A Logical Framework for Set Theories

Arnon Avron
School of Computer Science
Tel Aviv University, Tel Aviv 69978, Israel
aa@math.tau.ac.il

Axiomatic set theory is almost universally accepted as the basic theory which provides the foundations of mathematics, and in which the whole of present day mathematics can be developed. As such, it is the most natural framework for Mathematical Knowledge Management. However, in order to be used for this task it is necessary to overcome serious gaps that exist between the "official" formulations of set theory (as given e.g. by formal set theory ZF) and actual mathematical practice.

In this work we present a new unified framework for formalizations of axiomatic set theories of different strength, from rudimentary set theory to full ZF. It allows the use of set terms, but provides a *static* check of their validity. Like the inconsistent "ideal calculus" for set theory, it is essentially based on just two set-theoretical principles: extensionality and comprehension (to which we add $\in$-induction and optionally the axiom of choice). Comprehension is formulated as: $x \in \{x \mid \varphi\} \leftrightarrow \varphi$, where $\{x \mid \varphi\}$ is a legal set term of the theory. In order for $\{x \mid \varphi\}$ to be legal, φ should be *safe* with respect to $\{x\}$, where safety is a relation between formulas and finite sets of variables. The various systems we consider differ from each other mainly with respect to the safety relations they employ. These relations are all defined purely syntactically (using an induction on the logical structure of formulas). The basic one is based on the safety relation which implicitly underlies commercial query languages for relational database systems (like SQL).

1 Introduction

Axiomatic set theory is almost universally accepted as the basic theory which provides the foundations of mathematics, and in which the whole of present day mathematics can (and many say: should) be developed. As such, it is the most natural framework for MKM (Mathematical Knowledge Management). Moreover: as is emphasized and demonstrated in [8], set theory has not only a great pragmatic advantage as a basic language for mathematical discourse, but it also has a great computational potential as a basis for specification languages, declarative programming, and proof verifiers. However, in order to be used for any of these tasks it is necessary to overcome the following serious gaps that exist between the "official" formulations of set theory (as given e.g. by Zermelo Fränkel Set Theory ZF; see e.g. [7]). and actual mathematical practice:

- ZF treats all the mathematical objects on a par, and so hid the computational significance of many of them. Thus although certain functions are first-class citizens in many programming languages, in set theory they are just "infinite sets", and ZF in its usual presentation is an extremely poor framework for computing with such sets (or handling them in a constructive way).
- Full ZF is far too strong for core mathematics, which practically deals only with a small fraction of the set-theoretical "universe". It is obvious that much weaker systems, corresponding to universes which are smaller, *more effective*, and better suited for computations , would do (presumably, such weaker systems will also be easier to mechanize).

The goal of this paper is to present a unified, user-friendly framework (originally developed in [5]) for formalizations of axiomatic set theories of different strength, from rudimentary set theory to full ZF. Our framework makes it possible to employ in a natural way all the usual set notations and constructs as found in textbooks on naive or axiomatic set theory (and *only* such notations). Another important feature of this framework is that its set of closed terms suffices for denoting every concrete set (including infinite ones!) that might be needed in applications, as well as for *computations* with sets.

Perhaps the most important problem which is solved in our framework is that official formalizations of axiomatic set theories in almost all textbooks are based on some *standard* first-order languages. In such languages terms are variables, constants, and sometimes function applications (like $x \cap y$). What is *not* available in the *official* languages of these formalizations is the use of set terms of the form ($\{x \mid \varphi\}$). As a result, already the formulation of the axioms is quite cumbersome, and even the formalization of elementary proofs becomes something practically incomprehensible. In contrast, *all* modern texts in all areas of mathematics (including set theory itself) use such terms extensively. For the purpose of mechanizing real mathematical practice and for automated or interactive theorem proving, it is therefore important to have formalizations of ZF and related systems which allow the use of such terms. Now, set terms *are* used in all textbooks on first-order set theories, as well as in several computerized systems. However, whenever they are intended to denote *sets* (rather than classes) they are introduced (at least partially) in a *dynamic* way, based for example on the “extension by definitions” procedure (see [16], Sect. 4.6): In order to be able to introduce some set term for a *set* (as well as a new operation on *sets*) it is necessary first to justify this introduction by *proving* a corresponding existence *theorem*. The very useful complete separation we have in first-order logic between the (easy) check whether a given expression is a well-formed term or formula, and the (difficult) check whether it is a theorem, is thus lost. By analogy to programs: texts in such dynamic languages can only be “interpreted”, but not “compiled”. In contrast, a crucial feature of our framework is that although it makes extensive use of set terms, the languages used in it are all *static*: the task of verifying that a given term or formula is well-formed is decidable, easily mechanizable, and completely separated from any task connected with proving theorems (like finding proofs or checking validity of given ones). Expanding the language is allowed only through *explicit* definitions (i.e. new valid expressions of an extended language will just be abbreviations for expressions in the original language). This feature has the same obvious advantages that static type-checking has over dynamic type-checking.

