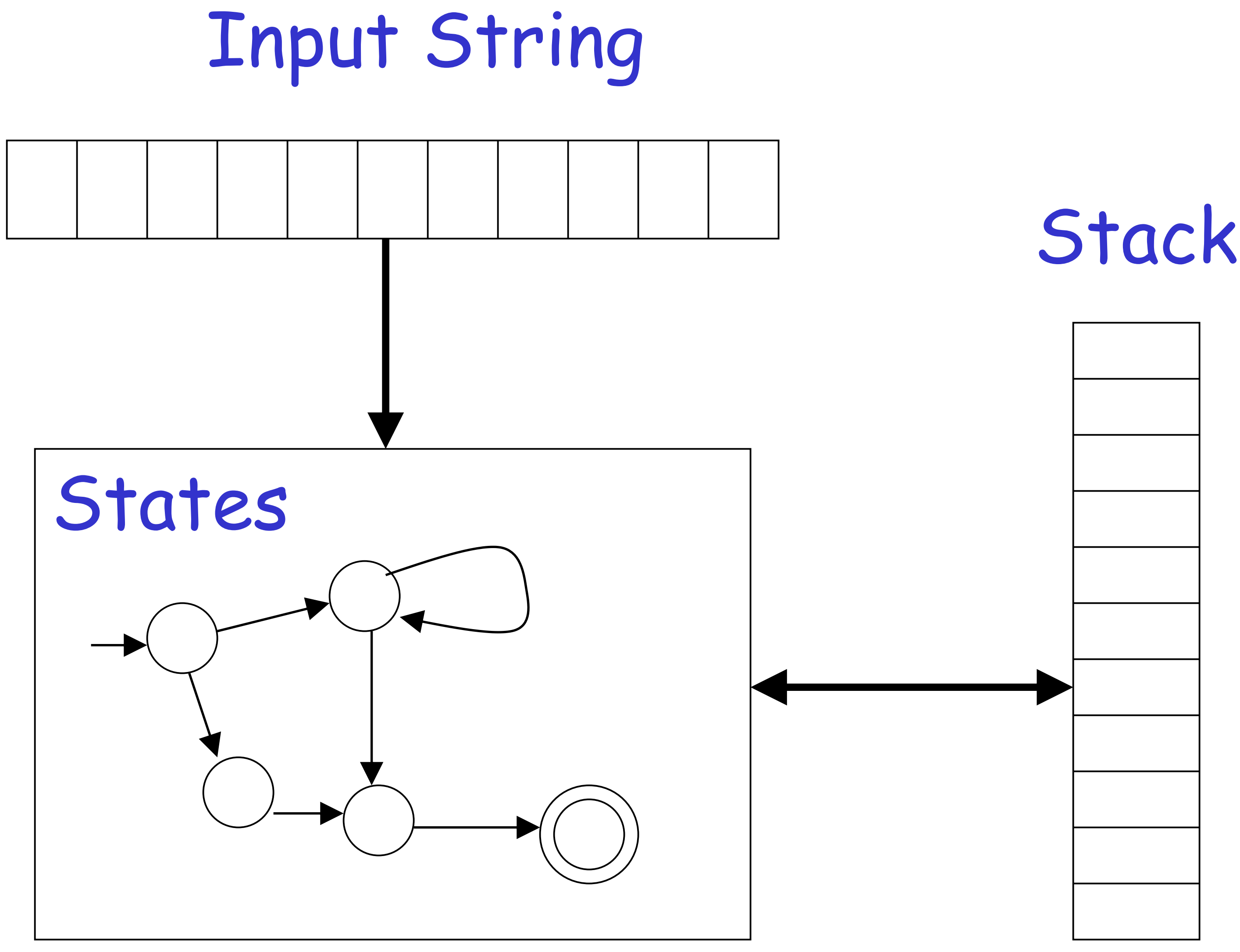
CHAPTER 7

Pushdown Automata
PDAs

By R.Ameri

Pushdown Automaton -- PDA



PDA

- PDA is a finite automata with extra memory called stack which helps Pushdown automata to recognize Context Free Languages.
- PDA has more powerful than Finite Automata automata.
- PDA is divided into
  - nondeterministic pushdown accepter (npda)
  - deterministic pushdown accepter (dpda)

Formalities for NPDAs

Formal Definition

Non-Deterministic Pushdown Automaton

NPDA

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

Final states

States

Input alphabet

Stack alphabet

Transition function

Initial state

Stack start symbol

- $\delta : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \rightarrow$ set of finite subsets of $Q \times \Gamma^*$
- $z \in \Gamma$
- $F \subseteq Q$
- $q_0 \in Q$

Instantaneous Description

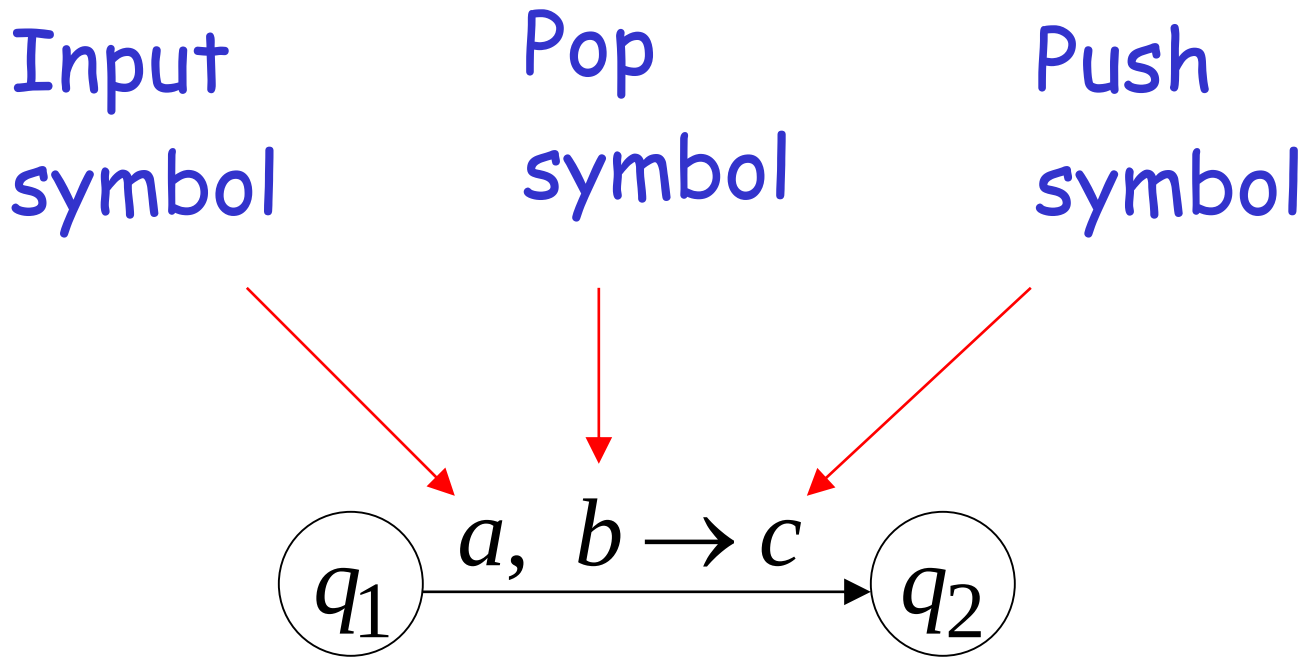
$$(q,\ u,\ s)$$

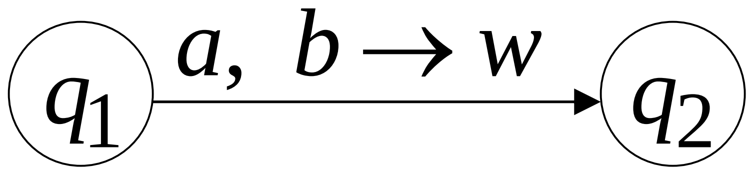
Current state

Remaining input

Current stack contents

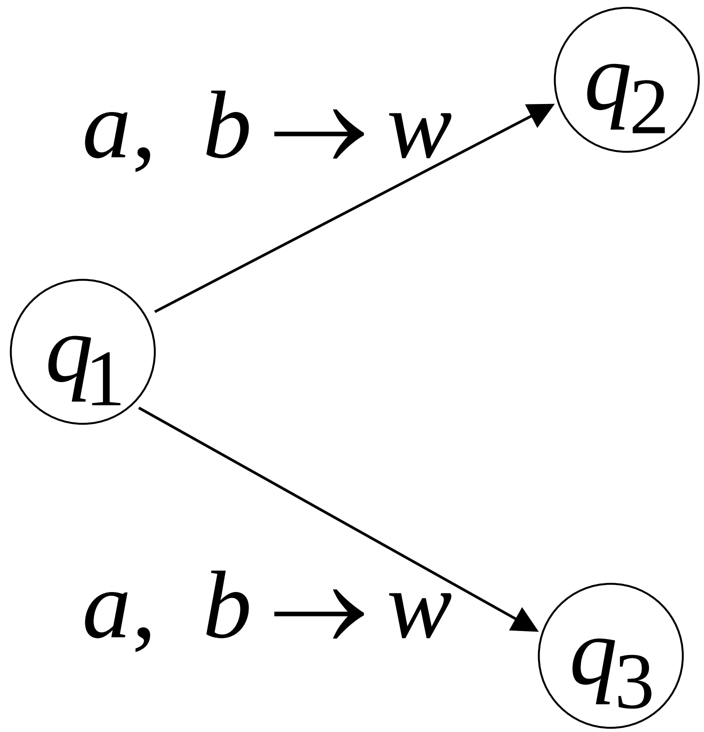
The States





Transition function:

$$\delta(q_1, a, b) = \{(q_2, w)\}$$



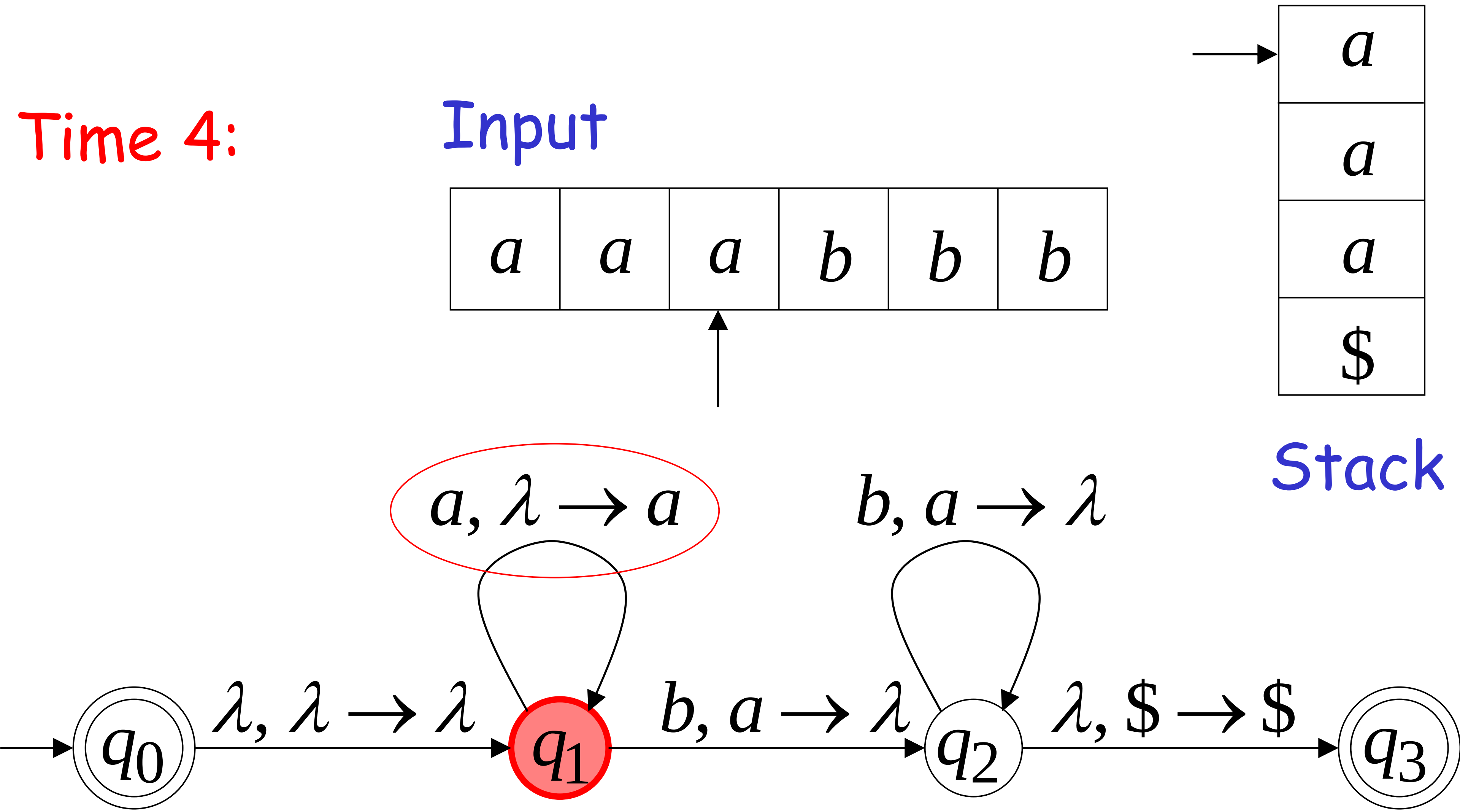
Transition function:

$$\delta(q_1, a, b) = \{(q_2, w), \ (q_3, w)\}$$

Example: **Instantaneous Description**

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

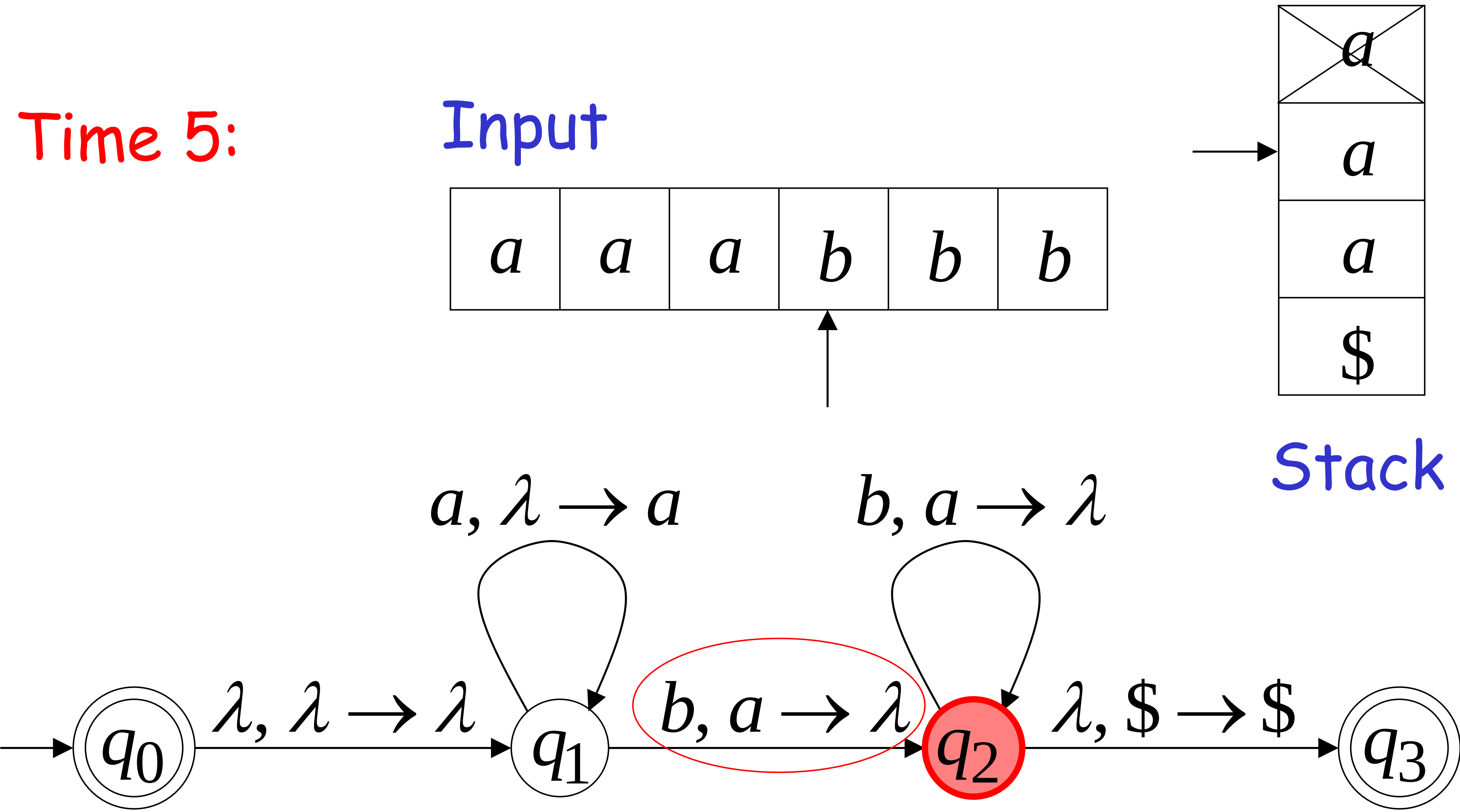
Time 4:



Example: **Instantaneous Description**

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

Time 5:



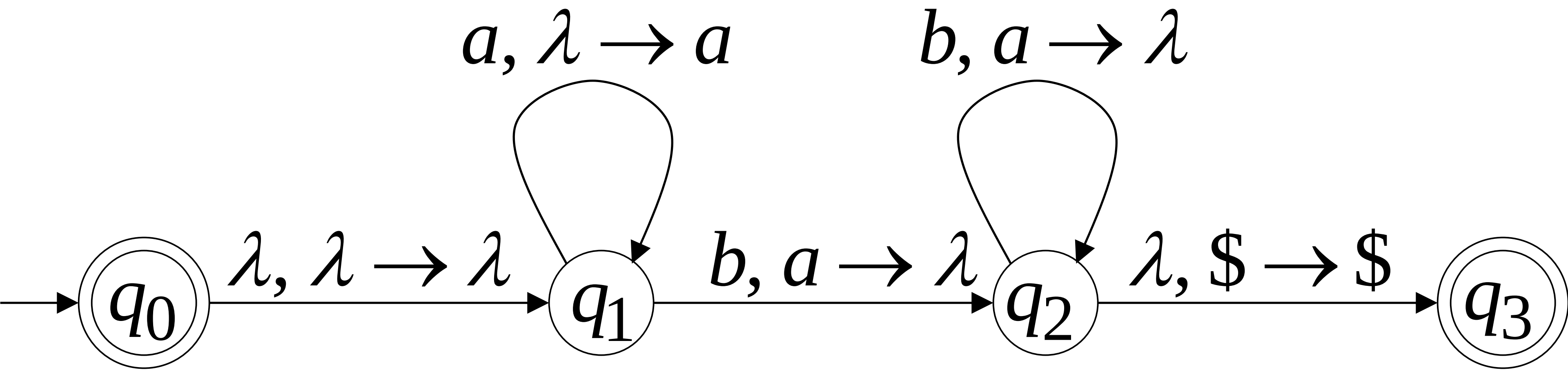
We write:

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

Time 4 Time 5

A computation:

$$(q_0, aaabbb, \$) \succ (q_1, aaabbb, \$) \succ$$
$$(q_1, aabbb, a\$) \succ (q_1, abbb, aa\$) \succ (q_1, bbb, aaa\$) \succ$$
$$(q_2, bb, aa\$) \succ (q_2, b, a\$) \succ (q_2, \lambda, \$) \succ (q_3, \lambda, \$)$$



$$(q_0, aaabbb, \$) \succ (q_1, aaabbb, \$) \succ$$
$$(q_1, aabbb, a\$) \succ (q_1, abbb, aa\$) \succ (q_1, bbb, aaa\$) \succ$$
$$(q_2, bb, aa\$) \succ (q_2, b, a\$) \succ (q_2, \lambda, \$) \succ (q_3, \lambda, \$)$$

For convenience we write:

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

Formal Definition

Language $L(M)$ of NPDA M:

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

Initial state

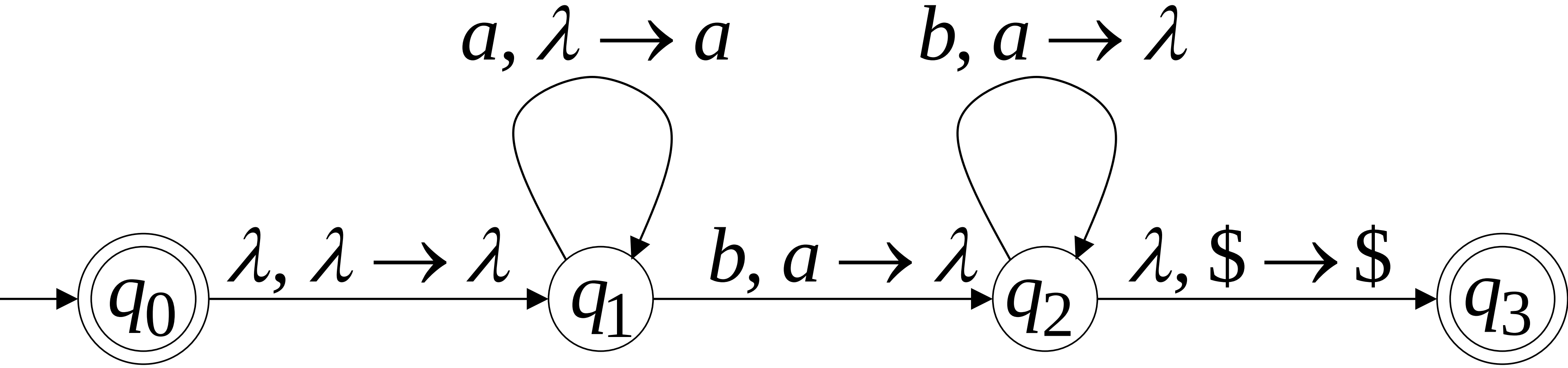
Final state

Example:

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

$$aaabbb \in L(M)$$

NPDA M :

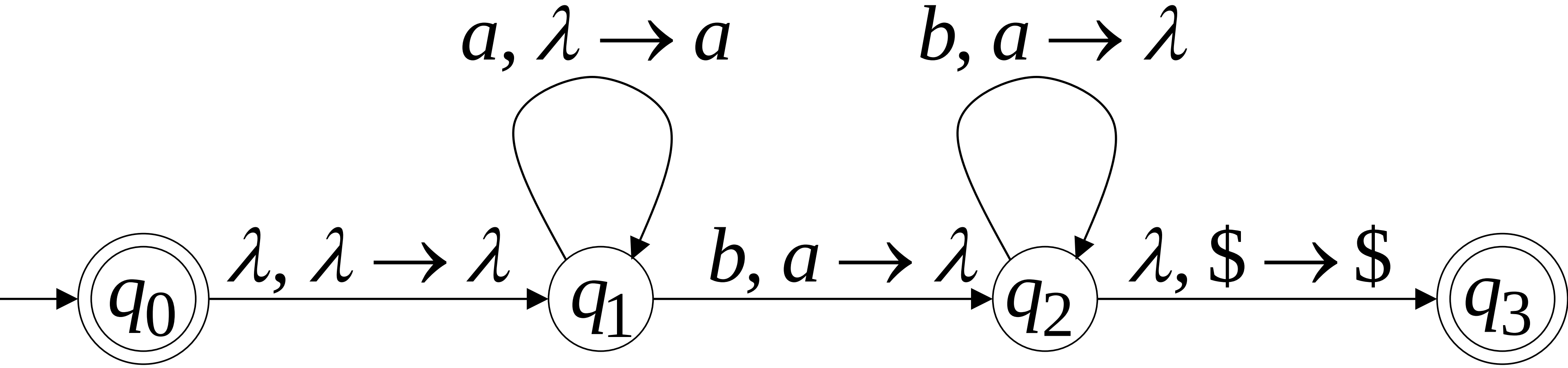


$$(q_0, a^n b^n, \$) \overset{*}{\succ} (q_3, \lambda, \$)$$

$$\Downarrow$$

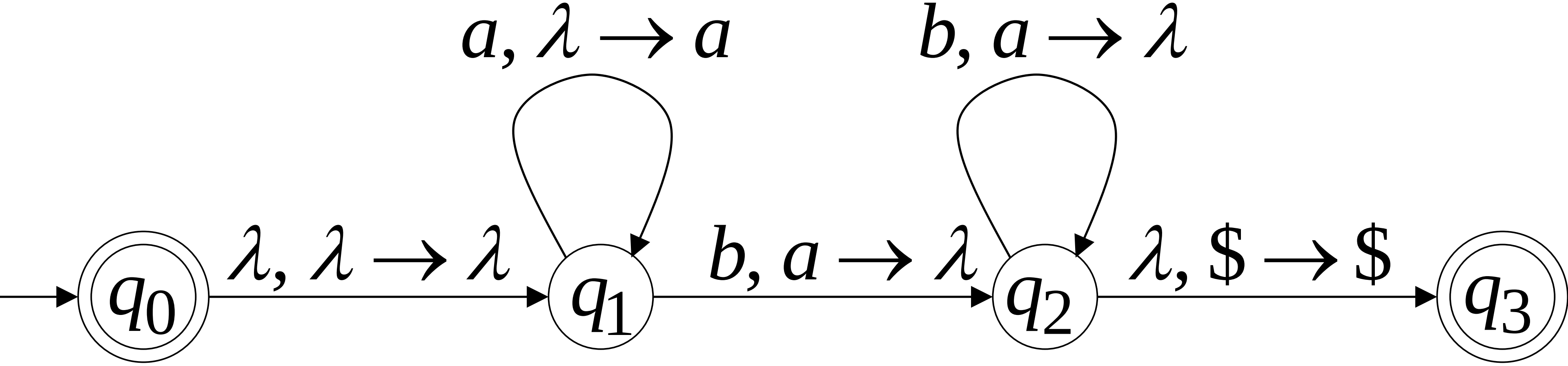
$$a^n b^n \in L(M)$$

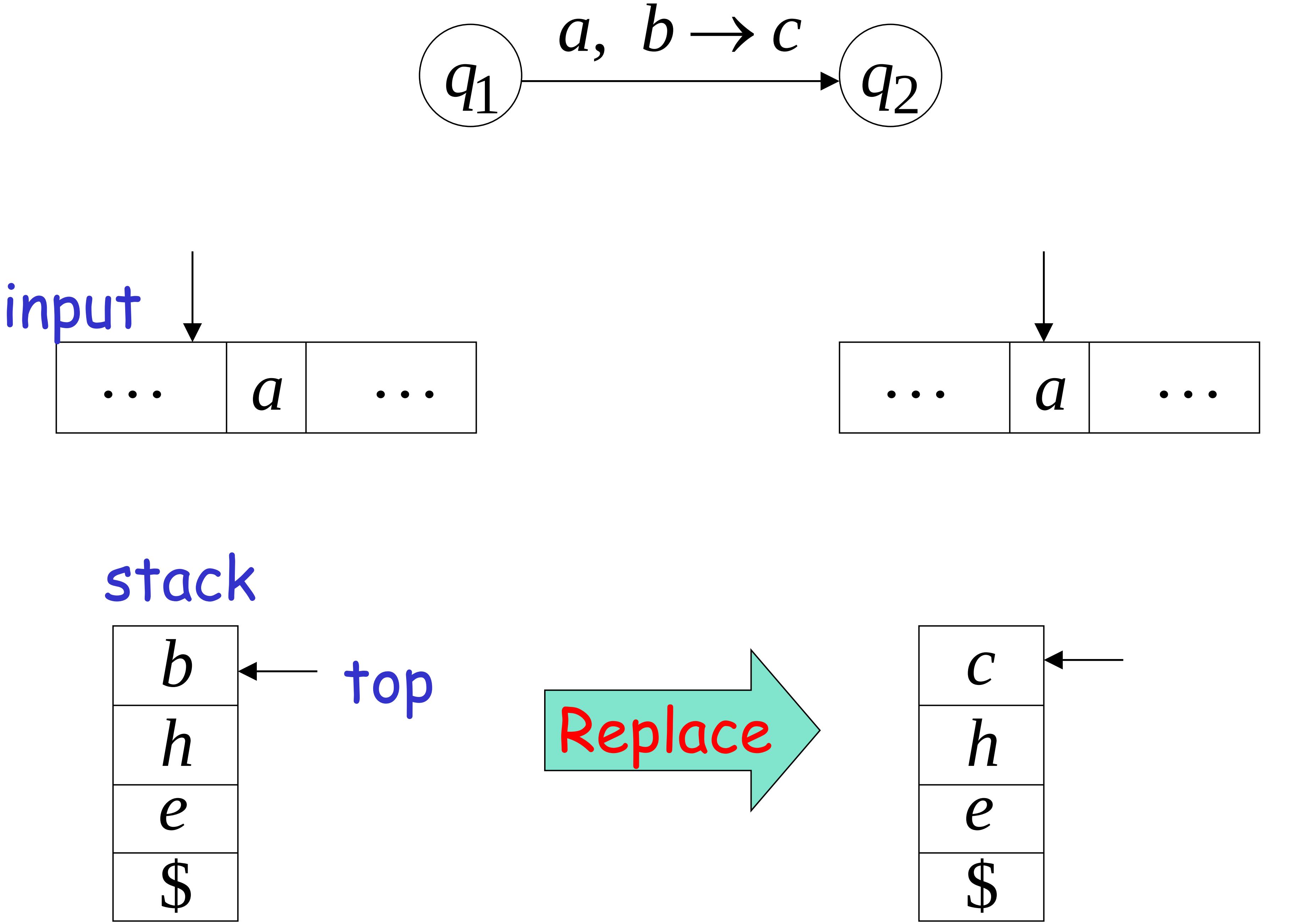
NPDA M :



