An Overview of Model Checking

Ma Li

Center for Logic, Language and Cognition
Philosophy Department
Peking University

November 23, 2009
The Need for Formal Methods

- informal techniques
  - simulation
  - testing
- formal techniques
  - model checking
The Need for Formal Methods

- informal techniques
  - simulation
  - testing
- formal techniques
  - model checking
The Need for Formal Methods

- informal techniques
  - simulation
  - testing
- formal techniques
  - model checking
The Need for Formal Methods

- informal techniques
  - simulation
  - testing
- formal techniques
  - model checking
The Need for Formal Methods

- informal techniques
  - simulation
  - testing
- formal techniques
  - model checking
Advantages that Model Checking Enjoys

- completely automatical
- terminate with true or a counterexample
- fast due to Partial specification
- logic used for specification has strong expressive power
Advantages that Model Checking Enjoys

- completely automatical
- terminate with true or a counterexample
- fast due to Partial specification
- logic used for specification has strong expressive power
Advantages that Model Checking Enjoys

- completely automatical
- terminate with true or a counterexample
- fast due to Partial specification
- logic used for specification has strong expressive power
Advantages that Model Checking Enjoys

- completely automatical
- terminate with true or a counterexample
- fast due to Partial specification
- logic used for specification has strong expressive power
disadvantages that Model Checking suffers from

- state space explosion
Outline

1 Introduction
   - The Need for Formal Methods
   - Advantages that Model Checking Enjoys
   - disadvantages that Model Checking suffers from

2 How to Do Model Checking

3 Temporal Logic
   - The Computation Tree Logic $CTL^*$
     - syntax
     - semantics
   - CTL and LTL

4 Breakthroughs on State Space Explosion
   - Ordered Binary Decision Diagram
   - Symbolic Model Checking
the first step: convert a design into a formalism, sometimes use abstraction.

the second step: specify the properties that the designs must satisfy.

the third step: verify whether the model obtained in the first step satisfied the specification got in the second step.

error trace may result from both incorrect modeling and incorrect specification.
How to Do Model Checking

- the first step: convert a design into a formalism, sometimes use abstraction.
- the second step: specify the properties that the designs must satisfy.
- the third step: verify whether the model obtained in the first step satisfied the specification got in the second step.
- error trace may result from both incorrect modeling and incorrect specification.
How to Do Model Checking

- the first step: convert a design into a formalism, sometimes use abstraction.
- the second step: specify the properties that the designs must satisfy.
- the third step: verify whether the model obtained in the first step satisfied the specification got in the second step.
- error trace may result from both incorrect modeling and incorrect specification.
How to Do Model Checking

- the first step: convert a design into a formalism, sometimes use abstraction.
- the second step: specify the properties that the designs must satisfy.
- the third step: verify whether the model obtained in the first step satisfied the specification got in the second step.
- error trace may result from both incorrect modeling and incorrect specification.
Outline

1. Introduction
   - The Need for Formal Methods
   - Advantages that Model Checking Enjoys
   - Disadvantages that Model Checking suffers from

2. How to Do Model Checking

3. Temporal Logic
   - The Computation Tree Logic $CTL^*$
     - Syntax
     - Semantics
   - $CTL^*$ and LTL

4. Breakthroughs on State Space Explosion
   - Ordered Binary Decision Diagram
   - Symbolic Model Checking
syntax

- $\mathbf{X}("next\ time")$ requires that a property holds in the second state of the path.
- The $\mathbf{F}("eventually\"\ or\ "in\ the\ future")$ operator is used to assert that a property will hold at some state on the path.
- $\mathbf{G}("always\"\ or\ "globally")$ specifies that a property holds at every state on the path.
The Computation Tree Logic $CTL^*$

### Syntax

- $X$ ("next time") requires that a property holds in the second state of the path.

- The $F$ ("eventually" or "in the future") operator is used to assert that a property will hold at some state on the path.

- $G$ ("always" or "globally") specifies that a property holds at every state on the path.
X("next time") requires that a property holds in the second state of the path.

The F("eventually" or "in the future") operator is used to assert that a property will hold at some state on the path.

G("always" or "globally") specifies that a property holds at every state on the path.
The $U$ ("until") operator is a bit more complicated since it is used to combine two properties. It holds if there is a state on the path where the second property holds, and at every preceding state on the path, the first property holds.

$R$("release") is the logical dual of the $U$ operator. It requires that the second property holds along the path up to and including the first state where the first property hold. However, the first property is not required to hold eventually.
The \textbf{U} ("until") operator is a bit more complicated since it is used to combine two properties. It holds if there is a state on the path where the second property holds, and at every preceding state on the path, the first property holds.

\textbf{R}"release") is the logical dual of the \textbf{U} operator. It requires that the second property holds along the path up to and including the first state where the first property hold. However, the first property is not required to hold eventually.
Introduction
How to Do Model Checking
Temporal Logic
Breakthroughs on State Space Explosion
Reference

Introduction
How to Do Model Checking
Temporal Logic
Breakthroughs on State Space Explosion
Reference

The Computation Tree Logic $CTL^*$
CTL and LTL

syntax

- If $p \in AP$, then $p$ is a state formula.
- If $f$ and $g$ are state formulas, then $\neg f$, $f \lor g$ and $f \land g$ are state formulas.
- $f$ is a path formula, the $Ef$ and $Af$ are state formulas.

