Reduced Ordered Binary Decision Diagrams

Lecture #13 of Advanced Model Checking

Joost-Pieter Katoen

Lehrstuhl 2: Software Modeling & Verification

E-mail: katoen@cs.rwth-aachen.de

June 5, 2012
Switching functions

• Let $\text{Var} = \{z_1, \ldots, z_m\}$ be a finite set of Boolean variables

• An evaluation is a function $\eta : \text{Var} \rightarrow \{0, 1\}$
  – let $\text{Eval}(z_1, \ldots, z_m)$ denote the set of evaluations for $z_1, \ldots, z_m$
  – shorthand $[z_1 = b_1, \ldots, z_m = b_m]$ for $\eta(z_1) = b_1, \ldots, \eta(z_m) = b_m$

• $f : \text{Eval} (\text{Var}) \rightarrow \{0, 1\}$ is a switching function for $\text{Var} = \{z_1, \ldots, z_m\}$

• Logical operations and quantification are defined by:

$$
\begin{align*}
  f_1(\cdot) \land f_2(\cdot) &= \min \{ f_1(\cdot), f_2(\cdot) \} \\
  f_1(\cdot) \lor f_2(\cdot) &= \max \{ f_1(\cdot), f_2(\cdot) \} \\
  \exists z. f(\cdot) &= f(\cdot)|_{z=0} \lor f(\cdot)|_{z=1}, \text{ and} \\
  \forall z. f(\cdot) &= f(\cdot)|_{z=0} \land f(\cdot)|_{z=1}
\end{align*}
$$
Ordered Binary Decision Diagram

Let $\varphi$ be a variable ordering for $Var$ where $z_1 <_\varphi \ldots <_\varphi z_m$

An $\varphi$-OBDD is a tuple $\mathcal{B} = (V, V_I, V_T, succ_0, succ_1, var, val, v_0)$ with

- a finite set $V$ of nodes, partitioned into $V_I$ (inner) and $V_T$ (terminals)
  - and a distinguished root $v_0 \in V$

- successor functions $succ_0, succ_1 : V_I \to V$
  - such that each node $v \in V \setminus \{v_0\}$ has at least one predecessor

- labeling functions $var : V_I \to Var$ and $val : V_T \to \{0, 1\}$ satisfying

\[
  v \in V_I \land w \in \{succ_0(v), succ_1(v)\} \cap V_I \Rightarrow var(v) <_\varphi var(w)
\]
Transition relation as an OBDD

An example OBDD representing $f \rightarrow$ for our example using $x_1 < x_2 < x'_1 < x'_2$
Symbolic composition operators
Consistent co-factors in OBDDs

- Let $f$ be a switching function for $\text{Var}$

- Let $\varphi = (z_1, \ldots, z_m)$ a variable ordering for $\text{Var}$, i.e., $z_1 \prec \varphi \ldots \prec \varphi z_m$

- Switching function $g$ is a $\varphi$-consistent cofactor of $f$ if

\[ g = f \mid z_1 = b_1, \ldots, z_i = b_i \quad \text{for some } i \in \{0, 1, \ldots, m\} \]

- Then it holds that:
  1. for each node $v$ of an $\varphi$-OBDD $\mathcal{B}$, $f_v$ is a $\varphi$-consistent cofactor of $f_\mathcal{B}$
  2. for each $\varphi$-consistent cofactor $g$ of $f_\mathcal{B}$ there is a node $v \in \mathcal{B}$ with $f_v = g$
Reduced OBDDs

A $\wp$-OBDD $\mathcal{B}$ is reduced if for every pair $(v, w)$ of nodes in $\mathcal{B}$:

$$v \neq w \text{ implies } f_v \neq f_w$$

(A reduced $\wp$-OBDD is abbreviated as $\wp$-ROBDD)

$\Rightarrow$ $\wp$-ROBDDs any $\wp$-consistent cofactor is represented by exactly one node
Transition relation as an ROBDD

