Context-free Languages

Formal Languages and Abstract Machines

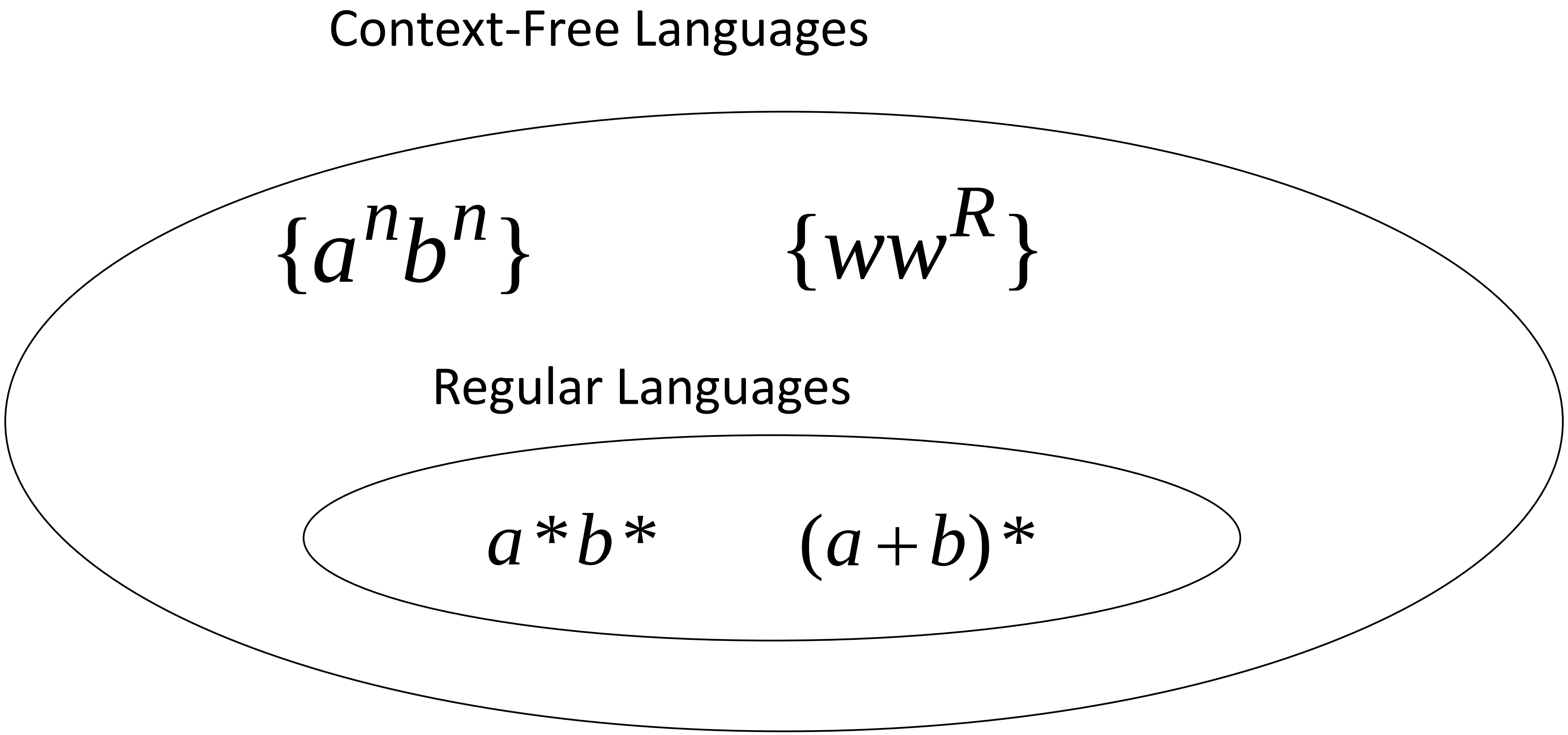
Week 08

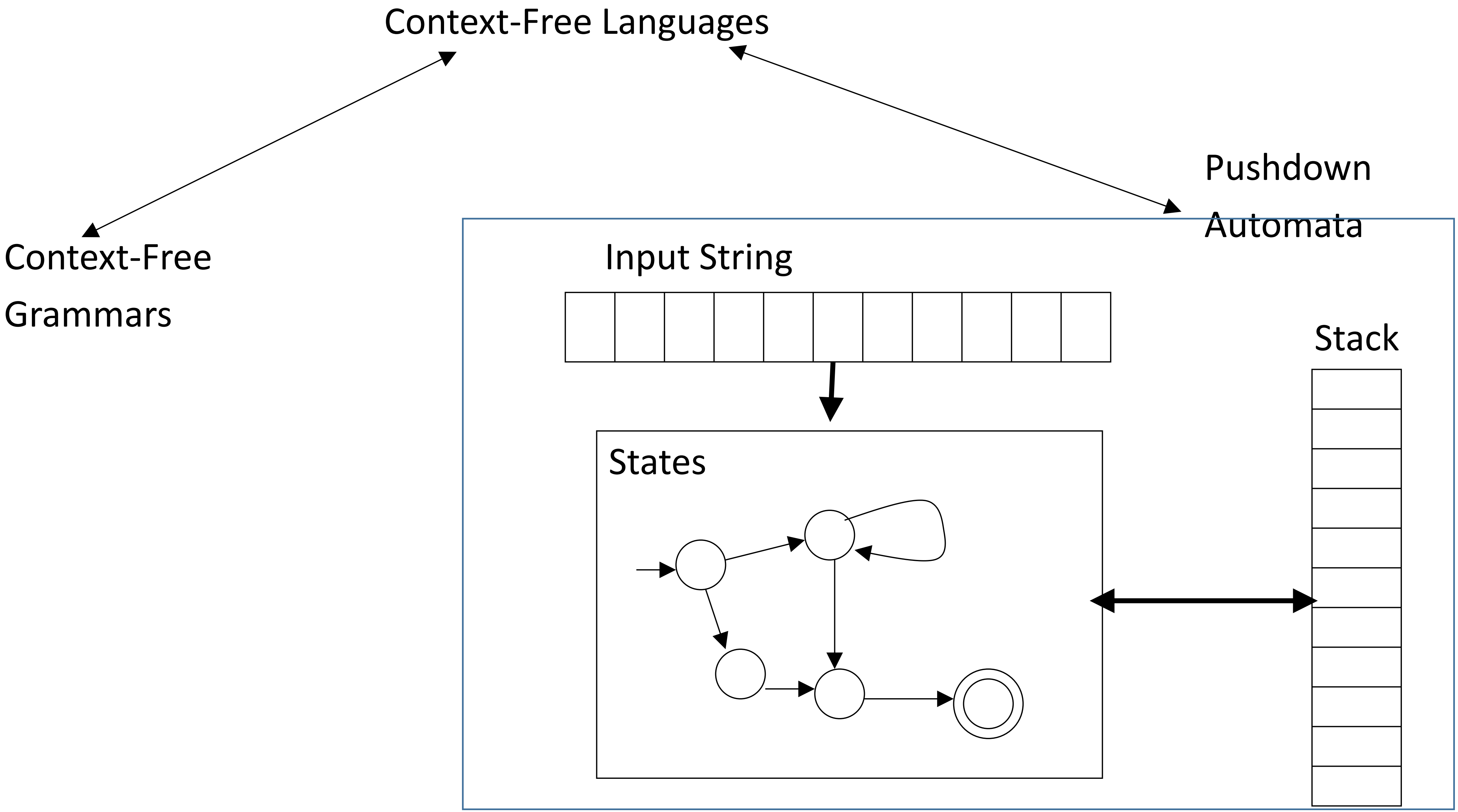
Baris E. Suzek, PhD

Outline

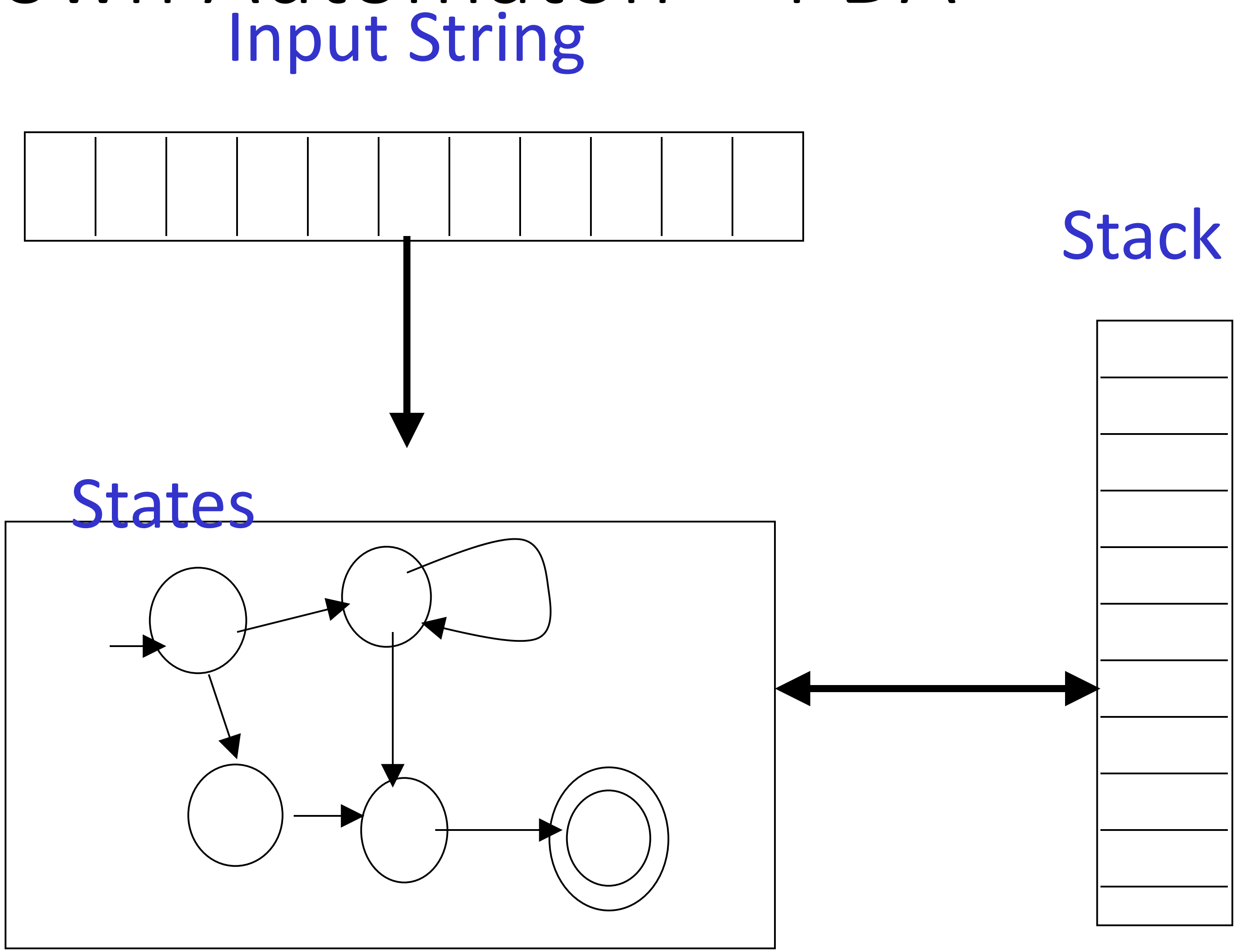
- Last week
- Conversions around Context-free Languages
- Deterministic PDA(DPDA)
- Turing Machines
- Review

Context-Free and Regular Languages

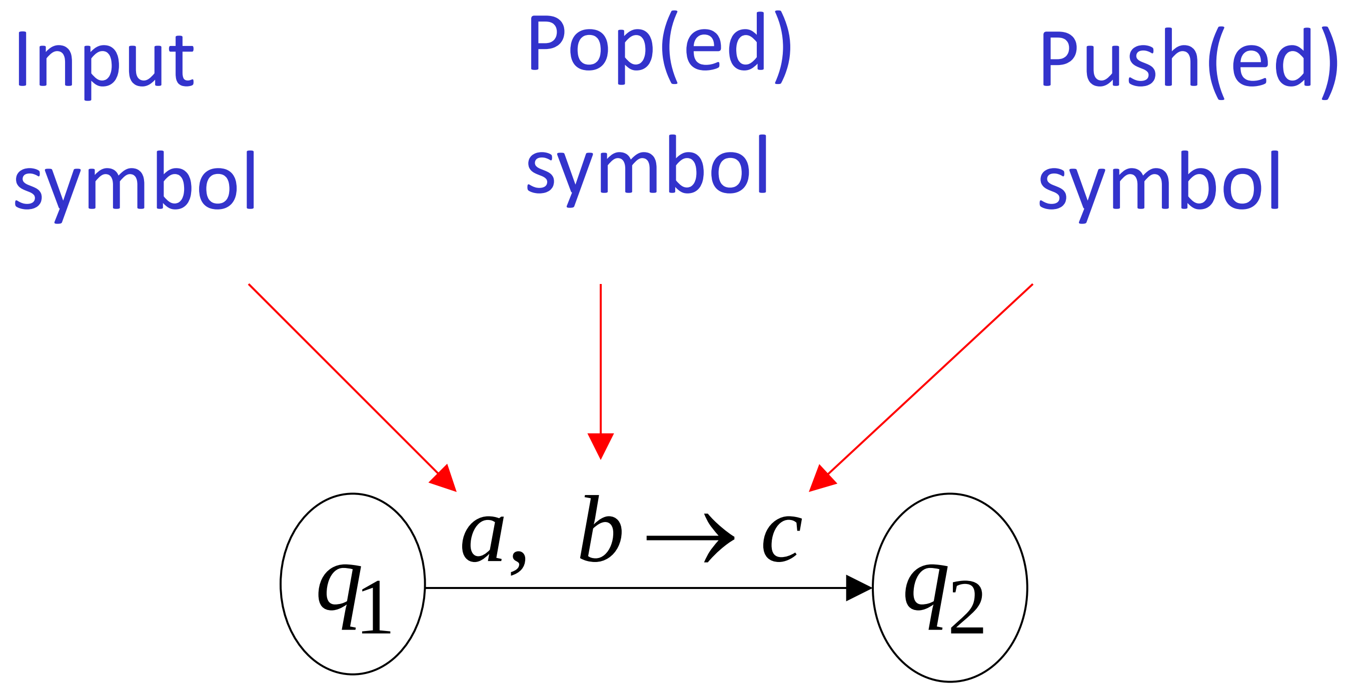



Context-Free Languages
Context-Free
Grammars
Pushdown
Automata
Input String
States
Stack

Pushdown Automaton -- PDA

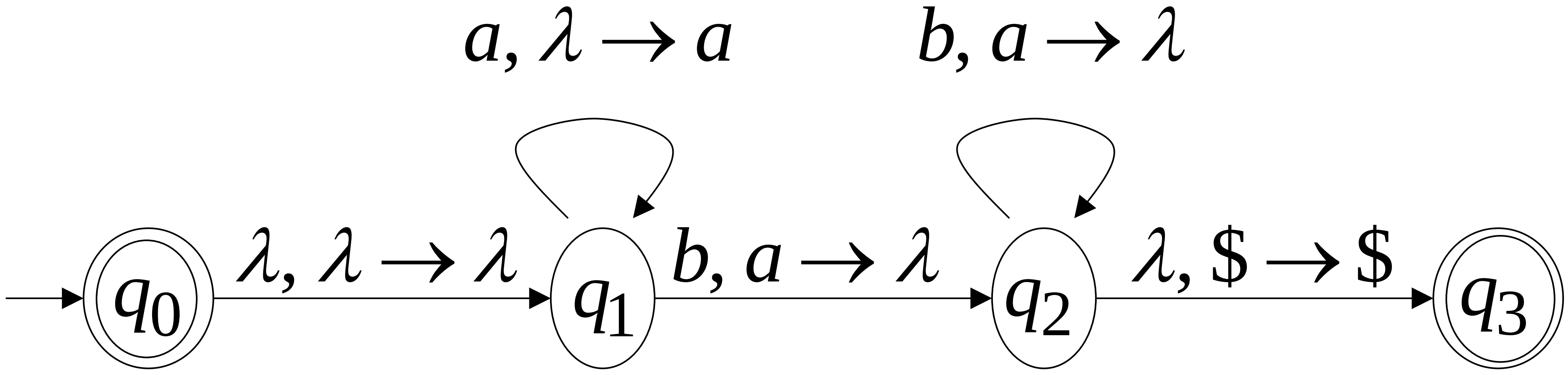


The States


Input symbol
Pop(ed) symbol
Push(ed) symbol
$a, \; b \rightarrow c$
q_1
q_2

NPDA: Non-Deterministic PDA

Example:



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

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

A string is rejected

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

Formal Definition

Non-Deterministic Pushdown Automaton

NPDA

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

States

Input alphabet

Stack alphabet

Transition function

Initial state

Stack start symbol

Final states

Instantaneous Description

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

Current state

Remaining input

Current stack contents

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

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

Star Operation

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

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

Intersection

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

L_1 is context free

L_2 is context free

$\Rightarrow$ $L_1 \cap L_2$ **not** necessarily context-free

Complement

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

L is context free $\Rightarrow$ $\overline{L}$ **not** necessarily context-free

Outline

- Last week
- Conversions around Context-free Languages
- Deterministic PDA(DPDA)
- Turing Machines
- Review

NPDAs Accept
Context-Free Languages

Theorem:

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

Proof - Step 1:

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

Convert any context-free grammar G

to a NPDA M with: $L(G) = L(M)$

Proof - Step 2:

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

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

Proof - step 1

Converting
Context-Free Grammars
to
NPDAs

We will convert any context-free grammar G

to an NPDA automaton M

Such that:

M Simulates leftmost derivations of G

Leftmost derivation

$G:$

$$S \Rightarrow \cdots\cdots \Rightarrow \underbrace{\sigma_1\sigma_2\cdots\sigma_k}_{\text{Input processed}}\underbrace{V_1V_2\cdots V_m}_{\text{Stack contents}} \Rightarrow \cdots$$

leftmost variable

$M:$ Simulation of derivation

Input

σ_1	σ_2	$\cdots$	σ_k	σ_{k+1}	$\cdots$

Stack

V_1
V_2
$\vdots$
V_m
$\$$

Leftmost derivation

$G:$

$$S \Rightarrow \cdots\cdots\cdots \Rightarrow \sigma_1\sigma_2\cdots\sigma_n$$

string of terminals

$M:$ Simulation of derivation

Stack

Input

σ_1	σ_2	$\cdots\cdots$	σ_n

$\$$

end of input is reached

An example grammar:

$$S \rightarrow aSTb$$

$$S \rightarrow b$$

$$T \rightarrow Ta$$

$$T \rightarrow \lambda$$

What is the equivalent NPDA?

Grammar:

$$S \to aSTb$$

$$S \to b$$

$$T \to Ta$$

$$T \to \lambda$$

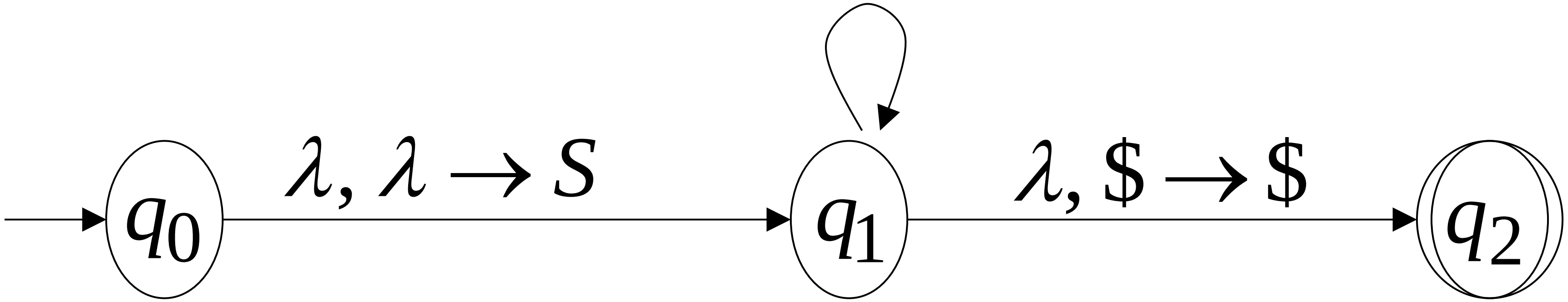
NPDA:

$$\lambda, S \to aSTb$$

$$\lambda, S \to b$$

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

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



Grammar:

$$S \to aSTb$$

$$S \to b$$

$$T \to Ta$$

$$T \to \lambda$$

A leftmost derivation:

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

Derivation:

Input

a	b	a	b

Time 0

$$\lambda, S \to aSTb$$

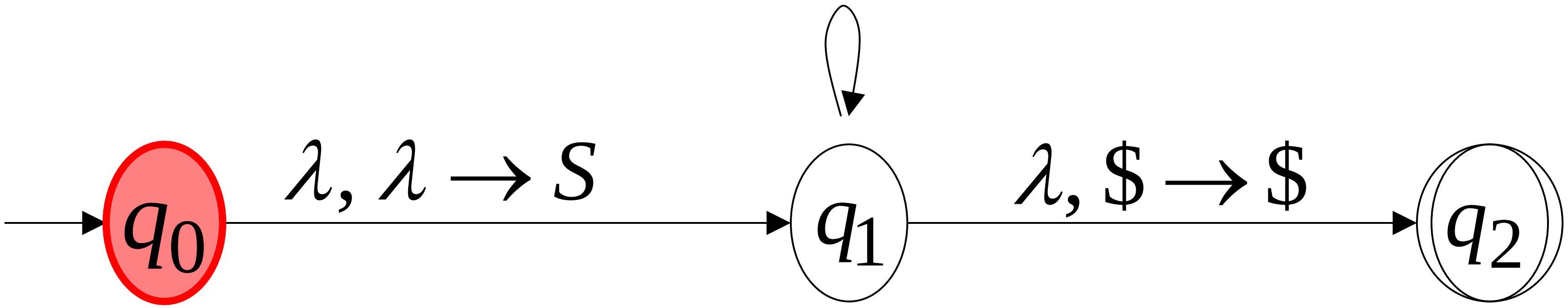
$$\lambda, S \to b$$

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

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

$

Stack



Derivation: S

Input

a	b	a	b

Time 0

$\lambda, S \rightarrow aSTb$

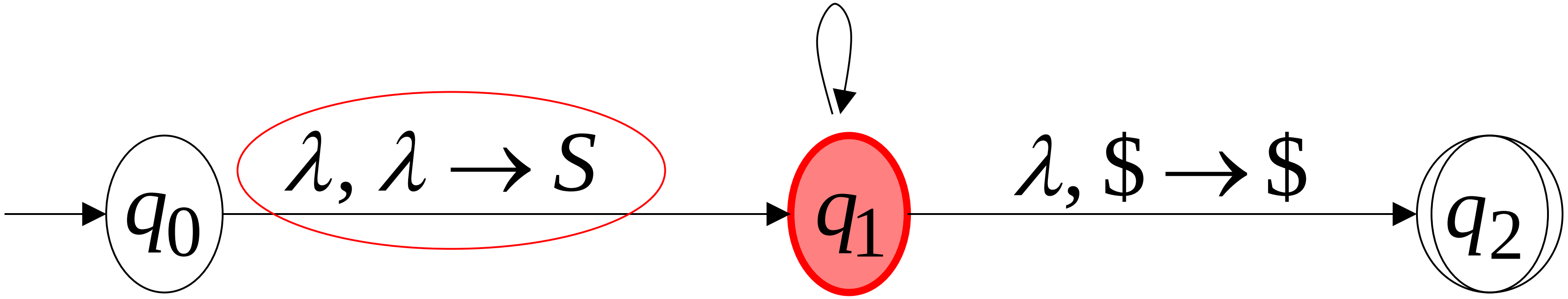
$\lambda, S \rightarrow b$

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

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

S
$\$$

Stack



Derivation: $S \Rightarrow aSTb$

Input

a	b	a	b

Time 1

$\lambda, S \rightarrow aSTb$

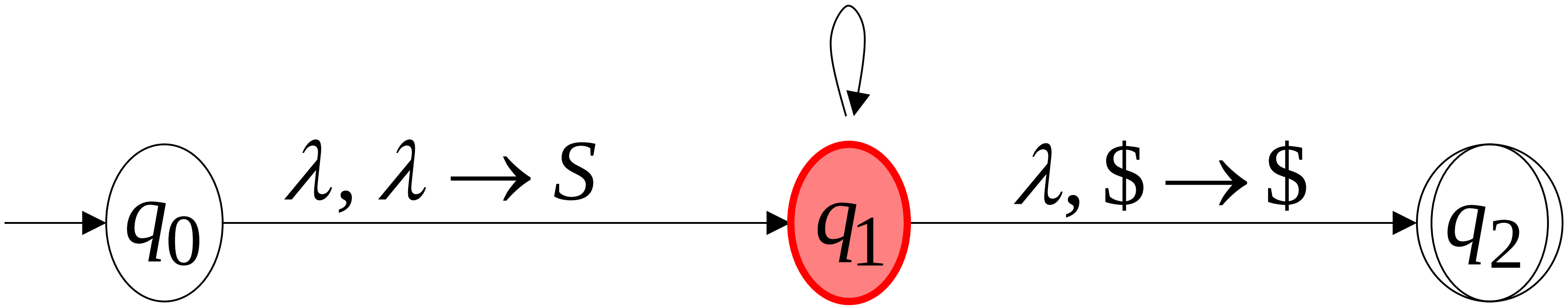
$\lambda, S \rightarrow b$

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

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

a
S
T
b
$\$$

Stack



Derivation: $S \Rightarrow aSTb$

Input

a	b	a	b

Time 2

$\lambda, S \rightarrow aSTb$

$\lambda, S \rightarrow b$

$\lambda, T \rightarrow Ta$

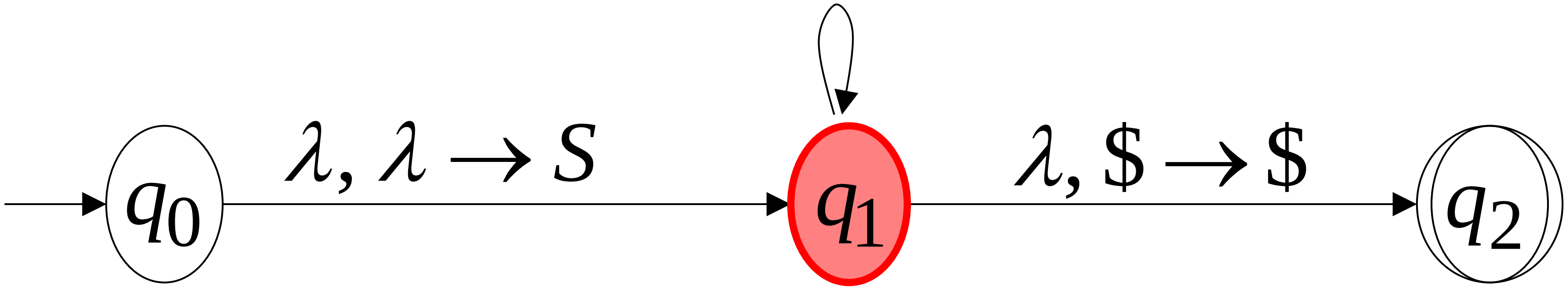
$\lambda, T \rightarrow \lambda$

$a, a \rightarrow \lambda$

$b, b \rightarrow \lambda$

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

Stack



Derivation: $S \Rightarrow aSTb \Rightarrow abTb$

Input

a	b	a	b

Time 3

$\lambda, S \to aSTb$

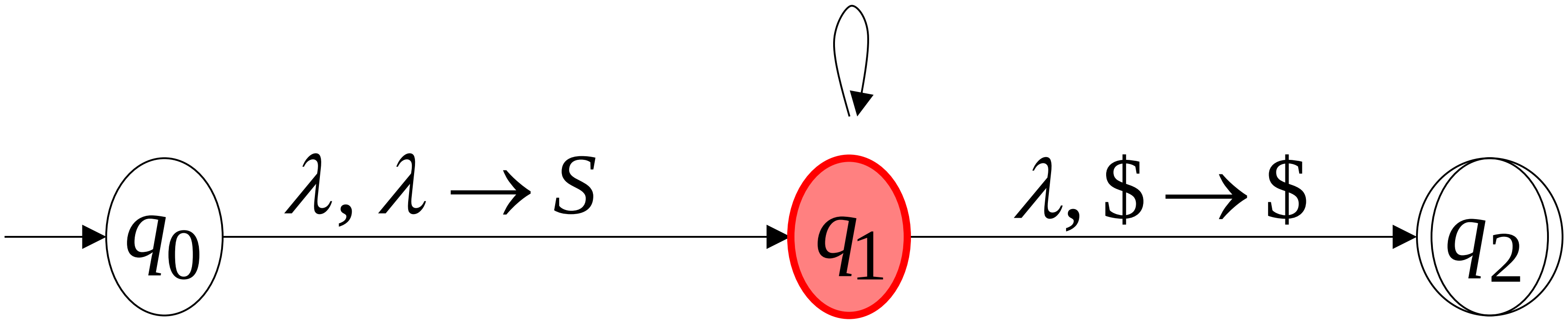
$\lambda, S \to b$

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

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

Stack

b
T
b
$\$$



Derivation: $S \Rightarrow aSTb \Rightarrow abTb$

Input

a	b	a	b

Time 4

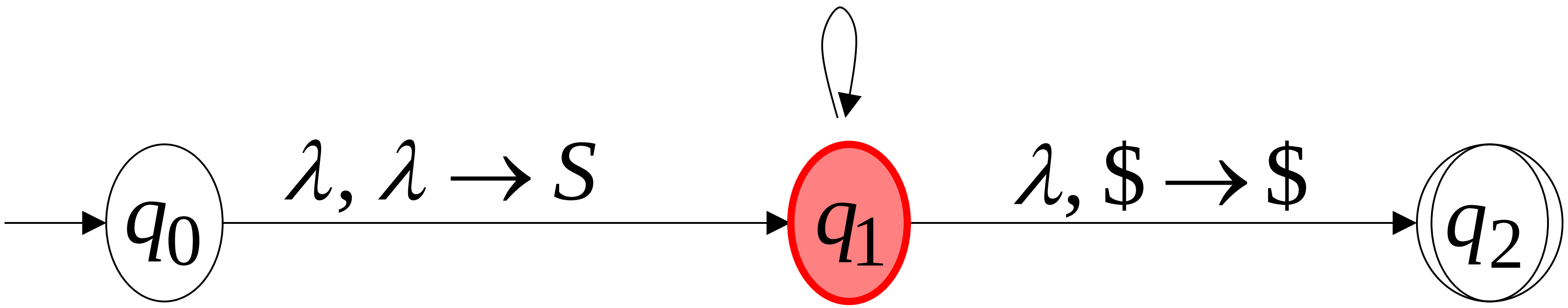
$\lambda, S \rightarrow aSTb$

$\lambda, S \rightarrow b$

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

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

Stack:
- $\cancel{b}$
- T
- b
- $\$$



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

Input

a	b	a	b

Time 5

$\lambda, S \rightarrow aSTb$

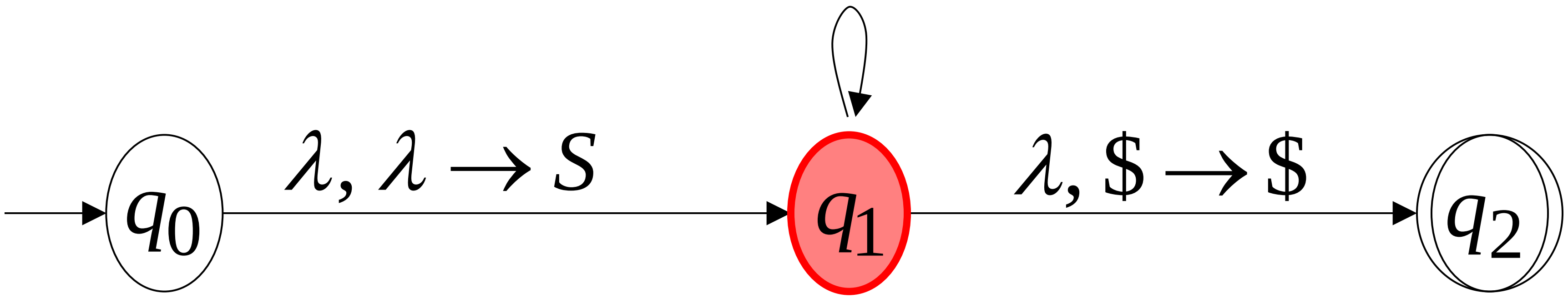
$\lambda, S \rightarrow b$

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

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

T
a
b
$\$$

Stack



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

Input

a	b	a	b

Time 6

$\lambda, S \rightarrow aSTb$

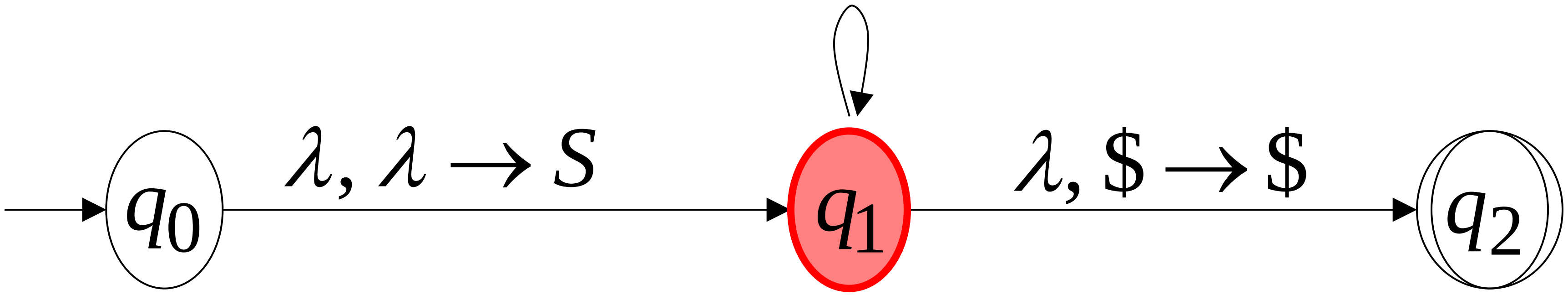
$\lambda, S \rightarrow b$

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

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

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

Stack



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

Input

a	b	a	b

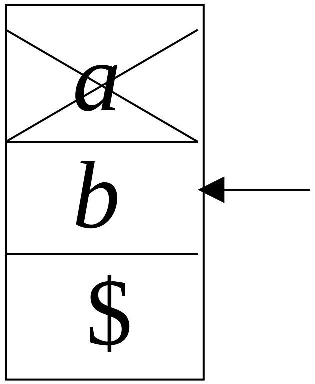
Time 7

$\lambda, S \to aSTb$

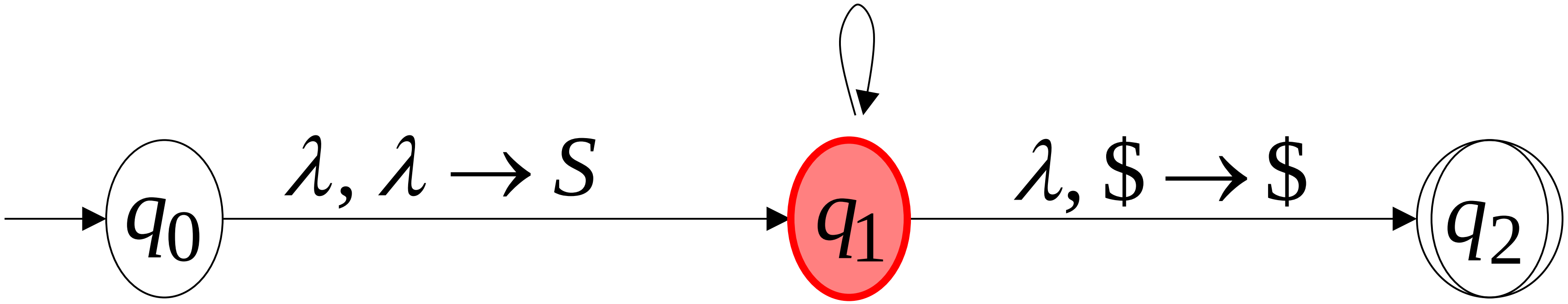
$\lambda, S \to b$

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

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



Stack



Derivation:

Input

a	b	a	b

Time 8

$\lambda, S \rightarrow aSTb$

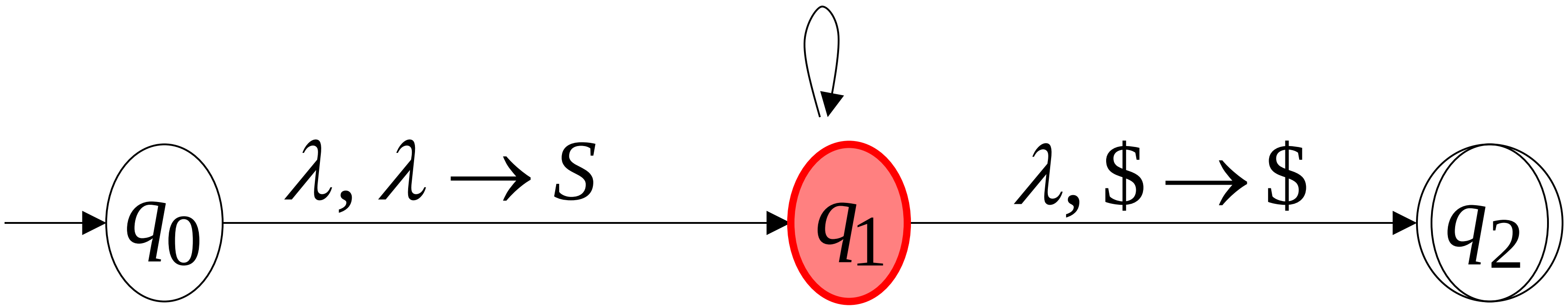
$\lambda, S \rightarrow b$

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

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

$\cancel{b}$
$\$$

Stack



Derivation:

Input

a	b	a	b

Time 9

$$\lambda, S \rightarrow aSTb$$

$$\lambda, S \rightarrow b$$

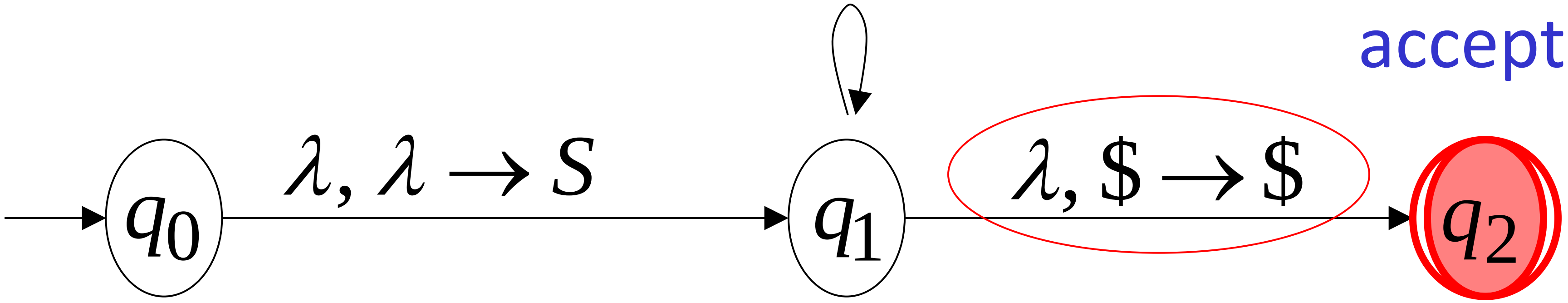
$$\lambda, T \rightarrow Ta \qquad a, a \rightarrow \lambda$$

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

$\$$

Stack

accept



In general:

Given any grammar G

We can construct a NPDA M

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

Constructing NPDA M from grammar : G

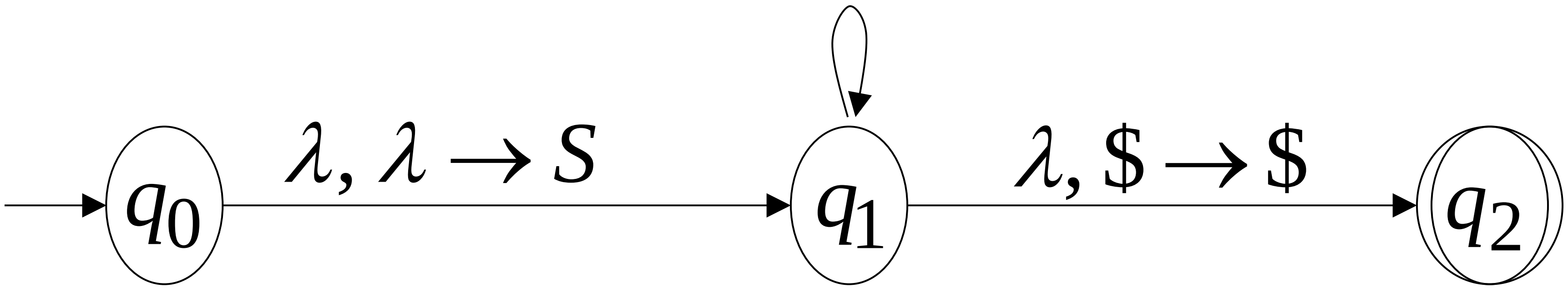
For any production

$A \to w$

For any terminal

a

$\lambda,\, A \to w$ $\qquad$ $a,\, a \to \lambda$



Grammar G generates string w

if and only if

NPDA M accepts w

$$\Downarrow$$

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

Therefore:

For any context-free language
there is a NPDA
that accepts the same language

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

Proof - step 2

Converting
NPDAs
to
Context-Free Grammars
is possible

For any NPDA M

we will construct

a context-free grammar G with

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

Intuition: The grammar simulates the machine

A derivation in Grammar : G

terminals variables

$$\underbrace{S \Rightarrow \cdots \Rightarrow abc\ldots}_{\text{Input processed}} \underbrace{ABC\ldots}_{\text{Stack contents}} \Rightarrow \ldots \Rightarrow abc\ldots$$

Current configuration in NPDA M

Some Necessary Modifications

Modify (if necessary) the NPDA so that:

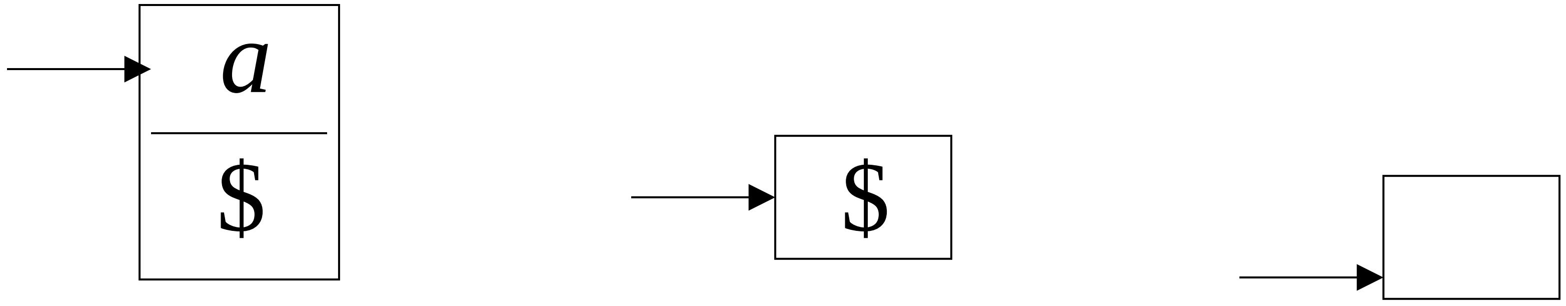
1) The stack is never empty

2) It has a single final state
and empties the stack when it accepts a string

3) Has transitions in a special form

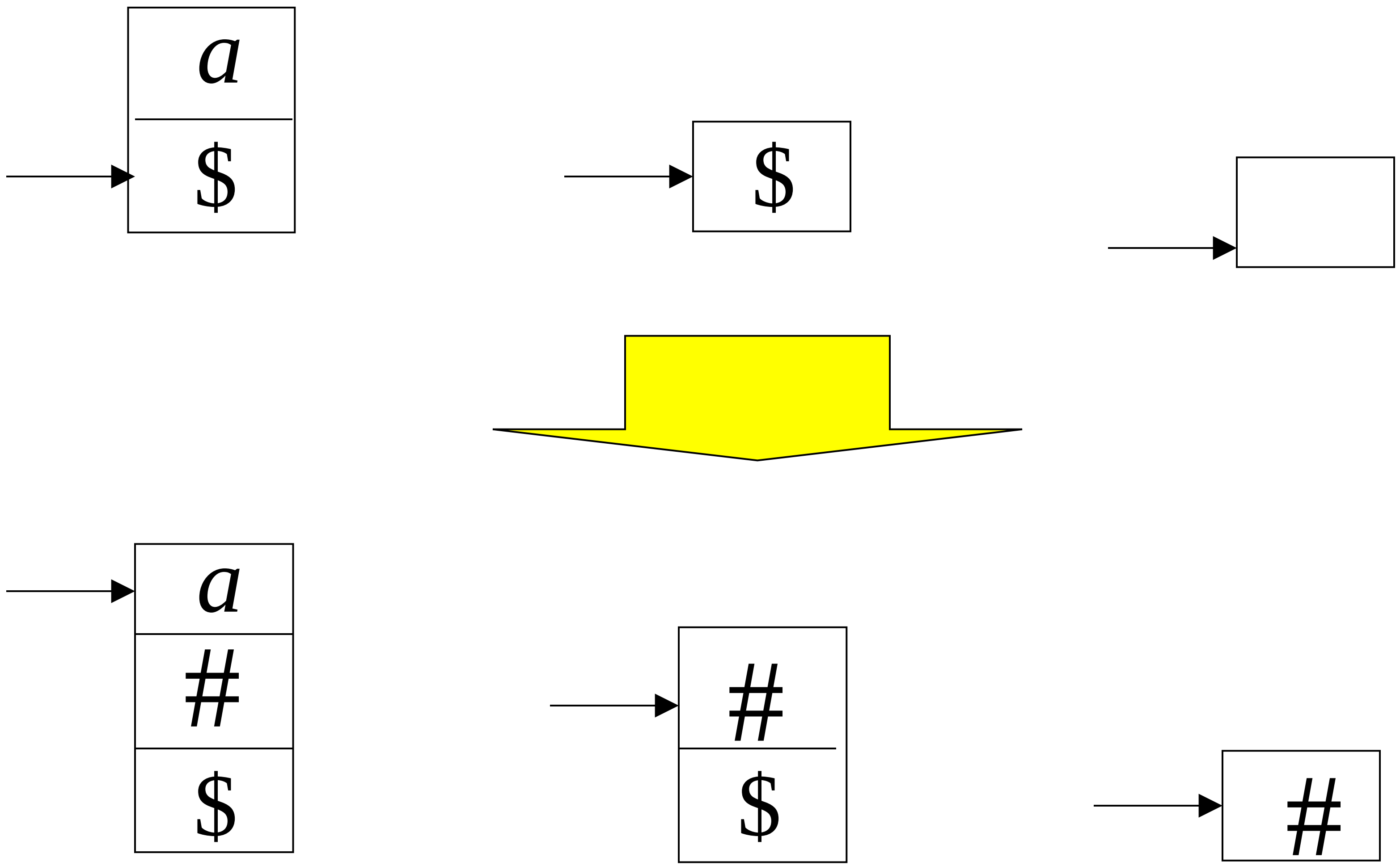
1) Modify the NPDA so that the stack is never empty

Stack

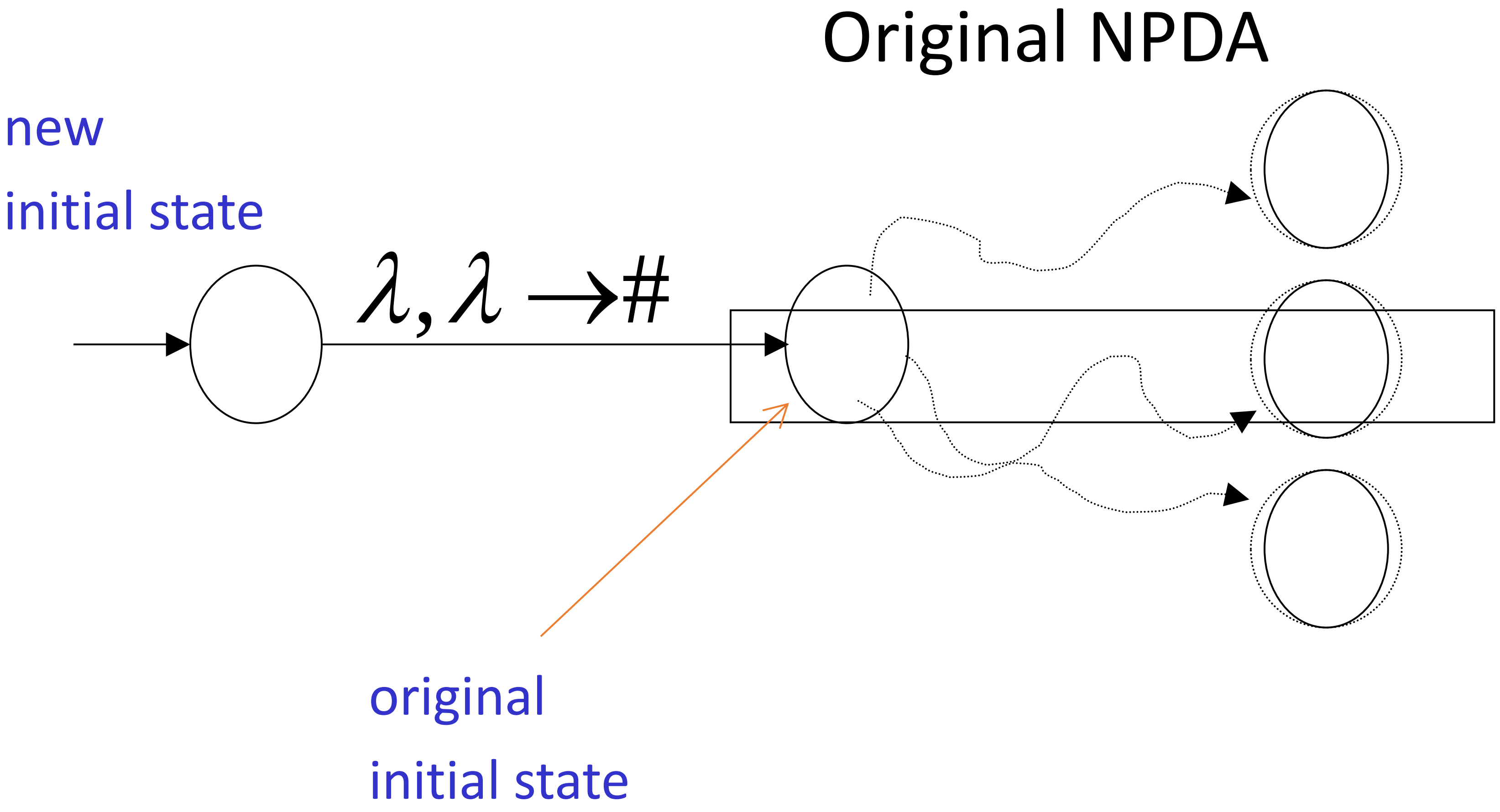


OK OK NOT OK

Introduce the new symbol $\#$ to denote the bottom of the stack



At the beginning push $\#$ into the stack



In transitions:

replace every instance of $\$$ with $\#$

Example:

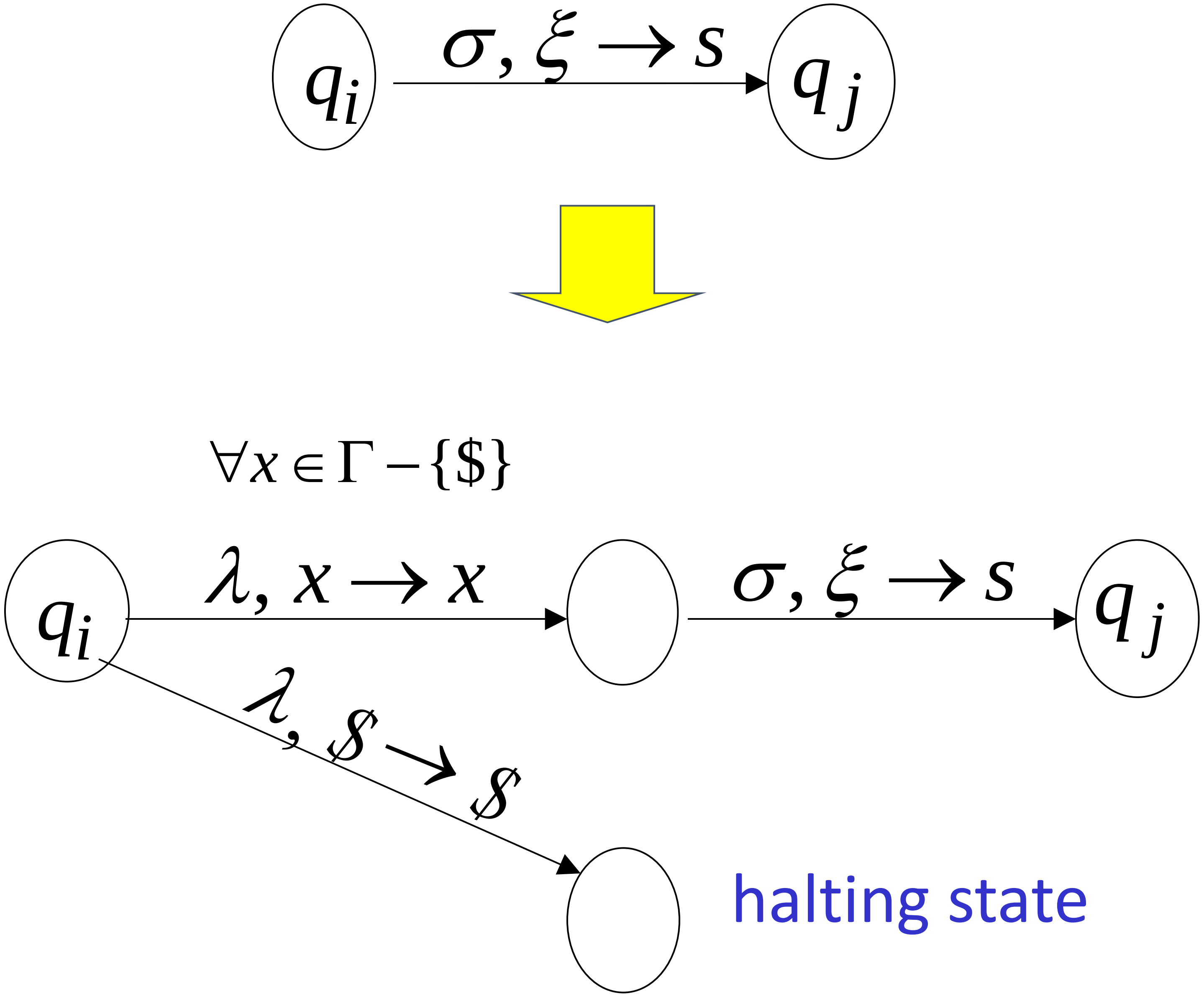
$$q_i \xrightarrow{a, \$ \rightarrow b} q_j$$

$$q_i \xrightarrow{a, \# \rightarrow b} q_j$$

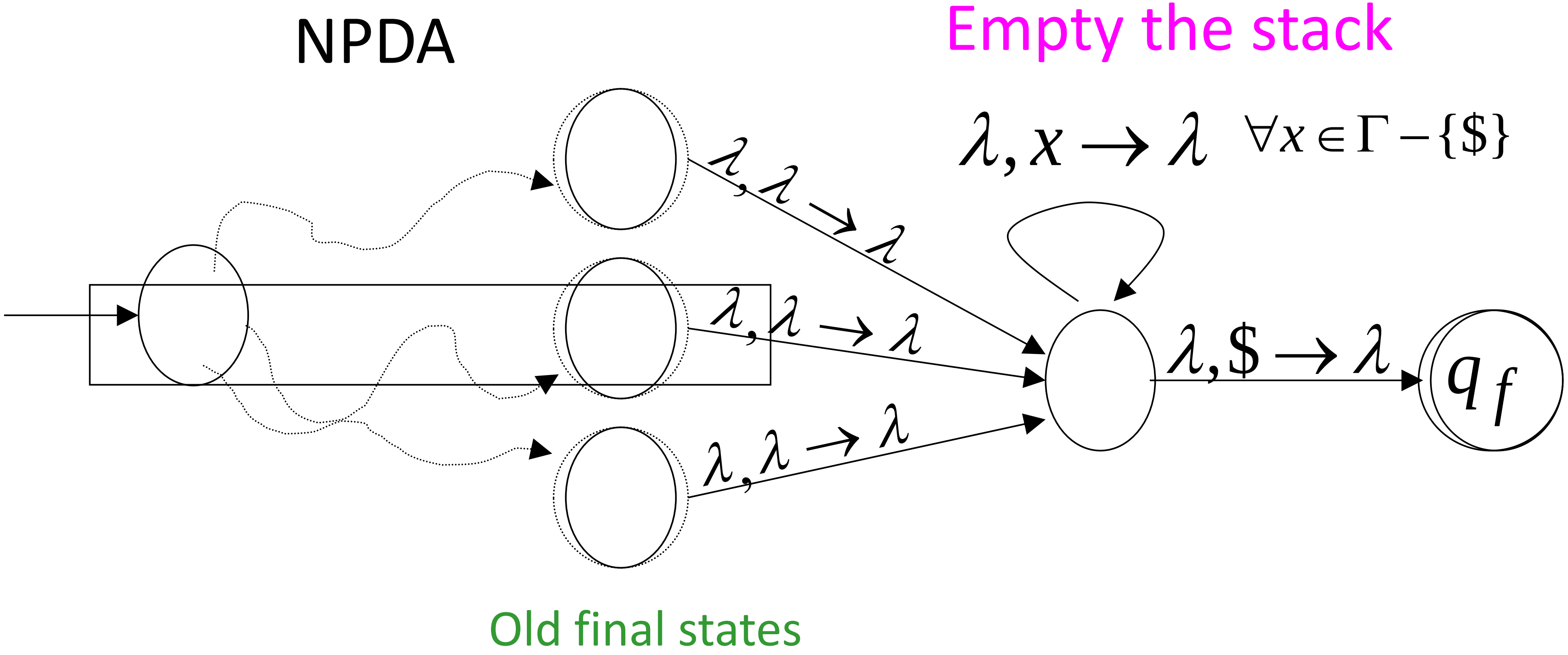
Convert all transitions so that:

if the automaton attempts to pop
or replace $\$$ it will halt

Convert transitions as follows:



2) Modify the NPDA so that it empties the stack and has a unique final state



3) modify the NPDA so that

transitions have the following forms:

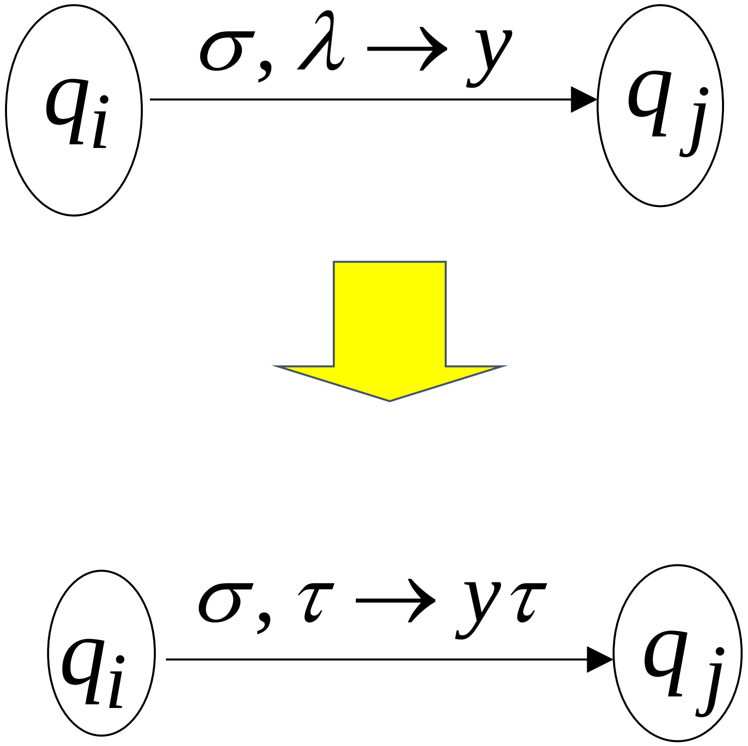
$$q_i \xrightarrow{\sigma,\, B \to \lambda} q_j$$

OR

$$q_i \xrightarrow{\sigma,\, B \to CD} q_j$$

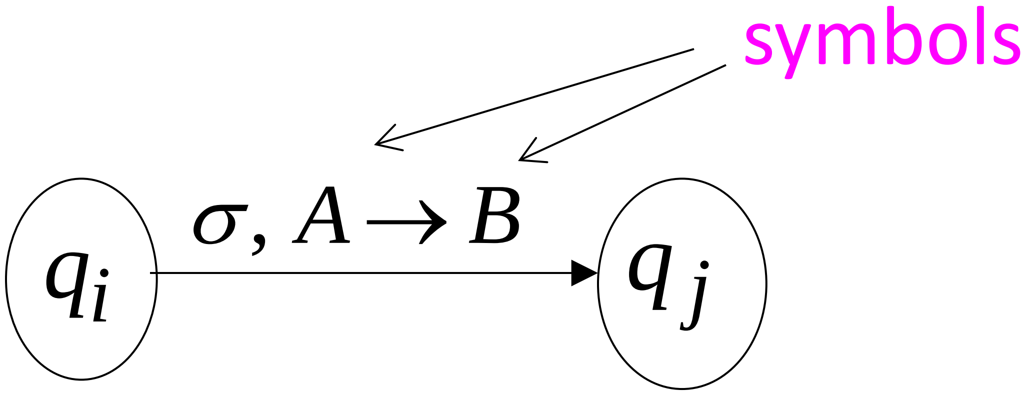
$B, C, D:$ stack symbols

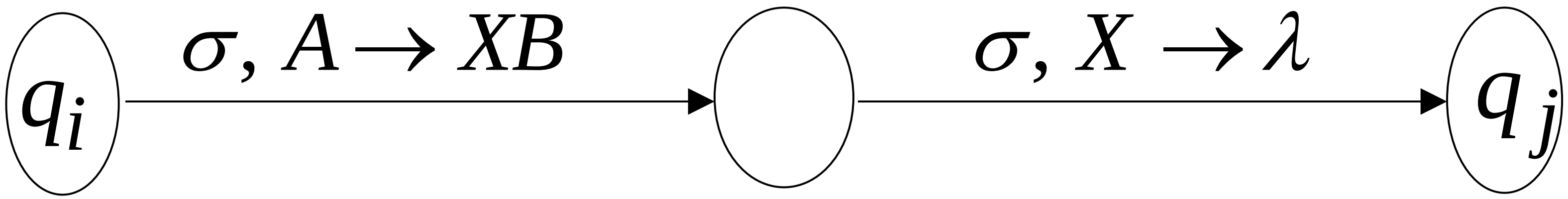
Convert:



$$\forall \tau \in \Gamma - \{\$\}$$

Convert:

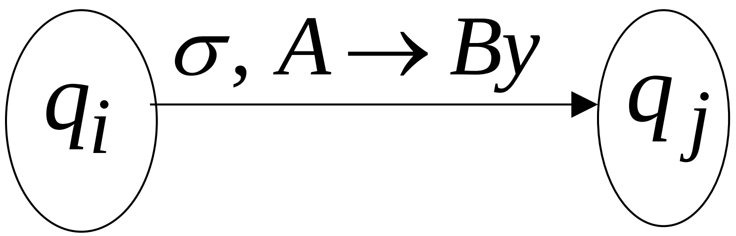




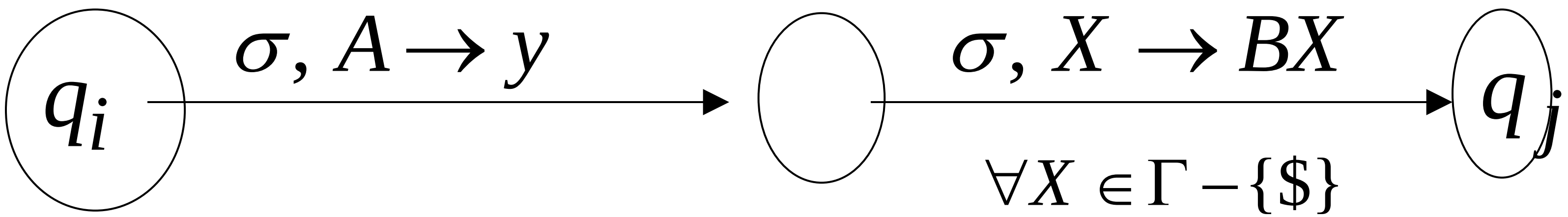
$$X \in \Gamma - \{\$\}$$

Convert: $|y| \geq 2$

symbols



Convert recursively

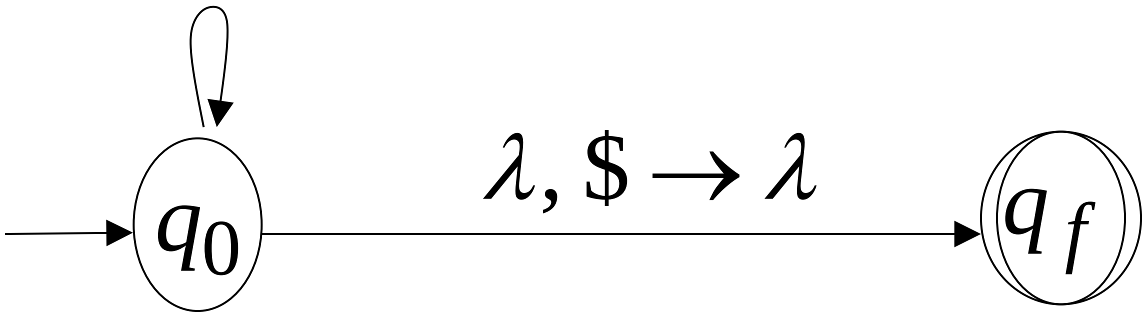


Example of a NPDA in correct form:

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

$\$$: initial stack symbol

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



The Grammar Construction

In grammar G:

Stack symbol

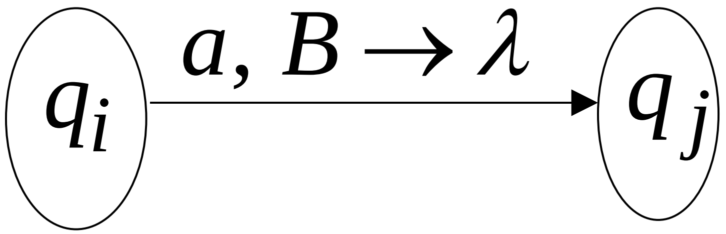
Variables: $(q_i B q_j)$

states

Terminals:

