Pushdown Automata
PDAs

N Geetha
AM & CS
PSG Tech

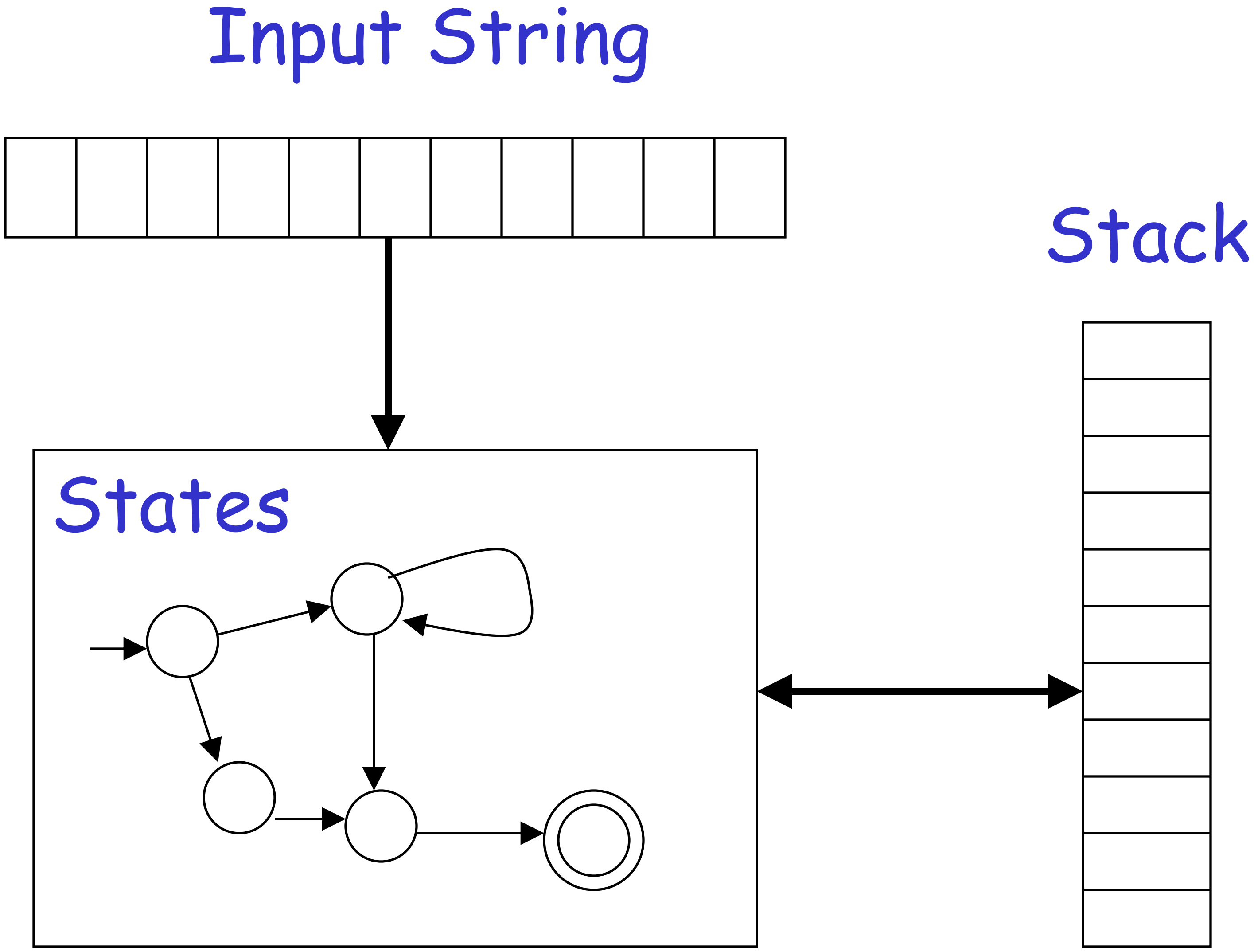
Intuition: PDA

Think of an ϵ-NFA with the additional power that it can manipulate a stack.

Its moves are determined by:

1. current state,
2. current input symbol (or ϵ), and
3. current symbol on top of its stack.

Pushdown Automaton -- PDA



Intuition: PDA – (2)

Being nondeterministic, the PDA can have a choice of next moves.

In each choice, the PDA can:

1. Change state, and also
2. Replace the top symbol on the stack by a sequence of zero or more symbols.
    - Zero symbols = "pop."
    - Many symbols = sequence of "pushes."

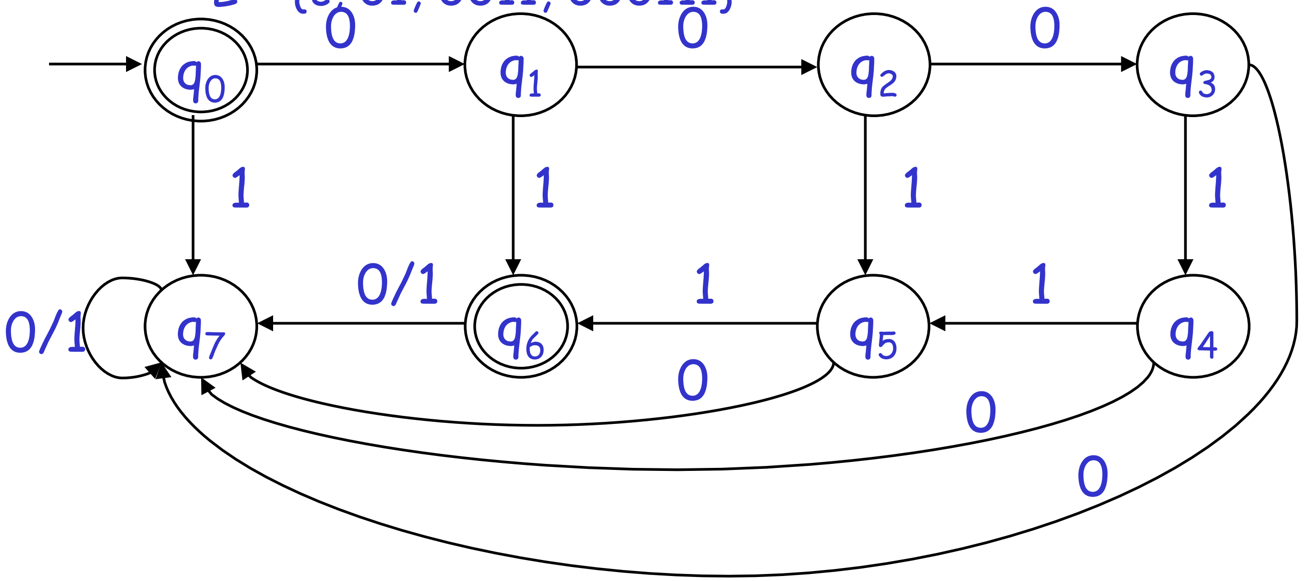
Example:

$\{0^n1^n \mid 0 =< n\}$ Is *not* regular

$\{0^n1^n \mid 0 \leq n \leq k$, for some fixed $k\}$ Is regular, for any fixed k.

