



## Logic

Diana Gründlinger

Aart Middeldorp

Fabian Mitterwallner

Alexander Montag

Johannes Niederhauser

Daniel Rainer

 **articify**  
ars.uibk.ac.at

with session ID **0992 9580** for anonymous questions



# Outline

- 1. Summary of Previous Lecture**
- 2. Algorithms for Binary Decision Diagrams**
- 3. Intermezzo**
- 4. Hidden Weighted Bit Function**
- 5. Predicate Logic**
- 6. Further Reading**

## Theorem

natural deduction is **complete**:  $\varphi_1, \varphi_2, \dots, \varphi_n \models \psi \implies \varphi_1, \varphi_2, \dots, \varphi_n \vdash \psi$  is valid

## Definitions

- ▶ **clause** is set of literals  $\{l_1, \dots, l_n\}$
- ▶  $\square$  denotes **empty clause**
- ▶ **clausal form** is set of clauses  $\{C_1, \dots, C_m\}$
- ▶ literals  $l_1$  and  $l_2$  are **complementary** if  $l_1 = l_2^c = \begin{cases} \neg p & \text{if } l_2 = p \\ p & \text{if } l_2 = \neg p \end{cases}$
- ▶ clauses  $C_1$  and  $C_2$  **clash** on literal  $l$  if  $l \in C_1$  and  $l^c \in C_2$
- ▶ **resolvent** of clashing clauses  $C_1$  and  $C_2$  on literal  $l$  is clause  $(C_1 \setminus \{l\}) \cup (C_2 \setminus \{l^c\})$

## Resolution

input: clausal form  $S$

output: yes if  $S$  is satisfiable    no if  $S$  is unsatisfiable

- ① repeatedly add resolvent of clashing clauses in  $S$
- ② return **no** as soon as empty clause is derived
- ③ return **yes** if all clashing clauses have been resolved

## Definition

**refutation** of  $S$  is resolution derivation of  $\square$  from  $S$

## Theorem

- ▶ resolution is **terminating**
- ▶ resolution is **sound** and **complete**:  $S$  admits refutation  $\iff$  clausal form  $S$  is unsatisfiable

## Remark

binary decision diagram (BDD) is directed acyclic graph (dag) representing boolean function

## Definitions

- ▶ BDD is **reduced** if **C1**, **C2**, **C3** are not applicable
  - C1** remove duplicate terminals
  - C2** remove redundant tests
  - C3** remove duplicate non-terminals
- ▶ BDD  $B$  is **ordered** if there exists order  $[x_1, \dots, x_n]$  of variables in  $B$  such that



- ▶ orders  $o_1$  and  $o_2$  are **compatible** if  $o_1$  and  $o_2$  are subsequences of some order  $o$

## Theorem

reduced OBDD representation of boolean function for given order is **unique**

## Corollary

checking

- ▶ satisfiability
- ▶ validity
- ▶ equivalence

is **trivial** for reduced OBDDs (with compatible variable orderings)

## Part I: Propositional Logic

algebraic normal forms, binary decision diagrams, conjunctive normal forms, DPLL, Horn formulas, natural deduction, Post's adequacy theorem, resolution, SAT, semantics, sorting networks, soundness and completeness, syntax, Tseitin's transformation

## Part II: Predicate Logic

natural deduction, quantifier equivalences, resolution, semantics, Skolemization, syntax, undecidability, unification

## Part III: Model Checking

adequacy, branching-time temporal logic, CTL\*, fairness, linear-time temporal logic, model checking algorithms, symbolic model checking



## Part I: Propositional Logic

algebraic normal forms, **binary decision diagrams**, conjunctive normal forms, DPLL, Horn formulas, natural deduction, Post's adequacy theorem, resolution, SAT, semantics, sorting networks, soundness and completeness, syntax, Tseitin's transformation

## Part II: Predicate Logic

natural deduction, quantifier equivalences, resolution, semantics, Skolemization, **syntax**, undecidability, unification

## Part III: Model Checking

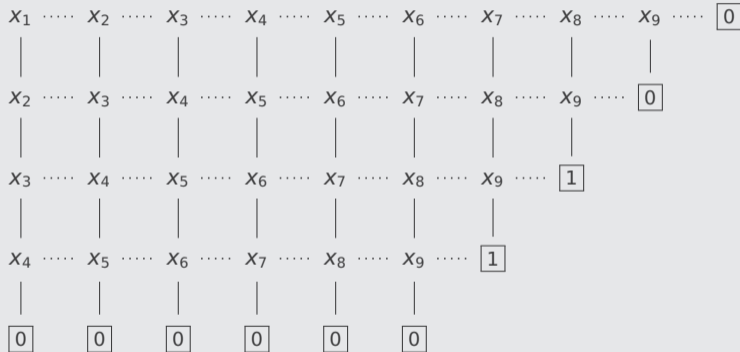
adequacy, branching-time temporal logic, CTL\*, fairness, linear-time temporal logic, model checking algorithms, symbolic model checking

## Example (Cardinality Constraints )

$$2 \leq x_1 + \dots + x_9 \leq 3$$

## Example (Cardinality Constraints using BDDs)

$$2 \leq x_1 + \dots + x_9 \leq 3$$



# Outline

## 1. Summary of Previous Lecture

## 2. Algorithms for Binary Decision Diagrams

Reduce

Restrict

Apply

Quantification

## 3. Intermezzo

## 4. Hidden Weighted Bit Function

## 5. Predicate Logic

## 6. Further Reading

## Reduce Algorithm

input: • OBDD

output: • equivalent **reduced** OBDD with compatible variable ordering

## Reduce Algorithm

input: • OBDD

output: • equivalent reduced OBDD with compatible variable ordering

## Idea

assign natural number  $id(n)$  to every node  $n$  while traversing input BDD layer by layer in bottom-up manner

## Reduce Algorithm

- input:     • OBDD
- output:    • equivalent reduced OBDD with compatible variable ordering

## Idea

assign natural number  $\text{id}(n)$  to every node  $n$  while traversing input BDD layer by layer in bottom-up manner

## Notation

BDD  $B_f$  of boolean function  $f$  has root node  $r_f$



## Reduce Algorithm

input:     • OBDD

output:    • equivalent reduced OBDD with compatible variable ordering

▶ assign #0 to all terminal nodes labelled 0



## Reduce Algorithm

input: • OBDD

output: • equivalent reduced OBDD with compatible variable ordering

- ▶ assign #0 to all terminal nodes labelled 0
- ▶ assign #1 to all terminal nodes labelled 1

## Reduce Algorithm

input: • OBDD

output: • equivalent reduced OBDD with compatible variable ordering

- ▶ assign #0 to all terminal nodes labelled 0
- ▶ assign #1 to all terminal nodes labelled 1
- ▶ non-terminal node  $n$  with variable  $x$ :

## Reduce Algorithm

input: • OBDD

output: • equivalent reduced OBDD with compatible variable ordering

- ▶ assign #0 to all terminal nodes labelled 0
- ▶ assign #1 to all terminal nodes labelled 1
- ▶ non-terminal node  $n$  with variable  $x$ :
  - ① if  $\text{id}(\text{lo}(n)) = \text{id}(\text{hi}(n))$  then  $\text{id}(n) = \text{id}(\text{lo}(n))$

## Reduce Algorithm

input: • OBDD

output: • equivalent reduced OBDD with compatible variable ordering

▶ assign #0 to all terminal nodes labelled 0

▶ assign #1 to all terminal nodes labelled 1

▶ non-terminal node  $n$  with variable  $x$ :

① if  $\text{id}(\text{lo}(n)) = \text{id}(\text{hi}(n))$  then  $\text{id}(n) = \text{id}(\text{lo}(n))$

② if there exists node  $m \neq n$  with same variable  $x$  and  $\text{id}(m)$  defined such that

$$\text{id}(\text{lo}(m)) = \text{id}(\text{lo}(n)) \quad \text{and} \quad \text{id}(\text{hi}(m)) = \text{id}(\text{hi}(n))$$

then  $\text{id}(n) = \text{id}(m)$

## Reduce Algorithm

input: • OBDD

output: • equivalent reduced OBDD with compatible variable ordering

▶ assign #0 to all terminal nodes labelled 0

▶ assign #1 to all terminal nodes labelled 1

▶ non-terminal node  $n$  with variable  $x$ :

① if  $\text{id}(\text{lo}(n)) = \text{id}(\text{hi}(n))$  then  $\text{id}(n) = \text{id}(\text{lo}(n))$

② if there exists node  $m \neq n$  with same variable  $x$  and  $\text{id}(m)$  defined such that

$$\text{id}(\text{lo}(m)) = \text{id}(\text{lo}(n)) \quad \text{and} \quad \text{id}(\text{hi}(m)) = \text{id}(\text{hi}(n))$$

then  $\text{id}(n) = \text{id}(m)$

③ otherwise  $\text{id}(n) = \text{next unused natural number}$

## Reduce Algorithm

input: • OBDD

output: • equivalent reduced OBDD with compatible variable ordering

▶ assign #0 to all terminal nodes labelled 0

▶ assign #1 to all terminal nodes labelled 1

▶ non-terminal node  $n$  with variable  $x$ :

① if  $\text{id}(\text{lo}(n)) = \text{id}(\text{hi}(n))$  then  $\text{id}(n) = \text{id}(\text{lo}(n))$

② if there exists node  $m \neq n$  with same variable  $x$  and  $\text{id}(m)$  defined such that

