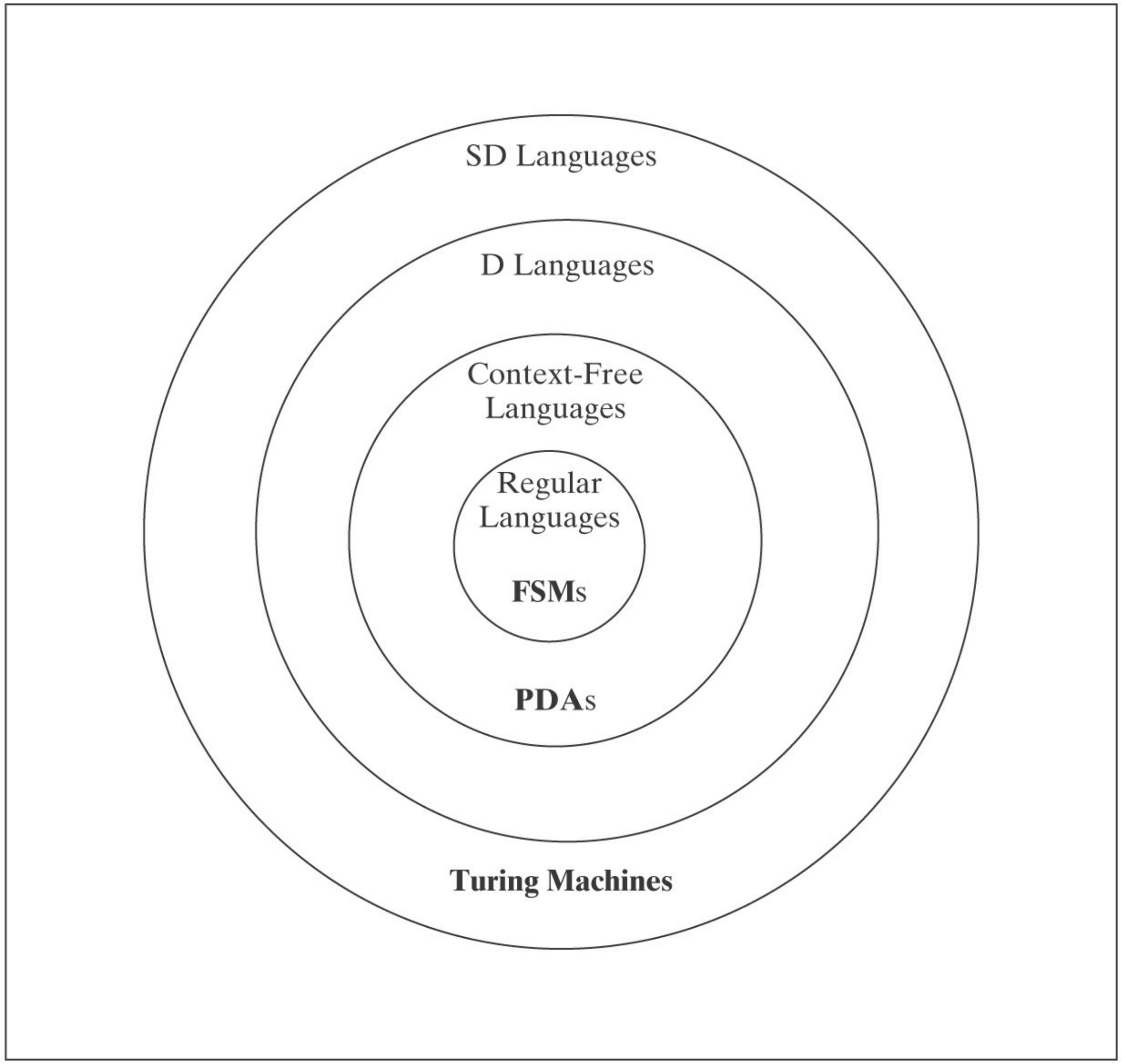
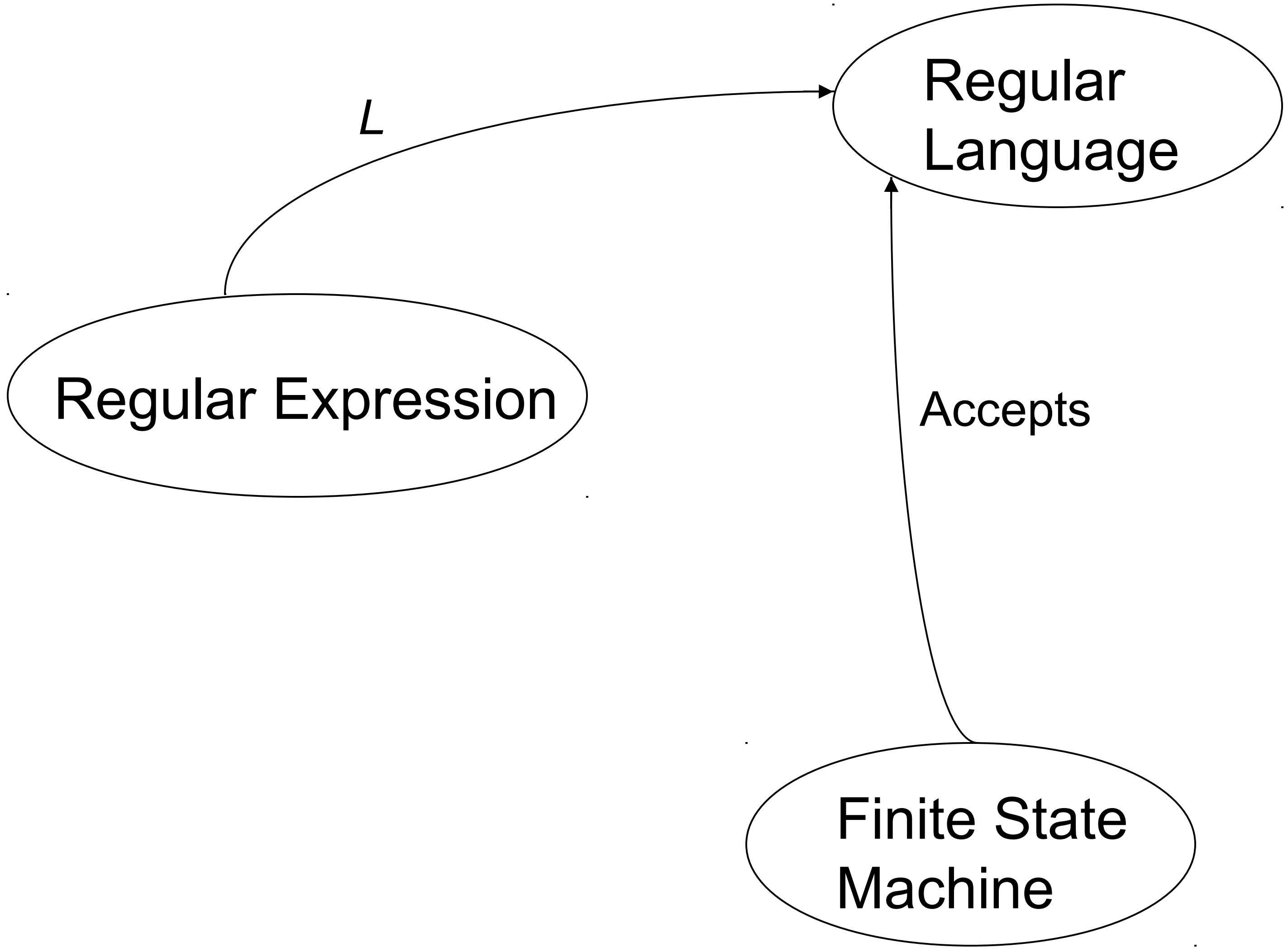
Finite State Machines

Chapter 5

Languages and Machines

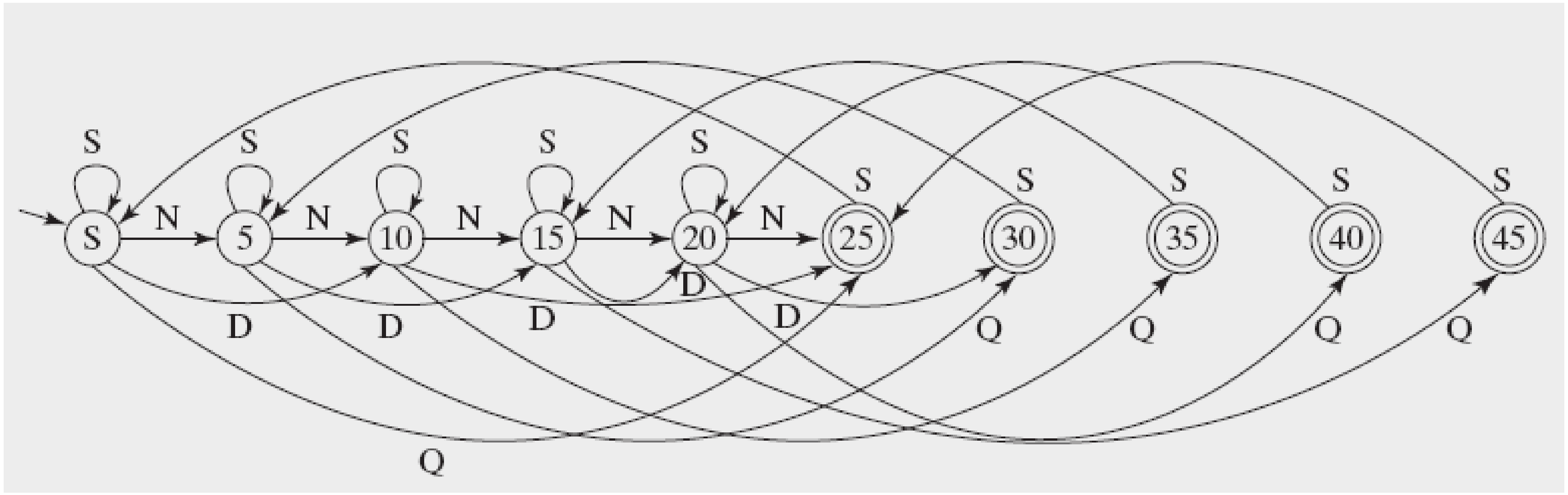


Regular Languages



Finite State Machines

- An FSM for a vending machine:
    - One soda (S) is $.25
    - No pennies
    - Max credit $.45



Definition of a DFSM

$M = (K, \Sigma, \delta, s, A)$, 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 function from $(K \times \Sigma)$ to K

Accepting by a DFSM

Informally, *M* **accepts a string** *w* iff *M* winds up in some element of *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 *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, 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 FSMs

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 $q \in K_M$,
- $C_0 \vdash_M C_1 \vdash_M C_2 \vdash_M \ldots \vdash_M C_n$.

