# Equality handling and efficiency improvement of SMT for non-linear constraints over reals.

August 27, 2014

Satisfiability Modulo Theories (SMT) problem is a decision problem for logical formulas with respect to background theories capturing the meaning of the formulas. There is a number of background theories such as theory of real numbers, theory of integers, and the theories of various data structures such as lists, arrays and bit vectors. Based on the background theories, SMT solvers decide whether a set of formulas is satisfiable or not. If it is (SAT), a satisfying assignment of variables which makes the constraints satisfiable is returned; otherwise (UNSAT) the proof of un-satisfiability is optionally generated. SMT has a wide range of applications, namely test-case generation and model checking. We have developed an SMT solver called raSAT [1], which solves polynomial constraints overs reals (logic division of quantifier-free non-linear reals arithmetic, QF_NRA). We choose QF_NRA because of the following two reasons:

1. In SMT-LIB, there is a number of methodologies for benchmarks on QF_NRA, but each of them has its own disadvantages:

    - Quantifier elimination by cylindrical algebraic decomposition (QE-CAD) was implemented in QEPCAD-B, Mathematica, Reduce/Redlog. DEXPTIME complexity in the number of variables is the problem of QE-CAD.
    - Interval Arithmetic is applied in many tools such as RSOLVER, dReal and iSAT. Since iSAT use only interval arithmetic to solve the constraints, its ability to solve SAT problem is really limited.
    - Bit-blasting: Input formulas are bit-blasting to propositional formulas which are then solved by SAT solver. MiniSmt and UCLID

applied this method. Because each variable will be represented by a number of propositional variables, this method suffers from high number of variables and high degree of polynomials.

- Linearization: CORD uses CORDIC [2] to approximate multiplication by a sequence of linear constraints. For each multiplication, CORD introduces a number of new variables in approximated formulas, the method thus suffers from high degree of polynomials.
- Virtual substitution is applied in Z3, Mathematica and Reduce. In this method, we need the formulas of roots of a polynomial. The degree of polynomials (with respect to the considered variable) is thus required to be up to 5.

2. Many applications are encoded into polynomial constraints such as:
    - Automated detection of round-off and overflow error [2] [3] which is the initial motivation of our work.
    - Automatic termination proving of term rewriting system: This is a second order SAT problem which is undecidable. It can be solve by interpreting terms into polynomials over the reals and amounting termination conditions to the constraints over the coefficients of such polynomials.
    - Invariant generation: [4] proposed a method of generating linear invariant using non-linear constraints solving based on Farkass Lemma.

raSAT focuses on strict inequalities since they allows approximations. raSAT uses raSATloop, a method of iterative approximation refinement, to solve polynomial constraints. Interval arithmetic (IA) (over-approximation) is used for disproving the constraints and testing (under-approximation) is used for proving. When raSAT neither proves nor disproves the constraints, refinements are applied to decompose the intervals of variables. In this work, I am going to solve the following problems: Improve the efficiency of raSAT. Handle equality. Handle polynomial constraints over Integer (QF_NIA). Current status Improve the efficiency of raSAT. a. Exploration of test cases: If for each variable we generate 2 test cases, totally we will have $2^n$ test cases with n is the number of variables. Testing phase will consume much time if n becomes large. We have the following solutions: Computation of affine interval maintains coefficient of the noise error reflecting the influence of a variable on the value of the polynomial. I implemented the strategy of prioritizing most influential variables in testing. When the intervals of variables are the same, we will probe on direction such as $x_1 = x_2 = \ = x_n = x$. Then, we can use the Sturm sequence as heuristics on test data generation. This is our future work in Master course. Also with the idea of Sturm sequence, we consider the multivariate polynomial, for instance, f(x, y) as a univariate polynomial g(x) with intervals as coefficients by IA as over-approximation. This is our future work in Master course. Handle equality. Handle polynomial constraints over Integer (QF_NIA). SAT, UNSAT confirmation: Roundoff, overflow errors can make our SAT and UNSAT result unsound. We plan to use iRRAM, package for error-bounded real arithmetic, to verify raSAT results. UNSAT core computation: By making statistics on running time of raSAT, we detected that UNSAT core takes quite long time relatively to the total running time. Testing phase: Statistics also showed that testing phase also accounts for a long running time. We plan to reduce the number of testing cases by focusing on only sensitive variables when generating test cases. Equality handling: We will the intermediate value theorem first to simply solve the equalities. With the above problems, the current status of my research is as follow. Our work is going to be presented at SMT Workshop 2014. raSAT also participated the SMT competition 2014 on QF_NRA. Our experiments on the competitions servers show that raSAT solved 57 problems of zankl family which is beyond the winner of 2010 competition (MiniSmt) SAT, UNSAT confirmation: I implemented SAT verification using iRRAM. UNSAT case is left for future work. UNSAT core computation: Because UNSAT requires exhaustive search for all the possibilities, we define UNSAT core to reduce the target constraints. Possible choices are: A subset  of constraints from the original ones : It is clear that if  is UNSAT then  is also UNSAT. We are investigating how to choose such subset. In the current research, I sorted the APIs using dependency between polynomials. For example, fg if all variables of f are also variables of g. A sub-polynomials g of f in an API f¿0: such that UNSAT of g¿0 results in UNSAT of f¿0 in a box. This idea was implemented in the previous work [1]. A subset V of variables appearing in an API f¿0: such that xV in a box are enough to lead UNSAT. This idea is new and was implemented in the current research. Equality handling: We consider a simple approach of using the intermediate value theorem. First, we find a box of variables in which all inequality constraints are valid. For single equation, suppose it is f(x)=0. We will find two points ($x_1$ ) and ($x_2$)

inside the box such that f(($x_1$ ))* f(($x_1$ ))¡0. Because f(x) is continuous (it is a polynomial), we can conclude that the equation has one solution lying between $x_1$ and $x_2$. Since all the in-equations are valid, the whole problem is SAT. In current research, we added handling for single equation. For multiple equations, suppose we have two equations $g_1$=0 and $g_2 = 0$ and the set of variables in $g_1$ and $g_2$ is x,y. Let the current considered box is x in $[a_1, b_1]$ and y in $[a_2, b_2]$. If, for example: $g_1(c_1, y) * g_1(d_1, y) < 0$ for some$c_1, d_1$ in $[a_1, b_1]$ and each y in $[a_2, b_2]$ and, $g_2(x, c_2) * g_2(x, d_2) < 0$ for some$c_2, d_2$ in $[a_2, b_2]$ and each x in $[a_1, b_1]$ We can conclude that there exist a common solution of $g_1 and g_2$ inside the box. Therefore the whole problem is SAT. We are implementing the case of multiple equations. NIA implementation: We will also extend raSAT for nonlinear integer arithmetics. One possible way is setting the minimum searching box to 1.