Finite State Automata

CSCI 1030U - Intro to Computer Science
@IntroCS

Randy J. Fortier
@randy_fortier

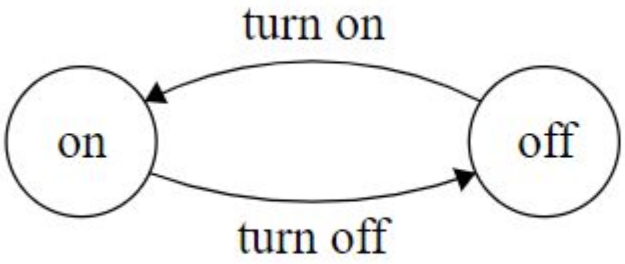
Outline

- Finite state automata
- Finite state automata types
  - Deterministic FSAs (DFAs)
  - Push-down Automata (PDAs)
  - Non-deterministic FSAs (NFAs)
- Parsers (lexical analysis)
- Conway's Game of Life

Finite State Automata

Finite State Automata (FSA)

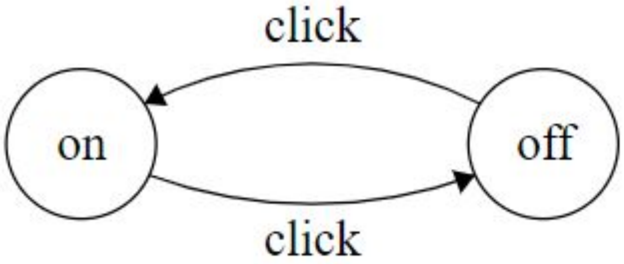
- A finite state automaton (also called a finite state machine) is a model that describes transitions between a fixed number of states
- Example:
  - Here is an FSA modelling the operation of a simple lamp:



Drawn using http://madebyevan.com/fsm/

Finite State Automata (FSA)

- The next state is determined using the current state, and the input
  - The same input may lead to different states, depending on the current state
- Example:
  - Here is an FSA modelling the operation of a push-button flashlight:



Drawn using http://madebyevan.com/fsm/

Exercise 06a.1

- Let's create an FSA to recognize any binary number with an even number of digits
  - e.g. 01101000 -> True
  - e.g. 1000111011 -> True
  - e.g. 00 -> True
  - e.g. 100 -> False
  - e.g. 10b -> False

Exercise 06a.1

- Let’s create an FSA to recognize any binary number with an even number of digits
  - States:
  - Transitions from even:
  - Transitions from odd:

Exercise 06a.1

- Let’s create an FSA to recognize any binary number with an even number of digits
  - States: even, odd
  - Transitions from even:
  - Transitions from odd:

Exercise 06a.1

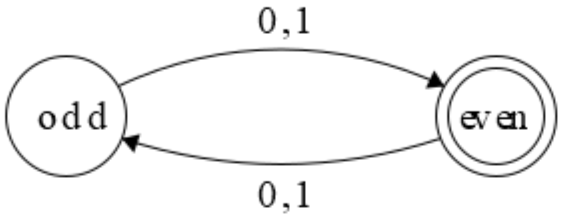
- Let's create an FSA to recognize any binary number with an even number of digits
  - States: even, odd
  - Transitions from even: 0 or 1, go to odd
  - Transitions from odd:

Exercise 06a.1

- Let's create an FSA to recognize any binary number with an even number of digits
  - States: even, odd
  - Transitions from even: 0 or 1, go to odd
  - Transitions from odd: 0 or 1, go to even