Input symbols of NPDA

For each transition



We add production

$$(q_i B q_j) \to a$$

For each transition

$$q_i \xrightarrow{a,\ B \to CD} q_j$$

We add productions

$$(q_i B q_k) \to a(q_j C q_l)(q_l D q_k)$$

For all possible states q_k, q_l

in the automaton

Stack bottom symbol

Start Variable: $(q_o \$ q_f)$

Start state

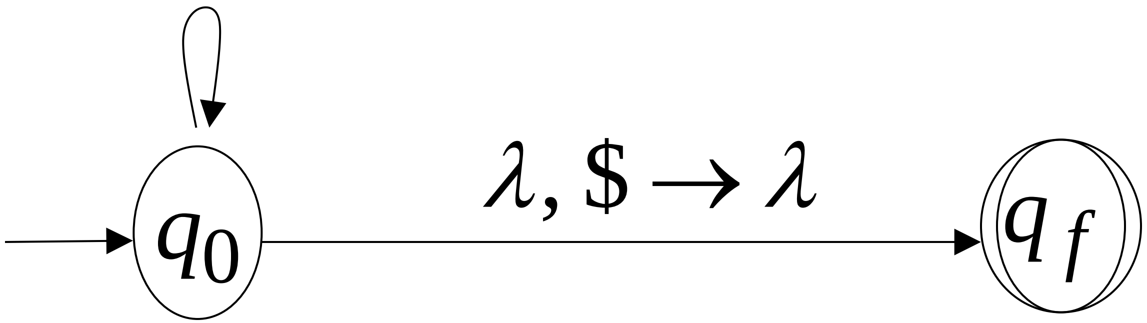
final state

Example:

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

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

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



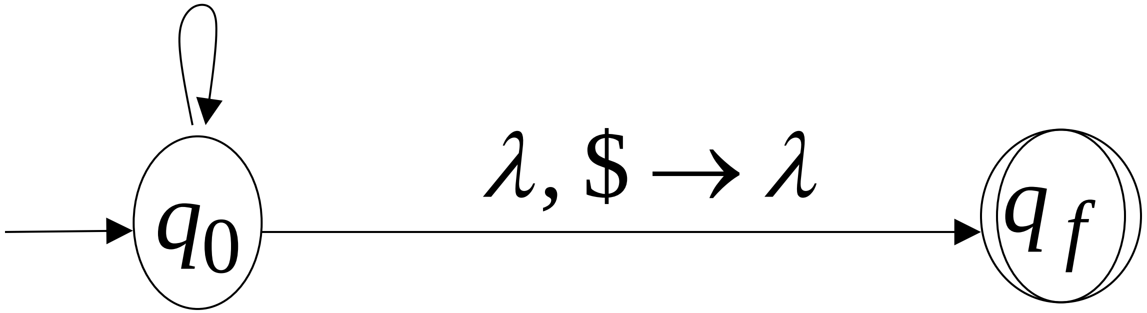
Grammar production: $(q_0 1 q_0) \rightarrow a$

Example:

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

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

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



Grammar productions:

$$(q_0 \$ q_0) \rightarrow b(q_0 1 q_0)(q_0 \$ q_0) \mid b(q_0 1 q_f)(q_f \$ q_0)$$

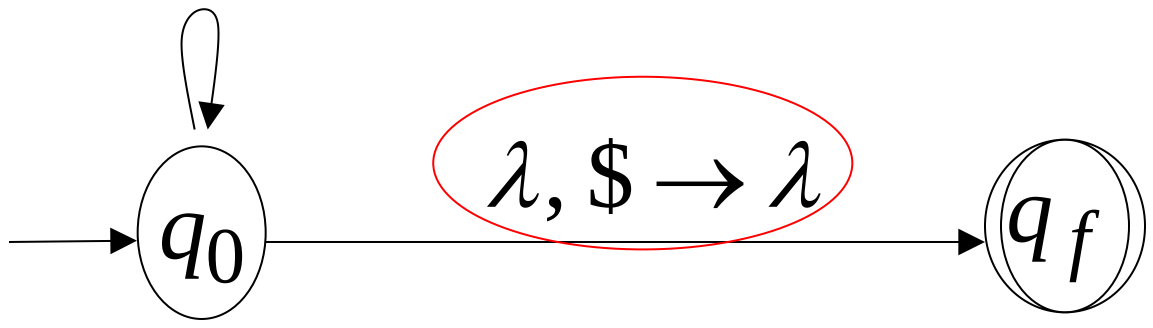
$$(q_0 \$ q_f) \rightarrow b(q_0 1 q_0)(q_0 \$ q_f) \mid b(q_0 1 q_f)(q_f \$ q_f)$$

Example:

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

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

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



Grammar production: $(q_0 \$ q_f) \to \lambda$

Resulting Grammar:

$(q_0 \$ q_f)$: start variable

$$(q_0 \$ q_0) \rightarrow b(q_0 1 q_0)(q_0 \$ q_0) \mid b(q_0 1 q_f)(q_f \$ q_0)$$

$$(q_0 \$ q_f) \rightarrow b(q_0 1 q_0)(q_0 \$ q_f) \mid b(q_0 1 q_f)(q_f \$ q_f)$$

$$(q_0 1 q_0) \rightarrow b(q_0 1 q_0)(q_0 1 q_0) \mid b(q_0 1 q_f)(q_f 1 q_0)$$

$$(q_0 1 q_f) \rightarrow b(q_0 1 q_0)(q_0 1 q_f) \mid b(q_0 1 q_f)(q_f 1 q_f)$$

$$(q_0 \$ q_0) \rightarrow a(q_0 0 q_0)(q_0 \$ q_0) \mid a(q_0 0 q_f)(q_f \$ q_0)$$

$$(q_0 \$ q_f) \rightarrow a(q_0 0 q_0)(q_0 \$ q_f) \mid a(q_0 0 q_f)(q_f \$ q_f)$$

$$(q_0 0 q_0) \to a(q_0 0 q_0)(q_0 0 q_0) \mid a(q_0 0 q_f)(q_f 0 q_0)$$

$$(q_0 0 q_f) \to a(q_0 0 q_0)(q_0 0 q_f) \mid a(q_0 0 q_f)(q_f 0 q_f)$$

$$(q_0 1 q_0) \to a$$

$$(q_0 0 q_0) \to b$$

$$(q_0 \$ q_f) \to \lambda$$

Derivation of string $abba$

$$(q_0\$q_f) \Rightarrow \; a(q_0 0 q_0)(q_0\$q_f) \Rightarrow$$

$$ab(q_0\$q_f) \Rightarrow$$

$$abb(q_0 1 q_0)(q_0\$q_f) \Rightarrow$$

$$abba(q_0\$q_f) \Rightarrow \; abba$$

In general:

$$(q_i A q_j) \overset{*}{\Rightarrow} w$$

if and only if

the NPDA goes from q_i to q_j
by reading string w and A
the stack doesn't change below
and then A is removed from stack

Therefore:

$$(q_0\$q_f) \overset{*}{\Rightarrow} w$$

if and only if

w is accepted by the NPDA

Therefore:

For any NPDA
there is a context-free grammar
that accepts the same language

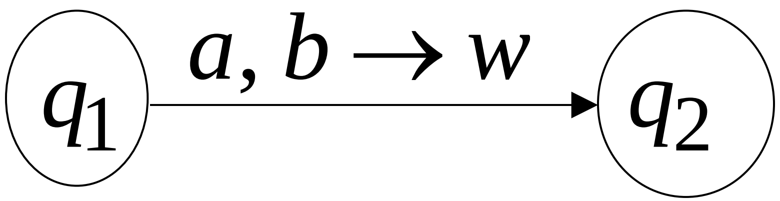
$$\left\{ \begin{array}{c} \text{Context-Free} \\ \text{Languages} \\ \text{(Grammars)} \end{array} \right\} \supseteq \left\{ \begin{array}{c} \text{Languages} \\ \text{Accepted by} \\ \text{NPDAs} \end{array} \right\}$$

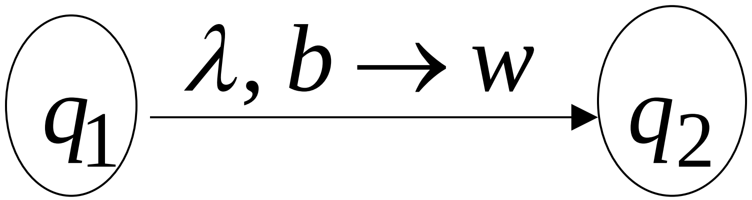
Outline

- Last week
- Conversions around Context-free Languages
- Deterministic PDA(DPDA)
- Turing Machines
- Review

Deterministic PDA: DPDA

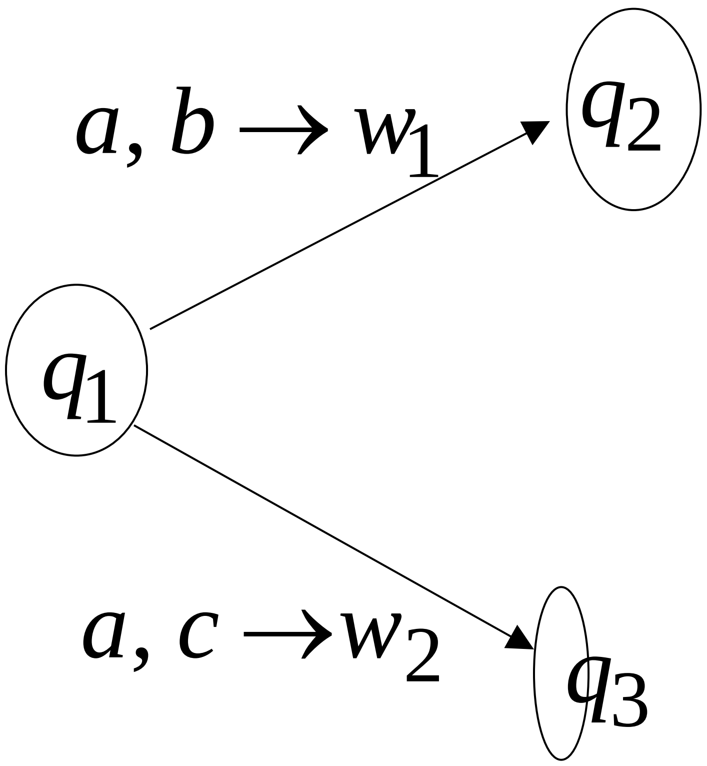
Allowed transitions:

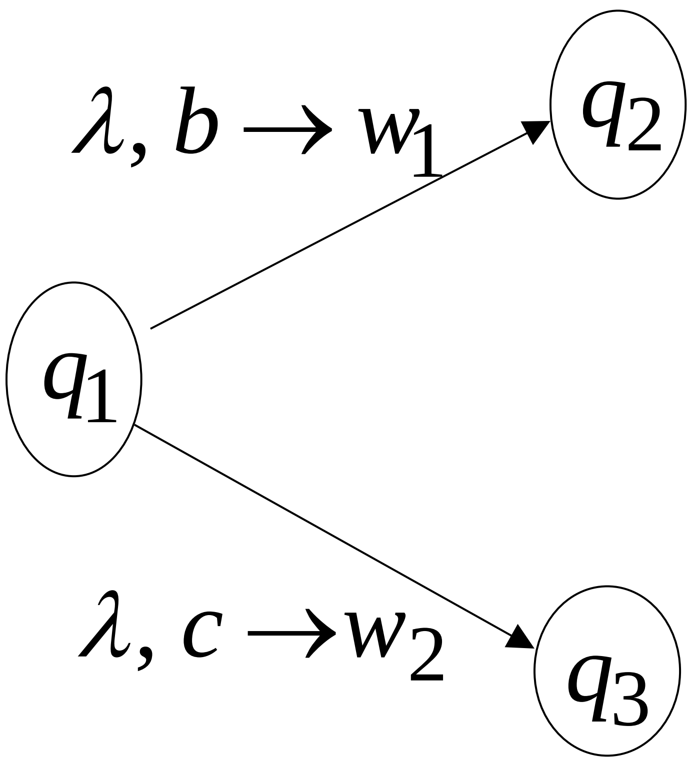




(deterministic choices)

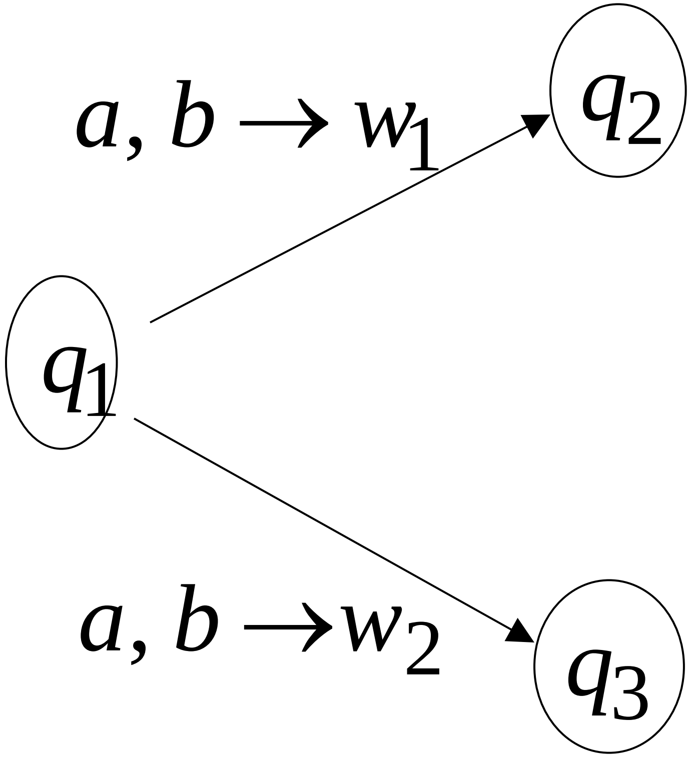
Allowed transitions:

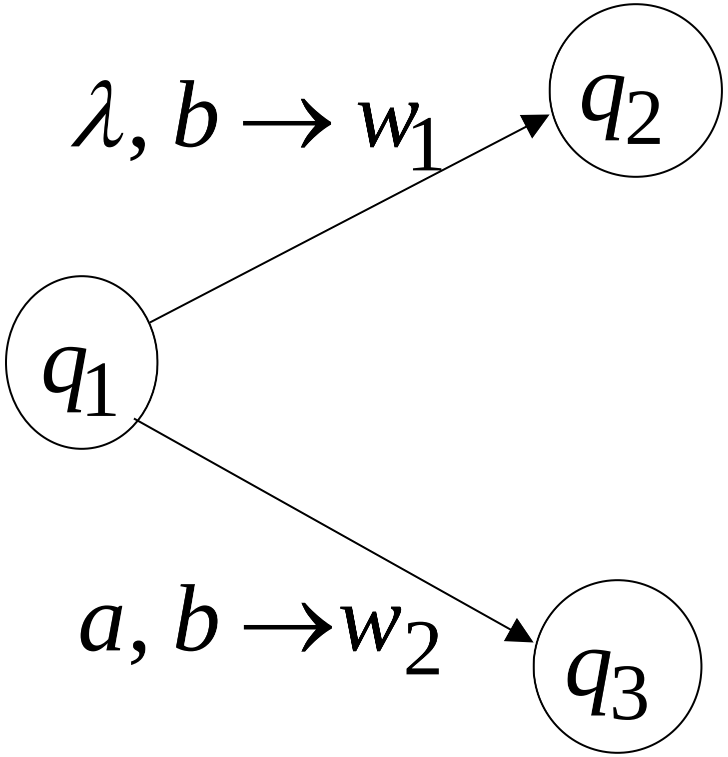




(deterministic choices)

Not allowed:

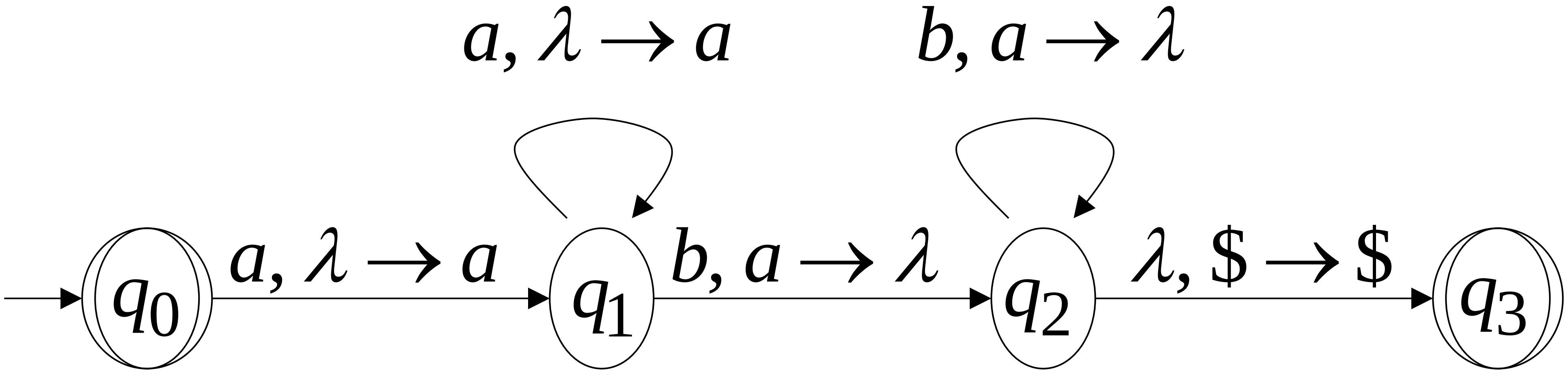




(non deterministic choices)

DPDA example

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



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

is **deterministic context-free**

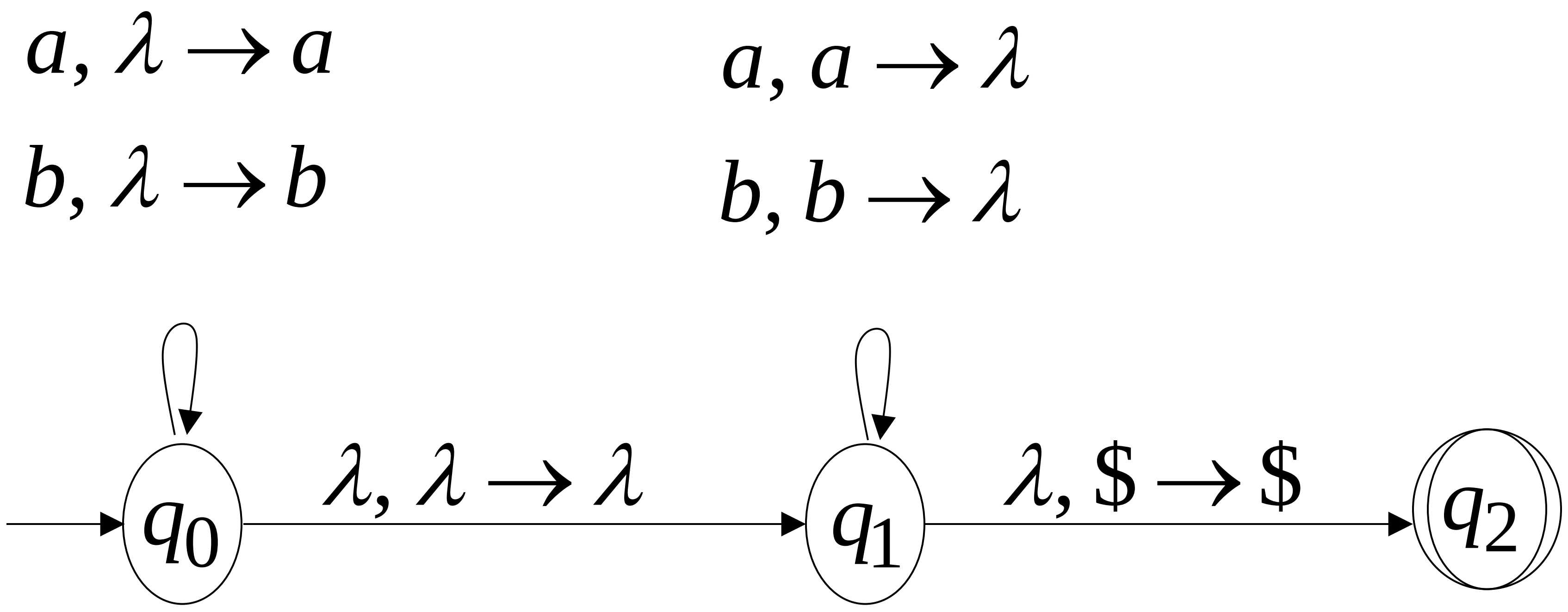
Definition:

A language L is **deterministic context-free**

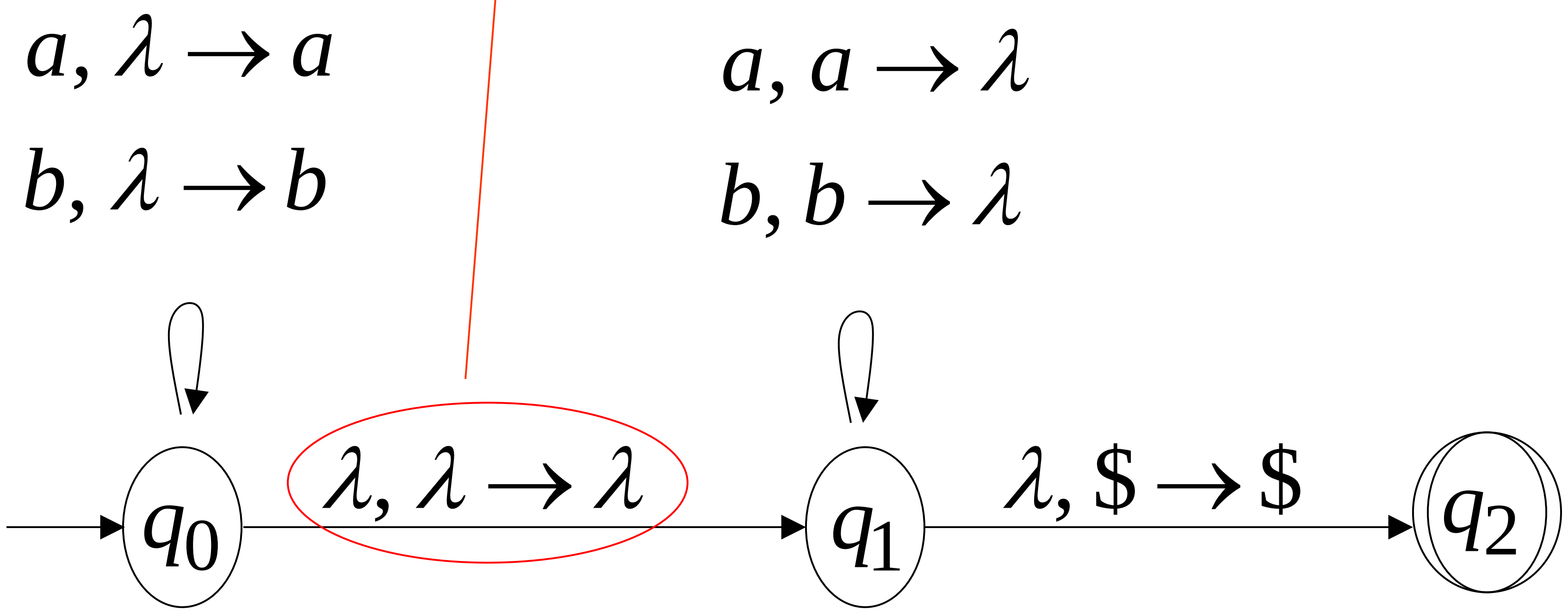
if there exists some DPDA that accepts it

Example of Non-DPDA (NPDA)

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



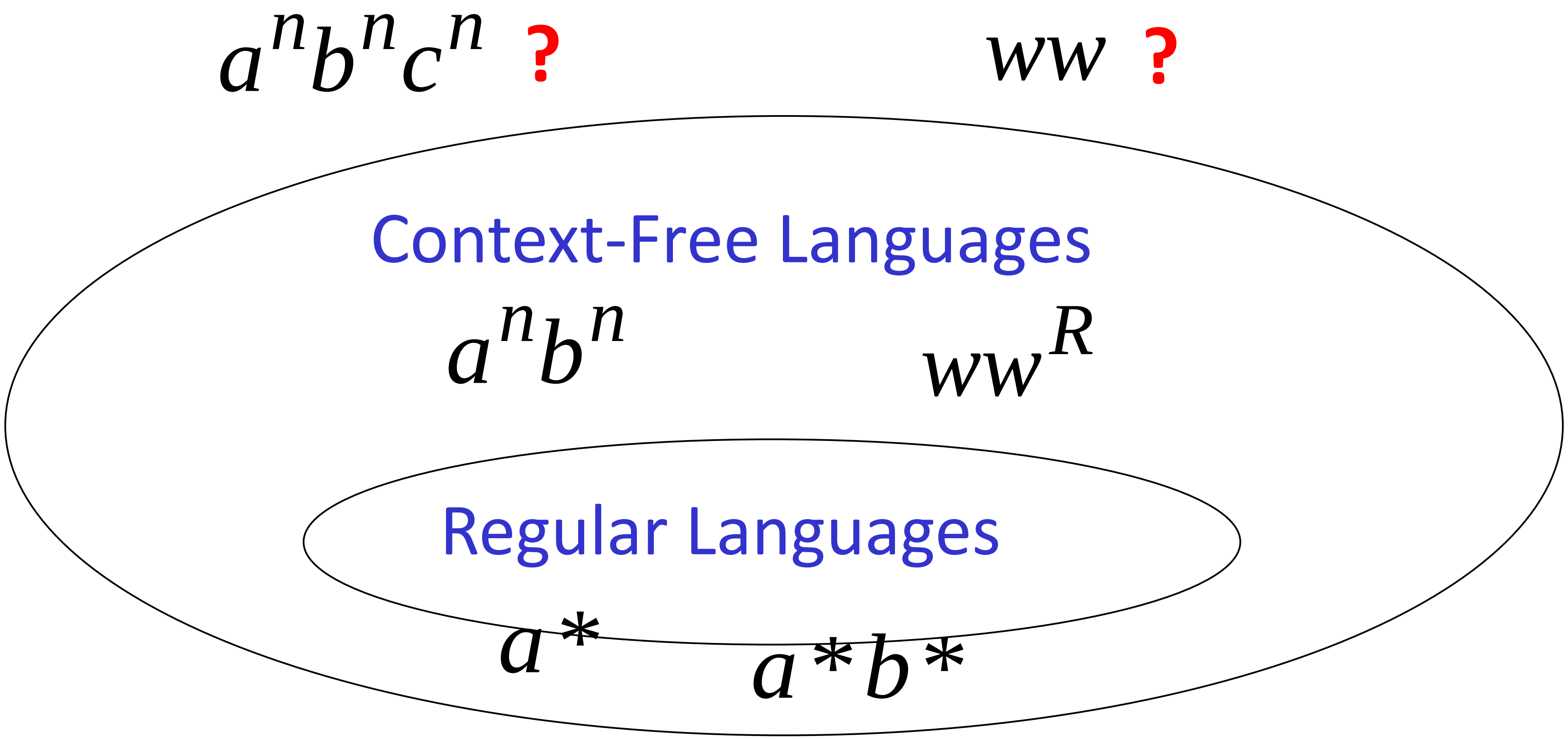
Not allowed in DPDAs

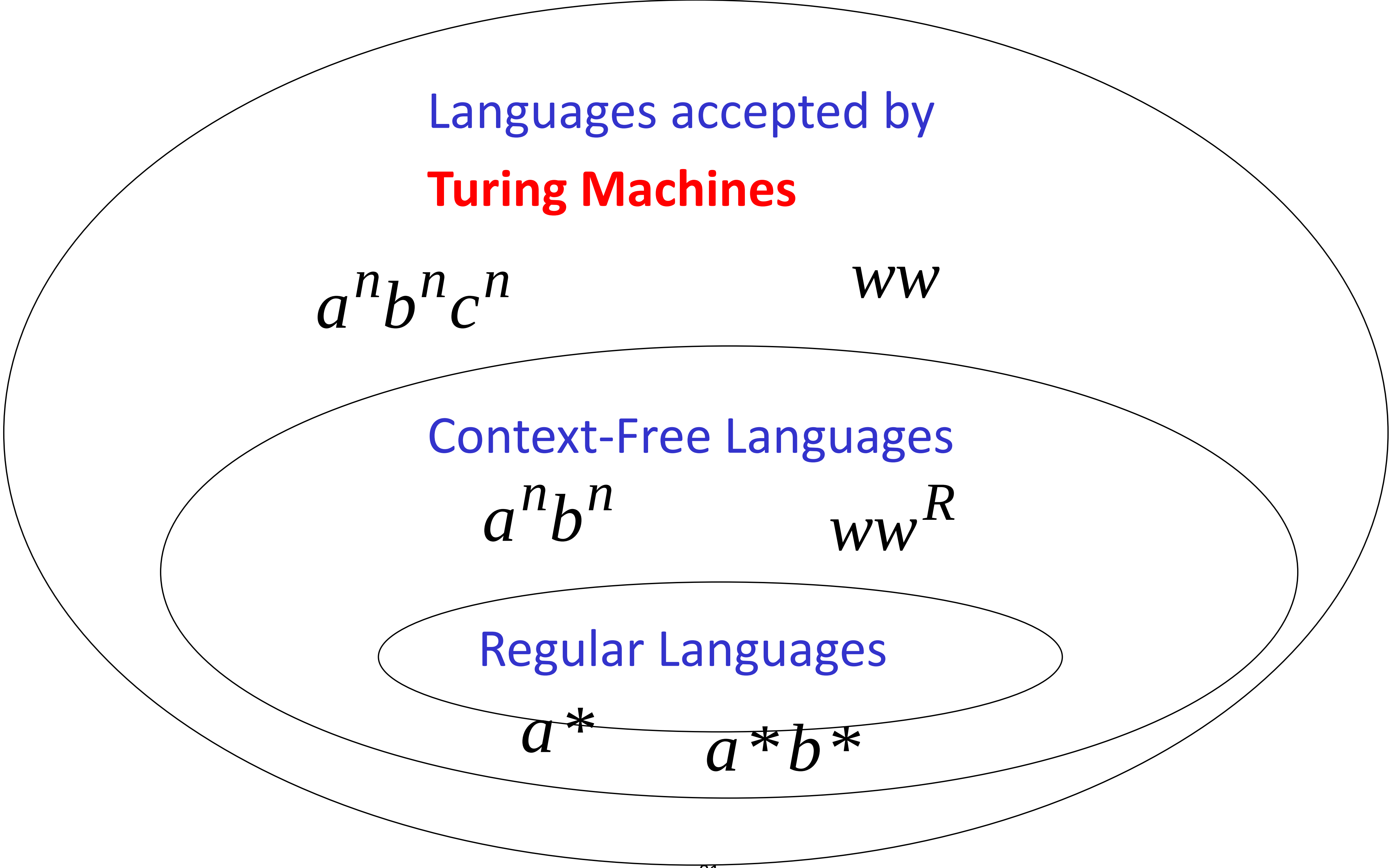


Outline

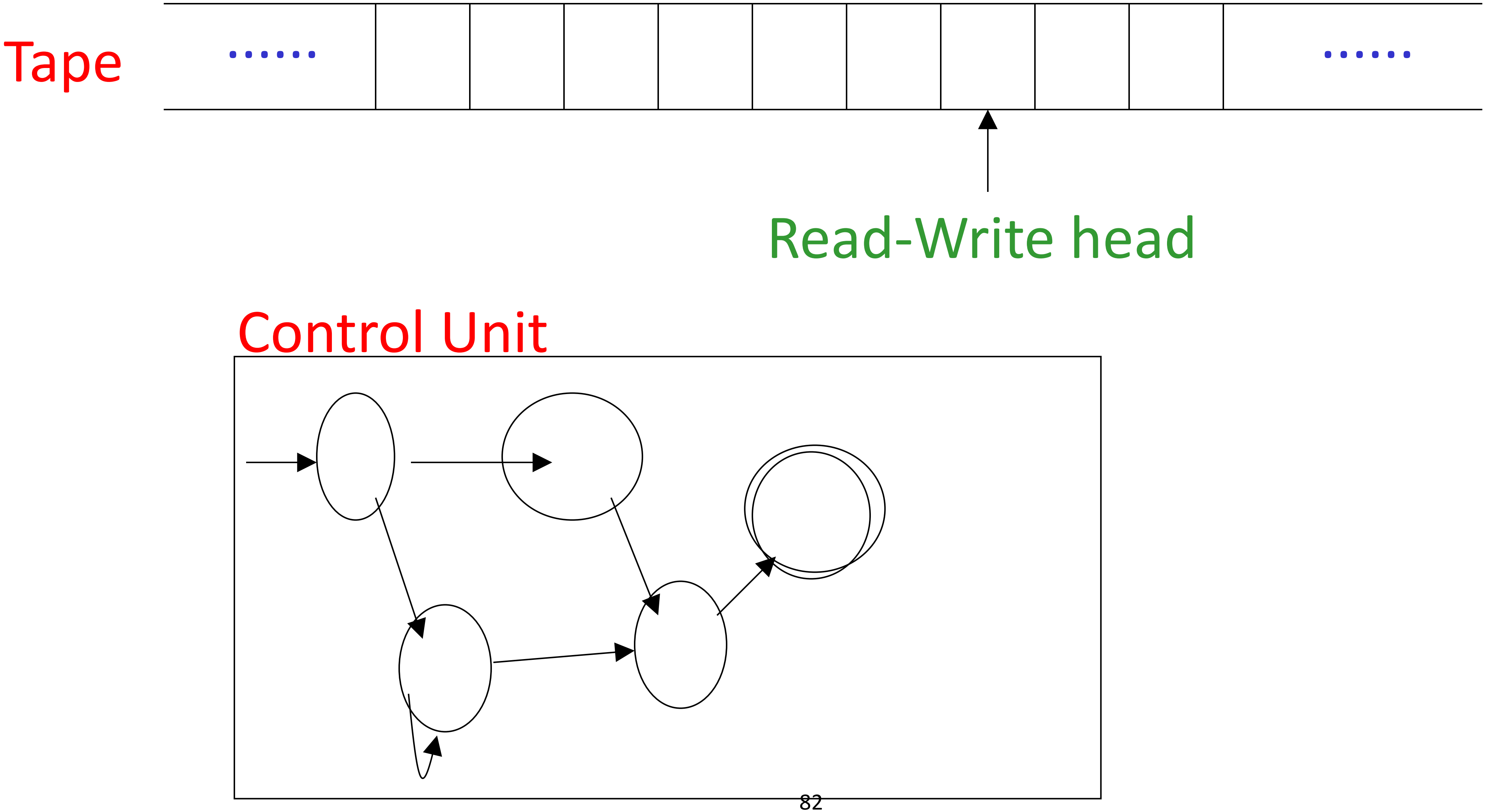
- Last week
- Conversions around Context-free Languages
- Deterministic PDA(DPDA)
- Turing Machines
- Review

