What Computers Can and Can’t Do!

this_stuff_is_cool = True
while this_stuff_is_cool:
turtle.dance()  # dance method?!

Is there a method to the Prof’s madness?
Introduction

• My name
  Jessica Wu (Prof Wu)

• My hometown
  Taipei, Taiwan (Sugar Land, Texas)

• My research
  Phylogenetics!

• Something that I really like (a food, a movie, a color, anything!)
  Board games (I spent a stupid amount of $$ on holiday sales)

• Something about me (it can be random!)
  In grad school, on a weekend trip to Maine, my three friends and I ate ~20 lobsters among us
Stuff

• Project
  – Choose your project!
  – Milestone due this Sunday at 11:59 PM
  – Remember the pair programming rules…
    • Both people must be together at all times
    • Switch who is at the keyboard every 30 minutes
  – Friday labs are just to work on your project (no other assigned work)

• Theory
  – Material these next two weeks is on final exam
  – More practice problems on course website
The next two weeks in CS5…

(1) define “computer” precisely
(2) define “compute” precisely
(3) see what computers *provably* can’t compute
(4) go back to step (1) and define things *better*…
(5) … *until* time runs out…

*CS Foundations*  What *can* we compute…
... and *how well*?
Unifying idea: State

The **state** of a computation is all the internal information needed to take the next step.

For Picobot, *next step* is taken literally!
states as subtasks

State Machine:
- each oval represents a different robot state

surroundings
state pattern -> move new state

0 \text{x***} -> N 0
0 \text{N***} -> X 1
1 \text{***x} -> S 1
1 \text{***S} -> X 0

starting funnel

transitions move from state to state

the "go North" state

the "go South" state
Computation is a deliberate sequence of state-changes

10101001011
00000000000

bits before

10101001011
00000001110

bits after
Von Neumann Architecture

central processing unit **registers**

**CPU**  

**Von Neumann bottleneck**

**RAM**  

random access memory locations

**Instruction Register**

**General-Purpose Register, r1**

**Assembly language** is *human-readable* machine language

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>read r1</strong></td>
<td><strong>mul r2 r1 r1</strong></td>
<td><strong>add r2 r2 r1</strong></td>
<td><strong>write r2</strong></td>
<td><strong>halt</strong></td>
<td>“mnemonics” instead of bits</td>
</tr>
</tbody>
</table>

Human readable?  
I doubt it!
Programs are stored in memory in *machine language*.

Von Neumann Architecture

<table>
<thead>
<tr>
<th>IR</th>
<th>r1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction Register</td>
<td>General-Purpose Register, r1</td>
</tr>
</tbody>
</table>

- **CPU** (central processing unit) and **RAM** (random access memory) are connected by a **Von Neumann bottleneck**.
- **IR** (Instruction Register) contains the current instruction.
- **r1** (General-Purpose Register) holds the next instruction address or data.

<table>
<thead>
<tr>
<th>Address</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0000 0001 0000 0001</td>
</tr>
<tr>
<td>1</td>
<td>1000 0010 0001 0001</td>
</tr>
<tr>
<td>2</td>
<td>0110 0010 0010 0001</td>
</tr>
<tr>
<td>3</td>
<td>0000 0010 0000 0010</td>
</tr>
<tr>
<td>4</td>
<td>0000 0000 0000 0000</td>
</tr>
<tr>
<td>5</td>
<td>(all bits)</td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
</tbody>
</table>
A Computer

CPU (Processor)
- Instruction Pointer (Program Counter)
- Instruction Register
- Other Registers

RAM (Memory)

"States"

Time

10110
external input

total snapshot of computer’s state!
A model of computation: FSM

Finite State Machine
aka DFA (Deterministic Finite Automata)

Example FSM

start state
“input funnel”

transitions
“where to go”
labeled by input !

accepting states
double circled

Example input

input sequence
read left-to-right

100101

This input’s output:
Accept

Example input
FSM: Finite state machine

another input sequence 0010111 output for this input
always left-to-right

What does each state MEAN?
What does this FSM do overall?

State 0

transition on 0

transition on 1

State 1

transition on 0

transition on 1
FSM: *Finite state machine*

another input sequence always left-to-right 0010111

output for this input Reject

State 0 transition on 0 transition on 1

State 1 transition on 0 transition on 1

has not moved or even # of 1’s

What does each state MEAN? What does this FSM do overall?

odd # of 1’s
What does each state say about the current state of the input?!

