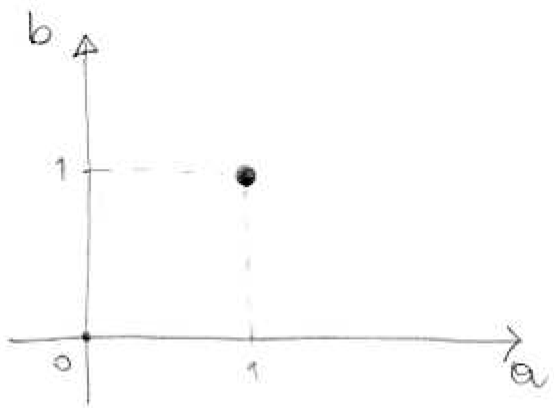
# LOGIC MINIMIZATION

We now introduce a way of representing Boolean functions that will help us in understanding logic minimization.
Consider the two variables function:

| $a$ | $b$ | $f(a,b)$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |



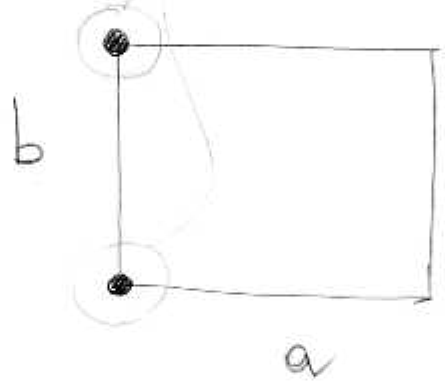

As shown on the right, we can represent the function on a plane whose coordinate are the Boolean variables. Variables can only be 0 or 1 because they are Boolean.
In this representation we use a dot to say that, for that input combination, the function evaluates to 1.
For instance, consider the following function:

| a | b | f |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

Minterm canonical form:

$f = a'b' + a'b$

cube representation




This function can be simplified:

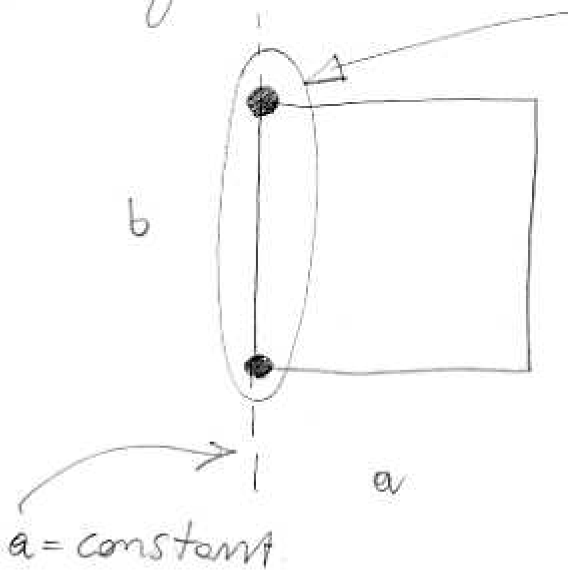
$f = a'b' + a'b = a'(b'+b) = a'$

because $b+b'=1$ (it is one of the postulates of a ~~Boolean~~ algebra).

If we look at the function (I mean the table) we note that ther are two rows for which one variable is constant, the other changes and the function is equal to 1. Of course, both minterms are going to appear in the formula representing f and since one variable is constant we can use associativity an put the variable in evidence.

The other variable changes so it is going to appear as the sum of a variable and its negation which is equal to 1.

