

25 November 2002, Berkeley

Bounded Model Checking: from Refutation to Verification

Harald Rueß

(joint work with Leonardo de Moura and Maria Sorea)

email: `ruess@csl.sri.com`

url: <http://www.csl.sri.com/~ruess>

Organization Computer Science Laboratory

SRI International

Menlo Park, CA

Outline

- Bounded Model Checking (BMC) on Infinite State Systems
- Lazy Theorem Proving
- BMC and k -induction
- Lazy invariant strengthening from failed induction proofs

Bounded Model Checking (BMC)

Given.

- Transition system $M = (I, T)$
- Temporal logic formula φ
- Natural number k

Problem.

Is there a counterexample of length k to the model checking problem $M \models \varphi$.

BMC for finite-state systems by reduction to SAT [Clarke et al, 99]

$$I(s_0) \wedge T(s_0, s_1) \wedge \dots \wedge T(s_{k-1}, s_k) \wedge \neg\varphi(s_0) \vee \dots \vee \neg\varphi(s_k)$$

Here. BMC for infinite-state systems

Example: Shift register

Three bit shift register x , with individual bits: $x[0], x[1], x[2]$.

Encoding.

$$I(x) = \text{true}$$

$$T(x, x') = (x'[0] = x[1]) \wedge (x'[1] = x[2]) \wedge (x'[2] = 1)$$

Hypothesis

$$\mathbf{F}(x = 0)$$

Check for an execution sequence of length $k = 2$ for

$$\mathbf{G}(x \neq 0)$$

Example (Cont.)

2-step unrolling of T

$$I(x_0) \wedge T(x_0, x_1) \wedge T(x_1, x_2)$$

Expansion of T and I

$true \wedge$

$$(x_1[0] = x_0[1]) \wedge (x_1[1] = x_0[2]) \wedge (x_1[2] = 1) \wedge \quad \text{1st step}$$

$$(x_2[0] = x_1[1]) \wedge (x_2[1] = x_1[2]) \wedge (x_2[2] = 1) \quad \text{2nd step}$$

Example (Cont.)

	L_0			
	x_0	x_1	x_2	
	$x_0[0]$	$x_1[0]$	$x_2[0]$	
	$x_0[1]$	$x_1[1]$	$x_2[1]$	L_2
	$x_0[2]$	$x_1[2]$	$x_2[2]$	
		L_1		

Loop from x_2 to x_i , $i = 0, 1, 2$

$$L_i = (x_i[0] = x_i[1]) \wedge (x_i[1] = x_i[2]) \wedge (x_i[2] = 1)$$

Invariant $\mathbf{G}(x \neq 0)$ must hold:

$$S_i = (x_i[0] = 1) \vee (x_i[1] = 1) \vee (x_i[2] = 1)$$

Example (Cont.)

Propositional formula.

$$I(x_0) \wedge T(x_0, x_1) \wedge T(x_1, x_2) \wedge \bigvee_{i=0}^2 L_i \wedge \bigvee_{i=0}^2 S_i$$

Experimental results.

bits	3	5	8	10
time	0s	1s	7s	55s
memory	1MB	2MB	5MB	11MB

(using Prover, timings as reported in [Clarke et al, 2001])

Example: Shift Register using Word-Level Encoding

Bitvectors.

- x, y interpreted over $B[n]$, the bitvectors of size n
- Operations $x[i : j]$, $x \circ y$

Encoding.

$$I(x) := true \quad T(x, y) := (y = x[1 : n - 1] \circ 1_1)$$

Hypothesis. $\mathbf{F}(x = 0_n)$

BMC problem for $k = 2$

$$(x_1 = x_0[1 : n - 1] \circ 1_1) \wedge (x_2 = x_1[1 : n - 1] \circ 1_1) \wedge \\ (x_0 \neq 0_n \vee x_1 \neq 0_n \vee x_2 \neq 0_n) \wedge (x_0 = x_2 \vee x_1 = x_2).$$

Shift register (Cont.)

Choose all unit literals to hold

Canonization.

$$\begin{aligned}x_2 &= x_1[1 : n - 1] \circ 1_1 \\ &= (x_0[1 : n - 1] \circ 1_1)[1 : n - 1] \circ 1_1 \\ &= x_0[2 : n - 1] \circ 1_2\end{aligned}$$

Case Split. ($x_0 = x_2$)

$$x_0 = x_0[2 : n - 1] \circ 1_2$$

Solving. (e.g.)

$$x_0 = 1_n$$

Example (Cont.)

