Bounded Model Checking
SAT-Based Model Checking

Wallace Wu

Department of Electrical and Computer Engineering
University of Waterloo

March 3, 2011
Outline

Introduction to Bounded Model Checking (BMC)
   BMC vs. BDD Model Checkers

A Quick Review of Model Checking Concepts
   Kripke Structures
   LTL Syntax and Semantics
   LTL Model Checking

Bounded Model Checking
   Bounded Path Syntax and Semantics
   Model Checking Problem Reduction

Reducing BMC to SAT
   Example of Checking Mutual Exclusion using BMC

Techniques for Completeness
   Completeness Threshold
   Liveness
   Induction
Outline

Introduction to Bounded Model Checking (BMC)
   BMC vs. BDD Model Checkers

A Quick Review of Model Checking Concepts
   Kripke Structures
   LTL Syntax and Semantics
   LTL Model Checking

Bounded Model Checking
   Bounded Path Syntax and Semantics
   Model Checking Problem Reduction

Reducing BMC to SAT
   Example of Checking Mutual Exclusion using BMC

Techniques for Completeness
   Completeness Threshold
   Liveness
   Induction
Model Checking

- Exhaustive model checking algorithms
  - Inefficient; enumerate all states and transitions
  - Check a few million states in reasonable amount of time
Model Checking

- Exhaustive model checking algorithms
  - Inefficient; enumerate all states and transitions
  - Check a few million states in reasonable amount of time
- Symbolic model checking
  - Represent states using Boolean functions
  - Manipulating Boolean formulas: Reduced Ordered Binary Decision Trees (ROBDD or BDD for short)
  - Check $\geq 10^{20}$ states in reasonable amount of time
  - **Bottleneck**: Memory required for storing and manipulating BDDs
  - Full design verifications is generally still beyond the capacity of BDD-based model checkers
About BMC

- Proposed by Biere, Clarke et al. [Biere et al., 1999]
About BMC

- Proposed by Biere, Clarke et al. [Biere et al., 1999]
- BMC relies on exponential procedure (still limited in its capacity)
About BMC

- Proposed by Biere, Clarke et al. [Biere et al., 1999]
- BMC relies on exponential procedure (still limited in its capacity)
- Complimentary to BDD-based model checking
  - BMC can solve many cases that BDD-based techniques cannot and vice versa
  - No correlation between hardness of SAT and BDD problems
  - Does NOT replace other model checking techniques
About BMC

- Proposed by Biere, Clarke et al. [Biere et al., 1999]
- BMC relies on exponential procedure (still limited in its capacity)
- Complimentary to BDD-based model checking
  - BMC can solve many cases that BDD-based techniques cannot and vice versa
  - No correlation between hardness of SAT and BDD problems
  - Does NOT replace other model checking techniques
- **Disadvantage:** Cannot prove absence of errors in most realistic cases
Overview of BMC

Idea: Search for a counterexample in executions (paths) whose length is \( \leq k \) for some integer \( k \)
Overview of BMC

Idea: Search for a counterexample in executions (paths) whose length is $\leq k$ for some integer $k$

Method: Efficiently reduce problem to a propositional satisfiability (SAT) problem
  ▶ Resolves state explosion problem
Overview of BMC

Idea: Search for a counterexample in executions (paths) whose length is \( \leq k \) for some integer \( k \)

Method: Efficiently reduce problem to a propositional satisfiability (SAT) problem

\[ \text{Resolves state explosion problem} \]

Process: If no bug is found, increase \( k \) until either:

\[ \begin{align*}
\text{A bug is found} \\
\text{Problem becomes intractable} \\
\text{Some predetermined upper bound for } k \text{ is reached}
\end{align*} \]
Unique Characteristics of BMC

- User must provide a bound on the number of cycles that should be explored
  - Experiments show that BMC outperforms BDD-based techniques for $k$ up to $\sim 60 - 80$
Unique Characteristics of BMC

- User must provide a bound on the number of cycles that should be explored
  - Experiments show that BMC outperforms BDD-based techniques for $k$ up to $\sim 60 - 80$
- Uses SAT solving techniques to check models
  - Details of SAT solvers not covered
Outline

Introduction to Bounded Model Checking (BMC)
  BMC vs. BDD Model Checkers

A Quick Review of Model Checking Concepts
  Kripke Structures
  LTL Syntax and Semantics
  LTL Model Checking

Bounded Model Checking
  Bounded Path Syntax and Semantics
  Model Checking Problem Reduction

Reducing BMC to SAT
  Example of Checking Mutual Exclusion using BMC

Techniques for Completeness
  Completeness Threshold
  Liveness
  Induction
Verifying a 16 × 16 bit shift and add multiplier

<table>
<thead>
<tr>
<th>bit</th>
<th>k</th>
<th>$t_{exec}$ (s)</th>
<th>Memory (MB)</th>
<th>$t_{exec}$ (s)</th>
<th>Memory (MB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>25</td>
<td>79</td>
<td>&lt;1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>25</td>
<td>79</td>
<td>&lt;1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>26</td>
<td>80</td>
<td>&lt;1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>27</td>
<td>82</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>33</td>
<td>92</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>67</td>
<td>102</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>258</td>
<td>172</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>1741</td>
<td>492</td>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td>&gt;1024</td>
<td>29</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>58</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>11</td>
<td>91</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>125</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>13</td>
<td>156</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>14</td>
<td>186</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>15</td>
<td>226</td>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>16</td>
<td>183</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Verifying Various Designs

<table>
<thead>
<tr>
<th>Model</th>
<th>$k$</th>
<th>Rulebase&lt;sub&gt;1&lt;/sub&gt;</th>
<th>Rulebase&lt;sub&gt;2&lt;/sub&gt;</th>
<th>Grasp</th>
<th>Grasp (tuned)</th>
<th>Chaff</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>18</td>
<td>7</td>
<td>6</td>
<td>282</td>
<td>3</td>
<td>2.2</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>70</td>
<td>8</td>
<td>1.1</td>
<td>0.8</td>
<td>&lt;1</td>
</tr>
<tr>
<td>3</td>
<td>14</td>
<td>597</td>
<td>375</td>
<td>76</td>
<td>3</td>
<td>&lt;1</td>
</tr>
<tr>
<td>4</td>
<td>24</td>
<td>690</td>
<td>261</td>
<td>510</td>
<td>12</td>
<td>3.7</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>803</td>
<td>184</td>
<td>24</td>
<td>2</td>
<td>&lt;1</td>
</tr>
<tr>
<td>6</td>
<td>22</td>
<td>–</td>
<td>356</td>
<td>–</td>
<td>18</td>
<td>12.2</td>
</tr>
<tr>
<td>7</td>
<td>9</td>
<td>–</td>
<td>2671</td>
<td>10</td>
<td>2</td>
<td>&lt;1</td>
</tr>
<tr>
<td>8</td>
<td>35</td>
<td>–</td>
<td>–</td>
<td>6317</td>
<td>20</td>
<td>85</td>
</tr>
<tr>
<td>9</td>
<td>38</td>
<td>–</td>
<td>–</td>
<td>9035</td>
<td>25</td>
<td>131.6</td>
</tr>
<tr>
<td>10</td>
<td>31</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>312</td>
<td>380.5</td>
</tr>
<tr>
<td>11</td>
<td>32</td>
<td>152</td>
<td>60</td>
<td>–</td>
<td>–</td>
<td>34.7</td>
</tr>
<tr>
<td>12</td>
<td>31</td>
<td>1419</td>
<td>1126</td>
<td>–</td>
<td>–</td>
<td>194.3</td>
</tr>
<tr>
<td>13</td>
<td>4</td>
<td>–</td>
<td>3626</td>
<td>–</td>
<td>–</td>
<td>9.8</td>
</tr>
</tbody>
</table>

