Semantics and Verification 2005

Lecture 13

- boolean expressions and normal forms
- binary decision diagrams (BDDs)
- algorithms on BDDs

Boolean Functions

Let $\mathbb{B} = \{0, 1\}$. 1 ... true, 0 ... false

Boolean Function of Arity n

$$f : \mathbb{B}^n \to \mathbb{B}$$

Boolean functions are often described using truth tables.

x_1	x_2	x_3	$f(x_1, x_2, x_3)$
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

Problem: arity n gives truth table of size $\theta(2^n)$.

Boolean Expressions

Let $x_1, x_2, \ldots, x_n$ be boolean variables.

Abstract Syntax for Boolean Expressions (x ranges over variables)

$$t, t_1, t_2 ::= 0 \mid 1 \mid x \mid \neg t \mid t_1 \wedge t_2 \mid t_1 \vee t_2 \mid t_1 \Rightarrow t_2 \mid t_1 \Leftrightarrow t_2$$

Truth Assignment

$$v : \{x_1, \ldots, x_n\} \to \mathbb{B}$$

Function v is often written as $[v(x_1)/x_1, v(x_2)/x_2, \ldots, v(x_n)/x_n]$.

Example:

- boolean expression: $\neg(x_1 \wedge x_2) \Rightarrow (\neg x_1 \vee x_4)$
- truth assignment: $[1/x_1, 0/x_2, 1/x_3, 1/x_4]$

Evaluation of Boolean Expressions

A boolean expression t defines a boolean function $f^t : \mathbb{B}^n \to \mathbb{B}$ by the following (structural) rules:

t	$\neg t$
0	1
1	0

t_1	t_2	$t_1 \wedge t_2$
0	0	0
0	1	0
1	0	0
1	1	1

t_1	t_2	$t_1 \vee t_2$
0	0	0
0	1	1
1	0	1
1	1	1

t_1	t_2	$t_1 \Rightarrow t_2$
0	0	1
0	1	1
1	0	0
1	1	1

t_1	t_2	$t_1 \Leftrightarrow t_2$
0	0	1
0	1	0
1	0	0
1	1	1

Terminology

Equivalent Boolean Expressions

Boolean expressions t_1 and t_2 are equivalent iff $f^{t_1} = f^{t_2}$, i.e., they yield the same truth value for all truth assignments.

Example: $\neg(x_1 \wedge x_2)$ is equivalent to $\neg x_1 \vee \neg x_2$

Tautology

A boolean expression t is a tautology if it yields true for all truth assignment.

Satisfiability

A boolean expression t is satisfiable if it yields true for at least one truth assignment.

Conjunctive Normal Form (CNF)

Definitions

- Literal is a boolean variable or its negation.
- Clause is a disjunction of literals.
- A boolean expression if CNF is a conjunction of clauses.

Example: $(x_1 \vee \neg x_3) \wedge (\neg x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee \neg x_3)$

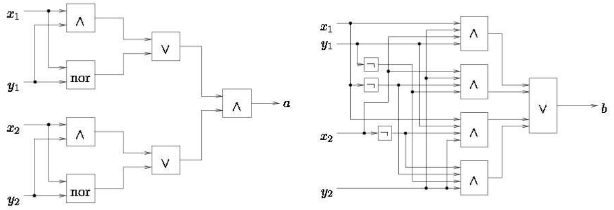
Theorem

For any boolean expression there is an equivalent one in CNF.

Cook's Theorem

Satisfiability of boolean expressions (in CNF) is NP-complete.

Combinatorial Circuits



Are these two circuits equivalent?

co-NP-hard problem!

Representations of Boolean Functions

Problems over Boolean Expressions are Hard

Many problems related to boolean expressions are hard from the theoretical point of view (NP-complete or co-NP-complete).

Our Aim

We are looking for

- compact representation and
- efficient manipulation

with boolean expressions for real-life examples.

We will have a look at Binary Decision Diagrams (BDDs) [Randal E. Bryant'86].

If-Then-Else Operator

Let t, t_1 and t_2 be boolean expressions.

Syntax

$$t \rightarrow t_1, t_2$$

Semantics

If-Then-Else operator $t \rightarrow t_1, t_2$ is equivalent to $(t \wedge t_1) \vee (\neg t \wedge t_2)$.

t	t_1	t_2	$t \rightarrow t_l, t_2$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

If-Then-Else Normal Form

Definition

A boolean expression is in If-Then-Else normal form (INF) iff it is given by the following abstract syntax

$$t, t_1, t_2 ::= 0 \mid 1 \mid x \rightarrow t_1, t_2$$

where x ranges over boolean variables.

