BMC and UMC of transition systems using SAT and SMT

Rémi Delmas

3 février 2010
1 Solvers
   - SAT solvers
   - SMT solvers

2 Transition Systems

3 Bounded Model Checking

4 Unbounded Model Checking
   - Completeness Threshold
   - k-Induction
   - Interpolation

5 Abstraction Techniques
   - Counter Example Guided Driven Abstraction Refinement
   - Proof Driven Abstraction

6 Applications
Propositional Satisfiability Procedures

Given a propositional formula, the solver either returns:

- a model of the formula,
- an UNSAT proof (resolution tree).

Most effective procedures as of today:

- DPLL: unit propagation + heuristics driven branching + conflict analysis + backjumping,
- Stalmarck: equivalence classes + saturation + dilemma rule,
- Resolution: limited forms used as preprocessing.
In its simplest form: collaboration of a theory-specific solver $S(T)$ with a SAT solver. $S(T)$ usually solves *conjunctions* of constraints.

- Equalities, inequalities and predicates over theory-specific terms abstracted as boolean propositions,
- A SAT solver searches for SAT assignments to the boolean abstraction,
- Each time an assignment $A$ is found, $S(T)$ is called to find an extension of $A$ in the theory $T$,
  - If $S(T)$ answers SAT, solution found,
  - If $S(T)$ answers UNSAT, a boolean conflict clause is derived and added in the boolean abstraction. The SAT solver analyses the conflict and backjumps, and will never take the bad path again.
The SMT-lib project

SMT-lib : A standardised collection of theories and a language for expressing SMT problems.
http://combination.cs.uiowa.edu/smtlib/
Available theories :

- Uninterpreted functions with equality,
- Linear arithemtic over rationals and integers,
- Separation logic over rationals and integers (special case of lin arith),
- Bit precise integer arithmetic,
- Theory of arrays (read(a,i,v) and write(a,i) operations),
- LISP lists (car, cdr, ...),
- Some first order theories.

SMT-lib compliant solvers :
Alt-Ergo, Barcelogic, Beaver, Boolector, CVC3, DPT, MathSAT, OpenSMT, SatEEn, Spear, STP, UCLID, veriT, Yices, Z3.
Industrial Command/Control systems/programs are modelled in the following framework:

Transition System: \( TS = \{ V, D, I, T \} \)

- \( V = \{ x_1, \ldots, x_n \} \): state vector of the system,
- \( D = D_1 \times \cdots \times D_n \cap C \): domain of the state vector (explicit and/or modeled by constraints \( C \)),
- \( I : D \rightarrow \{ \top, \bot \} \): For \( s \in D \), \( I(s) = \top \) if \( s \) is an initial state, \( I(s) = \bot \) otherwise,
- \( T : D^2 \rightarrow \{ \top, \bot \} \): For \( (s, s') \in D^2 \), \( T(s, s') = \top \) iff \( s' \) is an acceptable successor of \( s \), \( \bot \) otherwise.
Unrolling a transition system yields an SMT formula, in which a fresh state variable \( s_i \) is introduced for each step, and the transition predicate is instanciated to model a trace of \( k \) transitions.

\[
\exists s_0, \ldots, s_k \in D^{k+1} \mid I(s_0) \land T(s_0, s_1) \land \cdots \land T(s_{k-1}, s_k)
\]

Satisfying assignments to this formula are exactly all possible execution traces of \( k \) transitions from initial states.

Using the following compact notation:

- \( I(s_k) \Rightarrow I_k \)
- \( T(s_i, s_{i+1}) \Rightarrow T_i \)

and dropping the explicit existential quantification, the formula can be written as:

\[
I_0 \land T_0 \land T_1 \land \cdots \land T_{k-1}
\]
Bounded Model Checking

BMC problem = find states which falsify some state predicate $P$ within $k$ transitions from initial states, by analysing the satisfiability of the following formula:

$$BMC(TS, P, k) \equiv I_0 \land T_0 \land \cdots \land T_{k-1} \land \neg (P_0 \land \cdots \land P_k)$$

Formula UNSAT proves that no state violating $P$ can be reached in $k$ transitions.

In an incremental fashion, repeatedly solve this formula for increasing values of $k$:

$$BMC(TS, P, k) \equiv I_0 \land T_0 \land \cdots \land T_{k-1} \land \neg P_k$$
UMC problem = produce a proof that no state falsifying $P$ can be reached in any number of transitions. As we will see, it is possible to obtain unbounded proofs by analysing bounded unrollings of transition systems.
If $D$ is finitely enumerable, there is a certain unrolling depth from which no new state can be been visited. It is called the *diameter* of the system.

Proving that $BMC(TS, P, k)$ is UNSAT up to the diameter is a proof that no reachable state falsifies $P$.

BDD based model checkers compute the diameter by maintaining the characteristic function of the reachable states, and looking for a fixed point, and blow up while doing so.

A safe upper bound to the diameter can be computed by unrolling and solving this formula until it becomes UNSAT

$$l_0 \land T_0 \land \cdots \land T_{k-1} \land \text{AllDiff}_{[0,k]}$$

Where $\text{AllDiff}_{[0,k]} = \top$ iff all $s_0, \ldots, s_k$ are pairwise distinct.
Induction Principle

Idea: show that the transition relation inductively preserves $P$, by checking these formulas for increasing values of $k$:

$$Base(TS, P, k) \equiv l_0 \land T_0 \land \cdots \land T_{k-1} \land \neg P_k$$

$$Step(TS, P, k) \equiv (P_0 \land \cdots \land P_k) \land (T_0 \land \cdots \land T_k) \land \neg P_{k+1} \land AllDiff_{[0,k+1]}$$

- If $Base$ is UNSAT and $Step$ is SAT, unroll Base and Step once more and analyse again.
- If $Base$ is UNSAT and $Step$ is UNSAT, we have proved that no reachable state falsifies $P$.
- At depth $CT$ at worst, either $Base$ becomes SAT (and $P$ is falsified), or $Step$ becomes UNSAT (the method is complete).
Relatively often, $P$ is true but the Step instance still remains SAT. One then has to strengthen $P$ with some lemma $L$ to make the Step instance UNSAT.

