UNIT- IV

PUSHDOWN AUTOMATA

6

PUSH DOWN AUTOMATA

After going through this chapter, you should be able to understand :

- Push down automata
- Acceptance by final state and by empty stack
- Equivalence of CFL and PDA
- Interconversion
- Introduction to DCFL and DPDA

6.1 INTRODUCTION

A PDA is an enhancement of finite automata (FA). Finite automata with a stack memory can be viewed as pushdown automata. Addition of stack memory enhances the capability of Pushdown automata as compared to finite automata. The stack memory is potentially infinite and it is a data structure. Its operation is based on last - in - first - out (LIFO). It means, the last object pushed on the stack is popped first for operation. We assume a stack is long enough and linearly arranged. We add or remove objects at the left end.

6.1.1 Model of Pushdown Automata (PDA)

A model of pushdown automata is shown in below figure. It consists of a finite tape, a reading head, which reads from the tape, a stack memory operating in LIFO fashion.

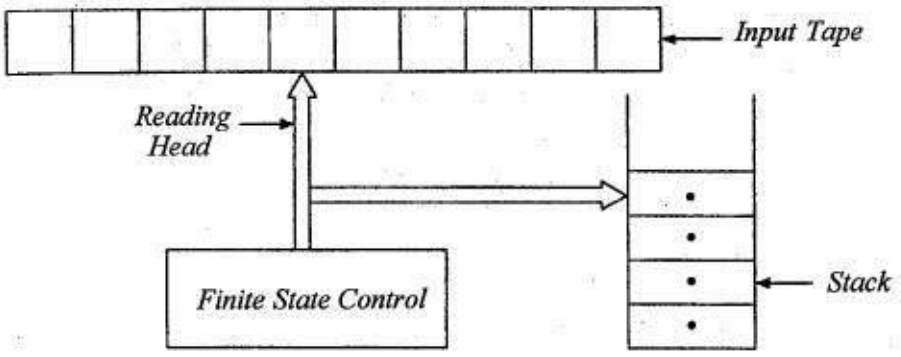


FIGURE : Model of Pushdown Automata

There are two alphabets ; one for input tape and another for stack. The stack alphabet is denoted by Γ and input alphabet is denoted by Σ. PDA reads from both the alphabets ; one symbol from the input and one symbol from the stack.

6.1.2 Mathematical Description of PDA

A pushdown automata is described by 7 - tuple $(Q,\Sigma,\Gamma,\delta,q_0,Z_0,F)$, where

1. Q is finite and nonempty set of states,
2. Σ is input alphabet,
3. Γ is finite and nonempty set of pushdown symbols,
4. δ is the transition function which maps
   From $Q \times (\Sigma \cup \{\in\}) \times \Gamma$ to (**finite subset** of) $Q \times \Gamma^*$,
5. $q_0 \in Q$, is the starting state,
6. $Z_0 \in \Gamma$, is the starting (top most or initial) stack symbol, and
7. $F \subseteq Q$, is the set of final states.

6.1.3 Moves of PDA

The move of PDA means that what are the options to proceed further after reading inputs in some state and writing some string on the stack. As we have discussed earlier that PDA is nondeterministic device having some finite number of choices of moves in each situation.

The **move** will be of two types :

1. In the first type of move, an input symbol is read from the tape, it means, the head is advanced and depending upon the topmost symbol on the stack and present state, PDA has number of choices to proceed further.
2. In the second type of move, the input symbol is not read from the tape, it means, head is not advanced and the topmost symbol of stack is used. The topmost of stack is modified without reading the input symbol. It is also known as an $\in$ - move.

Mathematically first type of move is defined as follows.

$\delta(q,a,Z) = \{(p_1,\alpha_1),(p_2,\alpha_2),...(p_n,\alpha_n)\}$, where for $1 \le i \le n$, q, p_i are states in Q, $a \in \Sigma$, $Z \in \Gamma$, *and* $\alpha_i \in \Gamma^*$.

PDA reads an input symbol a and one stack symbol Z in present state q and for any value(s) of i, enters state p_i, replaces stack symbol Z by string $\alpha_i \in \Gamma^*$, and head is advanced one cell on the tape. Now, the leftmost symbol of string α_i is assumed as the topmost symbol on the stack.

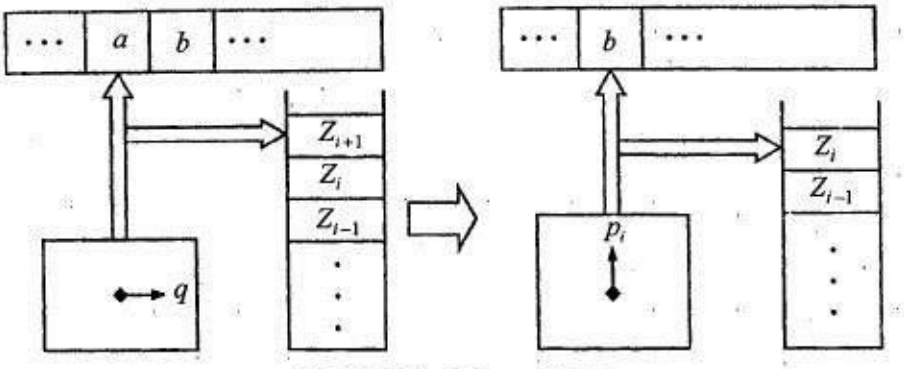
Mathematically second type of move is defined as follows.

$\delta(q,\in,Z) = \{(p_1,\alpha_1),(p_2,\alpha_2),....(p_n,\alpha_n)\}$, where for $1 \le i \le n$, q, p_i are states in Q, $a \in \Sigma$, $Z \in \Gamma$, *and* $\alpha_i \in \Gamma^*$.

PDA does not read input symbol but it reads stack symbol Z in present state q and for any value(s) of i, enters state p_i, replaces stack symbol Z by string $\alpha_i \in \Gamma^*$, and head is not advanced on the tape. Now, the leftmost symbol of string α_i is assumed as the topmost symbol on the stack.

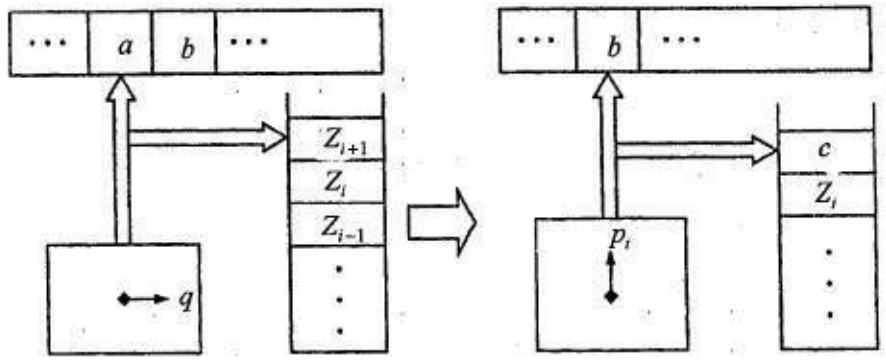
The **string** α_i be any one of the following :

1. $\alpha_i = \in$ in this case the topmost stack symbol Z_{i+1} is erased and second topmost symbol becomes the topmost symbol in the next move. It is shown in figure (a).