All values that are right of the column $k$ are given in seconds.
Outline

Introduction to Bounded Model Checking (BMC)
- BMC vs. BDD Model Checkers

A Quick Review of Model Checking Concepts
- Kripke Structures
- LTL Syntax and Semantics
- LTL Model Checking

Bounded Model Checking
- Bounded Path Syntax and Semantics
- Model Checking Problem Reduction

Reducing BMC to SAT
- Example of Checking Mutual Exclusion using BMC

Techniques for Completeness
- Completeness Threshold
- Liveness
- Induction
Outline

Introduction to Bounded Model Checking (BMC)
  BMC vs. BDD Model Checkers

A Quick Review of Model Checking Concepts
  Kripke Structures
  LTL Syntax and Semantics
  LTL Model Checking

Bounded Model Checking
  Bounded Path Syntax and Semantics
  Model Checking Problem Reduction

Reducing BMC to SAT
  Example of Checking Mutual Exclusion using BMC

Techniques for Completeness
  Completeness Threshold
  Liveness
  Induction
Kripke Structure

- The finite automaton can be represented by a Kripke structure, a quadruple $M = (S, I, T, L)$ where
  - $S$ is the set of states
  - $I$ is the set of initial states, $I \subseteq S$
  - $T$ is the translation relation, $T \subseteq S \times S$
  - $L$ is the labeling function, $L : S \rightarrow 2^A$, where $A$ is the set of atomic propositions, and $2^A$ is the powerset of $A$
    - $L(s), s \in S$, is made of $A_s \subseteq A$ that hold in $s$
Sequential Behaviour of Kripke Structures

- Use the notion of paths to define behaviour of a Kripke structure, $M$.
- Each path, $\pi$ in $M$ is an infinite OR finite sequence of states in an order that respects $T$

$$\pi = (s_0, s_1, \ldots), \quad T(s_i, s_{i+1}) \forall 0 \leq i < |\pi| - 1$$

- For $i < |\pi|$:
  - $\pi(i)$ denotes the $i$-th state $s_i$ in the sequence
  - $\pi_j = (s_i, s_{i+1}, \ldots)$ denotes the suffix of $\pi$ starting with state $s_i$
If $I(s_0)$ (i.e., $s_0$ is an initial state) and $s_0 \in \pi$, $\pi$ is an initialized path

- If a state is not reachable $\implies$ no initialized paths that contain it
Assumptions about Kripke Structures

For a Kripke structure, $M$,

- $I \neq \emptyset$
- $\forall s \in S, \exists t \in S$ with $T(s, t)$ (total transition relation)
Mutual Exclusion Example

Pseudocode

PROCESS A
1  \hspace{5pt} A.pc = 0
2  \hspace{5pt} while TRUE
3      \hspace{5pt} wait for B.pc == 0
4  \hspace{5pt} A.pc = 1
5      \hspace{5pt} // critical section
6  \hspace{5pt} A.pc = 0

PROCESS B
1  \hspace{5pt} B.pc = 0
2  \hspace{5pt} while TRUE
3      \hspace{5pt} wait for A.pc == 0
4  \hspace{5pt} B.pc = 1
5      \hspace{5pt} // critical section
6  \hspace{5pt} B.pc = 0
Mutual Exclusion Example

Modeling

- We can encode the set of states using $A.pc$ and $B.pc$: $A.pc \cdot B.pc$. 
Mutual Exclusion Example

Modeling

- We can encode the set of states using $A.pc$ and $B.pc$: $A.pc \cdot B.pc$.

![Diagram]

- The transition relation $T \subseteq S^2 = \{0, 1\}^4$ is:

$$T = \{0100, 1000, 1100, 0001, 0010\}$$
Mutual Exclusion Example

Modeling

- We can encode the set of states using $A.pc$ and $B.pc$: $A.pc \cdot B.pc$.

- The transition relation $T \subseteq S^2 = \{0, 1\}^4$ is:

  $$T = \{0100, 1000, 1100, 0001, 0010\}$$

- The sequence $11, 00, 10, \ldots$ is a valid path, but it is not initialized, since $I = \{s_0\}$
Outline

Introduction to Bounded Model Checking (BMC)
  BMC vs. BDD Model Checkers

A Quick Review of Model Checking Concepts
  Kripke Structures
  LTL Syntax and Semantics
  LTL Model Checking

Bounded Model Checking
  Bounded Path Syntax and Semantics
  Model Checking Problem Reduction

Reducing BMC to SAT
  Example of Checking Mutual Exclusion using BMC

Techniques for Completeness
  Completeness Threshold
  Liveness
  Induction
LTL Temporal Operators

Let $f$ and $g$ be temporal formulas. The temporal operators are:

**Next time:** $\Diamond f$

Next time: $\Diamond f$

**Globally:** $\Box f$

Globally: $\Box f$

**Finally:** $\Diamond f$

Finally: $\Diamond f$

**Until:** $f U g$

Until: $f U g$

**Release:** $f R g$

Release: $f R g$
LTL Semantics

Let $\pi$ be an infinite path of a Kripke structure $M$ and let $f, g, p$ be temporal formulas. We recursively define LTL semantics as:

- $\pi \models p$ if $p \in L(\pi(0))$
- $\pi \models \neg p$ if $\pi \not\models f$
- $\pi \models f \land g$ if $\pi \models f$ and $\pi \models g$
- $\pi \models \Diamond f$ if $\pi_1 \models f$
- $\pi \models \Box f$ if $\pi_i \models f$ for all $i \geq 0$
- $\pi \models \Diamond f$ if $\pi_i \models f$ for some $i \geq 0$
- $\pi \models f U g$ if $\pi_i \models g$ for some $i \geq 0$ and $\pi_j \not\models f$ for all $0 \leq j < i$
- $\pi \models f R g$ if $\pi_i \models g$ if for all $j < i$, $\pi_j \not\models f$
LTL Semantics (cont’d)

- $M \models f \Rightarrow \pi \models f \quad \forall$ initialized paths $\pi$ of $M$
- LTL formulas $f$ and $g$ are equivalent (i.e., $f \equiv g$) iff $M \models f \iff M \models g \quad \forall M$
- **Duality:** $\neg \Diamond \neg p \equiv \Box p$
Outline

