Reduced Ordered Binary Decision Diagrams

Lecture #12 of Advanced Model Checking

Joost-Pieter Katoen

Lehrstuhl 2: Software Modeling & Verification

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

December 13, 2016

Symbolic CTL model checking

- Explicit representation of transition system: state explosion problem
- Idea: reformulate model-checking in a symbolic way
- Concept: represent sets of states and transitions symbolically
- Approach: binary encoding of states + switching functions for sets
- Compact representation of switching functions by binary decision diagrams
- Alternative: conjunctive normal form (basis for SAT-based model checking)

Basic approach

- let *TS* $= (S, \rightarrow, I, AP, L)$ be a "large" finite transition system
  - the set of actions is irrelevant here and has been omitted, i.e., $\rightarrow \subseteq S \times S$
- For $n \geqslant \lceil \log |S| \rceil$, let injective function *enc* $: S \rightarrow \{0,1\}^n$
  - note: *enc*$(S) = \{0,1\}^n$ is no restriction, as all elements $\{0,1\}^n \setminus$ *enc*(S) can be treated as the encoding of pseudo states that are unreachable
- Identify the states $s \in S =$ *enc*$^{-1}(\{0,1\}^n)$ with *enc*$(s) \in \{0,1\}^n$
- And $T \subseteq S$ by its characteristic function $\chi_T : \{0,1\}^n \rightarrow \{0,1\}$
  - that is $\chi_T($*enc*$(s)) = 1$ if and only if $s \in T$
- And $\rightarrow \subseteq S \times S$ by the Boolean function $\Delta : \{0,1\}^{2n} \rightarrow \{0,1\}$
  - such that $\Delta\left(\textit{enc}(s), \textit{enc}(s')\right) = 1$ if and only if $s \rightarrow s'$

Switching functions

- Let *Var* $= \{z_1, \ldots, z_m\}$ be a finite set of Boolean variables, $m \geqslant 0$
- An evaluation is a function η : *Var* $\to \{0, 1\}$
  - let *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 : *Eval*(*Var*) $\to \{0, 1\}$ is a *switching function* for *Var* $= \{z_1, \ldots, z_m\}$
- Logical operations and quantification are defined as expected
  - $f_1(\cdot) \wedge f_2(\cdot) \ = \ \min\{ f_1(\cdot), f_2(\cdot) \}$
  - $f_1(\cdot) \ \vee \ f_2(\cdot) \ = \ \max\{ f_1(\cdot), f_2(\cdot) \}$
  - $\exists z.\, f(\cdot) \ = \ f(\cdot)|_{z=0} \ \vee \ f(\cdot)|_{z=1}$, and
  - $\forall z.\, f(\cdot) \ = \ f(\cdot)|_{z=0} \wedge f(\cdot)|_{z=1}$

Polynomial-size data structure impossible

- There is no poly-size data structure for all switching functions
  - $|Eval(z_1, \ldots, z_m)| = 2^m$, so #functions $Eval(z_1, \ldots, z_m) \to \{ 0, 1 \}$ is 2^{2^m}

- Suppose there is a data structure that can represent K_m switching functions by at most 2^{m-1} bits

- Then $K_m \leqslant \sum_{i=0}^{2^{m-1}} 2^i = 2^{2^{m-1}+1} - 1 < 2^{2^{m-1}+1}$

- But then there are at least

$$2^{2^m} - 2^{2^{m-1}+1} = 2^{2^{m-1}+1} \cdot \left(2^{2^m - 2^{m-1} - 1} - 1 \right) = 2^{2^{m-1}+1} \cdot \left(2^{2^{m-1}-1} - 1 \right)$$

switching functions whose representation needs more than 2^{m-1} bits

Possible representations of switching functions

- Truth tables
  - very space inefficient: 2^n entries for n variables
  - satisfiability and equivalence check: easy; boolean operations also easy
  - . . . but have to consider exponentially many lines (so are hard)
- . . . in Disjunctive Normal Form (DNF)
  - satisfiability is easy: find a disjunct that does have complementary literals
  - negation and conjunction complicated
  - equivalence checking ($f = g$?) is coNP-complete
- . . . in Conjunctive Normal Form (CNF)
  - satisfiability problem is NP-complete (Cook's theorem)
  - negation and disjunction complicated

