IMPS: An Updated System Description*

William M. Farmer[1], Joshua D. Guttman[1], F. Javier Thayer Fábrega[2]

[1] The MITRE Corporation, 202 Burlington Road, Bedford, MA 01730-1420, USA

[2] Mitretek Systems, 25 Burlington Mall Road, Burlington, MA 01803, USA

Telephone: 617-271-2907

E-mail: {farmer,guttman,jt}@mitre.org

IMPS, an Interactive Mathematical Proof System, is intended to provide mechanical support for traditional mathematical techniques and styles of practice. The system consists of a library of axiomatic theories and a collection of tools for exploring and extending the mathematics embodied in the theory library. One of the chief tools is a facility for developing formal proofs. IMPS is equally well-suited for applications in mathematics education and in the development of high assurance hardware and software.

The IMPS system is available without fee (under the terms of a public license) at the ftp site `math.harvard.edu` and at the following Web pages:

```
file://math.harvard.edu/imps/imps_html/imps.html
http://www.tiac.net/users/thayer/imps.html
```

Documentation for the system is also available at the IMPS ftp site and Web pages. This includes a substantial user's manual, a full IMPS bibliography, and several IMPS papers. The principal IMPS publications are listed below in the **References** section. Reference [10] presents an overview of the system.

The IMPS system has changed little since Version 1.2 was released in July 1994. Since that time, a new graphical user interface has been developed (see the **User Interface** section), the theory library has been expanded (see the **Applications** section), and a few minor bugs have been fixed.

Distinguishing Characteristics

IMPS is based on four characteristics of mathematics practice:

- The axiomatic method is the dominant organizing principle in mathematics.
- The development of a piece of mathematics quite often involves many different kinds of mathematical knowledge.
- Functions, including partial and higher-order functions, are widely used to represent mathematical objects and rules.
- Proofs are a blend of computation and deduction.

* Supported by the MITRE-Sponsored Research program. Published in: M. McRobbie and J. Slaney, eds., *Automated Deduction—CADE-13*, *Lecture Notes in Computer Science*, Vol. 1104, Springer-Verlag, Berlin, 1996, pp. 298–302.

Support for the Axiomatic Method. IMPS emphasizes 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 linked by theory interpretations. Results are typically proved in compact, abstract theories, and then transported as needed to more concrete theories, or indeed to other equally abstract theories. The theory interpretations provide the mechanism for transporting theorems. The little theories style of the axiomatic method is employed extensively in mathematical practice; in [9] we discuss its benefits for mechanized mathematics systems and how the approach is used in IMPS.

Theory Library. The IMPS theory library is a collection of theories, theory interpretations, and theory constituents (e.g., definitions and theorems) which serves as a database of mathematics. Containing a large amount and variety of basic mathematics, it offers the user a well-developed starting point for developing his or her own mathematics. It includes formalizations of the real number system and objects like sets and sequences; theories of abstract mathematical structures such as groups and metric spaces; and theories to support specific applications of IMPS in computer science.

Logic. The IMPS logic is a nonconstructive version of simple type theory called LUTINS[1] [1, 2, 3]. Since partial functions are ubiquitous in both mathematics and computer science, LUTINS allows partial functions to be represented directly. As a consequence, terms like $2/0$ may be nondenoting. However, the logic is bivalent: formulas are either true or false. In particular, the application of a predicate is false if any of the arguments is nondenoting. The handling of nondenoting terms in LUTINS follows an approach common in traditional mathematics; it entails only small changes in the axioms and rules of classical simple type theory [4].

In LUTINS, *sorts* are used to indicate the domain and ranges of *partial* functions. The denotations of sorts may overlap and are included in the denotations of types. Thus, for instance, the natural numbers form a subsort of the real numbers, and the partial functions from the integers to the rationals form a subsort of the partial functions from the reals to the reals. Every term is assigned a sort (and a type). This facilitates machine deduction since the sort of a term gives some immediate information about its value: a term t has sort α means that, *if t is denoting*, the value of t is contained in the denotation of α. Sorts are also used to restrict the binding operators of LUTINS such as λ and $\forall$.