(a) ordering $x_1 < x_2 < x'_1 < x'_2$

(b) ordering $x_1 < x'_1 < x_2 < x'_2$
Universality and canonicity theorem

Let $\text{Var}$ be a finite set of Boolean variables and $\wp$ a variable ordering for $\text{Var}$. Then:

(a) For each switching function $f$ for $\text{Var}$ there exists a $\wp$-ROBDD $\mathcal{B}$ with $f_{\mathcal{B}} = f$

(b) Any $\wp$-ROBDDs $\mathcal{B}$ and $\mathcal{C}$ with $f_{\mathcal{B}} = f_{\mathcal{C}}$ are isomorphic

Any $\wp$-OBDD $\mathcal{B}$ for $f$ is reduced iff $\text{size}(\mathcal{B}) \leq \text{size}(\mathcal{C})$ for each $\wp$-OBDD $\mathcal{C}$ for $f$
Reducing OBDDs

- Generate an OBDD (or BDT) for a switching function, then reduce
  - by means of a recursive descent over the OBDD

- Elimination of duplicate leaves
  - for a duplicate 0-leaf (or 1-leaf), redirect all incoming edges to just one of them

- Elimination of “don’t care” (non-leaf) vertices
  - if $\text{succ}_0(v) = \text{succ}_1(v) = w$, delete $v$ and redirect all its incoming edges to $w$

- Elimination of isomorphic subtrees
  - if $v \neq w$ are roots of isomorphic subtrees, remove $w$ and redirect all incoming edges to $w$ to $v$

note that the first reduction is a special case of the latter
How to reduce an OBDD?

becomes

*elimination of duplicated leaves*
How to reduce an OBDD?

\[ \begin{array}{c}
\text{becomes} \\
\end{array} \]

\[ \begin{array}{c}
\text{isomorphism rule} \\
\end{array} \]
How to reduce an OBDD?

becomes

elimination rule
Soundness and completeness

if $\mathcal{C}$ arises from a $\wp$-OBDD $\mathcal{B}$ by applying the elimination or isomorphism rule, then:
$\mathcal{C}$ is a $\wp$-OBDD with $f_\mathcal{B} = f_\mathcal{C}$

$\wp$-OBDD $\mathcal{B}$ is reduced if and only if no reduction rule is applicable to $\mathcal{B}$
Proof
Variable ordering

• ROBDDs are canonical for a \textbf{fixed} variable ordering
  - the size of the ROBDD crucially depends on the variable ordering
  - $\#$ nodes in ROBDD $B = \#$ of $\varphi$-consistent co-factors of $f$

• Some switching functions have \textbf{linear and exponential} ROBDDs
  - e.g., the addition function, or the stable function

• Some switching functions only have \textbf{polynomial} ROBDDs
  - this holds, e.g., for symmetric functions (see next)
  - examples $f(\ldots) = x_1 \oplus \ldots \oplus x_n$, or $f(\ldots) = 1$ iff $\geq k$ variables $x_i$ are true

• Some switching functions only have \textbf{exponential} ROBDDs
  - this holds, e.g., for the middle bit of the multiplication function
The function stable with exponential ROBDD

The ROBDD of $f_{stab}(\overline{x}, \overline{y}) = (x_1 \leftrightarrow y_1) \land \ldots \land (x_n \leftrightarrow y_n)$

has $3 \cdot 2^n - 1$ vertices under ordering $x_1 < \ldots < x_n < y_1 < \ldots < y_n$
The function stable with linear ROBDD

The ROBDD of $f_{stab}(\overline{x}, \overline{y}) = (x_1 \leftrightarrow y_1) \land \ldots \land (x_n \leftrightarrow y_n)$ has $3 \cdot n + 2$ vertices under ordering $x_1 < y_1 < \ldots < x_n < y_n$
Another function with an exponential ROBDD

ROBDD for $f_3(\overline{z}, \overline{y}) = (z_1 \land y_1) \lor (z_2 \land y_2) \lor (z_3 \land y_3)$

for the variable ordering $z_1 < z_2 < z_3 < y_1 < y_2 < y_3$
And an optimal linear ROBDD

