Symbolic Model Checking

Hao Zheng
Dept. of Computer Science & Eng.
Univ. of South Florida

Overview

- CTL model checking operates on sets.
  - Calculates the fix points over finite state sets.
- Systems are described with states and transitions.
  - Can be represented as Boolean functions.
- Symbolic model checking operates on boolean functions.
- OBDD enables much larger designs to be handled.
- Topics:
  - OBDD
  - Symbolic model checking algorithms

Sets and Boolean Functions

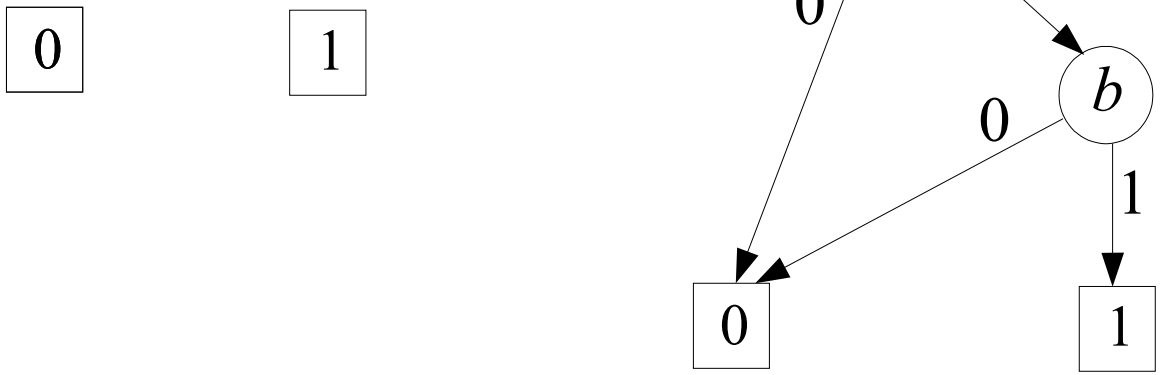
- Sets can be represented by boolean functions.
  - A boolean function is the characteristic function of a set.
  - $S = \{01, 10\}$, $Ch(S) = a \oplus b$.
- Set and boolean operations:
  - $\cap \leftrightarrow \wedge, \quad \cup \leftrightarrow \vee, \quad - \leftrightarrow \neg$
- Boolean functions can be represented as:
  - Truth tables,
  - Boolean formulas,
  - OBDD

Comparisons of Boolean Representations

Boolean Representation	Compact	SAT	Validity	$\vee$	$\wedge$	$\neg$
truth table	never	hard	hard	hard	hard	hard
formula CNF	sometimes	hard	easy	hard	easy	hard
formula DNF	sometimes	easy	hard	easy	hard	hard
ORBDD	often	easy	easy	medium	medium	easy

Overview of BDD

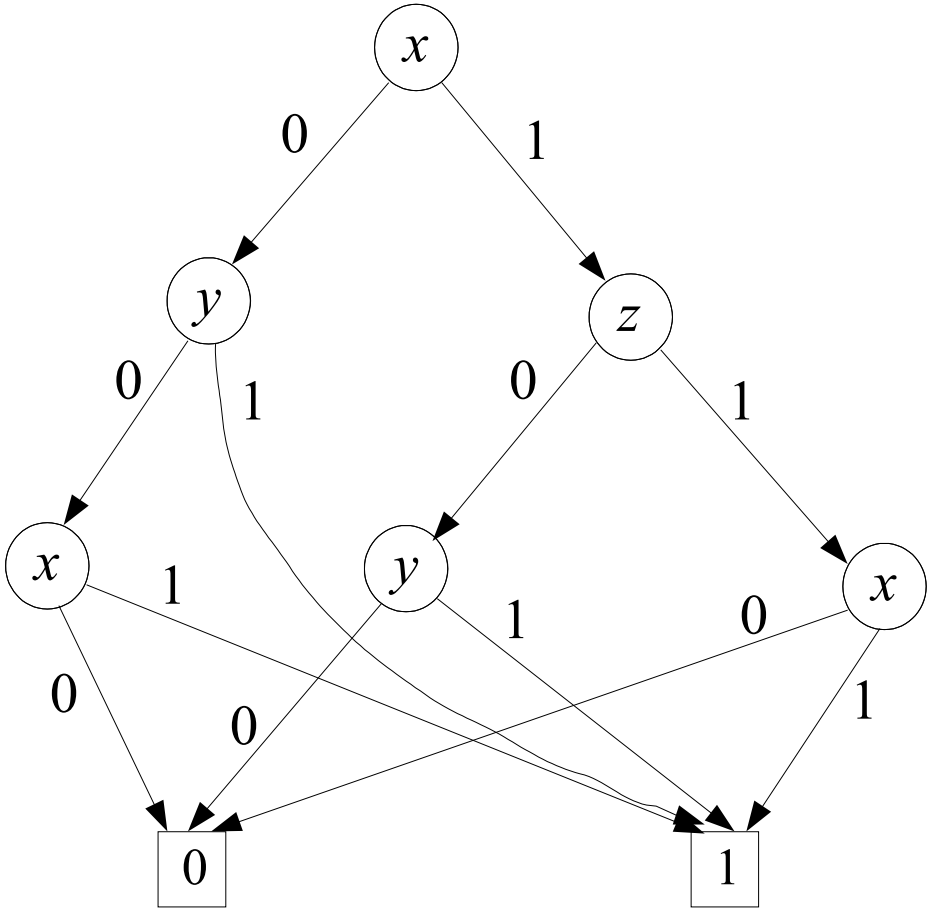
- BDD is a rooted DAG.
- There are two terminals: B_0 and B_1.
- Each non-terminal corresponds to a boolean variable.
  - It has two outgoing edges reflecting the value of the variable.

