CS116-Automata Theory and Formal Languages

Lecture 10

PDAs Accept Context-Free Languages

Computer Science Department

1st Semester 2025-2026

Theorem:

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

Proof - Step 1:

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

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

Proof - Step 2:

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

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

Proof - step 1

Convert

Context-Free Grammars
to
PDAs

Take an arbitrary context-free grammar G

We will convert G to a PDA M such that:

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

Conversion Procedure:

For each production in G

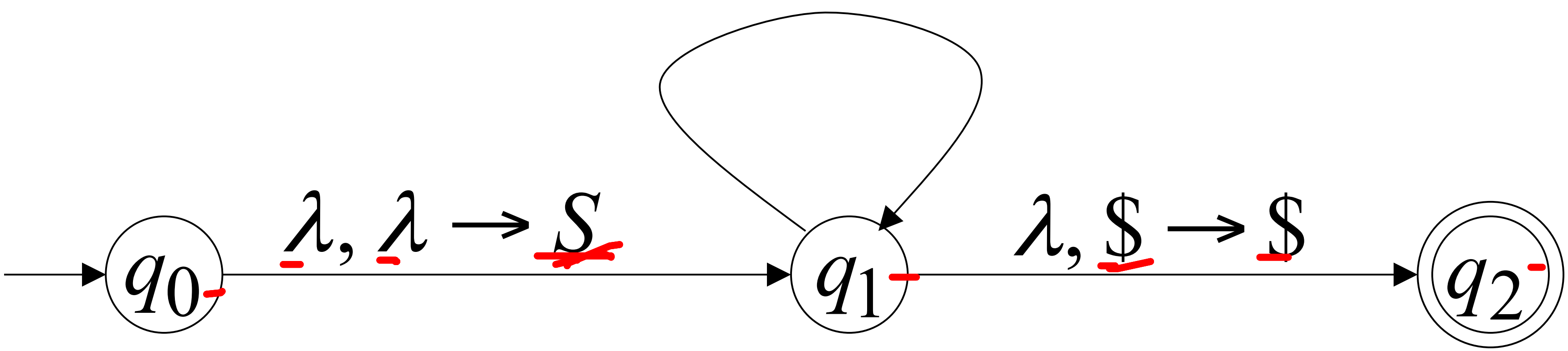
$$A \rightarrow w$$

For each terminal in G

$$a$$

Add transitions

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



Example

Grammar

$S \rightarrow aSTb$

$S \rightarrow b$

$T \rightarrow Ta$

$T \rightarrow \lambda$

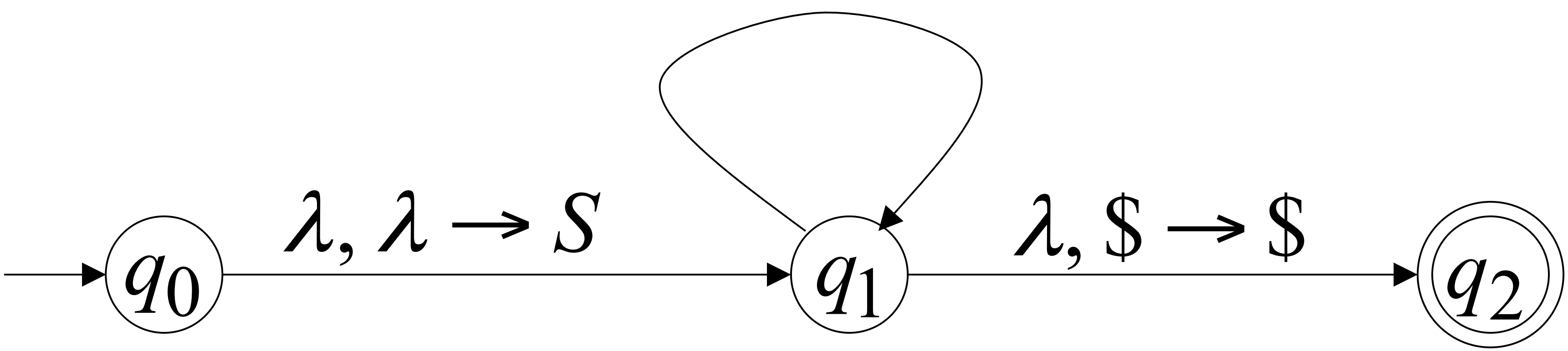
PDA

$\lambda, S \rightarrow aSTb$

$\lambda, S \rightarrow b$

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

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



PDA simulates leftmost derivations

Grammar Leftmost Derivation

S

$\Rightarrow \cdots$

$\Rightarrow \sigma_1 \cdots \sigma_k X_1 \cdots X_m$

$\Rightarrow \cdots$

$\Rightarrow \sigma_1 \cdots \sigma_k \sigma_{k+1} \cdots \sigma_n$

PDA Computation

$(q_0, \sigma_1 \cdots \sigma_k \sigma_{k+1} \cdots \sigma_n, \$)$

$\succ (q_1, \sigma_1 \cdots \sigma_k \sigma_{k+1} \cdots \sigma_n, S\$)$

$\succ \cdots$

$\succ (q_1, \sigma_{k+1} \cdots \sigma_n, X_1 \cdots X_m \$)$

$\succ \cdots$

$\succ (q_2, \lambda, \$)$

Scanned symbols

Stack contents

Grammar

Leftmost Derivation

Terminals

Leftmost variable

Variables or terminals

$$\Rightarrow \cdots$$

$$\Rightarrow xAy$$

$$\Rightarrow x\sigma_i \cdots \sigma_j Bzy$$

Production applied

$$A \rightarrow \underbrace{\sigma_i \cdots \sigma_j}_{\text{Terminals}} Bz$$

Variable

Variables or terminals

Grammar

Leftmost Derivation

PDA Computation

$a, a \to \lambda$

$$\Rightarrow \cdots \qquad \succ \cdots$$

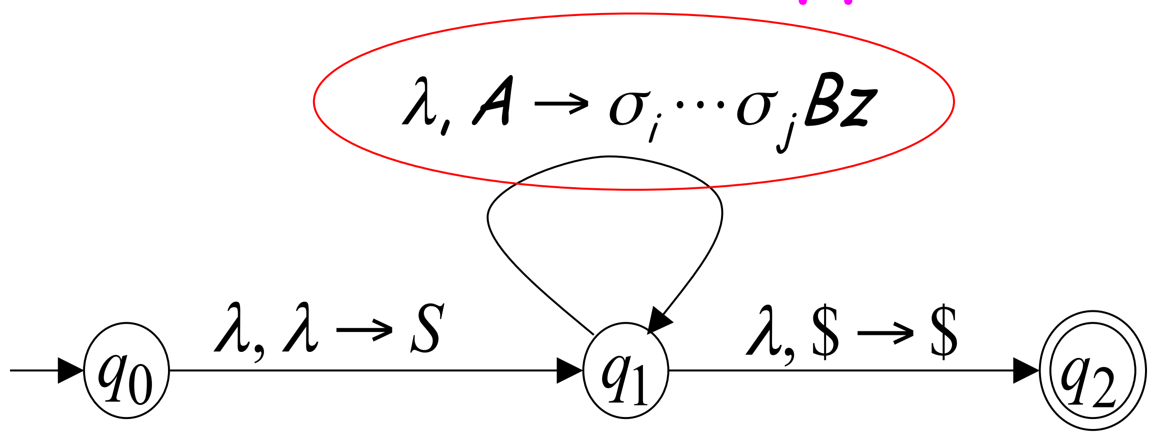
$$\Rightarrow xAy \qquad \succ (q_1, \sigma_i \cdots \sigma_n, Ay\$)$$

$$\Rightarrow x\sigma_i \cdots \sigma_j Bzy \qquad \succ (q_1, \sigma_i \cdots \sigma_n, \sigma_i \cdots \sigma_j Bzy\$)$$

Production applied

$$A \to \sigma_i \cdots \sigma_j Bz$$

Transition applied



Grammar Leftmost Derivation

$\Rightarrow \cdots$

$\Rightarrow xAy$

$\Rightarrow x\sigma_i \cdots \sigma_j Bzy$

PDA Computation

$\succ \cdots$

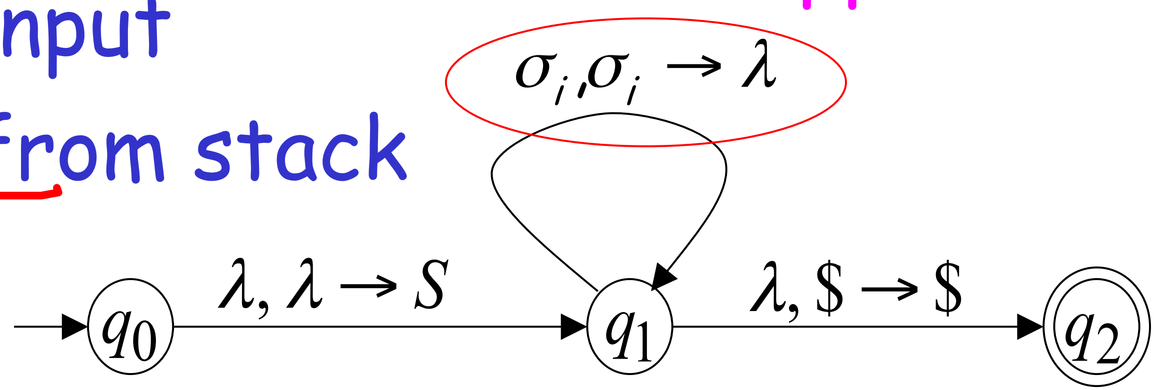
$\succ (q_1, \sigma_i \cdots \sigma_n, Ay\$)$

