University of Antwerp

Specification and Verification

Lecture 5: Binary decision diagrams

Guillermo A. Pérez

October 27, 2024

BDDs in short

What are they?

A graph-based data structure for Boolean functions and sets

BDDs in short

What are they?

A graph-based data structure for Boolean functions and sets

Why use them?

They provide a canonical, sometimes space-efficient, representation of Boolean functions; allow to implement logical operations $\vee, \wedge, \neg$ using efficient graph transformations.

References

Main references

- **Binary Decision Diagrams**. Randal E. Bryant. 2018. In Handbook of Model Checking. Springer, 2018.
- **The Art of Computer Programming, Vol. 4, Pre-Fascicle 1B: Binary Decision Diagrams**. Don Knuth. Addison-Wesley, 2008.

Required and target competences

What tools do we need?

Discrete mathematics, data abstraction and structures; algorithms and complexity

Required and target competences

What tools do we need?

Discrete mathematics, data abstraction and structures; algorithms and complexity

What skills will we obtain?

- theory: connect functions, logic, and BDDs
- practice: manipulate BDD-represented functions & sets

Required and target competences

What tools do we need?

Discrete mathematics, data abstraction and structures; algorithms and complexity

What skills will we obtain?

- theory: connect functions, logic, and BDDs
- practice: manipulate BDD-represented functions & sets

Why?

May be useful for professional work on circuits/hardware and

- data structures and graph algos
- artificial intelligence (knowledge representation)
- specification and verification

Outline

1. Recap on Boolean functions and sets
2. Example: from characteristic functions to BDDs
3. History
4. Motivation and applications
5. Reduced ordered BDDs
6. BDD operations
7. Shared table and other implementation tricks
8. Conclusions

Boolean functions

Definition (Boolean or switching functions)

A Boolean function f is of the form $\mathbb{B}^k \to \mathbb{B}$, where $\mathbb{B} = \{0, 1\}$ and $k \in \mathbb{N}$ is the arity of f.

Boolean functions

Definition (Boolean or switching functions)

A Boolean function f is of the form $\mathbb{B}^k \to \mathbb{B}$, where $\mathbb{B} = \{0, 1\}$ and $k \in \mathbb{N}$ is the arity of f.

Example

Let us define the function $g : \mathbb{B}^2 \to \mathbb{B}$ such that $g(0, 0) = 0$, $g(0, 1) = 1$, $g(1, 0) = 1$, and $g(1, 1) = 0$. Do you recognize the function?

Boolean functions

Definition (Boolean or switching functions)

A Boolean function f is of the form $\mathbb{B}^k \to \mathbb{B}$, where $\mathbb{B} = \{0, 1\}$ and $k \in \mathbb{N}$ is the arity of f.

Example

Let us define the function $g : \mathbb{B}^2 \to \mathbb{B}$ such that $g(0, 0) = 0$, $g(0, 1) = 1$, $g(1, 0) = 1$, and $g(1, 1) = 0$. Do you recognize the function?

x_1	x_2	XOR
0	0	0
0	1	1
1	0	1
1	1	0

Defining Boolean functions

How can we define a Boolean function?

- They are expressively equivalent to
  - truth tables
  - propositional formulas
- They can encode sets via their characteristic function
- They can be represented as
  - And-inverter graphs
  - Negation normal forms
  - Propositional directed acyclic graphs
  - Binary decision diagrams

Propositional formulas & Bool functions

Propositional formulas

For a set $P = \{p_1, \ldots, p_k\}$ of propositions, a propositional formula φ over P is constructed using the logical connectives $\vee, \wedge, \neg$. E.g. $(p_1 \wedge \neg p_2) \vee (\neg p_1 \wedge p_2)$.

Propositional formulas & Bool functions

Propositional formulas

For a set $P = \{p_1, \ldots, p_k\}$ of propositions, a propositional formula φ over P is constructed using the logical connectives $\vee, \wedge, \neg$. E.g. $(p_1 \wedge \neg p_2) \vee (\neg p_1 \wedge p_2)$.

- φ induces a Boolean function f_φ that maps $(x_1, \ldots, x_k)$ to the truth value of φ under the truth-value assignment $p_1 = x_1, \ldots, p_k = x_k$.

