Boolean Algebra

Lecturer: Guillaume Beslon
(Lecture adapted from Lionel Morel)

Computer Science and Information Technologies - INSA Lyon

Fall 2023

Interpretation of bits as Boolean values

Two elementary values:

- `0` $\Rightarrow$ "false"
- `1` $\Rightarrow$ "true"

From these values, we will (1) use Boolean algebra to build expressions that transform bit vectors into other bit vectors (i.e. an information into another information) and (2) implement them as logical circuits.

$\rightarrow$ Two types of expressions/circuits:

- **Combinatorial expressions/circuits**: output bits only depend on values **currently** available on input bits.
- **Sequential expressions/circuits**: The circuit is considered to have a **state** (i.e., a memory of its past activity/inputs). Output bits depend both on values currently available on input bits **+ circuit state**.

Boolean functions

Boole Algebra is equipped with three operations

- a unary operation, **negation**, noted `NOT`;
- two binary commutative, associative operations:
  - **conjunction** — `AND`, with 1 as neutral element;
  - **disjunction** — `OR`, with 0 as neutral element;
- `AND` is distributive over `OR` *and* `OR` is distributive over `AND` (!!! Boolean algebra is *NOT* elementary algebra !!!)

If a and b are 2 Boolean variables, we write:

$$\mathrm{NOT}(a) = \overline{a}, \quad \mathrm{AND}(a, b) = ab = a.b, \quad \mathrm{OR}(a, b) = a + b$$

or

$$\mathrm{NOT}(a) = \neg a, \quad \mathrm{AND}(a, b) = a \wedge b, \quad \mathrm{OR}(a, b) = a \vee b$$

Boolean Cheat Sheet[1]

- absorbing elements: $a + 1 = 1, \quad a \cdot 0 = 0$
- neutral elements: $a + 0 = a, \quad a \cdot 1 = a$
- idempotence: $a + a = a, \quad a \cdot a = a$
- tautology/antilogy: $a + \overline{a} = 1, \quad a \cdot \overline{a} = 0$
- commutativity: $a + b = b + a, \quad ab = ba$
- distributivity: $a + (bc) = (a + b)(a + c),$
  $a(b + c) = ab + ac$
- associativity: $a + (b + c) = (a + b) + c = a + b + c,$
  $a(bc) = (ab)c = abc$
- De Morgan's law: $\overline{ab} = \overline{a} + \overline{b},$
  $\overline{a + b} = \overline{a} \cdot \overline{b}$
- others: $a + (ab) = a, \quad a + (\overline{a}b) = a + b,$
  $a(a + b) = a, \quad (a + b)(a + \overline{b}) = a$
- Always keep in mind: $\overline{ab} \neq \overline{a}\overline{b}, \quad \overline{a + b} \neq \overline{a} + \overline{b}$

[1] Probably one of the rare things to be known by heart

Boolean expressions

Using the basic operations, we can form **Boolean expressions**.

A **literal** is a Boolean (potentially negated) variable in an expression (e.g.,: if a, b, c are 3 boolean variables, we can write the Boolean expression $\overline{a}b + c\overline{b}$, which has 4 literals).

$\rightarrow$ **Any function from $\mathbb{B}^n$ to $\mathbb{B}$ can be described with a Boolean expression.**

$\rightarrow$ **Any function from $\mathbb{B}^n$ to $\mathbb{B}^m$ can be described with m Boolean expressions.**

$\rightarrow$ **Any boolean expression with n variables represents a function from $\mathbb{B}^n$ to $\mathbb{B}$.**

Truth tables

- A Boolean function can be represented by its **truth table**
- Columns list all variables and the result(s) of the expression(s)
- Rows list all possible combinations of the variable's values
- n variables $\Rightarrow 2^n$ values $\Rightarrow 2^n$ rows

x	y	$\overline{x}$	xy	$x+y$	$x \oplus y$	$\overline{xy}$	$\overline{x+y}$	$\overline{x \oplus y}$
0	0	1	0	0	0	1	1	1
0	1	1	0	1	1	1	0	0
1	0	0	0	1	1	1	0	0
1	1	0	1	1	0	0	0	1

Blackboard Examples

Normal forms

A Boolean function can be expressed with many different Boolean expressions:
e.g., if $f(a, b) = \overline{a}\overline{b}$, then we also have $f(a, b) = \overline{a + b}$ or $f(a, b) = \overline{(a + (ab)) + (b(a + b))}$ (cf Boolean Cheat Sheet)

We thus are interested in two **normal forms** (*forme canonique*):

- **Disjunctive normal form (DNF)**: a disjunction of conjunctions of literals (sum of products $\rightarrow$ sop):

$$\overline{a}b\overline{c} + \overline{a}\overline{b} + \overline{a}\overline{b}$$

- **Conjunctive normal form (CNF)**: a conjunction of disjunctions of literals (product of sums $\rightarrow$ pos):

$$(\overline{a} + b + \overline{c})\cdot(\overline{a} + \overline{b} + c)\cdot(a + \overline{b} + \overline{c})$$

From Boolean functions to DNF

1. given a Boolean function f
2. build the truth table of f
3. for each line where $f = 1$, form a conjunction of the literals in the line (the “min-terms”)

a	b	c	$f(a,b,c)$	
0	0	0	1	$\to \overline{a}\,\overline{b}\,\overline{c}$
0	0	1	0	
0	1	0	1	$\to \overline{a}b\overline{c}$
0	1	1	0	
1	0	0	1	$\to a\overline{b}\,\overline{c}$
1	0	1	1	$\to a\overline{b}c$
1	1	0	0	
1	1	1	0	

4. then build the disjunction (sum) of the min-terms.

$$f(a,b,c) = \overline{a}\,\overline{b}\,\overline{c} + \overline{a}b\overline{c} + a\overline{b}\,\overline{c} + a\overline{b}c.$$

From Boolean functions to CNF

1. write $\overline{f(a,b,c)}$ in DNF

a	b	c	$f(a,b,c)$	$\overline{f(a,b,c)}$	
0	0	0	1	0	
0	0	1	0	1	$\to \overline{a}\overline{b}c$
0	1	0	1	0	
0	1	1	0	1	$\to \overline{a}bc$
1	0	0	1	0	
1	0	1	1	0	
1	1	0	0	1	$\to ab\overline{c}$
1	1	1	0	1	$\to abc$

$$\overline{f(a,b,c)} = \overline{a}\overline{b}c + \overline{a}bc + ab\overline{c} + abc.$$

2. Use De Morgan's law to build the CNF[2]

$$f(a,b,c) = (a+b+\overline{c})\cdot(a+\overline{b}+\overline{c})\cdot(\overline{a}+\overline{b}+c)\cdot(\overline{a}+\overline{b}+\overline{c}).$$

[2] May seems difficult but CNF are useful – and actually simple – when the truth table mainly contains false values...

Blackboard Examples

DNF and CNF can be roborative

Once you've got the DNF/CNF, you can use Boolean logic to reduce the expression...

Useful operator - XOR (eXclusive-OR) noted
$XOR(a, b) = a \oplus b$:
$a \oplus b = 1$ **iff one and only one of** a **and** b **has value 1.**

a	b	$a \oplus b$
0	0	0
0	1	1
1	0	1
1	1	0

XOR can be expressed with negations, conjonctions and disjonctions:

$$a \oplus b = (a + b)\overline{ab} = (a + b)(\overline{a} + \overline{b}) = \overline{a}b + a\overline{b}.$$

XOR is commutative and associative...