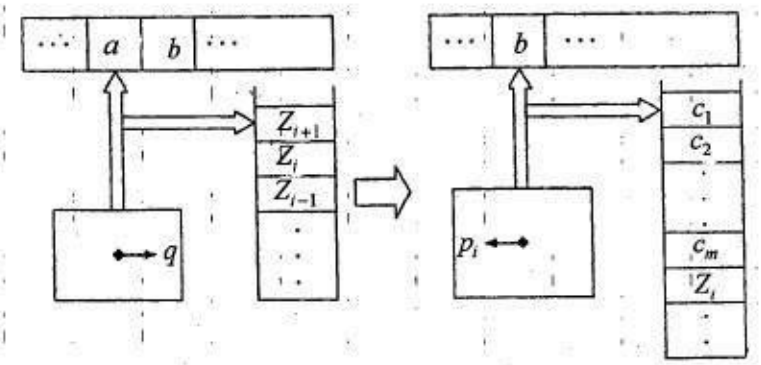
FIGURE(a): Move of PDA

2. $\alpha_i = c, c \in \Gamma$, in this case the topmost stack symbol Z_{i+1} is replaced by symbol c. It is shown in figure(b)



FIGURE(b): Move of PDA

3. $\alpha_i = c_1 c_2 \ldots c_m$, in this case the topmost stack symbol Z_{i+1} is replaced by string $c_1 c_2 \ldots c_m$. It is shown in figure(c).



FIGURE(c): Move of PDA

6.1.4 Instantaneous Description (ID) of PDA

Let PDA $M = (Q,\Sigma,\Gamma,\delta,q_0,Z_0,F)$, then its configuration at a given instant can be defined by instantaneous description (ID). An ID includes state, remaining input string, and remaining stack string (symbols). So, an ID is (q,x,α), where $q \in Q, x \in \Sigma^*, \alpha \in \Gamma^*$.

The relation between two consecutive IDs is represented by the sign $\vdash$.

We say $(q,ax,Z\beta) \vdash_M (p,x,\alpha\beta)$ if $\delta(q,a,Z)$ contains (p,α), where $Z,\beta,\alpha \in \Gamma^*$, a may be null or $a \in \Sigma$, $p,q \in Q$ for M

The reflexive and transitive closure of the relation $\vdash_M$ is denoted by $\vdash_M^*$

Properties :

1. If $(q,x,\alpha) \vdash_M^* (p,\in,\alpha)$, where $\alpha \in \Gamma^*, x \in \Sigma^*$, and $p,q \in Q$, then for all $y \in \Sigma^*$. $(q,xy,\alpha) \vdash_M^* (p,y,\alpha)$,
2. If $(q,xy,\alpha) \vdash_M^* (p,y,\alpha)$, where $\alpha \in \Gamma^*, x,y \in \Sigma^*$, and $p,q \in Q$, then $(q,x,\alpha) \vdash_M^* (p,\in,\alpha)$, and
3. If $(q,x,\alpha) \vdash_M^* (p,\in,\beta)$, where $\alpha,\beta \in \Gamma^*, x \in \Sigma^*$, and $p,q \in Q$, then $(q,x,\alpha\gamma) \vdash_M^* (p,\in,\beta\gamma)$, where $\gamma \in \Gamma^*$

6.1.5 Acceptance by PDA

Let M be a PDA, the accepted language is represented by N(M). We defined the acceptance by PDA in two ways.

1. Let $M = (Q,\Sigma,\Gamma,\delta, q_0, Z_0,F)$, then N(M) is accepted by final state such that

   $N(M)=\{w:(q_0,w,Z_0)\overset{*}{\underset{M}{\vdash}}(q_f,\in,\beta)$, where $q \in Q$, $w \in \Sigma^*, Z_0, \beta \in \Gamma^*$, and $q_f \in F\}$

   It is similar to the acceptance by FA discussed earlier. We define some final states and the accepted language *N(M)* is the set of all input strings for which some choice of moves leads to some final state.

2. Let $M = (Q,\Sigma,\Gamma,\delta,q_0,Z_0,\phi)$, then *N(M)* is accepted by empty stack or null stack such that $N(M) = \{w:(q_0,w,Z_0)\overset{*}{\underset{M}{\vdash}}(p,\in,\in)$, where $p \in Q, w \in \Sigma^*\}$

   The language *N(M)* is the set of all input strings for which some sequence of moves causes the PDA to empty its stack.

Note : If acceptance is defined by empty stack then there is no meaning of final state and it is represented by ϕ.

Example : consider a PDA $M = (\{q_0,q_1,q_2\},\{a,c\},\{a,Z_0\},\delta,q_0,Z_0,\{q_2\})$ shown in below figure. Check the acceptability of string aacaa.

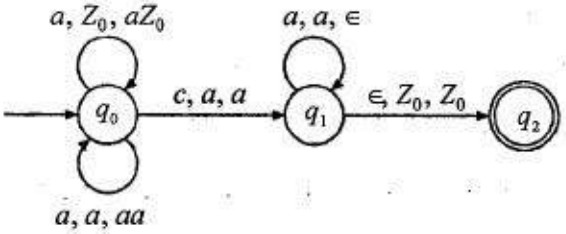


FIGURE : PDA accepting $\{a^n ca^n : n \geq 1\}$

Note : *Edges are labeled with Input symbol, stack symbol, written symbol on the stack.*

Solution :

The transition function δ is defined as follows :

$\delta(q_0,a,Z_0) = \{(q_0,aZ_0)\}$,

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

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

$\delta(q_1,a,a) = \{(q_1,\in)\}$, and

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

Following moves are carried out in order to check acceptability of string $aacaa$:

$$(q_0,aacaa,Z_0) \vdash (q_0,acaa,aZ_0)$$
$$\vdash (q_0,caa,aaZ_0)$$
$$\vdash (q_1,aa,aaZ_0)$$
$$\vdash (q_1,a,aZ_0)$$
$$\vdash (q_1,\in,Z_0)$$
$$\vdash (q_2,\in,Z_0)$$

Hence, $(q_0,aacaa,Z_0) \vdash_M^* (q_2,\in,Z_0)$.

Therefore, the string **aacaa** is accepted by M.

6. 2 CONSTRUCTION OF PDA

In this section, we shall see how PDA's can be constructed.

Example 1 : Obtain a PDA to accept the language $L(M) = \{ wCw^R \mid w \in (a+b)^* \}$ where W^R is reverse of W .

Solution:

It is clear from the language $L(M) = \{ wCw^R \}$ that if $w = abb$

then reverse of w denoted by W^R will be $W^R = bba$ and the language L will be wCw^R
i. e., abbCbba which is a string of palindrome.

So, we have to construct a PDA which accepts a palindrome consisting of a's and b's with the symbol C in the middle.

General Procedure :

To check for the palindrome, let us push all scanned symbols onto the stack till we encounter the letter C. Once we pass the middle string, if the string is a palindrome, for each scanned input symbol, there should be a corresponding symbol (same as input symbol) on the stack. Finally, if there is no input and stack is empty, we say that the given string is a palindrome.

