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].

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].

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].
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].
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].
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].

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].
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].

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].
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. 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].

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].
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. 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].

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].
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].

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].
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].
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].
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].
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].

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].
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].

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].

Our Number: Zero
Review: Binary Addition

\[
\begin{array}{cccccc}
1 & 0 & 0 & 0 & 1 \\
+ & 1 & 0 & 1 & 0 \\
\hline
1 & 0 & 1 & 1 & 0 \\
\end{array}
\]
Review: Binary Addition

\[
\begin{array}{cccccc}
1 & 0 & 0 & 1 \\
+ & 1 & 0 & 1 & 0 \\
\hline
1 & 0 & 1 & 0 & 1
\end{array}
\]
Review: Binary Addition

\[ \begin{array}{cccccc}
1 & 0 & 0 & 1 \\
+ & 1 & 0 & 1 & 0 \\
\hline
1 & 1 & 1
\end{array} \]
Review: Binary Addition

\[
\begin{array}{cccccc}
1 & 0 & 0 & 1 \\
+ & 1 & 0 & 1 & 0 \\
\hline
0 & 1 & 1 & 1 \\
\end{array}
\]
Review: Binary Addition

\[
\begin{array}{c}
1 \\
\hline
1 & 0 & 0 & 1 \\
+ & 1 & 0 & 1 & 0 \\
\hline
0 & 0 & 1 & 1 \\
\end{array}
\]
Review: Binary Addition

\[
\begin{array}{cccccc}
1 \\
\hline
1 & 0 & 0 & 1 \\
+ & 1 & 0 & 1 & 0 \\
\hline
1 & 0 & 0 & 1 & 1
\end{array}
\]
Review: Binary Addition

Sanity check: nine + ten = nineteen.
Review: Binary Subtraction

A: 
\[
\begin{array}{c}
1 \\
- 0 \\
\hline
1
\end{array}
\]

B: 
\[
\begin{array}{c}
1 \\
- 1 \\
\hline
0
\end{array}
\]

C: 
\[
\begin{array}{c}
0 \\
- 0 \\
\hline
0
\end{array}
\]
Review: Binary Subtraction

\[
\begin{array}{c}
\text{D:} \\
1 \\
1 \\
1 0 \\
- 0 1 \\
\hline \\
1 \\
\end{array}
\]
Review: Binary Subtraction

\[
\begin{array}{cccccc}
1 & 1 & 0 & 1 & 1 & 1 \\
- & 0 & 0 & 1 & 0 & 0 \\
\hline
1 & 1 & 0 & 1 & 0 & 0 \\
\end{array}
\]
Review: Binary Subtraction

\[
\begin{array}{cccccc}
1 & 1 & 0 & 1 & 1 \\
- & 0 & 0 & 1 & 0 \\
\hline \\
& & & & 0 \\
\end{array}
\]
Review: Binary Subtraction

\[
\begin{array}{c}
1 \\
1 \\
1 \\
1 \\
\hline
1 & 1 & 0 & 1 \\
- & 0 & 0 & 1 & 0 \\
\hline
0
\end{array}
\]
Review: Binary Subtraction

\[
\begin{array}{c}
1 \\
1 \\
\hline
1 \ 1 \\
- 0 \ 0 \\
\hline
1 \ 0
\end{array}
\]
Review: Binary Subtraction

\[
\begin{array}{cccccc}
1 & 1 & 1 & 1 & 1 & 1 \\
- & 0 & 0 & 1 & 0 & 0 \\
\hline
1 & 0 & 0 & 1 & 0 & 0
\end{array}
\]
Review: Binary Subtraction

\[
\begin{array}{ccccccc}
1 & 0 & 0 & 0 & 1 \\
- & 0 & 0 & 1 & 0 \\
\hline
1 & 0 & 0 & 1 & 0
\end{array}
\]
Review: Binary Subtraction

\[
\begin{array}{cccc}
1 & 0 & 0 & 1 \\
\hline
0 & 0 & 1 & 0 \\
\end{array}
\]

\[
0 & 1 & 0
\]
Review: Binary Subtraction

\[
\begin{array}{c}
1 \\
1 \\
1 \\
1001 \\
-0010 \\
\hline
1010
\end{array}
\]

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:
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:

```
  1 1 1
* 0 1 0
```

   0 0 0 0

(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:

\[
\begin{array}{c}
\phantom{1} & \phantom{1} & 1 & 1 & 1 \\
\times & 0 & 1 & 0 \\
\hline
1 & 1 & 1 & 1
\end{array}
\]
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:

\[
\begin{array}{cccc}
  & 1 & 1 & 1 \\
* & 0 & 1 & 0 \\
\hline
 1 & 1 & 1 & 1 \\
\end{array}
\]
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:

\[
\begin{array}{c}
1 & 1 & 1 & 1 \\
\times & 0 & 1 & 1 & 0 \\
\hline
1 & 1 & 1 & 0 \\
\end{array}
\]
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:

\[
\begin{array}{c}
  \text{1} \\
  \text{1} \\
  \text{1} \\
\end{array}
\]

* 

\[
\begin{array}{c}
  \text{0} \\
  \text{1} \\
  \text{0} \\
\end{array}
\]

---

Sanity Check: seven * two = fourteen
Review: Representation
Review: Representation

- Blue bits
- Green bits
- Red bits
Review: Representation
Review: Representation

c = 10, h = 100 ...
Q: How can we represent 0’s and 1’s in the real world?
Q: How can we represent 0’s and 1’s in the real world?

