Constraint Programming for Analysis of Rewriting (Proposal for course in Advanced Track of ISR)

Johannes Waldmann, HTWK Leipzig
https://www.imn.htwk-leipzig.de/~waldmann/

Abstract

We show how to use methods and tools of constraint programming, in particular, SAT and SMT solving, to analyze properties of rewrite systems. The technical focus is not on the properties themselves (we assume students know them already) but on their formulation via embedded domain-specific languages for propositional and predicate logic.

1 Motivation

Certain properties of rewriting systems are implied by the existence of certain objects. For instance, a precedence, or a matrix interpretation, can serve as a certificate of termination. We will show how to implement the search for these certificates via Constraint Programming.

In this approach, we write the desired properties of the certificate object as a formula in propositional, or predicate logic; and then apply a constraint solver to find a satisfying assignment (a model). The solver contains search algorithms that are domain-specific (e.g., domain = linear inequalities), but problem-agnostic (e.g., problem = the termination problem). Solvers are sophisticated, and powerful. Progress is measured at `http://satcompetition.org/` and `https://smt-comp.github.io/`.

Propositional encoding of termination certificates was pioneered by Kurihara and Kondo in 1999, cf. [2], and has been widely applied since around 2006. For an overview, see, e.g., Fuhs' thesis [1].

In practice, it is important to write the constraints in a language that is readable, expressive, and safe. The actual interface languages of the solvers (DIMACS, SMTLIB) do not meet these criteria, since they are low-level by design. In the course, we will use constraint programming languages that are embedded (as libraries) in a high-level programming language (Haskell).

2 Course Outline

- Introduction
  - definition of the satisfiability problem (SAT), the SAT-modulo-Theory problem (SMT);
  - the standard interface languages (DIMACS, SMTLIB), small hand-written examples
  - using state-of-the-art solvers (e.g., kissat, z3)
- Practical Aspects of SAT encoding
  - examples of rewrite properties that we want to encode:
    - reachability, LPO precedence, matrix interpretation for string rewriting
    - confluence, termination of (finite) abstract reduction systems [3]
  - the `ersatz` library — how to: write formulas, call solvers, process their output,
  - using the power of the host language: higher-order functions, polymorphism and type classes, type-level numbers (e.g., for matrix dimensions)

  - (optional) some internals of `ersatz`
- (optional) Extensions
  - SMT encoding, for T $\in$ { linear arithmetic, polynomial arithmetic, bit vectors, word equations }
  - Finite Domain constraint programming, the minizinc standard
  - Answer Sets, and their relation to SAT

  for SMT, we can do one or two examples; for the other topics, only an overview

3 Organization

Expected duration: the above is for 3 slots of 90 min each but it can be modified by dropping optional parts.
Each class will be a mixture of presentation (by me) and experiment (by students).
Students should already have a working understanding

- of the rewriting properties that we are going to encode,
- of a functional programming language and compiler, preferably Haskell and GHC.

For the experiments, I expect that students

- bring their computer, have online access (to read documentation, to install missing libraries)
- have compiler (GHC), tools (stack), libraries (ersatz) and solvers (minisat, kissat, z3) already installed
- will git-clone a repository provided by me, containing examples and exercises

Students will work on code that looks like this: a self-contained SAT encoding (125 lines only) of a matrix interpretation for Zantema's problem $a^2b^2 \to b^3a^3$: `https://github.com/ekmett/ersatz/blob/master/tests/Z001.hs`

4 About the Speaker

Johannes Waldmann uses SAT encoding for termination analysis in his program `matchbox` `https://gitlab.imn.htwk-leipzig.de/waldmann/pure-matchbox` since 2005. He is a contributor to the `ersatz` library for SAT encoding.
He is a professor of computer science at Hochschule für Technik, Wirtschaft und Kultur, Leipzig, Germany. His teaching includes Concepts of Programming Languages, Declarative Programming, Constraint Programming, Symbolic Computation, and Computer Music.

References

[1] Carsten Fuhs. *SAT encodings: from constraint-based termination analysis to circuit synthesis*. PhD thesis, RWTH Aachen University, 2012.

[2] Masahito Kurihara and Hisashi Kondo. Efficient BDD encodings for partial order constraints with application to expert systems in software verification. In Robert Orchard, Chunsheng Yang, and Moonis Ali, editors, *Innovations in Applied Artificial Intelligence, 17th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, IEA/AIE 2004, Ottawa, Canada, May 17-20, 2004. Proceedings*, volume 3029 of *Lecture Notes in Computer Science*, pages 827–837. Springer, 2004.

[3] Hans Zantema. Finding small counterexamples for abstract rewriting properties. *Mathematical Structures in Computer Science*, 28(8):1485–1505, 2018.