Introduction to Software Verification

Orna Grumberg

Lectures Material
winter 2017-18
Lecture 8
Symbolic (BDD-based) Model Checking for CTL

A solution to the state-explosion problem
Binary Decision Diagrams (BDDs)

• Data structure for representing Boolean functions

• Boolean function:

\[ f: \{0,1\}^k \rightarrow \{0,1\} \]
\[ f(x_1, ..., x_k) = x_{k+1} \]
where \( x_1, ..., x_k, x_{k+1} \in \{0,1\} \)
BDD for \( f(a, b, c) = (a \land b) \lor c \)

Decision tree

BDD
Binary Decision Diagrams (BDDs)

Advantages of BDDs:

• Often (but not always) concise in size

• Canonical representation

• Most Boolean operations can be performed on BDDs in polynomial time in the BDD size
BDDs in Model Checking

• Every set \( A \subseteq U \) can be represented by its characteristic function

\[
f_A(u) = \begin{cases} 
1 & \text{if } u \in A \\
0 & \text{if } u \notin A 
\end{cases}
\]

• If the elements of \( U \) are encoded by sequences over \( \{0,1\}^n \) then \( f_A \) is a Boolean function and can be represented by a BDD
• A Boolean function represents the set of all elements for which the function is 1
Representing a Model with BDDs

• Assume that states in model $M$ are encoded by $\{0,1\}^n$ and described by Boolean variables $v_1...v_n$

• $S_f$ can be represented by a Boolean function (BDD) over $v_1...v_n$

• $R$ (a set of pairs of states $(s,s')$) can be represented by a BDD over $v_1...v_n$ $v'_1...v'_n$
Example: Representing a Model with BDDs

\[ S = \{ s_1, s_2, s_3 \} \]
\[ R = \{ (s_1,s_2), (s_2,s_2), (s_3,s_1) \} \]

State encoding:
\[ s_1: v_1v_2=00 \quad s_2: v_1v_2=01 \quad s_3: v_1v_2=11 \]

For \( A = \{s_1, s_2\} \) the Boolean formula representing \( A \):
\[ f_A(v_1,v_2) = (\neg v_1 \land \neg v_2) \lor (\neg v_1 \land v_2) = \neg v_1 \]
Example: Representing a Model with BDDs

State encoding:

\(s_1: \, v_1v_2=00\) \hspace{1em} \(s_2: \, v_1v_2=01\) \hspace{1em} \(s_3: \, v_1v_2=11\)

For \(A = \{s_1, s_2\}\) the Boolean formula representing \(A\):

\[f_A(v_1,v_2) = (\neg v_1 \land \neg v_2) \lor (\neg v_1 \land v_2) = \neg v_1\]

\(f_A\) represents \(A\) if it gets the value 1 for every assignment which is an encoding of an element of \(A\)
\[ f_R(v_1, v_2, v'_1, v'_2) = \]
\[ (\neg v_1 \land \neg v_2 \land \neg v'_1 \land v'_2) \lor \]
\[ (\neg v_1 \land v_2 \land \neg v'_1 \land v'_2) \lor \]
\[ (v_1 \land v_2 \land \neg v'_1 \land \neg v'_2) \]

\( f_A \) and \( f_R \) can be represented by BDDs.
BDDs as a data structure

A BDD for a Boolean function \( f(x_1, \ldots, x_k) \) is a directed acyclic graph \((\text{DAG})\) with a root and two types of nodes:

- **Internal nodes** \( v \) with fields
  - \( \text{var}(v) \) containing a variable name
  - Pointers \( \text{low}(v), \text{high}(v) \) to other nodes

- **End nodes** (leaves) \( v \) with field
  - \( \text{value}(v) \in \{0,1\} \)
Reduced, Ordered BDD (ROBDD)

To obtain a canonical representation, the following requirements are added:

• A variable appears at most once along every path from root to leaf

• The variables appear in the same order along every path from root to leaf

• The graph does not contain
  - isomorphic sub-graphs
  - Redundant nodes
Two graphs with root nodes $u$ and $v$ are isomorphic iff

• If $u$ and $v$ are leaves then $\text{value}(u) = \text{value}(v)$

• If $u$ and $v$ are internal nodes then
  - $\text{var}(u) = \text{var}(v)$
  - $\text{low}(u)$ and $\text{low}(v)$ are isomorphic
  - $\text{high}(u)$ and $\text{high}(v)$ are isomorphic

A node $v$ is redundant if $\text{low}(v) = \text{high}(v)$
Remark: From now on we will use $\text{BDD}$ to denote $\text{ROBDD}$
Order of Variables in the BDD

- $f (a_1, b_1, a_2, b_2) = (a_1 \leftrightarrow b_1) \land (a_2 \leftrightarrow b_2)$
- Assume the variable order is: $a_1 < b_1 < a_2 < b_2$:
Order of Variables in the BDD

• In the general case: \( a_1 < \ldots < a_n < b_1 < \ldots < b_n \)
• \( f(a_1, b_1, \ldots, a_n, b_n) = (a_1 \leftrightarrow b_1) \land \ldots \land (a_n \leftrightarrow b_n) \)
• BDD with \( 3n+2 \) nodes
Order of Variables in the BDD

