PDAs Accept Context-Free Languages

Theorem:

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

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

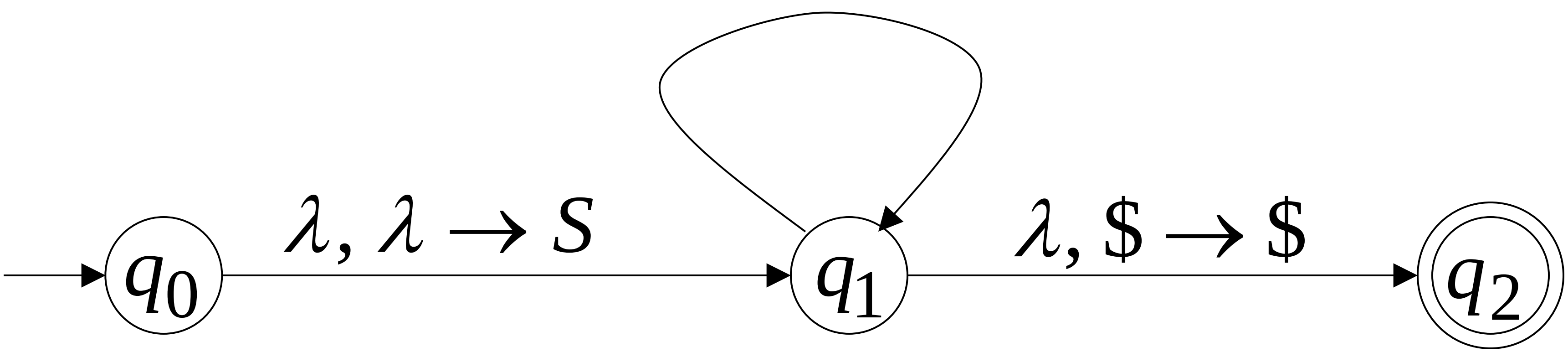
$$A \rightarrow w$$

For each terminal in G

$$a$$

Add transitions

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



Example

Grammar

$S \to aSTb$

$S \to b$

$T \to Ta$

$T \to \lambda$

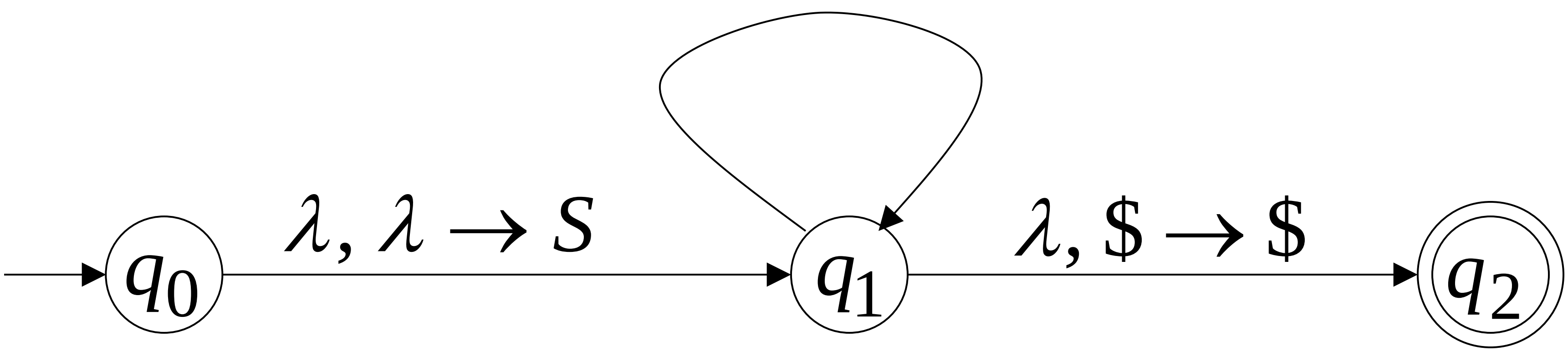
PDA

$\lambda, S \to aSTb$

$\lambda, S \to b$

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

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



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

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

Variables or terminals

Variable

Grammar
Leftmost Derivation

PDA Computation

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

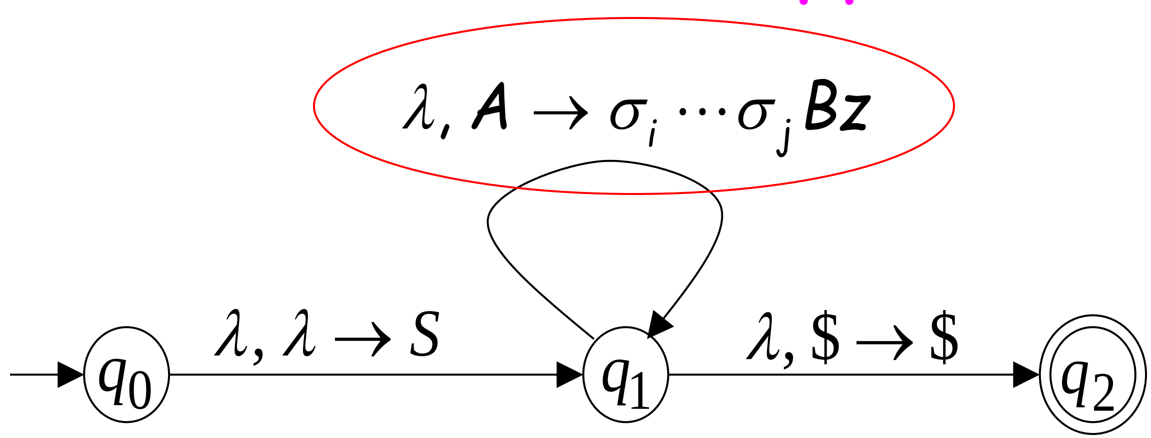
$$\Rightarrow xAy \qquad\qquad \succ (q_1, \sigma_i \cdots \sigma_n, Ay\$)$$

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

Production applied

$$A \rightarrow \sigma_i \cdots \sigma_j Bz$$

Transition applied



Grammar
Leftmost Derivation

PDA Computation

$$\Rightarrow \cdots$$

$$\Rightarrow xAy$$

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

$$\succ \cdots$$

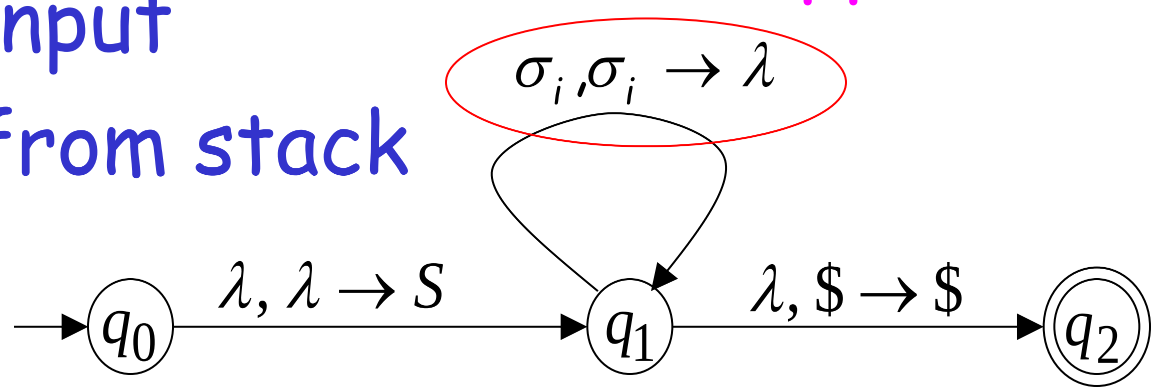
$$\succ (q_1, \sigma_i \cdots \sigma_n, Ay\$)$$

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

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

Transition applied

Read σ_i from input
and remove it from stack



Grammar
Leftmost Derivation

$\Rightarrow \cdots$

$\Rightarrow xAy$

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

PDA Computation

$\succ \cdots$

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

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

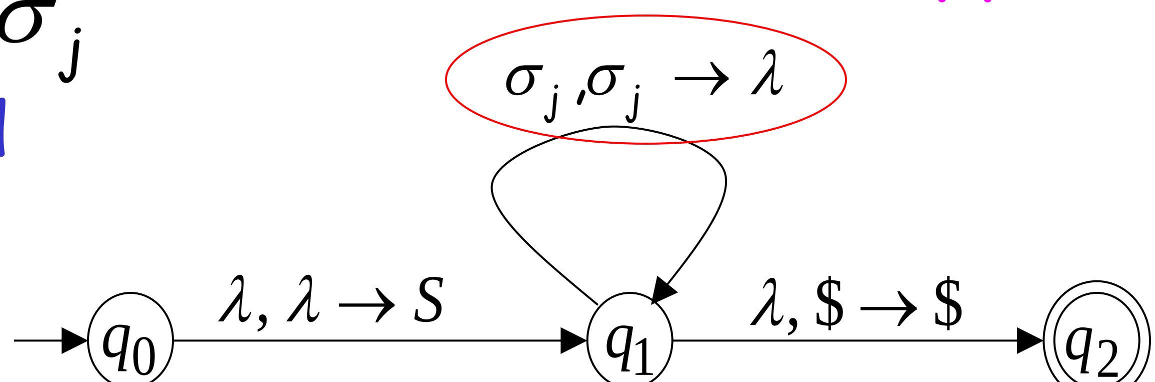
$\succ (q_1, \sigma_{i+1} \cdots \sigma_n, \sigma_{i+1} \cdots \sigma_j Bzy\$)$