Ma Li PKU
An Overview of Model Checking
syntax

- If $p \in AP$, then $p$ is a state formula.
- If $f$ and $g$ are state formulas, then $\neg f$, $f \lor g$ and $f \land g$ are state formulas.
- $f$ is a path formula, the $Ef$ and $Af$ are state formulas.
syntax

- If $p \in AP$, then $p$ is a state formula.
- If $f$ and $g$ are state formulas, then $\neg f$, $f \lor g$ and $f \land g$ are state formulas.
- $f$ is a path formula, the $Ef$ and $Af$ are state formulas.
It is easy to see that the operators $\lor$, $\neg$, $X$, $U$, and $E$ are sufficient to express any other $CTL^*$ formulas.

- $f \land g \equiv \neg(\neg f \lor \neg g)$
- $fRg \equiv \neg(\neg fU\neg g)$
- $Ff \equiv True U f$
- $Gf \equiv \neg F \neg f$
- $A(f) \equiv \neg E \neg (f)$
It is easy to see that the operators ∨, ¬, X, U, and E are sufficient to express any other CTL* formulas.

- \( f \land g \equiv \neg(\neg f \lor \neg g) \)
- \( f \mathcal{R} g \equiv \neg(\neg f \mathcal{U} \neg g) \)
- \( Ff \equiv True \mathcal{U} f \)
- \( Gf \equiv \neg F \neg f \)
- \( A(f) \equiv \neg E \neg (f) \)
syntax

It is easy to see that the operators $\lor$, $\neg$, $X$, $U$, and $E$ are sufficient to express any other $CTL^*$ formulas.

- $f \land g \equiv \neg(\neg f \lor \neg g)$
- $fRg \equiv \neg(\neg fU\neg g)$
- $Ff \equiv True \ U \ f$
- $Gf \equiv \neg F \ \neg f$
- $A(f) \equiv \neg E \ \neg (f)$
It is easy to see that the operators \( \lor, \neg, X, U, \text{ and } E \) are sufficient to express any other \( CTL^* \) formulas.

- \( f \land g \equiv \neg(\neg f \lor \neg g) \)
- \( f R g \equiv \neg(\neg f U \neg g) \)
- \( Ff \equiv True U f \)
- \( Gf \equiv \neg F \neg f \)
- \( A(f) \equiv \neg E \neg (f) \)
It is easy to see that the operators $\lor$, $\neg$, $X$, $U$, and $E$ are sufficient to express any other $CTL^*$ formulas.

- $f \land g \equiv \neg(\neg f \lor \neg g)$
- $fRg \equiv \neg(\neg fU\neg g)$
- $Ff \equiv True \ U \ f$
- $Gf \equiv \neg F \neg f$
- $A(f) \equiv \neg E \neg (f)$
Two additional rules are needed to specify the syntax of path formulas:

- If $f$ is a state formula, then $f$ is also a path formula.
- If $f$ and $g$ are path formulas, then $\neg f$, $f \lor g$, $f \land g$, $Xf$, $Ff$, $Gf$, $fUg$, and $fRg$ are path formulas.

$CTL^*$ is the set of state formulas generated by the above rules.
Two additional rules are needed to specify the syntax of path formulas:

- If $f$ is a state formula, then $f$ is also a path formula.
- If $f$ and $g$ are path formulas, then $\neg f$, $f \lor g$, $f \land g$, $Xf$, $Ff$, $Gf$, $fUg$, and $fRg$ are path formulas.

$CTL^*$ is the set of state formulas generated by the above rules.
Introduction
How to Do Model Checking
Temporal Logic
Breakthroughs on State Space Explosion
Reference

The Computation Tree Logic $CTL^*$

CTL and LTL

semantics

- $M, s \models p \iff p \in L(s)$.
- $M, s \models \neg f_1 \iff M, s \not\models f_1$
- $M, s \models f_1 \lor f_2 \iff M, s \models f_1 \text { or } M, s \models f_2$
- $M, s \models f_1 \land f_2 \iff M, s \models f_1 \text { and } M, s \models f_2$
- $M, s \models Eg_1 \iff \text{there is a path } \pi \text{ from } s \text{ such that } M, \pi \models g_1$. 

