A Framework for Formalizing Set Theories Based on the Use of Static Set Terms

Arnon Avron

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

To Boaz Trakhtenbrot: a scientific father, a friend, and a great man.

Abstract. 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).

Our framework makes it possible to reduce all extensions by definitions to abbreviations. Hence it is very convenient for mechanical manipulations and for interactive theorem proving. It also provides a unified treatment of comprehension axioms and of absoluteness properties of formulas.

1 Introduction

The goal of this paper is to develop a unified, user-friendly framework for formalizations of axiomatic set theories of different strength, from rudimentary set theory to full ZF. The work in a formal system that is constructed within such a framework should be very close to the way work in set theories is practically done in reality. In particular, it should be 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).

Our starting point is what is known as the "ideal calculus" for naive set theory (see [10], Sect. III.1). This very simple calculus is based on just two set-theoretical principles: extensionality and full comprehension. It thus exactly reflects our initial, immediate intuitions concerning sets (before becoming aware

A. Avron et al. (Eds.): Trakhtenbrot/Festschrift, LNCS 4800, pp. 87–106, 2008.

of the inconsistencies they involve). Now in its most transparent formal presentation, the ideal calculus employs set terms of the form $\{x \mid \varphi\}$, where x is a variable and φ is *any* formula in which x occurs free. Then the comprehension principle is most succinctly formulated as follows:

$$x \in \{x \mid \varphi\} \leftrightarrow \varphi$$

Unfortunately, it is well known that this principle leads to paradoxes (like Russel's paradox). Hence all set theories that are believed to be consistent impose constraints on the use of this principle. In all textbooks the choice of these constraints is guided by semantic intuitions (like the limitation of size doctrine [10,16]), especially the question: what operations on sets are "safe". Since it is one of our main purposes to remain as close to the "ideal calculus" as possible, on one hand, and we aim at computerized systems, on the other, we shall translate the various semantic principles into *syntactic* constraints on the logical form of formulas. Given a set theory S, we shall call a formula $\varphi(x)$ (which may have free variables other than x) S-safe with respect to x if $\{x \mid \varphi\}$ is a valid term of S (which intuitively means that according to the principles accepted by S, the set denoted by this term exists for all values of the other parameters). Thus "safety" will basically be here a relation between formulas and variables. (Actually, in order to define it syntactically we shall need to generalize it to a relation between formulas and finite sets of variables.) The various systems we consider differ from each other only with respect to the safety relations they employ.

Another problem 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 usually *not* available in the *official* languages of these formalizations is the use of set terms of the form described above ($\{x \mid \varphi\}$). 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 [20], 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 same is basically true in case set terms are *officially* used to denote "classes", as in [18], Sect. I.4.) 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.[1]

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. [17]). 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 definitions of the safety relations.[2]
- 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 denoting it. More precisely: if $\varphi(x, y_1, \ldots, y_n)$ is a formula such that $\forall y_1, \ldots, y_n \exists! x \varphi$ is provable, then there is a term $t(y_1, \ldots, y_n)$ such that $\varphi(y_1, \ldots, y_n, t(y_1, \ldots, y_n))$ is provable. Hence, there is no need for the procedure of extension by definitions, and introduction of new symbols is reduced to using *abbreviations*.

[1] The closest attempt I am aware of to develop a language for sets that employs static set terms can be found Sect. 5.1 of [7]. However, the construction there is rather complicated, and far remoted from actual mathematical practice. (The terms have the form: $\{t_{n+1} : x_0 C_0 t_0, x_1 C_1 t_1, \ldots, x_n C_n t_n \mid \varphi\}$, where each C_i is either $\in$ or $\subseteq$, φ is a formula, and $t_1, \ldots, t_n$ are terms such that $Fv(t_i) \cap \{x_1, \ldots, x_n\} \subseteq \{x_0, \ldots, x_{i-1}\}$). Moreover: the use of these terms does not have the two important features described below, and cannot serve as a basis for a framework of the type developed here.

[2] It should perhaps be noted that the idea that existence of sets $\{x \mid \varphi\}$ might be connected with absoluteness properties of φ occurs also (though with a very different formalization) in Ackermann's set theory [1], which turned out to be equivalent (once one adds regularity) to ZF [19]. The connections (if any) between Ackermann's approach and the present one are yet to be determined, and will be investigated in the future. (I am grateful to an anonymous referee for bringing Ackermann's set theory to my attention).

2 A Description of the General Framework

2.1 Languages

Officially, every set theory S has in our formal framework its own language $L(S)$. $L(S)$ is determined by the *safety relation* $\succ_S$ on which S is based. The sets of terms and formulas of $L(S)$ and $\succ_S$ are usually defined by a simultaneous recursion. For every S the clauses for $L(S)$ in this recursive definition are the following (where $Fv(exp)$ denotes the set of free variables of exp):

- Every variable is a term.
- The constant ω is a term.
- If x is a variable, and φ is a formula such that $\varphi \succ_S \{x\}$, then $\{x \mid \varphi\}$ is a term (and $Fv(\{x \mid \varphi\}) = Fv(\varphi) - \{x\}$).
- If t and s are terms then $t = s$ and $t \in s$ are atomic formulas.
- 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.

Note. We have included the constant ω in all our languages in order to be able to have in all of them closed terms for denoting constant sets (see e.g. the definition of $\emptyset$ in Sect. 3.1). However, in most of our systems nothing is assumed about ω and its interpretation. Only in systems that include the infinity axiom we put the constant ω (which is available anyway) to the further use of denoting the set whose existence is guaranteed by this axiom.

2.2 Logic

Basically, the logic we will use in most of our systems is the usual first-order logic with equality. One should note however the following differences/additions:

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: $\varphi[x \mapsto t] \rightarrow \exists x\varphi$ is valid for *every* term t which is free for x in φ).
2. The rule of α-conversion (change of bound variables) is included in the logic.
3. The substitution of equals for equals is allowed within any context (under the usual conditions concerning bound variables).
4. In analogy to the previous rule concerning identity of terms, we assume similar rule(s) allowing the substitution of a formula for an equivalent formula in any context in which the substitution makes sense. In particular, the following schema is valid whenever $\{x \mid \varphi\}$ and $\{x \mid \psi\}$ are legal terms:

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

2.3 Axioms

The main part of all our systems consists of the following axioms and axiom schemes (our version of the ideal calculus, 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 shall consider will differ only with respect to the power of their comprehension scheme. This, in turn, again depends only on the safety relation used by each. Hence also the differences in strength between the systems will mainly be due to the differences between their safety relations.
2. It is easy to see (see [4]) 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:

$$\forall z(z \in x \leftrightarrow z \in y) \to x = y$$

3. The first two axioms immediately entail the following two principles (where t is an arbitrary valid 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).

