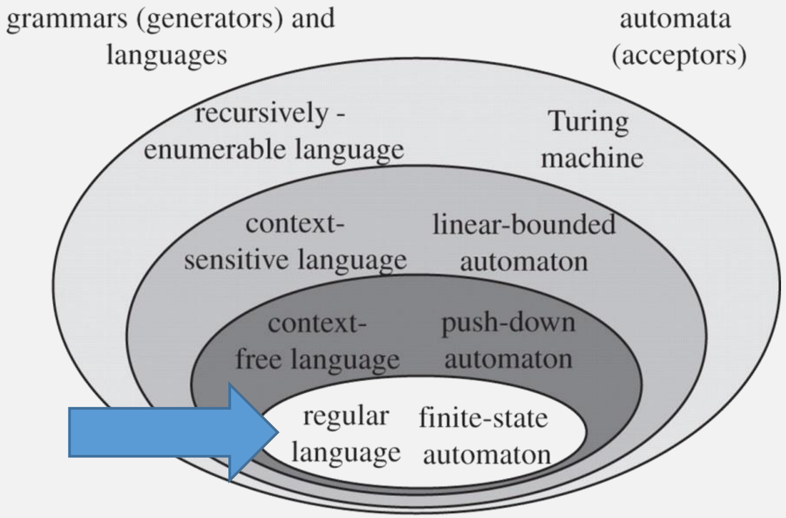
CS420
Regular Languages

Mon Feb 1, 2021



Logistics

- Piazza
  - Also, Discord
- TA: Welcome Anh!
  - Office hours Tues 2:30-4pm, Thurs 3-4:30pm
- Welcome new students
  - Watch old lectures, take old quizzes, do hw0
- HW0 deadline extended to Wed Feb 3 11:59pm EST
- Next HW gets ~1.5 weeks, BUT due in 2 parts:
  - HW1 (out) due: Sun Feb 7 11:59pm EST
  - HW2 (out later) due: Sun Feb 14 11:59pm EST

HW0, So Far: A Makefile

comment

Grader Preinstalled langs:
Python, Java, C, C++, JS, Racket

Targets
(see hw for names)

```
setup:  # install your language here (you can probably leave it blank)

run-hw0-stdio:
	racket hello.rkt # this line must start with a tab

run-hw0-alphabet:
	racket alphabet.rkt

run-hw0-powerset:
	racket powerset.rkt

run-hw0-xml:
	racket xml.rkt
```

Commands to run
(these files better exist)

HW0, So Far

- Read from stdin, write to stdout:
  - Python: `sys.stdin, print`
  - C++: `cin`, `cout`
  - Java: `System.in`, `System.out`
    - `Scanner scanner = new Scanner(System.in).readline` drops newlines!
- Power set of the empty set?
  - The power set of a set S is the set of all possible subsets of S
  - This includes empty set, and S itself!
  - $\mathcal{P}(\{\}) = \{\{\}\}$
- XML parsing:
  - Java: `javax.xml.parsers`
  - Python: `xml.etree.ElementTree: parse and findall`
  - `C++: pugixml`

HW1 Pre-game

- In CS 420 we primarily learn about abstract mathematical objects

- But we may use code as a way to explore these math objects

- So it's important to understand the distinction: math vs code

- E.g., a set is an abstract mathematical object
  - contains other math objects like: strings, nums, characters, and other sets!

- A set's (data) representation in code can take many forms:
  - e.g., a list, an array, a space-separated string (hw0)

