PART 1:

Automata:

Finite State Machines (Finite Automata)

Formal Language:

Regular Languages

Non-regular Languages

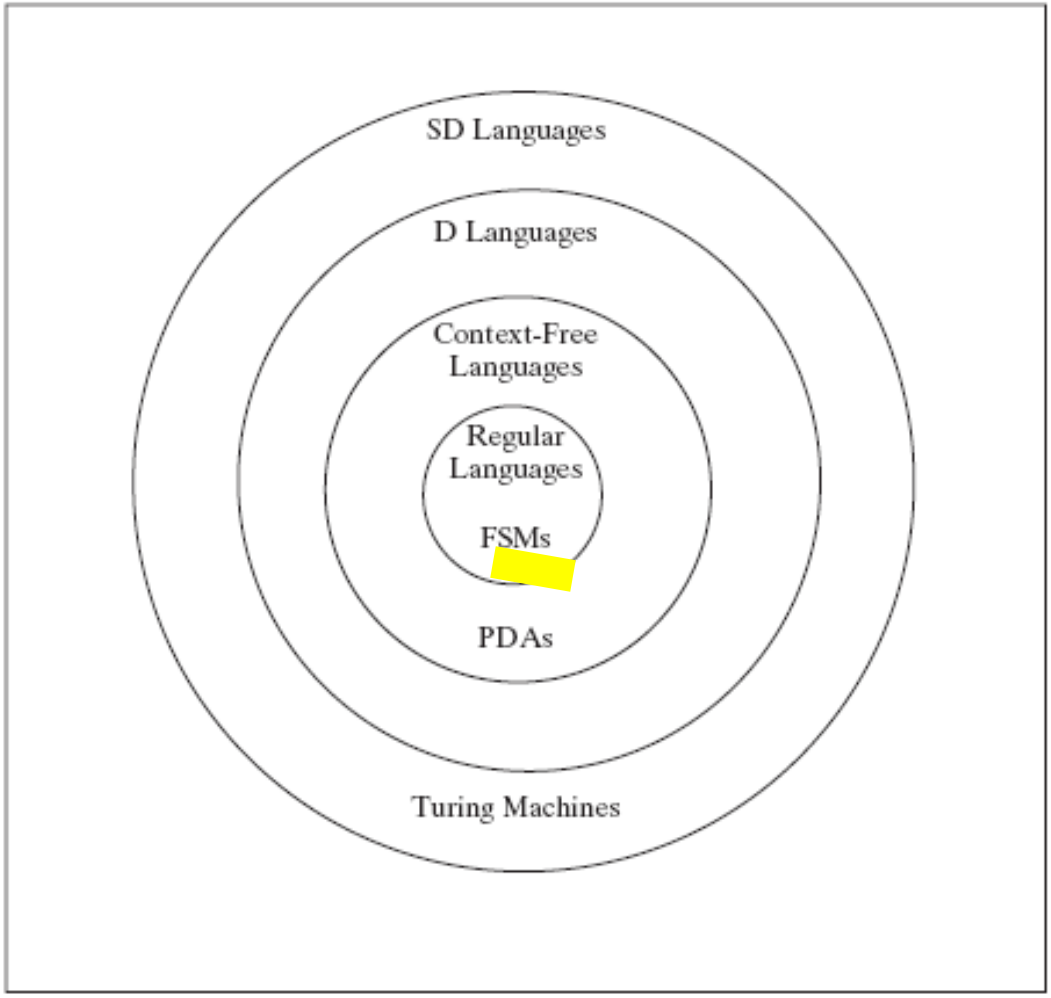
Grammar:

Regular Expressions

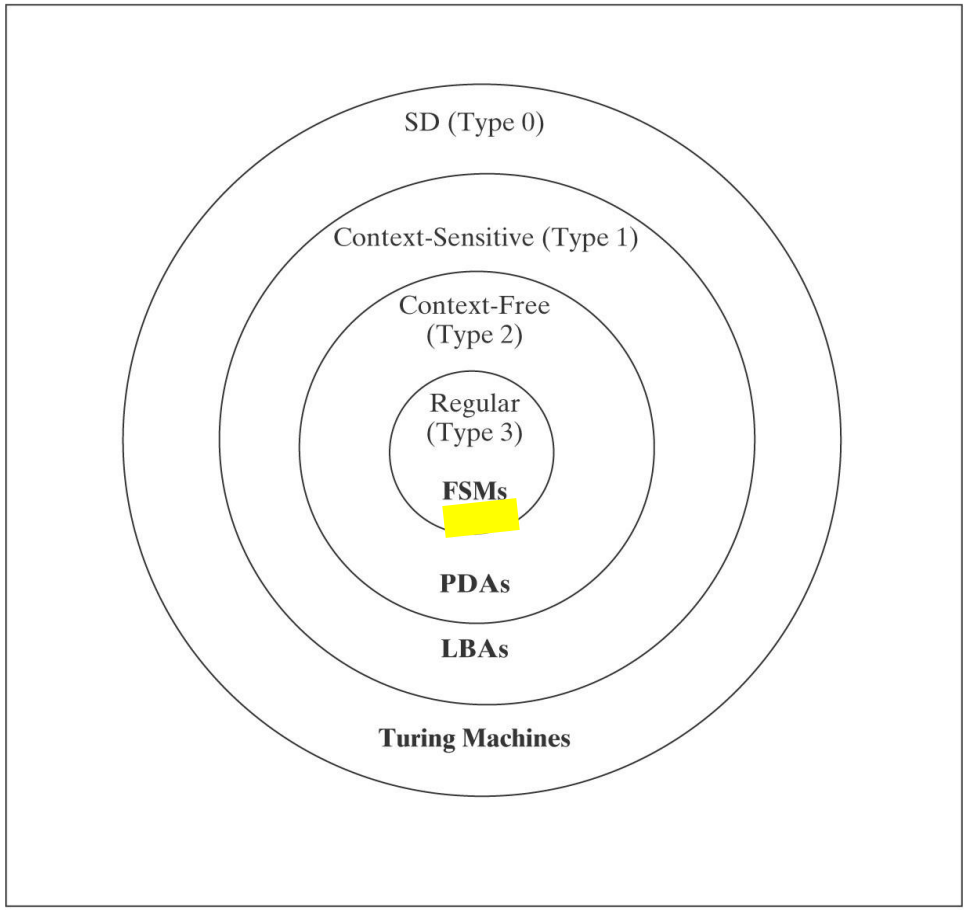
Regular Grammars

Finite State Machines (Finite Automata)

