Pushdown Automata
(PDA)

Regular Languages (Review)

- Every Regular language has

  regular expression / regular grammar / Finite Accepter

- Every regular language RL is **Context-Free Language**.

- But some CFL are not regular:

  The language $L = \{a^n b^n : n \geq 1\}$ has CFG $S \rightarrow aSb \mid ab$

  The language $\{ww^R : w \in \{a, b\}^*\}$ has CFG $S \rightarrow aSa/bSb/a/b/\varepsilon$

Context-Free Languages (Review)

A grammar $G = (V, T, S, P)$ is CFG if all production rules have the form

$A \rightarrow x$, where $A \in V$, and $x \in (V \cup T)^*$

A language L is CFL iff there is a CFG G such that $L = L(G)$

All regular languages, and some non-regular languages, can be generated by CFGs

The family of regular languages is proper subset of the family of context-free languages.

Stack Memory

The language $L = \{wcw^R : w \in \{a, b\}^*\}$ is CFL but not RL

- We can not have a DFA for L
- Problem is **memory**, DFA's cannot remember left hand substring

What kind of memory do we need to be able to recognize strings in L?

- Answer: a **stack**

Stack Memory

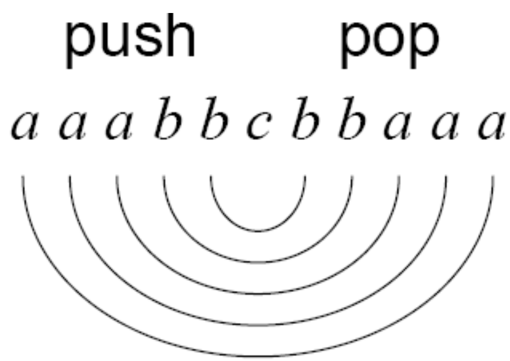
Example: $u = aaabbcbbaaa \in L$

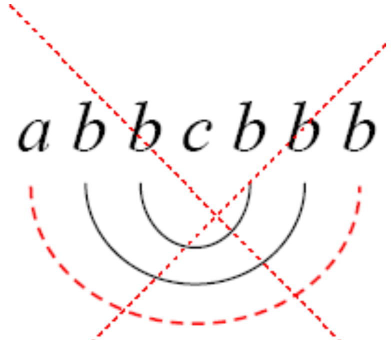
We **push** the first part of the string onto the stack and

after the c is encountered

start **popping** characters from the stack and matching them with each character.

if everything matches, this string u $\in L$





Stack Memory

We can also use a stack for counting out equal numbers of *a*'s and *b*'s.

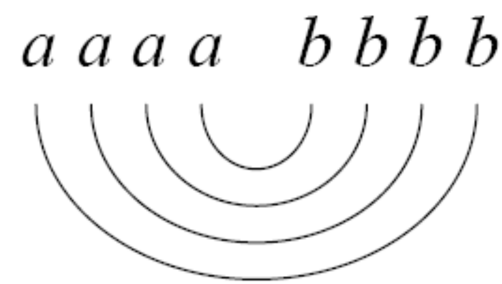
Example:

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

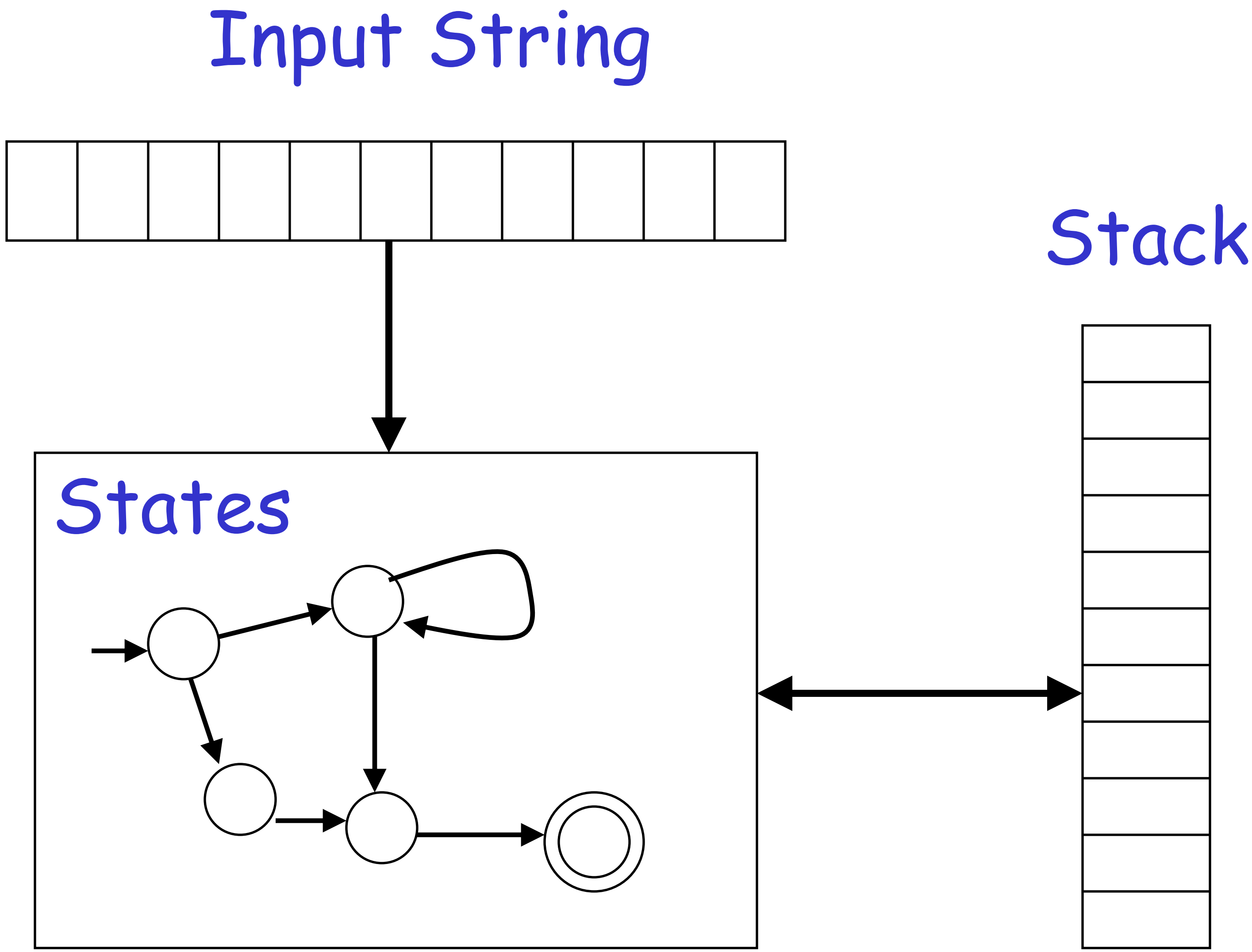
$w = aaaabbbb \in L$

Push the *a*'s onto the stack, then **pop** an *a* off and match it with each *b*.

If we finish processing the string successfully (and there are no more *a*'s on our stack), then the string belongs to L.



Pushdown Automaton -- PDA



Nondeterministic Push-Down Automata

A language is ***context free*** iff some **Nondeterministic PushDown Automata** (NPDA) recognizes (accepts) it.

- Intuition: **NPDA** = **NFA** + one **stack** for memory.
- Stack remembers info about previous part of string
- E.g., $a^n b^n$

Deterministic PushDown Automata (DPDA) can accept some but *not all* of the CFLs.

Thus, there is no longer an equivalence between the deterministic and nondeterministic versions,

- i.e., languages recognized by DPDA are a proper subset of context-free languages.

NPDA

A **NPDA** is a seven-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, z, F)$ where

Q finite set of states

Σ finite set of input alphabet

Γ finite set of stack alphabet

δ transition function from

$Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow$ finite subsets of $Q \times \Gamma^*$

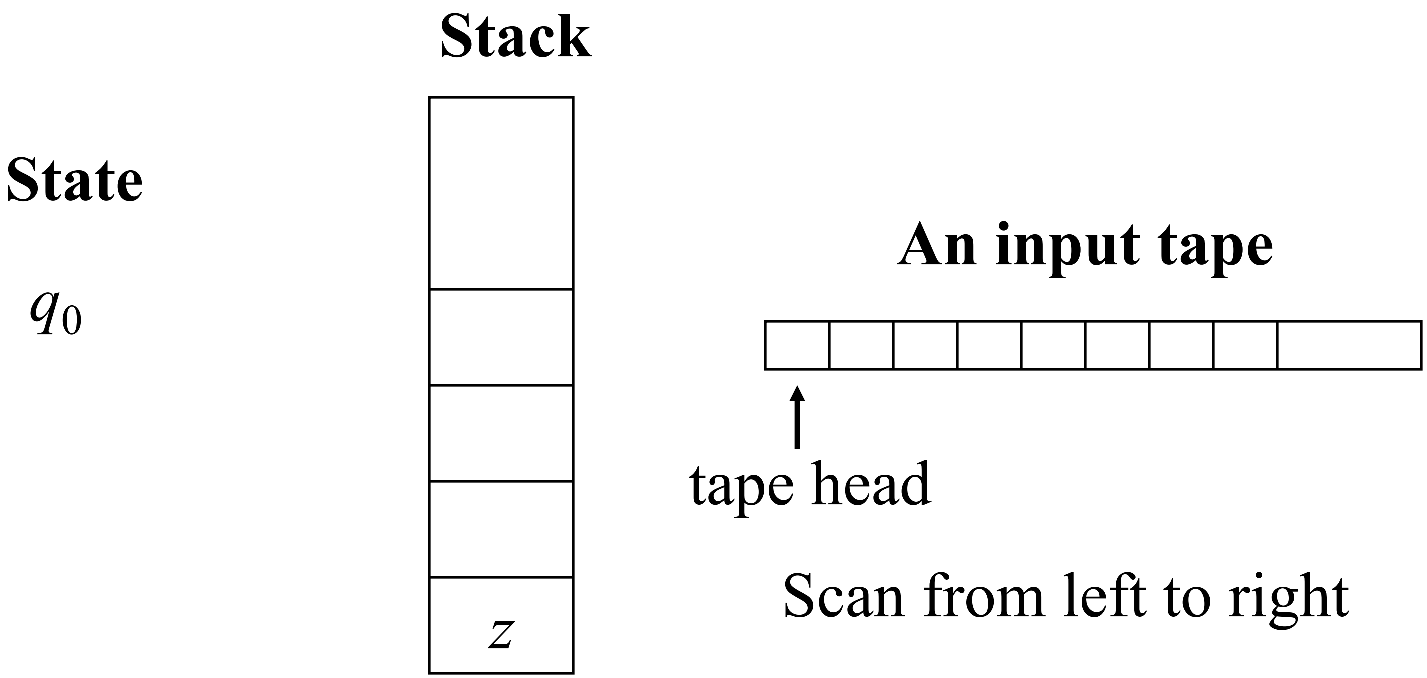
q_0 start state $\qquad q_0 \in Q$

z initial stack symbol $\qquad z \in \Gamma$

F set of final states $\qquad F \subseteq Q$

NPDA

There are three things in an NPDA:



NPDA : Transition Function

The transition function deserves further explanation

$$\delta : \underbrace{Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma}_{\text{INPUT}} \rightarrow \underbrace{\text{finite subsets of } Q \times \Gamma^*}_{\text{OUTPUT}}$$

stack symbol popped

next state

string of stack symbols pushed onto stack (right-to-left), i.e., the left-most symbol is at top of stack

- $\delta(q_h, a, p) = \underbrace{\{(q_i, s_n), \ldots, (q_j, s_m)\}}$

current state

input symbol read (includes λ)

nondeterministically chooses one of these pairs

NPDA : Transition Function

In an NPDA, we read an input symbol in a state, but

we also need to know what is on the stack before

- we can decide what is new state.
  - When moving to the new state, we also need to decide what to do with the stack.

NPDA : Transition Function

$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \rightarrow$ finite subsets of $Q \times \Gamma^*$

The second argument of δ is ε.

It means transition is possible without consuming input symbol.

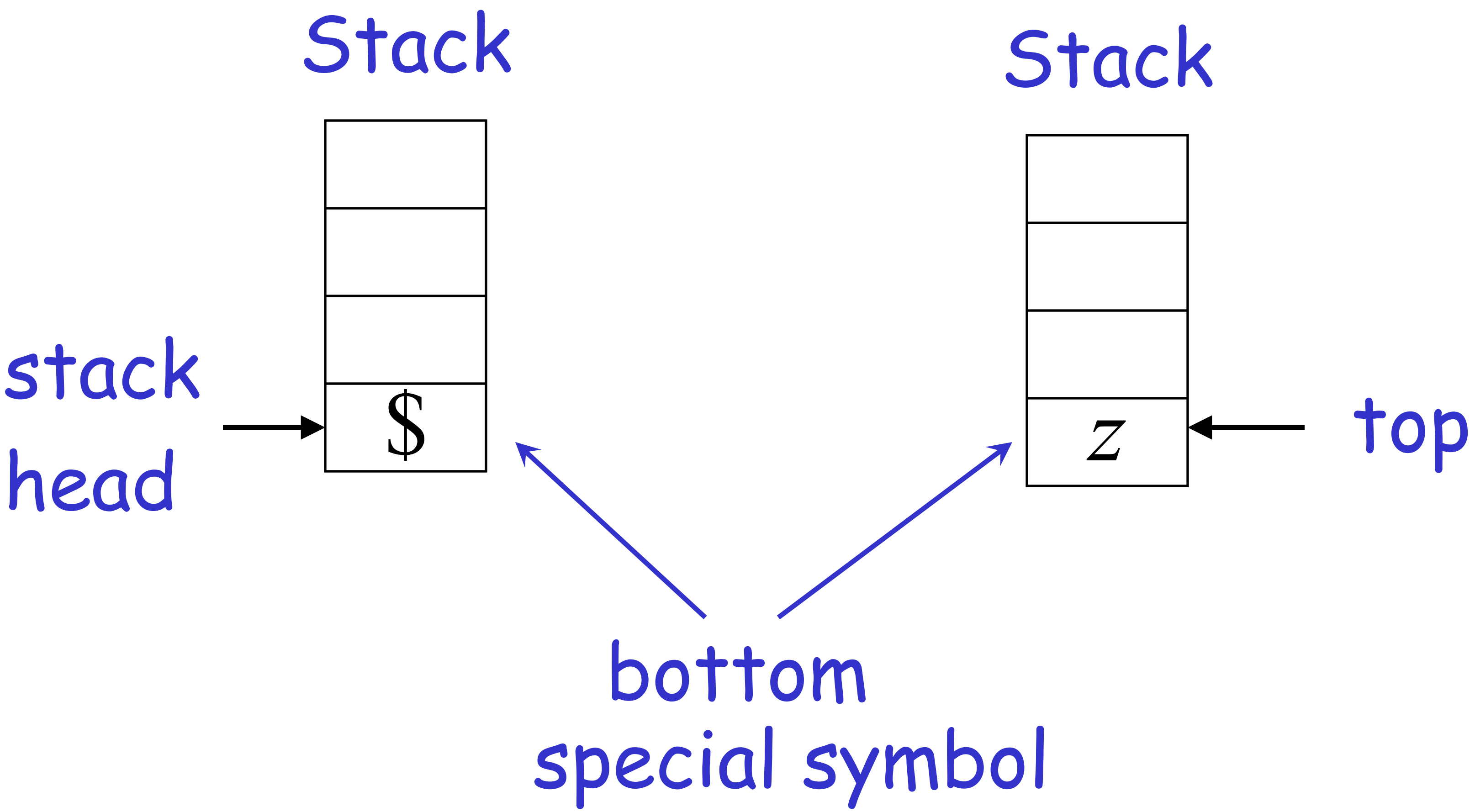
The third argument of δ is from Γ.

It means transition is not possible if stack is empty.

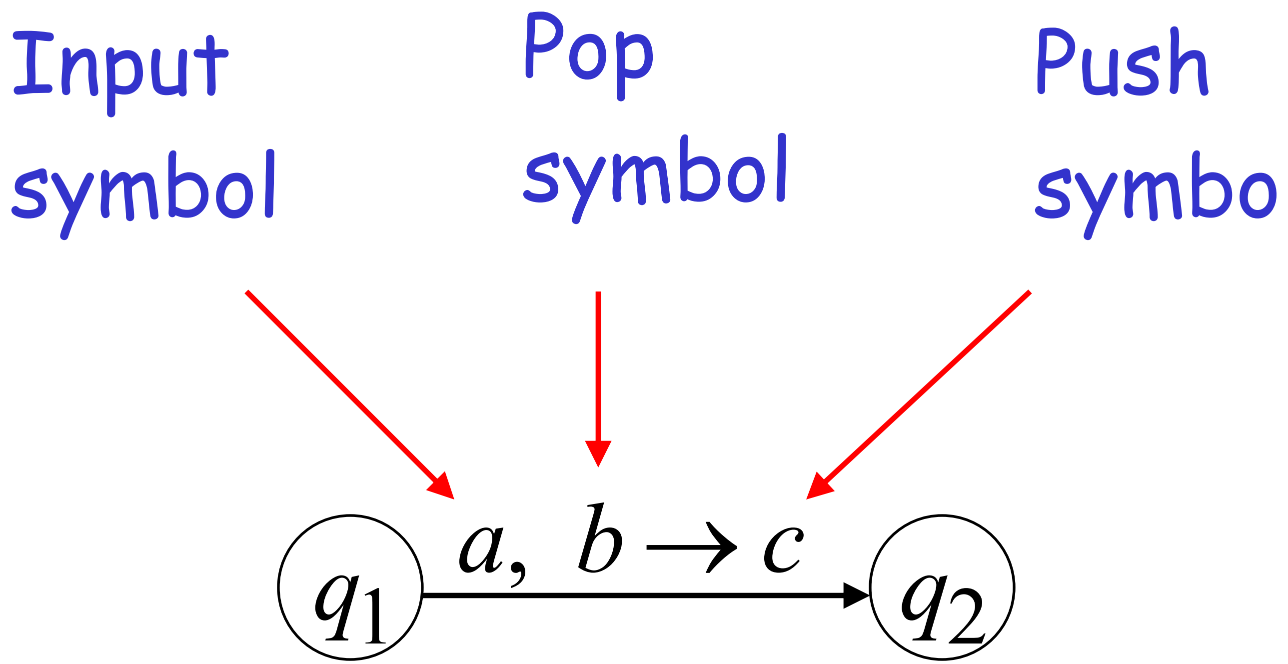
Finally, finite subset is necessary because $Q \times \Gamma^*$ *is infinite set.*

Initial Stack Symbol

Initially, It is assumed that stack consists of start symbol (z/$/#/0)

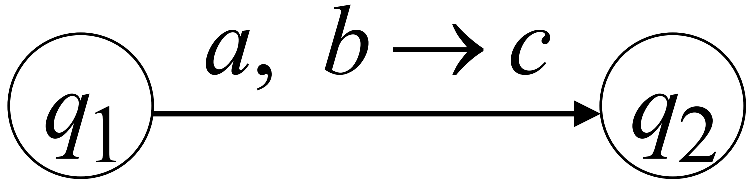


The Transition

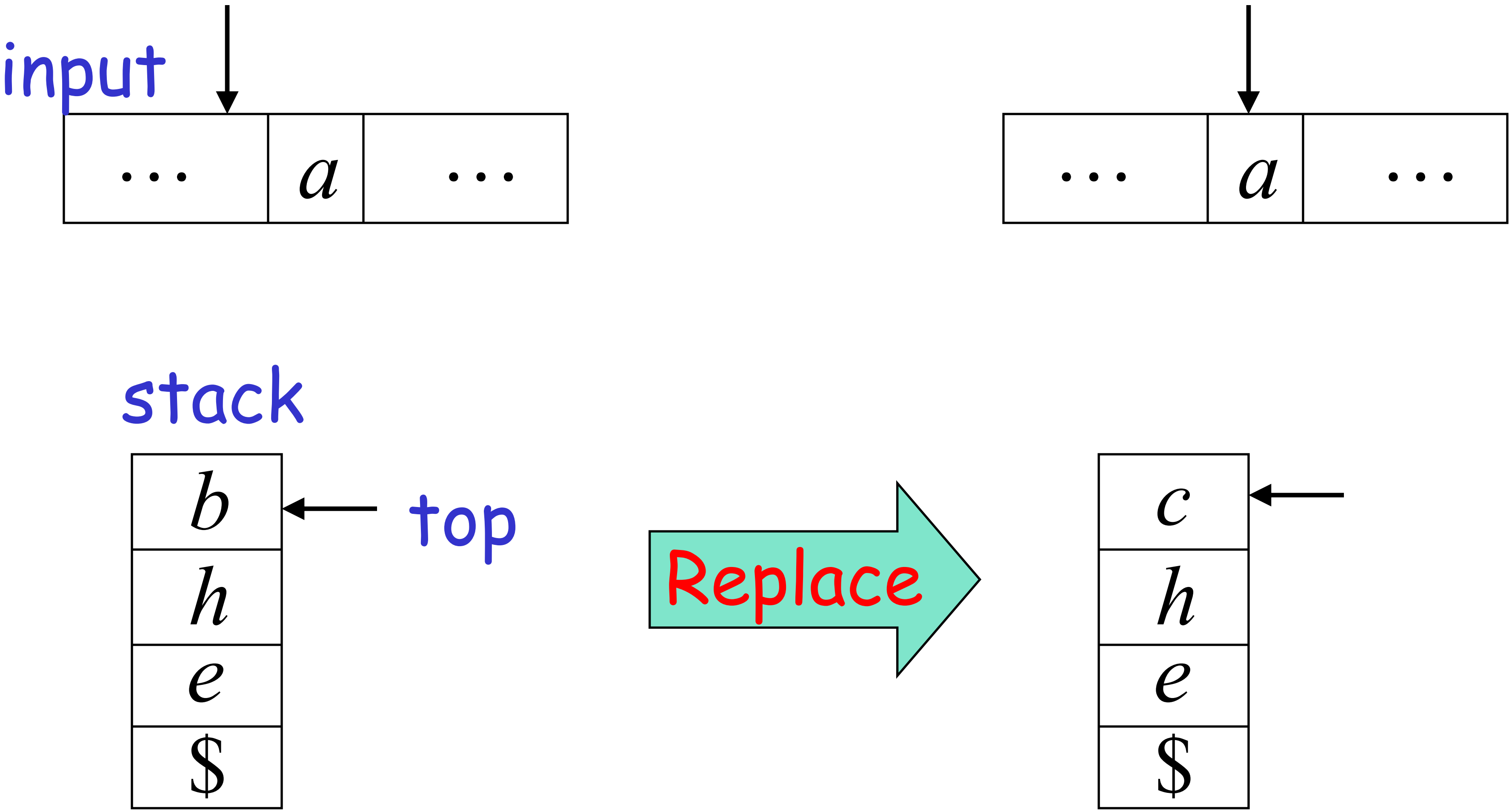


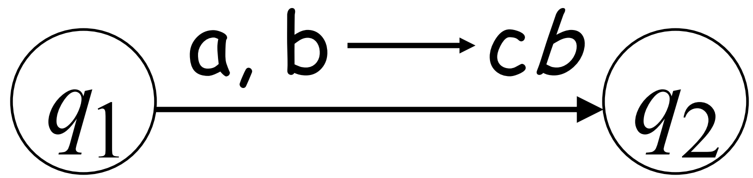
$$\delta(q_1, a, b) = \{ (q_2, c) \}$$

$$\delta : Q \times (\Sigma \cup \{\varepsilon\}) \times \Gamma \to \text{finite subsets of } Q \times \Gamma^*$$