The Axiom of Choice. The full set theory ZFC has one more axiom that 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.

2.4 Safety Relations

As emphasized above, the core of each of our systems is the safety relation it employs. Now the idea of using such relations is due to the similarity (noted first in [4]) between issues of safety and domain independence in database theory ([2,24]), and issues of set-existence and absoluteness in set theory. This similarity allows us to apply in the context of set theories the purely syntactic approach to safety of formulas that has been developed in database theory.

From a logical point of view, a database of scheme $D = \{P_1, \ldots, P_n\}$ is just a given set of *finite* interpretations of the predicate symbols $P_1, \ldots, P_n$. A query language for such a database is an ordinary first-order language with equality, the signature of which includes $\{P_1, \ldots, P_n\}$. Ideally, every formula ψ of a query language can serve as a query. If ψ has free variables then the answer to ψ is the set of tuples which satisfy it in some intended structure, where the interpretations of $P_1, \ldots, P_n$ is given by the database. If ψ is closed then the answer to the query is either "yes" or "no" (which can be interpreted as $\{\emptyset\}$ and $\emptyset$, respectively). However, an answer to a query should be finite and computable, even if the intended domain is infinite. Hence only "safe" formulas, the answers to which always have these properties, should be used as queries. In fact, an even stronger property of formulas is usually taken to be crucial. Safe queries should be *domain independent* ([24,2]) in the following sense:

Definition 1. [3] *Let σ be a signature which has no function symbols, and whose set of predicate symbols includes $D = \{P_1, \ldots, P_n\}$. A query $\varphi(x_1 \ldots, x_n)$ in σ is called* D-d.i. (D-domain-independent) *if whenever S_1 and S_2 are structures for σ such that S_1 is a substructure of S_2, and the interpretations of $\{P_1, \ldots, P_n\}$ in S_1 and S_2 are identical, then for all $a_1 \in S_2, \ldots, a_n \in S_2$:*

$$S_2 \models \varphi(a_1, \ldots, a_n) \quad \leftrightarrow \quad a_1 \in S_1 \wedge \ldots \wedge a_n \in S_1 \wedge S_1 \models \varphi(a_1, \ldots, a_n)$$

Thus a domain-independent 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. Practical database query languages are designed so that only d.i. queries can be formulated in them. Unfortunately, it easily follows from *Trakhtenbrot's Theorem* (see [9]) that it is undecidable which formulas are d.i. (or "safe" in any other reasonable notion of safety of queries, like "finite and computable"). Therefore all commercial query languages (like SQL) allow to use as queries only formulas from some syntactically defined class of d.i. formulas. Many explicit proposals of decidable, syntactically defined classes of safe formulas have been made in the literature. Perhaps the simplest among them is the following class $SS(D)$ ("syntactically safe" formulas for a database scheme D) from [24] (originally designed for languages in which every term is either a variable or a constant):[4]

[3] This is a slight generalization of the usual definition ([24]), which applies only to free Herbrand structures which are generated by adding to σ some new set of constants.

[4] What we present below is both a generalization and a simplification of Ullman's original definition.

1. $P_i(t_1, \ldots, t_{n_i}) \in SS(D)$ in case P_i (of arity n_i) is in D.
2. $x = c$ and $c = x$ are in $SS(D)$ (where x is a variable and c is a constant).
3. $\varphi \vee \psi \in SS(D)$ if $\varphi \in SS(D)$, $\psi \in SS(D)$, and $Fv(\varphi) = Fv(\psi)$.
4. $\exists x\varphi \in SS(D)$ if $\varphi \in SS(D)$.
5. If $\varphi = \varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_k$ then $\varphi \in SS(D)$ if the following conditions are met:
   (a) For each $1 \leq i \leq k$, either φ_i is atomic, or φ_i is in $SS(D)$, or φ_i is a negation of a formula of either type.
   (b) Every free variable x of φ is limited in φ. This means that there exists $1 \leq i \leq k$ such that x is free in φ_i, and either $\varphi_i \in SS(D)$, or there exists y which is already limited in φ, and $\varphi_i \in \{x = y, y = x\}$.

There is one clause in this definition which is somewhat strange: the last one, which treats conjunction. The reason why this clause does not simply tell us (like in the case of disjunction) when a conjunction of *two* formulas is in $SS(D)$, is the desire to take into account the fact that once the value of y (say) is known, the formula $x = y$ becomes safe. In order to replace this problematic clause by a more concise one (which at the same time is more general) the formula *property* of d.i. was turned in [4] into the following *relation* between a formula φ and finite subsets of $Fv(\varphi)$:

Definition 2. *Let σ be as in Definition 1. A formula $\varphi(x_1, \ldots, x_n, y_1, \ldots, y_k)$ in σ is D-d.i. with respect to $\{x_1, \ldots, x_n\}$ if whenever S_1 and S_2 are structures as in Definition 1, then for all $a_1 \in S_2, \ldots, a_n \in S_2$ and $b_1 \in S_1, \ldots, b_k \in S_1$:*

$$S_2 \models \varphi(\overrightarrow{a}, \overrightarrow{b}) \;\leftrightarrow\; a_1 \in S_1 \wedge \ldots \wedge a_n \in S_1 \;\wedge S_1 \models \varphi(\overrightarrow{a}, \overrightarrow{b})$$

Obviously, a formula φ is D-d.i. iff it is D-d.i. with respect to $Fv(\varphi)$. On the other hand the formula $x = y$ is only partially D-d.i.: it is D-d.i. with respect to $\{x\}$ and $\{y\}$, but not with respect to $\{x, y\}$.

