z\ w\ (R\ x\ y\ u))\\ \to\bot.\end{array}$$

If both of these sentences are provable in HF, then The AIM Conjecture holds for all finite loops.

13. Combinators

The two constants and definitional axioms have to do with untyped combinatory logic. This is a particularly simple language that turns out to provide a Turing complete programming language.

Let combinator be a constant of type ιo and let combinator_equiv be a constant of type $\iota\iota o$. The meaning of combinator Z is that Z is a combinator, where combinators are formed from two basic combinators K and S using a (syntactic) binary application operation. The meaning of combinator_equiv Y Z is that Y and Z are equivalent as combinators, up to combinatory conversion.

We will represent the two combinators K and S by specific distinct sets. Let K be Inj0 $\emptyset$. Let S be Inj0 $(\wp\emptyset)$. We will next represent the (syntactic) combinatory logic application operation by pairing the function and argument (using $\uplus$) and ensuring we do not obtain K or S by using Inj1. Let Ap be the term

$$\lambda YZ.\textsf{Inj1}\ (Y\uplus Z)$$

of type $\iota\iota\iota$.

The definitional axiom for combinator ensures combinator is the least predicate containing K and S and closed under Ap.

Axiom 13.1.

$$\textsf{combinator}=(\lambda X.\forall p:\iota o.p\ K\to p\ S\to(\forall YZ.p\ Y\to p\ Z\to p\ (Ap\ Y\ Z))\to p\ X).$$

The definitional axiom for $\mathsf{combinator\_equiv}$ ensures $\mathsf{combinator\_equiv}$ is the least congruence relation on $\mathsf{combinator}$ such that $Ap\ (Ap\ K\ W)\ Z$ is equivalent to W and $Ap\ (Ap\ (Ap\ S\ W)\ Z)\ V$ is equivalent to $Ap\ (Ap\ W\ V)\ (Z\ V)$.

Axiom 13.2.

$$\begin{gathered}
\mathsf{combinator_equiv} = (\lambda XY.\forall r : \iota\iota o.\mathsf{per_i}\ r \\
\to (\forall Z.\mathsf{combinator}\ Z \to r\ Z\ Z) \\
\to (\forall W_1 Z_1 W_2 Z_2.\mathsf{combinator}\ W_1 \to \mathsf{combinator}\ Z_1 \\
\to \mathsf{combinator}\ W_2 \to \mathsf{combinator}\ Z_2 \\
\to r\ W_1\ W_2 \to r\ Z_1\ Z_2 \\
\to r\ (Ap\ W_1\ Z_1)\ (Ap\ W_2\ Z_2)) \\
\to (\forall WZ.r\ (Ap\ (Ap\ K\ W)\ Z)\ W) \\
\to (\forall WZV.r\ (Ap\ (Ap\ (Ap\ S\ W)\ Z)\ V)\ (Ap\ (Ap\ W\ V)\ (Ap\ Z\ V))) \\
\to r\ X\ Y).
\end{gathered}$$

14. Conjectures as Part of Proofgold's Consensus Algorithm

The hash of each Proofgold block is included in a Litecoin transaction using the script command `OP_RETURN`. With the exception of the genesis block, the transaction id of the Litecoin transaction recording the previous Proofgold block is also included in this `OP_RETURN`. This makes it easy to scan the Litecoin blockchain to determine an outline of the Proofgold blockchain. When the Litecoin transaction is included in a Litecoin block, the transaction id is hashed together the the Litecoin block id to provide 256 bits of information to use to generate the conjecture on which a bounty must be placed using half of the block reward of the next Proofgold block. Each conjecture is interpreted in the context of the HF theory we have just described.

The conjectures fall into one of several classes and we give a brief description of each class below.[10] Using the 256 bits Proofgold decides to attempt to make a conjecture of one of the classes. Before starting the generation of the proposition within the class, hashing is used to expand the 256 bits to 2048 bits. If less than 10 bytes are used in the generation, it is considered a failure. If the generation process tries to use more than 2048 bits, it is also a failure. In the case of failure Proofgold falls back on the last class (Diophantine style problems) which cannot fail by design.

