

Unit-5Push Down Automata (PDA)

Introduction:- The context free languages have a type of automaton that defines them. This automaton is called "pushdown automaton" which can be thought as a  $\epsilon$ -NFA with the addition of stack. The presence of stack means that, the pushdown automata can remember infinity amount of information. However, the pushdown automaton can only access the information on its stack in a last-in-first-out way.

We can define PDA informally as shown in figure:

PDA is an abstract machine determined by following three things:

- Input tape
- Finite state control
- A stack.

Each moves of the machine is determined by three things;

- The current state.
- Next input symbol
- Symbol on the top of stack.

The moves consist of;

→ Changing state | staying on same state.

→ Replacing the stack top by string of zero or more symbols.

Popping the top symbol off the stack means replacing it by  $\epsilon$ .

Pushing  $y$  on the stack means replacing stack's top, say  $x$ , by  $yx$ .

Formal Definition:

A PDA is defined by seven tuples  $(Q, \Sigma, \Gamma, S, q_0, z_0, F)$  where,

$Q$  = A finite set of states.

$\Sigma$  = A finite set of input symbols.

$\Gamma$  = A finite stack Alphabets.

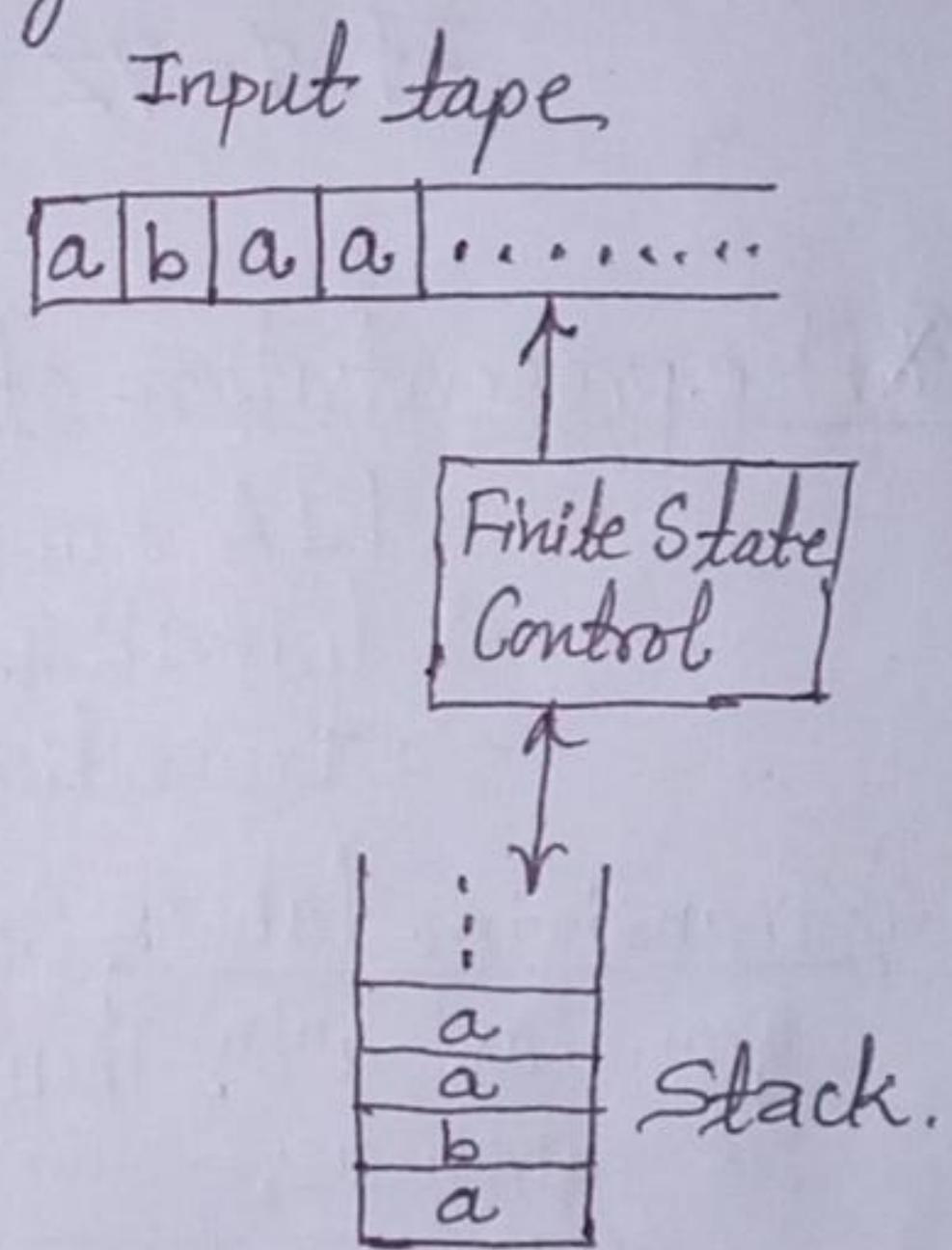
$S$  = transition function

$q_0$  = The start state.

Gramma or  
uppercase gamma

Gramma  
 $\rightarrow$   
del

$z_0$  = The Start stack symbol  
 $F$  = The set of Final/Accepting States.



Here,  $S$  takes as argument a triple  $S(q, a, X)$  where:

- i)  $q$  is a state in  $Q$ .
- ii)  $a$  is either an input symbol  $m \in \Sigma$  or  $a = \epsilon$ .
- iii)  $X$  is a stack symbol, that is a member of  $\Gamma$ .

The output of  $S$  is finite set of pairs  $(p, T)$  where;

$p$  is a new state.

$T$  is a string of stack symbols that replaces  $X$  at the top of the stack.

Example:

If  $T = \epsilon$ , then the stack is popped.

If  $T = X$  then the stack is unchanged.

If  $T = YZ$  then  $X$  is replaced by  $Z$  and  $Y$  is pushed onto the stack.

## \* Representation of PDA:

PDA can be represented by using two techniques:

→ transition table

→ transition diagram.

i) Transition table: Consider a context free language defined by the grammar  $a^n b^n$ . Then the transition table used to represent this language is given as;

Move Number	State	Input	Stack Symbol	Move(s)
1	$q_0$	A	$z_0$	$(q_0, a z_0)$
2	$q_0$	A	A	$(q_0, a)$
3	$q_0$	$\epsilon$	A	$(q_0, a)$
4	$q_1$	B	A	$(q_1, \epsilon)$
5	$q_1$	$\epsilon$	$z_0$	$(q_1, z_0)$

Here,  $q_0$  is the starting state of PDA,  $q_1$  is the final state of PDA and  $z_0$  is the initial stack symbol.

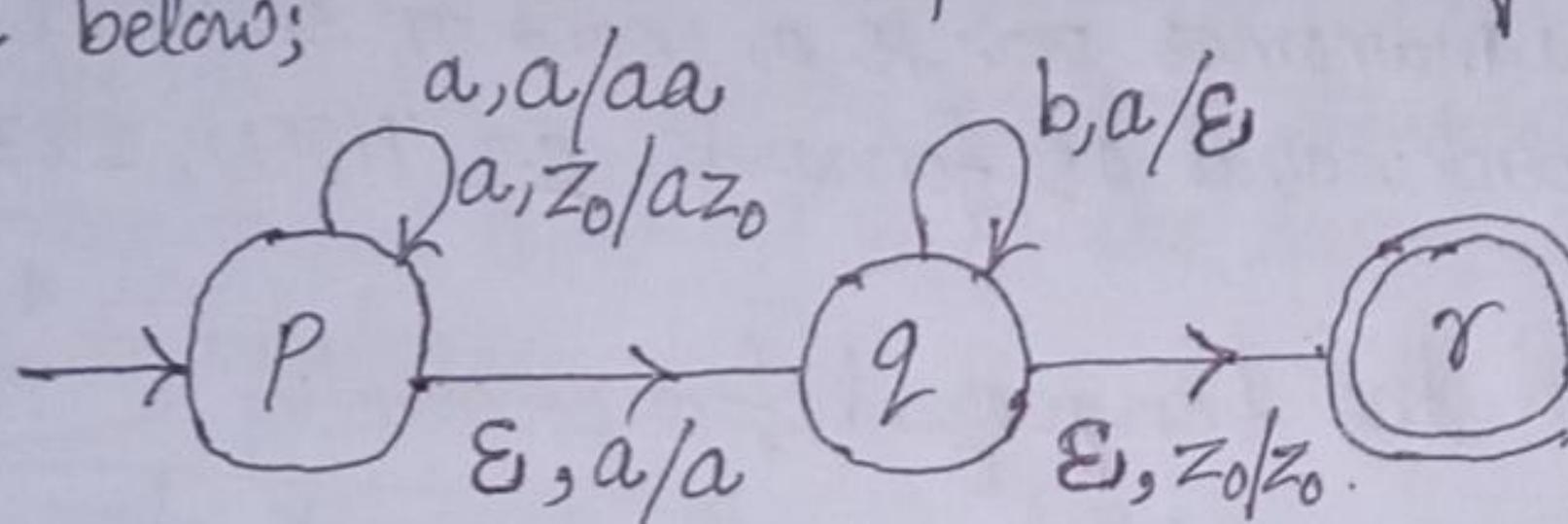
$q_1$  on getting input B to top of stack A switches the state  $q_1$  and pop the stack.

