

Karnaugh Maps

Simplifying Boolean expressions systematically

Simplifying Boolean Expressions

- We've seen why it's advantageous to simplify Boolean expressions
- But doing that manually (using the Boolean laws) is not a very systematic process
- There are **algorithms** that are guaranteed to produce the simplest possible expression
- We will look at a particular one here

Karnaugh Maps (K-Maps)

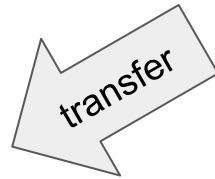
- Systematic manual method for simplification
- Best for at most 6 variables
- Will produce the simplest Sums-Of-Products
- Similar to a truth table – maps all combinations

Karnaugh Maps (K-Maps)

3-Variable K-Map:

XY \ Z	Z	
	0	1
00	0	0
01	1	1
11	1	1
10	1	0

Note: 11
before 10!



X	Y	Z	$F(X,Y,Z)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

How do K-Maps help us simplify?

A common form of simplification in Boolean algebra is

$$A\overline{B} + AB = A(\overline{B} + B) = A$$

Also works for 3 variables:

$$A\overline{B}C + ABC = AC$$

Both are independent of B !

K-Maps are a visual way for finding independent terms.

3-variable K-Map

		Z	
		0	1
XY	00	0	0
	01	1	1
	11	1	1
	10	1	0

Goal: Find **groups of ones**!

- Groups must be as large as possible
- Group size must be power of 2
- Contain only ones, no zeros
- All ones must be in at least one group
- Groups can overlap

3-variable K-Map

XY \ Z	Z	
	0	1
00	0	0
01	1	1
11	1	1
10	1	0

X and Z are irrelevant (group covers both 0 and 1 for these variables).
Y needs to be 1 for this group.

$$\overline{X}Y\overline{Z} + \overline{X}YZ + XY\overline{Z} + XYZ = Y$$

Y is irrelevant (group covers both 0 and 1 for Y).
X needs to be 1, Z needs to be 0 for this group.

$$XY\overline{Z} + X\overline{Y}\overline{Z} = X\overline{Z}$$

Result: $F(X, Y, Z) = X\overline{Z} + Y$

K-map rules

- Groups can't contain 0

XY \ Z		0	
00		0	0
01		1	1
11		1	1
10		1	0

Correct

XY \ Z		0	
00		0	0
01		1	1
11		1	1
10		1	0

Wrong

K-map rules

- Groups must be rectangular (can't be e.g. diagonal)

XY \ Z		0	1
		0	1
00	0	0	0
01	0	0	1
11	1	1	1
10	0	0	0

Correct

XY \ Z		0	1
		0	1
00	0	0	0
01	0	0	1
11	1	1	1
10	0	0	0

Wrong

K-map rules

- Group size is power of 2

XY \ Z		Z	
		0	1
00		0	0
01		1	1
11		1	1
10		1	0

Correct

XY \ Z		Z	
		0	1
00		0	0
01		1	1
11		1	1
10		1	0

Wrong

K-map rules

- Each group is as large as possible
- Groups may overlap

XY \ Z		Z	
		0	1
00		0	0
01		1	1
11		1	1
10		1	0

Correct

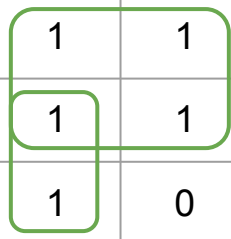
XY \ Z		Z	
		0	1
00		0	0
01		1	1
11		1	1
10		1	0

Wrong

K-map rules

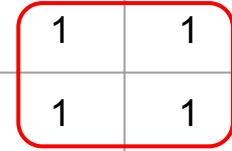
- Each 1 must be in at least one group

XY \ Z		Z	
		0	1
00		0	0
01		1	1
11		1	1
10		1	0

The K-map shows three green groupings: a 2x2 square covering cells (01,0), (01,1), (11,0), and (11,1); a 2x1 vertical rectangle covering cells (11,0) and (10,0); and a 1x1 square covering cell (10,0). All four 1s in the map are enclosed by at least one green line.

Correct

XY \ Z		Z	
		0	1
00		0	0
01		1	1
11		1	1
10		1	0

The K-map shows a single red grouping: a 2x2 square covering cells (01,0), (01,1), (11,0), and (11,1). The 1s at (10,0) and (10,1) are not enclosed by any red line.

Wrong

K-map rules

- Groups may wrap around

XY \ Z		0	1
		0	1
00	1	1	
01	0	0	
11	0	0	
10	1	1	

Correct

XY \ Z		0	1
		0	1
00	1	1	
01	0	0	
11	0	0	
10	1	1	

Wrong

K-Maps with more variables

Same idea, but more variables on the axes:

AB \ CD				
	00	01	11	10
00				
01				
11				
10				

AB \ CDE								
	000	001	011	010	110	111	101	100
00								
01								
11								
10								

Summary

- K-Maps are a visual method for simplifying Boolean functions
- They produce *minimal* sum-of-product representations
- Work well for small functions (up to 6 inputs)
- Automatic approaches exist for optimising large functions

EOF