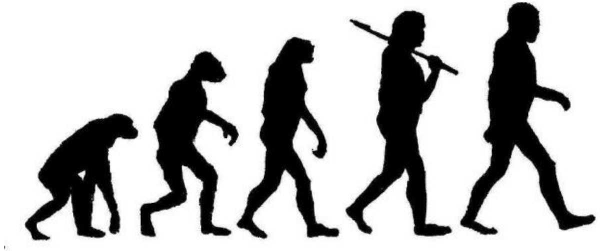
CSE 105
THEORY OF COMPUTATION

Fall 2016

http://cseweb.ucsd.edu/classes/fa16/cse105-abc/

Learning goals



Introductions

Clickers

LEDDEN Hall: CA

To change your remote frequency

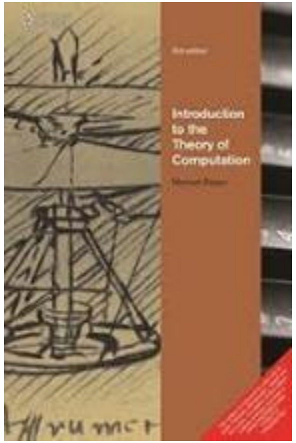
1. Press and hold power button until flashing
2. Enter two-letter code
3. Checkmark / green light indicates success

Why use clickers?

Have you used iClickers before?

A. Yes
B. No

Logistics



Homework: ~1 / week; 7 over the quarter (drop lowest score)

Participation: Class times (**iClicker** questions)
Discussion (5 options each **Friday**)
Haskell projects (4 over the quarter)

Exams: **Thursday** October 13
Thursday November 3
Thursday November 17 ** No makeup exams **
Final Exam **Thursday** December 8, 8:00am-11:00am

Gradescope: Homework submission, exam return, discussion attendance, interim reports
Piazza: announcements and Q&A. Contact instructors here!
Class podcast: podcast.ucsd.edu

Logistics

- Exams 65%		HW 30%			Participation + Preparation 5%
- Details on class website: http://cseweb.ucsd.edu/classes/fa16/cse105-abc
- Register iClickers on https://www1.iclicker.com/register-clicker/ by Friday October 7
- 7 HW assignments, can be done in groups of 1-3. **No late HW accepted.**
- Can change HW groups throughout quarter
- Drop lowest HW score
- Drop lowest midterm score if do better on final
- Can use note card for exams
- Need to earn at least 50% on final exam to pass the class
- Participation earned via either class participation, discussion quizzes, or haskell projects
- Drop lowest discussion score, two lowest class participation scores, lowest haskell project
- Credit for participation if answer 80% of clicker question in that day's class
- HW and exams answers evaluated not only on the correctness of your answers, but on your ability to effectively communicate your ideas and convince the reader of your conclusions through proofs and logical reasoning
- HW solutions on Piazza

How to succeed

- Prepare ahead of class
  - Read assigned sections, read homework questions
- Engage in class
  - Discuss questions with your neighbors, look for (counter)examples
  - Go over wrong choices too!
- Re-inforce after class
  - Briefly summarize what you learned
- Start homework early and **work in a group**
  - Tackle problems together: brainstorm, plan, and solve together
- Seek help and seek to help others, with integrity

Learning
How to Learn

About this class: Academic integrity

It's an integrity violation to…

- Click in for someone who is absent
- Sign discussion attendance sheet for someone who is absent
- Ask others to give you specific HW or quiz or test answers
- Share your answers on HW or quiz or test
- Work on HW with anyone other than your HW partners
- Search the internet or other resources not provided for the class for HW solutions
- Share answers or notes while taking an exam

This not a complete list … you are responsible for knowing and following the guidelines *Academic integrity violations will be taken seriously and reported immediately*

About this class: Academic integrity

You are working on a homework question with your group members and are stuck on a question. You run into a friend who solved the problem already and shows you her solution. You look at it, but put it away before continuing the group conversation. Is this acceptable?

A. Yes
B. No

About this class: Academic integrity

You’re working on a homework question and run across a definition you don’t understand. You Google the term and, ‘lo and behold, the first hit is a full solution to the homework question. You avoid reading the solution and close the browser. You keep working on the solution and hand in the assignment, without mentioning the Google search since you didn’t use the result. Is this acceptable?

A. Yes but please report!
B. No

About this class: Academic integrity

You're not sure if you are interpreting a homework problem correctly. You write a post on Piazza explaining your approach to answering it, and asking if this is the correct way to interpret the question. Is this acceptable?

A. Yes
B. No

private only!

