Formal Languages and Automata Theory(18IS54)

Unit 1

FORMAL LANGUAGES AND AUTOMATA THEORY

Course Code	18IS54	**Credits**	4
Course type	PC	**CIE Marks**	50 marks
Hours/week: L-T-P	4-0-0	**SEE Marks**	50 marks
Total Hours:	40	**SEE Duration**	3 Hours for 100 marks

Book

1. John E. Hopcroft, Rajeev Motwani, Jeffrey D. Ullman, "Introduction to Automata Theory, Languages and Computation", Pearson Education, 3/E, 2013.
2. John R. Levine and Tony Mason and Doug Brown, Lex and Yacc, "UNIX programming tools", 2/E, 2012.
3. S . P. Euguene Xavier "Theory of Automata , Formal Languages and Computation ", 5 / E 2011.

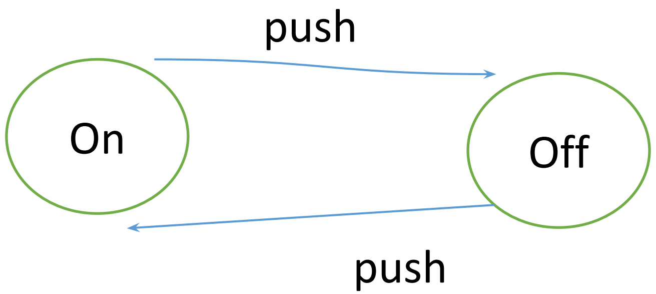
Reference Books

1. Alfred V Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman , "Compilers *Principles, Techniques and Tools",* Pearson Education , 2 / E,2008
2. Peter Linz, "An Introduction to Formal Languages and Automata",Narosa Publishing House, 5/E, 2011.

Automata Theory

- It is study of abstract computing devices.
- Finite Automaton are useful models.
- In finite automata components are viewed as finite number of states.

  Ex: on/off switch



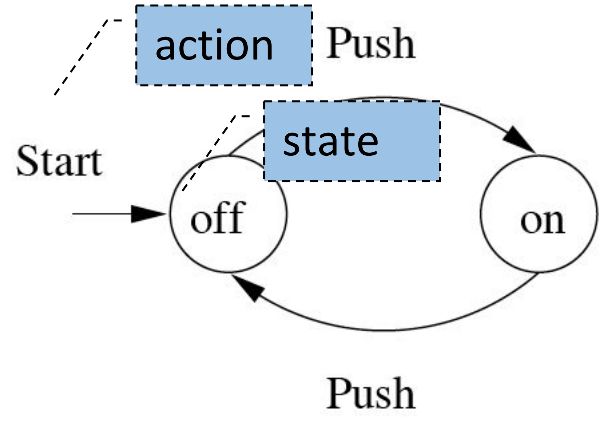
- Finite automata which is part of lexical analyser the job may be to recognize the keyword

Finite Automata

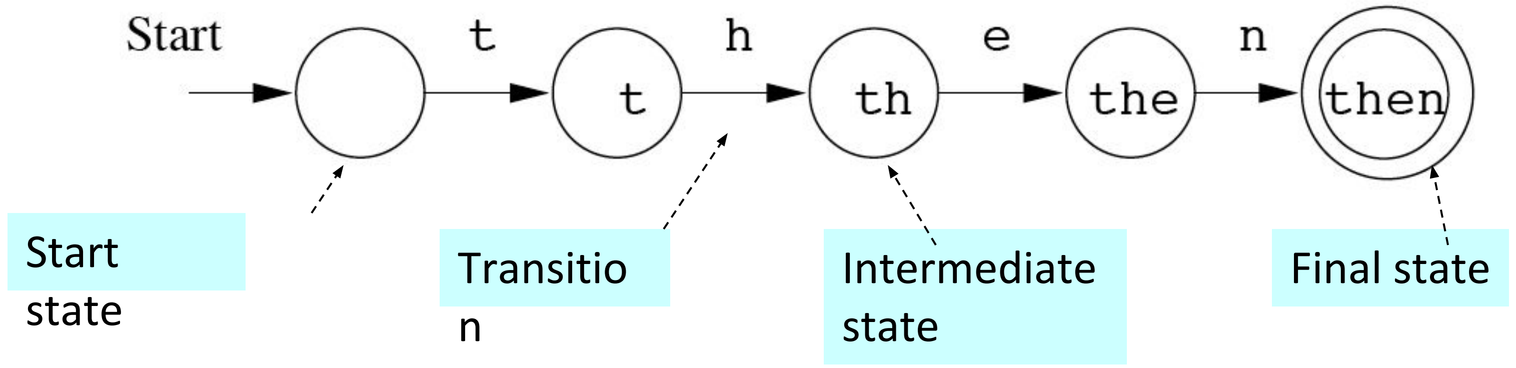
- Some Applications
  - Software for designing and checking the behavior of digital circuits
  - Lexical analyzer of a typical compiler
  - Software for scanning large bodies of text (e.g., web pages) for pattern finding
  - Software for verifying systems of all types that have a finite number of states (e.g., stock market transaction, communication/network protocol)

Finite Automata : Examples

- On/Off switch



- Modeling recognition of the word "*then*"



Automata Theory

- Grammars are useful models.

  Ex: parser: E => E + E
- Regular Expression : [A-Z][a-z]*[ ]
- Two views decidability and intractability
- Concepts of automata theory

  alphabet

  strings

  power of an alphabet

  languages

Alphabet

An alphabet is a finite, non-empty set of symbols

- We use the symbol $\sum$ (sigma) to denote an alphabet
- Examples:
  - Binary: $\sum = \{0,1\}$
  - All lower case letters: $\sum = \{a,b,c,..z\}$
  - Alphanumeric: $\sum = \{a\text{-}z, A\text{-}Z, 0\text{-}9\}$
  - …

Strings

A string or word is a finite sequence of symbols chosen from ∑

- ***Empty string is ε (or "epsilon")***

- Length of a string *w,* denoted by "|*w*|", is equal to the *number of (non-ε) characters in the string*
  - *E.g., x = 010100* *|x| = 6*
  - *x = 01 ε 0 ε 1 ε 00 ε* *|x| = ?*

- *xy* = concatentation of two strings *x* and *y*

Powers of an alphabet

Let ∑ be an alphabet.

- $\sum^{k}$ = the set of all strings of length k $\sum=\{0,1\}$ $\sum^2 = \{00,11,10,01\}$
- $\sum^{*} = \sum^{0} \cup \sum^{1} \cup \sum^{2} \cup \ldots$
- $\sum^{+} = \sum^{1} \cup \sum^{2} \cup \sum^{3} \cup \ldots$

Languages

*L is a said to be a language over alphabet ∑, only if L ⊆ ∑**

 this is because ∑* is the set of all strings (of all possible length including 0) over the given alphabet ∑

Examples:

1. Let L be *the* language of all strings consisting of *n* 0's followed by *n* 1's:
   L = {ε, 01, 0011, 000111,…}
2. Let L be *the* language of all strings of with equal number of 0's and 1's:
   L = {ε, 01, 10, 0011, 1100, 0101, 1010, 1001,…}

Ø denotes the Empty language

- Let L = {ε}; Is L=Ø

Finite Automaton (FA)

- Informally, a state diagram that comprehensively captures all possible states and transitions that a machine can take while responding to a stream or sequence of input symbols
- Recognizer for "Regular Languages"

- Deterministic Finite Automata (DFA)
  - The machine can exist in only one state at any given time
- Non-deterministic Finite Automata (NFA)
  - The machine can exist in multiple states at the same time

DFA Deterministic Finite Automata

A Deterministic Finite Automata denoted by 5-tuple (Q, ∑,δ,$q_{0,}$ F) consists of

- The set Q which is simply a set with a finite number of states. Its states can, however,be interpreted as a state that the system (automaton) is in.
- ∑ is finite set of input symbols
- The transition function is also called a next state function meaning that the automaton moves into the state δ(p, a) =q if it receives the input symbol a while in state p.
- q_0 is the start state.
- F is set of final or accepting states and is subset of Q.

Deterministic Finite Automata - Definition

- A Deterministic Finite Automaton (DFA) consists of:
  - Q ==> a finite set of states
  - ∑ ==> a finite set of input symbols (alphabet)
  - δ ==> a transition function, which is a mapping between Q x ∑ ==> Q
  - q_0 ==> a start state
  - F ==> set of accepting states
- A DFA is defined by the 5-tuple:
  - (Q, ∑,δ,$q_{0,}$ F)

What does a DFA do on reading an input string?

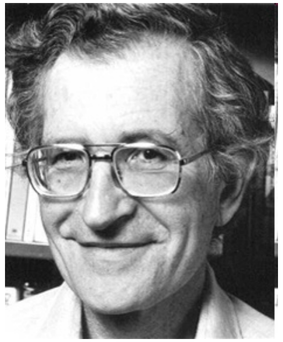
- Input: a word w in Σ^*
- Question: Is w acceptable by the DFA?
- Steps:
  - Start at the "start state" q_0
  - For every input symbol in the sequence w do
    - Compute the next state from the current state, given the current input symbol in w and the transition function
  - If after all symbols in w are consumed, the current state is one of the accepting states (F) then *accept w;*
  - Otherwise, *reject w.*