A particularly important observation is that a formula φ is D-d.i. with respect to $\emptyset$, if whenever S_1 and S_2 are structures as in Definition 1, then for all $b_1, \ldots, b_k \in S_1$, $S_2 \models \varphi(\overrightarrow{b}) \leftrightarrow S_1 \models \varphi(\overrightarrow{b})$. Such formulas may be called D-absolute. Obviously, this notion of D-absoluteness is closely related to the set-theoretical notion of absoluteness. However, as it is, it is not really a generalization of the notion used in set theory. In addition to $=$, the language of set theory has only one binary predicate symbol: $\in$. Now the notion of $\{\in\}$-absoluteness is useless (since if the interpretations of $\in$ in two standard models S_1 and S_2 of ZF are identical, then S_1 and S_2 are identical). The notion of $\emptyset$-absoluteness, in contrast, *is* identical to the most general notion of absoluteness as defined e.g. in [17] (p. 117), but that notion is of little use in set theory. Thus Δ_0-formulas are not $\emptyset$-absolute. Indeed, in order for Δ_0-formulas to be absolute for structures S_1 and S_2 (where S_1 is a substructure of S_2), we should assume that S_1 is a *transitive* substructure of S_2. This means that if b is an element of S_1, and $S_2 \models a \in b$, then a belongs to S_1, and $S_1 \models a \in b$. In other words: the formula $x \in y$ should be d.i. with respect to $\{x\}$ (but not with respect to $\{y\}$). In [4] and [6] this observation was used for developing a general framework for

domain independence and absoluteness, and it was shown that this framework has deep applications in computability theory.

The similarity between d.i. and absoluteness is also the crucial observation on which the present framework for set-theories is based. However, in order to exploit this similarity here we do not need the full general framework developed in [4,6]. It suffices to introduce the following general, abstract notion of a safety relation (which is based on Ullman's notion of syntactic safety, but its use is not confined to database theory):

Definition 3. *A relation $\succ$ between formulas φ and subsets of $Fv(\varphi)$ is a* safety relation *if it satisfies the following conditions:*

1. *If $\varphi \succ X$ then $X \subseteq Fv(\varphi)$.*
2. *If $\varphi \succ X$ and $Z \subseteq X$, then $\varphi \succ Z$.*
3. *If $\varphi \succ \{x_1, \ldots, x_n\}$ and $v_1, \ldots v_n$ are n distinct variables not occurring in φ, then $\varphi[x_1 \mapsto v_1, \ldots, x_n \mapsto v_n]$.*
4. *$\varphi \succ \emptyset$ if φ is atomic.*
5. *$t = x \succ \{x\}$ and $x = t \succ \{x\}$ if $x \notin Fv(t)$.*
6. *$\neg\varphi \succ \emptyset$ if $\varphi \succ \emptyset$.*
7. *$\varphi \vee \psi \succ X$ if $\varphi \succ X$ and $\psi \succ X$.*
8. *$\varphi \wedge \psi \succ X \cup Y$ if $\varphi \succ X$, $\psi \succ Y$, and $Y \cap Fv(\varphi) = \emptyset$.*
9. *$\exists y\varphi \succ X - \{y\}$ if $y \in X$ and $\varphi \succ X$.*

Note. Recall that we are taking $\wedge, \vee, \neg$ and $\exists$ as our primitives. Moreover: we take $\neg(\varphi \to \psi)$ as an abbreviation for $\varphi \wedge \neg\psi$, and $\forall x_1, \ldots, x_k\varphi$ as an abbreviation for $\neg\exists x_1, \ldots, x_k\neg\varphi$. This entails the following important property of "bounded quantification": *If $\succ$ is a safety relation, $\varphi \succ \{x_1, \ldots, x_n\}$, and $\psi \succ \emptyset$, then $\exists x_1 \ldots x_n(\varphi \wedge \psi) \succ \emptyset$ and $\forall x_1 \ldots x_n(\varphi \to \psi) \succ \emptyset$.* The latter can easily be generalized, and the generalization can be used for an alternative definition of safety relations in case the negation connective may be used only before atomic formulas, and the negation of φ, $\overline{\varphi}$, is inductively defined for complex formulas (a common procedure in proof theory): strengthen condition 4 above to $\varphi \succ \emptyset$ if φ is a literal, and replace condition 6 by: $\forall x_1 \ldots x_n\varphi \succ \emptyset$ if $\overline{\varphi} \succ \{x_1, \ldots, x_n\}$.

Examples

- For first order languages with equality, having no function symbols and no predicate symbols other than those in D, partial D-d.i. (Definition 2) is a safety relation. A syntactic counterpart $\succ$ directly corresponding to $SS(D)$ is inductively defined by using the clauses of Definition 3 and the assumption that $\varphi \succ Fv(\varphi)$ for every atomic formula φ of the form $P_i(t_1, \ldots, t_{n_i})$.
- Let L be the language of PA (Peano's Arithmetic), and let $\mathcal{N}$ be the standard model of PA. Define a relation $\succ_{\mathcal{N}}$ on L by: $\varphi(x_1, \ldots, x_n, y_1, \ldots, y_l) \succ_{\mathcal{N}} \{x_1, \ldots, x_n\}$ if the set $\{\langle k_1, \ldots, k_n\rangle \in \mathcal{N}^n \mid \varphi(k_1, \ldots, k_n, m_1, \ldots, m_l)\}$ is finite and computable (as a function of $m_1, \ldots, m_l$) for all $m_1, \ldots, m_l$ in $\mathcal{N}$.[5]

[5] In the case $l = 0$ an intentional meaning of "computable" is meant, but we shall not get into details here. See [4,6] for more details.

Then $\succ_{\mathcal{N}}$ is a safety relation, and $\varphi \succ_{\mathcal{N}} \emptyset$ iff φ defines a decidable predicate. A useful syntactic approximation $\succ_b$ of $\succ_{\mathcal{N}}$ can in this case inductively be defined by using the clauses of Definition 3 and the assumption that $x < t \succ_b x$ if $x \notin Fv(t)$. The set $\{\varphi \mid \varphi \succ_b \emptyset\}$ is a straightforward extension of Smullyan's set of Σ_0 formulas (see [23], P. 41), which can serve as a basis for the usual arithmetical hierarchy. It is interesting to note that a succinct inductive definition of $\succ_b$ can be given which is almost identical to that of the basic safety relation $\succ_{RST}$ of set theory (see Definition 5). The only difference is that the condition $x \in t \succ_b x$ in Definition 5 should be replaced by $x < t \succ_b x$.

Next we describe the way safety relations are used in our framework for set theories. The basic idea is that φ should be safe for $\{x\}$ in a set theory S iff the collection $\{x \mid \varphi\}$ is accepted as a set by S. This leads to the following definition:

Definition 4. *Let L be a language which has $\in$ among its binary predicate symbols.* An $\in$-safety relation *for L is a safety relation $\succ$ for L which satisfies the following condition:*

- $x \in t \succ \{x\}$ *if x is a variable such that $x \notin Fv(t)$.*

All the safety relations used in our framework are $\in$-safety relations.

