A 'theory' mechanism for a proof-verifier based on first-order set theory ⋆

Eugenio G. Omodeo[1] and Jacob T. Schwartz[2]

[1] University of L'Aquila, Dipartimento di Informatica omodeo@di.univaq.it
[2] University of New York, Department of Computer Science, Courant Institute of Mathematical Sciences schwartz@cs.nyu.edu

We often need to associate some highly compound meaning with a symbol. Such a symbol serves us as a kind of container carrying this meaning, always with the understanding that it can be opened if we need its content.
(Translated from [12, pp. 101–102])

Abstract. We propose classical set theory as the core of an automated proof-verifier and outline a version of it, designed to assist in proof development, which is indefinitely expansible with function symbols generated by Skolemization and embodies a modularization mechanism named 'theory'. Through several examples, centered on the finite summation operation, we illustrate the potential utility in large-scale proof-development of the 'theory' mechanism: utility which stems in part from the power of the underlying set theory and in part from Skolemization.

Key words: Proof-verification technology, set theory, proof modularization.

1 Introduction

Set theory is highly versatile and possesses great expressive power. One can readily find terse set-theoretic equivalents of established mathematical notions and express theorems in purely set-theoretic terms.

Checking any deep fact (say the Cauchy integral theorem) using a proof-verifier requires a large number of logical statements to be fed into the system. These must formalize a line of reasoning that leads from bare set rudiments to the specialized topic of interest (say, functional analysis) and then to a target theorem. Such an enterprise can only be managed effectively if suitable modularization constructs are available.

This paper outlines a version of the Zermelo-Fraenkel theory designed to assist in automated proof-verification of mathematical theorems. This system incorporates a technical notion of "theory" designed, for large-scale proof-development, to play a role similar to the notion of object class in large-scale programming. Such a mechanism can be very useful for "proof-engineering".

⋆ E.G. Omodeo enjoyed a Short-term mobility grant of the Italian National Research Council (CNR) enabling him to stay at the University of New York during the preparation of this work.

The theories we propose, like procedures in a programming language, have lists of formal parameters. Each "theory" requires its parameters to meet a set of assumptions. When "applied" to a list of actual parameters that have been shown to meet the assumptions, a theory will instantiate several additional "output" set, predicate, and function symbols, and then supply a list of theorems initially proved explicitly by the user inside the theory itself. These theorems will generally involve the new symbols.

Such use of "theories" and their application adds a touch of second-order logic capability to the first-order system which we describe. Since set theory has full multi-tier power, this should be all the second-order capability that is needed.

We illustrate the usefulness of the proposed theory notion via examples ranging from mere "utilities" (e.g. the specification of ordered pairs and associated projections, and the thinning of a binary predicate into a global single-valued map) to an example which characterizes a very flexible recursive definition scheme. As an application of this latter scheme, we outline a proof that a finite summation operation which is insensitive to operand rearrangement and grouping can be associated with any commutative-associative operation. This is an intuitively obvious fact (seldom, if ever, proved explicitly in algebra texts), but nevertheless it must be verified in a fully formalized context. Even this task can become unnecessarily challenging without an appropriate set-theoretic support, or without the ability to indefinitely extend the formal language with new Skolem symbols such as those resulting from "theory" invocations.

Our provisional assessment of the number of "proofware" lines necessary to reach the Cauchy integral theorem in a system like the one which we outline is 20–30 thousand statements.

2 Set theory as the core of a proof-verifier

A fully satisfactory formal logical system should be able to digest 'the whole of mathematics', as this develops by progressive extension of mathematics-like reasoning to new domains of thought. To avoid continual reworking of foundations, one wants the formal system taken as basic to remain unchanged, or at any rate to change only by extension as such efforts progress. In any fundamentally new area work and language will initially be controlled more by guiding intuitions than by entirely precise formal rules, as when Euclid and his predecessors first realized that the intuitive properties of geometric figures in 2 and 3 dimensions, and also some familiar properties of whole numbers, could be covered by modes of reasoning more precise than those used in everyday life. But mathematical developments during the last two centuries have reduced the intuitive content of geometry, arithmetic, and calculus ('analysis') in set-theoretic terms. The geometric notion of 'space' maps into 'set of all pairs (or triples) of real numbers', allowing consideration of the 'set of all n-tuples of real numbers' as 'n-dimensional space', and of more general related constructs as 'infinite dimensional' and 'functional' spaces. The 'figures' originally studied in geometry map,

via the 'locus' concept, into sets of such pairs, triples, etc. Dedekind reduced 'real number x' to 'set x of rational numbers, bounded above, such that every rational not in x is larger than every rational in x'. To eliminate everything but set theory from the formal foundations of mathematics, it only remained (since 'fractions' can be seen as pairs of numbers) to reduce the notion of 'integer' to set-theoretic terms. This was done by Cantor and Frege: an integer is the class of all finite sets in 1-1 correspondence with any one such set. Subsequently Kolmogorov modeled 'random' variables as functions defined on an implicit set-theoretic measure space, and Laurent Schwartz interpreted the initially puzzling 'delta functions' in terms of a broader notion of generalized function systematically defined in set-theoretic terms. So all of these concepts can be digested without forcing any adjustment of the set-theoretic foundation constructed for arithmetic, analysis, and geometry. This foundation also supports all the more abstract mathematical constructions elaborated in such 20th century fields as topology, abstract algebra, and category theory. Indeed, these were expressed set-theoretically from their inception. So (if we ignore a few ongoing explorations whose significance remains to be determined) set theory currently stands as a comfortable and universal basis for the whole of mathematics—cf. [5].

It can even be said that set theory captures a set of reality-derived intuitions more fundamental than such basic mathematical ideas as that of number. Arithmetic would be very different if the real-world process of counting did not return the same result each time a set of objects was counted, or if a subset of a finite set S of objects proved to have a larger count than S. So, even though Peano showed how to characterize the integers and derive many of their properties using axioms free of any explicit set-theoretic content, his approach robs the integers of much of their intuitive significance, since in his reduced context they cannot be used to count anything. For this and the other reasons listed above, we prefer to work with a thoroughly set-theoretic formalism, contrived to mimic the language and procedures of standard mathematics closely.

3 Set theory in a nutshell

Set theory is based on the handful of very powerful ideas summarized below. All notions and notation are more or less standard (cf. [16]).[1]

– The dyadic *Boolean operations* $\cap, \setminus, \cup$ are available, and there is a *null set*, $\emptyset$, devoid of elements. The *membership* relation $\in$ is available, and set *nesting* is made possible via the singleton operation $X \mapsto \{X\}$. Derived from this, we have single-element addition and removal, and useful increment/decrement operations:

$$X \text{ with } Y := X \cup \{Y\}, \quad X \text{ less } Y := X \setminus \{Y\}, \quad \mathsf{next}(X) := X \text{ with } X.$$

[1] As a notational convenience, we usually omit writing universal quantifiers at the beginning of a sentence, denoting the variables which are ruled by these understood quantifiers by single uppercase Italic letters.

