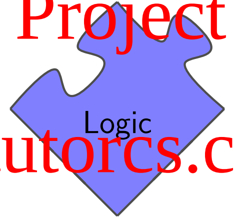
COMP9020

Foundations of Computer Science

Lecture 12: Boolean Logic



		[LLM]	[RW]	[Rosen]
Week 8	Boolean Logic	Ch. 3	Ch. 2, 10	Ch. 12
Week 8	Propositional Logic	Ch. 3	Ch. 2	Ch. 1

What is logic?

Logic is about **formalizing reasoning** and **defining truth**

- Adding rigour
- Removing ambiguity
- Mechanizing the process of reasoning

Loose history of logic

- (Ancient times): Logic exlusive to philosophy
- Mid-19th Century: Logical foundations of Mathematics (Boole, Jevons, Schröder, etc)
- 1910: Russell and Whitehead's Principia Mathematica
- 1928: Hilbert proposes *Entscheidungsproblem*
- 1931: Gödel's Incompleteness Theorem
- 1935: Church's Lambda calculus
- 1936: Turing's Machine-based approach
- 1930s: Shannon develops Circuit logic
- 1960s: Formal verification; Relational databases

Computation = Calculation + Symbolic manipulation

Logic in Computer Science

Computation = Calculation + Symbolic manipulation

Logic as 2-valued computation (Boolean logic):

- Circuit design
- Code optimization
- Boolean algebra
- Nand game

$$\text{Computation} \quad = \quad \text{Calculation} + \text{Symbolic manipulation}$$

Logic as symbolic reasoning (Propositional logic, and beyond):

- Formal verification
- Proof assistance
- Knowledge Representation and Reasoning
- Automated reasoning
- Databases

Outline

Boolean Logic

Boolean Functions

Conjunctive and Disjunctive Normal Form

Karnaugh Maps

Boolean Algebras

Outline

Boolean Logic

Boolean Functions

Conjunctive and Disjunctive Normal Form

Karnaugh Maps

Boolean Algebras

Boolean logic is about performing calculations in a "simple" mathematical structure.

- complex calculations can be built entirely from these simple ones
- can help identify simplifications that improve performance at the circuit level
- can help identify simplifications that improve presentation at the programming level

The Boolean Algebra $\mathbb{B}$

Definition

The (two-element) **Boolean algebra** is defined to be the set $\mathbb{B} = \{0, 1\}$, together with the functions $! : \mathbb{B} \to \mathbb{B}$, $\&\&\colon \mathbb{B}^2 \to \mathbb{B}$, and $\| \colon \mathbb{B}^2 \to \mathbb{B}$, defined as follows:

$$!x = (1 - x) \qquad x \;\&\&\; y = \min\{x, y\} \qquad x \parallel y = \max\{x, y\}$$

The Boolean Algebra $\mathbb{B}$ – Alternative definition

Definition

The (two-element) **Boolean algebra** is defined to be the set $\mathbb{B} = \{\text{false}, \text{true}\}$, together with the functions $! : \mathbb{B} \to \mathbb{B}$, $\&\& : \mathbb{B}^2 \to \mathbb{B}$, and $\| : \mathbb{B}^2 \to \mathbb{B}$, defined as follows:

x	$!x$
false	true
true	false

x	y	$x \&\& y$
false	false	false
false	true	false
true	false	false
true	true	true

x	y	$x \| y$
false	false	false
false	true	true
true	false	true
true	true	true

Alternative notation

Commonly, the following alternative notation is used:

For $\mathbb{B}$:	$\{F, T\}$
For $!x$:	$\overline{x}$, x', $\sim x$, $\neg x$
For x && y:	xy, $x \wedge y$
For $x \parallel y$:	$x + y$, $x \vee y$

Properties

We observe that !, &&, and $\|$ satisfy the following:

For all $x, y, z \in \mathbb{B}$:

Commutativity	$x \| y = y \| x$
	$x \&\& y = y \&\& x$
Associativity	$(x \| y) \| z = x \| (y \| z)$
	$(x \&\& y) \&\& z = x \&\& (y \&\& z)$
Distribution	$x \| (y \&\& z) = (x \| y) \&\& (x \| z)$
	$x \&\& (y \| z) = (x \&\& y) \| (x \&\& z)$
Identity	$x \| 0 = x$
	$x \&\& 1 = x$
Complementation	$x \| (!x) = 1$
	$x \&\& (!x) = 0$

Examples

- Calculate $x \mathbin{\&\&} x$ for all $x \in \mathbb{B}$
- Calculate $((1 \mathbin{\&\&} 0) \parallel ((!1) \mathbin{\&\&} (!0)))$

Outline

Boolean Logic

Boolean Functions

Conjunctive and Disjunctive Normal Form

Karnaugh Maps

Boolean Algebras

Definition

An n-**ary Boolean function** is a map $f : \mathbb{B}^n \to \mathbb{B}$.

Question

How many unary Boolean functions are there?

How many binary functions?

n-ary?

Examples

Examples

- ! is a unary Boolean function
- &&, || are binary Boolean functions
- $f(x, y) = !(x \,\&\&\, y)$ is a binary Boolean function (NAND)
- $\textsc{And}(x_0, x_1, \ldots) = (\cdots((x_0 \,\&\&\, x_1) \,\&\&\, x_2)\cdots)$ is a (family) of Boolean functions
- $\textsc{Or}(x_0, x_1, \ldots) = (\cdots((x_0 \,\|\, x_1) \,\|\, x_2)\cdots)$ is a (family) of Boolean functions

Application: Adding two one-bit numbers

How can we implement:

$$\mathrm{add} : \mathbb{B}^2 \to \mathbb{B}^2$$

defined as

x	y	$\mathrm{add}(x, y)$
0	0	00
0	1	01
1	0	01
1	1	10

Use two Boolean functions!

NB

Digital circuits are just sequences of Boolean functions.

Outline

Boolean Logic

Boolean Functions

Conjunctive and Disjunctive Normal Form

Karnaugh Maps

Boolean Algebras

Conjunctive and Disjunctive normal form

Definition

- A **literal** is a unary Boolean function.
- A **minterm** is a Boolean function of the form $\mathrm{AND}(l_1(x_1), l_2(x_2), \ldots, l_n(x_n))$ where the l_i are literals
- A **maxterm** is a Boolean function of the form $\mathrm{OR}(l_1(x_1), l_2(x_2), \ldots, l_n(x_n))$ where the l_i are literals
- A **CNF Boolean function** is a function of the form $\mathrm{AND}(m_1, m_2, \ldots)$ where the m_i are maxterms.
- A **DNF Boolean function** is a function of the form $\mathrm{OR}(m_1, m_2, \ldots)$, where the m_i are minterms.

Examples

Examples

- $f(x, y, z) = (x \;\&\&\; (!y) \;\&\&\; z) \,\|\, (x \;\&\&\; (!y) \;\&\&\; (!z)) = x\,\overline{y}\,z + x\,\overline{y}\,\overline{z}$: DNF , but not CNF
- $g(x, y, z) = (x \,\|\, (!y) \,\|\, z) \;\&\&\; (x \,\|\, (!y) \,\|\, (!z)) = (x + \overline{y} + z)(x + \overline{y} + \overline{z})$: CNF function but not DNF
- $h(x, y, z) = (x \;\&\&\; (!y) \;\&\&\; z) = x\,\overline{y}\,z$: both CNF and DNF
- $j(x, y, z) = x + y(z + x)$: Neither CNF nor DNF

NB

CNF: product of sums; DNF: sum of products

Theorem

Every Boolean function can be written as a function in DNF/CNF

Proof...

