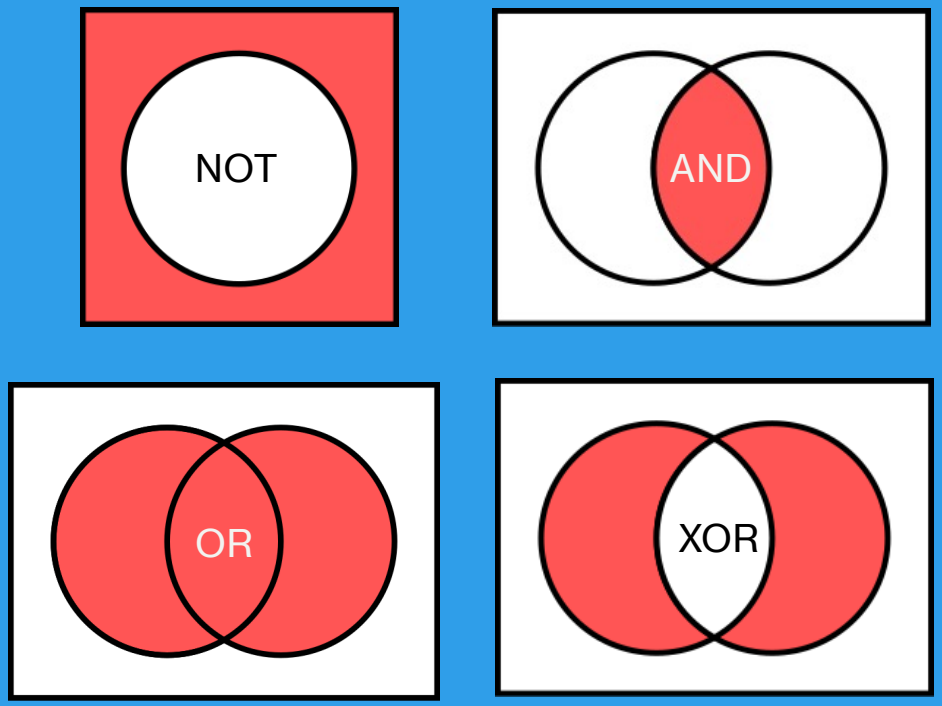
Computer Systems



Boolean Algebra

CS51 – Spring 2026

Admin

How is Assignment 1 going?

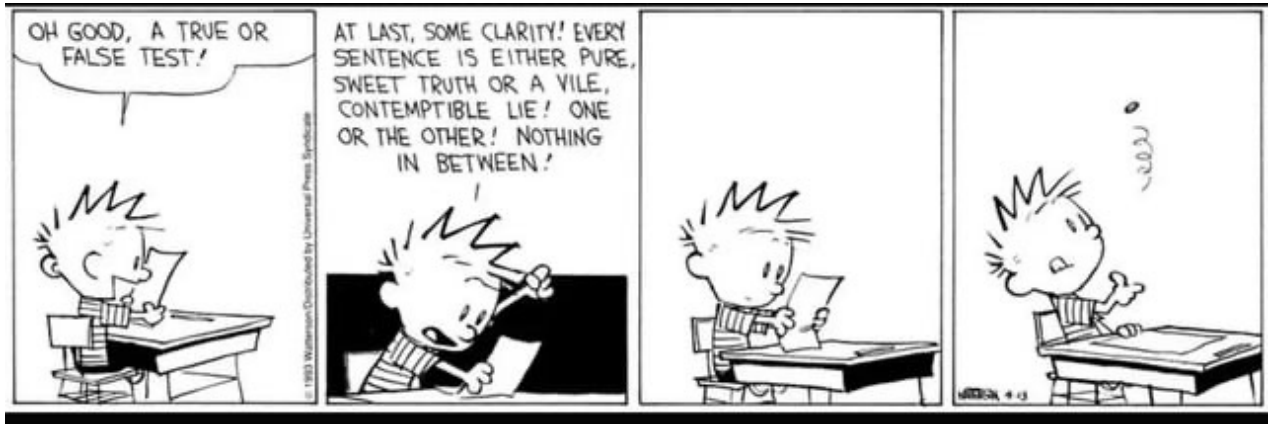
Booleans

What is a Boolean (e.g., a Boolean variable)?

Booleans

What is a Boolean (e.g., Boolean variable)?

A variable/value that only has two possible values True/1 or False/0



Why are they called Booleans?

Why are they called Booleans?

George Boole was a self-taught English mathematician (1815-1864)

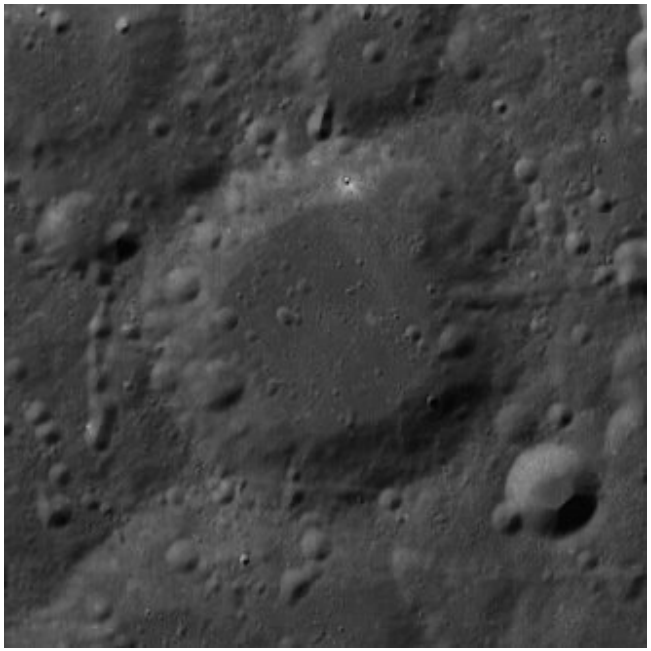
He introduced **binary variables** and **binary logic**

which laid the foundation for **Boolean algebra** (which looks at how you can combine Boolean variables)

George Boole

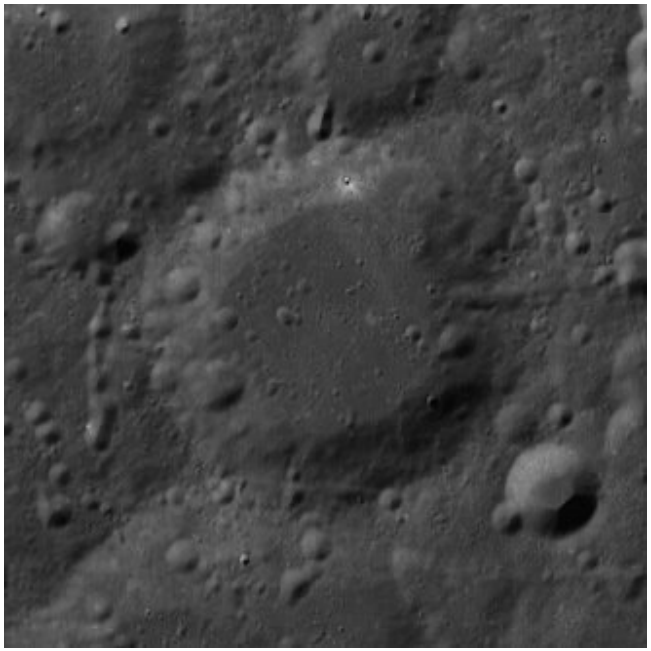
An aside

Anyone know what this is?