Two other important features of the framework we propose are:

- It provides a unified treatment of two important subjects of set theory: axiomatization and absoluteness (the latter is a crucial issue in independence proofs and in the study of models of set theories – see e.g. [13]). In the usual approaches these subjects are completely separated. Absoluteness is investigated mainly from a syntactic point of view, axiomatizations – from a semantic one. Here both are given the same syntactic treatment. In fact, the basis of the framework is its formulation of rudimentary set theory, in which only terms for absolute sets are allowed. The other set theories are obtained from it by small changes in the syntactic definitions.
- Most of our systems (including the one which is equivalent to ZF) have the remarkable property that every set or function that is implicitly definable in them already has a term in the corresponding language which denotes it. More precisely: if $\varphi(x, y_1, \dots, y_n)$ is a formula such that $\forall y_1, \dots, y_n \exists! x \varphi$ is provable, then there is a term $t(y_1, \dots, y_n)$ such that $\varphi(y_1, \dots, y_n, t(y_1, \dots, y_n))$ is provable. Hence, there is no need at all for the procedure of extension by definitions (and introduction of new symbols is completely reduced to using *abbreviations*).

2 The Major Ideas

Our basic assumption is that the sets which are interesting from a computational point of view are those which can be *defined* in the form $\{x \mid \varphi\}$ using a formula φ in some, intuitively meaningful, formal language. Of course, the paradoxes of naive set theory have shown that not every formula of such a language can be used for defining sets. Accordingly, the crucial question is: what formulas are "safe" for this task, and more generally: what formulas can be taken as defining a *construction* of a set from given objects (including other sets)? Various set theories provide different answers to this question. These answers are usually guided by semantic intuitions (like the limitation of size doctrine [7]). Since here we aim at a computerized system, we shall translate the various semantic principles into *syntactic* (and in our opinion, less ad-hoc) constraints on the logical form of formulas. For this, we combine ideas from three seemingly very different sources:

Set Theory Gödel's classical work [11] on the constructible universe L is best known for its use in consistency and independence proofs. However, it is of course of great interest also for the study of the general notion of constructions with sets. Thus for characterizing the "constructible sets" Gödel identified a set of operations on sets (which we may call "computable"), that can be used for "effectively" constructing new sets from given ones. For example, binary union and intersection are "effective", while the powerset operation is not. Gödel has provided a finite list of basic operations, from which all other "effective" (for his purposes) constructions can be obtained through compositions. Another very important idea which was introduced in [11] is *absoluteness* — a key property (see [13]) of formulas which are used for defining "constructible sets". Roughly, a formula is absolute if its truth value in a transitive class M, for some assignment v of objects from M to its free variables, depends only on v, but not on M.

Formal arithmetic Absoluteness is not a decidable property. Therefore a certain set Δ_0 of absolute formulas is extensively used in set theory as a syntactically defined approximation. Now a similar set Δ_0 of formulas (also called in [17] "bounded formulas" or "Σ_0-formulas") which has *exactly the same definition* (except that $\in$ is replaced by $<$) is used in formal arithmetic in order to characterize the *decidable* and the semi-decidable (r.e.) relations on the natural numbers. This fact hints at an intimate connection (investigated in [4]) between absoluteness/constructibility and decidability/computability.

Relational database theory: The importance of computations with sets to this area is obvious: to provide an answer to a query in a relational database, a computation should be made in which the input is a finite set of finite sets of tuples (the "tables" of the database), and the output should also be a finite set of tuples. In other words: the computation is done with (finite) sets. Accordingly, for *effective* computations with finite relations some finite set of basic operations has been identified in database theory, and this basic set defines (via composition) what is called there "the relational algebra" ([1, 18]). Interestingly, there is a lot of similarity between the list of operations used in the relational algebra and Gödel's list of basic operations mentioned above. However, much more important is again the strong connection (observed in ([3, 4]) between the notion of absoluteness used in set theory, and the notion of *domain independence* ([1, 18]) used in database theory, and practically serving as its counterpart of the notion of computability. A query in a database can be construe as a formula φ in the language of set theory, augmented with constants for the relations in the database. The answer to such query is the set of all n-tuples that satisfy φ, given the interpretations provided by the database for the extra constants (here n is the number of free variables in φ. If $n = 0$ then the answer to the query is either "yes" or "no"). A domain-independent (d.i.)

query is a query the answer to which depends only on the information included in the database, and on the objects which are mentioned in the query. Only such queries are considered meaningful. Moreover: the answer to such queries is always finite and *computable*. Therefore practical database query languages (like SQL) are designed so that only d.i. queries can be formulated in them, and each such query language is based on some syntactic criteria that ensure this property. In order to give these criteria a concise logical characterization, and in order to unify the notions of absoluteness and domain-independence, the formula *property* of d.i. was turned in [3, 4] into a *safety relation* $\succ$ between a formula φ and finite subsets of $Fv(\varphi)$. The intuitive meaning of "$\varphi(x_1,\dots,x_n,y_1,\dots,y_k) \succ \{x_1,\dots,x_n\}$" in databases is: "$\varphi(x_1,\dots,x_n,d_1,\dots,d_k)$ is d.i. for all values $d_1,\dots,d_k$". In particular, $\varphi \succ \emptyset$ if φ is *absolute* in the sense of axiomatic set theory.

In view of the connections between "absolute" and "decidable" and between "domain-independent" and "computable", (or "constructible"), in the realm of sets we shall intuitively take the meaning of "$\varphi(x_1,\dots,x_n,y_1,\dots,y_k) \succ \{x_1,\dots,x_n\}$" to be: "The collection $\{\langle x_1,\dots,x_n\rangle \mid \varphi\}$ is an acceptable set for all acceptable values of $y_1,\dots,y_k$, and it can be *constructed* from these values". The differences between the strength of systems is intuitively due to different interpretations of the vague notions of "acceptable" and "can be constructed". At least in the basic systems, but also in some of the less basic ones, a crucial part of the meaning of both concepts is the demand that $\{\langle x_1,\dots,x_n\rangle \mid \varphi\}$ is "domain independent" in a sense close to that used in database theory, i.e.: that φ determines this collection in an absolute way, independent of the extension of the "surrounding universe" V. In particular: $\varphi \succ \emptyset$ implies in such set theories that φ is absolute (in the set-theoretical sense mentioned above).