Propositional formulas & Bool functions

Propositional formulas

For a set $P = \{p_1, \dots, p_k\}$ of propositions, a propositional formula φ over P is constructed using the logical connectives $\vee, \wedge, \neg$. E.g. $(p_1 \wedge \neg p_2) \vee (\neg p_1 \wedge p_2)$.

- φ induces a Boolean function f_φ that maps $(x_1, \dots, x_k)$ to the truth value of φ under the truth-value assignment $p_1 = x_1, \dots, p_k = x_k$.

Completeness

Every Boolean function g has a corresponding propositional formula φ, i.e. $g = f_\varphi$ for some φ. (Graphical proof via truth tables, see board).

Subsets and Boolean functions

Characteristic functions of subsets

Let $S = \{s_1, \ldots, s_k\}$ be a finite set of elements. Every subset $P \subseteq S$ induces a function $\chi_P : S \to \mathbb{B}$

$$\chi_P(s) = \begin{cases} 1 & \text{if } s \in P \\ 0 & \text{otherwise.} \end{cases}$$

Subsets and Boolean functions

Characteristic functions of subsets

Let $S = \{s_1, \ldots, s_k\}$ be a finite set of elements. Every subset $P \subseteq S$ induces a function $\chi_P : S \to \mathbb{B}$

$$\chi_P(s) = \begin{cases} 1 & \text{if } s \in P \\ 0 & \text{otherwise.} \end{cases}$$

Binary encoding of the set

Let $\ell = \lceil \log_2 k \rceil$ and consider a mapping $\beta : \mathbb{B}^\ell \to S \cup \{\bot\}$ such that $\beta|_{\beta^{-1}(S)}$ is injective and surjective. Every subset $P \subseteq S$ induces a function $\chi_P^\beta : \mathbb{B}^\ell \to \mathbb{B}$

$$\chi_P^\beta(x_1, \ldots, x_\ell) = \begin{cases} 1 & \text{if } \beta(x_1, \ldots, x_\ell) \in P \\ 0 & \text{otherwise.} \end{cases}$$

Example: binary encoding of a set

1. $S = \{\text{Belgium, Panama, Tunisia, England, Honduras}\}$

Example: binary encoding of a set

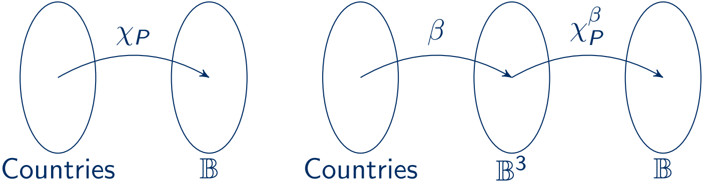
1. $S = \{\text{Belgium, Panama, Tunisia, England, Honduras}\}$
2. $P = \{\text{Belgium, Panama, Tunisia, England}\}$

$$\chi_P(\text{Belgium}) = 1 \qquad \chi_P(\text{Panama}) = 1$$
$$\chi_P(\text{Tunisia}) = 1 \qquad \chi_P(\text{England}) = 1$$

Example: binary encoding of a set

1. $S = \{\text{Belgium, Panama, Tunisia, England, Honduras}\}$
2. $P = \{\text{Belgium, Panama, Tunisia, England}\}$

$$\chi_P(\text{Belgium}) = 1 \qquad \chi_P(\text{Panama}) = 1$$
$$\chi_P(\text{Tunisia}) = 1 \qquad \chi_P(\text{England}) = 1$$



Example: binary encoding of a set

1. $S = \{\text{Belgium, Panama, Tunisia, England, Honduras}\}$
2. $P = \{\text{Belgium, Panama, Tunisia, England}\}$
3. We choose some β, for example:

x_1	x_2	x_3	$s \in S \cup \{\perp\}$
0	0	0	Belgium
0	0	1	Panama
0	1	0	$\perp$
0	1	1	Tunisia
1	0	0	$\perp$
1	0	1	Honduras
1	1	0	$\perp$
1	1	1	England

