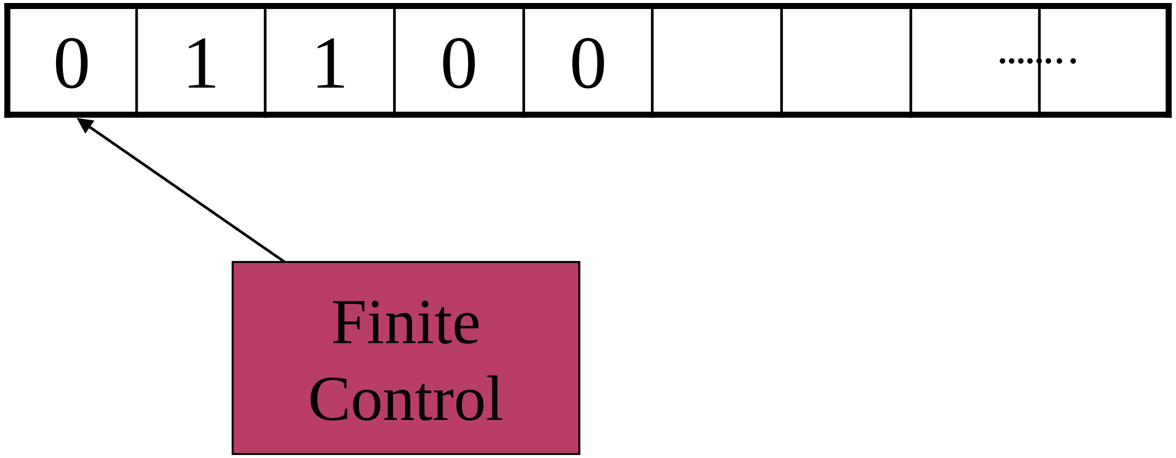
COURSE: THEORY OF AUTOMATA COMPUTATION

TOPICS TO BE COVERED

- Finite automata and regular sets
- Definition of deterministic finite automata
- String accepted by DFA

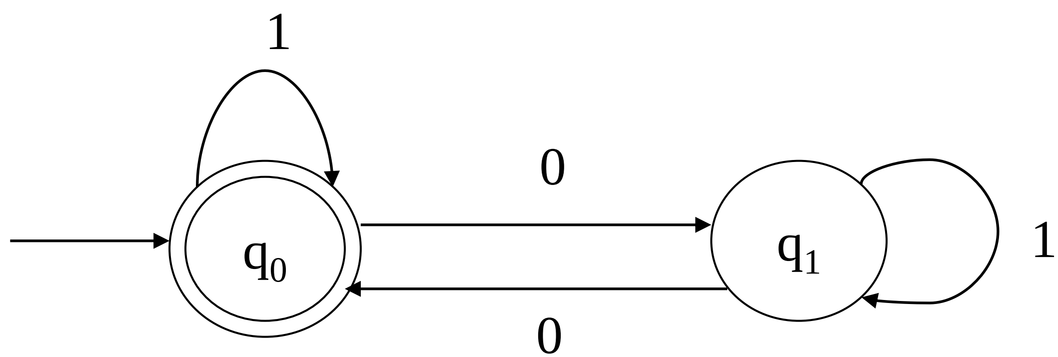
DETERMINISTIC FINITE STATE AUTOMATA (DFA)



- One-way, infinite tape, broken into cells
- One-way, read-only tape head.
- Finite control, I.e., a program, containing the position of the read head, current symbol being scanned, and the current “state.”
- A string is placed on the tape, read head is positioned at the left end, and the DFA will read the string one symbol at a time until all symbols have been read. The DFA will then either accept or reject.

