SMT-like Queries in Maple

Stephen A. Forrest

Maplesoft Europe Ltd., Cambridge, UK
sforrest@maplesoft.com

Abstract. The recognition that Symbolic Computation tools could benefit from techniques from the world of Satisfiability Checking was a primary motive for the founding of the SC^2 community. These benefits would be further demonstrated by the existence of "SMT-like" queries in legacy computer algebra systems; that is, computations which seek to decide satisfiability or identify a satisfying witness.
The Maple CAS has been under continuous development since the 1980s and its core symbolic routines incorporate many heuristics. We describe ongoing work to compose an inventory of such "SMT-like" queries extracted from the built-in Maple library, most of which were added long before the inclusion in Maple of explicit software links to SAT/SMT tools. Some of these queries are expressible in the SMT-LIB format using an existing logic, and it hoped that those that are not could help inform future development of the SMT-LIB standard.

1 Introduction

1.1 Maple

Maple is a computer algebra system originally developed by members of the Symbolic Computation Group in the Faculty of Mathematics at the University of Waterloo. Since 1988, it has been developed and commercially distributed by Maplesoft (formally Waterloo Maple Inc.), a company based in Waterloo, Ontario, Canada, with ongoing contributions from affiliated research centres. The core Maple language is implemented in a kernel written in C++ and much of the computational library is written in the Maple language, though the system does employ external libraries such as LAPACK and the GNU Multiprecision Library (GMP) for special-purpose computations.

1.2 is and coulditbe

Consistent with Maple's roots as a computer algebra system, its core symbolic solvers (such as `solve`, `dsolve`, `int`) generally aim to provide a general solution to a posed problem which is both compact and useful. Further transformation or simplification of such solutions using simplifiers based on heuristic methods [9] is often necessary.

Nevertheless the approach of posing queries as questions about satisfiability or requests for a satisfying witness is not unknown in Maple. The most obvious

example is in the commands `is` and `coulditbe`. These are the standard general-purpose commands in Maple for querying universal and existential properties, respectively, about a given expression. [11] They are widely used by other symbolic commands in Maple (e.g. `solve`, `int`).

The `is` command accepts an expression `p` and asks if `p` evaluates to the value `true` for every possible assignment of values to the symbols in `p`. The `coulditbe` command operates similarly but asks if there is any assignment of values to the symbols in `p` which could cause `p` to evaluate to `true`.

Both `is` and `coulditbe` return results in ternary logic: `true`, `false`, or `FAIL`. Both also make use of the "assume facility", which is a system for associating Boolean properties with symbolic variables. This provides limits on the range of possible assignments considered by `is` and `coulditbe` and is roughly analogous to a type declaration. For example, the expression `is(x^2>=0)` evaluates to `false` because there are many possible values of `x` which do not evaluate to nonnegative real numbers, in particular the imaginary unit $\sqrt{-1}$. By contrast, the expression `is(x^2>=0) assuming x::real` returns `true` because the range of possible values of `x` has been constrained to real numbers.

An illustrative example is found in the function `product`. In the evaluation of the expression `product(f(n),n=a..b)`, the system seeks to compute a symbolic formula for the product $\prod_{n=a}^{b} f(n)$. As one can verify by inspecting the source code with `showstat(product)`, the implementation of `product` computes a set of roots of $f(n)$ and, if neither a nor b is infinite, checks whether there exists a root r such that r is an integer and $a \leq r \leq b$. If so, it returns zero as the result of the product. (Similar logic is applied if either of a or b is infinite.)

Description	**is**	**coulditbe**
Queries with result `true`	2335	3582
Queries with result `false`	19020	1519
Queries with result `FAIL`	2730	670
Expressible with `QF_LIA`	1759	582
Expressible with `QF_NIA`	6853	1267
Expressible with `QF_LRA`	3059	2394
Expressible with `QF_NRA`	4578	1475
Expressible with `AUFNIRA`	7569	4426
Linear Gaussian integer arithmetic	?	?
Nonlinear Gaussian integer arithmetic	?	?
Linear complex arithmetic	?	?
Nonlinear complex arithmetic	9931	985
Real numbers with trigometric functions	?	?
Real numbers with trigometric functions, exponentials, and logarithms	?	?
Complex numbers with trigometric functions	?	?
Complex numbers with elementary functions	?	?
Total distinct queries	24085	5771

Table 1. Distinct `is` and `coulditbe` queries encountered in a full library test run

As evidence of the ubiquity of such queries, Table 1 summarizes the distinct invocations of `is` and `coulditbe` encountered during a complete run through Maplesoft's internal test suite for the Maple library performed on 24 April 2018. (An investigation into an earlier version of this dataset was published in [10]). This includes both instances in which the test case explicitly calls `is`/`coulditbe` and instances in which `is`/`coulditbe` are invoked by other library functions such as `product`, as shown previously.