For k=3:

$L = \{\varepsilon, 01, 0011, 000111\}$



In a DFA, each state remembers a finite amount of information.

To get $\{0^n1^n \mid 0 \leq n\}$ with a DFA would require an infinite number of states using the preceding technique.

An infinite stack solves the problem for $\{0^n1^n \mid 0 \leq n\}$ as follows:

- Read all 0's and place them on a stack
- Read all 1's and match with the corresponding 0's on the stack

Only need three states to do this in a PDA

Similarly for $\{0^n1^m0^{n+m} \mid n,m \geq 0\}$

Formal Definition of a PDA

A pushdown automaton (PDA) is a seven-tuple:

$M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$

Q	A finite set of states
Σ	A finite set of input alphabet
Γ	A finite set of stack alphabet
q_0	The initial/starting state, q_0 is in Q
z_0	A starting stack symbol, is in Γ
F	A set of final/accepting states, which is a subset of Q
δ	A transition function, where

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

Consider the various parts of δ:

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

- Q on the LHS means that at each step in a computation, a PDA must consider its' current state.
- Γ on the LHS means that at each step in a computation, a PDA must consider the symbol on top of its' stack.
- $\Sigma \cup \{\varepsilon\}$ on the LHS means that at each step in a computation, a PDA may or may not consider the current input symbol, i.e., it may have epsilon transitions.

- "Finite subsets" on the RHS means that at each step in a computation, a PDA will have several options.
- Q on the RHS means that each option specifies a new state.
- Γ^* on the RHS means that each option specifies zero or more stack symbols that will replace the top stack symbol.

Two types of PDA transitions:

$\delta(q, a, z) = \{(p_1,\gamma_1), (p_2,\gamma_2),\ldots, (p_m,\gamma_m)\}$

Current state is q

Current input symbol is a

Symbol currently on top of the stack z

Move to state p_i from q

Replace z with γ_i on the stack (leftmost symbol on top)

Move the input head to the next input symbol

$$\begin{tikzcd}
 & p_1 \\
q \arrow[ur, "{a,z \to \gamma_1}"] \arrow[r, "{a,z \to \gamma_2}"] \arrow[ddr, "{a,z \to \gamma_m}"] & p_2 \\
 & \vdots \\
 & p_m
\end{tikzcd}$$

Two types of PDA transitions:

$\delta(q, \varepsilon, z) = \{(p_1,\gamma_1), (p_2,\gamma_2),\ldots, (p_m,\gamma_m)\}$