The Language Hierarchy



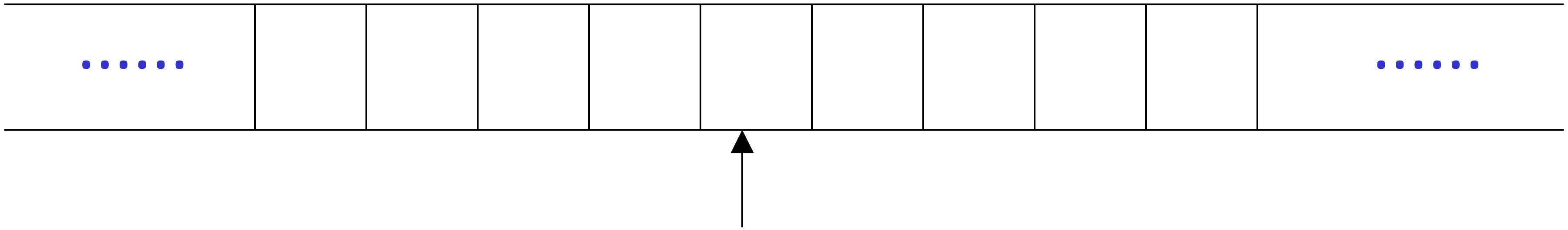

Languages accepted by
Turing Machines
$a^n b^n c^n$
ww
Context-Free Languages
$a^n b^n$
ww^R
Regular Languages
a^*
a^*b^*

A Turing Machine



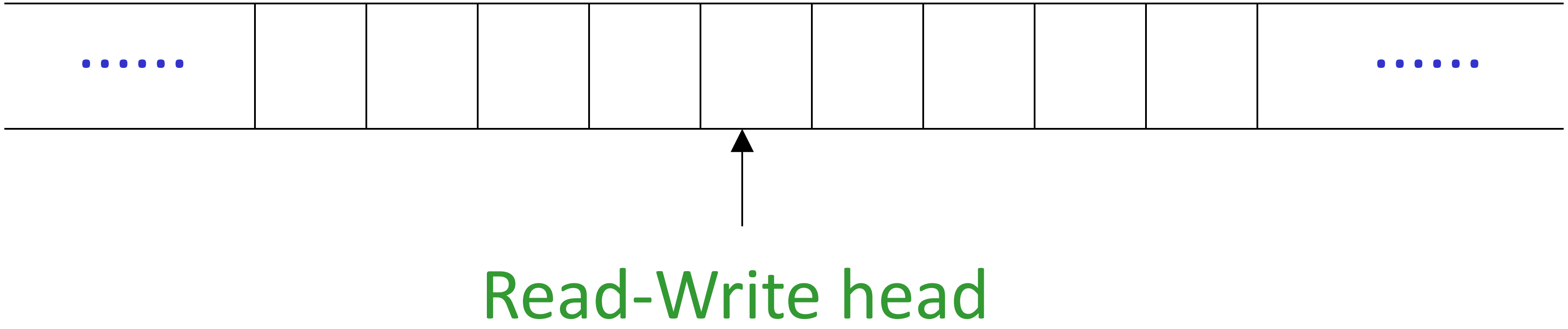
The Tape

No boundaries -- infinite length



Read-Write head

The head moves Left or Right

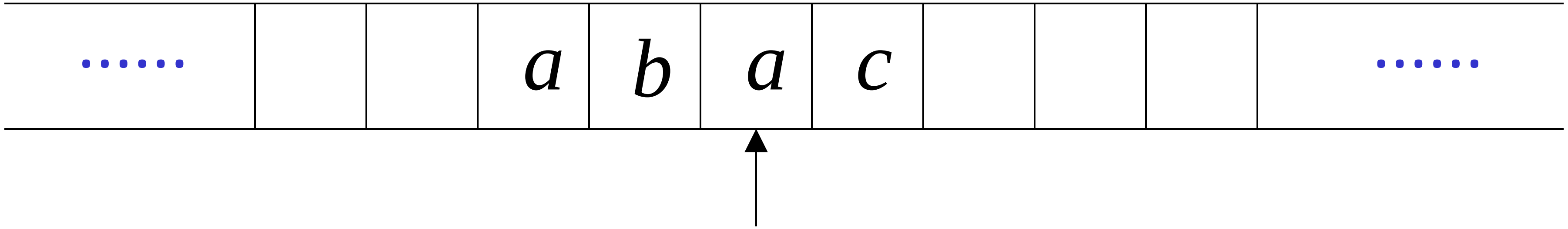


The head at each time step:

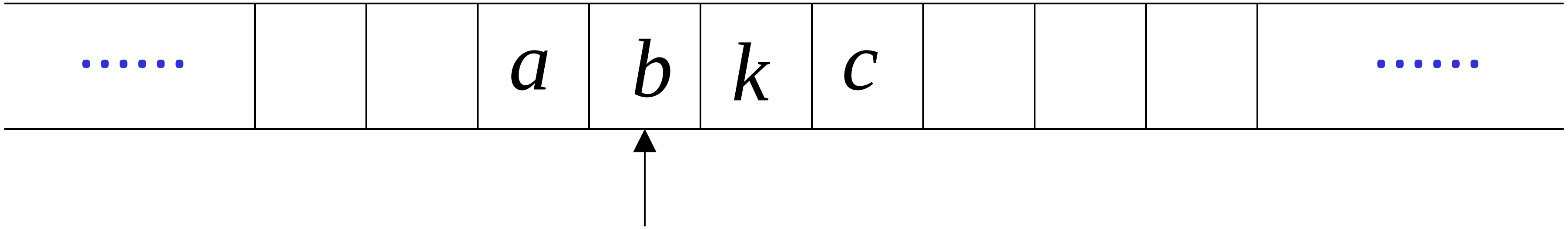
1. Reads a symbol
2. Writes a symbol
3. Moves Left or Right

Example:

Time 0



Time 1

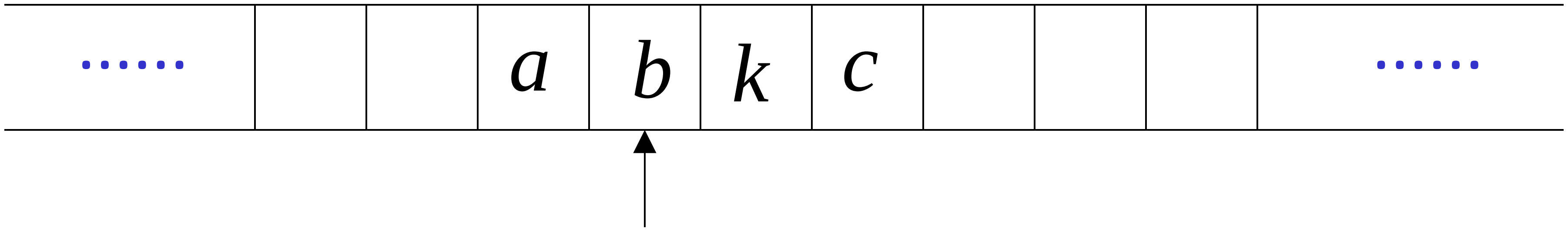


1. Reads a

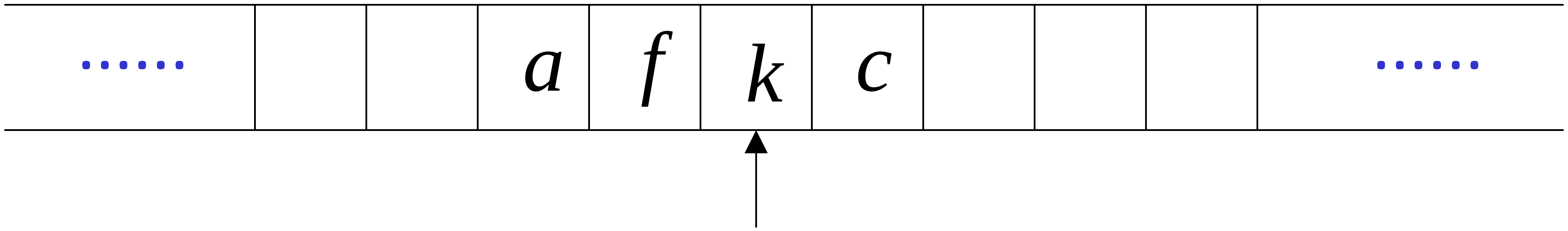
2. Writes k

3. Moves Left

Time 1



Time 2

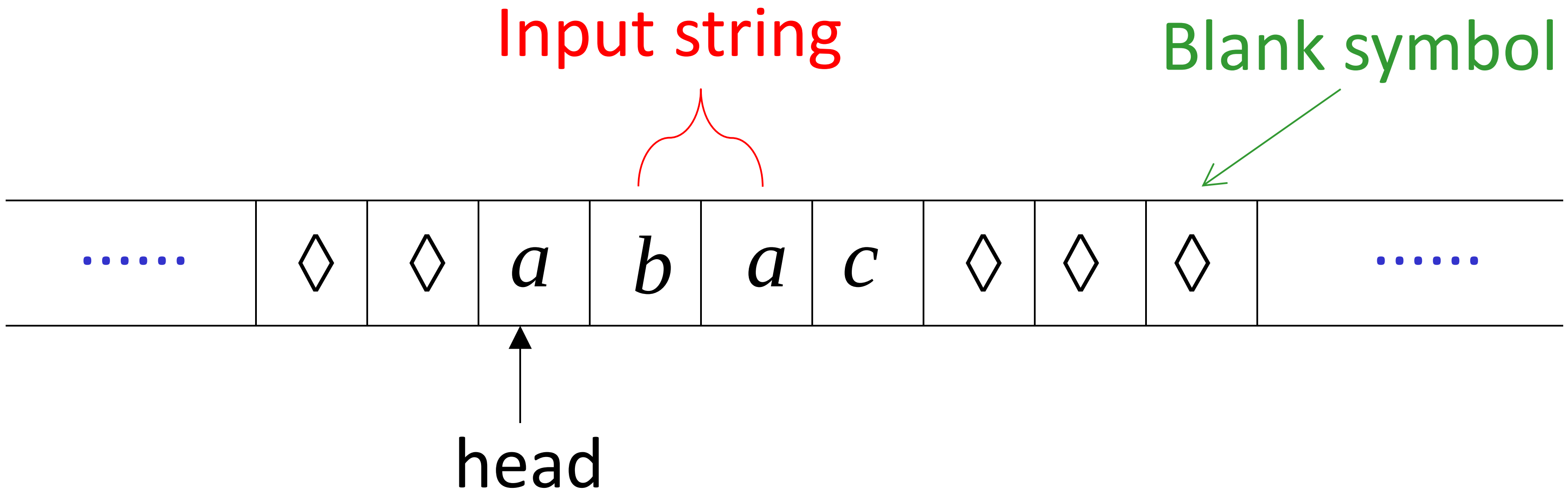


1. Reads b

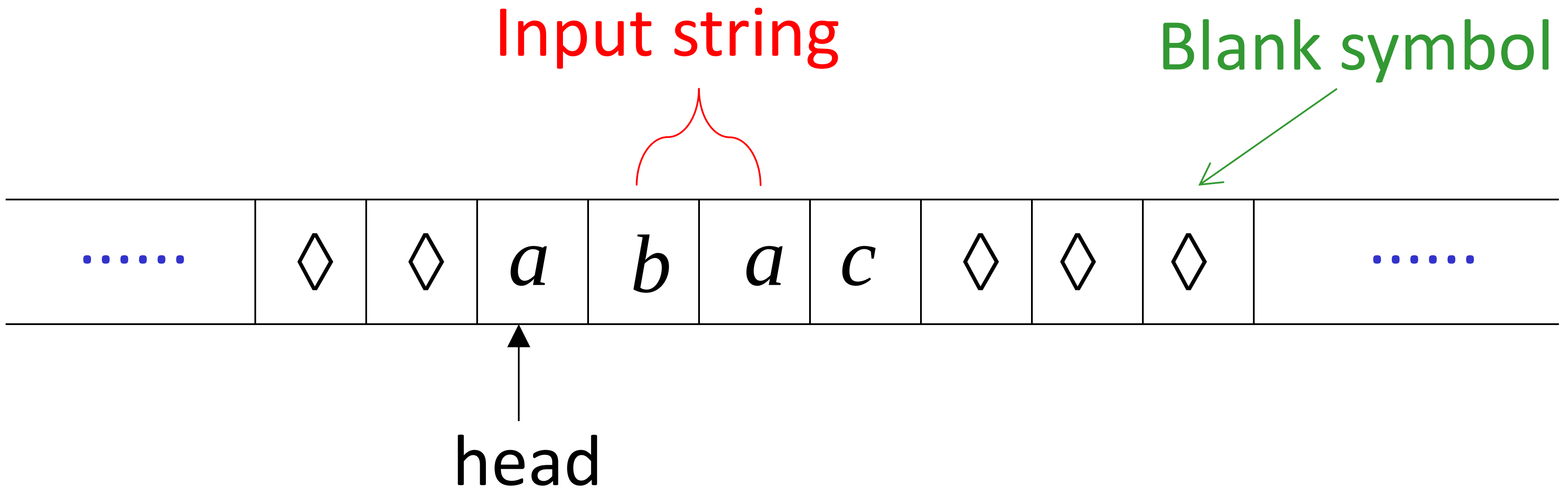
2. Writes f

3. Moves Right

The Input String

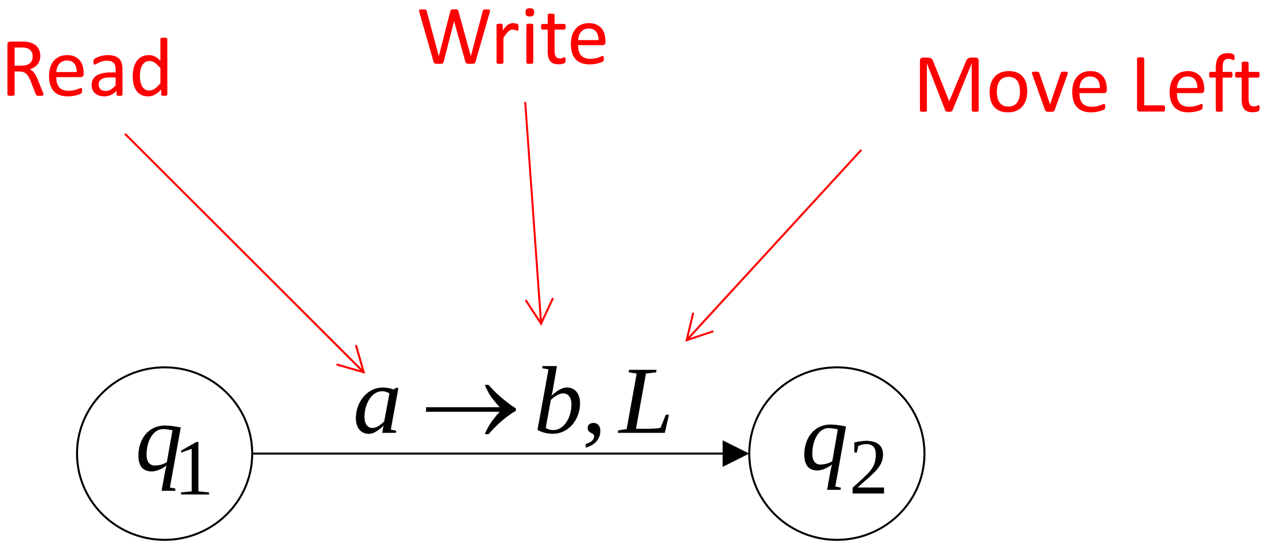


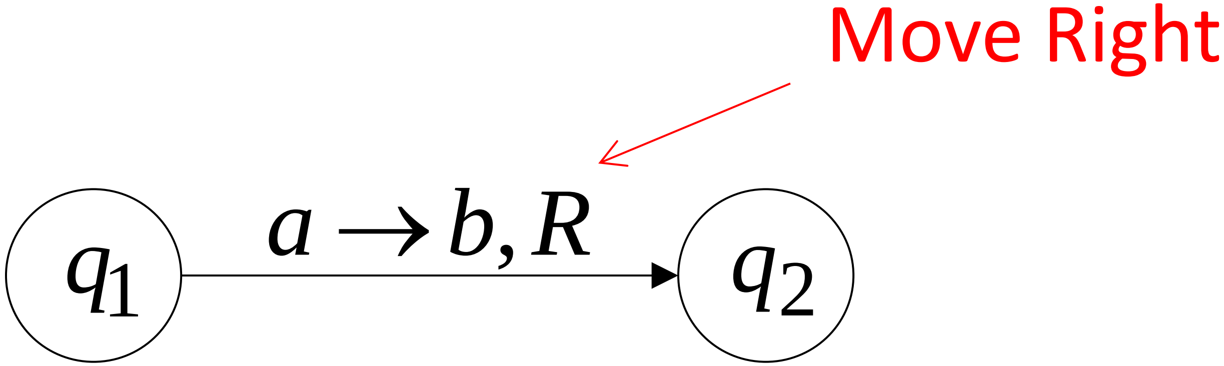
Head starts at the leftmost position of the input string



Remark: The input string is never empty

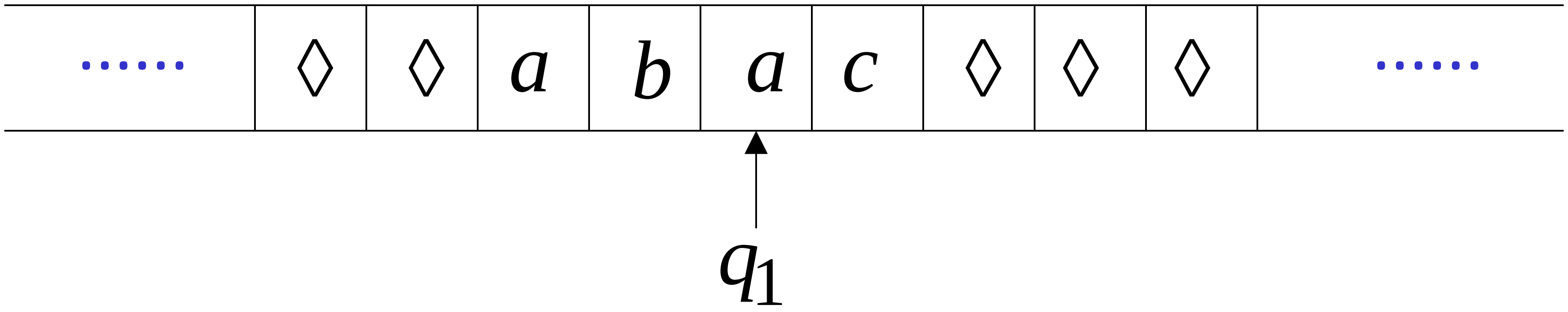
States & Transitions



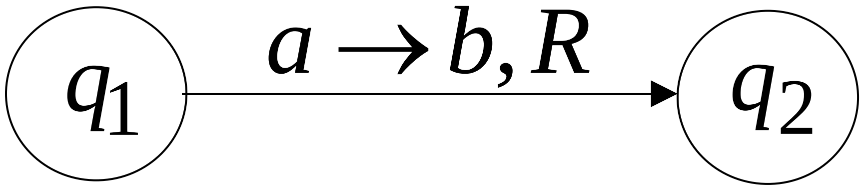


Example:

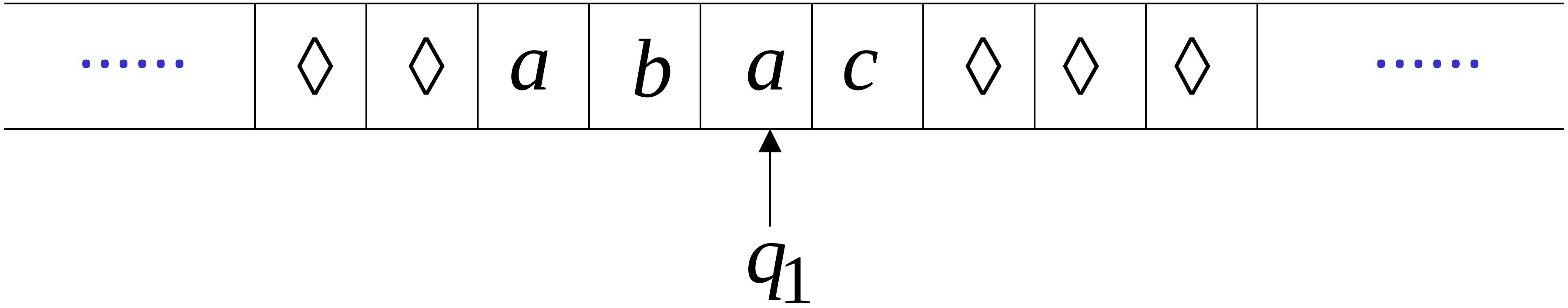
Time 1



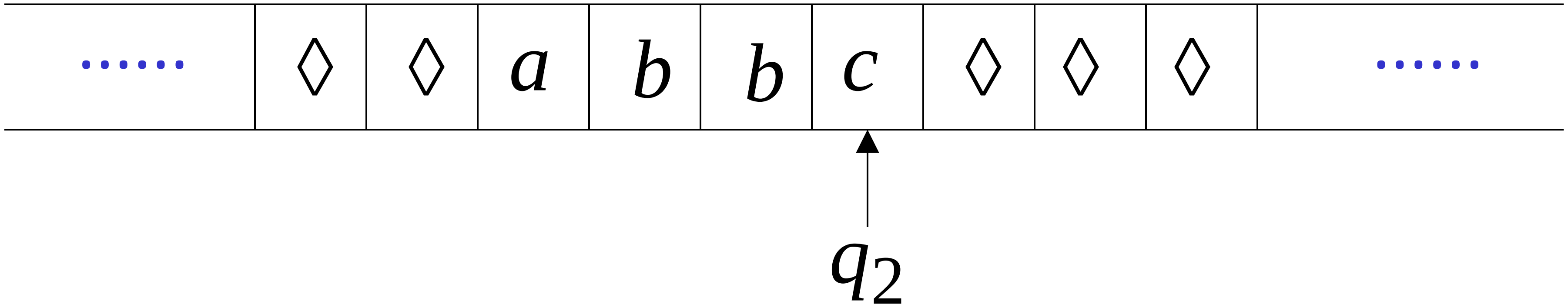
current state

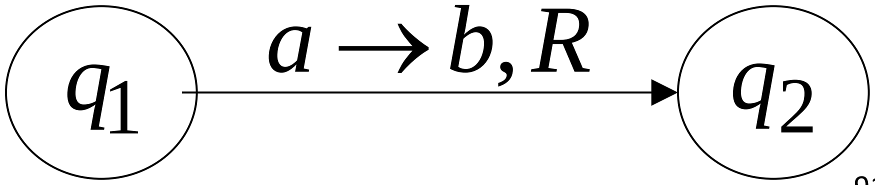


Time 1

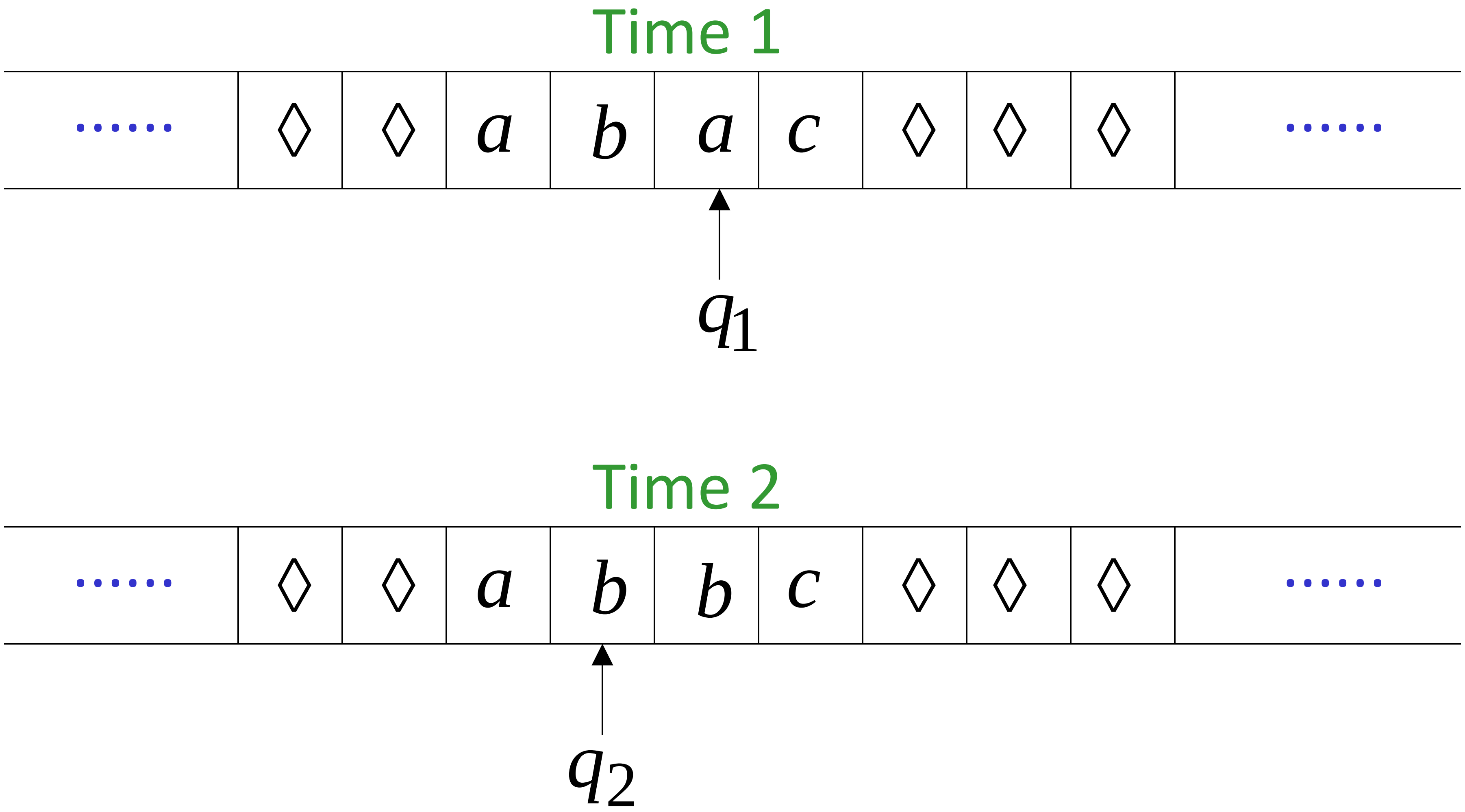


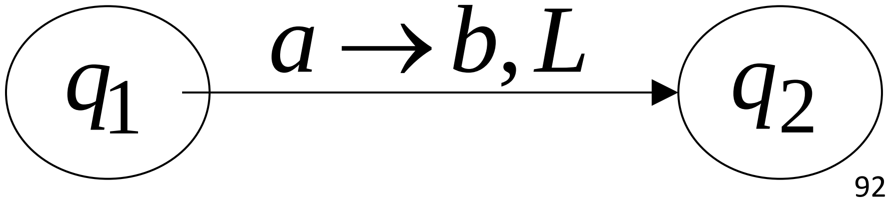
Time 2





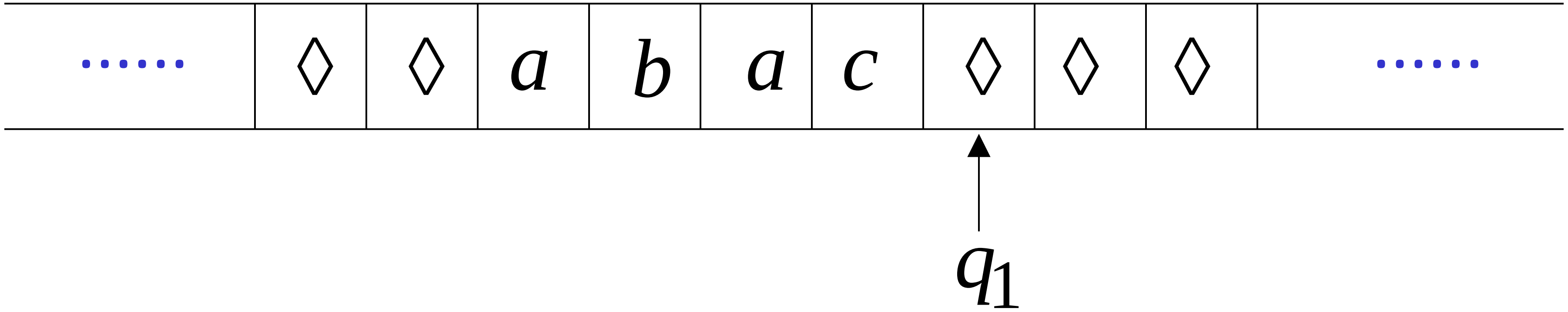
Example:



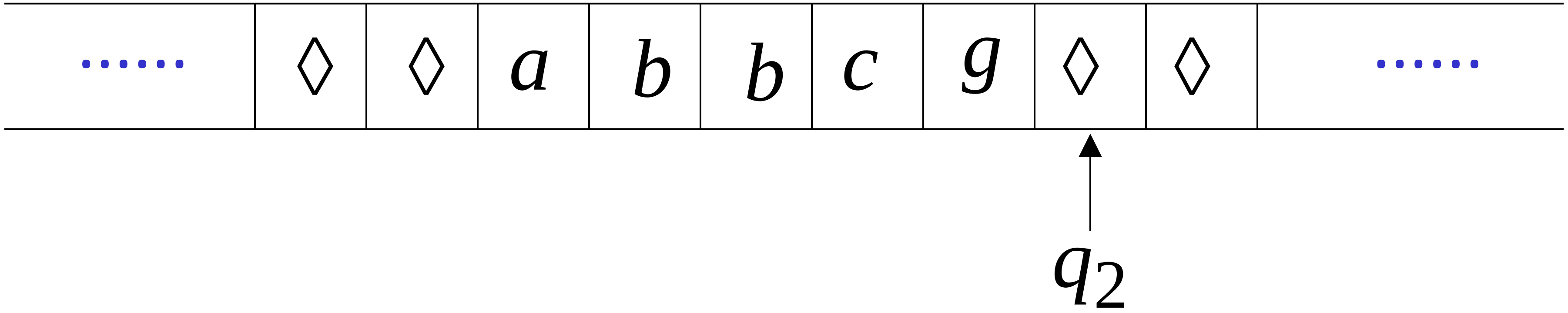


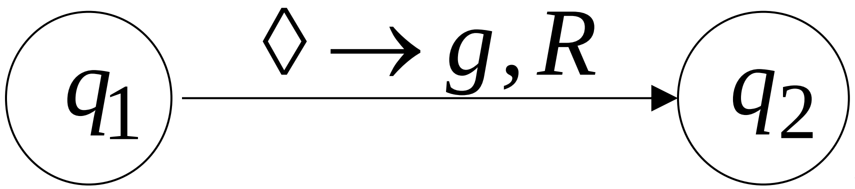
Example:

Time 1



Time 2

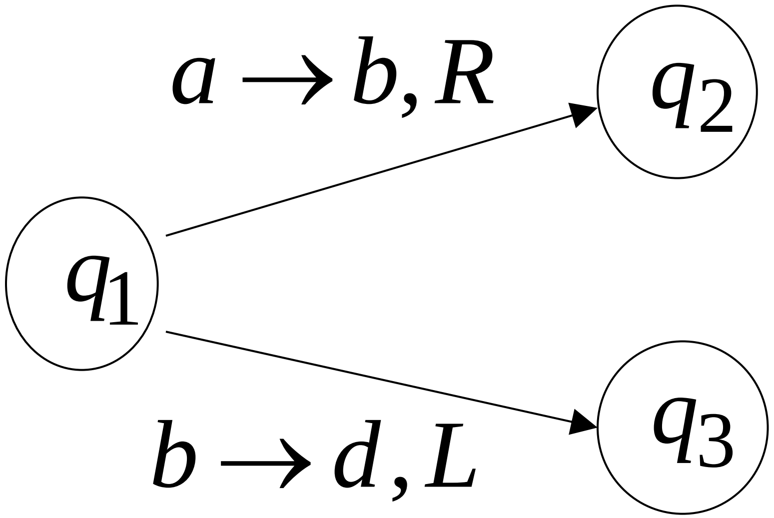




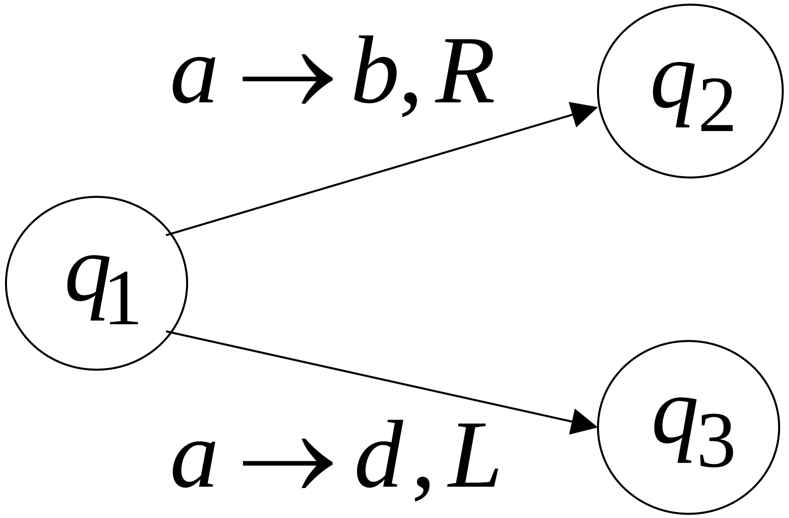
Determinism

Turing Machines are deterministic

Allowed



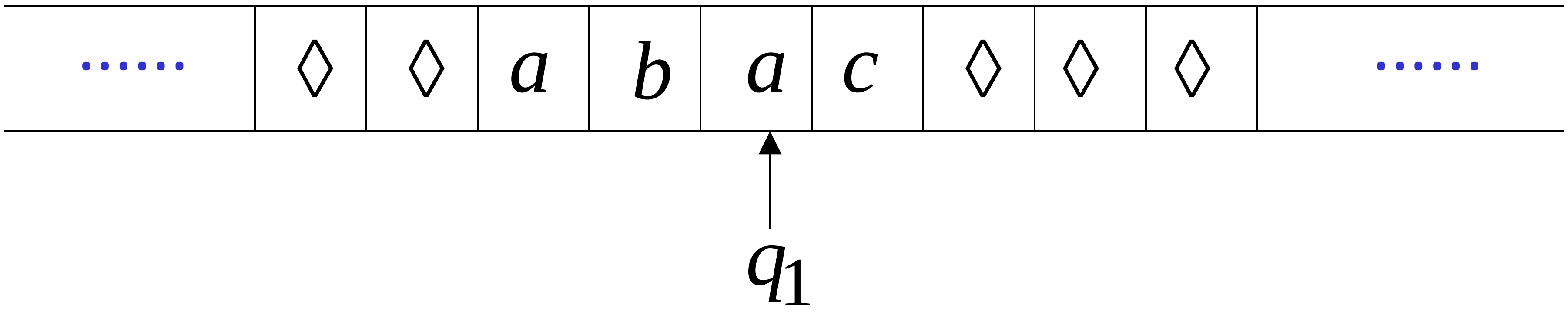
Not Allowed

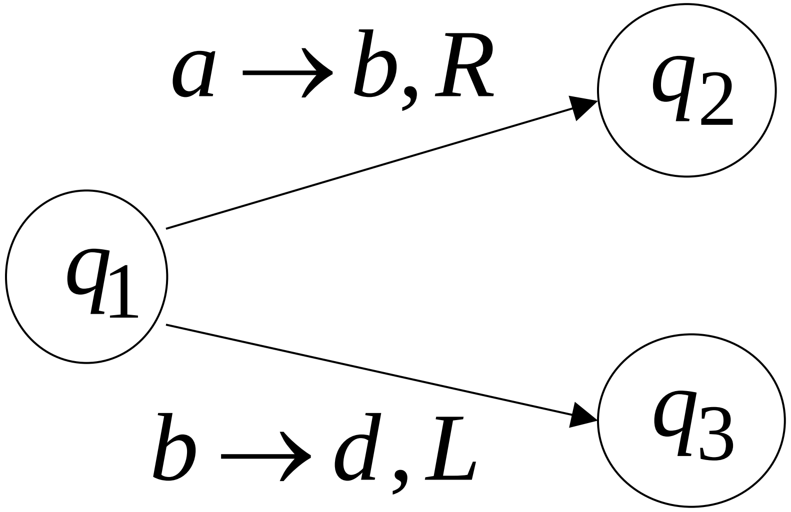


No lambda transitions allowed

Partial Transition Function

Example:





Allowed:

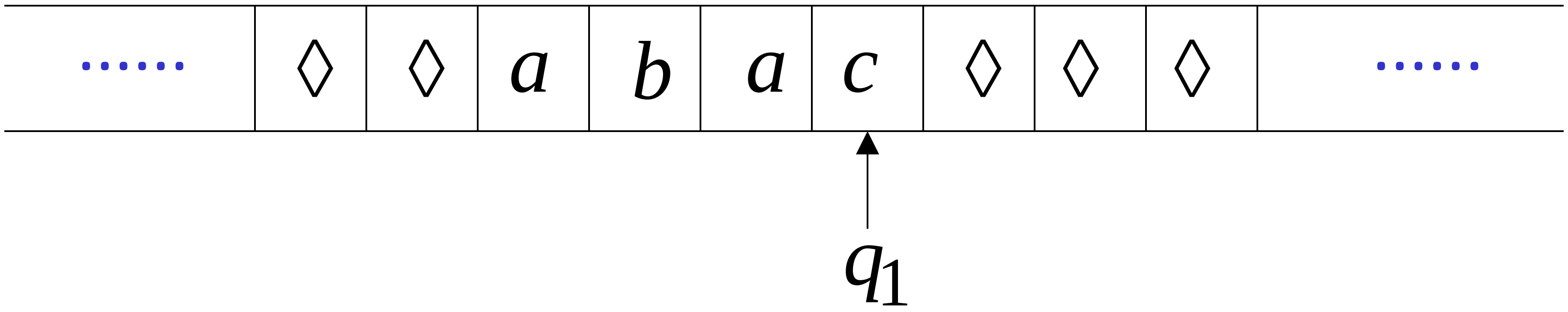
No transition

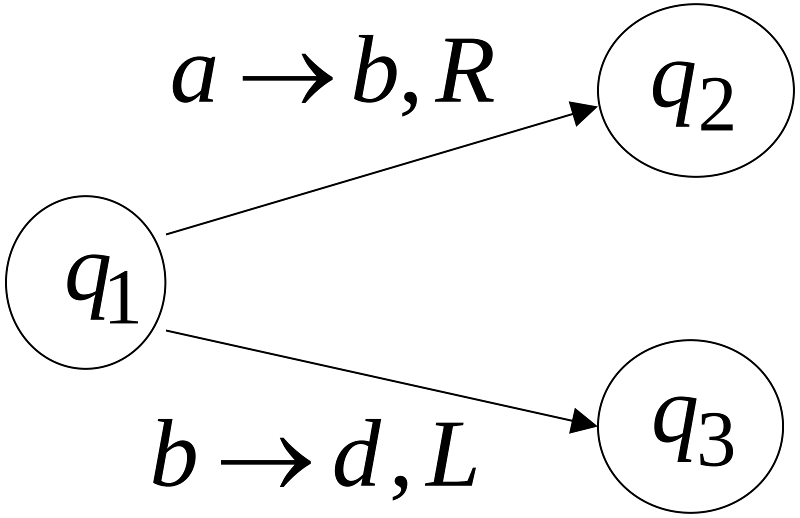
for input symbol c

Halting

The machine ***halts*** if there are
no possible transitions to follow

Example:



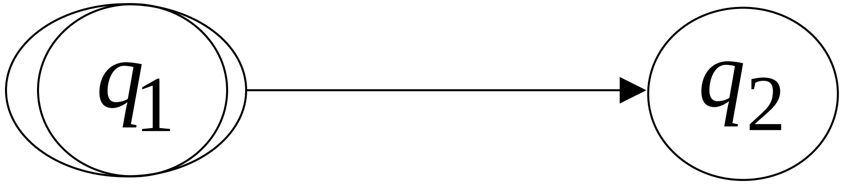


No possible transition

HALT!!!

Final States

Allowed



Not Allowed

- Final states have no outgoing transitions

- In a final state the machine halts

Acceptance

Accept Input ⇒ If machine halts in a final state

Reject Input ⇒ If machine halts in a non-final state

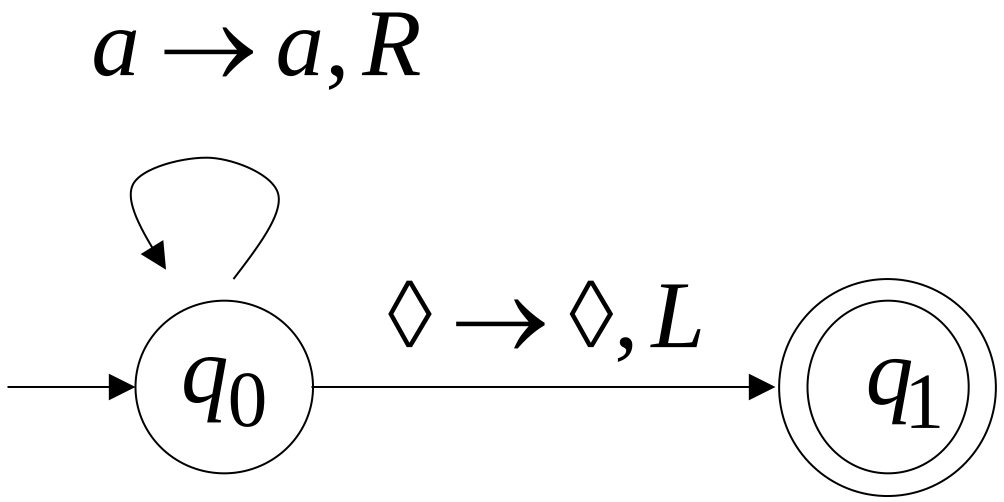
or

If machine enters an *infinite loop*

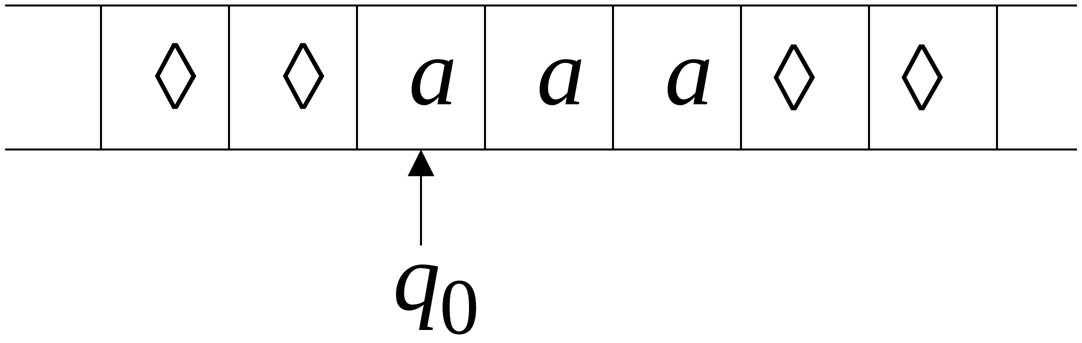
Turing Machine Example

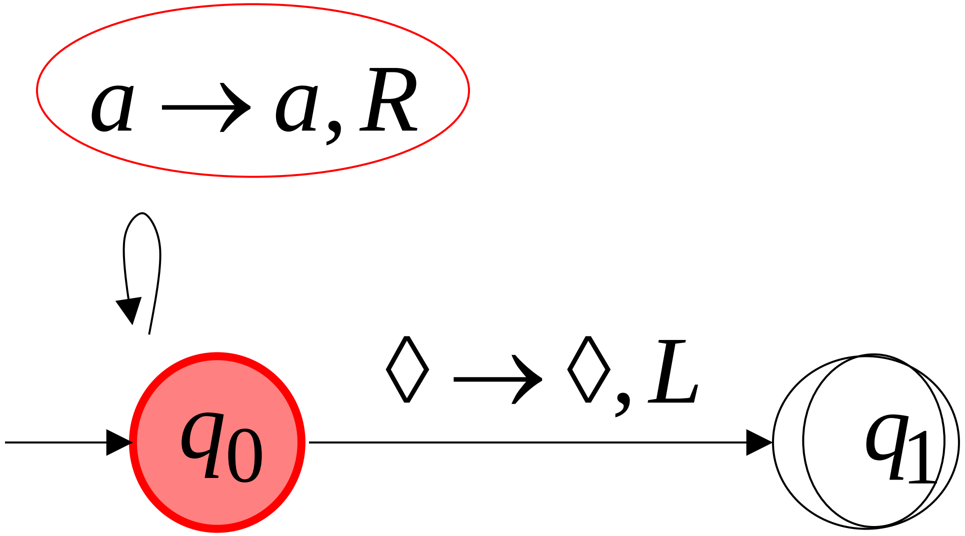
A Turing machine that accepts the language:

$$aa^*$$

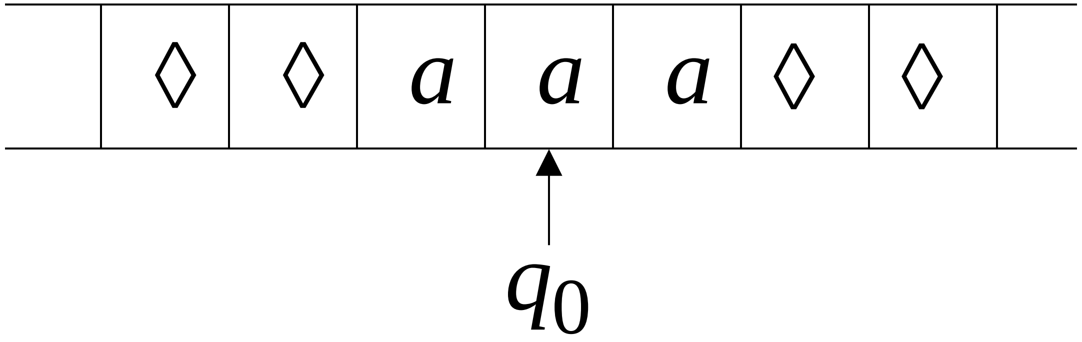


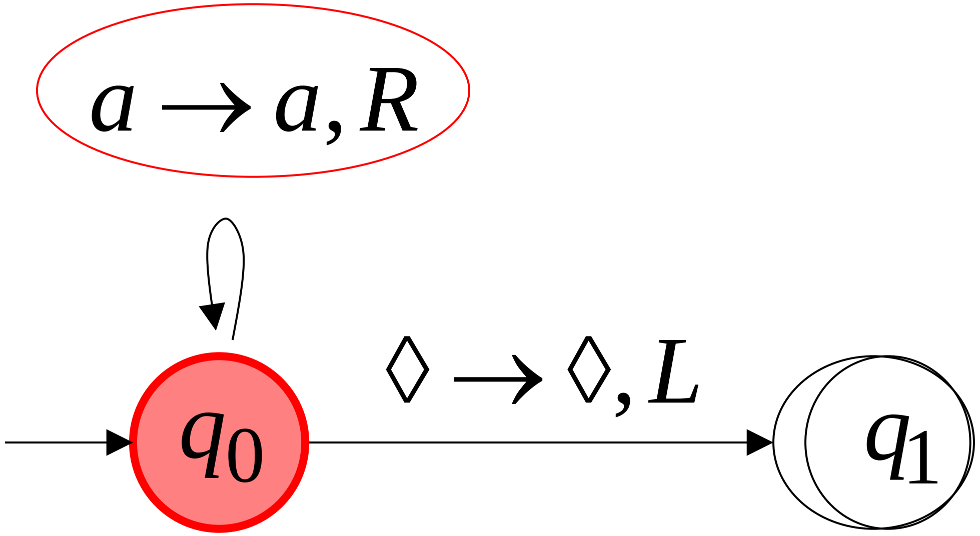
Time 0


$\Diamond$ $\Diamond$ a a a $\Diamond$ $\Diamond$
q_0

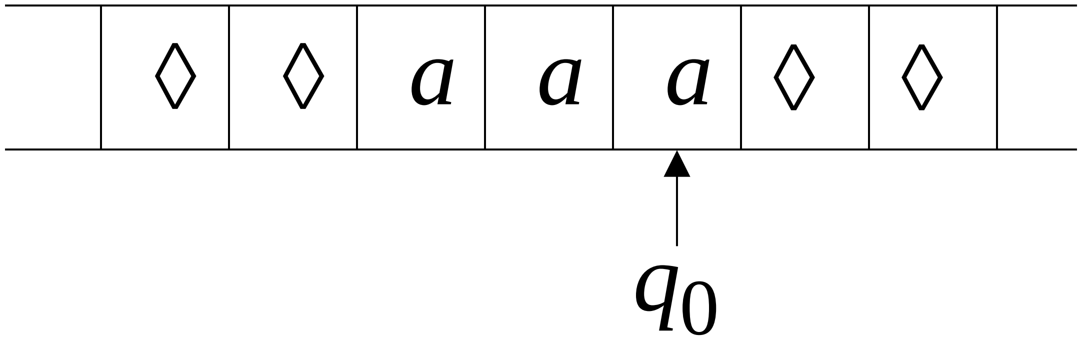

$a \rightarrow a, R$
$\Diamond \rightarrow \Diamond, L$
q_0
q_1

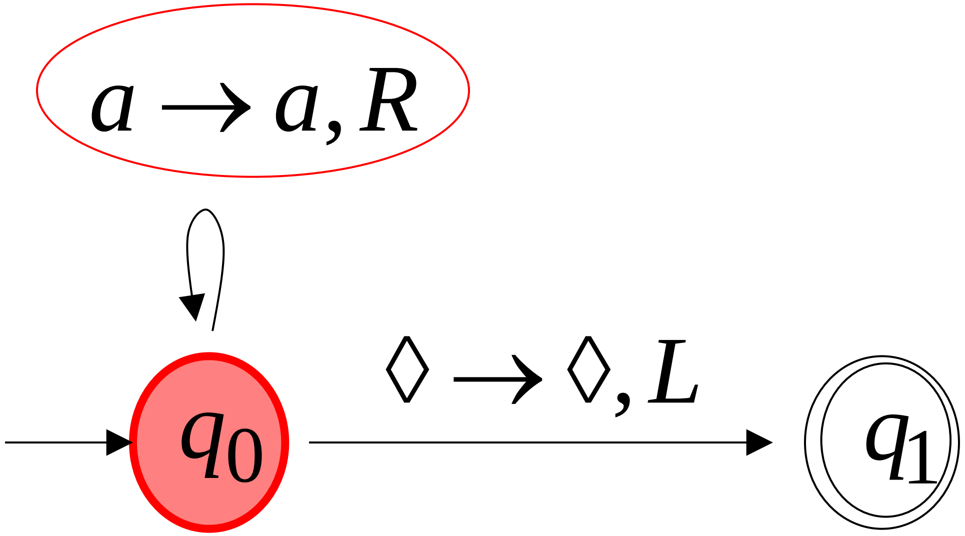
Time 1



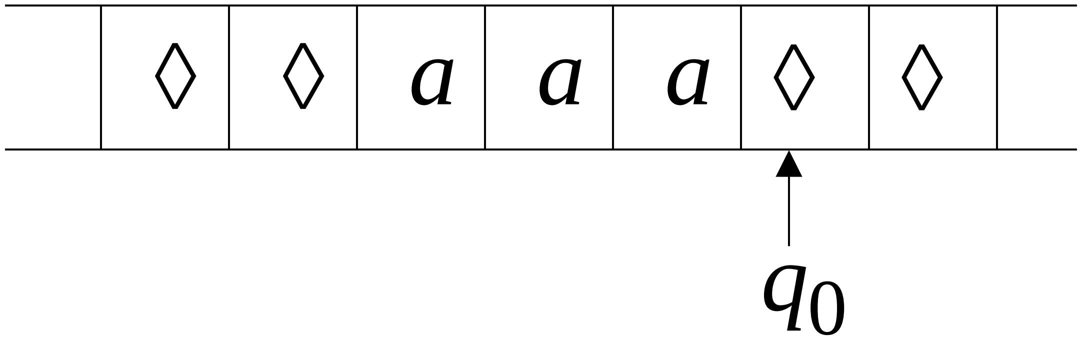


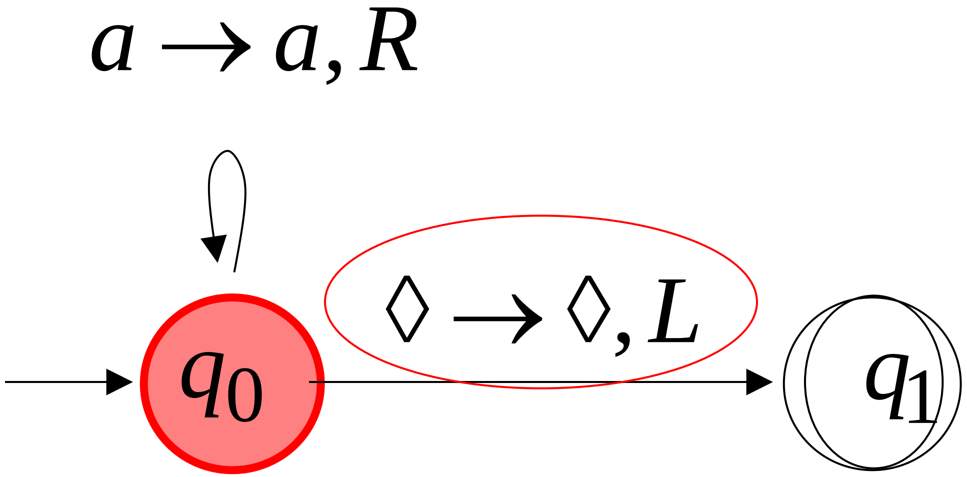
Time 2



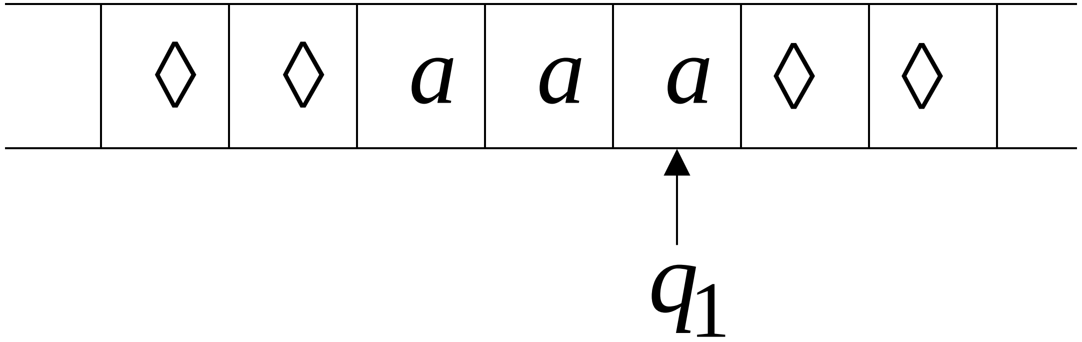


Time 3

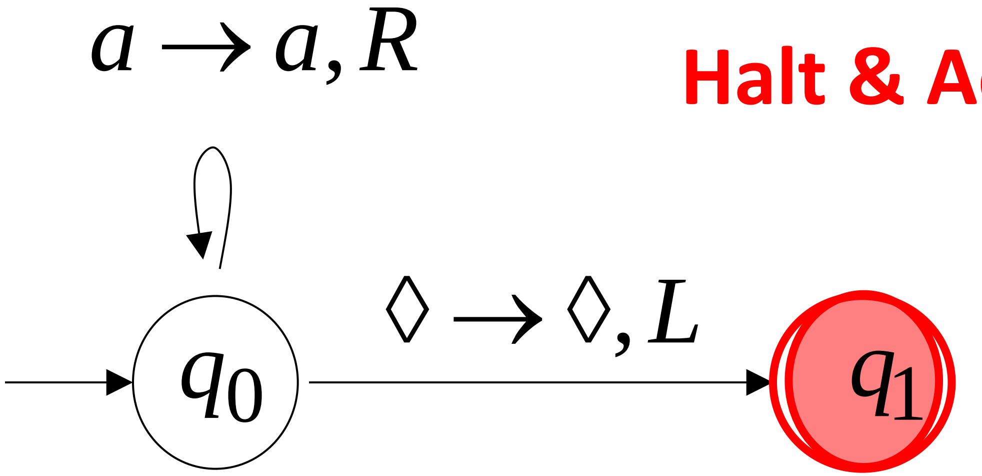




Time 4

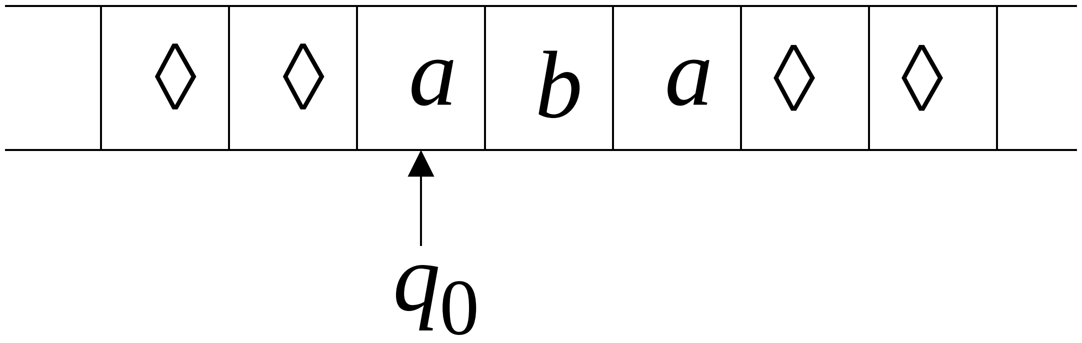


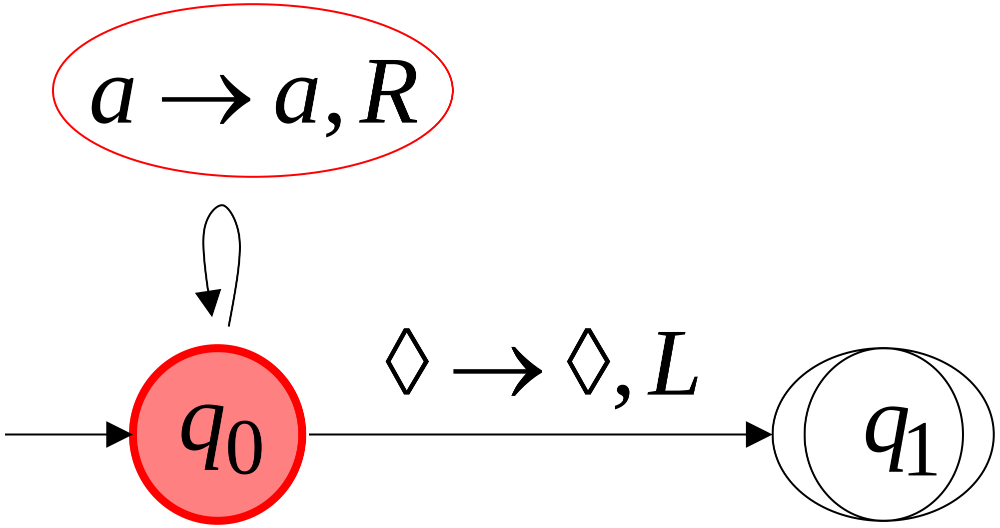
Halt & Accept



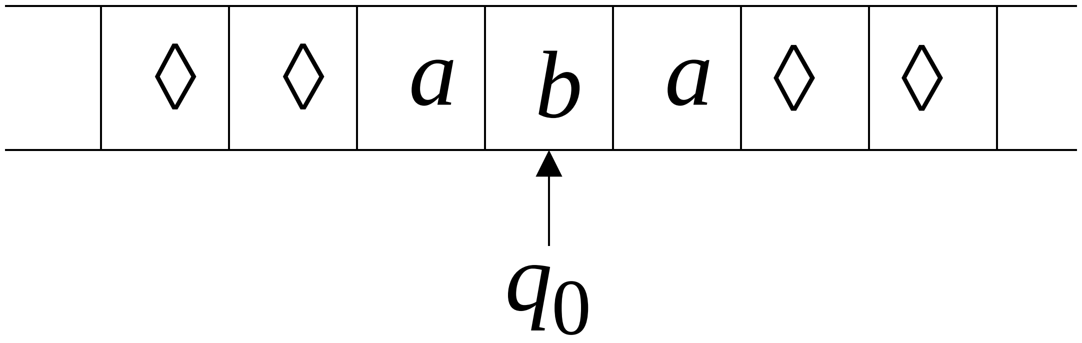
Rejection Example

Time 0



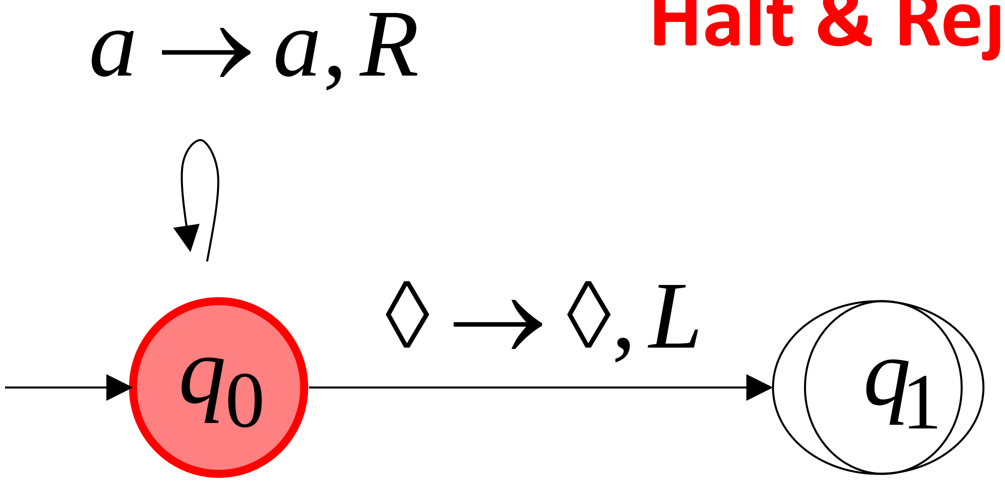


Time 1



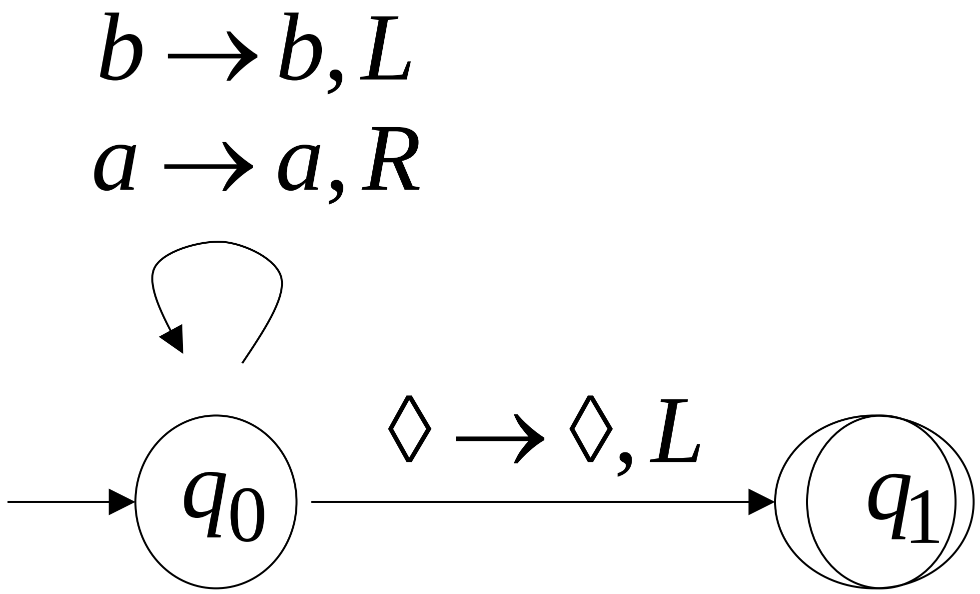
No possible Transition

Halt & Reject

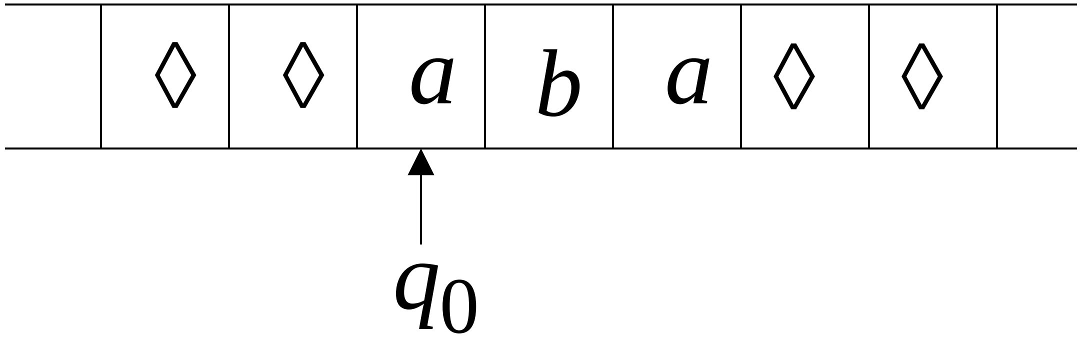


Infinite Loop Example

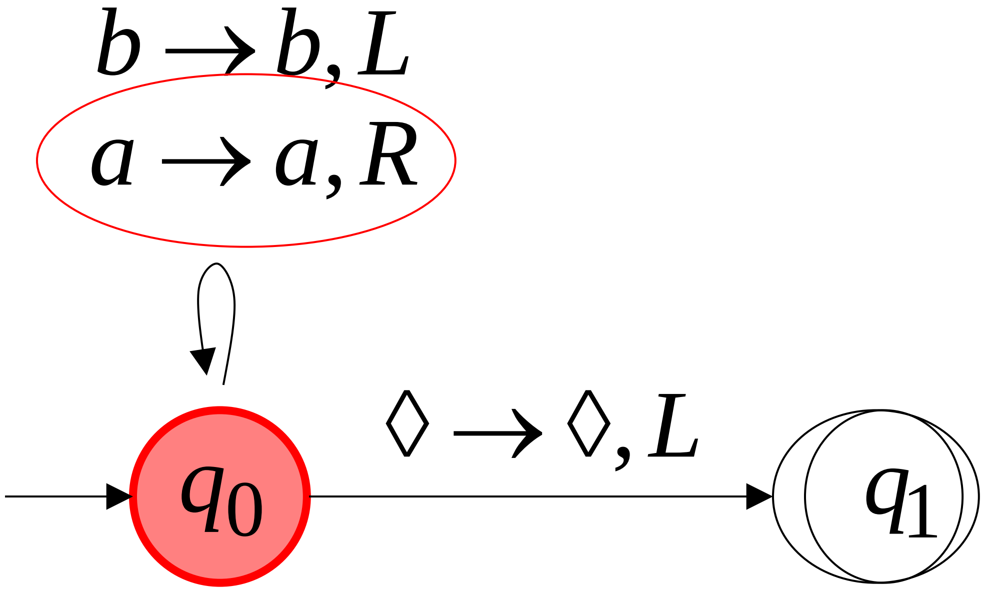
A Turing machine for language $aa^{*}+b(a+b)^{*}$