Unordered lists $\{t_1, \ldots, t_n\}$ and ordered tuples $[t_1, \ldots, t_n]$ are definable too: in particular, $\{X_1, \ldots, X_n\} := \{X_1\} \cup \cdots \cup \{X_n\}$.

– *'Sets whose elements are the same are identical':* Following a step $\ell \neq r$ in a proof, one can introduce a new constant b subject to the condition $b \in \ell \;\leftrightarrow\; b \notin r$; no subsequent conclusions where b does not appear will depend on this condition. Negated set inclusion $\not\subseteq$ can be treated similarly, since $X \subseteq Y := X \setminus Y = \emptyset$.

– *Global choice:* We use an operation **arb** which, from any non-null set X, deterministically extracts an element which does not intersect X. Assuming $\mathbf{arb}\,\emptyset = \emptyset$ for definiteness, this means that

$$\mathbf{arb}\,X \in \mathsf{next}(X) \;\&\; X \cap \mathbf{arb}\,X = \emptyset$$

for all X.

– *Set-formation:* By (possibly transfinite) element- or subset-iteration over the sets represented by the terms $t_0, t_1 \equiv t_1(x_0), ..., t_n \equiv t_n(x_0, ..., x_{n-1})$, we can form the set

$$\{\, e \,:\, x_0\, C_0\, t_0,\; x_1\, C_1\, t_1, \ldots, x_n\, C_n\, t_n \mid \varphi \,\},$$

where each C_i is either $\in$ or $\subseteq$, and where $e \equiv e(x_0, \ldots, x_n)$ and $\varphi \equiv \varphi(x_0, \ldots, x_n)$ are a set-term and a condition in which the p.w. distinct variables x_i can occur free (similarly, each t_{j+1} may involve $x_0, \ldots, x_j$).
Many operations are readily definable using setformers, e.g.

$$\bigcup Y := \{\, x_2 \,:\, x_1 \in Y, x_2 \in x_1 \,\}, \quad Y \times Z := \{\, [x_1, x_2] \,:\, x_1 \in Y,\; x_2 \in Z \,\},$$
$$\mathcal{P}(Y) := \{\, x \,:\, x \subseteq Y \,\}, \quad \mathsf{pred}(X) := \mathbf{arb}\,\{\, y \in X \mid \mathsf{next}(y) = X \,\},$$

where if the condition φ is omitted it is understood to be **true**, and if the term e is omitted it is understood to be the same as the first variable inside the braces.

– *$\in$-recursion:* ("Transfinite") recursion over the elements of any set allows one to introduce global set operations; e.g.,

$$\begin{aligned} \mathsf{Ult\_membs}(S) \;&:=\; S \cup \bigcup\{\, \mathsf{Ult\_membs}(x) \,:\, x \in S \,\} \text{ and} \\ \mathsf{rank}(S) \;&:=\; \bigcup\{\, \mathsf{next}\big(\,\mathsf{rank}(x)\,\big) \,:\, x \in S \,\}, \end{aligned}$$

which respectively give the set of all "ultimate members" (i.e. elements, elements of elements, etc.) of S and the maximum "depth of nesting" of sets inside S.

– *'Infinite sets exist':* There is at least one **s_inf** satisfying

$$\mathbf{s\_inf} \neq \emptyset \;\&\; (\forall\, x \;\in\; \mathbf{s\_inf})(\{x\} \in \mathbf{s\_inf}),$$

so that the p.w. distinct elements $b, \{b\}, \{\{b\}\}, \{\{\{b\}\}\}, \ldots$ belong to **s_inf** for each b in **s_inf**.

The historical controversies concerning the *choice* and *replacement* axioms of set theory are all hidden in our use of setformers and in our ability, after a statement of the form $\exists\, y\, \psi(X_1, \ldots, X_n, y)$ has been proved, to introduce a Skolem function $f(X_1, \ldots, X_n)$ satisfying the condition $\psi\big(\,X_1, \ldots, X_n, f(X_1, \ldots, X_n)\,\big)$.

In particular, combined use of **arb** and of the setformer construct lets us write the choice set of any set X of non-null pairwise disjoint sets simply as $\{\, \mathbf{arb}\, y \,:\, y \in X \,\}$.[2]

To appreciate the power of the above formal language, consider von Neumann's elegant definition of the predicate 'X is a (possibly transfinite) ordinal', and the characterization of $\mathbb{R}$, the set of real numbers, as the set of Dedekind cuts (cf. [17]):

$$\begin{aligned} \mathsf{Ord}(X) \;&:=\; X \subseteq \mathcal{P}(X) \,\&\, (\forall\, y, z \in X)(y \in z \,\vee\, y = z \,\vee\, z \in y)\,, \\ \mathbb{R} \;&:=\; \big\{\, c \subseteq \mathbb{Q} \mid (\forall\, y \in c)(\exists\, z \in c)(y < z) \,\& \\ &\qquad\qquad (\forall\, y \in c)(\forall\, z \in \mathbb{Q})(z < y \;\rightarrow\; z \in c) \,\big\} \setminus \{\emptyset, \mathbb{Q}\}; \end{aligned}$$

here the ordered field $\mathbb{Q}, <$ of rational numbers is assumed to have been defined before $\mathbb{R}$.[3]

4 Theories in action: First examples

Here is one of the most obvious theories one can think of:

THEORY ordered_pair()
==>(opair, car, cdr)
car(opair(X, Y)) $= X$
cdr(opair(X, Y)) $= Y$
opair(X, Y) = opair(U, V) $\rightarrow$ $X = U \,\&\, Y = V$
END ordered_pair.

This **THEORY** has no input parameters and no assumptions, and returns three global functions: a pairing function and its projections. To start its construction, the user simply has to

SUPPOSE_THEORY ordered_pair()
==>
END ordered_pair,

then to **ENTER_THEORY** ordered_pair, and next to define e.g.

$$\begin{aligned} \mathsf{opair}(X, Y) \;&:=\; \big\{\, \{X\}, \big\{\, \{X\}, \{Y, \{Y\}\} \,\big\} \,\big\}\,, \\ \mathsf{car}(P) \;&:=\; \mathbf{arb}\, \mathbf{arb}\, P\,, \\ \mathsf{cdr}(P) \;&:=\; \mathsf{car}\big(\, \mathbf{arb}\, (P \setminus \{\mathbf{arb}\, P\}) \;\setminus\; \{\mathbf{arb}\, P\} \,\big)\,. \end{aligned}$$

[2] Cf. [18, p. 177]. Even in the more basic framework of first-order predicate calculus, the availability of choice constructs can be highly desirable, cf. [1].

[3] For an alternative definition of real numbers which works very well too, see E.A. Bishop's adaptation of Cauchy's construction of $\mathbb{R}$ in [2, pp. 291–297].