14.1. **Random.** Conjectures in this class are generally not meaningful, but the choices made during the generation are also not uniformly random. The conjecture must start with at least two (possibly bounded) quantifiers. When a term of type ι must be generated and a bound variable is not being chosen, then half the time the binary representation of a number between 5 and 20 is used, a quarter of the time the unary representation of a number between 5 and 20 is used. In the remaining quarter of the cases, half the time a unary function is chosen (leaving the argument to be generated), a quarter of the term a binary function is chosen (leaving two arguments to be generated) and the remaining quarter some other set former is used (e.g., Sep). In case the

[10] A description can also be found at `https://proofgold.org/rewardbounties.html` which provided a starting point for the description here. More details are in the Proofgold source file `checking.ml`.

generation seems to be running out of bits of information, then it restricts the choices available.

There are three subclasses of random conjectures. The first kind is simply a sentence constructed as roughly described above. The second kind is of the form $\forall p : \iota o.\forall f : \iota\iota.s$ where s is generated as above but is allowed to use the (uninterpreted) unary predicate p and unary function f. The third kind is of the form

$$\forall xyz.\forall f : \iota\iota.\forall pq : \iota o.\forall g : \iota\iota\iota.\forall r : \iota\iota o.s$$

where s is a generated as above though it is allowed to use x, y, z, f, g to construct sets, to use p, q, r to construct atomic propositions and is (mostly) disallowed from using the constants from the HF set theory.

14.2. **Quantified Boolean Formulas (QBF).** Conjectures in the QBF class are of the form

$$Q_1 p_1 : o. \cdots .Q_n p_n : o.s \leftrightarrow t$$

where $50 \leq n \leq 55$, each Q_i is $\forall$ or $\exists$ and s and t are propostions such that $\mathcal{F}(s) = \mathcal{F}(t) = \{p_1, \ldots, p_n\}$. The propositions s and t are generated using the same process (but different bits for making choices, of course). We describe the process below.

At each stage there is a set V of variables that need to occur free. Initially $V = \{p_1, \ldots, p_n\}$. If V has more than 4 variables, then it is randomly split into two sets V_1 and V_2 such that $V = V_1 \cup V_2$. It is not required that V_1 and V_2 are disjoint, but $V_1 \cap V_2$ may not have more than 3 variables. Now assume s_1 is generated for V_1 and s_2 is V_2. Either $s_1 \to s_2$, $\neg(s_2 \to s_1)$, or $s_1 \leftrightarrow s_2$ is generated to cover V.

Assume V has at most 4 variables, e.g., $q_1, \ldots, q_k$ with $k \leq 4$. In this case the formula generated is $L_1 \to \cdots \to L_k \to \bot$ where L_i is either q_i or $\neg q_i$. This is essentially a clause with k literals.

14.3. **Set Constraints.** One of the most challenging aspects of higher-order theorem proving is instantiating set variables, i.e., variables of a type like ιo [3]. The only known complete procedure requires enumeration of $\beta\eta$-normal terms of this type.

In order to describe the types involved in the set constraint conjectures as well as the higher-order unification conjectures we introduce some new terminology. We say a type is *ι-pure* if there are no occurrences of o. We say a type is *ι-relational* if it has the form it has the form $\alpha_1 \cdots \alpha_n o$ where each α_i is ι-pure. Next note that we can view such a type as a binary tree with the implicit function type arrow as the nodes. We say a type has *minimum depth n* if in this tree view every branch has at least depth n and say a type has *maximum depth n* if every branch has at most depth n.

Let V be a finite set of variables. For $P \in V$ of ι-relational type, we call a term a *V-atom with head P* if it has the form $P\ s_1\ \cdots\ s_n$ where each s_i has the appropriate type and $\mathcal{F}(s_i) \subseteq V$. A *flexible V-atom* is a V-atom with head P for some $P \in V$ of ι-relational type. A *rigid V-atom* is a term of the form $s_1 \in s_2$ where $\mathcal{F}(s_1) \subseteq V$ and $\mathcal{F}(s_2) \subseteq V$. (That is, the only rigid relation considered is the constant $\in$ from the HF theory.)