Step 1 : Input symbols can be a or b .

Let q_0 be the initial state and Z_0 be the initial symbol on the stack. In state q_0 and when top of the stack is Z_0, whether the input symbol is a or b push it on to the stack, and remain in q_0. The transitions defined for this can be of the form

$$\delta(q_0, a, Z_0) = (q_0, aZ_0)$$
$$\delta(q_0, b, Z_0) = (q_0, bZ_0)$$

Once the first scanned input symbol is pushed on to the stack, the stack may contain either a or b. Now, in state q_0, the input symbol can be either a or b. Note that irrespective of what is the input or what is there on the stack, we have to keep pushing all the symbols on to the stack, till we encounter C. So, the transitions defined for this can be of the form

$$\delta(q_0, a, a) = (q_0, aa)$$
$$\delta(q_0, b, a) = (q_0, ba)$$
$$\delta(q_0, a, b) = (q_0, ab)$$
$$\delta(q_0, b, b) = (q_0, bb)$$

Step 2 : Input symbol is C

Now, if the next input symbol is C, the top of the stack may be a or b. Another possibility is that, in state q_0, the first symbol itself can be C. In this case w is null string and Z_0 will be on the stack. In all these cases, the input symbol is C i. e., the symbol which is present in the middle of the string. So, change the state to q_1 and do not alter the contents of the stack. The transitions defined for this can be of the form

$$\delta(q_0, C, Z_0) = (q_1, Z_0)$$
$$\delta(q_0, C, a) = (q_1, a)$$
$$\delta(q_0, C, b) = (q_1, b)$$

Now, we have passed the center of the string.

Step 3 : Input symbols can be a or b .

To be a palindrome, for each input symbol there should be a corresponding symbol (same as input symbol) on the stack. So, whenever the input symbol is same as symbol on the stack, remain in state q_1 and delete that symbol from the stack and repeat the process. The transitions defined for this can be of the form

$$\delta(q_1, a, a) = (q_1, \in)$$
$$\delta(q_1, b, b) = (q_1, \in)$$

Step 4 : Finally, in state q_1 , if the string is a palindrome, there is no input symbol to be scanned and the stack should be empty i. e., the stack should contain Z_0. Now, change the state to q_2 and do not alter the contents of the stack. The transition for this can be of the form

$$\delta(q_1, \in, Z_0) = (q_2, Z_0)$$

So, the PDA M to accept the language $L(M) = \{wCw^R \mid w \in (a,b)^*\}$ is given by

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

Where $Q = \{q_0, q_1, q_2\}$; $\Sigma = \{a, b, C\}$; $\Gamma = \{a, b, Z_0\}$

δ : is shown below,

$$\delta(q_0, a, Z_0) = (q_0, aZ_0)$$
$$\delta(q_0, b, Z_0) = (q_0, bZ_0)$$
$$\delta(q_0, a, a) = (q_0, aa)$$
$$\delta(q_0, b, a) = (q_0, ba)$$
$$\delta(q_0, a, b) = (q_0, ab)$$
$$\delta(q_0, b, b) = (q_0, bb)$$
$$\delta(q_0, C, Z_0) = (q_1, Z_0)$$
$$\delta(q_0, C, a) = (q_1, a)$$
$$\delta(q_0, C, b) = (q_1, b)$$
$$\delta(q_1, a, a) = (q_1, \in)$$
$$\delta(q_1, b, b) = (q_1, \in)$$
$$\delta(q_1, \in, Z_0) = (q_2, Z_0)$$

$q_0 \in Q$ is the start state of machine.

$Z_0 \in \Gamma$ is the initial symbol on the stack.

$F = \{q_2\}$ is the final state.

To accept the string :

The sequence of moves made by the PDA for the string **aabCbaa** is shown below.

Initial ID

$(q_0, aabCbaa, Z_0) \vdash (q_0, abCbaa, aZ_0)$

$\vdash (q_0, bCbaa, aaZ_0)$

$\vdash (q_0, Cbaa, baaZ_0)$

$\vdash (q_1, baa, baaZ_0)$

$\vdash (q_1, aa, aaZ_0)$

$\vdash (q_1, a, aZ_0)$

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

$\vdash (q_2, \in, Z_0)$

(Final Configuration)

Since q_2 is the final state and input string is $\in$ in the final configuration, the string **aabCbaa** is accepted by the PDA.

To reject the string :

The sequence of moves made by the PDA for the string **aabCbab** is shown below.

Initial ID

$(q_0, aabCbab, Z_0) \vdash (q_0, abCbab, aZ_0)$

$\vdash (q_0, bCbab, aaZ_0)$

$\vdash (q_0, Cbab, baaZ_0)$

$\vdash (q_1, bab, baaZ_0)$

$\vdash (q_1, ab, aaZ_0)$

$\vdash (q_1, b, aZ_0)$

(Final Configuration)

Since the transition $\delta(q_1, b, a)$ is not defined, the string **aabCbab** is not a palindrome and the machine halts and the string is rejected by the PDA.

Example 2 : Obtain a PDA to accept the language $L = \{ a^n b^n \mid n \geq 1 \}$ by a final state.

Solution :

The machine should accept n number of a's followed by n number of b's.

General Procedure :

Since n number of a's should be followed by n number of b's, let us push all the symbols on to the stack as long as the scanned input symbol is a. Once we encounter b's, we should see that for each b in the input, there should be corresponding a on the stack. When the input pointer reaches the end of the string, the stack should be empty. If stack is empty, it indicates that the string scanned has n number of a's followed by n number of b's.

Step 1 : Let q_0 be the start state and Z_0 be the initial symbol on the stack. As long as the next input symbol to be scanned is a, irrespective of what is there on the stack, keep pushing all the symbols onto the stack and remain in q_0. The transitions defined for this can be of the form

$$\delta(q_0, a, Z_0) = (q_0, aZ_0)$$

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

Step 2 : In state q_0, if the next input symbol to be scanned is b and if the top of the stack is a, change the state to q_1 and delete one b from the stack. The transition for this can be of the form

$$\delta(q_0, b, a) = (q_1, \in)$$

Step 3 : Once the machine is in state q_1, the rest of the symbols to be scanned will be only b's and for each b there should be corresponding symbol a on the stack. So, as the scanned input symbol is b and if there is a matching a on the stack, remain in q_1 and delete the corresponding a from the stack. The transitions defined for this can be of the form

$$\delta(q_1, b, a) = (q_1, \in)$$

Step 4 : In state q_1, if the next input symbol to be scanned is $\in$ and if the top of the stack is Z_0, (it means that for each b in the input there exists corresponding a on the stack) change the state to q_2 which is an accepting state. The transition defined for this can be of the form

$$\delta(q_1, \in, Z_0) = (q_2, \in)$$

So, the PDA to accept the language $L = \{ a^n b^n \mid n \geq 1 \}$ is given by $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

where $Q = \{q_0, q_1, q_2\}$; $\Sigma = \{a, b\}$; $\Gamma = \{a, Z_0\}$

δ : is shown below.

$$\delta(q_0, a, Z_0) = (q_0, aZ_0)$$

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

$$\delta(q_0, b, a) = (q_1, \in)$$

$$\delta(q_1, b, a) = (q_1, \in)$$

$$\delta(q_1, \in, Z_0) = (q_2, \in)$$

$q_0 \in Q$ is the start state of machine

$Z_0 \in \Gamma$ is the initial symbol on the stack

$F = \{q_2\}$ is the final state.

