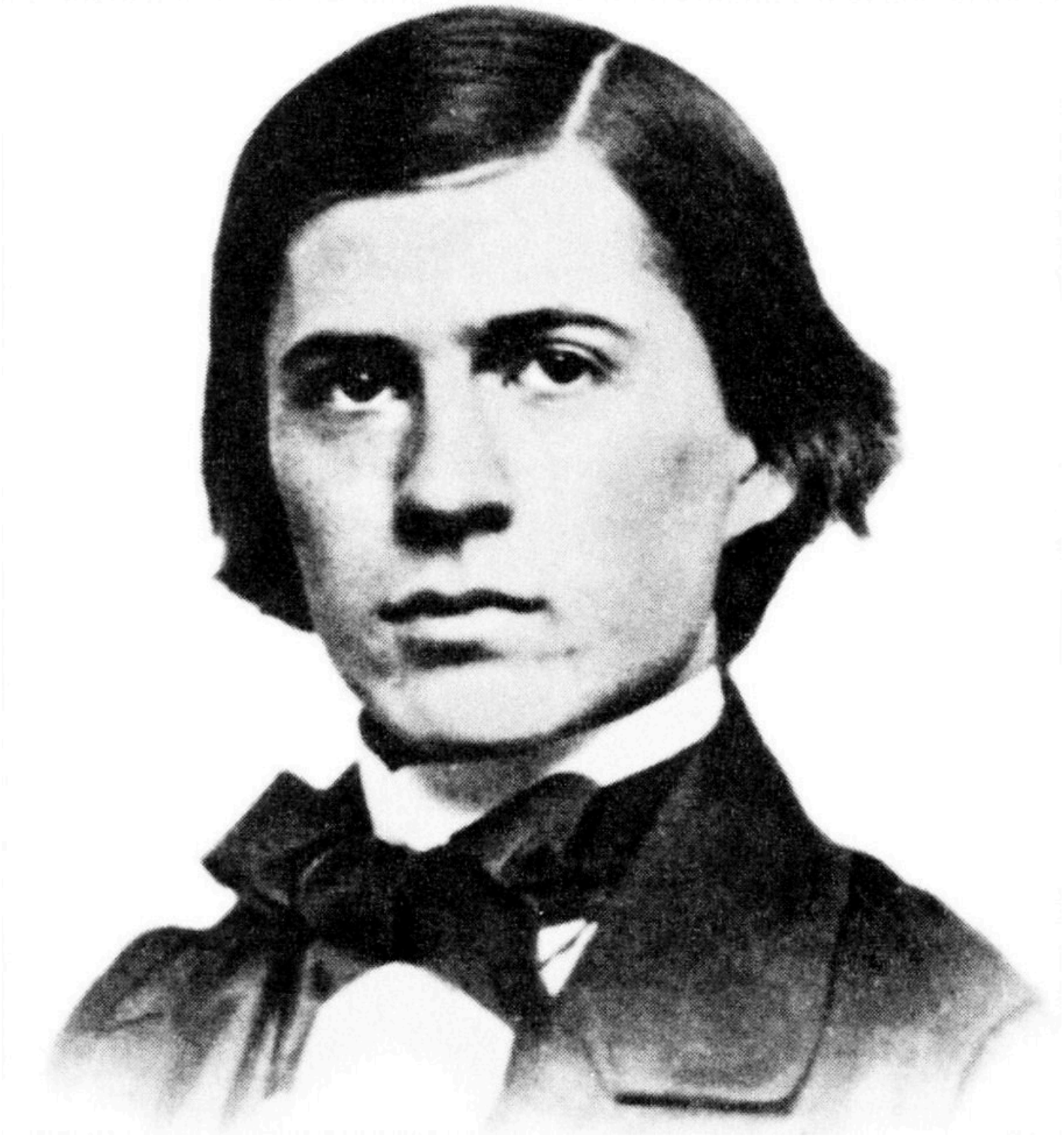
Logic



Learning Objectives

You will be able to:

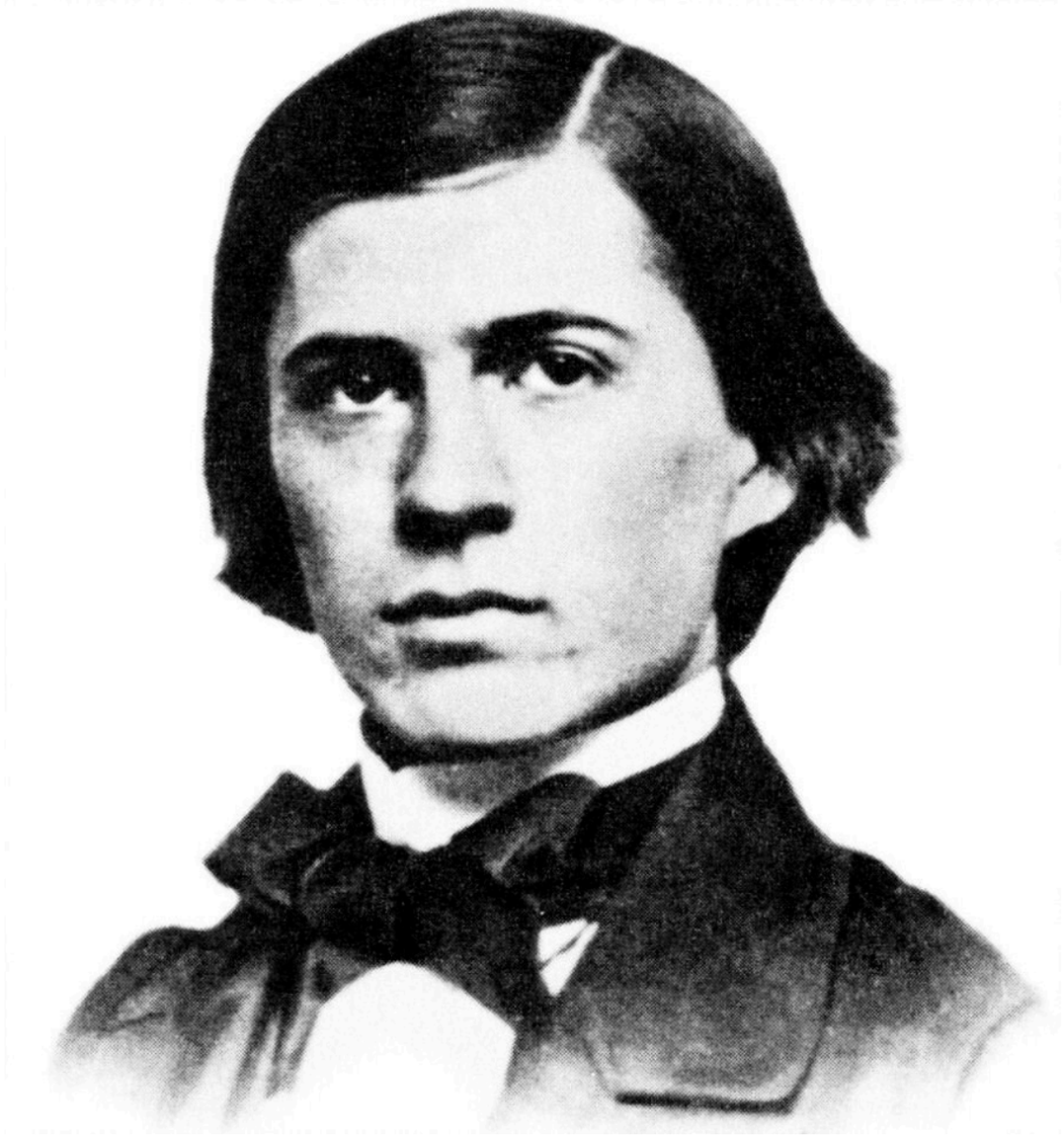
1. Build truth tables for simple logic gates
2. Build truth tables for simple logic expressions
3. Explain the duality between sets and logic operators
4. Build simple logic circuits in logic.ly

