# Model Checking

Lecture #19: Symbolic Model Checking with SAT

[Biere, Chapter Handbook SAT & Wimmer, Lecture Slides]

Joost-Pieter Katoen

Software Modeling and Verification Group

Model Checking Course, RWTH Aachen, WiSe 2019/2020



## Overview







## Overview







## Objective

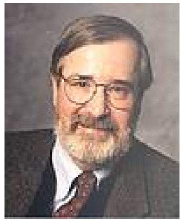
Consider model checking as a bug-hunting technique,
not as verification engine.

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 ($\exists\bigcirc$), 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)

$\Rightarrow$ Cast bug hunting $TS \not\models \varphi$ as Boolean satisfiability problem
  - for $\varphi \in$ LTL, $TS \not\models \varphi$ iff $TS \models \exists\neg\varphi$

  $\Rightarrow$ 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 $\exists$)

  $\Rightarrow$ 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 [1]
- 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 $\pi \models \Diamond^{\leq k} a$ implies $\pi \models \varphi$, and
  2. if $\pi$ is a lasso and $\pi \models \Box^{\leq k} a$, then $\pi \models \Box\, a$
- Transform this search into a SAT problem and exploit a SAT solver

[1] Compare the counterexample of $\Box\, a$ and of $\exists\Box\forall\bigcirc\, a$.





## A Simple Example

The same example as before but not tackled using BMC.





## Overview

## 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, $\pi \;=\; \underbrace{s_0 \ldots s_{\ell-1}}_{\pi_{stem}} \cdot \big(\underbrace{s_\ell \ldots s_k}_{\pi_{loop}}\big)^\omega$.

**Example**





## LTL Semantics Rephrased

Let $\varphi$ be an LTL-formula (without until, release) in positive normal form.

Let $\pi^i = \pi[i\ldots]$, 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:

$$\begin{array}{rcl}
\pi^i \models a & \text{iff} & a \in L(\pi[i]) \\
\pi^i \models \neg a & \text{iff} & a \notin L(\pi[i]) \\
\pi^i \models \varphi \wedge \psi & \text{iff} & \pi^i \models \varphi \text{ and } \pi^i \models \psi \\
\pi^i \models \varphi \vee \psi & \text{iff} & \pi^i \models \varphi \text{ or } \pi^i \models \psi \\
\pi^i \models \bigcirc \varphi & \text{iff} & \pi^{i+1} \models \varphi \\
\pi^i \models \Diamond \varphi & \text{iff} & \pi^{i+j} \text{ for some } j \geq i \\
\pi^i \models \Box \varphi & \text{iff} & \pi^{i+j} \text{ for all } j \geq i.
\end{array}$$





## 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 $\pi^{k+1+j} = \pi^{\ell+j}$ for all $j$, and:

$$\pi^i \models_k \bigcirc \varphi \quad \text{iff} \quad \begin{cases} \pi^{i+1} \models_k \varphi & \text{if} \quad i < k \\ \pi^{\ell} \models_k \varphi & \text{if} \quad i = k \end{cases}$$

$$\begin{array}{rcl}
\pi^i \models_k \Diamond \varphi & \text{iff} & \pi^j \models_k \varphi \text{ for some } j \in \{\min(i,\ell), \ldots, k\} \\
\pi^i \models_k \Box \varphi & \text{iff} & \pi^j \models_k \varphi \text{ for all } j \in \{\min(i,\ell), \ldots, k\}
\end{array}$$





## Bounded LTL Semantics for Non-Lassos

If $\pi$ is not an $(k, \ell)$-lasso for any $\ell$, then $\pi^{k+1}$ can have an arbitrary shape.

Considering only the first $k{+}1$ states of $\pi$ is insufficient to decide whether $\pi \models \varphi$.

But the following holds:

$$\begin{array}{rcl}
\pi^i \models_k \bigcirc \varphi & \text{if} & \pi^{i+1} \models_k \varphi \text{ and } i < k \\
\pi^i \models_k \Diamond \varphi & \text{if} & \pi^j \models_k \varphi \text{ for some } j \in \{i, \ldots, k\}
\end{array}$$

These sufficient conditions suffice for refuting safety properties.

Note: $\pi^i \not\models_k \Box \varphi$, for all $k$; so, the duality $\Box a \equiv \neg \Diamond \neg a$ no longer applies

## Some Properties

For every $\varphi \in$ 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 $TS \models_k \exists\varphi$.





