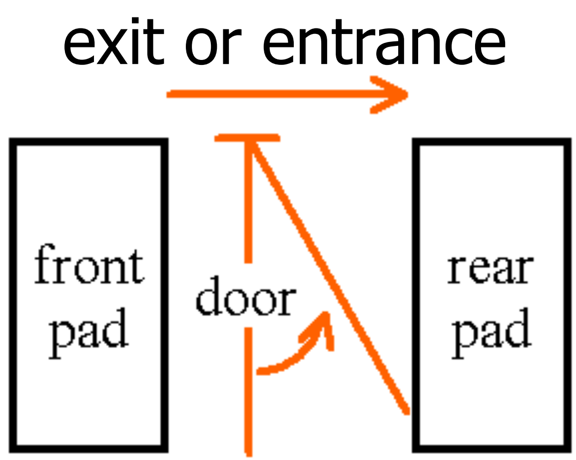
Regular Languages

- Finite Automata
  - e.g., Supermarket automatic door:



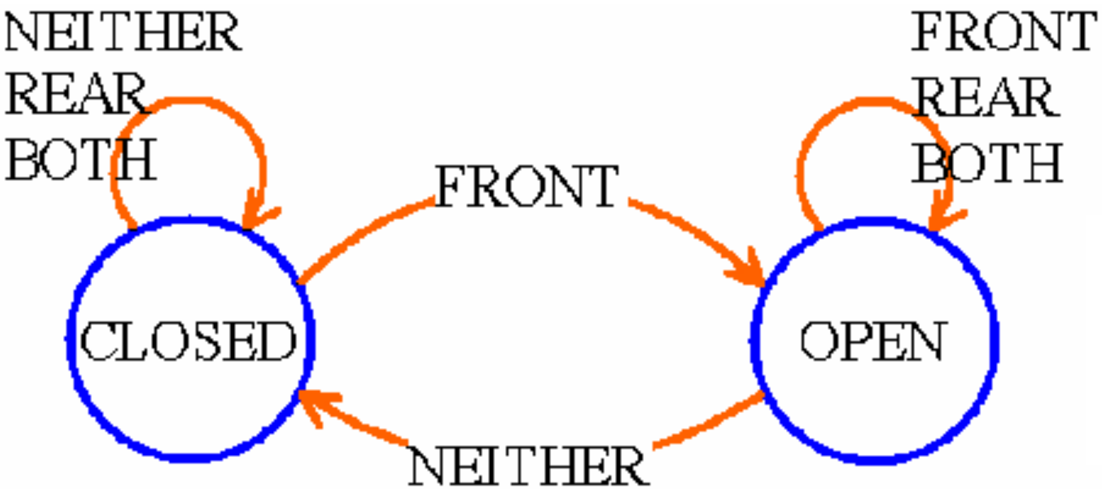
Finite Automata

Input signal

Door state

	NEITHER	FRONT	REAR	BOTH
CLOSED	CLOSED	OPEN	CLOSED	CLOSED
OPEN	CLOSED	OPEN	OPEN	OPEN

State transition table



State diagram

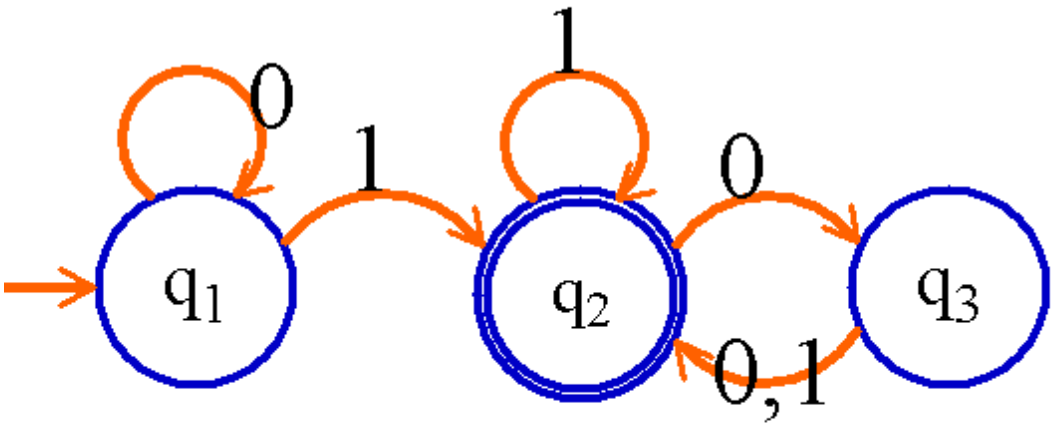
Definition

- A finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$
  - Q : a finite set called the states
  - Σ : a finite set called the alphabet
  - $\delta : Q \times \Sigma \rightarrow Q$ is the transition function
  - $q_0 \in Q$ is the initial (or start) state
  - $F \subseteq Q$ is the set of accept (or final) states

- $M_1 = (Q, \Sigma, \delta, q_0, F)$
  - $Q = \{q_1, q_2, q_3\}$
  - $\Sigma = \{0, 1\}$
  - δ :

δ	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_2	q_2

  - q_1: the start state
  - $F = \{q_2\}$



$L(M_1) = A$

M recognizes A or

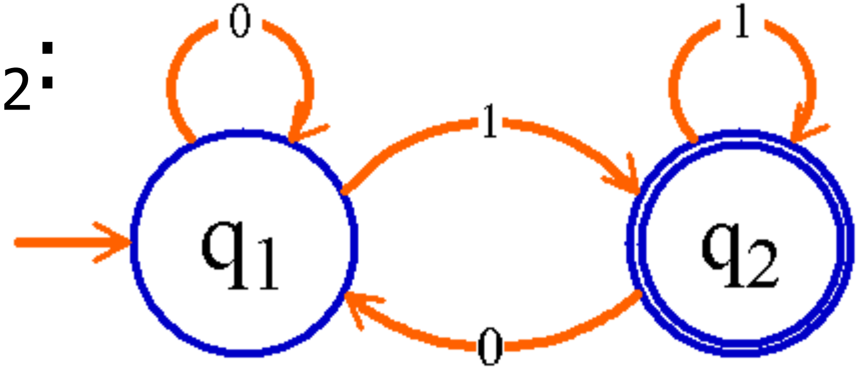
M accepts A

the set of all strings that M accepts

$L(M_1) = ?$

e.g.:

M_2:

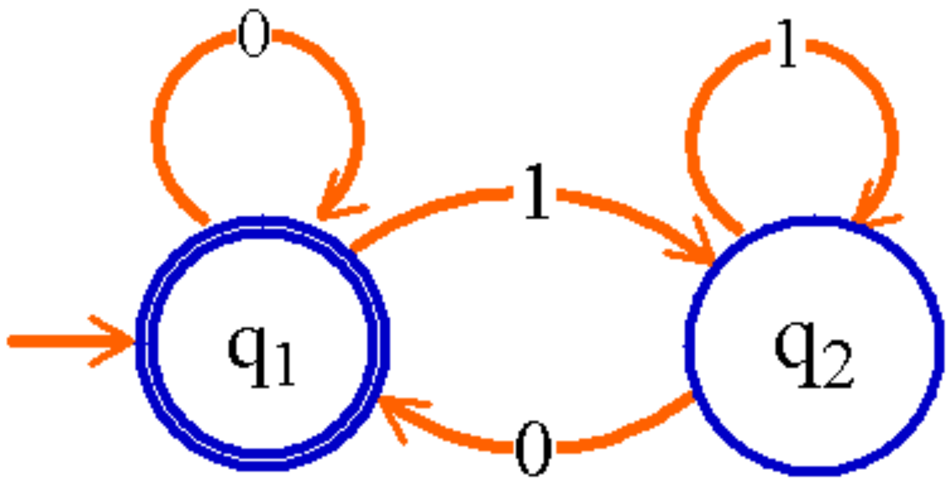


$M_2 = (\{q_1, q_2\}, \{0,1\}, \delta, q_1, \{q_2\})$

δ:	0	1
q_1	q_1	q_2
q_2	q_1	q_2

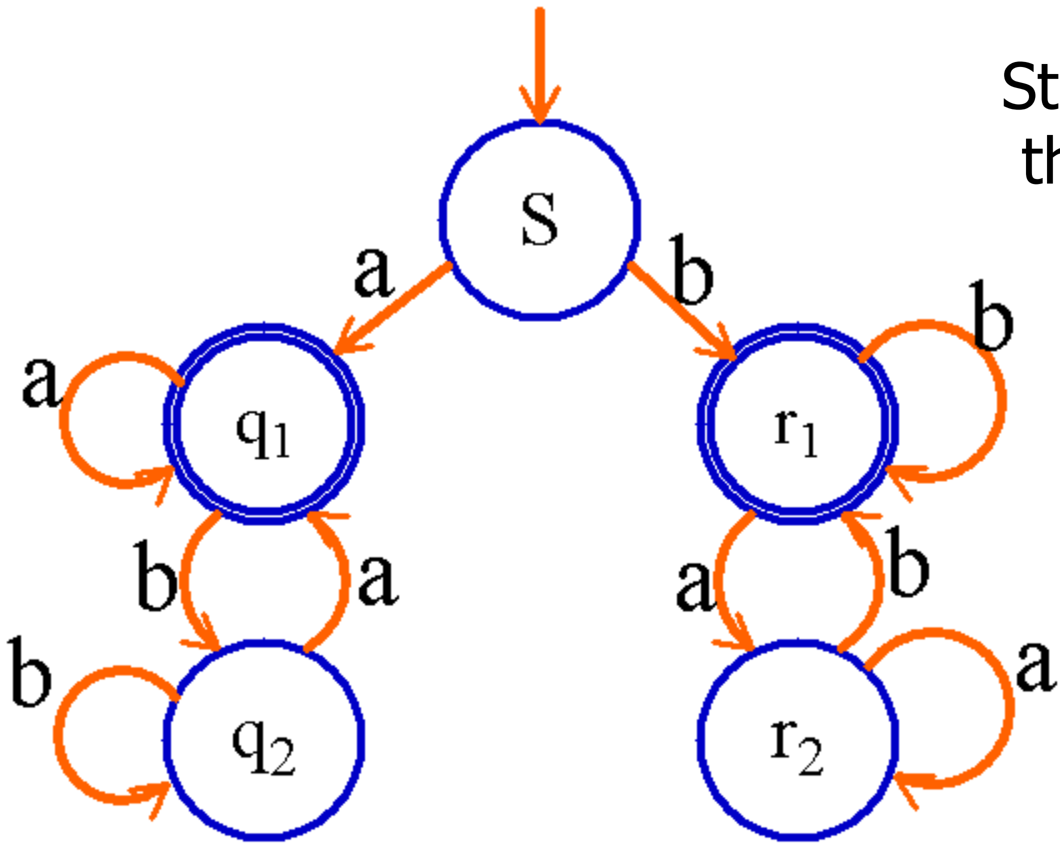
$L(M_2) = \{w \mid w \text{ ends in a } 1\}$

M_3:



