Formal Methods
Lecture VII

Symbolic Model Checking

Alessandro Artale

Faculty of Computer Science – Free University of Bozen-Bolzano
artale@inf.unibz.it http://www.inf.unibz.it/~artale/
Room 2.03
Academic Year: 2010/11

Some material (text, figures) displayed in these slides is courtesy of:
M. Benerecetti, A. Cimatti, M. Fisher, F. Giunchiglia, M. Pistore,
M. Roveri, R. Sebastiani.
1. Representing Set of States as OBDD's

2. Symbolic Model-Checking Algorithm
Main Ideas

- OBDD’s allow systems with a large state space to be verified.
- The Labeling algorithm takes a CTL formula and returns a **set of states** manipulating intermediate **set of states**.
- The algorithm is changed by storing set of states as OBDD’s and then manipulating them.
- Model checking using OBDD’s is called **Symbolic Model Checking**.
Symbolic Representation of States

Example:

- Three state variables $x_1, x_2, x_3$:
  \{000, 001, 010, 011\} represented as “first bit false”: $\neg x_1$

- With five state variables $x_1, x_2, x_3, x_4, x_5$:
  \{00000, 00001, 00010, 00011, 00100, 00101, 00110, 00111, \ldots, 01111\} still represented as “first bit false”: $\neg x_1$
Let $M = (S, I, R, L, AP)$ be a Kripke structure.

States $s \in S$ are described by means of a vector $V = (v_1, v_2, \ldots, v_n)$ of boolean values: One for each $x_i \in AP$.

- A state, $s$, is a truth assignment to each variable in $AP$ such that $v_i = 1$ iff $x_i \in L(s)$.
- Example: $0100$ represents the state $s$ where only $x_2 \in L(s)$. 

Boolean vectors can be represented by boolean formulas

**Example:** 0100 can be represented by the formula
\[ \xi(s) = (\neg x_1 \land x_2 \land \neg x_3 \land \neg x_4) \]

We call \( \xi(s) \) the formula representing the state \( s \in S \)
(Intuition: \( \xi(s) \) holds iff the system is in the state \( s \))

A set of states, \( Q \subseteq S \), can be represented by the formula –

**Characteristic Function of \( Q \):**

\[ \xi(Q) = \bigvee_{s \in Q} \xi(s) \]

Thus, (set of) states can be encoded as OBDD’s!
Remark

- Any propositional formula is a (typically very compact) representation of the set of assignments satisfying it.
- **Any formula equivalent to** \( \xi(Q) \) **is a representation of** \( Q \).
  
  \( \Rightarrow \) Typically \( Q \) can be encoded by much smaller formulas than \( \bigvee_{s \in Q} \xi(s)! \).

- **Example:** \( Q = \{00000, 00001, 00010, 00011, 00100, 00101, 00110, 00111, \ldots, 01111\} \) represented as “first bit false”: \( \neg x_1 \)

\[
\bigvee_{s \in Q} \xi(s) = (\neg x_1 \land \neg x_2 \land \neg x_3 \land \neg x_4 \land \neg x_5) \lor \\
(\neg x_1 \land \neg x_2 \land \neg x_3 \land \neg x_4 \land x_5) \lor \\
(\neg x_1 \land \neg x_2 \land \neg x_3 \land x_4 \land \neg x_5) \lor \\
\ldots \\
(\neg x_1 \land x_2 \land x_3 \land x_4 \land x_5) \Bigg\} 2^4 \text{ disjuncts}
\]
The transition relation $R$ is a set of pairs of states: $R \subseteq S \times S$.

Then, a single transition is a pair of states $(s, s')$.

A new vector of variables $V'$ (the next state vector) represents the value of variables after the transition has occurred.

$\xi(s, s')$ defined as $\xi(s) \land \xi(s')$.

The transition relation $R$ can be (naively) represented by

$$\bigvee_{(s,s') \in R} \xi(s, s') = \bigvee_{(s,s') \in R} \xi(s) \land \xi(s')$$
Remark

- Any formula equivalent to $\xi(R)$ is a representation of $R$
  
  $\Rightarrow$ Typically $R$ can be encoded by a much smaller formula than $V_{(s,s') \in R} \xi(s) \land \xi(s')$!

