Normal Forms
for
Context-free Grammars

Linz 6th, Section 6.2

“Two Important Normal Forms,“

pages 171--178

Chomsky Normal Form

All productions have form:

$A \to BC$ and $A \to a$

variable variable terminal

Examples:

$$S \rightarrow AS$$

$$S \rightarrow a$$

$$A \rightarrow SA$$

$$A \rightarrow b$$

Chomsky
Normal Form

$$S \rightarrow AS$$

$$S \rightarrow \boxed{AAS}$$

$$A \rightarrow SA$$

$$A \rightarrow \boxed{aa}$$

Not Chomsky
Normal Form

Conversion to Chomsky Normal Form

Example:

$$S \to ABa$$

$$A \to aab$$

$$B \to Ac$$

Not Chomsky
Normal Form

Introduce variables for terminals: T_a, T_b, T_c

$$S \to ABa$$

$$A \to aab$$

$$B \to Ac$$

$$S \to ABT_a$$

$$A \to T_a T_a T_b$$

$$B \to AT_c$$

$$T_a \to a$$

$$T_b \to b$$

$$T_c \to c$$

Introduce intermediate variable: V_1

$$S \to ABT_a$$

$$A \to T_aT_aT_b$$

$$B \to AT_c$$

$$T_a \to a$$

$$T_b \to b$$

$$T_c \to c$$

⇒

$$S \to AV_1$$

$$V_1 \to BT_a$$

$$A \to T_aT_aT_b$$

$$B \to AT_c$$

$$T_a \to a$$

$$T_b \to b$$

$$T_c \to c$$

Introduce intermediate variable: V_2

$$S \to AV_1$$
$$V_1 \to BT_a$$
$$A \to T_aT_aT_b$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

$$S \to AV_1$$
$$V_1 \to BT_a$$
$$A \to T_aV_2$$
$$V_2 \to T_aT_b$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

Final grammar in Chomsky Normal Form:

$$S \to AV_1$$

$$V_1 \to BT_a$$

$$A \to T_a V_2$$

$$V_2 \to T_a T_b$$

$$B \to AT_c$$

$$T_a \to a$$

$$T_b \to b$$

$$T_c \to c$$

Initial grammar

$$S \to ABa$$

$$A \to aab$$

$$B \to Ac$$

In general:

From any context-free grammar
not in Chomsky Normal Form

we can obtain:

An equivalent grammar
in Chomsky Normal Form

The Procedure

First remove:

Nullable variables

Unit productions

For every symbol a :

Add production $T_a \rightarrow a$

In productions: replace a with T_a

New variable: T_a

Replace any production $A \to C_1 C_2 \cdots C_n$

with
$$
\begin{aligned}
&A \to C_1 V_1 \\
&V_1 \to C_2 V_2 \\
&\cdots \\
&V_{n-2} \to C_{n-1} C_n
\end{aligned}
$$

New intermediate variables: $V_1, V_2, \ldots, V_{n-2}$

Theorem: For any context-free grammar G such that the empty string is not in L(G), there is an equivalent grammar in Chomsky Normal Form

Linz, 6th, Theorem 6.6, page 172.

Observations

- Chomsky normal forms are good for parsing and proving theorems

- It is very easy to find the Chomsky normal form of any context-free grammar

Greibach Normal Form

All productions have form:

$$A \to \underset{\substack{\uparrow \\ \text{symbol}}}{a}\, \underset{\substack{\uparrow \\ \text{variables}}}{V_1 V_2 \cdots V_k} \qquad k \geq 0$$

Examples:

$$S \to cAB$$

$$A \to aA \mid bB \mid b$$

$$B \to b$$

Greibach
Normal Form

$$S \to abSb$$

$$S \to aa$$

Not Greibach
Normal Form

Conversion to Greibach Normal Form:

$$S \to abSb$$

$$S \to aa$$

⇒

$$S \to aT_bST_b$$

$$S \to aT_a$$

$$T_a \to a$$

$$T_b \to b$$

Greibach
Normal Form

Theorem: For any context-free grammar G such that the empty string is not in L(G), there is an equivalent grammar in Greibach Normal Form

Linz, 6th, Theorem 6.7, page 176.
Proof not given because it is too complicated.

Observations

- Greibach normal forms are very good for parsing

- It is hard to find the Greibach normal form of any context-free grammar

An Application
of
Chomsky Normal Forms

The CYK Membership Algorithm

J. Cocke, D. H. Younger, and T. Kasami

Input:

- Grammar G in Chomsky Normal Form
- String w

Output:

find if $w \in L(G)$

Considers every possible consecutive subsequence of letters and sets K∈T[i,j] if the sequence of letters starting from i to j can be generated from the non-terminal K.

Once it has considered sequences of length

1, it goes on to sequences of length 2, and so on.

For subsequences of length 2 and greater, it considers every possible partition of the subsequence into two halves, and checks to see if there is some production A-> BC such that B matches the first half and C Matches the second half. If so, it records A as matching the whole subsequence.

Once this process is completed, the sentence is recognized by the grammar if the entire string is matched by the start symbol.

The Algorithm

Input example:

- Grammar G:
$$S \rightarrow AB$$
$$A \rightarrow BB$$
$$A \rightarrow a$$
$$B \rightarrow AB$$
$$B \rightarrow b$$

- String w: $aabbb$

aabbb

[0:1] [1:2] [2:3] [3:4] [4:5]

[0:2] [1:3] [2:4] [3:5]

[0:3] [1:4] [2:5]

[0:4] [1:5]

[0:5]

aabbb

a	a	b	b	b
aa	ab	bb	bb	
aab	abb	bbb		
aabb	abbb			
aabbb				

$S \to AB$

$A \to BB$

$A \to a$

$B \to AB$

$B \to b$

a	a	b	b	b
A	A	B	B	B
aa	ab	bb	bb	
aab	abb	bbb		
aabb	abbb			
aabbb				

$$S \to AB$$

$$A \to BB$$

$$A \to a$$

$$B \to AB$$

$$B \to b$$

a	a	b	b	b
A	A	B	B	B
aa	ab	bb	bb	
	S,B	A	A	
aab	abb	bbb		
aabb	abbb			
aabbb				

$$S \to AB$$

$$A \to BB$$

$$A \to a$$

$$B \to AB$$

$$B \to b$$

a	a	b	b	b
A	A	B	B	B
aa	ab	bb	bb	
	S,B	A	A	
aab	abb	bbb		
S,B	A	S,B		
aabb	abbb			
A	S,B			
aabbb				
S,B				

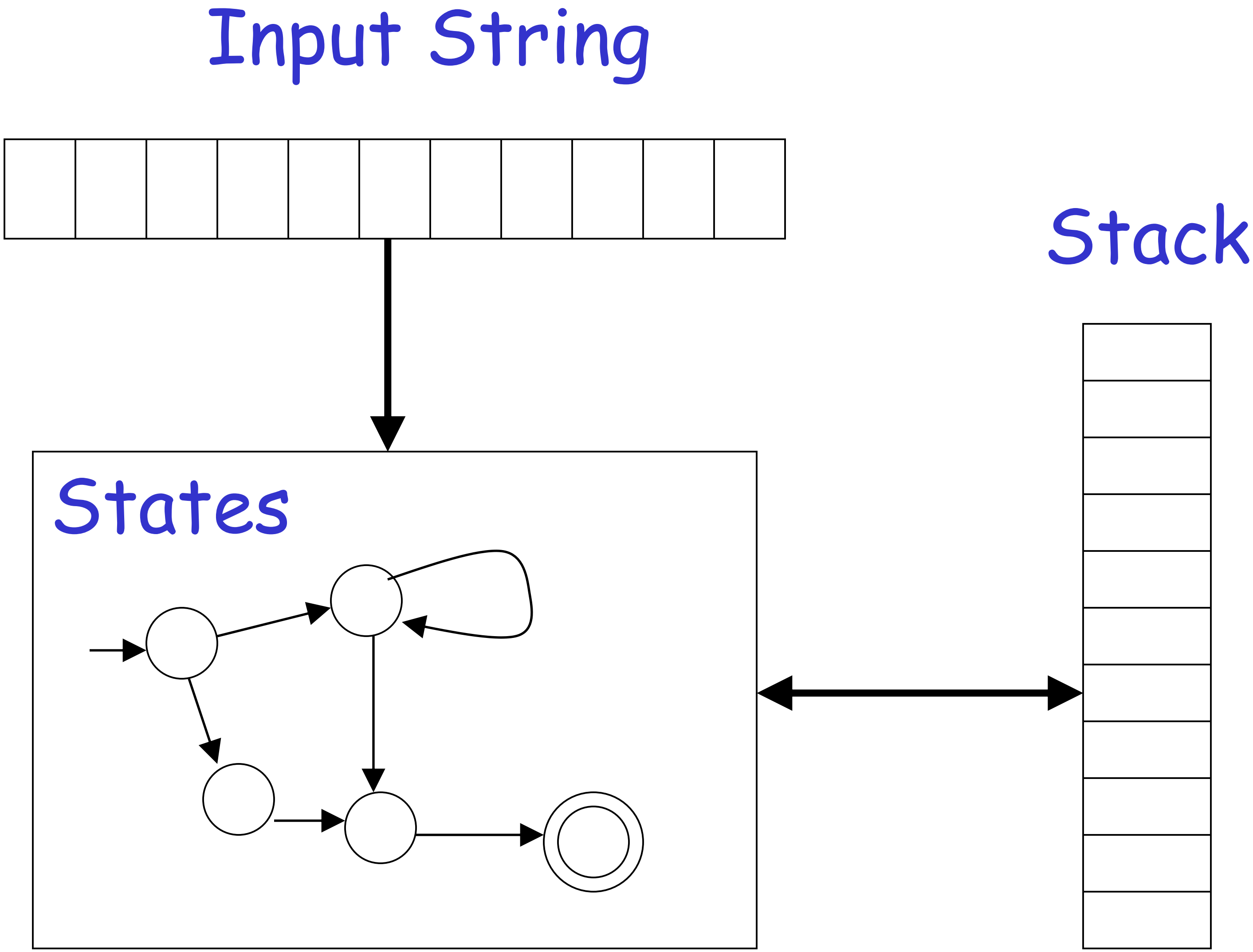
Therefore: $aabbb \in L(G)$

Time Complexity: $|w|^3$

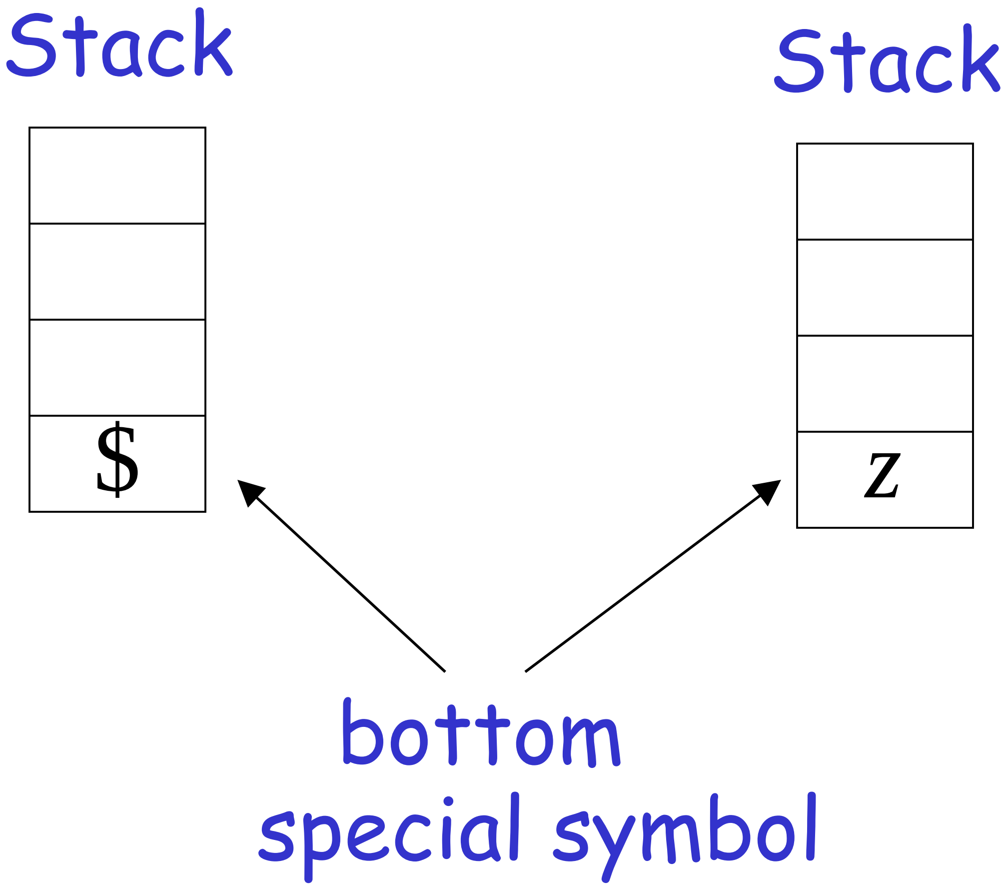
Observation: The CYK algorithm can be easily converted to a parser