$L(M_3)$={ε or ends in 0}

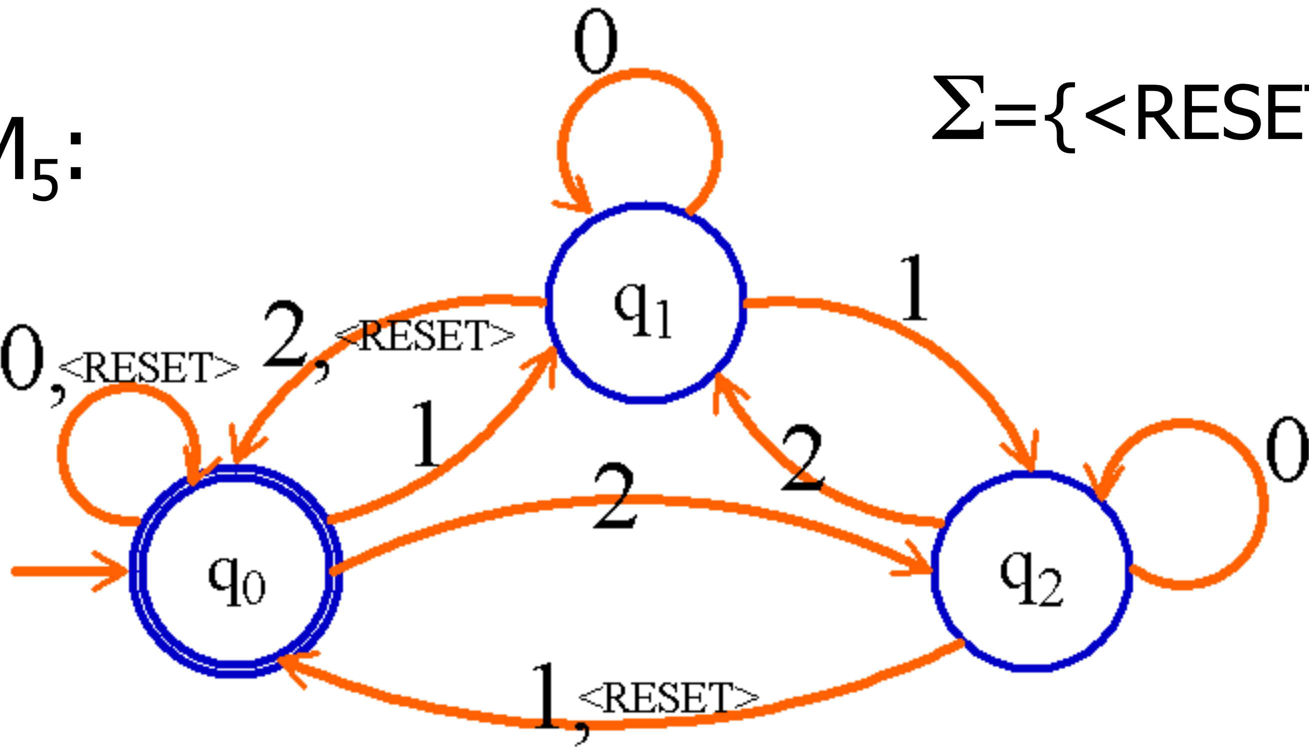
M$_4$:



Starts and ends with the same symbol

M$_5$:

Σ={<RESET>, 0, 1, 2}

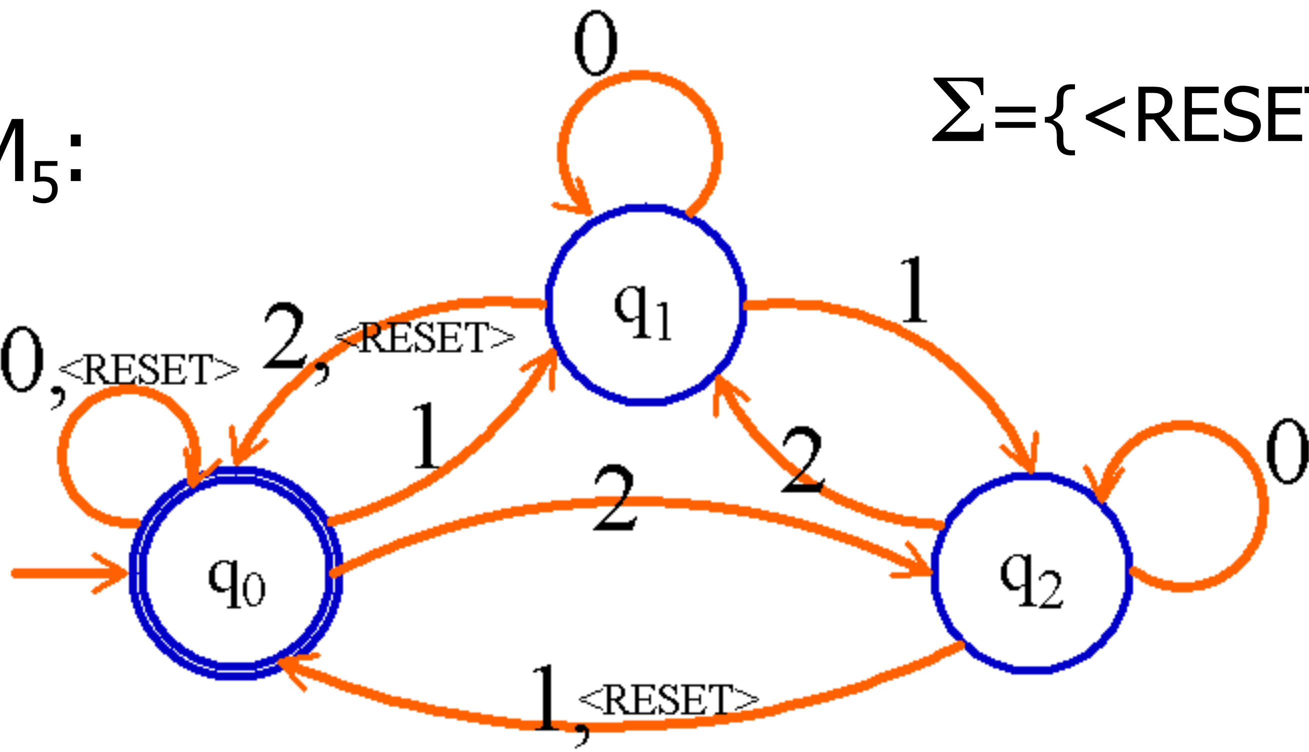


1 0 <RESET> 2 2 <RESET> 0 1 2

What is the language accepted by the above automata?

M_5:

Σ={<RESET>, 0, 1, 2}



L(M_5) = {w | the sum of the symbols in w is 0 modulo 3 except the <RESET> resets to 0}

Formal definition of computation

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

  w : a string over Σ, $w_i \in \Sigma$, $w = w_1 w_2 \ldots w_n$

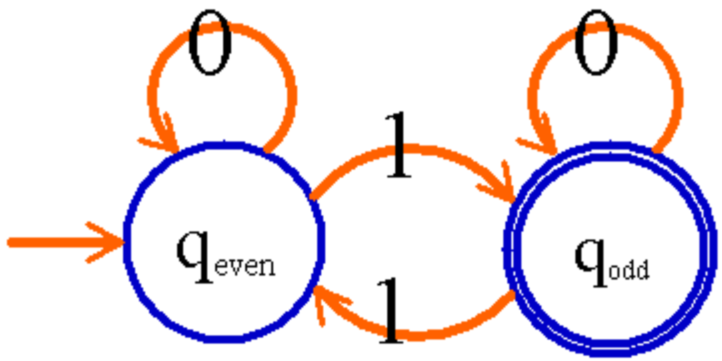
- M accepts w if a sequence of states $r_0, r_1, \ldots, r_n$ exists in Q and satisfies the following 3 conditions:
  - $r_0 = q_0$
  - $\delta(r_i, w_{i+1}) = r_{i+1}$, for $i = 0, \ldots, n-1$
  - $r_n \in F$

- We say that M recognizes language A if
  $A = \{w \mid M \text{ accepts } w\}$

- Def:
  A language is called a regular language if some finite automaton recognizes it.

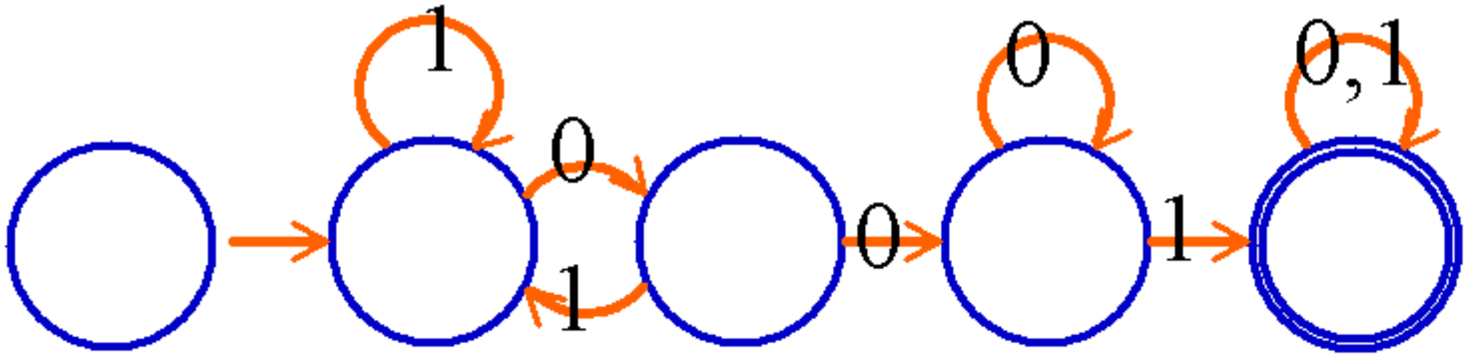
Designing finite automata

- Let $A = \{w \mid w$ is 0-1 string & has odd number of 1$\}$
  - Is A a regular language ?
  - What is the automata accepting A ?



Designing finite automata

- E.g.: Design a finite automaton to recognize all strings that contains 001 as a substring



Regular operations

- A, B : languages

  Regular operations :
  union, concatenation, star

  – Union :

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

  – Concatenation :

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

  – Star :

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

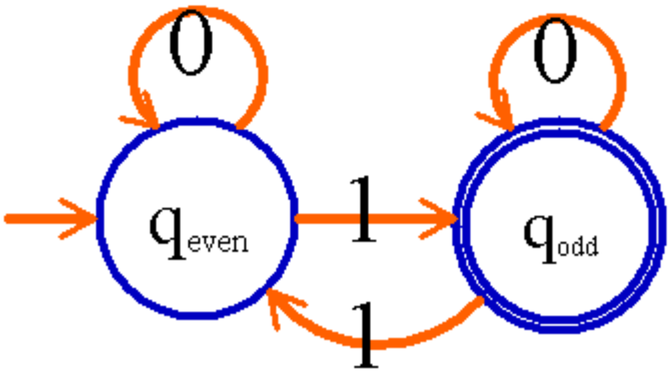
Regular operations

- eg:
  Σ = { a, b, c, …, z} *(26 letters)*
  A = { good, bad}
  B = { boy, girl}
- $A \cup B$ = { good, bad, boy, girl}
  $A \circ B$= {goodboy, goodgirl, badboy,badgirl}
  A^* = {ε, good, bad, goodbad, badgood, goodgood, badbad, goodgoodgood, goodgoodbad, ………}

- E.g. $A_1 = \{w \mid w$ has odd number of 1$\}$
  - $Q_1 = \{q_{even}, q_{odd}\}$
  - $\Sigma = \{0,1\}$
  - δ_1:

	0	1
q_{even}	q_{even}	q_{odd}
q_{odd}	q_{odd}	q_{even}

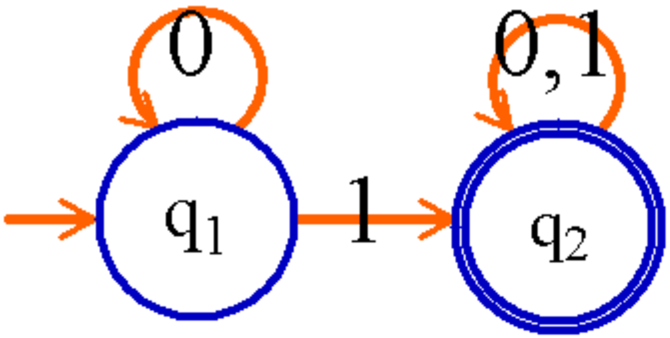
  - q_{even} : start state
  - q_{odd} : accept state



- E.g. $A_2=\{w \mid w$ has at least a "1"$\}$
  - $Q_2 = \{q_1, q_2\}$
  - $\Sigma = \{0,1\}$
  - δ_2:

	0	1
q_1	q_1	q_2
q_2	q_2	q_2

  - q_1: start state
  - q_2: accept state



- $A_1 \cup A_2 = \{w \mid w$ has odd number of 1 or w has a 1$\}$
  - $Q = \{(q_{even}, q_1), (q_{even}, q_2), (q_{odd}, q_1), (q_{odd}, q_2)\}$
  - $\Sigma = \{0, 1\}$
  - δ:

	0	1
(q_{even}, q_1)	(q_{even}, q_1)	(q_{odd}, q_2)
(q_{even}, q_2)	(q_{even}, q_2)	(q_{odd}, q_2)
(q_{odd}, q_1)	(q_{odd}, q_1)	(q_{even}, q_2)
(q_{odd}, q_2)	(q_{odd}, q_2)	(q_{even}, q_2)

  - (q_{even}, q_1): start state
  - $(q_{odd}, q_1)(q_{odd}, q_2)(q_{even}, q_2)$: accept state

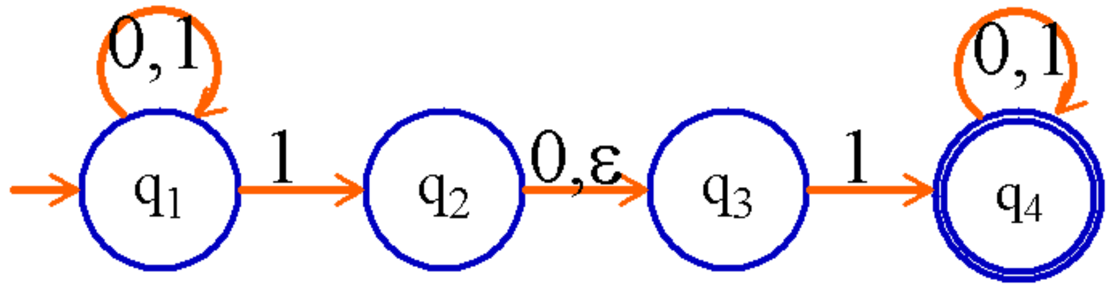
- Theorem:
  The class of regular languages is closed under the union operation. (In other words, if A_1 and A_2 are regular languages, so is $A_1 \cup A_2$.)

- Pf : Let
  M_1 recognize A_1, where $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$
  M_2 recognize A_2, where $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$
  Goal: Construct $M=(Q, \Sigma, \delta, q_0, F)$ to recognize $A_1 \cup A_2$
  - $Q=\{(r_1,r_2) | r_1 \in Q_1 \text{ and } r_2 \in Q_2\}$
  - Σ is the same in M_1, M_2
  - For each $(r_1, r_2) \in Q$ and each $a \in \Sigma$, let
    $\delta((r_1,r_2), a)=(\delta_1(r_1,a), \delta_2(r_2,a))$: [If $\delta_1(r_1,a)$ is not defined, then how to define this transition?]
  - $q_0=(q_1, q_2)$, (q_0 is a new state $\notin Q_1 \cup Q_2$)
  - $F=\{(r_1, r_2) | r_1 \in F_1$ **or** $r_2 \in F_2\}$

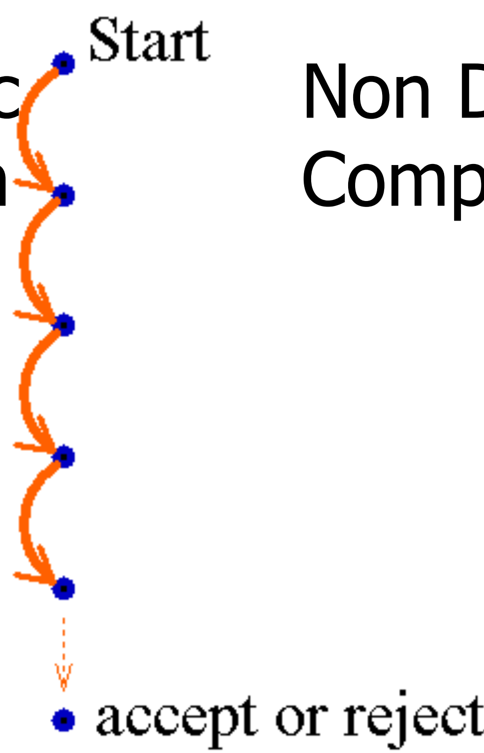
- Theorem:
  The class of regular languages is close under the concatenation. (In other words, if A_1 and A_2 are regular languages, so is $A_1 \circ A_2$.)

Nondeterminism:

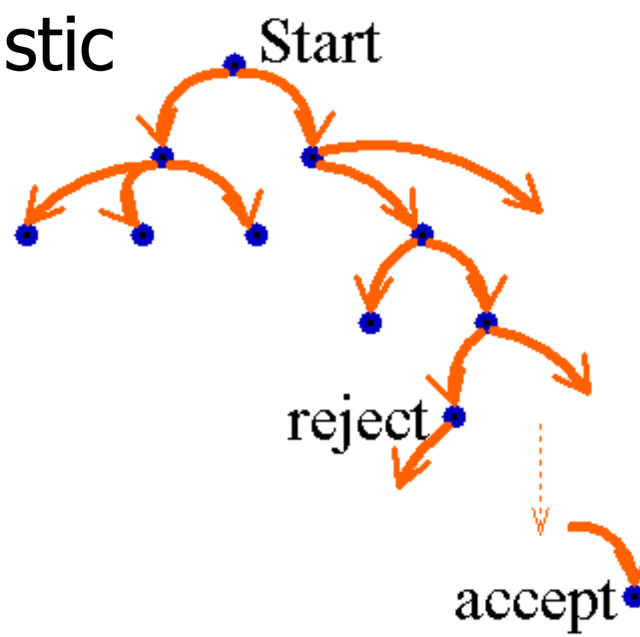
- E.g.