- ROBDD for $f_3(x) = (z_1 \land y_1) \lor (z_2 \land y_2) \lor (z_3 \land y_3)$

- for ordering $z_1 < y_1 < z_2 < y_2 < z_3 < y_3$

- as all variables are essential for $f$, this ROBDD is optimal

- that is, for no variable ordering a smaller ROBDD exists
Symmetric functions

\[ f \in \text{ Eval}(z_1, \ldots, z_m) \text{ is symmetric if and only if } \]

\[ f([z_1 = b_1, \ldots, z_m = b_m]) = f([z_1 = b_{i_1}, \ldots, z_m = b_{i_m}]) \]

for each permutation \((i_1, \ldots, i_m)\) of \((1, \ldots, m)\).

E.g.: \(z_1 \lor z_2 \lor \ldots \lor z_m, z_1 \land z_2 \land \ldots \land z_m\), the parity function, and the majority function

If \(f\) is a symmetric function with \(m\) essential variables, then for each variable ordering \(\wp\) the \(\wp\)-ROBDD has size \(O(m^2)\)
The even parity function

\[ f_{\text{even}}(x_1, \ldots, x_n) = 1 \text{ iff the number of variables } x_i \text{ with value 1 is even} \]

*truth table or propositional formula for \( f_{\text{even}} \) has exponential size*

*but an ROBDD of linear size is possible*
The multiplication function

- Consider two \( n \)-bit integers
  - let \( b_{n-1}b_{n-2} \ldots b_0 \) and \( c_{n-1}c_{n-2} \ldots c_0 \)
  - where \( b_{n-1} \) is the most significant bit, and \( b_0 \) the least significant bit

- Multiplication yields a \( 2n \)-bit integer
  - the ROBDD \( B_{f_{n-1}} \) has at least \( 1.09^n \) vertices
  - where \( f_{n-1} \) denotes the \((n-1)\)-st output bit of the multiplication
Optimal variable ordering

- The size of ROBDDs is dependent on the variable ordering

- Is it possible to determine $\mathcal{O}$ such that the ROBDD has minimal size?
  - to check whether a variable ordering is optimal is NP-hard
  - polynomial reduction from the 3SAT problem [Bollig & Wegener, 1996]

- There are many switching functions with large ROBDDs
  - for almost all switching functions the minimal size is in $\Omega\left(\frac{2^n}{n}\right)$

- How to deal with this problem in practice?
  - guess a variable ordering in advance
  - rearrange the variable ordering during the ROBDD manipulations
  - not necessary to test all $n!$ orderings, best known algorithm in $\mathcal{O}(3^n \cdot n^2)$
Variable swapping
Sifting algorithm

[Rudell, 1993]

Dynamic variable ordering using variable swapping:

1. Select a variable $x_i$ in OBDD at hand

2. Successively swap $x_i$ to determine $\text{size}(\mathcal{B})$ at any position for $x_i$

3. Shift $x_i$ to position for which $\text{size}(\mathcal{B})$ is minimal

4. Go back to the first step until no improvement is made

- Characteristics:
  - a variable may change position several times during a single sifting iteration
  - often yields a local optimum, but works well in practice
**Interleaved variable ordering**

- Which variable ordering to use for transition relations?

- The **interleaved** variable ordering:
  - for encodings $x_1, \ldots, x_n$ and $y_1, \ldots, y_n$ of state $s$ and $t$ respectively:
    \[ x_1 < y_1 < x_2 < y_2 < \ldots < x_n < y_n \]

- This variable ordering yields compact ROBDDs for binary relations
  - for transition relation with $z_1 \ldots z_m$ be the encoding of action $\alpha$, take:
    \[ \underbrace{z_1 < z_2 < \ldots < z_m}_{\text{encoding of } \alpha} < \underbrace{x_1 < y_1 < x_2 < y_2 < \ldots < x_n < y_n}_{\text{interleaved order of states}} \]
Symbolic model checking

- Take a symbolic representation of a transition system ($\Delta$ and $\chi_B$)