Representing switching functions

representation	*compact?*	*sat*	*equi*	$\wedge$	$\vee$	$\neg$
propositional formula	often	hard	hard	easy	easy	easy
DNF	sometimes	easy	hard	hard	easy	hard
CNF	sometimes	hard	hard	easy	hard	hard
(ordered) truth table	never	hard	hard	hard	hard	hard

There is hope . . . . . . perhaps

Nevertheless there are data structures which yield compact representations for many switching functions that appear in practical applications

for hardware circuits, ordered binary decision diagrams (OBDDs) are successful

Representing boolean functions

representation	*compact?*	*sat*	*equ*	$\wedge$	$\vee$	$\neg$
propositional formula	often	hard	hard	easy	easy	easy
DNF	sometimes	easy	hard	hard	easy	hard
CNF	sometimes	hard	hard	easy	hard	hard
(ordered) truth table	never	hard	hard	hard	hard	hard
reduced ordered binary decision diagram	often	easy	easy*	medium	medium	easy

* provided appropriate implementation techniques are used

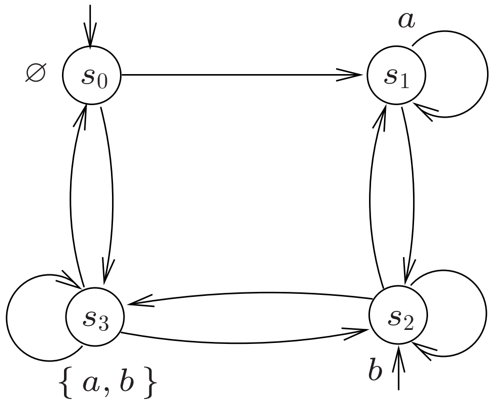
Binary decision tree

- The BDT for function f on *Var* $= \{ z_1, \ldots, z_m \}$ has depth m
  - outgoing edges for node at level i stand for $z_i = 0$ (dashed) and $z_i = 1$ (solid)
- For evaluation $s = [z_1 = b_1, \ldots, z_m = b_m]$, $f(s)$ is the value of the leaf
  - reached by traversing the BDT from the root using branch $z_i = b_i$ for at level i
- The subtree of node v at level i for variable ordering $z_1 < \ldots < z_m$ represents

$$f_v \;=\; f|_{z_1=b_1,\ldots,z_{i-1}=b_{i-1}}$$

  - which is a switching function over $\{ z_i, \ldots, z_m \}$ and
  - where $z_1 = b_1, \ldots, z_{i-1} = b_{i-1}$ is the sequence of decisions made along the path from the root to node v

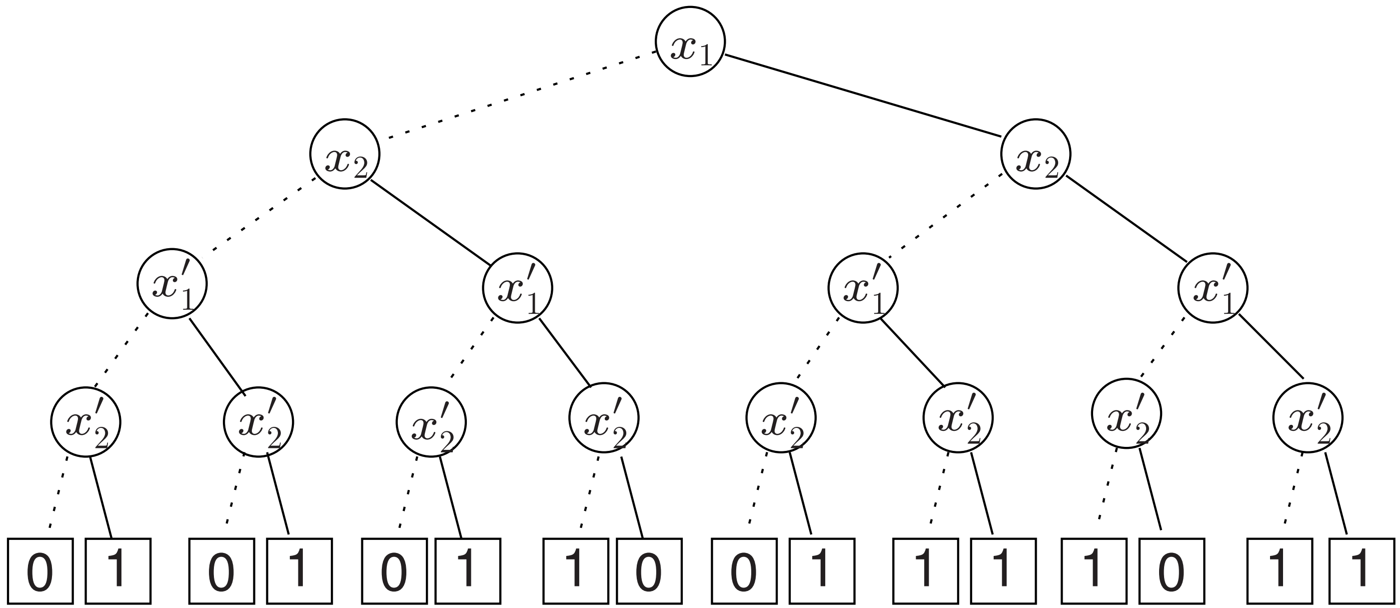
Symbolic representation of a transition system



