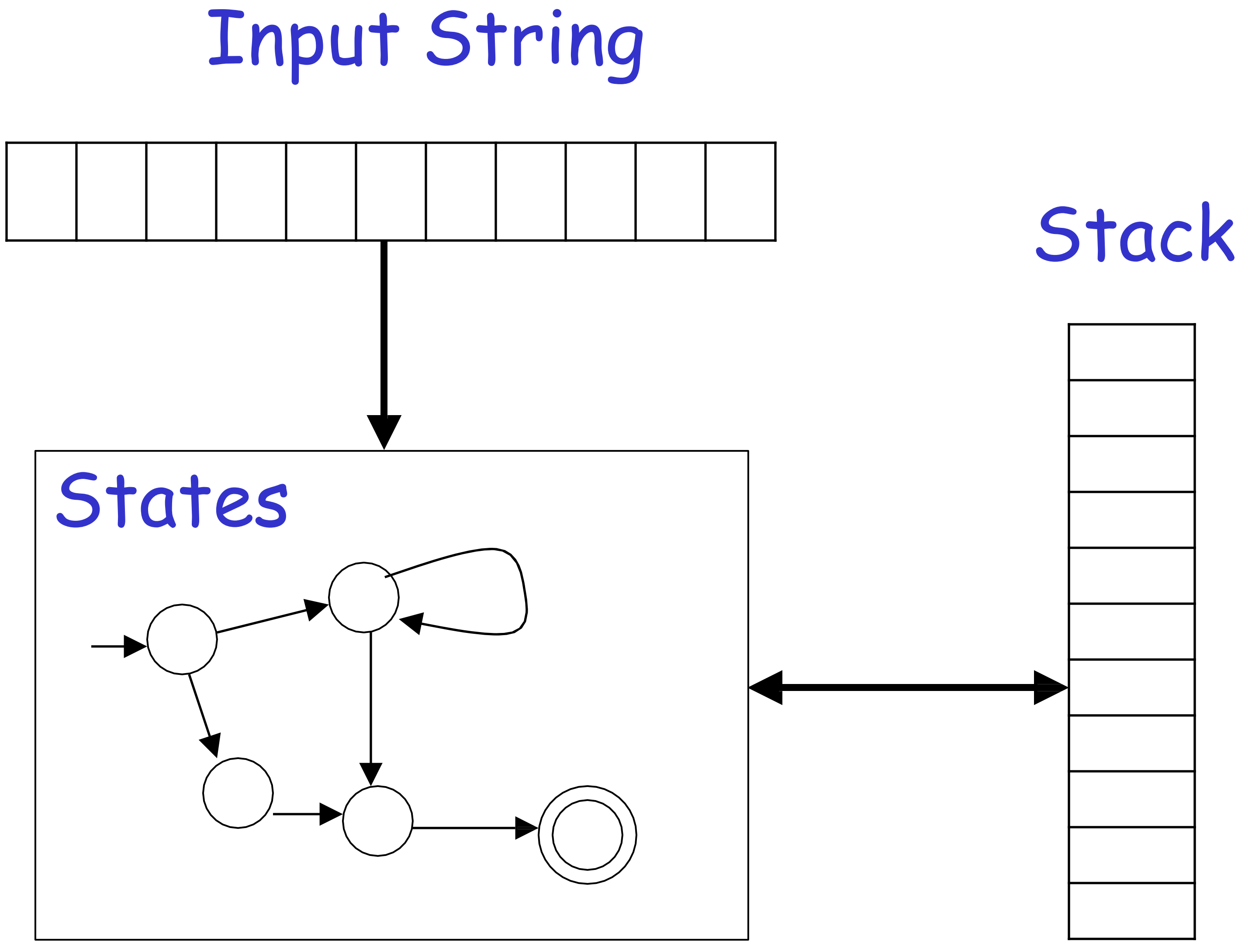
Unit IV

PUSHDOWN AUTOMATA AND POST MACHINES

Outline

- **Push Down Automata:**
  - Introduction and Definition of PDA,
  - Construction (Pictorial/ Transition diagram) of PDA,
  - Instantaneous Description and ACCEPTANCE of CFL by empty stack and final state,
  - Deterministic PDA Vs Nondeterministic PDA,
  - Closure properties of CFLs,
  - pumping lemma for CFL.
- **Post Machine:**
  - Definition and construction

FA +Stack =Pushdown Automaton -- PDA



What is?

- ✓ FA to Regular Language, PDA is to CFL

Formal Definition

Pushdown Automaton (PDA)

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

Accept states

States

Input alphabet

Stack alphabet

Transition function

Initial state

Stack start symbol

Transition Function: δ

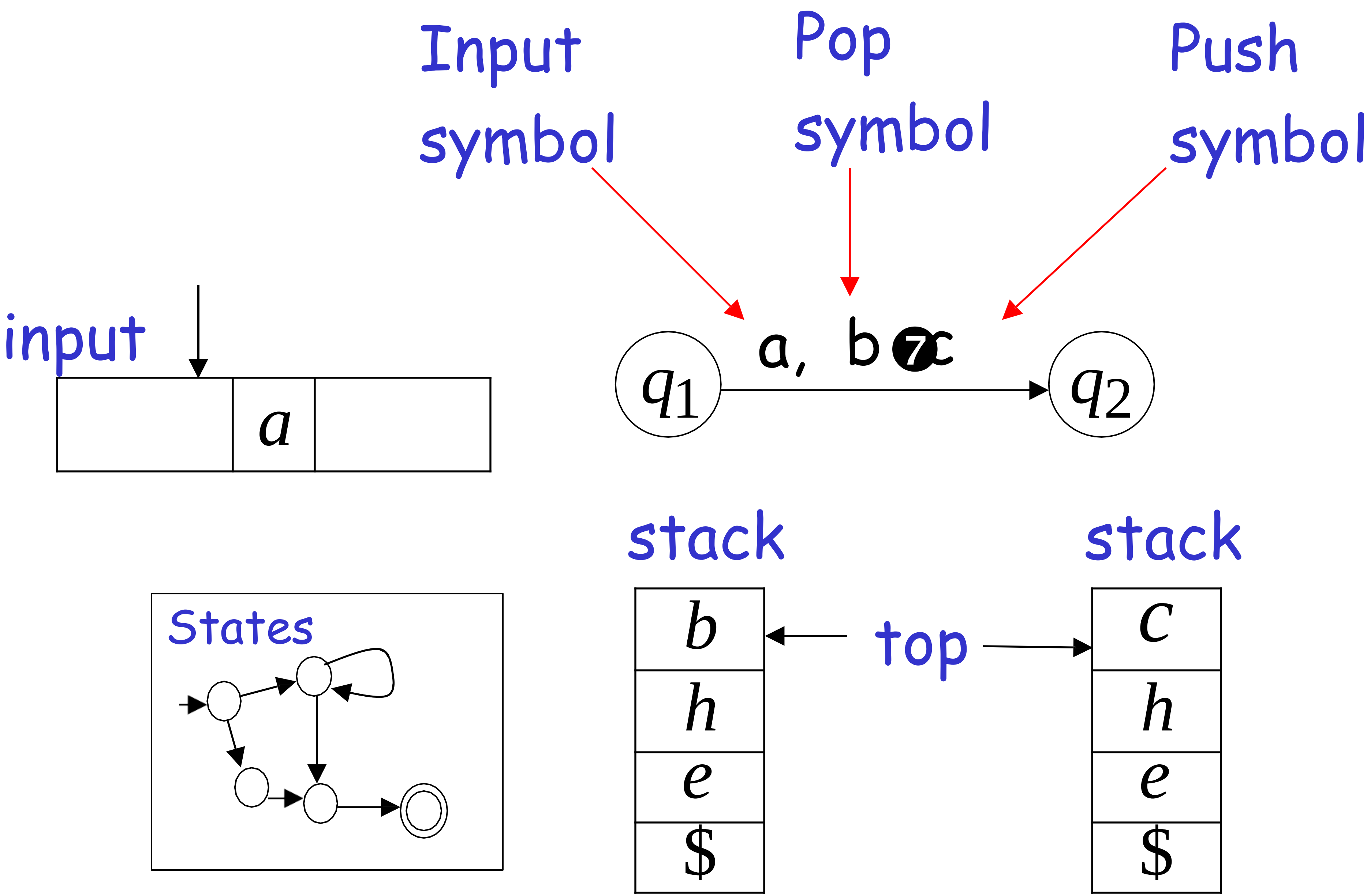
Deterministic PDA:

$$\delta(Q \times \{\Sigma \cup \varepsilon\} \times \Gamma) = Q \times \Gamma^*$$

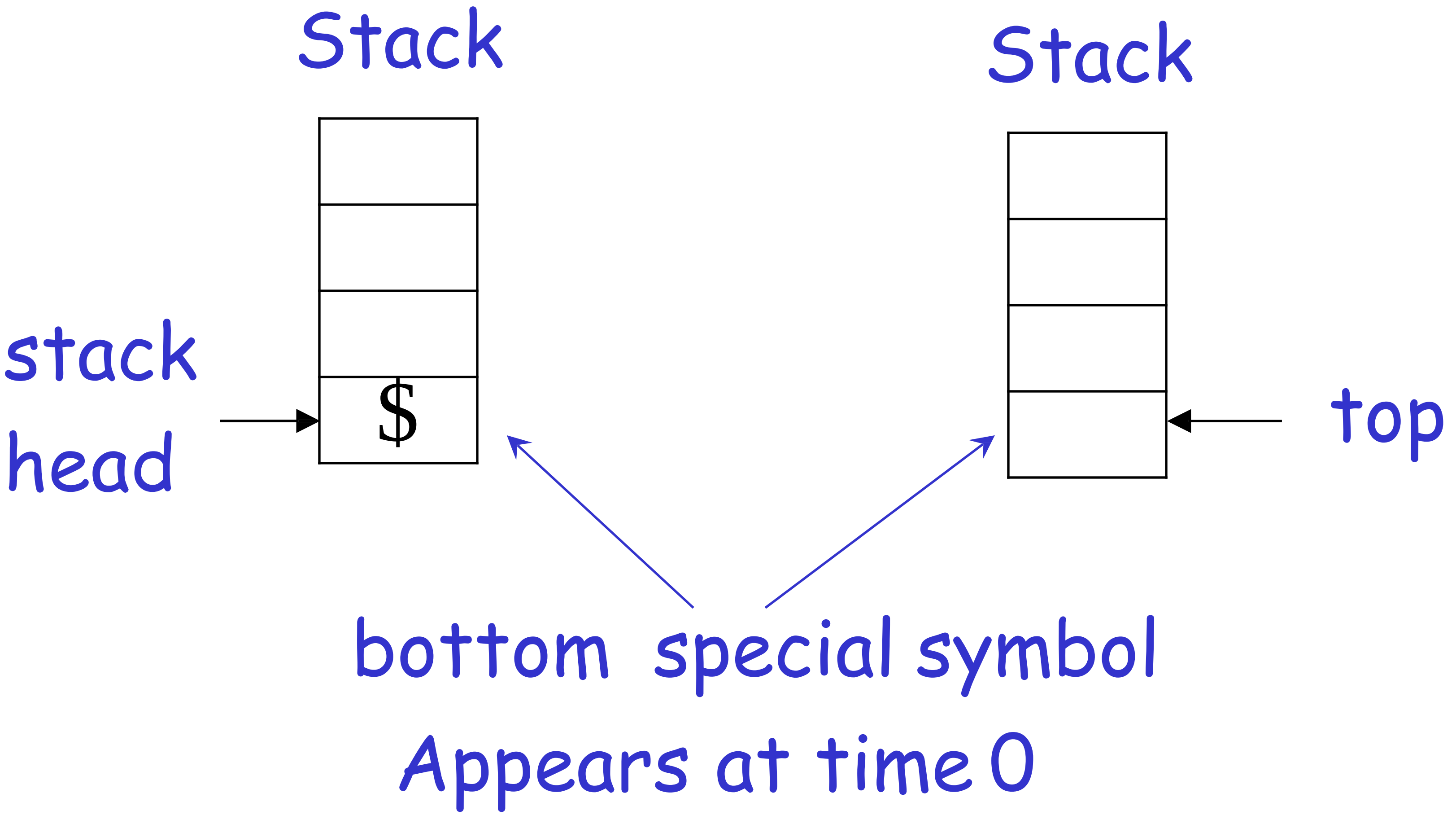
Non-Deterministic PDA:

$$\delta(Q \times \{\Sigma \cup \varepsilon\} \times \Gamma) = 2^{Q \times \Gamma^*}$$

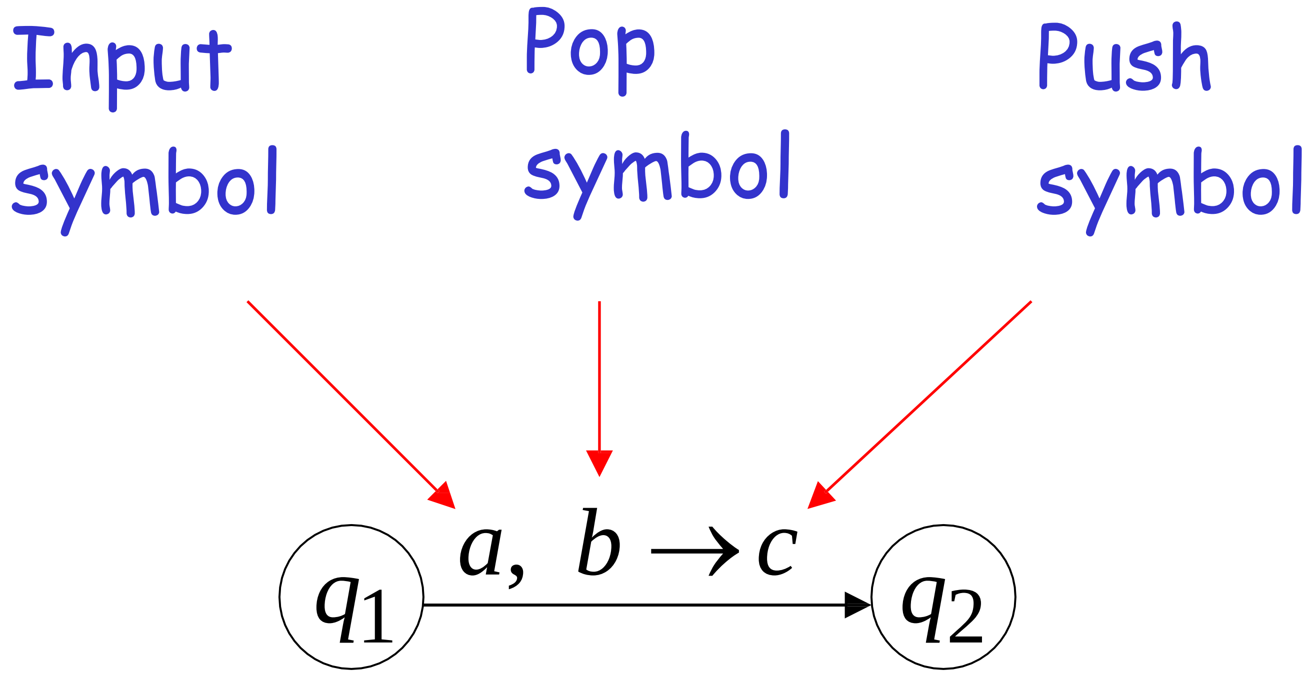
$$\delta(Q \times \{\Sigma \cup \varepsilon\} \times \Gamma) = Q \times \Gamma^*$$

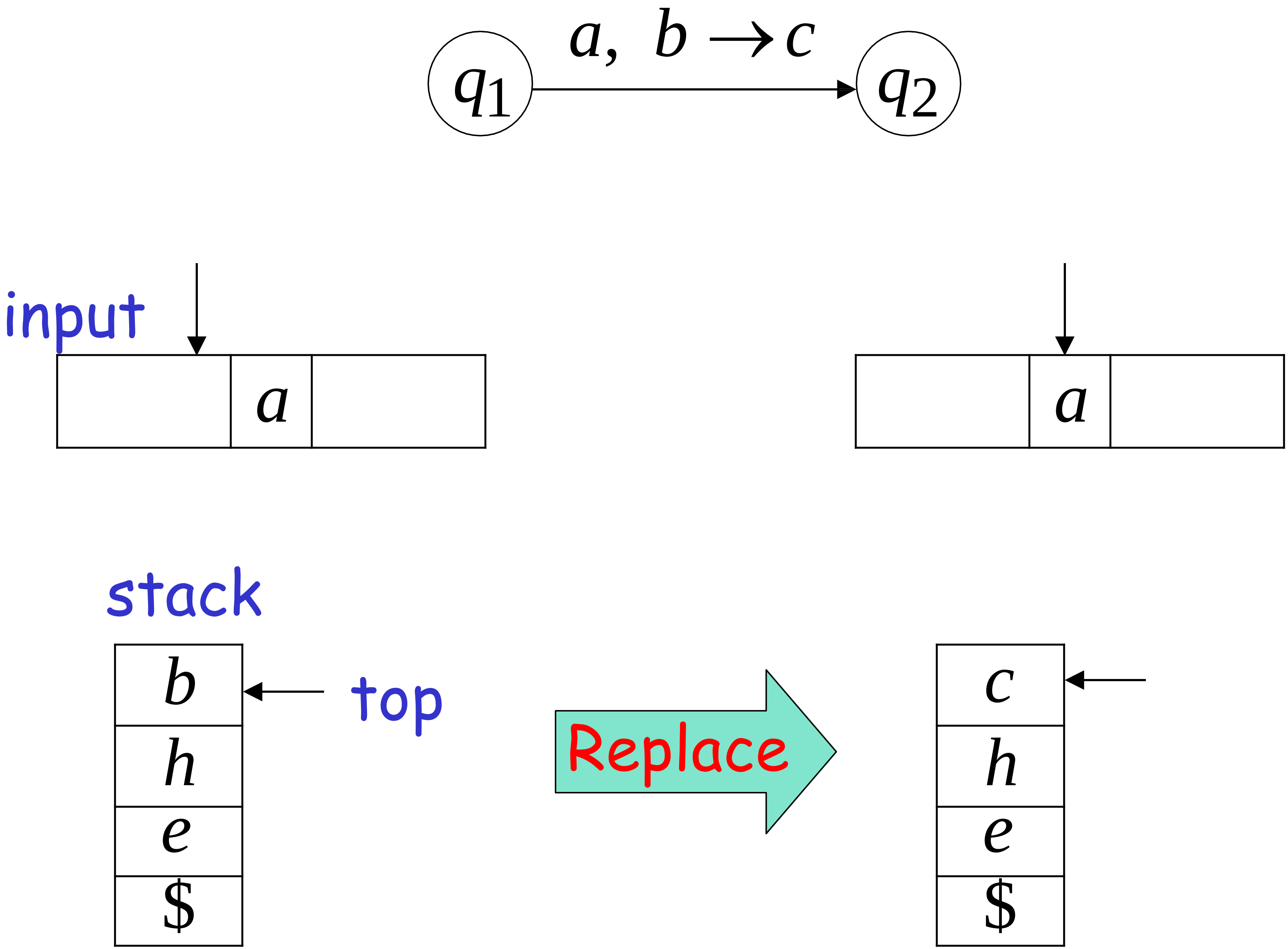


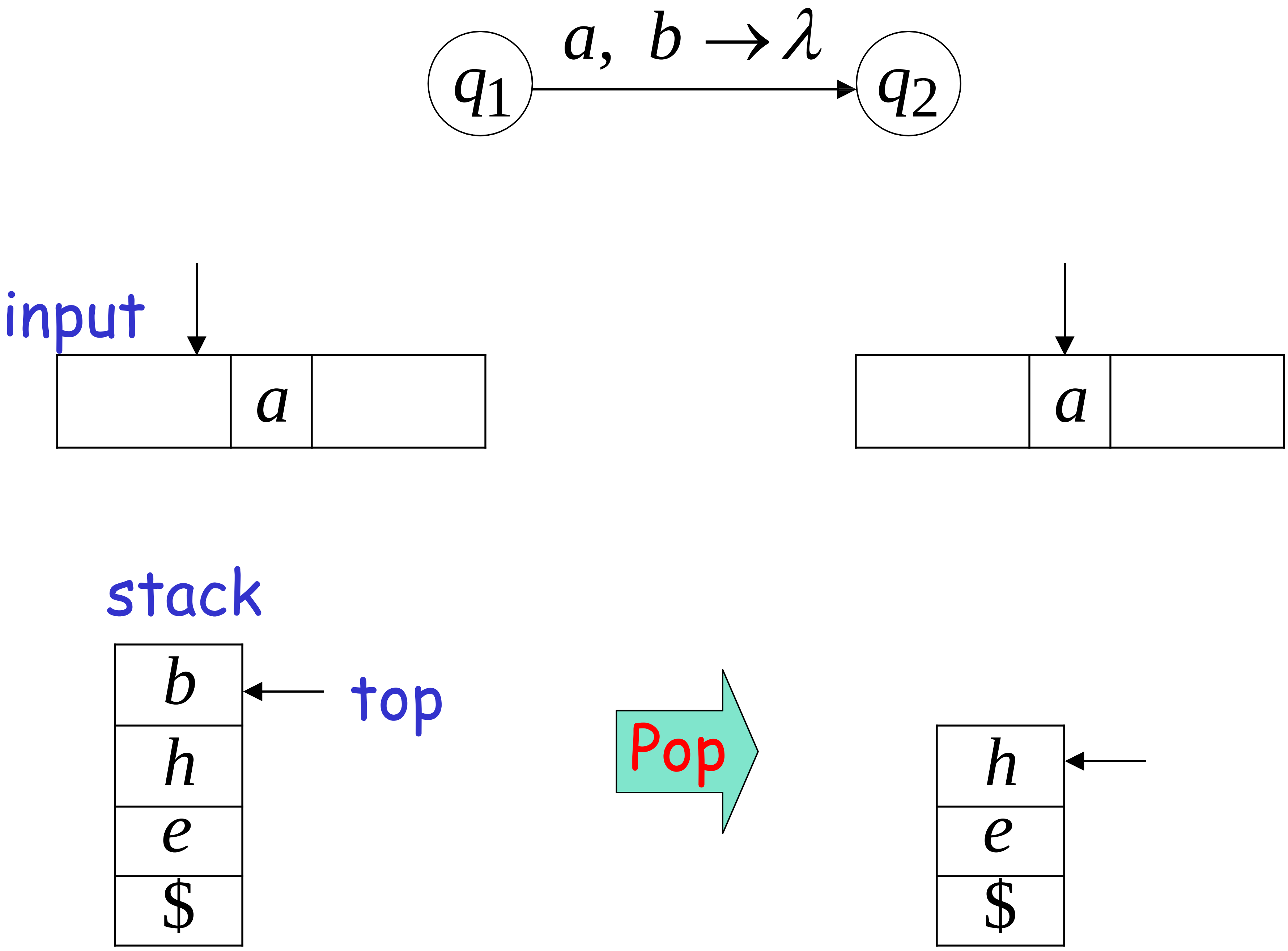
Initial Stack Symbol



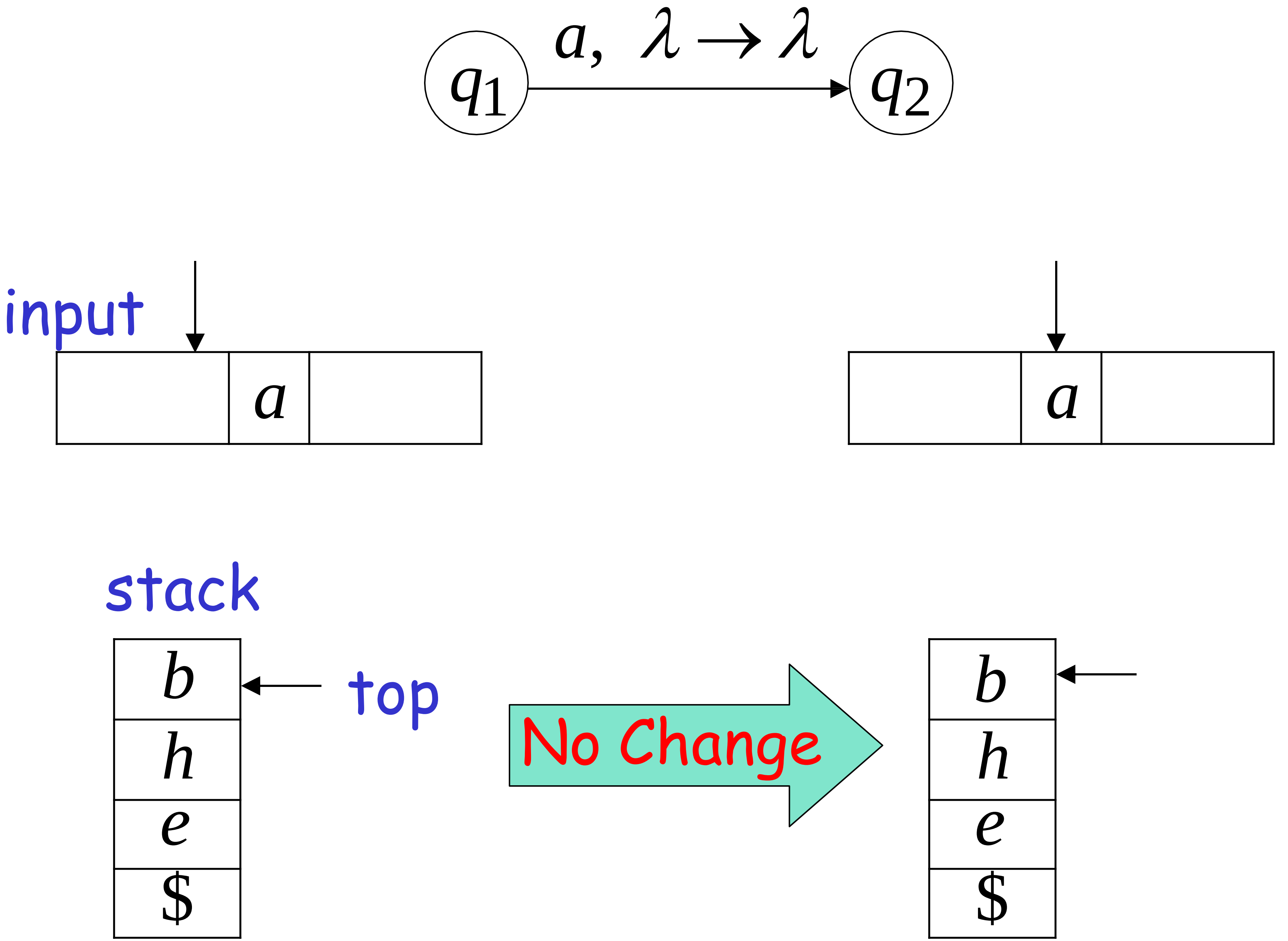
The States




$a, \ b \to c$
q_1
q_2
input
a
a
stack
b
h
e
$\$$
top
Replace
c
h
e
$\$$

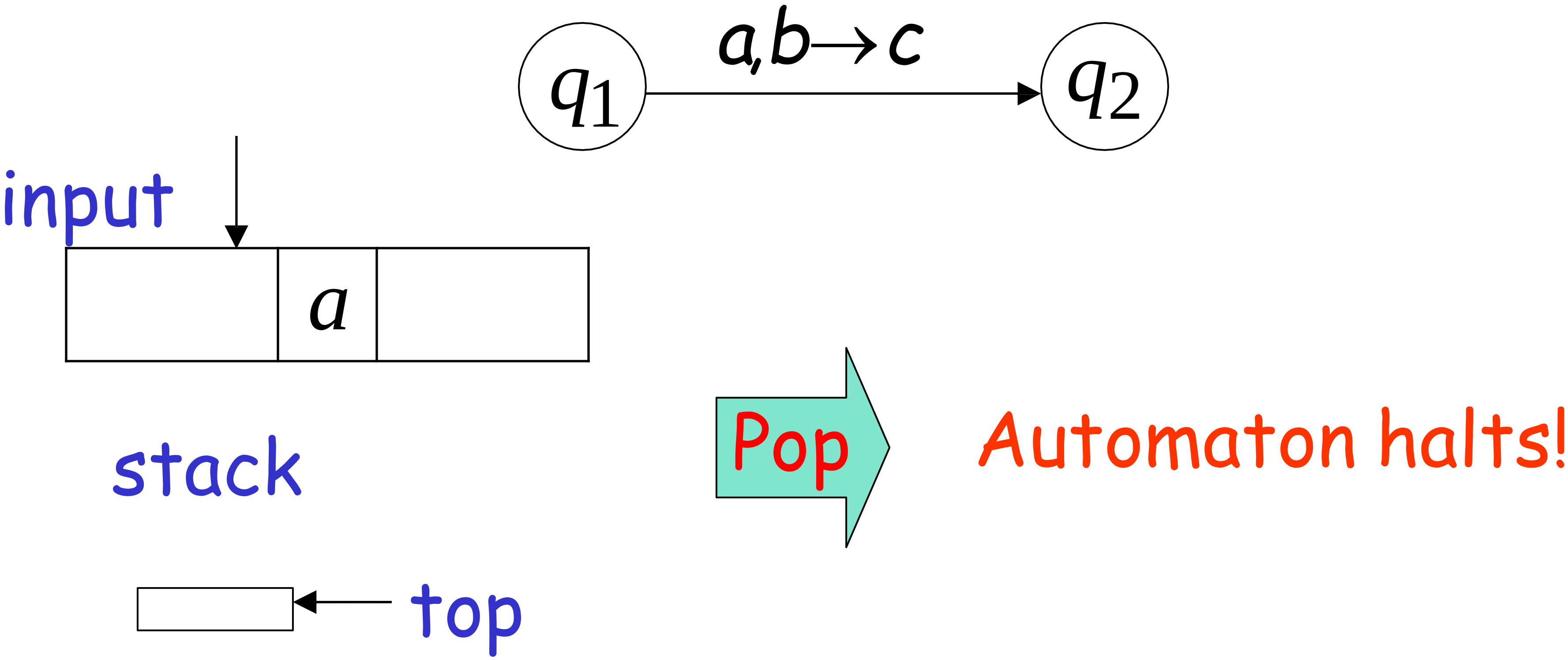


$a, \; b \rightarrow \lambda$
q_1
q_2
input
a
a
stack
b
h
e
$\$$
top
Pop
h
e
$\$$



$a, \lambda \to \lambda$
q_1
q_2
input
a
a
stack
b
h
e
$\$$
top
No Change
b
h
e
$\$$

Pop from Empty Stack

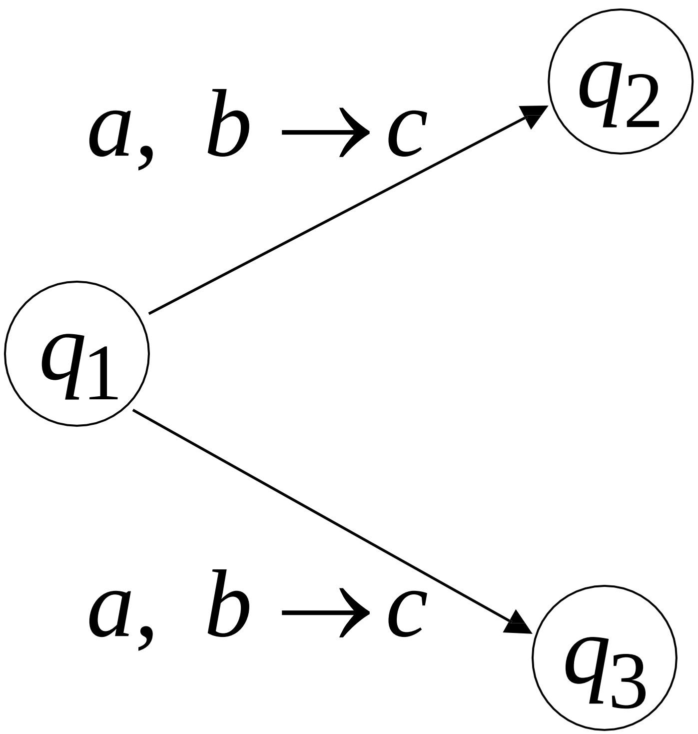


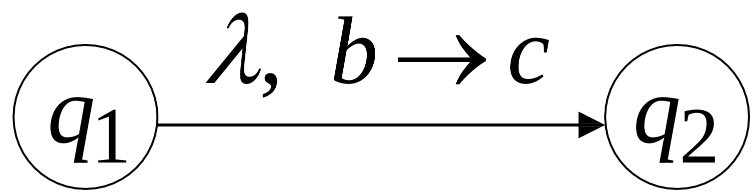
If the automaton attempts to pop from empty stack then it halts and rejects input

Non-Determinism

PDAs are non-deterministic

Allowed non-deterministic transitions



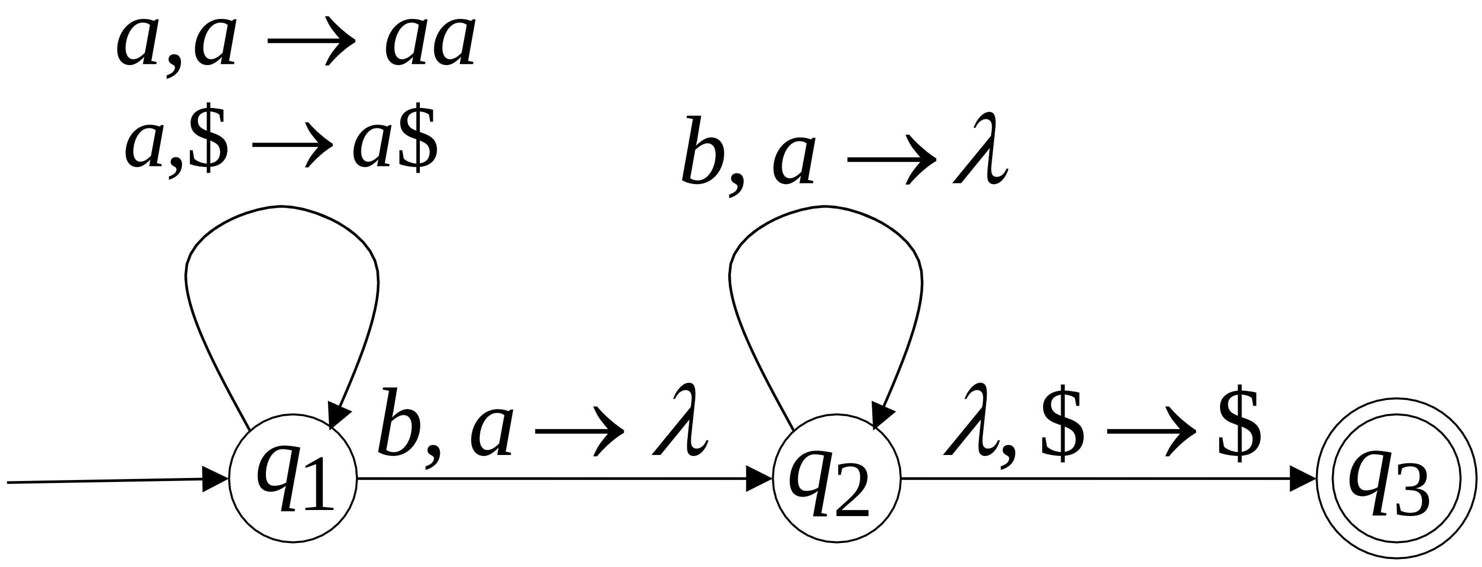


λ–transition

Example PDA

PDA M :

$$L(M) = \{a^n b^n : n \geq 1\}$$



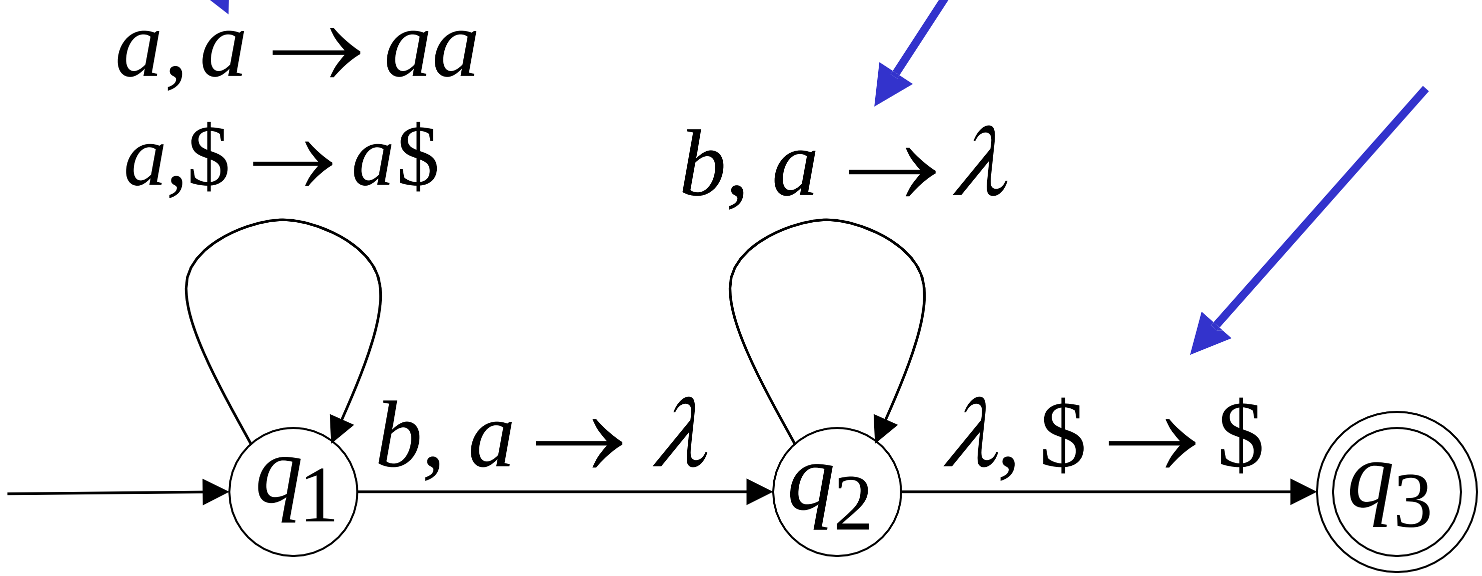
$$L(M) = \{a^n b^n : n \geq 0\}$$

Basic Idea:

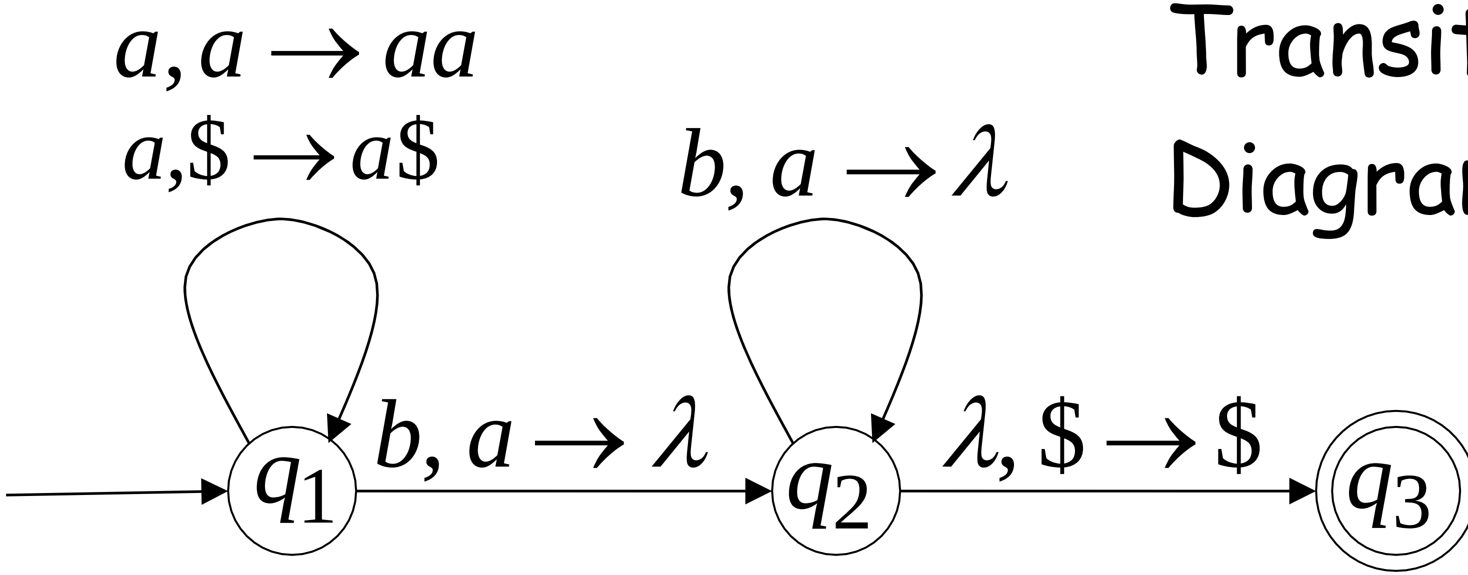
1. Push the a's on the stack

2. Match the b's on input with a's on stack

3. Match found



Transition Diagram



Transition function

$\delta(\ q_1 x\ a x\ \$)=(q_1, a\$)$

$\delta(\ q_1 x\ a x\ a)=(q_1, aa)$

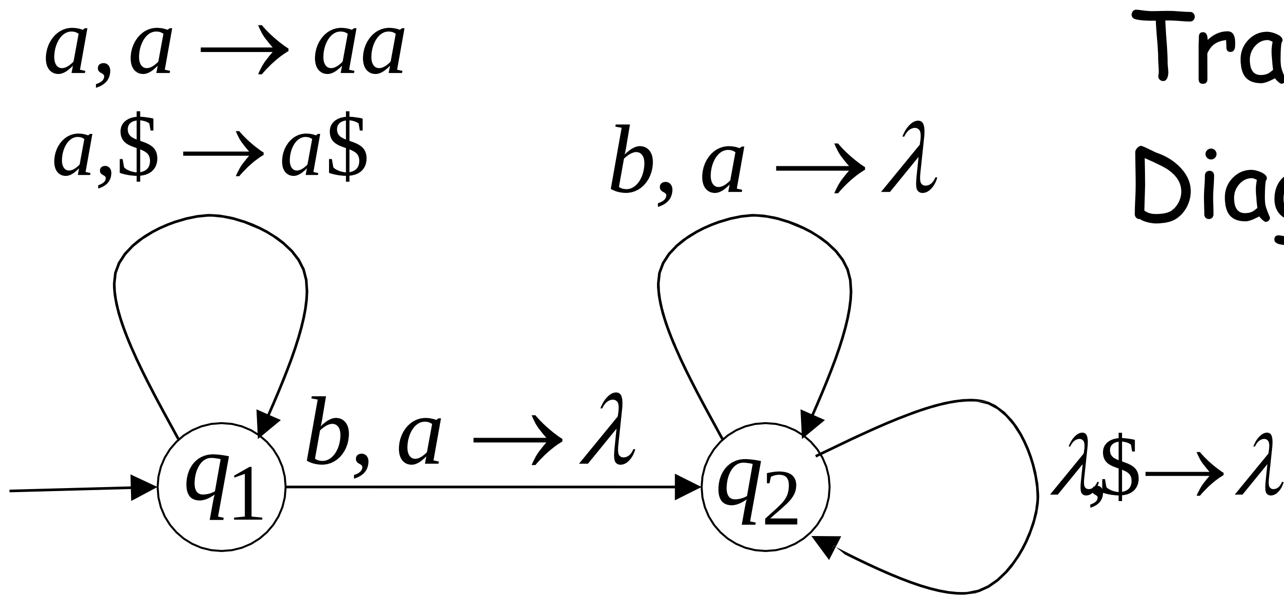
$\delta(\ q_1 x\ b x\ a)=(q_2, \varepsilon)$

$\delta(\ q_2 x\ b x\ a)=(q_2, \varepsilon)$

$\delta(\ q_2 x\ \varepsilon\ x\ \$)=(q_3, \$)$

PDA Acceptance by Final State

Transition Diagram



Transition function

$\delta(q_1 \times a \times \$) = (q_1, a\$)$

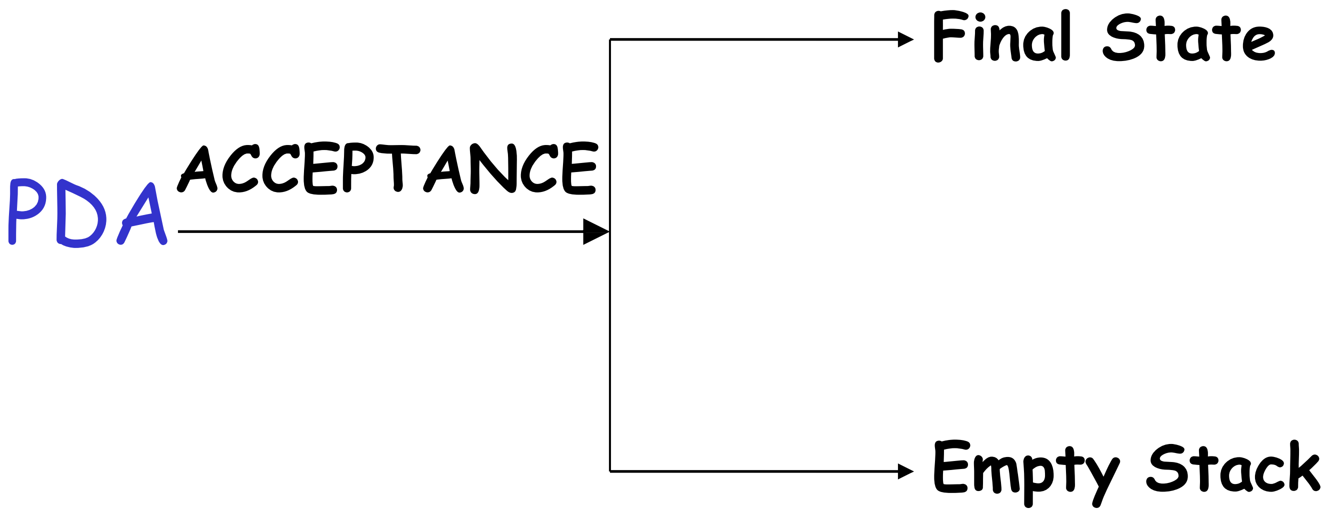
$\delta(q_1 \times a \times a) = (q_1, aa)$

$\delta(q_1 \times b \times a) = (q_2, \varepsilon)$

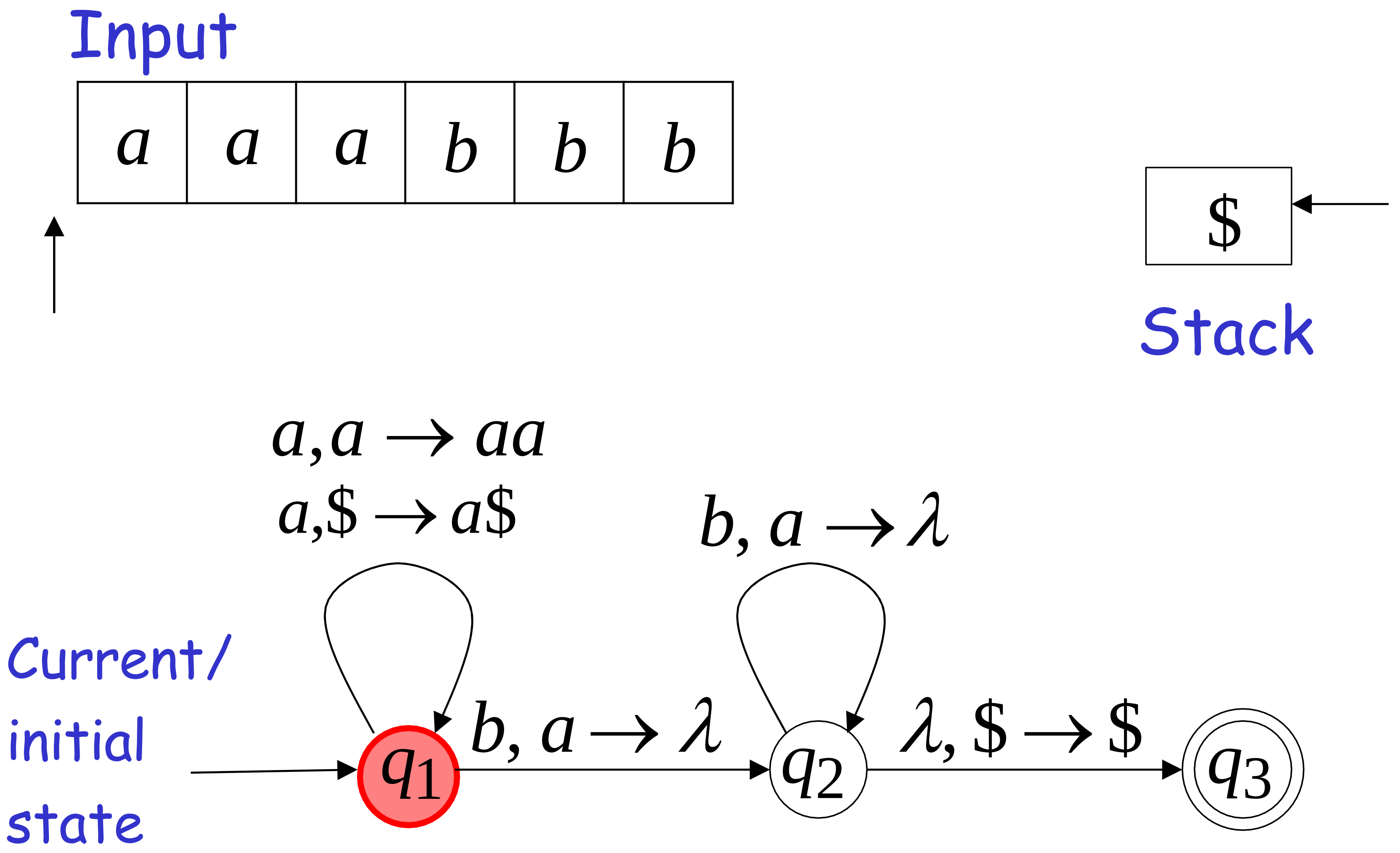
$\delta(q_2 \times b \times a) = (q_2, \varepsilon)$

