System Specification, Verification and Synthesis (SSVS) – CS 4830/7485, Fall 2019

18: Formal Verification: Bounded Model Checking

Stavros Tripakis

Northeastern University
Khoury College of Computer Sciences

FINITE-HORIZON REACHABILITY
(a.k.a. BOUNDED MODEL-CHECKING)

Bounded reachability

Question:

> *Can a "bad" state be reached in up to n steps (transitions)?*

i.e., given a transition system (P, S, S_0, L, R) and a set of states $Bad \subseteq S$, does there exist a path

$$s_0 \longrightarrow s_1 \longrightarrow \cdots \longrightarrow s_k$$

in the transition system such that $s_0 \in S_0$ and $s_k \in Bad$, and $k \leq n$.

Bounded reachability

Question:

Can a "bad" state be reached in up to n steps (transitions)?

i.e., given a transition system (P, S, S_0, L, R) and a set of states $Bad \subseteq S$, does there exist a path

$$s_0 \longrightarrow s_1 \longrightarrow \cdots \longrightarrow s_k$$

in the transition system such that $s_0 \in S_0$ and $s_k \in Bad$, and $k \leq n$.

Key idea:

Reduce the above question to a SAT (satisfiability) problem.

- SAT problem NP-complete for propositional logic.
- In practice, today's SAT solvers can handle formulas with thousands of variables (or more!): see [Malik and Zhang, 2009].
- BMC (**bounded model-checking**) has emerged thanks to the advances in SAT solver technology.

Bounded reachability

Suppose I have predicates $Init(\vec{x})$, $Trans(\vec{x}, \vec{x}')$, and $Bad(\vec{x})$.

How to use them for bounded reachability?

- Bad state reachable in 0 steps iff

$$\text{SAT}(\qquad\qquad\qquad)$$

Bounded reachability

Suppose I have predicates $Init(\vec{x})$, $Trans(\vec{x}, \vec{x}')$, and $Bad(\vec{x})$.

How to use them for bounded reachability?

- Bad state reachable in 0 steps iff

$$\mathsf{SAT}\big(Init(\vec{x}) \wedge Bad(\vec{x})\big)$$

Bounded reachability

Suppose I have predicates $Init(\vec{x})$, $Trans(\vec{x}, \vec{x}')$, and $Bad(\vec{x})$.

How to use them for bounded reachability?

- Bad state reachable in 0 steps iff

$$\mathsf{SAT}\big(Init(\vec{x}) \wedge Bad(\vec{x})\big)$$

- Bad state reachable in 1 step iff

$$\mathsf{SAT}\big(\qquad\qquad\qquad\qquad \big)$$

Bounded reachability

Suppose I have predicates $Init(\vec{x})$, $Trans(\vec{x}, \vec{x}')$, and $Bad(\vec{x})$.

How to use them for bounded reachability?

- Bad state reachable in 0 steps iff

$$\mathsf{SAT}\big(Init(\vec{x}) \wedge Bad(\vec{x})\big)$$

- Bad state reachable in 1 step iff

$$\mathsf{SAT}\big(Init(\vec{x}_0) \wedge Trans(\vec{x}_0, \vec{x}_1) \wedge Bad(\vec{x}_1)\big)$$

Bounded reachability

Suppose I have predicates $\mathit{Init}(\vec{x})$, $\mathit{Trans}(\vec{x}, \vec{x}')$, and $\mathit{Bad}(\vec{x})$.

How to use them for bounded reachability?

- Bad state reachable in 0 steps iff

$$\mathsf{SAT}\big(\mathit{Init}(\vec{x}) \wedge \mathit{Bad}(\vec{x})\big)$$

- Bad state reachable in 1 step iff

$$\mathsf{SAT}\big(\mathit{Init}(\vec{x}_0) \wedge \mathit{Trans}(\vec{x}_0, \vec{x}_1) \wedge \mathit{Bad}(\vec{x}_1)\big)$$

- ...
- Bad state reachable in n steps iff

$$\mathsf{SAT}\big(\qquad\qquad\qquad\qquad \big)$$

Bounded reachability

Suppose I have predicates $\mathit{Init}(\vec{x})$, $\mathit{Trans}(\vec{x}, \vec{x}')$, and $\mathit{Bad}(\vec{x})$.