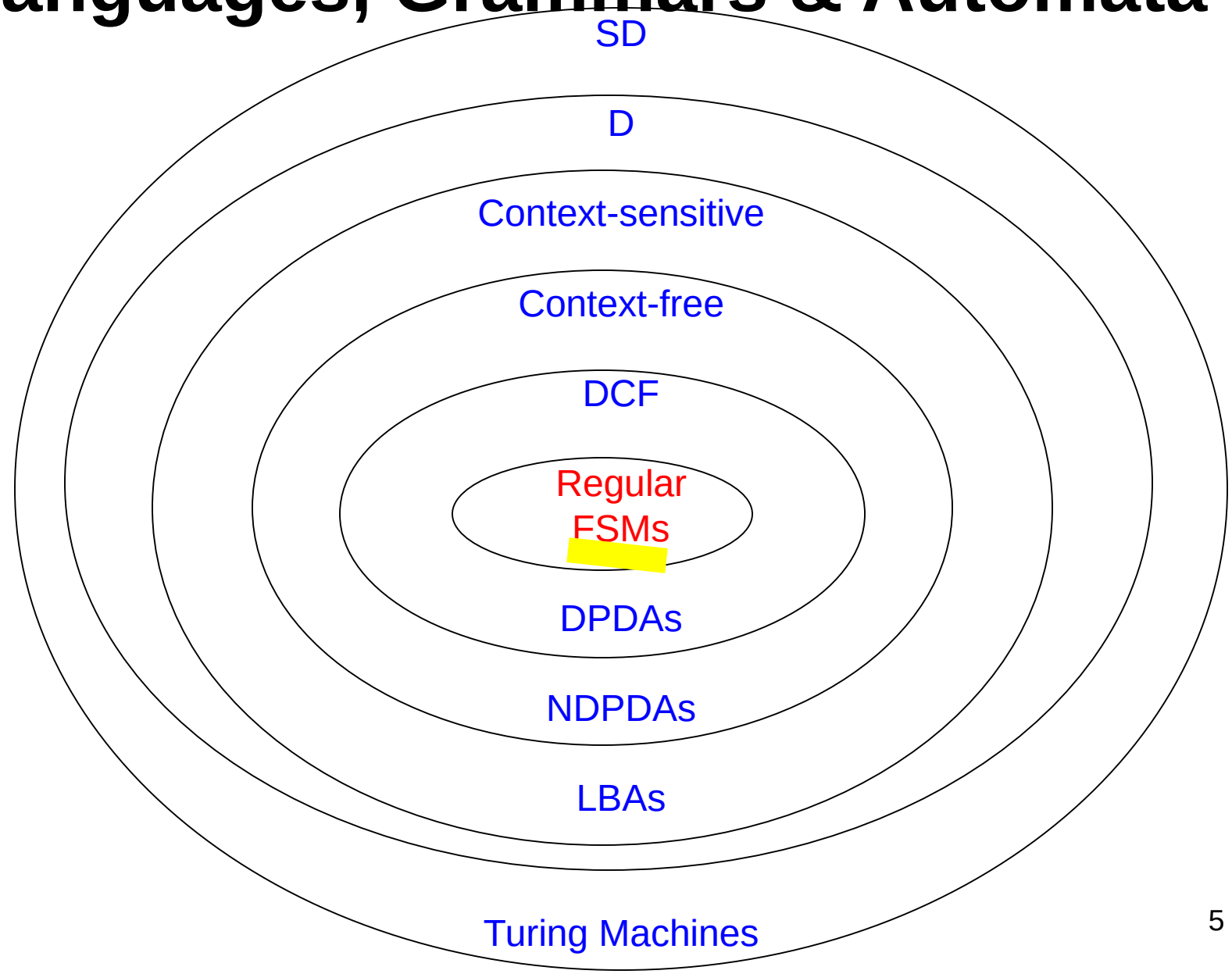
Languages, Grammars & Automata



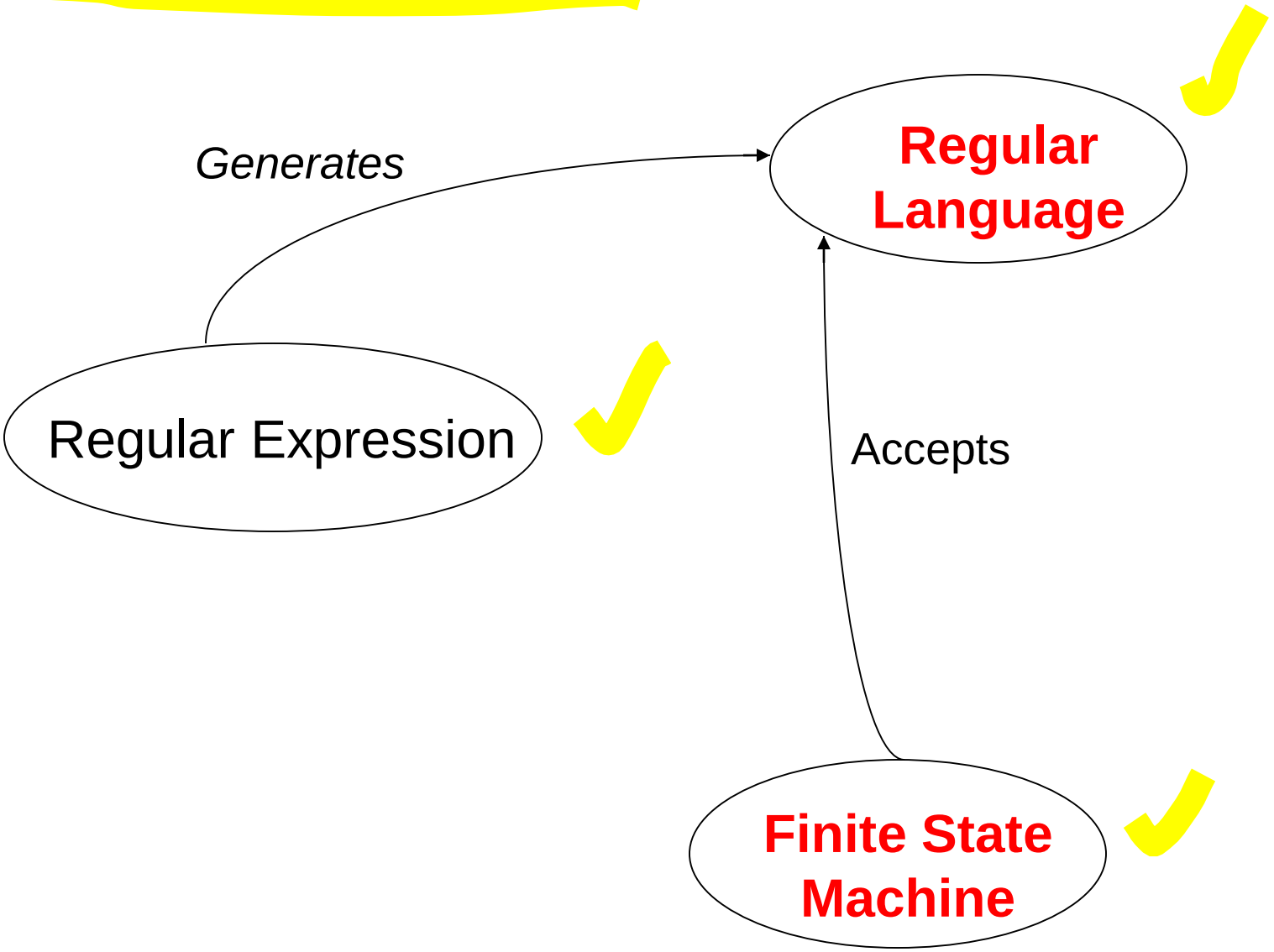
Languages, Grammars & Automata



Languages, Grammars & Automata



Regular Languages



FSM (Finite Automaton)

- A computational model as an idealized small computer with **limited memory** – finite and small amount of memory!

- The simplest model of computation!