$\delta(q_2 \times \varepsilon \times \$) = (q_2, \varepsilon)$

PDA Acceptance by Empty Stack

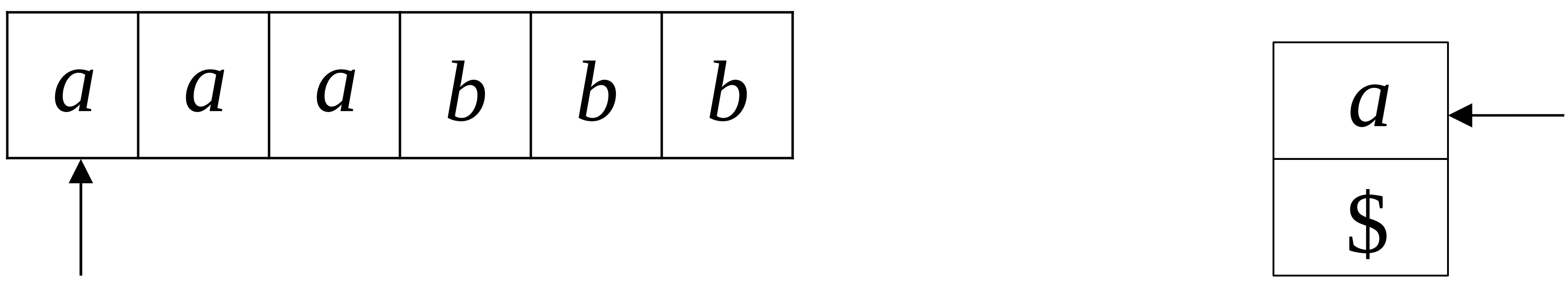

PDA
ACCEPTANCE
Final State
Empty Stack

Execution Example: Time 0

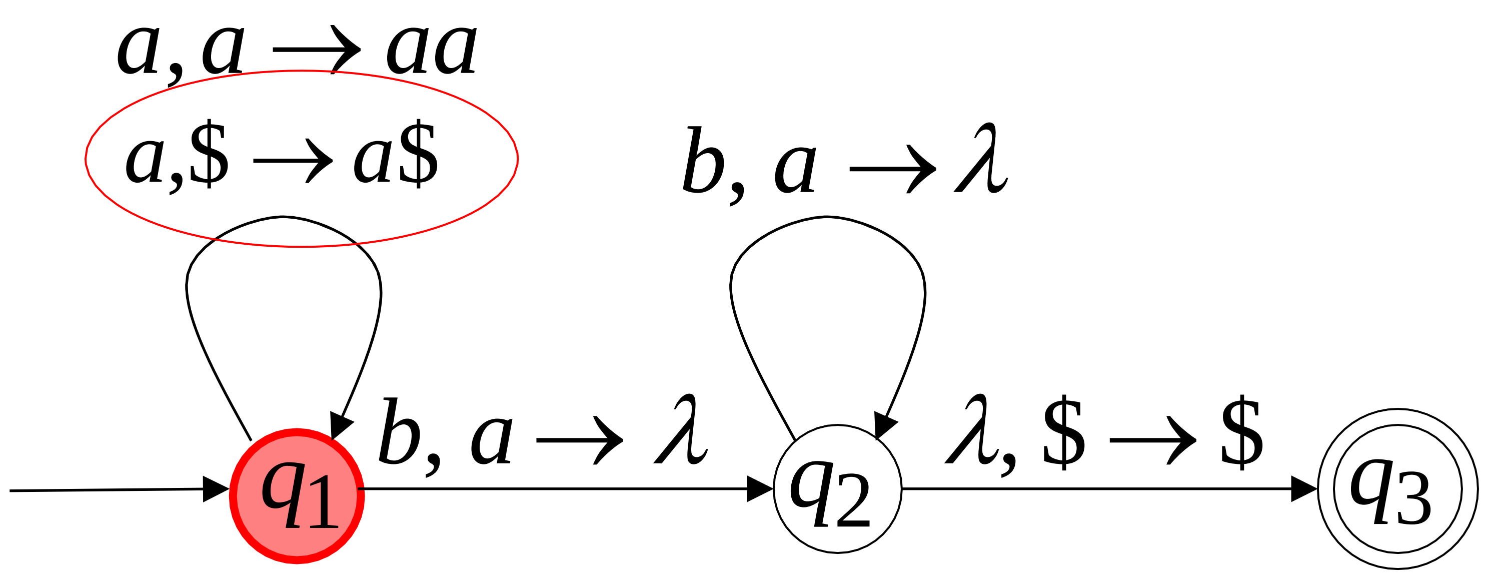


Time 1

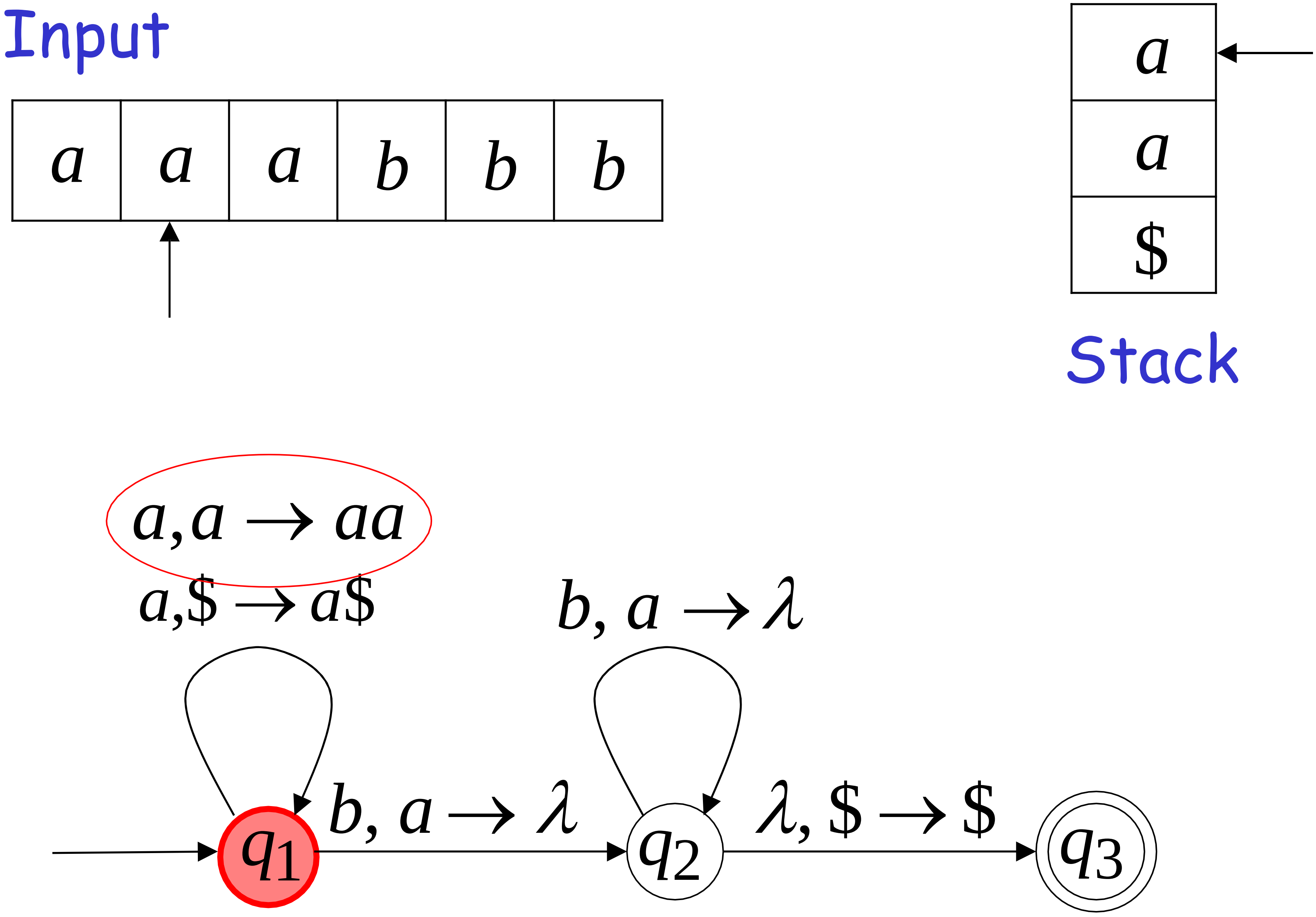
Input



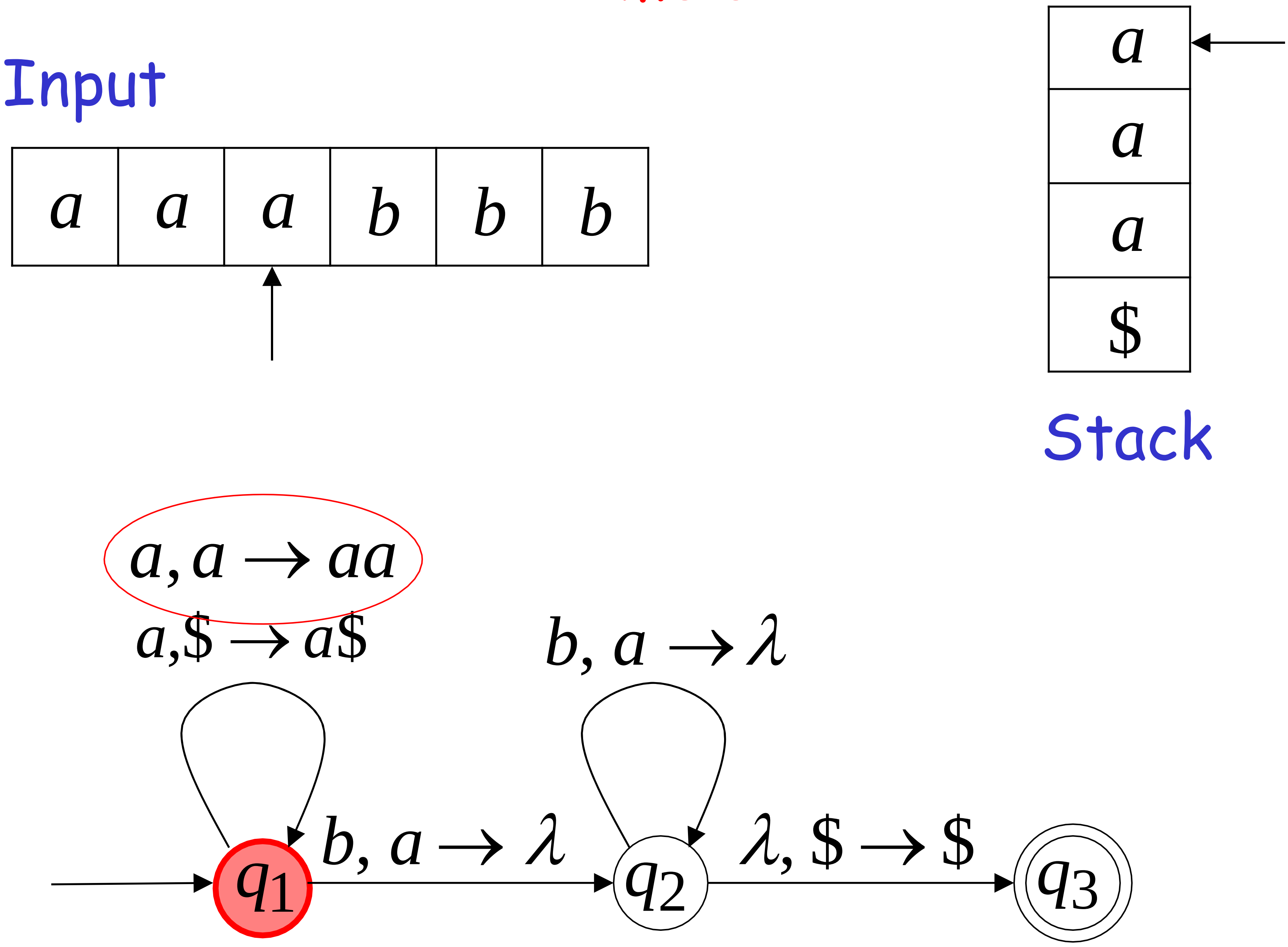
Stack



Time 2

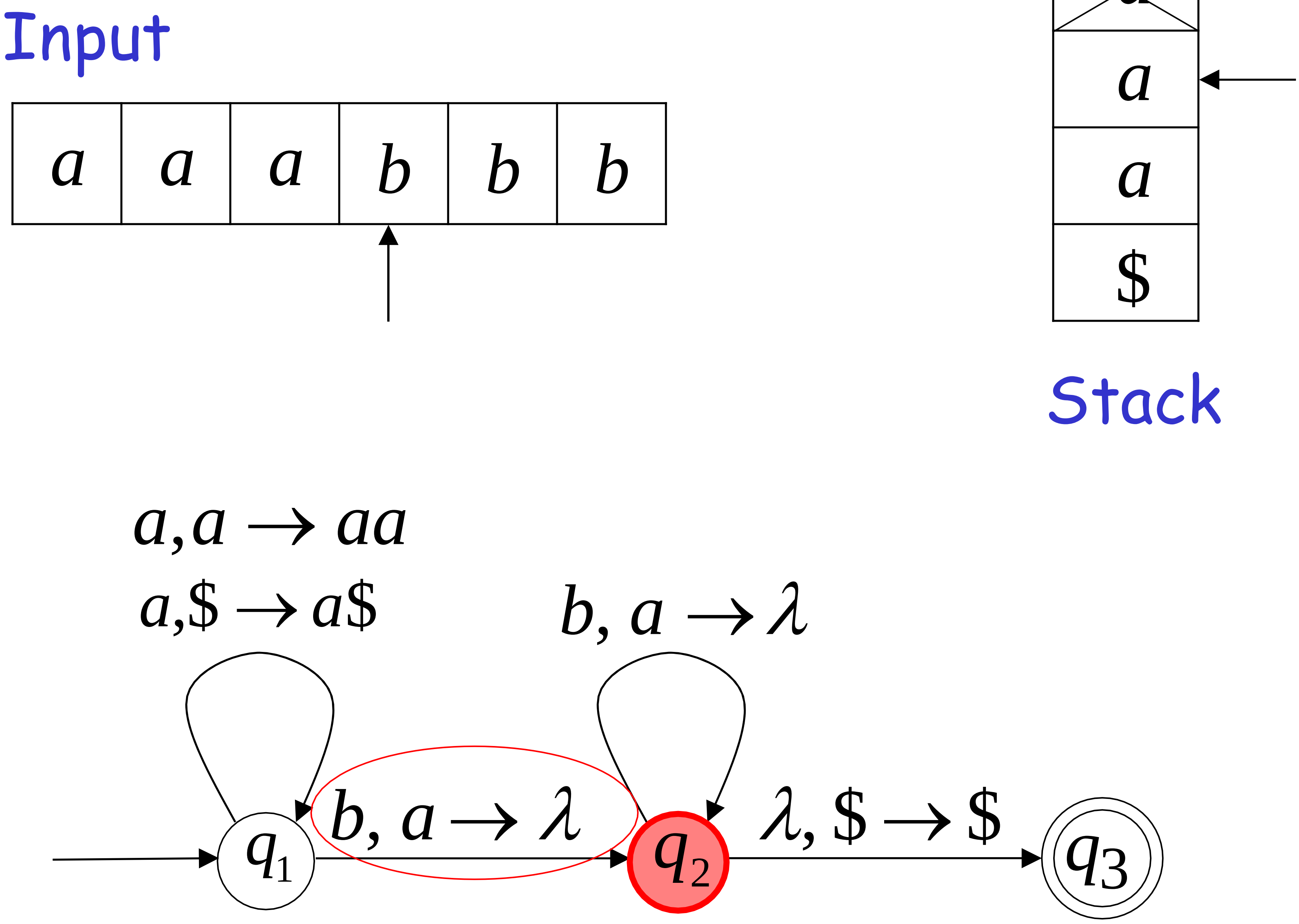


Time 3



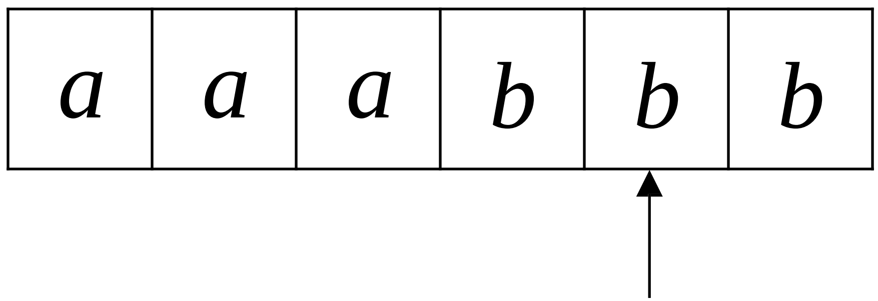
Input
a a a b b b
a
a
a
$
Stack
$a, a \rightarrow aa$
$a, \$ \rightarrow a\$$
$b, a \rightarrow \lambda$
$b, a \rightarrow \lambda$
$\lambda, \$ \rightarrow \$$
q_1
q_2
q_3

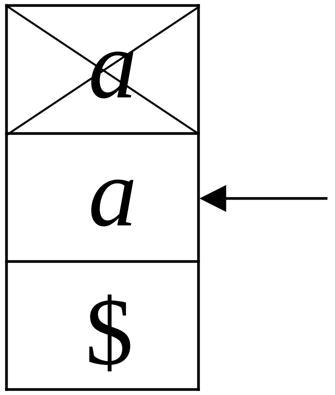
Time 4



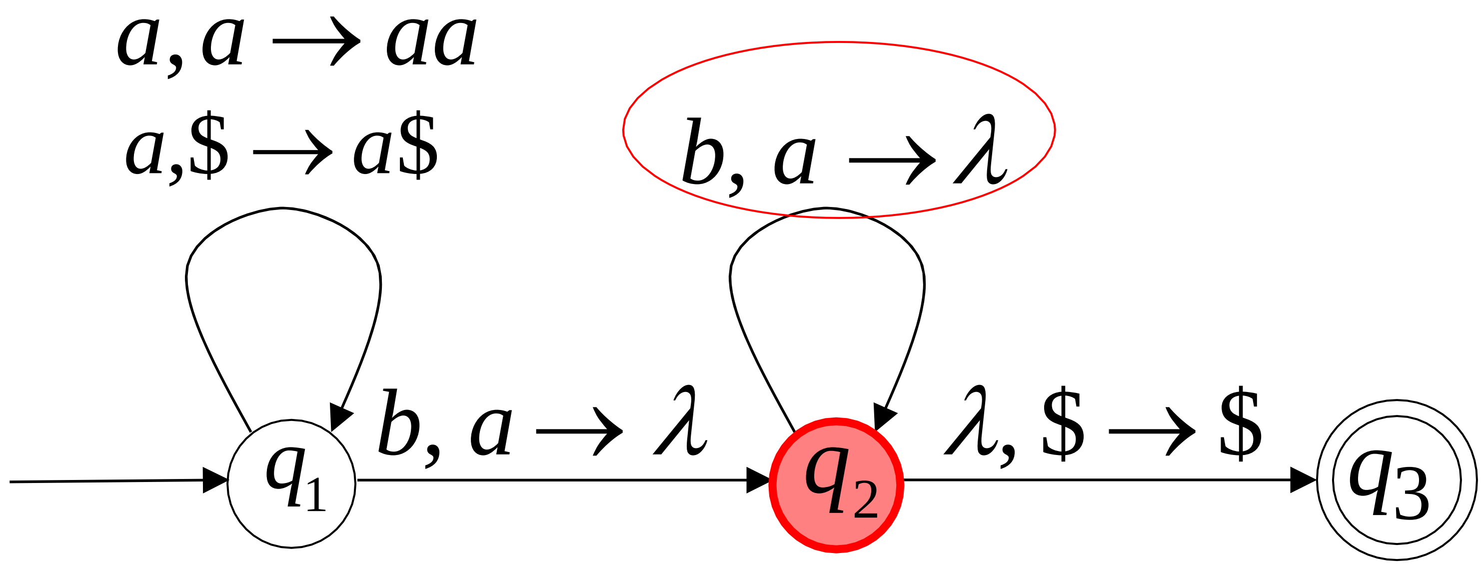
Time 5

Input

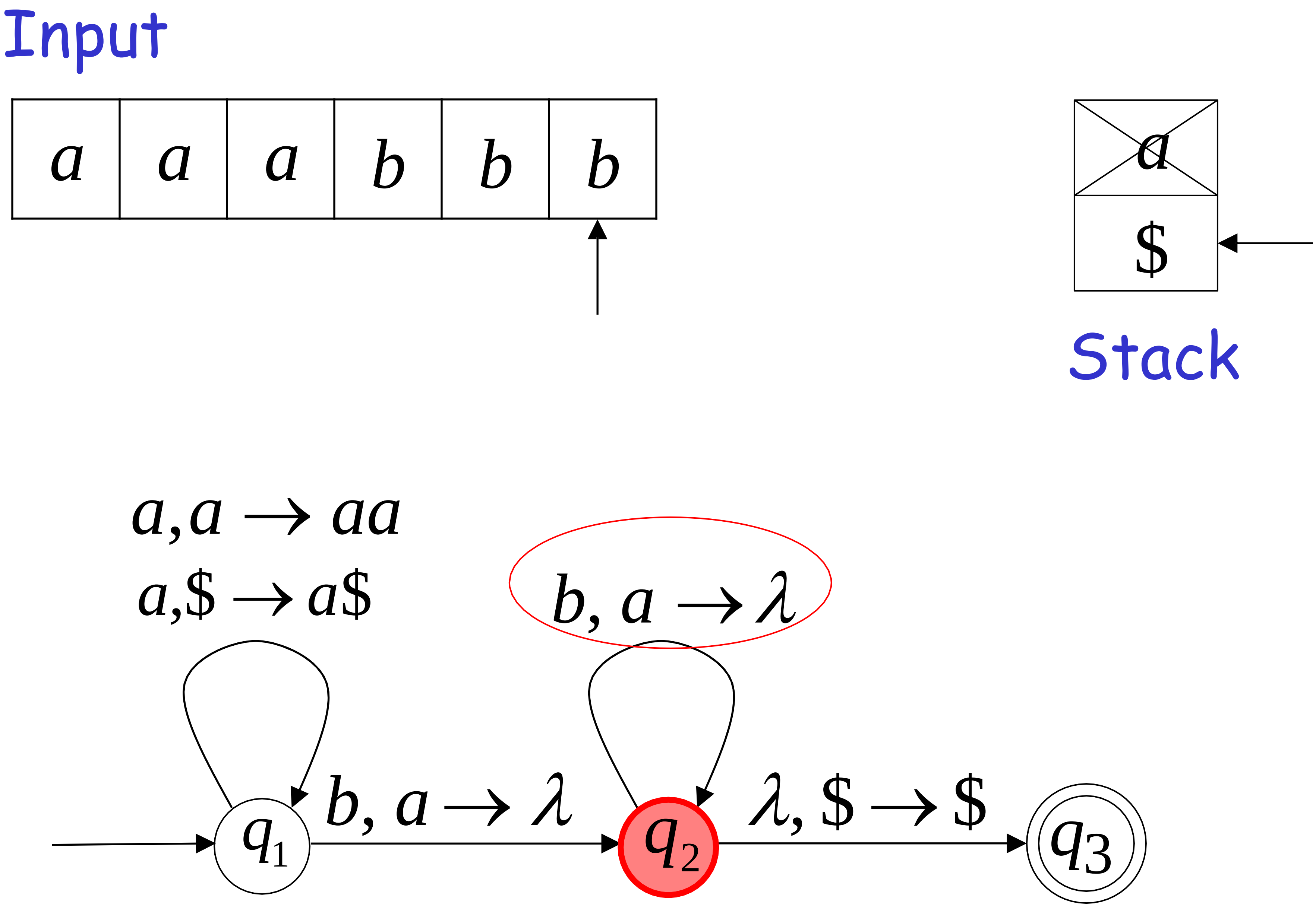




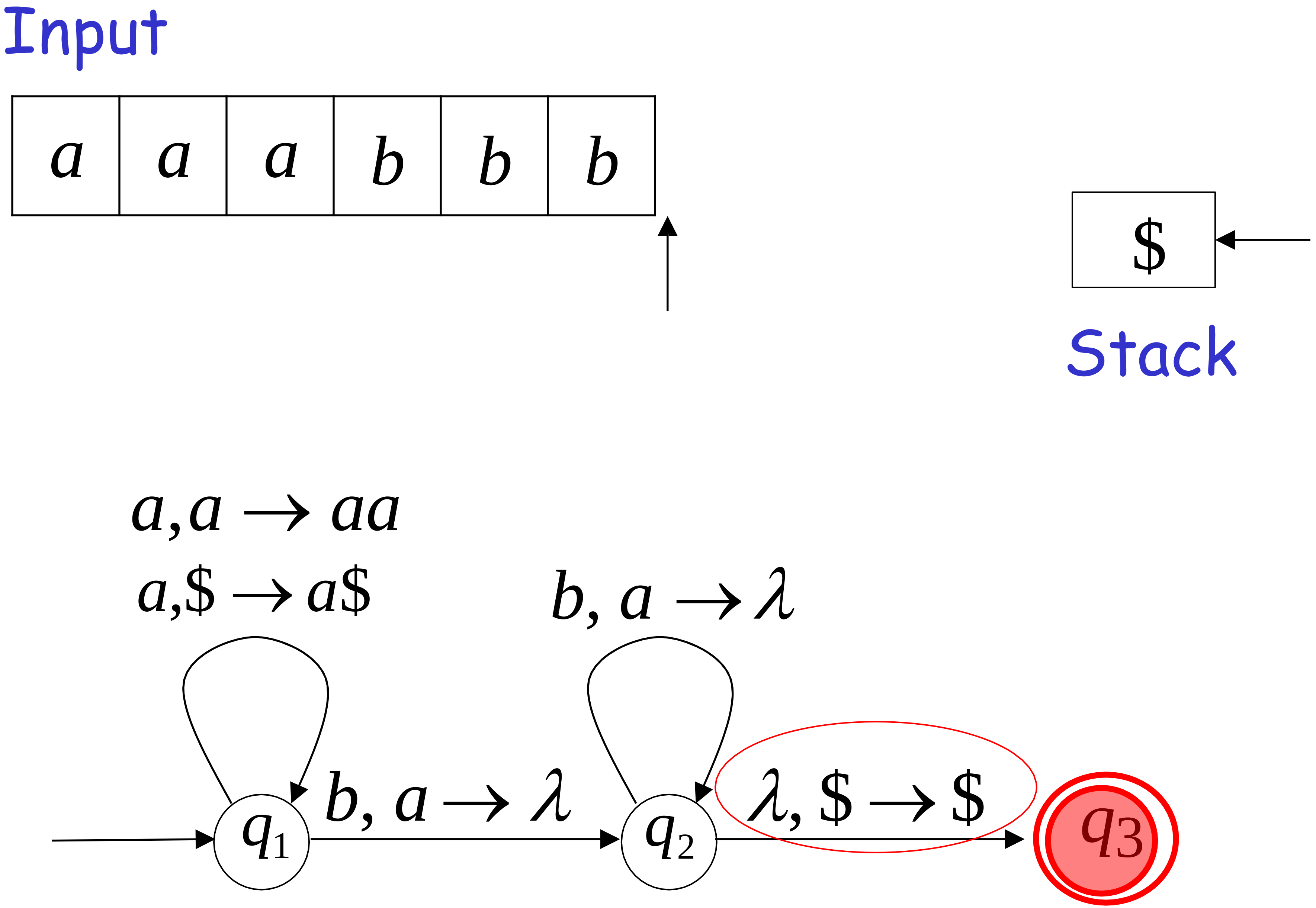
Stack



Time 6


Input
a a a b b b
$\cancel{a}$
$\$$
Stack
$a, a \to aa$
$a, \$ \to a\$$
$b, a \to \lambda$
$b, a \to \lambda$
$\lambda, \$ \to \$$
q_1
q_2
q_3

Time 7



Input
a a a b b b
$
Stack
$a,a \rightarrow aa$
$a,\$ \rightarrow a\$$
$b, a \rightarrow \lambda$
$b, a \rightarrow \lambda$
$\lambda, \$ \rightarrow \$$
q_1
q_2
q_3

A string is accepted if there is
a computation such that:

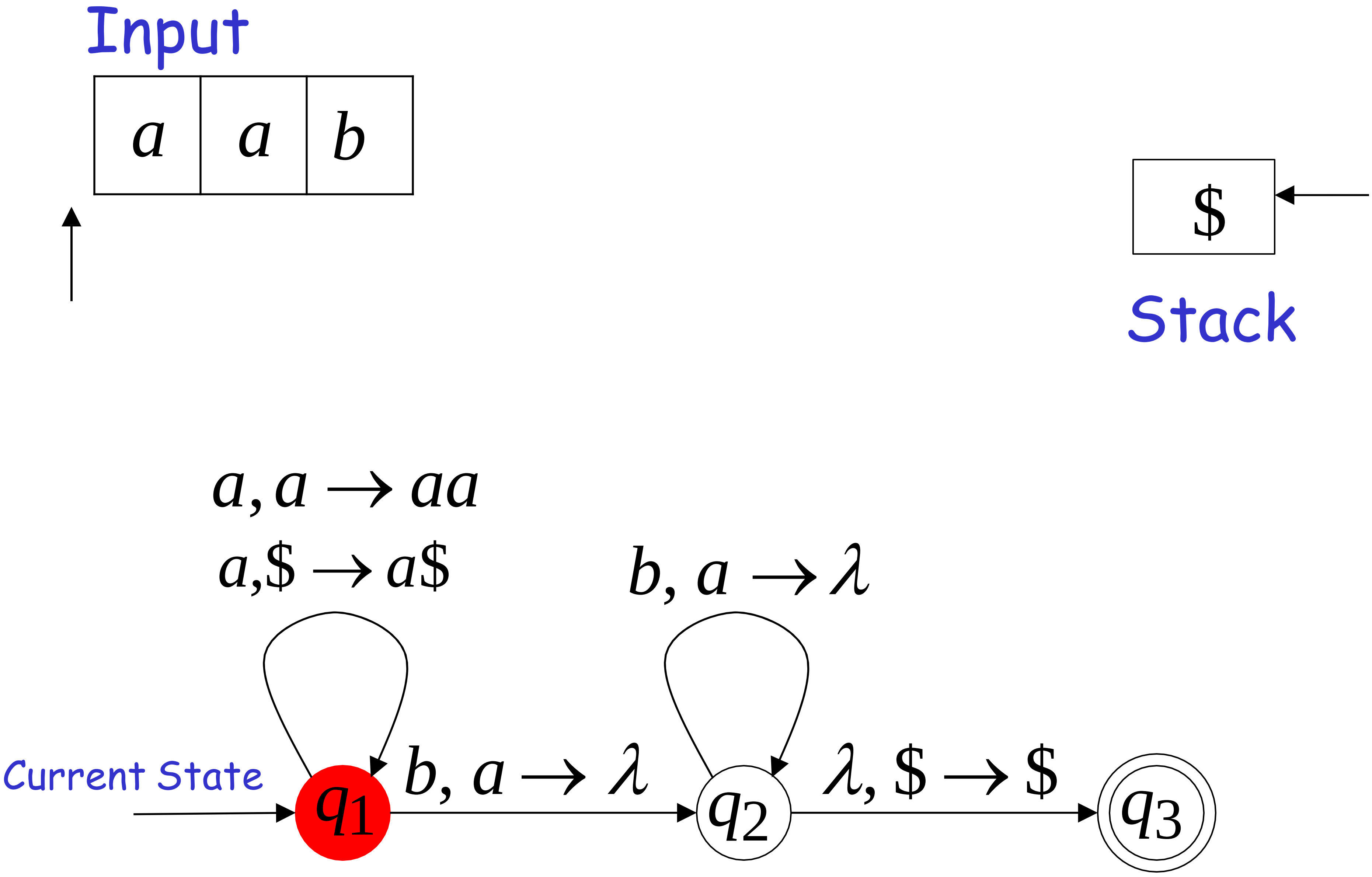
All the input is consumed

AND

The last state is an accepting state

we do not care about the stack contents
at the end of the accepting computation

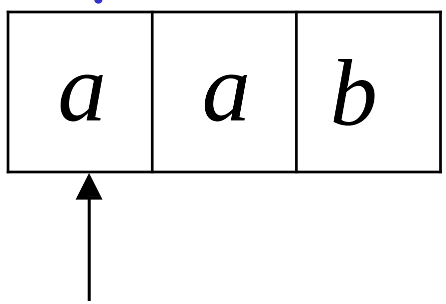
Rejection Example: Time 0

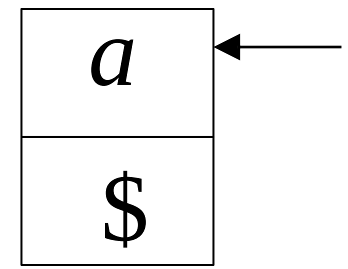


Rejection Example:

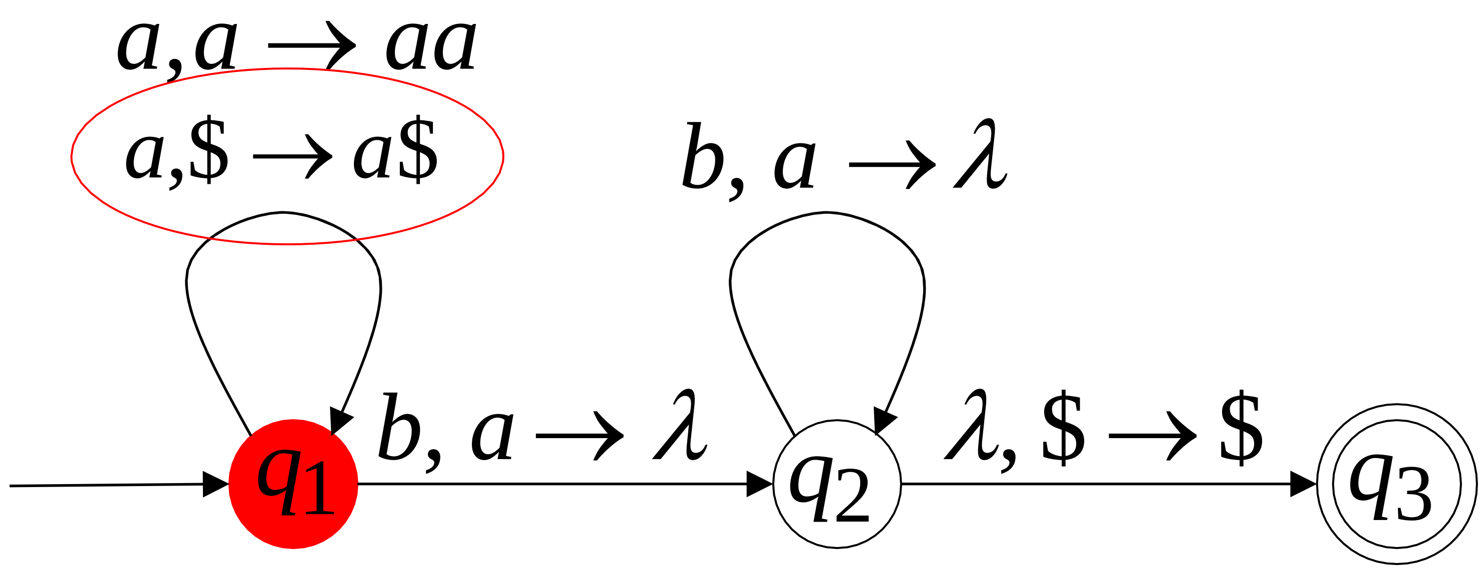
Time 2

Input



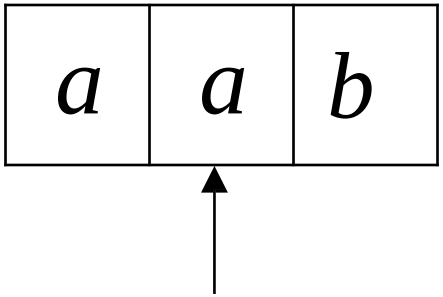


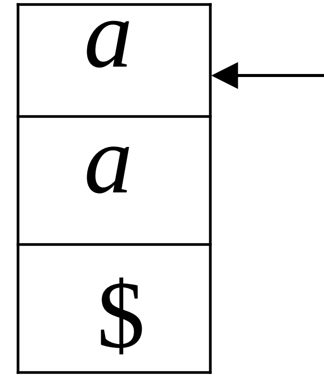
Stack



Rejection Example: Time 3

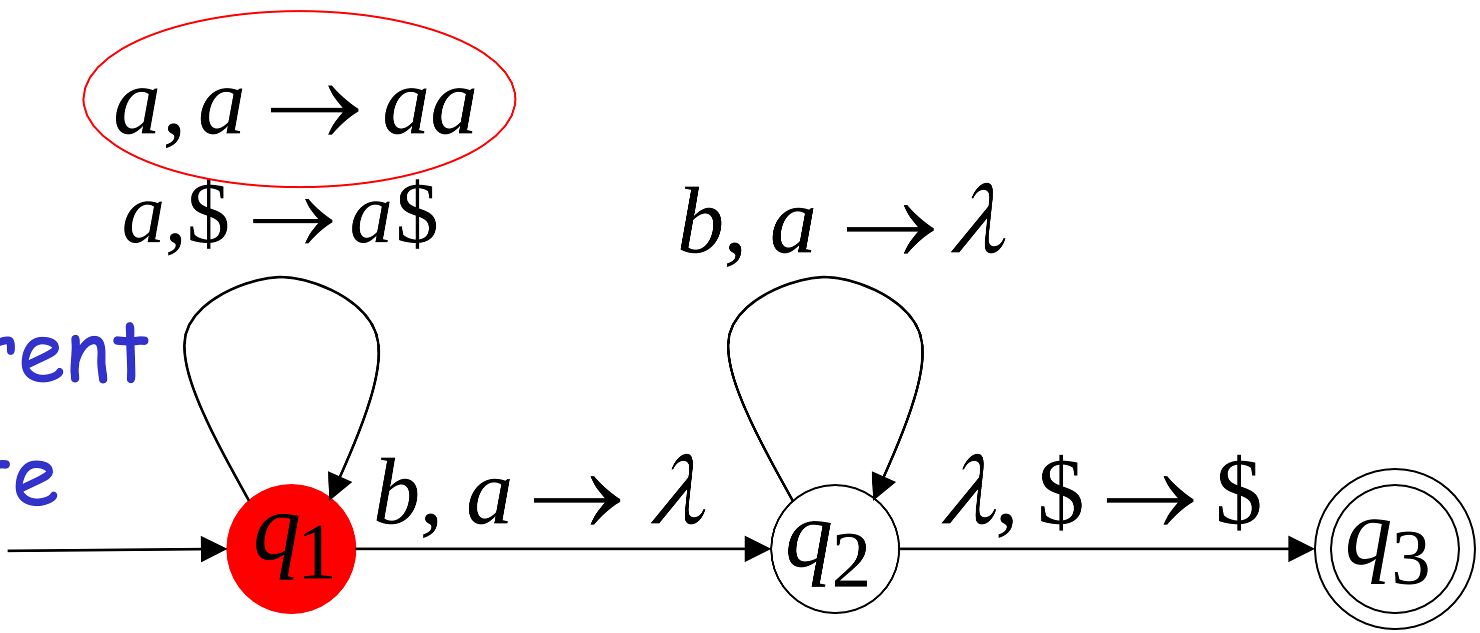
Input



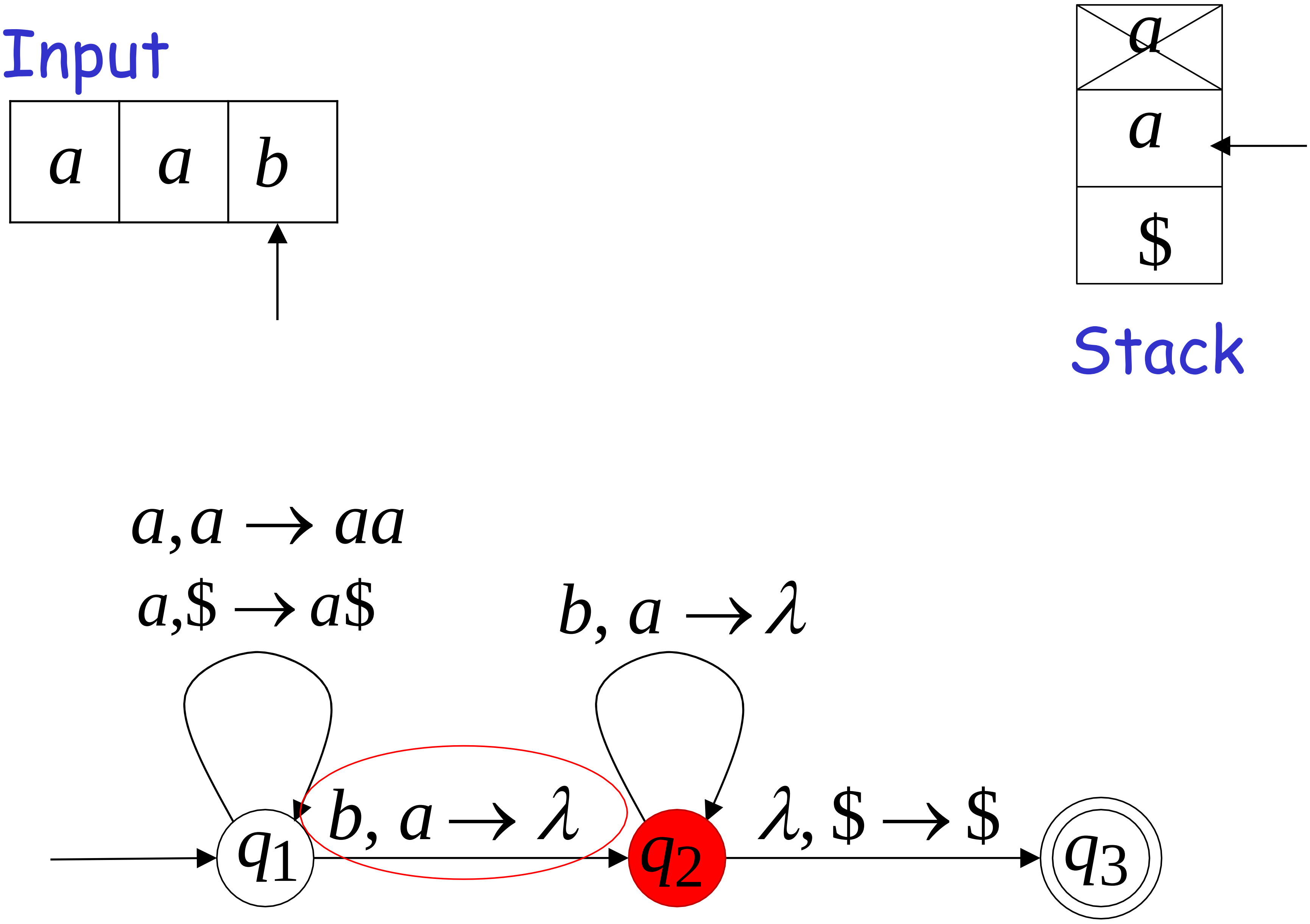


Stack

current state



Rejection Example: Time 4



Rejection Example:

Time 4

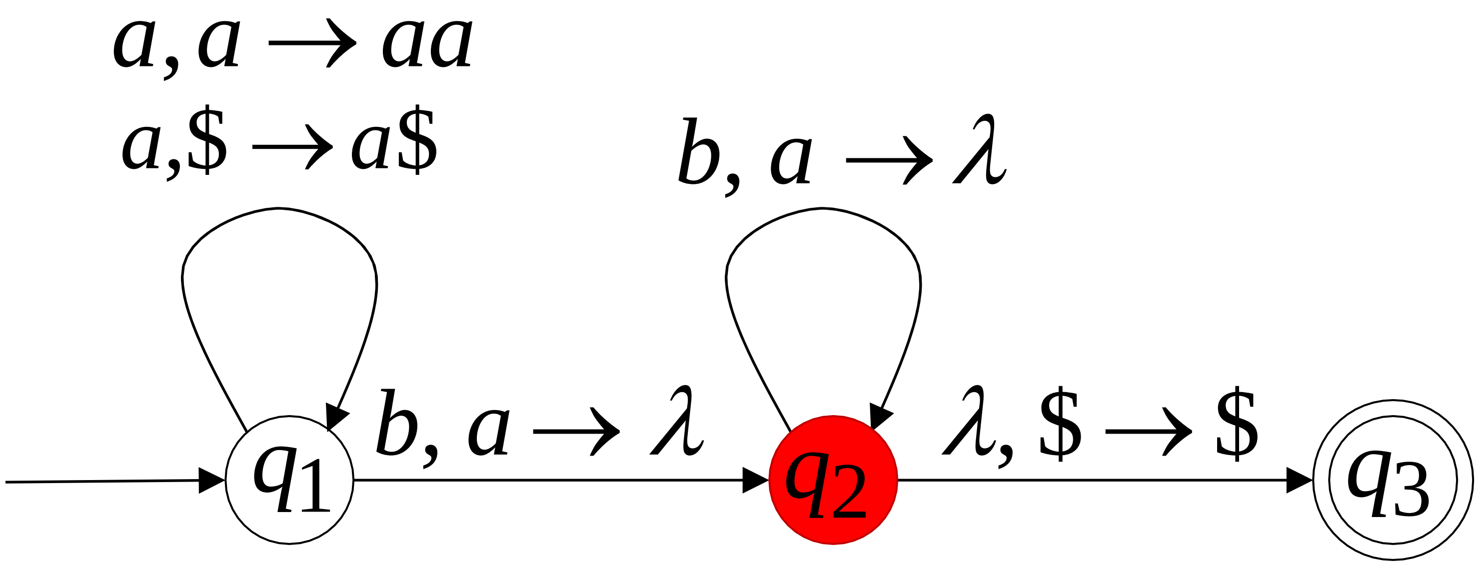
Input

a	a	b

$\cancel{a}$
a
$\$$

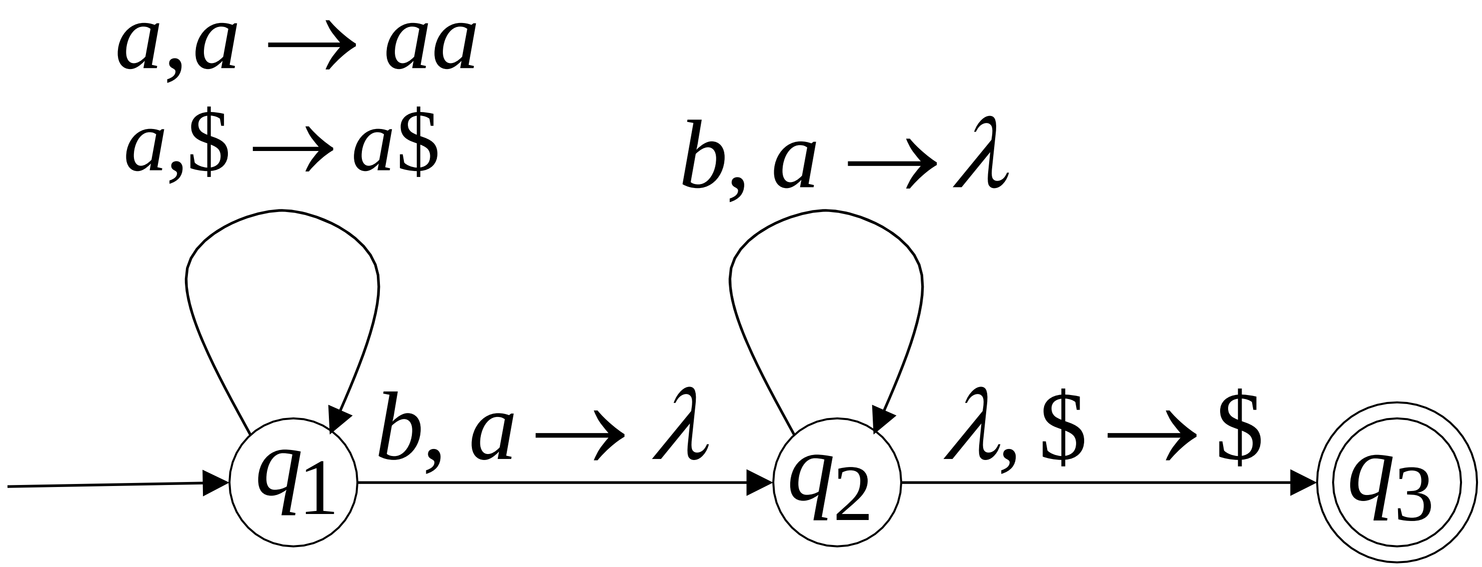
Stack

reject



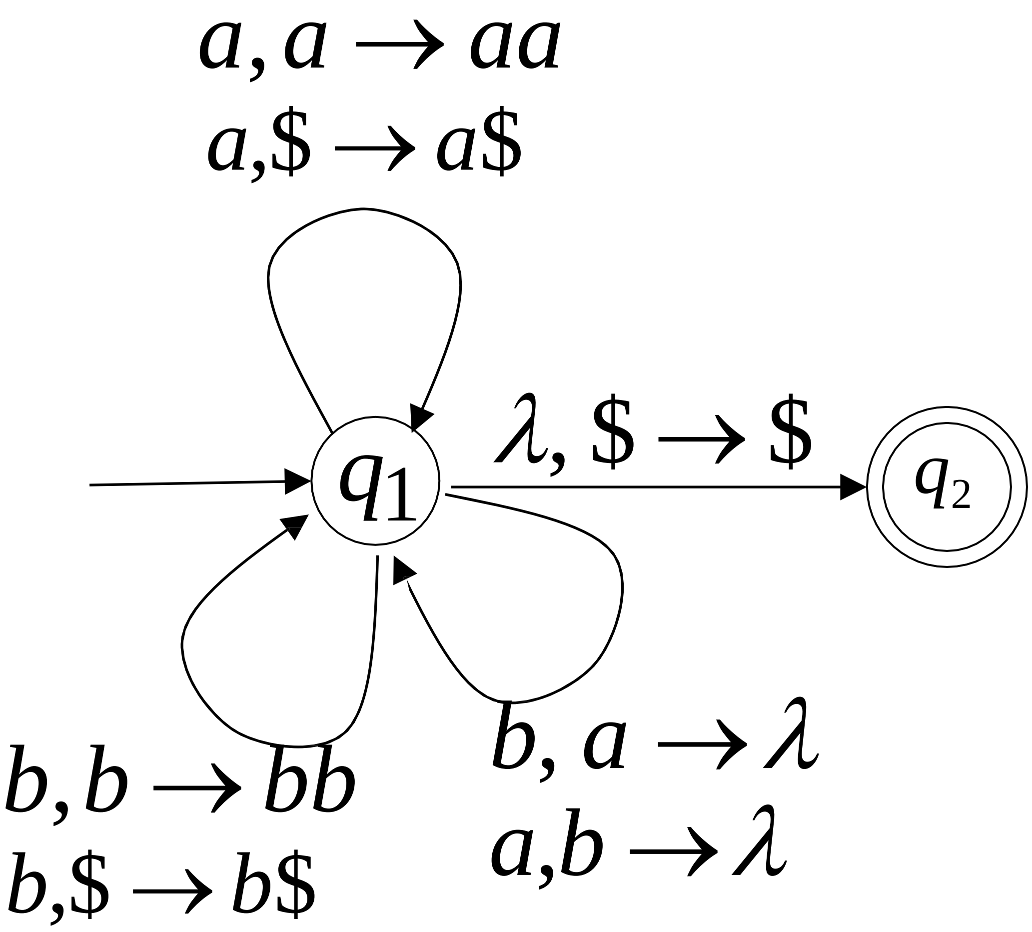
There is no accepting computation for aab

The string aab is rejected by the PDA



Example PDA

PDA M :L(M) such that $n_a(w)=n_b(w), w€(a,b)^*$

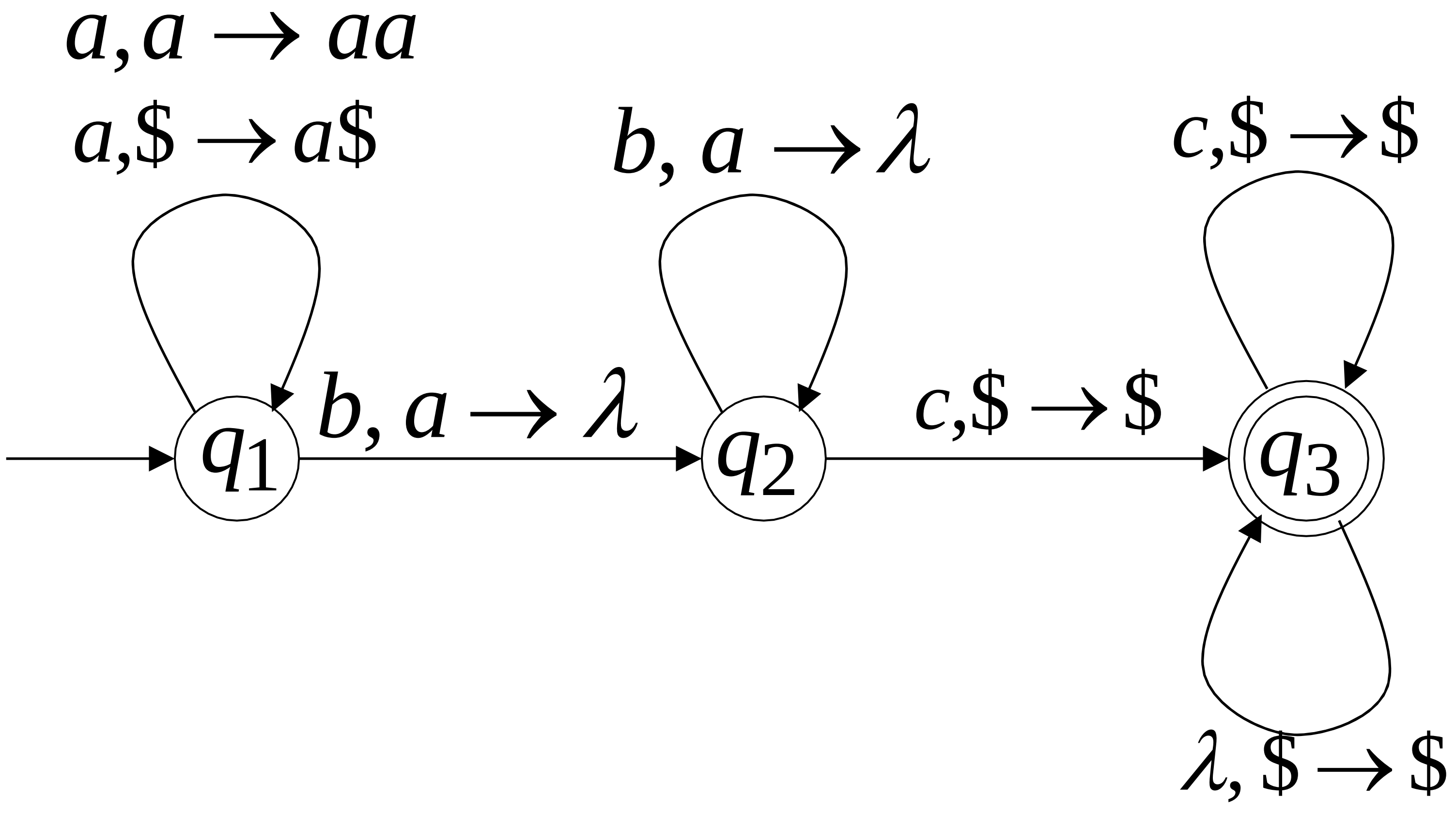


Review

- PDA
- Formal Definition of PDA
- Transition Function
- Transition Diagram
- Design of PDA
- Examples
- Acceptance by Final State
- Acceptance by Empty Stack

