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 7, 2006

Boolean functions

- Boolean functions $f : \mathbb{B}^n \to \mathbb{B}$ for $n \geqslant 0$ where $\mathbb{B} = \{ 0, 1 \}$
  - examples: $f(x_1, x_2) = x_1 \wedge (x_2 \ \vee \ \neg x_1)$, and $f(x_1, x_2) = x_1 \leftrightarrow x_2$
- Finite sets are boolean functions
  - let $|S| = N$ and $2^{n-1} < N \leqslant 2^n$
  - encode any element $s \in S$ as boolean vector of length n: $[\![\ ]\!] : S \to \mathbb{B}^n$
  - $T \subseteq S$ is represented by f_T such that:

$$f_T([\![s]\!]) = 1 \quad \text{iff} \quad s \in T$$

  - this is the characteristic function of T
- Relations are boolean functions
  - $\mathcal{R} \subseteq S \times S$ is represented by $f_{\mathcal{R}}$ such that:

$$f_R([\![s]\!], [\![t]\!]) = 1 \quad \text{iff} \quad (s, t) \in \mathcal{R}$$

Representing boolean functions

- Truth tables
  - very space inefficient (2^n lines)
  - satisfiability and equivalence check: easy; boolean operations also easy
  - . . . but have to consider exponentially many lines (so are hard)
- Propositional formulas
  - more compact representation
  - satisfiability problem is NP-complete (Cook's theorem)
  - boolean operations are just syntactic operations
- . . . in Disjunctive Normal Form (DNF)
  - satisfiability is easy: find a disjunct that does have complementary literals
  - negation expensive (dnf of $\neg\Phi$ may be exponentially longer than Φ)
  - conjunction complicated ($\Phi \wedge (\Psi_1 \vee \Psi_2) \equiv (\Phi \wedge \Psi_1) \vee (\Phi \wedge \Psi_2)$
- . . . in Conjunctive Normal Form (CNF)

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	easy	easy	hard	hard
(ordered) truth table	never	hard	hard	hard	hard	hard

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	easy	easy	hard	hard
(ordered) truth table	never	hard	hard	hard	hard	hard
reduced ordered binary decision diagram	often	easy	easy	medium	medium	easy

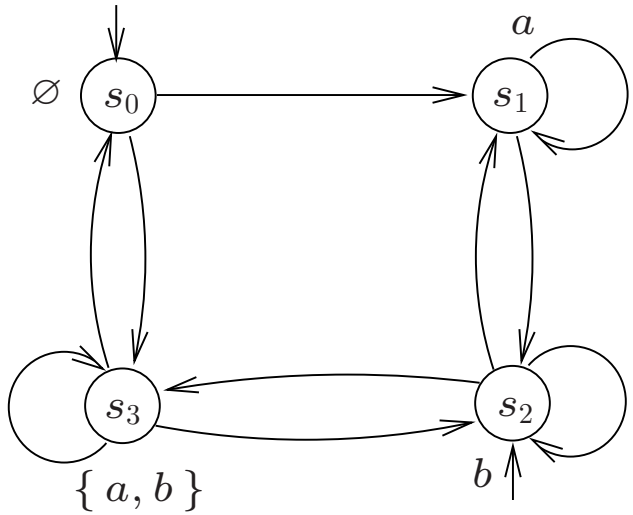
Explicitly representing transition systems

$TS = (S, Act, \rightarrow, I, AP, L)$ with $|S| = N$, $|Act| = M$ and $|AP| = K$:

- Identify the N states by numbers
- Represent the set of initial states I as boolean vector $\underline{i}$
  - $\underline{i}(s_j) = 1$ if and only if state $s_j \in I$
- Represent $\xrightarrow{\alpha}$ by M boolean matrices $\mathbf{T}_\alpha$ of size $N \times N$
  - $\mathbf{T}_\alpha(s_i, s_j) = 1$ if and only if $s_i \xrightarrow{\alpha} s_j$
- Represent L by an $N \times K$-boolean matrix $\mathbf{L}$
  - $\mathbf{L}(s_i, a_j) = 1$ if and only if $a_j \in L(s_i)$

