Normal Forms for Context-free Grammars

Chomsky Normal Form

All productions have form:

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

Examples:

$$S \to AS$$
$$S \to a$$
$$A \to SA$$
$$A \to b$$

Chomsky Normal Form

$$S \to AS$$
$$S \to AAS$$
$$A \to SA$$
$$A \to aa$$

Not Chomsky Normal Form

Convertion to Chomsky Normal Form

Example:

$$S \rightarrow ABa$$

$$A \rightarrow aab$$

$$B \rightarrow Ac$$

Not Chomsky Normal Form

Introduce variables for terminals: T_a, T_b, T_c

$$S \to ABa$$
$$A \to aab$$
$$B \to Ac$$

$$S \to ABT_a$$
$$A \to T_aT_aT_b$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

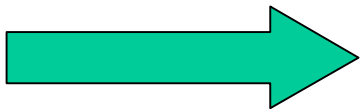
Introduce intermediate variable: V_1

$$S \to ABT_a$$
$$A \to T_a T_a T_b$$
$$B \to AT_c$$
$$T_a \to a$$
$$T_b \to b$$
$$T_c \to c$$

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

Introduce intermediate variable: V_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:

Initial grammar

$$S \to ABa$$

$$A \to aab$$

$$B \to Ac$$

$$S \to AV_1$$

$$V_1 \to BT_a$$

$$A \to T_aV_2$$

$$V_2 \to T_aT_b$$

$$B \to AT_c$$

$$T_a \to a$$

$$T_b \to b$$

$$T_c \to c$$

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 $T_a \to a$

In productions: replace a with T_a

New variable: T_a

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

with

$$A \to C_1 V_1$$

$$V_1 \to C_2 V_2$$

$$\cdots$$

$$V_{n-2} \to C_{n-1} C_n$$

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

Theorem: For any context-free grammar there is an equivalent grammar in Chomsky Normal Form

Observations

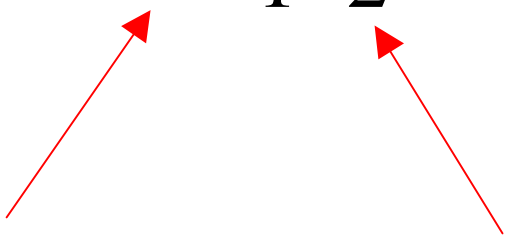
- Chomsky normal forms are good for parsing and proving theorems

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

Greinbach Normal Form

All productions have form:

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



symbol variables

Examples:

$$S \to cAB$$
$$A \to aA \,|\, bB \,|\, b$$
$$B \to b$$

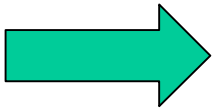
Greinbach Normal Form

$$S \to abSb$$
$$S \to aa$$

Not Greinbach Normal Form

Conversion to Greinbach Normal Form:

$$S \to abSb$$
$$S \to aa$$



$$S \to aT_bST_b$$
$$S \to aT_a$$
$$T_a \to a$$
$$T_b \to b$$

Greinbach Normal Form

Theorem: For any context-free grammar there is an equivalent grammar in Greinbach Normal Form

Observations

- Greinbach normal forms are very good for parsing

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

An Application of Chomsky Normal Forms

The CYK Membership Algorithm

Input:

- Grammar G in Chomsky Normal Form
- String w

Output:

find if $w \in L(G)$

The Algorithm

Input example:

- Grammar G:

$$S \to AB$$

$$A \to BB$$

$$A \to a$$

$$B \to AB$$

$$B \to b$$

- String w : $aabbb$

aabbb

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

$S \rightarrow AB$

$A \rightarrow BB$

$A \rightarrow a$

$B \rightarrow AB$

$B \rightarrow b$

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

$S \rightarrow AB$

$A \rightarrow BB$

$A \rightarrow a$

$B \rightarrow AB$

$B \rightarrow b$

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

$S \rightarrow AB$

$A \rightarrow BB$

$A \rightarrow a$

$B \rightarrow AB$

$B \rightarrow b$

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				

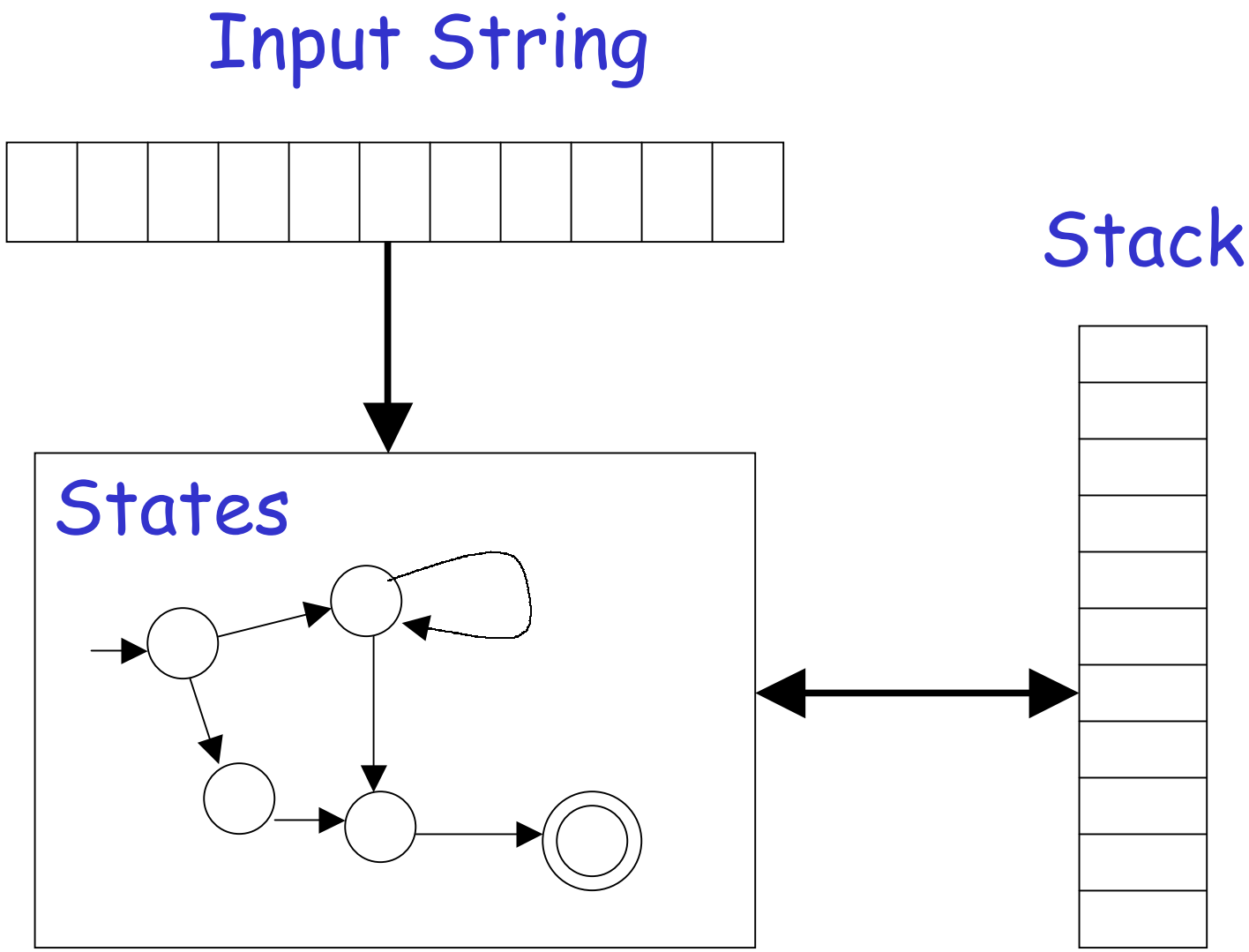
Therefore: $aabbb \in L(G)$

Time Complexity: $|w|^3$

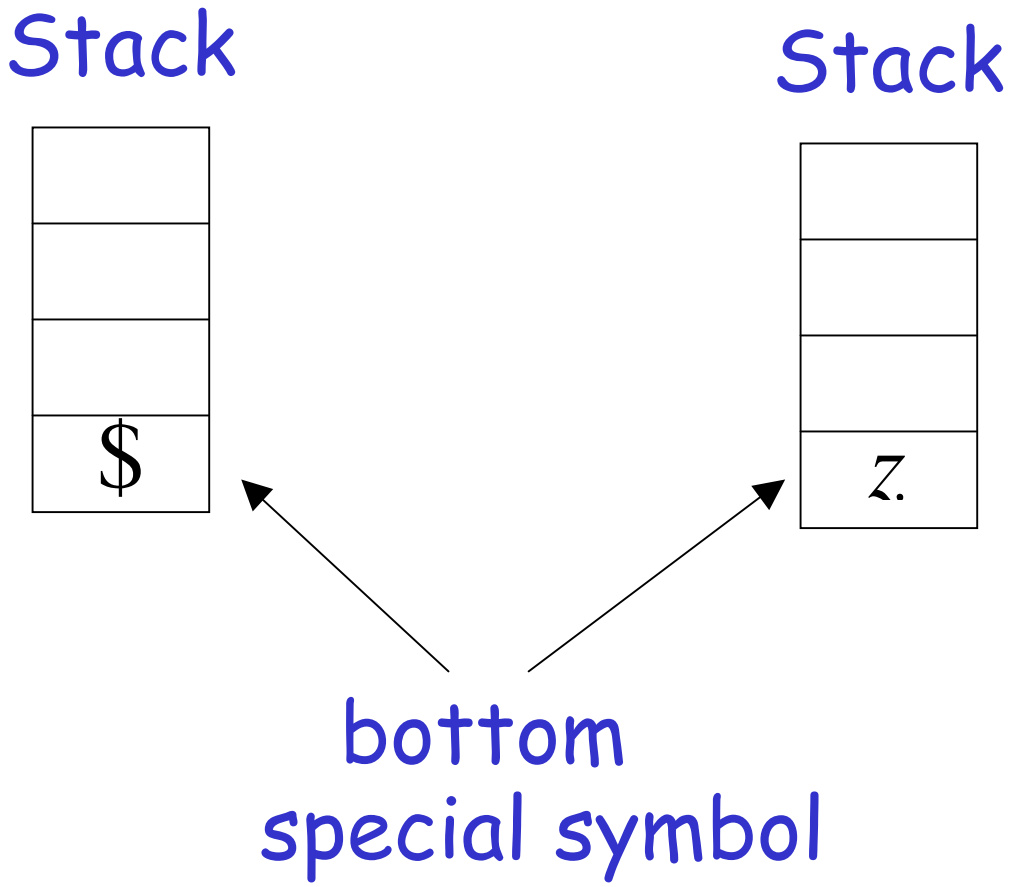
Observation: The CYK algorithm can be easily converted to a parser