An aside

Anyone know what this is?



Boole crater

https://en.wikipedia.org/wiki/Boole_(crater)

Boolean algebra

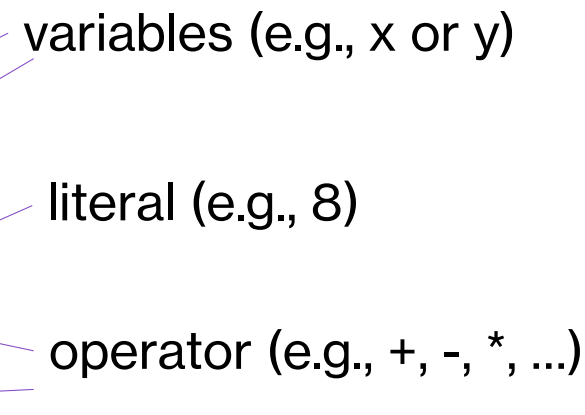
Boolean algebra is a branch of algebra where variables can only have two values

Algebra:

$x + y$

$(3x - 5)/8$

$x^2 - y^2$



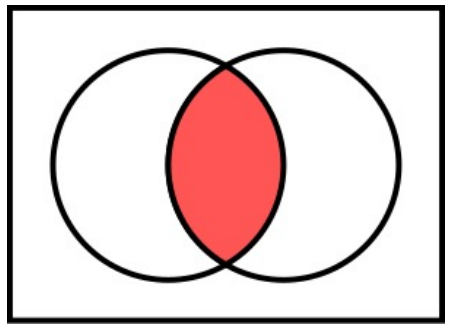
Boolean algebra

Boolean algebra is a branch of algebra where variables can only have two values

Literals: True, False (alternatively: 1, 0)

Operators?

AND operation – Conjunction



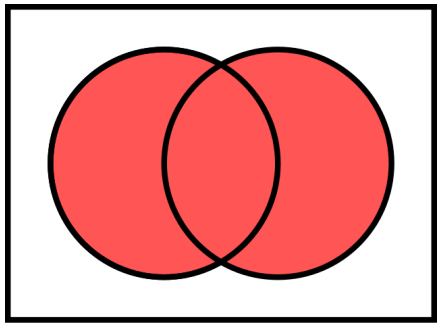
X	Y	X AND Y
FALSE	FALSE	FALSE
FALSE	TRUE	FALSE
TRUE	FALSE	FALSE
TRUE	TRUE	TRUE

X	Y	X ∧ Y
0	0	0
0	1	0
1	0	0
1	1	1

Only TRUE if both (or all) values are TRUE

Example: I'm going to hang out if I finish my homework **AND** it doesn't get too late.

OR operation – Disjunction

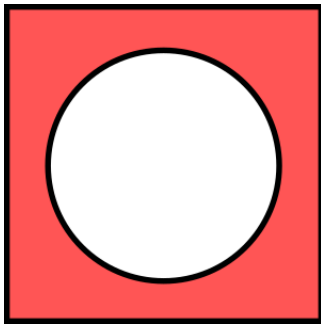


X	Y	X OR Y
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	TRUE

X	Y	X v Y
0	0	0
0	1	1
1	0	1
1	1	1

TRUE at least one value is TRUE

NOT operation – Negation



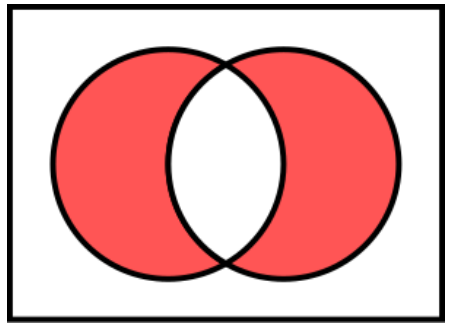
X	NOT X
FALSE	TRUE
TRUE	FALSE

X	¬ X
0	1
1	0

Negates or flips the value

Example: I'm going to hate CS51. NOT.

XOR operation – Exclusive OR



X	Y	X XOR Y
FALSE	FALSE	FALSE
FALSE	TRUE	TRUE
TRUE	FALSE	TRUE
TRUE	TRUE	FALSE

X	Y	X ⊕ Y
0	0	0
0	1	1
1	0	1
1	1	0

TRUE if *exactly one* value is TRUE

PRACTICE TIME – Evaluate these expressions

- $0 \vee \neg 0$
- $1 \wedge \neg 1$
- $\neg (1 \wedge 0)$
- $0 \wedge \neg 1$
- $\neg (1 \vee \neg 1)$
- $1 \oplus \neg 1$
- $(1 \wedge (1 \oplus \neg(0 \vee \neg 1)))$

AND

X	Y	X ∧ Y
0	0	0
0	1	0
1	0	0
1	1	1

NOT

X	¬ X
0	1
1	0

OR

X	Y	X ∨ Y
0	0	0
0	1	1
1	0	1
1	1	1

XOR

X	Y	X ⊕ Y
0	0	0
0	1	1
1	0	1
1	1	0

ANSWER– Evaluate these expressions

- $0 \vee \neg 0 = 0 \vee 1 = 1$
- $1 \wedge \neg 1 = 1 \wedge 0 = 0$
- $\neg (1 \wedge 0) = \neg 0 = 1$
- $0 \wedge \neg 1 = 0 \wedge 0 = 0$
- $\neg (1 \vee \neg 1) = \neg (1 \vee 0) = \neg 1 = 0$
- $1 \oplus \neg 1 = 1 \oplus 0 = 1$
- $(1 \wedge (1 \oplus \neg(0 \vee \neg 1))) =$
  $(1 \wedge (1 \oplus \neg(0 \vee 0))) =$
  $(1 \wedge (1 \oplus \neg 0)) =$
  $(1 \wedge (1 \oplus 1)) = 1 \wedge 0 = 0$

Boolean functions of three+ variables

We can have Boolean functions with any number of variables:

For example, AND and OR generalize as expected:

X	Y	Z	AND(X,Y,Z)	OR(X,Y,Z)
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Operator precedence

2 + 3 * 5 =

(2 + 3) * 5 =

Operator precedence

2 + 3 * 5 = 17

(2 + 3) * 5 = 25

Boolean operator precedence

$1 \vee 1 \wedge 0 =$

$\neg 0 \wedge 0 =$

Boolean operator precedence

$1 \vee 1 \wedge 0 = 1$

$\neg 0 \wedge 0 = 0$

Most common convention:

highest precedence (happens first)

$\neg$

$\wedge$

$\vee$

lowest precedence (happens later)

Boolean operator precedence: use parens

$(1 \vee 1) \wedge 0 = 0$

$\neg(0 \wedge 0) = 1$

Most common convention:

highest precedence (happens first)

()

¬

∧

∨

lowest precedence (happens later)

Programming lang

Distributive law

$x\ (y + z) =$

Distributive law

$$x\,(y+z) = xy + zy$$

Distributive law in Boolean algebra

$x \wedge (y \vee z) =$

Distributive law in Boolean algebra

$$x \wedge (y \vee z) = (x \wedge y) \vee (\mathrm{x} \wedge z)$$

How would you check this?