Deterministic FSM (DFSM or DFA)

Definition of a Deterministic FSM

A **DFSM (DFA)** $M = (K, \Sigma, \delta, s, A)$ where:

K is a finite set of states

Σ is an alphabet of symbols

$s \in K$ is the initial (starting) state

$A \subseteq K$ is the set of accepting states, and

δ is the transition **function** from $(K \times \Sigma)$ to K

Example: DFSM

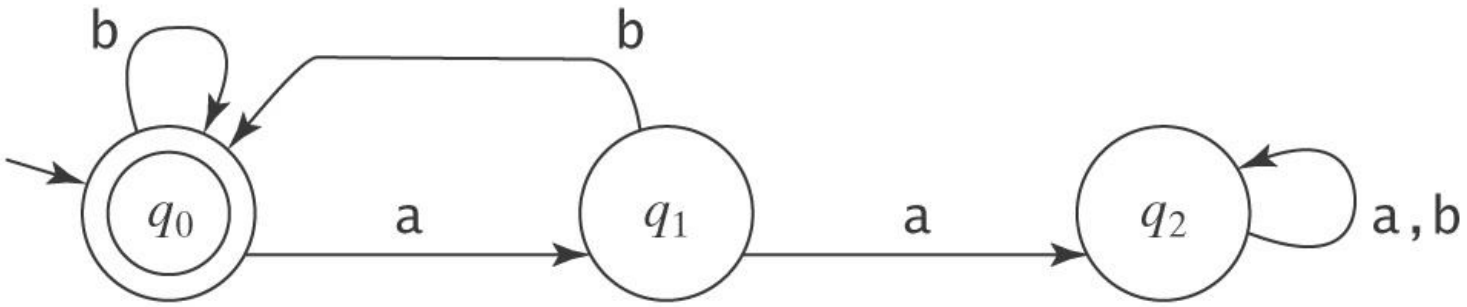
$K =$

$\Sigma =$

$s \in K =$

$A \subseteq K =$

$\delta =$ Transition Diagram!



Configurations of DFSMs

A *configuration* of a DFSM *M* is an element of:

$$K \times \Sigma^*$$

It captures the two things that can make a difference to *M*'s future behavior:

- its current state
- the input that is still left to read.

The *initial configuration* of a DFSM *M*, on input *w*, is: (s_M, w)

The Yields Relations

The *yields-in-one-step* relation $|-_M$:

$(q, w) |-_M (q', w')$ iff

- $w = a\ w'$ for some symbol $a \in \Sigma$, and
- $\delta\ (q, a) = q'$

$|-_M$ * is the *reflexive, transitive closure* of $|-_M$.

Computations Using DFSMs

A *computation* by M is a finite sequence of configurations $C_0, C_1, \ldots, C_n$ for some $n \geq 0$ such that:

- C_0 is an initial configuration,
- C_n is of the form (q, ε), for some state $q \in K$,
- $C_0 \vdash_M C_1 \vdash_M C_2 \vdash_M \ldots \vdash_M C_n$.

Accepting and Rejecting

A DFSM M *accepts* a string w iff:

$(s, w) \vdash_M^* (q, \varepsilon)$, for some $q \in A$.

A DFSM M *rejects* a string w iff:

$(s, w) \vdash_M^* (q, \varepsilon)$, for some $q \notin A$.

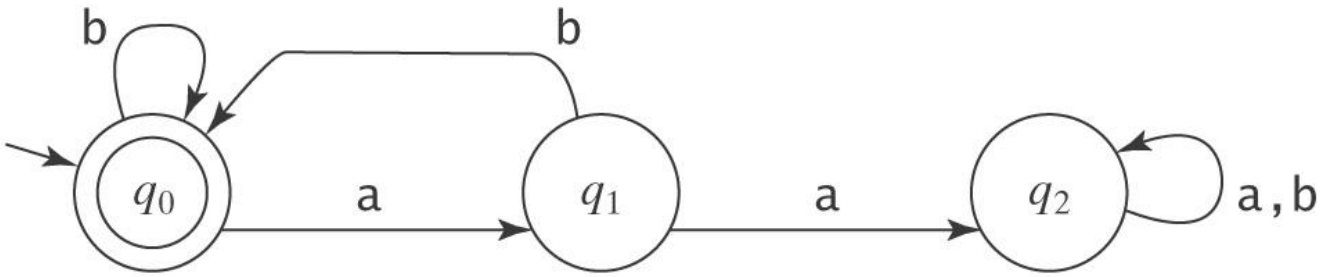
The ***language accepted by*** DFSM M, denoted ***L*(*M*)**, is the set of all strings accepted by M.

Examples and Designing DFSMs

Example 5.2

L = {*w* ∈ {a, b}* : every a is immediately followed by a b}.

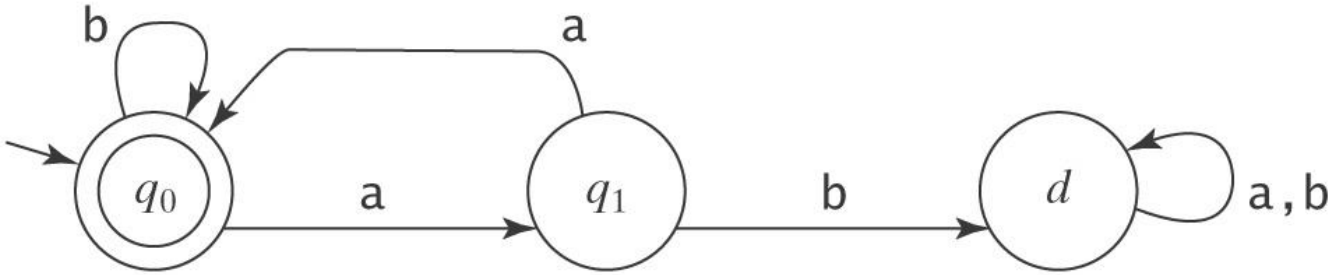
DFSM?



Example 5.3

L = {$w \in$ {a, b}* : every a region in w is of even length}

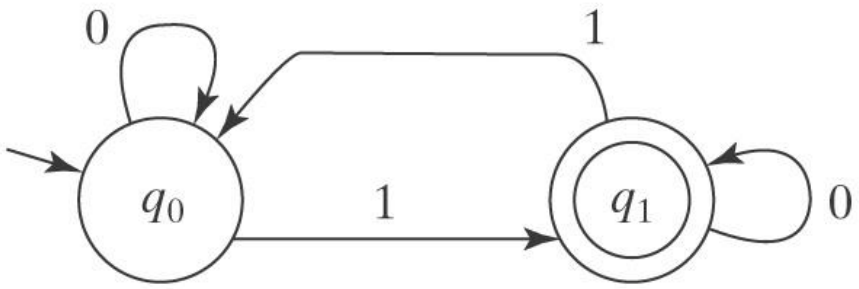
DFSM?



Example 5.4

L = {$w \in \{0, 1\}^*$: w has odd parity - odd # of 1's}.

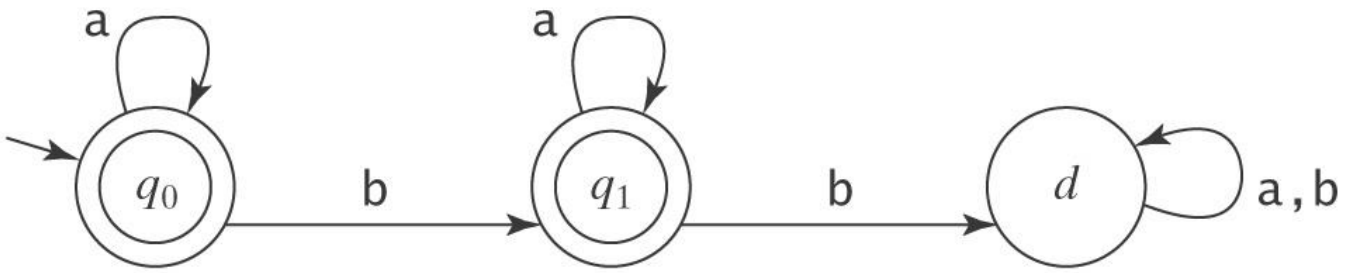
DFSM?