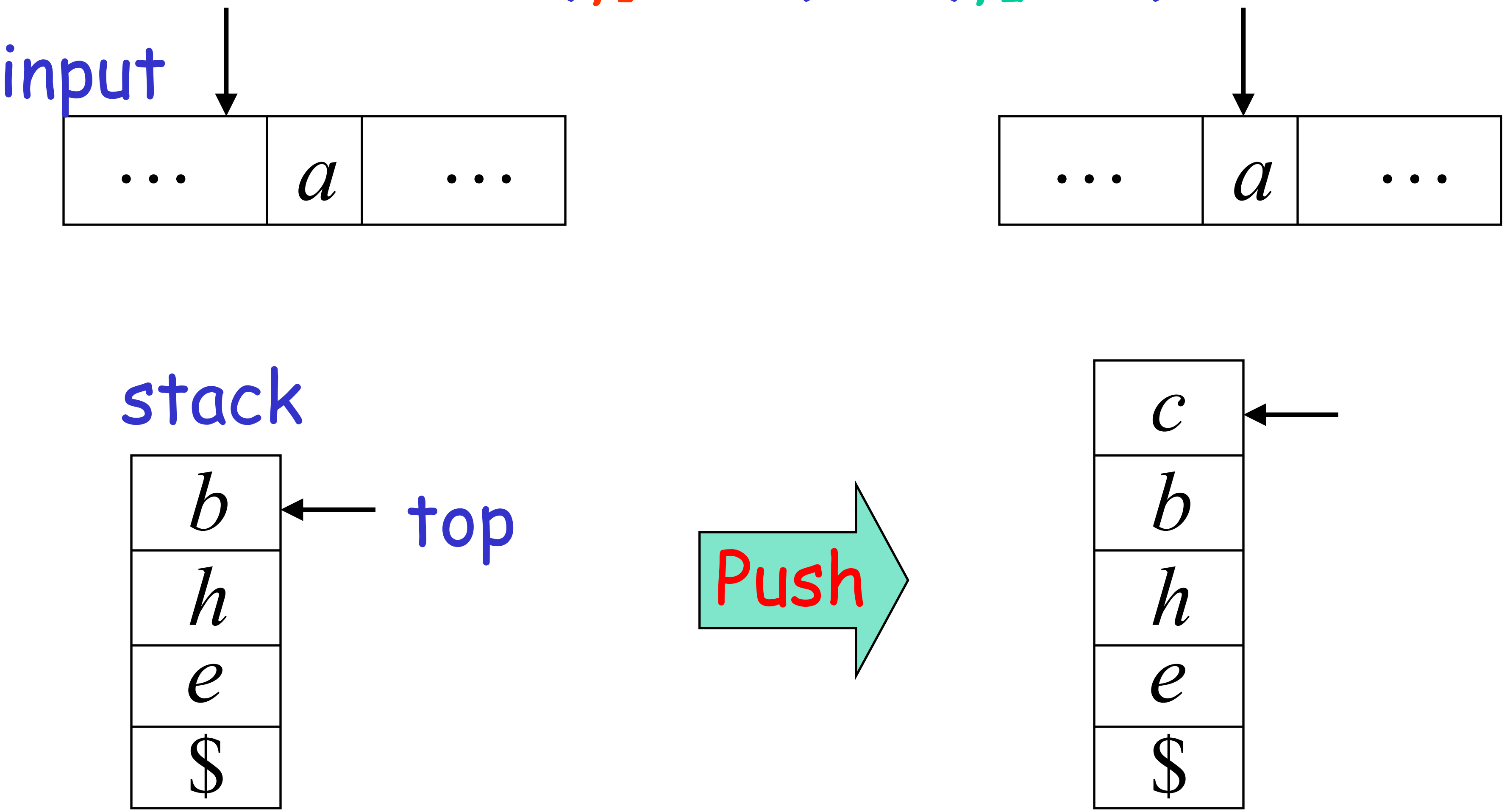


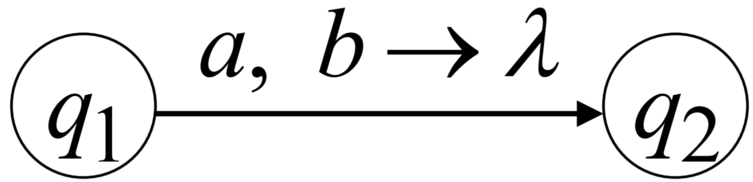
$\delta(q_1, a, b) = \{ (q_2, c) \}$



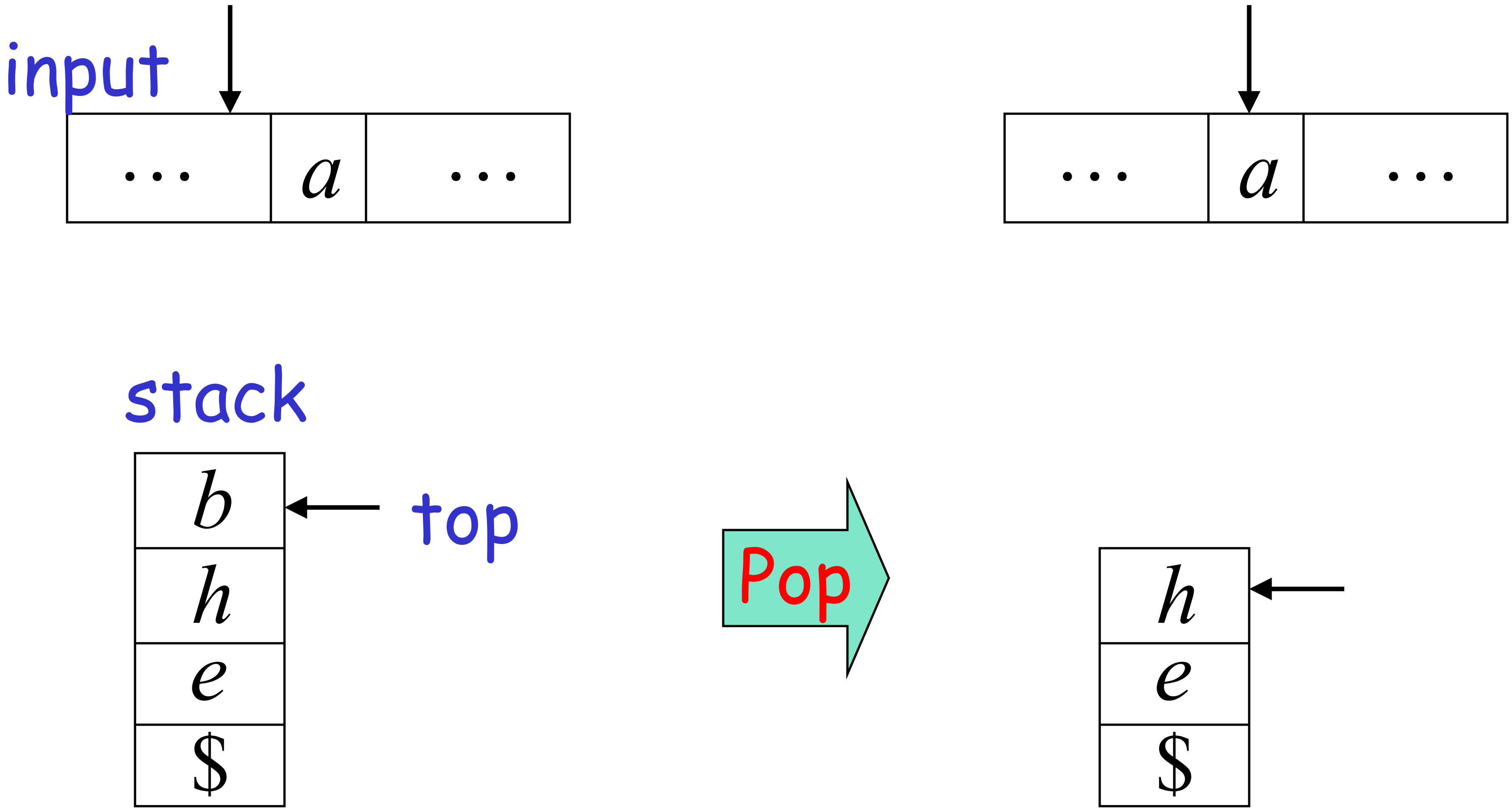


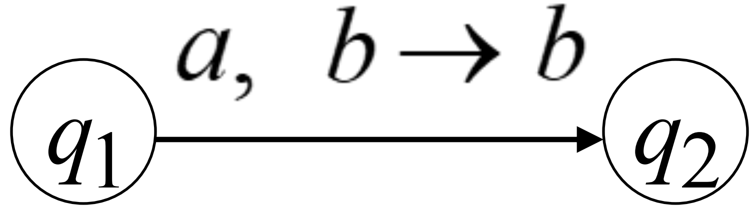
$$\delta(q_1, a, b) = \{(q_2, cb)\}$$



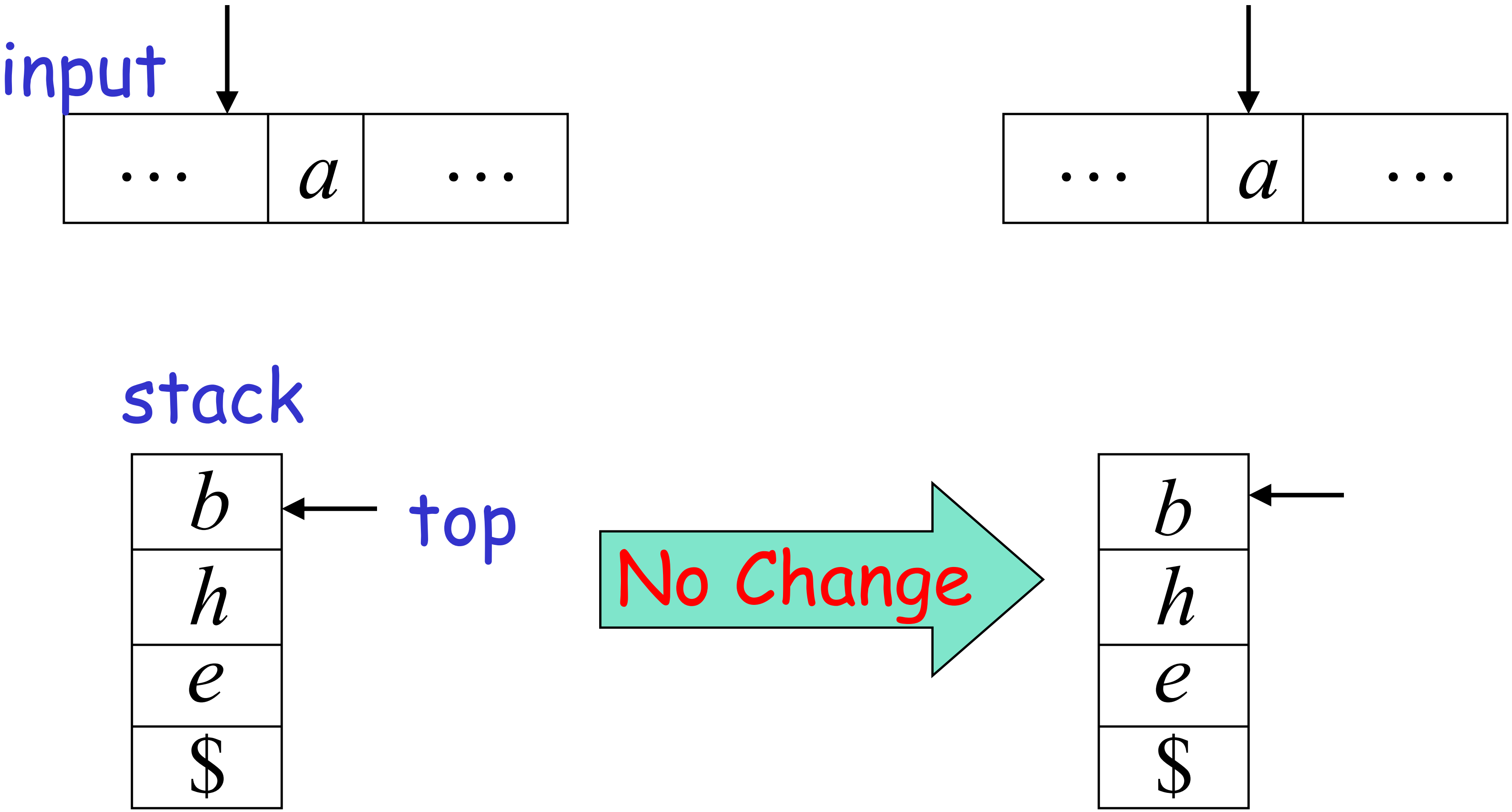


$$\delta(q_1, a, b) = \{(q_2, \lambda)\}$$

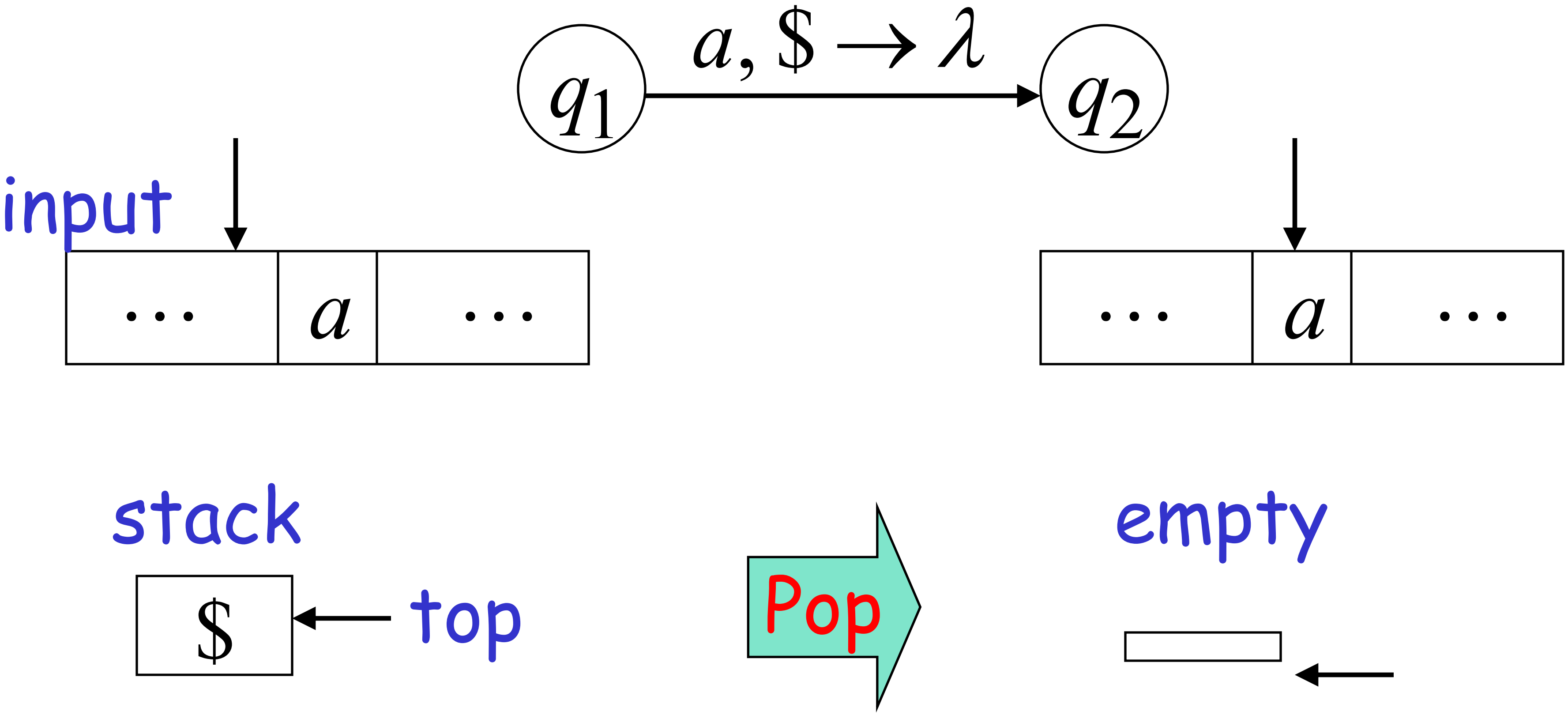




$\delta(q_1, a, b) = \{(q_2, b)\}$

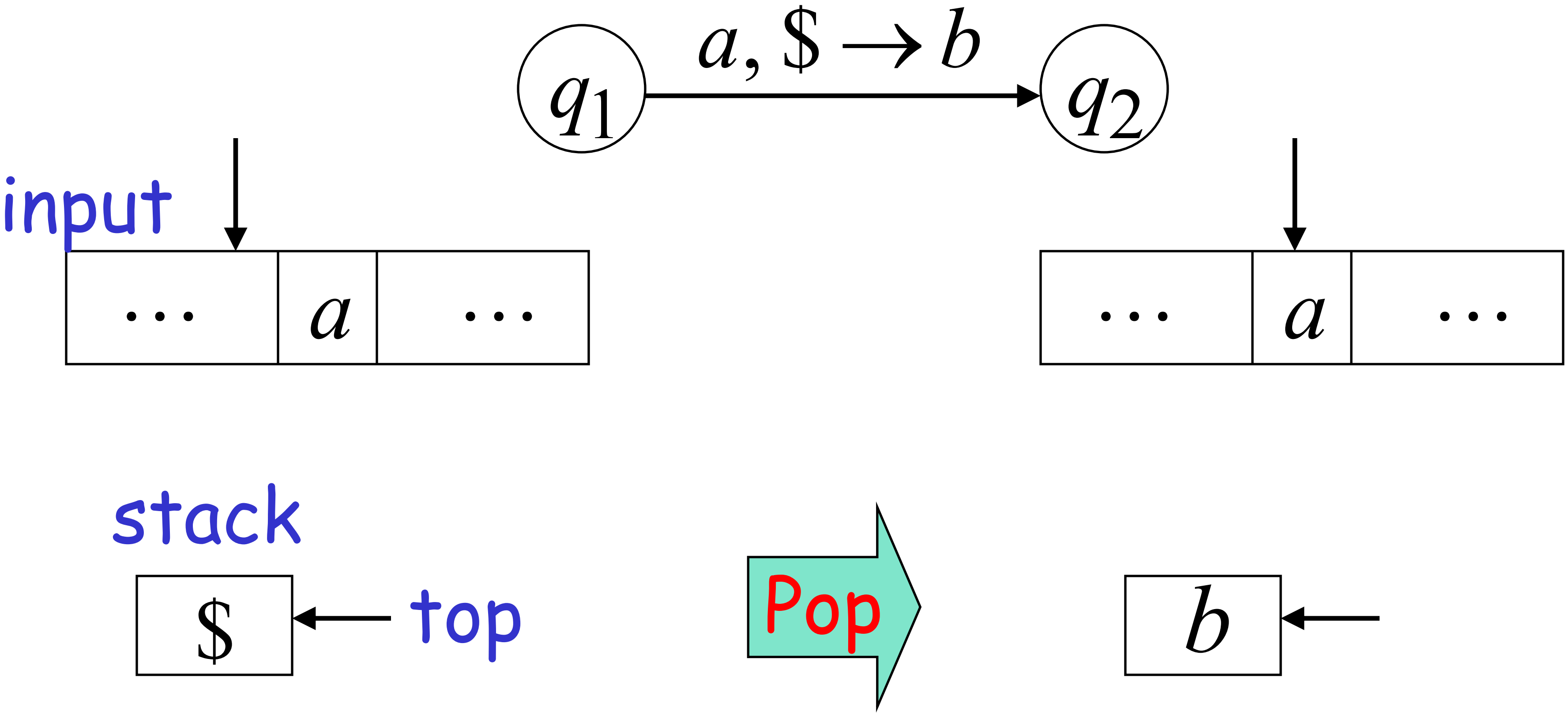


A Possible Transition

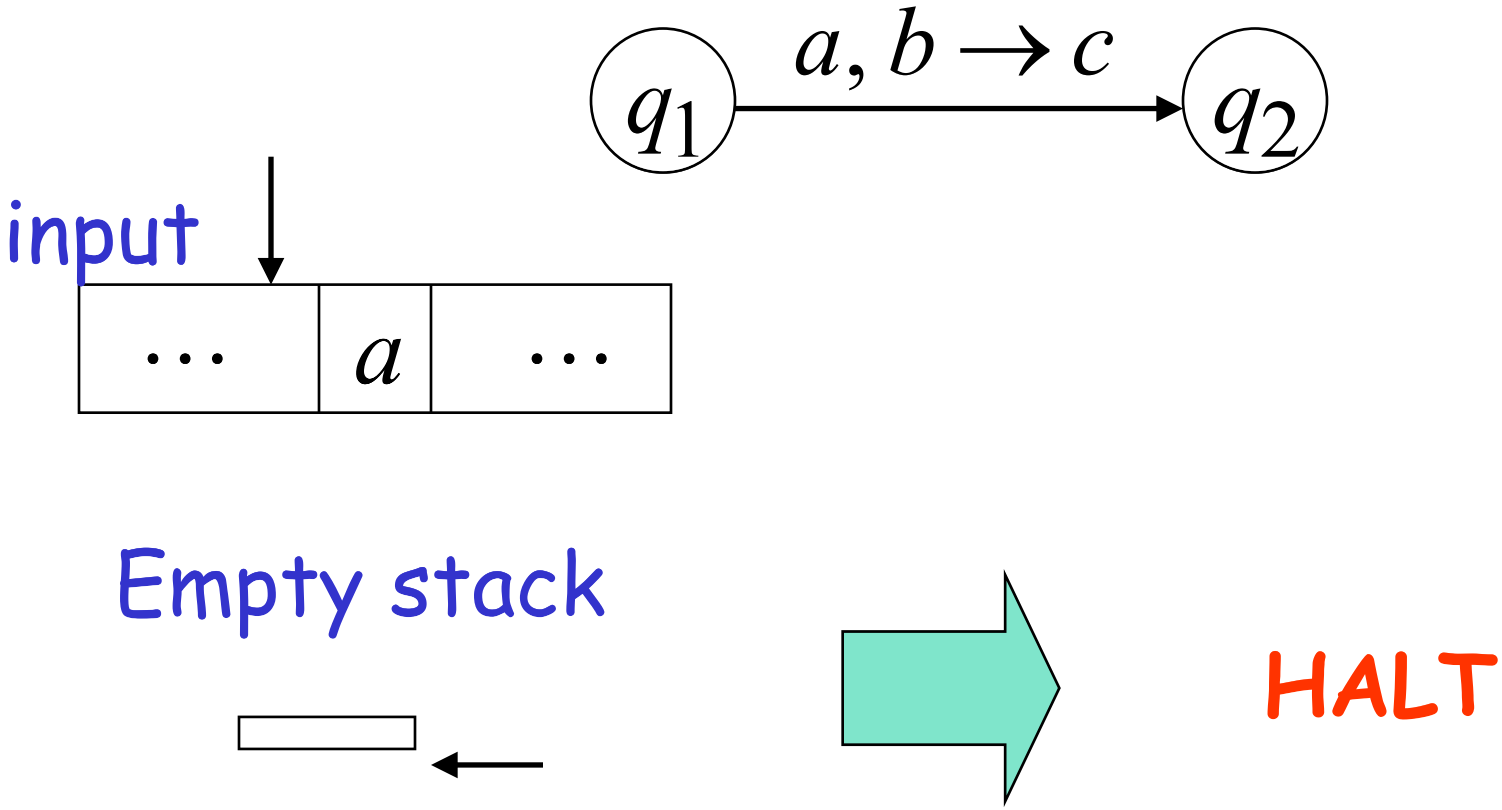


$$\delta(q_1, a, \$) = \{ (q_2, \lambda) \}$$

A Possible Transition

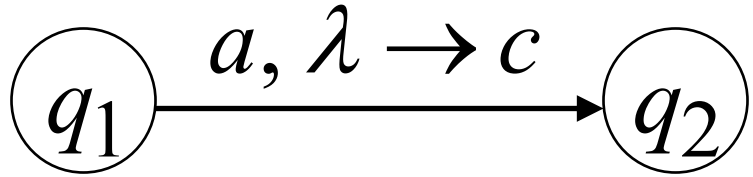


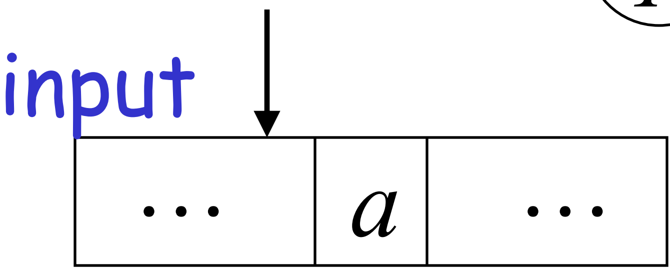
A Bad Transition



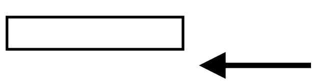
The automaton **Halts** in state q_1
and **Rejects** the input string

A Bad Transition





Empty stack



HALT

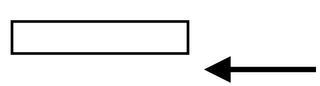
The automaton **Halts** in state q_1
and **Rejects** the input string

No transition is allowed to be followed

When the stack is empty

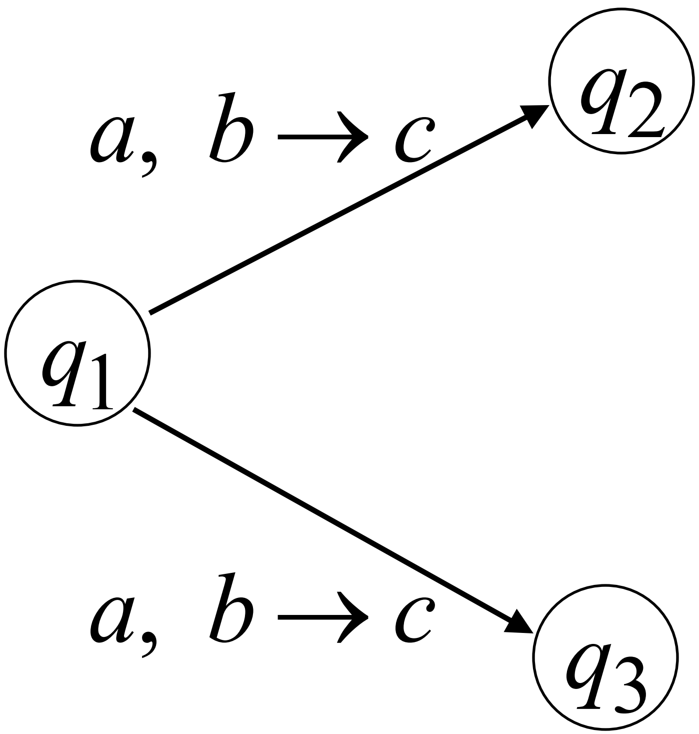
$$q_1 \xrightarrow{x, y \rightarrow z} q_2$$

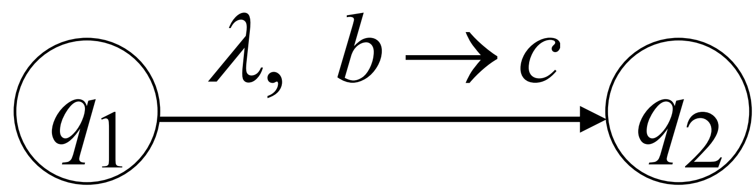
Empty stack



Non-Determinism

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





λ – transition

These are allowed transitions in a

Non-deterministic PDA (NPDA)

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

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

where

$Q = \{q_0, q_1, q_2, q_3\}$

$\Sigma = \{a, b\}$

$\Gamma = \{Z, a\}$

δ

input popped pushed

$\delta(q_0, \varepsilon, Z) = \{(q_3, \varepsilon)\}$

$\delta(q_0, a, Z) = \{(q_1, aZ)\}$

$\delta(q_1, a, a) = \{(q_1, aa)\}$

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

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

$\delta(q_2, \varepsilon, Z) = \{(q_3, \varepsilon)\}$

q_0 is the start state

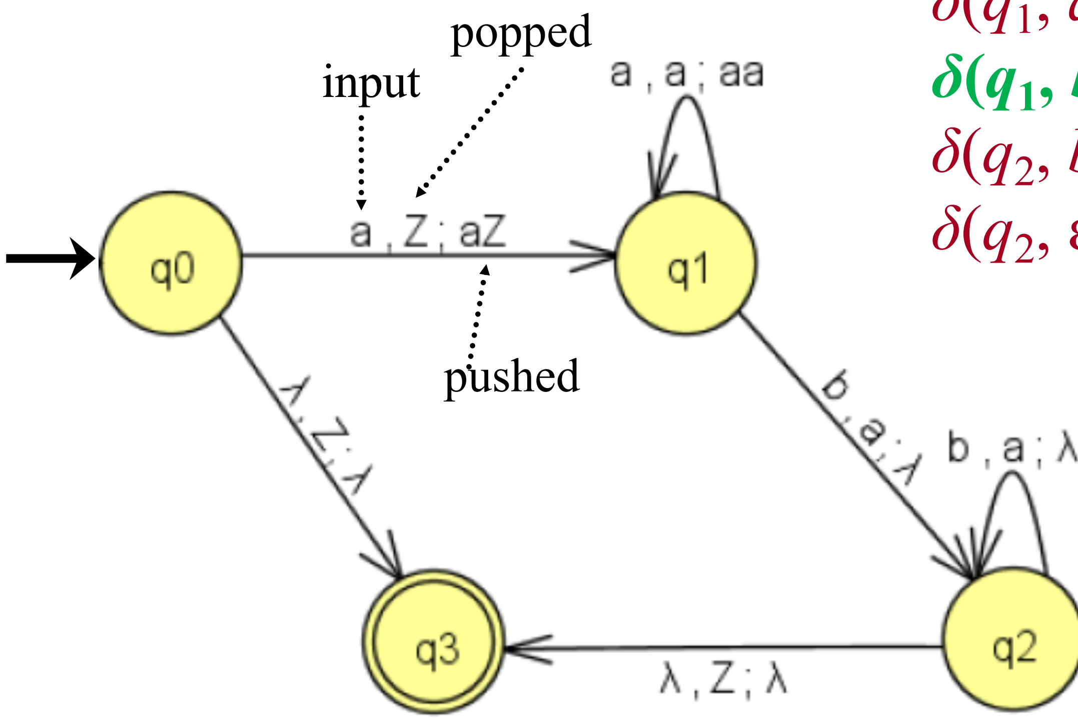
Z is the initial stack symbol

$F = \{q_3\}$

Construct NPDA

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