Computation and Deduction. In contrast to the formal proofs described in logic textbooks, IMPS proofs are a blend of computation and deduction. Consequently, they resemble intelligible informal proofs, but unlike informal proofs, all the details of an IMPS proof are machine checked. Proofs are constructed interactively using a natural style of inference based on sequent calculus. IMPS builds a data

[1] Pronounced as the word in French.

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 or she works on different subgoals. Alternative approaches may be tried on the same subgoal. Deduction graphs also are suitable for analysis by software.

Low-level inferences can be performed by two kinds of computation: automatic expression simplification and semi-automatic theorem application via procedures called *macetes*. Both kinds of computation utilize algorithms whose actions depend on a "local context" of assumptions [12, 17]. High-level inferences are performed by applying proof commands or by executing proof scripts that apply a sequence of proof commands in an intelligent manner [6]. Proof commands (and thus proof scripts) may include calls to the simplifier and to macetes.

User Interface

The IMPS user interface provides the user with three critical facilities:

- Convenient mechanisms for processing user commands and submitting them to the IMPS theory development and theorem proving software.
- The means to present the state of an interactive proof attempt in various forms: as a graphical display of the corresponding deduction graph, as a history of events typeset in TEX, and as a script that can be easily edited and then executed to produce a new deduction graph [6].
- A procedure that, for a given subgoal in a deduction graph, presents to the user a well-pruned list of the macetes (both inside and outside the home theory of the deduction graph) that may be applicable to the subgoal.

The user interface is a completely detachable component of IMPS. In fact, there are currently two supported interfaces for IMPS, one written primarily in Emacs version 19 from the Free Software Foundation, and another graphical interface written in TCL/Tk. The Emacs interface is documented in the IMPS user's manual [11].

The graphical interface has a number of innovative characteristics which distinguish it from the Emacs interface. In addition to providing more graphical cues for input, the new interface has a client-server design which allows IMPS to be used over a TCP connection from many hardware platforms. Moreover, this design allows not only interactive use, but also permits various users at different locations to work cooperatively on the same problem.

Applications

IMPS was designed for use in three main areas, namely mathematics, education, and formal methods. The IMPS theory library contains a variety of theories to support these purposes, and some that illustrate how to carry out tasks of these kinds. Some additional theories were developed purely to support other activities that we were pursuing.

Mathematics

The library develops a substantial part of the core of mathematics. A theory of the real numbers forms the basis for the development and is a building block in almost all other theories. It axiomatizes the reals as a complete ordered field, using a second-order axiom to express completeness. The rationals, integers, and natural numbers are introduced as sorts included within the reals. Since the theory requires partial functions (such as division), higher-order functions (such as the limit of a sequence), and recursively defined functions (such as summation), most of the advantages of LUTINS are already illustrated in this theory. The theory of the reals is used to develop theories of metric spaces, normed spaces, and vector spaces.

Basic facts about divisibility and primes lead to the infinitude of primes, the fundamental theorem of arithmetic, and the principal ideal theorem for the integers.

A generic "little theory" of sets, cardinality, and finiteness is widely used by means of theory interpretations.

Basic calculus is introduced for functions from the reals to an arbitrary normed space. This includes the mean value theorem for derivatives and integrals, the fundamental theorem of calculus, and other basic properties of the derivative and integral. Banach's fixed-point theorem for contractive mappings leads to a number of applications, such as an open mapping theorem for mappings on a Banach space which are near the identity (see [9]).

A theory of partial orders leads to the Knaster fixed-point theorem and applications. One application is a conceptually simple proof of the Schröder-Bernstein theorem. A more traditional proof of the Schröder-Bernstein theorem is developed elsewhere. The fundamentals of group theory are developed, leading to the Fundamental Counting Theorem of group theory. Lagrange's theorem and related facts are derived from this. A portion of elementary geometry based only on betweenness leads to a proof of Sylvester's theorem.