Logic

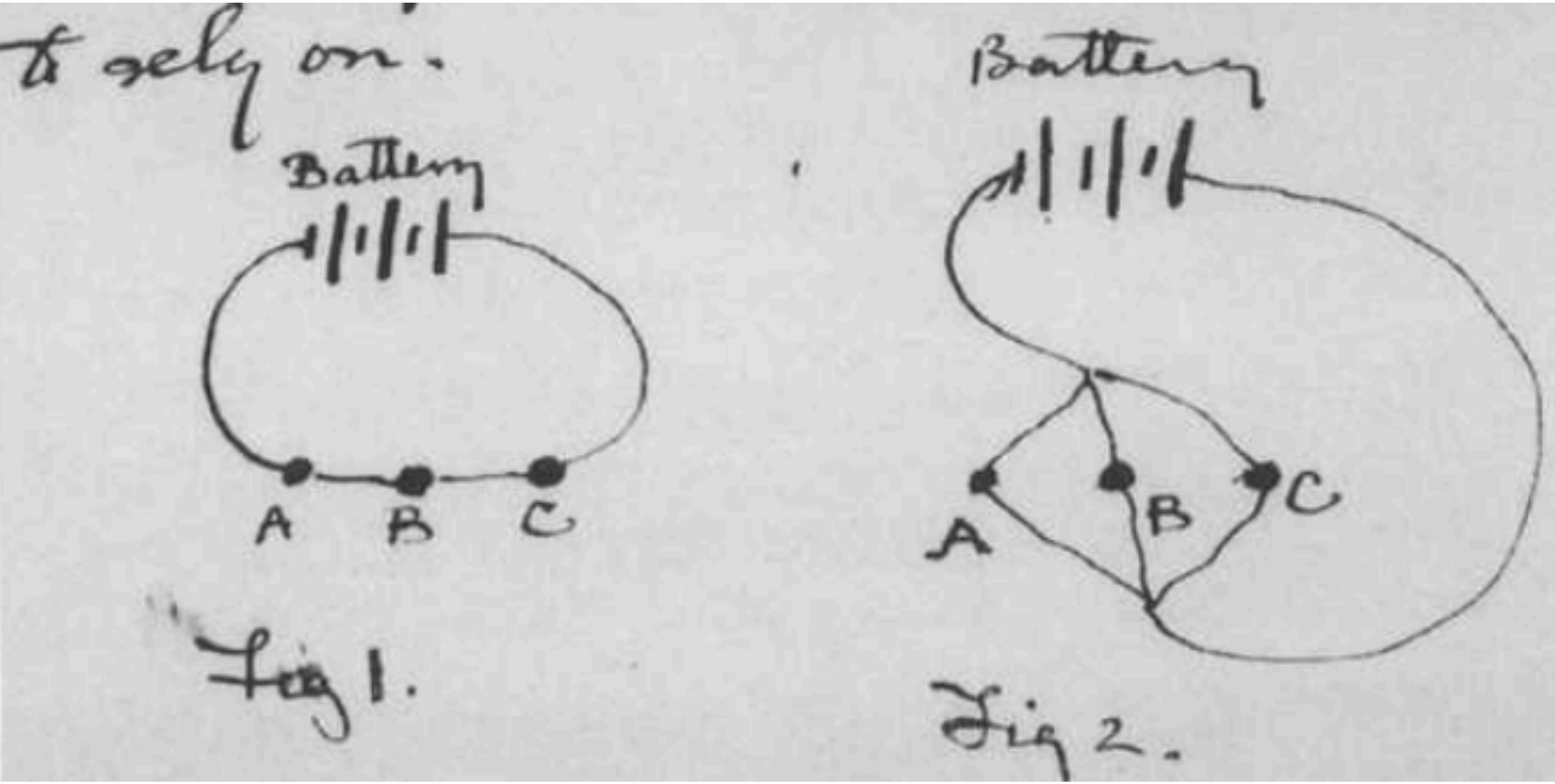
How do we manipulate 0s and 1s?

Boolean Algebra

Introduced by an English Mathematics George Boole professor while at University College Cork in 1854.