Distributive law in Boolean algebra

$$x \wedge (y \vee z) = (x \wedge y) \vee (\mathrm{x} \wedge z)$$

X	Y	Z	Y ∨ Z	X ∧ (Y∨ Z)	(X ∧ Y)	(X ∧ Z)	(X ∧ Y) ∨ (X ∧ Z)
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

PRACTICE TIME – De Morgan's laws

- ¬ (X ∧ Y) =
- ¬ (X ∨ Y) =

X	Y	X ∧ Y	¬ (X ∧ Y)	X ∨ Y	¬ (X ∨ Y)	¬ X	¬ Y	¬ X ∧ ¬ Y	¬ X ∨ ¬ Y
0	0								
0	1								
1	0								
1	1								

ANSWER – De Morgan's laws

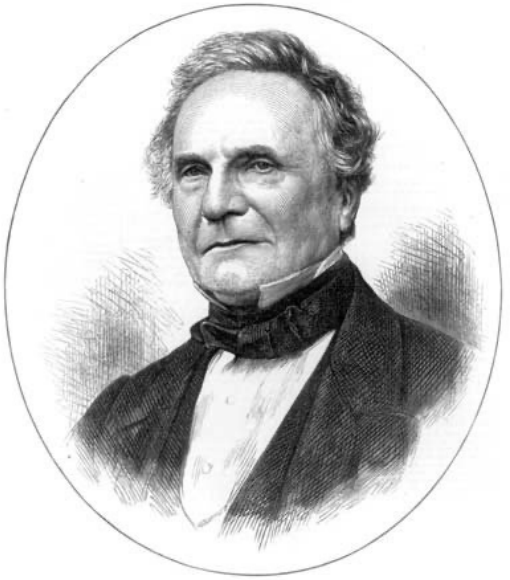
- ¬ (X ∧ Y) = ¬ X ∨ ¬ Y
- ¬ (X ∨ Y) = ¬ X ∧ ¬ Y

X	Y	X ∧ Y	¬ (X ∧ Y)	X ∨ Y	¬ (X ∨ Y)	¬ X	¬ Y	¬ X ∧ ¬ Y	¬ X ∨ ¬ Y
0	0	0	1	0	1	1	1	1	1
0	1	0	1	1	0	1	0	0	1
1	0	0	1	1	0	0	1	0	1
1	1	1	0	1	0	0	0	0	0

A mathematical lineage to the first computer

Augustus De Morgan

Ada Lovelace

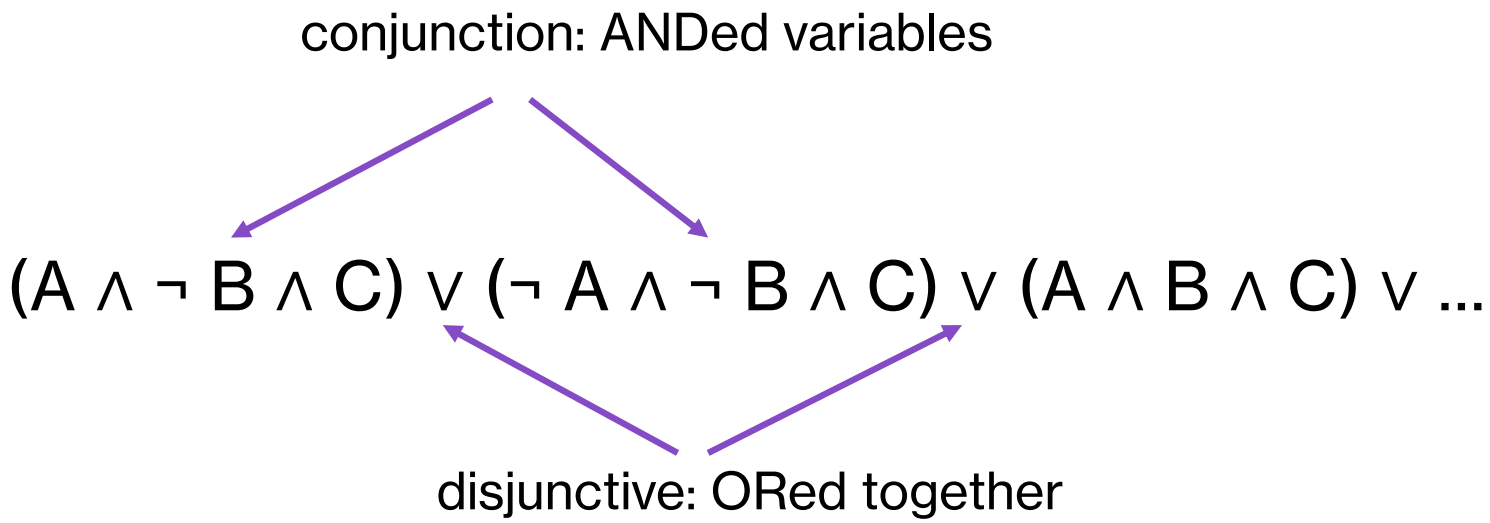


Charles Babbage

Disjunctive normal form (DNF)

Theorem (Boole, 1847):

Any Boolean function can be represented as a disjunction (i.e., OR) of conjunctions (i.e., ANDs) of its arguments and their negation (NOT). This is also known as **disjunctive normal form** (DNF) or a sum of products/minterms.



Disjunctive normal form (DNF)

One way this can happen is by using “laws”, specifically:

$$
\begin{array}{rcl}
(\neg\neg x) & \rightsquigarrow & x \\
(\neg(x \vee y)) & \rightsquigarrow & ((\neg x) \wedge (\neg y)) \\
(\neg(x \wedge y)) & \rightsquigarrow & ((\neg x) \vee (\neg y)) \\
(x \wedge (y \vee z)) & \rightsquigarrow & ((x \wedge y) \vee (x \wedge z)) \\
((x \vee y) \wedge z) & \rightsquigarrow & ((x \wedge z) \vee (y \wedge z))
\end{array}
$$

double negation

De Morgan’s laws

distributive law

Minterms

Another way this can happen is directly from the truth table

Row number	X	Y	Z	Minterm
0	0	0	0	$m_0 = \neg X \wedge \neg Y \wedge \neg Z$
1	0	0	1	$m_1 = \neg X \wedge \neg Y \wedge Z$
2	0	1	0	$m_2 = \neg X \wedge Y \wedge \neg Z$
3	0	1	1	$m_3 = \neg X \wedge Y \wedge Z$
4	1	0	0	$m_4 = X \wedge \neg Y \wedge \neg Z$
5	1	0	1	$m_5 = X \wedge \neg Y \wedge Z$
6	1	1	0	$m_6 = X \wedge Y \wedge \neg Z$
7	1	1	1	$m_7 = X \wedge Y \wedge Z$

Minterm: a Boolean expression consisting of the conjunction (AND) of all variables of a Boolean function

Minterms

Note that a minterm will be TRUE/1 for **exactly one entry** in the truth table

Row number	X	Y	Z	$m_3 = \neg X \wedge Y \wedge Z$
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	0
6	1	1	0	0
7	1	1	1	0