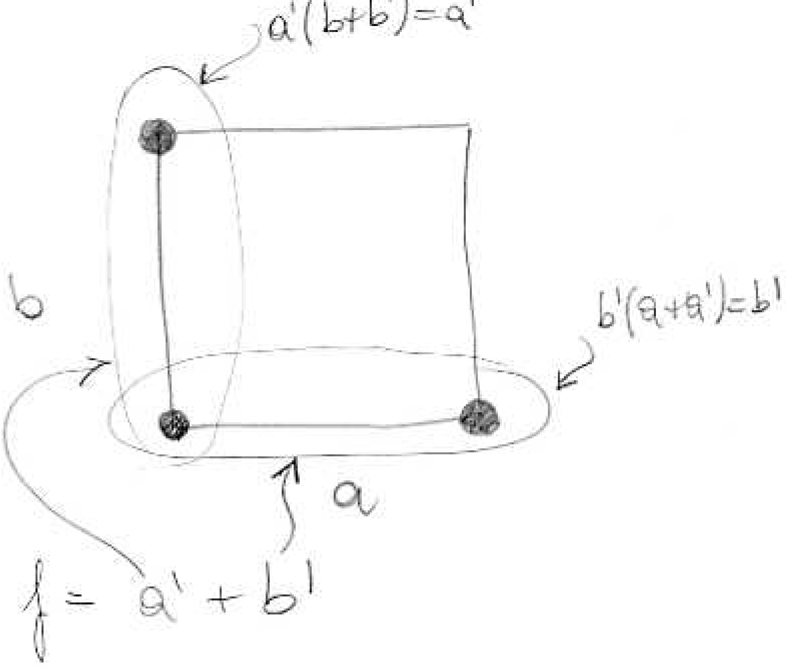
Using the cube representation:




If we group points along the line a = constant then the term corresponding to that group is equal just to $a'$ ($a'$ because the line is $a=0$).

For instance;

| a | b | f |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

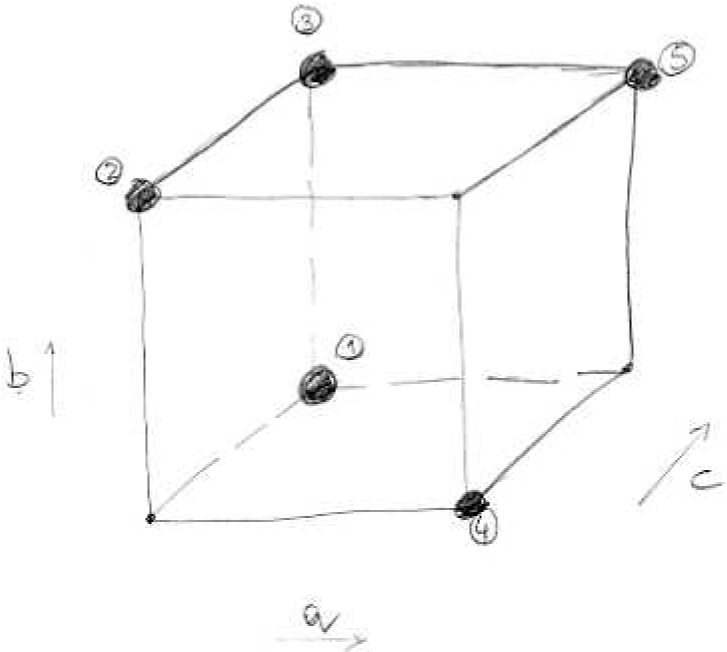


The function is in fact a nand function $(a \cdot b)' = a' + b'$.

What if we have three variables?

We can use a cube instead of a square:

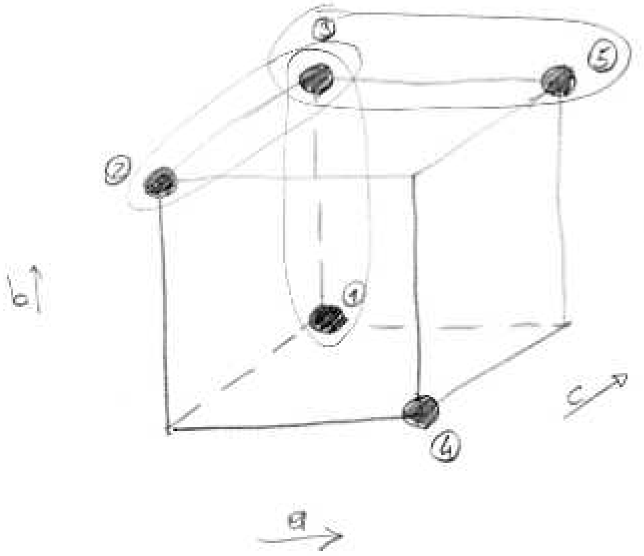
| | $a$ | $b$ | $c$ | $f$ |
|---|---|---|---|---|
| | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 |
| 2 | 0 | 1 | 0 | 1 |
| 3 | 0 | 1 | 1 | 1 |
| 4 | 1 | 0 | 0 | 1 |
| | 1 | 0 | 1 | 0 |
| | 1 | 1 | 0 | 0 |
| 5 | 1 | 1 | 1 | 1 |




Minterm canonical form:

$$f = \underset{1}{a'b'c} + \underset{2}{a'bc'} + \underset{3}{a'bc} + \underset{4}{ab'c'} + \underset{5}{abc}$$

In our function there are adjacent points that we can group to eliminate variables:



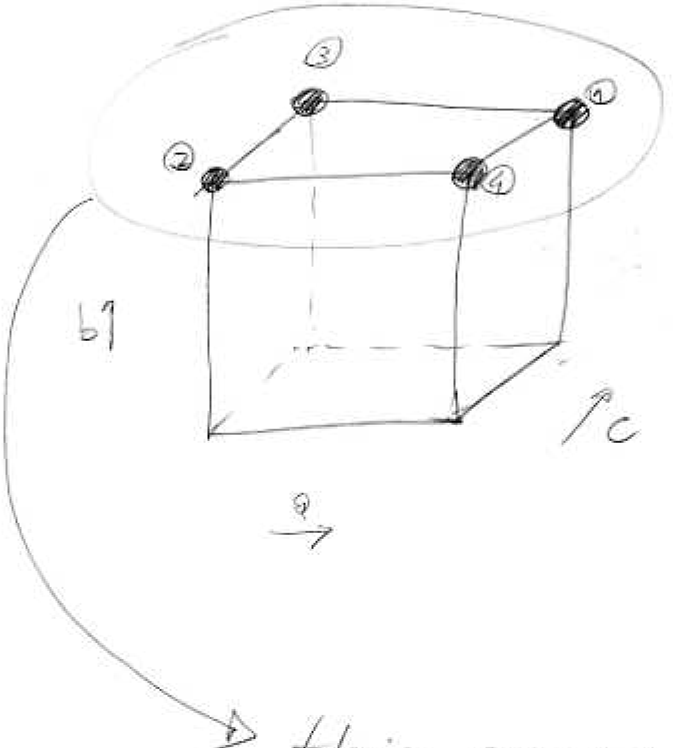

$$f = \underbrace{a'b}_{2+3} + \underbrace{a'c}_{1+3} + \overbrace{bc}^{3+5} + \underbrace{ab'c'}_{4}$$

This formula is "simpler" than the minterm canonical form. It is then clear that grouping minterms is beneficial from complexity point of view.

If a function has $n$ variables then a minterm is the "and" of $n$ variables. Now in order for one of them to disappear we need the sum of two minterm where that only variable changes. Meaning:

$$x_1 \cdots x_j \cdots x_n + x_1 \cdots x_j' \cdots x_n = x_1 \cdots x_{j-1} x_{j+1} \cdots x_n \overbrace{(x_j + x_j')}^{1}$$

where $x_1, \dots, x_n$ are our boolean variables.
If we want two variables to disappear then we need $n-2$ variables to be fixed and the remaining 2 have to change in all possible combinations.
So basically in order to drop two variables we should be able to group 4 adjacient points.
Note that grouping 3 points doesn't give use any benefit.
For instance:




$$f = \underset{①}{bac} + \underset{②}{ba'c'} + \underset{③}{ba'c} + \underset{④}{bac'}$$

$$= b(\underset{②+③}{a'c' + a'c}) + b(\underset{①+④}{ac + ac'})$$

$$= ba'(c+c') + b(a(c+c')) =$$

$$= ba' + ba = b(a+a') = \underline{\underline{b}}$$

this group is a plane where $b$ is constant and $a$ and $c$ change in all possible combination so $f = b$.

