CSCI 340: Computational Models

Finite Automata

Chapter 5

Department of Computer Science

Background: States and Determinism

Aside: Discussion on board games and **states**

- Where can pieces exist on a board?
- How do pieces move? (deterministic)
- When is the game over?

When we consider a “map” of all of the states and where they go, we create a **state diagram**

Finite Automaton

A **finite automaton** is a collection of three things:

1. A finite set of states, one of which is designated as the initial state, called the **start state**, and some (maybe none) of which are designated as **final states**.
2. An **alphabet** Σ of possible input letters
3. A finite set of **transitions** that tell for each state and for each letter of the input alphabet which state to go to next.

A Brief Example

1. Three states: x, y, z. Of which x is the starting state and z is the only final state
2. $\Sigma = \{\, a \quad b \,\}$
3. Transition Rules:
    1. From state x and input a , go to state y.
    2. From state x and input b , go to state z.
    3. From state y and input a , go to state x.
    4. From state y and input b , go to state z.
    5. From state z and input *any* , go to state z.

This defines a **language recognizer**. What language does it accept?

Definition of FA as Transition Table

	a	b
Start x	y	z
y	x	z
Final z	z	z

- *states* are listed along the left
- *alphabet characters* are listed along the top
- The "cell" at the intersection of a *state* and *character* indicate which *state* should be transitioned to

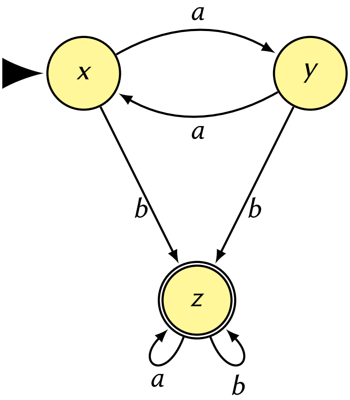
Mathematical Definition

1. A finite set of states $Q = \{q_0 \;\; q_1 \;\; q_2 \;\; \ldots\}$ of which q_0 is the start.
2. A subset of Q called the final states.
3. An alphabet $\Sigma = \{x_1 \;\; x_2 \;\; x_3 \;\; \ldots\}$.
4. A transition function mapping each state-letter pair with a state:

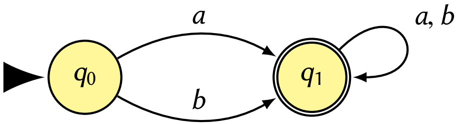
$$\delta(q_i, x_j) = x_k$$

NOTE: Every state has as many **outgoing edges** as there are letters in the alphabet. It is possible for a state to have no **incoming edges** or to have many.

Transition Diagram



Another Example

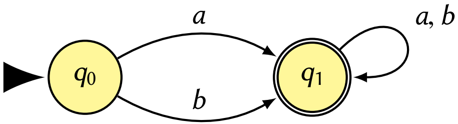


Question

What language does this FA accept?

Simplification

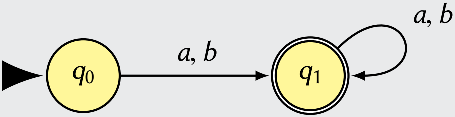
Another Example



Question

What language does this FA accept?

Simplification

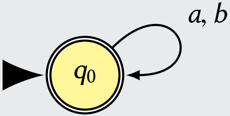


Examples

Finite Automaton Accepting Everything $\Sigma = \{a, b\}$

Examples

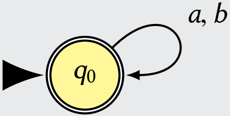
Finite Automaton Accepting Everything $\Sigma = \{a, b\}$



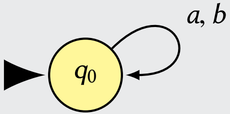
Finite Automaton Accepting Nothing $\Sigma = \{a, b\}$

Examples

Finite Automaton Accepting Everything $\Sigma = \{a, b\}$



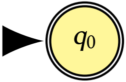
Finite Automaton Accepting Nothing $\Sigma = \{a, b\}$



Accepting an even-length string

Suppose we wanted to define an FA which accepts any string of an even length

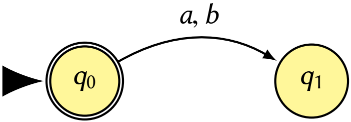
- How would we do this **programmatically**?
- How can we represent this with *states*?



Accepting an even-length string

Suppose we wanted to define an FA which accepts any string of an even length

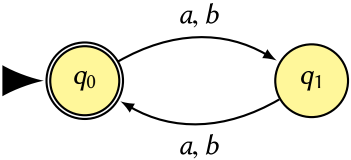
- How would we do this **programmatically**?
- How can we represent this with *states*?