Minterm: a Boolean expression consisting of the conjunction (AND) of all variables of a Boolean function

An example: majority function

MAJ: A majority of the variables are TRUE (1)

X	Y	Z	MAJ(X,Y,Z)
0	0	0	0
0	0	1	0
0	1	0	0
0	**1**	**1**	**1**
1	0	0	0
1	**0**	**1**	**1**
1	**1**	**0**	**1**
1	**1**	**1**	**1**

Minterms -> DNF

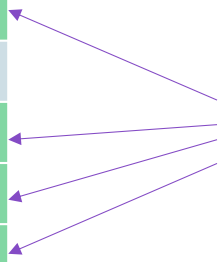
Key idea: identify and combine entries that are 1

X	Y	Z	MAJ(X,Y,Z)
0	0	0	0
0	0	1	0
0	1	0	0
0	**1**	**1**	**1**
1	0	0	0
1	**0**	**1**	**1**
1	**1**	**0**	**1**
1	**1**	**1**	**1**

Minterms -> DNF

Key idea: identify and combine entries that are 1

X	Y	Z	MAJ(X,Y,Z)
0	0	0	0
0	0	1	0
0	1	0	0
0	**1**	**1**	**1**
1	0	0	0
1	**0**	**1**	**1**
1	**1**	**0**	**1**
1	**1**	**1**	**1**



OR these together

If any one of these is the entry, then we are 1

DNF of MAJ function

X	Y	Z	MAJ(X,Y,Z)	Minterm
0	0	0	0	$m_0 = \neg X \wedge \neg Y \wedge \neg Z$
0	0	1	0	$m_1 = \neg X \wedge \neg Y \wedge Z$
0	1	0	0	$m_2 = \neg X \wedge Y \wedge \neg Z$
0	**1**	**1**	**1**	$\mathbf{m_3 = \neg X \wedge Y \wedge Z}$
1	0	0	0	$m_4 = X \wedge \neg Y \wedge \neg Z$
1	**0**	**1**	**1**	$\mathbf{m_5 = X \wedge \neg Y \wedge Z}$
1	**1**	**0**	**1**	$\mathbf{m_6 = X \wedge Y \wedge \neg Z}$
1	**1**	**1**	**1**	$\mathbf{m_7 = X \wedge Y \wedge Z}$

MAJ(X, Y, Z) = (¬X ∧ Y ∧ Z) ∨ (X ∧ ¬Y ∧ Z) ∨ (X ∧ Y ∧ ¬Z) ∨ (X ∧ Y ∧ Z)

PRACTICE TIME - ODD function using DNF

ODD($X_1, X_2, \ldots, X_n$) = 1 if an odd number of arguments is 1, 0 otherwise

X	Y	Z	ODD(X,Y,Z)	Minterm
0	0	0	0	$m_0 =$
0	0	1	1	$m_1 =$
0	1	0	1	$m_2 =$
0	1	1	0	$m_3 =$
1	0	0	1	$m_4 =$
1	0	1	0	$m_5 =$
1	1	0	0	$m_6 =$
1	1	1	1	$m_7 =$

ODD(X, Y, Z) =

PRACTICE TIME - ODD function using DNF

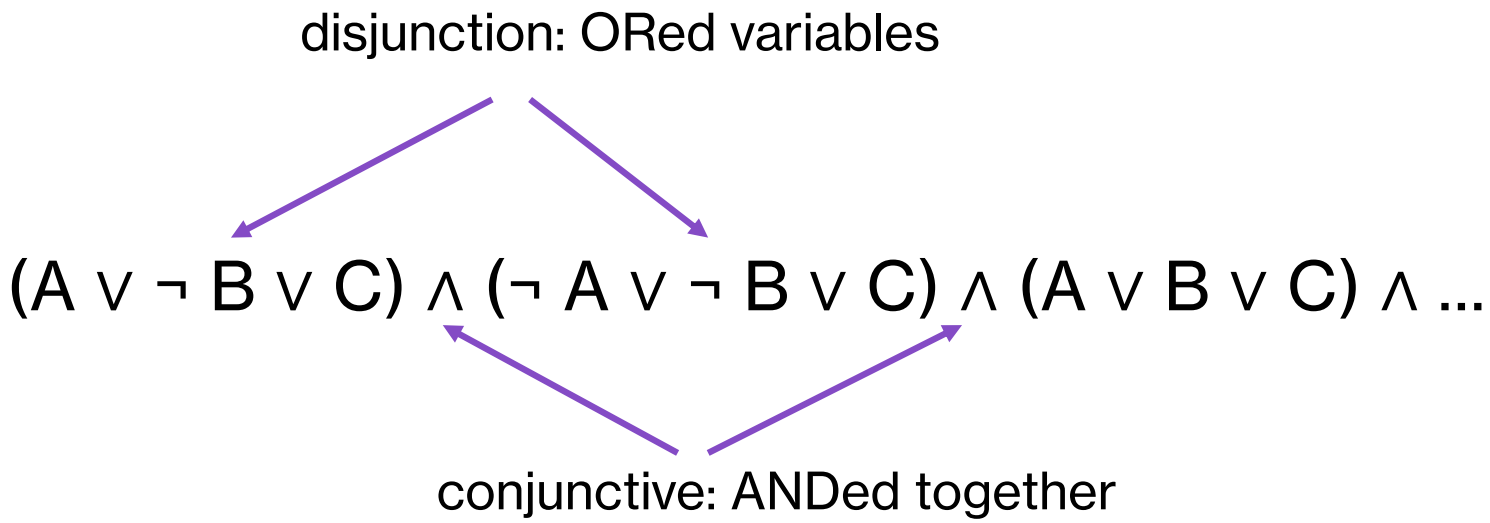
ODD($X_1, X_2, \ldots, X_n$) = 1 if an odd number of arguments is 1, 0 otherwise

X	Y	Z	ODD(X,Y,Z)	Minterm
0	0	0	0	$m_0 =$
0	0	1	1	$m_1 = \neg X \wedge \neg Y \wedge Z$
0	1	0	1	$m_2 = \neg X \wedge Y \wedge \neg Z$
0	1	1	0	$m_3 =$
1	0	0	1	$m_4 = X \wedge \neg Y \wedge \neg Z$
1	0	1	0	$m_5 =$
1	1	0	0	$m_6 =$
1	1	1	1	$m_7 = \mathrm{X} \wedge \mathrm{Y} \wedge \mathrm{Z}$

ODD(X, Y, Z) = (¬X ∧ ¬Y ∧ Z) ∨ (¬X ∧ Y ∧ ¬Z) ∨ (X ∧ ¬Y ∧ ¬Z) ∨ (X ∧ Y ∧ Z)

Conjunctive normal form (CNF)

Any Boolean function can be represented as a conjunction (i.e., AND) of disjunctions (i.e., ORs) of its arguments and their negation (NOT). This is also known as **conjunctive normal form** (CNF) or a product of sums/maxterms.