To accept the string :

The sequence of moves made by the PDA for the string **aaabbb** is shown below.

Initial ID

$(q_0, aaabbb, Z_0)$ $\vdash$ $(q_0, aabbb, aZ_0)$

$\vdash$ $(q_0, abbb, aaZ_0)$

$\vdash$ $(q_0, bbb, aaaZ_0)$

$\vdash$ (q_1, bb, aaZ_0)

$\vdash$ (q_1, b, aZ_0)

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

$\vdash$ $(q_2, \in, Z_0)$

(Final Configuration)

Since q_2 is the final state and input string is $\in$ in the final configuration, the string **aaabbb** is accepted by the PDA.

To reject the string :

The sequence of moves made by the PDA for the string **aabbb** is shown below.

Initial ID

$(q_0, aabbb, Z_0)$ $\vdash$ $(q_0, abbb, aZ_0)$

$\vdash$ (q_0, bbb, aaZ_0)

$\vdash$ (q_1, bb, aZ_0)

$\vdash$ (q_1, b, Z_0)

(Final Configuration)

Since the transition $\delta(q_1, b, Z_0)$ is not defined, the string **aabbb** is rejected by the PDA.

Example 3 : Obtain a PDA to accept the language $L(M) = \{ w | w \in (a+b)^* \text{ and } n_a(w) = n_b(w) \}$.

solution :

The language accepted by the machine should consist strings of a's and b's of any length. Only restriction is that number of a's in string w should be equal to number of b's. The order of a's and b's is irrelevant. For example aaabbb, ababab, aababb etc. are all the strings in the language L(M).

General Procedure :

The first scanned input symbol is pushed on to the stack. From this point onwards, if the scanned symbol is same as the symbol on to the stack, push the current input symbol on to the stack. If the input symbol is different from the symbol on the top of the stack, pop one symbol from the stack and repeat the process. Finally, when end of string is encountered, if the stack is empty, the string w has equal number of a's and b's, otherwise number of a's and b's are different.

Step 1 : Let q_0 be the start state and Z_0 be the initial symbol on the stack. When the machine is in state q_0 and when top of the stack contains Z_0, scan the input symbol (either a or b) and push it on to the stack. The transitions defined for this can be of the form.

$$\delta(q_0, a, Z_0) = (q_0, aZ_0)$$
$$\delta(q_0, b, Z_0) = (q_0, bZ_0)$$

Step 2 : Once the first input symbol is pushed on to the stack, the top of stack may contain either a or b and the next input symbol to be scanned may be a or b. If the input symbol is same as the symbol on top of the stack, push the current input symbol on to the stack and remain in state q_0 only. Otherwise, pop an element from the stack. The transitions defined for this can be of the form

$$\delta(q_0, a, a) = (q_0, aa)$$
$$\delta(q_0, b, b) = (q_0, bb)$$
$$\delta(q_0, a, b) = (q_0, \in)$$
$$\delta(q_0, b, a) = (q_0, \in)$$

Step 3 : In state q_0, if the next symbol to be scanned is $\in$ (empty string) and top of the stack is Z_0, it means that for each symbol a there exists a corresponding b and for each symbol b, there exists a symbol a. So, the string w consists of equal number of a's and b's and change the state to q_1. The transition defined for this can be of the form

$$\delta(q_0, \in, Z_0) = (q_1, Z_0)$$

So, the PDA to accept the language $L=\{w|n_a(w)=n_b(w)\}$ is given by $M=(Q,\Sigma,\Gamma,\delta,q_0,Z_0,F)$

where $Q=\{q_0, q_1\}$; $\Sigma=\{a, b\}$; $\Gamma=\{a, b, Z_0\}$

δ : is shown below

$$\delta(q_0,a,Z_0)=(q_0,aZ_0)$$
$$\delta(q_0,b,Z_0)=(q_0,bZ_0)$$
$$\delta(q_0,a,a)=(q_0,aa)$$
$$\delta(q_0,b,b)=(q_0,bb)$$
$$\delta(q_0,a,b)=(q_0,\in)$$
$$\delta(q_0,b,a)=(q_0,\in)$$
$$\delta(q_0,\in,Z_0)=(q_1,Z_0)$$

$q_0 \in Q$ is the start state of machine.

$Z_0 \in \Gamma$ is the initial symbol on the stack.

$F=\{q_1\}$ is the final state.

To accept the string :

The sequence of moves made by the PDA for the string **abbbaa** is shown below.

Initial ID

$(q_0, abbbaa, Z_0) \vdash (q_0, bbbaa, aZ_0)$

$\vdash (q_0, bbaa, Z_0)$

$\vdash (q_0, baa, bZ_0)$

$\vdash (q_0, aa, bbZ_0)$

$\vdash (q_0, a, bZ_0)$

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

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

(Final Configuration)

Since q_1 is the final state and input string is $\in$ in the final configuration, the string **abbbaa** is accepted by the PDA.

To reject the string :

The sequence of moves made by the PDA for the string aabbb is shown below.

Initial ID

$(q_0, aabbb, Z_0) \vdash (q_0, abbb, aZ_0)$

$\vdash (q_0, bbb, aaZ_0)$

$\vdash (q_0, bb, aZ_0)$

$\vdash$ (q_0, b, Z_0)

$\vdash$ $(q_0, \in, bZ_0)$

(Final Configuration)

Since the transition $\delta(q_0, \in b)$ is not defined, the string **aabbb** is rejected by the PDA.

Note : To accept the language by an empty stack, the final state is irrelevant, the next input symbol to be scanned should be $\in$, and stack should to be empty. Even Z_0 should not be then on the stack. So, the PDA to accept equal number of a's and b's using an empty stack, change only the last transition in example 3. The last transition

$$\delta(q_0, \in, Z_0) = (q_1, Z_0)$$

can be changed as

$$\delta(q_0, \in, Z_0) = (q_1, \in)$$

So, the PDA to accept the language $L = \{w | n_a(w) = n_b(w)\}$ by an empty stack is given by

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \in)$$

where $Q = \{q_0, q_1\}$; $\Sigma = \{a, b\}$; $\Gamma = \{a, b, Z_0\}$

δ : is shown below $\delta(q_0, a, Z_0) = (q_0, aZ_0)$

$\delta(q_0, b, Z_0) = (q_0, bZ_0)$

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

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

$\delta(q_0, a, b) = (q_0, \in)$

$\delta(q_0, b, a) = (q_0, \in)$

$\delta(q_0, \in, Z_0) = (q_1, \in)$

$q_0 \in Q$ is the start state of machine.

$Z_0 \in \Gamma$ is the initial symbol on the stack.

$F = \{\phi\}$.

Note that PDA is accepted by an empty stack.

The sequence of moves made by the PDA for the string **aabbab** by an empty stack is shown below.

