Simplics: Tool Description

Bruno Dutertre, Leonardo de Moura
Computer Science Laboratory, SRI International, Menlo Park, CA
bruno@csl.sri.com,demoura@csl.sri.com

July 1, 2005

1 Introduction

Simplics is a recent successor of the Integrated Solver and Canonizer (ICS) [2]. However, unlike ICS, Simplics is specialized for a narrower set of problems. It is a tool for deciding the validity of boolean combinations of linear-arithmetic constraints over the reals. The main intended application of Simplics is bounded model checking of infinite transition systems, including several forms of timed transition systems. Examples of applications include distributed real-time fault-tolerant systems and clock-synchronization protocols.

2 Architecture and Implementation

Simplics is written entirely in Ocaml (version 3.08.1), with the exception of C-code for interfacing Ocaml with the GNU Multiprecision Arithmetic Library (GMP version 4.1.3).[1]

Simplex Solver

The core of Simplics is a solver for real linear arithmetic that relies on the simplex algorithm. This solver supports incremental addition of linear constraints (equalities, strict and non-strict inequalities, and disequalities). The solver maintains an internal state which is a tableau-based representation of the set of constraints asserted so far. It can detect and propagate implied equalities (e.g., $x \leq z, y \leq z, x + y = 2z \rightarrow x = y$). The solver also produces non-redundant explanations when an asserted constraint is not satisfiable or valid in the current state.

To generate such explanations, the solver keeps track of linear dependencies between the asserted constraints using "zero-slack variables" (in addition to the standard simplex stack variables). The method and algorithms employed are described in detail in [3]. These extra zero-slack variables also enable memory-efficient backtracking.

[1] This part of Simplics is derived from the Ocaml-GMP interface originally written by David Monniaux and revised for ICS by Jean-Christophe Filliâtre and Harald Rueß.

SAT Solver

The previous linear-arithmetic solver is combined with a relatively classic SAT solver, also implemented in Ocaml. This SAT solver works using a clausal representation of the input problem and uses well-known techniques introduced by Chaff and similar SAT solvers: two watched literals, non-chronological backtracking based on learned clauses, and UIP computation. The integration with the simplex solver is also fairly traditional: Explanations generated by the simplex solver when a conflict is detected are translated to clauses and used for backtracking. The general approach follows [1].

3 Acknowledgments

Simplics benefited greatly form the help and suggestions of Natarajan Shankar and Harald Rueß. The simplex implementation at the core of Simplics is based on their previous work on integrating a linear-arithmetic solver within ICS, as described in [3].

References

[1] Leonardo de Moura and Harald Rueß. Lemmas on Demand for Satisfiability Solvers. In: Fifth International Symposium on the Theory and Applications of Satisfiability Testing, (2002)

[2] Leonardo de Moura, Harald Ruess, Natarajan Shankar, and John Rushby, The ICS Decision Procedures for Embedded Deduction. Proc. 2nd International Joint Conference on Automated Reasoning (IJCAR'04), Volume 3097 of Lecture Notes in Computer Science, Springer-Verlag (2004) 218–222.

[3] Harald Rueß and Natarajan Shankar. Solving Linear Arithmetic Constraints. Technical Report, CSL-SRI-04-01, SRI International, January 2004.