Time 0

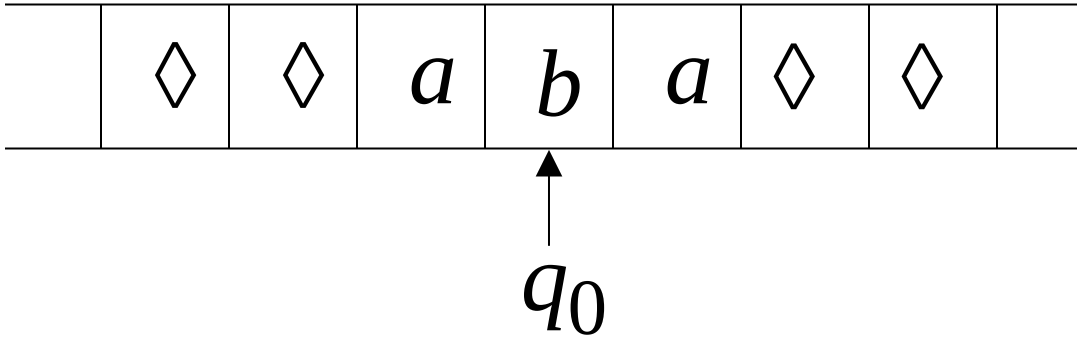


◊
◊
a
b
a
◊
◊
q_0

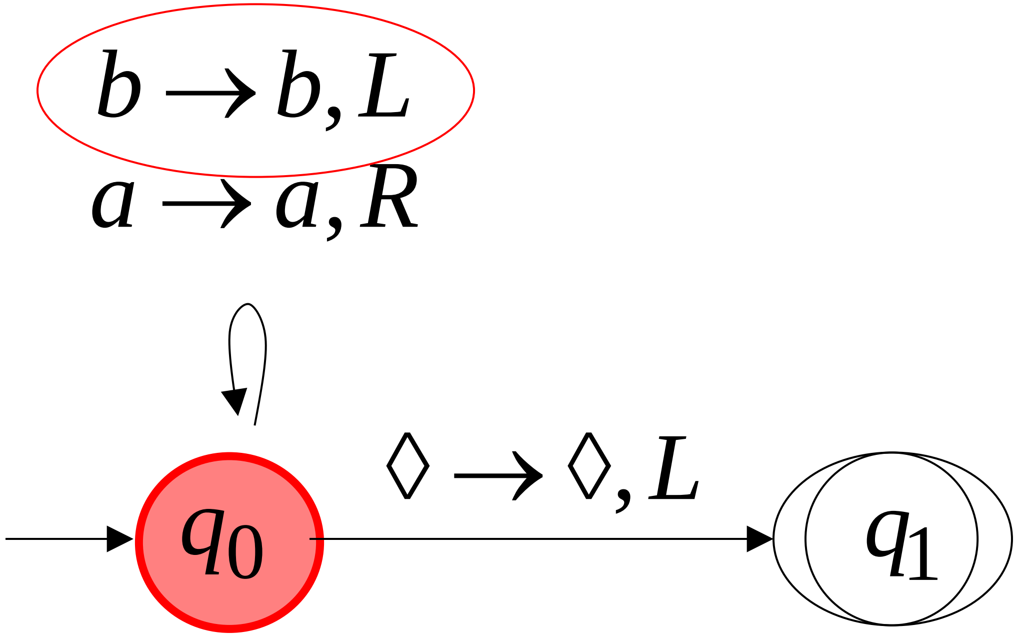


$b \rightarrow b, L$
$a \rightarrow a, R$
$\lozenge \rightarrow \lozenge, L$
q_0
q_1

Time 1

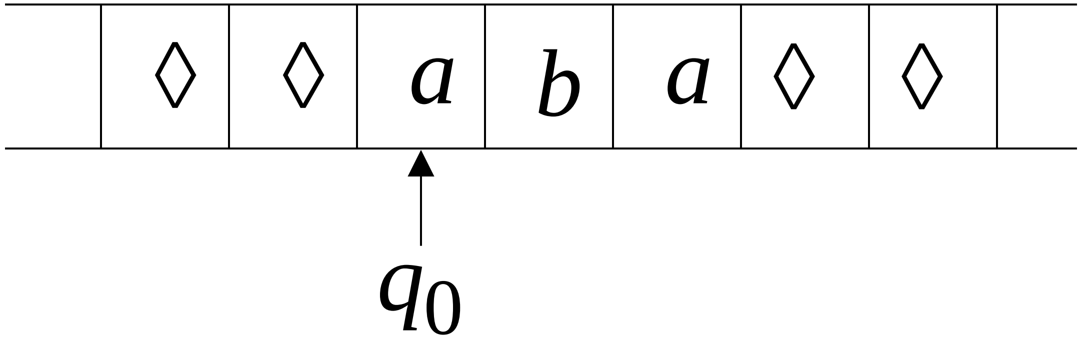


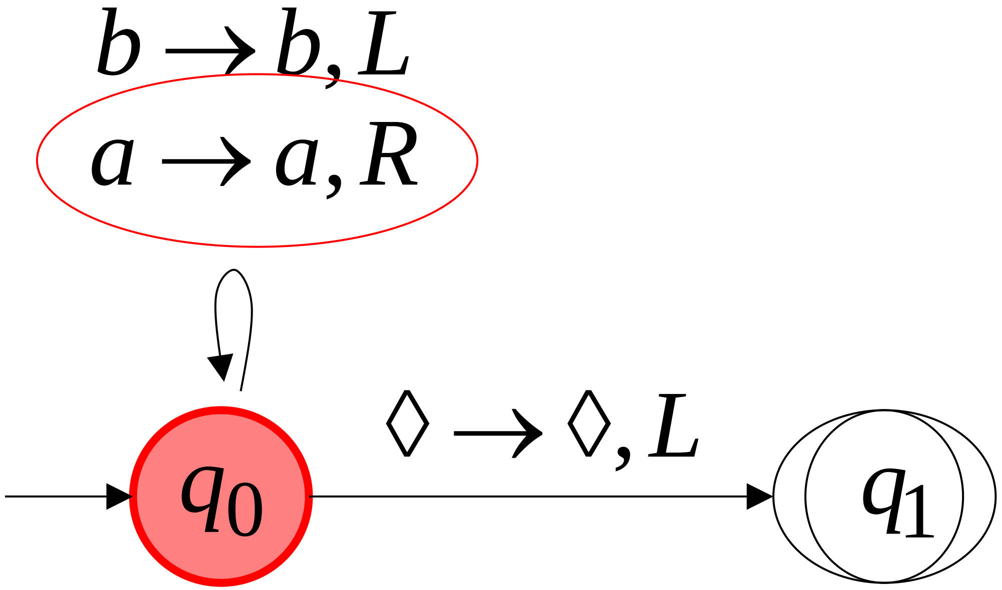
◊ ◊ a b a ◊ ◊
q_0

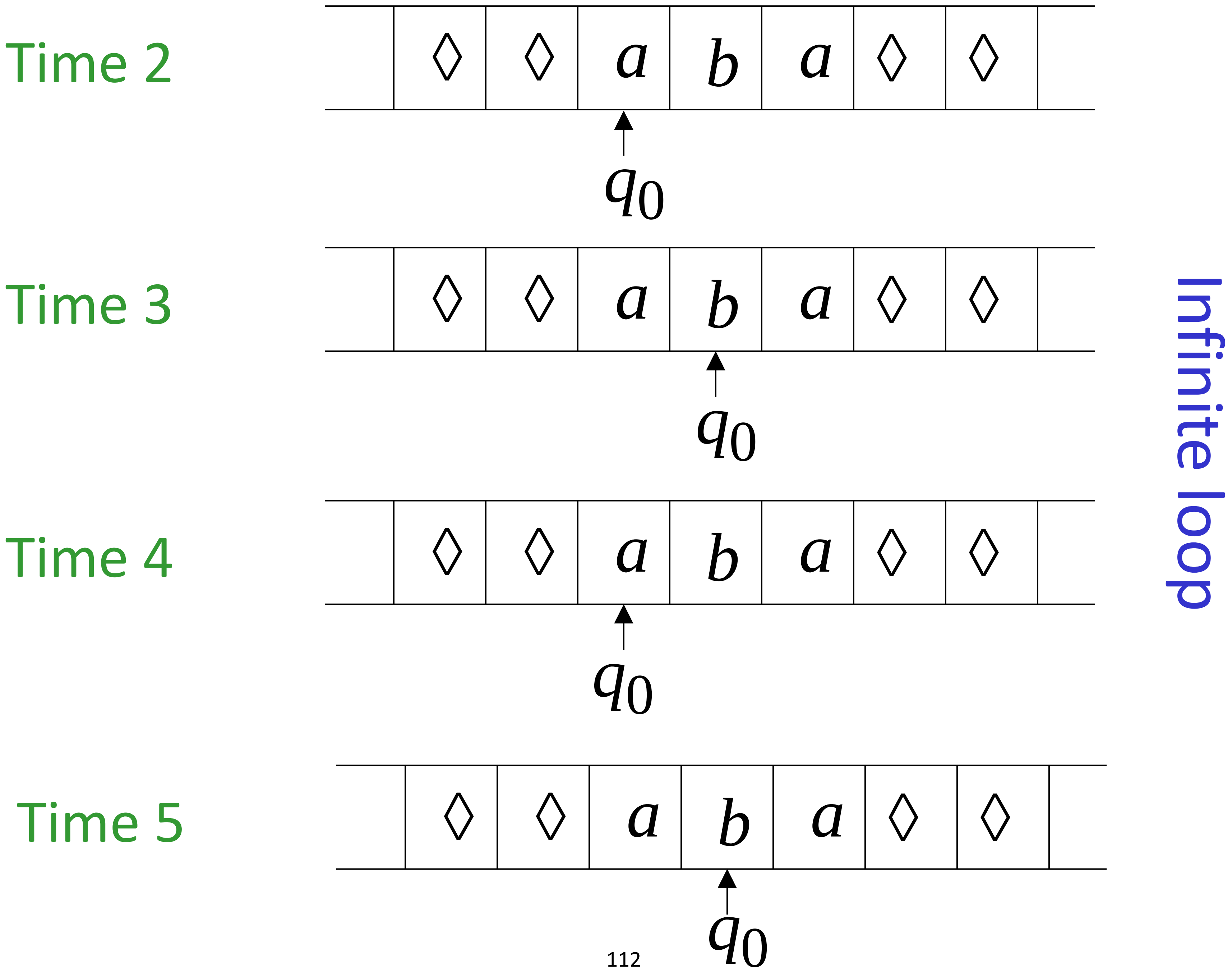


$b \rightarrow b, L$
$a \rightarrow a, R$
$\Diamond \rightarrow \Diamond, L$
q_0
q_1

Time 2


◊ ◊ a b a ◊ ◊
q_0


$b \rightarrow b, L$
$a \rightarrow a, R$
$\lozenge \rightarrow \lozenge, L$
q_0
q_1


Time 2
◊ ◊ a b a ◊ ◊
q0
Time 3
◊ ◊ a b a ◊ ◊
q0
Time 4
◊ ◊ a b a ◊ ◊
q0
Time 5
◊ ◊ a b a ◊ ◊
q0
Infinite loop

Because of the infinite loop:

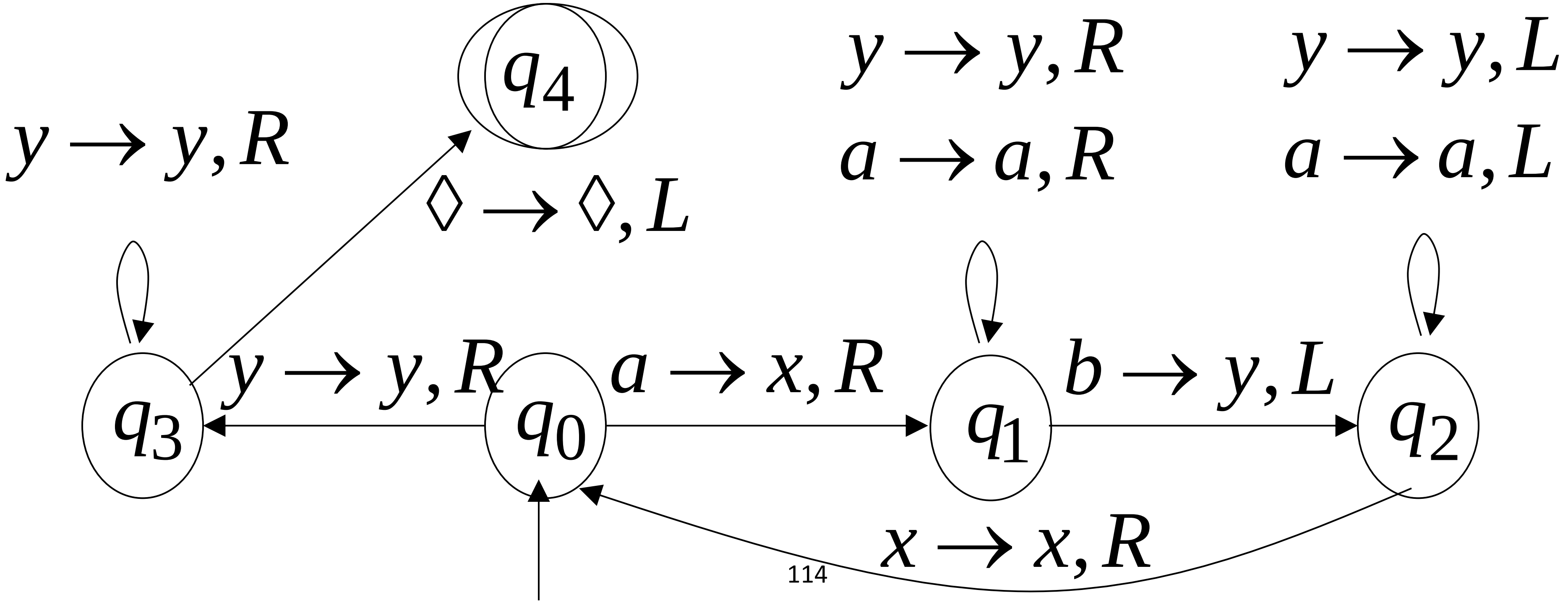
- The final state cannot be reached

- The machine never halts

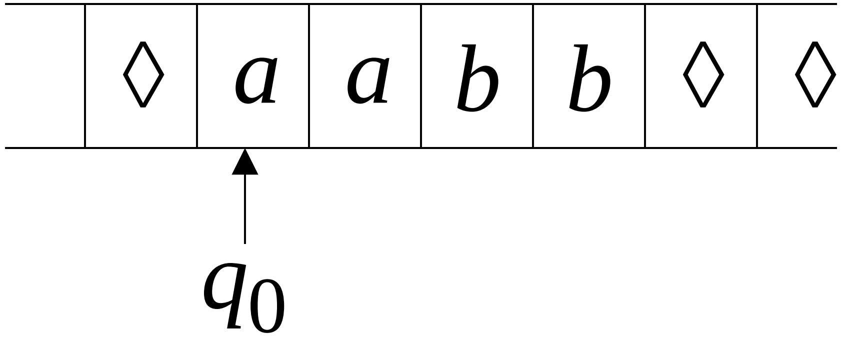
- The input is not accepted

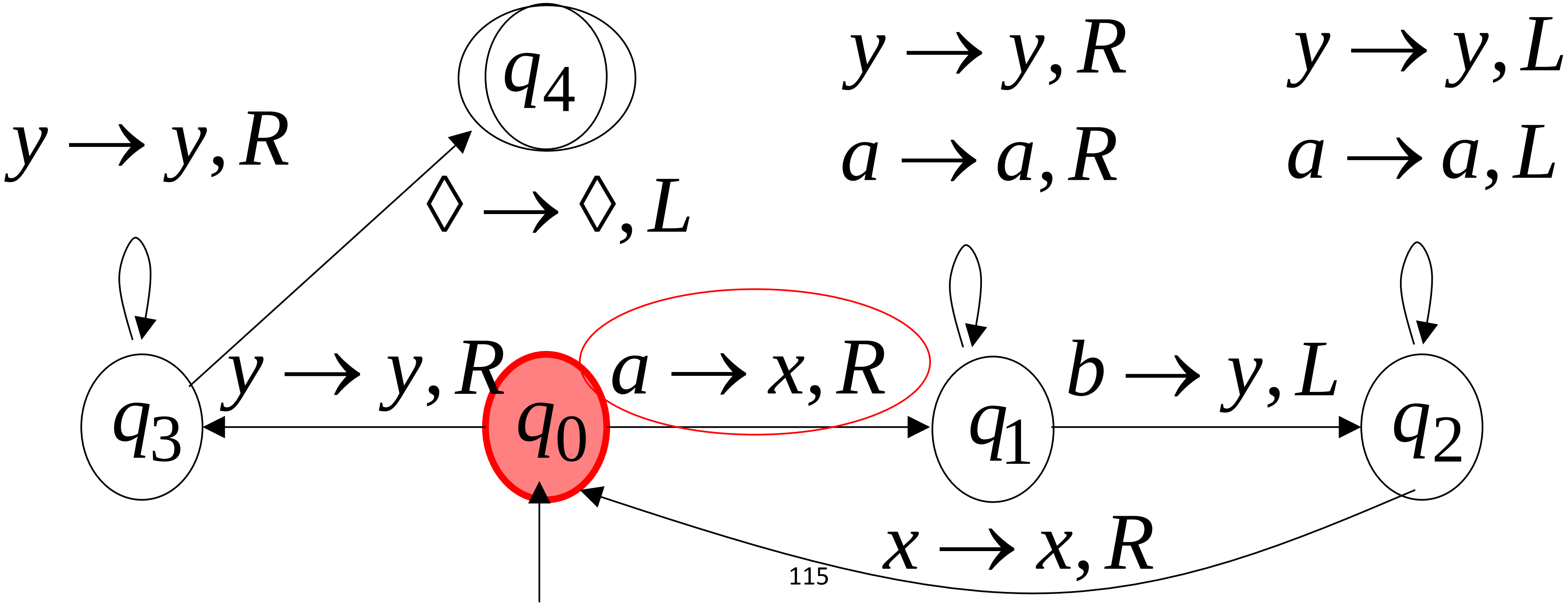
Another Turing Machine Example

Turing machine for the language $\{a^n b^n\}$

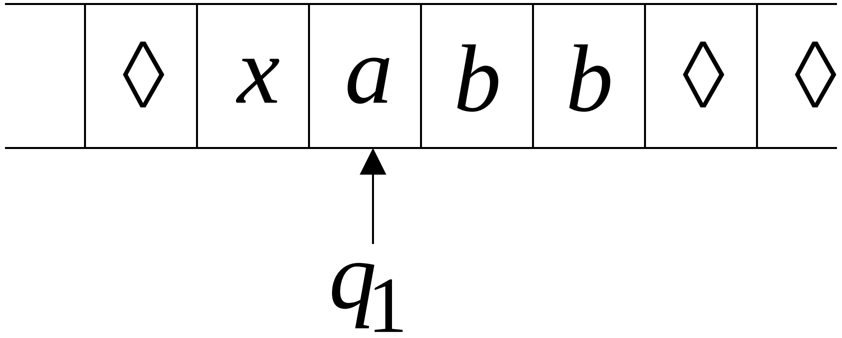


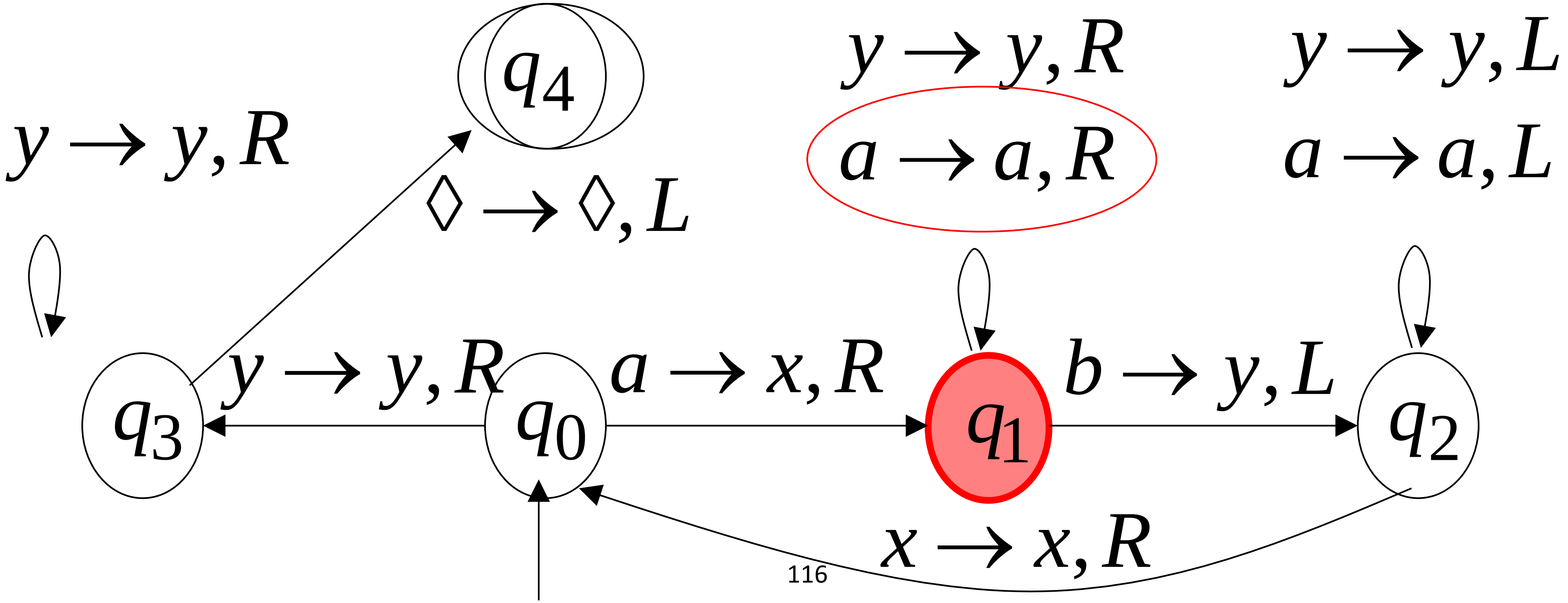
Time 0


◊
a
a
b
b
◊
◊
q_0

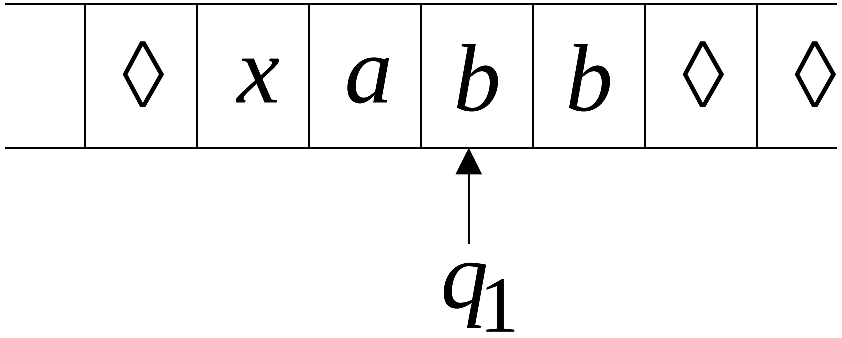

q_4
$y \to y, R$
$\lozenge \to \lozenge, L$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
q_3
$y \to y, R$
q_0
$a \to x, R$
q_1
$b \to y, L$
q_2
$x \to x, R$

Time 1

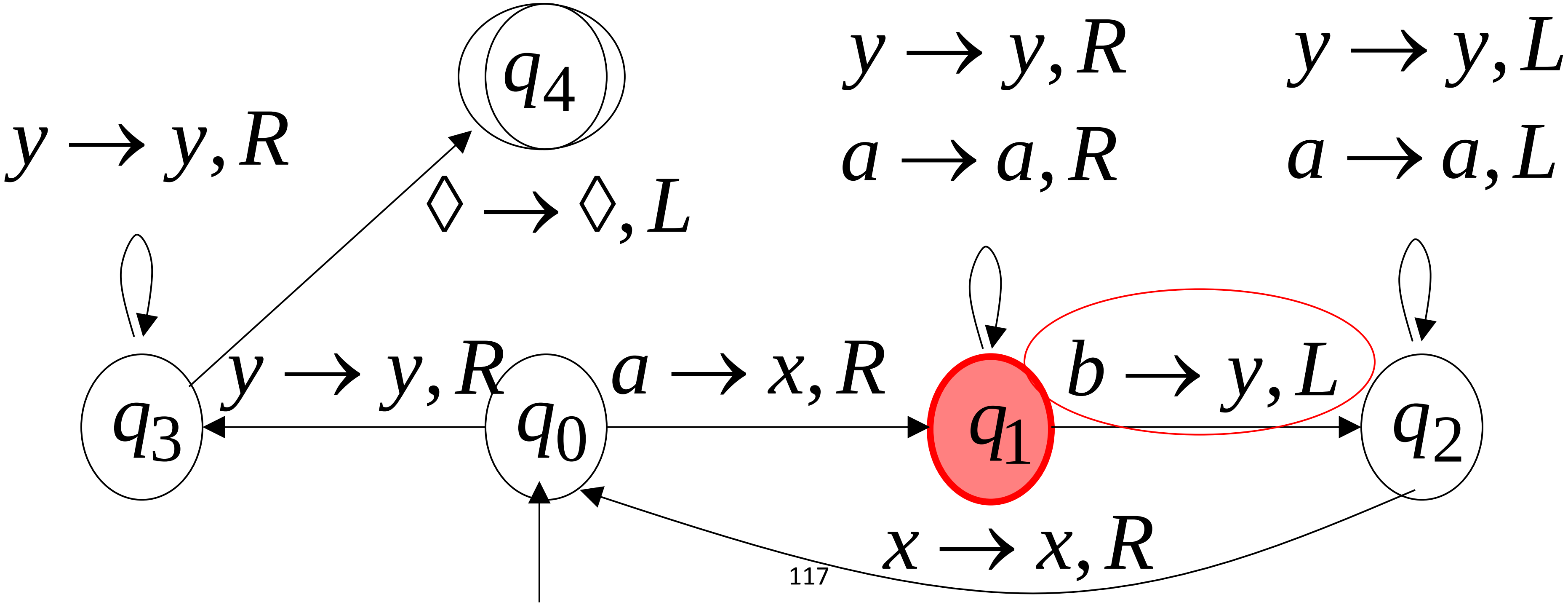

◊ x a b b ◊ ◊
q_1


q_4
$y \rightarrow y, R$
$\lozenge \rightarrow \lozenge, L$
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
q_3
$y \rightarrow y, R$
q_0
$a \rightarrow x, R$
q_1
$b \rightarrow y, L$
q_2
$x \rightarrow x, R$

Time 2

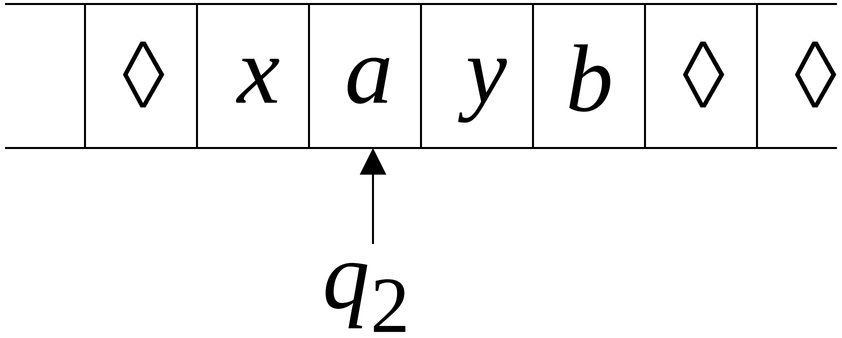


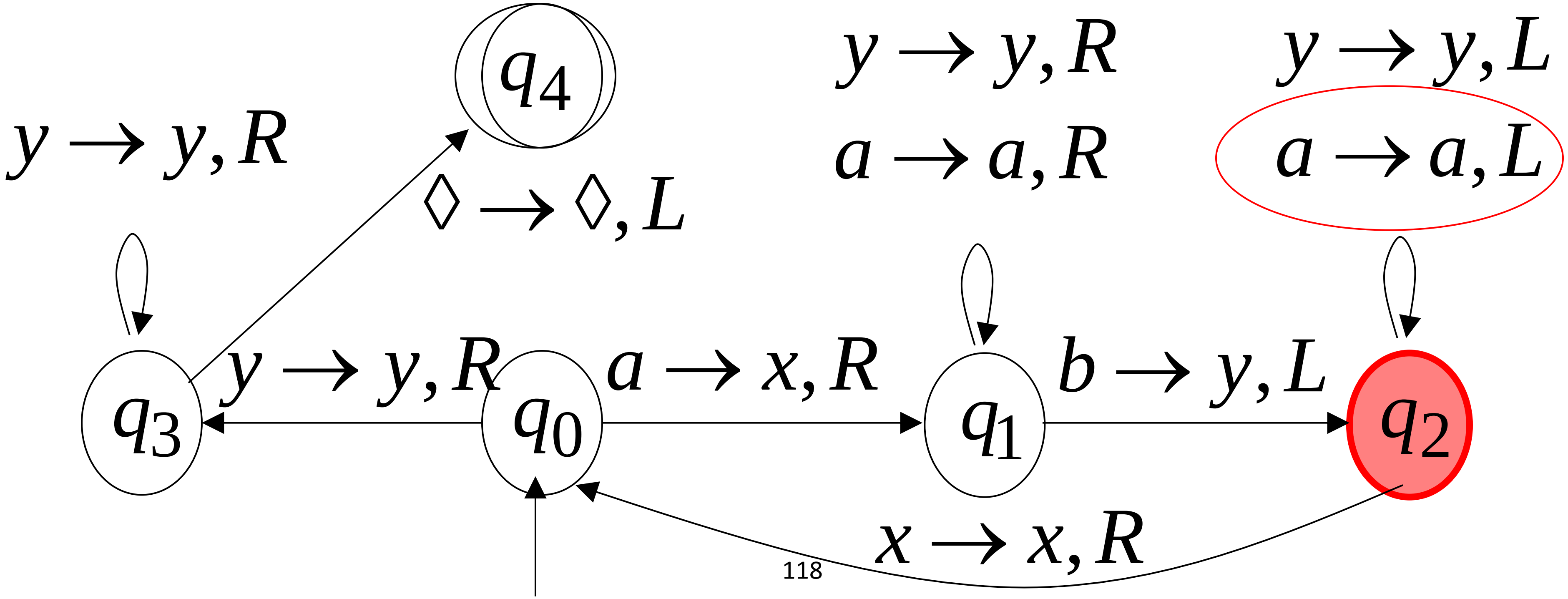
$\Diamond$ x a b b $\Diamond$ $\Diamond$
q_1



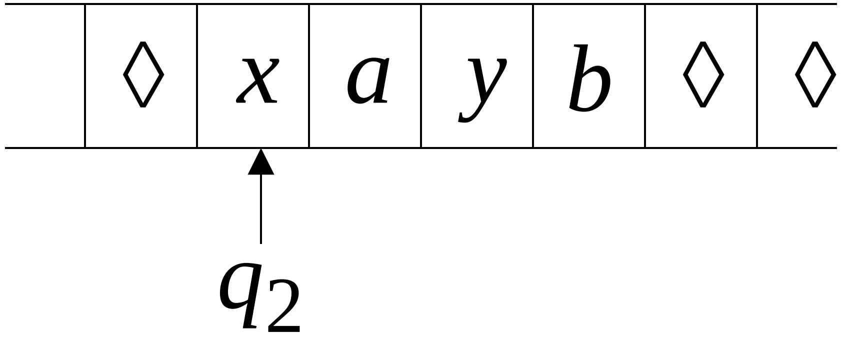
q_4
$y \rightarrow y, R$
$\Diamond \rightarrow \Diamond, L$
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
q_3
$y \rightarrow y, R$
q_0
$a \rightarrow x, R$
q_1
$b \rightarrow y, L$
q_2
$x \rightarrow x, R$