We want to find a way of grouping points in such a way that the resulting formula has the least number of terms an variables.
Note that, for the previous function, the minterm canonical form has 4 terms each containing 3 variables (the 4 points in the cube representation) while the formula we found contains only one term with one variable.
Also you can check that the function $f$ is:

| a | b | c | f | |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | |
| 0 | 0 | 1 | 0 | |
| 0 | 1 | 0 | 1 | 2 |
| 0 | 1 | 1 | 1 | 3 |
| 1 | 0 | 0 | 0 | |
| 1 | 0 | 1 | 0 | |
| 1 | 1 | 0 | 1 | 4 |
| 1 | 1 | 1 | 1 | 1 |

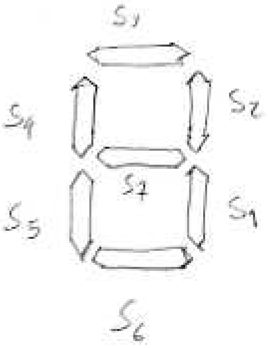
and $f$ is equal to the second column which is $b$, hence $f = b$

# Queen McCluskey algorithm

this algorithm finds the minimum formula representation of a Boolean function when the cost of a formula is the number of terms and/or the number of variables instances in the formula.

Intuitively, the algorithm start from the function points (in the cube representation) and tries to expand the point along all possible axes finding all groups of two points. The recursively the Q.M. algorithm looks for groups of 4 points, then 8 points and so on. At the end it takes the minimum number of groups covering all points in the function. The algorithm is best explained with an example. For instance:

Consider a seven segment display.
It is a device which has 7 LED's laied out as follows




Depending on which LED you switch on, this device displays a different number.

For instance, if $S_3, S_2, S_7, S_1, S_6$ are switched on then it displays mumer three (of course all other LED's have to be off.)
We can write a logic function for each segment to display numbers. If we consider 3 inputs, we can display from 0 to 7. The Boolean function for $S_3$ for instanse is:

| a | b | c | $S_3$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

(Input number in binary format is abc)

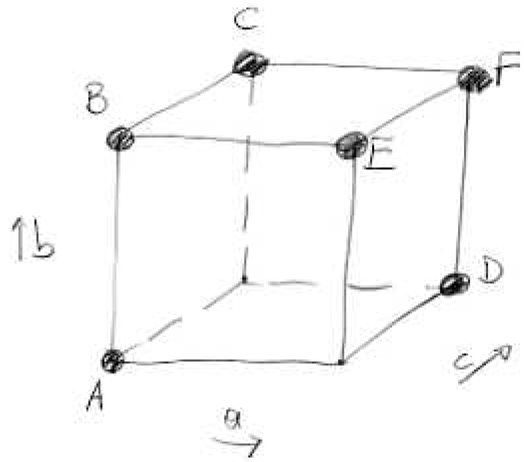
To understand the table consider the following two examples

| | | |
|---|---|---|
| number | 4 | 7 |
| binary representation | 1 0 0 | 111 |
| display | 4 | 7 |
| S3 | 0 | 1 |

First step) Consider all the rows for which the function evaluates to 1

| | | | |
|---|---|---|---|
| 0 | 0 | 0 | A |
| 0 | 1 | 0 | B |
| 0 | 1 | 1 | C |
| 1 | 0 | 1 | D |
| 1 | 1 | 0 | E |
| 1 | 1 | 1 | F |