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:

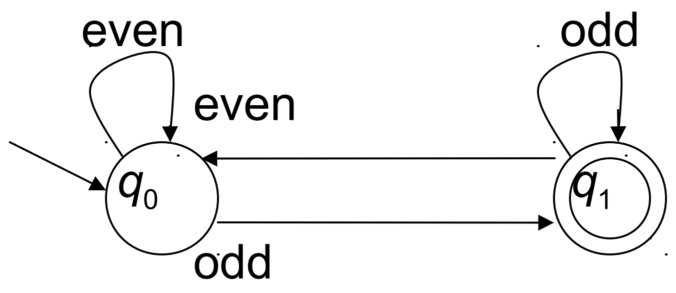
$(s, w) \vdash_M{}^* (q, \varepsilon)$, for some $q \notin A$.

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:

$(q_0, 235) \mid\!-_M (q_0, 35)$

$\mid\!-_M$

$\mid\!-_M$

Thus $(q_0, 235) \mid\!-_M{}^* (q_1, \varepsilon)$

Regular Languages

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

A Very Simple Example

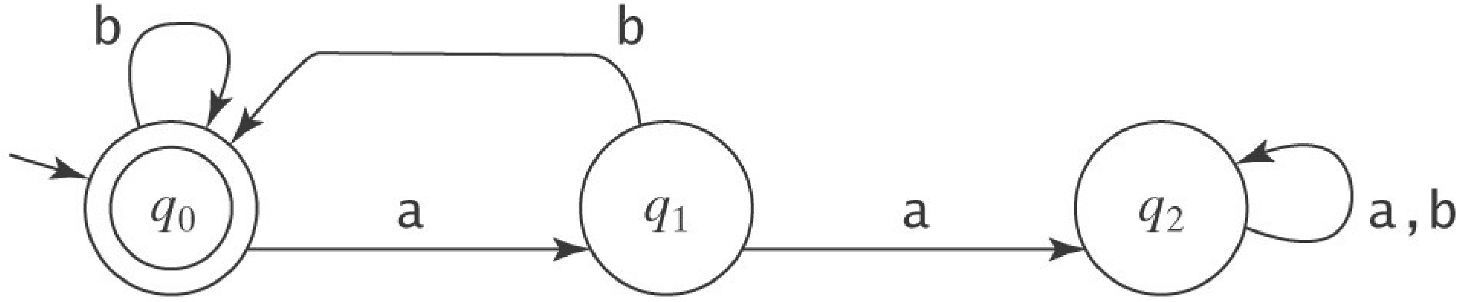
$L = \{w \in \{\texttt{a}, \texttt{b}\}^* :$

every $\texttt{a}$ is immediately followed by a $\texttt{b}\}$.

A Very Simple Example

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

every a is immediately followed by a b$\}$.

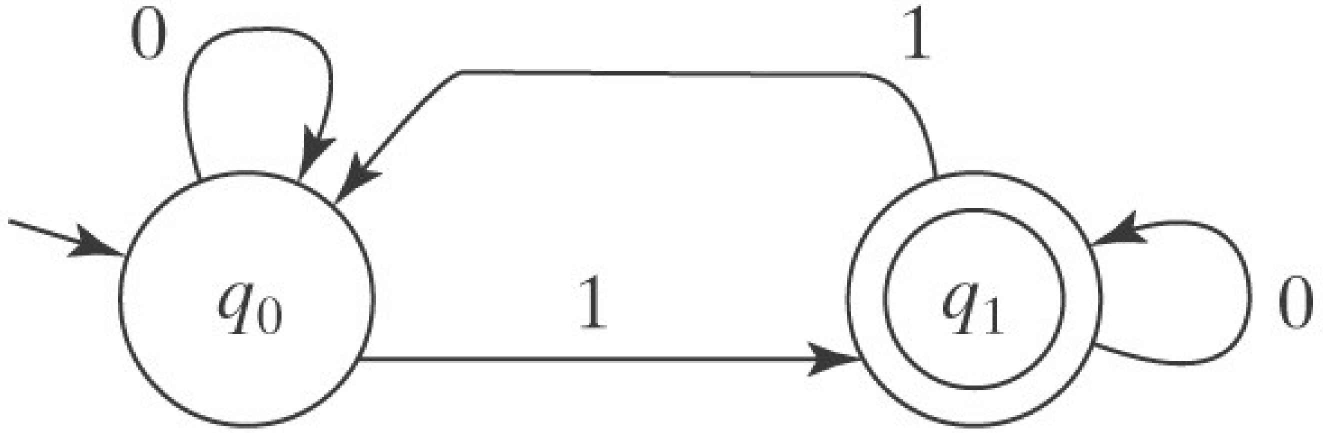


Parity Checking

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

Parity Checking

L = {*w* ∈ {`0`, `1`}* : *w* has odd parity}.

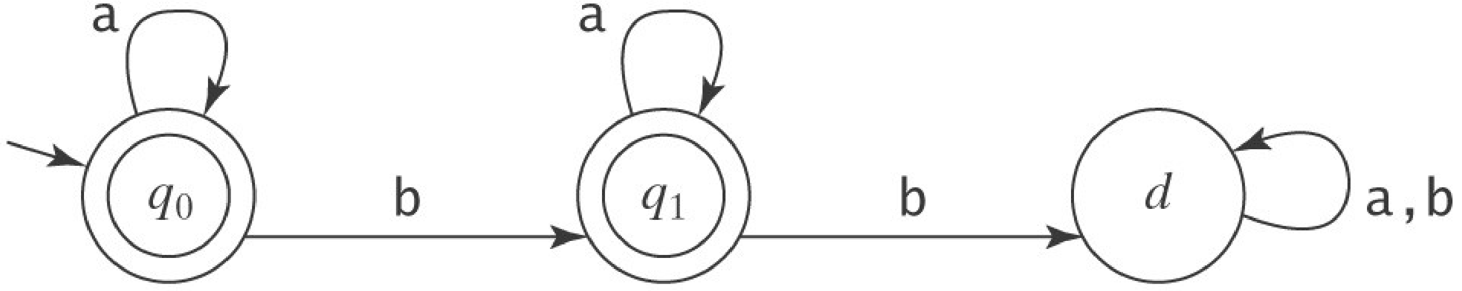


No More Than One b

L = {$w \in$ {`a`, `b`}* : w contains at most one `b`}.

No More Than One b

$L = \{w \in \{\texttt{a}, \texttt{b}\}^* : w$ contains at most one $\texttt{b}\}$.



Checking Consecutive Characters

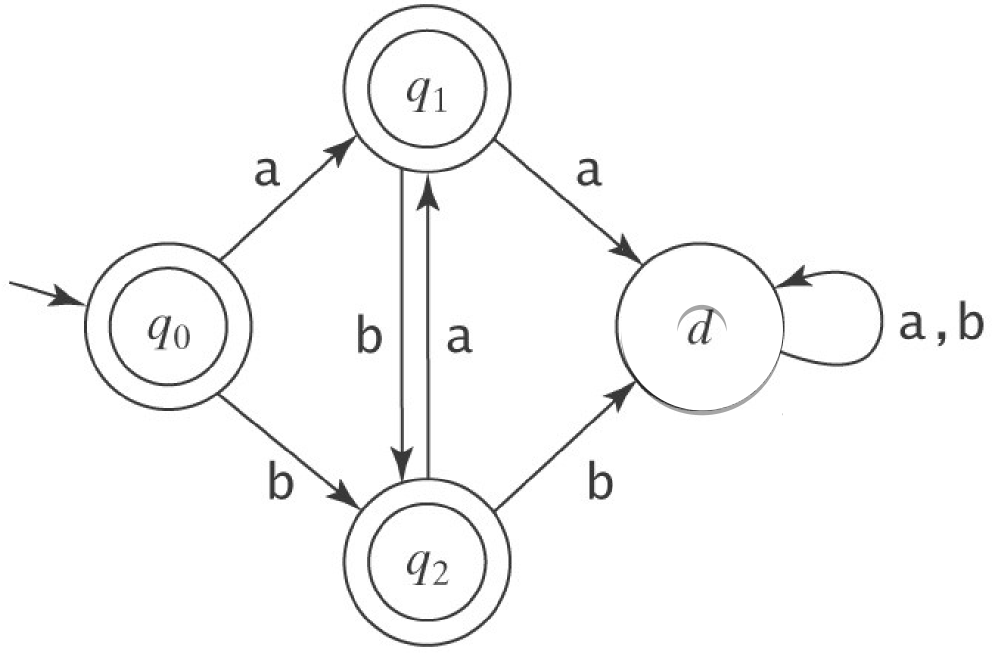
$L = \{w \in \{\texttt{a}, \texttt{b}\}^* :$

no two consecutive characters are the same$\}$.

Checking Consecutive Characters

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

no two consecutive characters are the same$\}$.



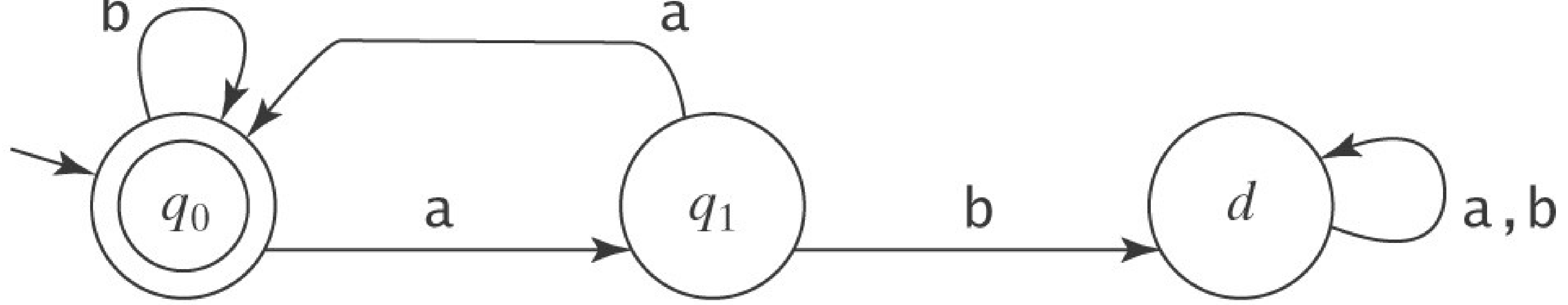
Dead States

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

Dead States

L =

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


$$\begin{tikzcd}
\arrow[r] & q_0 \arrow[loop above, "b"] \arrow[r, "a"] & q_1 \arrow[l, bend right, "a"'] \arrow[r, "b"] & d \arrow[loop right, "a,b"]
\end{tikzcd}$$