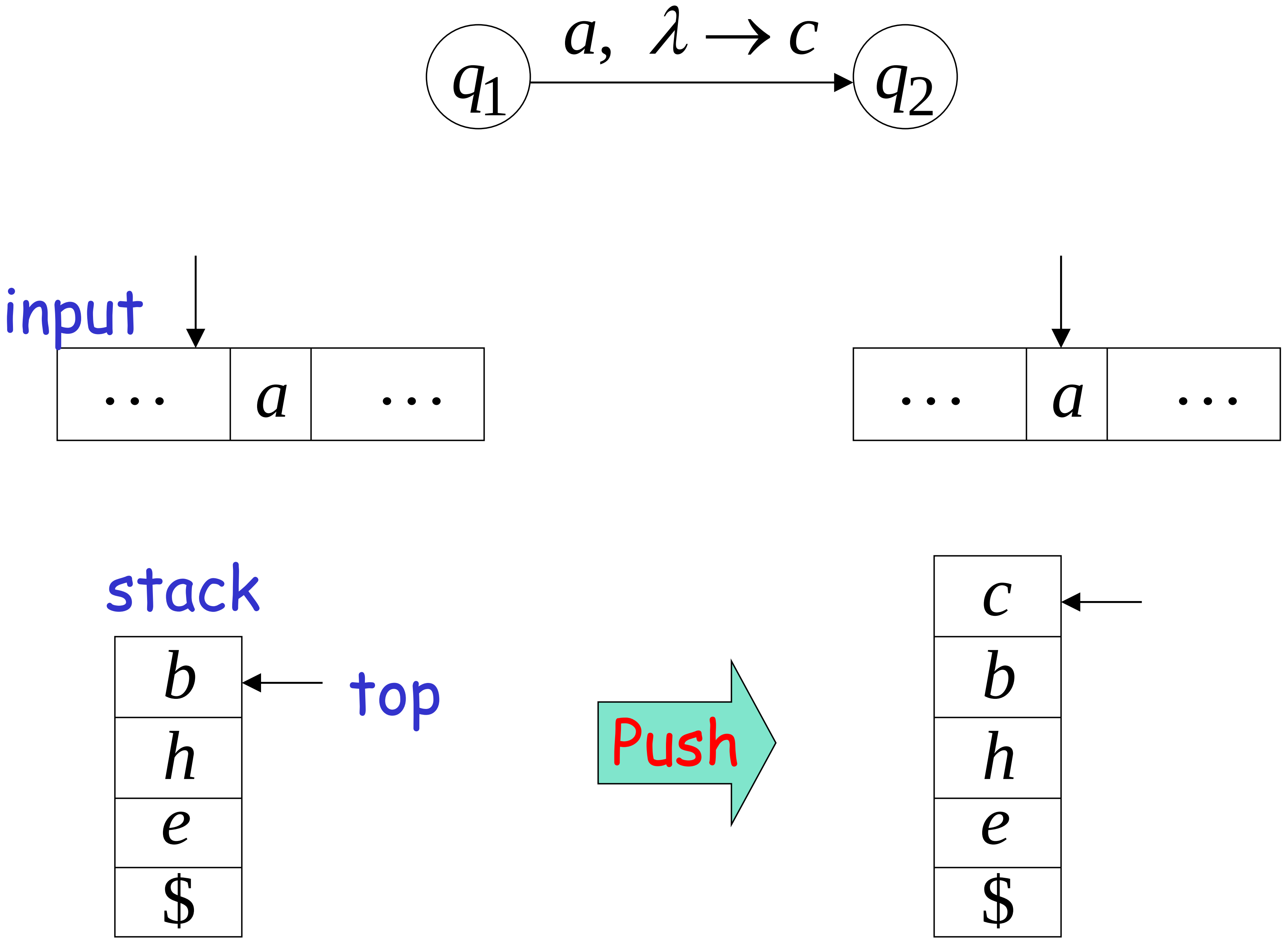
Therefore: $L(M) = \{a^n b^n : n \geq 0\}$

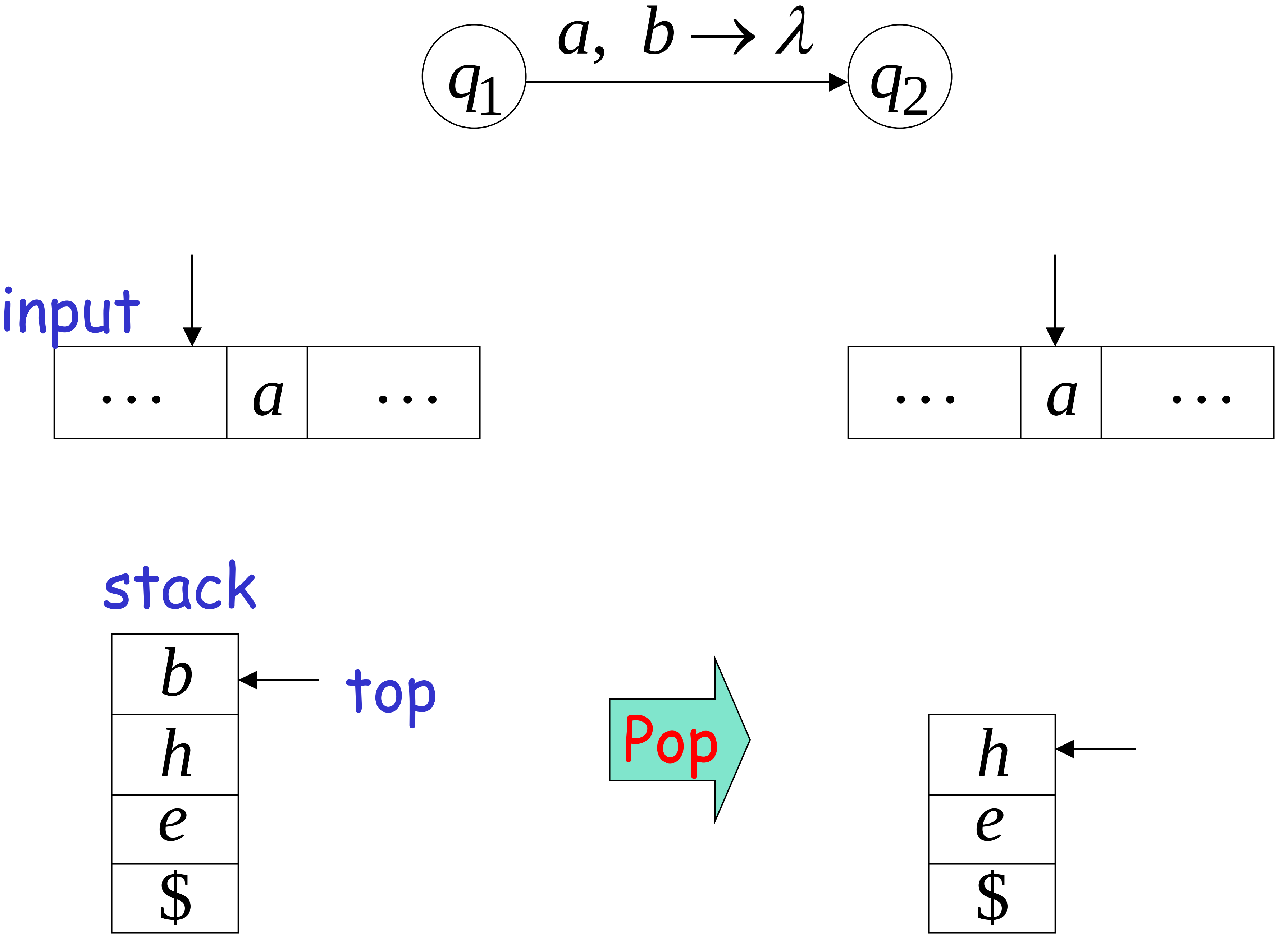
NPDA M:

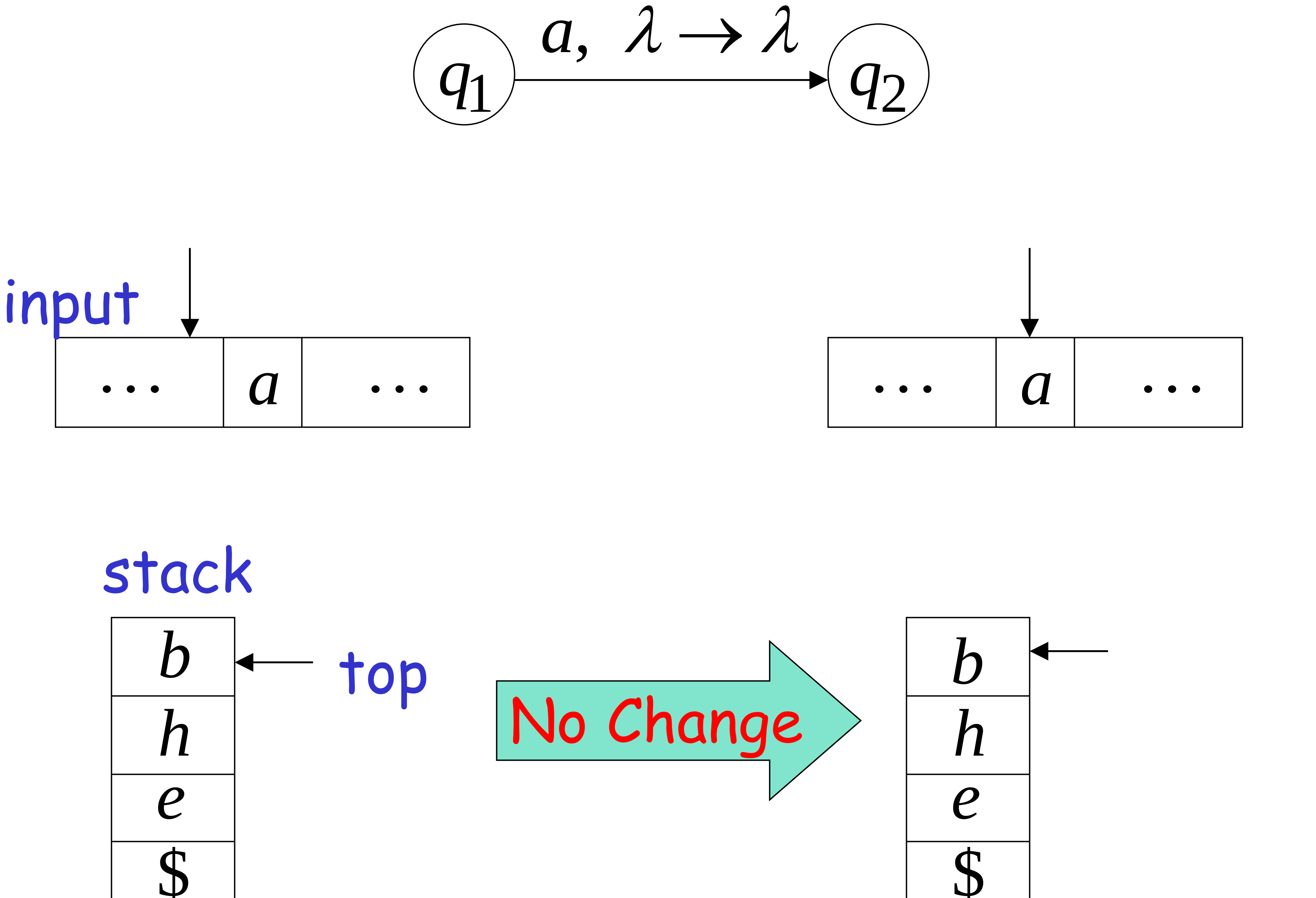




q_1
$a, \; b \to c$
q_2
input
$\cdots$ a $\cdots$
$\cdots$ a $\cdots$
stack
top
b
h
e
$\$$
Replace
c
h
e
$\$$

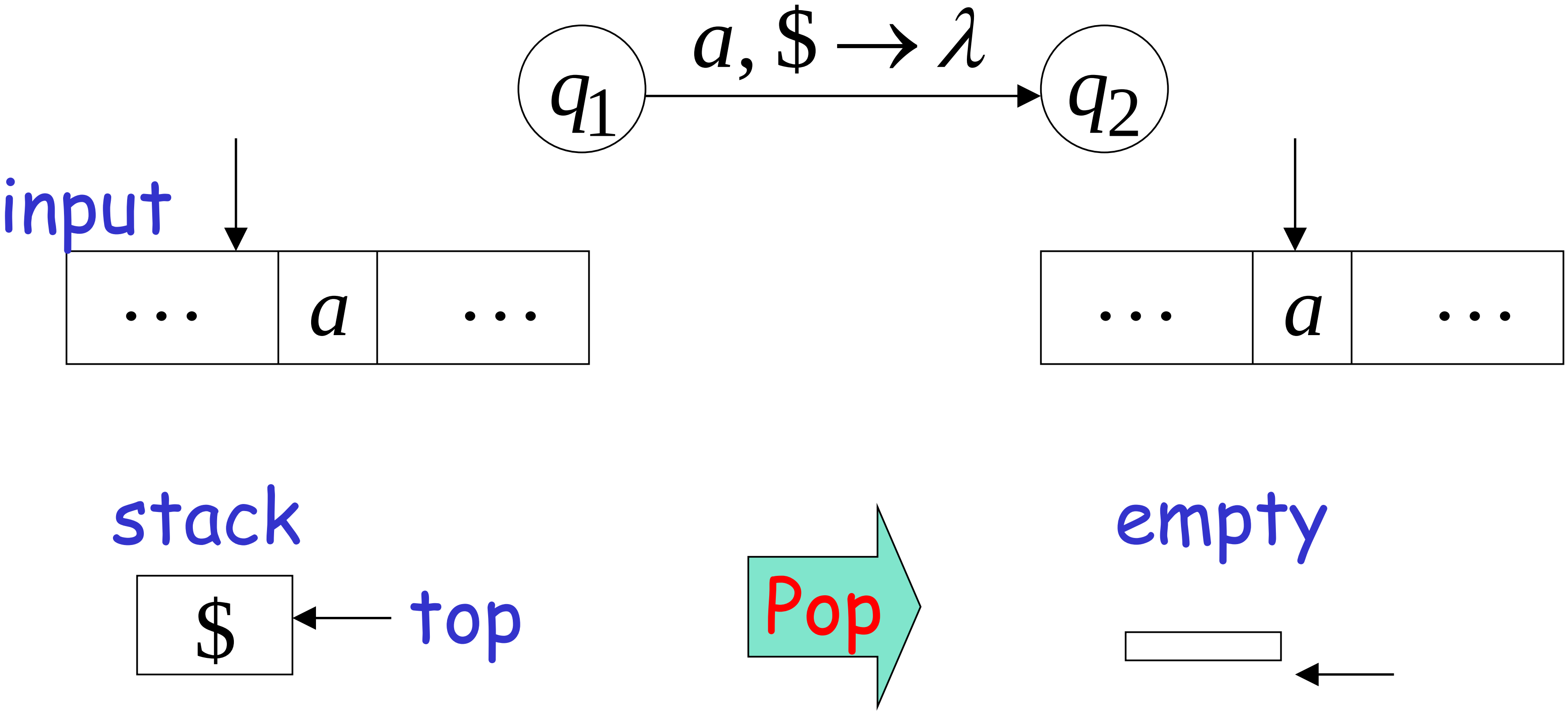

a, λ → c
q1
q2
input
...
a
...
...
a
...
stack
b
h
e
$
top
Push
c
b
h
e
$


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

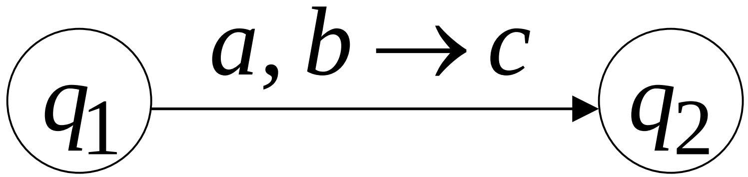


q_1
$a, \lambda \to \lambda$
q_2
input
$\cdots$ a $\cdots$
$\cdots$ a $\cdots$
stack
b
h
e
$\$$
top
No Change
b
h
e
$\$$

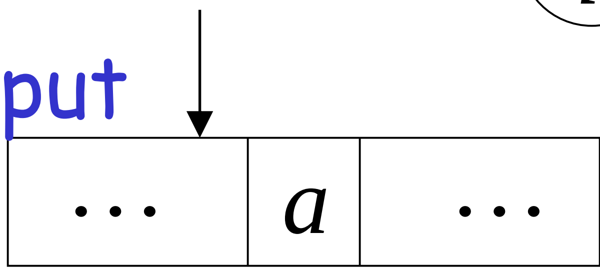
A Possible Transition



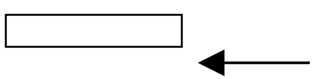
A Bad Transition

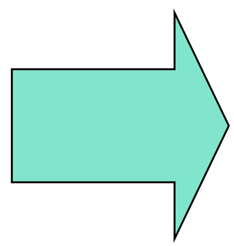


input



Empty stack





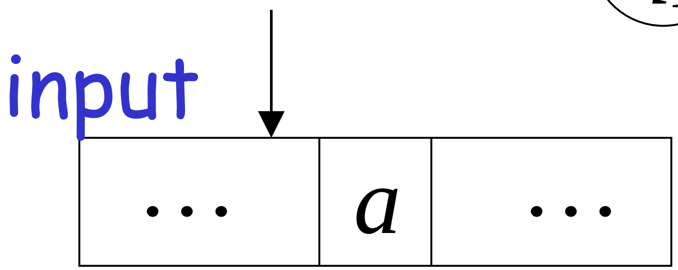
HALT

The automaton Halts in state q_1

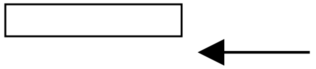
and Rejects the input string

A Bad Transition

$$q_1 \xrightarrow{a,\ \lambda \to c} q_2$$



Empty stack



HALT

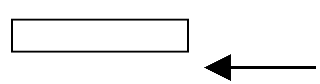
The automaton **Halts** in state q_1

and **Rejects** the input string

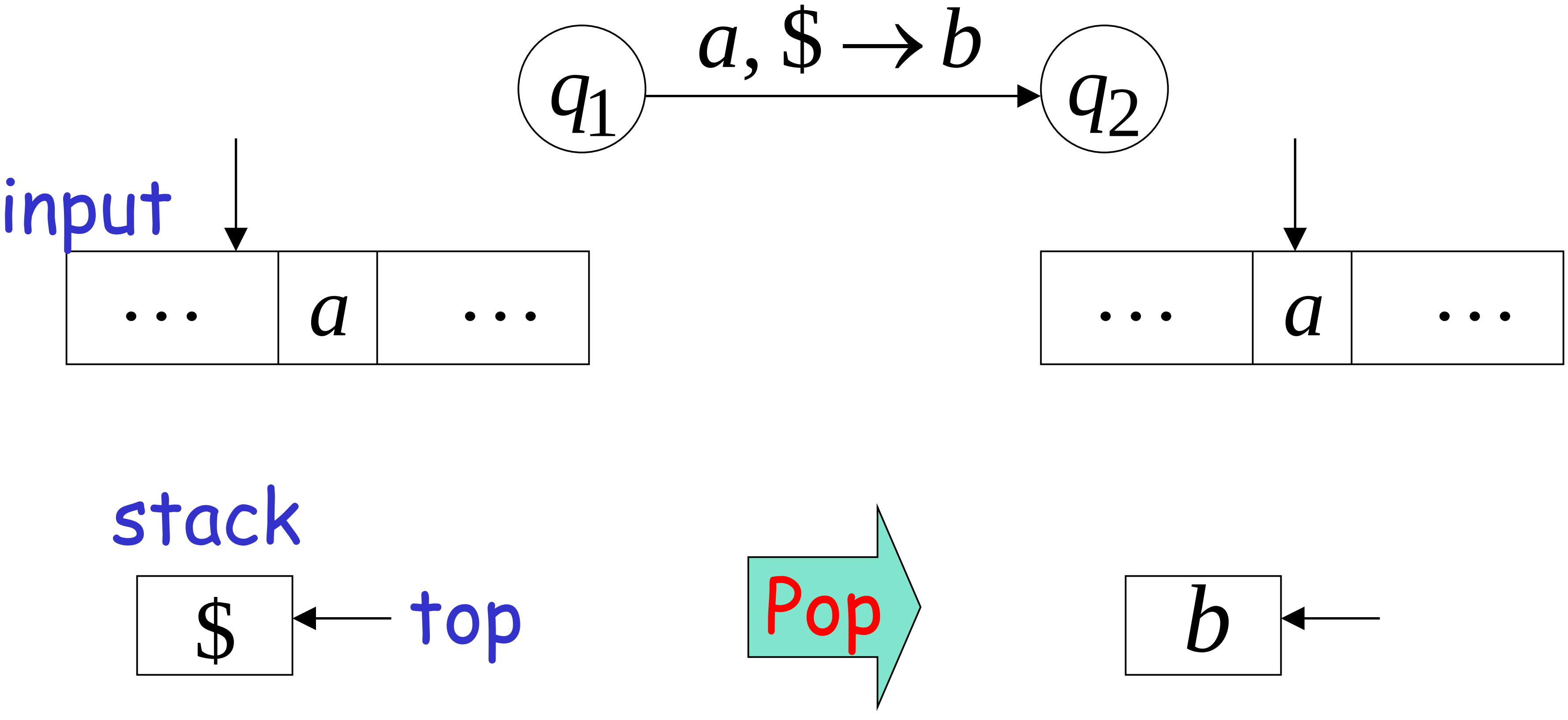
No transition is allowed to be followed
When the stack is empty

$$q_1 \xrightarrow{x,\ y \to z} q_2$$

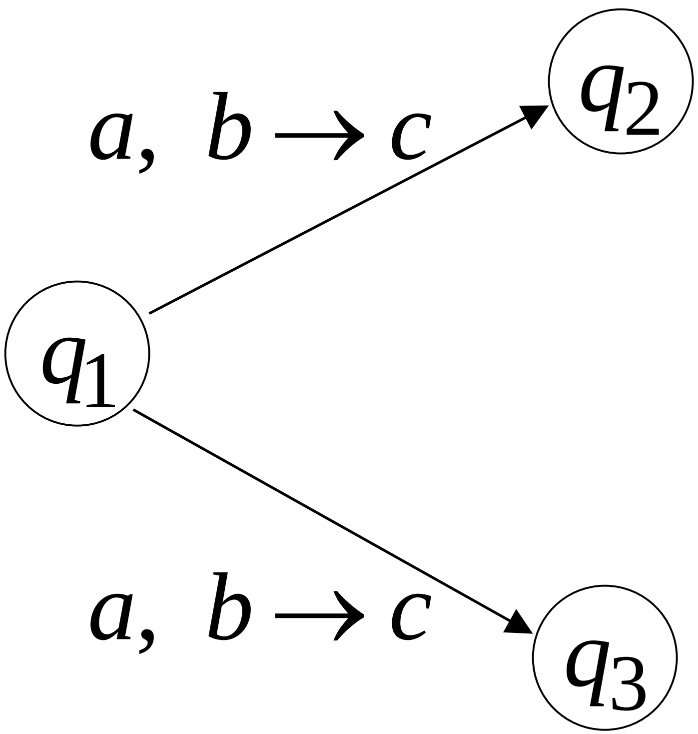
Empty stack

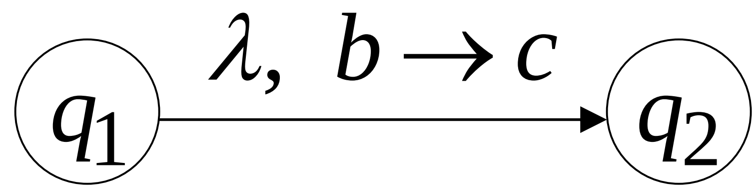


A Good Transition



Non-Determinism





$\lambda -$ transition

These are allowed transitions in a

Non-deterministic PDA (NPDA)

A string is accepted by:

- **Final State:**

  All the input is consumed

  AND

  The last state is a final state

At the end of the computation, we do not care about the stack contents

$$L(PDA) = \{w \mid (q_0, w, I) \vdash^* (q, \lambda, x), q \in F\}$$

a string is rejected in acceptance by Final State if in every computation with this string:

The input cannot be consumed

OR

The input is consumed and the last state is not a final state

OR

The stack head moves below the bottom of the stack

A string is accepted by:

- **Empty Stack**:

  All the input is consumed

  AND

  the PDA has emptied its stack

  At the end of the computation,
  we do not care about the last state.

$$L(PDA) = \{w \mid (q_0, w, \$) \vdash^* (q, \lambda, \$),\ q \in Q\}$$

a string is rejected in acceptance by Empty Stack if in every computation with this string:

The input cannot be consumed

OR

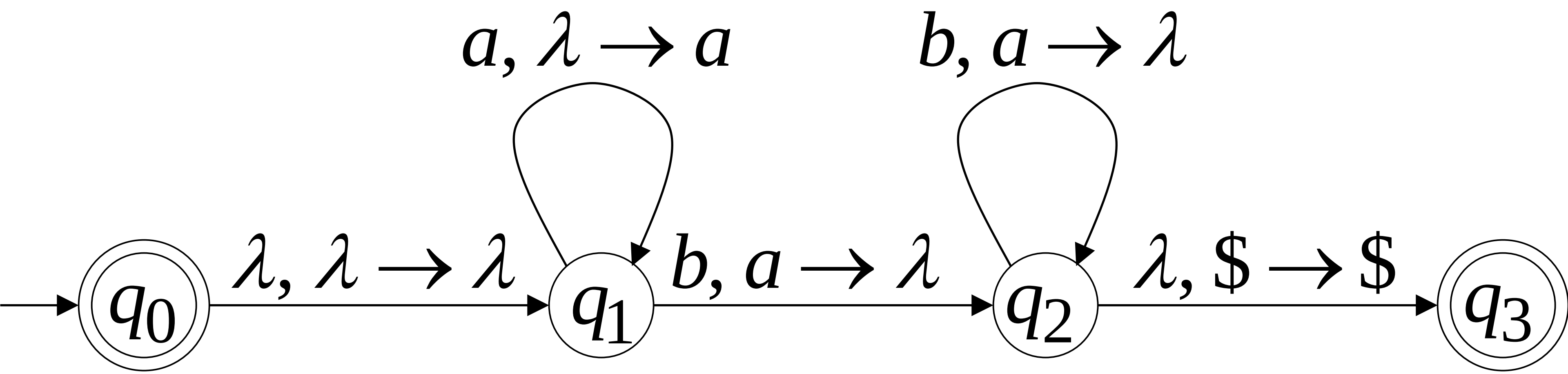
The input is consumed and stack is not empty

