Little Theories*

William M. Farmer, Joshua D. Guttman, F. Javier Thayer

The MITRE Corporation

15 October 1992

Abstract

In the "little theories" version of the axiomatic method, different portions of mathematics are developed in various different formal axiomatic theories. Axiomatic theories may be related by inclusion or by theory interpretation. We argue that the little theories approach is a desirable way to formalize mathematics, and we describe how IMPS, an Interactive Mathematical Proof System, supports it.

1 Introduction

In this paper, we will argue in favor of implementing a particular version of the axiomatic method in mechanical theorem provers. By the axiomatic method we mean the practice of reasoning logically from a set of sentences in a formal language. Such a set of sentences is called an axiomatic theory.[1] For instance, Peano arithmetic and the theory of an ordered field are familiar axiomatic theories.

*Supported by the MITRE-Sponsored Research Program. Presented at the 11th International Conference on Automated Deduction, Saratoga Springs, NY, USA, June 1992. Appears in D. Kapur, ed., *Automated Deduction—CADE-11*, Lecture Notes in Computer Science, vol. 607, Springer-Verlag, 1992, pp. 567–581. Authors' address: The MITRE Corporation, 202 Burlington Rd, Bedford MA 01730-1420 USA; Telephone: 617-271-2749; Email: `farmer, guttman, jt@mitre.org`.

[1]Thus, in our usage, a theory amounts to a set of axioms in a specified formal language. The word "theory" is also frequently used to mean a set of sentences closed under logical consequence. The latter usage allows one to speak, for instance, of different axiomatizations of the same theory. However, for our purposes it is convenient to focus on the axioms, and to express the idea of alternative axiomatizations by saying that two theories have the same consequences.

There are two contrasting ways of using the axiomatic method, both well established in modern mathematical practice. We will refer to them as the "big theory" version and the "little theories" version. In the big theory version, we select a powerful and highly expressive set of axioms, with the property that any model of these axioms will contain all the objects that will be of interest to us. Logical derivation from these powerful axioms will allow us to prove our theorems about these objects, so all of the reasoning can be carried out within this single theory. Zermelo-Fraenkel set theory is frequently used by mathematicians for this purpose.

In the little theories version of the axiomatic method, a number of theories will be used in the course of developing a portion of mathematics. Different theorems will be proved in different theories, depending on the amount of structure required. For instance, one theorem may be true in any arbitrary topological space, while another may hold only in a metric space. Theorems are proved in a particular theory by logical derivation from the axioms available in that theory.

The goal of this paper is to demonstrate the usefulness of the little theories approach in mechanized reasoning. In addition, we will indicate how IMPS, an Interactive Mathematical Proof System [10], supports the approach.

We will not focus on logical issues related to the little theories approach, including its use of theory interpretations. On the contrary, the logic of theory interpretations is well understood, and a version for the particular logic we use is available in [8]. Interpretations have been effectively used in the logical literature since at least the 1950's [27], and in mathematics for much longer. Indeed, this makes interpretations especially attractive to us. Our overall goal in IMPS is to mechanize traditional tools of classical mathematical reasoning, partly because they are understood by a wide range of potential users, and partly because their intellectual power has been demonstrated over a long period of time.

1.1 Little Theories in Mathematics: An Example

The utility of the little theories approach in mathematics is largely due to the power that theory interpretations provide. A theory interpretation is a syntactic translation between the languages of two theories which preserves theorems. That is, if a formula is a theorem of the source theory, then its image is a theorem of the target theory. To establish that a translation is a theory interpretation, one must show that the source theory *axioms* are translated to target theory theorems, together with some additional obligations that depend on the details of the logical context. Theory interpretations are used in two crucial ways in the little theories approach:

- To reuse theorems of the source theory in the target theory;
- To establish the consistency of the source theory relative to the target theory. As a special case, to establish that a formula φ is independent of a theory $\mathcal{T}$, one may show that both $\mathcal{T} \cup \{\varphi\}$ and $\mathcal{T} \cup \{\neg\varphi\}$ are consistent.

A theory interpretation from a source theory $\mathcal{T}$ to a target theory $\mathcal{T}'$ also allows us to infer a relation between models of the theories. Namely, given an arbitrary model of $\mathcal{T}'$, the interpretation tells us how to select a subdomain and distinguished values that furnish a model of $\mathcal{T}$.

Partly because of its usefulness for establishing consistency and independence, the little theories approach has become a deeply entrenched way of organizing mathematical knowledge. For instance, in the introduction to *The Foundations of Geometry* [19], Hilbert wrote that one of his aims was:

> to bring out as clearly as possible the significance of the different groups of axioms and the scope of the conclusions to be derived from the individual axioms.

This expresses one of the major themes of modern mathematics, which aims to determine not just which mathematical statements are true, but also which assumptions are needed to deduce them.