Let P be a variable of an ι-relational type $\beta_1 \cdots \beta_n o$ and V be a set of variables with $P \in V$. A *lower bound constraint for P over V* is of one of the forms (where z_1, z_2, z_3, z_4

are variables not in V)

$$\forall z_1 : \gamma_1.\forall z_2 : \gamma_2.\forall z_3 : \gamma_3.\forall z_4 : \gamma_4.\varphi$$

or

$$\forall z_1 : \gamma_1.\forall z_2 : \gamma_2.\forall z_3 : \gamma_3.\forall z_4 : \gamma_4.\psi \to \varphi$$

or

$$\forall z_1 : \gamma_1.\forall z_2 : \gamma_2.\forall z_3 : \gamma_3.\forall z_4 : \gamma_4.\psi \to \zeta \to \varphi$$

or

$$\forall z_1 : \gamma_1.\forall z_2 : \gamma_2.\forall z_3 : \gamma_3.\forall z_4 : \gamma_4.\zeta \to \varphi$$

where each γ_j is an ι-pure type with minimum depth 0 and maximum depth 4, φ is a $V \cup \{z_1, z_2, z_3, z_4\}$-atom with head P, ψ (if relevant) is a rigid $V \cup \{z_1, z_2, z_3, z_4\}$-atom. and ζ (if relevant) is a flexible $V \cup \{z_1, z_2, z_3, z_4\}$-atom.

The notion of upper bound constrain is dual, but we include it explicitly for clarity. An *upper bound constraint for P over V* is

$$\forall z_1 : \gamma_1.\forall z_2 : \gamma_2.\forall z_3 : \gamma_3.\forall z_4 : \gamma_4.\varphi \to \bot$$

or

$$\forall z_1 : \gamma_1.\forall z_2 : \gamma_2.\forall z_3 : \gamma_3.\forall z_4 : \gamma_4.\varphi \to \psi$$

or

$$\forall z_1 : \gamma_1.\forall z_2 : \gamma_2.\forall z_3 : \gamma_3.\forall z_4 : \gamma_4.\psi \to \varphi \to \zeta$$

or

$$\forall z_1 : \gamma_1.\forall z_2 : \gamma_2.\forall z_3 : \gamma_3.\forall z_4 : \gamma_4.\varphi \to \zeta$$

where γ_j, φ, ψ and ζ are as in the case of a lower bound constraint.

The set constraint conjectures are of the form

$$\begin{gathered}\forall P_1 : \alpha_1.\forall P_2 : \alpha_2.\forall P_3 : \alpha_3.\forall P_4 : \alpha_4. \\ \varphi_1^1 \to \varphi_2^1 \to \varphi_3^2 \to \varphi_4^2 \to \varphi_5^3 \to \varphi_6^3 \to \varphi_7^4 \to \varphi_8^4 \to \bot\end{gathered}$$

where each α_i is an ι-relational type with minimum depth 2 and maximum depth 6 and each proposition φ_j^i is a lower bound constraint for P_i over $\{P_1, P_2, P_3, P_4\}$ if j is odd and an upper bound constraint for P_i over $\{P_1, P_2, P_3, P_4\}$ if j is even.

The positive version of the conjecture states that there is no solution to this collection of set constraints. The negative version can be proven by giving a solution.

14.4. **Higher-Order Unification.** Unlike first-order unification, higher-order is undecidable. In spite of this Huet's preunification algorithm [16] provides a reasonable method to search for solutions. A great deal of research has been done on higher-order unification and is ongoing today [28].

The generated conjectures in this class are essentially higher-order unification problems with eight flex-rigid pairs and four variables to instantiate. The problems are given in a universal form, so that the positive form states that there is no solution. The negative form could be proven by giving a solution. In general the conjectures have the form

$$\begin{gathered}\forall X_1 : \alpha_1.\forall X_2 : \alpha_2.\forall X_3 : \alpha_3.\forall X_4 : \alpha_4. \\ \varphi_1^1 \to \varphi_2^1 \to \varphi_3^2 \to \varphi_4^2 \to \varphi_5^3 \to \varphi_6^3 \to \varphi_7^4 \to \varphi_8^4 \to \bot\end{gathered}$$

where α_i is ι-pure with minimum depth 2 and maximum depth 6 and φ^i_j is a proposition of the form described below.