Dead States

L =

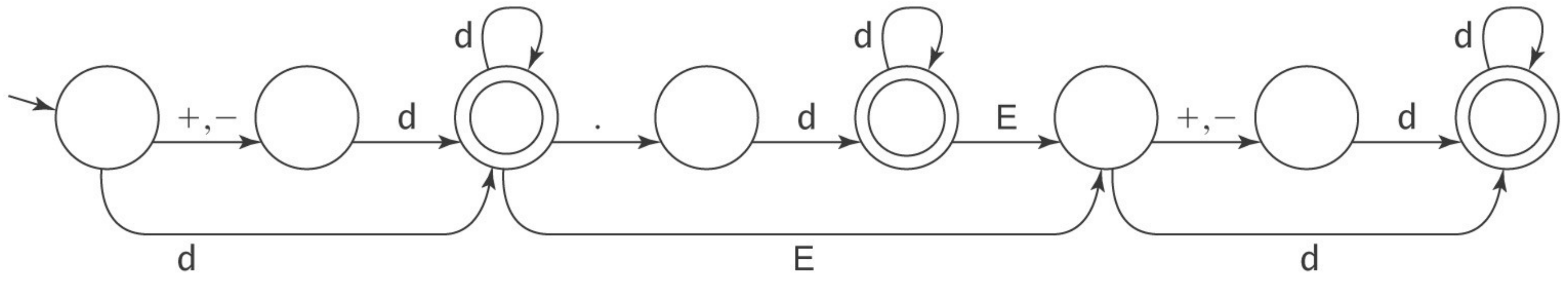
{$w \in$ {a, b}* : every b in w is surrounded by a's}

The Language of Floating Point Numbers is Regular

Example strings:

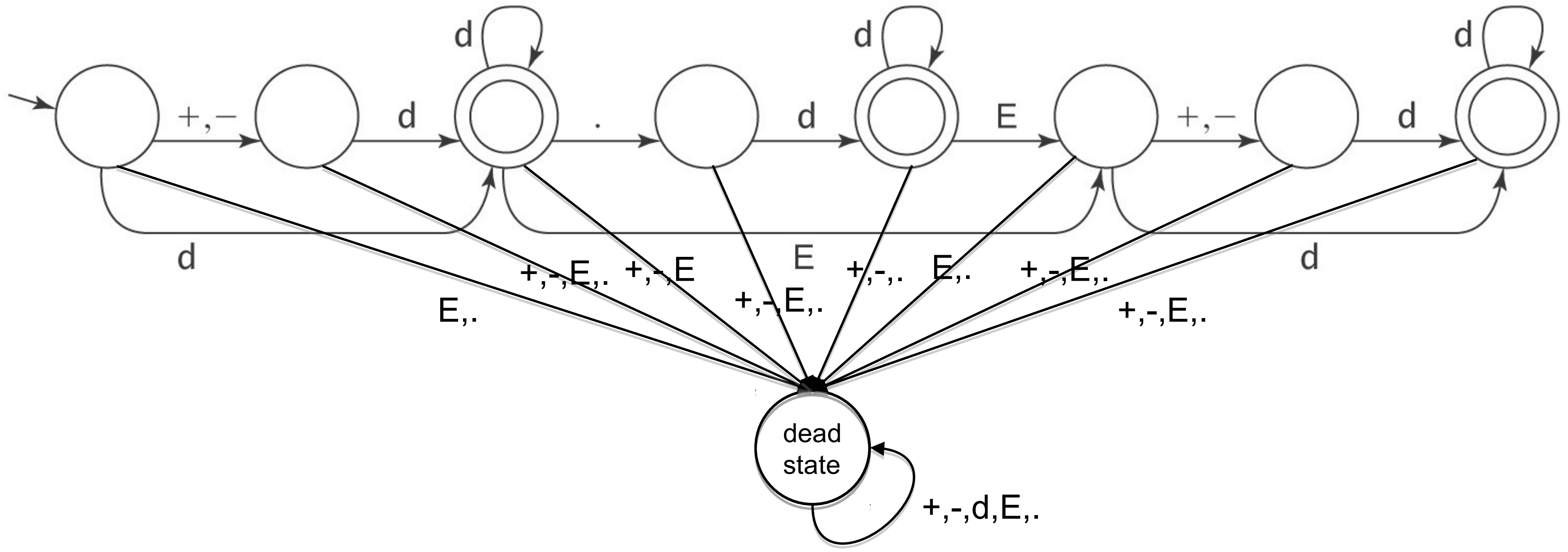
`+3.0, 3.0, 0.3E1, 0.3E+1, -0.3E+1, -3E8`

The language is accepted by the DFSM:

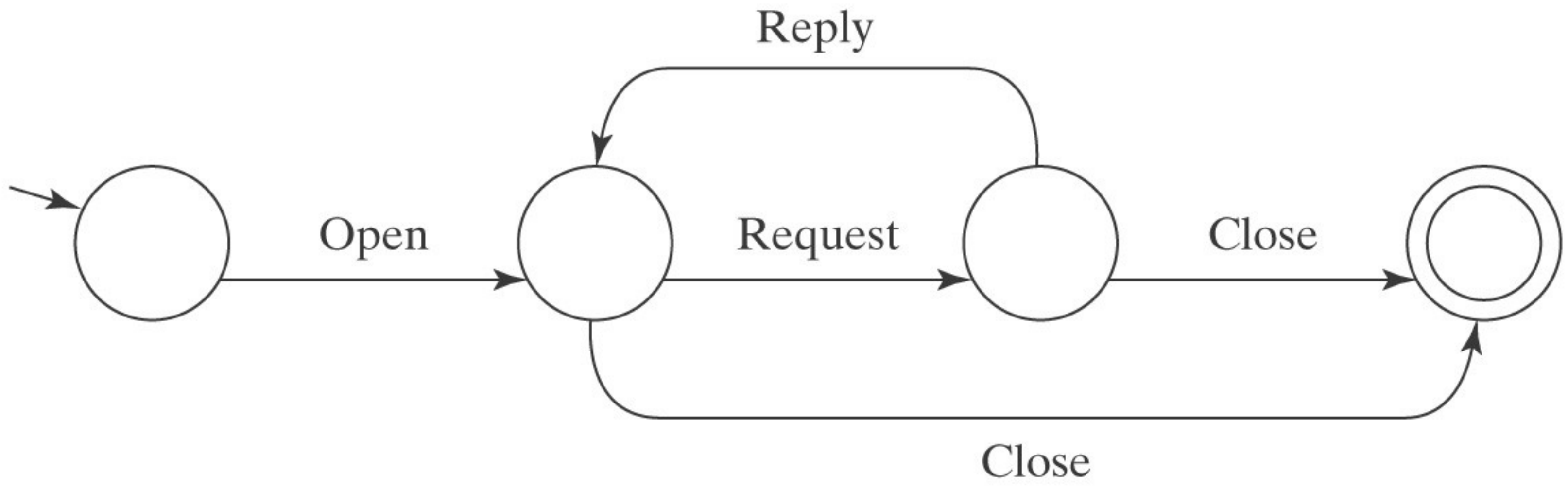


DFSMs are complete

- The transition function is always complete
- The missing transitions are leading to a “dead” state
- Often not shown for clarity