Current state is q

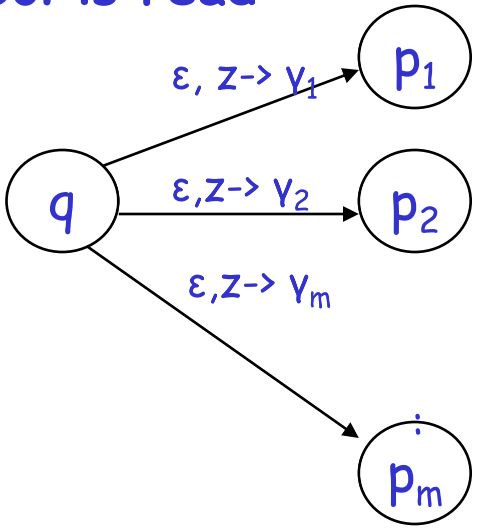
Current input symbol is not considered

Symbol currently on top of the stack z

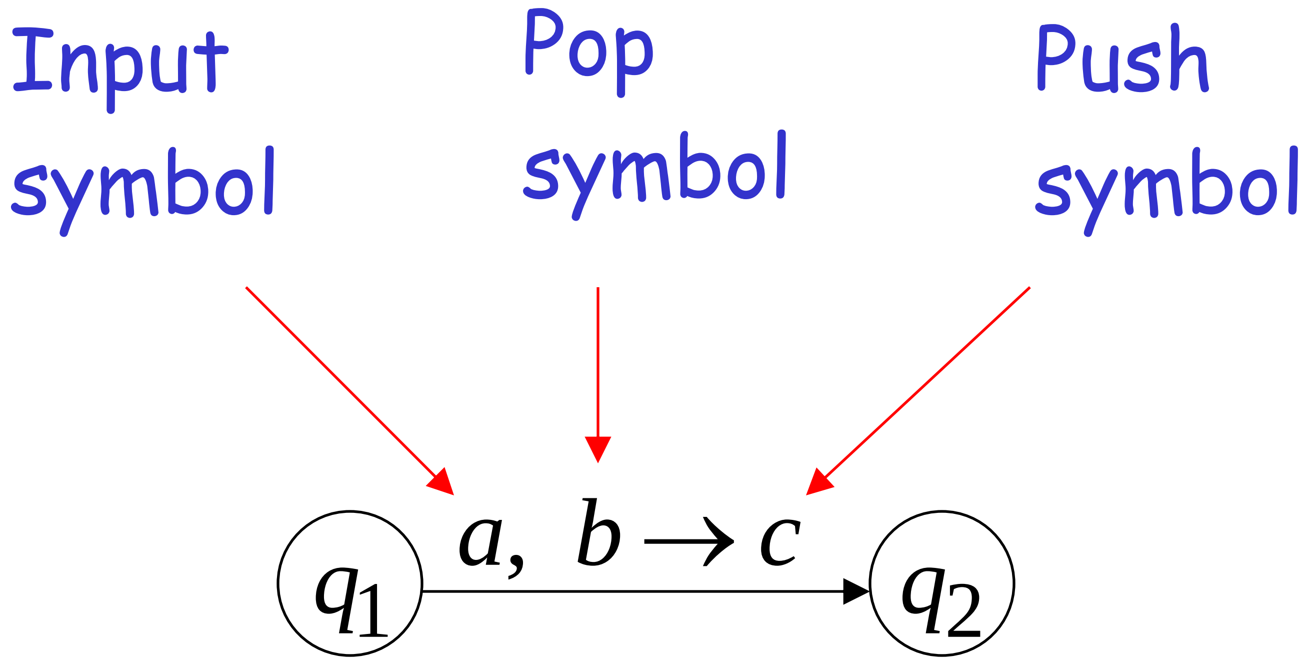
Move to state p_i from q

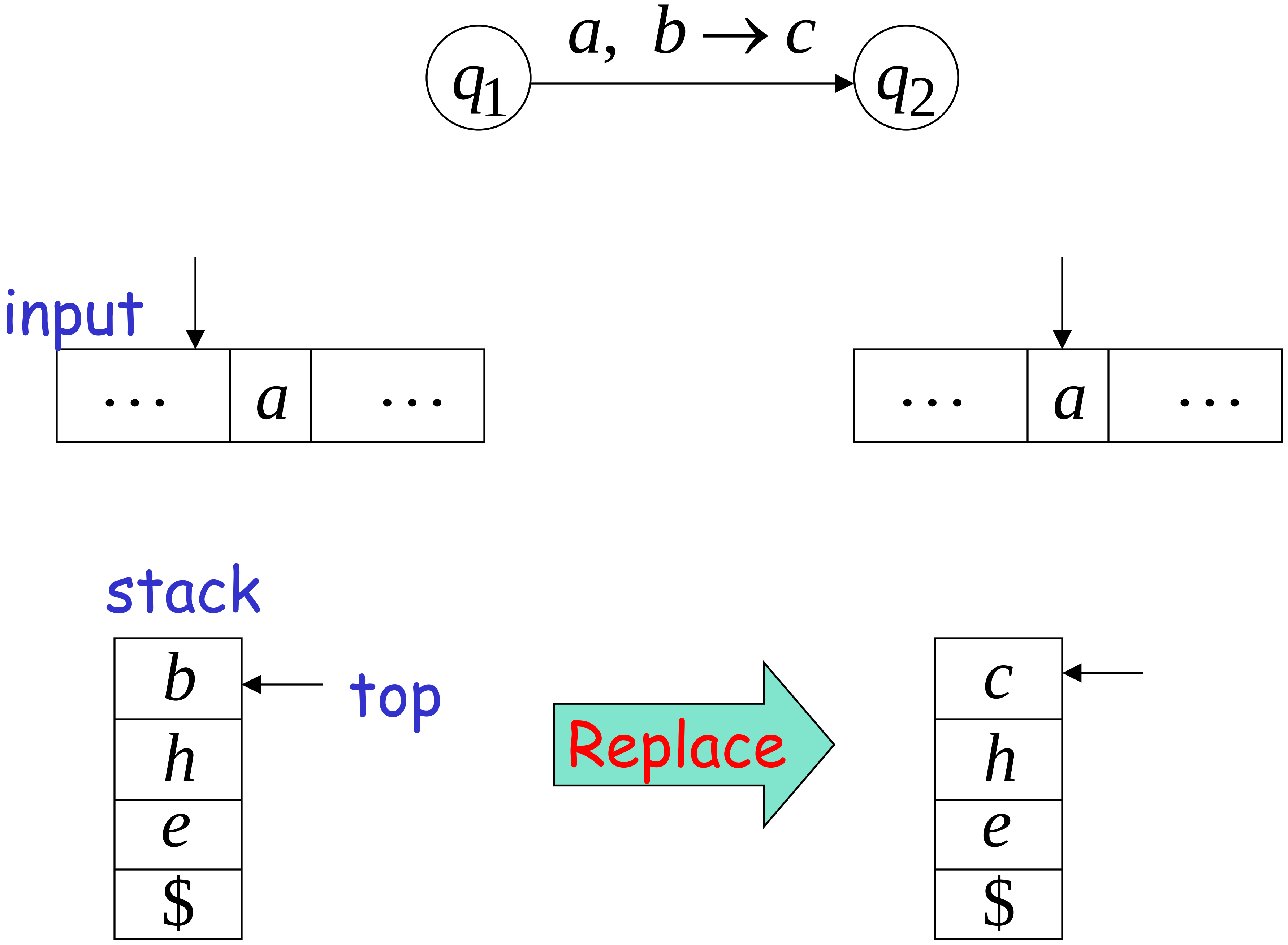
Replace z with γ_i on the stack (leftmost symbol on top)