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



$\delta(q_0, \varepsilon, Z) = \{(q_3, \varepsilon)\}$

$\delta(q_0, a, Z) = \{(q_1, aZ)\}$

$\delta(q_1, a, a) = \{(q_1, aa)\}$

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

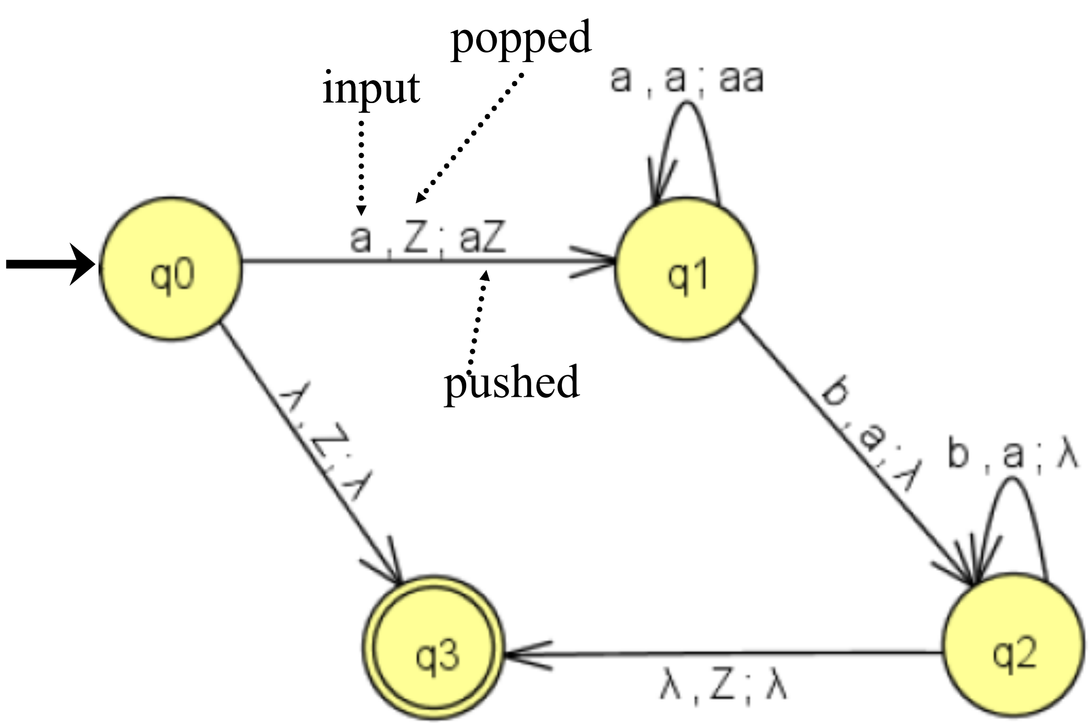
$\delta(q_2, b, a) = \{(q_2, \varepsilon)\}$

$\delta(q_2, \varepsilon, Z) = \{(q_3, \varepsilon)\}$

Construct NPDA

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

Add one more transition for $\{a\}$



NPDA

A NPDA **configuration** (**Instantaneous Description**) is represented by

$[q_n, u, \alpha]$ where

q_n :current state

u :unprocessed input (Total)

α :current stack content (Total)

if $\delta(q_n, a, A) = \{(q_m, B)\}$

then $[q_n, au, A\alpha] \vdash [q_m, u, B\alpha]$

The notation $[q_n, u, \alpha] \vdash [q_m, v, \beta]$ indicates that configuration $[q_m, v, \beta]$ is obtained from $[q_n, u, \alpha]$ **by a *single* transition of the NPDA**

NPDA

The notation $[q_n, u, \alpha] \vdash^* [q_m, v, \beta]$ indicates that configuration $[q_m, v, \beta]$ is obtained from $[q_n, u, \alpha]$ **by *zero or more* transitions of the NPDA**

A **computation** of a NPDA is a sequence of transitions beginning with *start state*.

Acceptance

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

All input symbols are consumed

AND

The last state is a final state

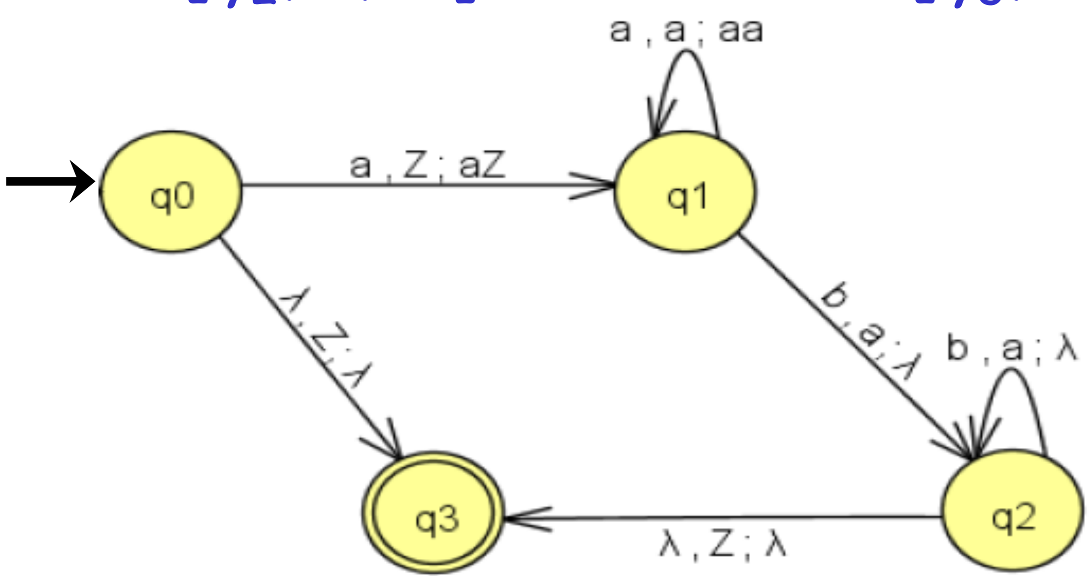
At the end of the computation,
we do not care about the stack contents

Acceptance

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

The computation generated by the input string **w = *aaabbb*** is

$[q_0,\ aaabbb,\ Z] \vdash [q_1, aabbb, aZ]$
$\vdash [q_1, abbb, aaZ] \vdash [q_1, bbb, aaaZ]$
$\vdash [q_2, bb, aaZ] \vdash [q_2, b, aZ]$
$\vdash [q_2, \varepsilon, Z] \vdash [q_3, \varepsilon, \varepsilon]$



state	string	stack
q_0	$aaabbb$	Z
q_1	$aabbb$	aZ
q_1	$abbb$	aaZ
q_1	bbb	$aaaZ$
q_2	bb	aaZ
q_2	b	aZ
q_2	λ	Z
q_3	λ	λ

Rejection

A string is rejected
if in **every computation** with the 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.