Regular Languages

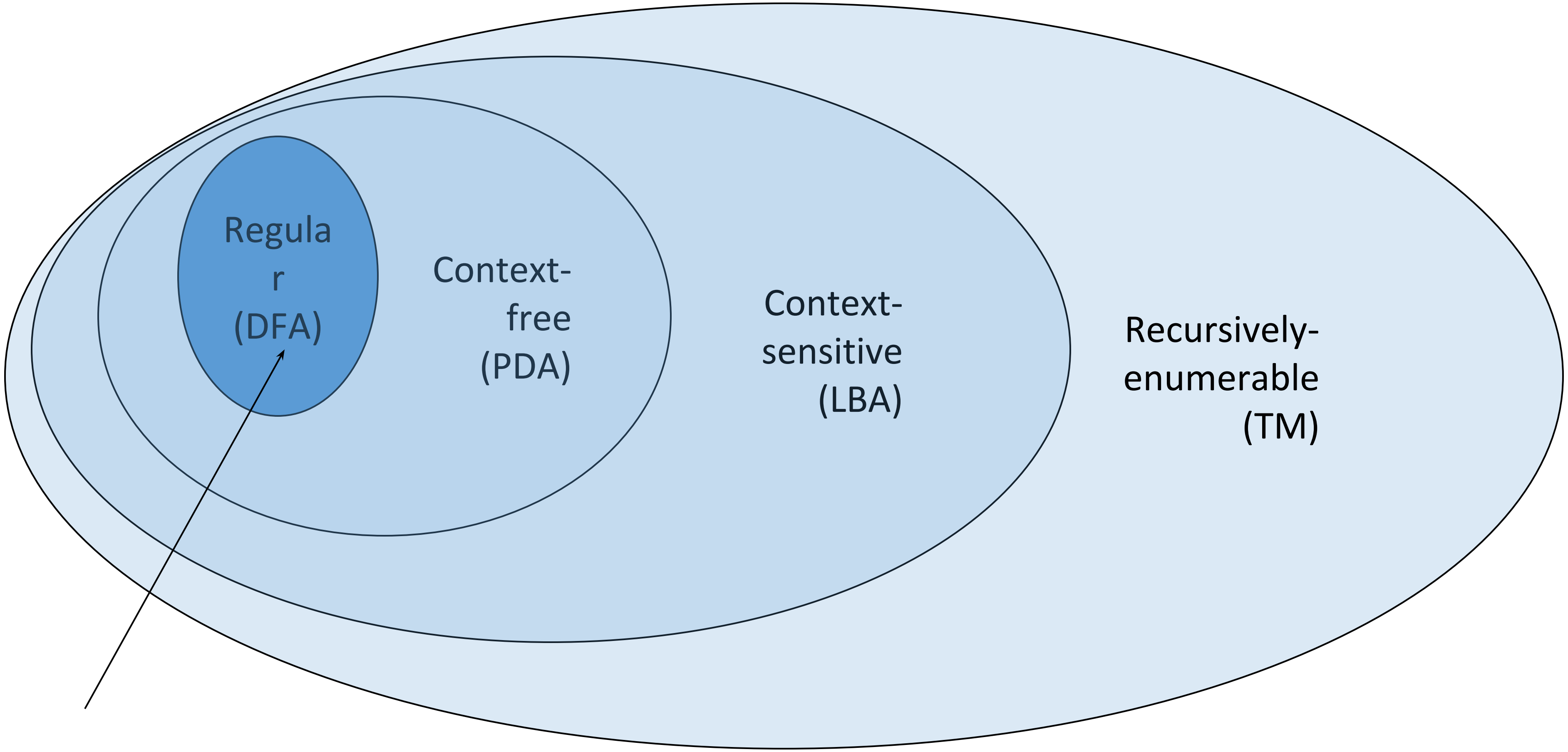
- Let L(A) be a language *recognized* by a DFA A.
  - Then L(A) is called a "*Regular Language*".

- Locate regular languages in the Chomsky Hierarchy

The Chomsky Hierachy



- A containment hierarchy of classes of formal languages



Example #1

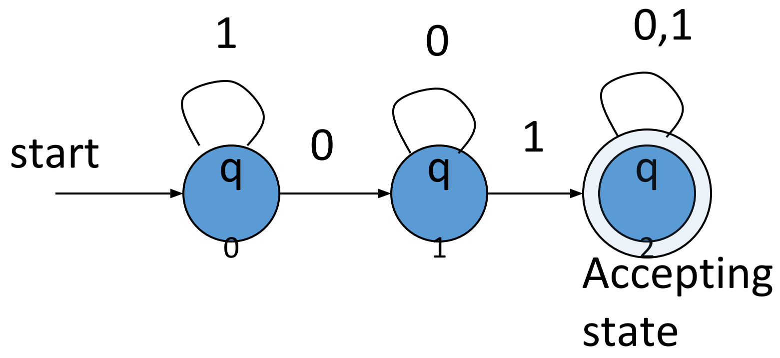
- Design a DFA that accepts all and only the strings of 0's and 1's that have the sequence 01 somewhere in the string.
- Build a DFA for the following language:
  - L = {w | w is a binary string of the form x01y for some strings x and y consisting of 0's and 1's}
- Steps for building a DFA to recognize L:
  - $\sum = \{0,1\}$
  - Decide on the states: Q
  - Designate start state and final state(s)
  - δ: Decide on the transitions:
- "Final" states == same as "accepting states"
- Other states == same as "non-accepting states"

x01y consisting of 0s and 1s

DFA for strings containing 01

1001 q_0-1>q_0-0>q_1-0>q_1-1>q_2

- What makes this DFA deterministic?



- What if the language allows empty strings?

- $Q = \{q_0, q_1, q_2\}$
- $\sum = \{0,1\}$
- start state = q_0
- $F = \{q_2\}$
- Transition table

symbols

δ	**0**	**1**
→ **q_0**	q_1	q_0
q_1	q_1	q_2
***q_2**	q_2	q_2

states

DFA

Example: Design a DFA to accept string of a's and b's having a substring aa

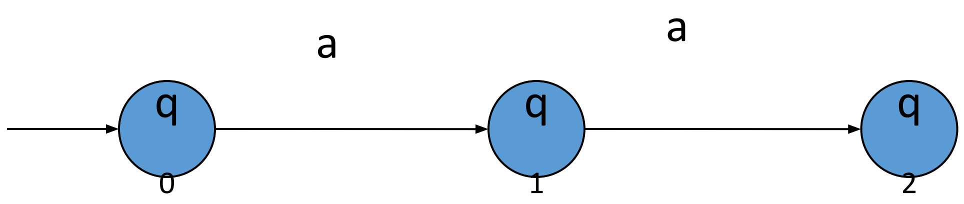
M= (Q, $\sum,\delta,q_{0,}$ F)

Q={$q_{0,}$ $q_{1,}$ q_2 }

$\sum$ = {a,b}

q_0 is the start state

F = {q_2 }



DFA

Example: Design a DFA to accept string of a's and b's having a substring aa

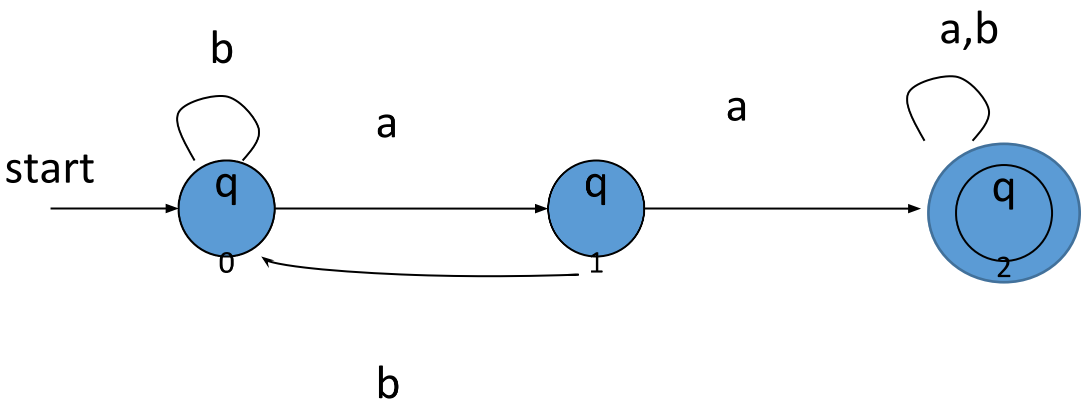
$M = (Q, \Sigma, \delta, q_0, F)$

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

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

q_0 is the start state

$F = \{q_2\}$



- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{a,b\}$
- start state = q_0
- $F = \{q_2\}$
- Transition table

symbols

δ	a	b
$\rightarrow q_0$	q_1	q_0
q_1	q_2	q_0
$*q_2$	q_2	q_2

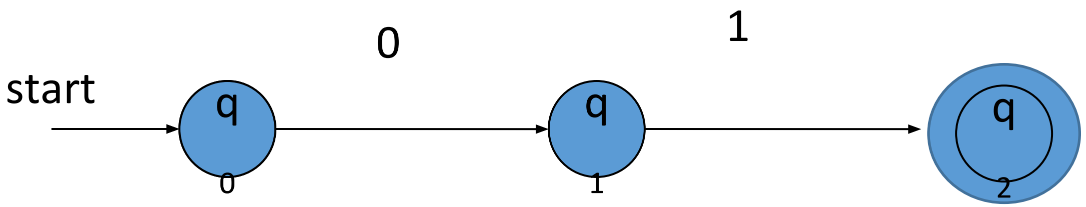
states

DFA

Example: Design a DFA to accept the language consisting of 0's and 1's and

L={w|w is of even length and begins with 01 }

M= (Q, ∑,δ,$q_{0,}$ F)