No input symbol is read

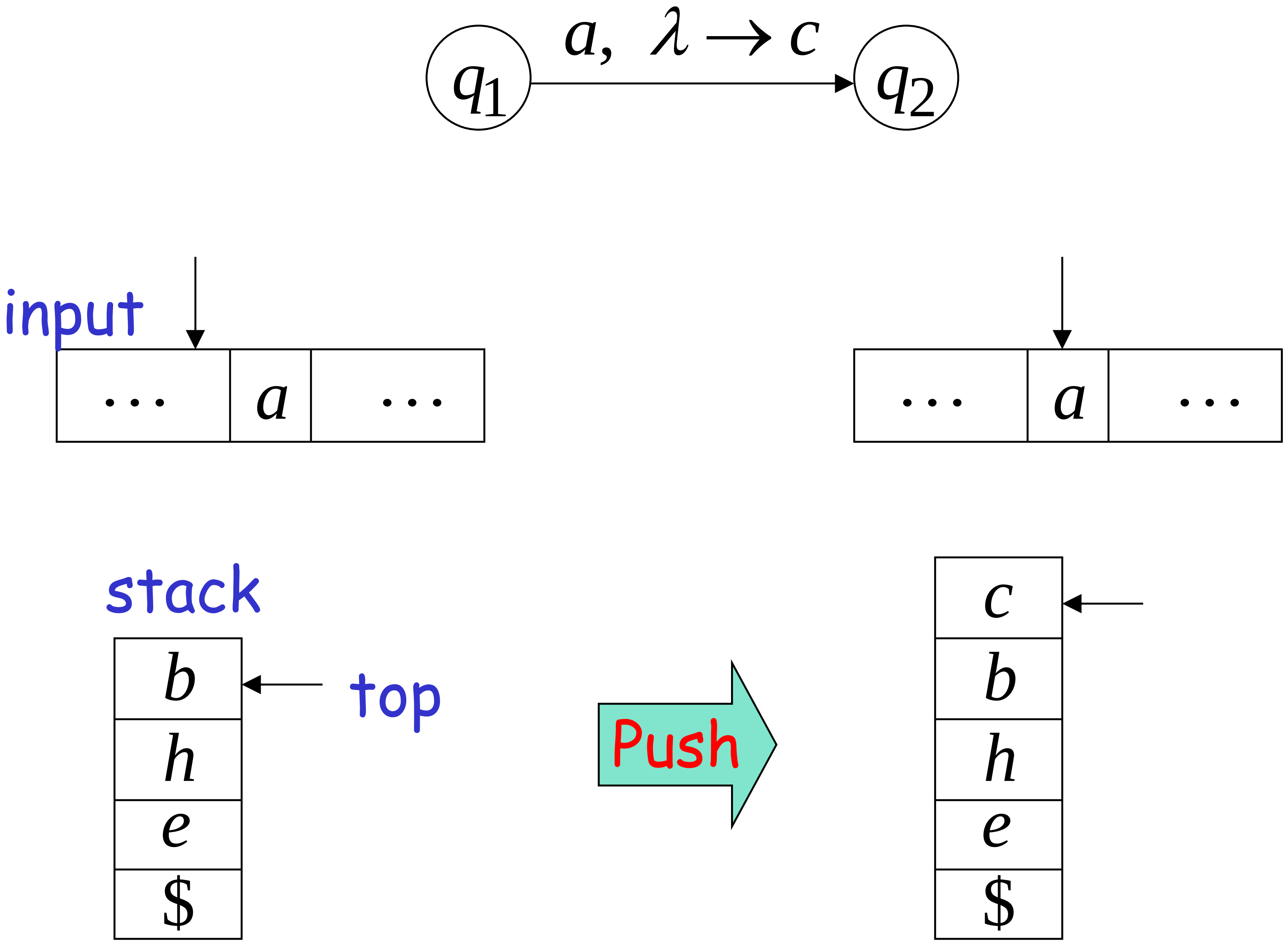


The States

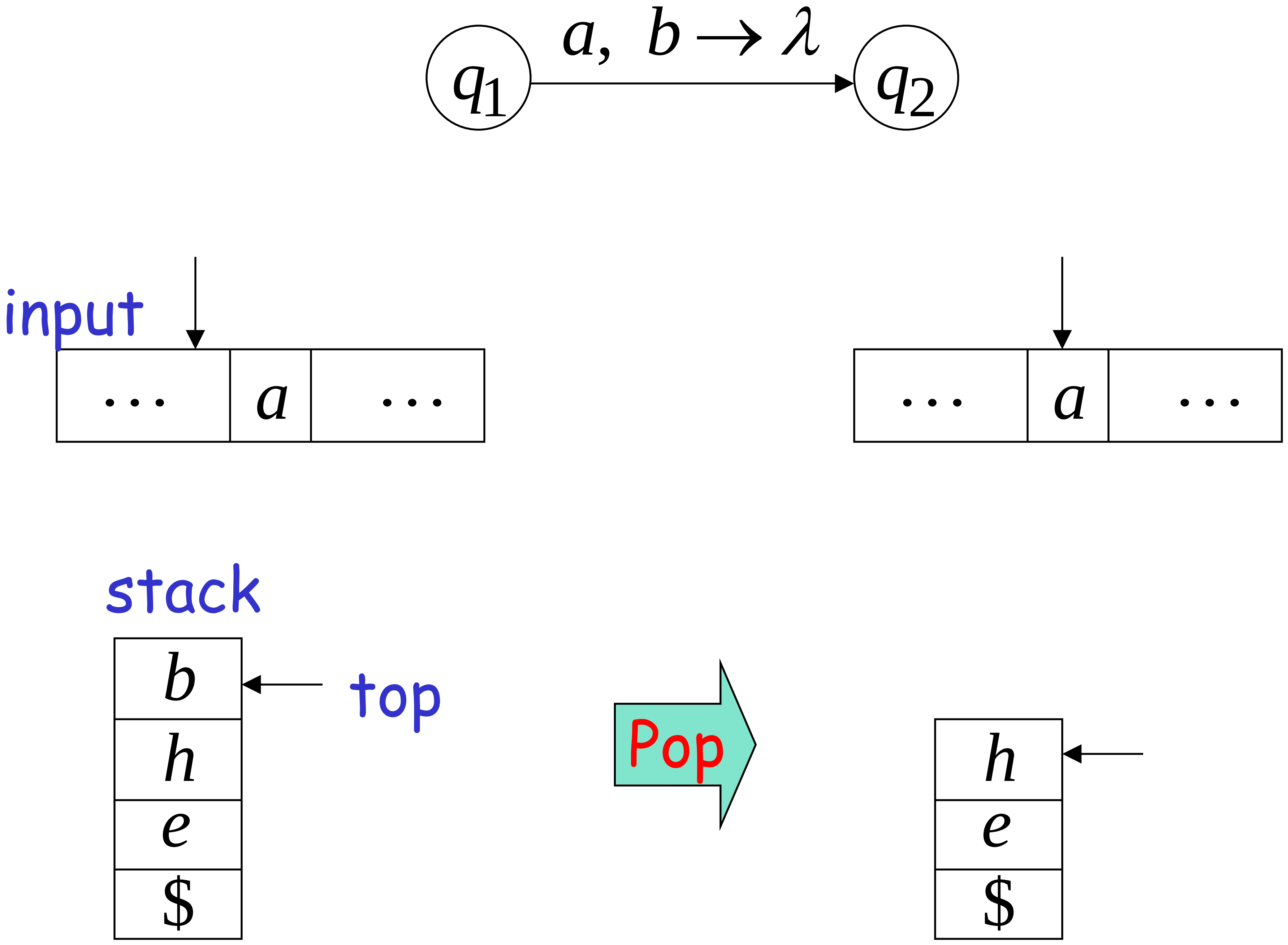




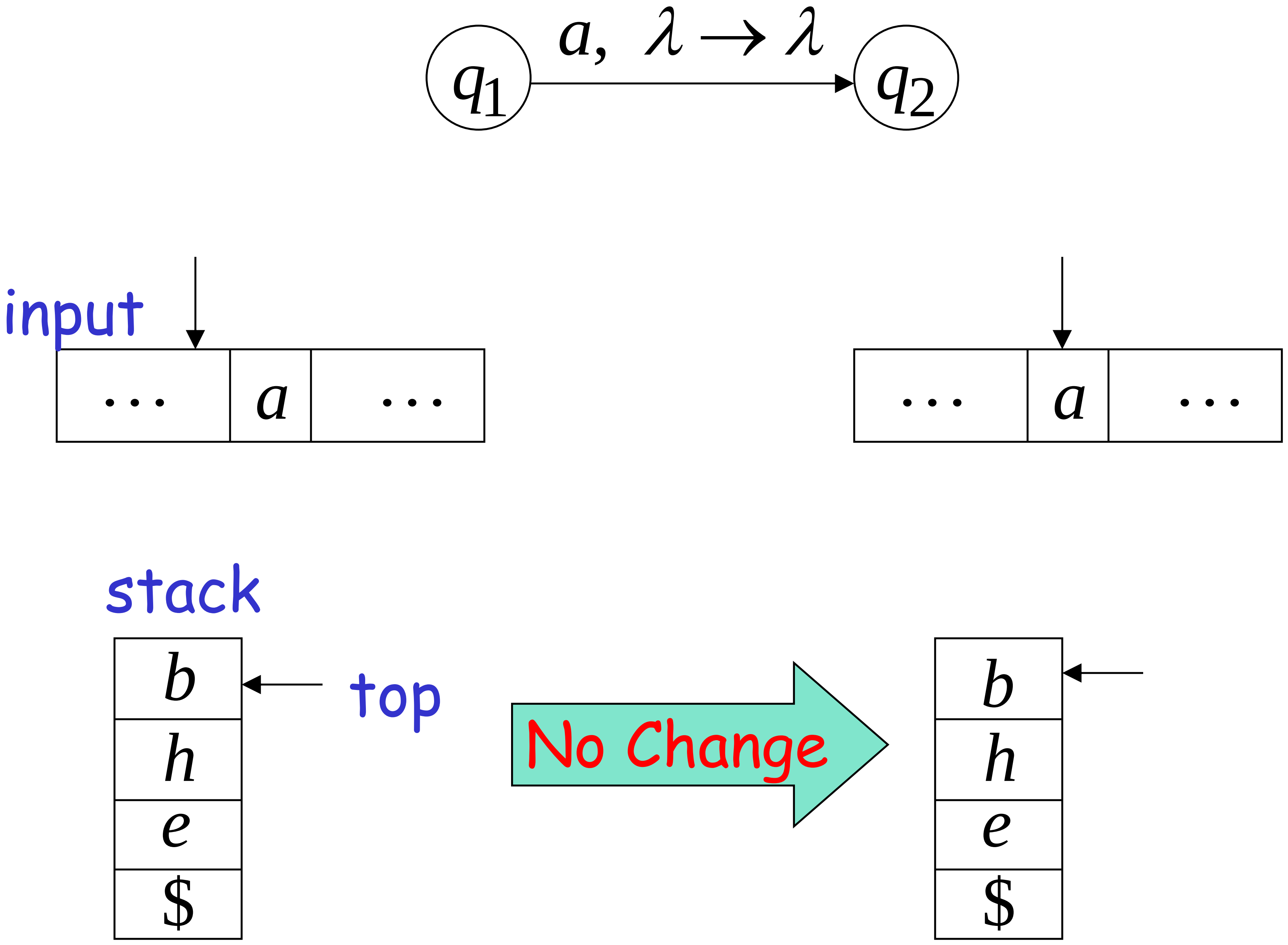
q_1
$a, \; b \rightarrow c$
q_2
input
$\cdots$
a
$\cdots$
stack
b
h
e
$\$$
top
Replace
c
h
e
$\$$