$\Rightarrow$ Use sparse matrix representations for $\mathbf{T}$ and $\mathbf{L}$

Example (no actions)



$$\underline{i} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \quad \text{and} \quad \mathbf{T} = \begin{pmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{L} = \begin{pmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}$$

for simplicity, actions are omitted here

Transition systems as boolean functions

- Assume each state is uniquely labeled
  - $L(s) = L(s')$ implies $s = s'$
  - no restriction: if needed extend *AP* and label states uniquely
- Assume a fixed total order on propositions: $a_1 < a_2 < \ldots < a_K$
- Represent a state by a *boolean function*
  - over the boolean variables x_1 through x_K such that

$$[\![s]\!] = x_1^* \wedge x_2^* \wedge \ldots \wedge x_K^*$$

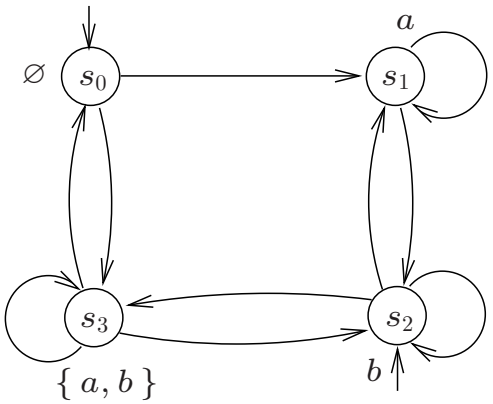
  - where the literal x_i^* equals x_i if $a_i \in L(s)$, and $\neg\, x_i$ otherwise

$\Rightarrow$ no need to explicitly represent function L

- Represent I and $\rightarrow$ by their characteristic (boolean) functions
  - e.g., $f_{\rightarrow}([\![s]\!], [\![\alpha]\!], [\![t]\!]) = 1$ if and only if $s \xrightarrow{\alpha} t$

A small example

An example (no actions)



state	*bit-vector*	*boolean function*
s_0	$\langle 0, 0 \rangle$	$\neg\, x_1 \wedge \neg\, x_2$
s_1	$\langle 0, 1 \rangle$	$\neg\, x_1 \wedge x_2$
s_2	$\langle 1, 0 \rangle$	$x_1 \wedge \neg\, x_2$
s_3	$\langle 1, 1 \rangle$	$x_1 \wedge x_2$

- States:
- Initial states:

$$f_I(x_1, x_2) = (\neg\, x_1 \wedge \neg\, x_2) \;\vee\; (x_1 \wedge \neg\, x_2)$$

Example (continued)

- Transition relation:

$f_{\rightarrow}$	$\langle 0,0\rangle$	$\langle 0,1\rangle$	$\langle 1,0\rangle$	$\langle 1,1\rangle$
$\langle 0,0\rangle$	0	1	0	1
$\langle 0,1\rangle$	0	1	1	0
$\langle 1,0\rangle$	0	1	1	1
$\langle 1,1\rangle$	1	0	1	1

- Alternatively: $f_{\rightarrow}(\underbrace{x_1, x_2}_{s}, \underbrace{x_1', x_2'}_{s'}) = 1$ if and only if $s \rightarrow s'$

$$
\begin{aligned}
f_{\rightarrow}(x_1, x_2, x_1', x_2') = \quad & (\neg x_1 \wedge \neg x_2 \wedge \neg x_1' \wedge x_2') \\
\vee \quad & (\neg x_1 \wedge \neg x_2 \wedge x_1' \wedge x_2') \\
\vee \quad & (\neg x_1 \wedge x_2 \wedge x_1' \wedge \neg x_2') \\
\vee \quad & \ldots \\
\vee \quad & (x_1 \wedge x_2 \wedge x_1' \wedge x_2')
\end{aligned}
$$