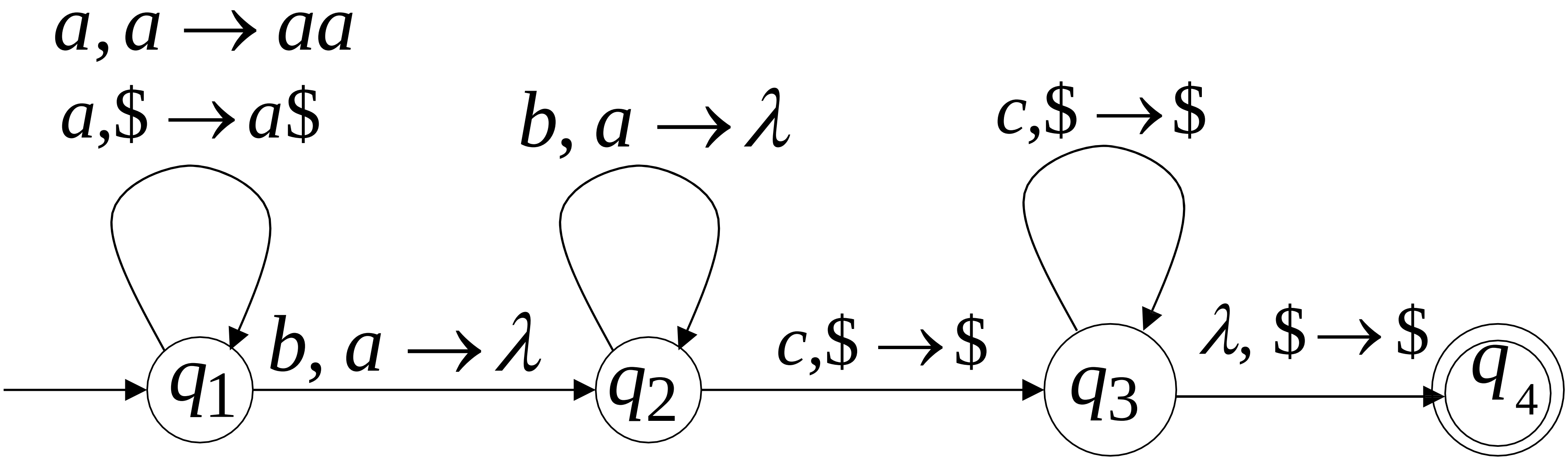
Example PDA

PDA M :L(M) such that $a^n b^n c^m$| n, m>=1



Example PDA

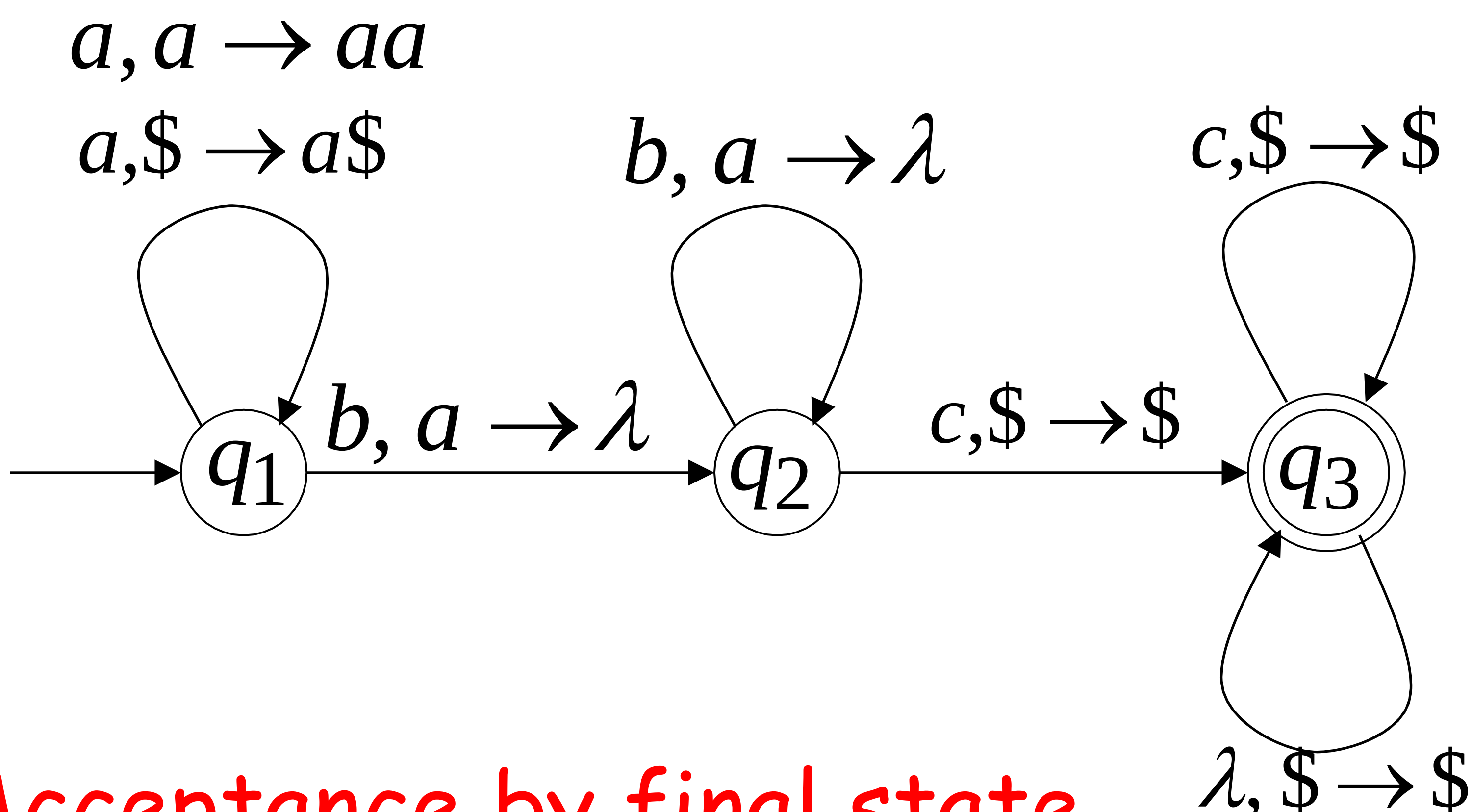
PDA M :L(M) such that $a^n b^n c^m$| n, m>=1



Acceptance by final state

Example PDA

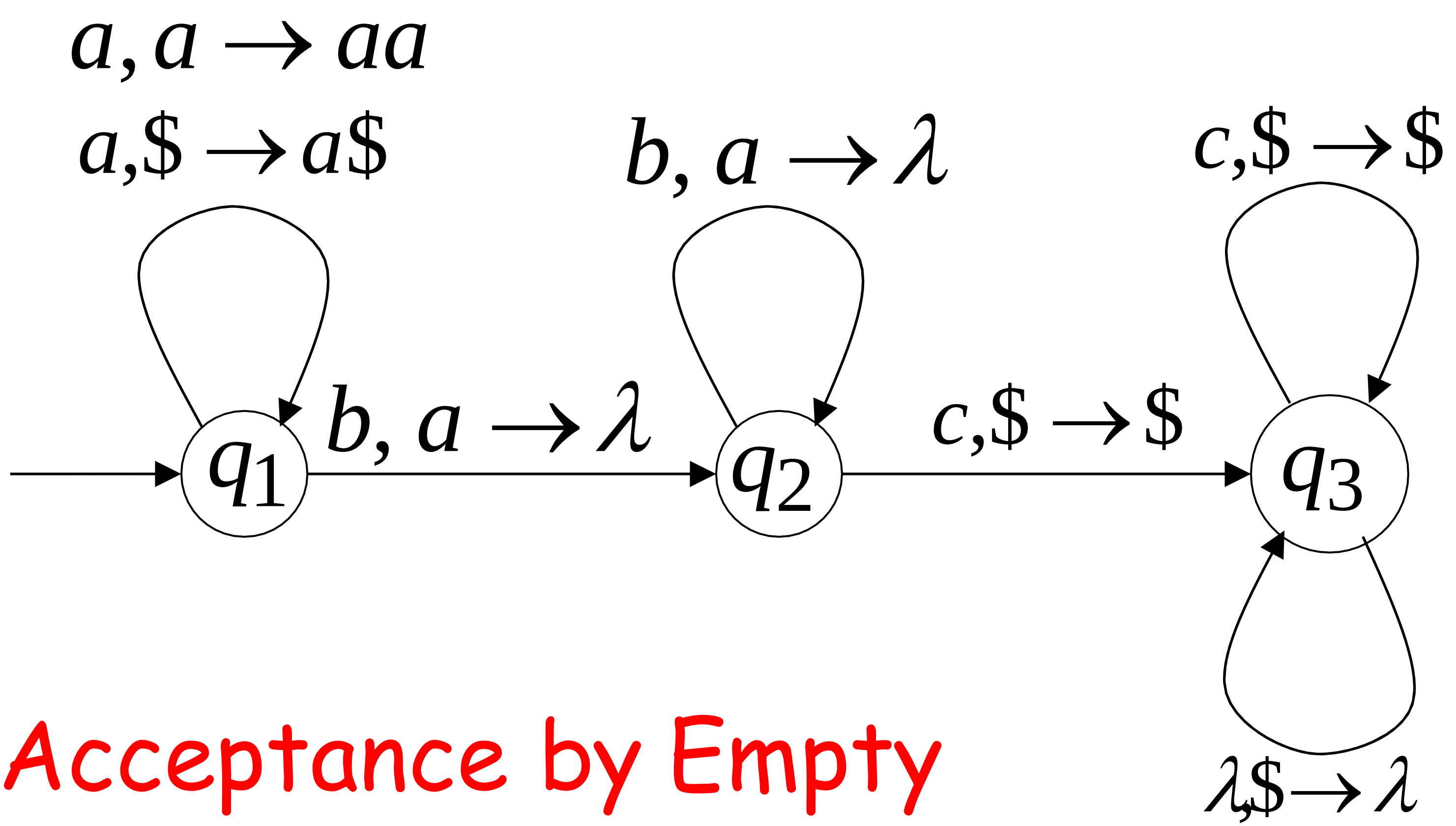
PDA M :L(M) such that $a^n b^n c^m$| n, m>=1



Acceptance by final state

Example PDA

PDA M :L(M) such that $a^n b^n c^m$| n, m>=1

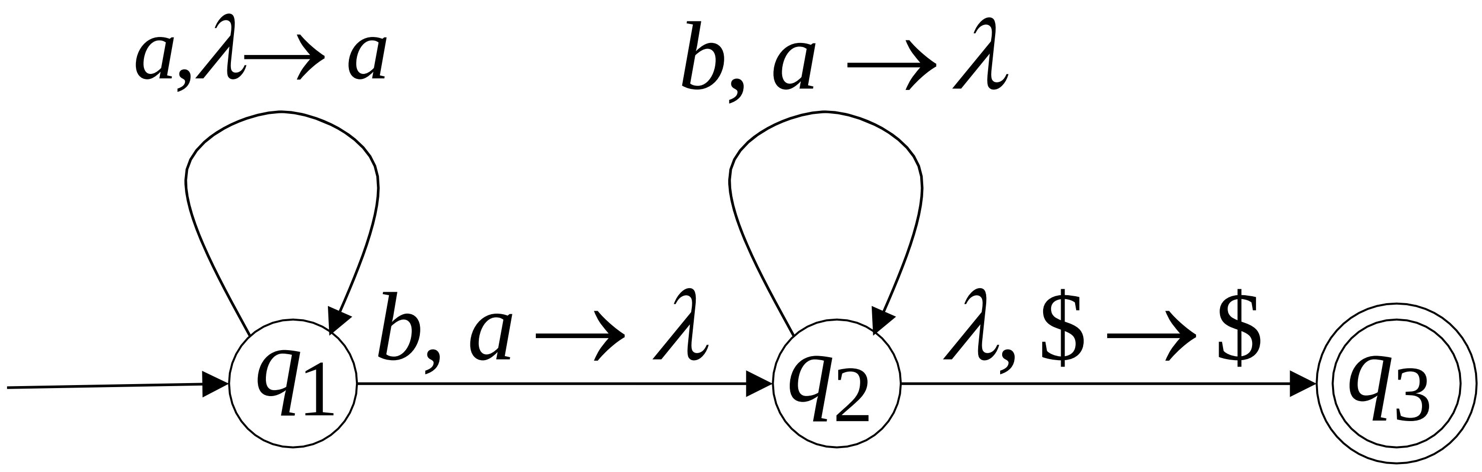


Acceptance by Empty stack

Example PDA

PDA M :

$$L(M) = \{a^n b^n : n \geq 1\}$$



Problems

1. Construct PDA that accepts language

   $L = \{ a^n b^m c^n \mid m, n > 1\}$

   (Nov-2017 EndSem 8 Marks)

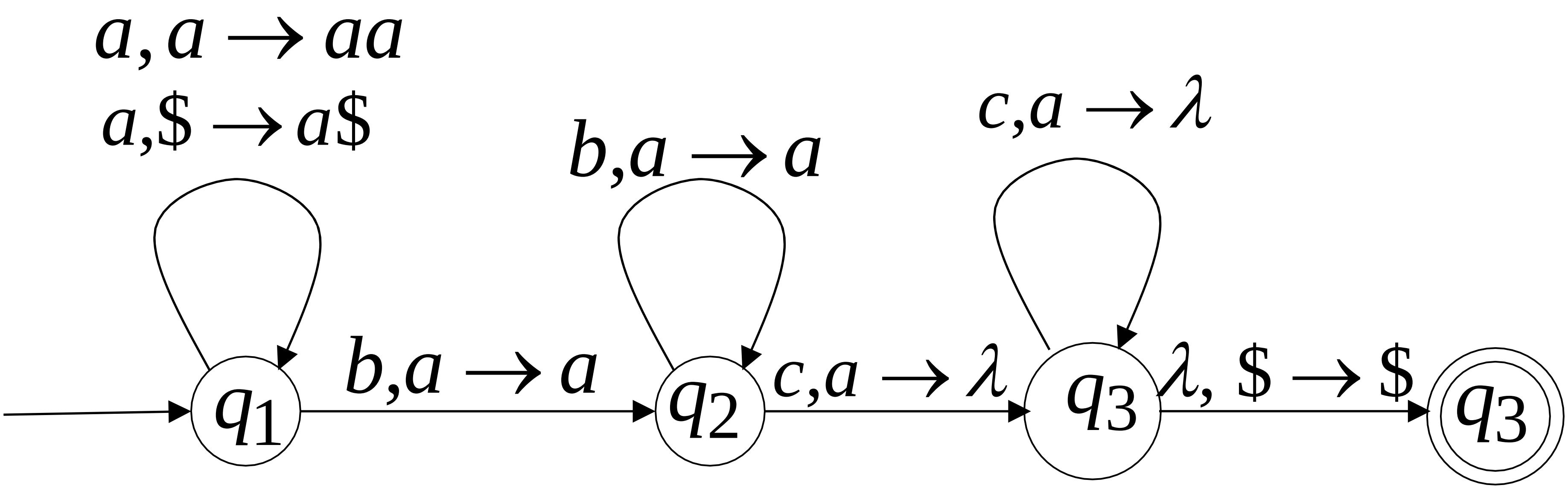
2. Design a PDA for the language

   $L = \{a^n b^m c^n \mid m, n >= l\}$ by empty stack.

   (Nov-2016 EndSem 8 Marks)

Example PDA

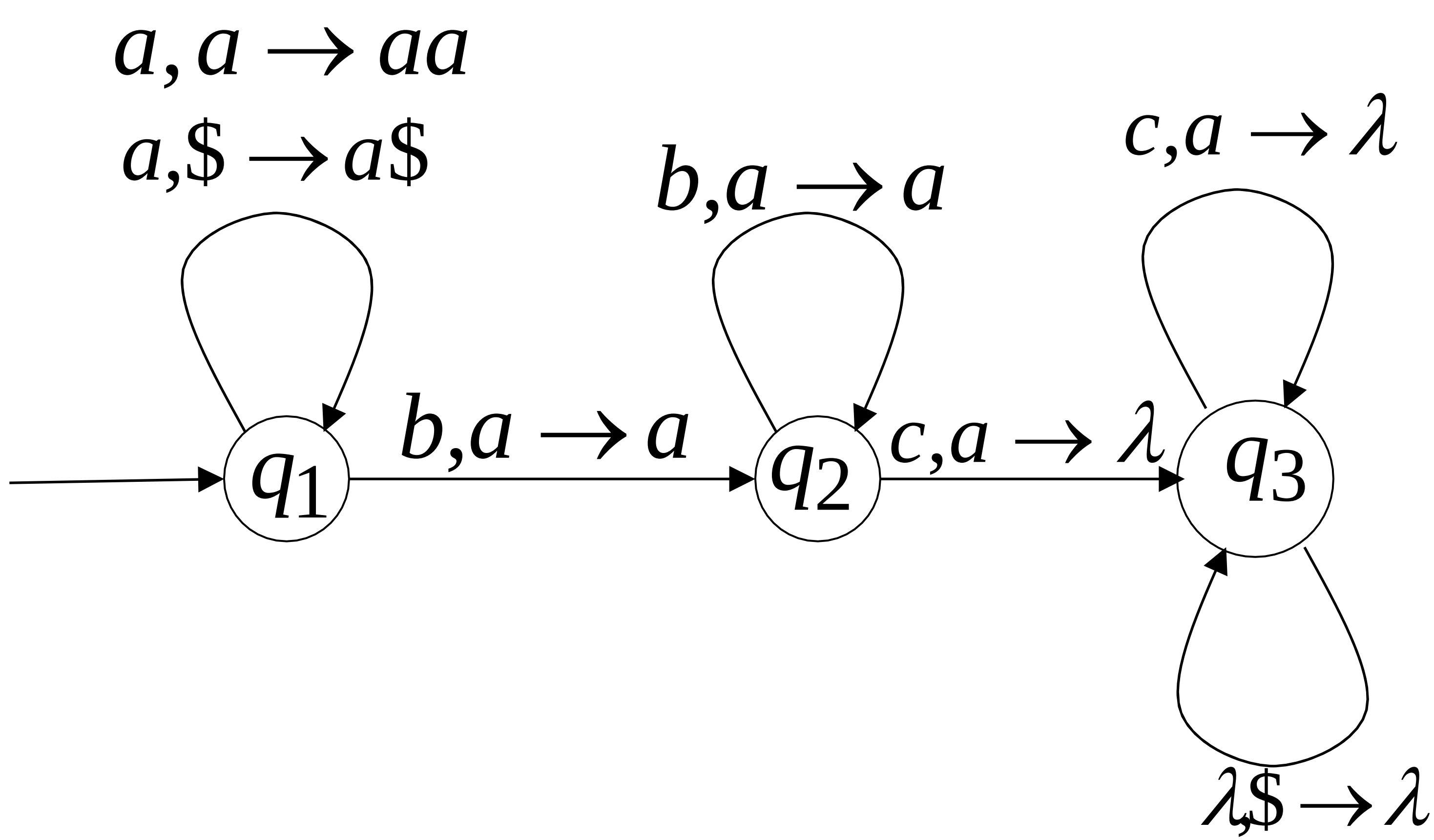
PDA M :L(M) such that $a^n b^m c^n$| n, m>=1



Acceptance by final state

Example PDA

PDA M :L(M) such that $a^n b^m c^n$| n, m>=1

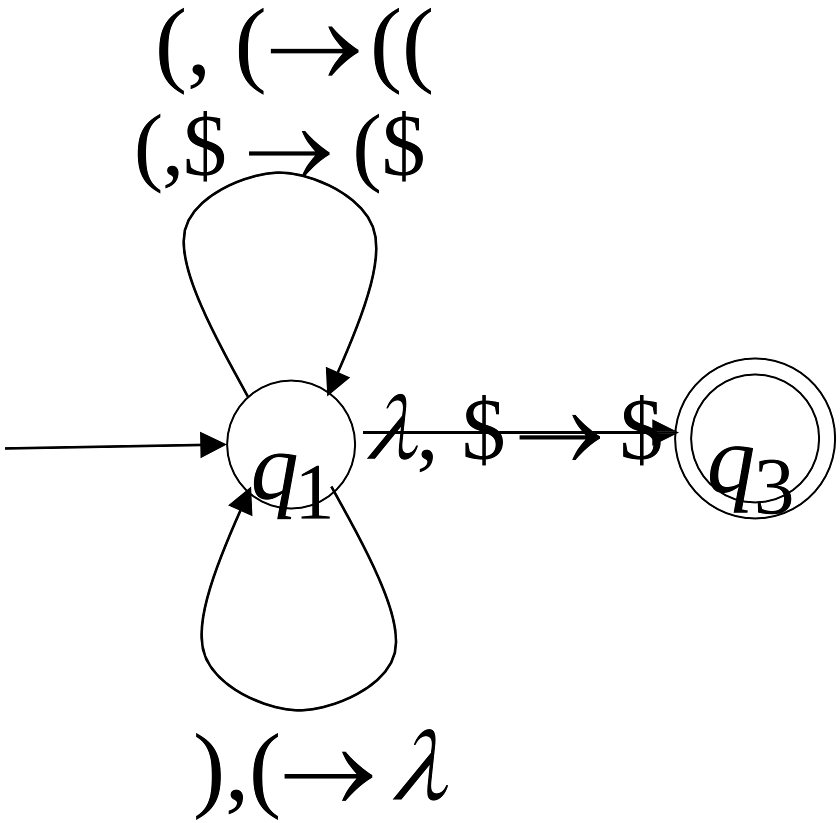


Acceptance by Empty Stack

Example PDA

2.Construct PDA to check for well formedness of paranthesis.

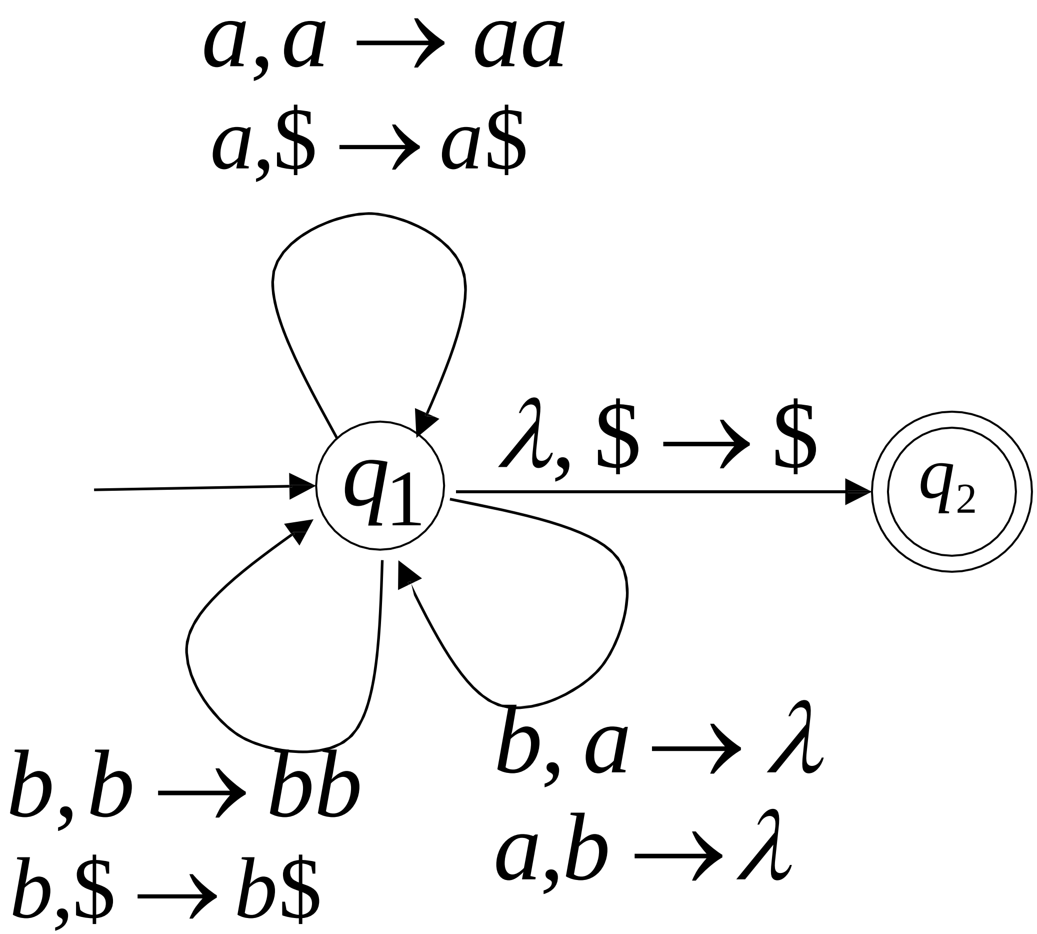
(Nov-2017 EndSem 8 Marks)



Acceptance by final state

Construct a PDA for the language described as “The set of all strings over $\sum = \{a, b\}$ with equal no. of a’s and b’s.

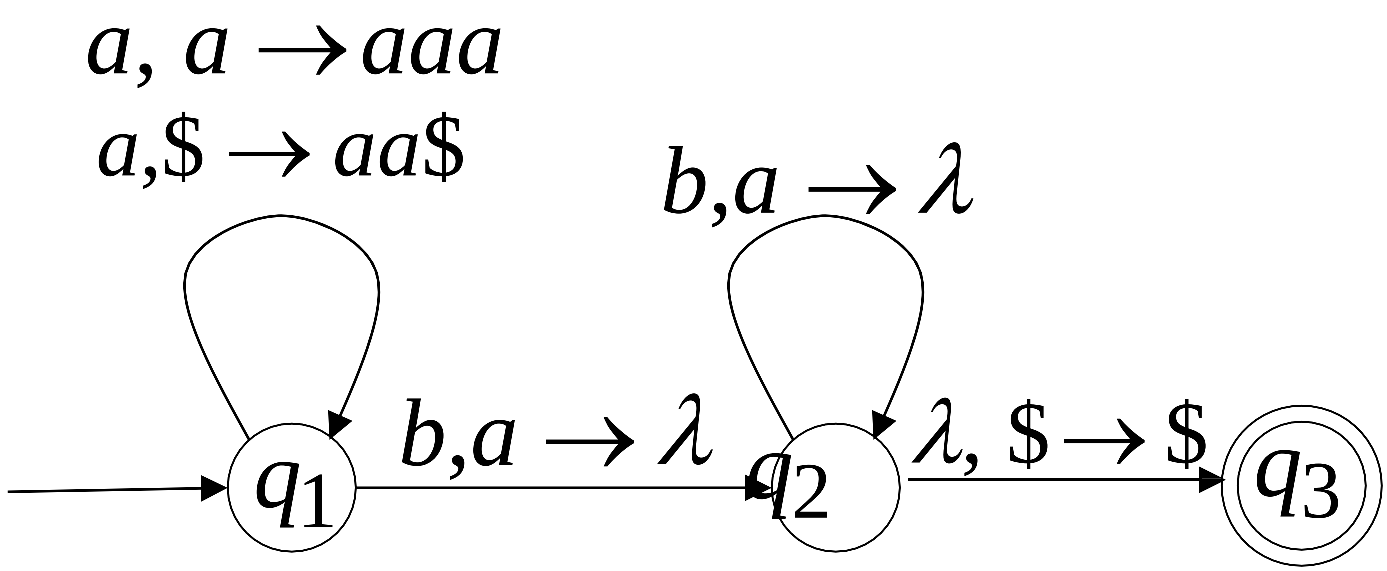
(Nov-2017 EndSem 8 Marks)



Example PDA

PDA M :L(M) such that $a^n b^{2n}$| n>=1

(Nov-2017 EndSem 8 Marks)



Another PDA example

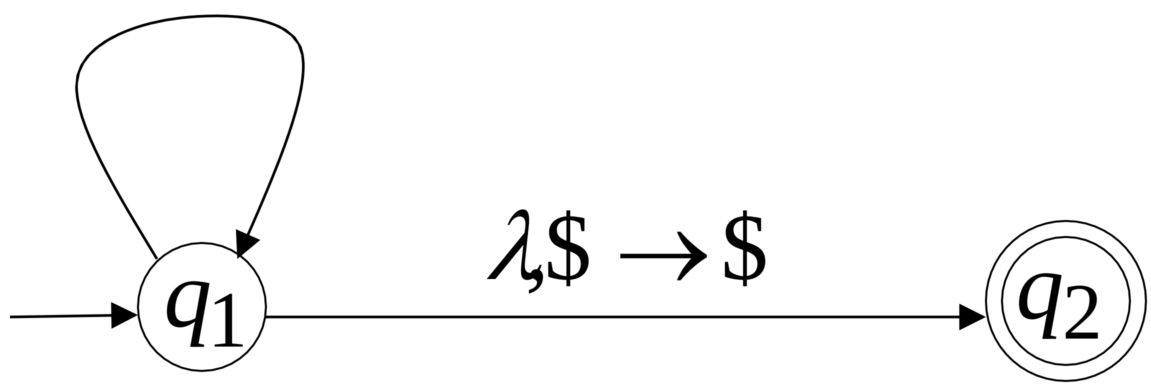
$$L(M) = \{w \in \{a,b\}^* : \quad n_a(w) = n_b(w)\}$$

PDA M

$a, \$ \rightarrow 0\$$ $b, \$ \rightarrow 1\$$

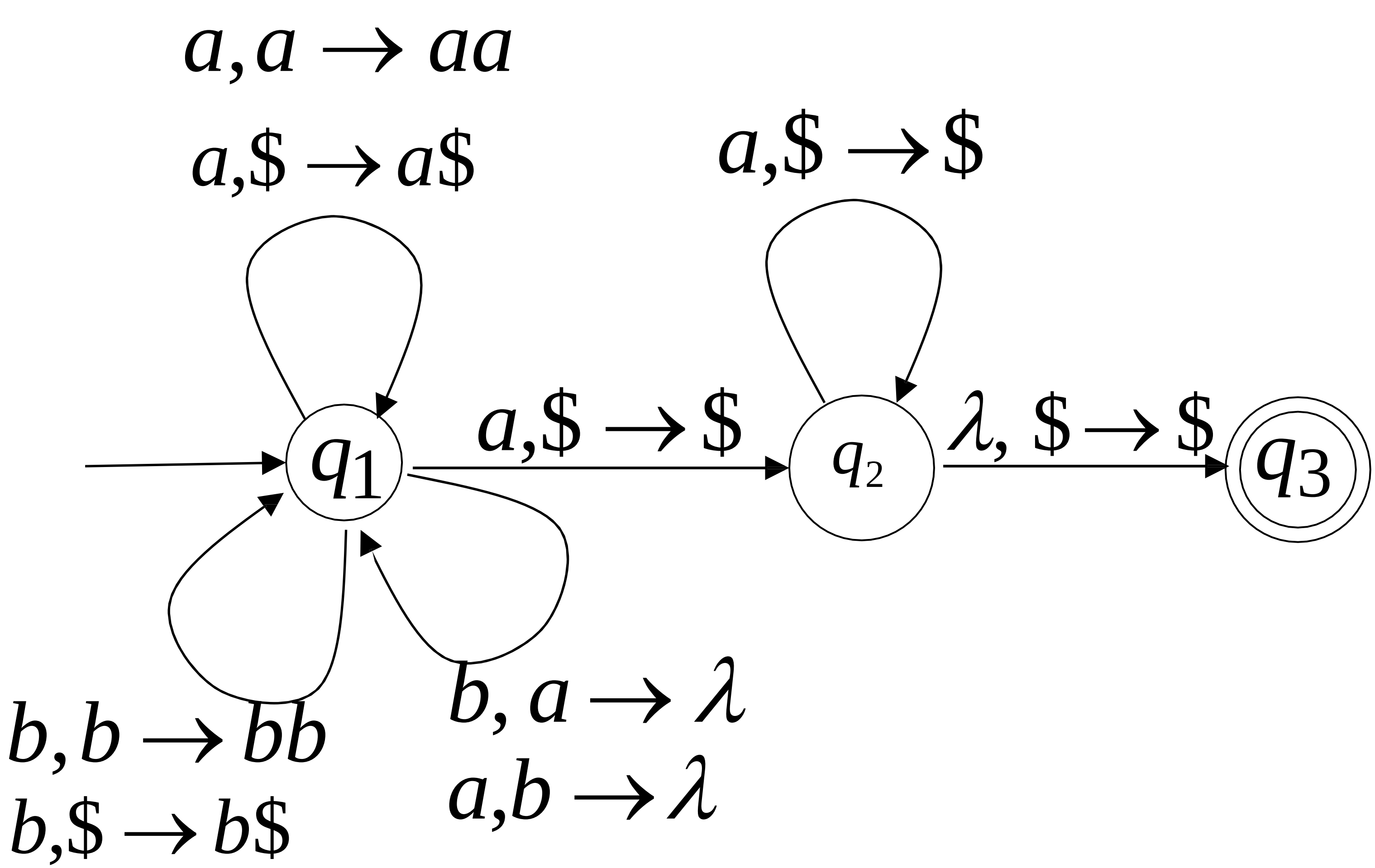
$a, 0 \rightarrow 00$ $b, 1 \rightarrow 11$

$a, 1 \rightarrow \lambda$ $b, 0 \rightarrow \lambda$



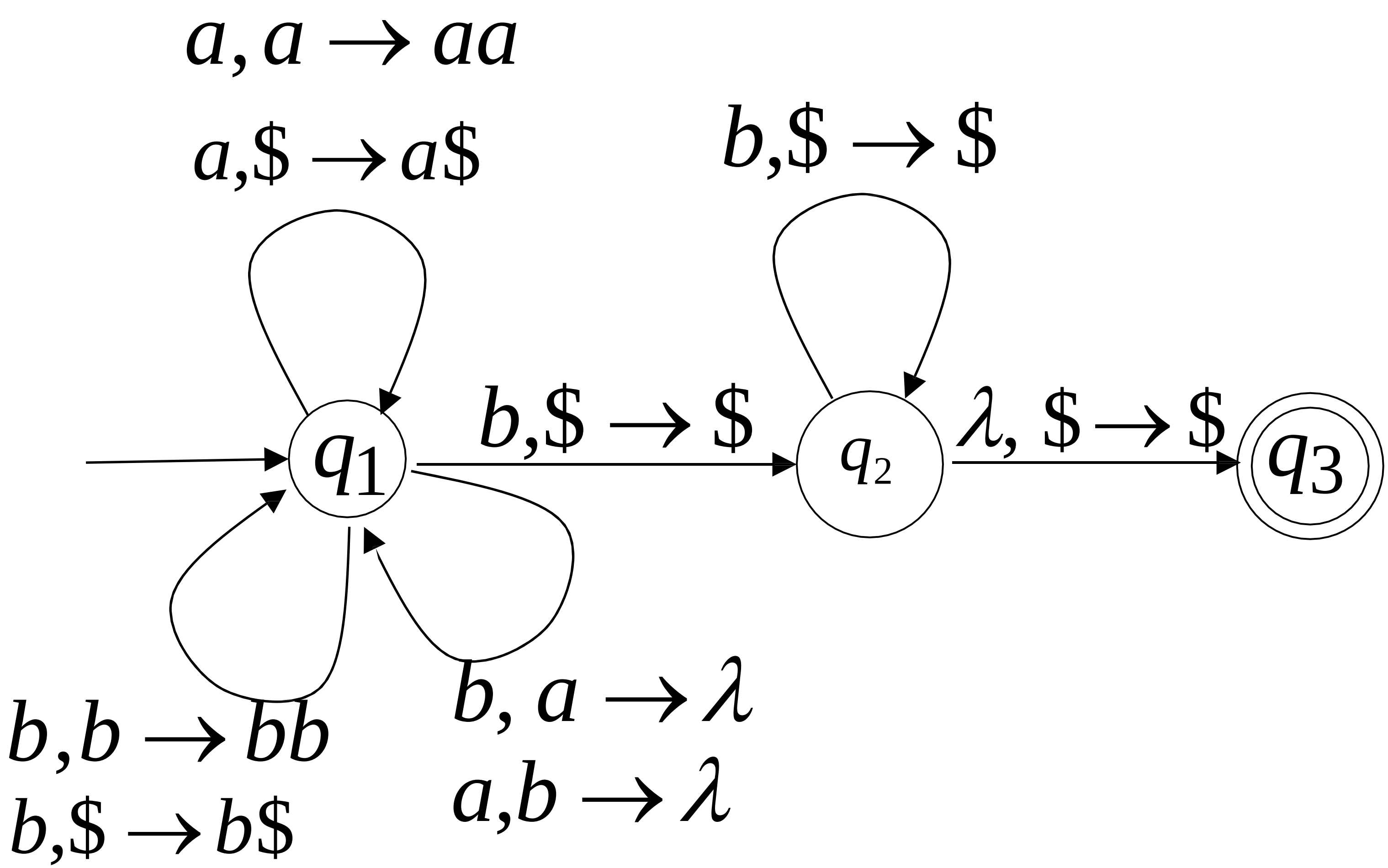
Example PDA

PDA M :L(M) such that $n_a(w)>n_b(w), w€(a,b)^*$



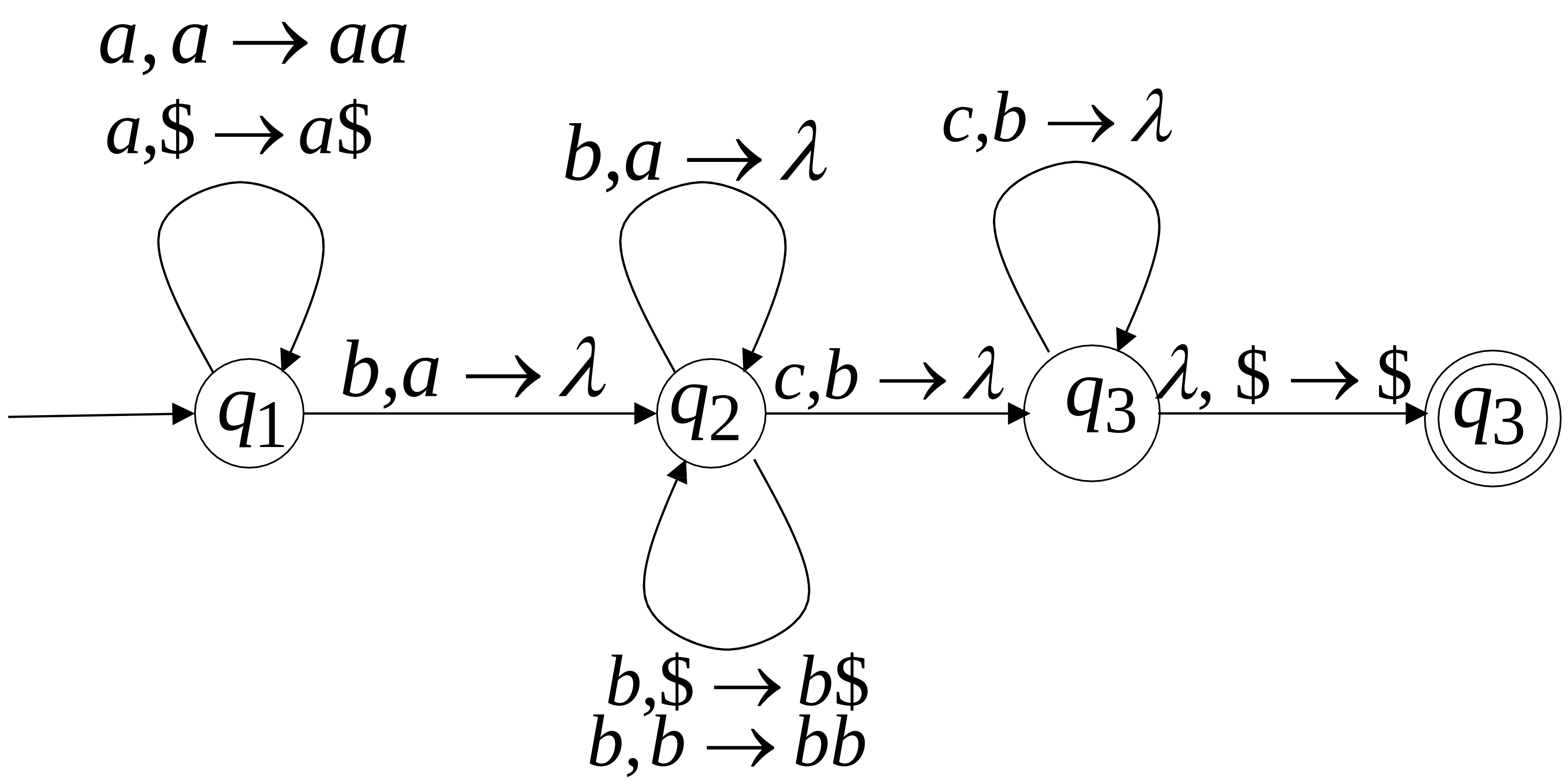
Example PDA

PDA M :L(M) such that $n_a(w)<n_b(w), w€(a,b)^*$



Example PDA

PDA M :L(M) such that $a^n b^{m+n} c^n$| n, m>=1



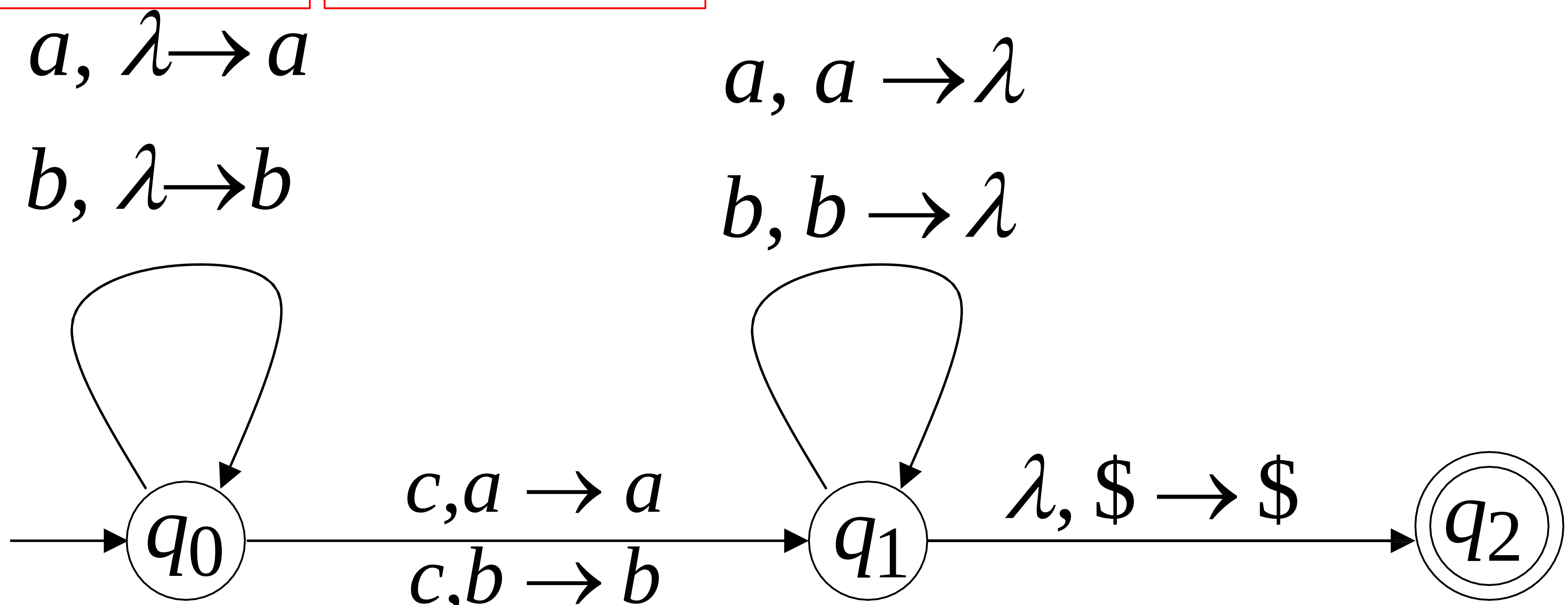
Review

- PDA
- Acceptance by Final State
- Acceptance by Empty Stack

Another PDA example

PDA M : $L(M) = \{vcv^R : v \in \{a,b\}^*\}$

$a,\$ \rightarrow a$	$b,\$ \rightarrow b$
$a,a \rightarrow aa$	$b,b \rightarrow bb$
$a,b \rightarrow ab$	$b,a \rightarrow ba$



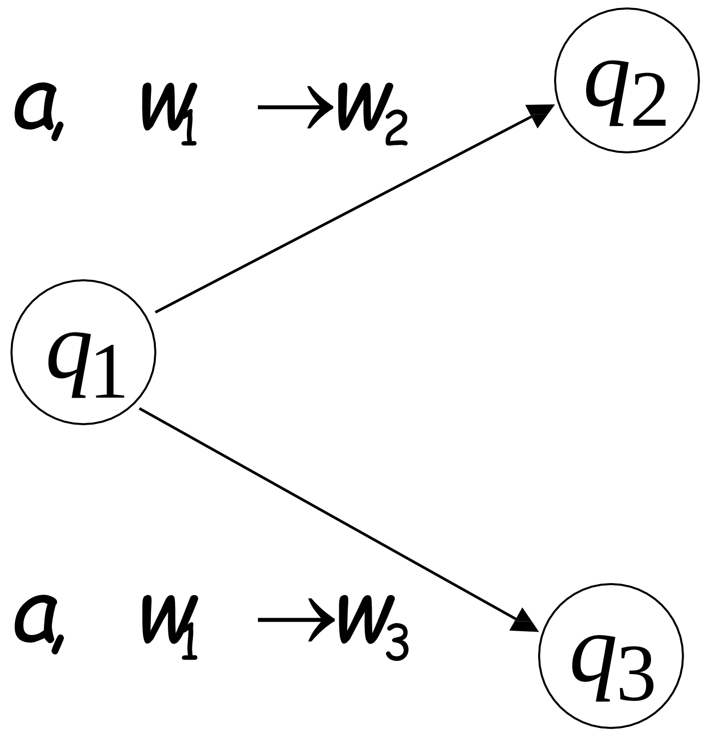
Construct the PDA that accepts a string of well formed parenthesis as (,),{,},[,]

Formalities for PDAs

$$q_1 \xrightarrow{a, w_1 \to w_2} q_2$$

Transition function:

$$\delta(q_1, a, w_1) = \{(q_2, w_2)\}$$



Transition function:

$$\delta(q_1, a, w_1) = \{(q_2, w_2), \qquad (q_3, w_3)\}$$

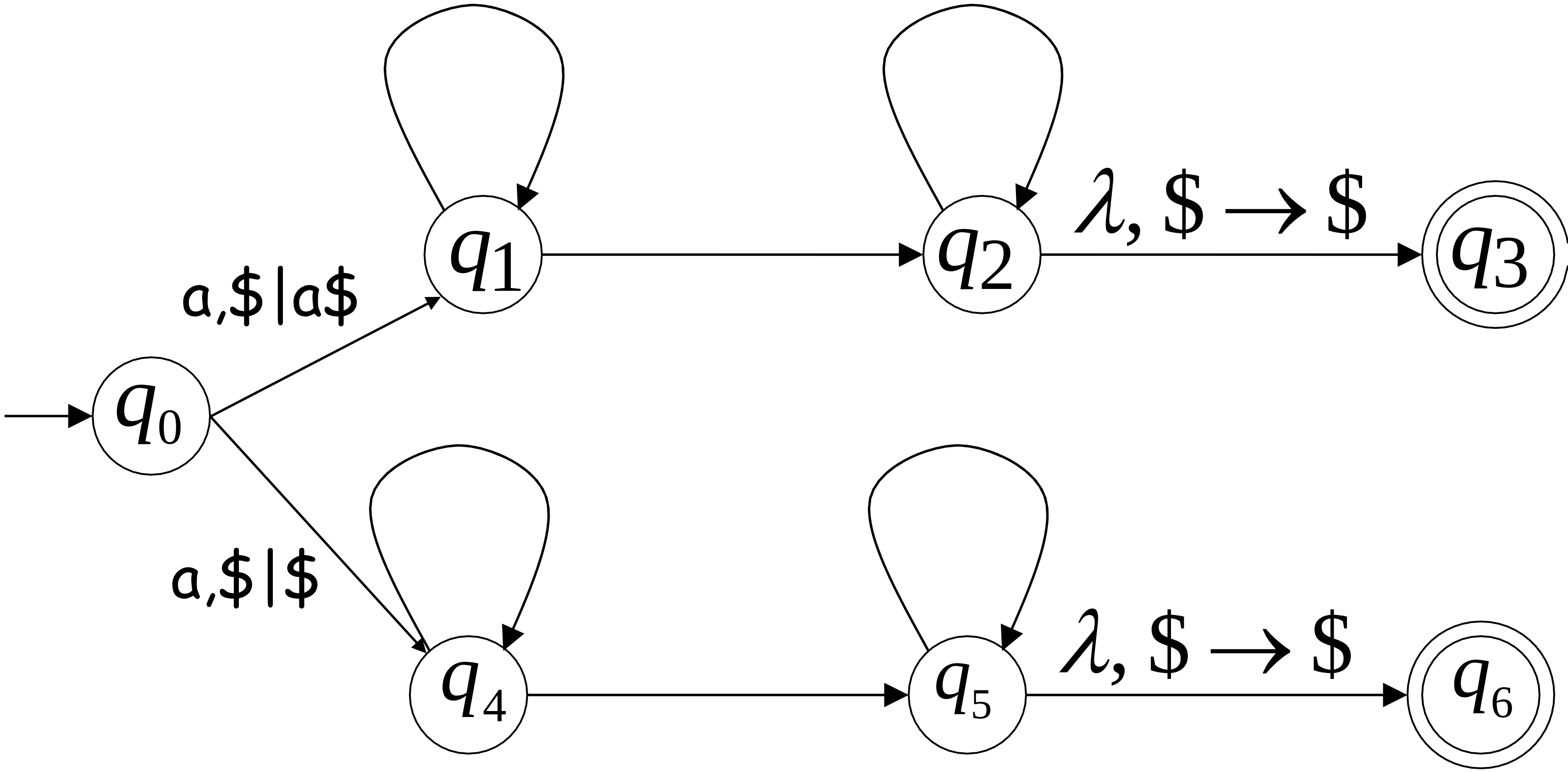
DPDA and NDPDA

- The main (and only) difference between DPDA and NPDA is that DPDAs are deterministic, whereas NPDAs are non-deterministic.
- With some of notation, we can say that NPDAs are a generalization of DPDAs: every DPDA can be simulated by an NPDA, but the converse doesn't hold (there are some context-free languages which cannot be accepted by a DPDA).
- Every DPDA contain equivalent NPDA,
- but, every NPDA may not contain equivalent DPDA sometimes.

