Decision Procedures and Verification

Martin Blicha

Charles University

19.3.2018

Binary Decision Diagrams (BDDs)

Representation of Boolean functions

- Propositional formulas represent Boolean function
  - φ over n variables defines a Boolean function

$$B : \{0,1\}^n \to \{0,1\}$$

- Single Boolean function has infinitely many representations as propositional formula
  - $\varphi = x_1 \wedge (x_2 \vee x_3)$ and $\psi = (x_1 \wedge x_2) \vee (x_1 \wedge x_3)$
  - φ and ψ are *equivalent* formulas, but *different syntactically*
- Unambiguous representation required:
  - One Boolean function $\to$ (syntactically) single representation

Binary decision tree

Definition (Binary decision tree)

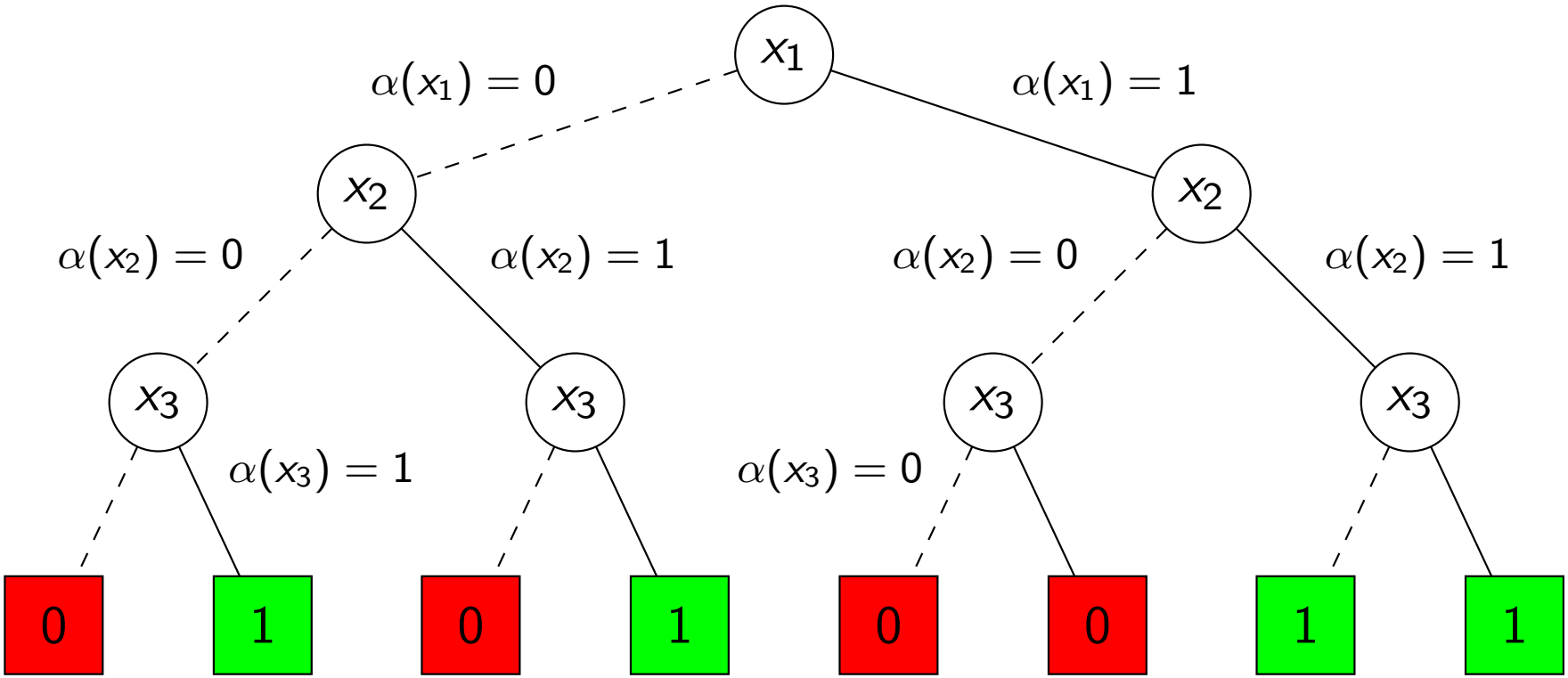
Let φ be a propositional formula with $Var(\varphi) = \{x_1, x_2, \ldots, x_n\}$. Assume fixed ordering of variables: $x_1, x_2, \ldots, x_n$.
Binary decision tree for φ w.r.t. given ordering of variables is a complete ordered binary tree with $n + 1$ levels where

- Nodes at level i are assigned variable x_i.
- Leaves are assigned either 0 or 1 (False or True).
- An edge connecting node x_i and its left successor correspond to an assignment $\alpha(x_i) = 0$.
- An edge connecting node x_i and its right successor correspond to an assignment $\alpha(x_i) = 1$.
- A path from the root to a leave determines a complete valuation α of $Var(\varphi)$ and the leave is assigned truth value $\alpha(\varphi)$.

