Digital Logic

Based on “Digital Design” by Mano and Ciletti

Digital logic gates

In lecture we talked about logic control and boolean logic

Remember our primary boolean operators are AND, OR, and NOT

As a result these are also the basis for hardware logic gates as well

Though, logic gates can be constructed for other operations as well

Truth Tables

Truth tables are a mathematical tool in logic and Boolean Algebra to represent all possible inputs and outputs

Truth tables typically list all possible combination of variables (True and False) and their output

For each set of inputs there is a corresponding output often represented as 1,0 or True, False

Truth tables essential for analyzing the behavior of logical expressions

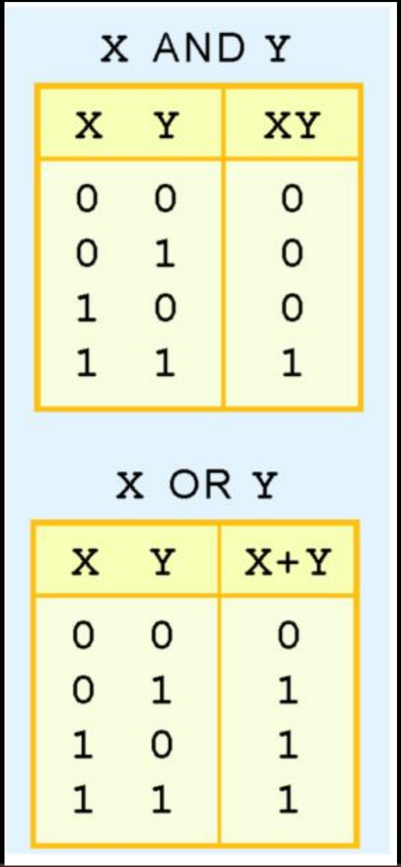
Boolean algebra

A boolean operator can be completely described using a truth table

The truth tables for AND and OR are shown here

The AND operator is also known as the boolean product

The OR operator is the Boolean sum



X AND Y

X	Y	XY
0	0	0
0	1	0
1	0	0
1	1	1

X OR Y

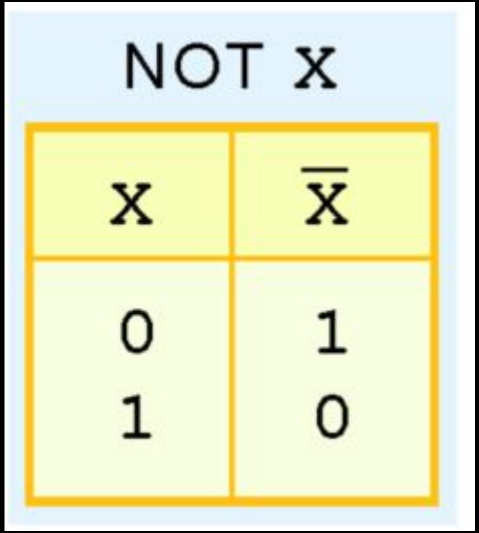
X	Y	X+Y
0	0	0
0	1	1
1	0	1
1	1	1

Boolean algebra

We can see the truth table for the NOT operator here

The NOT operator is often denoted with an over bar

It is sometimes denotes by a prime mark (‘)



NOT x

x	$\overline{x}$
0	1
1	0

Boolean algebra

A boolean function has:

- At least one Boolean variable
- At least one Boolean operator
- At least one input from the set {0,1}