Work related to mathematics education, apart from the material in geometry, include a variety of exercises illustrating IMPS's applicability to learning calculus, number theory, and aspects of computer science.

Formal Methods

The portion of the theory library devoted to formal methods includes theories formalizing various notions about state machines. These are used to prove correctness theorems for aspects of the Mach memory management system, including the use of demand-paged virtual memory [16] and the copy-on-write optimization [19].

The mathematical library was used to good effect in the course of developing a semantical theory suited for timed CSP [18]. The theory of the reals was extended to include an abstract theory of machine-representable integers. This became the basis for a software system [14] for formally analyzing computer programs (written in a language called PreScheme [7]) that manipulate machine integers.

References

1. W. M. Farmer. A partial functions version of Church's simple theory of types. *Journal of Symbolic Logic*, 55:1269–91, 1990.
2. W. M. Farmer. A simple type theory with partial functions and subtypes. *Annals of Pure and Applied Logic*, 64:211–240, 1993.
3. W. M. Farmer. Theory interpretation in simple type theory. In J. Heering et al., editor, *Higher-Order Algebra, Logic, and Term Rewriting*, volume 816 of *Lecture Notes in Computer Science*, pages 96–123. Springer-Verlag, 1994.
4. W. M. Farmer. Reasoning about partial functions with the aid of a computer. *Erkenntnis*, 43:279–294, 1995.
5. W. M. Farmer and J. D. Guttman. A set theory with support for partial functions. In E. Thysse and F. Lepage, editors, *Partial, Epistemic, and Dynamic Logic*, Applied Logic Series. Kluwer. Forthcoming.
6. W. M. Farmer, J. D. Guttman, M. E. Nadel, and F. J. Thayer. Proof script pragmatics in IMPS. In A. Bundy, editor, *Automated Deduction—CADE-12*, volume 814 of *Lecture Notes in Computer Science*, pages 356–370. Springer-Verlag, 1994.
7. W. M. Farmer, J. D. Guttman, and J. D. Ramsdell. A verified compiler for Multithreaded PreScheme. Technical Report MP-95B370, The MITRE Corporation, 1995.
8. 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.
9. 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.
10. W. M. Farmer, J. D. Guttman, and F. J. Thayer. IMPS: An Interactive Mathematical Proof System. *Journal of Automated Reasoning*, 11:213–248, 1993.
11. W. M. Farmer, J. D. Guttman, and F. J. Thayer. The IMPS user's manual. Technical Report M-93B138, The MITRE Corporation, 1993. Available at `http://imps.mcmaster.ca/`.
12. W. M. Farmer, J. D. Guttman, and F. J. Thayer. Contexts in mathematical reasoning and computation. *Journal of Symbolic Computation*, 19:201–216, 1995.
13. 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.
14. W. M. Farmer and F. J. Thayer. Formal numerical program analysis. Technical report, The MITRE Corporation, 1994.
15. J. D. Guttman. A proposed interface logic for verification environments. Technical Report M91-19, The MITRE Corporation, 1991.
16. J. D. Guttman and D. M. Johnson. Three applications of Formal Methods at MITRE. In M. Naftalin, T. Denvir, and M. Bertran, editors, *FME '94: Industrial Benefits of Formal Methods*, volume 873 of *Lecture Notes in Computer Science*, pages 55–65. Springer Verlag, 1994.
17. L. G. Monk. Inference rules using local contexts. *Journal of Automated Reasoning*, 4:445–462, 1988.
18. F. J. Thayer. An approach to process algebra using IMPS. Technical Report MP-94B193, The MITRE Corporation, 1994.
19. F. J. Thayer Fábrega and J. D. Guttman. Copy on write. Technical report, The MITRE Corporation, 1995.

This article was processed using the LaTeX macro package with LLNCS style