- The finite control can be described by a <u>transition diagram</u>:

- Example #1:

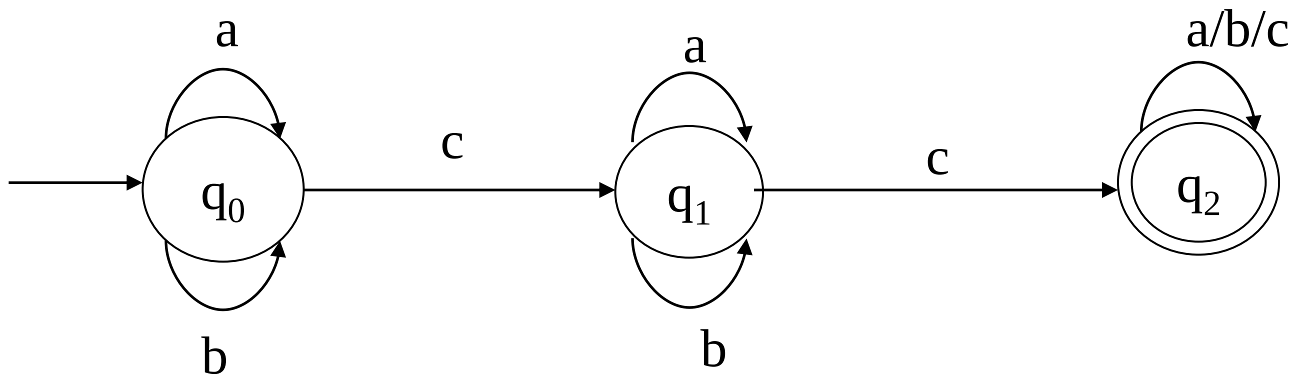


1 0 0 1 1

q_0 q_0 q_1 q_0 q_0 q_0

- One state is final/accepting, all others are rejecting.
- The above DFA accepts those strings that contain an even number of 0's

- Example #2:



	a	c	c	c	b	<u>accepted</u>
q_0	q_0	q_1	q_2	q_2	q_2	

	a	a	c	<u>rejected</u>
q_0	q_0	q_0	q_1	

- Accepts those strings that contain <u>at least</u> two c's

Inductive Proof (sketch):

Base: x a string with |x|=0. state will be q0 => rejected.

Inductive hypothesis: |x|=k, rejected -in state q0 (x must have 0 c),
OR, rejected – in state q1 (x must have 1 c),
OR, accepted – in state q2 (x already with 2 c's)

Inductive step: String xp, for p = a, b and c
q0 and, xa or xb: q0->q0 rejected, as should be (no c)
q0 and, xc: q0 -> q1 rejected, as should be (1 c)
q1 and xa or xb: q1 -> q1 rejected, …
q1 and xc: q1-> q2 accepted, as should be (2 c's now)
q2 and xa, or xb, or xc: q2 -> q2 accepted, (no change in c)

FORMAL DEFINITION OF A DFA

- A DFA is a five-tuple:

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

Q	A <u>finite</u> set of states
Σ	A <u>finite</u> input alphabet
q_0	The initial/starting state, q_0 is in Q
F	A set of final/accepting states, which is a subset of Q
δ	A transition function, which is a total function from Q x Σ to Q

δ: (Q x Σ) –> Q	δ is defined for any q in Q and s in Σ, and
δ(q,s) = q'	is equal to some state q' in Q, could be q'=q

Intuitively, δ(q,s) is the state entered by M after reading symbol s while in state q.

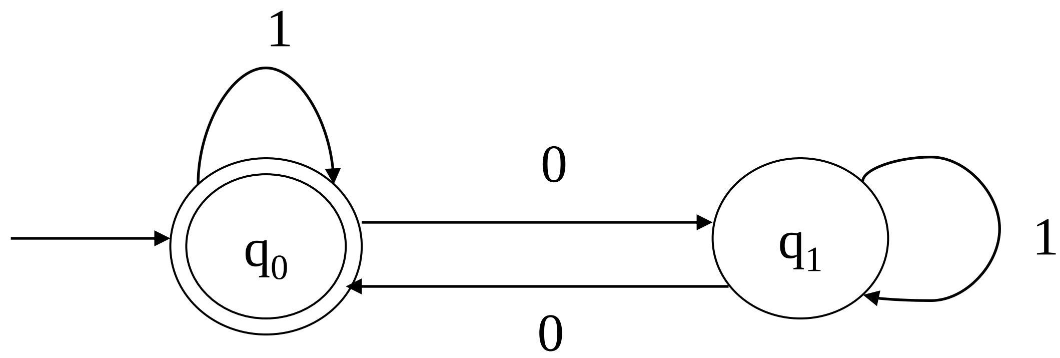
- For example #1:

$Q = \{q_0, q_1\}$

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

Start state is q_0

$F = \{q_0\}$

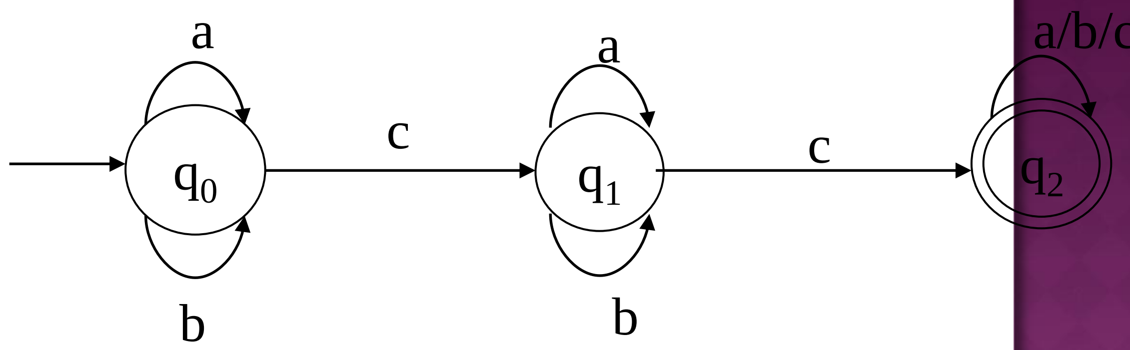


δ:

	0	1
q_0	q_1	q_0
q_1	q_0	q_1

- For example #2:

$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{a, b, c\}$
Start state is q_0
$F = \{q_2\}$



δ:

	a	b	c
q_0	q_0	q_0	q_1
q_1	q_1	q_1	q_2
q_2	q_2	q_2	q_2

- Since δ is a function, at each step M has exactly one option.
- It follows that for a given string, there is exactly one computation.

EXTENSION OF Δ TO STRINGS

$\delta^{\wedge} : (Q \times \Sigma^*) \rightarrow Q$

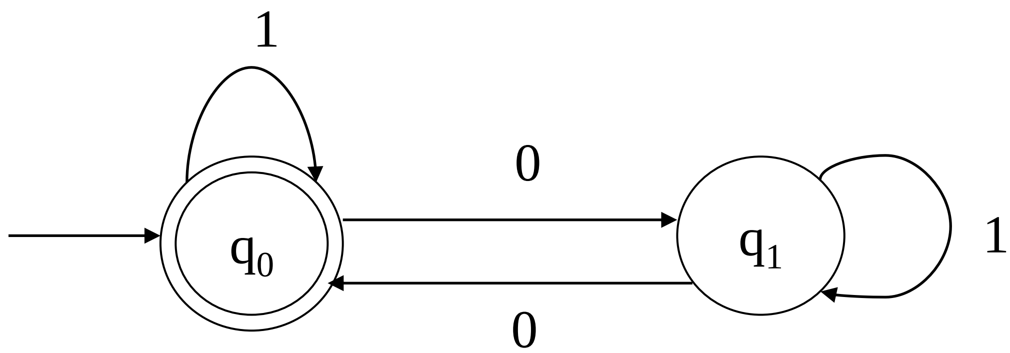
$\delta^{\wedge}(q,w)$ – The state entered after reading string w having started in state q.

Formally:

1) $\delta^{\wedge}(q, \varepsilon) = q$, and
2) For all w in Σ^* and a in Σ

$$\delta^{\wedge}(q,wa) = \delta(\delta^{\wedge}(q,w), a)$$

- Recall Example #1:



- What is $\hat{\delta}(q_0, 011)$? Informally, it is the state entered by M after processing 011 having started in state q_0.
- Formally:

$\hat{\delta}(q_0, 011)$ = $\delta(\hat{\delta}(q_0,01), 1)$ by rule #2
= $\delta(\delta(\hat{\delta}(q_0,0), 1), 1)$ by rule #2
= $\delta(\delta(\delta(\hat{\delta}(q_0, \lambda), 0), 1), 1)$ by rule #2
= $\delta(\delta(\delta(q_0,0), 1), 1)$ by rule #1
= $\delta(\delta(q_1, 1), 1)$ by definition of δ
= $\delta(q_1, 1)$ by definition of δ
= q_1 by definition of δ

- Is 011 accepted? No, since $\hat{\delta}(q_0, 011) = q_1$ is not a final state.