If state  $q_1$  is reading nothing ( $\epsilon$ ) and top of stack is  $z_0$  then it switches to final state  $q_2$ .

### iii) Transition Diagram:

- We can use transition diagram to represent a PDA, where
- Any state is represented by a node in a diagram.
  - Any arrow coming from nowhere to a state indicates the start state and doubly circled states are accepting / final states.
  - The arc corresponds to transition of PDA as;
    - Arc labelled as  $a, x | \alpha$  means;  $S(q, a, x) = (p, \alpha)$  for arc from state  $p$  to  $q$ .

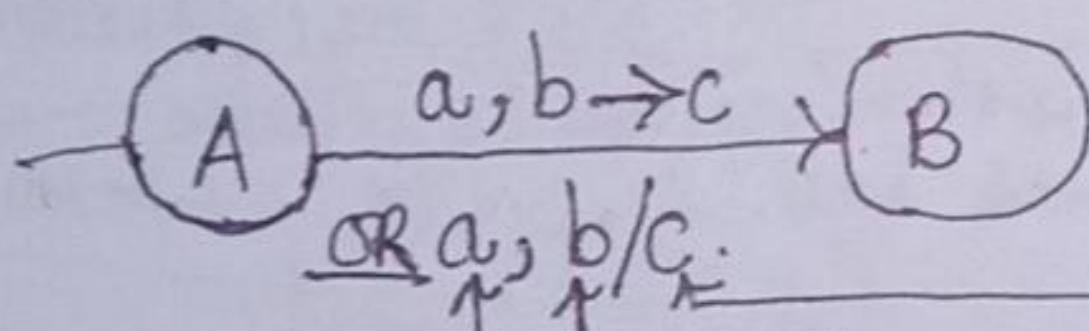
The PDA of  $a^n b^n$  can be represented using transition diagram as shown below;



where,  
p, q, r are state names  
we can name any

This explains  
Operation and more  
of PDA  
including examples

### Graphical Notation of Push down automata:



Input Symbol  
(It can also be  $\epsilon$ ),

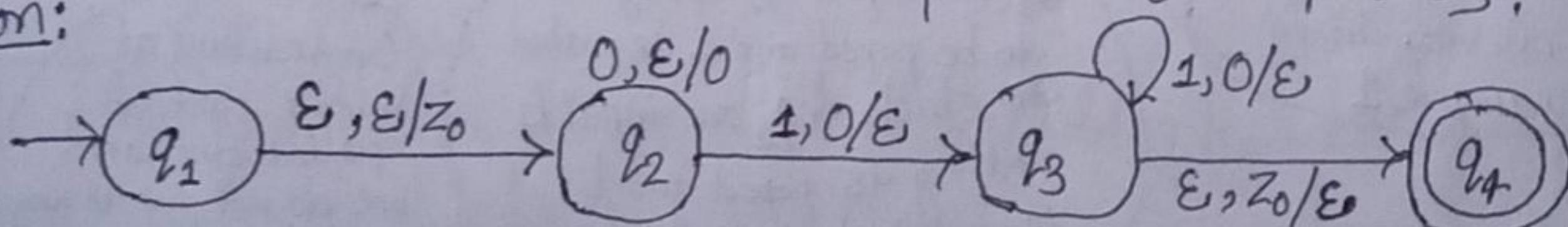
Symbol on top of  
the stack. This symbol  
is popped. (It can also  
be  $\epsilon$  which means stack  
is neither read nor  
popped)

This symbol is  
pushed onto the  
stack

(It can also be  $\epsilon$  which  
means nothing is pushed).

Example 1: Construct a PDA that accepts  $L = \{0^n 1^n \mid n \geq 0\}$ .

Solution:



Here, we started with any initial state  $q_1$ . In  $q_1$  we have transition  $\epsilon, \epsilon/z_0$  where  $\epsilon$  is input.  $\epsilon/z_0$  denotes initially stack contains  $\epsilon$  (empty) symbols and  $z_0$  is pushed symbol on stack which denotes it is the first (or bottom most) element of stack. Now we are on state  $q_2$  where we can get input 0, or 1 according to the question. Now on getting 0 as input we do not pop anything from stack (i.e.,  $\epsilon$ ), but we push the coming 0's to stack. Similarly on getting input 1 we go to next state  $q_3$ . Here  $0/\epsilon$  denotes 0 is the topmost element of stack which should be popped from the stack and  $\epsilon$  denotes

nothing to be pushed onto stack.

Now in state  $q_3$  on getting input 1, check if 0 is on the topmost position of stack, if yes then, pop it and nothing has to be pushed onto the stack. Now on getting input ε, we check if  $z_0$  is on the topmost position of stack, if yes then nothing has to be pushed onto the stack and we reached to final state  $q_4$  accepting the particular string.

Example 2: Construct a PDA that accepts Even Palindromes of the form  $L = \{WW^R \mid w = (a+b)^+\}$ .

Solution:-

We know that palindromes is a word or sequence that reads the same backwards as forwards. e.g. NOON, 123321, abba etc.

Let us understand the language before constructing PDA.

$$L = \{WW^R \mid w = (a+b)^+\}$$

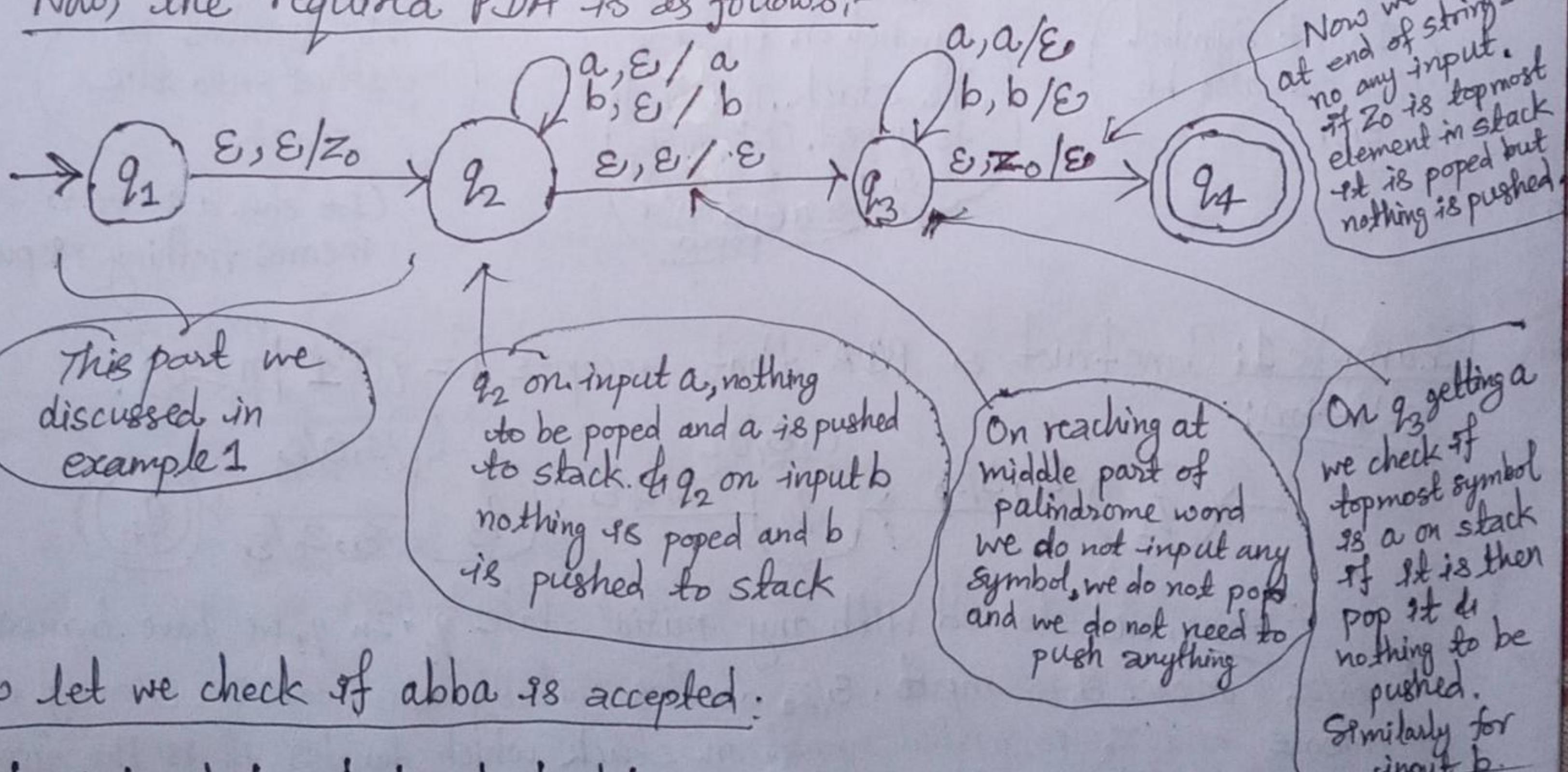
first w represents first half of palindrome for e.g. NO for NOON