Pushdown Automata
PDAs

Pushdown Automaton -- PDA



Initial Stack Symbol



Stack

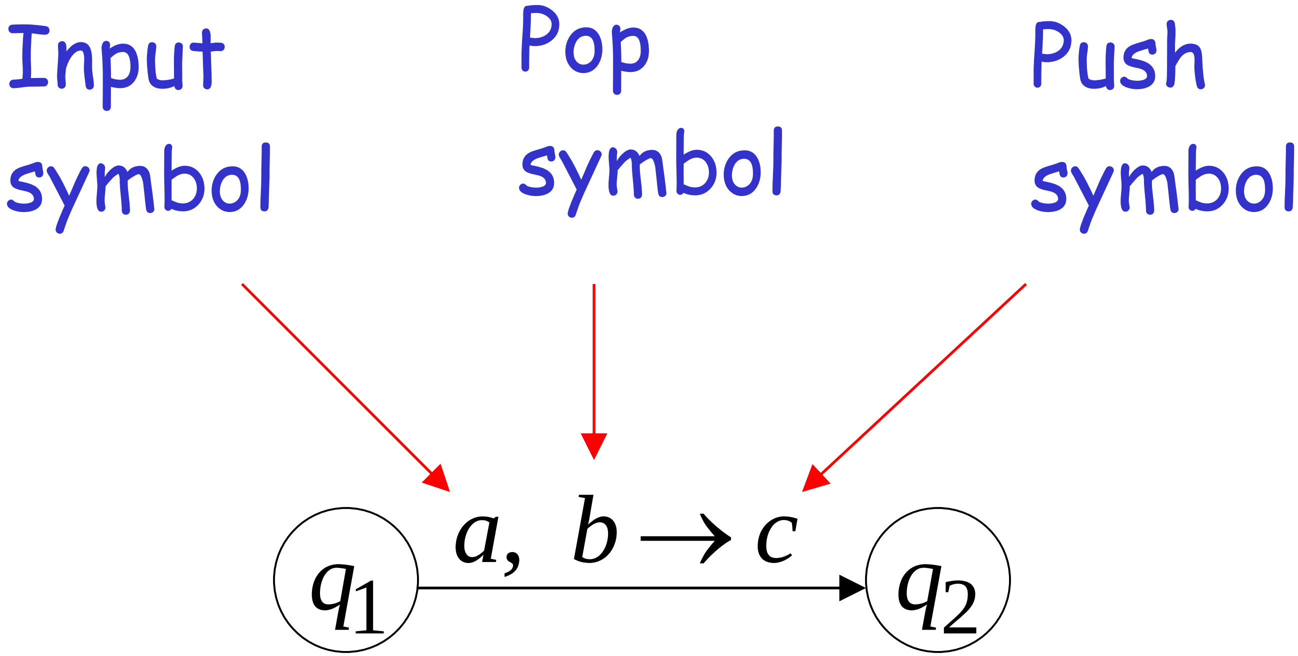
$\$$

Stack

z

bottom
special symbol

The States

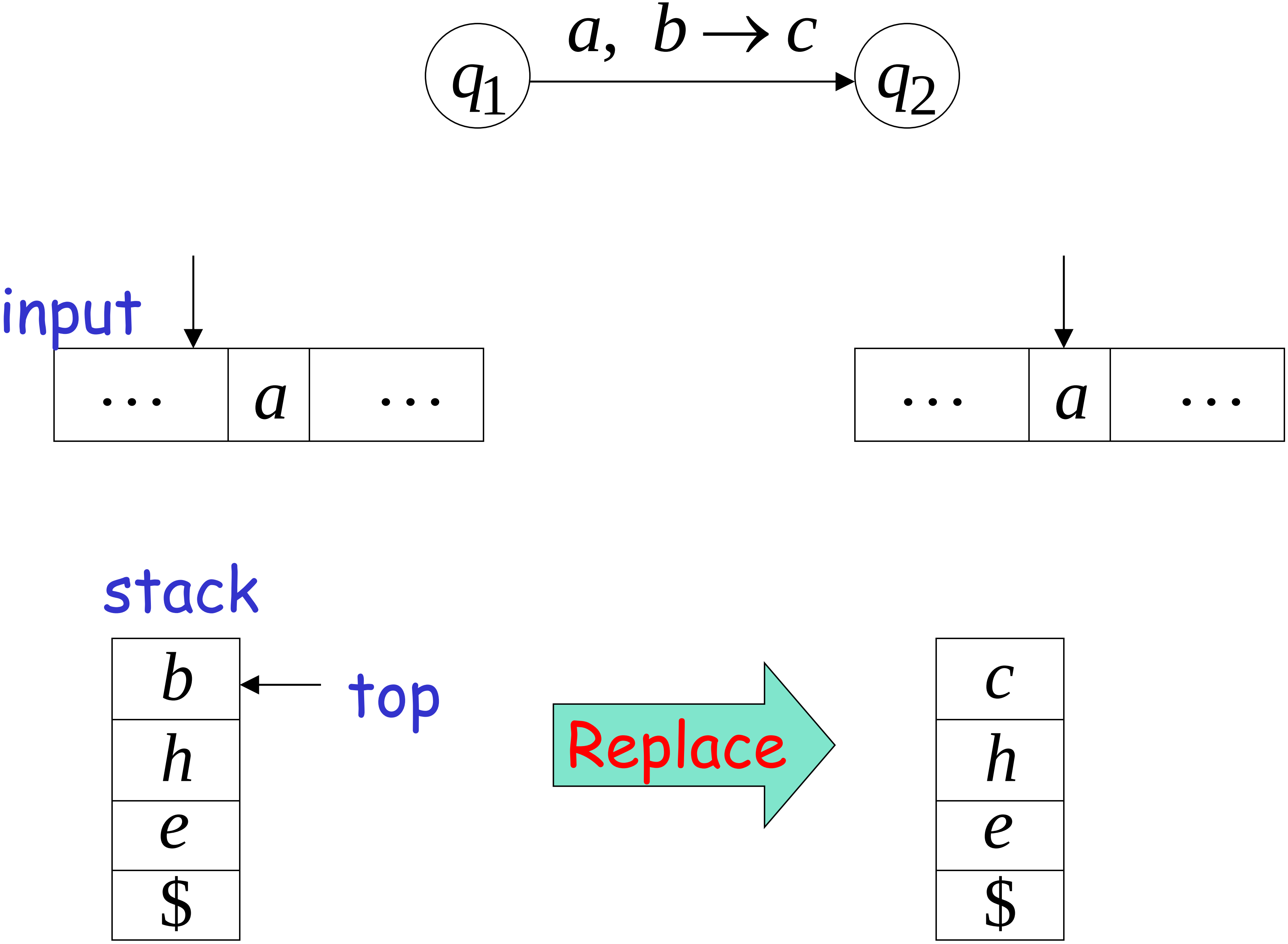


Input symbol

Pop symbol

Push symbol

$$a, \ b \to c$$



q_1 $a,\ b \to c$ q_2

input

$\ldots$ a $\ldots$

$\ldots$ a $\ldots$

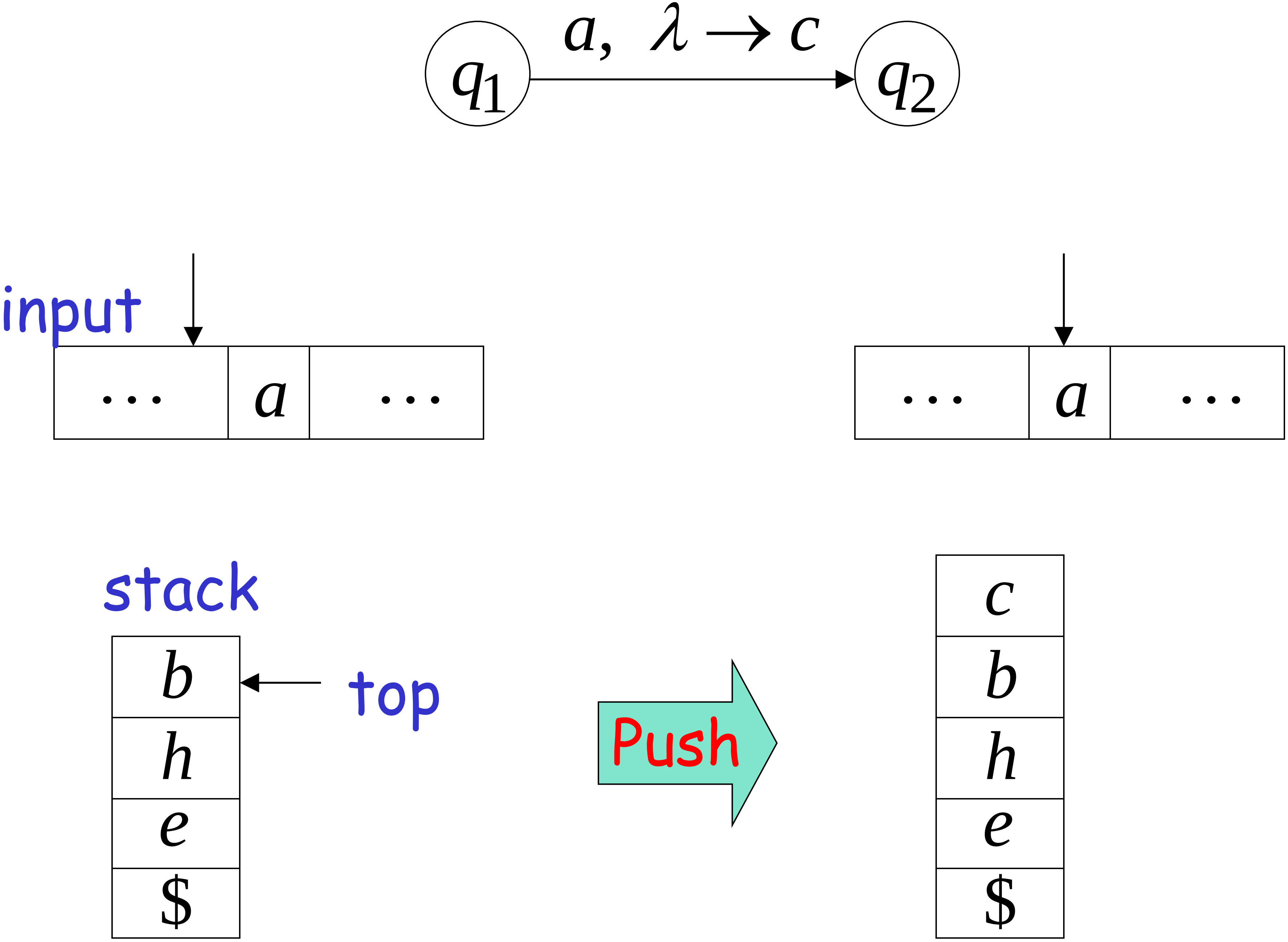
stack

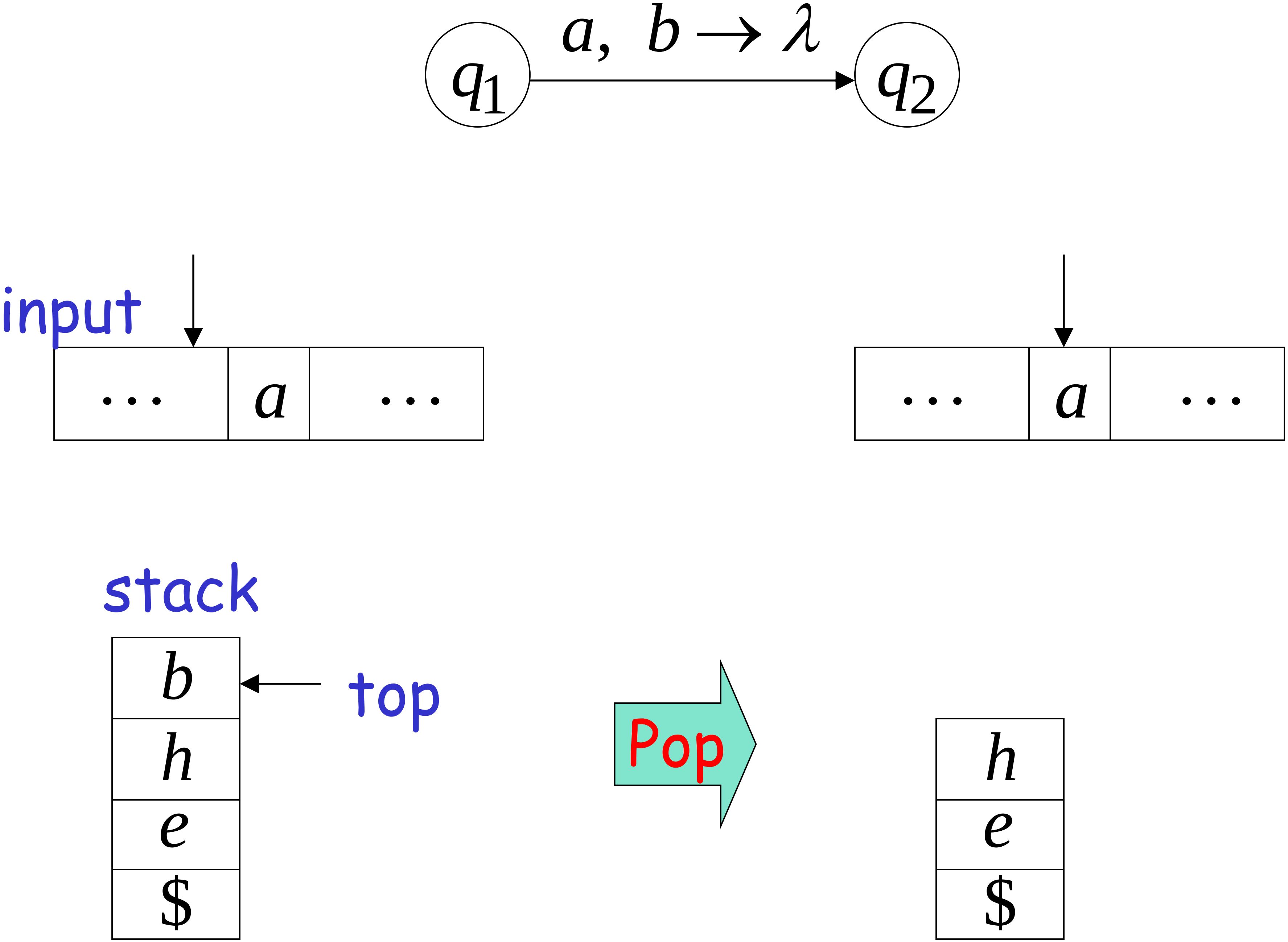
top

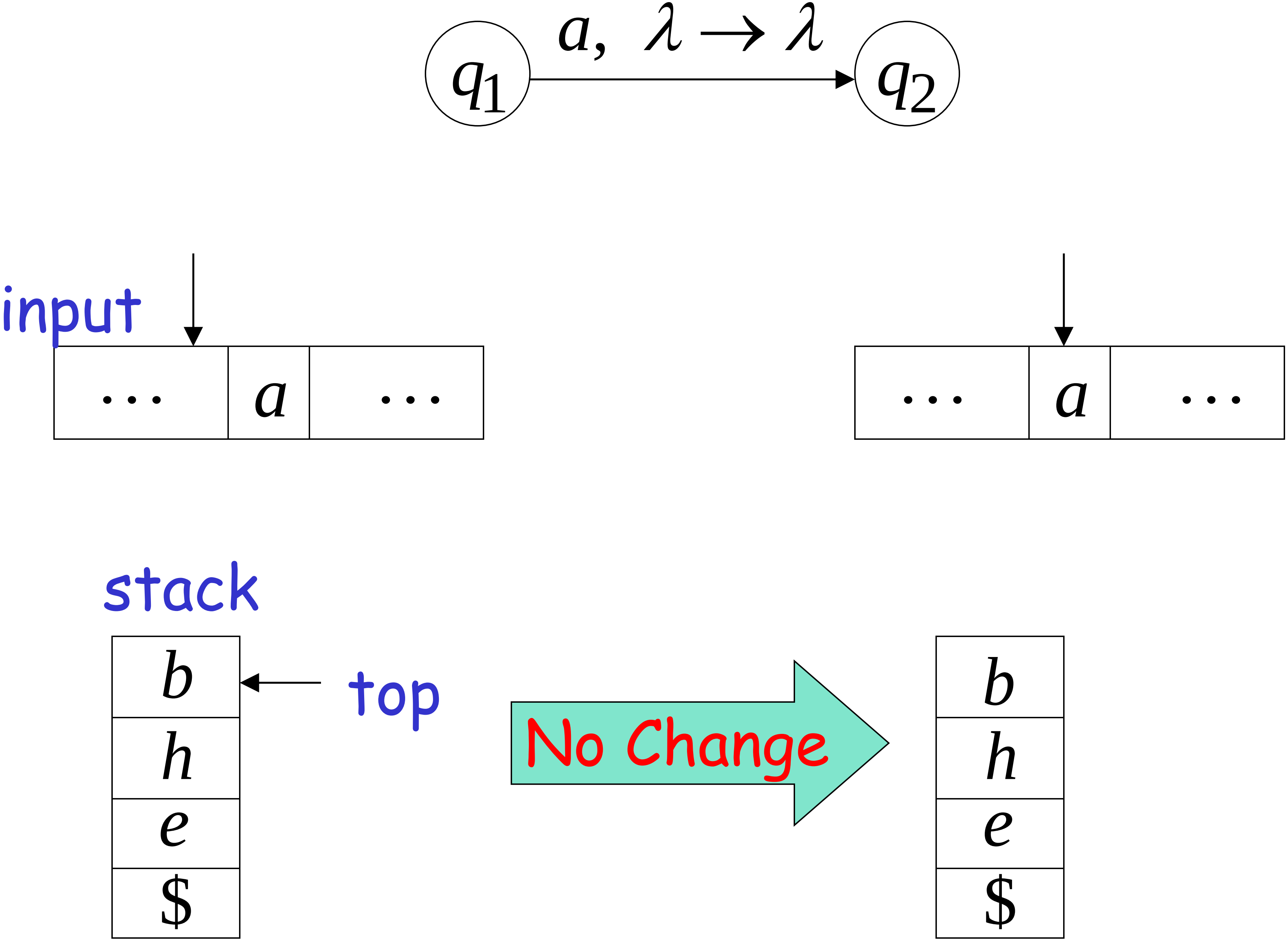
Replace

b
h
e
$\$$

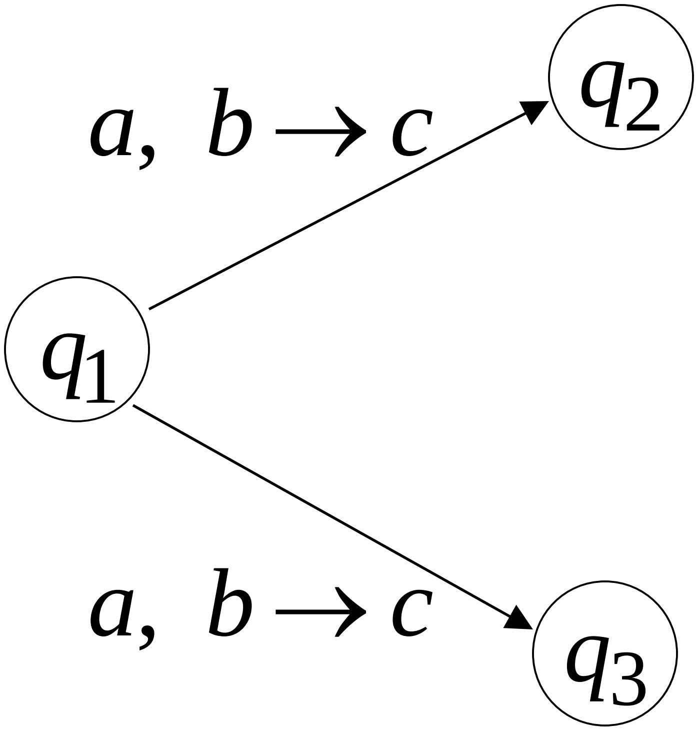
c
h
e
$\$$

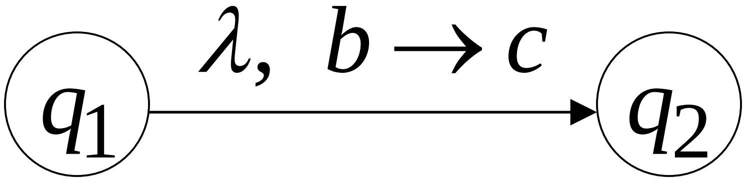

q_1
$a, \lambda \to c$
q_2
input
$\cdots$
a
$\cdots$
$\cdots$
a
$\cdots$
stack
b
h
e
$\$$
top
Push
c
b
h
e
$\$$


q_1
$a,\ b \rightarrow \lambda$
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
$\$$

Non-Determinism



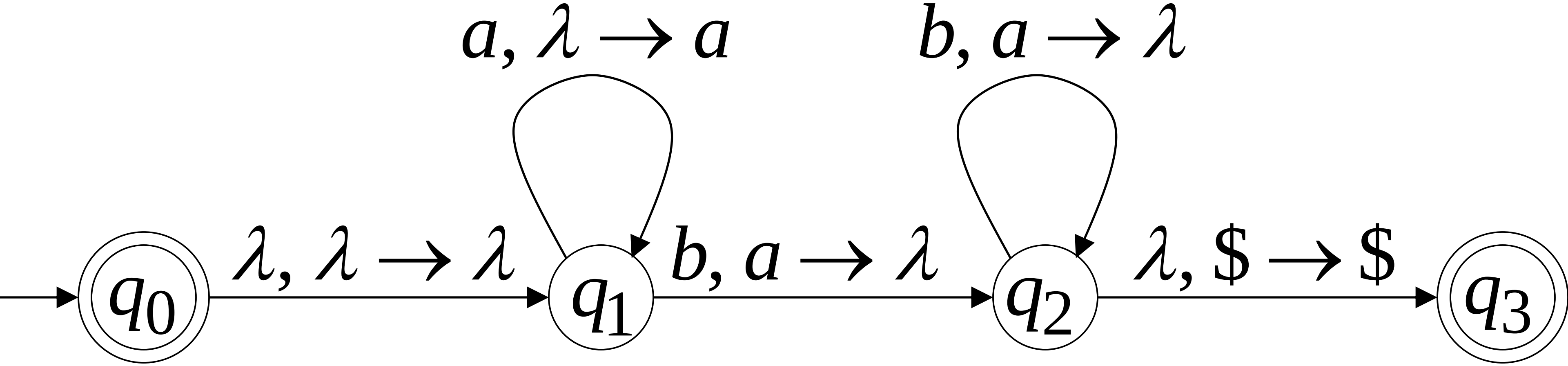


$\lambda -$ transition

These are allowed transitions in a Non-deterministic PDA (NPDA)

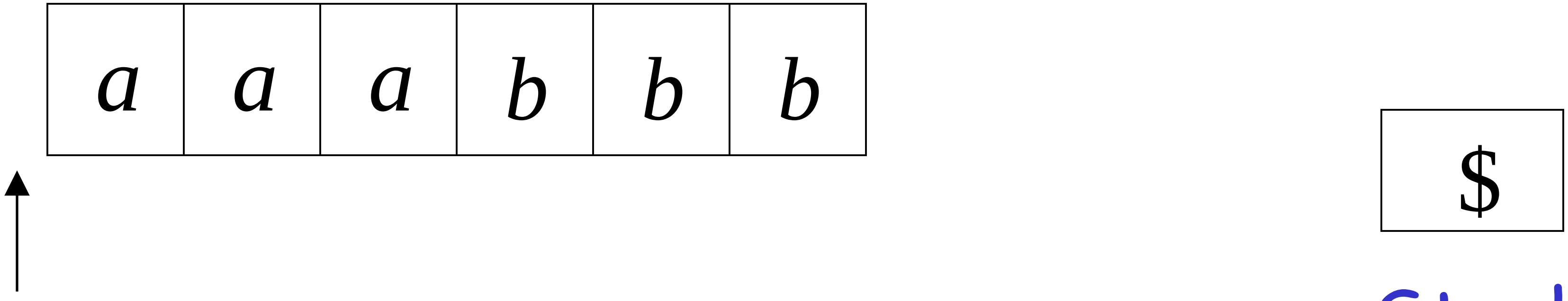
NPDA: Non-Deterministic PDA

Example:



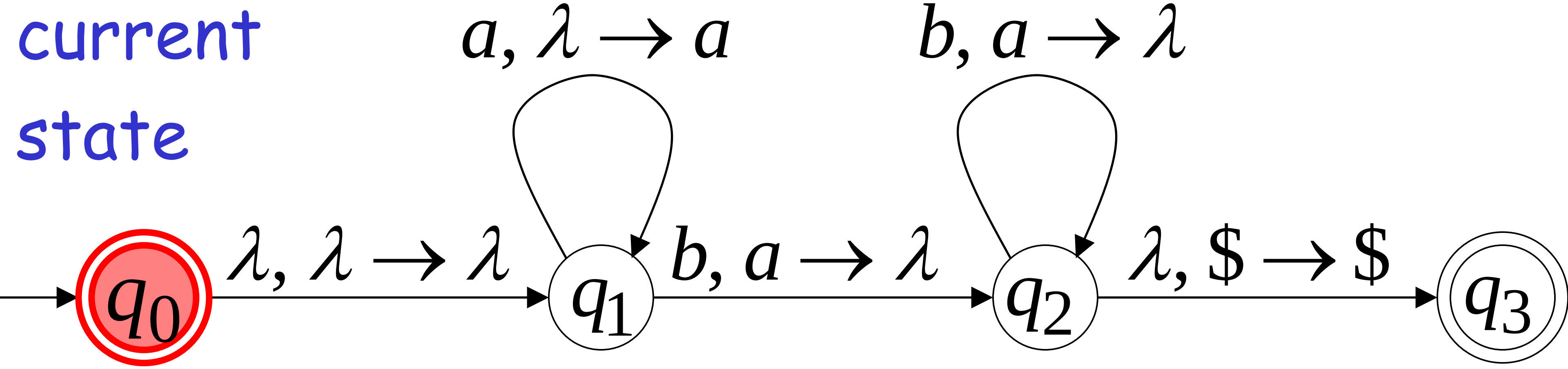
Execution Example: Time 0

Input

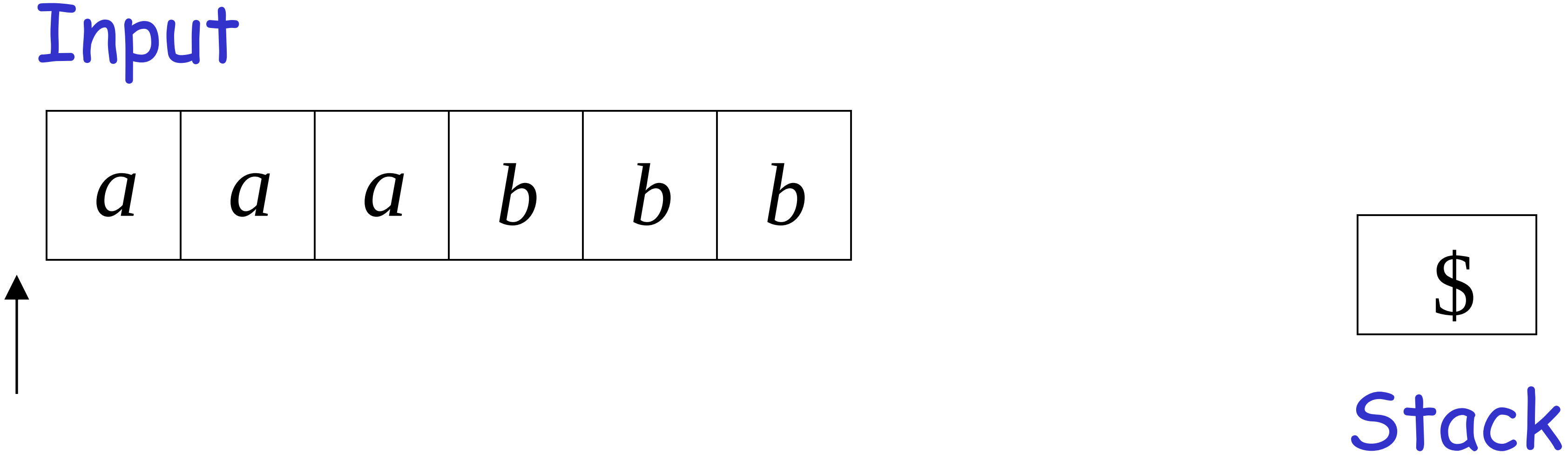


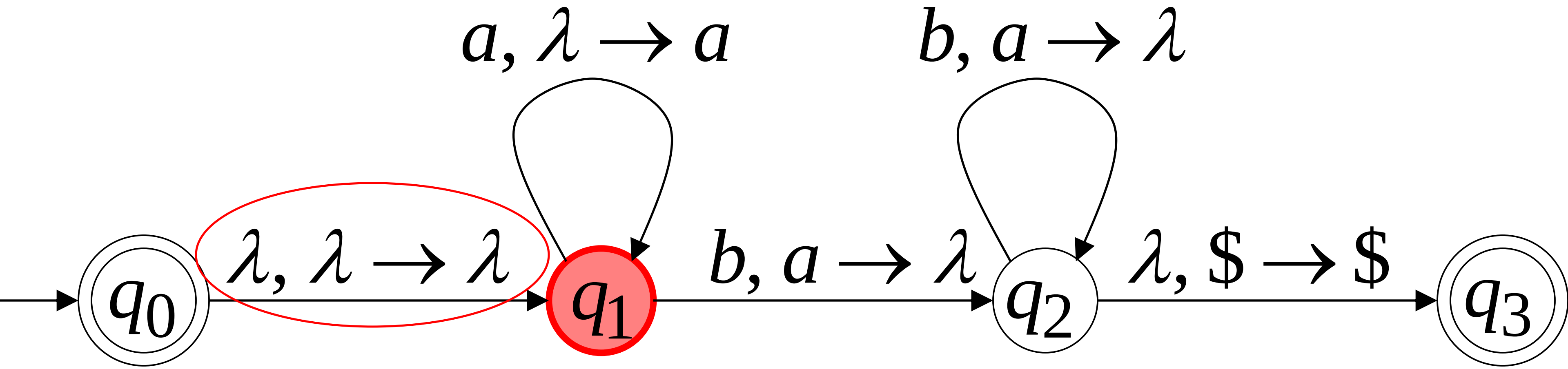
Stack

current state

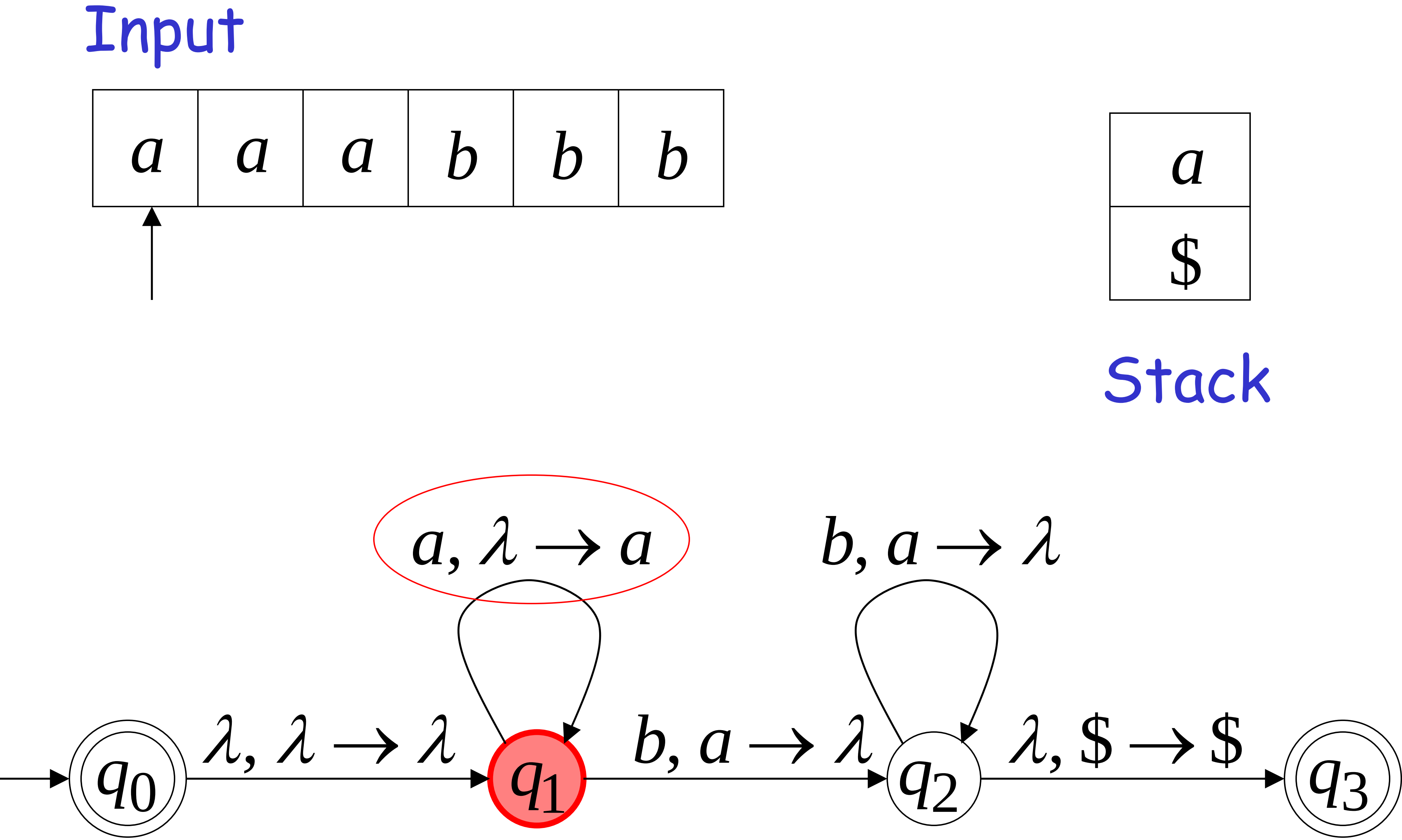


Time 1


Input
a
a
a
b
b
b
$\$$
Stack

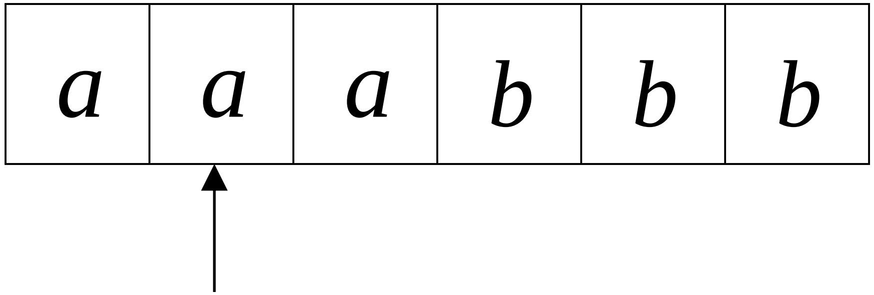

$a, \lambda \rightarrow a$
$b, a \rightarrow \lambda$
q_0
$\lambda, \lambda \rightarrow \lambda$
q_1
$b, a \rightarrow \lambda$
q_2
$\lambda, \$ \rightarrow \$$
q_3

