Unit 1: Logic & Gates

Dave Abel

February 1st, 2016

Takeaway: Logic

(1) There is a notion of correct reasoning (Logic)

(2) Computers automate this reasoning process

Outline of Unit

- Quick Review of Binary
- Physical Bits!
- Ambiguity of Regular Languages
- Logical Inference
- Logical Functions: *AND*, *OR, NOT*
- Truth Tables

Reading Binary: 27

1. Start with the biggest power of 2 no bigger than your number.

2. Write down a 1 in that power of two's column.

3. Subtract that power of 2 from your number.

4. Go to the next smallest power of 2.

5. If your remaining number is greater than or equal to the power of 2, write down a 1 and subtract the power of 2.

6. If not, write down 0.

7. Repeat from [4].

16	8	4	2	1

Our Number: Twenty Seven

Reading Binary: 27

1. Start with the biggest power of 2 no bigger than your number.

2. Write down a 1 in that power of two's column.

3. Subtract that power of 2 from your number.

4. Go to the next smallest power of 2.

5. If your remaining number is greater than or equal to the power of 2, write down a 1 and subtract the power of 2.

6. If not, write down 0.

7. Repeat from [4].

16	8	4	2	1

Our Number: Twenty Seven

Reading Binary: 27

1. Start with the biggest power of 2 no bigger than your number.
2. **Write down a 1 in that power of two's column.**
3. Subtract that power of 2 from your number.
4. Go to the next smallest power of 2.
5. If your remaining number is greater than or equal to the power of 2, write down a 1 and subtract the power of 2.
6. If not, write down 0.
7. Repeat from [4].

1				
16	8	4	2	1

Our Number: Twenty Seven

Reading Binary: 27

1. Start with the biggest power of 2 no bigger than your number.

2. Write down a 1 in that power of two's column.

3. Subtract that power of 2 from your number.

4. Go to the next smallest power of 2.

5. If your remaining number is greater than or equal to the power of 2, write down a 1 and subtract the power of 2.

6. If not, write down 0.

7. Repeat from [4].

1				
16	8	4	2	1

Our Number: Twenty Seven

Reading Binary: 27

1. Start with the biggest power of 2 no bigger than your number.

2. Write down a 1 in that power of two's column.

3. Subtract that power of 2 from your number.

4. Go to the next smallest power of 2.

5. If your remaining number is greater than or equal to the power of 2, write down a 1 and subtract the power of 2.

6. If not, write down 0.

7. Repeat from [4].

1				
16	8	4	2	1

Our Number: Eleven

Reading Binary: 27

1. Start with the biggest power of 2 no bigger than your number.
2. Write down a 1 in that power of two's column.
3. Subtract that power of 2 from your number.
4. **Go to the next smallest power of 2.**
5. If your remaining number is greater than or equal to the power of 2, write down a 1 and subtract the power of 2.
6. If not, write down 0.
7. Repeat from [4].

1				
16	8	4	2	1

Our Number: Eleven

Reading Binary: 27

1. Start with the biggest power of 2 no bigger than your number.
2. Write down a 1 in that power of two's column.
3. Subtract that power of 2 from your number.
4. Go to the next smallest power of 2.
5. **If your remaining number is greater than or equal to the power of 2, write down a 1 and subtract the power of 2.**
6. If not, write down 0.
7. Repeat from [4].

1				
16	8	4	2	1

Our Number: Eleven

Reading Binary: 27

1. Start with the biggest power of 2 no bigger than your number.
2. Write down a 1 in that power of two's column.
3. Subtract that power of 2 from your number.
4. Go to the next smallest power of 2.
5. **If your remaining number is greater than or equal to the power of 2, write down a 1 and subtract the power of 2.**
6. If not, write down 0.
7. Repeat from [4].

1	1			
16	8	4	2	1

Our Number: Eleven

Reading Binary: 27

1. Start with the biggest power of 2 no bigger than your number.
2. Write down a 1 in that power of two's column.
3. Subtract that power of 2 from your number.
4. Go to the next smallest power of 2.
5. **If your remaining number is greater than or equal to the power of 2, write down a 1 and subtract the power of 2.**
6. If not, write down 0.
7. Repeat from [4].

1	1			
16	8	4	2	1

Our Number: Three

Reading Binary: 27

1. Start with the biggest power of 2 no bigger than your number.
2. Write down a 1 in that power of two's column.
3. Subtract that power of 2 from your number.
4. Go to the next smallest power of 2.
5. If your remaining number is greater than or equal to the power of 2, write down a 1 and subtract the power of 2.
6. If not, write down 0.
7. **Repeat from [4].**

1	1			
16	8	4	2	1

Our Number: Three

Reading Binary: 27

1. Start with the biggest power of 2 no bigger than your number.
2. Write down a 1 in that power of two's column.
3. Subtract that power of 2 from your number.
4. **Go to the next smallest power of 2.**
5. If your remaining number is greater than or equal to the power of 2, write down a 1 and subtract the power of 2.
6. If not, write down 0.
7. Repeat from [4].

1	1			
16	8	4	2	1

Our Number: Three

Reading Binary: 27

1. Start with the biggest power of 2 no bigger than your number.

2. Write down a 1 in that power of two's column.

3. Subtract that power of 2 from your number.

4. Go to the next smallest power of 2.

5. If your remaining number is greater than or equal to the power of 2, write down a 1 and subtract the power of 2.

6. If not, write down 0.

7. Repeat from [4].

1	1			
16	8	4	2	1

Our Number: Three

Reading Binary: 27

1. Start with the biggest power of 2 no bigger than your number.
2. Write down a 1 in that power of two's column.
3. Subtract that power of 2 from your number.
4. Go to the next smallest power of 2.
5. **If your remaining number is greater than or equal to the power of 2, write down a 1 and subtract the power of 2.**
6. If not, write down 0.
7. Repeat from [4].

1	1			
16	8	4	2	1

Our Number: Three

Reading Binary: 27

1. Start with the biggest power of 2 no bigger than your number.
2. Write down a 1 in that power of two's column.
3. Subtract that power of 2 from your number.
4. Go to the next smallest power of 2.
5. If your remaining number is greater than or equal to the power of 2, write down a 1 and subtract the power of 2.
6. **If not, write down 0.**
7. Repeat from [4].

1	1	0		
16	8	4	2	1

Our Number: Three

Reading Binary: 27

1. Start with the biggest power of 2 no bigger than your number.
2. Write down a 1 in that power of two's column.
3. Subtract that power of 2 from your number.
4. Go to the next smallest power of 2.
5. If your remaining number is greater than or equal to the power of 2, write down a 1 and subtract the power of 2.
6. If not, write down 0.

7. Repeat from [4].

1	1	0		
16	8	4	2	1

Our Number: Three

Reading Binary: 27

1. Start with the biggest power of 2 no bigger than your number.

2. Write down a 1 in that power of two’s column.

3. Subtract that power of 2 from your number.

4. Go to the next smallest power of 2.

5. If your remaining number is greater than or equal to the power of 2, write down a 1 and subtract the power of 2.

6. If not, write down 0.

7. Repeat from [4].

1	1	0		
16	8	4	2	1

Our Number: Three

Reading Binary: 27

1. Start with the biggest power of 2 no bigger than your number.
2. Write down a 1 in that power of two's column.
3. Subtract that power of 2 from your number.
4. **Go to the next smallest power of 2.**
5. If your remaining number is greater than or equal to the power of 2, write down a 1 and subtract the power of 2.
6. If not, write down 0.
7. Repeat from [4].

1	1	0		
16	8	4	2	1

Our Number: Three

Reading Binary: 27

1. Start with the biggest power of 2 no bigger than your number.
2. Write down a 1 in that power of two's column.
3. Subtract that power of 2 from your number.
4. Go to the next smallest power of 2.
5. **If your remaining number is greater than or equal to the power of 2, write down a 1 and subtract the power of 2.**
6. If not, write down 0.
7. Repeat from [4].

1	1	0	1	
16	8	4	2	1

Our Number: Three

Reading Binary: 27

1. Start with the biggest power of 2 no bigger than your number.
2. Write down a 1 in that power of two's column.
3. Subtract that power of 2 from your number.
4. Go to the next smallest power of 2.
5. If your remaining number is greater than or equal to the power of 2, write down a 1 and subtract the power of 2.
6. If not, write down 0.
7. **Repeat from [4].**

1	1	0	1	
16	8	4	2	1

Our Number: One

Reading Binary: 27

1. Start with the biggest power of 2 no bigger than your number.
2. Write down a 1 in that power of two's column.
3. Subtract that power of 2 from your number.
4. **Go to the next smallest power of 2.**
5. If your remaining number is greater than or equal to the power of 2, write down a 1 and subtract the power of 2.
6. If not, write down 0.
7. Repeat from [4].

1	1	0	1	
16	8	4	2	1

Our Number: One

Reading Binary: 27

1. Start with the biggest power of 2 no bigger than your number.
2. Write down a 1 in that power of two's column.
3. Subtract that power of 2 from your number.
4. **Go to the next smallest power of 2.**
5. If your remaining number is greater than or equal to the power of 2, write down a 1 and subtract the power of 2.
6. If not, write down 0.
7. Repeat from [4].

1	1	0	1	
16	8	4	2	1

Our Number: One

Reading Binary: 27

1. Start with the biggest power of 2 no bigger than your number.
2. Write down a 1 in that power of two's column.
3. Subtract that power of 2 from your number.
4. Go to the next smallest power of 2.
5. **If your remaining number is greater than or equal to the power of 2, write down a 1 and subtract the power of 2.**
6. If not, write down 0.
7. Repeat from [4].

1	1	0	1	
16	8	4	2	1

Our Number: One

Reading Binary: 27

1. Start with the biggest power of 2 no bigger than your number.
2. Write down a 1 in that power of two's column.
3. Subtract that power of 2 from your number.
4. Go to the next smallest power of 2.
5. **If your remaining number is greater than or equal to the power of 2, write down a 1 and subtract the power of 2.**
6. If not, write down 0.
7. Repeat from [4].

1	1	0	1	1
16	8	4	2	1

Our Number: Zero

Reading Binary: 27

1. Start with the biggest power of 2 no bigger than your number.
2. Write down a 1 in that power of two's column.
3. Subtract that power of 2 from your number.
4. Go to the next smallest power of 2.
5. **If your remaining number is greater than or equal to the power of 2, write down a 1 and subtract the power of 2.**
6. If not, write down 0.
7. Repeat from [4].

1	1	0	1	1
16	8	4	2	1

Our Number: Zero ✓

Review: Binary Addition

$$\begin{array}{r@{\quad}cccc} & 1 & 0 & 0 & 1 \\ + & 1 & 0 & 1 & 0 \\ \hline \end{array}$$

Review: Binary Addition

$$
\begin{array}{rcccc}
 & 1 & 0 & 0 & 1 \\
+ & 1 & 0 & 1 & 0 \\
\hline
 & & & & 1
\end{array}
$$

Review: Binary Addition

$$
\begin{array}{rcccc}
 & 1 & 0 & 0 & 1 \\
+ & 1 & 0 & 1 & 0 \\
\hline
 & & & 1 & 1
\end{array}
$$

Review: Binary Addition

$$
\begin{array}{rcccc}
 & 1 & 0 & 0 & 1 \\
+ & 1 & 0 & 1 & 0 \\
\hline
 & & 0 & 1 & 1
\end{array}
$$

Review: Binary Addition

$$\begin{array}{rcccc} 1 & & & & \\ & 1 & 0 & 0 & 1 \\ + & 1 & 0 & 1 & 0 \\ \hline & 0 & 0 & 1 & 1 \end{array}$$

Review: Binary Addition

$$\begin{array}{cccccc}
 & 1 & & & & \\
 & & 1 & 0 & 0 & 1 \\
+ & & 1 & 0 & 1 & 0 \\
\hline
 & 1 & 0 & 0 & 1 & 1
\end{array}$$

Review: Binary Addition

$$
\begin{array}{rrrrr}
1 & & & & \\
 & 1 & 0 & 0 & 1 \\
+ & 1 & 0 & 1 & 0 \\
\hline
1 & 0 & 0 & 1 & 1
\end{array}
$$

Sanity check: nine + ten = nineteen.

Review: Binary Subtraction

A:

$$\begin{array}{rr} & 1 \\ - & 0 \\ \hline & 1 \end{array}$$

B:

$$\begin{array}{rr} & 1 \\ - & 1 \\ \hline & 0 \end{array}$$

C:

$$\begin{array}{rr} & 0 \\ - & 0 \\ \hline & 0 \end{array}$$

Review: Binary Subtraction

			1
D:			1
		1	0
	-	0	1
			1

Review: Binary Subtraction

$$\begin{array}{ccccc} & 1 & 1 & 0 & 1 \\ - & 0 & 0 & 1 & 0 \\ \hline \end{array}$$

Review: Binary Subtraction

$$\begin{array}{ccccc} & 1 & 1 & 0 & 1 \\ - & 0 & 0 & 1 & 0 \\ \hline & & & & 0 \end{array}$$

Review: Binary Subtraction

$$\begin{array}{ccccc} & & & 1 & \\ & & & 1 & \\ & 1 & \cancel{1} & 0 & 1 \\ - & 0 & 0 & 1 & 0 \\ \hline & & & & 0 \end{array}$$

Review: Binary Subtraction

$$
\begin{array}{ccccc}
 & & & 1 & \\
 & & & 1 & \\
 & 1 & \cancel{1} & 0 & 1 \\
- & 0 & 0 & 1 & 0 \\
\hline
 & & & 1 & 0
\end{array}
$$

Review: Binary Subtraction

$$
\begin{array}{rcccc}
 & & & 1 & \\
 & & & 1 & \\
 & 1 & 0 & 0 & 1 \\
- & 0 & 0 & 1 & 0 \\
\hline
 & & & 1 & 0
\end{array}
$$

Review: Binary Subtraction

$$
\begin{array}{rcccc}
 & & & 1 & \\
 & & & 1 & \\
 & 1 & 0 & 0 & 1 \\
- & 0 & 0 & 1 & 0 \\
\hline
 & & & 1 & 0
\end{array}
$$

Review: Binary Subtraction

$$
\begin{array}{ccccc}
 & & & 1 & \\
 & & & 1 & \\
 & 1 & 0 & 0 & 1 \\
- & 0 & 0 & 1 & 0 \\
\hline
 & & 0 & 1 & 0
\end{array}
$$

Review: Binary Subtraction

$$\begin{array}{ccccc} & & & 1 & \\ & & & 1 & \\ & 1 & 0 & 0 & 1 \\ - & 0 & 0 & 1 & 0 \\ \hline & 1 & 0 & 1 & 0 \end{array}$$

Sanity check: nine - two = seven

Review: Binary Multiplication

- Idea: move over each bit in bottom number, if there's a 1, add the entire top number, otherwise, shift over and repeat:

$$
\begin{array}{rccc}
 & 1 & 1 & 1 \\
* & 0 & 1 & \boxed{0} \\
\hline
\end{array}
$$

Review: Binary Multiplication

- Idea: move over each bit in bottom number, if there's a 1, add the entire top number, otherwise, shift over and repeat:

$$
\begin{array}{cccc}
 & 1 & 1 & 1 \\
* & 0 & 1 & \boxed{0} \\
\hline
 & 0 & 0 & 0
\end{array}
$$

(Recall: we're really just multiplying by 0)

Review: Binary Multiplication

- Idea: move over each bit in bottom number, if there's a 1, add the entire top number, otherwise, shift over and repeat:

$$
\begin{array}{cccc}
 & 1 & 1 & 1 \\
* & 0 & \boxed{1} & 0 \\
\hline
1 & 1 & 1 &
\end{array}
$$