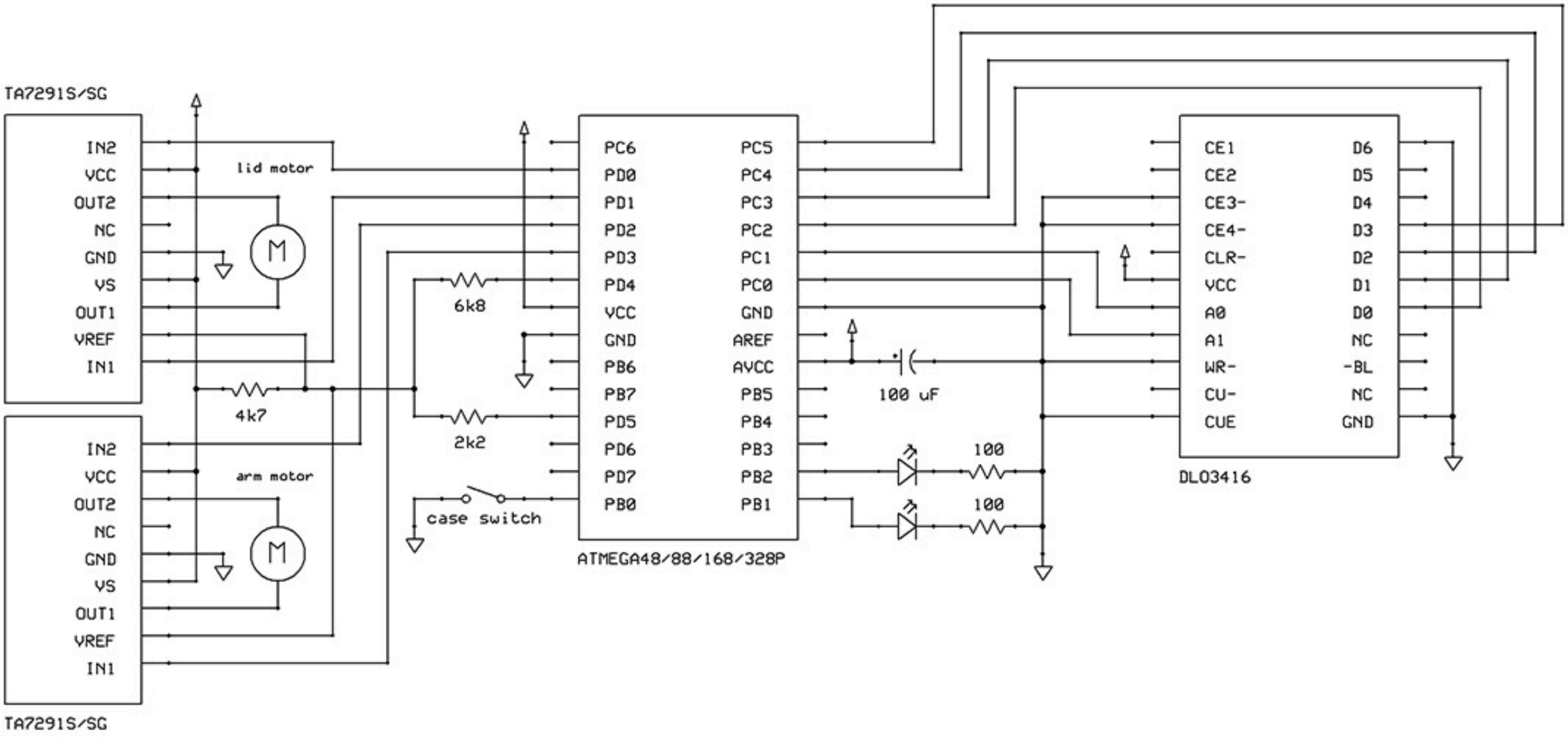
Charles Peirce 1886



Claude Shannon

https://www.youtube.com/watch?v=z7bVw7IMtUg

Aside: Shannon's Useless Machine



Basic Boolean operations

Values true and false (0 and 1)

Basic operations

- And (conjunction), x∧y, X AND Y
- Or (disjunction), x∨y, X OR Y
- Not (negation), ¬x, NOT X

Implemented using circuits (using logic gates)

Logic is like real life

If it is **dry** AND **warm**
I will go for a run.

Dry	Warm	Run
N	N	N
N	Y	N
Y	N	N
Y	Y	Y

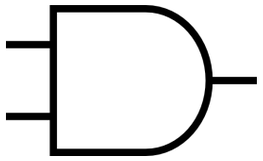
Dry	Warm	Run
0	0	0
0	1	0
1	0	0
1	1	1

If it is **wet** OR **windy**
I will skip school.