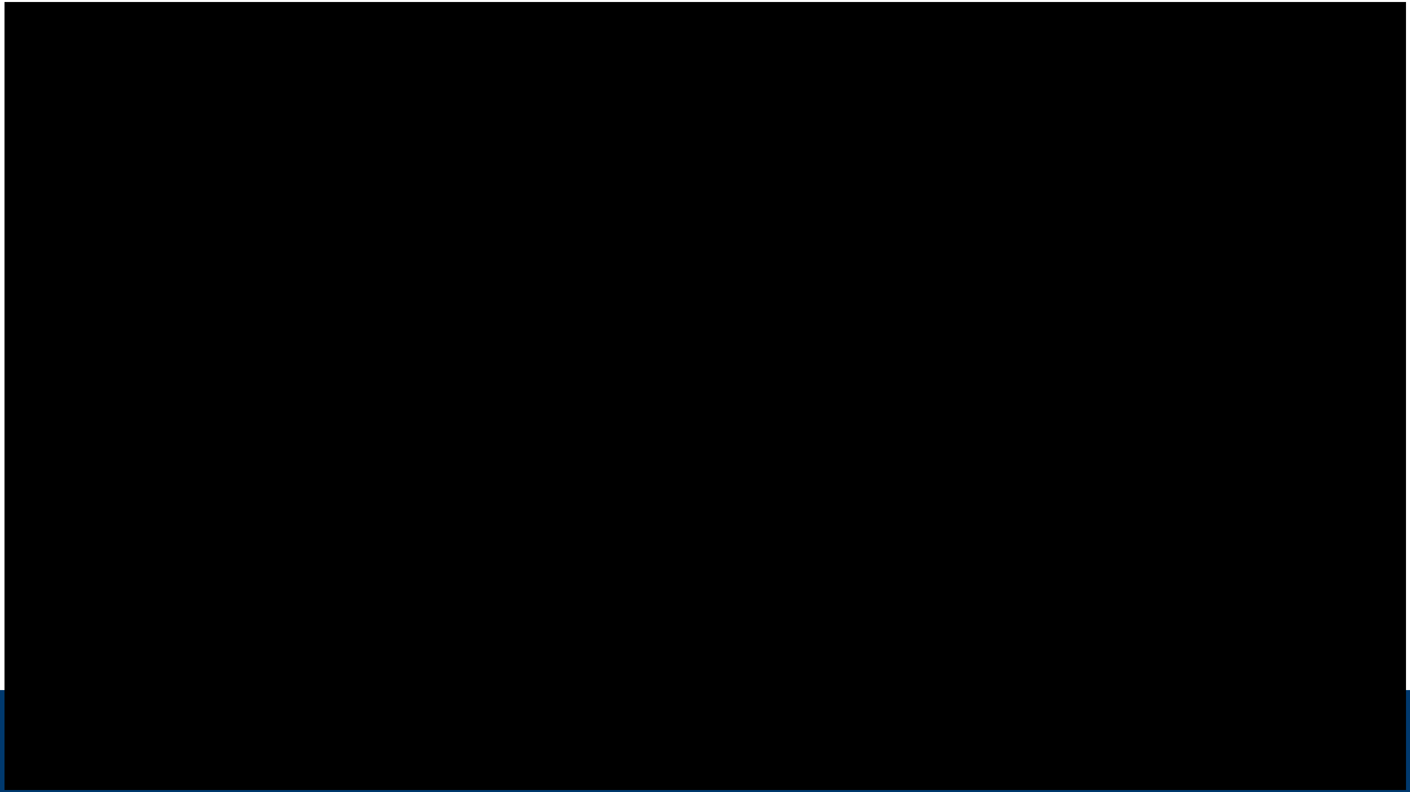
Exercise 06a.1

- Let's create an FSA to recognize any binary number with an even number of digits
  - States: even, odd
  - Transitions from even: 0 or 1, go to odd
  - Transitions from odd: 0 or 1, go to even