This makes it possible to prove such intermediate lemmas as

$$\begin{array}{l}
\mathbf{arb}\,\{U\} \;=\; U\,, \\
V \in Z \;\rightarrow\; \mathbf{arb}\,\{V,Z\} = V\,, \\
\mathsf{car}\big(\,\big\{\,\{X\},\big\{\,\{X\},W\,\big\}\,\big\}\,\big) \;=\; X\,, \\
\mathbf{arb}\;\mathsf{opair}(X,Y) \;=\; \{X\}\,, \\
\mathsf{cdr}\big(\,\mathsf{opair}(X,Y)\,\big) \;=\; \mathsf{car}\big(\,\big\{\,\{Y,\{Y\}\,\}\,\big\}\,\big) = Y\,.
\end{array}$$

Once these intermediate results have been used to prove the three theorems listed earlier, the user can indicate that they are the ones he wants to be externally visible, and that the return-parameter list consists of opair, car, cdr (the detailed definitions of these symbols, as well as the intermediate lemmas, have hardly any significance outside the **THEORY** itself[4]). Then, after re-entering the main **THEORY**, which is set_theory, the user can

$$\begin{array}{l}
\mathbf{APPLY}(\mathsf{opair},\,\mathsf{head},\,\mathsf{tail})\;\mathsf{ordered_pair}()\;\Longrightarrow \\
\quad \mathsf{head}\big(\,\mathsf{opair}(X,Y)\,\big) = X \\
\quad \mathsf{tail}\big(\,\mathsf{opair}(X,Y)\,\big) = Y \\
\quad \mathsf{opair}(X,Y) = \mathsf{opair}(U,V) \;\rightarrow\; X = U \,\&\, Y = V,
\end{array}$$

thus importing the three theorems into the main proof level. As written, this application also changes the designations 'car' and 'cdr' into 'head' and 'tail'.

Fig.1 shows how to take advantage of the functions just introduced to define notions related to maps that will be needed later on.[5]

$$\begin{array}{lll}
\mathsf{is_map}(F) & := & F = \{[\mathsf{head}(x),\mathsf{tail}(x)] \,:\, x \in F\} \\
\mathsf{Svm}(F) & := & \mathsf{is_map}(F)\;\&\;(\forall\, x,y \in F)\big(\,\mathsf{head}(x) = \mathsf{head}(y) \;\rightarrow\; x = y\,\big) \\
\mathsf{1_1_map}(F) & := & \mathsf{Svm}(F)\;\&\;(\forall\, x,y \in F)\big(\,\mathsf{tail}(x) = \mathsf{tail}(y) \;\rightarrow\; x = y\,\big) \\
F^{-1} & := & \{[\mathsf{tail}(x),\mathsf{head}(x)] \,:\, x \in F\}
\end{array}$$

$$\begin{array}{lll}
\mathsf{domain}(F) := \{\mathsf{head}(x) \,:\, x \in F\} & \qquad & \mathsf{range}(F) := \{\mathsf{tail}(x) \,:\, x \in F\} \\
F\{X\} := \{\, y \in \mathsf{range}(F) \mid [X,y] \in F\,\} & & F_{|S} := F \cap \big(\,S \times \mathsf{range}(F)\,\big) \\
\mathsf{Finite}(S) := \neg\exists\, f\big(\,\mathsf{1_1_map}(f)\;\&\;S = \mathsf{domain}(f) \neq \mathsf{range}(f) \subseteq S\,\big) &&
\end{array}$$

Fig. 1. Notions related to maps, single-valued maps, and 1-1 maps

For another simple example, suppose that the theory

$$\begin{array}{l}
\mathbf{THEORY}\;\mathsf{setformer0}(\mathsf{e},\mathsf{s},\mathsf{p}) \\
\Longrightarrow \\
\quad \mathsf{s} \neq \emptyset \;\rightarrow\; \{\,\mathsf{e}(x) \,:\, x \in \mathsf{s}\,\} \neq \emptyset \\
\quad \{\,x \in \mathsf{s} \mid \mathsf{p}(x)\,\} \neq \emptyset \;\rightarrow\; \{\,\mathsf{e}(x) \,:\, x \in \mathsf{s} \mid \mathsf{p}(x)\,\} \neq \emptyset \\
\mathbf{END}\;\mathsf{setformer0}
\end{array}$$

[4] A similar remark on Kuratowski's encoding of an ordered pair as a set of the form $\{\{x,y\},\{x\}\}$ is made in [14, pp. 50–51].

[5] We subsequently return to the notation $[X,Y]$ for $\mathsf{opair}(X,Y)$.

has been proved, but that its user subsequently realizes that the reverse implications could be helpful too; and that the formulae

$$\mathsf{s} \subseteq T \;\rightarrow\; \{\, \mathsf{e}(x) \,:\, x \in \mathsf{s} \mid \mathsf{p}(x) \,\} \subseteq \{\, \mathsf{e}(x) \,:\, x \in T \mid \mathsf{p}(x) \,\}\,,$$
$$\mathsf{s} \subseteq T \;\&\; (\forall\, x \in T \setminus \mathsf{s}) \neg\, \mathsf{p}(x) \;\rightarrow\; \{\, \mathsf{e}(x) \,:\, x \in \mathsf{s} \mid \mathsf{p}(x) \,\} = \{\, \mathsf{e}(x) \,:\, x \in T \mid \mathsf{p}(x) \,\}$$

are also needed. He can then re-enter the **THEORY** setformer0, strengthen the implications already proved into bi-implications, and add the new results: of course he must then supply proofs of the new facts.

Our next sample **THEORY** receives as input a predicate $\mathsf{P} \equiv \mathsf{P}(X, V)$ and an "exception" function $\mathsf{xcp} \equiv \mathsf{xcp}(X)$; it returns a global function $\mathsf{img} \equiv \mathsf{img}(X)$ which, when possible, associates with its argument X some Y such that $\mathsf{P}(X, Y)$ holds, and otherwise associates with X the "fictitious" image $\mathsf{xcp}(X)$. The **THEORY** has an assumption, intended to guarantee non-ambiguity of the fictitious value:

THEORY fcn_from_pred(P, xcp)
  $\neg\; \mathsf{P}\big(\, X, \mathsf{xcp}(X) \,\big)$ -- *convenient "guard"*
==>(img)
  $\mathsf{img}(X) \neq \mathsf{xcp}(X) \;\leftrightarrow\; \exists\, v\; \mathsf{P}(X, v)$
  $\mathsf{P}(X, V) \;\rightarrow\; \mathsf{P}\big(\, X, \mathsf{img}(X) \,\big)$
END fcn_from_pred.

To construct this **THEORY** from its assumption, the user can simply define

$$\mathsf{img}(X) \;:=\; \textbf{if}\; \mathsf{P}\big(\, X, \mathsf{try}(X) \,\big)\; \textbf{then}\; \mathsf{try}(X)\; \textbf{else}\; \mathsf{xcp}(X)\; \textbf{end if}\,,$$

where try results from Skolemization of the valid first-order formula

$$\exists\, y\, \forall\, v \big(\, \mathsf{P}(X, v) \;\rightarrow\; \mathsf{P}(X, y) \,\big)\,,$$

after which the proofs of the theorems of fcn_from_pred pose no problems.

As an easy example of the use of this **THEORY**, note that it can be invoked in the special form

$$\textbf{APPLY}(\mathsf{img})\; \mathsf{fcn\_from\_pred}\big(\, \mathsf{P}(X, Y) \mapsto Y \in X \;\&\; Q(Y),\; \mathsf{xcp}(X) \mapsto X \,\big) ==> \cdots$$

for any monadic predicate Q (because $\in$ is acyclic); without the condition $Y \in X$ such an invocation would instead result in an error indication, except in the uninteresting case in which one has proved that $\forall\, x\, \neg\, Q(x)$.