OR

The stack head moves below the bottom of the stack

NPDA: Non-Deterministic PDA

Example:



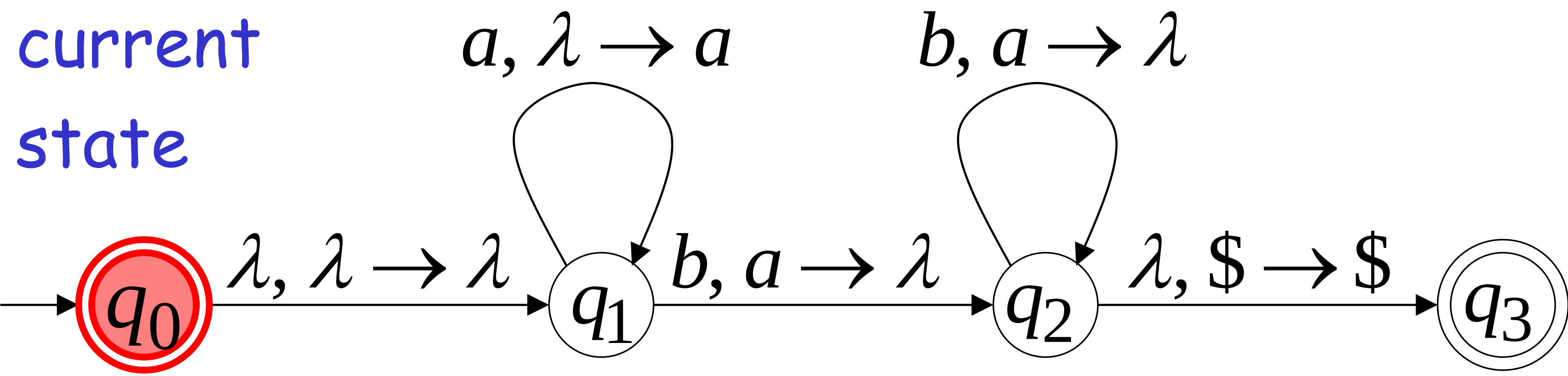
Execution Example: Time 0

Input

a	a	a	b	b	b

$\$$

Stack



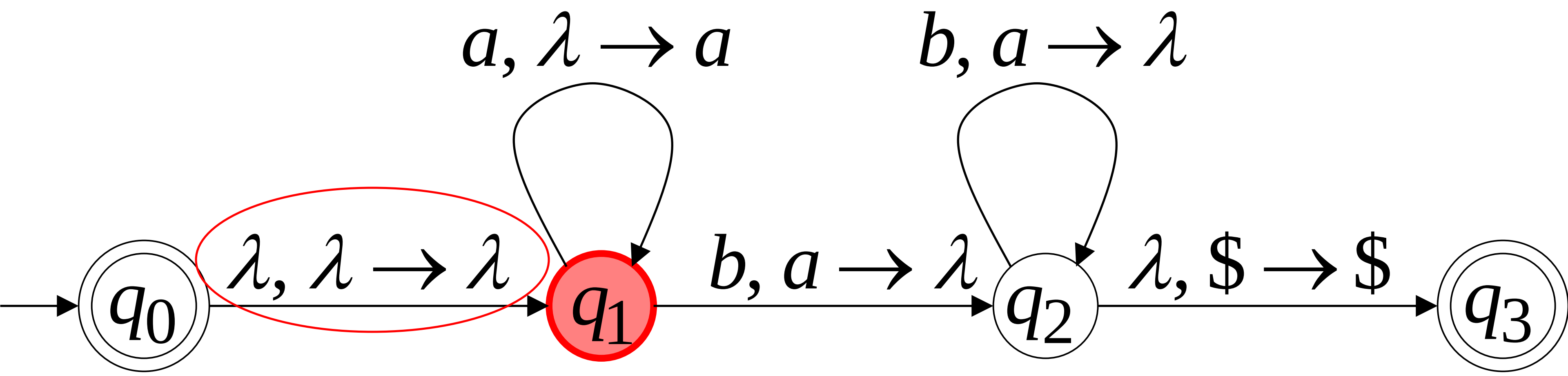
Time 1

Input

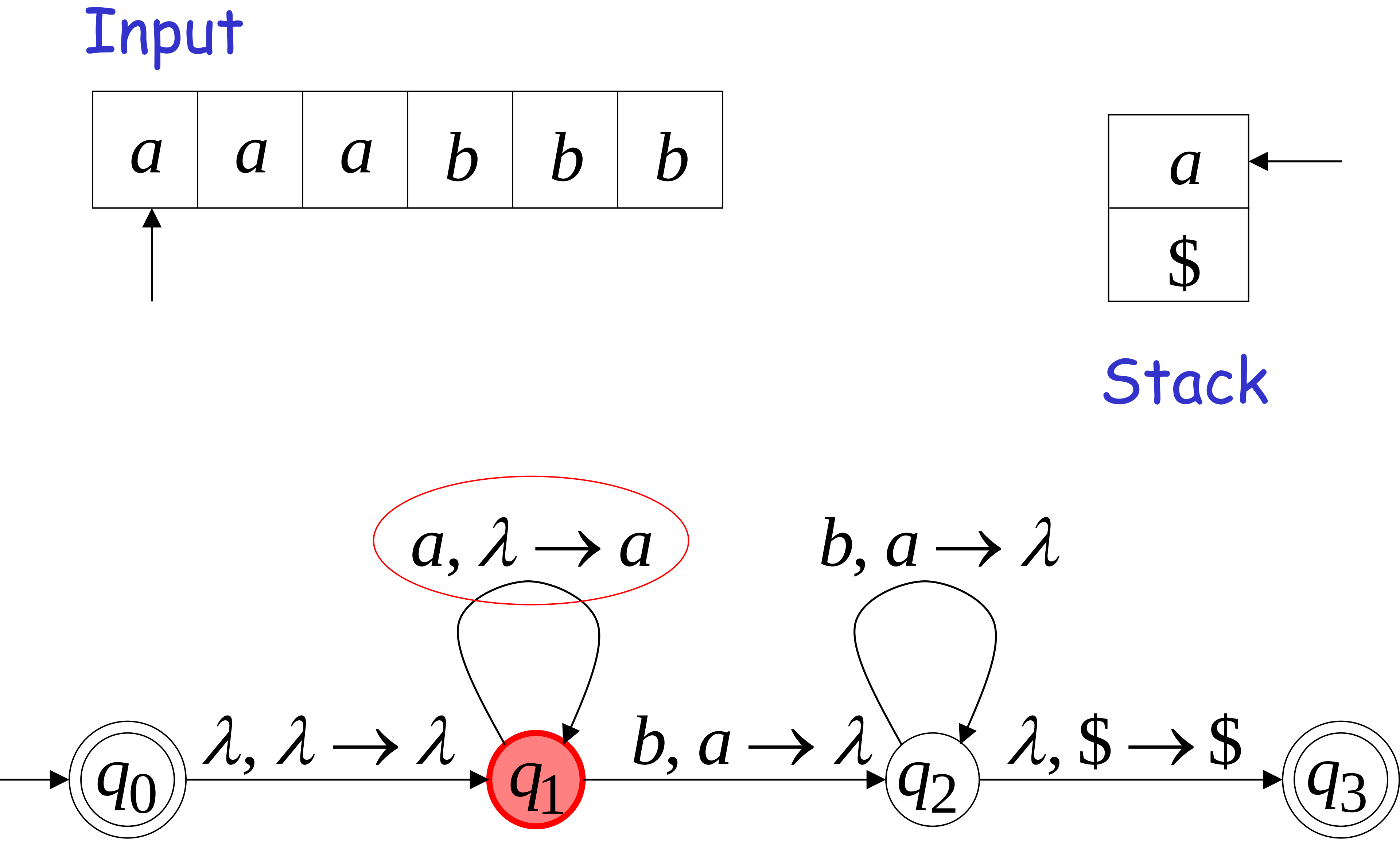
a	a	a	b	b	b

$\$$

Stack

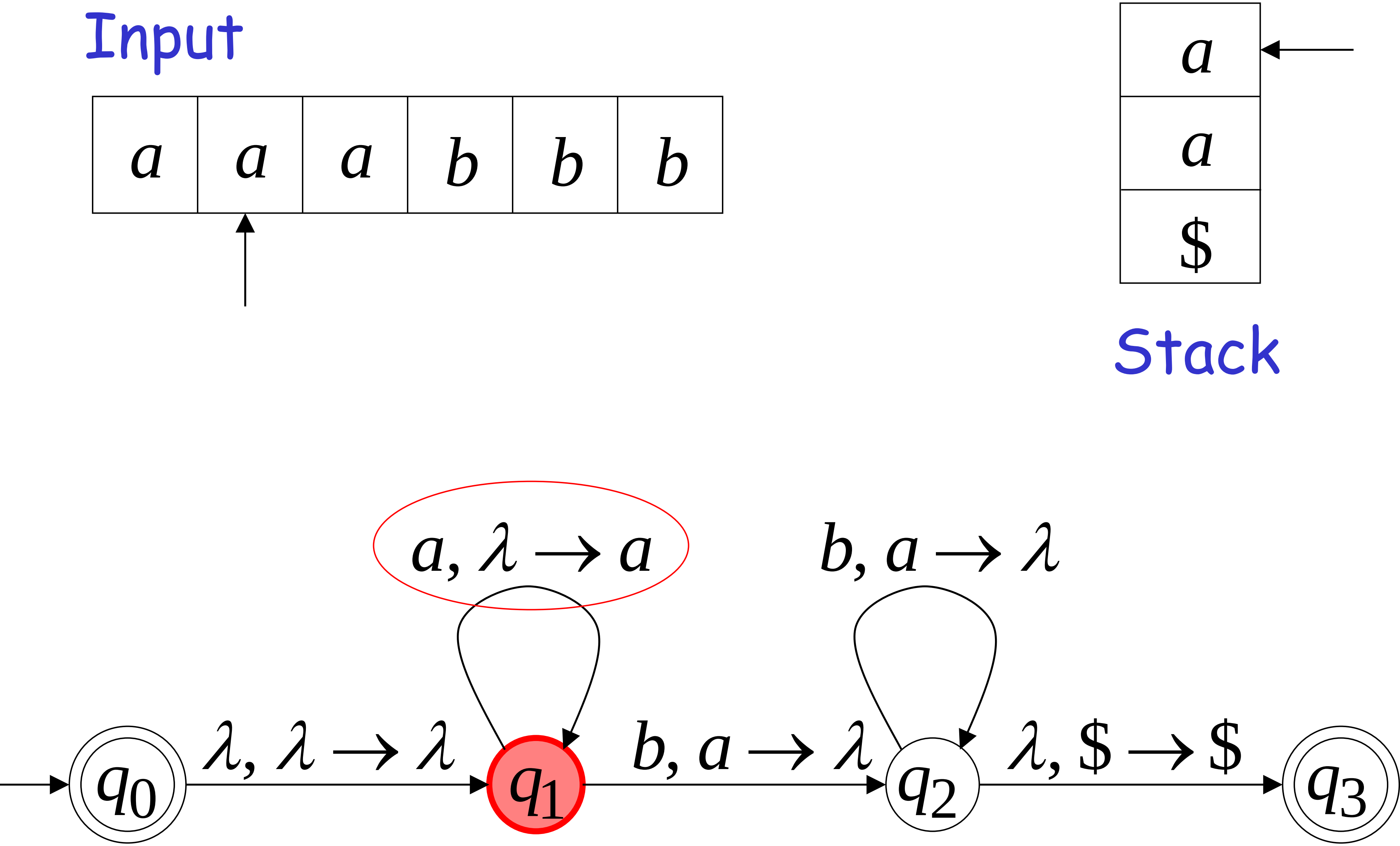


Time 2

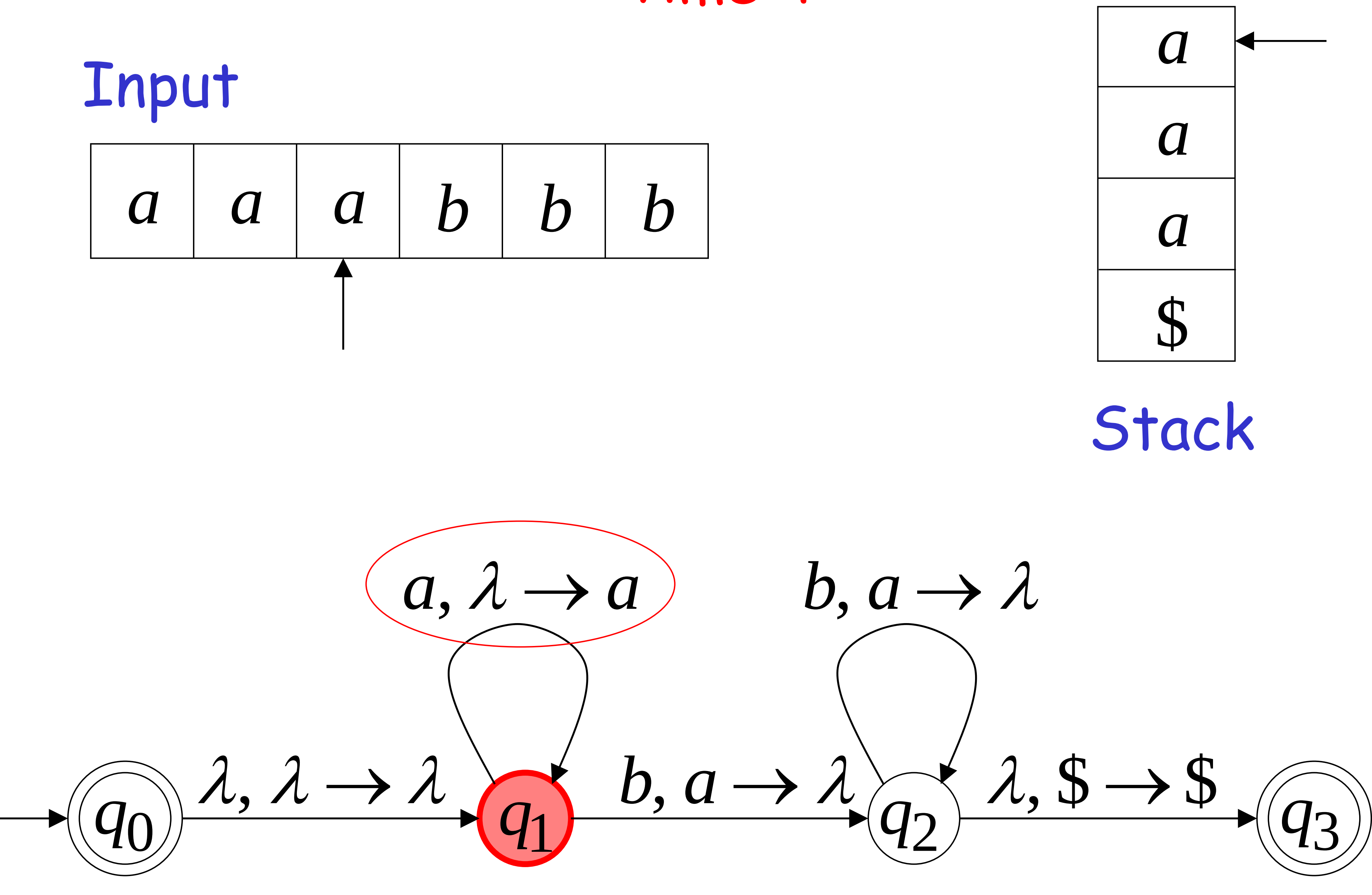


Input
a a a b b b
a
$
Stack
$a, \lambda \rightarrow a$
$b, a \rightarrow \lambda$
$\lambda, \lambda \rightarrow \lambda$
$b, a \rightarrow \lambda$
$\lambda, \$ \rightarrow \$$
q_0
q_1
q_2
q_3

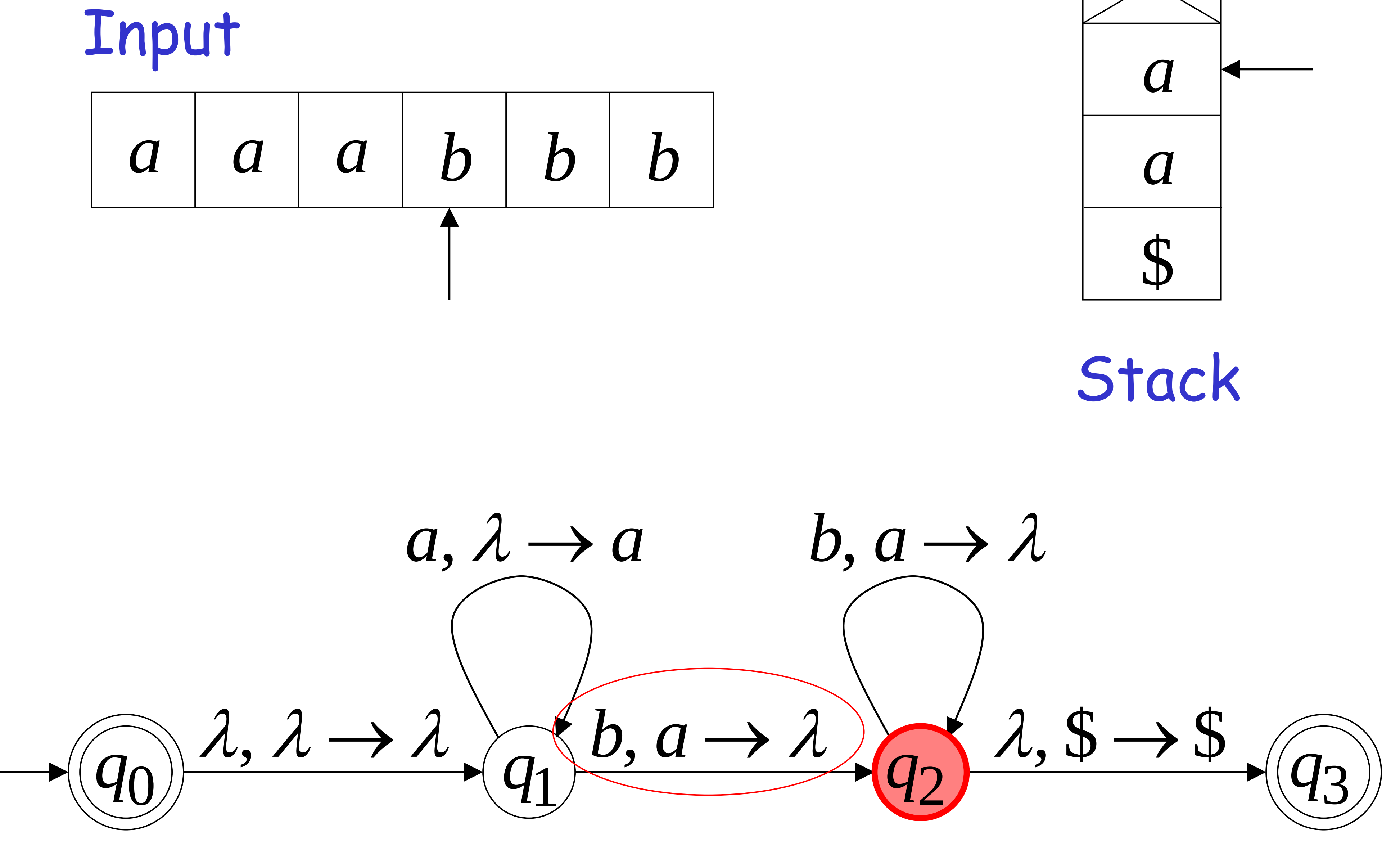
Time 3


Input
a a a b b b
a
a
$
Stack
$a, \lambda \to a$
$b, a \to \lambda$
$\lambda, \lambda \to \lambda$
$b, a \to \lambda$
$\lambda, \$ \to \$$
q_0
q_1
q_2
q_3

Time 4



Time 5


Input
a a a b b b
a
a
a
$
Stack
$a, \lambda \to a$
$b, a \to \lambda$
$\lambda, \lambda \to \lambda$
$b, a \to \lambda$
$\lambda, \$ \to \$$
q_0
q_1
q_2
q_3

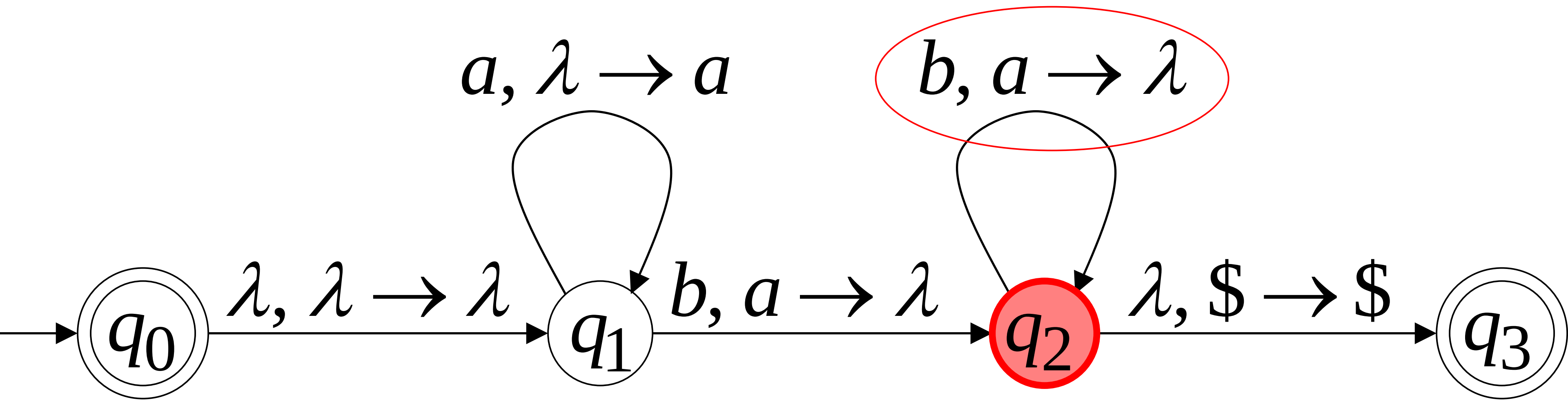
Time 6

Input

a	a	a	b	b	b

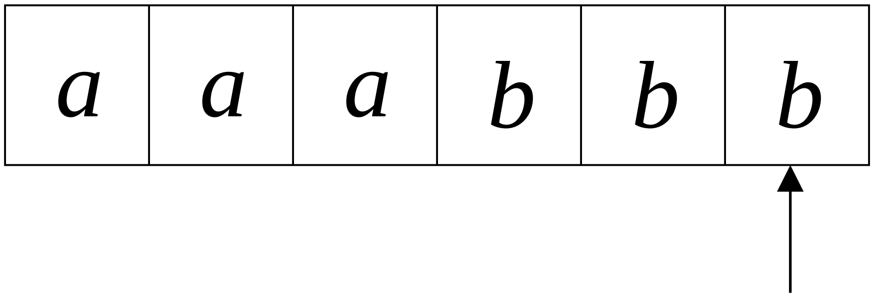
Stack

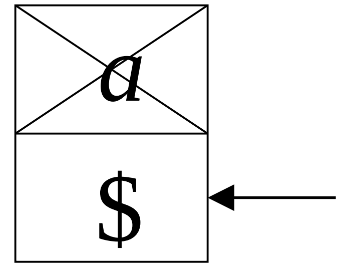
$\cancel{a}$
a
$\$$



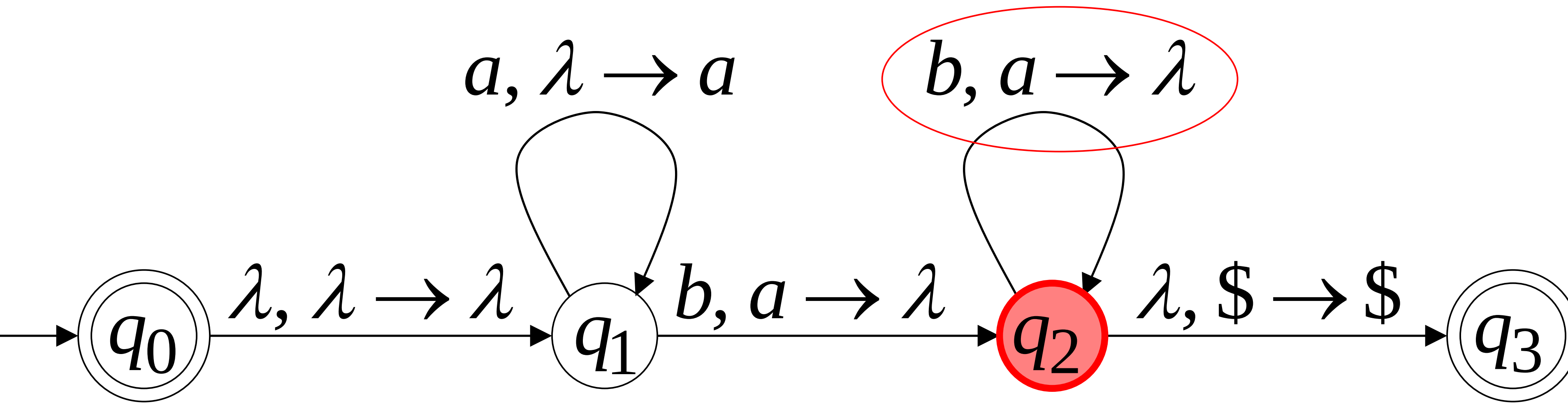
Time 7

Input

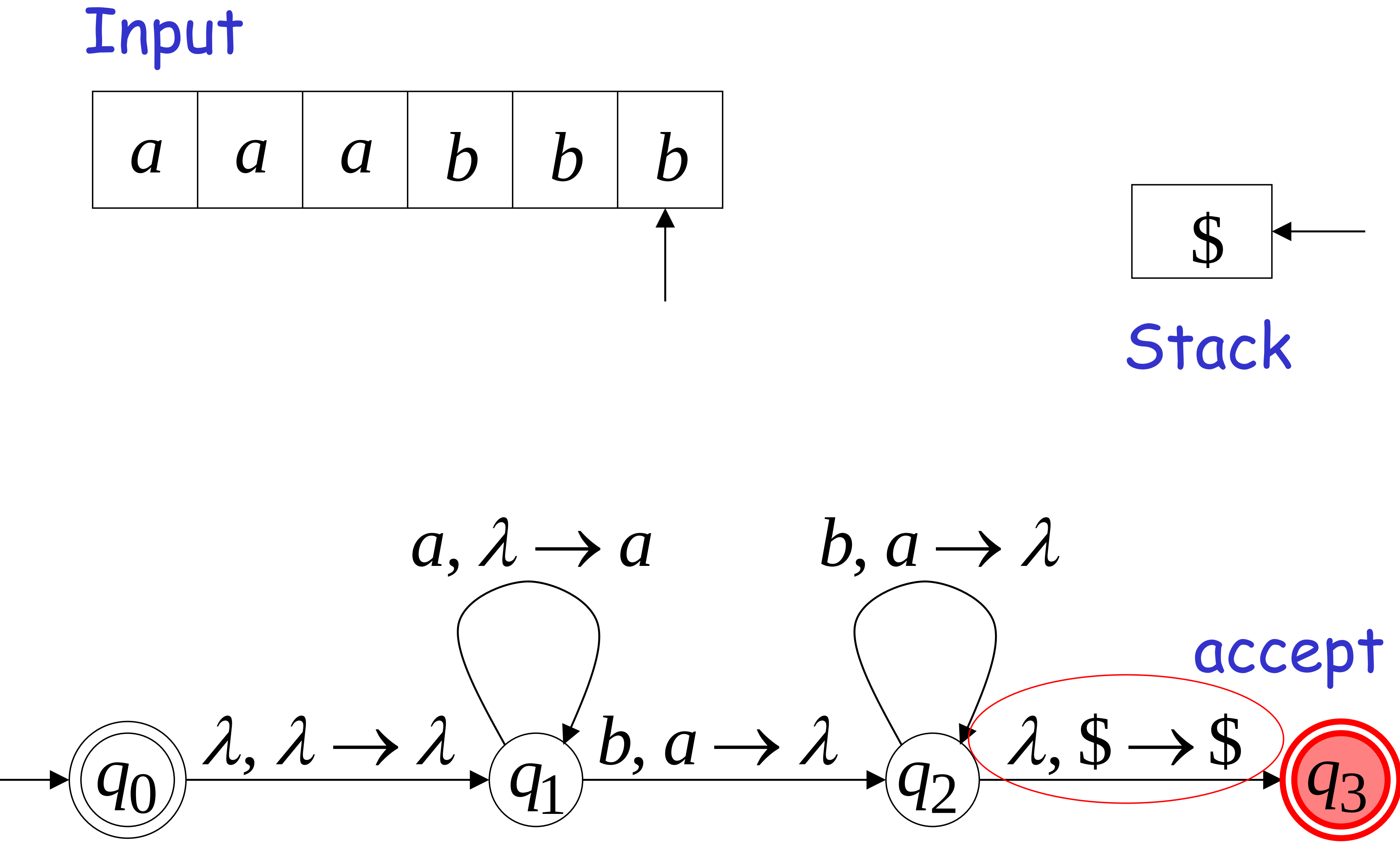




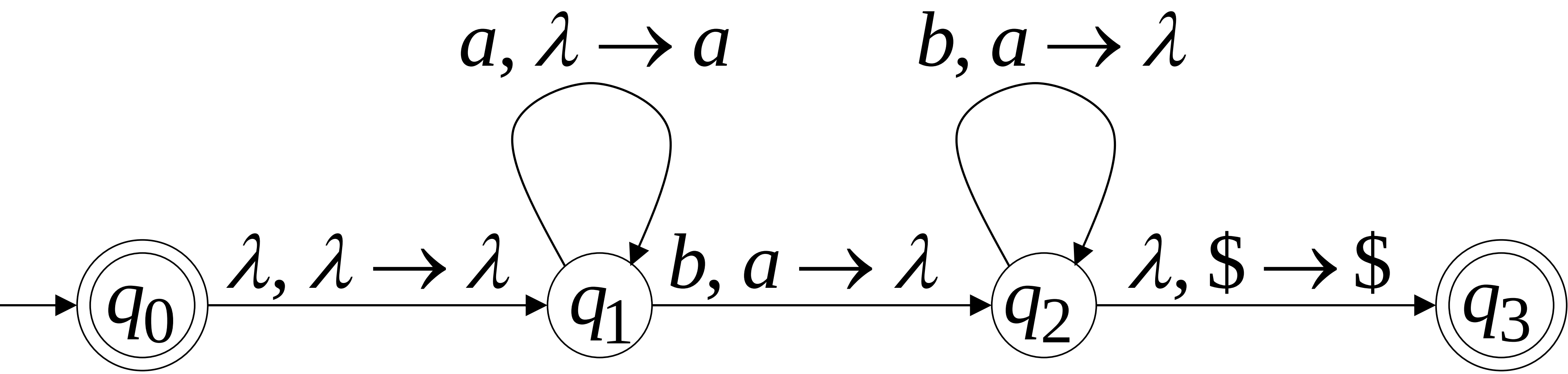
Stack



Time 8


Input
a a a b b b
$
Stack
$a, \lambda \to a$
$b, a \to \lambda$
accept
$\lambda, \lambda \to \lambda$
$b, a \to \lambda$
$\lambda, \$ \to \$$
q_0
q_1
q_2
q_3