3 The Rudimentary Set Theory *RST*

Our basic system is the one which corresponds to the minimal $\in$-safety relation:

Definition 5. *The relation $\succ_{RST}$ is inductively defined as follows:*

1. $\varphi \succ_{RST} \emptyset$ *if φ is atomic.*
2. $\varphi \succ_{RST} \{x\}$ *if $\varphi \in \{x = t, t = x, x \in t\}$, 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 $\varphi \succ_{RST} X$.*

It is easy to see that $\succ_{RST}$ is indeed an $\in$-safety relation. We denote by *RST* (Rudimentary Set Theory) the set theory it induces (within the framework described above). 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 $\vdash_{RST} \varphi \leftrightarrow \psi$ then $\vdash_{RST} x \in \{x \mid \varphi\} \leftrightarrow \psi$, and so $\vdash_{RST} \exists Z \forall x. x \in Z \leftrightarrow \psi$.

3.1 The Power of *RST*

In the language of *RST* we can introduce as *abbreviations* (rather than as extensions by definitions) 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:

- $\emptyset =_{Df} \{x \mid x \in \omega \wedge x \neq x\}$.
- $\{t_1, \ldots, t_n\} =_{Df} \{x \mid x = t_1 \vee \ldots \vee x = t_n\}$ (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).
- $\{\langle x_1, \ldots, x_n\rangle \mid \varphi\} =_{Df} \{z \mid \exists x_1 \ldots \exists x_n.\varphi \wedge z = \langle x_1, \ldots, x_n\rangle\}$, provided $\varphi \succ_{RST} \{x_1, \ldots, x_n\}$, 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).
- $\bigcap t =_{Df} \{x \mid \exists y(y \in t \wedge x \in y) \wedge \forall y(y \in t \rightarrow x \in y)\}$ (where x, y are new).

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*.

Lemma 1. *There is a formula $OP(z, x, y)$ in the basic language of RST (i.e.: without set terms) such that:*

1. $\vdash_{RST} OP(z, x, y) \leftrightarrow z = \langle x, y\rangle$
2. $OP(z, x, y) \succ_{RST} \{x, y\}$.

Proof: Let $Pa(z, x, y) \equiv_{Df} x \in z \wedge y \in z \wedge \forall w(w \in z \rightarrow w = x \vee w = y)$. Then $Pa(z, x, y) \succ_{RST} \{x, y\}$, and $\vdash_{RST} Pa(z, x, y) \leftrightarrow z = \{x, y\}$. Let $OP(z, x, y)$ be the formula $\exists u \exists v(Pa(z, u, v) \wedge Pa(u, x, x) \wedge Pa(v, x, y))$. □

With the help of OP we can define all the standard basic operations related to relations and functions. For example:

- $Dom(s) =_{Df} \{x \mid \exists z \exists y(z \in s \wedge OP(z, x, y)\}$
- $Rng(s) =_{Df} \{y \mid \exists z \exists x(z \in s \wedge OP(z, x, y)\}$
- $t \restriction s =_{Df} \{x \in t \mid \exists z \exists y OP(x, y, z) \wedge y \in s\}$

In RST we can also introduce as abbreviations the terms used in the λ-calculus for handling explicitly defined functions which are sets (except that our terms for functions should specify the domains of these functions, which should also be explicitly definable sets). Moreover: the reduction rules of the λ-calculus for these terms are easy theorems of RST. Thus the notation for λ-set and function application are introduced as follows:

- $\lambda x \in s.t =_{Df} \{\langle x, t\rangle \mid x \in s\}$ (where $x \notin Fv(s)$)
- $f(t) =_{Df} \bigcup Rng(f \restriction \{t\})$

(Note that $f(t)$ is defined for *every* f and t, but when f denotes a function F, and t denotes an element a in F's domain, then $f(t)$ indeed denotes the value of F at a.) We can easily check now that rules β and η obtain in RST:

- $\vdash_{RST} u \in s \to (\lambda x \in s.t)u = t[x \mapsto u]$ (if u is free for x in t).
- $\vdash_{RST} u \notin s \to (\lambda x \in s.t)u = \emptyset$ (if u is free for x in t).
- $\vdash_{RST} \lambda x \in s.t(x) = t \restriction s$ (in case $x \notin Fv(t)$).

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 (the proofs of which will be given in [6]).

Theorem 2

1. *If F is an n-ary rudimentary function*[6] *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 a rudimentary predicate then there is a formula φ such that $\varphi \succ_{RST} \emptyset$ and φ defines P.*

Theorem 3. *RST is equivalent to the system obtained from Gandy's "Basic Set Theory" BST ([12]) by the addition of the $\in-$induction schema.*

[6] The class of rudimentary set functions was introduced independently by Gandy ([12]) and Jensen ([15]). See also [8], Sect. IV.1.

3.2 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 6. *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 7. *Let T be an extension of RST such that $V \models T$.*

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 $\mathcal{M}$ of T:*

$$\forall y_1 \ldots \forall y_n . y_1 \in \mathcal{M} \wedge \ldots \wedge y_n \in \mathcal{M} \rightarrow t_{\mathcal{M}} = t$$

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 $\mathcal{M}$ of RST:*

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

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. [17]), while φ is T-absolute for $Fv(\varphi)$ iff it is domain-independent in the sense of database theory (see Definition 1) for transitive models of T.

Theorem 4

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

The proof is by a simultaneous induction on the complexity of t and φ.

4 Stronger Set Theories

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 first with the axioms that remain valid if we limit ourselves to hereditarily finite sets. We show that the addition of each of them to RST corresponds to adding to the definition of $\succ_{RST}$ a certain syntactic condition.

4.1 Basic ZF: The Full Separation and Replacement Schemes

Theorem 5. *Let T be an extension of RST, based on some safety relation $\succ_T$ which extends $\succ_{RST}$.*

1. *If $\succ_T$ satisfies the condition:*
   (Sep) $\varphi \succ_T \emptyset$ *for every formula* φ
   then the axiom schema of separation is derivable in T.
2. *If $\succ_T$ satisfies the condition:*
   (Rep) $\exists y\varphi \wedge \forall y(\varphi \to \psi) \succ_T X$ *if* $\psi \succ X$, *and* $X \cap Fv(\varphi) = \emptyset$.
   then the axiom schema of replacement is derivable in T.

Proof: In the presence of condition (Sep), $\{x \mid x \in z \wedge \varphi\}$ is a valid term for every φ, and this implies the separation schema.