$\succ \cdots$

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

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

$$\lambda, S \to aSTb$$

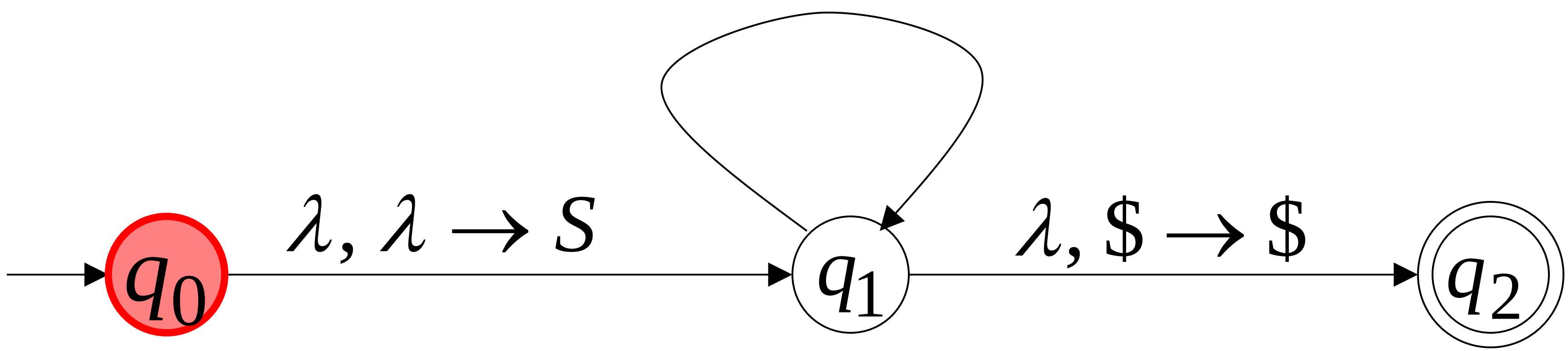
$$\lambda, S \to b$$

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

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

$\$$

Stack



Derivation: S

Input

a	b	a	b

Time 1

$\lambda, S \rightarrow aSTb$

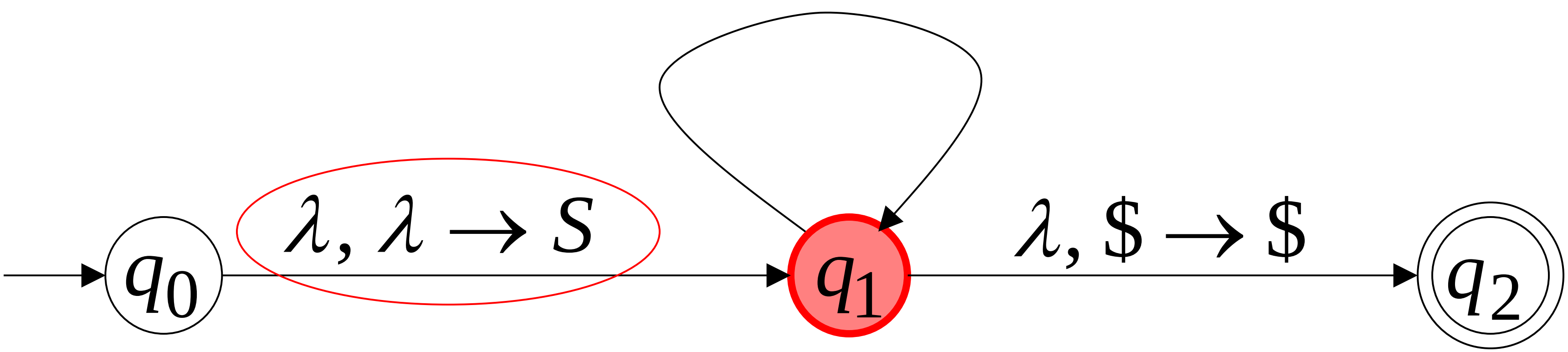
$\lambda, S \rightarrow b$

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

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

S
$\$$

Stack



Derivation: $S \Rightarrow aSTb$

Input

a	b	a	b

Time 2

$\lambda, S \rightarrow aSTb$

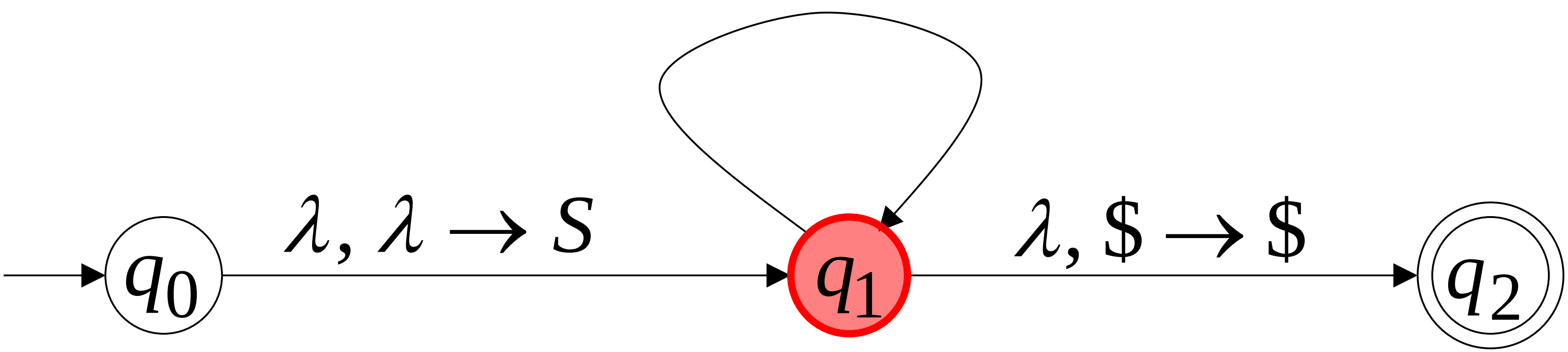
$\lambda, S \rightarrow b$

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

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

a
S
T
b
$\$$

Stack



Derivation: $S \Rightarrow aSTb$

Input

a	b	a	b

Time 3

$\lambda, S \rightarrow aSTb$

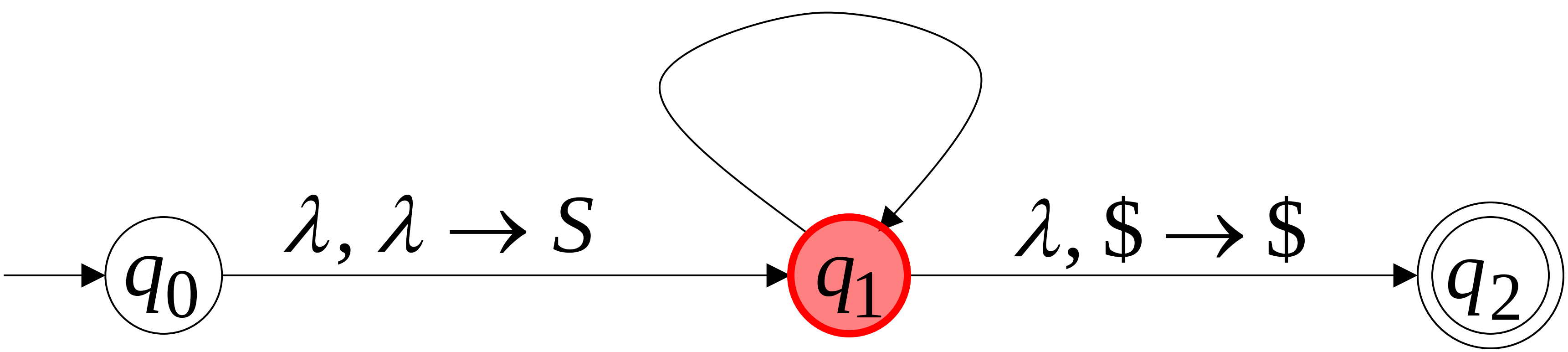
$\lambda, S \rightarrow b$

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

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

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

Stack



Derivation: $S \Rightarrow aSTb \Rightarrow abTb$

Input

a	b	a	b

Time 4

$\lambda, S \rightarrow aSTb$

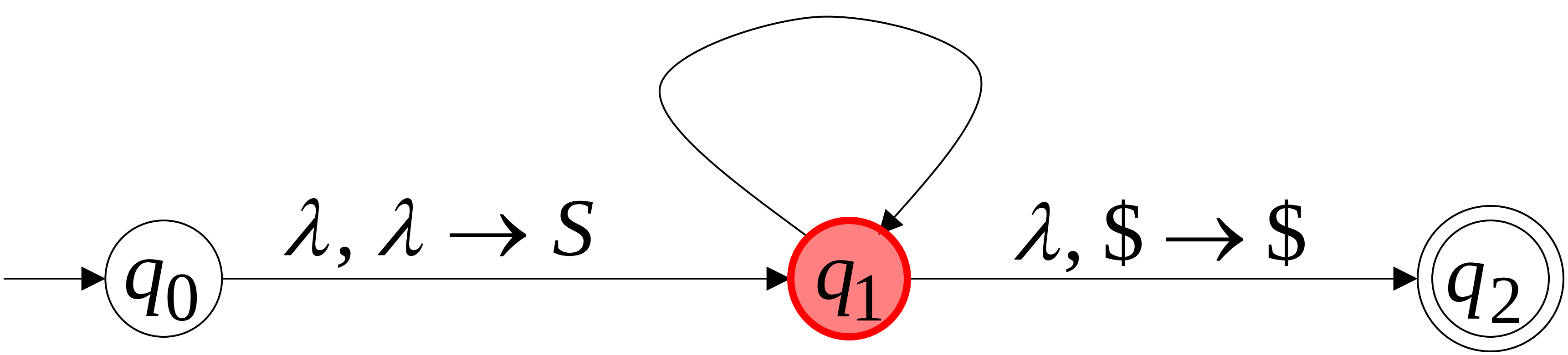
$\lambda, S \rightarrow b$

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