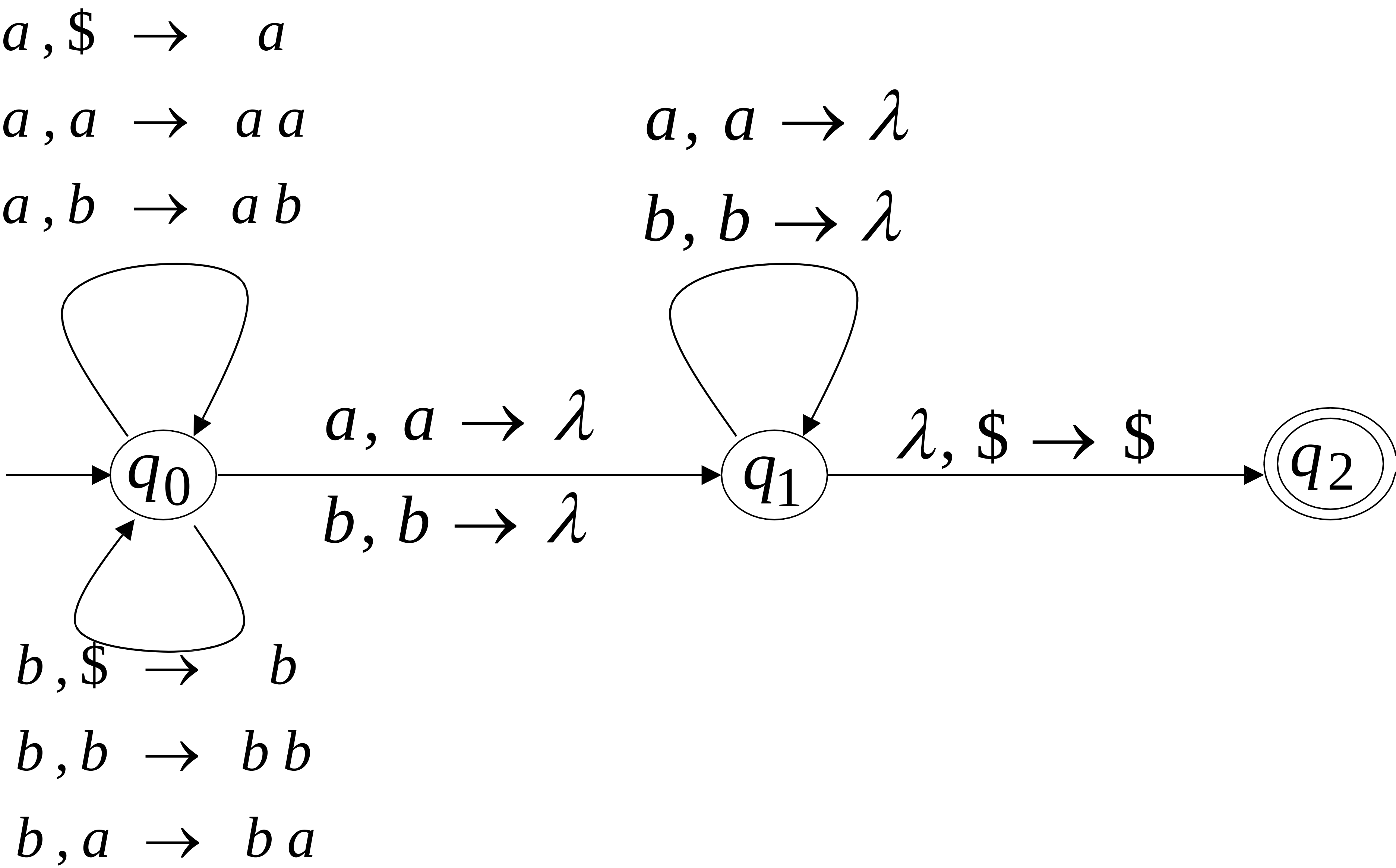
NPDA example

$$L(M)=\{w\in\{a,b\}^*: a^i b^j c^k d^m \mid i=k \text{ or } j=m\}$$

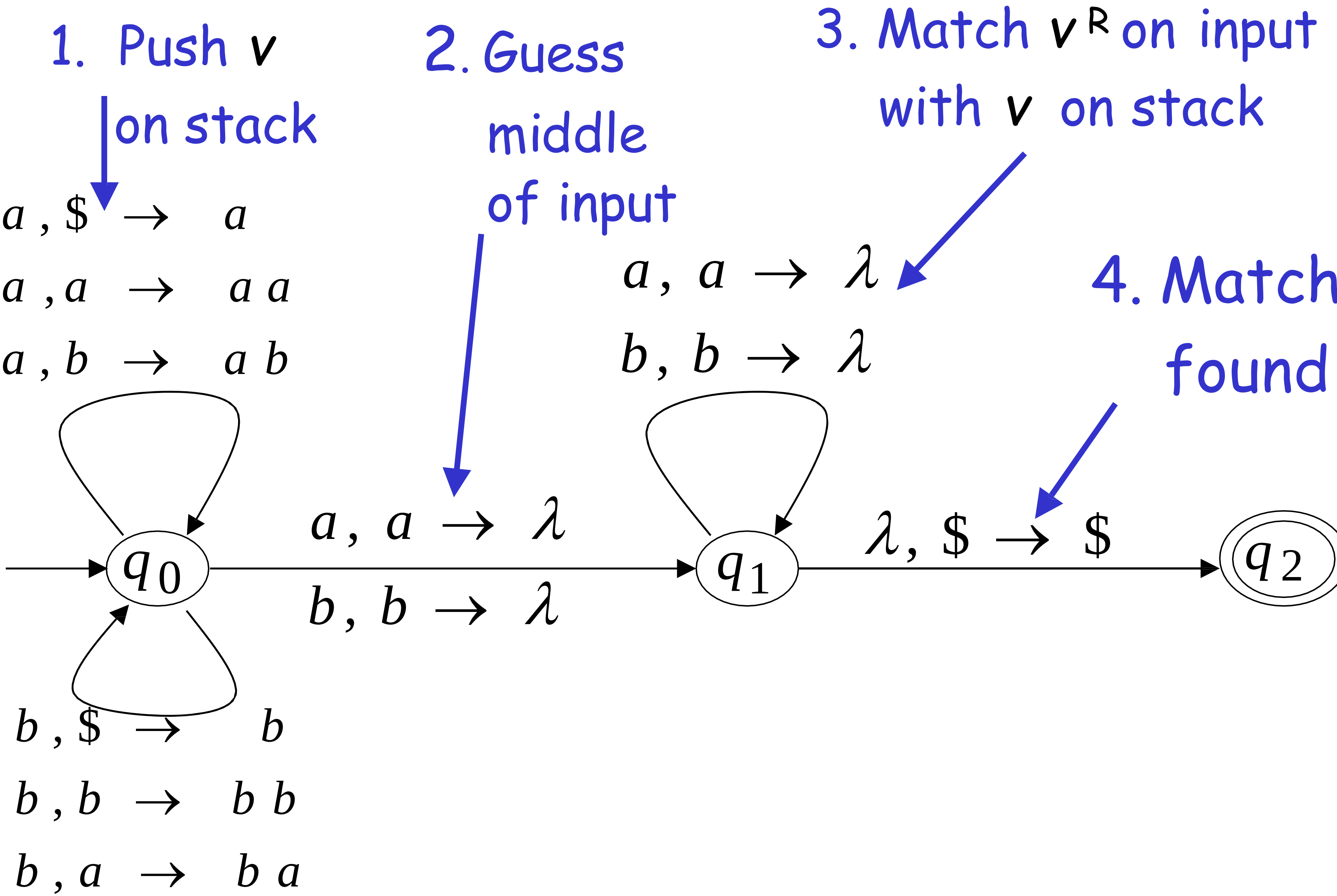
$$L(M)=\{a^n b^j c^n d^m\} \cup \{a^i b^n c^k d^n\}$$



PDA M : $L(M) = \{vv^R : v \in \{a,b\}^*\}$



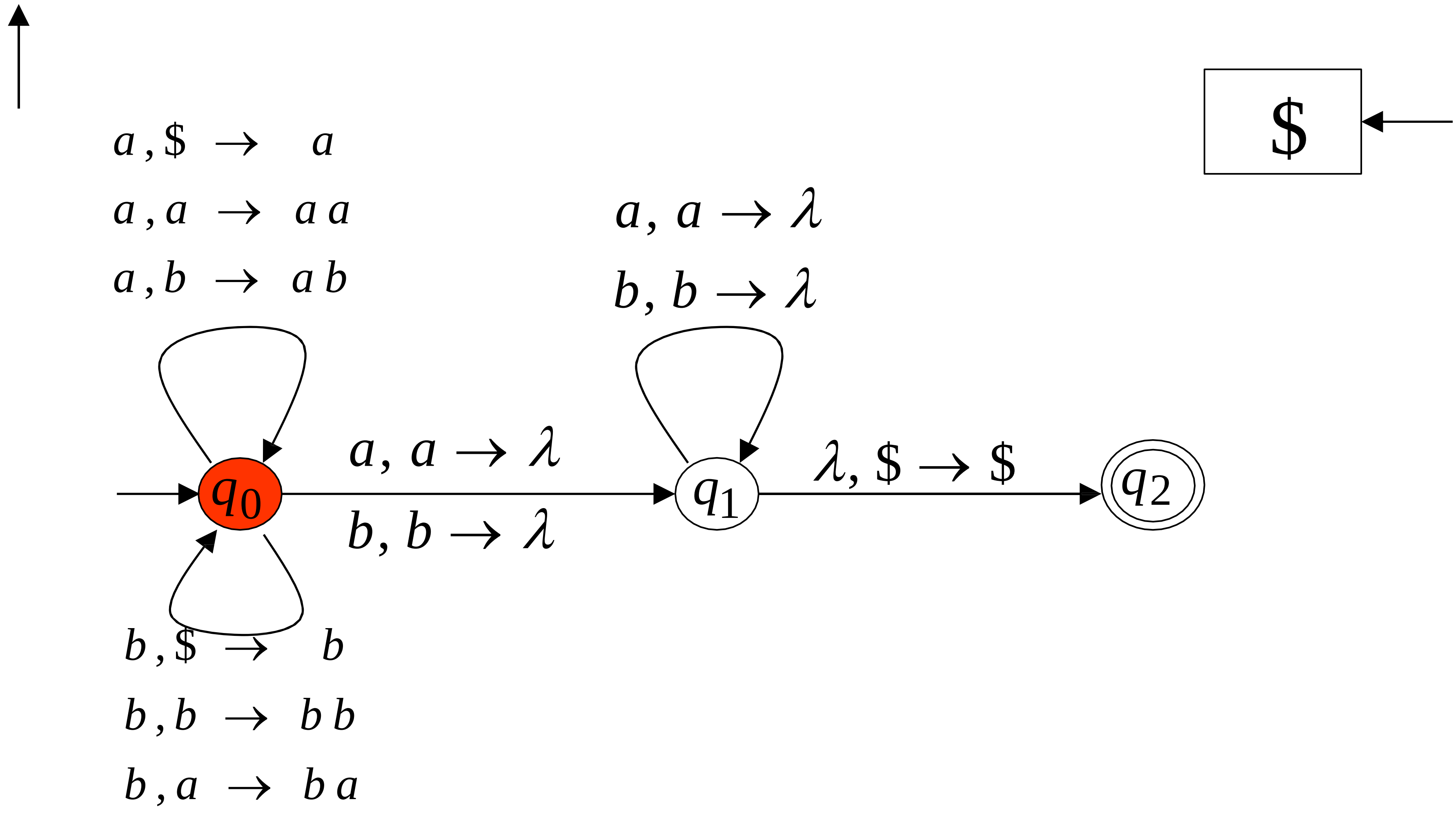
PDA M : $L(M) = \{vv^R : v \in \{a,b\}^*\}$



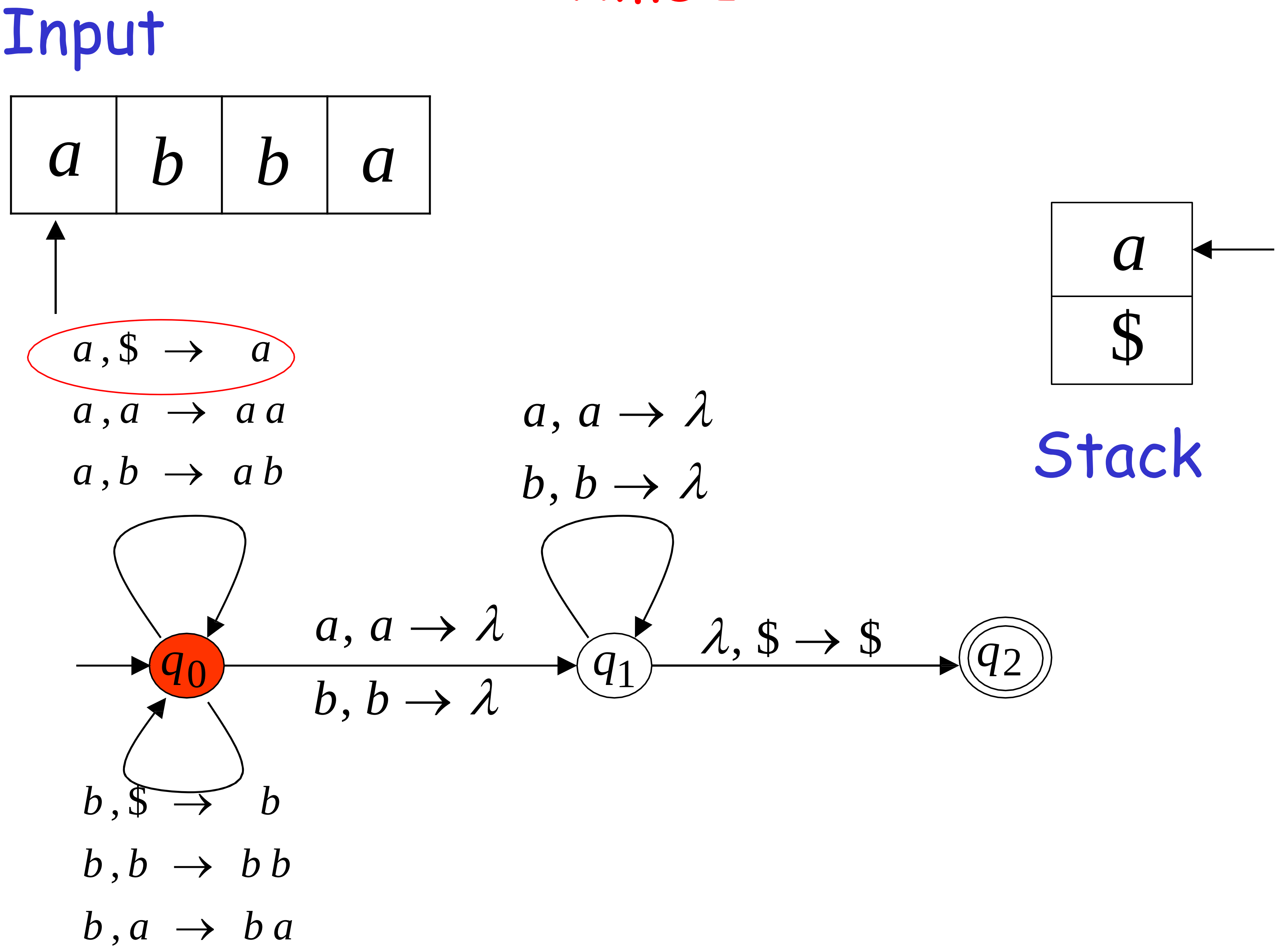
Execution Example: Time 0

Input

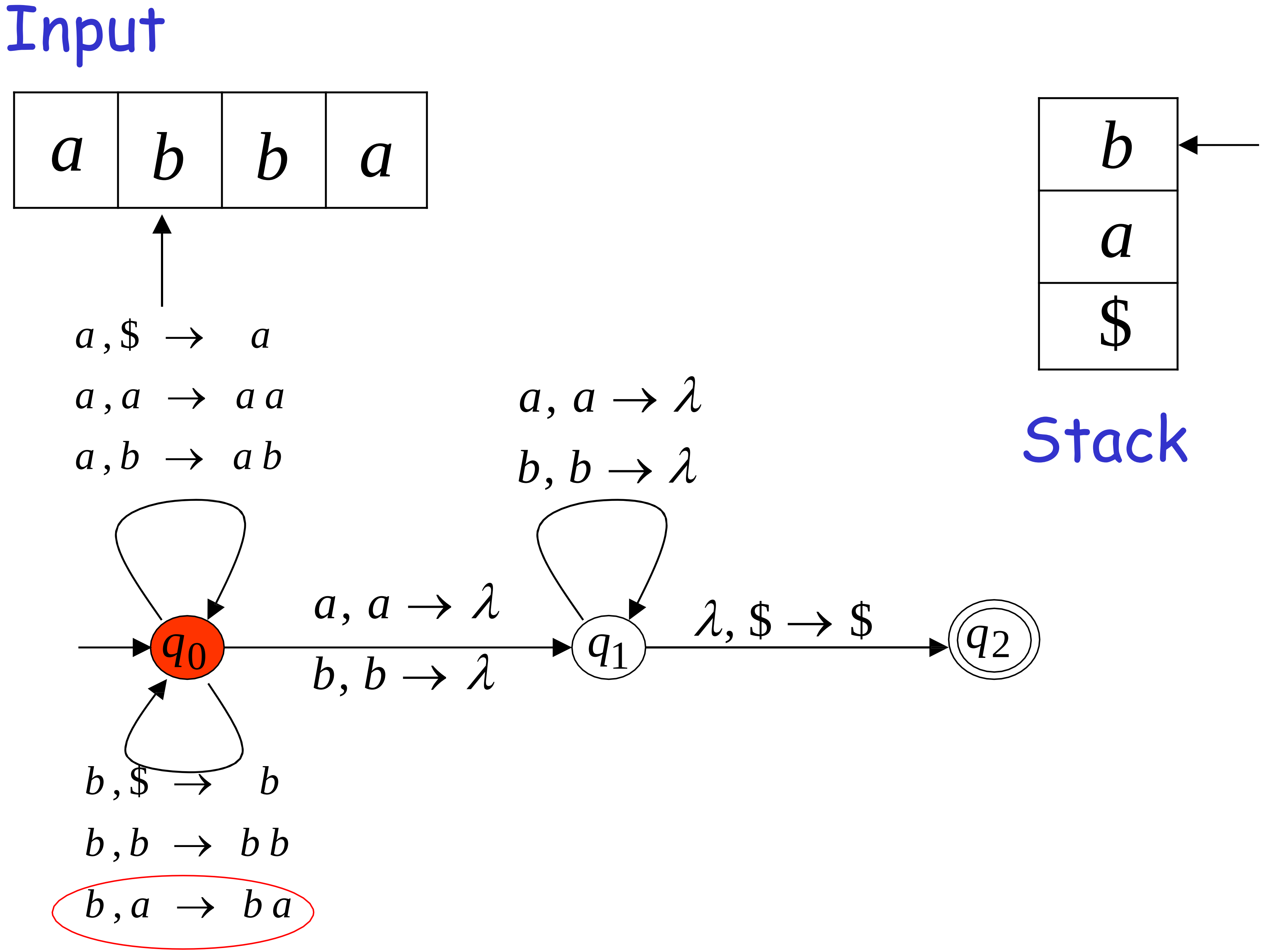
a	b	b	a



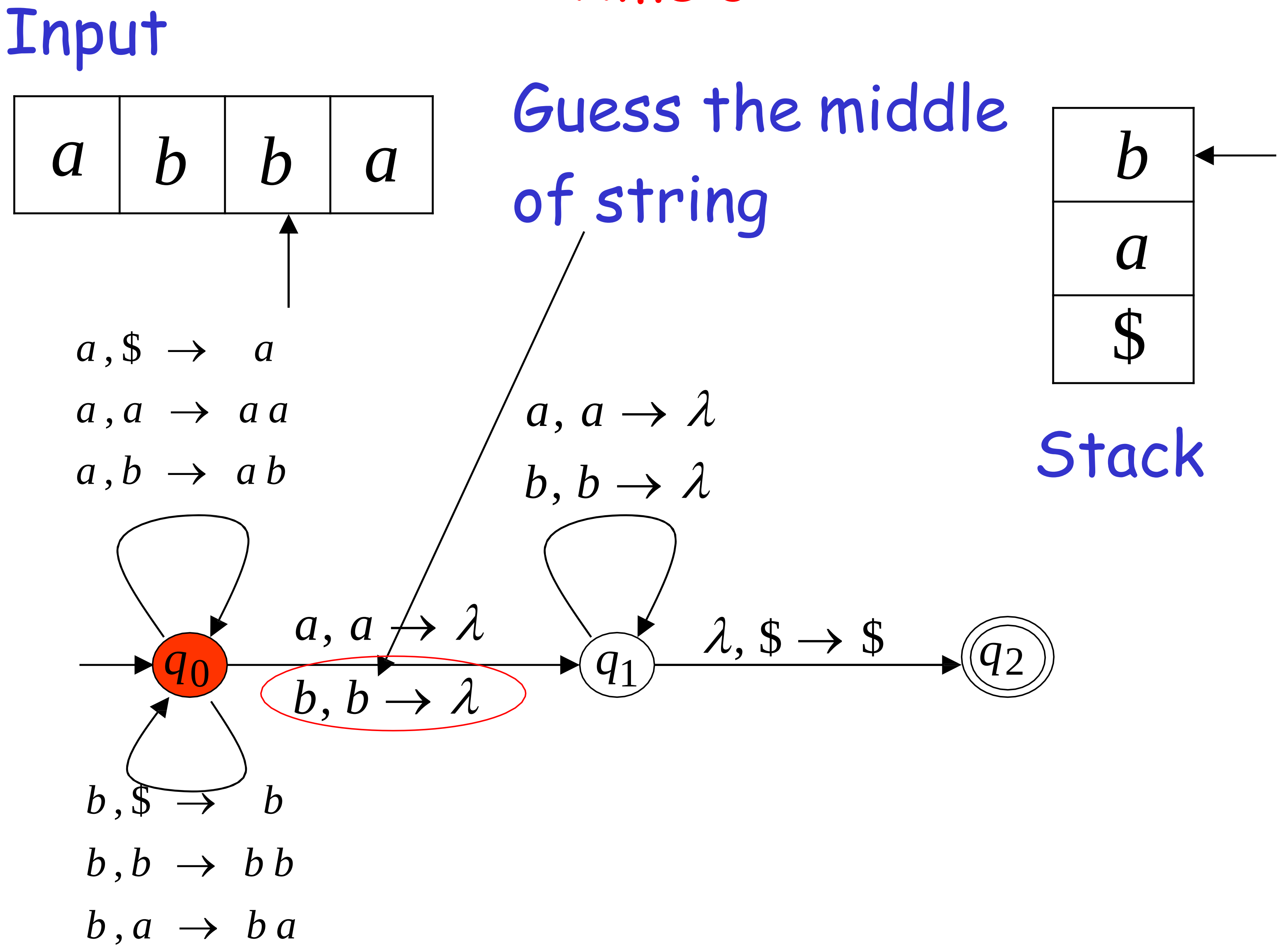
Time 1



Time 2

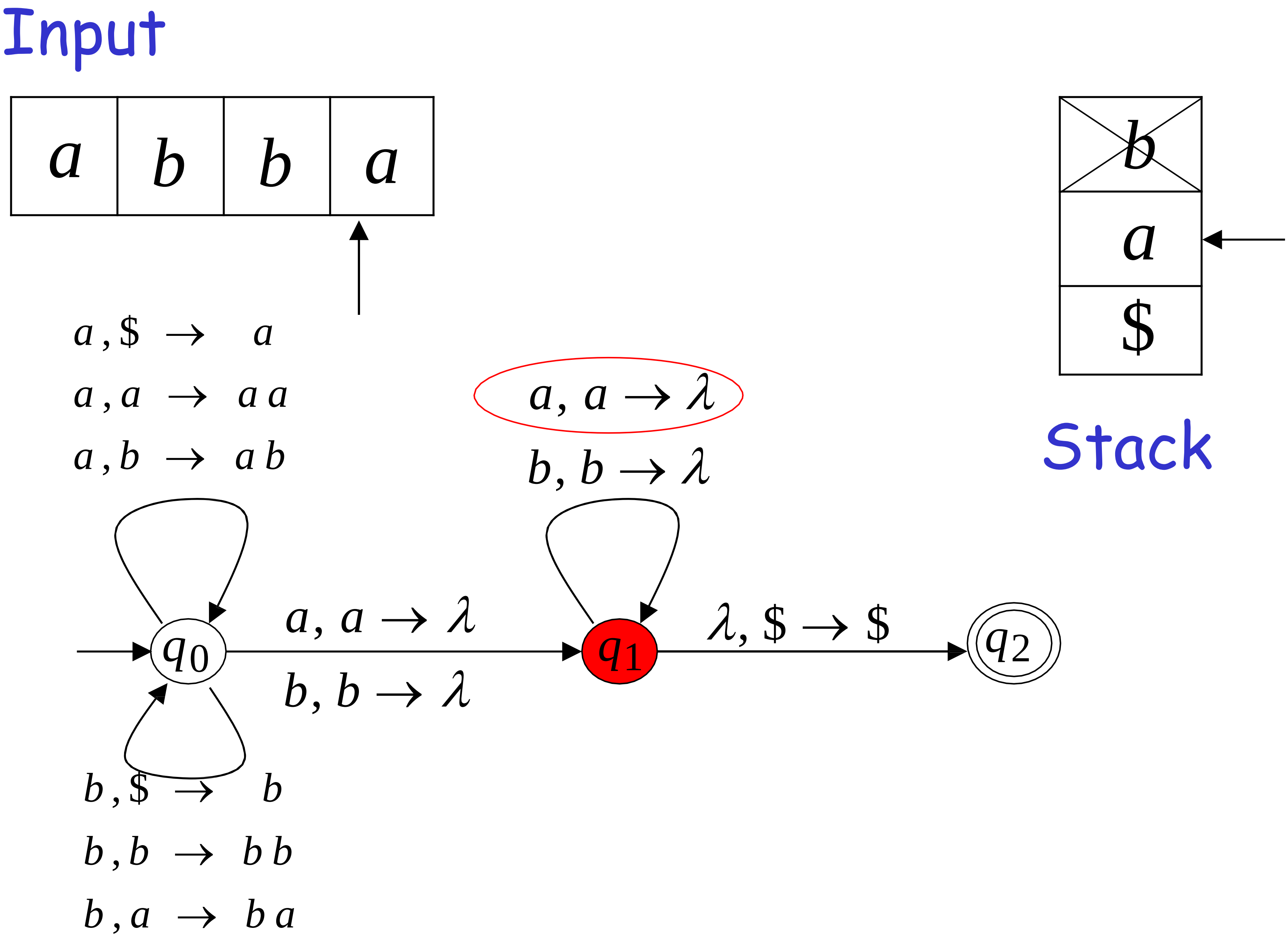

Input
a b b a
$a, \$ \rightarrow a$
$a, a \rightarrow aa$
$a, b \rightarrow ab$
$b, \$ \rightarrow b$
$b, b \rightarrow bb$
$b, a \rightarrow ba$
q_0
$a, a \rightarrow \lambda$
$b, b \rightarrow \lambda$
q_1
$a, a \rightarrow \lambda$
$b, b \rightarrow \lambda$
$\lambda, \$ \rightarrow \$$
q_2
b
a
$
Stack

Time 3

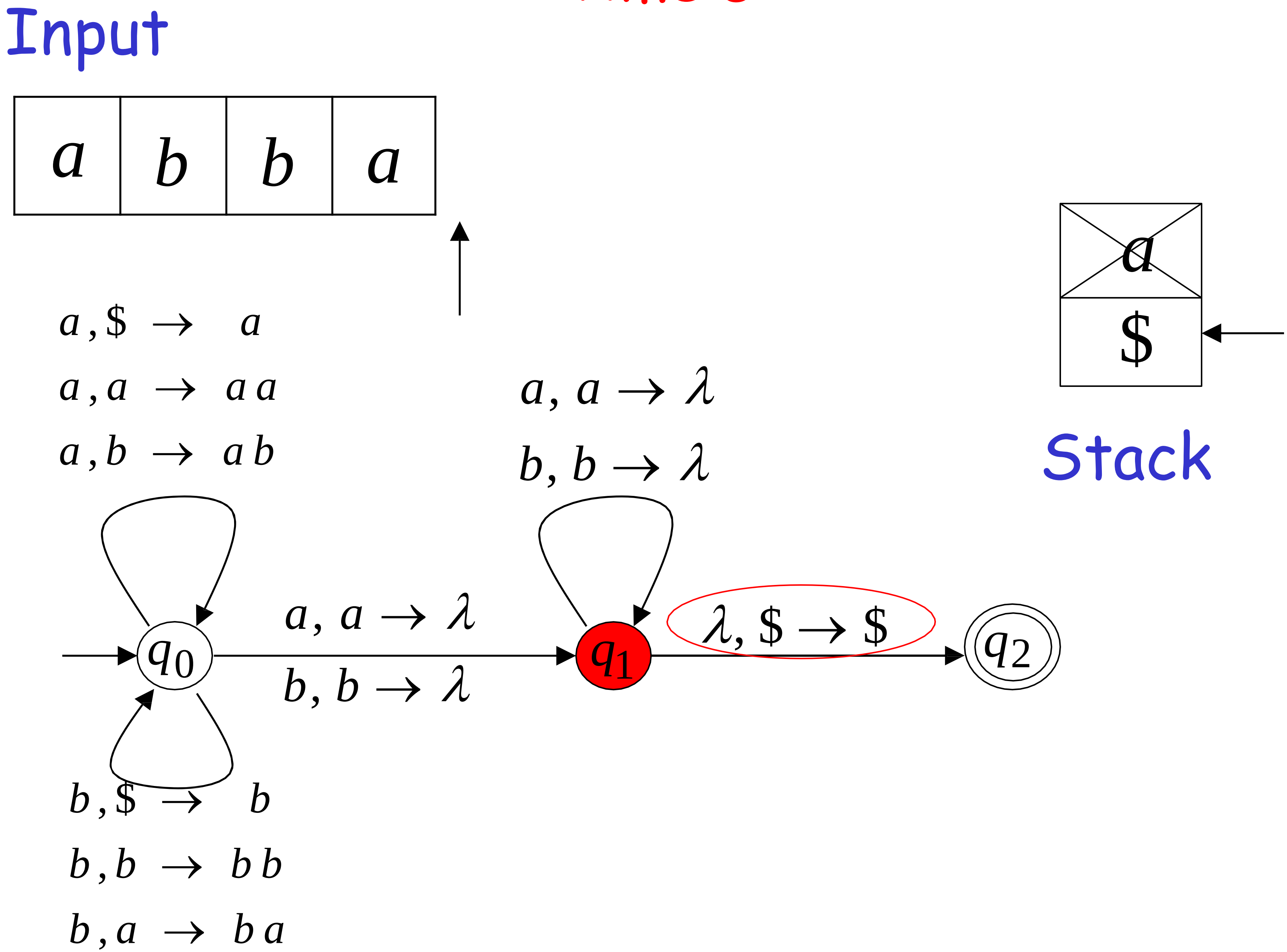


Input
a b b a
Guess the middle of string
b
a
$
Stack
a , $ → a
a , a → a a
a , b → a b
a, a → λ
b, b → λ
a, a → λ
b, b → λ
λ, $ → $
q0
q1
q2
b , $ → b
b , b → b b
b , a → b a

Time 4

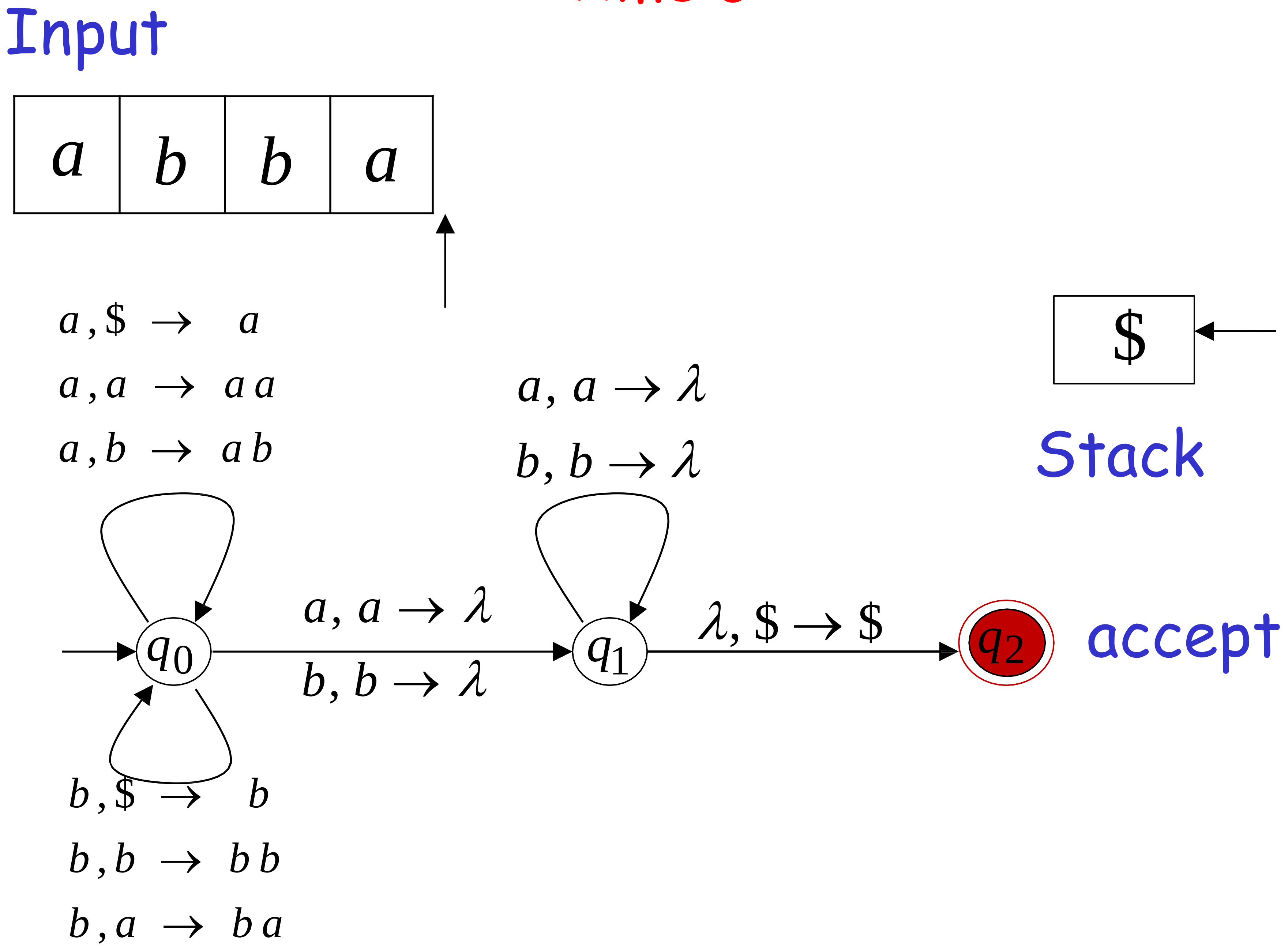

Input
a b b a
$\cancel{b}$
a
$
Stack
$a, \$ \to a$
$a, a \to aa$
$a, b \to ab$
$a, a \to \lambda$
$b, b \to \lambda$
$a, a \to \lambda$
$b, b \to \lambda$
$\lambda, \$ \to \$$
q_0
q_1
q_2
$b, \$ \to b$
$b, b \to bb$
$b, a \to ba$

Time 5



Input
a b b a
Stack
a
$
$a, \$ \rightarrow a$
$a, a \rightarrow aa$
$a, b \rightarrow ab$
$b, \$ \rightarrow b$
$b, b \rightarrow bb$
$b, a \rightarrow ba$
$a, a \rightarrow \lambda$
$b, b \rightarrow \lambda$
$a, a \rightarrow \lambda$
$b, b \rightarrow \lambda$
$\lambda, \$ \rightarrow \$$
q_0
q_1
q_2

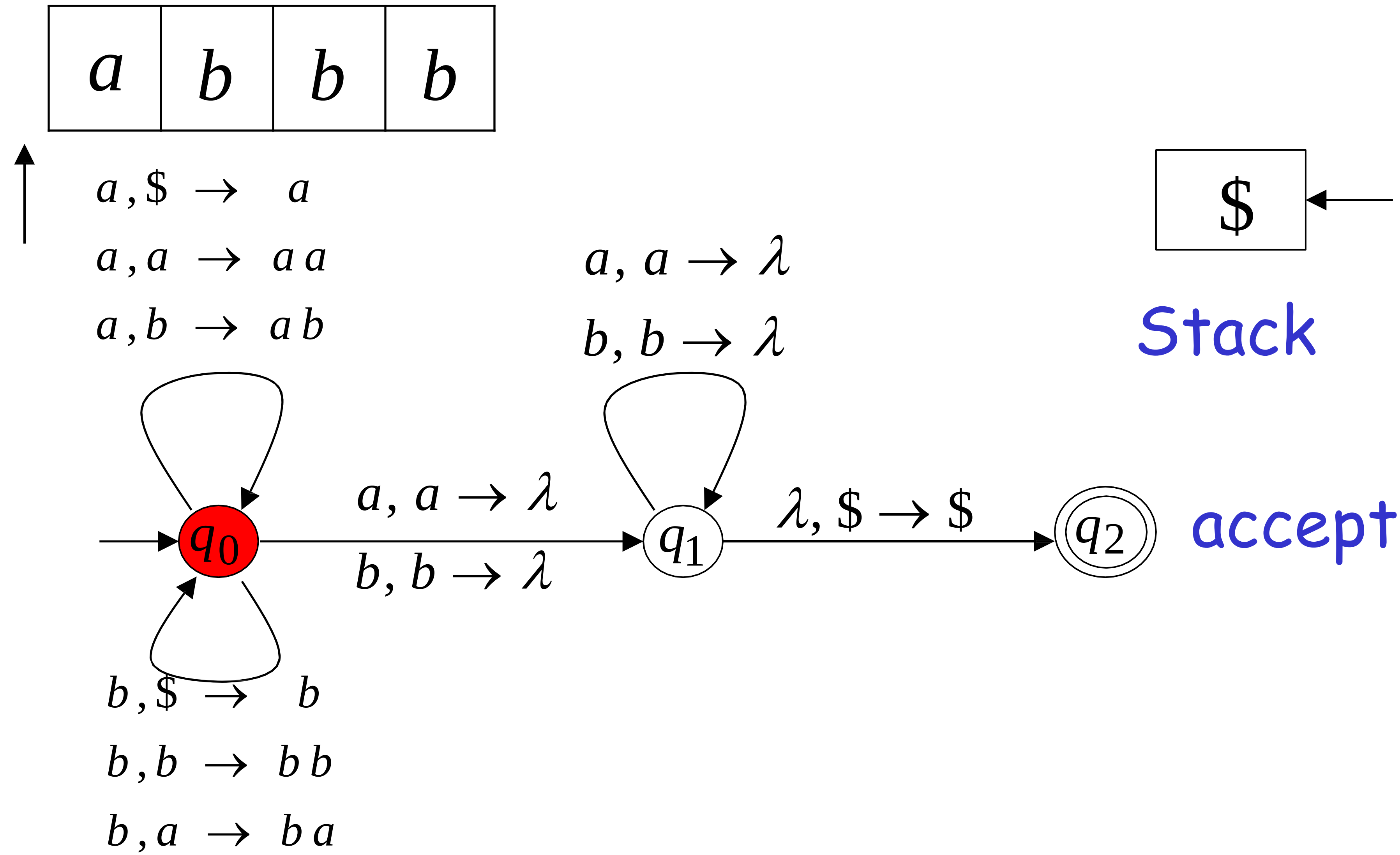
Time 6



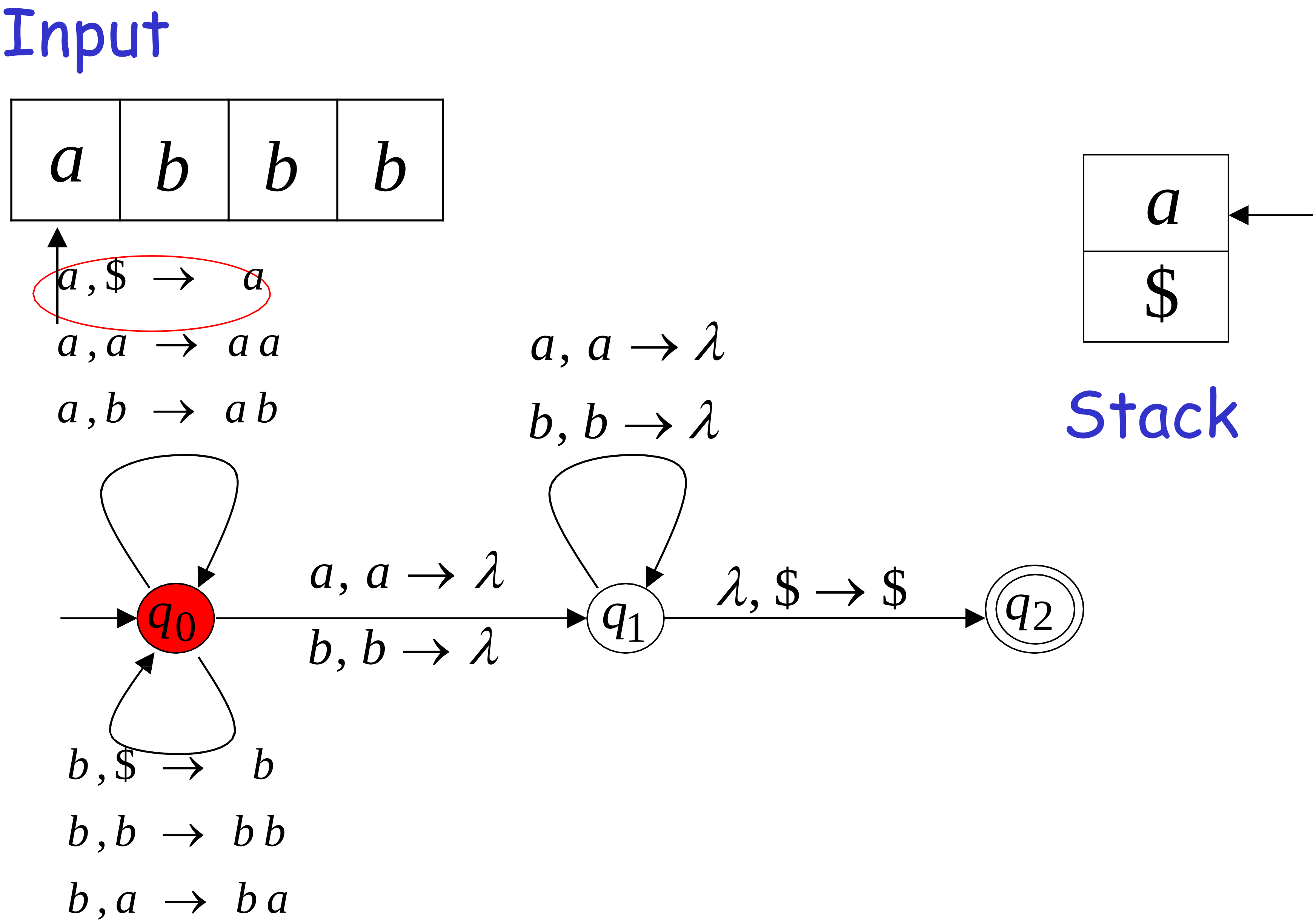
Input
a b b a
$a, \$ \rightarrow a$
$a, a \rightarrow aa$
$a, b \rightarrow ab$
$a, a \rightarrow \lambda$
$b, b \rightarrow \lambda$
$\$$
Stack
q_0
$a, a \rightarrow \lambda$
$b, b \rightarrow \lambda$
q_1
$\lambda, \$ \rightarrow \$$
q_2
accept
$b, \$ \rightarrow b$
$b, b \rightarrow bb$
$b, a \rightarrow ba$

Rejection Example: Time 0

Input

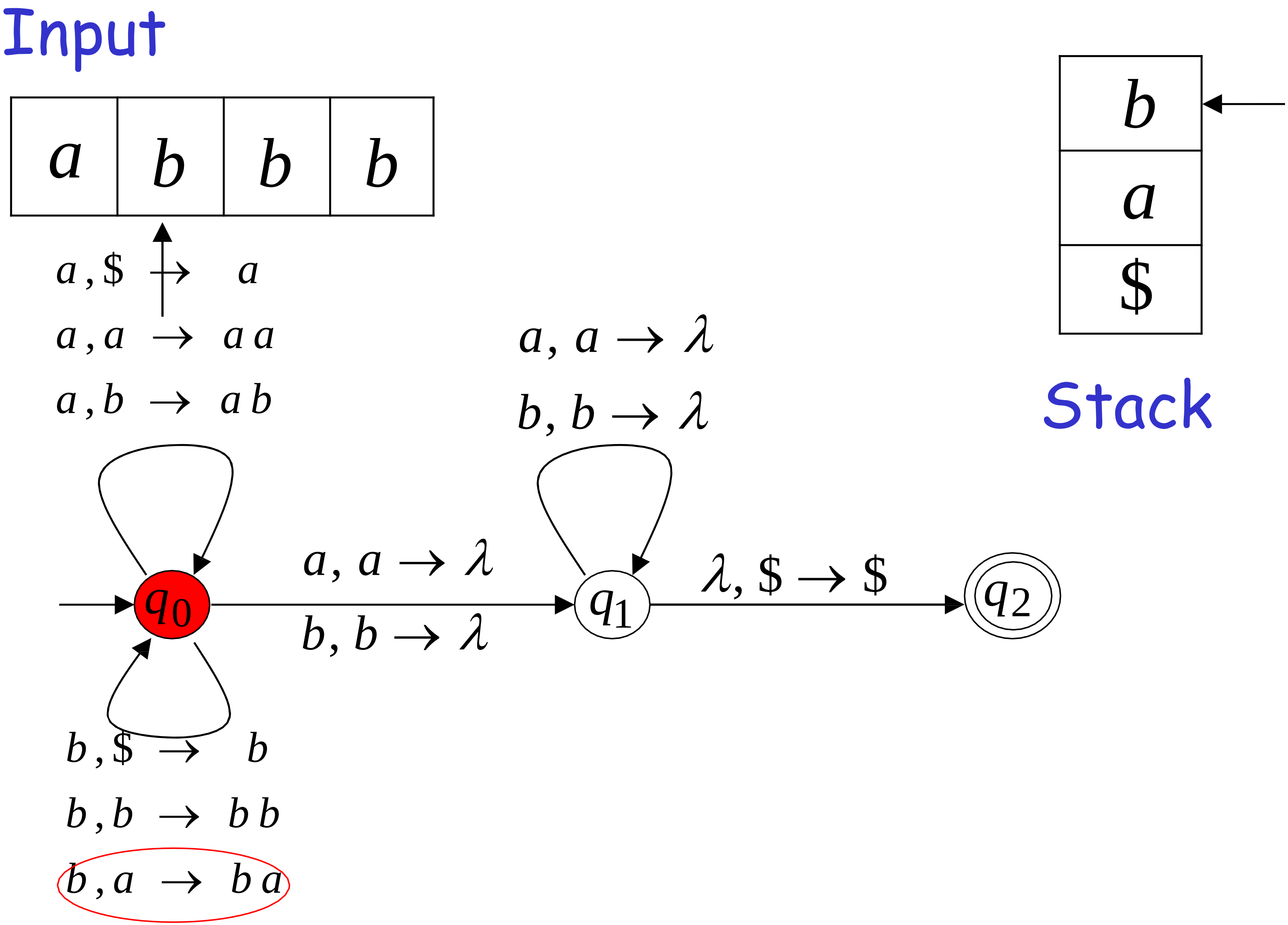


Time 1

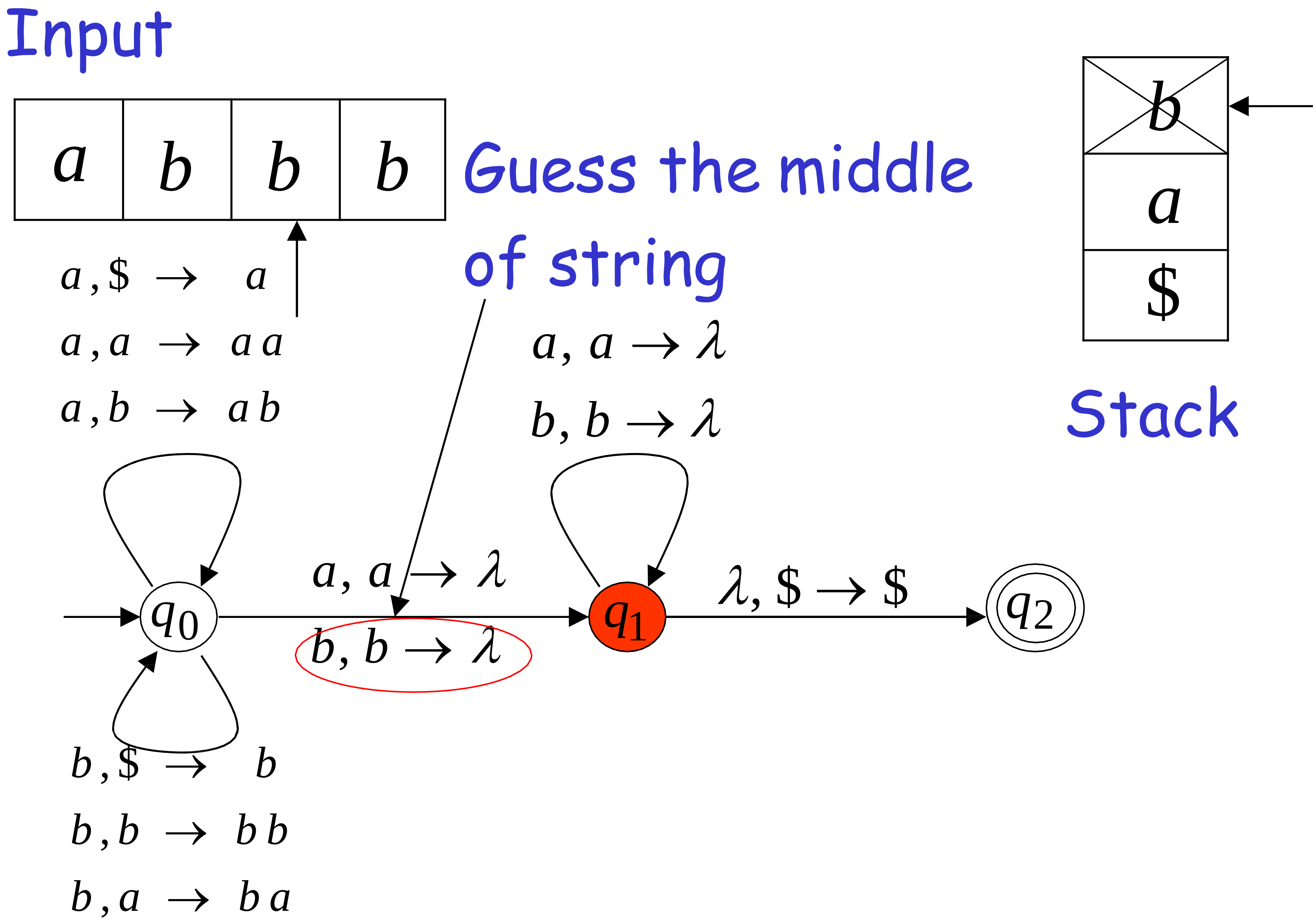


Input
a b b b
Stack
a
$
a , $ → a
a , a → a a
a , b → a b
b , $ → b
b , b → b b
b , a → b a
a, a → λ
b, b → λ
a, a → λ
b, b → λ
λ, $ → $
q0
q1
q2

Time 2


Input
a
b
b
b
b
a
$
Stack
a , $ → a
a , a → a a
a , b → a b
a, a → λ
b, b → λ
a, a → λ
b, b → λ
λ, $ → $
q0
q1
q2
b , $ → b
b , b → b b
b , a → b a

Time 3



Input
a b b b
Guess the middle of string
Stack
b
a
$
$a, \$ \rightarrow a$
$a, a \rightarrow aa$
$a, b \rightarrow ab$
$b, \$ \rightarrow b$
$b, b \rightarrow bb$
$b, a \rightarrow ba$
$a, a \rightarrow \lambda$
$b, b \rightarrow \lambda$
$a, a \rightarrow \lambda$
$b, b \rightarrow \lambda$
$\lambda, \$ \rightarrow \$$
q_0
q_1
q_2

Time 4

There is no possible transition.

