CS5's view from here...

I've got *all* my eyes on this view!

What's next?

Final Projects

CS Practice

Picobot

vPool

TextID

Board Game

More Life

Models of Computation

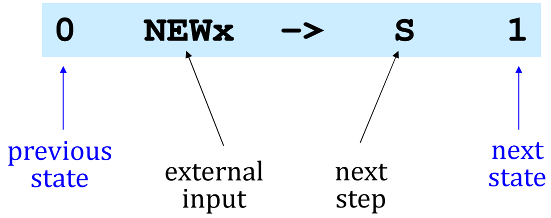
Uncomputable Problems

Impossible Problems!

... and the CS5 **finale**.

No reading this week!

Unifying idea: **State**



```
0   NEWx   ->   S   1
```

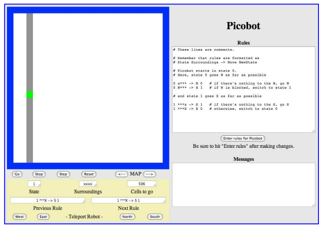
The ***state*** of a computation is

all the internal information

needed to take the next step

For Picobot, *next step* is taken literally!

States as ***subtasks***



```
# These lines are comments.

# Remember that rules are formatted as
# State Surroundings -> Move NewState

# Picobot starts in state 0.
# Here, state 0 goes N as far as possible

0 x*** -> N 0   # if there's nothing to the N, go N
0 N*** -> X 1   # if N is blocked, switch to state 1

# and state 1 goes S as far as possible

1 ***x -> S 1   # if there's nothing to the S, go S
1 ***S -> X 0   # otherwise, switch to state 0
```

Be sure to hit "Enter rules" after making changes.

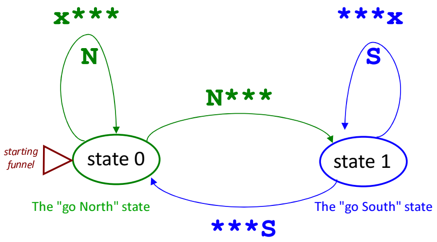
State: 1 Surroundings: xxxx Cells to go: 506

Previous Rule: 1 ***X -> S 1 Next Rule: 1 ***X -> S 1

state	surroundings pattern	->	move	new state
0	x***	->	N	0
0	N***	->	X	1
1	***x	->	S	1
1	***S	->	X	0

State Machine:

Each oval represents a different Picobot state



The "go North" state

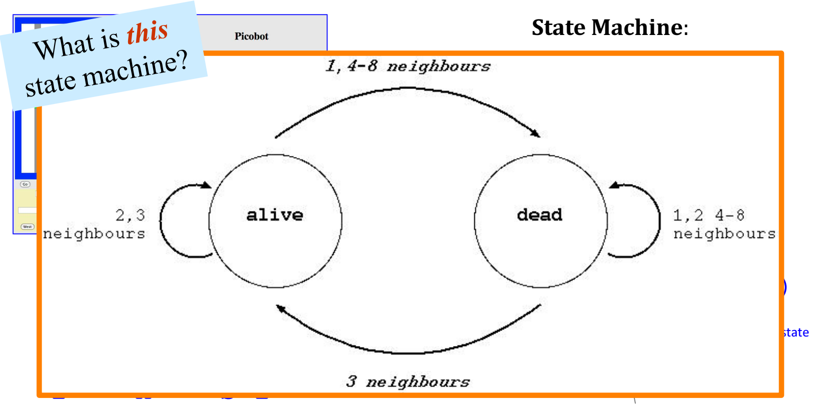
The "go South" state

Transitions move from state to state

States as ***subtasks***

State Machine:

What is *this* state machine?



transitions move from state to state

A *model* of computation: **FSM**

Finite State Machine

Example input

Input sequence

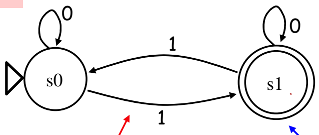
Read left-to-right

100101

This input's output:

Another accepted string:

Rejected example string:



Start state

“Input funnel”

Transitions

“Where to go”

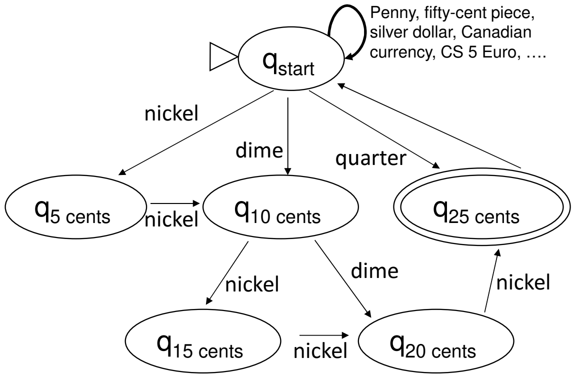
Labeled by input !