$\succ (q_1, \sigma_i \cdots \sigma_n, \sigma_i \cdots \sigma_j Bzy\$)$

$\succ (q_1, \sigma_{i+1} \cdots \sigma_n, \sigma_{i+1} \cdots \sigma_j Bzy\$)$

Read σ_i from input and remove it from stack

Transition applied



Grammar Leftmost Derivation

$\Rightarrow \cdots$

$\Rightarrow xAy$

$\Rightarrow x\sigma_i \cdots \sigma_j Bzy$

PDA Computation

$\succ \cdots$

$\succ (q_1, \sigma_i \cdots \sigma_n, Ay\$)$

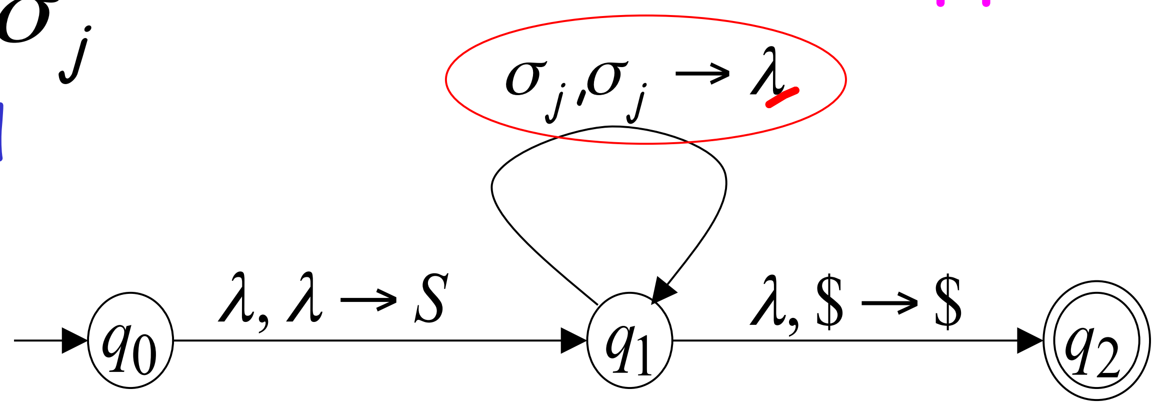
$\succ (q_1, \sigma_i \cdots \sigma_n, \sigma_i \cdots \sigma_j Bzy\$)$

$\succ (q_1, \sigma_{i+1} \cdots \sigma_n, \sigma_{i+1} \cdots \sigma_j Bzy\$)$

$\succ \cdots$

$\succ (q_1, \sigma_{j+1} \cdots \sigma_n, Bzy\$)$

Last Transition applied



All symbols $\sigma_i \cdots \sigma_j$ have been removed from top of stack

The process repeats with the next leftmost variable

$\Rightarrow \cdots$

$\Rightarrow xAy$

$\Rightarrow x\sigma_i \cdots \sigma_j Bzy$

$\Rightarrow x\sigma_i \cdots \sigma_j \sigma_{j+1} \cdots \sigma_k Cpzy$

$\succ \cdots$

$\succ (q_1, \sigma_{j+1} \cdots \sigma_n, Bzy\$)$

$\succ (q_1, \sigma_{j+1} \cdots \sigma_n, \sigma_{j+1} \cdots \sigma_k Cpzy\$)$

$\succ \cdots$

$\succ (q_1, \sigma_{k+1} \cdots \sigma_n, Cpzy\$)$

Production applied

$B \rightarrow \sigma_{j+1} \cdots \sigma_k Cp$

And so on.....

Example:

Input

a	b	a	b

Time 0

$$\lambda, S \rightarrow aSTb$$

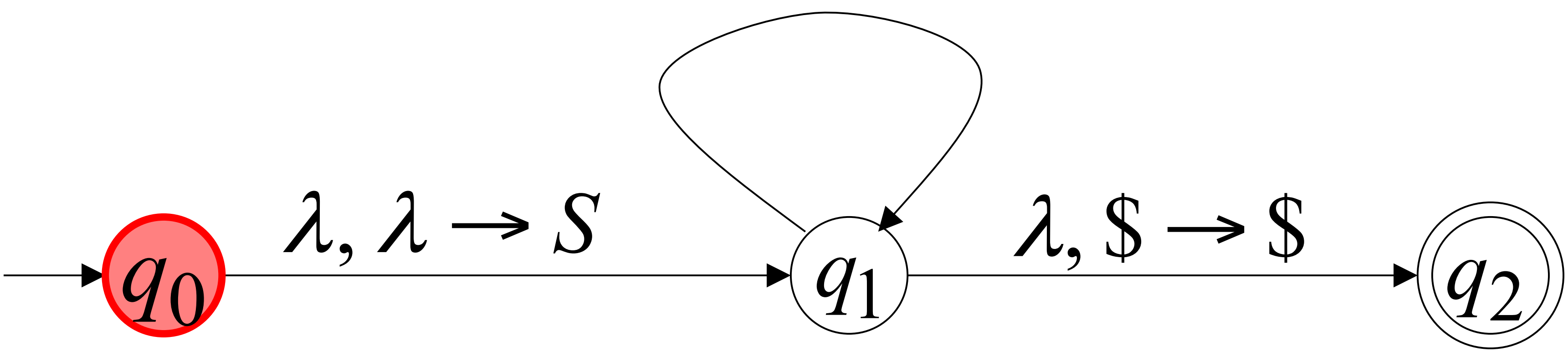
$$\lambda, S \rightarrow b$$

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

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

$\$$

Stack



Derivation: S

Input

a	b	a	b

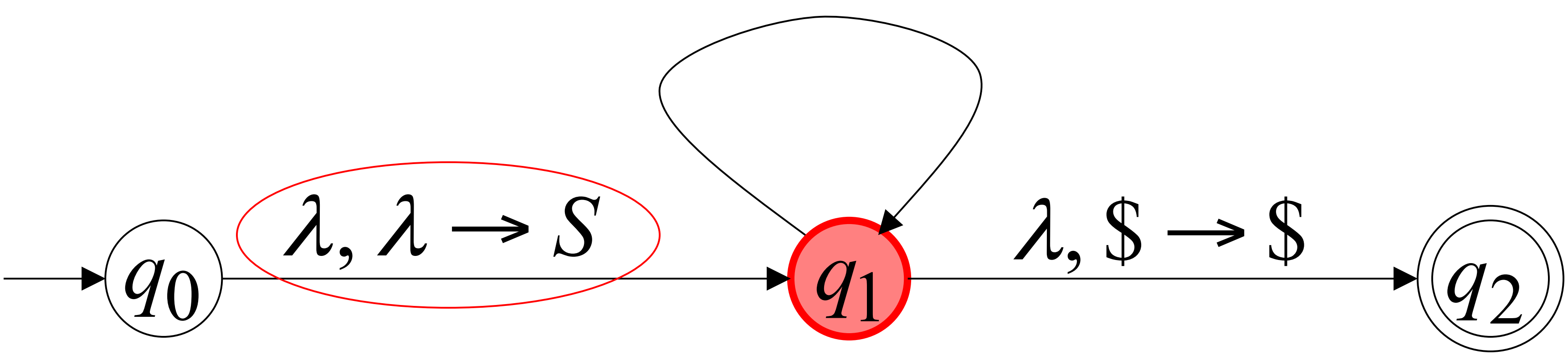
Time 1

$\lambda, S \to aSTb$

$\lambda, S \to b$

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

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



S
$\$$

Stack

Derivation: $S \Rightarrow aSTb$

Input

a	b	a	b

Stack

a
S
T
b
$\$$

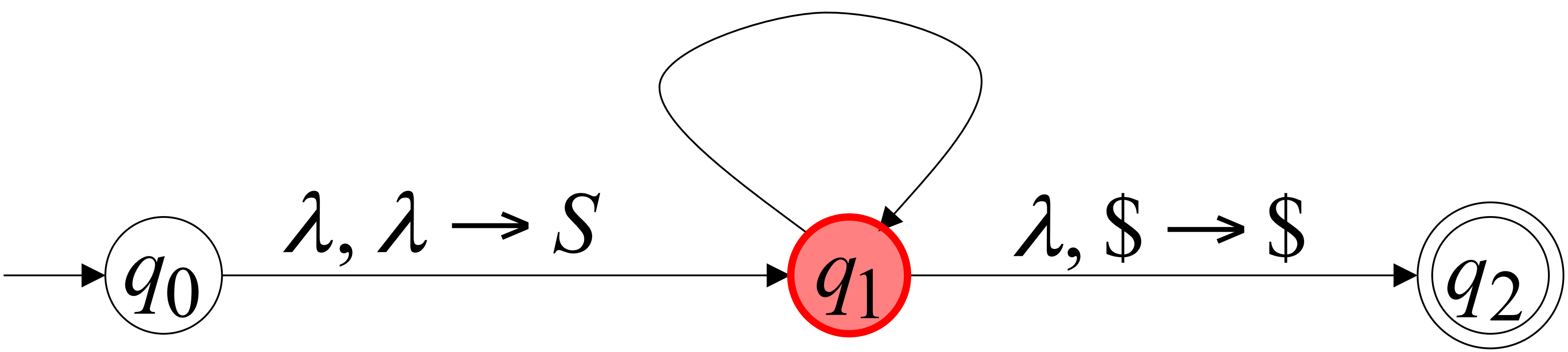
Time 2

$\lambda, S \rightarrow aSTb$

$\lambda, S \rightarrow b$

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

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



Derivation: $S \Rightarrow aSTb$

Input

a	b	a	b

Time 3

$\lambda, S \rightarrow aSTb$

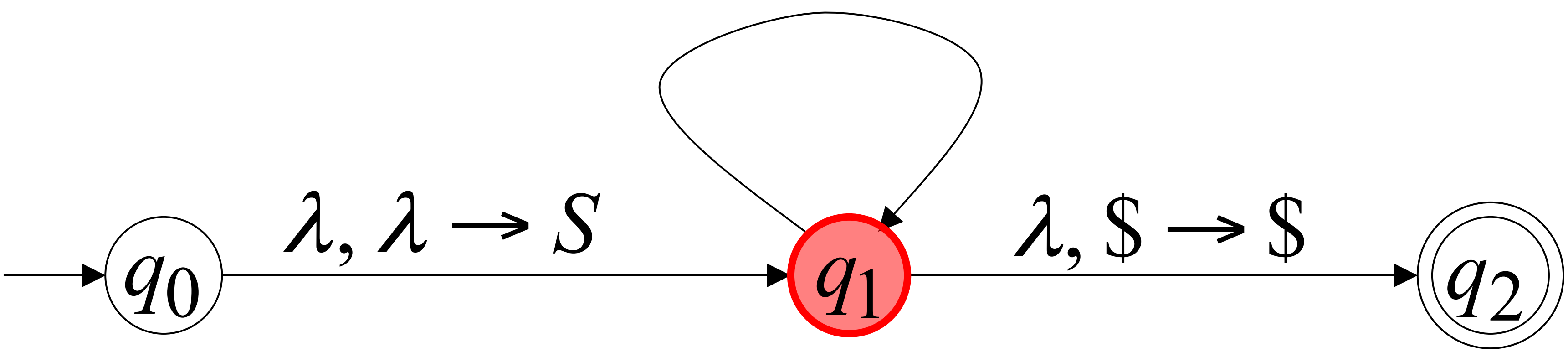
$\cancel{\lambda, S \rightarrow b}$

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

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

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

Stack



Derivation: $S \Rightarrow aSTb \Rightarrow abTb$

Input

a	b	a	b

Time 4

$\lambda, S \rightarrow aSTb$

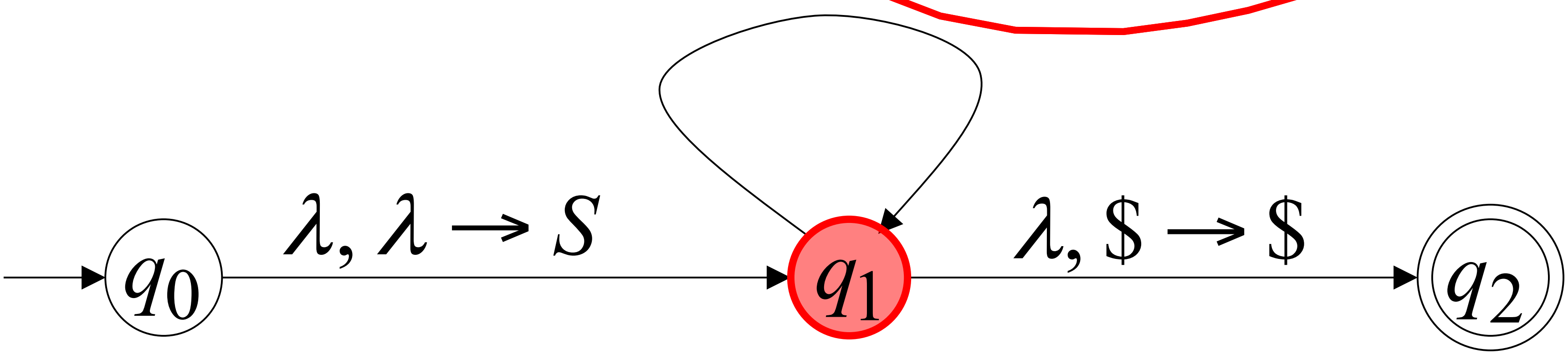
$\lambda, S \rightarrow b$

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

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

Stack

b
T
b
$\$$



Derivation: $S \Rightarrow aSTb \Rightarrow abTb$

Input

a	b	a	b

Stack

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

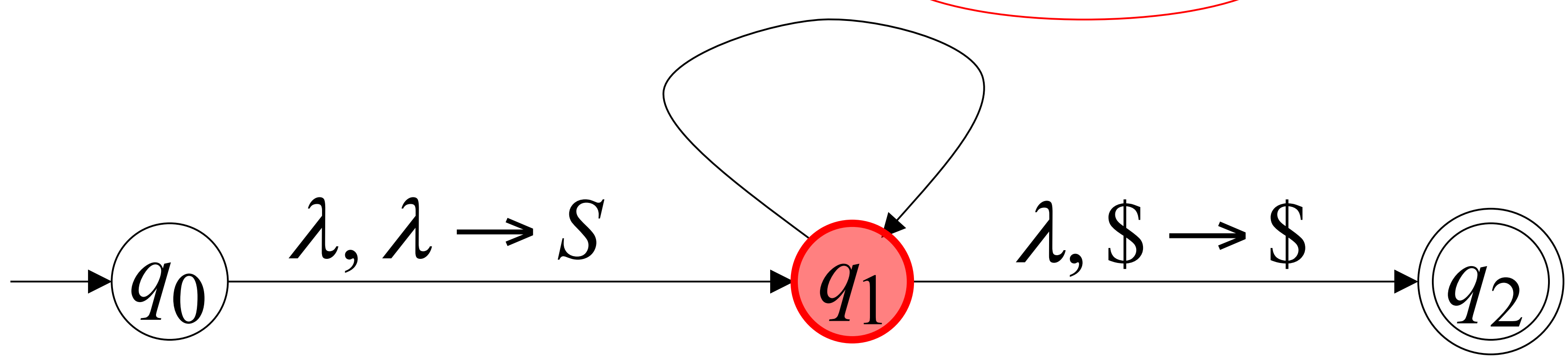