Suppose now that $\succ_T$ Satisfies (Rep). The proof that the replacement schema is derivable in T is more difficult than in the previous case, because unlike the other comprehension axioms of ZF, the official formulation of replacement has the form of a *conditional*:

$$(\forall y \exists v \forall x(\varphi \Leftrightarrow x = v)) \Rightarrow (\exists Z \forall x.x \in Z \Leftrightarrow (\exists y.y \in w \wedge \varphi))$$

where $v, w, Z \notin Fv(\varphi)$. To prove this in T, let A be the formula $\forall x(\varphi \Leftrightarrow x = v)$. Reasoning in T, assume $\forall y \exists v A$ (this is the left hand side of the implication we want to prove). This and the definition of the formula A logically imply $(\exists v A \wedge \forall v(A \to x = v)) \Leftrightarrow \varphi$. But by (Rep), $\exists v A \wedge \forall v(A \to x = v) \succ_T \{x\}$. Hence $\exists y.y \in w \wedge (\exists v A \wedge \forall v(A \to x = v)) \succ_T \{x\}$. Thus the comprehension axiom of T implies: $\exists Z \forall x.x \in Z \Leftrightarrow (\exists y.y \in w \wedge (\exists v A \wedge \forall v(A \to x = v)))$. This and the above conclusion of $\forall y \exists v A$ together entail $\exists Z \forall x.x \in Z \Leftrightarrow (\exists y.y \in w \wedge \varphi)$. □

Definition 8

1. *The safety relation $\succ_{BZF}$ is obtained from $\succ_{RST}$ by replacing clauses* 1 *and* 3 *of its definition with (Sep) and (Rep).*
2. *The system BZF is defined like RST, using $\succ_{BZF}$ instead of $\succ_{RST}$.*

Note. Any formula φ is logically equivalent to $\exists y\varphi \wedge \forall y(\varphi \to \exists x.x = \omega)$, where y is a dummy variable. Hence (Sep) is superfluous in the presence of (Rep) (This corresponds to the well-known fact that separation is derivable from replacement). In particular, to get BZF it suffices to add to $\succ_{RST}$ only (Rep).

Theorem 6. *Let BZF^* be the system in the pure first-order fragment of the language of BZF (i.e. with no set terms) which is obtained from BZF by replacing its comprehension axiom with the following safe comprehension schema:*

$$(SCn) \qquad \exists Z(\forall x.x \in Z \Leftrightarrow \varphi)$$

where φ is in the language of BZF^*, $\varphi \succ_{BZF} \{x\}$, and $Z \notin Fv(\varphi)$. Let ZF^{--} be ZF without the powerset axiom and the infinity axiom. Then BZF, BZF^*, and ZF^{--} are all equivalent.[7]

[7] Note again (see the note in Sect. 2.1) that although ZF^{--} can talk about ω, as far as this theory is concerned, ω could be any set whatsoever.

Proof: Obviously, every theorem of BZF^* is also a theorem of BZF. That every theorem of ZF^{--} is a theorem of BZF^* can be shown exactly like in the proof of Theorem 5.

To complete the cycle, it remains to show that BZF is a conservative extension of ZF^{--}. For this we define recursively for every formula φ of BZF a translation $\varphi^{(I)}$ into the language of ZF^{--} such that $Fv(\varphi^{(I)}) = Fv(\varphi)$:

- If φ is an atomic formula in the language of ZF^{--} then $\varphi^{(I)} = \varphi$.
- Suppose φ is an atomic formula which contains a set term. Let $t = \{x \mid \psi\}$ (where $\psi \succ_{BZF} x$) be a maximal set term of φ. Define:

$$\varphi^{(I)} = \exists Z(\forall x(x \in Z \Leftrightarrow \psi^{(I)}) \wedge (\varphi[t \mapsto Z])^{(I)})$$

  where Z is a new variable, and $\varphi[t \mapsto Z]$ is the formula obtained from φ by replacing every occurrence of t in φ by Z.
- Let $(\varphi \wedge \psi)^{(I)} = (\varphi)^{(I)} \wedge (\psi)^{(I)}$, $(\exists x\varphi)^{(I)} = \exists x(\varphi)^{(I)}$ etc.

Next, we show how to express the safety relation $\succ_{BZF}$ within the language of ZF^{--}. From Lemma 1 it easily follows that there is a formula $B_n(x_1, \ldots, x_n, z)$ in the language of ZF^{--} such that $B_n(x_1, \ldots, x_n, z) \succ_{RST} \{x_1, \ldots, x_n\}$ and $B_n(x_1, \ldots, x_n, z)$ is equivalent in RST to $\langle x_1, \ldots, x_n\rangle \in z$. Let $set_{x_1,\ldots,x_n}\varphi$ be $(\varphi \to \varphi)$ for $n = 0$, $\exists Z\forall x_1 \ldots \forall x_n(B_n(x_1, \ldots, x_n, Z) \Leftrightarrow \varphi)$ for $n > 0$ (where $Z \notin Fv(\varphi)$) [8]. Let $Set_{x_1,\ldots,x_n}\varphi$ be the universal closure of $set_{x_1,\ldots,x_n}\varphi$. Note that $Set_x\varphi$ formalizes the application to φ of the comprehension principle. We show by induction on the structure of a formula φ of BZF that if $\varphi \succ_{BZF} \{x_1, \ldots, x_n\}$ then $Set_{x_1,\ldots,x_n}\varphi^{(I)}$ is a theorem of ZF^{--}.

1. The case $n = 0$ is trivial
2. (a) If t is a variable or a constant of BZF then
   - $set_x x = t$ and $set_x t = x$ follow from the pairing axiom.
   - $set_x x \in t$ is a logically valid formula.

   (b) If $t = \{y \mid \psi\}$ (where $\psi \succ_{BZF} y$) and $\varphi = p(x, t)$, where $p(x, t)$ is in $\{x = t, t = x, x \in t\}$, and $x \notin Fv(t)$ $(= Fv(\psi) - \{y\})$, then $\varphi^{(I)}$ is $\exists Z(\forall y(y \in Z \Leftrightarrow \psi^{(I)}) \wedge p(x, Z))$. By induction hypothesis for ψ we have $\vdash_{ZF^{--}} Set_y\psi^{(I)}$. This means that $\vdash_{ZF^{--}} \exists Z(\forall y(y \in Z \Leftrightarrow \psi^{(I)})$, and so $\vdash_{ZF^{--}} \exists!Z(\forall y(y \in Z \Leftrightarrow \psi^{(I)})$. By part (a) also $\vdash_{ZF^{--}} Set_x p(x, Z)$. Now it is easy to show that $(\exists!ZA \wedge \forall Z set_x B) \to set_x\exists Z(A \wedge B)$ is logically valid in case $x \notin Fv(A)$. This implies that $\vdash_{ZF^{--}} Set_x\varphi^{(I)}$.
3. $set_{x_1,\ldots,x_n}(\varphi \vee \psi)^{(I)}$ follows from $set_{x_1,\ldots,x_n}\varphi^{(I)}$ and $set_{x_1,\ldots,x_n}\psi^{(I)}$ by the axioms of union and pairing.