Introduction to Bounded Model Checking (BMC)
   BMC vs. BDD Model Checkers

A Quick Review of Model Checking Concepts
   Kripke Structures
   LTL Syntax and Semantics
   LTL Model Checking

Bounded Model Checking
   Bounded Path Syntax and Semantics
   Model Checking Problem Reduction

Reducing BMC to SAT
   Example of Checking Mutual Exclusion using BMC

Techniques for Completeness
   Completeness Threshold
   Liveness
   Induction
LTL Model Checking

- Standard technique:
  - compute **product** of Kripke structure $M$ with automaton representing the negation of the property to be checked

$$A_{\neg \phi}$$

- **emptiness** of the product automaton $\Rightarrow$ correctness of the property

$$L(M \parallel A_{\neg \phi}) = \emptyset \Rightarrow M \models \phi$$
Outline

Introduction to Bounded Model Checking (BMC)
- BMC vs. BDD Model Checkers

A Quick Review of Model Checking Concepts
- Kripke Structures
- LTL Syntax and Semantics
- LTL Model Checking

Bounded Model Checking
- Bounded Path Syntax and Semantics
- Model Checking Problem Reduction

Reducing BMC to SAT
- Example of Checking Mutual Exclusion using BMC

Techniques for Completeness
- Completeness Threshold
- Liveness
- Induction
Motivation was to leverage success in SAT solving in model checking
Motivation was to leverage success in SAT solving in model checking

Search for counterexample within a predetermined bound

- i.e., Consider only prefixes of paths bounded by \( k \) in the search
- In practice, progressively increase \( k \), looking for longer witnesses in longer traces
Motivation was to leverage success in SAT solving in model checking

Search for counterexample within a predetermined bound
  i.e., Consider only prefixes of paths bounded by $k$ in the search
  In practice, progressively increase $k$, looking for longer witnesses in longer traces

Since LTL formulas are defined over all paths, a counterexample is a trace/path that contradicts the property
  such a trace is called a witness for the property
  **Example:** a counterexample to $M \models \Box p$ is the existence of a witness such that $\Diamond \neg p$
Outline

Introduction to Bounded Model Checking (BMC)
  BMC vs. BDD Model Checkers

A Quick Review of Model Checking Concepts
  Kripke Structures
  LTL Syntax and Semantics
  LTL Model Checking

Bounded Model Checking
  Bounded Path Syntax and Semantics
  Model Checking Problem Reduction

Reducing BMC to SAT
  Example of Checking Mutual Exclusion using BMC

Techniques for Completeness
  Completeness Threshold
  Liveness
  Induction
Prefixes

- Bounded semantics approximate unbounded semantics
Prefixes

- Bounded semantics approximate unbounded semantics
- In BMC, finite prefixes of paths are considered
Prefixes

- Bounded semantics approximate unbounded semantics
- In BMC, finite prefixes of paths are considered
- Only the first \( k + 1 \) states \((s_0, \ldots, s_k)\) of a path are used
Prefixes

- Bounded semantics approximate unbounded semantics
- In BMC, finite prefixes of paths are considered
- Only the first $k + 1$ states ($s_0, \ldots, s_k$) of a path are used
- A finite length prefix represents an infinite path if there is a back loop from the last state of the prefix to any previous state(s)

Figure: Prefix with back loop
Prefixes

- Bounded semantics approximate unbounded semantics
- In BMC, finite prefixes of paths are considered
- Only the first \( k + 1 \) states \((s_0, \ldots, s_k)\) of a path are used
- A finite length prefix represents an infinite path if there is a back loop from the last state of the prefix to any previous state(s)
- A prefix without back loop(s) only represents the finite behaviour of the path up to state \( s_k \)

Figure: Prefix with back loop

Figure: Prefix without back loop
Example of Prefix Behaviour

- Consider the LTL property: $\Box p$
Example of Prefix Behaviour

- Consider the LTL property: □ \( p \)
- With back loop:
  - Property can be satisfied in the prefix
Example of Prefix Behaviour

- Consider the LTL property: □p
- With back loop:
  - Property can be satisfied in the prefix
- Without back loop:
  - Property CANNOT be satisfied in the prefix
  - If p holds for all states s_0, . . . , s_k, we still cannot conclude that the property holds since p may not hold at s_{k+1}
Prefixes With Back Loops - \((k, l)\)-loop

**Definition 1**
For \(1 \leq k\), we call a path \(\pi\) a \((k, l)\)-loop if \(T(\pi(k), \pi(l))\) and \(\pi = u \cdot v^\omega\) with \(u = (\pi(0), \ldots, \pi(l-1))\) and \(v = (\pi(l), \ldots, \pi(k))\). We call \(\pi\) a \(k\)-loop if there exists \(k \geq l \geq 0\) for which \(\pi\) is a \((k, l)\)-loop.
Prefixes With Back Loops - \((k, l)\)-loop

**Definition 1**

For \(1 \leq k\), we call a path \(\pi\) a \((k, l)\)-loop if \(T(\pi(k), \pi(l))\) and 
\[\pi = u \cdot v^\omega\text{ with } u = (\pi(0), \ldots, \pi(l-1))\text{ and } v = (\pi(l), \ldots, \pi(k)).\]

We call \(\pi\) a \(k\)-loop if there exists \(k \geq l \geq 0\) for which \(\pi\) is a \((k, l)\)-loop.

- If a path is a \(k\)-loop, then the original LTL semantics are maintained \(\because\) infinite path represented in prefix.
Prefixes With Back Loops - $(k, l)$-loop

**Definition 1**
For $1 \leq k$, we call a path $\pi$ a $(k, l)$-loop if $T(\pi(k), \pi(l))$ and $\pi = u \cdot v^\omega$ with $u = (\pi(0), \ldots, \pi(l - 1))$ and $v = (\pi(l), \ldots, \pi(k))$. We call $\pi$ a $k$-loop if there exists $k \geq l \geq 0$ for which $\pi$ is a $(k, l)$-loop.

- If a path is a $k$-loop, then the original LTL semantics are maintained ($\because$ infinite path represented in prefix)

**Definition 2 (Bounded Semantics for a Loop)**
Let $k \geq 0$ and $\pi$ be a $k$-loop. Then an LTL formula $f$ is valid along the path $\pi$ with bound $k$ (denoted by $\pi \models_k f$) iff $\pi \models f$. 