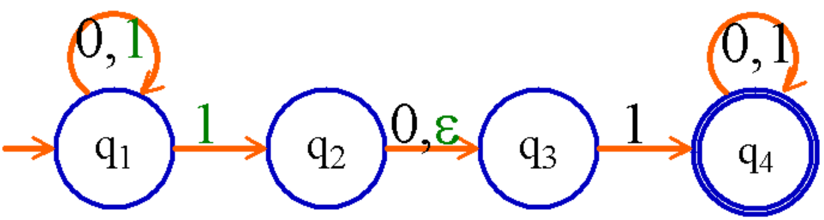


Deterministic Computation

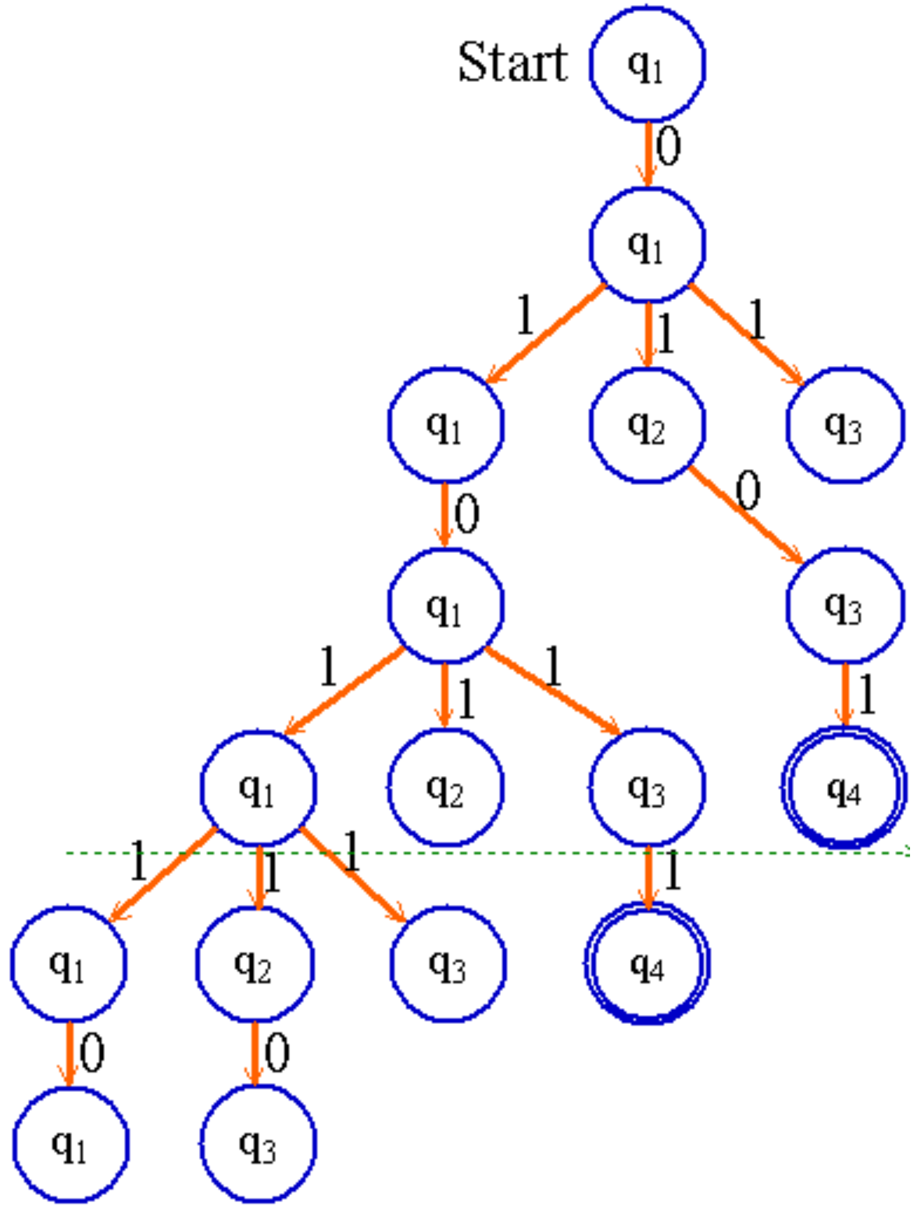


Non Deterministic Computation

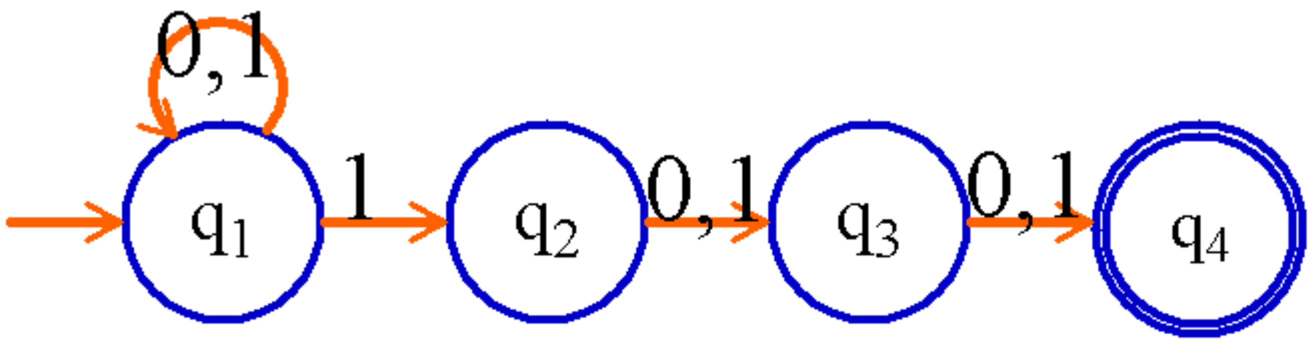




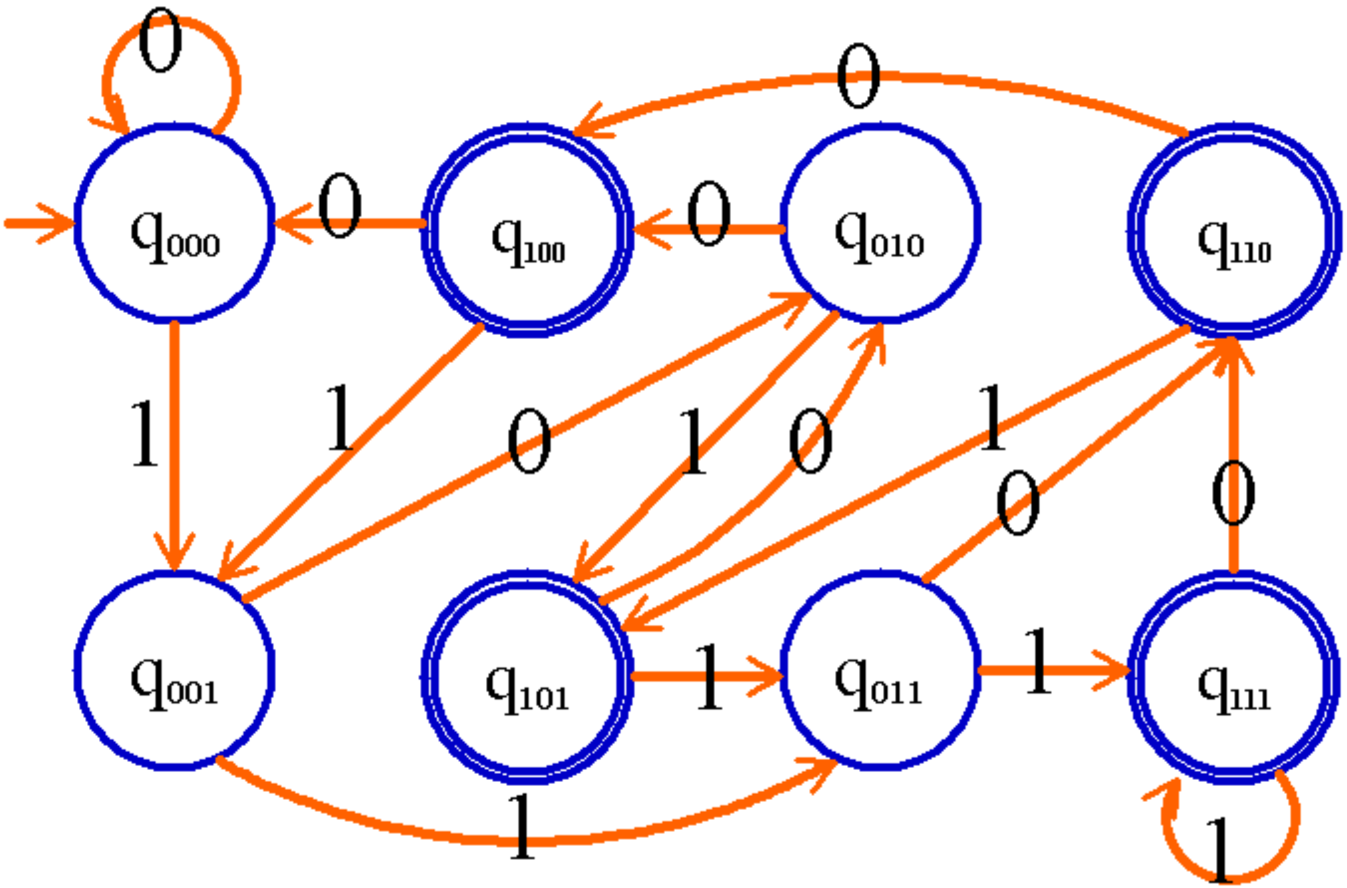
Input : 0 1 0 1 1 0



- eg:



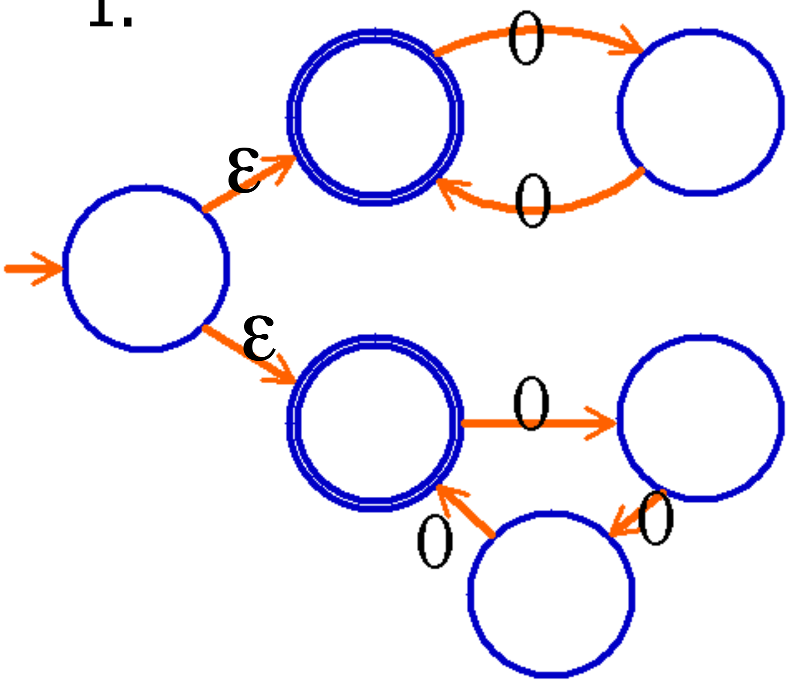
A : the language consisting of all strings over {0, 1} containing a 1 in the 3rd position from the end



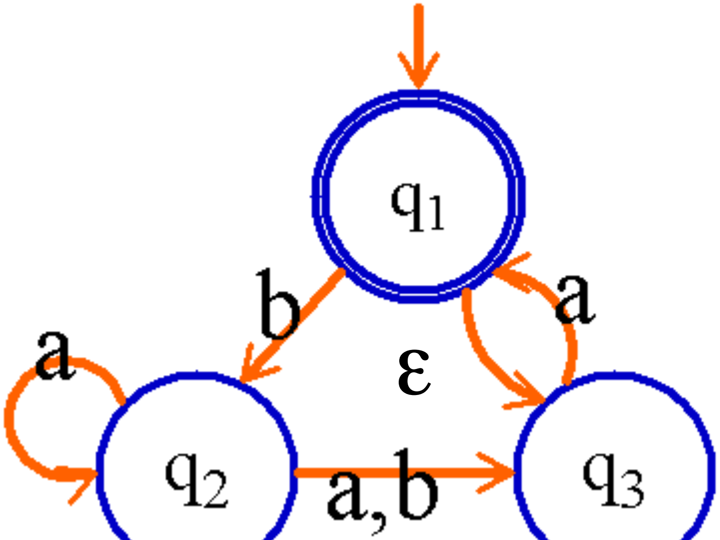
0 1 1 0 0 1

- E.g.: What does the following automaton accept?

1.



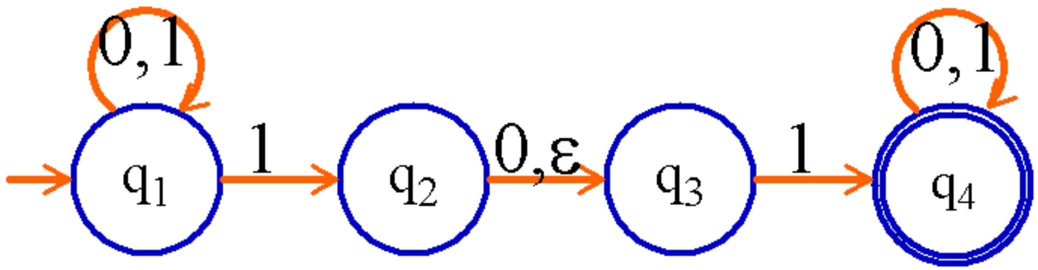
2.



Formal definition of a nondeterministic finite automaton

- $\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$,
  $\mathcal{P}$(Q) : the collection of all subsets of Q
- A nondeterministic finite automaton is a 5-tuple (Q, Σ, δ, q_0, F), where
  - Q : a finite set of states
  - Σ : a finite set of alphabet
  - δ : $Q \times \Sigma_\varepsilon \rightarrow \mathcal{P}(Q)$ transition function
  - $q_0 \in Q$: start state
  - $F \subseteq Q$: the set of accept states

- E.g.



  - $Q = \{q_1, q_2, q_3, q_4\}$
  - $\Sigma = \{0, 1\}$
  - δ :

	0	1	ε
q_1	$\{q_1\}$	$\{q_1, q_2\}$	ϕ
q_2	$\{q_3\}$	ϕ	$\{q_3\}$
q_3	ϕ	$\{q_4\}$	ϕ
q_4	$\{q_4\}$	$\{q_4\}$	ϕ

  - q_1 : start state
  - q_4 : accept state

- $N=(Q, \Sigma, \delta, q_0, F)$ ~NFA
  w : string over Σ , $w= y_1 y_2 y_3 \ldots y_m$, $y_i \in \Sigma_\varepsilon$, $r_0, r_1, r_2, \ldots, r_m \in Q$

- N accepts w if there exist $r_0, r_1, r_2, \ldots, r_m$ such that
  - $r_0=q_0$
  - $r_{i+1} \in \delta(r_i, y_{i+1})$, for $i=0,\ldots m-1$
  - $r_m \in F$

Equivalence of NFA and DFA

- Theorem : Every nondeterministic finite automaton has an equivalent deterministic finite automaton
- Pf: Let N=(Q, Σ, δ, q_0, F) be the NFA recognizing A.

