CICM 2018

Biform Theories: Project Description

Jacques Carette, William M. Farmer, and Yasmine Sharoda

Department of Computing and Software
McMaster University

16 August 2018

Outline

- Motivation.
- Notion of a biform theory.
- Project objectives.
- Project status.

Semantics vs. Syntax

- Consider the mathematical expression

  $$(x + 2) * (2 * x + 1) + 3 * x$$

  where x denotes a natural number.

Semantics vs. Syntax

- Consider the mathematical expression

$$(x + 2) * (2 * x + 1) + 3 * x$$

  where x denotes a natural number.
- This expression e has two values:

Semantics vs. Syntax

- Consider the mathematical expression

  $$(x + 2) * (2 * x + 1) + 3 * x$$

  where x denotes a natural number.
- This expression e has two values:
  1. A semantic value that is the natural number denoted by e.

Semantics vs. Syntax

- Consider the mathematical expression

  $$(x + 2) * (2 * x + 1) + 3 * x$$

  where x denotes a natural number.
- This expression e has two values:
  1. A semantic value that is the natural number denoted by e.
  2. A syntactic value that is the expression e itself having the form of a polynomial (which we denote by the quotation $\ulcorner e \urcorner$).

Semantics vs. Syntax

- Consider the mathematical expression

  $$(x + 2) * (2 * x + 1) + 3 * x$$

  where x denotes a natural number.
- This expression e has two values:
  1. A semantic value that is the natural number denoted by e.
  2. A syntactic value that is the expression e itself having the form of a polynomial (which we denote by the quotation $\ulcorner e \urcorner$).
- Some operations apply to semantic values.
  - Examples: $+$ and $*$.
- Other operations apply to syntactic values.
  - Examples: normalize and factor.

Transformers

- Let $\mathcal{E}$ be a set of expressions.
- A transformer is an algorithm that implements a function $\mathcal{E}^n \to \mathcal{E}$.
  - Examples: normalize and factor.

Transformers

- Let $\mathcal{E}$ be a set of expressions.
- A transformer is an algorithm that implements a function $\mathcal{E}^n \to \mathcal{E}$.
  - Examples: normalize and factor.
- Operations on semantic values can often be computed by transformers.

$$
\begin{tikzcd}
\ulcorner 6 \urcorner, \ulcorner 11 \urcorner \arrow[rr, "\text{add}"] & & \ulcorner 17 \urcorner \arrow[d, "\text{evaluation } \llbracket \cdot \rrbracket"] \\
6, 11 \arrow[u, "\text{quotation } \ulcorner \cdot \urcorner"] \arrow[rr, "+"] & & 17 = \llbracket \ulcorner 17 \urcorner \rrbracket
\end{tikzcd}
$$

- Note: The two operators are related by the law of disquotation:

  $\llbracket \ulcorner e \urcorner \rrbracket = e.$

Syntax-Based Mathematical Algorithms

- A syntax-based mathematical algorithm (SBMA) [Far13] is an transformer that manipulates the syntax of mathematical expressions in a mathematically meaningful way.
  - Examples: normalize, factor, add.

Syntax-Based Mathematical Algorithms

- A syntax-based mathematical algorithm (SBMA) [Far13] is an transformer that manipulates the syntax of mathematical expressions in a mathematically meaningful way.
  - Examples: normalize, factor, add.
- SBMAs are commonplace in mathematics!

Syntax-Based Mathematical Algorithms

- A syntax-based mathematical algorithm (SBMA) [Far13] is an transformer that manipulates the syntax of mathematical expressions in a mathematically meaningful way.
  - Examples: normalize, factor, add.
- SBMAs are commonplace in mathematics!
- A SBMA A has two fundamental properties:
  1. The computational behavior of A is the relationship between the input and output expressions of A.
  2. The mathematical meaning of A is the relationship between what the input and output expressions of A mean mathematically.
- A meaning formula for A is a statement that expresses the mathematical meaning of A.