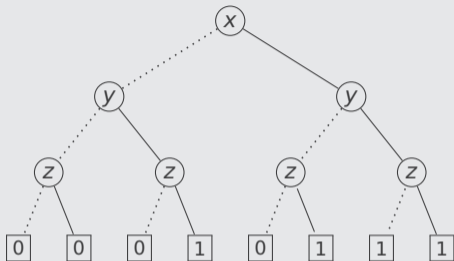
$$\text{id}(\text{lo}(m)) = \text{id}(\text{lo}(n)) \quad \text{and} \quad \text{id}(\text{hi}(m)) = \text{id}(\text{hi}(n))$$

then  $\text{id}(n) = \text{id}(m)$

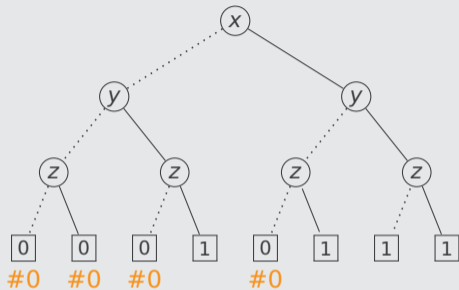
③ otherwise  $\text{id}(n) = \text{next unused natural number}$

▶ share nodes with same label

## Example

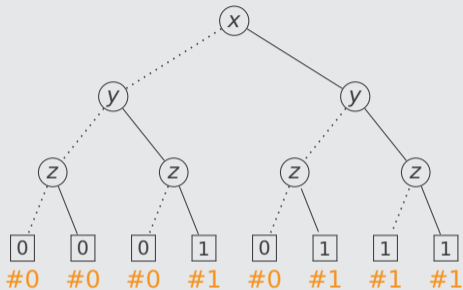


# Example

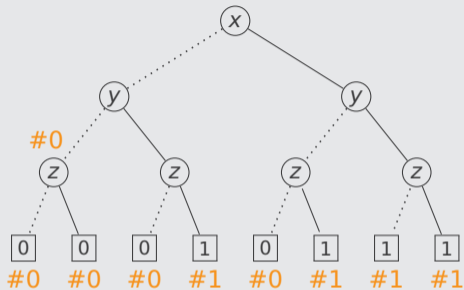




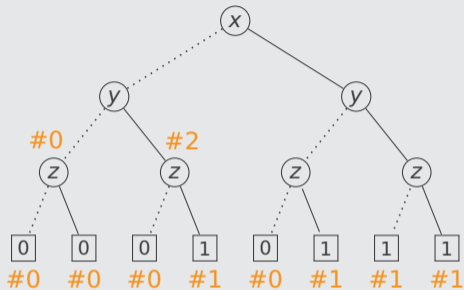
# Example



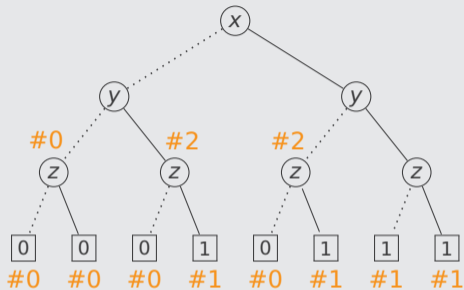
# Example



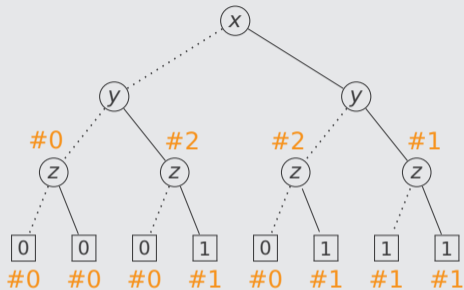
# Example



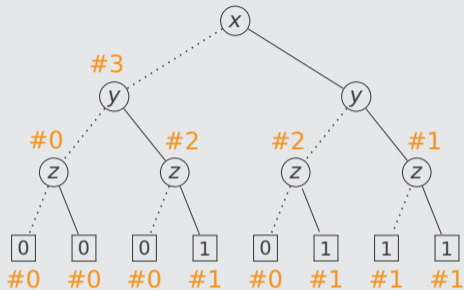
# Example



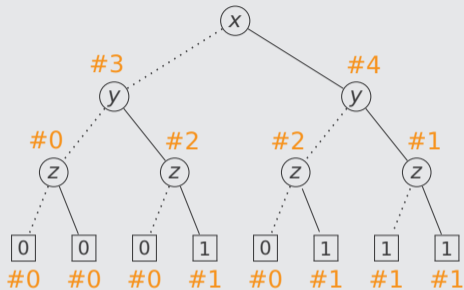
# Example



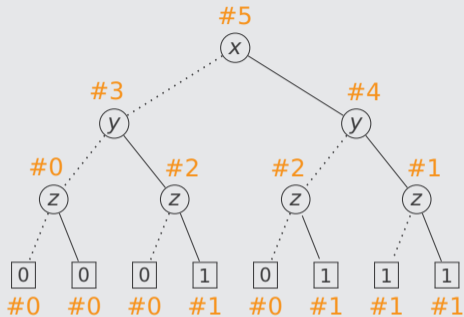
# Example



# Example

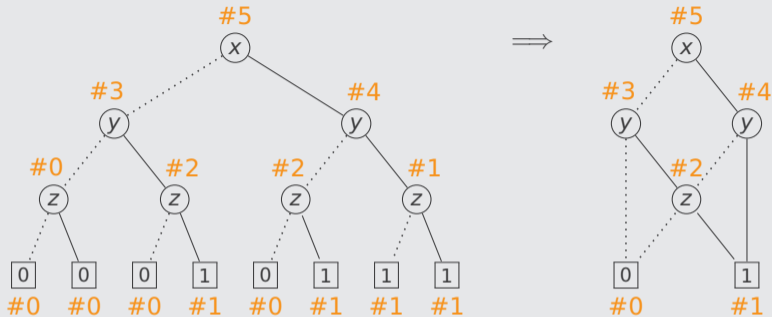


# Example





# Example



# Outline

## 1. Summary of Previous Lecture

## 2. Algorithms for Binary Decision Diagrams

Reduce

Restrict

Apply

Quantification

## 3. Intermezzo

## 4. Hidden Weighted Bit Function

## 5. Predicate Logic

## 6. Further Reading

## Definition

**restriction** of boolean function  $f$  with respect to variable  $x$ :

$f[0/x]$  replace all occurrences of  $x$  in  $f$  by 0

$f[1/x]$  replace all occurrences of  $x$  in  $f$  by 1

## Definition

restriction of boolean function  $f$  with respect to variable  $x$ :

$f[0/x]$  replace all occurrences of  $x$  in  $f$  by 0

$f[1/x]$  replace all occurrences of  $x$  in  $f$  by 1

## Example

$$f = x \cdot (y + \bar{x})$$

## Definition

restriction of boolean function  $f$  with respect to variable  $x$ :

$f[0/x]$  replace all occurrences of  $x$  in  $f$  by 0

$f[1/x]$  replace all occurrences of  $x$  in  $f$  by 1

## Example

$$f = x \cdot (y + \bar{x})$$

▶  $f[0/x] = 0 \cdot (y + \bar{0})$

## Definition

restriction of boolean function  $f$  with respect to variable  $x$ :

$f[0/x]$  replace all occurrences of  $x$  in  $f$  by 0

$f[1/x]$  replace all occurrences of  $x$  in  $f$  by 1

## Example

$$f = x \cdot (y + \bar{x})$$

►  $f[0/x] = 0 \cdot (y + \bar{0}) = 0$

## Definition

restriction of boolean function  $f$  with respect to variable  $x$ :

$f[0/x]$  replace all occurrences of  $x$  in  $f$  by 0

$f[1/x]$  replace all occurrences of  $x$  in  $f$  by 1

## Example

$$f = x \cdot (y + \bar{x})$$

▶  $f[0/x] = 0 \cdot (y + \bar{0}) = 0$

▶  $f[1/x] = 1 \cdot (y + \bar{1}) = y$

## Definition

restriction of boolean function  $f$  with respect to variable  $x$ :

$f[0/x]$  replace all occurrences of  $x$  in  $f$  by 0

$f[1/x]$  replace all occurrences of  $x$  in  $f$  by 1

## Example

$$f = x \cdot (y + \bar{x})$$

▶  $f[0/x] = 0 \cdot (y + \bar{0}) = 0$

▶  $f[1/x] = 1 \cdot (y + \bar{1}) = y$

▶  $f[0/y] = x \cdot (0 + \bar{x}) = 0$



## Definition

restriction of boolean function  $f$  with respect to variable  $x$ :

$f[0/x]$  replace all occurrences of  $x$  in  $f$  by 0

$f[1/x]$  replace all occurrences of  $x$  in  $f$  by 1

## Example

$$f = x \cdot (y + \bar{x})$$

▶  $f[0/x] = 0 \cdot (y + \bar{0}) = 0$

▶  $f[1/x] = 1 \cdot (y + \bar{1}) = y$

▶  $f[0/y] = x \cdot (0 + \bar{x}) = 0$

▶  $f[1/y] = x \cdot (1 + \bar{x}) = x$

## Theorem (Shannon expansion)

$f = \bar{x} \cdot f[0/x] + x \cdot f[1/x]$  for every boolean function  $f$  and variable  $x$

## Theorem (Shannon expansion)

$f = \bar{x} \cdot f[0/x] + x \cdot f[1/x]$  for every boolean function  $f$  and variable  $x$

## Notational Convention

operator precedence  $\cdot > \oplus, +$

## Theorem (Shannon expansion)

$f = \bar{x} \cdot f[0/x] + x \cdot f[1/x]$  for every boolean function  $f$  and variable  $x$

## Notational Convention

operator precedence  $\cdot > \oplus, +$

## Restrict Algorithm

input: 

- OBDD  $B_f$ , variable  $x$ , value  $i \in \{0, 1\}$

output: 

- reduced OBDD of  $f[i/x]$  with compatible variable ordering

## Theorem (Shannon expansion)

$f = \bar{x} \cdot f[0/x] + x \cdot f[1/x]$  for every boolean function  $f$  and variable  $x$