Example: binary encoding of a set

1. We choose some β, for example:

x_1	x_2	x_3	$s \in S \cup \{\perp\}$
0	0	0	Belgium
0	0	1	Panama
0	1	0	$\perp$
0	1	1	Tunisia
1	0	0	$\perp$
1	0	1	Honduras
1	1	0	$\perp$
1	1	1	England

Example: binary encoding of a set

1. We choose some β, for example:

x_1	x_2	x_3	$s \in S \cup \{\perp\}$
0	0	0	Belgium
0	0	1	Panama
0	1	0	$\perp$
0	1	1	Tunisia
1	0	0	$\perp$
1	0	1	Honduras
1	1	0	$\perp$
1	1	1	England

2. $P = \{\text{Belgium, Panama, Tunisia, England}\}$

$$\chi_P^\beta(0,0,0) = 1 \qquad \chi_P^\beta(0,0,1) = 1$$

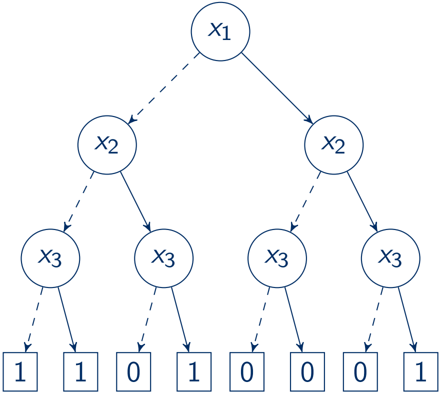
$$\chi_P^\beta(0,1,1) = 1 \qquad \chi_P^\beta(1,1,1) = 1$$

and all other combinations map to 0. Also, $\chi_P^\beta = f_\varphi$ with $\varphi = (\neg x_1 \wedge \neg x_2) \vee (x_2 \wedge x_3)$.

Example: truth tables to dec. trees

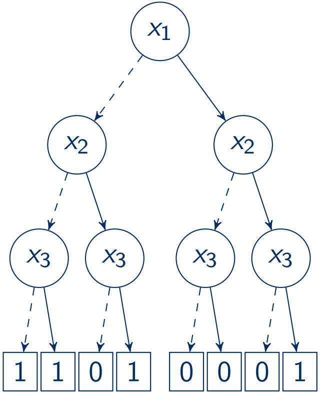
x_1	x_2	x_3	$s \in S \cup \{\bot\}$	χ_P^β
0	0	0	Belgium	1
0	0	1	Panama	1
0	1	0	$\bot$	0
0	1	1	Tunisia	1
1	0	0	$\bot$	0
1	0	1	Honduras	0
1	1	0	$\bot$	0
1	1	1	England	1

- dashed: assignment $x_1 = 0$; solid arrows, $x_1 = 1$
- leaves: value of χ_P^β



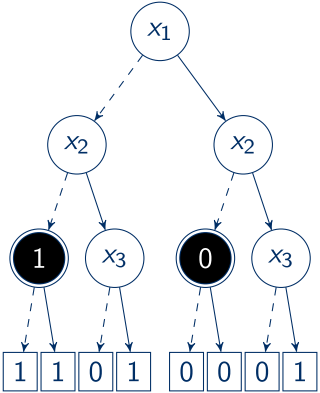
Example: decision trees to ROBDDs

1. Identify useless sub-graphs: those whose children are equivalent



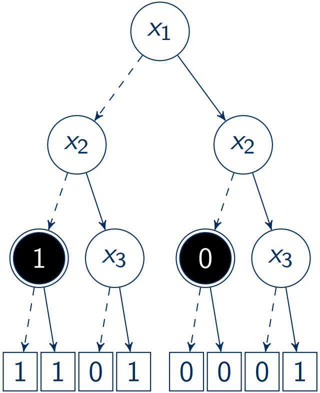
Example: decision trees to ROBDDs

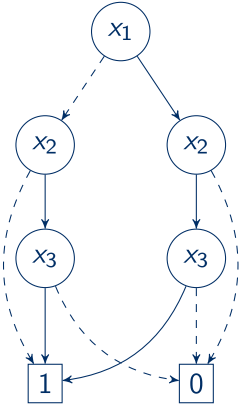
1. Identify useless sub-graphs: those whose children are equivalent
2. Merge equivalent leaves and remove useless sub-trees