The number of case splits is only linear in the bound k of the length on counterexamples

Runtime of bitvector decision procedure is (largely) independent of the size of the shift register.

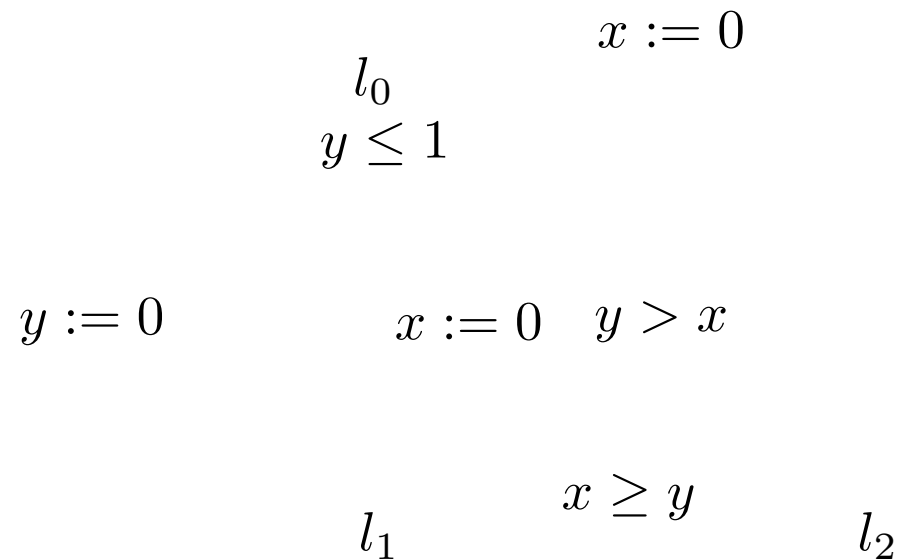
This differs from approaches based on bitwise splitting.

Solving word-level formulas with domain-specific theories instead of translating into the Boolean domain bears a high potential for optimization.

For the independence of the word size n of the shift register example, one may leave n uninterpreted.

In this way, our procedure invalidates a family of shift registers with almost no runtime penalties.

Example: Timed Automata



States

- State variables $V = \{at, x, y\}$ and $V' = \{at', x', y'\}$.
- at, at' interpreted over $\{l_0, l_1, l_2\}$, and x, y, x', y' over \mathbb{R}_0^+ .
- A **state** is just an assignment for these variables

Example (Cont.)

Initial state

$$I(at, x, y) := (at = l_0 \wedge x = 0 \wedge y = 0)$$

Transition relation

$$T(at, x, y, at', x', y') :=$$

$$(at = l_0 \wedge x' = 0 \wedge y' = y \wedge at' = l_0 \wedge y' \leq 1) \vee$$

$$(at = l_0 \wedge x' = 0 \wedge y' = y \wedge at' = l_1) \vee$$

$$(at = l_0 \wedge y > x \wedge x' = x \wedge y' = y \wedge at' = l_1) \vee$$

$$(at = l_1 \wedge y' = 0 \wedge x' = x \wedge at' = l_0) \vee$$

$$(at = l_1 \wedge x \geq y \wedge x' = x \wedge y' = y \wedge at' = l_2) \vee$$

$$D(at, x, y, at', x', y')$$

Example (Cont.)

Delay steps:

$$D(at, x, y, at', x', y') = \\ \exists \delta \geq 0. ((at = l_0 \Rightarrow y' \leq 1) \wedge at' = at \wedge \\ x' = x + \delta \wedge y' = y + \delta).$$

Quantifier elimination:

$$D(at, x, y, at', x', y') := \\ (at = l_0 \Rightarrow y' \leq 1) \wedge at' = at \wedge \\ x' - x \geq 0 \wedge y' - y = x' - x$$

Hypothesis. $G(at \neq l_2)$

Counterexample. $(l_0, 0, 0) \rightsquigarrow (l_1, 0, 0) \rightsquigarrow (l_2, 0, 0)$

BMC(C)

Let C be some constraint theory with decidable satisfiability problem.

$\text{Bool}(C)$ propositional combinations of atoms in C .