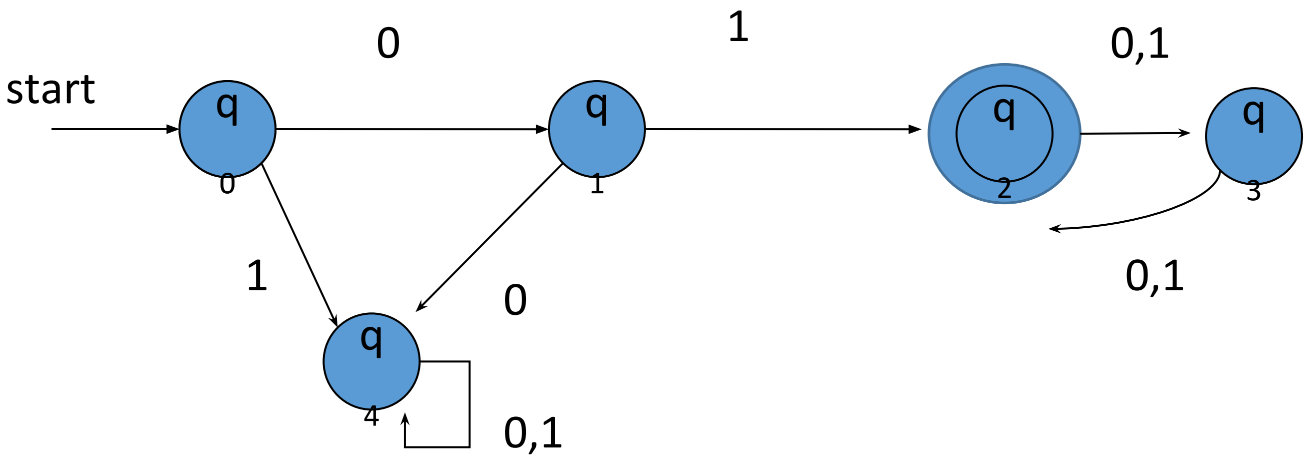


DFA

Example: Design a DFA to accept the language consisting of 0's and 1's and

$L=\{w|w \text{ is of even length and begins with } 01\}$

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



- $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- $\Sigma = \{0,1\}$
- start state = q_0
- $F = \{q_2\}$
- Transition table

symbols

δ (states)	0	1
→ q_0	q_1	q_4
q_1	q_4	q_2
$*q_2$	q_3	q_3
q_3	q_2	q_2
q_4	q_4	q_4

DFA

Example: Design a DFA to accept string of a's and b's having eactly one a

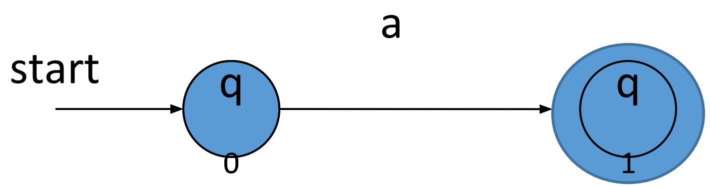
M= (Q, $\sum$,δ,$q_{0,}$ F)

Q={$q_{0,}$ $q_{1,}$ q_2 }

$\sum$ = {a,b}

q_0 is the start state

F = {q1 }



DFA

Example: Design a DFA to accept string of a's and b's having eactly one a

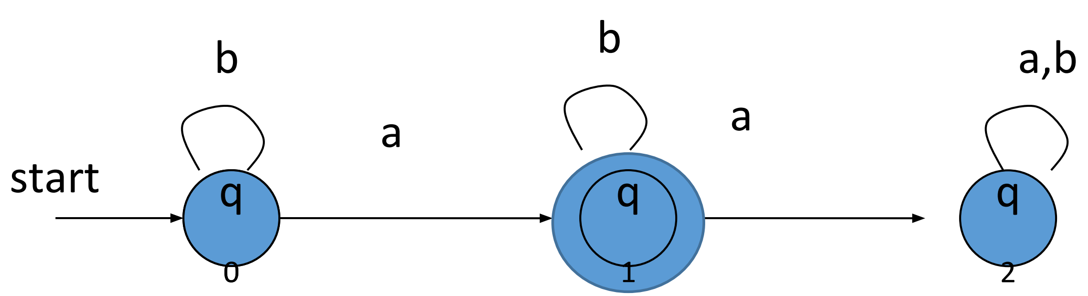
M= (Q, $\sum$,δ,$q_{0,}$ F)

Q={$q_{0,}$ $q_{1,}$ q_2 }

$\sum$ = {a,b}

q_0 is the start state

F = {q1 }



- Q = {q_0,q_1,q_2}
- $\sum$ = {a,b}
- start state = q_0
- F = {q1}
- Transition table

symbols

δ	a	b
q_0	q_1	q_0
*q_1	q_2	q_1
q_2	q_2	q_2

states

NFA

- Like DFA NFA has finite set of states, finite set of input symbols, one start state and a set of accepting states.
- Difference between DFA and NFA is δ function in NFA it takes a state and input symbol as arguments but returns a set of zero, one or more states.

A Non Deterministic Finite Automaton (NFA) consists of:

Q ==> a finite set of states

$\sum$ ==> a finite set of input symbols (alphabet)

δ ==> a transition function, which is a mapping between $Q \times \sum$ ==> Q

q_0 ==> a start state

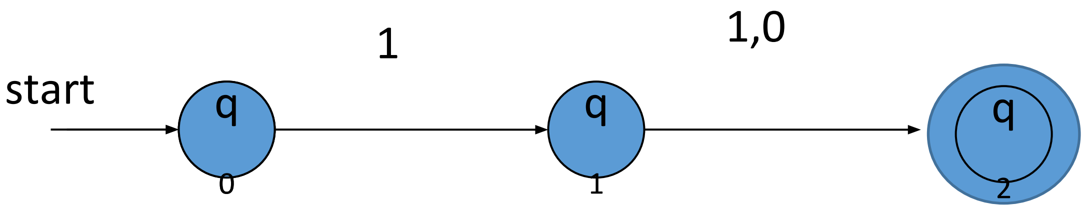
F ==> set of accepting states

A NFA is defined by the 5-tuple:

$(Q, \sum,\delta,q_{0,} F)$

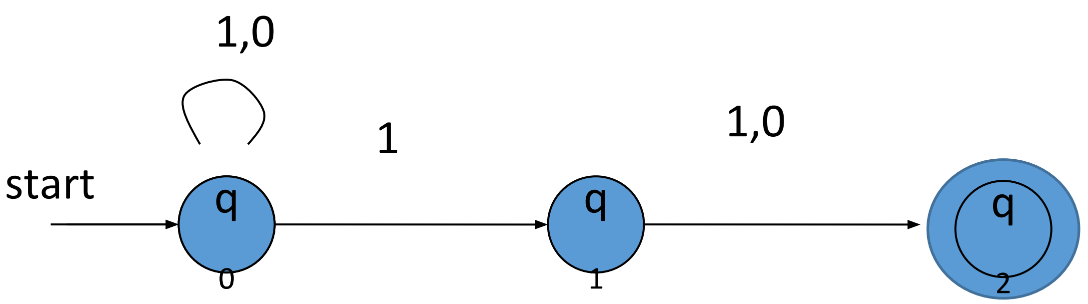
NFA

- Design a NFA which accepts exactly those strings that have the symbol 1 in second last position.



NFA

- Design a NFA which accepts exactly those strings that have the symbol 1 in second last position.



- $Q = \{q_0, q_1, q_2\}$
- $\Sigma = \{0,1\}$
- start state = q_0
- $F = \{q_2\}$
- Transition table

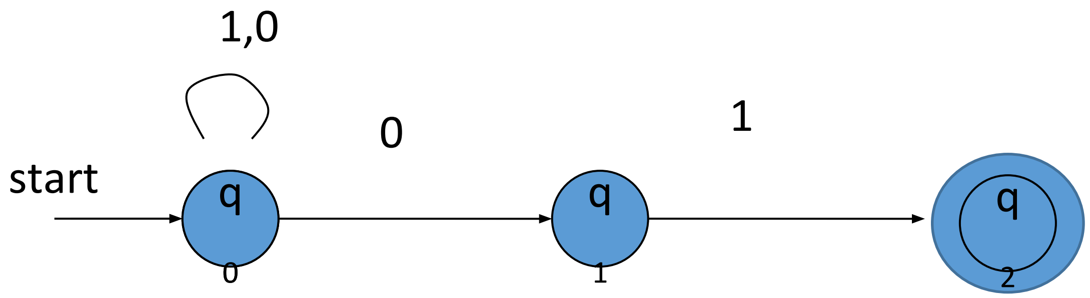
symbols

states

δ	**0**	**1**
→ **q_0**	q_0	$\{q_0, q_1\}$
q_1	q_2	q2
***q_2**	$\emptyset$	$\emptyset$

NFA

- Design a NFA where L={w|w ends in 01}.



- $Q = \{q_0, q_1, q_2\}$
- $\sum = \{0,1\}$
- start state = q_0
- $F = \{q_2\}$
- Transition table

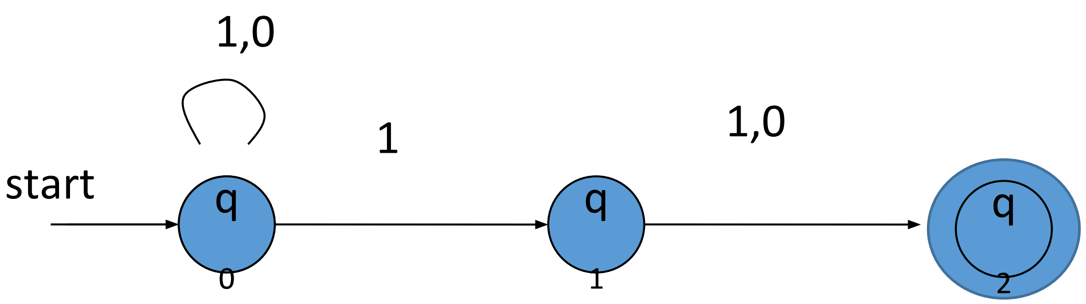
symbols

δ	**0**	**1**
→ q_0	$\{q_0, q_1\}$	q_0
q_1	Ø	q_2
$*q_2$	Ø	Ø

states

Convert NFA to DFA

- Design a NFA which accepts exactly those strings that have the symbol 1 in second last position.



- $Q = \{q_0,q_1,q_2\}$
- $\sum = \{a,b\}$
- start state = q_0
- F = {q1}
- Transition table

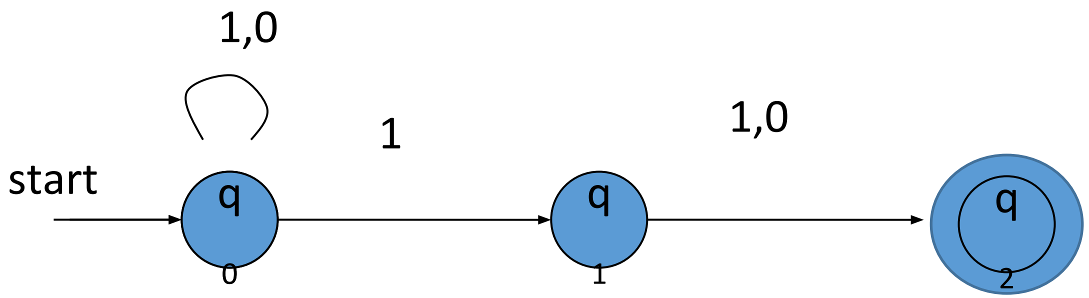
symbols

states

δ	0	1
$\rightarrow q_0$	q_0	$\{q_0, q_1\}$
q_1	q_2	q_2
$*q_2$	Ø	Ø

Converting NFA to DFA

- Design a NFA which accepts exactly those those strings that have the symbol 1 in second last position.

