mSAT: an ocaml sat solver

Guillaume Bury

Inria/ LSV/ENS Paris-Saclay, Cachan, France
guillaume.bury@inria.fr

Abstract. We present mSAT, a modular sat solving library written entirely in ocaml. While there already exists ocaml bindings for sat solvers written in C, mSAT provides more features, such as proof output, and the ability to instantiate an SMT solver using a first-order theory given by the user, while being reasonnably efficient compared to the existing bindings.

Introduction

Deciding the satisfiability of propositional logical formulae is an interesting and recurring problem for which sat solvers are now the standard solution. However, there exists very few sat solvers available in ocaml. There are currently 4 opam[2] packages that provides an ocaml library for sat solving[1], 3 of which are bindings to one or more sat solvers written in C (such as minisat[3]). The last and only one written in pure ocaml is mSAT, which we will present.

Sat and SMT solving

Sat solvers work on propositional clauses, i.e disjunctions of atomic propositions. A set of propositional clauses is satisfiable iff there exists a model for it, i.e an assignment from the atomics propositions to booleans, such that for every clause there exists an atomic proposition in the clause which is assigned to $\top$. A sat solver solves the satisfiability problems, i.e for an input problem given as a set of clauses, it returns either 'sat' if the problem is satisfiable, or 'unsat' if it is not.

SMT[2] solvers are extensions of sat solvers to first-order quantifier-free clauses such as $\neg(x + 1 > y) \lor \neg(y > z - 5) \lor (x > z - 10)$. In order to verify the satisfiability of a set of first-order clauses, a sat solver is combined with a first-order theory: the basic idea is for the sat solver to enumerate all propositional models, and for each of them, ask the theory if it is satisfiable[3]. If the theory returns 'sat' for at least one propositional model, the SMT solver returns 'sat', in the other case, it returns 'unsat'. That way the theory only has to deal with the satisfiability of a set of first-order atomic propositions, while the sat solver deals with the propositional structure of the input problem.

[1] There are a few additional packages for SMT solving, which we do not include because of the added complexity of SMT solving compared to sat solving

[2] Acronym for Satisfiability Modulo Theory

[3] In practice, incomplete models are incrementally given to the theory, and efficient backtracking is used in order to reach good performances

mSAT[1] is a library entirely written in OCaml, which derives from Alt-Ergo Zero[4], itself derived from the SMT solver Alt-Ergo[5][4]. mSAT provides functors to instantiate SMT solvers, and by extension sat solvers, since an SMT solver with an empty theory (i.e a theory which always returns ‘sat’) is a sat solver. mSAT supports the same features as current sat solvers, for instance, local assumptions, but more importantly provides unique and original features like the ability to instantiate one's own SMT solver and a proof output for unsatisfiable problems.

Presentation

The presentation will introduce the basics of sat and SMT solving, and then explain the interfaces exported by the library, focusing on three different points:

- How to use the sat solver exported by mSAT
- How to instantiate a custom SMT solver using mSAT, explaining in detail the various documented invariant expected to create a fully operational SMT solver. This will be accompanied by an example of such instantiation.
- How to use and benefit from the proof output of mSAT

Lastly, there will be a comparison between mSAT and the other sat solvers in ocaml ecosystem, concerning available features as well as performance.

References

1. mSAT: a modular sat/smt solver with proof output. `https://github.com/Gbury/mSAT`
2. Opam: a Package manager for ocaml. `https://opam.ocaml.org`
3. MiniSat is a minimalistic, open-source SAT solver. `http://minisat.se/Main.html`
4. Alt-Ergo Zero is an OCaml library for an SMT solver. `http://cubicle.lri.fr/alt-ergo-zero/`
5. Alt-Ergo. `http://alt-ergo.lri.fr/`

[4] Apart from the inital fork of Alt-Ergo Zero, it is of interest to note that the developpement of mSAT is completely separate from that of Alt-Ergo Zero and Alt-Ergo