The Foundations of Geometry illustrates how the axiomatic method can serve as an organizing principle in examining an axiomatization for Euclidean geometry. The goal of the book is to study relations of independence among subsets of the axioms, in combination with two crucial theorems, namely Pascal's theorem and Desargues's theorem. As a tool, Hilbert also considers a number of algebraic theories, primarily the theory of an ordered field.

A simple example of a theory interpretation is given in §§15, 17, where certain line segments are shown to form an ordered field with appropriately chosen operations. In this particular case, we are not concerned with an independence proof; instead, Hilbert introduces the theory interpretation so that he can use the familiar algebraic theorems to carry out computations in the course of giving geometrical proofs. The interpretation guarantees that the results of the algebraic computation will be sound in the geometrical context. This illustrates theorem reuse, in the helpful case of equations and conditional equations.[2]

In these cases, Hilbert makes no reference to any particular model of either theory (in the usual semantic sense). In some other cases, his language is more ambiguous, and he appears to be working within a specific model (for instance, the reals). However, he is explicit about the algebraic axioms his construction actually relies on, and emphasizes that the theory has denumerable models (see, for instance, §9). Thus, these other cases could be easily recast into a terminology in which we are unambiguously concerned with theory interpretations.

1.2 Little Theories in Mechanized Reasoning

Quite apart from the intellectual appeal of fine-grained knowledge about the logical power of particular axioms, and the relations among them, the little theories approach has two important practical advantages for mechanized reasoning.

- It allows the use of minimal axiomatizations for specific groups of theorems. This ensures maximal generality of reuse for the results, through interpretation into other theories, or through direct inclusion in larger theories.

  When equipped with procedures that can construct the great majority of the interpretations needed, an interactive theorem prover can offer the user great power in applying previously proved theorems across a wide range of new theories.

- The hand-crafted framework of a little theory allows theorems to be written in simpler forms. Cues suggesting the applicability of particular lemmas or other techniques are easier to identify.

  Moreover, the work of establishing the applicability of a whole group of theorems can be carried out once and for all, and then encoded in a theory interpretation. The theory interpretation then serves as a "license" authorizing us to use them repeatedly in the future.

The first of these advantages will be discussed primarily in Section 2. The second will be the focus of Section 3. Section 4 discusses the big theory

[2]See Section 2.3 for a mechanism in IMPS implementing a similar kind of reuse. For a more elaborate use of theory interpretations in Hilbert, see for instance [19, §§24–26, 28–29], where he exploits a whole sequence of theory interpretations.

approach. We will argue that the little theory approach does not create a disadvantage when we may want to use a powerful theory like **ZF**. Moreover, we will point out some cases in which this is desirable. In Section 5, we will present a substantial piece of mathematics as it appears in the little theory approach; this example has been developed with the interactive theorem prover IMPS.

1.3 Previous Work; IMPS

In spite of its utility, the little theories version of the axiomatic method lies off the main path of previous work in mechanized theorem proving.

The little theories idea is, however, a familiar ingredient in work on specification languages, probably first introduced by Burstall and Goguen [2]. It was a central tenet of work on Clear [3, 4], and it was also a motivating idea in Larch [16]. The idea is also an ingredient in more recent work on logical frameworks [18]. In the logical frameworks context, however, it appears in an unusual guise in which not just *theories* but also *logics* may be combined. It is not clear whether this additional generality will prove to be a benefit in practical use, as it may impede the process by which users develop strong and reliable semantic intuitions. Moreover, a single familiar logic, such as simple type theory, suffices to formalize conveniently a wide range of problems.

There has also been some work on supporting little theories in mechanized theorem proving. Sannella and Burstall undertook to implement some of the ideas of Clear in an extended version of LCF [24]. However, according to the the published description, theory interpretation and parameterized theories had not yet been implemented [24, pp. 384, 389]. Although OBJ [14] incorporates a translation ("view") mechanism, the user is responsible for deciding whether the translation is in fact a theory interpretation: OBJ itself makes no attempt to prove the images of the source theory axioms. Moreover, because of OBJ's equational logic, its usefulness as a theorem prover is, in our opinion, highly restricted. Curiously, the Larch Prover LP [12] does not give strong support for the little theories approach: there is only one theory available in the prover at a time, and thus theory interpretations cannot be used in proofs. E. Gunter [15] has made a start on implementing little theories within HOL.

So far as we know, IMPS, an Interactive Mathematical Proof System, is the first interactive theorem prover to have been designed from the start to support little theories. Moreover, IMPS implements a strong logic—rather than the weak systems generally used in algebraic specification languages—that is suited to expressing and proving sophisticated mathematical statements. One example, a simple inverse function theorem for Banach spaces, is presented in Section 5.

For a general description of IMPS, see [9, 10]. Examples of IMPS proofs are found in [9, 11]. All of the examples given below (except where explicitly noted) represent material we have developed using IMPS.

All concept formulation, calculation, and inference in IMPS is performed with respect to a formal logic that is a version of simple type theory. The logic, called LUTINS[3], provides strong support for specifying and reasoning about partial (and total) functions, and is equipped with a system of types and subtypes. Types and subtypes are collectively called *sorts*.