Review: Binary Multiplication

- Idea: move over each bit in bottom number, if there's a 1, add the entire top number, otherwise, shift over and repeat:

$$
\begin{array}{rrrr}
 & 1 & 1 & 1 \\
* & \boxed{0} & 1 & 0 \\
\hline
1 & 1 & 1 &
\end{array}
$$

Review: Binary Multiplication

- Idea: move over each bit in bottom number, if there's a 1, add the entire top number, otherwise, shift over and repeat:

$$\begin{array}{cccc} & 1 & 1 & 1 \\ * & \boxed{0} & 1 & 0 \\ \hline 1 & 1 & 1 & 0 \end{array}$$

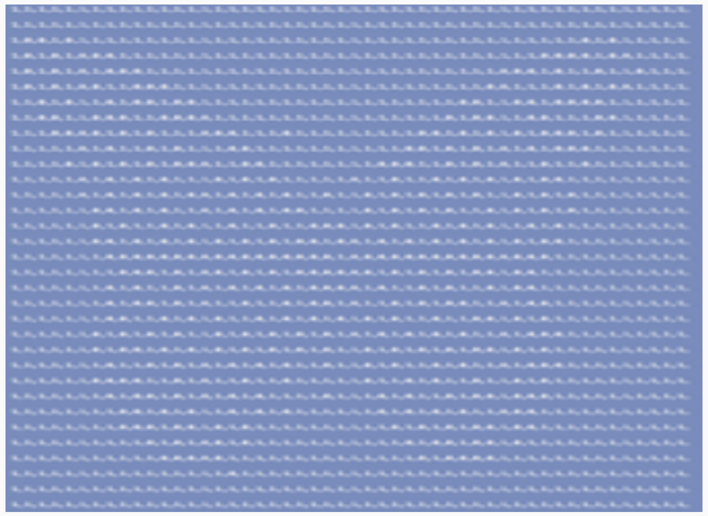
Review: Binary Multiplication

- Idea: move over each bit in bottom number, if there's a 1, add the entire top number, otherwise, shift over and repeat:

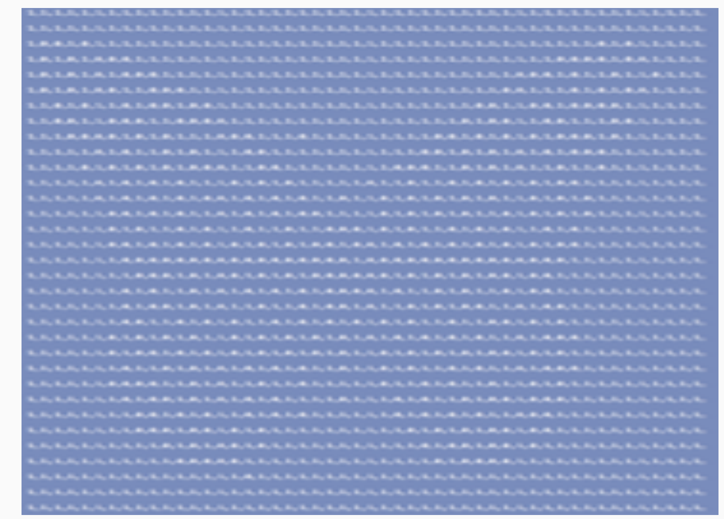
$$\begin{array}{cccc} & 1 & 1 & 1 \\ * & \boxed{0} & 1 & 0 \\ \hline 1 & 1 & 1 & 0 \end{array}$$

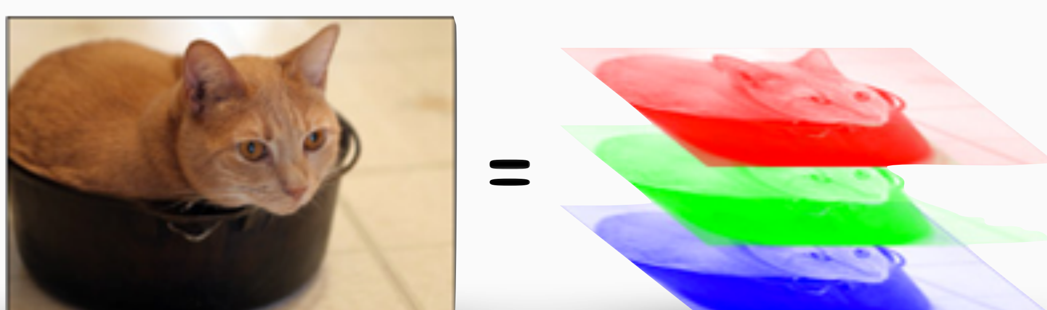
Sanity Check: seven * two = fourteen

Review: Representation



Review: Representation



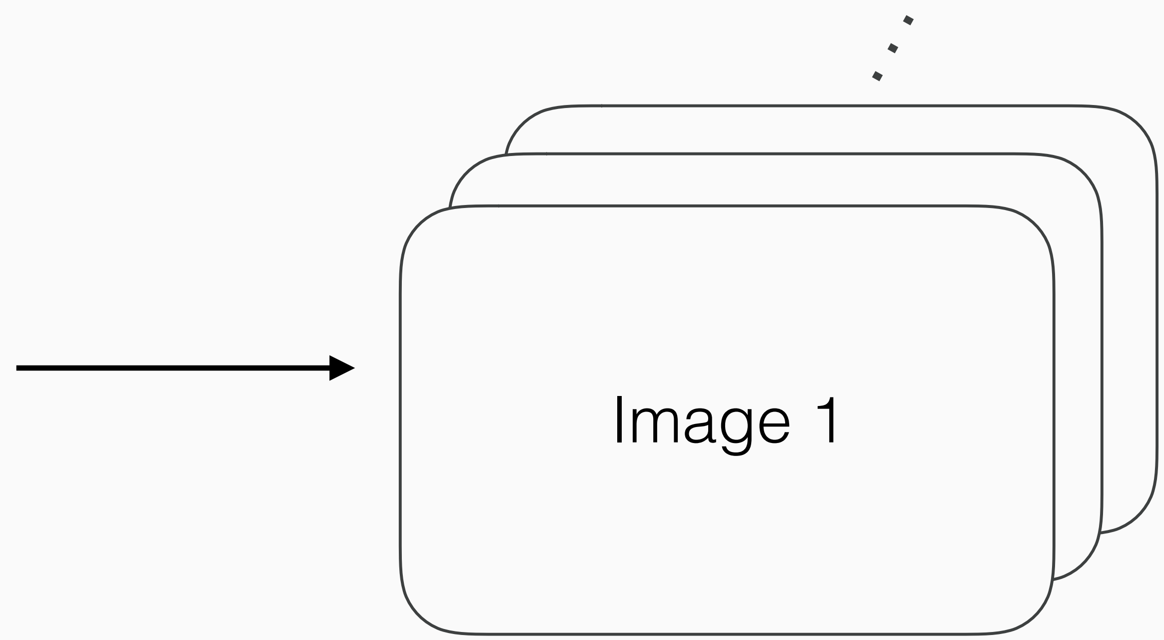


red bits

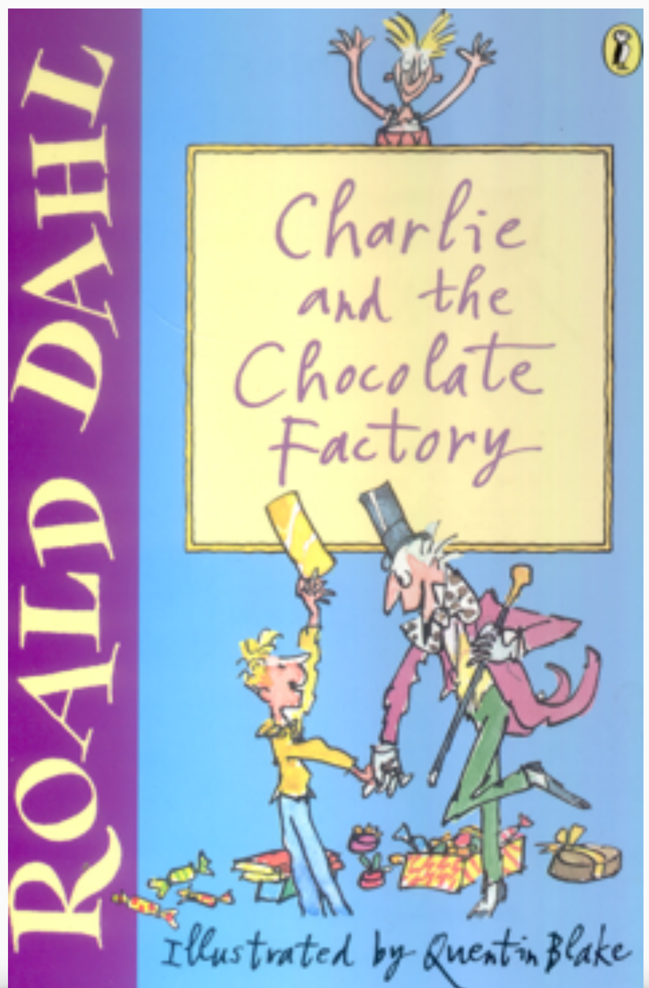
green bits

blue bits

Review: Representation



Review: Representation



0	1	10	11
a	b	c	d

c = 10, h = 100 ...

Onward! Physical Bits

Q: How can we represent 0’s and 1’s in the real world?

Onward! Physical Bits

Q: How can we represent 0's and 1's in the real world?

A: How about people! Raising hand, not raising hand.

Onward! Physical Bits

Q: How can we represent 0's and 1's in the real world?

A: How about people! Raising hand, not raising hand.

A: How about fingers! Raising finger, not raising finger.

Onward! Physical Bits

Q: How can we represent 0's and 1's in the real world?

A: How about people! Raising hand, not raising hand.

A: How about fingers! Raising finger, not raising finger.

Q: Try to represent six with your hands!

Onward! Physical Bits

Q: How can we represent 0’s and 1’s in the real world?

A: How about people! Raising hand, not raising hand.

A: How about fingers! Raising finger, not raising finger.

Q: Try to represent six with your hands!

A: 

Onward! Physical Bits

Q: How can we represent 0’s and 1’s in the real world?

A: How about people! Raising hand, not raising hand.

A: How about fingers! Raising finger, not raising finger.

Q: What can we count up to by using binary with hands?

Onward! Physical Bits

Q: How can we represent 0's and 1's in the real world?

A: How about people! Raising hand, not raising hand.

A: How about fingers! Raising finger, not raising finger.

Q: What can we count up to by using binary with hands?

A: Homework question

Onward! Physical Bits

Q: How can we represent 0's and 1's in the real world?

A: How about people! Raising hand, not raising hand.

A: How about fingers! Raising finger, not raising finger.

Onward! Physical Bits

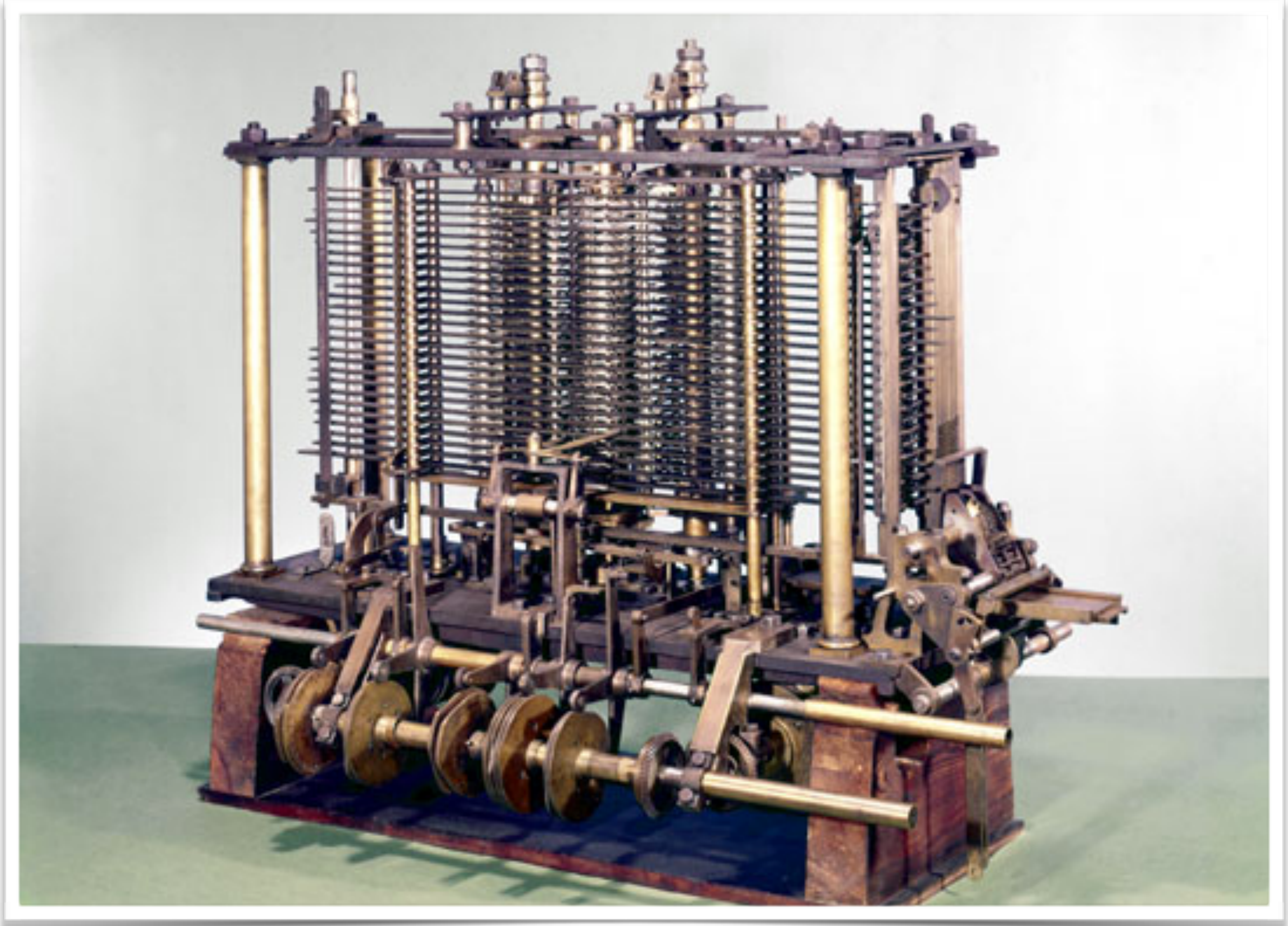
Q: How can we represent 0's and 1's in the real world?

A: How about people! Raising hand, not raising hand.

