The MathScheme Project

William M. Farmer

McMaster University

13 June 2002

The MathScheme Project

- **Objective**: To develop a new approach to mechanized mathematics in which computer theorem proving and computer algebra are integrated and generalized

- **MathScheme Group at McMaster**:
  - Bill Farmer (project leader)
  - Wolfram Kahl
  - Rheda Khedri
  - Mark Lawford
  - Martin v. Mohrenschildt
  - Dave Parnas
  - Jeff Zucker

- **Web site**: `http://imps.mcmaster.ca/mathscheme/`

Project Stages

1. **Infrastructure:**
   - Develop a **formal framework for managing mathematics**
   - Develop a **microkernel** for a mechanized mathematics system based on the framework
   - Build on the ideas embodied in IMPS and Axiom

2. **Application:** Pursue a series of **applications** that require support for formal deduction and symbolic computation

3. **Long Range:** Build an **interactive mathematics laboratory** that will support a wide range of mathematical activity

Example: Axiomatic Theory

- Let **PA** be the following axiomatic theory of Peano arithmetic:
  - **Background logic**: Second-order logic (SOL)
  - **Language**: Language of SOL with constants 0, S, $=$
  - **Axioms**:

    $\forall x \,.\, \neg(S(x) = 0)$ (0 is not a successor)

    $\forall x, y \,.\, S(x) = S(y) \supset x = y$ (S is injective)

    $\forall P \,.\, [P(0) \wedge (\forall x \,.\, P(x) \supset P(S(x)))] \supset \forall x \,.\, P(x)$ (induction axiom)

- $+$, $*$, $<$ are definable in **PA**, but simple computations require many applications of the axioms

- An infinite number of new axioms are need to define $1, 2, 3, \ldots$ in **PA**

Example: Algorithmic Theory

- Let **NNA** be the following algorithmic theory of natural number arithmetic:
  - **Background logic**: First-order logic (FOL)
  - **Language**: Language L of FOL with constants $0, 1, 2, \ldots$, $+$, $*$, $=$, $<$, `true`, and `false`
  - **Algorithms**:
    * `eval` computes the numeral that "equals" a ground term of L
    * `reduce` computes the truth value of a ground equation or inequality of L

- Computations of ground expressions are limited in efficiency only by the efficiency of `eval` and `reduce`

- Abstract expressions cannot be computed or proved using **NNA**

Principal Ideas Behind the Framework

- Facilitate the full mathematics process of creating, exploring, and connecting mathematical models
- Allow both formulas and algorithms to be assumed as axioms
- Merge deduction and computation into a single activity (called **derivation**)
- Use the **little theories method** to organized mathematics
- Allow different background logics to be used simultaneously

Transformers

- A **transformer** Π from L_1 to L_2 is an algorithm that implements a partial function $\pi : \mathcal{E}_1 \rightharpoonup \mathcal{E}_2$

- Examples of transformers:
  - Rules of inference
  - Computational rules
  - Translations between biform theories

Formuloids

- A **assertional formuloid** θ of L is a formula A of L
  - $\mathtt{span}(\theta) = \{A\}$
  - $\mathtt{operation}(\theta) = \Pi_{A \mapsto \mathtt{true}}$
- An **equational formuloid** θ of L is a transformer Π from L to L
  - $\mathtt{span}(\theta) = \{E = \Pi(E) \mid E \in \mathcal{E} \text{ and } \Pi(E) \text{ is defined}\}$
  - $\mathtt{operation}(\theta) = \Pi$

Biform Theories

- A **biform theory** is a triple $T = (\mathbf{K}, L, \Gamma)$ where:
  - $\mathbf{K}$ is an admissible background logic
  - L is a language of $\mathbf{K}$
  - Γ is a set of formuloids of L called the **axiomoids** of T

- The axiomoids are used to specify:
  - The basic objects and concepts of T
  - The basic deduction and computation rules of T

- T can be viewed as being simultaneously an axiomatic theory and an algorithmic theory
  - The set of **axioms** of T is the union of the spans of the axiomoids of T
  - The set of **algorithms** of T is set of operations of the axiomoids of T

Interpretations

- An **interpretation** of T_1 in T_2 is a transformer Φ from L_1 to L_2 that:
  - Satisfies certain syntactic conditions
  - Maps theorems of T_1 to theorems of T_2

- Interpretations are a powerful mechanism for connecting biform theories with similar structure
  - Serve as conduits for passing information (in the form of theorems) from abstract theories to more concrete theories or equally abstract theories
  - Provide the basis for the little theories method

Theoremoids

- A **theoremoid** θ of T is a formuloid of L such that $T \models A$ for each $A \in \mathtt{span}(\theta)$

- A **transformational theoremoid** is a sound deduction or computational rule

- How are new transformational theoremoids constructed?
  - Generated automatically from theorems
  - Build from other theoremoids using combinators
  - Instances of generic theoremoids

Conclusion

- A biform theory T is simultaneously an axiomatic theory and an algorithmic theory

- The axiomoids of T are the assumptions of T
  - They are expressed both declaratively and procedurally

- The theoremoids of T are deduction and computation rules for T
  - Derivations are created by applying theoremoids
  - New theoremoids are constructed from theorems and theoremoids of T via techniques that guarantee soundness

Challenges for MKM

1. Sharing mathematical knowledge
2. Expanding the MKM community
3. Ownership of mathematical knowledge
4. Internal representation
5. External presentation
6. Organization of mathematical knowledge
7. Users and applications
8. Improving the accessibility of tools
9. Certification