Pushdown Automata
PDAs

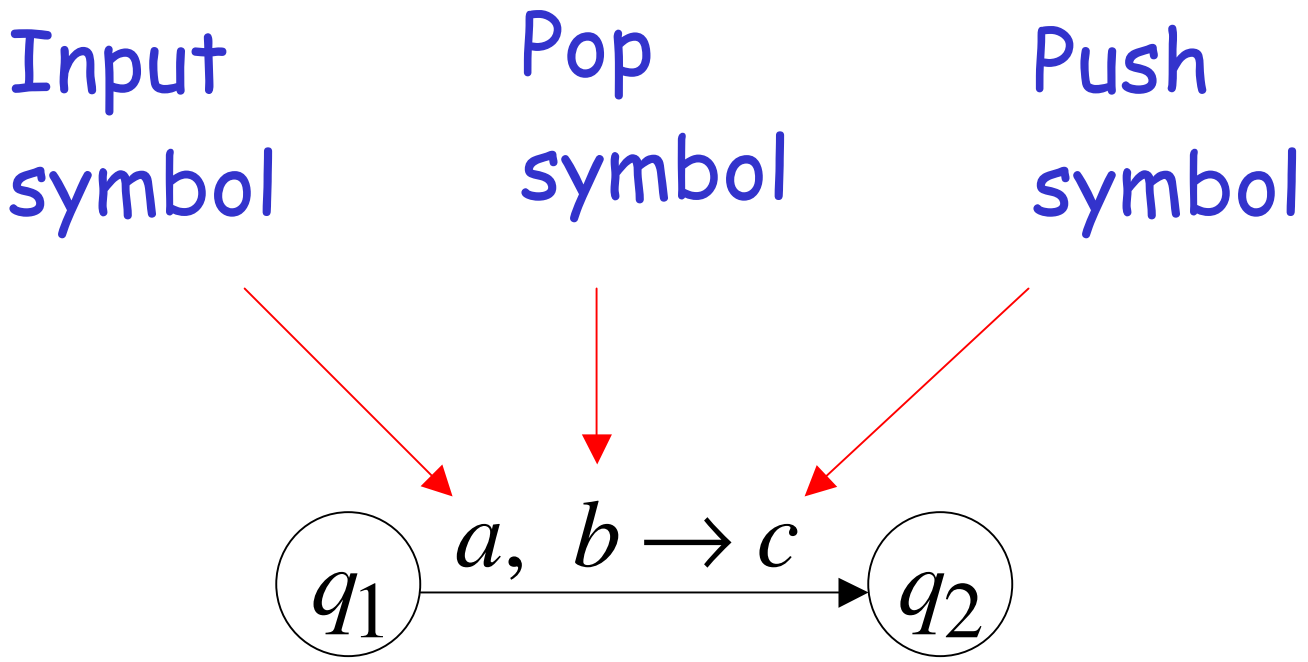
Pushdown Automaton -- PDA

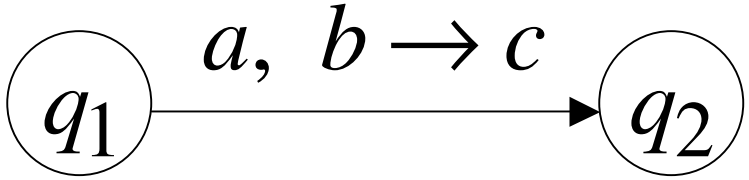


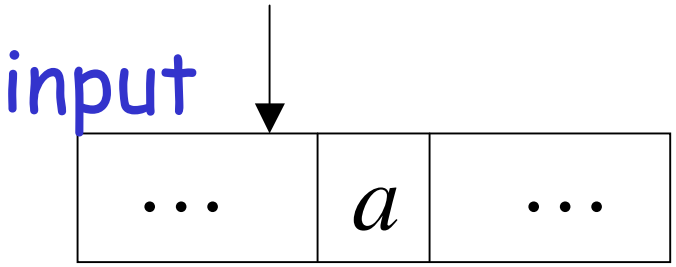
Initial Stack Symbol

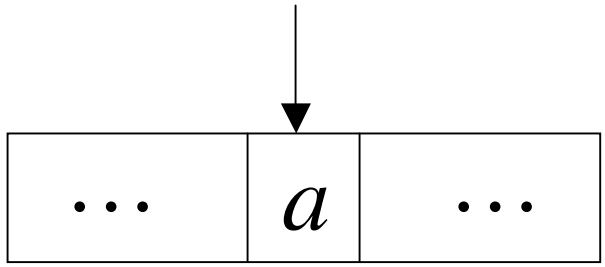


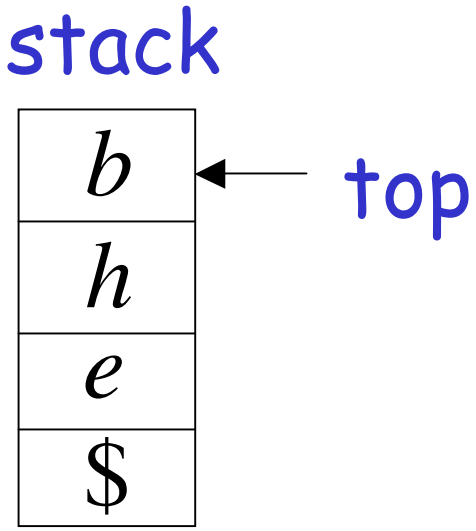
The States

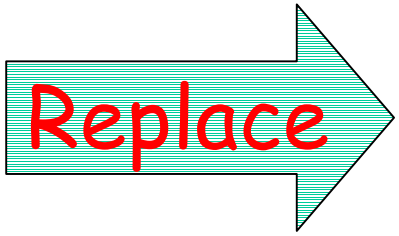


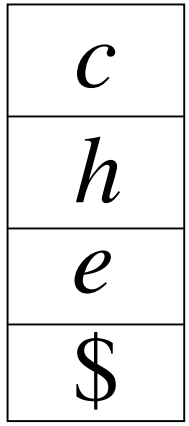

$a,\ b \rightarrow c$
q_1
q_2

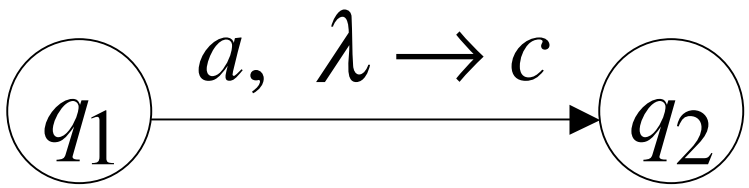

input
…
a
…

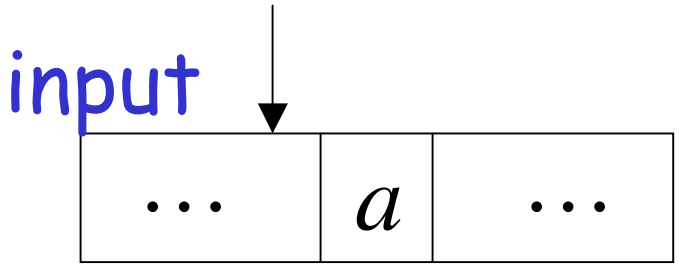

…
a
…

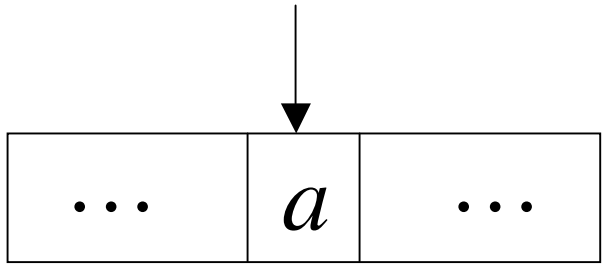

stack
b
h
e
$
top

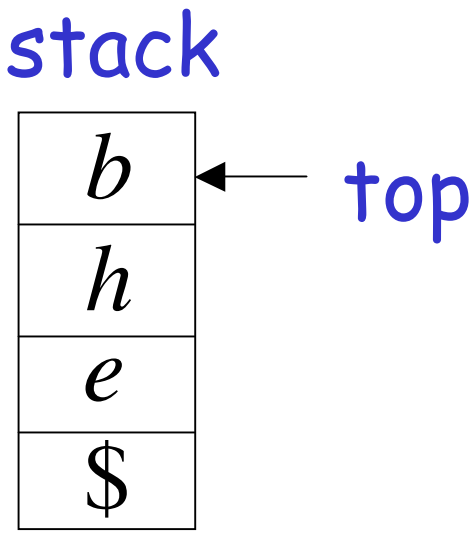

Replace

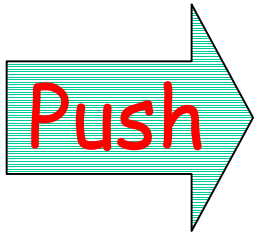

c
h
e
$

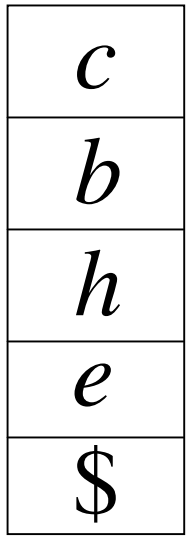