Each φ^i_j is a proposition of the form

$$\forall z_1 : \beta_1.\forall z_2 : \beta_2.\forall z_3 : \beta_3.\forall z_4 : \beta_4.X_i\ s_1\ \cdots\ s_n = t$$

where each β_k is an ι-pure type of minimum depth 0 and maximum depth 4 and t and each s_l are appropriately typed terms. The term t must be rigid, and specifically its head must be either one of z_1, z_2, z_3, z_4 (requiring a projection in Huet's terminology) or one of $\mathsf{Inj1}$, $\mathsf{Inj0}$ or setsum. The arguments the head of t are applied to are randomly generated in the same way as the arguments $s_1, \ldots, s_n$ of X_i. Terms are generated in their η-long form so that the important choices are always what to take as the head of a term of type ι. The allowed heads when generating random terms are the constant $\emptyset$, the unary functions $\mathsf{Inj1}$ and $\mathsf{Inj0}$, the binary function setsum and any variables in context (i.e., X_1, X_2, X_3, X_4, z_1, z_2, z_3, z_4 and any other variables that have been introduced due to new λ-abstractions due to generating the long normal form).

14.5. **Untyped Combinator Unification.** Since we are in a simply typed setting the untyped combinators are encoded as sets as described in Section 13. The generated conjectures are in the form of eight flex-rigid pairs making using four variables to be instantiated. Each conjecture is stated in a universal form that means there is no solution. Proving the negation of the conjecture will usually mean giving a solution, though given the classical setting it is also possible to provide multiple instantiations and prove one must be a solution. (This was also the case for the previous two classes of conjectures.) The conjectures have the form

$$\begin{gathered}\forall X.\mathsf{combinator}\ X \to \forall Y.\mathsf{combinator}\ Y \to \forall Z.\mathsf{combinator}\ Z \to \forall W.\mathsf{combinator}\ W \to \\ \varphi^X_1 \to \varphi^X_2 \to \varphi^Y_3 \to \varphi^Y_4 \to \varphi^Z_5 \to \varphi^Z_6 \to \varphi^W_7 \to \varphi^W_8 \to \bot\end{gathered}$$

where φ^V_i is a proposition giving a flex-rigid pair with local variables and with V as the head of the left. To be more specific each φ^V_i has the form

$$\begin{gathered}\forall x.\mathsf{combinator}\ x \to \forall y.\mathsf{combinator}\ y \to \forall z.\mathsf{combinator}\ z \to \forall w.\mathsf{combinator}\ w \to \\ \mathsf{combinator\_equiv}\ (V\ v_1\ v_2\ v_3\ v_4\ s_1\ \ldots\ s_n)\ t\end{gathered}$$

where each $v_i \in \{x, y, z, w\}$, t is a random rigid combinator and each of $s_1, \ldots, s_n$ is a random combinator. In this context a random rigid combinator is either $K\ t_1$ or $S\ t_1$ where t_1 is a random combinator, or $S\ t_1\ t_2$ where t_1 and t_2 are random combinators, or $v\ t_1\ \cdots\ t_n$ where $v \in \{x, y, z, w\}$ and $t_1, \ldots, t_n$ are random combinators. A random combinator is $h\ t_1\ \cdots\ t_n$ where $h \in \{S, K, X, Y, Z, W, x, y, z, w\}$ and $t_1, \ldots, t_n$ are random combinators.

Each of these problems can be viewed as a first-order problem. In the first-order variant we could assume everything is a combinator (so $\mathsf{combinator}$ can be omitted) and use equality to play the role of $\mathsf{combinator\_equiv}$. It should generally be possible to mimic the equational reasoning of a first-order proof in the set theory representation by using appropriate lemmas about $\mathsf{combinator}$ and $\mathsf{combinator\_equiv}$.

Furthermore it should be possible to define a notion of reduction and prove that if two terms are equivalent via $\mathsf{combinator\_equiv}$, then they must have a common reduct.

This would allow one to prove the positive version of the conjecture (meaning there is no solution).