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

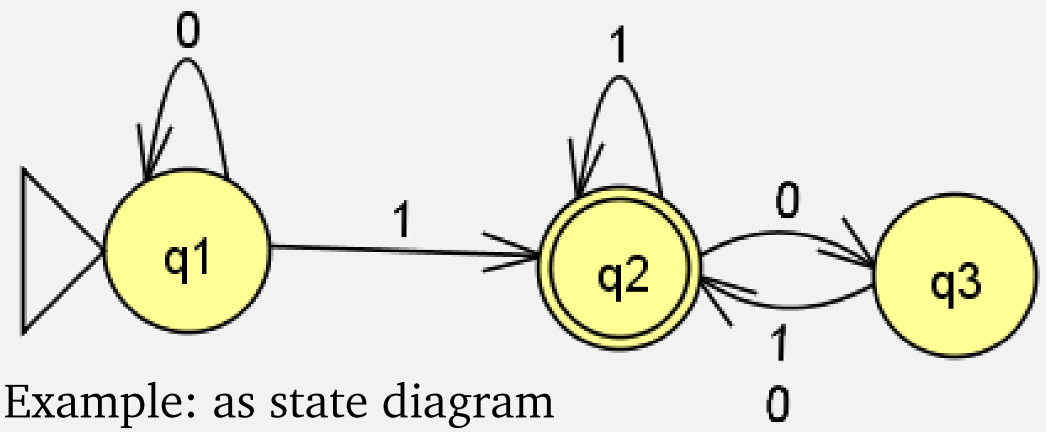
Abstract Math Concept	Possible Data Representation
Numbers	Int, BigInt, float, double
Set	List, array, tree
Tuple (i.e., a small finite set)	Struct, object, list
Function, i.e., a set of pairs	Function, dict, map, hash, tree
Finite automata	XML str, <your choice here>

Last Time:

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***.



Example: as state diagram

Example: as formal description

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

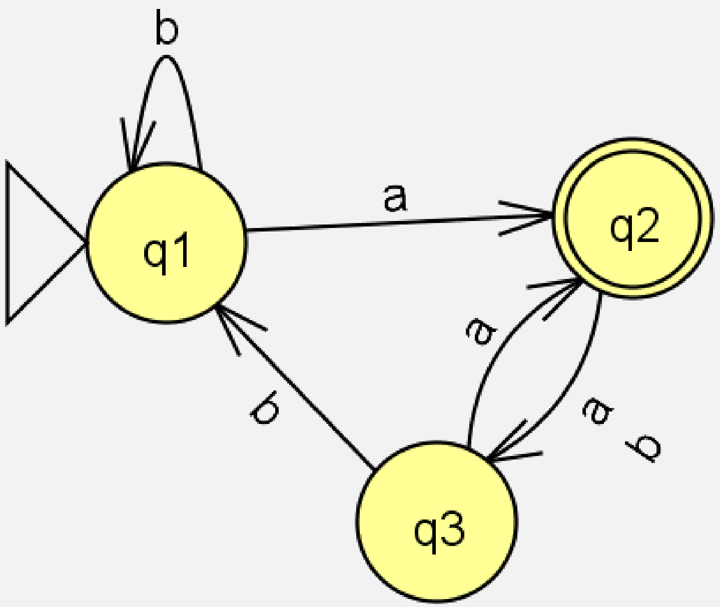
1. $Q = \{q_1, q_2, q_3\}$,
2. $\Sigma = \{0,1\}$,
3. δ is described as

	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 1

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

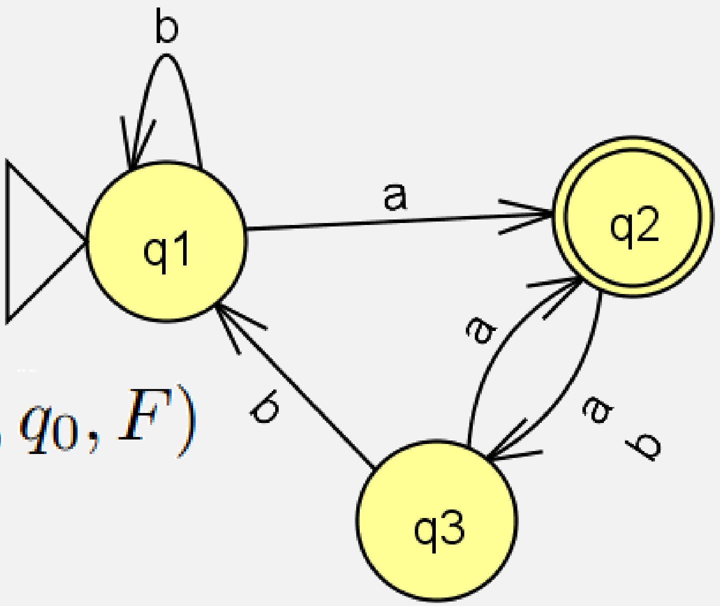


DEFINITION 1.5

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. $\delta : Q \times \Sigma \longrightarrow Q$ is the ***transition function***,
4. $q_0 \in Q$ is the ***start state***, and
5. $F \subseteq Q$ is the ***set of accept states***.