Getting down to business

- This week's discussion: **Review** of prerequisites + DFA
  - Sets, functions, operations
  - Proofs (e.g. proof strategies, closure proofs)
  - Strings accepted / rejected by machines
- **Textbook** reference: Chapter 0, Section 1.1
- **Wednesday**: HW1 due (via Gradescope)

Today's learning goals Sipser Ch 0, 1.1

- Explain the difference between computability theory and complexity theory
- Relate finite automata to pattern tracking and matching
- Define key terms and operations associated to finite automata
  - Alphabet - Strings -Languages -Star, reversal
- Use and design a finite automaton via its
  - Formal definition - state diagram
- Identify the strings and languages accepted by a given finite automaton
- Design a finite automaton which accepts a given language

Questions about computers

Can they solve problems?

Can algorithms be provably correct?

Can algorithms be made more efficient?

Can algorithms always be found?

Questions about computers

Can they solve problems?

implementation: Java, C++, Python …

Can algorithms be provably correct?

CSE 20, 21, 101

Can algorithms be made more efficient?

Can algorithms always be found?

Questions? Problems?

- Scheduling
- Sorting
- Classifying
- Computing a value
- Predicting

Making a decision or computing a value based on some input

Questions about computers

Can they solve problems?

implementation: Java, C++, Python …

Can algorithms be provably correct?

CSE 20, 21, 101

Can algorithms be made more efficient?

Can algorithms always be found?

are there problems that no computer can solve?

Questions about computers

Can they solve problems?

implementation: Java, C++, Python …

Can algorithms be provably correct?

CSE 20, 21, 101

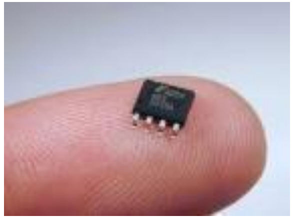
Can algorithms be made more efficient?

Complexity theory **[Last week of CSE 105]**

Can algorithms always be found?

Computability theory **[Last third of CSE 105]**

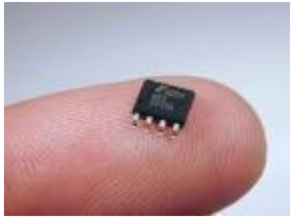
But what is an algorithm?



- Computers are everywhere in various forms
  - Mobile,
  - Embedded,
  - High-performance,
  - Hardware,
  - Cryptography,
  - Big data
  - DNA/RNA computing
  - Quantum

Different contexts call for different algorithms + different performance constraints

Any common thread?

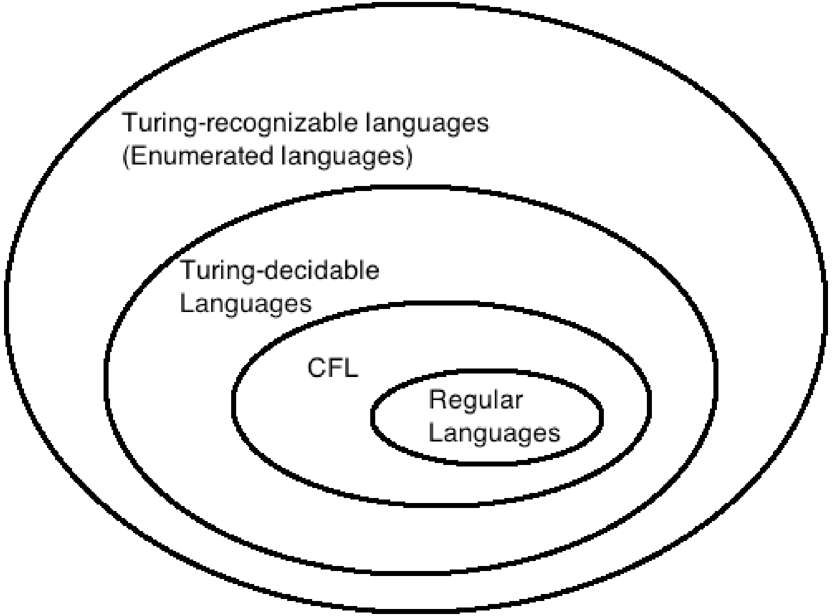


- Model of computation
  - Abstraction
  - Isolate common features of computation
  - May be removed from implementation