## Notational Convention

operator precedence  $\cdot > \oplus, +$

## Restrict Algorithm

input: 

- OBDD  $B_f$ , variable  $x$ , value  $i \in \{0, 1\}$

output: 

- reduced OBDD of  $f[i/x]$  with compatible variable ordering

① **redirect** every incoming edge of node  $n$  labelled with  $x$  to

- ▶  $lo(n)$  if  $i = 0$
- ▶  $hi(n)$  if  $i = 1$

## Theorem (Shannon expansion)

$f = \bar{x} \cdot f[0/x] + x \cdot f[1/x]$  for every boolean function  $f$  and variable  $x$

## Notational Convention

operator precedence  $\cdot > \oplus, +$

## Restrict Algorithm

input: 

- OBDD  $B_f$ , variable  $x$ , value  $i \in \{0, 1\}$

output: 

- reduced OBDD of  $f[i/x]$  with compatible variable ordering

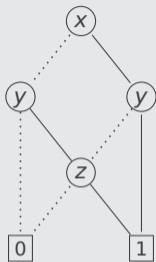
① redirect every incoming edge of node  $n$  labelled with  $x$  to

▶  $lo(n)$  if  $i = 0$

▶  $hi(n)$  if  $i = 1$

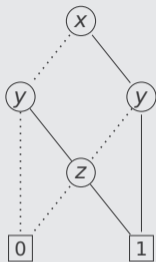
② **reduce** resulting OBDD

## Example



$$f = \bar{x}yz + x(y + z)$$

## Example

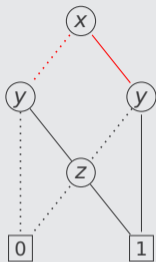


$$f = \bar{x}yz + x(y + z)$$

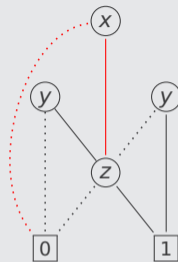
$$f[0/y]$$



## Example

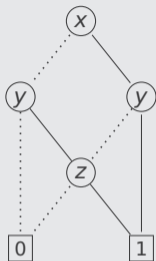


$$f = \bar{x}yz + x(y + z)$$

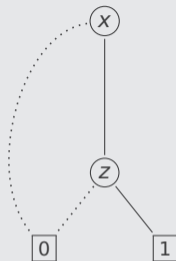


$$f[0/y]$$

## Example



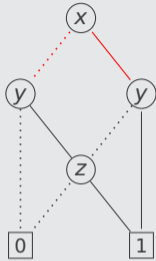
$$f = \bar{x}yz + x(y + z)$$



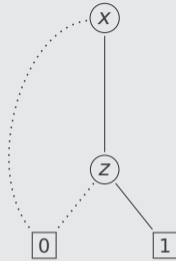
$$f[0/y]$$

inaccessible nodes are taken care of by garbage collector

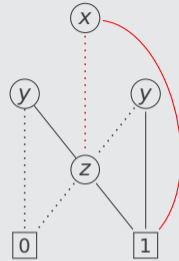
# Example



$$f = \bar{x}yz + x(y + z)$$

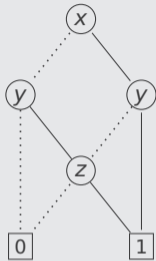


$$f[0/y]$$

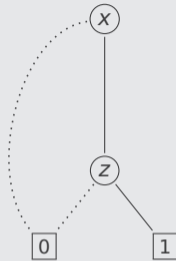


$$f[1/y]$$

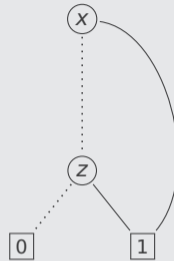
## Example



$$f = \bar{x}yz + x(y + z)$$



$$f[0/y]$$



$$f[1/y]$$

inaccessible nodes are taken care of by garbage collector

# Outline

## 1. Summary of Previous Lecture

## 2. Algorithms for Binary Decision Diagrams

Reduce

Restrict

Apply

Quantification

## 3. Intermezzo

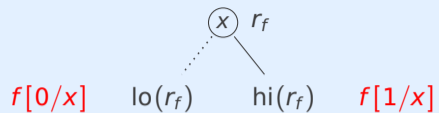
## 4. Hidden Weighted Bit Function

## 5. Predicate Logic

## 6. Further Reading

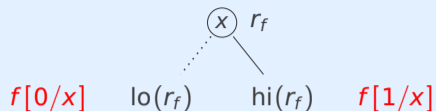
## Notation

BDD  $B_f$  of boolean function  $f$  has root node  $r_f$



## Notation

BDD  $B_f$  of boolean function  $f$  has root node  $r_f$

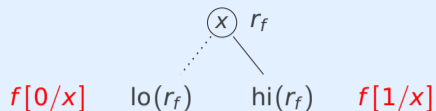


## Apply Algorithm

- input:
- binary operation  $\star$  on boolean functions
  - OBDDs  $B_f$  and  $B_g$  with compatible variable orderings
- output:
- reduced OBDD of  $f \star g$  with compatible variable ordering

## Notation

BDD  $B_f$  of boolean function  $f$  has root node  $r_f$



## Apply Algorithm

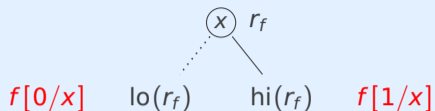
- input:
- binary operation  $\star$  on boolean functions
  - OBDDs  $B_f$  and  $B_g$  with compatible variable orderings
- output:
- reduced OBDD of  $f \star g$  with compatible variable ordering

$$f \star g = \bar{x} \cdot (f \star g)[0/x] + x \cdot (f \star g)[1/x]$$



## Notation

BDD  $B_f$  of boolean function  $f$  has root node  $r_f$



## Apply Algorithm

- input:
- binary operation  $\star$  on boolean functions
  - OBDDs  $B_f$  and  $B_g$  with compatible variable orderings
- output:
- reduced OBDD of  $f \star g$  with compatible variable ordering

$$\begin{aligned} f \star g &= \bar{x} \cdot (f \star g)[0/x] + x \cdot (f \star g)[1/x] \\ &= \bar{x} \cdot \underbrace{(f[0/x] \star g[0/x])}_{\text{simpler than } f \star g} + x \cdot \underbrace{(f[1/x] \star g[1/x])}_{\text{simpler than } f \star g} \end{aligned}$$

**case I**     $r_f, r_g$  terminal nodes with labels  $l_f, l_g$

## Apply Algorithm $\text{apply}(\star, B_f, B_g)$

**case I**  $r_f, r_g$  terminal nodes with labels  $l_f, l_g$

**return**  $l_f \star l_g$

## Apply Algorithm $\text{apply}(\star, B_f, B_g)$

**case I**  $r_f, r_g$  terminal nodes with labels  $l_f, l_g$

**return**  $l_f \star l_g$

**case II**  $r_f, r_g$  non-terminal nodes with same label  $x$

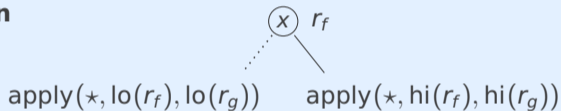
## Apply Algorithm $\text{apply}(\star, B_f, B_g)$

**case I**  $r_f, r_g$  terminal nodes with labels  $l_f, l_g$

**return**  $l_f \star l_g$

**case II**  $r_f, r_g$  non-terminal nodes with same label  $x$

**return**



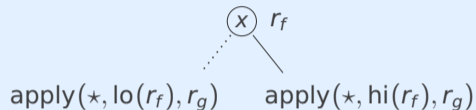
**case III**     $r_f$  non-terminal node with label  $x$

$r_g$  terminal node or non-terminal node with label  $y > x$

**case III**     $r_f$  non-terminal node with label  $x$

$r_g$  terminal node or non-terminal node with label  $y > x$

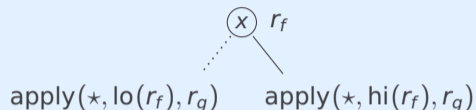
**return**



**case III**  $r_f$  non-terminal node with label  $x$

$r_g$  terminal node or non-terminal node with label  $y > x$

**return**



**case IV**  $r_g$  non-terminal node with label  $x$

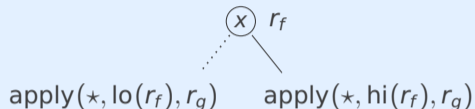
$r_f$  terminal node or non-terminal node with label  $y > x$



**case III**  $r_f$  non-terminal node with label  $x$

$r_g$  terminal node or non-terminal node with label  $y > x$

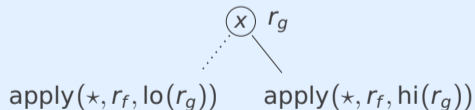
**return**



**case IV**  $r_g$  non-terminal node with label  $x$

$r_f$  terminal node or non-terminal node with label  $y > x$

**return**



**case III**     $r_f$  non-terminal node with label  $x$

$r_g$  terminal node or non-terminal node with label  $y > x$

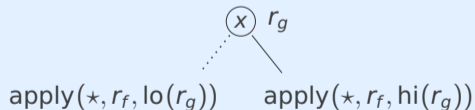
**return**



**case IV**     $r_g$  non-terminal node with label  $x$

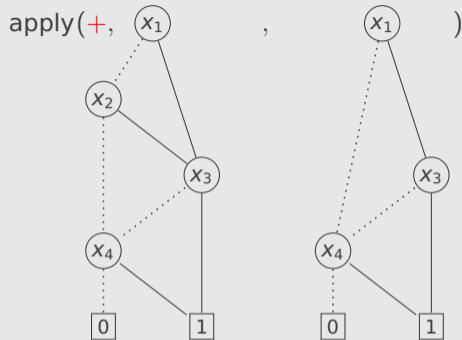
$r_f$  terminal node or non-terminal node with label  $y > x$