Binary decision tree

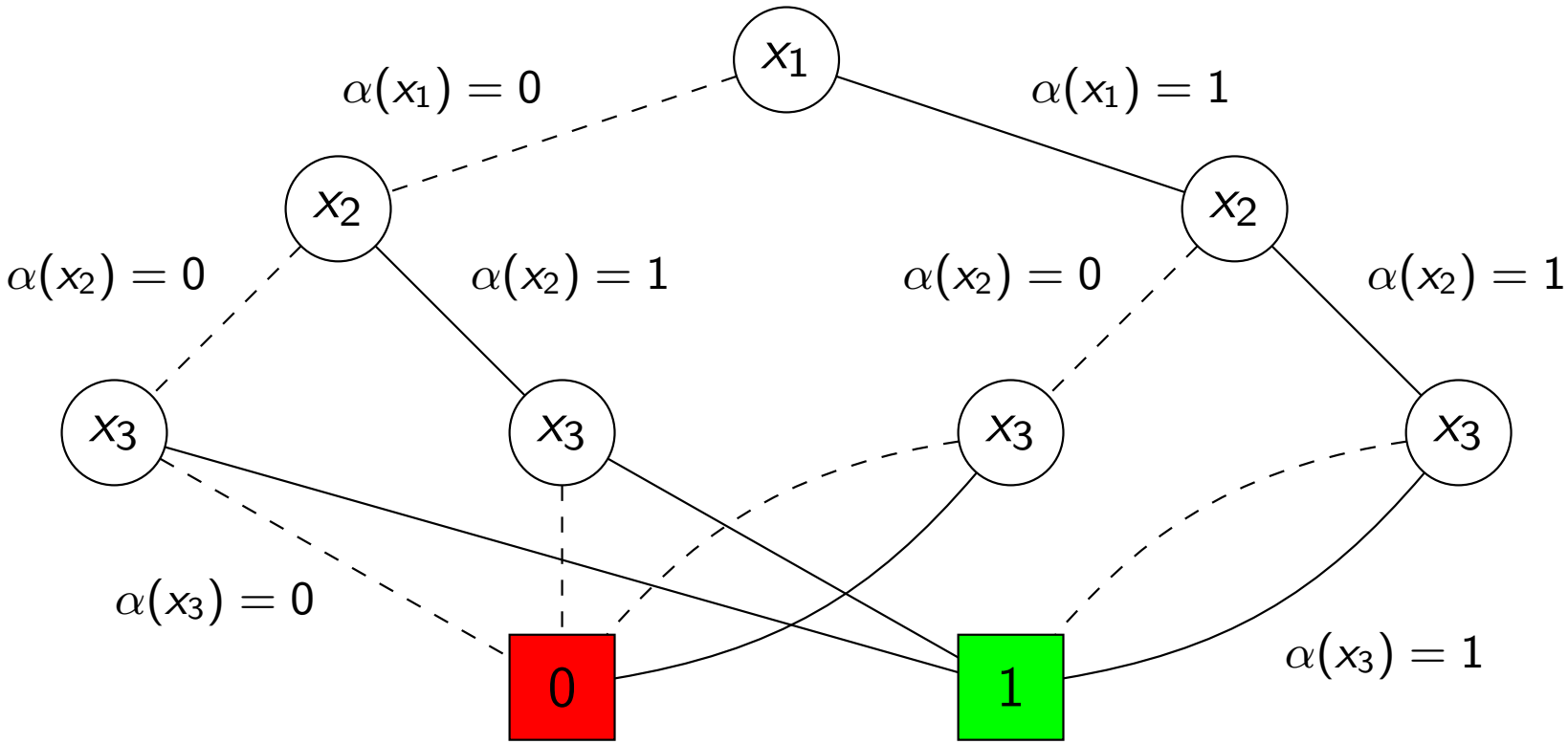
Example

- Consider $\varphi = (x_1 \wedge x_2) \vee (\neg x_1 \wedge x_3)$
  - with ordering of the variables x_1, x_2, x_3



Reductions of BDT

- Reduction (i): Contract leaves to two unique leaves: 0 and 1.