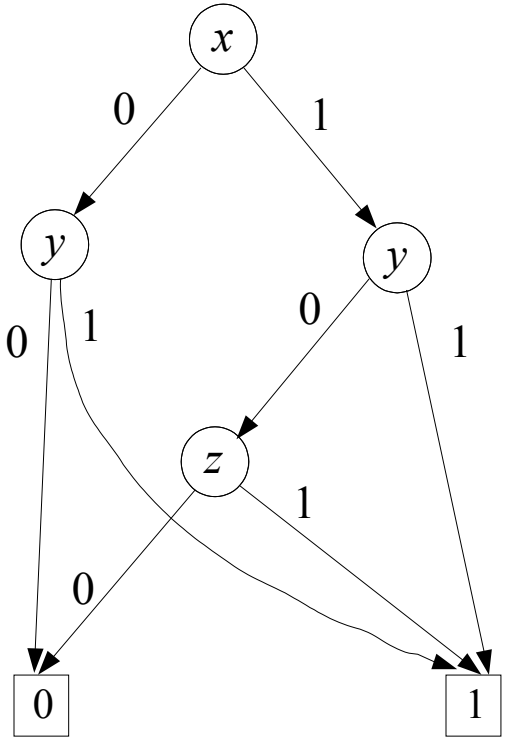


Overview of BDD (cont'd)

- The value of function is the value of the terminal that is reached through a path from the root.
  - The valuation of variables is determined by the values labeled for the edge of that path.
- A variable can happen multiple times in a path.
  - Results in redundant non-terminals.
  - Blows up the BDD size.
  - Expensive to decide the equivalence of boolean functions.
-

A BDD Example





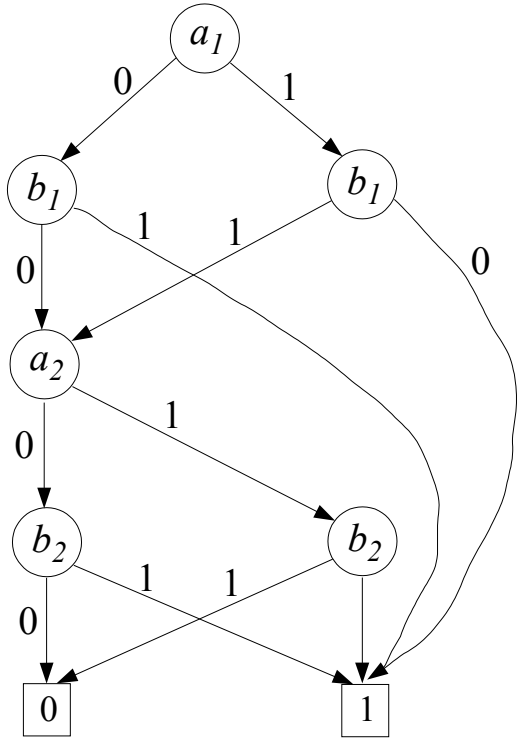
Ordered BDD

- A variable order constrains BDD.
  - Variables on all path in BDD follows that variable order.
  - Each variable happens only once along any path.
  - Not every variable needs to appear in a path.
- BDDs are required to have a compatible variable order.
  - *x* and *y* are in the same order in all BDDs.
- OBDDs have a canonical form.
  - OBDDs represent the same boolean function if they have identical structure.
  - The canonical form is derived with BDD reductions.