Examples of Meaning Formulas

- The meaning formula for add is:

$$\forall\, x, y : \mathsf{Numeral} \,.\, \mathsf{add}(x, y) = x + y.$$

Examples of Meaning Formulas

- The meaning formula for add is:

$$\forall\, x, y : \mathsf{Numeral}\; .\; [\![\mathsf{add}(x, y)]\!] = [\![x]\!] + [\![y]\!].$$

Examples of Meaning Formulas

- The meaning formula for add is:

  $$\forall\, x, y : \mathsf{Numeral}\ .\ [\![\mathsf{add}(x, y)]\!] = [\![x]\!] + [\![y]\!].$$

  An instance of the meaning formula is:

  $$[\![\mathsf{add}(\ulcorner 6 \urcorner, \ulcorner 11 \urcorner)]\!] = [\![\ulcorner 6 \urcorner]\!] + [\![\ulcorner 11 \urcorner]\!]$$

Examples of Meaning Formulas

- The meaning formula for add is:

  $$\forall\, x, y : \mathrm{Numeral}\ .\ [\![\mathrm{add}(x, y)]\!] = [\![x]\!] + [\![y]\!].$$

  An instance of the meaning formula is:

  $$[\![\mathrm{add}(\ulcorner 6 \urcorner, \ulcorner 11 \urcorner)]\!] = [\![\ulcorner 6 \urcorner]\!] + [\![\ulcorner 11 \urcorner]\!]$$

- The meaning formula for normalize is:

  $$\begin{array}{l} \forall\, p, q : \mathrm{Poly}\ . \\ \quad (\forall\, x : \mathbb{N}\ .\ [\![p]\!] = [\![\mathrm{normalize}(p)]\!]) \wedge \\ \quad (\forall\, x : \mathbb{N}\ .\ [\![p]\!] = [\![q]\!]) \equiv \mathrm{normalize}(p) = \mathrm{normalize}(q) \end{array}$$

Axiomatic Theories vs. Algorithmic Theories

- Let L be a language in some underlying logic.
- An axiomatic theory is a pair $T = (L, \Gamma)$ where Γ is a set of formulas of L that serve as the axioms of T.
  - Axiomatic theories are implemented in proof assistants.

Axiomatic Theories vs. Algorithmic Theories

- Let L be a language in some underlying logic.
- An axiomatic theory is a pair $T = (L, \Gamma)$ where Γ is a set of formulas of L that serve as the axioms of T.
  - Axiomatic theories are implemented in proof assistants.
- An algorithmic theory is a pair (L, Π) where Π is is a set of transformers that implement functions on the expressions of L.
  - Algorithmic theories are implemented in computer algebra systems.

Axiomatic Theories vs. Algorithmic Theories

- Let L be a language in some underlying logic.
- An axiomatic theory is a pair $T = (L, \Gamma)$ where Γ is a set of formulas of L that serve as the axioms of T.
  - Axiomatic theories are implemented in proof assistants.
- An algorithmic theory is a pair (L, Π) where Π is is a set of transformers that implement functions on the expressions of L.
  - Algorithmic theories are implemented in computer algebra systems.
- Problem. Can an axiomatic theory and algorithmic theory be combined so that we can define and reason about SBMAs in the same context?

Axiomatic Theories vs. Algorithmic Theories

- Let L be a language in some underlying logic.
- An axiomatic theory is a pair $T = (L, \Gamma)$ where Γ is a set of formulas of L that serve as the axioms of T.
  - Axiomatic theories are implemented in proof assistants.
- An algorithmic theory is a pair (L, Π) where Π is is a set of transformers that implement functions on the expressions of L.
  - Algorithmic theories are implemented in computer algebra systems.
- Problem. Can an axiomatic theory and algorithmic theory be combined so that we can define and reason about SBMAs in the same context?
- Our solution is the notion of a biform theory.

Biform Theories

- A biform theory is a triple $T = (L, \Pi, \Gamma)$ where:
  1. L is a language of some underlying logic.
  2. Π is a set of transformers that implement functions on the expressions of L.
  3. Γ is a set of formulas of L that serve as the axioms of T.
- For each $\pi \in \Pi$, L includes a name for the function implemented by π that serves as a name for π.
- The axioms of T specify the meaning of the nonlogical symbols of L including the names of the transformers of T.
- The transformers may be written in L or in a programming language external to L.
- T is an axiomatic theory if Π is empty and is an algorithmic theory if Γ is empty.