Canonical DNF

Given an n-ary boolean function $f : \mathbb{B}^n \to \mathbb{B}$ we construct an equivalent DNF boolean function as follows:
For each $\mathbf{b} = (b_1, \ldots, b_n) \in \mathbb{B}^n$ we define the minterm

$$m_{\mathbf{b}} = \textsc{And}(l_1(x_1), l_2(x_2), \ldots, l_n(x_n))$$

where

$$l_i(x_i) = \begin{cases} x_i & \text{if } b_i = 1 \\ !x_i & \text{if } b_i = 0 \end{cases}$$

We then define the DNF formula:

$$f_{\mathrm{DNF}} = \sum_{f(\mathbf{b})=1} m_{\mathbf{b}},$$

that is, f_{DNF} is the disjunction (or) over all minterms corresponding to elements $\mathbf{b} \in \mathbb{B}$ where $f(\mathbf{b}) = 1$.

Theorem

f and f_{DNF} are the same function.

Exercises

$\boxed{\text{RW: 10.2.3}}$ Find the canonical DNF form of each of the following expressions in variables x, y, z

- xy
- $\overline{z}$
- $xy + \overline{z}$
- $f(x, y, z) = 1$

Outline

Boolean Logic

Boolean Functions

Conjunctive and Disjunctive Normal Form

Karnaugh Maps

Boolean Algebras

Karnaugh Maps

For up to four variables (propositional symbols) a diagrammatic method of simplification called **Karnaugh maps** works quite well. For every propositional function of $k = 2, 3, 4$ variables we construct a rectangular array of 2^k cells. We mark the squares corresponding to the value `true` with eg "+" and try to cover these squares with as few rectangles with sides 1 or 2 or 4 as possible.

Example

	yz	$y\bar{z}$	$\bar{y}\bar{z}$	$\bar{y}z$
x	+	+		+
$\bar{x}$	+		+	+

For optimisation, the idea is to cover the $+$ squares with the minimum number of rectangles. One *cannot* cover any empty cells.

- The rectangles can go 'around the corner'/the actual map should be seen as a *torus*.
- Rectangles must have sides of 1, 2 or 4 squares (three adjacent cells are useless).

Example

	yz	$y\bar{z}$	$\bar{y}\bar{z}$	$\bar{y}z$
x	$+$	$+$		$+$
$\bar{x}$	$+$		$+$	$+$

$$E = (xy) \vee (\bar{x}\bar{y}) \vee z$$

Canonical form would consist of writing all cells separately (6 clauses).

Exercise

Exercise

RW: 10.6.6(c)

	yz	$y\bar{z}$	$\bar{y}\bar{z}$	$\bar{y}z$
wx	+	+		+
$w\bar{x}$	+	+	+	+
$\bar{w}\bar{x}$			+	+
$\bar{w}x$	+			+

Outline

Boolean Logic

Boolean Functions

Conjunctive and Disjunctive Normal Form

Karnaugh Maps

Boolean Algebras

Definition: Boolean Algebra

Definition

A **Boolean algebra** is a structure $(T, \vee, \wedge, ', \mathbb{0}, \mathbb{1})$ where

- $\mathbb{0}, \mathbb{1} \in T$
- $\vee, \wedge : T \times T \to T$ (called **join** and **meet** respectively)
- $' : T \to T$ (called **complementation**)

and the following laws hold for all $x, y, z \in T$:

Commutativity:	$x \vee y = y \vee x, \quad x \wedge y = y \wedge x$
Associativity:	$(x \vee y) \vee z = x \vee (y \vee z)$ $(x \wedge y) \wedge z = x \wedge (y \wedge z)$
Distributivity:	$x \vee (y \wedge z) = (x \vee y) \wedge (x \vee z)$ $x \wedge (y \vee z) = (x \wedge y) \vee (x \wedge z)$
Identity:	$x \vee \mathbb{0} = x, \quad x \wedge \mathbb{1} = x$
Complementation:	$x \vee x' = \mathbb{1}, \quad x \wedge x' = \mathbb{0}$

Examples of Boolean Algebras

Example

The set of subsets of a set X:

$$\begin{aligned} T &: \quad \mathsf{Pow}(X) \\ \vee \text{ (join)} &: \quad \cup \\ \wedge \text{ (meet)} &: \quad \cap \\ {}' \text{ (complement)} &: \quad \cdot^c \\ \mathbb{0} &: \quad \emptyset \\ \mathbb{1} &: \quad X \quad (\mathcal{U}) \end{aligned}$$

The Laws of Boolean algebra follow from the Laws of Set Operations.