Switching function: $\Delta(\underbrace{x_1, x_2}_{s}, \underbrace{x_1', x_2'}_{s'}) = 1$ if and only if $s \rightarrow s'$

$$
\begin{array}{rcl}
\Delta(x_1, x_2, x_1', x_2') = & & (\neg x_1 \wedge \neg x_2 \wedge \neg x_1' \wedge x_2') \\
& \vee & (\neg x_1 \wedge \neg x_2 \wedge x_1' \wedge x_2') \\
& \vee & (\neg x_1 \wedge x_2 \wedge x_1' \wedge \neg x_2') \\
& \vee & \ldots \\
& \vee & (x_1 \wedge x_2 \wedge x_1' \wedge x_2')
\end{array}
$$

Transition relation as a BDT



A BDT representing Δ for our example using ordering $x_1 < x_2 < x_1' < x_2'$

Facts about BDTs

- BDTs are not compact
  - a BDT for switching function f on n variables has 2^n leafs

  $\Rightarrow$ they are as space inefficient as truth tables!

$\Rightarrow$ BDTs contain quite some redundancy

  - all leafs with value one (zero) could be collapsed into a single leaf
  - a similar scheme could be adopted for isomorphic subtrees

- The size of a BDT does not change if the variable order changes

Ordered Binary Decision Diagram

- OBDDs rely on compactifying BDT representations
- Idea: skip redundant fragments of BDT representations
- Collapse subtrees with all terminals having same value
- Identify nodes with isomorphic sub-trees
- This yields directed acyclic graphs with out-degree two
- Inner nodes are labeled with variables
- Leafs are labeled with function values (zero and one)

Ordered Binary Decision Diagram

Let $\wp$ be a (total) variable ordering for *Var* where $\wp = (z_1, \ldots, z_m)$.

An $\wp$-OBDD is a tuple $\mathfrak{B} = (V, V_I, V_T, \mathit{succ}_0, \mathit{succ}_1, \mathit{var}, \mathit{val}, v_0)$ with

