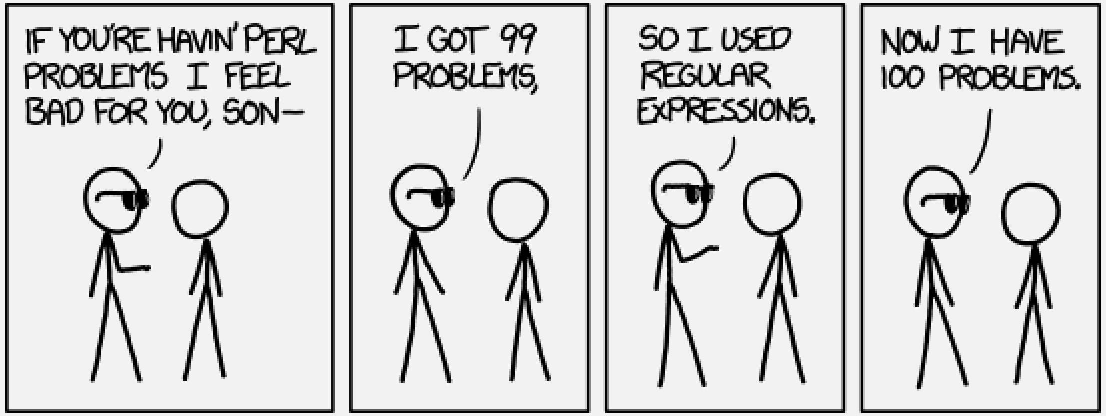
CS420
Finite Automata and Regular Languages

Instructor: Stephen Chang

Mon Sept 14, 2020

UMass Boston Computer Science

HW 0 Questions?



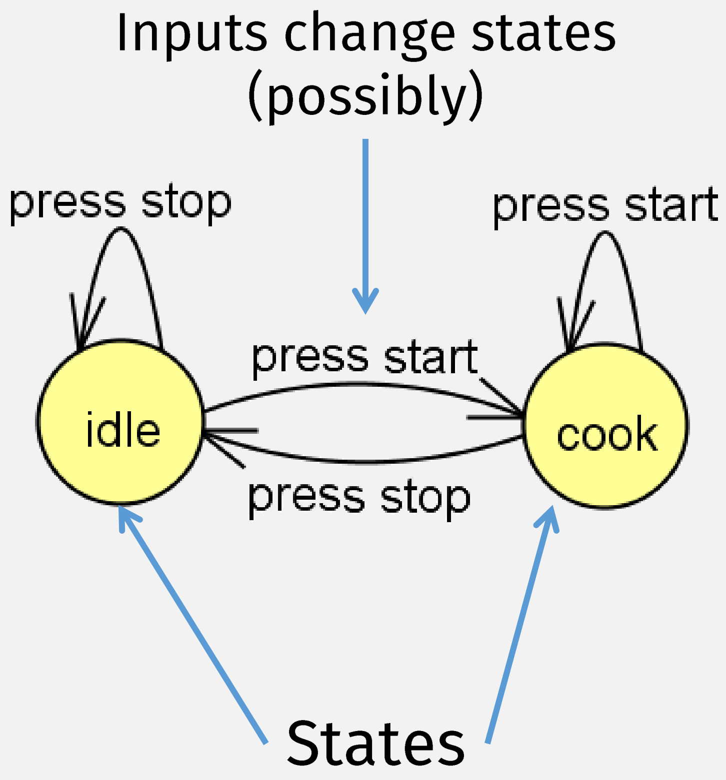
Quick Poll: Regular Expressions

The Good, the Bad, the … Weird?