![Diagram](image-url)
Prefixes Without Back Loops

- ◇\(p\) is valid along \(\pi\) in unbounded semantics if \(\exists i \geq 0\) s.t. \(p\) is valid along the suffix \(\pi_i\) of \(\pi\).
Prefixes Without Back Loops

- ♦ \( p \) is valid along \( \pi \) in unbounded semantics if \( \exists i \geq 0 \) s.t. \( p \) is valid along the suffix \( \pi_i \) of \( \pi \)

- In bounded semantics, \((k + 1)\)-th state \( \pi(k) \) does not have a successor
  - Cannot define bounded semantics recursively over suffixes of \( \pi \)
Prefixes Without Back Loops

- ♦ $p$ is valid along $\pi$ in **unbounded** semantics if $\exists i \geq 0$ s.t. $p$ is valid along the suffix $\pi_i$ of $\pi$
- In **bounded** semantics, $(k + 1)$-th state $\pi(k)$ does not have a successor
  - Cannot define bounded semantics recursively over suffixes of $\pi$
- Introduce notation:
  \[ \pi \models^i_k f \]
  where $i$ is the current position in the prefix of $\pi$
  - Implies suffix $\pi_i$ of $\pi$ satisfies $f$, i.e.,
  \[ \pi \models^i_k \Rightarrow \pi_i \models f \]
Semantics of Prefixes Without Back Loops

Definition 3 (Bounded Semantics without a Loop)
Let $k \geq 0$, and $\pi$ be a path that is not a $k$-loop. An LTL formula $f$ is valid along $\pi$ with bound $k$ (denoted by $\pi \models_k f$) iff $\pi \models^0_k f$ where

\[
\begin{align*}
\pi \models^i_k p & \iff p \in L(\pi(i)) \\
\pi \models^i_k \neg p & \iff p \notin L(\pi(i)) \\
\pi \models^i_k f \land g & \iff \pi \models^i_k f \text{ and } \pi \models^i_k g \\
\pi \models^i_k f \lor g & \iff \pi \models^i_k f \text{ or } \pi \models^i_k g \\
\pi \models^i_k \Box f & \text{ is always false} \\
\pi \models^i_k \Diamond f & \iff \exists j, i \leq j \leq k \bullet \pi \models^j_k f
\end{align*}
\]
Semantics of Prefixes Without Back Loops

\[ \pi \models^i_k \bigcirc f \iff i < k \text{ and } \pi \models^i_{k+1} f \]
\[ \pi \models^i_k f \mathcal{U} g \iff \exists j, i \leq j \leq k \bullet \pi \models^j_k g \]

and \( \forall n, i \leq n < j \bullet \pi \models^n_k f \)

\[ \pi \models^i_k f \mathcal{R} g \iff \exists j, i \leq j \leq k \bullet \pi \models^i_k f \]

and \( \forall n, i \leq n < j \bullet \pi \models^n_k g \)
Semantics of Prefixes Without Back Loops

\[ \pi \models_{k}^i f \circ g \iff i < k \text{ and } \pi \models_{k}^{i+1} f \]
\[ \pi \models_{k}^i f U g \iff \exists j, i \leq j \leq k \bullet \pi \models_{k}^j g \]
\[ \quad \text{and } \forall n, i \leq n < j \bullet \pi \models_{k}^n f \]
\[ \pi \models_{k}^i f R g \iff \exists j, i \leq j \leq k \bullet \pi \models_{k}^j f \]
\[ \quad \text{and } \forall n, i \leq n < j \bullet \pi \models_{k}^n g \]

\[ \square f \text{ is not valid along } \pi \text{ in } k\text{-bounded semantics since } f \text{ may not hold for } \pi_{k+1} \]
Semantics of Prefixes Without Back Loops

\[ \pi \models^i_k f \circ f \iff i < k \text{ and } \pi \models^{i+1}_k f \]
\[ \pi \models^i_k f \mathbf{U} g \iff \exists j, i \leq j \leq k \cdot \pi \models^j_k g \]
\[ \quad \text{and } \forall n, i \leq n < j \cdot \pi \models^n_k f \]
\[ \pi \models^i_k f \mathbf{R} g \iff \exists j, i \leq j \leq k \cdot \pi \models^j_k f \]
\[ \quad \text{and } \forall n, i \leq n < j \cdot \pi \models^n_k g \]

\[ \square f \text{ is not valid along } \pi \text{ in } k\text{-bounded semantics since } f \text{ may not hold for } \pi_{k+1} \]
\[ \neg \diamond f \not\equiv \square \neg f, \text{ i.e., duality between } \square \text{ and } \diamond \text{ no longer holds} \]
Outline

Introduction to Bounded Model Checking (BMC)
  BMC vs. BDD Model Checkers

A Quick Review of Model Checking Concepts
  Kripke Structures
  LTL Syntax and Semantics
  LTL Model Checking

Bounded Model Checking
  Bounded Path Syntax and Semantics
  Model Checking Problem Reduction

Reducing BMC to SAT
  Example of Checking Mutual Exclusion using BMC

Techniques for Completeness
  Completeness Threshold
  Liveness
  Induction
Model Checking Problem Reduction

- Introduce path quantifiers $\mathbf{E}$ and $\mathbf{A}$
  - $\mathbf{E}$ denotes that an LTL formula is expected to be correct over some path
  - $\mathbf{A}$ denotes that an LTL formula is expected to be correct over all paths
Model Checking Problem Reduction

- Introduce path quantifiers $E$ and $A$
  - $E$ denotes that an LTL formula is expected to be correct over some path
  - $A$ denotes that an LTL formula is expected to be correct over all paths
- The existential model checking problem $M \models Ef$ can be reduced to a bounded existential model checking problem $M \models_{k} Ef$
Model Checking Problem Reduction

- Introduce path quantifiers $E$ and $A$
  - $E$ denotes that an LTL formula is expected to be correct over some path
  - $A$ denotes that an LTL formula is expected to be correct over all paths
- The existential model checking problem $M \models E \phi$ can reduced to a bounded existential model checking problem $M \models_k E \phi$
- $M \models E \phi$ means $\exists$ an initialized path in $M$ that satisfies $\phi$
Model Checking Problem Reduction (cont’d)

- Basis for this reduction lies in the following lemmas
Basis for this reduction lies in the following lemmas

Lemma 1

Let \( f \) be an LTL formula and \( \pi \) a path, then \( \pi \models_k f \Rightarrow \pi \models f \)
Basis for this reduction lies in the following lemmas

Lemma 1
Let $f$ be an LTL formula and $\pi$ a path, then $\pi \models \forall k \ f \rightarrow \pi \models f$

Lemma 2
Let $f$ be an LTL formula and $M$ a Kripke structure. If $M \models E f$, there exists $k \geq 0$ with $M \models E f$. 
The following theorem is derived from the lemmas:

**Theorem 1**

*Let $f$ be an LTL formula and $M$ a Kripke structure. Then $M \models Ef$ iff $\exists k \geq 0$ such that $M \models_k Ef$.***
The following theorem is derived from the lemmas:

**Theorem 1**

Let $f$ be an LTL formula and $M$ a Kripke structure. Then $M \models Ef$ iff $\exists k \geq 0$ such that $M \models_k Ef$.

