universität innsbruck

SS 2024 lecture 5

Logic

Diana Gründlinger Aart Middeldorp Fabian Mitterwallner

Alexander Montag Johannes Niederhauser Daniel Rainer

Particify ars.uibk.ac.at with session ID **0992 9580** for anonymous questions

Outline

Theorem

natural deduction is complete: $\varphi_1, \varphi_2, \ldots, \varphi_n \vDash \psi \implies \varphi_1, \varphi_2, \ldots, \varphi_n \vdash \psi$ is valid

Definitions

- clause is set of literals $\{\ell_1, \ldots, \ell_n\}$
- $\square$ denotes empty clause
- clausal form is set of clauses $\{C_1, \ldots, C_m\}$
- literals ℓ_1 and ℓ_2 are complementary if $\ell_1 = \ell_2^c = \begin{cases} \neg p & \text{if } \ell_2 = p \\ p & \text{if } \ell_2 = \neg p \end{cases}$
- clauses C_1 and C_2 clash on literal ℓ if $\ell \in C_1$ and $\ell^c \in C_2$
- resolvent of clashing clauses C_1 and C_2 on literal ℓ is clause $(C_1 \setminus \{\ell\}) \cup (C_2 \setminus \{\ell^c\})$

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

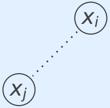
- resolution is terminating
- resolution is sound and complete: S admits refutation $\iff$ clausal form S is unsatisfiable

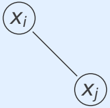
Remark

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 $[x_1, \ldots, x_n]$ of variables in B such that

$i < j$ for all edges 

and 

in B

- 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

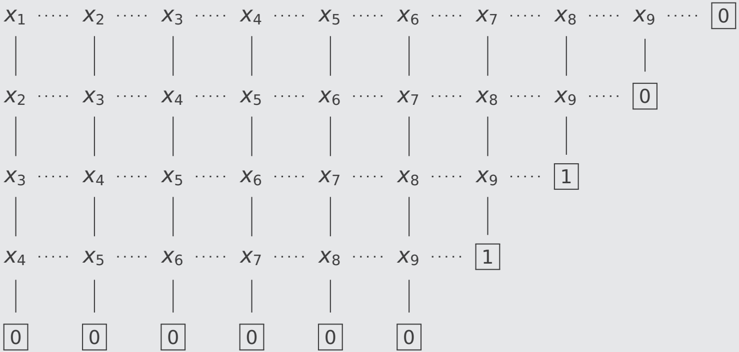
natural deduction, quantifier equivalences, resolution, semantics, Skolemization, syntax, undecidability, unification

Part III: Model Checking

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

Example (Cardinality Constraints using BDDs)

$$2 \leqslant x_1 + \cdots + x_9 \leqslant 3$$



Outline

Reduce Algorithm

input: • OBDD

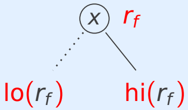
output: • equivalent reduced OBDD with compatible variable ordering

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
- 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 $\text{id}(n) = \text{id}(\text{lo}(n))$

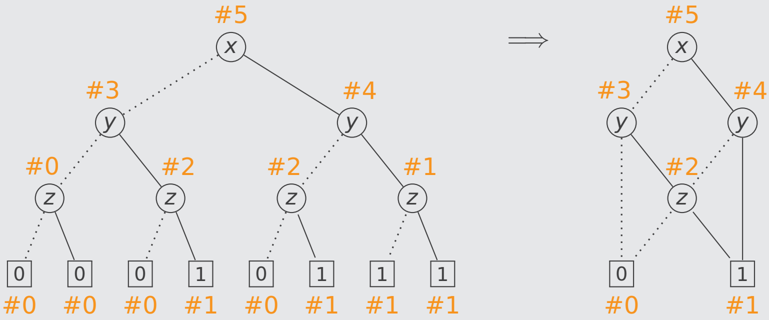
  ② if there exists node $m \neq n$ with same variable x and $\text{id}(m)$ defined such that