Deterministic Finite Automata (DFAs)

A computational model for …

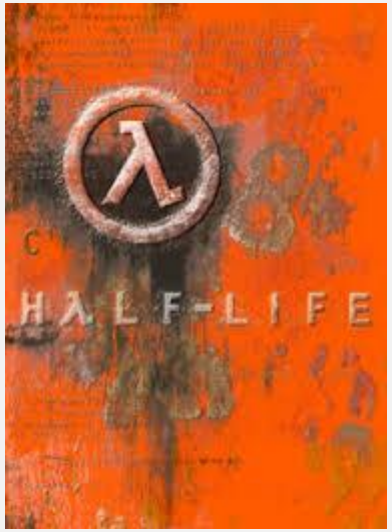
A Finite Automata (or State Machine)



Finite Automata: Not Just for Microwaves

Finite Automata:
a common
programming pattern

Finite Automata in: Video Games



ValveSoftware / **halflife**

Code | Issues 1.6k | Pull requests 23 | Actions | Projects | Wiki

5d761709a3 ▾ halflife / game_shared / bot / **simple_state_machine.h**

Alfred Reynolds initial seed of Half-Life 1 SDK

0 contributors

85 lines (67 sloc) | 2.15 KB

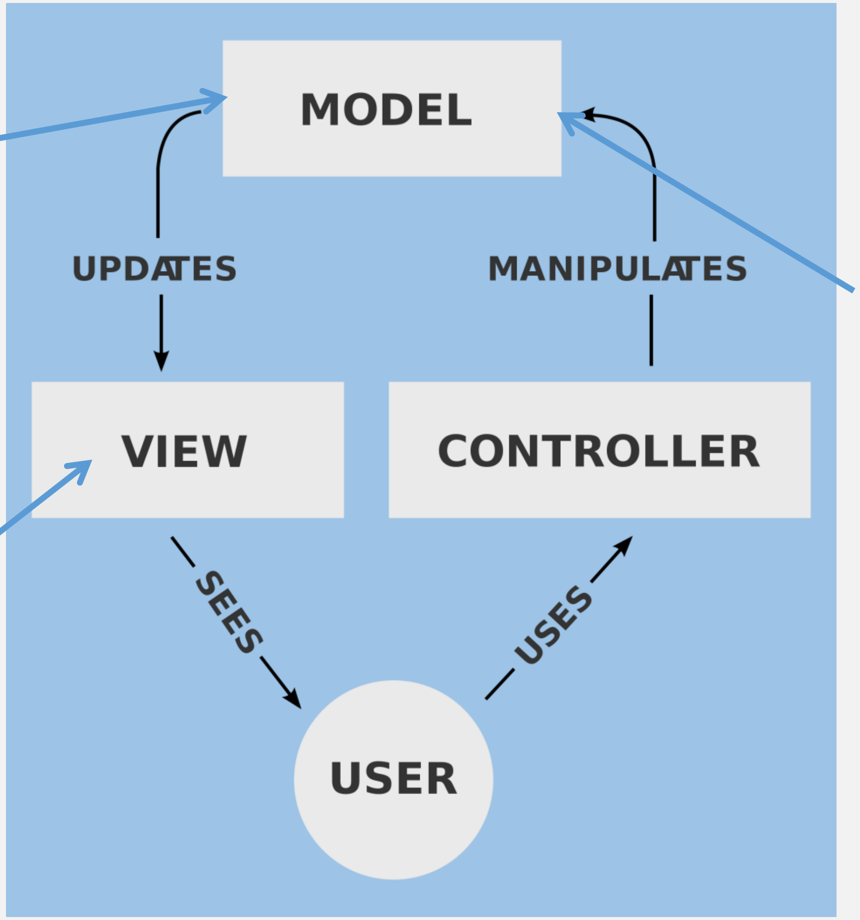
```
// simple_state_machine.h
// Simple finite state machine encapsulation
// Author: Michael S. Booth (mike@turtlerockstudios.com), November 2003

#ifndef _SIMPLE_STATE_MACHINE_H_
#define _SIMPLE_STATE_MACHINE_H_

//----------------------------------------------------------------------------------------
/**
 * Encapsulation of a finite-state-machine state
 */
template < typename T >
class SimpleState
```