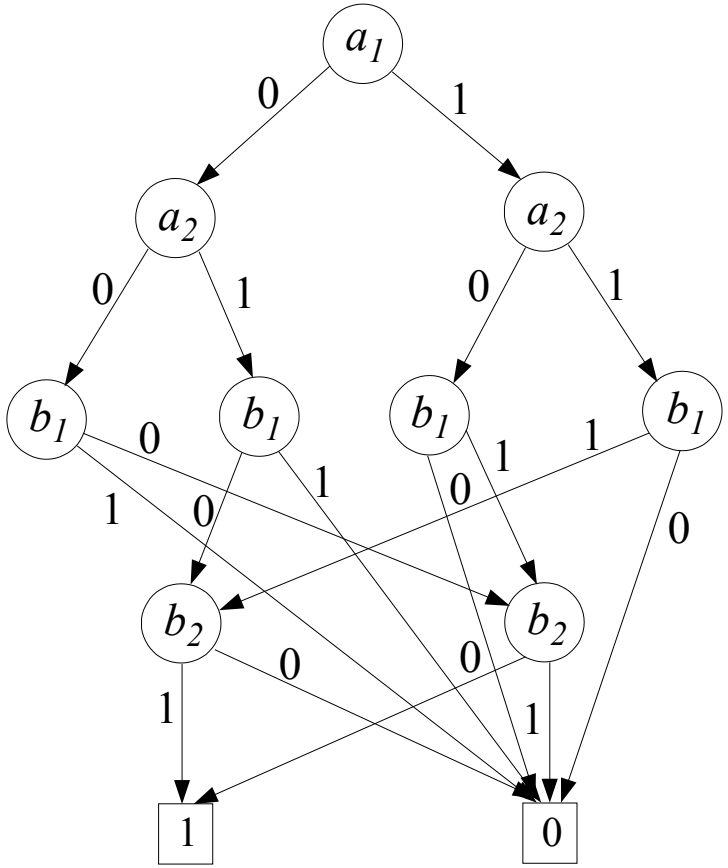
Impact of BDD Order

- Variable order critically decides BDD size.
  - Polynomial vs Exponential
- Finding optimal ordering is computationally costly.
  - Some systems do not have any optimal orderings.
- There are good heuristics to find good orderings.
  - Ex.: group related decision making variables together.
- Variables can also be ordered on-the-fly.

A 2-bit Comparator Example



variable order: (a_1, b_1, a_2, b_2)



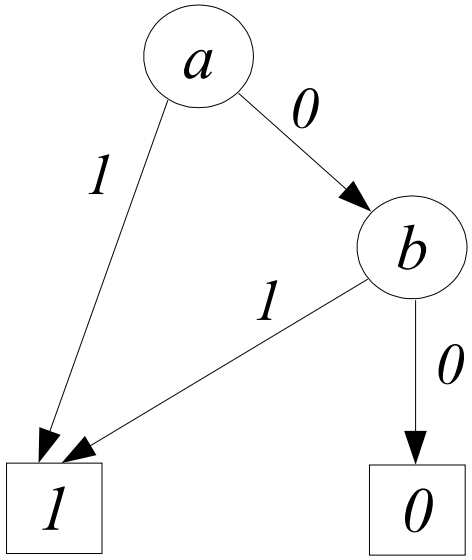
variable order: (a_1, a_2, b_1, b_2)

Importance of Canonical Form

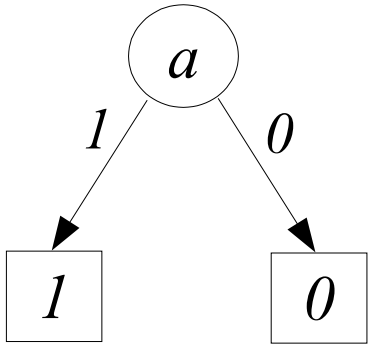
- ORBDDs do not have nodes for redundant variables.
- Semantically equivalent boolean functions are represented by a single ORBDD.
- Easy to check satisfiability and validity.
  - Check satisfiability: ORBDD has 1-terminal.
  - Check validity: ORBDD is 1-terminal.

Algorithms for BDDs

- Restrict: $f[1/x_i], f[0/x_i]$.
  - $f = ab,\ f[1/a] = b,\ f[0/a] = 0.$
  - For BDDs, the node x_i is removed, and its incoming edges are re-directed to $lo(x_i)$ or $hi(x_i)$.