Example

The two element Boolean Algebra :

$$\mathbb{B} = (\{\text{true}, \text{false}\}, \|, \&\&, !, \text{false}, \text{true})$$

where $!, \&\&, \|$ are defined as:

- $!\text{true} = \text{false}$; $!\text{false} = \text{true}$,
- $\text{true}\ \&\&\ \text{true} = \text{true}$; ...
- $\text{true} \| \text{true} = \text{true}$; ...

Examples of Boolean Algebras

Example

Cartesian products of $\mathbb{B}$, that is n-tuples of 0's and 1's with Boolean operations, e.g. $\mathbb{B}^4$:

$$
\begin{aligned}
\textit{join:} \quad & (1,0,0,1) \vee (1,1,0,0) = (1,1,0,1) \\
\textit{meet:} \quad & (1,0,0,1) \wedge (1,1,0,0) = (1,0,0,0) \\
\textit{complement:} \quad & (1,0,0,1)' = (0,1,1,0) \\
\mathbb{0}: \quad & (0,0,0,0) \\
\mathbb{1}: \quad & (1,1,1,1).
\end{aligned}
$$

Examples of Boolean Algebras

Example

Functions from any set S to $\mathbb{B}$: that is, $\mathbb{B}^S$

If $f, g : S \longrightarrow \mathbb{B}$ then

$$
\begin{array}{rll}
(f \vee g) : S \to \mathbb{B} & \text{defined by} & s \mapsto f(s) \parallel g(s) \\
(f \wedge g) : S \to \mathbb{B} & \text{defined by} & s \mapsto f(s) \,\&\&\, g(s) \\
f' : S \to \mathbb{B} & \text{defined by} & s \mapsto !f(s) \\
\mathbb{0} : S \to \mathbb{B} & \text{is the function} & s \mapsto 0 \\
\mathbb{1} : S \to \mathbb{B} & \text{is the function} & s \mapsto 1
\end{array}
$$

Show an identity holds using the laws of Boolean Algebra, then that identity holds **in all Boolean Algebras**.

Example

In all Boolean Algebras

$$x \wedge x = x$$

for all $x \in T$.

Proof:

$$\begin{aligned}
x &= x \wedge \mathbb{1} && \text{[Identity]} \\
&= x \wedge (x \vee x') && \text{[Complement]} \\
&= (x \wedge x) \vee (x \wedge x') && \text{[Distributivity]} \\
&= (x \wedge x) \vee \mathbb{0} && \text{[Complement]} \\
&= (x \wedge x) && \text{[Identity]}
\end{aligned}$$

Duality

Definition

If E is an expression defined using variables (x, y, z, etc), constants (0 and 1), and the operations of Boolean Algebra ($\wedge$, $\vee$, and $'$) then dual(E) is the expression obtained by replacing $\wedge$ with $\vee$ (and vice-versa) and 0 with 1 (and vice-versa).

Definition

If $(T, \vee, \wedge, ', \mathbb{0}, \mathbb{1})$ is a Boolean Algebra, then $(T, \wedge, \vee, ', \mathbb{1}, \mathbb{0})$ is also a Boolean algebra, known as the **dual** Boolean algebra.

Theorem (Principle of duality)

If you can show $E_1 = E_2$ using the laws of Boolean Algebra, then dual(E_1) = dual(E_2).

Example

We have shown $x \wedge x = x$.

By duality: $x \vee x = x$.