The treatment of partial functions in LUTINS is studied in [7], while the treatment of sorts and interpretations is the subject of [8]. For a detailed presentation of the syntax and semantics of LUTINS, see [17].

A language is built in IMPS from a signature—a list of sort and constant declarations. A theory consists of a language $\mathcal{L}$ plus a set of axioms (i.e., sentences of $\mathcal{L}$). Theories are the basic units in IMPS for specifying mathematical objects and concepts and for organizing automated deduction.

2 Theory Interpretations and Theorem Reuse

The notion of an interpretation of one axiomatic theory in another is a fundamental concept of mathematics and logic. As we mentioned in the introduction, this notion is formalized using certain syntactic translations that are known in logic as "theory interpretations" [6, 25]. Intuitively, a theory interpretation from $\mathcal{T}$ to $\mathcal{T}'$ specifies one of the (possibly many) ways of embedding $\mathcal{T}$ in $\mathcal{T}'$, while preserving theorems.

Logicians have used theory interpretations to prove metamathematical properties about theories, particularly consistency, decidability, and undecidability. The classic work of Tarski, Mostowski, and Robinson [27], for example, illustrates how theories can be proved undecidable by means of theory interpretation. References on theory interpretations include [23, 26, 28, 29].

[3] Pronounced as the word in French.

2.1 Theory Interpretations in IMPS

The notion of a theory interpretation in LUTINS is similar to the standard notion in first-order logic (see [6, 25]). However, since LUTINS admits partial functions and subtypes, the notion in LUTINS is necessarily more complicated. A precise definition of a LUTINS theory interpretation is given in [8].

In brief, a LUTINS *translation* from a theory $\mathcal{T}$ to a theory $\mathcal{T}'$ is specified by a pair (μ, ν) of functions—where μ maps the sorts of $\mathcal{T}$ to sorts, sets, or unary predicates of $\mathcal{T}'$ and ν maps the constants of $\mathcal{T}$ to expressions of $\mathcal{T}'$—which satisfy certain syntactic conditions. The translation is a homomorphism I from the expressions of the source theory to the expressions of the target theory, i.e., $I(c(e_1, \ldots, e_n)) = c(I(e_1), \ldots, I(e_n))$, where $c(e_1, \ldots, e_n)$ is a compound expression composed of a logical constant c and n subexpressions $e_1, \ldots, e_n$.[4] Every translation I from $\mathcal{T}$ to $\mathcal{T}'$ determines a set of formulas in the target theory called *obligations*, which includes $I(\theta)$ for each axiom θ of $\mathcal{T}$. By the Interpretation Theorem for LUTINS [8], if each obligation of I is a theorem of $\mathcal{T}'$, then I translates each theorem of $\mathcal{T}$ to a theorem of $\mathcal{T}'$, i.e., I is a theory interpretation.

2.2 Theorem Reuse in Mathematics

Mathematicians commonly use a kind of informal theory interpretation for reusing theorems. A result about an abstract theory, such as a group, will be applied to a more concrete theory such as a field, by (for instance) observing that multiplication over the nonzero elements has the structure of a group.

In the the little theories version of the axiomatic method, mathematical reasoning is distributed over a network of theories linked to one another via theory interpretations. These theory interpretations provide the means to "transport" a theorem from the theory it was proved in to any number of other theories. For instance, the theorem that a sequence of points converges to at most one limit can be proved once in a theory of an abstract metric space and then applied to a sequence of reals, to a sequence of points in a normed space, and so on.

Similarly, a theory of an abstract monoid (that is, an associative operator $\oplus$ with an identity e), taken together with the integers as an additional type, allows one to define an iterated monoid summation operator $\sum$ such that

[4] When μ maps sorts to sets or unary predicates, expressions beginning with variable binders such as $\forall$ or λ must be "relativized." For example, if I maps a sort α to a unary predicate φ on a sort β, then $I(\forall x{:}\alpha . \psi) = \forall x{:}\beta . \varphi(x) \supset I(\psi)$.

$\sum_{k=i}^{j} f = e$ for $j < i$, and $\sum_{k=i}^{j} f = (\sum_{k=i}^{j-1} f) \oplus f(j)$ otherwise. Facts derived in this theory can then be applied to a large class of iterated operators, including the normal numerical sum and product operators. Many interesting facts about algorithms can be developed in this general framework [1].

Another very useful technique is to interpret a theory into itself or one of its subtheories. For example, many similar theorems about algebraic groups are just different instantiations of a particular abstract theorem about group actions [20, pp. 71–79]. The abstract theorem φ can be proved in a theory $\mathcal{A}$ of an abstract group action, which includes a theory $\mathcal{B}$ of an abstract group as a subtheory, and then the various instantiations can be obtained by transporting φ from $\mathcal{A}$ to $\mathcal{B}$ via appropriate interpretations.