Drawn using http://madebyevan.com/fsm/

FSAs - Video Example 1

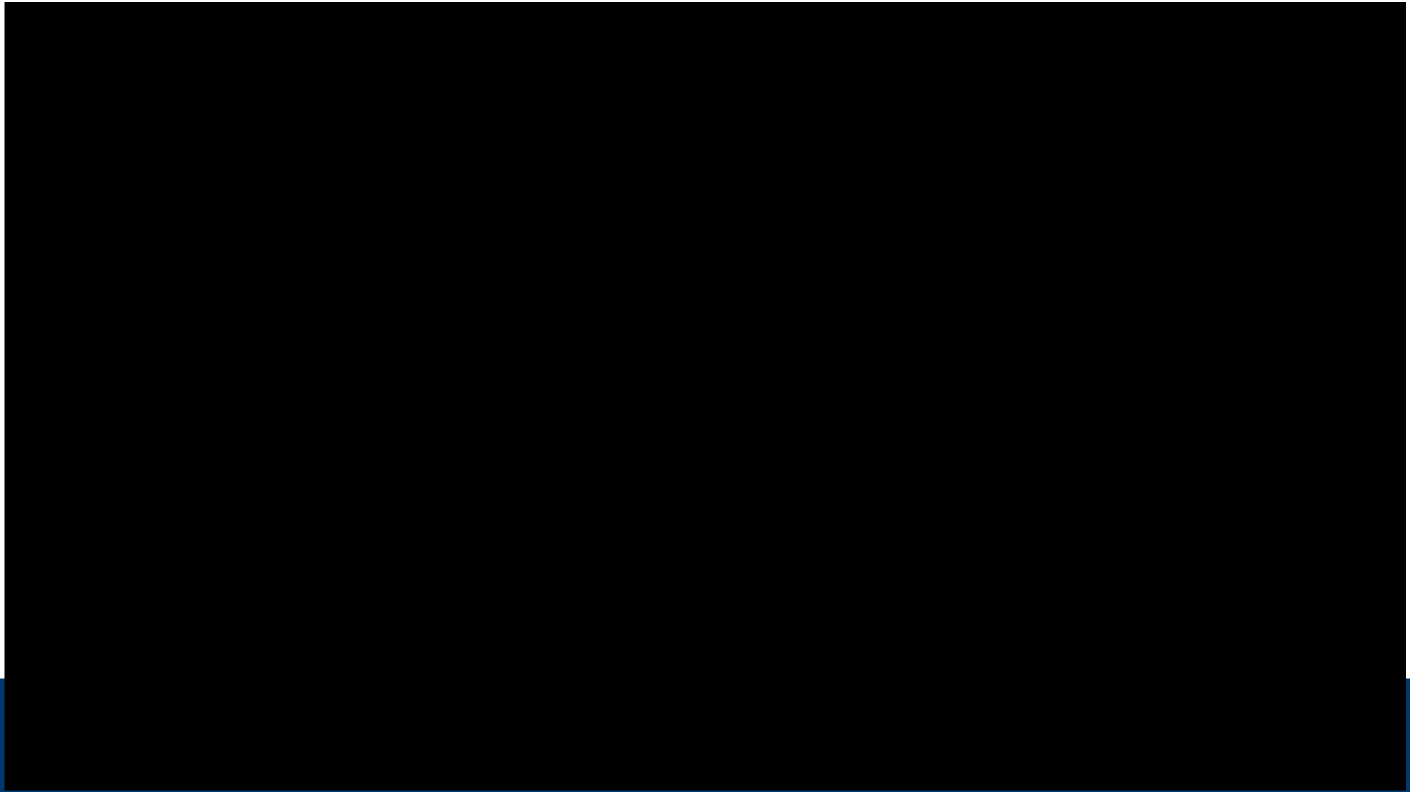


FSAs - Video Example 1

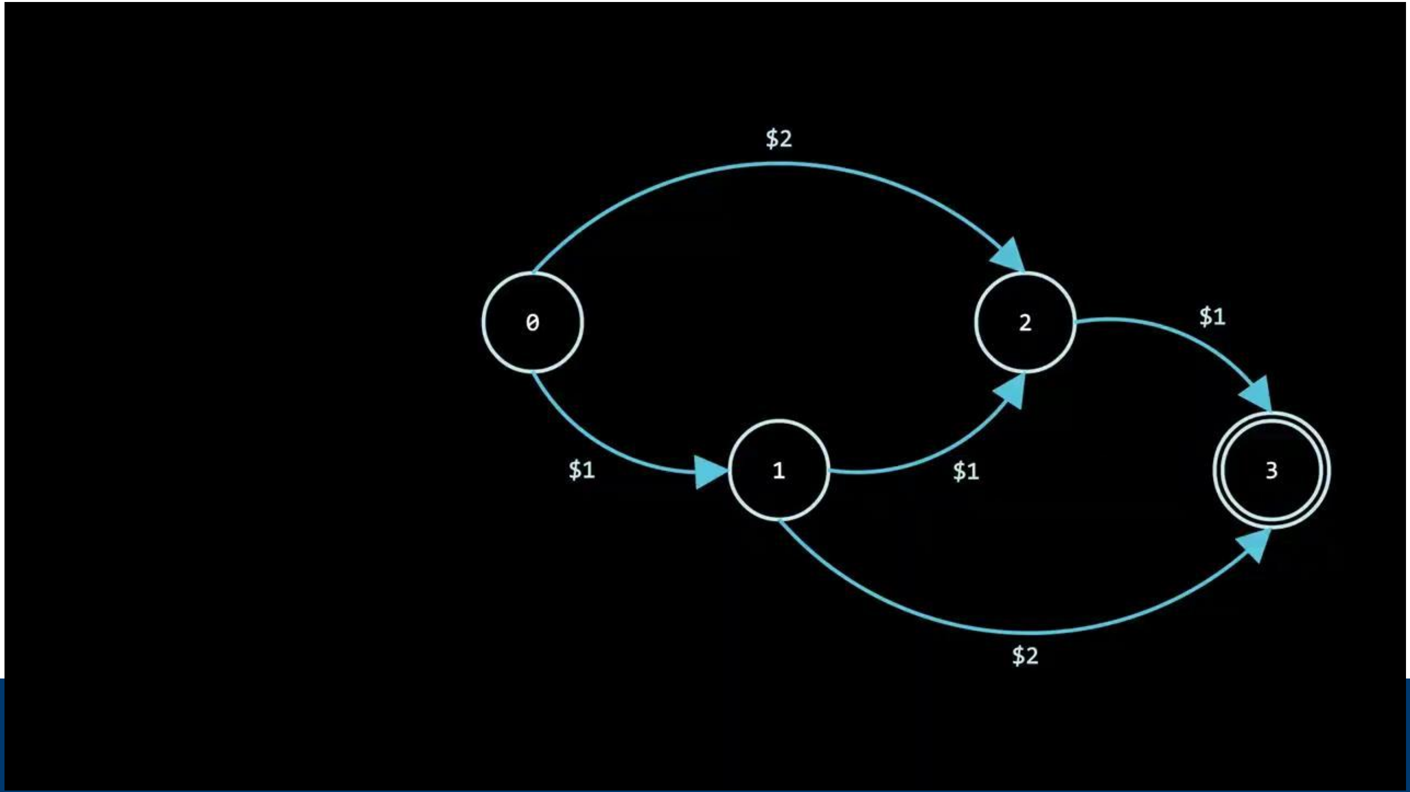
Exercise 06a.2

- Let’s create an FSA to simulate the operation of a parking gate that accepts $1 and $2 coins, and costs $3 to park
  – *Use: http://madebyevan.com/fsm/*