Ma Li PKU
An Overview of Model Checking
semantics

- $M, s \models p \iff p \in L(s)$.
- $M, s \models \neg f_1 \iff M, s \not\models f_1$
- $M, s \models f_1 \lor f_2 \iff M, s \models f_1 \text{ or } M, s \models f_2$
- $M, s \models f_1 \land f_2 \iff M, s \models f_1 \text{ and } M, s \models f_2$
- $M, s \models \text{E}g_1 \iff \text{there is a path } \pi \text{ from } s \text{ such that } M, \pi \models g_1$. 
semantics

- $M, s \models p \iff p \in L(s)$.
- $M, s \models \neg f_1 \iff M, s \not\models f_1$
- $M, s \models f_1 \lor f_2 \iff M, s \models f_1 \text{ or } M, s \models f_2$
- $M, s \models f_1 \land f_2 \iff M, s \models f_1 \text{ and } M, s \models f_2$
- $M, s \models Eg_1 \iff \text{there is a path } \pi \text{ from } s \text{ such that } M, \pi \models g_1$. 
Introduction
How to Do Model Checking
Temporal Logic
Breakthroughs on State Space Explosion
Reference

The Computation Tree Logic $CTL^*$

CTL and LTL

semantics

- $M, s \models p \iff p \in L(s)$.
- $M, s \models \neg f_1 \iff M, s \not\models f_1$
- $M, s \models f_1 \lor f_2 \iff M, s \models f_1 \text{ or } M, s \models f_2$
- $M, s \models f_1 \land f_2 \iff M, s \models f_1 \text{ and } M, s \models f_2$
- $M, s \models E g_1 \iff$ there is a path $\pi$ from $s$ such that $M, \pi \models g_1$. 
Introduction
How to Do Model Checking
Temporal Logic
Breakthroughs on State Space Explosion
Reference

The Computation Tree Logic $CTL^*$
CTL and LTL

semantics

- $M, s \models p \iff p \in L(s)$.
- $M, s \models \neg f_1 \iff M, s \not\models f_1$
- $M, s \models f_1 \lor f_2 \iff M, s \models f_1$ or $M, s \models f_2$
- $M, s \models f_1 \land f_2 \iff M, s \models f_1$ and $M, s \models f_2$
- $M, s \models Eg_1 \iff$ there is a path $\pi$ from $s$ such that $M, \pi \models g_1$. 

An Overview of Model Checking

Ma Li  PKU  🌟
semantics

- $M, s \models Ag_1$ $\iff$ for every path $\pi$ starting from $s$, $M, \pi \models g_1$.
- $M, \pi \models f_1$ $\iff$ $s$ is the first state of $\pi$ and $M, s \models f_1$
- $M, s \models \neg g_1$ $\iff$ $M, s \not\models g_1$
- $M, \pi \models g_1 \lor g_2$ $\iff$ $M, \pi \models g_1$ or $M, \pi \models g_2$
- $M, \pi \models g_1 \land g_2$ $\iff$ $M, s \models g_1$ and $M, s \models g_2$
semantics

- \( M, s \models \mathsf{Ag}_1 \iff \text{for every path } \pi \text{ starting from } s, \ M, \pi \models g_1. \)
- \( M, \pi \models f_1 \iff s \text{ is the first state of } \pi \text{ and } M, s \models f_1 \)
- \( M, s \models \neg g_1 \iff M, s \nvDash g_1 \)
- \( M, \pi \models g_1 \lor g_2 \iff M, \pi \models g_1 \text{ or } M, \pi \models g_2 \)
- \( M, \pi \models g_1 \land g_2 \iff M, s \models g_1 \text{ and } M, s \models g_2 \)
<table>
<thead>
<tr>
<th>$M, s \models \mathbf{A}g_1$</th>
<th>$\iff$ for every path $\pi$ starting from $s$, $M, \pi \models g_1$.</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M, \pi \models f_1$</td>
<td>$\iff$ $s$ is the first state of $\pi$ and $M, s \models f_1$</td>
</tr>
<tr>
<td>$M, s \models \neg g_1$</td>
<td>$\iff M, s \not\models g_1$</td>
</tr>
<tr>
<td>$M, \pi \models g_1 \lor g_2$</td>
<td>$\iff M, \pi \models g_1$ or $M, \pi \models g_2$</td>
</tr>
<tr>
<td>$M, \pi \models g_1 \land g_2$</td>
<td>$\iff M, s \models g_1$ and $M, s \models g_2$</td>
</tr>
</tbody>
</table>
semantics

- $M, s \models \mathbf{A}g_1 \iff$ for every path $\pi$ starting from $s$, $M, \pi \models g_1$.
- $M, \pi \models f_1 \iff s$ is the first state of $\pi$ and $M, s \models f_1$
- $M, s \models \neg g_1 \iff M, s \not\models g_1$
- $M, \pi \models g_1 \lor g_2 \iff M, \pi \models g_1$ or $M, \pi \models g_2$
- $M, \pi \models g_1 \land g_2 \iff M, s \models g_1$ and $M, s \models g_2$
semantics

- \( M, s \models Ag_1 \iff \text{for every path } \pi \text{ starting from } s, \ M, \pi \models g_1. \)
- \( M, \pi \models f_1 \iff s \text{ is the first state of } \pi \text{ and } M, s \models f_1. \)
- \( M, s \models \neg g_1 \iff M, s \not\models g_1. \)
- \( M, \pi \models g_1 \lor g_2 \iff M, \pi \models g_1 \text{ or } M, \pi \models g_2. \)
- \( M, \pi \models g_1 \land g_2 \iff M, s \models g_1 \text{ and } M, s \models g_2. \)
semantics