  Goal: Construct a DFA recognizing A

  1. First we don't consider ε. Construct M= (Q', Σ, δ', q_0', F')

     1. Q' = $\mathcal{P}$(Q)
     2. For R$\in$Q' and a$\in\Sigma$

        let $$\delta'(R,a) = \{q \in Q \mid q \in \delta(r,a), r \in R\}$$

        $$= \bigcup_{r \in R} \delta(r,a)$$
     3. q_0' = {q_0}
     4. F' = {R$\in$Q' | R contains an accept state of N}

Equivalence of NFA and DFA

2. Consider ε : (subset construction)

For any $R \in M$, define

$E(R)=\{q \mid q$ can be reached from R by traveling along 0 or more $\varepsilon\}$

Replace $\delta(r,a)$ by $E(\delta(r,a))$

Thus,

$\delta'(R, a)=\{q \in Q : q \in E(\delta(r,a))$ for some $r \in R\}$

Change q_0' to be $E(\{q_0\})$

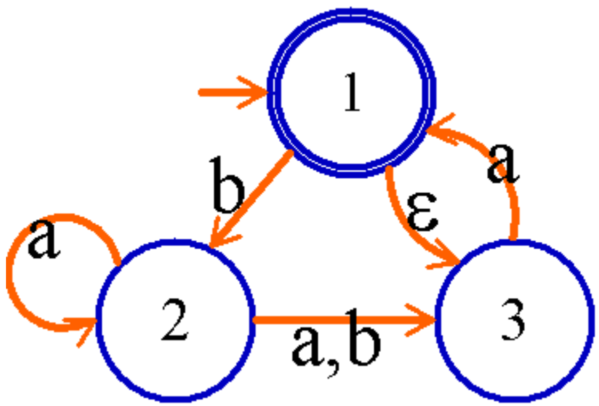
- Corollary: A language is regular if and only if some NFA recognizes it.

e.q.,

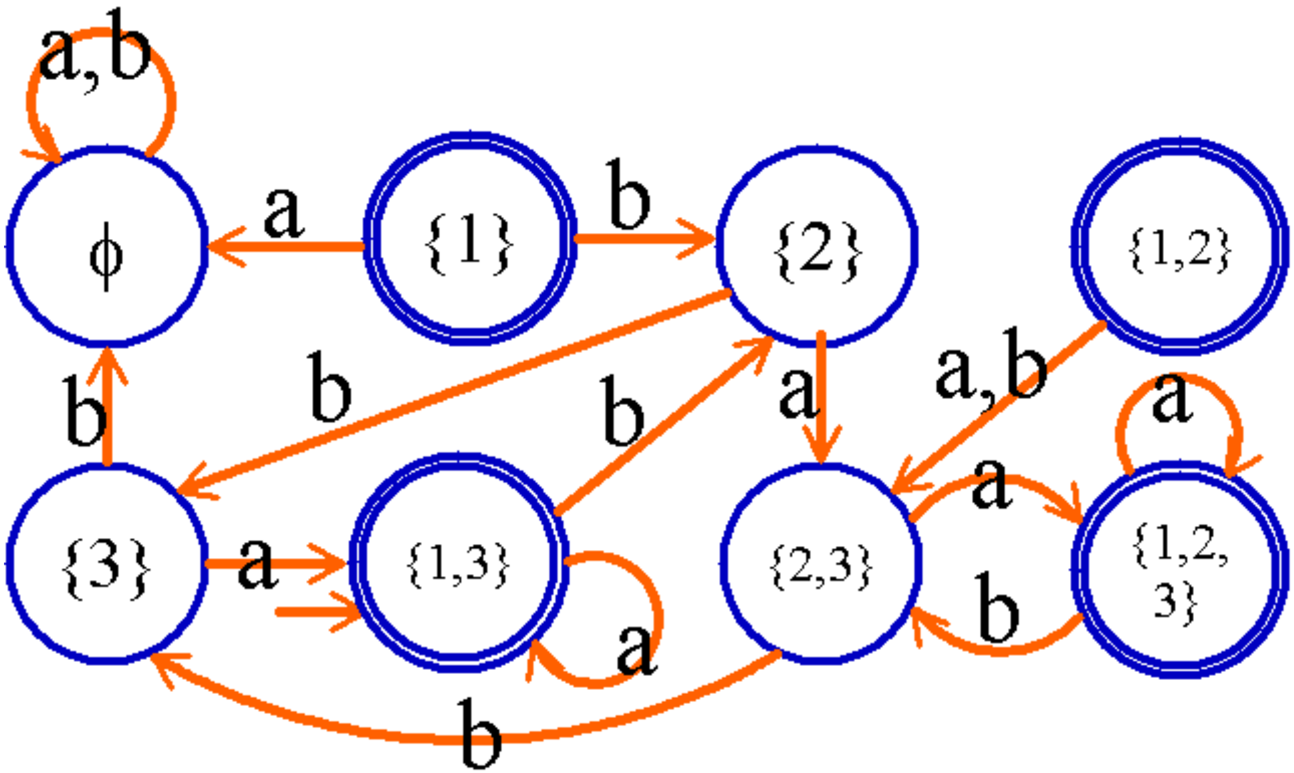
D's states:

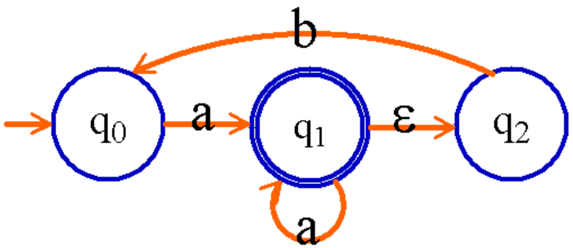
$\{\phi, \{1\}, \{2\}, \{3\}, \{1,2\}, \{1,3\}, \{2,3\}, \{1,2,3\}\}$

$N_4 = (\{1,2,3\}, \{a,b\}, \delta, 1, \{1\})$



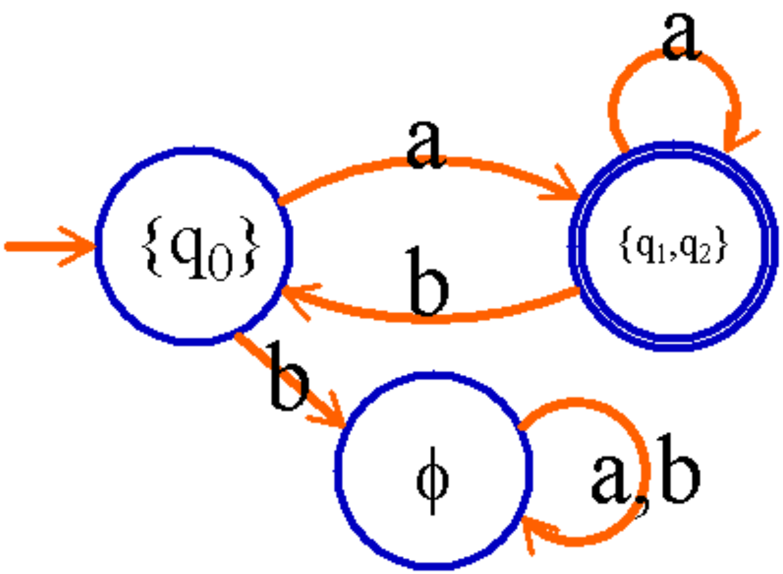
- Construct a DFA D equivalent to N_4





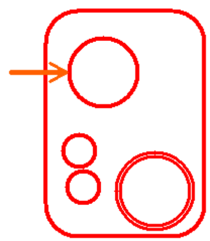
- E.g., Convert the following NFA to an equivalent DFA

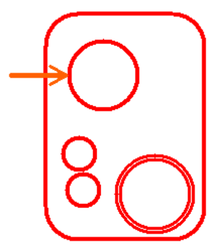
- $E(\{q_0\}) = \{q_0\}$
  $E(\delta(\{q_0\},a)) = \{q_1,q_2\}$
  $E(\delta(\{q_0\},b)) = \phi$
  $E(\delta(\{q_1,q_2\},a) = \{q_1,q_2\}$
  $E(\delta(\{q_1,q_2\},b) = \{q_0\}$



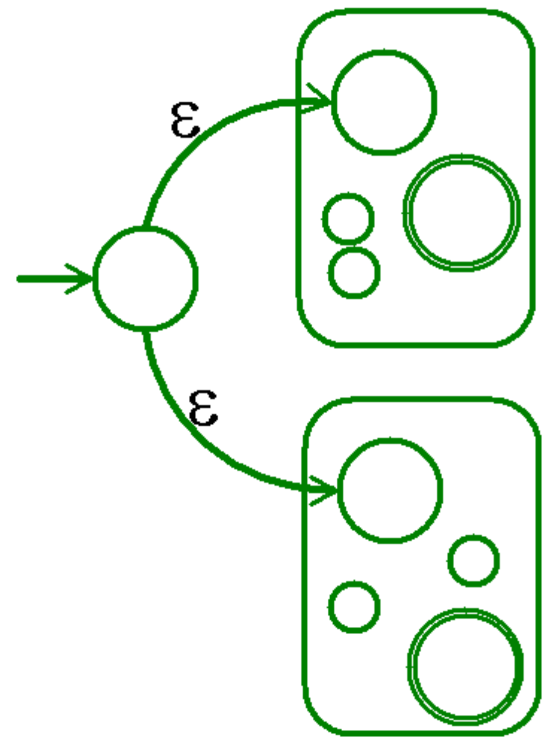
Closure under the regular operations

- Theorem : The class of regular languages is closed under the union operation
- Pf:

  A_1: N_1: 

  A_2: N_2: 

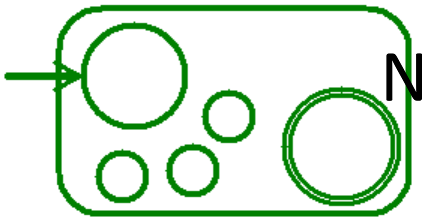
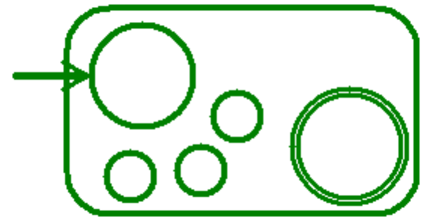
  A1∪A2

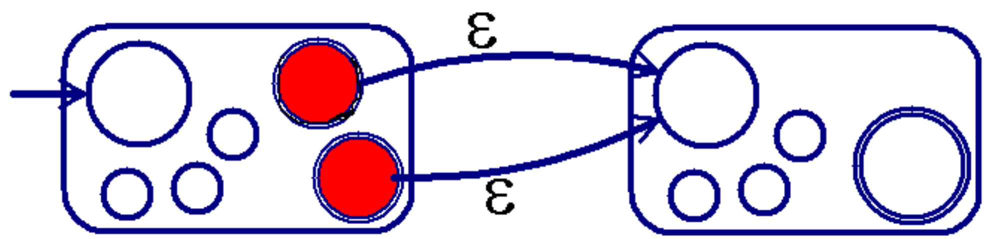
  – $N_1=(Q_1, \Sigma, \delta_1, q_1, F_1)$ for A_1
  – $N_2=(Q_2, \Sigma, \delta_2, q_2, F_2)$ for A_2

- Construct $N=(Q, \Sigma, \delta, q_0, F)$ to recognize $A1 \cup A2$
  - Q : $\{q_0\} \cup Q_1 \cup Q_2$
  - q_0 : start state
  - F : $F_1 \cup F_2$
  - For $q \in Q$ and $a \in \Sigma_\varepsilon$

$$\delta(q, a)= \begin{cases} \delta_1(q, a), & q \in Q_1 \\ \delta_2(q, a), & q \in Q_2 \\ \{q_1, q_2\}, & q=q_0 \text{ and } a=\varepsilon \\ \phi, & q=q_0 \text{ and } a \neq \varepsilon \end{cases}$$

- Theorem : The class of regular language is closed under the concatenation operation.
- Pf: N_1:  N_2: 

$A1 \circ A2$

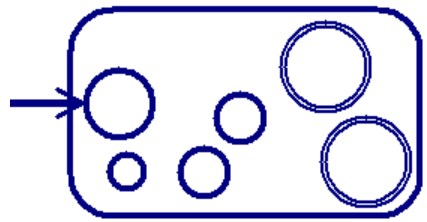


  - – $N_1=(Q_1, \Sigma, \delta_1, q_1, F_1)$ recognizes A_1.
  - – $N_2=(Q_2, \Sigma, \delta_2, q_2, F_2)$ recognizes A_2.