How to use them for bounded reachability?

- Bad state reachable in 0 steps iff

$$\mathsf{SAT}\big(\mathit{Init}(\vec{x}) \wedge \mathit{Bad}(\vec{x})\big)$$

- Bad state reachable in 1 step iff

$$\mathsf{SAT}\big(\mathit{Init}(\vec{x}_0) \wedge \mathit{Trans}(\vec{x}_0, \vec{x}_1) \wedge \mathit{Bad}(\vec{x}_1)\big)$$

- ...
- Bad state reachable in n steps iff

$$\mathsf{SAT}\big(\mathit{Init}(\vec{x}_0) \wedge \mathit{Trans}(\vec{x}_0, \vec{x}_1) \wedge \cdots \wedge \mathit{Trans}(\vec{x}_{n-1}, \vec{x}_n) \wedge \mathit{Bad}(\vec{x}_n)\big)$$

Bounded reachability algorithm – outer loop

for all $k = 0, 1, ..., n$ **do**
  $\phi := Init(\vec{x}_0) \wedge Trans(\vec{x}_0, \vec{x}_1) \wedge \cdots \wedge Trans(\vec{x}_{k-1}, \vec{x}_k) \wedge Bad(\vec{x}_k)$;
  if SAT(ϕ) **then**
    print "Bad state reachable in k steps";
    output solution as counter-example;
  end if
end for
print "Bad state unreachable up to n steps";

Bounded reachability: soundness and completeness

for all $k = 0, 1, ..., n$ **do**
    $\phi := Init(\vec{x}_0) \wedge Trans(\vec{x}_0, \vec{x}_1) \wedge \cdots \wedge Trans(\vec{x}_{k-1}, \vec{x}_k) \wedge Bad(\vec{x}_k)$;
    if SAT(ϕ) **then**
        print "Bad state reachable in k steps";
        output solution as counter-example;
    end if
end for
print "Bad state unreachable up to n steps";

BMC algorithm is **sound** in the following sense:

- if algorithm reports "reachable" then indeed a bad state is reachable
- if algorithm reports "unreachable up to n steps" then there is no path of length $\leq n$ that reaches a bad state.

Bounded reachability: soundness and completeness

for all $k = 0, 1, ..., n$ **do**
  $\phi := Init(\vec{x}_0) \wedge Trans(\vec{x}_0, \vec{x}_1) \wedge \cdots \wedge Trans(\vec{x}_{k-1}, \vec{x}_k) \wedge Bad(\vec{x}_k)$;
  if SAT(ϕ) **then**
    print "Bad state reachable in k steps";
    output solution as counter-example;
  end if
end for
print "Bad state unreachable up to n steps";

BMC algorithm is **sound** in the following sense:

- if algorithm reports "reachable" then indeed a bad state is reachable
- if algorithm reports "unreachable up to n steps" then there is no path of length $\leq n$ that reaches a bad state.

Can we make BMC **complete**?

- It should report unreachable iff there are no reachable bad states (w.r.t. any bound).
- Is this even possible in general?

Bounded reachability: soundness and completeness

```
for all k = 0, 1, ..., n do
    φ := Init(x⃗_0) ∧ Trans(x⃗_0, x⃗_1) ∧ ··· ∧ Trans(x⃗_{k−1}, x⃗_k) ∧ Bad(x⃗_k);
    if SAT(φ) then
        print "Bad state reachable in k steps";
        output solution as counter-example;
    end if
end for
print "Bad state unreachable up to n steps";
```

BMC algorithm is **sound** in the following sense:

- if algorithm reports "reachable" then indeed a bad state is reachable
- if algorithm reports "unreachable up to n steps" then there is no path of length $\leq n$ that reaches a bad state.

