The MathScheme Library: Some Preliminary Experiments

Jacques Carette, William M. Farmer, Filip Jeremic, Vincent Maccio, Russell O’Connor and Quang M. Tran

McMaster University

CICM 2011, Bertinoro, Italy

22nd July 2011
Overview

- With MathScheme, we wish to **significantly advance** the capabilities of mechanized mathematics systems.

- To achieve this, we need to (at least):
  - Learn from previous systems,
  - Learn from experiments whether our own ideas will work.

- Here we present 3 such experiments in “library building”: Abstract Theories, Concrete Theories and Applied Universal Algebra

- Experimental set-up
  - Pick a topic which needs implemented,
  - Collect ideas for implementation (old and new),
  - Build enough to be be able to see patterns,
  - Evaluate results.

- See WiP paper for a lot more details.
Experiment 1: Abstract Theories

The Theories of mathematics are highly related:

\begin{align*}
\text{Ring} & := \text{combine} \ Rng, \ \text{SemiRing} \ \text{over} \ \text{Semirng} \\
\text{Field} & := \text{combine} \ \text{DivisionRing}, \ \text{CommutativeRing} \ \text{over} \ \text{Ring}
\end{align*}
Experiment 1: Abstract Theories

The **Theories** of mathematics are highly related:

\[
\text{Ring} := \text{combine } \text{Rng}, \text{SemiRing} \text{ over } \text{Semirng} \\
\text{Field} := \text{combine } \text{DivisionRing}, \text{CommutativeRing} \text{ over } \text{Ring}
\]

More precisely: the **presentations of theories** are related.


Experiment 1: Abstract Theories

The Theories of mathematics are highly related:

\[
\text{Ring} := \text{combine Rng, SemiRing over Semirng} \\
\text{Field} := \text{combine DivisionRing, CommutativeRing over Ring}
\]

More precisely: the presentations of theories are related.

Main Idea: Tiny Theories.

Build up “network” one concept at a time.
Experiment 1: Abstract Theories

The Theories of mathematics are highly related:

\[
\text{Ring} := \text{combine Rng, SemiRing over Semirng} \\
\text{Field} := \text{combine DivisionRing, CommutativeRing over Ring}
\]

More precisely: the presentations of theories are related.

Main Idea: Tiny Theories.

Build up “network” one concept at a time.

Empty := Theory {}
Carrier := Empty extended by \{ U : type \}
PointedCarrier := Carrier extended by \{ e : U \}
UnaryOperation := Carrier extended by \{ prime : U \rightarrow U \}
PointedUnarySystem := combine UnaryOperation, PointedCarrier over Carrier
DoublyPointed := PointedCarrier extended by \{ e2 : U \}
BinaryOperation := Carrier extended by \{ ** : (U,U) \rightarrow U \}
Magma := BinaryOperation [** \rightarrow *]
CarrierS := Carrier[U \rightarrow S]
MultiCarrier := combine Carrier, CarrierS over Empty
UnaryRelation := Carrier extended by \{ R : U ?\}
BinaryRelation := Carrier extended by \{ R : (U,U)? \}
InvolutionaryUnarySystem := UnaryOperation extended by \{ 
  axiom involutive_prime : forall x:domain(prime). prime(prime x) = x \}
Semigroup := Magma extended by \{ 
  axiom associative_* : associative((*)) \}
Experiment 1: Abstract Theories

The Theories of mathematics are highly related:

\[ \text{Ring} := \text{combine Rng, SemiRing over Semirng} \]
\[ \text{Field} := \text{combine DivisionRing, CommutativeRing over Ring} \]

More precisely: the presentations of theories are related.

Main Idea: Tiny Theories.