(TODO: replace ? in table above with computed values in advance of workshop.)

In total, 24085 distinct `is` and 5771 distinct `coulditbe` queries were issued during the course of the test run. The inputs vary considerably in size and in the complexity of the underlying theory, and for both `is` and `coulditbe` approximately 11% of queries cannot be decided (i.e. return `FAIL` rather than `true` or `false`). A complete list of queries encountered may be viewed at https://doi.org/10.5281/zenodo.943349.

[Summary here on categorization of SMT instances by underlying theory]

2 Future Work

Recent versions of Maple have seen the addition of explicit links to SAT and SMT solvers: Maple 2018 is distributed with both the SAT solver MapleSAT [1] and the SMT solver [13]. In future, we aim to examine the effectiveness of using these packaged solvers on SMT instances which arise during evaluation of symbolic expressions.

An important factor in this assessment will be whether this implementation offers better performance and meaningful answers (not `FAIL`) for a larger class of such queries than existing tools in Maple.

References

1. Hui Liang, Vijay Ganesh. MapleSAT development site. https://sites.google.com/a/gsd.uwaterloo.ca/maplesat/
2. E. Ábrahám, J. Abbott, B. Becker, A.M. Bigatti, M. Brain, B. Buchberger, A. Cimatti, J.H. Davenport M. England, P. Fontaine, S. Forrest, A. Griggio, D. Kroening, W.M. Seiler, and T. Sturm. SC^2: Satisfiability Checking Meets Symbolic Computation. In: M. Kohlhase, M. Johansson, B. Miller, L. de Moura, F. Tompa, eds., Intelligent Computer Mathematics (Proceedings of CICM 2016), pp. 28-43, (Lecture Notes in Computer Science, 9791). Springer International Publishing, 2016. http://www.sc-square.org/Papers/CICM16.pdf.
3. Erika Ábrahám. 2015. Building Bridges between Symbolic Computation and Satisfiability Checking. In *Proceedings of the 2015 ACM on International Symposium on Symbolic and Algebraic Computation* (ISSAC 2015). ACM, New York, NY, USA, 1-6. doi:10.1145/2755996.2756636.
4. Thomas Bouton, Diego Caminha B. de Oliveira, David Déharbe, Pascal Fontaine. (2009). *veriT: An Open, Trustable and Efficient SMT-Solver*. 151-156. doi:10.1007/978-3-642-02959-2_12.

5. Florian Corzilius, Ulrich Loup, Sebastian Junges, and Erika Ábrahám. 2012. *SMT-RAT: an SMT-compliant nonlinear real arithmetic toolbox.* In Proceedings of the 15th international conference on Theory and Applications of Satisfiability Testing (SAT'12), Alessandro Cimatti and Roberto Sebastiani (Eds.). Springer-Verlag, Berlin, Heidelberg, 442-448. doi:10.1007/978-3-642-31612-8_35.
6. Konstantin Korovin, Marek Kosta, Thomas Sturm. *Towards Conflict-Driven Learning for Virtual Substitution.* Vladimir P. Gerdt, Wolfram Koepf, Werner M. Seiler, Evgenii V. Vorozhtsov. Computer Algebra in Scientific Computing - 16th International Workshop, CASC 2014, 2014, Warsaw, Poland. Springer, 8660, pp.256-270, 2014, Lecture Notes in Computer Science. doi:10.1007/978-3-319-10515-4_19.
7. *Maple Programming Guide*, Toronto: Maplesoft, a division of Waterloo Maple Inc., 2005-2016.
8. Clark Barrett, Pascal Fontaine, and Cesare Tinelli. *The Satisfiability Modulo Theories Library (SMT-LIB)*, http://www.smt-lib.org, 2016.
9. Jacques Carette. 2004. Understanding Expression Simplification. In *Proceedings of the 2004 International Symposium on Symbolic and Algebraic Computation, Santander, Spain* (ISSAC 2004), ACM, New York, NY, USA, 72-79. doi:10.1145/1005285.1005298.
10. Stephen A. Forrest. 2017. Integration of SMT-LIB Support into Maple. Second Annual SC^2 Workshop, ISSAC 2017, Kaiserslautern, Germany. http://www.sc-square.org/CSA/workshop2-papers/EA5-FinalVersion.pdf
11. *The Assume Facility in Maple*, Maple Online Help : The Assume Facility.
12. *Logics in SMT-LIB*, http://smtlib.org/logics.shtml.
13. de Moura, L. M., and Bjørner, N. *Z3: an efficient SMT solver.* In TACAS (2008), vol. 4963 of Lecture Notes in Computer Science, Springer, pp. 337–340. https://github.com/Z3Prover/z3.