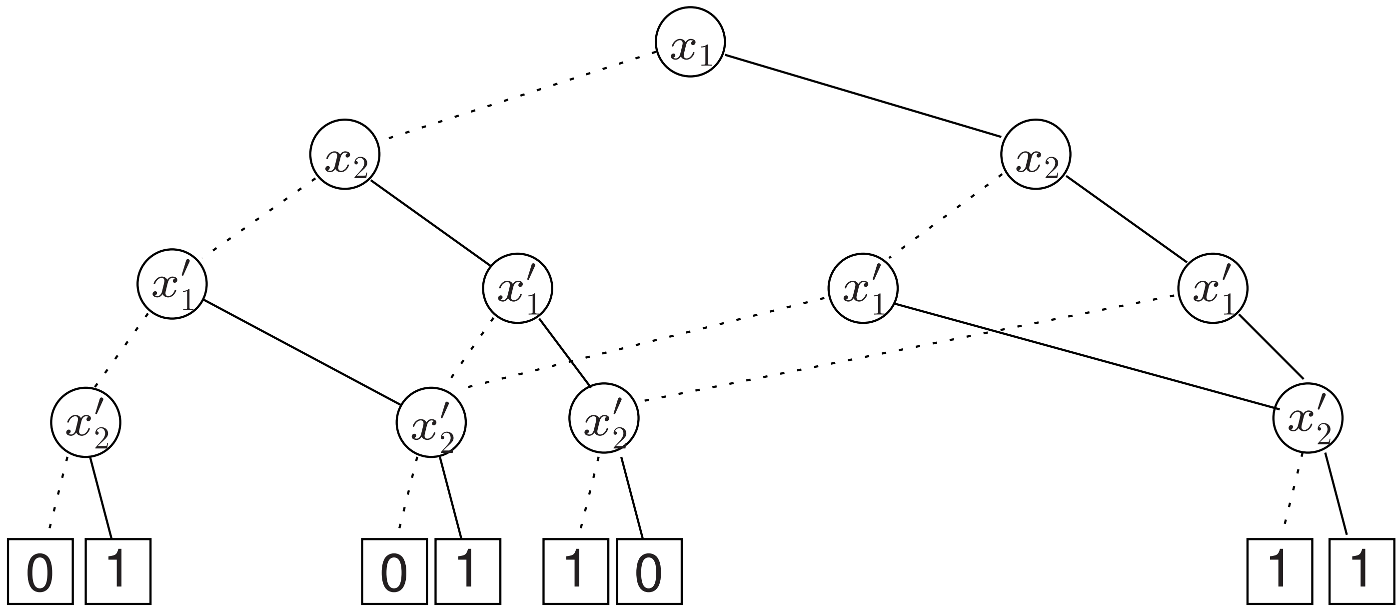
- a finite set V of nodes, partitioned into V_I (inner) and V_T (terminals)
  - and a distinguished root (node) $v_0 \in V$
- successor functions $\mathit{succ}_0, \mathit{succ}_1 : V_I \rightarrow V$
  - such that each node $v \in V \setminus \{ v_0 \}$ has at least one predecessor
  - i.e., all nodes of the OBDD $\mathfrak{B}$ are reachable from the root
- a labeling functions $\mathit{var} : V_I \rightarrow \mathit{Var}$ and $\mathit{val} : V_T \rightarrow \{ 0, 1 \}$

satisfying for $\wp = (z_1, \ldots, z_m)$ and $v \in V_I$:

$$\mathit{var}(v) = z_i \;\wedge\; w \in \{ \mathit{succ}_0(v), \mathit{succ}_1(v) \} \cap V_I \;\Rightarrow\; \mathit{var}(w) = z_j \text{ for } j > i$$

Some example OBDDs

Transition relation as an OBDD



An example OBDD representing $f_{\rightarrow}$ for our example using $x_1 < x_2 < x_1' < x_2'$