Can we make BMC **complete**?

- It should report unreachable iff there are no reachable bad states (w.r.t. any bound).
- Is this even possible in general? For finite-state systems?

Bounded reachability: soundness and completeness

for all $k = 0, 1, ..., n$ **do**
  $\phi := Init(\vec{x}_0) \wedge Trans(\vec{x}_0, \vec{x}_1) \wedge \cdots \wedge Trans(\vec{x}_{k-1}, \vec{x}_k) \wedge Bad(\vec{x}_k)$;
  if SAT(ϕ) **then**
    print "Bad state reachable in k steps";
    output solution as counter-example;
  end if
end for
print "Bad state unreachable up to n steps";

BMC algorithm is **sound** in the following sense:

- if algorithm reports "reachable" then indeed a bad state is reachable
- if algorithm reports "unreachable up to n steps" then there is no path of length $\leq n$ that reaches a bad state.

Can we make BMC **complete**?

- It should report unreachable iff there are no reachable bad states (w.r.t. any bound).
- Is this even possible in general? For finite-state systems? Yes!

Complete BMC: “brute-force” threshold

```
for all k = 0, 1, ..., n do
    φ := Init(x⃗_0) ∧ Trans(x⃗_0, x⃗_1) ∧ ··· ∧ Trans(x⃗_{k−1}, x⃗_k) ∧ Bad(x⃗_k);
    if SAT(φ) then
        print "Bad state reachable in k steps";
        output solution as counter-example;
    end if
end for
print "Bad state unreachable up to n steps";
```

A finite-state transition system is essentially a finite graph.

How can we turn BMC into a complete method for finite-state systems?

If we know $|S|$ (the number of all possible states) then we can set $n := |S|$. Because no acyclic path can have length greater than $|S|$, and we only care about acyclic paths.

Complete BMC: “brute-force” threshold

for all $k = 0, 1, ..., n$ **do**
  $\phi := Init(\vec{x}_0) \wedge Trans(\vec{x}_0, \vec{x}_1) \wedge \cdots \wedge Trans(\vec{x}_{k-1}, \vec{x}_k) \wedge Bad(\vec{x}_k)$;
  if SAT(ϕ) **then**
    print “Bad state reachable in k steps”;
    output solution as counter-example;
  end if
end for
print “Bad state unreachable up to n steps”;

A finite-state transition system is essentially a finite graph.

How can we turn BMC into a complete method for finite-state systems?

If we know $|S|$ (the number of all possible states) then we can set $n := |S|$. Because no acyclic path can have length greater than $|S|$, and we only care about acyclic paths.

But: with 100 boolean variables, $|S| = 2^{100}$, so this isn't practical ... (formulas become too big).

Complete BMC: a better threshold

Reachability diameter: number of steps that it takes to reach any reachable state.

$$d := \min\{i \mid \forall s \in \textsf{Reach} : \exists \textsf{ path } s_0, s_1, ..., s_j : j \leq i \wedge s_0 \in S_0 \wedge s_j = s\}$$

where Reach is the set of reachable states.

Complete BMC: a better threshold

Reachability diameter: number of steps that it takes to reach any reachable state.

$$d := \min\{i \mid \forall s \in \text{Reach} : \exists \text{ path } s_0, s_1, ..., s_j : j \leq i \wedge s_0 \in S_0 \wedge s_j = s\}$$

where Reach is the set of reachable states.

d is generally a much better threshold than $|S|$. Why?

Complete BMC: a better threshold

Reachability diameter: number of steps that it takes to reach any reachable state.

$$d := \min\{i \mid \forall s \in \text{Reach} : \exists \text{ path } s_0, s_1, ..., s_j : j \leq i \wedge s_0 \in S_0 \wedge s_j = s\}$$

where Reach is the set of reachable states.

d is generally a much better threshold than $|S|$. Why?
$d \leq |\text{Reach}| \leq |S|$.