Wet	Wind	Skip
N	N	N
N	Y	Y
Y	N	Y
Y	Y	Y

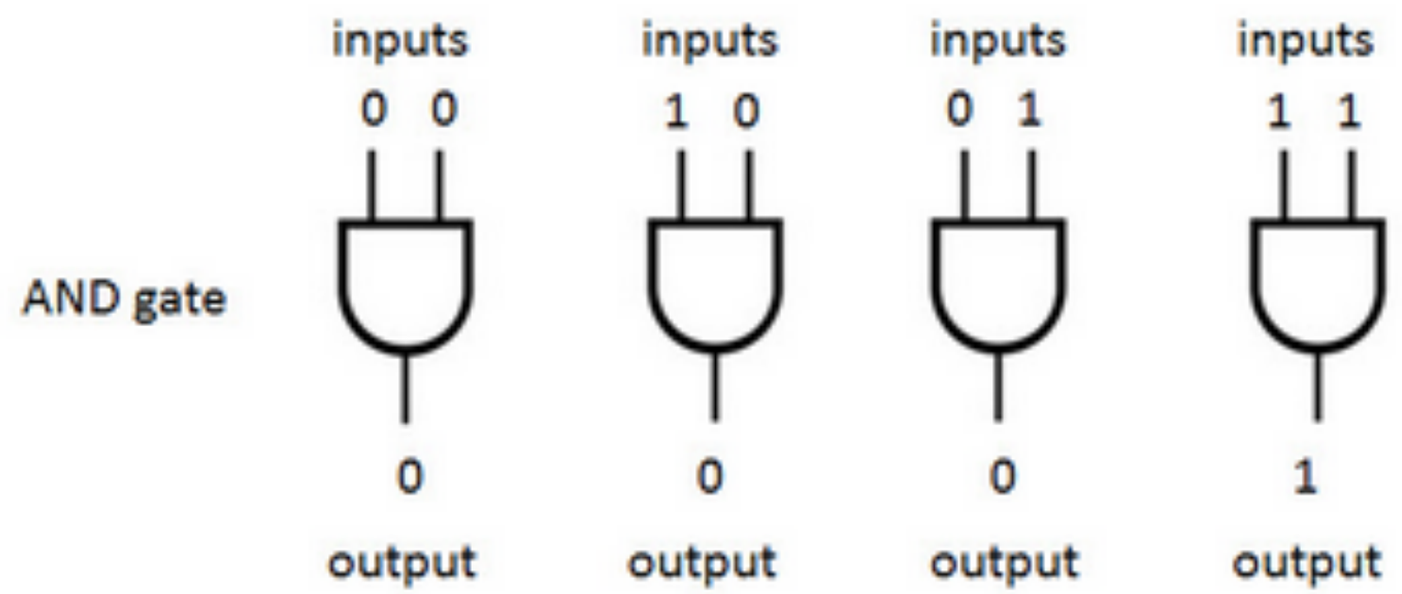
Wet	Wind	Skip
0	0	0
0	1	1
1	0	1
1	1	1

AND

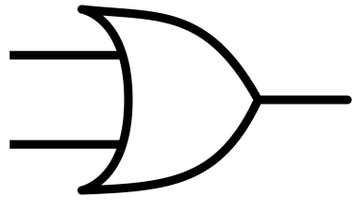


AND	0	1
0	0	0
1	0	1

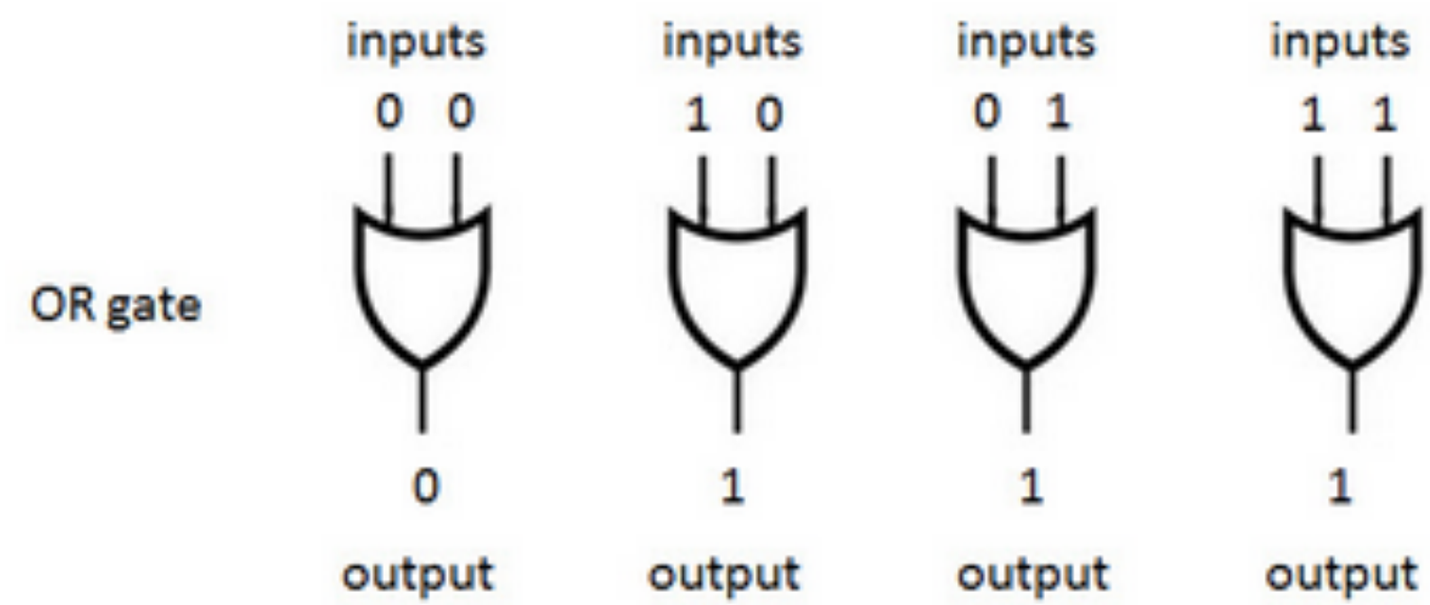
And in circuits



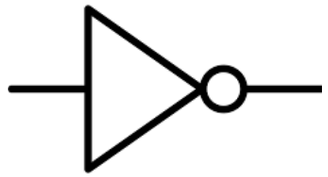
OR



OR	0	1
0	0	1
1	1	1



Not Gate



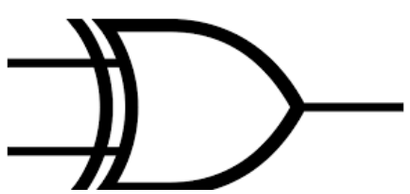
Output 0 if input 1 and vice versa

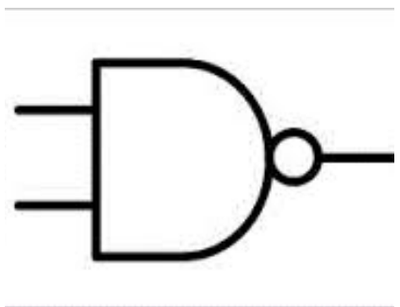
Also important

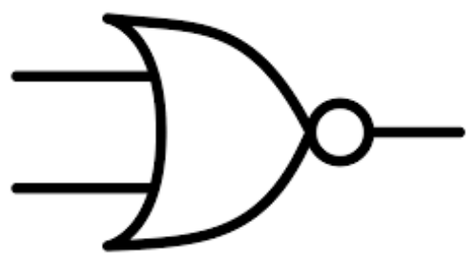
XOR	0	1
0	0	1
1	1	0

NAND	0	1
0	1	1
1	1	0

NOR	0	1
0	1	0
1	0	0





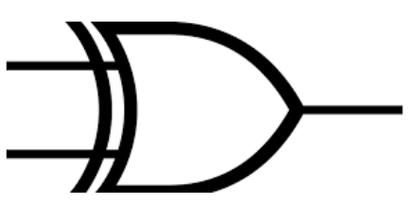


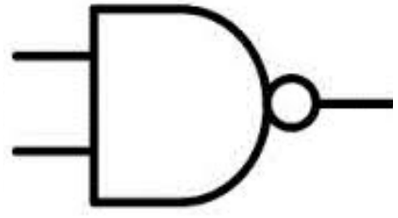
Also important

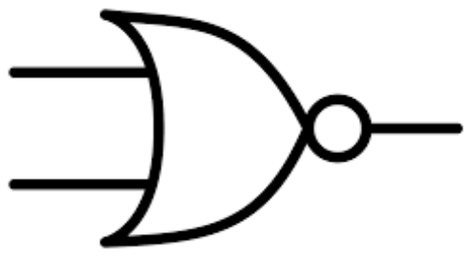
XOR	0	1
0	0	1
1	1	0

NAND	0	1
0	1	1
1	1	0

NOR	0	1
0	1	0
1	0	0







Logical vs Bitwise operations

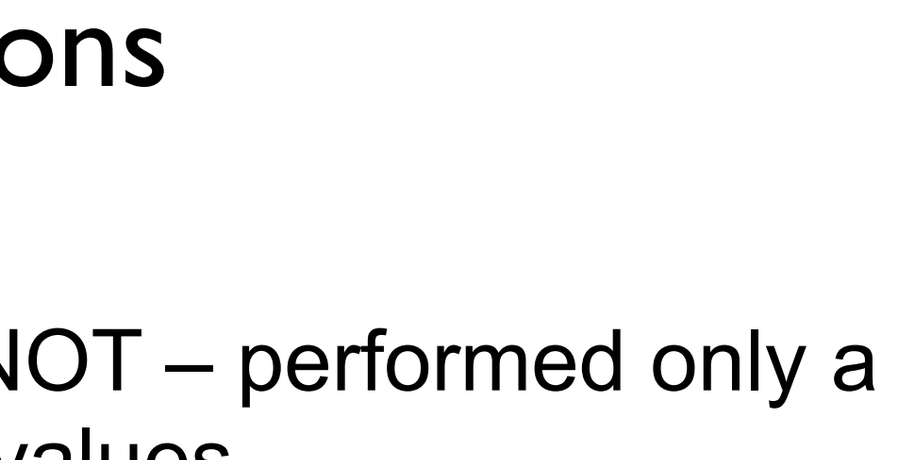
Logical operators: AND, OR, NOT – performed only a single pair of 0/1 or true/false values

Bitwise operators: AND, OR, NOT, etc, performed on all bits in a value individually, eg

In programming && vs &, II vs I

```
      0110
AND   1110
-----------
      0110
```

Note: different from addition!!