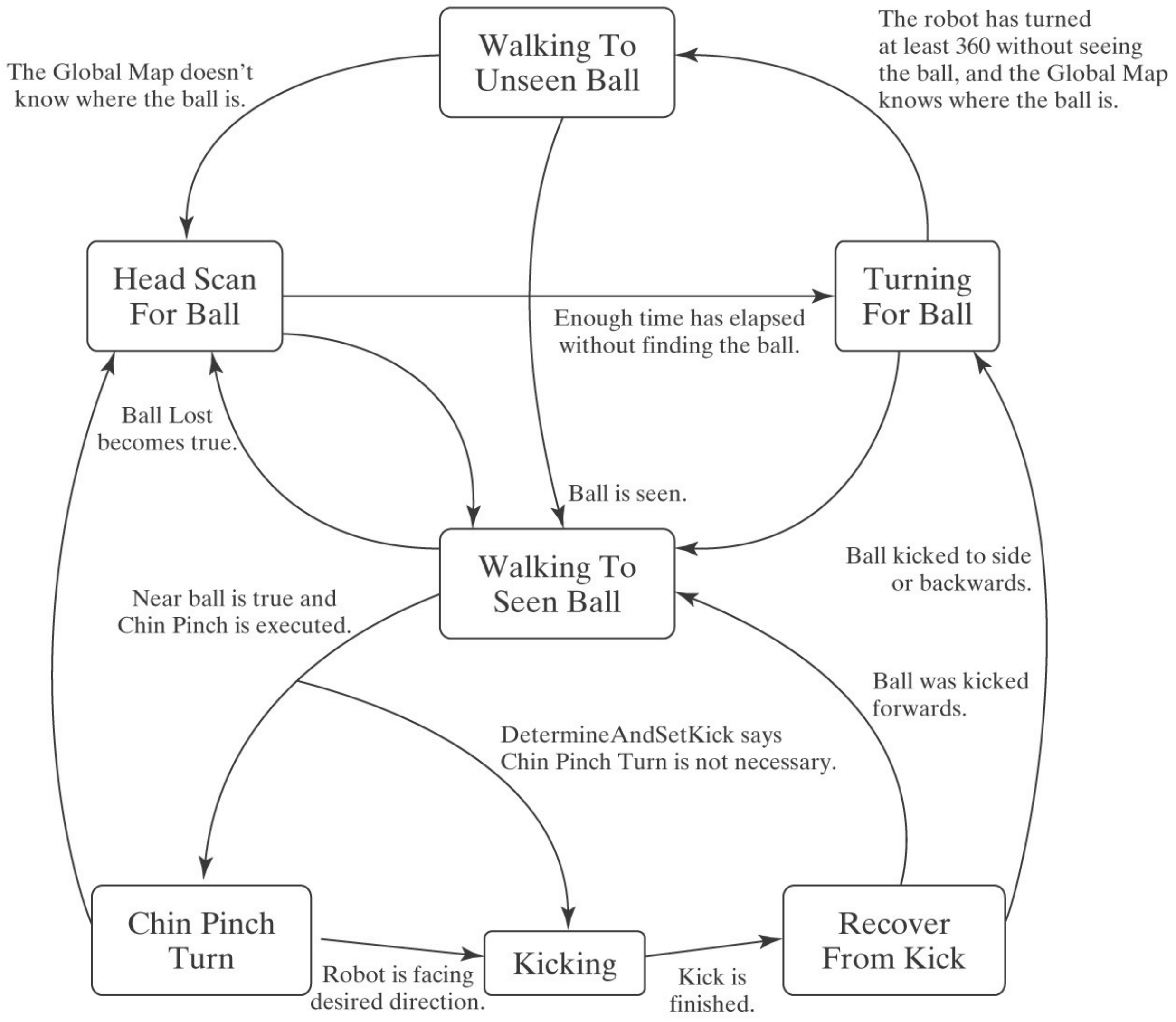


A Simple Communication Protocol



Controlling a Soccer-Playing Robot

A Simple Controller

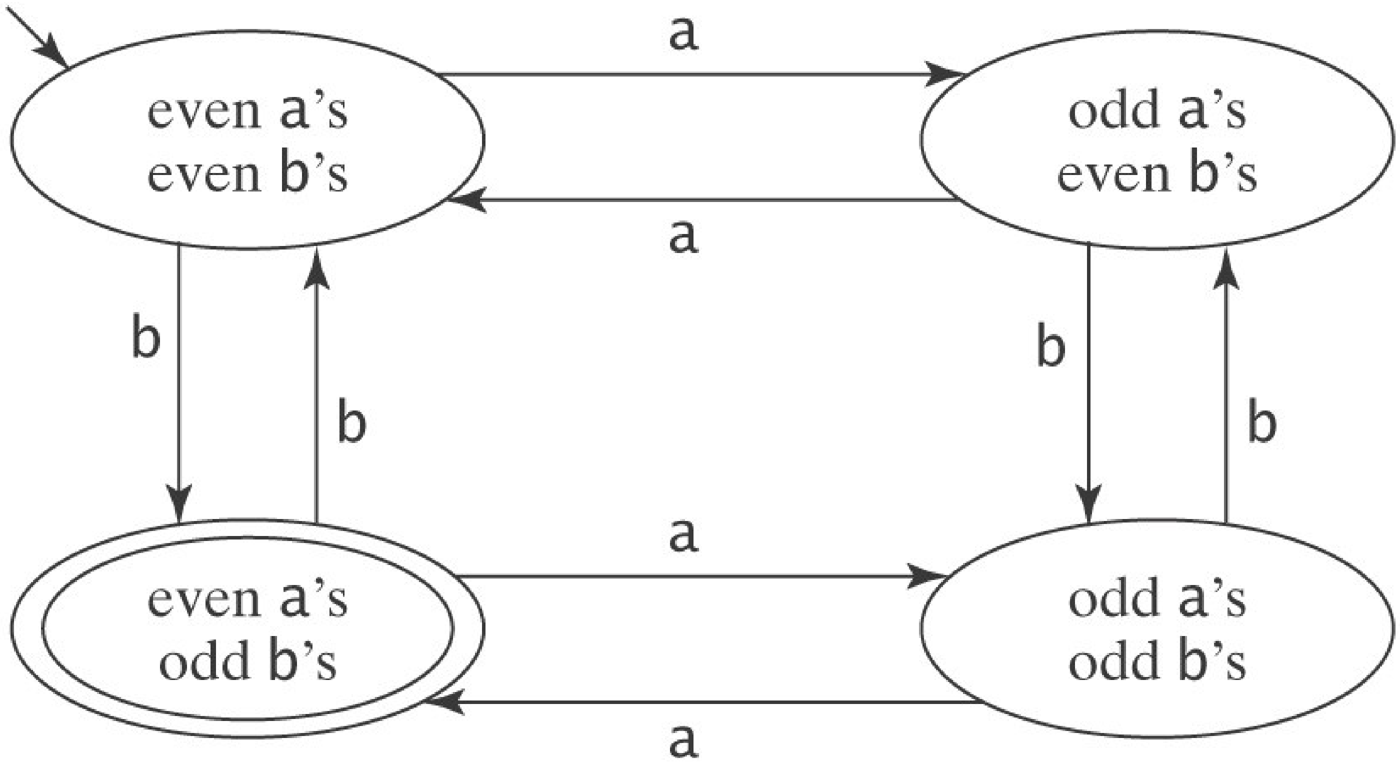

Walking To Unseen Ball
The Global Map doesn't know where the ball is.
The robot has turned at least 360 without seeing the ball, and the Global Map knows where the ball is.
Head Scan For Ball
Turning For Ball
Enough time has elapsed without finding the ball.
Ball Lost becomes true.
Ball is seen.
Walking To Seen Ball
Ball kicked to side or backwards.
Near ball is true and Chin Pinch is executed.
Ball was kicked forwards.
DetermineAndSetKick says Chin Pinch Turn is not necessary.
Chin Pinch Turn
Kicking
Recover From Kick
Robot is facing desired direction.
Kick is finished.

Programming FSMs

Cluster strings that share a "future".

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

Even a's Odd b's

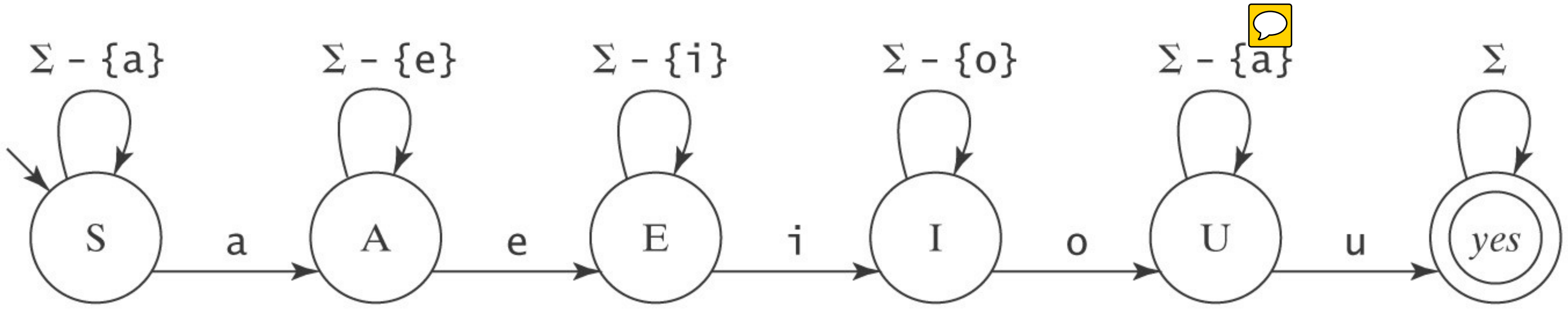


Vowels in Alphabetical Order

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

Vowels in Alphabetical Order

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



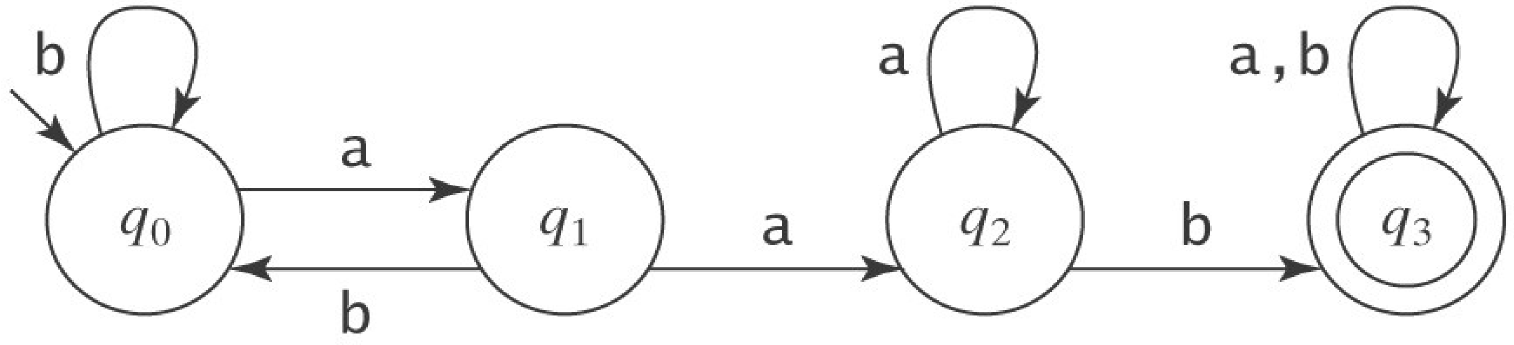
Complementing FSMs

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

Complementing FSMs

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

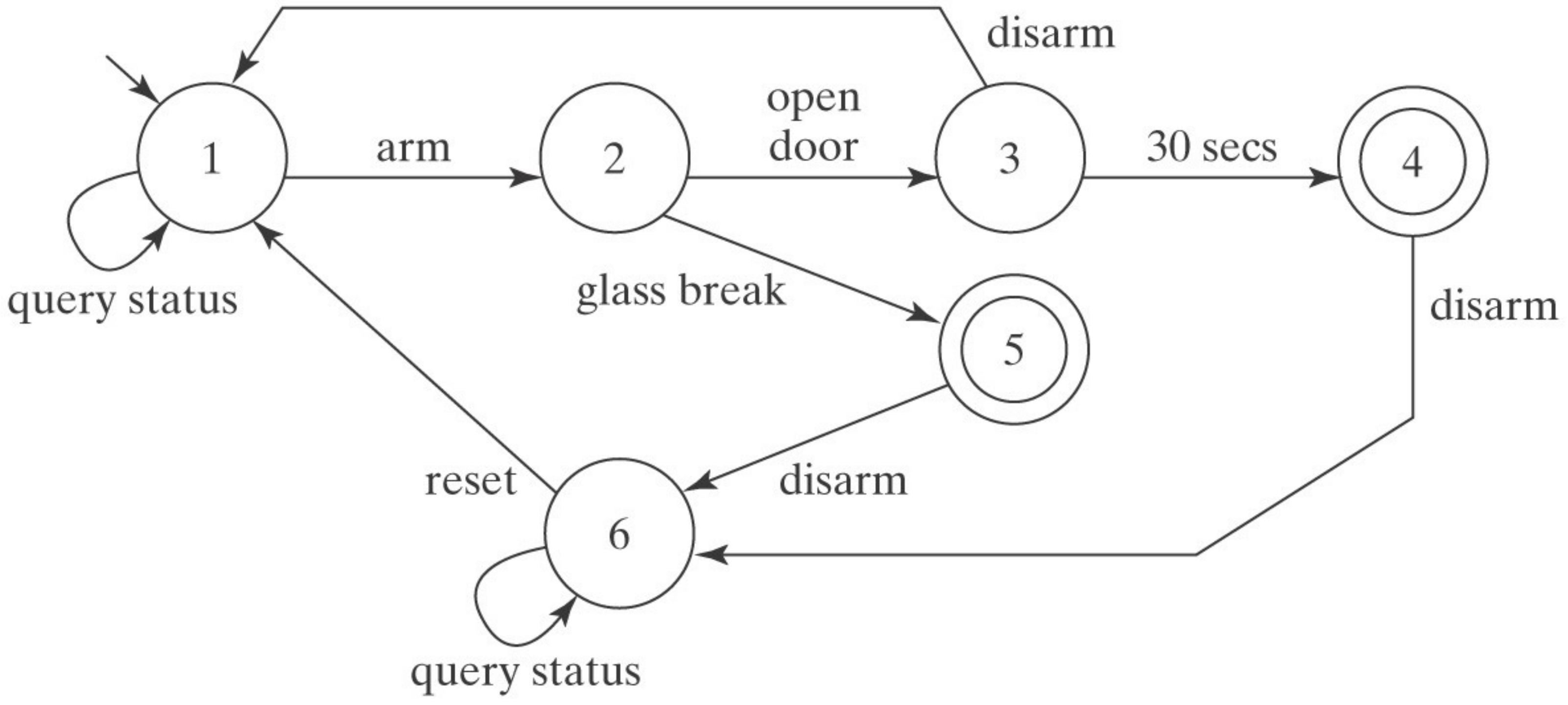
Start with a machine for $\neg L$:



How must it be changed?