- Backward reachability $\text{Pre}^*(B) = \{ s \in S \mid s \models \exists \Diamond B \}$

- Initially: $f_0 = \chi_B$ characterizes the set $T_0 = B$

- Then, successively compute the functions $f_{j+1} = \chi_{T_{j+1}}$ for:

  $$T_{j+1} = T_j \cup \{ s \in S \mid \exists s' \in S. s' \in \text{Post}(s) \land s' \in T_j \}$$

- The second set in the above union is given by: $\exists \overline{x}' . \left( \Delta(\overline{x}, \overline{x}') \land f_j(\overline{x}') \right)$

  - $f_j(\overline{x}')$ arises from $f_j$ by renaming the variables $x_i$ into their primed copies $x_i'$
Symbolic computation of $\text{Sat}(\exists (C \cup B))$

\[
\begin{align*}
f_0(\overline{x}) &:= \chi_B(\overline{x}); \\
j &:= 0; \\
\text{repeat} \quad &f_{j+1}(\overline{x}) := f_j(\overline{x}) \lor (\chi_C(\overline{x}) \land \exists \overline{x}'. (\Delta(\overline{x}, \overline{x}') \land f_j(\overline{x}'))); \\
&j := j + 1 \\
\text{until} \quad &f_j(\overline{x}) = f_{j-1}(\overline{x}); \\
\text{return} \quad &f_j(\overline{x}).
\end{align*}
\]
Symbolic computation of $Sat(\exists \Box B)$

Compute the largest set $T \subseteq B$ with $Post(t) \cap T \neq \emptyset$ for all $t \in T$

Take $T_0 = B$ and $T_{j+1} = T_j \cap \{ s \in S \mid \exists s' \in S. s' \in Post(s) \land s' \in T_j \}$

Symbolically this amounts to:

\[
\begin{align*}
f_0(x) & := \chi_B(x); \\
j & := 0; \\
\text{repeat} \\
f_{j+1}(x) & := f_j(x) \land \exists x'. (\Delta(x, x') \land f_j(x')); \\
j & := j + 1 \\
\text{until} & \quad f_j(x) = f_{j-1}(x); \\
\text{return} & \quad f_j(x).
\end{align*}
\]

Symbolic model checkers mostly use ROBDDs to represent switching functions
Synthesis of ROBDDs

- Construct a $\wp$-ROBDD for $f_1 \ op \ f_2$ given $\wp$-ROBDDs for $f_1$ and $f_2$
  - where $op$ is a Boolean connective such as disjunction, implication, etc.

- Idea: use a single ROBDD with (global) variable ordering $\wp$ to represent several switching functions

- This yields a shared OBDD, which is:
  a combination of several ROBDDs with variable ordering $\wp$
  by sharing nodes for common $\wp$-consistent cofactors

- The size of $\wp$-SOBDD $\overline{B}$ for functions $f_1, \ldots, f_k$ is at most $N_{f_1} + \ldots + N_{f_k}$ where $N_f$ denotes the size of the $\wp$-ROBDD for $f$
Implementation: shared OBDDs

A shared \( \varphi \)-OBDD is an OBDD with multiple roots

\[
\begin{align*}
\text{Shared OBDD representing } & \quad z_1 \land \neg z_2, \quad \neg z_2, \quad z_1 \lor z_2 \text{ and } \neg z_1 \lor z_2 \\
& \quad f_1, \quad f_2, \quad f_3, \quad f_4
\end{align*}
\]

Main underlying idea: combine several OBDDs with same variable ordering such that common \( \varphi \)-consistent co-factors are shared
Synthesizing shared ROBDDs

Relies on the use of two tables

- **The unique table**
  - keeps track of ROBDD nodes that already have been created
  - table entry \( \langle \text{var}(v), \text{succ}_1(v), \text{succ}_0(v) \rangle \) for each inner node \( v \)
  - main operation: \( \text{find}_\text{or}_\text{add}(z, v_1, v_0) \) with \( v_1 \neq v_0 \)
    - return \( v \) if there exists a node \( v = \langle z, v_1, v_0 \rangle \) in the ROBDD
    - if not, create a new \( z \)-node \( v \) with \( \text{succ}_0(v) = v_0 \) and \( \text{succ}_1(v) = v_1 \)
  - implemented using hash functions (expected access time is \( \mathcal{O}(1) \))