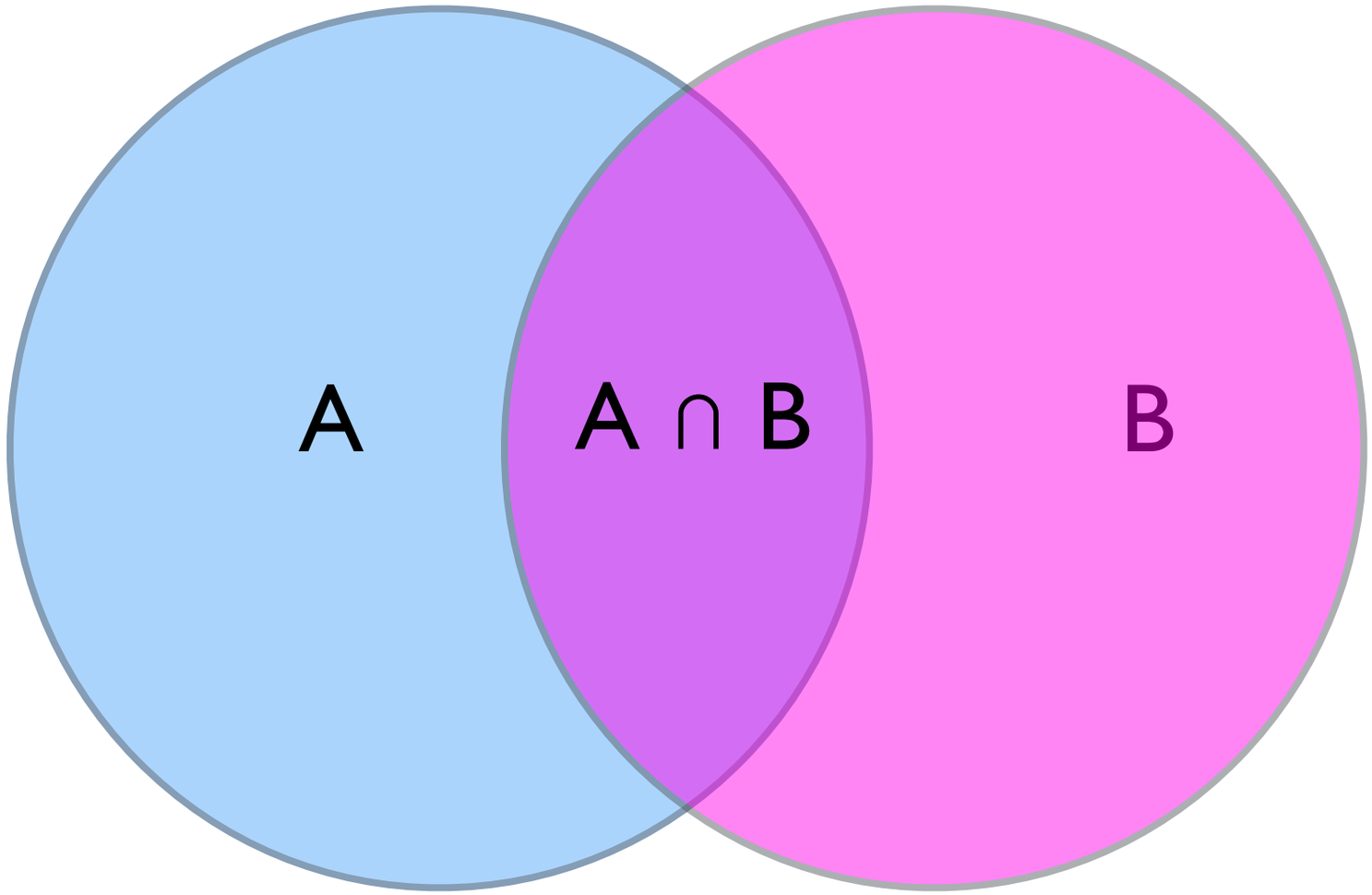


Logic and Sets

AND corresponds to set intersection

Logic ∧ corresponds to Set ∩

A ∧ B = A ∩ B

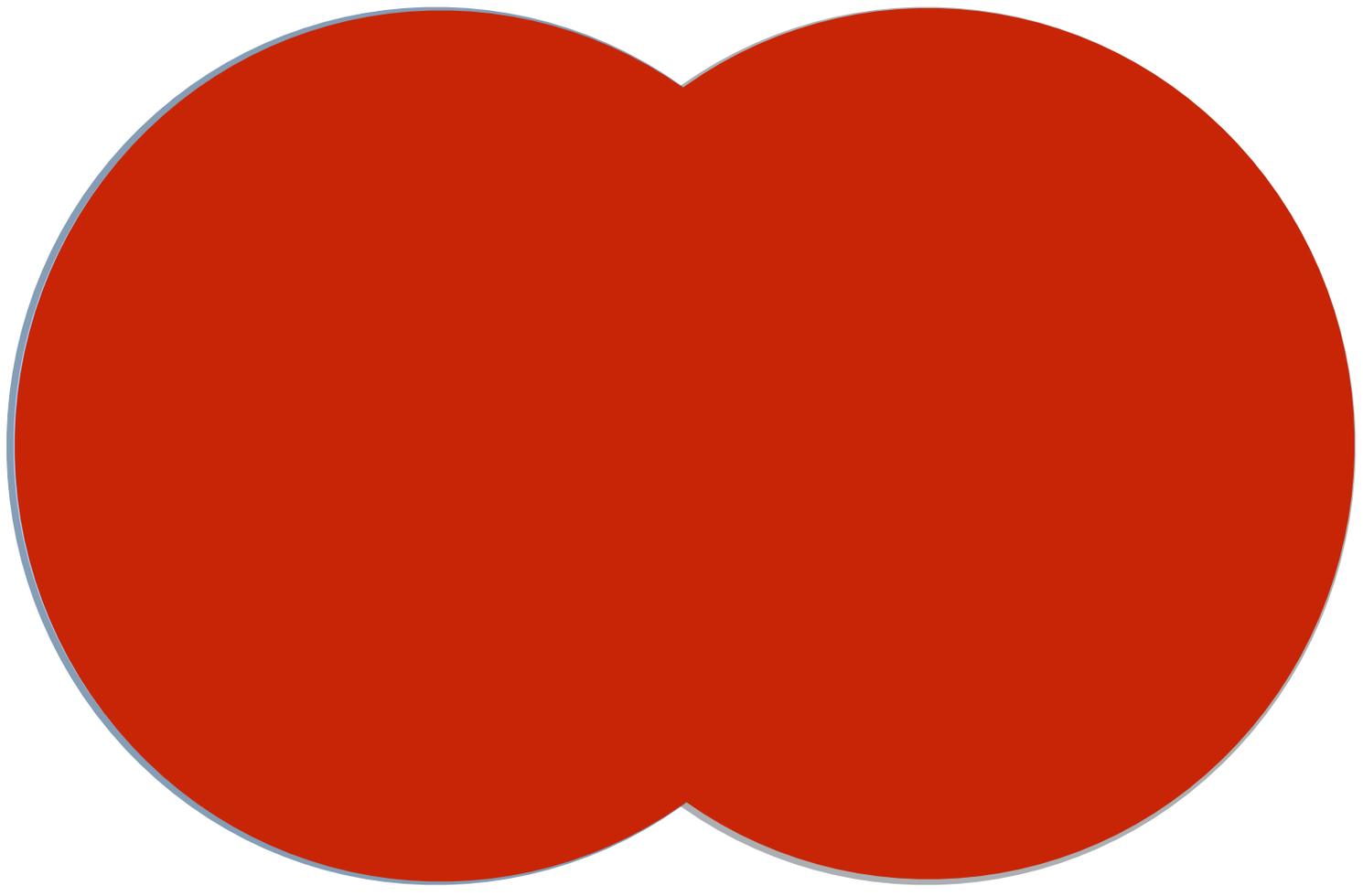


Logic and Sets

OR corresponds to set union

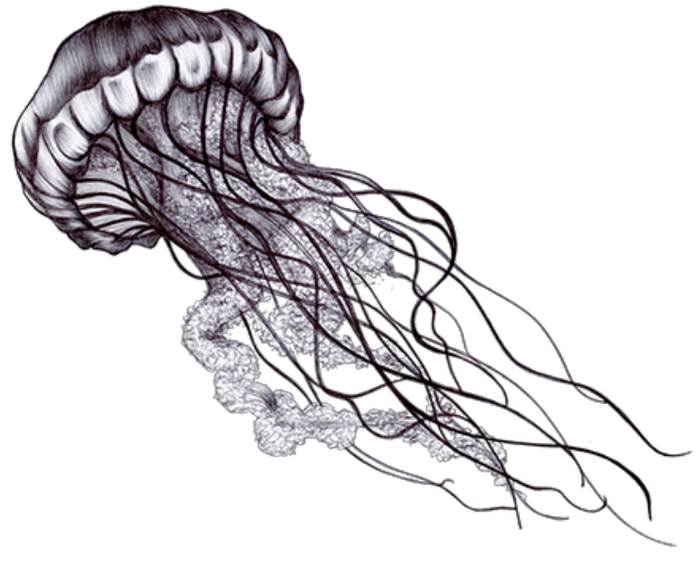
Logic $\vee$ corresponds to Set $\cup$

$A \vee B = A \cup B$



$A \cup B$

Truth Table Example



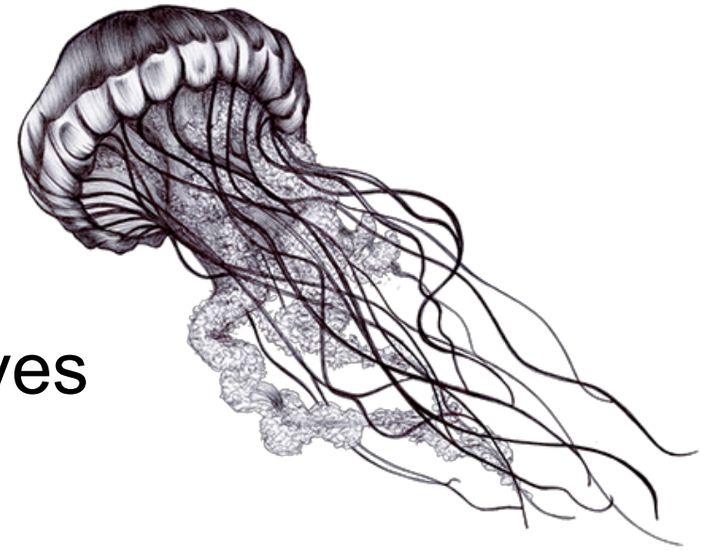
Swim on Sunny Days with No Jelly Fish or Big Waves

SD ∧ ¬(JF ∨ BW)

2 x 2 x 2 = 8 options

SD	JF	B W	JF ∨ BW	¬(JF ∨ BW)	Swim
1	1	1	1	0	0
1	1	0	1	0	0
1	0	1	1	0	0
1	0	0	0	1	1
0	1	1	1	0	0
0	1	0	1	0	0
0	0	1	1	0	0
0	0	0	0	1	0

Truth Table Example



Swim on Sunny Days with No Jelly Fish and No Big Waves