Maxterms

Maxterm: a disjunction (OR) of all variables of a Boolean function

Row number	X	Y	Z	Maxterm
0	0	0	0	$M_0 = X \vee Y \vee Z$
1	0	0	1	$M_1 = X \vee Y \vee \neg Z$
2	0	1	0	$M_2 = X \vee \neg Y \vee Z$
3	0	1	1	$M_3 = X \vee \neg Y \vee \neg Z$
4	1	0	0	$M_4 = \neg X \vee Y \vee Z$
5	1	0	1	$M_5 = \neg X \vee Y \vee \neg Z$
6	1	1	0	$M_6 = \neg X \vee \neg Y \vee Z$
7	1	1	1	$M_7 = \neg X \vee \neg Y \vee \neg Z$

The maxterm is formed by ORing together the "opposite" of the value in the truth table

Maxterms

Note that a maxterm will be TRUE/1 for **all entries except one** in the truth table

Row number	X	Y	Z	$M_5 = \neg X \vee Y \vee \neg Z$
0	0	0	0	1
1	0	0	1	1
2	0	1	0	1
3	0	1	1	1
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

Maxterms -> CNF

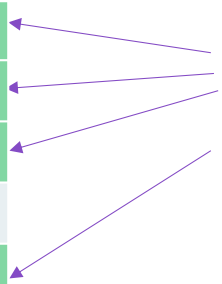
Key idea: identify and combine entries that are 0, i.e., as long as it's not one of these entries, it's 1

X	Y	Z	MAJ(X,Y,Z)
0	**0**	**0**	**0**
0	**0**	**1**	**0**
0	**1**	**0**	**0**
0	1	1	1
1	**0**	**0**	**0**
1	0	1	1
1	1	0	1
1	1	1	1

Maxterms -> CNF

Key idea: identify and combine entries that are 0, i.e., **as long as it's not one of these entries, it's 1**

X	Y	Z	MAJ(X,Y,Z)
0	**0**	**0**	**0**
0	**0**	**1**	**0**
0	**1**	**0**	**0**
0	1	1	1
1	**0**	**0**	**0**
1	0	1	1
1	1	0	1
1	1	1	1



AND these together

CNF of MAJ function

X	Y	Z	MAJ(X,Y,Z)	Maxterm
0	**0**	**0**	**0**	$M_0 = X \vee Y \vee Z$
0	**0**	**1**	**0**	$M_1 = X \vee Y \vee \neg Z$
0	**1**	**0**	**0**	$M_2 = X \vee \neg Y \vee Z$
0	1	1	1	$M_3 = X \vee \neg Y \vee \neg Z$
1	**0**	**0**	**0**	$M_4 = \neg X \vee Y \vee Z$
1	0	1	1	$M_5 = \neg X \vee Y \vee \neg Z$
1	1	0	1	$M_6 = \neg X \vee \neg Y \vee Z$
1	1	1	1	$M_7 = \neg X \vee \neg Y \vee \neg Z$

$$MAJ(X, Y, Z) = (X \vee Y \vee Z) \wedge (X \vee Y \vee \neg Z) \wedge (X \vee \neg Y \vee Z) \wedge (\neg X \vee Y \vee Z)$$

MAJ(0, 1, 1) = ?

X	Y	Z	MAJ(X,Y,Z)	Maxterm
0	0	0	0	$M_0 = X \vee Y \vee Z$
0	0	1	0	$M_1 = X \vee Y \vee \neg Z$
0	1	0	0	$M_2 = X \vee \neg Y \vee Z$
0	1	1	1	$M_3 = X \vee \neg Y \vee \neg Z$
1	0	0	0	$M_4 = \neg X \vee Y \vee Z$
1	0	1	1	$M_5 = \neg X \vee Y \vee \neg Z$
1	1	0	1	$M_6 = \neg X \vee \neg Y \vee Z$
1	1	1	1	$M_7 = \neg X \vee \neg Y \vee \neg Z$

MAJ(X, Y, Z) = (X ∨ Y ∨ Z) ∧ (X ∨ Y ∨ ¬Z) ∧ (X ∨ ¬Y ∨ Z) ∧ (¬X ∨ Y ∨ Z)

MAJ(0, 1, 1) = ?

X	Y	Z	MAJ(X,Y,Z)	Maxterm
0	0	0	0	$M_0 = X \vee Y \vee Z$
0	0	1	0	$M_1 = X \vee Y \vee \neg Z$
0	1	0	0	$M_2 = X \vee \neg Y \vee Z$
0	1	1	1	$M_3 = X \vee \neg Y \vee \neg Z$
1	0	0	0	$M_4 = \neg X \vee Y \vee Z$
1	0	1	1	$M_5 = \neg X \vee Y \vee \neg Z$
1	1	0	1	$M_6 = \neg X \vee \neg Y \vee Z$
1	1	1	1	$M_7 = \neg X \vee \neg Y \vee \neg Z$

1 1 1 1

MAJ(X, Y, Z) = (X ∨ **Y** ∨ **Z**) ∧ (X ∨ **Y** ∨ ¬Z) ∧ (X ∨ ¬Y ∨ **Z**) ∧ (**¬X** ∨ **Y** ∨ **Z**)

MAJ(1, 0, 0) = ?

X	Y	Z	MAJ(X,Y,Z)	Maxterms
0	0	0	0	$M_0 = X \vee Y \vee Z$
0	0	1	0	$M_1 = X \vee Y \vee \neg Z$
0	1	0	0	$M_2 = X \vee \neg Y \vee Z$
0	1	1	1	$M_3 = X \vee \neg Y \vee \neg Z$
1	0	0	0	$M_4 = \neg X \vee Y \vee Z$
1	0	1	1	$M_5 = \neg X \vee Y \vee \neg Z$
1	1	0	1	$M_6 = \neg X \vee \neg Y \vee Z$
1	1	1	1	$M_7 = \neg X \vee \neg Y \vee \neg Z$

MAJ(X, Y, Z) = (X ∨ Y ∨ Z) ∧ (X ∨ Y ∨ ¬Z) ∧ (X ∨ ¬Y ∨ Z) ∧ (¬X ∨ Y ∨ Z)

MAJ(1, 0, 0) = ?

X	Y	Z	MAJ(X,Y,Z)	Maxterms
0	0	0	0	$M_0 = X \vee Y \vee Z$
0	0	1	0	$M_1 = X \vee Y \vee \neg Z$
0	1	0	0	$M_2 = X \vee \neg Y \vee Z$
0	1	1	1	$M_3 = X \vee \neg Y \vee \neg Z$
1	0	0	0	$M_4 = \neg X \vee Y \vee Z$
1	0	1	1	$M_5 = \neg X \vee Y \vee \neg Z$
1	1	0	1	$M_6 = \neg X \vee \neg Y \vee Z$
1	1	1	1	$M_7 = \neg X \vee \neg Y \vee \neg Z$

1 **1** **1** **0**

MAJ(X, Y, Z) = (**X** ∨ Y ∨ Z) ∧ (**X** ∨ Y ∨ **¬Z**) ∧ (**X** ∨ **¬Y** ∨ Z) ∧ (¬X ∨ Y ∨ Z)

PRACTICE TIME - ODD function using CNF

ODD($X_1, X_2, \ldots, X_n$) = 1 if an odd number of arguments is 1, 0 otherwise

X	Y	Z	ODD(X,Y,Z)	Maxterms
0	0	0	0	$M_0 =$
0	0	1	1	$M_1 =$
0	1	0	1	$M_2 =$
0	1	1	0	$M_3 =$
1	0	0	1	$M_4 =$
1	0	1	0	$M_5 =$
1	1	0	0	$M_6 =$
1	1	1	1	$M_7 =$