Sometimes, this second technique provides proofs by symmetry or duality. For instance, the duality of points and lines in projective geometry can be formalized by observing that the translation that interchanges them is an interpretation. Having done so, we can prove a theorem in one form and then immediately infer that its dual is also a theorem.[5]

2.3 Theorem Reuse in IMPS

Although theory interpretations can be used in IMPS in several different ways, theorem reuse is certainly the most important application. An IMPS user can build translations, verify that they are theory interpretations, and transport theorems with them as desired. The development of a portion of mathematics in IMPS will usually involve several theories and a great many theory interpretations, most of which are quite simple. However, building theory interpretations can be distracting or burdensome, especially in the midst of a proof. If the user were responsible for explicitly creating all these theory interpretations, it would be very difficult to concentrate on the major task at hand.

Consequently, the great majority of the theory interpretations needed by the IMPS user are built by software without user assistance. In addition, when hand-crafted interpretations have been added by the user, the system can retrieve the right one in most situations.

There are about a half dozen mechanisms in IMPS that automatically find, extend, or create theory interpretations. Of these the most useful is a kind of polymorphic matching called *translation matching*, which allows for inter-theory matching of expressions. An expression e in a theory $\mathcal{T}'$

[5] This example has not been done in IMPS, but several other examples of proof by symmetry have been formalized in IMPS using interpretations.

translation matches a pattern expression p in a theory $\mathcal{T}$ if there is a theory interpretation I from $\mathcal{T}$ to $\mathcal{T}'$ and a substitution σ such that $\sigma(I(p)) = e$. The translation matching algorithm works in IMPS as follows. First, the variables and constants in p are matched with subexpressions of e yielding a "sort association list" and a "constant association list." If the entries of these lists are compatible with each other (e.g., no constant is associated with two distinct expressions), IMPS will try to find a theory interpretation defined by (μ, ν) such that the sort and constant association lists specify subfunctions of μ and ν, respectively. If that fails, IMPS will try to build a theory interpretation directly from the two association lists. If a theory interpretation I is obtained and e matches $I(p)$ in the ordinary sense, then σ is just the substitution which matches the expressions.

Translation matching is employed in several kinds of machinery in IMPS for theorem reuse. For example, when theorems about generic objects like sets and sequences are of appropriate form to be used as rewrite rules, IMPS uses translation matching to automatically create the theory interpretations and substitutions needed to apply them whenever relevant in the course of simplification.

A collection of theorems can be applied as conditional rewrite rules in an organized way in IMPS using extremely simple procedures called *macetes*[6]. The exact behavior of an *elementary macete* (built from a particular theorem) depends on the syntactic form of the theorem. In the central case of a theorem of the form $\forall \vec{x} \,.\, \varphi_1 \wedge \ldots \wedge \varphi_n \supset s = t$, the elementary macete replaces matches to s by the corresponding matches to t, if the instances of the φ_i can be recognized to be true. Compound macetes may be constructed from elementary and other kinds of atomic macetes using a small number of very simple combining forms. For instance, the `repeat` form takes a list of macetes and applies them repeatedly until they are no longer applicable. A small number of special atomic macetes allow the user to intersperse beta-reduction, definition expansion, and calls to the simplifier [9].

A *transportable macete*, like an elementary macete, is another kind of atomic macete that is built from a theorem. They are used, when working in a theory $\mathcal{T}$, to apply theorems that reside outside of $\mathcal{T}$. The mechanism is similar to that for elementary macetes except that translation matching is used instead of ordinary matching.

[6] *Macete*, in Portuguese, means a chisel, or in informal usage, a clever trick.

3 Controlling Theorem Provers

The axiomatic theory is a natural unit to use in organizing and guiding the behavior of a theorem prover. In this section we will discuss several ways that information is gathered around an axiomatic theory in order to control the behavior of IMPS.

3.1 Expressing Theorems in Simple Form

There is frequently an advantage to expressing facts in a simple form, as they can then be used by a theorem prover in some special way in the course of computing with expressions. For instance, equations and biconditionals can be used as (unconditional) rewrite rules in the course of simplification. Within the logical framework we use, there are two ways that the little theories approach aids in recasting facts in the simplest syntactic form.

First, a theorem that in pure logic would take the form $\bigwedge_{i<n} \varphi_i \supset \psi$ may be written in the form ψ in any axiomatic theory whose axioms include (or entail) the formulas φ_i. Naturally, in order to *apply* the theorem in a new theory under some translation (possibly the identity translation), one must check that the translations of the assumptions φ_i are in fact satisfied. This requirement amounts to proving that the translation is an interpretation.

Although the work of discharging these obligations must still be done, there is often an advantage to this approach. For, the interpretation itself can be treated in the theorem proving program as a data object, so that once it has been certified, all the theorems of the source theory may be made available. Similarly, the interpretation as a data object makes it easy to avoid proving the same conditions repeatedly when a theorem (or group of related theorems) are to be applied many times.