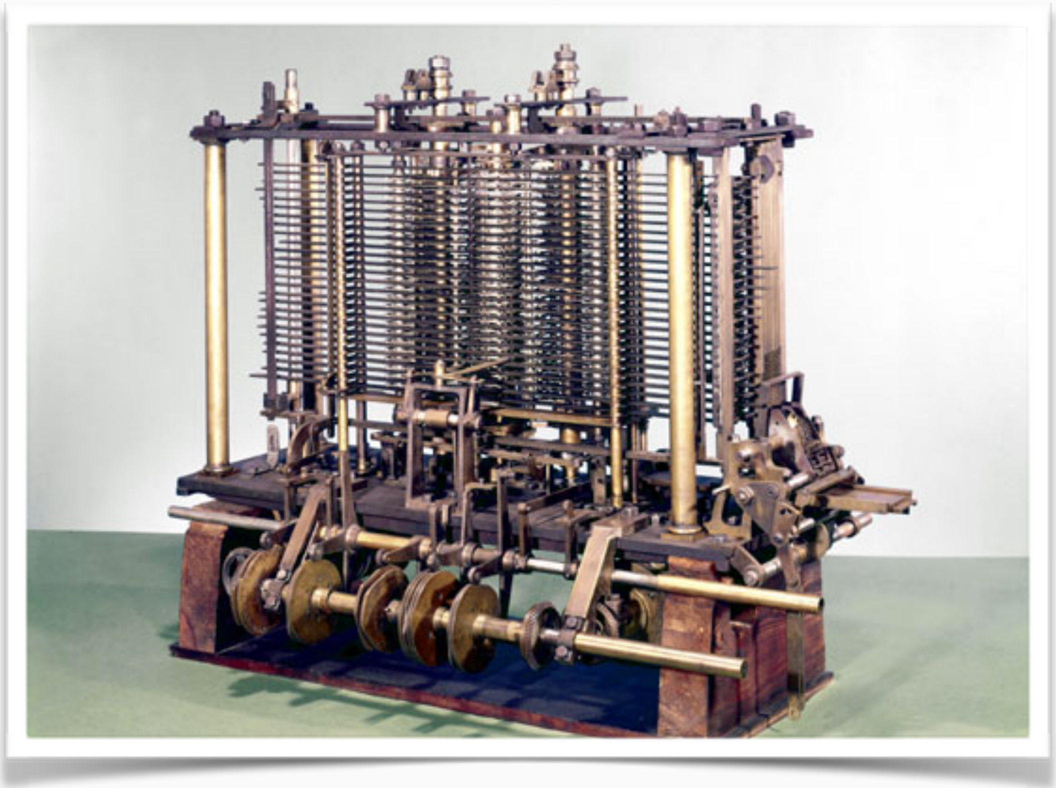
A: How about fingers! Raising finger, not raising finger.

Thought: but we need something smaller…

Some History

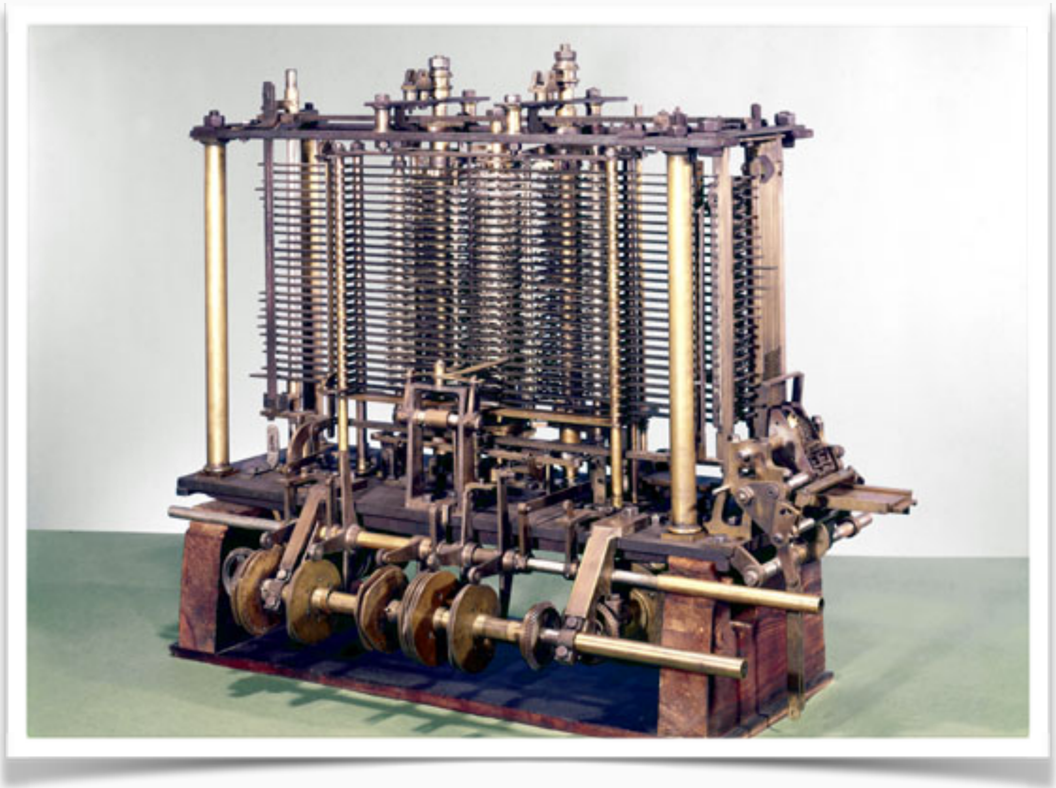


Some History



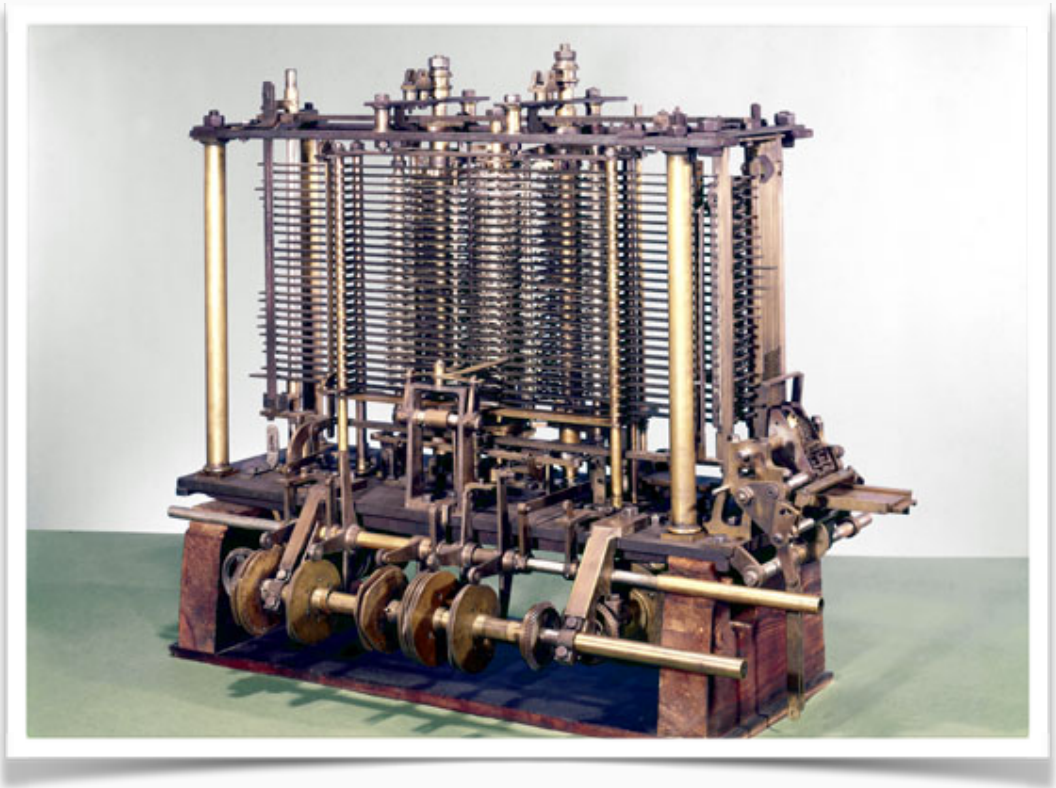
Proposed by Ada Lovelace/ Charles Babbage in 1837