ODD(X, Y, Z) =

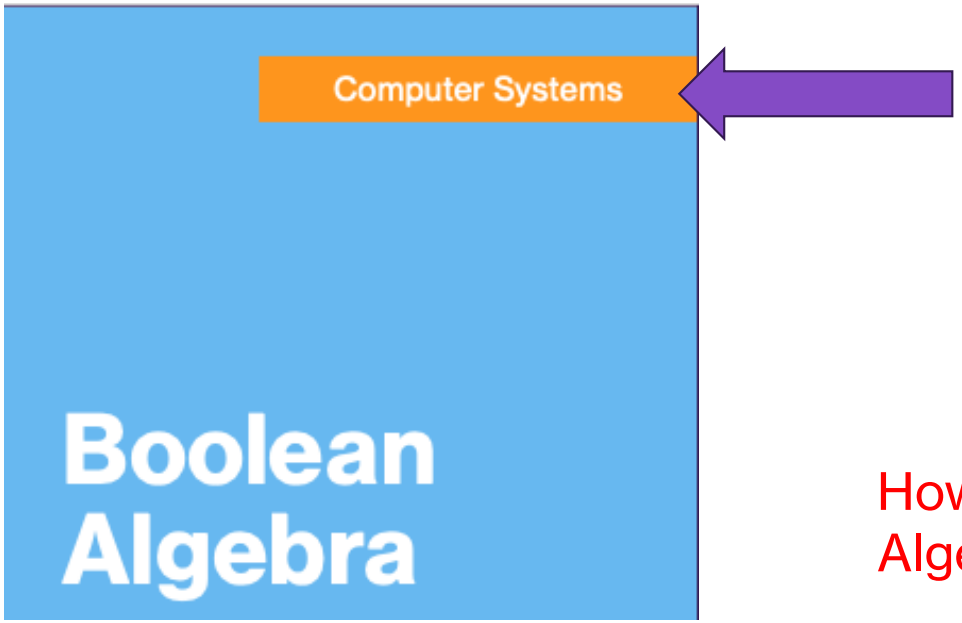
PRACTICE TIME - ODD function using CNF

ODD($X_1, X_2, \ldots, X_n$) = 1 if an odd number of arguments is 1, 0 otherwise

X	Y	Z	ODD(X,Y,Z)	Maxterms
0	0	0	0	$M_0 = X \vee Y \vee Z$
0	0	1	1	$M_1 =$
0	1	0	1	$M_2 =$
0	1	1	0	$M_3 = X \vee \neg Y \vee \neg Z$
1	0	0	1	$M_4 =$
1	0	1	0	$M_5 = \neg X \vee Y \vee \neg Z$
1	1	0	0	$M_6 = \neg X \vee \neg Y \vee Z$
1	1	1	1	$M_7 =$

ODD(X, Y, Z) = (X ∨ Y ∨ Z) ∧ (X ∨ Y ∨ ¬Z) ∧ (X ∨ ¬Y ∨ Z) ∧ (¬X ∨ Y ∨ Z)

Circling back...



This was the first lecture for our "Computer Systems" section

How does Boolean Algebra fit in?