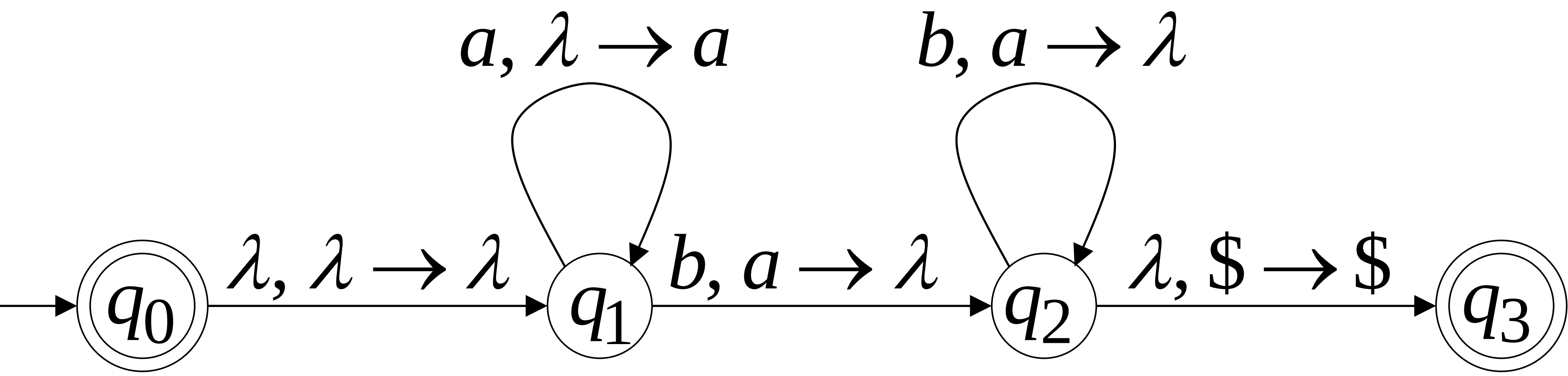
The input string $aaabbb$
is accepted by the NPDA:



In general,

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

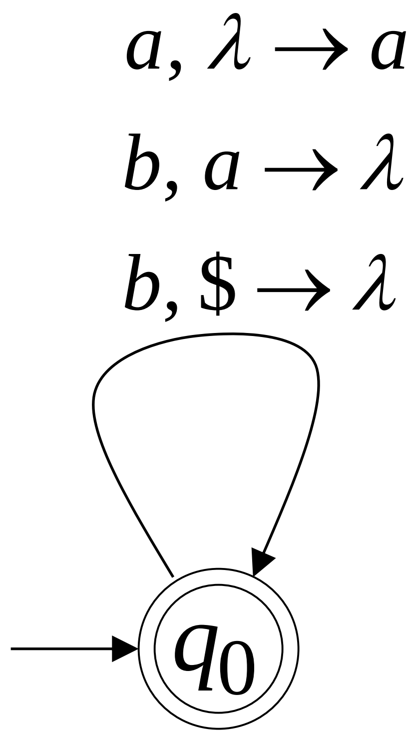
is the language accepted by the NPDA:



Another NPDA example

NPDA M

$$L(M) = \{w: \quad n_a \geq n_b - 1\}$$



Execution Example: Time 0

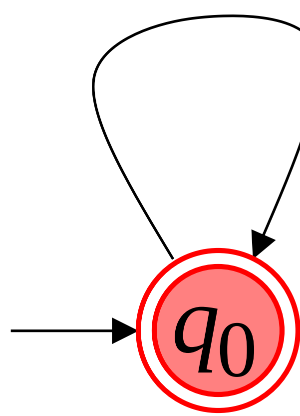
Input

a	a	b

$a, \lambda \rightarrow a$

$b, a \rightarrow \lambda$

$b, \$ \rightarrow \lambda$



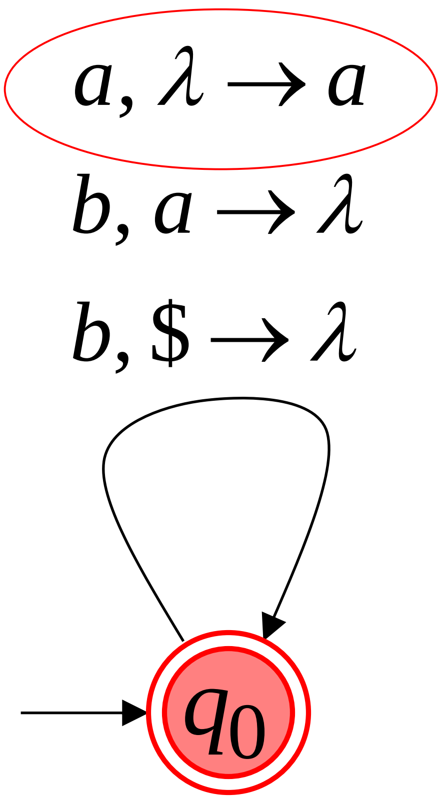
$\$$

Stack

Time 1

Input

a	a	b

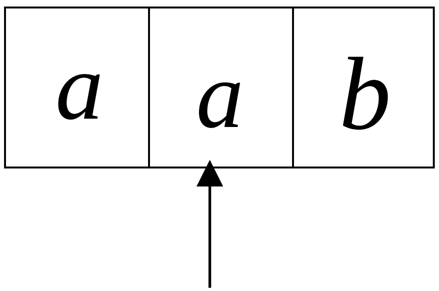


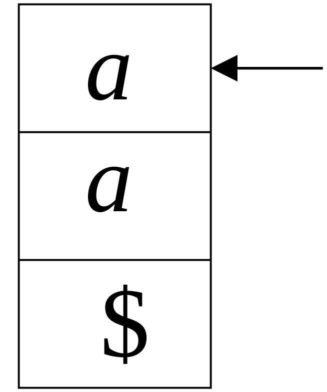
a
$\$$

Stack

Time 2

Input



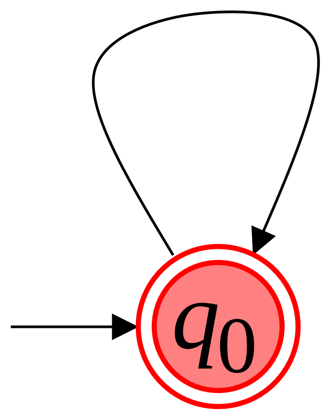


Stack

$$a, \lambda \rightarrow a$$

$$b, a \rightarrow \lambda$$

$$b, \$ \rightarrow \lambda$$



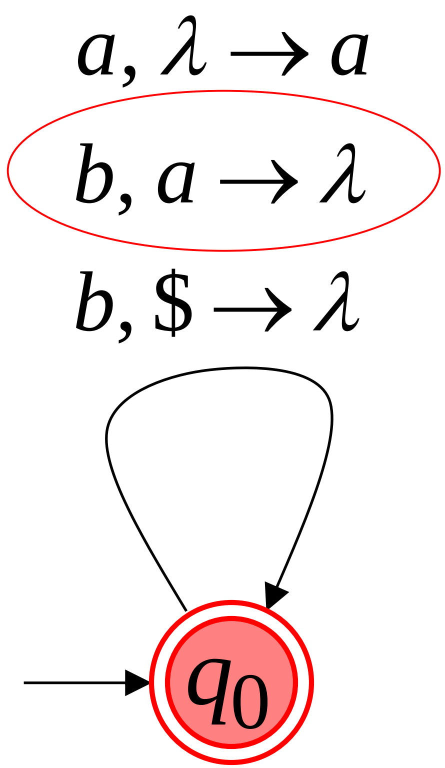
Time 3

Input

a	a	b

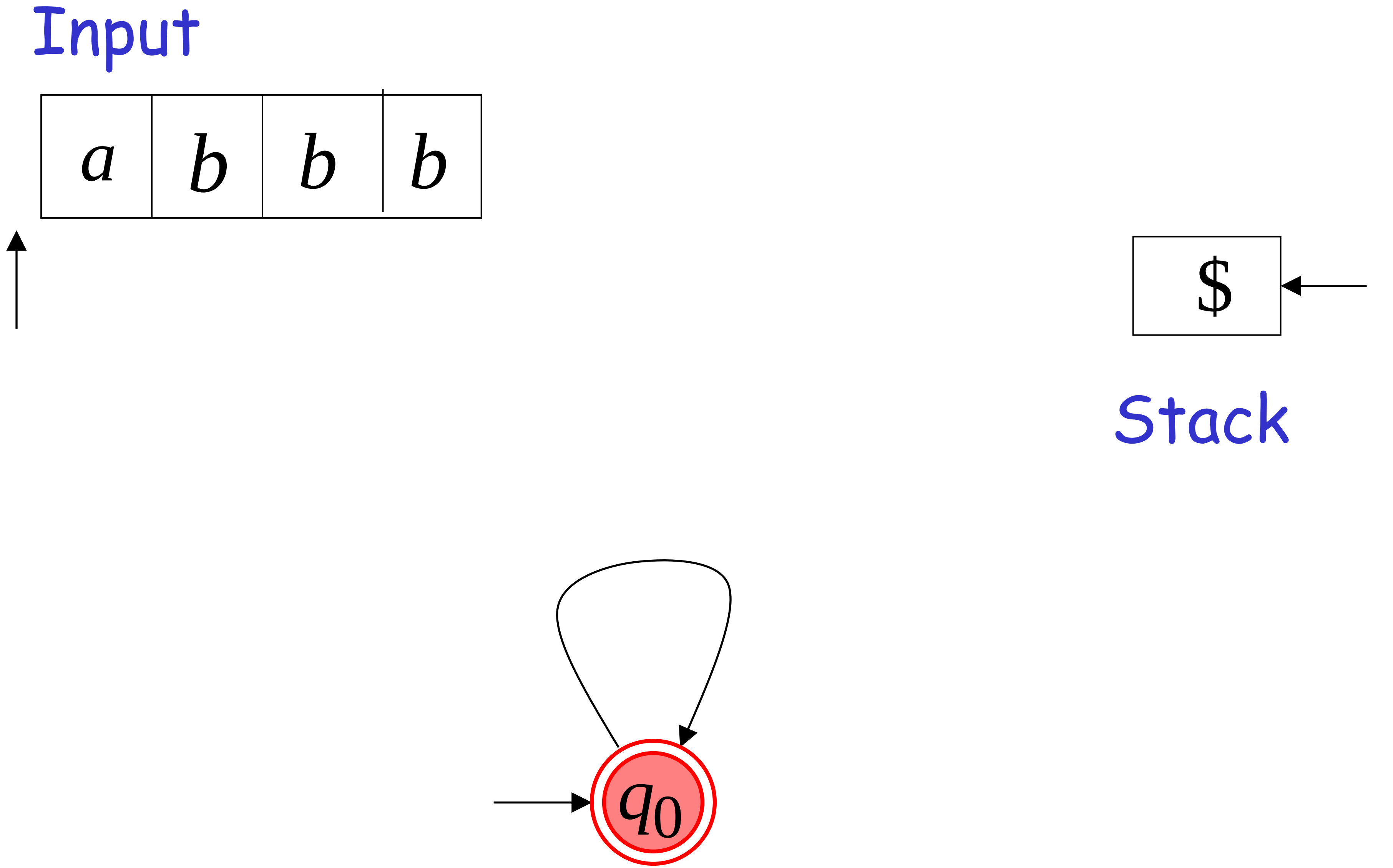
Stack

a
$\$$

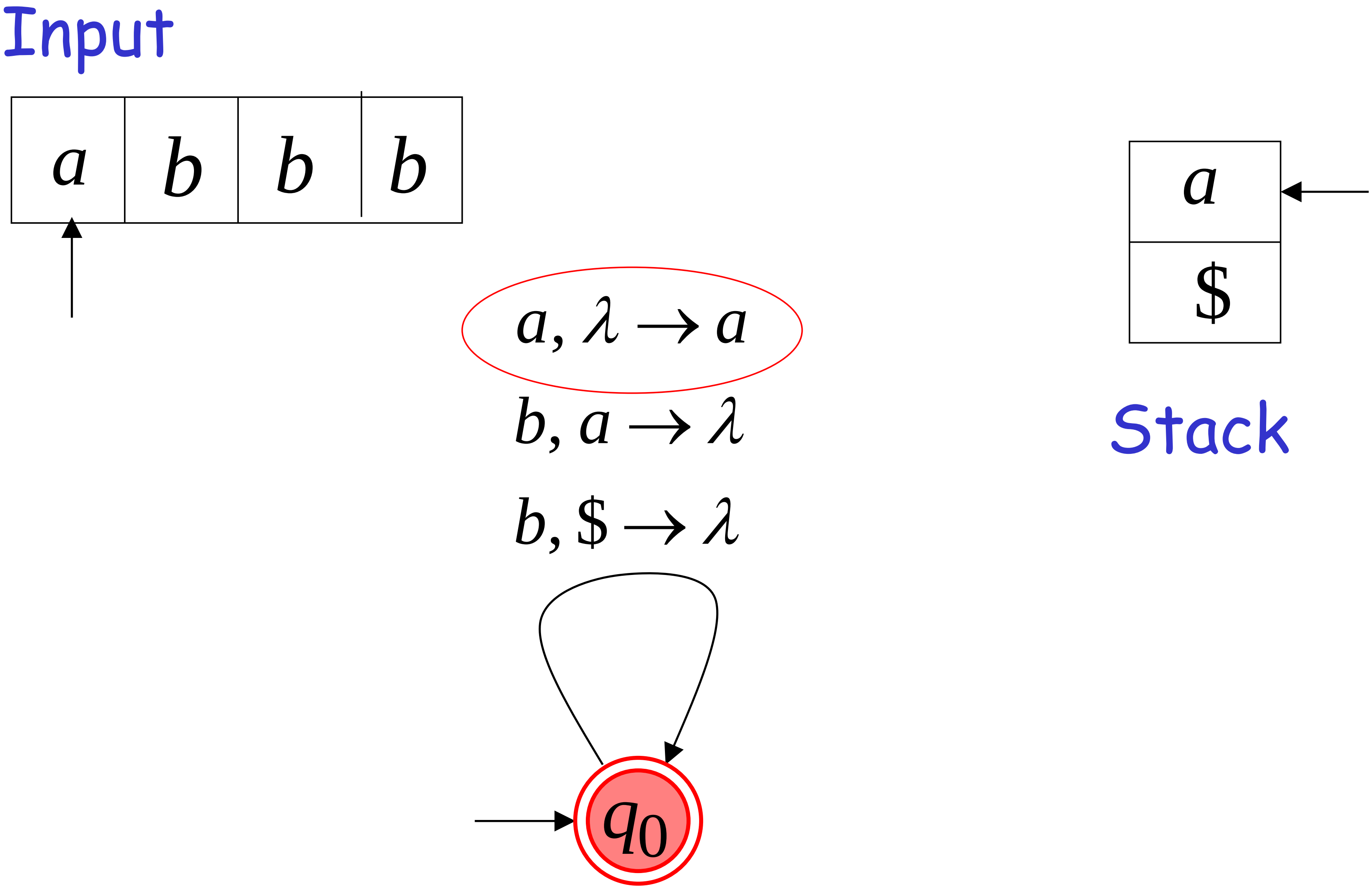


accept

Rejection example: Time 0



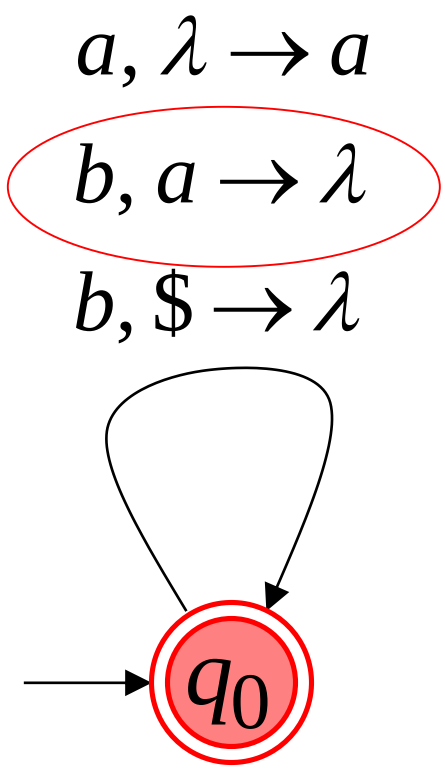
Time 1



Time 2

Input

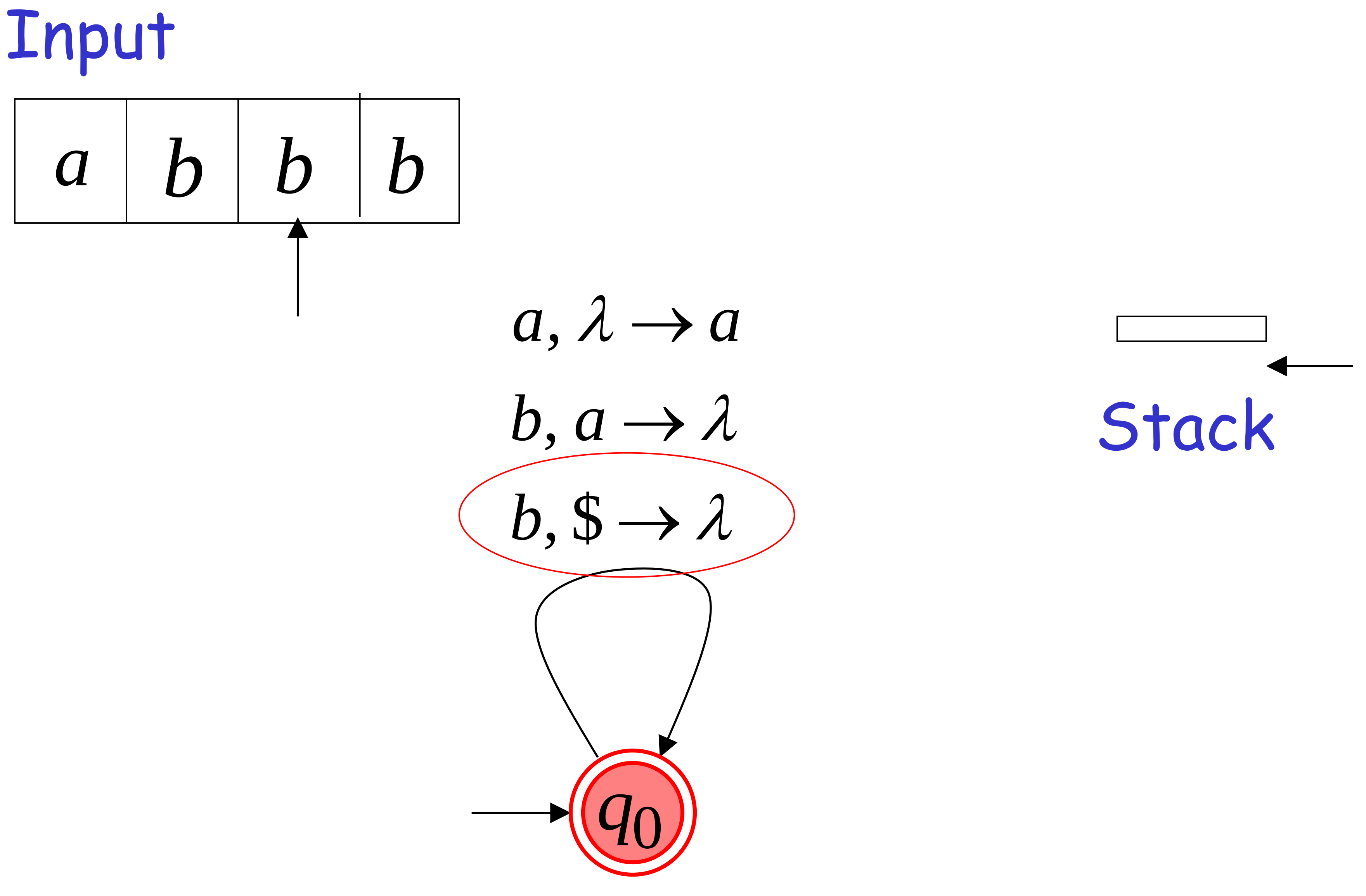
a	b	b	b



$\$$

Stack

Time 3



Input
a
b
b
b
$a, \lambda \rightarrow a$
$b, a \rightarrow \lambda$
$b, \$ \rightarrow \lambda$
q_0
Stack

Time 4

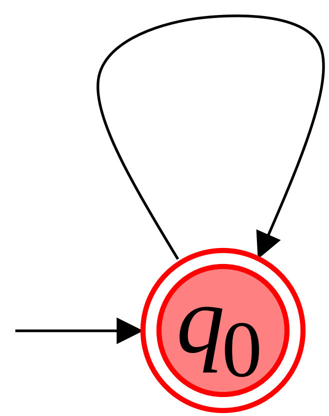
Input

a	b	b	b

$a, \lambda \rightarrow a$

$b, a \rightarrow \lambda$

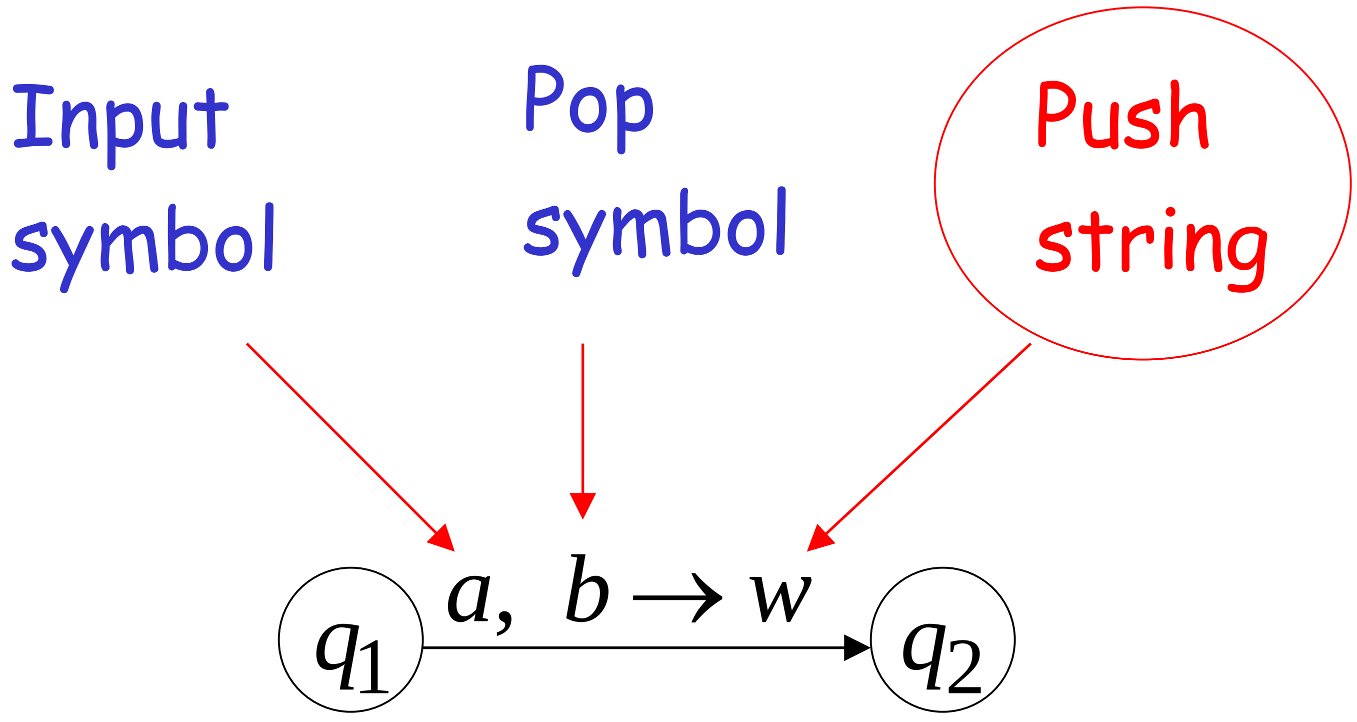
$b, \$ \rightarrow \lambda$



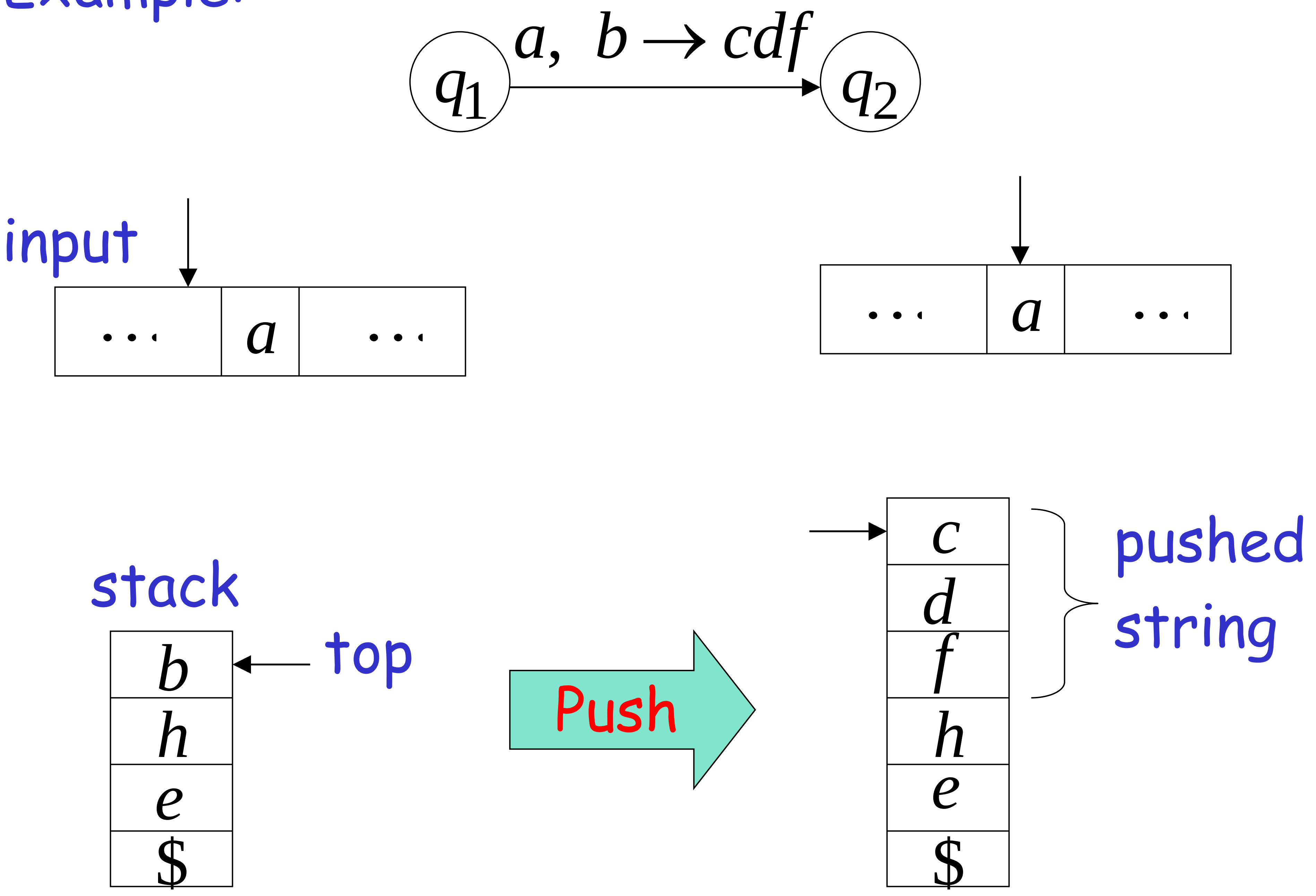
Stack

Halt and <u>Reject</u>

Pushing Strings



Example:


$a, \; b \rightarrow cdf$
q_1
q_2
input
$\cdots$
a
$\cdots$
$\cdots$
a
$\cdots$
stack
b
h
e
$\$$
top
Push
c
d
f
h
e
$\$$
pushed
string

Another NPDA example

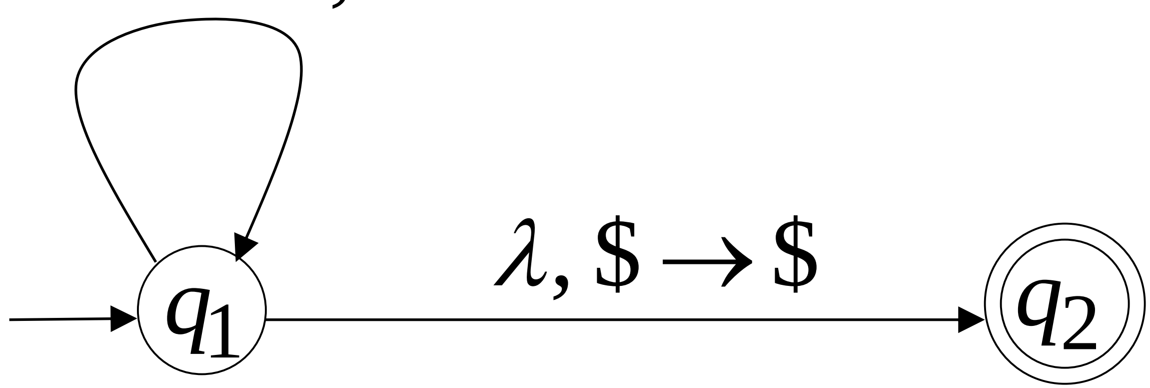
NPDA M

$$L(M) = \{w: \quad n_a = n_b\}$$

$$a, \$ \to 0\$ \qquad b, \$ \to 1\$$$

$$a, 0 \to 00 \qquad b, 1 \to 11$$

$$a, 1 \to \lambda \qquad b, 0 \to \lambda$$



Execution Example: Time 0

Input

a	b	b	a	a	b

$\$$

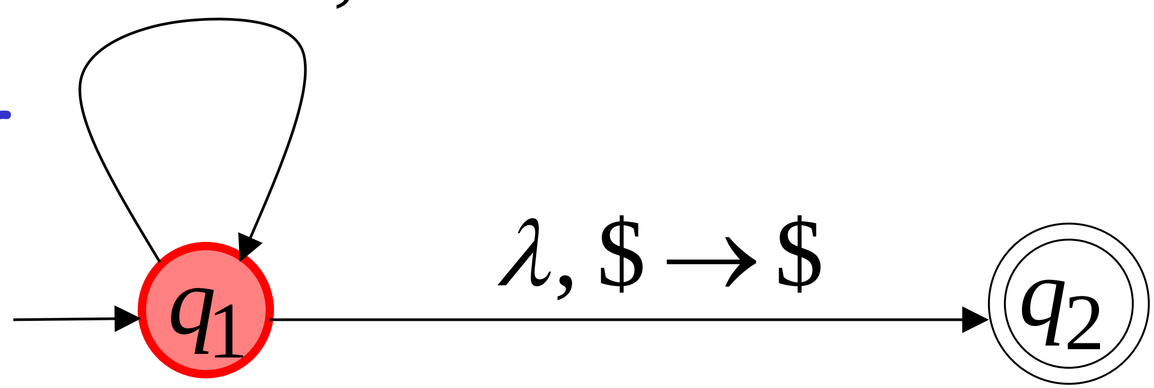
Stack

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

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

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

current state



Time 1

Input

a	b	b	a	a	b

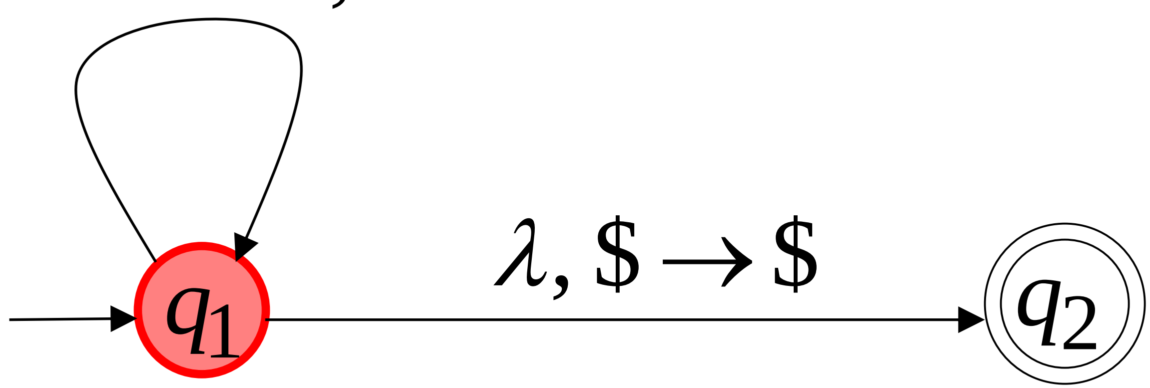
0
\$

Stack

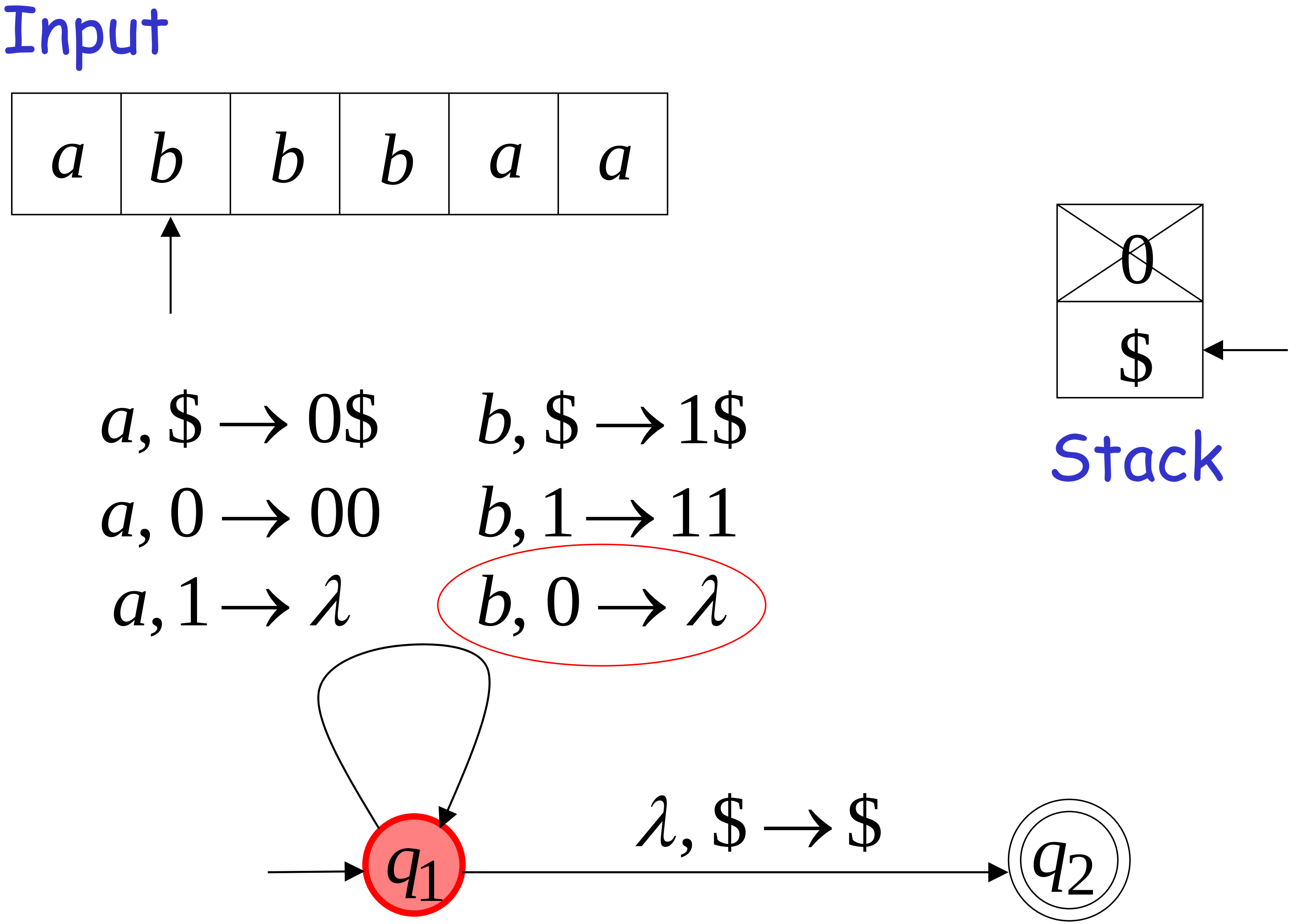
$a, \$ \rightarrow 0\$ \qquad b, \$ \rightarrow 1\$$

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

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



Time 2



Input
a b b b a a
0
$
Stack
$a, \$ \rightarrow 0\$$
$a, 0 \rightarrow 00$
$a, 1 \rightarrow \lambda$
$b, \$ \rightarrow 1\$$
$b, 1 \rightarrow 11$
$b, 0 \rightarrow \lambda$
$\lambda, \$ \rightarrow \$$
q_1
q_2

Time 3

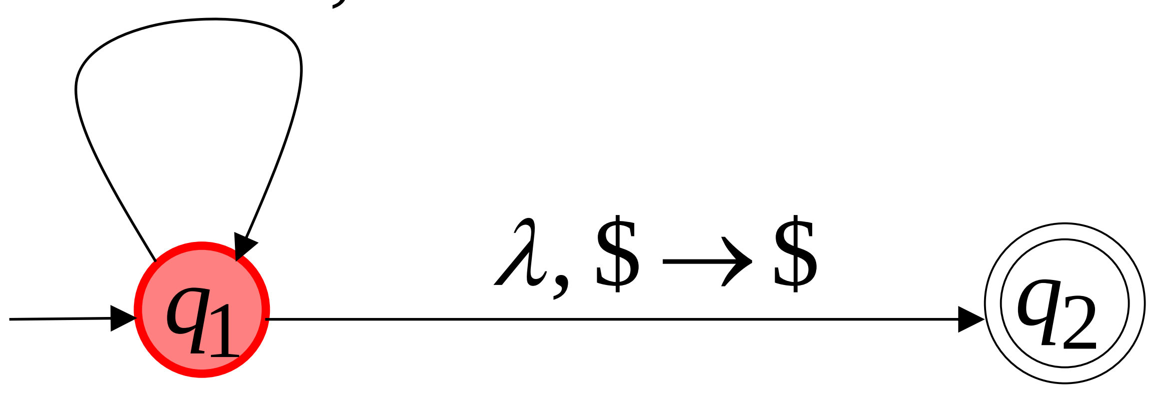
Input

a	b	b	b	a	a

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

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

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



1
$

Stack

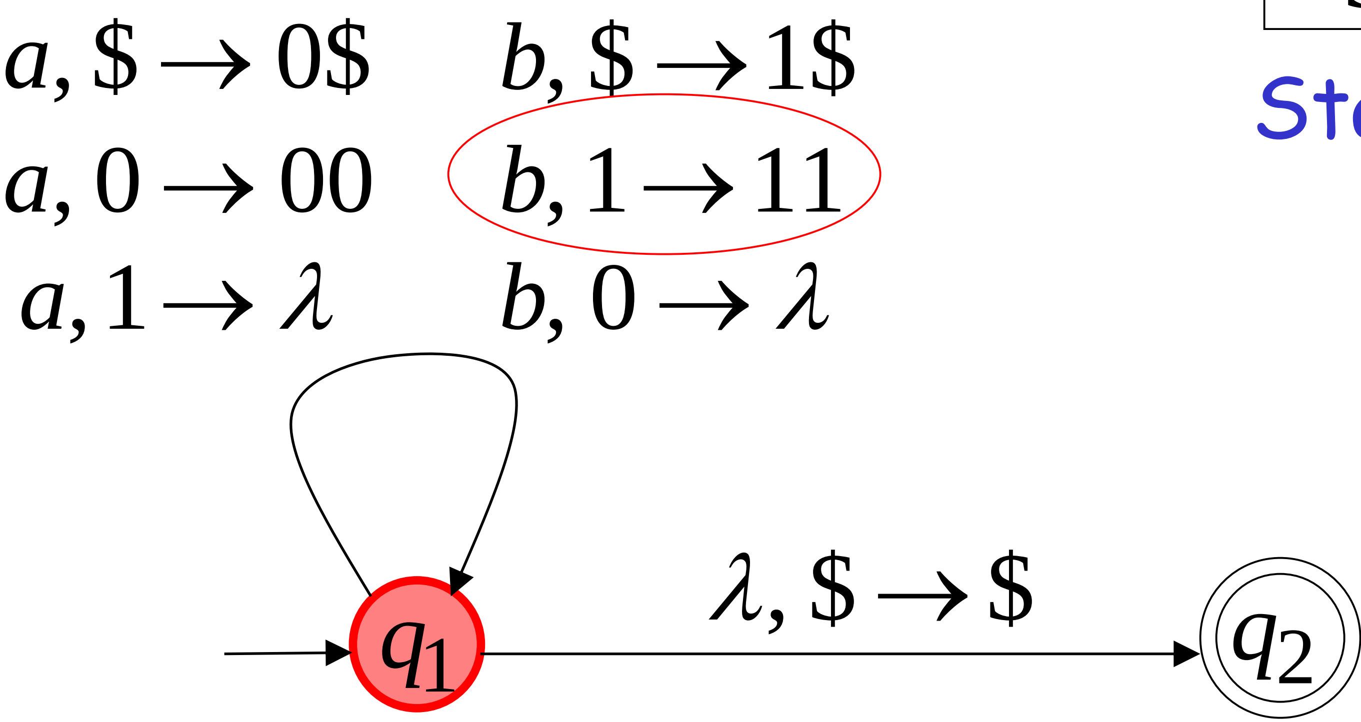
Time 4

Input

a	b	b	b	a	a

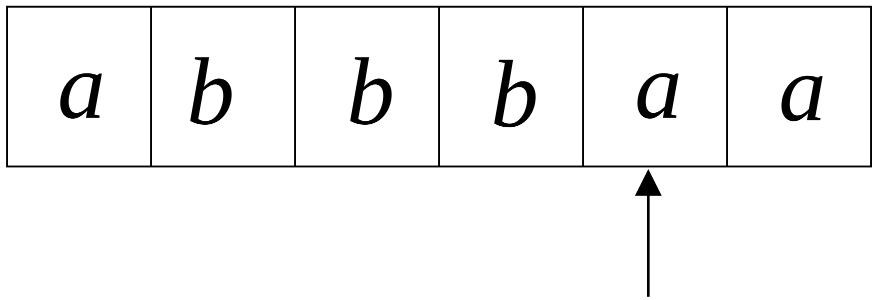
1
1
$

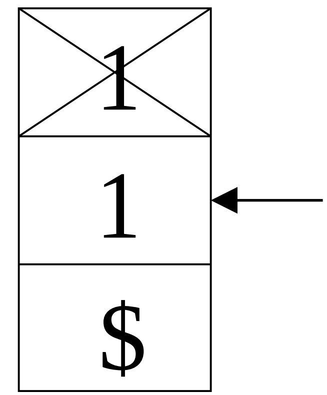
Stack



Time 5

Input



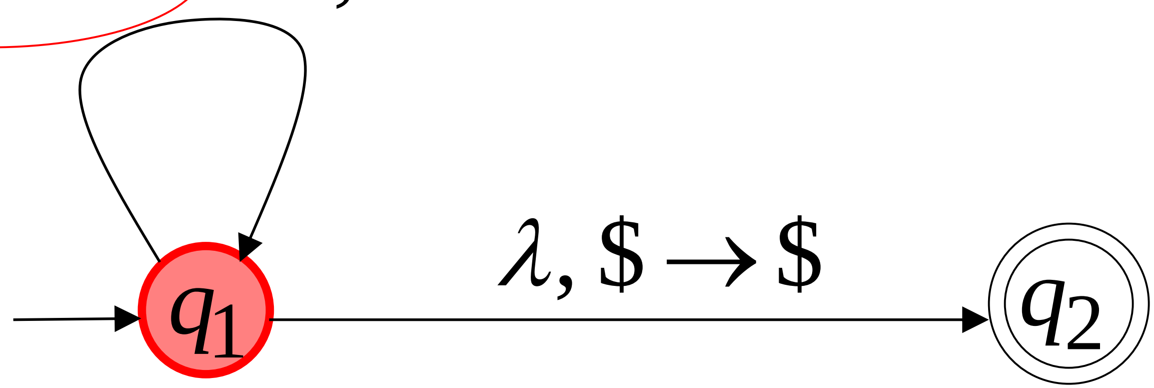


Stack

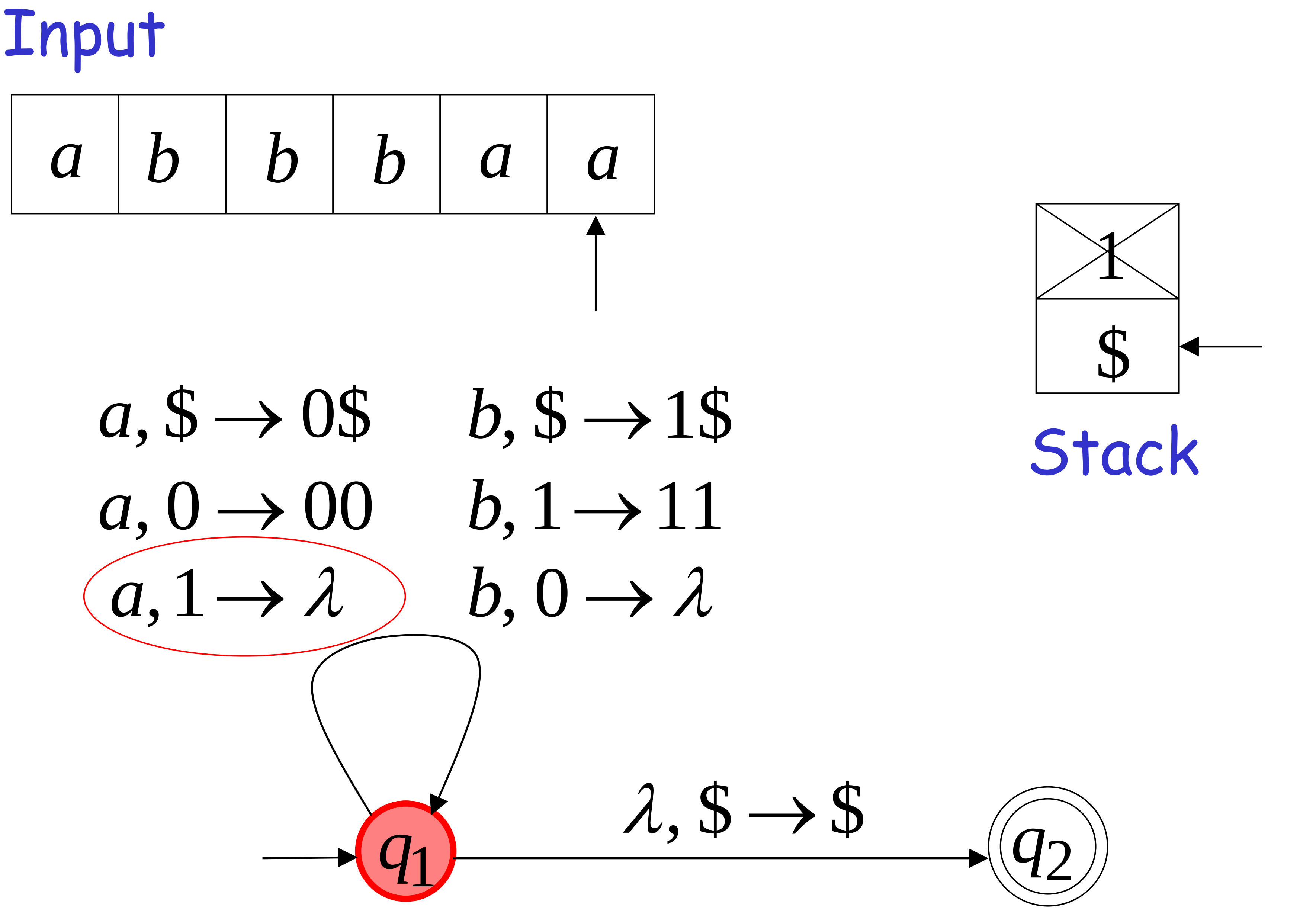
$a, \$ \rightarrow 0\$$ $\quad$ $b, \$ \rightarrow 1\$$

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

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



Time 6

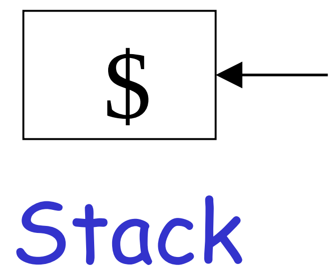


Input
a b b b a a
1
$
Stack
$a, \$ \rightarrow 0\$$
$b, \$ \rightarrow 1\$$
$a, 0 \rightarrow 00$
$b, 1 \rightarrow 11$
$a, 1 \rightarrow \lambda$
$b, 0 \rightarrow \lambda$
$\lambda, \$ \rightarrow \$$
q_1
q_2

Time 7

Input

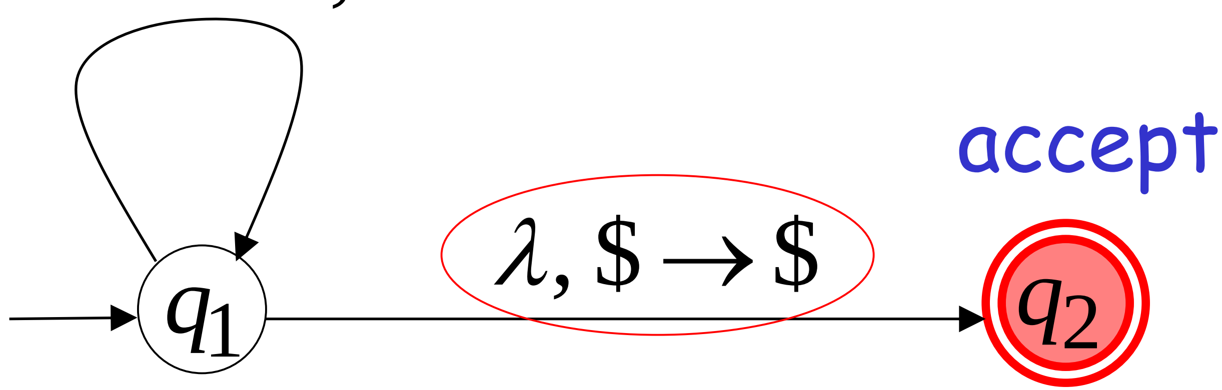
a	b	b	b	a	a



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

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

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



Deterministic Pushdown Automaton

- Let $M = (Q, \Sigma, \Gamma, \delta, q0, Z0, F)$ be a PDA
- M is deterministic if ($a \in \Sigma$ & $X \in \Gamma$):

  - $\delta(q, a, X)$ has at most one element
  - If $\delta(q, \Lambda, X) \neq \emptyset$ then $\delta(q, a, X) = \emptyset$ for all $a \in \Sigma$

Deterministic PDAs

In other words:

- There is no configuration where the machine has a "choice" of moves
  - Each transition has at most 1 element.
  - If you can make a λ-transition from a state with a given symbol on the stack,
    - You cannot make that same transition on any tape input symbol.