$a,\ \lambda \to c$
q_1
q_2

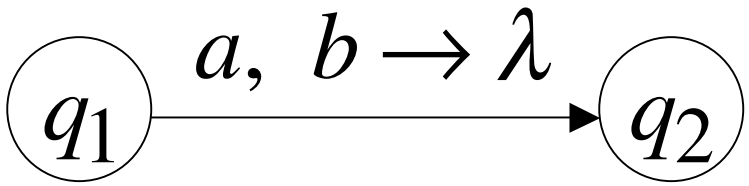

input
…
a
…


…
a
…

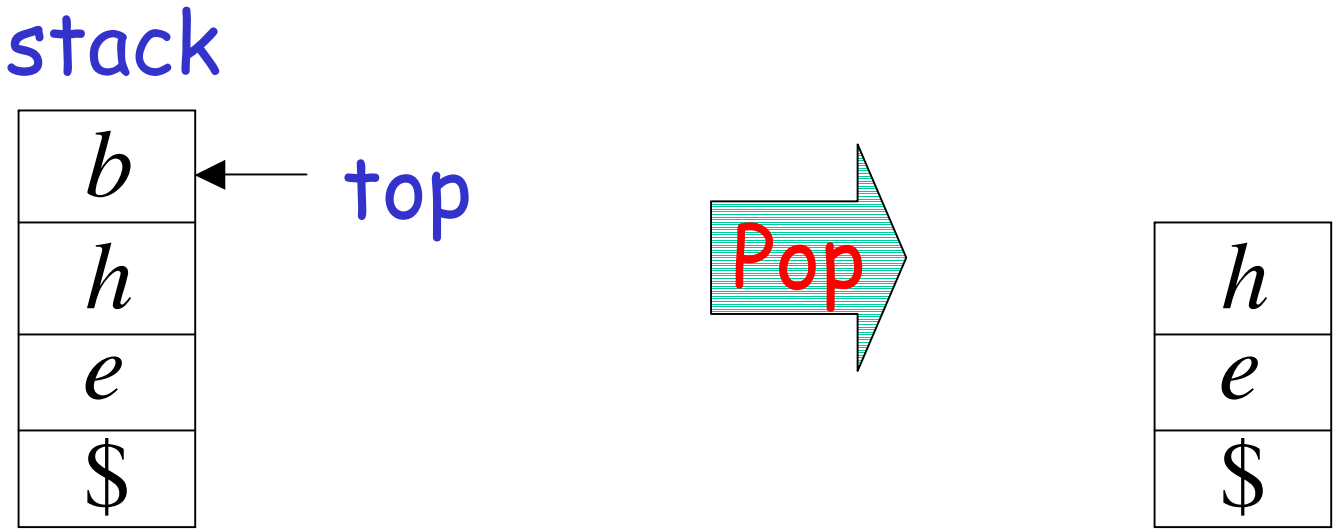

stack
b
h
e
$\$$
top

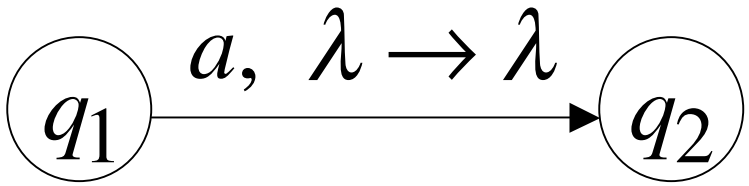

Push


c
b
h
e
$\$$

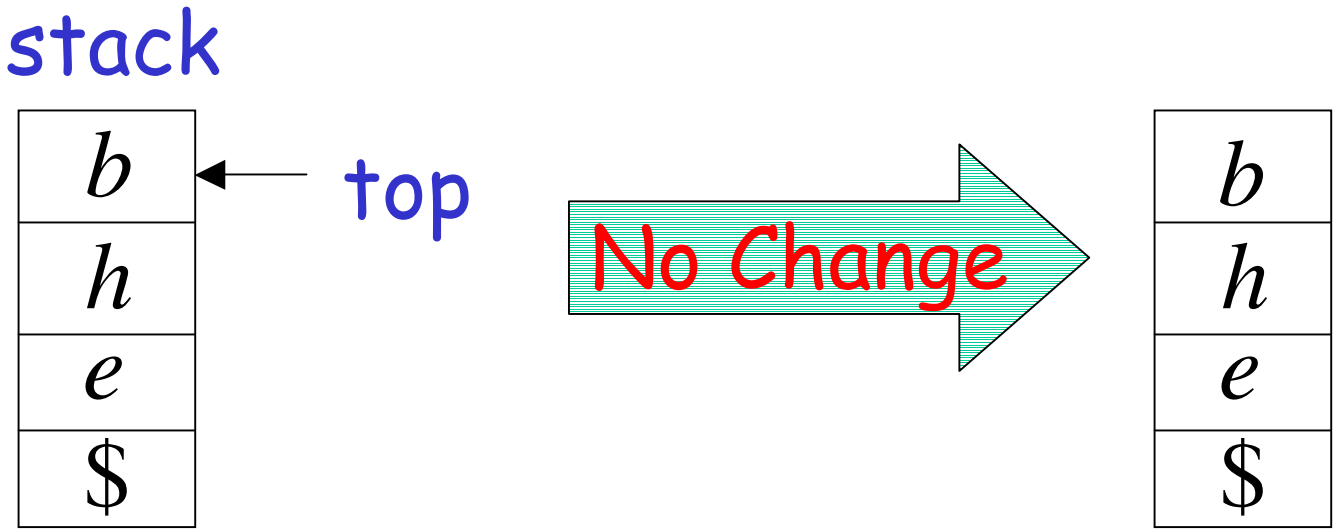

$a,\ b \to \lambda$
q_1
q_2

input
… a …
… a …

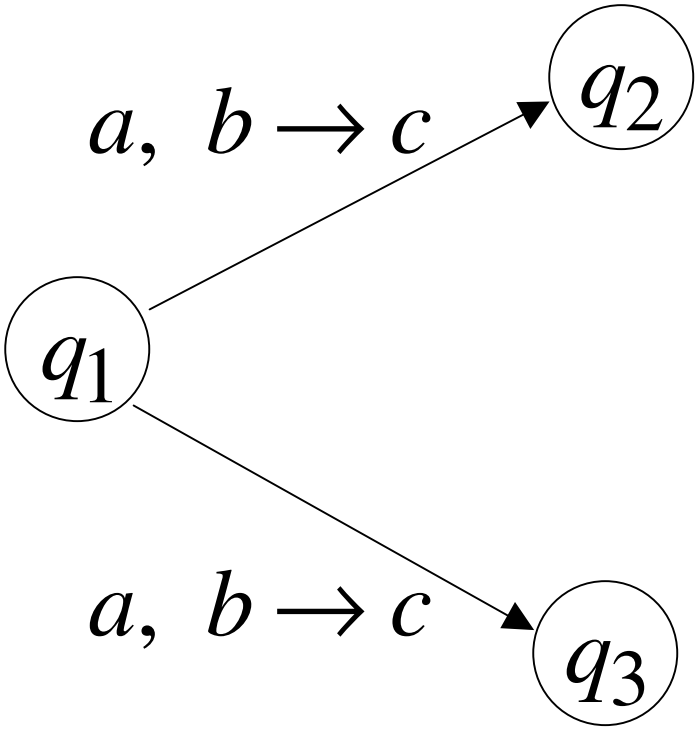

stack
b
h
e
$
top
Pop
h
e
$

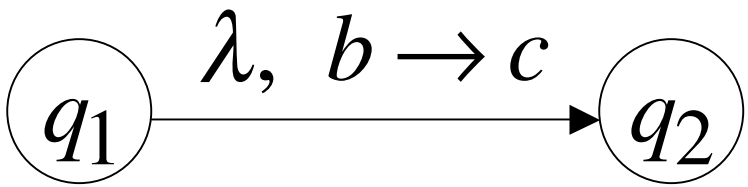

$a,\ \lambda \rightarrow \lambda$
q_1
q_2

input
… a …
… a …


stack
b
h
e
$
top
No Change
b
h
e
$

Non-Determinism

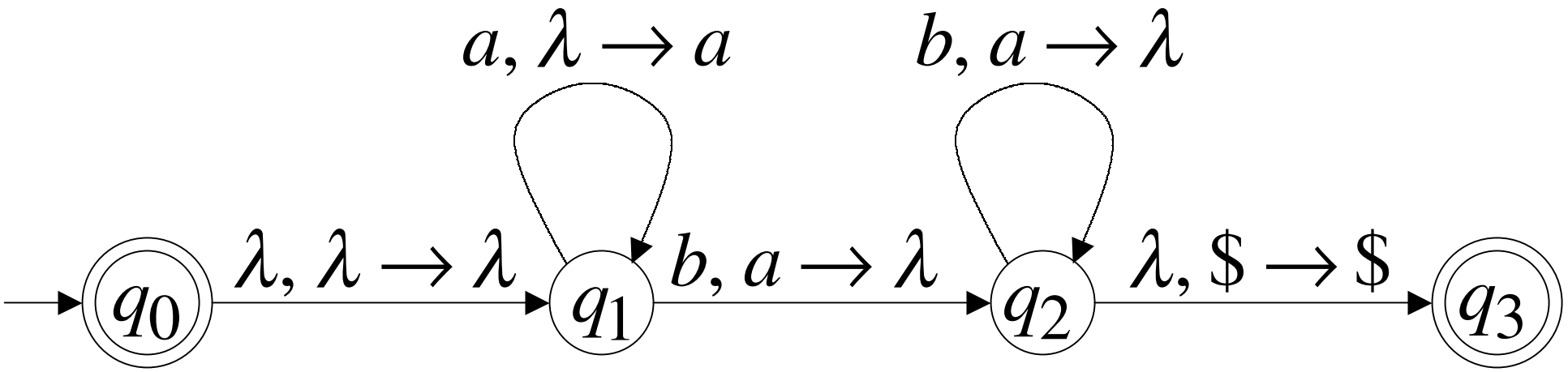




$\lambda-$transition

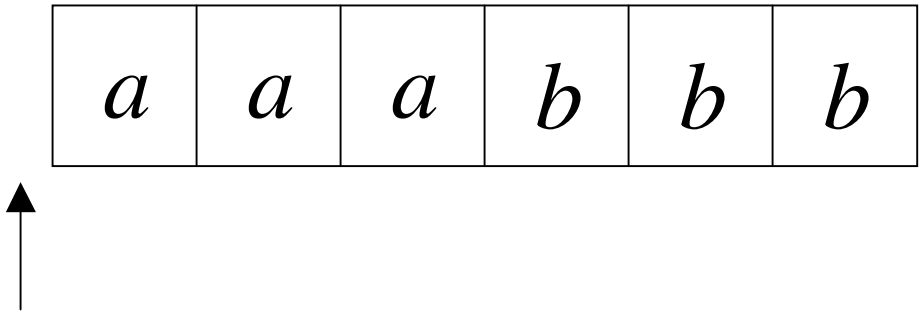
NPDA: Non-Deterministic PDA

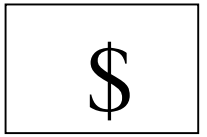
Example:



Execution Example: Time 0

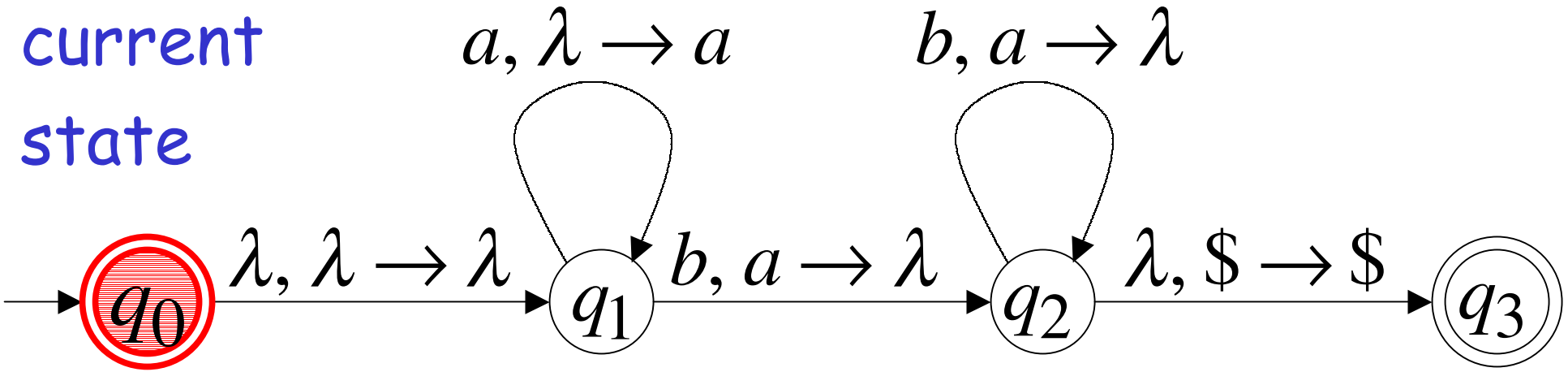
Input





Stack

current state



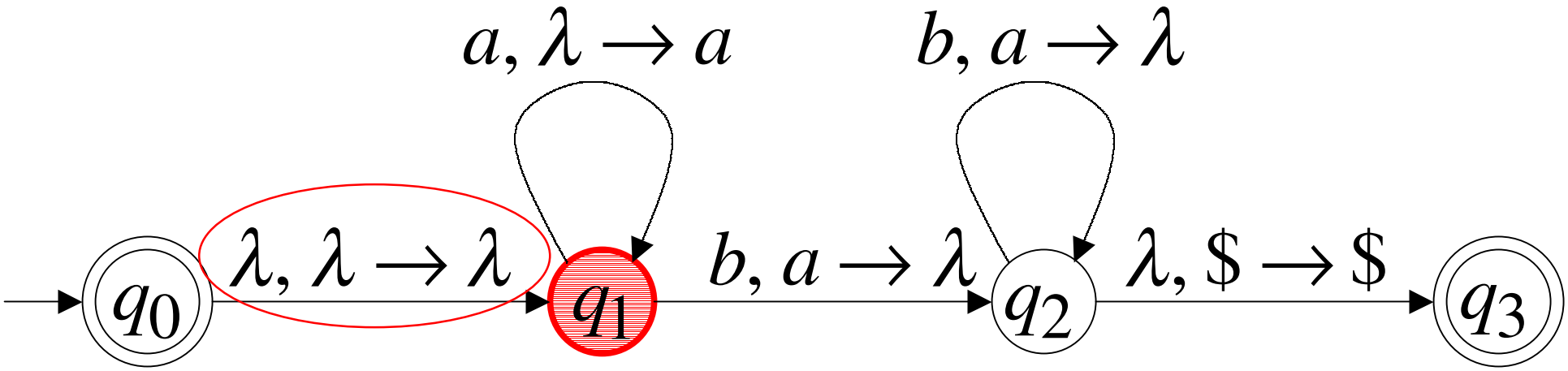
Time 1

Input

a	a	a	b	b	b

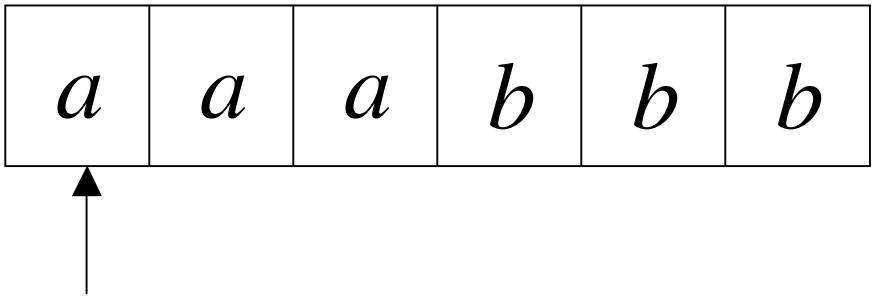
$

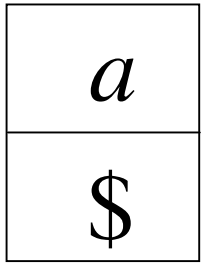
Stack



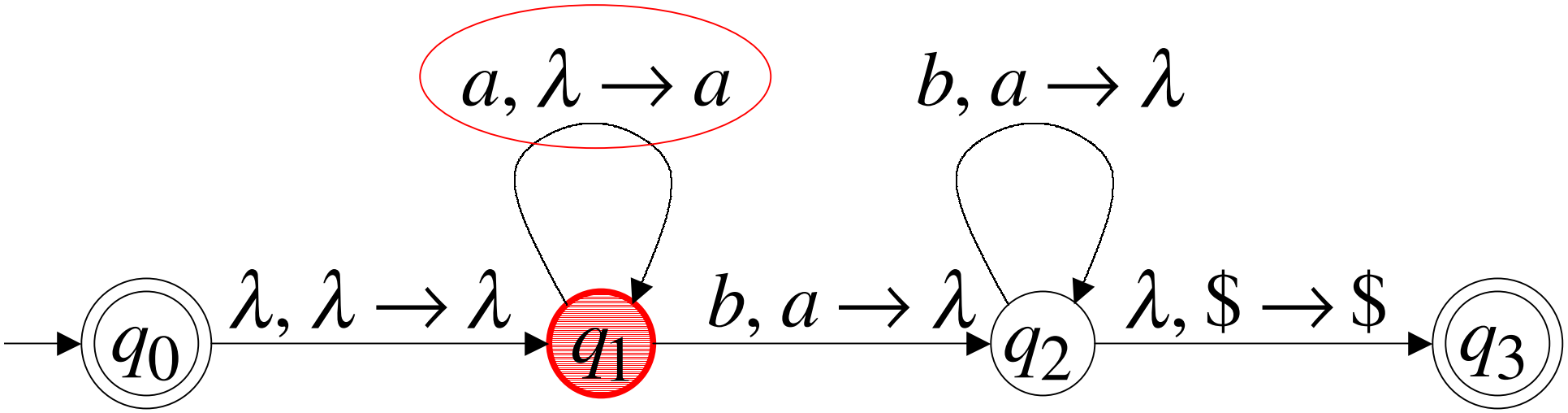
Time 2

Input



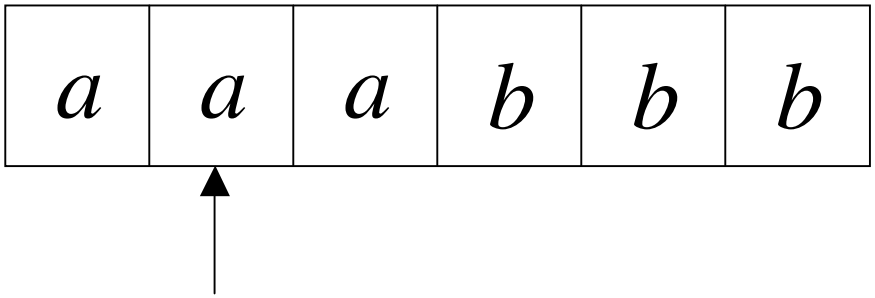


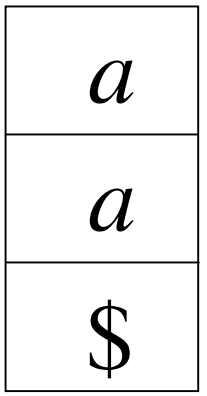
Stack



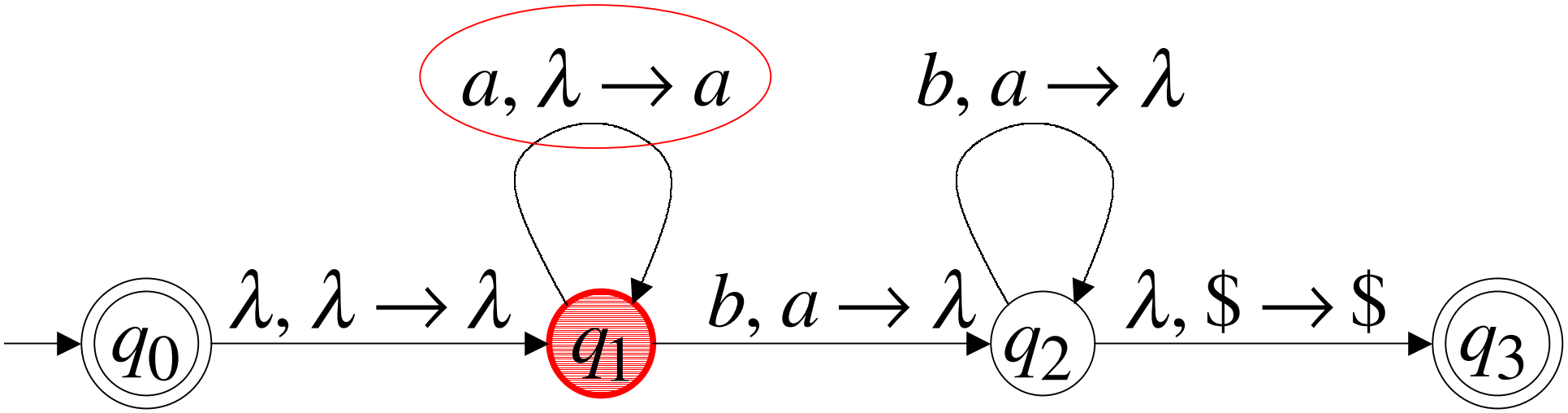
Time 3

Input



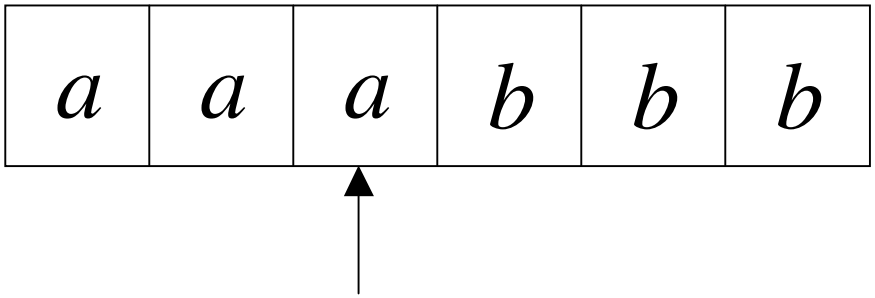


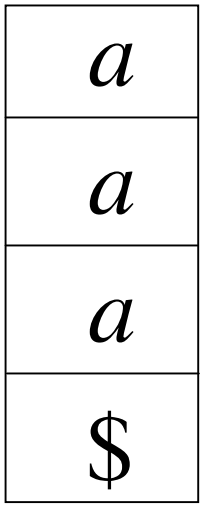