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

b
T
b
$\$$

Stack



Derivation: $S \Rightarrow aSTb \Rightarrow abTb$

Input

a	b	a	b

Time 5

$$\lambda, S \to aSTb$$

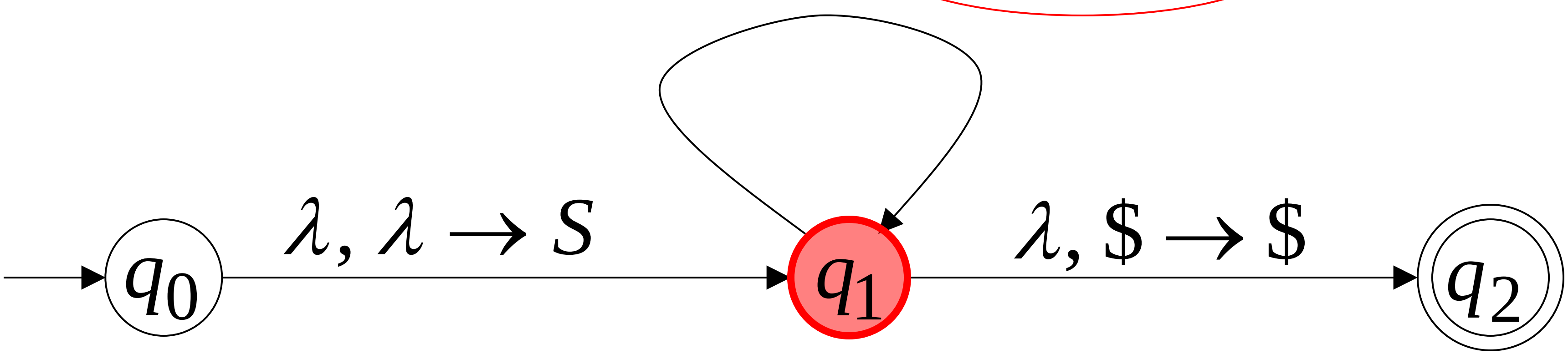
$$\lambda, S \to b$$

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

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

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

Stack



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

Input

a	b	a	b

Time 6

T
a
b
$\$$

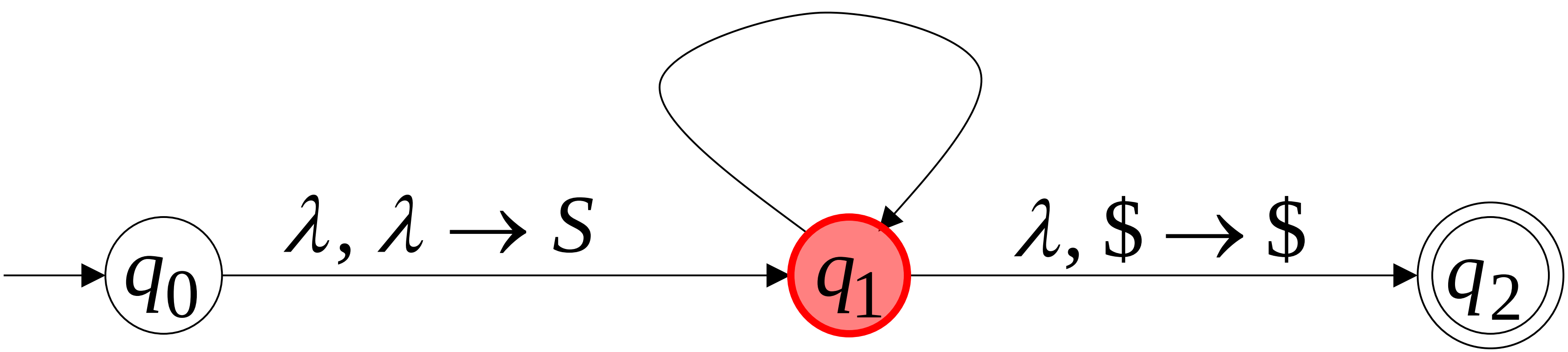
Stack

$\lambda, S \rightarrow aSTb$

$\lambda, S \rightarrow b$

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

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



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

Input

a	b	a	b

Time 7

Stack

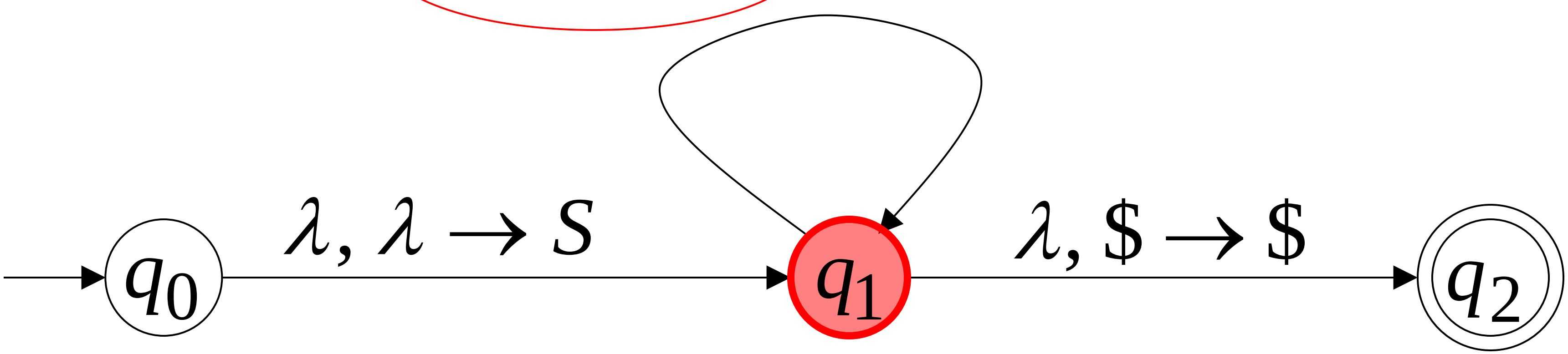
$\cancel{T}$
a
b
$\$$

$\lambda, S \rightarrow aSTb$

$\lambda, S \rightarrow b$

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

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



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

Input

a	b	a	b

Time 8

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

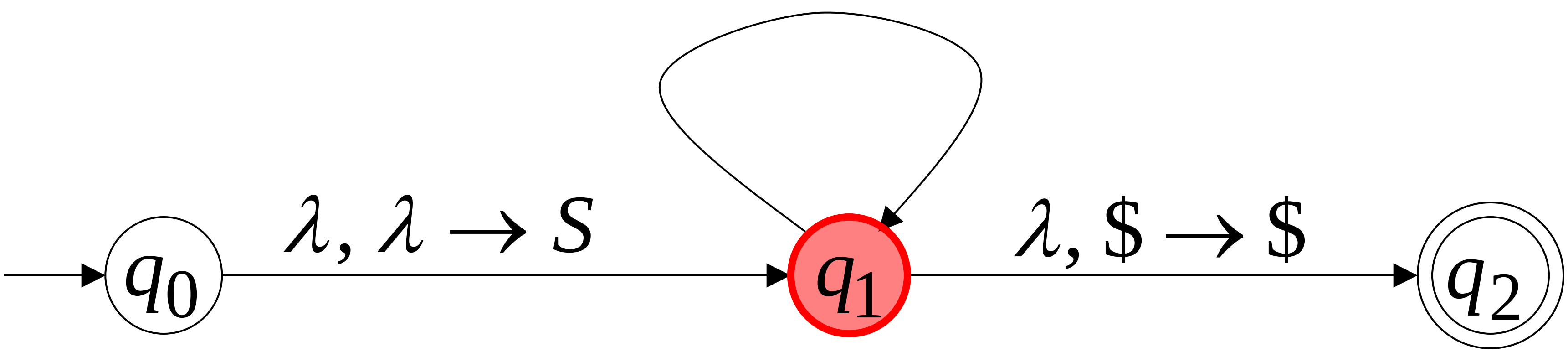
Stack

$\lambda, S \rightarrow aSTb$

$\lambda, S \rightarrow b$

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

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



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

Input

a	b	a	b