Here is a slightly more elaborate example of a familiar **THEORY**:

THEORY equivalence_classes(s, Eq)
  $(\forall\, x \;\in\; \mathsf{s}) \big(\, \mathsf{Eq}(x, x) \,\big)$
  $(\forall\, x, y, z \;\in\; \mathsf{s}) \big(\, \mathsf{Eq}(x, y) \;\rightarrow\; \big(\, \mathsf{Eq}(y, z) \;\leftrightarrow\; \mathsf{Eq}(z, x) \,\big) \,\big)$
==>(quot, cl_of) -- *"quotient"-set and globalized "canonical embedding"*
  $(\forall\, x, y \;\in\; \mathsf{s}) \big(\, \mathsf{Eq}(x, y) \;\leftrightarrow\; \mathsf{Eq}(y, x) \,\big)$

$(\forall\, x \ \in\ \mathsf{s})\big(\ \mathsf{cl\_of}(x) \in \mathsf{quot}\ \big)$
$(\forall\, b \ \in\ \mathsf{quot})\big(\ \mathbf{arb}\ b \in \mathsf{s}\ \&\ \mathsf{cl\_of}(\mathbf{arb}\ b) = b\ \big)$
$(\forall\, y \ \in\ \mathsf{s})\big(\ \mathsf{Eq}(x, y)\ \leftrightarrow\ \mathsf{cl\_of}(x) = \mathsf{cl\_of}(y)\ \big)$
END equivalence_classes.

Suppose that this **THEORY** has been established, and that $\mathbb{N}, \mathbb{Z}$, and the multiplication operation $*$ have been defined already, where $\mathbb{N}$ is the set of natural numbers, and $\mathbb{Z}$, intended to be the set of signed integers, is defined (somewhat arbitrarily) as

$$\mathbb{Z} := \{[n, m]\ :\ n, m \in \mathbb{N} \mid n = 0\ \vee\ m = 0\}\,.$$

Here the position of 0 in a pair serves as a sign indication, and the restriction of $*$ to $\mathbb{Z} \times \mathbb{Z}$ is integer multiplication (but actually, $x * y$ is always defined, whether or not $x, y \in \mathbb{Z}$). Then the set Fr of fractions and the set $\mathbb{Q}$ of rational numbers can be defined as follows:

$\mathsf{Fr} := \{\ [x, y]\ :\ x, y \in \mathbb{Z} \mid y \neq [0, 0]\ \}\,,$
$\mathsf{Same\_frac}(F, G) := \big(\ \mathsf{head}(F) * \mathsf{tail}(G) = \mathsf{tail}(F) * \mathsf{head}(G)\ \big),$
$\mathbf{APPLY}(\mathbb{Q}, \mathsf{Fr\_to\_}\mathbb{Q})\ \mathsf{equivalence\_classes}\big(\ \mathsf{s} \mapsto \mathsf{Fr},$
$\mathsf{Eq}(F, G) \mapsto \mathsf{Same\_frac}(F, G)\ \big) ==> \cdots$

Before **APPLY** can be invoked, one must prove that the restriction of $\mathsf{Same\_frac}$ to Fr meets the **THEORY** assumptions, i.e. it is an equivalence relation. Then the system will not simply return the two new symbols $\mathbb{Q}$ and $\mathsf{Fr\_to\_}\mathbb{Q}$, but will provide theorems insuring that these represent the standard equivalence-class reduction $\mathsf{Fr/Same\_frac}$ and the canonical embedding of Fr into this quotient. Note as a curiosity —which however hints at the type of hiding implicit in the **THEORY** mechanism— that a $\mathbb{Q}$ satisfying the conclusions of the **THEORY** is not actually forced to be the standard partition of Fr but can consist of singletons or even of supersets of the equivalence classes (which is harmless).

5 A final case study: Finite summation

Consider the operation $\Sigma(F)$ or, more explicitly,

$$\sum_{x \in \mathsf{domain}(F)} \ \sum_{[x,y] \in F} y$$

available for any *finite* map F (and in particular when $\mathsf{domain}(F) = d \in \mathbb{N}$, so that $x \in d$ amounts to saying that $x = 0, 1, \ldots, d-1$) such that $\mathsf{range}(F) \subseteq \mathsf{abel}$, where abel is a set on which a given operation $\mathsf{+}$ is associative and commutative and has a unit element u. Most of this is captured formally by the following **THEORY**:

THEORY sigma_add(abel, +, u)
$(\forall x, y \in \mathsf{abel})(x{+}y \in \mathsf{abel}\ \&$ -- *closure w.r.t. ...*
$x{+}y = y{+}x)$ -- *... commutative operation*
$\mathsf{u} \in \mathsf{abel}\ \&\ (\forall x \in \mathsf{abel})(x{+}\mathsf{u} = x)$ -- *designated unit element*
$(\forall x, y, z \in \mathsf{abel})\big((x{+}y){+}z = x{+}(y{+}z)\big)$ -- *associativity*
==>(Σ) -- *summation operation*
$\Sigma(\emptyset) = \mathsf{u}\ \&\ (\forall x \in \mathbb{N})(\forall y \in \mathsf{abel})\big(\Sigma(\{[x, y]\}) = y\big)$
$\mathsf{is\_map}(F)\ \&\ \mathsf{Finite}(F)\ \&\ \mathsf{range}(F) \subseteq \mathsf{abel}\ \&\ \mathsf{domain}(F) \subseteq \mathbb{N} \rightarrow$
$\Sigma(F) = \Sigma(F \cap G) + \Sigma(F \setminus G)$ -- *additivity*
END sigma_add.

We show below how to construct this **THEORY** from its assumptions, and how to generalize it into a **THEORY** gen_sigma_add in which the condition $\mathsf{domain}(F) \subseteq \mathbb{N}$ is dropped, allowing the condition $(\forall x \in \mathbb{N})(\forall y \in \mathsf{abel})\big(\Sigma(\{[x, y]\}) = y\big)$ to be simplified into $(\forall y \in \mathsf{abel})\big(\Sigma(\{[X, y]\}) = y\big)$. After this, we will sketch the proof of a basic property ('rearrangement of terms') of this generalized summation operation.

5.1 Existence of a finite summation operation

In order to tackle even the simple sigma_add, it is convenient to make use of recursions somewhat different (and actually simpler) than the fully general transfinite $\in$-recursion axiomatically available in our version of set theory. Specifically, we can write

$$\Sigma(F) := \textbf{if } F = \emptyset \textbf{ then } \mathsf{u} \textbf{ else } \mathsf{tail}(\textbf{arb } F) + \Sigma(F \textbf{ less arb } F) \textbf{ end if},$$

which is a sort of "tail recursion" based on set inclusion.

To see why such constructions are allowed we can use the fact that strict inclusion is a well-founded relation between finite sets, and in particular that it is well-founded over $\{\, f \subseteq \mathbb{N} \times \mathsf{abel} \mid \mathsf{Finite}(f) \,\}$: this makes the above form of recursive definition acceptable.