SD ∧ ¬JF ∧ ¬BW

SD	JF	B W	¬JF	¬BW	¬JF ∧ ¬BW	Swim
1	1	1				
1	1	0				
1	0	1				
1	0	0				
0	1	1				
0	1	0				
0	0	1				
0	0	0				

De Morgen's Laws

Not (**A** *and* **B**) is the same as *Not* **A** *or Not* **B**.

Not (**A** *or* **B**) is the same as *Not* **A** *and Not* **B**.

De Morgan's Laws apply to sets, propositions, or logic gates, the structure is always the same.

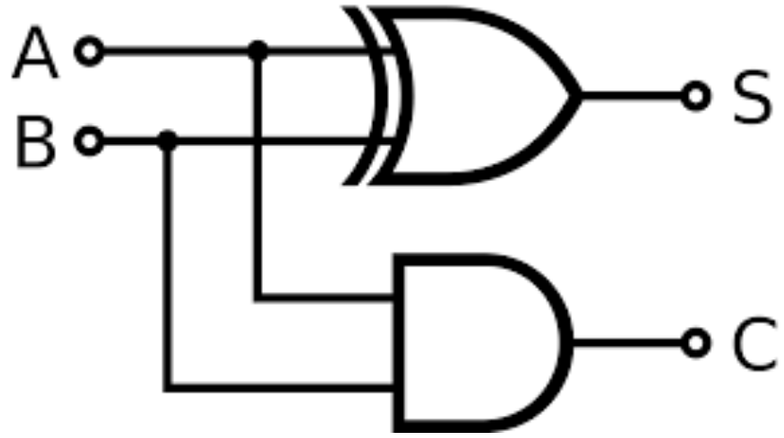
For more see:
https://brilliant.org/wiki/de-morgans-laws/

Logic Simulators

Online:
http://logic.ly/demo/
http://academo.org/demos/logic-gate-simulator/

Example – half adder

The **half adder** adds two single binary digits *A* and *B*. It has two outputs, sum (*S*) and carry (*C*).