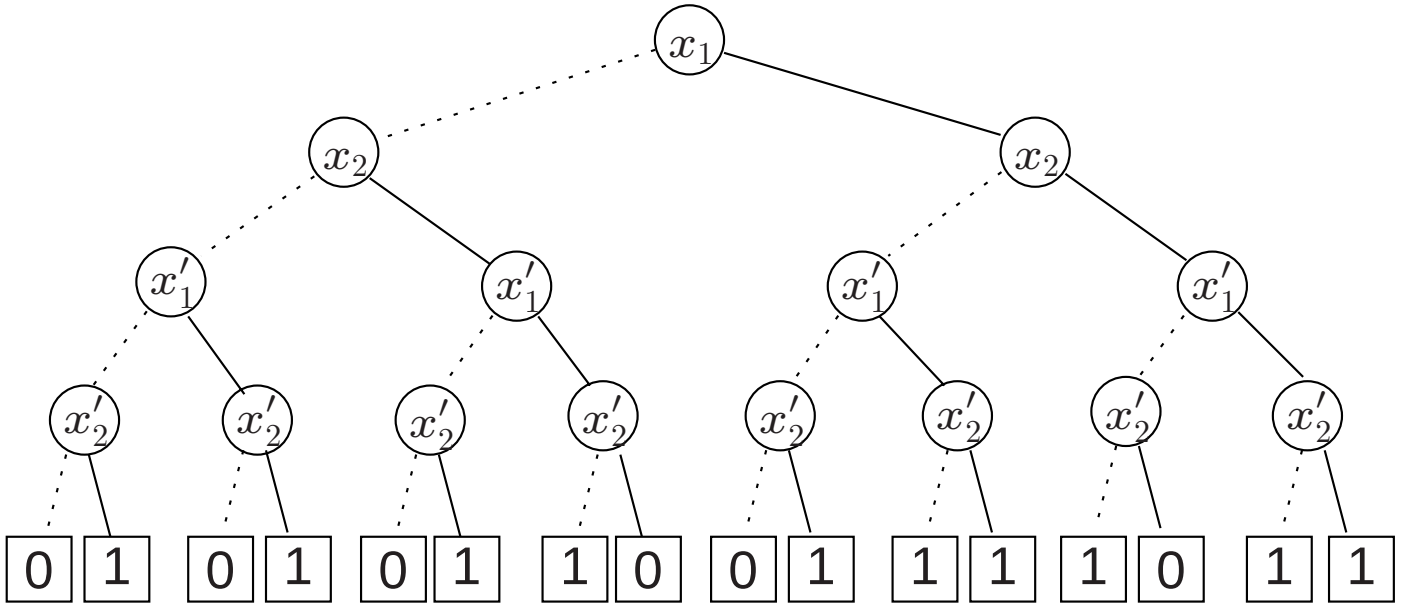
Binary decision trees

- Let X be a set of boolean variables and $<$ a total order on X
- Binary decision tree (BDT) is a complete binary tree over $\langle X, < \rangle$
  - each leaf v is labeled with a boolean value $\mathit{val}(v) \in \mathbb{B}$
  - non-leaf v is labeled by a boolean variable $\mathit{Var}(v) \in X$
  - such that for each non-leaf v and vertex w:

$$w \in \{\, \mathit{left}(v), \mathit{right}(v) \,\} \;\Rightarrow\; (\mathit{Var}(v) < \mathit{Var}(w) \;\vee\; w \text{ is a leaf})$$

$\Rightarrow$ On each path from root to leaf, variables occur in the same order

Transition relation as a BDT



A BDT representing $f_{\rightarrow}$ for our example using $x_1 < x_2 < x'_1 < x'_2$

Branching program

on the black board

any boolean expression is equivalent to an expression in if-then-else (ITE) normal form

in particular $f \equiv \mathbf{if}\ x\ \mathbf{then}\ f[x := 1]\ \mathbf{else}\ f[x := 0]$

