A Rational Reconstruction of a System for Experimental Mathematics

Jacques Carette[1], William M. Farmer[1], and Volker Sorge[2]

[1]Department of Computing and Software
McMaster University, Hamilton, Ontario, Canada
{carette,wmfarmer}@mcmaster.ca
http://www.cas.mcmaster.ca{˜carette,˜wmfarmer}

[2]School of Computer Science, University of Birmingham, UK,
V.Sorge@cs.bham.ac.uk, http://www.cs.bham.ac.uk/˜vxs

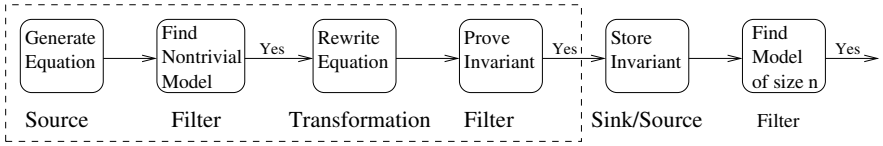
Over the last decade several environments and formalisms for the combination and integration of mathematical software systems have been proposed. Many of these systems aim at a traditional automated theorem proving approach, in which a given conjecture is to be proved or refuted by the cooperation of different reasoning engines. However, they offer little support for experimental mathematics in which new conjectures are constructed by an interleaved process of model computation, model inspection, property conjecture and verification. In particular, despite some previous research in that direction, there are currently no systems available that provide, in an easy to use environment, the flexible combination of diverse reasoning system in a plug-and-play fashion via a high level specification of experiments.

[2, 3] presents an integration of more than a dozen different reasoning systems — first order theorem provers, SAT solvers, SMT solvers, model generators, computer algebra, and machine learning systems — in a general bootstrapping algorithm to generate novel theorems in the specialised algebraic domain of quasigroups and loops. While the integration leads to provably correct results, the integration itself was achieved in an ad-hoc manner, i.e., systems where combined and recombined in an experimental fashion with a set of purpose built bridges that not only perform syntax translations but also semantic filtering.

In recent work we have started a rational reconstruction of the system, in order to expose the general principles behind the combination and communications of the single systems. We use the framework for trustable communication between mathematics systems that was put forth in [1]. It employs the concept of biform theories that enable the combined formalisation of the axiomatic and algorithmic theory behind the generation process. These should ultimately be used in the design of a flexible environment for experimental mathematics that enables a user to specify complex experiments on a high level without the need of detailed knowledge of the underlying logical relations and the particularities of the integrated systems.

In a first stage we concentrate on an interesting sub-process of the bootstrapping algorithm, namely the generation of isotopy invariants for loops. A loop is a simple algebraic structure with a single operation — generally non-associative — and isotopy is an equivalence relation, which is a generalisation of isomor-

phism. The basic idea of our procedure is to find identities (i.e., universally quantified equations) that hold for some loop, by first generating identities and then checking, which identity has a loop of small, but non-trivial size satisfying it, using a model generator. All identities for which a loop exists are then transformed into derived identities, by extending the factors of the original identity systematically using two additional operations defined on the loop. All derived identities for which we can prove, by means of a first order automated theorem prover, that they are invariant under isotopy are universal identities. Note that, for each universal identity, we show that it is an invariant under isotopy independently of the size of a loop. We can therefore reuse these universal identities in different classifications. Consequently, we collect universal identities in a pool of confirmed isotopy invariants, and these are used in other steps of the larger process. We can view the generation of isotopy invariants as a sequence of single computational processes as displayed in the figure below. Each process accomplishes a different function in the overall computational process, e.g., is a source of equations, transforms expressions, or filters with respect to different criteria.



For each module, we have a background (biform) theory that expresses the language and axioms necessary to describe the rules (and thus the inputs and outputs) encapsulated in that module. Using appropriate translations and interpretations, we can set up a communication channel (a connection in the language of [1]). We have worked out detailed formal specifications for each process and their communications. Following [1] we abstract out the details of the idiosyncratic syntax of each of the systems. We use a uniform abstract syntax (in this case, we need 4 separate languages, each embedded in the other) for the specification. This allows us to abstract out the tedious engineering of transformations in and out of the actual systems. On the other hand, any transformation beyond trivial parsing and pretty-printing are explicitly specified. It turns out that it is surprisingly difficult to separate the syntactic, semantic, and algorithmic level of the current implementation.

References

1. J. Carette, W. M. Farmer, and J. Wajs. Trustable communication between mathematical systems. In *Proc. of Calculemus 2003*, pages 58–68, 2003.
2. S. Colton, A. Meier, V. Sorge, and R. McCasland. Automatic generation of classification theorems for finite algebras. In *Proc. of IJCAR 2004*, volume 3097 of *LNAI*, pages 400–414. Springer Verlag, 2004.
3. V. Sorge, A. Meier, R. McCasland, and S. Colton. Automatic construction and verification of isotopy invariants. In *Proc. of IJCAR 2006*, volume 4130 of *LNAI*, pages 36–51. Springer Verlag, 2006.