Example: decision trees to ROBDDs

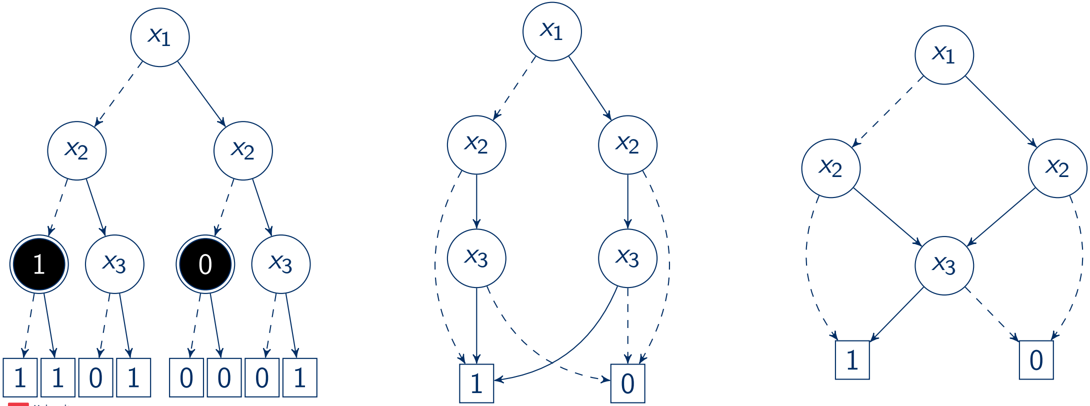
1. Identify useless sub-graphs: those whose children are equivalent
2. Merge equivalent leaves and remove useless sub-trees
3. Merge sub-graph-equivalent inner vertices





Example: decision trees to ROBDDs

1. Identify useless sub-graphs: those whose children are equivalent
2. Merge equivalent leaves and remove useless sub-trees
3. Merge sub-graph-equivalent inner vertices



Exercise: from a subset to its ROBDD

- $S = \{\text{Brussels, Antwerp, Leuven, Amsterdam}\}$
- We choose β as follows

x_1	x_2	$s \in S \cup \{\perp\}$
0	0	Brussels
0	1	Antwerp
1	0	Amsterdam
1	1	Leuven

- $P = \{\text{Brussels, Leuven, Antwerp}\}$
- What is the ROBDD for χ_P^β?

History

From Boole to Lee

- The Shannon expansion (due to Boole) is the basic theoretical idea behind BDDs
$$f(x_1, \ldots, x_k) = \left(\neg x_i \wedge f|_{x_i \leftarrow 0}\right) \vee \left(x_i \wedge f|_{x_i \leftarrow 1}\right)$$
for any x_i.
- **Representation of Switching Circuits by Binary-Decision Programs**. C. Y. Lee. 1959. Bell System Tech. Journal.

History

From Boole to Lee

- The Shannon expansion (due to Boole) is the basic theoretical idea behind BDDs
$$f(x_1, \ldots, x_k) = \left(\neg x_i \wedge f|_{x_i \leftarrow 0}\right) \vee \left(x_i \wedge f|_{x_i \leftarrow 1}\right)$$
for any x_i.
- **Representation of Switching Circuits by Binary-Decision Programs**. C. Y. Lee. 1959. Bell System Tech. Journal.

BDDs become efficient

- Randal E. Bryant introduces sharing (for compression) and order fixing (for canonicity) [1986–1990]
- Don Knuth: "one of the only really fundamental data structures that came out in the last . . . years"

Motivation and applications

Any $f(x_1, \ldots, x_k)$ for which you can build a BDD

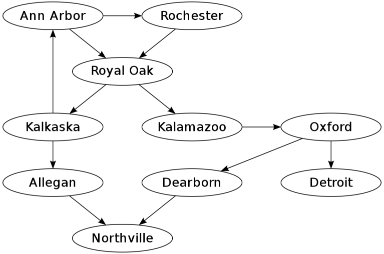
- we can evaluate $f(x)$ in at most k steps,
- we can find the (lexicographically) smallest x such that $f(x) = 1$,
- we can count and/or list the number of solutions to $f(x) = 1$,
- ...