- Example: a synchronous sequential circuit

\[
\begin{array}{cccc}
0 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 \\
1 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 \\
\end{array}
\]

\[
\begin{align*}
\xi(R) & = (v'_0 \iff \neg v_0) \land (v'_1 \iff v_0 \oplus v_1) \\
V_{(s,s') \in R} \xi(s) \land \xi(s') & = (\neg v_0 \land \neg v_1 \land v'_0 \land \neg v'_1) \lor \\
& (v_0 \land \neg v_1 \land \neg v'_0 \land v'_1) \lor \\
& (\neg v_0 \land v_1 \land v'_0 \land v'_1) \lor \\
& (v_0 \land v_1 \land \neg v'_0 \land \neg v'_1)
\end{align*}
\]
Summary

- Representing Set of States as OBDD’s.
- **Symbolic Model-Checking Algorithm.**
**Problem:** \( M \models \varphi \)?

- Let \( M = \langle S, I, R, L, AP \rangle \) be a Kripke structure and \( \varphi \) be a CTL formula.
- The Symbolic Model-Checking algorithm is a Labeling algorithm that makes use of OBDD.
- It is implemented by a recursive procedure \texttt{CHECK} with:
  - **Input:** \( \varphi \), the formula to be checked;
  - **Output:** \( B_\varphi \), the OBDD representing the states satisfying \( \varphi \).
To check whether $I \subseteq \llbracket \varphi \rrbracket$:

$$(B_I \Rightarrow B_\varphi) \equiv B_T$$

i.e.,

$${\text{APPLY}}(\Rightarrow, B_I, B_\varphi) \equiv B_T$$

To compute OBDD’s for CTL formulas we need to understand how to compute them in case of the temporal operators: $\Diamond \bigcirc, \Diamond \mathcal{U}, \Diamond \square$. 
PreImage

- Backward (pre) image of a set:

  \[ \text{PreImage}(P) \]

  Evaluate all transitions ending in the states of the set.

- Set theoretic view:

  \[
  \text{PreImage}(P, R) := \{ s \in S \mid \exists s'. (s, s') \in R \text{ and } s' \in P \} 
  \]

- Logical Characterization:

  \[
  \xi(\text{PreImage}(P, R)) := \exists V'. (\xi(P)[V'] \land \xi(R)[V, V']) 
  \]

- N.B.: quantification over propositional variables
Example: A synchronous sequential circuit

- \( \xi(R) = (v_0' \iff \neg v_0) \land (v_1' \iff v_0 \oplus v_1) \)
- \( \xi(P) := (v_0 \iff v_1) \) (i.e., \( P = \{00, 11\} \))

Pre Image:

\[
\xi(\text{PreImage}(P, R)) = \exists V'. (\xi(P)[V'] \land \xi(R)[V, V']) \\
= \exists V'. ((v_0' \iff v_1') \land (v_0' \iff \neg v_0) \land (v_1' \iff v_0 \oplus v_1))
\]
OBDD for PrelImages

- $B\bigcirc \varphi$, the OBDD for $\bigcirc \varphi$, is computed starting from the OBDD's for both $\varphi$, $B\varphi$, and the transition relation, $B_R$.
- $\xi(\text{PrelImage}(\varphi, R)) := \exists V'.(\xi(\varphi)[V'] \land \xi(R)[V, V'])$, then:
  1. Rename the variables in $B\varphi$ to their primed version, $B\varphi'$
  2. Compute $B(\varphi' \land R) = \text{APPLY}(\land, B\varphi', B_R)$;
  3. $B\bigcirc \varphi$ is a sequence of:

    $$\text{APPLY}(\lor, \text{RESTRICT}(0, x_i', B(\varphi' \land R)), \text{RESTRICT}(1, x_i', B(\varphi' \land R)))$$

    where $x_i' \in V'$
- We call $\text{PRE}(B\varphi)$ the procedure that computes $B\bigcirc \varphi$. 