- Construct $N=(Q, \Sigma, \delta, q_1, F_2)$ to recognize $A1 \circ A2$
  - $Q : Q_1 \cup Q_2$
  - q_1 : start state
  - F_2 : accept state
  - $$\delta(q, a) = \begin{cases} \delta_1(q, a), & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q, a), & q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q, a) \cup \{q_2\}, & q \in F_1 \text{ and } a = \varepsilon \\ \delta_2(q, a), & q \in Q_2 \end{cases}$$

- Theorem : The class of regular language is closed under the star operation.

Pf: A:

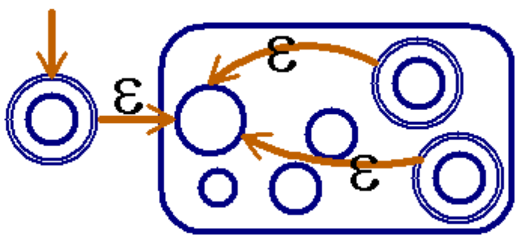


$A = \{a, b\}$

$A^* = \{\varepsilon, a, b, \ldots\}$

ε, A
$A \circ A = \{aa, ab, ba, bb\}$
$A \circ A \circ A$
…

A*:



– $N_1=(Q_1, \Sigma, \delta_1, q_1, F_1)$ recognize A

- Construct $N=(Q, \Sigma, \delta, q_0, F)$ to recognize A^*
  - Q : $\{q_0\} \cup Q$
  - q_0 : start state
  - F : $\{q_0\} \cup F$
  - $$\delta(q, a) = \begin{cases} \delta_1(q, a), & q \in Q_1 \text{ and } q \notin F_1 \\ \delta_1(q, a), & q \in F_1 \text{ and } a \neq \varepsilon \\ \delta_1(q, a) \cup \{q_1\}, & q \in F_1 \text{ and } a = \varepsilon \\ \{q_1\}, & q = q_0 \text{ and } a = \varepsilon \\ \phi & q = q_0 \text{ and } a \neq \varepsilon \end{cases}$$

Regular Expressions

- AWK, GREP in UNIX
  PERL, text editors.
- E.g.
  $(0 \cup 1)0^* = (\{0\} \cup \{1\}) \circ \{0\}^*$
- E.g.
  $(0 \cup 1)^* = \{0, 1\}^*$

Formal definition of a regular expression

- Definition:
  R is a regular expression if R is
  - $a \in \Sigma$
  - ε
  - ϕ
  - $(R_1 \cup R_2)$, R_1 and R_2 are regular
  - $(R_1 \circ R_2)$, R_1 and R_2 are regular
  - $(R_1{}^*)$, R_1 is regular expression

  Inductive definition

- E.g., $\Sigma = \{0, 1\}$
  - $0^*10^* = \{w : w \text{ has exactly a single } 1\}$
  - $\Sigma^*1\Sigma^*$
  - $\Sigma^*001\Sigma^*$
  - $(\Sigma\Sigma)^*$
  - $01 \cup 10$
  - $0\Sigma^*0 \cup 1\Sigma^*1 \cup 0 \cup 1$
  - $(0\cup\varepsilon)1^*=01^*\cup1^*$
  - $(0\cup\varepsilon)(1\cup\varepsilon) = \{\varepsilon, 0, 1, 01\}$
  - $1^*\phi = \phi$
  - $\phi^* = \{\varepsilon\}$

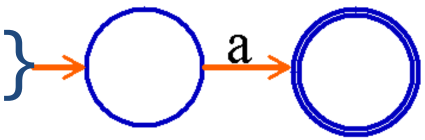
- E.g.
  R : regular expression
  $R \cup \phi = R$
  $R \circ \varepsilon = R$
- $R \cup \varepsilon$?$=R$
  $R \circ \phi$?$= R$
- $R = \{0\}$. Then $R \cup \varepsilon = \{0, \varepsilon\} \neq R$
  $R \circ \phi = \phi$

Equivalence with finite automata

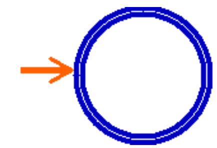
- Theorem:
  A language is regular if and only if some regular expression describes it.

- Lemma: ($\leftarrow$)
  If a language is described by a regular expression then it is regular.

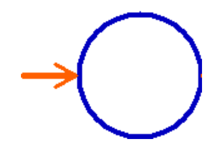
- Pf:
  Let R be a regular expression describing some language A.
  Goal: Convert R into an NFA N.
  - $R=a \in \Sigma$, $L(R)=\{a\}$