Formalizing Biform Theories

- To formalize a biform theory in a logic **Log** we need to be able to formalize SBMAs in **Log**.

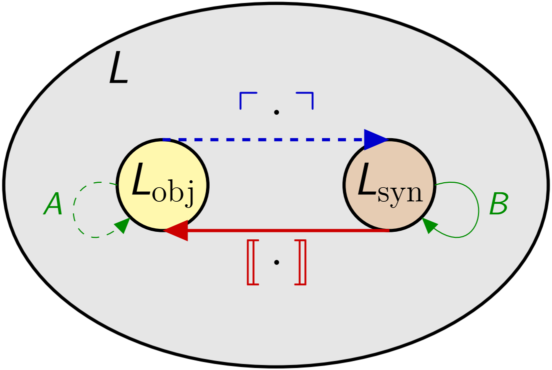
Formalizing Biform Theories

- To formalize a biform theory in a logic **Log** we need to be able to formalize SBMAs in **Log**.
- To formalize an SBMA A in **Log** we must:
  1. Define or specify in **Log** a function B on syntactic values representing A.
  2. State and prove in **Log** the meaning formula for B from the definition or specification of B.
  3. Apply B to mathematical expressions in **Log** by instantiating the meaning formula for B and then applying the result.

Standard Approach: Local Reflection

- Let A be an SBMA on expressions in a language L_{obj} of some logic **Log**.
- We build a metareasoning infrastructure in **Log** consisting of:
  1. An inductive type L_{syn} of syntactic values representing the expressions in L_{obj}.
  2. A quotation operator $\ulcorner \cdot \urcorner$ mapping expressions in L_{obj} to syntactic values of L_{syn}.
  3. An evaluation operator $[\![\cdot]\!]$ mapping syntactic values of L_{syn} to values of L_{obj}.
- We define a function B in **Log** from syntactic values representing inputs of A to syntactic values representing outputs of A.
- The infrastructure is local in the sense that L_{obj} is not the whole language L of Log.

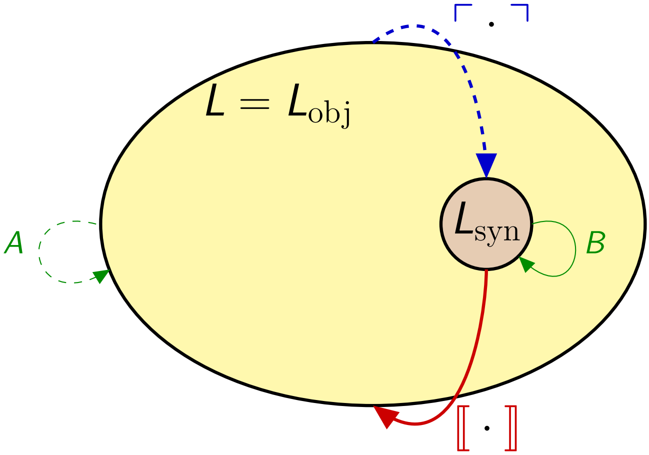
Local Reflection



An Alternate Approach: Global Reflection

- Local reflection does not scale up well:
  - Each collection of SBMAs requires a separate infrastructure.
  - Extending an SBMA to a new domain requires a new infrastructure.
- Global reflection employs a single infrastructure for all SBMAs:
  1. An inductive type representing the entire set of expressions.
  2. A global quotation operator $\ulcorner \cdot \urcorner$.
  3. A global evaluation operator $\llbracket \cdot \rrbracket$.
- Global reflection requires a logic with global quotation and evaluation operators.
- It is an open problem whether global reflection is viable!

Global Reflection



Project Objectives

- Primary objective. Develop a methodology for expressing, manipulating, managing, and generating mathematical knowledge as a graph of biform theories.
- The project is a subproject of MathScheme, a long-term project to produce a framework for integrating formal deduction and symbolic computation.
- Our strategy is to break down the problem into five subprojects.

1. Logic

- Objective. Design a logic **Log** that is a version of simple type theory with an inductive type of syntactic values, a global quotation operator, and a global evaluation operator.