← lables



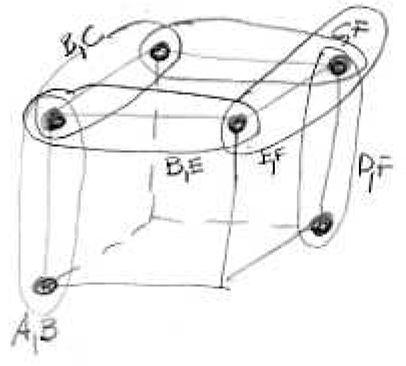

Second step) During this step the Q.M. algorithm trys to find all possible pairs of adjacient points. When a pair is found, the variable that changes is substituted with the symbol – . Also we mark the two points that originated the new group, to remember that the new group actually covers the previous two points. In order to do this, we look at all possible pair of point. If only one variable changes between the two minterms then we generate a new element substituting a dash to that variable:

| | | | | |
|---|---|---|---|---|
| A | 0 | 0 | 0 | ✓ |
| B | 0 | 1 | 0 | ✓ |
| C | 0 | 1 | 1 | ✓ |
| D | 1 | 0 | 1 | ✓ |
| E | 1 | 1 | 0 | ✓ |
| F | 1 | 1 | 1 | ✓ |

mark

$\longrightarrow$

| | | | |
|---|---|---|---|
| A,B | 0 | – | 0 |
| B,C | 0 | 1 | – |
| B,E | – | 1 | 0 |
| C,F | – | 1 | 1 |
| D,F | 1 | – | 1 |
| E,F | 1 | 1 | – |




Whe have basically found all groups of 2 adjacient points.

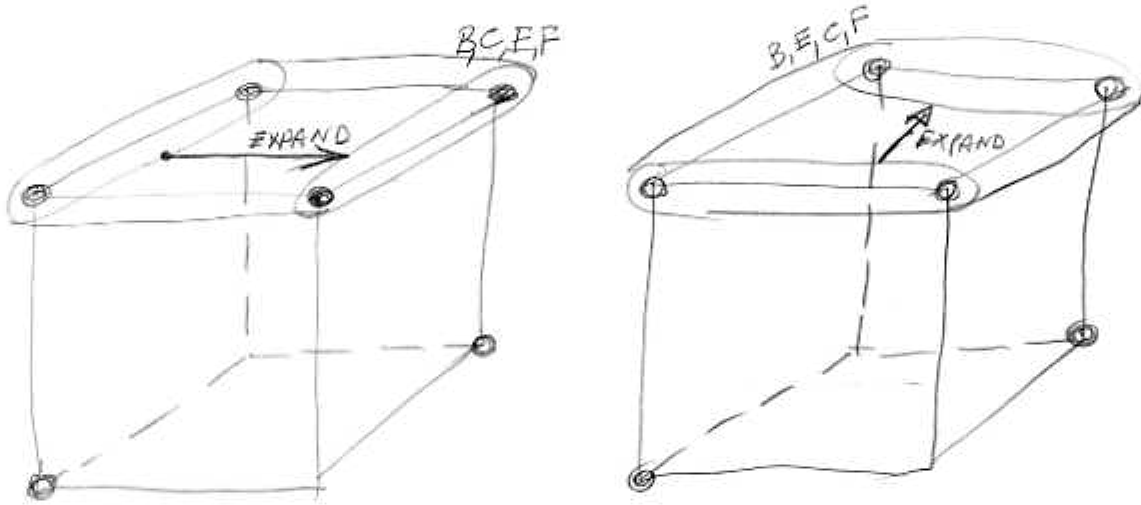
Third step) Now we want to find all possible groups of 4 points. We then look at all elements generated in the second step. We group them in pairs if they have a dash in the same position and if, among all other variables, only one changes:

| | | | | |
|---|---|---|---|---|
| A, B | 0 | – | 0 | |
| B, C | 0 | 1 | – | ✓ |
| B, E | – | 1 | 0 | ✓ |
| C, F | – | 1 | 1 | ✓ → |
| D, F | 1 | – | 1 | |
| E, F | 1 | 1 | – | ✓ |

| | | | |
|---|---|---|---|
| B, C, E, F | – | 1 | – |
| B, E, C, F | – | 1 | – |

Now we have only one term so the algorithm cannot go ahead. In general it should be continued untill no more pairs are possible (so after this step, look for groups of 8 points, 16 points and so on)