Applications

- Computer-aided design of circuits
- Formal verification
- Bayesian reasoning
- ...

Sample application: the geography game

- Alternate naming cities, no repetition allowed
- The next city's name should start with the same letter as the last letter of the previously named city



- If I start, can I win or make sure the game lasts for at least three rounds (i.e. I play at least 2 cities)?

ROBDDs: the formal definition (1/4)

Reduced ordered BDDs

A (RO)BDD is a rooted directed acyclic graph (DAG) with inner decision vertices and constant-value leaves (with values 0 or 1).

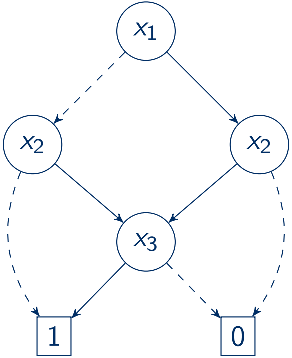
- Every decision vertex v is labelled by a Boolean variable $\texttt{VAR}(v)$ and has two children: $\texttt{LO}(v)$ and $\texttt{HI}(v)$ respectively representing a truth-value assignment of 0 or 1 for v.

ROBDDs: the formal definition (1/4)

ROBDDS, continued

- All paths $v_1 \dots v_\ell$ are such that $\texttt{VAR}(v_i) < \texttt{VAR}(v_j)$ for all $i < j$ (for some ordering of the variables).
- There are at most two constant-value leaves with distinct values.
- No two vertices u, v induce isomorphic graphs ($\texttt{VAR}(u) = \texttt{VAR}(v)$, $\texttt{LO}(u) = \texttt{LO}(v)$, and $\texttt{HI}(u) = \texttt{HI}(v)$) and
- all vertices v have non-isomorphic children, i.e. $\texttt{LO}(v) \neq \texttt{HI}(v)$.

ROBDDs: an example (2/4)



- for the order $x_1 < x_2 < x_3$

ROBDDs: from BDDs to functions (3/4)

Given a BDD, what is its function/formula?

For a BDD with variables $x_1, \ldots, x_k$ and root u, the Boolean function $f_u(x_1, \ldots, x_k)$ induced by it is defined as follows. For all vertices v

- if $\mathtt{VAR}(v) = 0$ then $f_v(x_1, \ldots, x_k) = 0$,
- if $\mathtt{VAR}(v) = 1$ then $f_v(x_1, \ldots, x_k) = 1$,
- otherwise, by the Shannon expansion,

$$f_v(x_1, \ldots, x_k) = \left(\neg\mathtt{VAR}(v) \land f_{\mathtt{LO}(v)}\right) \lor \left(\mathtt{VAR}(v) \land f_{\mathtt{HI}(v)}\right).$$

ROBDDs: from BDDs to functions (3/4)

Given a BDD, what is its function/formula?

For a BDD with variables $x_1, \ldots, x_k$ and root u, the Boolean function $f_u(x_1, \ldots, x_k)$ induced by it is defined as follows. For all vertices v

- if $\mathtt{VAR}(v) = 0$ then $f_v(x_1, \ldots, x_k) = 0$,
- if $\mathtt{VAR}(v) = 1$ then $f_v(x_1, \ldots, x_k) = 1$,
- otherwise, by the Shannon expansion,

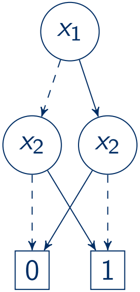
$$f_v(x_1, \ldots, x_k) = \left(\neg\mathtt{VAR}(v) \wedge f_{\mathtt{LO}(v)}\right) \vee \left(\mathtt{VAR}(v) \wedge f_{\mathtt{HI}(v)}\right).$$

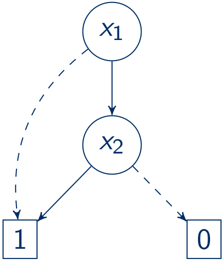
Theorem (Canonicity [Bryant '86])

For all Boolean functions $f(x_1, \ldots, x_k)$, for every ordering of $x_1, \ldots, x_k$, there is a unique BDD with root u such that $f = f_u$.