- Why?
  - Wide application
  - Can study limits

Models of computation

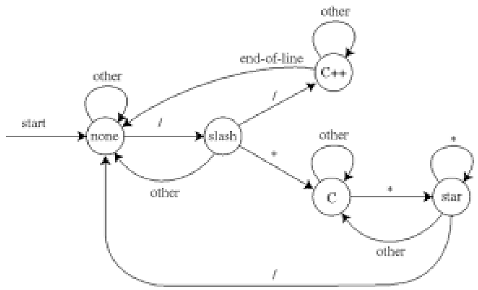
- Finite automata and their variants
- Context free grammars
- Turing machines

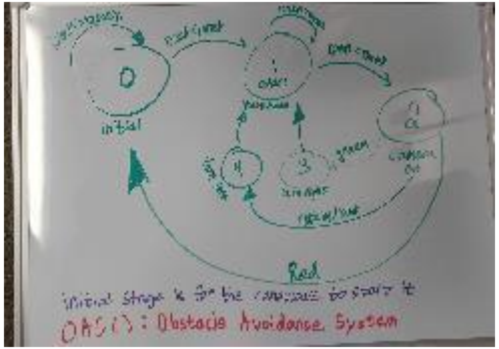


Automata

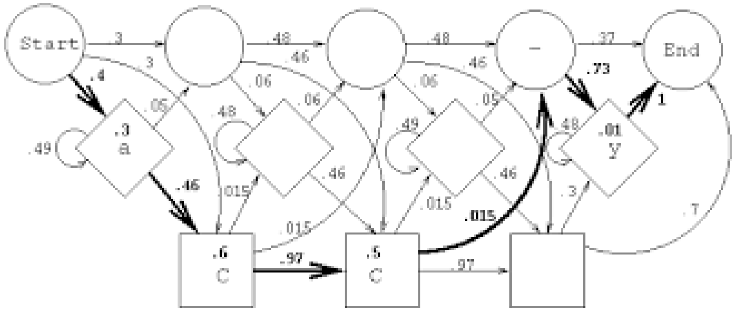
Code input as strings
Model memory using states

- Text processing
  - grep, regexp





- Hardware design
  - Moore machines, Mealy machines
- Controllers / Robots
  - SPIS!
- Statistical models



Example: subway turnstile

- A subway turnstile is locked until a token is entered, at which point it unlocks in response to a single push, after which it locks again.

When you approach turnstile, will it open?

How can we model this problem?

Example: subway turnstile

- A subway turnstile is locked until a token is entered, at which point it unlocks in response to a single push, after which it locks again.

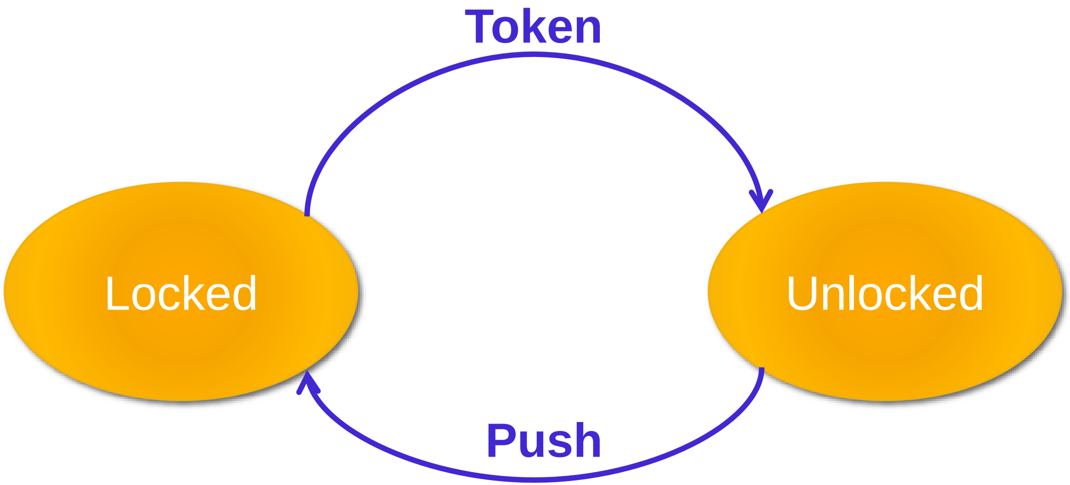
When you approach turnstile, will it open?

How can we model this problem?

Inputs: {token entered, push}

Example: subway turnstile

- A subway turnstile is locked until a token is entered, at which point it unlocks in response to a single push, after which it locks again.