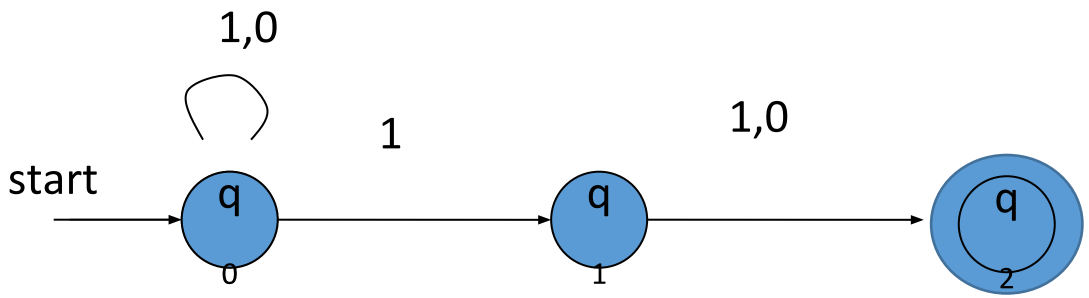


states

δ	0	1
→ q_0	q_0	$\{q_0, q_1\}$
q_0, q_1	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$
$q_0 q_2$		

Converting NFA to DFA

- Design a NFA which accepts exactly those those strings that have the symbol 1 in second last position.

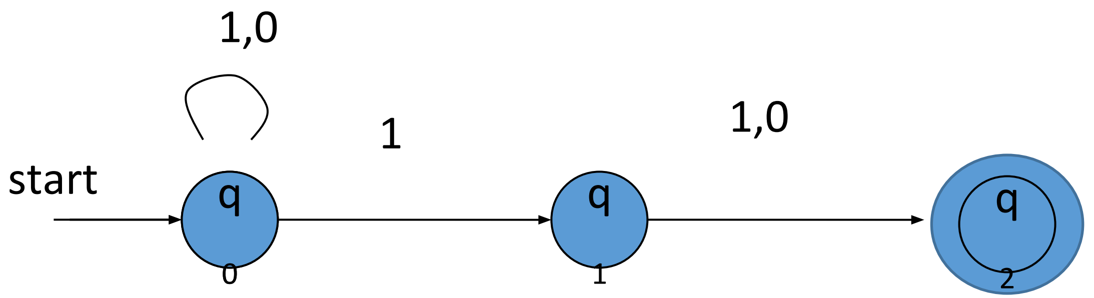


states

δ	**0**	**1**
→ **q_0**	q_0	$\{q_0, q_1\}$
q_0, q_1	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$
$*q_0 q_2$	q_0	$\{q_0, q_1\}$
$*\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$

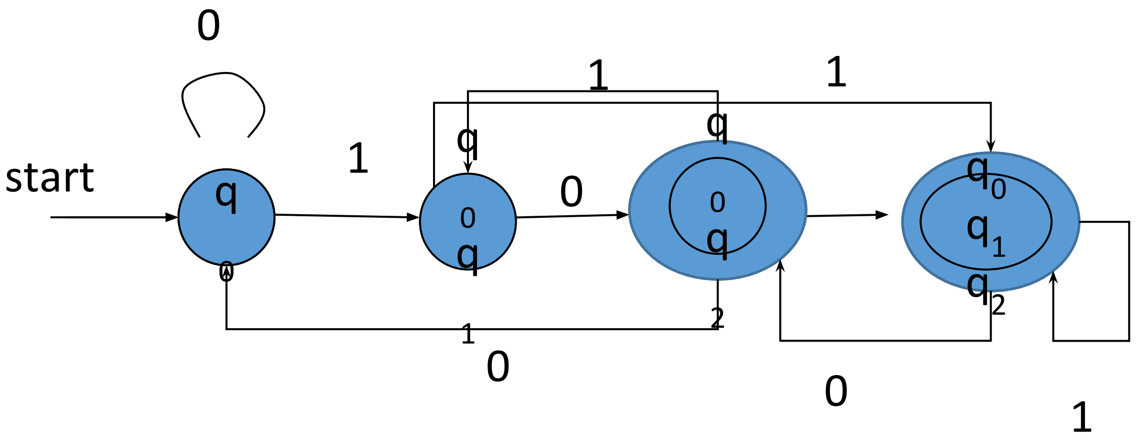
Converting NFA to DFA

- Design a NFA which accepts exactly those those strings that have the symbol 1 in second last position.



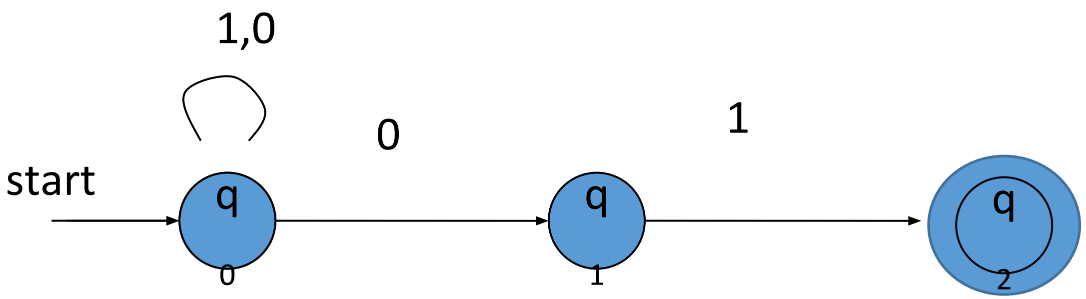
δ	0	1
→ q_0	q_0	$\{q_0, q_1\}$
q_0, q_1	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$
*$q_0 q_2$	q_0	$\{q_0, q_1\}$
*$\{q_0, q_1, q_2\}$	$\{q_0, q_2\}$	$\{q_0, q_1, q_2\}$

states



Converting NFA to DFA

- Design a NFA where L={w|w ends in 01}.



symbol

δ	0	1
→ q_0	$\{q_0, q_1\}$	q_0
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$

states

- Q = $\{q_0, q_1, q_2\}$
- ∑ = {0,1}
- start state = q_0
- F = $\{q_2\}$
- Transition table

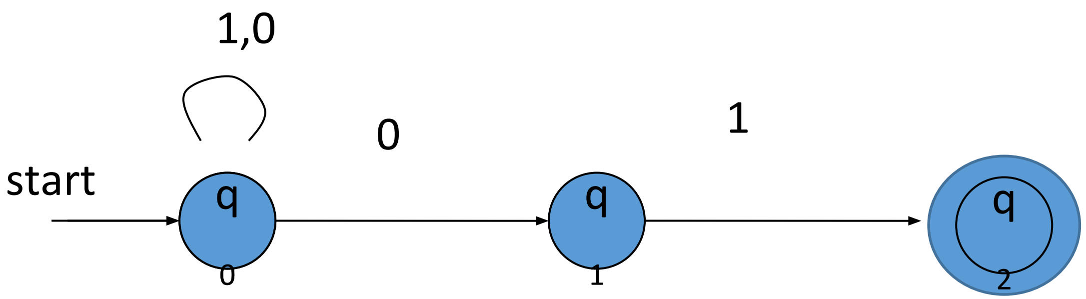
symbols

δ	0	1
→ q_0	$\{q_0, q_1\}$	q_0
q_1	Ø	q_2
*q_2	Ø	Ø

states

Converting NFA to DFA

- Design a NFA where L={w|w ends in 01}.



symbol

δ	0	1
→ q_0	$\{q_0, q_1\}$	q_0
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$*\{q_0, q_2\}$	$\{q_0, q_1\}$	q_0

states

- $Q = \{q_0, q_1, q_2\}$
- $\sum = \{0,1\}$
- start state = q_0
- $F = \{q_2\}$
- Transition table