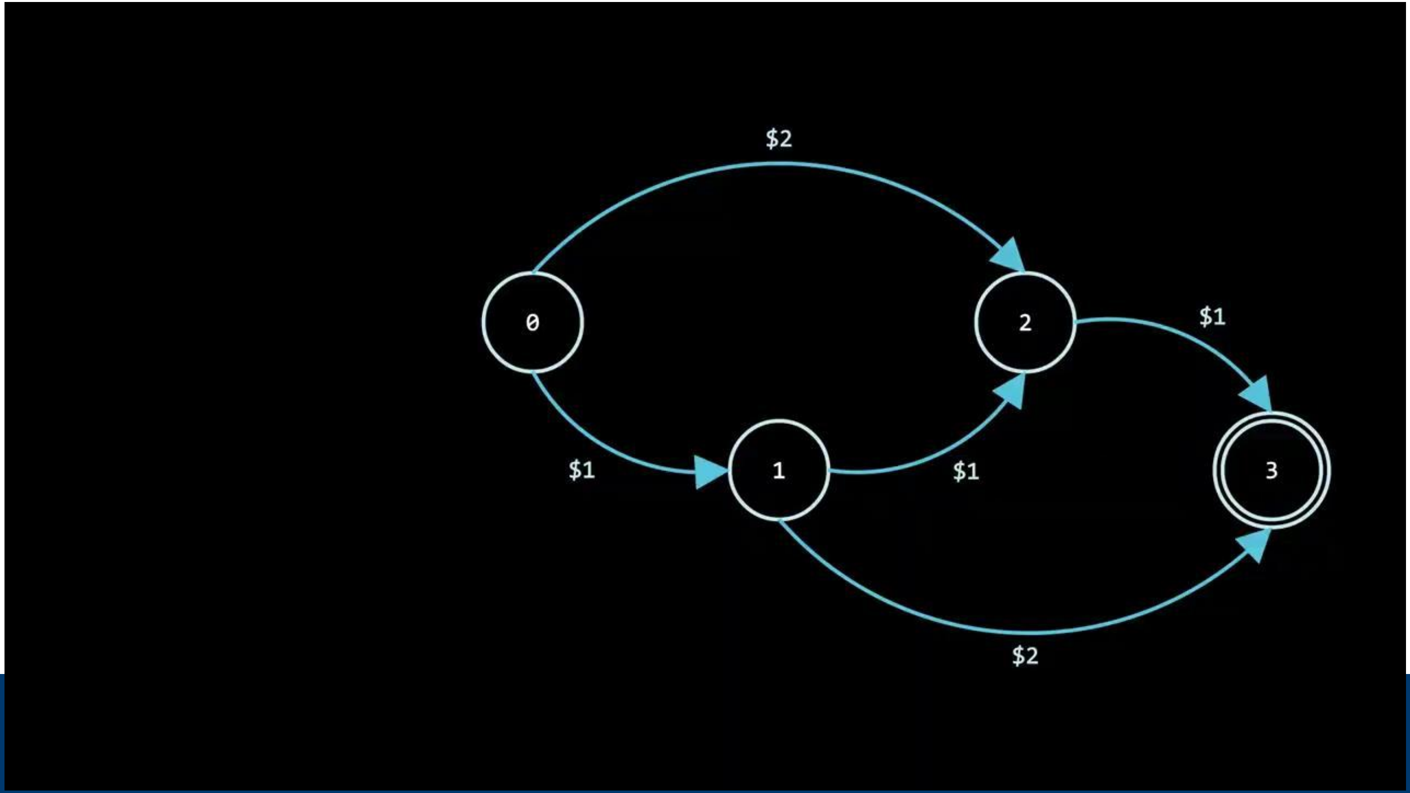
FSAs - Video Example 2



FSAs - Video Example 2



FSAs - Video Example 2

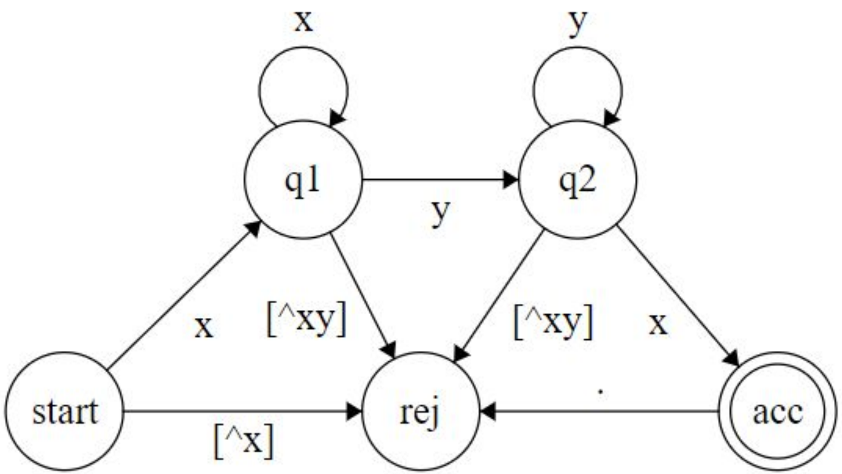


Challenge 06a.1

- Create an FSA to recognize any string with length a multiple of 4
  - Use: http://madebyevan.com/fsm/

Challenge 06a.2

- What strings does the following FSA recognize?



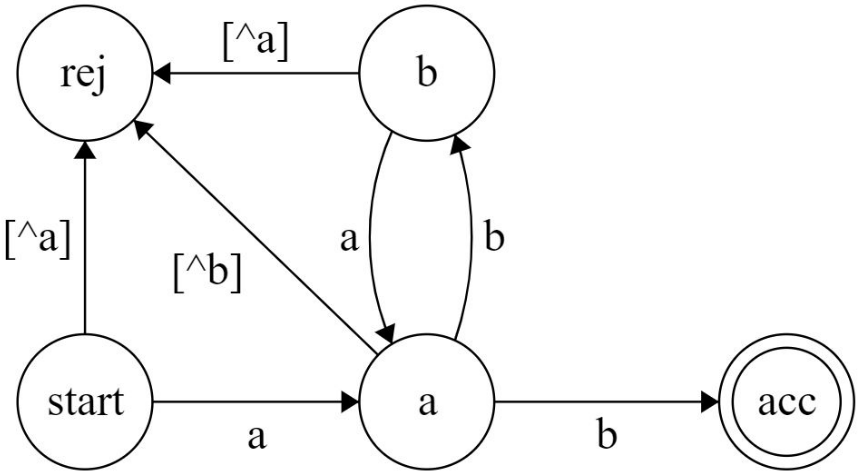
Drawn using http://madebyevan.com/fsm/

Types of Finite State Automata

- Deterministic (DFA)
  - This is the kind we've been dealing with
  - For each input, there is exactly one matching transition
- Push-down automata (PDA)
  - A DFA that also has a stack; stores more history than just the state
  - Can recognize deterministic context-free languages
- Non-deterministic (NFA)
  - For each input, there may be multiple matching transitions
  - We may choose between them randomly, or choose and then backtrack
  - Can recognize all context-free languages

Non-deterministic FSAs

- This FSA is a non-deterministic FSA (NFA):



Drawn using http://madebyevan.com/fsm/

Applications of FSAs

Uses of Finite State Automata

- FSAs have many more uses than just regular expressions
  - Machine logic
    - Robotics
    - Vending machines
    - Traffic lights
    - Turnstyles/traffic counters
  - Pattern recognition in genetics
  - AI in games (https://www.youtube.com/watch?v=JyF0oyarz4U)
  - Circuits with memory
  - Compilers and parsing
  - Natural language processing