- **The computed table**
  - keeps track of tuples for which ITE has been executed (memoization)
  \( \Rightarrow \) realizes a kind of dynamic programming
Using shared OBDDs for model checking $\Phi$

Use a single SOBDD for:

- $\Delta(\overline{x}, \overline{x'})$ for the transition relation
- $f_a(\overline{x})$, $a \in AP$, for the satisfaction sets of the atomic propositions
- The satisfaction sets $Sat(\Psi)$ for the state subformulae $\Psi$ of $\Phi$

In practice, often the interleaved variable order for $\Delta$ is used.
ITE normal form

The ITE (if-then-else) operator: \( ITE(g, f_1, f_2) = (g \land f_1) \lor (\lnot g \land f_2) \)

The ITE operator and the representation of the SOBDD nodes in the unique table:

\[
f_v = ITE\left(z, f_{\text{succ}_1(v)}, f_{\text{succ}_0(v)}\right)
\]

Then:

\[
\begin{align*}
\neg f &= ITE(f, 0, 1) \\
f_1 \lor f_2 &= ITE(f_1, 1, f_2) \\
f_1 \land f_2 &= ITE(f_1, f_2, 0) \\
f_1 \oplus f_2 &= ITE(f_1, \neg f_2, f_2) = ITE(f_1, ITE(f_2, 0, 1), f_2)
\end{align*}
\]

If \( g, f_1, f_2 \) are switching functions for \( \text{Var} \), \( z \in \text{Var} \) and \( b \in \{0, 1\} \), then

\[
ITE(g, f_1, f_2)|_{z=b} = ITE(g|_{z=b}, f_1|_{z=b}, f_2|_{z=b})
\]
ITE-operator on shared OBDDs

• A node in a $\wp$-SOBDD for representing $ITE(g, f_1, f_2)$ is a node $w$ with $info \langle z, w_1, w_0 \rangle$ where:
  – $z$ is the minimal (wrt. $\wp$) essential variable of $ITE(g, f_1, f_2)$
  – $w_b$ is an SOBDD-node with $f_{w_b} = ITE(g|z=b, f_1|z=b, f_2|z=b)$

• This suggests a recursive algorithm:
  – determine $z$
  – recursively compute the nodes for ITE for the cofactors of $g$, $f_1$ and $f_2$
\textit{ITE}(u, v_1, v_2) \textbf{ on shared OBDDs (initial version)}

\begin{enumerate}
  \item \textbf{if} \(u\) \textbf{is terminal} \textbf{then}
    \begin{enumerate}
      \item \textbf{if} \(\text{val}(u) = 1\) \textbf{then}
        \begin{enumerate}
          \item \(w := v_1\)
        \end{enumerate}
      \item \textbf{else}
        \begin{enumerate}
          \item \(w := v_2\)
        \end{enumerate}
    \end{enumerate}
  \item \textbf{else}
    \begin{enumerate}
      \item \(z := \min\{\text{var}(u), \text{var}(v_1), \text{var}(v_2)\}\);
      \item \(w_1 := \text{ITE}(u|_{z=1}, v_1|_{z=1}, v_2|_{z=1})\);
      \item \(w_0 := \text{ITE}(u|_{z=0}, v_1|_{z=0}, v_2|_{z=0})\);
      \item \textbf{if} \(w_0 = w_1\) \textbf{then}
        \begin{enumerate}
          \item \(w := w_1\);
        \end{enumerate}
      \item \textbf{else}
        \begin{enumerate}
          \item \(w := \text{find\_or\_add}(z, w_1, w_0)\);
        \end{enumerate}
    \end{enumerate}
  \end{enumerate}
\end{enumerate}

\textbf{return} \(w\)