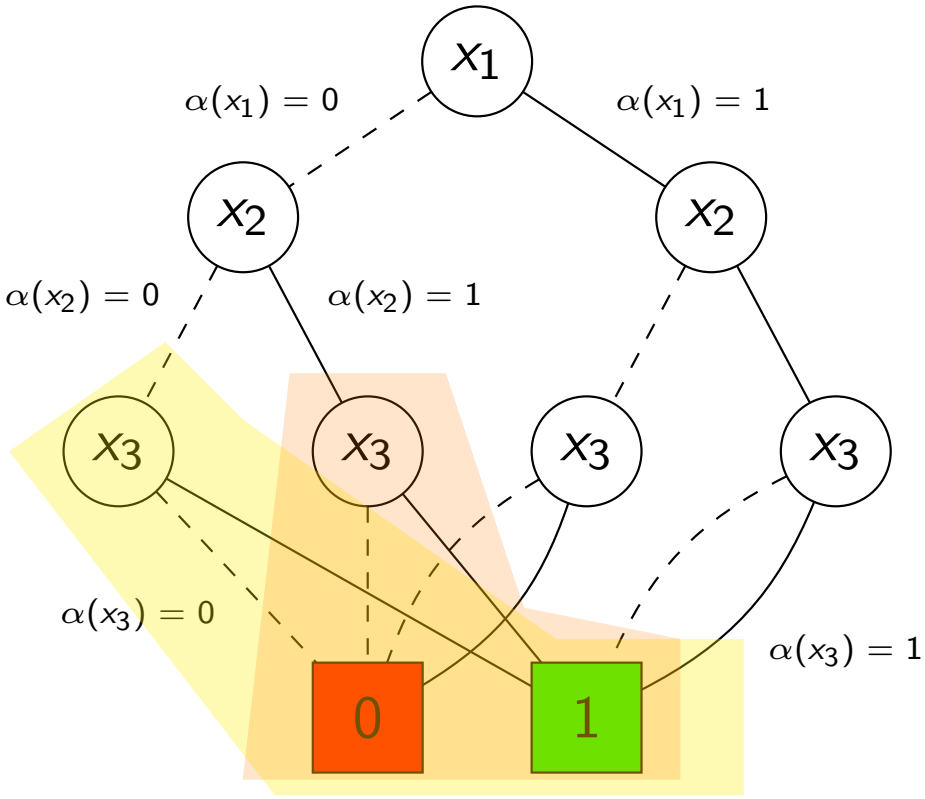
Reductions of BDT

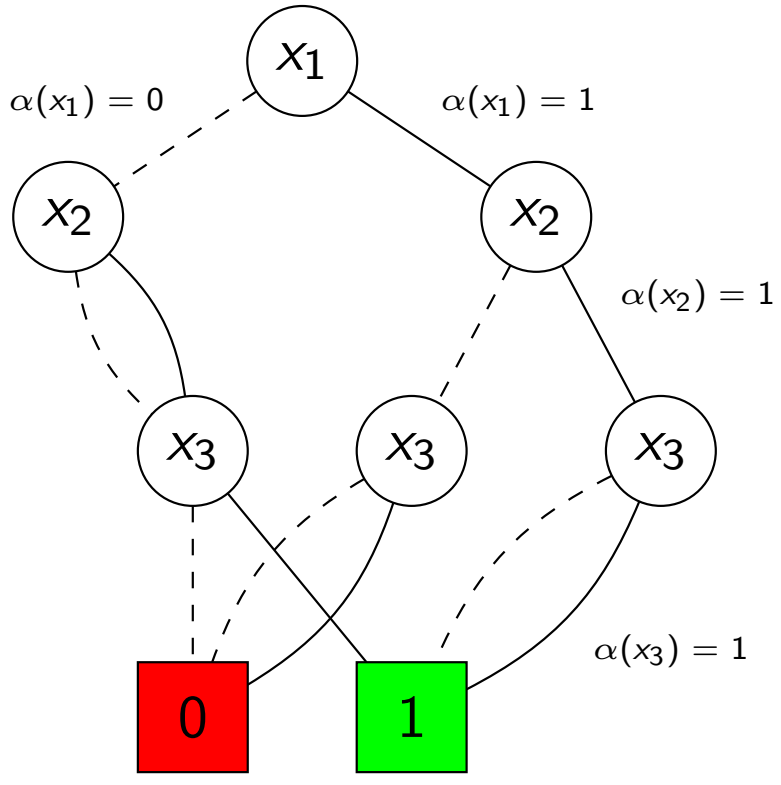
- Reduction (ii): Merge isomorphic subtrees.

Definition (isomorphic subtrees)

Leaves representing the same value are isomorphic subtrees.
Subtrees whose roots represent the same variable and their left and right subtrees respectively are isomorphic, are isomorphic as well.