Initial ID

$(q_0, aabbab, Z_0)$ $\vdash$ $(q_0, abbab, aZ_0)$

$\vdash$ $(q_0, bbab, aaZ_0)$

$\vdash$ (q_0, bab, aZ_0)

$\vdash (q_0, ab, Z_0)$

$\vdash (q_0, b, aZ_0)$

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

$\vdash (q_1, \in, \in)$

(Final Configuration)

Since the next input symbol to be scanned is $\in$ and the stack is empty, the string **aabbab** is accepted by an empty stack.

Note : q_1 is not the final state. Stack is empty .

Example 4 : Obtain a PDA to accept a string of balanced parentheses. The parentheses to be considered are (,) , [,] .

Solution :

Note1 : Some of the valid strings are : [( ) ( ) ([ ])] , $\in$, [ ] [ ] ( )
and invalid string are : [ ) ( ) [ ] ,) (, [ )

Note 2 : $\in$ (null string) is valid

Note 3 : Left parentheses can either be '(' or '[' and right parentheses can either be ')' or ']' .

Step 1 : Let q_0 be the start state and Z_0 be the initial symbol on the stack. The state q_0 itself is the final state accepting $\in$ (an empty string).

Step 2 : In the state q_0 , if the first scanned parentheses is '(' or '[', push the scanned symbol on to the stack and change the state to q_1 . The transition defined for this can be of the form

$$\delta(q_0, (, Z_0) = (q_1, (Z_0)$$

$$\delta(q_0, [, Z_0) = (q_1, [Z_0)$$

Step 3 : If at least one parentheses either '(' or '[' is present on the stack and if the scanned symbol is left parentheses, then push the left parentheses on to the stack. The transitions defined for this can be of the form

$$\delta(q_1, (, () = (q_1, (()$$

$$\delta(q_1, (, [) = (q_1, ([)$$

$$\delta(q_1, [, () = (q_1, [()$$

$$\delta(q_1, [, [) = (q_1, [[)$$

Step 4 : If the scanned symbol is ')' and if the top of the stack is '(' pop an element from the stack. Similarly, if the scanned symbol is ']' and if the top of the stack is '[' pop an element from the stack. The transitions defined for this can be of the form

$$\delta(q_1,) , ( ) = (q_1, \in)$$

$$\delta(q_1,] , [ ) = (q_1, \in)$$

Step 5 : When top of the stack is Z_0, it indicates that so far all the parentheses have been matched. At this point, on $\in$ - transition, the PDA enters into state q_0 and all the steps from step1 are repeated . The transition for this can be of the form

$$\delta(q_1, \in, Z_0) = (q_0, Z_0)$$

So, the PDA to accept the language consisting of balanced parentheses is given by

$$M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$$

where $Q = \{ q_0, q_1 \}$; $\Sigma = \{ (,), [,] \}$; $\Gamma = \{ (, [, Z_0 \}$

δ : is shown below .

$$\delta(q_0, (, Z_0) = (q_1, (Z_0)$$

$$\delta(q_0, [, Z_0) = (q_1, [Z_0)$$

$$\delta(q_1, (, ( ) = (q_1, (( )$$

$$\delta(q_1, (, [ ) = (q_1, ([ )$$

$$\delta(q_1, [, ( ) = (q_1, [( )$$

$$\delta(q_1, [, [ ) = (q_1, [[ )$$

$$\delta(q_1,), ( ) = (q_1, \in)$$

$$\delta(q_1,], [ ) = (q_1, \in)$$

$$\delta(q_1, \in, Z_0) = (q_0, Z_0)$$

$q_0 \in Q$ is the start state of machine.

$Z_0 \in \Gamma$ is the initial symbol on the stack.

$F = \{q_0\}$.

Note that even $\in$ is accepted by PDA and is valid.

To accept the string :

The sequence of moves made by the PDA for the string [( ) ( ) ([ ])] is shown below.

Initial ID

$$(q_0, [( ) ( ) ([ ])], Z_0) \vdash (q_1, ( ) ( ) ([ ])], [Z_0)$$

$$\vdash (q_1,) ( ) ([ ])], ([Z_0)$$

$$\vdash (q_1, ( ) ([ ])], [Z_0)$$

$\vdash (q_1, \quad)([\,])\,], ([Z_0)$

$\vdash (q_1, \quad ([\,])\,], [Z_0)$

$\vdash (q_1, \quad [\,])\,], ([Z_0)$

$\vdash (q_1, \quad])\,], [([Z_0)$

$\vdash (q_1, \quad)\,], ([Z_0)$

$\vdash (q_1, \quad], [Z_0)$

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

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

(Final Configuration)

Since the next input symbol to be scanned is $\in$ and the stack is empty, the string [()()([])] is accepted by the PDA.

Example 5 : Obtain a PDA to accept the language $L = \{ w \mid w \in (a, b)^* \text{ and } n_a(w) > n_b(w) \}$.

solution :

Note : The solution is similar to that of the problem discussed in example 3 in which we are accepting strings of a's and b's of equal numbers. When we encounter end of the input i. e., $\in$ and top of the stack is Z_0, it has equal number of a's and b's. But, what we want is a machine to accept more number of a's than b's. For this, only change to be made is that when we encounter $\in$ (i. e., end of the input), if top of the stack contains at least one a, then change the final state to q_1 and do not alter the contents of the stack. The transition defined remains same as problem shown in example 3, except the last transition. The last transition is of the form

$$\delta(q_0, \in, a) = (q_1, a)$$

So, the PDA to accept the language $L = \{w \mid n_a(w) > n_b(w)\}$

is given by $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

where $Q = \{q_0, q_1\}$; $\Sigma = \{a, b\}$; $\Gamma = \{a, b, Z_0\}$

δ : is shown below.

$$\delta(q_0, a, Z_0) = (q_0, aZ_0)$$

$$\delta(q_0, b, Z_0) = (q_0, bZ_0)$$

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

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

$$\delta(q_0, a, b) = (q_0, \in)$$
$$\delta(q_0, b, a) = (q_0, \in)$$
$$\delta(q_0, \in, a) = (q_1, a)$$

$q_0 \in Q$ is the start state of machine ; $Z_0 \in \Gamma$ is the initial symbol on the stack.

$F = \{q_1\}$ is the final state.

Note : On similar lines we can find a PDA to accept the language

$$L=\{w \mid w\in(a,b)^* and\ n_a(w)< n_b(w)\}$$

i. e., strings of a's and b's where number of b's are more than number of a's . To achieve this only change to be made in the above machine is change the final transition.

$$\delta(q_0, \in, a) = (q_1, a)$$

to

$$\delta(q_0, \in, b) = (q_1, b)$$

So, the PDA to accept the language $L = \{w|w\in(a,b)^* and\ n_a(w)< n_b(w)\}$ is given by $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

where $Q = \{q_0, q_1\}$; $\Sigma = \{a, b\}$; $\Gamma = \{a, b, Z_0\}$;

δ : is shown below.

$$\delta(q_0, a, Z_0) = (q_0, aZ_0)$$
$$\delta(q_0, b, Z_0) = (q_0, bZ_0)$$
$$\delta(q_0, a, a) = (q_0, aa)$$
$$\delta(q_0, b, b) = (q_0, bb)$$
$$\delta(q_0, a, b) = (q_0, \in)$$
$$\delta(q_0, b, a) = (q_0, \in)$$
$$\delta(q_0, \in, b) = (q_1, b)$$

$q_0 \in Q$ is the start state of machine ; $Z_0 \in \Gamma$ is the initial symbol on the stack

$F = \{q_1\}$ is the final state.