\textbf{(*) \text{ITE}(1, f_{v_1}, f_{v_2}) = f_{v_1} *)}

\textbf{(*) \text{ITE}(0, f_{v_1}, f_{v_2}) = f_{v_2} *)}

\textbf{(*) minimal essential variable *)}

\textbf{(*) elimination rule *)}

\textbf{(*) isomorphism rule *)}
ROBDD size under ITE

The size of the \( \wp \)-ROBDD for \( ITE(g, f_1, f_2) \) is bounded by \( N_g \cdot N_{f_1} \cdot N_{f_2} \)
where \( N_f \) denotes the size of the \( \wp \)-ROBDD for \( f \)
ROBDD size under ITE

The size of the $\varphi$-ROBDD for $\text{ITE}(g, f_1, f_2)$ is bounded by $N_g \cdot N_{f_1} \cdot N_{f_2}$

where $N_f$ denotes the size of the $\varphi$-ROBDD for $f$

But how to avoid multiple invocations to ITE?

⇒ Store triples $(u, v_1, v_2)$ for which ITE already has been computed
Efficiency improvement by memoization

if there is an entry for \((u, v_1, v_2, w)\) in the computed table then
return node \(w\)
else
if \(u\) is terminal then
if \(\text{val}(u) = 1\) then \(w := v_1\) else \(w := v_2\) fi
else
\(z := \min\{\text{var}(u), \text{var}(v_1), \text{var}(v_2)\}\);
\(w_1 := \text{ITE}(u|_{z=1}, v_1|_{z=1}, v_2|_{z=1})\);
\(w_0 := \text{ITE}(u|_{z=0}, v_1|_{z=0}, v_2|_{z=0})\);
if \(w_0 = w_1\) then \(w := w_1\) else \(w := \text{find_or_add}(z, w_1, w_0)\) fi;
insert \((u, v_1, v_2, w)\) in the computed table;
return node \(w\)
fi
fi

The number of recursive calls for the nodes \(u, v_1, v_2\) equals the \(\varnothing\)-ROBDD size of \(\text{ITE}(f_u, f_{v_1}, f_{v_2})\), which is bounded by \(N_u \cdot N_{v_1} \cdot N_{v_2}\)
Some experimental results

- **Traffic alert and collision avoidance system (TCAS)** (1998)
  - 277 boolean variables, reachable state space is about $9.6 \times 10^{56}$ states
  - $|B| = 124$, 618 vertices (about 7.1 MB), construction time 46.6 sec
  - checking $\forall \Box (p \rightarrow q)$ takes 290 sec and 717,000 BDD vertices

- **Synchronous pipeline circuit** (1992)
  - pipeline with 12 bits: reachable state space of $1.5 \times 10^{29}$ states
  - checking safety property takes about $10^4 - 10^5$ sec
  - $|B\rightarrow|$ is linear in data path width
  - verification of 32 bits (about $10^{120}$ states): 1h 25m
  - using partitioned transition relations
Some other types of BDDs

- Zero-suppressed BDDs
  - like ROBDDs, but non-terminals whose 1-child is leaf 0 are omitted

- Parity BDDs
  - like ROBDDs, but non-terminals may be labeled with $\oplus$; no canonical form

- Edge-valued BDDs

- Multi-terminal BDDs (or: algebraic BDDs)
  - like ROBDDs, but terminals have values in $\mathbb{R}$, or $\mathbb{N}$, etc.

- Binary moment diagrams (BMD)
  - generalization of ROBDD to linear functions over bool, int and real
  - uses edge weights
Further reading

- R. Bryant: Graph-based algorithms for Boolean function manipulation, 1986
- R. Bryant: Symbolic boolean manipulation with OBDDs, Computing Surveys, 1992
- M. Huth and M. Ryan: Binary decision diagrams, Ch 6 of book on Logics, 1999
- H.R. Andersen: Introduction to BDDs, Tech Rep, 1994
- K. McMillan: Symbolic model checking, 1992
- Rudell: Dynamic variable reordering for OBDDs, 1993

Advanced reading: Ch. Meinel & Th. Theobald (Springer 1998)