Example 5.5

L = {*w* ∈ {a, b}* : *w* contains no more than one b}.

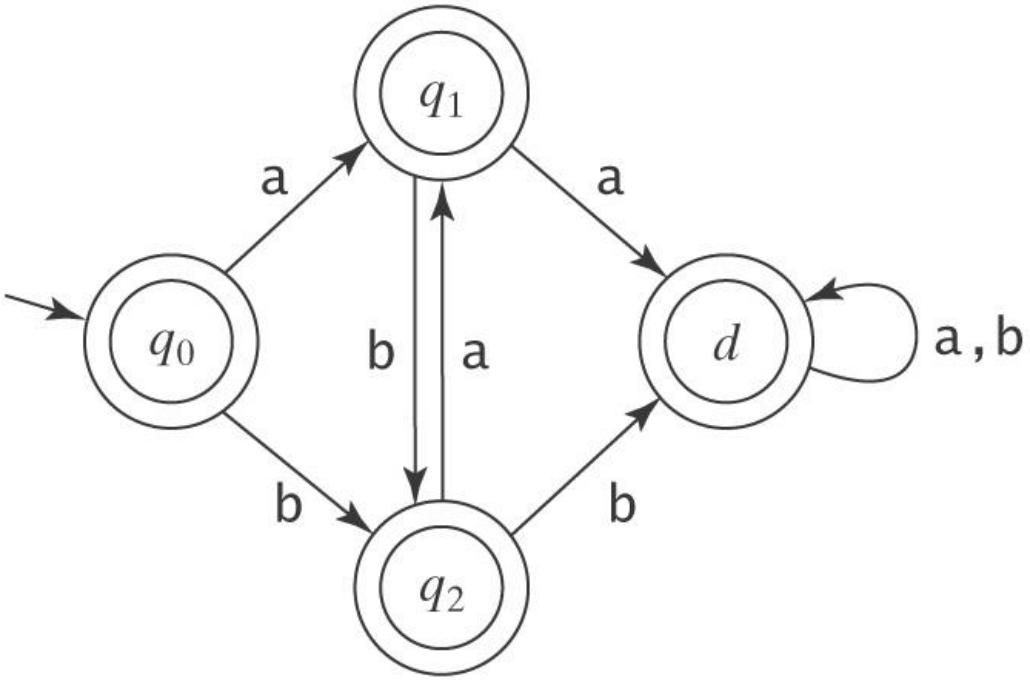
DFSM?



Example 5.6

L = {$w \in$ {a, b}* : no two consecutive characters are the same}.

DFSM?

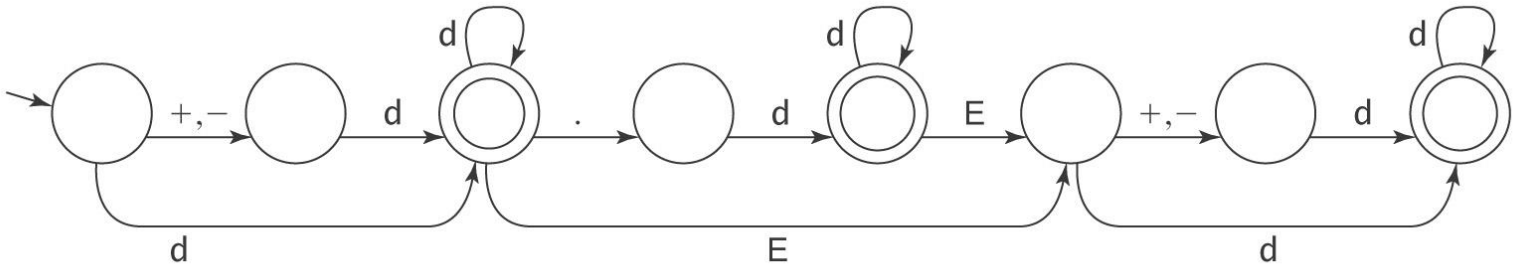


Example 5.7

L = FLOAT = {*w*: w is the string representation of a floating point number}.

`+3.0, 3.0, 0.3E1, 0.3E+1, -0.3E+1, -3E8`

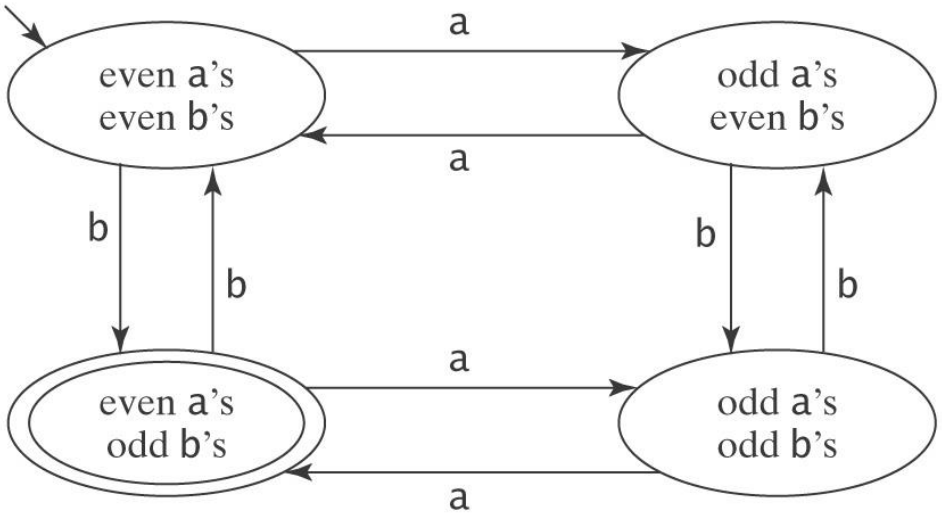
DFSM?



Example 5.9

L = {*w* ∈ {a, b}* : *w* contains an even number of a's and an odd number of b's}

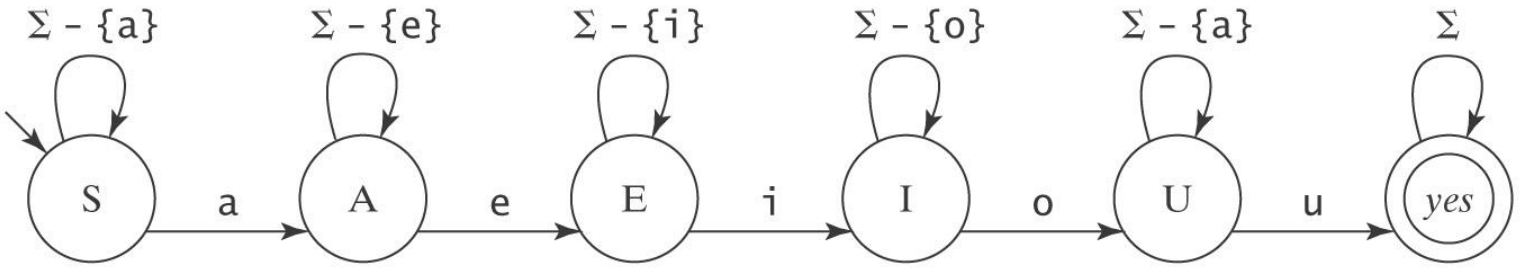
DFSM?



Example 5.10

L = {$w \in$ {a - z}* : all five vowels, a, e, i, o, and u, occur in w in alphabetical order}.

DFSM?