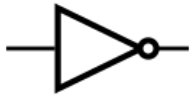
CPU

CPU: central processing unit

CPU

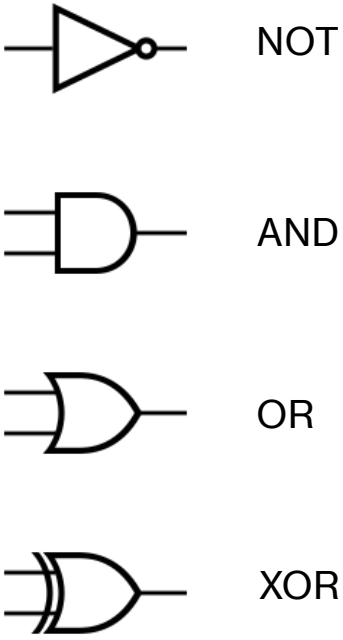
Inside a CPU

Gates



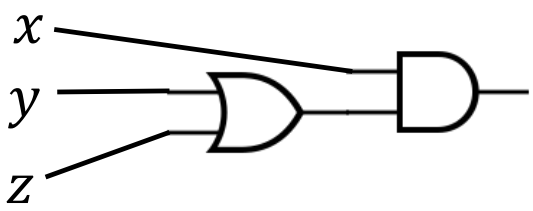
Inside a CPU

Gates

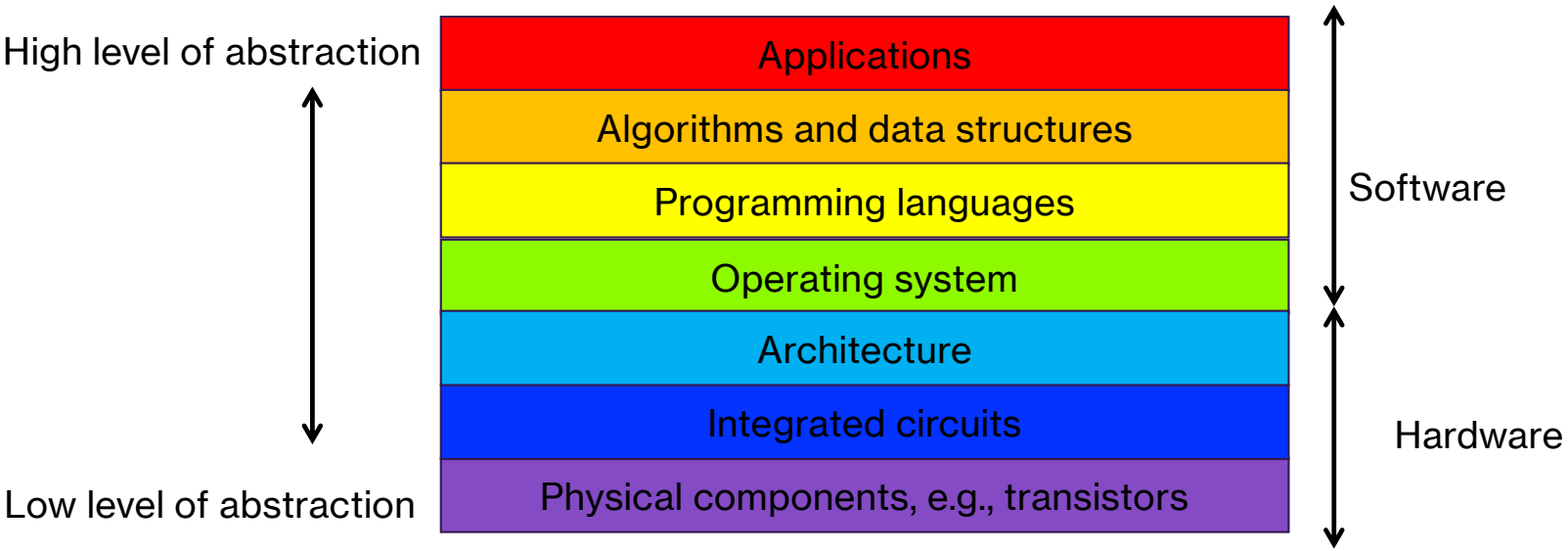


Gates

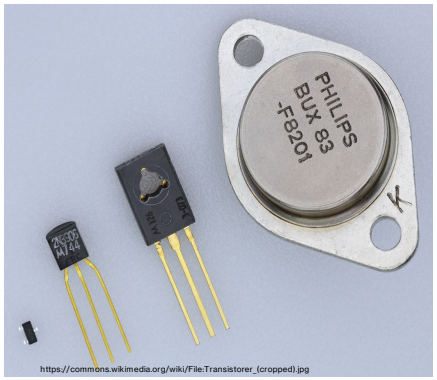
$$x \wedge (y \vee z) =$$



Abstraction

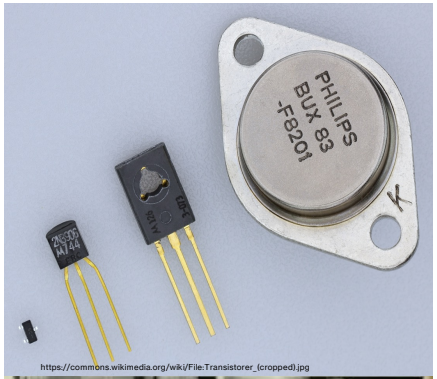


Transistors



Transistors

- Transistors are semiconductor devices made out of silicon.
- They act as switches that can be open or closed by applying electricity.
- Integrated circuits in modern computers are built off billions of transistors.
  - They are **small** – you can fit a few thousand of them across the width of a human hair!
  - They are **fast** – they can switch states millions of times per second.
  - They are **reliable** – they can run without any issue for decades.
- Invented in 1947 by Bardeen, Brattain, and Shockley at Bell Lab.
  - They shared the 1956 Nobel Prize in Physics for their achievement.
- A lot of development of transistors and circuits took place in Santa Clara Valley, which by the 1980s started being referred to as the **Silicon Valley**.



https://commons.wikimedia.org/wiki/File:Transistorer_(cropped).jpg

Simplifying Boolean formulas

DNF

$$MAJ(X, Y, Z) = (\neg X \wedge Y \wedge Z) \vee (X \wedge \neg Y \wedge Z) \vee (X \wedge Y \wedge \neg Z) \vee (X \wedge Y \wedge Z)$$

CNF

$$MAJ(X, Y, Z) = (X \vee Y \vee Z) \wedge (X \vee Y \vee \neg Z) \wedge (X \vee \neg Y \vee Z) \wedge (\neg X \vee Y \vee Z)$$

Both of these are pretty long. Can we come up with a simpler (shorter) representation?

Simplifying Boolean formulas

DNF

$$MAJ(X, Y, Z) = (\neg X \wedge Y \wedge Z) \vee (X \wedge \neg Y \wedge Z) \vee (X \wedge Y \wedge \neg Z) \vee (X \wedge Y \wedge Z)$$

CNF

$$MAJ(X, Y, Z) = (X \vee Y \vee Z) \wedge (X \vee Y \vee \neg Z) \wedge (X \vee \neg Y \vee Z) \wedge (\neg X \vee Y \vee Z)$$

For example:

$$MAJ(X, Y, Z) = (X \wedge Y) \vee (Y \wedge Z) \vee (Z \wedge X)$$

Karnaugh Maps (aka, K-maps)

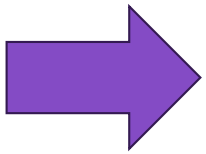
Another approach for creating a Boolean function from a truth table

X	Y	F
0	0	1
0	1	0
1	0	1
1	1	0

Karnaugh Maps (aka, K-maps)

Another approach for creating a Boolean function from a truth table

X	Y	F
0	0	1
0	1	0
1	0	1
1	1	0



Y \ X	0	1
0	1	1
1	0	0

Gray Code

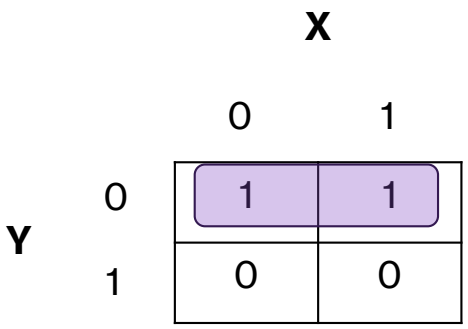
Karnaugh Maps (aka, K-maps)

Y \ X	0	1
0	1	1
1	0	0

Create groups of 1s:

- Only contain 1s
- Must be a square or rectangle
- Area must be a power of 2
- Groups should be as large as possible
- Groups can overlap
- Groups can wrap around
- All 1s must be covered

Karnaugh Maps (aka, K-maps)



Create groups of 1s:

- Only contain 1s
- Must be a square or rectangle
- Area must be a power of 2
- Groups should be as large as possible
- Groups can overlap
- Groups can wrap around
- All 1s must be covered

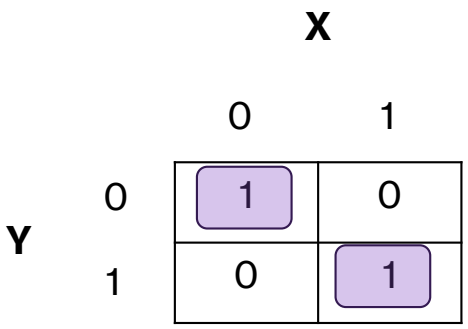
Karnaugh Maps (aka, K-maps)

Y \ X	0	1
0	1	0
1	0	1

Create groups of 1s:

- Only contain 1s
- Must be a square or rectangle
- Area must be a power of 2
- Groups should be as large as possible
- Groups can overlap
- Groups can wrap around
- All 1s must be covered

Karnaugh Maps (aka, K-maps)



Create groups of 1s:

- Only contain 1s
- Must be a square or rectangle
- Area must be a power of 2
- Groups should be as large as possible
- Groups can overlap
- Groups can wrap around
- All 1s must be covered

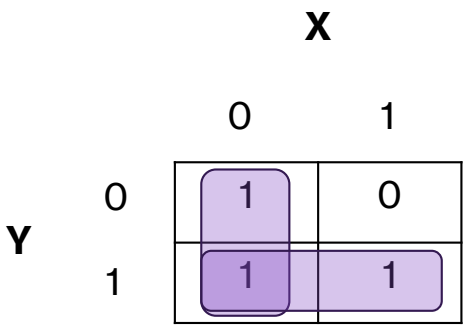
Karnaugh Maps (aka, K-maps)

Y \ X	0	1
0	1	0
1	1	1

Create groups of 1s:

- Only contain 1s
- Must be a square or rectangle
- Area must be a power of 2
- Groups should be as large as possible
- Groups can overlap
- Groups can wrap around
- All 1s must be covered

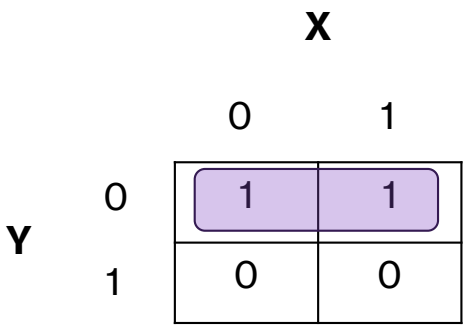
Karnaugh Maps (aka, K-maps)



Create groups of 1s:

- Only contain 1s
- Must be a square or rectangle
- Area must be a power of 2
- Groups should be as large as possible
- Groups can overlap
- Groups can wrap around
- All 1s must be covered

Karnaugh Maps (aka, K-maps)

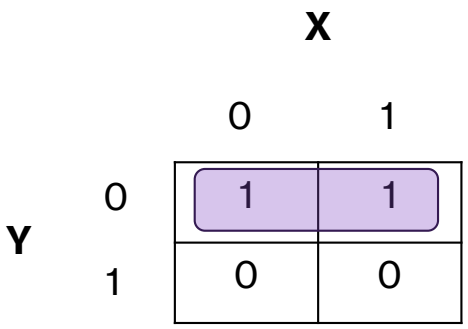


What does this become?