Reductions of BDT





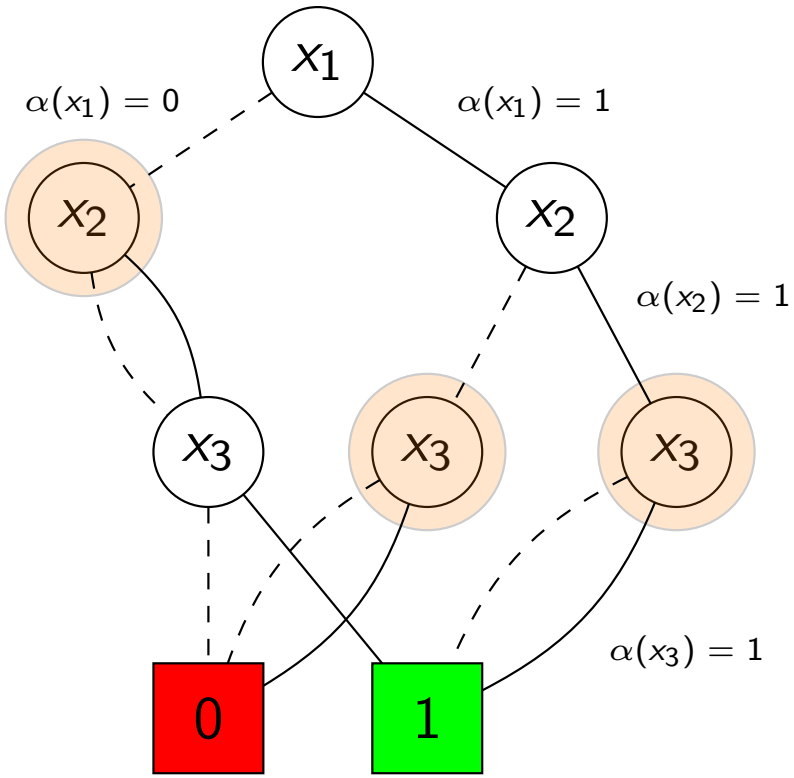
Reductions of BDT

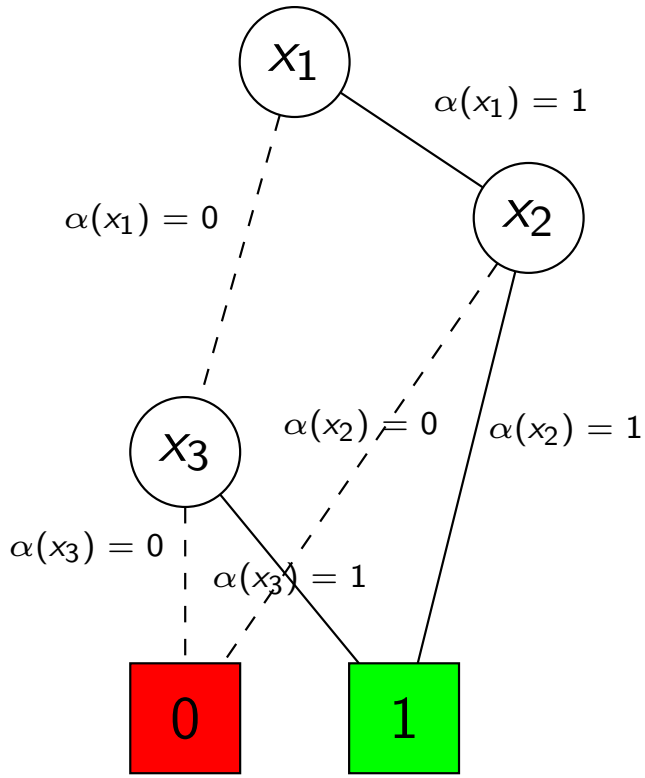
- Reduction (iii): Removing redundant nodes

Definition (redundant node)

Node v in a binary decision tree is redundant if all the valuations determined by paths going through v give the same truth value after changing valuation of v.

Reductions of BDT





Reductions of BDT

- Reductions (ii) and (iii) are repeated as long as they can be applied.
- Result is a *binary decision diagram - BDD*
  - Also called (reduced) ordered binary decision diagram - (R)OBDD.

Lemma

Formulas φ and ψ representing the same Boolean function have the same BDDs for the same ordering of variables.

- BDD is a compact structure for representing all the satisfying and falsifying valuations of a formula.