Model-view-controller (MVC) is a FSM

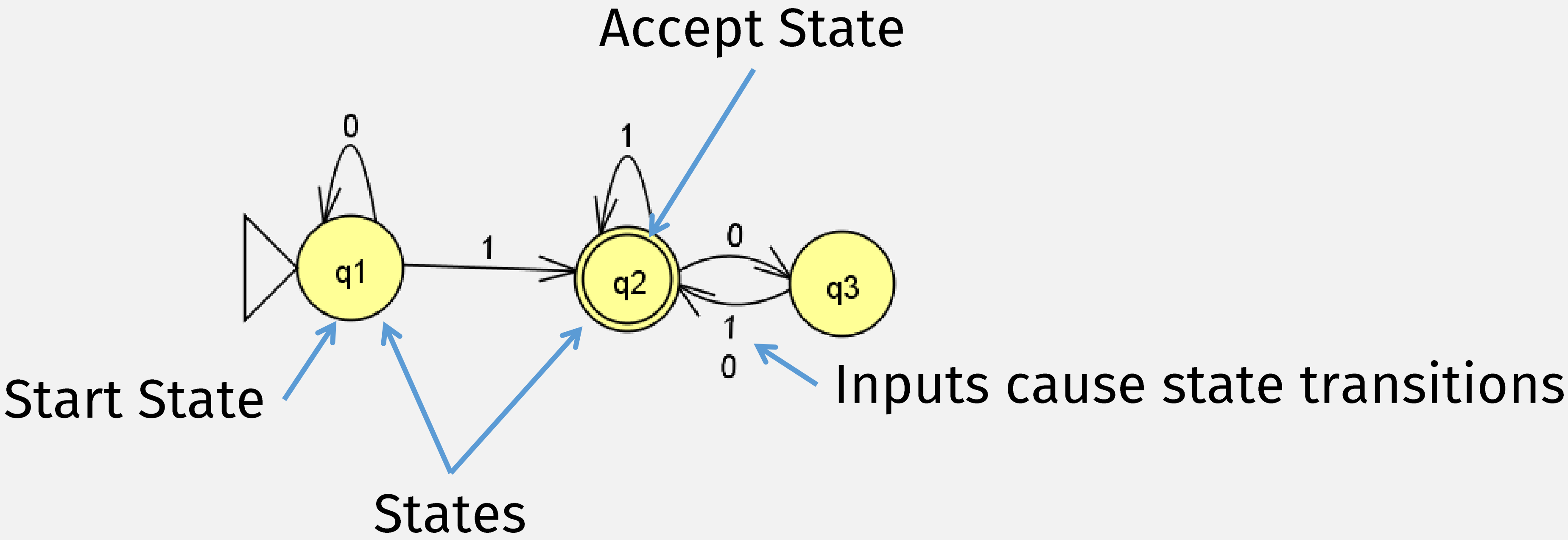
States



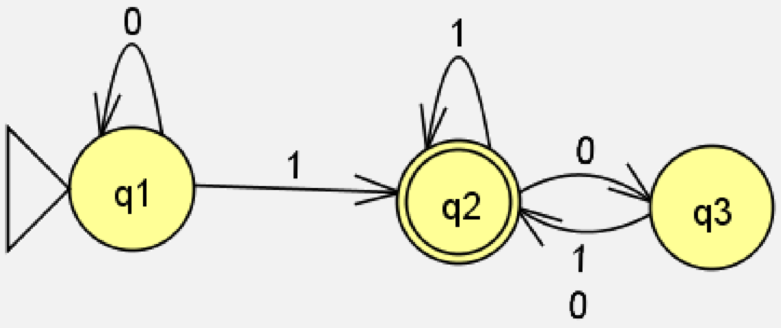
Inputs change states

View Draw states

Finite Automata in this class: state diagram



JFLAP demo: “Running” an FSM “Program”

- FSM: 

- Program: “1101”