In preparing to feed this definition —or something closely equivalent to it— into our proof-verifier, we can conveniently make a *détour* through the following **THEORY** (note that in the following formulae $\mathsf{Ord}(X)$ designates the predicate 'X is an ordinal'—see end of Sec.3):

THEORY well_founded_set(s, Lt)
$(\forall t \subseteq \mathsf{s})\big(t \neq \emptyset \rightarrow (\exists m \in t)(\forall u \in t) \neg \mathsf{Lt}(u, m)\big)$
-- Lt *is thereby assumed to be irreflexive and well-founded on* s
==>(orden)
$(\forall x, y \in \mathsf{s})\Big(\big(\mathsf{Lt}(x, y) \rightarrow \neg\,\mathsf{Lt}(y, x)\big)\ \&\ \neg\,\mathsf{Lt}(x, x)\Big)$
$\mathsf{s} \subseteq \{\, \mathsf{orden}(y) : y \in X \,\} \leftrightarrow \mathsf{orden}(X) = \mathsf{s}$
$\mathsf{orden}(X) \neq \mathsf{s} \leftrightarrow \mathsf{orden}(X) \in \mathsf{s}$
$\mathsf{Ord}(U)\ \&\ \mathsf{Ord}(V)\ \&\ \mathsf{orden}(U) \neq \mathsf{s} \neq \mathsf{orden}(V) \rightarrow$
$\Big(\mathsf{Lt}\big(\mathsf{orden}(U), \mathsf{orden}(V)\big) \rightarrow U \in V\Big)$

$\{\, u \in \mathsf{s} \mid \mathsf{Lt}\big(u, \mathsf{orden}(V)\big)\,\} \subseteq \{\, \mathsf{orden}(x) \,:\, x \in V\,\}$
$\mathsf{Ord}(U)\ \&\ \mathsf{Ord}(V)\ \&\ \mathsf{orden}(U) \neq \mathsf{s} \neq \mathsf{orden}(V)\ \&\ U \neq V\ \rightarrow$
$\mathsf{orden}(U) \neq \mathsf{orden}(V)$
$\exists\, o\Big(\ \mathsf{Ord}(o)\ \&\ \mathsf{s} = \{\, \mathsf{orden}(x) \,:\, x \in o\,\}\ \&$
$\mathsf{1\_1\_map}\big(\,\{[x, \mathsf{orden}(x)] \,:\, x \in o\}\,\big)\ \Big)$

END well_founded_set.

Within this **THEORY** and in justification of it, orden can be defined in two steps:

$$\begin{aligned} \mathsf{Minrel}(T)\ &:=\ \textbf{if}\ \emptyset \neq T \subseteq \mathsf{s}\ \textbf{then arb}\ \{\, m \in T \mid (\forall x \in T)\neg\,\mathsf{Lt}(x, m)\,\} \\ &\qquad\ \textbf{else}\ \mathsf{s}\ \textbf{end if}\,, \\ \mathsf{orden}(X)\ &:=\ \mathsf{Minrel}\big(\,\mathsf{s} \setminus \{\, \mathsf{orden}(y) \,:\, y \in X\}\,\big)\,, \end{aligned}$$

after which the proof of the output theorems of the **THEORY** just described will take approximately one hundred lines.

Next we introduce a **THEORY** of recursion on well-founded sets. Even though the definition of Σ only requires much less, other kinds of recursive definition benefit if we provide a generous scheme like the following:

THEORY recursive_fcn(dom, Lt, a, b, P)
$(\forall t \subseteq \mathsf{dom})\big(\, t \neq \emptyset\ \rightarrow\ (\exists\, m \in t)(\forall u \in t)\neg\,\mathsf{Lt}(u, m)\,\big)$
-- Lt *is thereby assumed to be irreflexive and well-founded on* dom
==>(rec)
$(\forall\, v \in \mathsf{dom})\big(\,\mathsf{rec}(v) =$
$\mathsf{a}\big(\, v, \{\ \mathsf{b}\big(\, v, w, \mathsf{rec}(w)\,\big)\ :\ w \in \mathsf{dom} \mid \mathsf{Lt}(w, v)\ \&\ \mathsf{P}\big(\, v, w, \mathsf{rec}(w)\,\big)\ \}\,\big)\,\big)$
END recursive_fcn.

The output symbol rec of this **THEORY** is easily definable as follows:

$$\begin{aligned} \mathsf{G}(X)\ &:=\ \mathsf{a}\Big(\,\mathsf{orden}(X), \{\ \mathsf{b}\big(\,\mathsf{orden}(X), \mathsf{orden}(y), \mathsf{G}(y)\,\big)\ :\ y \in X \mid \\ &\qquad\qquad \mathsf{Lt}\big(\,\mathsf{orden}(y), \mathsf{orden}(X)\,\big)\ \&\ \mathsf{P}\big(\,\mathsf{orden}(X), \mathsf{orden}(y), \mathsf{G}(y)\,\big)\ \}\,\Big)\,, \\ \mathsf{rec}(V)\ &:=\ \mathsf{G}\big(\,\mathsf{index\_of}(V)\,\big)\,; \end{aligned}$$

here orden results from an invocation of our previous **THEORY** well_founded_set, namely

$$\textbf{APPLY}(\mathsf{orden})\ \mathsf{well\_founded\_set}\big(\,\mathsf{s} \mapsto \mathsf{dom},\ \mathsf{Lt}(X, Y) \mapsto \mathsf{Lt}(X, Y)\,\big) ==> \cdots;$$

also, the restriction of index_to to dom is assumed to be the local inverse of the function orden. Note that the recursive characterization of rec in the theorem of recursive_fcn is thus ultimately justified in terms of the very general form of $\in$-recursion built into our system, as appears from the definition of G.

Since we cannot take it for granted that we have an inverse of orden, a second auxiliary **THEORY**, invokable as

$$\textbf{APPLY}(\mathsf{index\_of})\ \mathsf{bijection}\big(\,\mathsf{f}(X) \mapsto \mathsf{orden}(X), \mathsf{d} \mapsto \mathsf{o1}, \mathsf{r} \mapsto \mathsf{dom}\,\big) ==> \cdots,$$

is useful. Here o1 results from Skolemization of the last theorem in well_founded_set. The new **THEORY** used here can be specified as follows:

THEORY bijection(f, d, r)
  1_1_map({[x, f(x)] : x ∈ d}) & r = { f(x) : x ∈ d }
  f(X) ∈ r → X ∈ d -- *convenient "guard"*
==>(finv)
  Y ∈ r → f(finv(Y)) = Y
  Y ∈ r → finv(Y) ∈ d
  X ∈ d ↔ f(X) ∈ r
  X ∈ d → finv(f(X)) = X
  (finv(Y) ∈ d & ∃ x(f(x) = Y)) ↔ Y ∈ r
  d = { finv(y) : y ∈ r } & 1_1_map({[y, finv(y)] : y ∈ r})
END bijection.

