Lecture 3: Jan. 10, 2007

Definition of literals:

When a Boolean function such as

$$F = x'y'z + xyz' + x'z'$$

is implemented using logic gates, each term is represented by a gate and each variable in a term is an input to that gate. Each variable appearing in a function, whether in complemented or original form is called a literal. For example, the function F has three terms and 8 literals.

Standard forms

In standard form a function may be represented as a sum of product terms or product of sums.

Standard sum of product form:

In this form the function is represented as the sum (OR) of several terms. Each term

is the product (AND) of one or more literals.
For example

$$F_1 = x + x'y + xyz + x'y'z'$$

is in sum of products standard form.
In <u>product of sums standard</u> form, a function is represented as product (AND) of several sum (OR) "terms".
For example:

$$F_2 = x(x'+y')(x+z')(x+y+z).$$

<u>Canonical forms</u>:
Take two variables x and y, we can form with x and y (and their complements) $2^2=4$ <u>standard products</u> or <u>minterm</u>:

$$xy,\ x'y,\ xy',\ x'y'$$

In general with n variables, we can form 2^n minterms.
To enumerate the 2^n minterms corresponding to n variables, we write number 0 to 2^n-1 (in binary) under the n variables, we form

the minterm corresponding to each row is formed by ANDing the variable ~~and~~ or their complements. Those with a 1 under them appear uncomplemented and those with value 0 are complemented.

As an example ~~[illegible]~~ minterms for 3 variable x, y and z are shown in the following table:

x	y	z	Term	Designation
0	0	0	$x'y'z'$	m_0
0	0	1	$x'y'z$	m_1
0	1	0	$x'yz'$	m_2
0	1	1	$x'yz$	m_3
1	0	0	$xy'z'$	m_4
1	0	1	$xy'z$	m_5
1	1	0	xyz'	m_6
1	1	1	xyz	m_7

Standard Sum or Maxterm

A Standard sum or Maxterm is formed by ORing the n variable. Those with 0 value appear uncomplemented and those with value 1 appear complemented.

For example, the following table shows the maxterms for three variables x, y, z:

x	y	z	Maxterm	Designation
0	0	0	$x+y+z$	M_0
0	0	1	$x+y+z'$	M_1
0	1	0	$x+y'+z$	M_2
0	1	1	$x+y'+z'$	M_3
1	0	0	$x'+y+z$	M_4
1	0	1	$x'+y+z'$	M_5
1	1	0	$x'+y'+z$	M_6
1	1	1	$x'+y'+z'$	M_7

Note that each Maxterm is the Complement of the corresponding minterm.

A Boolean function can be expresse algebraically by forming the minterm for those rows of the truth table for which the function is 1.

For example

x	y	z	F_1	F_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Function F_1 is 1 for 001, 100 and 111. The minterm for these terms is: $x'y'z$, $xy'z'$, xyz, respectively. So,

$$F_1 = x'y'z + xy'z' + xyz = m_1 + m_4 + m_7$$

This sometimes is shown as

$$F_1(x, y, z) = \Sigma(1, 4, 7)$$

Similarly,

$$F_2 = x'yz + xy'z + xyz' + xyz = m_3 + m_5 + m_6 + m_7$$

or

$$F_2(x, y, z) = \Sigma(3, 5, 6, 7)$$

This representation is an example of <u>Canonical form representation</u>.

Another <u>Canonical</u> representation is formed by <u>ANDing</u> the <u>Maxterms</u> Corresponding to those rows of the truth table for which the function is equal to 0.

For example:

$$F_1 = (x+y+z)\cdot(x+y'+z)\cdot(x+y'+z')(x'+y+z')$$
$$(x'+y'+z) = M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6$$

Note that this canonical form can be formed by first find the sum of products canonical form for the complement of the function (by ORing the minterms corresponding to 0's of the original function) and then taking the complement to get the product of sums canonical form. That is:

$$F_1' = x'y'z' + x'yz' + x'yz + xy'z + xyz'$$

$$\begin{aligned} F_1 &= (x'y'z' + x'yz' + x'yz + xy'z + xyz')' \\ &= (x+y+z)(x+y'+z)(x+y'+z')(x'+y+z')(x'+y'+z) \\ &= M_0 M_2 M_3 M_5 M_6 \end{aligned}$$

This can be represented as

$$F_1(x,y,z) = \prod(0,2,3,5,6)$$

Similarly:

$$\begin{aligned} F_2 &= (x+y+z)(x+y+z')(x+y'+z)(x'+y+z) \\ &= M_0 \cdot M_1 \cdot M_2 \cdot M_4 = \prod(0,1,2,4) \end{aligned}$$

A function represented in sum of minterms can be converted to a product of maxterms form and vice versa.

