CSE 105
THEORY OF COMPUTATION

"Winter" 2018

http://cseweb.ucsd.edu/classes/wi18/cse105-ab/

Today's learning goals

Sipser Section 1.1

- Design an automaton that recognizes a given language.
- Specify each of the components in a formal definition of an automaton.
- Prove that an automaton recognizes a specific language.

Deterministic 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.

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

Deterministic finite automaton

Sipser p. 35 Def 1.5

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

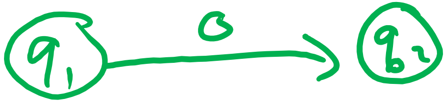
1. Q is a finite set called the states
2. Σ is a finite set called the alphabet
3. $\delta : Q \times \Sigma \to Q$ is the transition function
4. $q_0 \in Q$ is the start state
5. $F \subseteq Q$ is the set of accept states.

$Q = \{q_0, q_1, q_2\}$

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

$Q \times \Sigma = \{(q_0, 0), (q_0, 1), (q_1, 0), (q_1, 1), (q_2, 0), (q_2, 1)\}$

$\delta(q_1, 0) = q_2$



How many outgoing arrows from each state?

A. May be different number at each state.
B. Must be 2.
C. Must be $|Q|$.
D. Must be $|\Sigma|$
E. I don't know.

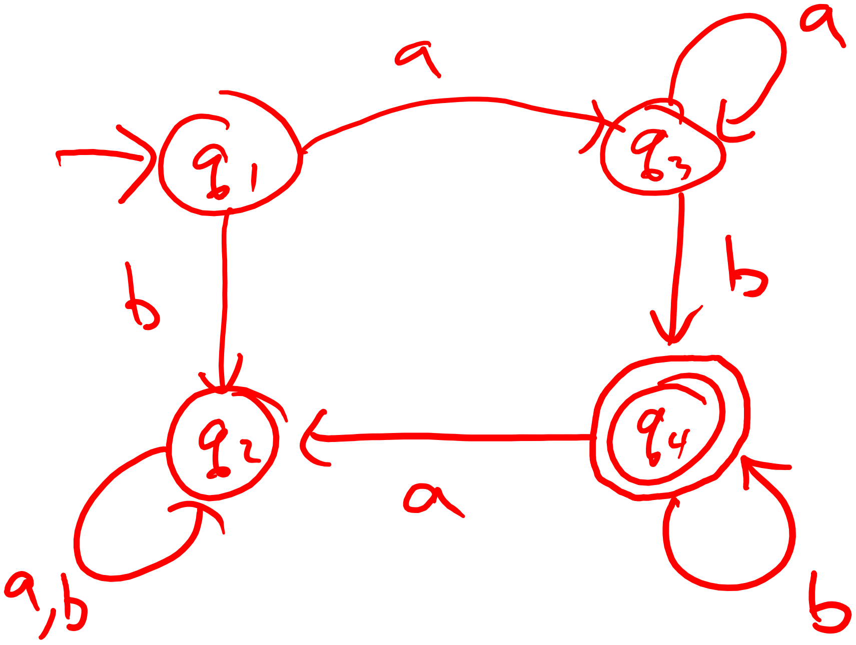
An example

Define M = $(\{q1, q2, q3, q4\}, \{a, b\}, \delta, q1, \{q4\})$ where the function δ is specified by its table of values:

Input in Q x Σ	Output in Q
(q1,a)	q3
(q2,a)	q2
(q3,a)	q3
(q4,a)	q2

Input in Q x Σ	Output in Q
(q1,b)	q2
(q2,b)	q2
(q3,b)	q4
(q4,b)	q4

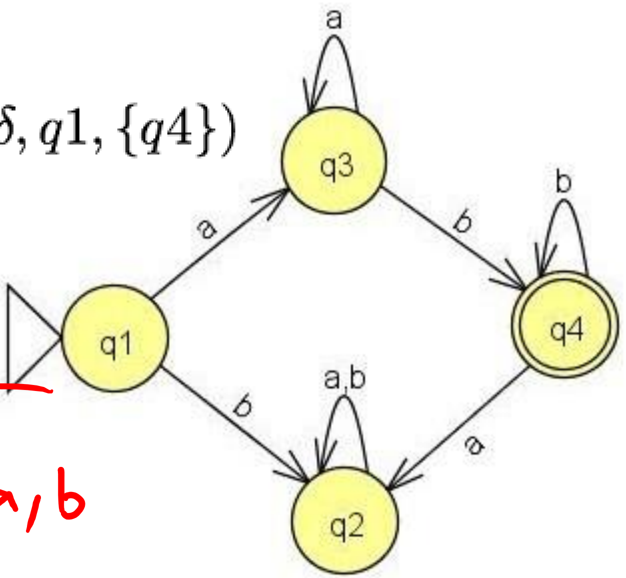
Draw the state diagram for the DFA with this formal definition.



q1
q2
q3
q4
a
b
a
a,b
b
a
b

An example

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



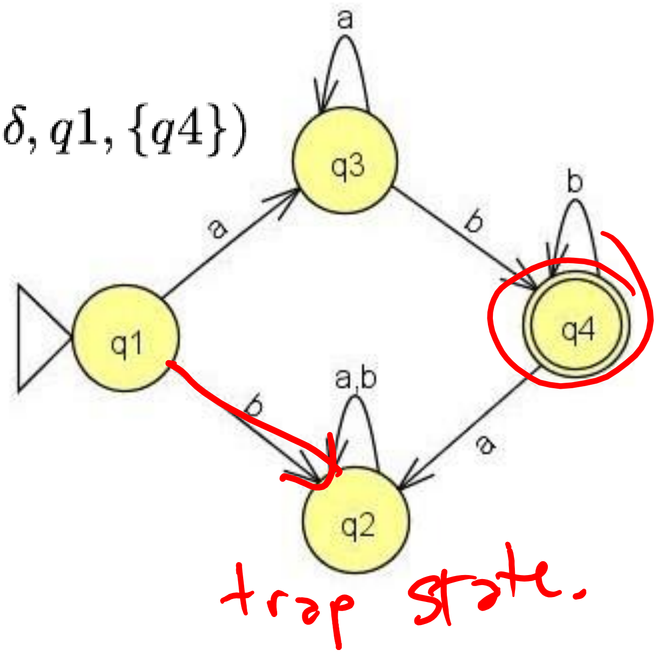
What's an example of a

- length 1 string accepted by this DFA?
- length 1 string rejected by this DFA? a, b

- length 2 string accepted by this DFA? ab
- length 2 string rejected by this DFA? aa, ba, bb

An example

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



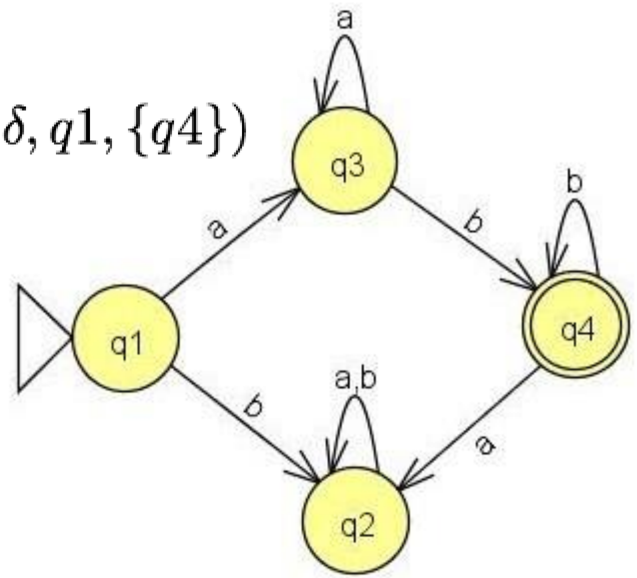
What's the best description of the language recognized by this DFA?