  - $R=\varepsilon$, $L(R)=\{\varepsilon\}$



  - $R=\phi$, $L(R)=\phi$

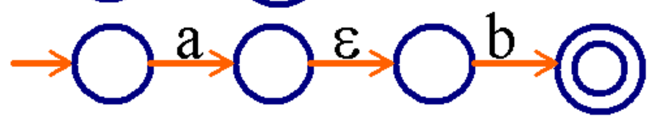


  - $R = R_1 \cup R_2$
  - $R = R_1 \circ R_2$
  - $R = R_1^*$

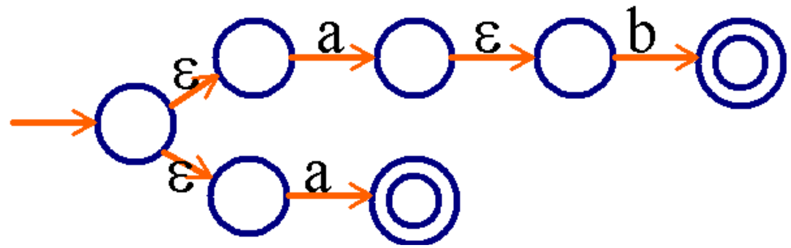
- E.g. (ab ∪ a)*
  - a

  - b

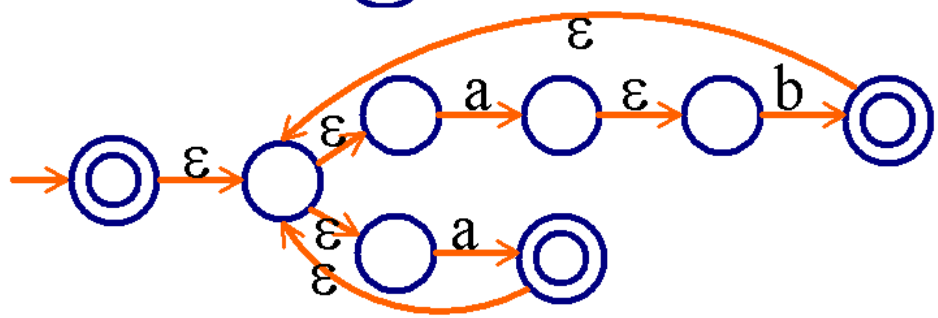
  - ab



  - ab ∪ a

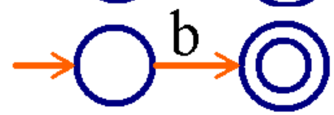


  - (ab ∪ a)*

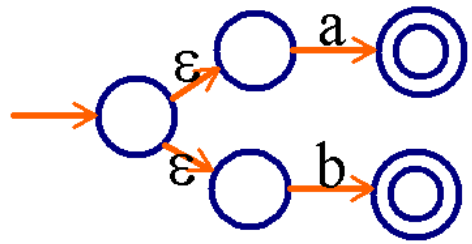


- eg. (a ∪ b)*aba
  - a

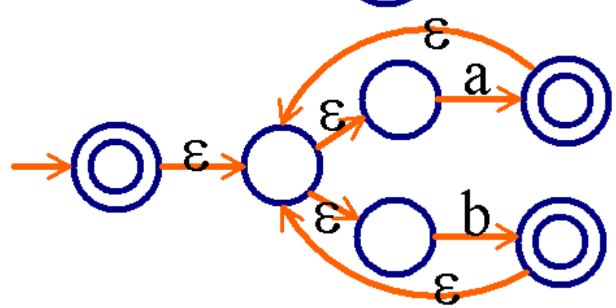
  - b



  - a ∪ b

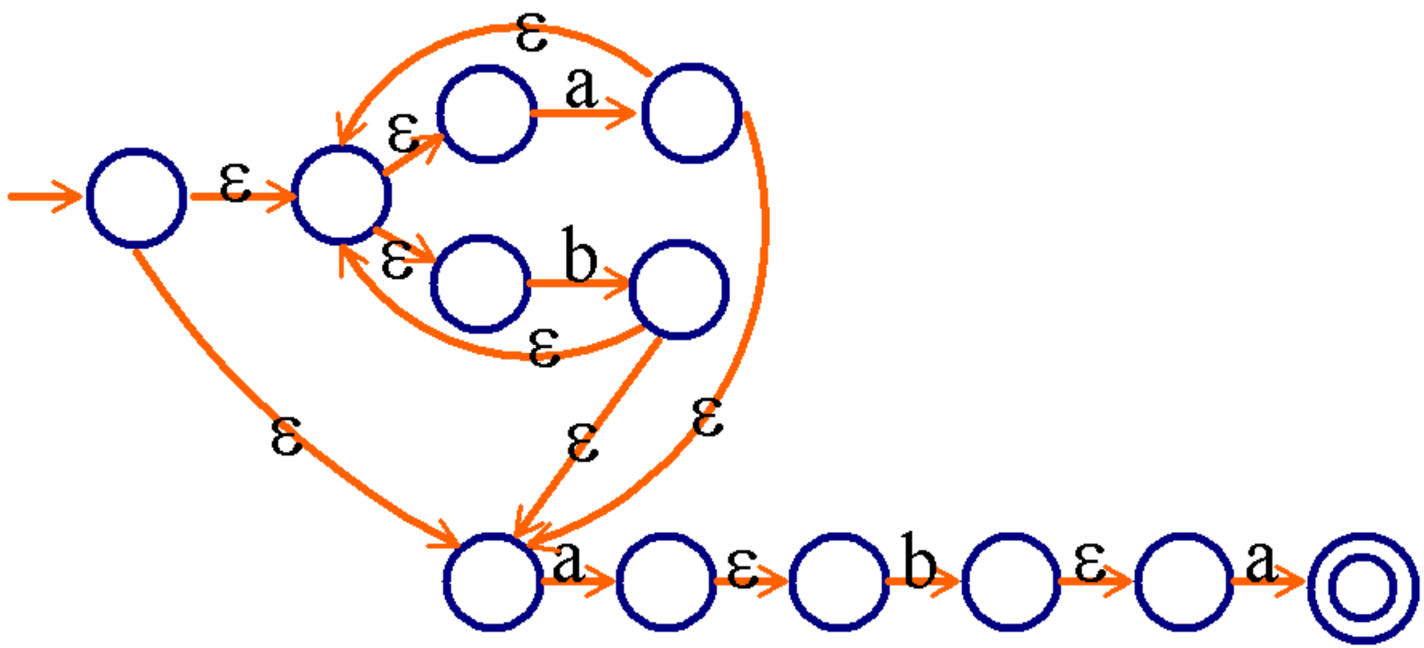


  - (a ∪ b)*



  - aba

– (a ∪ b)*aba

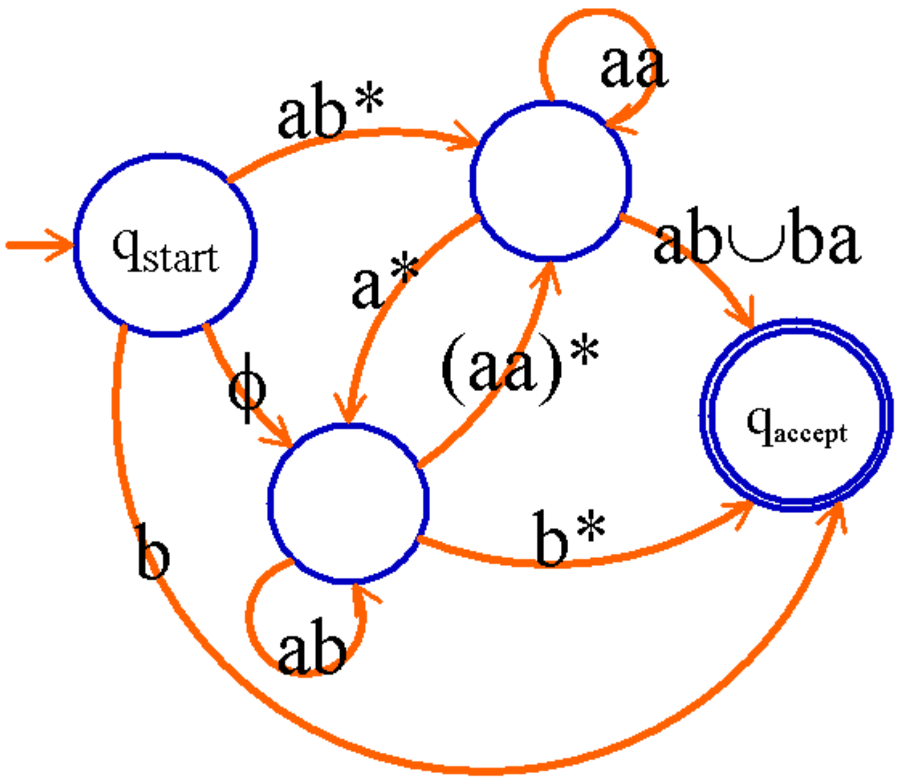


Generalized nondeterministic finite automaton (GNFA)

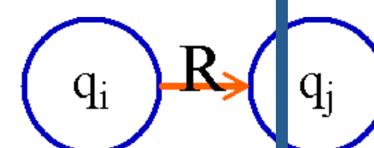
- (Q, Σ, δ, q_{start}, q_{accept})

  $\delta : (Q - \{q_{accept}\}) \times (Q - \{q_{start}\}) \rightarrow$ R

Set of all regular expressions over Σ



$\delta(q_i, q_j) = R$



- Lemma: (→)
  If a language is regular, then it is described by a regular expression
- Pf:
  a language A is regular
  →There is a DFA M accepting A
  - Goal: Convert DFAs into equivalent regular expressions.

– A GNFA accepts a string w in Σ^* if $w = w_1 w_2 \ldots w_k$, where each w_i is in Σ^* and a sequence of states $q_0 q_1 q_2 \ldots q_k$ exist, such that

- $q_0 = q_{start}$ is the start state
- $q_k = q_{accept}$ is the accept state
- for each i, we have $w_i \in L(R_i)$, where $q_i = \delta(q_{i-1}, R_i)$

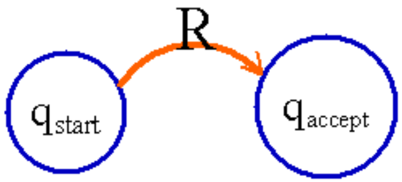
– DFA M --------------------→ GNFA G

$$\text{adding} \begin{cases} \text{start state} \\ \text{accept state} \\ \text{transition arrows} \end{cases}$$

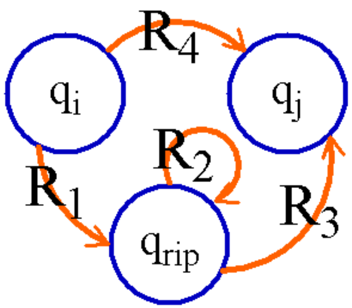
– Convert (G) : takes a GNFA and return an equivalent regular expression

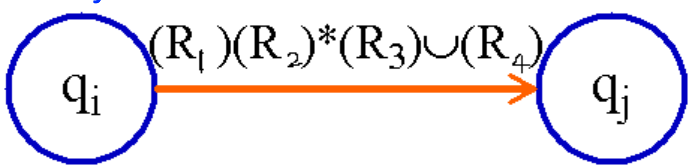
Convert(G)

- Convert(G)
  - Let k be the number of states of G
  - If k=2, return R



  - If k>2, select any $q_{rip} \in Q$ ($\neq q_{start}, q_{accept}$),
    Let G' = GNFA(Q', Σ, δ', q_{start}, q_{accept}),
    where $Q' = Q - \{q_{rip}\}$,
    and for any $q_i \in Q' - \{q_{accept}\}$ and any $q_j \in Q' - \{q_{start}\}$,
    let $\delta'(q_i, q_j) = (R_1)(R_2)^*(R_3) \cup (R_4)$

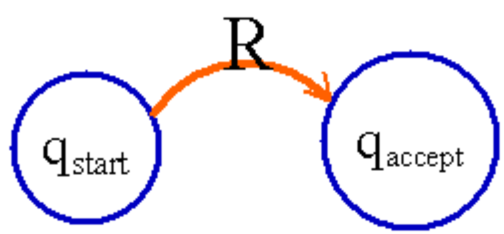




  - Compute CONVERT(G') and return this value

- Claim:
  For any GNFA G, CONVERT(G) is equivalent to G.

- Pf:
  By induction on k, the number of states of the GNFA
  - Basis: It is clear for k=2

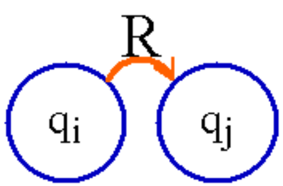


Induction step

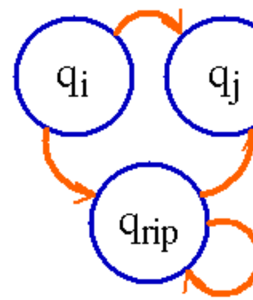
Assume that the claim is true for k-1 states

- Suppose G accepts an input w, Then in an accepting branch of the computation G enters a sequence of states: q_{start}, q_1, q_2, …, q_{accept}
    - If none of them is q_{rip}, G' accepts w.
    - If q_{rip} does appear in the above states, removing each run of consecutive q_{rip} states forms an accepting computation for G'
- Suppose G' accepts an input w

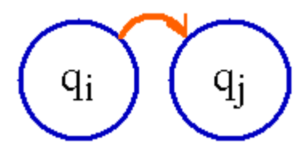
  G' :

  G:

  or

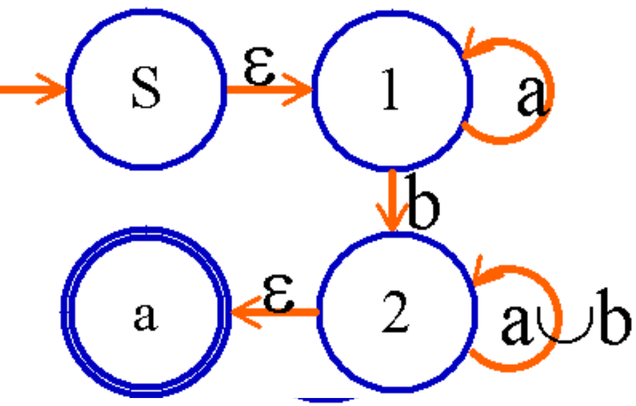
  

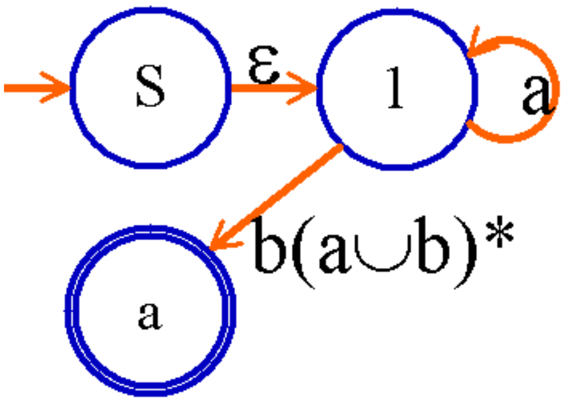
  $\Rightarrow$ G accepts w

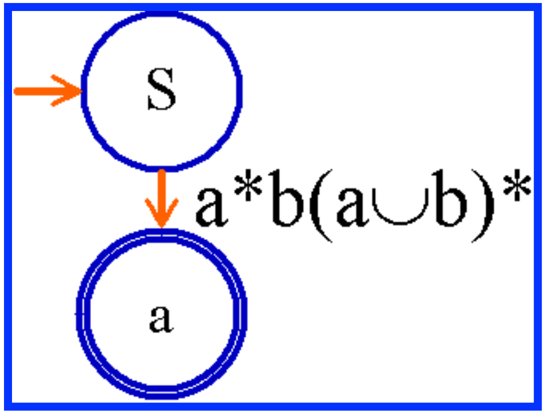
- G $\cong$ G' , by induction hypothesis, the claim is true