- $Base(TS, P \land L, k)$
- $Step(TS, P \land L, k)$

$L$ filters out unreachable states on which paths leading to $\neg P$ are rooted.

The best $L$ is the characteristic function of the set of reachable states.

In practice, finding the right lemmas is crucial to the success of k-induction.
Craig Interpolation Theorem for Propositional Logic

Given two formulas $A$, $B$ s.t.

$$A \land B \equiv UNSAT$$

A formula $Itp$ can be computed s.t. :

- $A \rightarrow Itp$,
- $Itp \land B \equiv UNSAT$,
- $\text{Vars}(Itp) \subseteq \text{Vars}(A) \cap \text{Vars}(B)$,

In propositional logic :

- interpolants can be extracted from resolution proofs,
- SAT solvers can export resolution proofs.
Assuming the following BMC formula is UNSAT:

\[ I_0 \land T_0 \land \cdots \land T_{j-1} \land T_j \land \cdots \land T_{k-1} \land \neg P_k \]

An interpolant \( Itp \) for the prefix \( A \) is such that:

- \( Itp \) holds for all states reachable in \( j \) transitions from an initial state (because \( Vars(Itp) = s_j \) and \( A \rightarrow Itp \)),

- \( P \) cannot be falsified within \( k - j - 1 \) transitions from any \( s_j \), (because \( Itp \land B \equiv \bot \)).

If \( Itp \) happens to inductively stable (i.e. is a lemma of the system), we have a proof that no reachable state falsifies \( P \), because where \( Itp \) holds, \( \neg P \) cannot be reached in less than some non-zero number of transitions.
Interpolation-based Proof Strategy

Build a state predicate $R$ by iterative widening using interpolants:

- Check $I_0 \land \neg P_0$, if SAT return falsifiable
- Initialise $R = I$, $k = 1$, choose some $0 \leq j \leq k$
- While($\top$), unroll and check satisfiability of formula:

$$
\begin{align*}
R_0 \land T_0 \land \cdots \land T_{j-1} \land T_j \land \cdots \land T_{j-1} \land \neg P_k
\end{align*}
$$

- SAT: if $R=I$, return falsifiable, else increment $k$ and move on to next iteration.
- UNSAT: Extract $ltp$. Let $R' = ltp(s_0/s_1)$
- Check satisfiability of $\neg(R' \rightarrow R)$:
  - SAT: move on to next iteration using $R = R \lor R'$.
  - UNSAT: $R$ is inductively stable, return valid.

In comparison, BDD based model checkers compute an $R$ which is the exact characteristic function of the set of reachable states. Here $R$ is abstract (small), and discards information not useful for proving $P$. 
Other theories besides propositional logic have interpolants:

- quantifier free linear rational/integer arithmetic with uninterpreted functions,
- theory of lisp structures (car, cdr, ...),
- FOL,

Generic methods for generating interpolants for combined theories from theory specific interpolants have been proposed. Interpolation is a really active research topic.
Clever transition relation modelling and abstraction are key to keeping satisfiability problems within manageable sizes and complexities.
Discard constraints from the transition relation (discards variables or turns functionally dependent variables into free variables),

Model check system (Base+Step),

In case of SAT Base, validate CEX by simulation,

If CEX is spurious, reintroduce violated constraints and solve again until:
  - a valid CEX is found,
  - the abstract Step becomes UNSAT,
  - all constraints have been reintroduced (then you can increment k and start abstracting again).
Proof Driven Abstraction

- Check Base instance up to some depth $k$ against $P$, produce UNSAT resolution proof,
- Identify constraints of the transition relation used in the proof, discard others,
- Generate Step instance from simplified transition relation,
- If Step fails, increment $k$ and iterate again (or reintroduce discarded constraints).
For BMC/UMC to be efficient, you need to have:

- A compact state vector $V$ (avoid redundant memories in the model),
- A compact domain $D$ (statically rule out unreachable values from the search space, provide lemmas which characterise reachable states),
- A transition relation which does a lot of things at once (avoid small step semantics which just modify 1 model variable per transition),

For stateless problems (static systems), the UMC boils down to BMC at depth 0.


Effective Preprocessing in SAT through Variable and Clause Elimination, Niklas Eén (Cadence Berkeley Laboratories), Armin Biere (Johannes Kepler University Linz), SAT 2005.
References: Unbounded Model Checking

- Checking Safety Properties Using Induction and a SAT-Solver, Mary Sheeran (Chalmers Univ.), Gunnar Stålmarck (Prover Technology AB), LNCS FMCAD00
- Applications of Craig Interpolants in Model Checking, Kenneth McMillan, (Cadence Berkeley Laboratories), TACAS05
Yices 1.0 : An Efficient SMT Solver : John Rushby SRI
SMTCOMP-06

A Combination Method for Generating Interpolants, Greta Yorsh,
(School of Comp. Sci., Tel Aviv Univ), Madanlal Musuvathi,
(Microsoft Research, Redmond), CADE-20 : automated deduction (Tallinn, 22-27 July 2005)