q_1
$a, \lambda \rightarrow c$
q_2
input
$\cdots$ a $\cdots$
$\cdots$ a $\cdots$
stack
b h e $\$$
top
Push
c b h e $\$$



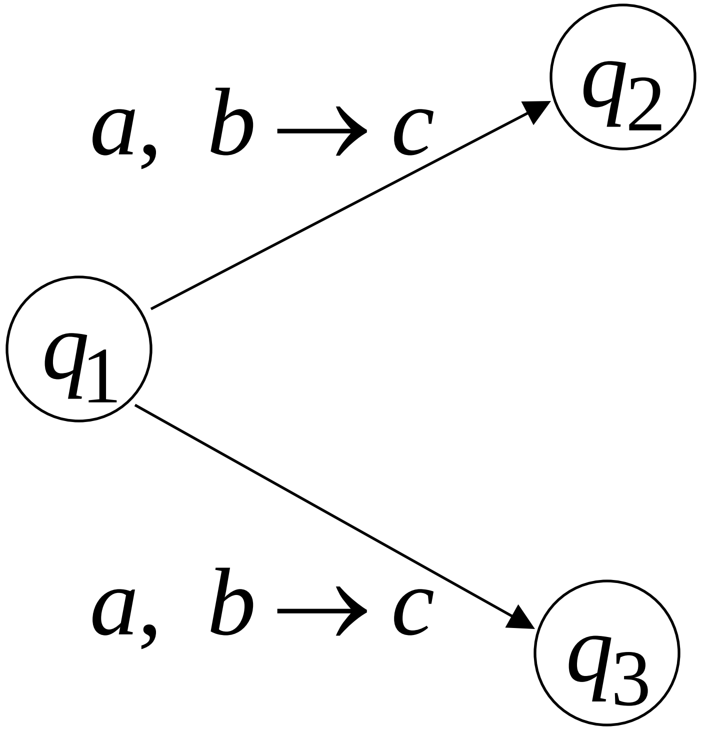
q_1
$a,\ b \to \lambda$
q_2
input
$\cdots$
a
$\cdots$
stack
b
h
e
$\$$
top
Pop

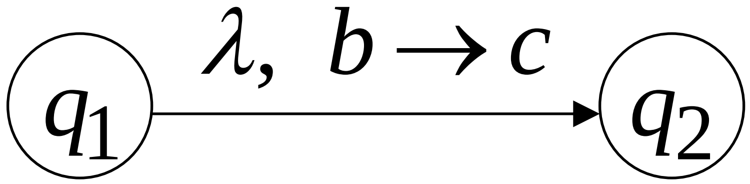

q_1
$a, \lambda \to \lambda$
q_2
input
$\cdots$ a $\cdots$
$\cdots$ a $\cdots$
stack
b
h
e
$\$$
top
No Change
b
h
e
$\$$

Non-Determinism

PDAs are non-deterministic

Allowed non-deterministic transitions


$$\begin{tikzcd} & q_2 \\ q_1 \arrow[ur, "{a,\ b \to c}"] \arrow[dr, "{a,\ b \to c}"'] & \\ & q_3 \end{tikzcd}$$


$$\begin{tikzcd} q_1 \arrow[r, "{\lambda,\ b \to c}"] & q_2 \end{tikzcd}$$

$\lambda - \text{transition}$

Example: PDA

Design a PDA to accept $\{0^n1^n \mid n \geq 1\}$.

The states:

- q = start state. We are in state q if we have seen only 0's so far.
- p = we've seen at least one 1 and may now proceed only if the inputs are 1's.
- f = final state; accept.

Example: PDA – (2)

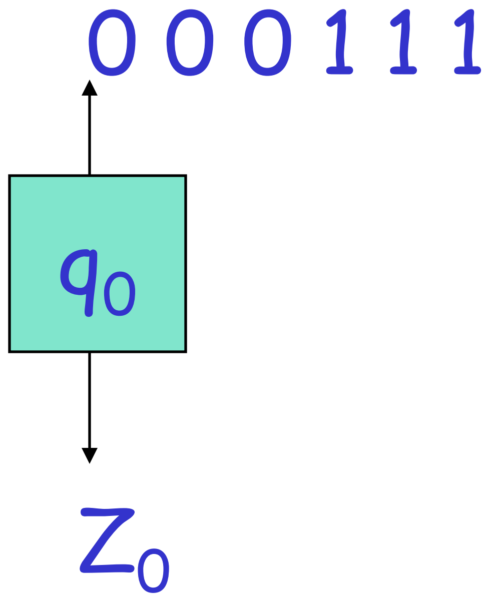
The stack symbols:

- Z_0 = start symbol. Also marks the bottom of the stack.

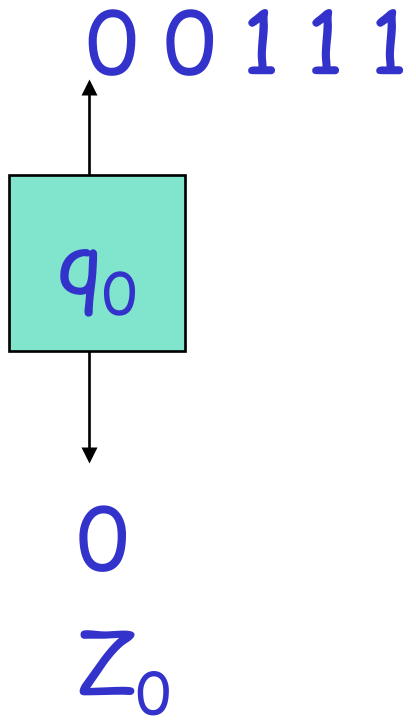
The transitions:

- $\delta(q_0, 0, Z_0) = \{(q_0, 0Z_0)\}$.
- $\delta(q_0, 0, 0) = \{(q_0, 00)\}$. These two rules cause one 0 to be pushed onto the stack for each 0 read from the input.
- $\delta(q_0, 1, 0) = \{(q_1, \epsilon)\}$. When we see a 1, go to state p and pop one 0.
- $\delta(q_1, 1, 0) = \{(q_1, \epsilon)\}$. Pop one 0 per 1.
- $\delta(q_1, \epsilon, Z_0) = \{(q_f, Z_0)\}$. Accept at bottom.