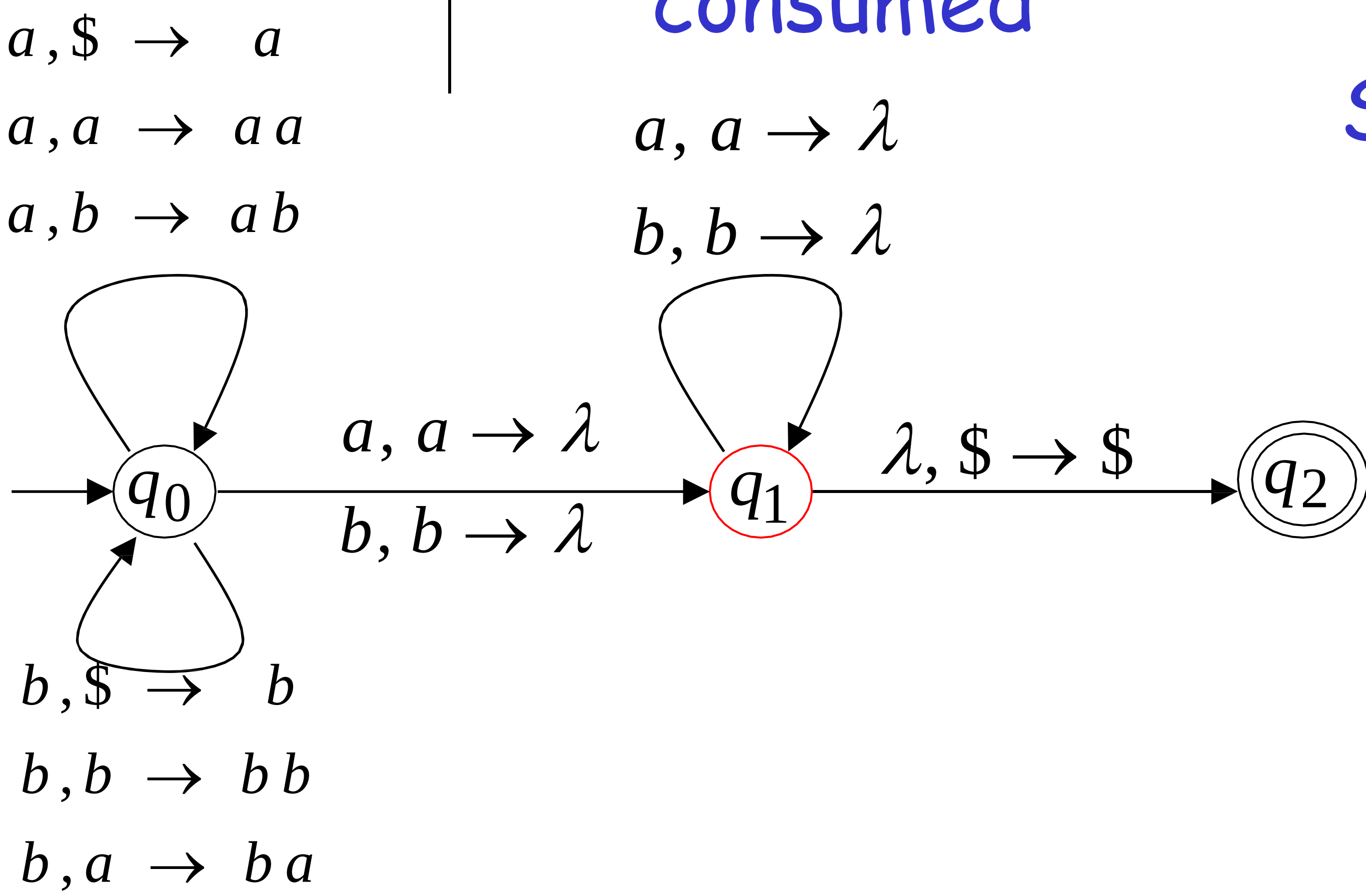
Input

a	b	b	b

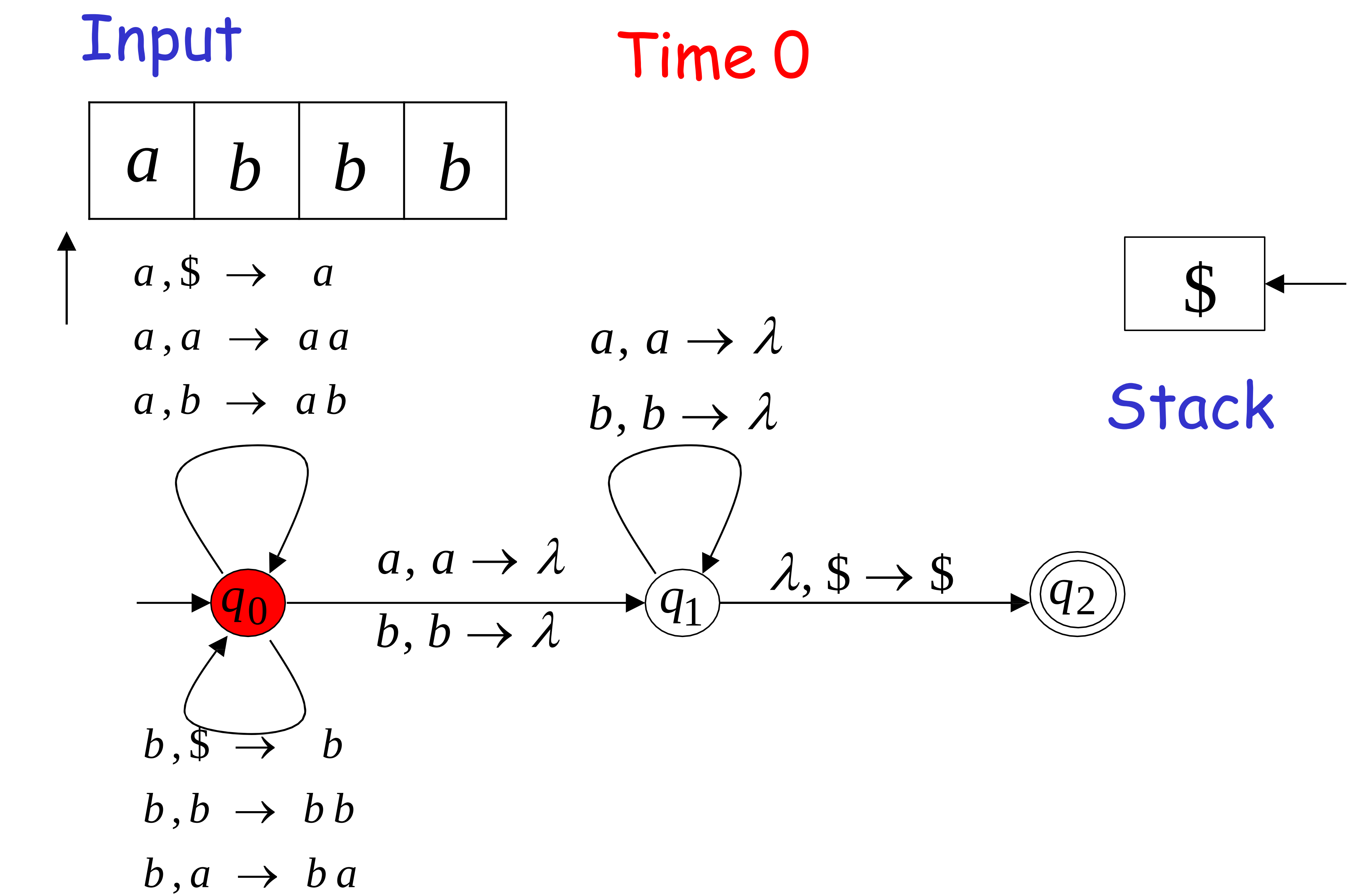
Input is not consumed

a
$\$$

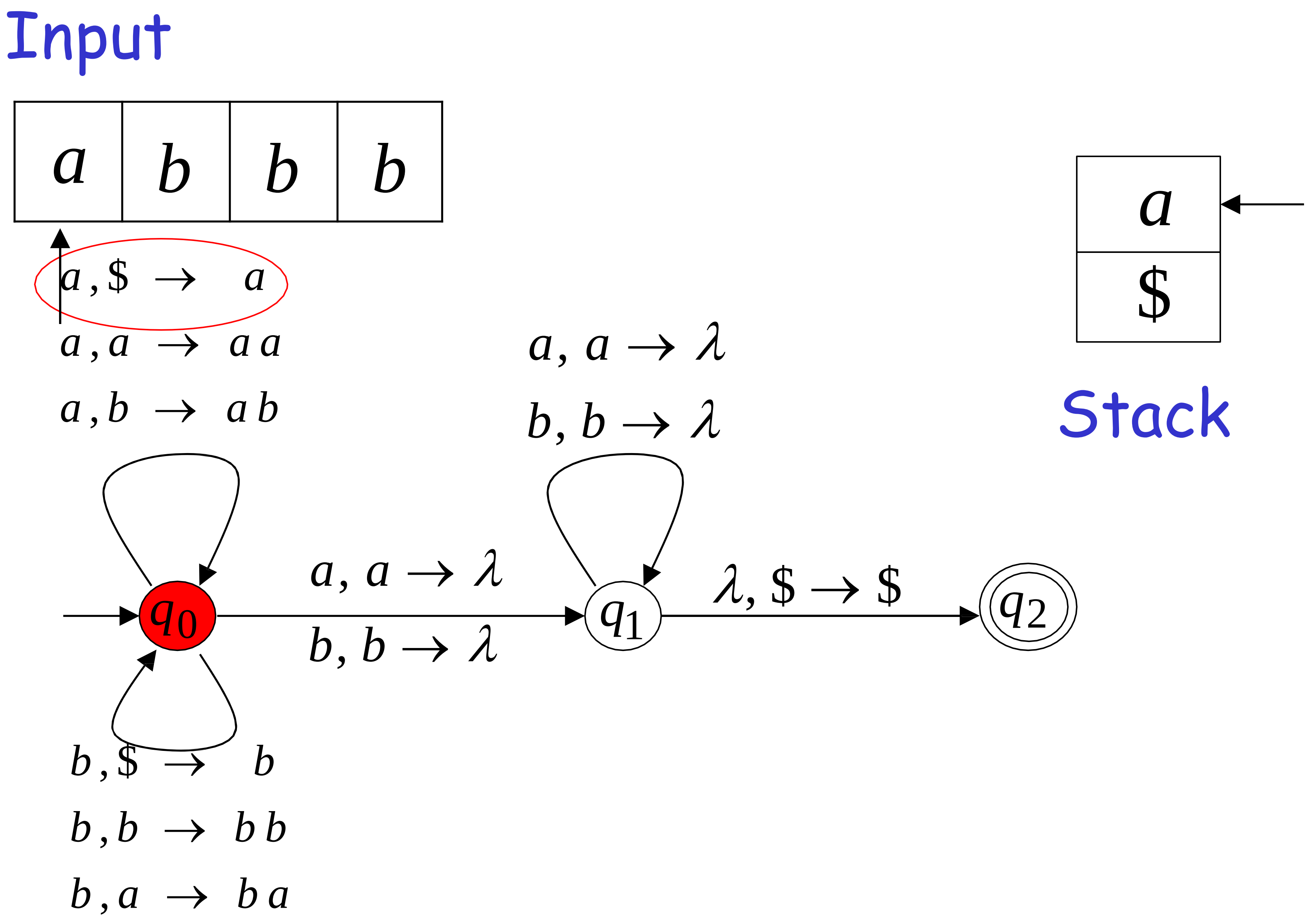
Stack



Another computation on same string:

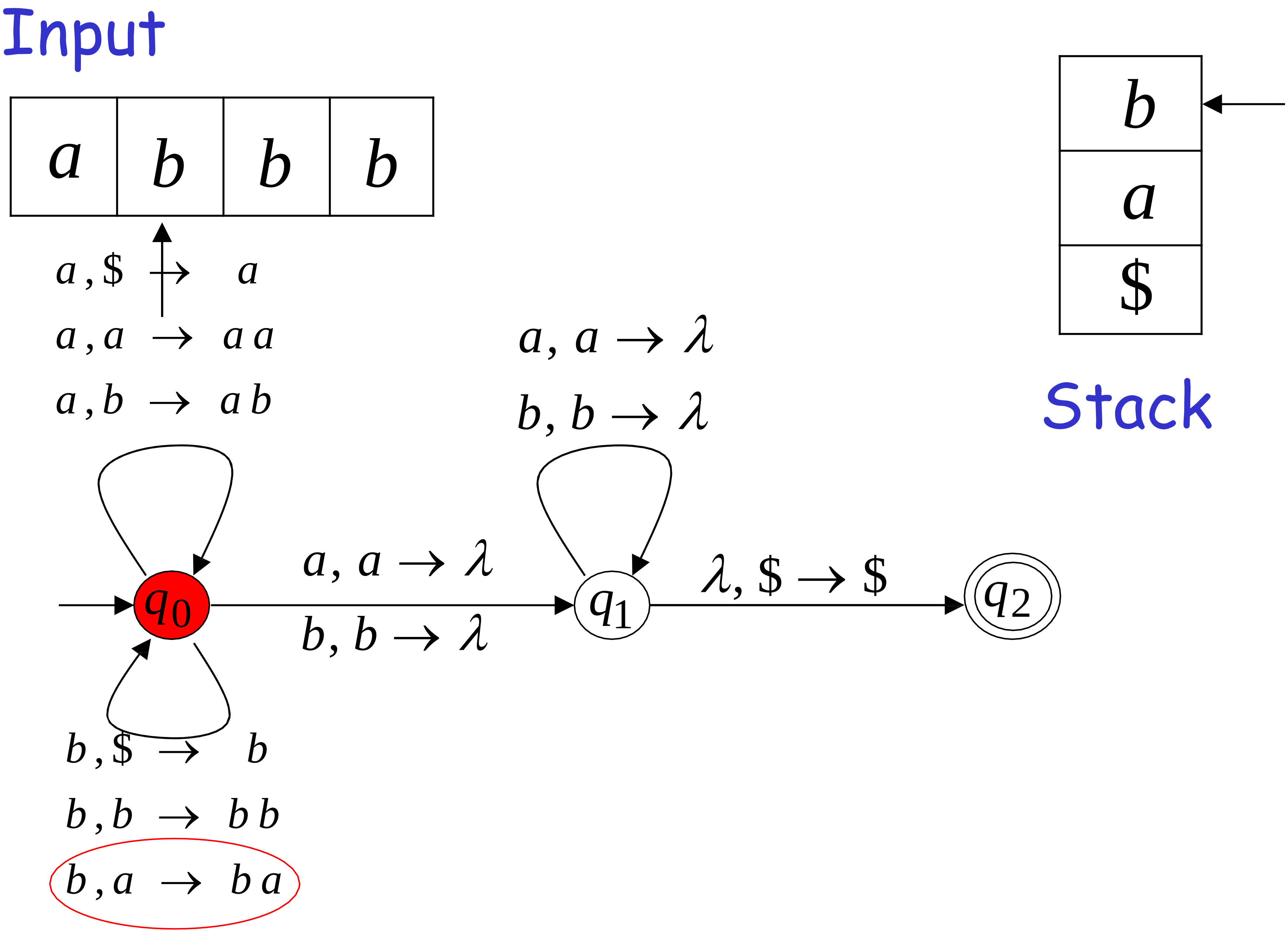


Time 1



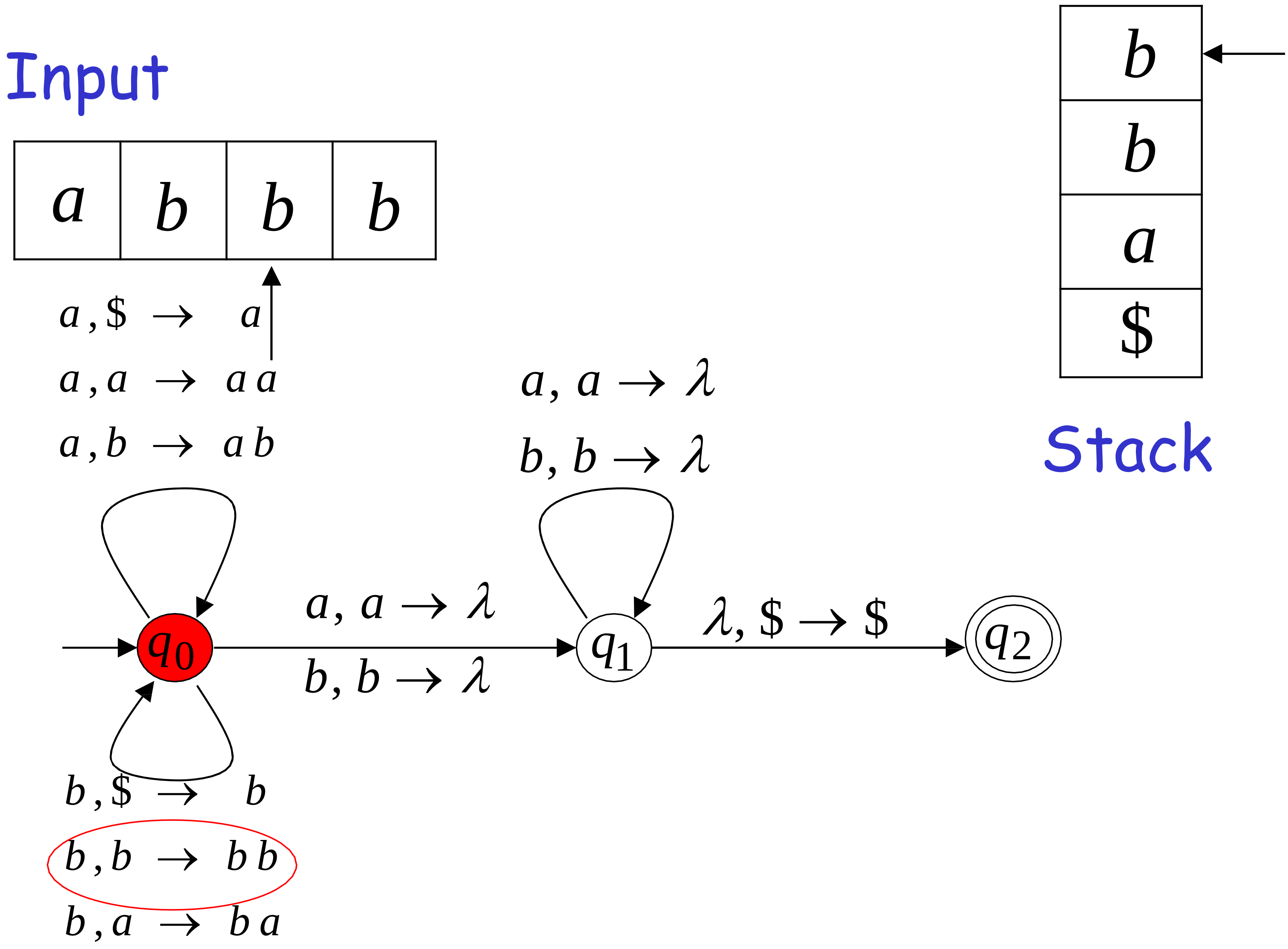
Input
a b b b
a , $ → a
a , a → a a
a , b → a b
a, a → λ
b, b → λ
Stack
a
$
q0
a, a → λ
b, b → λ
q1
λ, $ → $
q2
b , $ → b
b , b → b b
b , a → b a

Time 2

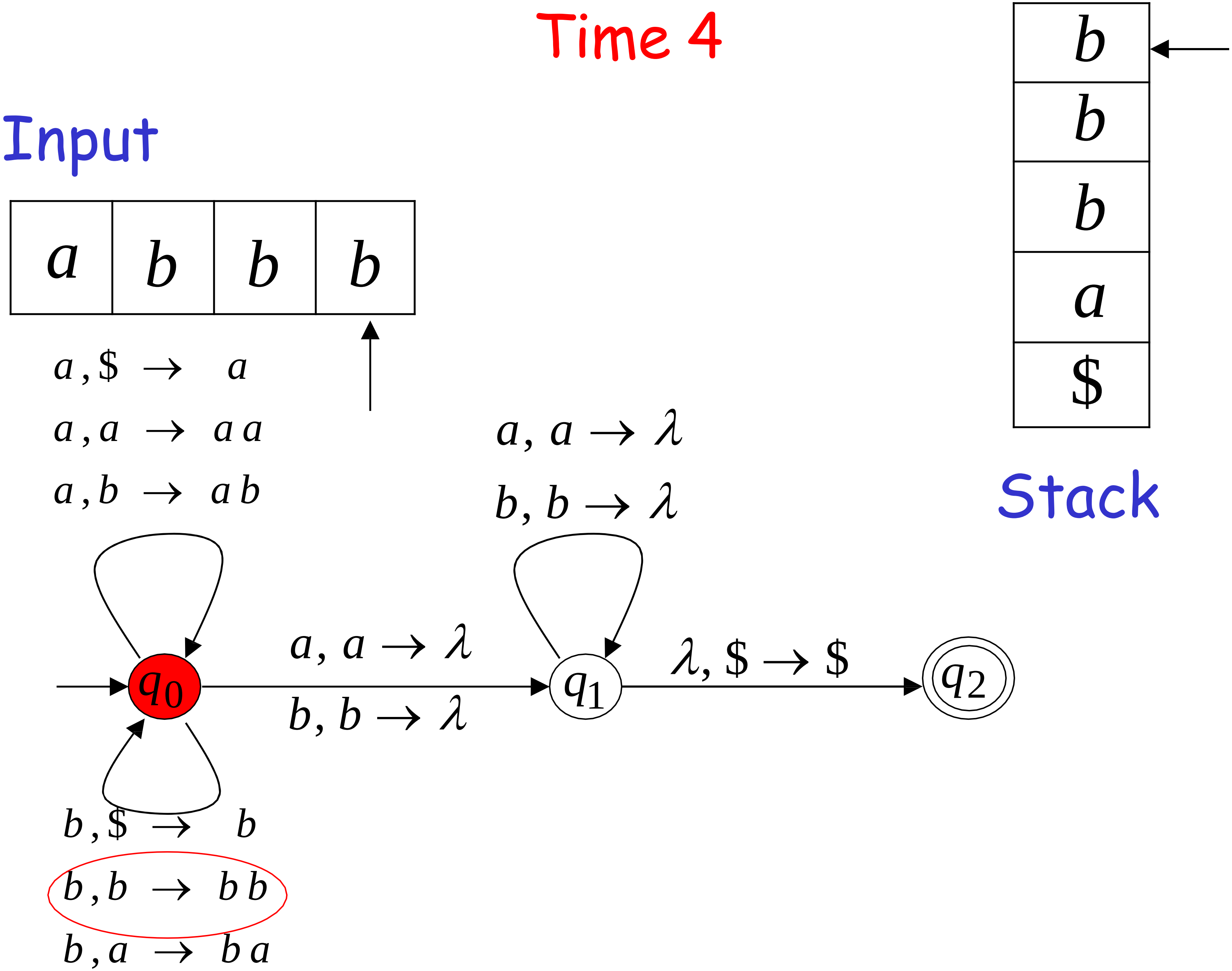


Input
a b b b
Stack
b
a
$
$a, \$ \to a$
$a, a \to aa$
$a, b \to ab$
$b, \$ \to b$
$b, b \to bb$
$b, a \to ba$
$a, a \to \lambda$
$b, b \to \lambda$
$a, a \to \lambda$
$b, b \to \lambda$
$\lambda, \$ \to \$$
q_0
q_1
q_2

Time 3


Input
a b b b
$a, \$ \rightarrow a$
$a, a \rightarrow aa$
$a, b \rightarrow ab$
$b, \$ \rightarrow b$
$b, b \rightarrow bb$
$b, a \rightarrow ba$
q_0
$a, a \rightarrow \lambda$
$b, b \rightarrow \lambda$
q_1
$a, a \rightarrow \lambda$
$b, b \rightarrow \lambda$
$\lambda, \$ \rightarrow \$$
q_2
b
b
a
$
Stack

Time 4



Input
a b b b
a , $ → a
a , a → a a
a , b → a b
b , $ → b
b , b → b b
b , a → b a
a, a → λ
b, b → λ
λ, $ → $
q0
q1
q2
Stack
b
b
b
a
$

Time 5

There is no possible transition.

No accept state

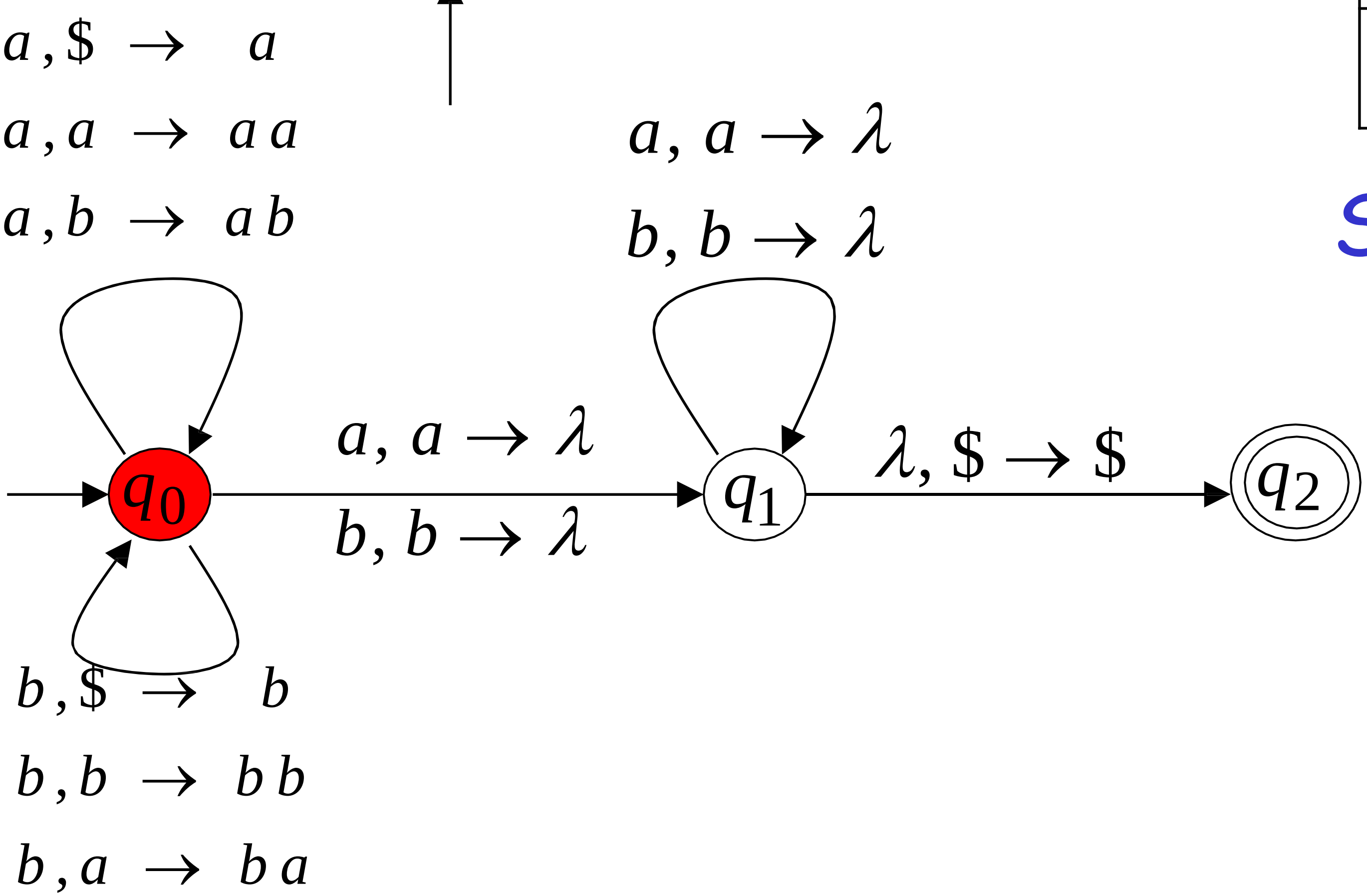
is reached

Input

a	b	b	b

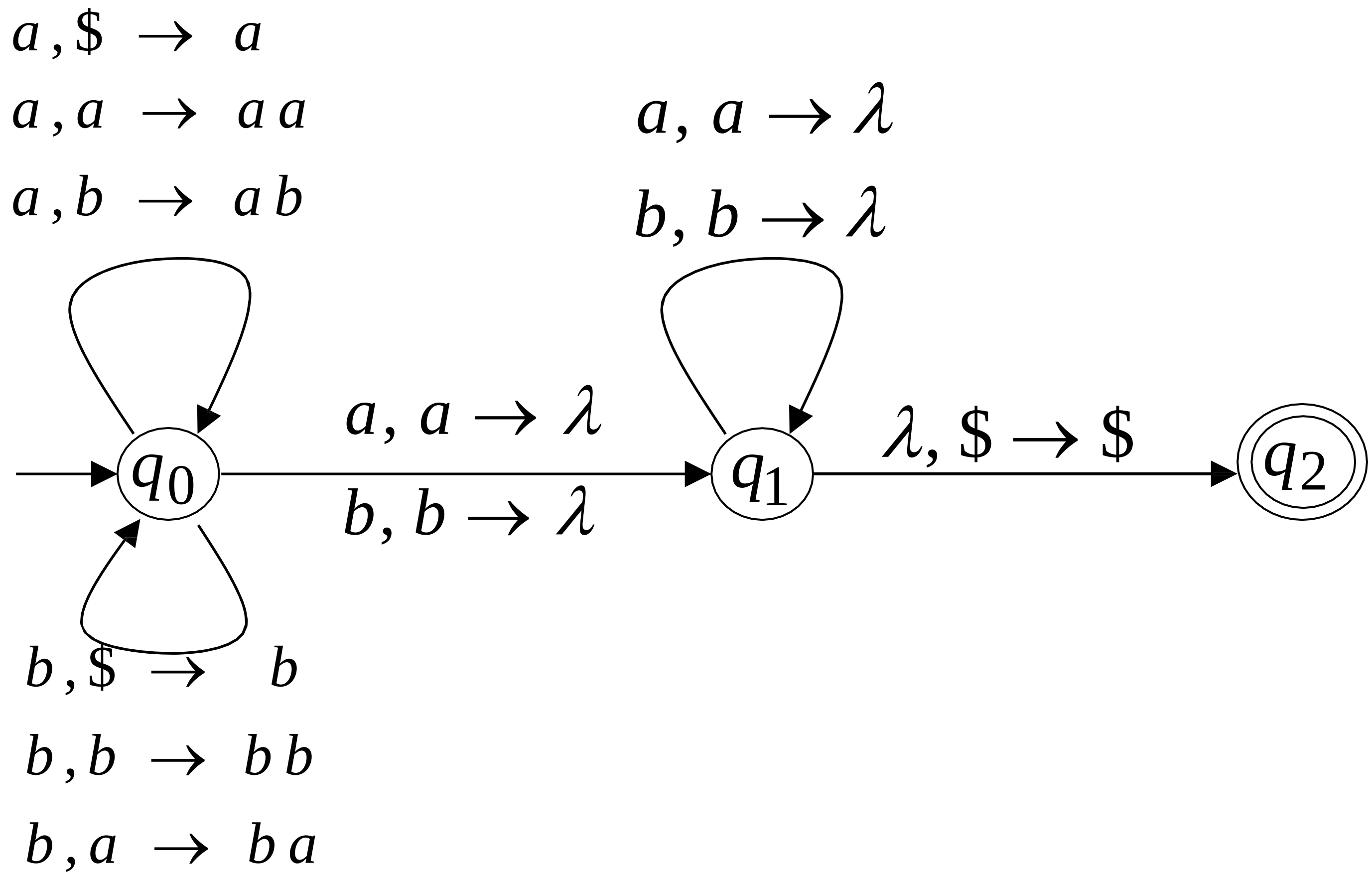
Stack

b
b
b
a
$\$$



There is no computation that accepts string $abbb$

$$abbb \notin L(M)$$

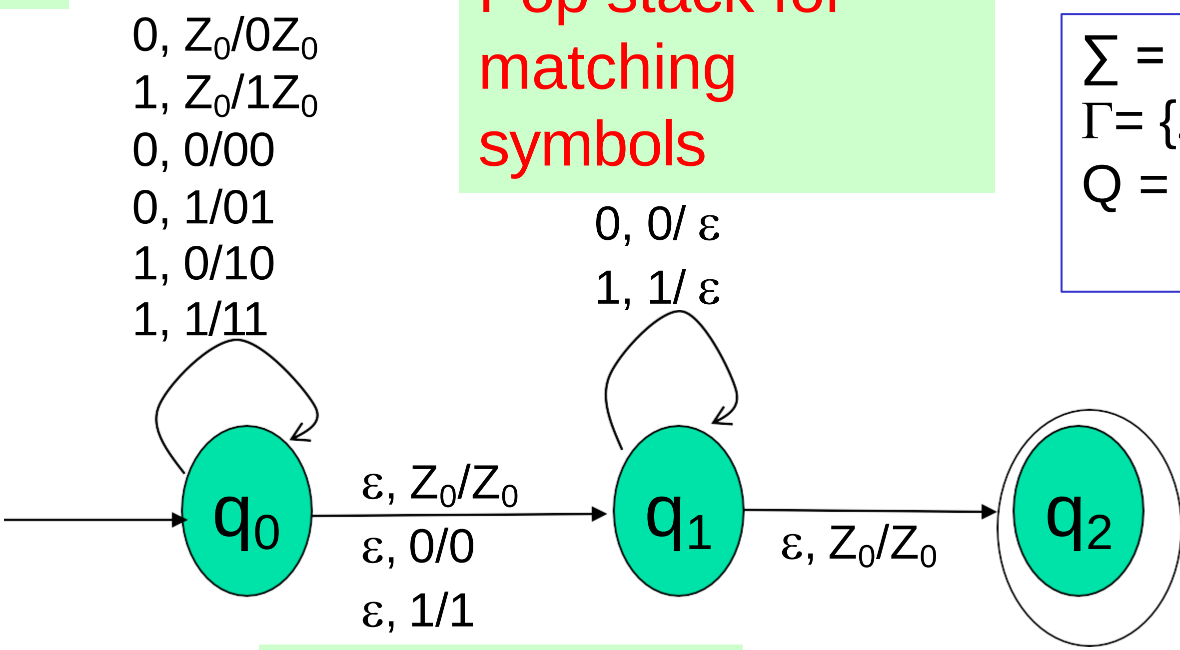


PDA for L_{wwr}: Transition Diagram

Grow stack

Pop stack for matching symbols

$\Sigma = \{0, 1\}$
$\Gamma = \{Z_0, 0, 1\}$
$Q = \{q_0, q_1, q_2\}$

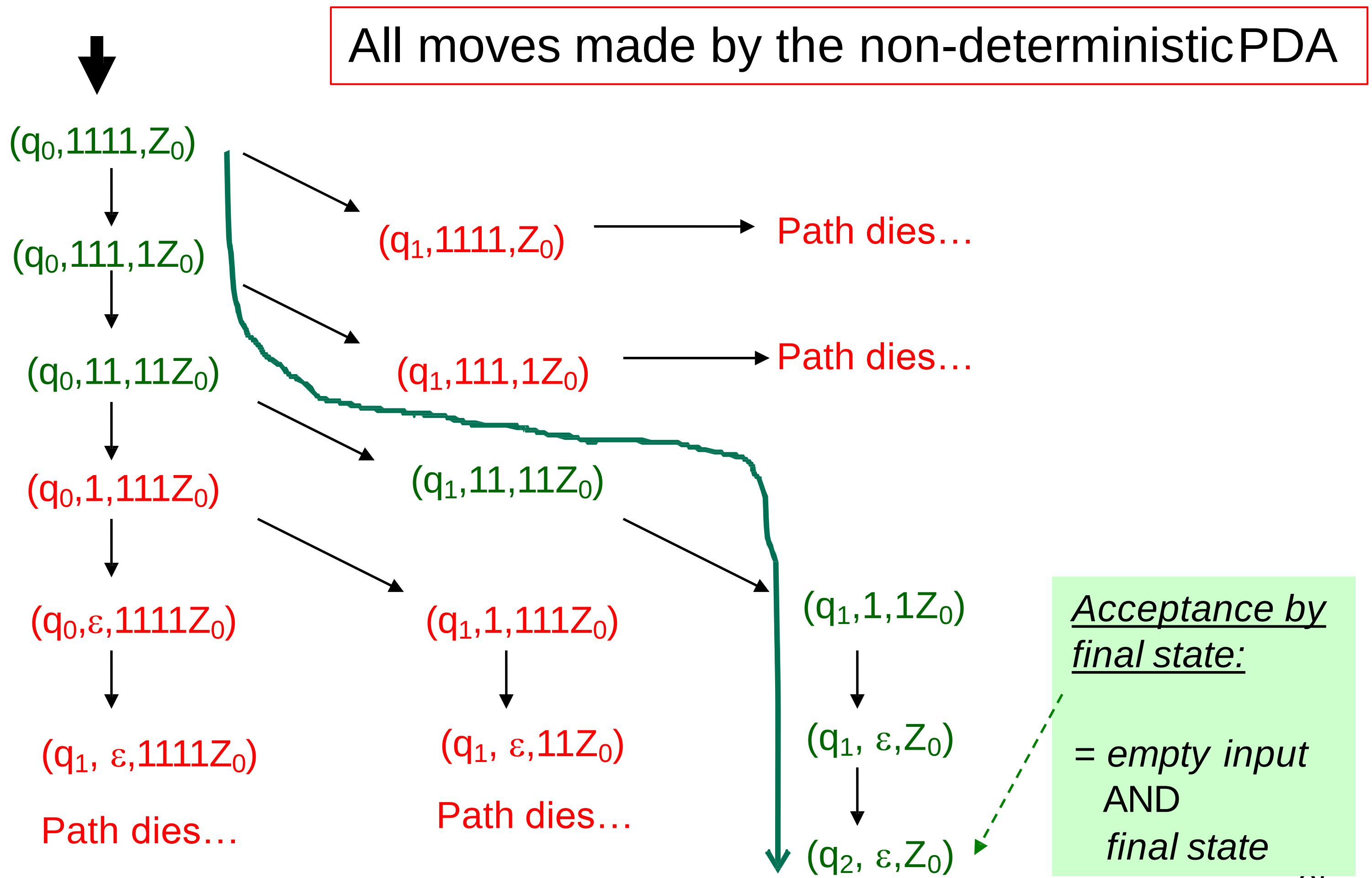


Switch to popping mode

Go to acceptance

This would be a non-deterministic PDA

How does the PDA for L_{wwr} work on input “1111”?



Recall

- PDA
- Deterministic PDA
- Non-Deterministic PDA

Instantaneous Description

$$(q,\ u,\ s)$$

Current state

Remaining input

Current stack contents

PDA's Instantaneous Description (ID)

A PDA has a configuration at any given instance:

(q,w,y)

- q - current state
- w - remainder of the input (i.e., unconsumed part)
- y - current stack contents as a string from top to bottom of stack

---

If δ(q,a, X)={(p, A)} is a transition, then the following are also true:

- (q, a, X) |--- (p,ε,A)
- (q, aw, XB) |--- (p,w,AB)

|--- sign is called a "turnstile notation" and represents one move

|---* sign represents a sequence of moves

Then the following move is possible:

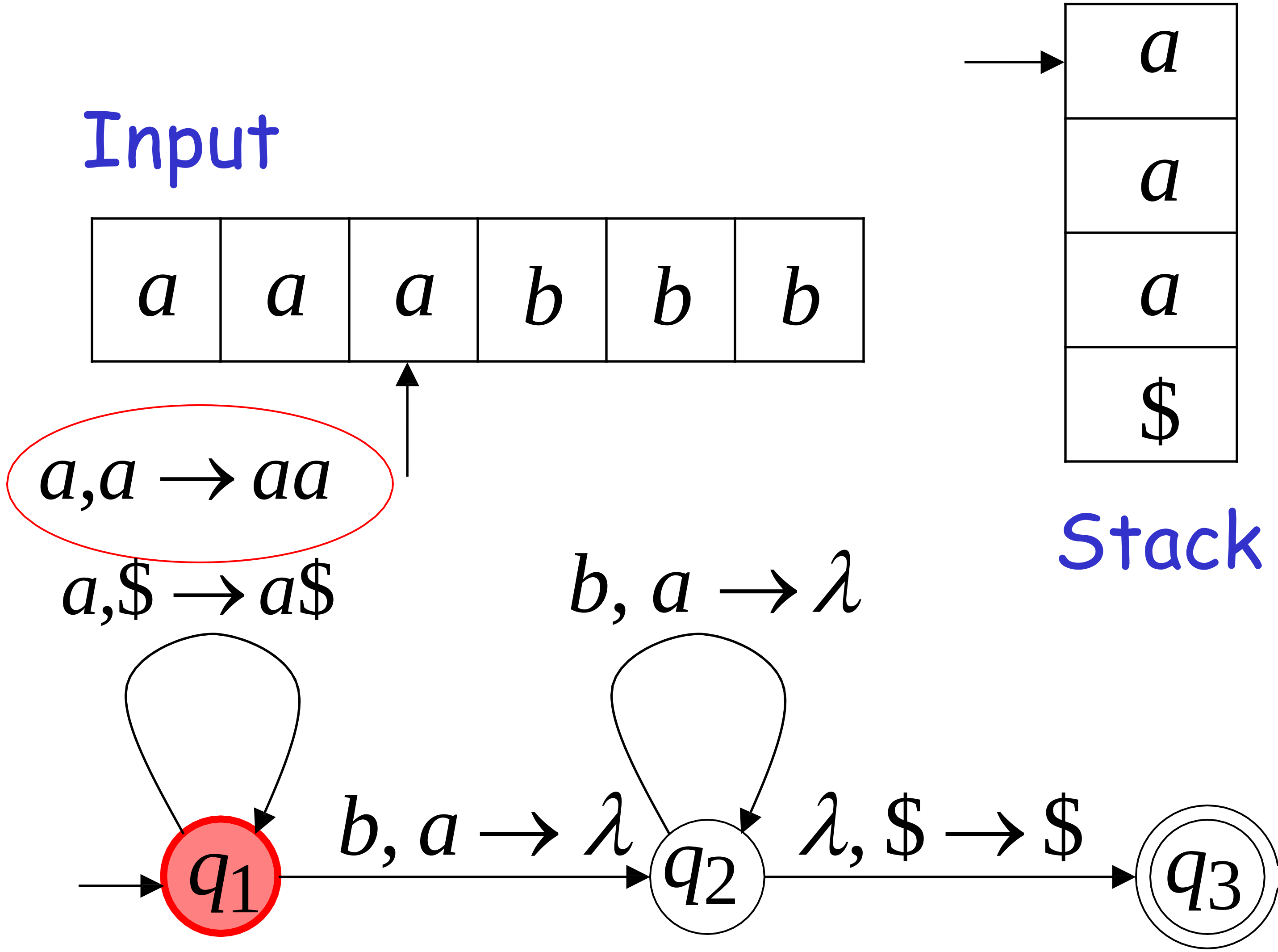
$$(q1, aW, xZ) \ \vdash\!\!-\!\!-\ (q2, W, yZ)$$

where W indicates the rest of the string following the a, and Z indicates the rest of the stack contents underneath the x. This notation says that in moving from state q1 to state q2, an a is consumed from the input string aW, and the x at the top (left) of the stack xZ is replaced with y, leaving yZ on the stack.

Example: Instantaneous Description

$$(q_1, bbb, aaa\$)$$

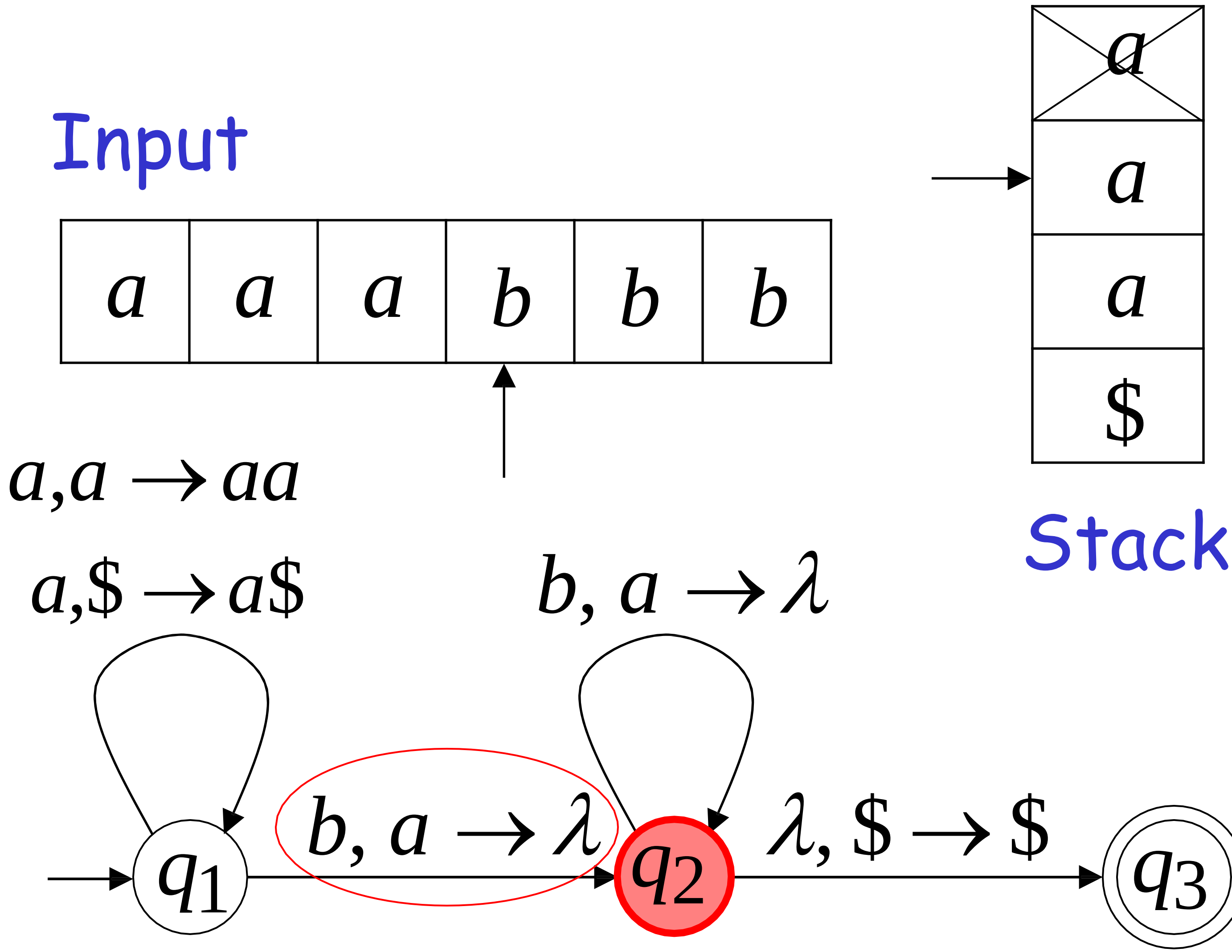
Time 4:



Example: **Instantaneous Description**

$$(q_2, bb, aa\$)$$

Time 5:



We write:

$$(q_1, bbb, aaa\$) \ \phi \ (q_2, bb, aa\$)$$

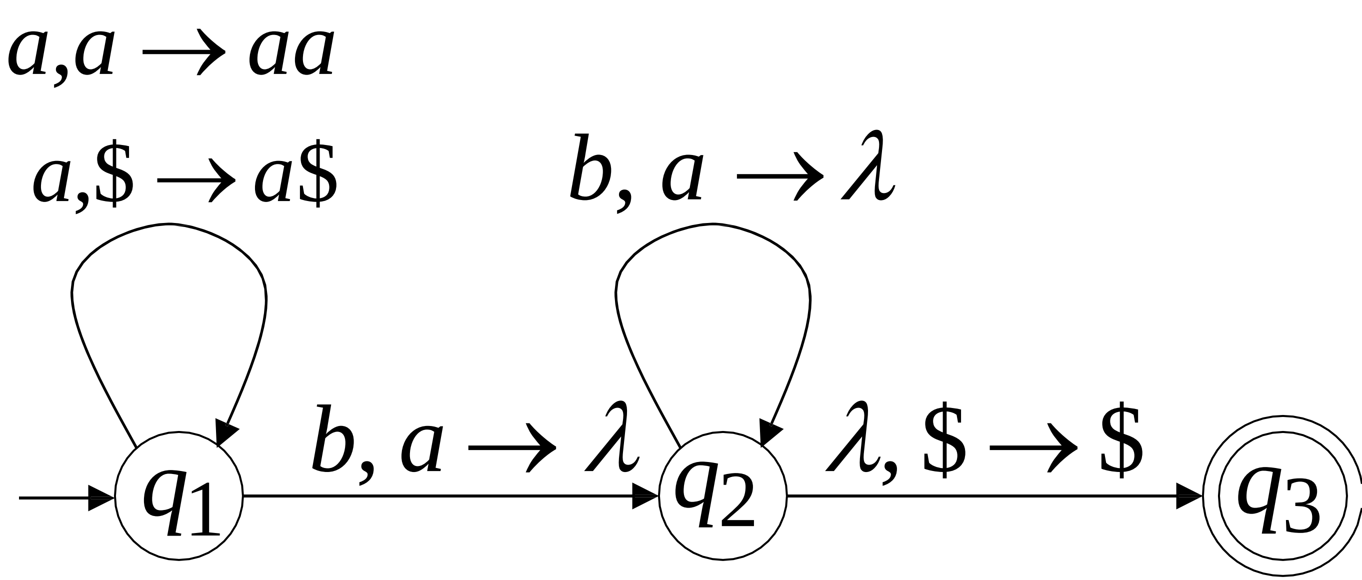
Time 4 Time 5

A computation:

$$(q_0, aaabbb, \$) \phi (q_1, aaabbb, \$) \phi$$

$$(q_1, aabbb, a\$) \phi (q_1, abbb, aa\$) \phi (q_1, bbb, aaa\$) \phi$$

$$(q_2, bb, aa\$) \phi (q_2, b, a\$) \phi (q_2, \lambda, \$) \phi (q_3, \lambda, \$)$$



$$(q_0, aaabbb, \$) \phi (q_1, aaabbb, \$) \phi$$

$$(q_1, aabbb, a\$) \phi (q_1, abbb, aa\$) \phi (q_1, bbb, aaa\$) \phi$$

$$(q_2, bb, aa\$) \phi (q_2, b, a\$) \phi (q_2, \lambda, \$) \phi (q_3, \lambda, \$)$$

For convenience we write:

$$(q_0, aaabbb, \$) \overset{*}{\phi} (q_3, \lambda, \$)$$

Language of PDA

Language $L(M)$ accepted by PDA M:

$$L(M) = \{w : \quad (q_0, w, \$) \overset{*}{\succ} (q_f, \lambda, \$)\}$$

Initial state

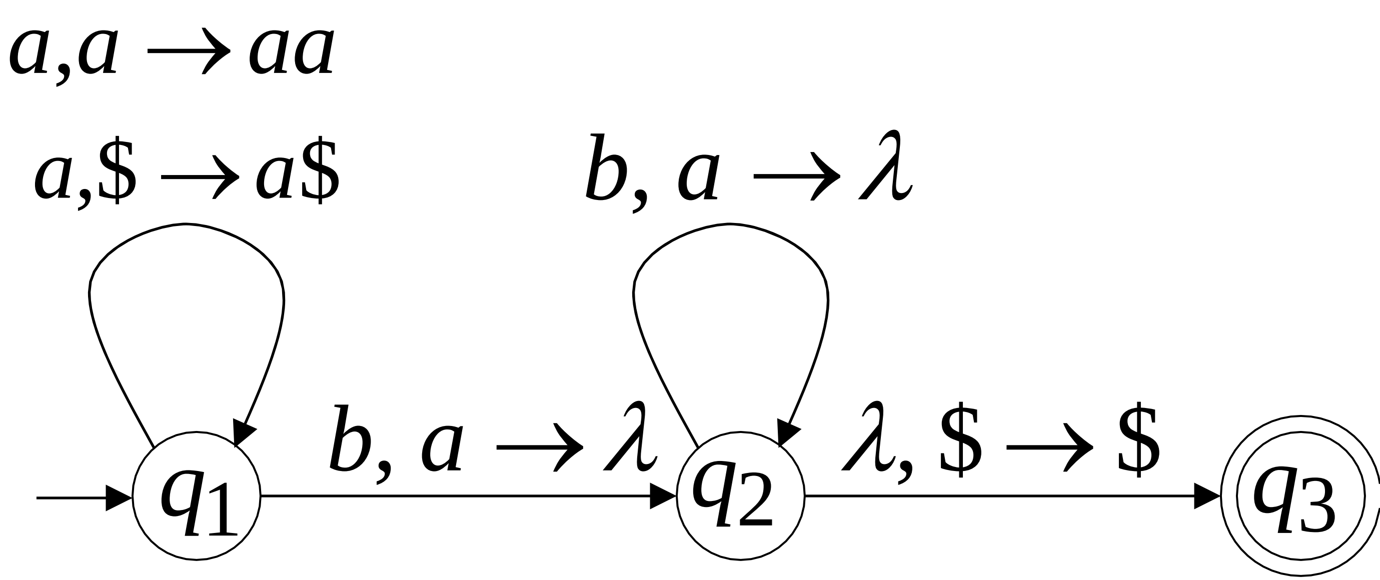
Accept state

Example:

$$(q_0, aaabbb, \$) \overset{*}{\succ} (q_3, \lambda, \$)$$

$$aaabbb \in L(M)$$

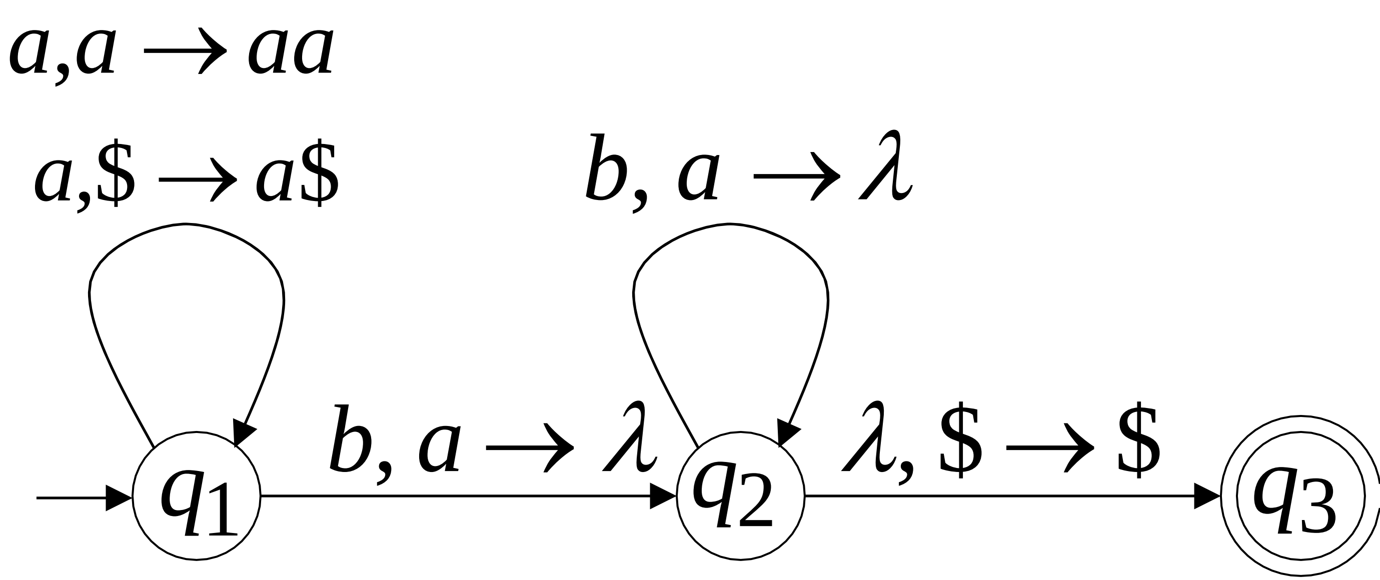
PDA M:



$$(q_0, a^n b^n, \$) \overset{*}{\vdash} (q_3, \lambda, \$)$$

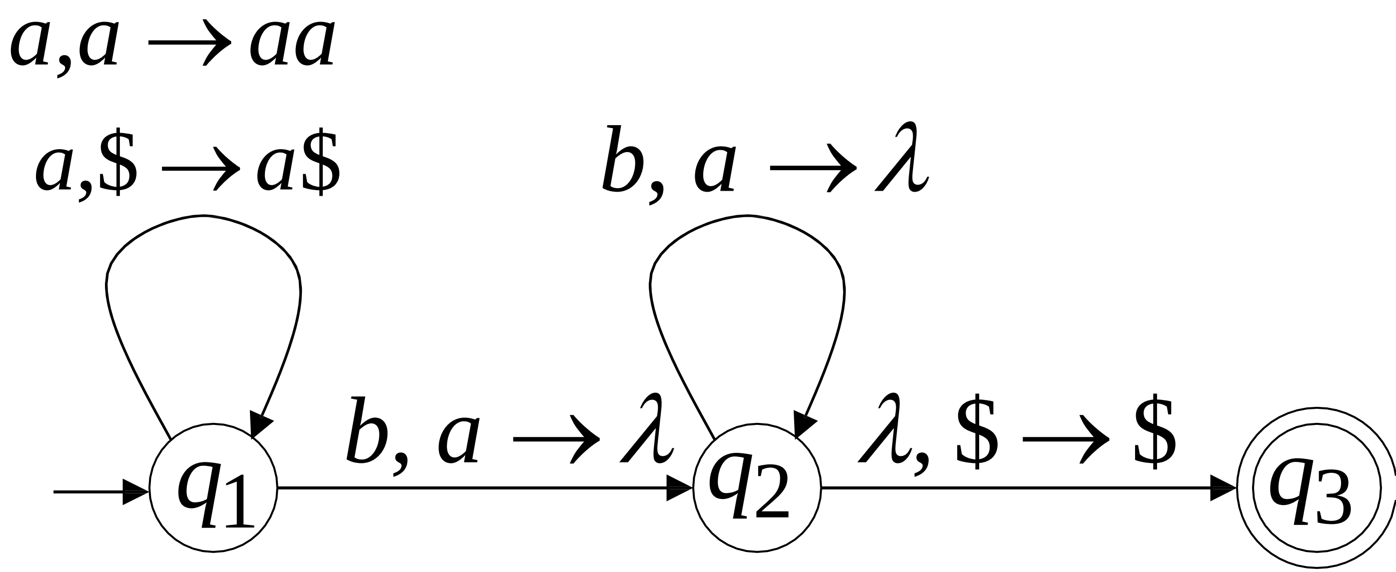
$$a^n b^n \in L(M)$$

PDA M :