A: How about people! Raising hand, not raising hand.
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: 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!
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: [Hand gesture diagram]
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?
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
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: Levers

ON

OFF

0
Bits: Levers
Q: What Is This Number?
Q: What Is This Number?

1
1
0
1
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)

Number Transistors

Year


0 7500000 15000000 22500000 30000000
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: $P, Q, R, S$
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.*
Logic: A Formal Language

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

- Example:
  - If $P$, then $Q$
  - $P$
  - Therefore, $Q$

\[
\text{P} \rightarrow \text{Q} = \text{The snozzberry is a berry.} \\
\text{P} \land \text{Q} = \text{The snozzberry is a fruit.}
\]
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 George 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 George 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 George 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)?
The question is: in what ways can we reason (about Boolean Sentences)?

We saw one:

If $P$, then $Q$
$P$
Therefore $Q$. 
The question is: in what ways can we reason (about Boolean Sentences)?

We saw one:

If $P$, then $Q$
$P$
Therefore $Q$.          Modus Ponens
The question is: in what ways can we reason (about Boolean Sentences)?

- There are others!

- Our rules come from thousands of years of intuition about what is perfect reasoning.

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

- 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: \textit{AND}

\texttt{AND(P,Q)}

Outputs True if \textbf{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: \( \text{NOT} \)

\( \text{AND}(P,Q) \)

Outputs True if \textbf{both} \( P \) and \( Q \) are True.

\( \text{OR}(P,Q) \)

Outputs True if \textbf{at least one of} \( P \) or \( Q \) is True.

\( \text{NOT}(P) \)

Outputs True if \( P \) is False.
Example

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

1. $\text{AND}(P,Q)$
2. $\text{OR}(P,Q)$
3. $\text{NOT}(P)$
4. $\text{NOT}(Q)$
Example

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

1. $\text{AND}(P,Q)$
   - Outputs True if both $P$ and $Q$ are True.

2. $\text{OR}(P,Q)$
   - Outputs True if at least one of $P$ or $Q$ is True.

3. $\text{NOT}(P)$
   - Outputs True if $P$ is False.

4. $\text{NOT}(Q)$
Example

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

1. $\text{AND}(P,Q)$
2. $\text{OR}(P,Q)$
3. $\text{NOT}(P)$
4. $\text{NOT}(Q)$

\text{AND}(P,Q)

Outputs True if \textbf{both} $P$ and $Q$ are True.

\text{OR}(P,Q)

Outputs True if \textbf{at least one of} $P$ or $Q$ is True.

\text{NOT}(P)

Outputs True if $P$ is False.
Example

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

1. $\text{AND}(P,Q)$
   - Outputs True if both $P$ and $Q$ are True.

2. $\text{OR}(P,Q)$
   - Outputs True if at least one of $P$ or $Q$ is True.

3. $\text{NOT}(P)$
   - Outputs True if $P$ is False.

4. $\text{NOT}(Q)$
Example

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

  1. $\text{AND}(P,Q)$
  2. $\text{OR}(P,Q)$
  3. $\text{NOT}(P)$
  4. $\text{NOT}(Q)$

$\text{AND}(P,Q)$

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

$\text{OR}(P,Q)$

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

$\text{NOT}(P)$

Outputs True if $P$ is False.
Example

- Suppose $P$ is True, $Q$ is False: which of the following are True?
  
  1. $\text{AND}(P,Q)$
  
  2. $\text{OR}(P,Q)$
  
  3. $\text{NOT}(P)$
  
  4. $\text{NOT}(Q)$

$\text{AND}(P,Q)$

Outputs True if $\text{both } P \text{ and } Q \text{ are True.}$

$\text{OR}(P,Q)$

Outputs True if $\text{at least one of } P \text{ or } Q \text{ is True.}$

$\text{NOT}(P)$

Outputs True if $P$ is False.
Truth Tables: \( NOT \)
Truth Tables: \textit{NOT}

<table>
<thead>
<tr>
<th>( P )</th>
<th>( \textit{NOT}(P) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T )</td>
<td>( T )</td>
</tr>
<tr>
<td>( F )</td>
<td>( F )</td>
</tr>
</tbody>
</table>
Truth Tables: \textit{NOT}