Time 5

$\lambda, S \rightarrow aSTb$

$\lambda, S \rightarrow b$

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

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



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

Input

a	b	a	b

Stack

T
a
b
$\$$

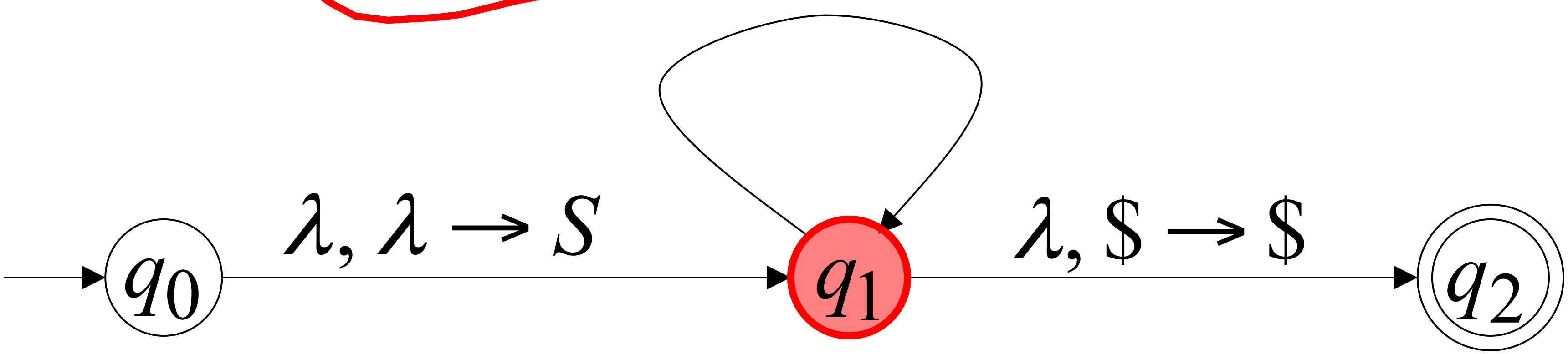
Time 6

$\lambda, S \rightarrow aSTb$

$\lambda, S \rightarrow b$

$\lambda, T \rightarrow T\underline{a}$ $\qquad a, a \rightarrow \lambda$

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



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

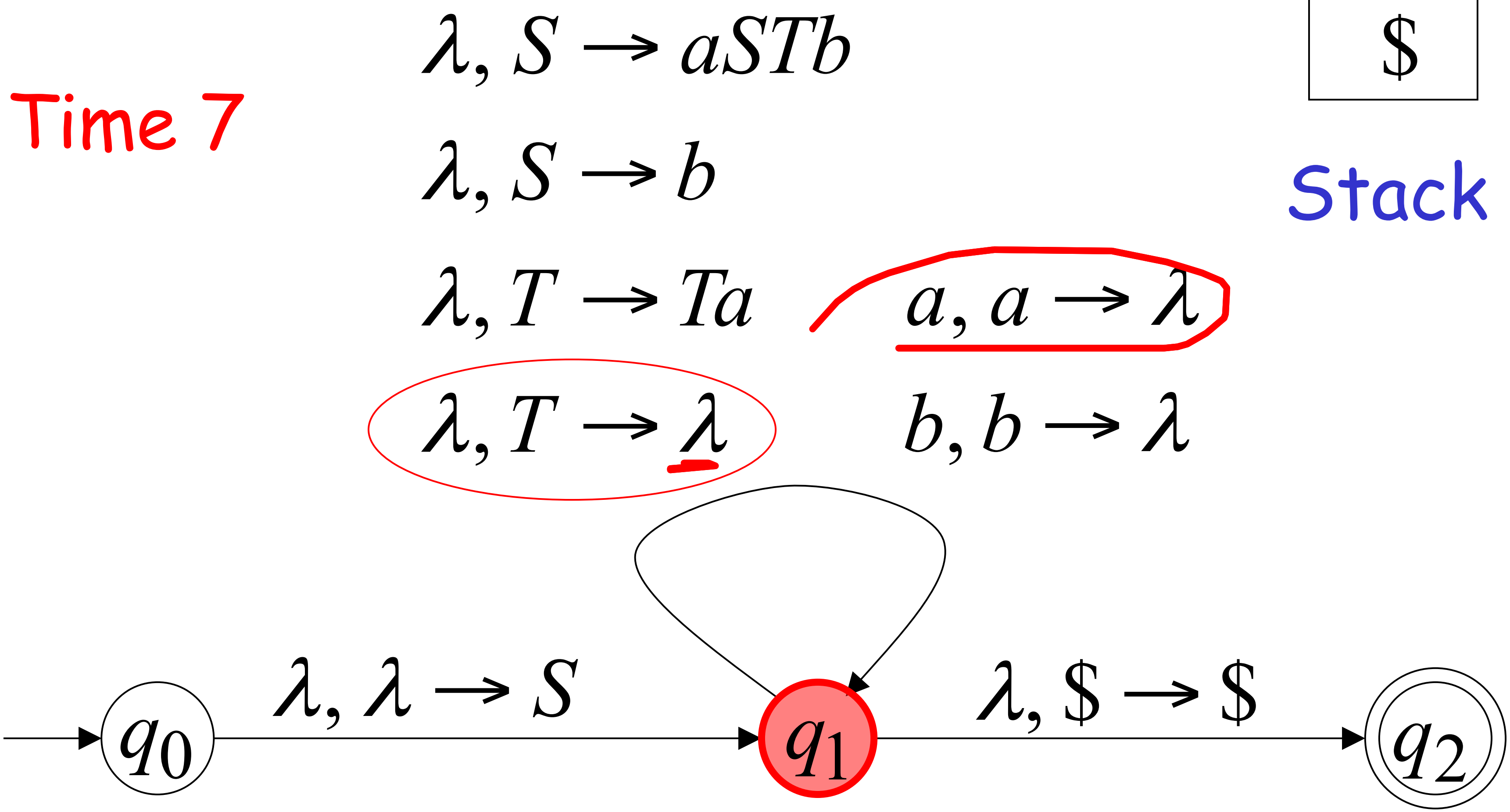
Input

a	b	a	b

Time 7

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

Stack



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

Input

a	b	a	b

Stack

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

Time 8

$\lambda, S \rightarrow aSTb$

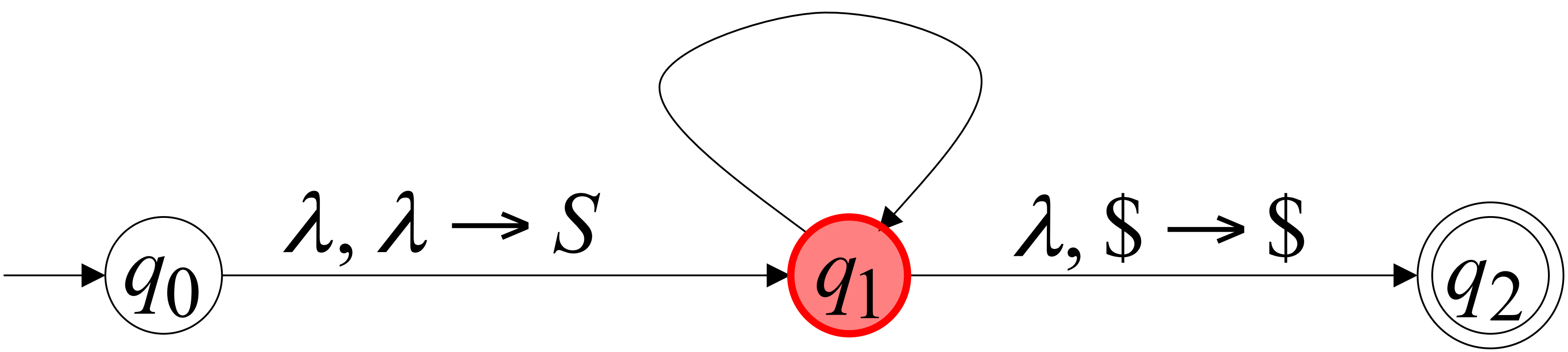
$\lambda, S \rightarrow b$

$\lambda, T \rightarrow Ta$

$\lambda, T \rightarrow \lambda$

$a, a \rightarrow \lambda$

$b, b \rightarrow \lambda$



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

Input

a	b	a	b

Time 9

$\lambda, S \rightarrow aSTb$

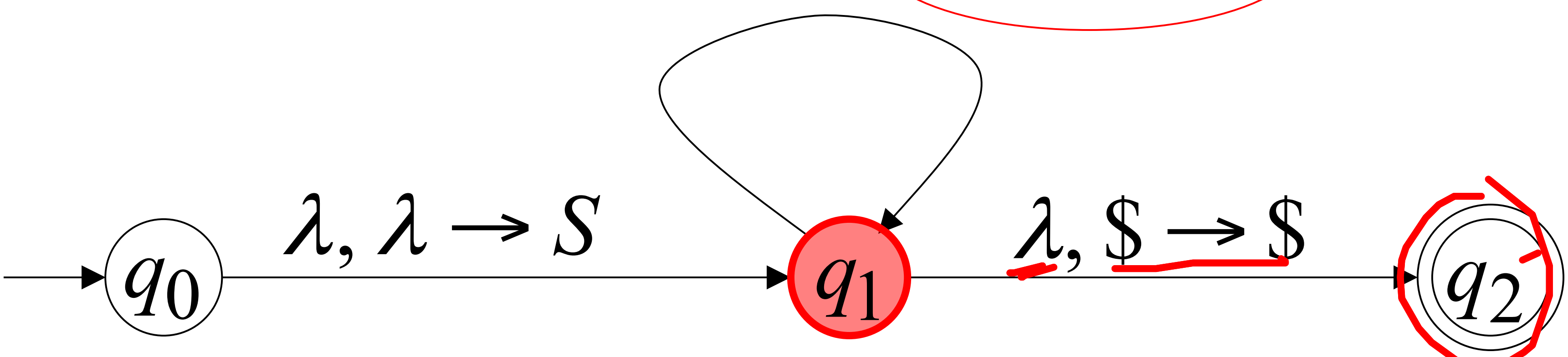
$\lambda, S \rightarrow b$

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

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

$\cancel{b}$
$\$$

Stack



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

Input

a	b	a	b

Time 10

$\lambda, S \rightarrow aSTb$

$\lambda, S \rightarrow b$

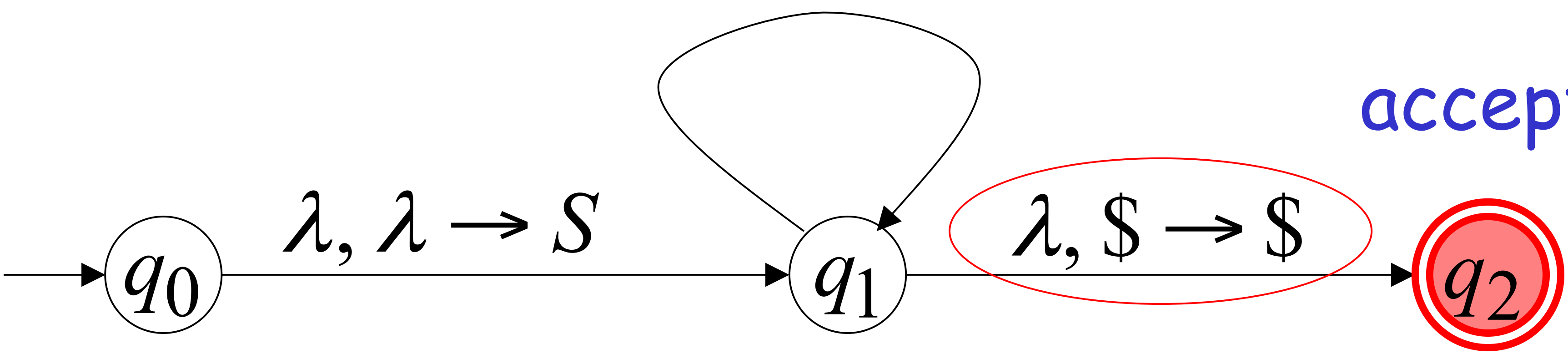
$\lambda, T \rightarrow Ta$ $\quad a, a \rightarrow \lambda$