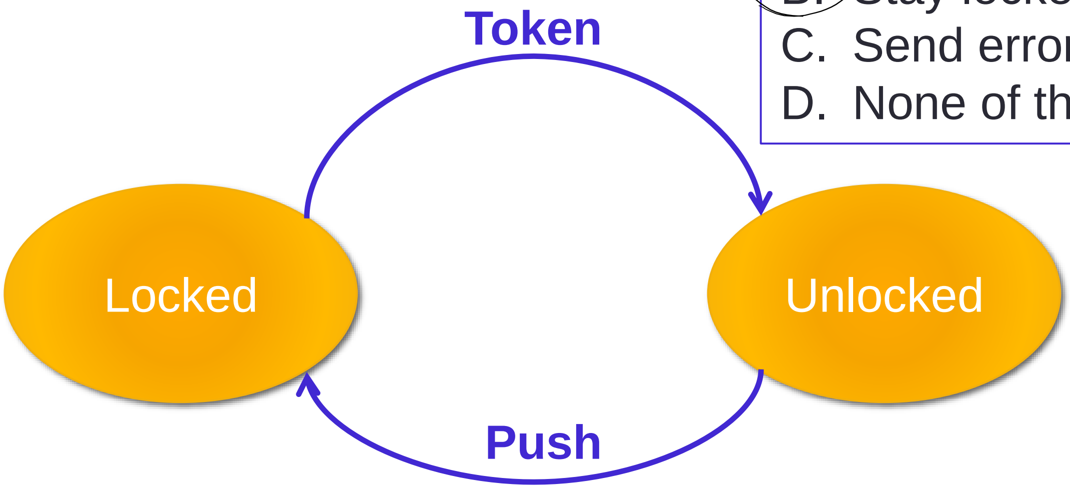


Example: subway turnstile

- A subway turnstile is locked unt[...] which point it unlocks in respon[...] which it locks again.

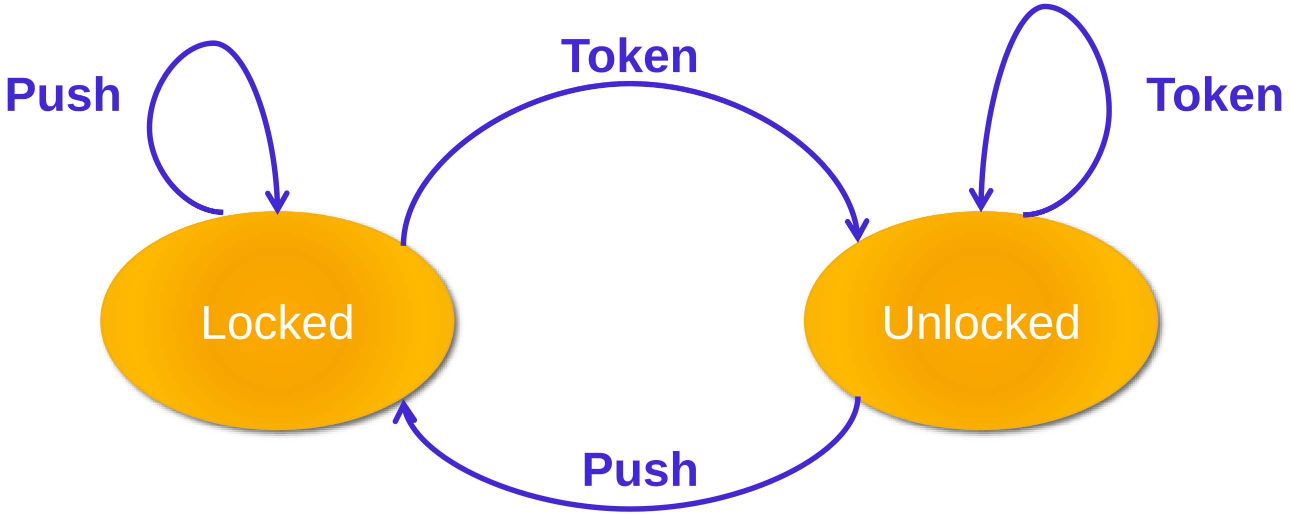
What happens if the turnstile is pushed while locked?

A. Transition to unlocked
B. Stay locked
C. Send error message
D. None of the above



Example: subway turnstile

- A subway turnstile is locked until a token is entered, at which point it unlocks in response to a single push, after which it locks again.