$f[1/b] = 1$



$f[0/b] = a$

Algorithms for BDDs (cont'd)

- Shannon expansion: $f = x \cdot f[1/x] + \neg x \cdot f[0/x]$.
- Boolean operations based on Shannon expansion:
  $f \; op \; g = x \cdot (f[1/x] \; op \; g[1/x]) + \neg x \cdot (f[0/x_i] \; op \; g[0/x_i])$.
  - $B_{f \; op \; g}$ = **apply** (op, B_f, B_g).
- Special handling of negation: swap 0- and 1-terminals.
  - What is the other way to compute nagation?
- A variable is a constraint on a boolean function.
- Function **exists**: de-couples function f from variable x.

  exists(x, f) = $\exists x.\, f = f[1/x] + f[0/x]$.
  - Determines the truth condition of f without constraint x.

Complexity of OBDD Operations

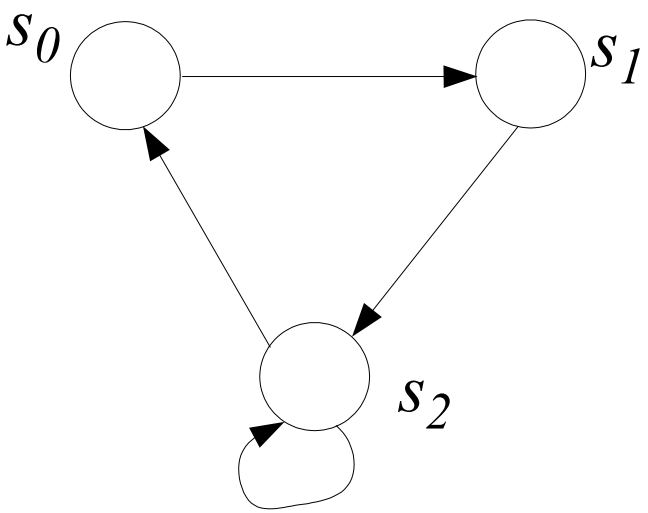
Algorithms	Input OBDDs	Output OBDDs	Time Complexity
reduce	B	reduced B	$O(\|B\| \cdot \log\|B\|)$
apply	reduced B_f, B_g	$B_{f\ op\ g}$	$O(\|B_f\| \cdot \|B_g\|)$
restrict	reduced B_f	reduced $B_{f[0/x]}$ $B_{f[0/x]}$	$O(\|B\| \cdot \log\|B\|)$
exist	reduced B_f	reduced $B_{\exists x1 \ldots \exists xn \cdot f}$	NP-complete

Symbolic Model Checking

- Given $M = (S, T, L)$, S and T are represented as boolean formulas.
  - Enables very large state space to be manipulated.
  - How large a state space can **true** represent?
- Model checking manipulates boolean formulas.
- Underlying data structure is OBDD.
- Very efficient in many cases
  - Can blow up and hard to predict when.
  - Big problem when used in production environment.

Symbolic Representation of States

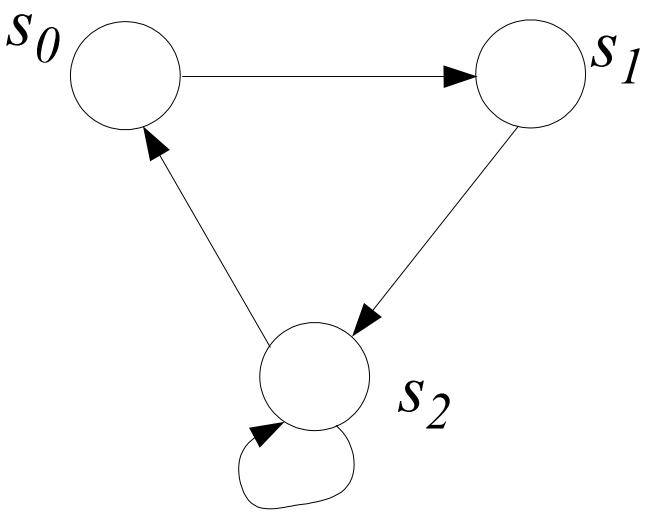
- Encode each state with a distinct binary vector.
  - Let $f(s)$ denote the boolean formula for the binary vector.
  - Requires $k = \log|S|+1$ boolean variables.
- S is representes as $ch(S) = f(s_0)+\ldots+f(s_n)$.