14.6. **Abstract HF Problems.** The conjectures in the Abstract HF class are about hereditarily finite sets, but without assuming the full properties about the relevant relations, sets and functions. We fix 24 distinct variables: r_0, r_1 and r_2 of type $\iota\iota o$, x_0, x_1, x_2, x_3 and x_4 of type ι, f_0 and f_1 of type $\iota\iota$, g_0, g_1 and g_2 of type $\iota\iota\iota$ and p_0, p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9 and p_{10} of type ιo. Each of these variable has an intended meaning which we record in a substitution θ. Let θ be the substitution with $\mathsf{dom}(\theta)$ being the set of these 24 variables such that

$$\theta(r_0) = \in \qquad \theta(r_1) = \subseteq \qquad \theta(r_2) = (\lambda xy.x \cap y = \emptyset) \qquad \theta(x_0) = 0 \qquad \theta(x_1) = 1$$

$$\theta(x_2) = 2 \qquad \theta(x_3) = 3 \qquad \theta(x_4) = 4 \qquad \theta(f_0) = \wp \qquad \theta(f_1) = \mathsf{Sing}$$

$$\theta(g_0) = \mathsf{binunion} \qquad \theta(g_1) = \mathsf{binintersect} \qquad \theta(g_2) = \mathsf{setminus} \qquad \theta(p_0) = \mathsf{atleast2}$$

$$\theta(p_1) = \mathsf{atleast3} \qquad \theta(p_2) = \mathsf{atleast4} \qquad \theta(p_3) = \mathsf{atleast5} \qquad \theta(p_4) = \mathsf{atleast6}$$

$$\theta(p_5) = (\lambda X.\exists Y.Y \subseteq X \wedge (\neg(X \subseteq Y) \wedge \mathsf{atleast6}\ Y)) \qquad \theta(p_6) = \mathsf{exactly2}$$

$$\theta(p_7) = \mathsf{exactly3} \qquad \theta(p_8) = \mathsf{exactly4} \qquad \theta(p_9) = \mathsf{exactly5}$$

$$\theta(p_{10}) = (\lambda X.\mathsf{atleast6}\ X \wedge \neg\mathsf{atleast7}\ X).$$

Each generated conjecture is of the form

$$\forall r_0 r_1 r_2 : \iota\iota o.\forall x_0 x_1 x_2 x_3 x_4.\forall f_0 f_1 : \iota\iota.\forall g_0 g_1 g_2 : \iota\iota\iota.\forall p_0 \cdots p_{10} : \iota o.$$
$$\varphi_1 \to \cdots \to \varphi_n \to \psi.$$

The propositions $\varphi_1, \ldots, \varphi_n, \psi$ are chosen from a set of 1229 specific propositions.[11]

If the 24 variables above are considered constants (and we let y range over $\mathcal{V}_\iota \setminus \{x_0, x_1, x_2, x_3, x_4\}$, then we can describe the first-order terms and propositions from which the propositions are chosen. The first-order terms s, t range over

$$y \mid f_i\ s \mid g_i\ s\ t$$

and the first-order propositions φ, ψ range over

$$s = t \mid r_i\ s\ t \mid \forall y \in s.\varphi \mid \exists y \in s.\varphi \mid \neg\varphi \mid \varphi \to \psi \mid \varphi \wedge \psi \mid \varphi \vee \psi \mid \varphi \leftrightarrow \psi.$$

Note that all quantifiers in φ are bounded. This, combined with the nature of θ, means that each φ can be evaluated using the Ackermann interpretation of hereditarily finite sets as natural numbers [1]. Each of the 1229 specific propositions φ is such that $\hat{\theta}\varphi$ evaluates to true.

We finally describe how the choices of the specific ψ and $\varphi_1, \ldots, \varphi_n$ from the 1229 propositions in order to form the conjecture are made. First one of the 1229 is chosen to be the conclusion ψ (using 11 bits to make the choice). Next, for the remaining 1228 propositions, 4 bits are used to give one chance out of 16 that the proposition should be included among the hypotheses φ_i.

[11] The propositions are in the array `ahfprops` in `checking.ml`.