What we have done in step three is to expand 2 points group into 4 points groups:




The terms that are not marked are:

| | | |
|---|---|---|
| A,B | 0 — 0 | P1 |
| D,F | 1 — 1 | P2 |
| B,C,E,F, | — 1 — | P3 |

these are called <u>prime implicants</u>

The last step) Requires to build a table whose rows are the minterms and whose columns are the prime implicants

An entry $(i, j)$ in the table is marked if minterm $i$ is covered by prime implicant $j$.

In our case:

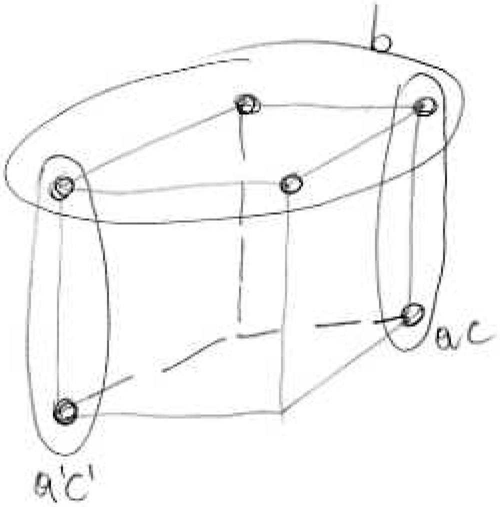
| | $P_1$ | $P_2$ | $P_3$ |
|---|---|---|---|
| A | X | | |
| B | X | | X |
| C | | | X |
| D | | X | |
| E | | | X |
| F | | X | X |

Now we have to find the minimum number of columns such that there is an x in every row. This problem is called unate covering and it is an NP-complete problem.

In our case we need all three prime implicants to cover the minterms so the minimum formula for S3 is:

$$S3 = \underset{P1}{a'c'} + \underset{P2}{ac} + \underset{P3}{b}$$

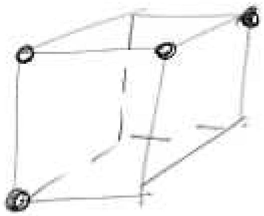
which represents the following groups:




---

As an example of the last step where some prime implicants may disappear consider the function:

| a | b | c | f |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |



| | | | |
|---|---|---|---|
| A | 0 | 0 | 0 |
| B | 0 | 1 | 0 |
| C | 1 | 1 | 0 |
| D | 1 | 1 | 1 |

$\longrightarrow$

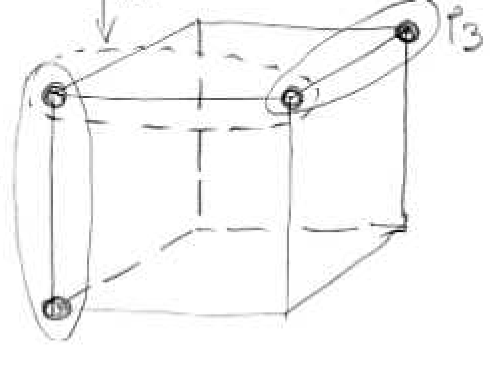
| | | | | |
|---|---|---|---|---|
| A, B | 0 | — | 0 | $P_1$ |
| B, C | — | 1 | 0 | $P_2$ |
| C, D | 1 | 1 | — | $P_3$ |

no other pair are possible (basically ther are no 4 points groups)

| | $P_1$ | $P_2$ | $P_3$ |
|---|---|---|---|
| A | X | | |
| B | X | X | |
| C | | X | X |
| D | | | X |

In this case $P_1$ an $P_3$ cover all the minterms an $P_2$ is not needed:

$$f = a'c' + ab$$



# Karnaugh maps

It is a 2 dimensional representation of a Boolean function.