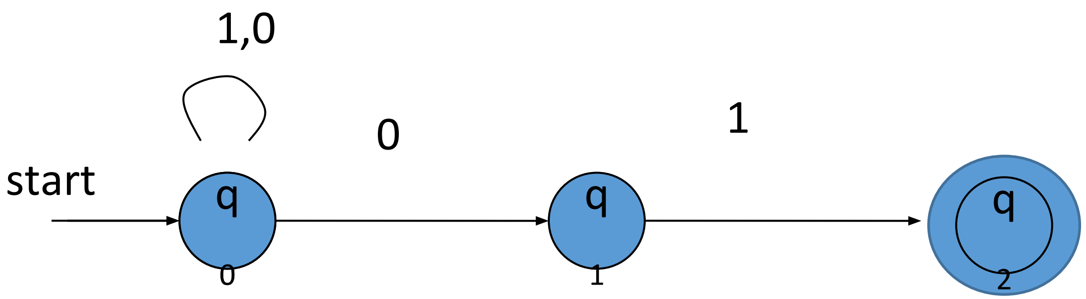
symbols

δ	0	1
→ q_0	$\{q_0, q_1\}$	q_0
q_1	Ø	q_2
$*q_2$	Ø	Ø

states

Converting NFA to DFA

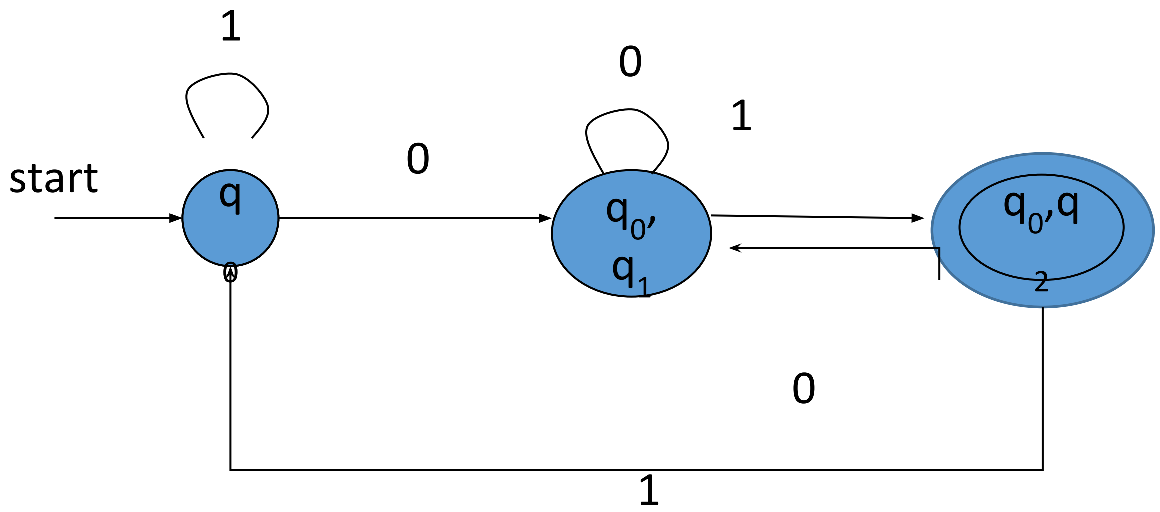
- Design a NFA where L={w|w ends in 01}.



symbol

states

δ	**0**	**1**
$\mathbf{q_0}$	$\{q_0, q_1\}$	q_0
$\{q_0, q_1\}$	$\{q_0, q_1\}$	$\{q_0, q_2\}$
$*\{q_0, q_2\}$	$\{q_0, q_1\}$	q_0



ε-NFA

ε-NFA is represented by A = (Q, ∑,δ,$q_{0,}$ F)

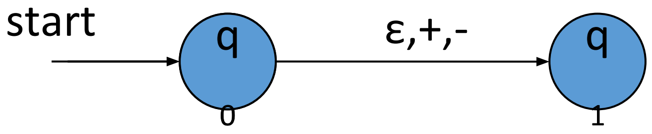
δ is now a function that takes as arguments
1. A state in Q and
2. A member of ∑ U {ε} either an input symbol or the symbol ε

With this feature a transition is allowed on ε the empty string, ε contributes nothing to the string along the path.

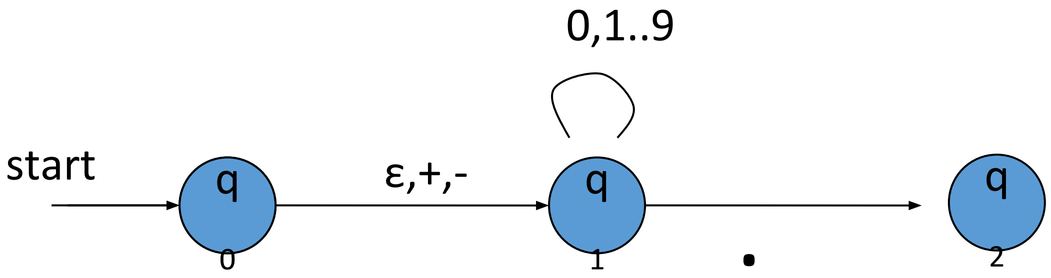
Ex: Design an ε-NFA that accepts decimal numbers consisting of

1. an optional + or - sign
2.a string of digits
3.a decimal point
4. Another string of digits. Either this string of digits or the string in 2 can be empty but atleast one of the two strings of digits must be non empty.

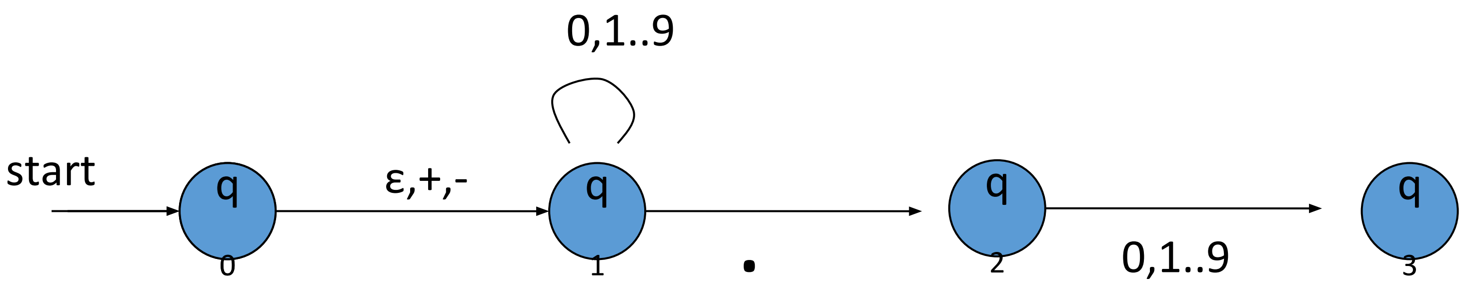
ε-NFA



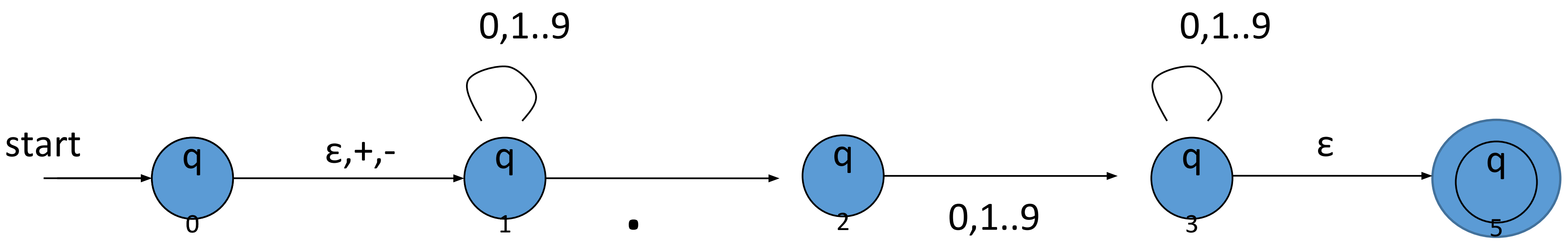
ε-NFA



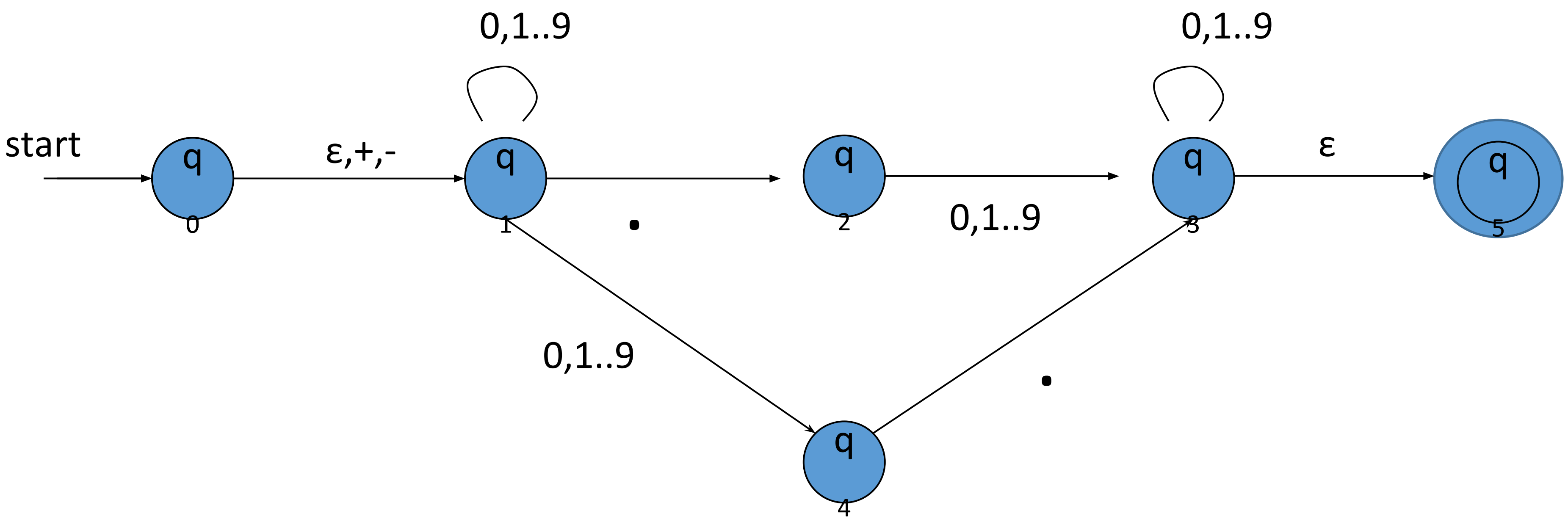
ε-NFA



ε-NFA



ε-NFA



ε-NFA

symbol

states

δ	ε	+,-	.	0,1, ...9
→ q_0	$\{q_1\}$	q1	∅	∅
q_1	∅	∅	$\{q_2\}$	$\{q_1,q_4\}$
q_2	∅	∅	∅	$\{q_3\}$
q_3	$\{q_5\}$	∅	∅	$\{q_3\}$
q_4	∅	∅	$\{q_3\}$	∅
$*q_5$	∅	∅	∅	∅