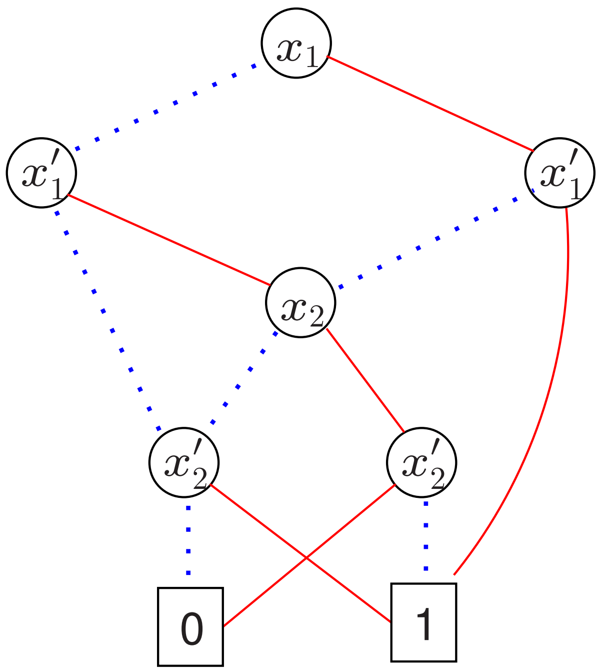
Semantics of an OBDD

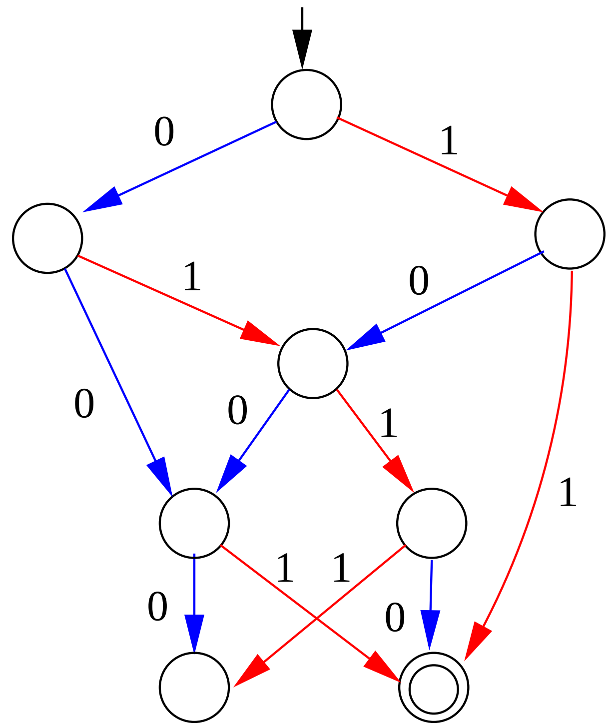
The semantics of $\wp$-OBDD $\mathfrak{B}$ is the switching function $f_{\mathfrak{B}}$ where

$$f_{\mathfrak{B}}([z_1 = b_1, \ldots, z_m = b_m])$$

is the value of the resulting leaf when traversing $\mathfrak{B}$ starting in v_0 and branching according to the evaluation $[z_1 = b_1, \ldots, z_m = b_m]$

Intermezzo: OBDDs versus DFA





each OBDD $\mathfrak{B}$ is a deterministic automaton $\mathcal{A}_{\mathfrak{B}}$ with $f_{\mathfrak{B}}^{-1}(1) = L(\mathcal{A}_{\mathfrak{B}})$

Bottom-up characterization of $f_{\mathfrak{B}}$

Let $\mathfrak{B}$ be a $\wp$-OBDD. Switching function f_v for node $v \in V$:

- If $v \in V_T$, then f_v is the constant switching function with value *val*(v)
- If $v \in V_I$ with *var*$(v) = z$, then $f_v \; = \; \underbrace{\big(\neg z \wedge f_{\mathit{succ}_0(v)}\big) \;\vee\; \big(z \wedge f_{\mathit{succ}_1(v)}\big)}_{\text{Shannon expansion}}$

Furthermore, $f_{\mathfrak{B}} \; = \; f_{v_0}$ for the root v_0 of $\mathfrak{B}$

Consistent co-factors in OBDDs

- Let f be a switching function for *Var*
- Let $\wp = (z_1, \ldots, z_m)$ a variable ordering for *Var*, i.e., $z_1 <_\wp \ldots <_\wp z_m$
- Switching function g is a *$\wp$-consistent cofactor* of f if

$$g = f|_{z_1=b_1,\ldots,z_i=b_i} \quad \text{for some } i \in \{0, 1, \ldots, m\}$$

- It holds that:
  1. for each node v of an $\wp$-OBDD $\mathfrak{B}$, f_v is a $\wp$-consistent cofactor of $f_{\mathfrak{B}}$
  2. for each $\wp$-consistent cofactor g of $f_{\mathfrak{B}}$ there is a node $v \in \mathfrak{B}$ with $f_v = g$