Second, the LUTINS logic supports subtypes called sorts. For instance, in our theory of arithmetic, the natural numbers and integers are subtypes of the reals. In an axiomatic theory that involves a subtype σ of some type τ, $\forall x : \sigma \,.\, \psi(x)$ can replace $\forall y : \tau \,.\, \varphi(y) \supset \psi(y)$, where φ is the unary predicate corresponding to membership in the subtype σ. As before, this is no magic way to eliminate work: given a term t, we must ensure that t is defined with a value in the sort σ. Although this is semantically similar to discharging the assumption $\varphi(t)$, a theorem prover may be equipped with algorithms to resolve questions of this form effectively.

IMPS gains considerably from a range of algorithms for these "sort definedness" assertions. They use information extracted from theorems, such as that σ is closed under particular functions. They frequently reduce the assertion that a complicated term t is defined with a value in σ to a few assertions about small subterms. If these in turn are not discharged by the simplifier, they may be presented to the user for proof.

3.2 Simplification and Decision Procedures

Many theorem provers use efficient algorithms, exploiting facts about particular operators, either to simplify expressions or to decide formulas in some syntactic class. These hand-coded procedures, which we will call *processors*, may be far more efficient than applying basic inferences to derive the same conclusion.

However, the same algorithm is often sound for a range of theories, so long as they satisfy a number of presuppositions. Suppose a number of formal symbols, such as $+$ and $*$, or $\oplus$ and $\otimes$, represent operations the algorithm might be applied to. The processor can then generate a number of concrete formulas representing presuppositions for the manipulations that it will perform. For instance, these presuppositions might assert that the operators are associative and satisfy a distributive property. The simplifier for that theory can soundly call the processor if these presuppositions are theorems.

Moreover, a processor may be installed with subsets of the possible operations, so that a processor for simplifying polynomials in a field can be installed in a commutative ring (with unit) simply by not supplying a division operator. Alternatively, some properties of an operator may be optional: a processor designed for a commutative ring may be applied to a non-commutative ring simply by stipulating that the multiplication may not commute. The code of the processor must of course be written so that these optional choices make sense.

IMPS allows a user to tailor such a processor for his theory with no need to write code. For instance, if a theory of bitstrings is needed, it is trivial to instantiate an existing algebraic processor to provide efficient simplification for expressions involving the bitstring operations. The theory is the natural unit for this sort of theorem proving aid.

4 Relation to the Big Theory Approach

It might be thought that there is also a disadvantage to the little theories approach. Namely, in some situations we may want to use a "big theory" like **ZF** as a context to reason about structures satisfying different properties. Naturally, set theories can also be treated within IMPS. But, if we do so, can we make use of theorems that we have proved using the little axiomatic theories that characterize these structures?

In fact, we do not sacrifice the opportunity to use a "big theory" in connection with results proved using the little theories approach. For instance, suppose that we have a collection of theorems we have proved in a theory $\mathcal{M}$ of monoids. $\mathcal{M}$ asserts of the constants e and $\oplus$ that e is an identity for $\oplus$, and that $\oplus$ is associative. Within the chosen big theory $\mathcal{B}$, define a ternary relation *monoid* between a set, an object, and a binary function by the condition:

$$\begin{array}{ll} \multicolumn{2}{l}{\mathit{monoid}(s,x,f) =_{df}} \\ & x \in s \\ \wedge & \forall z \,.\, z \in s \supset f(z,x) = f(x,z) = z \\ \wedge & \forall z_0, z_1, z_2 \,.\; z_0, z_1, z_2 \in s \supset f(z_0, f(z_1,z_2)) = f(f(z_0,z_1),z_2) \end{array}$$

Suppose we can establish a formula of the form $\mathit{monoid}(s_0, x_0, f_0)$, possibly relative to some assumptions $\{\varphi_1, \ldots, \varphi_n\}$. Then we will want to apply the theorems of $\mathcal{M}$ (under these assumptions) to the structure that $\langle s_0, x_0, f_0\rangle$ represents.

Context Theory Interpretations. To see how this can be accomplished, consider the syntactic translation I that sends the domain to s_0, the constant e to x_0, and the operator $\oplus$ to f_0. I is unlike the most usual notion of interpretation in two ways:

- The expressions s_0, x_0 and f_0 may involve free variables, so that the image of a closed expression under I need not be closed;
- If the set of assumptions $\{\varphi_1, \ldots, \varphi_n\}$ is non-empty, then the images of the monoid axioms under I may not be theorems of $\mathcal{B}$.

Although I is not properly a theory interpretation, it is what we call a *context theory interpretation* from $\mathcal{M}$ to $\mathcal{B}$ relative to the "context" [22, 9] containing the assumptions $\{\varphi_1, \ldots, \varphi_n\}$. Context theory interpretations are used in

IMPS in much the same as way as ordinary theory interpretations, so long as our position in a proof licenses us to use the assumptions $\{\varphi_1, \ldots, \varphi_n\}$. In the case at hand, they justify citing the theorems of $\mathcal{M}$ in reasoning about s_0, x_0 and f_0.