- $M, \pi \models X g_1 \iff M, \pi^1 \models g_1$.
- $M, \pi \models F g_1 \iff \text{there exists a } k \geq 0 \text{ such that } M, \pi^k \models g_1$.
- $M, \pi \models G g_1 \iff \text{for all } i \geq 0, M, \pi^i \models g_1$.
- $M, \pi \models g_1 U g_2 \iff \text{there exists a } k \geq 0 \text{ such that } M, \pi^k \models g_2 \text{ and for all } 0 \leq j < k, M, \pi^j \models g_1$.
- $M, \pi \models g_1 R g_2 \iff \text{for all } j \geq 0, \text{ if for every } i < j, M, \pi^i \text{ un-satisfies } g_1 \text{ then } M, \pi^j \models g_2$. 
Introduction
How to Do Model Checking
Temporal Logic
Breakthroughs on State Space Explosion
Reference

The Computation Tree Logic $CTL^*$

CTL and LTL

semantics

- $M, \pi \models X g_1 \iff M, \pi^1 \models g_1$.
- $M, \pi \models F g_1 \iff$ there exists a $k \geq 0$ such that $M, \pi^k \models g_1$.
- $M, \pi \models G g_1 \iff$ for all $i \geq 0$, $M, \pi^i \models g_1$.
- $M, \pi \models g_1 U g_2 \iff$ there exists a $k \geq 0$ such that $M, \pi^k \models g_2$ and for all $0 \leq j < k$, $M, \pi^j \models g_1$.
- $M, \pi \models g_1 R g_2 \iff$ for all $j \geq 0$, if for every $i < j$, $M, \pi^i$ un-satisfies $g_1$ then $M, \pi^j \models g_2$. 
semantics

- $M, \pi \models Xg_1 \iff M, \pi^1 \models g_1$.
- $M, \pi \models Fg_1 \iff$ there exists a $k \geq 0$ such that $M, \pi^k \models g_1$.
- $M, \pi \models Gg_1 \iff$ for all $i \geq 0$, $M, \pi^i \models g_1$.
- $M, \pi \models g_1 U g_2 \iff$ there exists a $k \geq 0$ such that $M, \pi^k \models g_2$ and for all $0 \leq j < k$, $M, \pi^j \models g_1$.
- $M, \pi \models g_1 R g_2 \iff$ for all $j \geq 0$, if for every $i < j$, $M, \pi^i$ un-satisfies $g_1$ then $M, \pi^j \models g_2$.
Introduction
How to Do Model Checking
Temporal Logic
Breakthroughs on State Space Explosion
Reference

The Computation Tree Logic $CTL^*$

CTL and LTL

semantics

- $M, \pi \models Xg_1 \iff M, \pi^1 \models g_1$
- $M, \pi \models Fg_1 \iff$ there exists a $k \geq 0$ such that $M, \pi^k \models g_1$
- $M, \pi \models Gg_1 \iff$ for all $i \geq 0$, $M, \pi^i \models g_1$
- $M, \pi \models g_1 U g_2 \iff$ there exists a $k \geq 0$ such that
  $M, \pi^k \models g_2$ and for all $0 \leq j < k$, $M, \pi^j \models g_1$
- $M, \pi \models g_1 R g_2 \iff$ for all $j \geq 0$, if for every $i < j$, $M, \pi^i$
  un-satisfies $g_1$ then $M, \pi^j \models g_2$

Ma Li  PKU  An Overview of Model Checking
The Computation Tree Logic $CTL^*$

CTL and LTL

Ma Li PKU

An Overview of Model Checking
CTL is the subset of $CTL^*$ that is obtained by restricting the syntax of path formulas using the following rule.

- If $f$ and $g$ are state formulas, then $Xf$, $Ff$, $Gf$, $fUg$ and $fRg$ are path formulas.
An LTL path formula is either:

- If \( p \in AP \), then \( p \) is a path formula.
- If \( f \) and \( g \) are path formulas, then \( \neg f, f \lor g, f \land g, Xf, Ff, Gf, fUg, \) and \( fRg \) are path formulas.
semantics

An LTL path formula is either £^0

- If \( p \in AP \), then \( p \) is a path formula.
- If \( f \) and \( g \) are path formulas, then \( \neg f \), \( f \lor g \), \( f \land g \), \( Xf \), \( Ff \), \( Gf \), \( fUg \), and \( fRg \) are path formulas.
Most of the specifications in the following part of this article will be written in the logic CTL. There are ten basic CTL operators:

- **AX** and **EX**,
- **AF** and **EF**
- **AG** and **EG**
- **AU** and **EU**
- **AR** and **ER**
Most of the specifications in the following part of this article will be written in the logic CTL. There are ten basic CTL operators:

- **AX** and **EX**,
- **AF** and **EF**
- **AG** and **EG**
- **AU** and **EU**
- **AR** and **ER**
semantics

Most of the specifications in the following part of this article will be written in the logic CTL. There are ten basic CTL operators:

- AX and EX,
- AF and EF
- AG and EG
- AU and EU
- AR and ER
Most of the specifications in the following part of this article will be written in the logic CTL. There are ten basic CTL operators:

- $\text{AX}$ and $\text{EX}$,
- $\text{AF}$ and $\text{EF}$,
- $\text{AG}$ and $\text{EG}$,
- $\text{AU}$ and $\text{EU}$,
- $\text{AR}$ and $\text{ER}$.
Most of the specifications in the following part of this article will be written in the logic CTL. There are ten basic CTL operators:

- AX and EX,
- AF and EF
- AG and EG
- AU and EU
- AR and ER

**Temporal Logic**

Introduction

How to Do Model Checking

The Computation Tree Logic $CTL^*$

 CTL and LTL

**Breakthroughs on State Space Explosion**

Reference

---

Ma Li PKU

An Overview of Model Checking
Each of the ten operators can be expressed in terms of three operators $\textbf{EX}$, $\textbf{EG}$ and $\textbf{EU}$:

- $\textbf{AX} f = \neg \textbf{EX}(\neg f)$
- $\textbf{EF} f = E[\textbf{True} \textbf{U} f]$  
- $\textbf{AG} f = \neg \textbf{EF}(\neg f)$
- $A[fUg] = \neg E[\neg gU(\neg f \land \neg g)] \land \neg \textbf{EG} \neg g$
- $A[fRg] = \neg E[\neg fU \neg g]$
Each of the ten operators can be expressed in terms of three operators $\text{EX}$, $\text{EG}$ and $\text{EU}$:

- $\text{AX}f = \neg \text{EX}(\neg f)$
- $\text{EF}f = \text{E}[\text{True} \cup f]$
- $\text{AG}f = \neg \text{EF}(\neg f)$
- $\text{A}[f \cup g] = \neg \text{E}[\neg g \cup (\neg f \land \neg g)] \land \neg \text{EG} \neg g$
- $\text{A}[f \cup R g] = \neg \text{E}[\neg f \cup \neg g]$
Each of the ten operators can be expressed in terms of three operators $\text{EX}$, $\text{EG}$ and $\text{EU}$:

- $\text{AX} f = \neg \text{EX} (\neg f)$
- $\text{EF} f = \text{E} [\text{True} \cup f]$
- $\text{AG} f = \neg \text{EF} (\neg f)$
- $\text{A} [f \cup g] = \neg \text{E} [\neg g \cup (\neg f \land \neg g)] \land \neg \text{EG} \neg g$
- $\text{A} [f \text{R} g] = \neg \text{E} [\neg f \cup \neg g]$
Each of the ten operators can be expressed in terms of three operators $\text{EX}$, $\text{EG}$ and $\text{EU}$:

- $\text{AX}f = \neg \text{EX}(\neg f)$
- $\text{EF}f = \text{E}[\text{True}\text{U}f]$  
- $\text{AG}f = \neg \text{EF}(\neg f)$
- $\text{A}[f\text{U}g] = \neg \text{E}[\neg g\text{U}(\neg f \land \neg g)] \land \neg \text{EG}\neg g$
- $\text{A}[f\text{R}g] = \neg \text{E}[\neg f\text{U}\neg g]$
Each of the ten operators can be expressed in terms of three operators $\text{EX}$, $\text{EG}$ and $\text{EU}$:

- $\text{AX}f = \neg \text{EX}(\neg f)$
- $\text{EF}f = \text{E}[\text{True} \text{U} f]$
- $\text{AG}f = \neg \text{EF}(\neg f)$
- $\text{A}[f \text{U} g] = \neg \text{E}[\neg g \text{U}(\neg f \land \neg g)] \land \neg \text{EG} \neg g$
- $\text{A}[f \text{R} g] = \neg \text{E}[\neg f \text{U} \neg g]$
Outline

1. Introduction
   - The Need for Formal Methods
   - Advantages that Model Checking Enjoys
   - Disadvantages that Model Checking suffers from

2. How to Do Model Checking

3. Temporal Logic
   - The Computation Tree Logic $CTL^*$
     - Syntax
     - Semantics
   - $CTL$ and $LTL$

4. Breakthroughs on State Space Explosion
   - Ordered Binary Decision Diagram
   - Symbolic Model Checking
Every binary decision diagram $B$ with root $\nu$ determines a boolean function $f_{\nu}(x_1, \ldots, x_n)$ in the following manner:

- If $\nu$ is a terminal vertex
  - If $\text{value}(\nu) = 1$ then $f_{\nu}(x_1, \ldots, x_n) = 1$.
  - If $\text{value}(\nu) = 0$ then $f_{\nu}(x_1, \ldots, x_n) = 0$.