Reduced OBDDs

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

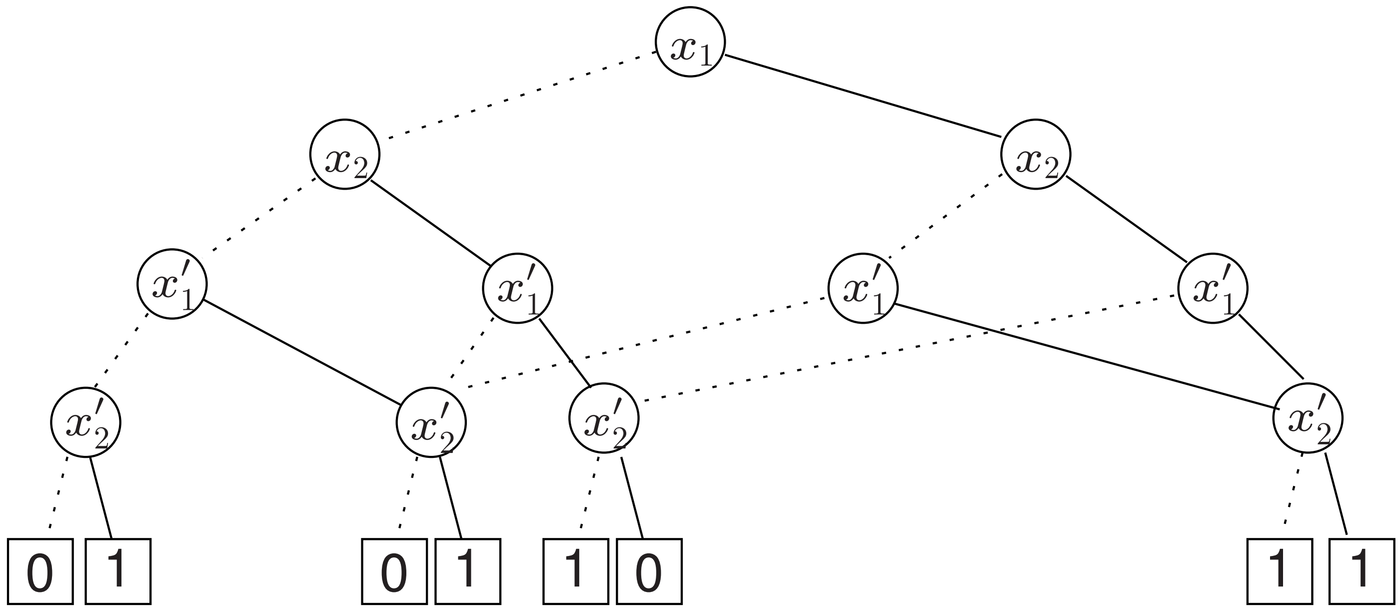
$v \neq w$ implies $f_v \neq f_w$

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

In $\Rightarrow$ $\wp$-ROBDDs any $\wp$-consistent cofactor is represented by exactly one node

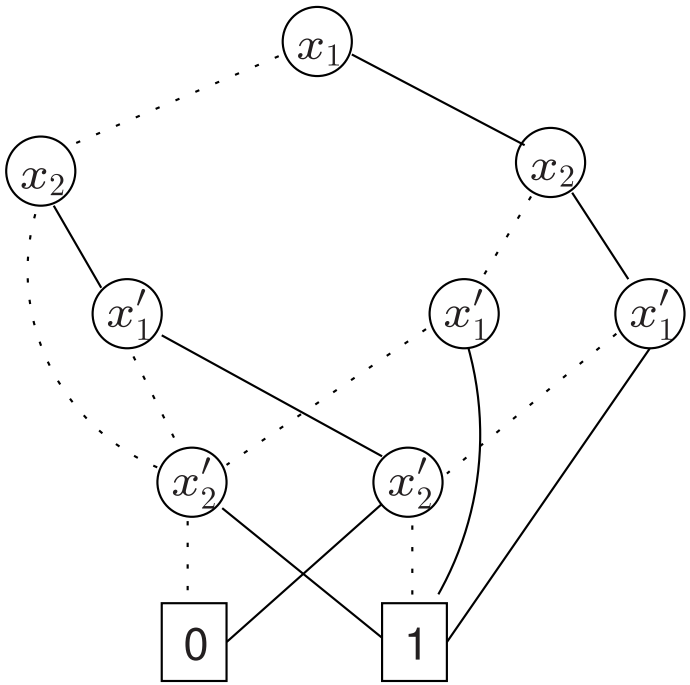
Example ROBDDs

Transition relation as an OBDD

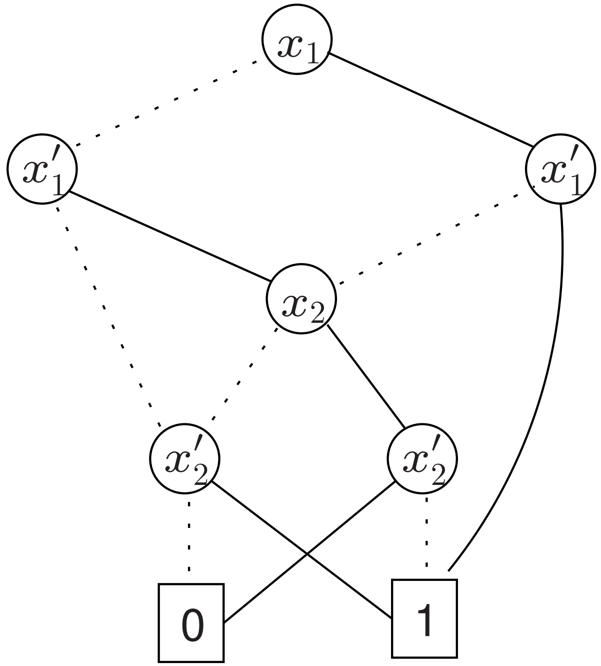


An example OBDD representing $f_{\rightarrow}$ for our example using $x_1 < x_2 < x_1' < x_2'$