Time 9

$\lambda, S \rightarrow aSTb$

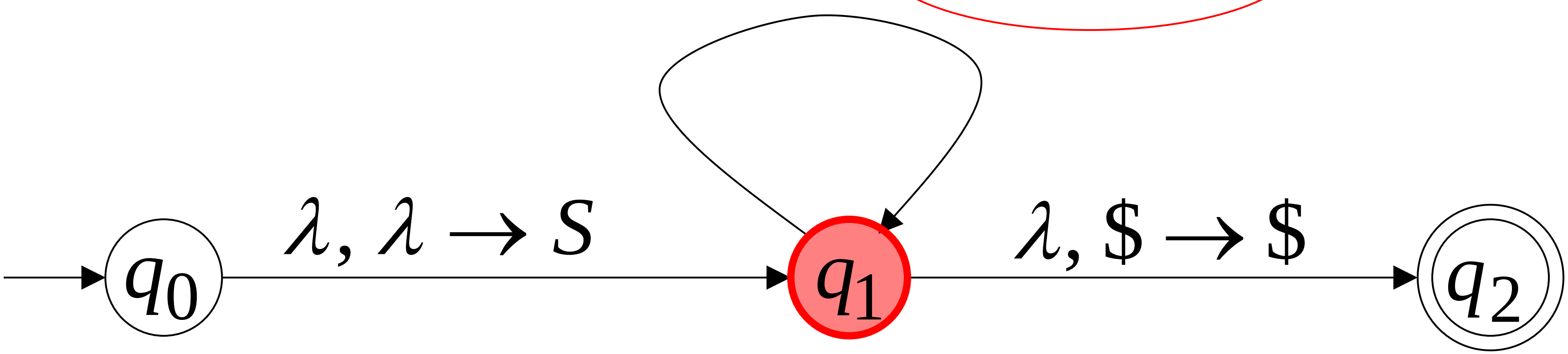
$\lambda, S \rightarrow b$

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

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

$\cancel{b}$
$\$$

Stack



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

Input

a	b	a	b

Time 10

$\lambda, S \rightarrow aSTb$

$\lambda, S \rightarrow b$

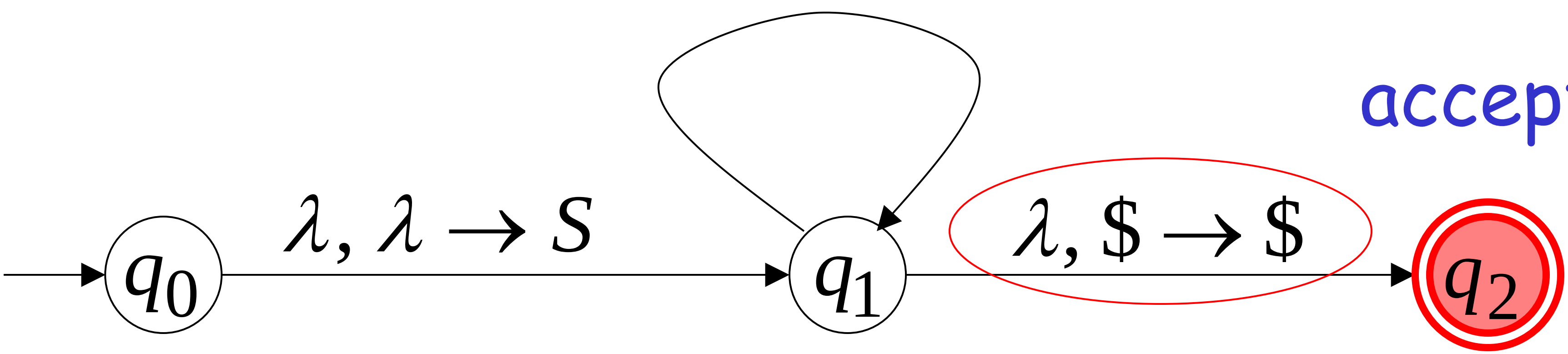
$\lambda, T \rightarrow Ta$ $\quad a, a \rightarrow \lambda$

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

$\$$

Stack

accept



Grammar
Leftmost Derivation

PDA Computation

$$S$$

$$(q_0, abab, \$)$$
$$\succ (q_1, abab, S\$)$$

$$\Rightarrow aSTb$$

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

$$\Rightarrow abTb$$

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

$$\Rightarrow abTab$$

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

$$\Rightarrow abab$$

$$\succ (q_1, ab, ab\$)$$
$$\succ (q_1, b, b\$)$$
$$\succ (q_1, \lambda, \$)$$
$$\succ (q_2, \lambda, \$)$$

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

Take an arbitrary PDA M

We will convert M
to a context-free grammar G such that:

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

First modify PDA M so that:

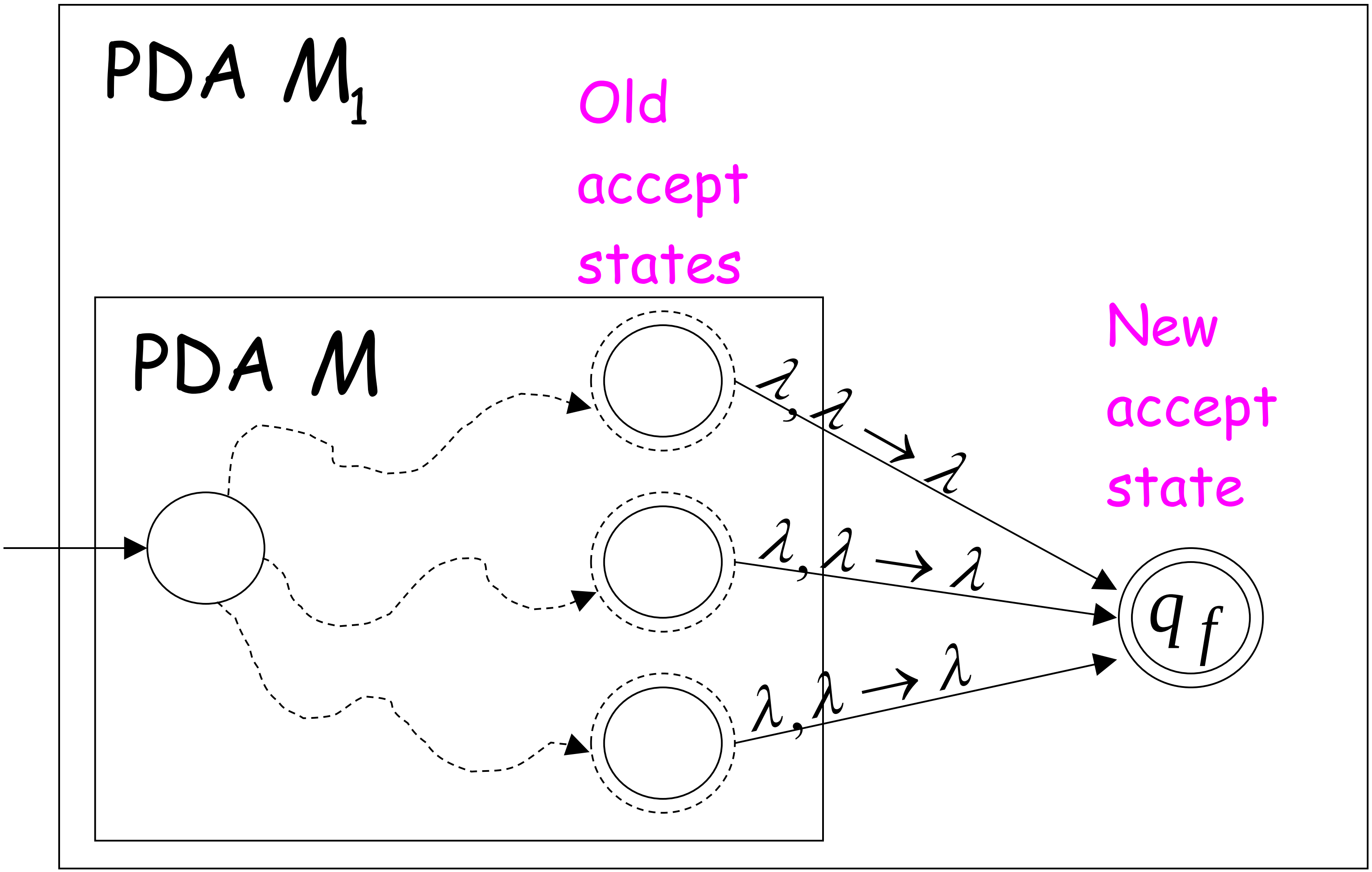
1. The PDA has a single accept state

2. Use new initial stack symbol #

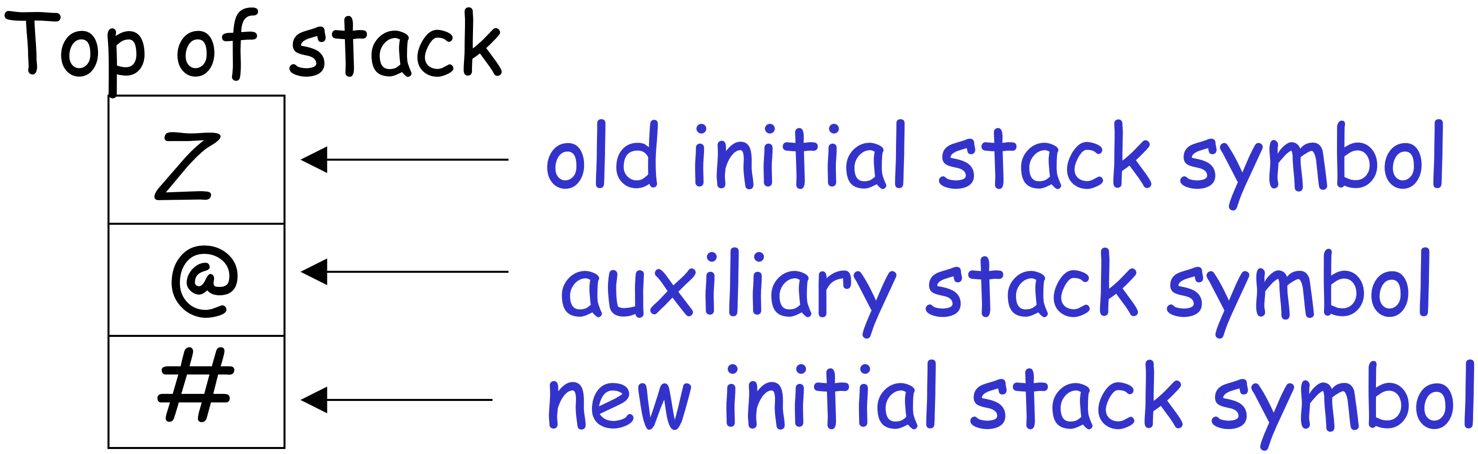
3. On acceptance the stack contains only stack symbol #