Some History



Proposed by Ada Lovelace/ Charles Babbage in 1837

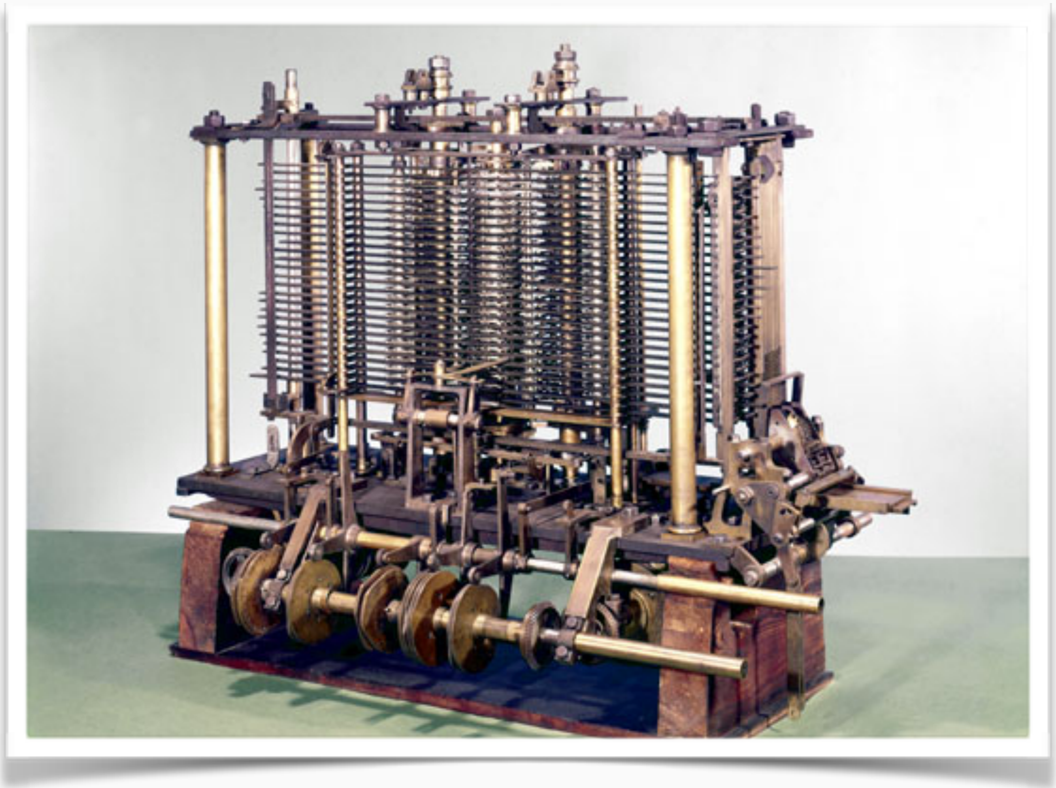
Some History



Proposed by Ada Lovelace/ Charles Babbage in 1837

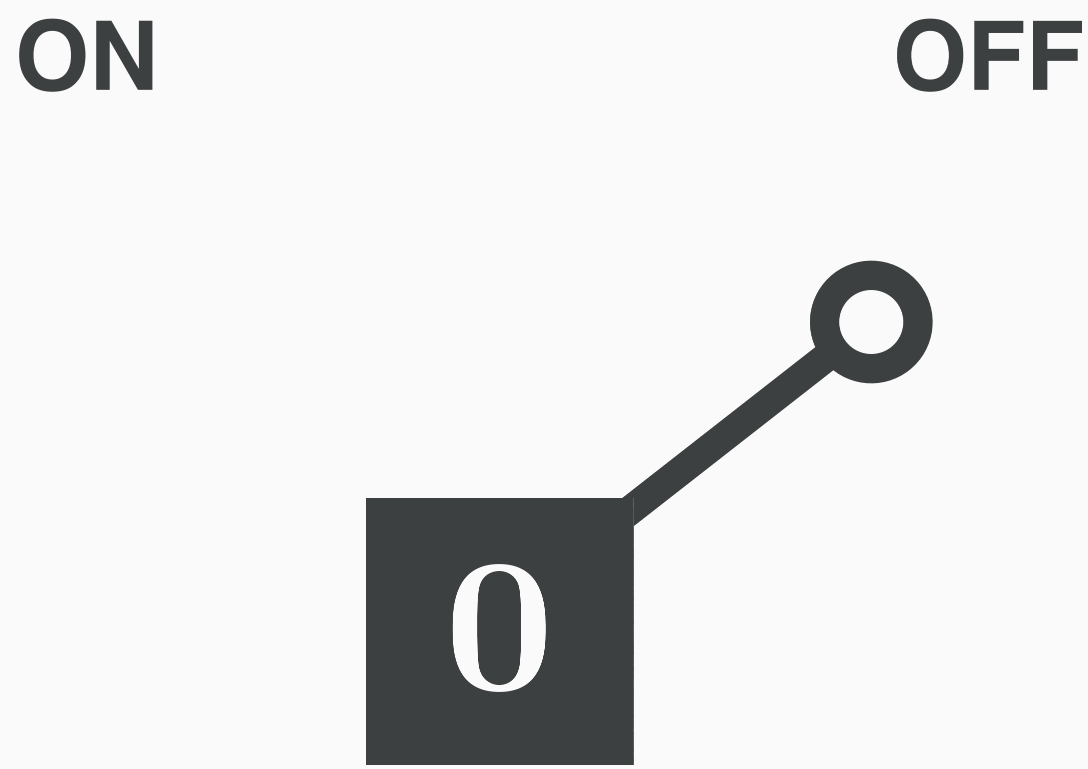
Built in 1910

Some History

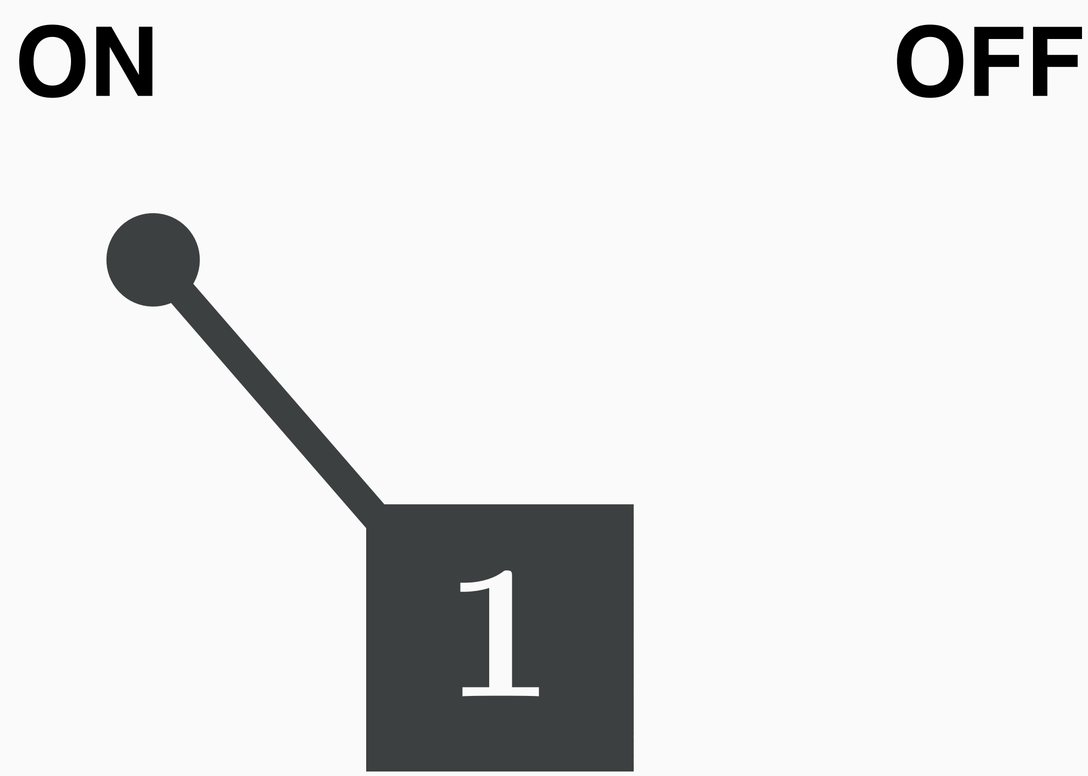


Bits

Bits: Levers



Bits: Levers

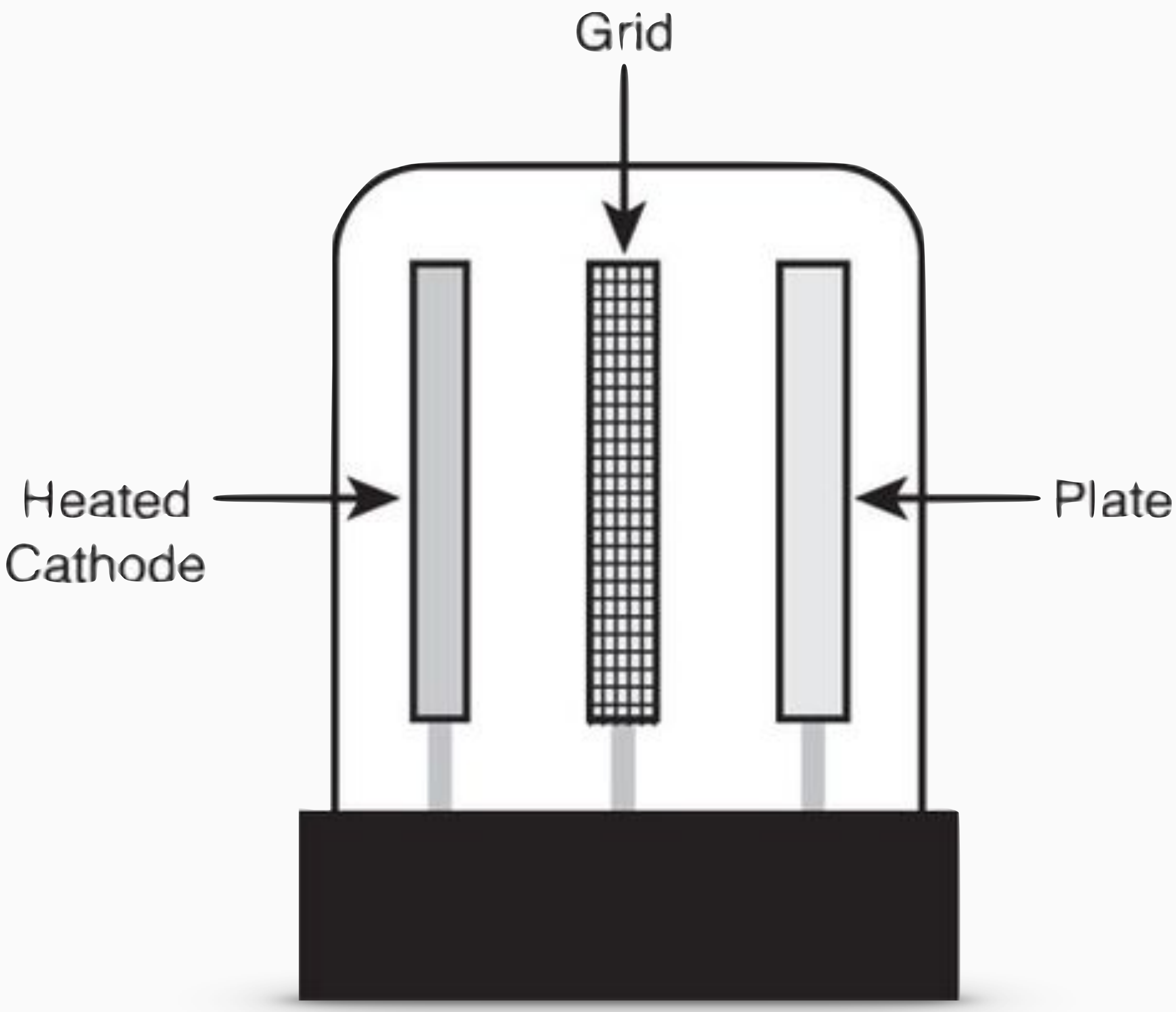


Q: What Is This Number?

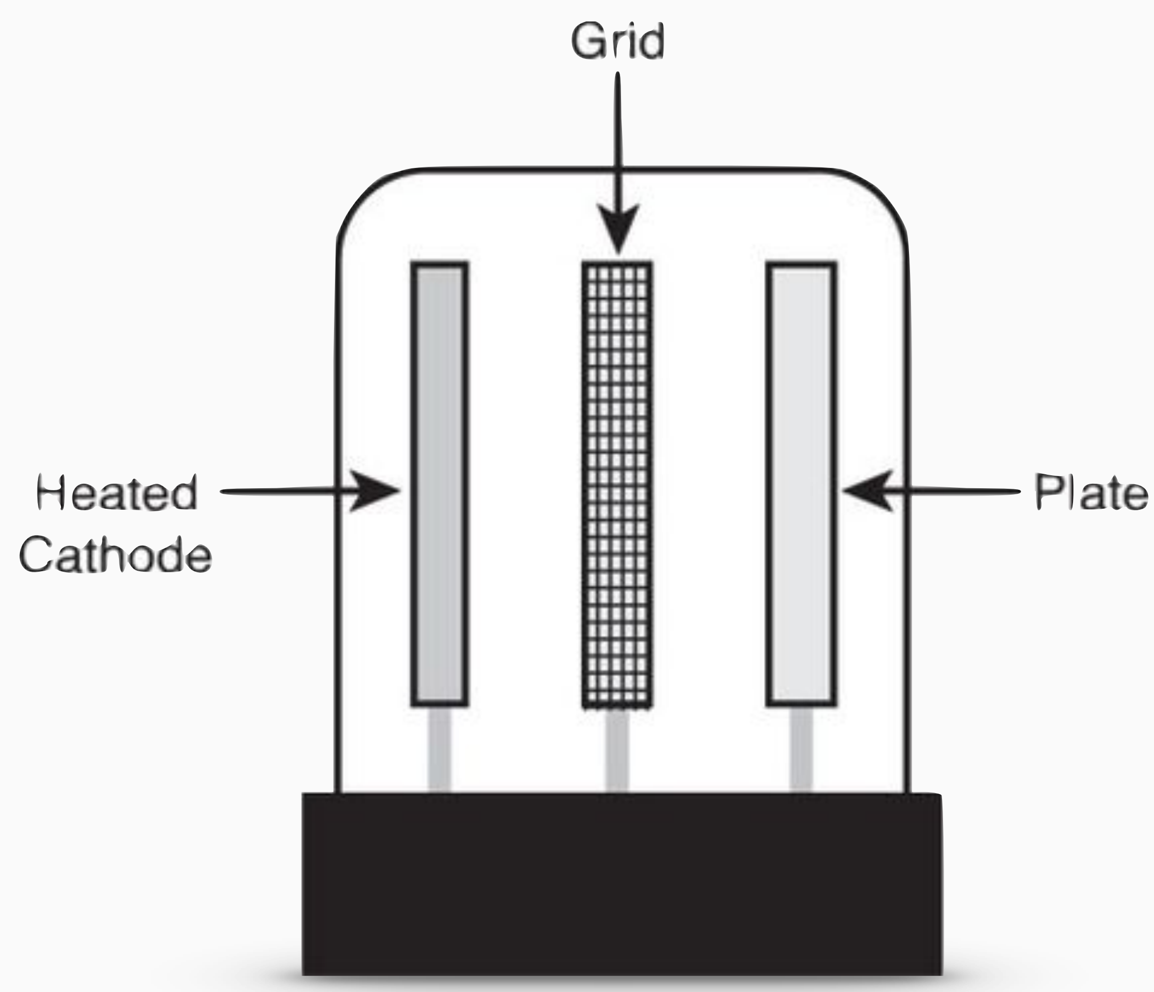
Q: What Is This Number?

Next Up: Vacuum Tubes!

Next Up: Vacuum Tubes!

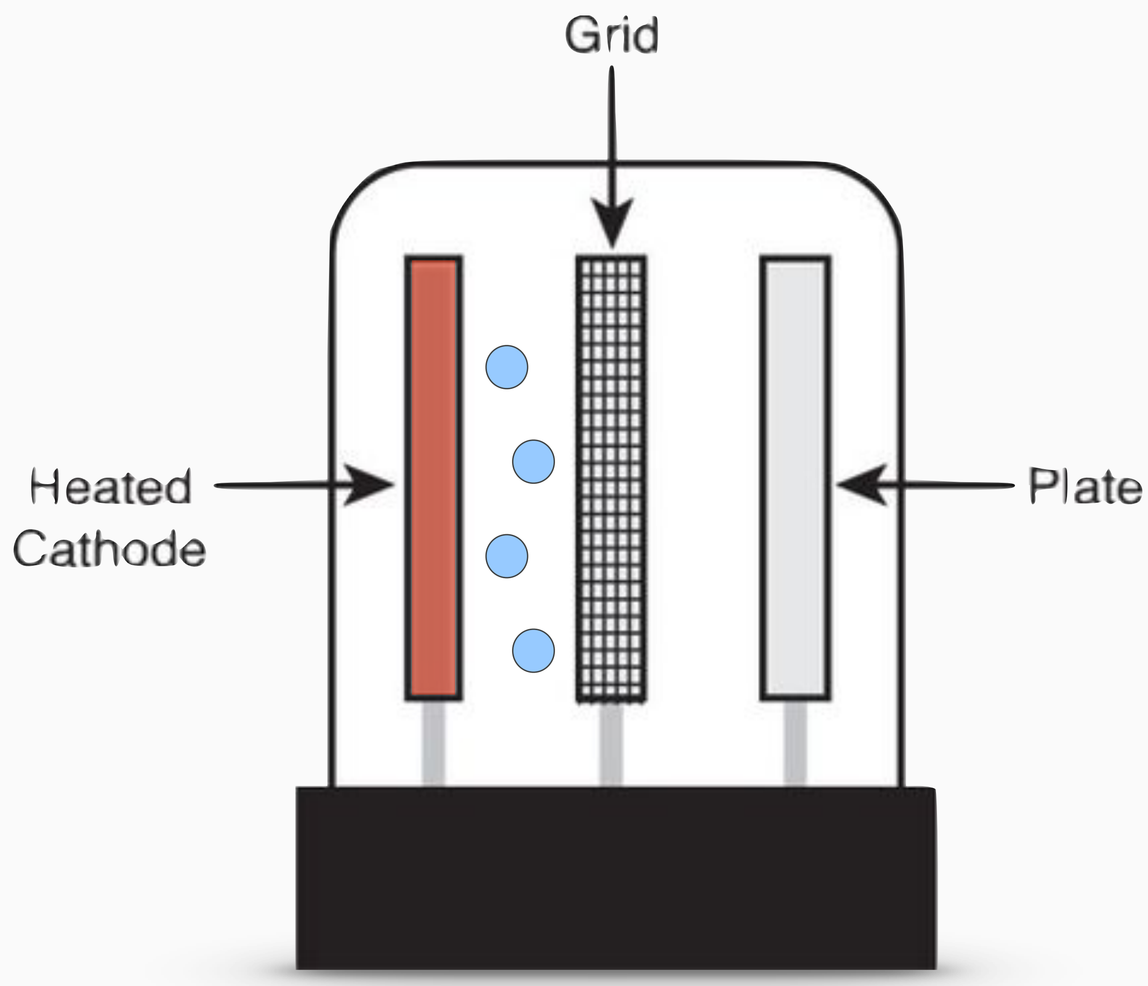


Next Up: Vacuum Tubes!



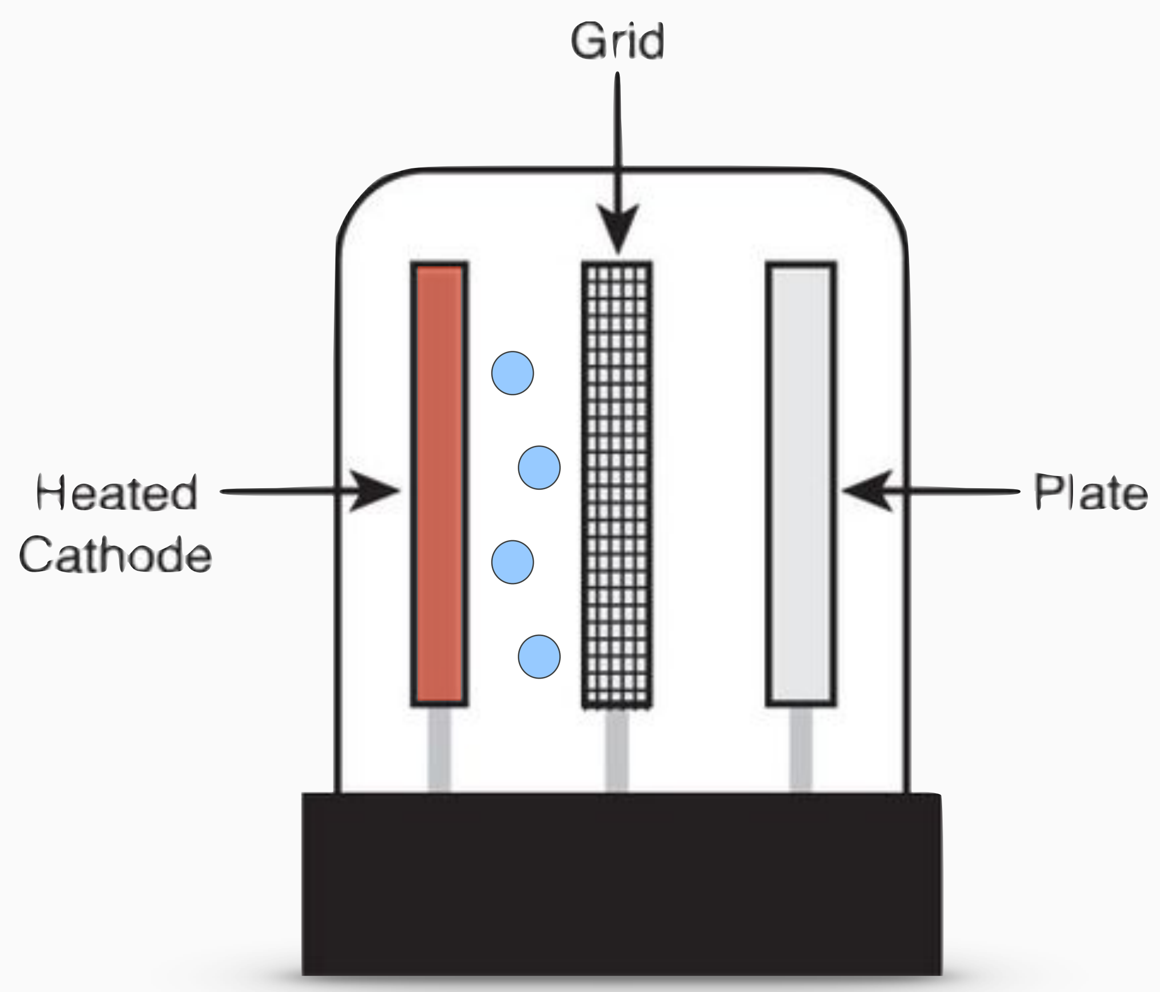
(Called a “Triode”, because three prongs)

Next Up: Vacuum Tubes!