second w with R represents reverse of first half. i.e., ON. (if we take NOON)

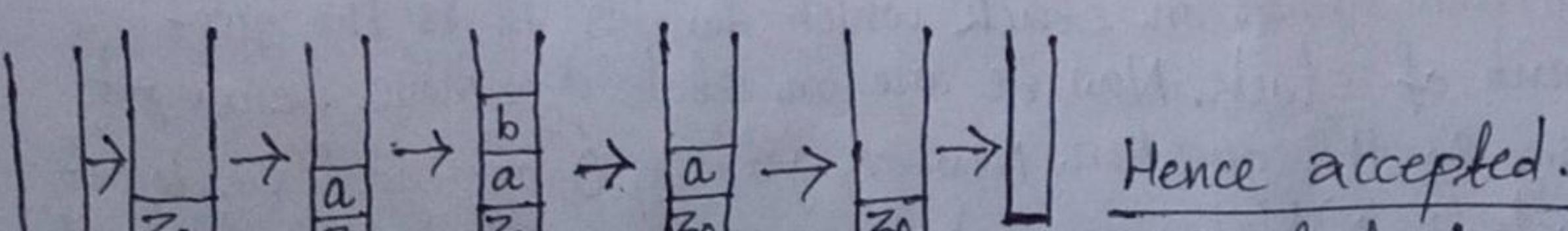
this +ve closure shows that there must be atleast one symbol be there, it cannot be empty & a, b are our inputs.

4 letters so even palindrome

Now, the required PDA is as follows:-



Now let we check if abba is accepted:



# If stack does not remain empty finally on tracing then that string is not accepted by PDA. (like string abab).

## ⊗. Instantaneous Description for PDA:

Any configuration of a PDA can be described by a triplet  $(q, w, r)$  where,

- $q$  is the state
- $w$  is the remaining input.
- $r$  is the stack contents.

In short we say it as ID

38.

$\in \rightarrow$  belongs to  
 $\epsilon \rightarrow$  epsilon.

Such a description by triple is called an instantaneous description of PDA (ID). Instantaneous description is helpful for describing changes in the state, input and stack of the PDA.

Let  $P = \{Q, \Sigma, \Gamma, S, q_0, z_0, F\}$  be a PDA. Then, we define a relation  $\vdash$ , "yields" as;  $(q, aw, zac) \vdash (p, w, bac)$  if  $S(q, a, z)$  contains  $(p, b)$  and may be a  $\epsilon$ . This move reflects the idea that, by consuming "a" from the input, and replacing  $z$  on the top of stack by  $b$ , we can go from state  $q$  to state  $p$ .

Example: For the PDA described earlier accepting language  $WW^R$ , [see example 2] the accepting sequence of ID's for string 1001 can be shown as;

$$\vdash (q_0, 1001, z_0)$$

Here 1001 is input string

$$\vdash (q_1, 001, 1z_0)$$

Here POP if there is mismatch between Input &  $z_0$  content i.e 0 and 1 mismatch here

$$\vdash (q_1, 01, z_0)$$

$$\vdash (q_1, 1, 0z_0)$$

$$\vdash (q_1, \epsilon, z_0)$$

$$\vdash (q_2, \epsilon, \epsilon) \text{ Accepted.}$$

Therefore  $(q_0, 1001, z_0) \vdash^* (q_2, \epsilon, \epsilon)$ .

$\vdash$  we read it as tungsten.  
 It connects the two or more symbol denotes moves can be more than one.

## ⊗. Language of a PDA:

We can define acceptance of any string by a PDA in two ways;

### i) Acceptance by final state:

Given a PDA,  $P$  the language accepted by final state,  $L(P)$  is;

$$\{w \mid (q, w, z_0) \vdash^* (p, \epsilon, r)\} \text{ where } p \in F \text{ and } r \in \Gamma^*$$

### ii) Acceptance by empty stack:

Given a PDA,  $P$  the language accepted by empty stack,  $L(P)$  is

$$\{w \mid (q, w, z_0) \vdash^* (p, \epsilon, \epsilon)\} \text{ where } p \in Q$$

i.e., finally anything can be in stack no matter

i.e., finally stack should be empty

## ⊗. Deterministic Pushdown Automata (DPDA) :-

A PDA is said to be deterministic if for every input alphabet 'a' (may be  $\epsilon$ ) and top of stack symbol, PDA has either unique transition (i.e, moves to only one state) or its transition is not defined.

Formally A pushdown automata  $P = (Q, \Sigma, \Gamma, S, q_0, z_0, F)$  is deterministic pushdown automata if there is no configuration for which P has a choice of more than one moves. i.e, P is deterministic if following two conditions are satisfied;

- 1) For any  $q \in Q, a \in \Sigma$ , if  $S(q, a, x) \neq \emptyset$  for some  $a \in \Sigma$  then  $S(q, \epsilon, x) = \emptyset$   
i.e, if  $S(q, a, x)$  is non-empty for some a, then  $S(q, \epsilon, x)$  must be empty.
- 2) For any  $q \in Q, a \in \Sigma \cup \{\epsilon\}$  and  $x \in \Gamma$ ,  $S(q, a, x)$  has at most one element.

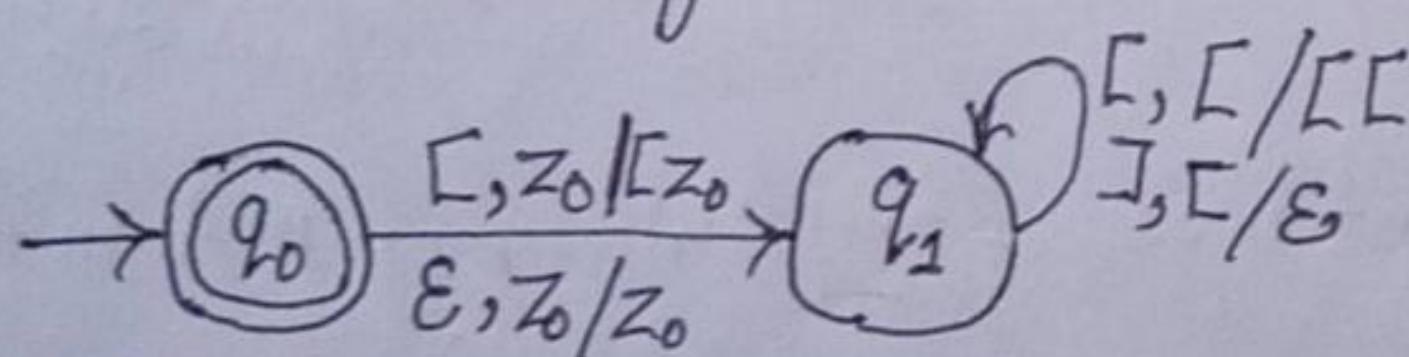
Example 1: Construct DPDA that accepts balanced parentheses i.e, equal no. of opening and closing braces. eg: [[ ]], [ ] [ ] etc.