It produces an output from the set {0,1}

{0,1} A akin to off/one or false/true

This is why binary is so helpful

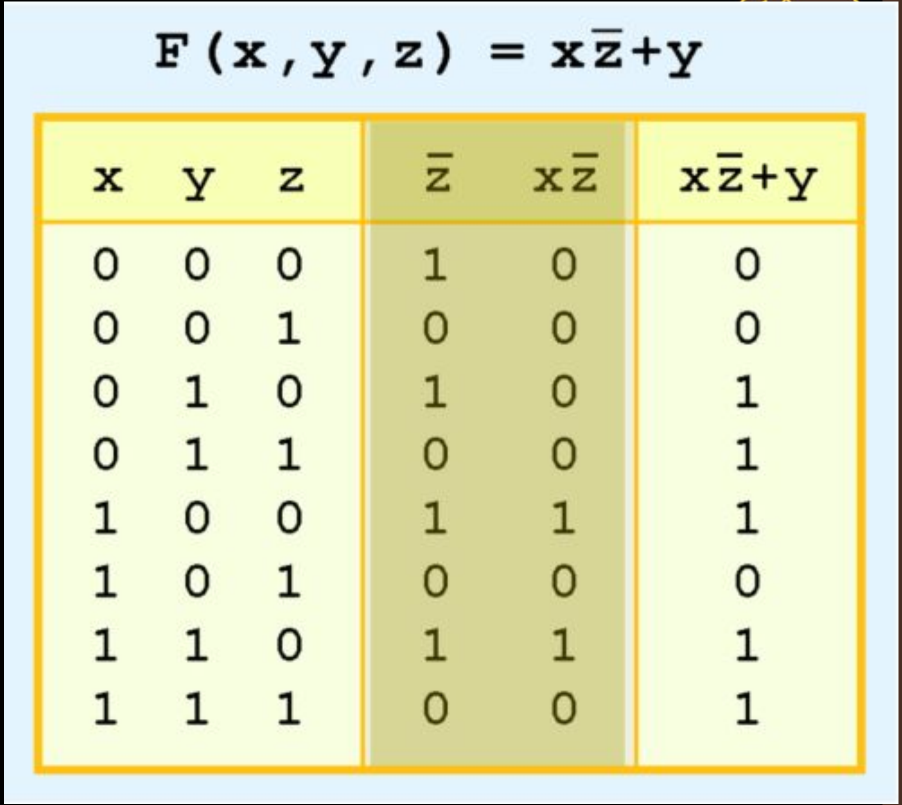
More complex truth tables

We can make more complex truth tables for more complex equations

For example F(x,y,z) = xz’ +y

Where F is a function of x,y,z such that the and of x and z’ are OR-ed with y

To make evaluation easier, the truth table contains extra (shared) columns of sub parts



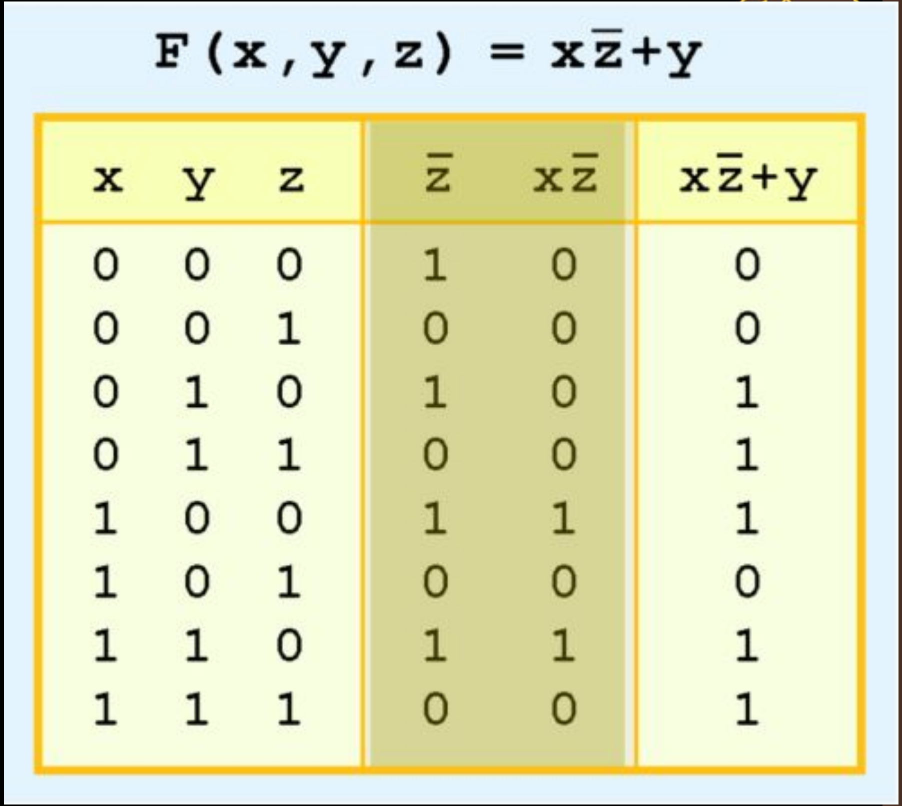
$F(x,y,z) = x\bar{z}+y$

x	y	z	$\bar{z}$	$x\bar{z}$	$x\bar{z}+y$
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	0	1

