Leveraging Information Contained in Theory Presentations

Yasmine Sharoda
sharodym@mcmaster.ca

Supervisors: Jacques Carette and William Farmer

Computing and Software, McMaster University, Canada

Building large libraries of mathematics is an active area of research in Mathematical Knowledge Management (MKM). A library structures information using some unit of knowledge. Whether that unit is called a theory, specification, module, locale or something else, we can abstractly see it as a way for encapsulating knowledge. We want to study the units of encapsulated information in formal systems, abstracting over the language details. Our abstraction consider units in which one can define sorts, typed symbols (optionally providing their definitions) and axioms stating the properties of the defined sorts and symbols. We use the term *theory presentation* to refer to this abstract unit. The theory of this abstraction has been well-studied under the name universal algebra [5]. Our goal is to identify what knowledge can be automatically generated from theory presentations. We want to provide a library of algorithms to generate this knowledge. This way, we provide tool support for library development and formalization tasks by removing some of the boilerplate associated with defining constructs that can be automatically generated. Not only does generating these new pieces save time and effort for the developer, but, more importantly, leverages the information contained in the theory graph by establishing new connections between the theories. A *theory graph* is a collection of theories and theory morphisms. We want to be able to generate the new pieces of knowledge and the morphisms connecting them to the graph. We describe our contributions as

- Evaluating the status of current popular libraries of mathematics, in terms of modularity and reusability.
- Compiling a list of constructions that can be automatically generated from the syntax of a theory presentation.
- Building a library of generic algorithms for computing these constructions, given a theory written in an abstract description language. At this point, we only consider theory presentations in their most abstract format as a collection of sorts, function symbols and axioms written in some formal language.
- Implementing generators that can produce these constructions for a specific theory in a specific language, taking into considerations all similarities and differences between different formal systems.

We now list some of the constructions that can be generated from theory presentations: signatures, sub-algebras, product-algebras, projection, congruence, quotient algebra, record definition, homomorphism, homomorphism- equality, isomorphism, endomorphism, automorphism, composition-of- morphisms, closed term language, open term language, staged-term- language, structural induction principle, evaluation functions, rewrite rules, equivalence of terms, and parse trees.

In: A. Editor, B. Coeditor (eds.): Proceedings of the XYZ Workshop, Location, Country, DD-MMM-YYYY, published at http://ceur-ws.org

Some of these constructions are inspired by universal algebra, others are more related to programming disciplines, like meta-programming.

Approach

We are interested in syntactically manipulating theory presentations, in their most abstract form. To achieve a high level of abstraction, we use MMT [3] as a data description language (DDL). MMT is a foundation independent framework for managing mathematical knowledge. Knowledge is structured as a theory graph. Being foundation independent, it avoids the commitment to any specific logical foundation and allows the translation between the different ones. This makes it appealing to our research work as it enables us to abstract over language details, explore the minimal logic in which the different pieces of knowledge can be generated, and explore whether these pieces can be formalized in other logics.

After generating the different pieces of knowledge, we want to be able to translate them into different formal systems. Considering systems with very different foundations, like Isabelle/HOL and Agda. The first is based on HOL, while the second is based on dependent type theory. They have different representations of theories, yet the information contained in them is the same. For example, a `Monoid` theory consists of a sort, a point, a binary operation and three axioms. These pieces do not change based on the logic, but the way they are formalized does. We are interested in automating the generation of theories in different formal systems. For this, we aim to develop a feature model exploring the similarities and differences between the different systems. This feature model will be used to design generators from our abstract model to the different systems. This way, the information is generated once, and translated automatically to different representations.

Evaluation

We use the MathScheme library [1] to test our implementation of combinators to generate information. The library contains over 1000 axiomatic theories, defined by using the combinators introduced in [2] and structured as a theory graph. To be able to use the library, we translate it into MMT. The combinators used to build the library are interpreted as diagram level combinators [4]. Theories in the library have different foundations. Some theories are based on simple logics, like equational logic, but others are based on more expressive[1] ones, like dependent type theory.

In conclusion, we aim at supporting the development of libraries by generating some related construction that can be made available by syntactic manipulation of theory presentations. As an outcome of our work, a user should be able to write something like

```
BooleanAlgebra := Theory {
  U : type;
  * : (U, U) -> U;
  + : (U, U) -> U;
  --> : (U, U) -> U;
  0 : U;
  1 : U;
  compl : U -> U;
  axiom unipotent_-->_1 : forall x : U. (x --> x) = (1);
  ....
}
generate (termLang, simplify)
```

and then ask the system to evaluate `simplify ((x * y) + 1)` without any extra effort to define the term language or simplification rules.

[1] By expressivity we mean practical expressivity, in terms that information is expressed in a practical and convenient way

References

[1] Carette, J., Farmer, W.M., Jeremic, F., Maccio, V., O'Connor, R., Tran, Q.M.: The mathscheme library: Some preliminary experiments. arXiv preprint arXiv:1106.1862 (2011)

[2] Carette, J., O'Connor, R.: Theory presentation combinators. In: International Conference on Intelligent Computer Mathematics. pp. 202–215. Springer (2012)

[3] Kohlhase, M., Rabe, F., Zholudev, V.: Towards MKM in the large: Modular representation and scalable software architecture. In: International Conference on Intelligent Computer Mathematics. pp. 370–384. Springer (2010)

[4] Rabe, F., Sharoda, Y.: Diagram combinators in mmt. In: International Conference on Intelligent Computer Mathematics. Springer (2019)

[5] Wechler, W.: Universal algebra for computer scientists, volume 25 of eatcs monographs on theoretical computer science (1992)