Therefore: $L(M) = \{a^n b^n : n \geq 0\}$

PDA M :



More Examples on DCFL and CFL

1) $a^{m+n}b^mc^n \mid n,m>1$ DCFL CFL

2) $a^mb^{m+n}c^n \mid n,m>1$ DCFL CFL

3) $a^nb^mc^{m+n} \mid n,m>1$ DCFL CFL

4) $a^mb^mc^nd^n \mid n,m>1$ DCFL CFL

5) $a^mb^nc^md^n \mid n,m>1$ Not DCFL Not CFL

6) $a^mb^nc^nd^m \mid n,m>1$ DCFL CFL

7) $a^mb^ic^md^k \mid m>1$ DCFL CFL

8) $a^mb^n \mid m>n$ DCFL CFL

9) $a^mb^{2m} \mid m>1$ DCFL CFL

1) $a^n b^{n^2} | m,n>1$ Not DCFL Not CFL

2) $a^m b^{2^n} | n,m>1$ Not DCFL Not CFL

3) $ww^R | w€(a,b)^*$ NDCFL CFL

4) $ww | w€(a,b)^*$ Not DCFL Not CFL

5) $a^n b^n c^m | n>m$ Not DCFL Not CFL

6) $a^n b^n c^n d^n | n<10^{10}$ DCFL CFL

7) $a^m b^{2m} c^{3m} | m>1$ Not CFL

8) $xcy | x,y€(a,b)^*$ RL DCFL CFL

9) $a^m b^n | m\#n$ DCFL CFL

Recall

- PDA
- Deterministic PDA
- Non-Deterministic PDA
- (Instantaneous Description) ID's
- construction of PDA from CFG

PDAs Accept
Context-Free Languages

Theorem:

$$\left\{\begin{array}{c}\text{Context-Free} \\ \text{Languages} \\ \text{(Grammars)}\end{array}\right\} \qquad \left\{\begin{array}{c}\text{Languages} \\ \text{Accepted by} \\ \text{PDAs}\end{array}\right\}$$

Proof - Step 1:

$$\left\{\begin{array}{c}\text{Context-Free} \\ \text{Languages} \\ \text{(Grammars)}\end{array}\right\} \subseteq \left\{\begin{array}{c}\text{Languages} \\ \text{Accepted by} \\ \text{PDAs}\end{array}\right\}$$

Convert any context-free grammar G to a PDA M with: $L(G) = L(M)$

Proof - Step 2:

$$\left\{ \begin{array}{c} \text{Context-Free} \\ \text{Languages} \\ \text{(Grammars)} \end{array} \right\} \supseteq \left\{ \begin{array}{c} \text{Languages} \\ \text{Accepted by} \\ \text{PDAs} \end{array} \right\}$$

Convert any PDA M to a context-free grammar G with: $L(G) = L(M)$

Proof - step 1

Convert

Context-Free Grammars
to
PDAs

Take an arbitrary context-free grammar G

We will convert G to a PDA M such that:

$$L(G) = L(M)$$

Procedure to Convert CFG to PDA

- $G = (V, \sum, S, P)$
- $NT(G)=(Q, \sum, \Gamma, q_0, \$, A, \delta)$

  $Q = \{q_0, q_1, q_2\}$ $A = \{q_2\}$ $\Gamma = V \cup \sum \cup \{\$\}$
- *On initial state,* $\delta(q_0, \lambda, \lambda) = \{(q_1, S)\}$
- The moves from q_1 are the following:

  For every $A \in V$,

  $\delta(q_1, \lambda, A) = \{(q_1, \alpha) \mid A \rightarrow \alpha \text{ is a production in } G\}$

  For every *terminal* $a \in \sum$,

  $\delta(q_1, a, a) = \{(q_1, \lambda)\}$
- *For accepting state*

  $\delta(q_1, \lambda, \$) = \{(q_2, \$)\}$

Conversion Procedure:

S❼ b
Ab

A❼ a
A|a

For each Variable/production in G

For each terminal in G

$A \rightarrow a$

$A \rightarrow aA$

$S \rightarrow bAb$

a

b

Add transitions

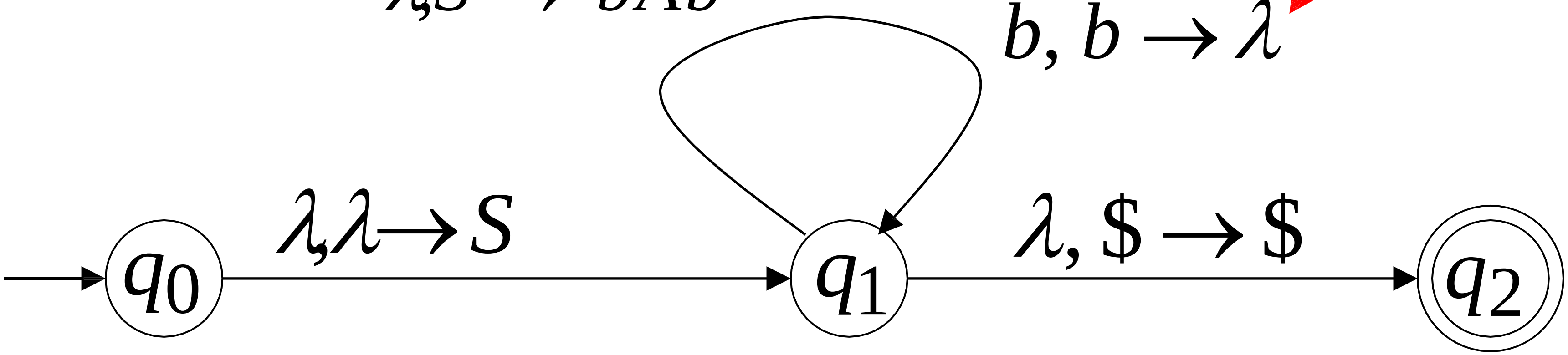
$\lambda, A \rightarrow a$

$\lambda, A \rightarrow aA$

$\lambda, S \rightarrow bAb$

$a, a \rightarrow \lambda$

$b, b \rightarrow \lambda$

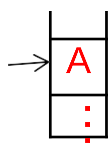


Formal construction of PDA from CFG

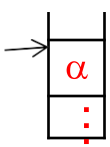
Note: Initial stack symbol (S) same as the start variable in the grammar

- **Given:** G= (V,T,P,S)
- **Output:** $P_N = (\{q\}, T, V \cup T, \delta, q, S)$
- δ:

Before:

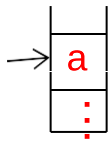


After:



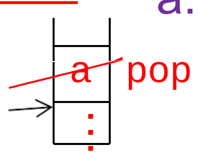
- For all $A \in V$, add the following transition(s) in the PDA:
  - $\delta(q, \varepsilon, A) = \{ (q, \alpha) \mid$ "$A ==> \alpha \in P\}$

Before:



After:

a…



- For all $a \in T$, add the following transition(s) in the PDA:
  - $\delta(q,a,a)= \{ (q, \varepsilon) \}$

Example

Grammar

$S \to aSTb$

$S \to b$

$T \to Ta$

$T \to \lambda$

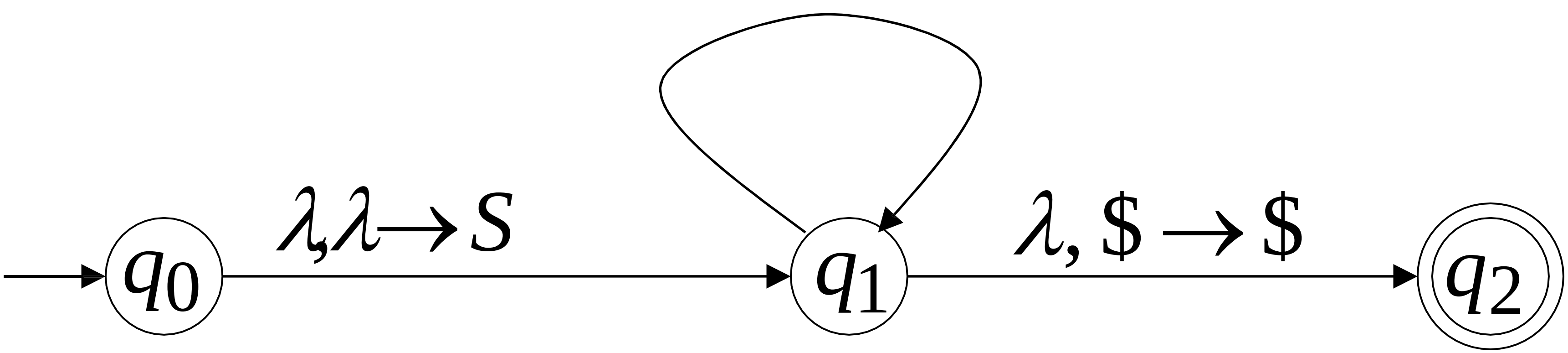
PDA

$\lambda, S \to aSTb$

$\lambda, S \to b$

$\lambda, T \to Ta$ $\qquad a, a \to \lambda$

$\lambda, T \to \lambda$ $\qquad b, b \to \lambda$



Example:

Input

a	b	a	b

Time 0

$\lambda, S \rightarrow aSTb$

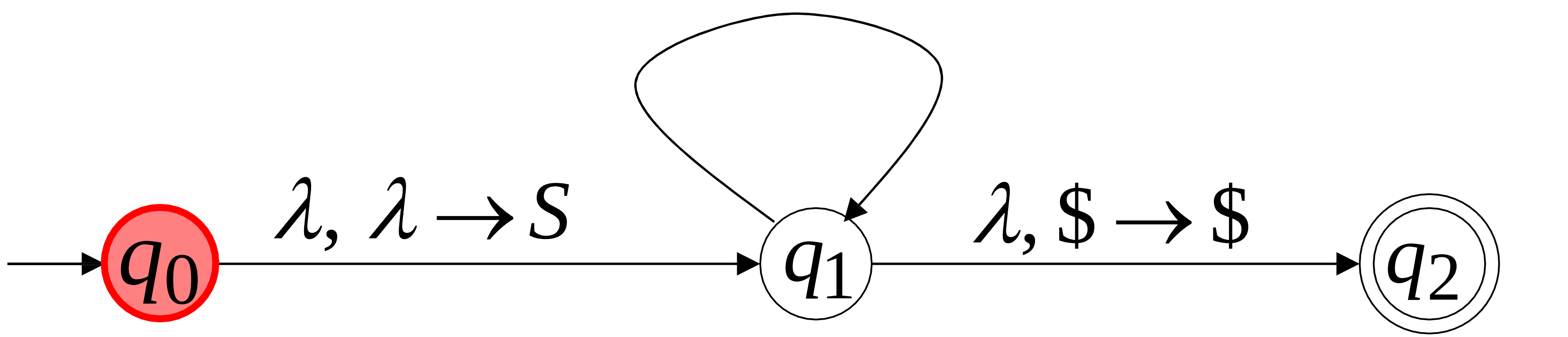
$\lambda, S \rightarrow b$

$\lambda, T \rightarrow Ta$ $\quad$ $a, a \rightarrow \lambda$

$\lambda, T \rightarrow \lambda$ $\quad$ $b, b \rightarrow \lambda$

$\$$

Stack



Derivation: S

Input

a	b	a	b

Time 1

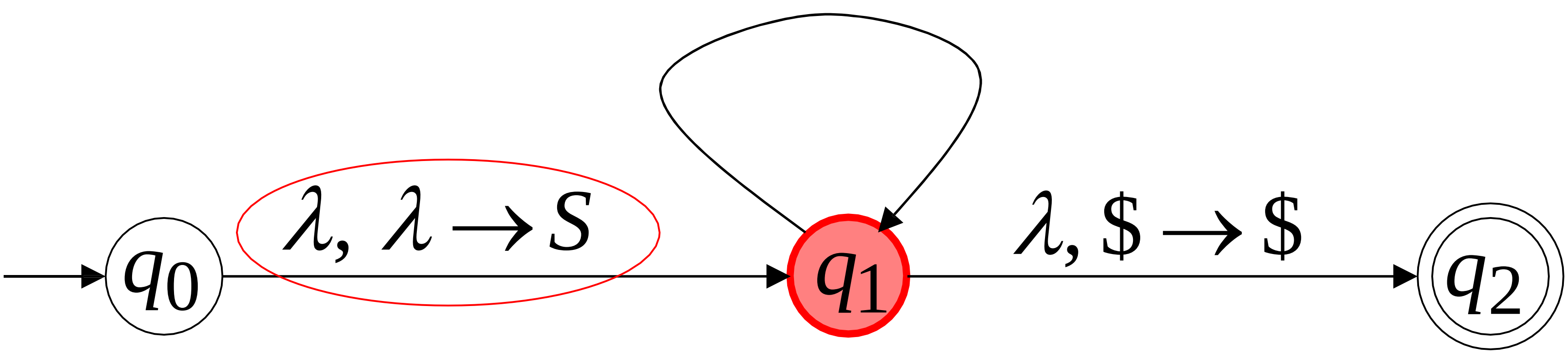
$\lambda, S \rightarrow aSTb$

$\lambda, S \rightarrow b$

$\lambda, T \rightarrow Ta$ $\qquad a, a \rightarrow \lambda$

$\lambda, T \rightarrow \lambda$ $\qquad b, b \rightarrow \lambda$

Stack
S
$\$$



Derivation: $S \Rightarrow aSTb$

Input

a	b	a	b

Time 2

$\lambda, S \to aSTb$

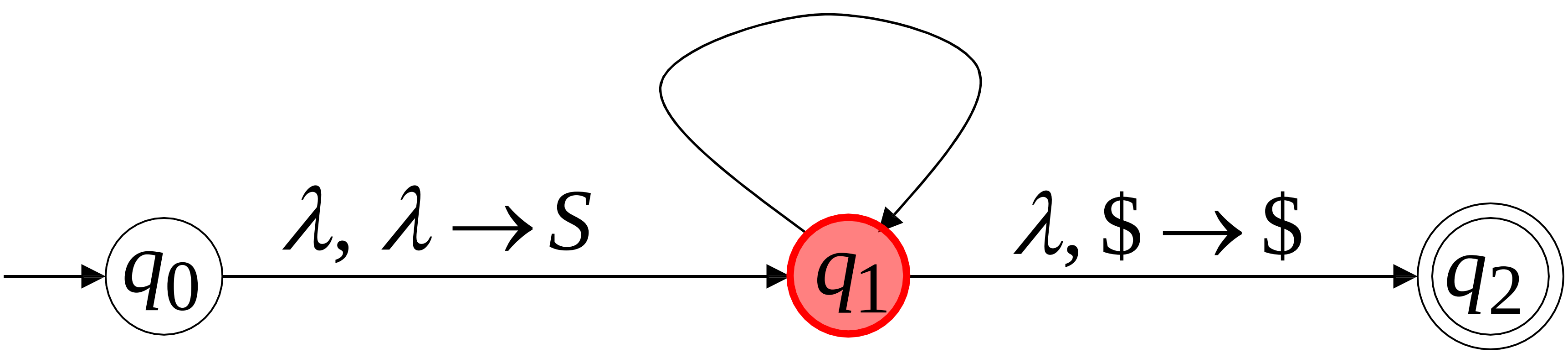
$\lambda, S \to b$

$\lambda, T \to Ta$ $\quad$ $a, a \to \lambda$

$\lambda, T \to \lambda$ $\quad$ $b, b \to \lambda$

a
S
T
b
$\$$

Stack



Derivation: $S \Rightarrow aSTb$

Input

a	b	a	b

Time 3

$\lambda, S \rightarrow aSTb$

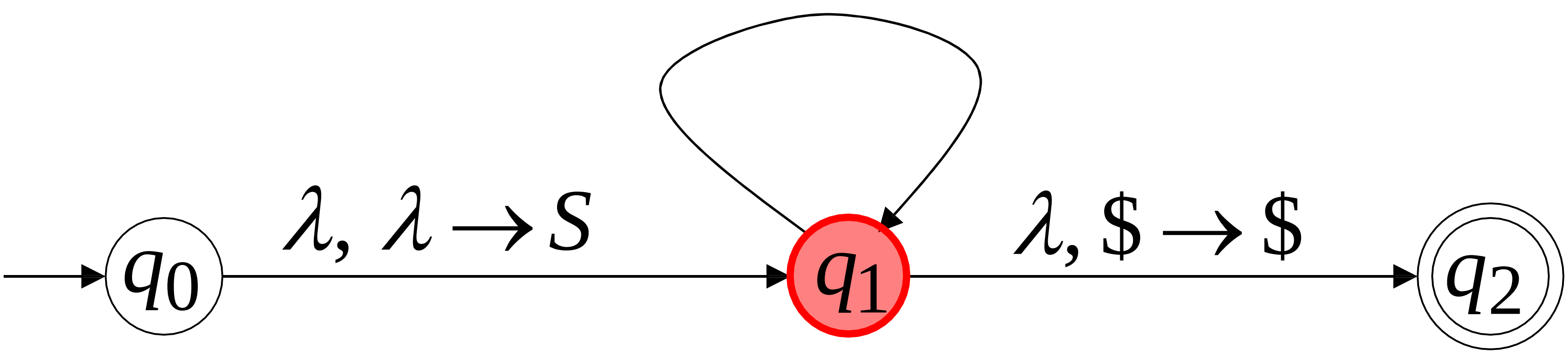
$\lambda, S \rightarrow b$

$\lambda, T \rightarrow Ta$ $\quad$ $a, a \rightarrow \lambda$

$\lambda, T \rightarrow \lambda$ $\quad$ $b, b \rightarrow \lambda$

$\cancel{a}$
S
T
b
$\$$

Stack



Derivation: $S \Rightarrow aSTb \Rightarrow abTb$

Input

a	b	a	b

Time 4

$\lambda, S \rightarrow aSTb$

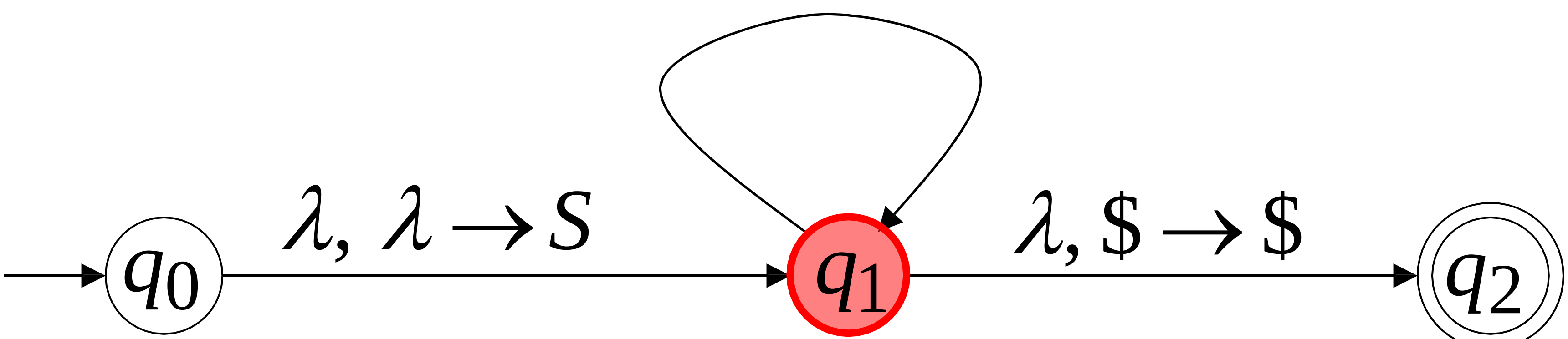
$\lambda, S \rightarrow b$

$\lambda, T \rightarrow Ta$ $\quad a, a \rightarrow \lambda$

$\lambda, T \rightarrow \lambda$ $\quad b, b \rightarrow \lambda$

b
T
b
$\$$

Stack



Derivation: $S \Rightarrow aSTb \Rightarrow abTb$

Input

a	b	a	b

Stack

$\cancel{b}$
T
b
$\$$

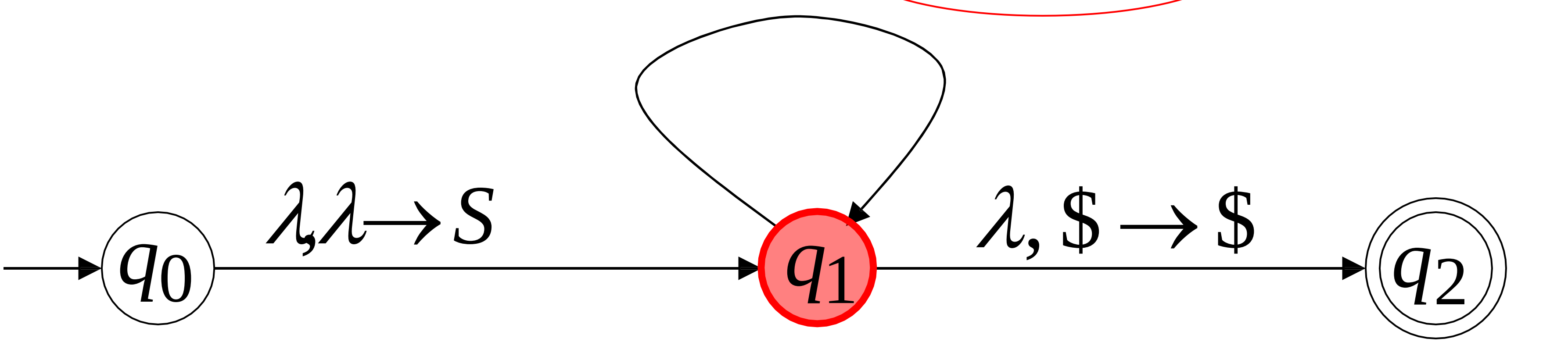
Time 5

$\lambda, S \rightarrow aSTb$

$\lambda, S \rightarrow b$

$\lambda, T \rightarrow Ta$ $\qquad a, a \rightarrow \lambda$

$\lambda, T \rightarrow \lambda$ $\qquad b, b \rightarrow \lambda$



Derivation: $S \Rightarrow aSTb \Rightarrow abTb \Rightarrow abTab$

Input

a	b	a	b

Time 6

$\lambda, S \rightarrow aSTb$

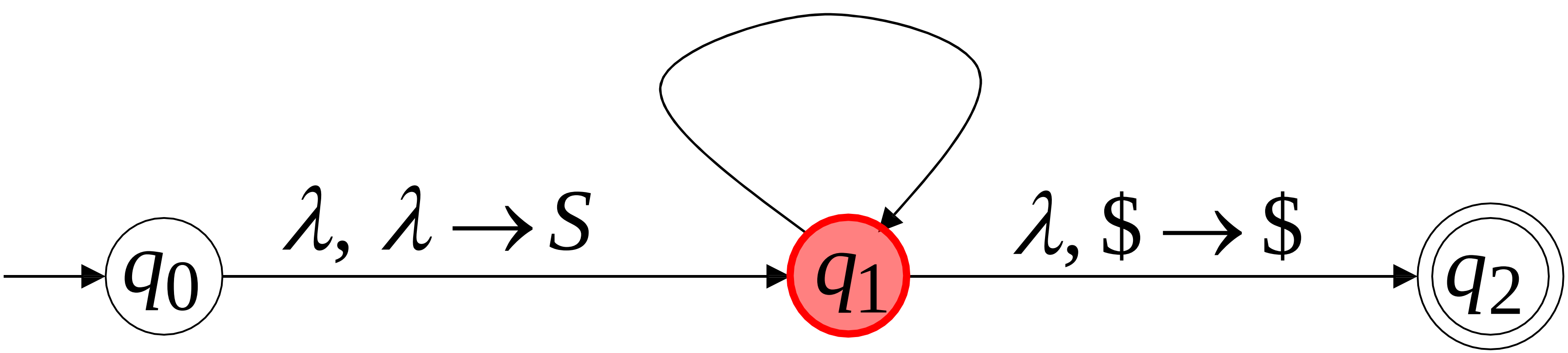
$\lambda, S \rightarrow b$

$\lambda, T \rightarrow Ta$ $\qquad a, a \rightarrow \lambda$

$\lambda, T \rightarrow \lambda$ $\qquad b, b \rightarrow \lambda$

T
a
b
$\$$

Stack



Derivation: $S \Rightarrow aSTb \Rightarrow abTb \Rightarrow abTab \Rightarrow abab$

Input

a	b	a	b

Time 7

$\lambda, S \rightarrow aSTb$

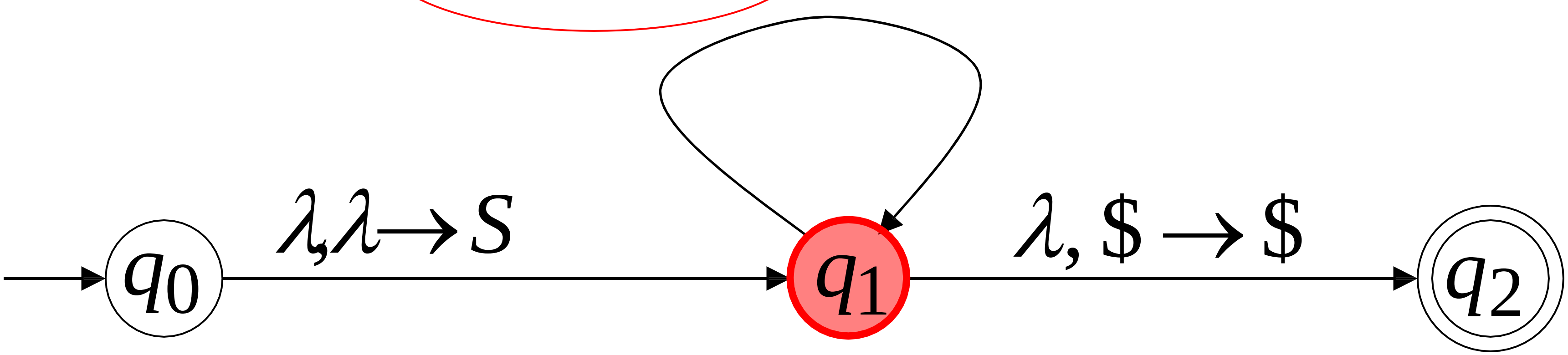
$\lambda, S \rightarrow b$

$\lambda, T \rightarrow Ta$ $\qquad a, a \rightarrow \lambda$

$\lambda, T \rightarrow \lambda$ $\qquad b, b \rightarrow \lambda$

$\cancel{T}$
a
b
$\$$

Stack



Derivation: $S \Rightarrow aSTb \Rightarrow abTb \Rightarrow abTab \Rightarrow abab$

Input

a	b	a	b

Time 8

$\lambda, S \rightarrow aSTb$

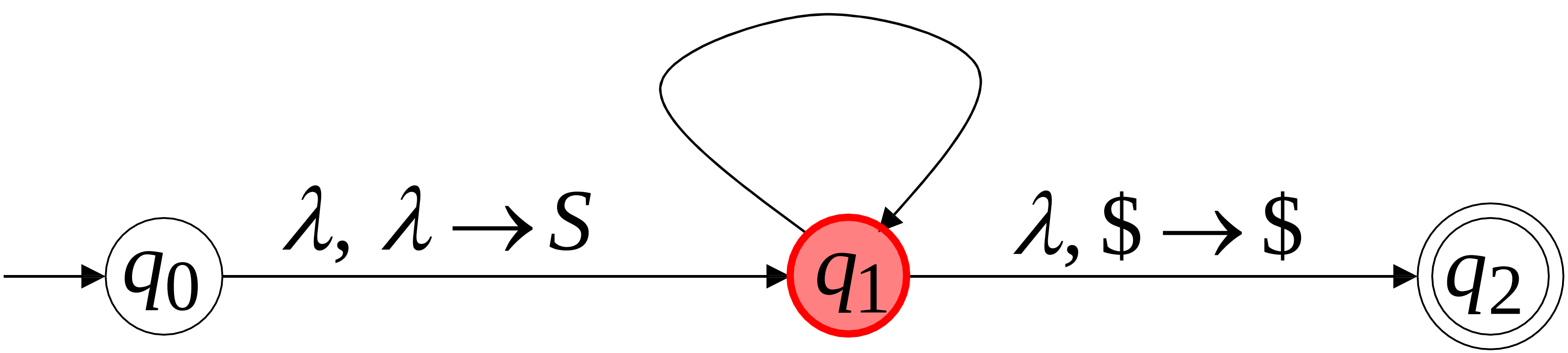
$\lambda, S \rightarrow b$

$\lambda, T \rightarrow Ta$ $\quad$ $a, a \rightarrow \lambda$

$\lambda, T \rightarrow \lambda$ $\quad$ $b, b \rightarrow \lambda$

$\cancel{a}$
b
$\$$

Stack



Derivation: $S \Rightarrow aSTb \Rightarrow abTb \Rightarrow abTab \Rightarrow abab$

Input

a	b	a	b

Time 9

$\lambda, S \rightarrow aSTb$

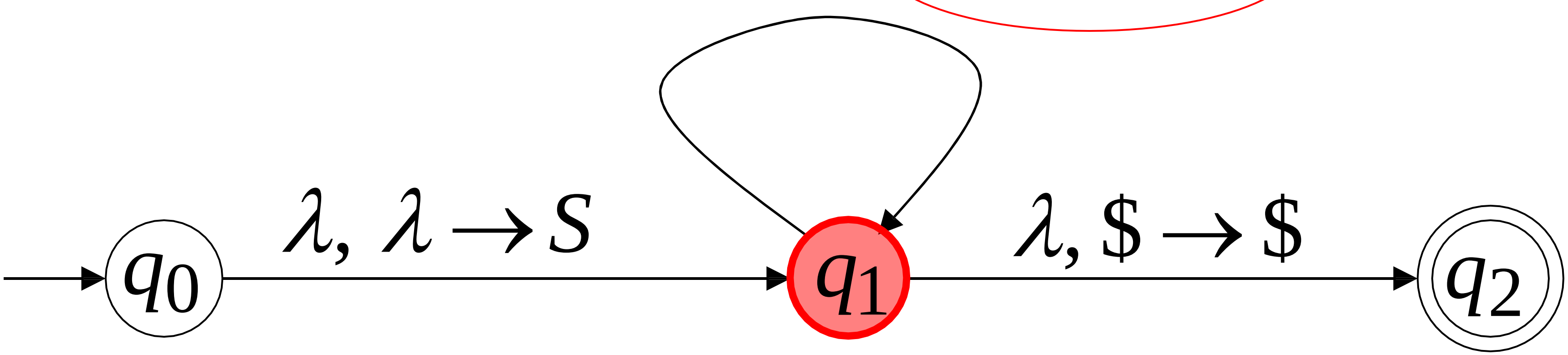
$\lambda, S \rightarrow b$

$\lambda, T \rightarrow Ta$ $\quad$ $a, a \rightarrow \lambda$

$\lambda, T \rightarrow \lambda$ $\quad$ $b, b \rightarrow \lambda$

$\cancel{b}$
$\$$

Stack



Derivation: $S \Rightarrow aSTb \Rightarrow abTb \Rightarrow abTab \Rightarrow abab$

Input

a	b	a	b

Time 10

$\lambda, S \rightarrow aSTb$

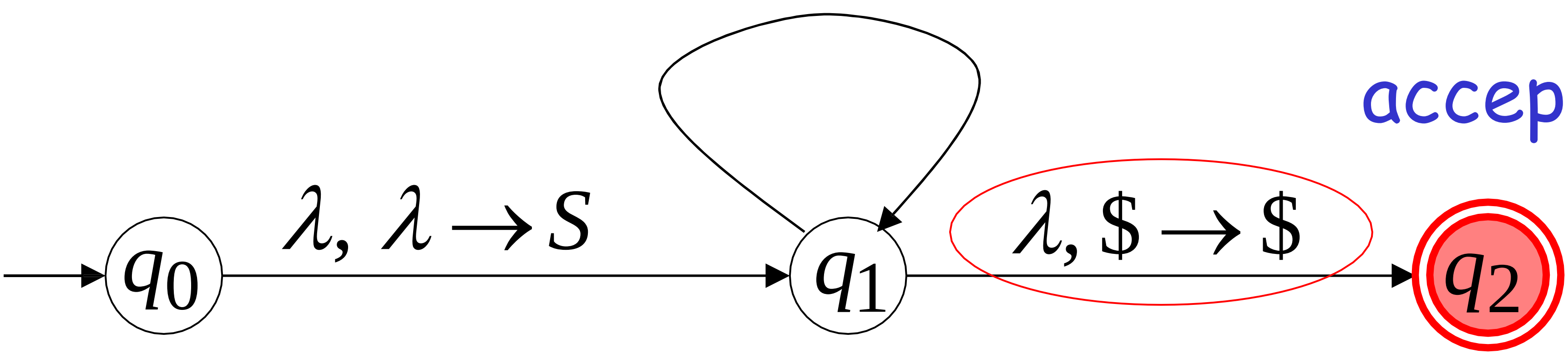
$\lambda, S \rightarrow b$

$\lambda, T \rightarrow Ta$ $\qquad a, a \rightarrow \lambda$

$\lambda, T \rightarrow \lambda$ $\qquad b, b \rightarrow \lambda$

Stack
$\$$

accept



Grammar
Leftmost Derivation

S

$\Rightarrow aSTb$

$\Rightarrow abTb$

$\Rightarrow abTab$

$\Rightarrow abab$

PDA Computation

$(q_0, abab, \$)$

□ $(q_1, abab, S\$)$

□ $(q_1, bab, STb\$)$

□ $(q_1, bab, bTb\$)$

□ $(q_1, ab, Tb\$)$

□ $(q_1, ab, Tab\$)$

□ $(q_1, ab, ab\$)$

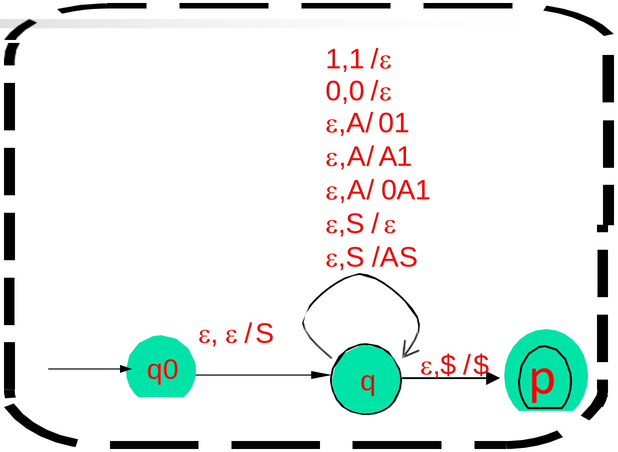
□ $(q_1, b, b\$)$

□ $(q_1, \lambda, \$)$

□ $(q_2, \lambda, \$)$

Example: CFG to PDA

- $G = (\{S,A\}, \{0,1\}, P, S)$
- P:
  - S ==> AS | ε
  - A ==> 0A1 | A1 | 01
- PDA = $(\{q_0,q,p\}, \{0,1\}, \{0,1,A,S\}, \delta, q_0, S)$
- δ:
  - $\delta(q_0, \varepsilon, \varepsilon)=(q,S)$
  - $\delta(q, \varepsilon, S) = \{ (q, AS), (q, \varepsilon)\}$
  - $\delta(q, \varepsilon, A) = \{ (q,0A1), (q,A1), (q,01) \}$
  - $\delta(q, 0, 0) = \{ (q, \varepsilon) \}$
  - $\delta(q, 1, 1) = \{ (q, \varepsilon) \}$
  - $\delta(q, \varepsilon, \$) = \{ (P, \$) \}$



How will this new PDA work?

Lets simulate string <u>0011</u>

Simulating string 0011 on the new PDA …

PDA(δ):

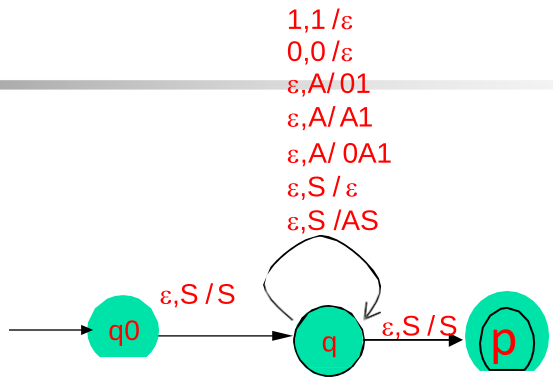
$\delta(q_0, \varepsilon, \varepsilon) = (q, S)$

$\delta(q, \varepsilon, S) = \{ (q, AS), (q, \varepsilon) \}$

$\delta(q, \varepsilon, A) = \{ (q, 0A1), (q, A1), (q, 01) \}$

$\delta(q, 0, 0) = \{ (q, \varepsilon) \}$

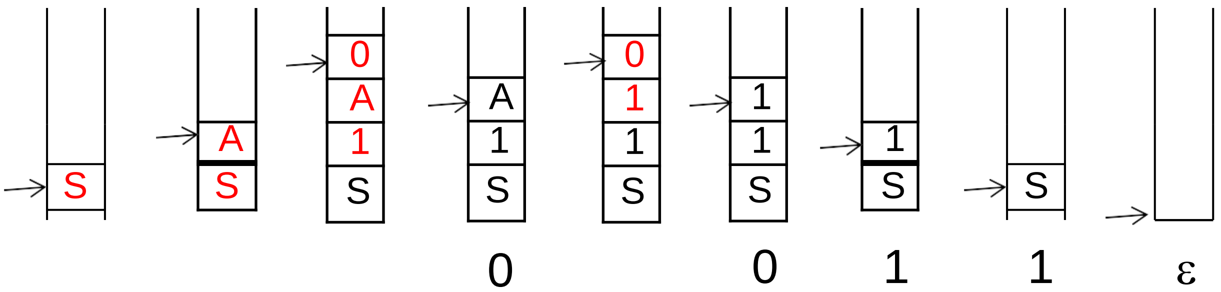
$\delta(q, 1, 1) = \{ (q, \varepsilon) \}$



Leftmost deriv.:

S =>AS
=> 0A1S
=> 0011S
=> 0011

Stack moves (shows only the successful path):



Accept by final state

S =>AS =>0A1S =>0011S => 0011

In general, it can be shown that:

Grammar G generates string w

$$S \overset{*}{\Rightarrow} w$$

If and Only if

PDA M accepts w

$$(q_0, w, \$) \;\square\; (q_2, \lambda, \$)$$

Therefore $L(G) = L(M)$

Construct PDA equivalent to the following CFG.

S->0A1 | 0BA (May-2017 EndSem 8 Marks)

A->S01 | 0

B->1B | 1

$\lambda, S \rightarrow 0A1$

$\lambda, S \rightarrow 0BA$

$\lambda, A \rightarrow S01$

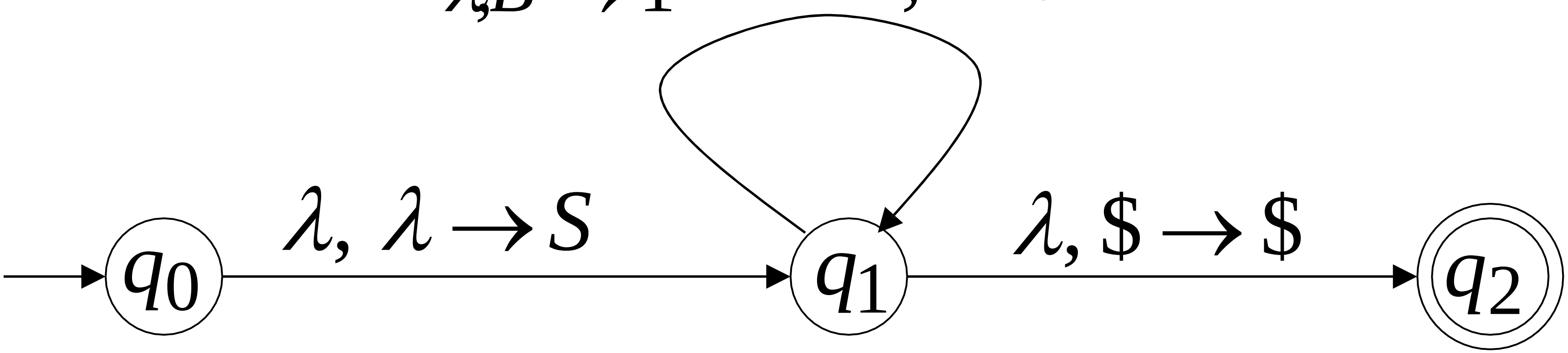
$\lambda, A \rightarrow 0$

$\lambda, B \rightarrow 1B$

$\lambda, B \rightarrow 1$

$1,1 \rightarrow \lambda$

$0,0 \rightarrow \lambda$



There are two types of PDAs that one can design:
those that accept by <u>final state</u> or by <u>empty stack</u>

Acceptance by…

- ***PDAs that accept by final state:***
  - For a PDA P, the language accepted by P, denoted by L(P) by *final state*, is:
    - $\{w \mid (q_0,w,Z_0) \mid\text{---}^* (q,\varepsilon, A) \}$, s.t., $q \in F$

Checklist:
- input exhausted?
- in a final state?

- ***PDAs that accept by empty stack:***
  - For a PDA P, the language accepted by P, denoted by N(P) by *empty stack*, is:
    - $\{w \mid (q_0,w,Z_0) \mid\text{---}^* (q, \varepsilon, \varepsilon) \}$, for any $q \in Q$.

Q) Does a PDA that accepts by <u>empty stack</u>
need any final state specified in the design?

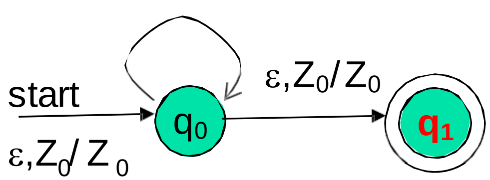
Checklist:
- input exhausted?
- is the stack empty?

Example: L of balanced parenthesis

<u>PDA that accepts by final state</u>

P_F:

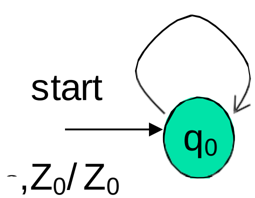
$(,Z_0 / (Z_0$
$(,(/ (($
$), (/ \varepsilon$



<u>An equivalent PDA that accepts by empty stack</u>

P_N:

$(,Z_0 / (Z_0$
$(, (/ (($
$), (/ \varepsilon$
$\boldsymbol{\varepsilon,Z_0 / \varepsilon}$



How will these two PDAs work on the input: ((( )) ( )) ( )

PDA- Acceptance by final state

Let

$$P = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

be a PDA. The **language accepted by** P **by final state is**:

$$L(P) = \{w \mid (q_0, w, Z_0) \overset{*}{\vdash} (q, \epsilon, \alpha), q \in F\}$$

for some state q in F and any input stack string α.

Starting in the initial ID with w waiting on the input, P consumes w from the input and enters an accepting state. **The contents of the stack at that time is irrelevant**.

PDA- Acceptance by empty stack

Let

$$P = (Q,\Sigma,\Gamma,\delta,q_0,Z_0,F)$$

be a PDA. The **language accepted by** P **by empty stackis**:

$$N(P) = \{w \mid (q_0,w,Z_0) \overset{*}{\vdash} (q,\epsilon,\epsilon)\}$$

where q is any state

$N(P)$ is the set of inputs w that P can consume an at the same time empty the stack.

From Empty stack to Final State

- Let P_N be a PDA by empty stack.
- Let P_F be a PDA by final state.

Theorem:

If $L = N(P_N)$ for some PDA P_N, then there exist a PDA P_F, such that

$$L = L(P_F)$$

From Empty Stack to Final State

The specification of P_f is asfollows:

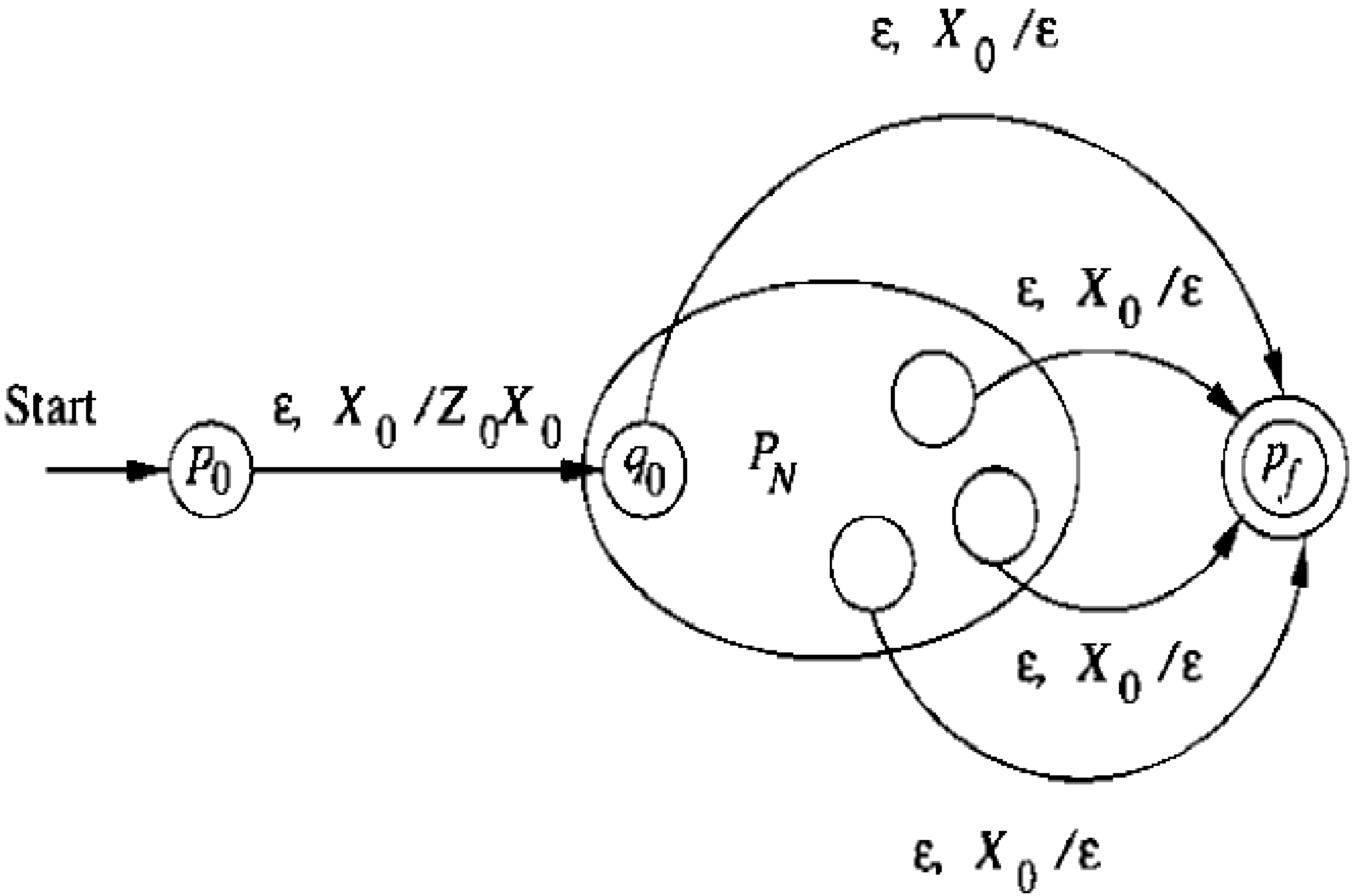
$$Q_f = Q_n \cup \{p_0, p_f\}.$$

$$\Gamma_f = \Gamma_n \cup \{X_0\}.$$

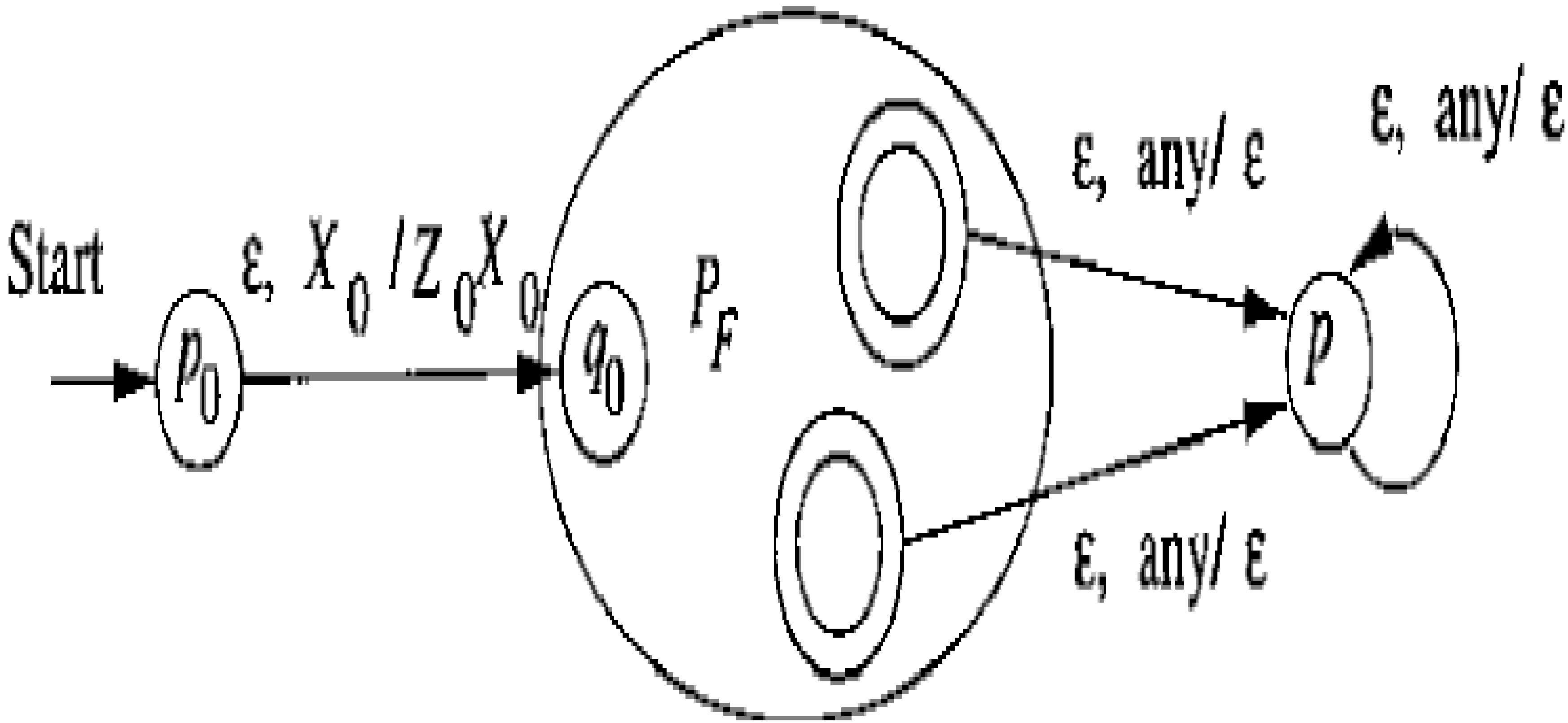
$$F_f = \{p_f\}.$$

δ_f is defined by

1. $\delta_f(p_0, \varepsilon, X_0) = \{(q_0, Z_0X_0)\}$. In its start state, P_f makes a spontaneous transition to the start state of P_n, pushing its start symbol Z_0 onto the stack.
2. For all state $q \in Q_n$, inputs $a \in \Sigma_n$ or $a = \varepsilon$, and stack symbol $Y \in \Gamma_n$, $\delta_f(q, a, Y)$ contains all the pairs in $\delta_n(q, a, Y)$.
3. In addition to rule (2), $\delta_f(q, \varepsilon, X_0)$ contains (p_f, ε) for every state $q \in Q_n$.