More complex truth tables

As with arithmetic, Boolean operations have common rules of precedence

The NOT operator has highest priority, followed by AND and OR



$F(x,y,z) = x\bar{z}+y$

x	y	z	$\bar{z}$	$x\bar{z}$	$x\bar{z}+y$
0	0	0	1	0	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	0	1
1	0	0	1	1	1
1	0	1	0	0	0
1	1	0	1	1	1
1	1	1	0	0	1

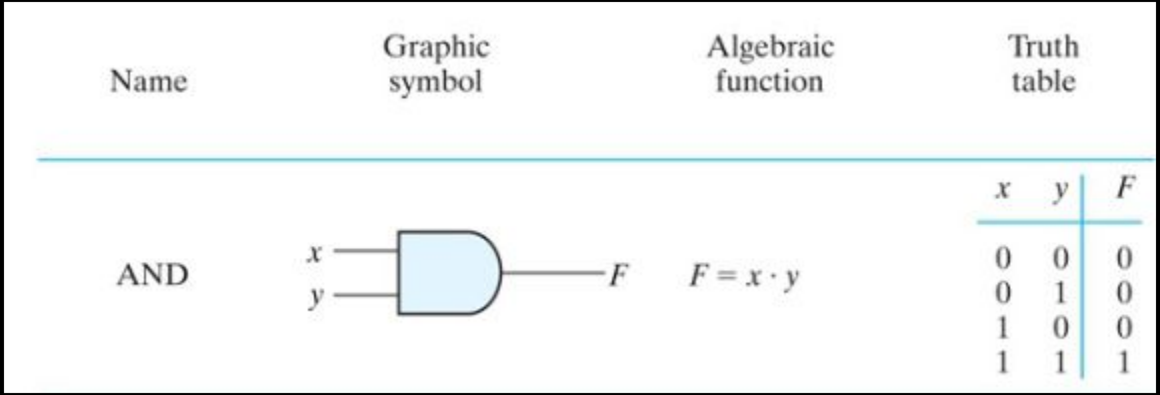
Boolean Algebra

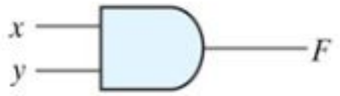
Digital computers contain circuits that implement Boolean functions

The simpler that we can make our Boolean functions, the smaller the circuit

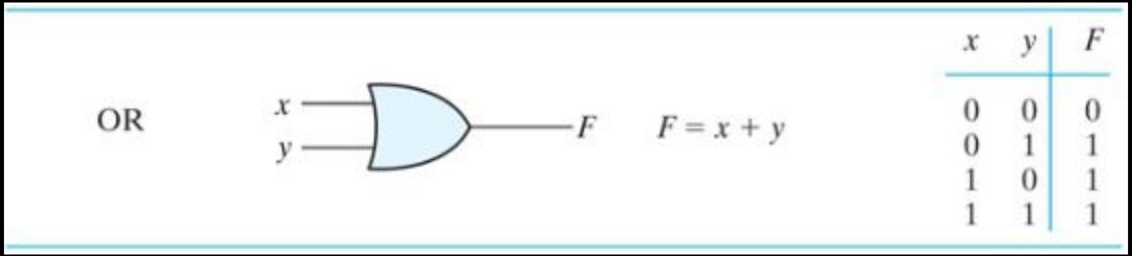
- The smaller the circuit the cheaper it is, the less power it needs, and the faster it will run

Digital logic gates



Name	Graphic symbol	Algebraic function	Truth table
AND		$F = x \cdot y$	x y | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1