Stack



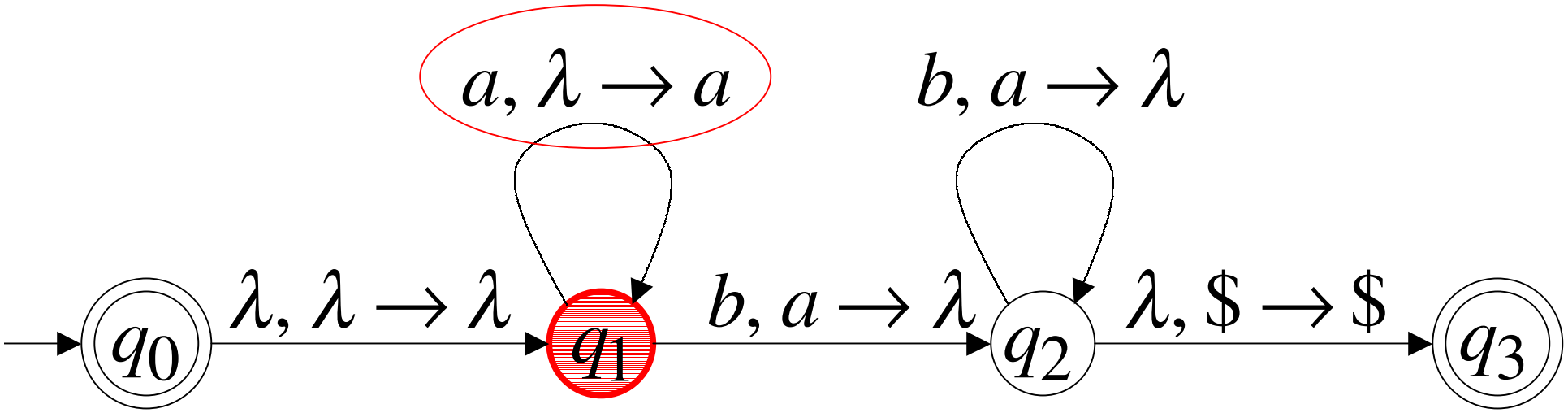
Time 4

Input



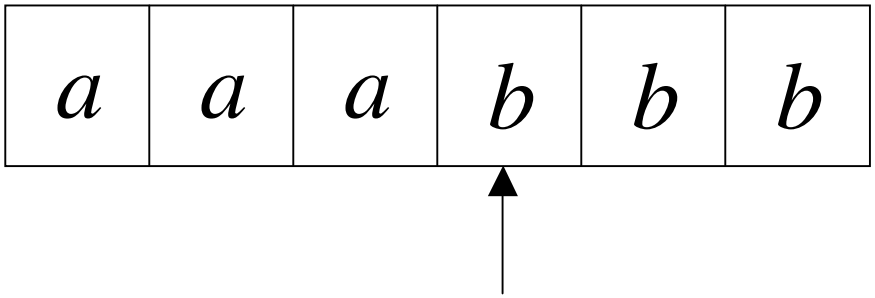


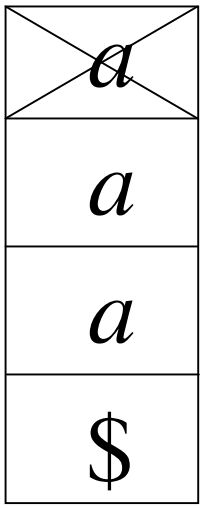
Stack



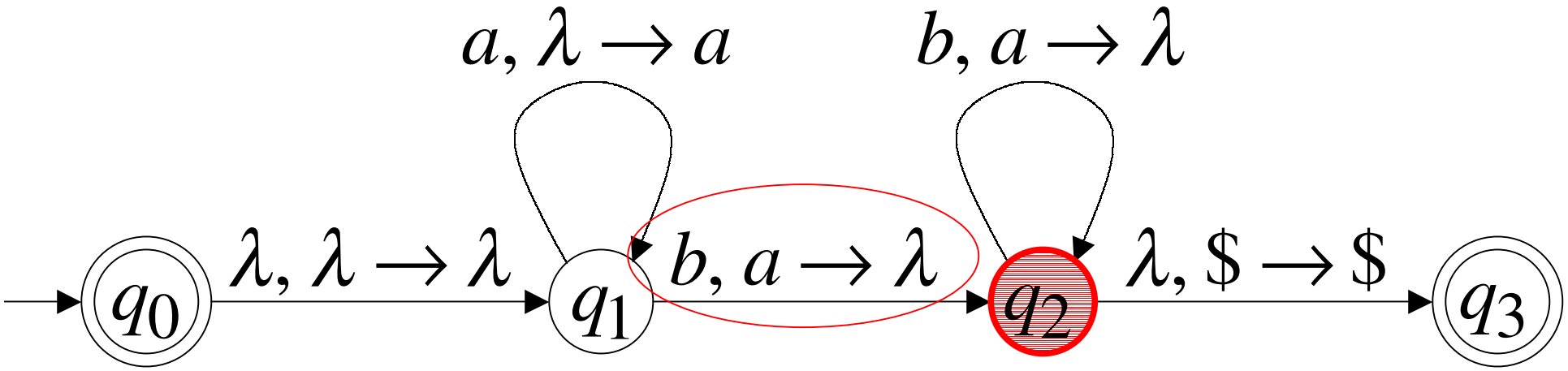
Time 5

Input



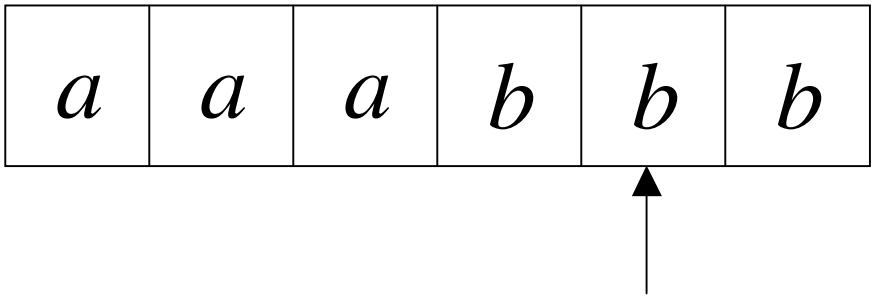


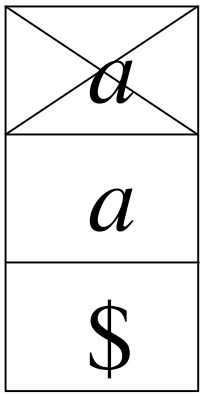
Stack



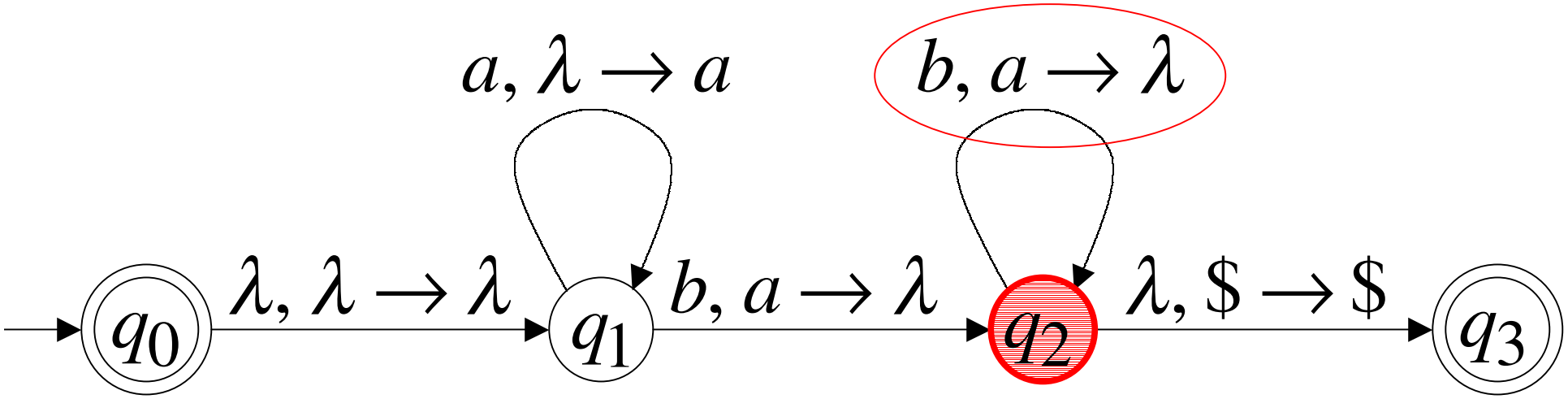
Time 6

Input



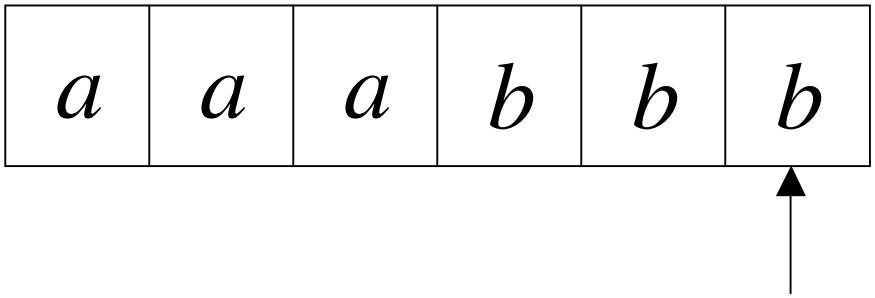


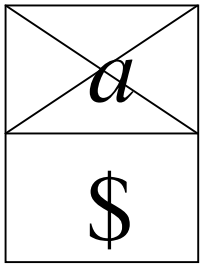
Stack



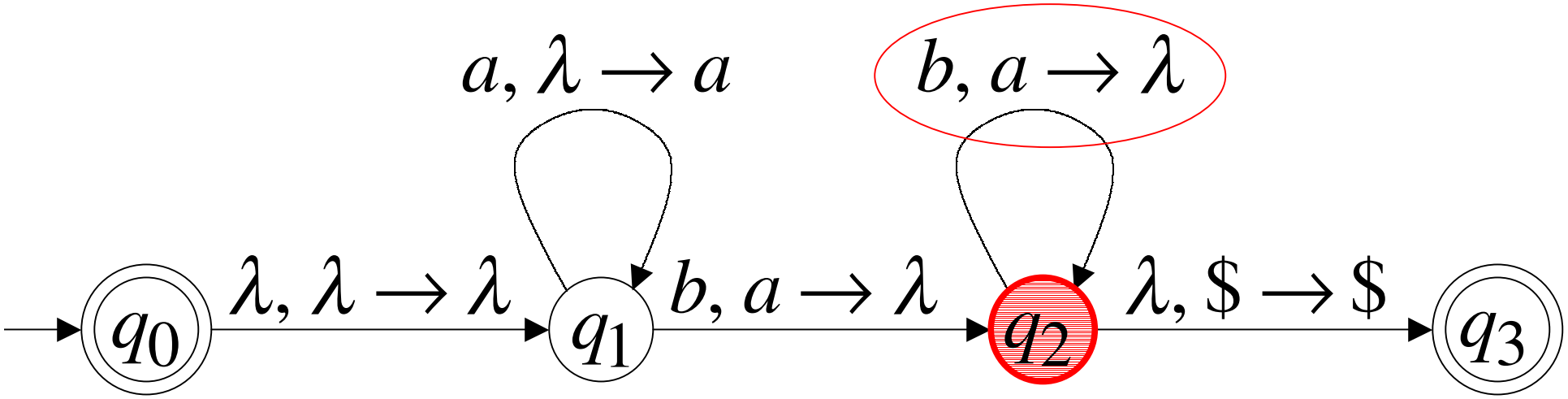
Time 7

Input