3 A Description of the General Framework

3.1 Languages

In our framework a language L for a set theory S should be based on some first-order signature σ which includes the binary predicate symbols $\in$ and $=$. Moreover: it should be introduced using a simultaneous recursive definition of the following three components: its set of terms, its set of formulas, and the *safety relation* $\succ$ that it uses between formulas and finite sets of variables. The recursive definition of these components includes at least the following conditions:

Terms:

- Every variable and every constant of σ is a term.
- If f is an n-ary function symbol of σ, and $t_1,\dots,t_n$ are terms, then $f(t_1,\dots,t_n)$ is a term.
- If x is a variable, and φ is a formula such that $\varphi \succ \{x\}$, then $\{x \mid \varphi\}$ is a term.

Formulas:

- If P is an n-ary predicate symbol of σ, and $t_1,\dots,t_n$ are terms, then $P(t_1,\dots,t_n)$ is an atomic formula.
- If φ and ψ are formulas, and x is a variable, then $\neg\varphi$, $(\varphi\wedge\psi)$, $(\varphi\vee\psi)$, and $\exists x\varphi$ are formulas. In an intuitionistic system so are also $(\varphi\rightarrow\psi)$ and $\forall x\varphi$ (but in the classical case $\rightarrow$ and $\forall$ are better taken as defined in terms of $\neg$, $\wedge$, and $\exists$).
- An optional construct which may be useful in our framework and is not available in first-order languages is the transitive closure operation TC. If it is included, then $(TC_{x,y}\varphi)(t,s)$ is a formula whenever φ is a formula, x,y are distinct variables, and t,s are terms. In this formula

all occurrences of x and y in φ are bound. The intended meaning of $(TC_{x,y}\varphi)(t,s)$ is the "disjunction": $\varphi\{s/x,t/y\} \vee \exists w_1(\varphi\{s/x,w_1/y\}) \wedge \varphi\{w_1/x,t/y\}) \vee \exists w_1 \exists w_2(\varphi\{s/x,w_1/y\} \wedge \varphi\{w_1/x,w_2/y\} \wedge \varphi\{w_2/x,t/y\}) \vee \dots$ (where $w_1, w_2, \dots,$ are all new variables)).

Safety Relation:

- $\varphi \succ \emptyset$ if φ is atomic.
- $\varphi \succ \{x\}$ if $\varphi \in \{x = t, t = x, x \in x, x \in t\}$, and $x \notin Fv(t)$.
- $\neg\varphi \succ \emptyset$ if $\varphi \succ \emptyset$.
- $\varphi \vee \psi \succ X$ if $\varphi \succ X$ and $\psi \succ X$.
- $\varphi \wedge \psi \succ X \cup Y$ if $\varphi \succ X$, $\psi \succ Y$ and $Y \cap Fv(\varphi) = \emptyset$, or $X \cap Fv(\psi) = \emptyset$.
- $\exists y\varphi \succ X - \{y\}$ if $y \in X$ and $\varphi \succ X$.
- $\forall x(\varphi \to \psi) \succ \emptyset$ if $\varphi \succ \{x\}$ and $\psi \succ \emptyset$[1]
- If TC is included in the language then $(TC_{x,y}\varphi)(x,y) \succ X$ if $\varphi \succ X$, and $\{x,y\} \cap X \neq \emptyset$.

Notes:

1. The clauses concerning $\succ$ form a generalization (and simplification) of the definition of "syntactically safe" formulas from [18] (see [3, 4, 6]). The passage from the property of domain independence to the safety relation is mainly needed for an appropriate handling of conjunction.

2. Recalling the intended intuitive meaning(s) of our safety relations, is not difficult to see that any safety relation $\succ$ should satisfy the conditions listed above. As an example, we explain the most complicated of them: the one connected with $\wedge$. Assume for simplicity that $\theta = \varphi \wedge \psi$, where $Fv(\varphi) = \{x,z\}, Fv(\psi) = \{x,y,z\}, \varphi \succ \{x\}$, and $\psi \succ \{y\}$. Given some "acceptable" set c, we should show that the collection $E(c)$ of all $\langle x,y \rangle$ such that $\theta(x,y,c)$ should also be taken as "acceptable". Now the assumption that $\varphi \succ \{x\}$ implies that the collection $Z(c)$ of all x such that $\varphi(x,c)$ is "acceptable". In turn, the the assumption that $\psi \succ \{y\}$ implies that for every d in this set, the collection $W(c,d)$ of all y such that $\psi(d,y,c)$ is "acceptable" . Since $E(c)$ is the union for $d \in Z(c)$ of the sets $\{d\} \times W(c,d)$, it is constructible from "acceptable" sets using Gödel's basic operations mentioned above, and so it too should intuitively be "acceptable" in any reasonable set theory. What is more, if $Z(c)$ is "constructible" from c (in an absolute way), and $W(c,d)$ is "constructible" from c and d (in an absolute way), then this argument shows that $E(c)$ is "constructible" from c (in an absolute way) as well.