Time 2


Input
a
a
a
b
b
b
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 3

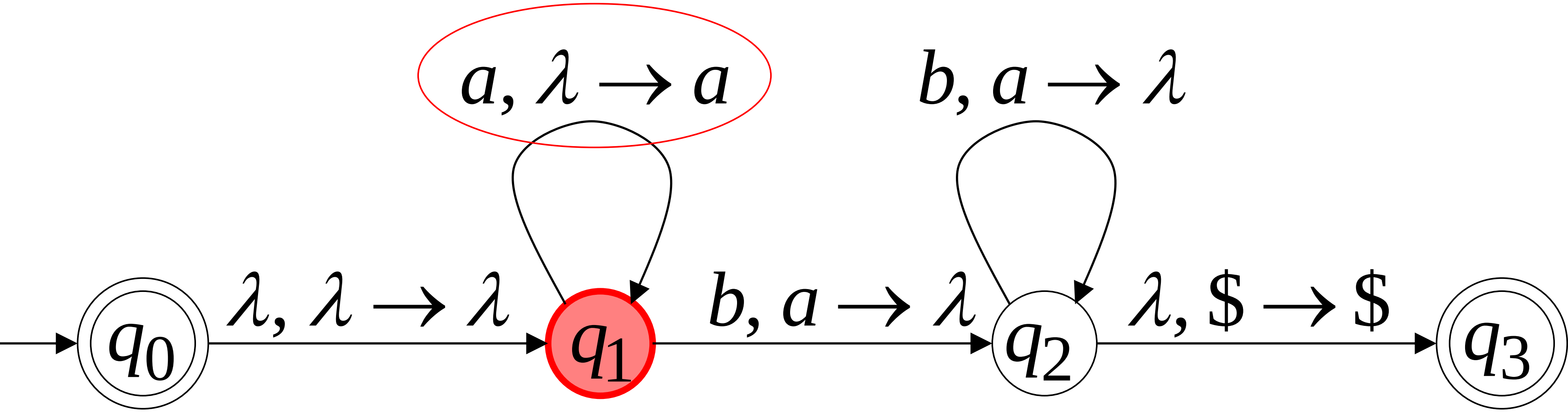
Input



a	a	a	b	b	b

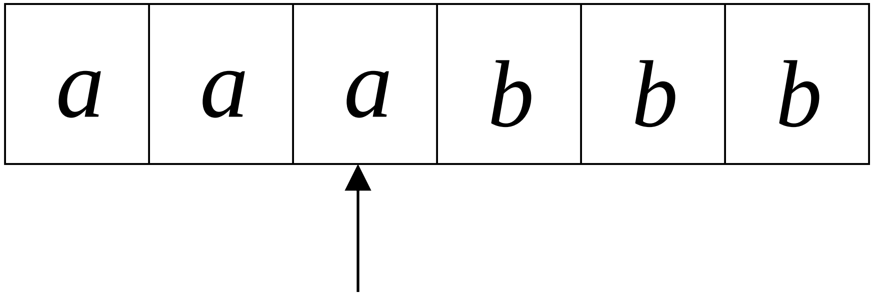
a
a
$\$$

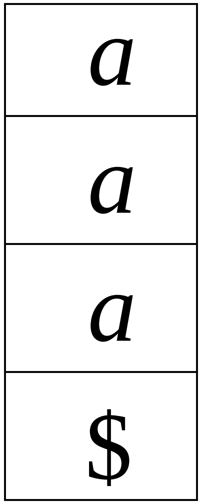
Stack



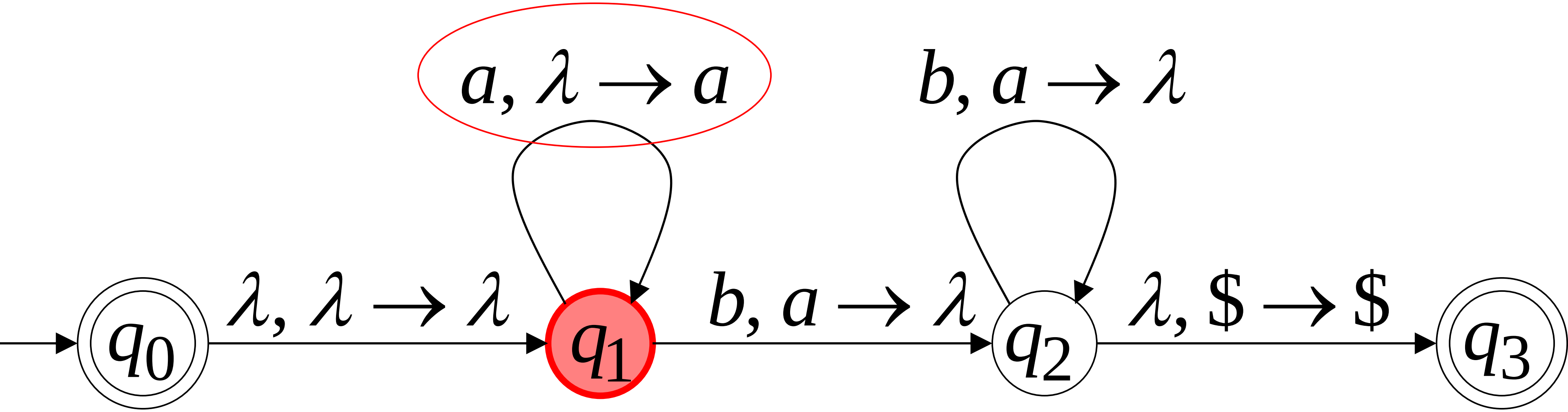
Time 4

Input

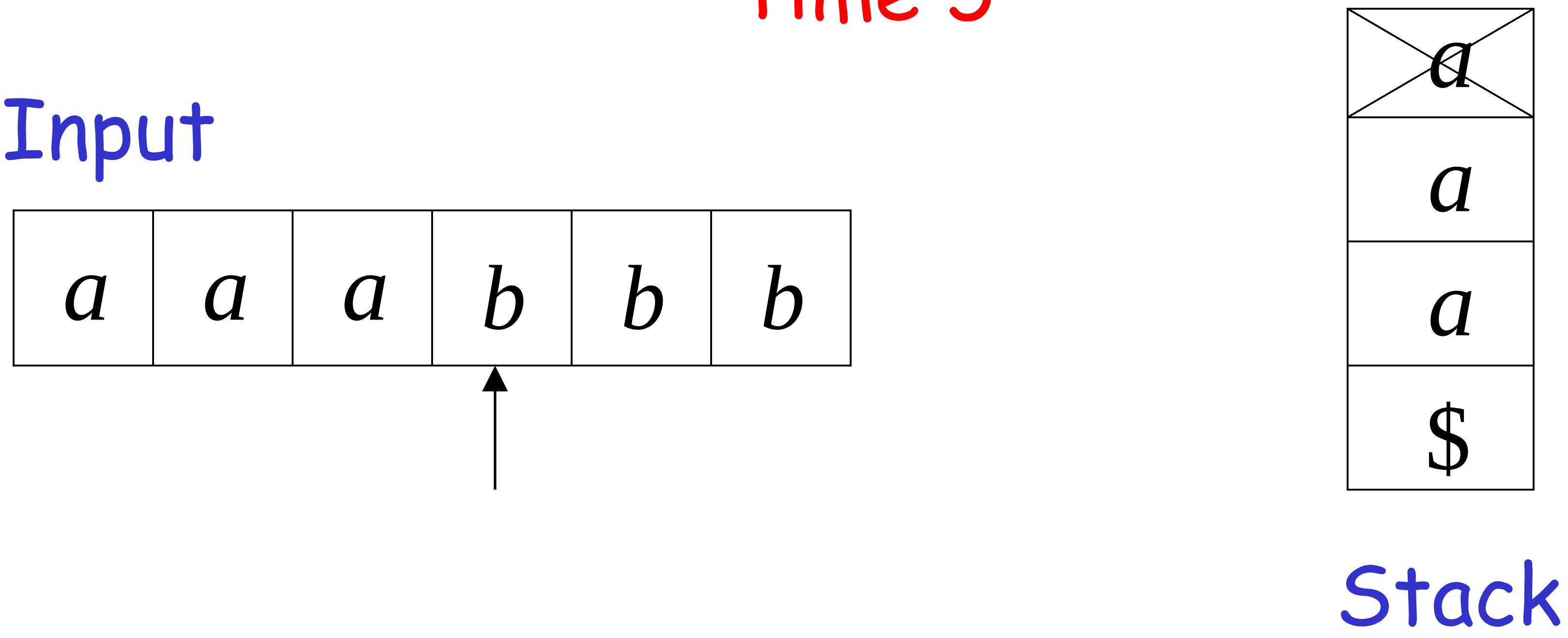


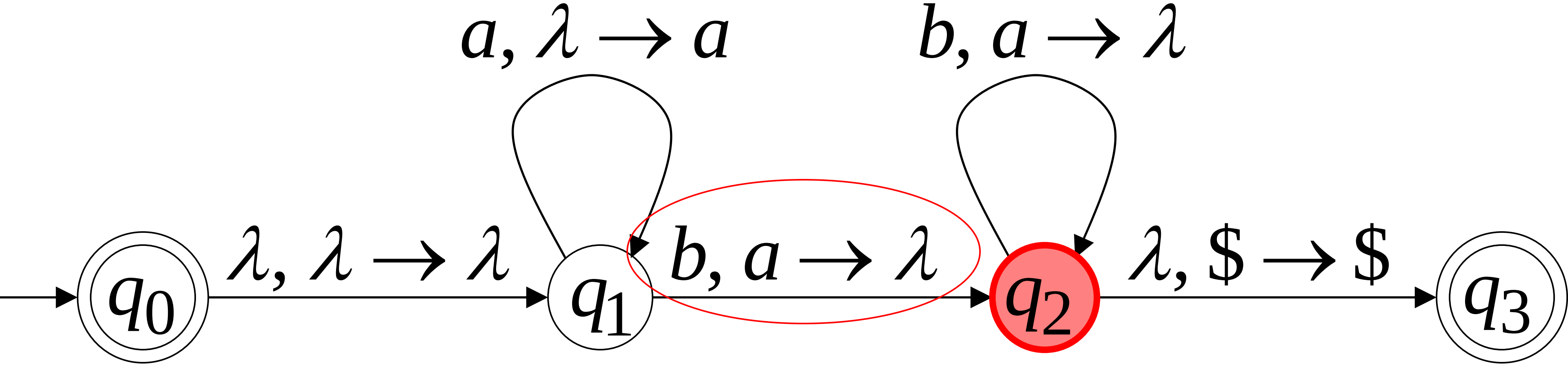


Stack



Time 5

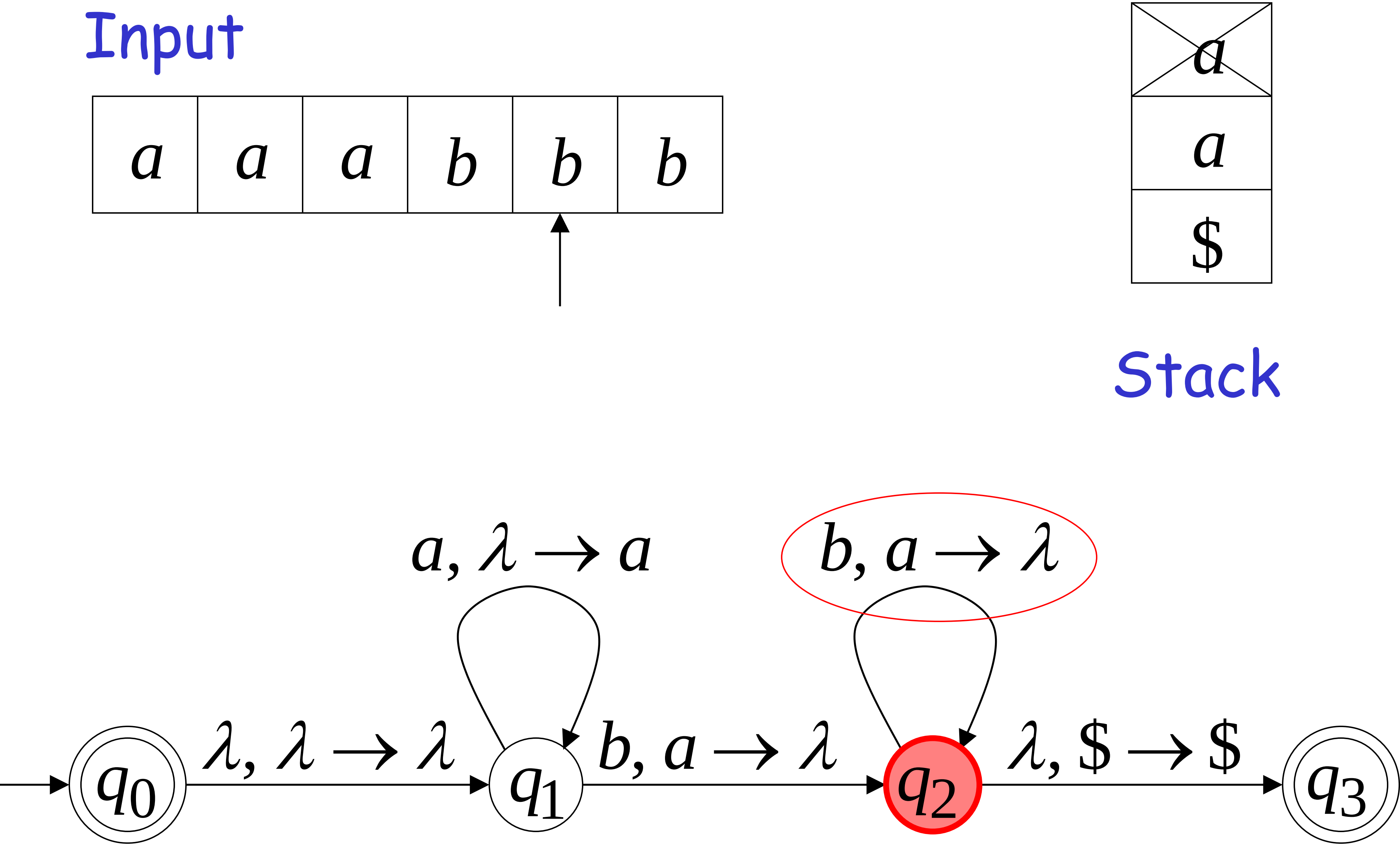

Input
a
a
a
b
b
b
~~a~~
a
a
$
Stack


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

Time 6

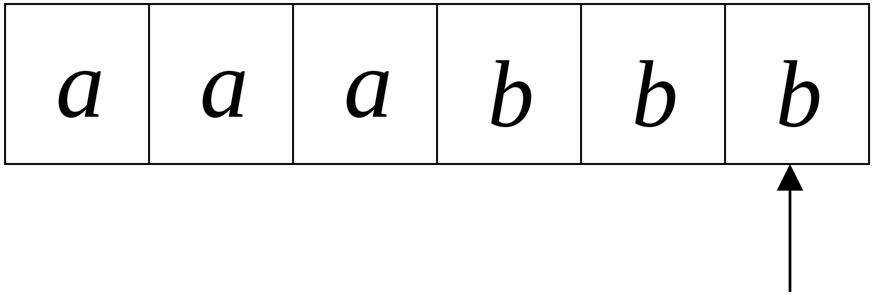
Input

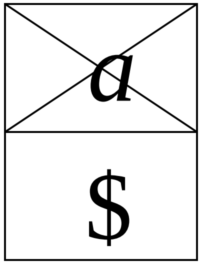
Stack



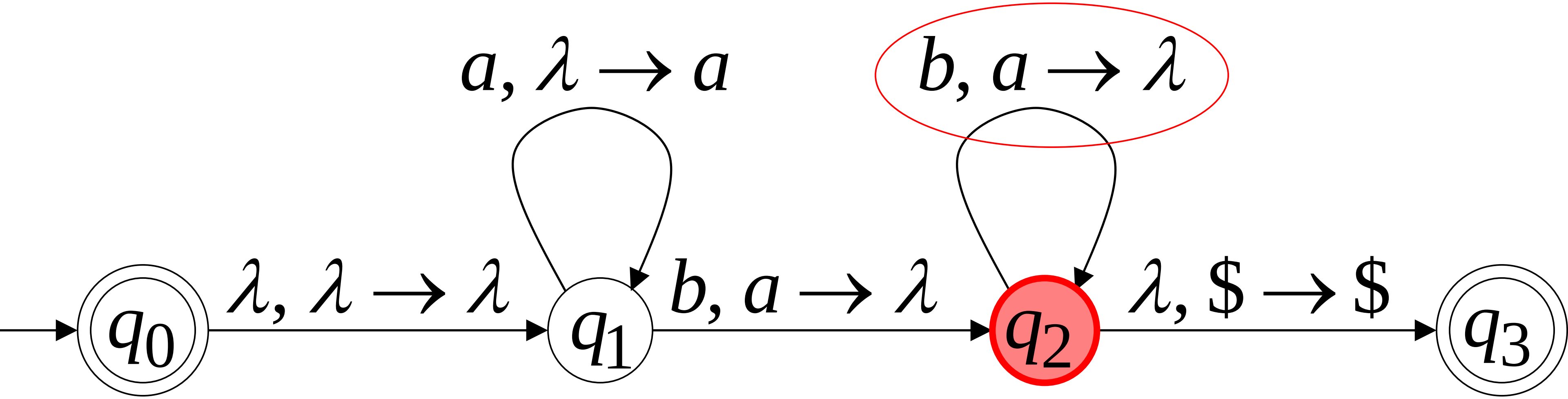
Time 7

Input

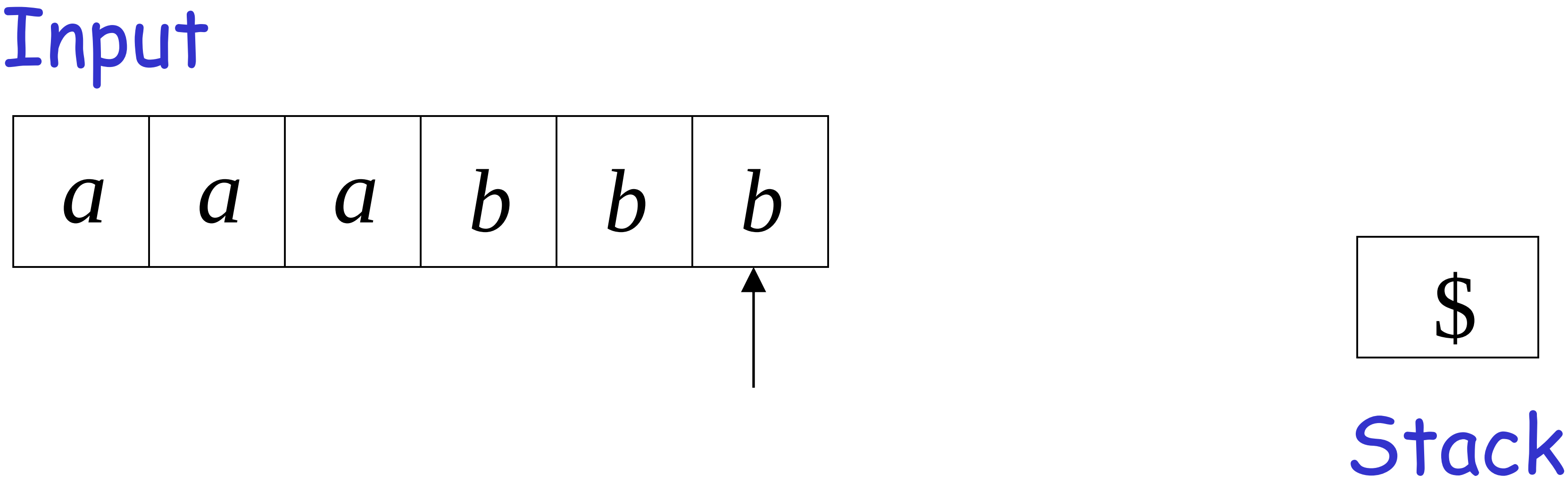


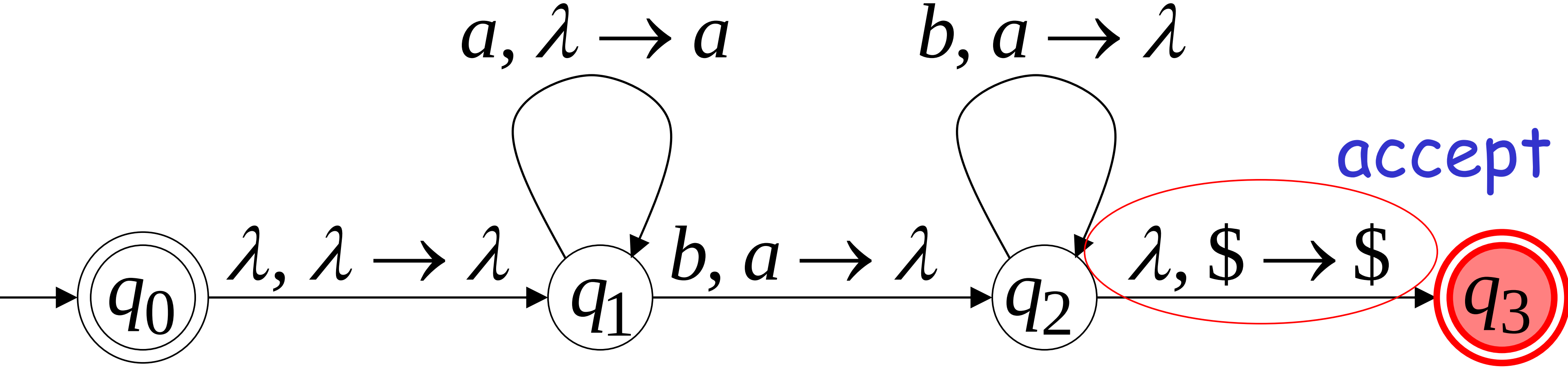


Stack



Time 8


Input
a a a b b b
$\$$
Stack

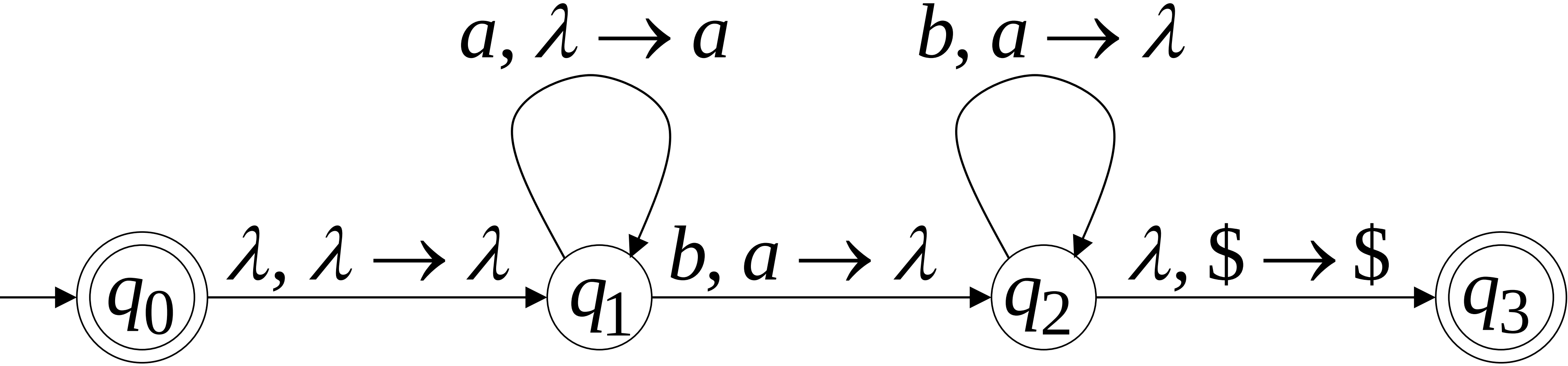

$a, \lambda \to a$
$b, a \to \lambda$
accept
q_0
$\lambda, \lambda \to \lambda$
q_1
$b, a \to \lambda$
q_2
$\lambda, \$ \to \$$
q_3