Rejection

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

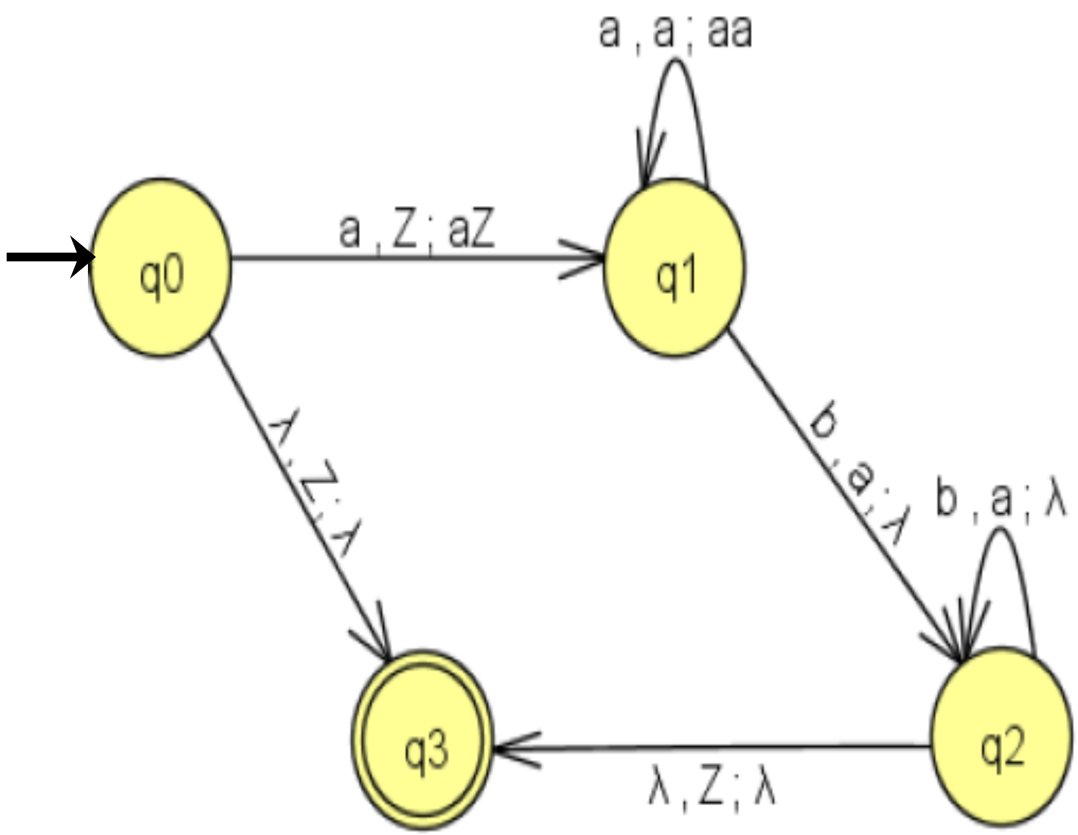
The computation generated by the input string **w = *aabbb*** is

$[q_0, \textbf{\textit{aabbb}}, Z] \vdash [q_1, abbb, aZ]$

$\vdash [q_1, bbb, aaZ] \vdash [q_2, bb, aZ]$

$\vdash [q_2, b, Z]$

Reject w = aabbb



Rejection

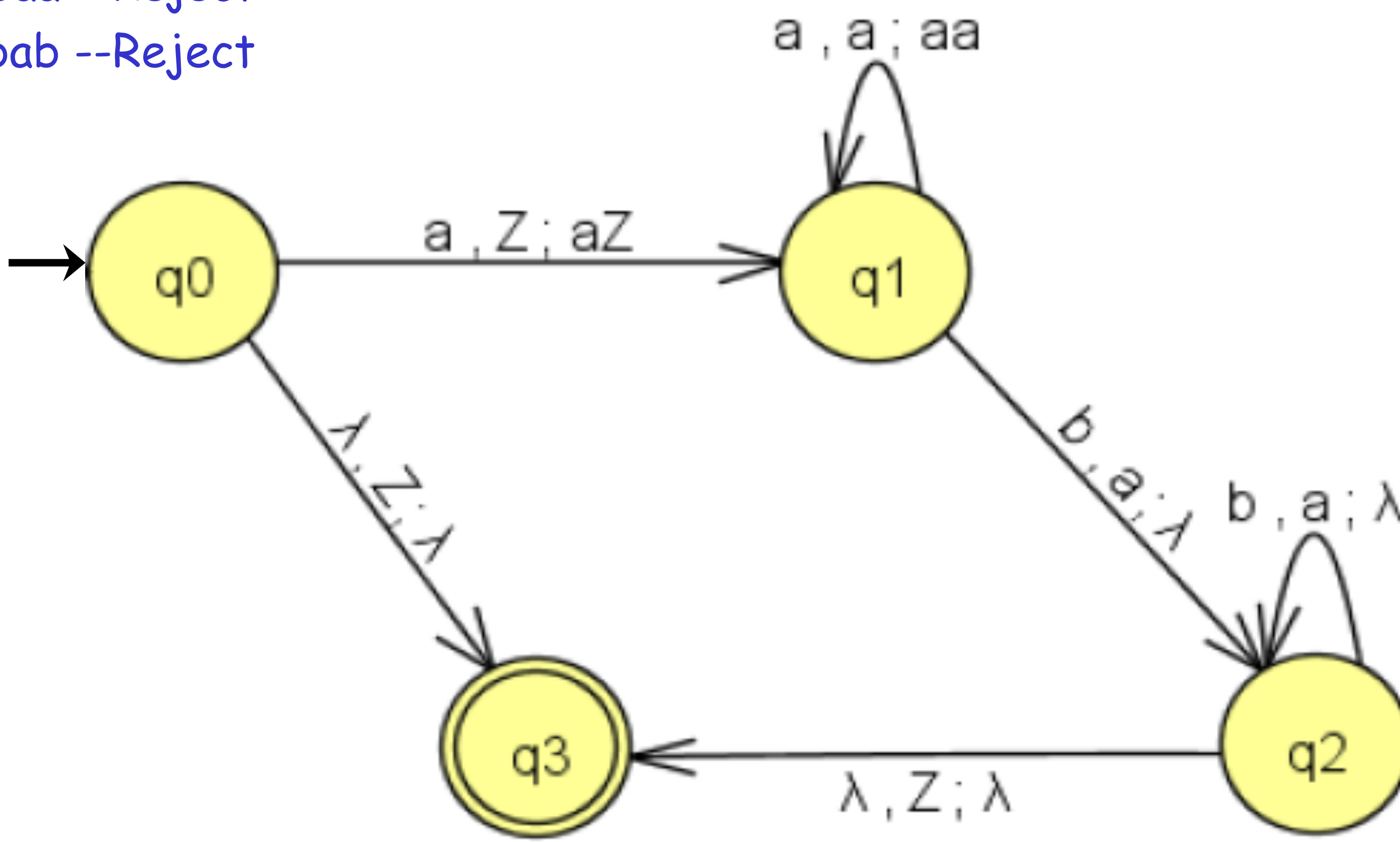
w = aaaaabbbbb -Accept

w = abbb –Reject

w = aaabb –Reject

w = bbaa --Reject

w = abab --Reject



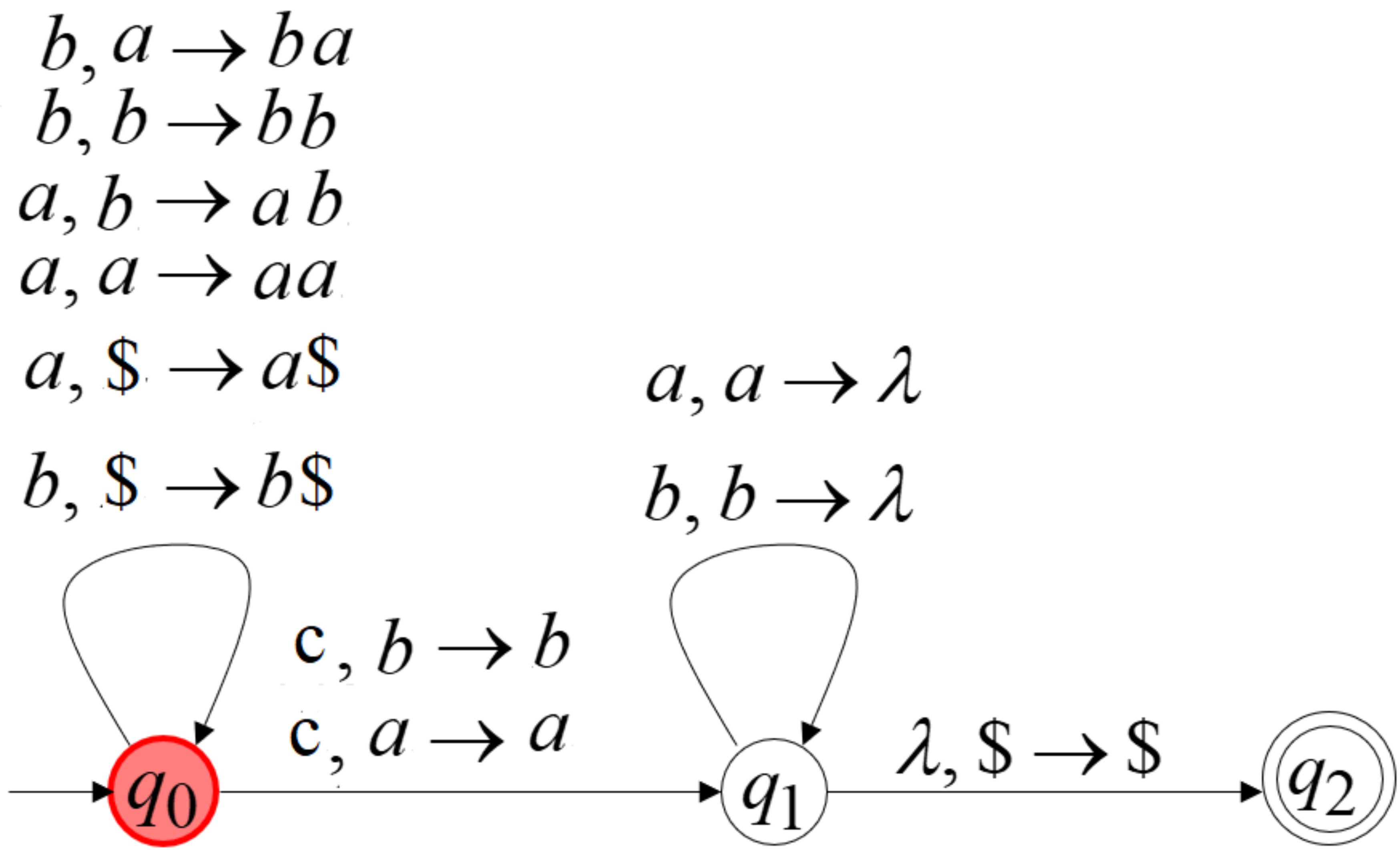
Language of NPDA

The language accepted by NPDA M is

$L(M) = \{w \in \Sigma^* : [q_0, w, z] \vdash^* [p, \varepsilon, u]$
with $p \in F, u \in \Gamma^*\}$

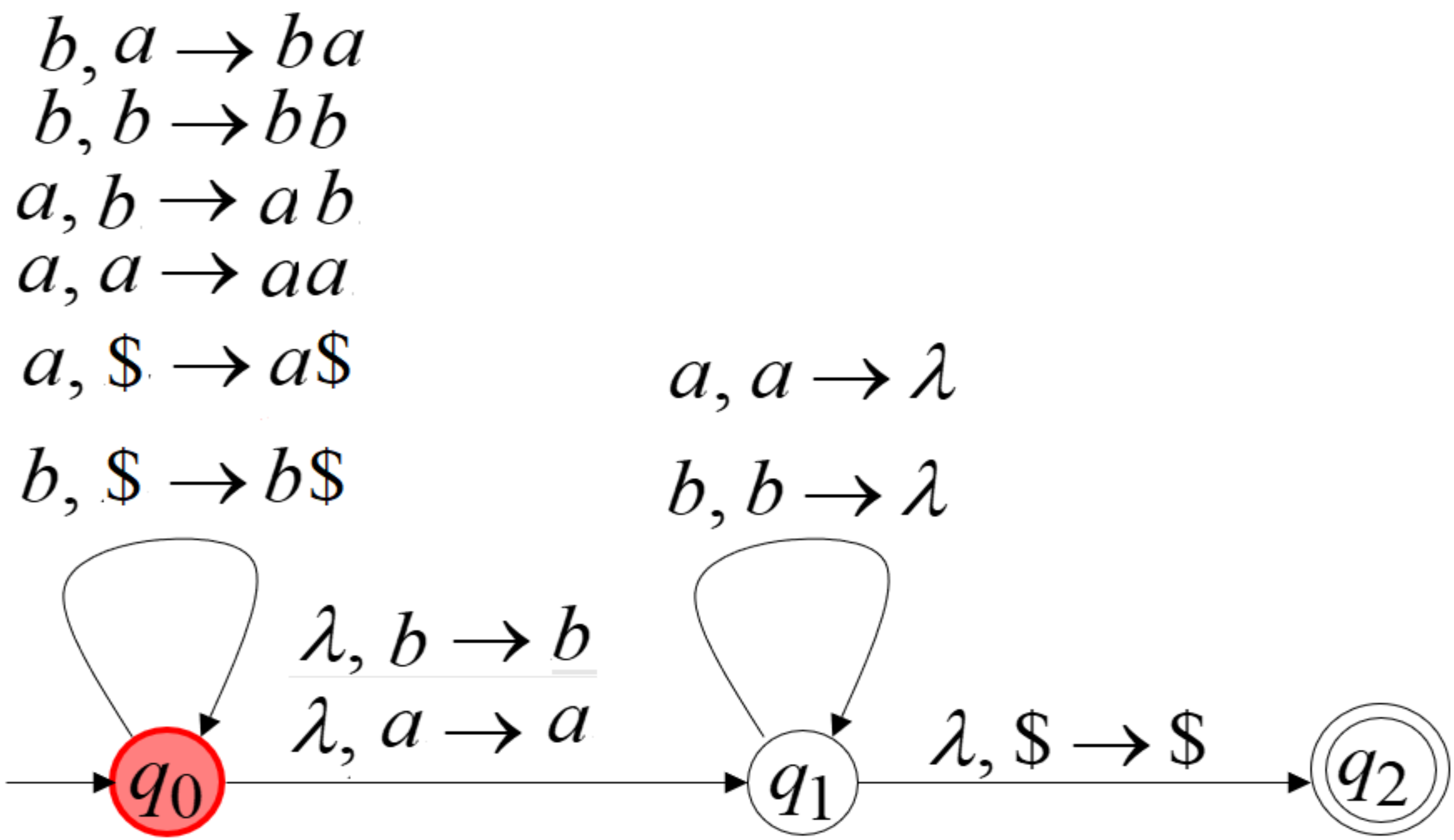
Construct NPDA

Language: $L = \{wcw^R : w \in \{a, b\}^+ \}$



Construct NPDA

Language: $L = \{ww^R : w \in \{a, b\}^+ \}$



Another NPDA example

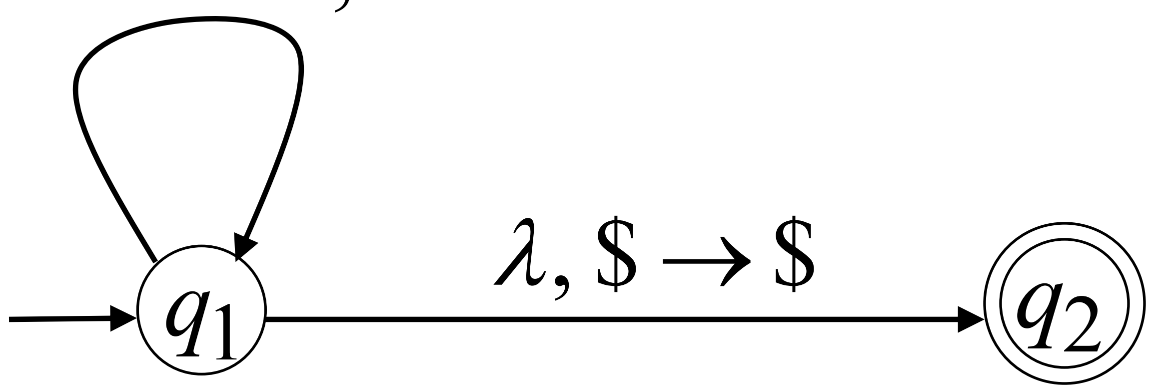
NPDA M

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

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

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

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



Execution Example: Time 0

Input

a	b	b	a	a	b

Stack

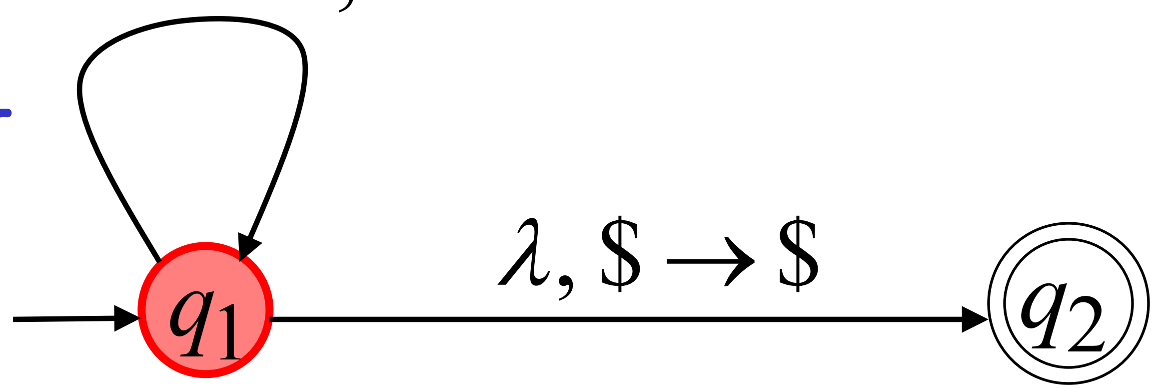
$\$$

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

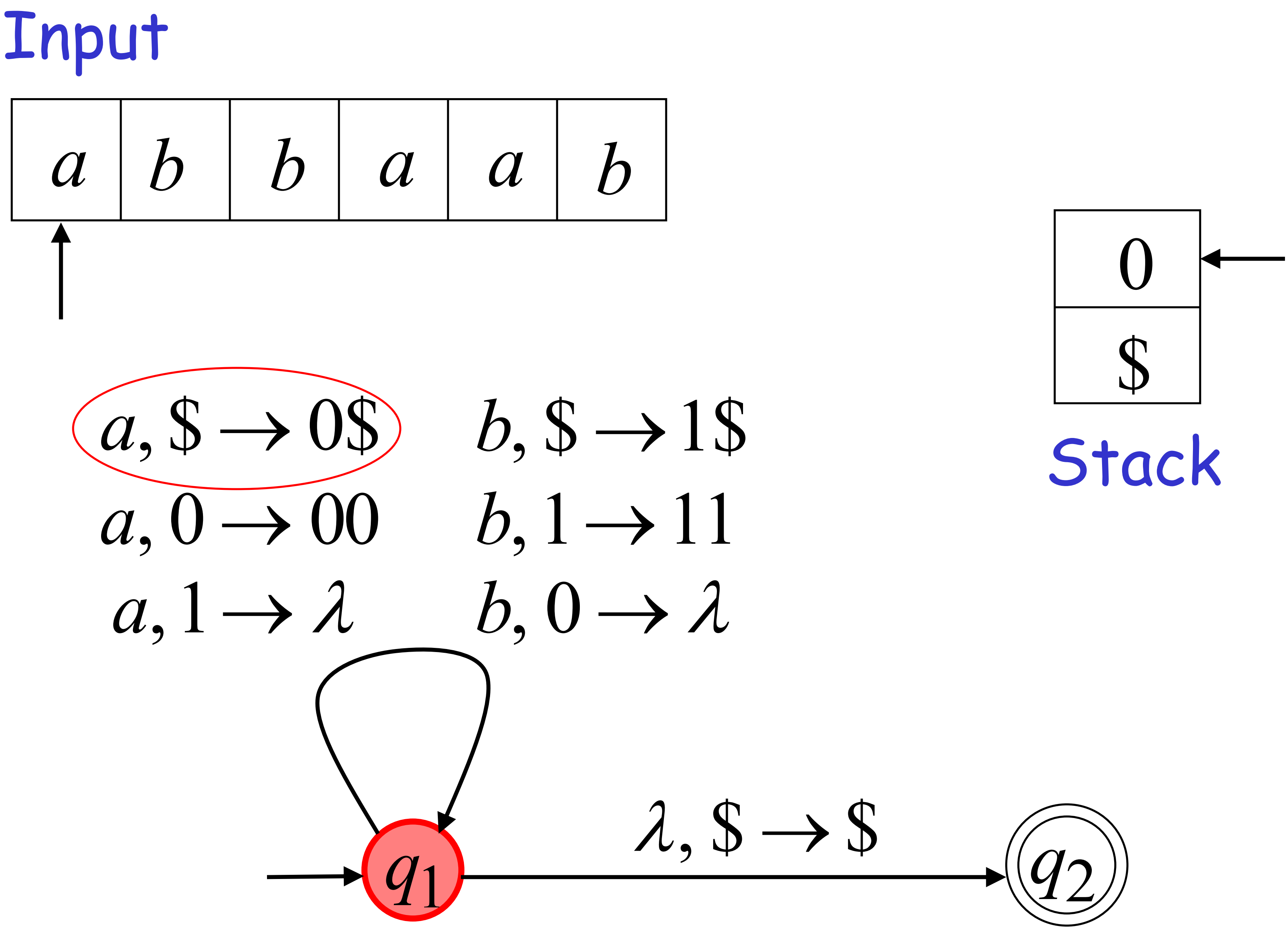
$$a, 0 \rightarrow 00 \qquad b, 1 \rightarrow 11$$

