LOGIC

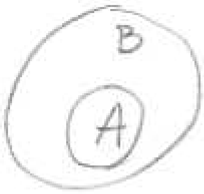
Long time ago, Aristotle set the basis of logic reasoning. He was trying to formalize what was a common practice of philosophers, namely to arrive at some conclusions starting from a set of premises.
His work is summarized is a set of books under the name of "Organon"
Aristotle was born in 384 BC, more than 2200 years ago (if you search in google you should be able to find an on line version of the books. Also, Organon means "Instrument").
Here I just want to introduce some basic concepts.
We talk about class of objects lik the class of human beings or the class of animals.
a clas is a set of objects with some properties. Aristotle calls this class Terms.

We use then capital letters for terms.

For instance:

① all A are B

if we use sets to represent this situation then the set A is contained in the set B:



Saying that all A are B also means that if something is not B then it cannot be A otherwise it wold be B. Then ① implies:

② None not B is A

We can use symbles to write ① and ② in the following way:

① $A \Rightarrow B$ ($\Rightarrow$ implication)

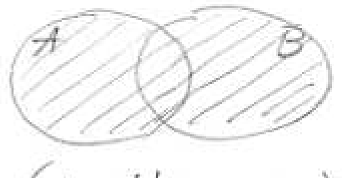
② $B' \Rightarrow A'$ (' means not)

Using sets, ② means the following:

an element outside B cannot be in A because A is contained in B (from ①)

Anothe example of the use of logic is the following:

3 all elements A or B



(is the union)

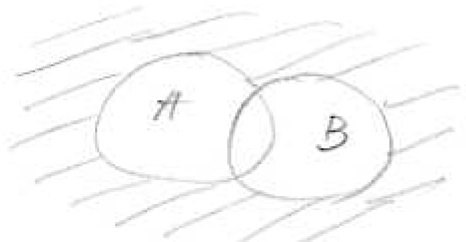
3 means all the element with property A or property B. Using set representation it is the union of two sets

Saying:

none of the element of A or B

mens all the elements that are neither im A nor im B:

all element not im A and not im B



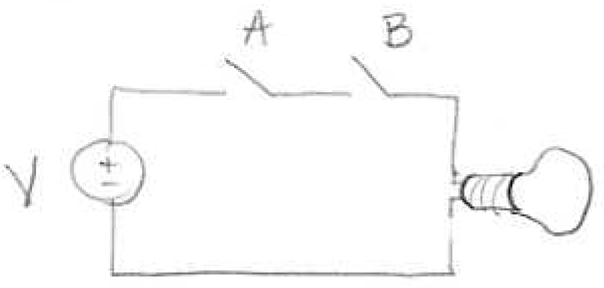
Im symbles:

A or B $\Rightarrow$ not A and not B

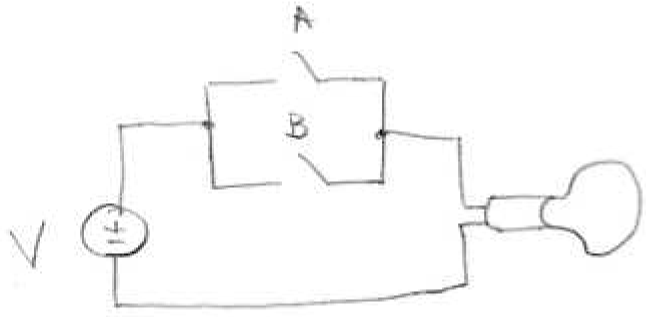
this is the De Morgan law, im set theory:

$$\overline{A \cup B} = \bar{A} \cap \bar{B}$$

As a practical example, we can use a simple circuit and do some reasoning about it :



AND



OR

AND) A and B ⇒ light

OR) A or B ⇒ light

In the and case, if we don't se light what does it means?

$$\text{no light} \Rightarrow \text{not}\,(A \text{ and } B) \equiv \text{not } A \text{ or not } B$$

it is not an exclusive or, it says that when at least one switch is open the light is off, which is correct.