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

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

  ③ otherwise $\text{id}(n) =$ next unused natural number
- share nodes with same label

Example



Outline

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

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

- $f[0/x] = 0 \cdot (y + \overline{0}) = 0$
- $f[1/x] = 1 \cdot (y + \overline{1}) = y$
- $f[0/y] = x \cdot (0 + \overline{x}) = 0$
- $f[1/y] = x \cdot (1 + \overline{x}) = x$

Theorem (Shannon expansion)

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

Notational Convention

operator precedence $\cdot > \oplus, +$

Restrict Algorithm

input: • OBDD B_f, variable x, value $i \in \{0,1\}$

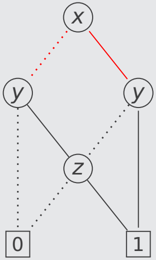
output: • reduced OBDD of $f[i/x]$ with compatible variable ordering

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

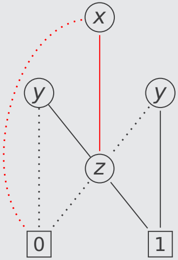
- $\text{lo}(n)$ if $i = 0$
- $\text{hi}(n)$ if $i = 1$

② reduce resulting OBDD

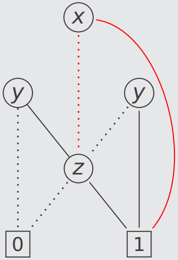
Example



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



$f[0/y]$



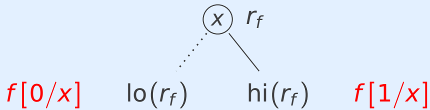
$f[1/y]$

inaccessible nodes are taken care of by garbage collector

Outline

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

$$\begin{aligned} f \star g &= \overline{x} \cdot (f \star g)[0/x] + x \cdot (f \star g)[1/x] \\ &= \overline{x} \cdot \underbrace{(f[0/x] \star g[0/x])} + x \cdot \underbrace{(f[1/x] \star g[1/x])} \end{aligned}$$

simpler than $f \star g$

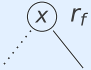
Apply Algorithm **apply**$(\star, B_f, B_g)$

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

return $\ell_f \star \ell_g$

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

return

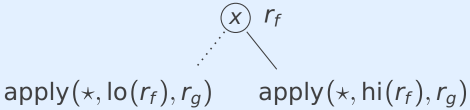


$\mathrm{apply}(\star, \mathrm{lo}(r_f), \mathrm{lo}(r_g))$ $\mathrm{apply}(\star, \mathrm{hi}(r_f), \mathrm{hi}(r_g))$

case III r_f non-terminal node with label x

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

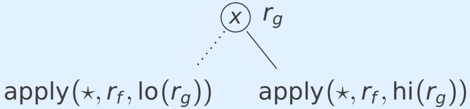
return



case IV r_g non-terminal node with label x

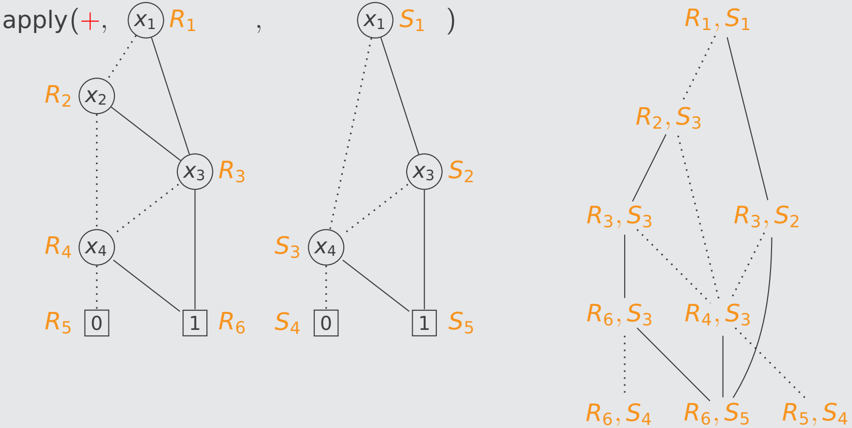
r_f terminal node or non-terminal node with label $y > x$