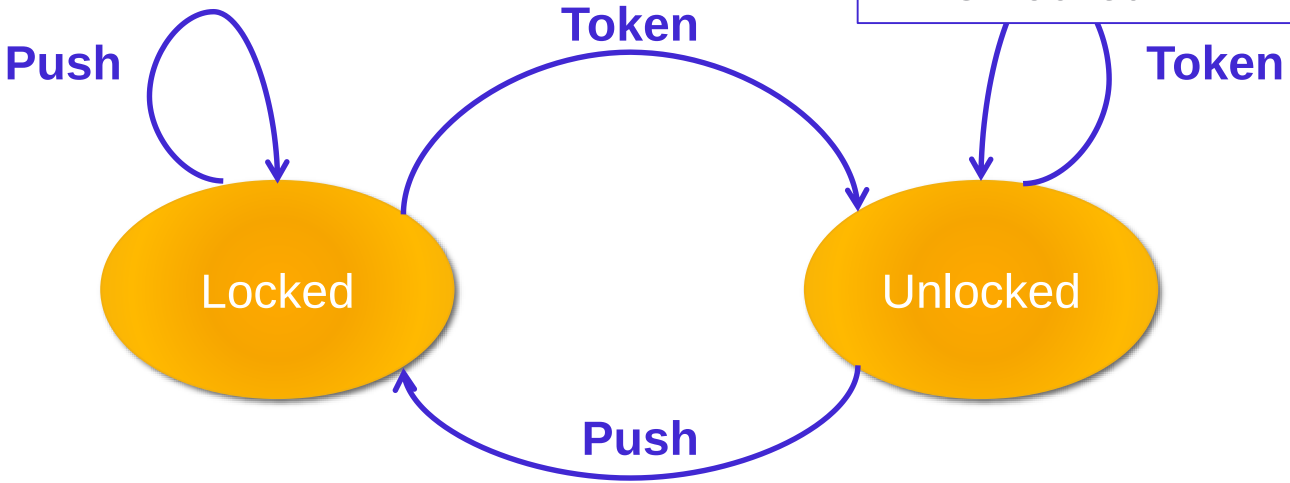


Example: subway turnstile

- A subway turnstile is locked unt
which point it unlocks in respon
which it locks again.

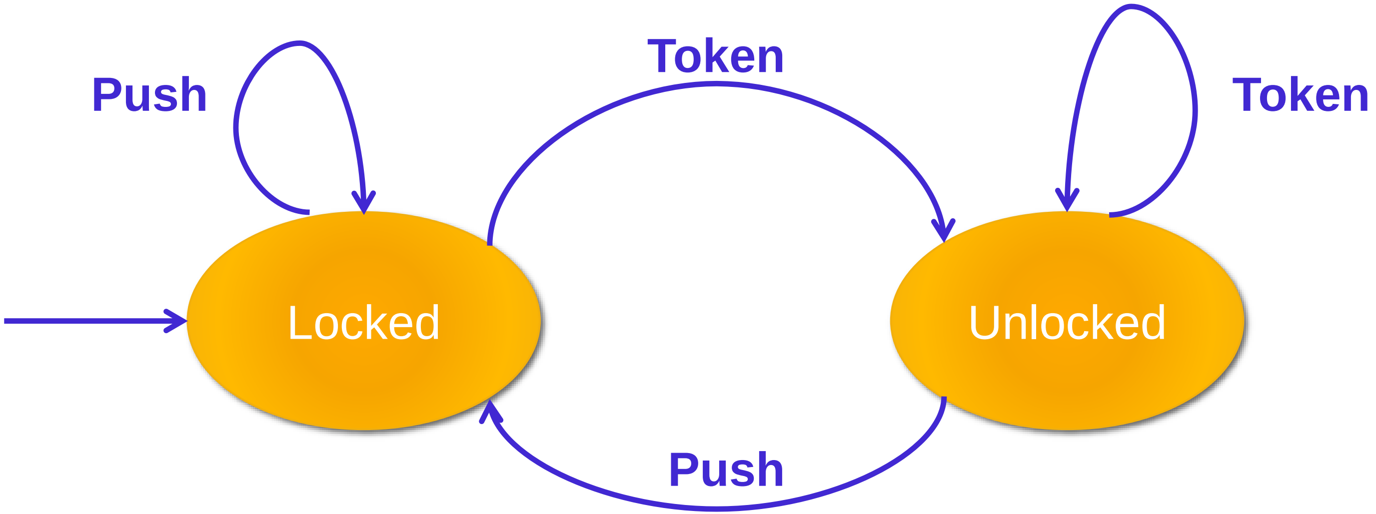
What's the initial state of the turnstile?

A. Locked
B. Unlocked



Example: subway turnstile

- A subway turnstile is locked until a token is entered, at which point it unlocks in response to a single push, after which it locks again.



