Chapter 1: Nuts and Bolts

CS105: Great Insights in Computer Science

The Lowly Bit

- Chemistry has its molecules.
- Physics has its strings.
- Math has its sets. \( \mathbb{N}_i := \{1, 2, \ldots, i\} \subseteq \mathbb{N} \)
- Computer science has its bits.
  - They are the smallest unit of information.
  - True (1, on)
  - False (0, off)
What's a "Bit"?

- Like “craptacular” (crap + spectacular) or “chillax” (chill + relax); it is a *portmanteau*. More?

- It’s a contraction of “binary digit”, used by Claude Shannon, attributed to John Tukey.

- In our decimal (base ten) number system, a digit can be any of the ten values 0,...,9.

- In the binary (base 2) number system, a digit can be any of the two values 0 or 1.

Why a Bit?

- Bits have the property of being...
  - **simple**: There’s no smaller discrete distinction to be made.
  - **powerful**: sequences of bits can represent seemingly anything!
0/1, Black/White

- Escher created spectacular images with just a single distinction.
- “...God divided the light from the darkness.” (Genesis Chapter 1, Line 4)
- Ying and Yang

Bits Abound

- Nearly everything has gone digital.
- Digital sound: groups of bits represent the acoustic intensity at discrete time points.
- Digital images: groups of bits represent the color at each particular discrete image location: pixel.
But There Are Lots of Them

- iPod: 60Gb. 1Gb = one billion bytes, each of which is 8 bits.
- Format uses 128 Kbps (kilobits or 1000 bits per second of sound).
- So, 62,500 minutes of sound or 15,000 songs at 4 minutes per song.
- Screen: 320x240 pixels, each of which stores 1 byte each of R,G,B intensity (24 bits). That’s 76,800 pixels and 1.8M bits.
- At 30 frames per sec., that’s 55.3 million bits per second or 144 min. of (quiet) video.

“Regular” Algebra

- Variables stand for numbers.
  - $x = 3, p = \frac{22}{7}, a = -2$
- Binary operations (two numbers in, one out)
  - addition: $2 + 3 = 5$; subtraction: $1 - 4 = -3$
  - multiplication: $\frac{2}{3} \times \frac{7}{4} = \frac{7}{6}$
- Unary operations (one number in, one out)
  - negation: $-(11) = -11$, square root: $\sqrt{16} = 4$
### Boolean Algebra

- Variables stand for bits (True/False).
  - $x = \text{True}, \ p = \text{False}$
- Binary operations (two bits in, one out)
  - and ("conjunction"): True and False = False
  - or ("disjunction"): True or False = True
- Unary operations (one number in, one out)
  - not ("negation"): not (False) = True

### Follow the Rules #1

- When is it ok to be in an R movie? You are 17-or-older or you have an adult guardian with you.
  - $x$: Person is 17 or older.
  - $y$: Person is accompanied by an adult guardian.
  - $x \text{ or } y$: Person can see an R-rated movie.
“OR” Examples

• Bill is 22 and is seeing the movie with his stepdad.
  • $x = \text{True}, y = \text{True}, x \text{ or } y = \text{True}$
• Samantha is 17 and is seeing the movie alone.
  • $x = \text{True}, y = \text{False}, x \text{ or } y = \text{True}$
• Seth is 16 and is there with both of his parents.
  • $x = \text{False}, y = \text{True}, x \text{ or } y = \text{True}$
• Jessica is 13 and is there with friends from school.
  • $x = \text{False}, y = \text{False}, x \text{ or } y = \text{False}$

Follow the Rules #2

All vehicles parked on University property must be registered and display a valid Rutgers permit.

• $x$: Car is registered.
• $y$: Car displays a valid Rutgers permit.
• $x$ and $y$: Car can be parked at Rutgers.
“AND” Examples

• Bill’s car is registered and displays a valid permit.
  • \( x = \text{True}, \ y = \text{True}, \ x \text{ and } y = \text{True} \)

• Samantha’s is registered, but the permit fell off.
  • \( x = \text{True}, \ y = \text{False}, \ x \text{ and } y = \text{False} \)

• Al’s registration expired, but his permit is still ok.
  • \( x = \text{False}, \ y = \text{True}, \ x \text{ and } y = \text{False} \)

• Jessica is visiting with no registration or permit.
  • \( x = \text{False}, \ y = \text{False}, \ x \text{ and } y = \text{False} \)

Follow the Rules #3

• It is not allowed by federal law to provide tobacco products to persons under 18 years of age.
  • \( x: \) Person is under 18.
  • \( \text{not } x: \) Person can buy tobacco
“NOT” Examples

- Samantha is 17 and is buying cigarettes.
  - $x = True$, not $x = False$

- Seth is 21 and purchased a cigar.
  - $x = False$, not $x = True$

And, Or, Not

- The most important logical operations are and, or, and not.

- $x \text{ and } y$ is True only if both $x$ and $y$ are True.