Solution:

The transition table for the given DPDA is as follows;

Move number	state	Input	Stack Symbol	Move(s)
1.	$q_0$	[	$z_0$	$(q_1, [z_0])$
2.	$q_1$	[	[	$(q_1, [ ])$
3.	$q_1$	]	[	$(q_1, \epsilon)$
4.	$q_1$	$\epsilon$	$z_0$	$(q_0, z_0)$

The transition diagram for this is shown below;



Example 2: Construct a DPDA accepting language  $L = \{wCw^R | w \in (0+1)^*\}$ .

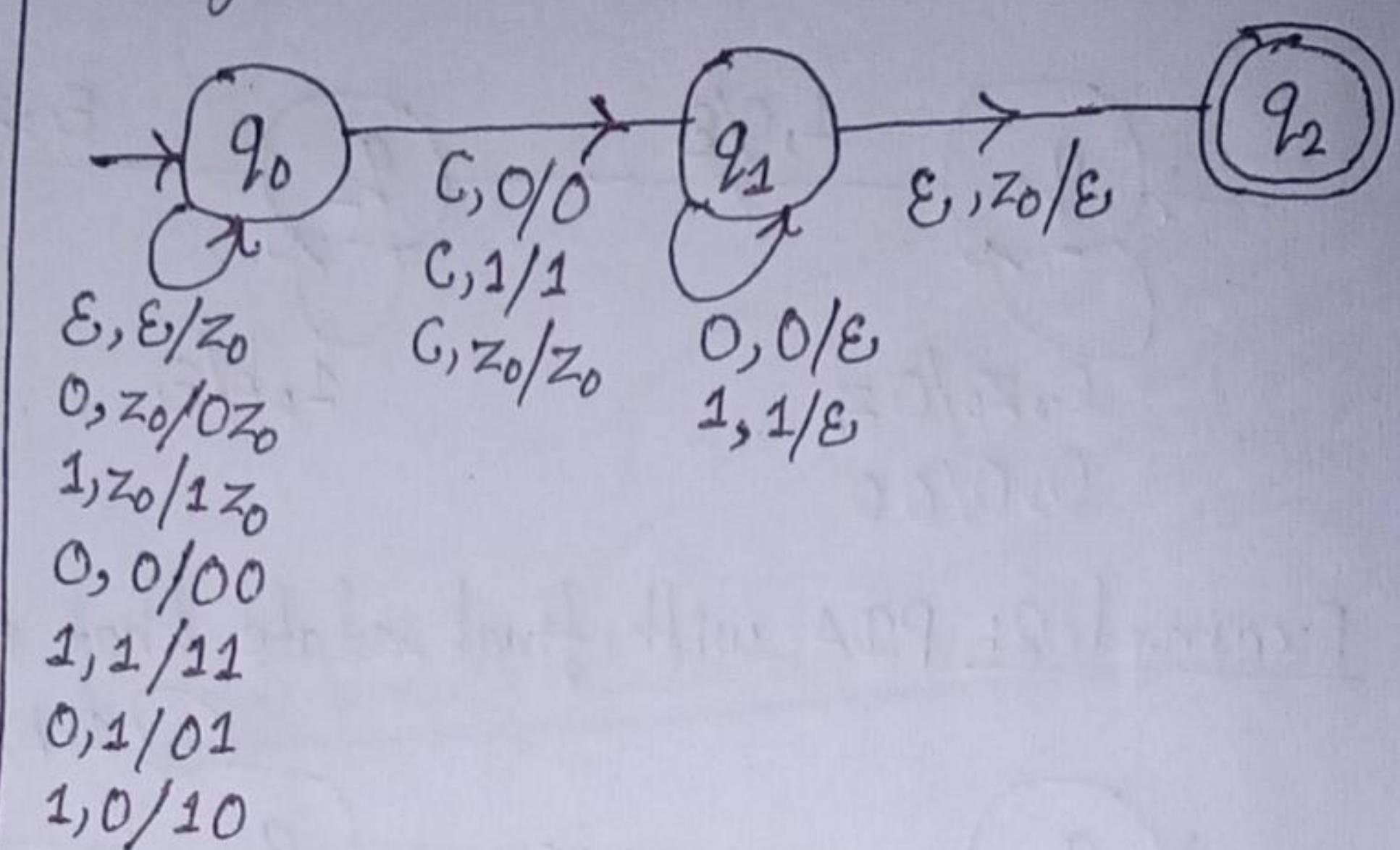
Solution:

$P = (Q, \Sigma, \Gamma, S, q_0, z_0, F)$  where S is defined as;

1.  $S(q_0, \epsilon, \epsilon) = (q_0, z_0)$
2.  $S(q_0, 0, z_0) = (q_0, 0z_0)$
3.  $S(q_0, 1, z_0) = (q_0, 1z_0)$
4.  $S(q_0, 0, 0) = (q_0, 00)$

5.  $S(q_0, 1, 1) = (q_0, 11)$   
 6.  $S(q_0, 0, 1) = (q_0, 01)$   
 7.  $S(q_0, 1, 0) = (q_0, 10)$   
 8.  $S(q_0, 0, 0) = (q_1, 0)$   
 9.  $S(q_0, C, 1) = (q_1, 1)$   
 10.  $S(q_0, C, Z_0) = (q_1, Z_0)$   
 11.  $S(q_1, 0, 0) = (q_1, \epsilon)$   
 12.  $S(q_1, 1, 1) = (q_1, \epsilon)$   
 13.  $S(q_1, \epsilon, Z_0) = (q_2, \epsilon)$ .

Now, the general notation or transition diagram is as follows:



### ⊗ Non Deterministic PDA (NPDA):

A nondeterministic pushdown automaton (npda) is basically an nfa with a stack added to it. A nondeterministic pushdown automaton is a 7-tuple  $(Q, \Sigma, \Gamma, S, q_0, z_0, F)$ . It is an NFA which is a 5-tuple, and added two things to it:

$\Gamma$  is a finite set of symbols called the stack alphabet and  $z_0 \in \Gamma$  is the stack start symbol. Other 5 tuples are as usual we can describe them.

We also need to modify  $S$ , the transition function, so that it manipulates the stack. The transition function for an npda has the form:

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

### ⊗ Construction of PDA by Final State:

A PDA accepts a string when after reading the entire string in the final state. From the starting state we can make moves that end up in a final state with any stack values. The stack values are irrelevant as long as we end up in a final state.

For a PDA  $(Q, \Sigma, \Gamma, S, q_0, z_0, F)$  the language accepted by the set of final states  $F$  is;

$$L(PDA) = \{ w \mid (q_0, w, z_0) \xrightarrow{*} (q, \epsilon, \alpha) \} \text{ where } q \in F.$$