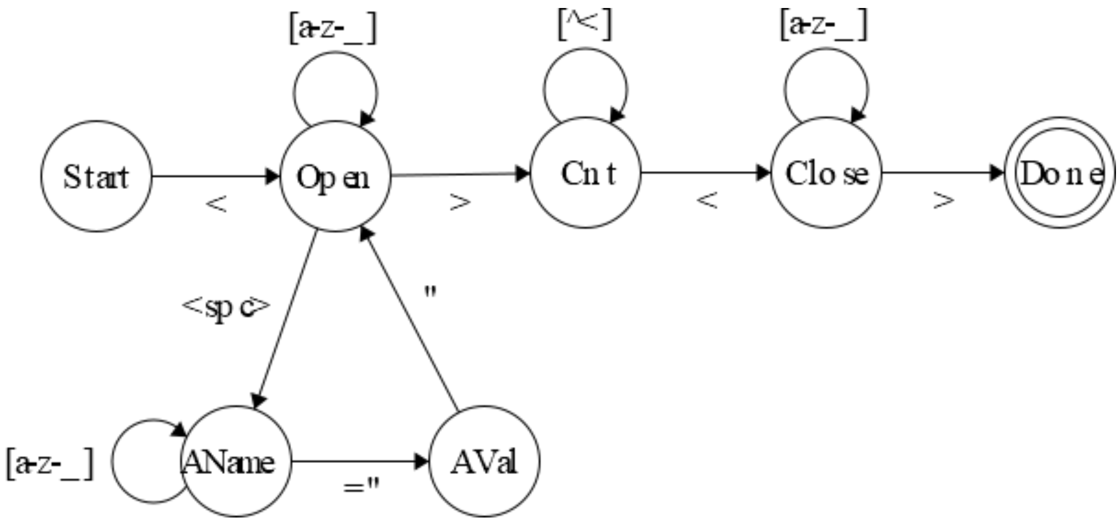
Parsing

- It is common to use FSAs within parsers, at least for relatively simple languages
  - e.g. recognizing variable names - lexical analysis
  - More complex languages require other mechanisms
- Consider parsing an HTML tag like the following:

```
<div class="title" id="details">Picnic Event Details</div>
```

Parsing

- This HTML tag might be recognized by an FSA like the following:



Conway's Game of Life

- Developed by John Conway in 1970, the game of life is a self-playing game which uses finite state automata for its main operation
  - A variant, called cellular automata, is used for this game
  - Cellular automata uses a grid of cells, and the states of those cells to determine the next state
  - The game of life can be written as a typical FSA, however

http://pi.math.cornell.edu/~lipa/mec/lesson6.html

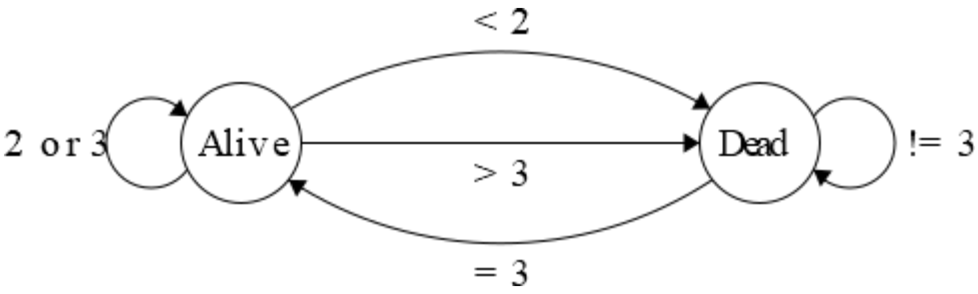
Conway's Game of Life

- The game of life involves a grid of cells in a binary state, dead or alive
- Updates follow these simple rules:
    1. Any live cell that has < 2 alive neighbours, dies (underpopulation)
    2. Any live cell that has 2-3 alive neighbours, survives
    3. Any live cell that has > 3 alive neighbours, dies (overpopulation)
    4. Any dead cell that has exactly 3 alive neighbours, becomes alive (reproduction)

http://pi.math.cornell.edu/~lipa/mec/lesson6.html

Conway's Game of Life

- A simple FSA for Conway's game of life:



http://pi.math.cornell.edu/~lipa/mec/lesson6.html

Coding Exercise 07a.2 (Time Permitting)

- Create Conway's Game of Life in Python

Wrap-up

- Finite state automata
- Finite state automata types
  - Deterministic FSAs (DFAs)
  - Push-down Automata (PDAs)
  - Non-deterministic FSAs (NFAs)
- Parsers (lexical analysis)
- Conway's Game of Life

Coming Up

- Regular expressions (regex)
  - Regular languages
  - Kleene’s Theorem
  - Regex in Python