A. Starts 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

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



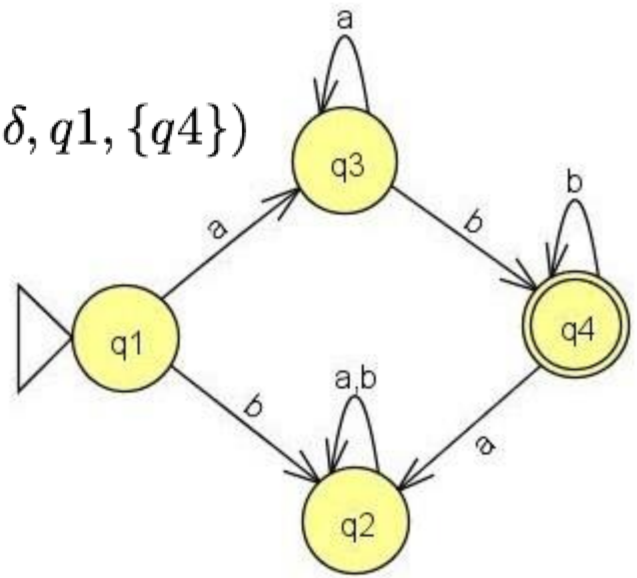
This DFA recognizes
the language of all strings
of the form a's followed by b's

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

An example

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

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



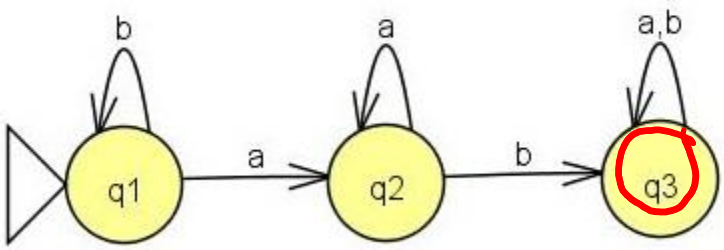
Is this the same as the language described by

A. a*b*
B. a(ba)*b
C. a* U b*
D. (aaa)*
E. a(a)*b(b)*

Specifying an automaton

({q1,q2,q3}, {a,b}, δ, q1, ?)

set of accept states.



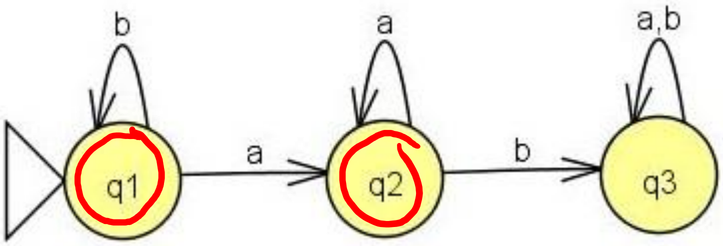
What state(s) should be in F so that the language of this machine is

{ w | ab is a substring of w}?

A. {q2}
B. {q3}
C. {q1,q2}
D. {q1,q3}
E. I don't know.

Specifying an automaton

({q1,q2,q3}, {a,b}, δ, q1, ?)



What state(s) should be in F so that the language of this machine is
{ w | b's never occur after a's in w}?

A. {q2}
B. {q3}
C. {q1,q2}
D. {q1,q3}
E. I don't know.

Building DFA

Typical questions

Define a DFA which recognizes the given language L.

or

Prove that the (given) language L is regular.

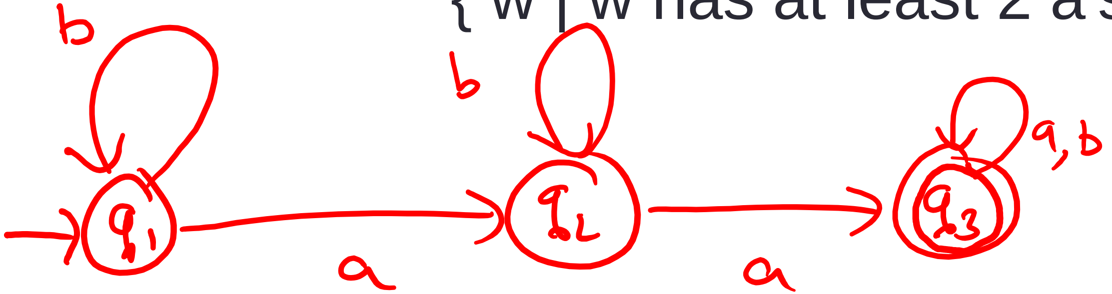
Building DFA

$\Sigma = \{a, b\}$

Example

Define a DFA which recognizes

{ w | w has at least 2 a's}



Building DFA

Example

Define a DFA which recognizes

{ w | w has at most 2 a's}

Building DFA

Remember

States are our only (computer) memory.

