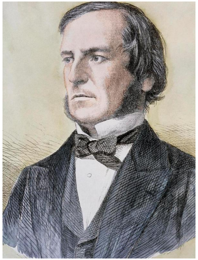
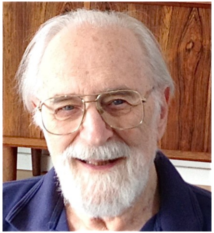
Informatics 1 – Introduction to Computation
Computation and Logic

Julian Bradfield
based on materials by
Michael P. Fourman

Karnaugh Maps



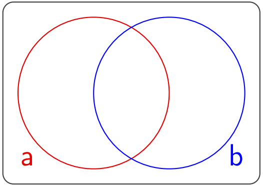
George Boole,
1815–1864



Maurice Karnaugh,
1924–2022

How many distinguishable universes?

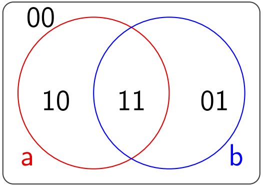
Suppose we have two predicates. How many different true/false combinations are there?



So how many universes can we distinguish with two predicates?

How many distinguishable universes?

Suppose we have two predicates. How many different true/false combinations are there?

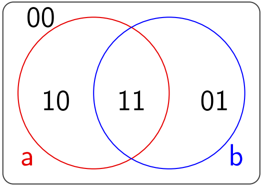


$$2^2 = 4$$

So how many universes can we distinguish with two predicates?

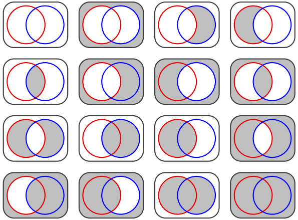
How many distinguishable universes?

Suppose we have two predicates. How many different true/false combinations are there?



$$2^2 = 4$$

So how many universes can we distinguish with two predicates?



$$2^4 = 16 = 2^{2^2}$$

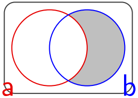
Lots of distinguishable universes

3 predicates, $2^3 = 8$ regions, $2^8 = 256$ different universes

4 predicates, $2^4 = 16$ regions, $2^{16} = 65536$ different universes

How can we characterize a universe?

By saying which regions (i.e. which boolean combinations of predicates) are inhabited/empty.

Consider 

Its inhabited regions are 00,10,11, so it is described by

$$(\neg a \wedge \neg b) \vee (a \wedge \neg b) \vee (a \wedge b)$$

$$\neg b \vee (a \wedge b)$$

$$\neg(\neg a \wedge b)$$

$$a \vee \neg b$$

Binary not-SI prefixes:
Ki (Kibi) 1024 (2^{10})
Mi (Mebi) 1048576 (2^{20})
Gi (Gibi) 1073741824 (2^{30})
etc.
$65536 = 64$ Ki

From Truth Tables to Karnaugh Maps

There are algorithmic ways to simplify boolean formulae.

But Karnaugh Maps are a human way, exploiting our pattern-matching abilities.

We start with tables of values:

What formula describes

$$\begin{array}{cc|cc} & & \multicolumn{2}{c}{b} \\ & & 0 & 1 \\ \hline & 0 & 1 & 0 \\ a & 1 & 1 & 1 \end{array}$$

?

From Truth Tables to Karnaugh Maps

There are algorithmic ways to simplify boolean formulae.

But Karnaugh Maps are a human way, exploiting our pattern-matching abilities.

We start with tables of values:

What formula describes

$a \backslash b$	0	1
0	1	0
1	1	1

?

Highlight the 1 cells and look for the largest **even rectangles** that cover them.

From Truth Tables to Karnaugh Maps

There are algorithmic ways to simplify boolean formulae.

But Karnaugh Maps are a human way, exploiting our pattern-matching abilities.

We start with tables of values:

What formula describes

a \ b	0	1
0	1	0
1	1	1

?

Highlight the 1 cells and look for the largest **even rectangles** that cover them.

From Truth Tables to Karnaugh Maps

There are algorithmic ways to simplify boolean formulae.