- Note that:

$\hat{\delta}(q,a) = \delta(\hat{\delta}(q, \varepsilon), a)$ by definition of $\hat{\delta}$, rule #2

$= \delta(q, a)$ by definition of $\hat{\delta}$, rule #1

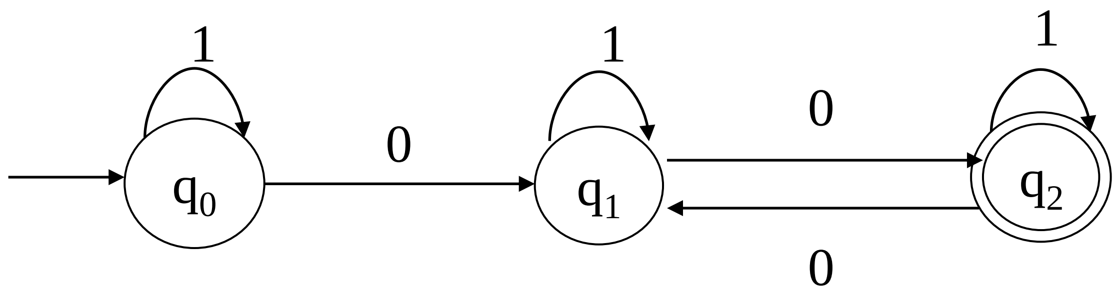
- Therefore:

$\hat{\delta}(q, a_1a_2 \ldots a_n) = \delta(\delta(\ldots\delta(\delta(q, a1), a2)\ldots), a_n)$

- Hence, we can use δ in place of $\hat{\delta}$:

$\hat{\delta}(q, a_1a_2 \ldots a_n) = \delta(q, a_1a_2 \ldots a_n)$

- Recall Example #2:



- What is $\delta(q_0, 011)$? Informally, it is the state entered by M after processing 011 having started in state q_0.
- Formally:

$\delta(q_0, 011) = \delta(\delta(q_0,01), 1)$ by rule #2

$= \delta(\delta(\delta(q_0,0), 1), 1)$ by rule #2

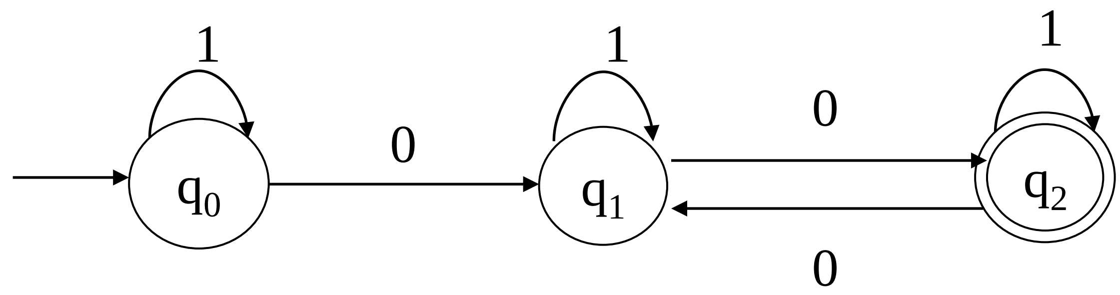
$= \delta(\delta(q_1, 1), 1)$ by definition of δ

$= \delta(q_1, 1)$ by definition of δ

$= q_1$ by definition of δ

- Is 011 accepted? No, since $\delta(q_0, 011) = q_1$ is not a final state.

- Recall Example #2:



- What is $\delta(q_1, 10)$?

$\delta(q_1, 10)$	$= \delta\ (\delta(q_1,1), 0)$	by rule #2
	$= \delta\ (q_1, 0)$	by definition of δ
	$= q_2$	by definition of δ

- Is 10 accepted? No, since $\delta(q_0, 10) = q_1$ is not a final state. The fact that $\delta(q_1, 10) = q_2$ is irrelevant!