## Overview







## 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 $k_{max}$) is reached.





## Pictorially

## 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 iff $\pi = s_0 \ldots s_k \models \varphi$ for some $\pi \in Paths(TS)$.

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 iff 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





## Translating BMC into SAT

### Unfolding the Transition Relation

For $k \geq 0$, let $[\![ \, TS \, ]\!]_k \;=\; \underbrace{I(s_0)}_{s \in I} \;\wedge\; \bigwedge_{i=0}^{k-1} \underbrace{T(s_i, s_{i+1})}_{s \longrightarrow s_{i+1}}$.

### Loop condition

For $k \geq 0$, let $loop_k \;=\; \bigvee_{\ell=0}^{k} T(s_k, s_\ell)$.

ßskip

### Encoding of bounded LTL

For $k \geq 0$ and LTL-formula $\varphi$, let: $[\![ \, \varphi \, ]\!]_k \;=\; [\![ \, \varphi \, ]\!]_k^0 \;\wedge\; \left( \bigvee_{\ell=0}^{k} loop_k \wedge [\![ \, \varphi \, ]\!]_{\ell,k}^0 \right)$

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





## Encoding Bounded LTL as SAT





## Property

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 $O(k^2)$ combinations. This holds for any sub-formula of $\varphi$, so there $O(|\varphi| \cdot k^2)$ possible parameter values. Applying the encoding for $\Diamond$ and $\Box$ introduces $O(k)$ connectives. $\square$

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

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

## Translating BMC into SAT

**Definition: encoding BMC as SAT instance**

Let:

$$[\![ TS ]\!]_k = \underbrace{I(s_0)}_{s \in I} \wedge \bigwedge_{i=0}^{k-1} \underbrace{T(s_i, s_{i+1})}_{s \to s_{i+1}}$$

and

$$[\![ \varphi ]\!]_k = [\![ \varphi ]\!]_k^0 \wedge \left( \bigvee_{\ell=0}^{k} loop_k \wedge [\![ \varphi ]\!]_{\ell,k}^0 \right)$$

Then:

$$[\![ TS, \varphi ]\!]_k = [\![ TS ]\!]_k \wedge [\![ \varphi ]\!]_k.$$

The input to a SAT solver is the CNF transformation of $[\![ TS, \varphi ]\!]_k$.

As $k$ is increased iteratively and $[\![ TS, \varphi ]\!]_k$ and $[\![ TS, \varphi ]\!]_{k+1}$ have a lot in common, incremental SAT solving algorithms are exploited.





## Review of BMC

- BMC can be used to disprove invariants $\Box \varphi$
  - by proving $\exists \Diamond \neg \varphi$ considering paths of length $k$
  - if paths longer than $k$ are needed to show $\exists \neg \Diamond \varphi$, 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 counterexample
- Complete variant: using completeness thresholds or $k$-induction

How to obtain a completeness threshold for $k$?





## Overview







## 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 $\varphi$.

$$TS \models \varphi \quad \text{iff} \quad \forall \pi \in Paths(TS). |\pi| \leq ct \Rightarrow \pi \models_{ct} \varphi.$$

If no path of length at most $ct$ refutes $\varphi$, the invariant holds

## Radius

**Definition: radius of a transition system**

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

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

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 $\Box\, p$
- 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:

$$\forall s_0, \ldots, s_{r+1}. \Big( I(s_0) \wedge \textstyle\bigwedge_{i=0}^{r} T(s_i, s_{i+1}) \Big)$$

$$\Rightarrow\; \exists n \geq r.\, \exists t_0, \ldots, t_n. \Big( I(t_0) \wedge \textstyle\bigwedge_{i=0}^{r} T(t_i, t_{i+1}) \wedge t_n = s_{r+1} \Big).$$

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 $rd$ of transition system $TS$ is the maximal number $r$ which makes the following formula satisfiable:

$$rd_{TS} \;=\; \underbrace{I(s_0) \wedge \bigwedge_{i=0}^{r-1} T(s_i, s_{i+1})}_{[\![\, TS \,]\!]_r} \wedge \underbrace{\bigwedge_{i=0}^{r-1} \bigwedge_{k=i+1}^{r} (s_j \neq s_k)}_{\text{loop freeness}}.$$

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







## 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