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

$\$$

Stack

accept



Grammar

Leftmost Derivation

S

$\Rightarrow aSTb$

$\Rightarrow abTb$

$\Rightarrow abTab$

$\Rightarrow abab$

PDA Computation

$(q_0, abab, \$)$

$\succ (q_1, abab, S\$)$

$\succ (q_1, bab, STb\$)$

$\succ (q_1, bab, bTb\$)$

$\succ (q_1, ab, Tb\$)$

$\succ (q_1, ab, Tab\$)$

$\succ (q_1, ab, ab\$)$

$\succ (q_1, b, b\$)$

$\succ (q_1, \lambda, \$)$

$\succ (q_2, \lambda, \$)$

In general, it can be shown that:

Grammar G generates string w

$$S \overset{*}{\Rightarrow} w$$

If and Only if

PDA M accepts w

$$(q_0, w, \$) \succ (q_2, \lambda, \$)$$

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

Proof - step 2

Convert

PDAs
to
Context-Free Grammars

Main idea: Reverse engineer the productions from transitions

If $\delta(q,a,Z) \Rightarrow (p, Y_1Y_2Y_3\ldots Y_k)$:

- State is changed from q to p;
- Terminal a is consumed;
- Stack top symbol Z is popped and replaced
- with a sequence of k variables.

Action: Create a grammar variable called "[qZp]" which includes the following production:

- $[qZp] \Rightarrow a[pY_1q_1]\ [q_1Y_2q_2]\ [q_2Y_3q_3]\ldots\ [q_{k-1}Y_kq_k]$

Example: Bracket matching

P_N: ($\{q_0\}$, $\{b,e\}$, $\{Z_0,Z_1\}$, δ, q_0, Z_0)

1. $\delta(q_0,b,Z_0) = \{ (q_0,Z_1Z_0) \}$
2. $\delta(q_0,b,Z_1) = \{ (q_0,Z_1Z_1) \}$
3. $\delta(q_0,e,Z_1) = \{ (q_0, \varepsilon) \}$
4. $\delta(q_0, \varepsilon, Z_0) = \{ (q_0, \varepsilon) \}$

0. $S \Rightarrow [q_0Z_0q_0]$
1. $[q_0Z_0q_0] \Rightarrow b\ [q_0Z_1q_0]\ [q_0Z_0q_0]$
2. $[q_0Z_1q_0] \Rightarrow b\ [q_0Z_1q_0]\ [q_0Z_1q_0]$
3. $[q_0Z_1q_0] \Rightarrow e$
4. $[q_0Z_0q_0] \Rightarrow \varepsilon$

Let $A=[q_0Z_0q_0]$
Let $B=[q_0Z_1q_0]$

0. $S \Rightarrow A$
1. $A \Rightarrow b\ B\ A$
2. $B \Rightarrow b\ B\ B$
3. $B \Rightarrow e$
4. $A \Rightarrow \varepsilon$

Simplifying,

0. $S \Rightarrow b\ B\ S \mid \varepsilon$
1. $B \Rightarrow b\ B\ B \mid e$

If you were to directly write a CFG:

$S \Rightarrow b\ S\ e\ S \mid \varepsilon$

More -detailed process of converting PDA to CFG

First modify PDA M so that:

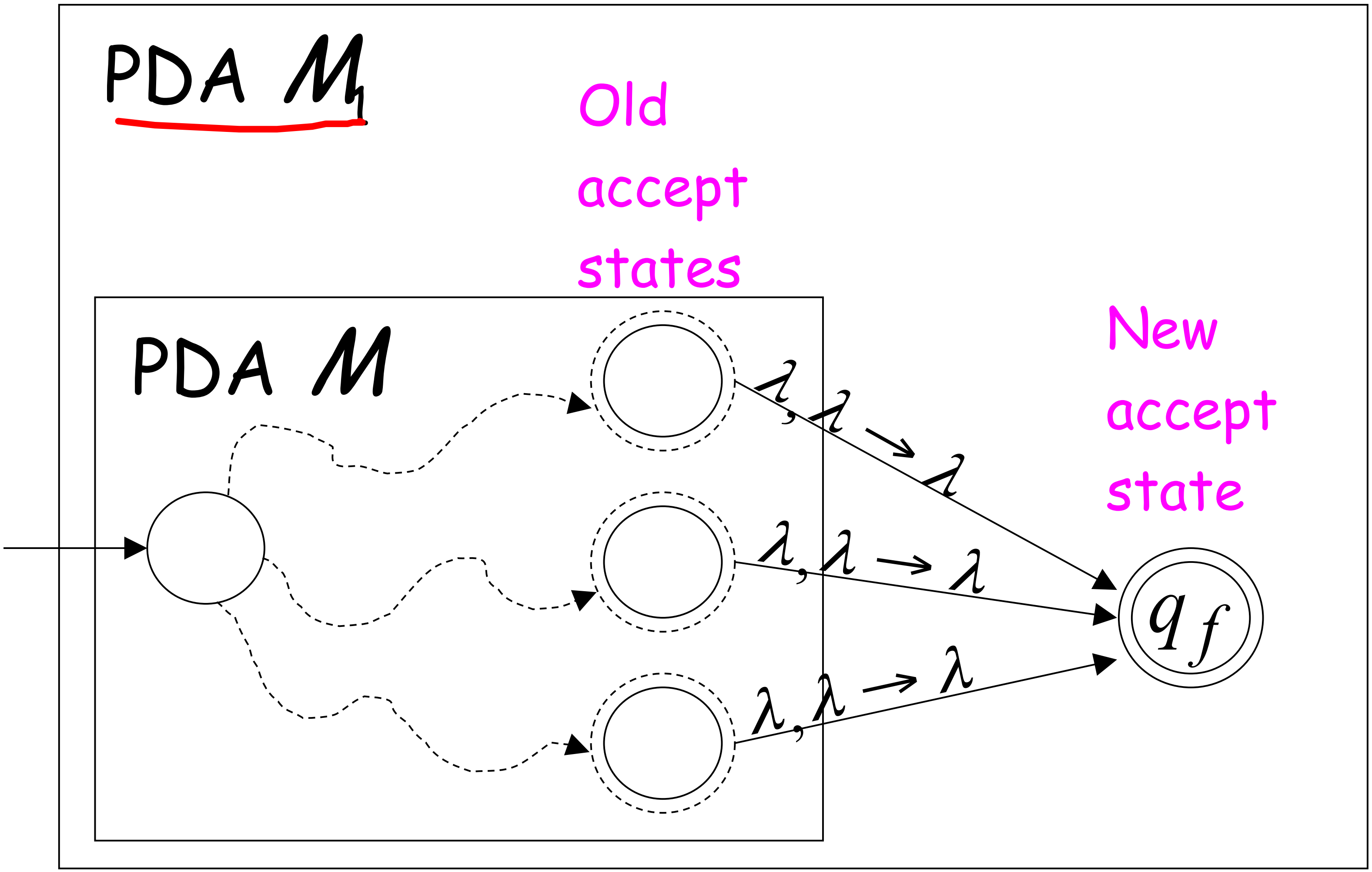
1. The PDA has a single accept state

2. Use new initial stack symbol #

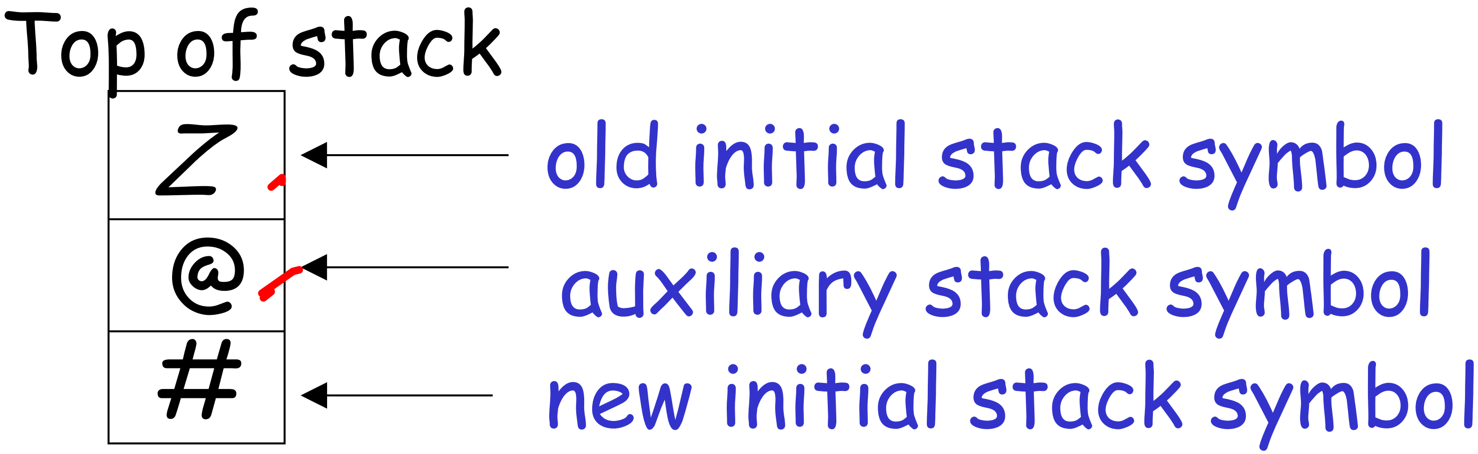
3. On acceptance the stack contains only stack symbol #