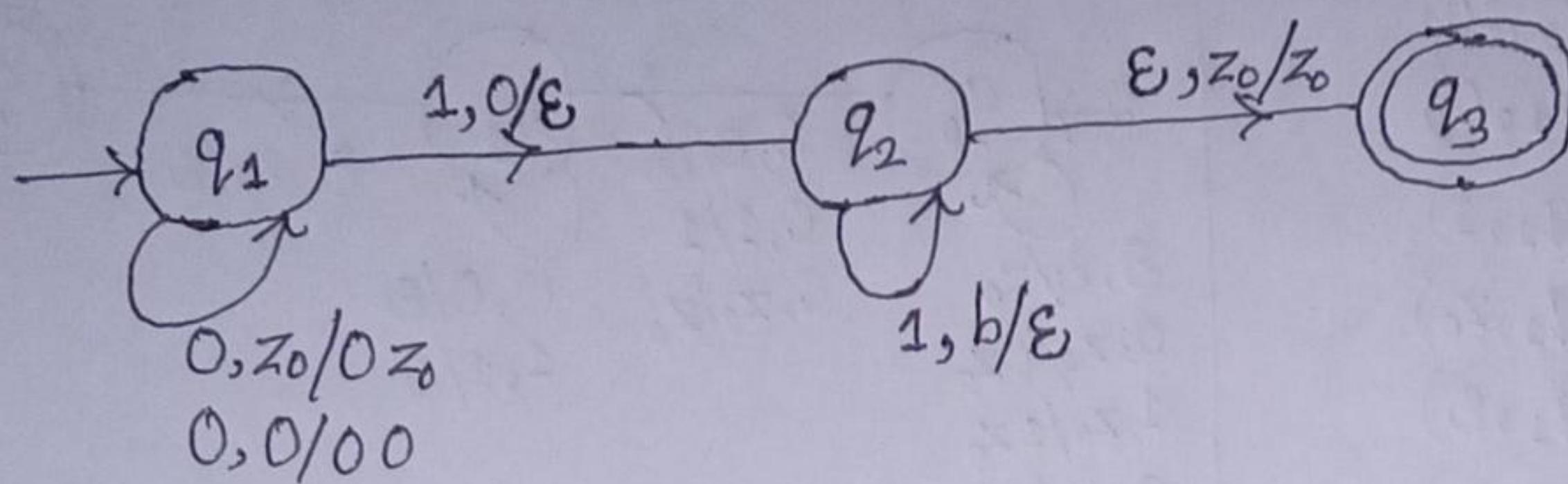
such that

ID consisting of triplets

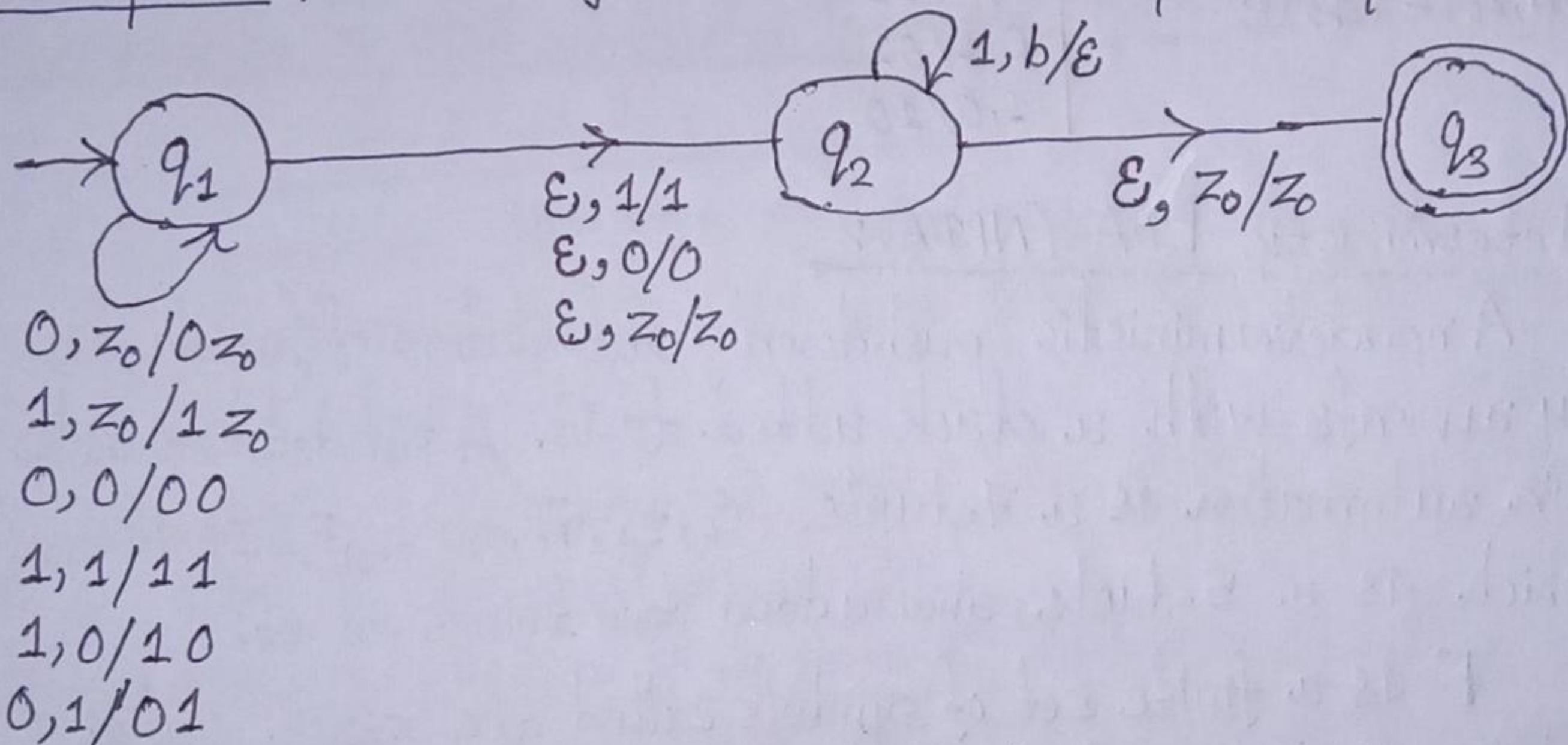
string notation connecting ID's

another ID consisting triplets

Example 1: PDA with final state with equal number of zeros (0's) followed by equal numbers ones (1's).



Example 2: PDA with final state that accept the palindrome.

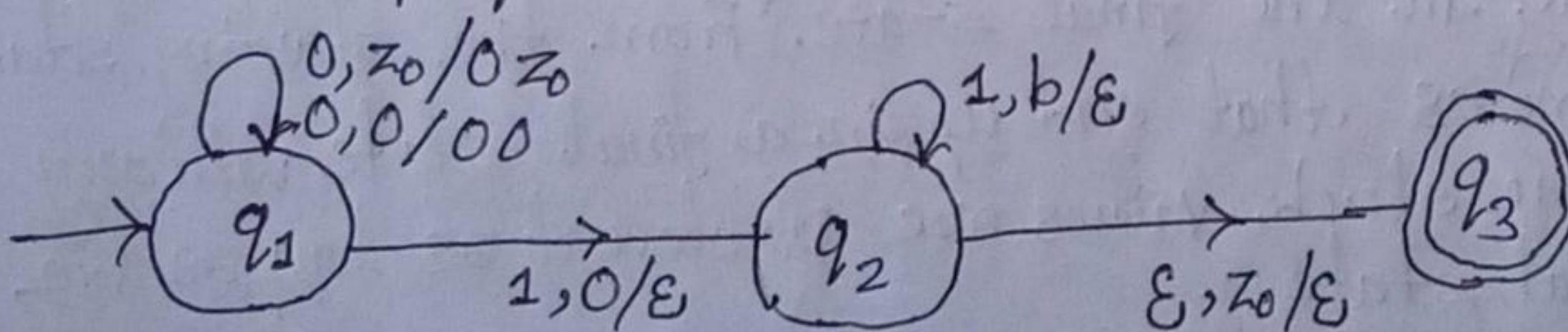


#### ④ Construction of PDA by Empty Stack:-

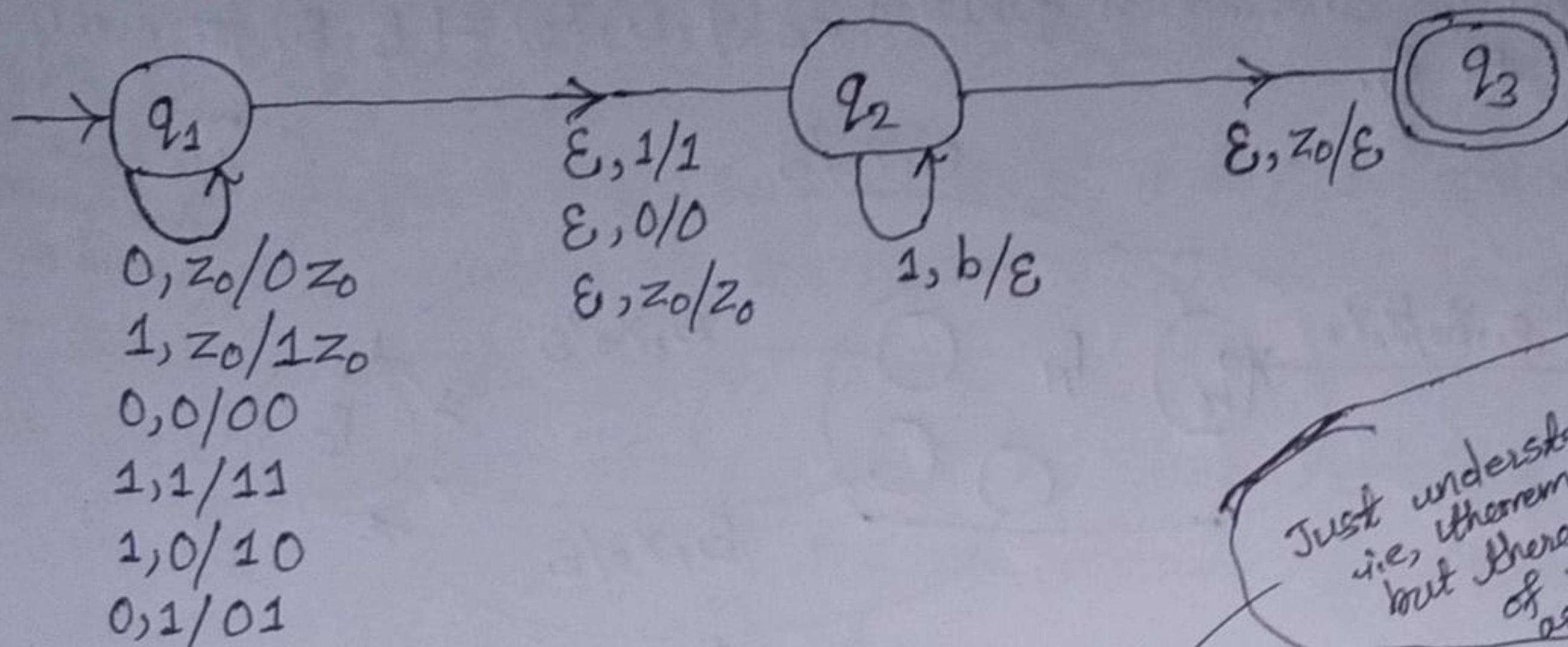
A PDA accepts a string when, after reading the entire string, when PDA has emptied its stack. For a PDA  $(Q, \Sigma, \Gamma, S, q_0, z_0, F)$  the language accepted by empty stack is;