- If $\nu$ is a nonterminal vertex with $\text{var}(\nu) = x_i$ then $f_{\nu}$ is the function
  $$f_{\nu}(x_1, \ldots, x_n) = \neg x_i \land f_{\text{low}(\nu)}(x_1, \ldots, x_n) \lor x_i \land f_{\text{high}(\nu)}(x_1, \ldots, x_n)$$
Ordered Binary Decision Diagram

Every binary decision diagram $B$ with root $\nu$ determines a boolean function $f_{\nu}(x_1, \ldots, x_n)$ in the following manner:

- If $\nu$ is a terminal vertex
  - If $\text{value}(\nu) = 1$ then $f_{\nu}(x_1, \ldots, x_n) = 1$.
  - If $\text{value}(\nu) = 0$ then $f_{\nu}(x_1, \ldots, x_n) = 0$.

- $\nu$ is a nonterminal vertex with $\text{var}(\nu) = x_i$ then $f_{\nu}$ is the function
  - $f_{\nu}(x_1, \ldots, x_n) = (\neg x_i \land f_{\text{low}(\nu)}(x_1, \ldots, x_n)) \lor (x_i \land f_{\text{high}(\nu)}(x_1, \ldots, x_n))$.
Every binary decision diagram $B$ with root $v$ determines a boolean function $f_v(x_1, \ldots, x_n)$ in the following manner:

- If $v$ is a terminal vertex
  - If $\text{value}(v) = 1$ then $f_v(x_1, \ldots, x_n) = 1$.
  - If $\text{value}(v) = 0$ then $f_v(x_1, \ldots, x_n) = 0$.

- If $v$ is a nonterminal vertex with $\text{var}(v) = x_i$ then $f_v$ is the function
  $$f_v(x_1, \ldots, x_n) = (\neg x_i \land f_{\text{low}}(v)(x_1, \ldots, x_n)) \lor (x_i \land f_{\text{high}}(v)(x_1, \ldots, x_n))$$
Every binary decision diagram B with root $\nu$ determines a boolean function $f_{\nu}(x_1, \ldots, x_n)$ in the following manner:

- If $\nu$ is a terminal vertex
  - If $\text{value}(\nu) = 1$ then $f_{\nu}(x_1, \ldots, x_n) = 1$.
  - If $\text{value}(\nu) = 0$ then $f_{\nu}(x_1, \ldots, x_n) = 0$.

- If $\nu$ is a nonterminal vertex with $\text{var}(\nu) = x_i$ then $f_{\nu}$ is the function
  
  \[ f_{\nu}(x_1, \ldots, x_n) = (\neg x_i \land f_{\text{low}(\nu)}(x_1, \ldots, x_n)) \lor (x_i \land f_{\text{high}(\nu)}(x_1, \ldots, x_n)) \]
Every binary decision diagram $B$ with root $v$ determines a boolean function $f_v(x_1, \ldots, x_n)$ in the following manner:

- If $v$ is a terminal vertex
  - If $\text{value}(v) = 1$ then $f_v(x_1, \ldots, x_n) = 1$.
  - If $\text{value}(v) = 0$ then $f_v(x_1, \ldots, x_n) = 0$.
- $v$ is a nonterminal vertex with $\text{var}(v) = x_i$ then $f_v$ is the function
  
  $$f_v(x_1, \ldots, x_n) = (\neg x_i \land f_{\text{low}(v)}(x_1, \ldots, x_n)) \lor (x_i \land f_{\text{high}(v)}(x_1, \ldots, x_n))$$

Ma Li PKU

An Overview of Model Checking
Ordered Binary Decision Diagram

- canonical representation for boolean functions by restricting OBDDs.
- represent kripke structures by OBDDs
Ordered Binary Decision Diagram

- canonical representation for boolean functions by restricting OBDDs.
- represent kripke structures by OBDDs
Symbolic Model Checking

mainly focus on the symbolic model checking algorithm for CTL

- Fixpoint Representations
  - $AF \ f = \mu Z. \ f_1 \lor AXZ$
  - $EF \ f = \mu Z. \ f_1 \lor EXZ$
  - $AG \ f = \nu Z. \ f_1 \land AXZ$
  - $EG \ f = \nu Z. \ f_1 \land EXZ$
  - $A[f_1 \lor f_2] = \mu Z. \ f_2 \lor (f_1 \land AXZ)$
  - $E[f_1 \lor f_2] = \mu Z. \ f_2 \lor (f_1 \land EXZ)$
  - $A[f_1 \land f_2] = \nu Z. \ f_2 \land (f_1 \lor AXZ)$
  - $A[f_1 \land f_2] = \nu Z. \ f_2 \land (f_1 \lor EXZ)$
Symbolic Model Checking

mainly focus on the symbolic model checking algorithm for CTL

- Fixpoint Representations
  - \( \textbf{AF} f = \mu Z. f_1 \lor \text{AX} Z \)
  - \( \textbf{EF} f = \mu Z. f_1 \lor \text{EX} Z \)
  - \( \textbf{AG} f = \nu Z. f_1 \land \text{AX} Z \)
  - \( \textbf{EG} f = \nu Z. f_1 \land \text{EX} Z \)
  - \( A[ f_1 \cup f_2 ] = \mu Z. f_2 \lor (f_1 \land \text{AX} Z) \)
  - \( E[ f_1 \cup f_2 ] = \mu Z. f_2 \lor (f_1 \land \text{EX} Z) \)
  - \( A[ f_1 \lor f_2 ] = \nu Z. f_2 \land (f_1 \lor \text{AX} Z) \)
  - \( A[ f_1 \land f_2 ] = \nu Z. f_2 \land (f_1 \lor \text{EX} Z) \)
Symbolic Model Checking