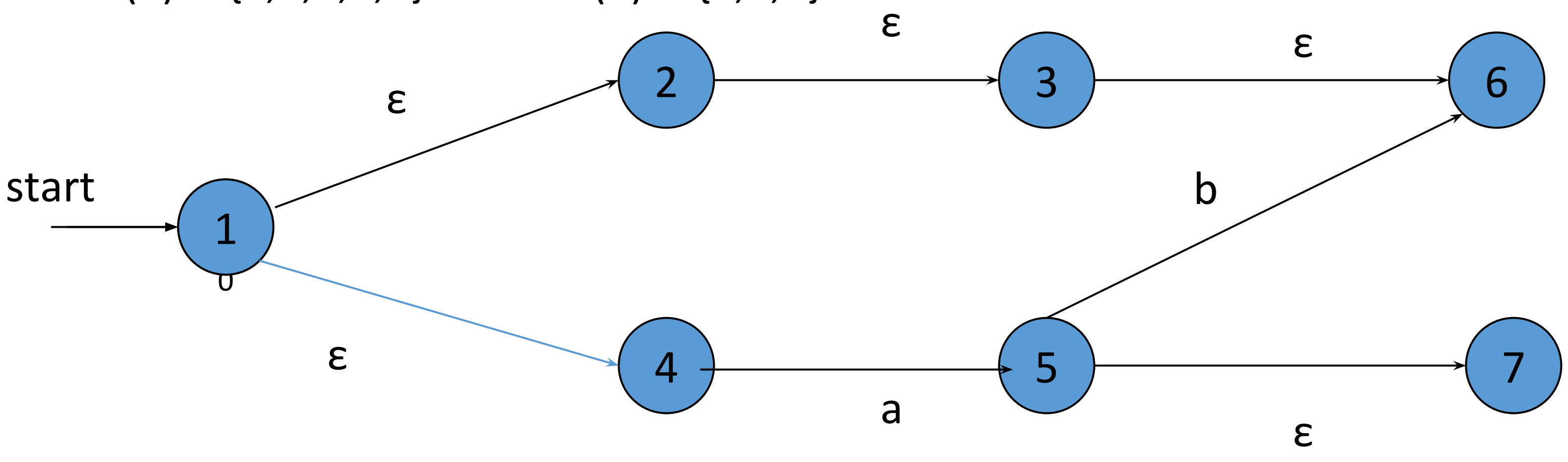
ε-closures

ECLOSE:

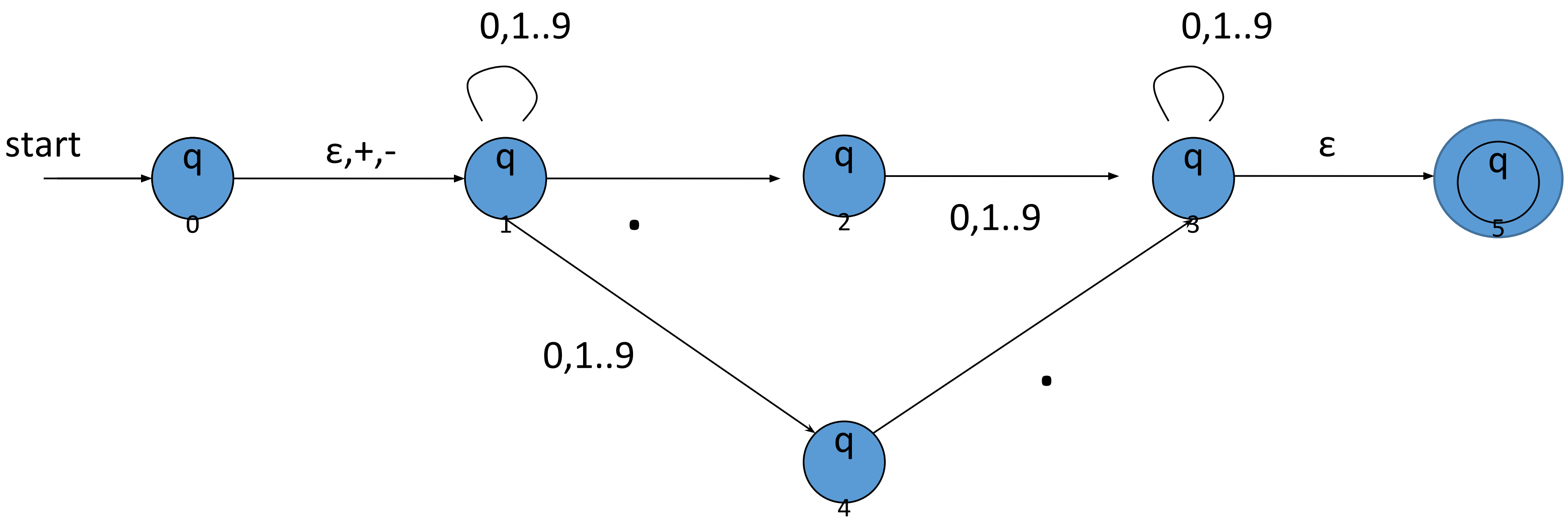
ECLOSE(q_0) in previous example is {q0,q1}

ECLOSE(q_3) is {q3,q5}

ECLOSE(1) is {1,2,3,4,6} ECLOSE(2) is {2,3,6}



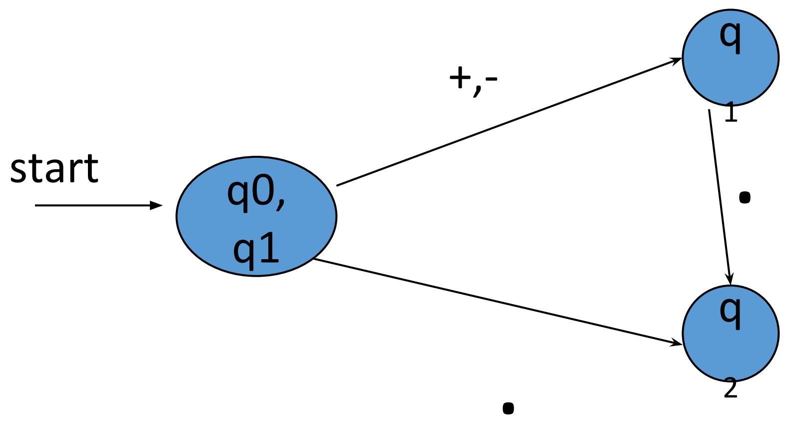
ε-NFA



Eliminating ε transistions

Eliminating ε transitions results in DFA that accepts the same language as E

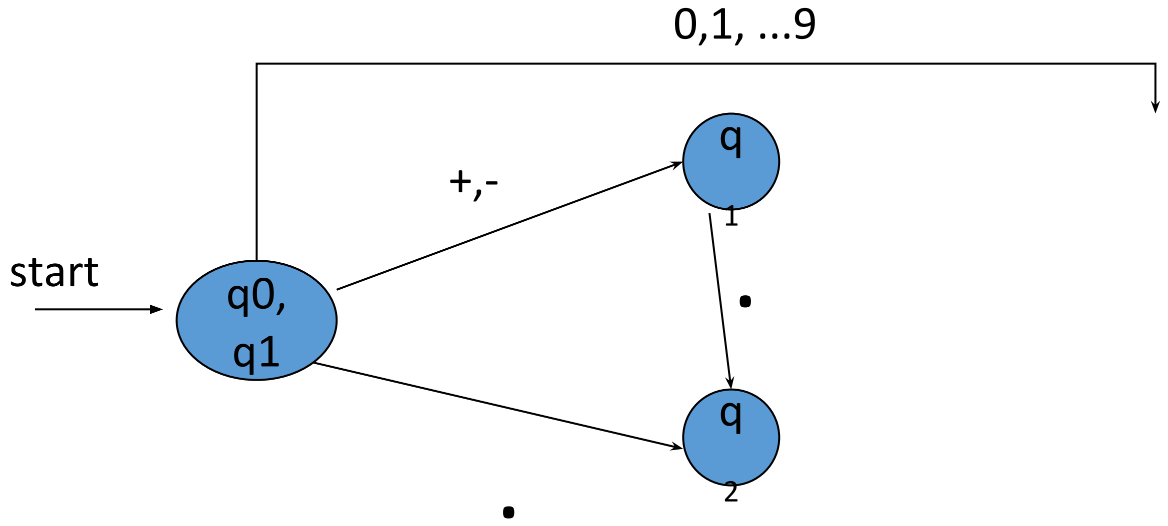
- Construction is very close to the subset construction as the states of D are subsets of E.



Eliminating ε transistions

Eliminating ε transitions results in DFA that accepts the same language as E

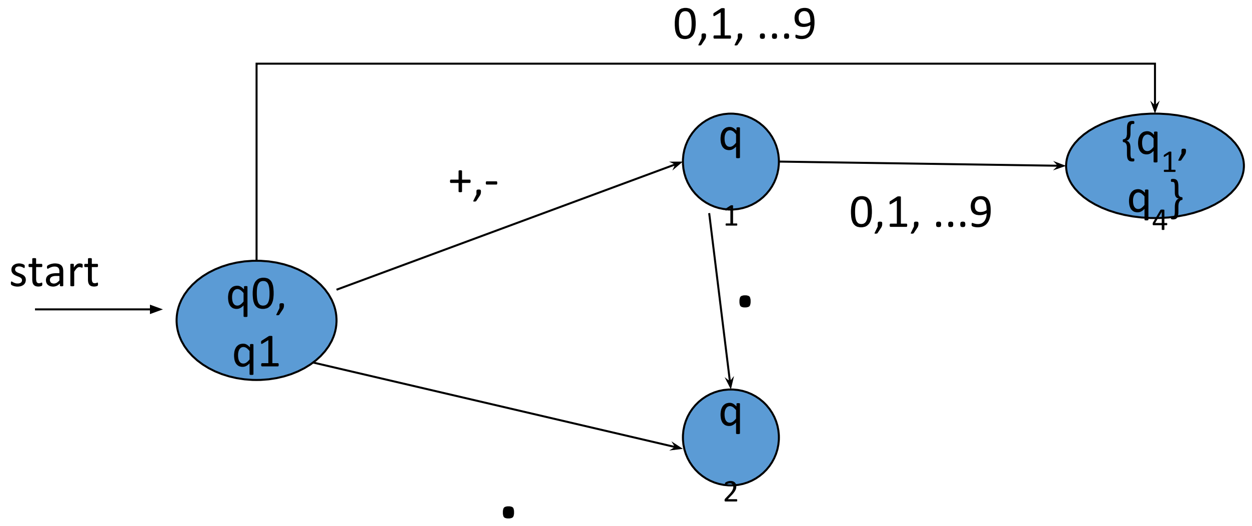
- Construction is very close to the subset constructiton as the states of D are subsets of E.