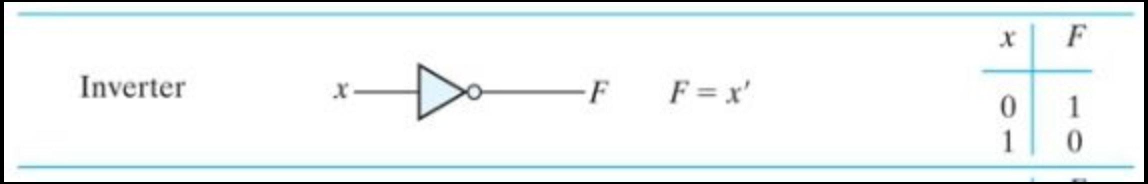
Digital logic gates


OR

$F = x + y$

x	y	F
0	0	0
0	1	1
1	0	1
1	1	1

Digital logic gates

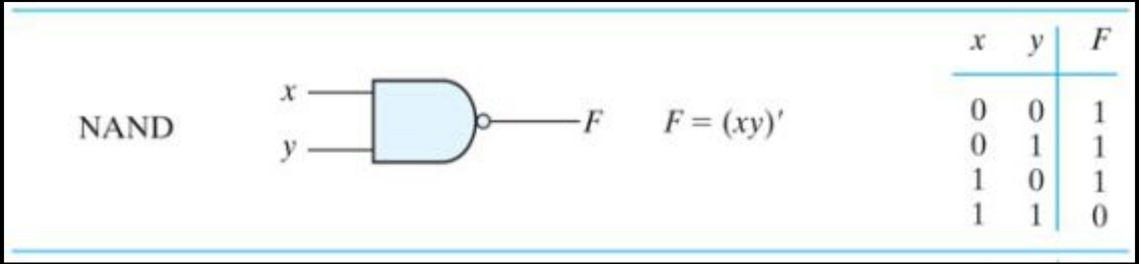


Inverter

$F = x'$

x	F
0	1
1	0

Digital logic gates

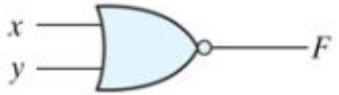

NAND

$F = (xy)'$

x	y	F
0	0	1
0	1	1
1	0	1
1	1	0

Digital logic gates

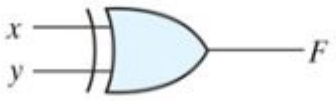
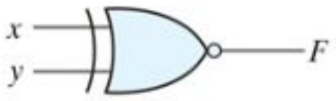
NOR



$F = (x + y)'$

x	y	F
0	0	1
0	1	0
1	0	0
1	1	0

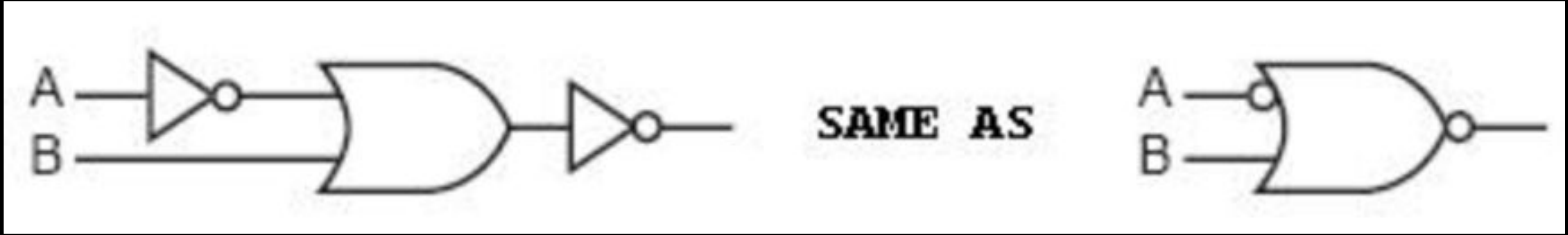
Digital logic gates

			x	y	F
Exclusive-OR (XOR)		$F = xy' + x'y$ $= x \oplus y$	0	0	0
			0	1	1
			1	0	1
			1	1	0
			x	y	F
Exclusive-NOR or equivalence		$F = xy + x'y'$ $= (x \oplus y)'$	0	0	1
			0	1	0
			1	0	0
			1	1	1