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

[Fortune, Hopcroft & Schmidt, 1978]

For finite set *Var* of Boolean variables and $\wp$ a variable ordering for *Var*:

(a) For each switching function f on *Var*
there exists a $\wp$-ROBDD $\mathfrak{B}$ with $f_{\mathfrak{B}} = f$

(b) For any $\wp$-ROBDDs $\mathfrak{B}$ and $\mathfrak{C}$ with $f_{\mathfrak{B}} = f_{\mathfrak{C}}$,
$\mathfrak{B}$ and $\mathfrak{C}$ are isomorphic, i.e., agree up to renaming of nodes

Proof

The importance of canonicity

- Absence of redundant vertices
  - if $f_{\mathfrak{B}}$ does not depend on x_i, ROBDD $\mathfrak{B}$ does not contain an x_i node
- Test for equivalence: $f(x_1, \ldots, x_n) \equiv g(x_1, \ldots, x_n)$?
  - generate ROBDDs $\mathfrak{B}_f$ and $\mathfrak{B}_g$, and check isomorphism
- Test for validity: for all $x_1, \ldots, x_n$, is $f(x_1, \ldots, x_n) = 1$?
  - generate ROBDD $\mathfrak{B}_f$ and check whether it only consists of a 1-leaf
- Test for implication: $f(x_1, \ldots, x_n) \rightarrow g(x_1, \ldots, x_n)$?
  - generate ROBDD $\mathfrak{B}_{f \wedge \neg g}$ and check if it just consists of a 0-leaf
- Test for satisfiability
  - f is satisfiable if and only if $\mathfrak{B}_f$ has a reachable 1-leaf

Minimality of ROBDDs

Let $\mathfrak{B}$ be an $\wp$-OBDD for f.

Then: $\mathfrak{B}$ is reduced iff *size*$(\mathfrak{B}) \leqslant$ *size*$(\mathfrak{C})$ for each $\wp$-OBDD $\mathfrak{C}$ for f.

This follows from the fact that:

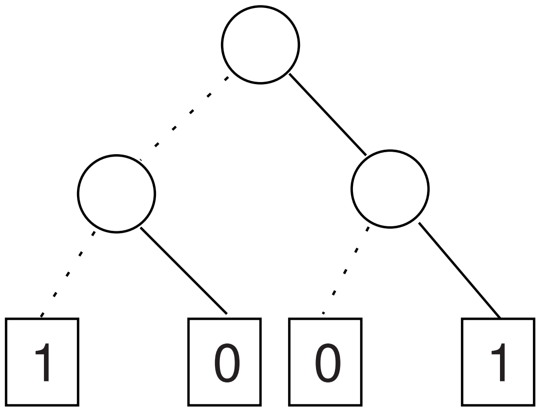
1. Each $\wp$-consistent cofactor of f is represented in any $\wp$-OBDD for f by at least one node, and
2. A $\wp$-OBDD $\mathfrak{B}$ for f is reduced iff there is a 1-to-1 correspondence between the nodes in $\mathfrak{B}$ and the $\wp$-consistent cofactors of $\mathfrak{B}$.

Reducing OBDDs

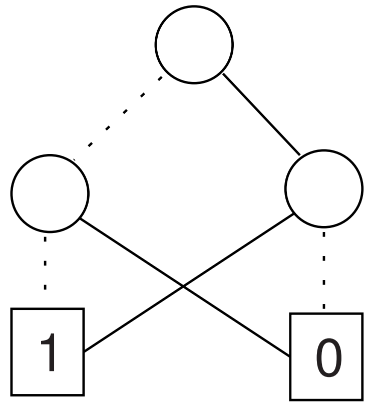
- Generate an OBDD (or BDT) for a boolean expression, then reduce
  - by means of a recursive descent over the OBDD
- Elimination of duplicate leafs
  - for a duplicate 0-leaf (or 1-leaf), redirect all incoming edges to just one of them
- Elimination of “don’t care” (non-leaf) vertices
  - if $\mathit{succ}_0(v) = \mathit{succ}_1(v) = w$, 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?