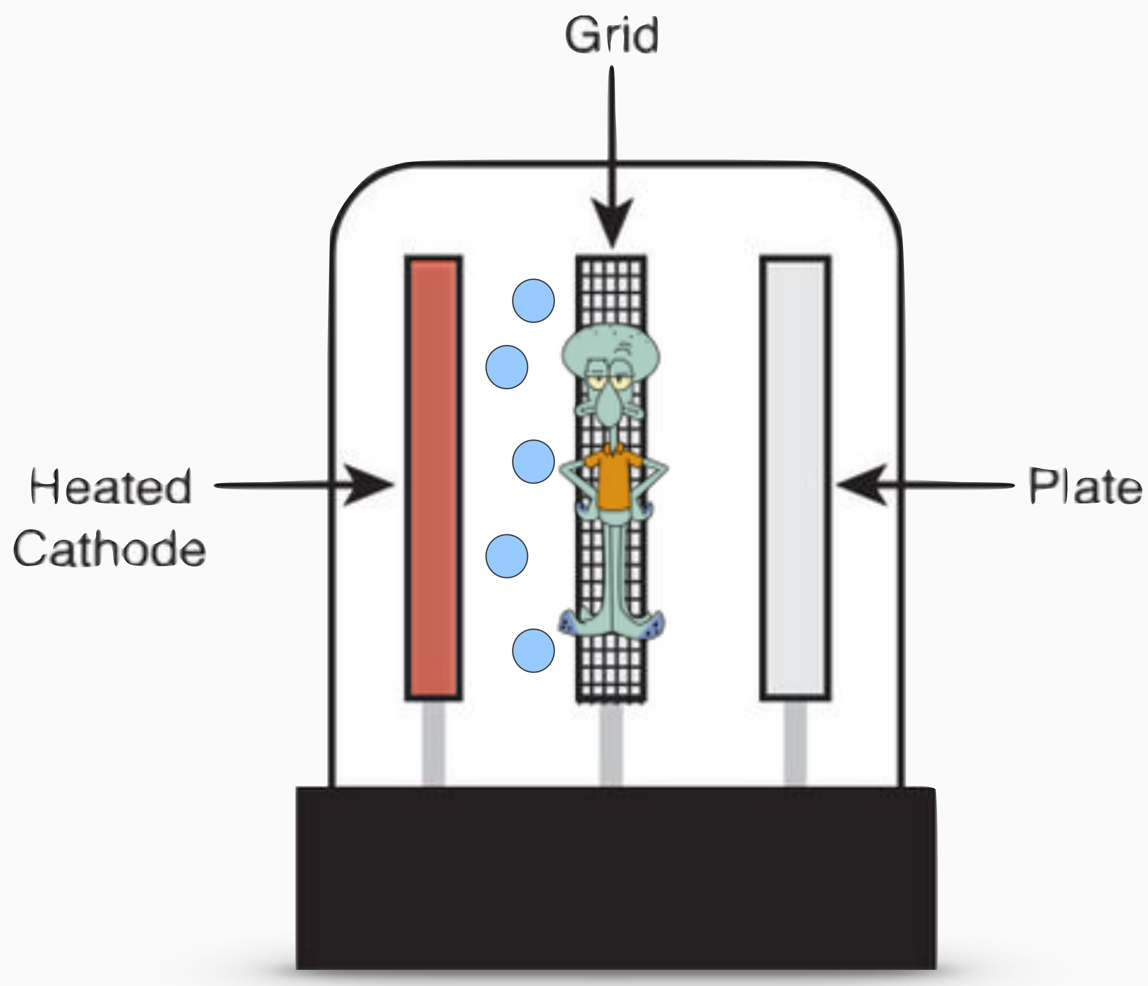
When cathode heated, emits electrons attracted to the plate

Next Up: Vacuum Tubes!



Grid controls the flow of electrons:

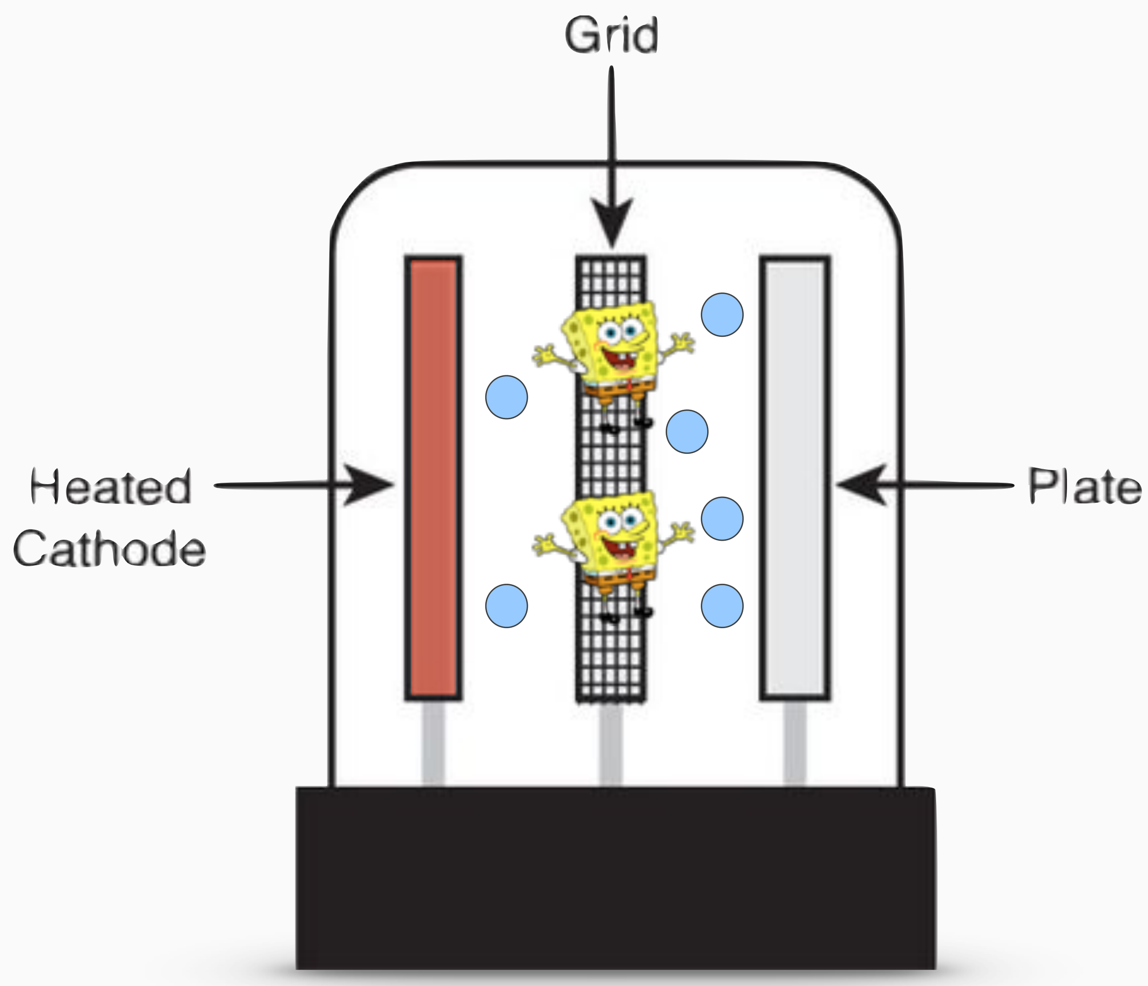
Next Up: Vacuum Tubes!



Grid controls the flow of electrons:

Negative: electrons bounce back to Cathode

Next Up: Vacuum Tubes!

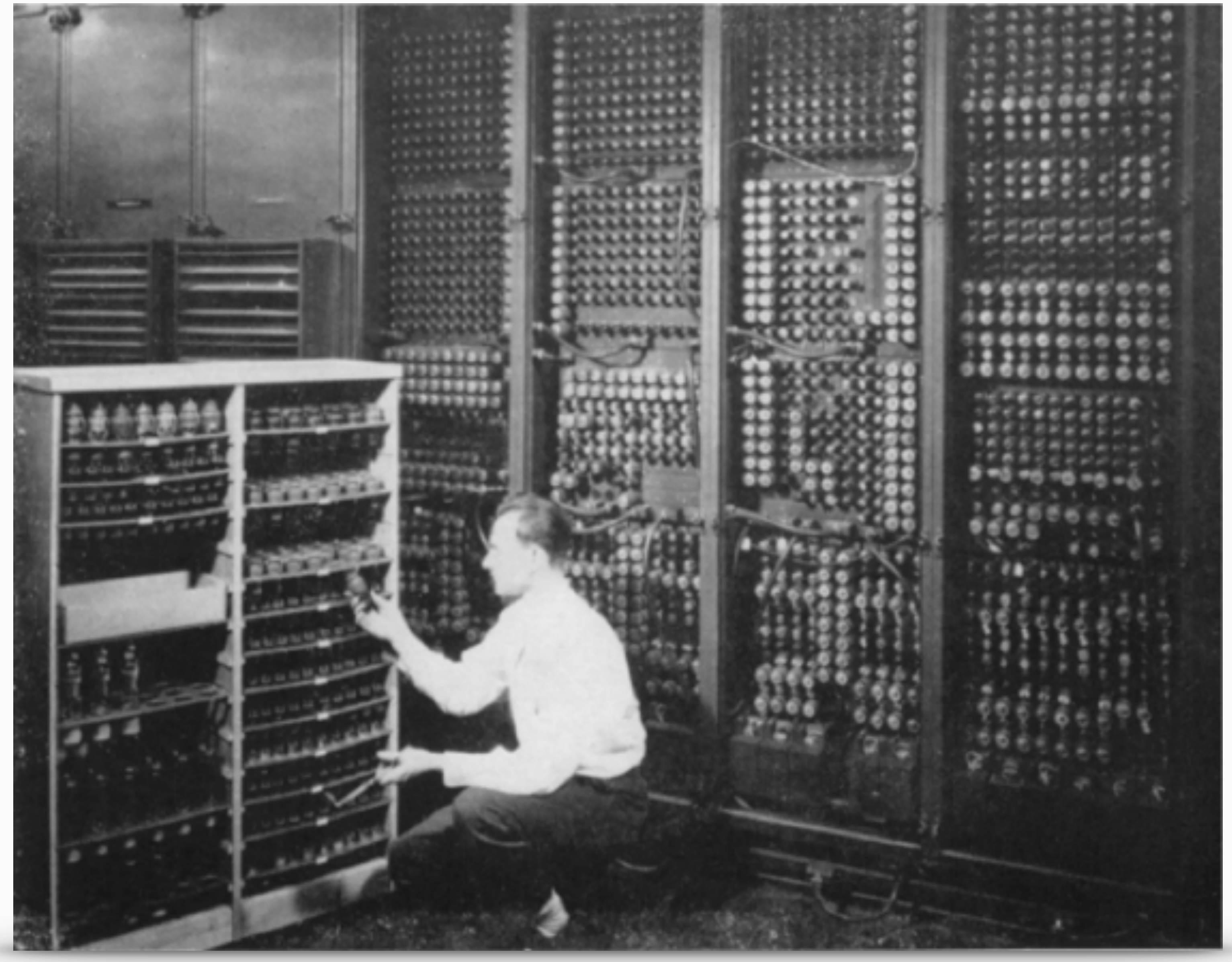


Grid controls the flow of electrons:

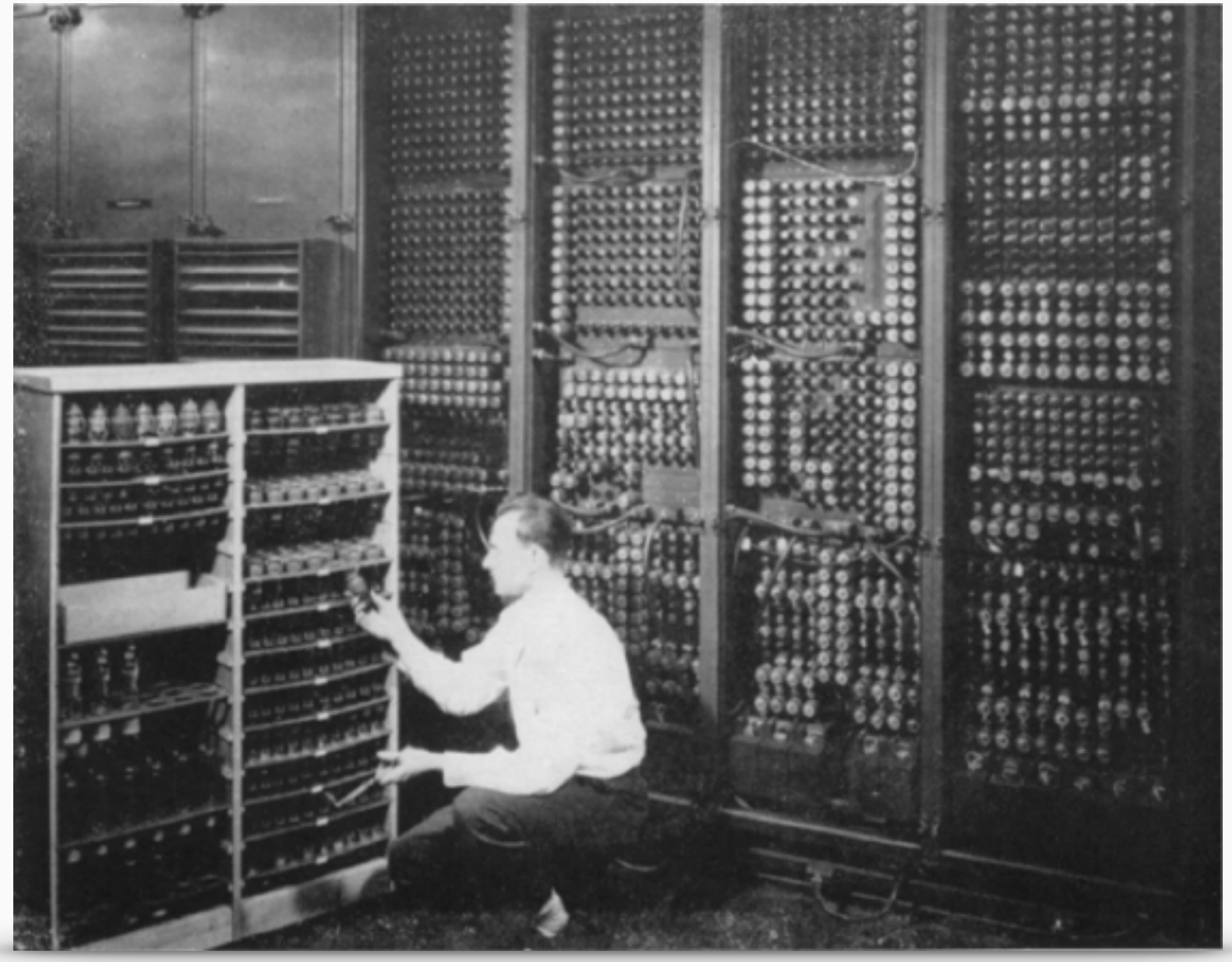
Negative: electrons bounce back to Cathode

Positive: electrons travel through to Cathode

Next Up: Vacuum Tubes!