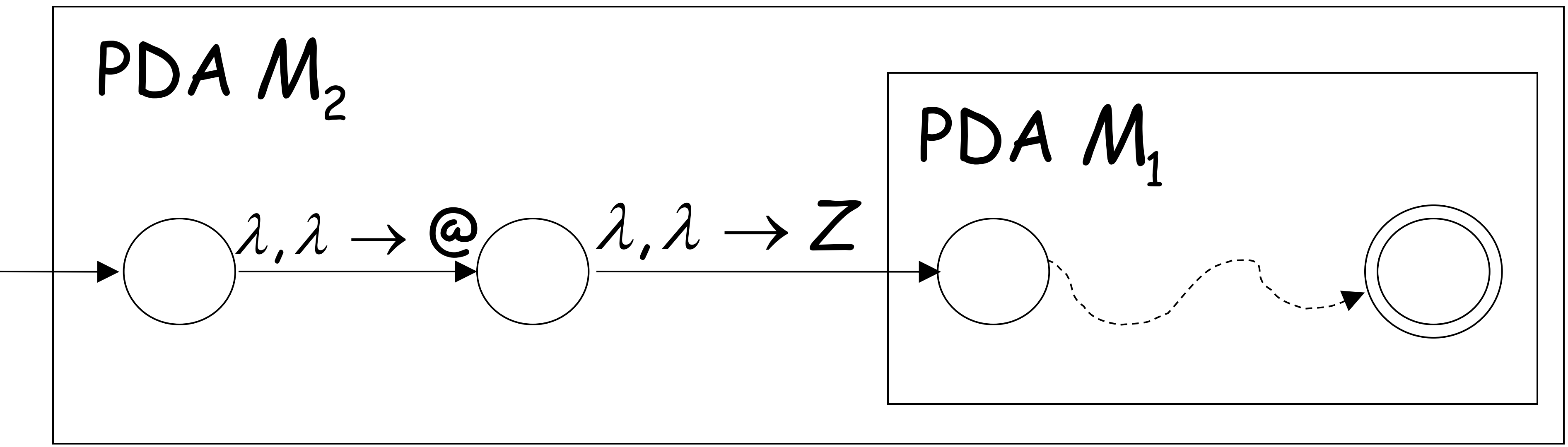
4. Each transition either pushes a symbol or pops a symbol but not both together

1. The PDA has a single accept state



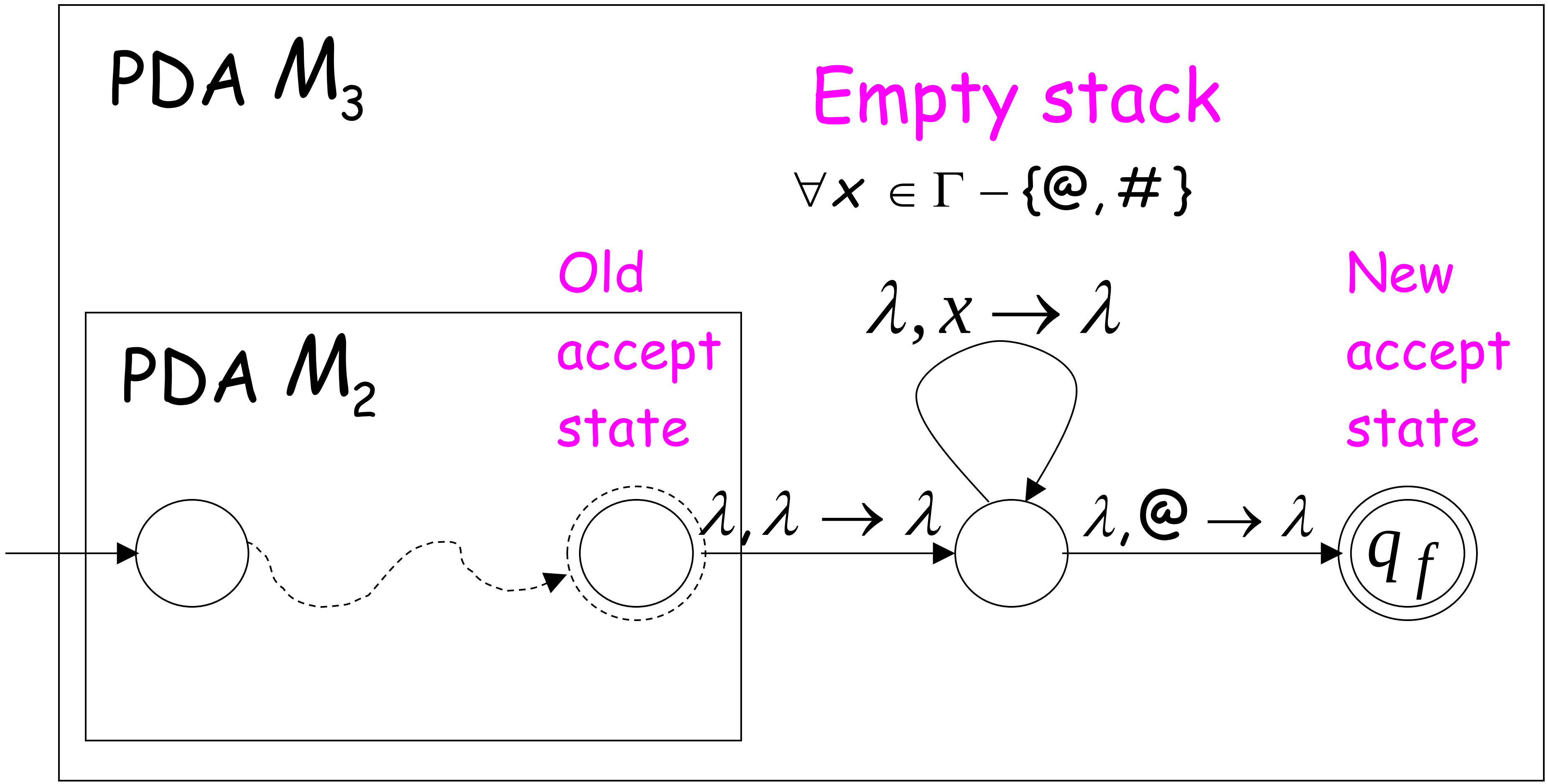
2. Use new initial stack symbol





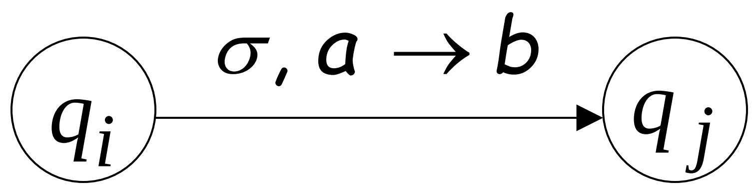
M_1 still thinks that Z is the initial stack