This little digression gives us one more opportunity to show the interplay between theories, because one way of defining finv inside bijection would be as follows:

APPLY(finv) fcn_from_pred(
  P(Y, X) ↦ f(X) = Y & d ≠ ∅ ,
  e(Y) ↦ **if** Y ∈ r **then** d **else arb** d **end if**)==> ··· ,

where fcn_from_pred is as shown in Sec.4.

We can now recast our first-attempt definition of Σ as

APPLY(Σ) recursive_fcn(
  dom ↦ { f ⊆ ℕ × abel | is_map(f) & Finite(f) } ,
  Lt(W, V) ↦ W ⊆ V & W ≠ V ,
  a(V, Z) ↦ **if** V = ∅ **then** u **else** tail(**arb** V) + **arb** Z **end if** ,
  b(V, W, Z) ↦ Z ,
  P(V, W, Z) ↦ W = V less **arb** V)==> ··· ,

whose slight intricacy is the price being paid to our earlier decision to keep the recursive definition scheme very general.

We skip the proofs that Σ(∅) = u and (∀ x ∈ ℕ)(∀ y ∈ abel)(Σ({[x, y]}) = y), which are straightforward. Concerning additivity, assume by absurd hypothesis that f is a finite map with domain(f) ⊆ ℕ and range(f) ⊆ abel such that Σ(f) ≠ Σ(f ∩ g) + Σ(f \ g) holds for some g, and then use the following tiny but extremely useful **THEORY** (of induction over the subsets of any finite set)

THEORY finite_induction(n, P)
  Finite(n) & P(n)
==>(m)
  m ⊆ n & P(m) & (∀ k ⊆ m)(k ≠ m → ¬ P(k))
END finite_induction,

to get an inclusion-minimal such map, f0, by performing an

$$\textbf{APPLY}(\mathsf{f0})\ \mathsf{finite\_induction}\big(\ \mathsf{n} \mapsto \mathsf{f},\ \mathsf{P}(F) \mapsto \exists\, g\big(\ \Sigma(F) \neq \Sigma(F \cap g) + \Sigma(F \setminus g)\ \big)\ \big) ==> \cdots\ .$$

Reaching a contradiction from this is very easy.

5.2 Generalized notion of finite summation

Our next goal is to generalize the finite summation operation $\Sigma(F)$ to any finite map F with $\mathsf{range}(F) \subseteq \mathsf{abel}$. To do this we can use a few basic theorems on ordinals, which can be summarized as follows. Define

$$\begin{aligned} \mathsf{min\_el}(T,S)\ &:=\ \textbf{if}\ S \subseteq T\ \textbf{then}\ S\ \textbf{else arb}\ (S \setminus T)\ \textbf{end if}\,, \\ \mathsf{enum}(X,S)\ &:=\ \mathsf{min\_el}\big(\ \{\,\mathsf{enum}(y)\ :\ y \in X\},S\ \big)\,, \end{aligned}$$

for all sets S, T (a use of $\in$-recursion quite similar to the construction used inside the **THEORY** well_founded_set![6]). Then the following *enumeration theorem* holds:

$$\begin{aligned} \exists\, o\,\big(\ \mathsf{Ord}(o)\ \&\ S = \big\{\,\mathsf{enum}(x,S)\ :\ x \in o\,\big\} \\ \&\ (\forall\, x,y \in o)\big(\ x \neq y\ \rightarrow\ \mathsf{enum}(x,S) \neq \mathsf{enum}(y,S)\ \big)\ \big)\,. \end{aligned}$$

From this one gets the function ordin by Skolemization.

Using the predicate Finite of Fig.1, and exploiting the infinite set **s_inf** axiomatically available in our version of set theory, we can give the following definition of natural numbers:

$$\mathbb{N}\ :=\ \textbf{arb}\ \big\{\,x \in \mathsf{next}\big(\,\mathsf{ordin}(\textbf{s\_inf})\,\big)\ |\ \neg\,\mathsf{Finite}(x)\,\big\}\,.$$

These characterizations of Finite and $\mathbb{N}$ yield

$$\begin{aligned} X \in \mathbb{N}\ &\leftrightarrow\ \mathsf{ordin}(X) = X\ \&\ \mathsf{Finite}(X)\,, \\ \mathsf{Finite}(X)\ &\leftrightarrow\ \mathsf{ordin}(X) \in \mathbb{N}\,, \\ \mathsf{Finite}(F)\ &\rightarrow\ \mathsf{Finite}\big(\,\mathsf{domain}(F)\,\big)\ \&\ \mathsf{Finite}\big(\,\mathsf{range}(F)\,\big)\,. \end{aligned}$$

Using these results and working inside the **THEORY** gen_sigma_add, we can obtain the generalized operation Σ by first invoking

$$\textbf{APPLY}(\sigma)\ \mathsf{sigma\_add}\big(\ \mathsf{abel} \mapsto \mathsf{abel}, \mathsf{+} \mapsto \mathsf{+}, \mathsf{u} \mapsto \mathsf{u}\ \big) ==> \cdots$$

and then defining:

$$\begin{aligned} \Sigma(F)\ :=\ \sigma\big(\ \big\{\,[x,y]\ :\ x \in \mathsf{ordin}\big(\,\mathsf{domain}(F)\,\big),\ y \in \mathsf{range}(F) \\ |\ \big[\,\mathsf{enum}\big(\,x,\mathsf{domain}(F)\,\big),y\,\big] \in F\,\big\}\ \big)\,. \end{aligned}$$

We omit the proofs that $\Sigma(\emptyset) = \mathsf{u}$, $(\forall\, y \in \mathsf{abel})\big(\,\Sigma(\{[X,y]\}) = y\,\big)$, and $\Sigma(F) = \Sigma(F \cap G) + \Sigma(F \setminus G)$, which are straightforward.

[6] This is more than just an analogy: we could exploit the well-foundedness of $\in$ to hide the details of the construction of enum into an invocation of the **THEORY** well_founded_set.

5.3 Rearrangement of terms in finite summations

To be most useful, the **THEORY** of Σ needs to encompass various strong statements of the additivity property. Writing

$$\begin{aligned} \Phi(F) &\equiv \mathsf{is\_map}(F) \ \&\ \mathsf{Finite}\big(\ \mathsf{domain}(F)\ \big) \ \&\ \mathsf{range}(F) \subseteq \mathsf{abel}\,, \\ \Psi(P,X) &\equiv\ X = \bigcup P \ \ \&\ \ (\forall\, b,v \in P)(b \neq v \ \ \rightarrow\ b \cap v = \emptyset) \end{aligned}$$

for brevity, much of what is wanted can be specified e.g. as follows:

THEORY $\mathsf{gen\_sigma\_add}(\mathsf{abel},\mathsf{+},\mathsf{u})$
$(\forall\, x,y \in\ \mathsf{abel})(x\mathsf{+}y \in \mathsf{abel}\ \&$ -- *closure w.r.t. ...*
$\qquad x\mathsf{+}y = y\mathsf{+}x)$ -- *... commutative operation*
$\mathsf{u} \in \mathsf{abel}\ \&\ (\forall\, x\ \in\ \mathsf{abel})(x\mathsf{+}\mathsf{u} = x)$ -- *designated unit element*
$(\forall\, x,y,z\ \in\ \mathsf{abel})\big(\ (x\mathsf{+}y)\mathsf{+}z = x\mathsf{+}(y\mathsf{+}z)\ \big)$ -- *associativity*
$\Longrightarrow(\Sigma)$ -- *summation operation*
$\Sigma(\emptyset) = \mathsf{u}\ \&\ (\forall\, y \in \mathsf{abel})\big(\ \Sigma(\{[X,y]\}) = y\ \big)$
$\Phi(F)\ \rightarrow\ \Sigma(F) \in \mathsf{abel}$
$\Phi(F) \rightarrow \Sigma(F) = \Sigma(F \cap G)\,\mathsf{+}\,\Sigma(F \setminus G)$ -- *additivity*
$\Phi(F)\ \&\ \Psi\big(\ P,F\ \big)\ \rightarrow\ \Sigma(F) = \Sigma\big(\ \big\{\ [g,\Sigma(g)]\ :\ g \in P\ \big\}\ \big)$
$\Phi(F)\ \&\ \Psi\big(\ P,\mathsf{domain}(F)\ \big)\ \rightarrow\ \Sigma(F) = \Sigma\Big(\ \big\{\ [b,\Sigma\big(\ F_{|b}\ \big)]\ :\ b \in P\ \big\}\ \Big)$
$\Phi(F)\ \&\ \mathsf{Svm}(G)\ \&\ \mathsf{domain}(F) = \mathsf{domain}(G)\ \rightarrow$
$\qquad \Sigma(F) = \Sigma\Big(\ \big\{\ [x,\Sigma\big(\ F_{|G^{-1}\{x\}}\ \big)]\ :\ x \in \mathsf{range}(G)\ \big\}\ \Big)$
END $\mathsf{gen\_sigma\_add}$.

A proof of the last of these theorems, which states that Σ is insensitive to operand rearrangement and grouping, is sketched below.

Generalized additivity is proved first: starting with the absurd hypothesis that specific f, p exist for which

$$\Phi(\mathsf{f})\ \&\ \Psi\big(\ \mathsf{p},\mathsf{f}\ \big)\ \&\ \Sigma(\mathsf{f}) \neq \Sigma\big(\ \big\{\ [g,\Sigma(g)]\ :\ g \in \mathsf{p}\ \big\}\ \big)$$

holds, one can choose an inclusion-minimal such p referring to the same f and included in the p chosen at first, by an invocation

$$\mathbf{APPLY}(\mathsf{p0})\ \mathsf{finite\_induction}\big(\ \mathsf{n} \mapsto \mathsf{p},\ \mathsf{P}(Q) \mapsto \Psi\big(\ Q,\mathsf{f}\ \big)\ \&\ \Sigma(\mathsf{f}) \neq \Sigma\big(\ \big\{\ [g,\Sigma(g)]\ :\ g \in Q\ \big\}\ \big)\ \big) \Longrightarrow \cdots\ .$$

From this, a contradiction is easily reached.

The next theorem, namely

$$\Phi(F)\ \&\ \Psi\big(\ P,\mathsf{domain}(F)\ \big)\ \rightarrow\ \Sigma(F) = \Sigma\Big(\ \big\{\ [b,\Sigma\big(\ F_{|b}\ \big)]\ :\ b \in P\ \big\}\ \Big)$$

follows since $\Psi\big(\ P,\mathsf{domain}(F)\ \big)$ implies $\Psi\big(\ \{F_{|b}\ :\ b \in P\},F\ \big)$.

Proof of the summand rearrangement theorem seen above is now easy, because

$$\mathsf{Svm}(G)\ \&\ D = \mathsf{domain}(G)\ \rightarrow\ \Psi\big(\ \big\{\ G^{-1}\{x\}\ :\ x \in \mathsf{range}(G)\ \big\},\ D\ \big)$$

holds for any D and hence in particular for $D = \mathsf{domain}(F)$.

The above line of proof suggests a useful preamble is to construct the following theory of Ψ:

THEORY is_partition(p, s)
==>(flag) -- *this indicates whether or not* s *is partitioned by* p
flag $\leftrightarrow$ s $= \bigcup$ p & $(\forall\, b, v)(b \neq v \;\rightarrow\; b \cap v = \emptyset)$
flag & Finite(s) $\rightarrow$ Finite(p)
flag & s = domain(F) & $Q = \{\, F_{|b} \,:\, b \in \mathsf{p} \,\} \;\rightarrow\; F = \bigcup Q$ &
$(\forall\, f, g \in Q)(f \neq g \;\rightarrow\; f \cap g = \emptyset)$
Svm(G) & s = domain(G) & $\mathsf{p} = \left\{\, G^{-1}\{y\} \,:\, y \in \mathsf{range}(G) \,\right\} \;\rightarrow$ flag
END is_partition.

6 Related work

To support software design and specification, rapid prototyping, theorem proving, user interface design, and hardware verification, various authors have proposed systems embodying constructs for modularization which are, under one respect or another, akin to our **THEORY** construct. Among such proposals lies the OBJ family of languages [15], which integrates specification, prototyping, and verification into a system with a single underlying equational logic.

In the implementation OBJ3 of OBJ, a module can either be an *object* or a *theory*: in either case it will have a set of equations as its body, but an object is executable and has a fixed standard model whereas a theory describes non-executable properties and has loose semantics, namely a variety of admissible models. As early as in 1985, OBJ2 [13] was endowed with a generic module mechanism inspired by the mechanism for parameterized specifications of the Clear specification language [3]; the interface declarations of OBJ2 generics were not purely syntactic but contained semantic requirements that actual modules had to satisfy before they could be meaningfully substituted.

The use of OBJ for theorem-proving is aimed at providing mechanical assistance for proofs that are needed in the development of software and hardware, more than at mechanizing mathematical proofs in the broad sense. This partly explains the big emphasis which the design of OBJ places on equational reasoning and the privileged role assigned to universal algebra: equational logic is in fact sufficiently powerful to describe any standard model within which one may want to carry out computations.

We observe that an equational formulation of set theory can be designed [11], and may even offer advantages w.r.t. a more traditional formulation of Zermelo-Fraenkel in limited applications where it is reasonable to expect that proofs can be found in fully automatic mode; nevertheless, overly insisting on equational reasoning in the realm of set theory would be preposterous in light of the highly interactive proof-verification environment which we envision.

We like to mention another ambitious project, closer in spirit to this paper although based on a sophisticated variant of Church's typed lambda-calculus

[6]: the Interactive Mathematical Proof System (IMPS) described in [10]. This system manages a database of mathematics, represented as a collection of interconnected axiomatic "*little theories*" which span graduate-level parts of analysis (about 25 theories: real numbers, partial orders, metric spaces, normed spaces, etc.), some algebra (monoids, groups, and fields), and also some theories more directly relevant to computer science (concerning state machines, domains for denotational semantics, and free recursive datatypes). The initial library caters for some fragments of set theory too: in particular, it contains theorems about cardinalities. Mathematical analysis is regarded as a significant arena for testing the adequacy of formalizations of mathematics, because analysis requires great expressive power for constructing proofs.

