Tutorial: SAT and SMT their algorithm designs and applications

Mizuhito Ogawa
School of Information Science,
Japan Advanced Institute of Science and Technology
mizuhito@jaist.ac.jp

Khanh, To Van
University of Engineering and Technology,
Vietnam National University, Hanoi
khanhtv@vnu.edu.vn

I. SAT

SAT and SMT (SAT modulo theory) aim to find a satisfiable instance of given constraints. SAT computes a boolean instance of constraints in a conjunctive normal form (CNF) of propositional logic, and SMT accepts constraints described in background theory, such as arithmetic.

This tutorial consists of Part 1: SAT and Part 2: SMT. Part 1 focuses on SAT solver, and we will overview de-facto-standard algorithm designs, such as non-chronological back tracking with implication graphs, conflict driven learning, and two watched literals [1]. Then, we investigate how to encode problems into CNF. Examples are taken from puzzles. Although puzzles are problems on bounded domains, there is certain hierarchy of difficulties, corresponding to the logical hierarchy of problems. Our examples are SUDOKU [2], Logic pictures [3], and Slitherlink [4][1,2], which correspond to descriptions in CNF, general propositional logic, and higher order logic, respectively. As conversion techniques to efficient CNFs, a popular Tseitin conversion and two special techniques (for the latter two, respectively) are introduced.

II. SMT

When a constraint described in a background theory is given, SMT separates case analysis and satisfiability checking in the background theory. Linear arithmetic (Presburger arithmetic) is one of most popular background theory for SMT, and mostly it is implemented with the simplex method [1]. We first briefly see it, and then we focus on SMT for non-linear arithmetic.

QF_NIA, non-linear arithmetic on integers, is known as Hilbert's 10th problem and undecidable. Practical solutions bound the range for search and apply either of the following.

- **bit-blasting**. Most of fast implementations of SMTs in QF_NIA category uses it. UCLID [5] further boost it by applying abstractions.
- **linearization**. Barcelogic [6] instantiates one of arguments in multiplication by all possible integers in a given bound. Then, non-linear arithmetic is reduced to Presburger arithmetic, which is solved by backend SMTs, e.g., Yices.

[1] http://www.nikoli.co.jp/en
[2] http://bach.istc.kobe-u.ac.jp/sugar/puzzles/

Our extreme focus of this tutorial is QF_NRA category, after general introduction on SAT and SMT. QF_NRA, non-linear constraints on real numbers, is known to be decidable. It was firstly shown by Tarski in 1930's [7] and later an efficient (but still DEXPTIME) QE-CAD (quantifier elimination by cylindrical algebraic decomposition) was proposed [8]. In symbolic computation community, QE-CAD has been implemented as Mathematica, Reduce/Redlog, QEPCAD-B, and Maple/SyNRAC. Recently, SMT activity starts to merge these techniques. For instance, RAHD applies different versions of QE-CAD implementations (QEPCAD-B, Reduce/Redlog) as a backend, and Z3 4.3 (equivalently, nlsat in [9]) includes its own QE-CAD implementation. Earlier versions of Z3 (e.g., Z3 3.1) and SMT-RAT applied Virtual Substitution, which is a special case of QE-CAD for small degrees.

Apart from QE-CAD, recent SMTs in QF_NRA category also apply approximations. For instance, interval constraint propagation is an over-approximation, and Bit-blasting, Linearization, testing are regarded as under-approximations.

- **Interval Constraint Propagation (ICP)**. RSOLVER [10] and iSAT [11] apply input range decomposition and classical interval arithmetic. iSAT refines its search by binary interval decomposition. raSAT [12] applied Affine intervals and combines testing to boost satisfiability checking. It uses more sophisticated interval decomposition guided by Affine interval computation and testing results.
- **Bit-blasting**. MiniSMT [13] describes rational numbers as pairs of integers and restricts possible irrational numbers appearing in instances. For instance, $\sqrt{2}$ is introduced as $\alpha^2 - 2 = 0$ with $\alpha \in [1.3, 1.4]$ before solving satisfiability. Then, it bounds the range of search for bit-blasting.
- **Linearization**. CORD [14] uses CORDIC (Coordinate Rotation Digital Computer), which reduces non-linear constraints to linear constraints under given precision.
- **δ-complete procedure**. dReal [15] is based on the delta complete procedure, which decides SAT and weak-UNSAT of inequalities. raSAT shares a similar idea.