BMC(C) problems are given by

- Encoding of $M = (I, T)$ as a formula in $\text{Bool}(C)$,
- $\text{LTL}(C)$ formula φ , and
- bound k

Reduction of $\text{BMC}(C)$ to satisfiability in $\text{Bool}(C)$

1. $\llbracket M \rrbracket_k$ as the unfolding of the program M up to step k from initial states (requires k disjoint copies of V).
2. Translation of $\neg\varphi$ into a corresponding Büchi automaton $\mathcal{B}_{\neg\varphi}$ whose language of accepting words consists of the satisfying paths of $\neg\varphi$.
3. Encoding of the transition system for $\mathcal{B}_{\neg\varphi}$ and the Büchi acceptance condition as a Boolean formula, say $\llbracket \mathcal{B} \rrbracket_k$.
4. $\llbracket M, \varphi \rrbracket := \llbracket \mathcal{B} \rrbracket_k \wedge \llbracket M \rrbracket_k$.

A satisfying assignment for the formula $\llbracket M, \varphi \rrbracket$ induces a counterexample of length k

Results

Soundness. If there is a k s.t. $\llbracket M, \varphi \rrbracket_k$ is satisfiable, then $M \not\models \varphi$

Completeness. $M \not\models \varphi$ implies there exists k s.t. $\llbracket M, \varphi \rrbracket_k$ is satisfiable, given that

- M has a finite set of reachable states, **OR**
- φ is a safety property, **OR**
- M is a timed automata, and φ is in LTL(CC), where CC are clock constraints.

Incompleteness of BMC(C)

In general: BMC over infinite domains is not complete.

$$M = \langle I, T \rangle, V = \{x\}, I = (x = 0), T = (x' = x + 1)$$

$$\varphi = \mathbf{F}(x < 0)$$

$$x = x + 1$$

$$x \geq 0$$

$$M \not\models \varphi$$

$$l_0$$

$$q_0$$

There exists no $k \in \mathbb{N}$, such that $\llbracket M, \varphi \rrbracket_k$ is satisfiable.

\mathcal{B}_k must contain a finite accepting cycle

$$\bigvee_{j=0}^{k-1} (x[k] = x[j])$$

Such a cycle does not exist; $x[i + 1] = x[i] + 1$, for all $i \geq 0$.

Satisfiability for Boolean Constraint Formulas

Given. Propositional formula with constraints as literals.

Assumption. *CSAT* decides the satisfiability problem for conjunctions of constraints.

Problem. Efficient satisfiability for Boolean combinations of constraints.

Experiment. Using a combination of BDDs with linear arithmetic decision procedures in PVS2.4 for finding counterexamples in a modified train-gate controller example

$k = 2$	70s
$k = 3$	8500s

~> new techniques required

Reduction to SAT

Let φ be a Boolean constraint formulas with constraints c_i .

Translations.

- α replaces constraints with (fresh) propositional variables.
- γ substitutes constraints for corresponding variables.

Example.

$$\alpha(x = y \wedge f(x) = f(y)) \rightsquigarrow p \wedge q$$

A Boolean assignment ν induces a set of constraints $\gamma(\nu)$

$$\text{if } \nu = [p \mapsto 0, q \mapsto 1] \text{ then } \gamma(\nu) = \{x \neq y, f(x) = f(y)\}$$

Boolean Reduction Theorem.

Inconsistencies. ($l_i \in Lits(\varphi)$)

$$\{l_1, \dots, l_n\} \in I(\varphi)$$

iff

$$\gamma(l_1) \wedge \dots \wedge \gamma(l_n) \text{ is } C\text{-inconsistent}$$

Theorem.

- $\varphi \in \text{Bool}(C)$ and
- $\alpha(\varphi) \wedge (\bigwedge_{\{l_1, \dots, l_n\} \in I(\varphi)} (\neg l_1 \vee \dots \vee \neg l_n))$

are equisatisfiable

Usually, an exponential number of inconsistency tests is required.

Are there “good” enumeration strategies?

Lazy Theorem Proving

$p := \alpha(\varphi);$

loop

$\nu := SAT(p);$

if $\nu = \perp$ **then return** \perp

if $\gamma(\nu) \neq \perp$ **then return** $\gamma(\nu)$

else $I := \bigvee_{c \in \gamma(\nu)} \neg \alpha(c); p := p \wedge I$

endloop;

return \perp

Optimizations. don't cares, convergence acceleration

Acceleration of convergence

Problem. Find concise overapproximations of the minimal set of inconsistent constraints C .