- $x \text{ or } y$ is True only if either $x$ or $y$ are True.

- $\text{not } x$ is True only if $x$ is False.

- A lot like their English meanings, but unambiguous.
Relating And/Or/Not

• **Note:** not (not x) = x
• **DeMorgan’s Law:** not flips ands and ors
  • not (x and y) = (not x) or (not y)
  • not (x or y) = (not x) and (not y)

DeMorgan’s Law Example

• x: Rob can juggle.
• y: Rob can ride a unicycle.
• not (x and y): Rob is not a juggling unicycler. (It’s not the case that Rob can **both** juggle and ride a unicycle.)

  Equivalently:
  • (not x) or (not y): Either Rob can’t juggle or Rob can’t ride a unicycle. (Rob fails at at least one of those things.)
Implementing Logic

- Clearly, our brains can handle these sorts of expressions.
- But, can we automate them?
- Yes, obviously, but let’s start with a really simple way to do it before we move on to fancier stuff.

Simple Circuit

Switch A is either on or off making the light either on or off: lightOn = A.

Symbols:
- battery
- switch
- bulb
- ground (completes circuit)

http://scratch.mit.edu/projects/cs105/35907
Multiple Switches

Switches A and B are wired in parallel: either will light the bulb.

Switches C and D are wired in series: both are needed to light the bulb.

http://scratch.mit.edu/projects/cs105/35905
http://scratch.mit.edu/projects/cs105/35909

Multiple Circuits

Special switches allow a single mechanical switch to control two circuits simultaneously.
**miniNim**

- There’s a pile of objects, say 10.
- On her turn, a player can take away either one or two objects.
- Players alternate.
- The player to take the last object wins.

---

**Nim5Bot: Game Tree**

- Let’s start by considering the 5-object version.
- We’ll design a strategy for the computer “C” to beat the user “U”.

![Game Tree Diagram]
**Further Considerations**

- To win miniNim: if possible, remove pieces to leave opponent with a multiple of 3.
- Why does it work? We win if opponent has 3; if opponent has a multiple of 3, can leave her with next smaller multiple of 3.
- What if goal is to not take the last object?
- What if we can take 1, 2, or 3 objects per round? 2 or 3? 1 or 3? Is there a general rule?

**Complete Nim5 Logic**

- fiveLeft = True
- threeLeft = takeOne₁
- twoLeft = takeTwo₁
- C-Win = (threeLeft and (takeOne₂ or takeTwo₂)) or (twoLeft and takeOne₂)
- U-Win = twoLeft and takeTwo₂
What a Headache!

- I tried to create the circuit diagram for Nim10 and couldn’t do it. Why?
  - Since some switches are used in multiple places, needs more than a double-throw switch.
  - Since values are reused, hard to keep track of different circuits.
  - Appears to need a separate circuit for each output: Gets too complex too fast.
Bits, Inside and Out

Inputs: Switches

A = False
B = True

Outputs: Lights

light1On = False
light2On = True

Inside: Logic

AND
AND
AND
AND
OR
OR
OR
OR
NOT
NOT

Need a New Approach

• Let’s consider an alternate way of building “and” and “or” logic.
• Makes simple things more complex.
• Makes complex things much simpler!
• That’s a tradeoff we can deal with.
Electricity Activated Switches

- Easy to make “and” and “or” out of switches, but we need to make switches that other switches can switch!

Switch Switcher

Before, we used switches to control the flow of current; now, we will use current (via electromagnetism) to control switches (which control the flow of current)!
Relay Circuit: A=F, B=F

A = False
B = False
lightOn = False

Relay Circuit: A=F, B=T

A = False
B = True
lightOn = False
What Is This Thing?

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>

A. “or” gate  
B. “and” gate  
C. “not” gate  
D. none of these

And One For “Not”

- We saw several slides ago that a single relay “inverts” its input signal, turning a True to a False and vice versa.

- These output summaries are known as “truth tables”.

A = not B
Abstraction: The Black Box

- Our new relay-based “and” and “not” gates take current, not switches, as input.
- As a result, they are easier to chain together.
- The original switch-based scheme did not support this kind of modularity.

A Third Black Box

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>False</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>2</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>3</td>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<td>4</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>

It’s an “or gate”: a black box built out of other black boxes!
Simplified Nim5 Circuit

How to Make a Gate

- switches/relays
- hydraulic valves
- tinkertoys
- silicon: semiconductors/transistors
- soap bubble
- DNA
- quantum material
- optics
- nanotubes
- neurons
- dominoes
- legos/marbles
Movie

NOT Gate (v3)
Or Gate (v4)

Wire, transmitting bits (v1)

Release bottom row first
Could It Work?

- My domino “or” gate requires 24 dominoes.
- The first Pentium processor had 3.3M transistors, or roughly 800K gates.
- So, perhaps 19M dominoes needed.
- World record for domino toppling: 4M.
- Oh, and the Pentium did its computations 60M times a second, whereas dominoes might require a week to set up once.