3. The recursive definition of $\succ$ should ensure that $\succ$ has the following properties:

   - If $\varphi \succ X$ then $X \subseteq Fv(\varphi)$.
   - If $\varphi \succ X$ and $Z \subseteq X$, then $\varphi \succ Z$.
   - If $\varphi \succ \{x_1, \dots, x_n\}$, $v_1, \dots v_n$ are n distinct variables not occurring in φ, and φ' is obtained from φ by replacing all occurrences of x_i by v_i ($i = 1, \dots, n$), then $\varphi' \succ \{v_1, \dots, v_n\}$

   It is easy to verify that all the safety relations used in the examples below have these properties, and so there is no need to add corresponding clauses to their definitions (but this might not be the case in general).

[1] In the classical case this condition is derivable from the others.

3.2 Logics

Our framework allows the use of any logic that is based one of the two languages it employs (with classical and intuitionistic logics as the natural choices). One should note however the following points:

1. Our languages provide much richer classes of terms than those allowed in orthodox first-order systems. In particular: a variable can be bound in them within a term. The notion of a term being free for substitution is generalized accordingly (also for substitutions within terms!). As usual this amounts to avoiding the capture of free variables within the scope of an operator which binds them. Otherwise the rules/axioms concerning the quantifiers and terms remain unchanged (for example: $\varphi[x \mapsto t] \to \exists x\varphi$ is valid for *every* term t which is free for x in φ).
2. The rule of α-conversion (change of bound variables) should be available in the logic.
3. The substitution of equals for equals should be allowed within any context (under the usual conditions concerning bound variables). The same should apply for the substitution of a formula for an equivalent formula in any context in which the substitution makes sense. In particular, the following schema should be valid whenever $\{x \mid \varphi\}$ and $\{x \mid \psi\}$ are legal terms:

$$\forall x(\varphi \leftrightarrow \psi) \to \{x \mid \varphi\} = \{x \mid \psi\}$$

4. The set of valid formulas of first-order languages enriched with the TC operator is not even arithmetical. Hence no sound and complete formal system for it is possible. It follows that only appropriate formal approximations of the intended underlying logic may be used in practice. The best known approximation is the one given in [14], using a Hilbert-type system. An equivalent Gentzen-type formulation (*with* cuts) has been given in [2]. In that system mathematical induction is presented as the following logical rule:

$$\frac{\Gamma, \psi, \varphi \Rightarrow \Delta, \psi[x \mapsto y]}{\Gamma, \psi[x \mapsto s], (TC_{x,y}\varphi)(s,t) \Rightarrow \Delta, \psi[x \mapsto t]}$$

where x and y are not free in Γ, Δ, and y is not free in ψ.

3.3 Axioms

The main part of all systems in our framework consists of the following axioms and axiom schemes (our version of the "ideal calculus" [7], augmented with the assumption that we are dealing with the cumulative universe):

Extensionality:

- $\forall y(y = \{x \mid x \in y\})$

Comprehension Schema:

- $\forall x(x \in \{x \mid \varphi\} \leftrightarrow \varphi)$

The Regularity Schema ($\in$-induction):

- $(\forall x(\forall y(y \in x \to \varphi[x \mapsto y]) \to \varphi)) \to \forall x\varphi$

Notes:

1. Thus the main parts of the various set theories we consider differ only with respect to the power of their comprehension scheme. This, in turn, depends only on the safety relation used by each.
2. It is easy to see (see [3]) that our assumptions concerning the underlying logic and the comprehension schema together imply that the above formulation of the extensionality axiom is equivalent to the more usual one: $\forall z(z \in x \leftrightarrow z \in y) \rightarrow x = y$.
3. The first two axioms immediately entail the following two principles (where t is an arbitrary term):
   - $\{x \mid x \in t\} = t$ (provided $x \notin Fv(t)$)
   - $t \in \{x \mid \varphi\} \leftrightarrow \varphi[x \mapsto t]$ (provided t is free for x in φ)

   These principles are counterparts of the reduction rules (η) and (β) (respectively) from the λ-calculus. Like their counterparts, they are designed to be used as simplification rules (at least in the solution of elementary problems).

4 The Most Basic System

Our most basic system is the one which corresponds to the minimal safety relation (in a language without TC). For the reader convenience, we explicitly present the definition of this relation:

Definition 1 *The relation $\succ_{RST}$ is inductively defined as follows:*

1. $\varphi \succ_{RST} \emptyset$ *if* φ *is atomic.*
2. $\varphi \succ_{RST} \{x\}$ *if* $\varphi \in \{x = t, t = x, x \in t, x \in x\}$, *and* $x \notin Fv(t)$.
3. $\neg\varphi \succ_{RST} \emptyset$ *if* $\varphi \succ_{RST} \emptyset$.
4. $\varphi \vee \psi \succ_{RST} X$ *if* $\varphi \succ_{RST} X$ *and* $\psi \succ_{RST} X$.
5. $\varphi \wedge \psi \succ_{RST} X \cup Y$ *if* $\varphi \succ_{RST} X$, $\psi \succ_{RST} Y$, *and* $Y \cap Fv(\varphi) = \emptyset$.
6. $\exists y\varphi \succ_{RST} X - \{y\}$ *if* $y \in X$ *and* $\varphi \succ_{RST} X$.

We denote by RST (Rudimentary Set Theory) the set theory induced by $\succ_{RST}$ (within the framework described above). Note that RST without the $\in -$induction schema can be shown to be equivalent to Gandy's basic set theory [10], and to the system called BST_0 in [15]).