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

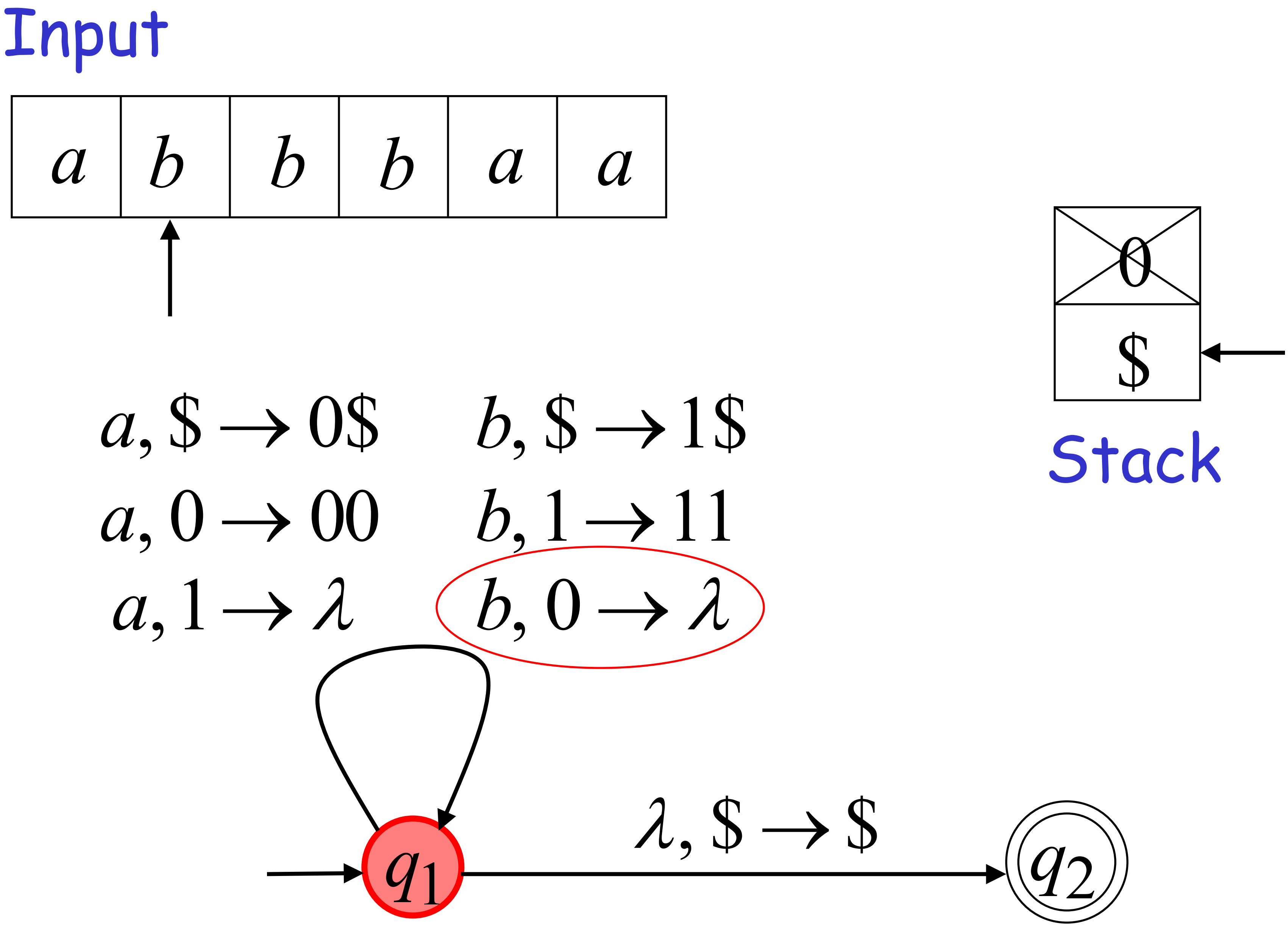
current state



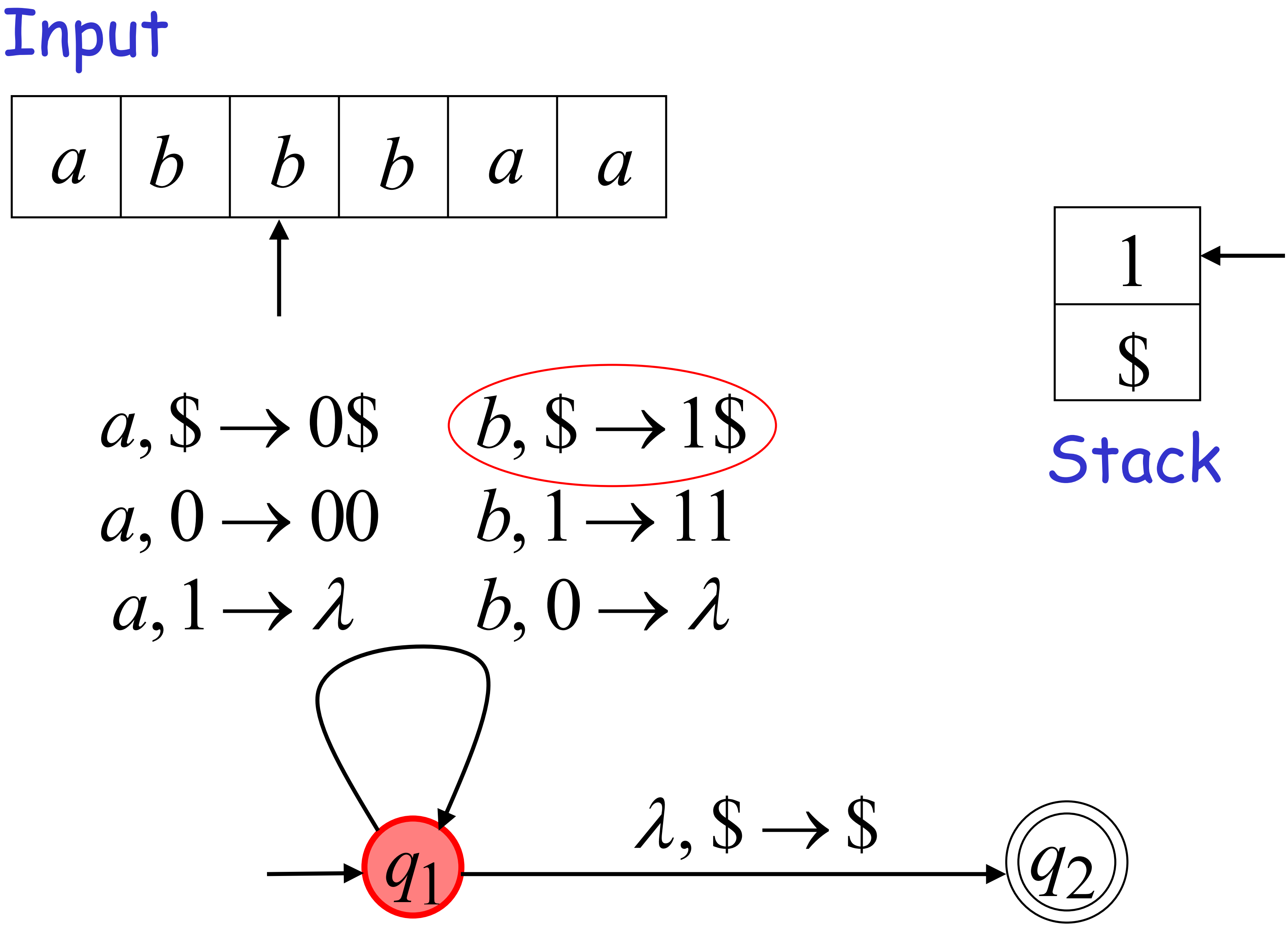
Time 1


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

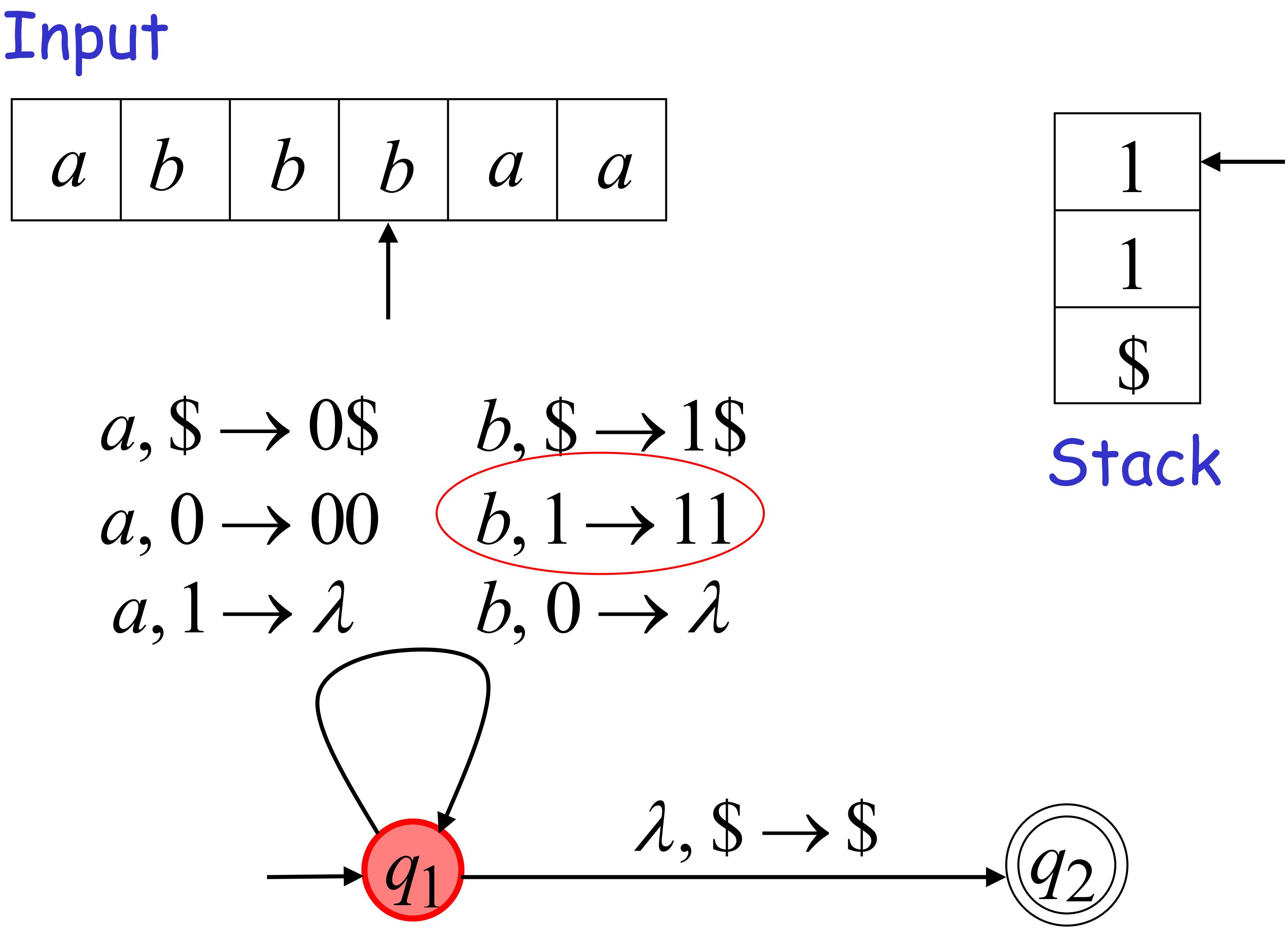
Time 3


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

Time 4



Time 5



Time 6

Input

a	b	b	b	a	a

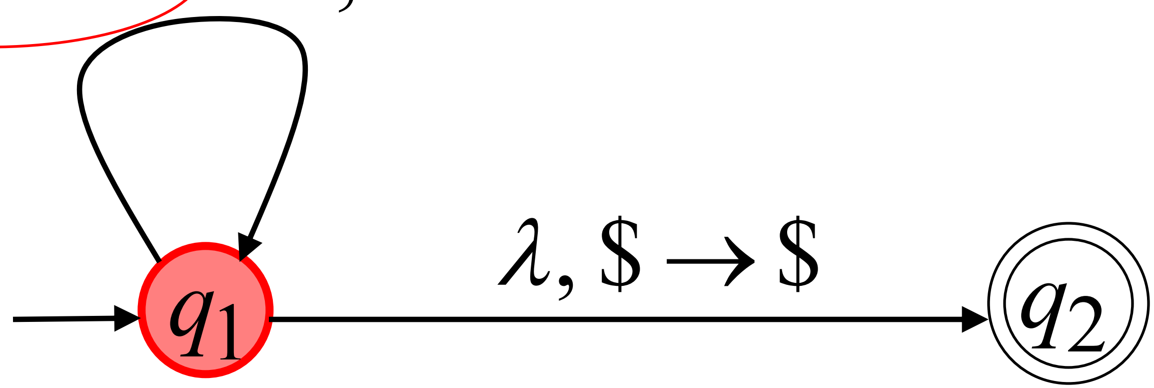
Stack

$\cancel{1}$
1
$\$$

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

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

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



Time 7

Input

a	b	b	b	a	a

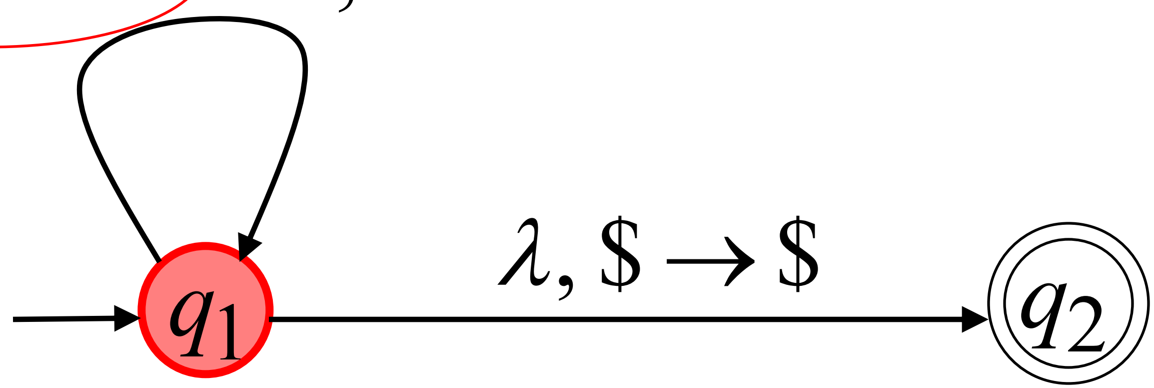
Stack

$\cancel{1}$
$\$$

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

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

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



Time 8

Input

a	b	b	b	a	a

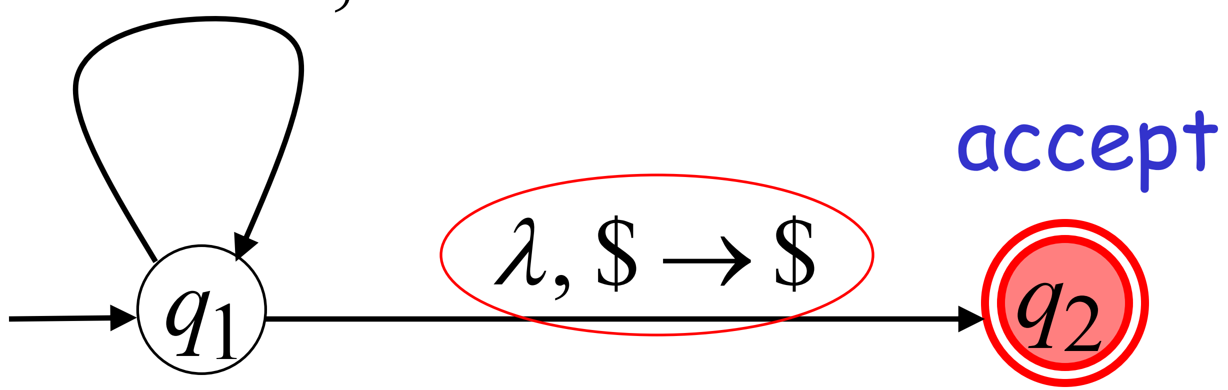
$\$$

Stack

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

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

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



Construct NPDAs

$L_1 = L(aaa^*b)$

$L_2 = L(aab^*aba^*)$

$L_3 = L_1 \cup L_2$

$L_4 = \{a^n b^{2n} : n >= 0\}$

$L_5 = \{a^n b^m c^{n+m} : m, n >= 0\}$

$L_6 = \{a^n b^{m+n} c^m : m, n >= 1\}$

$L_7 = \{a^3 b^m c^m : m >= 0\}$

$L_8 = \{a^n b^m : 2n <= m <= 3n, \}$

$L_9 = \{w : n_a(w) = n_b(w) + 1 , w \, \varepsilon \{a,b\}^* \}$

$L_{10} = \{w : n_a(w) = 2n_b(w) , w \, \varepsilon \{a,b\}^* \}$

$L_{11} = \{w : n_a(w) + n_b(w) = n_c(w), w \, \varepsilon \{a, b, c\}^* \}$

$L_{12} = \{ab(ab)^n b(ba)^n : n >= 0\}$

Exercise

Is it possible to find a dfa that accepts the same language as the pda

$$M = (\{q_0, q_1\}, \{a, b\}, \{z\}, \delta, q_0, z, \{q_1\}),$$

with

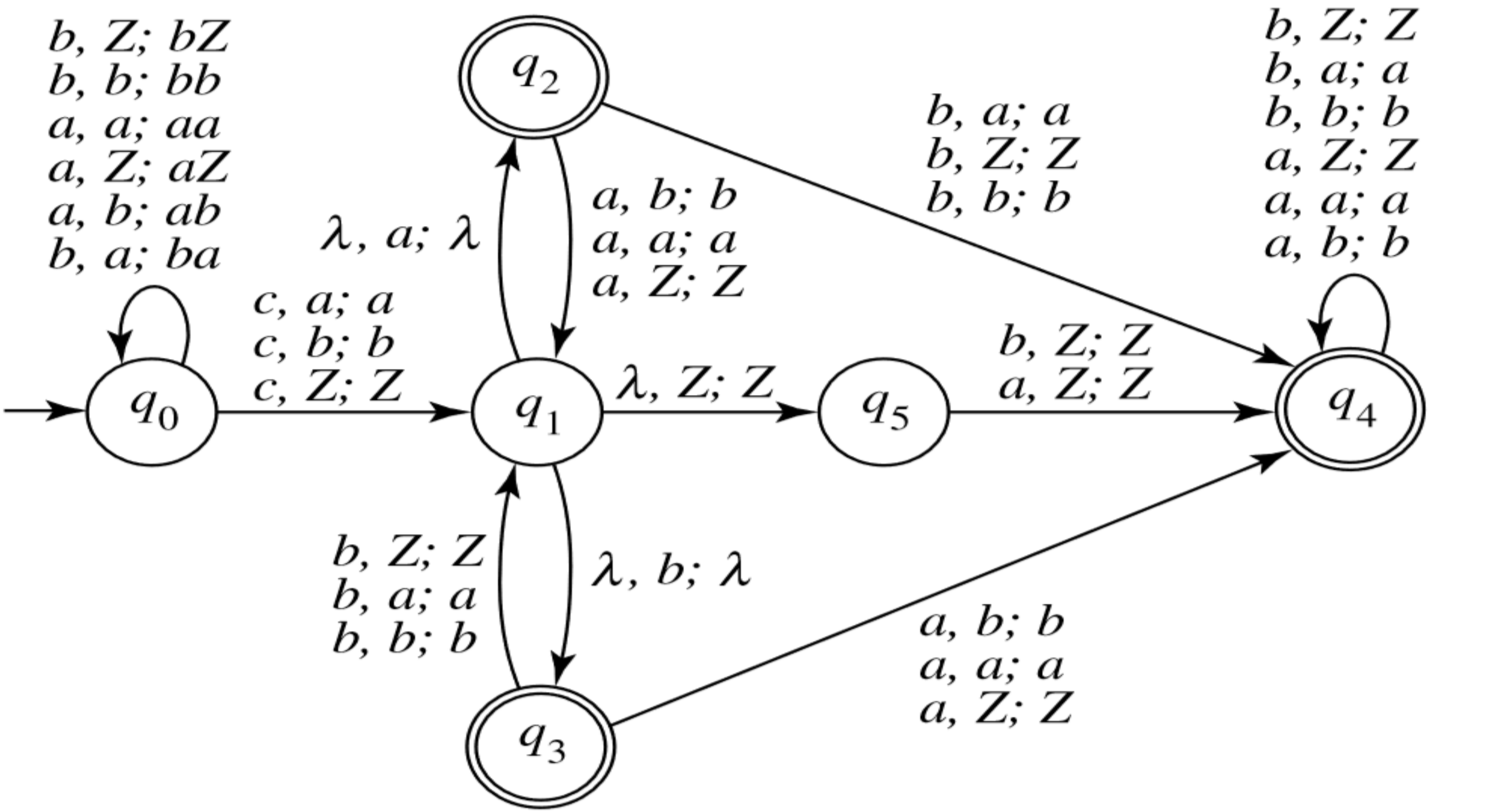
$$\begin{aligned}
\delta(q_0, a, z) &= \{(q_1, z)\},\\
\delta(q_0, b, z) &= \{(q_0, z)\},\\
\delta(q_1, a, z) &= \{(q_1, z)\},\\
\delta(q_1, b, z) &= \{(q_0, z)\}?
\end{aligned}$$

What language is accepted by the npda $M = (\{q_0, q_1, q_2\}, \{a, b\}, \{a, b, z\}, \delta, q_0, z, \{q_2\})$ with transitions

$$\begin{aligned}
\delta(q_0, a, z) &= \{(q_1, a), (q_2, \lambda)\},\\
\delta(q_1, b, a) &= \{(q_1, b)\},\\
\delta(q_1, b, b) &= \{(q_1, b)\},\\
\delta(q_1, a, b) &= \{(q_2, \lambda)\}?
\end{aligned}$$

Exercise

Language: $L = \{w_1 c w_2 : w_1, w_2 \in \{a, b\}^*, w_1 \neq w_2\}$



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

In general:

Given any grammar G

We can construct a NPDA M

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

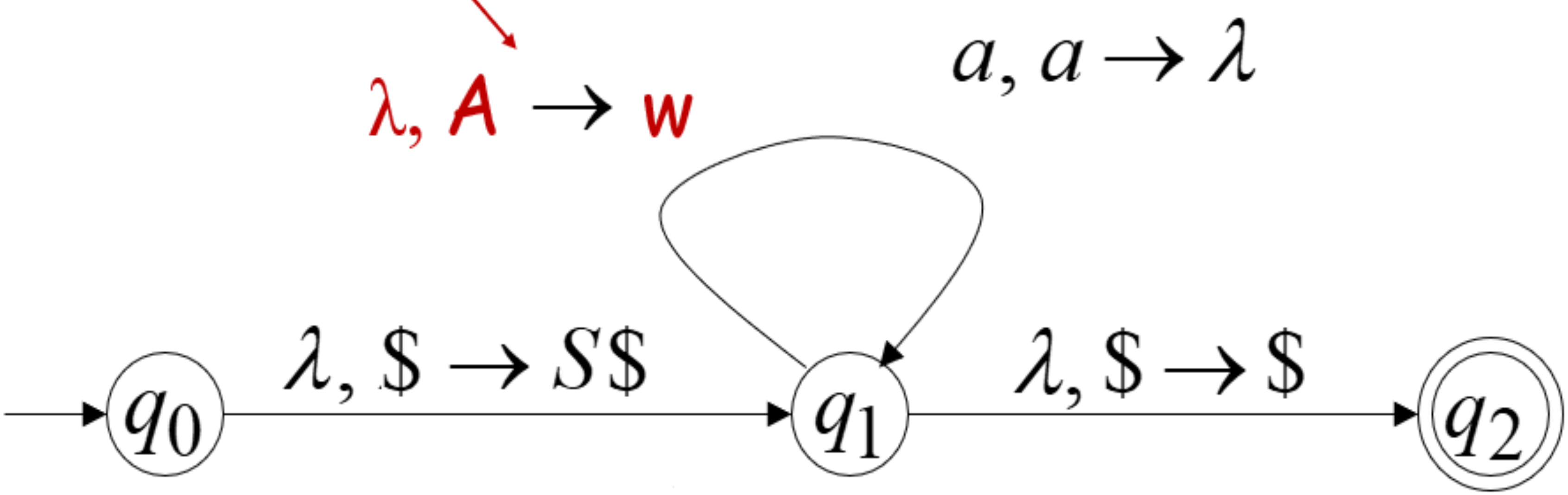
Constructing NPDA M from grammar G :

For any production

$A \to w$

For any terminal

a



Grammar G generates string w

if and only if

NPDA M accepts w

$$\Downarrow$$

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

An example grammar:

$$S \to aSTb$$

$$S \to b$$

$$T \to Ta$$

$$T \to \lambda$$

What is the equivalent NPDA?