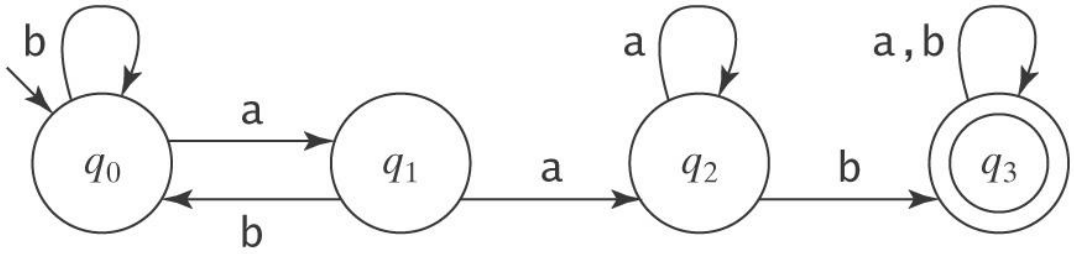


Example 5.11

L = {$w \in$ {a, b}* : w does not contain the substring aab}.

DFSM?

DFSM for $\neg$L:



Example 5.12

$\Sigma = \{\texttt{a}, \texttt{b}, \texttt{c}, \texttt{d}\}$.

$L_{Missing} = \{w$: there is a symbol $a_i \in \Sigma$ not appearing in $w\}$.

DFSM?

More Examples

L = {} = ∅

DFSM?

L= {ε}

DFSM?

DFSMs Halt

Theorem 5.1 Every DFSM *M*, on any input *s* of finite length, halts in |*s*| steps.

Regular Languages

A language is ***regular*** iff it is accepted by some DFSM.

RL = DFSM

Nondeterministic FSM (NDFSM or NDFA or NFA)

Determinism and Nondeterminism

- Deterministic computation on a deterministic machine
- Nonderetminstic computation on a nondeterministic machines

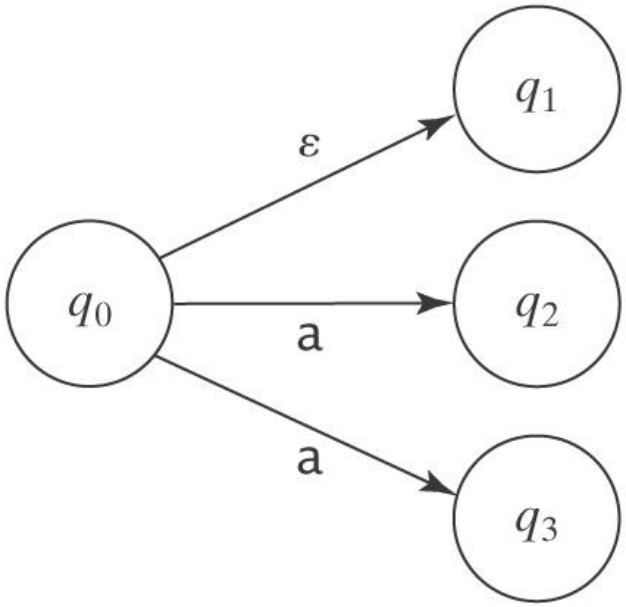
Determinism of DFSM

δ is the transition function from $(K \times \Sigma)$ to K

- Only one unique next state!
- No choices!
- No randomness!

Sources of Nondeterminism

- Multiple edges
- Epsilon edges



Nondeterminism of NDFSM

δ is the transition relation from $(K \times \Sigma)$ to K

- The next state is chosen at random!
- All next states are chosen in parallel and pursued simultaneously!

Definition of an NDFSM

A **NDFSM (NDFA)** $M = (K, \Sigma, \Delta, s, A)$ where:

K is a finite set of states

Σ is an alphabet

$s \in K$ is the initial state

$A \subseteq K$ is the set of accepting states, and

Δ is the transition **relation**. It is a finite subset of

$$(K \times (\Sigma \cup \{\varepsilon\})) \times K$$

Example: NDFSM

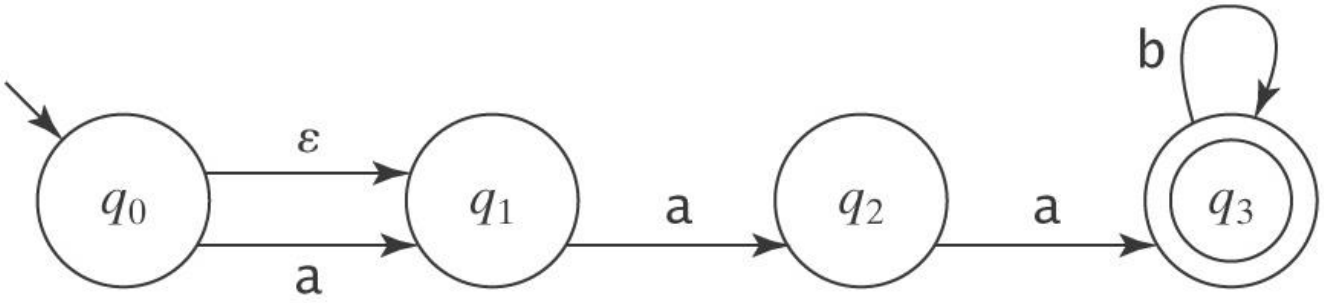
$K =$

$\Sigma =$

$s \in K =$

$A \subseteq K =$

$\Delta =$



Accepting by an NDFSM

- *M* accepts a string *w* iff at least one of its computations accepts, i.e., there exists *some path* along which *w* drives *M* to some element of *A*.

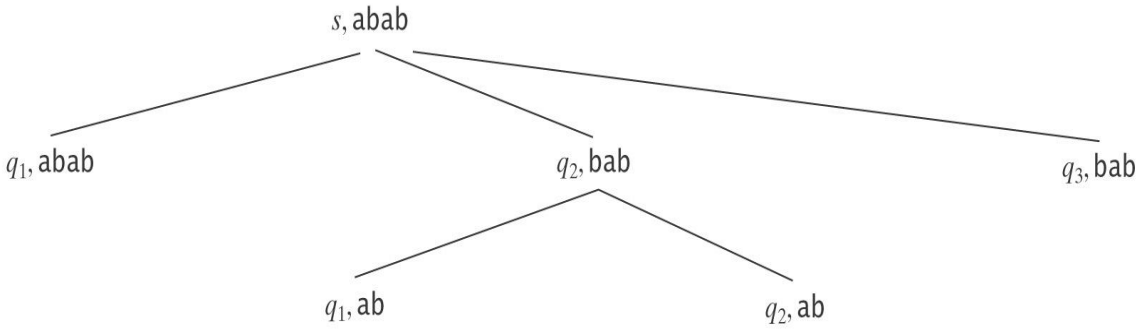
- *M* rejects a string *w* iff none of its computations accepts.

The language accepted by NDFSM *M*, denoted ***L*(*M*)**, is the set of all strings accepted by *M*.

Analyzing Nondeterministic FSMs

Two approaches:

- ✓ Explore a search tree:

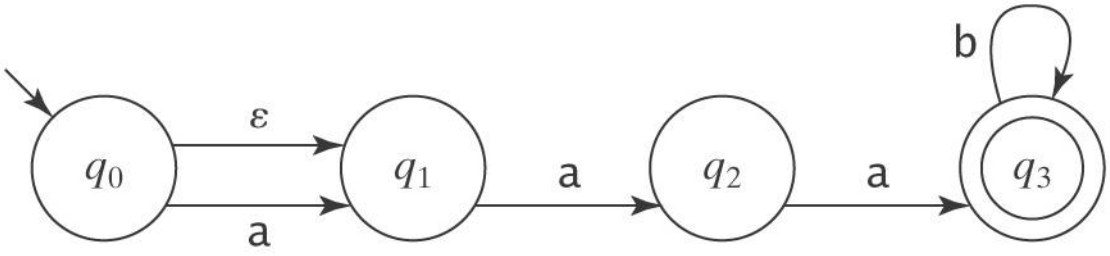


- ✓ Follow all paths in parallel!

