Objective

Consider model checking as a bug-hunting technique, not as verification engine.

The aim of bounded model checking is to find counterexamples of a certain length $k$. 
Inventors of Bounded Model Checking

- Armin Biere (AT)
- Alessandro Cimatti (I)
- Edmund Clarke Jr. (USA)
- Yunshan Zu

Recall ROBDD-Based Symbolic Model Checking

- ROBDDs are a canonical form for representing switching functions for a given variable ordering.
- Represent transition relation and sets of states as switching functions characteristic functions, also of the set of reachable states.
- CTL model checking := formula manipulation of switching functions existential variable elimination ($\exists\varnothing$), disjunction, conjunction, negation.
- Use ROBDDs as (often) compact data structure for switching functions.

A Simple Example

Deficiencies BDD-Based Symbolic Model Checking

- ROBDDs are canonical, but often can still become too large.
- The size of ROBDDs is highly sensitive to the variable ordering.
  - finding the optimal variable ordering is NP-complete
  - for some functions, no space-efficient variable ordering does exist.
- Alternative symbolic approach: manipulate switching functions (as is).
  - Cast bug hunting $TS \not\models \varphi$ as Boolean satisfiability problem
    - for $\varphi \in \text{LTL}$, $TS \not\models \varphi$ iff $TS \models \exists\neg\varphi$
    - look for finite paths to conclude $TS \models \exists\neg\varphi$.
Using SAT Solvers

- SAT procedures work on switching functions without canonical form
- Do not suffer from potential state explosion of ROBDDs
- Different variable orderings are possible on different branches (= clauses)
- SAT is NP-complete, but there exist very efficient SAT solvers
  - handling hundreds or thousands of variables and millions of clauses
- SAT has no possibility for variable elimination (no $\exists$)
  - focus on falsification rather than verification

Bounded Model Checking: Idea

- Bounded model checking = SAT-based symbolic model checking
- as BMC focuses on finding counterexamples, it is mostly used for LTL
- counterexamples for LTL are simpler than for CTL
- Represent transition relation and sets of states as switching functions
- Unroll the transition relation up to certain fixed bound $k$, say
- Search for counterexamples of $\varphi$ (aka: witnesses of $\neg \varphi$) of length $k$
  1. if $\pi \models \Diamond a$ implies $\pi \not\models \varphi$, and
  2. if $\pi$ is a lasso and $\pi \models \Box^k a$, then $\pi \not\models \Box a$
- Transform this search into a SAT problem and exploit a SAT solver

1. Compare the counterexample of $\Box a$ and of $\exists \forall \Diamond \Box a$.

A Simple Example

- The same example as before but not tackled using BMC.
Bounded LTL Semantics

Lassos

If a path satisfies LTL-formula $\varphi$ on a bounded semantics, then it satisfies $\varphi$ using the standard LTL semantics over infinite paths.

This is due to the fact that some infinite paths can be represented by a finite prefix with a loop, a lasso.

Definition: $(k, \ell)$-lasso

For $k, \ell \in \mathbb{N}$, the infinite path $\pi$ is a $(k, \ell)$-lasso if and only if for all $j \in \mathbb{N}$:

$$\pi[k+1+j] = \pi[\ell+j].$$

That is, $\pi = s_0 \ldots s_{\ell-1} \cdot (s_k \ldots s_k)^\omega$. -lasso

Example

Joost-Pieter Katoen Lecture#20 13/35

Bounded LTL Semantics

Bounded LTL Semantics for Lassos

Consider the first $k+1$ states of an infinite path. Let $i \in \{0, \ldots, k\}$.

The bounded satisfaction relation $\models_k$ is defined as follows.

If $\pi$ is an $(k, \ell)$-lasso, then $\pi^{k+1+j} = \pi^{i+j}$ for all $j$, and:

$$\pi^i \models_k \Box \varphi \quad \text{iff} \quad \pi^{i+1} \models_k \varphi \quad \text{if} \quad i < k$$

$$\pi^i \models_k \Diamond \varphi \quad \text{iff} \quad \pi^i \models_k \varphi \quad \text{for some} \quad j \in \{\min(i, \ell), \ldots, k\}$$

$$\pi^i \models_k \lozenge \varphi \quad \text{iff} \quad \pi^i \models_k \varphi \quad \text{for all} \quad j \in \{\min(i, \ell), \ldots, k\}$$