$$L(\text{PDA}) = \{w \mid (q_0, w, z_0) \xrightarrow{*} (q, \epsilon, \epsilon) \} \text{ where, } q \in Q.$$

Example 1: PDA with empty stack with equal number of zeros (0's) followed by equal numbers ones (1's).



Example 2: PDA with empty stack that accept the palindrome.



Just understand theories  
i.e., theorem may be asked  
but there is less chance  
of numerical being  
asked for this and  
vice-versa

### \* Conversion of PDA accepting by Empty stack to accepting by final state:

Let us consider the PDA  $P_N$  that accepts a language  $L$  by empty stack and  $P_F$  is its equivalent PDA that accepts  $L$  by final state.

Theorem: If  $L = L(P_N)$  for some PDA  $P_N = (\mathcal{Q}, \Sigma, \Gamma, S_N, q_0, z_0)$ , then there is a PDA  $P_F$  such that  $L = L(P_F)$ .

Proof: The construction of  $P_F$  from  $P_N$  is done as follows:

1. Introduce the new symbol  $x_0$  ( $x_0 \notin \Gamma$ ) and place this symbol into the bottom of the stack  $P_F$ .

2. Introduce a new start state  $p_0$ , whose sole function is to push  $z_0$ , the start symbol of  $P_N$ , onto the top of stack and enter state  $q_0$  (the start state of  $P_N$ ).

3. Copy the other states and transition functions of  $P_N$  in  $P_F$ .

4. Finally add another new state  $p_f$ , which is the accepting state of  $P_F$ ; this PDA transfers to state  $p_f$  whenever it discovers that  $P_N$  would have emptied its stack.

Thus the new constructed  $P_F$  has the components:

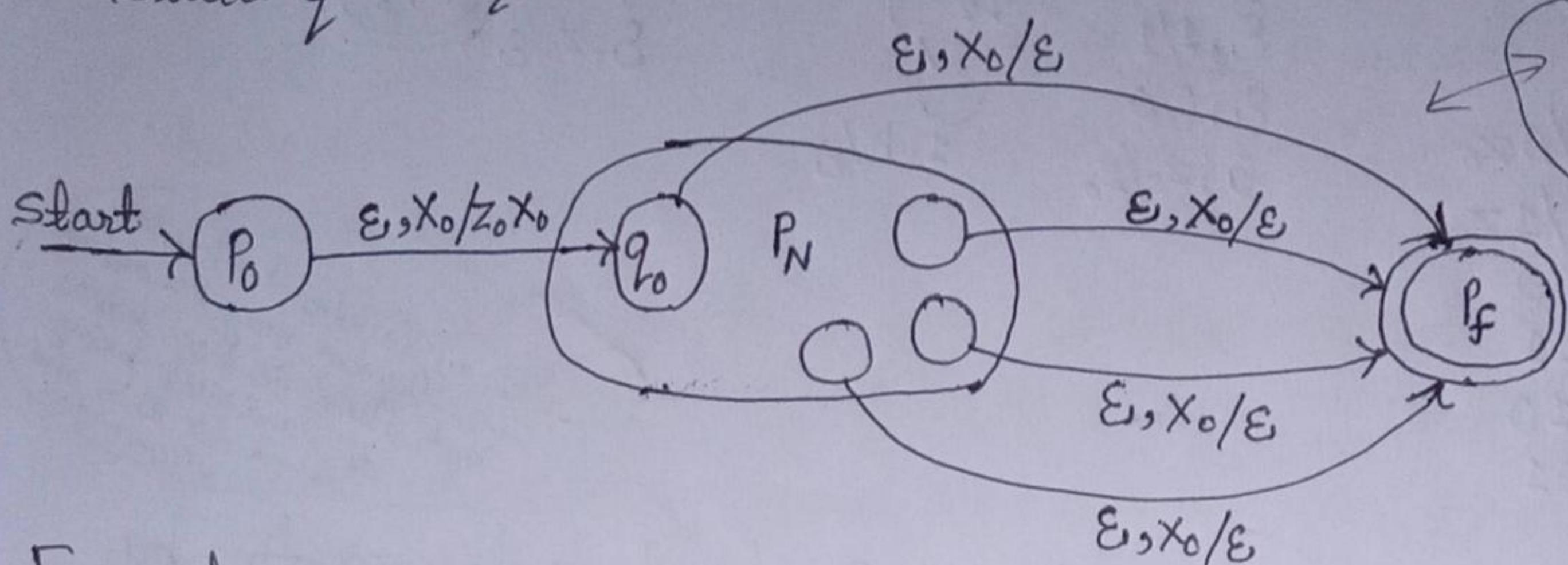
$$P_F = (\mathcal{Q} \cup \{p_0, p_f\}, \Sigma, \Gamma \cup \{x_0\}, S_F, p_0, x_0, \{p_f\})$$

where  $S_F$  is defined by:

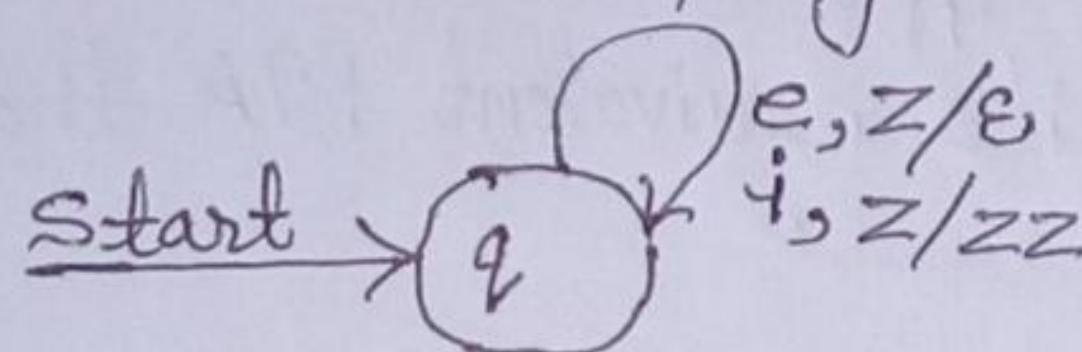
$$1. S_F(p_0, \epsilon, x_0) = (q_0, z_0 x_0)$$

2. For all states  $q$  in  $\mathcal{Q}$ , inputs  $a$  in  $\Sigma$  or  $a = \epsilon$ , and stack symbols  $Y$  in  $\Gamma$ ,  $S_F(q, a, Y)$  contains all the pairs in  $S_N(q, a, Y)$ . i.e., Copy all the transition functions of  $P_N$ .