- Informally, it means that for a sufficiently high bound, bounded and unbounded semantics are equivalent.
Outline

Introduction to Bounded Model Checking (BMC)
   BMC vs. BDD Model Checkers

A Quick Review of Model Checking Concepts
   Kripke Structures
   LTL Syntax and Semantics
   LTL Model Checking

Bounded Model Checking
   Bounded Path Syntax and Semantics
   Model Checking Problem Reduction

Reducing BMC to SAT
   Example of Checking Mutual Exclusion using BMC

Techniques for Completeness
   Completeness Threshold
   Liveness
   Induction
BMC Reduction

- Given a Kripke structure $M$, an LTL formula $f$, and a bound $k$, we can construct a propositional formula

$$[[M, f]]_k$$
BMC Reduction

- Given a Kripke structure $M$, an LTL formula $f$, and a bound $k$, we can construct a propositional formula

$$[[M, f]]_k$$

- Let $s_0, \ldots, s_k$ be a finite sequence of states on path $\pi$
BMC Reduction

- Given a Kripke structure $M$, an LTL formula $f$, and a bound $k$, we can construct a propositional formula $[[M, f]]_k$

- Let $s_0, \ldots, s_k$ be a finite sequence of states on path $\pi$
- Each state $s_i$ represents a state at time step $i$ and consists of an assignment of truth values to the set of state variables
BMC Reduction

- Given a Kripke structure $M$, an LTL formula $f$, and a bound $k$, we can construct a propositional formula

  $$[[M, f]]_k$$

- Let $s_0, \ldots, s_k$ be a finite sequence of states on path $\pi$
- Each state $s_i$ represents a state at time step $i$ and consists of an assignment of truth values to the set of state variables
- Encode constraints on $s_0, \ldots, s_k$ so that

  $$[[M, f]]_k$$ is satisfiable $\iff$ $\pi$ is a witness for $f$
Definition of $[[M, f]]_k$

- Three components define $[[M, f]]_k$
Definition of $[[M, f]]_k$

- Three components define $[[M, f]]_k$
  - Propositional formula $[[M]]_k$: constrains $s_0, \ldots, s_k$ to be a valid path starting from an initial state
Definition of $[[M, f]]_k$

- Three components define $[[M, f]]_k$
  - Propositional formula $[[M]]_k$: constrains $s_0, \ldots, s_k$ to be a valid path starting from an initial state
  - Loop condition: a propositional formula that is evaluated to true only if the path $\pi$ contains a loop
Definition of $[[M, f]]_k$

- Three components define $[[M, f]]_k$
  - Propositional formula $[[M]]_k$: constrains $s_0, \ldots, s_k$ to be a valid path starting from an initial state
  - Loop condition: a propositional formula that is evaluated to true only if the path $\pi$ contains a loop
  - Propositional formula that constrains $\pi$ to satisfy $f$
Definition 4 (Unfolding of the Transition Relation)
For a Kripke structure $M$, $k \geq 0$

$$[[M]]_k := I(s_0) \land \bigwedge_{i=0}^{k-1} T(s_i, s_{i+1})$$
Loop Condition

- Define propositional formula $\_L_k$ to be true iff there is a transition from state $s_k$ to state $s_l$.
- By definition, $\_L_k = T(s_k, s_l)$
Loop Condition

- Define propositional formula $\_L_k$ to be true iff there is a transition from state $s_k$ to state $s_l$.
- By definition, $\_L_k = T(s_k, s_l)$

Definition 5 (Loop Condition)
The loop condition $L_k$ is true iff there exists a back loop from state $s_k$ to a previous state or to itself. We define $L_k$ to be:

$$L_k := \bigvee_{l=0}^{k} \_L_k$$
Definition 6 (Successor in a Loop)

Let $k$, $l$ and $i$ be non-negative integers s.t. $l, i \leq k$. Define the successor $\text{succ}(i)$ of $i$ in a $(k, l)$-loop as:

$$\text{succ}(i) := \begin{cases} 
  i + 1 & i < k \\
  l & i = k 
\end{cases}$$
Definition 7 (Translation of an LTL Formula for a Loop)

Let $f$ be an LTL formula, $k, l, i \geq 0$, with $l, i \leq k$.

\[
\begin{align*}
\lbrack [p] \rbrack_k^i & := p(s_i) \\
\lbrack [\neg p] \rbrack_k^i & := \neg p(s_i) \\
\lbrack [f \lor g] \rbrack_k^i & := \lbrack [f] \rbrack_k^i \lor \lbrack [g] \rbrack_k^i \\
\lbrack [f \land g] \rbrack_k^i & := \lbrack [f] \rbrack_k^i \land \lbrack [g] \rbrack_k^i \\
\lbrack [\Box f] \rbrack_k^i & := \lbrack [f] \rbrack_k^i \land \lbrack [\Box f] \rbrack_{\text{succ}(i)}^i \\
\lbrack [\Diamond f] \rbrack_k^i & := \lbrack [f] \rbrack_k^i \lor \lbrack [\Diamond f] \rbrack_{\text{succ}(i)}^i \\
\lbrack [f \mathbf{U} g] \rbrack_k^i & := \lbrack [g] \rbrack_k^i \lor \left( \lbrack [f] \rbrack_k^i \land \lbrack [f \mathbf{U} g] \rbrack_{\text{succ}(i)}^i \right) \\
\lbrack [f \mathbf{R} g] \rbrack_k^i & := \lbrack [g] \rbrack_k^i \land \left( \lbrack [f] \rbrack_k^i \lor \lbrack [f \mathbf{R} g] \rbrack_{\text{succ}(i)}^i \right) \\
\lbrack [\circlearrowleft f] \rbrack_k^i & := \lbrack [f] \rbrack_{\text{succ}(i)}^i
\end{align*}
\]

- $\lbrack [\cdot] \rbrack_k^i$ is an intermediate formula
  - $l$ and $k$ defines the start and end of the $(k, l)$-loop
  - $i$ for the current position in the path
Definition 8 (Translation of an LTL Formula without a Loop)

Inductive Case $\forall i \leq k$

\[
\begin{align*}
[[p]]_k^i & := p(s_i) \\
[[\neg p]]_k^i & := \neg p(s_i) \\
[[f \lor g]]_k^i & := [[f]]_k^i \lor [[g]]_k^i \\
[[f \land g]]_k^i & := [[f]]_k^i \land [[g]]_k^i \\
[[\Box f]]_k^i & := [[f]]_k^i \land [[\Box f]]_k^{i+1} \\
[[\Diamond f]]_k^i & := [[f]]_k^i \lor [[\Diamond f]]_k^{i+1} \\
[[fUg]]_k^i & := [[g]]_k^i \lor \left( [[f]]_k^i \land [[fUg]]_k^{i+1} \right) \\
[[fRg]]_k^i & := [[g]]_k^i \land \left( [[f]]_k^i \lor [[fRg]]_k^{i+1} \right) \\
[[\bigcirc f]]_k^i & := [[f]]_k^{i+1}
\end{align*}
\]

Base Case

\[
[[f]]_k^{k+1} := 0
\]
Definition 9 (General Translation)

Let $f$ be an LTL formula, $M$ a Kripke structure and $k \geq 0$.

$$[[M, f]]_k := [[M]]_k \land \left( \neg L_k \land [[f]]_0^k \lor \bigvee_{l=0}^k \left( l L_k \land l [[f]]_k' \right) \right)$$
Satisfiability and Bounded Model Checking

