CS 773 Winter 2002

07. Introduction to IMPS

Instructor: W. M. Farmer

Revised: 8 February 2002

What is IMPS?

- IMPS is an **Interactive Mathematics Proof System** developed at The MITRE Corporation by W. Farmer, J. Guttman, and J. Thayer
- Principal goals:
  - Mechanize mathematical reasoning
  - Be useful to a wide range of people
- Approach:
  - Support traditional mathematical techniques
  - Human oriented instead of machine oriented
- Main application areas:
  - Mathematics education
  - Hardware and software development

Distinguishing Characteristics of IMPS

1. Logic that admits partial functions and undefined terms
   - Closely corresponds to mathematical practice
2. Proofs that combine deduction and calculation
   - IMPS proof system is eclectic
   - Calculation plays as essential role in IMPS proofs
3. Little theories method for organizing mathematics
   - Essential for formalizing large portions of mathematics

Goals for the IMPS Logic

- Familiarity: 2-valued, classical, predicate logic
- Expressiveness: higher-order quantification
- Support for functions:
  - Higher-order and partial functions
  - λ-notation
  - Definite description operator
- Simple type system:
  - No explicit polymorphism
  - Subtype system for classifying expressions by value

LUTINS, the Logic of IMPS

- Satisfies all the goals for the IMPS logic
- A version of Church's simple type theory with:
  - Traditional approach to partial functions and undefinedness
  - Additional constructors, including a definite description operator
  - Sort system for classifying expressions by value
- Laws of predicate logic are modified slightly
  - Instantiation and beta-reduction are restricted to defined expressions
  - Undefined expressions are indiscernible

Traditional Approach to Partial Functions and Undefinedness

- **Expressions may be undefined**
  - Constants, variables, λ-expressions are always defined
  - Definite descriptions may be undefined: $(\mathrm{I}x:\mathbf{R}\,.\,x*x=2)$
  - Functions may be partial and thus their applications may be undefined: $1/0$, $\sqrt{-1}$
  - An application of a function is undefined if any argument is undefined: $0*(1/0)$
- **Formulas are always true or false**
  - Predicates must be total
  - An application of a predicate is false if any argument is undefined: $1/0 = 1/0$

Sorts in LUTINS

- A **sort** α is a syntactic object intended to denote a nonempty set D_α of values
- Hierarchy of sorts
  - **Atomic sorts** like **N**, **Z**, **Q**, **R**
  - **Compound sorts** of the form $\alpha_1 \times \cdots \times \alpha_n \rightharpoonup \beta$
- A compound sort $\alpha_1 \times \cdots \times \alpha_n \rightharpoonup \beta$ denotes the set of partial functions from $D_{\alpha_1} \times \cdots \times D_{\alpha_n}$ to D_β
  - Sorts are **covariant** with respect $\rightharpoonup$:
    If $\alpha \ll \alpha'$ and $\beta \ll \beta'$, then $\alpha \rightharpoonup \beta \ll \alpha' \rightharpoonup \beta'$
- Every expression E is assigned a sort $\bar{\sigma}(E)$ according to its syntax (regardless of whether it is defined or not)
  - $\bar{\sigma}(E) = \alpha$ means the value of E is in D_α if E is defined

Conjecture Proving in IMPS

- Goals:
  - User controls deductive process
  - Intelligible proofs and proof attempts
- Proofs are a blend of deduction and calculation
  - High-level reasoning orchestrated by the user
  - Low-level reasoning done automatically
- Inference steps can be large
  - Proof commands
  - Theory-specific simplification
  - Semi-automatic theorem application
  - Procedural proof scripts
- Proofs are represented in multiple ways

Simplification

- Motivation
  - Users do not want to do low-level reasoning
  - Users are generally not interested in low-level details
  - Definedness checking should not be a burden
- Simplification is used systematically in IMPS
  - To simplify subgoals in the course of a proof
  - To recognize "immediately grounded" subgoals
  - To discharge definition and interpretation obligations
- Theory specific; tailored by user
  - Algebraic and order simplification
  - Application of rewrite rules
  - Definedness checking

Macetes ("Clever Tricks")

- **Macetes** are procedures for:
  - Applying theorems to a subgoal
  - Finding which theorems are applicable
- Supplement simplification
  - Offer more control than simplification
  - Flexible way to "compute with theorems"
- **Atomic macetes**
  - Apply individual theorems (**theorem macetes**)
  - Apply special procedures: simplify, beta-reduce
- **Compound macetes**
  - Apply collections of theorems in useful patterns
  - Constructed from atomic macetes using a few simple **macete constructors**

Proof Scripts

- **Deduction graphs** can be created both "by hand" and "by script"
- **Proof scripts** are used like other kinds of tactics:
  - To create new proof commands
  - To represent executable proof sketches
  - To store proofs in a compact, replayable form
- They provide an effective way to formalize and apply procedural knowledge
  - Automatically generated from deduction graphs
  - Utilize a default way of traveling through the graph
  - Can be modified by simple text editing
  - Have control structures for programming
  - Use formula patterns and "blocks" for robustness

Little Theories Method

- A complex body of mathematics is represented as a **network of axiomatic theories**
  - Bigger theories are composed of smaller theories
  - Theories are linked by interpretations
  - Reasoning is distributed over the network
- Benefits:
  - Theorems are proved at the right level of abstraction
  - Emphasizes reuse: if A is a theorem of T, then A may be reused in any "instance" of T
  - Allows multiple perspectives and parallel development
- IMPS provides stronger support for little theories than any other contemporary theorem proving system

Theory Interpretations

- A **theory interpretation** of T to T' is a mapping of the expressions of T to the expressions of T' such that theorems are mapped to theorems
- Interpretations enable theorems and definitions to be transported from abstract theories to more concrete theories or indeed to equally abstract theories
- **Interpretations are information conduits!**

General Conclusions about IMPS

1. IMPS is a partial Interactive Mathematics Laboratory
2. IMPS has introduced and tested many new ideas
3. IMPS has demonstrated that good system engineering is as important as good logical and deductive machinery
4. IMPS is inaccessible to most mathematics practitioners
5. IMPS indicates the profound impact that mechanized mathematics systems can have on mathematics practice

Availability of IMPS

- The IMPS system is available to the public without fee under a public license
  - System includes documentation and source code
  - Web site: `http://imps.mcmaster.ca`
- Newest version: IMPS 2.0
  - Written in Common Lisp
  - Runs on Unix platforms
  - User interface requires X Windows and XEmacs