Given Grammar:

$$S \to aSTb$$

$$S \to b$$

$$T \to Ta$$

$$T \to \lambda$$

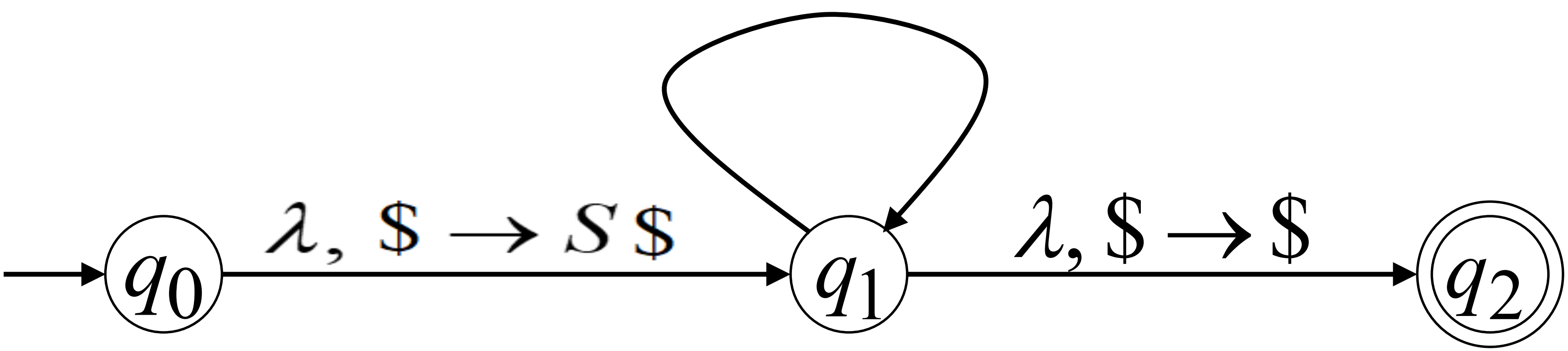
NPDA:

$$\lambda, S \to aSTb$$

$$\lambda, S \to b$$

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

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



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

Many derivations/Computations are possible.

Time 0

$$\lambda, S \rightarrow aSTb$$

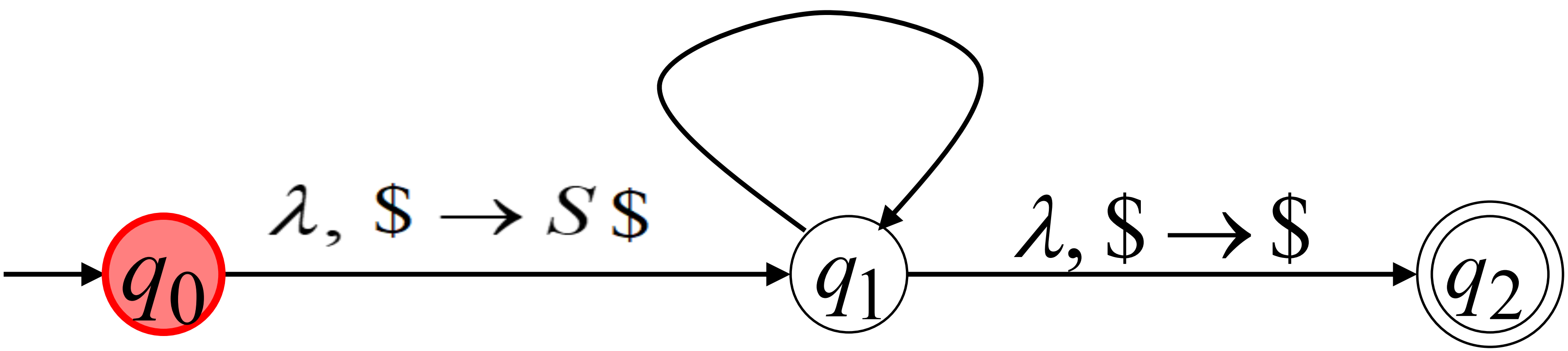
$$\lambda, S \rightarrow b$$

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

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

$\$$

Stack



Derivation: S

Input

a	b	a	b

Time 0

$\lambda, S \to aSTb$

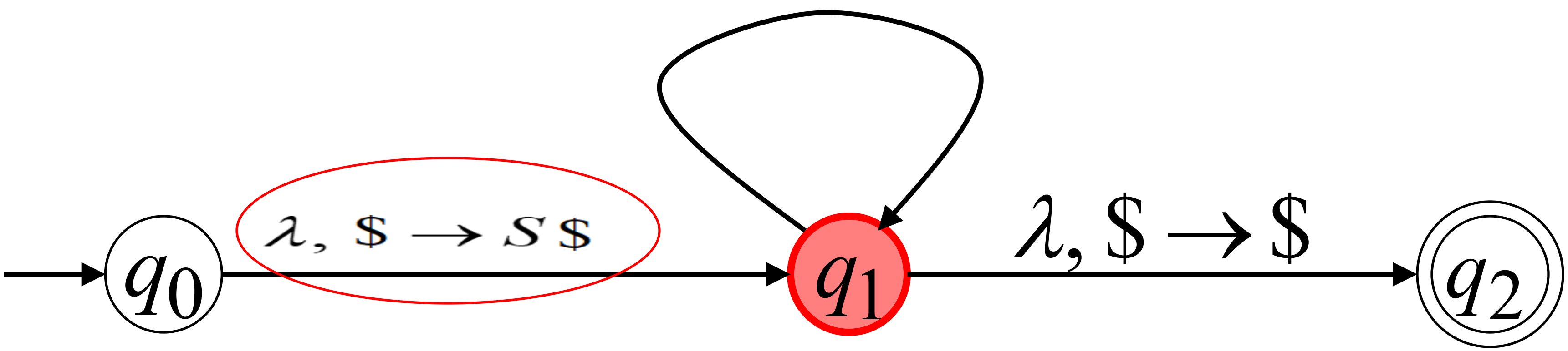
$\lambda, S \to b$

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

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

S
$\$$

Stack



Derivation: $S \Rightarrow aSTb$

Input

a	b	a	b

Time 1

$\lambda, S \rightarrow aSTb$

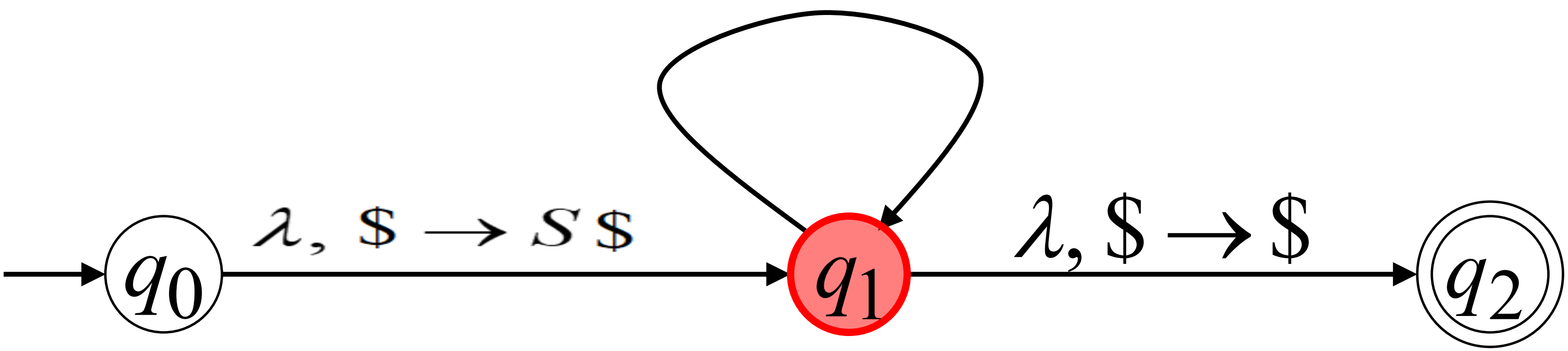
$\lambda, S \rightarrow b$

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

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

a
S
T
b
$\$$

Stack



Derivation: $S \Rightarrow aSTb$

Input

a	b	a	b

Time 2

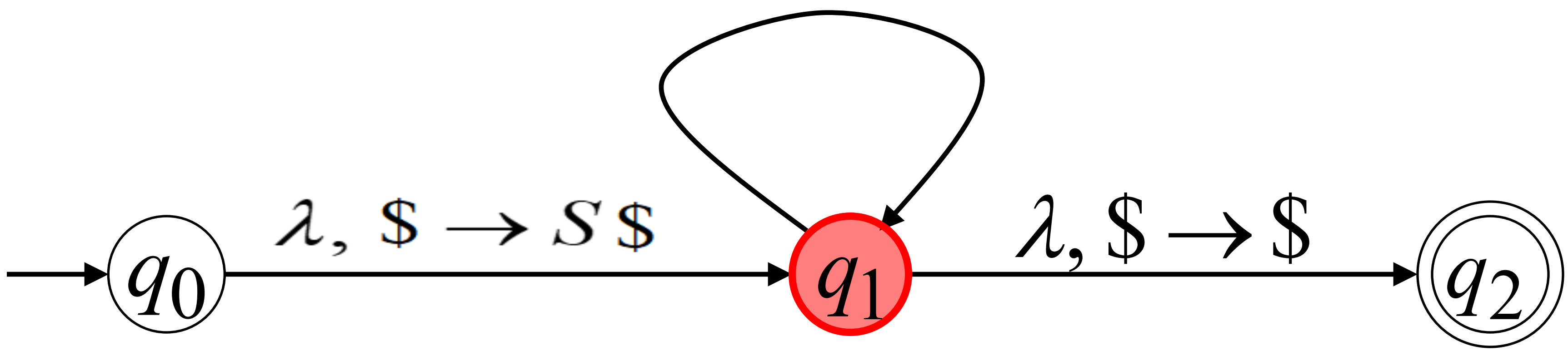
$\lambda, S \rightarrow aSTb$

$\lambda, S \rightarrow b$

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

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

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



Derivation: $S \Rightarrow aSTb \Rightarrow abTb$

Input

a	b	a	b

Time 3

$\lambda, S \rightarrow aSTb$

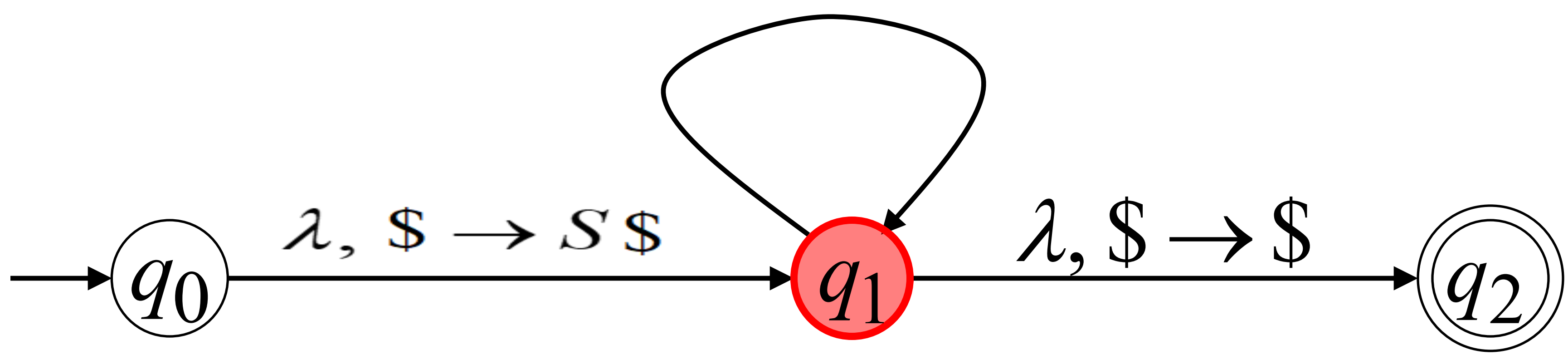
$\lambda, S \rightarrow b$

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

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

b
T
b
$\$$

Stack



Derivation: $S \Rightarrow aSTb \Rightarrow abTb$

Input

a	b	a	b

Time 4

$\lambda, S \rightarrow aSTb$

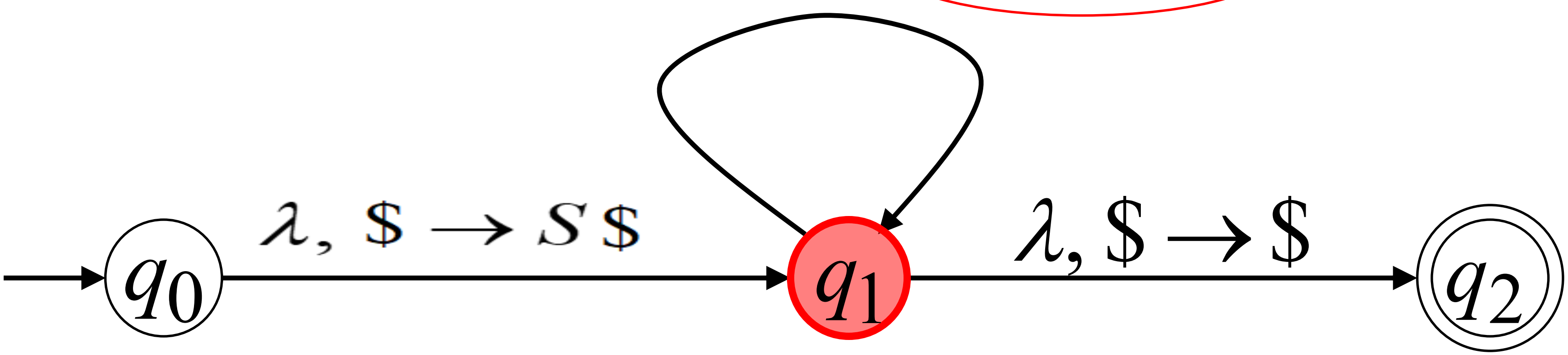
$\lambda, S \rightarrow b$

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

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

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

Stack



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

Input

a	b	a	b

Time 5

$\lambda, S \rightarrow aSTb$

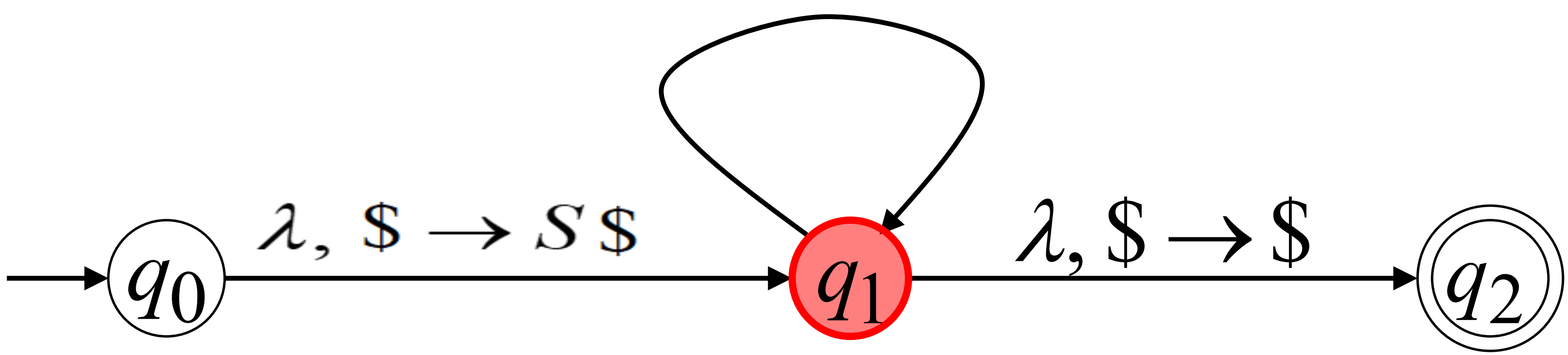
$\lambda, S \rightarrow b$

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

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

T
a
b
$\$$

Stack



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

Input

a	b	a	b

Time 6

$\lambda, S \rightarrow aSTb$

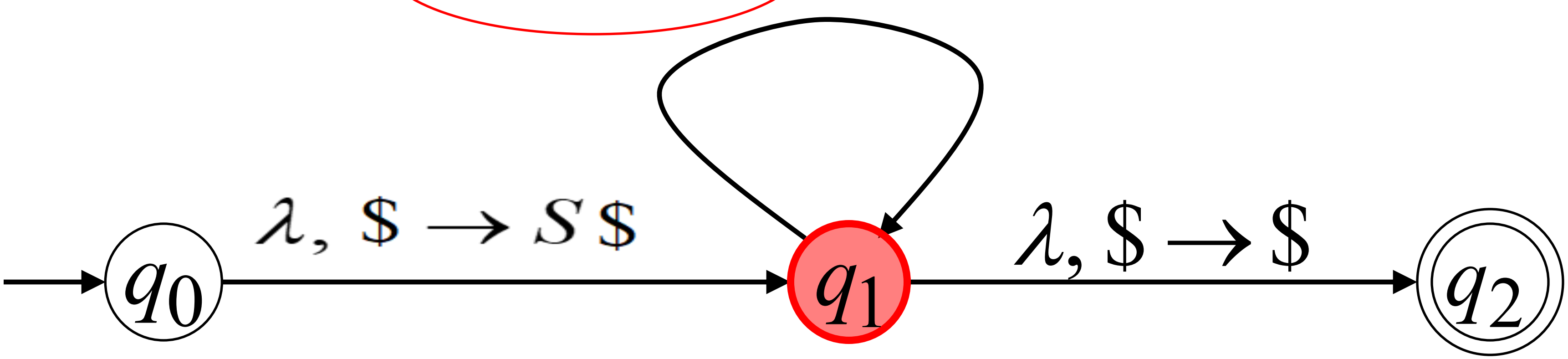
$\lambda, S \rightarrow b$

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

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

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

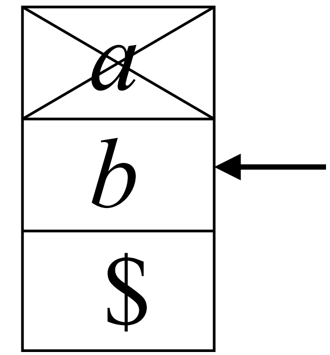
Stack



Derivation: $\underline{S \Rightarrow aSTb \Rightarrow abTb \Rightarrow abTab} \Rightarrow abab$