Theorem 2

\[[[M, f]]_k \text{ is satisfiable iff } M \models_k \text{ Ef} \]
Outline

Introduction to Bounded Model Checking (BMC)
  BMC vs. BDD Model Checkers

A Quick Review of Model Checking Concepts
  Kripke Structures
  LTL Syntax and Semantics
  LTL Model Checking

Bounded Model Checking
  Bounded Path Syntax and Semantics
  Model Checking Problem Reduction

Reducing BMC to SAT
  Example of Checking Mutual Exclusion using BMC

Techniques for Completeness
  Completeness Threshold
  Liveness
  Induction
Recall the earlier pseudocode for two processes that wish to gain access to a shared resource:

**PROCESS A**

1. $A.pc = 0$
2. **while** TRUE
3. wait for $B.pc == 0$
4. $A.pc = 1$
5. // critical section
6. $A.pc = 0$

**PROCESS B**

1. $B.pc = 0$
2. **while** TRUE
3. wait for $A.pc == 0$
4. $B.pc = 1$
5. // critical section
6. $B.pc = 0$
Mutual Exclusion Example

Modeling

- Each state $s$ of the system $M$ is represented by two-bit variables
  - $s[1]$: high bit (Process A)
  - $s[0]$: low bit (Process B)
Mutual Exclusion Example

Modeling

- Each state $s$ of the system $M$ is represented by two-bit variables
  - $s[1]$: high bit (Process A)
  - $s[0]$: low bit (Process B)
- Initial state:
  $$I(s) := \neg s[1] \land \neg s[0]$$
Mutual Exclusion Example
Modeling (cont’d)

Transition relation:

\[ T(s, s') := (\neg s[1] \land (s[0] \leftrightarrow \neg s'[0])) \lor \]

\[ 00 01 10 11 \]
Mutual Exclusion Example
Modeling (cont’d)

- Transition relation:

\[
T(s, s') := \ (\neg s[1] \land (s[0] \leftrightarrow \neg s'[0])) \lor \\
(\neg s[0] \land (s[1] \leftrightarrow \neg s'[1])) \lor 
\]

![Transition diagram](image-url)
Mutual Exclusion Example
Modeling (cont’d)

Transition relation:

\[ T(s, s') := (\neg s[1] \land (s[0] \iff \neg s'[0])) \lor \\
(\neg s[0] \land (s[1] \iff \neg s'[1])) \lor \\
(s[0] \land s[1] \land \neg s'[1] \land \neg s'[0]) \]
Mutual Exclusion Example
Adding Faulty Transition

- Suppose we add the fault transition $10 \rightarrow 11$ to the model, $M$: 

![Diagram of the model with the added transition](image-url)
Mutual Exclusion Example

Adding Faulty Transition

- Suppose we add the fault transition $10 \rightarrow 11$ to the model, $M$:

- Then we have the new transition relation:

  $$T_f(s, s') := T(s, s') \lor (s[1] \land \neg s[0] \land s'[1] \land s'[0])$$
Consider the safety property that at most one process can be in the critical section at any time, i.e.:

$$f = \square \neg p = \square \neg (s[1] \land s[0])$$
Consider the safety property that at most one process can be in the critical section at any time, i.e.:

\[ f = \Box \neg p = \Box \neg (s[1] \land s[0]) \]

Try to find a counterexample of this property, that is, a witness for \( \Diamond p \)

- If a witness exists, then \( M \not \models f \)
- If a witness cannot be found, then \( M \models_k f \) (i.e., property holds up to the bound \( k \))
Mutual Exclusion Example
LTL to Propositional Formula Translation

- Consider the bound $k = 2$. 
Consider the bound $k = 2$.

Unroll the transition relation:

$$[[M]]_2 : = I(s_0) \land \bigwedge_{l=0}^{k-1} T(s_l, s_{l+1})$$

$$= I(s_0) \land T_f(s_0, s_1) \land T_f(s_1, s_2)$$
Mutual Exclusion Example
LTL to Propositional Formula Translation

- Consider the bound \( k = 2 \).
- Unroll the transition relation:

\[
[[M]]_2 := I(s_0) \land \bigwedge_{l=0}^{k-1} T(s_l, s_{l+1})
\]

\[
= I(s_0) \land T_f(s_0, s_1) \land T_f(s_1, s_2)
\]

- The loop condition is:

\[
L_2 := \bigvee_{l=0}^{2} T_f(s_2, s_l)
\]
Translation for paths without loops is:

\[
\begin{align*}
[[\Diamond p]]_2^0 & := p(s_0) \lor [[\Diamond p]]_2^1 \\
[[\Diamond p]]_2^1 & := p(s_1) \lor [[\Diamond p]]_2^2 \\
[[\Diamond p]]_2^2 & := p(s_2) \lor [[\Diamond p]]_2^3 \\
[[\Diamond p]]_2^3 & := 0
\end{align*}
\]
Translation for paths without loops is:

\[
\begin{align*}
[[\Diamond p]]_0^2 & := p(s_0) \lor [[\Diamond p]]_1^2 \\
[[\Diamond p]]_1^2 & := p(s_1) \lor [[\Diamond p]]_2^2 \\
[[\Diamond p]]_2^2 & := p(s_2) \lor [[\Diamond p]]_3^2 \\
[[\Diamond p]]_3^2 & := 0
\end{align*}
\]

Substitute all intermediate terms to obtain:

\[
[[\Diamond p]]_2^2 := p(s_0) \lor p(s_1) \lor p(s_2)
\]
Translation for paths with loops is:

\[ 0[[\Diamond p]]_2^0 := p(s_0) \lor 0[[\Diamond p]]_2^1 \]
\[ 0[[\Diamond p]]_2^1 := p(s_1) \lor 0[[\Diamond p]]_2^2 \]
\[ 0[[\Diamond p]]_2^2 := p(s_2) \lor 0[[\Diamond p]]_2^0 \]
\[ 1[[\Diamond p]]_2^1 := p(s_1) \lor 1[[\Diamond p]]_2^2 \]
\[ 1[[\Diamond p]]_2^2 := p(s_2) \lor 1[[\Diamond p]]_2^1 \]
\[ 2[[\Diamond p]]_2^2 := p(s_2) \lor 2[[\Diamond p]]_2^2 \]
Mutual Exclusion Example
Check for a Witness

- Putting everything together:

\[
[[F, \Diamond p]]_2 := [[M]]_2 \land \left( \left( \neg L_2 \land [\Diamond p]_2^0 \right) \lor \bigvee_{l=0}^{2} \left( lL_2 \land l[\Diamond p]_2^l \right) \right)
\]
Mutual Exclusion Example

Check for a Witness

- Putting everything together:

$$[[F, \Diamond p]]_2 := [[M]]_2 \land \left( (\neg L_2 \land [[\Diamond p]]_2^0) \lor \bigvee_{l=0}^{2} (l L_2 \land l [[\Diamond p]]_2^l) \right)$$

- Since a finite path to a bad state is sufficient for falsifying a property, omit the loop condition.
Mutual Exclusion Example

Check for a Witness

- Putting everything together:

\[
[[F, \Diamond p]]_2 := [[M]]_2 \land \left( \left( -L_2 \land [[\Diamond p]]_2^0 \right) \lor \bigvee_{l=0}^2 \left( iL_2 \land i[[\Diamond p]]_2^l \right) \right)
\]

- Since a finite path to a bad state is sufficient for falsifying a property, omit the loop condition.

