研究計画提案書（修士論文研究） 　　　　　　　　　　　　　　　　平成26年7月1日

Research Proposal for Master’s Thesis

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 氏名Name | VU XUAN TUNG | | 学生番号  Student Number | 1310007 | |
|  | | | | | |
| 主指導教員  Supervisor | Mizuhito Ogawa | 印  Seal | 副テーマ指導教員  Advisor for Minor Research | Nguyen Le Minh | 印  Seal |
| 副指導教員  Second Supervisor | Nao Hirokawa | 印  Seal |
|  | | | | | |
| ＜研究テーマ＞ Research Title | | | | | |
| Equality handling and efficiency improvement of SMT for non-linear constraints over reals. | | | | | |
|  | | | | | |
| ＜研究の目的＞ Research Aim | | | | | |
| Satisfiability Modulo Theories (SMT) problem is a decision problem for logical formulas with respect to combinations of background theories with constraints. Background theories capture the meaning of the formulas. Examples for background theories are linear arithmetic and nonlinear arithmetic. While linear constraints involve only and , non-linear ones allow more functions like multiplication.  Based on the background theories, SMT solvers decide whether a set of formulas is satisfiable or not. If it is (SAT), a satisfying assignment is returned; otherwise (UNSAT) the proof of un-satisfiability is optionally generated.  We have developed an SMT solver called raSAT [1], which solves polynomial constraints overs reals. Many applications are encoded into polynomial constraints such as automated detection of round-off and overflow error, automatic termination proving, invariant generation.  At the moment, raSAT decides polynomial constraints by numerical methods. It focuses on inequality since inequality allows approximations. Equality, on the other hand, does not allow approximations. Theoretically, it can be handled by using the ideal method such as Grobner basis [2].  raSAT uses interval arithmetic (IA) (over-approximation) for disproving polynomial constraints and testing (under-approximation) for proving. When it neither proves nor disproves, refinements are applied to decompose the intervals of variables. During interval arithmetic, UNSAT core of polynomial constraints are computed to improve the efficiency of theory propagation.  From experiments of raSAT, we observed following efficiency problems by looking at statistics:   1. UNSAT core sometimes takes much time, causing the result to be Timeout or causing non-termination of program. 2. Z3 [3] solves many UNSAT problems quickly even if the number of polynomials are large. We observed that such problems often contain small subsets of polynomials that lead UNSAT. 3. Random test data causes quite unstable results, though it boosts efficiency.   In this proposal, we present our approaches for efficiency improvement as well as extending raSAT to include equality. Target topics include:   1. Improving efficiency:    1. Representing UNSAT core using Binary Decision Diagram. Authors in [13] presented an idea of using BDD for encoding pseudo-boolean constraints    2. Strategy on test data generation. With the number of variables, the number of test data can easily exploited.    3. Incremental UNSAT detection: Starting a small subset of polynomials, enlarge step-wise as long as UNSAT is not detected. 2. Equality handling:    1. The intermediate value theorem which may not find SAT instances of equality but find witness of such existence.    2. Grobner basis [2]. | | | | | |
|  | | | | | |
| ＜研究の背景・特色＞ Research Background, Originality | | | | | |
| In SMT-LIB [4], methodology for benchmarks on non-linear real number arithmetic (QF\_NRA) has not been established.  There are a number of candidates such as:   1. Quantifier elimination by cylindrical algebraic decomposition [5] (QE-CAD) is a technique for deciding polynomial constraints. It is implemented in QEPCAD-B, Reduce/Redlog. Because QE-CAD is DEXPTIME complexity in the number of variables, solving problems with a lot of variables is a challenge for QE-CAD. 2. Interval Arithmetic is applied in many tools such as RSOLVER [6] and iSAT [7]. 3. Bit-blasting: Input formulas are bit-blasting to propositional formulas which are then solved by SAT solver. MiniSmt [8] and UCLID [9] applied this method. 4. Linearization idea in Barcelogic [10] and CORD [11]. 5. Virtual substitution idea [12] is applied in Z3 [3].   