**return**

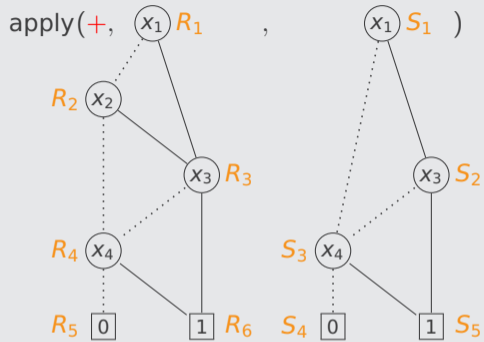


followed by application of **reduce** algorithm

# Example

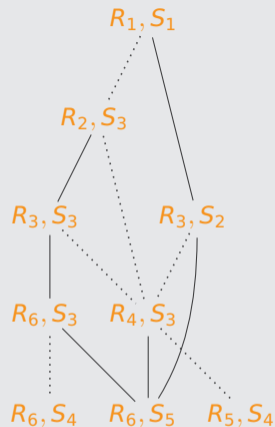
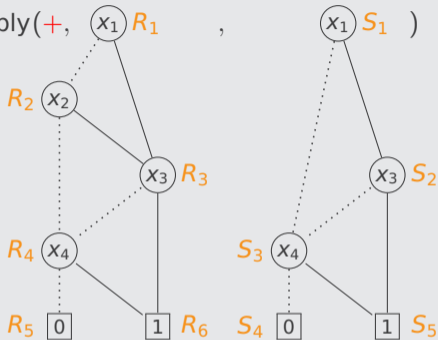


# Example



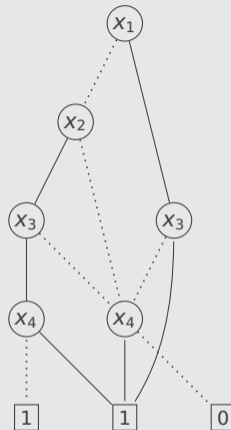
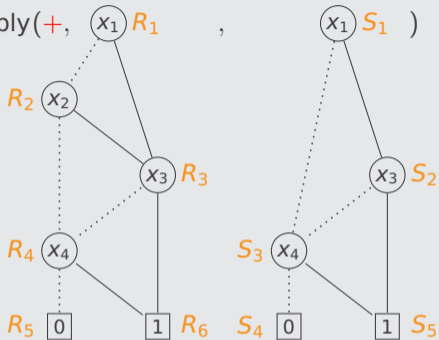
# Example

apply(+,  $x_1 R_1$  ,  $x_1 S_1$  )

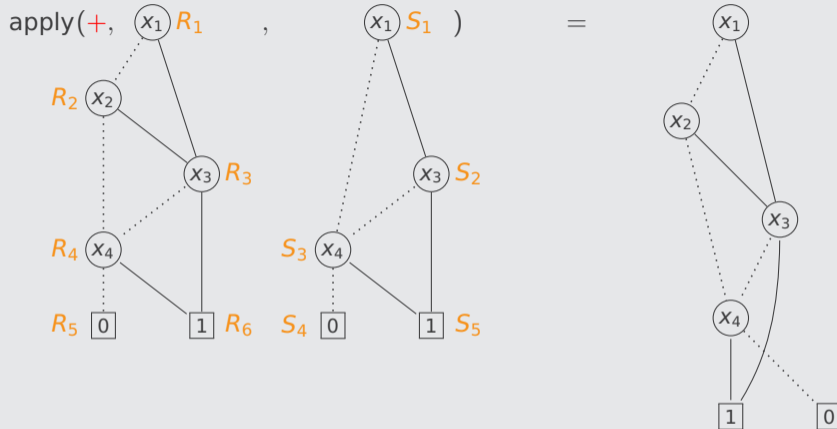


# Example

apply(+,  $x_1 R_1$  ,  $x_1 S_1$  )



# Example



# Outline

## 1. Summary of Previous Lecture

## 2. Algorithms for Binary Decision Diagrams

Reduce

Restrict

Apply

Quantification

## 3. Intermezzo

## 4. Hidden Weighted Bit Function

## 5. Predicate Logic

## 6. Further Reading



## Definition

quantification of boolean function  $f$  over variable  $x$ :

►  $\exists x.f$        $f[0/x] + f[1/x]$

## Definition

quantification of boolean function  $f$  over variable  $x$ :

▶  $\exists x.f$              $f[0/x] + f[1/x]$



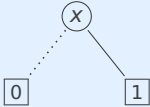
▶  $\forall x.f$              $f[0/x] \cdot f[1/x]$

## Definition

quantification of boolean function  $f$  over variable  $x$ :

- ▶  $\exists x.f$        $f[0/x] + f[1/x]$
- ▶  $\forall x.f$        $f[0/x] \cdot f[1/x]$

## Summary



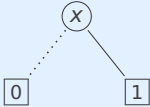
function $f$	OBDD $B_f$	function $f$	OBDD $B_f$	function $f$	OBDD $B_f$
0		$g + h$	$\text{apply}(+, B_g, B_h)$	$g[0/x]$	$\text{restrict}(0, x, B_g)$
1		$g \oplus h$	$\text{apply}(\oplus, B_g, B_h)$	$g[1/x]$	$\text{restrict}(1, x, B_g)$
$x$		$g \cdot h$	$\text{apply}(\cdot, B_g, B_h)$		
		$\bar{g}$	$\text{apply}(\oplus, B_g, B_1)$		

## Definition

quantification of boolean function  $f$  over variable  $x$ :

- ▶  $\exists x.f$        $f[0/x] + f[1/x]$
- ▶  $\forall x.f$        $f[0/x] \cdot f[1/x]$

## Summary

function $f$	OBDD $B_f$	function $f$	OBDD $B_f$	function $f$	OBDD $B_f$
0		$g + h$	$\text{apply}(+, B_g, B_h)$	$g[0/x]$	$\text{restrict}(0, x, B_g)$
1		$g \oplus h$	$\text{apply}(\oplus, B_g, B_h)$	$g[1/x]$	$\text{restrict}(1, x, B_g)$
$x$		$g \cdot h$	$\text{apply}(\cdot, B_g, B_h)$	$\exists x.g$	$\text{apply}(+, B_{g[0/x]}, B_{g[1/x]})$
		$\bar{g}$	$\text{apply}(\oplus, B_g, B_1)$	$\forall x.g$	$\text{apply}(\cdot, B_{g[0/x]}, B_{g[1/x]})$

### BoolTool

by Patrick Muxel (2004), Philipp Ruff (2006), Caroline Terzer (2006), Markus Plattner (2007), Elias Zischg (2012)

### BoolTool

by Patrick Muxel (2004), Philipp Ruff (2006), Caroline Terzer (2006), Markus Plattner (2007), Elias Zischg (2012)

### BoolTool Reloaded

by Martin Neuner (2023)

# Outline

1. Summary of Previous Lecture
2. Algorithms for Binary Decision Diagrams
- 3. Intermezzo**
4. Hidden Weighted Bit Function
5. Predicate Logic
6. Further Reading

## Questions

Which of the following statements are true ?

- A** The output of restrict has fewer nodes than the input.
- B** The number of edges in a reduced OBDD depends on the order.
- C** An OBDD for a formula with  $n$  variables has at most  $2^{n+1} - 1$  nodes.
- D** Negating a reduced OBDD does not change the number of nodes.
- E** A reduced OBDD with 12 nodes containing up to 4 variables exists.





# Outline

1. Summary of Previous Lecture
2. Algorithms for Binary Decision Diagrams
3. Intermezzo
- 4. Hidden Weighted Bit Function**
5. Predicate Logic
6. Further Reading

## Definitions

►  $\text{wt}(x_1, \dots, x_n) = \sum_{i=1}^n x_i$

## Definitions

- ▶  $\text{wt}(x_1, \dots, x_n) = \sum_{i=1}^n x_i$
- ▶  $\text{HWB}_n(x_1, \dots, x_n) = \begin{cases} 0 & \text{if } \text{wt}(x_1, \dots, x_n) = 0 \\ x_{\text{wt}(x_1, \dots, x_n)} & \text{otherwise} \end{cases}$

## Definitions

$$\text{wt}(x_1, \dots, x_n) = \sum_{i=1}^n x_i$$

$$\text{HWB}_n(x_1, \dots, x_n) = \begin{cases} 0 & \text{if } \text{wt}(x_1, \dots, x_n) = 0 \\ x_{\text{wt}(x_1, \dots, x_n)} & \text{otherwise} \end{cases}$$

## Example

$x_1$	$x_2$	$x_3$	$x_4$	$\text{HWB}_4$
0	0	0	0	0

## Definitions

$$\text{wt}(x_1, \dots, x_n) = \sum_{i=1}^n x_i$$

$$\text{HWB}_n(x_1, \dots, x_n) = \begin{cases} 0 & \text{if } \text{wt}(x_1, \dots, x_n) = 0 \\ x_{\text{wt}(x_1, \dots, x_n)} & \text{otherwise} \end{cases}$$

## Example

$x_1$	$x_2$	$x_3$	$x_4$	$\text{HWB}_4$
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0

## Definitions

$$\text{wt}(x_1, \dots, x_n) = \sum_{i=1}^n x_i$$

$$\text{HWB}_n(x_1, \dots, x_n) = \begin{cases} 0 & \text{if } \text{wt}(x_1, \dots, x_n) = 0 \\ x_{\text{wt}(x_1, \dots, x_n)} & \text{otherwise} \end{cases}$$

