Proof Assistants
Automated Reasoning

Guillaume Melquiond

INRIA Saclay – Île-de-France
Laboratoire de Recherche en Informatique

2011-02-15

Outline

How to automatically prove the numerous verification conditions produced by a weakest precondition calculus?

1. SMT Solvers
   - Basic Concepts
   - Supported Theories
2. Interactive Provers
   - Dedicated Approaches
   - Quantifier Elimination

SMT Solvers

SMT = Satisfiability Modulo Theories.

Components :

- a SAT solver for propositional formulas,
- a heuristic for instantiating lemmas (e.g., triggers),
- a way to combine theories (e.g., Nelson-Oppen algorithm),
- dedicated solvers for finding contradictory literals in particular theories (e.g., omega test, simplex).

Usage for program verification: negate the goal and prove it is unsatisfiable.

Propositional Solving

1. (Lazily) put the formula into CNF. (SAT)
2. Propagate any forced literal into the other disjunctions. (SAT)
   Does that produce an empty disjunction in the formula?
   If yes, the original goal cannot be satisfied.
3. Modify the formula in one of the following ways:
   1. Take a literal and add it, first as true, then as false. (SAT)
      Recursively handle the two new normal formulas.
   2. Ask the supported theories which literals are contradictory. (SMT)
      Negate them and add their disjunction to the normal formula.
   3. Select a lemma (syntactic: symbol occurrences). (Matching)
      Add its instantiation to the normal formula.
4. Go back to step 2.

Supported Theories

- Congruence:

$$a = c \wedge f(a, b) \neq f(c, d) \quad \Rightarrow \quad b \neq d.$$

- Presburger's Arithmetic:

$$2x + 1 = 2y \quad \Rightarrow \quad \perp.$$

- Linear arithmetic over $\mathbb{Q}$:

$$x + 2y \geq 5 \wedge 3x - y > 2 \quad \Rightarrow \quad x > 15/7.$$

- Unbounded functional arrays:

$$t[i] \neq t\{j \leftarrow v\}[i] \quad \Rightarrow \quad i = j.$$

- Lists and/or inductive types:

$$\mathrm{hd}(\mathrm{cons}(h, t)) = h.$$

- Bit-vectors.

Interactive Provers

SAT and SMT can still be used in interactive provers through reflection/extraction and/or execution traces.

But interactive provers also offer some dedicated automations:

- Prolog-like inference, (`[e]auto` in Coq)
- automated induction, (rippling in Isabelle)
- algebraic equalities, (`ring` and `field` in Coq)
- quantifier elimination.

Quantifier Elimination

Knowing how to eliminate $\exists x, \bigwedge L_i$ (with L_i literals) is enough.

- Dense linear orders: $x < y \Rightarrow \exists z,\ x < z < y$.
- Presburger's arithmetic: $(\mathbb{Z}, +, \leq, k|\cdot)$
- Linear arithmetic: $(\mathbb{Q}, +, \leq)$
- Mixed linear arithmetic: $(\mathbb{Q}, +, \lfloor\cdot\rfloor, \leq)$
- Polynomials: $(\mathbb{C}, +, \times, =)$
- Real closed fields: $(\mathbb{R}, +, \times, \leq)$

Note: quantifier elimination is a costly approach.