Design and pick states with specific roles / tasks in mind.

"Have not seen any of desired pattern yet"

"Trap state"

Justification?

To prove that the DFA we build, M, actually recognizes the language L

WTS L(M) = L

(1) Is every string accepted by M in L?

(2) Is every string from L accepted by M?

or contrapositive version: Is every string rejected by M not in L.

A useful (optional) bit of terminology

When is a string accepted by a DFA?

Computation of M on w: *where do we land when start at q_0 and read each symbol of w one-at-a-time?*

$$\delta^*(q, w) = \begin{cases} q & \text{if } w = \varepsilon \\ \text{say } w = aw' \\ \delta^*(\delta(q, a), w') \end{cases}$$

Recursively defined function

Regular languages

Sipser p. 35 Def 1.5

- DFA M over the alphabet Σ
  - For each string w over Σ, M either accepts w or rejects w
  - The **language recognized by M** is the set of strings M accepts
  - The **language of M** is the set of strings M accepts
  - **L(M)** = { w | w is a string over Σ and M accepts w}

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

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.

Building DFA

Remember

States are our only (computer) memory.

Design and pick states with specific roles / tasks in mind.

"Have not seen any of desired pattern yet"

"Trap state"

Building DFA

New strategy

Express L in terms of simpler languages – use them as building blocks.

Example

L = { w | w does not contain the substring baba }

= the complement of the set

{w | w contains the substring baba}

Building DFA

DFA recognizing {w | w contains the substring baba}

DFA recognizing {w | w doesn't contain the substring baba}

Complementation

Claim: If A is a regular language over $\{0,1\}^*$, then so is $\overline{A}$

aka "the class of regular languages is closed under complementation"

Complementation

Claim: If A is a regular language over $\{0,1\}^*$, then so is $\overline{A}$

aka "the class of regular languages is closed under complementation"

Proof: Let A be a regular language. Then there is a DFA $M=(Q,\Sigma,\delta,q_0,F)$ such that L(M) = A. We want to build a DFA whose language is $\overline{A}$. Define

M' = ?

Claim of Correctness L(M') = $\overline{A}$

Proof of claim…

Why closure proofs?

- General technique of proving a new language is regular
- Stretch the power of the model
- Puzzle!

Set operations

Input set(s) → OPERATION → Output set

Complementation

Kleene star

Concatenation

Union

Intersection

Set difference

The regular operations

Sipser Def 1.23 p. 44

For A, B languages over same alphabet, define:

$$A \cup B = \{x | x \in A \text{ or } x \in B\}$$

$$A \circ B = \{xy \mid x \in A \text{ and } y \in B\}$$

$$A^* = \{x_1 x_2 \ldots x_k | k \geq 0 \text{ and each } x_i \in A\}$$

These are operations on sets!

Union

Sipser Theorem 1.25 p. 45

Theorem: The class of regular languages over fixed alphabet Σ is closed under the union operation.

Proof:

What are we proving here?

A. For any set A, if A is regular then so is A U A.
B. For any sets A and B, if A U B is regular, then so is A.
C. For two DFAs M1 and M2, M1 U M2 is regular.
D. None of the above.
E. I don't know.

Union

Sipser Theorem 1.25 p. 45

Theorem: The class of regular languages over fixed alphabet Σ is closed under the union operation.

Proof: Let A1, A2 be any two regular languages over Σ.

WTS that A1 U A2 is regular.

Goal: build a machine that recognizes A1 U A2.

For next time

- Finish Individual Homework 0 **due Saturday**
- Review quiz 1 **due Sunday** (for credit)
- Read Individual Homework 1 **due Tuesday**

Pre class-reading for Wednesday:

Theorem 1.25, Theorem 1.26