Example 6 : Obtain a PDA to accept the language $L = \{a^n b^{2n} \mid n \geq 1\}$.

solution :

The machine should accept n number of a's followed by 2n number of b's.

General Procedure :

Since n number of a's should be followed by 2n number of b's, for each a in the input, push two a's on to the stack. Once we encounter b's, we should see that for each b in the input, there should be corresponding a on the stack. When the input pointer reaches the end of the string, the stack should be empty. If stack is empty, it indicates that the string scanned has n number of a's followed by 2n number of b's .

Step 1 : Let q_0 be the start state and Z_0 be the initial symbol on the stack. For each scanned input symbol a, push two a's on to the stack. The transitions defined for this can be of the form

$$\delta(q_0, a, Z_0) = (q_0, aaZ_0)$$

$$\delta(q_0, a, a) = (q_0, aaa)$$

Step 2 : In state q_0, if the next input symbol to be scanned is b and if the top of the stack is a, change the state to q_1 and delete one b from the stack. The transition for this can be of the form

$$\delta(q_0, b, a) = (q_1, \in)$$

Step 3 : Once the machine is in state q_1, the rest of the symbols to be scanned will be only b's and for each b there should be corresponding symbol a on the stack. So, as the scanned input symbol is b and if there is a matching a on the stack, remain in q_1 and delete the corresponding a from the stack. The transitions defined for this can be of the form

$$\delta(q_1, b, a) = (q_1, \in)$$

Step 4 : In state q_1, if the next input symbol to be scanned is $\in$ and if the top of the stack is Z_0, (it means that for each b in the input there exists corresponding a on the stack) change the state to q_2 which is an accepting state. The transition defined for this can be of the form

$$\delta(q_1, \in, Z_0) = (q_2, \in)$$

So, the PDA to accept the language $L = \{ a^n b^{2n} \mid n \geq 1 \}$

is given by $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

where $Q = \{ q_0, q_1, q_2 \}$; $\Sigma = \{a, b\}$; $\Gamma = \{ a, Z_0 \}$

δ : is shown below.

$$\delta(q_0, a, Z_0) = (q_0, aaZ_0)$$

$$\delta(q_0, a, a) = (q_0, aaa)$$

$$\delta(q_0, b, a) = (q_1, \in)$$

$$\delta(q_1, b, a) = (q_1, \in)$$

$$\delta(q_1, \in, Z_0) = (q_2, \in)$$

$q_0 \in Q$ is the start state of machine ; $Z_0 \in \Gamma$ is the initial symbol on the stack.

$F = \{q_2\}$ is the final state.

To accept the string :

The sequence of moves made by the PDA for the string **aabbbb** is shown below.

Initial ID

$(q_0,\ aabbbb,\ Z_0)$ $\vdash$ $(q_0,\ abbbb,\ aaZ_0)$

$\vdash$ $(q_0,\ bbbb,\ aaaaZ_0)$

$\vdash$ $(q_0,\ bbb,\ aaaZ_0)$

$\vdash$ $(q_1,\ bb,\ aaZ_0)$

$\vdash$ $(q_1,\ b,\ aZ_0)$

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

$\vdash$ $(q_2,\ \in,\ Z_0)$

(Final Configuration)

Since q_2 is the final state and input string is $\in$ in the final configuration, the string **aabbbb** is accepted by the PDA.

To reject the string :

The sequence of moves made by the PDA for the string **aabbb** is shown below.

Initial ID

$(q_0,\ aabbb,\ Z_0)$ $\vdash$ $(q_0,\ abbb,\ aaZ_0)$

$\vdash$ $(q_0,\ bbb,\ aaaaZ_0)$

$\vdash$ $(q_0,\ bb,\ aaaZ_0)$

$\vdash$ $(q_0,\ b,\ aaZ_0)$

$\vdash$ $(q_0,\ \in,\ aZ_0)$

(Final Configuration)

Since the transition $\delta(q_0,\ \in,\ a)$ is not defined, the string **aabbb** is rejected by the PDA.

Example 7 : Obtain a PDA to accept the language $L = \{ ww^R \mid w \in (a+b)^* \}$.

solution :

It is clear from the language $L(M) = \{ ww^R \}$ that if $w = abb$

then reverse of w denoted by w^R will be $w^R = bba$

and the language L will be ww^R i. e., abbbba which is a string of palindrome.

So, we have to construct a PDA which accepts a palindrome consisting of a's and b's. This problem is similar to the problem discussed in example 1. Only difference is that in example 1, an extra symbol C acts as a pointer to the middle string. But , here there is no way to find the mid point for the string.

General Procedure :

To check for the palindrome, let us push all scanned symbols onto the stack till we encounter the mid point (Remember that there is no way to find the midpoint). Once we pass the middle string, to be a palindrome, for each scanned input symbol , there should be a corresponding symbol (same as input symbol) on the stack. Finally, if there is no input and stack is empty, we say that the given string is a palindrome.

Step 1 : Let q_0 be the initial state and Z_0 be the initial symbol on the stack. In state q_0 and when top of the stack is Z_0, whether the input symbol is a or b push it on to the stack, and remain in q_0. The transitions defined for this can be of the form

$$\delta(q_0, a, Z_0) = (q_0, aZ_0)$$
$$\delta(q_0, b, Z_0) = (q_0, bZ_0)$$

Once the first scanned input symbol is pushed on to the stack, the stack may contain either a or b. Now, in state q_0, the input symbol can be either a or b. Note that irrespective of what is the input or what is there on the stack, we have to keep pushing all the symbols on to the stack, till we encounter midpoint (But, there is no way to find mid point. We continue this process till we encounter mid point through our common sense).

