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]12, 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.

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.


- **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