DEFINITIONS FOR DFAS

- Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA and let w be in Σ^*. Then w is *accepted* by M iff $\delta(q_0, w) = p$ for some state p in F.
- Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA. Then the *language accepted* by M is the set:

$$L(M) = \{w \mid w \text{ is in } \Sigma^* \text{ and } \delta(q_0, w) \text{ is in } F\}$$

- Another equivalent definition:

$$L(M) = \{w \mid w \text{ is in } \Sigma^* \text{ and } w \text{ is accepted by } M\}$$

- Let L be a language. Then L is a *regular language* iff there exists a DFA M such that $L = L(M)$.
- Let $M_1 = (Q_1, \Sigma_1, \delta_1, q_0, F_1)$ and $M_2 = (Q_2, \Sigma_2, \delta_2, p_0, F_2)$ be DFAs. Then M_1 and M_2 are *equivalent* iff $L(M_1) = L(M_2)$.

- Notes:
  - A DFA $M = (Q, \Sigma, \delta, q_0, F)$ partitions the set Σ^* into two sets: L(M) and Σ^* - L(M).
  - If L = L(M) then L is a subset of L(M) and L(M) is a subset of L.
  - Similarly, if $L(M_1) = L(M_2)$ then $L(M_1)$ is a subset of $L(M_2)$ and $L(M_2)$ is a subset of $L(M_1)$.
  - Some languages are regular, others are not. For example, if

    L_1 = {x | x is a string of 0's and 1's containing an even number of 1's} and

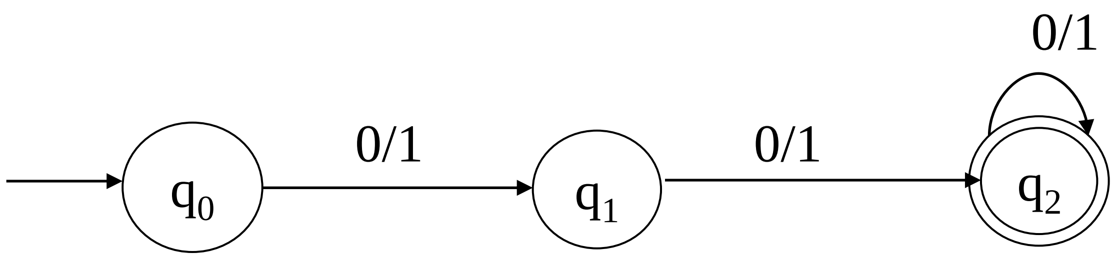
    $L_2 = \{x \mid x = 0^n 1^n \text{ for some } n >= 0\}$

    then L_1 is regular but L_2 is not.

- Questions:
  - How do we determine whether or not a given language is regular?
  - How could a program "simulate" a DFA?

- Give a DFA M such that:

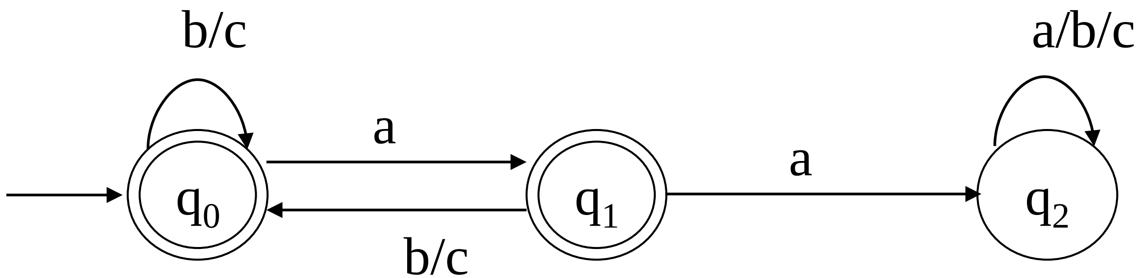
L(M) = {x | x is a string of 0's and 1's and |x| >= 2}