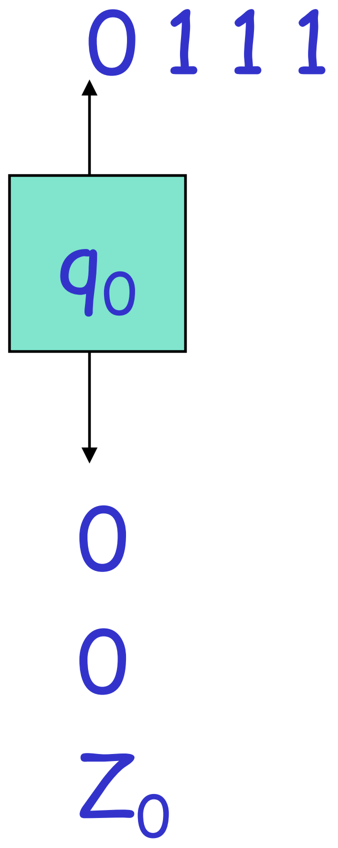
Actions of the Example PDA



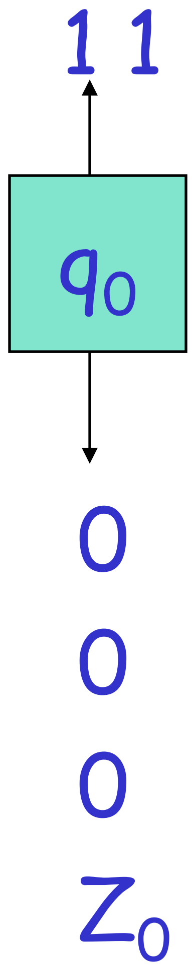
Actions of the Example PDA



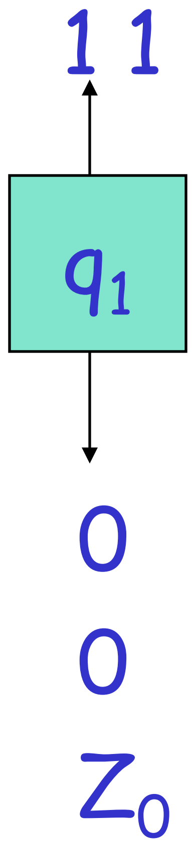
Actions of the Example PDA



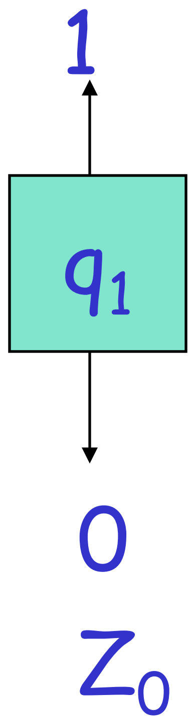
Actions of the Example PDA



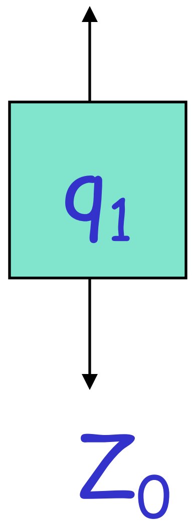
Actions of the Example PDA



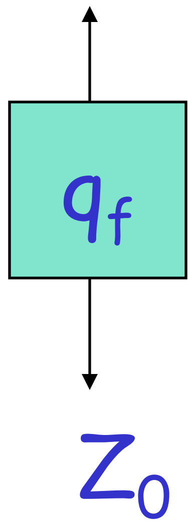
Actions of the Example PDA



Actions of the Example PDA



Actions of the Example PDA



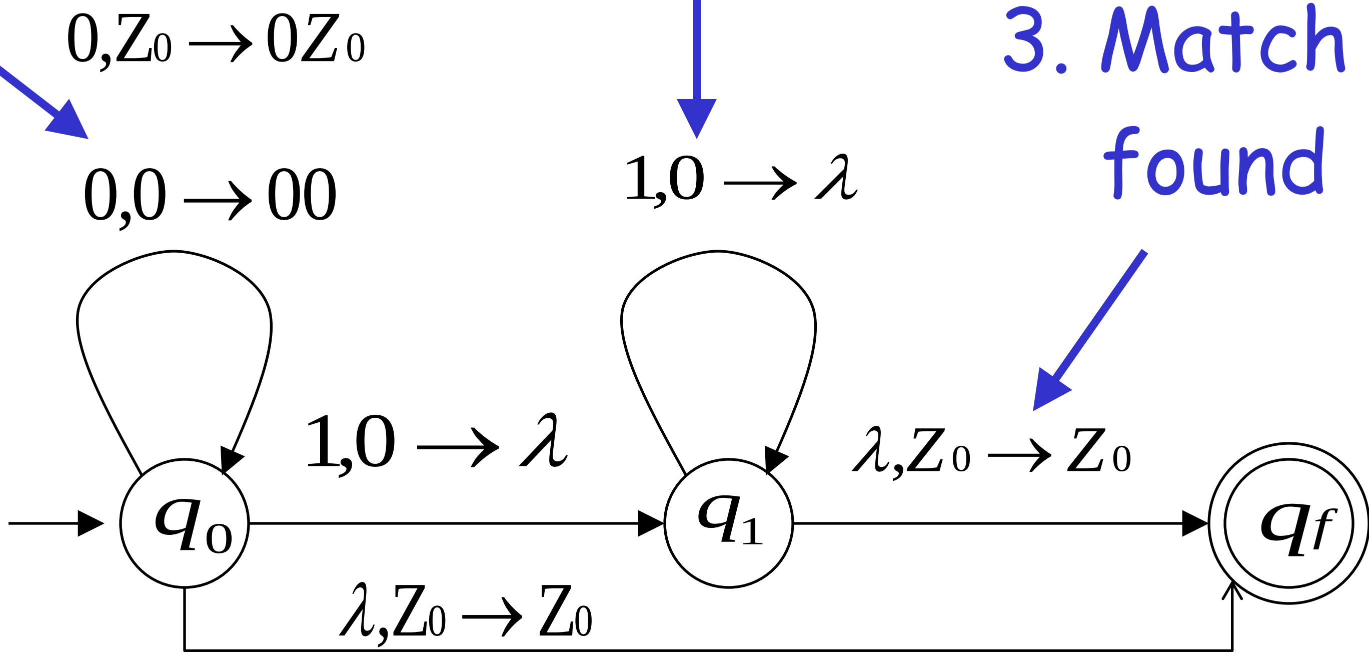
$$T(M) = \{0^n 1^n : n \geq 0\}$$

Basic Idea:

1. Push the 0's on the stack

2. Match the 1's on input with 0's on stack

3. Match found



$0,Z_0 \rightarrow 0Z_0$
$0,0 \rightarrow 00$
$1,0 \rightarrow \lambda$
$1,0 \rightarrow \lambda$
$\lambda,Z_0 \rightarrow Z_0$
$\lambda,Z_0 \rightarrow Z_0$
q_0
q_1
q_f

Instantaneous Descriptions

We can formalize the pictures just seen with an *instantaneous description* (ID).

