IFIP WG 10.4 Research Report, Tucson AZ, 19 Feb 2006

SMT Solvers

John Rushby

Computer Science Laboratory
SRI International
Menlo Park CA USA

SMT Solvers

- Anything a SAT solver can do, an SMT solver can do better
- SAT solvers are used for
  - Bounded model checking, and
  - AI planning,

  among other things

SAT Solving

- Find satisfying assignment to a propositional logic formula
- Formula often represented as a set of clauses
  - CNF: conjunction of disjunctions
  - Find an assignment of truth values to variable that makes at least one literal in each clause TRUE
- Example: given following 4 clauses
  - A, B
  - C, D
  - E
  - $\bar{A}, \bar{D}, \bar{E}$

  A solution is $A, C, \bar{D}, E$ (A, D, E is not)
- Do this when there are 1,000,000 variables and clauses

SAT Solvers

- SAT solving is quintessential NP-complete problem
- But now amazingly fast in practice (most of the time)
  - Breakthroughs (starting with Chaff) since 2001
  - Sustained improvements, honed by competition
- Has become commodity technology
  - Can think of it as massively efficient search
- Used in bounded model checking and in AI planning
  - Routine to handle 10^{300} states

Bounded Model Checking (BMC)

- Is there a counterexample to property P in k steps or less?
- System specified by initiality predicate I and transition relation T on states S
- Does there exist assignments to states $s_0, \ldots, s_k$ such that

$$I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge \cdots \wedge T(s_{k-1}, s_k) \wedge \neg(P(s_1) \wedge \cdots \wedge P(s_k))$$

- Given a Boolean encoding of I, T, and P (i.e., circuit), this is a propositional satisfiability (SAT) problem
  - Symbolic model checking uses same representation as BMC, different backend (BDDs)
- Extends from refutation to verification via k-induction
- Also generates plans (test cases)
  - counterexample to negation of property
  - Though specialized planning languages provide better frontends to the SAT solver than a model checker

Satisfiability Modulo Theories (SMT)

- SAT can encode operations and relations on bounded integers (bitvector representation), and other finite data types and structures
- But not unbounded or infinite types (e.g., reals), or structures (e.g., queues, lists)
- And even bounded arithmetic can be slow
- There are fast decision procedures for these theories
- But they work only on conjunctions of clauses
- General propositional structure requires case analysis
  - Should use efficient search strategies of SAT solvers

That's what an SMT solver does

SMT Solving

- Individual decision procedures decide conjunctions of formulas in their decided theories
- Combinations of decision procedures (using, e.g., Nelson-Oppen or Shostak methods) decide conjunctions over the combined theories (e.g., arithmetic plus arrays)
- SMT allows general propositional structure
  - e.g., $(x \leq y \vee y = 5) \wedge (x < 0 \vee y \leq x) \wedge x \neq y$

    ...possibly continued for 1000s of terms
- Should exploit search strategies of modern SAT solvers
- So replace the terms by propositional variables
  - $(A \vee B) \wedge (C \vee D) \wedge E$
- Get a solution from a SAT solver (if none, we are done)
  - e.g., A, D, E

SMT Solving by "Lemmas On Demand"

- Restore the interpretation of variables and send the conjunction to the core decision procedure
  - e.g., $x \leq y \wedge y \leq x \wedge x \neq y$
- If satisfiable, we are done
- If not, ask SAT solver for a new assignment—but isn't it expensive to keep doing this?
- Yes, so first, do a little bit of work to find fragments that explain the unsatisfiability, and send these back to the SAT solver as additional constraints (i.e., lemmas)
  - $A \wedge D \supset \neg E$
- Iterate to termination (e.g., B, D, E: $y = 5, y < x$: $y = 5, x = 6$)
- This is called "lemmas on demand" or "DPLL(T)"
- it works really well: yields effective SMT solvers

SMT Solvers

- SMT solvers are being honed by competition
- Various divisions (depending on the theories considered)
  - Equality and uninterpreted functions
  - Difference logic ($x - y < c$)
  - Full linear arithmetic
  - ... for integers as well as reals
  - Arrays
- Next competition at FLoC (Seattle, Summer 2006)
- SMT solvers enable infinite bounded model checking
  - And powerful backends to interactive theorem provers
  - And metric and temporal planning for AI
    (demonstrated by Martha Pollack et al using ARIO)

Example: Real Time

- Traditionally hard for automated analysis because continuous time excludes finite state methods
- Timed automata methods handle continuous time
  - But defeated by the case explosion when (discrete) faults are considered
- SMT solvers can handle both dimensions
  - Timeout automata, k-induction, disjunctive invariants
- E.g., Biphase Mark Protocol for asynchronous communic'n
  - Clocks at either end have different skew, rates, jitter
  - So have to encode a clock in the data stream
  - Used in CDs, Ethernet
  - Verify parameter values for reliable transmission

Real Time: Biphase Mark (ctd)

- First verified by human-guided proof in ACL2 by J Moore
- Three different verifications used PVS
    - One by Groote and Vaandrager used PVS + UPPAAL
    - Required 37 invariants, 4,000 proof steps, hours of prover time to check
- Brown and Pike recently did it with `sal-inf-bmc`
    - Three lemmas proved automatically with 1-induction,
    - Statement of theorem discovered systematically using disjunctive invariants (7 disjuncts)
    - Theorem proved automatically using 5-induction
    - Verification takes seconds to check
- Adapted verification to 8-N-1 protocol (used in UARTs)
    - Revealed a bug in published application note

Summary

- SAT can be extended to MaxSAT to deal with inconsistencies
  - e.g., to find best diagnosis, integrate learners
  - We have done this
- SMT can be extended similarly to MaxSMT
  - We are doing this
- SMT also can be extended to maximize any arithmetic expression, subject to constraints
  - We are doing this, too
- Anything a SAT solver can do
  - And anything a constraint solver can do

  An SMT solver can do better (we think)
- See ICS and its descendents at `fm.csl.sri.com`