Input

a	b	a	b



Stack

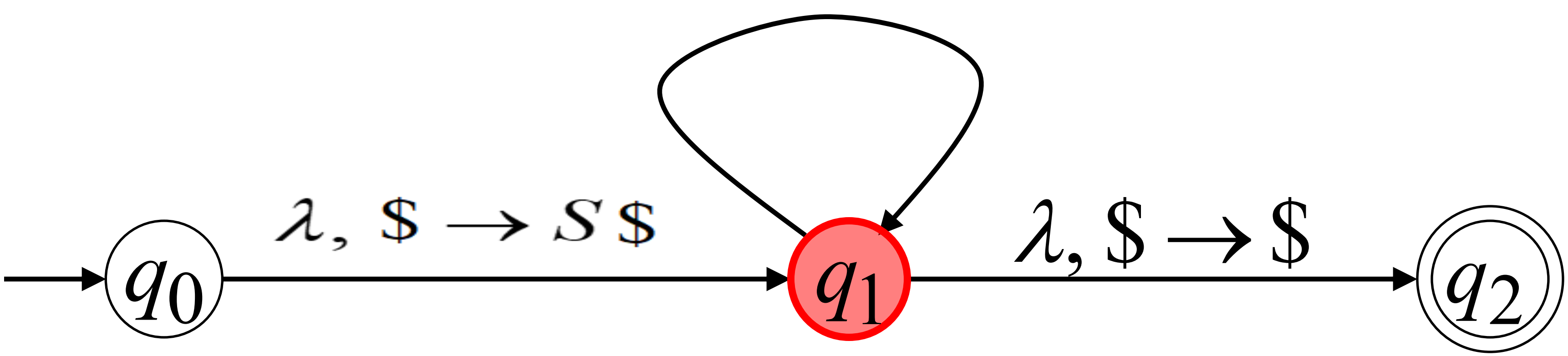
Time 7

$\lambda, S \rightarrow aSTb$

$\lambda, S \rightarrow b$

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

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



Derivation:

Input

a	b	a	b

$\cancel{b}$

$\$$

Stack

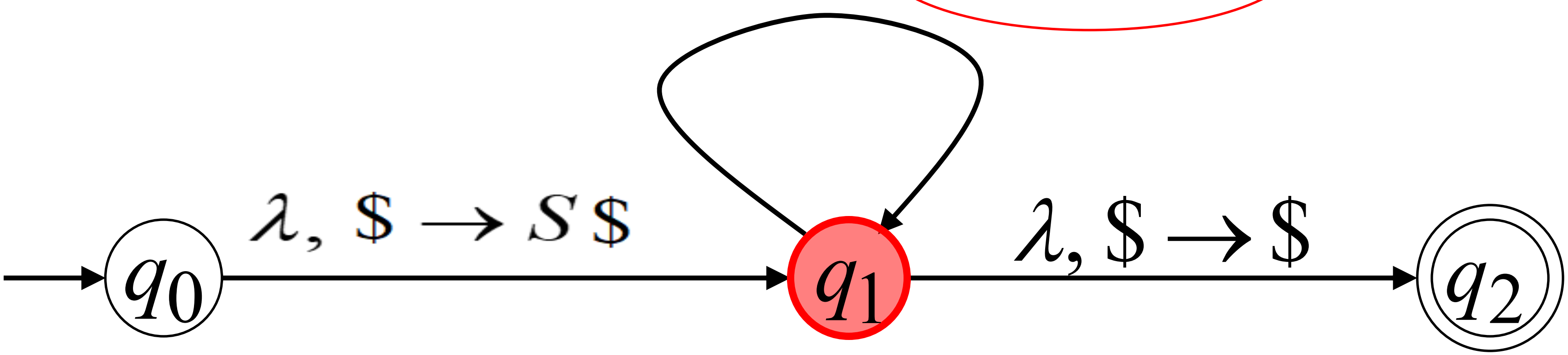
Time 8

$\lambda, S \rightarrow aSTb$

$\lambda, S \rightarrow b$

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

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



Derivation:

Input

a	b	a	b

Time 9

$\lambda, S \rightarrow aSTb$

$\lambda, S \rightarrow b$

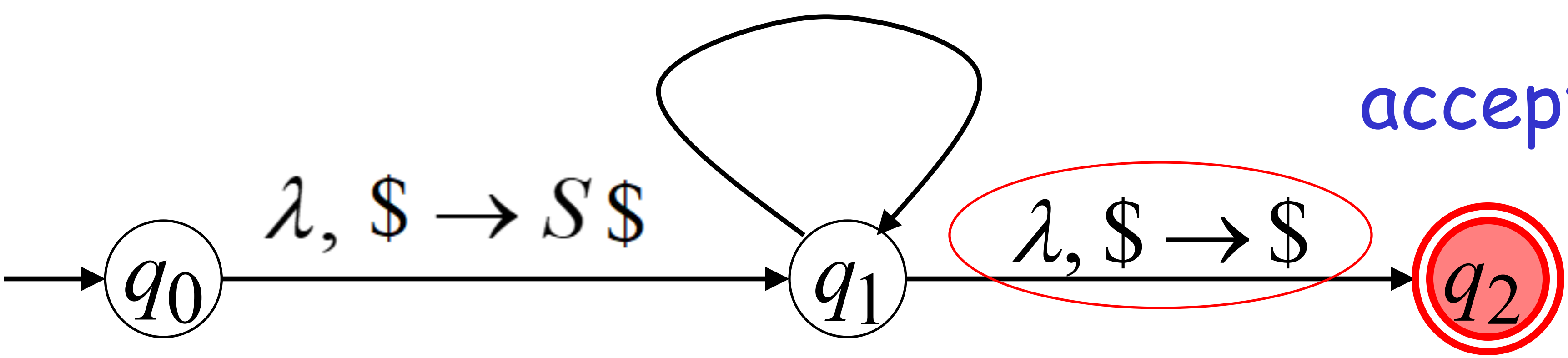
$\lambda, T \rightarrow Ta$ $\quad$ $a, a \rightarrow \lambda$

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

$\$$

Stack

accept



Therefore:

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

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

Construct NPDAs

$$S \to aA,$$
$$A \to aABC\,|bB|\,a,$$
$$B \to b,$$
$$C \to c.$$

$$S \to aSbb|aab.$$

$$S \to aABB|aAA,$$
$$A \to aBB|a,$$
$$B \to bBB|A.$$

$$S \to AA\,|a, A \to SA|\,b.$$

Deterministic Pushdown Automata

DPDA

Deterministic PDA

A PDA is **deterministic** if its transition function satisfies **both** of the following properties

For all $q \in Q$, $a \in (\Sigma \cup \{\varepsilon\})$, and $X \in \Gamma$,

the set $\delta(q, a, X)$ has ***at most*** one element

i.e. there is only one move for any input and stack combination

For all $q \in Q$ and $X \in \Gamma$,

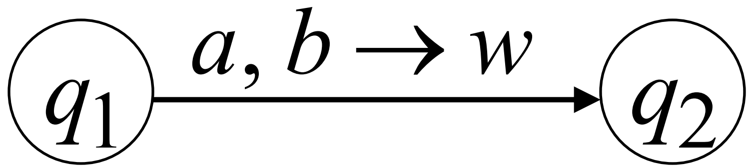
if $\delta(q, \varepsilon, X) \neq \{\}$, then $\delta(q, a, X) = \{\}\ \forall\ a \in \Sigma$

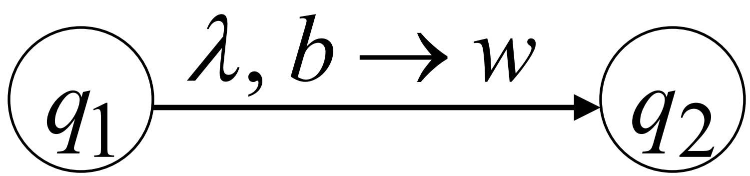
i.e. if there is an ε-transition from state q with X as stack symbol, **then**

there cannot exist another move with stack symbol *X* from the *same state q for any other input symbol.*

Deterministic PDA: DPDA

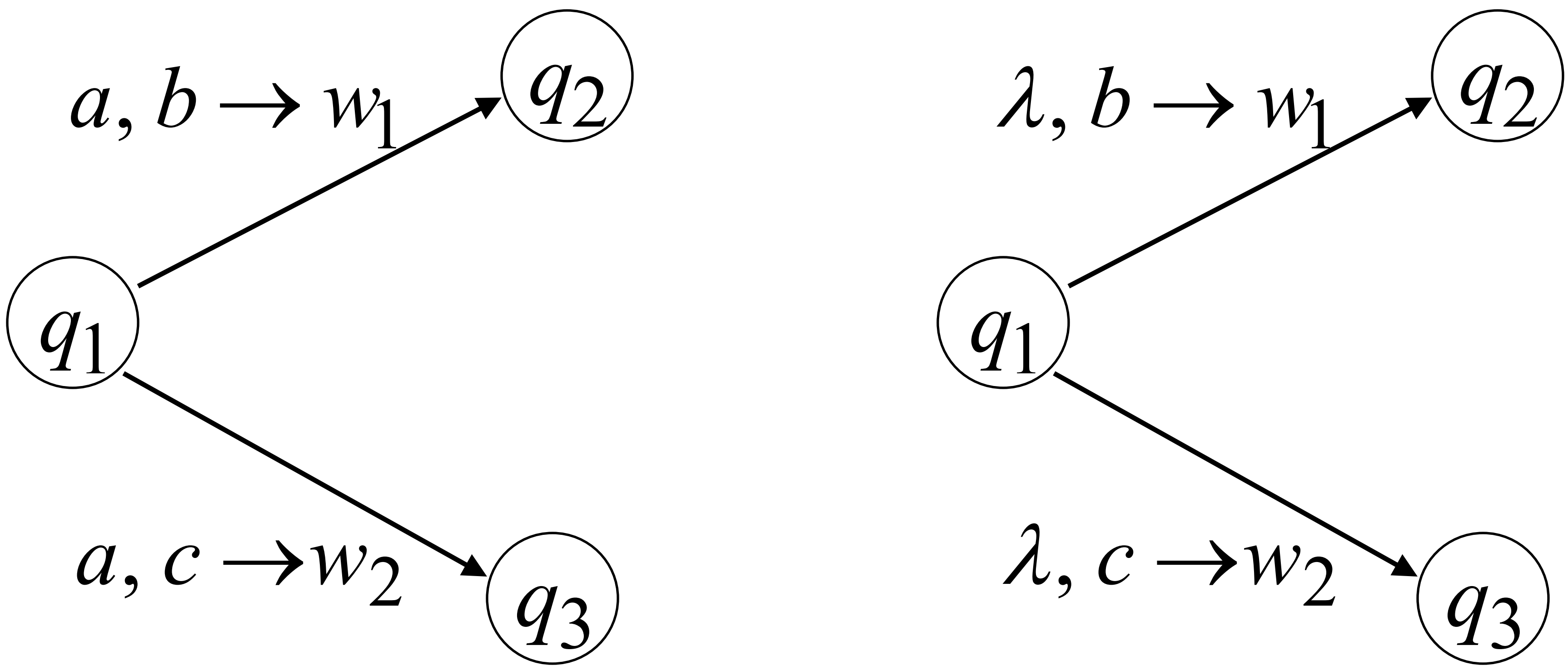
Allowed transitions:





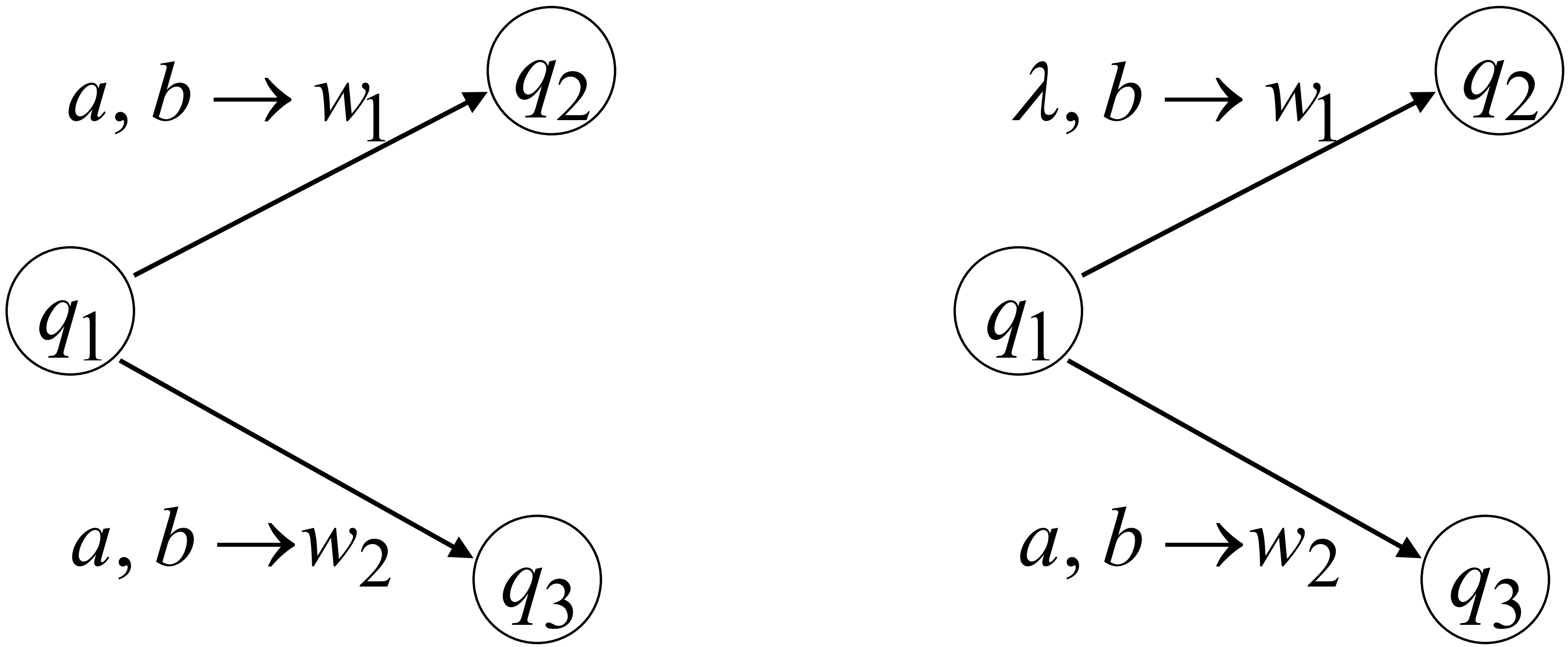
(deterministic choices)

Allowed transitions:



(deterministic choices)

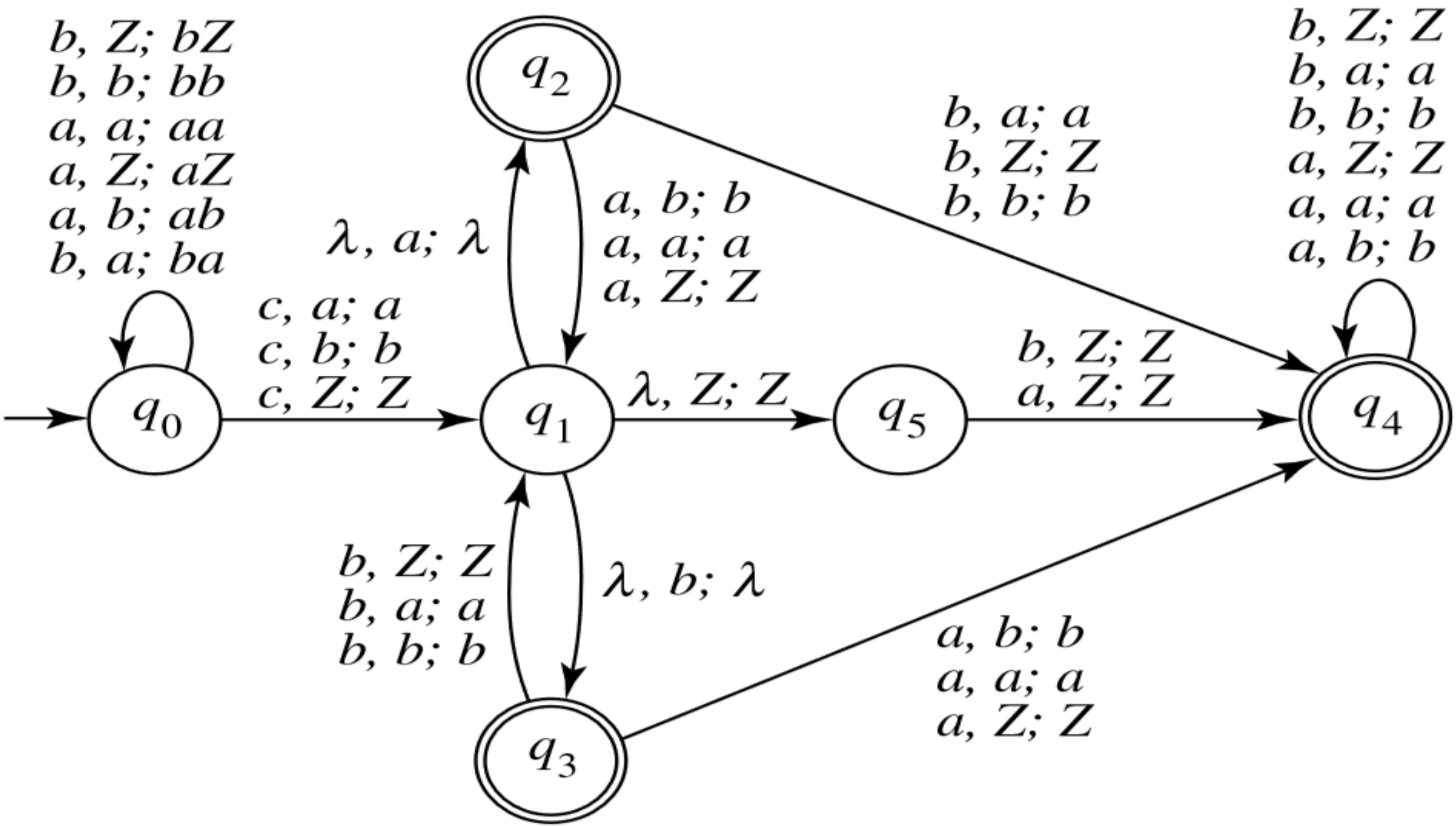
Not allowed in DPDA:



(non deterministic choices)

DPDA ?

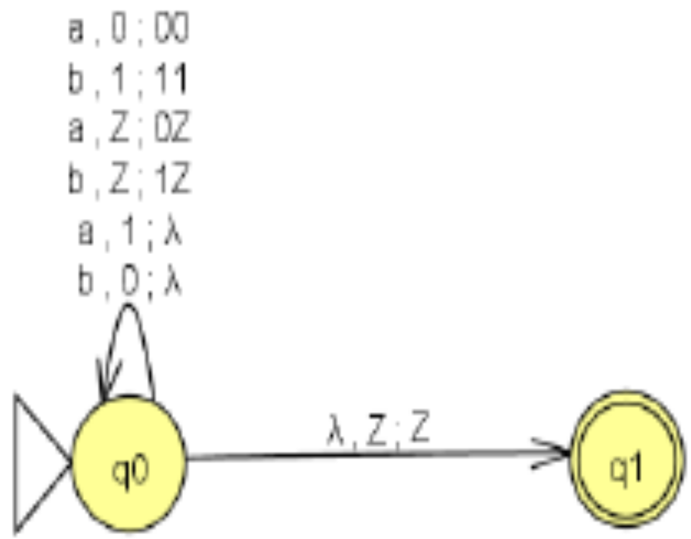
Language: $L = \{w_1 c w_2 : w_1, w_2 \in \{a, b\}^*, w_1 \neq w_2\}$



Non-deterministic PDA
Not DPDA

- 7.3.5 show example 7.4 is a npda but that $L = \{w \in \{a, b\}^* : n_a(w) = n_b(w)\}$ is a deterministic context-free language.
- Machine is npda

$$\delta(q_0, a, 0) = \{(q_0, 00)\}$$
$$\delta(q_0, b, 1) = \{(q_0, 11)\}$$
$$\delta(q_0, a, Z) = \{(q_0, 0Z)\}$$
$$\delta(q_0, b, Z) = \{(q_0, 1Z)\}$$
$$\delta(q_0, a, 1) = \{(q_0, \lambda)\}$$
$$\delta(q_0, b, 0) = \{(q_0, \lambda)\}$$
$$\delta(q_0, \lambda, Z) = \{(q_1, Z)\}$$



This is a npda because these transitions violate the 2nd rule associated with dpda,

$$\delta(q_0, \lambda, Z) \neq \emptyset \Rightarrow \forall c \in \Sigma \quad \delta(q_0, c, Z) = \emptyset$$

Deterministic PDA

Some context-free languages which are initially described in a nondeterministic way via a NPDA **can also be described** in a deterministic way via DPDA.

Some context-free languages are inherently nondeterministic, e.g., $L = \{w \in (a|b)^* : w = w^R\}$ **cannot be accepted by any dpda**.