Context theory interpretations greatly extend the power of the little theories approach. As a consequence, a commitment to the little theories approach need entail no reduplication of effort when structures within **ZF** are of interest.[7]

Proving Consistency. An objection is often raised to the free use of axiomatic theories in the formal verification of software or hardware. All but the most sophisticated users will introduce axioms that are wrong, and sometimes even inconsistent.

Naturally, no formal method can prevent a user from writing a specification that does not accurately reflect his intent. For instance, if the specification is to be built up by means of a sequence of definitions within a fixed big theory, the user may select the wrong definitions, and may sometimes even define unsatisfiable predicates. Then, although his theorems will be truths of the formal system he is working within, their content will not correspond to his intuitive understanding.

While the little theories approach does not prevent a user from doing stupid things, it is by no means worse off than the alternative. An interpretation I from $\mathcal{T}$ to $\mathcal{T}'$ will allow us to assure ourselves that a theory $\mathcal{T}$ is consistent, if $\mathcal{T}'$ is a well-established theory trusted to be consistent. Moreover, a theory interpretation also conveys more information. The user may have a conception of what sort of structures ought to satisfy $\mathcal{T}$. The interpretation I gives him a way of isolating, within any structure satisfying $\mathcal{T}'$, a structure satisfying $\mathcal{T}$. If these structures can not be made to look as he expects, then he has reason to think he has got the formulation of $\mathcal{T}$ wrong.

This method is by no means limited to formal methods, as it is also commonly used in giving relative consistency proofs in standard mathemat-

[7]Indeed, the big theories approach is particularly useful when a theorem concerns an arbitrary family of structures rather than a collection of a fixed number of structures. Since, within **ZF**, any family of structures is simply represented as another set, **ZF** is an advantageous framework for the reasoning.

By contrast, in case a theorem concerns some *fixed* number of structures of some kind, such as three arbitrary metric spaces, IMPS provides a convenient mechanism for synthesizing a suitable theory.

ical practice. For instance, the consistency of the Generalized Continuum Hypothesis with **ZF** is sometimes proved in essentially this way (see, for instance, [21]).[8] In this case, both $\mathcal{T}$ and $\mathcal{T}'$ happen to be the same theory, namely **ZF**.

5 Example

In this section we discuss an IMPS proof which exemplifies a number of advantages of the little theories approach. Our example is a well-known "inverse function theorem" (Theorem 1 below) for a mapping from a Banach space into itself which is near the identity (see [5, §10.1]). We begin by describing the sophisticated network of interrelated theories used in the proof.

5.1 The Network of Theories

The theories are constructed step-by-step using theory extension and theory instantiation (as recommended in [13]), beginning from a few general mathematical theories:

- A theory of an abstract ring.
- A theory of an abstract module over a ring.
- A theory of an abstract metric space (denoted $\mathcal{M}$).
- A theory of real arithmetic (formalized as a complete ordered field).

From these theories we build a number of other theories and interpretations:

- A theory of an abstract real vector space is obtained by instantiating the ring of the module theory with the field of real numbers via the obvious interpretation from the module theory to the theory of real arithmetic.
- A theory of an abstract normed space is built as an extension of the real vector space theory by adding a new constant (denoted $\|\cdot\|$) together with axioms which characterize this constant as a norm.
- A theory interpretation (denoted Φ) from the metric space theory $\mathcal{M}$ to the normed space theory is specified by interpreting the distance

[8]This example has not been done in IMPS.

function of $\mathcal{M}$ as the function $(x, y) \mapsto \|x - y\|$ and the sort of points of $\mathcal{M}$ as the sort of points of the normed space.

- Using Φ, the definitions in $\mathcal{M}$ of constants such as "open", "connected", and "complete" are transported to the normed space theory.

- A theory of an abstract Banach space (denoted $\mathcal{B}$) is formed by adding an axiom that says the propositional constant complete-normed-space holds.

Thus, this one theory $\mathcal{B}$, in which the theorem will be stated and proved, illustrates a variety of ways that theories can be built in IMPS.

5.2 The Theorem and Its Proof

Before stating the theorem we make some preliminary definitions.

We will use the symbols $\mathbf{P}$ and d to denote the sort of points and distance function in the metric space theory $\mathcal{M}$, and $\mathbf{V}$ and $\| \ \|$ to denote the sort of vectors and norm in the Banach space theory $\mathcal{B}$. A partial function $\varphi : \mathbf{P} \to \mathbf{P}$ is a *contraction with modulus* $c < 1$ if, for all x, y in the domain of φ, $d(\varphi(x), \varphi(y)) < c \cdot d(x, y)$. For $a \in \mathbf{P}$ and $r \in \mathbf{R}$, $\mathrm{B}(a, r)$ and $\bar{\mathrm{B}}(a, r)$ denote, respectively, the open and closed balls with center a and radius r. As illustrated above, these definitions can be readily transported via Φ to the normed space theory and thus to $\mathcal{B}$. Given a partial function $\varphi : \mathbf{V} \to \mathbf{V}$, $f_\varphi : \mathbf{V} \to \mathbf{V}$ is the partial function $x \mapsto x + \varphi(x)$, provided $\varphi(x)$ is defined.

