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?
Stuff

• Project
  – Choose your project!
  – Milestone 1 due this Friday 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  x***  ->  N  0
0  N***  ->  X  1
1  ***x  ->  S  1
1  ***S  ->  X  0
**Computation** is a deliberate sequence of state-changes.
Von Neumann Architecture

processing

CPU

central processing unit *registers*

Von Neumann bottleneck

program

RAM

random access memory locations

Assembly language is *human-readable* machine language

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

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>read r1</td>
</tr>
<tr>
<td>1</td>
<td>mul r2 r1 r1</td>
</tr>
<tr>
<td>2</td>
<td>add r2 r2 r1</td>
</tr>
<tr>
<td>3</td>
<td>write r2</td>
</tr>
<tr>
<td>4</td>
<td>halt</td>
</tr>
</tbody>
</table>

“mnemonics” instead of bits

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

**Von Neumann Architecture**

- **CPU** (central processing unit) registers
- **RAM** (random access memory locations)

- **IR** (Instruction Register)
- **r1** (General-Purpose Register, r1)

- **Von Neumann bottleneck**
A Computer

CPU (Processor)

Instruction Pointer (Program Counter)

Instruction Register

Other Registers

RAM (Memory)

Instruction

Register

Other Registers

“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
FSM: Finite state machine

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

What does each state MEAN? What does this FSM do overall?
FSM: *Finite state machine*

```
0010111
```

- **State 0**: Transition on 0
- **State 1**: Transition on 0, transition on 1

Output for this input: **Reject**

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

- **State 0**: Has not moved or even # of 1’s
- **State 1**: Odd # of 1’s

Another input sequence always left-to-right.
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?

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?

In general, what English phrase describes the rejected inputs?

This machine rejects strings that ...

List three different-length inputs that this FSM accepts:

List three different-length inputs that this FSM rejects:

Try on the separate worksheet, use this page for solutions.
Worksheet

List three *different-length* inputs that this FSM **accepts**: λ, 0, 1, 10

List three *different-length* inputs that this FSM **rejects**: 011, 11, 111, 11110

*In general*, what English phrase describes the **rejected inputs**?

This machine rejects strings that …

have two 1’s in a row (anywhere in string)

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

s0: never seen two 1’s in a row, last saw 0, and ACCEPT
s1: never seen two 1’s in a row, last saw 1, ACCEPT
s2: already seen two 1’s in a row, last saw 0, REJECT
s3: already seen three or more 1’s in a row, REJECT

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:

Draw a FSM that accepts strings that don’t contain the pattern 110 anywhere.

Accepted: 1010001, 011
Rejected: 10100, 1000, 000000, 1, 0, λ

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

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: 11010110, 11, 0000010
Rejected: 101, 0000, 111011100, 1110

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

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

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

Accepted: 1010001, 011
Rejected: 11000100, 11, 0

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

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

Accepted: 0100 and 01101
Rejected: 101001 and 11

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

6/10

2/10

4/10

8/10

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.

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

Minimum number of states?

Acceptance strings:
- 110101110
- 11
- 0000010

Rejection strings:
- 101
- 0000
- 111011101111

Diagram:

- State s0: # zeros % 3 == 0
- State s1: # zeros % 3 == 1
- State s2: # zeros % 3 == 2
- State s3: # zeros % 3 == 3

Combining two of these states:

- # zeros % 3 == 1
- # zeros % 3 == 2

These are the same state!
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 \( s_1 \) and \( s_2 \) be separate states?

eat rest of input!
**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

Do we need 15 states?

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

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

Do we need 15 states?

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!
XKCD FSMs!

My Problem with Phone Alarms

Hey Hey Goodbye

Na Na Na Na Na Na Na Na
Na Na Na Na

Batman! Katamari Damacy!

Land of 1,000 Dances
Have I seen ATG?

T,G,C

T,C

A

T

A

G

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!
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!

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
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>rejected</th>
<th>accepted</th>
</tr>
</thead>
<tbody>
<tr>
<td>01100</td>
<td>011001</td>
</tr>
<tr>
<td>01110</td>
<td>0110</td>
</tr>
<tr>
<td>0100</td>
<td>10</td>
</tr>
<tr>
<td>000</td>
<td>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**