$\Rightarrow$ *the Shannon expansion of* f *with respect to* x

Shannon expansion

- Each boolean function $f : \mathbb{B}^n \longrightarrow \mathbb{B}$ can be written as:

$$f(x_1, \ldots, x_n) = (x_i \ \wedge \ f[x_i := 1]) \ \vee \ (\neg\, x_i \ \wedge \ f[x_i := 0])$$

  - where $f[x_i := 1]$ stands for $f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n)$
  - and $f[x_i := 0]$ is a shorthand for $f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n)$

- The boolean function $f_{\mathsf{B}}(v)$ represented by vertex v in BDT B is:

  - for v a leaf: $f_{\mathsf{B}}(v) = \mathit{val}(v)$
  - otherwise:

$$f_{\mathsf{B}}(v) = (\mathit{Var}(v) \ \wedge \ f_{\mathsf{B}}(\mathit{right}(v))) \ \vee \ (\neg\, \mathit{Var}(v) \ \wedge \ f_{\mathsf{B}}(\mathit{left}(v)))$$

- $f_{\mathsf{B}} = f_{\mathsf{B}}(v)$ where v is the root of B

Example

Considerations on BDTs

- BDTs are not compact
  - a BDT for boolean function $f : \mathbb{B}^b \rightarrow \mathbb{B}$ 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

share equivalent expressions [Akers 76, Lee 59]

- Binary decision diagram (OBDD) is a directed graph over $\langle X, < \rangle$ with:
  - each leaf v is labeled with a boolean value *val*$(v) \in \{ 0, 1 \}$
  - non-leaf v is labeled by a boolean variable *Var*$(v) \in X$
  - such that for each non-leaf v and vertex w:

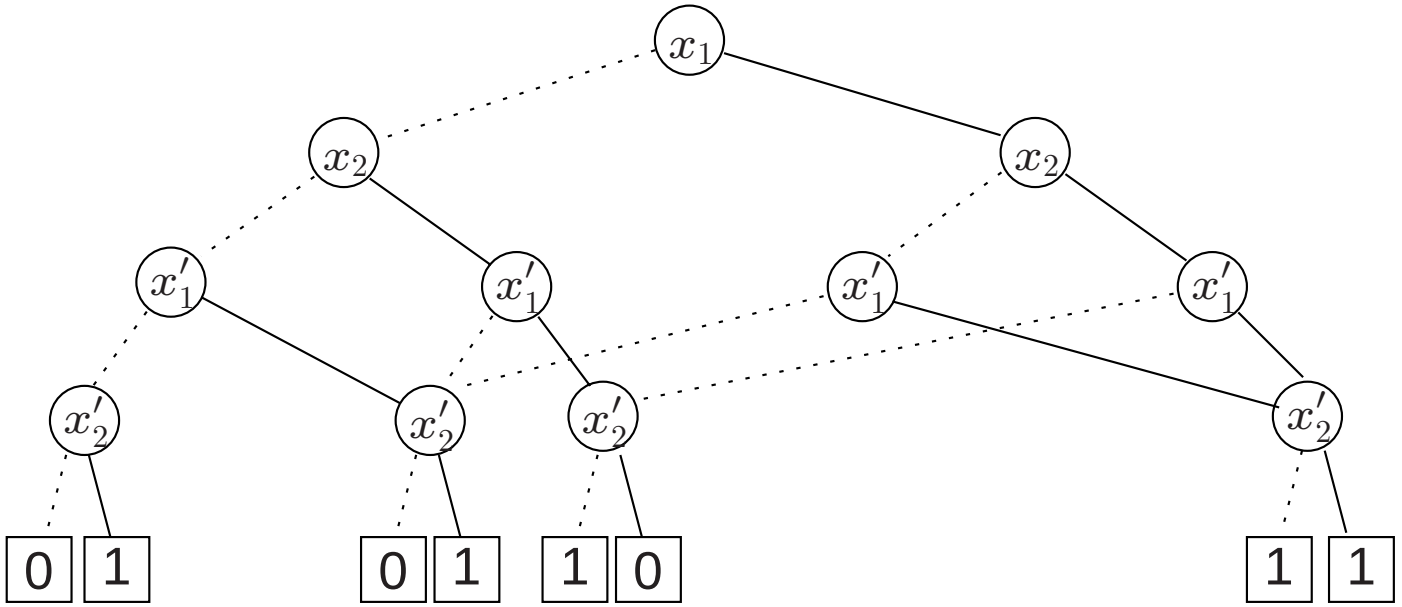
$$w \in \{ \mathit{left}(v), \mathit{right}(v) \} \Rightarrow (\mathit{Var}(v) < \mathit{Var}(w) \vee w \text{ is a leaf})$$