The procedure is as follows: A sum of minterms expression has those minterms for which the function is 1. The complement of the function consists of the rest of the terms. The complement of the complement will have maxterms for these rows (the rest of the terms). For example, in the previous example:

$$F_1 = \sum(1,4,7)$$

$\overline{F_1}$ is the sum of the rest of minterms, i.e.,

$$F_1' = \sum(0,2,3,5,6)$$

So

$$F_1 = (F_1')' = \prod(0,2,3,5,6)$$

Other logic operations

n variables take 2^n values, i.e., the truth table for an n-bit function has 2^n rows. Each row can have a 1 or a 0. So, there are 2^{2^n} functions with n binary inputs. For $n = 2$, we have for values for x and y. So there are $2^4 = 16$ function.

Truth Tables for the 16 Functions of Two Binary Variables

x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

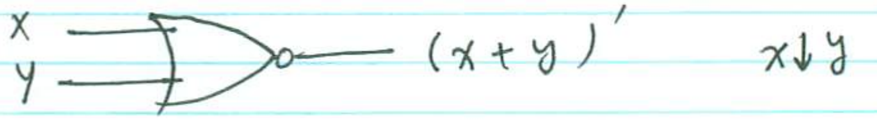
Following is the list of Boolean expressions for these functions.

Boolean Expressions for the 16 Functions of Two Variables

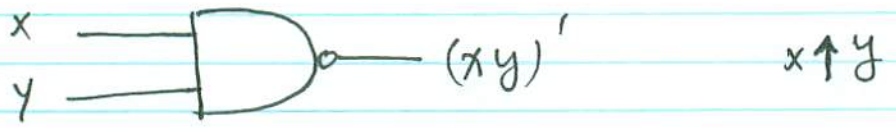
Boolean functions	**Operator symbol**	**Name**	**Comments**
$F_0 = 0$		Null	Binary constant 0
$F_1 = xy$	$x \cdot y$	AND	x and y
$F_2 = xy'$	x/y	Inhibition	x, but not y
$F_3 = x$		Transfer	x'
$F_4 = x'y$	y/x	Inhibition	y, but not x
$F_5 = y$		Transfer	y
$F_6 = xy' + x'y$	$x \oplus y$	Exclusive-OR	x or y, but not both
$F_7 = x + y$	$x + y$	OR	x or y
$F_8 = (x + y)'$	$x \downarrow y$	NOR	Not-OR
$F_9 = xy + x'y'$	$(x \oplus y)'$	Equivalence	x equals y
$F_{10} = y'$	y'	Complement	Not y
$F_{11} = x + y'$	$x \subset y$	Implication	If y, then x
$F_{12} = x'$	x'	Complement	Not x
$F_{13} = x' + y$	$x \supset y$	Implication	If x, then y
$F_{14} = (xy)'$	$x \uparrow y$	NAND	Not-AND
$F_{15} = 1$		Identity	Binary constant 1

We have learned about F_1 (AND) and F_7 (OR) already. Some other functions that are gates are:

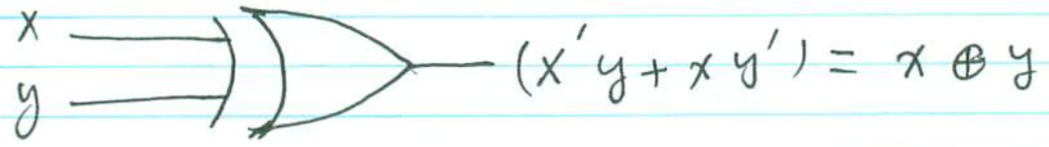
F_8 : $(x+y)'$ called NOR : NOT-OR



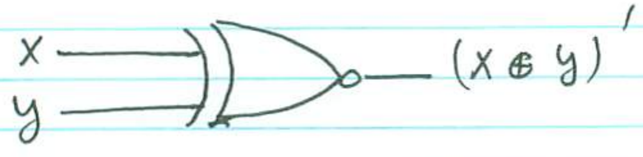
F_{14} : $(xy)'$ called NAND : NOT-AND



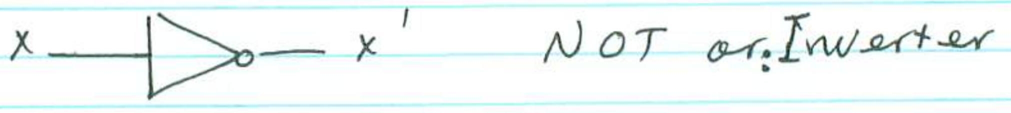
F_6 : $xy' + x'y$ called Exclusive OR : XOR

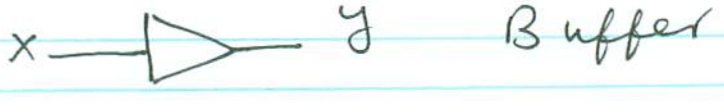


F_9 : $xy + x'y'$ called X-NOR or equivalence



unitary (single input) gates:



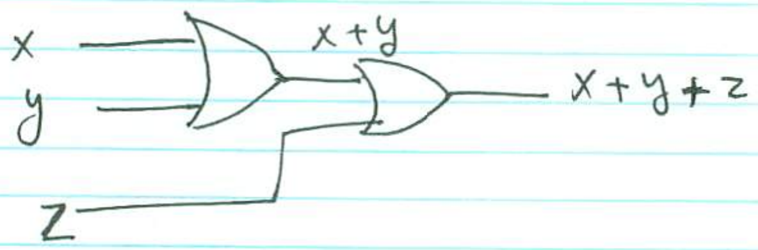


Gates with more than two inputs:

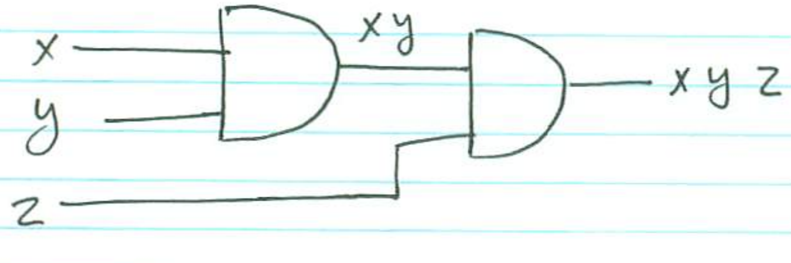
AND gates and OR gates are commutative and associative, i.e.,

$$(x+y)+z = x+(y+z)$$

So, a 3-input OR gate can be made from two OR gates with two inputs each



Similarly:



But NAND and NOR operations are not associative, e.g., for NOR gates

$$(x\downarrow y)\downarrow z = [(x+y)'+z]' = (x+y)z' = xz'+yz'$$

$$x\downarrow(y\downarrow z) = [x+(y+z)']' = x'(y+z) = x'y+x'z$$

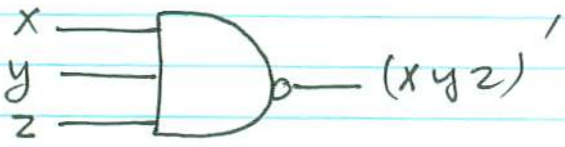
So:

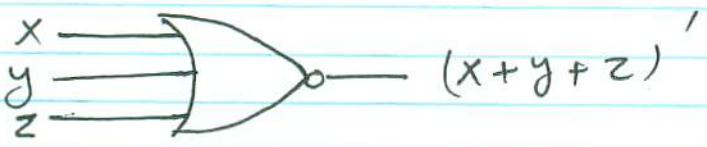
$$(x\downarrow y)\downarrow z \neq x\downarrow(y\downarrow z)$$

To avoid this difficulty NAND and NOR gates are implemented using complemented (inverted) AND and OR gates, i.e.,

$$x \uparrow y \uparrow z = (xyz)'$$

$$x \downarrow y \downarrow z = (x+y+z)'$$





XOR and X-NOR (equivalence) are both commutative and associative. So, they can be extended to more than two inputs