Inductive construction of BDD

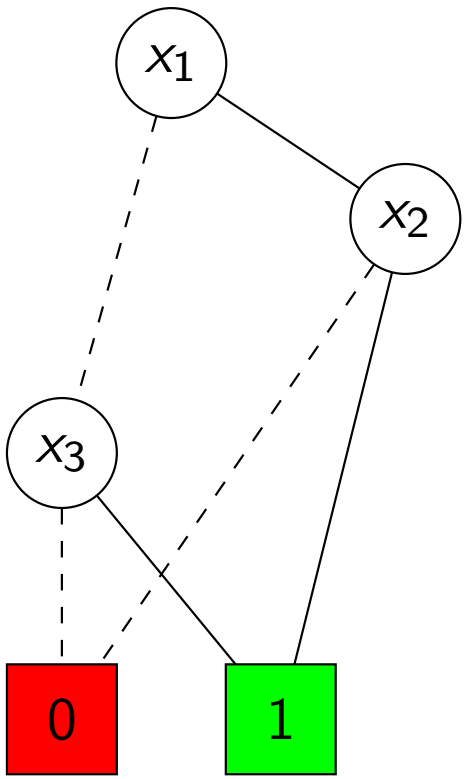
- Inductive according to the structure of the formula
  - Construct BDD for the formula from BDDs of its subformulas.
  - More efficient
- Let B^{φ} be a Boolean function determined by φ and let B^{ψ} be a Boolean function determined by ψ. Let BDD B^{φ} and BDD B^{ψ} be corresponding BDDs.

Definition (Boolean function restriction)

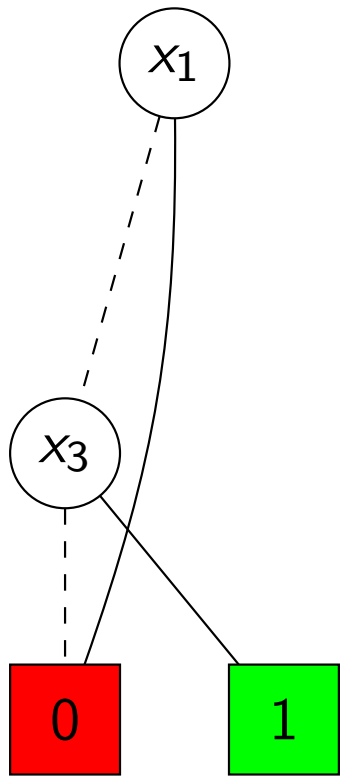
$B^{\varphi}|_{x=0}$ and $B^{\varphi}|_{x=1}$ respectively denotes Boolean function B^{φ} after assigning 0 and 1 respectively to variable x. $B^{\varphi}|_{x=0}$ and $B^{\varphi}|_{x=1}$ are restrictions of B^{φ} w.r.t. variable x.

Construction of restriction in BDD

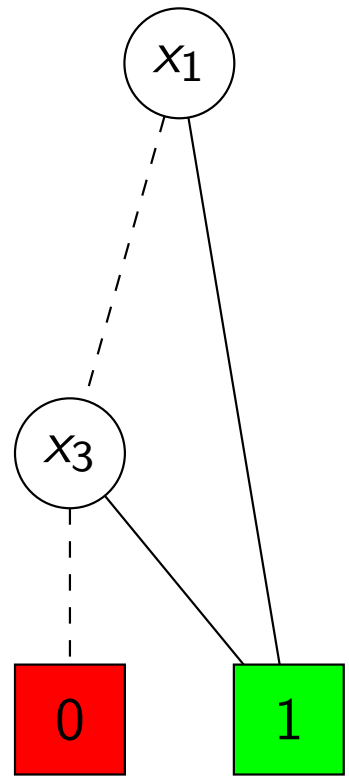
B^{φ}



$B^{\varphi}|_{x_2=0}$



$B^{\varphi}|_{x_2=1}$



Inductive construction of BDD

Let u be a root of BDD B^{φ} and v be a root of BDD B^{ψ}. Let $\otimes$ stand for a binary connective. Then BDD for $\varphi \otimes \psi$ is defined recursively as follows:

1. If u and v are both leaves then BDD $B^{\varphi} \otimes B^{\psi}$ is a leaf with value $val(u) \otimes val(v)$
2. If $var(u) = var(v) = x$, do *Shannon expansion*:
   $B^{\varphi} \otimes B^{\psi} = (\neg x \wedge (B^{\varphi}|_{x=0} \otimes B^{\psi}|_{x=0})) \vee (x \wedge (B^{\varphi}|_{x=1} \otimes B^{\psi}|_{x=1}))$
3. If $var(u) \neq var(v)$ (and w.l.o.g. $var(u) = x$ precedes $var(v)$) then do a modified Shannon expansion:
   $B^{\varphi} \otimes B^{\psi} = (\neg x \wedge (B^{\varphi}|_{x=0} \otimes B^{\psi})) \vee (x \wedge (B^{\varphi}|_{x=1} \otimes B^{\psi}))$