[8] This is a generalization of the notation $Set_x\varphi$ from [20], P. 240.

4. To simplify notation, assume that $Fv(\varphi) = \{x, z\}$, $Fv(\psi) = \{x, y, z\}$, and that $\varphi \succ_{BZF} \{x\}$, $\psi \succ_{BZF} \{y\}$ (and so $\varphi \wedge \psi \succ_{BZF} \{x, y\}$). By induction hypothesis, $\vdash_{ZF^{--}} Set_x\varphi^{(I)}$, and $\vdash_{ZF^{--}} Set_y\psi^{(I)}$. Reasoning in ZF^{--}, this means that there are sets Z(z) and W(x,z) such that $x \in Z(z) \Leftrightarrow \varphi^{(I)}$ and $y \in W(x, z) \Leftrightarrow \psi^{(I)}$. It follows that

$$\{\langle x, y\rangle \mid (\varphi \wedge \psi)^{(I)}\} = \bigcup_{x \in Z(z)} \{x\} \times W(x, z)$$

   $Set_{x,y}(\varphi \wedge \psi)^{(I)}$ follows therefore by the axioms of replacement and union, and the fact that the existence of Cartesian products is provable in ZF^{--}.

5. Deriving $Set_{X-\{y\}}\exists y\varphi^{(I)}$ in ZF^{--} from $Set_X\varphi^{(I)}$ is left to the reader.

6. Assume that $\vdash_{ZF^{--}} set_{x_1,\ldots,x_n}\psi^{(I)}$, and $\{x_1, \ldots, x_n\} \cap Fv(\varphi) = \emptyset$. We show that $\vdash_{ZF^{--}} set_{x_1,\ldots,x_n}(\exists y\varphi \wedge \forall y(\varphi \to \psi))^{(I)}$. This is immediate from the fact that if $\{x_1, \ldots, x_n\} \cap Fv(\varphi) = \emptyset$ then $\exists y \forall x_1 \ldots x_n((\exists y\varphi \wedge \forall y(\varphi \to \psi)) \to \psi)$ is logically valid [9], together with the following lemma:

   Lemma: Assume that $\{y_1, \ldots, y_k\} \cap Fv(\varphi) = \emptyset$, $\vdash_{ZF^{--}} set_{x_1,\ldots,x_n}\psi$ and $\exists y_1, \ldots, y_k \forall x_1, \ldots, x_n(\varphi \to \psi)$ is logically valid. Then $\vdash_{ZF^{--}} set_{x_1,\ldots,x_n}\varphi$.

   Proof of the Lemma: $\exists y_1, \ldots, y_k \forall x_1, \ldots, x_n(\varphi \to \psi)$ logically implies the formula $\exists y_1, \ldots, y_k \forall x_1, \ldots, x_n(\varphi \leftrightarrow .\psi \wedge \varphi)$. It is easy however to see that if $\{y_1 \ldots y_k\} \cap Fv(\varphi) = \emptyset$ then $Set_{x_1,\ldots,x_n}\varphi$ logically follows in first order logic from $Set_{x_1,\ldots,x_n}\phi$ and $\exists y_1 \ldots y_k \forall x_1 \ldots x_n(\varphi \leftrightarrow \phi)$. Hence we only need to prove that $set_{x_1,\ldots,x_n}(\psi \wedge \varphi)$ follows in ZF^{--} from $set_{x_1,\ldots,x_n}\psi$. This is immediate from the axiom of subsets.

Now we show that if $\vdash_{BZF} \varphi$ then $\vdash_{ZF^{--}} \varphi^{(I)}$. Since obviously $\varphi^{(I)} = \varphi$ in case φ is in the language of ZF^{--}, this will end the proof of the theorem. Now the inference rules are identical in the two systems, and our translation preserves applications of these rules. It suffices therefore to show that the translations of the comprehension axioms of BZF are theorems of ZF^{--}. Well, if $\varphi \succ_{BZF} \{x\}$ then the translation of the φ-instance of this schema is $\forall x(\exists Z(\forall x(x \in Z \leftrightarrow \varphi^{(I)}) \wedge x \in Z) \leftrightarrow \varphi^{(I)})$. It is easy to see that this formula follows in ZF^{--} from $Set_x\varphi^{(I)}$. The latter formula, in turn, is provable in ZF^{--} by what we have proved above (since $\varphi \succ_{BZF} \{x\}$).

This completes the proof of Theorem 6. □

As noted at the end of the introduction, in mathematical practice new symbols for relations and functions are regularly introduced in the course of developing a theory. This practice is formally based on the "extensions by definitions" procedure (see e.g. [20], Sect. 4.6). Now, while new relation symbols are introduced just as abbreviations for (usually) longer formulas, new function symbols are introduced in a dynamic way: once $\forall y_1, \ldots, y_n \exists! x\varphi$ is proved (where

[9] For the proof of the validity of this formula show that it follows from $\exists y_1 \ldots y_k\varphi$ as well as from $\neg\exists y_1 \ldots y_k\varphi$.

$Fv(\varphi) = \{y_1, \ldots, y_n, x\}$) then a new n-ary function symbol F_φ can conservatively be introduced, together with a new axiom: $\forall y_1, \ldots, y_n(\varphi[x \mapsto F_\varphi(y_1, \ldots, y_n)])$. Now a particularly remarkable property of BZF and its extensions is that this dynamic procedure is not needed for them. The required terms are available in advance, and every new function symbol we might wish to use may be introduced statically, as an abbreviation for an already existing term (in particular: any set which has an implicit definition in some extension of BZF has an explicit definition in that extension, using a set term):

Theorem 7. *For any formula φ of BZF such that $Fv(\varphi) = \{y_1, \ldots, y_n, x\})$, there exists a term t_φ of BZF such that $Fv(t_\varphi) = \{y_1, \ldots, y_n\}$, and*