Example 5.13

L = {*w* ∈ {a, b}* : *w* is made up of an optional a followed by aa followed by zero or more b's}.

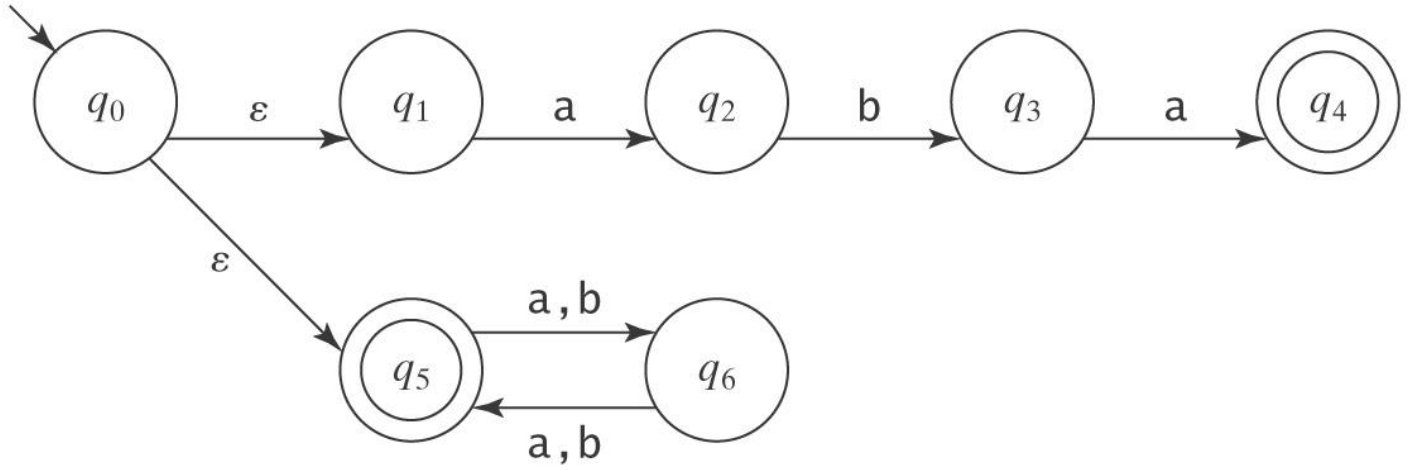
NDFSM?



Example 5.14

L = {*w* ∈ {a, b}* : *w* = aba or |*w*| is even}.

NDFSM?

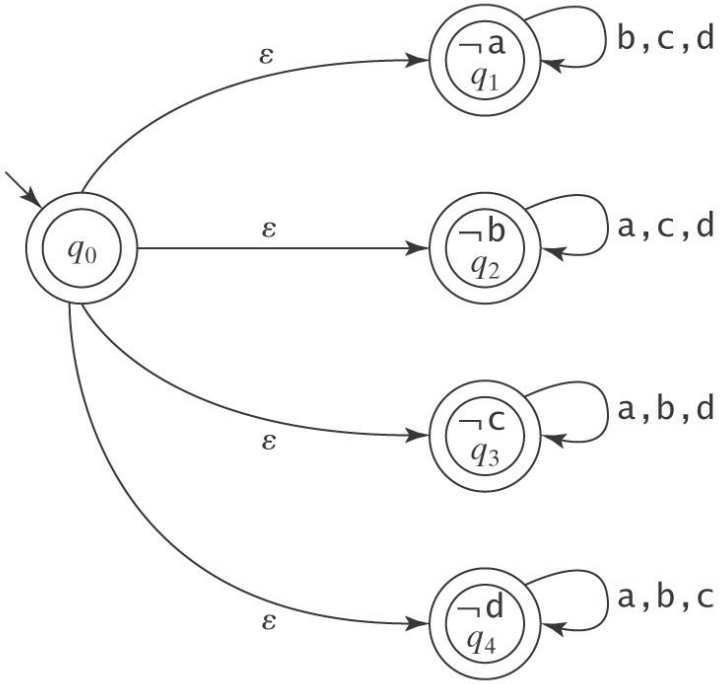


Example 5.15

Σ = {a, b, c, d}.

$L_{Missing}$ = {*w* : there is a symbol $a_i \in \Sigma$ not appearing in *w*}

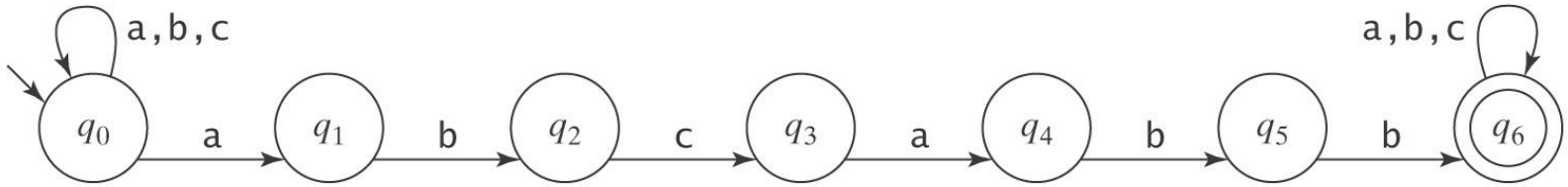
NDFSM?



Example 5.16

L = {$w \in$ {a, b, c}* : $\exists x, y \in$ {a, b, c}* ($w = x$ abcabb y)}.

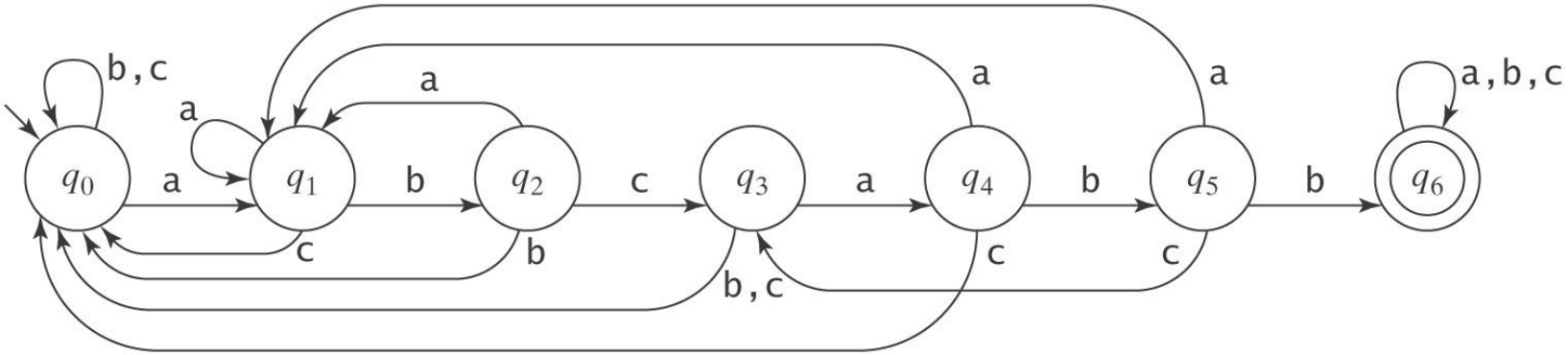
NDFSM?



Example 5.16

L = {$w \in$ {a, b, c}* : $\exists x, y \in$ {a, b, c}* ($w = x$ abcabb y)}.