Build up “network” one concept at a time.

\[ \text{LeftNearRing} := \text{combine LeftNearSemiring, AdditiveGroup over AdditiveMonoid} \]
\[ \text{LeftSemirng} := \text{combine LeftNearSemiring, AdditiveCommutativeMonoid over AdditiveMonoid} \]
\[ \text{LeftRng} := \text{combine LeftNearRing, LeftSemirng over LeftNearSemiring} \]
\[ \text{LeftSemiring} := \text{combine LeftSemirng, Monoid1 over Semigroup} \]
\[ \text{LeftRing} := \text{combine LeftRng, LeftSemiring over LeftSemiring} \]

\[ \text{Semirng} := \text{LeftSemiring extended by} \]
\[ \{ \text{axiom rightDistributive}_{-,*,+} : \text{leftDistributive((flipuc (*))),(+)) } \} \]
\[ \text{Rng} := \text{combine LeftRng, Semirng over LeftSemiring} \]
\[ \text{SemiRing} := \text{combine LeftSemiring, Semirng over LeftSemiring} \]
\[ \text{Dioid} := \text{SemiRing extended by} \{ \text{axiom idempotent}_{+} : \text{idempotent((+)) } \} \]

\[ \text{Ring} := \text{combine Rng, Semirng over Semirng} \]
Experiment 1: Abstract Theories

The Theories of mathematics are highly related:

\[
\text{Ring} := \text{combine Rng, SemiRing over Semirng}
\]

\[
\text{Field} := \text{combine DivisionRing, CommutativeRing over Ring}
\]

More precisely: the presentations of theories are related.

Main Idea: Tiny Theories.

Build up “network” one concept at a time.

\[
\text{LeftNearSemiring} := \text{Theory} \{ \\
\quad \text{U} : \text{type} ; \\
\quad \ast : (\text{U}, \text{U}) \rightarrow \text{U} ; \\
\quad + : (\text{U}, \text{U}) \rightarrow \text{U} ; \\
\quad 0 : \text{U} ; \\
\quad \text{axiom rightIdentity}_+0 := \text{forall } x : \text{U}. \ x + 0 = x ; \\
\quad \text{axiom leftIdentity}_+0 := \text{forall } x : \text{U}. \ 0 + x = x ; \\
\quad \text{axiom leftDistributive}_*+ := \text{forall } x,y,z : \text{U}. \ x \ast (y + z) = (x \ast y) + (x \ast z) ; \\
\quad \text{axiom left0} := \text{forall } x : \text{U}. \ 0 \ast x = 0 ; \\
\quad \text{axiom associative}_+ := \text{forall } x,y,z : \text{U}. \ (x + y) + z = x + (y + z) ; \\
\quad \text{axiom associative}_* := \text{forall } x,y,z : \text{U}. \ (x \ast y) \ast z = x \ast (y \ast z)
\}
\]
Experiment 2: Concrete Theories

- aka Concrete Structures
- Exact same idea as for Abstract Theories
Experiment 2: Concrete Theories

- aka **Concrete Structures**
- Exact same idea as for Abstract Theories
  - but many concrete structures are parametric, and most extensions are conservative.
Experiment 2: Concrete Theories

- aka Concrete Structures
- Exact same idea as for Abstract Theories
  - but many concrete structures are parametric, and most extensions are conservative.

\[
\text{Bit}.\text{Base} := \text{Empty extended conservatively by } \{ \\
\text{Inductive} \ bit \\
| \ 0: \text{bit} \\
| \ 1: \text{bit}
\}
\]

or

\[
\text{Bit}.\text{Base}.\text{Abstract} := \text{Empty extended by } \{ \\
\text{bit} : \text{type} \\
1 : \text{bit} \\
0 : \text{bit} \\
\text{axiom: } \forall b : \text{bit}. \ b = 1 \text{ or } b = 0 \\
\text{axiom: } \neg(1 = 0)
\}
\]
Experiment 2: Concrete Theories

- aka Concrete Structures
- Exact same idea as for Abstract Theories
  - but many concrete structures are parametric, and most extensions are conservative.

```
Bit_And := Bit_Base extended conservatively by {
    bit_and : (bit, bit) -> bit;
    bit_and(x,y) = case x of {
        | 0 -> 0
        | 1 -> y
    }
}
Bit := combine Bit_And, Bit_Or, Bit_Not, Bit_Implies, Bit_Xor, Bit_Xnor over Bit_Base
```
Experiment 2: Concrete Theories

- aka Concrete Structures
- Exact same idea as for Abstract Theories
  - but many concrete structures are parametric, and most extensions are conservative.