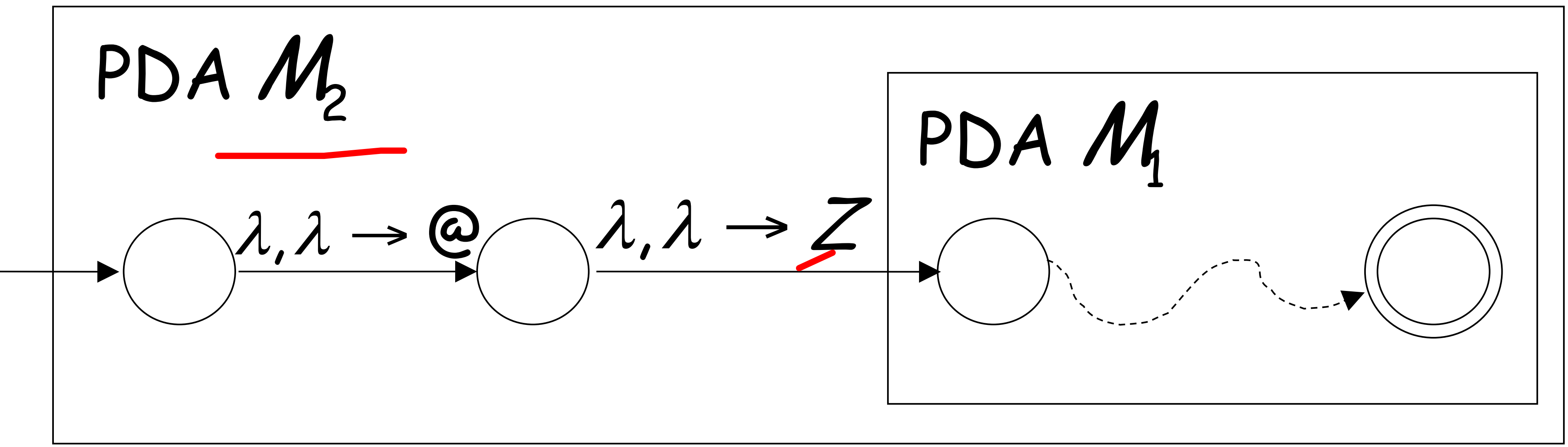
4. Each transition either pushes a symbol or pops a symbol but not both together

1. The PDA has a single accept state



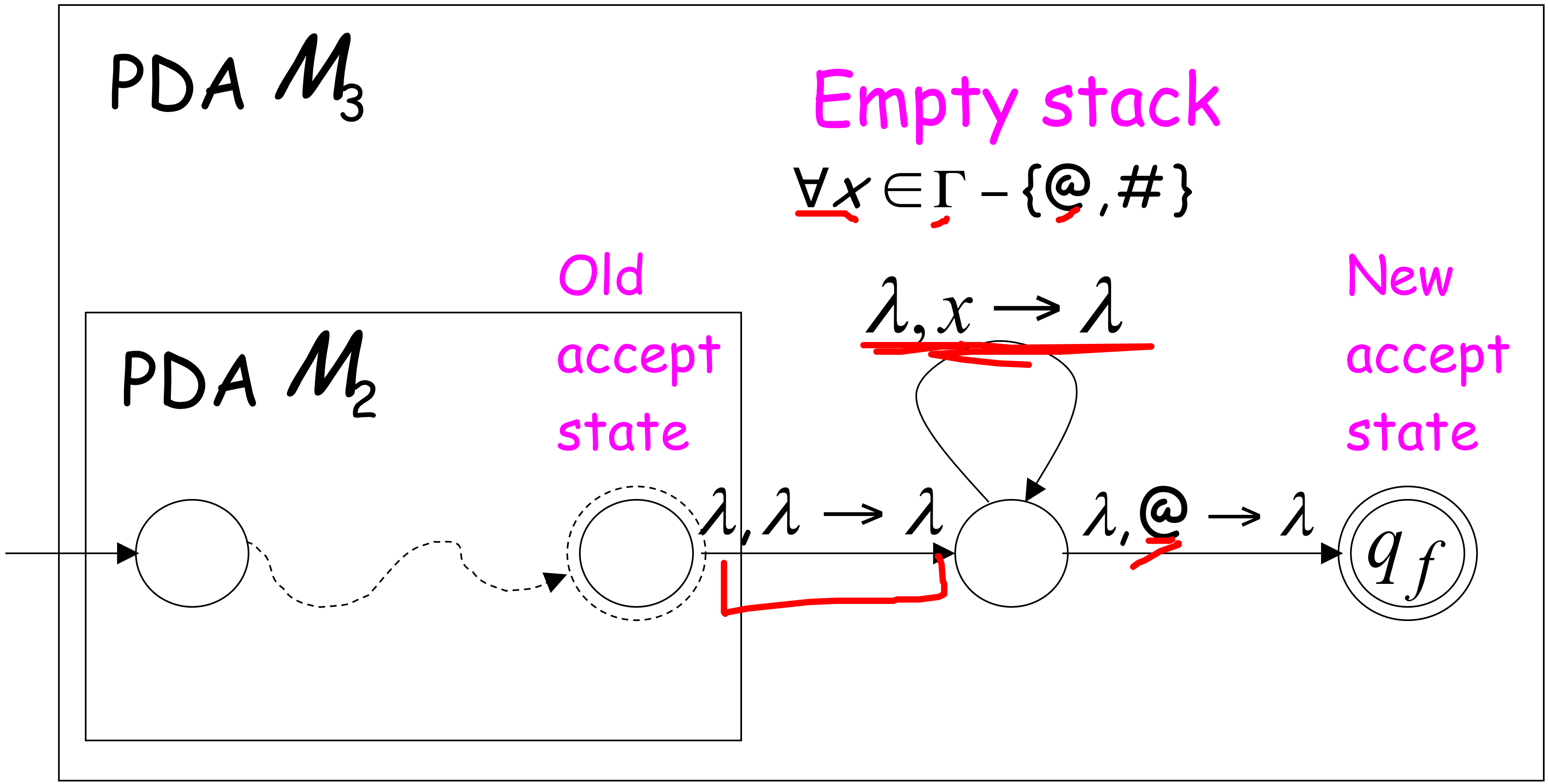
2. Use new initial stack symbol





M_1 still thinks that Z is the initial stack

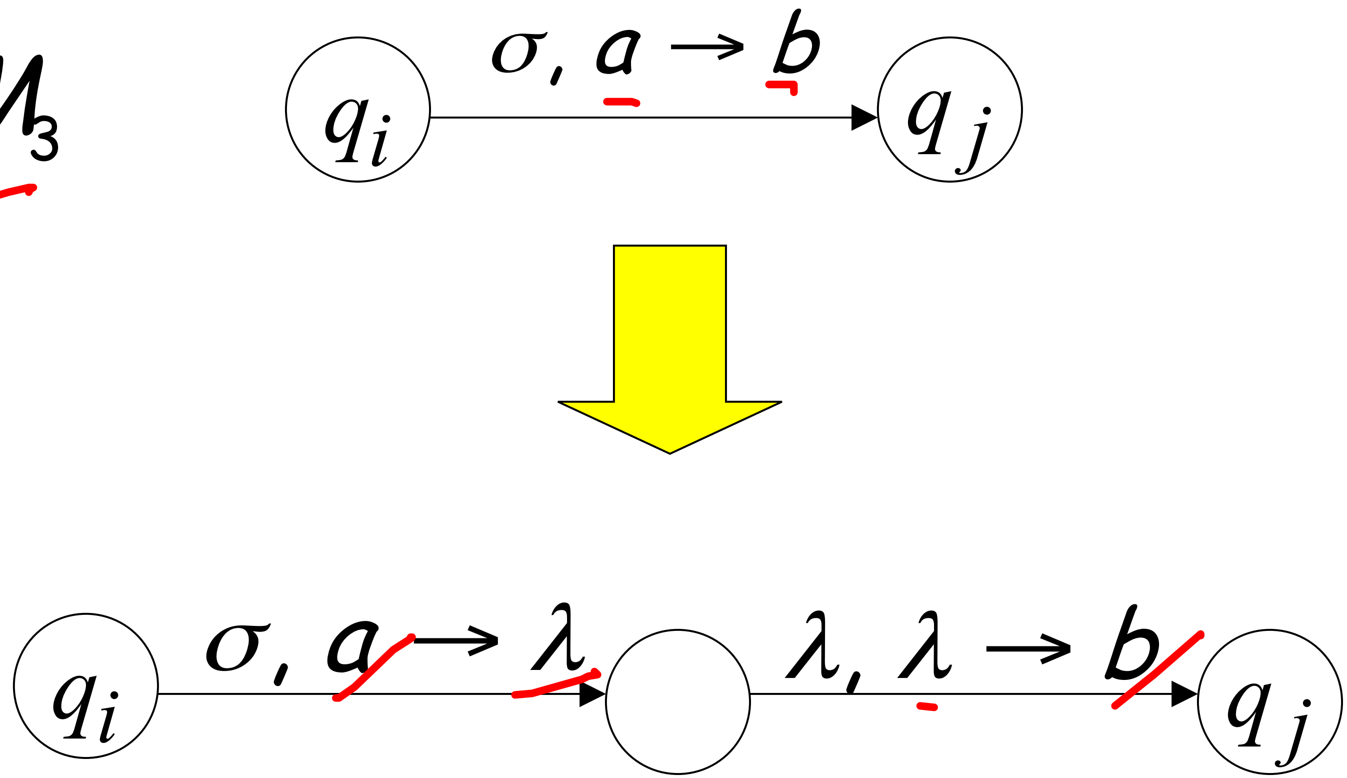
3. On acceptance the stack contains only stack symbol #