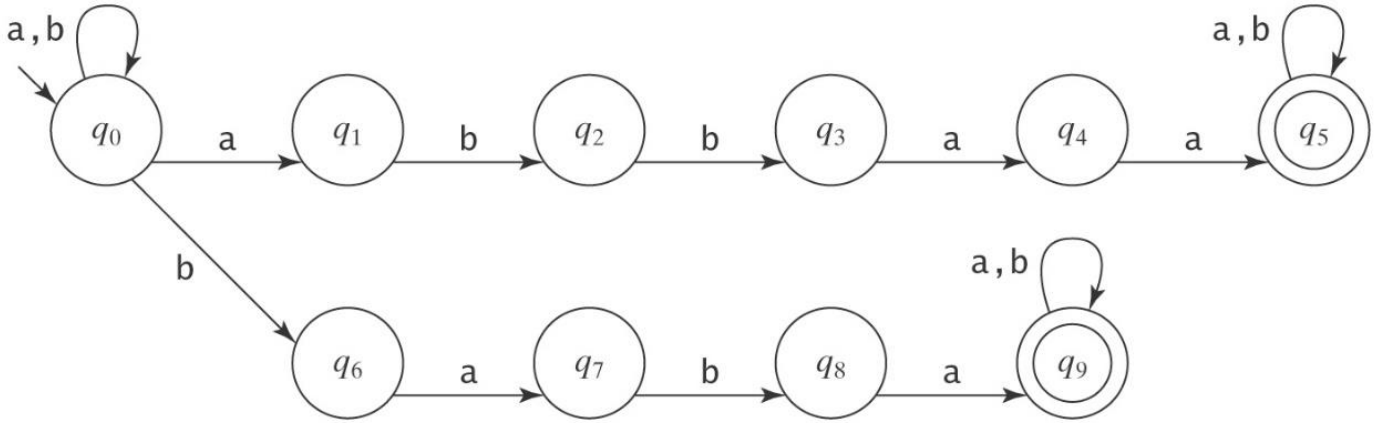
DFSM?



Example 5.17

L = {$w \in$ {a, b}* : $\exists x, y \in$ {a, b}* (($w = x$ abbaa y) $\vee$ ($w =$ x baba y))}.

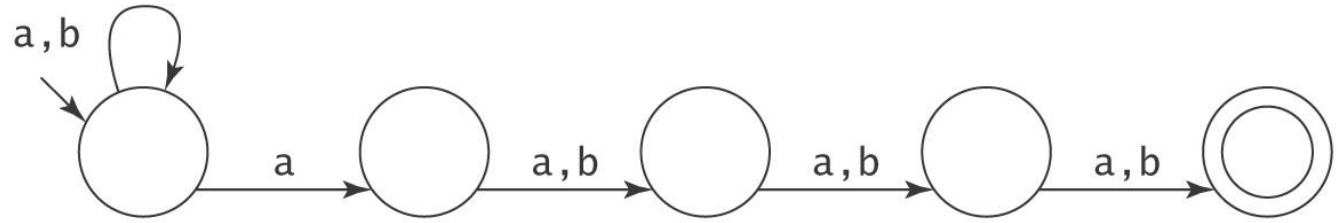
NDFSM?



Example 5.18

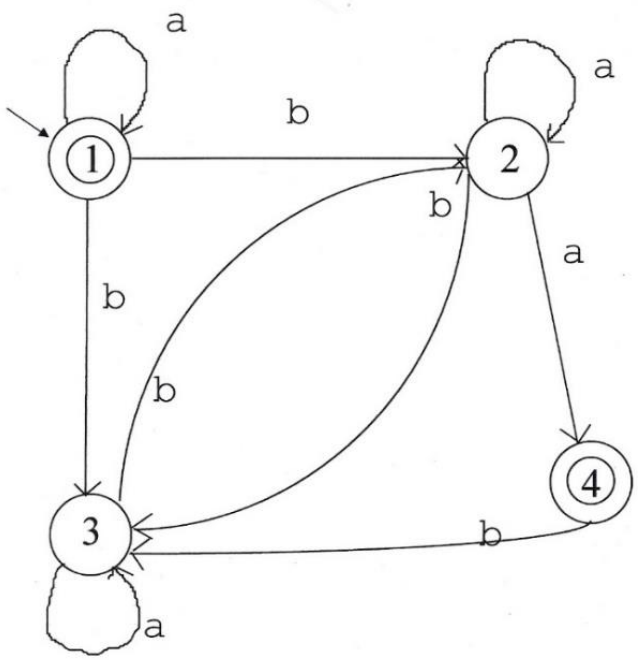
L = {*w* ∈ {a, b}* : the fourth to the last character is a}

NDFSM?



Analyzing Nondeterministic FSMs

Does this NDFSM accept `baaba` ?



Handling ε-Transitions via ε-Closure

ε -closure:

- ***eps(q)*** = The **set** of states that are reachable from *q* by following 0 or more ε–transitions.

- $eps(q) = \{ p \in K : (q, w) \vdash^*_M (p, w) \}$

- *eps*(*q*) is the *closure* of {*q*} under the relation {(*p*, *r*) : there is a transition (*p*, ε, *r*) ∈ Δ}.

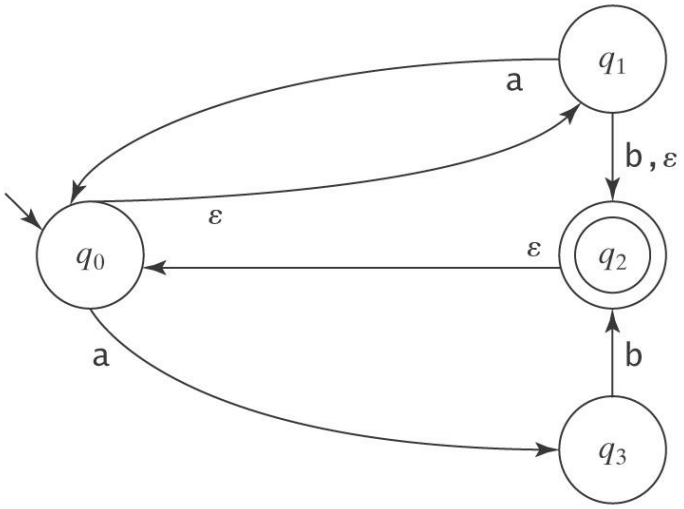
An Algorithm to Compute *eps*(*q*)

eps(*q*: state) =

result = {*q*};
while
there exists some $p \in$ *result* and
some $r \notin$ *result* and some transition $(p, \varepsilon, r) \in \Delta$
do
Insert *r* into *result;*
return *result;*

Example 5.19

NDFSM:



$eps(q_0)$ =
$eps(q_1)$ =
$eps(q_2)$ =
$eps(q_3)$ =

Equivalence of NDFSMs and DFSMs

NDFSM = DFSM

RL = DFSM = NDFSM

Equivalence of NDFSMs and DFSMs

Theorem 5.2 If there is a DFSM for L, there is an NDFSM for L.

Equivalence of NDFSMs and DFSMs

Theorem 5.3 If there is an NDFSM for L, there is a DFSM for L.