Eliminating ε transistions

Eliminating ε transitions results in DFA that accepts the same language as E

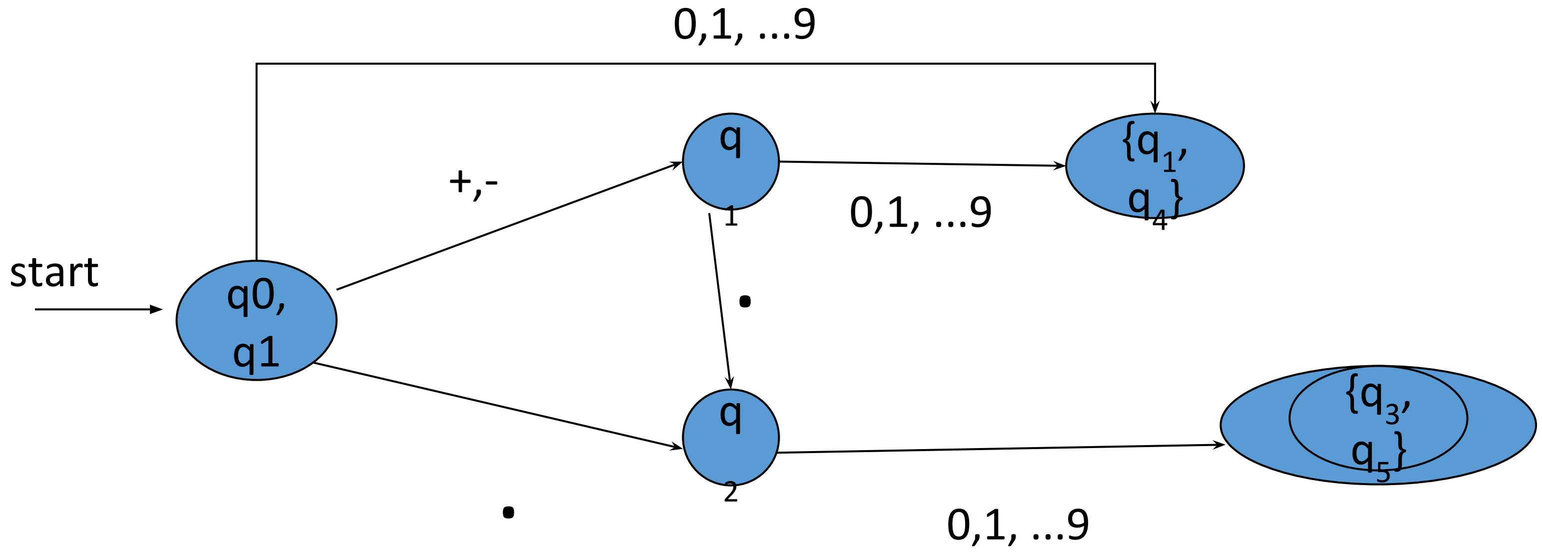
- Construction is very close to the subset constructiton as the states of D are subsets of E.



Eliminating ε transistions

Eliminating ε transitions results in DFA that accepts the same language as E

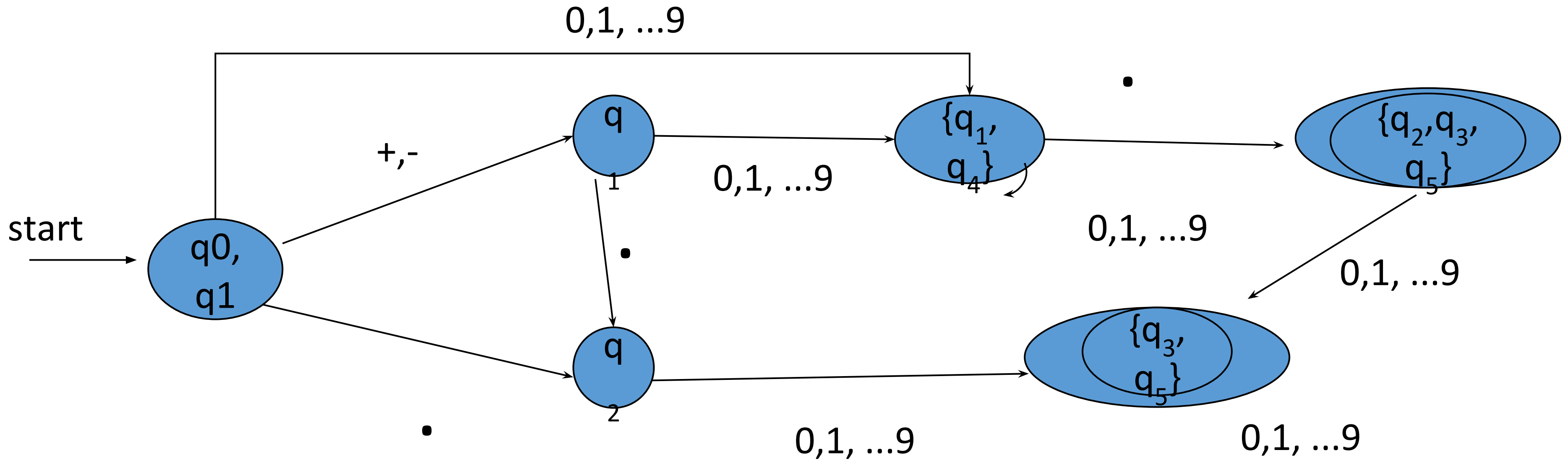
- Construction is very close to the subset constructiton as the states of D are subsets of E.



Eliminating ε transistions

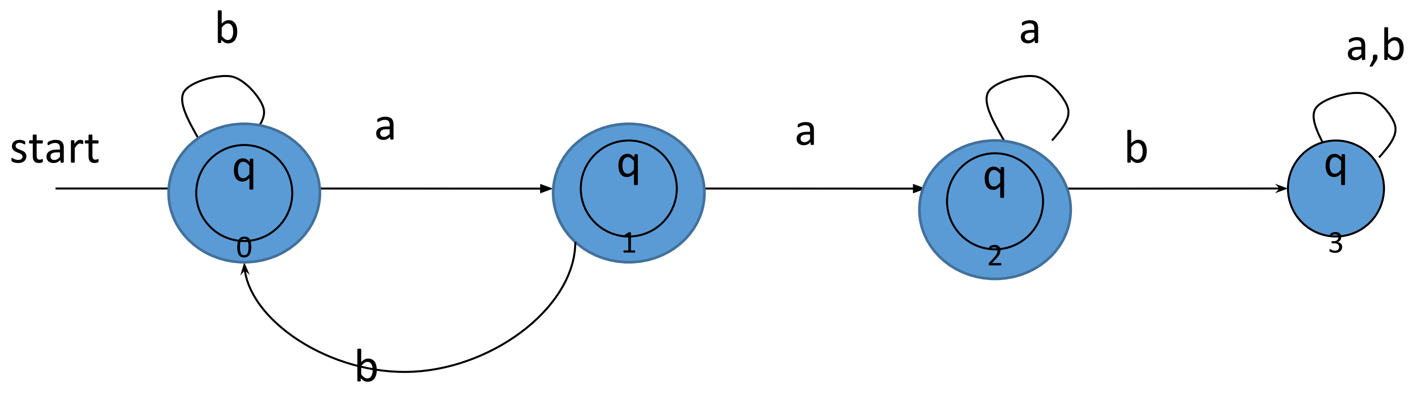
Eliminating ε transitions results in DFA that accepts the same language as E

- Construction is very close to the subset constructiton as the states of D are subsets of E.



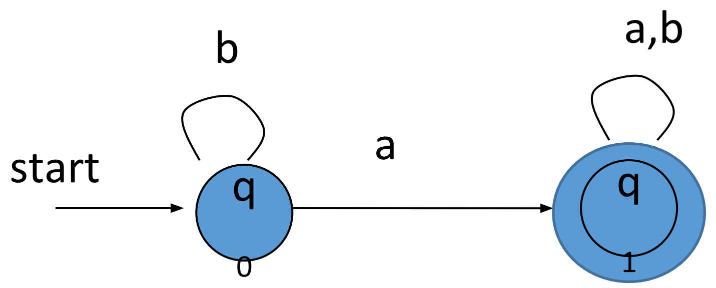
Design a DFA to accept strings of a's and b's except those containing the substring aab.

Design a DFA to accept strings of a's and b's except those containing the substring aab.