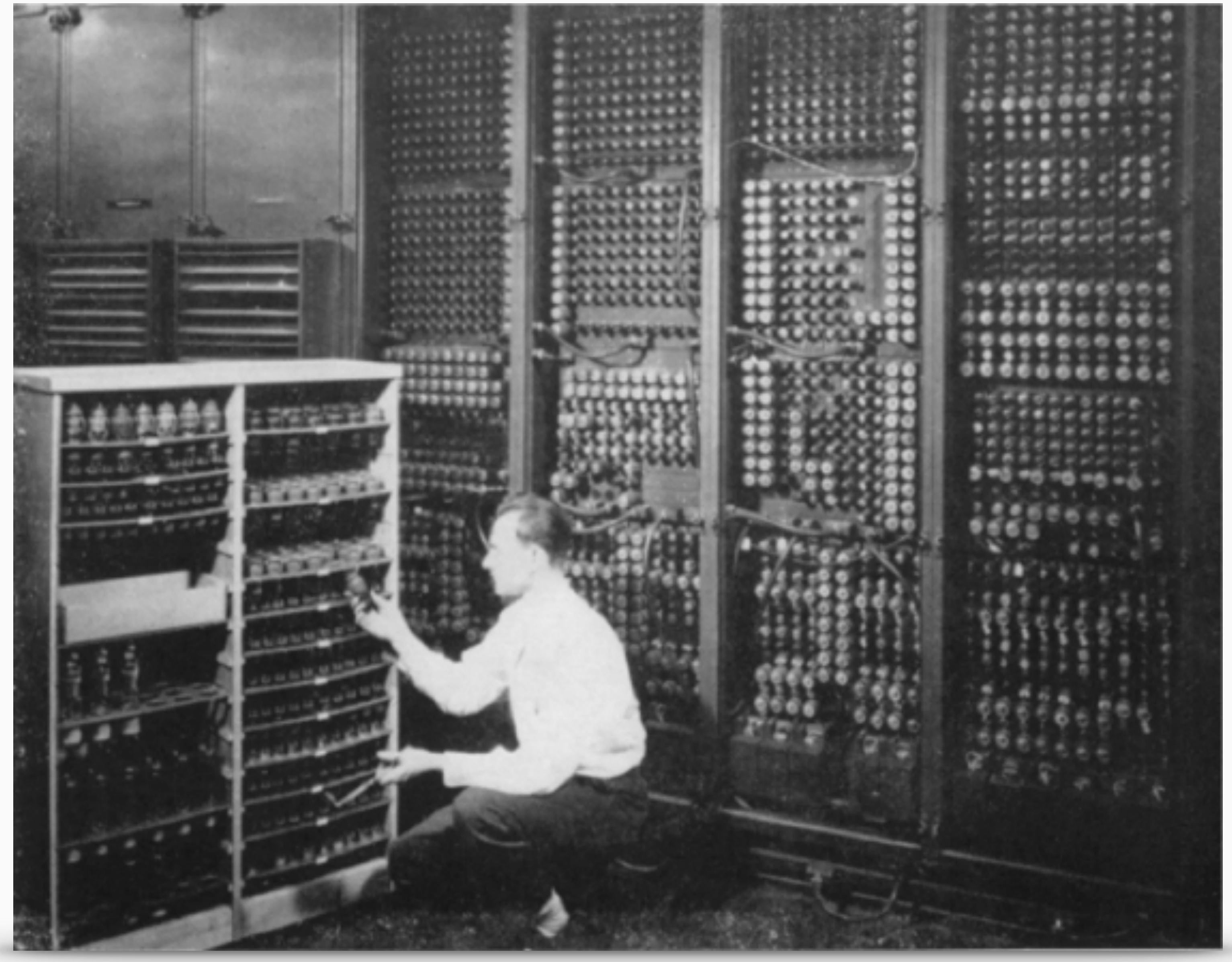


Next Up: Vacuum Tubes!



Q: What if one breaks?

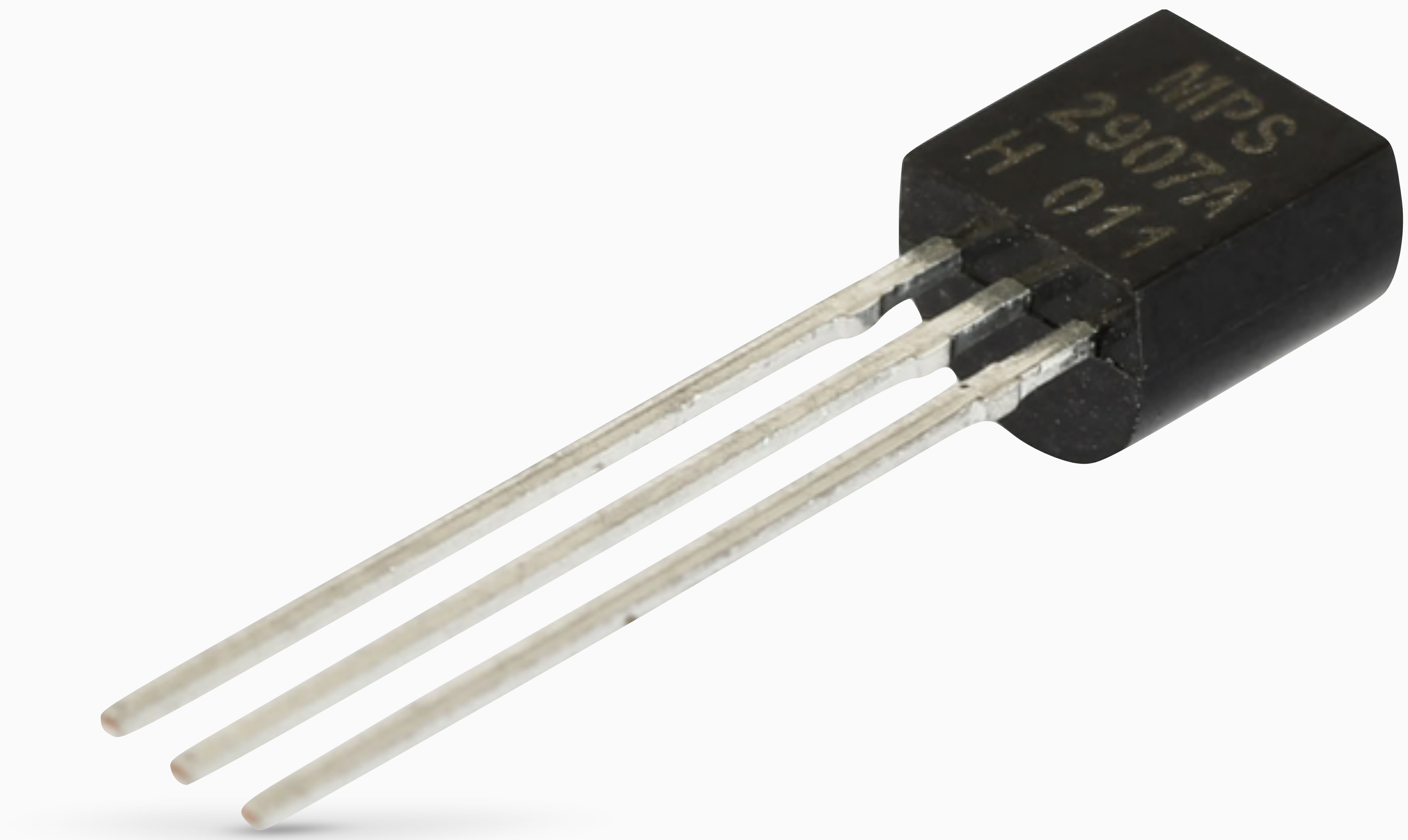
Next Up: Vacuum Tubes!



Q: What if one breaks?

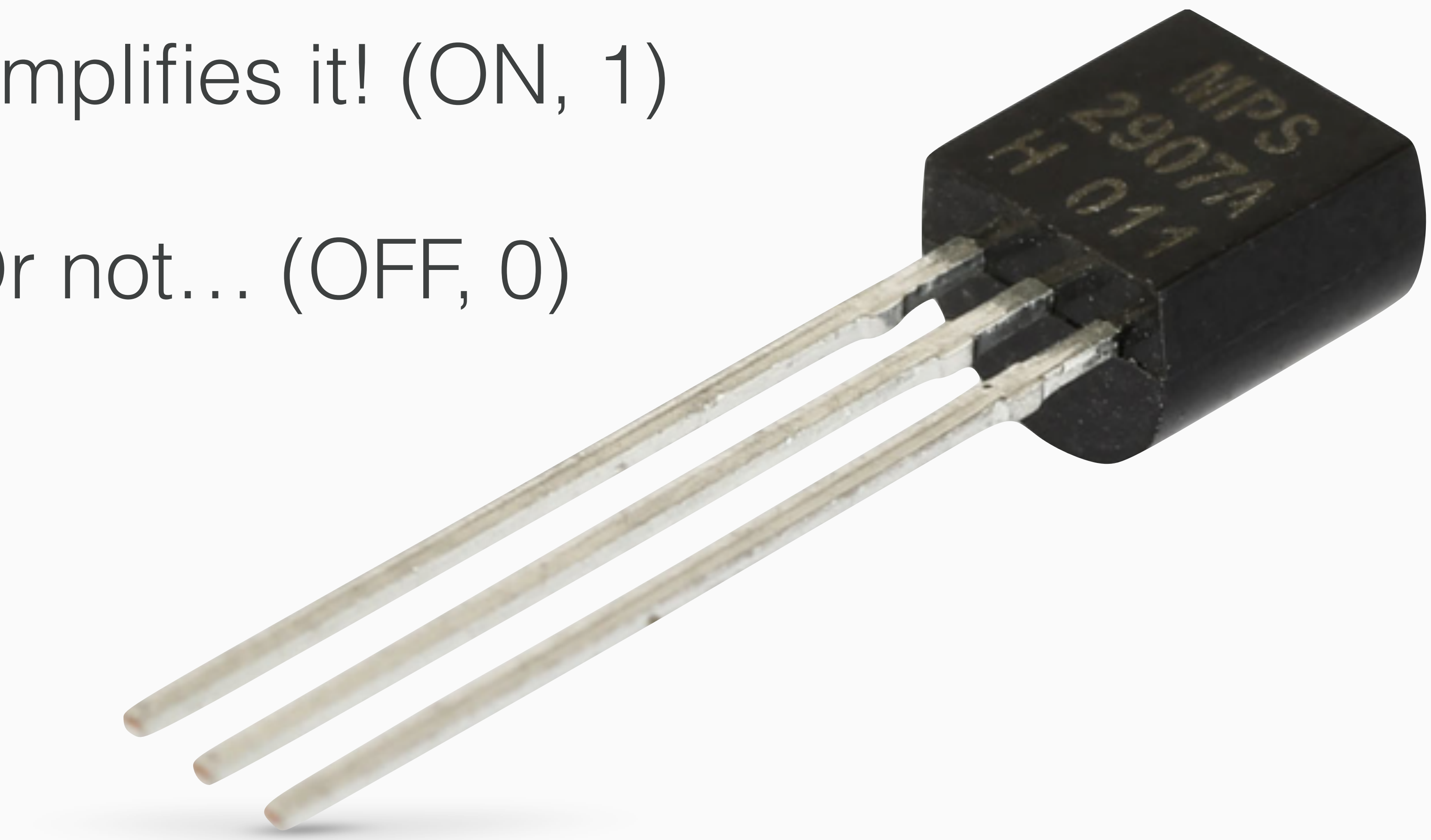
A: Better check all 19,000 tubes…

Now: The Transistor



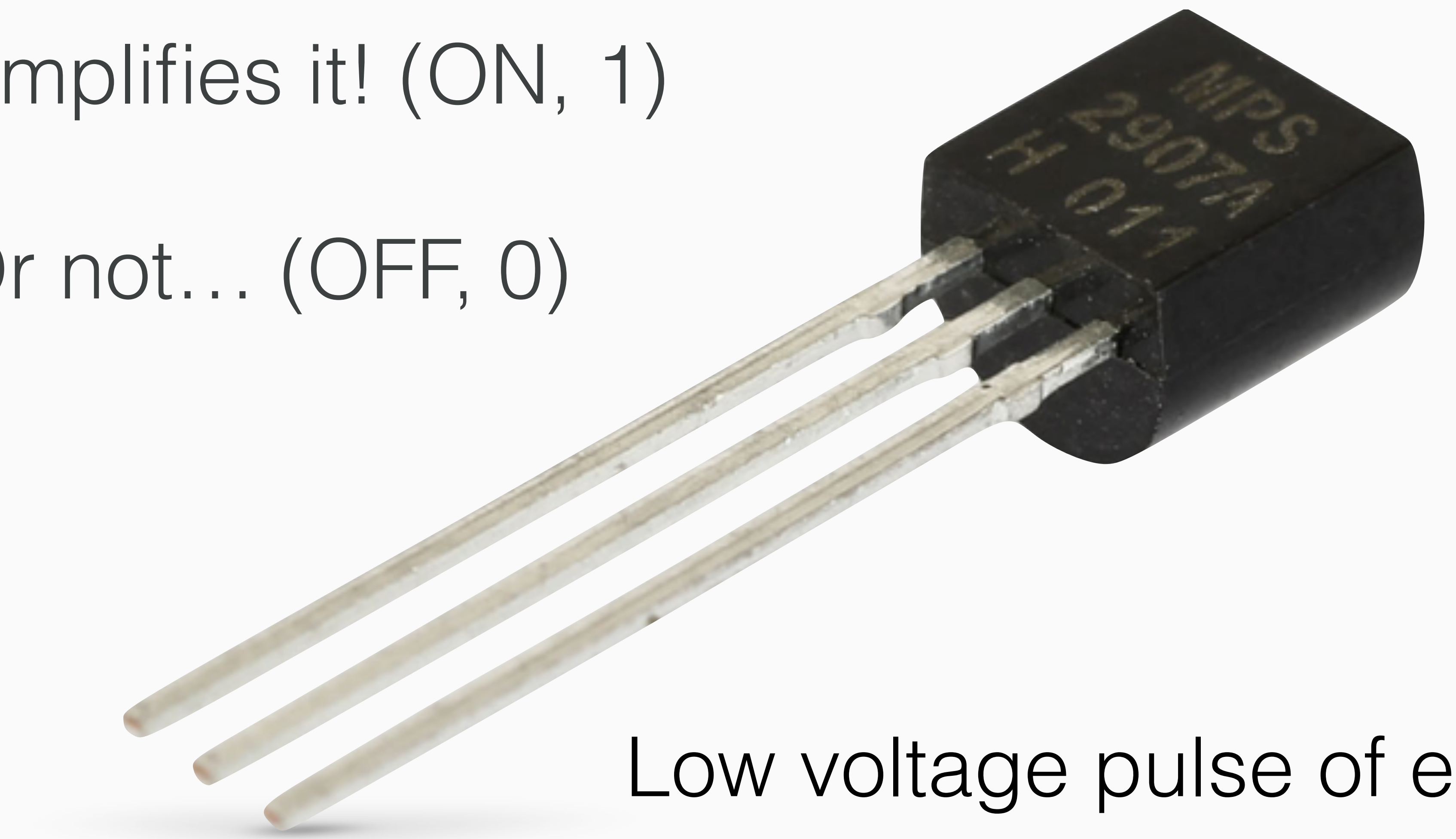
Now: The Transistor

- Takes in electric current:
  - Amplifies it! (ON, 1)
  - Or not… (OFF, 0)



Now: The Transistor

- Takes in electric current:
  - Amplifies it! (ON, 1)
  - Or not… (OFF, 0)