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

Proof: Define $\iota x\varphi = \{z \mid \exists x\varphi \wedge \forall x(\varphi \rightarrow z \in x)\}$ (where z is a new variable, not occurring in φ). This is a valid term of BZF by the new clause in the definition of $\succ_{BZF}$. Now it can easily be proved that

$$\vdash_{BZF} \forall y_1, \ldots, y_n(\exists! x\varphi \rightarrow \forall x(\varphi \leftrightarrow x = \iota x\varphi))$$

It follows that $\iota x\varphi$ is a term t_φ as required. □

Corollary 2. *Every instance of the replacement schema (in the language of BZF) is derivable in BZF.*[10]

Proof: From the last theorem (and the definition of $\succ_{BZF}$) it follows that

$$\vdash_{BZF} \forall y \exists! x\varphi \rightarrow \forall x(\exists y.y \in w \wedge \varphi \leftrightarrow x \in \{x \mid \exists y.y \in w \wedge x = \iota x\varphi\}).$$

Note. $\iota x\varphi$ intuitively denotes the unique x such that φ, in case such exists. However, our $\iota x\varphi$ is always meaningful, and denotes $\emptyset$ if there is no set that satisfies φ, and the intersection of all the sets which satisfy φ in case there is more than one such set.

4.2 The Powerset Axiom

Theorem 8

1. *Let T be an extension of BZF, based on some safety relation $\succ_T$ which extends $\succ_{BZF}$. If $\succ_T$ satisfies the condition:*

   (Pow) $\forall y(y \in x \rightarrow \varphi) \succ (X - \{y\}) \cup \{x\}$ *if* $\varphi \succ X$, $y \in X$, *and* $x \notin Fv(\varphi)$.

   Then the powerset axiom is derivable in T.
2. *Let $\succ_{BZFP}$ be the safety relation obtained from $\succ_{BZF}$ by adding condition (Pow) to its definition, and let the system $BZFP$ be defined like RST, using $\succ_{BZFP}$ instead of $\succ_{RST}$. Then $BZFP$ is equivalent to $ZF - Inf$ (ZF without the infinity axiom).*

[10] This corollary provides a direct, short proof that BZF is an extension of ZF^{--}.

Proof: For the first part, note that in the presence of condition (Pow) the powerset axiom immediately follows from the facts that $y \in z \succ_{RST} y$, and that $P(z) = \{x \mid \forall y(y \in x \rightarrow y \in z)\}$. The proof of the second part is similar to that of Theorem 6. □

Another method (which may look more natural and is the one used in [5]) to add the power of the powerset axiom to the systems described above, is to extend the language by taking $\subseteq$ as an extra primitive binary relation symbol. A definition of a system which is equivalent to $ZF - Inf$ can then be obtained from the definition of BZF by making the following two changes:

- Replace $\succ_{BZF}$ with $\succ_{ZF-I}$, where $\succ_{ZF-I}$ is defined like $\succ_{BZF}$, but with one extra condition:

  $x \subseteq t \succ_{ZF-I} \{x\}$ if x is a variable, t is a term, and $x \notin Fv(t)$.
- Add the usual definition of $\subseteq$ in terms of $\in$ as an extra *axiom*:

$$\forall x \forall y(x \subseteq y \leftrightarrow \forall z(z \in x \rightarrow z \in y))$$

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 extensionality axioms:

(Ex1) $x \subseteq y \wedge y \subseteq x \rightarrow x = y$

(Ex2) $z \in x \wedge x \subseteq y \rightarrow z \in y$

(Ex3) $x \subseteq y \vee \exists z(z \in x \wedge z \notin y)$

4.3 The Axiom of Infinity

Finally we turn to the axiom of infinity — the only axiom that necessarily takes us out of the realm of (hereditarily) finite sets. As long as we take FOL (First-Order Logic) as the underlying logic, it seems impossible to incorporate it into our systems by just imposing new simple syntactic conditions on the safety relation. Instead the easiest and most natural way to add its power to the systems discussed so far, is to add to them *Peano's Axioms* as new axioms:

- $\emptyset \in \omega$
- $\forall x(x \in \omega \rightarrow S(x) \in \omega)$ (where $S(x)$ is defined like in Sect. 3.1)
- $\varphi[x \mapsto \emptyset] \wedge \forall x(\varphi \rightarrow \varphi[x \mapsto S(x)]) \rightarrow \forall x(x \in \omega \rightarrow \varphi)$

Note that because we are assuming the $\in$-induction schema, the above induction schema can actually be replaced by the following single axiom:

$$(\emptyset \in y \wedge \forall x(x \in y \rightarrow S(x) \in y)) \rightarrow \omega \subseteq y$$

Theorem 9. ([5]) *Let ZF^+ be the system obtained from RST by adding (Rep) and (Pow) to the definition of the safety relation, and the above Peano's axioms to the set of axioms. Then ZF^+ is equivalent to ZF.*

5 Using Transitive Closure Logic

Introducing the infinity axioms into a system is a major step that from a computational and proof-theoretical point of view takes us to a completely different level. As is clear from the form we gave to this introduction, it incorporates inductive reasoning into the systems. In order to introduce such reasoning already on the *logical* level, and to keep as far as possible the uniformity of our framework, it is most natural to use as the underlying logic a logic which is stronger than FOL, but still reasonably manageable from a computational point of view. Now in [3] it was argued that languages and logics with transitive closure operation TC provide the best framework for the formalization of mathematics. Following this suggestion seems particularly suitable in the present context, since with TC the difference between set theories which assume infinity, and set theories which are valid also in the universe of hereditarily finite sets, can again be reduced to differences in the underlying syntactic safety relations.

Definition 9. ([14,22]) Let L be a (first-order) language. The language L_{TC} is obtained from L by adding the following clause to the definition of a formula: If φ is a formula, x, y are distinct variables, and t, s are terms, then $(TC_{x,y}\varphi)(t, s)$ is a formula (in which every occurrence of x and y in φ is bound). The intended meaning of $(TC_{x,y}\varphi)(t, s)$ is the following "infinite disjunction": (where $w_1, w_2, \ldots$, are all new):

$$\varphi[x \mapsto s, y \mapsto t] \vee \exists w_1(\varphi[x \mapsto s, y \mapsto w_1] \wedge \varphi[x \mapsto w_1, y \mapsto t]) \vee$$
$$\vee \exists w_1 \exists w_2(\varphi[x \mapsto s, y \mapsto w_1] \wedge \varphi[x \mapsto w_1, y \mapsto w_2] \wedge \varphi[x \mapsto w_2, y \mapsto t]) \vee \ldots$$

The most important relevant facts shown in [3] concerning TC are:

1. If L contains a constant 0 and a (symbol for) a pairing function, then all types of finitary inductive definitions of relations and functions (as defined by Feferman in [11]) are available in L_{TC}.
2. Let V_0 be the smallest set including 0 and closed under the operation of pairing. Let U be the smallest set of first-order terms in a language with a constant for 0 and a function symbol for pairing. Let $\mathcal{PTC}^+$ be the smallest set of formulas which includes all formulas of the form $t = s$ for $t, s \in U$, and is closed under $\vee, \wedge$ and TC. Then a subset S of V_0 is recursively enumerable iff there exists a formula $\varphi(x)$ of $\mathcal{PTC}^+$ such that $S = \{x \in V_0 \mid \varphi(x)\}$.
3. By generalizing a particular case which has been used by Gentzen in [13], mathematical induction can be presented as a logical rule of languages with TC. Indeed, Using a Gentzen-type format, a general form of this principle can be formulated as follows:

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

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

Now if we are interested in set theories which are valid under the assumption that all sets are (hereditarily) finite, then the comprehension axiom remains valid if TC is included in the language, and the following clause is added to the definition of a safety relation in the extended language:

(TC-fin) $(TC_{x,y}\varphi)(x,y) \succ X$ if $\varphi \succ X$, and $\{x,y\} \subseteq X$.

On the other hand, for set theories which assume the existence of infinite sets the following stronger principle should be adopted:

(TC-inf) $(TC_{x,y}\varphi)(x,y) \succ X$ if $\varphi \succ X$, and $\{x,y\} \cap X \neq \emptyset$.

Let PST (for "Predicative Set Theory") be the extension of RST which has TC in its language, and is based on the safety relation $\succ_{PST}$ obtained from $\succ_{RST}$ by adding (TC-inf) as a new clause. Then the infinity axiom is derivable in PST, since one can introduce there the set $\mathcal{N}$ of natural numbers as follows:[11]

$$\mathcal{N} = \{x \mid x = \emptyset \vee \exists y.y = \emptyset \wedge (TC_{x,y}(x = S(y)))(x,y)\}$$

It is not difficult to see that PST still has the properties of RST described in Theorems 4 and 5.

Note. The set of valid formulas of TC-logic is not r.e. (or even arithmetical). Hence no sound and complete formal system for it is possible. It follows that PST and its extensions cannot be fully formalized, and so appropriate formal approximations (yet to be determined) of the underlying *logic* should be used.

References

1. Ackermann, W.: Zur Axiomatik der Mengenlehre. Mathematische Annalen 131, 336–345 (1956)
2. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Reading (1995)
3. Avron, A.: Transitive closure and the mechanization of mathematics. In: Kamareddine, F. (ed.) Thirty Five Years of Automating Mathematics, pp. 149–171. Kluwer Academic Publishers, Dordrecht (2003)
4. Avron, A.: Safety signatures for first-order languages and their applications. In: Hendricks, et al. (eds.) First-Order Logic Revisited, pp. 37–58. Logos Verlag, Berlin (2004)

[11] We could then adopt $\mathcal{N} = \omega$ as an axiom. Alternatively, we could omit the constant ω from the language, add the clause $x \neq x \succ_{PST} \{x\}$ to the definition of $\succ_{PST}$, define $\emptyset$ in PST as $\{x \mid x \neq x\}$, and then use the name ω rather than $\mathcal{N}$. Note also that if we start with BZF (or one of its extensions) as the basis, then $\emptyset$ (and so $\mathcal{N}$) may be defined in the language without the constant ω by the closed term:

$$\emptyset =_{DF} \{x \mid (\exists y.y = y) \wedge \forall y(y = y \rightarrow x \in y)\}.$$

5. Avron, A.: Formalizing set theory as it is actually used. In: Asperti, A., Bancerek, G., Trybulec, A. (eds.) MKM 2004. LNCS, vol. 3119, pp. 32–43. Springer, Heidelberg (2004)
6. Avron, A.: Constructibility and decidability versus domain independence and absluteness. Theoretical Computer Science (2007), doi:10.1016/j.tcs.2007.12.008
7. Cantone, D., Omodeo, E., Policriti, A.: Set Theory for Computing. Springer, Heidelberg (2001)
8. Devlin, K.J.: Constructibility. Perspectives in Mathematical Logic. Springer, Heidelberg (1984)
9. Di Paola, R.A.: The recursive unsolvability of the decision problem for the class of definite formulas. J. ACM 16(2), 324–327 (1969)
10. Fraenkel, A., Bar-Hillel, Y., Levy, A.: Foundations of Set Theory. North-Holland, Amsterdam (1973)
11. Feferman, S.: Finitary inductively presented logics. In: Logic Colloquium 1988, pp. 191–220. North-Holland, Amsterdam (1989)
12. Gandy, R.O.: Set-theoretic functions for elementary syntax. In: Axiomatic Set Theory, Part 2, pp. 103–126. AMS, Providence, Rhode Island (1974)
13. Gentzen, G.: Neue fassung des widerspruchsfreiheitsbeweises für die reine zahlentheorie. Forschungen zur Logik, N.S. (4), 19–44 (1938)
14. Immerman, N.: Languages which capture complexity classes. In: 15th Symposium on Theory of Computing, Association for Computing Machinery, pp. 347–354 (1983)
15. Jensen, R.B.: The fine structure of the constructible hierarchy. Annals of Mathematical Logic 4, 229–308 (1972)
16. Hallett, M.: Cantorian Set Theory and Limitation of Size. Clarendon Press, Oxford (1984)
17. Kunen, K.: Set Theory, An Introduction to Independence Proofs. North-Holland, Amsterdam (1980)
18. Levy, A.: Basic Set Theory. Springer, Heidelberg (1979)
19. Reinhardt, W.R.: Ackermann's set theory Equals ZF. Annals of Mathematical Logic 2, 189–249 (1970)
20. Shoenfield, J.R.: Mathematical Logic. Addison-Wesley, Reading (1967)
21. Shoenfield, J.R.: Axioms of set theory. In: Barwise, J. (ed.) Handbook of Mathematical Logic, North-Holland, Amsterdam (1977)
22. Shapiro, S.: Foundations Without Foundationalism: A Case for Second-order Logic. Oxford University Press, Oxford (1991)
23. Smullyan, R.M.: The Incompleteness Theorems. Oxford University Press, Oxford (1992)
24. Ullman, J.D.: Principles of Database and Knowledge-Base Systems. Computer Science Press (1988)