The following theorem about RST can easily be proved:

Theorem 1 *Given an expression E and a finite set X of variables, it is decidable in polynomial time whether E is a valid term of RST, whether it is a valid formula of RST, and if the latter holds, whether $E \succ_{RST} X$.*

Note. The last theorem is of a crucial importance from implementability point of view, and it obtains also for all the extensions of RST discussed (explicitly or implicitly) below. In order to ensure it, we did not include in the definition of safety relations the natural condition that if $\varphi \succ X$ and ψ is (logically) equivalent to φ (where $Fv(\varphi) = Fv(\psi)$) then also $\psi \succ X$. However, we obviously do have that if $\varphi \succ_{RST} \{x\}$, and $\vdash_{RST} \varphi \leftrightarrow \psi$, then $\vdash_{RST} x \in \{x \mid \varphi\} \leftrightarrow \psi$, and so $\vdash_{RST} \exists Z \forall x. x \in Z \leftrightarrow \psi$. Again this is true for any system in our framework.

4.1 The Power of *RST*

In the language of *RST* we can introduce as *abbreviations* most of the standard notations for sets used in mathematics. Again, all these abbreviations should be used in a purely static way: no justifying propositions and proofs are needed. Here are some examples:

- $\emptyset =_{Df} \{x \mid x \in x\}$.
- $\{t_1,\ldots,t_n\} =_{Df} \{x \mid x = t_1 \vee \ldots \vee x = t_n\}$ (where x is new).
- $\langle t,s\rangle =_{Df} \{\{t\},\{t,s\}\}$.
- $\langle t_1,\ldots,t_n\rangle$ is $\emptyset$ if $n=0$, t_1 if $n=1$, $\langle\langle t_1,\ldots,t_{n-1}\rangle,t_n\rangle$ if $n\geq 2$.
- $\{x \in t \mid \varphi\} =_{Df} \{x \mid x \in t \wedge \varphi\}$, provided $\varphi \succ_{RST} \emptyset$. (where $x \notin Fv(t)$).
- $\{t \mid x \in s\} =_{Df} \{y \mid \exists x.x \in s \wedge y = t\}$ (where y is new, and $x \notin Fv(s)$).
- $s \times t =_{Df} \{x \mid \exists a \exists b.a \in s \wedge b \in t \wedge x = \langle a,b\rangle\}$ (where x,a and b are new).
- $\{\langle x_1,\ldots,x_n\rangle \mid \varphi\} =_{Df} \{z \mid \exists x_1 \ldots \exists x_n.\varphi \wedge z = \langle x_1,\ldots,x_n\rangle\}$, if $\varphi \succ_{RST} \{x_1,\ldots,x_n\}$ and $z \notin Fv(\varphi)$.
- $s \cap t =_{Df} \{x \mid x \in s \wedge x \in t\}$ (where x is new).
- $s \cup t =_{Df} \{x \mid x \in s \vee x \in t\}$ (where x is new).
- $s - t =_{Df} \{x \mid x \in s \wedge x \notin t\}$ (where x is new).
- $S(x) =_{Df} x \cup \{x\}$
- $\bigcup t =_{Df} \{x \mid \exists y.y \in t \wedge x \in y\}$ (where x and y are new).
- $\bigcap t =_{Df} \{x \mid \exists y(y \in t \wedge x \in y) \wedge \forall y(y \in t \to x \in y)\}$ (where x,y are new).
- $\iota x\varphi =_{Df} \bigcap\{x \mid \varphi\}$ (provided $\varphi \succ \{x\}$).
- $P_1(z) = \iota x.\exists v \exists y(v \in z \wedge x \in v \wedge y \in v \wedge z = \langle x,y\rangle)$
- $P_2(z) = \iota y.\exists v \exists x(v \in z \wedge x \in v \wedge y \in v \wedge z = \langle x,y\rangle)$
- $\lambda x \in s.t =_{Df} \{\langle x,t\rangle \mid x \in s\}$ (where $x \notin Fv(s)$)
- $f(x) =_{Df} \iota y.\exists z \exists v(z \in f \wedge v \in z \wedge y \in v \wedge z = \langle x,y\rangle)$
- $Dom(f) =_{Df} \{x \mid \exists z \exists v \exists y(z \in f \wedge v \in z \wedge y \in v \wedge x \in v \wedge y = f(x)\}$
- $Rng(f) =_{Df} \{y \mid \exists z \exists v \exists x(z \in f \wedge v \in z \wedge y \in v \wedge x \in v \wedge y = f(x)\}$
- $f/s =_{Df} \{\langle x, f(x)\rangle \mid x \in s\}$ (where x is new).

Notes

1. It is straightforward to check that in all these abbreviations the right hand side is a valid term of *RST* (provided that the terms/formulas occurring in it are valid terms/well-formed formulas of *RST*). We explain $s \times t$ by way of example: since a and b are new, $a \in s \succ_{RST} \{a\}$, and $b \in t \succ_{RST} \{b\}$. Since $b \notin Fv(a \in s)$, this implies that $a \in s \wedge b \in t \succ_{RST} \{a,b\}$. Similarly, $a \in s \wedge b \in t \wedge x = \langle a,b\rangle \succ_{RST} \{a,b,x\}$. It follows that $\exists a \exists b.a \in s \wedge b \in t \wedge x = \langle a,b\rangle \succ_{RST} \{x\}$. Hence our term for $s \times t$ (which is the most natural one) is a valid term of *RST*.