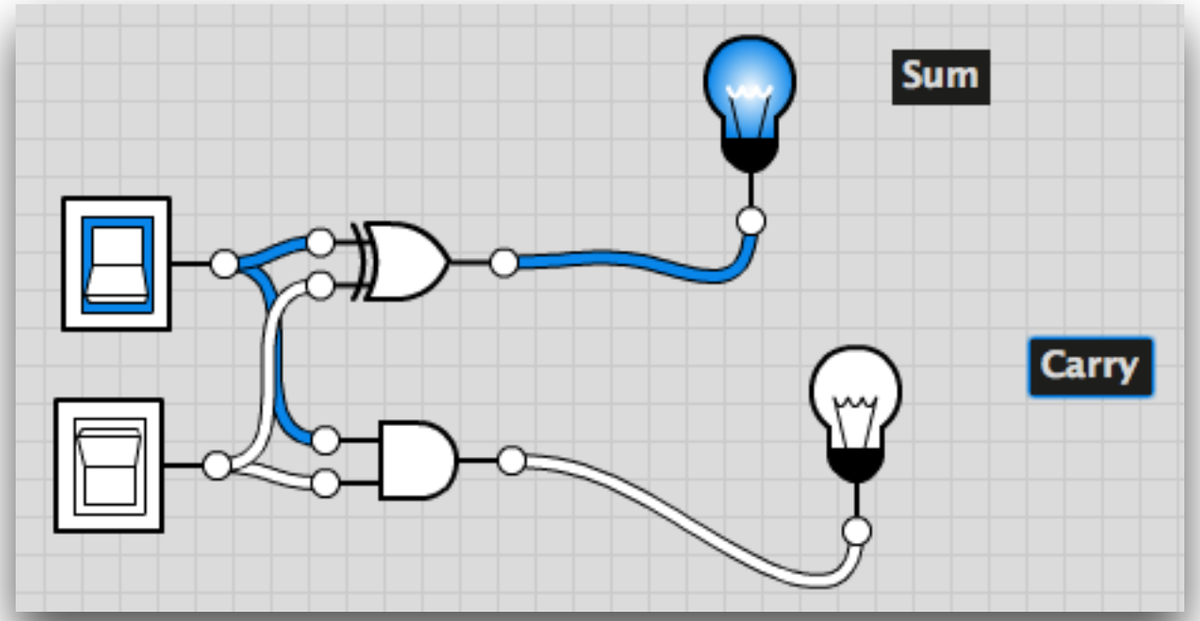


Exercises:

Write out the truth table for half adder

Simulate on logic.ly/demo

Half adder in logic.ly



logic.ly Swim decision

Create a logic.ly simulator to make the Swim decision

Swim on Sunny Days with No Jelly Fish or Big Waves

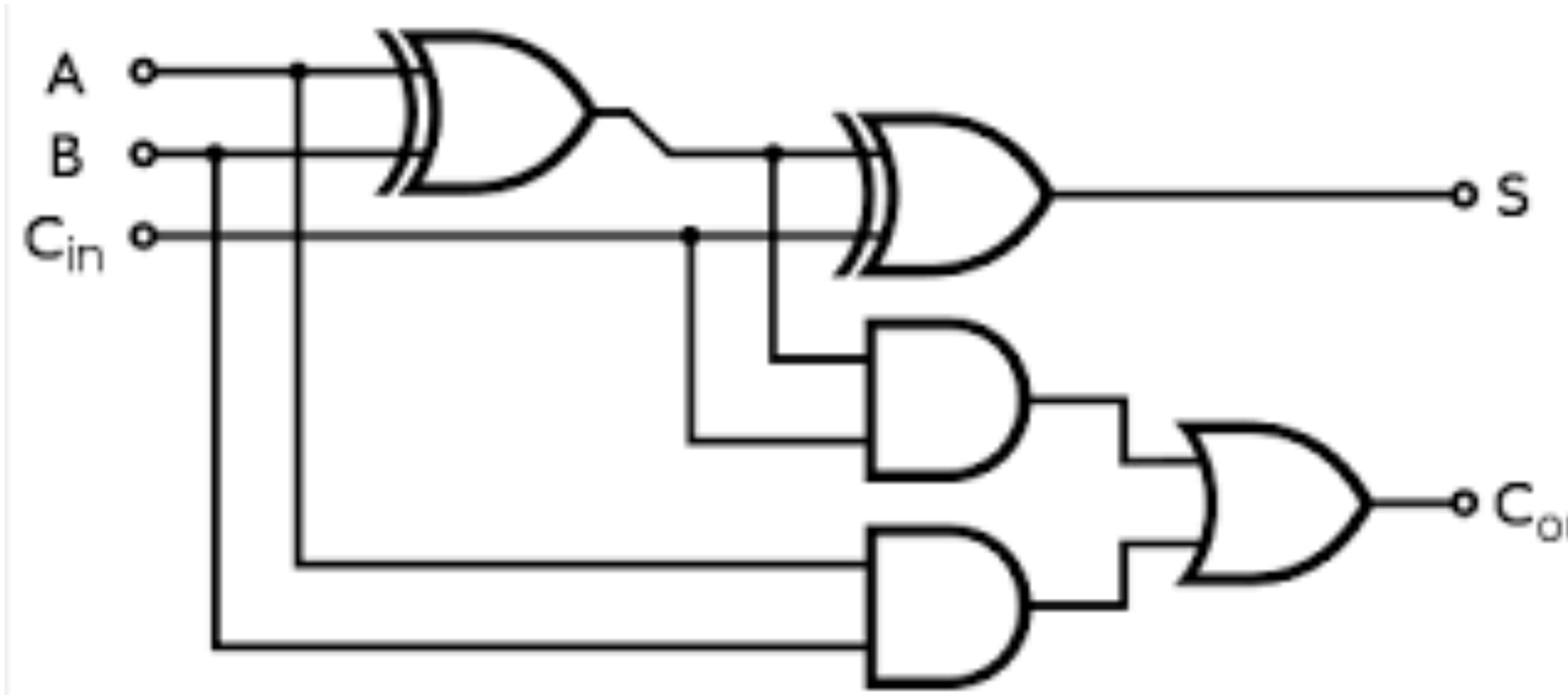
SD $\wedge$ $\neg$(JF $\vee$ BW)