## Example

$x_1$	$x_2$	$x_3$	$x_4$	$\text{HWB}_4$	$x_1$	$x_2$	$x_3$	$x_4$	$\text{HWB}_4$
0	0	0	0	0	0	1	0	0	0
0	0	0	1	0	0	1	0	1	1
0	0	1	0	0	0	1	1	0	1
0	0	1	1	0	0	1	1	1	1

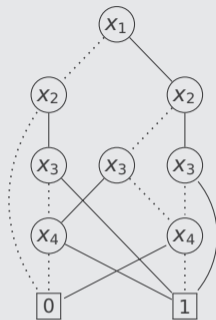
## Definitions

- ▶  $\text{wt}(x_1, \dots, x_n) = \sum_{i=1}^n x_i$
- ▶  $\text{HWB}_n(x_1, \dots, x_n) = \begin{cases} 0 & \text{if } \text{wt}(x_1, \dots, x_n) = 0 \\ x_{\text{wt}(x_1, \dots, x_n)} & \text{otherwise} \end{cases}$

## Example

$x_1$	$x_2$	$x_3$	$x_4$	$\text{HWB}_4$	$x_1$	$x_2$	$x_3$	$x_4$	$\text{HWB}_4$	$x_1$	$x_2$	$x_3$	$x_4$	$\text{HWB}_4$	$x_1$	$x_2$	$x_3$	$x_4$	$\text{HWB}_4$
0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	1	1	0	0	1
0	0	0	1	0	0	0	1	0	1	1	0	0	1	0	1	1	0	1	0
0	0	1	0	0	0	0	1	1	0	1	0	1	0	0	1	1	1	0	1
0	0	1	1	0	0	0	1	1	1	1	0	1	1	1	1	1	1	1	1

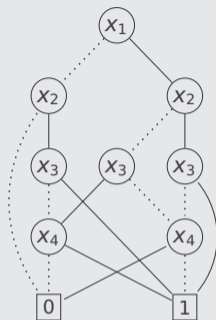
## Example



reduced OBDD



## Example

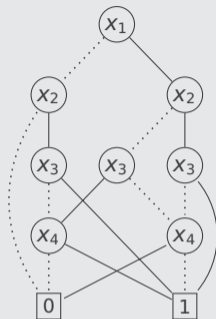


reduced OBDD

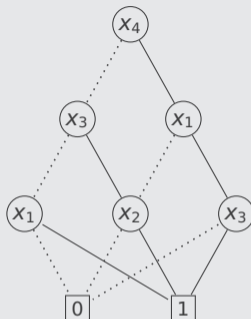
## Theorem

- ▶ every reduced OBDD computing  $\text{HWB}_n$  has size **exponential** in  $n$

## Example



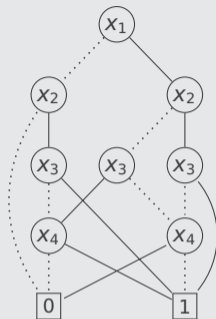
reduced OBDD



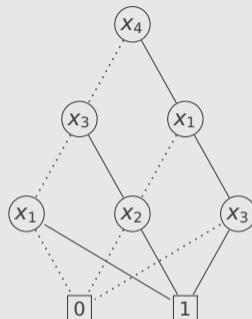
## Theorem

- ▶ every reduced OBDD computing  $\text{HWB}_n$  has size exponential in  $n$
- ▶ some reduced BDD computing  $\text{HWB}_n$  has size **quadratic** in  $n$

## Example



reduced OBDD



free (read-1) BDD

## Theorem

- ▶ every reduced OBDD computing  $\text{HWB}_n$  has size exponential in  $n$
- ▶ some reduced BDD computing  $\text{HWB}_n$  has size quadratic in  $n$

# Outline

1. Summary of Previous Lecture
2. Algorithms for Binary Decision Diagrams
3. Intermezzo
4. Hidden Weighted Bit Function

## 5. Predicate Logic

Introduction

Syntax

Free and Bound Variables

Substitution

## 6. Further Reading

## Definition

propositional formulas are built from

- ▶ atoms  $p, q, r, p_1, p_2, \dots$
- ▶ bottom  $\perp$
- ▶ top  $\top$
- ▶ negation  $\neg$   $\neg p$  "not  $p$ "
- ▶ conjunction  $\wedge$   $p \wedge q$  " $p$  and  $q$ "
- ▶ disjunction  $\vee$   $p \vee q$  " $p$  or  $q$ "
- ▶ implication  $\rightarrow$   $p \rightarrow q$  "if  $p$  then  $q$ "

according to following Backus–Naur Form:

$$\varphi ::= p \mid \perp \mid \top \mid (\neg \varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi)$$

- ▶ Mary admires every professor

## Propositional Logic is Not Very Expressive

- ▶ Mary admires every professor
- ▶ some professor admires Mary

## Propositional Logic is Not Very Expressive

- ▶ Mary admires every professor
- ▶ some professor admires Mary
- ▶ Mary admires herself



## Propositional Logic is Not Very Expressive

- ▶ Mary admires every professor
- ▶ some professor admires Mary
- ▶ Mary admires herself
- ▶ no student attended every lecture

## Propositional Logic is Not Very Expressive

- ▶ Mary admires every professor
- ▶ some professor admires Mary
- ▶ Mary admires herself
- ▶ no student attended every lecture
- ▶ no lecture was attended by every student

## Propositional Logic is Not Very Expressive

- ▶ Mary admires every professor
- ▶ some professor admires Mary
- ▶ Mary admires herself
- ▶ no student attended every lecture
- ▶ no lecture was attended by every student
- ▶ no lecture was attended by any student

## Propositional Logic is Not Very Expressive

statements like

- ▶ Mary admires every professor
- ▶ some professor admires Mary
- ▶ Mary admires herself
- ▶ no student attended every lecture
- ▶ no lecture was attended by every student
- ▶ no lecture was attended by any student

cannot be expressed **adequately** in propositional logic

concept

notation

intended meaning

predicate symbols

$P, Q, R, A, B, \dots$

relations over domain

concept	notation	intended meaning
predicate symbols	$P, Q, R, A, B, \dots$	relations over domain
<b>function symbols</b>	$f, g, h, a, b, \dots$	functions over domain

concept	notation	intended meaning
predicate symbols	$P, Q, R, A, B, \dots$	relations over domain
function symbols	$f, g, h, a, b, \dots$	functions over domain
<b>variables</b>	$x, y, z, \dots$	(unspecified) elements of domain

concept	notation	intended meaning
predicate symbols	$P, Q, R, A, B, \dots$	relations over domain
function symbols	$f, g, h, a, b, \dots$	functions over domain
variables	$x, y, z, \dots$	(unspecified) elements of domain
quantifiers	$\forall, \exists$	for all, for some



concept	notation	intended meaning
predicate symbols	$P, Q, R, A, B, \dots$	relations over domain
function symbols	$f, g, h, a, b, \dots$	functions over domain
variables	$x, y, z, \dots$	(unspecified) elements of domain
quantifiers	$\forall, \exists$	for all, for some
<b>connectives</b>	$\neg, \wedge, \vee, \rightarrow$	

concept	notation	intended meaning
predicate symbols	$P, Q, R, A, B, \dots$	relations over domain
function symbols	$f, g, h, a, b, \dots$	functions over domain
variables	$x, y, z, \dots$	(unspecified) elements of domain
quantifiers	$\forall, \exists$	for all, for some
connectives	$\neg, \wedge, \vee, \rightarrow$	

## Remarks

- ▶ function and predicate symbols take fixed number of arguments (**arity**)

concept	notation	intended meaning
predicate symbols	$P, Q, R, A, B, \dots$	relations over domain
function symbols	$f, g, h, a, b, \dots$	functions over domain
variables	$x, y, z, \dots$	(unspecified) elements of domain
quantifiers	$\forall, \exists$	for all, for some
connectives	$\neg, \wedge, \vee, \rightarrow$	

## Remarks

- ▶ function and predicate symbols take fixed number of arguments (arity)
- ▶ function and predicate symbols of arity 0 are called **constants**

concept	notation	intended meaning
predicate symbols	$P, Q, R, A, B, \dots$	relations over domain
function symbols	$f, g, h, a, b, \dots$	functions over domain
variables	$x, y, z, \dots$	(unspecified) elements of domain
quantifiers	$\forall, \exists$	for all, for some
connectives	$\neg, \wedge, \vee, \rightarrow$	

## Remarks

- ▶ function and predicate symbols take fixed number of arguments (arity)
- ▶ function and predicate symbols of arity 0 are called constants
- ▶  $=$  (equality) is designated predicate symbol of arity 2

## Example (Exercise 2.1.1)

- ▶ Mary admires every professor
- ▶ some professor admires Mary
- ▶ Mary admires herself
- ▶ no student attended every lecture
- ▶ no lecture was attended by every student
- ▶ no lecture was attended by any student

## Example (Exercise 2.1.1)

- ▶ Mary **admires** every professor
- ▶ some professor **admires** Mary
- ▶ Mary **admires** herself
- ▶ no student attended every lecture
- ▶ no lecture was attended by every student
- ▶ no lecture was attended by any student

$A(x,y)$   $x$  admires  $y$

## Example (Exercise 2.1.1)

- ▶ Mary admires every professor
- ▶ some professor admires Mary
- ▶ Mary admires herself
- ▶ no student **attended** every lecture
- ▶ no lecture was **attended** by every student
- ▶ no lecture was **attended** by any student

$A(x, y)$   $x$  admires  $y$

$B(x, y)$   $x$  attended  $y$

## Example (Exercise 2.1.1)

- ▶ Mary admires every professor
- ▶ some professor admires Mary
- ▶ Mary admires herself
- ▶ no student attended every lecture
- ▶ no lecture was attended by every student
- ▶ no lecture was attended by any student