Theorem 1 *Let $\varphi : \mathbf{V} \to \mathbf{V}$ be a contraction whose domain is open. Then the range of f_φ is itself open.*[9]

The key idea behind the proof of this result is the following classical theorem of abstract analysis attributed to Banach:

Theorem 2 (Contraction Principle) *A contraction on a complete metric space has a unique fixed point.*

The Contraction Principle is proved in IMPS in the theory $\mathcal{M}$. The proof is very similar to the standard textbook proof. From the perspective of machine-aided deduction, this proof is noteworthy for two reasons: First it

[9] From this conclusion, it is easy to show that the mapping is also a homeomorphism (i.e., that it is continuous with a continuous inverse).

involves reasoning at various levels of abstraction; secondly, it uses widely different proof techniques. Specifically, the proof uses definitions and properties naturally stated in a theory of an abstract metric space at the same time that it uses assorted facts about the real numbers. Some of these facts, such as the convergence to zero of r^n as $n \to \infty$ (which follows from Bernoulli's inequality) involve the order, metric, and algebraic structure of the real numbers. Other pertinent facts have a purely algebraic nature, such as the geometric series formula,

$$\sum_{j=p}^{q} r^j = \frac{r^p \cdot (1 - r^{q-p+1})}{1 - r},$$

which holds provided $0 \leq p \leq q$ and $r \neq 0, \neq 1$. Bernoulli's inequality and the geometric series formula are in turn proved using induction and algebraic and order properties of the reals.

The proof of Theorem 1 requires the following two lemmas:

Lemma 1 *Suppose the metric space* $(\mathbf{P}, d)$ *is complete;* $a \in \mathbf{P}$*;* $r \in \mathbf{R}$ *with* $0 \leq r$*;* $\varphi : \mathbf{P} \to \mathbf{P}$ *is a contraction with modulus* c *that is defined on* $\bar{\mathrm{B}}(a, r)$*; and* $d(a, \varphi(a)) < r \cdot (1 - c)$*. Then* φ *has a fixed point in* $\bar{\mathrm{B}}(a, r)$*.*

Lemma 2 *Suppose* $a \in \mathbf{V}$*,* $r \in \mathbf{R}$ *with* $0 \leq r$*, and* $\varphi : \mathbf{V} \to \mathbf{V}$ *is a contraction with modulus* c *that is defined on* $\bar{\mathrm{B}}(a, r)$*. Then*

$$\mathrm{B}(f_\varphi(a), r \cdot (1 - c)) \subseteq f_\varphi(\bar{\mathrm{B}}(a, r)).$$

Proof of Theorem 1 in IMPS**.** First, Lemma 1 is stated and proved in $\mathcal{M}$ as follows. For technical convenience, a new theory $\mathcal{M}'$ is constructed by adding individual constants a and r of sort $\mathbf{P}$ and $\mathbf{R}$, respectively, and the axiom $0 \leq r$ to $\mathcal{M}$. A version θ of Lemma 1 is formulated in $\mathcal{M}'$ by taking a and r in the statement of the lemma to be individual constants instead of universally quantified variables. θ is derived in $\mathcal{M}'$ from an appropriate instantiation of the Contraction Principle. The instantiation is obtained by transporting the Contraction Principle from $\mathcal{M}$ to $\mathcal{M}'$ via the interpretation which interprets the sort $\mathbf{P}$ as $\bar{\mathrm{B}}(a, r)$ and the distance function d as the restriction of d to $\bar{\mathrm{B}}(a, r)$. To get a version of Lemma 1 in $\mathcal{M}$, in which a and r are universally quantified variables, θ is transported from $\mathcal{M}'$ to $\mathcal{M}$ via a context interpretation that is created automatically by IMPS.

Next, Lemma 1 is transported to $\mathcal{B}$ via the interpretation Φ defined above. Lemma 2 is derived in $\mathcal{B}$ from the translation of Lemma 1 by unfolding the definition of open set and then performing, in a straightforward manner, some user-directed logical reasoning.

Finally, Theorem 1 follows from Lemma 2 by a straightforward proof in $\mathcal{B}$.$\Box$

Another theorem which can be proved with a technique similar to the one we have outlined here is the Picard-Lindelöf existence theorem for ordinary differential equations (see [5]). Its proof involves an application of the Contraction Principle to a space of continuous functions on an interval. Much of this proof (which involves the construction of a large network of theories) has already been carried out by Robert Givan using the IMPS system.