Accepting states

Double-circled

FSMs are everywhere!

Mechanical vending machine



(some transitions not shown)

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

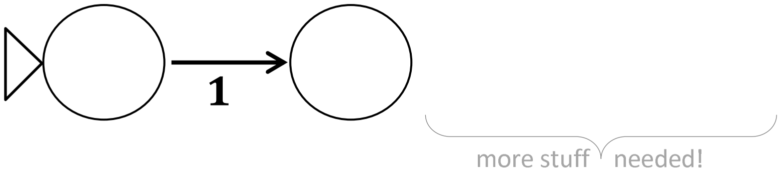
Build-your-own FSMs

2/10 Draw an FSM accepting strings with at least two **1**s (anywhere). Others are rejected.

Accepted examples: 0101, 00010110, 111011, 11

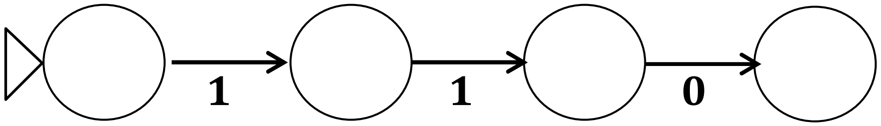
Rejected examples: 0100, 1000, 000000, 1, 0

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



6/10 Draw an FSM that accepts strings that ***don't*** contain the pattern **110** anywhere.

Accepted: 1010001, 011 **Rejected:** 101001100, 01101



Hint - there are FIVE more transitions – but no more states - needed here

Draw an FSM accepting strings in which the number of zeros (**0**s) is a multiple of 3, so there are 0, 3, 6, ... zeros. ***1**s don't matter!* 4/10

Accepted: 110101110, 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?

Draw an FSM accepting strings in which the third digit (3rd from the left) is a **1**. 8/10

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

What's the minimum number of states needed?

10/10 ***Extra!*** Draw an FSM accepting strings whose third-to-last digit (3rd from the **right**) is a **1**.

Acc: 0100 and 01101

Rej: 101001 and 11

Build-your-own FSMs

2 10 Draw an FSM accepting strings with at least two **1**s (anywhere). Others are rejected.

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

Hint - m labels, t

```
if count('1', s) >= 2:
    return True
else:
    return False
```

FSMs ~ "software circuits"

6 10 Draw an FSM that acc … contain the pattern **110** anywhere.

Accepted: 1010001, 011 **Rejected:** 101001100, 01101

```
if '110' not in s:
    return True
else:
    return False
```

Hint - there are FIVE more transitions – but no more states - needed here

Draw an FSM accepting strings in which the number of zeros (**0**s) is a multiple of 3, so there are 0, 3, 6, ... zeros. ***1**s don't matter!* 4 10

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

Hint: 1s never change the state!

Another hint: make a triangle!

```
if count('0', s)%3 == 0:
    return True
else:
    return False
```

… minimum number of states needed?

…ings in which the third digit (3rd from the left) is a **1**. 8 10

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

```
if s[2] == '1':
    return True
else:
    return False
```

```
if s[-3] == '1':
    return True
else:
    return False
```

What's the minim…

10 10 ***Extra!*** Draw an FSM accepting strings whose third-to-last digit (3rd from the **right**) is a **1**.

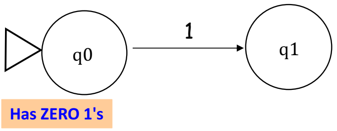
Acc: 0100 and 01101
Rej: 101001 and 11

Has at least two **1**s… ?

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

Accepted: 0101, 00010110, 111011, 11

Rejected: 0100, 1000, 000000, 1, 0



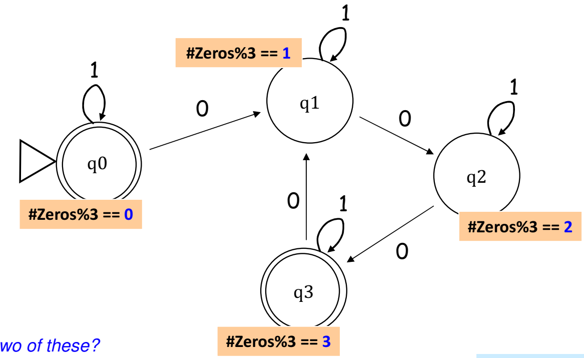
What do we need to complete this machine?