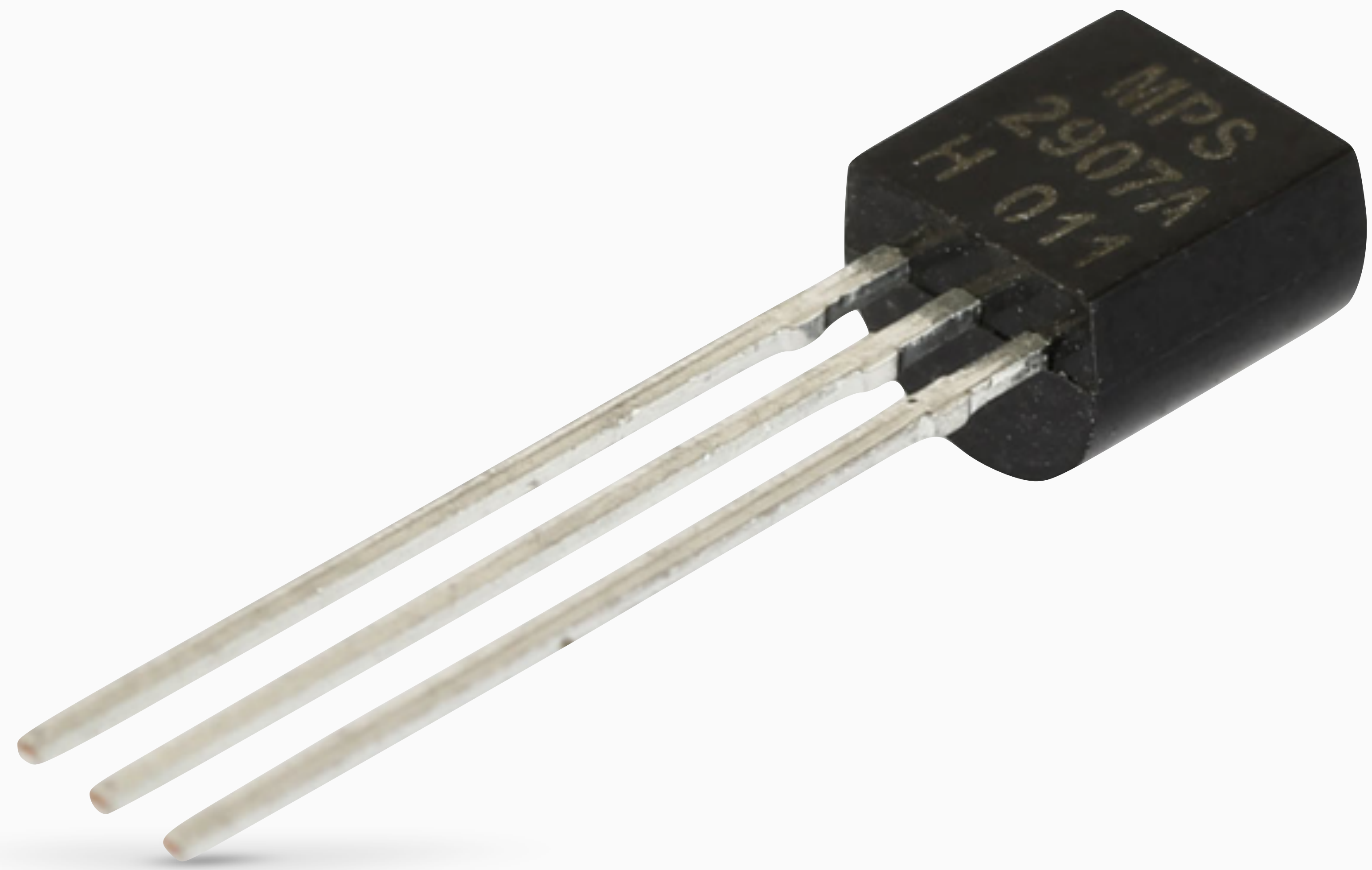


Low voltage pulse of electricity = 0

High voltage pulse of electricity = 1

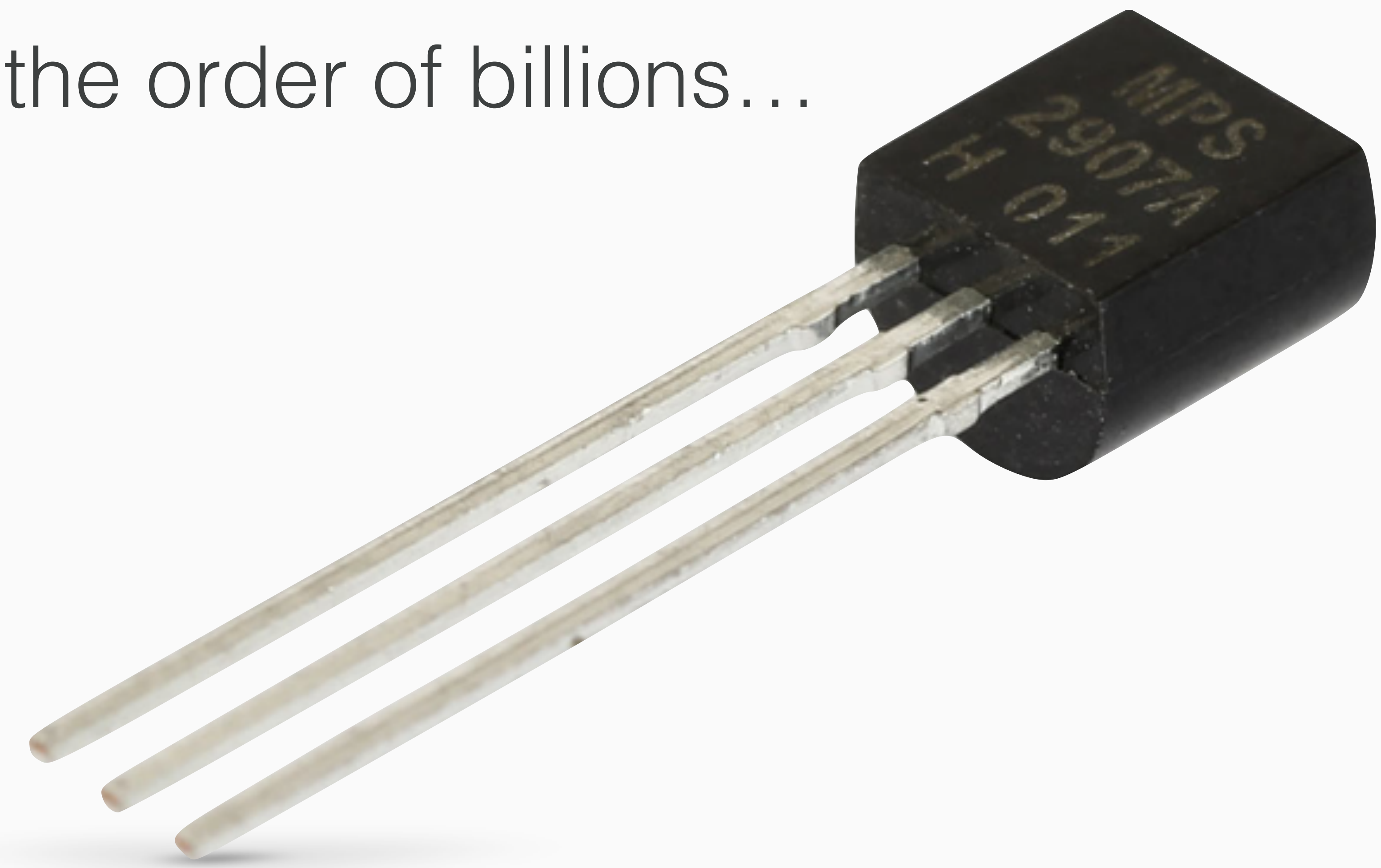
Now: The Transistor

- Q: Roughly how many in your phone/computer?

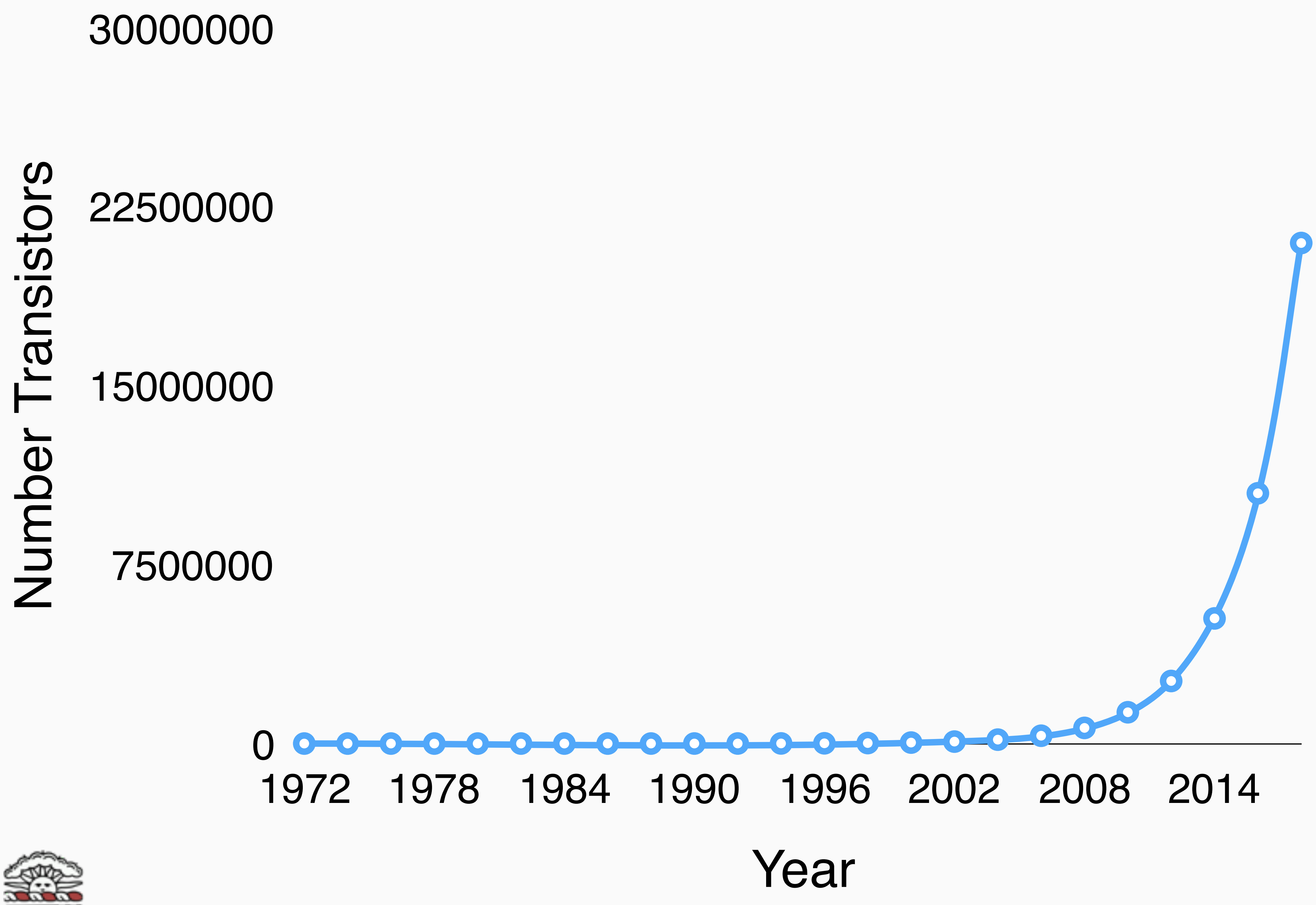


Now: The Transistor

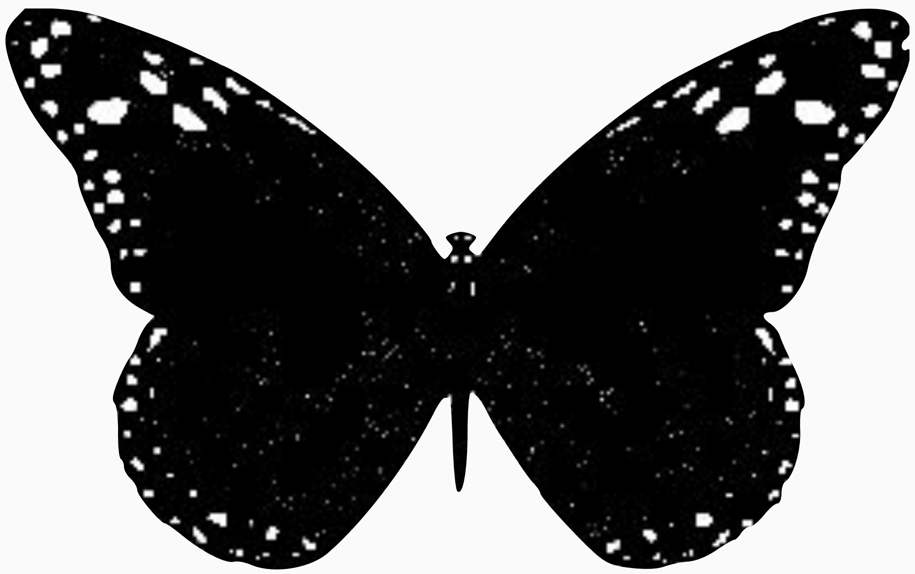
- Q: Roughly how many in your phone/computer?
- A: On the order of billions…

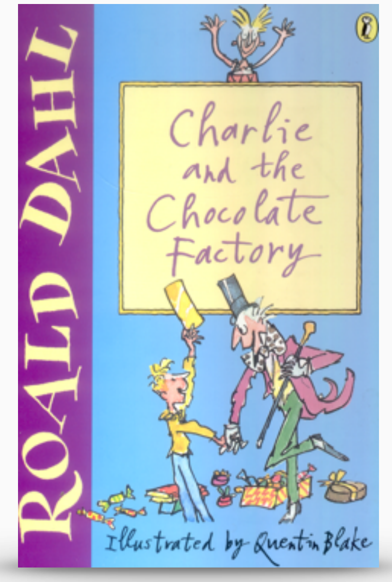


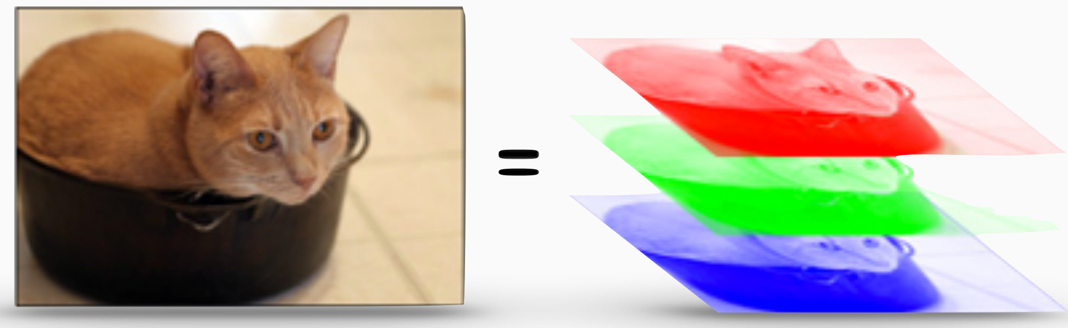
Moore’s Law (Predicted)