2. It can easily be seen that according to these definitions, $\bigcap\emptyset = \emptyset$, and so $\iota x\varphi$ denotes $\emptyset$ if there is no set which satisfies φ, while it denotes the intersection of all the sets which satisfy φ otherwise. In particular: if there is exactly one set which satisfies φ, and $\varphi \succ \{x\}$, then $\iota x\varphi$ denotes this unique set (this fact has already been used above). It follows that if $\varphi(y_1,\ldots,y_n,x)$ implicitly defines (in some theory extending the basic theory of our framework) a function f_φ such that for all $y_1,\ldots,y_n$, $f_\varphi(y_1,\ldots,y_n)$ is the unique x such that $\varphi(y_1,\ldots,y_n,x)$, and if $\varphi \succ \{x\}$, then there is a term in the language which explicitly denotes f_φ; no extension of the language is needed for that.
3. It is easy to see that the usual reduction rules of the typed λ-calculus follow from the corresponding reduction rules described in Section 3.3. In particular: $\vdash_{RST} a \in s \to (\lambda x \in s.t)(a) = t\{a/x\}$.

Exact characterizations of the operations that are explicitly definable in RST, and of the strength of RST, are given in the following theorems and corollary

Theorem 2

1. *If F is an n-ary rudimentary function*[2] *then there exists a formula φ s. t.:*
   - *(a)* $Fv(\varphi) = \{y, x_1,\ldots,x_n\}$
   - *(b)* $\varphi \succ_{RST} \{y\}$
   - *(c)* $F(x_1,\ldots,x_n) = \{y \mid \varphi\}$.
2. *If φ is a formula such that:*
   - *(a)* $Fv(\varphi) = \{y_1,\ldots,y_k,x_1,\ldots,x_n\}$
   - *(b)* $\varphi \succ_{RST} \{y_1,\ldots,y_k\}$

   then there exists a rudimentary function F such that:

$$F(x_1,\ldots,x_n) = \{\langle y_1,\ldots,y_k\rangle \mid \varphi\}$$

Corollary 1 *If $Fv(\varphi) = \{x_1,\ldots,x_n\}$, and $\varphi \succ_{RST} \emptyset$ then φ defines a rudimentary predicate P. Conversely, if P is rudimentary then there is a formula φ such that $\varphi \succ_{RST} \emptyset$ and φ defines P.*

4.2 Generalized Absoluteness

For simplicity of presentation, we assume the cumulative universe V of ZF, and formulate our definitions accordingly. It is easy to see that V is a model of RST (with the obvious interpretations of RST's terms).

Definition 2 *Let $\mathcal{M}$ be a transitive model of RST. Define the relativization to $\mathcal{M}$ of the terms and formulas of RST recursively as follows:*

- $t_{\mathcal{M}} = t$ *if t is a variable or a constant.*
- $\{x \mid \varphi\}_{\mathcal{M}} = \{x \mid x \in \mathcal{M} \wedge \varphi_{\mathcal{M}}\}$.
- $(t = s)_{\mathcal{M}} = (t_{\mathcal{M}} = s_{\mathcal{M}}) \quad (t \in s)_{\mathcal{M}} = (t_{\mathcal{M}} \in s_{\mathcal{M}})$.
- $(\neg\varphi)_{\mathcal{M}} = \neg\varphi_{\mathcal{M}} \quad (\varphi \vee \psi)_{\mathcal{M}} = \varphi_{\mathcal{M}} \vee \psi_{\mathcal{M}}. \quad (\varphi \wedge \psi)_{\mathcal{M}} = \varphi_{\mathcal{M}} \wedge \psi_{\mathcal{M}}$.
- $(\exists x\varphi)_{\mathcal{M}} = \exists x(x \in \mathcal{M} \wedge \varphi_{\mathcal{M}})$.

Definition 3 *Let T be an extension of RST such that $V \models T$.*

[2] The class of rudimentary set functions was introduced independently by Gandy ([10]) and Jensen ([12]). See also [9], Sect. IV.1.

1. *Let t be a term, and let $Fv(t) = \{y_1, \ldots, y_n\}$. We say that t is T-*absolute *if the following is true (in V) for every transitive model $\mathscr{M}$ of T:*

$$\forall y_1 \ldots \forall y_n . y_1 \in \mathscr{M} \wedge \ldots \wedge y_n \in \mathscr{M} \to t_{\mathscr{M}} = t$$

2. *Let φ be a formula, and let $Fv(\varphi) = \{y_1, \ldots, y_n, x_1, \ldots, x_k\}$. We say that φ is T-*absolute for *$\{x_1, \ldots, x_k\}$ if $\{\langle x_1, \ldots, x_k\rangle \mid \varphi\}$ is a set for all values of the parameters $y_1, \ldots, y_n$, and the following is true (in V) for every transitive model $\mathscr{M}$ of T:*

$$\forall y_1 \ldots \forall y_n . y_1 \in \mathscr{M} \wedge \ldots \wedge y_n \in \mathscr{M} \to [\varphi \leftrightarrow (x_1 \in \mathscr{M} \wedge \ldots \wedge x_k \in \mathscr{M} \wedge \varphi_{\mathscr{M}})]$$

Thus a term is T-absolute if it has the same interpretation in all transitive models of T which contains the values of its parameters, while a formula is T-absolute for $\{x_1, \ldots, x_k\}$ if it has the same extension (which should be a set) in all transitive models of T which contains the values of its other parameters. In particular: φ is T-absolute for $\emptyset$ iff it is absolute relative to T in the usual sense of set theory (see e.g. [13]), while φ is T-absolute for $Fv(\varphi)$ iff it is domain-independent in the sense of database theory for transitive models of T.

