IMPS: System Description*

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

The MITRE Corporation

3 November 1992

IMPS, an Interactive Mathematical Proof System [5], aims at computational support for traditional techniques of mathematics. It is based on three observations about rigorous mathematics:

- Mathematics emphasizes the axiomatic method. Characteristics of mathematical structures are summarized in axioms. Theorems are derived for all structures satisfying the axioms. Frequently, a clever change of perspective shows that a structure is an instance of another theory, thus also bringing its theorems to bear.
- Many branches of mathematics emphasize functions, including partial functions. Moreover, the classes of objects studied may be nested, as are the integers and the reals; or overlapping, as are the bounded functions and the continuous functions.
- Proof proceeds by a blend of computation and formal inference.

Support for the Axiomatic Method. IMPS supports the "little theories" version of the axiomatic method, as opposed to the "big theory" version. In the big theory approach, all reasoning is carried out within one theory—usually some highly expressive theory, such as the Zermelo-Fraenkel set theory. In the little theories approach, reasoning is distributed over a network of theories. Results are typically proved in compact, abstract theories, and then transported as needed to more concrete theories, or indeed to

*Supported by the MITRE-Sponsored Research Program. Published in: D. Kapur, ed., *Automated Deduction—CADE-11*, Lecture Notes in Computer Science, vol. 607, Springer-Verlag, 1992, pp. 701–705.

other equally abstract theories. Theory interpretations provide the mechanism for transporting theorems. The little theories style of the axiomatic method is employed extensively in mathematical practice; in [4], we discuss its benefits for mechanical theorem provers, and how the approach is used in IMPS.

Logic. Standard mathematical reasoning in many areas focuses on functions and their properties, together with operations on functions. For this reason, IMPS is based on a version of simple type theory.[1] However, we have adopted a version, called LUTINS,[2] containing partial functions, because they are ubiquitous in both mathematics and computer science. Although terms, such as $2/0$, may be nondenoting, the logic is bivalent and formulas always have a truth value. In particular, an atomic formula is false if any of its constituents is nondenoting. This convention follows an approach common in traditional rigorous mathematics, and it entails only small changes in the axioms and rules of classical simple type theory [2].

Moreover, LUTINS allows subtypes to be included within types. Thus, for instance, the natural numbers form a subtype of the reals, and the continuous (real) functions a subtype of the functions from reals to reals. The subtyping mechanism facilitates machine deduction, because the subtype of an expression gives some immediate information about the expression's value, if it is defined at all. Moreover, many theorems have restrictions that can be stated in terms of the subtype of a value, and the theorem prover can be programmed to handle these assertions using special algorithms [4].

This subtyping mechanism interacts well with the type theory only because functions may be partial. If σ_0 is a subtype of τ_0, while σ_1 is a subtype of τ_1, then $\sigma_0 \to \sigma_1$ is a subtype of $\tau_0 \to \tau_1$. In particular, it contains just those partial functions that are never defined for arguments outside σ_0, and which never yield values outside σ_1.

[1] This version is *many-sorted*, in that there may be several types of basic individuals. Moreover, it is *multivariate*, in that a function may take more than one argument. Currying is not required. However, taking (possibly n-ary) functions is the only type-forming operation.

[2] Pronounced as in French. See [2, 3] for studies of logical issues associated with LUTINS; see [7] for a detailed description of its syntax and semantics.

Computation and Proof. One problem in understanding and controlling the behavior of theorem provers is that a derivation in predicate logic contains too much information.

The mathematician devotes considerable effort to proving lemmas that justify computational procedures. Although these are frequently equations or conditional equations, they are sometimes more complicated quantified expressions, and sometimes they involve other relations, such as ordering relations. Once the lemmas are available, they are used repeatedly to transform expressions of interest. Algorithms justified by the lemmas may also be used; the algorithm for differentiating polynomials, for example. The mathematician has no interest in those parts of a formal derivation that "implement" these processes within predicate logic.

On the other hand, to understand the structure of a proof (or especially, a partial proof attempt), the mathematician wants to see the premises and conclusions of the informative formal inferences.

Thus, the right sort of proof is broader than the logician's notion of a formal derivation in, say, a Gentzen-style formal system. In addition to purely formal inferences, IMPS allows also inferences based on sound computations. They are treated as atomic inferences, even though a pure formalization might require hundreds of Gentzen-style formal inference steps. We believe that this more inclusive conception makes IMPS proofs more informative to its users.

The System

The IMPS system consists of four components.

Core. All the basic logical and deductive machinery of IMPS on which the soundness of the system depends is included in the *core* of IMPS. The core is the specification and inference engine of IMPS. The other components of the system provide the means for harnessing the power of the engine.

The organizing unit of the core is the IMPS *theory*. Mathematically, a theory consists of a LUTINS language plus a set of axioms. A theory is implemented, however, as a data structure to which a variety of information is associated. Some of this information procedurally encodes logical consequences of the theory that are especially relevant to low-level reasoning within the theory. For example, the great majority of questions about the definedness of expressions are answered automatically by IMPS using tabulated information about the domain and range of the functions in a theory.

Theories may be enriched by the definition of new sorts and constants and by the installation of theorems.

Proofs in a theory are constructed interactively using a natural style of inference based on sequent calculus. IMPS builds a data structure which records all the actions and results of a proof attempt. This object, called a *deduction graph*, allows the user to survey the proof and to choose the order in which he works on different subgoals. Alternative approaches may be tried on the same subgoal. Deduction graphs also are suitable for analysis by software.