A Building Security System

L = {event sequences such that the alarm should sound}



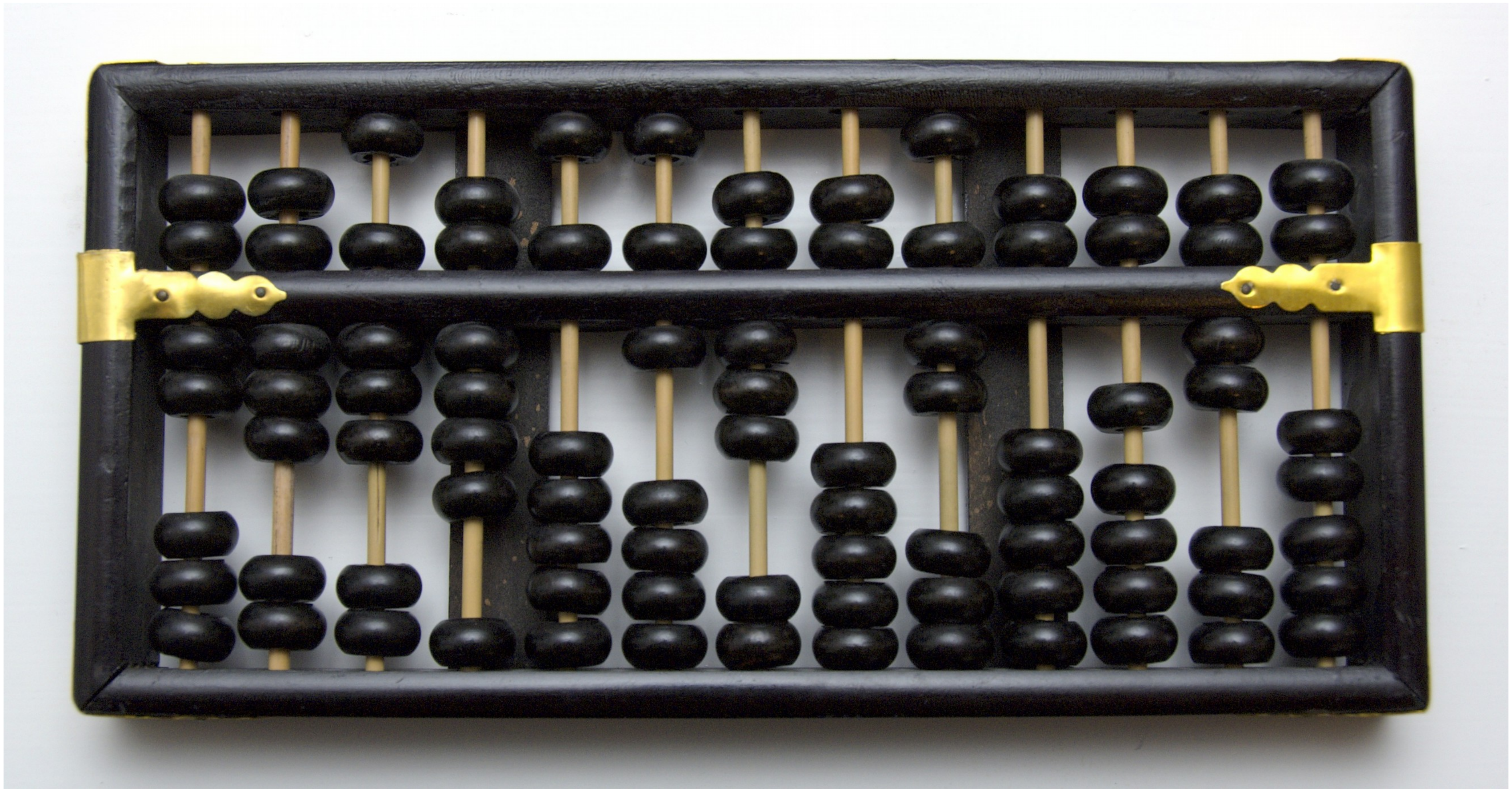
FSMs Predate Computers

The Antikythera mechanism (Greece, 80 BC)

FSMs Predate Computers

The Prague Orloj, originally built in 1410.

FSMs Predate Computers



The abacus

FSMs Predate Computers

The Jacquard Loom (1801)

The Missing Letter Language

Let Σ = {`a`, `b`, `c`, `d`}.

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

Try to make a DFSM for $L_{Missing}$:

Definition of an NDFSM

A **nondeterministic** FSM (NDFSM) is
$M = (K, \Sigma, \Delta, s, A)$, 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

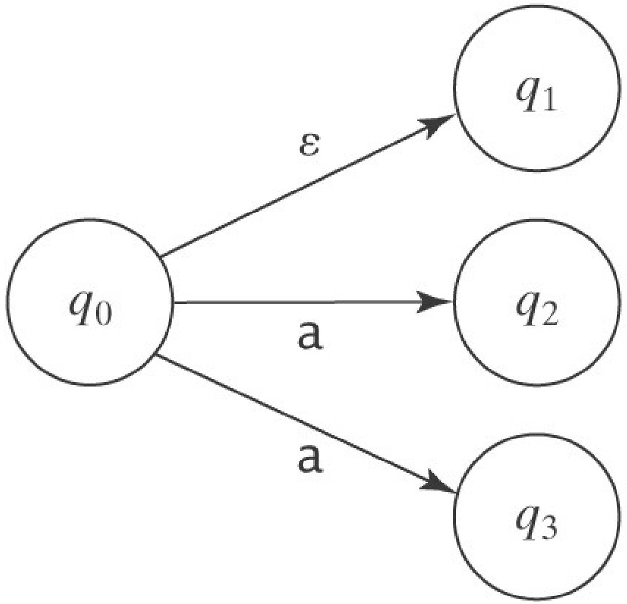
$$(K \times (\Sigma \cup \{\varepsilon\})) \times K$$

Accepting by an NDFSM

M **accepts** a string *w* iff there exists *some path* along which *w* drives *M* to some element of *A*.

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

Sources of Nondeterminism

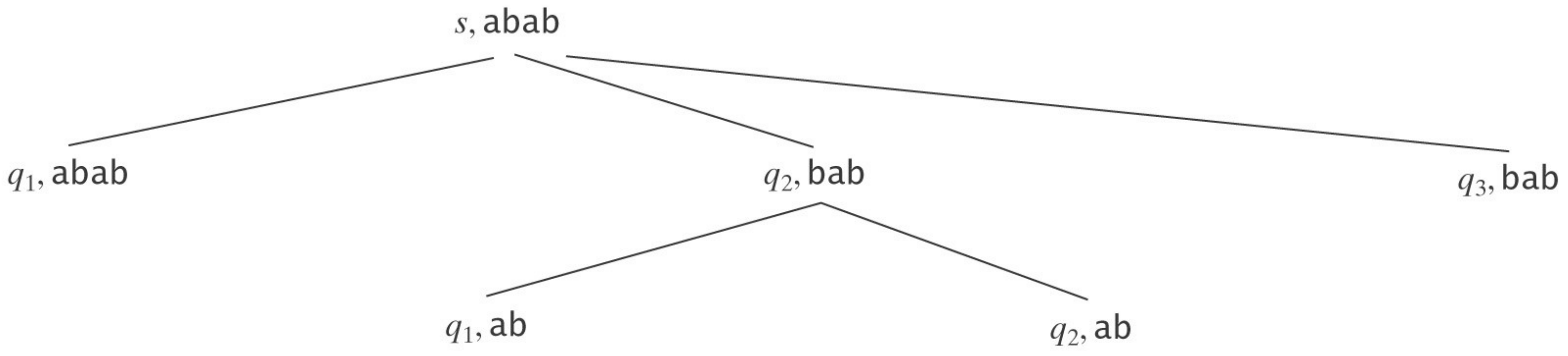


$$\begin{tikzcd}
 & q_1 \\
q_0 \arrow[ur, "\varepsilon"] \arrow[r, "a"'] \arrow[dr, "a"] & q_2 \\
 & q_3
\end{tikzcd}$$

Analyzing Nondeterministic FSMs

Two approaches:

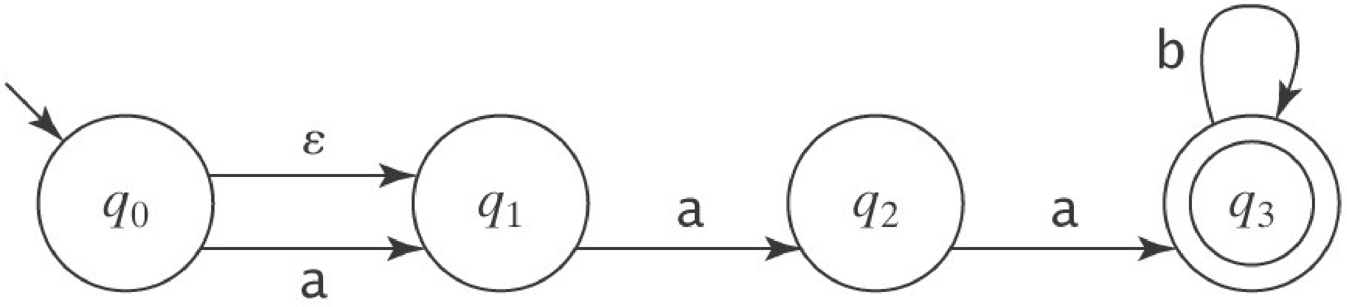
- Explore a search tree:



- Follow all paths in parallel

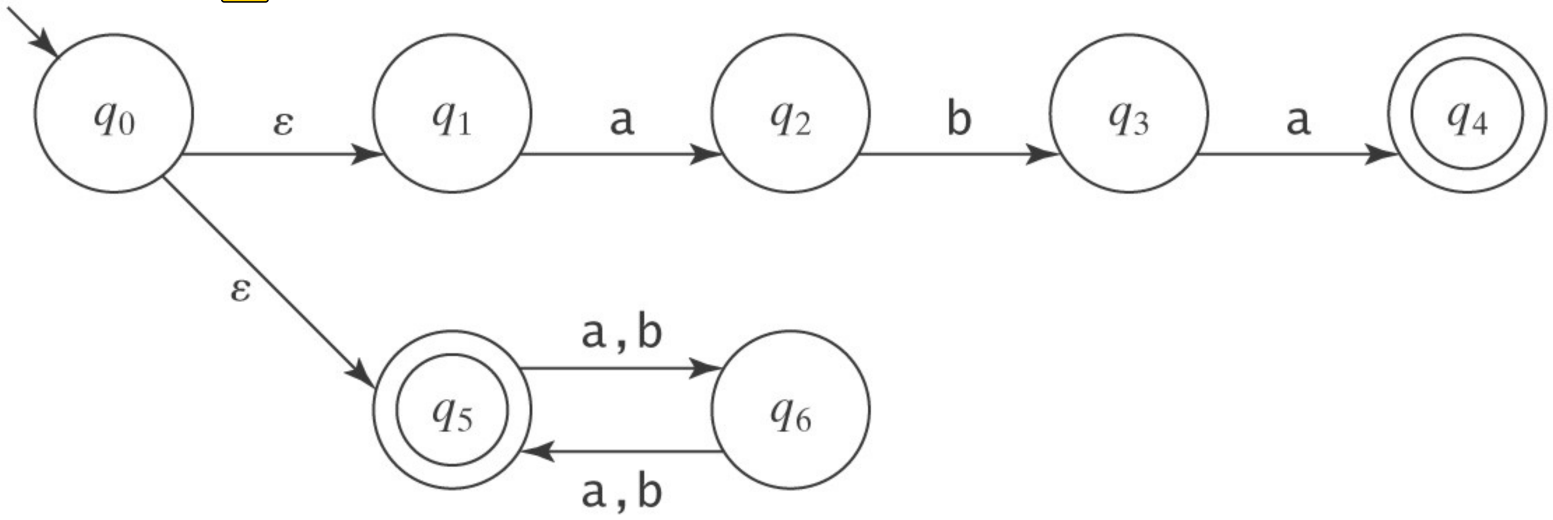
Optional Substrings

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



Multiple Sublanguages

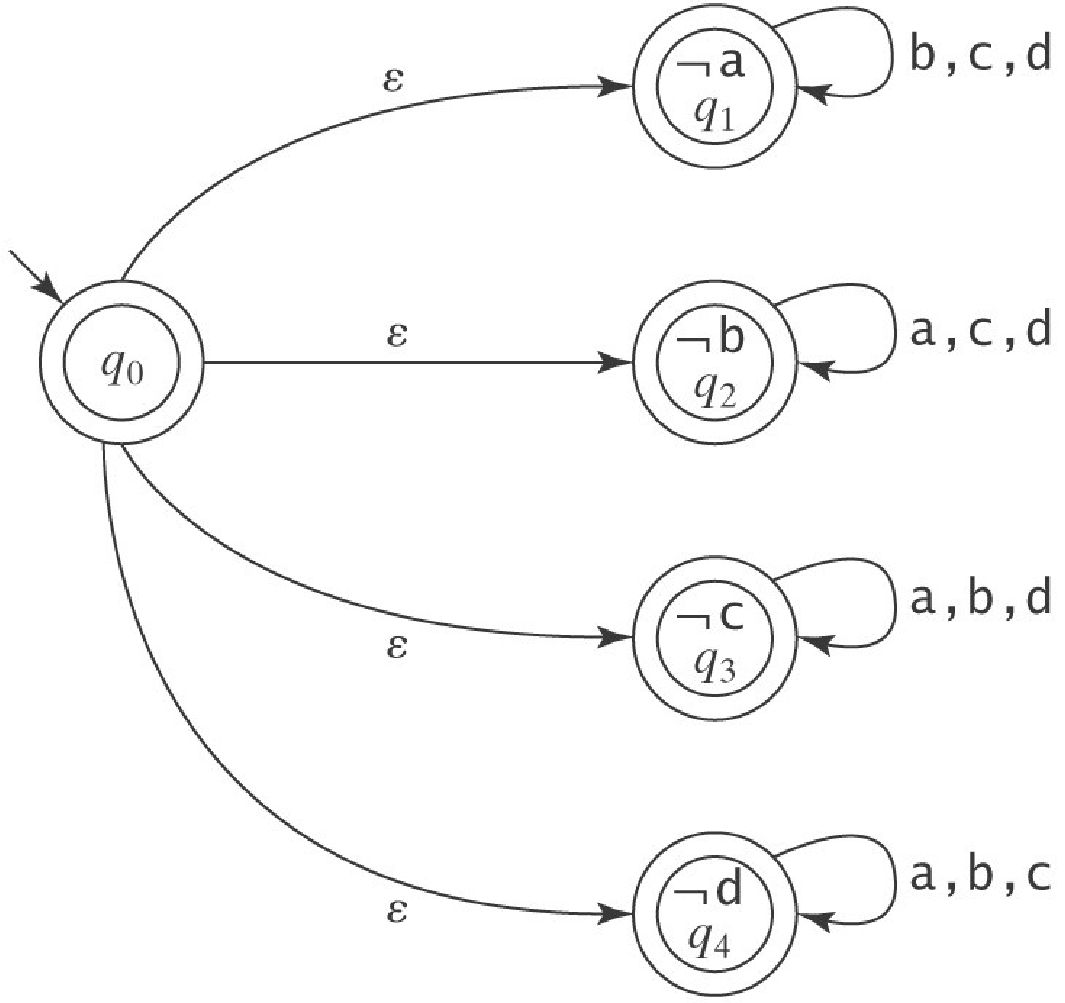
$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* : w = \mathtt{aba} \text{ or } |w| \text{ is even}\}$.



The Missing Letter Language

Let Σ = {a, b, c, d}. Let $L_{Missing}$ = {w : there is a symbol $a_i \in \Sigma$ not appearing in w}

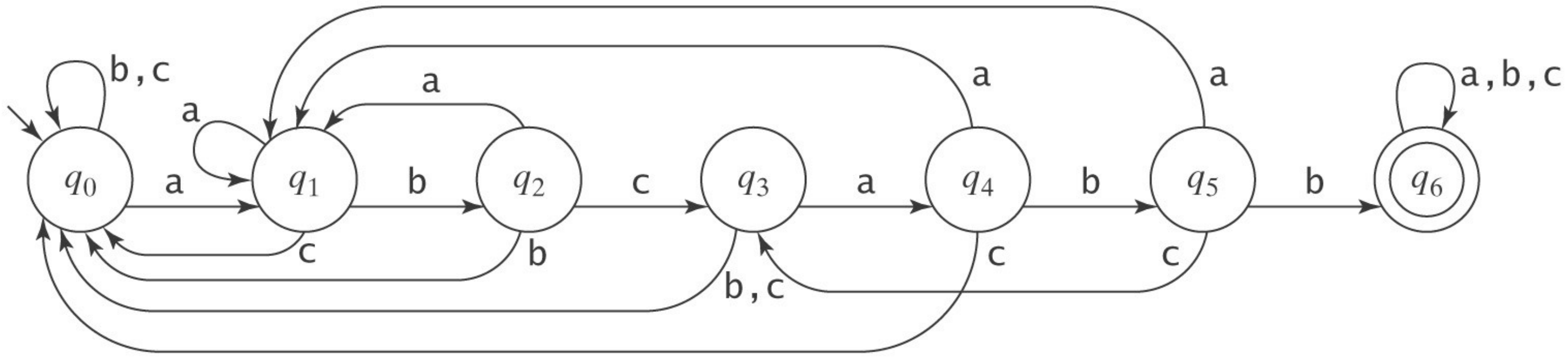
The Missing Letter Language



Pattern Matching

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

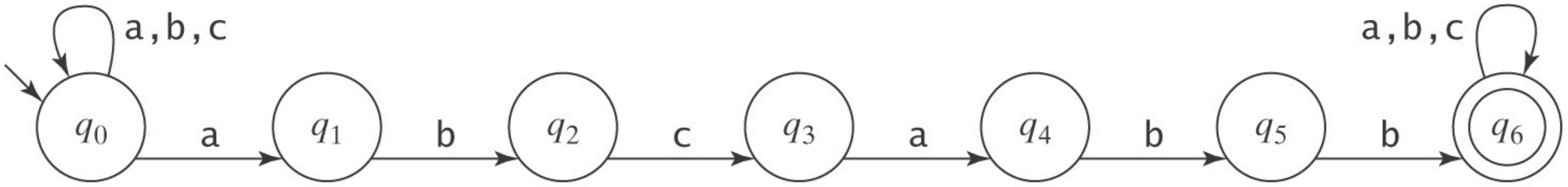
A DFSM:



Pattern Matching with NDFSMs

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

An NDFSM:


$$\begin{tikzcd} \arrow[r] & q_0 \arrow[loop above, "a,b,c"] \arrow[r, "a"] & q_1 \arrow[r, "b"] & q_2 \arrow[r, "c"] & q_3 \arrow[r, "a"] & q_4 \arrow[r, "b"] & q_5 \arrow[r, "b"] & \boxed{q_6} \arrow[loop above, "a,b,c"] \end{tikzcd}$$

Multiple Keywords

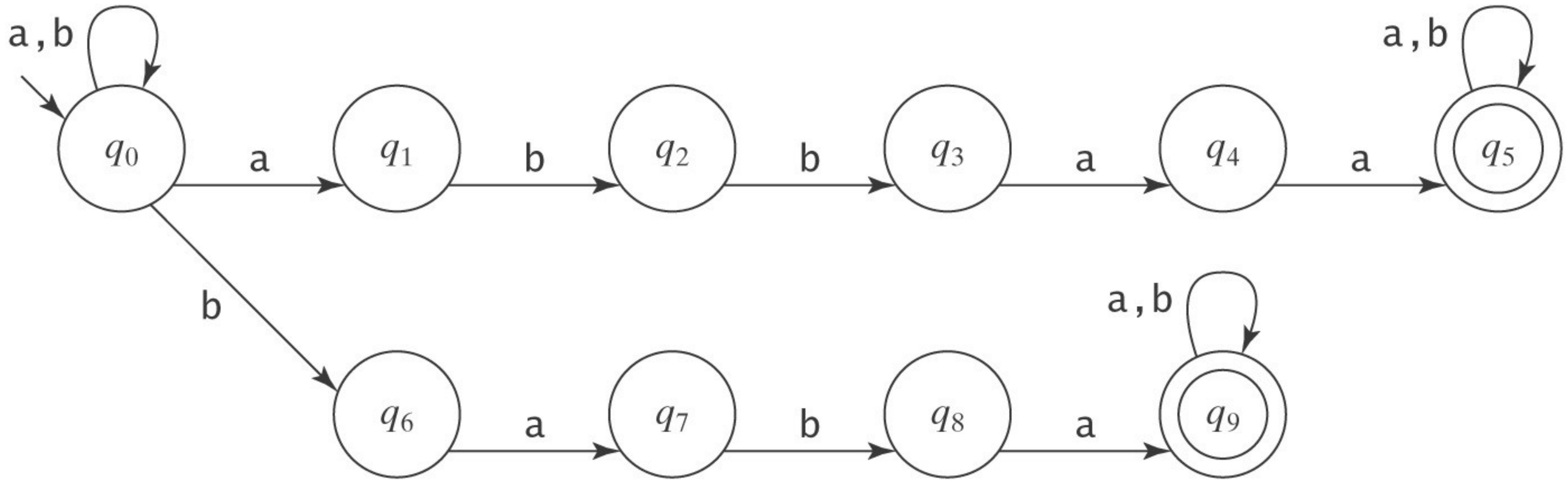
$L = \{w \in \{\texttt{a}, \texttt{b}\}^* :$