$\Rightarrow$ An OBDD is acyclic

  - f_{B} for OBDD B is obtained as for BDTs

Some example OBDDs

Transition relation as an OBDD



An example OBDD representing $f_{\rightarrow}$ for our example using $x_1 < x_2 < x'_1 < x'_2$

Isomorphism

- B and B′ over $\langle X, < \rangle$ are *isomorphic* iff their roots are isomorphic
- Vertices v in B and w in B′ are isomorphic, denoted $v \cong w$, iff
  there exists a bijection H between the vertices of B and B′ such that:
  1. if v is a leaf, then $H(v) = w$ is a leaf with $\mathit{val}(v) = \mathit{val}(H(v))$
  2. if v is a non-leaf, then $H(v) = w$ is a non-leaf such that

$$\mathit{Var}(v) = \mathit{Var}(w) \ \wedge \ H(\mathit{left}(v)) = \mathit{left}(H(v)) \ \wedge \ H(\mathit{right}(v)) = \mathit{right}(H(v))$$

- Testing B $\cong$ B′ can be done in linear time
  - due to the labels (0 and 1) of the edges.

Reduced OBDDs

OBDD B over $\langle X, < \rangle$ is called *reduced* iff:

1. for each leaf v, w: $(\mathit{val}(v) = \mathit{val}(w)) \Rightarrow v = w$

   $\Rightarrow$ identical terminal vertices are forbidden

2. for each non-leaf v: $\mathit{left}(v) \neq \mathit{right}(v)$

   $\Rightarrow$ non-leafs may not have identical children

3. for each non-leaf v, w:

$$(\mathit{Var}(v) = \mathit{Var}(w) \wedge \mathit{right}(v) \cong \mathit{right}(w) \wedge \mathit{left}(v) \cong \mathit{left}(w)) \Rightarrow v = w$$

   $\Rightarrow$ vertices may not have isomorphic sub-dags

this is what is mostly called BDD; in fact it is an ROBDD!

Example ROBDDs

Dynamic generation of ROBDDs

Main idea:

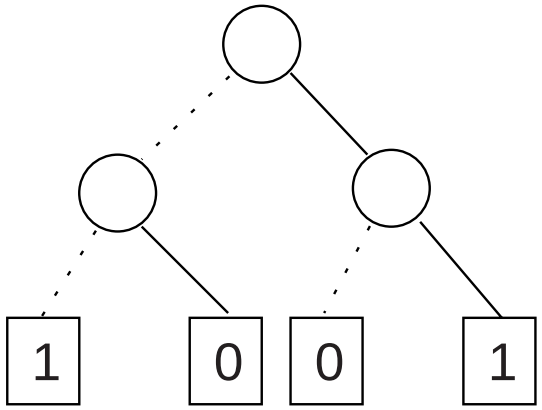
- Construct directly an ROBDD from a boolean expression
- Create vertices in depth-first search order
- On-the-fly reduction by applying hashing
  - on encountering a new vertex v, check whether:
  - an equivalent vertex w has been created (same label and children)
  - *left*$(v) =$ *right*(v), i.e., vertex v is a “don’t care” vertex