Proof Idea: Proof by Construction

The Subset Construction

Given a NDFSM $M = (K, \Sigma, \Delta, s, A)$, we construct DFSM $M' = (K', \Sigma, \delta', s', A')$, where

$K' = \mathcal{P}(K)$

$s' = eps(s)$

$A' = \{Q \subseteq K' : Q \cap A \neq \emptyset\}$

$\delta'(Q, a) = \bigcup\{$***eps(p)***: $p \in K$ and $(q, a, p) \in \Delta$ for some $q \in Q\}$

An Algorithm for Constructing the DFSM from NDFSM

1. Compute the *eps*(*q*)'s.

2. Compute $s' = eps(s)$.

3. Compute δ'.

4. Compute K' = a subset of $\mathcal{P}(K)$.

5. Compute $A' = \{Q \in K' : Q \cap A \neq \emptyset\}$.

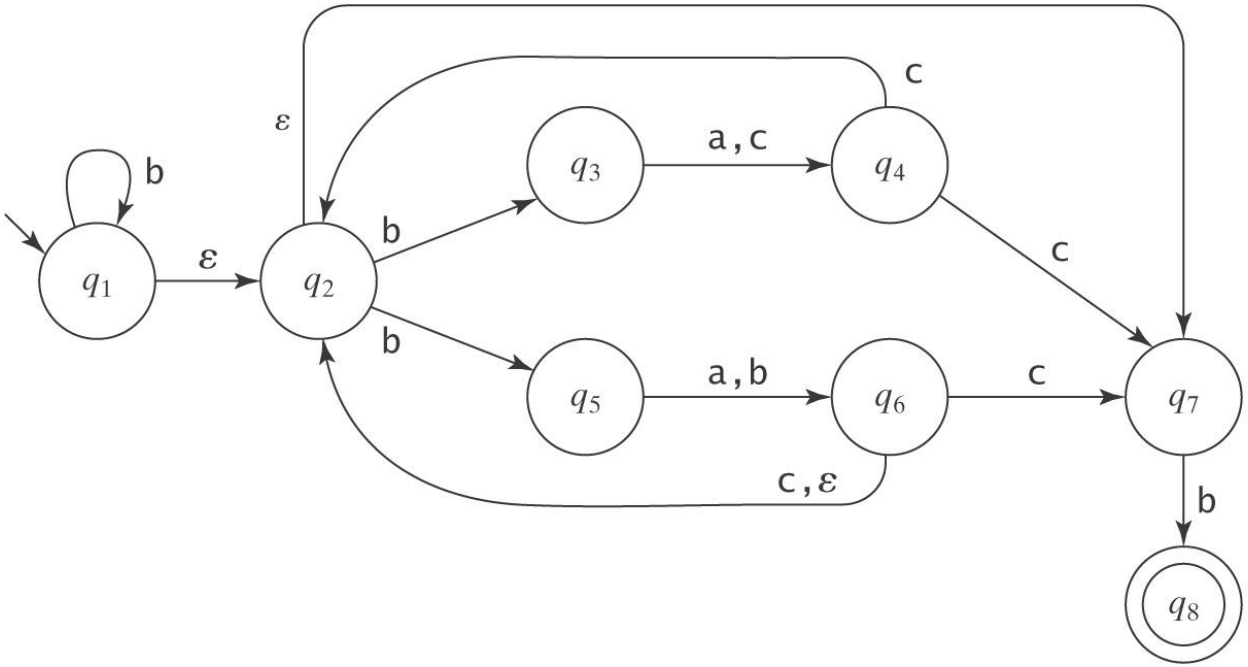
The Algorithm *ndfsmtodfsm*

ndfsmtodfsm(*M*: NDFSM) =

1. For each state *q* in K_M do:
    1.1 Compute *eps*(*q*).
2. $s' = eps(s)$
3. Compute δ':
    3.1 active-states = {*s'*}.
    3.2 $\delta' = \varnothing$.
    3.3 While there exists some element *Q* of *active-states* for which δ' has not yet been computed do:
        For each character *c* in Σ_M do:
            new-state = $\varnothing$.
            For each state *q* in *Q* do:
                For each state *p* such that $(q, c, p) \in \Delta$ do:
            new-state = *new-state* $\cup$ *eps*(*p*).
            Add the transition (*Q*, *c*, *new-state*) to δ'.
            If *new-state* $\notin$ *active-states* then insert it.
4. $K' = active\text{-}states.$
5. $A' = \{Q \in K' : Q \cap A \neq \varnothing\}$.

Example 5.20

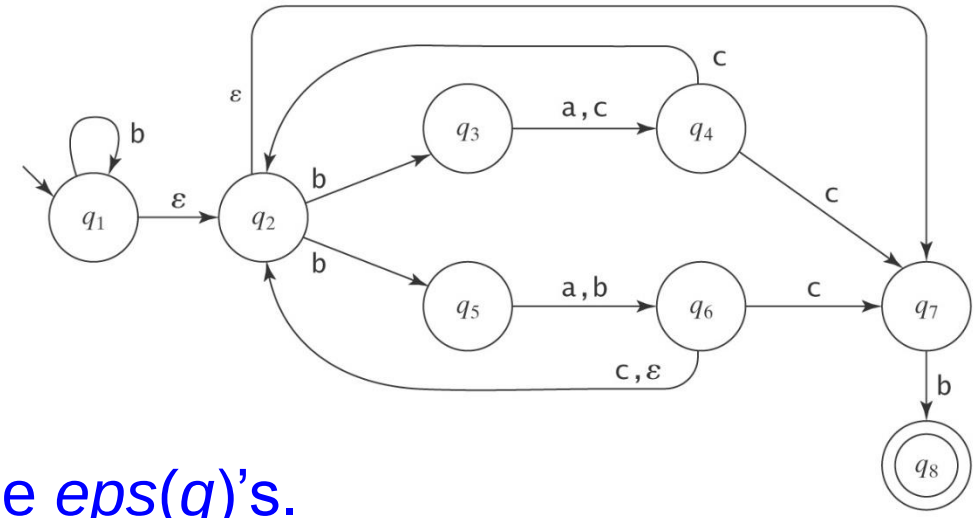
NDFSM:



DFSM?

Example 5.20

NDFSM:



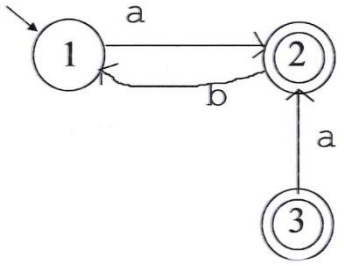
1. Compute the *eps*(*q*)'s.
2. Compute *s'* = *eps*(*s*).
3. Compute δ'.
4. Compute *K'* = a **subset** of $\mathcal{P}(K)$.
5. Compute $A' = \{Q \in K' : Q \cap A \neq \varnothing\}$.

Example 5.20

DFSM:

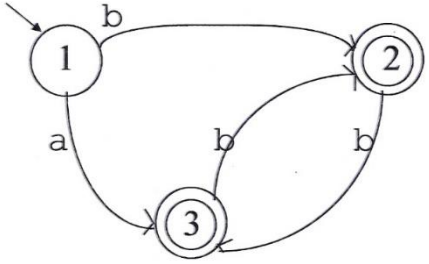
State Minimization

Step (1): Get rid of unreachable states.



State 3 is unreachable.

Step (2): Get rid of redundant states.



States 2 and 3 are redundant.

Minimizing DFSM

A DFSM M is *minimal* iff there is no other DFSM M' st L(M) = L(M') and M' as fewer states than M does.

- Given any regular language *L*, there exists a *minimal* DFSM *M* that accepts *L*.
- *M* is unique up to the naming of its states.
- Given any DFSM *M*, there exists an algorithm *minDFSM* that constructs a minimal DFSM that also accepts *L*(*M*).

Reading Assignment

Chapter 5:

Sections
5.1
5.2
5.3
5.4

In-Class Exercises

Chapter 5:

2- f & g
3
4
5
6 - b
7
9 - a