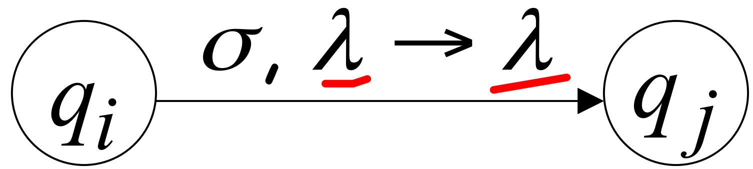
4. Each transition either pushes a symbol or pops a symbol but not both together

PDA M_3

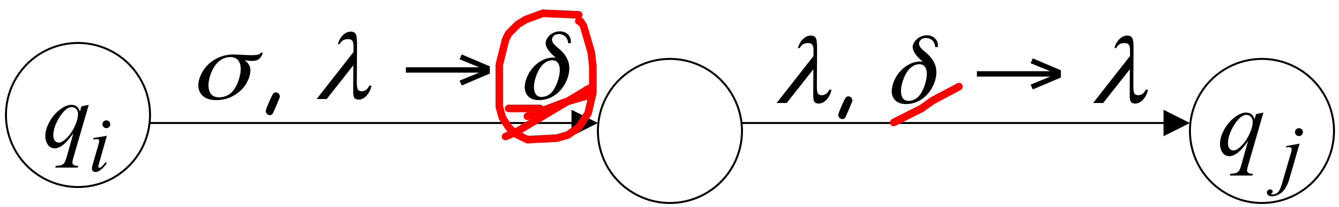
PDA M_4



PDA M_3



PDA M_4

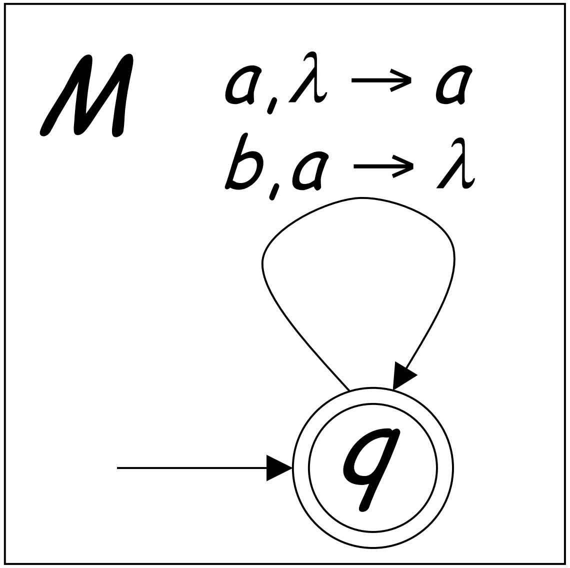


Where δ is a symbol of the stack alphabet

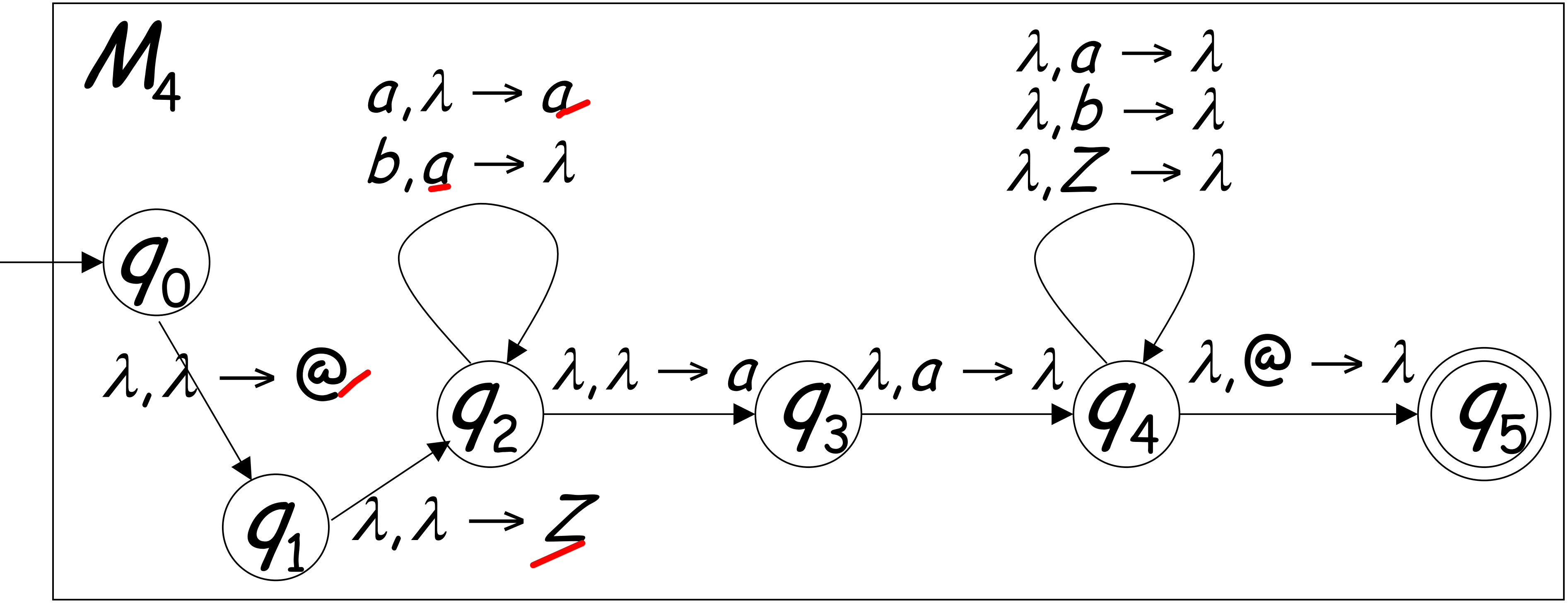
PDA M_4 is the final modified PDA

Note that the new initial stack symbol #
is never used in any transition

Example:



M
$a,\lambda \rightarrow a$
$b,a \rightarrow \lambda$
q



M_4
q_0
$\lambda,\lambda \rightarrow @$
q_1
$\lambda,\lambda \rightarrow Z$
$a,\lambda \rightarrow a$
$b,a \rightarrow \lambda$
q_2
$\lambda,\lambda \rightarrow a$
q_3
$\lambda,a \rightarrow \lambda$
$\lambda,a \rightarrow \lambda$
$\lambda,b \rightarrow \lambda$
$\lambda,Z \rightarrow \lambda$
q_4
$\lambda,@ \rightarrow \lambda$
q_5

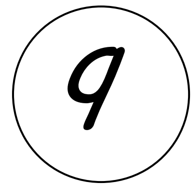
Grammar Construction

Variables: A_{q_i, q_j}

States of PDA

PDA

Kind 1: for each state

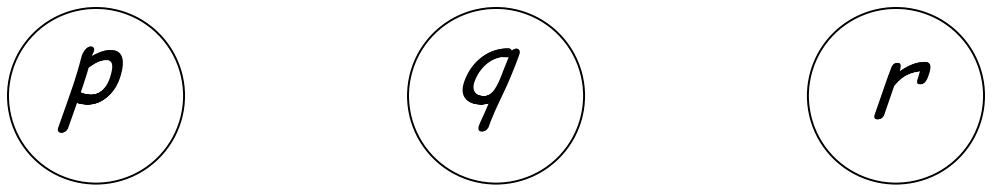


Grammar

$$\boxed{A_{qq}} \rightarrow \lambda$$

PDA

Kind 2: for every three states



Grammar

$$A_{pq} \rightarrow \boxed{A_{pr}A_{rq}}$$

PDA

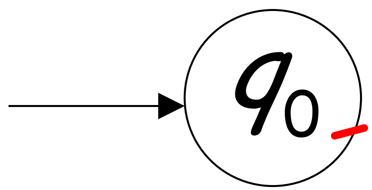
Kind 3: for every pair of such transitions