Example: $x_1 \rightarrow (x_2 \rightarrow 1, 0), 0$ (equivalent to $x_1 \wedge x_2$)

Boolean expressions in INF can be drawn as decision trees.

Shannon's Expansion Law

Let t be a boolean expression and x a variable. We define boolean expressions

- $t[0/x]$ where every occurrence of x in t is replaced with 0, and
- $t[1/x]$ where every occurrence of x in t is replaced with 1.

Shannon's Expansion Law

Let x be an arbitrary boolean variable. Any boolean expressions t is equivalent to

$$x \rightarrow t[1/x], t[0/x].$$

Corollary

For any boolean expression there is an equivalent one in INF.

Binary Decision Diagrams

Let the set of boolean variables be $\{x_1, \ldots, x_n\}$.

Binary Decision Diagram (BDD)

A BDD is a rooted, directed, acyclic graph (V, E) such that

- $0, 1 \in V$ (representing false and true) and the nodes 0 and 1 have no outgoing edges
- every node $v \in V \setminus \{0, 1\}$ has exactly two successors $low(v) \in V$ and $high(v) \in V$
- every node $v \in V \setminus \{0, 1\}$ has a label $var(v) \in \{x_1, \ldots, x_n\}$.

Assume a given total ordering $<$ on boolean variables.

Ordered BDD

A BDD is ordered if on all paths from the root the variables respect the ordering $<$.

Reduced Ordered BDDs (ROBDDs)

Reduced BDD

A BDD is reduced iff for all nodes $u, v \in V \setminus \{0, 1\}$:

1. $low(u) \neq high(u)$
2. $low(u) = low(v)$ and $high(u) = high(v)$ and $var(u) = var(v)$ implies that $u = v$.

ROBDD with a root node u describes a boolean expression t^u according to the following (inductive) definition:

- $t^0 \stackrel{\text{def}}{=} 0$
- $t^1 \stackrel{\text{def}}{=} 1$
- $t^u \stackrel{\text{def}}{=} var(u) \to t^{high(u)}, t^{low(u)}$

Canonicity of ROBDDs

Canonicity Lemma

For any boolean function $f : \mathbb{B}^n \to \mathbb{B}$ and a given ordering of variables $x_1 < x_2 < \cdots < x_n$ there is exactly one ROBDD with root u which describes the function f, i,e.

$$t^u[v_1/x_1, \ldots, v_n/x_n] = f(v_1, \ldots, v_n)$$

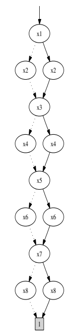
for all $(v_1, \ldots, v_n) \in \mathbb{B}^n$.

Consequences:

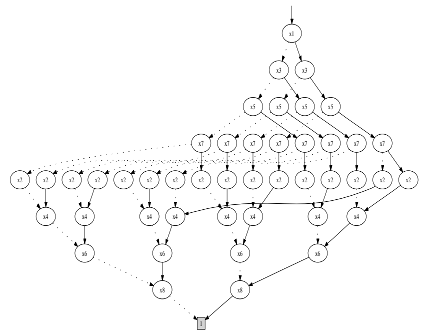
- A given ROBDD with root u is tautology iff $u = 1$.
- A given ROBDD with root u is satisfiable iff $u \neq 0$.

Ordering of Variables (Exponential Difference in Size)

$$(x_1 \Leftrightarrow x_2) \wedge (x_3 \Leftrightarrow x_4) \wedge (x_5 \Leftrightarrow x_6) \wedge (x_7 \Leftrightarrow x_8)$$



$x_1 < x_2 < \cdots < x_8$



$x_1 < x_3 < x_5 < x_7 < x_2 < x_4 < x_6 < x_8$

Representing ROBDDs in Memory – Array Implementation

Assume $x_1 < x_2 < x_3$.

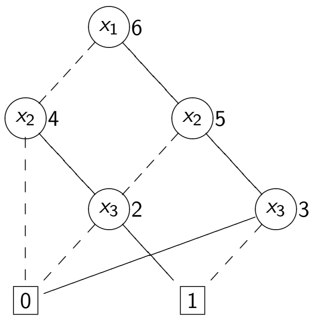


Table T :
$u \mapsto (var(u), low(u), high(u))$

u	var	low	high
0	4	-	-
1	4	-	-
2	3	0	1
3	3	1	0
4	2	0	2
5	2	2	3
6	1	4	5

Inverse table H :
$(var, low, high) \mapsto u$.

Example: $T(4) = (2, 0, 2)$, $H(1, 4, 5) = 6$, and $H(3, 0, 2) = undef$.