FROM LOGIC TO ALGEBRA

Aristotle was not using formulas for describing his reasoning. If you try to read "Prior Analytics", it is a continuous stream of premises and deductions.

George Boole (1815, Lincoln, UK) was able to express a reasoning into a system of algebraic equations.

Boole represented class of objects using letters and introduced operators on this classes. From his original work:

[If an adjective, as "good", is employed as a term of description, let us represent by a letter, as y, all thing to which the description "good" is applicable, i.e., "all good things", or the class of "good things". Let it further be agreed, that by the combination xy shall be represented that class of things to which the names or descriptions represented by x and y are simultaneously applicable. Thus, if x alon stands for "white things"

and y for "sheep" let xy stand for "white sheep";
and in like manner, if z stands for "horned things"
let zxy represent "horned white sheep"]

I think that it is very well explained.
Boole wanted to use numbers and standard algebra to represent a reasoning.
The problem that he met is the following.
Consider A to be the class "sheep".
Then AA is the class of sheep that are also sheep, so it is again sheep:

$$AA = A$$

Since he wanted to use numbers, then the only numbers that satisfy that equation, $A^2 = A$, are 0 and 1.
So if you were to use numbers to represent sets, they have to be either 0 or 1 to wich we have to give a meaning as sets:

0: since it must be $0A = 0 \ \forall A$, then 0 represents the empty set or the class to which nothing belongs.

1: since it must be $1A = A \;\forall A$, then 1 is the set that contains every object. It is called the universe

Operations like + and - also have a meaning.

$A + B$ is the class containing the elements of A and the elements of B. In set terminology $A+B$ is the union of A and B.

$A - B$ is the class of all element that are in A but not in B.

Using this operations we can write sentences of the Aristotelian logic:

All A is B $\rightarrow$ $AB = A$ or $A(1-B)=0$

mean that: there are no objects that are <u>in A and not in b</u>

No A is B $\rightarrow$ $AB = 0$

Also there are some properties that can be stated and verified:

$X + (1 - X) = 1$ (using simple algebra)

Let's see if this equation has the meaning that we expect using logic:

all elements in x plus all the elements not in x

(x) + $(1 - x)$

Of course it must be the universe which is the result of the equation (the universe is 1).

Also:

$X(1-X) = X - XX = X - X = 0$ (algebra)

Using logic:

all elements that are in x and not in x

x $(1-x)$

of course there are no element that is

in x and not in x at the same time so the result must be 0 (the emty set).

Now I'll use an exampli from the book "The universal computer" by Martin Davis.

SUSAN: Did you leave it in the supermarket when you were shopping?

JOE: No, I telephoned them, and they didn't find it. If I had left it there, they surely would have found it.

SUSAN: Wait a minute! You wrote a check at the restaurant last night and I saw you put your checkbook in your jacket pocket. If you haven't used it since, it must still be there.

JOE: You're right. I haven't used it. It's in my jacket pocket.

Joe looks and (if it's a good day for logic), the missing checkbook is there. Let us see how Boole's algebra could be used to analyze Joe and Susan's reasoning.

In their reasoning, Joe and Susan were dealing with the following propositions (each labeled with a letter):

$L =$ Joe left his checkbook at the supermarket,

$F =$ Joe's checkbook was found at the supermarket,

$W =$ Joe wrote a check at the restaurant last night,

$P =$ After writing the check last night, Joe put his checkbook in his jacket pocket,

$H =$ Joe hasn't used his checkbook since last night,

$S =$ Joe's checkbook is still in his jacket pocket.

They used the following pattern:

PREMISES:

If L, then F.

Not F.

W & P.

If W & P & H, then S.

H.

CONCLUSIONS:

Not L.

S.

Like Aristotle's syllogisms, this pattern forms a valid inference. As with any valid inference, the truth of sentences called *conclusions* is inferred from the truth of other sentences called *premises*.