3. On acceptance the stack contains only stack symbol

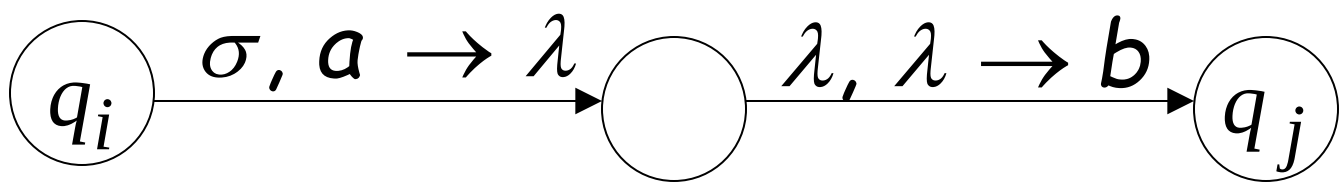


4. Each transition either pushes a symbol or pops a symbol but not both together

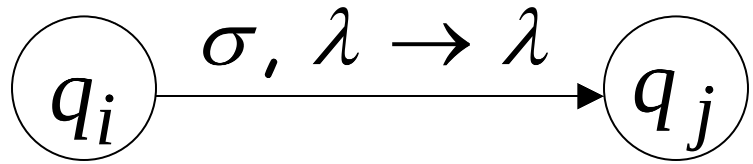
PDA M_3



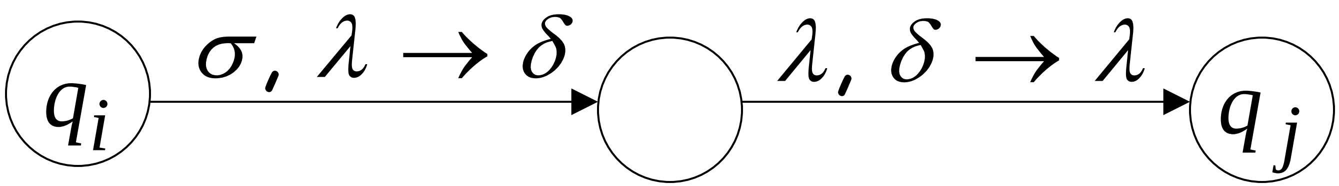
PDA M_4



PDA M_3



PDA M_4

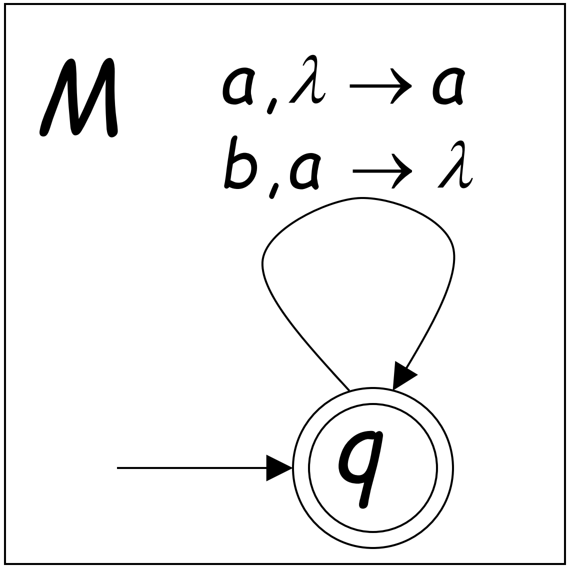


Where δ is a symbol of the stack alphabet

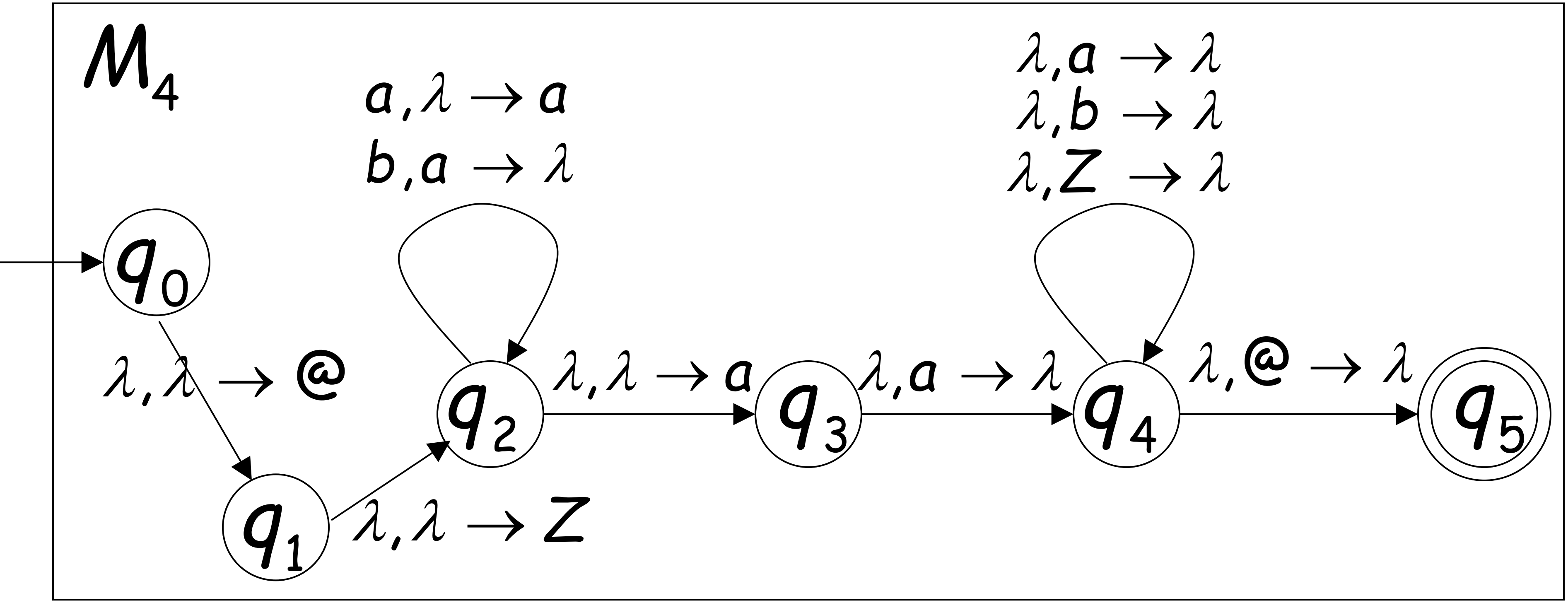
PDA M_4 is the final modified PDA

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

Example:



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



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

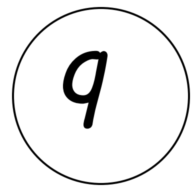
Grammar Construction

Variables: A_{q_i, q_j}

States of PDA

PDA

Kind 1: for each state

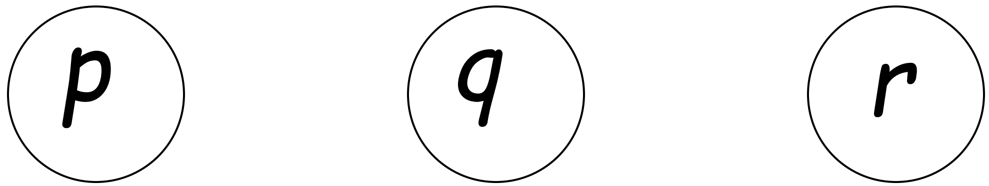


Grammar

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

PDA

Kind 2: for every three states



Grammar

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

PDA

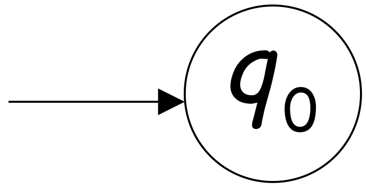
Kind 3: for every pair of such transitions