return



followed by application of reduce algorithm

Example



apply(+, ,)
x1 R1
R2 x2
x3 R3
R4 x4
R5 0
1 R6
x1 S1
x3 S2
S3 x4
S4 0
1 S5
R1, S1
R2, S3
R3, S3
R3, S2
R6, S3
R4, S3
R6, S4
R6, S5
R5, S4

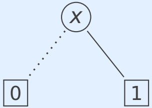
Outline

Definition

quantification of boolean function f over variable x:

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

Summary

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

Demo

BoolTool

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

BoolTool Reloaded

by Martin Neuner (2023)

Outline

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

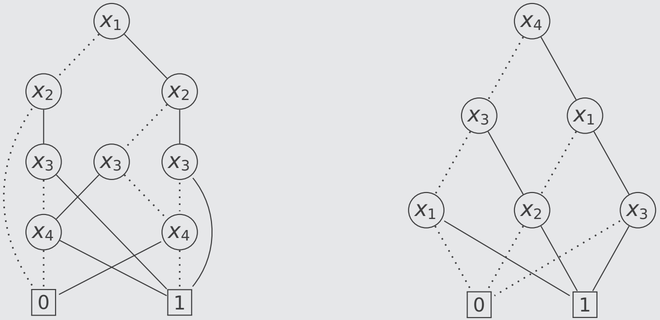
Definitions

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

Example

x_1 x_2 x_3 x_4	HWB_4	x_1 x_2 x_3 x_4	HWB_4	x_1 x_2 x_3 x_4	HWB_4	x_1 x_2 x_3 x_4	HWB_4
0 0 0 0	0	0 1 0 0	0	1 0 0 0	1	1 1 0 0	1
0 0 0 1	0	0 1 0 1	1	1 0 0 1	0	1 1 0 1	0
0 0 1 0	0	0 1 1 0	1	1 0 1 0	0	1 1 1 0	1
0 0 1 1	0	0 1 1 1	1	1 0 1 1	1	1 1 1 1	1

Example



reduced OBDD

free (read-1) BDD

Theorem

- every reduced OBDD computing HWB_n has size exponential in n
- some reduced BDD computing HWB_n has size quadratic in n

Outline

Definition

propositional formulas are built from

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

according to following Backus–Naur Form:

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

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
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

$A(x, y)$	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

Definitions

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

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

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

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

- notational conventions:
  - binding precedence $\quad = \quad > \quad \neg, \forall, \exists \quad > \quad \wedge, \vee \quad > \quad \rightarrow$
  - omit outer parentheses
  - $\rightarrow$, $\wedge$, $\vee$ are right-associative

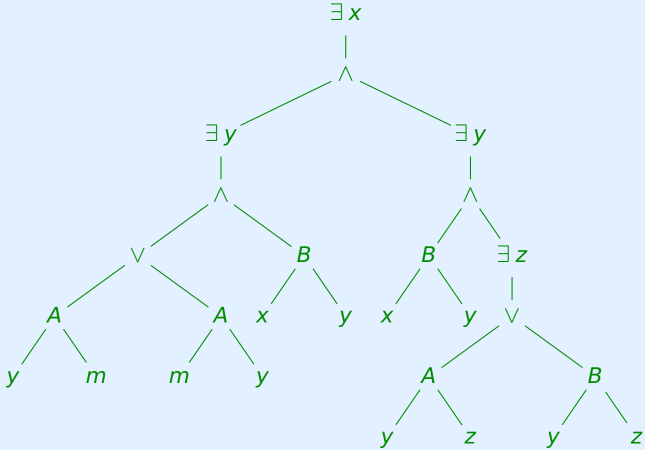
Example (Exercise 2.1.1, cont'd)

