raSAT: SMT for Polynomial Inequality

To Van Khanh[1] and Mizuhito Ogawa[2]

[1] University of Engineering and Technology, Vietnam National University, Hanoi
khanhtv@vnu.edu.vn
[2] Japan Advanced Institute of Science and Technology
mizuhito@jaist.ac.jp

1 Experiments

We implemented **raSAT** loop as an SMT **raSAT**, based on MiniSat 2.2 as a backend SAT solver. In this section, we are going to present the experiments results which reflect how effective our designed strategies are. In addition, comparison between raSAT, Z3 and iSAT3 will be also shown. The experiments were done on a system with Intel Xeon E5-2680v2 2.80GHz and 4 GB of RAM. In the experiments, we exclude the problems which contain equalities because currently raSAT focuses on inequalities only.

1.1 Effectiveness of designed strategies

There are three immediate measures on the size of polynomial constraints. They are the highest degree of polynomials, the number of variables, and the number of APIs. Our strategies focus on selecting APIs(for testing) and selecting variable (for multiple test cases and for decomposition), selecting decomposed box of the chosen variable in TEST-UNSAT API. Thus, in order to justify the effectiveness of these strategies, we need to test them on problems with varieties of APIs number (selecting APIs) and varieties of variables number in each API (selecting variable). For this criteria, we use Zankl family for evaluating strategies. The number of APIs for this family varies from 1 to 2231 and the maximum number of variables in each APIs varies from 1 to 422.

As mentioned, raSAT uses SAT likely-hood evaluation to judge the difficulty of an API. In testing, raSAT chooses the likely most difficult API to test first. In box decomposition, again SAT likely-hood is used to evaluate decomposed boxes again the TEST-UNSAT API. For easy reference, we will name the strategies as followings

- Selecting APIs, selecting decomposed boxes:
  (1) Using SAT likely-hood
  (2) Randomly
- Selecting one variable each API for testing, one variable of TEST-UNSAT API for decomposition:
  (3) Using sensitivity
  (5) Randomly

Strategies	Group	SAT	UNSAT	time(s)
(1)-(5)	matrix-1-all-*	22	11	27.09
(2)-(5)	matrix-1-all-*	21	10	933.78
(1)-(3)	matrix-1-all-*	31	10	765.48

Table 1. Experiments of strategies on Zankl family

We will compare the combinations $(1)-(5)$ and $(2)-(5)$ to see how SAT likely-hood affects the results. And comparison betweeen $(1)-(5)$ and $(1)-(3)$ illustrates the effectiveness of sensitivity. In these experiments, timeout is set to $500s$.

Table 1 shows the results of strategies. While $(1)-(5)$ solved 33 problems in $27.09s$, $(2)-(5)$ took $933.78s$ to solve 31 problems. In fact, among 31 solved problems of $(2)-(5)$, $(1)-(5)$ solved 30. The remaining problem is a SAT one, and $(2)-(5)$ solved it in $34.36s$. So excluding this problem, $(2)-(5)$ solved 30 problems in $933.78s - 34.36s = 899.42s$ and $(1)-(5)$ took $27.09s$ to solved all these 30 problems plus 3 other problems. Using SAT likely-hood to select the most difficult API first for testing is very reasonable. This is because the goal of testing is to find an assignment for variables that satisfies **all** APIs. Selecting the most difficult APIs first has the following benefits:

- The most difficult APIs are likely not satisfied by most of test cases (of variables in these APIs). This early avoids exploring additional unnecessary combination with test cases of variables in other easier APIs.
- If the most difficult APIs are satisfied by some test cases, these test cases might also satisfy the easier APIs.

raSAT uses SAT likely-hood to choose a decomposed box so that the chosen box will make the TEST-UNSAT API more likely to be SAT. This seems to work for SAT problems but not for UNSAT ones. In fact, this is not the case. Since for UNSAT problems, we need to check both of the decomposed boxes to prove the unsatisfiability (in any order). So for UNSAT constraints, it is not the problem of how to choose a decomposed box, but the problem of how to decompose a box. That is how to choose a point within an interval for decomposition so that the UNSAT boxes are early separated. Currently, raSAT uses the middle point. For example, $[0, 10]$ will be decomposed into $[0, 5]$ and $[5, 10]$. The question of how to decompose a box is left for our future work.