Finite state machine

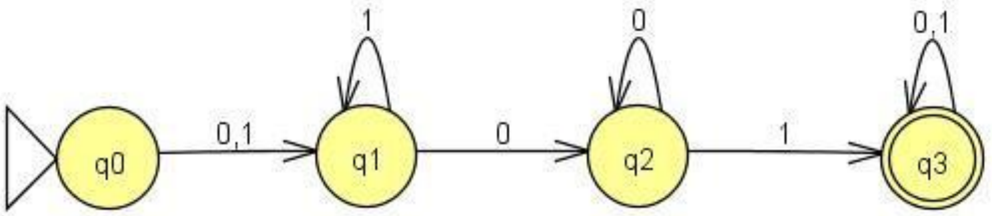
Capture patterns in behavior based on (limited) knowledge of what has happened in the past, and current input.

Abstract away details ….

- Input: an element from a finite set of symbols
- “Past”: string of input symbols

Finite automaton

- Input: finite **string** over a fixed **alphabet**
- Output: "accept" or "reject"



Start state
(triangle/arrow)

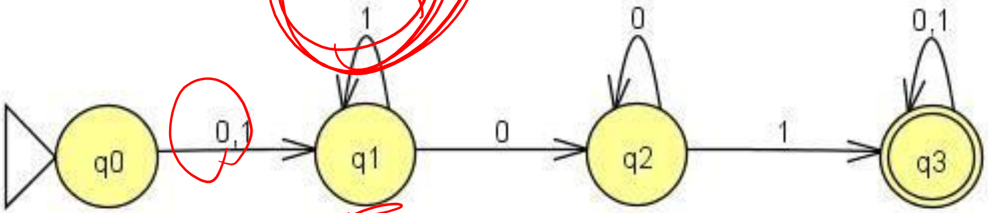
Accept state
(double circle)

Language of the machine is the set of strings it accepts

Finite automaton

String: 0 1 1 1
(1st) (2nd) 3rd 4th

- Input: finite string over a fixed alphabet
- Output: "accept" or "reject"



Start state
(triangle/arrow)

reject state

Accept state
(double circle)

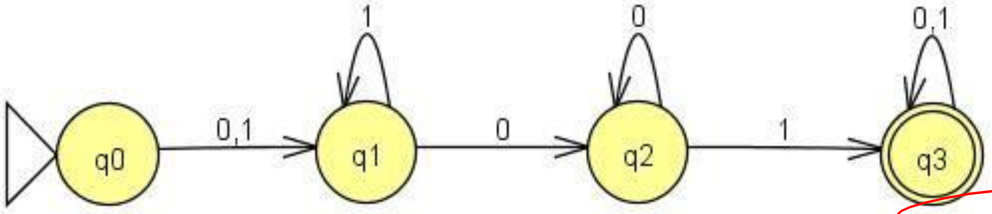
Computation of the machine on an input string?

q0 q1 q1 q1 q2

Each new symbol of input gives information about the string.

Finite automaton

- Input: finite string over a fixed alphabet
- Output: "accept" or "reject"



Start state
(triangle/arrow)

Accept state
(double circle)

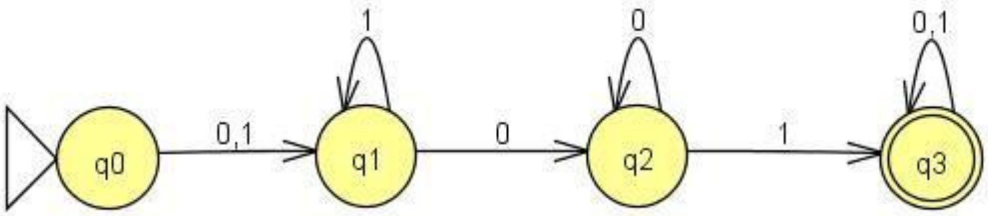
Computation:

q_0 q_1 q_2 q_2 q_3

Example input: 0001

Finite automaton

- Input: finite string over a fixed alphabet
- Output: "accept" or "reject"

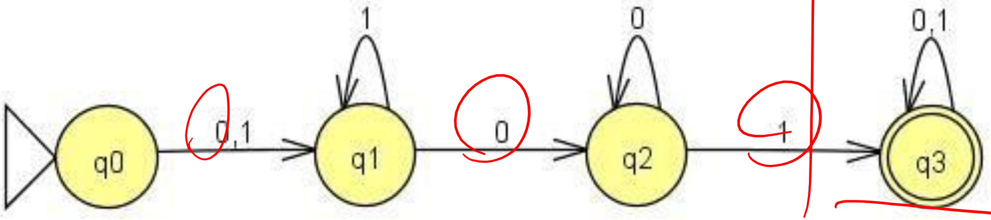


What is the sequence of states followed on input **110**?