Each group becomes a conjunction:

- Look for the variable(s) that don't change:
  - If it stays a 1, you add the variable as is
  - If it stays a 0, you add the negation of the variable
- OR the conjunctions together

Karnaugh Maps (aka, K-maps)

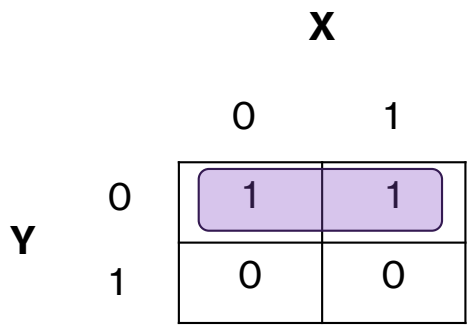


¬Y

Each group becomes a conjunction:

- Look for the variable(s) that don't change:
  - If it stays a 1, you add the variable as is
  - If it stays a 0, you add the negation of the variable
- OR the conjunctions together

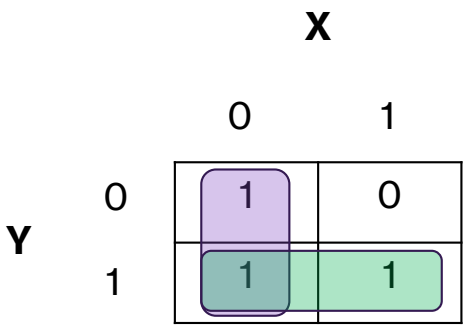
Karnaugh Maps (aka, K-maps)



¬Y

X	Y	F
0	0	1
0	1	0
1	0	1
1	1	0

Karnaugh Maps (aka, K-maps)

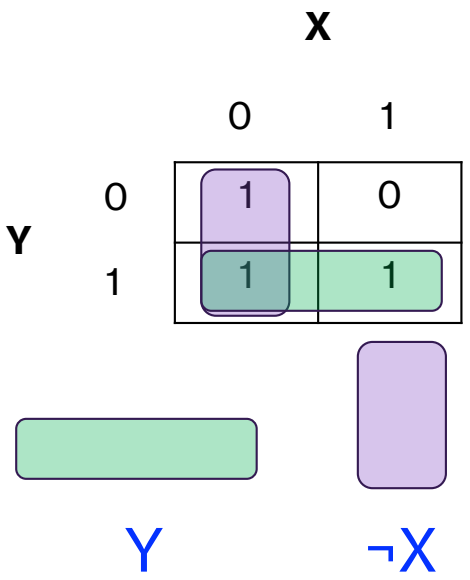


What does this become?

Each group becomes a conjunction:

- Look for the variable(s) that don't change:
  - If it stays a 1, you add the variable as is
  - If it stays a 0, you add the negation of the variable
- OR the conjunctions together

Karnaugh Maps (aka, K-maps)

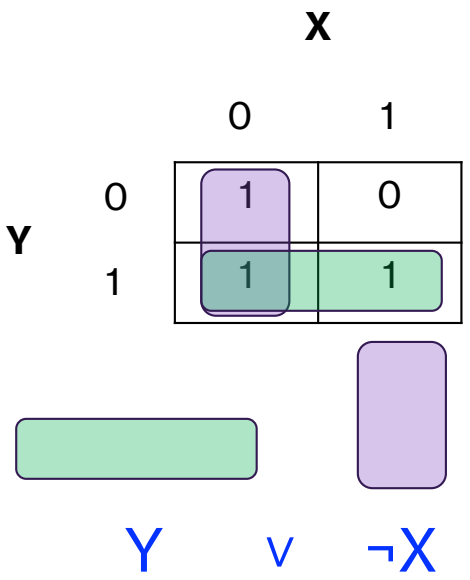


Each group becomes a conjunction:

- Look for the variable(s) that don't change:
  - If it stays a 1, you add the variable as is
  - If it stays a 0, you add the negation of the variable
- OR the conjunctions together

∨

Karnaugh Maps (aka, K-maps)



Each group becomes a conjunction:

- Look for the variable(s) that don't change:
  - If it stays a 1, you add the variable as is
  - If it stays a 0, you add the negation of the variable
- **OR the conjunctions together**

Practice Problems – Problem 1

- You are given the following truth table for a Boolean function with three variables, A, B, and C, and an output column F.
- What are the disjunctive and conjunctive normal forms based on this truth table?

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	1
1	1	0	1
1	1	1	0

Practice Problems – Problem 2

Assuming x = 6, y = 2, z = 12. What would the following Python expressions evaluate to?

- x == 10 or x < 5
- x != 7 and x > 3
- not (x >= 5 and y <= 2)
- (x < 0) or (x >= 0)
- not (x != x)
- x > 1 and (z / x) != 2
- x <= 0 or (z / x) == 1
- x == 4 and y == 4

Practice Problems – Answer 1

- Disjunctive normal form:

  $F = (\neg A \wedge \neg B \wedge \neg C) \vee (\neg A \wedge B \wedge \neg C) \vee (A \wedge \neg B \wedge C) \vee (A \wedge B \wedge \neg C)$

- Conjunctive normal form:

  $F = (A \vee B \vee \neg C) \wedge (A \vee \neg B \vee \neg C) \wedge (\neg A \vee B \vee C) \wedge (\neg A \vee \neg B \vee \neg C)$

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	1
1	1	0	1
1	1	1	0

Practice Problems – Answer 2

Assuming x = 6, y = 2, z = 12. What would the following Python expressions evaluate to?

- x == 10 or x < 5 → 6 == 10 or 6 < 5 → False or False → **False**
- x != 7 and x > 3 → 6 != 7 and 6 > 3 → True and True → **True**
- not (x >= 5 and y <= 2) → not (6 >= 5 and 2 <= 2) → not (True and True) → not True → **False**
- (x < 0) or (x >= 0) → 6 < 0 or 6 >= 0 → False or True → **True** (always true, no matter what x is)
- not (x != x) → not False → **True**
- x > 1 and (z / x) != 2 → 6 > 1 and (12 / 6 != 2) → True and False → **False**
- x <= 0 or (z / x) == 1 → 6 <= 0 or (12 / 6 == 1) → False or False → **False**
- x == 4 and y == 4 → 6 == 4 and 2 == 4 → False and False → **False**