A ID is a triple (q, w, α), where:

1. q is the current state.
2. w is the remaining input.
3. α is the stack contents, top at the left.

The MOVE Relation

- To say that ID I can become ID J in one move of the PDA, we write $I \vdash J$.
- Formally, $(q, aw, X\alpha) \vdash (p, w, \beta\alpha)$ for any w and α, if $\delta(q, a, X)$ contains (p, β).
- Extend $\vdash$ to $\vdash^*$, meaning "zero or more moves," by:
  - Basis: $I \vdash^* I$.
  - Induction: If $I \vdash^* J$ and $J \vdash K$, then $I \vdash^* K$.

Example: Move Relation

Using the previous example PDA, we can describe the sequence of moves by:

$(q_0, 000111, Z_0) \vdash (q_0, 00111, 0Z_0)$

$\vdash (q_0, 0111, 00Z_0)$

$\vdash (q_0, 111, 000Z_0)$

$\vdash (q_1, 11, 00Z_0)$

$\vdash (q_1, 1, 0Z_0) \vdash (q_1, \epsilon, Z_0)$

$\vdash (q_f, \epsilon, Z_0)$

Thus, $(q_0, 000111, Z_0) \vdash^* (q_f, \epsilon, Z_0)$.

What would happen on input 0001111?

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

All the input is consumed

AND

The last state is an accepting state

At the end of the computation,
we do not care about the stack contents
(the stack can be empty at the last state)

Answer

Legal because a PDA can use ϵ input even if input remains.

$(q_0, 0001111, Z_0) \vdash (q_0, 001111, 0Z_0)$

$\vdash (q_0, 01111, 00Z_0)$

$\vdash (q_0, 1111, 000Z_0)$

$\vdash (q_1, 111, 00Z_0)$

$\vdash (q_1, 11, 0Z_0)$

$\vdash (q_1, 1, Z_0) \vdash (q_f, 1, Z_0)$

Note the last ID has no move.

0001111 is not accepted, because the input is not completely consumed.

Language of a PDA

The common way to define the language of a PDA is by *final state.*

If M is a PDA, then T(M) is the set of strings w such that $(q_0, w, Z_0) \vdash^* (q_f, \epsilon, \alpha)$ for final state q_f and any α.

Language of a PDA – (2)

Another language defined by the same PDA is by *empty stack*.

If M is a PDA, then N(M) is the set of strings w such that $(q_0, w, Z_0) \vdash^* (q, \epsilon, \epsilon)$ for any state q.

Example 1: (balanced parentheses)

ACCEPTANCE BY EMPTY STACK

$M = (\{q_0, q_1, q_2\}, \{\text{“(“}, \text{“)”}\}, \{\text{“(“}, Z_0\}, \delta, q_0, Z_0, \varnothing)$

δ:

(1) $\delta(q_0, (, Z_0) = \{(q_1, (\ Z_0)\}$

(2) $\delta(q_1, (, (\ ) = \{(q_1, (\ (\ )\}$

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

(4) $\delta(q_1, \varepsilon, Z_0) = \{(q_2, \varepsilon)\}$

Goal: (acceptance)

Terminate in a non-null state

Read the entire input string

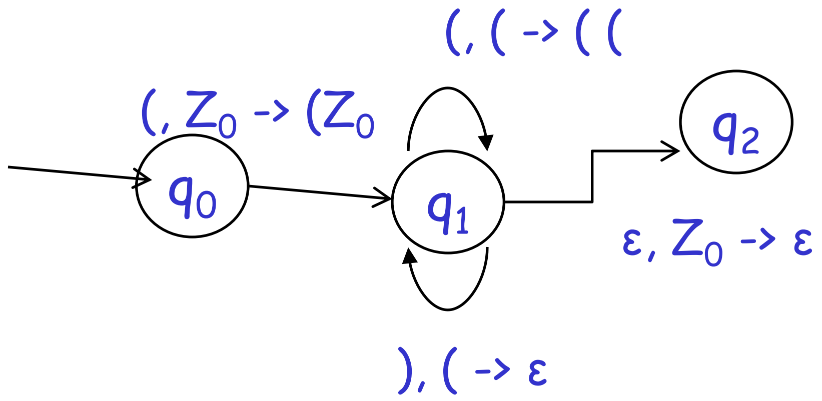
Terminate with an empty stack

Informally, a string is accepted if there exists a computation that uses up all the input and leaves the stack empty.

Deterministic PDA's

- To be deterministic, there must be at most one choice of move for any state q, input symbol a, and stack symbol X.
- In addition, there must not be a choice between using input ϵ or real input.
- Formally, $\delta(q, a, X)$ and $\delta(q, \epsilon, X)$ cannot both be nonempty.

Transition Diagram:



Example Computation:

Current Input	Stack	Transition
(())	Z_0	
())	(Z_0	(1)
))	((Z_0	(2)
)	(Z_0	(3)
ε	Z_0	(3)
ε	-	(4)

Example 2: For the language $\{x \mid x = wcw^r \text{ and } w \text{ in } \{0,1\}^*\}$

$M = (\{q_0, q_1, q_2, q_3\}, \{0, 1, c\}, \{0, 1\}, \delta, q_0, Z_0, \emptyset)$

δ:

(1)	$\delta(q_0, 0, Z_0) = \{(q_1, 0\,Z_0)\}$	(9)	$\delta(q_0, 1, Z_0) = \{(q_1, 1\,Z_0)\}$
(2)	$\delta(q_1, 0, 0) = \{(q_1, 0\,0)\}$	(10)	$\delta(q_1, 1, 0) = \{(q_1, 1\,0)\}$
(3)	$\delta(q_1, 0, 1) = \{(q_1, 0\,1)\}$	(11)	$\delta(q_1, 1, 1) = \{(q_1, 1\,1)\}$
(4)	$\delta(q_1, c, Z_0) = \{(q_2, Z_0)\}$		
(5)	$\delta(q_1, c, 0) = \{(q_2, 0)\}$		
(6)	$\delta(q_1, c, 1) = \{(q_2, 1)\}$		
(7)	$\delta(q_2, 0, 0) = \{(q_2, \varepsilon)\}$	(12)	$\delta(q_2, 1, 1) = \{(q_2, \varepsilon)\}$
(8)	$\delta(q_2, \varepsilon, Z_0) = \{(q_3, \varepsilon)\}$		

Notes:

Only rule #8 is non-deterministic.

Rule #8 is used to pop the final stack symbol off at the end of a computation.