- \( f(a_1, b_1, a_2, b_2) = (a_1 \leftrightarrow b_1) \land (a_2 \leftrightarrow b_2) \)
- Assume the variable order is: \( :a_1 < a_2 < b_1 < b_2 : \)
Order of Variables in the BDD

• In the general case: $a_1 < \ldots < a_n < b_1 < \ldots < b_n$:
• $f(a_1, b_1, a_2, b_2) = (a_1 \leftrightarrow b_1) \land \ldots \land (a_n \leftrightarrow b_n)$
• From $a_1$ to $b_1$ (including $b_1$) – full binary tree of depth $n+1$. Total of $2^{n+1}-1$ nodes
• Every $b_1$ remembers a specific assignment to $a_1, \ldots, a_n$. Thus $2^n$ nodes of $b_1$ (for all possible assignments of $a_1, \ldots, a_n$)
• Similarly, $2^{n-1}$ nodes of $b_2$ (for possible assignments of $a_2, \ldots, a_n$)
• ........
• 2 nodes of $b_n$ (for possible assignments of $a_n$)
Order of Variables in the BDD

- In the general case: $a_1<....<a_n<b_1<....<b_n$:
- $f(a_1,b_1, a_2,b_2) = (a_1\leftrightarrow b_1) \land ... \land (a_n\leftrightarrow b_n)$

- Sum of geometric sequence:
  $S_n= a_1*(q^n -1)/(q-1)$ (where $a_n=a_1*q^{n-1}$)

- For $b_2, ..., b_n$:
  - $2 + .... + 2*2^{n-1} = [2*(2^{n-1}-1)]/(2-1) = 2^{n-2}$
  - Total number of nodes:
    $2 + (2^{n-2}) + (2^{n+1}-1) = 3*2^n -1$
Conclusion:

- The order of variables can influence significantly the size of the BDD

- Finding an optimal variable order is a hard problem
  - Heuristics are used
Given a BDD – which Boolean function does it represent?

A BDD with root \( v \) represents the function \( f_v(x_1, \ldots, x_n) \) as follows:

- If \( v \) is a leaf then \( f_v(x_1, \ldots, x_n) = \text{value}(v) \)
- If \( v \) is internal and \( \text{var}(v) = x_i \) then
  \[
  f_v(x_1, \ldots, x_n) = (\neg x_i \land f_{\text{low}(v)}(x_1, \ldots, x_n)) \lor (x_i \land f_{\text{high}(v)}(x_1, \ldots, x_n))
  \]
Example

- \text{var}(v_1)=a, \text{var}(v_2)=b, \text{var}(v_3)=c
- \text{value}(v_4) = 0, \text{Value}(v_5) = 1
- f_{v_3}(a,b,c) = (\neg c \land f_{v_4}(a,b,c)) \lor (c \land f_{v_5}(a,b,c))
  = (\neg c \land 0) \lor (c \land 1) = c
- f_{v_2}(a,b,c) = (\neg b \land f_{v_3}(a,b,c)) \lor (b \land f_{v_5}(a,b,c))
  = (\neg b \land c) \lor (b \land 1) = (b \lor c)
\[ f_{v1}(a,b,c) = (\overline{a} \land f_{v3}(a,b,c)) \lor (a \land f_{v2}(a,b,c)) = (\overline{a} \land c) \lor (a \land (b \lor c)) = (c \lor (a \land b)) \]

The BDD represents many functions, all equivalent to each other.
Advantage of BDDs (revisited)

• Often (but not always) **concise** in size

• **Canonical** representation for a given variable ordering
  - Easy to check **equivalence** between two functions

• A function depends exactly on all variables that appear in its BDD

• Most **Boolean operations** can be performed on BDDs in **polynomial time** in the BDD size
Operations on BDDs
Operations on BDDs - Reduce

Reduce
Given an unreduced BDD:

• Eliminate isomorphic sub-graphs:
  - Eliminate duplicated end nodes
  - Eliminate duplicated internal nodes

• Eliminate redundant nodes

Reduce works bottom-up in linear time in the BDD size
Important remark:
BDD for a complex function is built bottom-up starting from small sub-functions to larger ones

We do not build a full decision tree and then reduce
Operations on BDDs - Restrict

Restrict

Given a BDD for \( f(x_1,\ldots,x_n) \), build a BDD for

\[
f|_{x_i=b}(x_1,\ldots,x_n) = f(x_1,\ldots,x_{i-1},b,x_{i+1},\ldots,x_n) \quad b \in \{0,1\}
\]

Example:

\[
f(x_1,x_2,x_3,x_4) = (x_1 \land x_2) \lor (x_3 \land x_4)
\]

\[
f|_{x_2=0}(x_1,x_2,x_3,x_4) = (x_1 \land 0) \lor (x_3 \land x_4) = (x_3 \land x_4)
\]
Operations on BDDs - Restrict

Given a BDD $A$ for $f(x_1,\ldots,x_n)$, build a BDD for $f|_{x_i=b}(x_1,\ldots,x_n) = f(x_1,\ldots,x_{i-1},b,x_{i+1},\ldots,x_n)$