Example and algorithm

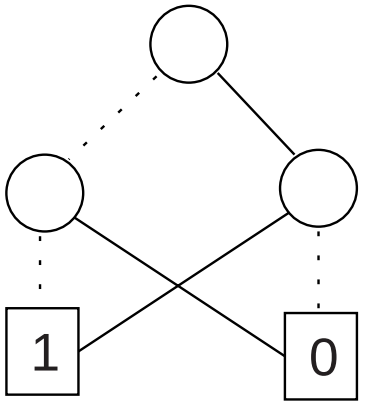
Reducing OBDDs

- 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 *left*$(v) =$ *right*$(v) = w$, eliminate 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

How to reduce an OBDD?

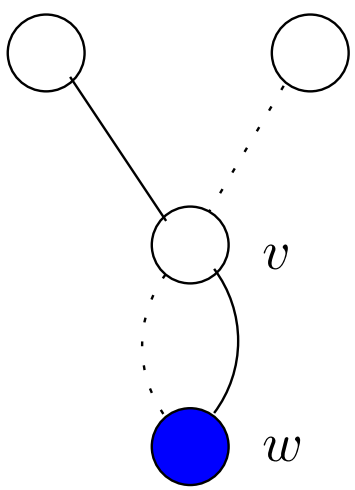


becomes

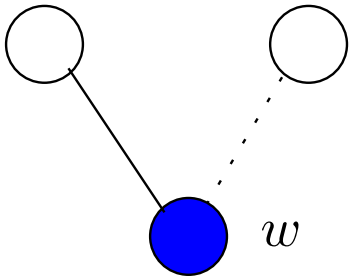


eliminating identical leafs

How to reduce an OBDD?

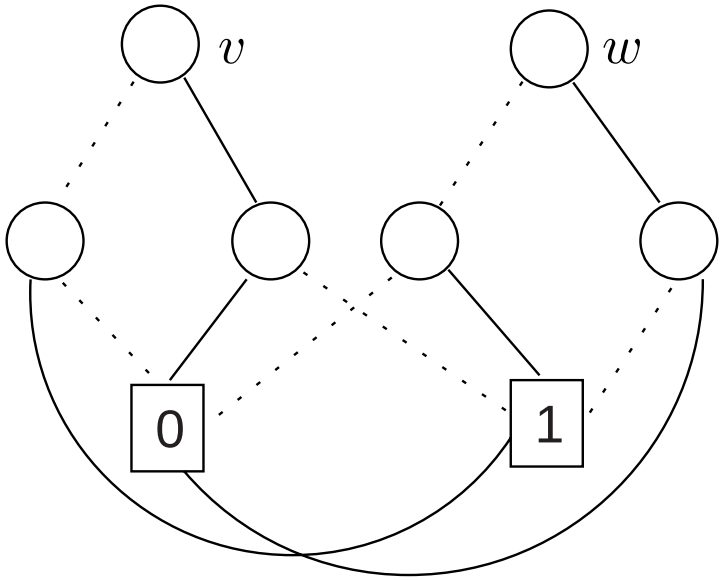


becomes

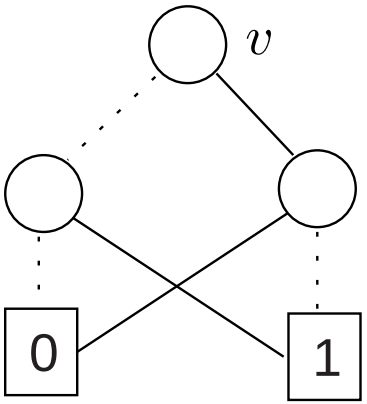


eliminating “don’t care” vertices

How to reduce a BDD?