Trade-off between conciseness of the approximation and the cost for computing it.

Approaches.

- Constructing a dependency graph while running $CSAT$.
- Rerun $CSAT$ on selected subsets and test for satisfiability.

Here. use both syntactic and semantic information for computing approximations of minimal inconsistent sets.

Experiment: Bakery protocol (Cont.)

(using ICSAT = ICS + SAT (ics.csl.sri.com))

depth	no explain		explain	
	conflicts	ICS	conflicts	ICS
5	24	162	7	71
6	48	348	7	83
7	96	744	7	94
8	420	3426	29	461
9	936	7936	70	1205
10	2008	17567	85	1543
15	-	-	530	13468

More Experiments

System Name	$k = 5$	$k = 10$	$k = 15$	$k = 20$
Bakery Protocol	0.01s	0.18s	0.641s	1.6s
Simpson Protocol	0.039s	0.271s	1.635s	18.05s
Train Gate Controller	0.059s	0.38s	2.77s	17.93s
Fischer Protocol	0.033s	1.066s	1.17s	1.256s
Water Level Monitor	0.031s	0.047s	0.057s	0.084s
Evader-Pursuer	0.74s	0.78s	0.88s	0.99s
Leaking Gas Burner	0.035s	0.117s	0.35s	0.85s
Multi Rate Fischer	0.033s	1.98s	2.09s	2.20s

BMC for Proving Invariants

$M \models \Box\varphi$ if φ holds for all reachable states of M .

BMC complete if an upper bound on the length of counterexamples, the system's **diameter**, is known.

d is larger than the diameter of a system if

$$I(s_0) \wedge \pi^A(s_0, \dots, s_d)$$

is unsatisfiable, where

$$\begin{aligned}\pi(s_0, s_1, \dots, s_n) &:= \bigwedge_{0 \leq i < n} T(s_i, s_{i+1}) \\ \pi^A(s_0, s_1, \dots, s_n) &:= \pi(s_0, s_1, \dots, s_n) \wedge \bigwedge_{0 \leq i < j \leq n} s_i \neq s_j.\end{aligned}$$

Improvements

- $I(s_0) \wedge (\bigwedge_{i=1}^d \neg I(s_i)) \wedge \pi^A(s_0, \dots, s_d)$
- $I(s_0) \wedge \pi^{min}(s_0, \dots, s_d)$, where

$$\begin{aligned} \text{conn}_i(s_0, s_i) &:= \exists s_1, \dots, s_{i-1}. \pi(s_0, s_1, \dots, s_i) \\ \pi^{min}(s_0, \dots, s_d) &:= \pi(s_0, \dots, s_d) \wedge \bigwedge_{i=0}^{d-1} \neg \text{conn}_i(s_0, s_d) \end{aligned}$$

k -Induction

Base $I(s_0) \wedge \pi(s_0, \dots, s_{k-1}) \rightarrow \varphi(s_0) \wedge \dots \wedge \varphi(s_{k-1})$

Step $\forall n . \varphi(s_n) \wedge \dots \wedge \varphi(s_{n+k}) \wedge \pi(s_k, \dots, s_{n+k}) \rightarrow \varphi(s_{n+k+1})$

This rule is incomplete even for finite-state systems



Bad state s_4

Counterexamples $s_3 \rightsquigarrow s_3 \rightsquigarrow \dots \rightsquigarrow s_3 \rightsquigarrow s_4$
 $\underbrace{\hspace{10em}}_k$

k -Induction on acyclic paths

$\mathbf{IND}^A(k)$ is k -induction restricted to acyclic paths.

$$\pi^A(s_0, s_1, \dots, s_n) := \pi(s_0, s_1, \dots, s_n) \wedge \bigwedge_{0 \leq i < j \leq n} s_i \neq s_j.$$

For finite systems M

$$M \models \Box \varphi \text{ iff } \exists k. \mathbf{IND}^A(k)$$

Invariant Strengthening

Failed k -induction for φ yields

Explicit counterexample

$$s_n \rightsquigarrow s_{n+1} \rightsquigarrow s_{n+k} \rightsquigarrow s_{n+k+1}$$

