Bull. Nov. Comp. Center, , 43 (2019), 15–19
© 2019 NCC Publisher

Notes on implementing the ksmt-solver*

F. Braße, K. Korovin, M. Korovina and N. Müller

Abstract. In this paper we report on new black-box and white-box approaches implemented in `ksmt`–solver for checking satisfiability of non-linear constraints over the reals. These approaches are applicable to a large number of constraints involving computable non-linear functions, piecewise polynomial splines, transcendental functions and beyond. A prototypical implementation has been evaluated on several non-linear SMT-LIB examples and the results have been compared with state-of-the-art SMT solvers.

Keywords: `ksmt`–solver, non-linear constraint, CDCL-style reasoning, bound propagation, conflict resolution, implementation.

1. Introduction

Non-linear constraint solving naturally arises in the development of formal methods for verification of safety critical systems, program analysis and information management. Implementations of formal methods are widely used to approve in advance that designed systems satisfy all specification requirements, such as reliability, safety and reachability. Historically, there have been two main approaches to deal with non-linear constrains: the symbolic one originated by Tarski's decision procedure for the real closed fields [9] and the numerical one based on interval constraint propagations [1]. It is well known that both approaches have their strength and weakness concerning completeness, efficiency and expressiveness. Nowdays, merging strengths of symbolical and numerical approaches is one of the challenging research aria in theoretical and applied computer science. In our recent theoretical framework [2] we integrated symbolic [5] and numerical techniques [8] to improve efficiency and to reduce the wrapping effect. This approach has been motivated by extensions of CDCL-style reasoning into domains beyond propositional logic such as linear [6, 7, 5, 4] and polynomial constraints [3]. This theoretical background has been realised in the `ksmt`-package by implementing black-box and white-box approaches. Both of them have been developed in the conflict driven clause learning style and are applicable to a wide class of non-linear constraints.

*The research leading to these results has received funding from the DFG grant WERA MU 1801/5-1, the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 731143 and RFBR-JSPS project number 20-51-50001.

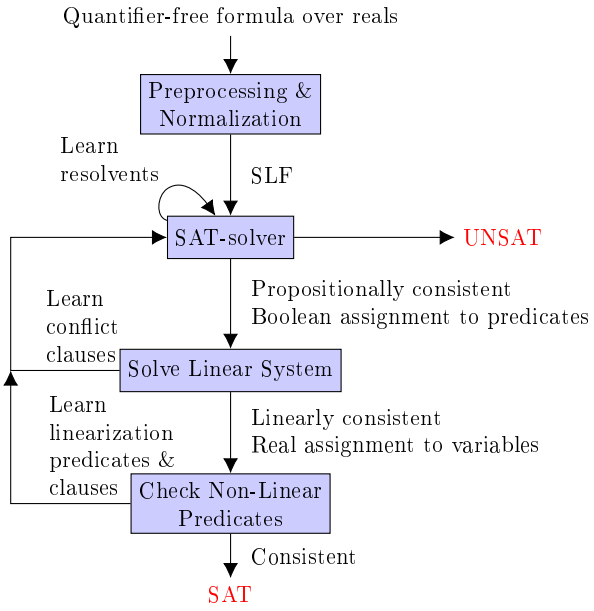

Figure 1. Black-box SAT-based implementation.

The `ksmt` solver is an open source (LGPL-3) publicly available at `http://informatik.uni-trier.de/~brausse/ksmt`. It supports a large subset of `QF_LRA` and `QF_NRA` logics as defined in the SMT-LIB standard[1] and contains a DPLL-based SAT solver, an arithmetical theory solver and tangent space constructors integrated in black-box and white-box `ksmt` solvers. These components also can be used as independent systems.

2. Black-box SAT-based implementation

In the `ksmt`-package, the black-box approach integrates a DPLL-based SAT solver and an arithmetical theory solver that handles systems of inequalities over the reals.

Figure 1 schematically shows this black-box approach. After standard normalisation and preprocessing of an input formula each atomic formula (being an inequality over the reals) is abstracted by a Boolean variable resulting in a propositional formula. The propositional formula is processed by the SAT-solver to produce a Boolean assignment to the atomic formulas that is consistent on the propositional level.

This assignment now also defines a system of inequalities over the reals which is forwarded to the arithmetical theory solver. This solver, in turn, proceeds to check the satisfiability of the received system. If the arithmetical

[1] `http://smtlib.cs.uiowa.edu/`

theory solver detects satisfiability, it provides a solution, i.e., an assignment for the real variables satisfying the system. Otherwise, if the real solver detects unsatisfiability, it generates a propositional unsat core formula describing a sufficient reason for the inconsistency, possibly even with the use of new derived inequalities (which translate to new propositional variables). The negation of this unsat core is added to the SAT solver's state so that propositional models containing the same cause of inconsistency will not be considered again by the SAT solver.

The main steps of the arithmetical theory solver are Conflict Resolutions and Linearisations. They are similar to the ones in the white-box approach and can be found in Section 3.

The specific features are the following: preprocessing input formulas into linear separated form, usage of grids to reduce the necessity for new Boolean variables, conflict driven clause learning loop, solving systems by combination of symbolic and numerical computations in conflict driven style, forgetting linear resolvents similar to forgetting clauses in SAT solvers, linear resolution on predicates.

3. White-box calculus based implementation

In the `ksmt`-package, the white-box approach is realised by implementing the `ksmt` calculus loop proposed in [2]. Figure 2 schematically shows this approach.