Grammar

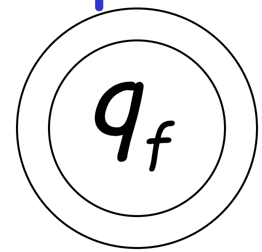
$$A_{pq} \rightarrow aA_{rs}b$$

PDA

Initial state



Accept state

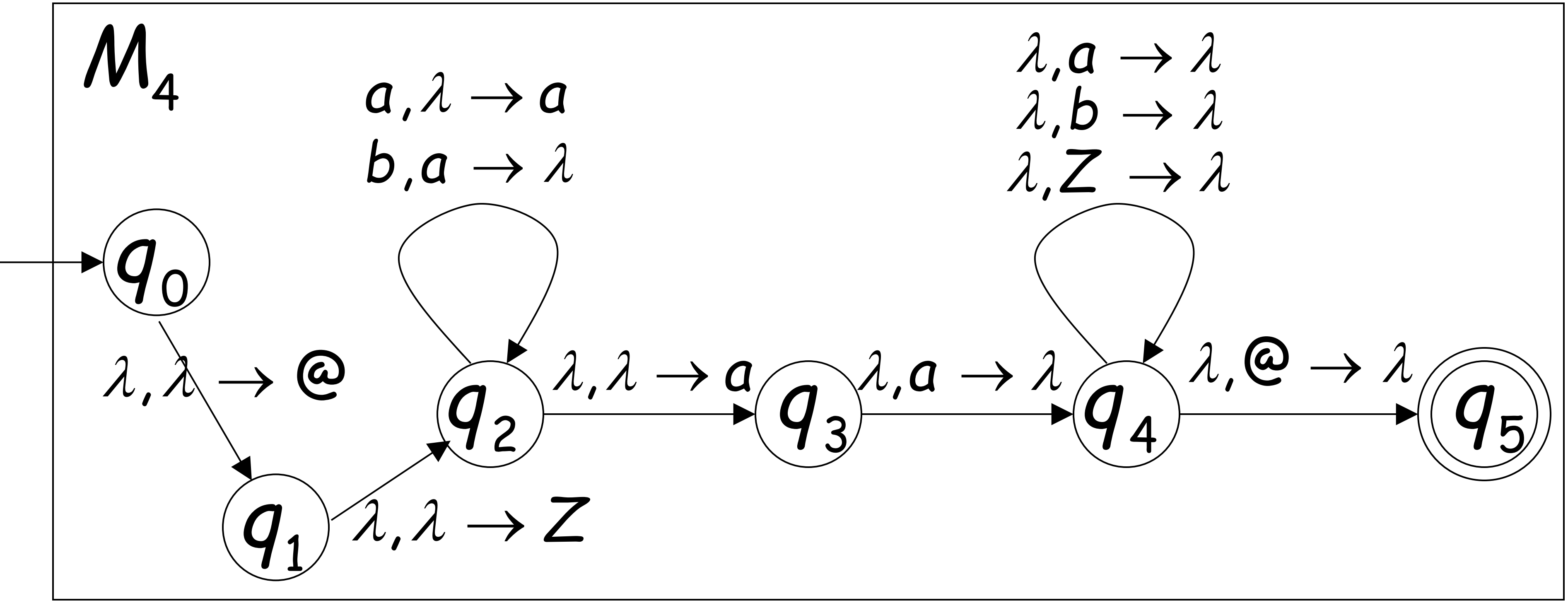


Grammar

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

$$A_{q_0q_0} \rightarrow \lambda$$

$$A_{q_1q_1} \rightarrow \lambda$$

$$A_{q_2q_2} \rightarrow \lambda$$

$$A_{q_3q_3} \rightarrow \lambda$$

$$A_{q_4q_4} \rightarrow \lambda$$

$$A_{q_5q_5} \rightarrow \lambda$$

Kind 2: from triplets of states

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

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

$$\vdots$$

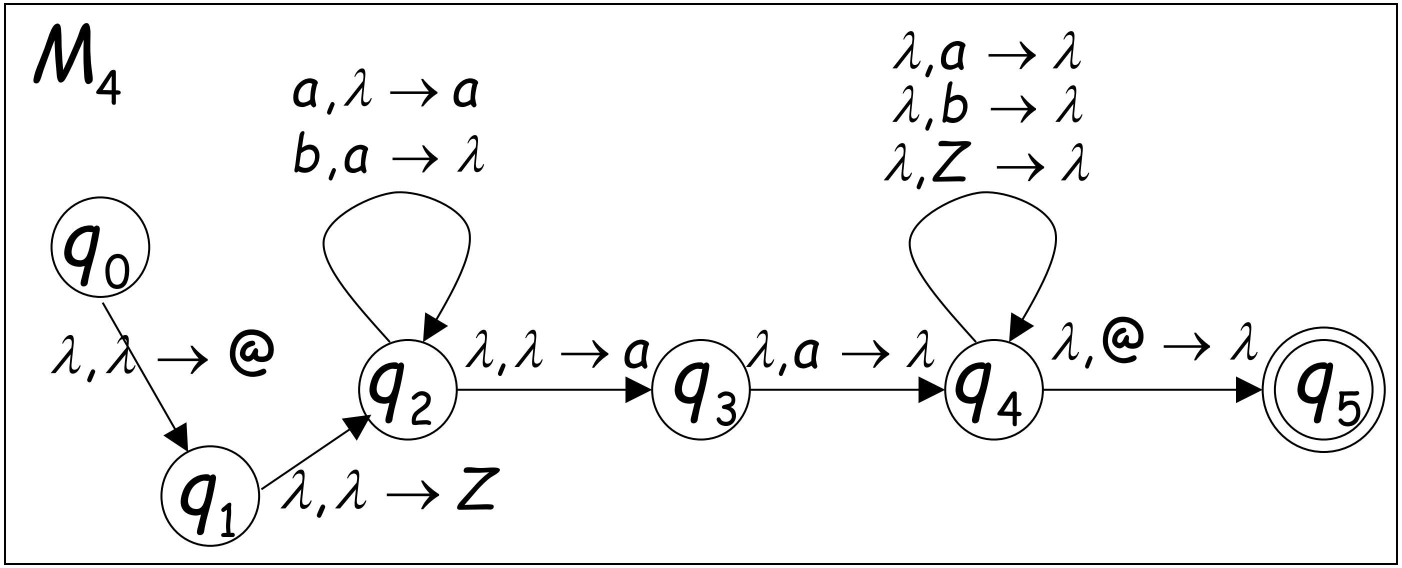
$$A_{q_0q_5} \to A_{q_0q_0}A_{q_0q_5} \mid A_{q_0q_1}A_{q_1q_5} \mid A_{q_0q_2}A_{q_2q_5} \mid A_{q_0q_3}A_{q_3q_5} \mid A_{q_0q_4}A_{q_4q_5} \mid A_{q_0q_5}A_{q_5q_5}$$

$$\vdots$$

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

Start variable $A_{q_0q_5}$

Kind 3: from pairs of transitions



$$A_{q_0q_5} \rightarrow A_{q_1q_4}$$

$$A_{q_1q_4} \rightarrow A_{q_2q_4}$$

$$A_{q_2q_4} \rightarrow aA_{q_2q_4}$$

$$A_{q_2q_2} \rightarrow aA_{q_2q_2}b$$

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

$$A_{q_2q_2} \rightarrow A_{q_3q_2}b$$

$$A_{q_2q_4} \rightarrow A_{q_3q_3}$$

$$A_{q_2q_4} \rightarrow A_{q_3q_4}$$

Suppose that a PDA M is converted to a context-free grammar G

We need to prove that $L(G) = L(M)$

or equivalently

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

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

We need to show that if G has derivation:

$$A_{q_0 q_f} \stackrel{*}{\Rightarrow} w \qquad \text{(string of terminals)}$$

Then there is an accepting computation in M :

$$(q_0, w, \#) \stackrel{*}{\succ} (q_f, \lambda, \#)$$

with input string w

We will actually show that if G has derivation:

$$A_{pq} \stackrel{*}{\Rightarrow} w$$

Then there is a computation in M :

$$(p, w, \lambda) \stackrel{*}{\succ} (q, \lambda, \lambda)$$

Therefore:

$$A_{q_0 q_f} \overset{*}{\Rightarrow} w$$

$$(q_0, w, \lambda) \overset{*}{\succ} (q_f, \lambda, \lambda)$$

Since there is no transition with the # symbol

$$(q_0, w, \#) \overset{*}{\succ} (q_f, \lambda, \#)$$

Lemma:

If $A_{pq} \stackrel{*}{\Rightarrow} w$ (string of terminals)

then there is a computation from state p to state q on string w which leaves the stack empty:

$$(p, w, \lambda) \stackrel{*}{\succ} (q, \lambda, \lambda)$$

Proof Intuition:

$$A_{pq} \Rightarrow \cdots \Rightarrow w$$

Type 2

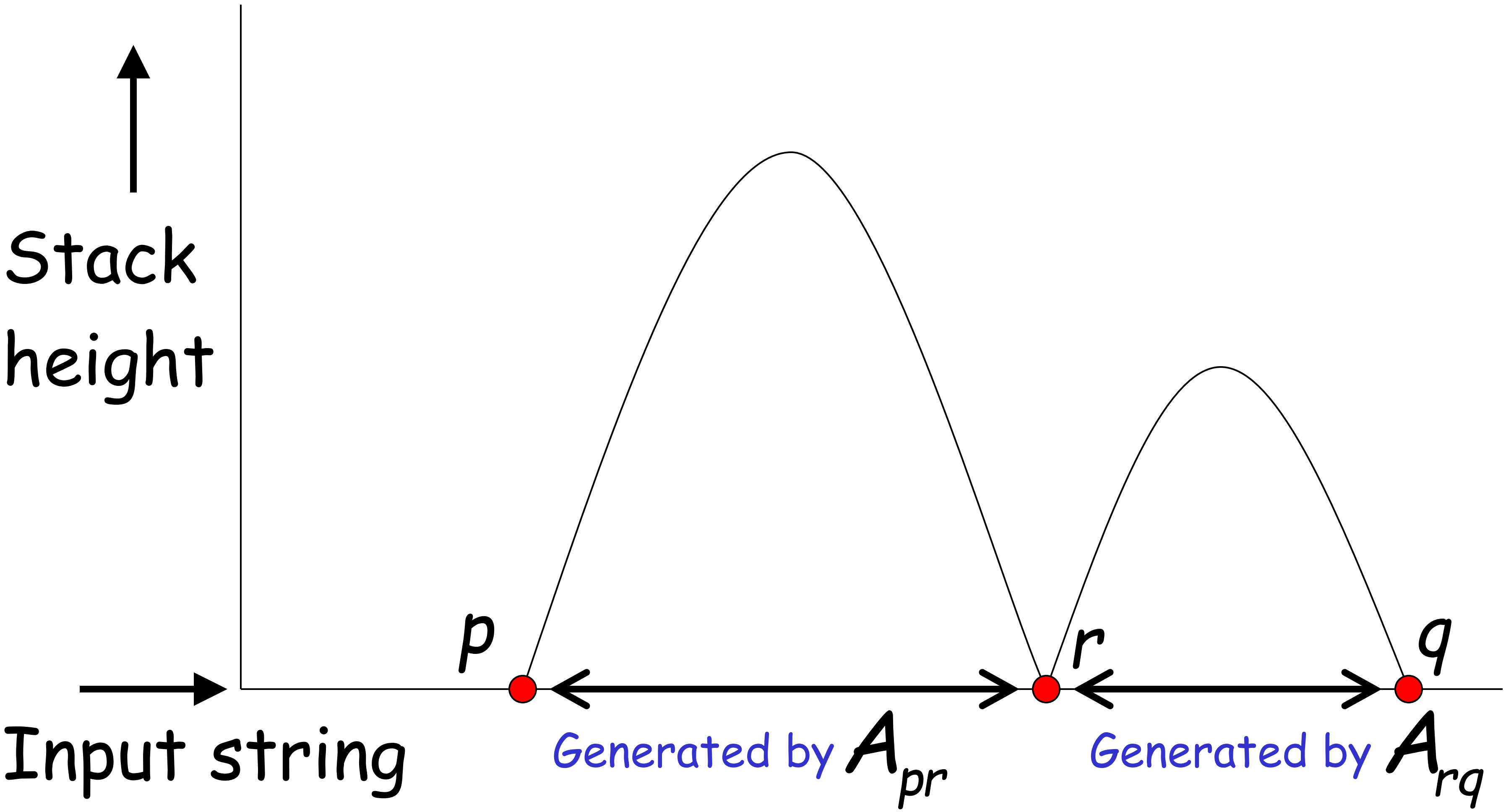
Case 1: $A_{pq} \Rightarrow A_{pr}A_{rq} \Rightarrow \cdots \Rightarrow w$