Finally, applications of QF_NRA are briefly mentioned, e.g.,

- Roundoff error analysis [16], [17],
- Linear invariant generation [18] by Farkas's lemma, and
- Polynomial and matrix interpretation in automatic termination detection [19]

References

[1] Kroening, D., Strichman, O.: Decision Procedures: An Algorithmic Point of View. Springer (2008)

[2] Lynce, I., Ouaknine: Sudoku as a SAT problem (2006) International Symposium on Artificial Intelligence and Mathematics (ISAIM 2006).

[3] Ito, H., Sakai, M., Kusakari, K., Nishida, N., Sakabe, T.: Logic picture puzzle generation based on SAT solver the 5th SIG on combinatorial games and puzzles, Tokyo, March, 2010 (in Japanese).

[4] Tamura, N.: Solving puzzles by sugar constraint solver the 1st SIG-CSPSAT project, Kobe, 21st August 2008 (in Japanese).

[5] Bryant, R.E., Kroening, D., Ouaknine, J., Seshia, S.A., Strichman, O., Brady, B.: Deciding bit-vector arithmetic with abstraction. In: Proceedings of the 13th international conference on Tools and algorithms for the construction and analysis of systems. TACAS'07, Springer-Verlag (2007) 358–372

[6] Borralleras, C., Lucas, S., Navarro-Marset, R., Rodríguez-Carbonell, E., Rubio, A.: Solving non-linear polynomial arithmetic via sat modulo linear arithmetic. In: Proceedings of the 22nd International Conference on Automated Deduction. CADE-22, Springer-Verlag (2009) 294–305

[7] Tarski, A.: A decision method for elementary algebra and geometry. Bulletin of the American Mathematical Society **59** (1951)

[8] Collins, G.E.: Quantifier elimination by cylindrical algebraic decomposition – twenty years of progress. In Caviness, B.F., Johnson, J.R., eds.: Quantifier Elimination and Cylindrical Algebraic Decomposition, Springer-Verlag (1998) 8–23

[9] Jovanović, D., de Moura, L.: Solving non-linear arithmetic. In: Proceedings of the 6th international joint conference on Automated Reasoning. IJCAR'12, Springer-Verlag (2012) 339–354

[10] Ratschan, S.: Efficient solving of quantified inequality constraints over the real numbers. ACM Trans. Comput. Logic **7**(4) (October 2006) 723–748

[11] Franzle, M., Herde, C., Teige, T., Ratschan, S., Schubert, T.: Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure. Journal on Satisfiability, Boolean Modeling and Computation **1** (2007) 209–236

[12] Khanh, T.V., Ogawa, M.: SMT for polynomial constraints on real numbers. Electr. Notes Theor. Comput. Sci. **289** (2012) 27–40

[13] Zankl, H., Middeldorp, A.: Satisfiability of non-linear (ir)rational arithmetic. In: Proceedings of the 16th international conference on Logic for programming, artificial intelligence, and reasoning. LPAR'10, Springer-Verlag (2010) 481–500

[14] Ganai, M., Ivancic, F.: Efficient decision procedure for non-linear arithmetic constraints using cordic. In: Formal Methods in Computer-Aided Design, 2009. FMCAD 2009. (2009) 61 –68

[15] Gao, S., Kong, S., Clarke, E.M.: dreal: An smt solver for nonlinear theories over the reals. In: CADE 2013, Springer-Verlag (2013) 208–214

[16] Ngoc, D.T.B., Ogawa, M.: Overflow and roundoff error analysis via model checking. In: Proceedings of the 2009 Seventh IEEE International Conference on Software Engineering and Formal Methods. SEFM '09, IEEE Computer Society (2009) 105–114

[17] Ngoc, D.T.B., Ogawa, M.: Checking roundoff errors using counterexample-guided narrowing. In: Proceedings of the IEEE/ACM international conference on Automated software engineering. ASE '10, ACM (2010) 301–304

[18] Colón, M., Sankaranarayanan, S., Sipma, H.: Linear invariant generation using non-linear constraint solving. In: CAV. Volume 2725 of Lecture Notes in Computer Science., Springer (2003) 420–432

[19] Hirokawa, N., Middeldorp, A.: Tyrolean termination tool: Techniques and features. Information and Computation **205**(4) (2007) 474–511