# Integration of SMT-LIB Support into Maple

Stephen A. Forrest

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

**Abstract.** The SC$^2$ project arose out of the recognition that the Symbolic Computation 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 its theory solver, 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 of both of these may be found. Here we describe on-going work in the second category: an API in Maple for dispatching computations to, and processing results from, an SMT solver supporting the SMT-LIB 2 format.

## 1 Introduction

### 1.1 SC$^2$, Maple, and SMT-LIB

The SC$^2$ project [1] was established with a mandate to promote the establishment of bridges between the Symbolic Computation and Satisfiability Checking communities in the form of common platforms and roadmaps. Applying this principle to the software tools themselves, it is well understood that SMT solvers can benefit substantially from incorporating computer algebra techniques such as symbolic simplification or quantifier elimination in their theory solvers. [2] This approach is 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].

We describe here a interface between Maple[6] and an arbitrary SMT solver implementing the SMT-LIB 2.0 [7] format. 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.

The SMT-LIB 2.0 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.

### 1.2 SMT-like Queries in Maple

Consistent with Maple's roots as a computer algebra system, its solvers (such as `solve`, `dsolve`, `int`) generally seek to provide the user with 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 [8] is often necessary.

Nevertheless the approach of posing queries as questions about satisfiability or requests for a satisfying witness is not unknown in Maple, and its core routines make regular use of satisfiability queries in the course of symbolic simplification.

The existing general-purpose commands in Maple for querying universal and existential properties about a given expression are named `is` and `coulditbe` respectively. [9] 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 accepts identical syntax 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: it may be a symbol with no numeric value, an expression of arbitrary size containing such a symbol, or simply 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` is limited to those corresponding to real numbers.

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 $\prod_{n=a}^{b} f(n)$. 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.)

This is an example of what is essentially an SMT instance appearing as a subproblem in the course of symbolic computation. Many examples of such queries may be found in the Maple library; a common pattern is to pose an `is` query (that is, verify that a specified condition holds universally) as a precondition to applying a certain transformation.

| Description | is | coulditbe |
|---|---|---|
| Queries with result `true` | 2335 | 3582 |
| Queries with result `false` | 19020 | 1519 |
| Queries with result `FAIL` | 2730 | 670 |
| Queries including addition | 9849 | 2022 |
| Queries including multiplication | 17097 | 3106 |
| Queries including equations ($=$), inequations ($\neq$), and inequalities ($<$, $\leq$) | 11333 | 4439 |
| Queries including integer exponents $\geq 2$ | 7965 | 875 |
| Queries including complex arithmetic | 6215 | 389 |
| Queries with Boolean structure ($\neg$, $\wedge$, $\vee$) | 564 | 168 |
| 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 26 July 2017. 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.

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.

It is probable that a significant subset of these queries are expressible in the SMT-LIB 2 format. Explicitly dispatching such queries from Maple to an external SMT solver could offer performance improvements and permit a broader class of queries to be decided (i.e. return answers other than `FAIL`) than is possible with analogous existing tools in Maple. More generally, the scope of these queries serves to provide evidence that posing existential questions about the satisfiability of problems over the integers and real numbers in the style of SMT is, in fact, a natural activity when doing computer algebra.

## 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 [10]. 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 [11]. 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 meaningful answers (not `FAIL`) for a larger class of such queries than existing tools in Maple.

We also hope to find ways of meaningfully incorporating other parts of the SMT-LIB 2 specification into the Maple interface. At present, in the case of unsatisfiability users of the interface must accept a mere `false` or `NULL`; they would benefit from access to concrete evidence of unsatisfiability (e.g. an unsatisfiable core).

The interface would also benefit from the addition of support for other existing SMT-LIB types, including

- Uninterpreted functions (`QF_UF` and logics which extend it)
- Arrays
- Bit vectors
- Floating-point arithmetic

## 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. 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.
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 2015). ACM, New York, NY, USA, 1-6. doi: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. doi: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. doi: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, 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.
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. 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.
9. *The Assume Facility in Maple*, Maple Online Help : The Assume Facility.
10. *Logics in SMT-LIB*, http://smtlib.org/logics.shtml.
11. 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. https://github.com/Z3Prover/z3.