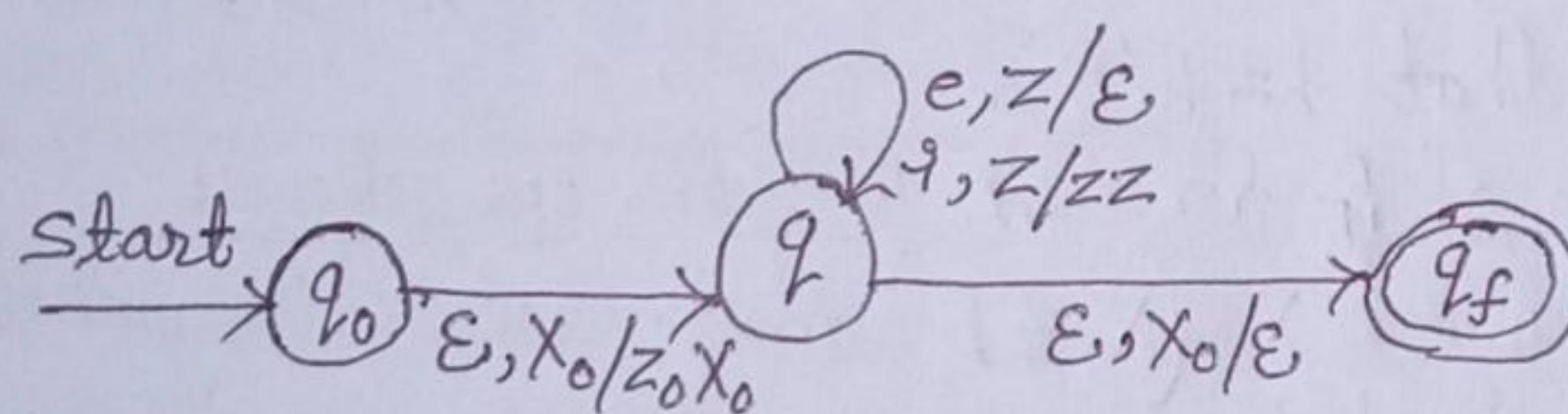
3. Add another transition function  $S_F(q, \epsilon, X_0) = (P_f, \epsilon)$  for every state  $q$  in  $Q$ .



Example: Convert the following PDA that accepts the string using empty stack into its equivalent PDA that accepts the string by entering into the accepting state.



Solution:



\*. Conversion of PDA accepting by final state to accepting by empty stack:

Theorem: Let  $L$  be  $L(P_f)$  for some PDA  $P_f = (Q, \Sigma, \Gamma, S_f, q_0, z_0, F)$ . Then there is a PDA  $P_N$  such that  $L = N(P_N)$ .

Proof: The construction of  $P_N$  from  $P_f$  is done as follows:

1. Introduce a new symbol  $X_0$  ( $X_0 \notin \Gamma$ ) and place this symbol into the bottom of the stack of  $P_N$ .
2. Introduce a new start state  $P_0$ , whose sole function is to push  $Z_0$  (the start symbol of  $P_f$ ), onto the top of the stack and enter state  $q_0$  (the start state of  $P_f$ ).
3. For each accepting state of  $P_f$ , add a transition on  $\epsilon$  to a new state  $p$ .

Thus the new constructed  $P_N$  has the components:

$$P_N = (Q \cup \{P_0, p\}, \Sigma, \Gamma \cup \{X_0\}, S_N, P_0, X_0)$$

Steps 1 & 2 same as before  
only  $P_0$  and  $F$  names are interchanged.

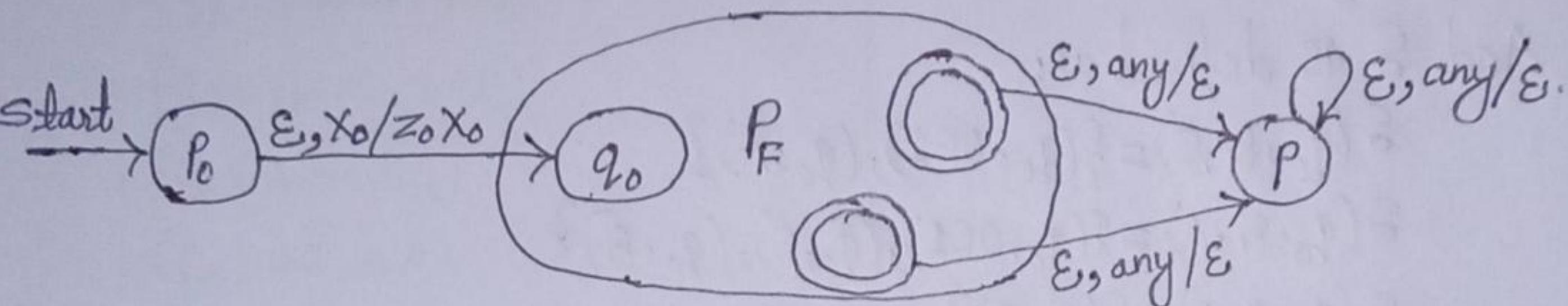
where,  $S_N$  is defined by;

$$1. S_N(p_0, \epsilon, x_0) = (q_0, z_0 x_0).$$

2. For all states  $q$  in  $Q$ , inputs  $a \in \Sigma$  or  $a = \epsilon$ , and stack symbols  $Y \in \Gamma$ ,  $S_N(q, a, Y)$  contains all the pairs in  $S_F(q, a, Y)$ . i.e, Copy all the transition functions of  $P_F$ .

3. For all accepting states  $q$  in  $F$  and stack symbols  $Y \in \Gamma$  or  $Y = x_0$ ,  $S_N(q, \epsilon, Y) = (p, \epsilon)$ . By this rule, whenever  $p_F$  accepts,  $p_N$  can start emptying its stack without consuming any more input.

4. For all stack symbols  $Y \in \Gamma$  or  $Y = x_0$ ,  $S_N(q, \epsilon, Y) = (p, \epsilon)$ .



### \* Conversion of CFG<sub>1</sub> to PDA:

Given a CFG<sub>1</sub>,  $G_1 = (V, T, P$  and  $S)$ , we can construct a push down automata,  $M$  which accepts the language generated by the grammar  $G_1$ . i.e,  $L(M) = L(G_1)$ .

The machine can be defined as;

$$M = (\{q\}, T, V \cup T, S, q, S, \emptyset)$$

where,  $Q = \{q\}$  is only the state in the PDA.

$$\Sigma = T$$

$\Gamma = V \cup T$  (i.e, PDA uses terminals and variables of  $G_1$ )

$z_0 = S$  (i.e, initial stack symbol as stack symbols).

$F = \emptyset$  (initial stack symbol is start symbol in grammar).

And  $S$  is defined as;

i).  $S(q, \epsilon, A) = \{(q, \alpha) / A \rightarrow \alpha \text{ is a production } P \text{ of } G_1\}$ .

ii).  $S(q, a, a) = \{(q, \epsilon) / a \in T\}$ .

Example 1: Convert the grammar defined by following production into PDA;

$$S \rightarrow 0S1 | A$$

$$A \rightarrow 1S0 | S | \epsilon$$

Solution:

Let  $G_1 = (V, T, P \text{ and } S)$  defined by following productions;

$$S \rightarrow 0S1 | A$$

$$A \rightarrow 1S0 | S | \epsilon$$

PDA equivalent to this grammar is as;

$$M = (\{q_0\}, \{0, 1\}, \{0, 1, S, A\}, S, q_0, S, \emptyset)$$

where,

$$Q = \{q_0\}$$

$$\Sigma = \{0, 1\}$$

$$\Gamma = \{0, 1, S, A\}$$

$$z_0 = S$$

$$F = \emptyset$$

And  $S$  is defined as;

$$S(q_0, \epsilon, S) = \{(q_0, 0S1), (q_0, A)\}$$

$$S(q_0, \epsilon, A) = \{(q_0, 0S1), (q_0, S), (q_0, \epsilon)\}$$

$$S(q_0, 0, 0) = \{(q_0, \epsilon)\}$$

$$S(q_0, 1, 1) = \{(q_0, \epsilon)\}$$

Example 2: Construct a PDA equivalent to following grammar defined by;

$$S \rightarrow aAA$$

$$A \rightarrow aS | bS | a. \quad \text{Also trace acceptance of } aaabaaaaaa.$$

Solution:

Let  $G_1 = (V, T, P \text{ and } S)$  be the grammar defined by the production;

$$S \rightarrow aAA$$

$$A \rightarrow aS | bS | a$$

Now, PDA equivalent to this grammar is as;

$$M = (\{q_0\}, \{a, b\}, \{a, b, S, A\}, S, q_0, S, \emptyset)$$

where  $S$  is defined as;

$$S(q_0, \epsilon, S) = \{q_0, aAA\}$$

$$S(q_0, \epsilon, A) = \{(q_0, aS), (q_0, bS), (q_0, a)\}$$

$$S(q_0, a, a) = \{q_0, \epsilon\}$$

$$S(q_0, b, b) = \{q_0, \epsilon\}$$

Now, we trace the acceptance of  $aaabaaaaaa$ .

$$(q_0, aaabaaaaaa, S)$$

$$T(q_0, aaabaaaaaa, aA)$$

$$T(q_0, aabaaaaaa, AA)$$

$\vdash(q_0, aabaaaaa, asA)$   
 $\vdash(q_0, abaaaaa, SA)$   
 $\vdash(q_0, abaaaaa, aAAA)$   
 $\vdash(q_0, baaaaa, AAA)$   
 $\vdash(q_0, baaaaa, bSAA)$   
 $\vdash(q_0, aaaaa, SAA)$   
 $\vdash(q_0, aaaaa, aAAAA)$   
 $\vdash(q_0, aaaa, AAAA)$   
 $\vdash(q_0, aaaa, aAAA)$   
 $\vdash(q_0, aaa, AAA)$   
 $\vdash(q_0, aaa, aAA)$   
 $\vdash(q_0, aa, AA)$   
 $\vdash(q_0, aa, aA)$   
 $\vdash(q_0, a, A)$   
 $\vdash(q_0, a)$

### In CFG<sub>1</sub>

$S \rightarrow aAA$   
 $\rightarrow aASA$   
 $\rightarrow aaaAAA$   
 $\rightarrow aaabsSAA$   
 $\rightarrow aaabaAAAA$   
 $\rightarrow aaabaaAAA$   
 $\rightarrow aaabaaaAA$   
 $\rightarrow aaabaaaaA$   
 $\rightarrow aaabaaaaa$

left most derivation that we read in regular expression chapter.  
 In this we expand left most non-terminal in our production and we keep expanding until we reach required string.

for eg.  
 A is replaced by aS in 2nd line.

### ⊗. Conversion of PDA into its equivalent CFG<sub>1</sub>:

Given a PDA  $M = (\Omega, \Sigma, \Gamma, S, q_0, z_0, F)$ ;  $F = \emptyset$ , we can obtain equivalent CFG<sub>1</sub>,  $G_1 = (V, T, P \text{ and } S)$  which generates the same language as accepted by the PDA M.

The set of variables in the grammar consist of;  
 → The special symbol S, which is start symbol.  
 → All the symbols of the form  $[pxq]$ ;  $p, q \in \Omega$  and  $x \in \Gamma$ .  
 i.e.,  $V = \{S\} \cup \{[pxq]\}$

The terminal in the grammar  $T = \Sigma$ .

The production of  $G_1$  is as follows;

→ For all states  $q \in \Omega$ ,  $S \rightarrow [q_0, z_0, q]$  is a production of  $G_1$ .

→ For any states  $q, r \in \Omega$ ,  $x \in \Gamma$  and  $a \in \Sigma \cup \{\epsilon\}$ ,

If  $S(q, a, x) = (p, \epsilon)$  then  $[pxq] \rightarrow a$

→ For any states  $q, r \in \Omega$ ,  $x \in \Gamma$  and  $a \in \Sigma \cup \{\epsilon\}$ ,

If  $S(q, a, x) = (r, Y_1 Y_2 \dots Y_k)$ ; where  $Y_1, Y_2, \dots, Y_k \in \Gamma$  and  $k \geq 0$ .

Then for all lists of states  $r_1, r_2, \dots, r_k$ ,  $G_1$  has the production

$[pxq] \rightarrow a [r_1 r_1] [r_1 r_2 r_2] \dots [r_{k-1} r_k r_k]$ .

This production says that one way to pop  $X$  and go from stack  $q$  to state  $r_k$  is to read "a" (which may be  $\epsilon$ ), then use some input to pop  $Y_1$  off the stack while going from state  $r$  to  $r_1$ , then read some more input that pops  $Y_2$  off the stack and goes from  $r_1$  to  $r_2$  and so on....

Example 1: Convert the PDA given below that recognizes a language

$L = \{a^n b^n \mid n \geq 0\}$  defined as;

$$1. S(q_0, a, z_0) = (q_1, az_0)$$

$$2. S(q_1, a, a) = (q_1, aa)$$

$$3. S(q_1, b, a) = (q_1, \epsilon)$$

4.  $S(q_1, \epsilon, z_0) = (q_1, \epsilon)$ . Also show the acceptance of aaabbb.

Solution:

Let  $G_1 = (V, T, P \text{ and } S)$  be the equivalent CFG for the given PDA where,

$$V = \{S\} \cup \{[pxq] \mid p, q \in Q, x \in \Sigma\}$$

$S$  is the start state

$$\Sigma = \Sigma$$

And  $P$  is defined by following production;

$$1. S \rightarrow [q_0 z_0 q_0] \mid [q_0 z_0 q_1]$$

i.e.,  $S \rightarrow [q_0 z_0 r_2]$ ; for  $r_2 \in \{q_0, q_1\}$ .

2. From the fact that  $S(q_0, a, z_0)$  contains  $(q_1, az_0)$ , we get production  $[q_0 z_0 r_2] \rightarrow a [q_1 a r_1] [r_1 z_0 r_2]$ ; for  $r_1 \in \{q_0, q_1\}$ .

3. From the fact that  $S(q_1, a, a)$  contains  $(q_1 a a)$ , we get production  $[q_1 a r_2] \rightarrow a [q_1 a r_1] [r_1 a r_2]$  for  $r_1 \in \{q_0, q_1\}$ .

4. From the fact that  $S(q_1, b, a)$  contains  $(q_1, \epsilon)$ , we get  $[q_1 a q_1] \rightarrow b$

5. From the fact that  $S(q_1, \epsilon, z_0)$  contains  $(q_1, \epsilon)$ , we get  $[q_1 z_0 q_1] \rightarrow \epsilon$

Now the acceptance of aaabbb can be shown as;

$$S \rightarrow [q_0 z_0 r_2]$$

$$\rightarrow a [q_1 a r_1] [r_1 z_0 r_2]$$

$$\rightarrow aa [q_1 a r_1] [r_1 a r_2] [r_1 z_0 r_2]$$

$\rightarrow \text{aaa} [q_1 x r_1] [r_1 x r_2] [r_1 x r_2] [r_1 z_0 r_2]$   
 $\rightarrow \text{aaab} [r_1 x r_2] [r_1 x r_2] [r_1 z_0 r_2]$   
 $\rightarrow \text{aaabb} [r_1 x r_2] [r_1 z_0 r_2]$   
 $\rightarrow \text{aaabbb} [r_1 z_0 r_2]$   
 $\rightarrow \text{aaabbb } \epsilon = \text{aaabbb.}$

Example 2: Convert the PDA  $P = (\{P, q\}, \{0, 1\}, \{x, z_0\}, S, q, z_0)$  to a CFG if  $S$  is given by;

$$\begin{aligned}S(q, 1, z_0) &= (q, x z_0) \\S(q, 1, x) &= (q, x x) \\S(q, 0, x) &= (q, x) \\S(q, \epsilon, z_0) &= (q, \epsilon) \\S(q, 0, z_0) &= (q, z_0).\end{aligned}$$

Solution:

The equivalent grammar can be written as;  
 $G_1 = (V, \Sigma, P, S)$  where  $P$  consists of productions as;

$$S \rightarrow [q z_0 r_2]; r_2 \in \{P, q\}.$$

$$[q z_0 r_2] \rightarrow 1[q x r_1] [r_1 z_0 r_2]; \text{ for } r_1 \in \{P, q\}.$$

$$[q x r_2] \rightarrow 1[q x r_1] [r_1 x r_2]; \text{ for } r_1 \in \{P, q\}.$$

$$[q x r_2] \rightarrow 0[P x r_2]; \text{ for } r_2 \in \{P, q\}.$$

$$[q x q] \rightarrow \epsilon$$

$$[P x P] \rightarrow 1$$

$$[P z_0 r_2] \rightarrow 0[q z_0 r_2]; \text{ for } r_2 \in \{P, q\}.$$