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

$w = x\ \texttt{baba}\ y\}$

Multiple Keywords

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



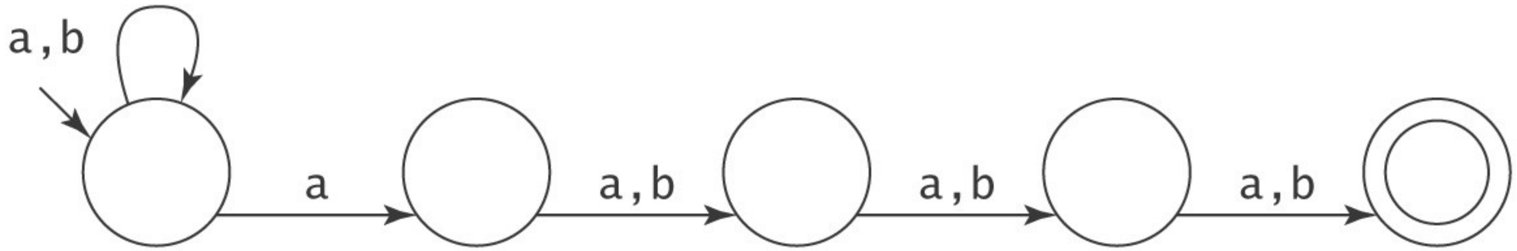
Checking from the End

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

the fourth to the last character is $\mathtt{a}\}$

Checking from the End

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



Another Pattern Matching Example

$L = \{w \in \{\texttt{0}, \texttt{1}\}^* : w$ is the binary encoding of a positive integer that is divisible by 16 or is odd$\}$

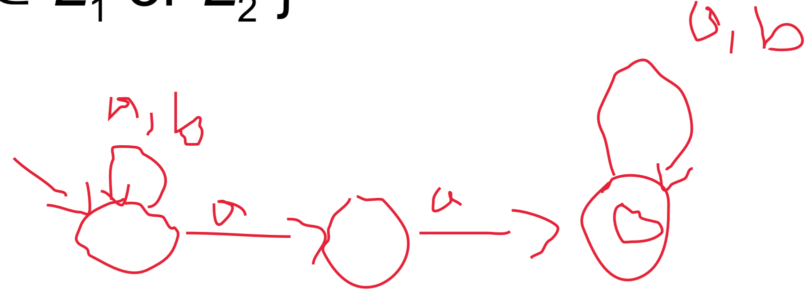
Another NDFSM

$L_1 = \{w \in \{\texttt{a}, \texttt{b}\}^*: \texttt{aa}$ occurs in $w\}$

$L_2 = \{x \in \{\texttt{a}, \texttt{b}\}^*: \texttt{bb}$ occurs in $x\}$

$L_3 = \{y : \in L_1$ or $L_2\}$

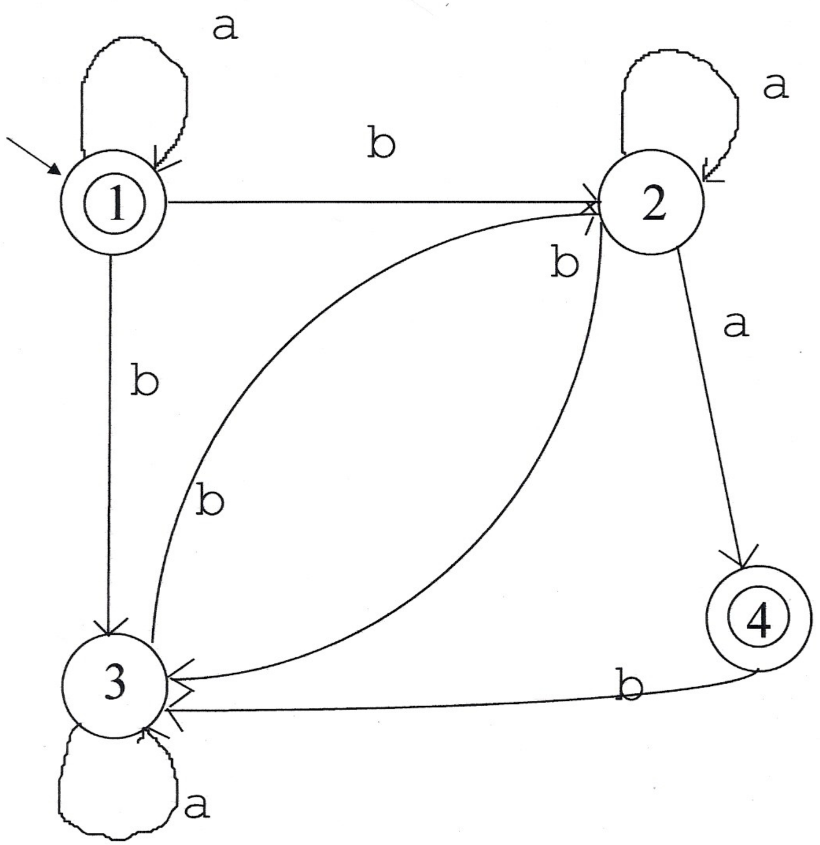
$M_1 =$



$M_2 =$

$M_3 =$

Analyzing Nondeterministic FSMs



Does this FSM accept:

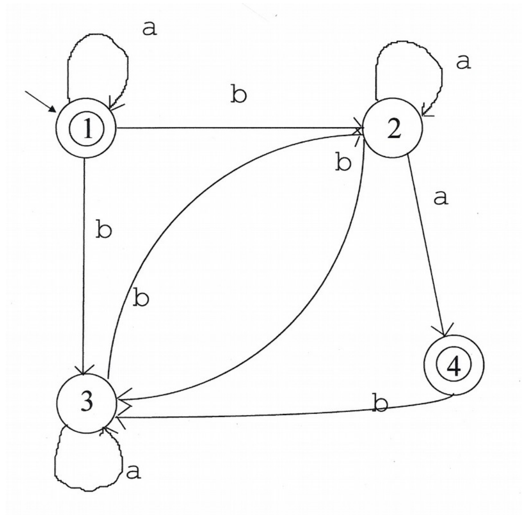
```
baaba
```

Remember: we just have to find one accepting path.

Analyzing Nondeterministic FSMs

Two approaches:

- Explore a search tree



$$\begin{tikzcd}
 & & & & 1,\text{baaba} \arrow[dll, dash] \arrow[drr, dash] & & & \\
 & & 2,\text{aaba} \arrow[dl, dash] \arrow[dr, dash] & & & & 3,\text{aaba} \arrow[d, dash] & \\
 & 2,\text{aba} \arrow[dl, dash] \arrow[dr, dash] & & 4,\text{aba} & & & 3,\text{aba} \arrow[d, dash] & \\
2,\text{ba} \arrow[d, dash] & & 4,\text{ba} \arrow[d, dash] & & & & 3,\text{ba} \arrow[d, dash] & \\
3,\text{a} \arrow[d, dash] & & 3,\text{a} \arrow[d, dash] & & & & 2,\text{a} \arrow[dl, dash] \arrow[dr, dash] & \\
3,\varepsilon & & 3,\varepsilon & & & 2,\varepsilon & & 4,\varepsilon
\end{tikzcd}$$

- Follow all paths in parallel

$$\begin{tikzcd}
\{1\} \arrow[r, "b"] & \{2,3\} \arrow[r, "a"] & \{2,3,4\} \arrow[r, "a"] & \{2,3,4\} \arrow[r, "b"] & \{2,3\} \arrow[r, "a"] & \{2,3,4\}
\end{tikzcd}$$

Dealing with ε Transitions

ε-transitions change state without using input symbols

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

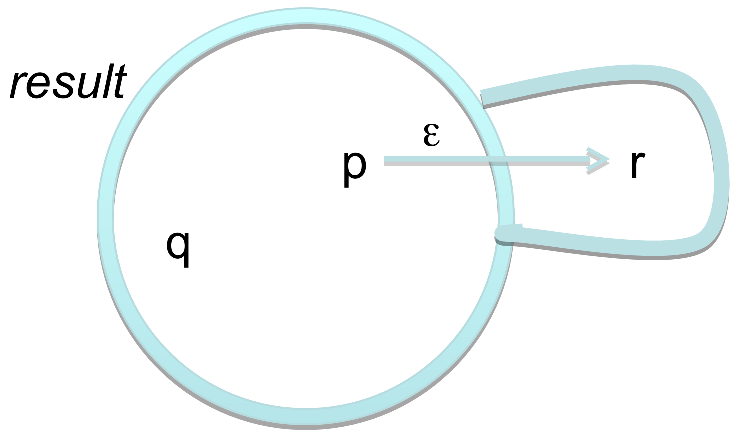
eps(*q*) is the closure of $\{q\}$ under the relation $\{(p, r) :$ there is a transition $(p, \varepsilon, r) \in \Delta\}$.

How shall we compute *eps*(*q*)?