And the other version

SD $\wedge$ $\neg$JF $\wedge$ $\neg$BW

Example – full adder

accounts for values carried in as well as out



Draw the truth table for full adder

Write the logic functions for S and C_{out}

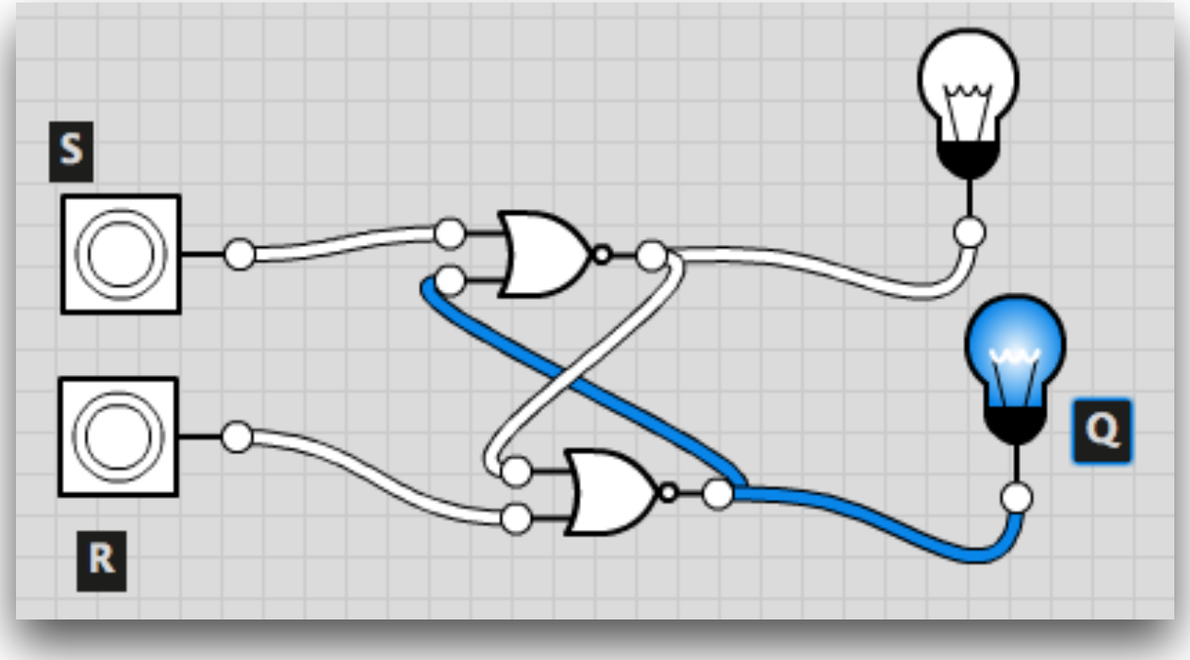
Flip Flop

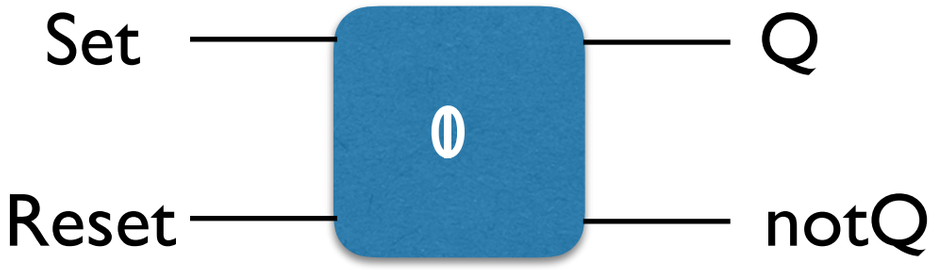
A circuit for saving state

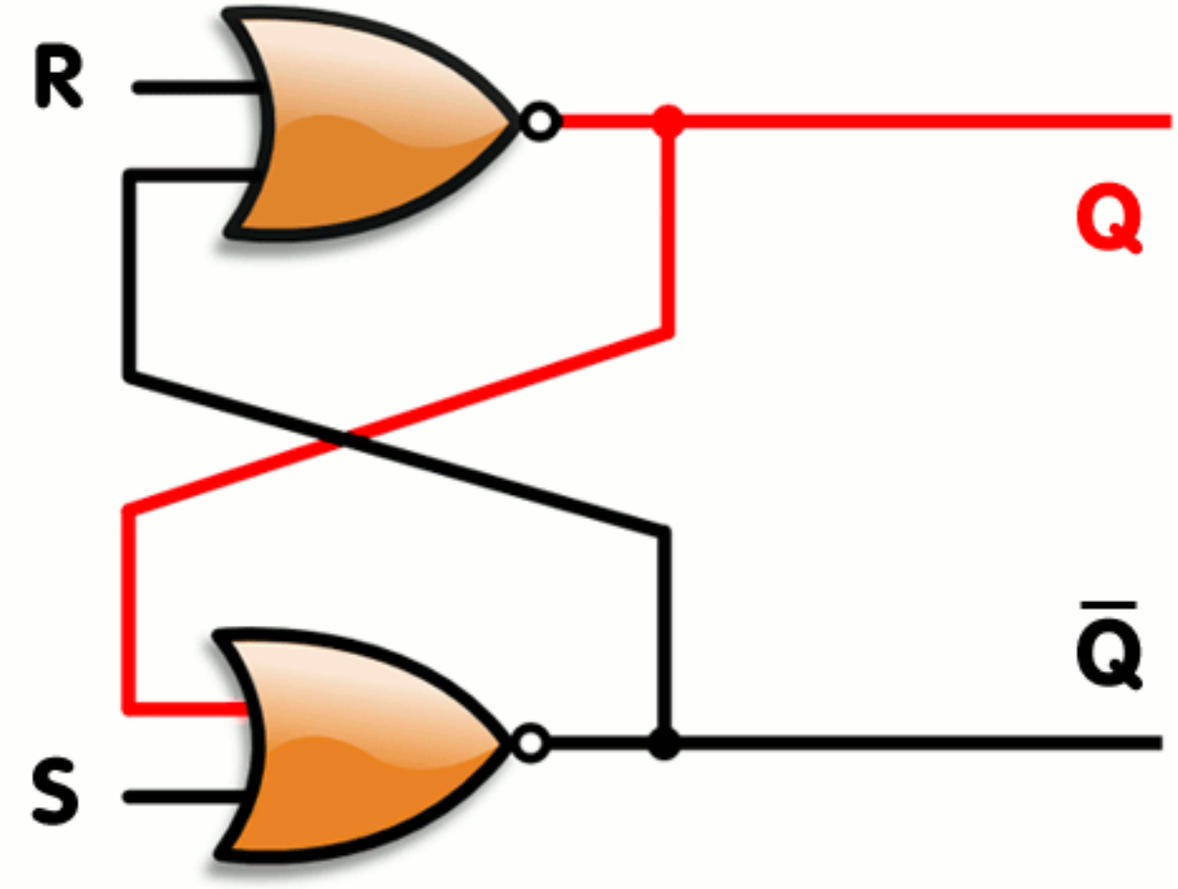
A simple memory cell

Simulate using logic.ly

Use push buttons rather than toggle switches







Other examples

Boolean algebra in search engines
- Macaroni AND cheese
- Macaroni -cheese
- Macaroni OR cheese

Programming (made up language)
- If (x==true) AND (y==true)
- If (x==true) OR (y==true)
- If NOT (x==true)

Short-circuiting

A $\wedge$ 1 (and)

A $\vee$ 1 (or)

Second argument is executed or evaluated only if the first argument does not suffice to determine the value of the expression

In AND, if first one is 0/false, no need to evaluate other one, as result is always 0/false

In OR, if first one is 1/true, no need to evaluate other one, as result is always 1/true

Optimisation technique

Learning Objectives

Be able to:

- Build truth tables for simple logic gates
- Build truth tables for simple logic expressions
- Explain the duality between sets and logic operators
- Build simple logic circuits in logic.ly