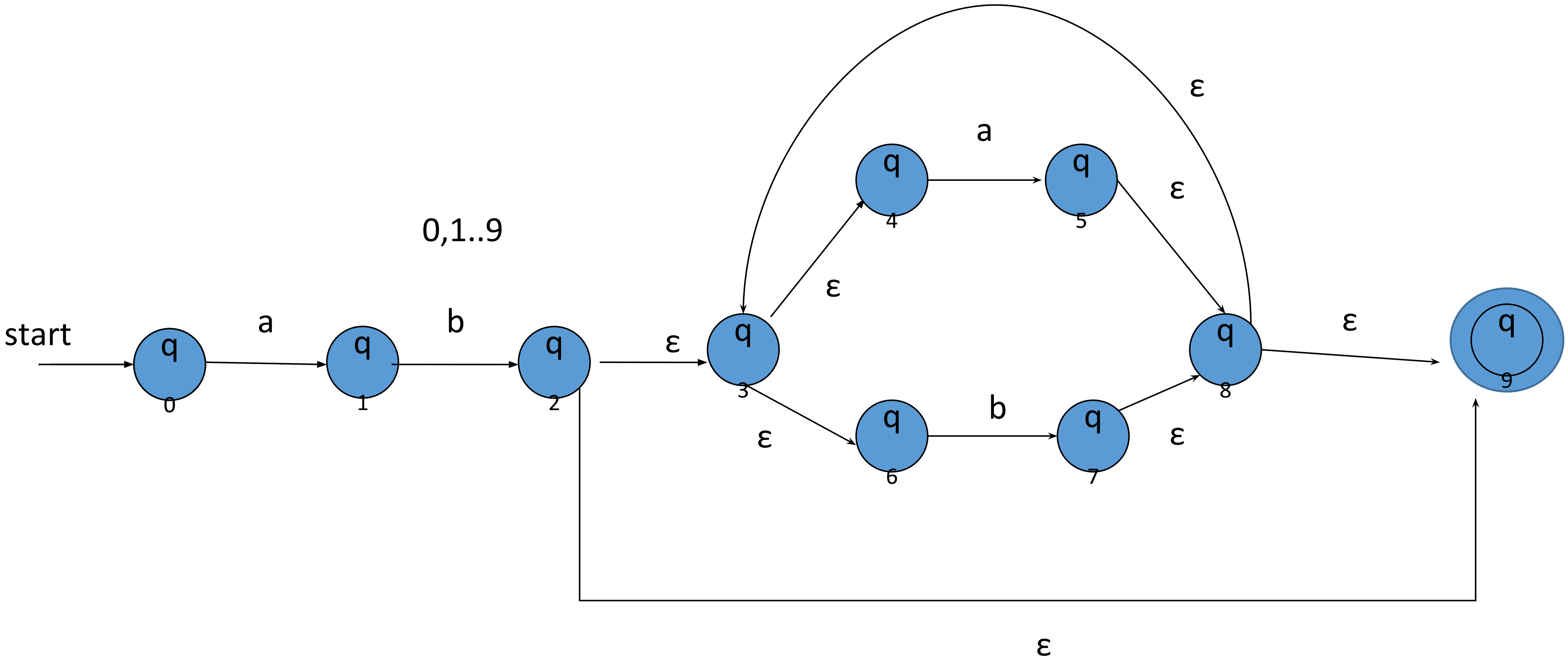


Design a DFA to accept strings of a's and b's having atleast one a

Design a DFA to accept strings of a's and b's having atleast one a



convert the ε-NFA to DFA



$\delta(q_0,a)=\{q1\}$ $\delta(q_0,b)=\emptyset$

$\delta(q_1,a)=\emptyset$ $\delta(q_1,b)=\{q2,q3,q4,q6\}$

εclosure(q0)={q0}-------------- (A)

δ(A,a) = εclosure(δN(A,a))= εclosure(δN(q0,a)) = {q1} ---------------- (B)

δ(A,b) = εclosure(δN(A,b))= εclosure(δN(q0,b)) = Ø

δ(B,a) = Ø δ(B,b) =

convert the ε-NFA to DFA

$\delta(q_0,a)=\{q1\}$ $\delta(q_0,b)=\emptyset$

$\delta(q_1,a)=\emptyset$ $\delta(q_1,b)=\{q2,q3,q4,q6\}$

εclosure(q0)={q0}-------------- (A)

δ(A,a) = εclosure(δN(A,a))= εclosure(δN(q0,a)) = {q1} ---------------- (B)

δ(A,b) = εclosure(δN(A,b))= εclosure(δN(q0,b)) = Ø

δ(B,a) = εclosure(δN(B,a))= εclosure(δN(q1,a)) = Ø

δ(B,b) = εclosure(δN(B,b))= εclosure(δN(q1,b)) = εclosure(q2)={q2,q3,q4,q6,q9} ----------- (C)

convert the ε-NFA to DFA

$\delta(q_0,a)=\{q1\}$ $\delta(q_0,b)=\emptyset$

$\delta(q_1,a)=\emptyset$ $\delta(q_1,b)=\{q2,q3,q4,q6\}$

εclosure(q0)={q0}-------------- (A)

δ(A,a) = εclosure(δN(A,a))= εclosure(δN(q0,a)) = {q1} ---------------- (B)

δ(A,b) = εclosure(δN(A,b))= εclosure(δN(q0,b)) = Ø

δ(B,a) = εclosure(δN(B,a))= εclosure(δN(q1,a)) = Ø

δ(B,b) = εclosure(δN(B,b))= εclosure(δN(q1,b)) = εclosure(q2)={q2,q3,q4,q6,q9} ----------- (C)

δ(C,a) = εclosure(δN(C,a))= εclosure(δN(q2,q3,q4,q6,q9),a)) = εclosure(q5)={q5,q8,q9,q3,q4,q6} ---- (D)

δ(C,b) = εclosure(δN(C,b))= εclosure(δN(q2,q3,q4,q6,q9),b)) = εclosure(q7)={q3,q4,q6,q7,q8,q9} ---- (E)

convert the ε-NFA to DFA

$\delta(q_0,a)=\{q1\}$ $\delta(q_0,b)=\emptyset$

$\delta(q_1,a)=\emptyset$ $\delta(q_1,b)=\{q2,q3,q4,q6\}$

εclosure(q0)={q0}-------------- (A)

δ(A,a) = εclosure(δN(A,a))= εclosure(δN(q0,a)) = {q1} ---------------- (B)

δ(A,b) = εclosure(δN(A,b))= εclosure(δN(q0,b)) = Ø

δ(B,a) = εclosure(δN(B,a))= εclosure(δN(q1,a)) = Ø

δ(B,a) = εclosure(δN(B,b))= εclosure(δN(q1,b)) = εclosure(q2)={q2,q3,q4,q6} ----------- (C)

δ(C,a) = εclosure(δN(C,a))= εclosure(δN(q2,q3,q4,q6,q9),a)) = εclosure(q5)={q5,q8,q9,q3,q4,q6} ---- (D)

δ(C,b) = εclosure(δN(C,b))= εclosure(δN(q2,q3,q4,q6,q9),b)) = εclosure(q7)={q3,q4,q6,q7,q8,q9} ---- (E)

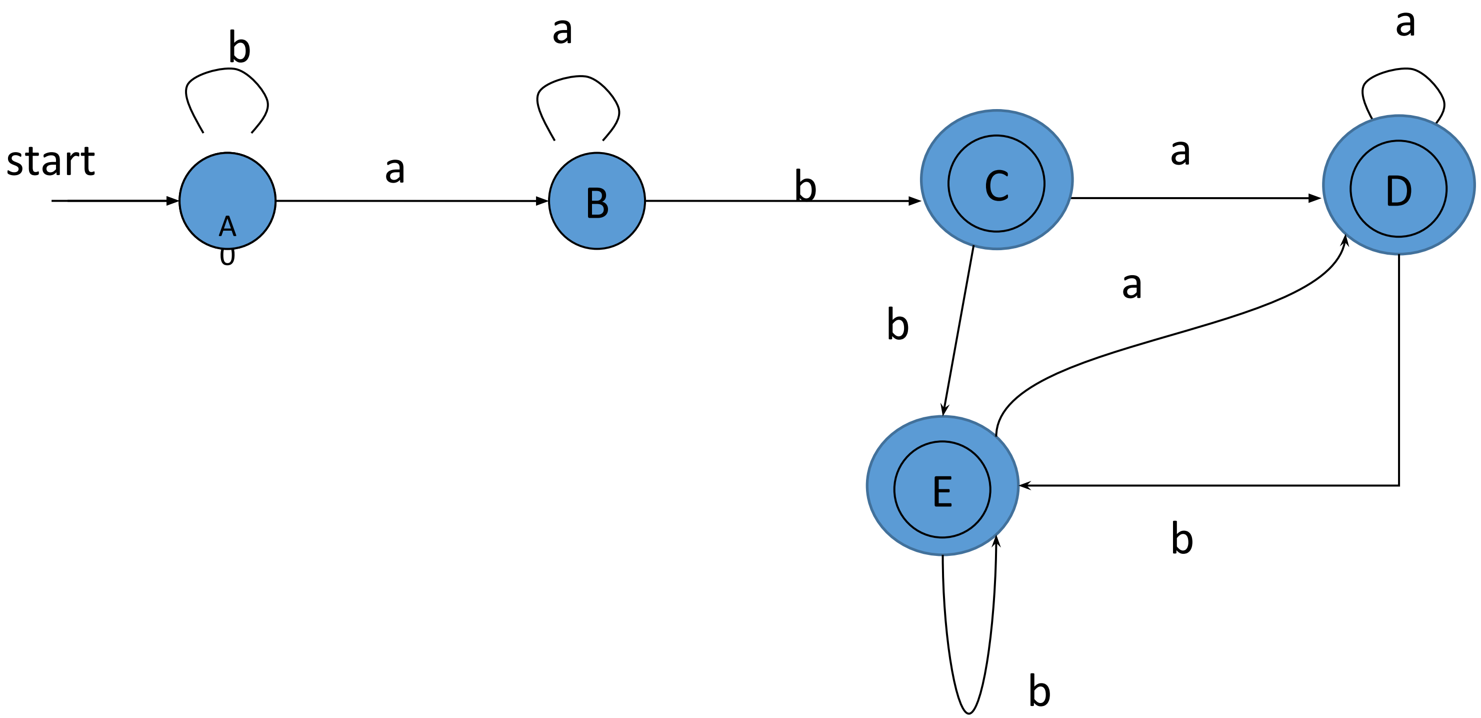
δ(D,a) = εclosure(δN(D,a))= εclosure(δN(q3,q4,q5,q6,q8,q9),a)) = εclosure(q5)={q5,q8,q9,q3,q4,q6} ---- (D)

δ(D,b) = εclosure(δN(D,b))= εclosure(δN(q5,q3,q4,q6,q7,q8,q9),b)) = εclosure(q7)={q3,q4,q6,q7,q8,q9} ---- (E)

δ(E,a) = εclosure(δN(E,a))= εclosure(δN(q3,q4,q6,q7,q8,q9),a)) = εclosure(q5)={q5,q8,q9,q3,q4,q6} ---- (D)

δ(E,b) = εclosure(δN(E,b))= εclosure(δN(q3,q4,q6,q7,q8,q9),b)) = εclosure(q7)={q3,q4,q6,q7,q8,q9} ---- (E)

ε-NFA to DFA



Convert ε NFA to DFA