# Finite State Machines

Chapter 5

# Pattern Recognition

- Pattern Recognition:
  - Given a specified pattern string, and
  - a text string provided by the user,
  - output:
    - "yes" if the text contains pattern
    - "no" if the text does not contain pattern

- DNA Example:
  - looking for "CTT" in string of characters {C, T, A, G}

- Use a Finite State Machine

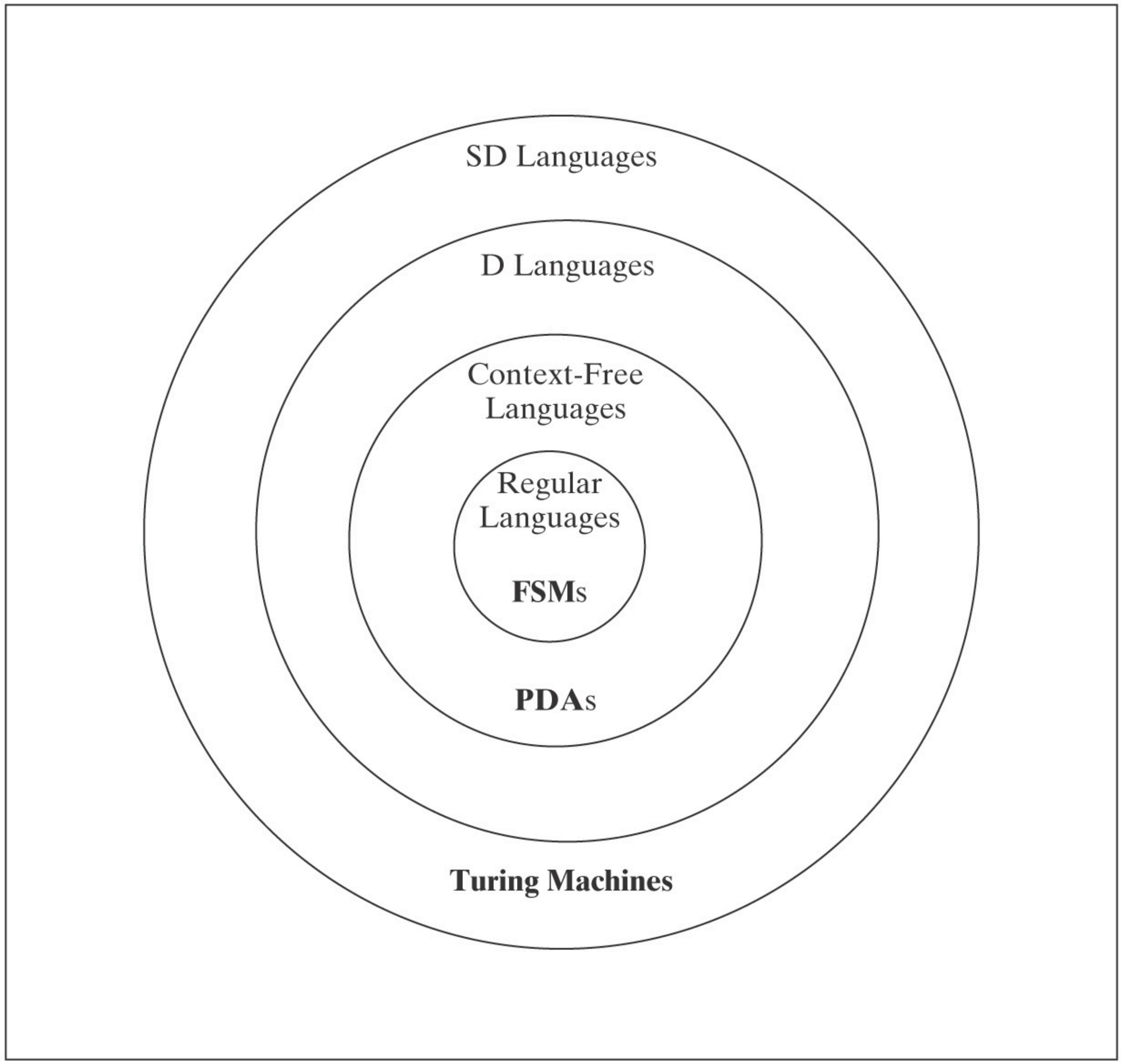
# Finite State Machine (FSM)

- Pattern matching:
  - Number of states? |P| + 1
  - Number of transitions? (|P| + 1) * (# chars in alphabet)

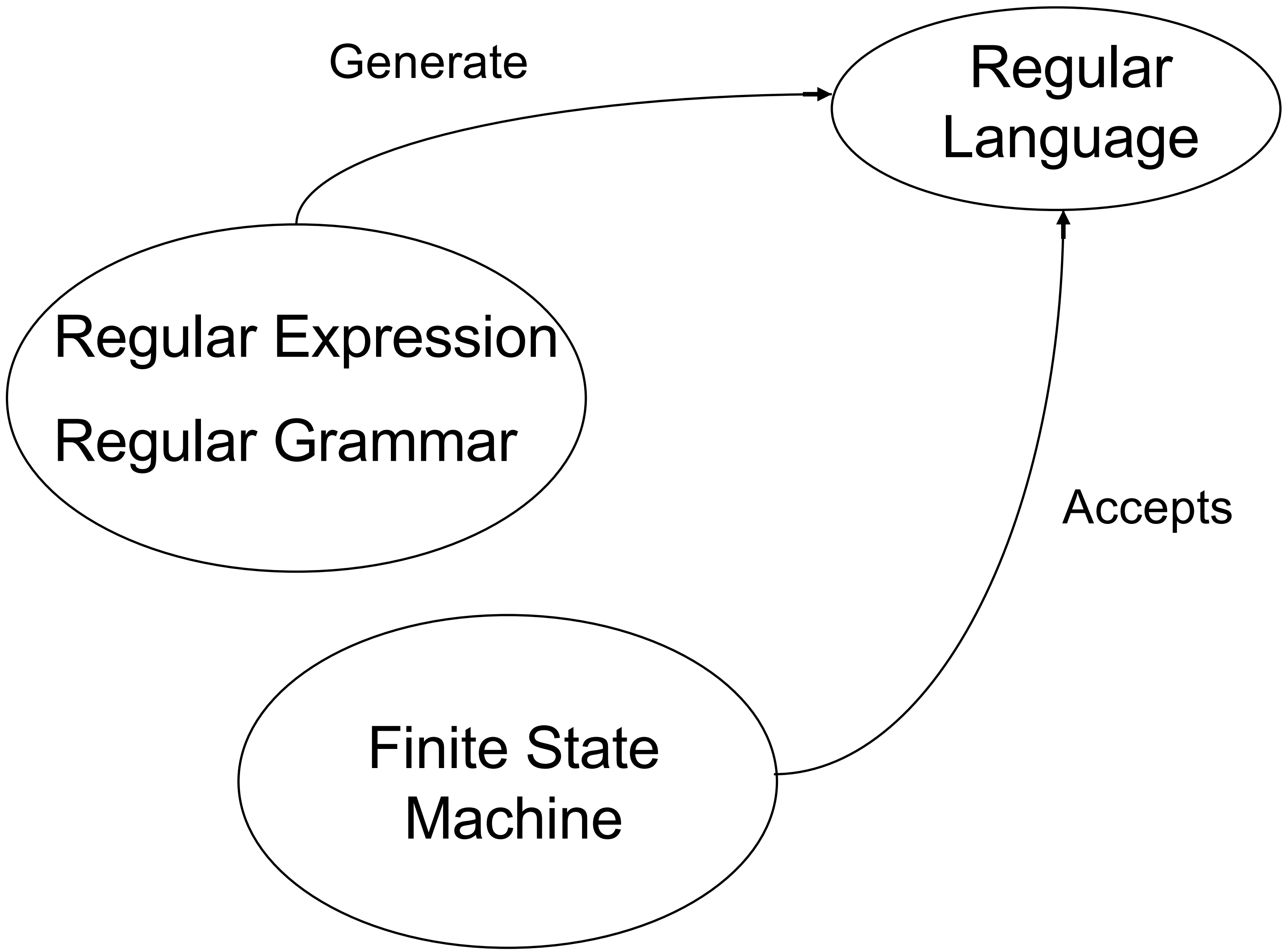
- Not so far from programming:
  - Sequential execution
  - Conditional execution
  - Iterative execution

- NOTE: Assuming text is being read in one character at a time, we only need memory to keep track of the state.