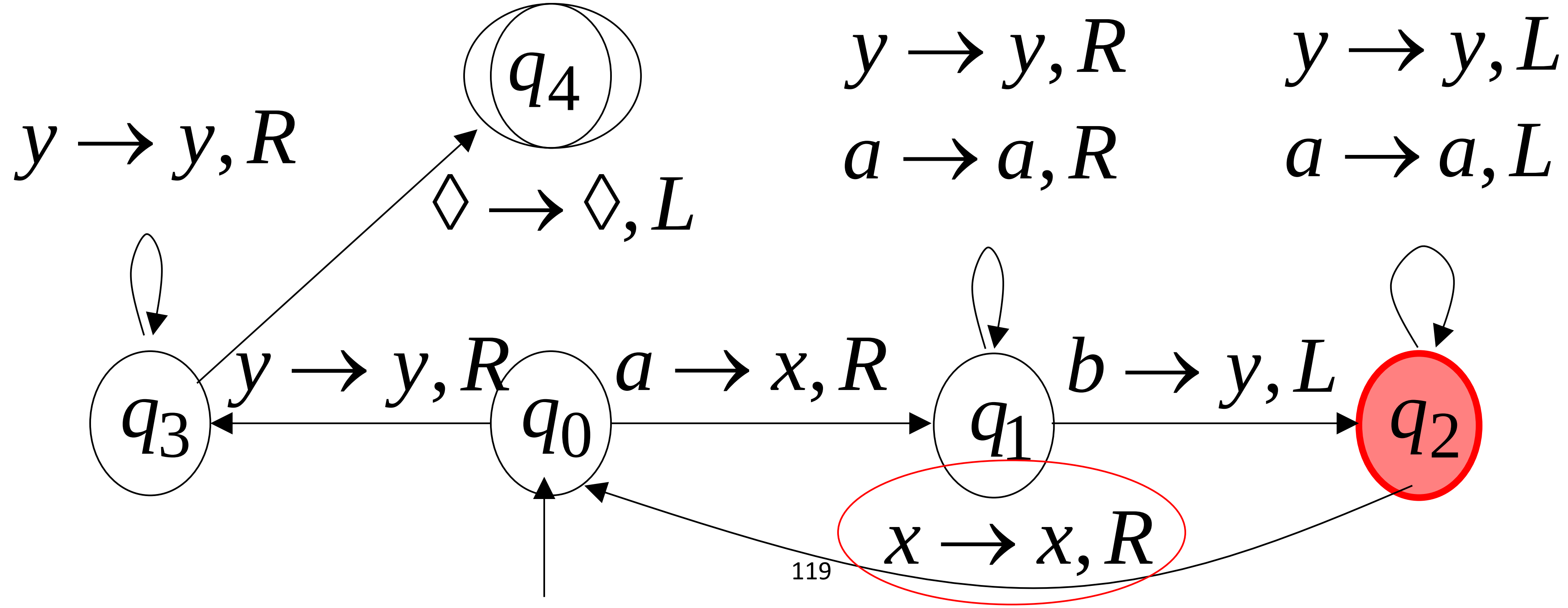
Time 3


◊ x a y b ◊ ◊
q_2

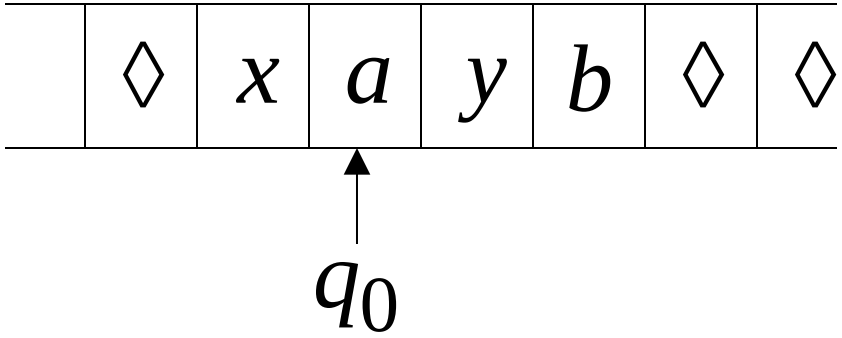

q_4
$y \to y, R$
$\lozenge \to \lozenge, L$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
q_3
$y \to y, R$
q_0
$a \to x, R$
q_1
$b \to y, L$
q_2
$x \to x, R$

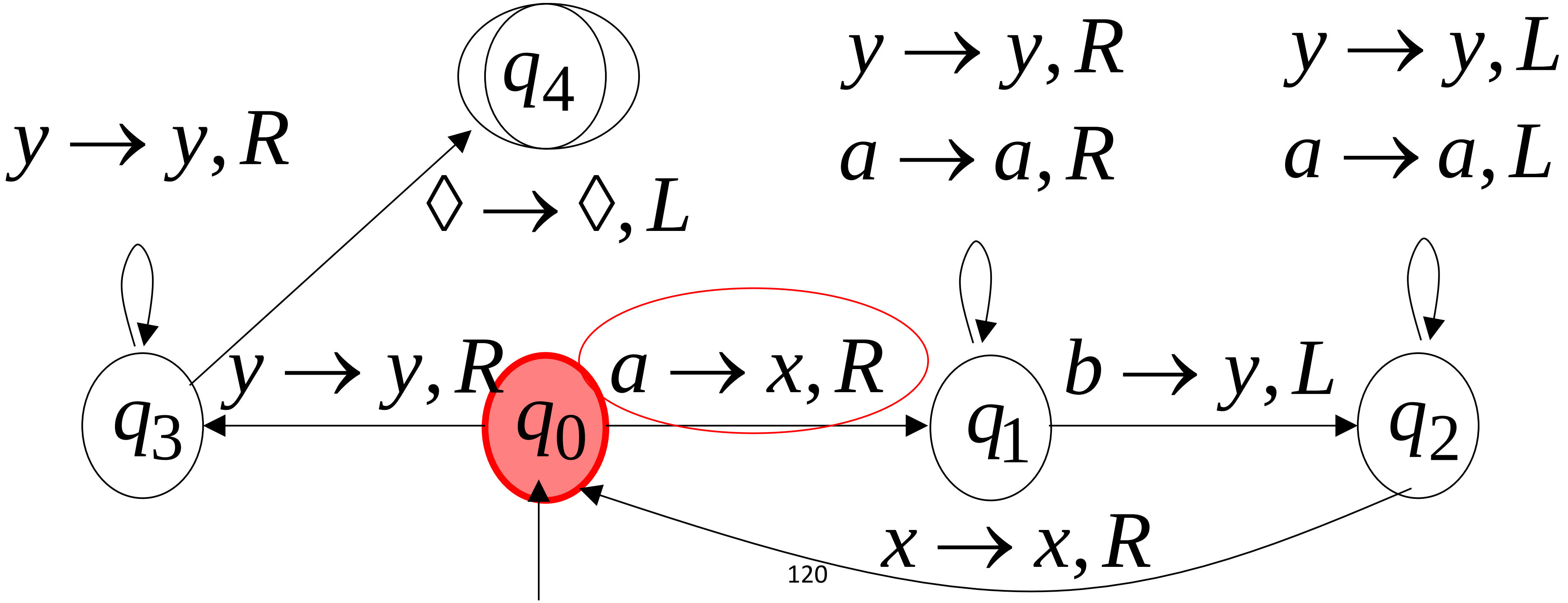
Time 4



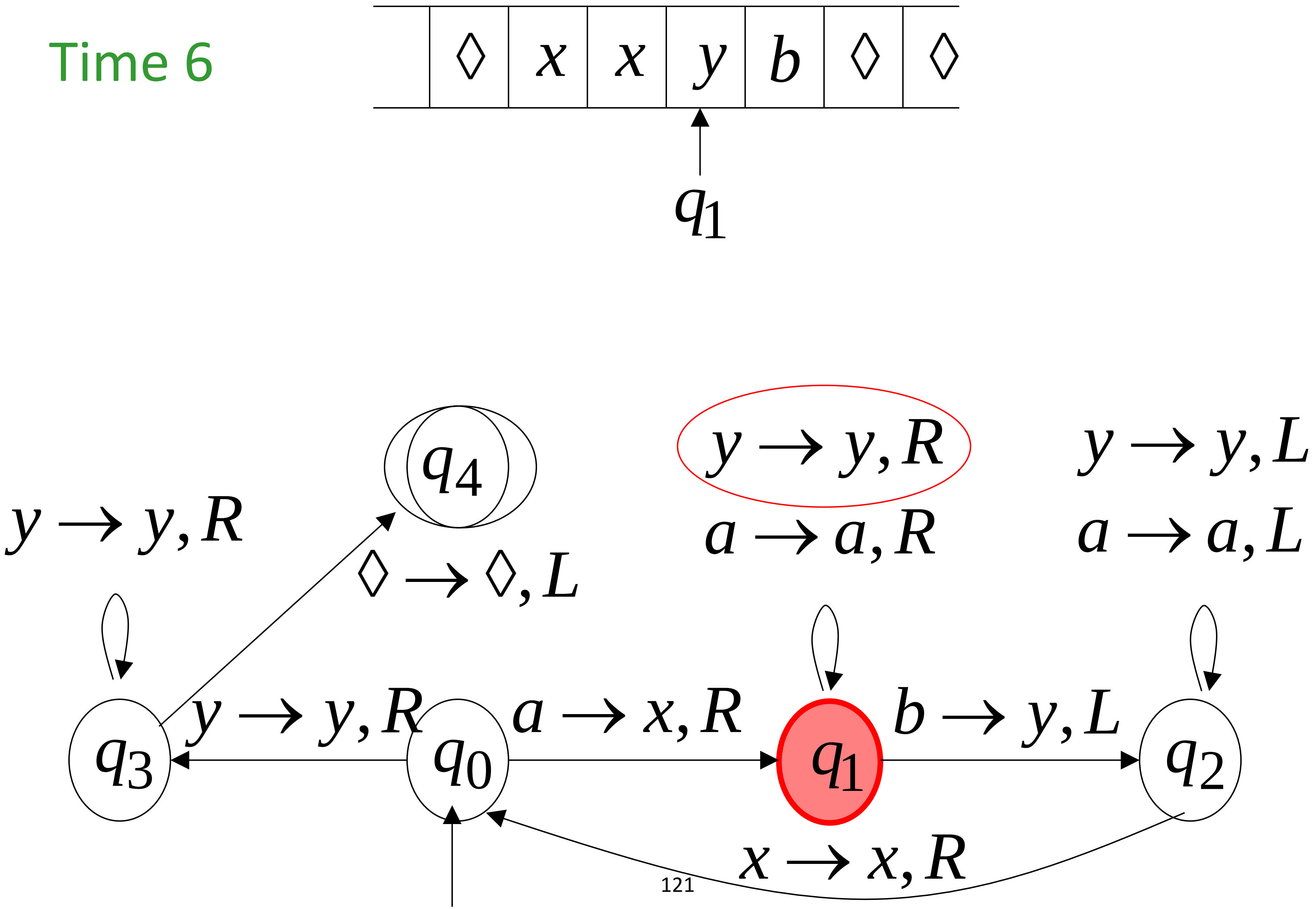


Time 5

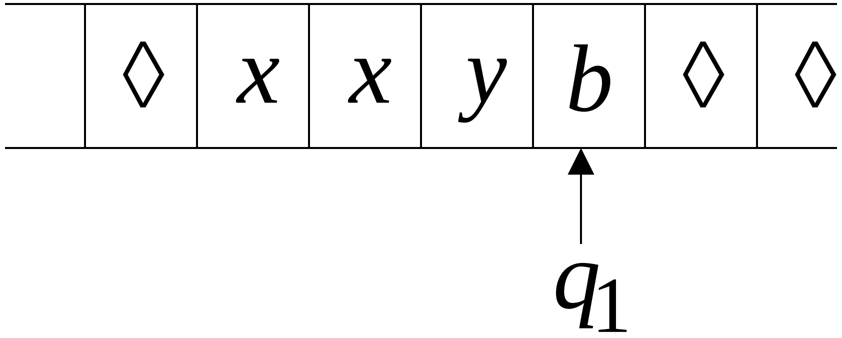

◊
x
a
y
b
◊
◊
q_0

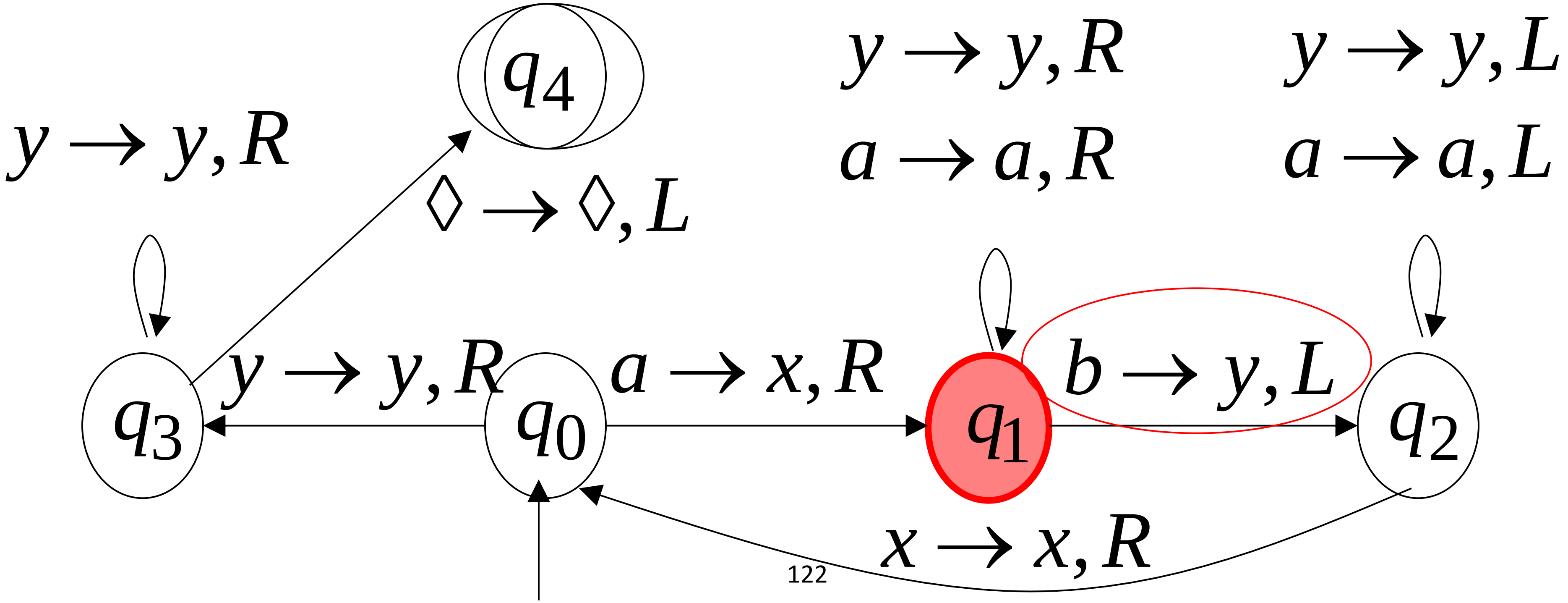

q_4
$y \to y, R$
$\Diamond \to \Diamond, L$
$y \to y, R$
$a \to a, R$
$y \to y, L$
$a \to a, L$
q_3
$y \to y, R$
q_0
$a \to x, R$
q_1
$b \to y, L$
q_2
$x \to x, R$

Time 6

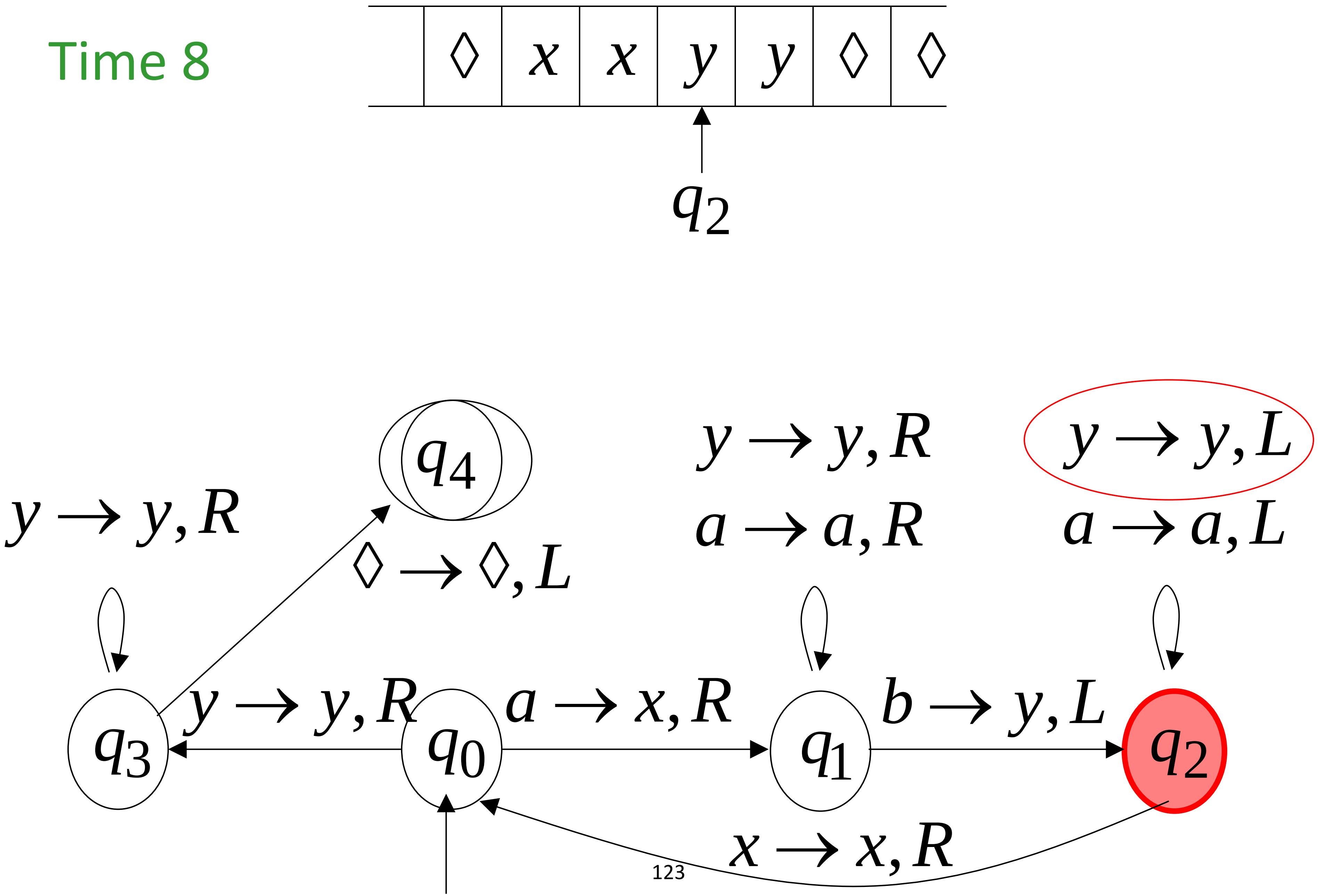

◊
x
x
y
b
◊
◊
q_1
q_4
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
$y \rightarrow y, R$
$\Diamond \rightarrow \Diamond, L$
$y \rightarrow y, R$
$a \rightarrow x, R$
$b \rightarrow y, L$
q_3
q_0
q_1
q_2
$x \rightarrow x, R$

Time 7

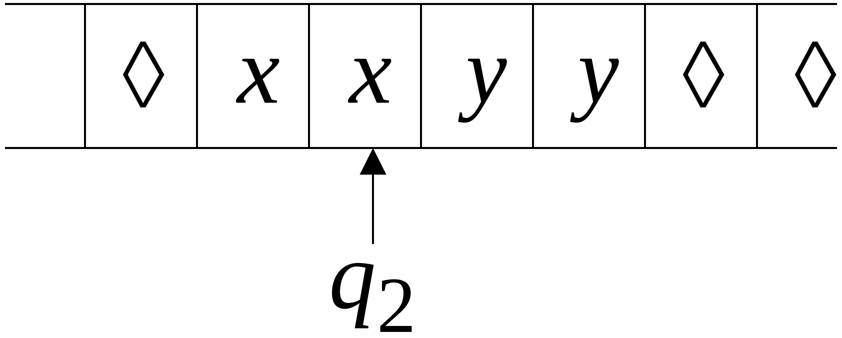

◊ x x y b ◊ ◊
q_1

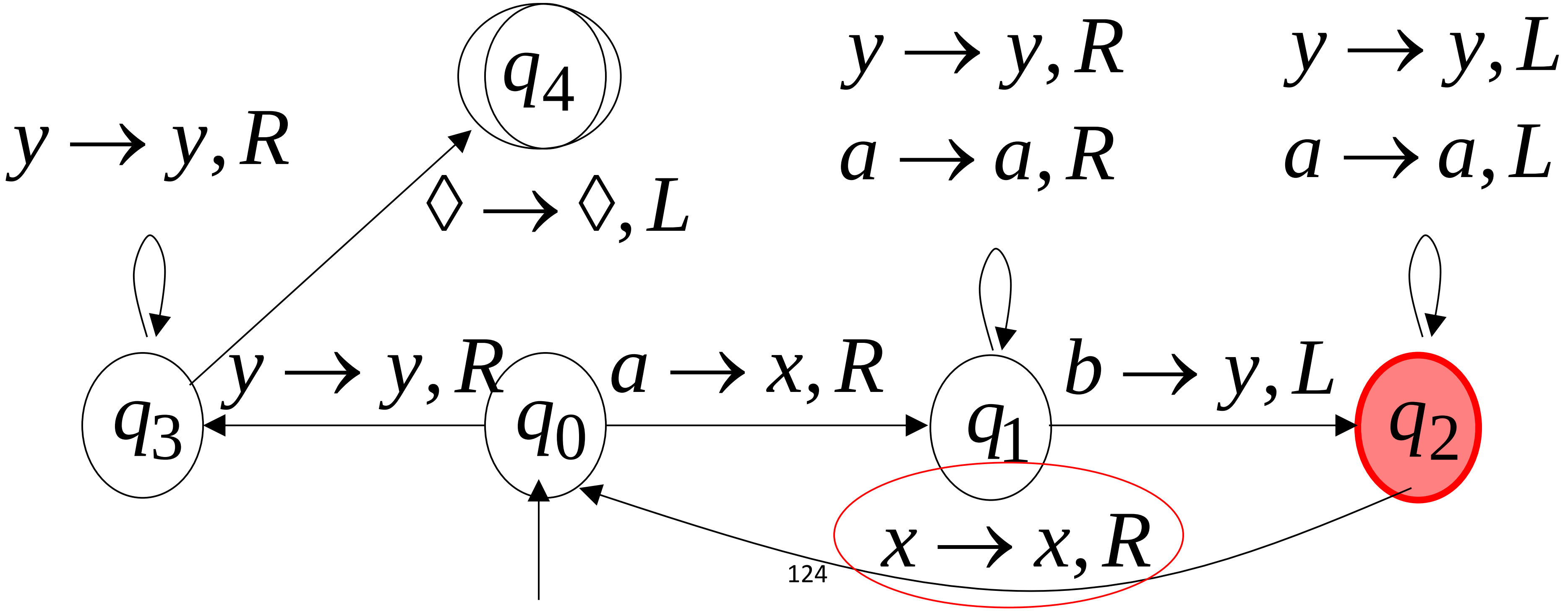

q_4
$y \rightarrow y, R$
$\lozenge \rightarrow \lozenge, L$
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
q_3
$y \rightarrow y, R$
q_0
$a \rightarrow x, R$
q_1
$b \rightarrow y, L$
q_2
$x \rightarrow x, R$

Time 8

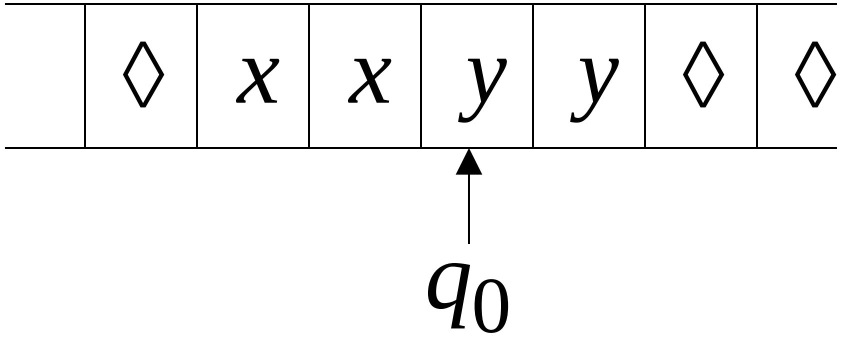

◊
x
x
y
y
◊
◊
q_2
$y \rightarrow y, R$
q_4
$\lozenge \rightarrow \lozenge, L$
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
q_3
$y \rightarrow y, R$
q_0
$a \rightarrow x, R$
q_1
$b \rightarrow y, L$
q_2
$x \rightarrow x, R$

Time 9

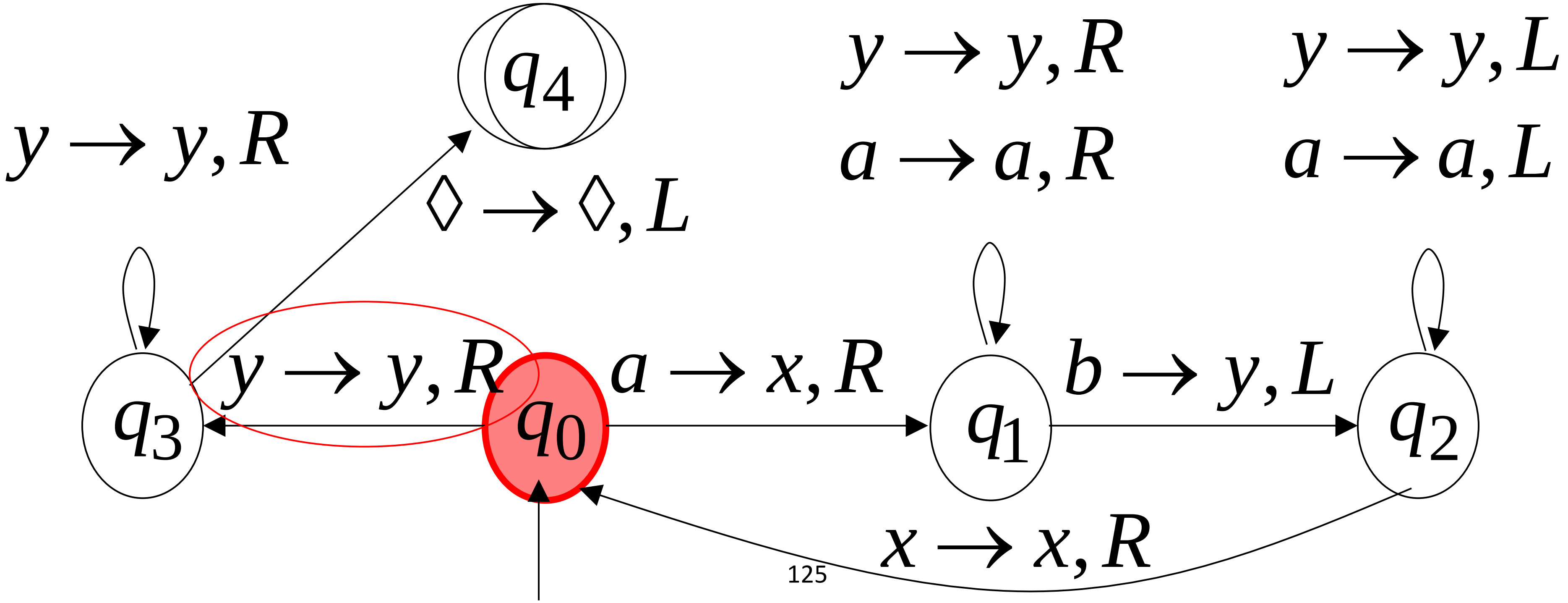




Time 10

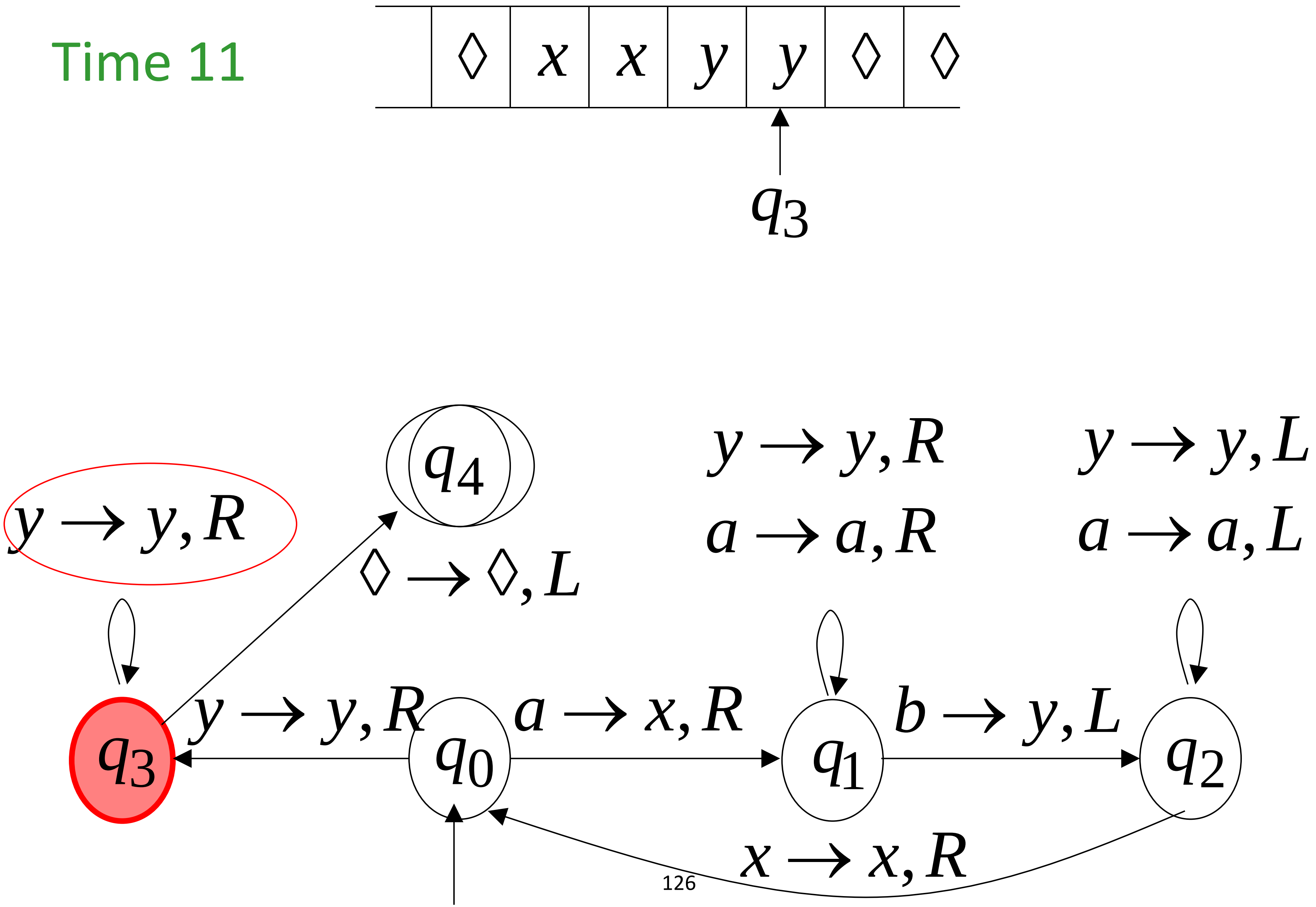


◊
x
x
y
y
◊
◊
q_0



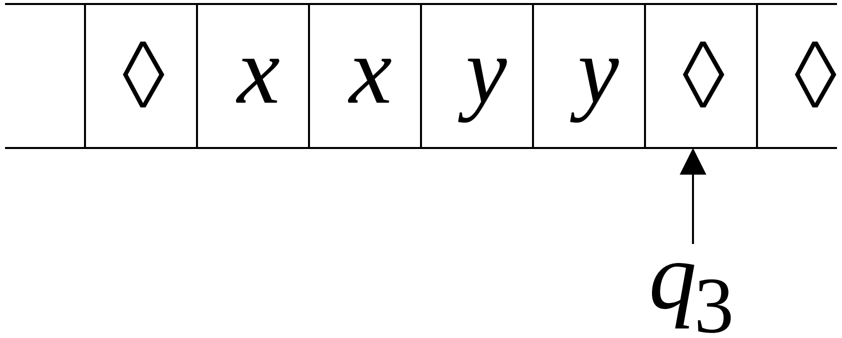
q_4
$y \rightarrow y, R$
$\Diamond \rightarrow \Diamond, L$
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
q_3
$y \rightarrow y, R$
q_0
$a \rightarrow x, R$
q_1
$b \rightarrow y, L$
q_2
$x \rightarrow x, R$