Proving the conjecture means the chosen hypotheses contain a sufficient amount of information to conclude ψ. To prove the negation of the conjecture requires finding an alternative substitution θ' for which all the φ_i are still true (or, more precisely, provable in HF) and yet ψ is false (its negation is provable in HF). This is not quite the same as finding a countermodel for a first-order sentence, since a finite countermodel is insufficient. The countermodel must have all the hereditarily finite sets in its universe of discourse. In practice this is unlikely to make a difference except in corner cases, so these conjectures can essentially be considered first-order problems.

14.7. **AIM Conjecture Problems.** There are two kinds of AIM Conjecture related problems: one using Loop_with_defs_cex1 and one using Loop_with_defs_cex2. In both cases the conjecture states that no loop exists with counterexamples of the first or second kind satisfying a number of extra equations. The extra equations either say that certain inner mappings commute (but not explicitly that *all* inner mappings commute) or say that certain inner mappings have a small order (at most 5). The first kind of extra equations will hold in all AIM loops while the second kind of extra equations will not. The result are conjectures that, roughly speaking, are in the neighborhood of The AIM Conjecture. In general each conjecture has the following form

$$\begin{gathered}\forall X.\forall mbs : \iota\iota.\forall e.\forall K : \iota\iota.\forall a : \iota\iota\iota.\forall T : \iota\iota.\forall LR : \iota\iota\iota.\forall I^1 J^1 I^2 J^2 : \iota\iota. \\ P\ X\ m\ b\ s\ e\ K\ a\ T\ L\ R\ I^1\ J^1\ I^2\ J^2 \\ \to A_1 \to \cdots \to A_l \\ \to O_1 \to \cdots \to O_k \\ \to \bot\end{gathered}$$

where P is either Loop_with_defs_cex1 or Loop_with_defs_cex2 and A_i and O_j are (locally quantified equations) described below. The value of l and k vary but we always have $l \leq 20$ and $k \leq 2$.

There are five available inner mappings with one parameter: T_x, I^1_x, J^1_x, I^2_x and J^2_x. There are two available inner mappings with two parameters: $L_{x,y}$ and $R_{x,y}$.

Each A_i is of the form $\forall x_1 \ldots x_n u.F(Gu) = G(Fu)$ where F and G are composites of randomly chosen inner mappings using some of the parameters chosen from $x_1, \ldots, x_n$. In the simplest case $n = 3$, F is a $F^1_{x_1}$ and G is $G^1_{x_2} \circ G^2_{x_3}$ where F^1, G^1 and G^2 are randomly chosen inner mappings with one parameter. For example, we could generate

$$\forall x_1 x_2 x_3 u.T_{x_1}(I^1_{x_2}(J^1_{x_3}u)) = I^1_{x_2}(J^1_{x_3}(T_{x_1}u)).$$

In the most complex cases $n = 8$ and F is of the form $F^1_{x_1,x_2} \circ F^2_{x_3} \circ F^3_{x_4}$ and G is of the form $G^1_{x_5,x_6} \circ G^2_{x_7,x_8}$ where F^1, G^1, G^2 are randomly chosen inner mappings with two parameters and F^2 and F^3 are randomly chosen inner mappings with with one parameter.

The O_j conditions (if any are included) are of the form

$$\forall x_1 \ldots x_n u. \underbrace{F \cdots (F}_{q} u) = u$$

where $q \in \{2, 3, 4, 5\}$, $n \in \{3, 4\}$ and F is formed as a composite of randomly chosen inner mappings with parameters from $x_1, \ldots, x_n$ as described above. This condition states that the order of the inner mapping F is finite and divides q (hence it can be at most 5).

Proving the negation of one of these conjectures involves giving a finite loop (implemented in the HF set theory) and proving all the appropriate properties. Proving one of the conjectures will often involve some equational reasoning that needs to be replayed as a proof term. In an unusual case, there might only be infinite counterexample. In that case the conjecture might be provable by an inductive argument since the conjecture says the proposition holds for all finite loops. Except for such unusual cases, these problems are first-order problems.

14.8. **Diophantine Modulo.** A Diophantine Modulo problem generates two polynomials p and q in variables x, y and z and a number m (of up to 64 bits). The conjecture is then as follows:

$$\forall xyz.\mathsf{equip\_mod}\ (p \uplus B(16))\ q\ B(m)\ \rightarrow\ \bot.$$

In this form the conjecture says there is no choice of (hereditarily finite) sets x, y and z such that the cardinality of p plus 16 is the same as the cardinality of q modulo m. The negation of the conjecture could be proven by giving appropriate x, y and z and proving they have the property.