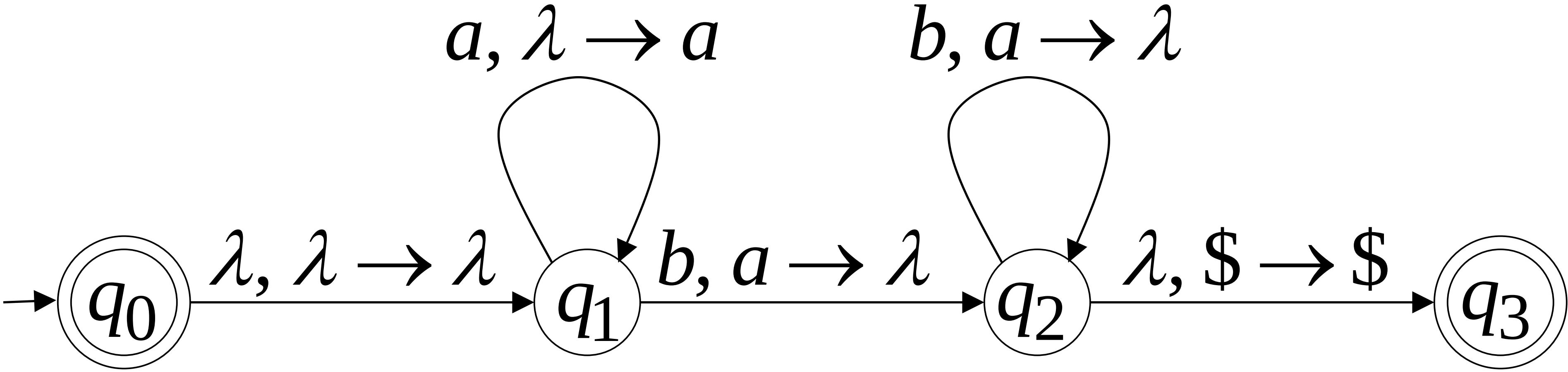
deterministic context-free language

- A language L is a deterministic context-free language (DCFL) if there is a DPDA that accepts L

deterministic context-free language

Example of DCFL:

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

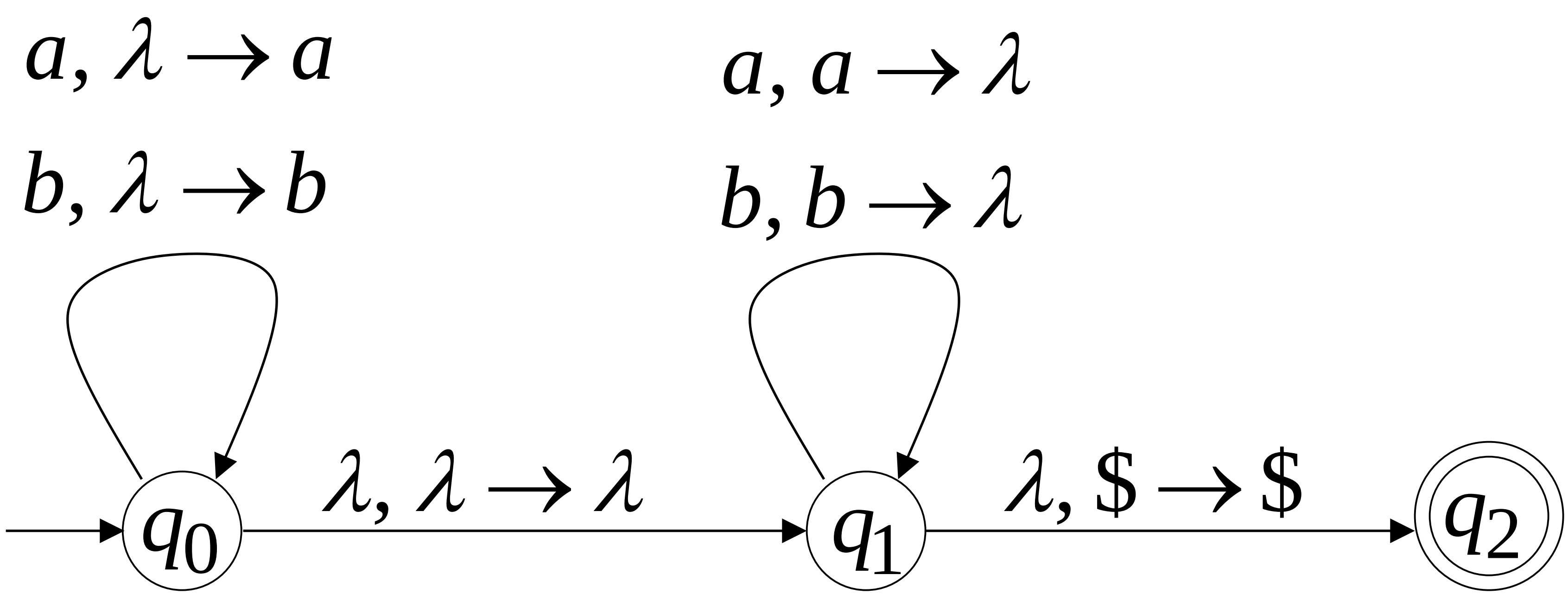


Another NPDA example

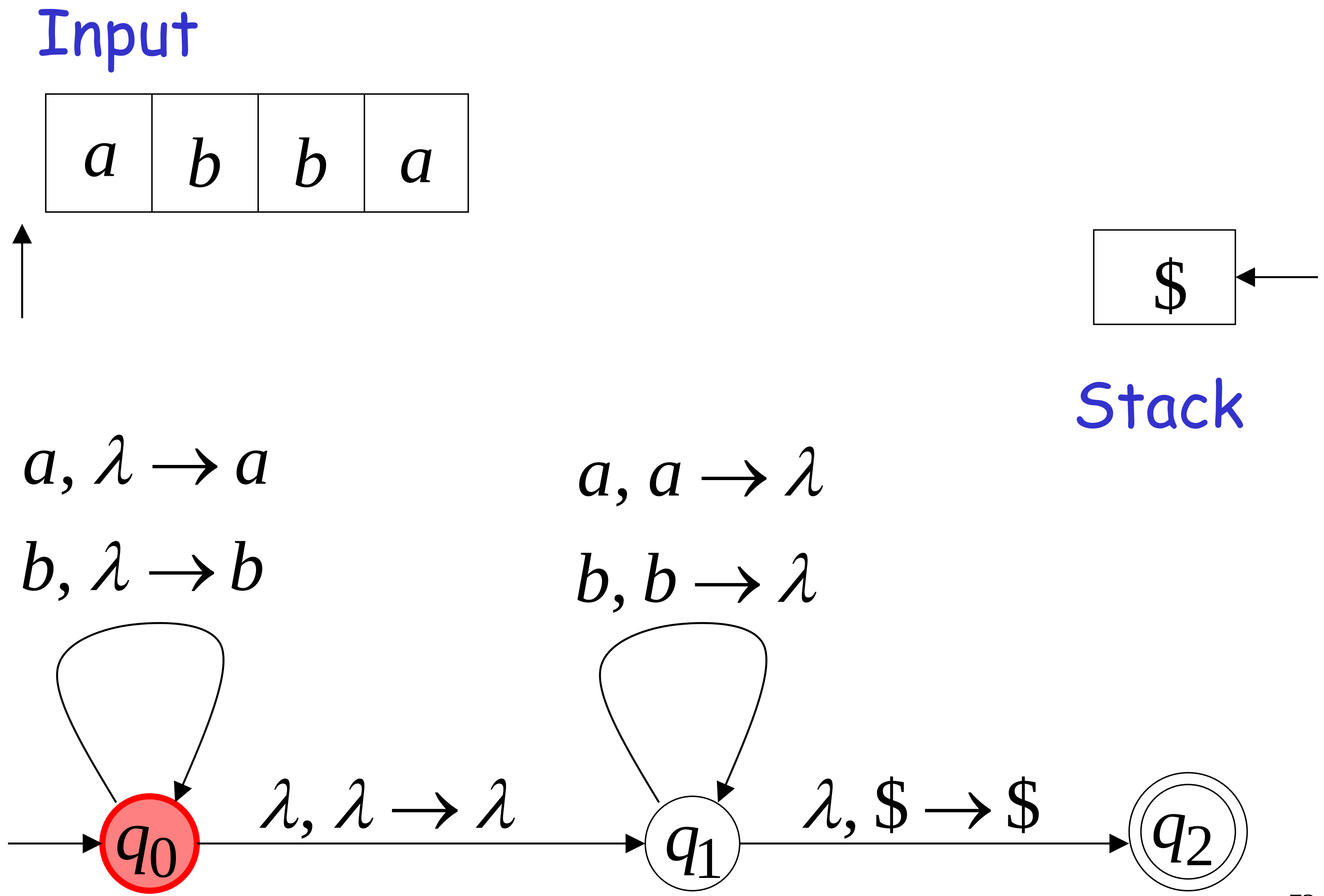
NPDA M

Example of NCFL:

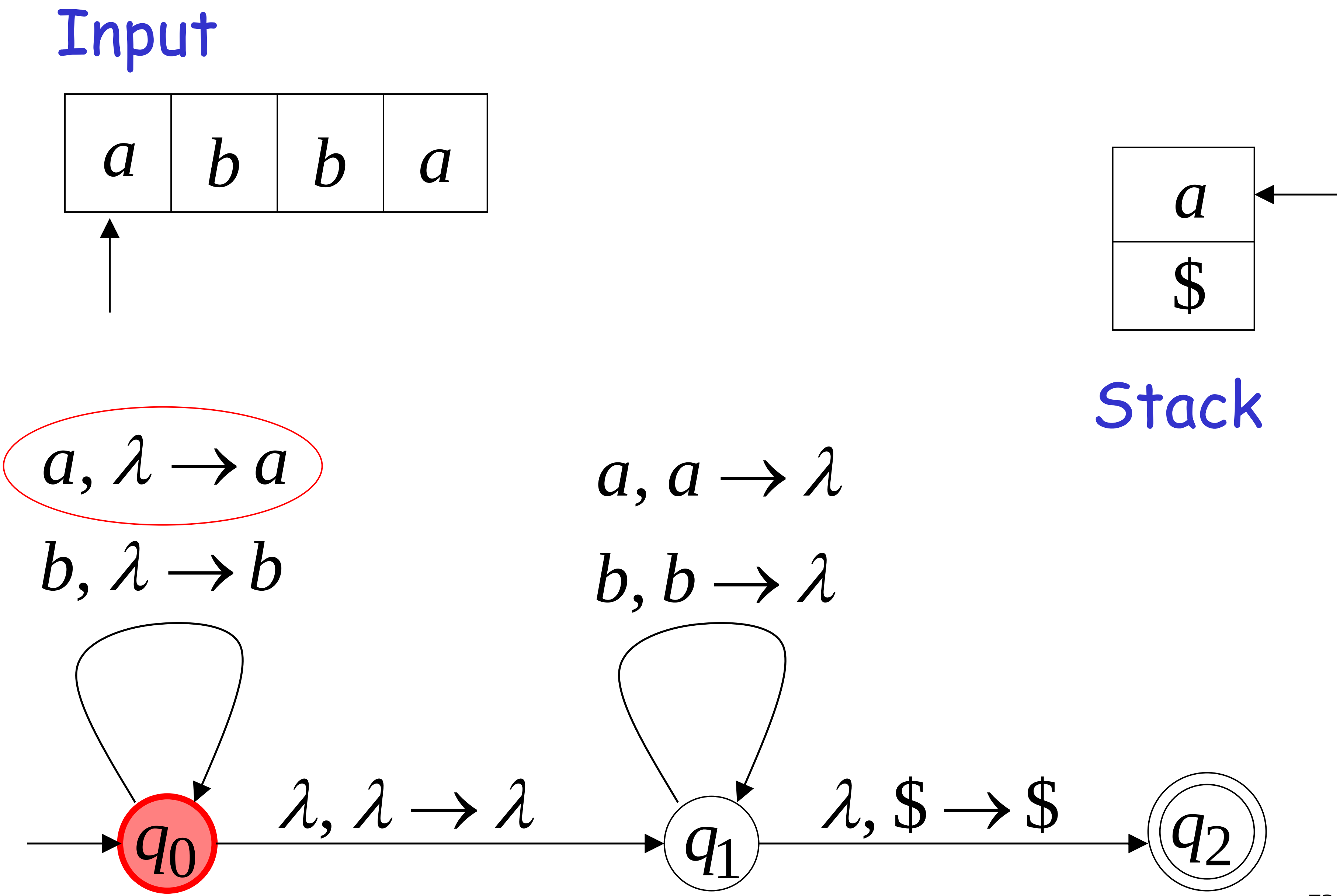
$$L(M) = \{ww^R\}$$



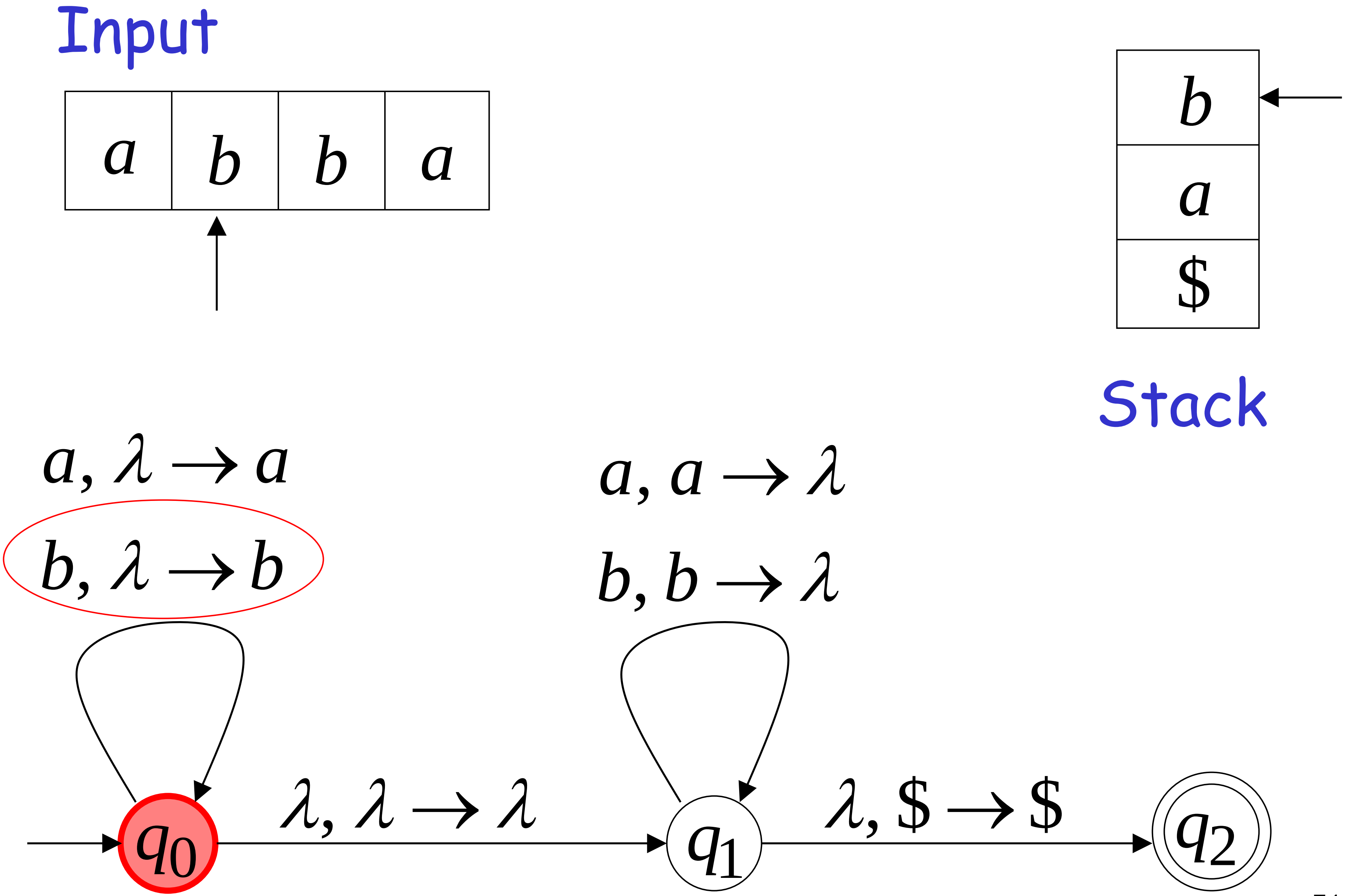
Execution Example: Time 0



Time 1

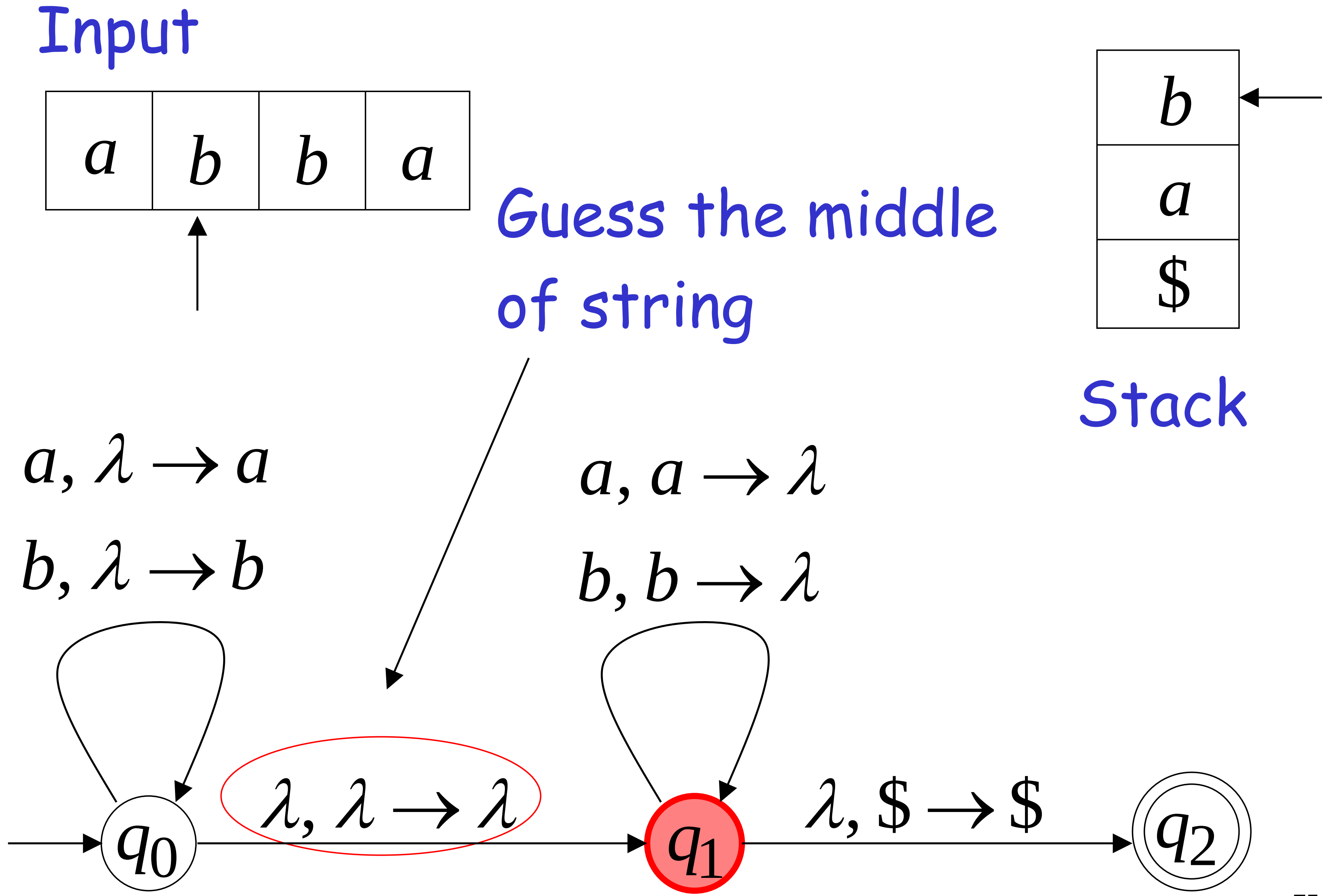


Time 2

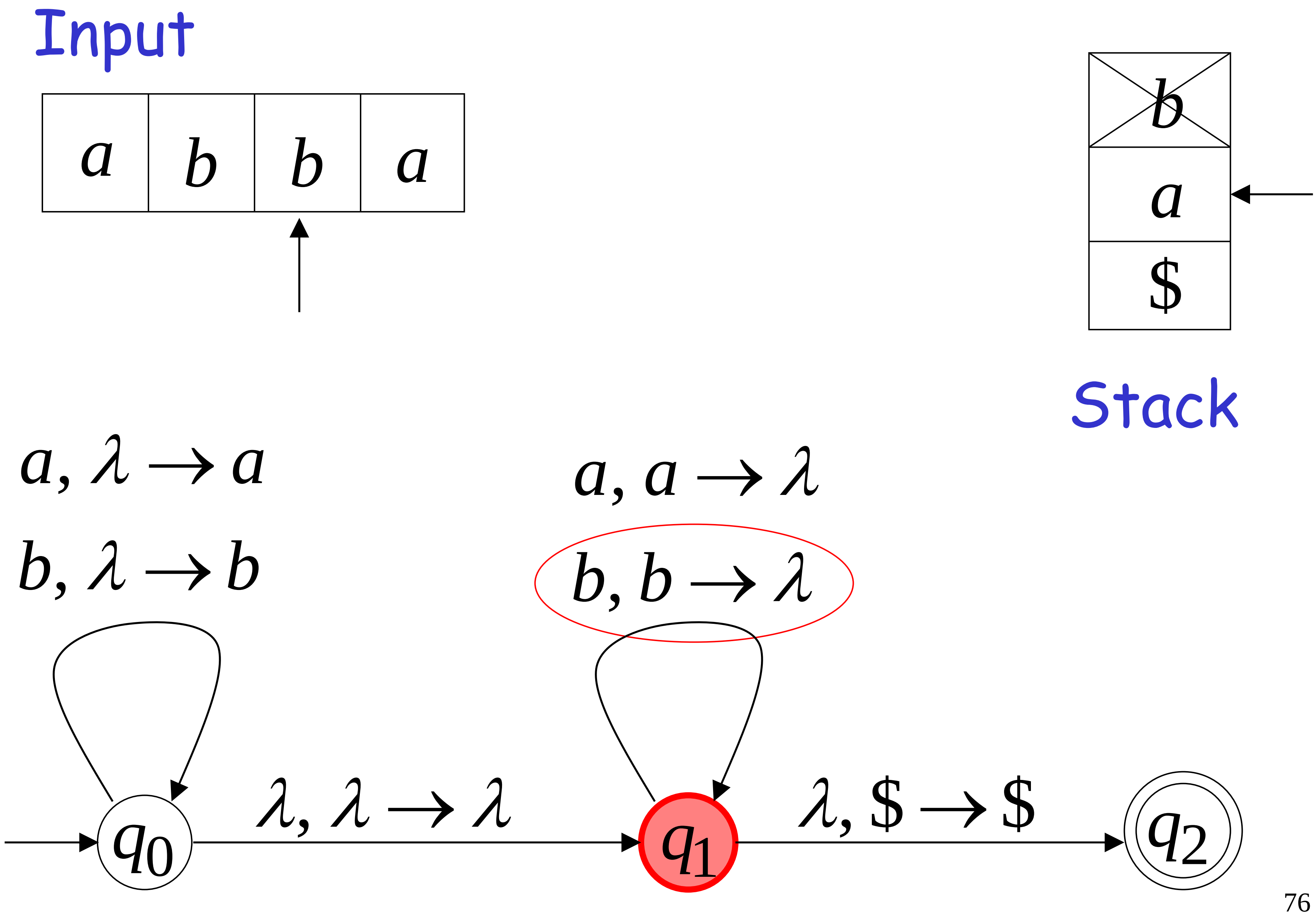


Input
a
b
b
a
b
a
$
Stack
$a, \lambda \rightarrow a$
$b, \lambda \rightarrow b$
$a, a \rightarrow \lambda$
$b, b \rightarrow \lambda$
$\lambda, \lambda \rightarrow \lambda$
$\lambda, \$ \rightarrow \$$
q_0
q_1
q_2

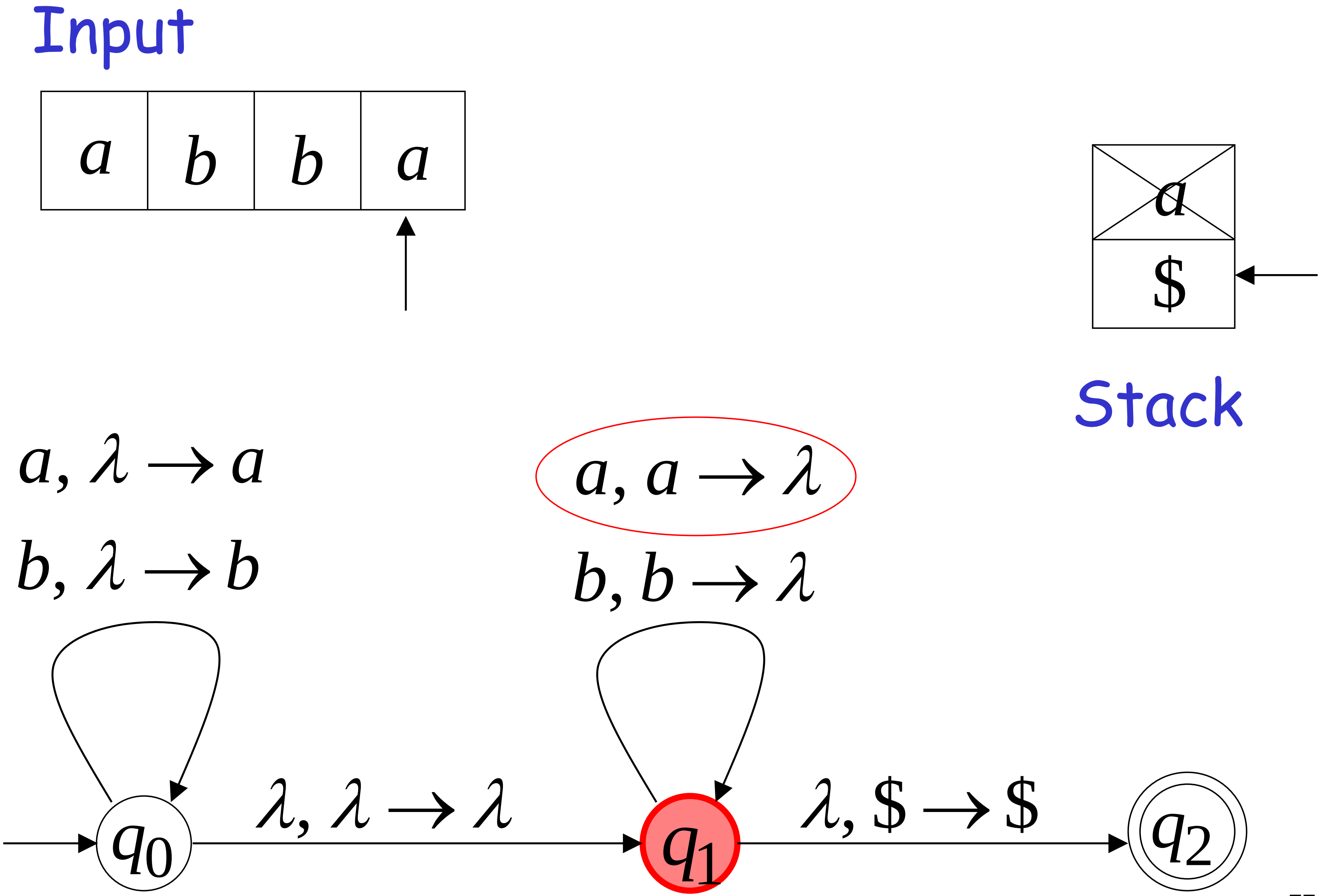
Time 3


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

Time 4

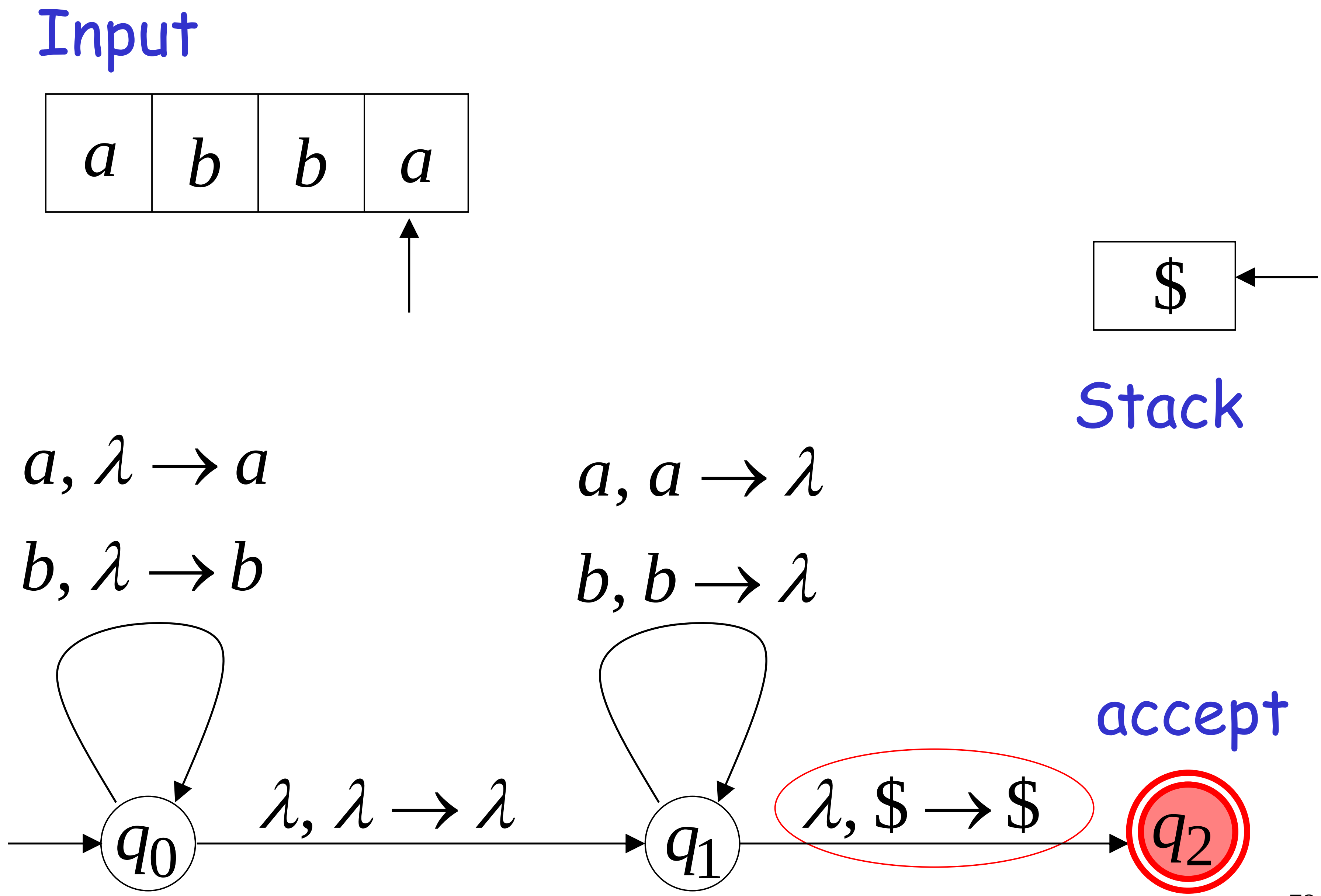

Input
a b b a
$\cancel{b}$
a
$\$$
Stack
$a, \lambda \to a$
$b, \lambda \to b$
$a, a \to \lambda$
$b, b \to \lambda$
$\lambda, \lambda \to \lambda$
$\lambda, \$ \to \$$
q_0
q_1
q_2