Exercise: from BDDs to functions

- What are the formulas for the following BDDs?





ROBBDs: on the ordering (4/4)

What order should one use?

Depending on the ordering of $x_1, \ldots, x_k$, the size of the BDD representing $f(x_1, \ldots, x_k)$ may vary exponentially!

- Deciding whether a given order is size-minimal is an NP-complete problem.
- In practice, heuristics are used to dynamically choose a "good" ordering.

ROBDDs: on the ordering (4/4)

Example

x_1	x_2	x_3	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

x_2	x_1	x_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

ROBDDs: on the ordering (4/4)

Example

x_1	x_2	x_3	f
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

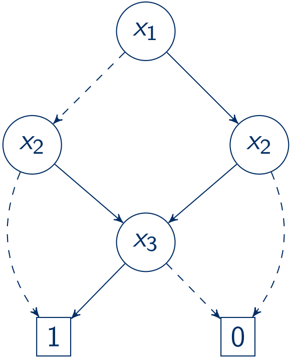
x_2	x_1	x_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

- with large BDDs the effect is more evident

Exercise: a different order

- construct the ROBDD for Belgium's World Cup group using the order $x_2 < x_3 < x_1$ and compare against the one we had before

x_1	x_2	x_3	$s \in S \cup \{\perp\}$
0	0	0	Belgium
0	0	1	Panama
0	1	0	$\perp$
0	1	1	Tunisia
1	0	0	$\perp$
1	0	1	Honduras
1	1	0	$\perp$
1	1	1	England



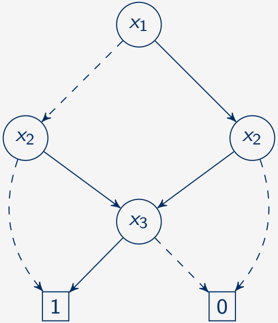
BDD operations: logical negation (1/4)

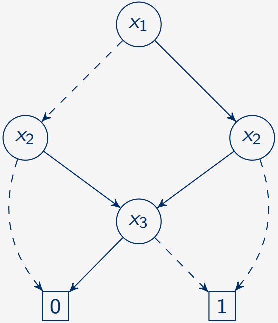
Negating a BDD

Given a BDD representing the Boolean function $f(x_1, \ldots, x_k)$, we can obtain a BDD for $g(x_1, \ldots, x_k) = \neg f(x_1, \ldots, x_k)$ by simply replacing the 0 and 1 constant-value leaves.

BDD operations: logical negation (1/4)

Example





BDD operations: $\vee, \wedge$ (2/4)

Divide and conquer using Shannon's expansion

For $\mathrm{op} \in \{\vee, \wedge\}$ we have that $f(x_1, \ldots, x_k)$ op $g(x_1, \ldots, x_k)$ is equivalent to the following for all $1 \leq i \leq k$

$$\left(\neg x_i \wedge \left(f|_{x_i \leftarrow 0} \ \mathrm{op} \ g|_{x_i \leftarrow 0}\right)\right) \vee \left(x_i \wedge \left(f|_{x_i \leftarrow 1} \ \mathrm{op} \ g|_{x_i \leftarrow 1}\right)\right).$$

BDD operations: $\vee, \wedge$ (2/4)

More graphically...

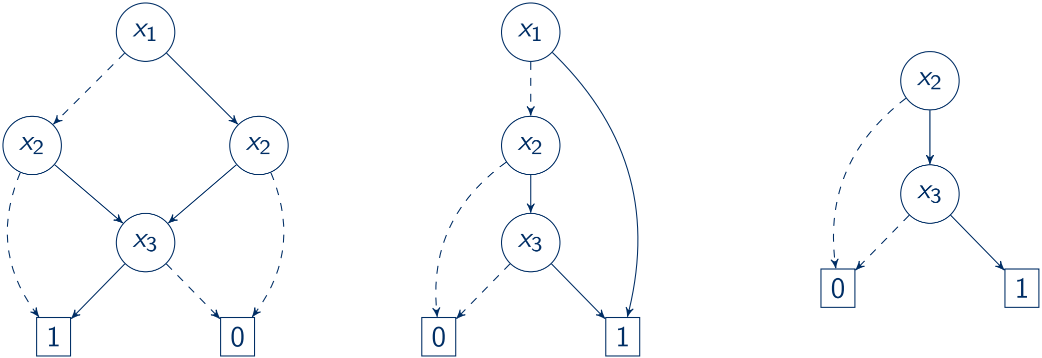
We do a DFS on both BDDs from the root stepping lexicographically and