A. q1, q1, q0
B. q1, q1, q2
C. q0, q1, q2, q3
D. q0, q1, q1, q2
E. None of the above

Finite automaton

- Input: finite string over a fixed alphabet
- Output: "accept" or "reject"



Does this DFA accept the string **001000** ?

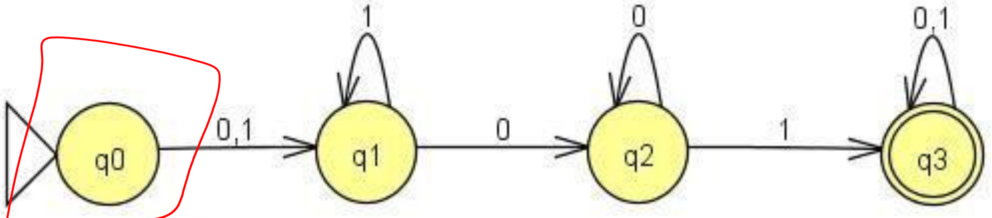
A. Yes
B. No
C. I don't know

q_0 q_1 q_2 q_3 q_3 ...

Deterministic

Finite automaton

- Input: finite string over a fixed alphabet
- Output: "accept" or "reject"



Does this DFA accept the **empty string** ?

A. Yes
B. No
C. I don't know

q_0

Finite automaton

Sipser p. 35 Def 1.5

deterministic

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. $\delta : Q \times \Sigma \to Q$ is the transition function
4. $q_0 \in Q$ is the start state (initial)
5. $F \subseteq Q$ is the set of accept states.

$\delta((q, b)) = q$

No circles and arrows, same information!

Finite automaton

Sipser p. 35 Def 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. $\delta : Q \times \Sigma \to Q$ is the transition function
4. $q_0 \in Q$ is the start state
5. $F \subseteq Q$ is the set of accept states.

Can there be more than one **start state** in a finite automaton?

A. Yes, because of line 4.
B. No, because of line 4.
C. I don't know

Finite automaton

Sipser p. 35 Def 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. $\delta : Q \times \Sigma \rightarrow Q$ is the transition function
4. $q_0 \in Q$ is the start state
5. $F \subseteq Q$ is the set of accept states.

set of accepted string

$F = \emptyset$

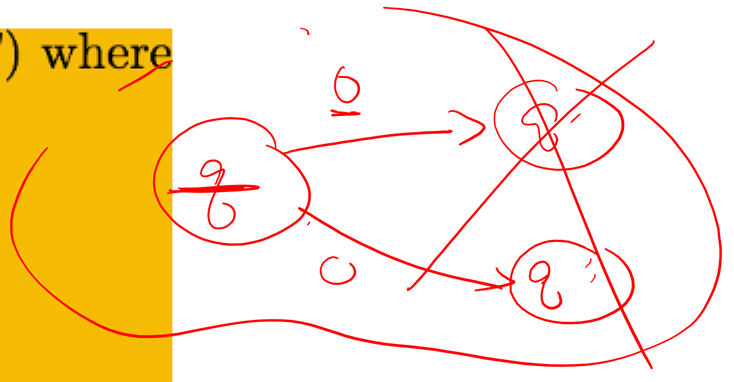
Can there be zero many **accept states**?

A. Yes, in which case the language is empty.
B. Yes, in which case the language is all strings.
C. No, because of line 5.
D. I don't know.

Finite automaton

Sipser p. 35 Def 1.5

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



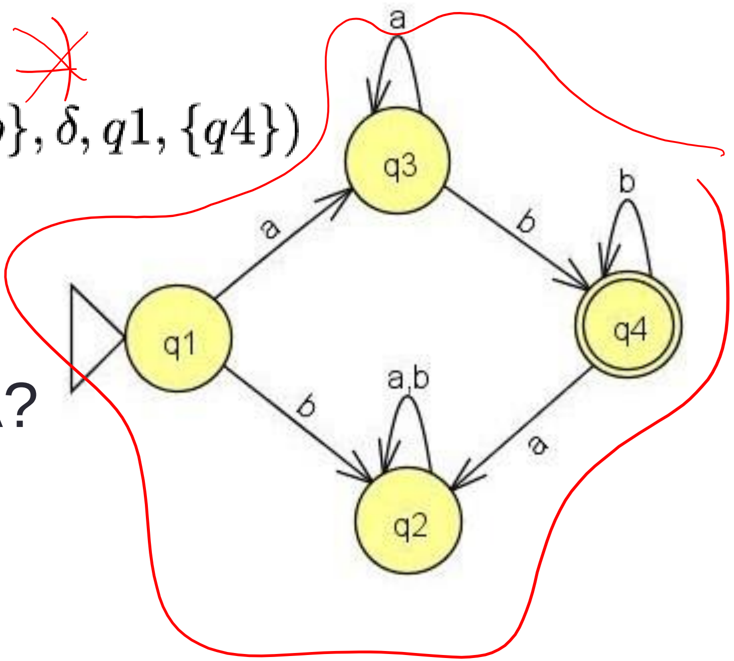
1. Q is a finite set called the states
2. Σ is a finite set called the alphabet
3. $\delta : Q \times \Sigma \to Q$ is the transition function
4. $q_0 \in Q$ is the start state
5. $F \subseteq Q$ is the set of accept states.