Complete BMC: a better threshold

Reachability diameter: number of steps that it takes to reach any reachable state.

$$d := \min\{i \mid \forall s \in \text{Reach} : \exists \text{ path } s_0, s_1, ..., s_j : j \leq i \wedge s_0 \in S_0 \wedge s_j = s\}$$

where Reach is the set of reachable states.

d is generally a much better threshold than $|S|$. Why?
$d \leq |\text{Reach}| \leq |S|$.

Problem: we don't know $|\text{Reach}|$, therefore how to compute d?

Complete BMC: the Completeness Threshold

Recurrence diameter : length of the longest cycle-free path.

$$r := \max\{i \mid \exists \text{ path } s_0, s_1, ..., s_i : s_0 \in S_0 \land \forall 0 \leq j < k \leq i : s_j \neq s_k\}$$

Complete BMC: the Completeness Threshold

Recurrence diameter : length of the longest cycle-free path.

$$r := \max\{i \mid \exists \text{ path } s_0, s_1, ..., s_i : s_0 \in S_0 \wedge \forall 0 \leq j < k \leq i : s_j \neq s_k\}$$

Claim: $d \leq r$. Why?

Complete BMC: the Completeness Threshold

Recurrence diameter : length of the longest cycle-free path.

$$r := \max\{i \mid \exists \text{ path } s_0, s_1, ..., s_i : s_0 \in S_0 \wedge \forall 0 \leq j < k \leq i : s_j \neq s_k\}$$

Claim: $d \leq r$. Why?
Because in the definition of d only acyclic paths matter.

$\Rightarrow$ using r instead of d is safe. Why?

Complete BMC: the Completeness Threshold

Recurrence diameter : length of the longest cycle-free path.

$$r := \max\{i \mid \exists \text{ path } s_0, s_1, ..., s_i : s_0 \in S_0 \wedge \forall 0 \le j < k \le i : s_j \ne s_k\}$$

Claim: $d \le r$. Why?
Because in the definition of d only acyclic paths matter.

$\Rightarrow$ using r instead of d is safe. Why?
Because r is an upper bound for d, so we are being conservative.

Can we compute r? How?

Complete BMC: the Completeness Threshold

Recurrence diameter : length of the longest cycle-free path.

$$r := \max\{i \mid \exists \text{ path } s_0, s_1, ..., s_i : s_0 \in S_0 \land \forall 0 \leq j < k \leq i : s_j \neq s_k\}$$

Claim: $d \leq r$. Why?
Because in the definition of d only acyclic paths matter.

$\Rightarrow$ using r instead of d is safe. Why?
Because r is an upper bound for d, so we are being conservative.

Can we compute r? How?

Use a SAT solver!

$$r := \max\{i \mid \text{SAT}\Big(\mathit{Init}(\vec{x}_0) \land \mathit{Trans}(\vec{x}_0, \vec{x}_1) \land \cdots \land \mathit{Trans}(\vec{x}_{i-1}, \vec{x}_i) \land \bigwedge_{j=0}^{i-1} \bigwedge_{k=j+1}^{i} \vec{x}_j \neq \vec{x}_k \Big)\}$$

Bibliography

Biere, A., Cimatti, A., Clarke, E. M., Strichman, O., and Zhu, Y. (2003).
Bounded model checking.
Advances in Computers, 58:117–148.

Latvala, T., Biere, A., Heljanko, K., and Junttila, T. (2004).
Simple Bounded LTL Model Checking.
In *Formal Methods in Computer-Aided Design*, volume 3312 of *LNCS*, pages 186–200. Springer.

Malik, S. and Zhang, L. (2009).
Boolean satisfiability: From theoretical hardness to practical success.
Communications of the ACM, 52(8):76–82.