The generation of a polynomial is simple and cannot fail. Polynomials are always of the form

$$\sum_{(i,j,k)\in\{0,1,2,3\}^3} B(n_{i,j,k}) x^i y^j z^k.$$

Here each $n_{i,j,k}$ is a natural number between 0 and 15 (using 4 bits of information, for a total of 256 bits total).[12] As a term, sums are represented using setsum ($\uplus$), products are represented using setprod ($\times$) and exponents are represented using setexp. Special cases are handled in special ways: if $c_{i,j,k}$ is 0, then the monomial is omitted; if $c_{i,j,k}$ is 1, then the factor is omitted; if the exponent of a variable is 0, then the factor is omitted; if the exponent of a variable is 1, then the exponent is omitted.

14.9. **Diophantine.** The final class is given by Diophantine problems (either equations or inequalities). Two polynomials p and q in variables x, y, z are generated (as described above). Each polynomial uses 256 bits of information. One extra bit is used to determine if the problem uses atleastp (for an inequality) or equip (for an equation). In total, no more than 513 bits are required and so this case never fails. The generated conjecture is then either of the form

$$\forall xyz.\mathsf{atleastp}\ (p \uplus B(16))\ q\ \rightarrow\ \bot$$

or

$$\forall xyz.\mathsf{equip}\ (p \uplus B(16))\ q\ \rightarrow\ \bot.$$

[12] A comment in the code in `checking.ml` claims this uses 128 bits, but this must be an error.

15. Conclusion

We have described the HF theory built into the Proofgold network. After each block Proofgold uses data from the Litecoin blockchain to pseudorandomly generate a conjecture within the HF theory for a bounty to be placed in the next Proofgold block. Resolving these conjectures is a form of delayed proof of work. By performing the delayed proof of work users can gain stake in the system that can be used to participate in the proof of stake part of the consensus algorithm.

Acknowledgment

This work has been supported by the European Research Council (ERC) Consolidator grant nr. 649043 *AI4REASON*.

References

[1] Ackermann, W.: Die Widerspruchsfreiheit der allgemeinen Mengenlehre. Mathematische Annalen **114**(1), 305–315 (1937)

[2] Aczel, P.: On relating type theories and set theories. In: Altenkirch, T., Naraschewski, W., Reus, B. (eds.) TYPES. Lecture Notes in Computer Science, vol. 1657, pp. 1–18. Springer (1998)

[3] Brown, C.E.: Solving for set variables in higher-order theorem proving. In: Voronkov, A. (ed.) Automated Deduction - CADE-18, 18th International Conference on Automated Deduction, Copenhagen, Denmark, July 27-30, 2002, Proceedings. Lecture Notes in Computer Science, vol. 2392, pp. 408–422. Springer (2002)

[4] Brown, C.E.: Reconsidering Pairs and Functions as Sets. Journal of Automated Reasoning **55**(3), 199–210 (Oct 2015)

[5] Brown, C.E.: Mizar's Tarski-Grothendieck as a Theory in Proofgold. Tech. rep., Czech Technical University in Prague (Aug 2020), http://grid01.ciirc.cvut.cz/~chad/pfgmizar.pdf

[6] Brown, C.E.: A theory supporting higher-order abstract syntax. Tech. rep., Czech Technical University in Prague (Aug 2020), http://grid01.ciirc.cvut.cz/~chad/hoas/pfghoas.pdf

[7] Brown, C.E., Pąk, K.: A tale of two set theories. In: Kaliszyk, C., Brady, E.C., Kohlhase, A., Coen, C.S. (eds.) Intelligent Computer Mathematics - 12th International Conference, CICM 2019, Prague, Czech Republic, July 8-12, 2019, Proceedings. Lecture Notes in Computer Science, vol. 11617, pp. 44–60. Springer (2019)

[8] Brown, C.E., Pąk, K.: AIM Loops and the AIM Conjecture. Formalized Mathematics **27**(4) (2019)