Boole saw that the same algebra that worked for classes would also work for inferences of this kind.[25] He used an equation like $X = 1$ to mean that the proposition X is true; likewise he used the equation $X = 0$ to mean that X is false. Thus, for "Not X," he could write the equation $X = 0$. Also, for "X & Y" he wrote the equation $XY = 1$. This works because X & Y is true precisely when X and Y are both true, while algebraically, $XY = 1$ if $X = Y = 1$, but $XY = 0$ if either $X = 0$ or $Y = 0$ (or both).

Finally, the statement "If X, then Y" can be represented by the equation

$$X(1 - Y) = 0.$$

To see this, think of this statement as asserting that

$$\textit{if } X = 1, \textit{ then } Y = 1.$$

But indeed, substituting $X = 1$ in the proposed equation leads to $1 - Y = 0$, that is, to $Y = 1$.

Using these ideas, Joe and Susan's premises can be expressed by the equations

$$\begin{aligned} L(1 - F) &= 0, \\ F &= 0, \\ WP &= 1, \\ WPH(1 - S) &= 0, \\ H &= 1. \end{aligned}$$

Substituting the second equation in the first, we get $L = 0$, the first desired conclusion. Substituting the third and fifth equations in the fourth, we get $1 - S = 0$, that is, $S = 1$, the other desired conclusion.

Now of course, Joe and Susan had no need for this algebra. But the fact that the kind of reasoning that takes place informally and implicitly in ordinary human interactions could be captured by Boole's algebra encouraged the hope that more complicated reasoning could be captured as well. Mathematics may be thought of as systematically encapsulating highly complex logical inferences, so an ultimate test of a theory of logic

BOOLEAN ALGEBRA

If you want to know more on this subject I would suggest "Boolean Reasoning" by F. M. Brown

We can formally define a Boolean algebra in the following way:

A Boolean algebra is an algebraic structure

$(B, +, \cdot, 0, 1)$ where:

- B is a set called carrier
- $+, \cdot$ are two binary operations on B
- $0, 1$ are two distinct members of B

that satisfies the following postulates:

- $\forall a, b \in B$

  $a + b \in B$ and $a \cdot b \in B$

- $\forall a, b \in B$ (commutative law)

  $a + b = b + a$ and $a \cdot b = b \cdot a$

- $\forall a, b, c \in B$ (distributive law)

  $a + (b \cdot c) = (a + b)(a + c)$

  $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$

- $\forall a \in B$

$$0 + a = a$$
$$1 \cdot a = a$$

- $\forall a \in B, \; \exists \; a' \in B$ s.t.

$$a + a' = 1$$
$$a \cdot a' = 0$$

Example

Boolean algebra of sets

S is a set

2^S is its powerset meaning the set of all possible subsets of S

The Boolean algebra is the following

$$(2^S, \cup, \cap, \emptyset, S)$$

You can easily check that all postulates are satisfied. For instance:

if $A \subseteq S$ then $\emptyset \cup A = A$
$S \cap A = A$

$\forall \; A \subseteq S, \exists \; B \subseteq S$ s.t.

$$A \cup B = S$$
$$A \cap B = \emptyset$$

(just take $B = S \setminus A$)

This algebra is important because there is an equivalence theorem that says that any Boolean algebra is isomorphic to the Boolean algebra of sets.

For our purposes, we use the Boolean algebra of Boolean functions.
A Boolean function of n variables is a table where there are n columns representing the n variables that can take values 0 or 1 and a column that is the value of the fuction which also takes values 0 or 1.
Formaly:

$$F_n : \{0,1\}^n \rightarrow \{0,1\}$$

For instance a boolean function of variables x_1 an x_2 could be:

x_1	x_2	$F_2(x_1, x_2)$
0	0	0
0	1	0
1	0	0
1	1	1

The algebra is then:

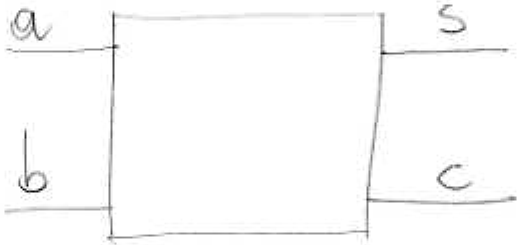
$$(F_n(B), +, \cdot, 0, 1)$$

all function of n variable. B is the set containing the n symbols used as variables.