The authors of [10] claim that IMPS supports a view of the axiomatic method based on "little theories" tailored to the diverse fields of mathematics as well as the "big theory" view in which all reasoning is performed within a single powerful and highly expressive set theory. Greater emphasis is placed on the former approach, anyhow: with this approach, links —"conduits", so to speak, to pass results from one theory to another— play a crucial role. To realize such links, a syntactic device named "theory interpretation" is used in a variety of ways to translate the language of a source theory to the language of a target theory so that the image of a theorem is always a theorem: this method enables reuse of mathematical results "transported" from relatively abstract theories to more specialized ones.

One main difference of our approach w.r.t. that of IMPS is that we are willing to invest more on the "big theory" approach and, accordingly, do not feel urged to rely on a higher-order logic where functions are organized according to a type hierarchy. It may be contended that the typing discipline complies with everyday mathematical practice, and perhaps gives helpful clues to the automated reasoning mechanisms so as to ensure better performance; nevertheless, a well-thought type-free environment can be conceptually simpler.

Both OBJ and IMPS attach great importance to interconnections across theories, inheritance to mention a most basic one, and "theory ensembles" to mention a nice feature of IMPS which enables one to move, e.g., from the formal theory of a metric space to a family of interrelated replicas of it, which also caters for continuous mappings between metric spaces. As regards theory interconnections, the proposal we have made in this paper still awaits being enriched.

The literature on the OBJ family and on the IMPS system also stresses the kinship between the activity of proving theorems and computing in general; even more so does the literature on systems, such as Nuprl [8] or the Calculus of Constructions [9], which rely on a constructive foundation, more or less close to Martin-Löf's intuitionistic type theory [19]. Important achievements, and in particular the conception of declarative programming languages such as Prolog, stem in fact from the view that proof-search can be taken as a general paradigm of computation. On the other hand, we feel that too little has been done, to date, in order to exploit a "proof-by-computation" paradigm aimed at enhanc-

ing theorem-proving by means of the ability to perform symbolic computations efficiently in specialized contexts of algebra and analysis (a step in this direction was moved with [7]). Here is an issue that we intend to deepen in a forthcoming paper.

7 Conclusions

We view the activity of setting up detailed formalized proofs of important theorems in analysis and number theory as an essential part of the feasibility study that must precede the development of any ambitious proof-checker. In mathematics, set theory has emerged as the standard framework for such an enterprise, and full computer-assisted certification of a modernized version of *Principia Mathematica* should now be possible. To convince ourselves of a verifier system's ability to handle large-scale mathematical proofs —and such proofs cannot always be avoided in program-correctness verification—, it is best to follow the royal road paved by the work of Cauchy, Dedekind, Frege, Cantor, Peano, Whitehead–Russell, Zermelo–Fraenkel–von Neumann, and many others.

Only one facet of our work on large-scale proof scenarios is presented in this paper. Discussion on the nature of the basic inference steps a proof-verifier should (and reasonably can) handle has been omitted to focus our discussion on the issue of proof modularization. The obvious goal of modularization is to avoid repeating similar steps when the proofs of two theorems are closely analogous. Modularization must also conceal the details of a proof once they have been fed into the system and successfully certified.

When coupled to a powerful underlying set theory, indefinitely expansible with new function symbols generated by Skolemization, the technical notion of "theory" proposed in this paper appears to meet such proof-modularization requirements. The examples provided, showing how often the **THEORY** construct can be exploited in proof scenarios, may convince the reader of the utility of this construct.

Acknowledgements

We thank Ernst-Erich Doberkat (Universität Dortmund, D), who brought to our attention the text by Frege cited in the epigraph of this paper. We are indebted to Patrick Cegielski (Université Paris XII, F) for helpful comments.

References

1. A. Blass and Y. Gurevich. The logic of choice. *J. of Symbolic Logic*, 65(3):1264–1310, 2000.
2. D. S. Bridges. *Foundations of real and abstract analysis*. Springer-Verlag, Graduate Texts in Mathematics vol.174, 1997.

3. R. Burstall and J. Goguen. Putting theories together to make specifications. In R. Reddy, ed, Proc. 5^{th} *International Joint Conference on Artificial Intelligence*. Cambridge, MA, pp. 1045–1058, 1977.
4. R. Caferra and G. Salzer, editors. *Automated Deduction in Classical and Non-Classical Logics*. LNCS 1761 (LNAI). Springer-Verlag, 2000.
5. P. Cegielski. Un fondement des mathématiques. In M. Barbut et al., eds, *La recherche de la vérité*. ACL – Les éditions du Kangourou, 1999.
6. A. Church. A formulation of the simple theory of types. *J. of Symbolic Logic*, 5:56–68, 1940.
7. E. Clarke and X. Zhao. Analytica—A theorem prover in Mathematica. In D. Kapur, ed, *Automated Deduction—CADE-11*. Springer-Verlag, LNCS vol. 607, pp. 761–765, 1992.
8. R. L. Constable, S. F. Allen, H. M. Bromley, W. R. Cleaveland, J. F. Cremer, R. W. Harper, D. J. Howe, T. B. Knoblock, N. P. Mendler, P. Panangaden, J. T. Sasaki, and S. F. Smith. *Implementing mathematics with the Nuprl development system*. Prentice-Hall, Englewood Cliffs, NJ, 1986.
9. Th. Coquand and G. Huet. The calculus of constructions. *Information and Computation*, 76(2/3):95–120, 1988.
10. W. M. Farmer, J. D. Guttman, F. J. Thayer. IMPS: An interactive mathematical proof system. *J. of Automated Reasoning*, 11:213–248, 1993.
11. A. Formisano and E. Omodeo. An equational re-engineering of set theories. In Caferra and Salzer [4, pp. 175–190].
12. G. Frege. Logik in der Mathematik. In *G. Frege, Schriften zur Logik und Sprachphilosophie*. Aus dem Nachlaß herausgegeben von G. Gabriel. Felix Meiner Verlag, Philosophische Bibliothek, Band 277, Hamburg, pp. 92–165, 1971.
13. K. Futatsugi, J. A. Goguen, J.-P. Jouannaud, J. Meseguer. Principles of OBJ2. Proc. 12^{th} *annual ACM Symp. on Principles of Programming Languages* (POPL'85), pp. 55-66, 1985.
14. R. Godement. *Cours d'algèbre*. Hermann, Paris, Collection Enseignement des Sciences, 3^{rd} edition, 1966.
15. J. A. Goguen and G. Malcolm. *Algebraic semantics of imperative programs*. MIT, 1996.
16. T. J. Jech. *Set theory*. Springer-Verlag, Perspectives in Mathematical Logic, 2^{nd} edition, 1997.
17. E. Landau. *Foundation of analysis. The arithmetic of whole, rational, irrational and complex numbers*. Chelsea Publishing Co., New York, 2^{nd} edition, 1960.
18. A. Levy. *Basic set theory*. Springer-Verlag, Perspectives in Mathematical Logic, 1979.
19. P. Martin-Löf. *Intuitionistic type theory*. Bibliopolis, Napoli, Studies in Proof Theory Series, 1984.