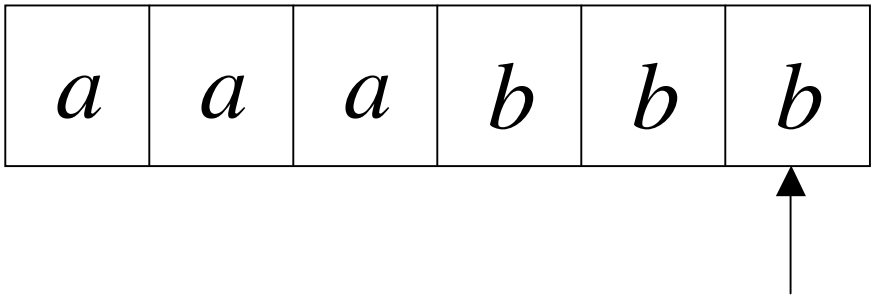


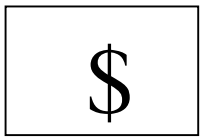
Stack



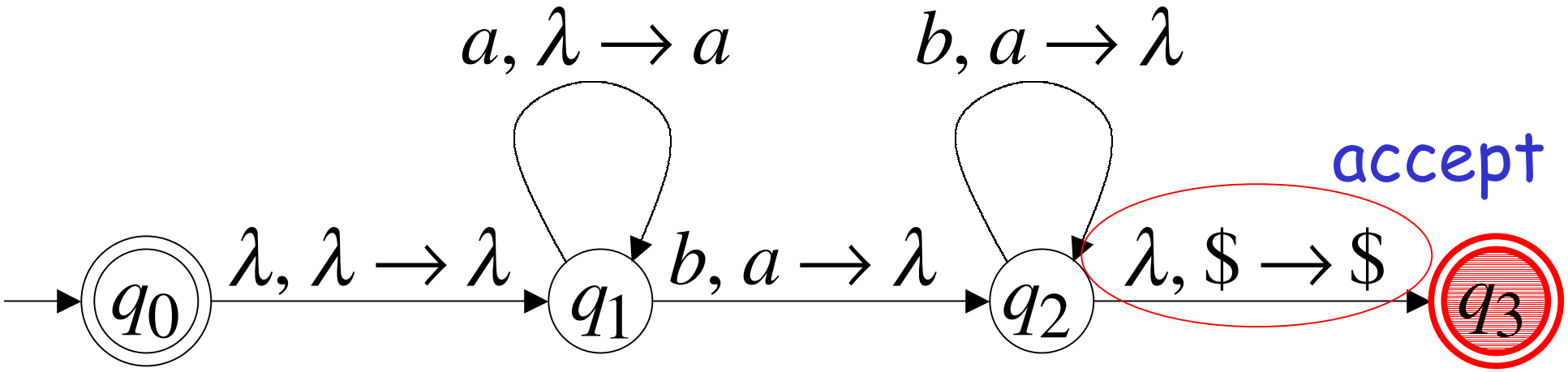
Time 8

Input





Stack

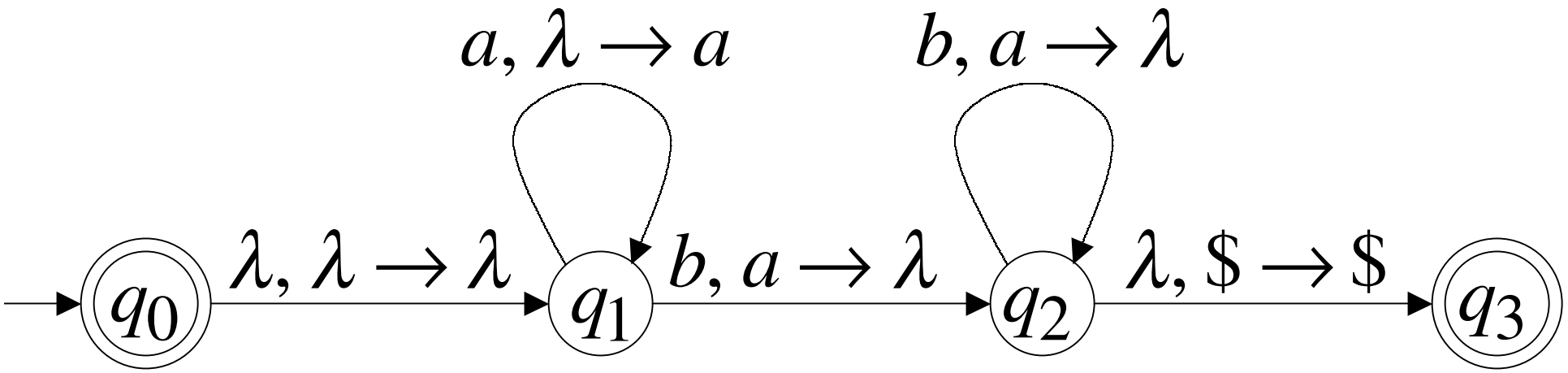


A string is accepted if:

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

We do not care about the stack contents

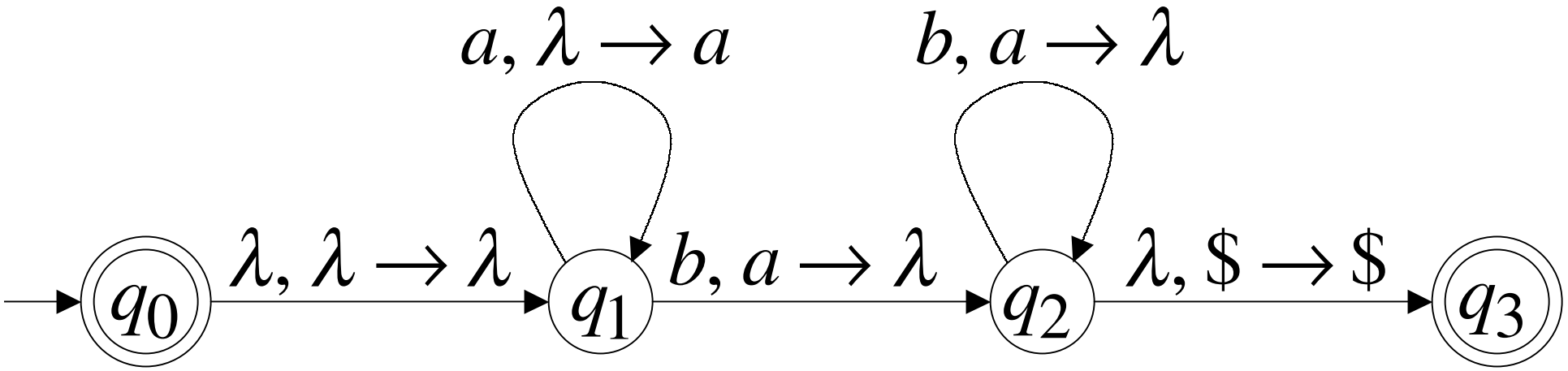
The input string $aaabbb$
is accepted by the NPDA:



In general,

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

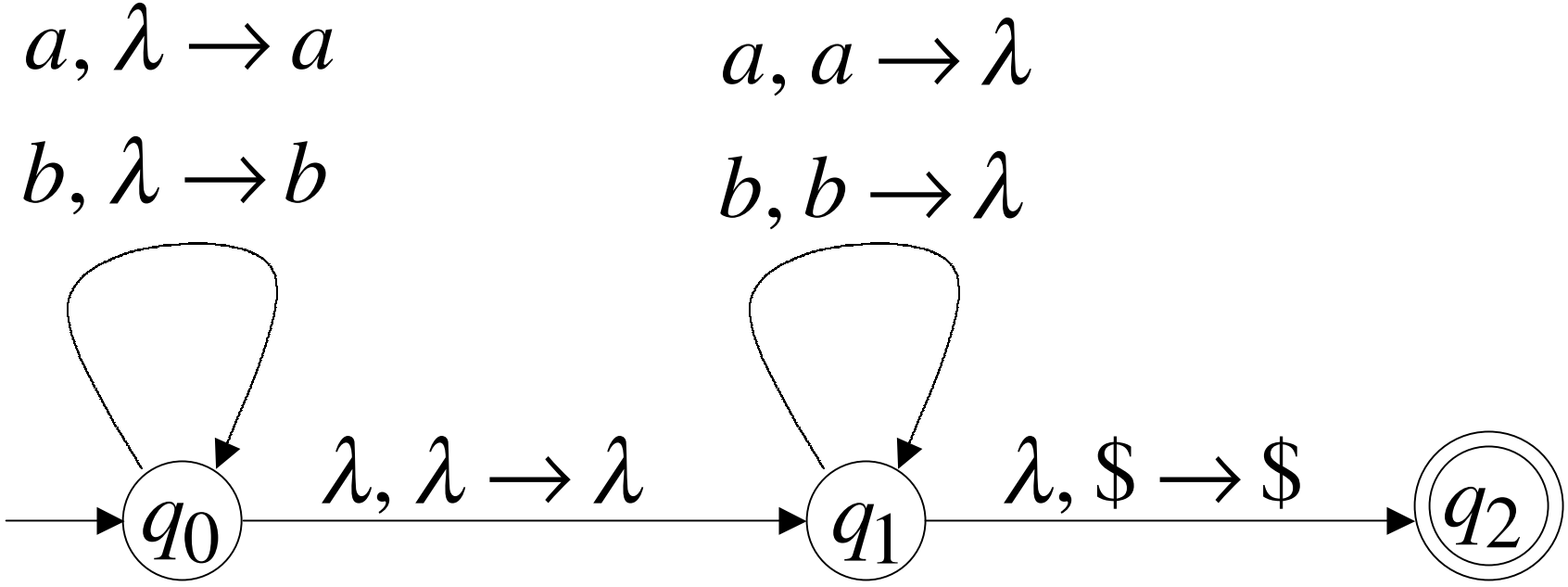
is the language accepted by the NPDA:



Another NPDA example

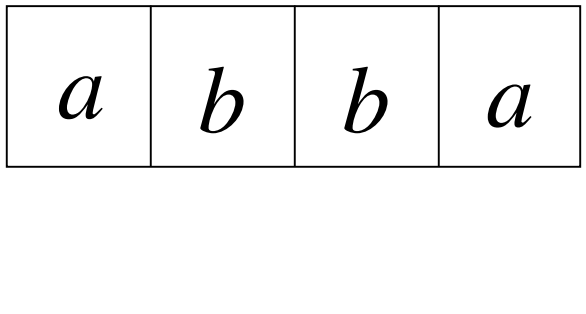
NPDA M

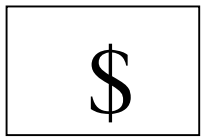
$$L(M) = \{ww^R\}$$



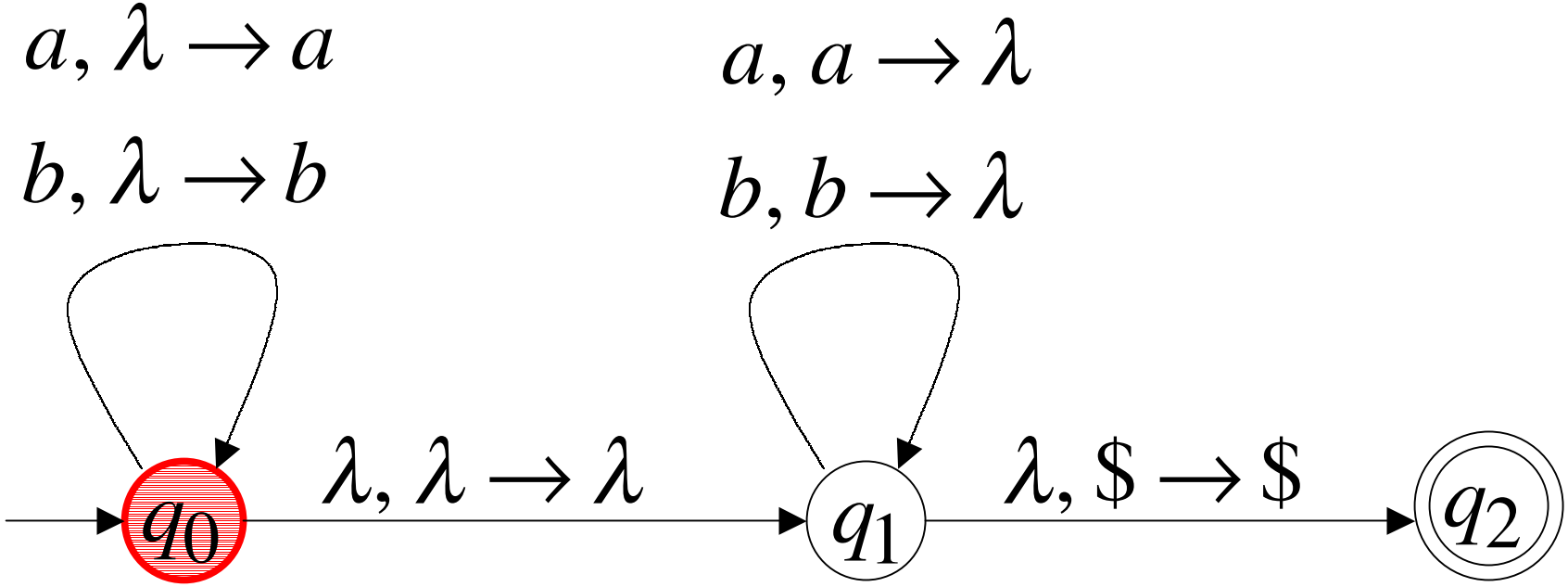
Execution Example: Time 0

Input



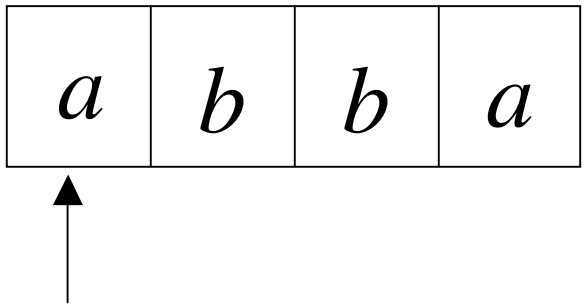


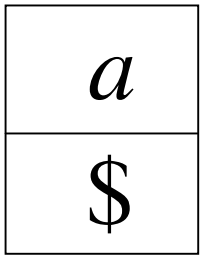
Stack



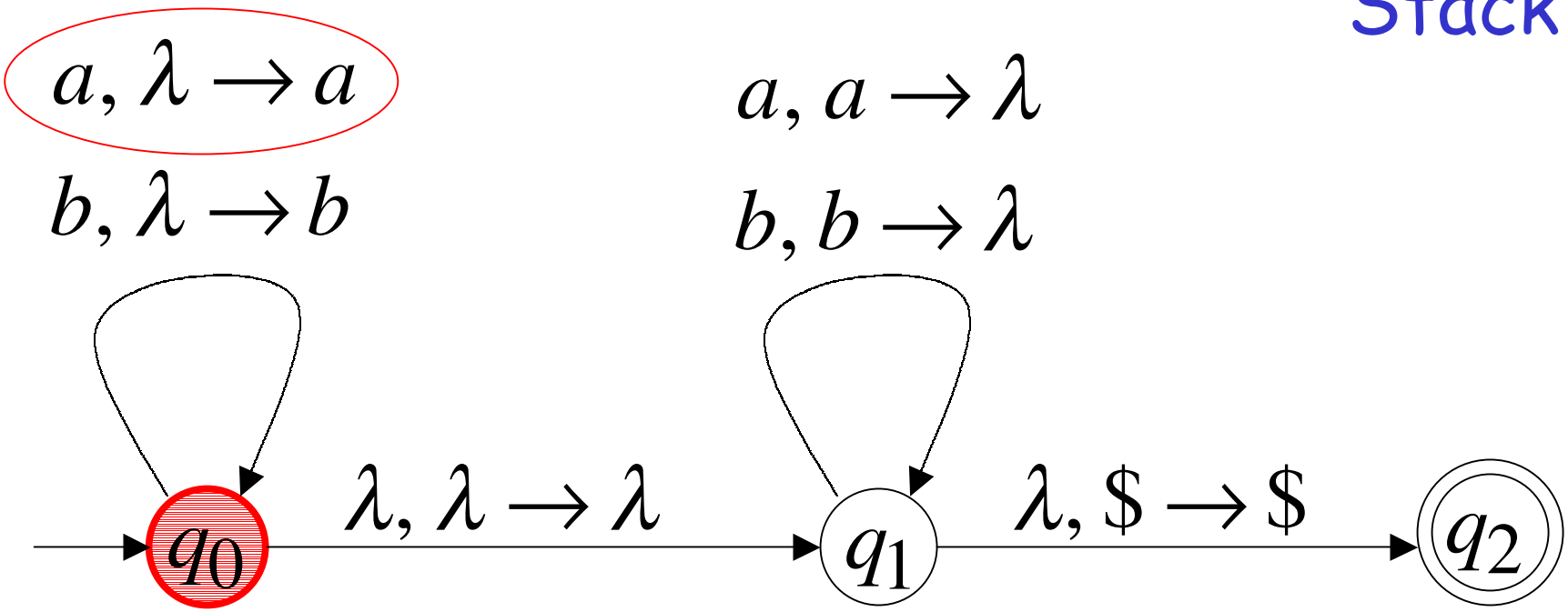
Time 1

Input



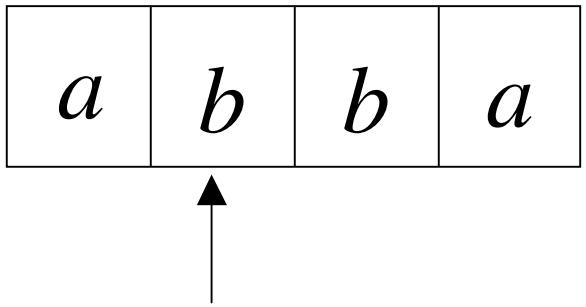


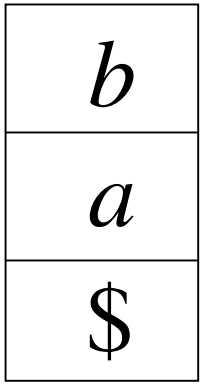
Stack



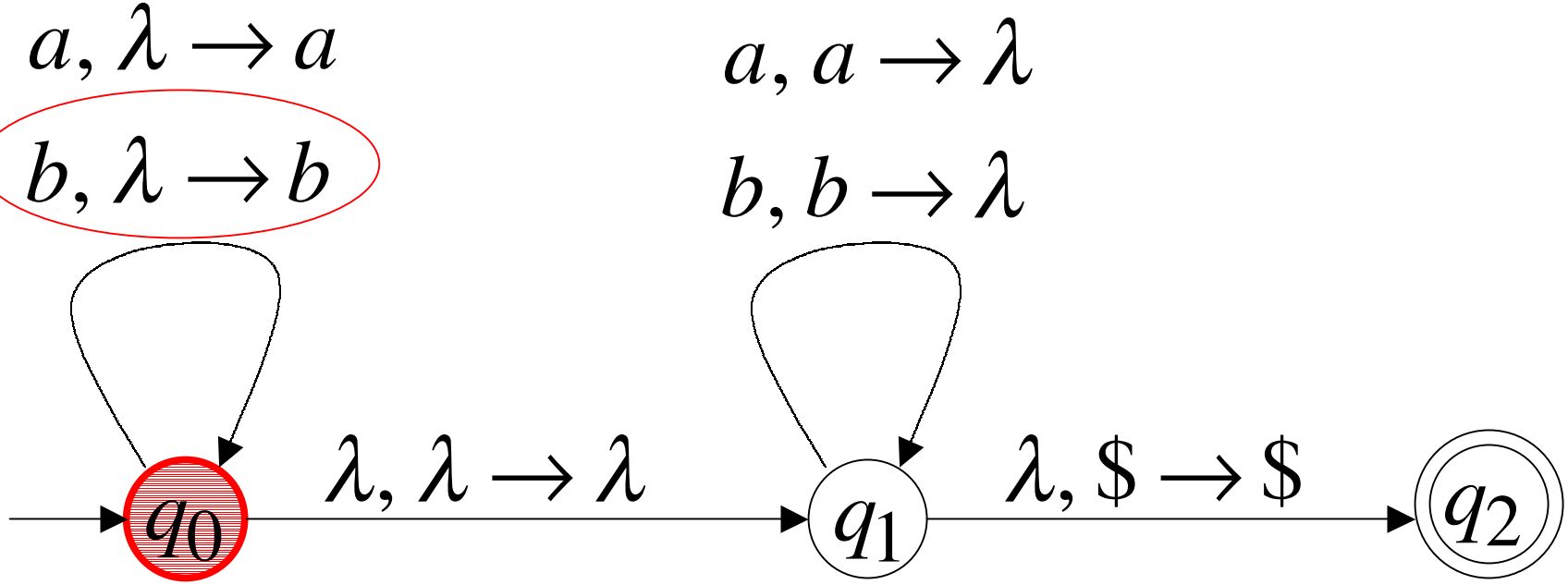
Time 2

Input



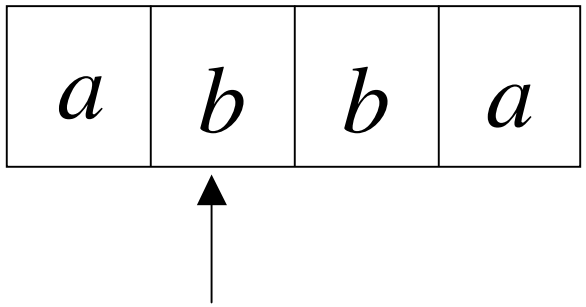


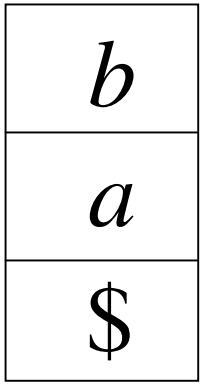
Stack



Time 3

Input





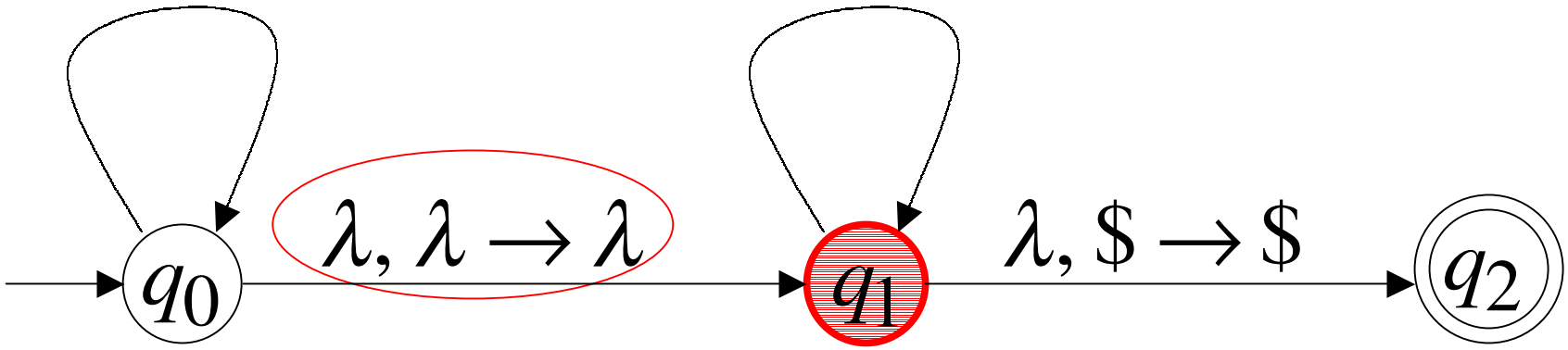