But Karnaugh Maps are a human way, exploiting our pattern-matching abilities.

We start with tables of values:

What formula describes

a \ b	0	1
0	1	0
1	1	1

?

Highlight the 1 cells and look for the largest **even rectangles** that cover them.

From Truth Tables to Karnaugh Maps

There are algorithmic ways to simplify boolean formulae.

But Karnaugh Maps are a human way, exploiting our pattern-matching abilities.

We start with tables of values:

What formula describes

		b	
		0	1
a	0	1	0
	1	1	1

?

Highlight the 1 cells and look for the largest **even rectangles** that cover them. Thus we see

$$a \vee \neg b$$

(By *even rectangle*, we mean rectangles with width and height powers of two.)

Four-variable Maps

Things get interesting when we have three or four variables. Write tables so:

$$\begin{array}{cc|cccc} & & \multicolumn{4}{c}{cd} \\ & & 00 & 01 & 11 & 10 \\ \hline & 00 & & & & \\ & 01 & & & & \\ ab & 11 & & & & \\ & 10 & & & & \end{array}$$

This order of values is a Gray code. The point is that only one variable changes as you go one step up/down/left/right.

Four-variable Maps

Things get interesting when we have three or four variables. Write tables so:

ab \ cd	00	01	11	10
00				
01				
11				
10				

This order of values is a Gray code. The point is that only one variable changes as you go one step up/down/left/right.

Four-variable Maps

Things get interesting when we have three or four variables. Write tables so:

ab \ cd	00	01	11	10
00				
01				
11				
10				

This order of values is a Gray code. The point is that only one variable changes as you go one step up/down/left/right.

Four-variable Maps

Things get interesting when we have three or four variables. Write tables so:

		cd			
		00	01	11	10
	00				
ab	01				
	11				
	10				

This order of values is a Gray code. The point is that only one variable changes as you go one step up/down/left/right.

Four-variable Maps

Things get interesting when we have three or four variables. Write tables so:

		cd			
		00	01	11	10
	00				
	01				
ab	11				
	10				

This order of values is a Gray code. The point is that only one variable changes as you go one step up/down/left/right.

The order of entries means that the 1-values of each variable occupy adjacent rows (c, d) or columns (a, b). What about the 0-values?

The Karnaugh Map as a torus

ab \ cd	00	01	11	10
00				
01				
11				
10				

The Karnaugh Map as a torus

cd

ab	00	01	11	10
00				
01				
11				
10				

The Karnaugh Map as a torus

		cd			
		00	01	11	10
ab	00				
	01				
	11				
	10				

The Karnaugh Map as a torus

ab \ cd	00	01	11	10
00				
01				
11				
10				

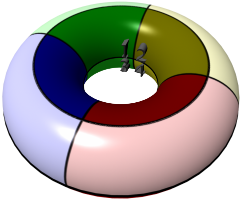
The Karnaugh Map as a torus

		cd			
		00	01	11	10
ab	00				
	01				
	11				
	10				

The Karnaugh Map as a torus

		cd			
		00	01	11	10
	00				
ab	01				
	11				
	10				

The 0-values for each variable occupy adjacent rows/columns *if we view the table as wrapping round bottom to top and right to left*.