LTL Semantics Rephrased

Let $\varphi$ be an LTL-formula (without until, release) in positive normal form. Let $\pi^i = \pi[i\ldots]$ be the suffix of $\pi$ starting from position $i$. The LTL semantics of $\varphi$ on (suffix) path $\pi^i$ for $i \in \mathbb{N}$ is defined by:

$$\pi^i \models a \quad \text{iff} \quad a \in L(\pi[i])$$

$$\pi^i \models \neg a \quad \text{iff} \quad a \notin L(\pi[i])$$

$$\pi^i \models \varphi \land \psi \quad \text{iff} \quad \pi^i \models \varphi \text{ and } \pi^i \models \psi$$

$$\pi^i \models \varphi \lor \psi \quad \text{iff} \quad \pi^i \models \varphi \text{ or } \pi^i \models \psi$$

$$\pi^i \models \Diamond \varphi \quad \text{iff} \quad \pi^{i+j} \models \varphi \text{ for some } j \geq i$$

$$\pi^i \models \lozenge \varphi \quad \text{iff} \quad \pi^{i+j} \models \varphi \text{ for all } j \geq i.$$
Some Properties

For every $\varphi \in \text{LTL}$ and $k \in \mathbb{N}$: $\pi \models_k \varphi$ implies $\pi \models \varphi$.

If $TS \models \exists \varphi$, then $TS \models_k \exists \varphi$ for some $k$.

For any finite $TS$, it holds: $TS \models \exists \varphi$ iff $TS \models_k \exists \varphi$.

Bounded Model Checking: Basic Scheme

**Procedure**

1. Generate a propositional logic formula $\Phi$ from transition system $TS$, LTL formula $\varphi$, and unrolling depth $k$ such that $\Phi$ is satisfiable iff $TS \models_k \varphi$

2. Translate the formula $\Phi$ into CNF-formula $\Psi$ using the Tseitin transformation

3. Solve the CNF-formula $\Psi$
   
   3.1 $\Psi$ satisfiable? $\Rightarrow$ $TS \models_k \varphi$ $\Rightarrow$ counterexample/witness
   
   3.2 $\Psi$ not satisfiable? $\Rightarrow$ $TS \not\models_k \varphi$ $\Rightarrow$ unknown

Repeat step 1.–3. with increased $k$ until either a counterexample is found, or some stopping criterion (e.g., maximal depth $k_{\text{max}}$) is reached.
Translating BMC into SAT

For transition system $TS, \varphi \in LTL$ and $k \in \mathbb{N}$, construct the propositional logic formula $[TS, \varphi]_k$ satisfying

$[TS, \varphi]_k$ is satisfiable if and only if $\pi = s_0 \ldots s_k \vdash \varphi$ for some $\pi \in \text{Paths}(TS)$.

The formula $[TS, \varphi]_k$ is obtained in three steps:

1. Encode paths $s_0 \ldots s_k$ in $TS$ of length $k$; this yields formula $[TS]_k$

2. Define an auxiliary propositional formula $loop_k$ which is true if there is a backward edge from $s_k$ to some “earlier” state $s_i, i \leq k$

3. Encode an LTL-formula as propositional formula $\betaskip$

Encoding Bounded LTL as SAT

The size of the propositional formula $[\varphi]_k$ is linear in $|\varphi|$ and at least cubic in $k$.

Proof.

The variables $i$ and $\ell$ range over $\{0, \ldots, k\}$. This yields $O(k^3)$ combinations. This holds for any sub-formula of $\varphi$, so there $O(|\varphi| \cdot k^3)$ possible parameter values. Applying the encoding for $\Diamond$ and $\Box$ introduces $O(k)$ connectives.

More efficient encodings do exist but are outside the scope of this lecture.

BMC is mostly used for invariants like $\varphi = \Box a$. 

Unfolding the Transition Relation

For $k \geq 0$, let $[TS]_k = \bigwedge_{s \in L} \bigwedge_{i=0}^{k-1} T(s_i, s_{i+1})$.

Loop condition

For $k \geq 0$, let $loop_k = \bigvee_{i=0}^{k} T(s_k, s_i)$.

Encoding of bounded LTL

For $k \geq 0$ and LTL-formula $\varphi$, let $[\varphi]_k = [\varphi]_0^k \land \left( \bigvee_{i=0}^{k} loop_k \land [\varphi]_{i,k}^0 \right)$

where $[\varphi]_i^k$ is the encoding of $\varphi$ under the assumption that $\pi'$ has no $(k, \ell)$ loop, and $[\varphi]_{i,k}^k$ is the encoding of $\varphi$ in case $\pi'$ has a $(k, \ell)$-loop.
Translating BMC into SAT

**Definition: encoding BMC as SAT instance**

Let:

\[
\mathbb{S}_k = \bigwedge_{s \in I} (s \rightarrow s_{i+1})
\]

and

\[
\phi_k = \bigwedge_{i=0}^{k-1} \bigvee \text{loop}_k \land \phi^0_k
\]

Then:

\[
\left[ \mathbb{S}, \phi \right]_k = \left[ \mathbb{S}_k \right] \land \left[ \phi \right]_k.
\]

The input to a SAT solver is the CNF transformation of \(\left[ \mathbb{S}, \phi \right]_k\).