- synchronously if the current vertices u, v have the same label, i.e. $\mathtt{VAR}(u) = \mathtt{VAR}(v)$,
- asynchronously from v if the current vertices u, v are such that $\mathtt{VAR}(u) > \mathtt{VAR}(v)$.[a]

For every vertex-pair whose children s, t have been visited, we add a new vertex r with $\mathtt{VAR}(r) = \min(\mathtt{VAR}(u), \mathtt{VAR}(v))$ and $\mathtt{LO}(r) = s, \mathtt{HI}(r) = t$ to the resulting BDD.

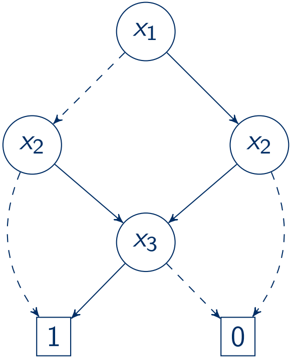
[a] We suppose $\mathtt{VAR}(t) \geq \mathtt{VAR}(v)$ for all constant-value vertices t and all vertices v.

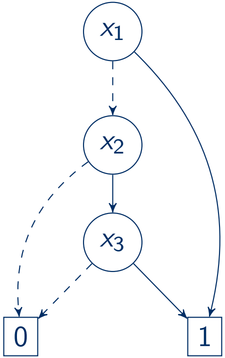
BDD operations: example (3/4)



- conjoining the first two BDDs yields the third (from left to right)

Exercise: disjunction of BDDs





?

- the disjunction of the first two BDDs yields...

BDD operations: others (4/4)

Useful classical operations

- existential quantification can be implemented using the equivalence

$$\exists x_1 : f(x_1, \ldots, x_k) = f|_{x_1 \leftarrow 0} \vee f|_{x_1 \leftarrow 1} ,$$

- universal quantification via the equivalence

$$\forall x_1 : f(x_1, \ldots, x_k) = f|_{x_1 \leftarrow 0} \wedge f|_{x_1 \leftarrow 1} ,$$

- composition $f|_{x_i \leftarrow g}$ for f, g Boolean functions,
- (partial) evaluation $f|_{x_i, \ldots, x_j \leftarrow b_i, \ldots, b_j}$
- ...

University of Antwerp

Shared table and other tricks

Implementing the data structure

Most implementations keep a unique table:

- every entry is of the form $\langle \texttt{id}, v, \ell, h, \rangle$ and every "function" is a reference to an entry of the table.
- Some implementations also include flags to indicate whether ℓ or h are negated (allows further sharing).

Shared table and other tricks

Implementing the data structure

Most implementations keep a unique table:

- every entry is of the form $\langle \texttt{id}, v, \ell, h, \rangle$ and every "function" is a reference to an entry of the table.
- Some implementations also include flags to indicate whether ℓ or h are negated (allows further sharing).

Algorithms for the data structure

- Most operations share some code that is usually refactored into the `ITE` (if-then-else) method.
- After every operation, the unique table is scanned to compute the size of referenced BDDs.
- Based on this, a dynamic-reordering algorithm may be called to attempt to reduce that size.

Conclusions

Theory

- Boolean functions can be represented by truth tables, propositional logic formulas, and BDDs.
- Reduced ordered BDDs are a canonical representation of Boolean functions for any fixed variable order.

Practice

- BDDs can easily be manipulated using the usual logical connectives to obtain BDDs for more complex formulas.
- Under the hood, these operations are implemented using graph algorithms.
- The latter allows for many useful operations to be applicable to an already constructed BDD.