[9] de Bruijn, N.G.: A survey of the project AUTOMATH. In: Seldin, J.P., Hindley, J.R. (eds.) To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus, and Formalism, pp. 579–606. Academic Press (1980)

[10] de Bruijn, N.: Lambda calculus notation with nameless dummies, a tool for automatic formula manipulation, with application to the Church-Rosser theorem. Indagationes Mathematicae (Proceedings) **34**(5), 381–392 (1972)

[11] Church, A.: A formulation of the simple theory of types. The Journal of Symbolic Logic **5**, 56–68 (1940)

[12] Conway, J.H.: On numbers and games, Second Edition. A K Peters (2001)

[13] van Heijenoort, J.: From Frege to Gödel. A Source Book in Mathematical Logic 1879–1931. Harvard University Press, Cambridge, Massachusetts (1967)

[14] Hindley, J.R.: Basic Simple Type Theory, Cambridge Tracts in Theoretical Computer Science, vol. 42. Cambridge University Press (1997)

[15] Howard, W.: The formulas-as-types notion of construction. In: Seldin, J., Hindley, J. (eds.) To H.B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism. pp. 479–490. Academic Press, New York (1980)

[16] Huet, G.P.: A unification algorithm for typed lambda-calculus. Theor. Comput. Sci. **1**(1), 27–57 (1975)

[17] Kinyon, M.K., Veroff, R., Vojtěchovský, P.: Loops with abelian inner mapping groups: An application of automated deduction. In: Bonacina, M.P., Stickel, M.E. (eds.) Automated Reasoning and Mathematics - Essays in Memory of William W. McCune. Lecture Notes in Computer Science, vol. 7788, pp. 151–164. Springer (2013)

[18] Lee, G., Werner, B.: Proof-irrelevant model of CC with predicative induction and judgmental equality. Logical Methods in Computer Science **7**(4) (2011)

[19] Mamane, L.E.: Surreal numbers in Coq. In: Filliâtre, J.C., Paulin-Mohring, C., Werner, B. (eds.) Types for Proofs and Programs. pp. 170–185. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)

[20] Morse, A.P.: A Theory of Sets. Academic Press (1965)

[21] Obua, S.: Partizan Games in Isabelle/HOLZF. In: Barkaoui, K., Cavalcanti, A., Cerone, A. (eds.) ICTAC. Lecture Notes in Computer Science, vol. 4281, pp. 272–286. Springer (2006)

[22] Paulson, L.C.: Set theory for verification: I. from foundations to functions. Journal of Automated Reasoning **11**, 353–389 (1993)

[23] Prawitz, D.: Natural deduction: a proof-theoretical study. Dover (2006)

[24] Russell, B.: The Principles of Mathematics. Cambridge University Press (1903)

[25] Sørensen, M., Urzyczyn, P.: Lectures on the Curry-Howard Isomorphism. Rapport (Københavns universitet. Datalogisk institut), Datalogisk Institut, Københavns Universitet (1998)

[26] Su, B.: Mathcoin: A blockchain proposal that helps verify mathematical theorems in public. IACR Cryptol. ePrint Arch. **2018**, 271 (2018)

[27] Suppes, P.: Axiomatic Set Theory. Dover Books on Mathematics Series, Dover Publications (1972)

[28] Vukmirovic, P., Bentkamp, A., Nummelin, V.: Efficient full higher-order unification. In: Ariola, Z.M. (ed.) 5th International Conference on Formal Structures for Computation and Deduction, FSCD 2020, June 29-July 6, 2020, Paris, France (Virtual Conference). LIPIcs, vol. 167, pp. 5:1–5:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020)

[29] Werner, B.: Sets in types, types in sets. In: Abadi, M., Ito, T. (eds.) TACS. Lecture Notes in Computer Science, vol. 1281, pp. 530–346. Springer (1997)

[30] White, B.: Qeditas: A formal library as a bitcoin spin-off (2016), http://qeditas.org/docs/qeditas.pdf

[31] Zermelo, E.: Untersuchungen über die Grundlagen der Mengenlehre I. Mathematische Annalen **65**, 261–281 (1908), English translation, "Investigations in the foundations of set theory" in [13], pages 199–215