via pngwing.com

KM example 1

			cd		
		00	01	11	10
ab	00	0	0	0	0
	01	0	1	1	0
	11	0	1	1	0
	10	0	0	0	0

KM example 1

ab \ cd	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	1	1	0
10	0	0	0	0

KM example 1

$ab \backslash cd$	00	01	11	10
00	0	0	0	0
01	0	1	1	0
11	0	1	1	0
10	0	0	0	0

$$b \wedge d$$

KM example 2

cd

ab	00	01	11	10
00	0	0	0	0
01	1	1	0	0
11	1	1	0	0
10	0	0	0	0

KM example 2

			cd		
		00	01	11	10
	00	0	0	0	0
ab	01	1	1	0	0
	11	1	1	0	0
	10	0	0	0	0

KM example 2

cd

ab	00	01	11	10
00	0	0	0	0
01	1	1	0	0
11	1	1	0	0
10	0	0	0	0

$$b \wedge \neg c$$

KM example 3

$ab \backslash cd$	00	01	11	10
00	1	1	1	1
01	0	0	0	0
11	0	0	0	0
10	0	0	0	0

KM example 3

ab \ cd	00	01	11	10
00	1	1	1	1
01	0	0	0	0
11	0	0	0	0
10	0	0	0	0

KM example 3

$ab \backslash cd$	00	01	11	10
00	1	1	1	1
01	0	0	0	0
11	0	0	0	0
10	0	0	0	0

$$\neg a \wedge \neg b$$

KM example 4

ab \ cd	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	0	0	0	0
10	0	0	0	0

KM example 4

$ab \backslash cd$	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	0	0	0	0
10	0	0	0	0

KM example 4

$ab \backslash cd$	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	0	0	0	0
10	0	0	0	0

$$\neg a \wedge \neg d$$

KM example 5

		cd			
		00	01	11	10
ab	00	1	1	0	0
	01	1	1	0	0
	11	0	0	1	1
	10	0	0	1	1

KM example 5

ab \ cd	00	01	11	10
00	1	1	0	0
01	1	1	0	0
11	0	0	1	1
10	0	0	1	1

KM example 5

$ab \backslash cd$	00	01	11	10
00	1	1	0	0
01	1	1	0	0
11	0	0	1	1
10	0	0	1	1

$$(\neg a \wedge \neg c) \vee (a \wedge c)$$

KM example 6

$ab \backslash cd$	00	01	11	10
00	1	0	0	1
01	0	0	0	0
11	0	0	0	0
10	1	0	0	1

KM example 6

$ab \backslash cd$	00	01	11	10
00	1	0	0	1
01	0	0	0	0
11	0	0	0	0
10	1	0	0	1

KM example 6

ab \ cd	00	01	11	10
00	1	0	0	1
01	0	0	0	0
11	0	0	0	0
10	1	0	0	1

$$\neg b \wedge \neg d$$

KM example 7

$ab \backslash cd$	00	01	11	10
00	1	0	0	0
01	1	1	0	0
11	0	1	1	1
10	0	0	1	1

KM example 7

ab \ cd	00	01	11	10
00	1	0	0	0
01	1	1	0	0
11	0	1	1	1
10	0	0	1	1

KM example 7

ab \ cd	00	01	11	10
00	1	0	0	0
01	1	1	0	0
11	0	1	1	1
10	0	0	1	1

$$
\begin{aligned}
&(\neg a \land \neg c \land \neg d) \\
\lor\ &(b \land \neg c \land d) \\
\lor\ &(a \land c)
\end{aligned}
$$

KM example 8

$ab \backslash cd$	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	1	1	1
10	0	0	1	1

KM example 8

ab \ *cd*	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	1	1	1
10	0	0	1	1

KM example 8

cd

ab \ cd	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	1	1	1
10	0	0	1	1

$$(a \wedge b \wedge \neg c \wedge d) \vee (a \wedge c)$$

KM example 8

$ab \backslash cd$	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	0	1	1	1
10	0	0	1	1

$$(a \wedge b \wedge d) \vee (a \wedge c)$$

Disjunctive Normal Form

The descriptions we've built from KMs all have the form

$$(\cdots \wedge \cdots) \vee (\cdots \wedge \cdots) \vee \cdots$$

so we're describing unions of even rectangles.

Disjunctive Normal Form

The descriptions we've built from KMs all have the form

$$(\cdots \wedge \cdots) \vee (\cdots \wedge \cdots) \vee \cdots$$

so we're describing unions of even rectangles.

This is disjunctive normal form (DNF). Formally, we say a formula is in DNF iff has the form

$$\bigvee_i \Big(\bigwedge_j p_{ij} \Big)$$

where each p_{ij} is a literal (either a boolean variable/atomic proposition $a, b, \ldots$ or a negated atom $\neg a, \neg b, \ldots$).