raSAT has the same approach with RSOLVER [6] and iSAT [7]. But while they use Classical Interval as IA, raSAT uses Affine Interval Arithmetic. In addition to IA, we also apply under-approximation (testing) which may find SAT instances. raSAT shares ideas of approximations with dReal [14] though the approaches are complementary.  For target problems on raSAT, our ideas for efficiency improvement are:   1. Representing UNSAT core using Binary Decision Diagram:    1. The idea comes from [13] in which the authors presented an idea of using BDD for encoding pseudo-boolean constraints.    2. We will apply this idea into UNSAT core representation.    3. The polynomial constraint, for example, f > c is represented by BDD. A node v is labelled with the interval [a, b] if f > [a, b] implies f > c. When two intervals at the same level of decision overlap, we can merge them. For example, if [1, 3] and [2, 5] are at the same decision level, then we can unify them and add a new interval [1, 5] as a label. 2. Strategy on test data generation:    1. When the number of variables increases, the number of test data increase exponentially. To adjust as strategy to generate test data, we first probe on one direction such as: . Then we can assume rough behavior of the changes of signs.    2. Use the idea of Sturm sequence as heuristics on test data generation.  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.   Sturm sequence is used for finding witness of sign changes of univariate polynomial, and we can use the idea for generating test data (each witness is a test point) on the above g(x).   1. Incremental UNSAT detection:    1. We plan to find strategies for finding a subset of constraints which is the cause of UNSAT.    2. First trial is the dependency between polynomials. For example, if all variables of f are also variables of g. We gradually check the polynomials based on such a relation.   Before applying ideal methods for handling equality, we will consider at two levels:   1. Use of the intermediate value theorem: 2. First, we find a box of variables in which all inequality constraints are valid. 3. For single equation:    1. Suppose the equation is .    2. We will find two points and inside the box such that \*.    3. Because is continuous (it is a polynomial), we can conclude that the equation has one solution lying between and .    4. Since all the in-equations are valid, the whole problem is SAT. 4. For multiple equations:    1. Suppose we have two equations and and the set of variables in and is .    2. Let the current considered box is and .    3. If, for example:       * for some and each and,       * for some and each   We can conclude that there exist a common solution of and inside the box.   * 1. Since all the in-equations are valid, the whole problem is SAT. | | | | | |
| 1. Grobner basis: Grobner has been implemented in Maple and Mathematica. Unfortunately, the performance is quite limited due to the high complexity. We will use Grobner basis as the last effort because: 2. The algorithm for constructing Grobner basis is easy to understand and implement. 3. Although the worst-case complexity is quite high (exponential), concrete examples often have a lot of special structures which cause the Grobner basis to be computed in reasonably short time. 4. There are ideas for speeding up the Grobner basis computation: 5. B. Buchberger in (14) suggested the use of criteria for eliminating unnecessary reductions during computation of Grobner basis. 6. A p-adic approach of Franz Winkler in (15) 7. A floating point approach of F. Pauer in (16) 8. The "Gröbner Walk" approach in (17) 9. The "linear algebra" approach in (18) 10. Heuristics and strategies for choosing favorable orderings of monomials during construction process. 11. Good implementation techniques and data structures.   We will apply the above ideas into our implementation. We will use Maude, a high performance language for rewriting. References 1. **To, Van Khanh and Ogawa, Mizuhito.** *raSAT: SMT for Polynomial Inequality.* s.l. : JAIST, 2013.  2. *Gröbner Bases and Applications.* **B. Buchberger, F. Winkler.** RISC, Austria : Cambridge University Press, 1970. Proceedings of the International Conference "33 Years of Gröebner Bases". pp. 535 -545.  3. *Z3: An Efficient SMT Solver.* **{De Moura, Leonardo and Bjorner, Nikolaj.** Budapest, Hungary : Springer-Verlag, 2008.  4. *The satisfiability modulo theories library (SMT-LIB).