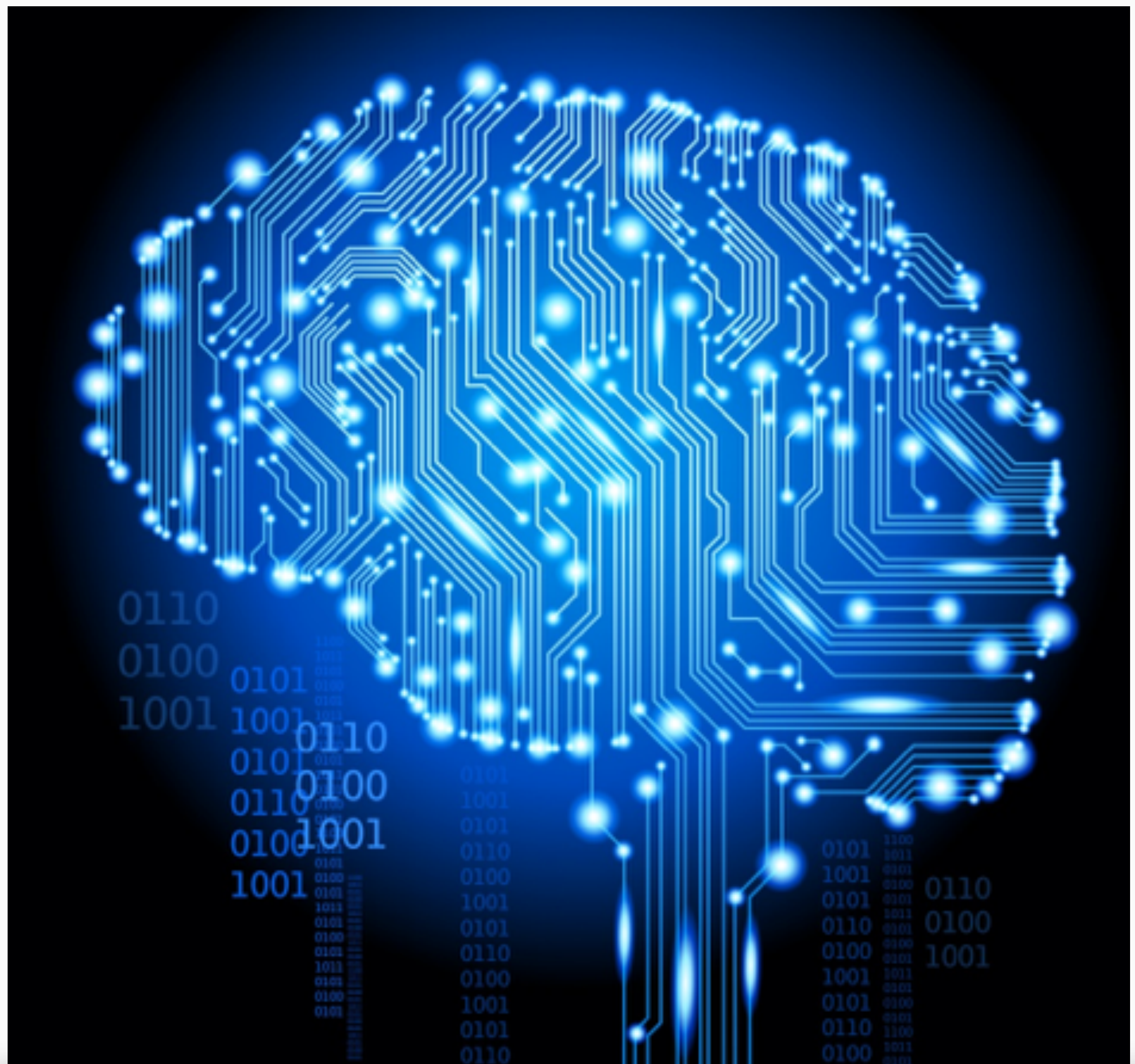
We Know How To Do This…







Still Need: Reasoning



Enter:

Enter:

Idea: What is valid reasoning?

Logic: Some History

Mohism: ~400 B.C.E.

Nyaya: 200 C.E.

Aristotle: ~350 B.C.E.

Logic: Example

- If the snozzberry is a berry, then it is a fruit.
- The snozzberry is a berry.

- Conclusion: The snozzberry is a fruit.

Q: Is this reasoning valid?

Logic: Example

- If the snozzberry is a berry, then it is a fruit.
- The snozzberry is a berry.

- Conclusion: The snozzberry is a fruit.

Q: Is this reasoning valid?

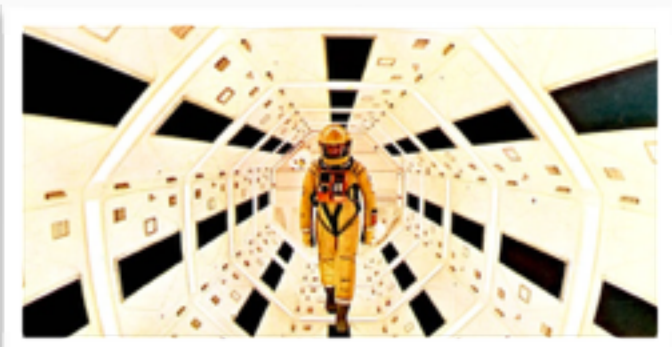
A: Yes! In this unit, we'll learn *what is and is not valid reasoning.*

Logic

Problem: Language is ambiguous, so truth can be finicky!

Language is Ambiguous

Problem: Language is ambiguous, so truth can be finicky!



Q: What *is* art?

Language is Ambiguous

Problem: Language is ambiguous, so truth can be finicky!

Q: What *is* a sport?

Language is Ambiguous

- Example 1: “I never said she stole my money”
  - *Changes meaning depending on the emphasis!*
- Example 2: “Time flies like an arrow”
  - *Is time a noun? (e.g. “Fruit flies like a banana”)*
- Example 3: “I have not slept for ten days”
  - *Has the speaker not slept in the last ten days? Or have they never slept for a period of ten days?*

Logic: A *Formal* Language

- **Variables that stand for sentences:** *P*, *Q*, *R*, *S*

Logic: A *Formal* Language

- **Variables that stand for sentences:** *P, Q, R, S*
- Example:
  - If the snozzberry is a berry, then it is a fruit.
  - The snozzberry is a berry.

  - Therefore, the snozzberry is a fruit.

Logic: A *Formal* Language

- **Variables that stand for sentences:** *P, Q, R, S*
- Example:
  - If **the snozzberry is a berry,** then **it is a fruit.**
  - **The snozzberry is a berry.**

  - Therefore, **the snozzberry is a fruit.**

Logic: A *Formal* Language

- **Variables that stand for sentences:** *P, Q, R, S*
- Example:
  - If **the snozzberry is a berry,** then **it is a fruit.**
  - **The snozzberry is a berry.**
  - Therefore, **the snozzberry is a fruit.**

P

Q

Logic: A *Formal* Language

- **Variables that stand for sentences:** *P, Q, R, S*
- Example:
  - If *P* then *Q*
  - *P*
  - Therefore, *Q*

P = ~~The snozzberry is a berry.~~

Q = ~~The snozzberry is a fruit.~~

Logic: A *Formal* Language

- **Variables that stand for sentences:** *P, Q, R, S*
- Example:
  - If *P* then *Q*
  - *P*
  - Therefore, *Q*

True for all sentences *P*,

All sentences *Q*!

Logic: A *Formal* Language

- **Variables that stand for sentences:** *P, Q, R, S*
- Example:
  - If *P* then *Q*
  - *P*
  - Therefore, *Q*

Premises: assume to be true.

Logic: A *Formal* Language

- **Variables that stand for sentences:** *P, Q, R, S*
- Example:
  - If *P* then *Q*
  - *P*
  - Therefore, *Q*

Premises: assume to be true.

Q: What can we conclude from our premises?

Logic: A *Formal* Language

- **Variables that stand for sentences:** *P, Q, R, S*
- We call sentences that can be True or False "**Boolean**", after Goerge Boole (1800's)

Logic: A *Formal* Language

- **Variables that stand for sentences:** *P, Q, R, S*
- We call sentences that can be True or False "**Boolean**", after Goerge Boole (1800's)
- Henceforth, *P, Q, R, S,* etc., will be called **Boolean Sentences.**

Logic: A *Formal* Language

- **Variables that stand for sentences:** *P, Q, R, S*
- We call sentences that can be True or False "**Boolean**", after Goerge Boole (1800's)
- Henceforth, *P, Q, R, S,* etc., will be called **Boolean Sentences.**

The question is: in what ways can we reason (about Boolean Sentences)?

Logic: Inference Rules

The question is: in what ways can we reason (about Boolean Sentences)?

We saw one:

If *P*, then *Q*
P
Therefore *Q*.

Logic: Inference Rules

The question is: in what ways can we reason (about Boolean Sentences)?

We saw one:

If *P*, then *Q*
P
Therefore *Q*.

Modus Ponens

Logic: Inference Rules

The question is: in what ways can we reason (about Boolean Sentences)?

Modus Ponens

- There are others!
- **Our rules come from thousands of years of intuition about what is perfect reasoning.**

Logic: Inference Rules

The question is: in what ways can we reason (about Boolean Sentences)?

Modus Ponens

- There are others!
- We get three functions:

AND(-,-)

OR(-,-)

NOT(-)

Logic: Boolean Functions

- We get three functions: *AND*, *OR*, *NOT*
  - Each function takes as input a Boolean Sentence (*P*, *Q*, etc.)
  - Outputs a Boolean value (True, False)

Logic: *AND*

AND(P,Q)

Outputs True if **both** *P* and *Q* are True.

Logic: *OR*

AND(P,Q)

Outputs True if **both** *P* and *Q* are True.

OR(P,Q)

Outputs True if **at least one of** *P* or *Q* is True.

Logic: *NOT*

AND(P,Q)

Outputs True if **both** *P* and *Q* are True.

OR(P,Q)

Outputs True if **at least one of** *P* or *Q* is True.

NOT(P)

Outputs True if *P* is False.

Example

- Suppose *P* is True, Q is False: which of the following are True?

  1. *AND(P,Q)*
  2. *OR(P,Q)*
  3. *NOT(P)*
  4. *NOT(Q)*

Example

- Suppose *P* is True, Q is False: which of the following are True?

1. *AND(P,Q)*
2. *OR(P,Q)*
3. *NOT(P)*
4. *NOT(Q)*

AND(P,Q)

Outputs True if **both** *P* and *Q* are True.

OR(P,Q)

Outputs True if **at least one of** *P* or *Q* is True.

NOT(P)

Outputs True if *P* is False.

Example

- Suspose *P* is True, Q is False: which of the following are True?

1. ~~*AND(P,Q)*~~
2. *OR(P,Q)*
3. *NOT(P)*
4. *NOT(Q)*

AND(P,Q)

Outputs True if **both** *P* and *Q* are True.

OR(P,Q)

Outputs True if **at least one of** *P* or *Q* is True.

NOT(P)

Outputs True if *P* is False.

Example

- Suppose *P* is True, Q is False: which of the following are True?

  1. *~~AND(P,Q)~~*
  2. ***OR(P,Q)***
  3. *NOT(P)*
  4. *NOT(Q)*

AND(P,Q)

Outputs True if **both** *P* and *Q* are True.

OR(P,Q)

Outputs True if **at least one of** *P* or *Q* is True.

NOT(P)

Outputs True if *P* is False.

Example

- Suppose *P* is True, Q is False: which of the following are True?

  1. ~~*AND(P,Q)*~~
  2. ***OR(P,Q)***
  3. ~~*NOT(P)*~~
  4. *NOT(Q)*

AND(P,Q)

Outputs True if **both** *P* and *Q* are True.

OR(P,Q)

Outputs True if **at least one of** *P* or *Q* is True.

NOT(P)

Outputs True if *P* is False.

Example

- Suppose *P* is True, Q is False: which of the following are True?

1. *~~AND(P,Q)~~*
2. ***OR(P,Q)***
3. *~~NOT(P)~~*
4. ***NOT(Q)***

AND(P,Q)

Outputs True if **both** *P* and *Q* are True.

OR(P,Q)

Outputs True if **at least one of** *P* or *Q* is True.

NOT(P)

Outputs True if *P* is False.

Truth Tables: *NOT*

P	$NOT($	$P)$
T		
F		

Truth Tables: *NOT*

P	*NOT(*	*P)*
T		T
F		F

Truth Tables: *NOT*

P	*NOT(*	*P)*
T	**F**	T
F	**T**	F

Truth Tables: *NOT*

P	*NOT(*	*P)*
T	**F**	T
F	**T**	F

→ When *P* is True, *NOT(P)* is False

→ When *P* is False, *NOT(P)* is True

Truth Tables: *AND*

P	*Q*	*AND*(	*P*	*Q*)
T	T			
T	F			
F	T			
F	F			

Truth Tables: *AND*

P	*Q*	*AND*(	*P*	*Q*)
T	T		T	T
T	F		T	F
F	T		F	T
F	F		F	F

Truth Tables: *AND*

P	*Q*	*AND*(	*P*	*Q*)
T	T	**T**	T	T
T	F	**F**	T	F
F	T	**F**	F	T
F	F	**F**	F	F

Truth Tables: *AND*

P	Q	$AND($	P	$Q)$
T	T	**T**	T	T
T	F	**F**	T	F
F	T	**F**	F	T
F	F	**F**	F	F

When P and Q are True, $AND(P,Q)$ is True

Otherwise, $AND(P,Q)$ is False

Truth Tables: *OR*

P	Q	$OR($	P	$Q)$
T	T		T	T
T	F		T	F
F	T		F	T
F	F		F	F

Truth Tables: *OR*

P	*Q*	*OR*(	*P*	*Q*)
T	T	**T**	T	T
T	F	**T**	T	F
F	T	**T**	F	T
F	F	**F**	F	F

Inference With *AND*!

AND(P,Q)	*P*
Therefore, *P*	*Q*
Therefore, *Q*	Therefore, *AND(P,Q)*

Logic: Composition

- Boolean Sentences represented with a letter are called **Atomic Sentences** (e.g. *P, Q, R, S,* etc.)
- But since *AND(-,-), OR(-,-),* and *NOT(-)*, also output Boolean Values, they are *also Boolean Sentences.*
- For example:
    - *AND(NOT(P),Q)*
    - *OR(AND(P,Q),NOT(R))*

Truth Tables: *Composite*

P	*Q*	*OR*(	*NOT*(	*Q*),	P)
T	T				
T	F				
F	T				
F	F				

Truth Tables: *Composite*

P	*Q*	*OR*(	*NOT*(	*Q*),	P)
T	T			T	T
T	F			F	T
F	T			T	F
F	F			F	F

Truth Tables: *Composite*

P	*Q*	*OR*(	*NOT*(	*Q*),	P)
T	T		F	T	T
T	F		T	F	T
F	T		F	T	F
F	F		T	F	F

Truth Tables: *Composite*

P	*Q*	*OR*(	*NOT*(	*Q*),	P)
T	T		F	T	T
T	F		T	F	T
F	T		F	T	F
F	F		T	F	F

Truth Tables: *Composite*

P	*Q*	*OR(*	*NOT(*	*Q*),	P)
T	T	**T**	F	T	T
T	F		T	F	T
F	T		F	T	F
F	F		T	F	F

Truth Tables: *Composite*

P	*Q*	*OR*(	*NOT*(	*Q*),	P)
T	T	**T**	F	T	T
T	F	**T**	T	F	T
F	T		F	T	F
F	F		T	F	F

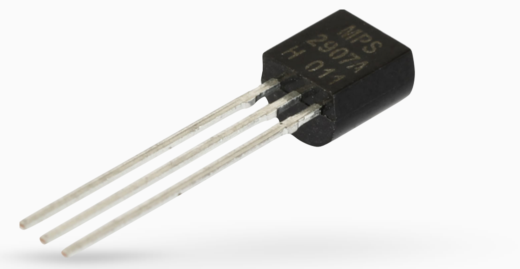
Truth Tables: *Composite*

P	*Q*	*OR(*	*NOT(*	*Q),*	P)
T	T	**T**	F	T	T
T	F	**T**	T	F	T
F	T	**F**	F	T	F
F	F		T	F	F

Truth Tables: *Composite*

P	*Q*	*OR*(	*NOT*(	*Q*),	P)
T	T	**T**	F	T	T
T	F	**T**	T	F	T
F	T	**F**	F	T	F
F	F	**T**	T	F	F

Reflection



- Physical Bits!

- Ambiguity of Regular Languages
- Logical Inference
- Logical Functions: *AND*, *OR*, *NOT*
- Truth Tables

P	*NOT(*	*P)*
T	**F**	T
F	**T**	F

If *P*, then *Q*
P
Therefore *Q*.