As \(k\) is increased iteratively and \(\left[ \mathbb{S}, \phi \right]_k\) and \(\left[ \mathbb{S}, \phi \right]_{k+1}\) have a lot in common, incremental SAT solving algorithms are exploited.

Review of BMC

- BMC can be used to **disprove invariants** \(\square \varphi\)
  - by proving \(\exists \Diamond \neg \varphi\) considering paths of length \(k\)
  - if paths longer than \(k\) are needed to show \(\exists \Diamond \neg \varphi\), then BMC fails

- BMC can be used to disprove liveness properties like \(\Diamond \varphi\)
  - by proving \(\exists \Box \neg \varphi\) for lassos of length \(k\)
  - if lassos longer than \(k\) are needed to show \(\exists \Box \neg \varphi\), then BMC fails

Thus: BMC is sound, but intrinsically incomplete

**BMC is in particular efficient if there are short counterexamples**

**Complete variant:** using completeness thresholds or \(k\)-induction

How to obtain a completeness threshold for \(k\)?

Towards Completeness

- Consider checking the invariant \(\varphi = \Box p\)

- Find bounds for the maximal length of counterexamples
  - these are referred to as **completeness threshold** \(ct\)
  - exact bounds are hard to find \(\Rightarrow\) use approximations

Idea: let \(ct\) be a completeness threshold for formula \(\varphi\).

\[
\mathbb{S} \not\models \varphi \iff \forall \pi \in \text{Paths}(\mathbb{S}), |\pi| \leq ct \Rightarrow \pi \not\models ct \varphi.
\]

If no path of length at most \(ct\) refutes \(\varphi\), the invariant holds
**Radius**

**Definition: radius of a transition system**

The radius of a transition system $TS$ is defined as:

$$ r_{TS} = \max\{ d(s, t) \mid s \in I \land s \rightarrow^* t \land d(s, t) = \min\{ d(s', t) \mid s' \in I \} \} $$

where $d(s, t)$ is the length of the shortest path from $s$ to $t$ in $TS$.

The radius is the maximal distance of a reachable state from some initial state in $TS$.

Radius = minimal number of steps to reach an arbitrary state in a BFS

**Completeness Threshold for Invariants**

- A bad state is reached in at most $r_{TS}$ steps from the initial states
- A bad state is a state violating the given invariant $\square p$
- Thus, the radius is a completeness threshold for invariants
- For invariants, the maximal $k$ for doing BMC is $r_{TS}$
- If no counterexample of this length can be found, the invariant holds.

How to obtain a propositional formula for radius for the SAT solver?

**Precise Bound**

Let $r$ be the radius of finite transition system $TS$. Then: $r$ is the minimal number such that:

$$ \forall s_0, \ldots, s_{r+1}. \left( l(s_0) \land \bigwedge_{i=0}^{r} T(s_i, s_{i+1}) \right) \Rightarrow \exists n \geq rr. \exists t_0, \ldots, t_n. \left( l(t_0) \land \bigwedge_{i=0}^{r-1} T(t_i, t_{i+1}) \land t_n = s_{r+1} \right). $$

Note that the conclusion is a quantified Boolean formula (QBF)

The satisfiability problem for such formulas is PSPACE-complete

This is much harder than solving the SAT problem for propositional logic

**Weakened Bounds**

The radius is mostly too hard to compute; thus: use an upper bound.

**Definition: recurrence radius**

The recurrence radius $rd_{TS}$ of transition system $TS$ is the maximal number $r$ which makes the following formula satisfiable:

$$ rd_{TS} = l(s_0) \land \bigwedge_{i=0}^{r-1} T(s_i, s_{i+1}) \land \bigwedge_{i=0}^{r-1} \bigwedge_{k=i+1}^{r} (s_j \neq s_k). $$

The recurrence radius can be computed by a SAT solver (instead of QBF)

It may be considerably larger than the radius of the transition system $TS$
Overview

1. Motivation
2. Bounded LTL Semantics
3. From BMC To SAT
4. Completeness and Distances
5. Summary

Summary

- Bounded model checking: use SAT solving
- To find finite counterexamples
- BMC unfolds the transition system up to a given depth
- BMC is incomplete: no counterexample of length $k$ does not mean that there are no counterexamples
- Extensions for completeness: completeness thresholds
- BMC is mostly used in practice for safety properties

Next Lecture

None