s_0	00	$\neg x_1 \neg x_2$
s_1	01	$\neg x_1 x_2$
s_2	10	$x_1 \neg x_2$
S	$?$	

Symbolic Representation of Transitions

- For $s \rightarrow s'$, two sets of variables are required.
  - One for the current state, and the other for the next state.
  - A state transition $t = f(s) \cdot f(s')$.
- T is represented as $ch(T) = t_0 + \ldots + t_m$.



$$s_0 \rightarrow s_1 \quad \neg x_1 \neg x_2 \neg x'_1 x'_2$$

$$s_1 \rightarrow s_2 \quad \neg x_1 x_2 x'_1 \neg x'_2$$

$$s_2 \rightarrow s_0 \quad x_1 \neg x_2 \neg x'_1 \neg x'_2$$

$$s_2 \rightarrow s_2 \quad x_1 \neg x_2 x'_1 \neg x'_2$$

Model Checking Algorithms

function $\mathbf{SAT_{EX}}(M, f)$
  begin
    $X = \mathbf{SAT}(f)$;
    $Y = \mathbf{pre}\exists(X)$;
    return Y;
end;

function $\mathbf{SAT_{EG}}(M, f)$
  begin
    $Y = \mathbf{SAT}(f)$;
    $X = \varnothing$;
    repeat until $X == Y$ **begin**
      $X = Y$;
      $Y = Y \cap \mathbf{pre}\exists(Y)$;
    return Y;
end;

function $\mathbf{SAT_{EU}}(M, f, g)$
  begin
    $X = \mathbf{SAT}(g)$;
    $Y = \varnothing$;
    $Z = \mathbf{SAT}(f)$;
    repeat until $X == Y$ **begin**
      $Y = X$;
      $X = X \cup (\mathbf{pre}\exists(X) \cap \mathrm{Z})$;
    return Y;
end;

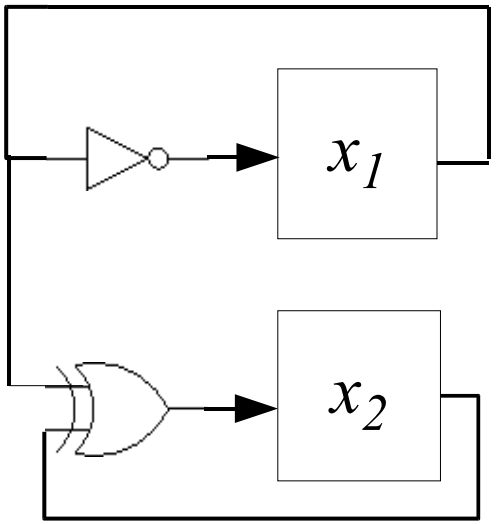
$\mathbf{pre}\exists(S') = \mathbf{exists}(X', \mathbf{apply}(\cdot, B_T, B_{S'}))$

$\mathbf{SAT}(f) = \mathbf{apply}(\cdot, B_S, B_f)$

Synthesizing OBDDs

- Previous approach requires M avaible first.
  - M may be too large.
- Transition relations (TR) can be derived directly from high-level specifications.
  - TR tells how state variables are updated.
- Then OBDDs are generated for TR.
- Let I and O be inputs and outputs.
- For all $x_i \in O$, TR is $x_i \leftrightarrow f_i(I, O)$.
  - $f \leftrightarrow g = 1$ iff f and g compute the same value.
  - $f \leftrightarrow g = \neg f \oplus g$

Modeling Sequential Circuits



- Synchronous circuits: all variables are updated in parallel at the same time. For example:

$$(x'_1 \leftrightarrow \neg x_1) \cdot (x'_2 \leftrightarrow x_1 \oplus x_2)$$

- Asynchronous circuits:
  1. Simultaneous model: variables are updated arbitrarily.
  2. Interleaving model: only one variable is updated at a time.

$$\textit{Simultaneous model}: \prod_{0 \le i \le n} ((x'_i \leftrightarrow f_i) + (x'_i \leftrightarrow x_i))$$

$$\textit{Interleaving model}: \sum_{0 \le i \le n} \left((x'_i \leftrightarrow f_i) \prod_{j \ne i} (x'_i \leftrightarrow x_i) \right)$$

Image Calculation

- Each sequential system can be described with the initial states B_I, and transition relation B_T.
- Reachable states are found with image calculation.

```
image( B_I , B_T ) {
    B_S = B_I ;
    Z = ∅ ;
    while B_S ≠ Z do
        Z = B_S ;
        new = exists( X, apply( · , B_S, B_T ) )[X'/X];
        B_S = apply ( +, B_S, new);
    end while;
    return B_S;
}
```