Grammar

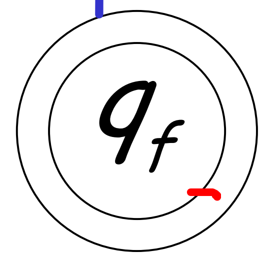
$$A_{pq} \rightarrow aA_{rs}b$$

PDA

Initial state



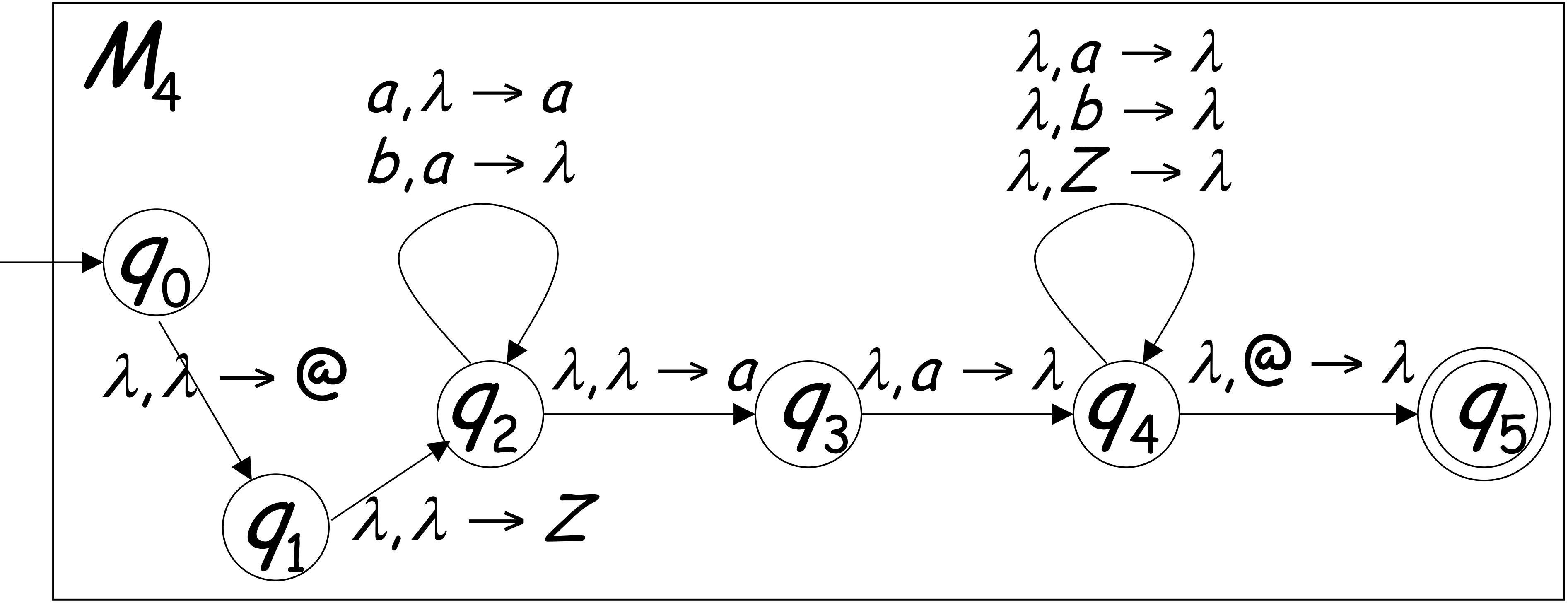
Accept state



Grammar

Start variable $A_{q_0 q_f}$

Example: PDA



M_4
$a, \lambda \rightarrow a$
$b, a \rightarrow \lambda$
$\lambda, a \rightarrow \lambda$
$\lambda, b \rightarrow \lambda$
$\lambda, Z \rightarrow \lambda$
q_0
$\lambda, \lambda \rightarrow @$
q_1
$\lambda, \lambda \rightarrow Z$
q_2
$\lambda, \lambda \rightarrow a$
q_3
$\lambda, a \rightarrow \lambda$
q_4
$\lambda, @ \rightarrow \lambda$
q_5

Grammar

Kind 1: from single states

$$A_{q_0 q_0} \rightarrow \lambda$$

$$A_{q_1 q_1} \rightarrow \lambda$$

$$A_{q_2 q_2} \rightarrow \lambda$$

$$A_{q_3 q_3} \rightarrow \lambda$$

$$A_{q_4 q_4} \rightarrow \lambda$$

$$A_{q_5 q_5} \rightarrow \lambda$$

Kind 2: from triplets of states

$$A_{q_0q_0} \rightarrow A_{q_0q_0}A_{q_0q_0} \mid A_{q_0q_1}A_{q_1q_0} \mid A_{q_0q_2}A_{q_2q_0} \mid A_{q_0q_3}A_{q_3q_0} \mid A_{q_0q_4}A_{q_4q_0} \mid A_{q_0q_5}A_{q_5q_0}$$

$$A_{q_0q_1} \rightarrow A_{q_0q_0}A_{q_0q_1} \mid A_{q_0q_1}A_{q_1q_1} \mid A_{q_0q_2}A_{q_2q_1} \mid A_{q_0q_3}A_{q_3q_1} \mid A_{q_0q_4}A_{q_4q_1} \mid A_{q_0q_5}A_{q_5q_1}$$

$\vdots$

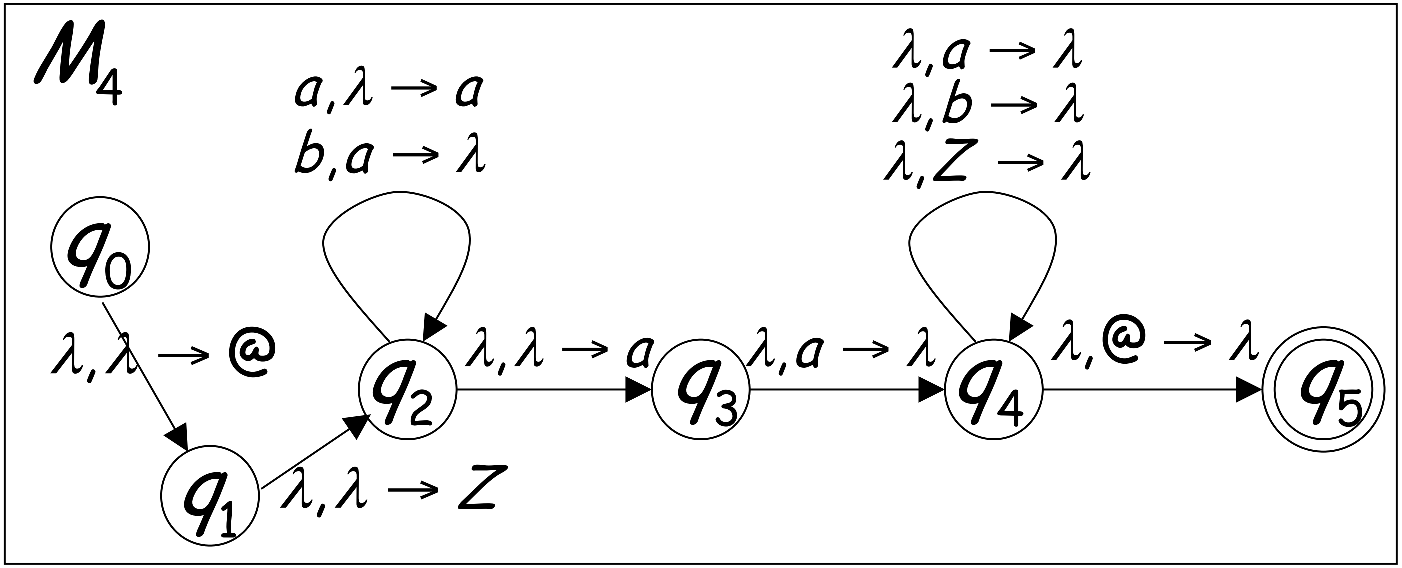
$$A_{q_0q_5} \rightarrow A_{q_0q_0}A_{q_0q_5} \mid A_{q_0q_1}A_{q_1q_5} \mid A_{q_0q_2}A_{q_2q_5} \mid A_{q_0q_3}A_{q_3q_5} \mid A_{q_0q_4}A_{q_4q_5} \mid A_{q_0q_5}A_{q_5q_5}$$

$\vdots$

$$A_{q_5q_5} \rightarrow A_{q_5q_0}A_{q_0q_5} \mid A_{q_5q_1}A_{q_1q_5} \mid A_{q_5q_2}A_{q_2q_5} \mid A_{q_5q_3}A_{q_3q_5} \mid A_{q_5q_4}A_{q_4q_5} \mid A_{q_5q_5}A_{q_5q_5}$$

Start variable $A_{q_0q_5}$

Kind 3: from pairs of transitions



$A_{q_0q_5} \rightarrow A_{q_1q_4}$ $\quad$ $A_{q_2q_4} \rightarrow aA_{q_2q_4}$ $\quad$ $A_{q_2q_2} \rightarrow A_{q_3q_2}b$

$A_{q_1q_4} \rightarrow A_{q_2q_4}$ $\quad$ $A_{q_2q_2} \rightarrow aA_{q_2q_2}b$ $\quad$ $A_{q_2q_4} \rightarrow A_{q_3q_3}$

$A_{q_2q_4} \rightarrow aA_{q_2q_3}$ $\quad$ $A_{q_2q_4} \rightarrow A_{q_3q_4}$

So far we have shown:

$$L(G) \subseteq L(M)$$

$$L(G) \supseteq L(M)$$

Therefore: $L(G) = L(M)$