Shannon expansion

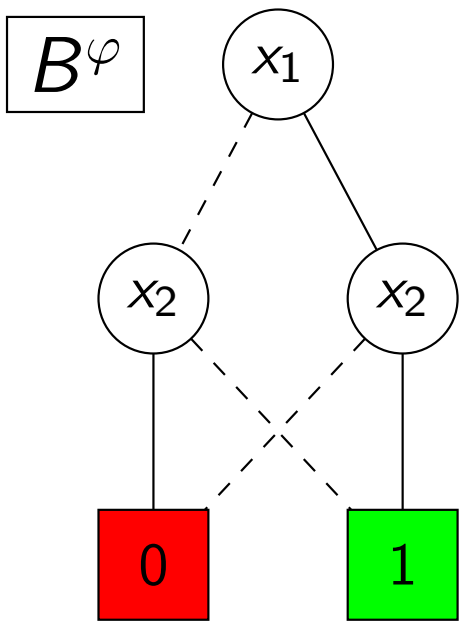
Notes

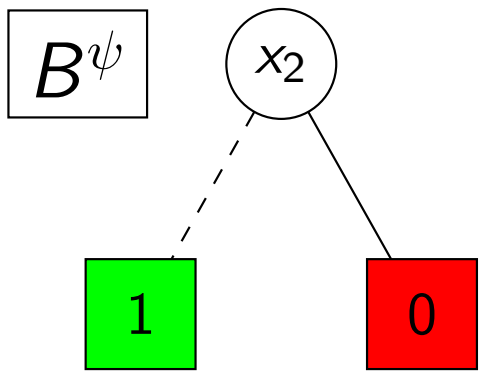
- BDD $B^{\varphi} \otimes B^{\psi}$ after Shannon expansion has new root w with $var(w) = x = var(u) = var(v)$ with left (negative) child BDD $B^{\varphi}|_{x=0} \otimes B^{\psi}|_{x=0}$ and with right (positive) child BDD $B^{\varphi}|_{x=1} \otimes B^{\psi}|_{x=1}$
- BDD $B^{\varphi} \otimes B^{\psi}$ after modified Shannon expansion has new root w with $var(w) = x = var(u)$ with left (negative) child BDD $B^{\varphi}|_{x=0} \otimes B^{\psi}$ and with right (positive) child BDD $B^{\varphi}|_{x=1} \otimes B^{\psi}$
  - Note that we assumed that $var(u)$ precedes $var(v)$ in the variable ordering, thus $var(u)$ is not present in BDD B^{ψ}.

Inductive construction of BDD

Example

- Assume BDD B^{φ} for $\varphi = (\neg x_1 \vee x_2) \wedge (x_1 \vee \neg x_2)$ and BDD B^{ψ} for $\psi = \neg x_2$.
- Construct BDD for $\varphi \vee \psi$ with variable ordering x_1, x_2.