- This results in the formula:

\[
[[M, \Diamond p]]_2 := [[M]]_2 \land [[\Diamond p]]_2^0
= l(s_0) \land T_f(s_0, s_1) \land T_f(s_1, s_2)
\land (p(s_0) \lor p(s_1) \lor p(s_2))
\]
Mutual Exclusion Example
Check for a Witness (cont’d)

- \((s_0, s_1, s_2) = (00, 10, 11)\) satisfies \([M, \Diamond \rho]_2\)
- an initialized path that \textit{violates} the safety property
Outline

Introduction to Bounded Model Checking (BMC)
  BMC vs. BDD Model Checkers

A Quick Review of Model Checking Concepts
  Kripke Structures
  LTL Syntax and Semantics
  LTL Model Checking

Bounded Model Checking
  Bounded Path Syntax and Semantics
  Model Checking Problem Reduction

Reducing BMC to SAT
  Example of Checking Mutual Exclusion using BMC

Techniques for Completeness
  Completeness Threshold
  Liveness
  Induction
Suppose we have a model checking problem $M \models Ef$, where $f$ is the negated version of the property to be checked.
Completeness

- Suppose we have a model checking problem $M \models Ef$, where $f$ is the negated version of the property to be checked.
- Increment bound $k$ until a finite-length witness is found.
  - In this case, we are done and $M \models Ef$ (i.e., model does not satisfy the property).
Suppose we have a model checking problem $M \models Ef$, where $f$ is the negated version of the property to be checked.

Increment bound $k$ until a finite-length witness is found.

- In this case, we are done and $M \models Ef$ (i.e., model does not satisfy the property).

- If $M \not\models Ef$, how do we know when to terminate the BMC model checker?
Outline

Introduction to Bounded Model Checking (BMC)
  BMC vs. BDD Model Checkers

A Quick Review of Model Checking Concepts
  Kripke Structures
  LTL Syntax and Semantics
  LTL Model Checking

Bounded Model Checking
  Bounded Path Syntax and Semantics
  Model Checking Problem Reduction

Reducing BMC to SAT
  Example of Checking Mutual Exclusion using BMC

Techniques for Completeness
  Completeness Threshold
  Liveness
  Induction
Reachability Diameter

- For every finite state system $M$, a property $\rho$, and a translation scheme, there is a number $CT$ where the absence of errors up to cycle $CT$ proves that $M \models \rho$
Reachability Diameter

- For every finite state system $M$, a property $p$, and a translation scheme, there is a number $CT$ where the absence of errors up to cycle $CT$ proves that $M \models p$
- Completeness threshold is the minimal bound on $k$ for $\Box p$ required to reach all states and called the reachability diameter
Reachability Diameter

- For every finite state system $M$, a property $p$, and a translation scheme, there is a number $\mathcal{CT}$ where the absence of errors up to cycle $\mathcal{CT}$ proves that $M \models p$

- Completeness threshold is the minimal bound on $k$ for $\Box p$ required to reach all states and called the reachability diameter

**Definition 10**

The reachability diameter $rd(M)$ is the minimal number of steps required for reaching all reachable states, i.e.:

$$rd(M) := \min \left\{ i \mid \forall s_0, \ldots, s_n \bullet \exists s'_0, \ldots, s'_t, \ t \leq i \bullet \right.$$  

\[
l(s_0) \land \bigwedge_{j=0}^{n-1} T(s_j, s_{j+1}) \rightarrow \left( l(s'_0) \land \bigwedge_{j=0}^{t-1} T(s'_j, s'_{j+1}) \land s'_t = s_n \right) \right\}
\]
Reachability Diameter

- For every finite state system $M$, a property $p$, and a translation scheme, there is a number $CT$ where the absence of errors up to cycle $CT$ proves that $M \models p$
- Completeness threshold is the minimal bound on $k$ for $\square p$ required to reach all states and called the reachability diameter

Definition 10
The reachability diameter $rd(M)$ is the minimal number of steps required for reaching all reachable states, i.e.:

$$rd(M) := \min \left\{ i | \forall s_0, \ldots, s_n \bullet \exists s'_0, \ldots, s'_t, t \leq i \bullet \right.$$  

$$I(s_0) \land \bigwedge_{j=0}^{n-1} T(s_j, s_{j+1}) \rightarrow \left( I(s'_0) \land \bigwedge_{j=0}^{t-1} T(s'_j, s'_{j+1}) \land s'_t = s_n \right) \right\}$$
Determining $rd(M)$

- Worst case for $n = 2^{|V|}$, where $V$ is the set of variables defining the states of $M$
Determining \( rd(M) \)

- Worst case for \( n = 2^{|V|} \), where \( V \) is the set of variables defining the states of \( M \)
- Determine best \( n \)
  - Let \( n = i + 1 \). Check whether every state that can be reached in \( i + 1 \) can be reached sooner
Determining $rd(M)$

- Worst case for $n = 2^{|V|}$, where $V$ is the set of variables defining the states of $M$
- Determine best $n$
  - Let $n = i + 1$. Check whether every state that can be reached in $i + 1$ can be reached sooner

$$rd(M) := \min \left\{ i | \forall s_0, \ldots, s_{i+1} \cdot \exists s'_0, \ldots, s'_i, \bullet \right\}$$

$$I(s_0) \land \bigwedge_{j=0}^{i} T(s_j, s_{j+1}) \rightarrow \left( I(s'_0) \land \bigwedge_{j=0}^{i-1} T(s'_j, s'_{j+1}) \land \bigvee_{j=0}^{i} s'_j = s_{i+1} \right)$$
Determining \( rd(M) \)

- Worst case for \( n = 2^{|V|} \), where \( V \) is the set of variables defining the states of \( M \)
- Determine best \( n \)
  - Let \( n = i + 1 \). Check whether every state that can be reached in \( i + 1 \) can be reached sooner

\[
rd(M) := \min \left\{ i \mid \forall s_0, \ldots, s_{i+1} \exists s'_0, \ldots, s'_i \right\} 
\]

\[
l(s_0) \land \bigwedge_{j=0}^{i} T(s_j, s_{j+1}) \rightarrow \left( l(s'_0) \land \bigwedge_{j=0}^{i-1} T(s'_j, s'_{j+1}) \land \bigvee_{j=0}^{i} s'_j = s_{i+1} \right) \}
\]

- Alternation of quantifiers in the two previous expressions are hard to solve in practice
Recurrence Diameter for Reachability

- Approximate the reachability diameter instead
Recurrence Diameter for Reachability