Accepting an even-length string

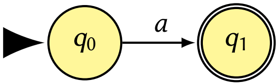
Suppose we wanted to define an FA which accepts any string of an even length

- How would we do this **programmatically**?
- How can we represent this with *states*?

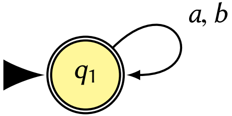


$\mathbf{a}(\mathbf{a} + \mathbf{b})^*$

$\mathbf{a}$:

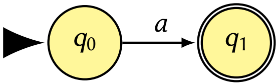


$(\mathbf{a} + \mathbf{b})^*$:

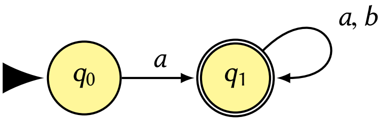


$\mathbf{a}(\mathbf{a} + \mathbf{b})^*$

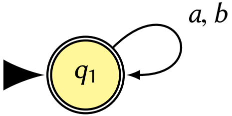
$\mathbf{a}$:



$\mathbf{a}(\mathbf{a} + \mathbf{b})^*$:



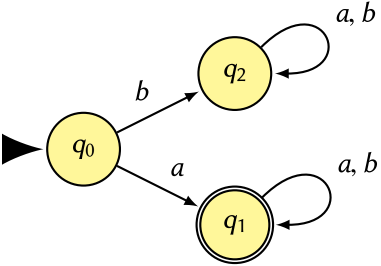
$(\mathbf{a} + \mathbf{b})^*$:

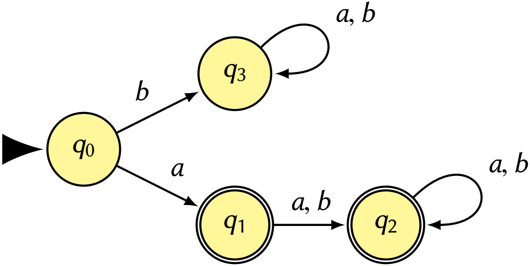


Question

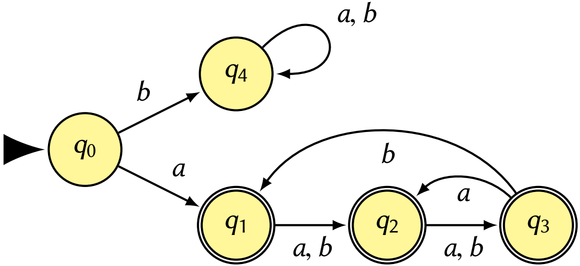
What if we encounter a b in q_0?

An extension on $\mathbf{a}(\mathbf{a}+\mathbf{b})^*$



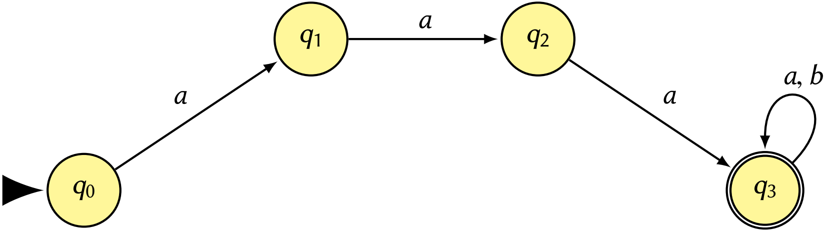


An extension on $\mathbf{a}(\mathbf{a}+\mathbf{b})^*$



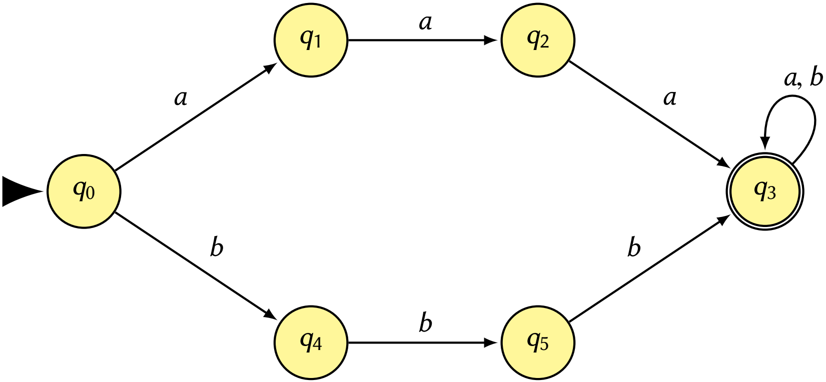
All of these are *equivalent*!