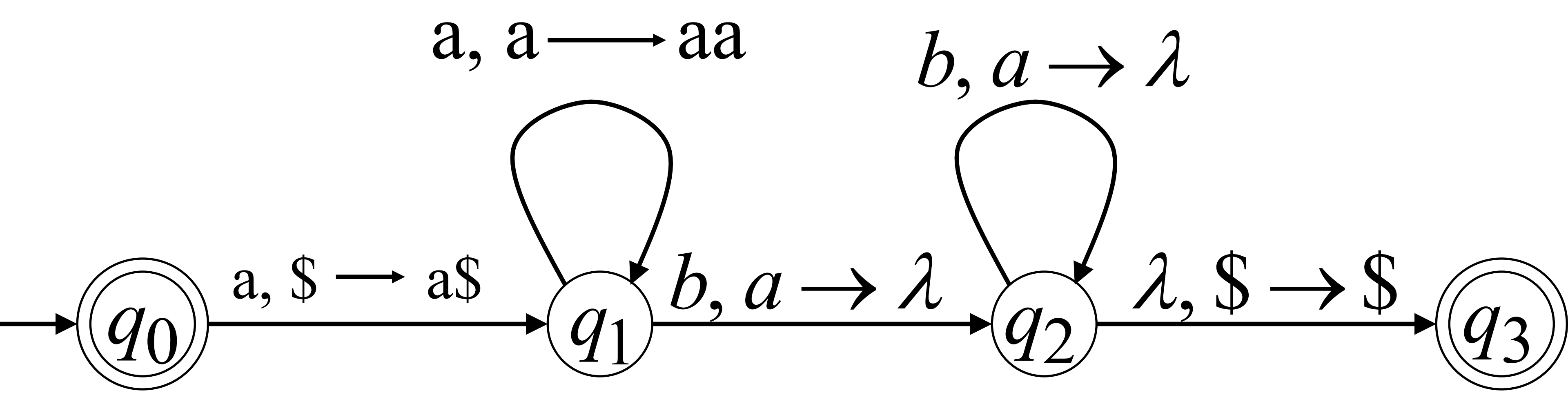
Deterministic PDA (DPDA) can only represent a subset of CFL, e.g., $L = \{ww^R \mid w \in (a|b)^*\}$ **cannot be represented by DPDA**

A key point in all this is that the **deterministic pushdown automata is not equivalent to nondeterministic pushdown automata**.

Unless otherwise stated, we assume that a PDA is nondeterministic

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

Deterministic PDA

- 7.3.1 show $L = \{a^n b^{2n} : n \geq 0\}$ is a deterministic context-free language
- Per definition 7.4 we need to find a dpda that accepts L
- $M = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$
- $M = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \{a, Z\}, \delta, q_0, Z, \{q_0, q_3\})$

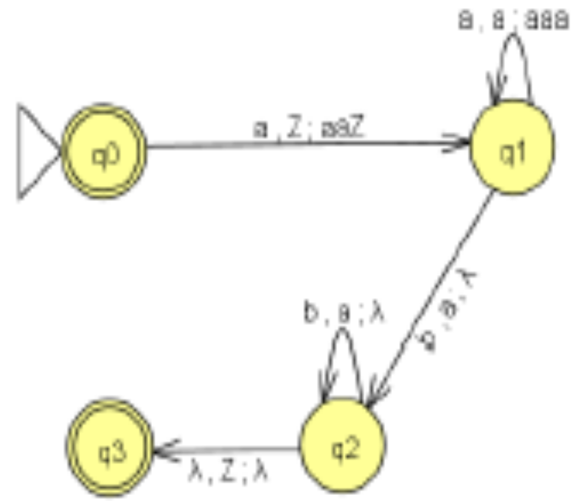
  $\delta(q_0, a, Z) = \{(q_1, aaZ)\}$

  $\delta(q_1, a, a) = \{(q_1, aaa)\}$

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

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

  $\delta(q_2, \lambda, Z) = \{(q_3, \lambda)\}$



- Operation of dpda
  - state q_0 accepts λ, if input is a, push 2 a's and goto q_1
  - state q_1 pushes 2 a's for each input a , if input is b, pop a and goto q_2
  - state q_2 pops a for each b, λ-move to q_3 if Z on top and no more input
  - state q_3 accepts $a^n b^{2n} : n > 0$

Non-deterministic PDA

- 7.3.5 show example 7.4 is a npda but that $L = \{w \in \{a, b\}^* : n_a(w) = n_b(w)\}$ is a deterministic context-free language.
- Machine is npda

$\delta(q_0, a, 0) = \{(q_0, 00)\}$

$\delta(q_0, b, 1) = \{(q_0, 11)\}$

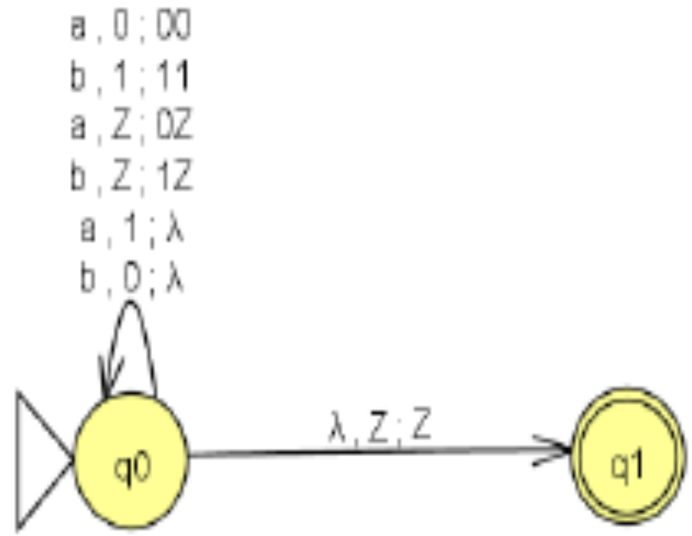
$\delta(q_0, a, Z) = \{(q_0, 0Z)\}$

$\delta(q_0, b, Z) = \{(q_0, 1Z)\}$

$\delta(q_0, a, 1) = \{(q_0, \lambda)\}$

$\delta(q_0, b, 0) = \{(q_0, \lambda)\}$

$\delta(q_0, \lambda, Z) = \{(q_1, Z)\}$



This is a npda because these transitions violate the 2nd rule associated with dpda,

$\delta(q_0, \lambda, Z) \neq \emptyset \Rightarrow \forall c \in \Sigma \quad \delta(q_0, c, Z) = \emptyset$

Deterministic PDA

- 7.3.5 (continued) show that $L = \{w \in \{a, b\}^* : n_a(w) = n_b(w)\}$ is a deterministic context-free language. $\delta(q, \lambda, b) \neq \emptyset \Rightarrow \forall c \in \Sigma \quad \delta(q, c, b) = \emptyset$
- A dpda that accepts L; $M = (\{q_0, q_1, q_2\}, \{a, b\}, \{a, b, Z\}, \delta, q_0, Z, \{q_0\})$

$\delta(q_0, a, Z) = \{(q_1, aZ)\}$

$\delta(q_0, b, Z) = \{(q_2, bZ)\}$

$\delta(q_1, a, a) = \{(q_1, aa)\}$

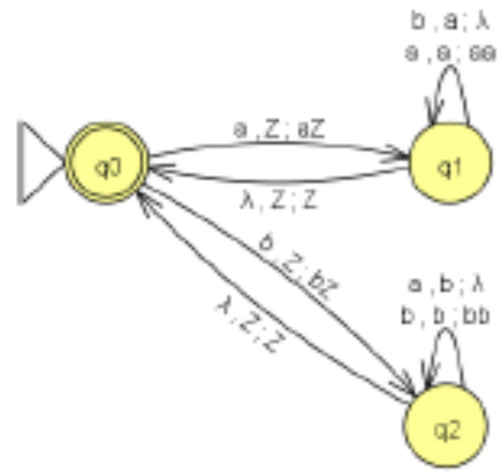
$\delta(q_1, b, a) = \{(q_1, \lambda)\}$

$\delta(q_1, \lambda, Z) = \{(q_0, Z)\}$

$\delta(q_2, b, b) = \{(q_2, bb)\}$

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

$\delta(q_2, \lambda, Z) = \{(q_0, \lambda)\}$



- Operation of dpda
  - state q_0 accepts strings in the language ($n_a(w) = n_b(w)$ including λ)
  - state q_1 adds a's and subtracts b's; on empty stack λ-move back to q_0
  - state q_2 adds b's and subtracts a's; on empty stack λ-move back to q_0

Non-Deterministic PDA

- Language $L = \{w \in \{a, b\}^* : n_a(w) > n_b(w)\}$ accepted via a npda

$\delta(q_0, a, Z) = \{(q_0, aZ)\}$

$\delta(q_0, b, Z) = \{(q_0, bZ)\}$

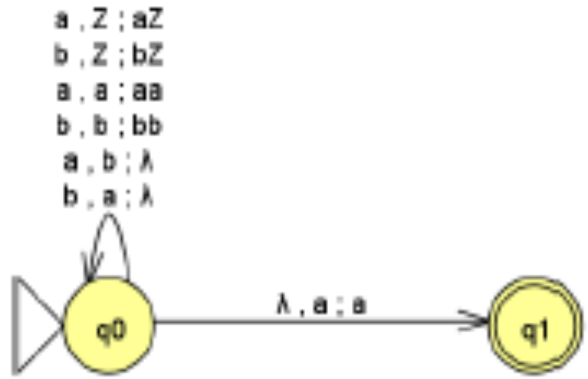
$\delta(q_0, a, a) = \{(q_0, aa)\}$

$\delta(q_0, b, b) = \{(q_0, bb)\}$

$\delta(q_0, a, b) = \{(q_0, \lambda)\}$

$\delta(q_0, b, a) = \{(q_0, \lambda)\}$

$\delta(q_0, \lambda, a) = \{(q_1, a)\}$



This is a npda because these transitions violate the 2nd rule associated with dpda,

$\delta(q, \lambda, b) \neq \emptyset \Rightarrow \forall c \in \Sigma \quad \delta(q, c, b) = \emptyset$

- Operation of npda
  - start in state q_0, read first symbol and push it onto the stack, then...
  - if input and stack symbols match, push both symbols onto the stack
  - if input and stack symbols differ, discard both
  - when no more input, if a on top of stack, λ-move to accepting state q_1

Non-Deterministic PDA

- Same language $L = \{w \in \{a, b\}^* : n_a(w) > n_b(w)\}$ accepted via a dpda

$\delta(q_0, a, Z) = \{(q_1, Z)\}$

$\delta(q_0, b, Z) = \{(q_0, bZ)\}$

$\delta(q_0, a, b) = \{(q_0, \lambda)\}$

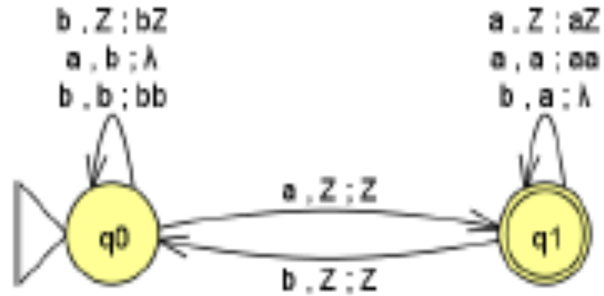
$\delta(q_0, b, b) = \{(q_0, bb)\}$

$\delta(q_1, a, Z) = \{(q_1, aZ)\}$

$\delta(q_1, b, Z) = \{(q_0, Z)\}$

$\delta(q_1, a, a) = \{(q_1, aa)\}$

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



$L = \{w \in \{a, b\}^* : n_a(w) > n_b(w)\}$ is accepted by a dpda. By definition 7.4, it is therefore a deterministic context-free language

- Operation of dpda
  - state q_0 means $n_a(w) \le n_b(w)$
  - state q_1 means $n_a(w) > n_b(w)$
  - jump between states based on input and current top of stack
  - when input ends, halt; q_1 is accepting state

Non-Deterministic PDA

$L = \{a^n \mid n \geq 0\} \cup \{a^n b^n \mid n \geq 0\}$ is CFL and accepted by a **non-deterministic** PDA M

$Q = \{q_0, q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{Z, A\}$, q_0, Z, $F = \{q_2\}$

$\delta(q_0, a, Z) = \{(q_0, AZ)\}$

$\delta(q_0, \varepsilon, Z) = \{(q_2, \varepsilon)\}$

$\delta(q_0, \varepsilon, Z) = \{(q_2, \varepsilon)\}$

$\delta(q_2, \varepsilon, Z) = \{(q_2, \varepsilon)\}$

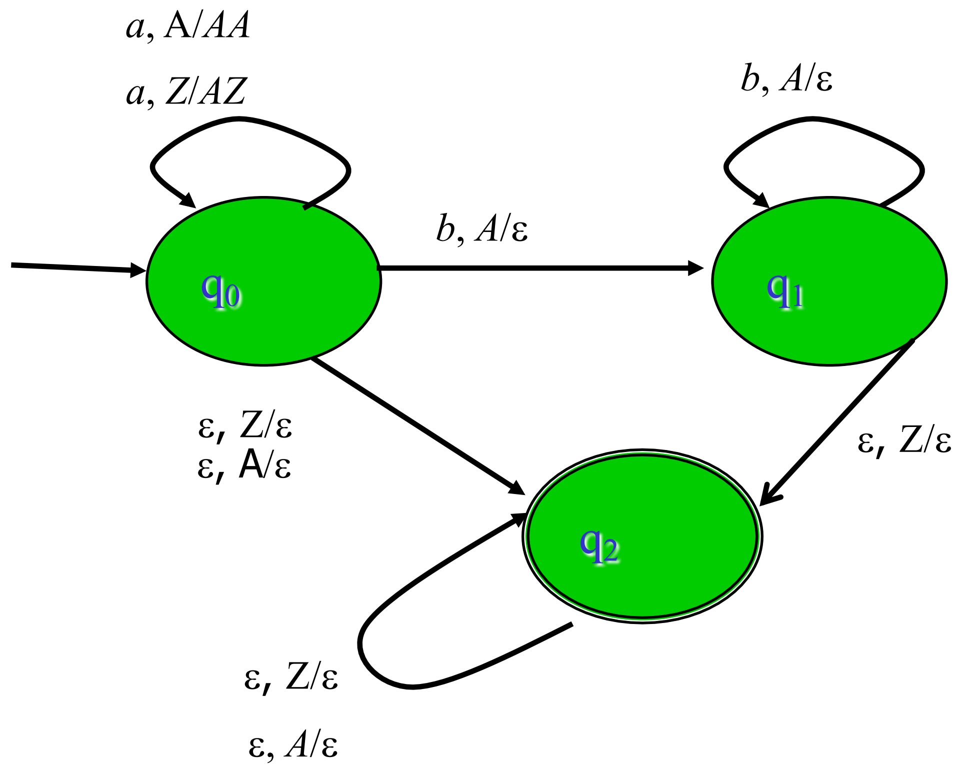
$\delta(q_0, a, A) = \{(q_0, AA)\}$

$\delta(q_0, b, A) = \{(q_1, \varepsilon)\}$

$\delta(q_1, b, A) = \{(q_1, \varepsilon)\}$

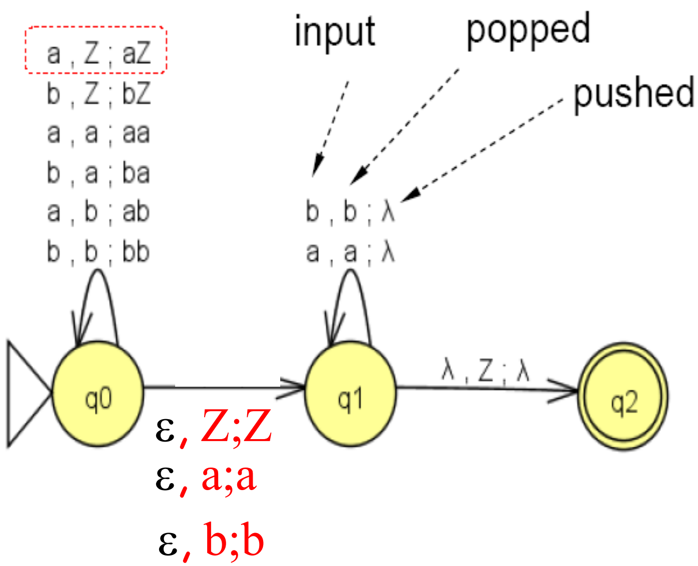
$\delta(q_1, \varepsilon, Z) = \{(q_1, \varepsilon)\}$

$\delta(q_2, \varepsilon, A) = \{(q_2, \varepsilon)\}$



Non-Deterministic PDA

The language of (strings over {a, b} of *even* length and spelled the same forwards and backwards) = $\{ww^R \mid w \in \{a, b\}^*\}$ is CFL and accepted by a **non-deterministic** PDA M



Exercise

- Show that the following languages are DCFL.
  - $L_1 = \{a^n b^{2n} : n >= 0\}$
  - $L_2 = \{a^n b^m : m >= n+2\}$
  - $L_3 = \{a^n b^n : n >= 1\} \cup \{b\}$

- Show that every regular language is DCFL.

- Are the following languages DCFL?
  - $L_4 = \{a^n b^n : n >= 1\} \cup \{a\}$
  - $L_5 = \{wcw^R : w \in \{a,b\}^*\}$
  - $L_6 = \{w : n_a(w) = 2n_b(w), w \in \{a,b\}^*\}$
  - $L_7 = \{w : n_a(w) + n_b(w) = n_c(w), w \in \{a, b, c\}^*\}$
  - $L_8 = \{ab(ab)^n b(ba)^n : n >= 0\}$

NPDAs

Have More Power than

DPDAs

It holds that:

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

Since every DPDA is also a NPDA

We will actually show:

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

$L \notin$ (Deterministic Context-Free Languages (DPDA))

$L \in$ (Context-Free Languages (NPDA))

We will show that there exists
a context-free language L which is not
accepted by any DPDA

The language is:

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

We will show:

- L is context-free

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

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

Language L is context-free

Context-free grammar for L:

$$S \to S_1 \mid S_2 \qquad \{a^n b^n\} \cup \{a^n b^{2n}\}$$

$$S_1 \to a S_1 b \mid \lambda \qquad \{a^n b^n\}$$

$$S_2 \to a S_2 bb \mid \lambda \qquad \{a^n b^{2n}\}$$

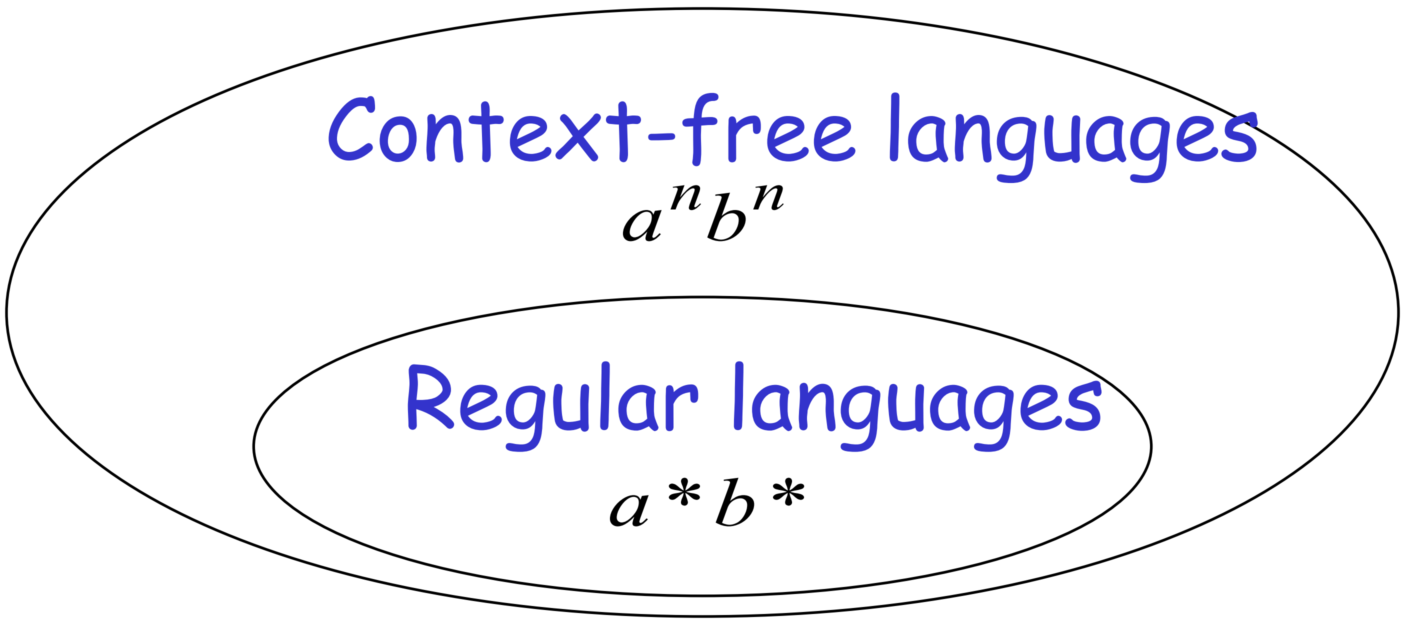
Theorem:

The language $L = \{a^n b^n\} \cup \{a^n b^{2n}\}$

is **not** deterministic context-free

(there is **no** DPDA that accepts L)

Fact 1: The language $\{a^n b^n c^n\}$ is not context-free



(we will prove this at a later class using pumping lemma for context-free languages)

Fact 2: The language $L \cup \{a^n b^n c^n\}$ is **not** context-free

$$(L = \{a^n b^n\} \cup \{a^n b^{2n}\})$$

(we can prove this using pumping lemma for context-free languages)

Thank You