- Approximate the reachability diameter instead

**Definition 11 (Recurrence Diameter for Reachability)**

The *recurrence diameter for reachability* with respect to a model $M$, denoted by $\text{rdr}(M)$, is the longest loop-free path in $M$ starting from an initial state:

$$
\text{rdr}(M) := \max \left\{ i \mid \exists s_0 \ldots s_i \bullet \\
I(s_0) \land \bigwedge_{j=0}^{i-1} T(s_j, s_{j+1}) \land \bigwedge_{j=0}^{i-1} \bigwedge_{k=j+1}^{i} s_j \neq s_k \right\}
$$
Recurrence Diameter for Reachability

- Approximate the reachability diameter instead

**Definition 11 (Recurrence Diameter for Reachability)**

The *recurrence diameter for reachability* with respect to a model $M$, denoted by $\text{rdr}(M)$, is the longest loop-free path in $M$ starting from an initial state:

$$
\text{rdr}(M) := \max \left\{ i \mid \exists s_0 \ldots s_i \cdot 
I(s_0) \land \bigwedge_{j=0}^{i-1} T(s_j, s_{j+1}) \land \bigwedge_{j=0}^{i-1} \bigwedge_{k=j+1}^{i} s_j \neq s_k \right\}
$$

- $\text{rdr}(M)$ is an over-approximation of $\text{rd}(M)$ because every shortest path is a loop-free path
Outline

Introduction to Bounded Model Checking (BMC)
  BMC vs. BDD Model Checkers

A Quick Review of Model Checking Concepts
  Kripke Structures
  LTL Syntax and Semantics
  LTL Model Checking

Bounded Model Checking
  Bounded Path Syntax and Semantics
  Model Checking Problem Reduction

Reducing BMC to SAT
  Example of Checking Mutual Exclusion using BMC

Techniques for Completeness
  Completeness Threshold
  Liveness
  Induction
Liveness

- If a proof for liveness exists, the proof can be established by examining all finite sequences of length $k$ starting from initial states.
Liveness

- If a proof for liveness exists, the proof can be established by examining all finite sequences of length $k$ starting from initial states.

Definition 12 (Translation for Liveness Properties)

$$[[M, \mathbf{A} \lozenge p]]_k := I(s_0) \land \bigwedge_{i=0}^{k-1} T(s_i, s_{i+1}) \rightarrow \bigvee_{i=0}^{k} p(s_i)$$
Liveness (cont’d)

Theorem 3

\[ M \models \mathbf{A} \diamond p \iff \exists k \cdot [[M, \mathbf{A} \diamond p]]_k \text{ is valid.} \]
Liveness (cont’d)

Theorem 3

\[ M \models A \Diamond p \iff \exists k \bullet [\lbrack M, A \Diamond p \rbrack]_k \text{ is valid.} \]

- Need to search for a \( k \) that makes the negation of \([\lbrack M, A \Diamond p \rbrack]_k\) unsatisfiable
Liveness (cont’d)

Theorem 3

\[ M \models A \lozenge p \iff \exists k \bullet [[M, A \lozenge p]]_k \text{ is valid.} \]

- Need to search for a \( k \) that makes the negation of \([ [M, A \lozenge p]]_k \) unsatisfiable
- Bound \( k \) needed for a proof represent length of longest sequence from an initial state without hitting a state where \( p \) holds
Liveness (cont’d)

Theorem 3

\[ M \models A \diamond p \iff \exists k \bullet [[M, A \diamond p]]_k \text{ is valid.} \]

- Need to search for a \( k \) that makes the negation of \([[M, A \diamond p]]_k \) unsatisfiable
- Bound \( k \) needed for a proof represent length of longest sequence from an initial state without hitting a state where \( p \) holds
- In BMC, we have semi-decision procedures for

\[ M \models E \Box \neg p \iff M \not\models A \diamond p \]

- \( \therefore \) either \( A \diamond p \) or \( E \Box \neg p \) must hold, one of the semi-decision procedures must terminate
Outline

Introduction to Bounded Model Checking (BMC)
   BMC vs. BDD Model Checkers

A Quick Review of Model Checking Concepts
   Kripke Structures
   LTL Syntax and Semantics
   LTL Model Checking

Bounded Model Checking
   Bounded Path Syntax and Semantics
   Model Checking Problem Reduction

Reducing BMC to SAT
   Example of Checking Mutual Exclusion using BMC

Techniques for Completeness
   Completeness Threshold
   Liveness
   Induction
Induction

- Inductive techniques can be used to make BMC complete for safety properties
Induction

- Inductive techniques can be used to make BMC complete for safety properties
- Proving that $M \models A\Box p$ by induction usually involves:
  - Manually finding a strengthening inductive invariant expression that is inductive and implies the property
Inductive techniques can be used to make BMC complete for safety properties

Proving that $M \models A\Box p$ by induction usually involves:
- Manually finding a strengthening inductive invariant - expression that is inductive and implies the property

Inductive proof:
- Base case
- Induction step
- Strengthening step
Show that inductive invariant $\phi$ holds in first $n$ steps by checking whether the following is unsatisfiable:

$$\exists s_0, \ldots, s_n \bullet l(s_0) \land \bigwedge_{i=0}^{n-1} T(s_i, s_{i+1}) \land \bigvee_{i=0}^k \neg \phi(s_i)$$
Prove Inductive Invariant Holds for First $n$ Steps

Show that inductive invariant $\phi$ holds in first $n$ steps by checking whether the following is unsatisfiable:

$$\exists s_0, \ldots, s_n \cdot I(s_0) \land \bigwedge_{i=0}^{n-1} T(s_i, s_{i+1}) \land \bigvee_{i=0}^{k} \neg \phi(s_i)$$

Base step is equivalent to searching for a counterexample to $\square p$
Inductive Step

- Prove induction step by showing that the following is unsatisfiable:

\[
\exists s_0, \ldots, s_{n+1} \bullet \bigwedge_{i=0}^{n} (\phi(s_i) \land T(s_i, s_{i+1})) \land \neg\phi(s_{n+1})
\]
Refining the Inductive Step

- Paths in $M$ restricted to contain distinct states
  - Preserves completeness of BMC for safety properties
  - A bad state is reachable (if it exists) is reachable via a simple path
Refining the Inductive Step

- Paths in $M$ restricted to contain distinct states
  - Preserves completeness of BMC for safety properties
  - A bad state is reachable (if it exists) is reachable via a simple path
- Sufficient to show that the following is unsatisfiable:

$$\exists s_0, \ldots, s_{n+1} \cdot \bigwedge_{j=0}^{n} \bigwedge_{k=j+1}^{n+1} (s_j \neq s_k) \wedge \bigwedge_{i=0}^{n} (\phi(s_i) \wedge T(s_i, s_{i+1})) \wedge \neg \phi(s_{n+1})$$
Strengthening Inductive Invariant Implies Property

- Establish that for an arbitrary $i$:

$$\forall s_i \, \phi(s_i) \rightarrow p(s_i)$$
References

A. Biere, A. Cimatti, E. Clarke, and Y. Zhu
Symbolic Model Checking without BDDs

A. Biere, A. Cimatti, E. Clarke, O. Strichman, and Y. Zhu
Bounded Model Checking