MathScheme: Project Description*

Jacques Carette, William M. Farmer, and Russell O'Connor**

Department of Computing and Software
McMaster University
Hamilton, Ontario, Canada

The mission of mechanized mathematics is to develop software systems that support the process people use to create, explore, connect, and apply mathematics. Working mathematicians routinely leverage a powerful synergy between deduction and computation. The artificial division between (axiomatic) theorem proving systems and (algorithmic) computer algebra systems has broken this synergy. To significantly advance mechanized mathematics, this synergy needs to be recaptured within a single framework. MathScheme [6] is a long-term project being pursued at McMaster University with the aim of producing such a framework in which formal deduction and symbolic computation are tightly integrated. In the short-term, we are developing tools and techniques to support this approach, with the long-term objective to produce a new system.

Towards this aim, we have already developed several techniques, with some laying the theoretical foundations of our framework, while others are implementation techniques. In particular, we rely on biform theories and an expressive logic (Chiron) for grounding. We rely on various meta-programming techniques as well as the increased safety offered by a modern statically typed programming language (Objective Caml [8]) to greatly simplify our implementation burden.

A **biform theory** [1, 3] is a combination of an axiomatic theory and an algorithmic theory. It is the basic unit of mathematical knowledge that consists of a set of *concepts*, *transformers*, and *facts*. The concepts are symbols that denote mathematical values and, together with the transformers, form a language L for the theory. The transformers are programs whose input and output are expressions of L. Transformers represent syntax-manipulating operations such as inference and computation rules. The facts are statements expressed in L about the concepts and transformers. In a typical biform theory, the concepts are classified as primitive or defined, the transformers as primitive or derived, and the facts as axioms, definitions, or theorems. A pure axiomatic theory is a biform theory with no transformers, and a pure algorithmic theory is a biform theory with no facts or only facts about the transformers.

Since transformers manipulate the syntax of expressions, biform theories are difficult to formalize in a traditional logic without the means to reason about syntax. **Chiron** [4, 5] is a derivative of von-Neumann-Bernays-Gödel (NBG) set theory that is intended to be a practical, general-purpose logic for mechanizing mathematics. It is equipped with a type system that includes dependent types, subtypes, and possibly empty types. It handles undefined expressions according

* This research was supported by NSERC.

** `{carette,wmfarmer}@mcmaster.ca`, `roconnor@theorem.ca`.

to the *traditional approach to undefinedness*. Its most noteworthy component is a facility for reasoning about the syntactic structure of expressions using quotation and evaluation *à la* Lisp. We have an implementation [7] of Chiron which uses the Objective Caml type system to track most of the invariants of Chiron expressions statically. It includes a convenient foreign function interface (to facilitate building of biform theories out of existing libraries) and a sophisticated set of pretty-printers for rendering Chiron in ASCII, MathML, and LaTeX.

We also have an implementation of the **MathScheme Language**, which has a user-oriented, high-level syntax (unlike Chiron), influenced by our work on *high-level theories* [1]. The **MathScheme Library** is an experimental formalization of the theories of abstract algebra, basic data-structures, and structured type constructors. The library is organized by the *tiny theories method* in which knowledge is distributed over a network of theories that are built up one concept at a time. Much of the structure of the library resides in theory morphisms instead of in the theories themselves. We have an expander (to see what our theories correspond to in traditional notation), a type-checker, pretty-printing facilities similar to Chiron's, as well as an experimental translator to Chiron.

Eventually, the library will form a network of biform theories interconnected by theory morphisms. Some biform theories are *implementations* for developers while others are *interfaces* for users. Meta-programming techniques [2] will be used to generate efficient implementations from the MathScheme Language.

We are actively working on MathScheme. Current work is first focusing on further leveraging the structure already present in our library to automatically generate as much information as possible, rather than implementing all of this by hand, as is traditionally done.

References

1. J. Carette and W. M. Farmer. High-level theories. In A. Autexier et al., editor, *Intelligent Computer Mathematics*, volume 5144 of *Lecture Notes in Computer Science*, pages 232–245. Springer-Verlag, 2008.
2. J. Carette and O. Kiselyov. Multi-stage programming with functors and monads: Eliminating abstraction overhead from generic code. *Science of Computer Programming*, 76(5):349 – 375, 2011.
3. W. M. Farmer. Biform theories in Chiron. In M. Kauers, M. Kerber, R. R. Miner, and W. Windsteiger, editors, *Towards Mechanized Mathematical Assistants*, volume 4573 of *Lecture Notes in Computer Science*, pages 66–79. Springer-Verlag, 2007.
4. W. M. Farmer. Chiron: A multi-paradigm logic. In R. Matuszewski and A. Zalewska, editors, *From Insight to Proof: Festschrift in Honour of Andrzej Trybulec*, volume 10(23) of *Studies in Logic, Grammar and Rhetoric*, pages 1–19. University of Białystok, 2007.
5. W. M. Farmer. Chiron: A set theory with types, undefinedness, quotation, and evaluation. SQRL Report No. 38, McMaster University, 2007. Revised 2011.
6. MathScheme Web Site. http://www.cas.mcmaster.ca/research/mathscheme/.
7. Hong Ni. Chiron: Mechanizing Mathematics in OCaml. Master's thesis, McMaster University, 2009.
8. Objective Caml. http://caml.inria.fr/.