Matching strings with triple letters (*aaa* or *bbb*)



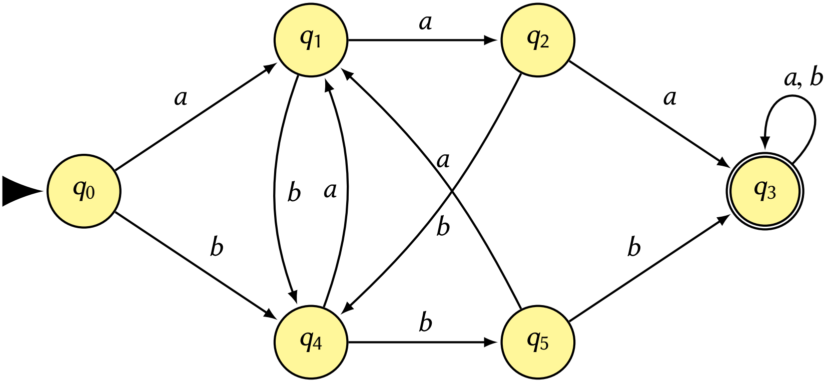
- Sequence of a's

Matching strings with triple letters (*aaa* or *bbb*)



- Sequence of a's or b's

Matching strings with triple letters (*aaa* or *bbb*)



- Proper state transitions when sequence broken

Chalkboard Examples

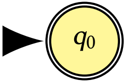
Construct FAs which accept the following:

- only the exact string **baa**
- all words not ending in b
- all words with an odd number of a's
- all words with different first and last letters
- all words with length divisible by 3

Revisiting EVEN-EVEN

Three cases:

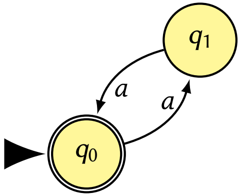
1. $\mathbf{aa}$
2. $\mathbf{bb}$
3. $(\mathbf{ab} + \mathbf{ba})(\mathbf{aa} + \mathbf{bb})^*(\mathbf{ab} + \mathbf{ba})$



Revisiting EVEN-EVEN

Three cases:

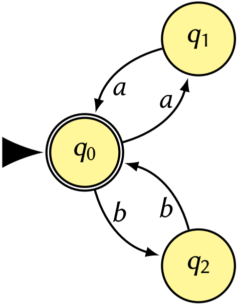
1. **aa** *handled here*
2. **bb**
3. $(\mathbf{ab} + \mathbf{ba})(\mathbf{aa} + \mathbf{bb})^*(\mathbf{ab} + \mathbf{ba})$



Revisiting EVEN-EVEN

Three cases:

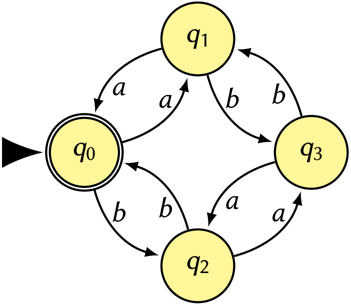
1. **aa**
2. **bb** *handled here*
3. $(\mathbf{ab} + \mathbf{ba})(\mathbf{aa} + \mathbf{bb})^*(\mathbf{ab} + \mathbf{ba})$



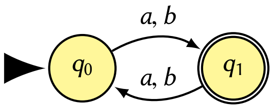
Revisiting EVEN-EVEN

Three cases:

1. **aa**
2. **bb**
3. (**ab** + **ba**)(**aa** + **bb**)*(**ab** + **ba**)
   handled here (q3 represents **ab** + **ba***)*



Homework 2b

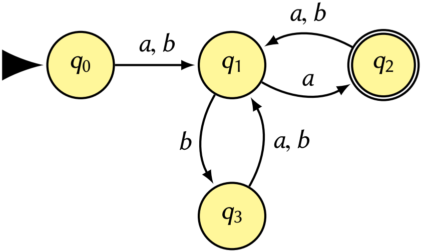
1. Build an FA that accepts only the language of all words with b as the second letter. Show both the picture and the transition table for this machine and find a regular expression for the language.
2. Find two FA's that satisfy these conditions: Between them they accept all words in $(\mathbf{a} + \mathbf{b})^*$, but there is no word accepted by both machines.
3. Describe the languages accepted by the following FA's:
   1. 

(continued on next page)

Homework 2b

3. Describe the languages accepted by the following FA's:

   ii.



   iii.