Lecture 19: How does a computer... act so logically?

Motivating Example

Ben only rides his bike to class if he overslept, but even then if it is raining he’ll walk and show up late (he hates to bike in the rain). But if there’s an exam that day he’ll bike if he overslept, even in the rain.

It’s raining, Ben overslept, and there’s an exam

Will Ben bike today????
Propositional Logic: History

Aristotle
- Law of contradiction
  - ...it will not be possible to be and not to be the same thing
- Law of excluded middle
  - Everything must either be or not be
- Man is mortal, Socrates is a man, therefore, Socrates is mortal

Stoic Philosophers (3rd century BC)
- Basic inference rules
  - If \( p \) then \( q \); not \( q \); therefore not \( p \)

De Morgan and Boole (19th century)
- Symbolic logic – “automated”, “mechanical”

C. Shannon (1930s)
- Proposal to use digital hardware

Simple Example: Boolean Variables

Ed goes to the party if and only if Stella does

Choose “Boolean variables” for 2 events
Each boolean variable is either TRUE or FALSE

\[ E: \text{Ed goes to party} \]
\[ S: \text{Stella goes to party} \]

Relationship between \( E \) and \( S \)?
\[ E = S \]
Simple Example: Logical AND

*Ed goes to the party if and only if Dan does not and Stella does.*

Choose “Boolean variables” for 3 events:

- **E**: Ed goes to party
- **D**: Dan goes to party
- **S**: Stella goes to party

\[
E = (\text{NOT } D) \text{ AND } S
\]

Alternately: \( E = \overline{D} \text{ AND } S \)

Simple Example: Logical OR

*Ed goes to the party if and only if Dan goes or Stella goes*

\[
E = D \text{ OR } S
\]

E is TRUE if one or both of D and S are TRUE

**CAUTION:** In everyday language OR has additional meaning!

Example: You can eat an orange OR an apple

Use term “Exclusive OR” or “XOR” for this usage
Boolean Expressions

Composed of Boolean variables (True=1, False=0)

Operators: **AND**, **OR**, and **NOT**
- \( D \text{ AND } (P \text{ OR } \neg Q) \)
- \( C \text{ OR } D \text{ OR } E \)

Boolean Algebra Shorthand

<table>
<thead>
<tr>
<th>( A \text{ AND } B )</th>
<th>( A \text{ OR } B )</th>
<th>( \neg A )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A \cdot B ) (( AB ))</td>
<td>( A + B )</td>
<td>( \bar{A} )</td>
</tr>
<tr>
<td>( 0 \cdot 0 = 0 )</td>
<td>( 0 + 0 = 0 )</td>
<td>( 0 = 1 )</td>
</tr>
<tr>
<td>( 0 \cdot 1 = 0 )</td>
<td>( 1 + 0 = 1 )</td>
<td>( 1 = 0 )</td>
</tr>
<tr>
<td>( 1 \cdot 1 = 1 )</td>
<td>( 1 + 1 = 1 )</td>
<td>( \bar{1} = 0 )</td>
</tr>
</tbody>
</table>

What does Boolean Logic have to do with Computers?
**Boolean gates**

High voltage = 1  
Low voltage = 0

\[ x \cdot y \]  
**AND:** Output voltage is high  
if *both* of the input voltages are high;  
otherwise output voltage low.

\[ x + y \]  
**OR:** Output voltage is high  
if *either* of the input voltages are high;  
otherwise output voltage low.

\[ \overline{x} \]  
**NOT:** Output voltage is high  
if the input voltage is low;  
otherwise output voltage high.

(implicit extra wires for power)

**Combinational circuit**

Boolean gates connected by wires with no cycles  
- Wires transmit voltage (and hence values)

- Dots are shorthand for inverter

- Crossed wires that are not connected are sometimes drawn like this.
Circuits compute functions

Every combinational circuit computes a Boolean function of its inputs.

Combinational circuits and control

How would you express:

- “If data has arrived and packet has not been sent, send a signal”
- Need 3 boolean variables: D, P, S

\[ S = D \text{ AND (NOT P)} \]

D Data arrived? \[ S \] Send signal

P Packet sent?
Truth Table

Lists the truth value of the Boolean expression for all combinations of values for the variables.

**Boolean Expression**

\[ E = \overline{D} \text{ AND } S \]

**Truth table**

Write \( E \) for all possible values of \( D, S \).

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Truth Table Example

Boolean Expression: \( E = D \text{ OR } \bar{S} \)

What is \( E \)??

<table>
<thead>
<tr>
<th>D</th>
<th>S</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Truth Table Example

Boolean Expression: \( E = D \text{ OR } \bar{S} \)

What is \( E \)??

<table>
<thead>
<tr>
<th>D</th>
<th>S</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Three Equivalent Representations

Boolean Expression

\[ E = S \text{ AND } \overline{D} \]

Boolean Circuit

![Boolean Circuit Diagram]

Truth table:
Value of \( E \) for every possible \( D, S \).
\[
\begin{array}{ccc}
D & S & E \\
0 & 0 & 0 \\
0 & 1 & 1 \\
1 & 0 & 0 \\
1 & 1 & 0 \\
\end{array}
\]

Ben Revisited

Ben only rides to class if he overslept, but even then if it is raining he’ll walk and show up late (he hates to bike in the rain). But if there’s an exam that day he’ll bike if he overslept, even in the rain.

What boolean variables do we need?

- \( B \): Ben Bikes (Output)
- \( R \): Raining
- \( E \): Exam today
- \( O \): Overslept

Can you create Boolean expression for \( B \) in terms of \( R, E \) and \( O \)?
Ben Revisited

Ben only rides to class if he overslept, but even then if it is raining he’ll walk and show up late (he hates to bike in the rain). But if there’s an exam that day he’ll bike if he overslept, even in the rain.

What boolean variables do we need?

- B: Ben Bikes (Output)
- R: Raining
- E: Exam today
- O: Overslept

\[ B = O \cdot \overline{R} + O \cdot E \]

Ben’s truth table

Ben only rides to class if he overslept, but even then if it is raining he’ll walk and show up late. But if there’s an exam that day he’ll bike if he overslept, even in the rain.

<table>
<thead>
<tr>
<th>O</th>
<th>R</th>
<th>E</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Ben’s truth table

Ben only rides to class if he overslept, but even then if it is raining he’ll walk and show up. But if there’s an exam that day he’ll bike if he overslept, even in the rain.

<table>
<thead>
<tr>
<th>O</th>
<th>R</th>
<th>E</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Truth table → Boolean expression

Sum of Products:
Use OR of all input combinations that lead to TRUE output

\[ B = O \cdot R \cdot \bar{E} + O \cdot \bar{R} \cdot E + O \cdot R \cdot E \]

Can simplify expression:

\[ B = O \cdot \bar{R} + O \cdot E \]

Note:
AND, OR, and NOT gates suffice to implement every Boolean function!
Three Equivalent Expressions

C = (NOT A)B + A(NOT B)
C = A XOR B

Use DeMorgan’s Law to Simplify

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Today’s Summary

Today’s topics

- Boolean logic: Operates on True (1) and False (0)
  - Operators: AND, OR, NOT
- Three equivalent representations:
  - Boolean expressions
  - Combinational circuits
  - Truth Tables

Announcements

- Homework 5 due today
- Grading of Project 1