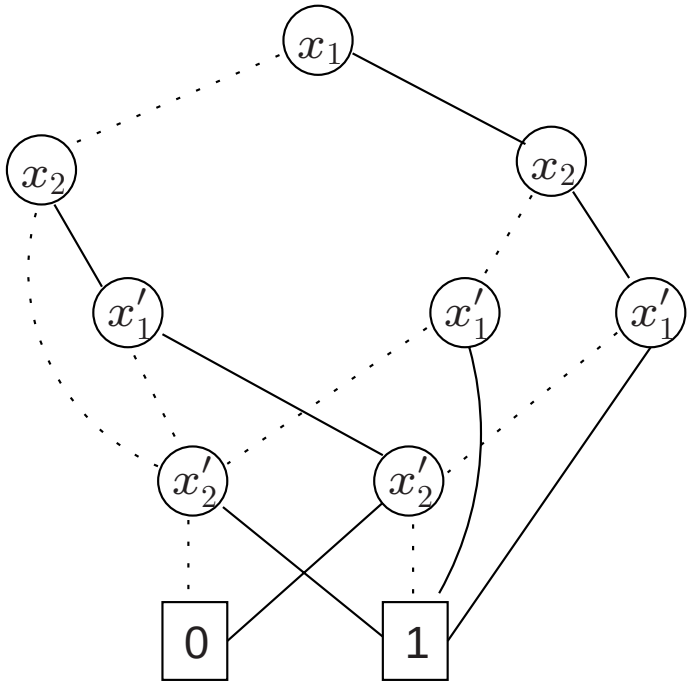


becomes

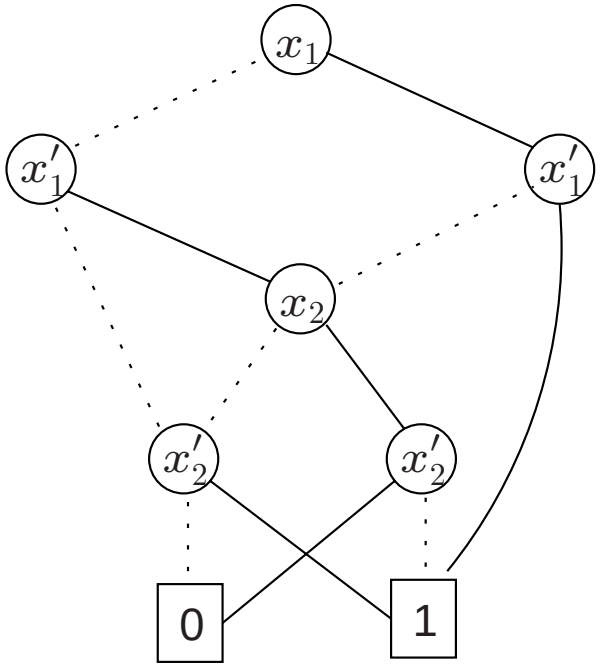


eliminating isomorphic subtrees

Transition relation as an ROBDD



(a) ordering $x_1 < x_2 < x_1' < x_2'$



(b) ordering $x_1 <' x_1' <' x_2 <' x_2'$

ROBDDs are canonical

[Fortune, Hopcroft & Schmidt, 1978]

For ROBDDs B and B′ over $\langle X, < \rangle$ we have:

$(f_{\mathsf{B}} = f_{\mathsf{B}'})$ implies B and B′ are isomorphic

$\Rightarrow$ *for a fixed variable ordering, any boolean function can be uniquely represented by an ROBDD (up to isomorphism)*

The importance of canonicity

- Absence of redundant vertices
  - if f_{B} does not depend on x_i, ROBDD B does not contain an x_i vertex
- Test for equivalence: $f(x_1, \ldots, x_n) \equiv g(x_1, \ldots, x_n)$?
  - generate ROBDDs B_f and B_g, and check isomorphism
- Test for validity: $f(x_1, \ldots, x_n) = 1$?
  - generate ROBDD 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 $\mathsf{B}_f \wedge \neg\mathsf{B}_g$ and check if it just consist of a 0-leaf
- Test for satisfiability
  - f is satisfiable if and only if B_f is not just the 1-leaf