# Languages and Machines



# Regular Languages



# Finite State Machines

- Deterministic (DFSM)
- Nondeterministic (NDFSM)

# Definition of a DFSM

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

$K$ is a finite set of states

$\Sigma$ is an alphabet

$s \in K$ is the initial state

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

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

# Accepting by a DFSM

Informally, *M* accepts a string *w* iff *M* is in an accepting state (a state in *A)* when it has finished reading *w*.

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

# Configurations of DFSMs

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

$$K \times \Sigma^*$$

It captures the two things that determine the DFSM's future behavior:

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

The ***initial configuration*** of a DFSM on input *w* is $(s, w)$

# 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
- $\delta\, (q, a) = q'$

The *yields-in-zero-or-more-steps relation* **$|-_M$ *** is the reflexive, transitive closure of $|-_M$

# Computations Using FSMs

A ***computation*** byDFSM $M$ 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 $(q, \varepsilon)$, for some state $q \in K_M$,

- $C_0 \vdash_M C_1 \vdash_M C_2 \vdash_M \ldots \vdash_M C_n$.

# 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:

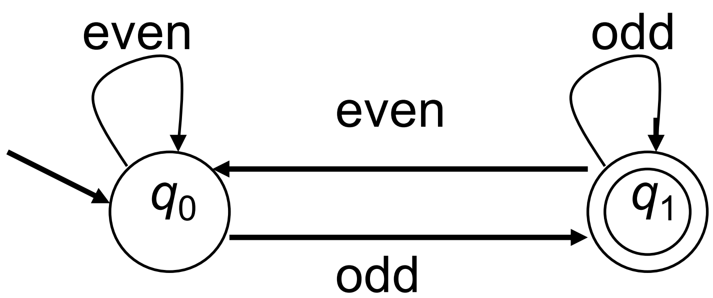
$(s, w) \vdash_M{}^* (q, \varepsilon)$, for some $q \notin A_M$.

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

***Theorem:*** Every DFSM $M$, on input $s$, halts in $|s|$ steps.

# An Example Computation

An FSM to accept odd integers:




On input 235, the configurations are:

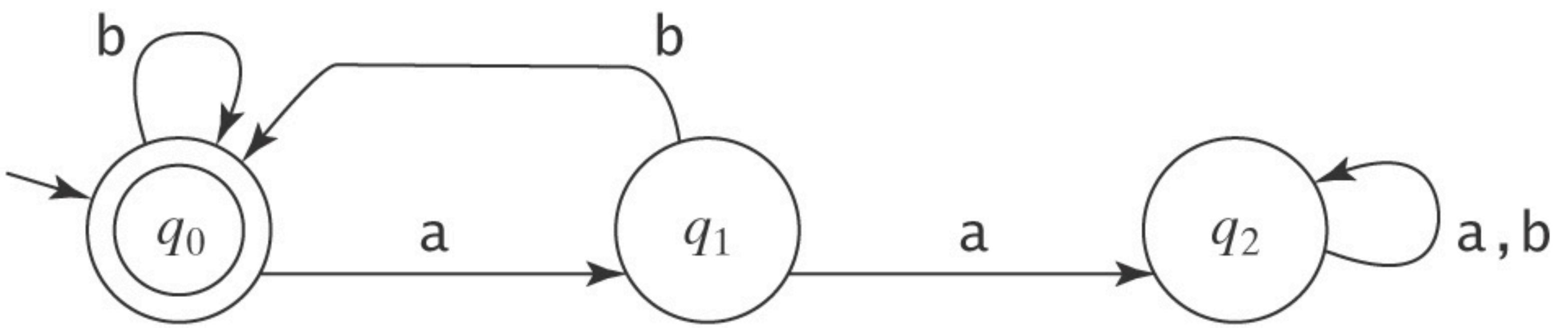
| | | |
|---|---|---|
| $(q_0, 235)$ | $\vdash_M$ | $(q_0, 35)$ |
| $(q_0, 35)$ | $\vdash_M$ | $(q_1, 5)$ |
| $(q_1, 5)$ | $\vdash_M$ | $(q_1, \varepsilon)$ |

Thus $(q_0, 235) \vdash_M^* (q_1, \varepsilon)$

# A Simple FSM Example

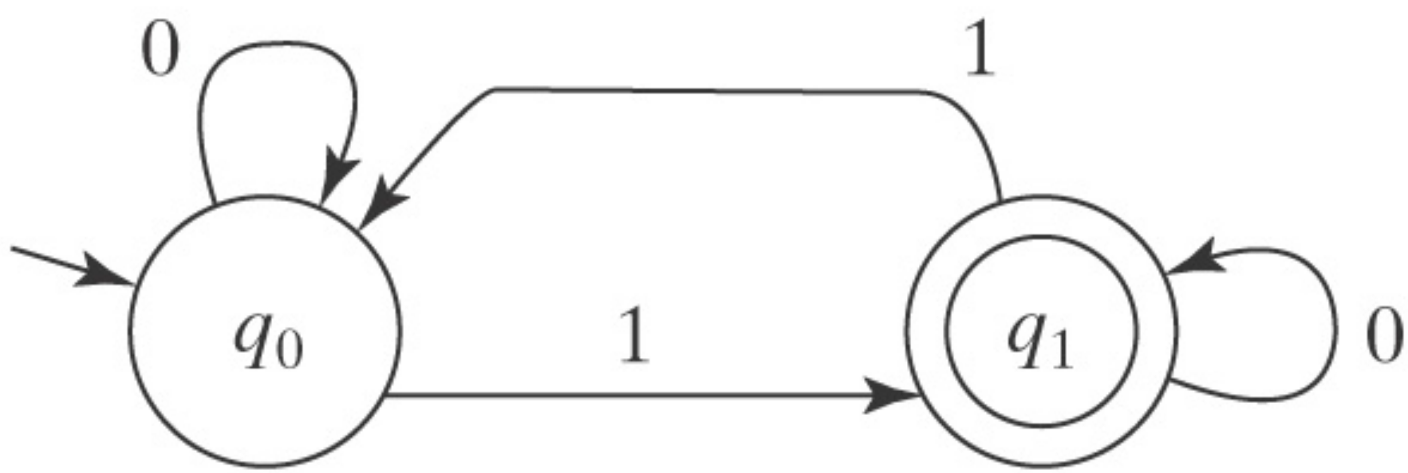
$L = \{w \in \{\texttt{a}, \texttt{b}\}^*$ :

every a is immediately followed by a b}.