ε, X_0 /ε
ε, X_0 /ε
Start
ε, X_0 /Z_0X_0
p_0
q_0
P_N
p_f
ε, X_0 /ε
ε, X_0 /ε



Start
ε, X_0 / Z_0X_0
p_0
q_0
P_F
p
ε, any/ ε
ε, any/ ε
ε, any/ ε

Problems

Q)Convert the following CFG into CNF & construct PDA for the same.

$S \rightarrow 0A1 \mid 0BA$

$A \rightarrow S01 \mid 0$

$B \rightarrow 1B \mid 1$

Nov-2017 8 Marks

Design a PDA which accepts only odd number of a's over $\Sigma = \{a, b\}$.

Simulate PDA for the string "aabab".

(9 Marks Nov-2015)

Review

- PDA Examples
- CFG and PDA Equivalency
- Conversion of CFG to PDA
-

GATE Question

Question : The language $L = \{ 0^i12^i \mid i \geq 0 \}$ over the alphabet {0, 1, 2} is:
A. Not recursive
B. deterministic CFL
C. Is regular
D. Is CFL but not deterministic CFL.

Question : Consider the following languages:
$L1 = \{ 0^n1^n \mid n \geq 0 \}$
$L2 = \{ wcw^r \mid w \, \varepsilon \{a,b\}^* \}$
$L3 = \{ ww^r \mid w \, \varepsilon \{a,b\}^* \}$
Which of these languages are deterministic context-free languages?
A. None of the languages
B. Only L1
C. Only L1 and L2
D. All three languages

Question : Consider the language L1,L2,L3 as given below.
$L1 = \{ a^mb^n \mid m, n >= 0 \}$
$L2 = \{ a^nb^n \mid n >= 0 \}$
$L3 = \{ a^nb^nc^n \mid n >= 0 \}$
Which of the following statements is NOT TRUE?
A. Push Down Automata (PDA) can be used to recognize L1 and L2
B. L1 is a regular language
C. All the three languages are context free

Closure Properties
of
Context-Free languages

Closure Property

- CFLs are closed under:
  - Union
  - Concatenation
  - Kleene closure operator
  - Substitution
  - Homomorphism, inverse homomorphism
  - reversal

- CFLs are *not* closed under:
  - Intersection
  - Difference
  - Complementation

Note: Reg languages are closed under these operators

Union

Context-free languages
are closed under: **Union**

L_1 is context free

L_2 is context free

$\Rightarrow$ $L_1 \cup L_2$ is context-free

Example

Language | **Grammar**

$$L_1 = \{a^n b^n\}$$

$$S_1 \to aS_1b \mid \lambda$$

$$L_2 = \{ww^R\}$$

$$S_2 \to aS_2a \mid bS_2b \mid \lambda$$

Union

$$L = \{a^n b^n\} \cup \{ww^R\}$$

$$S \to S_1 \mid S_2$$

In general:

For context-free languages $L_1, \; L_2$

with context-free grammars $G_1, \; G_2$

and start variables S_1, S_2

The grammar of the **union** $L_1 \cup L_2$

has new start variable S

and additional production $S \rightarrow S_1 \mid S_2$

Concatenation

Context-free languages
are closed under: **Concatenation**

L_1 is context free

L_2 is context free

$\Rightarrow$ $L_1 L_2$ is context-free

Example

Language	**Grammar**
$L_1 = \{a^n b^n\}$	$S_1 \rightarrow aS_1 b \mid \lambda$
$L_2 = \{ww^R\}$	$S_2 \rightarrow aS_2 a \mid bS_2 b \mid \lambda$

Concatenation

$$L = \{a^n b^n\}\{ww^R\}$$

$$S \rightarrow S_1 S_2$$

In general:

For context-free languages $L_1, \; L_2$
with context-free grammars $G_1, \; G_2$
and start variables S_1, S_2

The grammar of the **concatenation** $L_1 L_2$
has new start variable S
and additional production $S \to S_1 S_2$

Star Operation

Context-free languages are closed under: **Star-operation**

L is context free $\Rightarrow$ L^* is context-free

Example

Language

Grammar

$$L = \{a^n b^n\}$$

$$S \to aSb \mid \lambda$$

Star Operation

$$L = \{a^n b^n\}^*$$

$$S_1 \to SS_1 \mid \lambda$$

In general:

For context-free language L

with context-free grammar G

and start variable S

The grammar of the **star operation** L^*

has new start variable S_1

and additional production $S_1 \rightarrow SS_1 \mid \lambda$

Negative Properties of Context-Free Languages

Intersection

Context-free languages
are **not** closed under: **intersection**

L_1 is context free

L_2 is context free

$L_1 \cap L_2$

not necessarily context-free

Example

$$L_1 = \{a^n b^n c^m\}$$

Context-free:

$$S \to AC$$

$$A \to aAb \mid \lambda$$

$$C \to cC \mid \lambda$$

$$L_2 = \{a^n b^m c^m\}$$

Context-free:

$$S \to AB$$

$$A \to aA \mid \lambda$$

$$B \to bBc \mid \lambda$$

Intersection

$$L_1 \cap L_2 = \{a^n b^n c^n\}$$ **NOT** context-free

Complement

Context-free languages
are **<u>not</u>** closed under: **complement**

L is context free $\Rightarrow$ $\overline{L}$ **<u>not</u>** necessarily context-free

Example

$$L_1 = \{a^n b^n c^m\}$$

$$L_2 = \{a^n b^m c^m\}$$

Context-free:

$$S \rightarrow AC$$

$$A \rightarrow aAb \mid \lambda$$

$$C \rightarrow cC \mid \lambda$$

Context-free:

$$S \rightarrow AB$$

$$A \rightarrow aA \mid \lambda$$

$$B \rightarrow bBc \mid \lambda$$

Complement

$$\overline{\overline{L_1} \cup \overline{L_2}} = L_1 \cap L_2 = \{a^n b^n c^n\}$$

NOT context-free

Pumping Lemma
for
Context-free Languages

Return of the Pumping Lemma !!

Think of languages that cannot be CFL

== think of languages for which a stack will not be enough

e.g., the language of strings of the form ww

Why pumping lemma?

A result that will be useful inproving languages that *are not* CFLs
(just like we did for regular languages)

But before we prove the pumping lemma for CFLs ….
Let us first prove an important property about parse trees

The Pumping Lemma:

For any infinite context-free language L

there exists an integer m such that

for any string $w \in L,\quad |w| \geq m$

we can write $w = uvxyz$

with lengths $|vxy| \leq m$ and $|vy| \geq 1$

and it must be that:

$$uv^i xy^i z \in L, \quad \text{for all } i \geq 0$$

Applications
of
The Pumping Lemma

Non-context free languages

$$\{a^n b^n c^n : n \geq 0\}$$

Context-free languages

$$\{a^n b^n : n \geq 0\}$$

Theorem: The language

$$L = \{a^n b^n c^n : n \geq 0\}$$

is **not** context free

Proof: Use the Pumping Lemma for context-free languages

$$L = \{a^n b^n c^n : n \geq 0\}$$

Assume for contradiction that L is context-free

Since L is context-free and infinite we can apply the pumping lemma

$$L = \{a^n b^n c^n : n \geq 0\}$$

Let m be the critical length of the pumping lemma

Pick any string $w \in L$ with length $|w| \geq m$

We pick: $w = a^m b^m c^m$

$$L = \{a^n b^n c^n : n \geq 0\}$$

$$w = a^m b^m c^m$$

From pumping lemma:

we can write: $w = uvxyz$

with lengths $|vxy| \leq m$ and $|vy| \geq 1$

$$L = \{a^n b^n c^n : n \geq 0\}$$

$$w = a^m b^m c^m$$

$$w = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

Pumping Lemma says:

$uv^i xy^i z \in L$ for all $i \geq 0$

$$L = \{a^n b^n c^n : n \geq 0\}$$

$$w = a^m b^m c^m$$

$$w = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

We examine **all** the possible locations of string vxy in w

$$S \to ABE \mid bBd$$

$$A \to Aa \mid a$$

$$B \to bSD \mid cc$$

$$D \to Dd \mid d$$

$$E \to eE \mid e$$

$$L = \{a^n b^n c^n : n \geq 0\}$$

$$w = a^m b^m c^m$$

$$w = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

Case 1: vxy is in a^m

$$\overbrace{\underbrace{a\ldots a}_{u}\,\underbrace{a\ldots a}_{vxy}\,a\ldots a}^{m}\,\overbrace{bbb\ldots bbb}^{m}\,\overbrace{ccc\ldots ccc}^{m}$$

$$L = \{a^n b^n c^n : n \geq 0\}$$

$$w = a^m b^m c^m$$

$$w = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

$$v = a^{k_1} \qquad y = a^{k_2} \qquad k_1 + k_2 \geq 1$$

$$\overbrace{\underbrace{a...a}_{u}\underbrace{a...aa...a}_{v}\underbrace{a...aa...a}_{x}\underbrace{a...a}_{y}}^{m}\underbrace{\overbrace{bbb...bbb}^{m}\overbrace{ccc...ccc}^{m}}$$

$$L = \{a^n b^n c^n : n \geq 0\}$$

$$w = a^m b^m c^m$$

$$w = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

$$v = a^{k_1} \qquad y = a^{k_2} \qquad k_1 + k_2 \geq 1$$

$$\overbrace{\underbrace{a...a}_{u}\underbrace{a...a}_{v^2}\underbrace{a...a}_{x}\underbrace{a...a}_{y^2}}^{m+k_1+k_2}\underbrace{\overbrace{bbb...bbb}^{m}\overbrace{ccc...ccc}^{m}}$$

$$L = \{a^n b^n c^n : n \geq 0\}$$

$$w = a^m b^m c^m$$

$$w = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

From Pumping Lemma: $uv^2xy^2z \in L$

$$k_1 + k_2 \geq 1$$

However: $uv^2xy^2z = a^{m+k_1+k_2}b^mc^m \notin L$

Contradiction!!!

$$L = \{a^n b^n c^n : n \geq 0\}$$

$$w = a^m b^m c^m$$

$$w = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

Case 2: vxy is in b^m

Similar to case 1

$$\overbrace{aaa...aaa}^{m}\overbrace{b...bb...bb...b}^{m}\overbrace{ccc...ccc}^{m}$$

$$\underbrace{aaa...aaab...bb...}_{u}\underbrace{bb...b}_{vxy}\underbrace{ccc...ccc}$$

$$L = \{a^n b^n c^n : n \geq 0\}$$

$$w = a^m b^m c^m$$

$$w = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

Case 3: vxy is in c^m

Similar to case 1

$$\underbrace{\overbrace{aaa\ldots aaa}^{m}\overbrace{bbb\ldots bbb}^{m}c\ldots cc\ldots cc\ldots c}_{u}\;\underbrace{\qquad\qquad}_{vxy}\;\underbrace{\qquad\qquad}_{}$$

$$L=\{a^n b^n c^n : n \geq 0\}$$

$$w=a^m b^m c^m$$

$$w=uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

Case 4: vxy overlaps a^m and b^m

$$\overbrace{a...aa...aa}^{m}\overbrace{b...bb...bb}^{m}\overbrace{ccc...ccc}^{m}$$

$$\underbrace{a...a}_{u}\;\underbrace{a...aab...bb...b}_{vxy}\;\underbrace{bbccc...ccc}$$

$$L = \{a^n b^n c^n : n \geq 0\}$$

$$w = a^m b^m c^m$$
$$w = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

Sub-case 1: x contains only a

y contains only b

$$\overbrace{a...aa...aa...a}^{m}\overbrace{b...bb...bb...b}^{m}\overbrace{ccc...ccc}^{m}$$

$$\underbrace{a...a}_{u}\underbrace{a...a}_{v}\underbrace{a...ab...bb...b}_{x}\underbrace{b...b}_{y}\underbrace{ccc...ccc}_{}$$

$$L = \{a^n b^n c^n : n \geq 0\}$$

$$w = a^m b^m c^m$$

$$w = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

$$v = a^{k_1} \qquad y = a^{k_2} \qquad k_1 + k_2 \geq 1$$

$$\overbrace{a...aa...aa...a}^{m}\;\overbrace{b...bb...bb...b}^{m}\;\overbrace{ccc...ccc}^{m}$$

$$\underbrace{a...a}_{u}\underbrace{a...a}_{v}\underbrace{a...a\;b...b}_{x}\underbrace{b...b}_{y}\underbrace{b...b\;ccc...ccc}_{}$$

$$L = \{a^n b^n c^n : n \geq 0\}$$

$$w = a^m b^m c^m$$

$$w = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

$$v = a^{k_1} \qquad y = a^{k_2} \qquad k_1 + k_2 \geq 1$$

$$\overbrace{a...a\,a...a\,a...a}^{m+k_1}\ \overbrace{b...b\,b...b\,b...b}^{m+k_2}\ \overbrace{ccc...ccc}^{m}$$

$$\underbrace{a...a}_{u}\,\underbrace{a...a}_{v^2}\,\underbrace{a...a\ b...b}_{x}\,\underbrace{b...b}_{y^2}\,\underbrace{b...b\ ccc...ccc}_{}$$

$$L = \{a^n b^n c^n : n \geq 0\}$$

$$w = a^m b^m c^m$$

$$w = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

From Pumping Lemma: $uv^2xy^2z \in L$

$$k_1 + k_2 \geq 1$$

However: $uv^2xy^2z = a^{m+k_1} b^{m+k_2} c^m \notin L$

Contradiction!!!

$$L = \{a^n b^n c^n : n \geq 0\}$$

$$w = a^m b^m c^m$$

$$w = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

Sub-case 2: x. contains a and b

y. contains only b

$$\overbrace{}^{m}\qquad\qquad\overbrace{}^{m}\qquad\qquad\overbrace{}^{m}$$

$$\underbrace{a...a}_{u}\underbrace{a...ab...bb...b}_{v}\underbrace{b...b}_{x}\underbrace{b...b}_{y}\underbrace{ccc...ccc}$$

$$L = \{a^n b^n c^n : n \geq 0\}$$

$$w = a^m b^m c^m$$

$$w = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

By assumption

$$v = a^{k_1} b^{k_2} \qquad y = b^{k_3} \qquad k_1, k_2, k_3 \geq 1$$

$$\overbrace{\underbrace{a\ldots a}_{u}\,\overbrace{a\ldots a}^{k_1}}^{m}\ \overbrace{\overbrace{b\ldots b}^{k_2}\,\underbrace{b\ldots b}_{x}\,\overbrace{b\ldots b}^{k_3}\,b\ldots b}^{m}\ \overbrace{ccc\ldots ccc}^{m}$$

$$v = a\ldots a\,b\ldots b \qquad x = b\ldots b \qquad y = b\ldots b$$

$$\underbrace{a\ldots a}_{u}\ \underbrace{a\ldots a\ b\ldots b}_{v}\ \underbrace{b\ldots b}_{x}\ \underbrace{b\ldots b}_{y}\ \underbrace{b\ldots b\ ccc\ldots ccc}$$

$$L = \{a^n b^n c^n : n \geq 0\}$$

$$w = a^m b^m c^m$$

$$w = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

$$v = a^{k_1} b^{k_2} \qquad y = a^{k_3} \qquad k_1, k_2 \geq 1$$

$$m \qquad\qquad m + k_3 \qquad\qquad m$$

$$k_1 \quad k_2 \quad k_1 \quad k_2 \quad 2k_3$$

$$a...aa...ab...ba...ab...bb...bb...bb...bccc...ccc$$

$$u \qquad v^2 \qquad x \qquad y^2$$

$$L = \{a^n b^n c^n : n \geq 0\}$$

$$w = a^m b^m c^m$$

$$w = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

From Pumping Lemma: $uv^2xy^2z \in L$

$$k_1, k_2 \geq 1$$

However: $uv^2xy^2z = a^m b^{k_2} a^{k_1} b^{m+k_3} c^m \notin L$

Contradiction!!!

$$L = \{a^n b^n c^n : n \geq 0\}$$

$$w = a^m b^m c^m$$

$$w = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

Sub-case 3: v contains only a

y contains a and b

Similar to sub-case 2

m m m

a...aa...aa...aa...ab...bb...bccc...ccc

u v x y

$$L = \{a^n b^n c^n : n \geq 0\}$$

$$w = a^m b^m c^m$$
$$w = uvxyz \qquad |vxy| \leq m \qquad |vy| \geq 1$$

Case 5: vxy overlaps b^m and c^m

Similar to case 4

$$\underbrace{\overbrace{aaa...aaa}^{m}\,bbb}_{u}\overbrace{\underbrace{...bbb}\,}^{m}\!\!\!\underbrace{\overbrace{ccc...}^{m}}_{vxy}\underbrace{ccc}$$

$$L = \{a^n b^n c^n : n \geq 0\}$$

$$w = a^m b^m c^m$$
$$w = uvxyz$$

$$|vxy| \leq m \qquad\qquad |vy| \geq 1$$

Case 6: vxy overlaps a^m, b^m and c^m

Impossible!

$$\overbrace{aaa...aaa}^{m}\;\overbrace{bbb...bbb}^{m}\;\overbrace{ccc...ccc}^{m}$$

$$u \qquad\qquad vxy$$

In all cases we obtained a contradiction

Therefore: the original assumption that

$$L = \{a^n b^n c^n : n \geq 0\}$$

is context-free must be wrong

Conclusion: L is not context-free

Summary

Post Machine

Contents

- Post Machine
- Examples

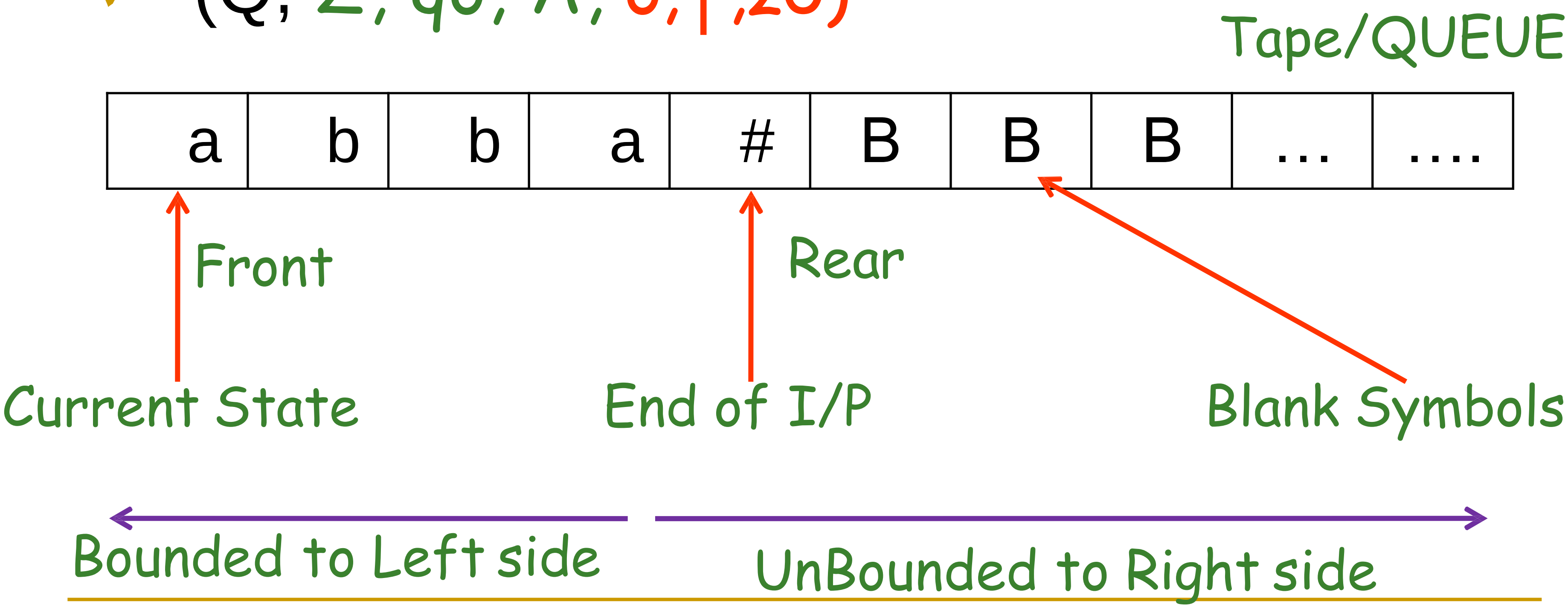
POST Machine

- **A Post machine** denoted by **PM** is a collection of five things
  1. The alphabet Σ plus the special symbol #. We generally use Σ = { a, b}.
  2. **A** linear storage location (a place where a string of symbols is kept called the **STORE or QUEUE,** which initially contains the input string. This location can be read by which we mean the leftmost character can be removed for inspection. The STORE can also be added to, which means a new character can be concatenated onto the right of whatever is there already. We allow for the possibility that characters not in Σ can be used in the **STORE,** characters from an alphabet Γ called the store alphabet.
  3. More Powerful than FA,PDA

- **A Post machine** does not have a separate input store like FA, PDA have.
- PM has a Queue.
- While Processing a string it is assumed that the string is initially loaded into the queue.
- PM has start execution at Start State.
- If post machine halt at ACCEPT state, the input string is said to be accepted else rejected.

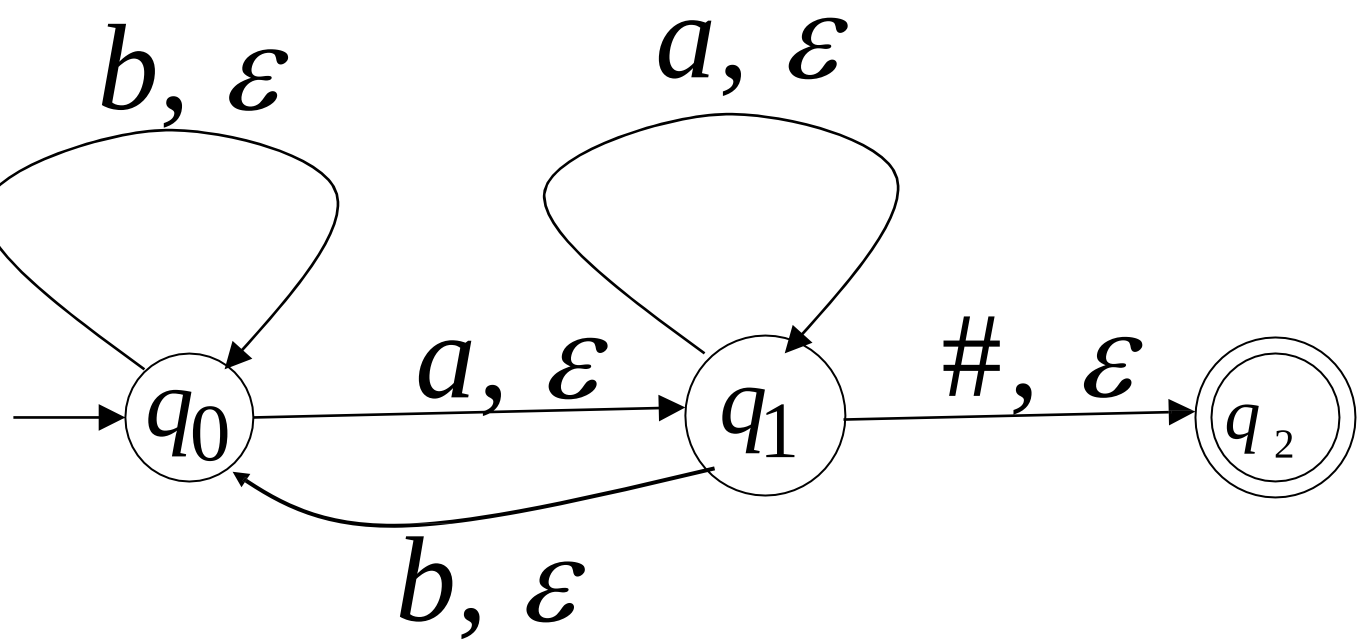
Formal Definition

- **A Post machine** =(Input alphabet,start state, Accept state, Reject State, Branching state Read, Queue/Tape alphabet, Transition function,)
- OR
- (Q, Σ, qo, A, δ, Γ,z0)



Example

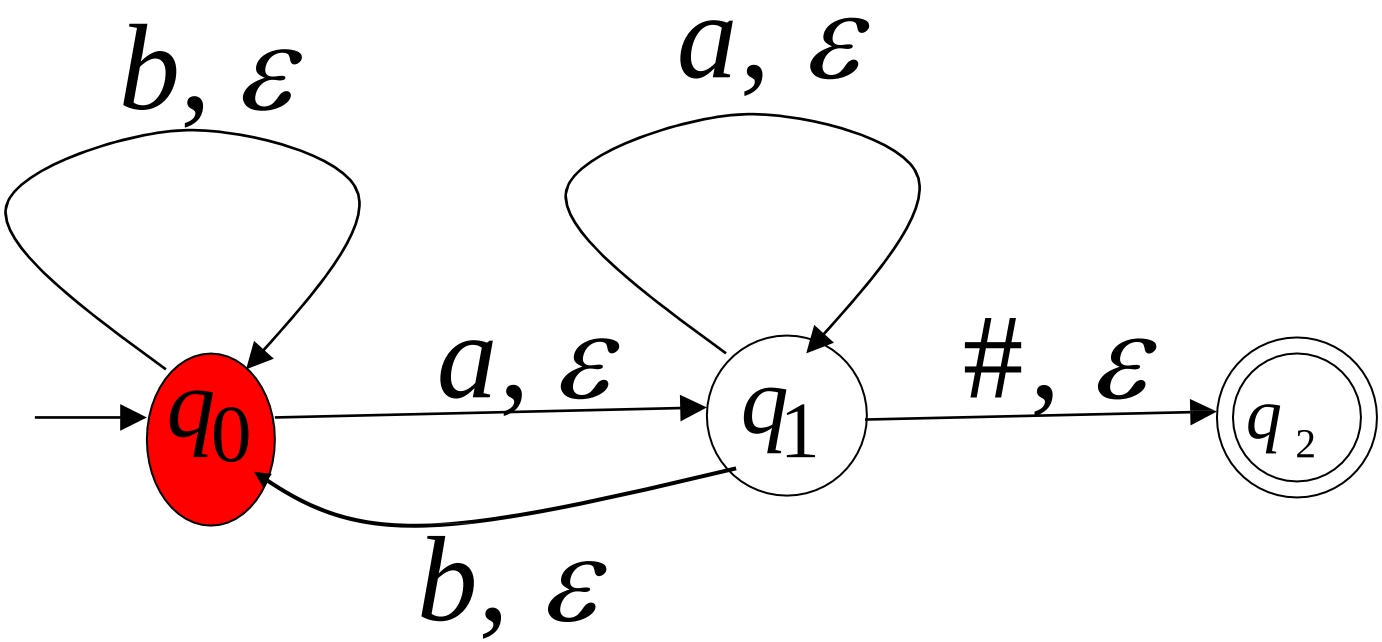
- ✓ Design a post machine that accepts the language which end by a.



a	b	b	a	#	B	B	B	…	….

Example

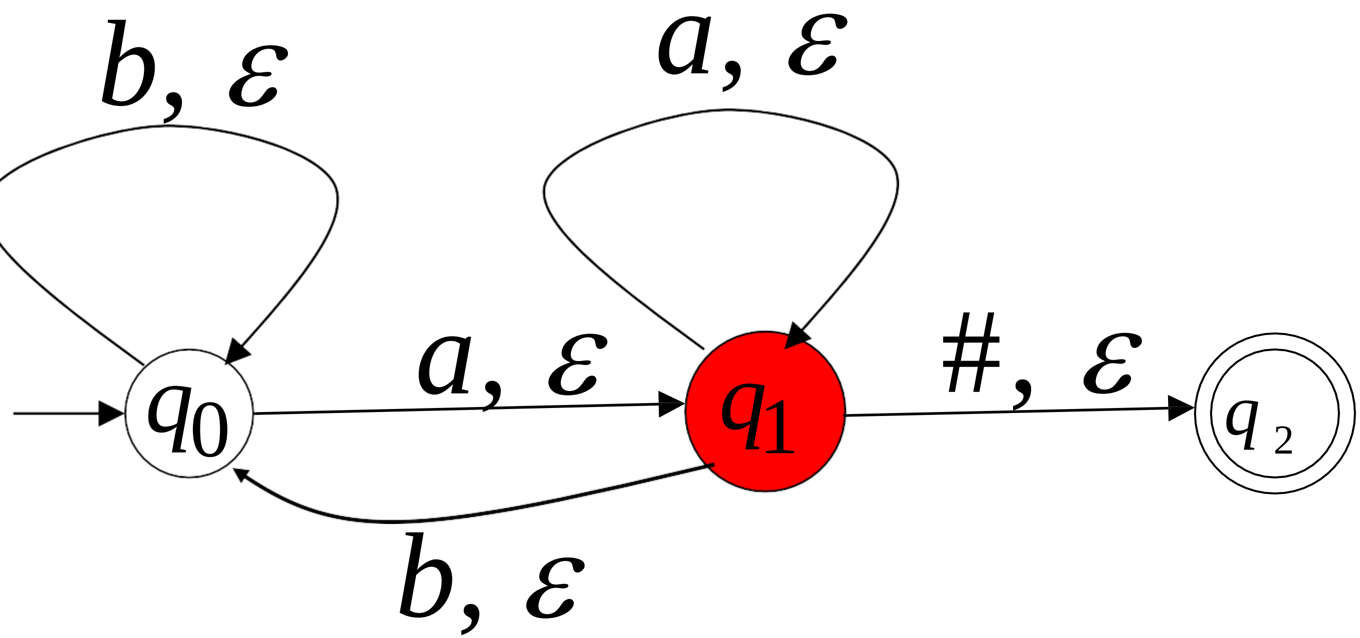
- Design a post machine that accepts the following language which end by a.



a	b	b	a	#	B	B	B	…	….

Example

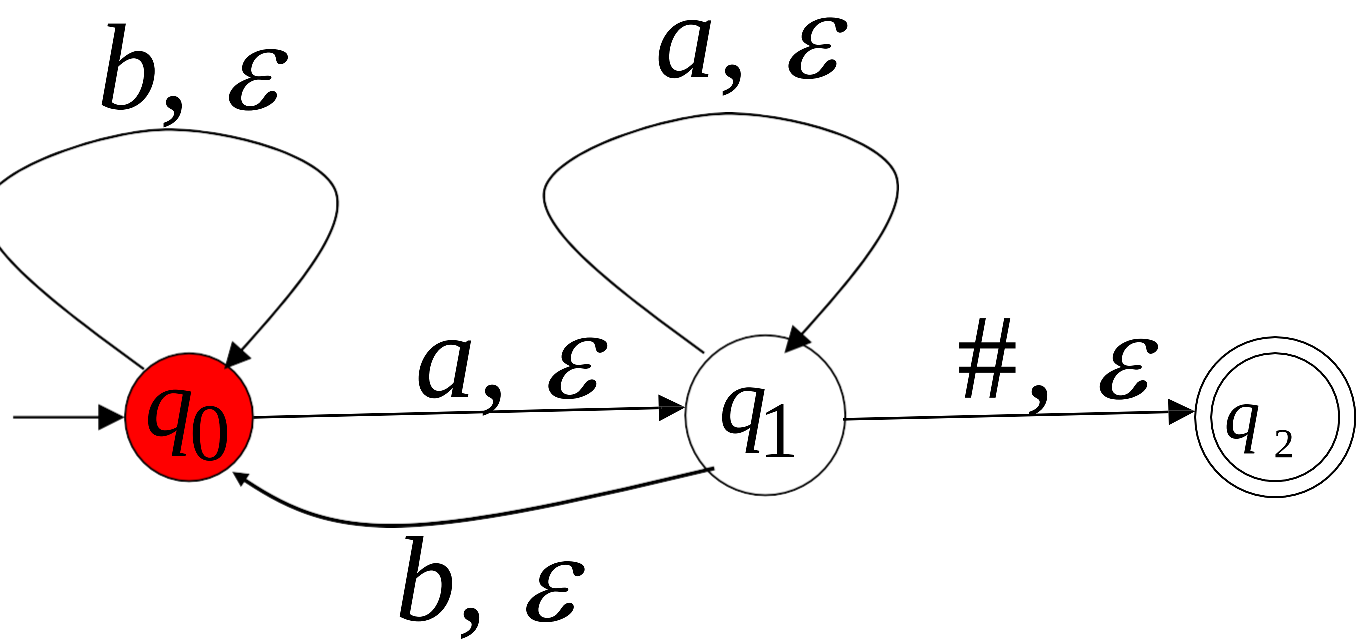
✓ Design a post machine that accepts the language which end by a.



a	b	b	a	#	B	B	B	…	….

Example

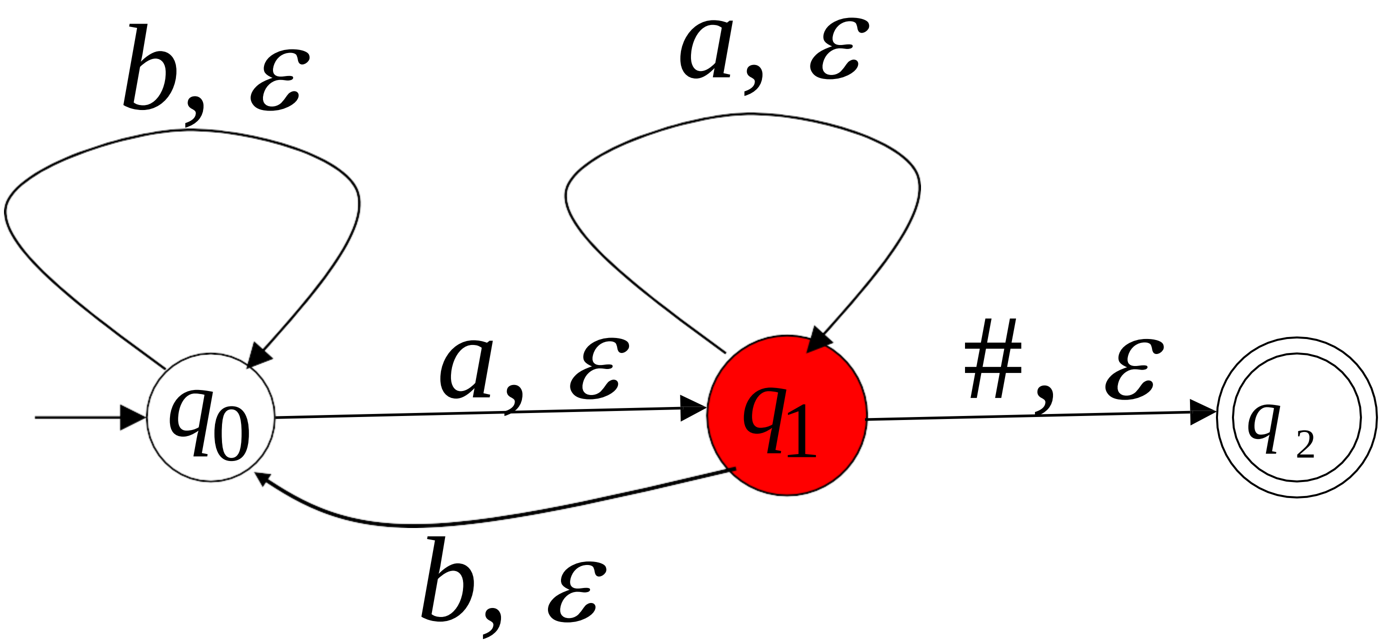
- ✓ Design a post machine that accepts the following language which end by a.



a	b	b	a	#	B	B	B	…	….

Example

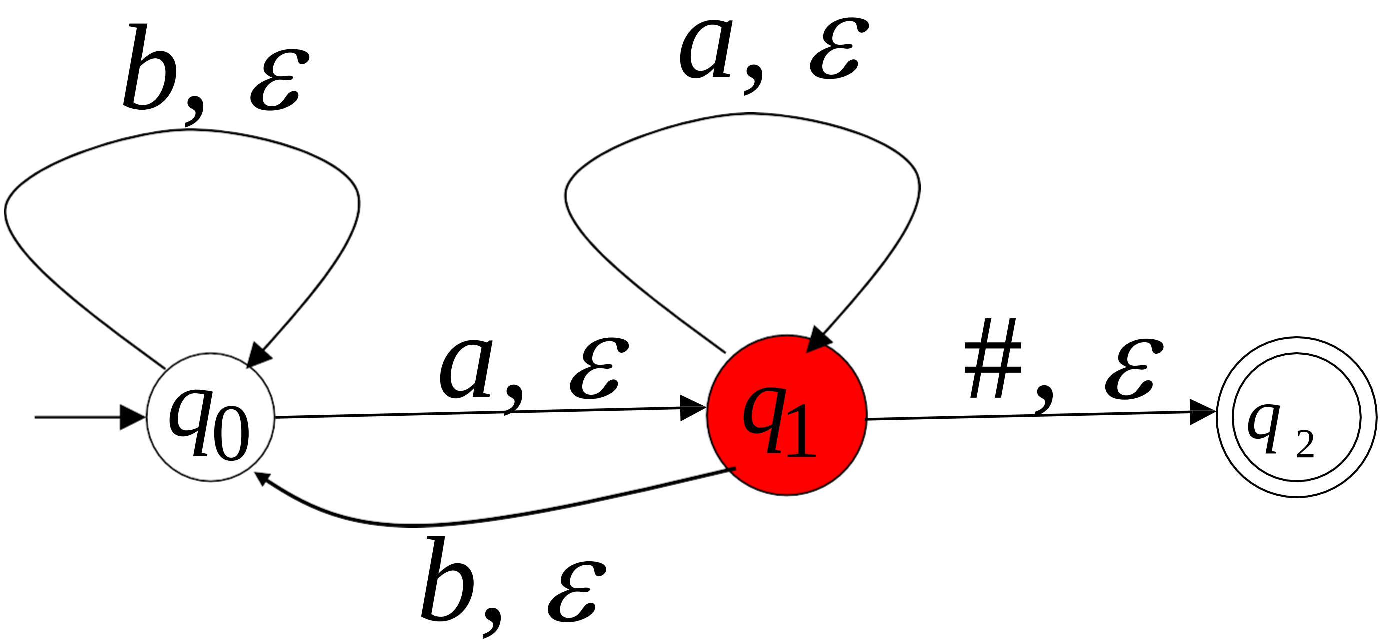
- ✓ Design a post machine that accepts the following language which end by a.



a	b	b	a	#	B	B	B	…	….

Example

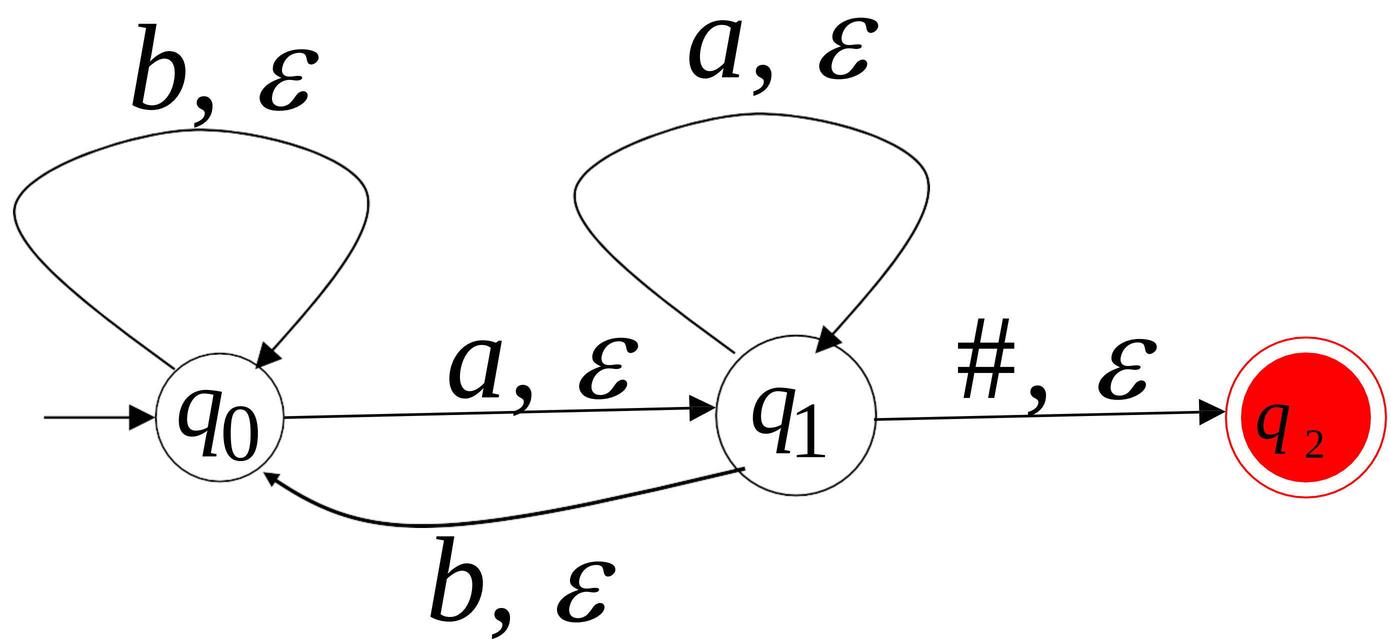
- Design a post machine that accepts the following language which end by a.



a	b	b	a	#	B	B	B	…	….

Example

- ✓ Design a post machine that accepts the following language which end by a.

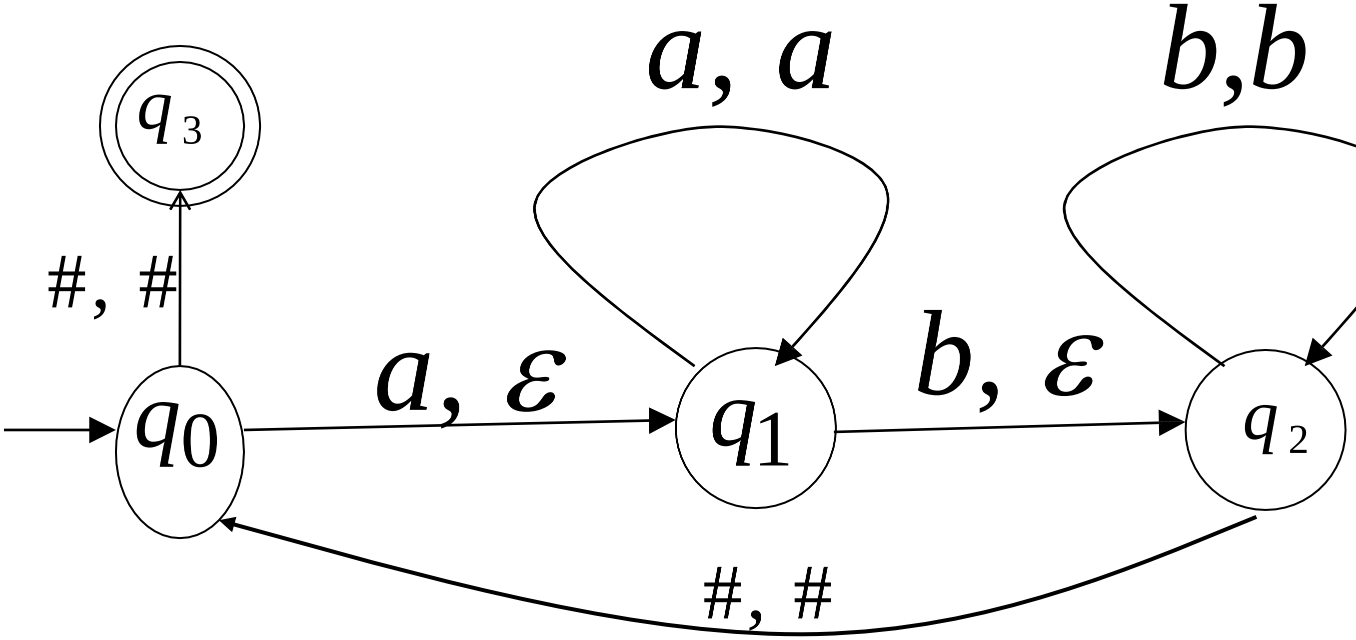


a	b	b	a	#	B	B	B	…	….

String is accepted..

Example

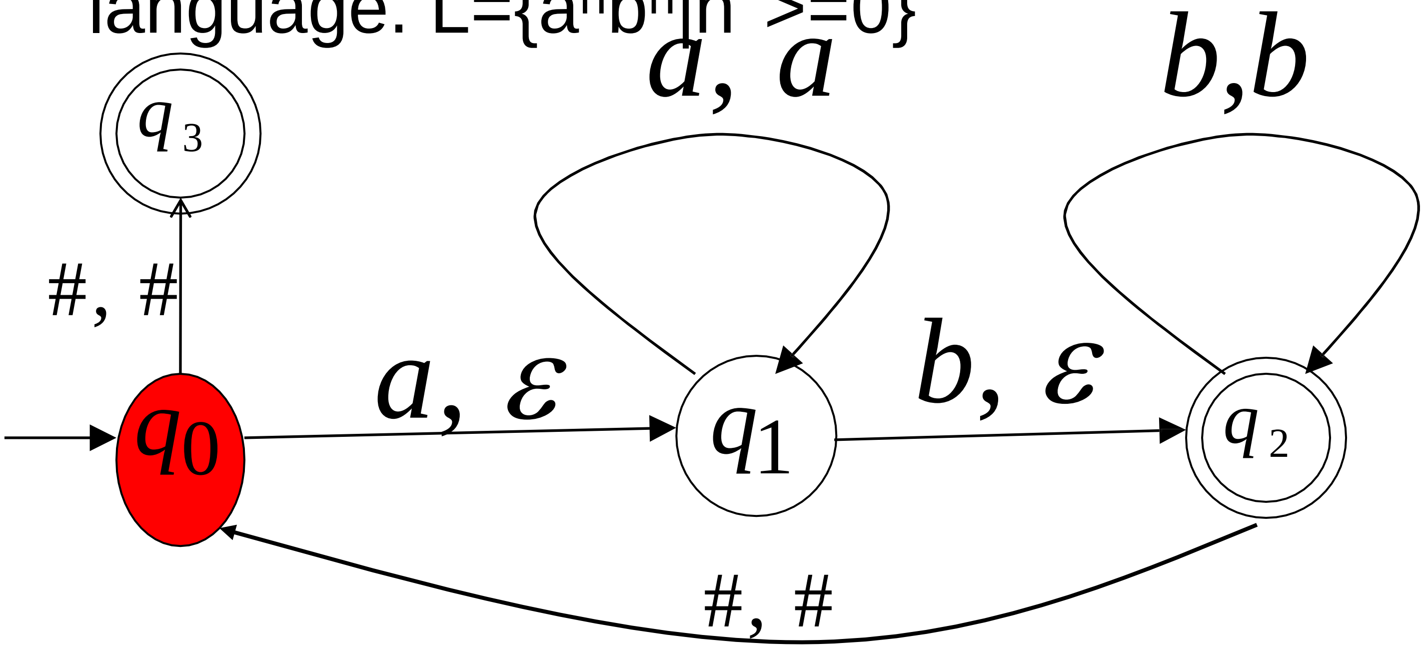
✓ Design a post machine that accepts the following language. L={$a^n b^n$|n >=0} (8 marks Nov-15)



a	a	b	b	#	.	.	.	…	….

Example

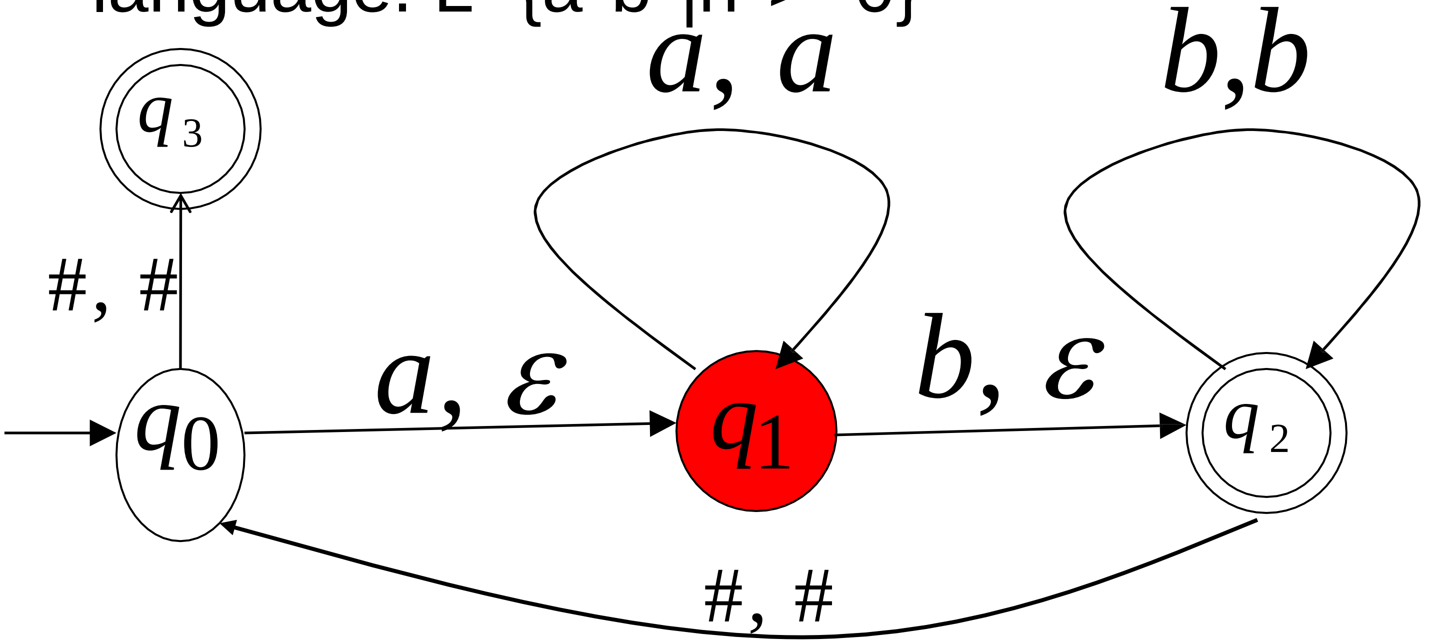
- Design a post machine that accepts the following language. $L=\{a^n b^n | n >= 0\}$ (8 marks Nov-15)

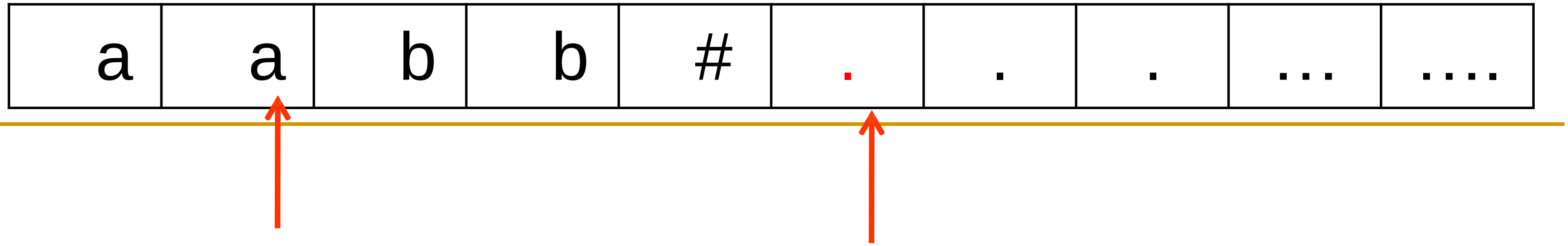


a	a	b	b	#	.	.	.	…	….