<table>
<thead>
<tr>
<th></th>
<th>\textit{NOT}(P)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>
Truth Tables: \( NOT \)

When \( P \) is True, \( NOT(P) \) is False

When \( P \) is False, \( NOT(P) \) is True
Truth Tables: \textit{AND}

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</th>
<th>$\text{AND}(P, Q)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
Truth Tables: *AND*

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</th>
<th>AND($P$, $Q$)</th>
<th>$P$</th>
<th>$Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
Truth Tables: **AND**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th><strong>AND</strong></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>P</strong></td>
<td></td>
<td><strong>T</strong></td>
<td><strong>P</strong></td>
<td><strong>T</strong></td>
</tr>
<tr>
<td><strong>Q</strong></td>
<td><strong>T</strong></td>
<td></td>
<td><strong>T</strong></td>
<td><strong>T</strong></td>
</tr>
<tr>
<td></td>
<td><strong>T</strong></td>
<td><strong>F</strong></td>
<td></td>
<td><strong>T</strong></td>
</tr>
<tr>
<td><strong>F</strong></td>
<td><strong>F</strong></td>
<td><strong>F</strong></td>
<td><strong>F</strong></td>
<td><strong>T</strong></td>
</tr>
<tr>
<td><strong>F</strong></td>
<td><strong>F</strong></td>
<td><strong>F</strong></td>
<td><strong>F</strong></td>
<td><strong>F</strong></td>
</tr>
</tbody>
</table>
Truth Tables: *AND*

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</th>
<th>$\text{AND}(P,Q)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

When $P$ and $Q$ are True, $\text{AND}(P,Q)$ is True.

Otherwise, $\text{AND}(P,Q)$ is False.
Truth Tables: OR

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</th>
<th>$OR(P \lor Q)$</th>
<th>$P$</th>
<th>$Q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
Truth Tables: \( OR \)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>( OR )</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

\( P \) | \( Q \) | \( OR(\) \) | \( P \) | \( Q \) |
Inference With $\text{AND}$!

\[
\begin{align*}
\text{AND}(P,Q) & \quad P \\
\quad & \quad Q \\
\hline
\text{Therefore, } P & \quad \text{Therefore, } Q \\
\text{Therefore, } Q & \quad \text{Therefore, } \text{AND}(P,Q)
\end{align*}
\]
Logic: Composition

- Boolean Sentences represented with a letter are called **Atomic Sentences** (e.g. \( P, Q, R, S, \) etc.)

- But since \( \text{AND}(-,-), \text{OR}(-,-), \) and \( \text{NOT}(-) \), also output Boolean Values, they are also **Boolean Sentences**.

- For example:
  - \( \text{AND}(\text{NOT}(P),Q) \)
  - \( \text{OR}(\text{AND}(P,Q),\text{NOT}(R)) \)
Truth Tables: *Composite*

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</th>
<th>$OR$</th>
<th>$NOT(Q)$</th>
<th>$Q,$</th>
<th>$P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Truth Tables: *Composite*

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>OR(</th>
<th>NOT(</th>
<th>Q),</th>
<th>P)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
Truth Tables: *Composite*

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</th>
<th>$OR(P, Q)$</th>
<th>$NOT(Q)$</th>
<th>$Q$, $P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T, T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T, T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T, F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F, F</td>
</tr>
</tbody>
</table>
Truth Tables: \textit{Composite}

<table>
<thead>
<tr>
<th></th>
<th>\textit{OR}</th>
<th>\textit{NOT}</th>
<th>\textit{Q),}</th>
<th>\textit{P}</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
## Truth Tables: Composite

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</th>
<th>$OR(Q)$</th>
<th>$NOT(Q)$</th>
<th>$(Q,P)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>$T$</td>
<td>$T$</td>
<td>$F$</td>
<td>$T$</td>
</tr>
<tr>
<td>$T$</td>
<td>$F$</td>
<td>$T$</td>
<td>$F$</td>
<td>$T$</td>
</tr>
<tr>
<td>$F$</td>
<td>$T$</td>
<td>$F$</td>
<td>$T$</td>
<td>$F$</td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
<td>$T$</td>
<td>$F$</td>
<td>$F$</td>
</tr>
</tbody>
</table>
Truth Tables: *Composite*

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</th>
<th>$OR(Q)$</th>
<th>$NOT(Q)$</th>
<th>$NOT(Q), P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
Truth Tables: *Composite*

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>OR(</th>
<th>NOT(</th>
<th>Q),</th>
<th>P)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
Truth Tables: *Composite*

<table>
<thead>
<tr>
<th>$P$</th>
<th>$Q$</th>
<th>$OR($</th>
<th>$NOT($</th>
<th>$Q)$,</th>
<th>$P)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
Reflection

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

<table>
<thead>
<tr>
<th>$P$</th>
<th>$\text{NOT}(P)$</th>
<th>$P$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

If $P$, then $Q$
$P$
Therefore $Q$. 