03 Logic networks

03.01 Boolean algebra

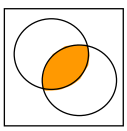
- Definitions
- Boolean functions
- Properties
- Canonical forms
- Synthesis and minimization

Definitions

- Boolean set: $\mathbf{B} = \{0,1\}$
- Boolean constants: 0, 1
- Boolean variable: $x \in \{0,1\}$
- Boolean functions: $z=f(x_1,x_2,\ldots,x_n)$ $\quad f:\mathbf{B}^n \rightarrow \mathbf{B}$
- Operations:

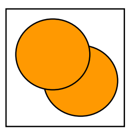
$$and: \mathbf{B}\times\mathbf{B} \rightarrow \mathbf{B} \qquad (x,y) \quad z = x\cdot y = xy$$

x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1



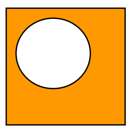
$$or: \mathbf{B}\times\mathbf{B} \rightarrow \mathbf{B} \qquad (x,y) \quad z = x+y$$

x	y	$x+y$
0	0	0
0	1	1
1	0	1
1	1	1



$$not: \mathbf{B} \rightarrow \mathbf{B} \qquad x \quad z = x' = \overline{x}$$

x	x'
0	1
1	0



Boolean functions

$$f : \mathbf{B}^n \rightarrow \mathbf{B}$$

- Truth table:

  Table of 2^n rows that associates a Boolean value to each configuration of *n* independent variables

  There are 2^{2^n} different functions of *n* variables

a	b	c	f
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

- Boolean expression:

  Expression of Boolean variables, Boolean constants and operators

$$f = a \cdot b + c' = ab + \overline{c} = (ab) + \overline{c}$$

Properties

Idempotent laws

$$x \cdot x = x \qquad x + x = x$$

Distributive laws

$$x \cdot (y+z) = (x \cdot y) + (x \cdot z) \qquad x + (y \cdot z) = (x+y) \cdot (x+z)$$

Associative laws

$$xyz = x(yz) = (xy)z \qquad x+y+z = x+(y+z) = (x+y)+z$$

Commutative laws

$$xy = yx \qquad x+y = y+x$$

Identity elements

$$x \cdot 1 = x \qquad x+0 = x$$

Null laws (forcing elements)

$$x \cdot 0 = 0 \qquad x+1 = 1$$

Complement laws

$$x \cdot x' = 0 \qquad x + x' = 1$$

Absorption laws

$$x + xy = x \qquad x(x+y) = x$$

De Morgan’s laws (duality principle)

$$(xy)' = x' + y' \qquad (x+y)' = x'y'$$

Idempotent laws

Idempotent laws

$$x \cdot x = x \qquad x + x = x$$

By perfect induction:

x	y	xy
0	0	0
0	1	0
1	0	0
1	1	1

x	$x*x$	
0	0*0	0
1	1*1	1

x	y	$x+y$
0	0	0
0	1	0
1	0	0
1	1	1

x	$x+x$	
0	0+0	0
1	1+1	1

Computer Architecture
03.01 Boolean algebra
applied **computer science**
urbino worldwide campus

Absorption laws

Absorption laws

$$x + xy = x \qquad x(x+y) = x$$

By Boolean manipulation:

$$x + x \cdot y = x \cdot (1 + y) = x \cdot 1 = x$$

$$x(x+y) = x \cdot x + x \cdot y = x + x \cdot y = x$$

De Morgan's laws

De Morgan's laws

$(xy)' = x' + y'$ $(x+y)' = x'y'$

By perfect induction:

$x=0, y=0$	➔	(00)'=0'=1	0'+0'=1+1=1
$x=0, y=1$	➔	(01)'=0'=1	0'+1'=1+0=1
$x=1, y=0$	➔	(10)'=0'=1	1'+0'=0+1=1
$x=1, y=1$	➔	(11)'=1'=0	1'+1'=0+0=0

$x=0, y=0$	➔	(0+0)'=0'=1	0'0'=11=1
$x=0, y=1$	➔	(0+1)'=1'=0	0'1'=10=0
$x=1, y=0$	➔	(1+0)'=1'=0	1'0'=01=0
$x=1, y=1$	➔	(1+1)'=1'=0	1'1'=00=0

Canonical forms

- There are infinite equivalent Boolean expressions.
- The equivalence (i.e., identity) between two expressions can be demonstrated:
  1. By perfect induction
  2. By Boolean manipulation
- Canonical forms associate unique expressions to each function
- Checking the equivalence between two functions reduces to a comparison of their canonical representations

Canonical forms (Sum of Products)

- *Literal*: independent variable taken either in true or complemented form (e.g., *x*, *x'*)
- *Minterm*: Product of all independent variables taken either in true or complemented form
  – A minterm represents a Boolean function that takes value 1 corresponding to a unique configuration of input variables (e.g., *f*(*a,b,c*)=*ab'c* takes vale 1 for *abc*=101)
  – A Boolean function that takes value 1 for *M* different configurations (that has *M* 1's in the truth table) can be expressed as the sum of the *M* minterms associated with the *M* 1's
- Any Boolean function can be expressed as a sum of products
- A sum of minterms, with fixed variable order, is a canonical form

From truth tables to SoPs

a	*b*	*c*	*f*										
0	0	0	1		1		0		0		0		0
0	0	1	0		0		0		0		0		0
0	1	0	1		0		1		0		0		0
0	1	1	0	=	0	+	0	+	0	+	0	+	0
1	0	0	1		0		0		1		0		0
1	0	1	0		0		0		0		0		0
1	1	0	1		0		0		0		1		0
1	1	1	1		0		0		0		0		1

$$f = a'b'c' + a'bc' + ab'c' + abc' + abc$$

Boolean minimization

*Given a Boolean function,
find a Boolean expression that represents the function
using a minimum number of literals*

$f = a'b'c' + a'bc' + ab'c' + abc' + abc$	15 literals
$f = a'c' + ab'c' + abc' + abc$	11 literals
$f = a'c' + ab'c' + ab$	7 literals

- In general, this is not an easy task
- There is a closed-form solution for *2-level* SoP expressions
- There is no closed-form solution for general *multi-level* expressions.
- Heuristic solutions found by Boolean manipulation

Boolean minimization (example)

$f = a'b'c' + a'bc' + ab'c' + abc' + abc$	15 literals (SoP)
$= a'c'(b' + b) + ab'c' + ab\ (c' + c)$	11 literals (distributive)
$= a'c' + ab'c' + ab$	7 literals (complement)
$= a'c' + a\ (b'c' + b)$	6 literals (distributive)
$= a'c' + a\ (b'c' + b\ c' + b)$	8 literals (absorption)
$= a'c' + a\ ((b'+b)\ c' + b')$	7 literals (distributive)
$= a'c' + a\ (c' + b)$	5 literals (complement)
$= a'c' + a\ c' + ab$	6 literals (distributive)
$= (a' + a)\ c' + ab$	5 literals (distributive)
$= c' + ab$	3 literals (complement)

Remark: the number of literals doesn't decrease at every step.

This makes the process non trivial