Finite Automata: The Formal Definition

DEFINITION 1.5

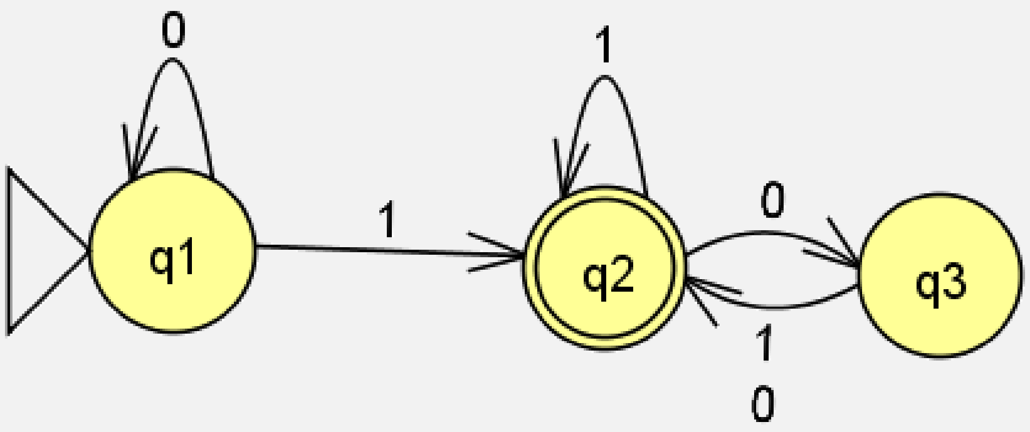
A ***finite automaton*** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the ***states***,
2. Σ is a finite set called the ***alphabet***,
3. $\delta : Q \times \Sigma \longrightarrow Q$ is the ***transition function***,
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.

DEFINITION 1.5

A ***finite automaton*** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the ***states***,
2. Σ is a finite set called the ***alphabet***,
3. $\delta: Q \times \Sigma \longrightarrow Q$ is the ***transition function***,
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.



$M_1 = (Q, \Sigma, \delta, q_1, F)$, where

1. $Q = \{q_1, q_2, q_3\}$,
2. $\Sigma = \{0,1\}$,
3. δ is described as

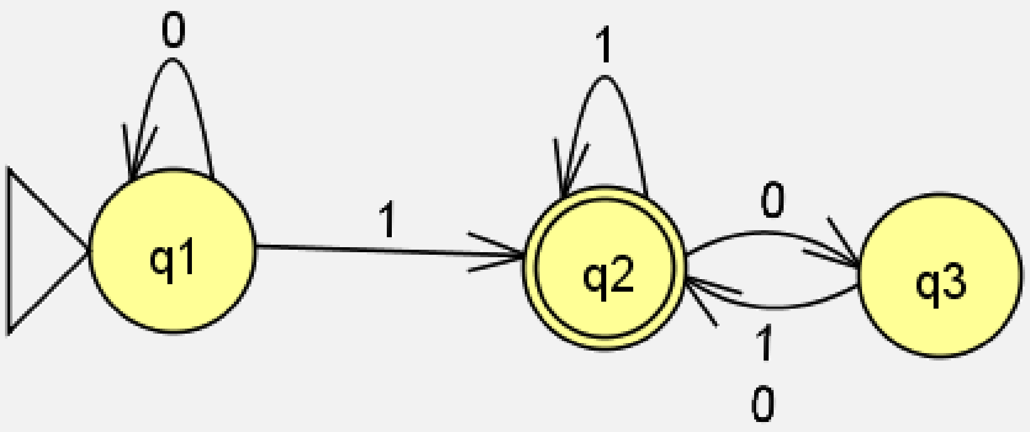
	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2,

4. q_1 is the start state, and
5. $F = \{q_2\}$.

DEFINITION 1.5

A ***finite automaton*** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the ***states***,
2. Σ is a finite set called the ***alphabet***,
3. $\delta : Q \times \Sigma \longrightarrow Q$ is the ***transition function***,
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.