example

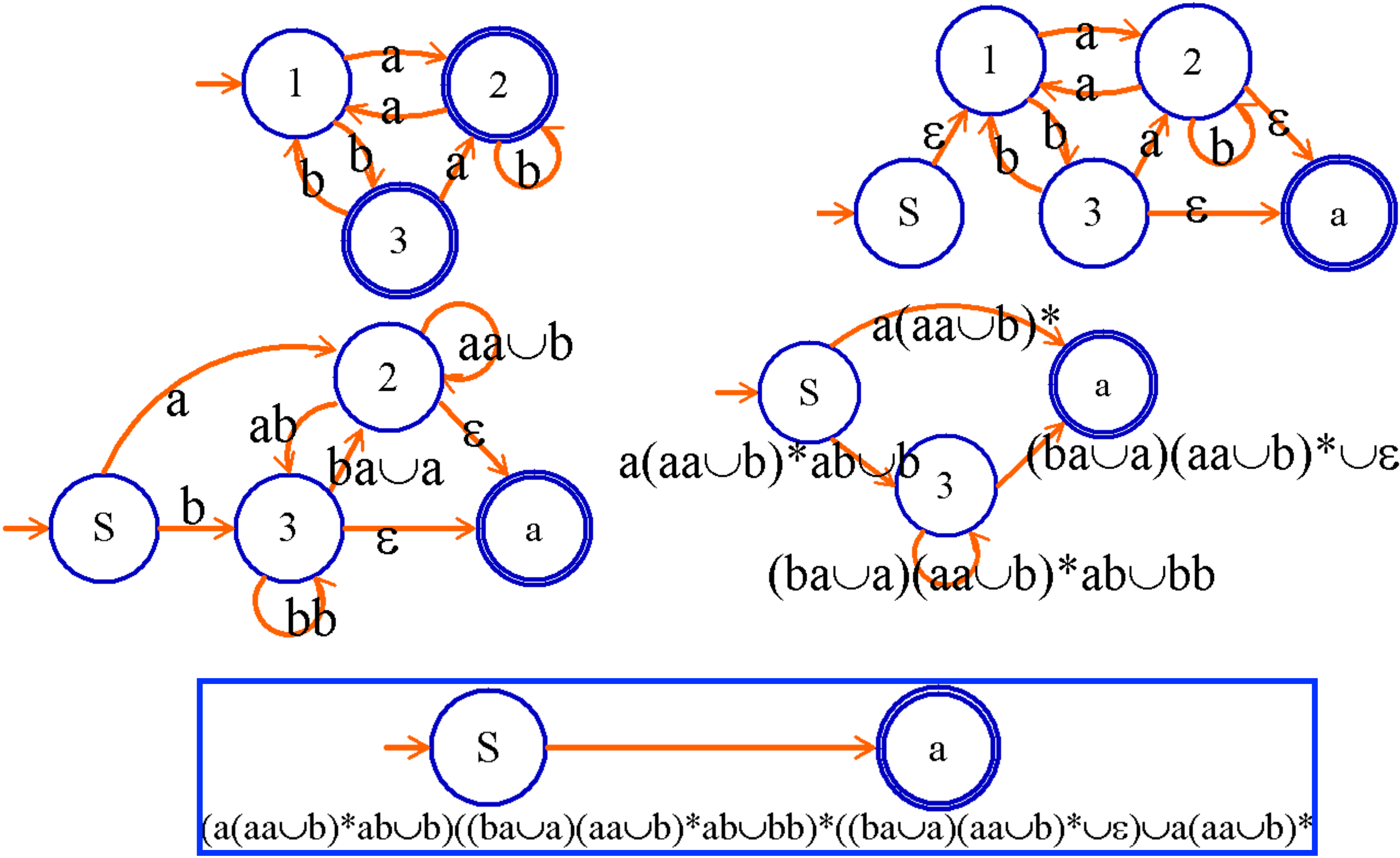
- E.g.:







example



1
2
3
a
a
b
b
a
b
S
ε
ε
ε
aa∪b
ab
ba∪a
bb
a(aa∪b)*
a(aa∪b)*ab∪b
(ba∪a)(aa∪b)*∪ε
(ba∪a)(aa∪b)*ab∪bb
(a(aa∪b)*ab∪b)((ba∪a)(aa∪b)*ab∪bb)*((ba∪a)(aa∪b)*∪ε)∪a(aa∪b)*

Arden's Rule

The set A^*B is the smallest language that is a solution for X in the linear equation

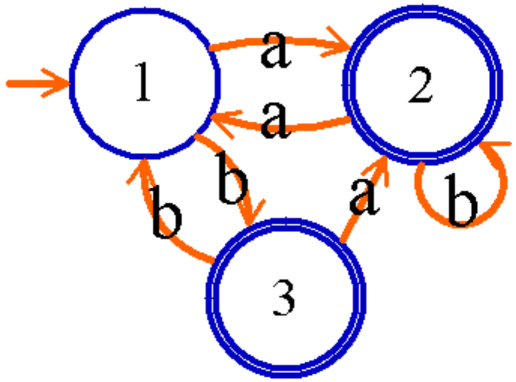
$$X = A \cdot X \cup B$$

where X, A, B are sets of strings. Moreover, if the set A does not contain the empty word, then this solution is unique.

Example

X_i := {all strings that reaching a final state from X_i}

We can write the following linear equations:



(1) $X_1 = aX_2 + bX_3$
(2) $X_2 = aX_1 + bX_2 + \varepsilon$ (where ε is the empty string)
(3) $X3 = bX_1 + aX_2 + \varepsilon$

The goal is to determine the regular expression for X_1.

From (1)+(2) we have

$X_2 = a(aX_2 + bX_3) + bX_2 + \varepsilon = (aa + b)X_2 + abX_3 + \varepsilon$

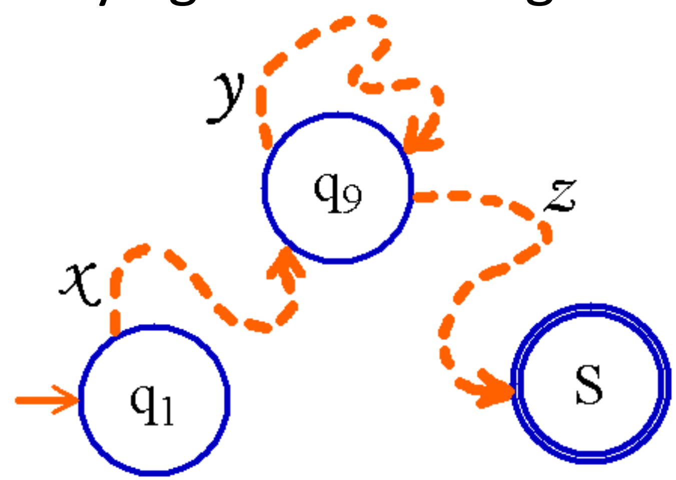
By Arden's Rule, we have

$X_2 = (aa + b)^*(abX_3 + \varepsilon) = (aa + b)^*abX_3 + (aa + b)^*$

Can you continue and get the final answer?

Pumping lemma for regular languages

- Is $\{0^n1^n: n\geq 0\}$ regular?
  How can we prove a language non-regular?
- Theorem: (Pumping Lemma)
  If A is a regular language, then there is a number $\mathcal{P}$ (the pumping length) where, if s is any string in A of length$\geq\mathcal{P}$, then s may be divided into 3 pieces, $s=xyz$, satisfying the following conditions:
  - for each $i\geq 0$, $xy^iz \in$A
  - $|y| > 0$
  - $|xy| \leq \mathcal{P}$



- proof
  - $M=(Q,\Sigma,\delta,q_1, F)$: a DFA recognizing A
  - $\mathcal{P}=|Q|$: number of state of M
  - $s = s_1 s_2 \ldots s_n \in A$, $n \geq \mathcal{P}$
  - $r_1, r_2, \ldots, r_{n+1}$: the seq. of states M enters while processing
  - $r_{i+1}=\delta(r_i, s_i)$ for $1 \leq i \leq n$. By pigeonhole principle, there must be 2 identical states, say $r_j=r_\ell$, $x= s_1 \ldots s_{j-1}$, $y=s_j \ldots s_{\ell-1}$, $z=s_\ell \ldots s_n$
  - x takes M from r_1 to r_j, y takes M from r_j to r_j, and z takes M from r_j to r_{n+1}

    M must accept $x y^i z$ for $i \geq 0$

    $\because j \neq \ell$, $|y|=|s_j \ldots s_{\ell-1}|>0$, $\ell \leq \mathcal{P}+1$, so $|xy| \leq \mathcal{P}$

- $B=\{0^n1^n : n\geq 0\}$ is not regular

proof:

- – Suppose B is regular
- – Let $\mathcal{P}$ be the pumping length
- – Choose $s = 0^{\mathcal{P}}1^{\mathcal{P}} = 0\ldots01\ldots1 \in B$
- – Let $s = xyz$, By pumping lemma, for any $i \geq 0$ $xy^iz \in B$. But
    - 0…001…1 : y has 0 only $\Rightarrow$ $\rightarrow\leftarrow$
    - 0…01…1 : y has 1 only $\Rightarrow$ $\rightarrow\leftarrow$
    - 0…01…1 : y has both 0 and 1 $\Rightarrow$ $xyyz \notin B$

- C={w | w has an equal number of 0's and 1's} is not regular.

proof: (By pumping lemma)

- Let $\mathcal{P}$ be the pumping length, Suppose C is regular.
- Let $s = 0^{\mathcal{P}}1^{\mathcal{P}} \in$ C, By pumping lemma, s can be split into 3 pieces, $s = xyz$, and $xy^iz \in$ C for any $i \geq 0$
- By condition 3 in the lemma: $|xy| \leq \mathcal{P}$
  Thus y must have only 0s.
  Then $xyyz \notin$ C

proof: (Not by pumping lemma)

- Suppose C is regular
- It is clear that 0*1* is regualr
- Then $C \cap 0^*1^* = \{0^n1^n : n \geq 0\}$ is regular

  $\rightarrow\leftarrow$

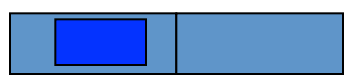
- F={ww | $w\in\{0,1\}^*$ } is non-regular.

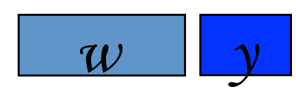
proof:

  - Suppose F is regular.
  - Let $\mathcal{P}$ be the pumping length given by the pumping lemma
  - Let $s = 0^{\mathcal{P}}10^{\mathcal{P}}1 \in$ F
  - Split s into 3 pieces, $s = xyz$
  - By condition 3 in the lemma: $|xy| \leq \mathcal{P}$

    Thus y must have 0 only.

    $\Rightarrow xyyz \notin$ F   0…010…01 $\rightarrow\leftarrow$





- E=$\{0^i 1^j : i > j\}$ is non-regular.

proof:

  - Assume E is regular.
  - Let $\mathcal{P}$ be the pumping length.
  - Let $s = 0^{\mathcal{P}+1} 1^{\mathcal{P}} \in$ E.
  - Split s into 3 pieces, $s = xyz$
  - By pumping lemma: $xy^i z \in$ E for any $i \geq 0$
    $|y| > 0$, y have 0 only. $xz \in$ E.
    But xz has $\#(0) \leq \#(1)$

- D=$\{1^{n^2} : n \geq 0\}$ is not regular.

proof:

  - Assume D is regular.
  - Let $\mathcal{P}$ be the pumping length.
  - Let $s = 1^{\mathcal{P}^2} \in$ D
  - Split s into 3 pieces, $s = xyz \Rightarrow xy^iz \in$ D, $i \geq 0$
  - Consider $xy^iz \in$ D and $xy^{i+1}z \in$ D
    $\Rightarrow |xy^iz|$ and $|xy^{i+1}z|$ are perfect square for any $i \geq 0$
  - If m=n^2, (n+1)^2 - n^2 =2n+1 = $2\sqrt{m}+1$

- Let m=$|x y^i z|$
- $|y| \leq |s| = \mathcal{P}^2$
- Let $i = \mathcal{P}^4$

$|y| = |x y^{i+1} z| - |x y^i z|$

$\leq \mathcal{P}^2 = (\mathcal{P}^4)^{1/2}$

$< 2(\mathcal{P}^4)^{1/2}+1$

$\leq 2(|x y^i z|)^{1/2}+1$

$= 2\sqrt{m} +1$

$\rightarrow\leftarrow$