NRA-LS at the SMT Competition 2022

Minghao Liu[1,3], Fuqi Jia[1,3], Rui Han[1,3], Yu Zhang[2,3], Pei Huang[1,3], Feifei Ma[1,2,3], and Jian Zhang[1,3]

[1] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
[2] Laboratory of Parallel Software and Computational Science, Institute of Software, Chinese Academy of Sciences, Beijing, China
[3] University of Chinese Academy of Sciences, Beijing, China
{liumh,jiafq,maff}@ios.ac.cn

1 Introduction

Satisfiability modulo theory (SMT) solving for quantifier-free formulas in non-linear real arithmetic (QF_NRA) is important in many applications. State-of-the-art SMT solvers have made great progress to solve this problem. However, the time and memory usage of them on some hard instances may be unacceptable, especially when high-order polynomials appear in the formula. NRA-LS is an SMT solver for QF_NRA theory, which can improve the performance on some high-order satisfiable instances through a local search (LS) algorithm. NRA-LS wraps CVC5-1.0.0[1] as the back-end solver.

2 Architecture of NRA-LS

The framework of NRA-LS is shown in Algorithm 1. At the beginning, the maximum order of polynomials in the formula is computed, and those formulas will be handled specially if they contain high-order polynomials, which means the order is larger than 10 in the implementation.

Initial model generation. NRA-LS tries to assign values to the variables, evaluates the level to which the assertions are satisfied, and adjusts the values. Then the top-k assignments are output as initial models. However, these 'models' cannot satisfy all the assertions in most cases, so NRA-LS makes fewer variables fixed and tests the satisfiability of a set of sub-formulas.

Sub-formulas Testing. Given an initial model, NRA-LS calls back-end solver to test if the model is valid by appending additional assertions to the original formula. If `unsat` is returned, NRA-LS will reduce the number of fixed variables, and test the new sub-formula iteratively until getting `sat` or the time limit is exceeded. If `sat` is returned, the original formula is also satisfiable.

[1] https://github.com/cvc5/cvc5.

Algorithm 1 Framework of NRA-LS

Input: an SMT(QF_NRA) formula ϕ
Output: `sat/unsat/unknown`

if ϕ contains high-order polynomial **then**
  $S_1, S_2, \ldots, S_k \leftarrow$ `generate_init_model(`ϕ`)`;
  for i from 1 to k **do**
    for num_fixed_vars from $\#var(\phi)$ to 1 **do**
      $\phi' \leftarrow$ `generate_sub_formula(`ϕ`,` S_i`,` num_fixed_vars`)`;
      $res \leftarrow$ `run_back_end_solver(`ϕ'`)`;
      if $res =$ `sat` **then**
        return res;
      else if $res =$ `unsat` **then**
        continue;
      else
        break;
      end if
    end for
  end for
end if
return `run_back_end_solver(`ϕ`)`;

Time slots assignment. To solve the SMT formulas with high-order polynomials, NRA-LS assigns the time slots into three parts. Suppose the time limit to solve a single formula is T. First, it takes $5\%T$ to run back-end solver on the original formula, which aims to exclude those easy benchmarks. Next, the time limit for each attempt that tests a sub-formula is set to $2.5\%T$. Finally, if the result cannot be determined, the rest of the time is assigned to run back-end solver on the original formula. Besides, for those SMT formulas without high-order polynomials, all time slots are assigned to back-end solver directly.

3 Project Website

More information and resources of NRA-LS are available on the website:

https://github.com/minghao-liu/NRA-LS.