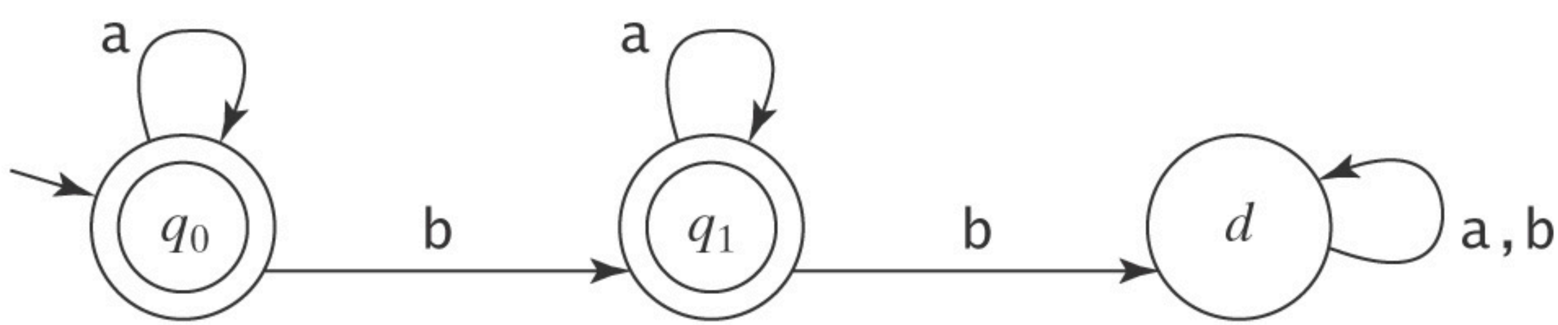
# Parity Checking

$L = \{w \in \{0, 1\}^* : w \text{ has odd parity}\}$.



# No More Than One b

$L = \{w \in \{\texttt{a}, \texttt{b}\}^* : w$ has no more than one $\texttt{b}\}$.



# Checking Consecutive Characters

$L = \{w \in \{\texttt{a}, \texttt{b}\}^*$ :

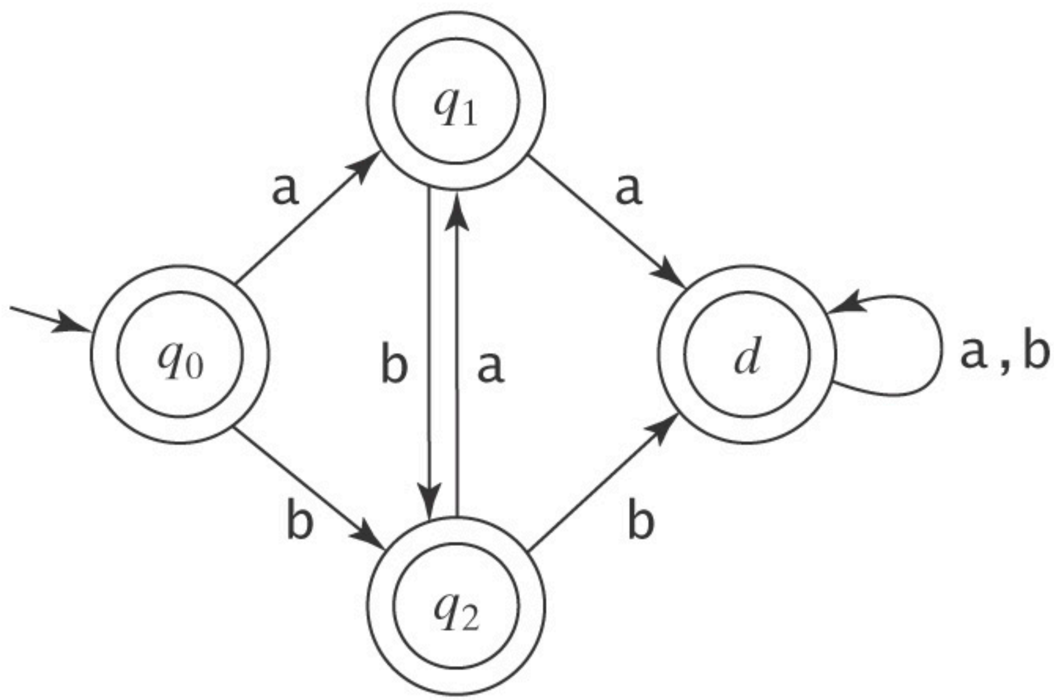
no two consecutive characters are the same}.

Exercise

# Checking Consecutive Characters

$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* :$

no two consecutive characters are the same}.




What's the mistake?

# Even Segments of a's

$L$ =

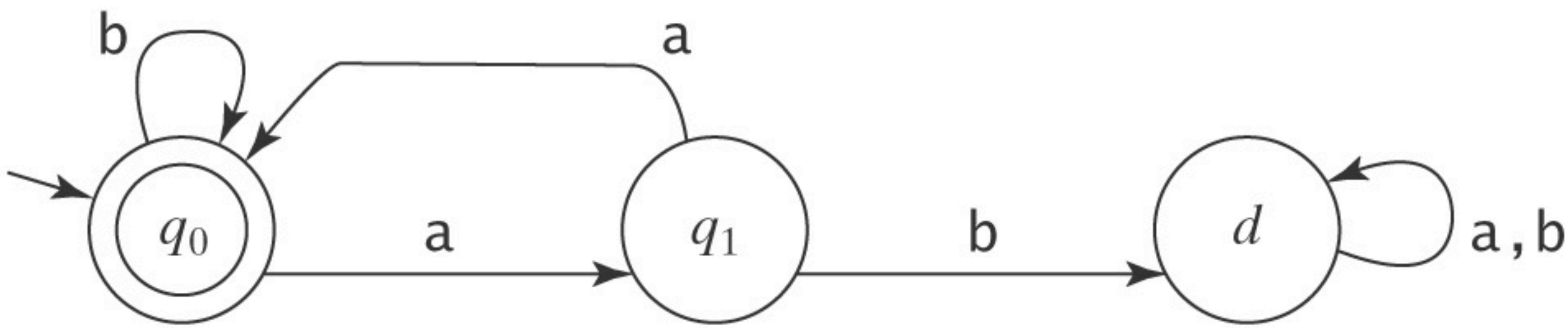
{$w \in$ {a, b}* : every a region in $w$ is of even length}