1. Logic

- Objective. Design a logic **Log** that is a version of simple type theory with an inductive type of syntactic values, a global quotation operator, and a global evaluation operator.
- Status. We have developed $\mathrm{CTT}_{\mathrm{qe}}$ [Far18], a version of Church's type theory with global quotation and evaluation operators.
  - $\mathrm{CTT}_{\mathrm{qe}}$ is suitable for defining SBMAs and stating, proving, and instantiating their meaning formulas.
  - We have defined in $\mathrm{CTT}_{\mathrm{qe}}$ a notion of a theory morphism [Far17].

2. Implementation

- Objective. Produce an implementation **Impl** of **Log** and demonstrate that SBMAs can be defined in **Impl** and their meaning formulas can be stated, proved, and instantiated in **Impl**.

2. Implementation

- Objective. Produce an implementation **Impl** of **Log** and demonstrate that SBMAs can be defined in **Impl** and their meaning formulas can be stated, proved, and instantiated in **Impl**.
- Status. We have produced an implementation of $\mathrm{CTT}_{\mathrm{qe}}$, called HOL Light QE [CarFarLas18], by modifying HOL Light.
  - We are working now on testing HOL Light QE by formalizing SBMAs in it.

3. Transformers

- Objective. Enable biform theories to be defined in **Impl** and introduce a mechanism for applying transformers defined outside of **Impl** to expressions of **Log**.

3. Transformers

- Objective. Enable biform theories to be defined in **Impl** and introduce a mechanism for applying transformers defined outside of **Impl** to expressions of **Log**.
- Status. We have not begun this subproject yet.

4. Theory Graphs

- Objective. Enable biform theory graphs to be defined in **Impl**.

4. Theory Graphs

- Objective. Enable biform theory graphs to be defined in **Impl**.
- Status. We have developed a case study of a biform theory graph consisting of eight biform theories encoding natural number arithmetic [CarFar17].
  - We have produced partial formalizations of the case study in $\mathrm{CTT}_{\mathrm{qe}}$ and Agda.
  - We intend to formalize the case study in HOL Light QE.

5. Generic, Specializable Transformers

- Objective. Design and develop in **Impl** a scheme for defining generic transformers in a biform theory T that can be automatically specialized when transported to an instance of T using code generation.

5. Generic, Specializable Transformers

- Objective. Design and develop in **Impl** a scheme for defining generic transformers in a biform theory T that can be automatically specialized when transported to an instance of T using code generation.
- Status. We have a great deal of experience producing generic programs of this form.

References

- [CarFar17] J. Carette and W. Farmer, "Formalizing Mathematical Knowledge as a Biform Theory Graph: A Case Study", in: *Intelligent Computer Mathematics*, LNCS 10383:9–24, 2017.
- [CarFarLas18] J. Carette, W. M. Farmer, and P. Laskowski, "HOL Light QE", *Interactive Theorem Proving*, LNCS 10895:215–234, 2018.
- [Far13] W. M. Farmer, "The Formalization of Syntax-Based Mathematical Algorithms using Quotation and Evaluation", in: *Intelligent Computer Mathematics*, LNCS 7961:35–50, 2013.
- [Far17] W. M. Farmer, "Theory Morphisms in Church's Type Theory with Quotation and Evaluation", *Intelligent Computer Mathematics*, LNCS 10383:147–162, 2017.
- [Far18] W. M. Farmer, "Incorporating Quotation and Evaluation into Church's Type Theory", *Information and Computation*, 260:9–50, 2018.

Conclusion

The Biform Theories project seeks to show that:

1. Global reflection is a viable approach for formalizing SBMAs in biform theories.
2. Biform theories provide an effective mechanism for integrating formal deduction and symbolic computation.
3. A biform theory graph is a structure well suited for formalizing large bodies of mathematical knowledge.

Conclusion

The Biform Theories project seeks to show that:

1. Global reflection is a viable approach for formalizing SBMAs in biform theories.
2. Biform theories provide an effective mechanism for integrating formal deduction and symbolic computation.
3. A biform theory graph is a structure well suited for formalizing large bodies of mathematical knowledge.

Thank You!