Could you get the same behavior with fewer states? What's the minimum # possible? How do you know?

List three different-length inputs that this FSM **accepts**:

List three different-length inputs that this FSM **rejects**:

In general, what English phrase describes the rejected inputs?

This machine rejects strings that …

What does each state say about the current state of the input?!!

Extra! Could you get the same behavior with fewer states? What's the minimum # possible? How do you know?

**Hint:** do strings have to be in separate states?
What does each state say about the current state of the input?!?

Extra! Could you get the same behavior with fewer states? What's the minimum # possible? How do you know?

Hint: do strings have to be in separate states?
State ~ fate

Strings with different possible fates must be in different states.

Can we find three strings – all with different possible fates? (4?)

If so, then three states (or 4) are necessary! If not, fewer will be OK.
**Build-your-own FSMs**

Draw a FSM accepting strings with at least two 1s (anywhere). Others are rejected.

- **Accepted examples:** 0101, 00010110, 111011, 11
- **Rejected examples:** 0100, 1000, 000000, 1, 0, λ

**Hint:** modify this starter FSM by adding labels, transitions, and one more state:

```
  1   1   0
 /   /   /
→   →   →
```

- What's the **minimum** number of states needed? 10/10

Draw a FSM accepting strings with at least two 1s (anywhere). Others are rejected.

- **Accepted:** 11010110, 11, 0000010
- **Rejected:** 101, 0000, 111011101111

**Hint:** 1s never change the state! Another hint: make a triangle!

What's the **minimum** number of states needed? 4/10

Extra! Draw a FSM accepting strings whose third-to-last digit (3rd digit from the **right**) is a 1.

- **Accepted:** 010001, 011
- **Rejected:** 11000100, 11, 0

What's the **minimum** number of states needed? 8/10

```
  1   1   0
 /   /   /
→   →   →
```

- **Hint:** There are FIVE more transitions – but no more states - needed here.

6/10

Draw a FSM accepting strings in which the number of zeros (0s) is a multiple of 3, so there are 0, 3, 6, … zeros. **1s don’t matter!**

- **Accepted:** 1010001, 011
- **Rejected:** 101, 0000, 110

**Hint:** Draw a tree to keep track of the characters seen so far...

10/10
Has at least two 1s... ?

Draw a FSM accepting strings with at least two 1s (anywhere). Others are rejected.

**Accepted:** 0101, 00010110, 111011, 11

**Rejected:** 0100, 1000, 000000, 1, 0, λ

**Hint** - modify this starter FSM by adding labels, transitions, and one more state:

What do we need to complete this machine?
Has at least two 1s…?

Draw a FSM accepting strings with at least two 1s (anywhere). Others are rejected.

Accepted: 0101, 00010110, 111011, 11
Rejected: 0100, 1000, 000000, 1, 0, λ

Hint - modify this starter FSM by adding labels, transitions, and one more state:

What do we need to complete this machine?
Number of 0s is multiple of 3

Draw a FSM accepting strings in which the number of zeros (0s) is a multiple of 3, so there are 0, 3, 6, … zeros. 1s don't matter.

Hint: 1s never change the state!
Another hint: make a triangle!

Accepted: 110101110, 11, 0000010
Rejected: 101, 0000, 111011101111

Minimum number of states?
Number of 0s is multiple of 3

Draw a FSM accepting strings in which the number of zeros (0s) is a multiple of 3, so there are 0, 3, 6, … zeros. 1s don't matter.

Accepted: 110101110, 11, 0000010
Rejected: 101, 0000, 111011101111

Hint: 1s never change the state!
Another hint: make a triangle!

Minimum number of states?

Combine two of these?
Number of 0s is multiple of 3

Draw a FSM accepting strings in which the number of zeros (0s) is a multiple of 3, so there are 0, 3, 6, … zeros. 1s don't matter.

Accepted: 110101110, 11, 0000010
Rejected: 101, 0000, 111011101111

Hint: 1s never change the state!
Another hint: make a triangle!

Minimum number of states?
No occurrences of $110$?

Draw a FSM accepting strings that do NOT anywhere contain the pattern $110$.