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

- All the input is consumed
- The last state is a final state

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

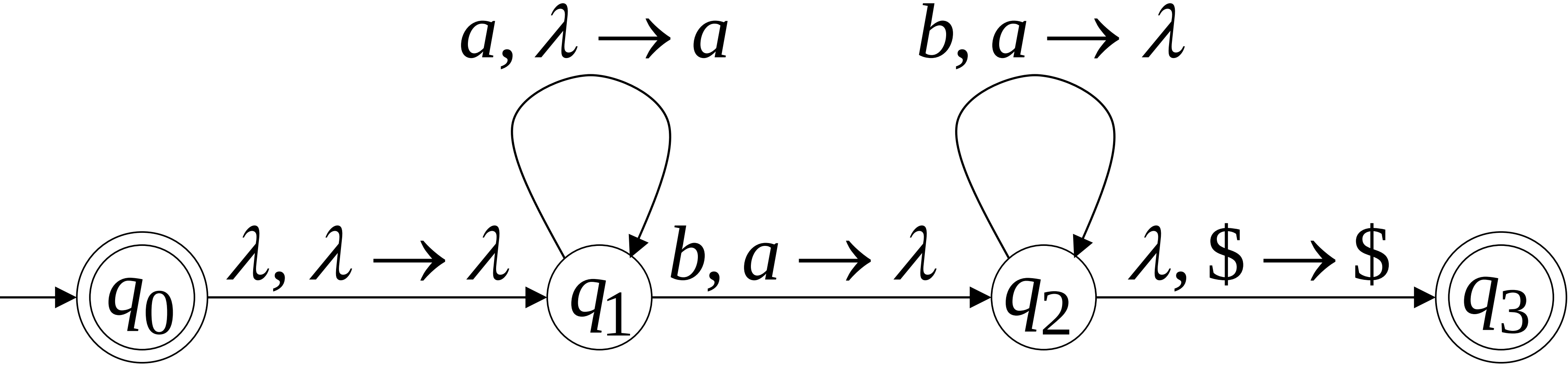
The input string $aaabbb$
is accepted by the NPDA:



In general,

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

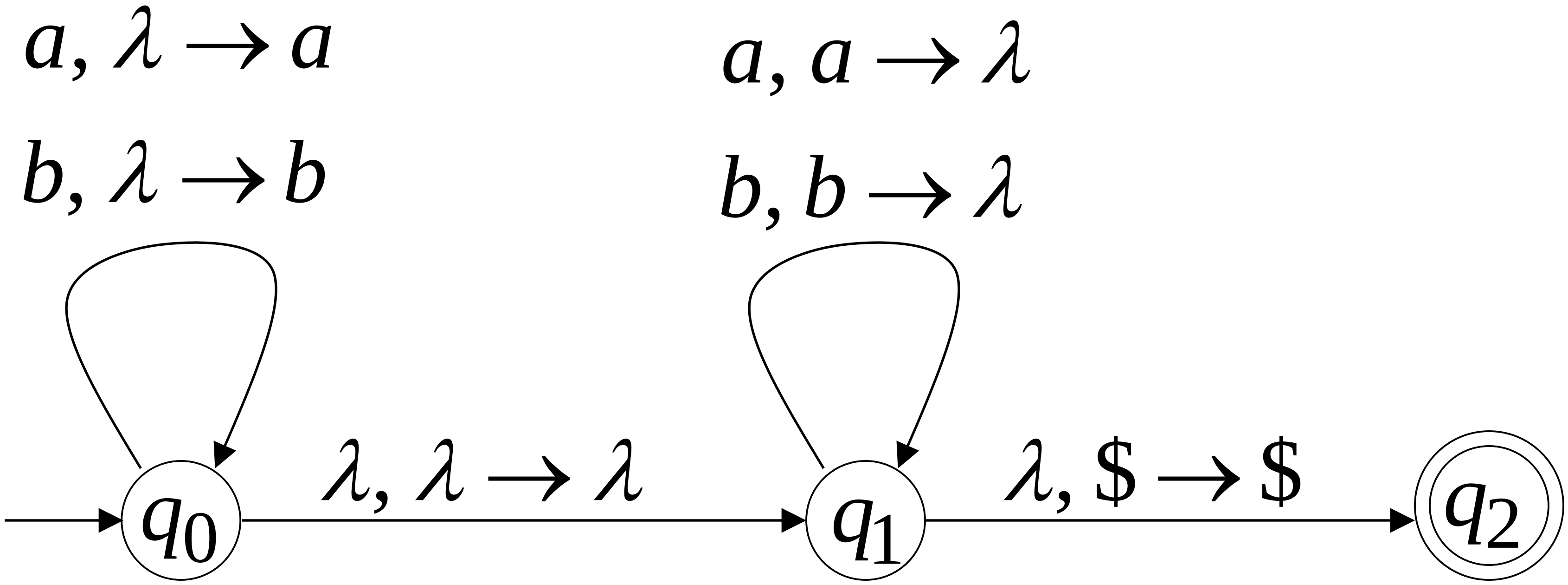
is the language accepted by the NPDA:



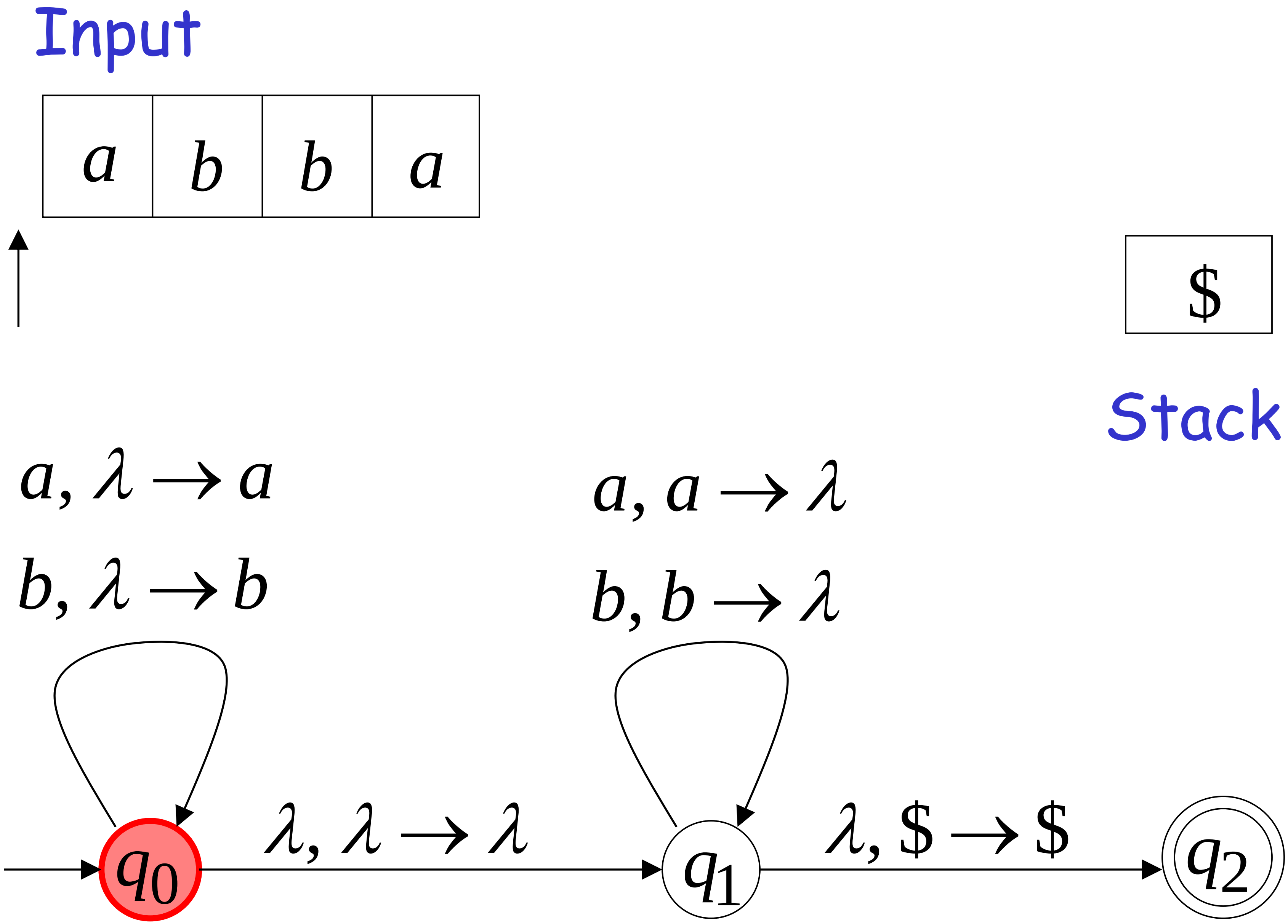
Another NPDA example

NPDA M

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



Execution Example: Time 0



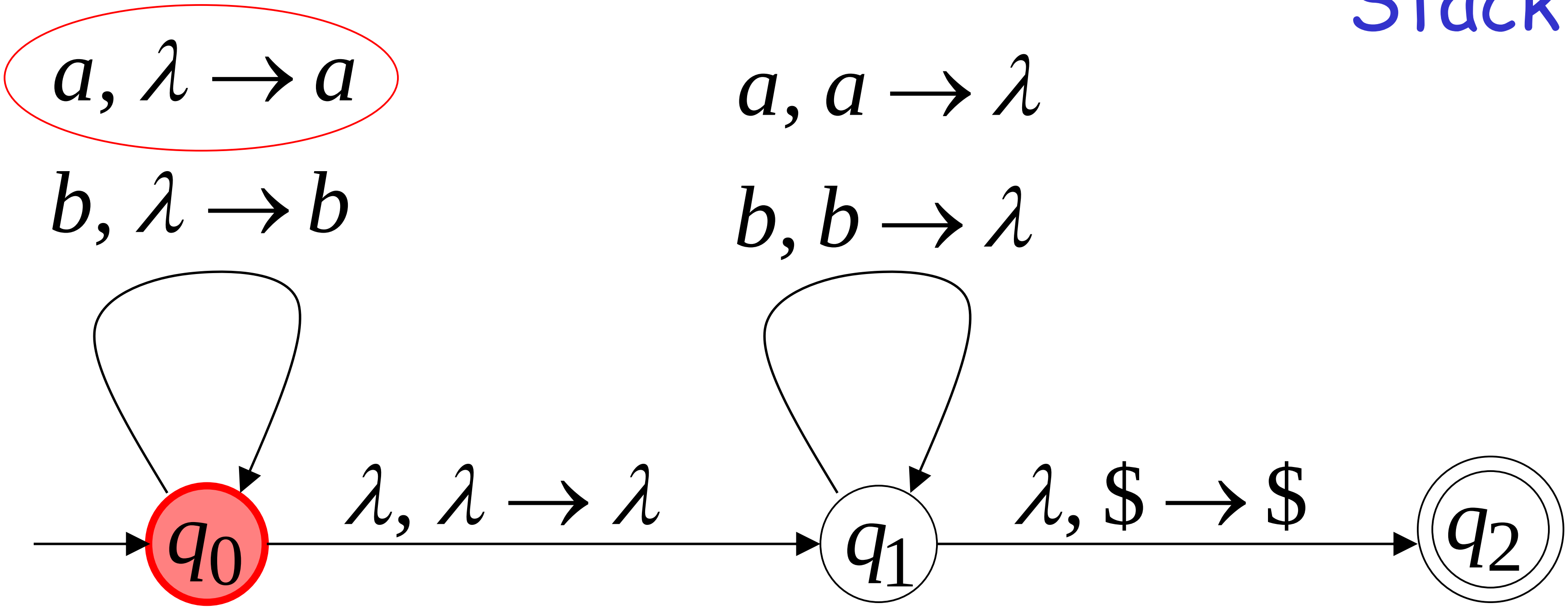
Time 1

Input

a	b	b	a

a
$\$$

Stack



$a, \lambda \rightarrow a$

$b, \lambda \rightarrow b$

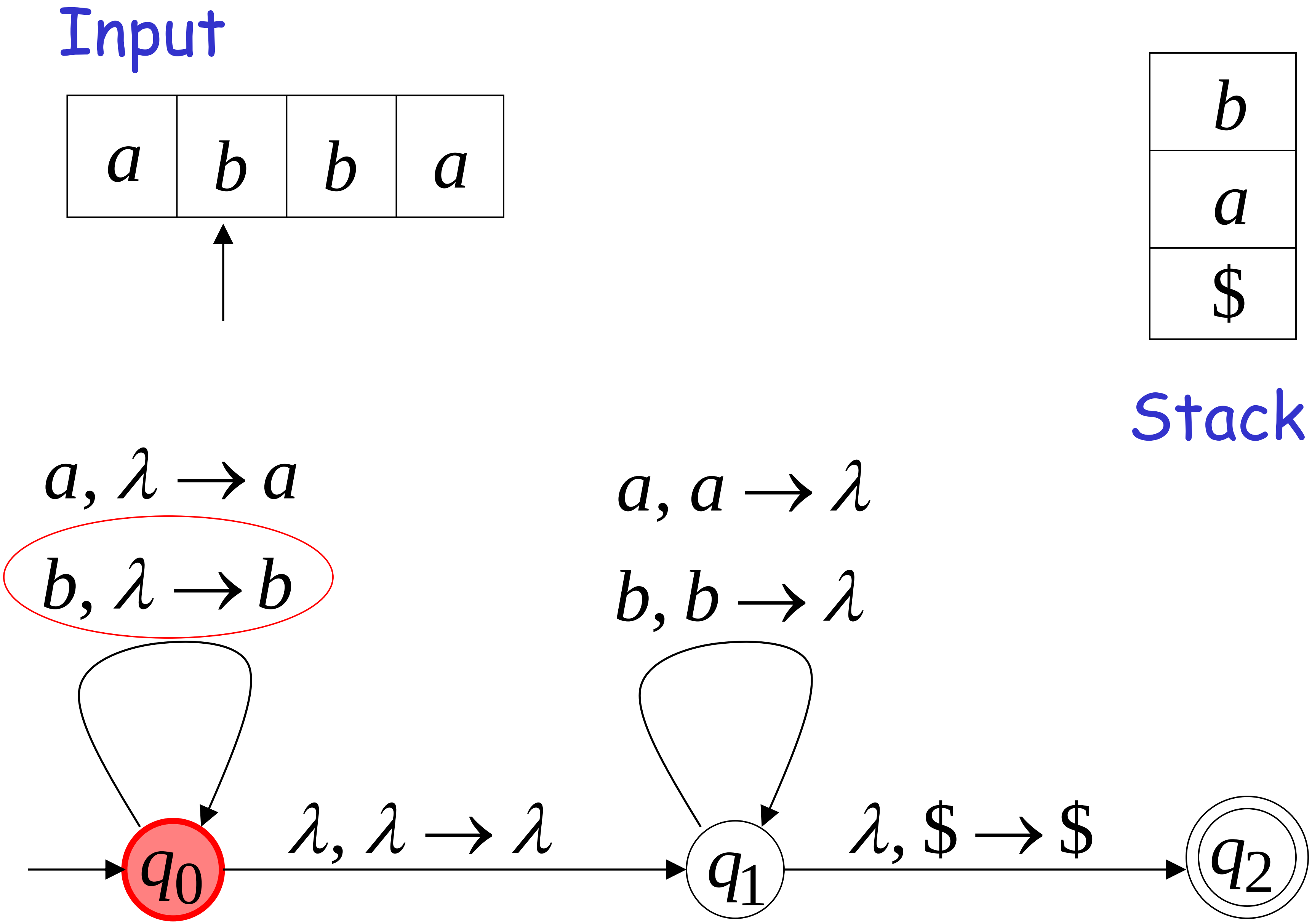
$a, a \rightarrow \lambda$

$b, b \rightarrow \lambda$

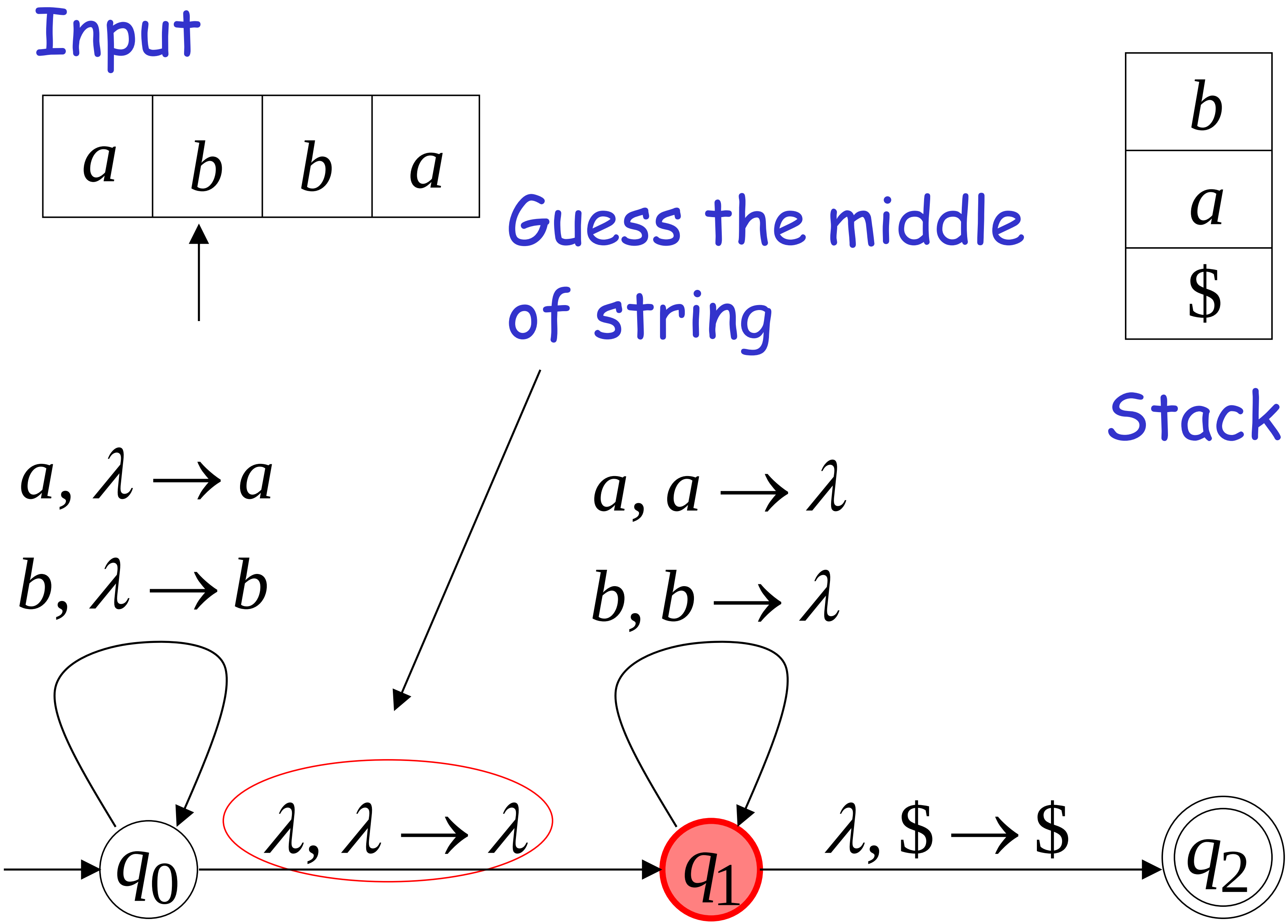
$\lambda, \lambda \rightarrow \lambda$

$\lambda, \$ \rightarrow \$$

Time 2

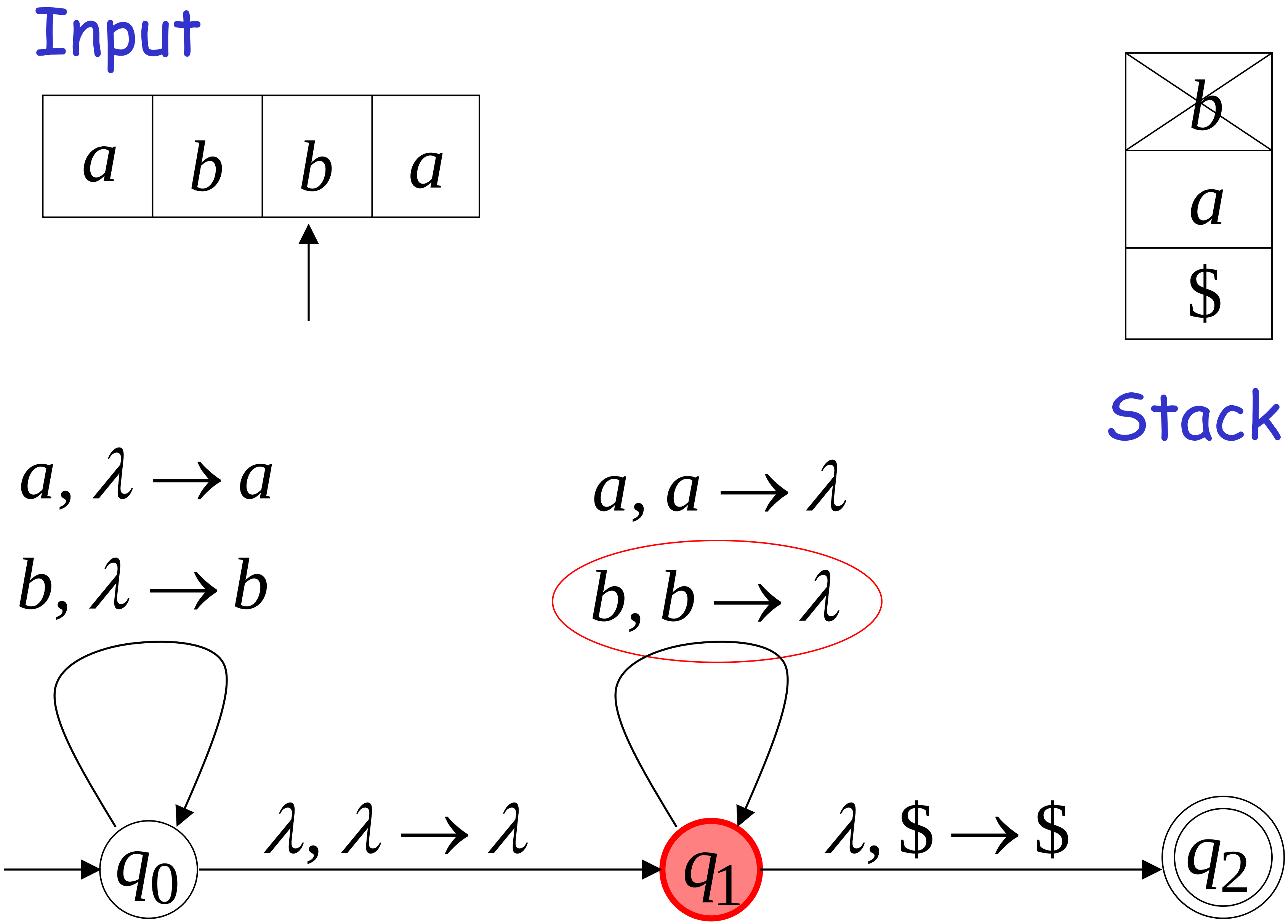

Input
a b b a
b
a
$
Stack
$a, \lambda \rightarrow a$
$b, \lambda \rightarrow b$
$a, a \rightarrow \lambda$
$b, b \rightarrow \lambda$
$\lambda, \lambda \rightarrow \lambda$
$\lambda, \$ \rightarrow \$$
q_0
q_1
q_2

Time 3



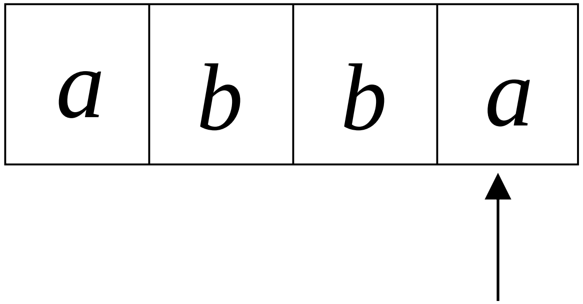
Guess the middle of string

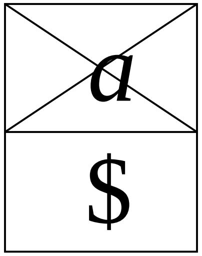
Time 4


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

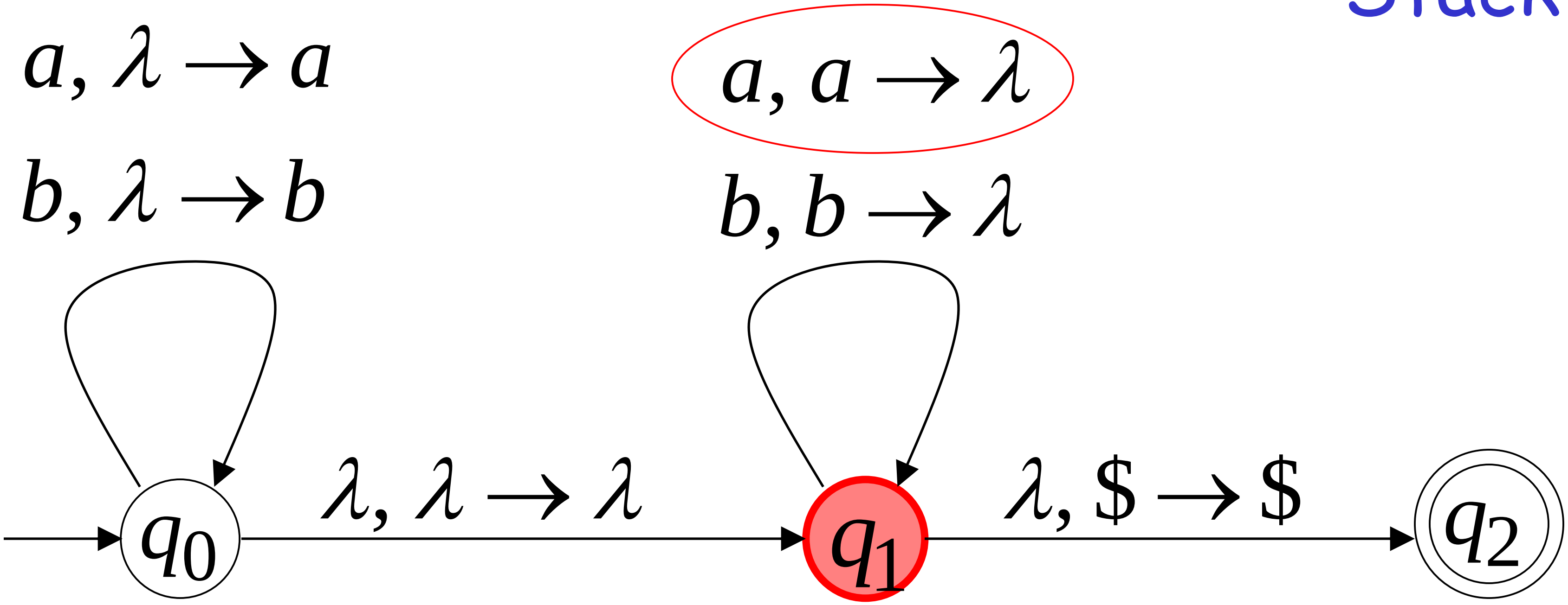
Time 5

Input

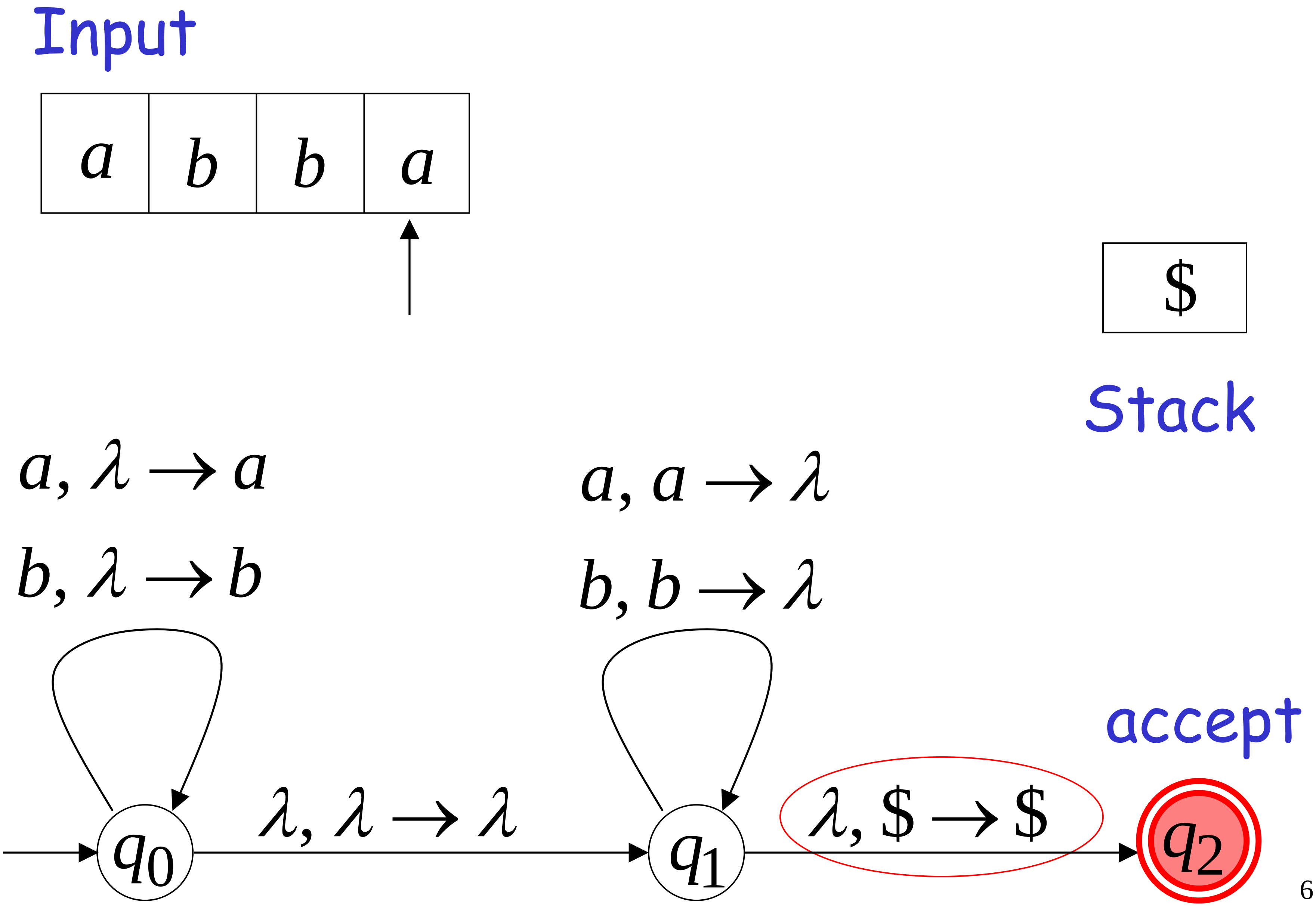




Stack

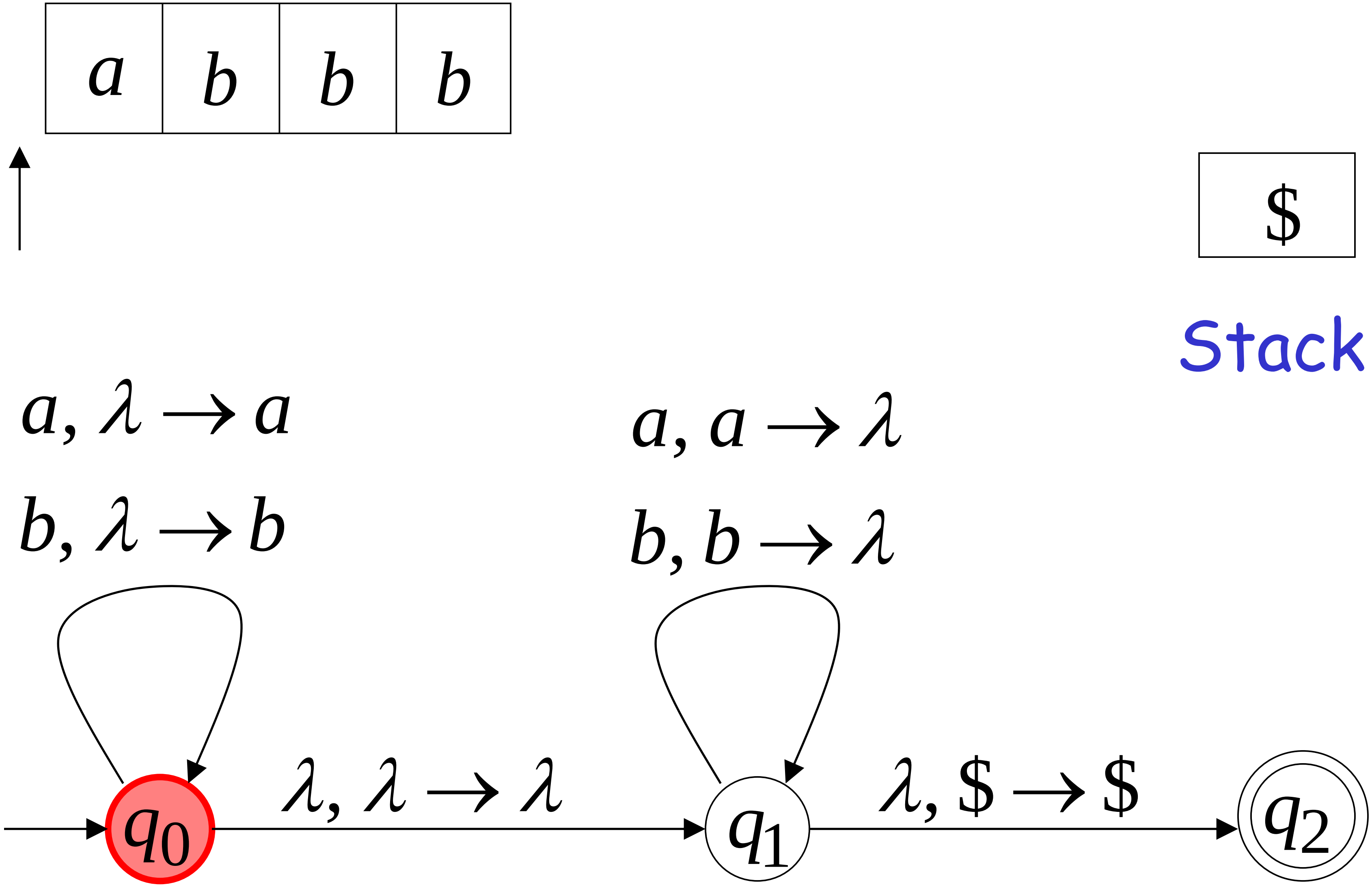


Time 6

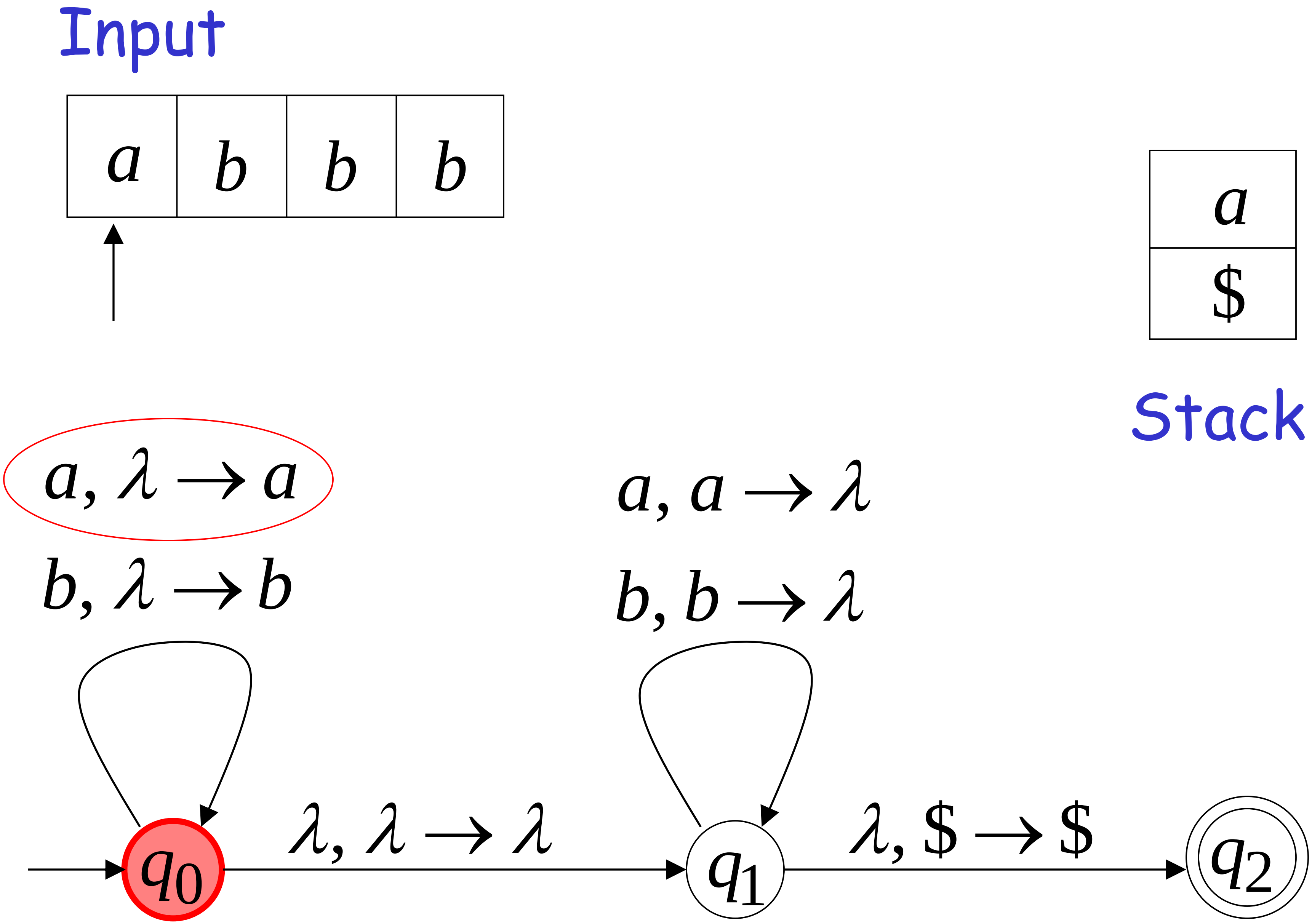

Input
a
b
b
a
$
Stack
$a, \lambda \rightarrow a$
$b, \lambda \rightarrow b$
$a, a \rightarrow \lambda$
$b, b \rightarrow \lambda$
$\lambda, \lambda \rightarrow \lambda$
$\lambda, \$ \rightarrow \$$
accept
q_0
q_1
q_2