Theorem 3

1. *Any valid term t of RST is RST-absolute.*
2. *If $\varphi \succ_{RST} X$ then φ is RST-absolute for X.*

5 Handling the Axioms of ZF and ZFC

5.1 Subsets, replacement, and Powerset

The definability of $\{t, s\}$ and of $\bigcup t$ in the language of RST means that the axioms of pairing and union are provable in RST. We turn now to the question how to deal with the other comprehension axioms of ZF within the proposed framework. We start with the comprehension axioms that remain valid if we limit ourselves to hereditarily finite sets. It can be shown ([4]) that each of them can be captured (in a modular way) by adding to the definition of $\succ_{RST}$ a certain syntactic condition. Here are those conditions:

Separation: $\varphi \succ \emptyset$ for every formula φ.

Replacement: $\exists y \varphi \wedge \forall y(\varphi \to \psi) \succ X$ if $\psi \succ X$, and $X \cap Fv(\varphi) = \emptyset$.

Powerset: $\forall y(y \in x \to \varphi) \succ (X - \{y\}) \cup \{x\}$ if $\varphi \succ X$, $y \in X$, and $x \notin Fv(\varphi)$.

Another (and perhaps simpler) method to handle the powerset axiom is to enrich first the language with a new binary relation $\subseteq$. Then add to the definition of the safety relation the condition: $x \subseteq t \succ \{x\}$ if $x \notin Fv(t)$. Finally, add the usual definition of $\subseteq$ in terms of $\in$ as an extra *axiom*: $\forall x \forall y(x \subseteq y \leftrightarrow \forall z(z \in x \to z \in y))$. Alternatively, since $\subseteq$ is now taken as primitive, it might be more natural to use it as such in our axioms. This means that instead of adding the above axiom, it might be preferable to replace the single extensionality axiom of BZF with the following three: (Ex1) $x \subseteq y \wedge y \subseteq x \to x = y$, (Ex2) $z \in x \wedge x \subseteq y \to z \in y$, and (Ex3) $x \subseteq y \vee \exists z(z \in x \wedge z \notin y)$.

Note. If any of the conditions introduced in this subsection is used then the counterpart of Theorem 3 is not valid for the resulting system. Hence these conditions are not coherent with our initial intuitions (Thus from the perspective of our framework, the condition that corresponds to the separation schema means that from the point of view of ZF, every formula defines a "decidable" relation on the universe V of sets). As a compensation, we have the following remarkable property of the condition that corresponds to replacement (see [5]):

Theorem 4 *Let $\mathscr{T}$ be a set theory in our framework such that the corresponding safety relation $\succ_{\mathscr{T}}$ satisfies the condition that corresponds to replacement. Then for any formula φ of $\mathscr{T}$ such that $Fv(\varphi) = \{y_1,\ldots,y_n,x\}$), there exists a term t_φ of $\mathscr{T}$ such that $Fv(t_\varphi) = \{y_1,\ldots,y_n\}$, and*

$$\vdash_{\mathscr{T}} \forall y_1,\ldots,y_n \exists! x\varphi \to \forall y_1,\ldots,y_n(\varphi[x \mapsto t_\varphi])$$

5.2 The Axiom of Infinity

Next we turn to the axiom of Infinity — the only comprehension axiom that necessarily takes us out of the realm of finite sets. As long as we stick to first-order languages, it seems impossible to incorporate it into our systems by just imposing new simple syntactic conditions on the safety relation. Instead, the best way to capture it is to add to the basic signature a new constant HF (interpreted as the collection $\mathscr{HF}$ of hereditarily finite sets) together with the following counterparts of *Peano's axioms:*

1. $\emptyset \in HF$
2. $\forall x \forall y. x \in HF \wedge y \in HF \to x \cup \{y\} \in HF$
3. $\varphi(0) \wedge (\forall x \forall y. \varphi(x) \wedge \varphi(y) \to \varphi(x \cup \{y\}) \to \forall x \in HF. \varphi(x)$

Definition 4 $RST\omega$ is the theory which is obtained from RST by the addition of the constant HF and the above counterparts of Peano's axioms.

On the other hand, if a language with TC is used, then we get the infinity axiom for free, since both $\mathscr{HF}$ and the set ω of the finite ordinals are definable in this extended language by valid terms (see [6]). Thus the one that defines ω is $\omega = \{y \mid \exists x. x = \emptyset \wedge (TC_{x,y} y = \{z \mid z = x \vee z \in x\})(x,y)\}$.

Definition 5 Let $\succ_{PZF}$ be the minimal safety relation in a language with TC (note that the only difference between $\succ_{PZF}$ and $\succ_{RST}$ is the extra clause for TC). We denote by PZF (predicative set theory) the set theory induced by $\succ_{PZF}$ within our framework.

Note. An important property of $RST\omega$ and PZF is that Theorem 3 does remain valid if instead of RST we consider either of them. Hence these systems *are* coherent with our initial motivations and intuitions.