$A(x, y)$   $x$  admires  $y$

$P(x)$   $x$  is professor

$B(x, y)$   $x$  attended  $y$



## Example (Exercise 2.1.1)

- ▶ Mary admires every professor
- ▶ some professor admires Mary
- ▶ Mary admires herself
- ▶ no **student** attended every lecture
- ▶ no lecture was attended by every **student**
- ▶ no lecture was attended by any **student**

$A(x, y)$   $x$  admires  $y$

$P(x)$   $x$  is professor

$B(x, y)$   $x$  attended  $y$

$S(x)$   $x$  is student

## Example (Exercise 2.1.1)

- ▶ Mary admires every professor
- ▶ some professor admires Mary
- ▶ Mary admires herself
- ▶ no student attended every lecture
- ▶ no lecture was attended by every student
- ▶ no lecture was attended by any student

$A(x, y)$   $x$  admires  $y$

$P(x)$   $x$  is professor

$L(x)$   $x$  is lecture

$B(x, y)$   $x$  attended  $y$

$S(x)$   $x$  is student

## Example (Exercise 2.1.1)

- ▶ **Mary** admires every professor
- ▶ some professor admires **Mary**
- ▶ **Mary** admires herself
- ▶ no student attended every lecture
- ▶ no lecture was attended by every student
- ▶ no lecture was attended by any student

$A(x, y)$   $x$  admires  $y$

$P(x)$   $x$  is professor

$L(x)$   $x$  is lecture

$B(x, y)$   $x$  attended  $y$

$S(x)$   $x$  is student

$m$  Mary

## Example (Exercise 2.1.1)

- ▶ Mary admires every professor
- ▶ some professor admires Mary
- ▶ Mary admires herself
- ▶ no student attended every lecture
- ▶ no lecture was attended by every student
- ▶ no lecture was attended by any student

$A(x, y)$   $x$  admires  $y$

$P(x)$   $x$  is professor

$L(x)$   $x$  is lecture

$B(x, y)$   $x$  attended  $y$

$S(x)$   $x$  is student

$m$  Mary

$A, B$  binary predicate symbols

$P, S, L$  unary predicate symbols

$m$  function symbol of arity 0

# Outline

1. Summary of Previous Lecture
2. Algorithms for Binary Decision Diagrams
3. Intermezzo
4. Hidden Weighted Bit Function

## 5. Predicate Logic

Introduction

Syntax

Free and Bound Variables

Substitution

## 6. Further Reading

## Definitions

- ▶ **terms** are built from function symbols and variables according to following BNF grammar:

$$t ::= x \mid c \mid f(t, \dots, t)$$

## Definitions

- ▶ terms are built from function symbols and variables according to following BNF grammar:

$$t ::= x \mid c \mid f(t, \dots, t)$$

- ▶ **formulas** are built from predicate symbols, terms, connectives and quantifiers according to following BNF grammar:

$$\varphi ::= P \mid P(t, \dots, t) \mid (t = t) \mid \perp \mid \top \mid (\neg \varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid (\forall x \varphi) \mid (\exists x \varphi)$$

## Definitions

- ▶ terms are built from function symbols and variables according to following BNF grammar:

$$t ::= x \mid c \mid f(t, \dots, t)$$

- ▶ formulas are built from predicate symbols, terms, connectives and quantifiers according to following BNF grammar:

$$\varphi ::= P \mid P(t, \dots, t) \mid (t = t) \mid \perp \mid \top \mid (\neg \varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid (\forall x \varphi) \mid (\exists x \varphi)$$

- ▶ notational conventions:

- ▶ binding precedence  $= > \neg, \forall, \exists > \wedge, \vee > \rightarrow$



## Definitions

- ▶ terms are built from function symbols and variables according to following BNF grammar:

$$t ::= x \mid c \mid f(t, \dots, t)$$

- ▶ formulas are built from predicate symbols, terms, connectives and quantifiers according to following BNF grammar:

$$\varphi ::= P \mid P(t, \dots, t) \mid (t = t) \mid \perp \mid \top \mid (\neg \varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid (\forall x \varphi) \mid (\exists x \varphi)$$

- ▶ notational conventions:

- ▶ binding precedence  $= > \neg, \forall, \exists > \wedge, \vee > \rightarrow$

- ▶ omit outer parentheses

## Definitions

- ▶ terms are built from function symbols and variables according to following BNF grammar:

$$t ::= x \mid c \mid f(t, \dots, t)$$

- ▶ formulas are built from predicate symbols, terms, connectives and quantifiers according to following BNF grammar:

$$\varphi ::= P \mid P(t, \dots, t) \mid (t = t) \mid \perp \mid \top \mid (\neg \varphi) \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid (\forall x \varphi) \mid (\exists x \varphi)$$

- ▶ notational conventions:

- ▶ binding precedence  $= > \neg, \forall, \exists > \wedge, \vee > \rightarrow$

- ▶ omit outer parentheses

- ▶  $\rightarrow, \wedge, \vee$  are right-associative

## Example (Exercise 2.1.1, cont'd)

$A(x, y)$   $x$  admires  $y$

$P(x)$   $x$  is professor

$L(x)$   $x$  is lecture

$B(x, y)$   $x$  attended  $y$

$S(x)$   $x$  is student

$m$  Mary

- ▶ Mary admires every professor
- ▶ some professor admires Mary
- ▶ Mary admires herself
- ▶ no student attended every lecture
- ▶ no lecture was attended by every student
- ▶ no lecture was attended by any student

## Example (Exercise 2.1.1, cont'd)

$A(x, y)$   $x$  admires  $y$

$P(x)$   $x$  is professor

$L(x)$   $x$  is lecture

$B(x, y)$   $x$  attended  $y$

$S(x)$   $x$  is student

$m$  Mary

▶ Mary admires **every** professor

$\forall x (P(x) \rightarrow A(m, x))$

▶ some professor admires Mary

▶ Mary admires herself

▶ no student attended every lecture

▶ no lecture was attended by every student

▶ no lecture was attended by any student

## Example (Exercise 2.1.1, cont'd)

$A(x, y)$   $x$  admires  $y$

$P(x)$   $x$  is professor

$L(x)$   $x$  is lecture

$B(x, y)$   $x$  attended  $y$

$S(x)$   $x$  is student

$m$  Mary

▶ Mary admires every professor

$$\forall x (P(x) \rightarrow A(m, x))$$

▶ **some** professor admires Mary

$$\exists x (P(x) \wedge A(x, m))$$

▶ Mary admires herself

▶ no student attended every lecture

▶ no lecture was attended by every student

▶ no lecture was attended by any student

## Example (Exercise 2.1.1, cont'd)

$A(x, y)$   $x$  admires  $y$

$P(x)$   $x$  is professor

$L(x)$   $x$  is lecture

$B(x, y)$   $x$  attended  $y$

$S(x)$   $x$  is student

$m$  Mary

▶ Mary admires every professor

$\forall x (P(x) \rightarrow A(m, x))$

▶ some professor admires Mary

$\exists x (P(x) \wedge A(x, m))$

▶ Mary admires herself

$A(m, m)$

▶ no student attended every lecture

▶ no lecture was attended by every student

▶ no lecture was attended by any student

## Example (Exercise 2.1.1, cont'd)

$A(x, y)$   $x$  admires  $y$

$B(x, y)$   $x$  attended  $y$

$P(x)$   $x$  is professor

$S(x)$   $x$  is student

$L(x)$   $x$  is lecture

$m$  Mary

► Mary admires every professor

$$\forall x (P(x) \rightarrow A(m, x))$$

► some professor admires Mary

$$\exists x (P(x) \wedge A(x, m))$$

► Mary admires herself

$$A(m, m)$$

► **no** student attended **every** lecture

$$\neg \exists x (S(x) \wedge \forall y (L(y) \rightarrow B(x, y)))$$

► no lecture was attended by every student

► no lecture was attended by any student

## Example (Exercise 2.1.1, cont'd)

$A(x, y)$   $x$  admires  $y$

$P(x)$   $x$  is professor

$L(x)$   $x$  is lecture

$B(x, y)$   $x$  attended  $y$

$S(x)$   $x$  is student

$m$  Mary

► Mary admires every professor

$\forall x (P(x) \rightarrow A(m, x))$

► some professor admires Mary

$\exists x (P(x) \wedge A(x, m))$

► Mary admires herself

$A(m, m)$

► no student attended every lecture

$\neg \exists x (S(x) \wedge \forall y (L(y) \rightarrow B(x, y)))$

► **no** lecture was attended by **every** student

$\neg \exists x (L(x) \wedge \forall y (S(y) \rightarrow B(y, x)))$

► no lecture was attended by any student



## Example (Exercise 2.1.1, cont'd)

$A(x, y)$   $x$  admires  $y$

$P(x)$   $x$  is professor

$L(x)$   $x$  is lecture

$B(x, y)$   $x$  attended  $y$

$S(x)$   $x$  is student

$m$  Mary

► Mary admires every professor

$\forall x (P(x) \rightarrow A(m, x))$

► some professor admires Mary

$\exists x (P(x) \wedge A(x, m))$

► Mary admires herself

$A(m, m)$

► no student attended every lecture

$\neg \exists x (S(x) \wedge \forall y (L(y) \rightarrow B(x, y)))$

► no lecture was attended by every student

$\neg \exists x (L(x) \wedge \forall y (S(y) \rightarrow B(y, x)))$

► no lecture was attended by any student

$\neg (\exists x \exists y (L(x) \wedge S(y) \wedge B(y, x)))$

## Example (Exercise 2.1.1, cont'd)

$A(x, y)$   $x$  admires  $y$

$P(x)$   $x$  is professor

$L(x)$   $x$  is lecture

$B(x, y)$   $x$  attended  $y$

$S(x)$   $x$  is student

$m$  Mary

► Mary admires every professor

$$\forall x (P(x) \rightarrow A(m, x))$$

► some professor admires Mary

$$\exists x (P(x) \wedge A(x, m))$$

► Mary admires herself

$$A(m, m)$$

► no student attended every lecture

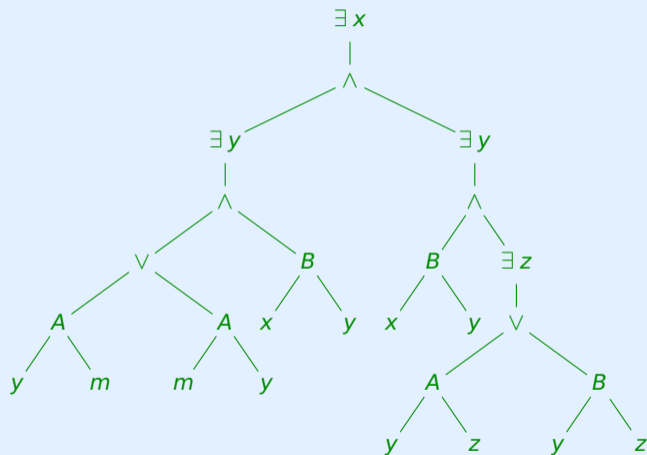
$$\neg \exists x (S(x) \wedge \forall y (L(y) \rightarrow B(x, y)))$$