Disjunctive Normal Form

The descriptions we've built from KMs all have the form

$$(\cdots \wedge \cdots) \vee (\cdots \wedge \cdots) \vee \cdots$$

so we're describing unions of even rectangles.

This is disjunctive normal form (DNF). Formally, we say a formula is in DNF iff has the form

$$\bigvee_i \Big(\bigwedge_j p_{ij} \Big)$$

where each p_{ij} is a literal (either a boolean variable/atomic proposition $a, b, \ldots$ or a negated atom $\neg a, \neg b, \ldots$).

Later we will see more mechanistic ways of converting to DNF.

KMs: looking at zeros

Sometimes it's a bit easier to look at the zeros.

Looking at the ones:

ab \ cd	00	01	11	10
00	1	1	1	1
01	1	0	0	1
11	1	0	0	1
10	1	1	1	1

KMs: looking at zeros

Sometimes it's a bit easier to look at the zeros.

Looking at the ones:

ab \ cd	00	01	11	10
00	1	1	1	1
01	1	0	0	1
11	1	0	0	1
10	1	1	1	1

$$\neg b \vee \neg d$$

KMs: looking at zeros

Sometimes it's a bit easier to look at the zeros.

Looking at the zeros:

$ab \backslash cd$	00	01	11	10
00	1	1	1	1
01	1	0	0	1
11	1	0	0	1
10	1	1	1	1

$$\neg(b \wedge d)$$

KMs: looking at zeros

Another example:

Looking at the ones:

$ab \backslash cd$	00	01	11	10
00	0	0	1	1
01	1	1	1	1
11	1	1	1	1
10	1	1	0	0

KMs: looking at zeros

Another example:

Looking at the ones:

ab \ cd	00	01	11	10
00	0	0	1	1
01	1	1	1	1
11	1	1	1	1
10	1	1	0	0

$$(\neg a \wedge c) \vee (a \wedge \neg c) \vee b$$

KMs: looking at zeros

Another example:

Looking at the zeros:

$ab \backslash cd$	00	01	11	10
00	0	0	1	1
01	1	1	1	1
11	1	1	1	1
10	1	1	0	0

$$\neg((\neg a \wedge \neg b \wedge \neg c) \vee (a \wedge \neg b \wedge c))$$

KMs: looking at zeros

Another example:

Looking at the zeros:

$ab \backslash cd$	00	01	11	10
00	0	0	1	1
01	1	1	1	1
11	1	1	1	1
10	1	1	0	0

$$(a \vee b \vee c) \wedge (\neg a \vee b \vee \neg c)$$

Conjunctive Normal Form

Take a formula $\bigvee_i (\bigwedge_j p_{ij})$ in DNF.

Conjunctive Normal Form

Take a formula $\bigvee_i (\bigwedge_j p_{ij})$ in DNF.

Its negation converts by De Morgan's laws to conjunctive normal form:

$$\neg \bigvee_i \Big(\bigwedge_j p_{ij} \Big) = \bigwedge_i \Big(\bigvee_j \neg p_{ij} \Big)$$

Conjunctive Normal Form

Take a formula $\bigvee_i (\bigwedge_j p_{ij})$ in DNF.

Its negation converts by De Morgan's laws to conjunctive normal form:

$$\neg \bigvee_i \Big(\bigwedge_j p_{ij} \Big) = \bigwedge_i \Big(\bigvee_j \neg p_{ij} \Big)$$

Any boolean expression can be put into either DNF or CNF – each is better for some applications.

Conjunctive Normal Form

Take a formula $\bigvee_i(\bigwedge_j p_{ij})$ in DNF.

Its negation converts by De Morgan's laws to conjunctive normal form:

$$\neg \bigvee_i \Big(\bigwedge_j p_{ij} \Big) = \bigwedge_i \Big(\bigvee_j \neg p_{ij} \Big)$$

Any boolean expression can be put into either DNF or CNF – each is better for some applications.

If I give you a formula in DNF, can you convert it (*not* its negation) to CNF? How big might the result be?