Assume s_n to be unreachable and strengthen $\varphi \wedge \neg(s_n)$

Symbolic counterexample described by a conjunction of constraints

$$P(s_n, s_{n+1}, \dots, s_{n+k}, s_{n+k+1})$$

- Projection

$$Q(s_n) := \exists s_{n+1}, \dots, s_{n+k+1}. P(s_n, s_{n+1}, \dots, s_{n+k}, s_{n+k+1})$$

- Strengthening $\varphi \wedge \neg Q$

Example: Bakery protocol

Transition System. (initially $y_1, y_2 \geq 0$)

$$a_1 \quad y'_1 := y_2 + 1 \quad a_2 \quad y_2 = 0 \vee y_1 \leq y_2 \quad a_3$$

||

$$y'_1 := 0$$

$$b_1 \quad y'_2 := y_1 + 1 \quad b_2 \quad y_1 = 0 \vee \neg(y_1 \leq y_2) \quad b_3$$

$$y'_2 := 0$$

Mutual Exclusion. $MX = \mathbf{G} (\neg(pc_1 = a_3 \wedge pc_2 = b_3))$

Bakery: Proof by Induction

1-Induction.

$$MX(s_n) \wedge T(s_n, s_{n+1}) \Rightarrow MX(s_{n+1})$$

1st counterexample

$$\begin{aligned} at_1[n] = a_2 \wedge at_2[n] = b_3 \wedge y_2[n] = 0 \wedge at_1[n+1] = a_3 \wedge \\ at_2[n+1] = b_3 \wedge y_1[n+1] = y_1[n] \wedge y_2[n+1] = y_2[n] \end{aligned}$$

Projection of the “ $n + 1$ ” state yields

$$C_1 := at_1[n] = a_2 \wedge at_2[n] = b_3 \wedge y_2[n] = 0$$

Bakery (Cont.)

2nd counterexample

$$at_1[n] = a_2 \wedge at_2[n] = b_3 \wedge y_1[n] \leq y_2[n] \wedge at_1[n+1] = a_3 \wedge \\ at_2[n+1] = b_3 \wedge y_1[n+1] = y_1[n] \wedge y_2[n+1] = y_2[n]$$

Projecting “ $n + 1$ ” variables yields

$$C_2 := at_1[n] = a_2 \wedge at_2[n] = b_3 \wedge y_1[n] \leq y_2[n]$$

Strengthened property

$$MX \wedge \neg C_1 \wedge \neg C_2$$

Is proven with 1-induction

Experiments with k -Induction

System Name	Proved with k	Time	Auxiliary Invariants
Bakery Protocol	3	0.195s	<i>interval analyzer</i>
Simpson Protocol	2	0.162s	<i>automatic strength.</i>
Train Gate Controller	5	0.57s	<i>no</i>
Fischer Protocol	4	0.713s	<i>no</i>
Water Level Monitor	1	0.082s	<i>no</i>
Pursuit Game	<i>not proved</i>	-	-
Leaking Gas Burner	6	15.38s	<i>interval analyzer</i>
Multi Rate Fischer	4	0.848	<i>no</i>

Related Work

East coast. *Eager* encoding of all possible instances of lemmas needed for a solution.

- Variations of Ackermann's trick for EUF [Shostak'77], [Goel et al.; 98],[Bryant, Velev; 90], [Pnueli, Shtrichman; 99], . . .
- Extension to fragments of arithmetic such as difference logic [Shtrichman, 02].

West coast. *Lazy* encodings

- [Barrett, Dill, Stump; 02]
- [Flanagan, Joshi, Saxe; ??]

Lazy ideas have also been investigated in AI Planning, Tableau search, . . .

Induction for propositional BMC has first been used by [Sheeran et al, 2000].

Conclusions

Satisfiability of Boolean constraint formulas by lazy reduction to SAT. (similar to abstraction refinement)

Lazy integration works uniformly for a large class of constraint theories.

Main obstacles for efficiency: identification of minimal inconsistent sets of constraints.

Speedups of a couple of order of magnitude over “traditional” theorem-proving methods. \rightsquigarrow deeper explorations of state-spaces using BMC.

Verification based on k -induction and BMC

Lazy generation of invariant strengthenings from failed induction proofs.

Altogether

. . . the ingredients we are using are fairly conventional. What seems to be novel is the effective use of a SAT solver to point us to the portion of the constraints that is actually relevant to obtain the solution.