For a two variables Boolean function is basically equivalent to the cube representation:

NAND

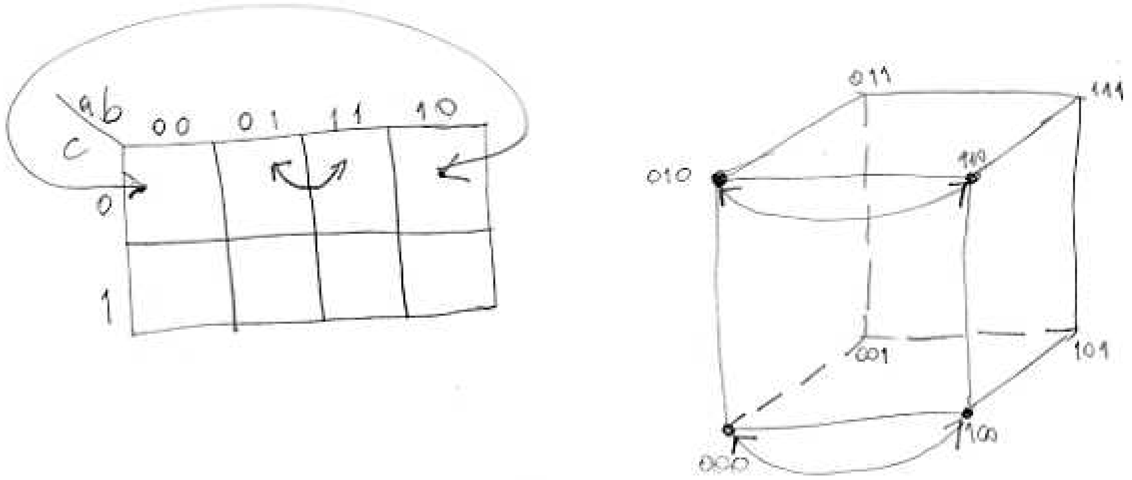
| b \ a | 0 | 1 |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 1 | 0 |

$b'$

$a'$

$$f = a' + b' = (ab)'$$

In the case of 3 variables than whe can put 2 variables on the columns an 1 on the rows:



Note that only one variable changes from one cell to the other and the reason can be understood by looking at the cube representation. All cells in the K-map are now adjacient!!!

Now we can use K-maps to group adjacient ones an simplify Boolean functions.

For S3, for instance:

| c \ ab | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 |

only b is constat, both a an c change:

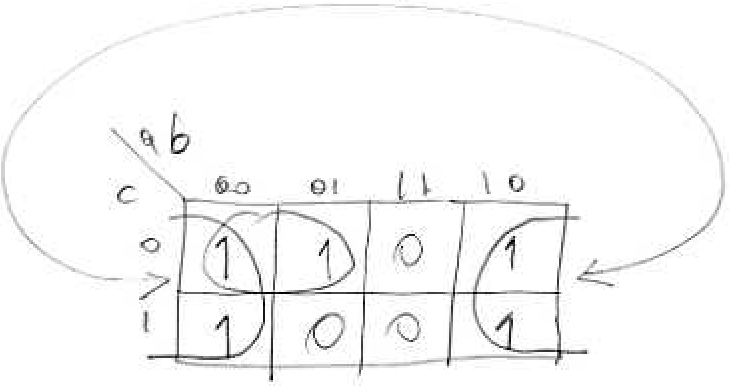
b

$a'c'$

c and a are constant while b changes so it will be dropped and this term is $ac$ (because their values is 1)

$$f = ac + a'c' + b$$

In K-maps then we have to group ones in as bigger groups as possible. Remember that the number of ones in a group has to be a power of 2 (2, 4, 8, 16 ---). Also consider the fact that K-maps are basically folded, so:




$$f = b' + a'c'$$