Time 5

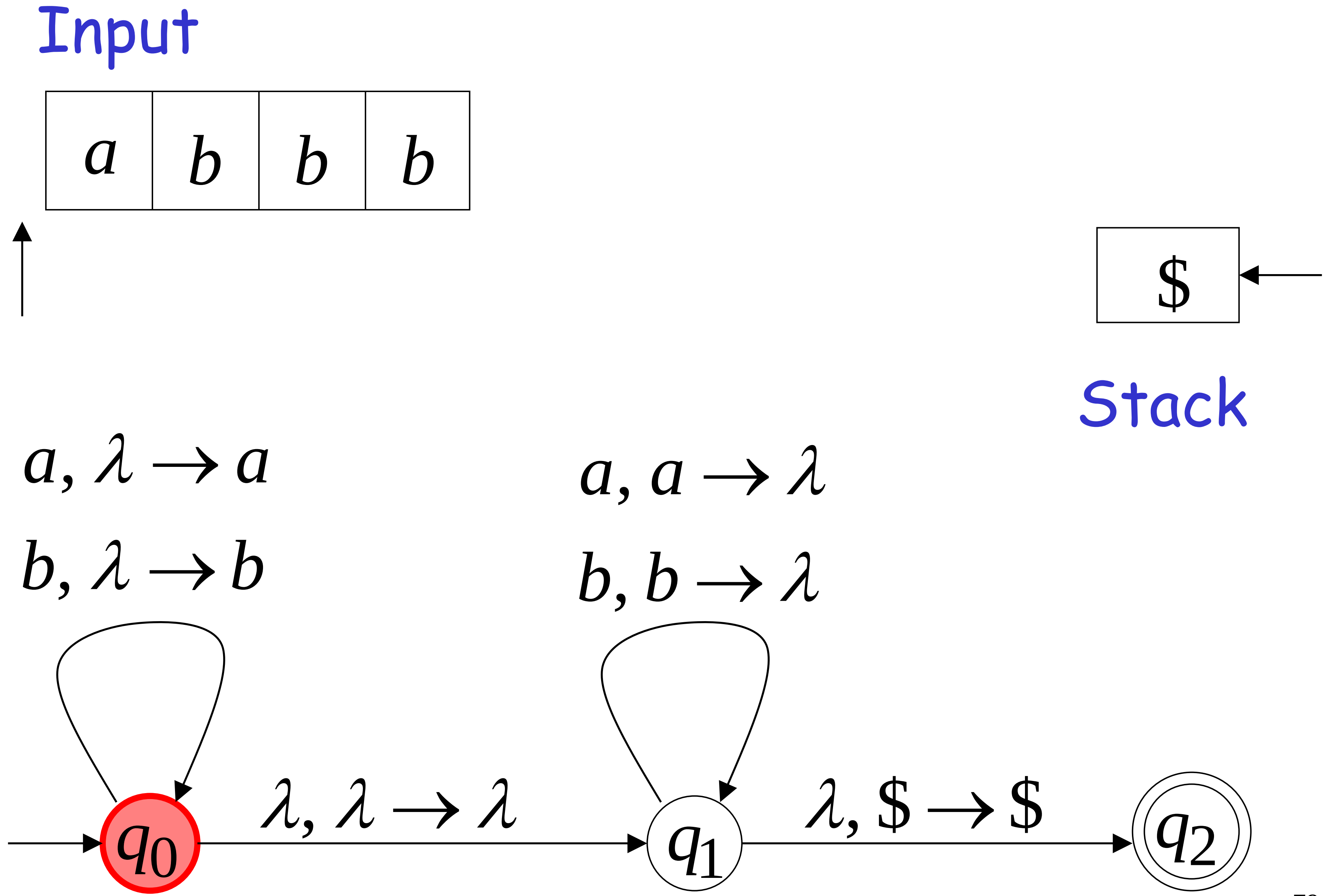


Input
a
b
b
a
a
$
Stack
$a, \lambda \to a$
$b, \lambda \to b$
$a, a \to \lambda$
$b, b \to \lambda$
$\lambda, \lambda \to \lambda$
$\lambda, \$ \to \$$
q_0
q_1
q_2

Time 6

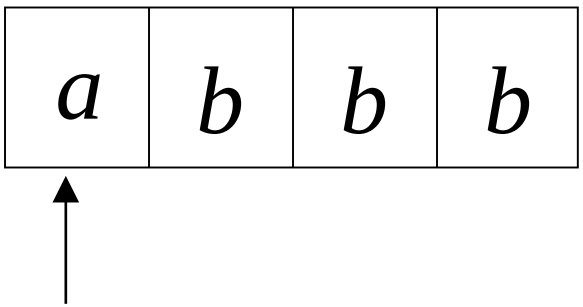


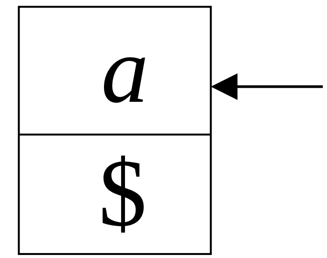
Rejection Example: Time 0



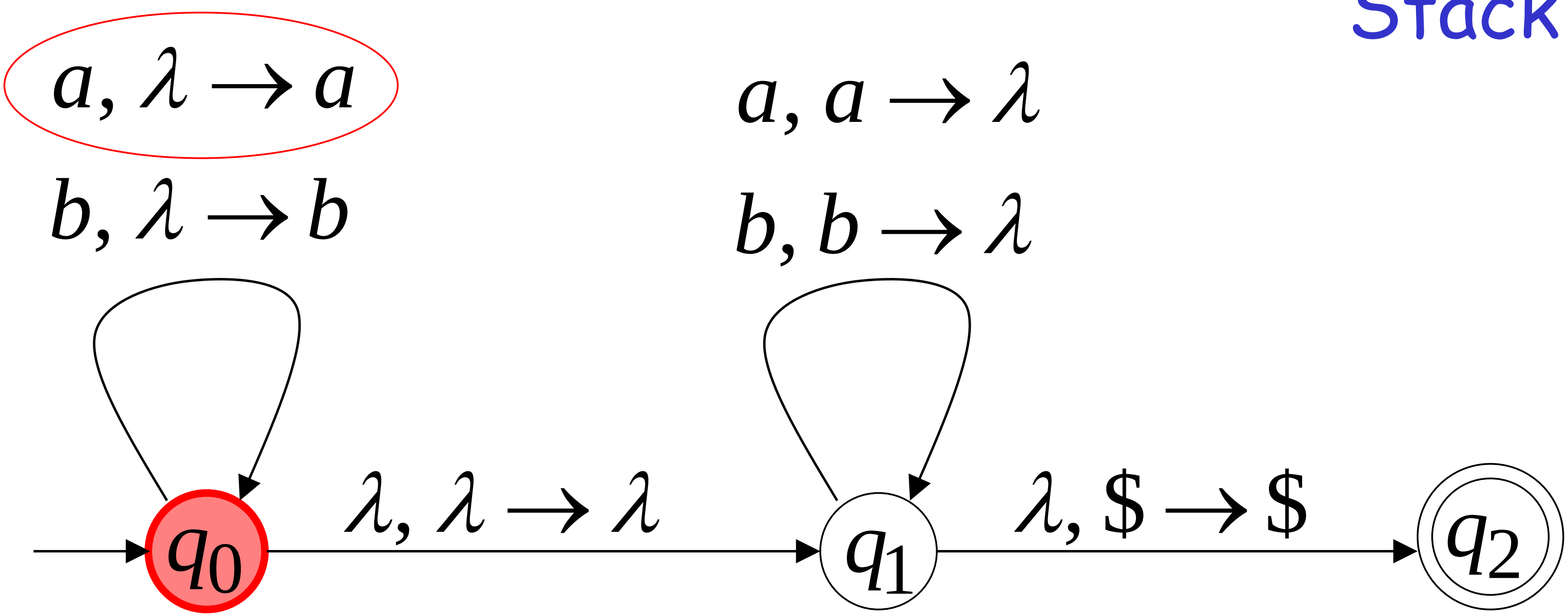
Time 1

Input

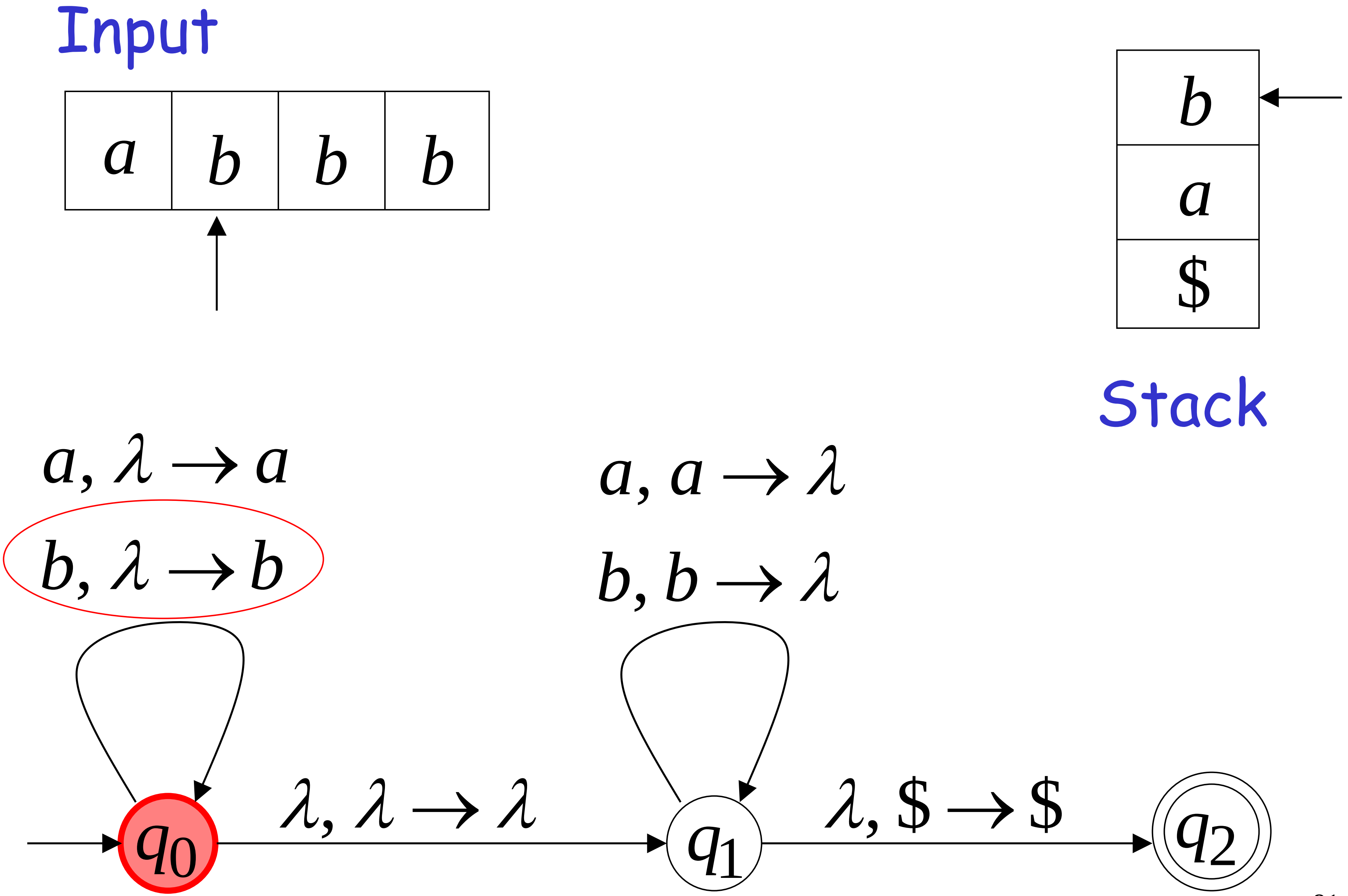




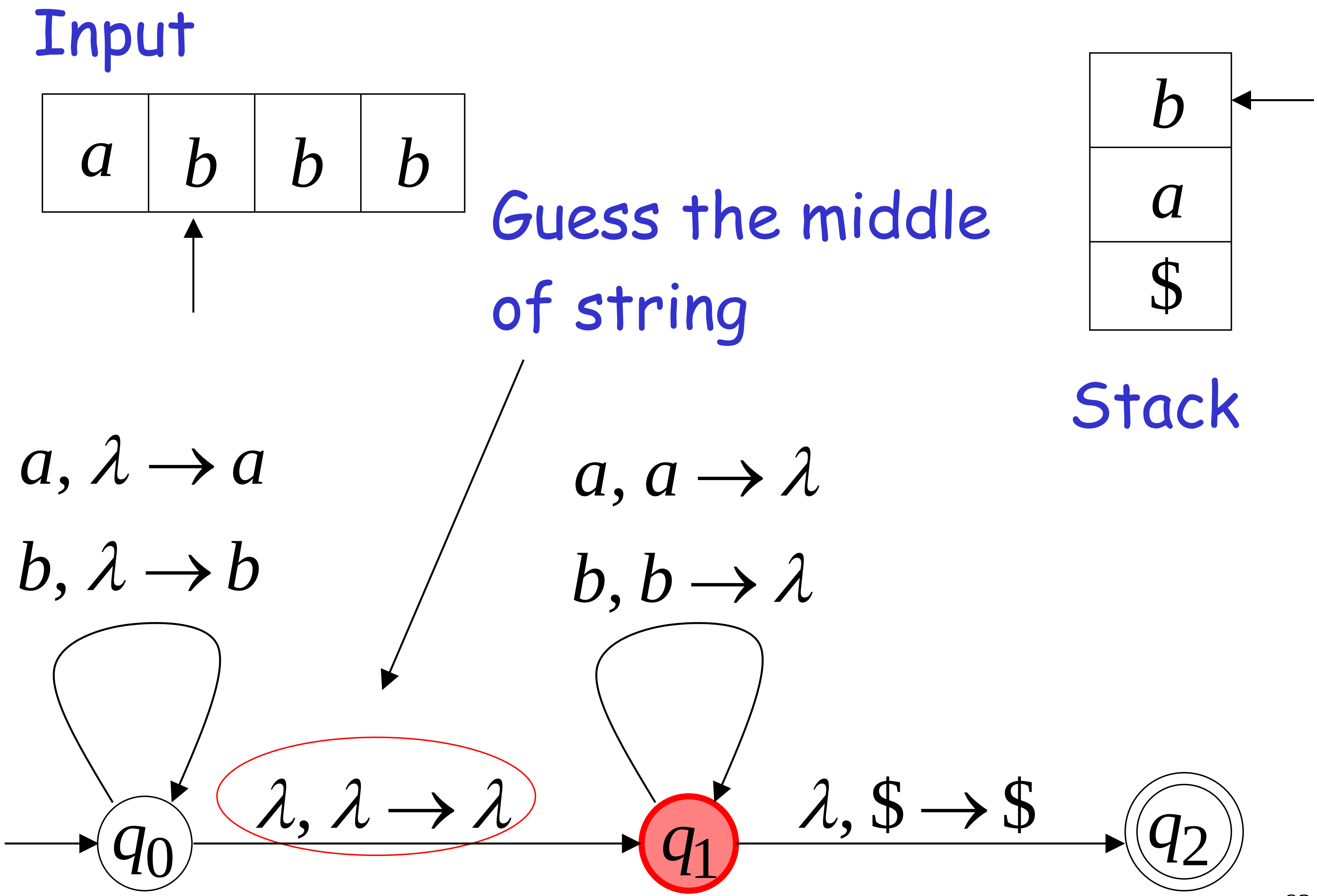
Stack



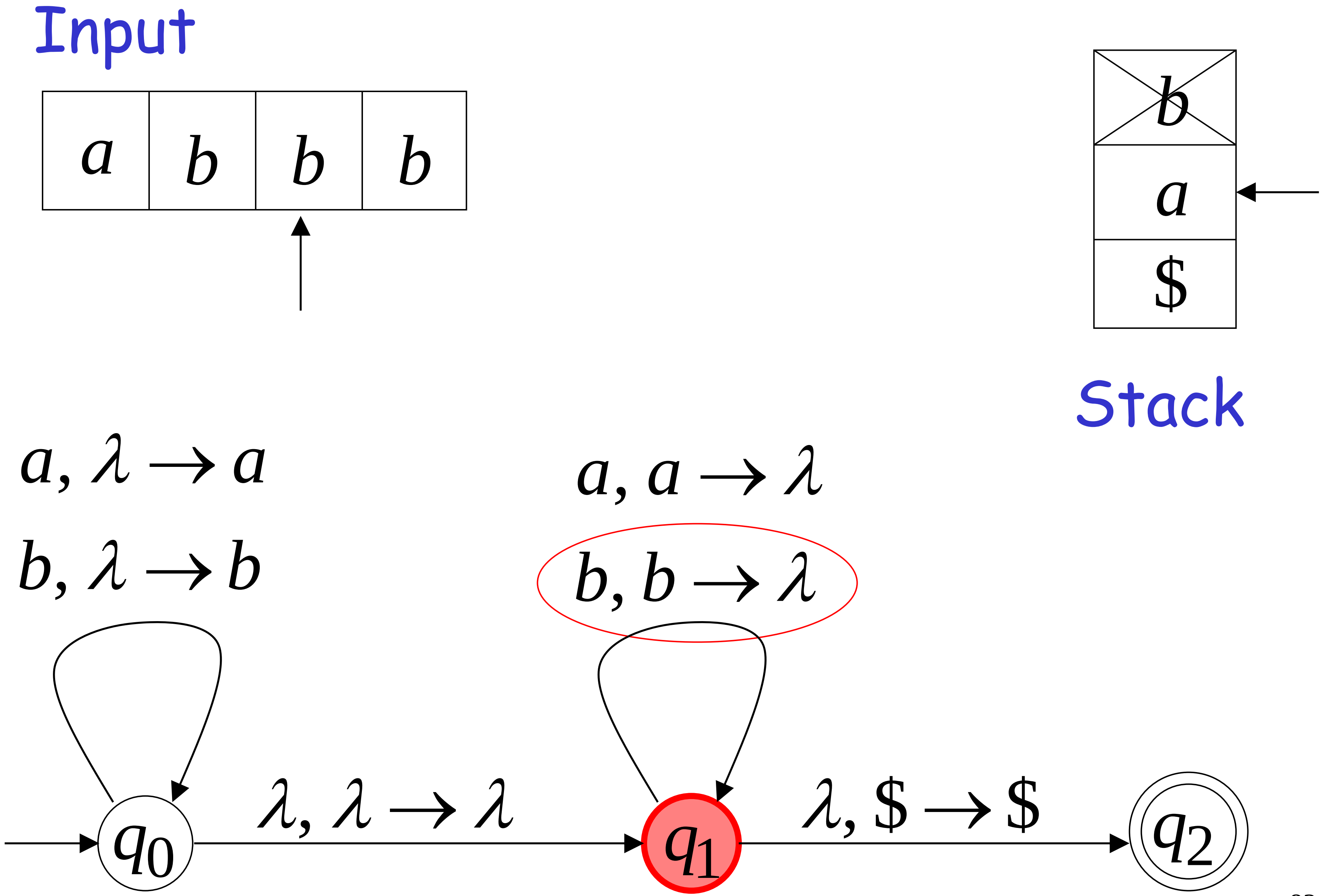
Time 2


Input
a b b b
b
a
$\$$
Stack
$a, \lambda \rightarrow a$
$b, \lambda \rightarrow b$
$a, a \rightarrow \lambda$
$b, b \rightarrow \lambda$
$\lambda, \lambda \rightarrow \lambda$
$\lambda, \$ \rightarrow \$$
q_0
q_1
q_2

Time 3

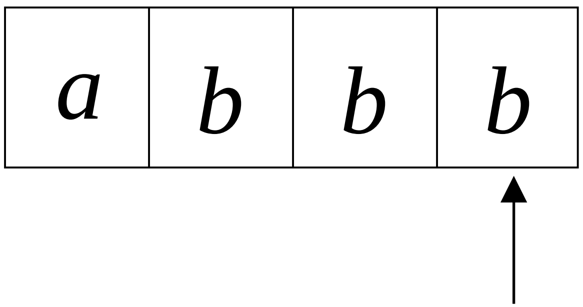

Input
a b b b
Guess the middle of string
b
a
$
Stack
$a, \lambda \to a$
$b, \lambda \to b$
$a, a \to \lambda$
$b, b \to \lambda$
$\lambda, \lambda \to \lambda$
$\lambda, \$ \to \$$
q_0
q_1
q_2

Time 4



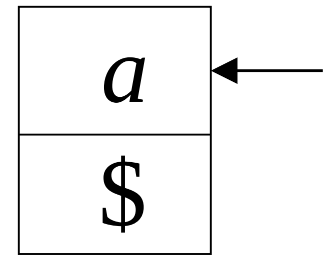
Time 5

Input

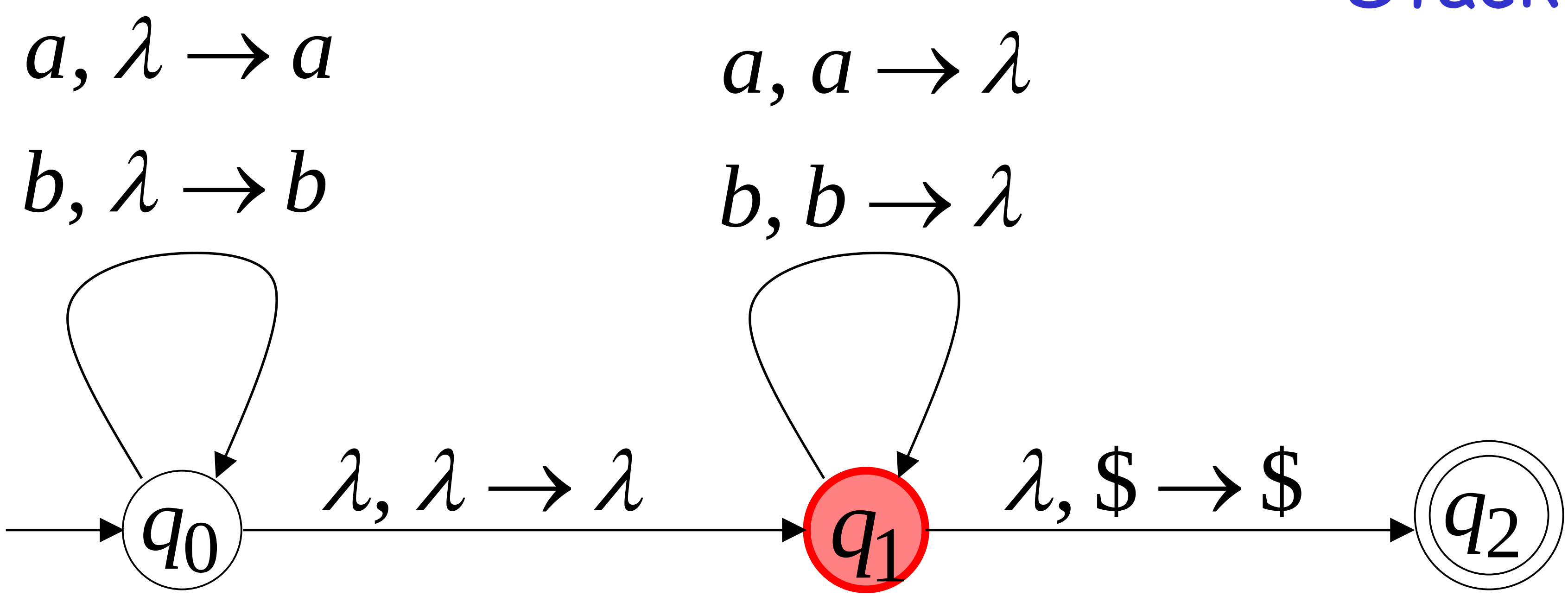


There is no possible transition.

