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”.

```
C  C  C  U
```

```
C  C
```

```
5 left
```

```
takeOne  takeTwo
```

```
3 left
```

```
takeOne  takeTwo
```

```
2 left
```

```
takeOne  takeTwo  takeOne  takeTwo
```

```
C  C  C
```

```
WIN
```
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.
Inputs: Switches
Outputs: Lights

Inside: Logic

A = False
B = True

light1On = False
light2On = True

• 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)!
• redo following example with 2 relays

Relay Circuit: A=F, B=F

A = False
B = False
lightOn = False
Relay Circuit: A=F, B=T

A = False
B = True
lightOn = False

Relay Circuit: A=T, B=F

A = True
B = False
lightOn = False
Relay Circuit: $A=T$, $B=T$

- $A = \text{True}$
- $B = \text{True}$
- $\text{lightOn} = \text{True}$

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”.

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

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>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>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>True</td>
</tr>
<tr>
<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)

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.