In-class exercise 1: solution



$$M = (Q, \Sigma, \delta, q_0, F)$$

- Q = {q1, q2, q3}
- Σ= {a, b}
- Delta
  - δ(q1,a) = q2
  - δ(q1,b) = q1
  - δ(q2,a) = q3
  - δ(q2,b) = q3
  - δ(q3,a) = q2
  - δ(q3,b) = q1
- q_0 = q1
- F = {q2}

Last Time: Computation, Formally

- A finite automata $M = (Q, \Sigma, \delta, q_0, F)$ is a computer
- We "run" on M an input string $w = w_1 w_2 \cdots w_n$, e.g. "1101"
- M **accepts** w if there is sequence of states $r_0, \ldots, r_n$ in Q where:
  - $r_0 = q_0$ **(start in "start" state)**
  - $\delta(r_i, w_{i+1}) = r_{i+1}, \text{ for } i = 0, \ldots, n-1$ **("next" states follow transition table)**
  - $r_n \in F$ **(last state is an "accept" state)**

A ***language*** is a set of strings.

Terminology

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

$$\text{if } A = \{w \mid M \text{ accepts } w\}$$

"the set of all …" "such that …"

A ***language*** is a set of strings.

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.

A *language* is a set of strings.

M ***recognizes language*** A if $A = \{w | M \text{ accepts } w\}$

A language, regular or not?

- <u>If given</u>: Finite Automata M
  - <u>We know</u>: the language recognized by M is a regular language

- <u>If given</u>: some Language A
  - Is A is a regular language?
    - Not necessarily
  - How do we determine, i.e., *prove*, that A is a regular language?

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

Kinds of Mathematical Proof

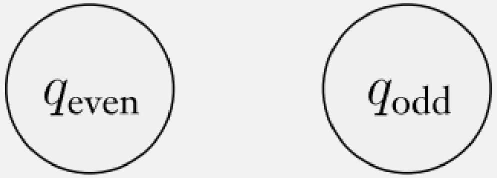
- Proof by construction
  - Construct the mathematical object in question
- Proof by contradiction
- Proof by induction

Designing Finite Automata: Tips

- Input may only be read once
- Must decide accept/reject after that
- States = the machine's **memory**!
  - Finite amount of memory: must be allocated in advance
  - Think about what information must be remembered.
- (For DFAs) Every state/symbol pair must have a transition
- Example: machine accepts strings with odd number of 0s

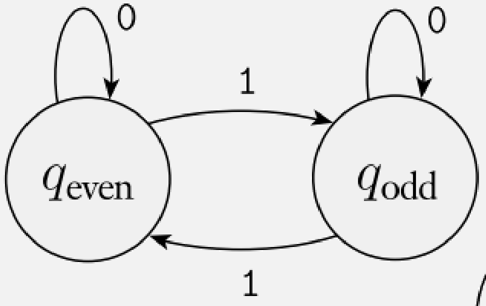
Design a DFA: accepts strs with odd # 0s

- <u>States</u>:
  - 2 states:
    - seen even 0s so far
    - seen odds 0s so far

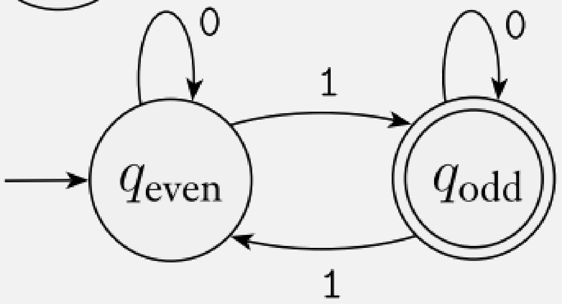


- <u>Alphabet</u>: 0 and 1

- <u>Transitions</u>:



- <u>Start</u> / <u>Accept</u> states?



In-class exercise 2

- Prove that this language is a regular language:
  - {w | w has exactly three 1's}
  - i.e., design a finite automata that recognizes it!

- Where $\Sigma = \{0, 1\}$,

- 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 2/1

See Gradescope