Type 3

Case 2: $A_{pq} \Rightarrow aA_{rs}b \Rightarrow \cdots \Rightarrow w$

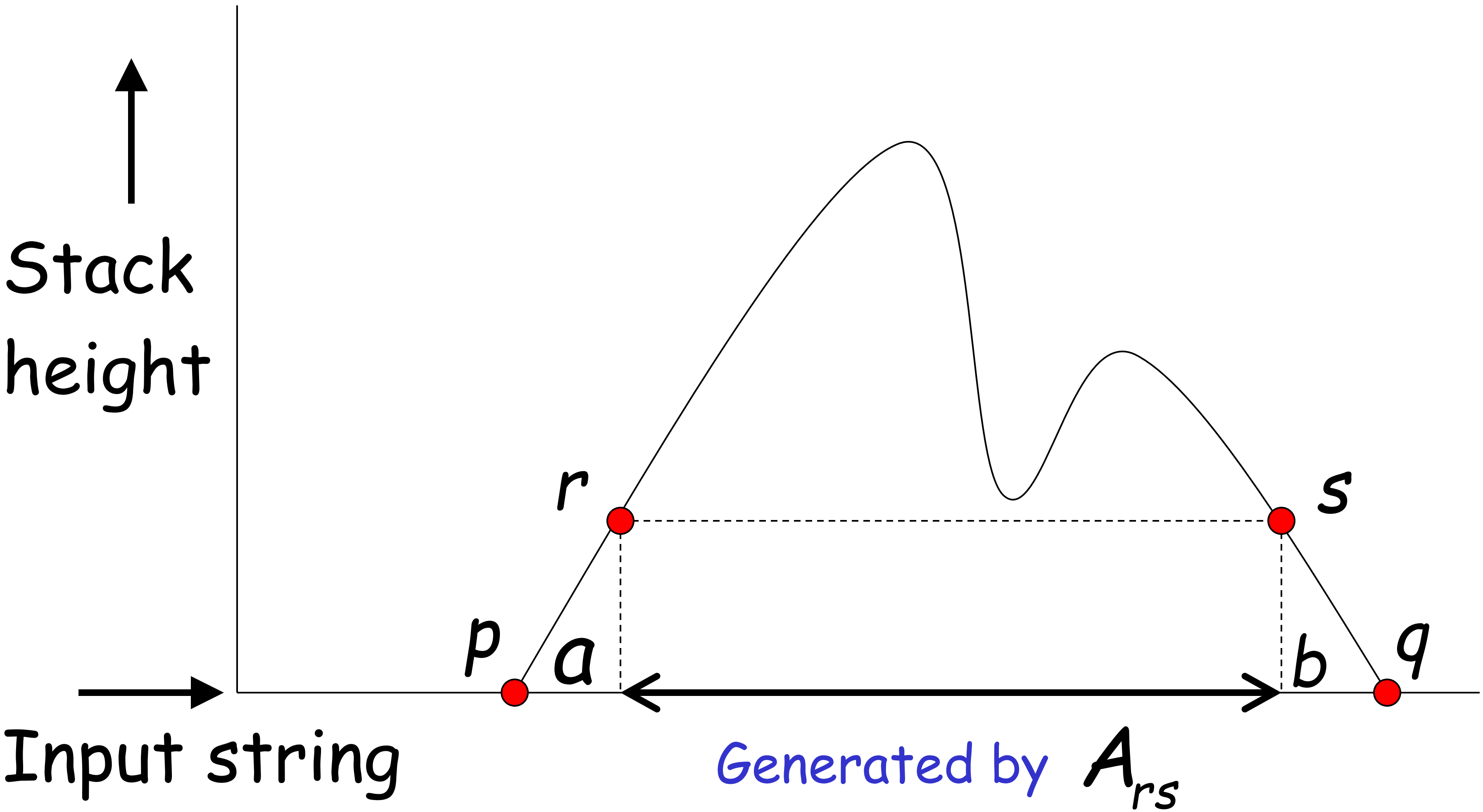
Type 2

Case 1: $A_{pq} \Rightarrow A_{pr} A_{rq} \Rightarrow \cdots \Rightarrow w$



Type 3

Case 2: $A_{pq} \Rightarrow aA_{rs}b \Rightarrow \cdots \Rightarrow w$



Formal Proof:

We formally prove this claim by induction on the number of steps in derivation:

$$A_{pq} \underbrace{\Rightarrow \cdots \Rightarrow}_{\text{number of steps}} w$$

Induction Basis: $A_{pq} \Rightarrow w$

(one derivation step)

A Kind 1 production must have been used:

$$A_{pp} \rightarrow \lambda$$

Therefore, $p = q$ and $w = \lambda$

This computation of PDA trivially exists:

$$(p, \lambda, \lambda) \overset{*}{\succ} (p, \lambda, \lambda)$$

Induction Hypothesis:

$$\underbrace{A_{pq} \Rightarrow \cdots \Rightarrow}_{k} w$$

k derivation steps

suppose it holds:

$$(p, w, \lambda) \overset{*}{\succ} (q, \lambda, \lambda)$$

Induction Step:

$$A_{pq} \underbrace{\Rightarrow \cdots \Rightarrow}_{k+1 \text{ derivation steps}} w$$

We have to show:

$$(p, w, \lambda) \overset{*}{\succ} (q, \lambda, \lambda)$$

$$\underbrace{A_{pq} \Rightarrow \cdots \Rightarrow}_{k+1 \text{ derivation steps}} w$$

Type 2

Case 1: $A_{pq} \Rightarrow A_{pr} A_{rq} \Rightarrow \cdots \Rightarrow w$

Type 3

Case 2: $A_{pq} \Rightarrow a A_{rs} b \Rightarrow \cdots \Rightarrow w$

Type 2

Case 1: $$A_{pq} \underbrace{\Rightarrow A_{pr} A_{rq} \Rightarrow \cdots \Rightarrow}_{k+1 \text{ steps}} w$$

We can write $w = yz$

$$A_{pr} \underbrace{\Rightarrow \cdots \Rightarrow}_{\text{At most } k \text{ steps}} y \qquad A_{rq} \underbrace{\Rightarrow \cdots \Rightarrow}_{\text{At most } k \text{ steps}} z$$

$$A_{pr} \underbrace{\Rightarrow \cdots \Rightarrow}_{\text{At most } k \text{ steps}} y$$

From induction hypothesis, in PDA:

$$(p, y, \lambda) \overset{*}{\succ} (r, \lambda, \lambda)$$

$$A_{rq} \underbrace{\Rightarrow \cdots \Rightarrow}_{\text{At most } k \text{ steps}} z$$

From induction hypothesis, in PDA:

$$(r, z, \lambda) \overset{*}{\succ} (q, \lambda, \lambda)$$

$$(p, y, \lambda) \overset{*}{\succ} (r, \lambda, \lambda) \qquad (r, z, \lambda) \overset{*}{\succ} (q, \lambda, \lambda)$$

$$\Downarrow$$

$$(p, yz, \lambda) \overset{*}{\succ} (r, z, \lambda) \overset{*}{\succ} (q, \lambda, \lambda)$$

since $w = yz$

$$\Downarrow$$

$$(p, w, \lambda) \overset{*}{\succ} (q, \lambda, \lambda)$$

Case 2: $A_{pq} \underbrace{\Rightarrow aA_{rs}b \Rightarrow \cdots \Rightarrow}_{k+1 \text{ steps}} w$

Type 3

We can write $w = ayb$

$A_{rs} \underbrace{\Rightarrow \cdots \Rightarrow}_{\text{At most } k \text{ steps}} y$

$$A_{rs} \underbrace{\Rightarrow \cdots \Rightarrow}_{\text{At most } k \text{ steps}} y$$

$$\Downarrow$$

From induction hypothesis,
the PDA has computation:

$$(r, y, \lambda) \overset{*}{\succ} (s, \lambda, \lambda)$$

Type 3

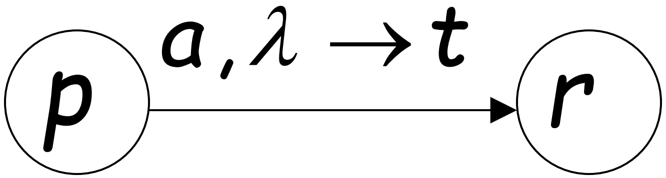
$$A_{pq} \Rightarrow aA_{rs}b \Rightarrow \cdots \Rightarrow w$$

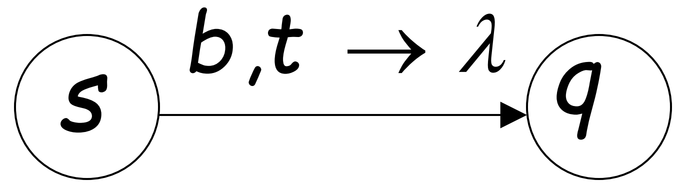
Grammar contains production

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

And PDA Contains transitions

$$\begin{tikzcd} p \arrow[r, "{a, \lambda \rightarrow t}"] & r \end{tikzcd} \qquad \begin{tikzcd} s \arrow[r, "{b, t \rightarrow \lambda}"] & q \end{tikzcd}$$





$$(p, ayb, \lambda) \succ (r, yb, t)$$

$$(s, b, t) \succ (q, \lambda, \lambda)$$

We know

$$(r, y, \lambda) \overset{*}{\succ} (s, \lambda, \lambda) \quad \Longrightarrow \quad (r, yb, t) \overset{*}{\succ} (s, b, t)$$

We also know

$$(p, ayb, \lambda) \succ (r, yb, t)$$

$$(s, b, t) \succ (q, \lambda, \lambda)$$

Therefore:

$$(p, ayb, \lambda) \succ (r, yb, t) \overset{*}{\succ} (s, b, t) \succ (q, \lambda, \lambda)$$

$$(p, ayb, \lambda) \succ (r, yb, t) \overset{*}{\succ} (s, b, t) \succ (q, \lambda, \lambda)$$

since $w = ayb$

$$(p, w, \lambda) \overset{*}{\succ} (q, \lambda, \lambda)$$

END OF PROOF

So far we have shown:

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

With a similar proof we can show

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

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