Nat := Empty extended conservatively by {
  Inductive nat
  zero : nat
  succ : nat → nat;
}
List := Carrier extended conservatively by {
  Inductive list
  nil : list
  cons : U → list → list;
}
BitCarrier := instance Bit_Base of Carrier via [ bit |→ U ]
BitList := combine List, BitCarrier over Carrier

and of course length, map, zipWith etc.
Experiment 3: Applied Universal Algebra

- **Universal Algebra**: study of algebraic structures themselves rather than their models.

- **Uniform Constructions**: 
Experiment 3: Applied Universal Algebra

- **Universal Algebra**: study of algebraic structures themselves rather than their models.
- **Uniform Constructions**:
  - homomorphism, term algebra, substructure, type representing a model, etc.
Experiment 3: Applied Universal Algebra

- Universal Algebra: study of algebraic structures themselves rather than their models.
- Uniform Constructions:
  - homomorphism, term algebra, substructure, type representing a model, etc.

\[
\text{MonoidTerm} := \text{Theory} \{ \text{type } \text{MTerm} = \&\text{Monoid} \}
\]

denotes the inductive term

\[
\text{MonoidTerm} := \text{Theory} \{
\text{type } \text{MTerm} = \text{data } X .
\text{#e : } X |
\text{#* : } (X, X) \rightarrow X
\}
\]
Experiment 3: Applied Universal Algebra

- **Universal Algebra**: study of algebraic structures themselves rather than their models.
- **Uniform Constructions**:
  - homomorphism, term algebra, substructure, type representing a model, etc.

\[
\text{type} \ \text{semigroup} = \text{TypeFrom}(\text{Semigroup})
\]

in the context of a Theory means (i.e. is expanded to)

| type | semigroup = | \{U: type,∗:(U,U)→U, 
|       |             | associative_∗ : ProofOf(∀ x,y,z:U. (x∗y)∗z = x∗(y∗z)) \} |
Experiment 3: Applied Universal Algebra

- **Universal Algebra**: study of algebraic structures themselves rather than their models.
- **Uniform Constructions**:
  - homomorphism, term algebra, substructure, type representing a model, etc.

\[
\text{MonoidH} := \text{Homomorphism}(\text{Monoid})
\]

means (expands to)

\[
\text{MonoidH} := \text{Theory} \{ \\
\quad \text{type} \ \text{monoid} = \text{TypeFrom}(\text{Monoid}); \\
\quad A, B : \text{monoid}; \\
\quad f : A.U \rightarrow B.U; \\
\quad \text{axiom} \ \text{pres}_*: \ \text{forall} \ x, y: A.U. \ f(x A.* y) = f(x) B.* f(y); \\
\quad \text{axiom} \ \text{pres}_e: \ f(A.e) = B.e;
\}
\]
Experiment 3: Applied Universal Algebra

- **Universal Algebra**: study of algebraic structures themselves rather than their models.
- **Uniform Constructions**:
  - homomorphism, term algebra, substructure, type representing a model, etc.

\[
\text{SubSemigroup} := \text{Substructure}(\text{Semigroup})
\]

means (expands to)

\[
\text{SubSemigroup} := \text{Theory} \{ \\
\quad \text{type semigroup} = \text{TypeFrom} (\text{SemiGroup}) ; \\
\quad A : \text{semigroup} ; \\
\quad V : \text{type} ; \\
\quad \text{axiom } V <: A.U ; \\
\quad \text{axiom } \text{closed}_*: \text{forall } x,y:V. \text{defined-in} (x A.* y, V)
\}
\]
Conclusion

- There is a lot of **structure** in Mathematics, and it can be **leveraged** to simplify builder’s lives.

**Process:**

1. Decide on a chunk of mathematics or CS to implement,
2. Implement prototype using proven methods,
3. Once enough done, **find patterns**, and either
   - Use state-of-the-art method to abstract pattern, or
   - Invent something (and make sure it is semantically sound)
4. Repeat