► no lecture was attended by every student

$$\neg \exists x (L(x) \wedge \forall y (S(y) \rightarrow B(y, x)))$$

► no lecture was attended by any student

$$\forall x \forall y (L(x) \wedge S(y) \rightarrow \neg B(y, x))$$

$$\exists x (\exists y ((A(y, m) \vee A(m, y)) \wedge B(x, y)) \wedge \exists y (B(x, y) \wedge \exists z (A(y, z) \vee B(y, z))))$$


# Outline

1. Summary of Previous Lecture
2. Algorithms for Binary Decision Diagrams
3. Intermezzo
4. Hidden Weighted Bit Function

## 5. Predicate Logic

Introduction

Syntax

Free and Bound Variables

Substitution

## 6. Further Reading

## Definitions

- ▶ occurrence of variable  $x$  in formula  $\varphi$  is **free in  $\varphi$**  if it is leaf node in parse tree of  $\varphi$  such that there is no node  $\forall x$  or  $\exists x$  on path to root node

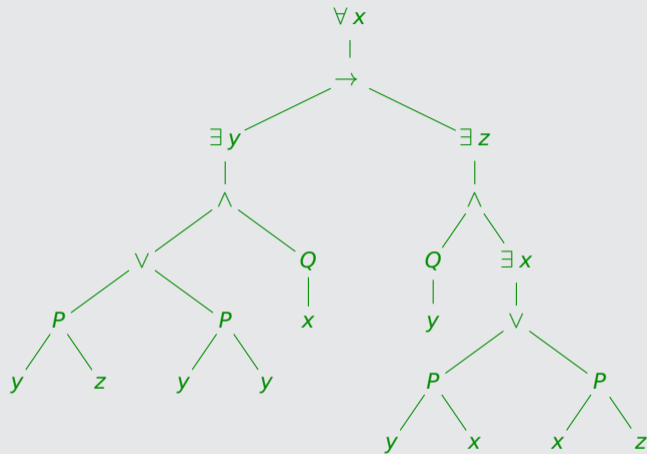
## Definitions

- ▶ occurrence of variable  $x$  in formula  $\varphi$  is free in  $\varphi$  if it is leaf node in parse tree of  $\varphi$  such that there is no node  $\forall x$  or  $\exists x$  on path to root node
- ▶ occurrence of variable  $x$  in formula  $\varphi$  is **bound** if this occurrence is not free in  $\varphi$

## Definitions

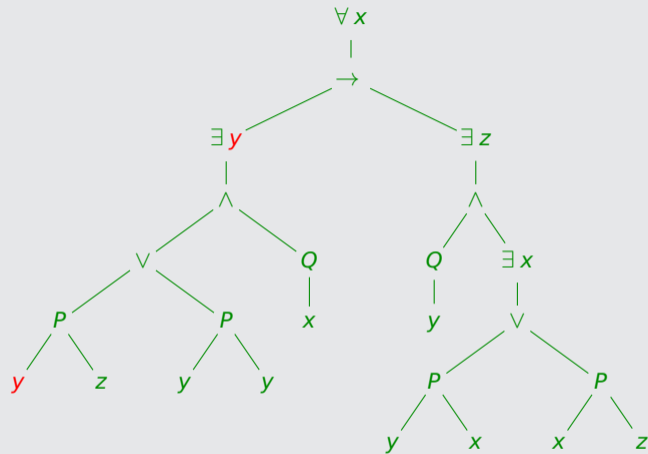
- ▶ occurrence of variable  $x$  in formula  $\varphi$  is free in  $\varphi$  if it is leaf node in parse tree of  $\varphi$  such that there is no node  $\forall x$  or  $\exists x$  on path to root node
- ▶ occurrence of variable  $x$  in formula  $\varphi$  is bound if this occurrence is not free in  $\varphi$
- ▶ **scope** of occurrence of  $\forall x$  ( $\exists x$ ) in formula  $\forall x \varphi$  ( $\exists x \varphi$ ) is  $\varphi$  except any subformula of  $\varphi$  of form  $\forall x \psi$  or  $\exists x \psi$

