Derivation of a Boolean Expression

When designing a logical circuit, the designer works from two sets of known values:

1. The various states that the inputs to the circuit can take,
2. The desired outputs for each input condition.

The logical expression is derived from these sets of values.

For example, let us say that we wish to build a circuit that behaves in the way defined by the following truth table:

For each row where the desired output is a one, we write the product term which would evaluate to give us a 1, like this:

	Inputs		Output	
X	Y	Z	A	
0	0	0	1	/X./Y./Z
0	0	1	0	
0	1	0	1	/X.Y./Z
0	1	1	0	
1	0	0	1	X./Y./Z
1	0	1	0	
1	1	0	1	X.Y./Z
1	1	1	0	

The desired expression is the sum of these products:

/X./Y./Z + /X.Y./Z + X./Y./Z + X.Y./Z   =   A

This expression could be simplified as follows:

Look for two subexpressions that share common factors, except for one term. You want to isolate that term.

Remove the common factor of /X from the first two subexpressions.

/X.(/Y./Z + Y./Z) + X./Y./Z + X.Y./Z   =   A

Do the same thing with /Z

/X . /Z . (/Y + Y) + X./Y./Z + X.Y./Z   =   A

(/Y + Y) will always evaluate to 1, no matter what value Y has, so /X . /Z . 1 becomes /X./Z

/X./Z   + X./Y./Z + X.Y./Z   =   A

Perform a similar trick on the two remaining subexpressions to give:

/X./Z   +   X./Z = A

Remove the common factor of /Z:

/Z . (/X + X)   = A

/X + X cancels out to give:

/Z = A

Which you might have been able to figure out by looking at the truth table carefully.