Digital logic gates

There is a simple graphical convention you likely noticed

The NOT gate is sometimes reduced to a small circle attached to some other gate



Digital logic gates

Gates can be negated to get the complement

For example, the NAND function is the complement of the AND function

Denoted with the standard gate followed by a small circle

The NOR function is the complement of the OR function

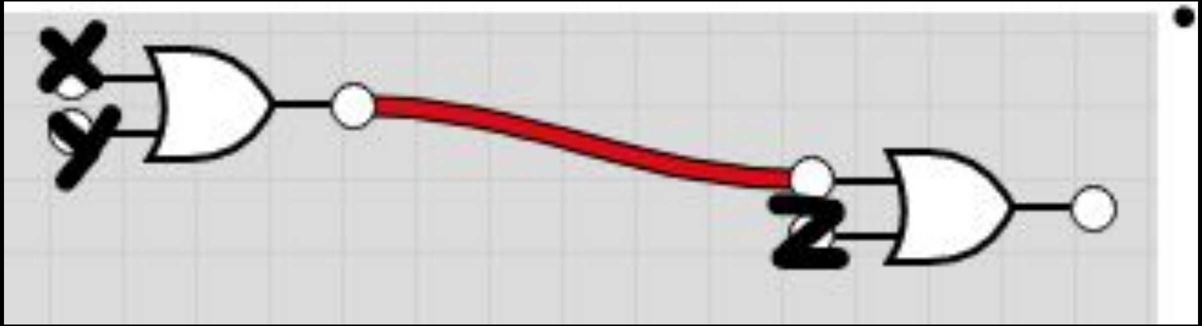
NAND and NOR gates are used as standard gates, more so than their complements

The exclusive-OR (XOR) gate includes the curved line on its side

Multiple inputs

You can pair gates together to create multiple inputs for example:

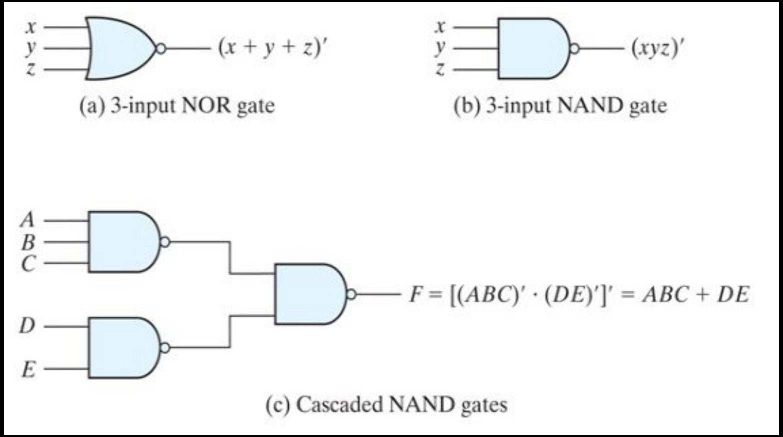
(x OR y) OR z == x OR (y OR z) == x OR y OR z