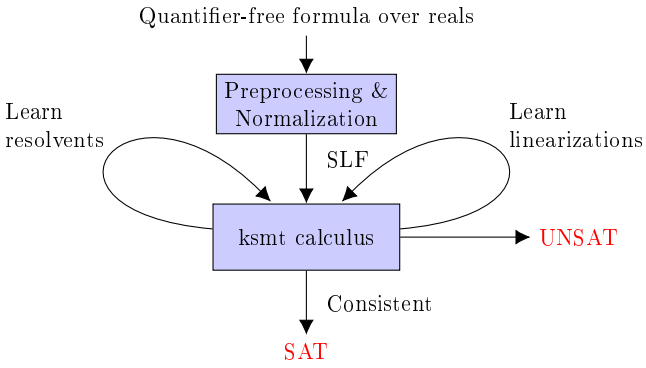

Figure 2. ksmt calculus loop.

The `ksmt` algorithm in our implementation works as follows. Given a set of non-linear constraints, we first separate the set into linear and non-linear parts ("SLF" in Figure 3). Then we incrementally extend a candidate solution into a solution of the whole constraint set (rule **A**), and when such an extension fails, we resolve the conflict by generating a lemma excluding a region which includes the falsifying assignment and apply backjumping (rule

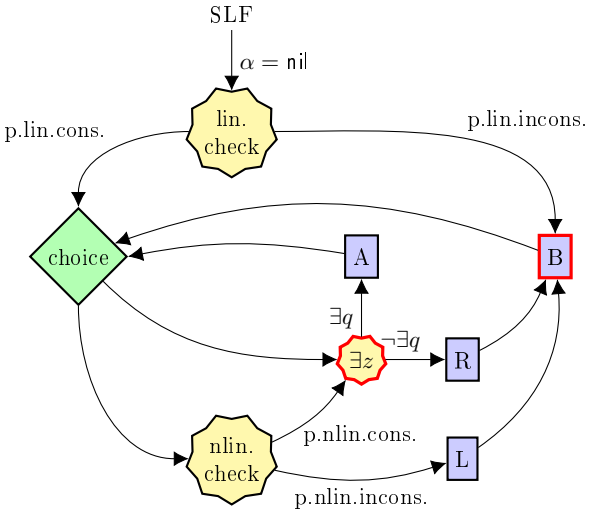

Figure 3. Core of ksmt calculus. Loop terminates in red notes.

B). There are two types of conflicts: between linear constraints, which are resolved in a similar way as in [6] (rule **R**), and non-linear conflicts involving non-linear constraints, these are resolved by local linearisations (rule **L**).

One of the important properties of our algorithm is that all generated lemmas are linear,9 and hence the non-linear part of the problem remains unchanged during the search. In other words, our algorithm can be seen as applying gradual linear approximations of non-linear constraints by local linearisations guided by solution search in CDCL-style.

The specific features are the following: preprocessing input formulas into linear separated form, independency from a SAT solver, conflict driven clause learning loop, solving systems by combination of symbolic and numerical computations in conflict driven style, combinations of model-guided solution search with targeted linearisations for resolving non-linear conflicts, forgetting linear resolvents similar to forgetting clauses in SAT solvers, conflict detection on clauses, linear resolution on clauses, nondeterministic choices between linear conflict resolutions and linearisations.

4. Conclusions and future work

We presented the black-box and white-box loops implemented in the `ksmt`-package for checking satisfiability of non-linear constraints. We already have tangent space constructors for computable analytical functions including exponential and trigonometric functions as well as piecewise polynomial splines and beyond. The next steps will be enlarging of the tangent space constructor class in order to deal with richer classes of continuous constraints, extend-

ing the applicability of our implementation and a more extensive evaluation.

References

[1] Benhamou F. and Granvilliers L. Continuous and interval constraints// In Handbook of Constraint Programming.– Elsevier, 2006.– P. 571—603.

[2] Braußе K., Korovin K., Korovina M. and Müller N.Th. A CDCL-style calculus for solving non-linear constraints// In Proc. FroCoS 2019.–Lect.Notes in Comput. Sci. – 2019.–Vol. 11715.–P.131—148.

[3] Jovanovic D. and de Moura L. Solving non-linear arithmetic. In Proc. IJCAR'2012.–Lect.Notes in Comput. Sci.–2012.– Vol. 7364.– P. 339—354, 2012.

[4] Korovin K., Kosta M. and Sturm Th. Towards conflict-driven learning for virtual substitution// In Proc. CASC 2014.–Lect.Notes in Comput. Sci.–2014.– Vol. 8660.– P. 256—270.

[5] Korovin K., Tsiskaridze N. and Voronkov A. Implementing conflict resolution. In Proc. PSI'2011..–Lect.Notes in Comput. Sci.–2012.– Vol. 7162.– P. 362—376.

[6] Korovin K., Tsiskaridze N. and Voronkov A. Conflict resolution// In Proc. CP'09.–Lect.Notes in Comput. Sci.–2009.– Vol. 5732.– P. 509—523.

[7] McMillan K. L., Kuehlmann A. and Sagiv M. Generalizing DPLL to richer logics// In Proc. CAV'09.–Lect.Notes in Comput. Sci.–2009.– Vol. 5643.– P. 462—476.

[8] Müller N.Th. The iRRAM: Exact arithmetic in C++// In Proc. CCA'2000.– Lect.Notes in Comput. Sci.–2001.– Vol. 2064.– P 222—252.

[9] Tarski A. A decision method for elementary algebra and geometry// 2nd ed. Univ. Cal.– 1951.