Integration of SMT-LIB Support into Maple

Stephen Forrest

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

Abstract. The SC^2 project [1] arose out of the recognition that the Computer Algebra and Satisfiability Checking communities mutually benefit from the sharing of results and techniques. An SMT solver can profit from the inclusion of computer algebra techniques in their theory solvers, while a computer algebra system can profit from dispatching SAT or SMT queries which arise as sub-problems during computation to a dedicated external solver; many existing implementations for both of these may be found. Here we describe on-going work in the second category: specifically, we describe an API in Maple for dispatching computations to, and processing results from, an SMT solver supporting the SMT-LIB format.

1 Introduction

The potential benefits of SMT solvers making effective use of computer algebra techniques in their theory solvers are understood. [2] This is the approach realized in the implementation of several SMT solvers, such as veriT [3] and SMT-RAT [4].

The opposite task of incorporating SAT- or SMT-solving techniques into a computer algebra system also has significant prior art. One such example from the computer algebra system Redlog is the integration of learning strategies from CDCL-based SMT solving into real quantifier elimination [5].

Maple[6] 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 by 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 it employs external libraries for special-purpose computations such as LAPACK and GMP.

The SMTLIB 2.0 [7] standard defines a language for writing terms and formulas in a sorted version of first-order logic, specifying background theories and logics, and interacting with SMT solvers in order to impose and retract assertions and inquire about their satisfiability.

While Maple as a computer system generally focuses on obtaining a compact general solution to a posed problem rather than a statement of satisfiability or a

satisfying witness, its core routines do make regular use of satisfiability queries in the course of symbolic simplification.

An illustrative example can be found with the Maple command `product`. In the evaluation of the expression `product(f(n),n=a..b)`, the system seeks to compute a symbolic formula for the product $\prod_{n=a}^{b} f(n)$. Among other steps, it 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.)

This is an example of what is essentially an SMT instance appearing as a sub-problem in the course of symbolic computation. Many examples of such queries may be found in the Maple library, frequently as universally-quantified statements: e.g. apply a certain transformation, provided a specified condition holds universally. It is hoped that dispatching such satisfiability queries to an external SMT solver could offer performance improvements and permit a broader class of such queries to be decided, compared with analogous existing tools in Maple.

2 Challenges

The Maple language is loosely-typed and permits identifiers which have not been previously defined to be freely used in algebraic expressions, with the understanding that such identifiers represent symbolic indeterminates. Maple therefore places no requirements on the user to provide an advance declaration of the mathematical domain associated with or theory underlying the input expression. Maple does possess a facility with which additional properties about symbols may be specified using the commands `assume` or `assuming`. In general however the effective interpretation of symbols is imposed by the particular command being invoked: for example, `coeffs(x^2+3,x)` interprets `x` as a transcendental element while `evalc((x+I)^2)` interprets `x` as a real number.

This overall flexibility presents a significant obstacle to translating an arbitrary algebraic expression from Maple to SMT-LIB: we must either oblige a user to specify the SMT-LIB logic underlying the expression and the type of each symbol explicitly, or attempt to detect them.

3 Results

We present a work-in-progress Maple package, `SMTLIB`, designed to facilitate interaction with an SMT solver supporting the SMT-LIB standard. This package offers three commands: `ToString`, `Satisfiable`, and `Satisfy`.

The first of these, `ToString`, accepts a Maple expression and returns a string output containing an SMT-LIB 2.0 script. By default, this simply asserts the truth of the expression corresponding to the Maple input and requests a satisfiability check (i.e. `(check-sat)`). It does not explicitly invoke an SMT solver, but

merely returns the input which would be sent to one if `Satisfiable` or `Satisfy` were invoked.

The SMT-LIB logic may be explicitly specified or inferred. In the following example, we ask about the satifiability of $x^2+1=0$ while instructing `ToString` to use the SMT-LIB logic `QF_LRA` (quantifier-free linear real arithmetic), implicitly forcing the variable x to be real. (Note that the input line is preceded by `>` and output lines follow afterwards.)

```
> SMTLIB:-ToString( x^2+1=0, logic="QF_LRA" );
"(set-logic QF_LRA)
 (declare-fun x () Real)
 (assert (= (+ (* x x) 1) 0))
 (check-sat)
 (exit)"
```

In the event that the logic is not specified, `ToString` will attempt to choose the "smallest" SMT-LIB logic in which the input can be represented, according to the partial order on SMT-LIB logics described in [8]. That is it will choose a logic which is sufficient to represent the input expression, and which will be a sub-logic of any logic in which the input can be represented.