Example:

consider Boolean functions of two variables.
We want to write the function that represents the binary sum of two variables. The sum of two binary digit needs actually two bits because if the two digits are both 1 then their bynary sum is 10 (which is 2 in decimal representation).
Since a boolean function can take value 0 or 1 we need two boolean functions, one that we call

sum and another one that we call carry.



a	0	0	1	1
b	0	1	0	1
a+b	00	01	01	10

c s

Here are the two boolean functions

a	b	S(a,b)
0	0	0
0	1	1
1	0	1
1	1	0

a	b	C(a,b)
0	0	0
0	1	0
1	0	0
1	1	1

The representation of a boolean function as a table is <u>canonical</u> meaning that a table represents one and only one function and viceversa, a function is represented by one and only one table.

We use a different representation for boolean function that we call Boolean Formula

Here is the formal definition of Boolean Formulas:

Given a Boolean algebra B and n symbols $x_1 \dots x_n$, the set of all boolean formulas on the n symbols is defined by:

1) The elements of B are boolean formulas
2) The symbols $x_1 \dots x_n$ are Boolean formulas
3) If g and h are Boolean formulas then so are:
   a) $(g)+(h)$
   b) $(g)(h)$
   c) $(g)'$
4) A string is a Boolean formula iff it is obtained by finitely many application of rules 1, 2, 3

There is a theorem that says how to write a formula tha corespond to a function. I will not state the theorem but I will only give an example in the case of 2 variables functions. Given the function $F_2(X_1, X_2)$ the formule can be written as:

$$f(x_1, x_2) = F_2(0,0)\, x_1' x_2' + F_2(0,1) X_1' x_2 + F_2(1,0) X_1 x_2' + F_2(1,1) x_1 x_2$$

So basically we multiply the value of the function, computed for a certain combination of the variabels' values, by a monomiel. The monomial is obtained as multiplication of all the variables where a variable is negated if it was considered zero in the computation of the function.

A function represented in this was is said to be in the <u>minterm canonical form</u>

For instance consider the full adder example:

$$s = S(0,0)\,a'b' + S(0,1)\,a'b + S(1,0)\,ab' + S(1,1)\,ab =$$
$$= a'b + ab'$$

$$c = C(0,0)\,a'b' + C(0,1)\,a'b + C(1,0)\,ab' + C(1,1)\,ab =$$
$$= ab$$

To understand better, looking at the function s we can say that:

$$\left.\begin{array}{l}\text{not } a \text{ and } b \Rightarrow s \\ a \text{ and not } b \Rightarrow s\end{array}\right\} \text{either one so is the union}$$

So we write

$$s = a'b + ab'$$

Only combinations of inputs that lead to 1 are present in the formula but this is obvious because in all other cases it is zero, meaning $s' = (a'b + ab')'$ meaning all combinations of inputs that lead to zero.

Unfortunately a Boolean function can be represented by many Boolean formulas. For instance consider the function:

a	b	c	F(a,b,c)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

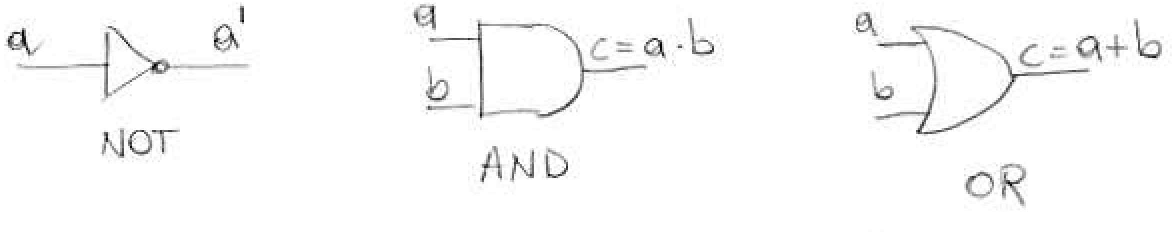
$$f(a,b) = a'bc + ab'c + abc'$$

or

$$f(a,b) = b(a'c + ac') + ab'c$$

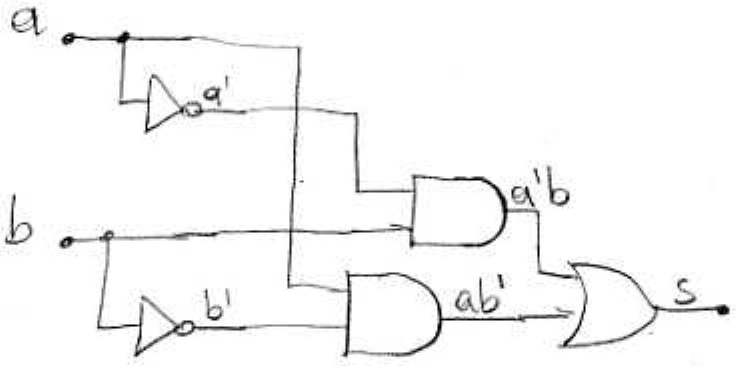
the same function is represented by two different formulas!!

Electrical engineers have introduced symbols to represent logic operations:

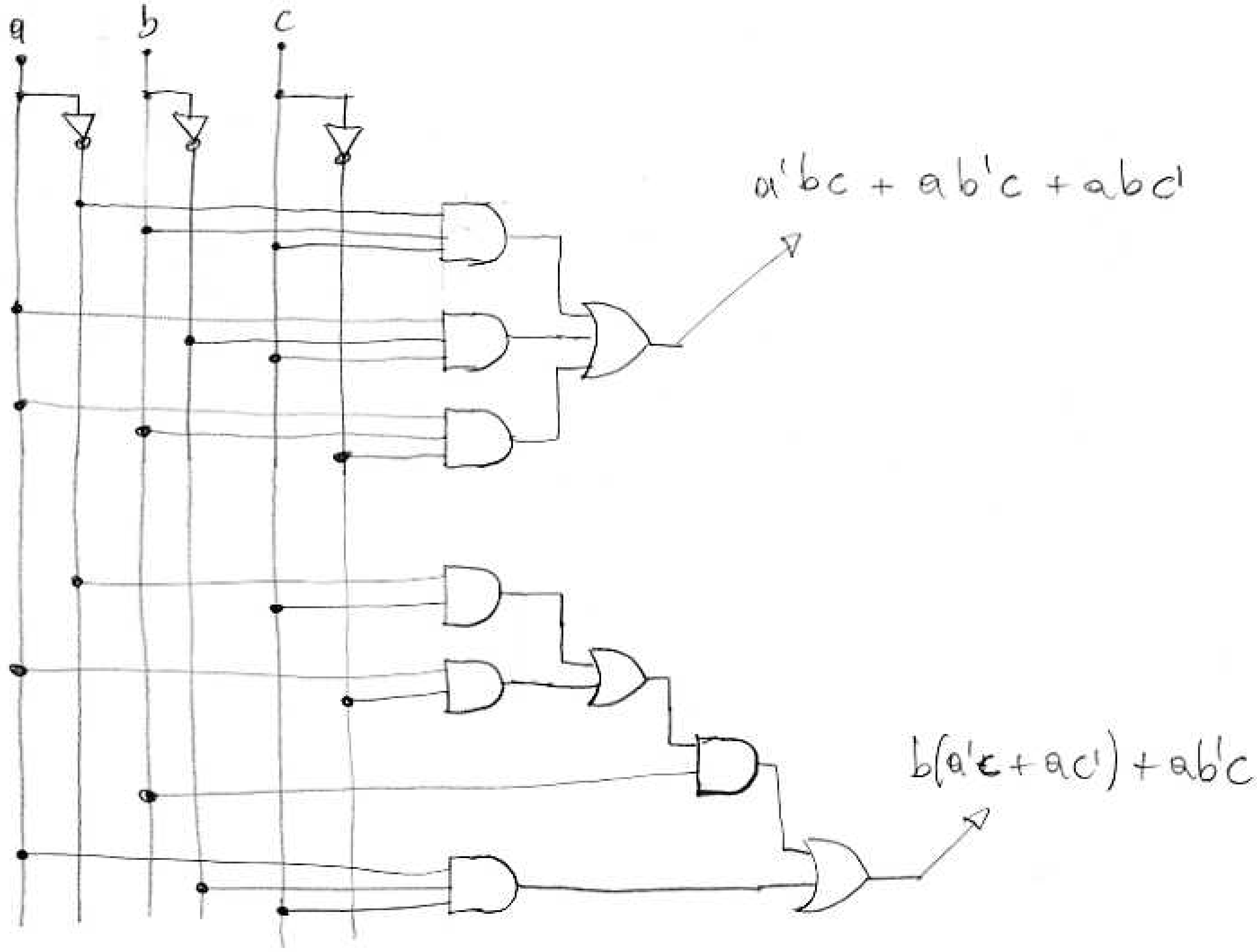


Using these graphical notation we can represent a Boolean formula and hence a Boolean function:

For instance let's represent the sum S:



Since a function can be represented by more than one formules, then also the graphical representation is not unique. For instance let's represent F(a, b, c):



The two graphical representation are very different. Not only the number of symbols (gates) that we used are different but, for instance, the first representation has only two levels (imagine to levelize the graphical representation) while the second one has 4 levels.

If we have a way of implementing the three basic blocks OR, AND, NOT using transistors, then we can implement any Boolean function with transistors.
This is because any Boolean function has a minterm canonical representation and such representation alway have a 2-level implementation in terms of OR, AND, NOT.
The question is whether we need all three operators or if we can use less operators by still being able to represent all possible Boolean function.

De Morgan Laws

1) $(x+y)' = x' \cdot y'$ (we have seen this already in the introduction to logic)

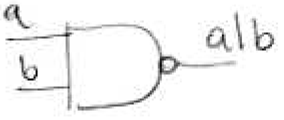
2) $(xy)' = x' + y'$

We notice that 1) also means:

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

but then we can replace the + operator using a combination of $\cdot$ an $'$.
So using the set of operators $\{\cdot, '\}$ is still sufficient to represent all logic functions.
Let's go even further. We introduce the NAND operator which we denote by $|$. It is described as follow:

a	b	a|b
0	0	1
0	1	1
1	0	1
1	1	0



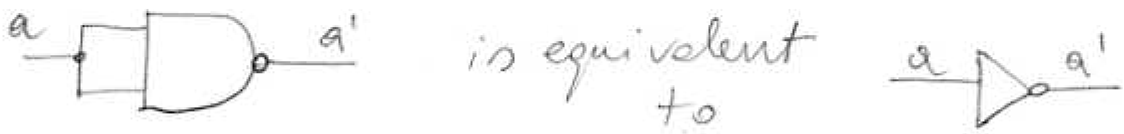
NAND

Consider now $a=b$, then we have that

$$a|a = \begin{cases} 0 & \text{if } a=1 \\ 1 & \text{if } a=0 \end{cases}$$

so it means that $a|a = a'$

and graphically that:



So we can implement NOT using NAND.

We also notice that $a|b = (ab)'$

and hence:

or using formulas $(a|b)|(a|b) = ab$

So we can implement AND using NAND.

Then the two operators $\{\cdot, '\}$ can be replaced by the only operator $\{|\}$

It means that if we know how to implement a NAND operator we can implement any Boolean function.