5.3 The Axiom of Choice

The full set theory ZFC has one more axiom, which does not fit into the formal framework described above: *AC* (the axiom of choice). It seems that the most natural way to incorporate it into our framework is by further extending the set of terms, using Hilbert's ε symbol, together with its usual characterizing axiom (which is equivalent to the axiom of global choice): $\exists x\varphi \to \varphi[x \mapsto \varepsilon x\varphi]$. It should be noted that this move is not in line with our stated goal of employing only standard notations used in textbooks, but some price should be paid for including the axiom of choice in a system.

6 Structures and Computations

Let $\mathscr{T}$ be a theory formulated within the classical part of our framework. From the Platonist point of view its set of closed terms induces some subset $\mathscr{S}(\mathscr{T})$ of the universe V of sets. The identity of $\mathscr{S}(\mathscr{T})$ depends only on the *language* of $\mathscr{T}$ and on the interpretations of the symbols its signature has in addition to $\in, =$, and $\subseteq$ (if such symbols exist). It does not depend on its axioms. In addition, for any transitive model $\mathscr{M}$ of $\mathscr{T}$, $\mathscr{S}(\mathscr{T})$ determines some subset $\mathscr{M}(\mathscr{T})$ of $\mathscr{M}$ (which might not be an element of $\mathscr{M}$).

Now a theory $\mathscr{T}$ is computationally interesting if the set $\mathscr{S}(\mathscr{T})$ it induces is a "universe" in the sense that it is a transitive model of $\mathscr{T}$. According to our guiding ideas, such a theory $\mathscr{T}$ and its model $\mathscr{S}(\mathscr{T})$ have a special significance from a computational point of view if the identity of the latter is *absolute* in the sense that $\mathscr{M}(\mathscr{T}) = \mathscr{S}(\mathscr{T})$ for any transitive model $\mathscr{M}$ of $\mathscr{T}$ (implying that $\mathscr{S}(\mathscr{T})$ is actually a *minimal* transitive model of $\mathscr{T}$). From results in [6] it follows that at least the following theories have both properties:

RST**:** Its minimal model $\mathscr{S}(RST)$ is identical to $\mathscr{HF}$, which is J_1 in Jensen's hierarchy ([12, 9]), and L_ω in Gödel hierarchy ([11, 9]) of constructible sets.

$RST\omega$**:** Its minimal model $\mathscr{S}(RST\omega)$ is J_2 in Jensen's hierarchy.

$PZF_{\mathscr{TCL}}$**:** Its minimal model is $J_{\omega^\omega} = L_{\omega^\omega}$.

References

[1] S. Abiteboul, R. Hull & V. Vianu (1995): *Foundations of Databases*. Addison-Wesley.

[2] A. Avron (2003): *Transitive closure and the mechanization of mathematics*, pp. 149–171. *Applied Logics* 28, Kluwer Academic Publishers.

[3] A. Avron (2004): *Safety signatures for first-order languages and their applications*. In Hendricks et al., editor: *In First-Order Logic Revisited*, Logos Verlag, pp. 37–58.

[4] A. Avron (2008): *Constructibility and decidability versus domain independence and absoluteness*. *Theoretical Computer Science* 394, pp. 144–158, doi:`10.1016/j.tcs.2007.12.008`.

[5] A. Avron (2008): *A Framework for Formalizing Set Theories Based on the Use of Static Set Terms*. In: *Pillars of Computer Science*, *Lecture Notes in Computer Science* 4800, Springer, pp. 87–106, doi:`10.1007/978-3-540-78127-1_6`.

[6] A. Avron (2010): *A new approach to predicative set theory*. *Ways of Proof Theory*, pp. 31–63.

[7] A. Fraenkel Y. Bar-Hillel & A. Levy (1973): *Foundations of Set Theory*, second edition. *Studies in Logic and the Foundations of Mathematics* 67, Elsevier, Amsterdam.

[8] D. Cantone, E. Omodeo & A. Policriti (2001): *Set Theory for Computing: From Decisions Procedures to Declarative Programming with Sets*. Monographs in Computer Science, Springer.

[9] K. J. Devlin (1984): *Constructibility*. 6, Springer-Verlag.

[10] R. O. Gandy (1974): *Set-theoretic functions for elementary syntax: in Proceedings of Symposia in Pure Mathematics*. In: *Axiomatic set theory, Part 2*, AMS, Providence, Rhode Island, pp. 103–126.

[11] K. Gödel (1940): *The Consistency of the Axion of Choice and of the Generalized Continuum Hypothesis with the Axioms of Set Theory*. 3, Princeton University Press, Princeton, N.J.

[12] R. B. Jensen (1972): *The fine structure of the constructible hierarchy*. *Annals of Mathematical Logic* 4, pp. 229–308.

[13] K. Kunen (1980): *Set Theory: an Introduction to Independence Proofs*. *Studies in Logic and the Foundations of Mathematics* 102, Elsevier, Amsterdam.

[14] J. Myhill (1952): *A derivation of number theory from ancestral theory*. *Journal of Symbolic Logic* 17, pp. 292–297.

[15] V. Y. Sazonov (1997): *On bounded set theory*. In: *Proceedings of the 10th International Congress on Logic, Methodology and Philosophy of Sciences*, I: Logic and Scientific Method, Kluwer Academic Publishers, Florence, pp. 85–103.

[16] J. R. Shoenfield (1967): *Mathematical Logic*. Addison-Wesley.

[17] R. M. Smullyan (1992): *The Incompleteness Theorems*. Oxford University Press.

[18] J. D. Ullman (1998): *Principles of database and knowledge-base systems*. Computer Science Press.