So, the transitions defined for this can be of the form

$$\delta(q_0, a, a) = (q_0, aa)$$
$$\delta(q_0, b, a) = (q_0, ba)$$
$$\delta(q_0, a, b) = (q_0, ab)$$
$$\delta(q_0, b, b) = (q_0, bb)$$

Step 2 : Now, once we reach the midpoint, the top of the stack may be a or b. To be a palindrome, for each input symbol there should be a corresponding symbol (same as input symbol) on the stack. So, whenever the input symbol is same as symbol on the stack, change the state to q_1 and delete that symbol from the stack. The transitions defined for this can be of the form

$$\delta(q_0, a, a) = (q_1, \in)$$
$$\delta(q_0, b, b) = (q_1, \in)$$

Step 3 : Now, once we are in state q_1, it means that we have passed the mid point. Now, the top of the stack may be a or b. To be a palindrome, for each input symbol there should be a corresponding symbol (same as input symbol) on the stack. So, whenever the input symbol is same as symbol on the stack, remain in state q_1 and delete that symbol from the stack. The transitions defined for this can be of the form

$$\delta(q_1, a, a) = (q_1, \in)$$
$$\delta(q_1, b, b) = (q_1, \in)$$

Step 4 : Finally, in state q_1, if the string is a palindrome, there is no input symbol to be scanned and the stack should be empty i. e., the stack should contain Z_0. Now, change the state to q_2 and do not alter the contents of the stack. The transition for this can be of the form

$$\delta(q_1, \in, Z_0) = (q_2, Z_0)$$

So, the PDA M to accept the language $L(M) = \{ ww^R \mid w \in (a, b)^* \}$

is given by $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

where $Q = \{ q_0, q_1, q_2 \}$; $\Sigma = \{ a, b \}$; $\Gamma = \{ a, b, Z_0 \}$

δ : is shown below.

$$\delta(q_0, a, Z_0) = (q_0, aZ_0)$$
$$\delta(q_0, b, Z_0) = (q_0, bZ_0)$$
$$\delta(q_0, a, a) = (q_0, aa)$$
$$\delta(q_0, b, a) = (q_0, ba)$$
$$\delta(q_0, a, b) = (q_0, ab)$$
$$\delta(q_0, b, b) = (q_0, bb)$$
$$\delta(q_0, a, a) = (q_1, \in)$$
$$\delta(q_0, b, b) = (q_1, \in)$$
$$\delta(q_1, a, a) = (q_1, \in)$$
$$\delta(q_1, b, b) = (q_1, \in)$$
$$\delta(q_1, \in, Z_0) = (q_2, Z_0)$$

$q_0 \in Q$ is the start state of machine ; $Z_0 \in \Gamma$ is the initial symbol on the stack.

$F = \{q_2\}$ is the final state.

Note that the transitions numbered 3 and 7, 6 and 8 can be combined and the transitions can be written as shown below also .

$$\delta(q_0, a, Z_0) = (q_0, aZ_0)$$
$$\delta(q_0, b, Z_0) = (q_0, bZ_0)$$
$$\delta(q_0, a, a) = \{ (q_0, aa), (q_1, \in) \}$$
$$\delta(q_0, b, a) = (q_0, ba)$$
$$\delta(q_0, a, b) = (q_0, ab)$$
$$\delta(q_0, b, b) = \{ (q_0, bb), (q_1, \in) \}$$
$$\delta(q_1, a, a) = (q_1, \in)$$
$$\delta(q_1, b, b) = (q_1, \in)$$
$$\delta(q_1, \in, Z_0) = (q_2, Z_0)$$

Note that once the following transitions are applied

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

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

if the input symbol is same as the symbol on top of the stack, the machine may push the current symbol on to the stack, or it may pop an element from the stack. At this point, the machine makes appropriate decision so that if the string is a palindrome, it has to accept. This machine is clearly a non - deterministic PDA (in short we call NPDA).

To accept the string :

The sequence of moves made by the PDA for the string **aabbaa** is shown below.

Initial ID

$(q_0, aabbaa, Z_0) \vdash (q_0, abbaa, aZ_0)$

$\vdash (q_0, bbaa, aaZ_0)$

$\vdash (q_0, baa, baaZ_0)$

PDA now pops an element instead of pushing $aa, aaZ_0)$

$a, aZ_0)$

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

$\vdash (q_2, \in, Z_0)$

(Final Configuration)

Since q_2 is the final state and input string is $\in$ in the final configuration, the string **aabbaa** is accepted by the PDA.

Example 8 : Construct a PDA which accepts the set of strings over $\{a, b\}$ with equal number of a's and b's such that all a's and b's are consecutive.

Solution :

We construct PDA M, which accepts given language by

(a) Empty store, and

(b) Final state

(a) By empty Store :

Let $M = (\{q_0\}, \{a,b\}, \{a,b,Z_0\}, \delta, q_0, Z_0, \phi)$. We know that the given language contains all the words over $\{a, b\}$ that have equal number of consecutive a's and b's. So, the given language

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

We use stack either to hold a's to match with b's or b's to match with a's.

Transition function δ is defined as follows :

$\delta(q_0,a,Z_0) = \{(q_0,aZ_0)\}$, (To store first a on the stack)

$\delta(q_0,a,a) = \{(q_0,aa)\}$, (To store remaining a's on the stack)

$\delta(q_0,b,Z_0) = \{(q_0,bZ_0)\}$, (To store first b on the stack)

$\delta(q_0,b,b) = \{(q_0,bb)\}$, (To store remaining b's on the stack)

$\delta(q_0,b,a) = \{(q_0,\in)\}$, (To match input b with a on the stack in case of $L = \{a^n b^n : n \geq 0\}$

$\delta(q_0,a,b) = \{(q_0,\in)\}$, (To match input a with b on the stack in case of $L = \{b^n a^n : n \geq 0\}$)

$\delta(q_0,\in,Z_0) = \{(q_0,\in)\}$ (To make the stack empty)

(b) By final State : Let $M = (\{q_0,q_f\},\{a,b\},\{a,b,Z_0\},\delta,q_0,Z_0,\{q_f\})$

$\delta(q_0,a,Z_0) = \{(q_0,aZ_0)\}$, (To store first a on the stack)

$\delta(q_0,a,a) = \{(q_0,aa)\}$, (To store remaining a's on the stack)

$\delta(q_0,b,Z_0) = \{(q_0,bZ_0)\}$, (To store first b on the stack)

$\delta(q_0,b,b) = \{(q_0,bb)\}$, (To store remaining b's on the stack)

$\delta(q_0,b,a) = \{(q_0,\in)\}$, (To match input b with a on the stack)

$\delta(q_0,a,b) = \{(q_0,\in)\}$, (To match input a with b on the stack)

$\delta(q_0,\in,Z_0) = \{(q_f,Z_0)\}$ (To reach the final state)

Example 9 : Construct a PDA, which accepts $L = \{a^n c^m b^n : m, n \geq 1\}$.

Solution : Suppose PDA $M = (\{q_0\},\{a,b\},\{a,b,Z_0\},\delta,q_0,Z_0,\phi)$ accepts L.

We have restriction imposed on the number of a's and b's that it should be equal but not on the number of c's. So, PDA stores all a's on the stack and when c's encounter, then keep the stack unchanged and when b's encounters then matches with a's stored on the stack.

δ is defined as follows

$\delta(q_0,a,Z_0) = \{(q_0,aZ_0)\}$, (To store first a on the stack)

$\delta(q_0,a,a) = \{(q_0,aa)\}$, (To store remaining a's on the stack)

$\delta(q_0,c,a) = \{(q_0,a)\}$, (To read all c's on the input tape and keep stack contents unchanged),

$\delta(q_0,b,a) = \{(q_0,\in)\}$, (To match input b with a on the stack)

$\delta(q_0,\in,Z_0) = \{(q_0,\in)\}$ (To empty the stack)

Example 10 : Construct a PDA M, which accepts $L = \{a^n b^m c^m d^n : m, n \geq 1\}$

Solution :

We construct M by using the grammar of given language L. But, if we see the format of language L, then it is clear that each word of language L contains number of a's in the starting which is equal to the number of d's at the end. In the middle the number of b's is followed by an equal number of c's. So, all the starting a's and following b's are loaded on to the stack. Now, stack will contain b's at the top and a's at the bottom (LIFO) and on the input tape $c^m d^n$ remains. After this PDA M matches the number of c's with b's and d's with a's. This is shown in the below figure.

