

Managing Mathematical Workflows

Volker Sorge

School of Computer Science,
University of Birmingham, UK,
V.Sorge@cs.bham.ac.uk

Over the last years there have been several projects on building environments to support the interconnection of mathematical software systems [2, 5] as well as some attempts to exploit these systems to support exploratory mathematics [1]. However, all of these attempts are generally built for proving a single, already given goal. That is, the theorem that is to be shown, is already fully formulated or at least known to a degree such that only preconditions of the theorem's statement have to be adjusted. Our recent experience has shown, that systems geared towards proving single theorems are generally neither powerful nor flexible enough to deal with experimental mathematics. In particular, when assembling a final theorem by interleaving sequences of discovery, computation, and verification steps [3] various techniques have to be experimentally combined until a working sequence is found. The aim of our work is therefore to provide a framework, in which a working mathematician can easily automate a large number of experiments by specifying a mathematical workflow between a multitude of systems. While it is necessary to guarantee automatic interoperability between the systems in the framework, it should also allow the user to easily specify and implement additional transformations on data when it is passed between systems.

We motivate our ideas with two examples of mathematical workflows that originate from our work on constructing and verifying classification theorems in quasigroup theory. Figures 1 and 2 present two UML sequence diagrams for these examples. The first is concerned with the construction of classification theorems in finite algebra using a bootstrapping algorithm (for details see [3]). The main idea of the workflow is as follows: given a set of axioms we construct two algebras Q and Q' with a model generator, then shown them to be non-isomorphic by a theorem prover, construct a discriminant P with HR, and subsequently show that P (1) is a proper discriminant for isomorphism, (2) holds for Q , and (3) does not hold for Q' , using again an ATP. Having these properties established, we can then either show that the original axioms together with P characterise an isomorphism class or construct another algebra that satisfies the axioms together with P , but is non-isomorphic to Q . Both problems are first simplified with a computer algebra system. The former problem is then tackled with a SAT solver and if successfully proved we stop. The latter problem is again passed to a model generator and if a model can be found, we restart the procedure. Analogously we can do the same for Q' and $\neg P$. The workflow eventually leads to a decision tree that represents a fully verified classification theorem.

The approach works well for classification with respect to isomorphism, it does not necessarily work for other equivalence relations. As a second example we take a look at the harder task of classifying loops (quasigroups with a unit element) with respect to isotopy. While for the overall construction of the classification theorem the same workflow as in Figure 1 can be used, the challenging part is to construct suitable isotopy discriminants since HR can no longer help. For this we have to establish another workflow, depicted in Figure 2, which is based on the concept of universal identities that form isotopy invariants [4]. The workflow proceeds as follows: Given a generated equation E we check whether a loop of size ≥ 8 can be generated that satisfies E . If this is the case, E is transformed into a universal identity I . (In the workflow the symbolic computation unit essentially performs the transformations necessary in between the computations of the other two systems.) We then verify, with a first-order ATP, whether I is indeed an isotopy invariant. If it is not it is discarded, otherwise I can be used for discriminating between two concrete loop

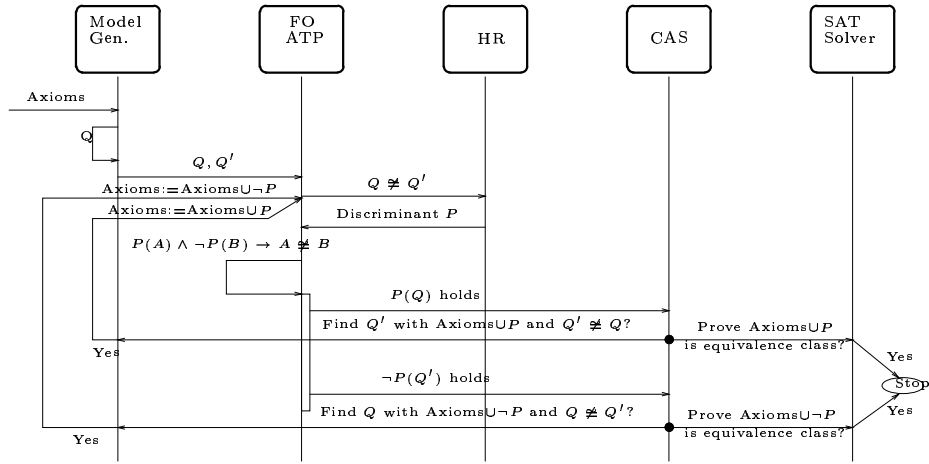


Figure 1: Workflow for constructing isomorphism classification theorems.

structures of order n during the classification. Before doing so, however, we first check whether I is at all useful by testing with a model generator that a loop of order n exists. Naturally, this does not guarantee that I can actually discriminate between the loops under consideration in the overall classification process. Therefore, we first systematically compute tens of thousands of equations and start the workflow with them. Then the model generator and the prover essentially have a filter function. This results in several thousand invariants which can be used during the classification. However, in case these are not enough, we can always restart the workflow in order to generate more.

References

- [1] B. Buchberger, et. al. Theorema: An Integrated System for Computation and Deduction in Natural Style. *Proc. CADE-15 Workshop on Integr. of Deductive Systems*, p. 96–102, 1998.
- [2] J. Calmet and K. Homann. Towards the Mathematical Software Bus. *Theoretical Computer Science*, 187(1–2):221–230, 1997.
- [3] S. Colton, A. Meier, V. Sorge, and R. McCasland. Automatic generation of classification theorems for finite algebras. *Proc. of IJCAR 2004, LNAI 3097*, pages 400–414. Springer, 2004.
- [4] E. Falconer. Isotopy invariants in quasigroups. *Transactions of the American Mathematical Society*, 151(2):511–526, October 1970.
- [5] J. Zimmer and M. Kohlhase. System Description: The MathWeb Software Bus for Distributed Mathematical Reasoning. In *Proc. of CADE-18*, 2002.

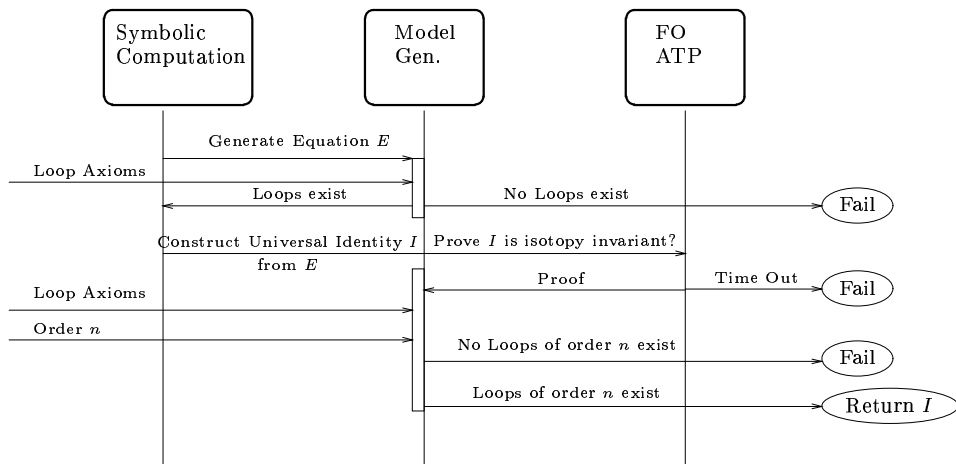


Figure 2: Workflow for constructing isotopy invariants for loops.