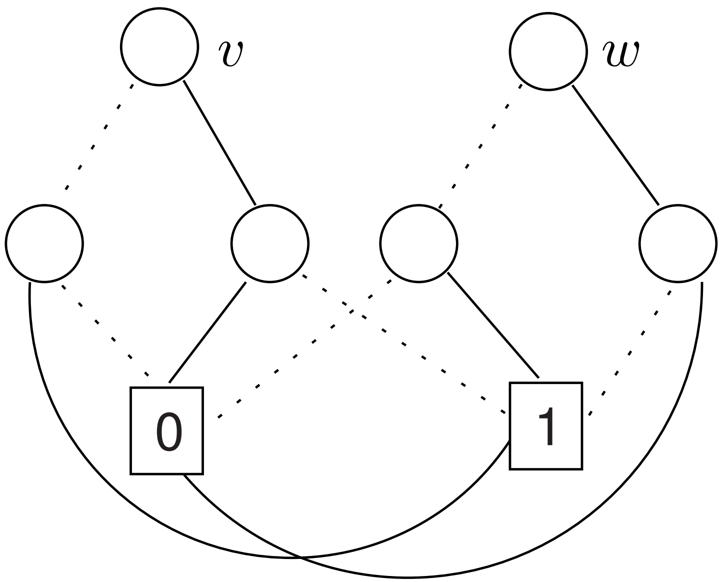


becomes

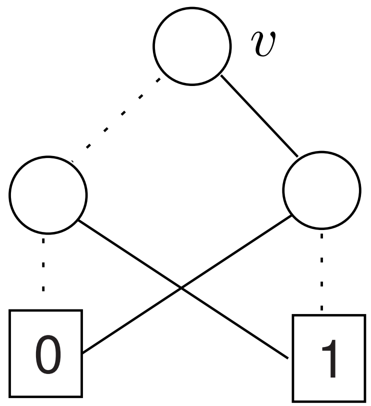


(special case of) isomorphism rule

How to reduce an OBDD?

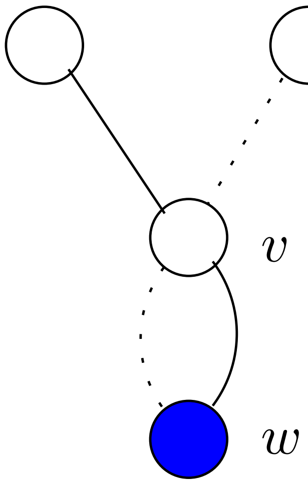


becomes

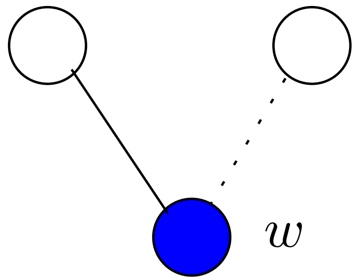


isomorphism rule

How to reduce an OBDD?



becomes



elimination rule

Example

Soundness of reduction rules

if $\mathfrak{C}$ arises from a $\wp$-OBDD $\mathfrak{B}$ by the elimination or isomorphism rule, then:

$\mathfrak{C}$ is a $\wp$-OBDD with $f_{\mathfrak{B}} = f_{\mathfrak{C}}$

Elimination rule for v with $var(v) = z$, and $w = succ_0(v) = succ_1(v)$:

$$f_v = \big(\neg z \wedge f_{succ_0(v)}\big) \vee \big(z \wedge f_{succ_1(v)}\big) = (\neg z \wedge f_w) \vee (z \wedge f_w) = f_w$$

Isomorphism rule for v, w with $var(v) = var(w) = z$ yields:

$$\begin{aligned} f_v &= \big(\neg z \wedge f_{succ_0(v)}\big) \vee \big(z \wedge f_{succ_1(v)}\big) \\ &= \big(\neg z \wedge f_{succ_0(w)}\big) \vee \big(z \wedge f_{succ_1(w)}\big) \\ &= f_w \end{aligned}$$

as each reduction rule decreases the $\#$ nodes, repeatedly applying them terminates

Completeness of reduction rules

$\wp$-OBDD $\mathfrak{B}$ is reduced if and only if

no reduction rule is applicable to $\mathfrak{B}$