BOOLEAN ALGEBRA EXAMPLE 1

Simplification from a truth table

Inputs			Output
A	B	C	D
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

From this truth table we can write a Boolean expression.

Step 1: Look at the values of inputs when the **output is 1**

A `0' input means NOT x and an input of `1' means x where x is the letter from the input.

SO

¬A ^ B ^ ¬C	¬AB¬C	**Notice that we can either use the signs V (or) or + interchangeably. And multiplying is the same as ^ (and)**
¬A ^ B ^ C	¬ABC	
A ^ B ^ ¬C	**AB**¬C	
A ^ B ^ C	**ABC**	

Step 2: So the sum is ¬AB¬C + ¬ABC + **AB**¬C + **ABC** **(now simplify with Boolean algebra techniques) or we can write this as** (¬A ^ B ^ ¬C) V (¬A ^ B ^ C) V (**A ^ B ^** ¬C) V (**A ^ B ^ C)**

Factor out ¬AB so ¬AB(¬C + C) + Factor out AB so AB(¬C + C)

Since anything that is not itself or itself is always 1 it eliminates: ¬AB + AB

Factor out B so B(¬A or A)

Since anything that is not itself or itself is always 1 it eliminates: B

Hence the entire truth table is equal to the expression: B

KARNAUGH MAP EXAMPLE 1

A Karnaugh map is similar to a truth table in the fact it as a box input for every possible input. However, the karnaugh map the column headings only change by 1 bit.

e.g. for the same logic algorithm:

Notice that the column heading goes 00, 01, 11, 10. This pattern is used to ensure that we are only changing 1 bit when looking across a period. So for (c1,r1) A = 0 B = 0 C= 0 but for (C2, r1) A = 0 still & B = 0 still but C = 1 (only 1 bit changes) etc. Similarly, if we move down a row for a particular column, it is only A that changes.

The first variable (A) goes in the left row heading, and both of the inputs are combined in an AND (^) expressions, BC.

A \ BC	00	01	11	10
0				
1				

We place the outputs (from previous truth table) in the corresponding fields.

A \ B C	00	01	11	10
0	0	0	1	1
1	0	0	1	1

I have colour coded them so you can see how they match up.

<u>Using a karnaugh map to determine the simplest expression</u>

Step 1: Build Karnaugh map
Step 2: Draw a square or a rectangle to combine outputs of 1 (these can be called "minterms")
Here are the rules for doing this.....

2) Combine minterms (where minterm = 1) into groups
   a. Group size is a power of 2 (i.e., 1, 2, 4, 8, 16...)
   b. Group shape is square or rectangle
   c. Make groups as large as possible
   d. Groups can overlap
   e. Groups can stretch around boundaries
   f. All boxes where minterm = 1 need to be in a group

<table>
<tr><td>Rule</td><td>Explanation</td></tr>
<tr><td>A</td><td>e.g. We can't have a group with 3 minterms</td></tr>
<tr><td>B</td><td>Must look like a square or rectangle (encompasses only minterms)</td></tr>
<tr><td>C</td><td>Make the groups as large as they can (without crossing other 0's</td></tr>
<tr><td>D</td><td>This means multiple groups may form e.g. an L shape
1 0
1 1</td></tr>
<tr><td>E</td><td>This means that a column at the side is really adjacent to the column on the other side.
E.g. just remember to move the correct column headings
1 0 0 1 1 = 0 0 1 1 1
1 0 0 1 1   0 0 1 1 1</td></tr>
<tr><td>F</td><td>Every box with a minterm (1) must belong to a group.</td></tr>
</table>

So in our example

A \ B C	00	01	11	10
0	0	0	1	1
1	0	0	1	1

Step 3: Determine the common value of the group. This means what variable has the same value in all of the boxes in the group. In our case B is always = 1 but A and C vary in the group. So the value is B =1

In our example, there is just one group but it is possible to have multiple groups and so multiple of these expressions.

Step 4: Determine an expression for each group: we look for the variable and if it equals 1 then the expression is just that variable but if the variable is equal to 0 then it is NOT that variable. Say if B was always equal to zero it would be NOT B.

In our case B = 1   so expression is **B**

Step 5: Do the same for all the other groups

Step 6: Add all the expressions together and simplify.

KARNAUGH MAP EXAMPLE 2

From this truth table, determine the simplest possible Karnaugh map and write an expression for the logic operation.

A	B	C	OUT
0	0	0	0
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

ANSWER

A \ BC	00	01	11	10
0	0	0	0	1
1	1	0	1	1

Notice that we cannot form a box with 3 minterms (1's in) as we can only have powers of 2 (1,2,4,8…) This means to encompass all whiel using a minimum number of boxes we need 3 boxes.

Group value	Expression (1 = term and 0 = not term)
Group gold: Common value is B= 1 AND C =1	**B ^ C** (can be written as BC)
Group red: common value is **A = 1 AND B = 1**	**A ^ C** (can be written as AC)
Green box: common value is **A = 1 AND C = 0**	**A ^ ¬C** (can be written as A¬C)

Summing the expressions (i.e. add means or)

So….   (B ^ C) V (A ^ B)   V (A ^ ¬C)

B ^ (C V A)   V (A ^ ¬C)

This is already in its simplest form?

Follow the link to a good tutorial: https://www.youtube.com/watch?v=-gQ6kN7qnmA

Recapping some simple rules

(A ^ B) V ¬ C
Order of precedence: NOT, AND then OR

A ^ ¬A = 0
A V ¬A = 1