The user is only allowed to modify a deduction through the application of *primitive inferences*, which are akin to rules of inference. Most primitive inferences formalize basic laws of predicate logic and higher-order functions. Others implement computational steps in proofs. For example, one class of primitive inferences performs expression simplification, which uses the logical structure of the expression [8], together with algebraic simplification, deciding linear inequalities, and applying rewrite rules.

Another special class of primitive inferences "compute with theorems" by applying extremely simple procedures called macetes [9].[3] An *elementary macete*, which is built by IMPS whenever a theorem is installed in a theory, applies the theorem in a manner determined by its syntax (e.g., as a conditional rewrite rule). Compound macetes are constructed from elementary and other kinds of atomic macetes (including macetes that beta-reduce, unfold defined constants, and perform expression simplification). They apply a collection of theorems in an organized way.

In addition to the machinery for building theories and reasoning within them, the core contains machinery for relating one theory to another via theory interpretations. A theory interpretation can be used to "transport" a theorem from the theory it is proved in to any number of other theories. Theory interpretations are also used in IMPS to show relative consistency of theories, to formalize symmetry and duality arguments, and to prove universal facts about polymorphic operators [5, 4, 1]. The great majority of the theory interpretations needed by the IMPS user are built by software without user assistance. For example, when a theorem is applied outside of its home theory via a *transportable macete*, IMPS automatically builds the required theory interpretation if needed.

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

Supporting Machinery. We have built an extension of the core machinery with the following goals in mind:

- To facilitate building and printing of expressions by providing various parsing and printing procedures.
- To make the inference mechanism more flexible and more autonomous. In particular, the set of commands available to users includes an extensible set of inference procedures called *strategies*. Strategies affect the deduction graph only by using the primitive inference procedures of the core machinery, but are otherwise unrestricted.
- To facilitate construction of theories and interpretations between them.

User Interface. From the outset IMPS has been designed to provide users with facilities to easily direct and monitor proofs. This is accomplished by a user interface which controls three critical elements of an interactive system:

- The display of the state of the proof. This includes graphical displays of the deduction graph as a tree, TEX typesetting of the proof history, and TEX typesetting of individual subgoals in the deduction graph. The graphical display of the deduction graph permits the user to visually determine the set of unproven subgoals and to select a suitable continuation point for the proof. On the other hand, the TEX typesetting facilities allow the user to examine the proof in a mathematically more appealing notation than is possible by raw textual presentation alone.
- The presentation of options for new proof steps. For any particular subgoal, the interface presents the user with a well-pruned list of commands and macetes to apply. This list is obtained by using syntactic and semantic information which is made available to the interface by the IMPS supporting machinery. For example, in situations where over 400 theorems are available to the user, there are rarely more than 10 macetes presented to the user as options.
- Processing of user commands and submitting them to the inference software, requesting from the user whenever necessary, additional arguments required to execute the command.

The user interface, which is a completely detachable component of IMPS, is primarily written in Emacs; the other components are written in T, a sophisticated version of Scheme.

Theory Library. IMPS is equipped with a library of basic theories, which can augmented as desired by the user. The theory of the reals, the central theory in the library, specifies the real numbers as a complete ordered field. (The completeness principle is formalized as a second-order axiom, and the rationals and integers are specified as the usual substructures of the reals.) The library also contains various "generic" theories that contain no nonlogical axioms (except possibly the axioms of the theory of the reals). These theories are used for reasoning about objects such as sets, unary functions, and sequences.

Applications

We have formulated a variety of different kinds of mathematics within IMPS, with emphasis on mathematical analysis and abstract algebra. In our theory of the reals, we have proved a large number of basic facts such as the combinatory identity, Bernoulli's inequality, and the Archimedean property of the reals. In analysis, we have proved Banach's Contraction Principle in a theory of an abstract metric space, and the theorem that the continuous image of a connected set is itself connected in a theory of two metric spaces (see [6]). The proof in IMPS of a simple "inverse function theorem" in a theory of an Banach space is described in [4]. In algebra, we have proved a version of the binomial theorem for commutative rings, and various facts about an abstract iterated product operator in a theory of an abstract monoid, which can be transported to the reals as theorems about the Σ and Π operators. The Schröder-Bernstein Theorem and Fundamental Counting Theorem of group theory, of which Lagrange's theorem is an immediate corollary, have also been proved in IMPS.

References

[1] W. M. Farmer. Abstract data types in many-sorted second-order logic. Technical Report M87-64, The MITRE Corporation, 1987.

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

[3] W. M. Farmer. A simple type theory with partial functions and subtypes. *Annals of Pure and Applied Logic*, 64:211–240, 1993.

[4] W. M. Farmer, J. D. Guttman, and F. J. Thayer. Little theories. In D. Kapur, editor, *Automated Deduction—CADE-11*, volume 607 of *Lecture Notes in Computer Science*, pages 567–581. Springer-Verlag, 1992.

[5] W. M. Farmer, J. D. Guttman, and F. J. Thayer. IMPS: An Interactive Mathematical Proof System. *Journal of Automated Reasoning*, 11:213–248, 1993.

[6] 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.

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

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

[9] F. J. Thayer. Obligated term replacements. Technical Report MTR-10301, The MITRE Corporation, 1987.