Input is not consumed



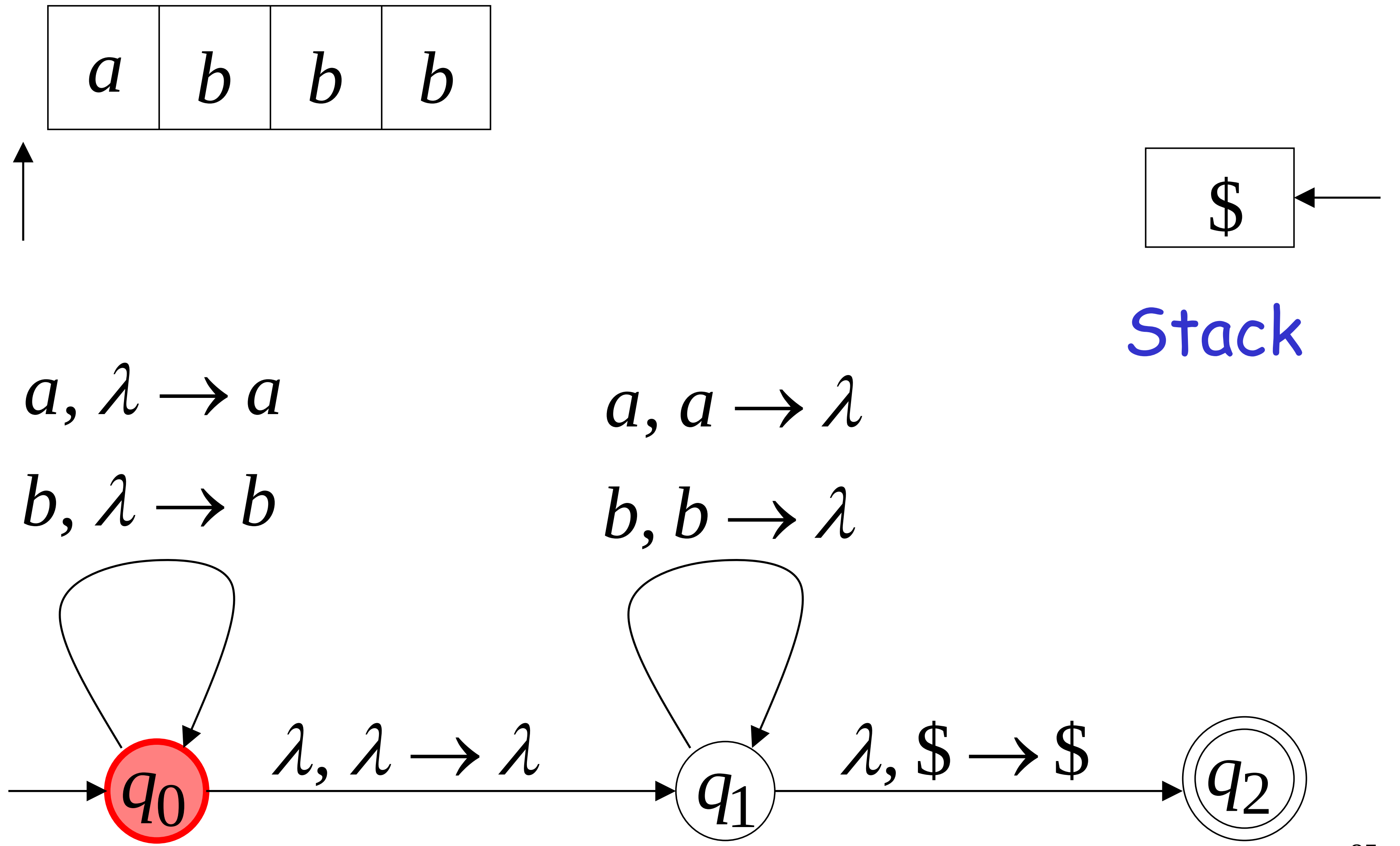
Stack



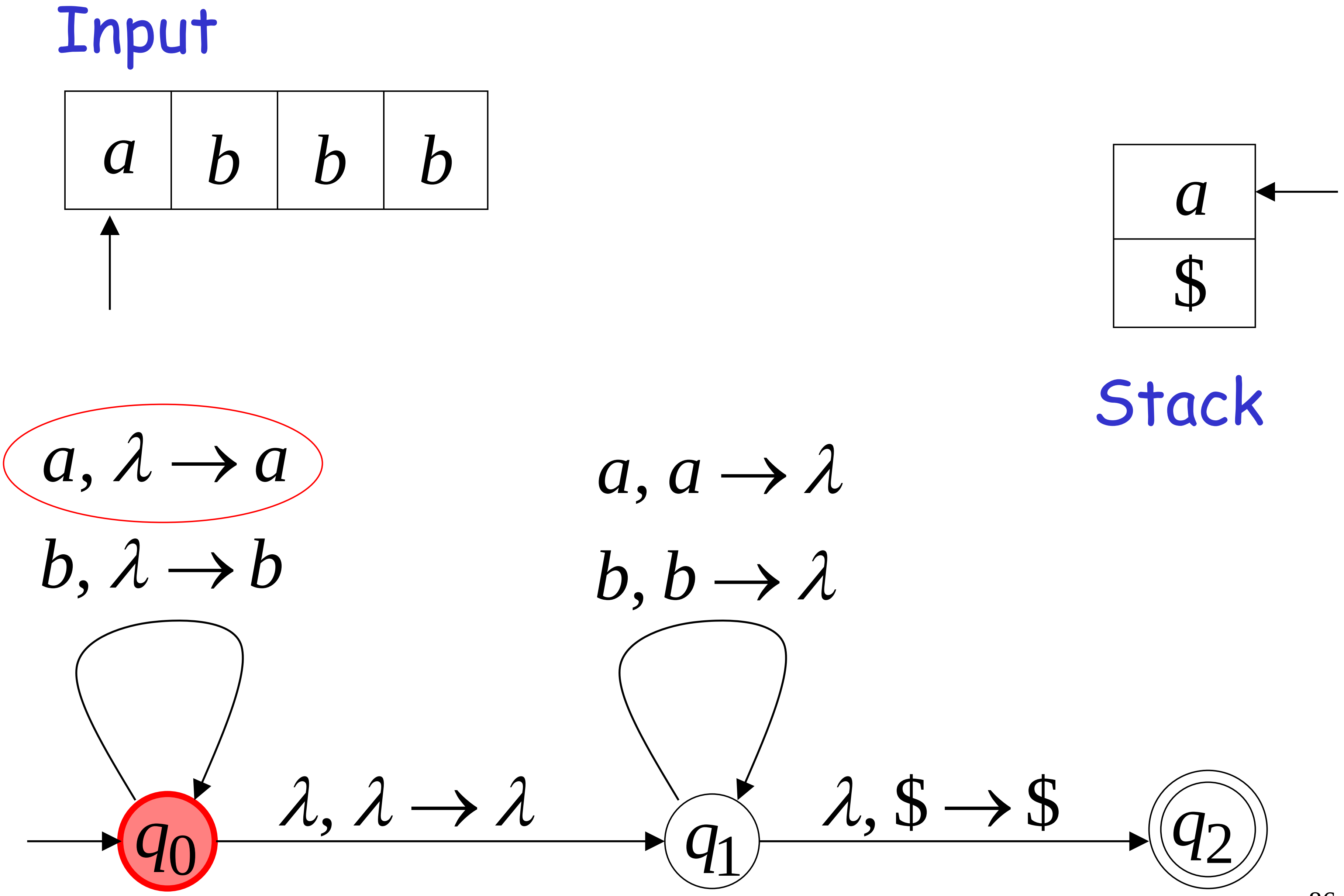
Another computation on same string:

Input

Time 0

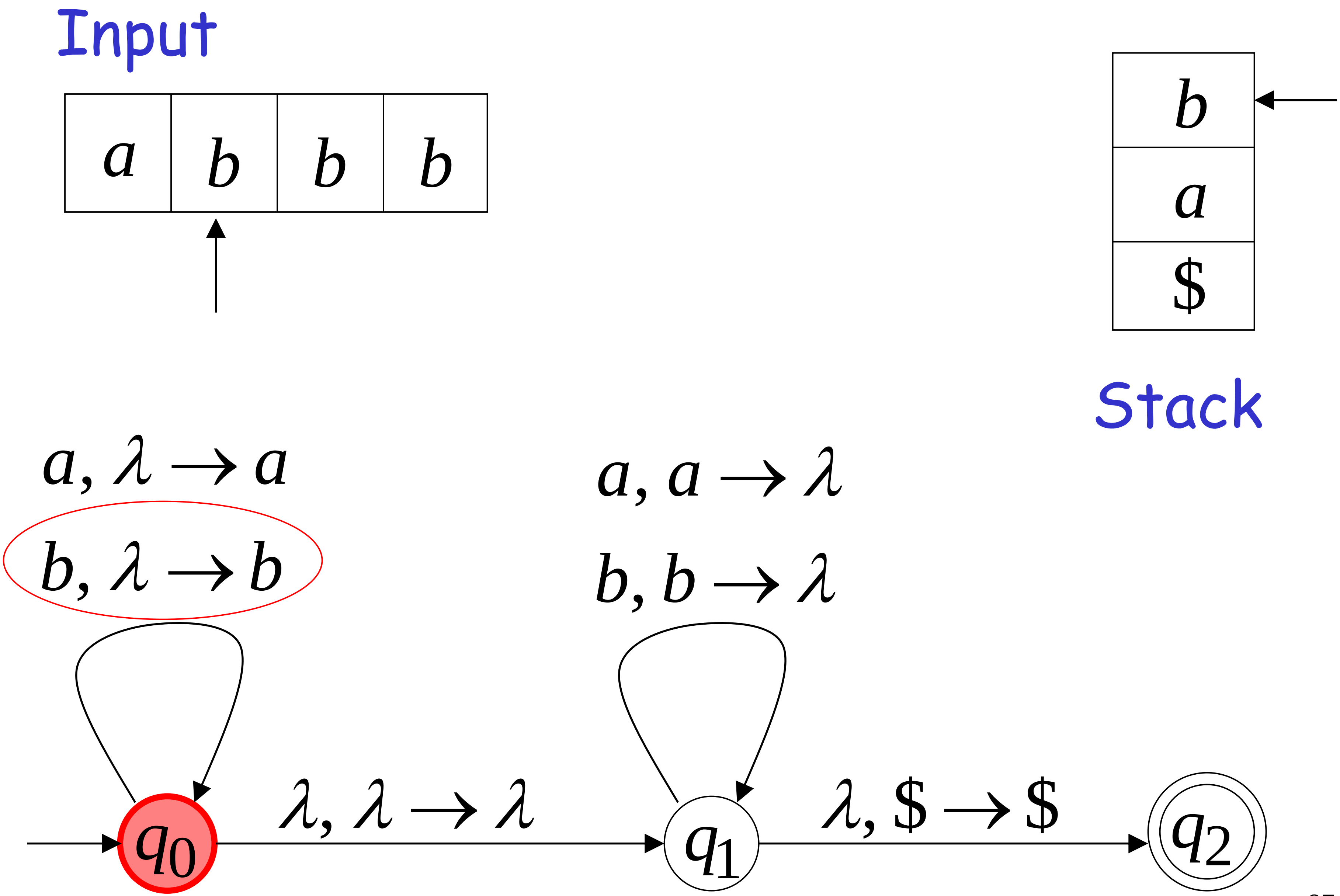


Time 1



Input
a b b b
a
$
Stack
$a, \lambda \rightarrow a$
$b, \lambda \rightarrow b$
$a, a \rightarrow \lambda$
$b, b \rightarrow \lambda$
$\lambda, \lambda \rightarrow \lambda$
$\lambda, \$ \rightarrow \$$
q_0
q_1
q_2

Time 2



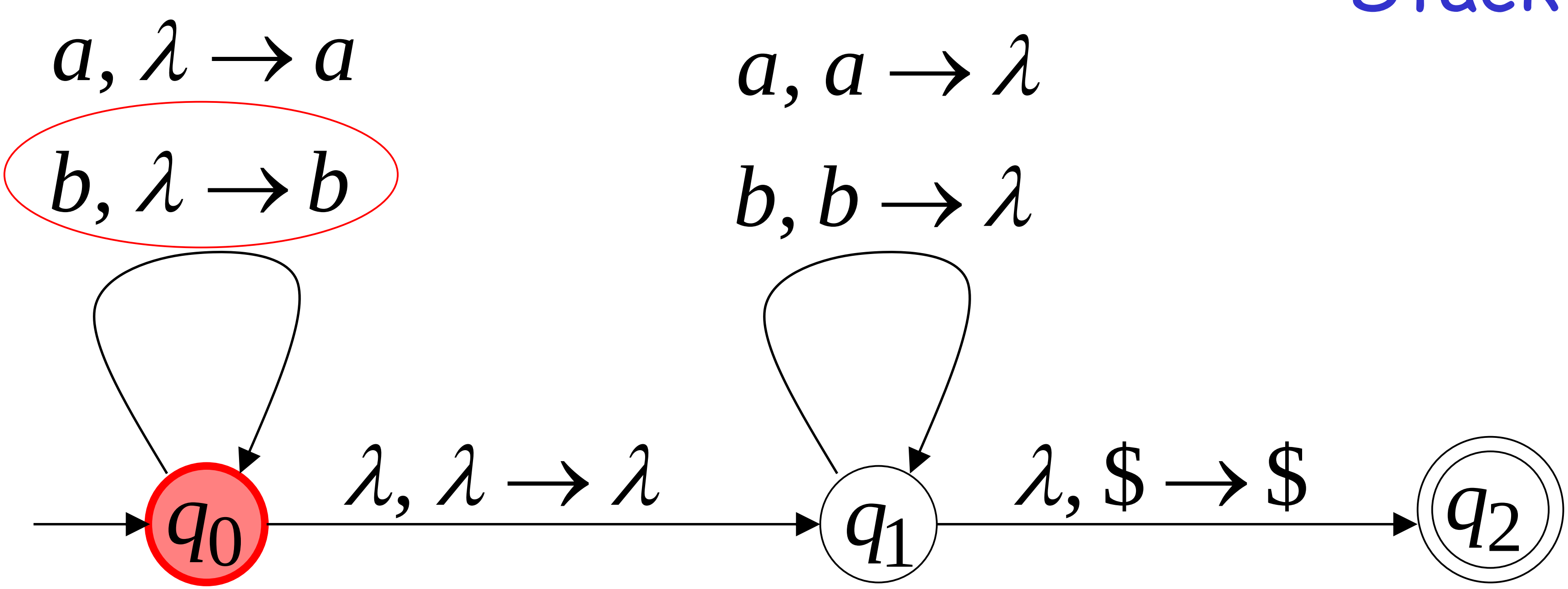
Time 3

Input

a	b	b	b

b
b
a
$\$$

Stack



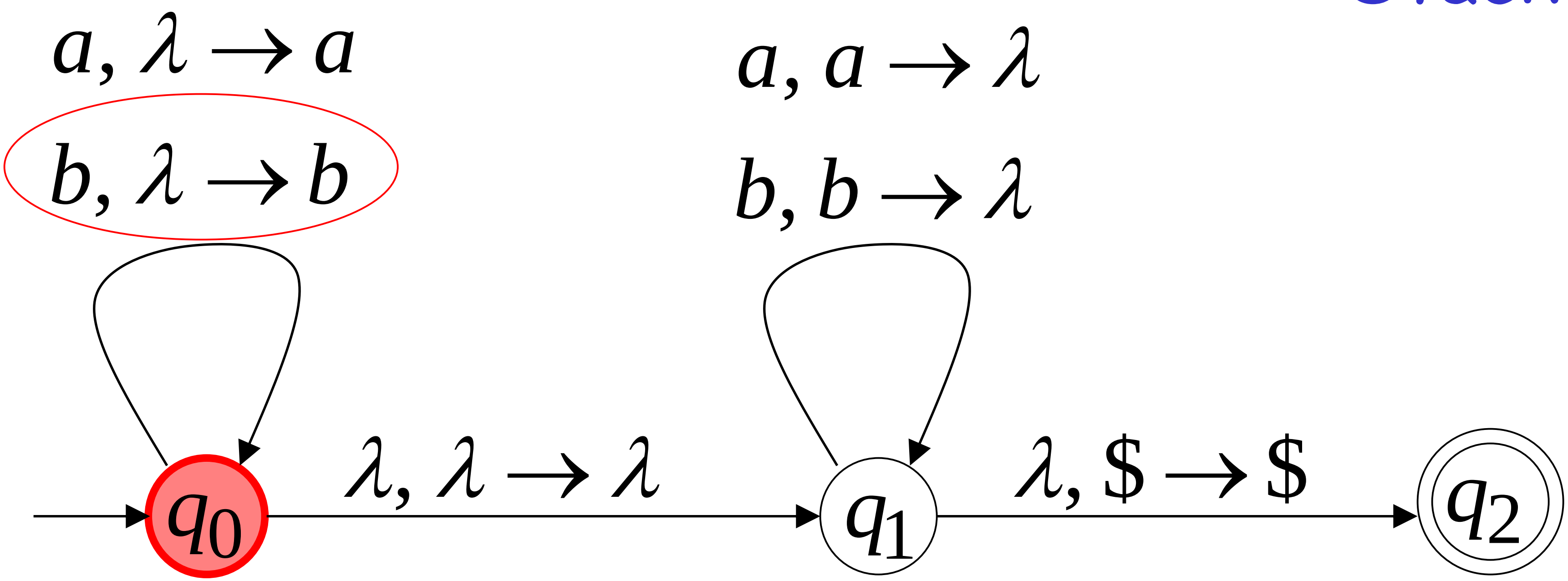
Time 4

Input

a	b	b	b

b
b
b
a
$\$$

Stack



Time 5

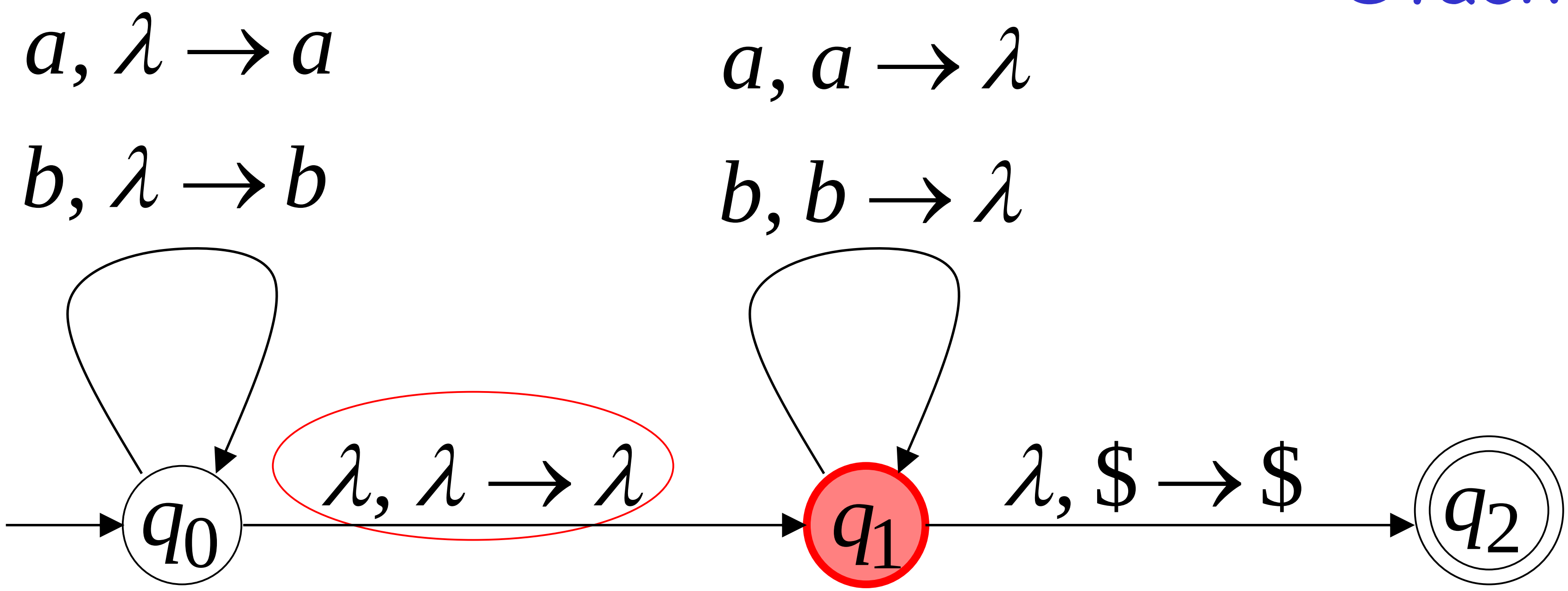
Input

a	b	b	b

No final state
is reached

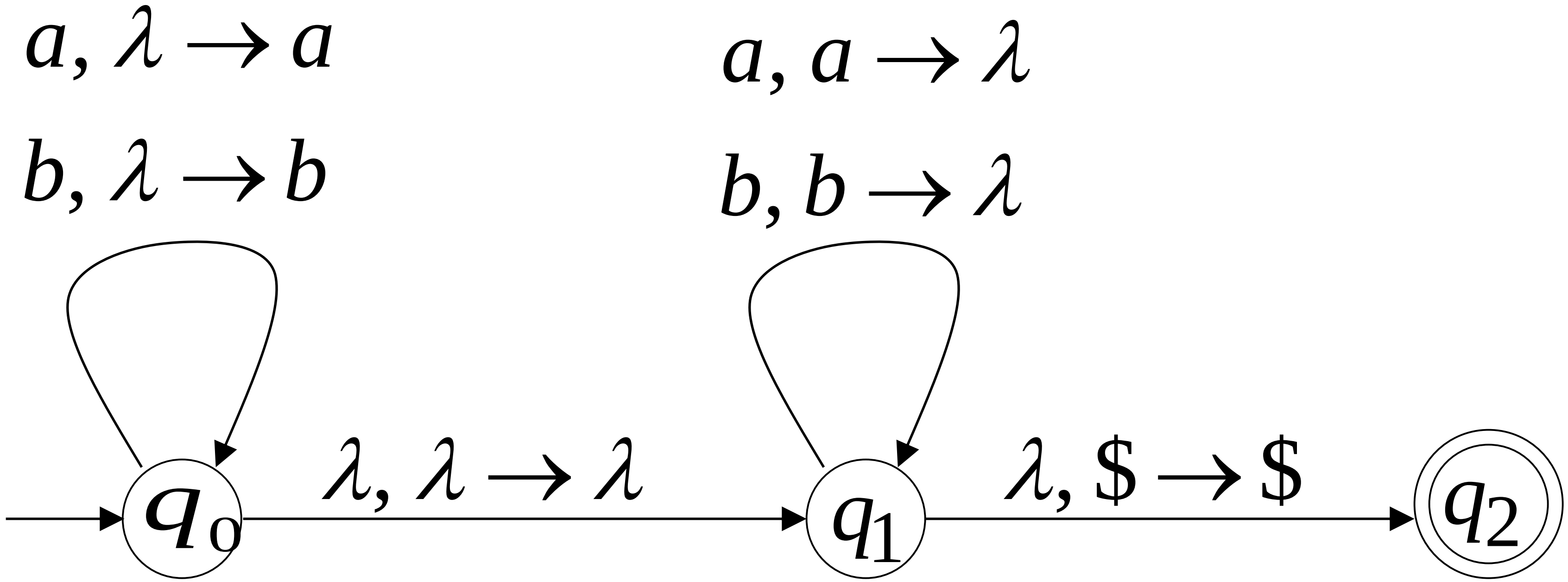
b
b
b
a
$\$$

Stack



There is no computation
that accepts string $abbb$

$$abbb \notin L(M)$$



DPDA example

DPDA M

Example of DCFL: $L(M) = \{wcw^R\}$

δ (q0,a,z)={(q1,az)},δ (q1,b,z)={(q1,bz)},

δ (q1,a,a)={(q1,aa)},δ (q1,b,b)={(q1,bb)}, δ(q1,a,b)={(q1,ab)},
δ(q1,b,a)={(q1,ba)},

δ (q1,c,a)={(q2,a)}, δ (q1,c,b)={(q2,b)},

δ (q2,a,a)={(q2,Λ)}, δ (q2,b,b)={(q2,Λ)}, δ (q2,Λ,z)={(qf,Λ)}

Example of NCFL

$$L(M) = \{a^n b^m c^k \mid \mathrm{n = m \; or \; m = k}\}$$

$$= \{a^n b^n c^k \mid \mathrm{n, k} \geq 0\} \cup \{a^n b^m c^m \mid \mathrm{n, m} \geq 0\}$$

Theorem:

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

PDAs And CLFs

For any context-free language L, there exists an NPDA M such that L=L(M)

Proof:

If L is a context-free language (without λ), there exists a context-free grammar G that generates it.

We can always convert a context-free grammar into Greibach Normal Form.

We can always construct an NPDA which simulates leftmost derivations in the GNF grammar.

Greinbach Normal Form

All productions have form:

$$A \to a\, V_1 V_2 \cdots V_k \qquad k \geq 0$$

symbol

variables

The Procedure for convert to Greinbach normal form

First remove:

1. *λ-productions*
2. left recursive productions
3. Unit productions

Then

Convert to Greinbach normal form

CFG to PDA

To convert a context-free grammar to an equivalent PDA:

1. Convert the grammar to Greibach Normal Form (GNF).

2. Write a transition rule for the PDA that pushes S (the start symbol in the grammar) onto the stack.

$$\delta\,(q0,\lambda,z)=\{(q1,Sz)\}$$

3. For each production rule in the grammar, write an equivalent transition rule.

$$A \rightarrow a\; B_1 B_2 \cdots B_n \Rightarrow \delta(q_1, a, A) = \{(q_1, B_1 B_2 \cdots B_n)\}$$

CFG to PDA

4. Write a transition rule that takes the automaton to the accepting state when you run out of characters in the output string and the stack is empty.

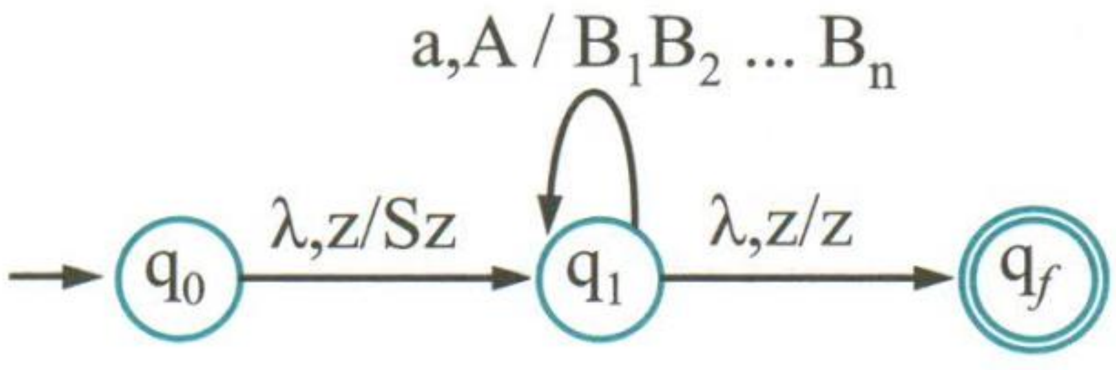
   $\delta\ (q1,\Lambda,z)=\{(qf,z)\}$

5. If the empty string is a legitimate string in the language described by the grammar, write a transition rule that takes the automaton to the accepting state directly from the start state.

   $\delta\ (q0,\Lambda,z)=\{(qf, z)\}$

CFG to PDA

Input: $G=(V,\Sigma, P,S)$
Output:
  PDA = $(\{q0,q1,qf\}, \Sigma, V\cup\{z\}, \delta, q0, z, \{qf\})$,
  accepting L(G)



CFG to PDA

Here is a grammar in GNF:
G=(V,T,S,P), where V={S,A,B,C}, T={a,b,c},S=S,
And P=

$$S \to aA$$

$$A \to aABC \mid bB \mid a$$

$$B \to b$$

$$C \to c$$

Let's convert this grammar to a PDA.

CFG to PDA

Grammer rule:	PDA transition rule:
$(none)$	$\delta(q_0, \lambda, z) = \{(q_1, Sz)\}$
$S \rightarrow aA$	$\delta(q_1, a, S) = \{(q_1, A)\}$
$A \rightarrow aABC$	$\delta(q_1, a, A) = \{(q_1, ABC)\}$
$A \rightarrow bB$	$\delta(q_1, b, A) = \{(q_1, B)\}$
$A \rightarrow a$	$\delta(q_1, a, A) = \{(q_1, \lambda)\}$
$B \rightarrow b$	$\delta(q_1, b, B) = \{(q_1, \lambda)\}$
$C \rightarrow c$	$\delta(q_1, c, C) = \{(q_1, \lambda)\}$
$(none)$	$\delta(q_1, \lambda, z) = \{(q_2, z)\}$

NPDAs

Have More Power than

DPDAs

It holds that:

$$\left\{\begin{array}{c}\text{Deterministic}\\ \text{Context-Free}\\ \text{Languages}\\ \text{(DPDA)}\end{array}\right\} \subseteq \left\{\begin{array}{c}\text{Context-Free}\\ \text{Languages}\\ \text{NPDAs}\end{array}\right\}$$

Since every DPDA is also a NPDA

We will actually show:

$$\left\{\begin{array}{c}\text{Deterministic} \\ \text{Context-Free} \\ \text{Languages} \\ \text{(DPDA)}\end{array}\right\} \subset \left\{\begin{array}{c}\text{Context-Free} \\ \text{Languages} \\ \text{(NPDA)}\end{array}\right\}$$

$L \notin$ $\qquad$ $L \in$

there exists a context-free language which is not accepted by any DPDA

For example: $L(M) = \{ww^R\}$

The language is:

$$L = \{a^n b^n\} \cup \{a^n b^{2n}\} \qquad n \geq 0$$

- L is context-free
- L is **not** deterministic context-free

Finite automaton & DPDA

- any language that can be accepted by a finite automaton can also be accepted by a deterministic pushdown automaton.

Venn-diagram for Chomsky classification of formal languages