$M_1 = (Q, \Sigma, \delta, q_1, F)$, where

1. $Q = \{q_1, q_2, q_3\}$,
2. $\Sigma = \{0,1\}$,
3. δ is described as

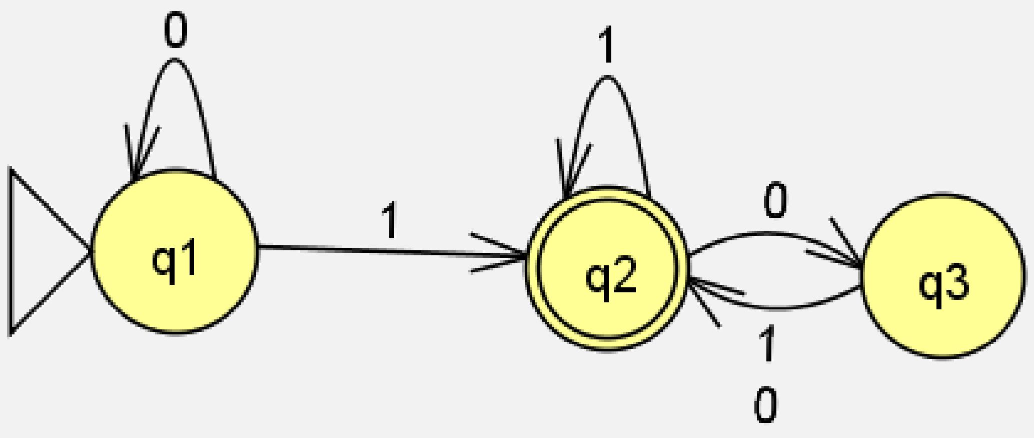
	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2,

4. q_1 is the start state, and
5. $F = \{q_2\}$.

DEFINITION 1.5

A ***finite automaton*** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the ***states***,
2. Σ is a finite set called the ***alphabet***,
3. $\delta \colon Q \times \Sigma \longrightarrow Q$ is the ***transition function***,
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.



$M_1 = (Q, \Sigma, \delta, q_1, F)$, where

1. $Q = \{q_1, q_2, q_3\}$,
2. $\Sigma = \{0,1\}$,
3. δ is described as

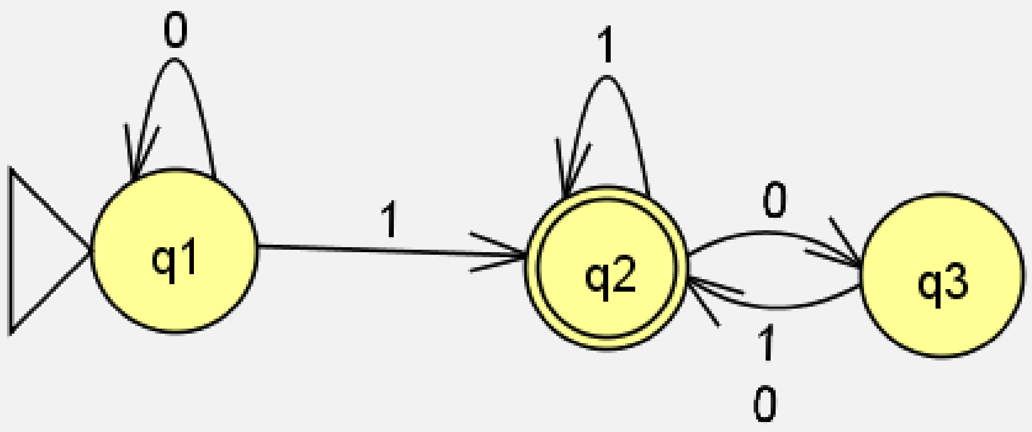
	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2,

4. q_1 is the start state, and
5. $F = \{q_2\}$.

DEFINITION 1.5

A ***finite automaton*** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the ***states***,
2. Σ is a finite set called the ***alphabet***,
3. $\delta : Q \times \Sigma \longrightarrow Q$ is the ***transition function***,
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.