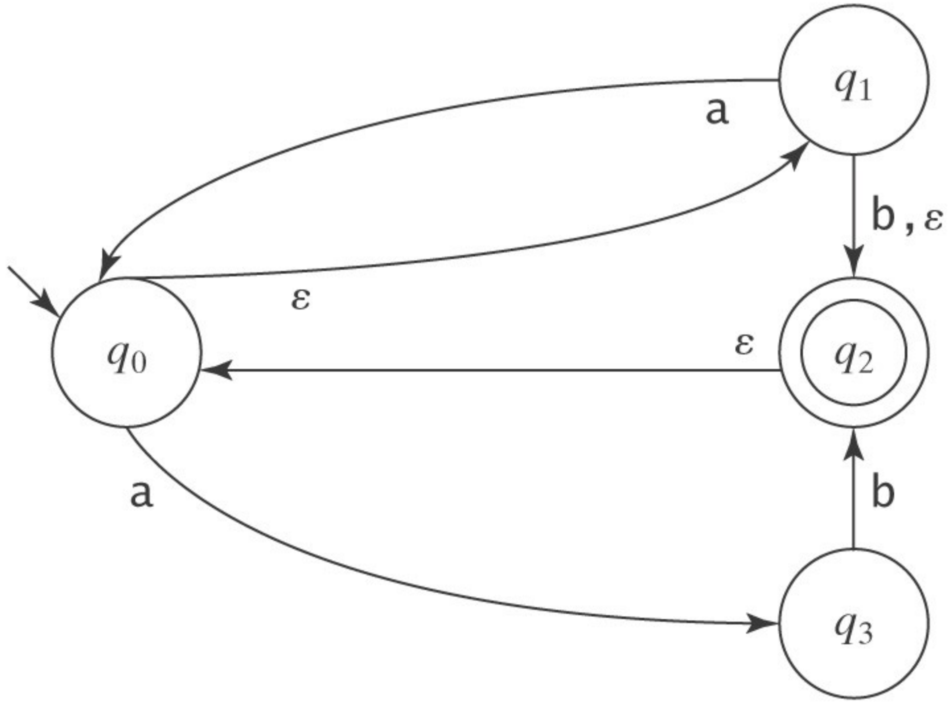
An Algorithm to Compute *eps*(*q*)

```
Compute 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.
```



An Example of *eps*

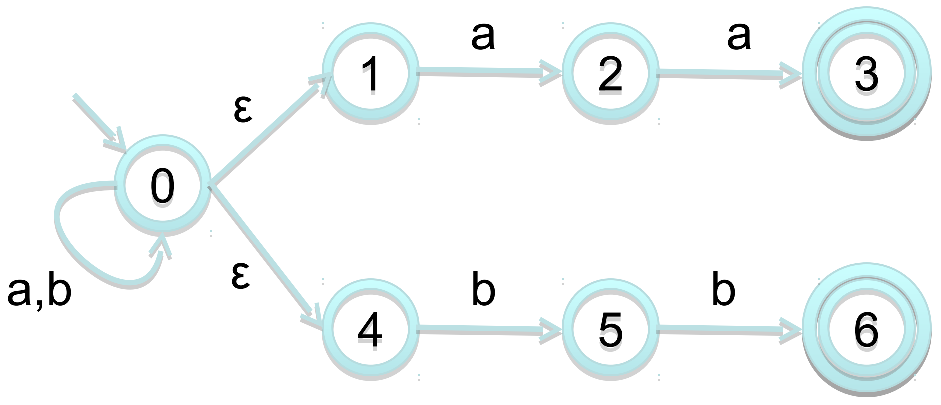


$eps(q_0) = \{q_0, q_1, q_2\}$
$eps(q_1) = \{q_0, q_1, q_2\}$
$eps(q_2) = \{q_0, q_1, q_2\}$
$eps(q_3) = \{q_3\}$

Simulating a NDFSM

```
ndfsmsimulate(M: NDFSM, w: string) =
1. current-state = eps(s).
2. For each c symbol of w do:
     1. next-state = $\varnothing$.
     2. 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).
     3. current-state = next-state.
3. If current-state contains states in A, then accept.
                                   else reject.
```

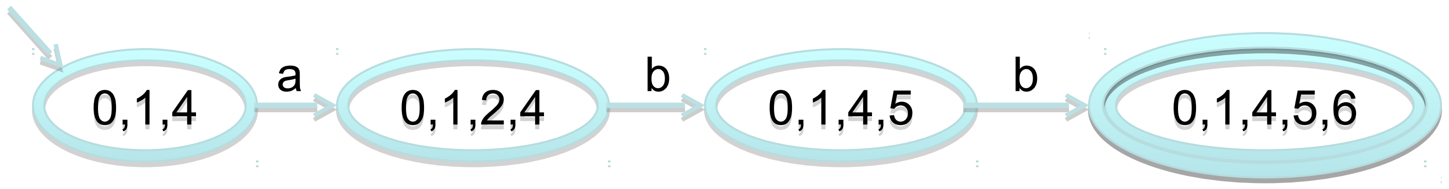
Example



w = abb

eps(0) = {0,1,4}

for 1 ≤ i ≤ 6,
eps(i) = { i }



Nondeterministic and Deterministic FSMs

Clearly: {Languages accepted by a DFSM} ⊆ {Languages accepted by a NDFSM}

More interestingly:

Theorem 5.3:

For each NDFSM, there is an equivalent DFSM.

Nondeterministic and Deterministic FSMs

***Proof**:* By construction:

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

$K' = \mathrm{P}(K)$
$s' = eps(s)$
$A' = \{Q \subseteq K : Q \cap A \neq \varnothing\}$

$\delta'(Q, a) = \bigcup\{eps(p): p \in K$ 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 δ'.

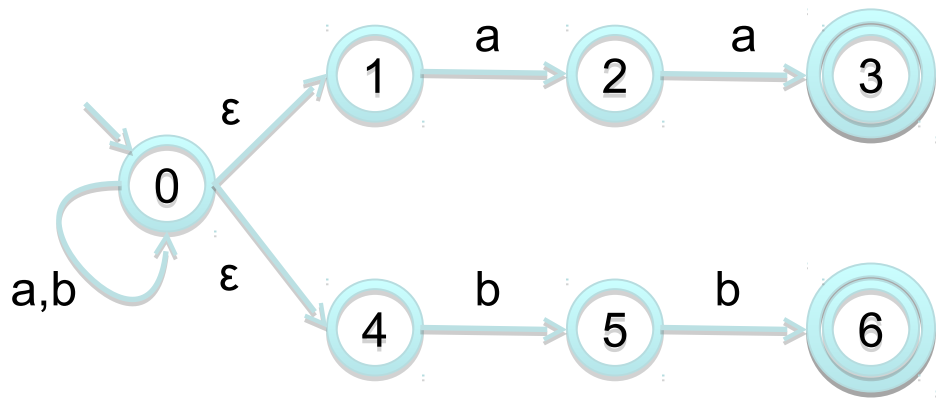
4. Compute K' = a subset of $\mathcal{P}(K)$.

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

The Algorithm *ndfsmtodfsm*

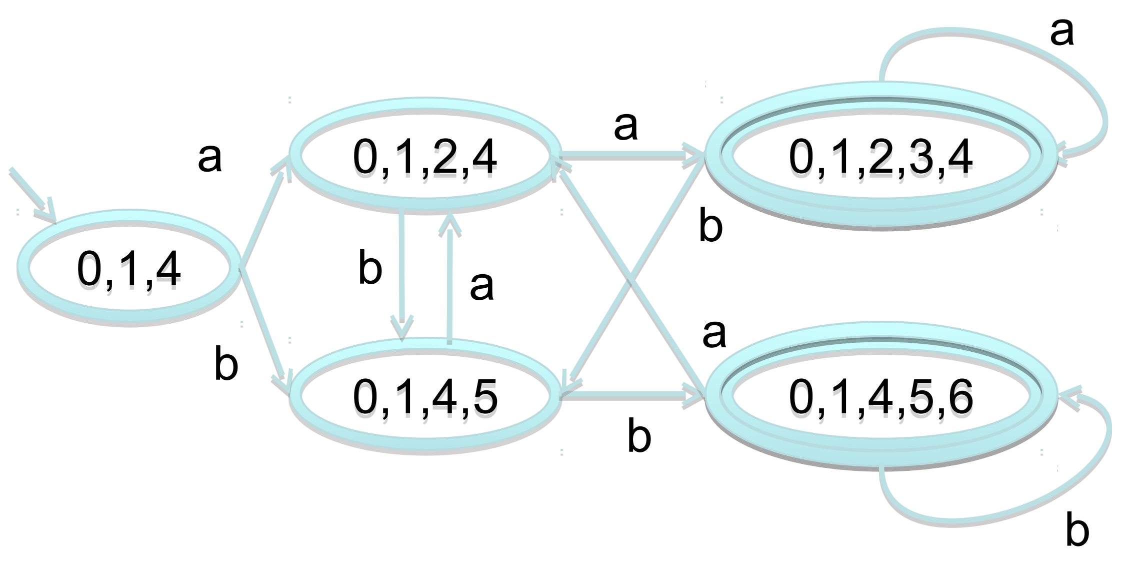
```
ndfsmtodfsm(M: NDFSM) =
    1. For each state q in K do:
            1.1 Compute eps(q).
    2. s' = eps(s)
    3. Compute δ':
            3.1 active-states = {s'}.
            3.2 δ' = ∅.
            3.3 While Q ∈ active-states, c ∈ Σ with δ'(Q,c) unknown do:
               new-state = ∅.
               For each state q in Q do:
                    For each state p such that (q, c, p) ∈ Δ do:
                     new-state = new-state ∪ eps(p).
               δ'(Q,c) = new-state
               active-states = active-states ∪ { new-state }
    4. K' = active-states.
    5. A' = {Q ∈ K' : Q ∩ A ≠ ∅ }.
```

Example



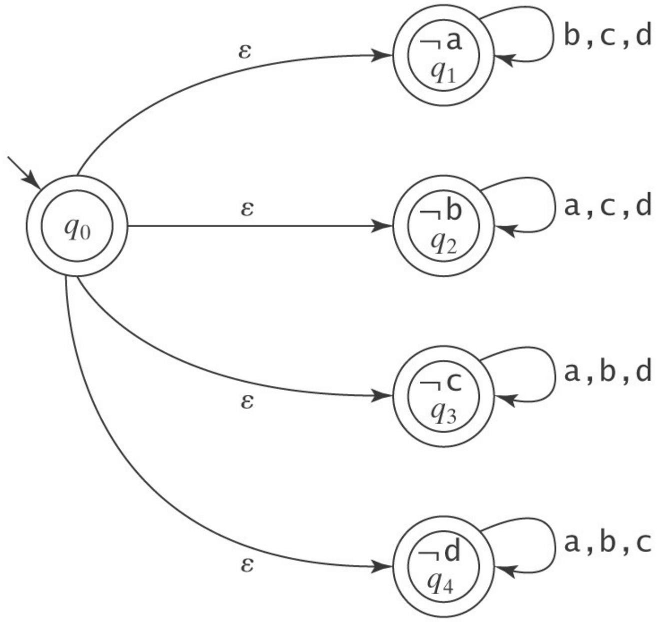
eps(0) = {0,1,4}

for 1 ≤ i ≤ 6,
eps(i) = { i }



The Number of States May Grow Exponentially

$|\Sigma| = n$



No. of states after 0 chars: $= 1$

No. of new states after 1 char: $\binom{n}{n-1} = n$

No. of new states after 2 chars: $\binom{n}{n-2} = n(n-1)/2$

No. of new states after 3 chars: $\binom{n}{n-3} = n(n-1)(n-2)/6$

Total number of states after *n-1* chars: $2^n - 1$

Another Hard Example

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