Exercise

# Even Segments of a's

$L$ =

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

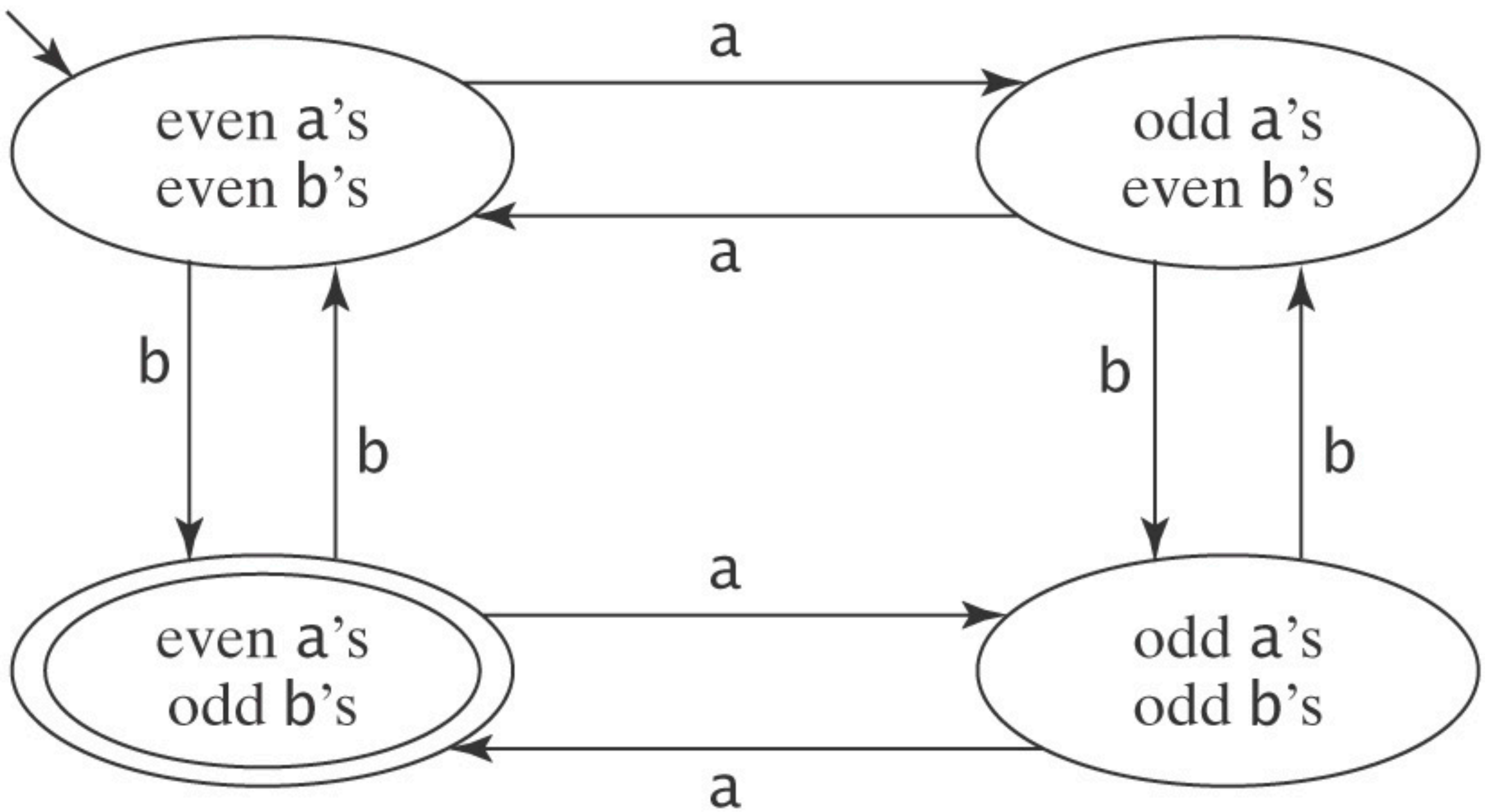


# Even # of a's and Odd # of b's

Let $L = \{w \in \{\texttt{a}, \texttt{b}\}^* : w$ contains an even number of a's and an odd number of b's$\}$

Exercise

# Even # of a's and Odd # of b's



# Programming FSMs

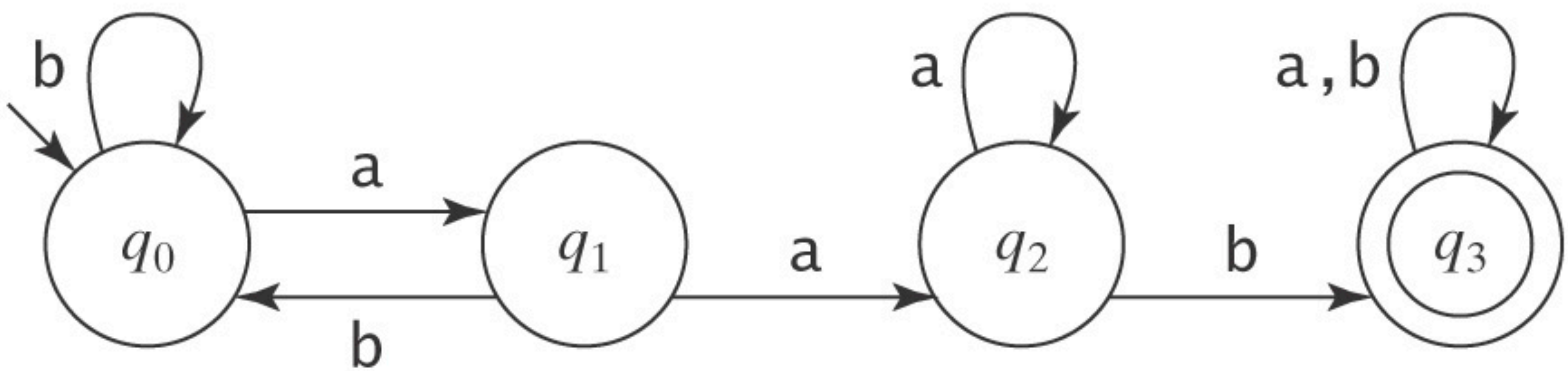
$L = \{w \in \{\texttt{a}, \texttt{b}\}^* : w$ does not contain the substring $\texttt{aab}\}$.

Hint: Start with a machine for $\neg L$:

# Programming FSMs

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

Start with a machine for $\neg L$:




How must it be changed?

# Homework

- Chapter 5
  - 2)
    - a)
    - e)
    - i) (DFSM and NDFSM)
    - m) (DFSM and NDFSM)
  - 4) all

# Definition of an NDFSM

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

$K$ is a finite set of states

$\Sigma$ is an alphabet

$s \in K$ is the initial state

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

$\Delta$ is the transition ***relation***. It is a finite subset of

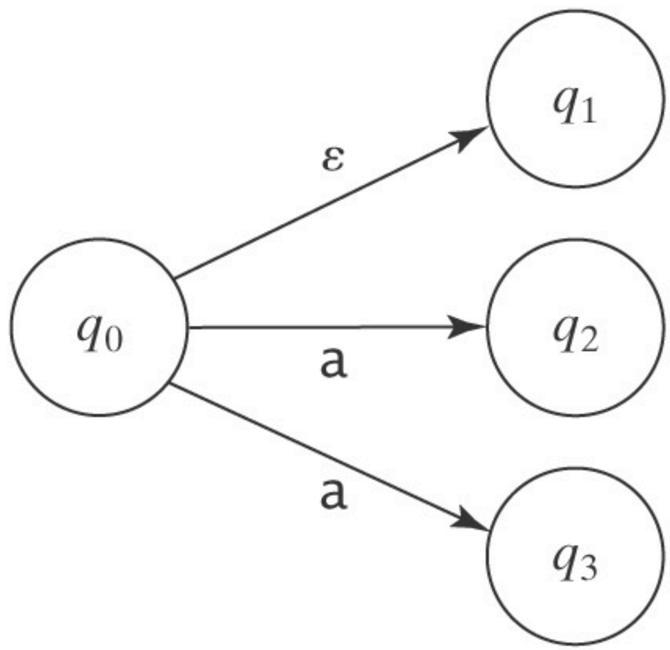
$$(K \times (\Sigma \cup \{\varepsilon\})) \times K$$

# NDFSM Transitions

Transitions are more complicated:

- "epsilon" transitions, i.e. change state without reading a character of input
- multiple transitions from a state for a given character
- no transition for a character from a state, i.e. the computation can "block"