$M_1 = (Q, \Sigma, \delta, q_1, F)$, where

1. $Q = \{q_1, q_2, q_3\}$,
2. $\Sigma = \{0,1\}$,
3. δ is described as

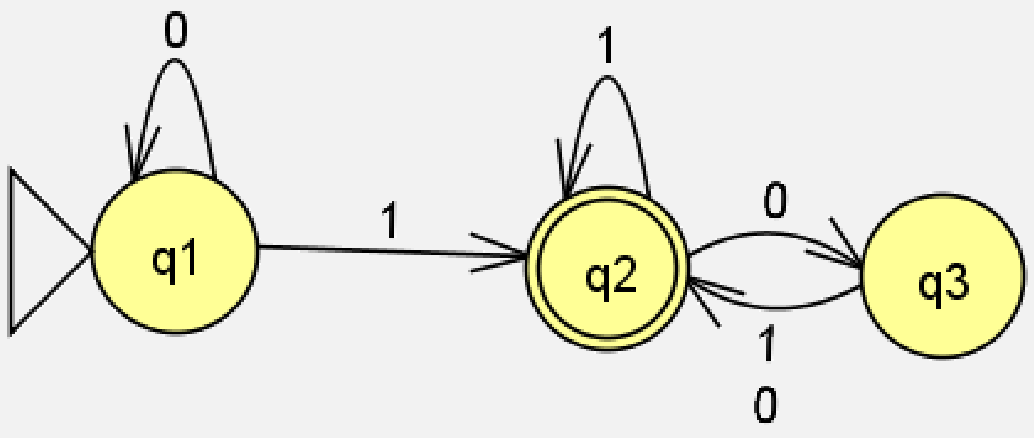
	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2,

4. q_1 is the start state, and
5. $F = \{q_2\}$.

DEFINITION 1.5

A ***finite automaton*** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the ***states***,
2. Σ is a finite set called the ***alphabet***,
3. $\delta : Q \times \Sigma \longrightarrow Q$ is the ***transition function***,
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.



$M_1 = (Q, \Sigma, \delta, q_1, F)$, where

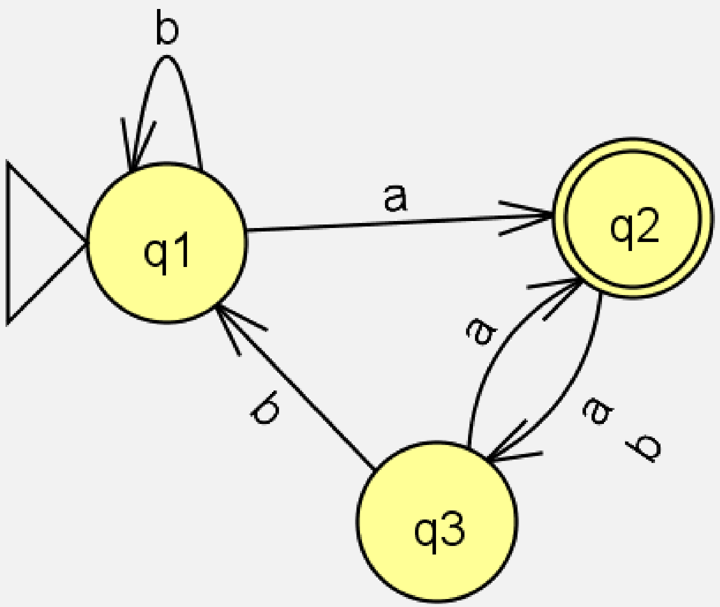
1. $Q = \{q_1, q_2, q_3\}$,
2. $\Sigma = \{0,1\}$,
3. δ is described as

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2,

4. q_1 is the start state, and
5. $F = \{q_2\}$.

In-class exercise

- Come up with a formal description of the following machine:



DEFINITION 1.5

A ***finite automaton*** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the ***states***,
2. Σ is a finite set called the ***alphabet***,
3. $\delta \colon Q \times \Sigma \longrightarrow Q$ is the ***transition function***,
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.

Terminology

- These are all equivalent:
  - Finite State Machine (FSM)
  - Finite Automaton, Automata, Automaton
  - State Machine
- They <u>generally</u> describe the class of machine**s** studied in Ch 1

- What I just introduced:
  - Deterministic Finite Automata (DFA)
- A specific kind of FSM, corresponding to Definition 1.5

- <u>**At this point** in the course</u> all terms on this slide are the same
  - But they wont be later

Math vs Its Code Representation

- In CS420 we use code to explore mathematical objects
- But it's important to understand the distinction
- E.g., a set is an <u>abstract</u> mathematical object
  - contains other math objects like: strings, nums, characters, and other sets!
- A set's (data) <u>representation</u> in code can take many forms:
  - e.g., a list, an array, a space-separated string
- This course teaches abstract mathematical concepts
  - It is up to you how to represent the math as code and data!