Can one state have two different **transitions** labelled with the same symbol going out of it?

A. Yes, because of 2.
B. Yes, because of 3.
C. No, because of 2.
D. No, because of 3.
E. I don't know.

An example

$(\{q1, q2, q3, q4\}, \{a, b\}, \delta, q1, \{q4\})$



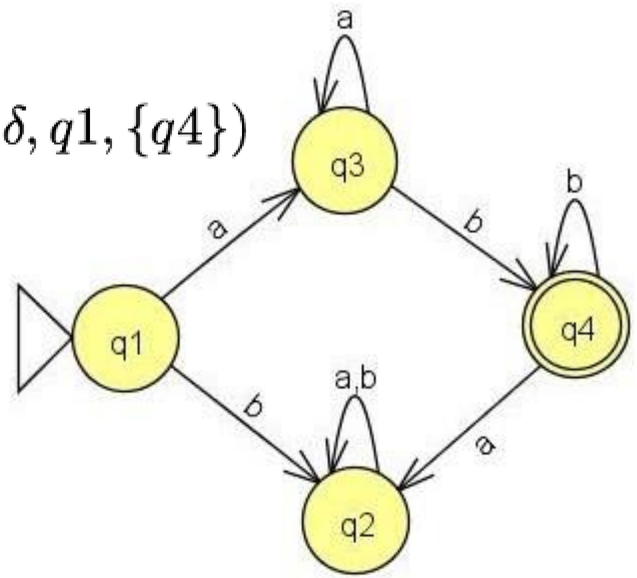
What's the best description of the language recognized by this DFA?

A. Start with b and ends with a or b
B. Starts with a and ends with a or b
C. a's followed by b's
D. More than one of the above
E. I don't know.

and using set notation?

An example

$$(\{q1, q2, q3, q4\}, \{a, b\}, \delta, q1, \{q4\})$$



This DFA recognizes
the language of all strings
of the form a's followed by b's

i.e. $\{ a^n b^k \mid n,k \geq 1\}$

Regular languages

Sipser p. 35 Def 1.5

- If A is the set of strings that DFA M recognizes (accepts)
  - We say A is the **language** of M
  - We write **L(M) = A**
  - We conclude that A is **regular** because…

A language is **regular** if there is some finite automaton that recognizes exactly it.

Regular languages: bounds?

Is there an infinite regular language?

A. No: all regular languages have to be finite.
B. Yes: all infinite sets of strings over an alphabet are regular.
C. Yes: some infinite sets of strings over each alphabet are regular and some are not.
D. I don't know.

Regular languages: bounds?

Is **every** finite language regular?

A. No: some finite languages are regular, and some are not.
B. No: there are no finite regular languages.
C. Yes: every finite language is regular.
D. I don't know.

Vocabulary review

From CSE20, etc. See Chapter 0

- { a,b,c,d,e } The set whose elements are a,b,c,d,e
- { a,b }* The set of finite strings over a,b
  - Includes empty string
  - Includes a, aa, aaa
  - Includes b, bb, bbb
  - Includes ab, ababab, aaaaaaabbb
  - Does **not** include infinite sequences of a's and b's
  - Has **infinitely many** different elements
- | ababab | = 6 The length of the string ababab is 6
- | { a,b,c,d,e } | = 5 The size of the set {a,b,c,d,e} is 5

For next time

- Start Homework 1 **due Wednesday**
  - Set up course tools: *Gradescope*, Piazza, JFLAP, haskell
  - Find group members
  - Read all the questions
  - Start working ☺
  - *Review CSE 20 / Math 109 / CSE 21 / Sipser Ch 0 as needed.*