Example

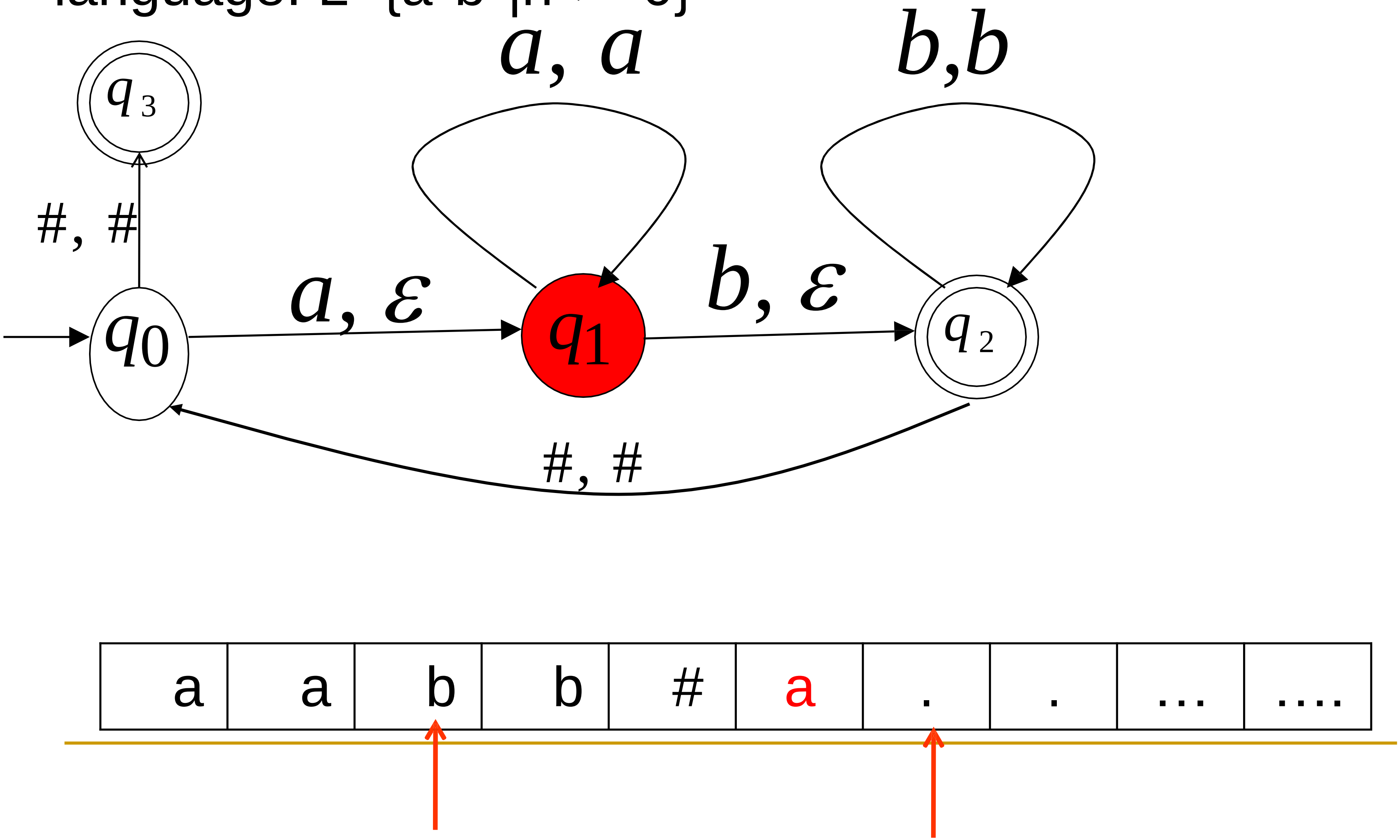
- Design a post machine that accepts the following language. L={$a^n b^n$|n >=0} (8 marks Nov-15)





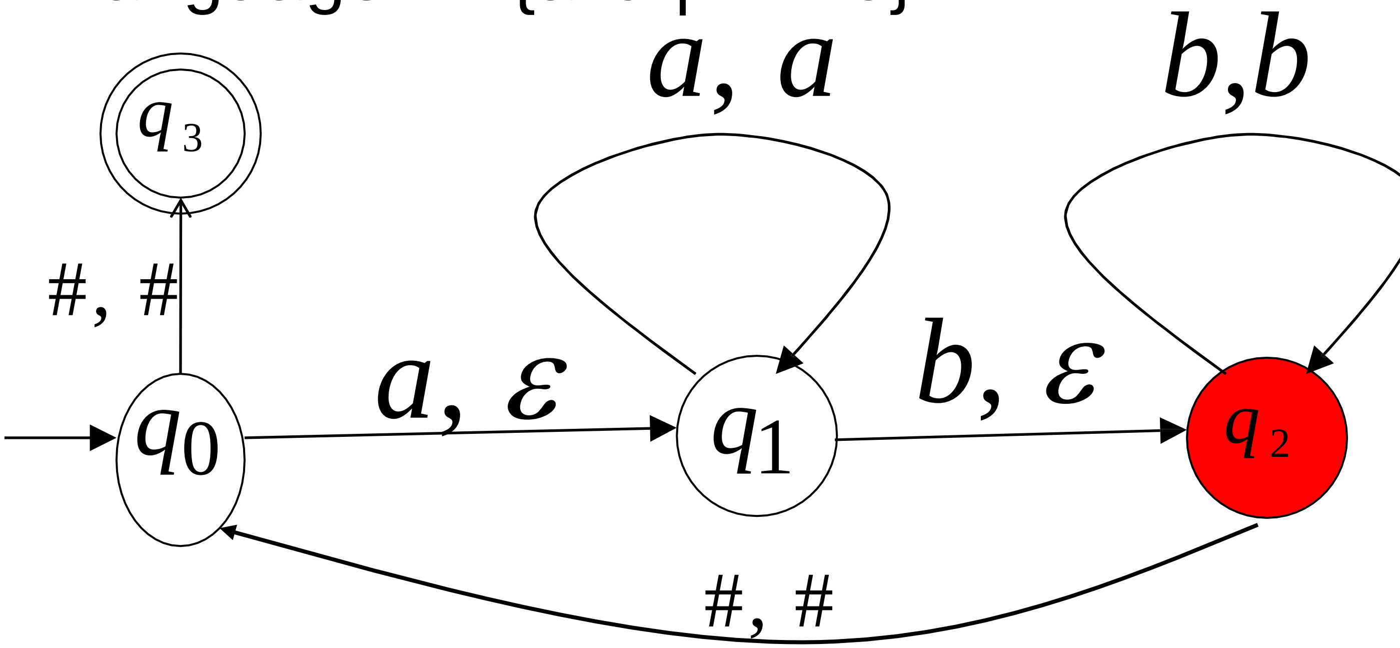
Example

- Design a post machine that accepts the following language. $L=\{a^n b^n | n >= 0\}$ (8 marks Nov-15)



Example

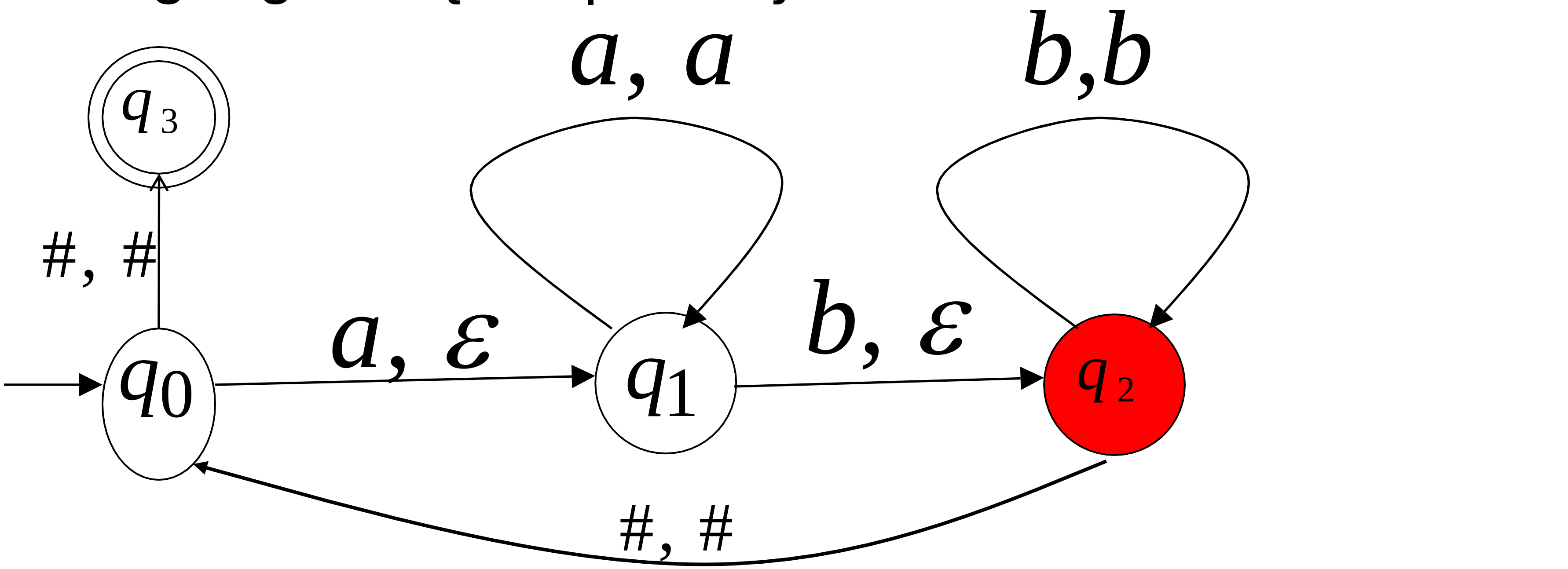
✓ Design a post machine that accepts the following language. $L=\{a^nb^n|n >=0\}$ (8 marks Nov-15)



a	a	b	b	#	a	.	.	…	….

Example

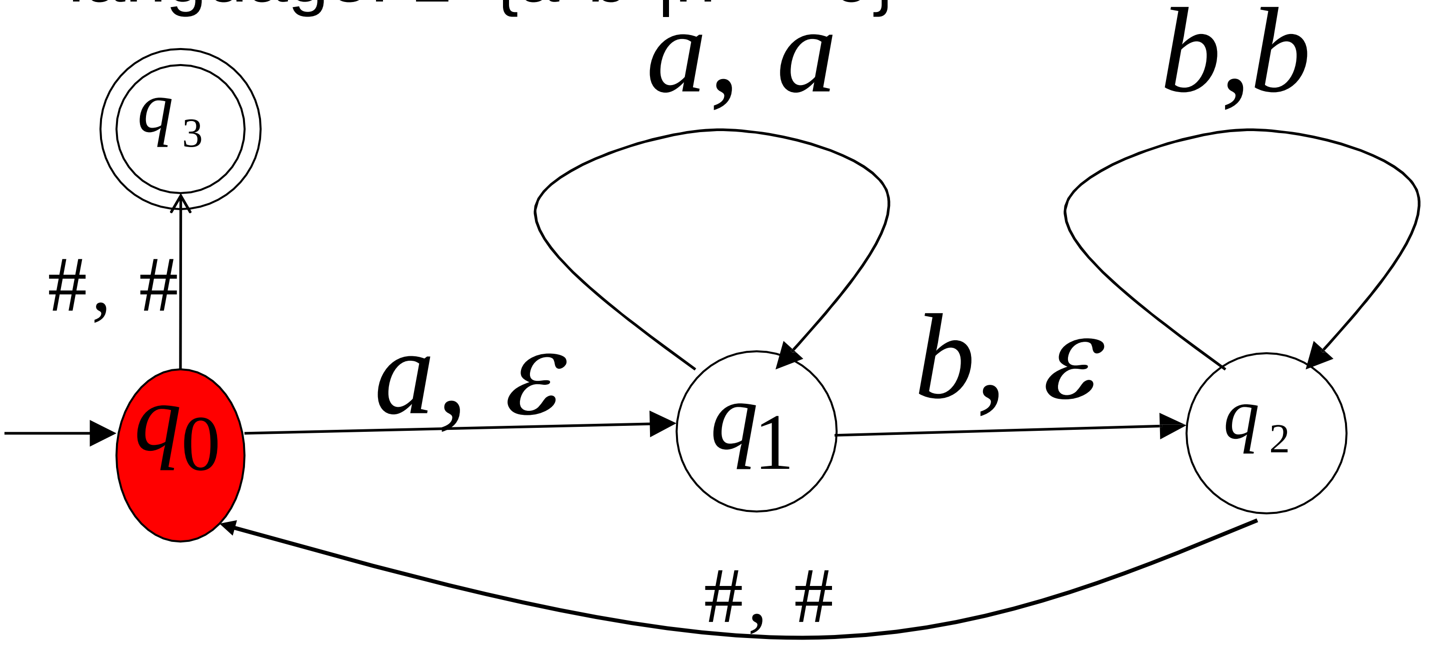
- Design a post machine that accepts the following language. $L=\{a^n b^n | n >= 0\}$ (8 marks Nov-15)



a	a	b	b	#	a	b	.	…	….

Example

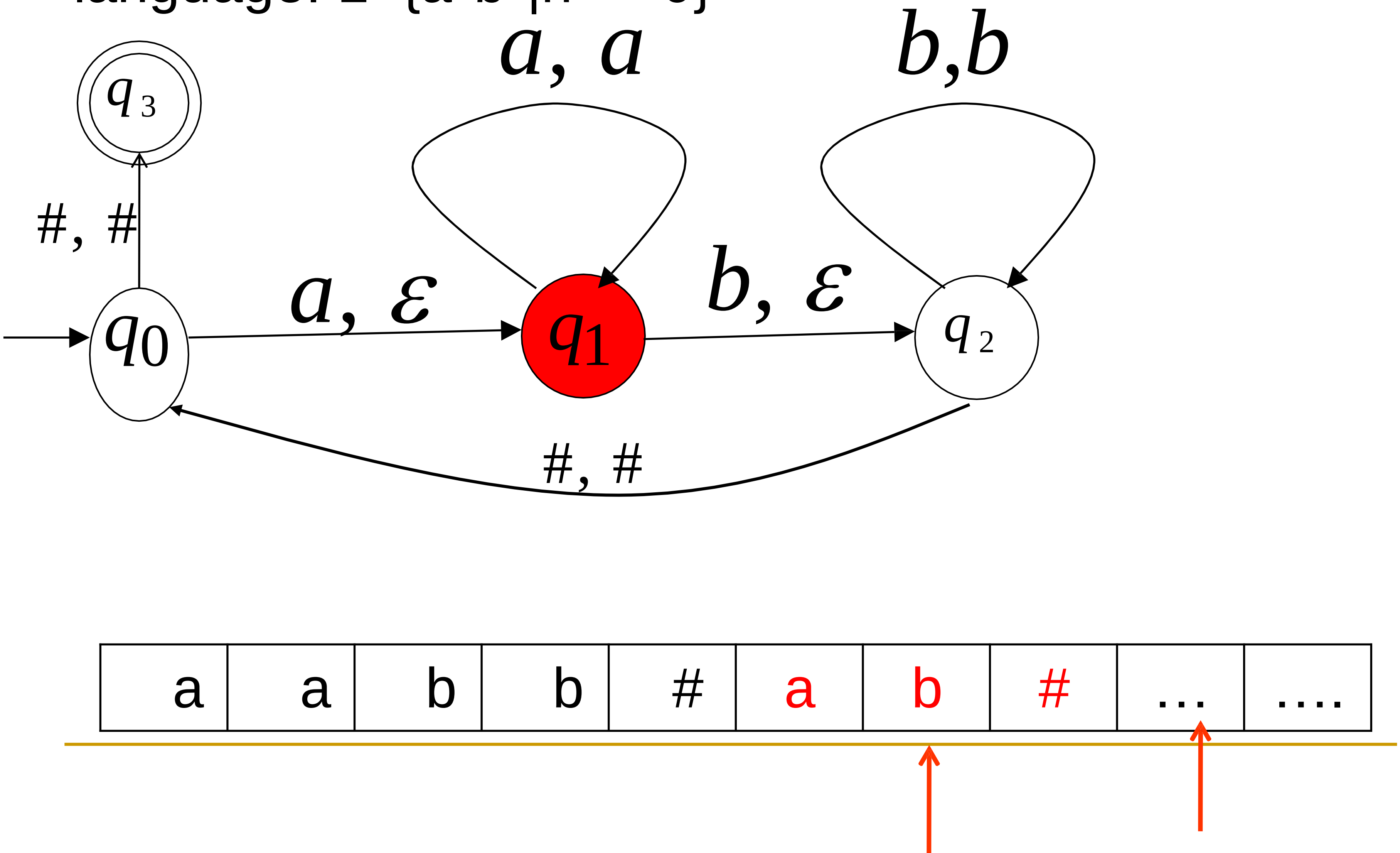
- Design a post machine that accepts the following language. $L=\{a^n b^n | n >= 0\}$ (8 marks Nov-15)



a	a	b	b	#	a	b	#	…	….

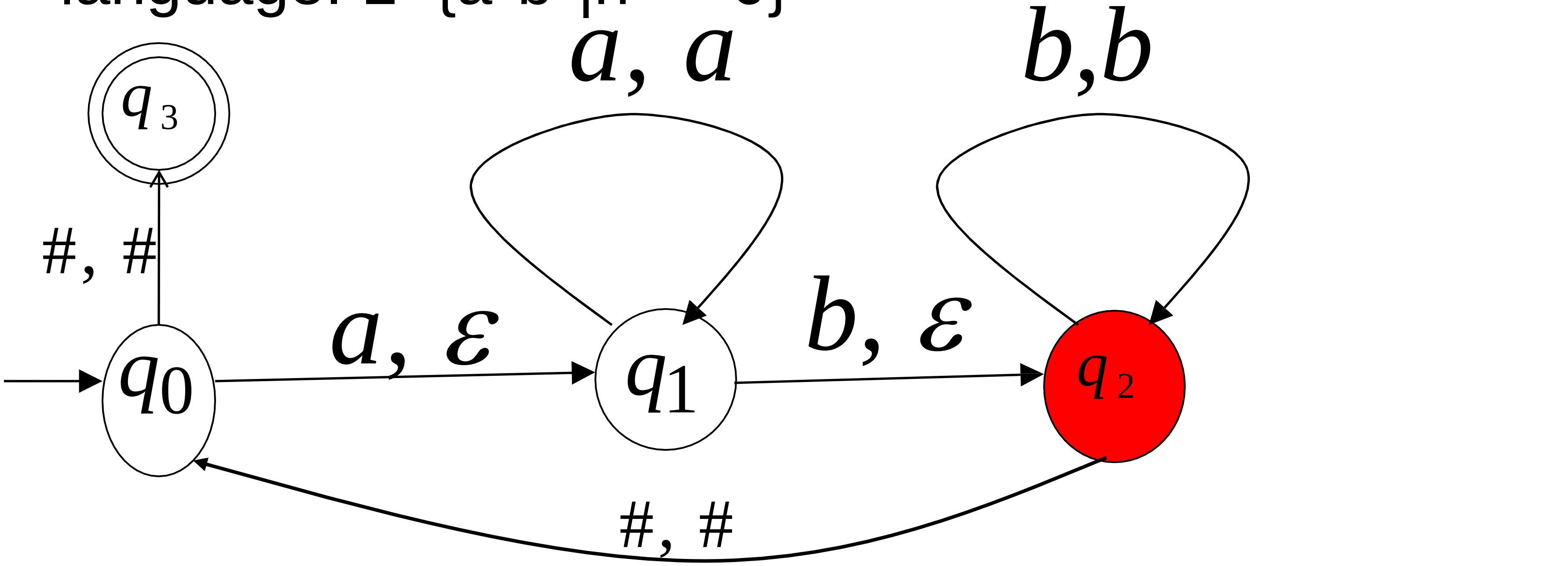
Example

- Design a post machine that accepts the following language. L={$a^n b^n$|n >=0} (8 marks Nov-15)



Example

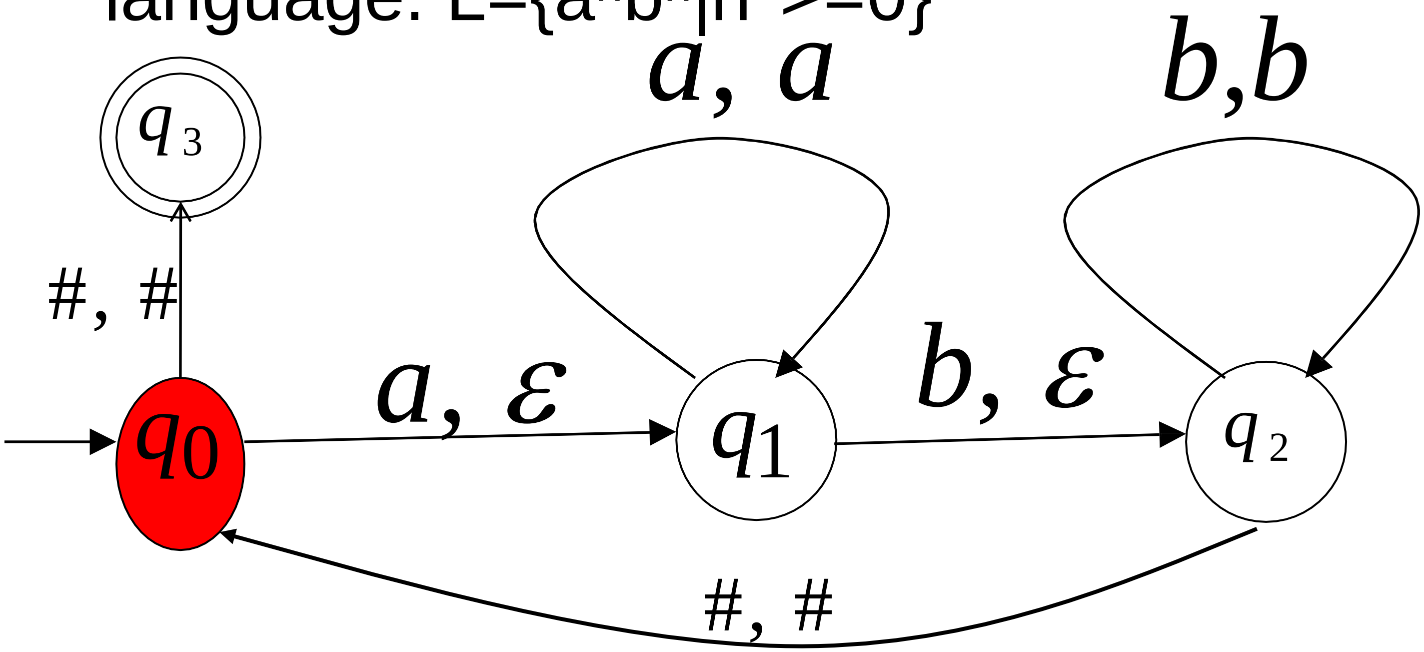
- Design a post machine that accepts the following language. $L=\{a^n b^n | n >= 0\}$ (8 marks Nov-15)



a	a	b	b	#	a	b	#	…	….

Example

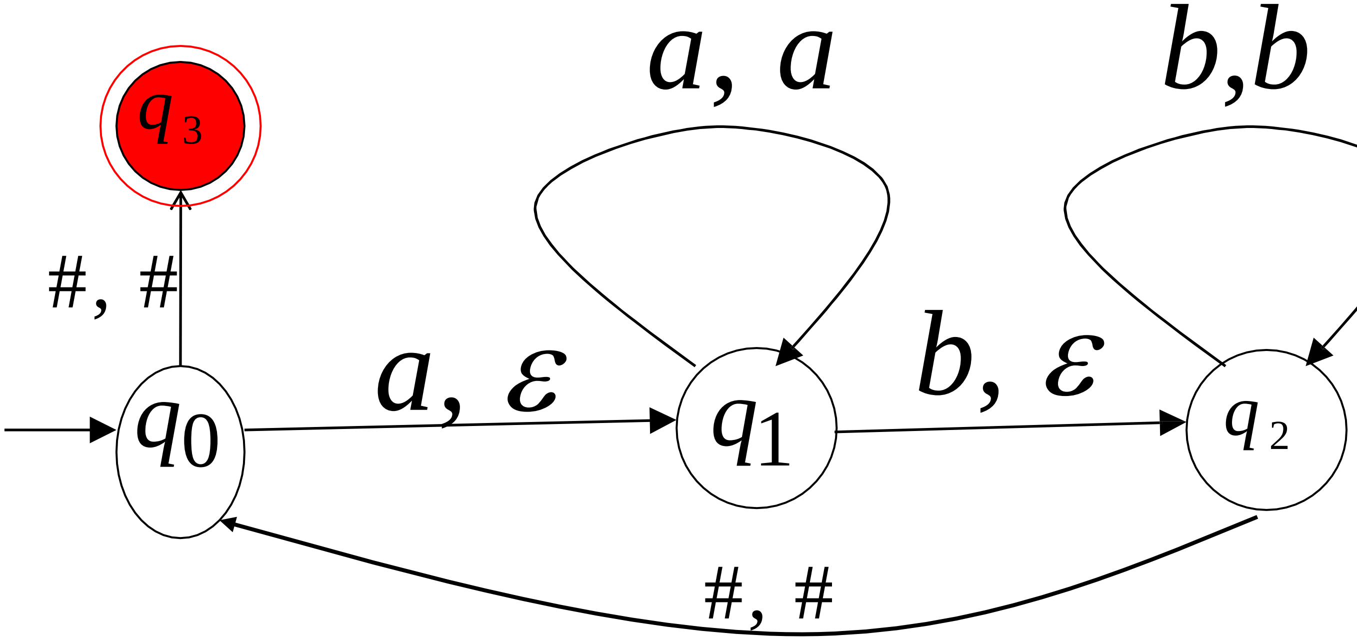
✓ Design a post machine that accepts the following (8 marks Nov-15)
language. $L=\{a^n b^n | n >= 0\}$



a	a	b	b	#	a	b	#	#	….

Example

- Design a post machine that accepts the following language. L={$a^n b^n$|n >=0} (8 marks Nov-15)

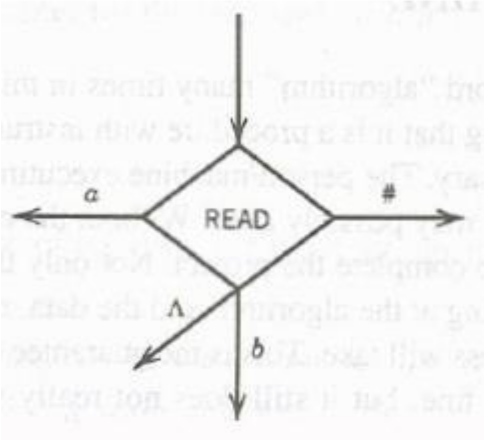


a	a	b	b	#	a	b	#	#	….

String is accepted..

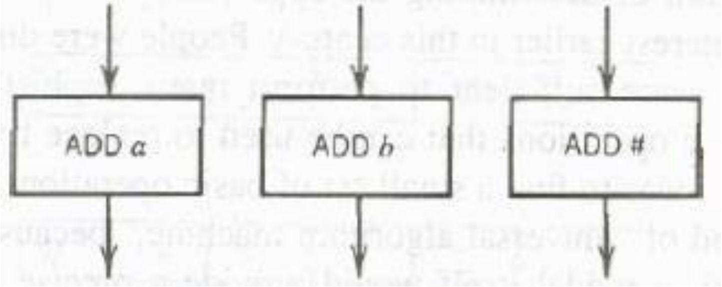
POST Machine Contd.

3. READ states, for example,



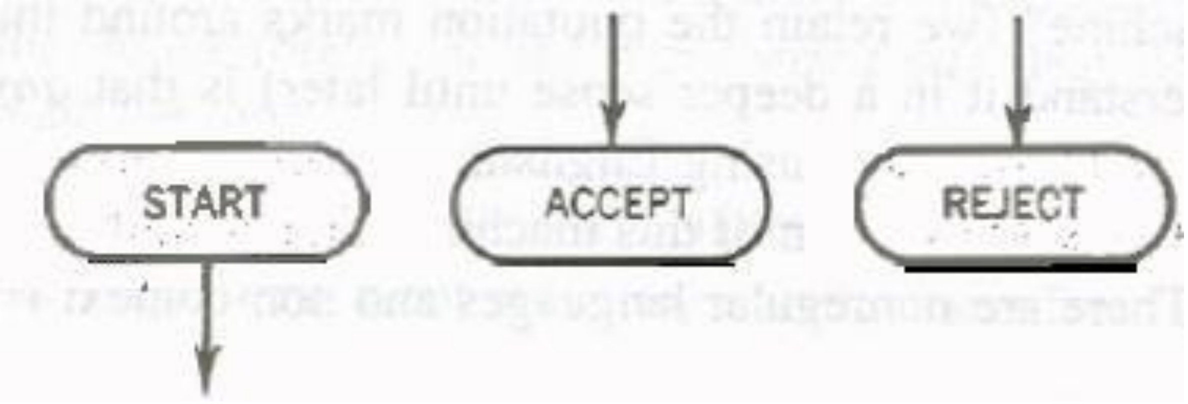
PMs **are** deterministic, so no two edges from the **READ** have the same label

4. ADD states:



POST Machine Contd.

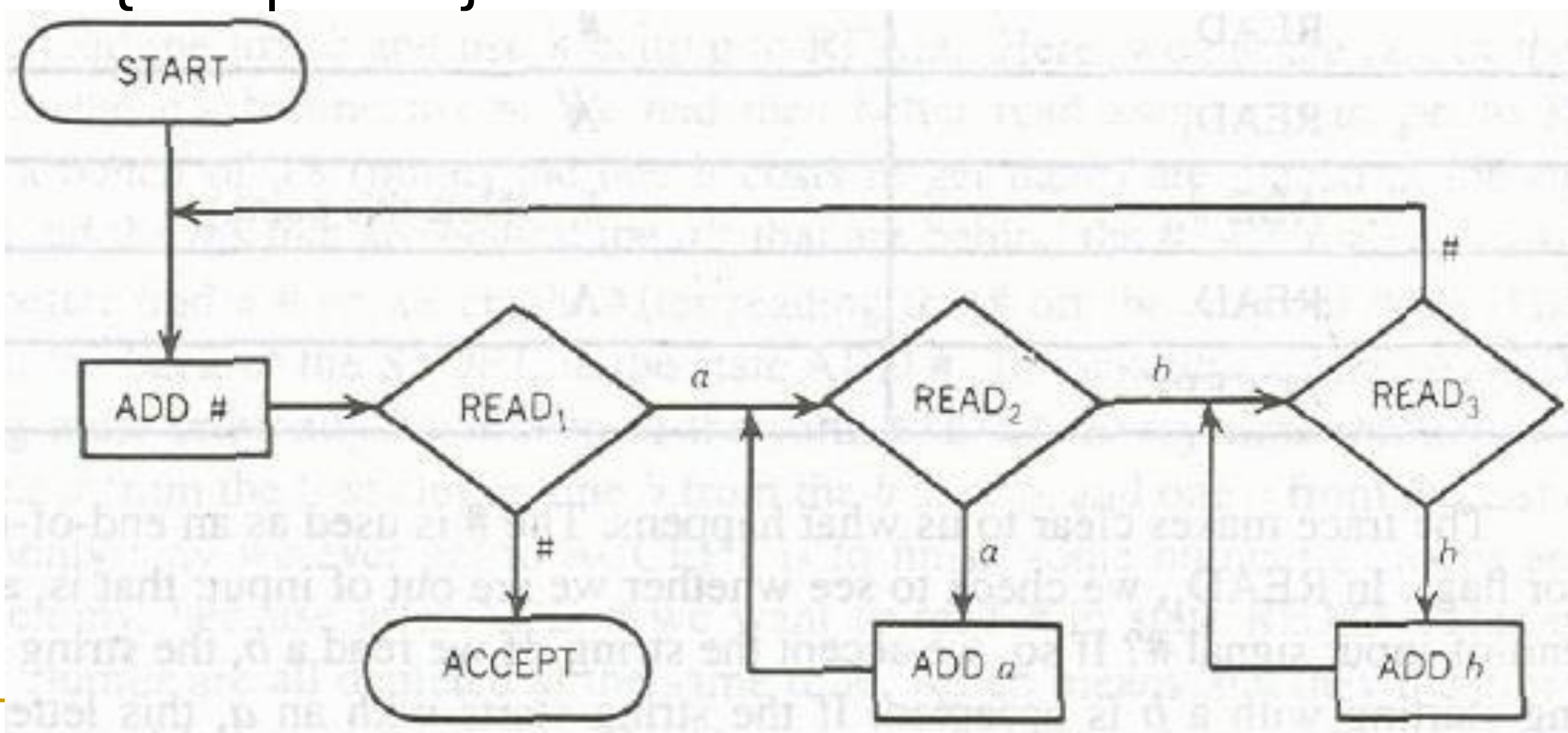
5. **A START** state (un-enterable) and some halt states called **ACCEPT and REJECT**



Example

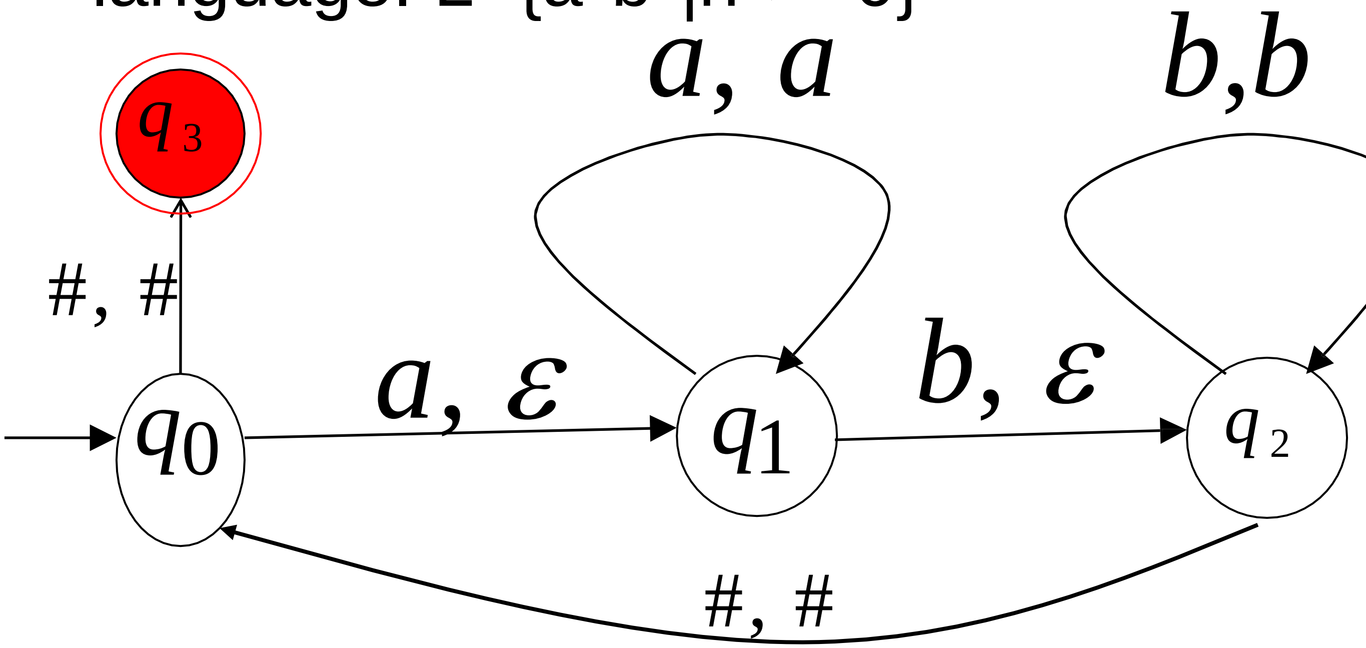
- Design a post machine that accepts the following language. (8 marks Nov-15)

$L=\{a^n b^n | n >= 0\}$



Example

- Design a post machine that accepts the following language. $L=\{a^n b^n | n >= 0\}$ (8 marks Nov-15)



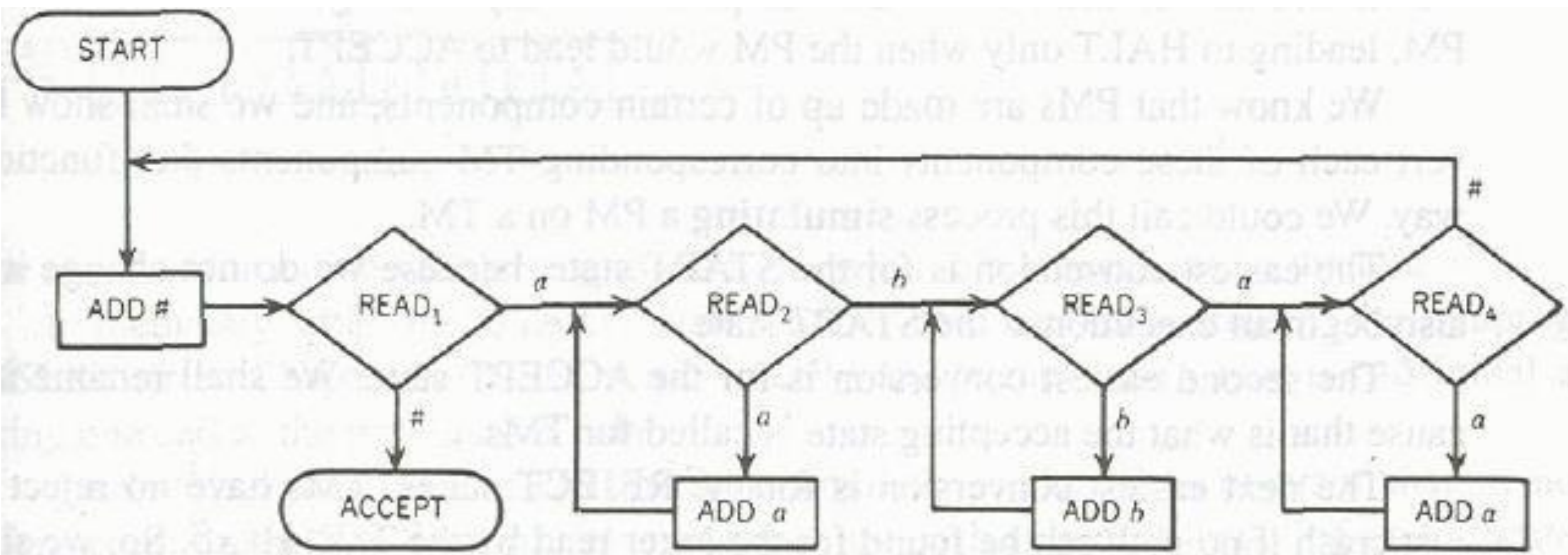
a	a	b	b	#					….

String is accepted..

Another POST Machine

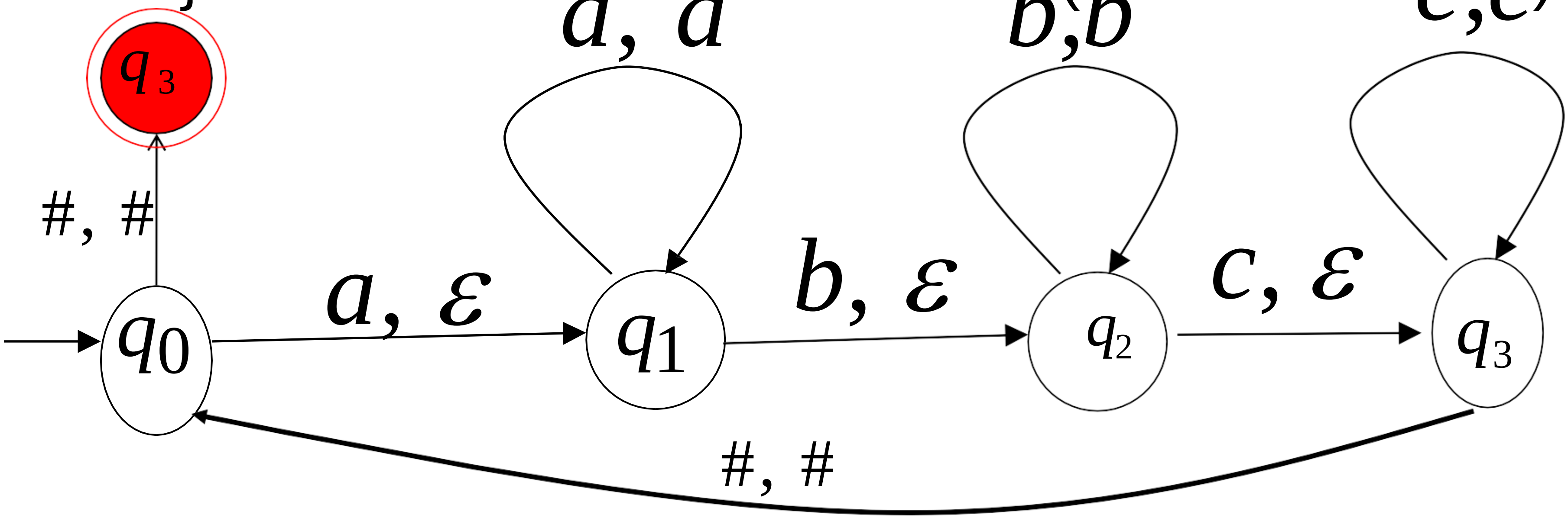
- Design a post machine that accepts the following language. (8 marks Nov-16)

$L=\{a^n b^n c^n | n >= 0\}$



Example

- Design a post machine that accepts the following language. $L=\{a^n b^n c^n | n >= 0\}$ (8 marks Nov-16)

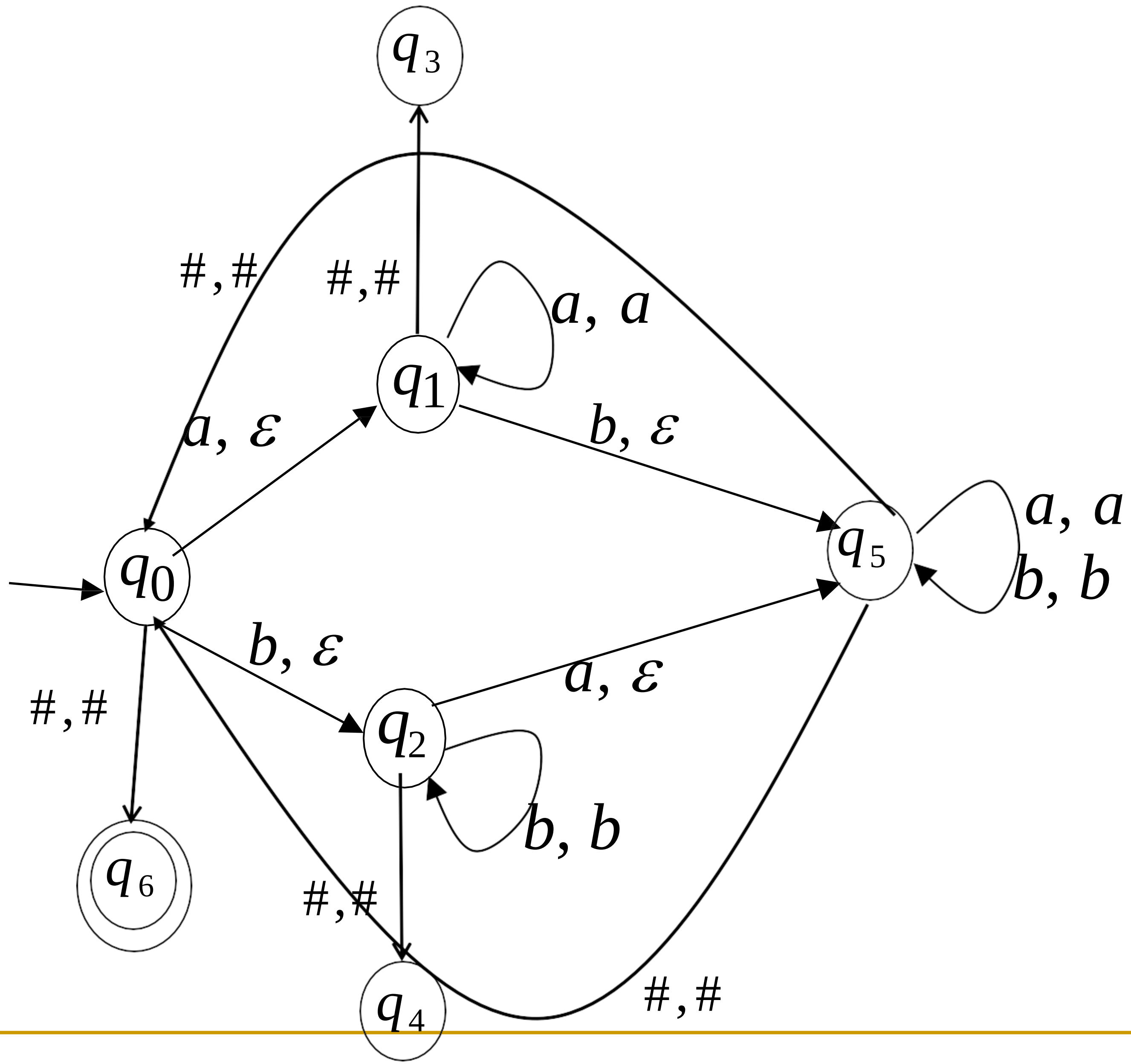


a	a	b	b	c	c	#			….

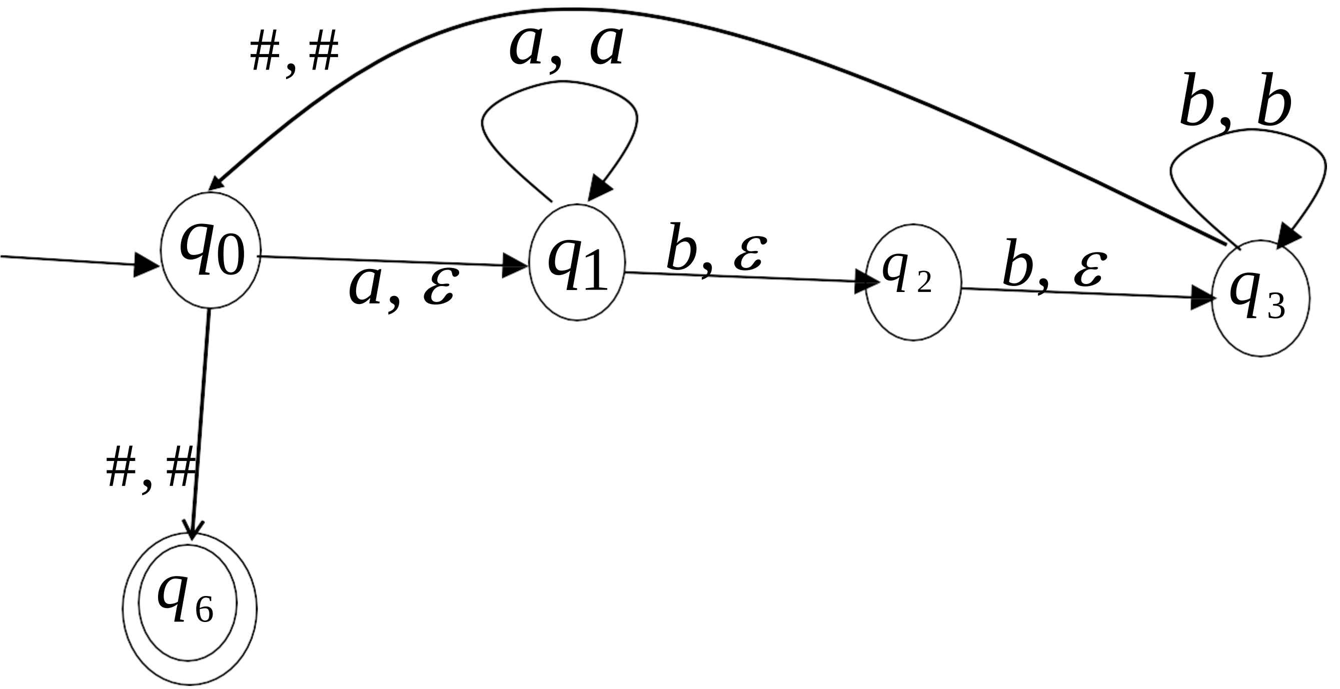
String is accepted..

Construct Post Machine which accepts the string over $\Sigma = \{a, b\}$ containing numbers of ‘a’ and ‘b’ same.

Write simulation for the string “abbaabba”



q_0
q_1
q_2
q_3
q_4
q_5
q_6
a, ε
b, ε
a, a
b, b
#,#

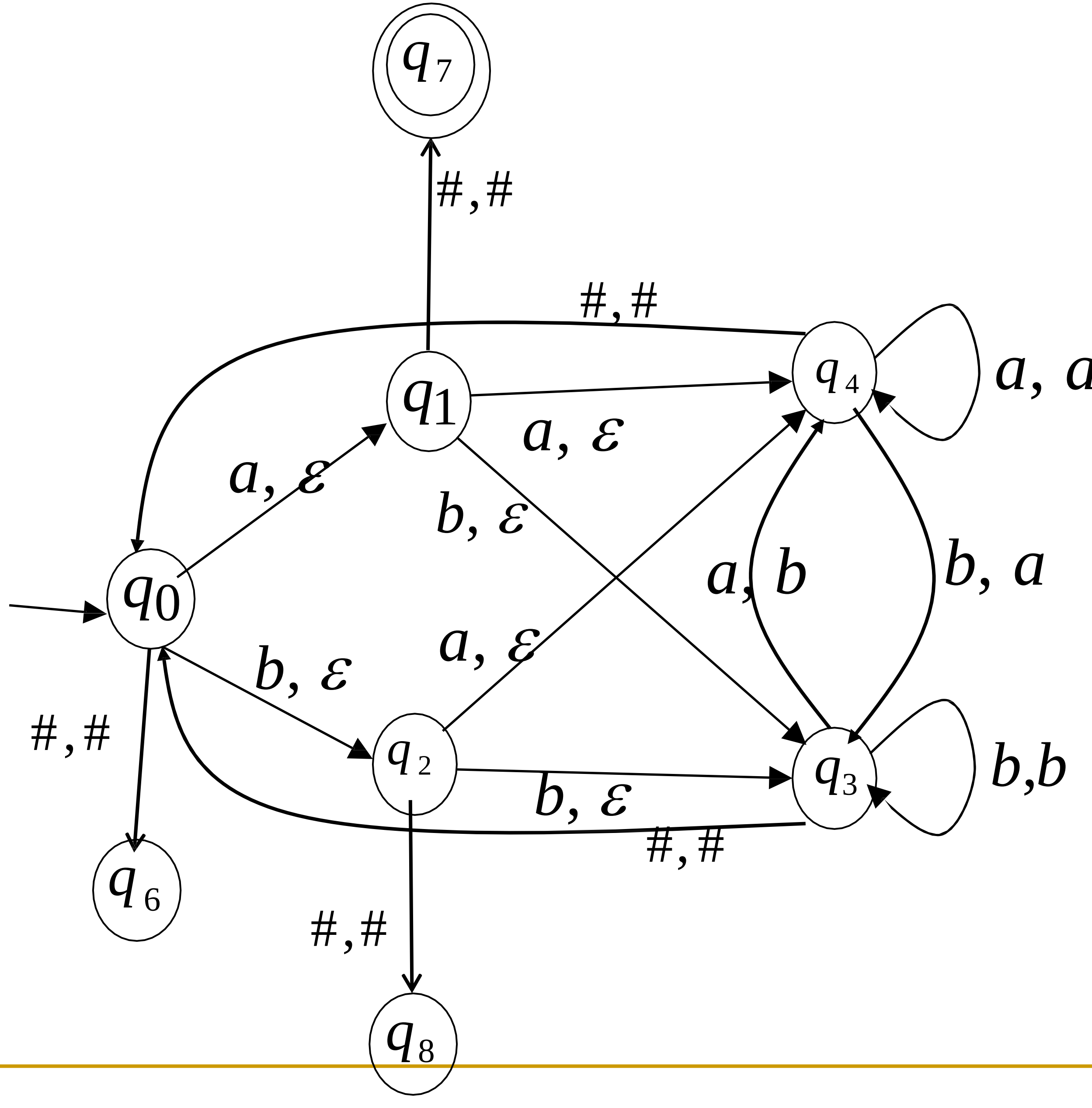

#, #
a, a
b, b
q0
q1
q2
q3
q6
a, ε
b, ε
b, ε
#, #

Recall

Post Machine

Construct Post Machine which accepts the string over $\boldsymbol{\Sigma}$ = {a, b} containing odd length & the element at the centre as 'a'.(8 Marks Nov-2017)

Write simulation for the string “abbabba”



q_7
#,#
#,#
q_4
$a,\ a$
q_1
$a,\ \varepsilon$
$a,\ \varepsilon$
$b,\ \varepsilon$
$a,\ b$
$b,\ a$
q_0
$b,\ \varepsilon$
$a,\ \varepsilon$
#,#
q_2
$b,\ \varepsilon$
q_3
b,b
#,#
q_6
#,#
q_8