Stack

$a, \lambda \to a$

$b, \lambda \to b$

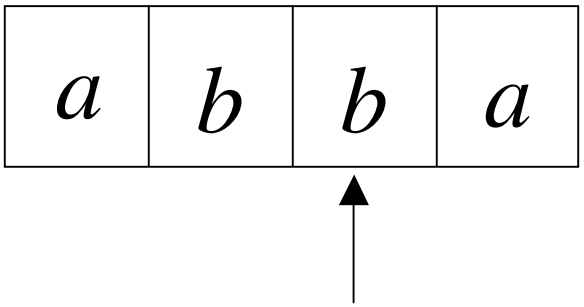
$a, a \to \lambda$

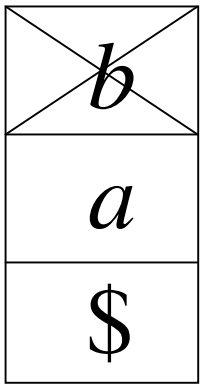
$b, b \to \lambda$



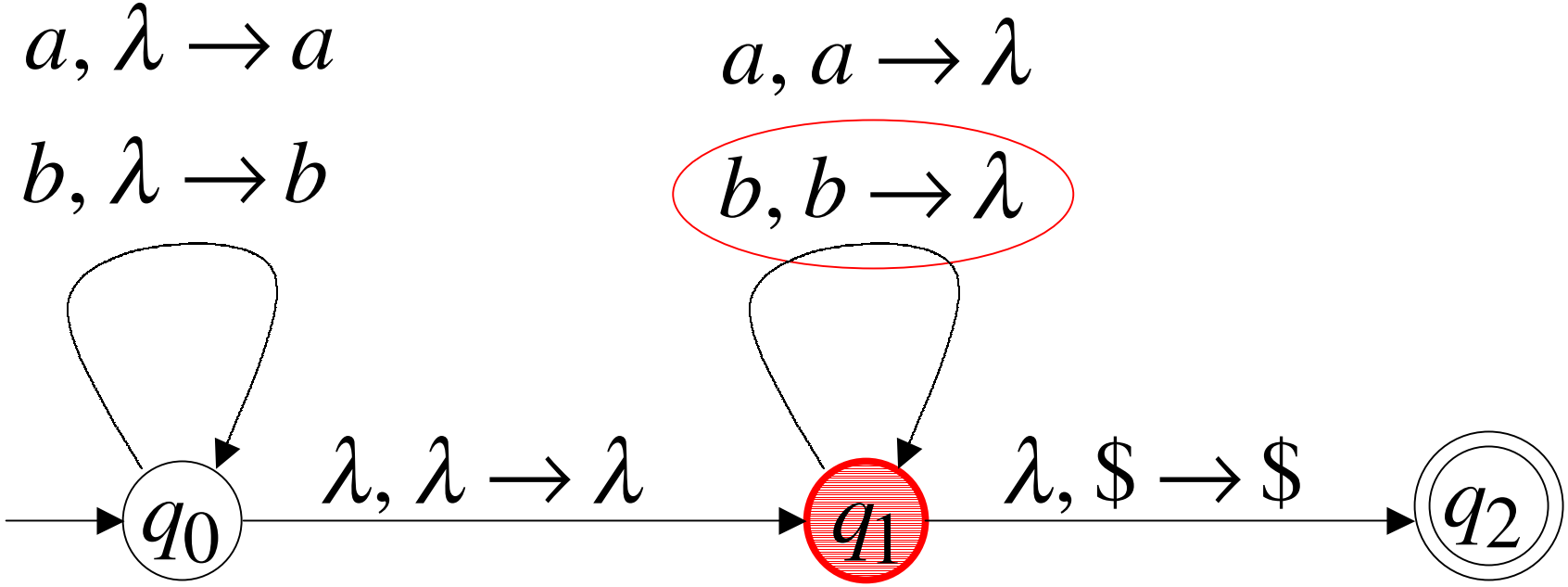
Time 4

Input



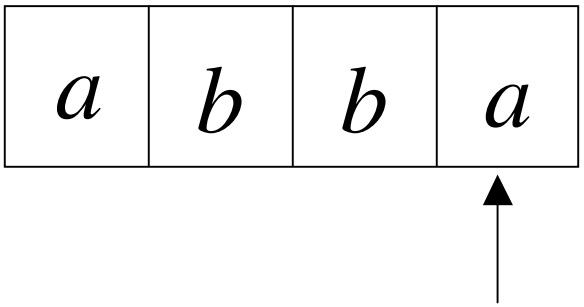


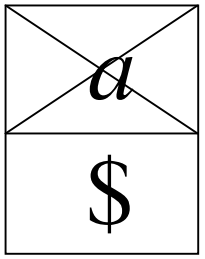
Stack



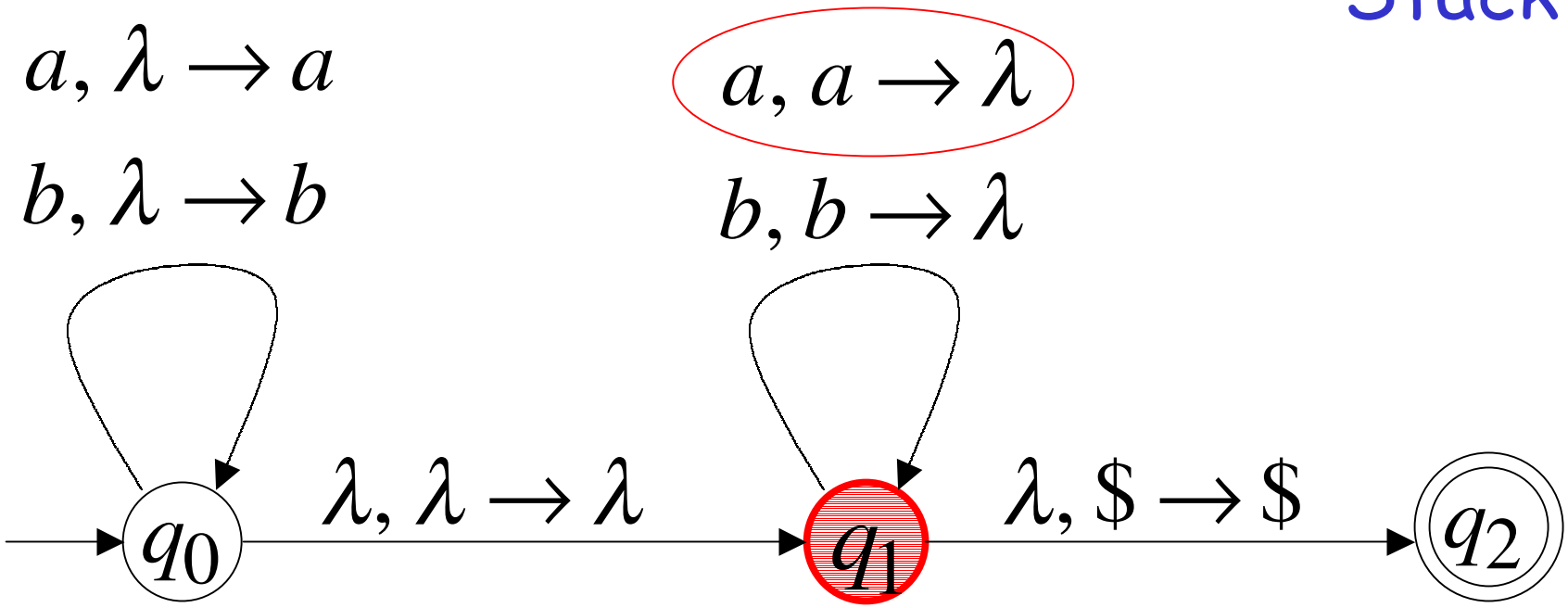
Time 5

Input



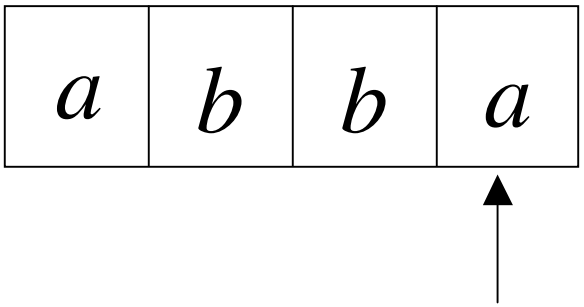


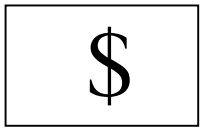
Stack



Time 6

Input





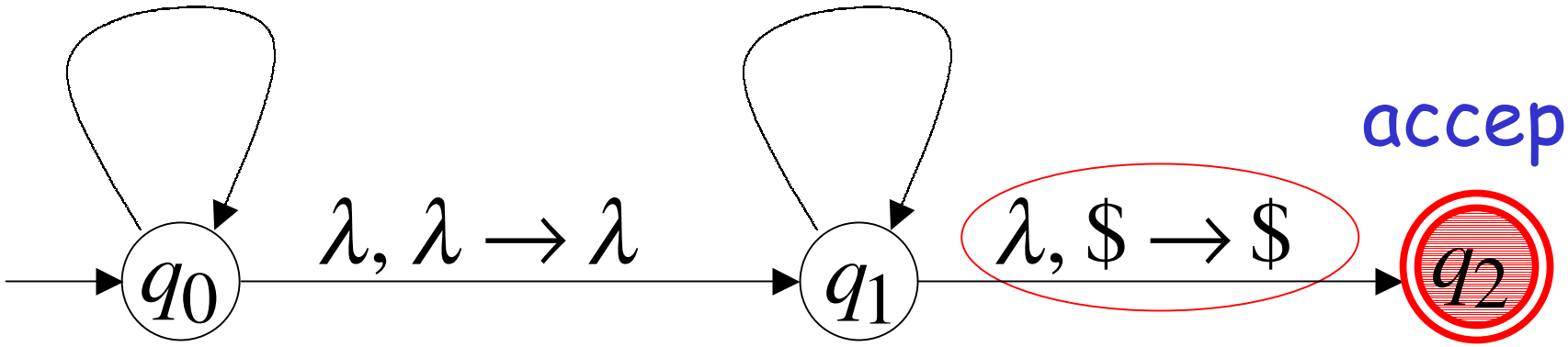
Stack

$a, \lambda \to a$

$b, \lambda \to b$

$a, a \to \lambda$

$b, b \to \lambda$



accept