Math vs Representation, Examples

Abstract Math Concept	Possible Data Representation
Numbers	
Set	
Tuple (i.e., a small finite set)	
Function, i.e., a set of pairs	
Finite automata	

Math vs Representation, Examples

Abstract Math Concept	Possible Data Representation
Numbers	Int, BigInt, float, double
Set	
Tuple (i.e., a small finite set)	
Function, i.e., a set of pairs	
Finite automata	

Math vs Representation, Examples

Abstract Math Concept	Possible Data Representation
Numbers	Int, BigInt, float, double
Set	List, array, tree
Tuple (i.e., a small finite set)	
Function, i.e., a set of pairs	
Finite automata	

Math vs Representation, Examples

Abstract Math Concept	Possible Data Representation
Numbers	Int, BigInt, float, double
Set	List, array, tree
Tuple (i.e., a small finite set)	Struct, object, list
Function, i.e., a set of pairs	
Finite automata	

Math vs Representation, Examples

Abstract Math Concept	Possible Data Representation
Numbers	Int, BigInt, float, double
Set	List, array, tree
Tuple (i.e., a small finite set)	Struct, object, list
Function, i.e., a set of pairs	Function, dict, map, hash, tree
Finite automata	

Math vs Representation, Examples

Abstract Math Concept	Possible Data Representation
Numbers	Int, BigInt, float, double
Set	List, array, tree
Tuple (i.e., a small finite set)	Struct, object, list
Function, i.e., a set of pairs	Function, dict, map, hash, tree
Finite automata	XML str, <your choice here>

“Running a Program” on a Finite Automata

- Program = an input string of characters
- Start in “Start State”
- One char at a time, follow transition table to change states
- Result of running the program:
  - “Accept” the input if last state is an “Accept State”
  - “Reject” otherwise

Formal Definition of "Computation"

$M = (Q, \Sigma, \delta, q_0, F)$ a finite automaton
$w = w_1 w_2 \cdots w_n$ a string where each w_i is a member of the alphabet Σ.

M ***accepts*** w if a sequence of states $r_0, r_1, \ldots, r_n$ in Q exists with three conditions:

1. $r_0 = q_0$,
2. $\delta(r_i, w_{i+1}) = r_{i+1}$, for $i = 0, \ldots, n-1$, and
3. $r_n \in F$.

Condition 1 machine starts in the start state.
Condition 2 machine goes from state to state according to the transition function.
Condition 3 machine accepts its input if it ends up in an accept state.

Terminology

- M ***accepts*** w
- M ***recognizes language*** A

  if $A = \{w | \ M \text{ accepts } w\}$
- A language is called a ***regular language*** if some finite automaton recognizes it.

Proving that a language is regular

Kinds of Mathematical Proof

- Proof by construction
  - Construct the mathematical object in question
- Proof by contradiction

- Proof by induction

Proving that a language is regular

- Often requires creating a FSM

A language is called a ***regular language*** if some finite automaton recognizes it.

Designing Finite Automata

- States = the machine's **memory**!
  - Finite amount of memory: must be allocated in advance
  - Think about what information must be remembered.

- <u>Example</u>: machine accepts strings with even number of 0s
  - Two states: 1) seen even number of 0s, 2) seen odd number of 0s

- Input may only be read once

- Must decide accept/reject after that

In-class example

- Design machine M that recognizes: {w |w has exactly three 1's}

- Where Σ= {0, 1},

- Remember:

DEFINITION 1.5

A ***finite automaton*** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. Q is a finite set called the ***states***,
2. Σ is a finite set called the ***alphabet***,
3. $\delta: Q \times \Sigma \longrightarrow Q$ is the ***transition function***,
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.

Check-in Quiz 1

https://www.gradescope.com/courses/160337/assignments/650219

End of Class survey 9/14

https://forms.gle/pZqmX3urYRN5sn3t5