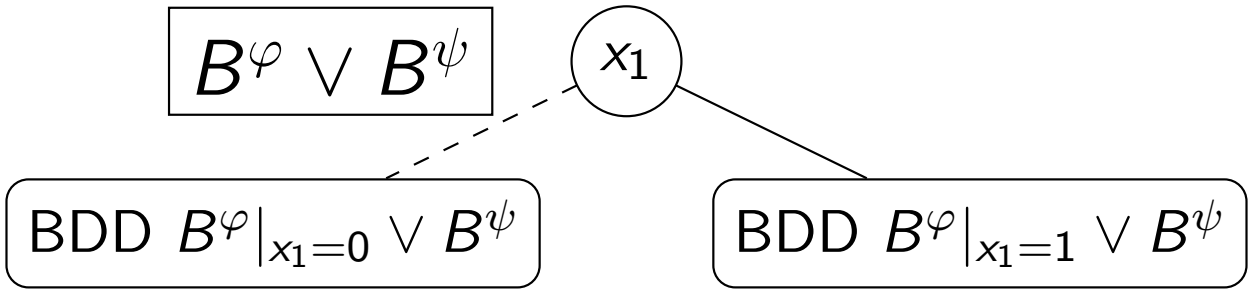




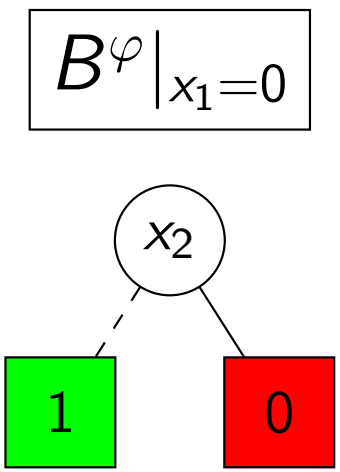
Inductive construction of BDD

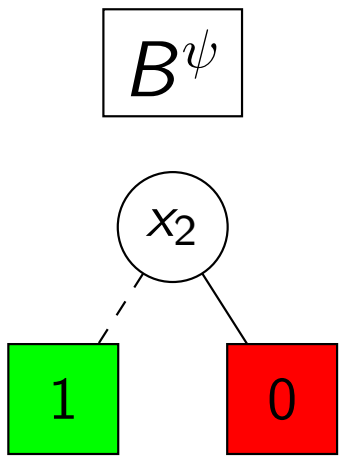
example

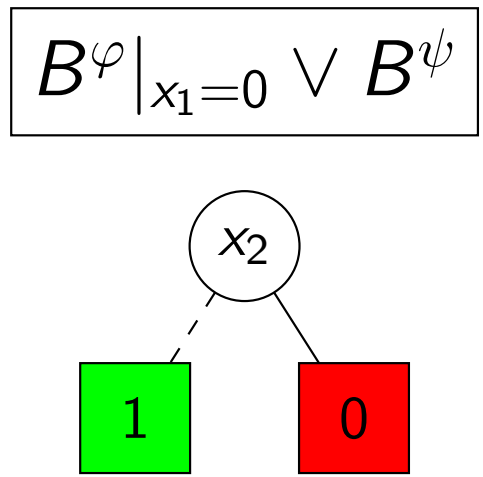
- $B^{\varphi} \vee B^{\psi} = (\neg x_1 \wedge (B^{\varphi}|_{x_1=0} \vee B^{\psi}) \vee (x_1 \wedge (B^{\varphi}|_{x_1=1} \vee B^{\psi})))$



- BDD $B^{\varphi}|_{x_1=0} \vee B^{\psi}$





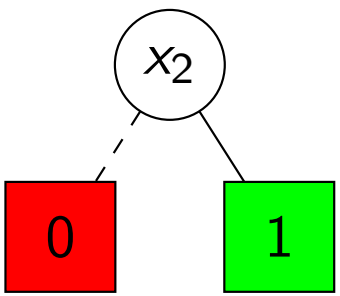


Inductive construction of BDD

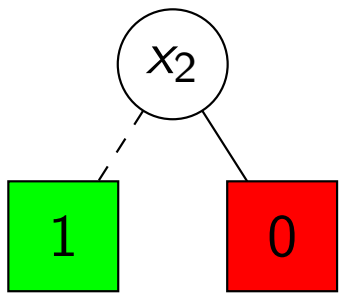
example

- BDD $B^{\varphi}|_{x_1=1} \vee B^{\psi}$

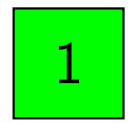
$B^{\varphi}|_{x_1=1}$



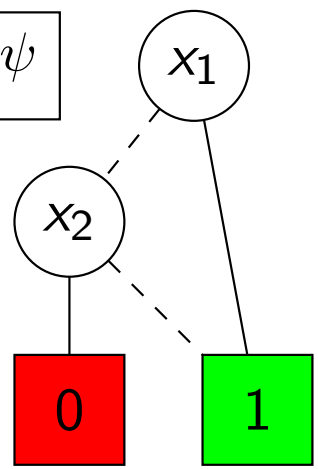
B^{ψ}



$B^{\varphi}|_{x_1=1} \vee B^{\psi}$



- Resulting BDD: $B^{\varphi} \vee B^{\psi}$



BDDs summary

- $|\mathrm{BDD}B^{\varphi} \otimes B^{\psi}| = O(|\mathrm{BDD}B^{\varphi}| \times |\mathrm{BDD}B^{\psi}|)$
- Size of BDDs can be exponential w.r.t. the number of variables
  - Depends on the variable ordering
  - $f(x_1, \ldots, x_{2n}) = x_1x_2 + \cdots + x_{2n-1}x_{2n}$
    - 2^{n+1} nodes for $x_1 < x_3 < \ldots x_{2n-1} < x_2 < x_4 < \cdots < x_{2n}$
    - $2n + 2$ nodes for $x_1 < x_2 < x_3 < \cdots < x_{2n-1} < x_{2n}$
  - Finding the best variable ordering is NP-hard
  - Good heuristics exist
- Applications: circuit synthesis, formal verification, ...