Example Computation:

(1)	$\delta(q_0, 0, Z_0) = \{(q_1, 0\ Z_0)\}$	(9)	$\delta(q_0, 1, Z_0) = \{(q_1, 1\ Z_0)\}$
(2)	$\delta(q_1, 0, 0) = \{(q_1, 0\ 0\ )\}$	(10)	$\delta(q_1, 1, 0) = \{(q_1,\ 1\ 0)\}$
(3)	$\delta(q_1, 0, 1) = \{(q_1, 0\ 1)\}$	(11)	$\delta(q_1, 1, 1) = \{(q_1, 1\ 1)\}$
(4)	$\delta(q_1, c, Z_0) = \{(q_2, Z_0)\}$		
(5)	$\delta(q_1, c, 0) = \{(q_2, 0)\}$		
(6)	$\delta(q_1, c, 1) = \{(q_2, 1)\}$		
(7)	$\delta(q_2, 0, 0) = \{(q_2, \varepsilon)\}$	(12)	$\delta(q_2, 1, 1) = \{(q_2, \varepsilon)\}$
(8)	$\delta(q_2, \varepsilon, Z_0) = \{(q_3, \varepsilon)\}$		

State	Input	Stack	Rule Applied	Rules Applicable
q_0	**01c10**	Z_0		(1)
q_1	**1**c10	$0\ Z_0$	(1)	(10)
q_1	**c**10	$10\ Z_0$	(10)	(6)
q_2	**10**	$10\ Z_0$	(6)	(12)
q_2	**0**	$0\ Z_0$	(12)	(7)
q_2	**ε**	Z_0	(7)	(8)
q_3	ε	ε	(8)	-

Example 3: For the language $\{x \mid x = ww^r \text{ and } w \text{ in } \{0,1\}^*\}$ **(ONLY NPDA)**

$M = (\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, Z_0\}, \delta, q_0, Z_0, \emptyset)$

δ:

(1)	$\delta(q_0, 0, Z_0) = \{(q_0, 0Z_0), (q_1, Z_0)\}$
(2)	$\delta(q_0, 1, Z_0) = \{(q_0, 1Z_0), (q_1, Z_0)\}$
(3)	$\delta(q_0, 0, 0) = \{(q_0, 00), (q_1, 0)\}$
(4)	$\delta(q_0, 0, 1) = \{(q_0, 01), (q_1, 1)\}$
(5)	$\delta(q_0, 1, 0) = \{(q_0, 10), (q_1, 0)\}$
(6)	$\delta(q_0, 1, 1) = \{(q_0, 11), (q_1, 1)\}$
(7)	$\delta(q_0, \varepsilon, Z_0) = \{(q_1, Z_0)\}$
(8)	$\delta(q_0, \varepsilon, 0) = \{(q_1, a)\}$
(9)	$\delta(q_0, \varepsilon, 1) = \{(q_1, b)\}$
(10)	$\delta(q_1, 1, 1) = \{(q_1, \varepsilon)\}$
(11)	$\delta(q_1, 0, 0) = \{(q_1, \varepsilon)\}$
(12)	$\delta(q_1, \varepsilon, Z_0) = \{(q_2, \varepsilon)\}$

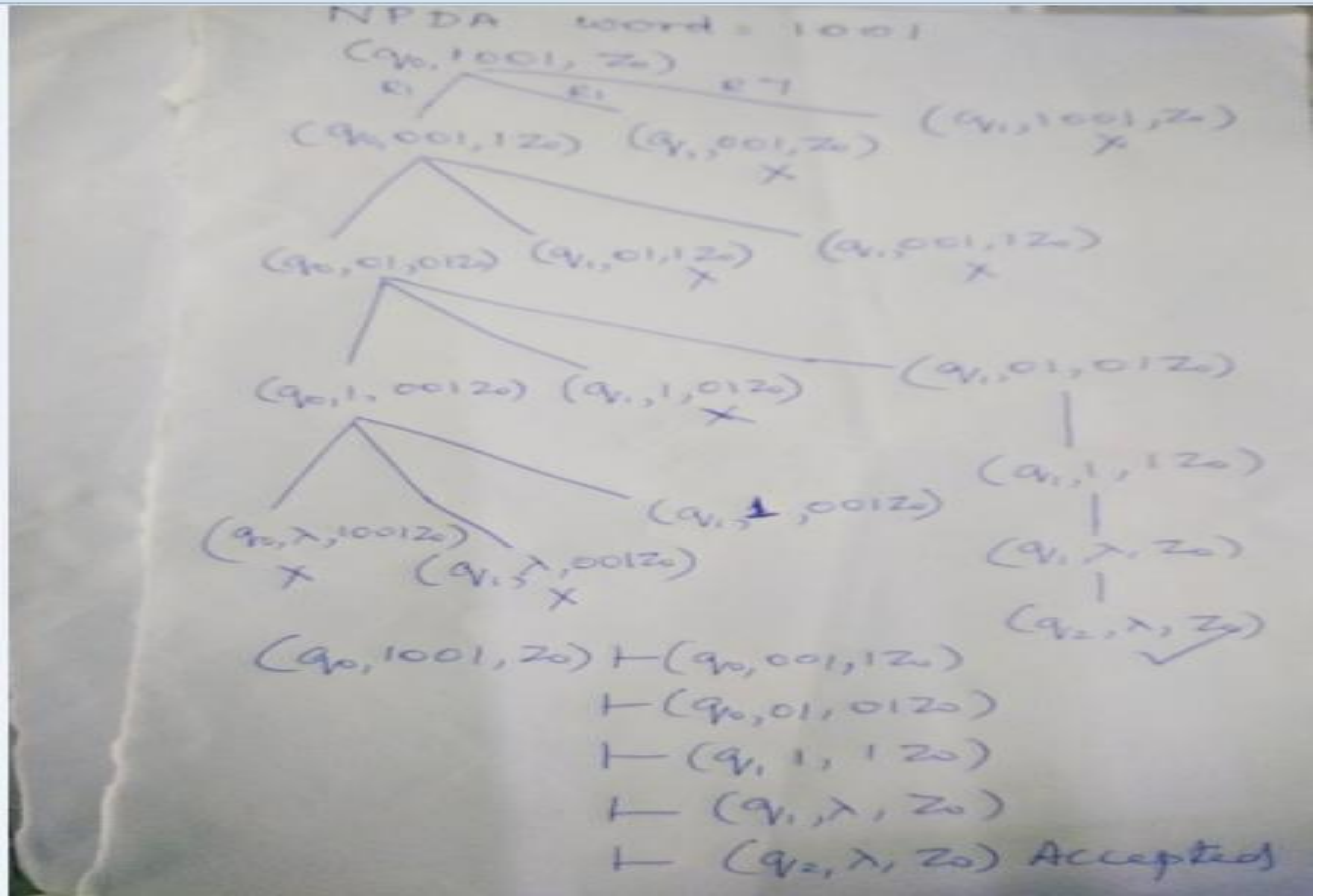
Notes:

Rules #1 to #6 are non-deterministic.

Rules #10 and #11 are used to pop the final stack symbol off

Rules #7 to #9 are used for mid of even length string. Eg 0000, 0110

NPDA derivation for 1001



$(q_0, 1001, Z_0) \vdash (q_0, 001, 1Z_0)$

$\vdash (q_0, 01, 01Z_0)$

$\vdash (q_1, 1, 1Z_0)$

$\vdash (q_1, \lambda, Z_0)$

$\vdash (q_2, \lambda, Z_0)$ Accepted