# Sources of Nondeterminism



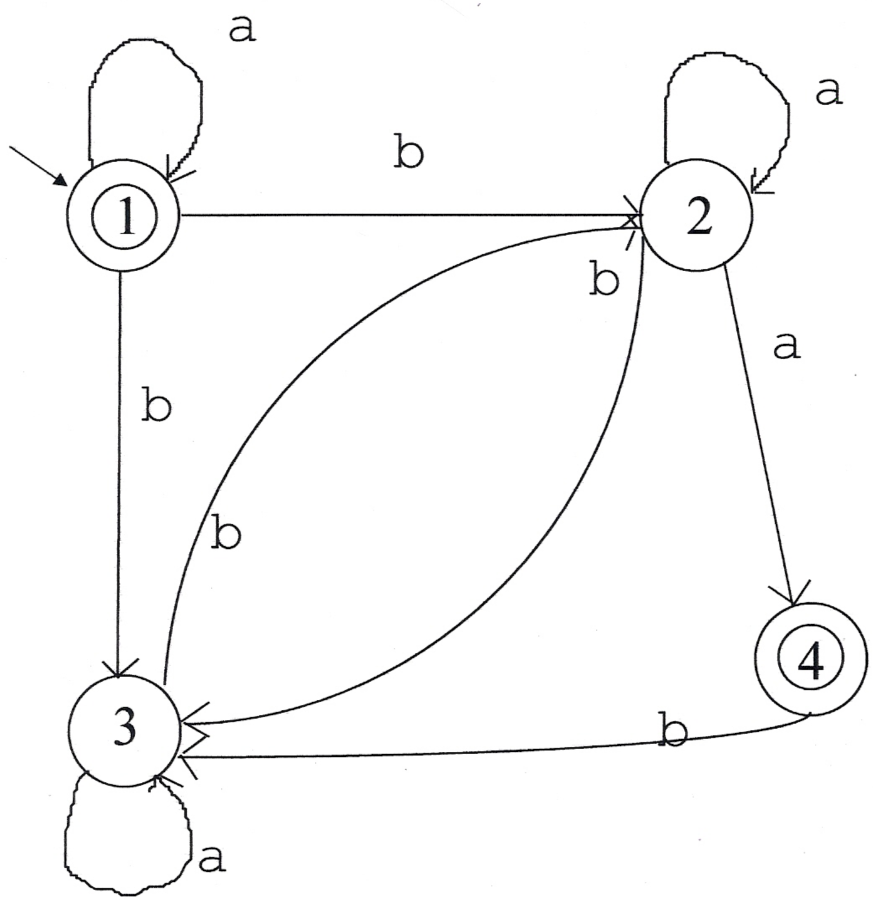
# Accepting by an NDFSM

*M* accepts a string *w* iff there exists *some path* that ends in an accepting state with the entire input read/consumed.

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

Sometimes simpler and smaller than a DFSM that recognizes the same language.

# Analyzing Nondeterministic FSMs




Does this FSM accept:

`baaba`

Remember: we just have to find one accepting path.

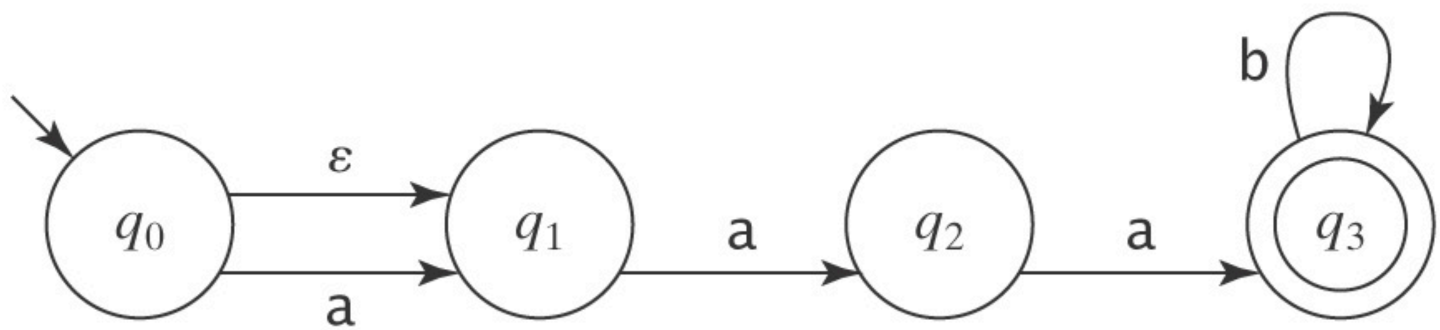
# NDFSM Example

$L = \{w \in \{a, b\}^* : w$ is made up of all a's or all b's$\}$.

- DFSM?
- NDFSM?

# Optional Substrings

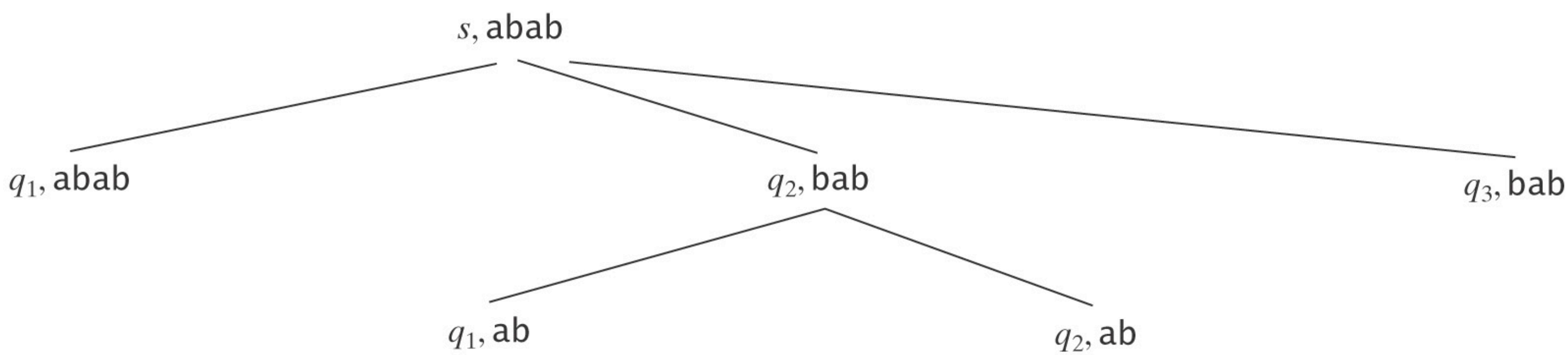
$L = \{w \in \{a, b\}^* : w$ is made up of an optional a followed by aa followed by zero or more b's$\}$.



# Analyzing Nondeterministic FSMs

Two ways to think about it:

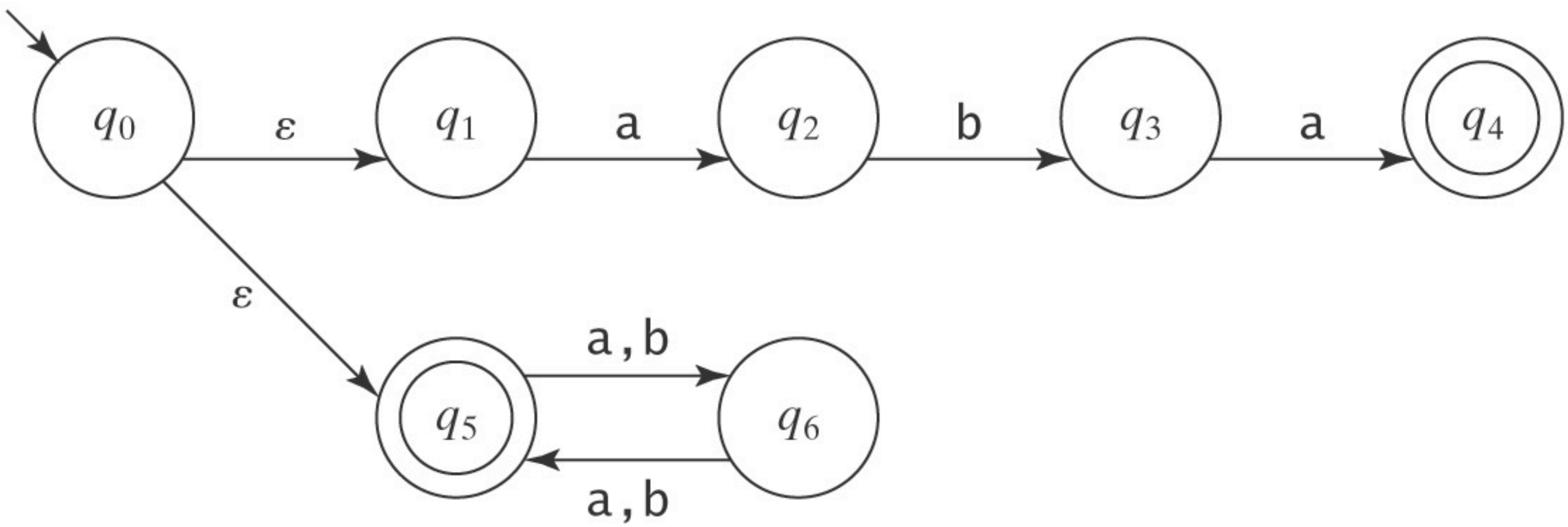
1) Explore a search tree:



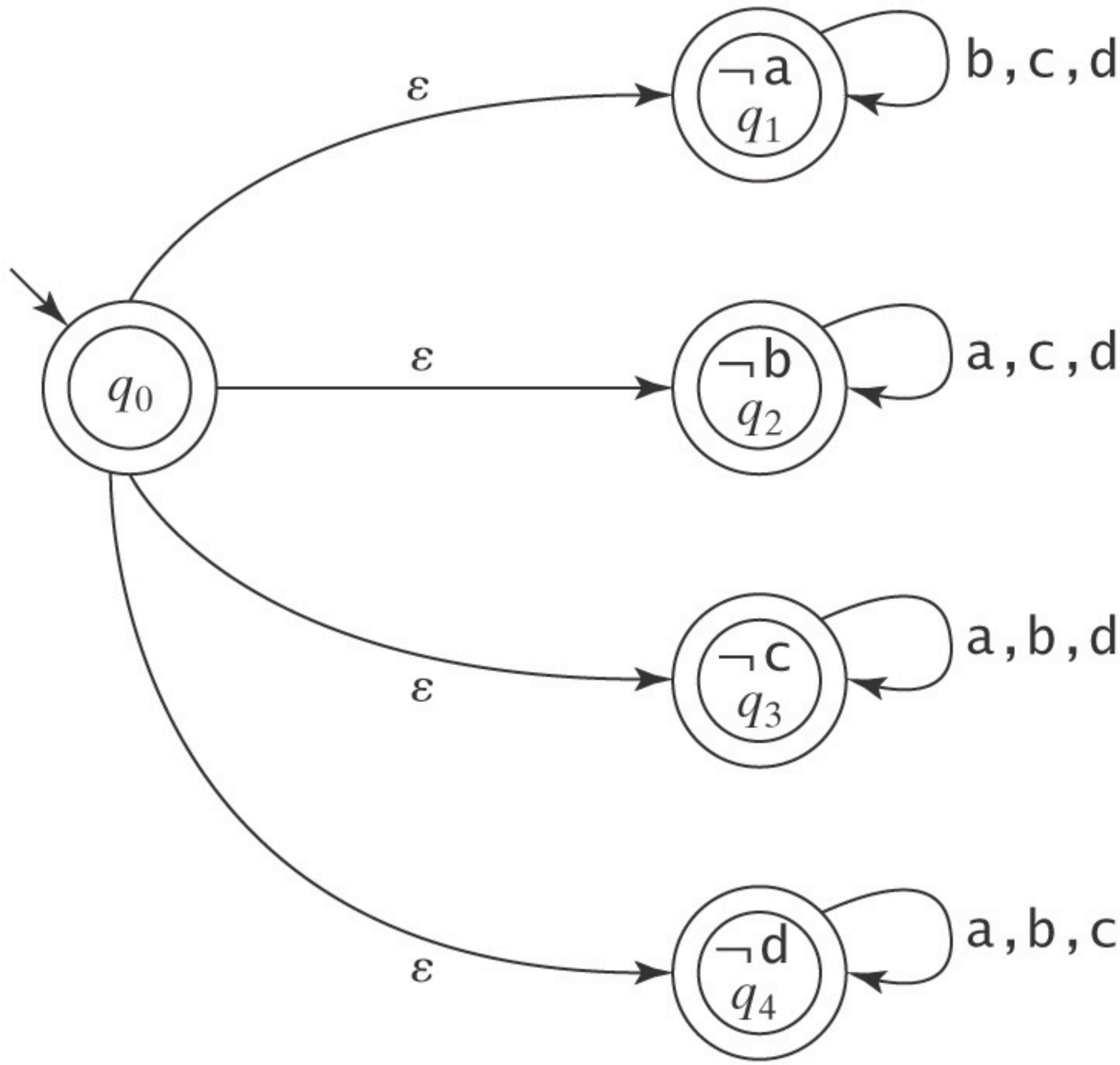

2) Follow all paths in parallel (sets of states M could be in)

# Multiple Sublanguages

$L = \{w \in \{\texttt{a}, \texttt{b}\}^* : w = \texttt{aba} \text{ or } |w| \text{ is even}\}$.



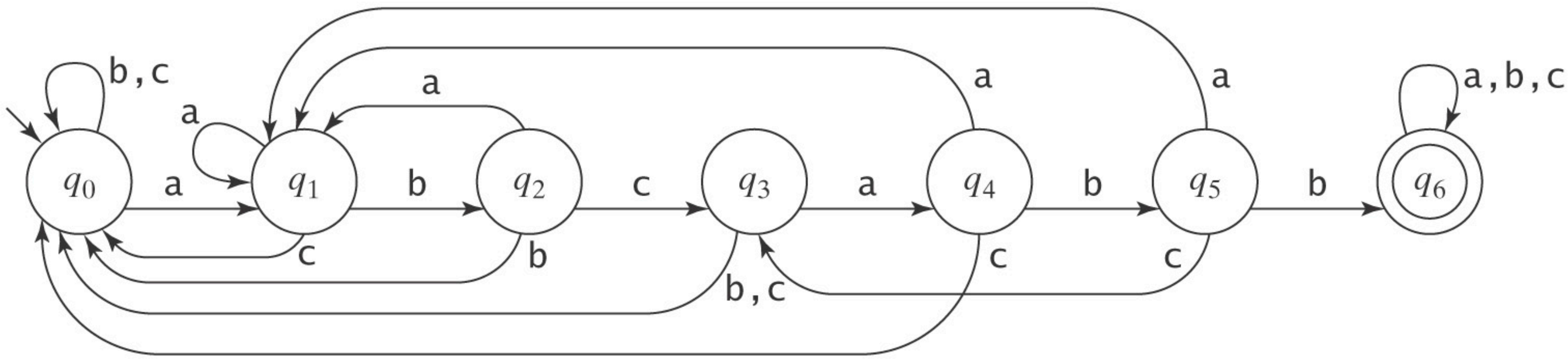
# The Missing Letter Language



# Pattern Matching

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

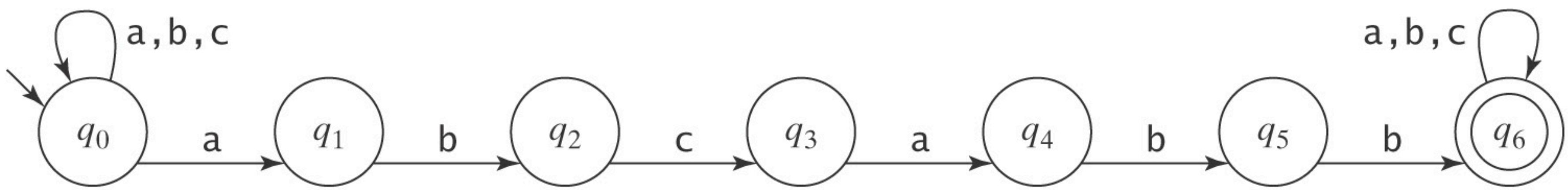
A DFSM:



# Pattern Matching

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

An NDFSM:



# Multiple Patterns

$L = \{w \in \{\texttt{a}, \texttt{b}\}^* : \exists x, y \in \{\texttt{a}, \texttt{b}\}^*$

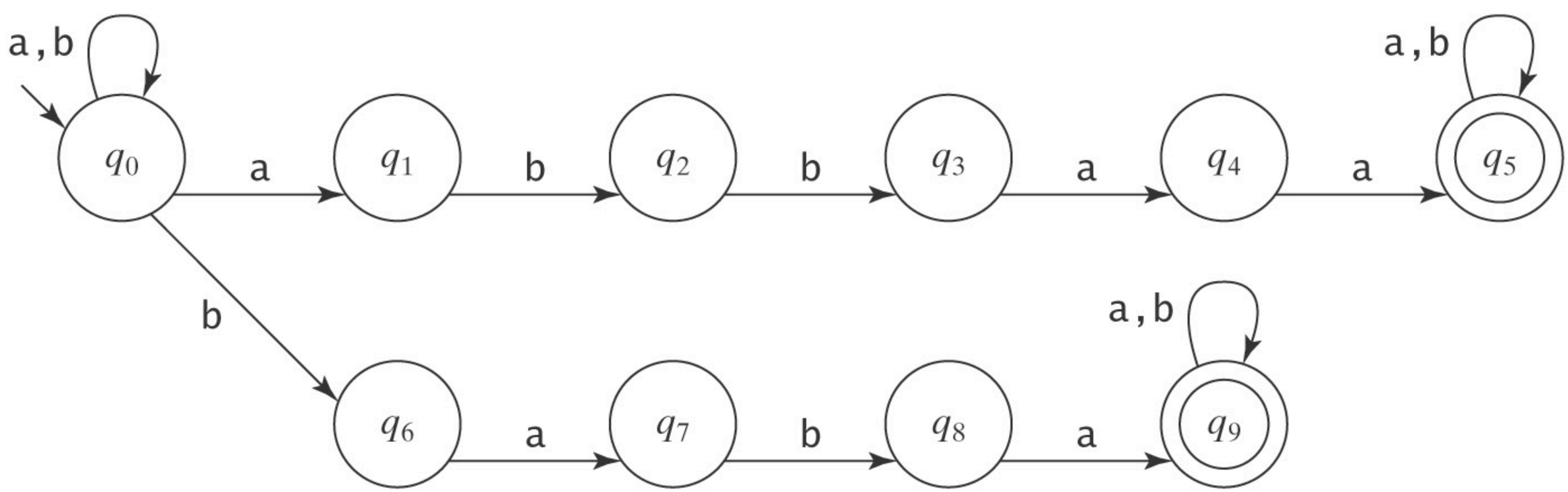
$((w = x\ \texttt{abbaa}\ y) \vee (w = x\ \texttt{baba}\ y))\}.$

Exercise

# Multiple Patterns

$L = \{w \in \{\texttt{a}, \texttt{b}\}^* : \exists x, y \in \{\texttt{a}, \texttt{b}\}^*$

$((w = x\ \texttt{abbaa}\ y) \vee (w = x\ \texttt{baba}\ y))\}$.



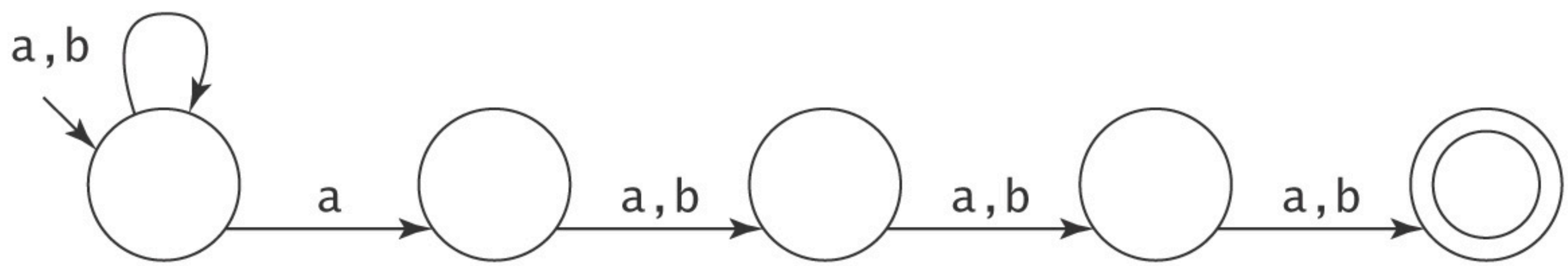
# Checking from the End

$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* :$
the fourth to the last character is $\mathtt{a}\}$

Exercise

# Checking from the End

$L = \{w \in \{a, b\}^* :$ the fourth to the last character is $a\}$



# Homework

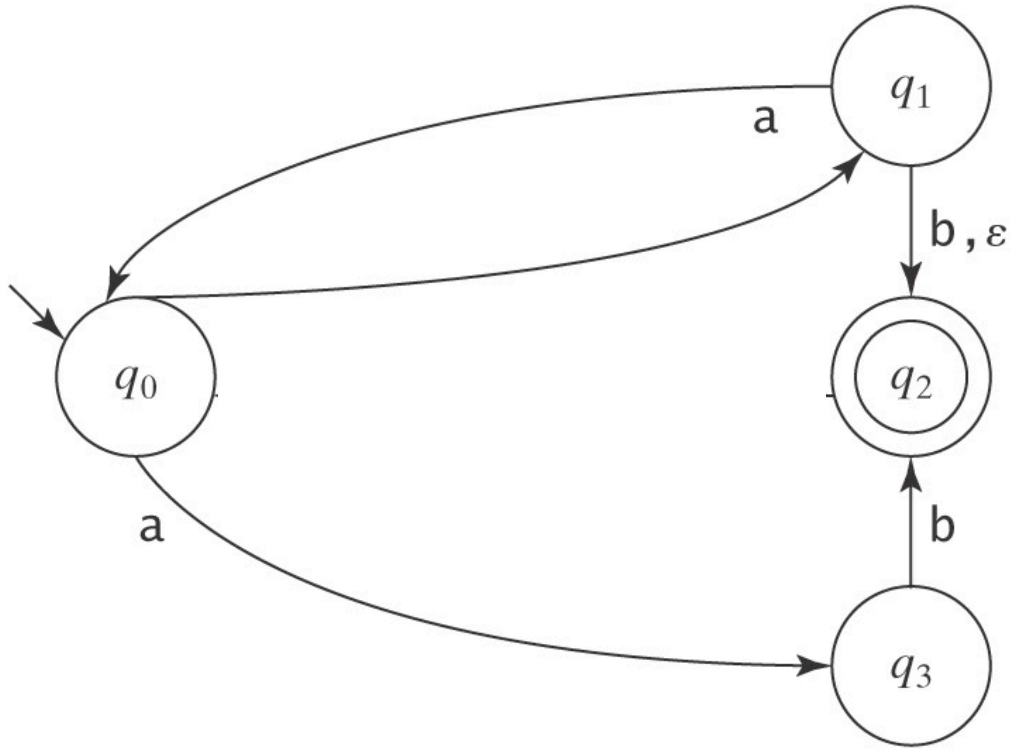
- Chapter 5
  - 2)
    - a)
    - e)
    - i) (DFSM and NDFSM)
    - m) (DFSM and NDFSM)
  - 4) all
  - 5) all
  - 6) f