Prove this by induction

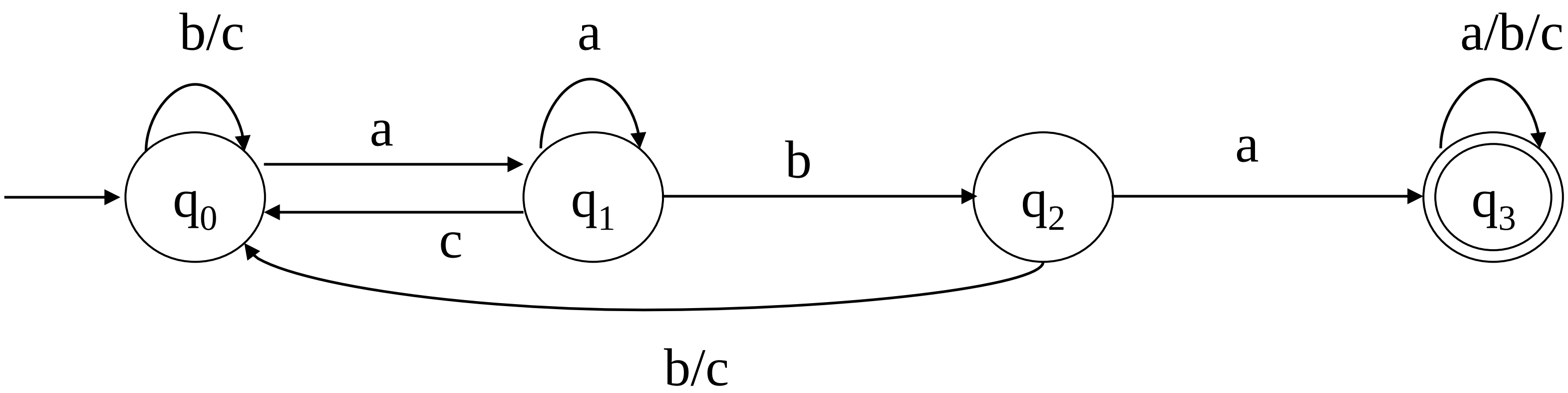
- Give a DFA M such that:

L(M) = {x | x is a string of (zero or more) a's, b's and c's such that x does not contain the substring *aa*}



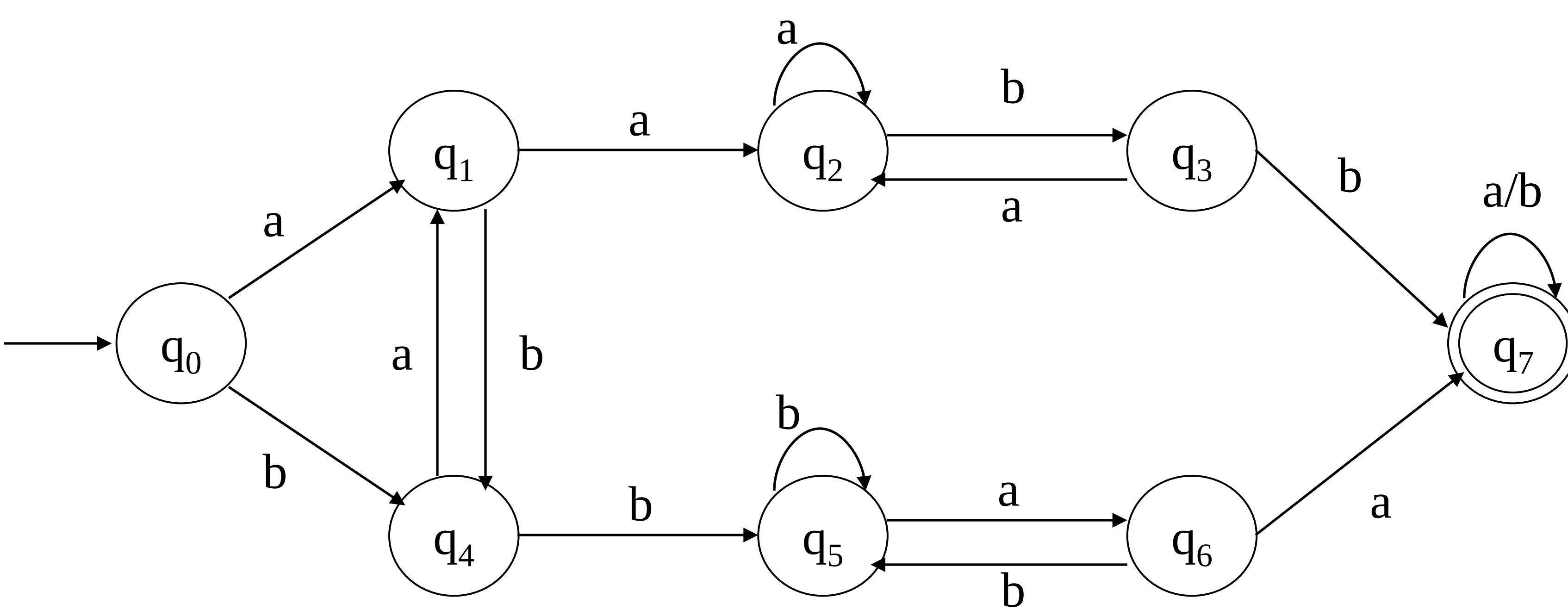
- Give a DFA M such that:

L(M) = {x | x is a string of a's, b's and c's such that x contains the substring *aba*}



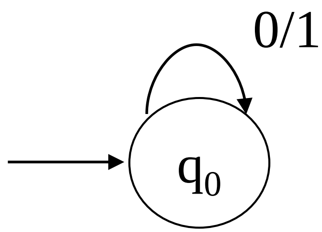
- Give a DFA M such that:

L(M) = {x | x is a string of a's and b's such that x contains both *aa* and *bb*}

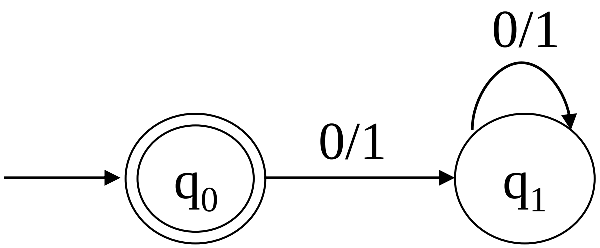


- Let $\Sigma = \{0, 1\}$. Give DFAs for $\{\}$, $\{\varepsilon\}$, Σ^*, and Σ^+.

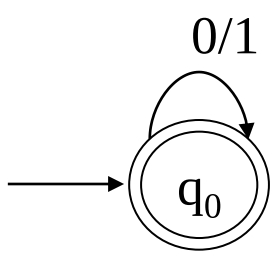
For $\{\}$:



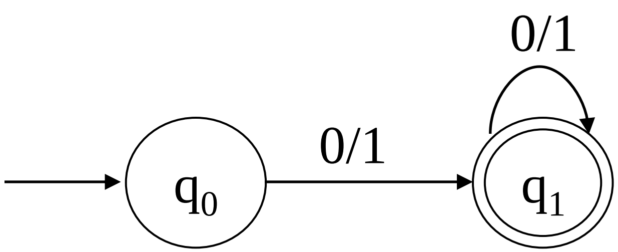
For $\{\varepsilon\}$:



For Σ^*:



For Σ^+:



SOME CLOSURE PROPERTIES OF REGULAR SETS

Issue: what languages can be accepted by finite automata ?

- Recall the definitions of some language operations:
  - $A \cup B = \{x \mid x \in A \text{ or } x \in B\}$.
  - $A \cap B = \{x \mid x \in A \wedge x \in B\}$
  - $\sim A = \Sigma^* - A = \{x \in \Sigma^* \mid x \notin A\}$
  - $AB = \{xy \mid x \in A \wedge y \in B\}$
  - $A^* = \{x_1 x_2 \ldots x_n \mid n \geq 0 \wedge x_i \in A \text{ for } 0 \leq i \leq n\}$
  - and more ... ex: $A / B = \{x \mid \exists y \in B \text{ s.t. } xy \in A\}$.
- Problem: If A and B are regular [languages], then which of the above sets are regular as well?

Ans: _______.