Time 11

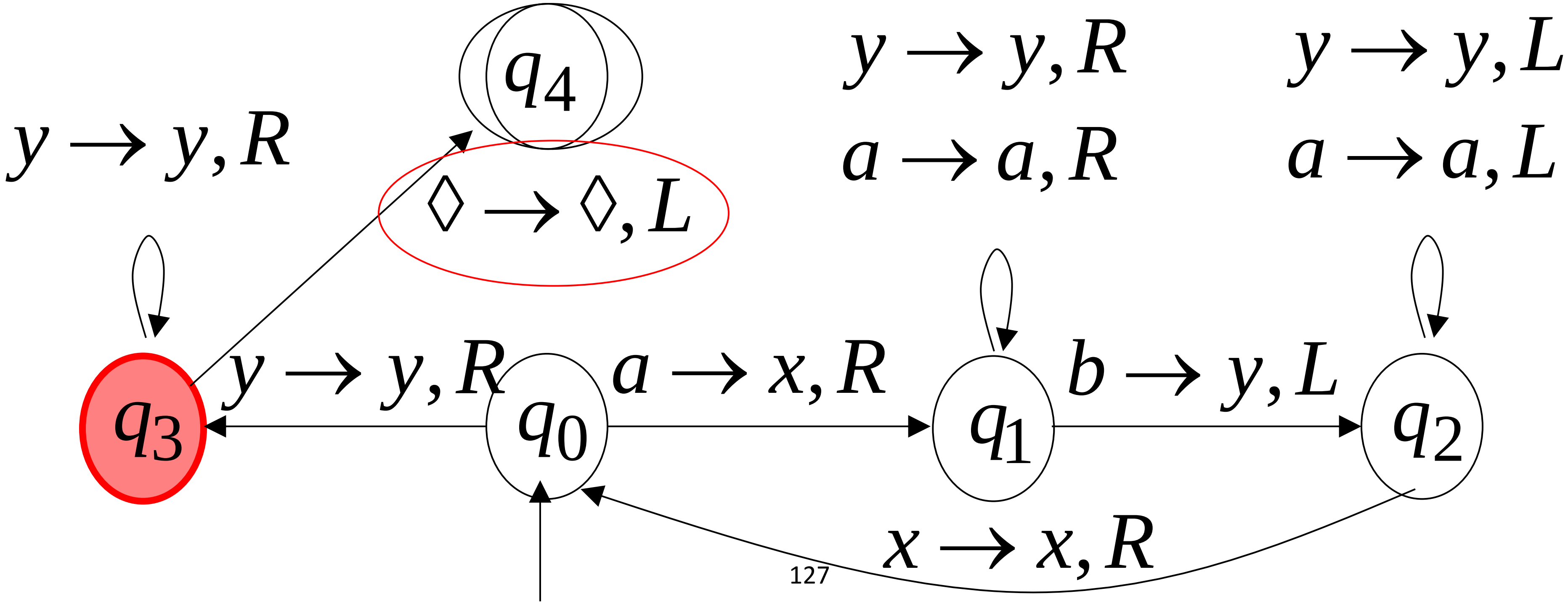


◊
x
x
y
y
◊
◊
q_3
$y \rightarrow y, R$
q_4
$\Diamond \rightarrow \Diamond, L$
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
q_3
$y \rightarrow y, R$
q_0
$a \rightarrow x, R$
q_1
$b \rightarrow y, L$
q_2
$x \rightarrow x, R$

Time 12

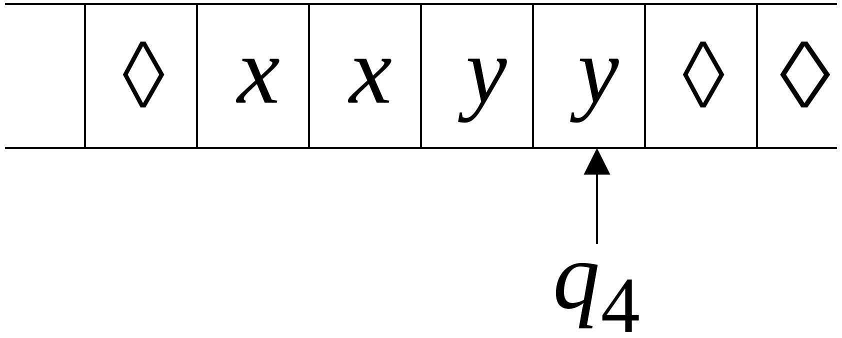


◊
x
x
y
y
◊
◊
q_3

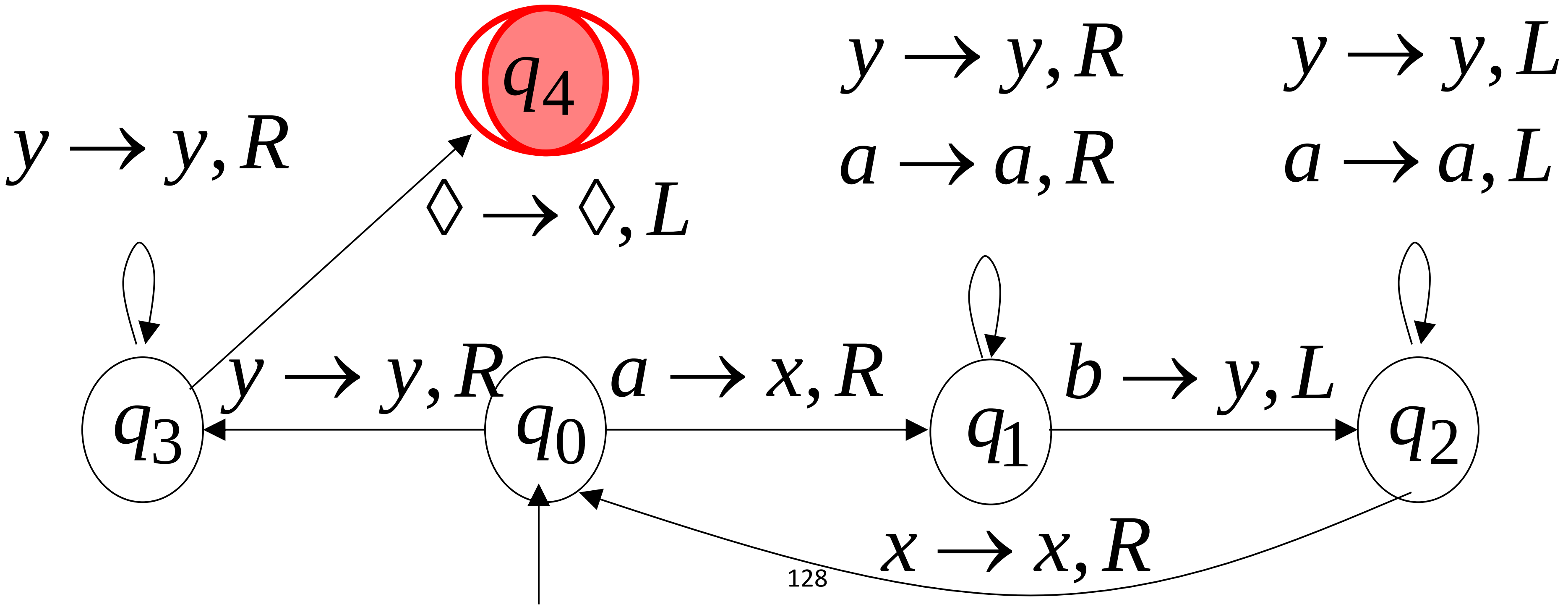


q_4
$y \rightarrow y, R$
$\Diamond \rightarrow \Diamond, L$
$y \rightarrow y, R$
$a \rightarrow a, R$
$y \rightarrow y, L$
$a \rightarrow a, L$
q_3
$y \rightarrow y, R$
q_0
$a \rightarrow x, R$
q_1
$b \rightarrow y, L$
q_2
$x \rightarrow x, R$

Time 13



Halt & Accept



Observation:

If we modify the
machine for the language $\{a^n b^n\}$

we can easily construct
a machine for the language $\{a^n b^n c^n\}$

Standard Turing Machine

The machine we described is the standard:

- Deterministic

- Infinite tape in both directions

- Tape is the input/output file

Outline

- Last week
- Conversions around Context-free Languages
- Deterministic PDA(DPDA)
- Turing Machines
- Review

Regular Languages

Regular Expressions

$$aaab^*a + b^*a$$

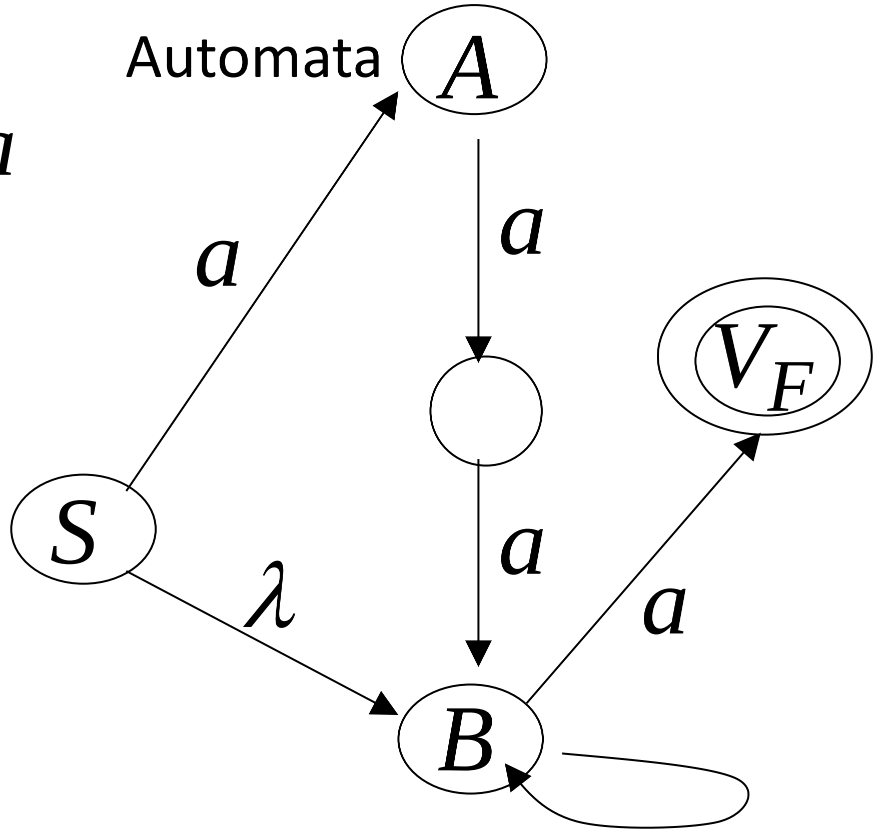
Regular Grammars

$$S \to aA \mid B$$

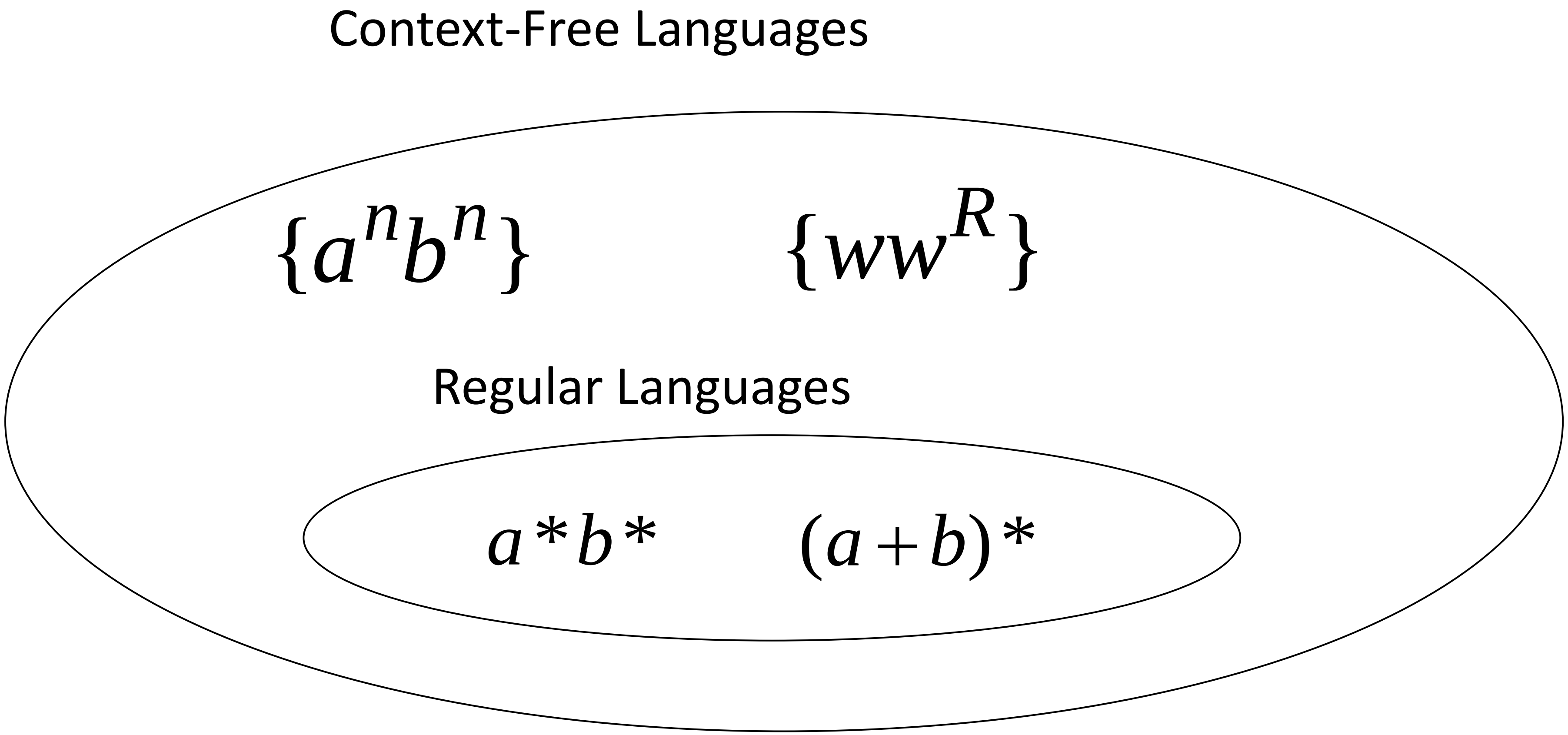
$$A \to aa\ B$$

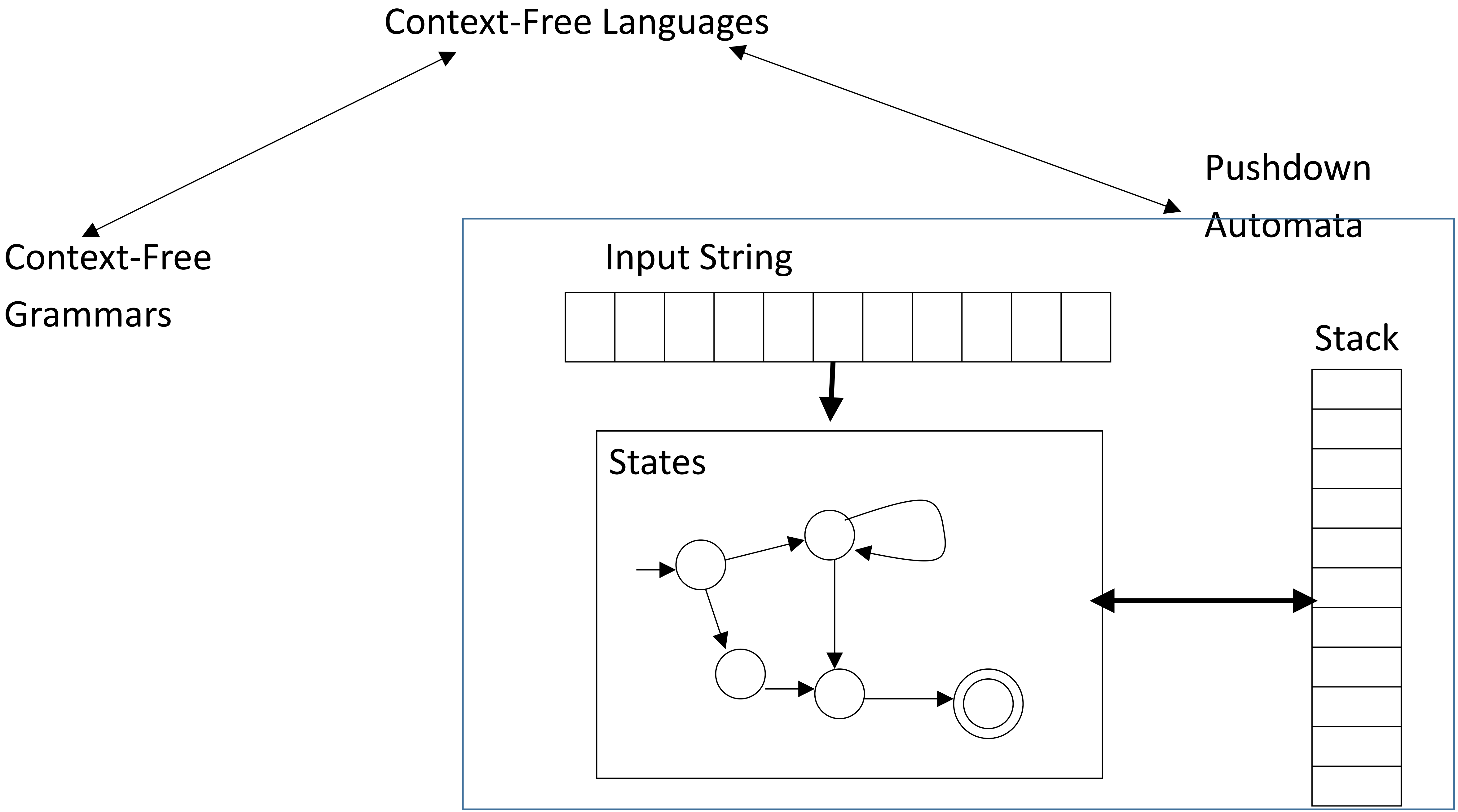
$$B \to bB \mid a$$

Finite State Automata



Context-Free and Regular Languages



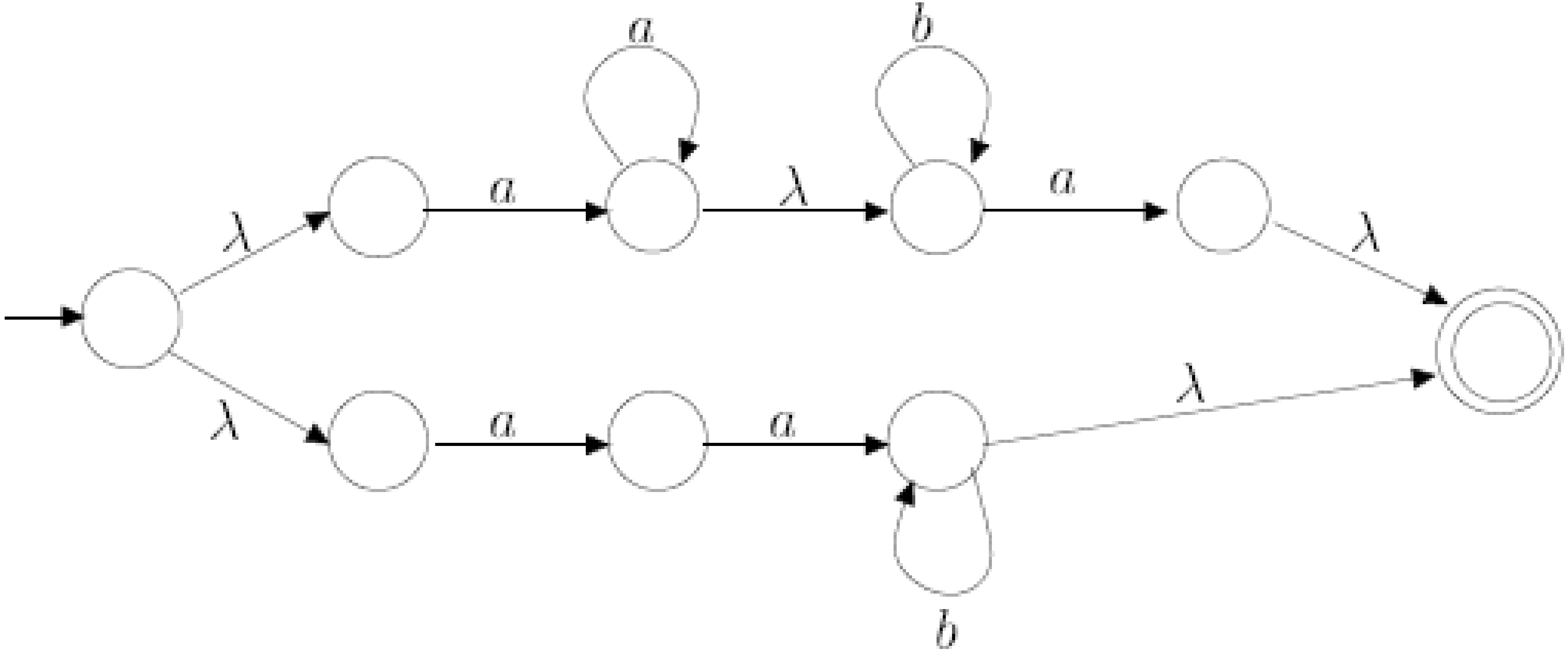

Context-Free Languages
Context-Free Grammars
Pushdown Automata
Input String
Stack
States

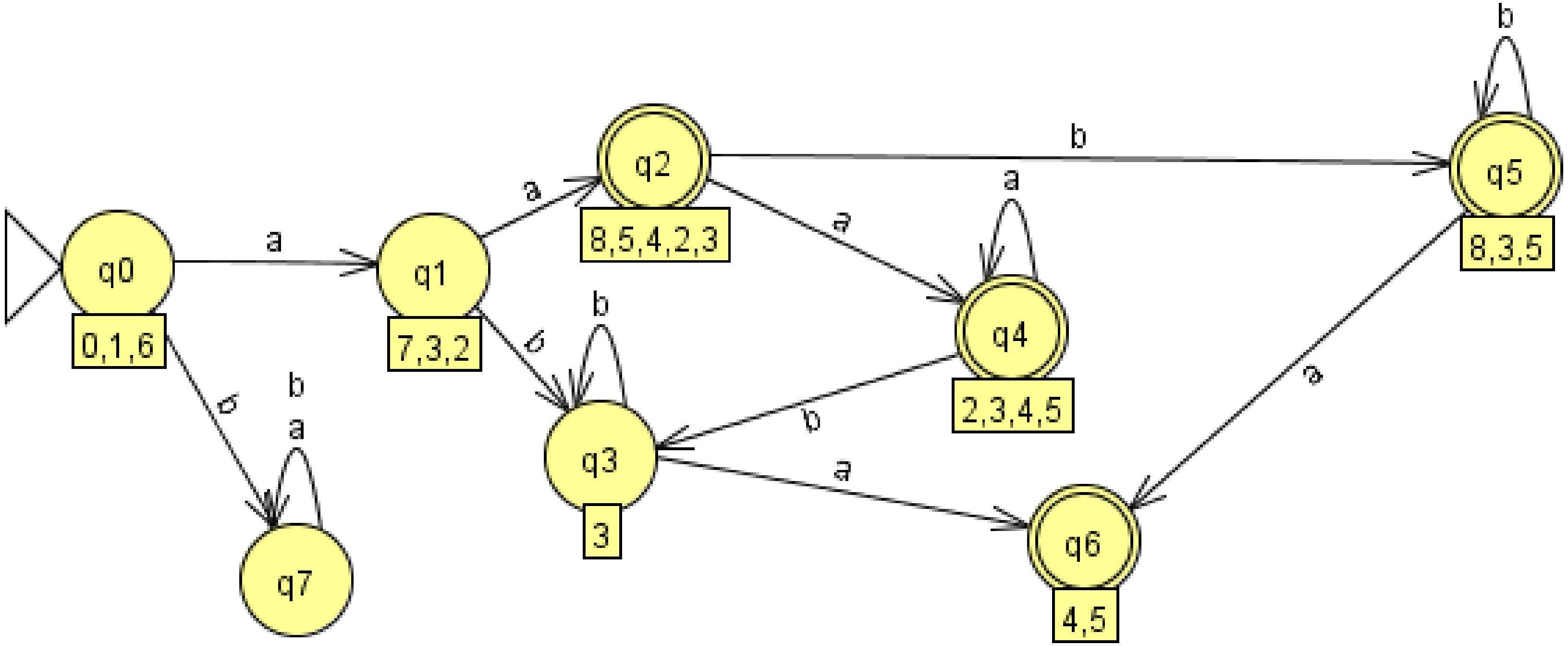
Topics

- Convert
  - RA/FA/RG
  - Any CFG to Chomsky NF
- Given language, define
  - Grammar
  - RA
  - Automata

Exercise

Give a NFA with a single final state for the language $L(aa^*b^*a + aab^*)$.


λ
a
a
λ
b
a
λ
λ
a
a
λ
b


a
b
a
b
a
a
b
a
b
a
b
b
a
a
q0
q1
q2
q3
q4
q5
q6
q7
0,1,6
7,3,2
8,5,4,2,3
3
2,3,4,5
8,3,5
4,5

Exercise

Give the regular expression for the following regular language defined over the alphabet $\Sigma = \{0, 1\}$:

$$L = \{\text{the binary positive integers (without 0)} \\ \text{whose most significant bit is 1} \\ \text{and contain the substring 11}\}$$

$$1(1+0)^*11(1+0)^* + 11(1+0)^*$$

Exercise

Create a NFA for following grammar. What kind of grammar is this?

$$S \to aaB \mid \lambda$$
$$B \to bB$$
$$B \to abS$$

Exercise

. (a) Give a context-free grammar for the following language:

$$L = \{wa^nb^nw^R \quad : w \in \{a,b\}^*, \quad n \geq 0\}$$

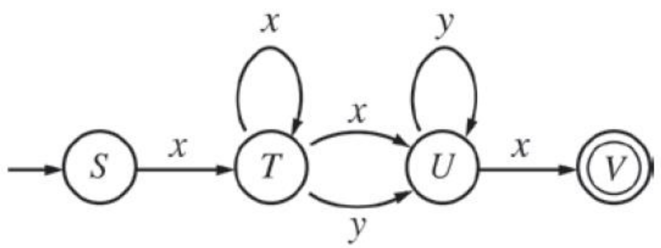
where w is any string over the alphabet $\Sigma = \{a,b\}$ including λ.
(b) Give the derivation of the string $abaabbba$ using your grammar.

$$S \rightarrow aSa|bSb|A$$

$$A \rightarrow aAb|\lambda$$

$$S \Rightarrow aSa \Rightarrow abSba \Rightarrow abAba \Rightarrow abaAbba \Rightarrow abaaAbbba \Rightarrow abaabbba$$

Consider the following nondeterministic finite state automaton over alphabet $\{x, y\}$ with start state S.



4. Which of the following is the regular expression corresponding to the automaton above?

(A) $xxx + yyx$

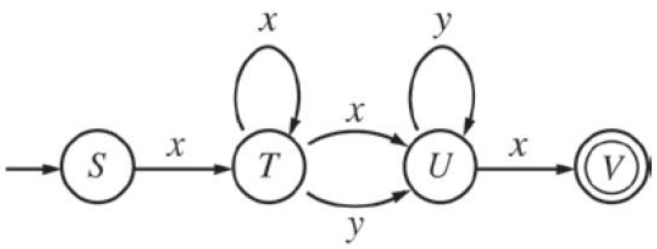
(B) x^3y^2x

(C) x^*y^*x

(D) $xx^*(x + y)y^*x$

(E) x^*xy^*x

Consider the following nondeterministic finite state automaton over alphabet $\{x, y\}$ with start state S.



5. Which of the following grammars over alphabet $\{x, y\}$ generates the language recognized by the automaton above?

(A) $S \to xT$
$T \to xT \mid xU \mid yU$
$U \to yU \mid xV$

(B) $S \to xT$
$T \to xT \mid xU \mid yU$
$U \to yU \mid x$

(C) $S \to xT \mid T$
$T \to xT \mid xU \mid yU \mid T \mid U$
$U \to yU \mid xV \mid V \mid x$

(D) $S \to xV$
$T \to xT \mid yU$
$U \to yU \mid xV$

(E) $S \to xT$
$T \to xT \mid T$
$U \to yU \mid V$
$V \to xV \mid x$

53. Consider a regular language L over $\{0, 1\}$. Which of the following languages over $\{0, 1\}$ must also be regular?

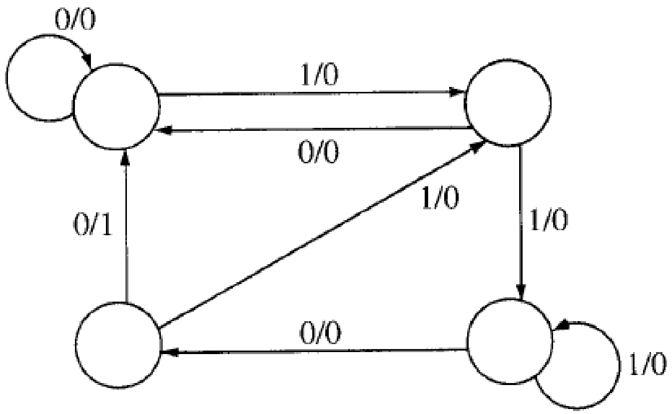
I. $\{w \in L \mid \text{the length of } w \text{ is even}\}$

II. $\{w \in L \mid \text{the length of } w \text{ is prime}\}$

III. $\{w \in L \mid \text{the length of } w \text{ is an integer power of } 2\}$

(A) None (B) I only (C) III only (D) I and III only (E) I, II, and III

11. Consider an output-producing, deterministic finite state automaton (DFA) of the kind indicated in the figure below, in which it is assumed that every state is a final state.



Assume that the input is at least four bits long. Which of the following is(are) true?

I. The last bit of the output depends on the start state.
II. If the input ends with "1100", then the output must end with "1".
III. The output cannot end with "1" unless the input ends with "1100".

(A) I only
(B) II only
(C) I and II only
(D) II and III only
(E) I, II, and III

12. A particular BNF definition for a “word” is given by the following rules.

```
<word>::=<letter> | <letter><pairlet> | <letter><pairdig>
<pairlet>::=<letter><letter> | <pairlet><letter><letter>
<pairdig>::=<digit><digit> | <pairdig><digit><digit>
<letter>::=a|b|c|. . .|y|z
<digit>::=0|1|2|. . .|9
```

Which of the following lexical entities can be derived from <word> ?

I. word
II. words
III. c22

(A) None
(B) I and II only
(C) I and III only
(D) II and III only
(E) I, II, and III

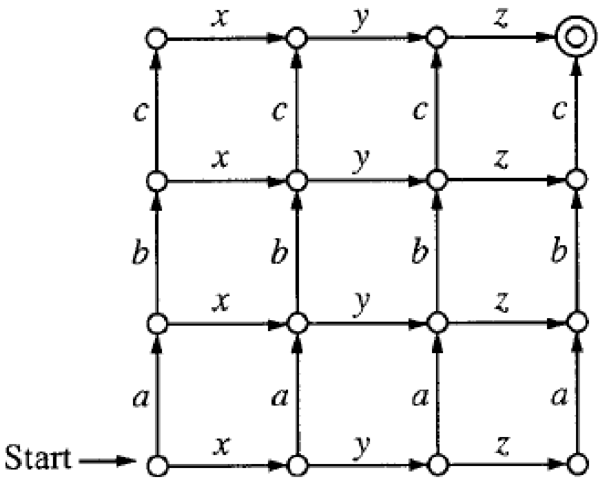
16. Consider the following grammar.

$$
\begin{aligned}
S &::= AB \\
A &::= \mathrm{a} \\
A &::= B\mathrm{a}B \\
B &::= \mathrm{bb}A
\end{aligned}
$$

Which of the following is FALSE?

(A) The length of every string produced by the grammar is even.
(B) No string produced by the grammar has an odd number of consecutive b's.
(C) No string produced by the grammar has three consecutive a's.
(D) No string produced by the grammar has four consecutive b's.
(E) Every string produced by the grammar has at least as many b's as a's.

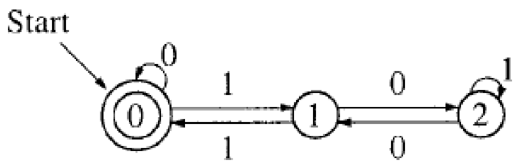
25.



The finite automaton above recognizes a set of strings of length 6. What is the total number of strings in the set?

(A) 18
(B) 20
(C) 30
(D) 32
(E) None of the above

28.



State 0 is both the starting state and the accepting state.

Each of the following is a regular expression that denotes a subset of the language recognized by the automaton above EXCEPT

(A) 0*(11)*0*

(B) 0*1(10*1)*1

(C) 0*1(10*1)*10*

(D) 0*1(10*1)0(100)*

(E) (0*1(10*1)*10* + 0*)*

70. If DFA denotes "deterministic finite automata" and NDFA denotes "nondeterministic finite automata," which of the following is FALSE?

(A) For any language L, if L can be recognized by a DFA, then $\overline{L}$ can be recognized by a DFA.
(B) For any language L, if L can be recognized by an NDFA, then $\overline{L}$ can be recognized by an NDFA.
(C) For any language L, if L is context-free, then $\overline{L}$ is context-free.
(D) For any language L, if L can be recognized in polynomial time, then $\overline{L}$ can be recognized in polynomial time.
(E) For any language L, if L is decidable, then $\overline{L}$ is decidable.

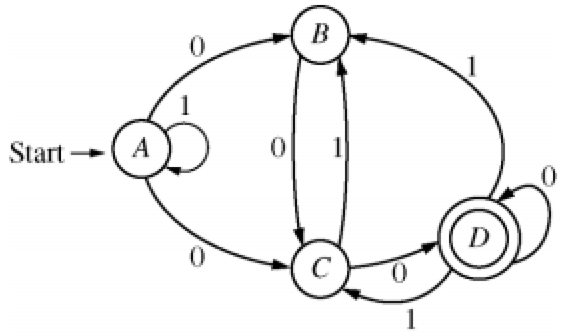
5. Which of the following regular expressions will not generate a string with two consecutive 1s? (Note that ε denotes the empty string.)

I. $(1 + \varepsilon)(01 + 0)^*$

II. $(01 + 10)^*$

III. $(0 + 1)^*(0 + \varepsilon)$

(A) I only
(B) II only
(C) III only
(D) I and II only
(E) II and III only



14. The figure above represents a nondeterministic finite automaton with accepting state D. Which of the following strings does the automaton accept?

(A) 001
(B) 1101
(C) 01100
(D) 000110
(E) 100100