# Dealing with ε Transitions

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

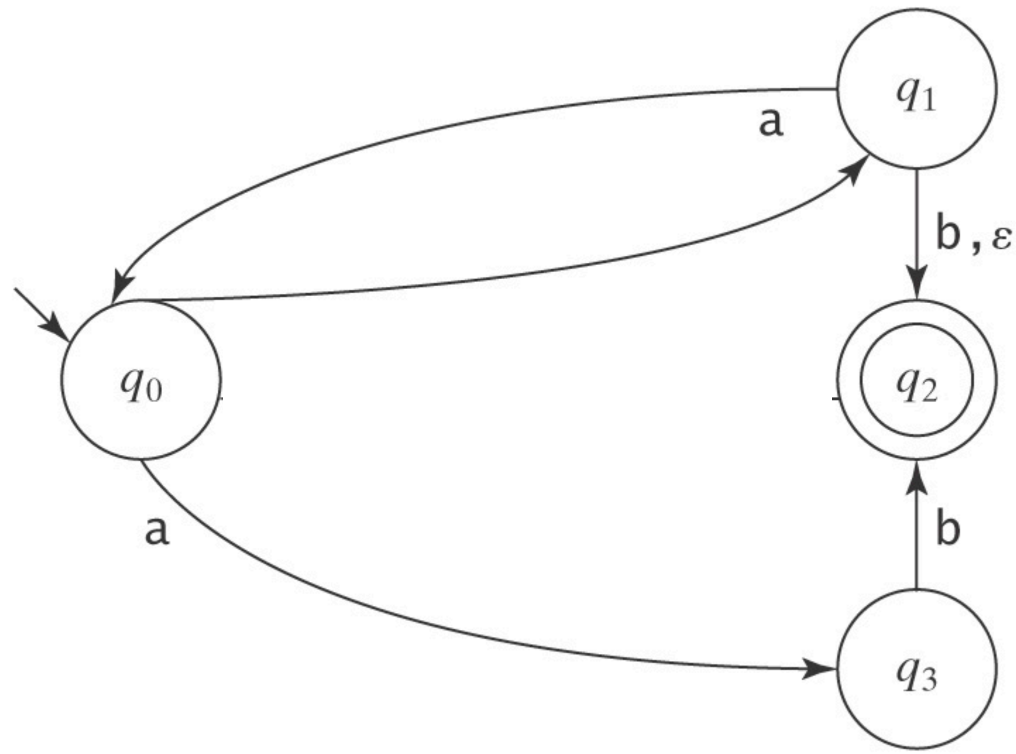
$eps(q)$ is the closure of $\{q\}$ under the relation $\{(p, r) :$ there is a transition $(p, \varepsilon, r) \in \Delta\}$.

# An Example of *eps*




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

# An Example of *eps*




$eps(q_0) = \{ q_0, q_1, q_2 \}$
$eps(q_1) = \{ q_1, q_2 \}$
$eps(q_2) = \{ q_2 \}$
$eps(q_3) = \{ q_3 \}$

# Simulating an NDFSM

**simulateNDFSM** (NDFSM M, string w) =

1. *current-state* = *eps*(*s*).
2. While any input symbols in *w* remain to be read do:
   1. *c* = get-next-symbol(*w*).
   2. *next-state* = $\varnothing$.
   3. For each state *q* in *current-state* do:

      For each state *p* such that (*q*, *c*, *p*) $\in \Delta$ do:

      *next-state* = *next-state* $\cup$ *eps*(*p*).
   4. *current-state* = *next-state*.
3. If *current-state* contains any states in *A*, accept. Else reject.

# Nondeterministic and Deterministic FSMs

Clearly: {Languages accepted by a DFSM} $\subseteq$ {Languages accepted by a NDFSM}

More interestingly:

***Theorem**:* For each NDFSM, there is an equivalent DFSM.

*HOW?* **(look back at "Simulating an NDFSM")**

# General Idea

- Sets of states; takes care of:
  - Epsilon transitions
  - Multiple transitions on same character
- Can take care of no transition situations with trap states
- NOTE: The number of states in the DFSM can grow exponentially!

# Nondeterministic and Deterministic FSMs

***Theorem****:* For each NDFSM, there is an equivalent DFSM.

***Proof****:* By construction:

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

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

$s' = eps(s)$

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

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

# An Algorithm for Constructing the Deterministic FSM

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

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

3. Compute $\delta'$.

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

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

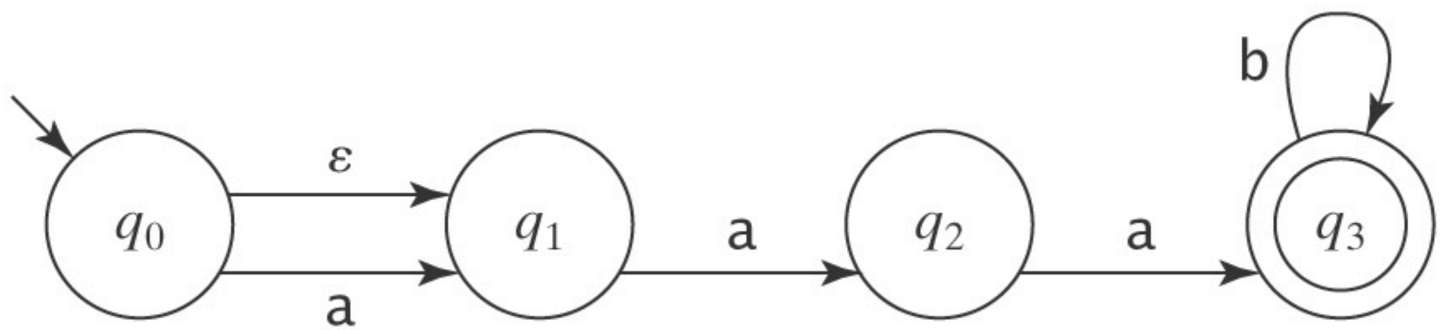
# The Algorithm NDFSMtoDFSM

**NDFSMtoDFSM** (NDFSM M) =

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

# Exercise

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



# Homework

- Chapter 5
  - 2)
    - a)
    - e)
    - i) (DFSM and NDFSM)
    - m) (DFSM and NDFSM)
  - 4) all
  - 5) all
  - 6) f
  - 9) a

# Homework

- Chapter 5
  2)
    a)
    e)
    i) (DFSM and NDFSM)
    m) (DFSM and NDFSM)
  4) all
  5) all
  6) f
  9) a

# Prove DFSM <-> NDFSM?

- DFSM -> NDFSM?
  - Trivial (why?)
- NDFSM -> DFSM?
  - Think about it for next time
  - Hint: Prove equivalent computation on any possible string *of any length*