Multiple inputs

Logic gates can also be shown with three inputs

Correct parentheses must be used to denote proper nesting



Combinational circuits

A combinational circuit has no memory

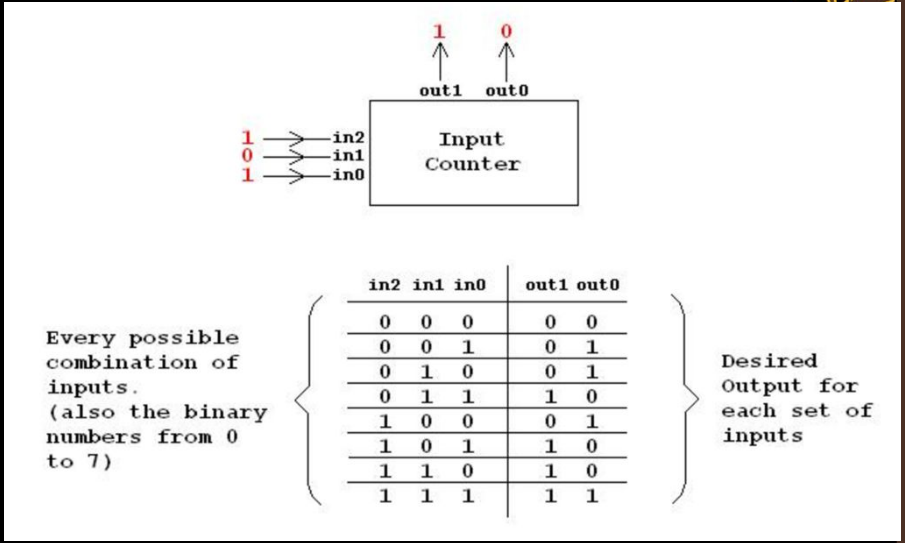
Its output depends on its input

Each input and output is either a 1 or a 0

To specify its behavior we use a truth table

An example:

- Imagine a circuit with 3 inputs and 2 outputs
- The outputs form a binary number



Every possible combination of inputs. (also the binary numbers from 0 to 7)

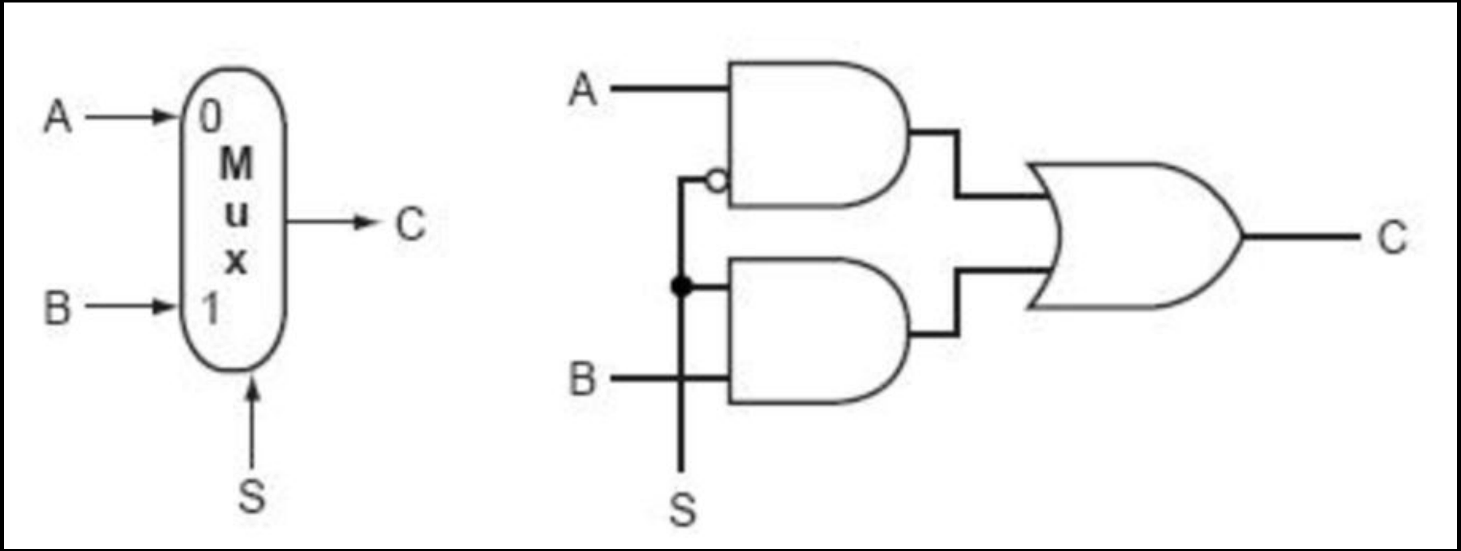
in2	in1	in0	out1	out0
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Desired Output for each set of inputs

Multiplexores

A multiplexor allows the choosing of one of two incoming bits

- S selects either A or B and sends its value to c



Positive and negative logic

The binary signal of output for any gate has one of two values

- 1, or high
- 0, or low

Two signal values are assigned to two logic values, so there exists two different assignments of signal value

- Positive logic, is used when the high level represents logic 1
- Negative logic is defined when a low power level defines 1

A variable can be noted as the complement (negation) with a ‘ so x’ is the complement of x