mainly focus on the symbolic model checking algorithm for CTL

- **Fixpoint Representations**
- \( \text{AF } f = \mu Z. f_1 \lor AXZ \)
- \( \text{EF } f = \mu Z. f_1 \lor EXZ \)
- \( \text{AG } f = \nu Z. f_1 \land AXZ \)
- \( \text{EG } f = \nu Z. f_1 \land EXZ \)
- \( \text{A}[f_1 \cup f_2] = \mu Z. f_2 \lor (f_1 \land AXZ) \)
- \( \text{E}[f_1 \cup f_2] = \mu Z. f_2 \lor (f_1 \land EXZ) \)
- \( \text{A}[f_1 \lor f_2] = \nu Z. f_2 \land (f_1 \lor AXZ) \)
- \( \text{A}[f_1 \land f_2] = \nu Z. f_2 \land (f_1 \lor EXZ) \)
Symbolic Model Checking

mainly focus on the symbolic model checking algorithm for CTL

- Fixpoint Representations
- \( \text{AF } f = \mu Z \cdot f_1 \lor AXZ \)
- \( \text{EF } f = \mu Z \cdot f_1 \lor EXZ \)
- \( \text{AG } f = \nu Z \cdot f_1 \land AXZ \)
- \( \text{EG } f = \nu Z \cdot f_1 \land EXZ \)
- \( A[f_1 U f_2 ] = \mu Z \cdot f_2 \lor (f_1 \land AXZ) \)
- \( E[f_1 U f_2 ] = \mu Z \cdot f_2 \lor (f_1 \land EXZ) \)
- \( A[f_1 R f_2 ] = \nu Z \cdot f_2 \land (f_1 \lor AXZ) \)
- \( A[f_1 R f_2 ] = \nu Z \cdot f_2 \land (f_1 \lor EXZ) \)
Symbolic Model Checking

mainly focus on the symbolic model checking algorithm for CTL

- Fixpoint Representations
- $\text{AF } f = \mu Z. f_1 \lor \text{AX} Z$
- $\text{EF } f = \mu Z. f_1 \lor \text{EX} Z$
- $\text{AG } f = \nu Z. f_1 \land \text{AX} Z$
- $\text{EG } f = \nu Z. f_1 \land \text{EX} Z$
- $\text{A}[f_1 \cup f_2 ] = \mu Z. f_2 \lor (f_1 \land \text{AX} Z)$
- $\text{E}[f_1 \cup f_2 ] = \mu Z. f_2 \lor (f_1 \land \text{EX} Z)$
- $\text{A}[f_1 \cap f_2 ] = \nu Z. f_2 \land (f_1 \lor \text{AX} Z)$
- $\text{A}[f_1 \cap f_2 ] = \nu Z. f_2 \land (f_1 \lor \text{EX} Z)$
Symbolic Model Checking

mainly focus on the symbolic model checking algorithm for CTL

- Fixpoint Representations
- \( AF \ f = \mu Z. \ f_1 \lor AXZ \)
- \( EF \ f = \mu Z. \ f_1 \lor EXZ \)
- \( AG \ f = \nu Z. \ f_1 \land AXZ \)
- \( EG \ f = \nu Z. \ f_1 \land EXZ \)
- \( A[f_1 U f_2] = \mu Z. \ f_2 \lor (f_1 \land AXZ) \)
- \( E[f_1 U f_2] = \mu Z. \ f_2 \lor (f_1 \land EXZ) \)
- \( A[f_1 R f_2] = \nu Z. \ f_2 \land (f_1 \lor AXZ) \)
- \( A[f_1 R f_2] = \nu Z. \ f_2 \land (f_1 \lor EXZ) \)
Symbolic Model Checking

mainly focus on the symbolic model checking algorithm for CTL

- **Fixpoint Representations**
  - \( AF \ f = \mu Z. \ f_1 \lor AXZ \)
  - \( EF \ f = \mu Z. \ f_1 \lor EXZ \)
  - \( AG \ f = \nu Z. \ f_1 \land AXZ \)
  - \( EG \ f = \nu Z. \ f_1 \land EXZ \)
  - \( A[f_1 \ U \ f_2] = \mu Z. \ f_2 \lor (f_1 \land AXZ) \)
  - \( E[f_1 \ U \ f_2] = \mu Z. \ f_2 \lor (f_1 \land EXZ) \)
  - \( A[f_1 \ R \ f_2] = \nu Z. \ f_2 \land (f_1 \lor AXZ) \)
  - \( A[f_1 \ R \ f_2] = \nu Z. \ f_2 \land (f_1 \lor EXZ) \)
Symbolic Model Checking