Outline

Representing Set of States as OBDD's
Symbolic Model-Checking Algorithm
The **Check** Symbolic M.C. Algorithm

\[
\text{Check}(\varphi) \left\{ \begin{array}{l}
    \text{case } \varphi \text{ of }
    \\
    \begin{array}{ll}
        \text{true:} & \text{return } B_T; \\
        \text{false:} & \text{return } B_\bot; \\
        \text{an atom } x_i: & \text{return } B_{x_i}; \\
        \neg \varphi_1: & \text{return } \text{Invert}(\text{Check}(\varphi_1)); \\
        \varphi_1 \land \varphi_2: & \text{return } \text{Apply}(\land, \text{Check}(\varphi_1), \text{Check}(\varphi_2)); \\
        \Diamond \lozenge \varphi_1: & \text{return } \text{Pre}(\text{Check}(\varphi_1)); \\
        \Diamond (\varphi_1 \cup \varphi_2): & \text{return } \text{Check\_EU}(\text{Check}(\varphi_1), \text{Check}(\varphi_2)); \\
        \Diamond \square \varphi_1: & \text{return } \text{Check\_EG}(\text{Check}(\varphi_1)); \\
    \end{array}
\end{array} \right.
\]

Alessandro Artale

Formal Methods Lecture VII Symbolic Model Checking
\[
[\Diamond \Box \varphi] = [\varphi] \cap \text{PRE}(\Diamond \Box \varphi)
\]

\text{CHECK}_{\text{EG}}(B\varphi)\
\begin{array}{l}
\text{var } X, \text{OLD-}X; \\
X := B\varphi; \\
\text{OLD-}X := B_{\bot}; \\
\text{while } X \neq \text{OLD-}X \\
\text{begin} \\
\quad \text{OLD-}X := X; \\
\quad X := \text{APPLY}(\land, X, \text{PRE}(X)) \\
\text{end} \\
\text{return } X
\end{array}
Check_EU

\[
[\Diamond (\varphi \mathcal{U} \psi)] = [\psi] \cup ([\varphi] \cap \text{PRE}([\Diamond (\varphi \mathcal{U} \psi)]))
\]

\text{CHECK\_EU}(B\varphi, B\psi)\
\begin{tabular}{l}
\textbf{var } X, OLD-X; \\
X := B\psi; \\
OLD-X := B_T; \\
\textbf{while } X \neq OLD-X \\
\textbf{begin} \\
\quad OLD-X := X; \\
\quad X := \text{APPLY}(\lor, X, \text{APPLY}(\land, B\varphi, \text{PRE}(X))) \\
\textbf{end} \\
\textbf{return } X
\end{tabular}
CTL Symbolic Model Checking—Summary

- Based on fixed point CTL M.C. algorithms
- Kripke structure encoded as boolean formulas (OBDDs)
- All operations handled as (quantified) boolean operations
- Avoids building the state graph explicitly
- Reduces dramatically the state explosion problem
  ⇒ problems of up to $10^{120}$ states handled!!
Partitioned Transition Relations

- There may be significant efficiency problems:
  - The transition relation may be too large to construct
  - Intermediate OBDDs may be too large to handle.

- IDEA: Partition conjunctively the transition relation:

\[ R(V, V') \leftrightarrow \bigwedge_{i} R_i(V_i, V'_i) \]

- Trade one “big” quantification for a sequence of “smaller” quantifications

  \[ \exists V'_1 \ldots V'_n . (R_1(V_1, V'_1) \land \ldots \land R_n(V_n, V'_n) \land Q(V')) \]

  by pushing quantifications inward can be reduced to

  \[ \exists V'_1 . (R_1(V_1, V'_1) \land \ldots \land \exists V'_n (R_n(V_n, V'_n) \land Q(V'))) \]

  which is typically much smaller
Symbolic Model Checkers

- Most hardware design companies have their own Symbolic Model Checker(s)
  - Intel, IBM, Motorola, Siemens, ST, Cadence, ...
  - very advanced tools
  - proprietary technology!

- On the academic side
  - CMU SMV [McMillan]
  - VIS [Berkeley, Colorado]
  - Bwolen Yang’s SMV [CMU]
  - NuSMV [CMU, IRST, UNITN, UNIGE]
  - ...
Summary of Lecture VII

- Representing Set of States as OBDD’s.
- Symbolic Model-Checking Algorithm.