THE PRODUCT CONSTRUCTION

- $M_1 = (Q_1,\Sigma,\delta_1,s_1,F_1)$, $M_2 = (Q_2,\Sigma,\delta_2,s_2,F_2)$: two DFAs

Define a new machine $M_3 = (Q_3, \Sigma, \delta_3, s_3, F_3)$ where

  - $Q_3 = Q_1 \times Q_2 = \{(q_1,q_2) \mid q_1 \in Q_1 \text{ and } q_2 \in Q_2 \}$
  - $s_3 = (s_1,s_2)$;
  - $F_3=F_1 x F_2 = \{(q_1,q_2) \mid q_1 \in F_1 \wedge q_2 \in F_2\}$ and
  - $\delta_3:Q_3 \times \Sigma \to Q_3$ is defined to be

    $\delta_3((q_1,q_2), a) = (\delta_1 (q_1,a), \delta_2 (q_2,a))$

    for all $(q_1,q_2)\in Q$, $a \in \Sigma$.

- The machine M_3, denoted $M_1 x M_2$, is called the *product* of M_1 and M_2. The behavior of M_3 may be viewed as the parallel execution of M_1 and M_2.
- Lem 4.1: For all $x \in \Sigma^*$, $\Delta_3((p,q),x) = (\Delta_1(p,x), \Delta_2(q,x))$.

Pf: By induction on the length $|x|$ of x.

Basis: $|x|= 0$: then $\Delta_3((p,q),\varepsilon) = (p,q) = (\Delta_1 (p,\varepsilon), \Delta_2(q,\varepsilon))$

THE PRODUCT CONSTRUCTION (CONT'D)

Ind. step: assume the lemma hold for x in Σ^*, we show it holds for xa, where a in Σ.

$\Delta_3((p,q),xa) = \delta_3(\Delta_3((p,q),x), a)$ --- definition of Δ_3

$= \delta_3((\Delta_1(p,x), \Delta_2(q,x)), a)$ --- Ind. hyp.

$= (\delta_1(\Delta_1(p,x),a), \delta_2(\Delta_2(q,x),a)$ --- def. of δ_3

$= (\Delta_1(p,xa), \Delta_2(p,xa))$ QED --- def of Δ_1 and Δ_2.

Theorem 4.2: $L(M_3) = L(M_1) \cap L(M_2)$.

pf: for all $x \in \Sigma^*$, $x \in L(M_3)$

iff $\Delta_3(s_3,x) \in F_3$ --- def. of acceptance

iff $\Delta_3((s_1,s_2),x) \in F_3$ --- def. of s_3

iff $(\Delta_1(s_1,x), \Delta_2(s_2,x)) \in F_3 = F_1xF2$ --- def. of F_3

iff $\Delta_1(s_1,x) \in F_1$ and $\Delta_2(s_2,x) \in F_2$ --- def. of set product

iff $x \in L(M_1)$ and $x \in L(M_2)$ --- def. of acceptance

iff $x \in L(M_1) \cap L(M_2)$. QED --- def. of intersection.

REGULAR LANGUAGES ARE CLOSED UNDER U, $\cap$ AND ~

Theorem: IF A and B are regular than so are $A \cap B$, ~A and AUB.

pf: (1) A and B are regular

=> $\exists$ DFA M_1 and M_2 s.t. $L(M_1) = A$ and $L(M_2) = B$ -- def. of RL

=> $L(M_1 x M_2) = L(M_1) \cap L(M_2) = A \cap B$ --- Theorem 4.2

==> $A \cap B$ is regular. -- def. of RL.

(2) Let $M = (Q,\Sigma,\delta,s,F)$ be the machine s.t. $L(M) = A$.

Define $M' = (Q,\Sigma,\delta,s,F')$ where $F' = {\sim}F = \{q \in Q \mid q \notin F\}$.

Now for all x in Σ^*, $x \in L(M')$

<=> $\Delta(s,x) \in F' = {\sim}F$ --- def. of acceptance

<=> $\Delta(s,x) \notin F$ --- def of ~F

<=> $x \notin L(M)$ iff $x \notin A$. -- def. of acceptance

Hence ~A is accepted by L(M') and is regular !

(3). Note that $AUB = {\sim}({\sim}A \cap {\sim}B)$. Hence the fact that A and B are regular implies ~A, ~B, $({\sim}A \cap {\sim}B)$ and ${\sim}({\sim}A \cap {\sim}B) = AUB$ are regular too.