mainly focus on the symbolic model checking algorithm for CTL

- **Fixpoint Representations**
  - $AF \ f = \mu Z. \ f_1 \lor A X Z$
  - $EF \ f = \mu Z. \ f_1 \lor E X Z$
  - $AG \ f = \nu Z. \ f_1 \land A X Z$
  - $EG \ f = \nu Z. \ f_1 \land E X Z$
  - $A[f_1 \ U f_2] = \mu Z. \ f_2 \lor (f_1 \land A X Z)$
  - $E[f_1 \ U f_2] = \mu Z. \ f_2 \lor (f_1 \land E X Z)$
  - $A[f_1 \ R f_2] = \nu Z. \ f_2 \land (f_1 \lor A X Z)$
  - $A[f_1 \ R f_2] = \nu Z. \ f_2 \land (f_1 \lor E X Z)$
Symbolic Model Checking

mainly focus on the symbolic model checking algorithm for CTL

- Fixpoint Representations
- $\text{AF } f = \mu Z. f_1 \lor \text{AXZ}$
- $\text{EF } f = \mu Z. f_1 \lor \text{EXZ}$
- $\text{AG } f = \nu Z. f_1 \land \text{AXZ}$
- $\text{EG } f = \nu Z. f_1 \land \text{EXZ}$
- $\text{A}[f_1 \text{ U } f_2 ] = \mu Z. f_2 \lor (f_1 \land \text{AXZ})$
- $\text{E}[f_1 \text{ U } f_2 ] = \mu Z. f_2 \lor (f_1 \land \text{EXZ})$
- $\text{A}[f_1 \text{ R } f_2 ] = \nu Z. f_2 \land (f_1 \lor \text{AXZ})$
- $\text{A}[f_1 \text{ R } f_2 ] = \nu Z. f_2 \land (f_1 \lor \text{EXZ})$
If $f$ is an atomic proposition $a$, then $\text{Check}(f)$ is the OBDD representing the set of states satisfying $a$. If $f = f_1 \land f_2$ or $f = \neg f_1$, then $\text{Check}(f)$ will be easily obtained according to $\text{Check}(f_1)$ and $\text{Check}(f_2)$. Formulas of the form $\text{EX } f$, $\text{E}[f \text{ U } g]$, and $\text{EG } f$ are handled by the procedures:

- $\text{Check}(\text{EX } f) = \text{CheckEX}(\text{Check}(f))$,
- $\text{Check}(\text{E}[f \text{ U } g]) = \text{CheckEU}(\text{Check}(f), \text{Check}(g))$,
- $\text{Check}(\text{EG } f) = \text{CheckEG}(\text{Check}(f))$. 
If $f$ is an atomic proposition $a$, then $\text{Check}(f)$ is the OBDD representing the set of states satisfying $a$. If $f = f_1 \land f_2$ or $f = \neg f_1$, then $\text{Check}(f)$ will be easily obtained according to $\text{Check}(f_1)$ and $\text{Check}(f_2)$. Formulas of the form $\textbf{EX} f$, $\textbf{E}[f \textbf{ U} g]$, and $\textbf{EG} f$ are handled by the procedures:

- $\text{Check(EX } f) = \text{CheckEX}(\text{Check}(f))$,
- $\text{Check(E } [f \textbf{ U} g] = \text{CheckEU}(\text{Check}(f), \text{Check}(g))$,
- $\text{Check(EG } f) = \text{CheckEG}(\text{check}(f))$. 

Ma Li  
PKU  
An Overview of Model Checking
Symbolic Model Checking

If \( f \) is an atomic proposition \( a \), then \( \text{Check}(f) \) is the OBDD representing the set of states satisfying \( a \). If \( f = f_1 \land f_2 \) or \( f = \neg f_1 \), then \( \text{Check}(f) \) will be easily obtained according to \( \text{Check}(f_1) \) and \( \text{Check}(f_2) \). Formulas of the form \( \text{EX} \ f \), \( \text{E}[f \ U g] \), and \( \text{EG}f \) are handled by the procedures:

- \( \text{Check}(\text{EX} \ f) = \text{CheckEX}(\text{Check}(f)) \),
- \( \text{Check}(\text{E} [f \ U g] = \text{CheckEU}(\text{Check}(f), \text{Check}(g)) \),
- \( \text{Check}(\text{EG} f) = \text{CheckEG}(\text{check}(f)) \).
Edmund M. Clarke, Jr., Oma Grumberg, and Doron A. Peled

*Model Checking.*

E. M. Clarke, E.A. Emerson, and A.P. Sistla.

*Automatic verification of finite-state concurrent systems using temporal logic specifications.*
Reference

Edmund M. Clarke, Jr., Oma Grumberg, and Doron A. Peled
*Model Checking.*

E. M. Clarke, E.A. Emerson, and A.P. Sistla.
*Automatic verification of finite-state concurrent systems using temporal logic specifications.*
Reference

Edmund M. Clarke, Jr., Oma Grumberg, and Doron A. Peled
*Model Checking.*

E. M. Clarke, E.A. Emerson, and A.P. Sistla.
*Automatic verification of finite-state concurrent systems using temporal logic specifications.*
Questions or Comments?