Number of **0**s is divisible by 3

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

Accepted: 110101110, 11, 0000010

Rejected: 101, 0000, 111011101111



Combine two of these?

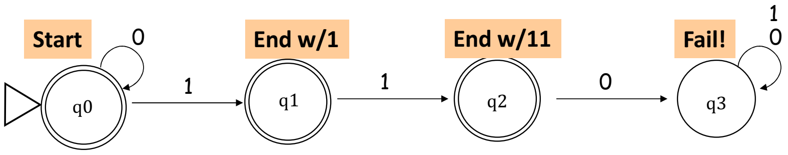
Minimum number of states?

No occurrences of **110**?

Draw an FSM accepting strings that do ***NOT*** contain the pattern **110** anywhere

Accepted: 1010001, 0001011

Rejected: 101001100, 011001



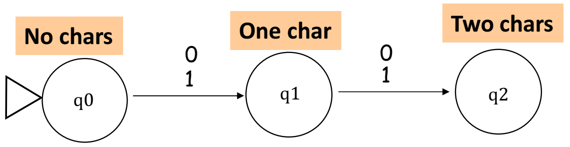
*Which **transitions** are still needed here?*

Third character is a **1**

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

Accepted: 10**1**0001 and 01**1**0

Rejected: 11**0**00100 and 11



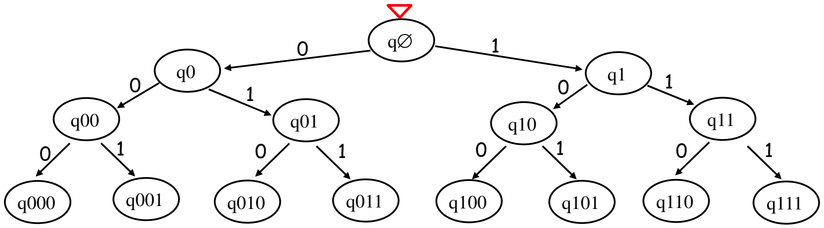
s1

Why must s1 and s2 be separate states?

s2

Minimum number of states?

Third-to-last character is a **1**

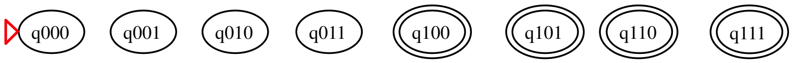


I don't accept this solution!

Something's not right here: it's down-right harrowing!

Do we ***need*** 15 states?

Third-to-last character is a **1**

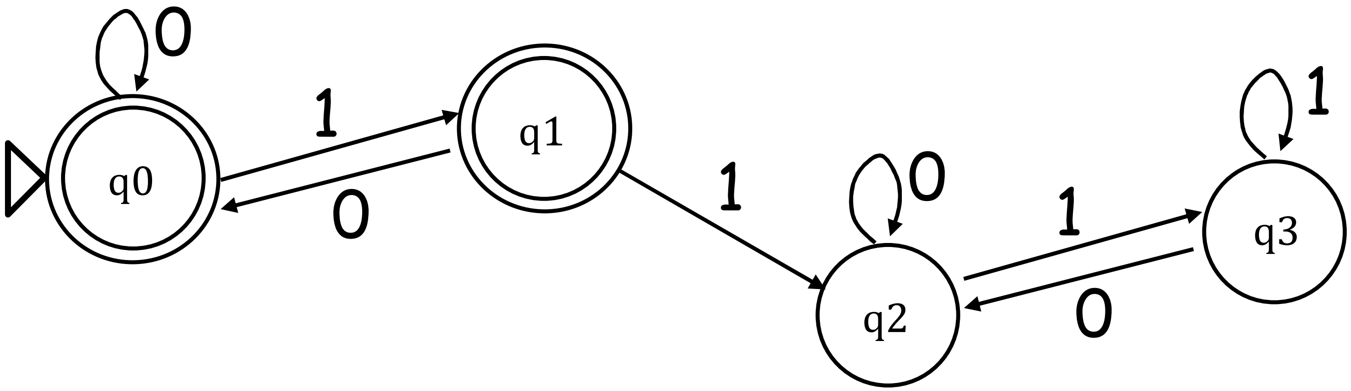


8 states?

Are 8 states required?

Quiz

Name(s) ______________________________



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?!?

q0 means ACCEPTING and ending in a 0 ... **or** ...

q1 means

q2 means ... **or** ...

q3 means REJECTING and

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?