6 Conclusion

Mathematics of any complexity requires a mixture of strikingly different kinds of reasoning. For instance, the proof of the Contraction Principle depends on the numerical properties of the reals, such as the geometric series formula, as well as using induction on integers and the much more abstract properties of a metric space. A sequence of points must be built up by applying a second order iteration operator to the contractive mapping, and an ϵ/δ argument must be given to show that the sequence is Cauchy. Algebra, ordering properties, and many lemmas must be used to compute with the various subgoals that arise. The proof illustrates what Wittgenstein called the "motley of mathematics."

Little theories provide a way of organizing this variety both at the implementation level and at the conceptual level.

References

[1] R. S. Bird. An introduction to the theory of lists. Technical Report PRG-56, Oxford University Computing Laboratory, 1986.

[2] R. Burstall and J. Goguen. Putting theories together to make specifications. In *Fifth International Joint Conference on Artificial Intelligence*, 1977.

[3] R. Burstall and J. Goguen. The semantics of Clear, a specification language. In *Advanced Course on Abstract Software Specifications*. Springer-Verlag, 1980.

[4] R. Burstall and J. Goguen. An informal introduction to specifications using Clear. In R. Boyer and J Moore, editors, *The Correctness Problem in Computer Science*, pages 185–213. Academic Press, 1981.

[5] J. Dieudonné. *Foundations of Modern Analysis*. Academic Press, 1960.

[6] H. B. Enderton. *A Mathematical Introduction to Logic*. Academic Press, 1972.

[7] W. M. Farmer. A partial functions version of Church's simple theory of types. *Journal of Symbolic Logic*, 55:1269–91, 1990.

[8] W. M. Farmer. A simple type theory with partial functions and subtypes. Technical report, The MITRE Corporation, 1991. Submitted to *Annals of Pure and Applied Logic*.

[9] W. M. Farmer, J. D. Guttman, and F. J. Thayer. IMPS: an Interactive Mathematical Proof System. Technical Report M90-19, The MITRE Corporation, 1991. Submitted to the *Journal of Automated Reasoning*.

[10] W. M. Farmer, J. D. Guttman, and F. J. Thayer. IMPS: System description. In D. Kapur, editor, *Automated Deduction—CADE-11*, volume 607 of *Lecture Notes in Computer Science*, pages 701–705. Springer-Verlag, 1992.

[11] W. M. Farmer and F. J. Thayer. Two computer-supported proofs in metric space topology. *Notices of the American Mathematical Society*, 38:1133–1138, 1991.

[12] S. J. Garland and J. V. Guttag. A guide to LP, the Larch prover. Technical report, MIT, 1991. Available by ftp from larch.lcs.mit.edu.

[13] J. A. Goguen. Principles of parameterized programming. Technical report, SRI International, 1987.

[14] J. A. Goguen and T. Winkler. Introducing OBJ3. Technical Report SRI-CSL-99-9, SRI International, August 1988.

[15] E. Gunter. The implementation and use of abstract theories in HOL. In *Third HOL Users Meeting*. Computer Science Department, Aarhus University, 1990.

[16] J. V. Guttag, J. J. Horning, and J. M. Wing. Larch in five easy pieces. Technical Report 5, Digital Systems Research Center, 1985.

[17] J. D. Guttman. A proposed interface logic for verification environments. Technical Report M91-19, The MITRE Corporation, 1991.

[18] R. Harper, D. Sannella, and A. Tarlecki. Structure and representation in LF. In *Fourth Annual Symposium on Logic in Computer Science*, pages 226–37. IEEE Computer Society, 1989.

[19] D. Hilbert. *The Foundations of Geometry.* Open Court, Chicago, 1902.

[20] N. Jacobson. *Basic Algebra I.* W. H. Freeman, second edition, 1985.

[21] J. Krivine. *Introduction to Axiomatic Set Theory.* Reidel, Dordrecht, 1971.

[22] L. G. Monk. Inference rules using local contexts. *Journal of Automated Reasoning*, 4:445–462, 1988.

[23] J. Mycielski. A lattice of interpretability types of theories. *Journal of Symbolic Logic*, 42, 1977.

[24] D. Sannella and R. Burstall. Structured theories in LCF. In *CAAP'83: Trees in Algebra and Programming, 8th Colloquium*, volume 159 of *Lecture Notes in Computer Science*, pages 377–91. Springer-Verlag, 1983.

[25] J. R. Shoenfield. *Mathematical Logic.* Addison-Wesley, 1967.

[26] L. W. Szczerba. Interpretability of elementary theories. In R. E. Butts and J. Hintikka, editors, *Logic, Foundations of Mathematics, and Computability Theory*, pages 129–145. Reidel, 1977.

[27] A. Tarski, A. Mostowski, and R. M. Robinson. *Undecidable theories.* North-Holland, 1953.

[28] J. van Bentham and D. Pearce. A mathematical characterization of interpretation between theories. *Logica Studia*, 43:295–303, 1984.

[29] A. Visser. The formalization of interpretability. *Studia Logica*, 50:81–105, 1991.

Contents