Makenode and Reducedness of BDDs

$T : u \mapsto (var(u), low(u), high(u))$ $\quad$ $H : (var, low, high) \mapsto u$

Makenode $(var, low, high)$: Node =
if $low = high$ **then**
  return low
else
  $u := H(var, low, high)$
  if $u \neq undef$ **then**
    return u
  else
    add a new node (row) to T with attributes $(var, low, high)$
    return $H(var, low, high)$
  end if
end if

Building an ROBDD from a Boolean Expression

Let t be a boolean expression and $x_1 < x_2 < \cdots < x_n$.
Build$(t, 1)$ builds a corresponding ROBDD and returns its root.

Build(t, i): Node =
if $i > n$ **then**
  if t is true **then return** 0 **else return** 1
else
  low := Build$(t[0/x_i], i + 1)$
  $high$:= Build$(t[1/x_i], i + 1)$
  var := i
  return Makenode$(var, low, high)$
end if

Complexity: exponentially many recursive calls!
Is this necessary? Yes, checking if t is a tautology is co-NP-hard!

Boolean Operations on ROBDDs

Let us assume that ROBDDs for boolean expressions t_1 and t_2 are already constructed.

How to construct ROBDD for

- $\neg t_1$
- $t_1 \wedge t_2$
- $t_1 \vee t_2$
- $t_1 \Rightarrow t_2$
- $t_1 \Leftrightarrow t_2$

with an emphasis on efficiency?

Idea (assume $x_1 < x_2 < \cdots < x_n$)

- $x_i = x_i$

$$(x_i \to t_1, t_2) \wedge (x_i \to t_1', t_2')$$
$$\equiv$$
$$x_i \to (t_1 \wedge t_1'), (t_2 \wedge t_2')$$

- $x_i < x_j$

$$(x_i \to t_1, t_2) \wedge (x_j \to t_1', t_2')$$
$$\equiv$$
$$x_i \to (t_1 \wedge (x_j \to t_1', t_2')), (t_2 \wedge (x_j \to t_1', t_2'))$$

The same equivalences hold also for $\vee$, $\Rightarrow$ and $\Leftrightarrow$.

Apply (for $op \in \{\wedge, \vee, \Rightarrow, \Leftrightarrow\}$)

Apply $(u_1, u_2$: Node): Node $=$
if $(u_1 \in \{0,1\}$ and $u_2 \in \{0,1\})$ **then**
  $u := u_1\ op\ u_2$
else if $var(u_1) = var(u_2)$ **then**
  $\ell :=$ Apply$(low(u_1), low(u_2))$; $h :=$ Apply$(high(u_1), high(u_2))$
  $u :=$ Makenode$(var(u_1), \ell, h)$
else if $var(u_1) < var(u_2)$ **then**
  $\ell :=$ Apply$(low(u_1), u_2)$; $h :=$ Apply$(high(u_1), u_2)$
  $u :=$ Makenode$(var(u_1), \ell, h)$
else if $var(u_1) > var(u_2)$ **then**
  $\ell :=$ Apply$(u_1, low(u_2))$; $h :=$ Apply$(u_1, high(u_2))$
  $u :=$ Makenode$(var(u_2), \ell, h)$
end if
return u

Problem: Exponentially many recursive calls!

Apply with Dynamic Programming in $O(|u_1| \cdot |u_2|)$

Two dimensional array $G(\_,\_)$ initially empty.

Apply $(u_1, u_2$: Node): Node $=$
if $G(u_1, u_2) \neq$ empty **then**
  return $G(u_1, u_2)$
else if $(u_1 \in \{0,1\}$ and $u_2 \in \{0,1\})$ **then**
  $u := u_1\ op\ u_2$
else if $var(u_1) = var(u_2)$ **then**
  $u := \ldots$
else if $var(u_1) < var(u_2)$ **then**
  $u := \ldots$
else if $var(u_1) > var(u_2)$ **then**
  $u := \ldots$
end if
$G(u_1, u_2) := u$
return u

Other Operations on ROBDDs

Let t be a boolean expression with its ROBDD representation.

The following operations can be done efficiently:

- Restriction $t[0/x_i]$ $(t[1/x_i])$: restricts the variable x_i to 0 (1)
- SatCount(t): returns the number of satisfying assignments
- AnySat(t): returns some satisfying assignment
- AllSat(t): returns all satisfying assignments
- Existential quantification $\exists x_i.t$: equivalent to $t[0/x_i] \vee t[1/x_i]$
- Composition $t[t'/x_i]$: equivalent to $t' \to t[1/x_i], t[0/x_i]$

Use of ROBDDs

- Combinatorial circuits.
- Combinatorial problems.
- Verification (equivalence checking, temporal logic model checking).
- Program analysis.
- ...