If we repeat the previous command while leaving the logic unspecified, `ToString` defaults to using the logic `QF_NIA` (quantifier-free nonlinear integer arithmetic) because that is the minimal logic in which both the integer addition and the square term can be represented.

```
> SMTLIB:-ToString( x^2+1=0 );
"(set-logic QF_NIA)
 (declare-fun x () Int)
 (assert (= (+ (* x x) 1) 0))
 (check-sat)
 (exit)"
```

The `Satisfiable` and `Satisfy` commands simply generate SMT-LIB scripts (which request a satisfiability check and a satisfying witness, respectively) and dispatch the query to an SMT solver. By specifying the path to the executable for the SMT solver, an arbitrary SMT-LIB compliant solver may be used, though the default implementation uses Z3 [9]. The output of the SMT solver is parsed and returned as a corresponding Maple object: a Boolean result (for `Satisfiable`) or either a satisfying assignment or the value `NULL`.

The following examples first confirm that a satisfying assignment exists and then returns a satisfying assignment for the equation $w^3+x^3=y^3+z^3$ in positive integers where $w \neq y, w \neq z$:

```
> SMTLIB:-Satisfiable( {w^3+x^3=y^3+z^3, w>0,x>0,y>0,z>0,w<>y,w<>z},
  logic="QF_NIA");
                              true
```

```
> SMTLIB:-Satisfy( {w^3+x^3=y^3+z^3, w>0,x>0,y>0,z>0,w<>y,w<>z},
  logic="QF_NIA");
                {w = 10, x = 9, y = 12, z = 1}
```

4 Conclusion

The SMTLIB Maple package presents a concrete example of a computer algebra system effectively harnessing the power of an SMT solver. The fact that its interface is sufficiently generic to be uncoupled from any particular SMT solver stands as testimony to the benefit of the widespread adoption of the SMT-LIB standard by implementors of SMT solvers.

This implementation nevertheless currently leaves a considerable portion of the functionality defined in the SMT-LIB 2 standard unexploited, including definition of new theories and use of the command language for interacting with assertions via stack push/pop operations.

5 Future Work

The inclusion of the SMTLIB package in Maple provides a facility for users explicitly interested in interacting with an SMT solver. In future, we aim to examine the utility of using as a general-purpose tool for solving SMT instances which arise during evaluation of symbolic expressions.

An important factor in this assessment will be whether this implementation offers better performance and returns definite solutions (i.e. not FAIL) for a larger class of such queries than existing tools in Maple.

References

1. 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. *SC2: 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.
2. 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 '15). ACM, New York, NY, USA, 1-6. DOI: https://doi.org/10.1145/2755996.2756636
3. Thomas Bouton, Diego Caminha B. de Oliveira, David Déharbe, Pascal Fontaine. (2009). *veriT: An Open, Trustable and Efficient SMT-Solver*. 151-156. https://doi.org/10.1007/978-3-642-02959-2_12.
4. 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. http://dx.doi.org/10.1007/978-3-642-31612-8_35.

5. Konstantin Korovin, Marek Kosta, Thomas Sturm. *Towards Conflict-Driven Learning for Virtual Substitution.* Vladimir P. Gerdt and Wolfram Koepf and Werner M. Seiler and 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. ¡http://dx.doi.org/10.1007/978-3-319-10515-4_19¿.
6. *Maple Programming Guide*, Toronto: Maplesoft, a division of Waterloo Maple Inc., 2005-2016.
7. Clark Barrett, Pascal Fontaine, and Cesare Tinelli. *The Satisfiability Modulo Theories Library (SMT-LIB)*, http://www.smt-lib.org, 2016.
8. *Logics in SMT-LIB*, http://smtlib.org/logics.shtml.
9. 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. 337340. Z3 is available at https://github.com/Z3Prover/z3.