Accepted: $1010001$, $0001011$

Rejected: $101001100$, $011001$

Hint: There are FIVE more transitions – but no more states - needed here.

Which transitions are still needed here?
No occurrences of 110?

Draw a FSM accepting strings that do NOT anywhere contain the pattern 110.

Accepted: 1010001, 0001011
Rejected: 101001100, 011001

Hint: There are FIVE more transitions – but no more states needed here.

Which transitions are still needed here?
Third character is a 1

Draw a FSM accepting strings in which the third digit (from the left) is a 1.

- **Accepted:** 1010001 and 0110
- **Rejected:** 11000100 and 11

*Minimum number of states?*
Third character is a 1

Draw a FSM accepting strings in which the third digit (from the left) is a 1.

Accepted: 1010001 and 0110
Rejected: 11000100 and 11

Minimum number of states?

Why must \( s_1 \) and \( s_2 \) be separate states?

Why must \( s_1 \) be a separate state?
Third-to-last character is a 1?

Draw a FSM accepting strings whose third-to-last digit (from the right) is a 1.

**Hint:** Draw a tree to keep track of the characters seen so far...

**Accepted:** 0100 and 01101

**Rejected:** 101001 and 11

Minimum number of states?
Third-to-last character is a 1

Which states are accepting states?
What are the missing transitions?

Do we need 15 states?

Fill this in in your notes.
Third-to-last character is a 1

Which states are accepting states?
What are the missing transitions?

I don't accept this solution!
Something's not right here: it's down-right harrowing!

Do we need 15 states?
Have I seen ATG?

Fill this in in your notes.

Now, build a similar finite state machine for AAA!
Have I seen ATG?

Now, build a similar finite state machine for AAA!
FSMs are everywhere!

Locks

q_{start} → q_1 → q_{12} → q_{12,3} → q_{12,5,34} → \text{q}_{12,5,34}

q_2 → q_{13} → q_{12,4} → \text{q}_{12,5,34}

q_3 → q_{14} → \text{q}_{12,5,34}

q_4

q_5

q_{23}

\text{q}_{12,5,34}
FSMs are everywhere!

mechanical vending machine

www.youtube.com/watch?v=85C4eh0mEJg @ 1:42!
FSM ~ Game AI

Recognize this street?

Here, it’s *Ghost AI*
All robots use FSM control

What states can you “factor out” from watching this towel-folding?
Towel-folding states!

(a) Towels remaining in pile?
   Yes → Done
   No → (b) Pickup towel from pile

(b) Pickup towel from pile
   (c) Detect first corner
      Miss → Fail
      Succeed → Grasp first corner
      (d) Grasp first corner
         Miss → Fail
         Succeed → Drop back on pile
         (e) Detect second corner
            Miss → Fail
            Succeed → Grasp second corner
            (f) Grasp second corner
               Miss → Fail
               Succeed → Fold and stack towel

(k)-(o) Fold and stack towel
           (i)-(j) Regrasp short side
              Long side → Short side
              Mis-grasp → Release from one gripper
              (g)-(h) Untwist, check configuration

Fig. 2. The state machine model of the procedure: dashed lines indicate failure recovery cases. The images show an actual run.
Towel-folding?

singled out as a questionable use of dollars...
An autonomous vehicle’s FSM

Fig. 9. Situational Interpreter State Transition Diagram. All modes are sub-modes of the system RUN mode (Fig 4(b)).
FSMs driving robots...

DARPA Urban Challenge 2007
FSMs driving robots...

MIT’s car, Talos - and its sensor suite
State-machine *limits*?

Are there *limits* to what FSMs can do?

They can’t necessarily drive safely...

But are there any binary-string problems that FSMs can't solve?

Let’s build a FSM to accept bit strings with the **SAME NUMBER** of 0s as 1s

<table>
<thead>
<tr>
<th><strong>rejected</strong></th>
<th>01100</th>
<th>01110</th>
<th>01000</th>
<th>000</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>accepted</strong></td>
<td>011001</td>
<td>0110</td>
<td>10</td>
<td>λ (this last string is empty)</td>
</tr>
</tbody>
</table>
State-machines are limited.

**FSMs can’t count**

at least not arbitrarily high...

We need a **more powerful model** than FSMs...

What do we need to add?

Thursday: **Turing Machines**