- Traverse $A$ from root to leaves
- For every node $v$ with $\text{var}(v)=x_i$
  - Eliminate $v$ from $B$
  - Replace edges to $v$ by edges to $\text{low}(v)$, if $b=0$
    and to $\text{high}(v)$, if $b=1$
- Run Reduce
Example: Restrict

\[ f(x_1, x_2, x_3, x_4) = (x_1 \land x_2) \lor (x_3 \land x_4) \]

\[ f|_{x_2=0}(x_1, x_2, x_3, x_4) = (x_1 \land 0) \lor (x_3 \land x_4) = (x_3 \land x_4) \]
Operations on BDDs - Apply

- Gets two BDDs, representing functions $f$ and $f'$ and an operation $*$
  - Over the same variable ordering

- Returns the BDD representing $f*f'$

- $*$ can be any of 16 binary operations on two Boolean functions
# 16 binary operations

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>$f_1$</th>
<th>$f_2$</th>
<th>$f_3$</th>
<th>...</th>
<th>$f_{15}$</th>
<th>$f_{16}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>...</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>...</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

const 0 AND ? OR const 1
Operations on BDDs - Apply

• Shannon expansion
  for every Boolean function $f$ and a variable $x$:
  
  $$f = (\neg x \land f|_{x=0}) \lor (x \land f|_{x=1})$$

Notation:
• $v, v'$ are the roots of $f, f'$, respectively
• If $v, v'$ are not end nodes then $\text{var}(v) = x$, $\text{var}(v') = x'$
Operations on BDDs - Apply

Computing $f \cdot f'$:

- **Case 1**: $v$ and $v'$ are end nodes
  
  $f \cdot f' = \text{value}(v) \cdot \text{value}(v')$

- The BDD for $f \cdot f'$

  consists of one leaf $v''$ with

  $\text{value}(v'') = \text{value}(v) \cdot \text{value}(v')$

This is the only case where $\cdot$ is taken into account
Operations on BDDs - Apply

Computing \( f \cdot f' \):

- **Case 2**: \( x = x' \)
- Use Shannon expansion:

\[
\begin{align*}
  f \cdot f' &= (\neg x \wedge (f|_{x=0} \cdot f'|_{x=0})) \lor \\
  &\quad (x \wedge (f|_{x=1} \cdot f'|_{x=1}))
\end{align*}
\]

- Two simpler sub-problems to solve
  - Each depends on one less variable
End of lecture
5.12.2017
Operations on BDDs - Apply

Computing $f \cdot f'$:

- **Case 2**: $x = x'$
- The BDD for $f \cdot f'$
- Root: a new node $v''$
  - $\text{var}(v'') = x$
  - $\text{low}(v'')$ points to the root of the BDD for $( f \|_{x=0} \cdot f' \|_{x=0} )$
  - $\text{high}(v'')$ points to the root of the BDD for $( f \|_{x=1} \cdot f' \|_{x=1} )$
Example

- \( f(a) = a, \quad f'(a) = \neg a, \quad * \text{ is } \lor \)

- The BDD for \( f \lor f' \) is:

- The BDD for \( f \lor f' \) is:

  - Reduce
Computing $f \ast f'$:

- **Case 3**: $x < x'$
- $x$ does not appear in $f'$

\[
\begin{align*}
  f' \mid_{x=0} &= f' \mid_{x=1} = f' \\
  f \ast f' &= (\neg x \wedge (f \mid_{x=0} \ast f')) \lor \ (x \wedge (f \mid_{x=1} \ast f'))
\end{align*}
\]
Operations on BDDs - Apply

Computing \( f \times f' \):

- **Case 4:** \( x > x' \)

  Similar to case 3
Example

• \( f(a,b) = a \rightarrow b \), \( f'(a,b) = \neg b \), * is \( \leftrightarrow \)

• \( f \leftrightarrow f' \equiv (a \rightarrow b) \leftrightarrow (\neg b) \equiv (\neg a \lor b) \leftrightarrow (\neg b) \)

\[ \equiv (\neg a \land \neg b) \]
Example

- $f(a,b) = a \rightarrow b$, $f'(a,b) = \neg b$, $*$ is $\leftrightarrow$, $a < b$
Example

• $f(a,b) = a \rightarrow b$, $f'(a,b) = \neg b$, * is $\iff$
Complexity of apply
Naive implementation

• two sub-problems for each variable
• exponential in the number of variables
Complexity of apply
Non-naive implementation

Notice:
• Every BDD node $u$ represents a function $f_u$
• $|f|$, $|f'|$ denote the number of nodes in the BDD for $f$, $f'$ respectively

Solution:
• Use hash table with entries:
  - Pointers to (the root node of) the BDDs for $g$, $g'$, and $*$
  - Pointer to the resulting BDD for $g*g'$
Consequences:

- Never redo an operation on the same BDDs
  - Never solve the same sub-problem twice
- Never insert into the BDD manager the same BDD twice

Complexity

- The number of different sub-problems is $O(|f| \times |f'|)$
  - Polynomial in the BDD sizes