Rejection Example: Time 0

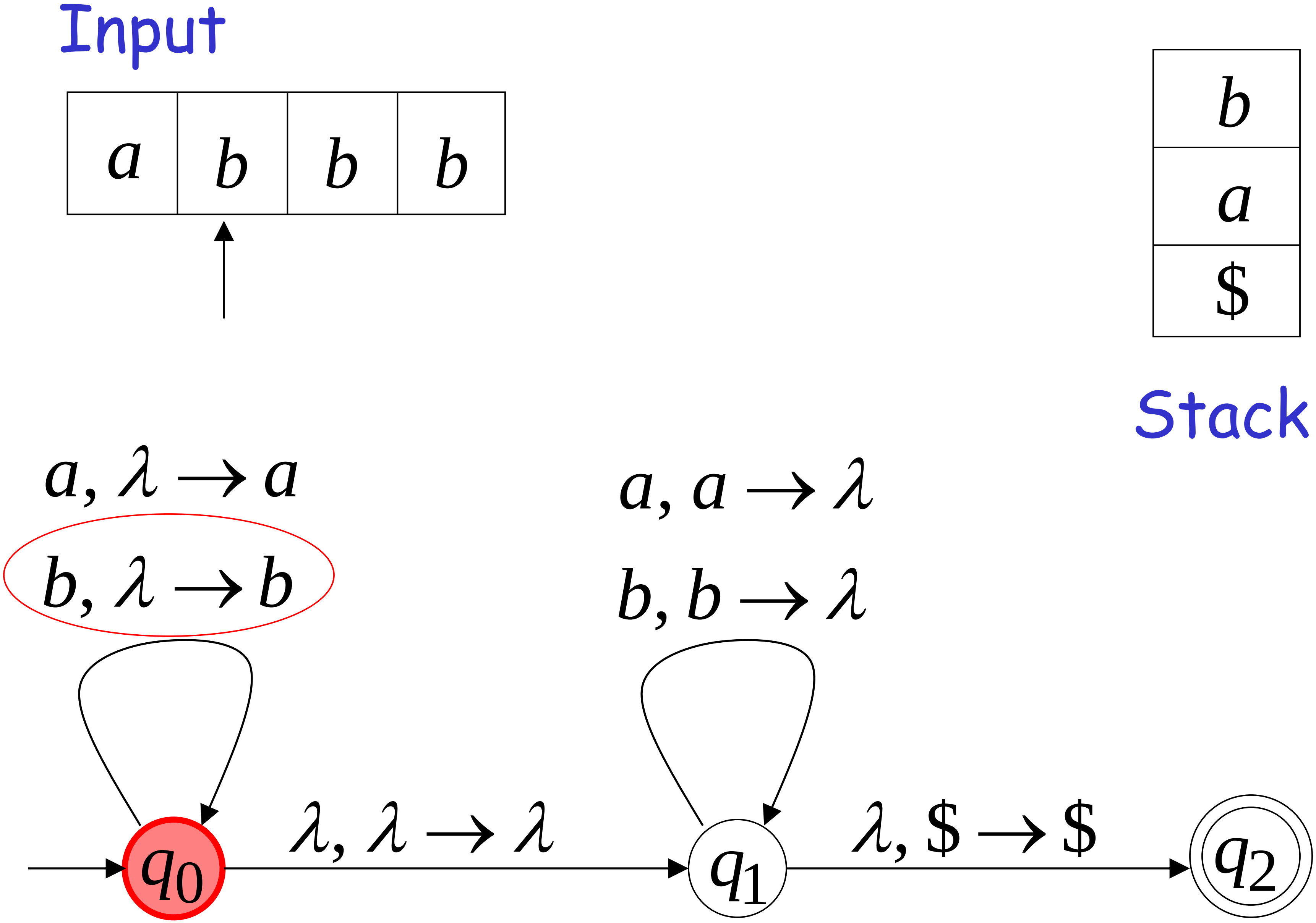
Input



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

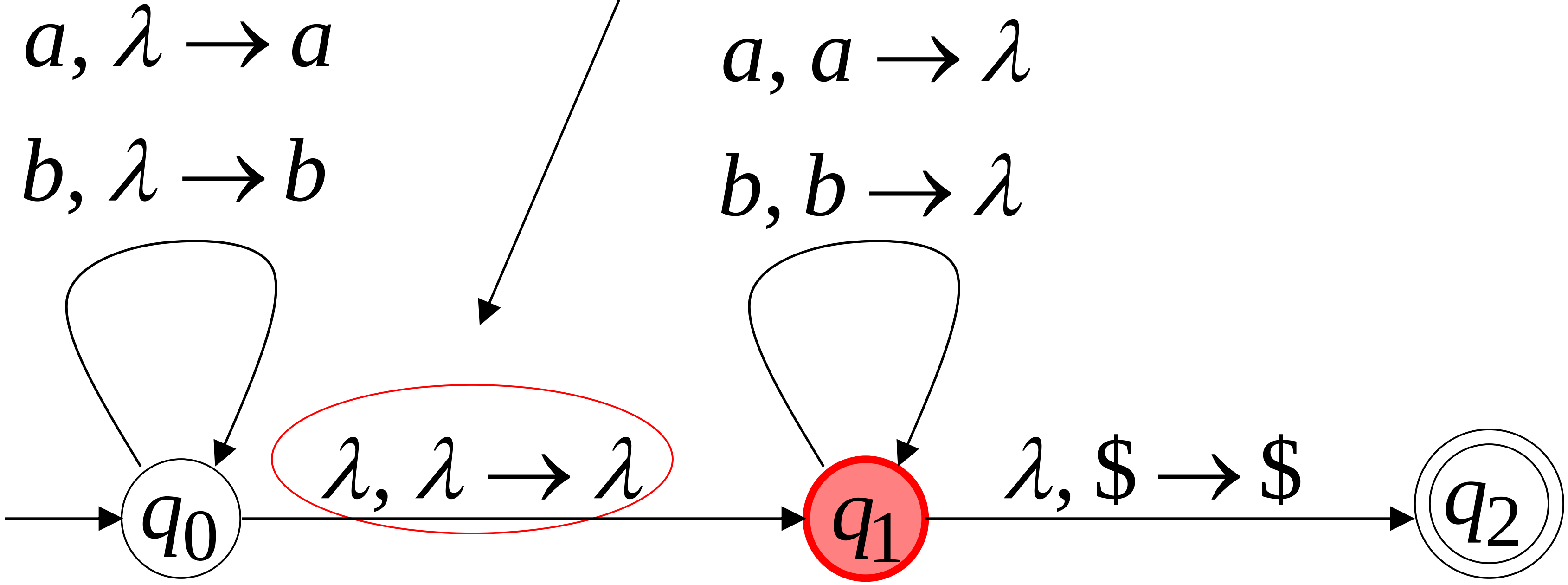
Input

a	b	b	b

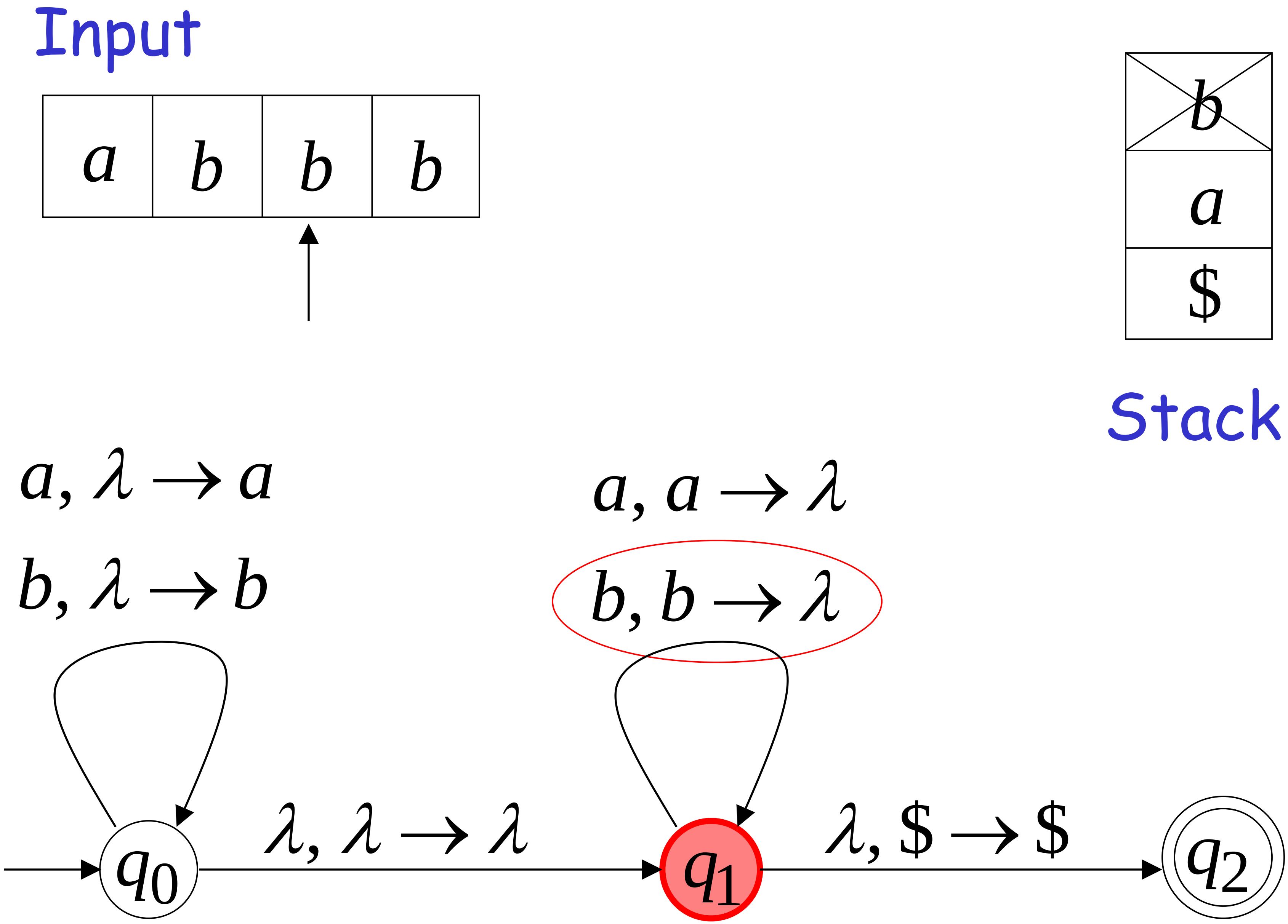
Guess the middle of string

b
a
$\$$

Stack



Time 4


Input
a b b b
$\cancel{b}$
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 5

Input

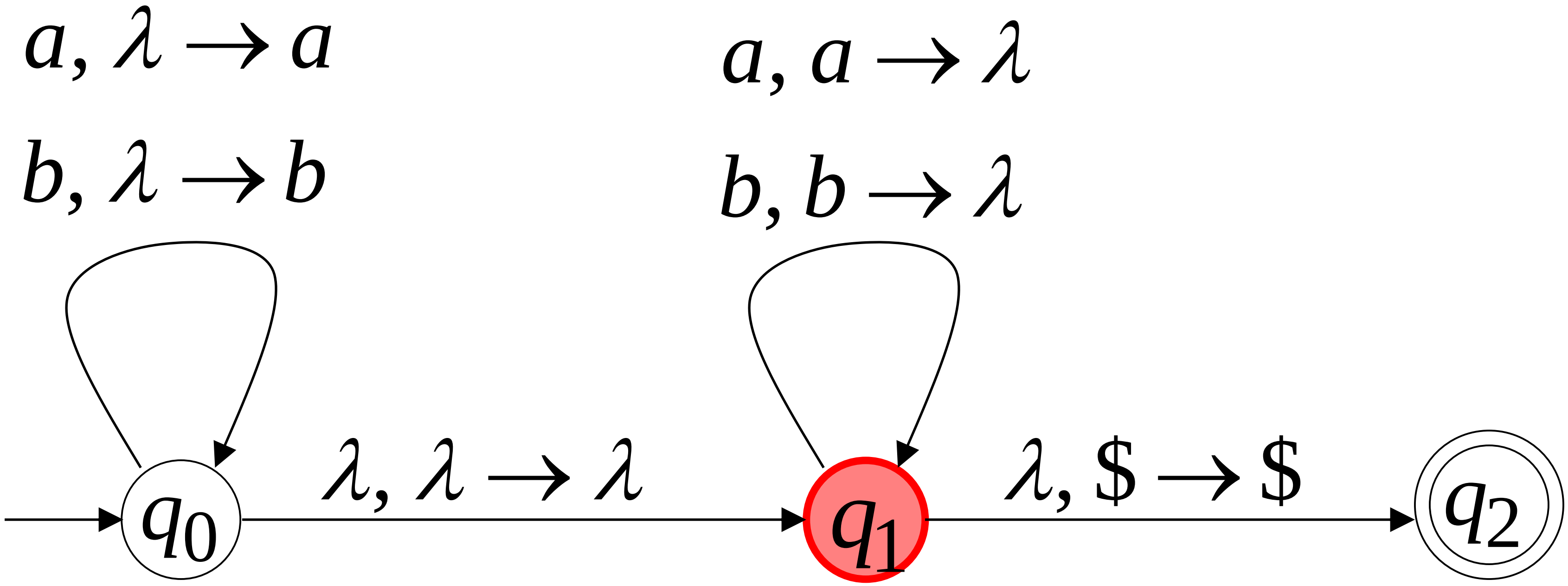
There is no possible transition.

a	b	b	b

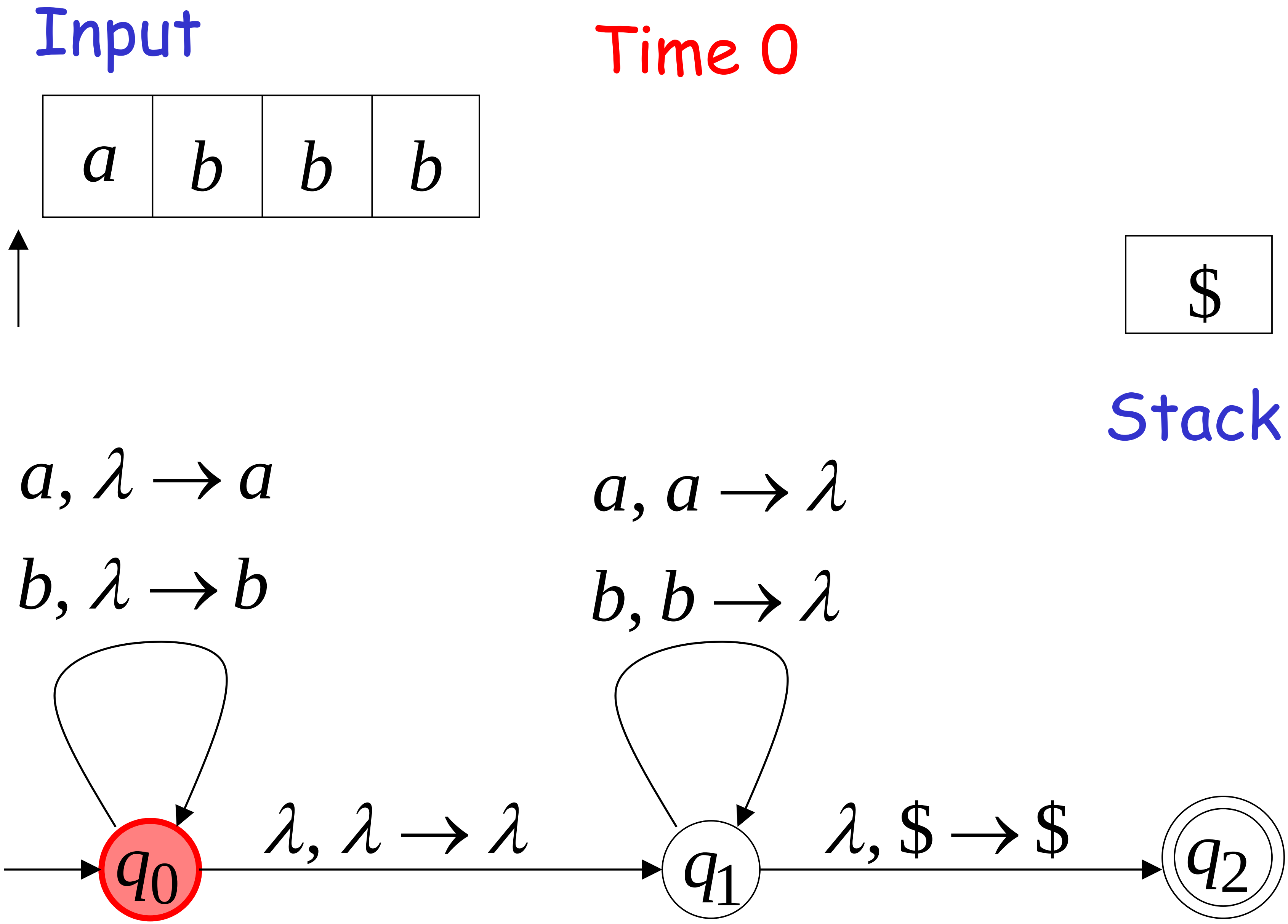
Input is not consumed

a
$\$$

Stack



Another computation on same string:

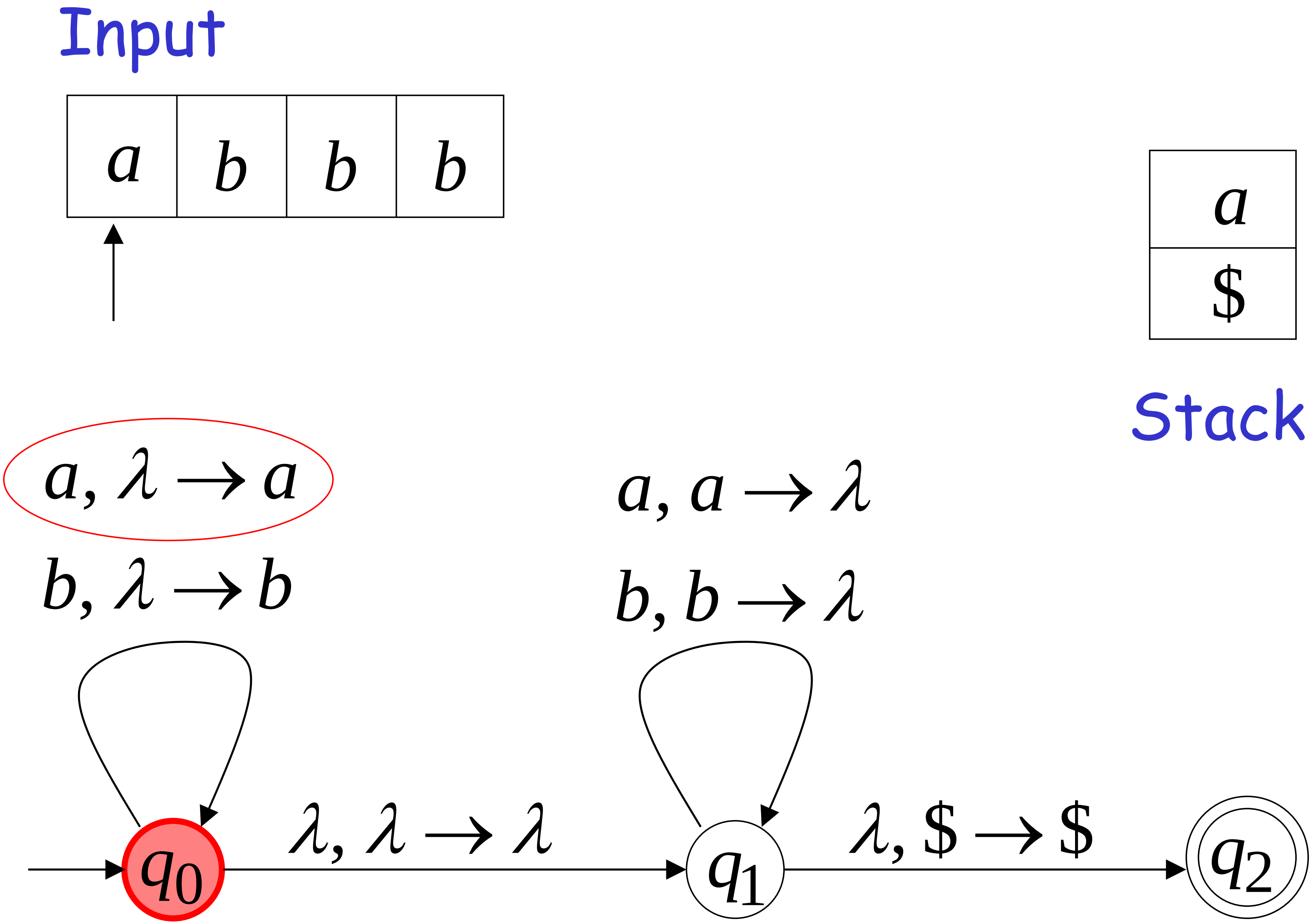


Input

Time 0

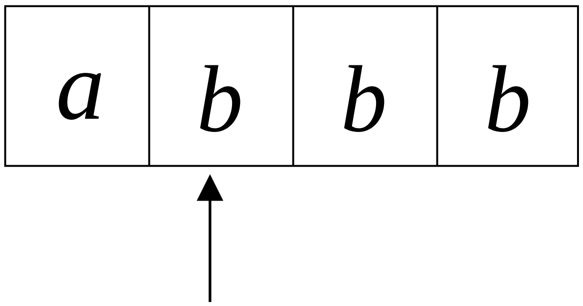
Stack

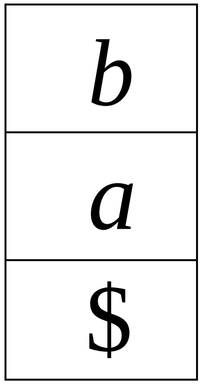
Time 1


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

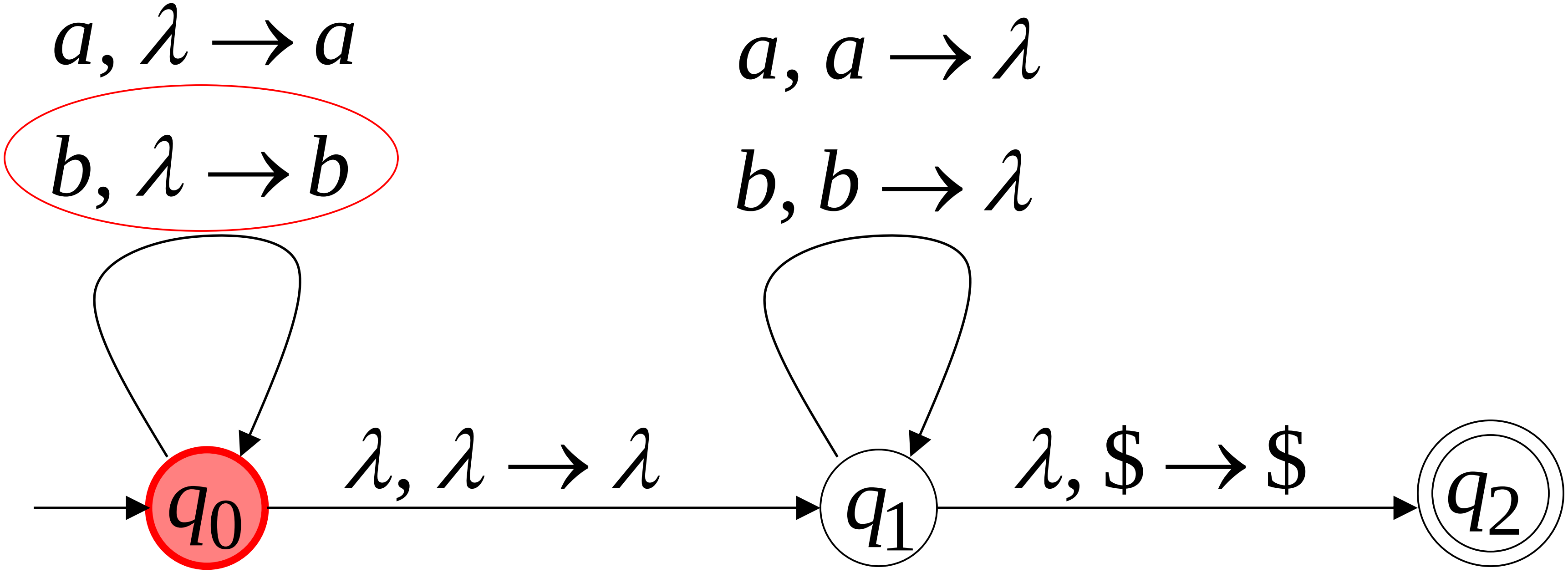
Time 2

Input

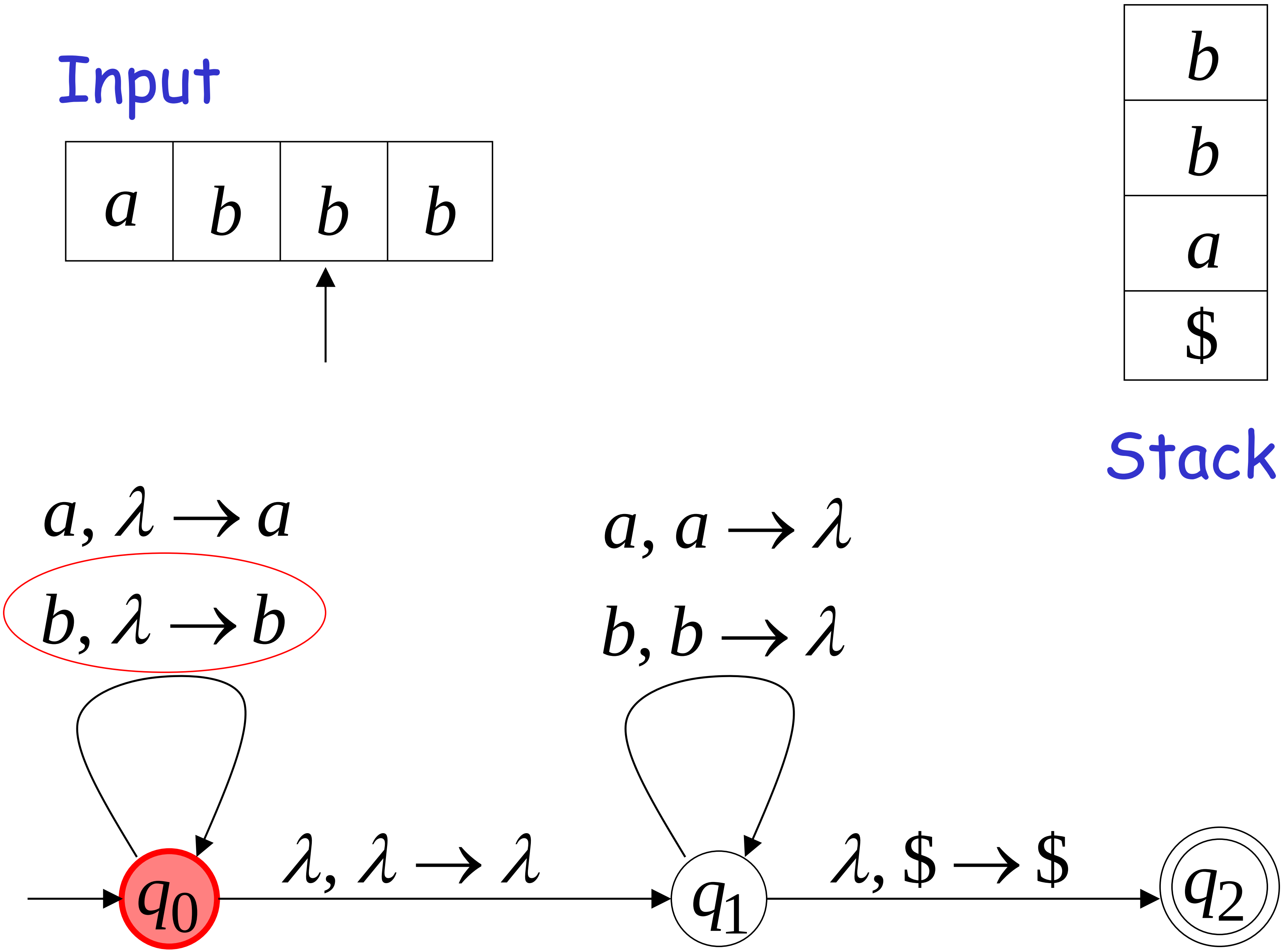




Stack



Time 3


Input
a b 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

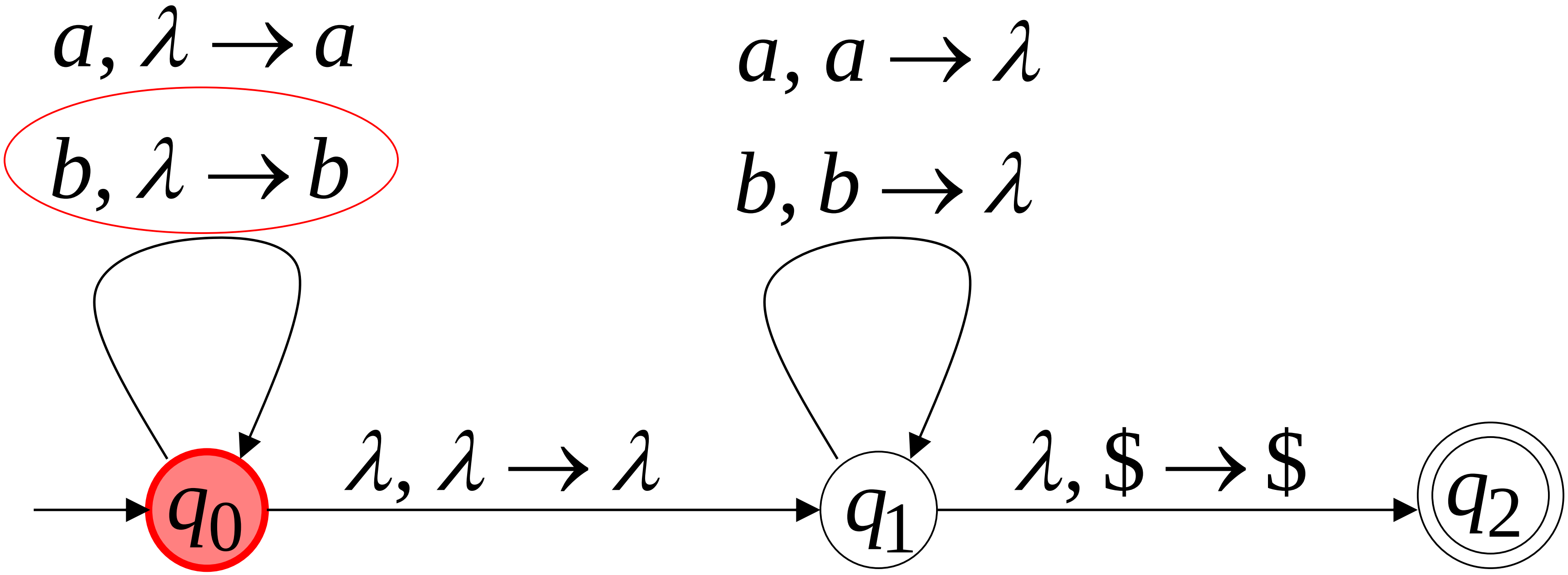
Time 4

Input

a	b	b	b

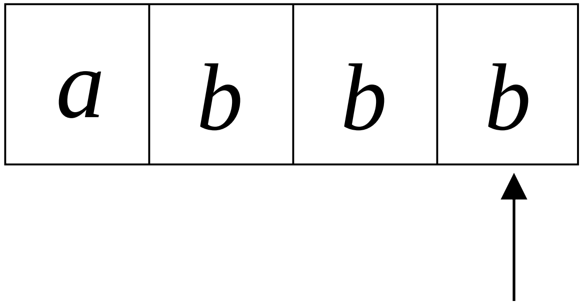
Stack

b
b
b
a
$\$$

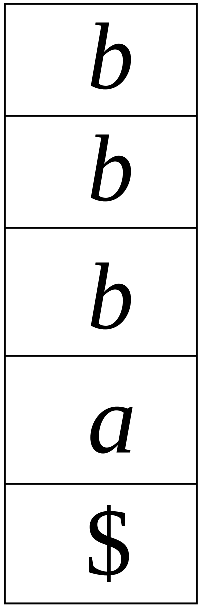


Time 5

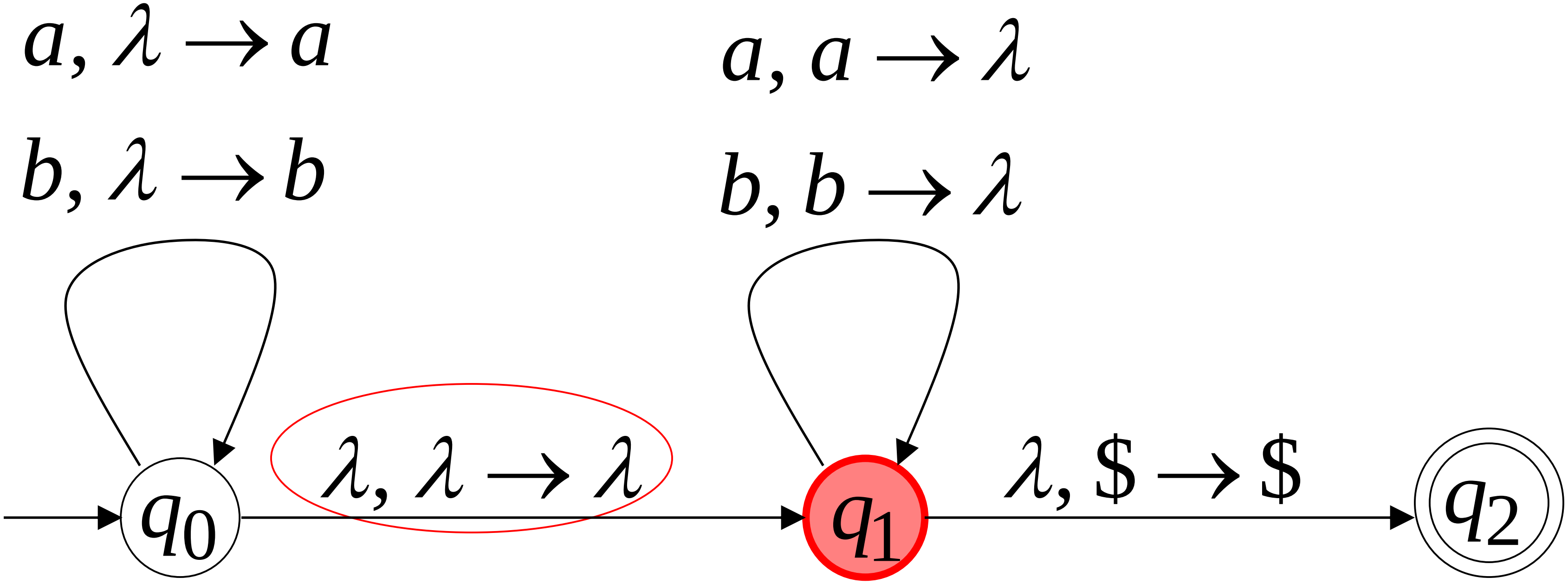
Input



No final state is reached



Stack



There is no computation that accepts string $abbb$

$$abbb \notin L(M)$$