# Example



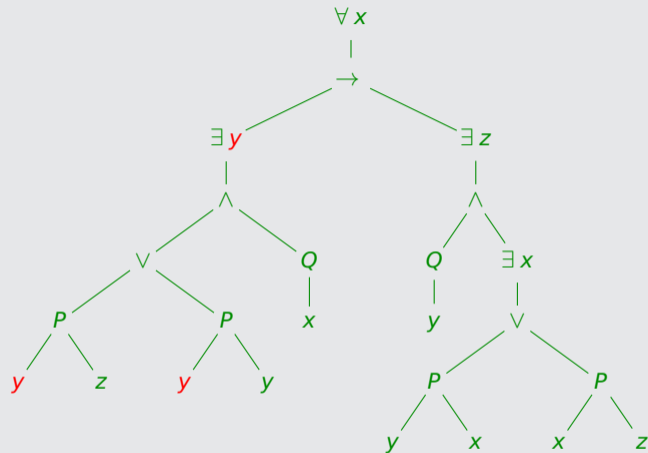


# Example



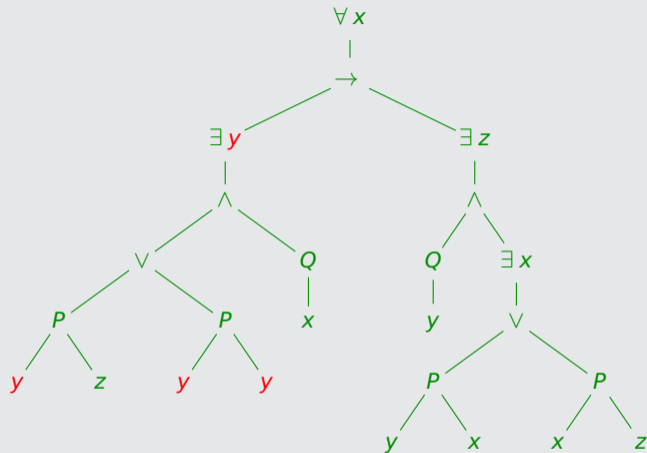
bound occurrences of variables

## Example



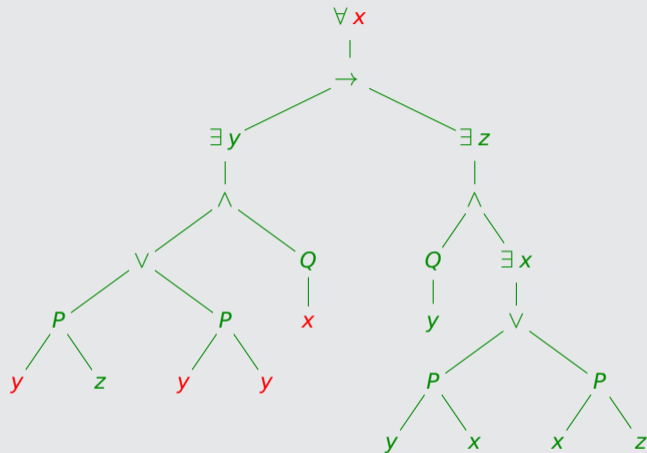
bound occurrences of variables

## Example



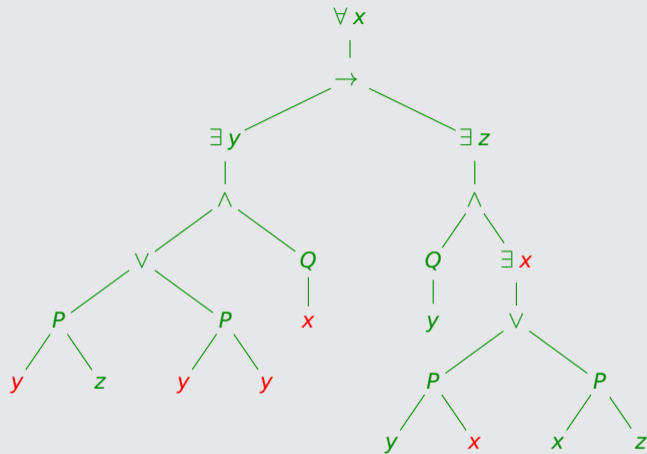
bound occurrences of variables

# Example



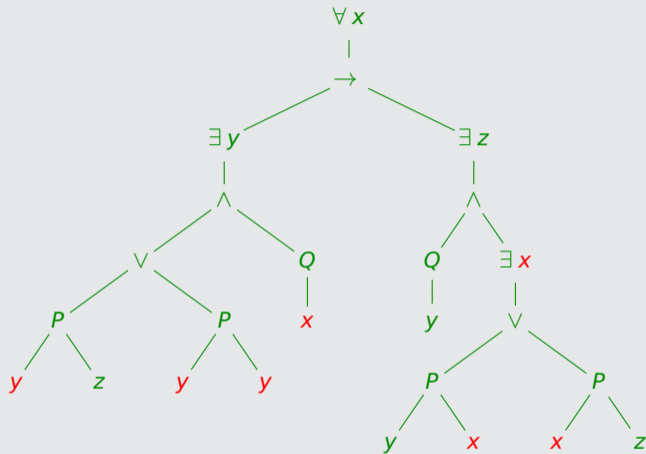
bound occurrences of variables

# Example



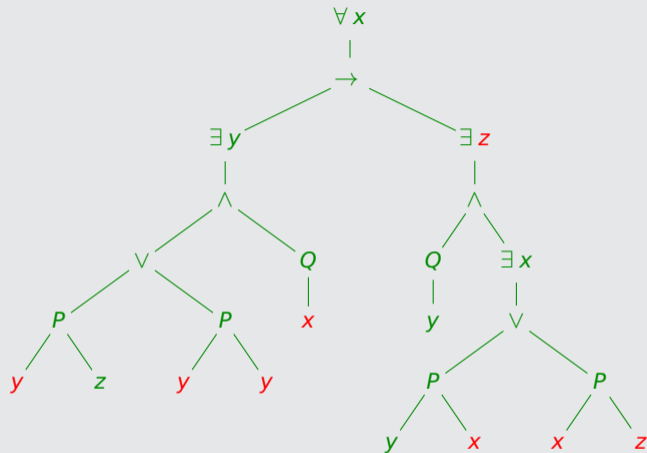
bound occurrences of variables

# Example



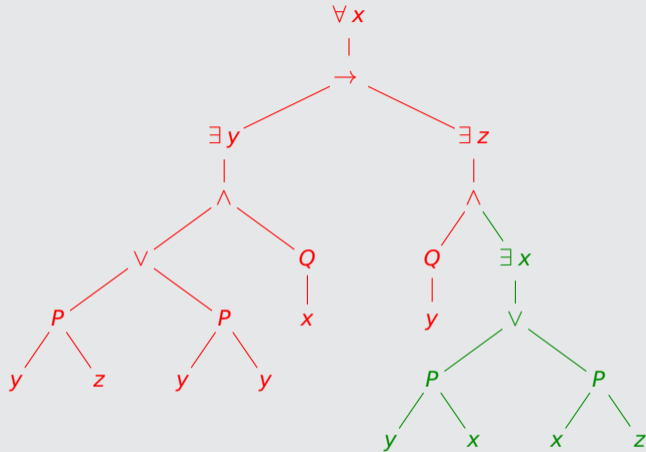
bound occurrences of variables

# Example



bound occurrences of variables

# Example



scope of  $\forall x$



# Outline

1. Summary of Previous Lecture
2. Algorithms for Binary Decision Diagrams
3. Intermezzo
4. Hidden Weighted Bit Function

## 5. Predicate Logic

Introduction

Syntax

Free and Bound Variables

Substitution

## 6. Further Reading

## Definition

$\varphi[t/x]$  is result of replacing all **free** occurrences of  $x$  in  $\varphi$  by  $t$

## Definition

$\varphi[t/x]$  is result of replacing all free occurrences of  $x$  in  $\varphi$  by  $t$

## Example

$$\varphi = \forall x (P(x) \wedge Q(y)) \rightarrow \neg P(x) \vee \exists y Q(y)$$

$$t = f(a, g(x))$$

## Definition

$\varphi[t/x]$  is result of replacing all free occurrences of  $x$  in  $\varphi$  by  $t$

## Example

$$\varphi = \forall x (P(x) \wedge Q(y)) \rightarrow \neg P(x) \vee \exists y Q(y)$$

$$t = f(a, g(x))$$

$$\varphi[t/x] = \forall x (P(x) \wedge Q(y)) \rightarrow \neg P(f(a, g(x))) \vee \exists y Q(y)$$

## Definition

$\varphi[t/x]$  is result of replacing all free occurrences of  $x$  in  $\varphi$  by  $t$

## Example

$$\varphi = \forall x (P(x) \wedge Q(y)) \rightarrow \neg P(x) \vee \exists y Q(y)$$

$$t = f(a, g(x))$$

$$\varphi[t/x] = \forall x (P(x) \wedge Q(y)) \rightarrow \neg P(f(a, g(x))) \vee \exists y Q(y)$$

$$\varphi[t/y] = \forall x (P(x) \wedge Q(f(a, g(x)))) \rightarrow \neg P(x) \vee \exists y Q(y)$$

## Definition

$\varphi[t/x]$  is result of replacing all free occurrences of  $x$  in  $\varphi$  by  $t$

## Example

$$\varphi = \forall x (P(x) \wedge Q(y)) \rightarrow \neg P(x) \vee \exists y Q(y)$$

$$t = f(a, g(x))$$

$$\varphi[t/x] = \forall x (P(x) \wedge Q(y)) \rightarrow \neg P(f(a, g(x))) \vee \exists y Q(y)$$

$$\varphi[t/y] = \forall x (P(x) \wedge Q(f(a, g(x)))) \rightarrow \neg P(x) \vee \exists y Q(y)$$

undesired effect:  $x$  is captured by  $\forall x$

## Definition

term  $t$  is **free for**  $x$  in  $\varphi$  if variables in  $t$  do not become bound in  $\varphi[t/x]$

## Definition

term  $t$  is **free for**  $x$  in  $\varphi$  if variables in  $t$  do not become bound in  $\varphi[t/x]$

## Example

$$\varphi = \forall x ((\forall z (P(z) \wedge Q(y))) \rightarrow \neg P(x) \vee Q(z))$$

$$t = f(y, z)$$



## Definition

term  $t$  is **free for**  $x$  in  $\varphi$  if variables in  $t$  do not become bound in  $\varphi[t/x]$

## Example

$$\varphi = \forall x ((\forall z (P(z) \wedge Q(y))) \rightarrow \neg P(x) \vee Q(z))$$

$$t = f(y, z)$$

►  $t$  is free for  $x$  in  $\varphi$

## Definition

term  $t$  is **free for**  $x$  in  $\varphi$  if variables in  $t$  do not become bound in  $\varphi[t/x]$

## Example

$$\varphi = \forall x ((\forall z (P(z) \wedge Q(y))) \rightarrow \neg P(x) \vee Q(z))$$

$$t = f(y, z)$$

- ▶  $t$  is free for  $x$  in  $\varphi$
- ▶  $t$  is not free for  $y$  in  $\varphi$

## Definition

term  $t$  is **free for**  $x$  in  $\varphi$  if variables in  $t$  do not become bound in  $\varphi[t/x]$

## Example

$$\varphi = \forall x ((\forall z (P(z) \wedge Q(y))) \rightarrow \neg P(x) \vee Q(z))$$

$$t = f(y, z)$$

- ▶  $t$  is free for  $x$  in  $\varphi$
- ▶  $t$  is not free for  $y$  in  $\varphi$
- ▶  $t$  is free for  $z$  in  $\varphi$

## Definition

term  $t$  is **free for**  $x$  in  $\varphi$  if variables in  $t$  do not become bound in  $\varphi[t/x]$

## Example

$$\varphi = \forall x ((\forall z (P(z) \wedge Q(y))) \rightarrow \neg P(x) \vee Q(z))$$

$$t = f(y, z)$$

- ▶  $t$  is free for  $x$  in  $\varphi$
- ▶  $t$  is not free for  $y$  in  $\varphi$
- ▶  $t$  is free for  $z$  in  $\varphi$

## Definition

**sentence** is formula without free variables

# Outline

1. Summary of Previous Lecture
2. Algorithms for Binary Decision Diagrams
3. Intermezzo
4. Hidden Weighted Bit Function
5. Predicate Logic
- 6. Further Reading**

- ▶ Section 2.1
- ▶ Section 2.2
- ▶ Section 6.2

## Extensions and Variants of OBDDs

- ▶ Algorithms and Data Structures in VLSI Design  
Christoph Meinel and Thorsten Theobald  
Springer-Verlag 1998  
[www.hpi.uni-potsdam.de/fileadmin/hpi/FG\\_ITS/books/OBDD-Book.pdf](http://www.hpi.uni-potsdam.de/fileadmin/hpi/FG_ITS/books/OBDD-Book.pdf)
- ▶ Zero-Suppressed BDDs and Their Applications  
Shin-ichi Minato  
International Journal on Software Tools for Technology Transfer 3, pp. 156–170, 2001  
doi: [10.1007/s100090100038](https://doi.org/10.1007/s100090100038)

## Important Concepts

- ▶ apply algorithm
- ▶ bound occurrence
- ▶ existential quantifier
- ▶ free BDD
- ▶ free occurrence
- ▶ function symbol
- ▶ hidden weighted bit function
- ▶ predicate symbol
- ▶ quantification
- ▶ quantifier
- ▶ reduce algorithm
- ▶ restrict algorithm
- ▶ restriction
- ▶ sentence
- ▶ scope
- ▶ Shannon expansion
- ▶ universal quantifier
- ▶ variable

## Important Concepts

- ▶ apply algorithm
- ▶ bound occurrence
- ▶ existential quantifier
- ▶ free BDD
- ▶ free occurrence
- ▶ function symbol
- ▶ hidden weighted bit function
- ▶ predicate symbol
- ▶ quantification
- ▶ quantifier
- ▶ reduce algorithm
- ▶ restrict algorithm
- ▶ restriction
- ▶ sentence
- ▶ scope
- ▶ Shannon expansion
- ▶ universal quantifier
- ▶ variable

homework for April 18