Quantified Boolean formulas (QBFs)

QBF - Introduction

- Language: quantified propositional logic
  - Quantifiers $(\forall, \exists)$ over propositional variables are allowed in the formula.
  - Example: $\forall x_1 \exists x_2 (x_1 \leftrightarrow x_2)$
  - Synatic sugar:
    - $\forall x \varphi(x) \Leftrightarrow (\varphi(0) \wedge \varphi(1))$
    - $\exists x \varphi(x) \Leftrightarrow (\varphi(0) \vee \varphi(1))$

QBF - Introduction

- Language: quantified propositional logic
  - Quantifiers ($\forall, \exists$) over propositional variables are allowed in the formula.
  - Example: $\forall x_1 \exists x_2 (x_1 \leftrightarrow x_2)$
  - Synatic sugar:
    - $\forall x \varphi(x) \Leftrightarrow (\varphi(0) \wedge \varphi(1))$
    - $\exists x \varphi(x) \Leftrightarrow (\varphi(0) \vee \varphi(1))$
- Potentially more succinct encodings than propositional logic

QBF - Introduction

- Language: quantified propositional logic
  - Quantifiers ($\forall, \exists$) over propositional variables are allowed in the formula.
  - Example: $\forall x_1 \exists x_2 (x_1 \leftrightarrow x_2)$
  - Synatic sugar:
    - $\forall x \varphi(x) \Leftrightarrow (\varphi(0) \wedge \varphi(1))$
    - $\exists x \varphi(x) \Leftrightarrow (\varphi(0) \vee \varphi(1))$
- Potentially more succinct encodings than propositional logic
- Satisfiability of QBF is a (canonical) PSPACE-complete problem

Prenex form

- Prenex CNF: $\varphi = \hat{Q}\psi$ where
  - quantifier prefix $\hat{Q} = Q_1 B_1 \dots Q_n B_n$, $Q_i \in \{\forall, \exists\}$, $Q_i \neq Q_{i+1}$, $B_i \subseteq Var(\psi)$, $B_i \cap B_j = \emptyset$ for $i \neq j$.
  - ψ is a propositional formula in CNF
- Linear ordering of variables: $x_i < x_j$ iff $x_i \in B_i$, $x_j \in B_j$ for $i < j$.
- All (and only those) variables of ψ are quantified.
  - Then ϕ is equivalent either to $\top$ or $\bot$.

Semantics

Classical

- The QBF $\bot$ is unsatisfiable, the QBF $\top$ is satisfiable.
- The QBF $\neg\psi$ is satisfiable iff ψ is unsatisfiable.
- The QBF $\psi_1 \wedge \psi_2$ is satisfiable iff ψ_1 and ψ_2 are satisfiable.
- The QBF $\psi_1 \vee \psi_2$ is satisfiable iff ψ_1 or ψ_2 is satisfiable.
- The QBF $\forall x\psi$ is satisfiable iff $\psi[\neg x]$ and $\psi[x]$ are satisfiable.
  The QBF $\psi[\neg x](\psi[x])$ results from ψ by replacing x in ψ by $\bot(\top)$.
- The QBF $\exists x\psi$ is satisfiable iff $\psi[\neg x]$ or $\psi[x]$ is satisfiable.

Semantics

Games

- A game between two players: universal ($P_\forall$) and existential ($P_\exists$).
- $P_\exists$ assigns existential variables and wants to satisfy the formula.
- $P_\forall$ assigns universal variables and wants to falsify the formula.
- Players pick variables from left, according to quantifier ordering.
- QBF is satisfiable (unsatisfiable) iff $P_\exists$ ($P_\forall$) has a winning strategy.
  - Winning strategy: $P_\exists$ ($P_\forall$) can satisfy (falsify) the formula regardless of opponent's choices.

DPLL algorithm for QBF

1: **procedure** QDPLL(PCNF φ, assignment α)
2: $\psi \leftarrow$ simplify($\varphi[\alpha]$)
3: **if** $\psi == \top$ **then return** TRUE
4: **if** $\psi == \bot$ **then return** FALSE
5: **if** first quantifier is existential **then**
return QDPLL($\psi, \alpha \cup \{\neg x\}$) **OR** QDPLL($\psi, \alpha \cup \{x\}$)
6: **if** first quantifier is universal **then**
return QDPLL($\psi, \alpha \cup \{\neg x\}$) **AND** QDPLL($\psi, \alpha \cup \{x\}$)

Improving QDPLL

- Boolean Constraint Propagation for QBF
- Non-chronological backtracking
- Learning clauses (from conflicts)
- Learning cubes (from satisfying assignments)

Boolean Constraint Propagation for QBF

1. Universal reduction
   - For a clause C,
     $UR(C) = C \setminus \{l \in C \mid q(l) = \forall \wedge \forall l' \in C\ (q(l') = \exists \rightarrow var(l') < var(l))\}$
2. Unit literal propagation
   - Unit literal is assigned to TRUE
   - If a clause C constains a single literal l and $q(l) = \exists$, l is a unit literal.
3. Pure literal propagation
   - Pure literal l is assigned to TRUE if $q(l) = \exists$ and to FALSE if $q(l) = \forall$.

- Applied iteratively until nothing changes or the formula has been solved.

QBF summary

- Generalization of SAT
- More compact encoding
- Younger field $\rightarrow$ research opportunities
  - New techniques being explored
- QBF research community: http://www.qbflib.org