Combination $(1)-(3)$ solved 41 problems in comparison with 33 problems of $(1)-(5)$. There are 5 large problems (more than 100 variables and more than 240 APIs in each problem) which were solved by $(1)-(3)$ (solving time was from $20s$ to $300s$) but not solved (timed out - more than $500s$) by $(1)-(5)$. Choosing the most important variable (using sensitivity) for multiple test cases and for decomposition worked here.

Solver	Zankl (151)			Meti-Tarski (5101)		
	SAT	UNSAT	time(s)	SAT	UNSAT	time(s)
isat3	14	15	209.20	2916	1225	885.36
raSAT	31	10	765.48	3322	1052	753.00
Z3 4.3	**54**	**23**	**1034.56**	**3528**	**1569**	**50235.39**

Table 2. Experimental results for Hong, Zankl, and Meti-Tarski families

1.2 Comparison with other tools on QF_NRA and QF_NIA benchmarks

Comparison with Z3 and isat3 on QF_NRA. Table 2 presents the results of isat3, raSAT and Z3 on 2 families (Zankl and Meti-tarski) of QF_NRA. Only problems with inequalities are extracted for testing. The timeout for Zankl family is $500s$ and for Meti-tarski is $60s$ (the problems in Meti-tarski are quite small - less than 8 variables and less than 25 APIs for each problem). For isat3, ranges of all variables are uniformly set to $[-1000, 1000]$

Both isat3 and raSAT use interval arithmetics for deciding constraints. However, raSAT additionally use testing for accelerating SAT detection. As the result, raSAT solved larger number of SAT problems in comparison with isat3. In Table 2, isat3 conludes UNSAT when variables are in the ranges $[-1000, 1000]$, while raSAT conclude UNSAT over $[-infinity, infinity]$.

In comparison with Z3, for Zankl family, Z3 4.3 showed better performance than raSAT in the problems with lots of small constraints: short monomial (less than 5), small number of variables (less than 10). For problems where most of constraints are long monomial (more than 5) and have large number of variables (more than 10), raSAT results are quite comparable with ones of Z3, often even outperfoming when the problem contains large number of vary long constraints (more than 40 monomials and/or more than 20 variables). For instance, *matrix-5-all-27, matrix-5-all-01, matrix-4-all-3, matrix-4-all-33, matrix-3-all-5, matrix-3-all-23, matrix-3-all-2, matrix-2-all-8* are such problems which can be solved by raSAT but not by Z3.

Meti-tarski family contains quite small problems (less than 8 variables and less than 25 APIs for each problem). Z3 solved all the SAT problems, while raSAT solved around 94% (33322/3528) of them. Approximately 70% of UNSAT problems are solved by raSAT. As mentioned, raSAT has to explore all the decomposed boxes for proving unsatisfiability. Thus the question "How to decompose boxes so that the UNSAT boxes are early detected?" needs to be solved in order to improve the ability of detecting UNSAT for raSAT.

Comparison with Z3 on QF_NIA. We also did experiments on QF_NIA/AProVE benchmarks to evaluate raSAT loop for Integer constraints. There are 6850 prob-

Solver	SAT	UNSAT	time (s)
raSAT	6764	0	1230.54
Z3	**6784**	**36**	**139.78**

Table 3. Experiments on QF_NIA/AProVE

lems which do not contain equalities. The timeout for this experiment is $60s$. Table 3 shows the result of Z3 and raSAT for this family.

raSAT solved almost the same number of SAT problems as Z3 (6764 for raSAT with 6784 for Z3). Nearly 99% problems having been solved by raSAT is a quite encouraging number. Currently, raSAT cannot conclude any UNSAT problems. Again, this is due to exhaustive balanced decompositions.