* **Barrett, C., Stump, A., and Tinelli, C.**  5. **Collins, G.E.** *Quantiﬁer elimination by cylindrical algebraic decomposition – twenty years of progress.* s.l. : Springer-Verlag, 1998.  6. *Efficient Solving of Quantified Inequality Constraints over the Real Numbers.* **Ratschan, Stefan.** s.l. : ACM, 2006, Vol. 7.  7. *Efﬁcient solving of large non-linear arithmetic constraint systems with complex boolean structure.* **Franzle, M., Herde, C., Teige, T., Ratschan, S., Schubert, T.**  8. *Satisfiability of non-linear (ir)rational arithmetic.* **Zankl, H., and Middeldorp, A.**  9. *Deciding bit-vector arithmetic with abstraction.* **Bryant, R. E., Kroening, D., Ouaknine, J., Seshia, S. A., Strichman, O.,and Brady, B.**  10. *Solving non-linear polynomial arithmetic via sat modulo linear arithmetic.*  **11. *Effcient decision procedure for non-linear arithmetic constraints using cordic.* Ganai, M., and Ivancic, F.**  **12. *Quantifier for elimination for real algebra - the quadratic case and beyond.* Weispfenning, V.**  **13. *BDDs for Pseudo-boolean Constraints: Revisited.* Abío, Ignasi and Nieuwenhuis, Robert and Oliveras, Albert and Rodríguez-Carbonell, Enric. s.l. : Springer-Verlag, 2011. Proceedings of the 14th International Conference on Theory and Application of Satisfiability Testing. pp. 61--75.**  **14. *A Criterion for Detecting Unnecessary Reductions in the Construction of Groebner Bases.* Buchberger, Bruno. s.l. : Springer-Verlag, 1979. Proceedings of the International Symposiumon on Symbolic and Algebraic Computation. pp. 3--21.**  **15. *A P-adic Approach to the Computation ofGöbner Bases.* Winkler, Franz. 1988, J. Symb. Comput., pp. 287--304.**  **16. *On Lucky Ideals for Gröbner Basis Computations.* Pauer, Franz. 1992, J. Symb. Comput., pp. 471--482.**  **17. *Converting Bases with the Gröbner Walk.* Collart, S. and Kalkbrener, M. and Mall, D. 1997, J. Symb. Comput., pp. 465--469.**  **18. *Efficient Computation of Zero-dimensional Gröbner Bases by Change of Ordering.* Faugère, J. C. and Gianni, P. and Lazard, D. and Mora, T. 1993, J. Symb. Comput., pp. 329--344.** | | | | | |
| ＜研究計画・方法＞Research Plan, Method | | | | | |
| This research comprises:   1. April: Representing UNSAT core using Binary Decision Diagram.   We investigate the ideas in the referenced paper and try to apply it into raSAT.   1. May: Strategy on test data generation.   We investigate mathematical theories on one-variables polynomials and try to apply them in generating test data.   1. June – July: Incremental UNSAT detection and Strategy for nested let-constructs problems   The above ideas for let-constructs problems will be applied, just for detecting UNSAT first. Because at this moment, the equality handling has not been added.  We also investigate for finding strategies of incremental UNSAT and implement them.   1. August: Simple equality handling using the intermediate value theorem.   Above detailed ideas will be implemented.   1. September: Implement Buchberger’s algorithm for finding Grobner basis,   We design the data structures, techniques and implement the algorithm by Maude language.   1. October: applying into raSAT.   After generating Grobner basis, we need to decide the satisfiability of the polynomials in Grobner basis along with the inequalities in the original problem.  We also integrate the equality techniques in solving SAT instances in let-constructs problems.  After this period, raSAT should be able to solve equations. The optimization would has not been applied, the running time may be high. But the running time should be controlled. That mean if the timeout has been reached, the computation of Grobner basis should be stopped even if it was not been completed.   1. November: Optimization of Grobner transformation.  We will read in details the above ideas and implement it. 2. December: experiences on SMT benchmarks and make comparison with Z3.ssss 3. Writing the thesis (January –March) | | | | | |
|  | | | | | |
| ＜現在までに単位修得した専門科目＞ 科目数：専門科目数を書く、領域数：専門領域数を書く  IS courses you have obtained credits from:　 (Number of courses: 9 Number of areas: 4 ) | | | | | |
| Programming Laboratory II, Mathematical Logic, Natural Language Processing I, Computer Networks, Functional Programming, Software Design Methodology, Foundation of Software Environment, Algebraic Formal Methods, Term Rewriting. | | | | | |