$A(x,y)$	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 $\forall x\,(P(x) \to A(m,x))$
- some professor admires Mary $\exists x\,(P(x) \wedge A(x,m))$
- Mary admires herself $A(m,m)$
- no student attended every lecture $\neg\exists x\,(S(x) \wedge \forall y\,(L(y) \to B(x,y)))$
- no lecture was attended by every student $\neg\exists x\,(L(x) \wedge \forall y\,(S(y) \to B(y,x)))$
- no lecture was attended by any student $\forall x\,\forall y\,(L(x) \wedge S(y) \to \neg B(y,x))$

Parse Tree

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

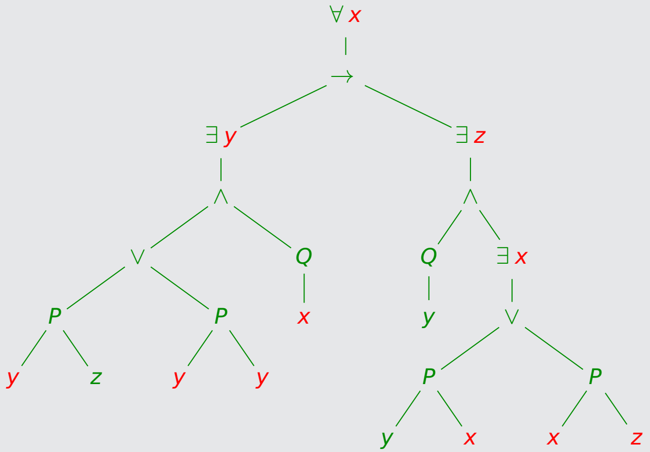


Outline

Definitions

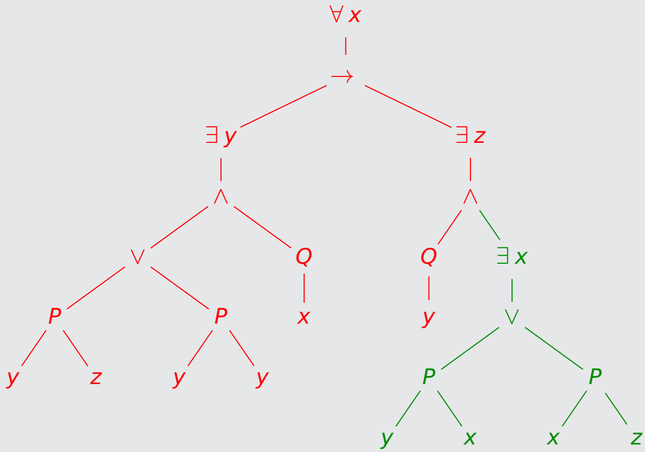
- occurrence of variable x in formula φ is free in φ if it is leaf node in parse tree of φ such that there is no node $\forall x$ or $\exists x$ on path to root node
- occurrence of variable x in formula φ is bound if this occurrence is not free in φ
- scope of occurrence of $\forall x$ $(\exists x)$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

Example



bound occurrences of variables

Example



scope of $\forall x$

Outline

Definition

$\varphi[t/x]$ is result of replacing all free occurrences of x in φ 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 φ if variables in t do not become bound in $\varphi[t/x]$

Example

$$\begin{aligned} \varphi &= \forall x\, ((\forall z\, (P(z) \wedge Q(y))) \rightarrow \neg P(x) \vee Q(z)) \\ t &= f(y,z) \end{aligned}$$

- t is free for x in φ
- t is not free for y in φ
- t is free for z in φ

Definition

sentence is formula without free variables

Outline

Huth and Ryan

- 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
- 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

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