Let PDA $M = (\{q_0, q_f\}, \{a, b\}, \{a, b, Z_0\}, \delta, q_0, Z_0, \{q_f\})$, and δ is defined as follows

$\delta(q_0, a, Z_0) = \{(q_0, aZ_0)\}$ (To store first a on the stack),

$\delta(q_0, a, a) = \{(q_0, aa)\}$ (To store remaining a's on the stack),

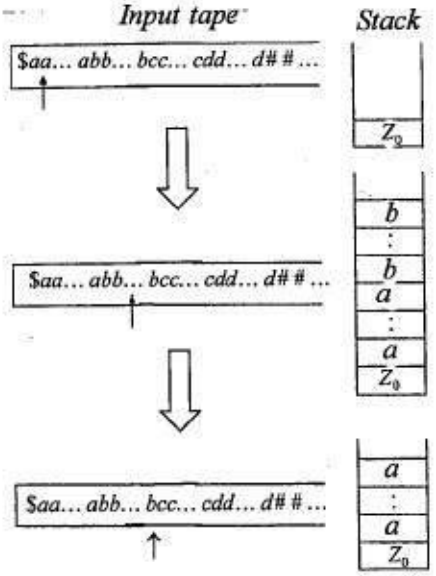
$\delta(q_0, b, a) = \{(q_0, ba)\}$ (To store first b on the stack and keeping stack contents unchanged),

$\delta(q_0, b, b) = \{(q_0, bb)\}$ (To store remaining b's on the stack),

$\delta(q_0, c, b) = \{(q_0, \in)\}$ (To match c's on the tape with b's on the stack),

$\delta(q_0, d, a) = \{(q_0, \in)\}$ (To match d's on the tape with a's on the stack), and

$\delta(q_0, \in, Z_0) = \{(q_f, Z_0)\}$ (To reach the final state)



Example 11 : Let $M = (\{p,q\},\{0,1\},\{Z_0,X\}, \delta, p, Z_0, \phi)$ be a PDA where δ is given by following transitions.

$\delta(p,1,Z_0) = \{(p,XZ_0)\}$,

$\delta(p,\in,Z_0) = \{(p,\in)\}$,

$\delta(p,1,X) = \{(p,XX)\}$,

$\delta(q,1,X) = \{(q,\in)\}$,

$\delta(p,0,X) = \{(q,X)\}$, *and*

$\delta(q,0,Z_0) = \{(p,Z_0)\}$

(a) What is the language accepted by this PDA by empty store ?

(b) Describe informally the working of the PDA.

Solution :

Let $R_1 : \delta(p,1,Z_0) = \{(p,XZ_0)\}$,

$R_2 : \delta(p,\in,Z_0) = \{(p,\in)\}$,

$R_3 : \delta(p,1,X) = \{(p,XX)\}$,

$R_4 : \delta(q,1,X) = \{(q,\in)\}$,

$R_5 : \delta(p,0,X) = \{(q,X)\}$, and

$R_6 : \delta(q,0,Z_0) = \{(p,Z_0)\}$

From the given transitions, we analyze the following things.

1. Using R_1 terminal 1 is stored as X on the stack in the state p .
2. Using R_2 , with no input PDA makes the stack empty in the state p .
3. Using R_3 , remaining 1's are stored as X's on the stack in the state p .
4. Using R_4 , input 1's are matched with X's stored on the stack in state q.
5. Using R_5 , PDA reads 0 and moves to the state q while maintaining X on the stack.
6. Using R_6, PDA reads 0 on the tape and moves to the state p while maintaining Z_0 on the stack.

So, PDA reads 1's on the tape and loads these on to stack as X's (R_1 *and* R_3) . When 0 is read in state p then moves to the state $q(R_5)$. In the state q , 1's on the tape are matched with X's on the stack (R_4) and when 0 is read on the tape then PDA changes its state to $p(R_6)$. In the state p if no input is remaining on the tape and Z_0 is on the stack then PDA makes the stack empty (R_2) .

So, the accepted language $L = \{1^n 0 1^n 0 : n \geq 0\}$.

Example 12 : Let $Q = (\{q_0, q_1\},\{a,b\},\{a,b,Z\}, \delta, q_0, Z, \phi)$ be a PDA accepting by empty stack for the language which is the set of all nonempty even palindromes over the set $\{a, b\}$. Below is an incomplete specification of the transition δ. Complete the specification. The top of the stack is assumed to be at the right end of the string representing stack contents.

1. $\delta(q_0, a, Z) = \{(q_0, Za)\}$
2. $\delta(q_0, b, Z) = \{(q_0, Zb)\}$
3. $\delta(q_0, a, a) =$ ..... ,
4. $\delta(q_0, b, b) =$ ..... ,
5. $\delta(q_1, a, a) = \{(q_1, \in)\}$
6. $\delta(q_1, b, b) = \{(q_1, \in)\}$ and
7. $\delta(q_1, \in, Z) = \{(q_1, \in)\}$

Solution :

Let the set of even palindromes over $\{a, b\}$ is L, then

$L = \{\in, aa, bb, abba, baba, aaaa, bbbb, \ldots\}$

If we find the mid of a palindrome then left is the mirror image of right and right is the mirror image of the left. So, deciding the mid point is the problem here. We design a DPDA for given set.

In the given example either terminal symbol is stored on the stack to be matched later on after the mid point or if the corresponding match is there then PDA popped the stack symbol.

In the given transitions using (1) and (2) PDA loads a or b symbol on the stack, using (5) and (6) PDA matches the input with stack symbol and using (7) PDA makes the stack empty. So, if more a or b symbols are there then PDA will use (3) and (4) and in these either symbol will be loaded on the stack and remain in the state q_1 or matched with the same symbol and moved to state q_2.

So, (3) and (4) solve the non-determinism problem to select the mid point into the palindromes.

So, transitions are given below.

1. $\delta(q_0, a, Z) = \{(q_0, Za)\}$,
2. $\delta(q_0, b, Z) = \{(q_0, Zb)\}$
3. $\delta(q_0, a, a) = \{(q_0, aa), (q_1, \in)\}$
4. $\delta(q_0, b, b) = \{(q_0, bb), (q_1, \in)\}$
5. $\delta(q_1, a, a) = \{(q_1, \in)\}$
6. $\delta(q_1, b, b) = \{(q_1, \in)\}$ and
7. $\delta(q_1, \in, Z) = \{(q_1, \in)\}$

6.3 DETERMINISTIC AND NONDETERMINISTIC PUSHDOWN AUTOMATA

In this section, we will discuss about the deterministic and nondeterministic behavior of pushdown automata.

6.3.1 Nondeterministic PDA (NPDA)

Like NFA, nondeterministic PDA (NPDA) has finite number of choices for its inputs. As we have discussed in the mathematical description that transition function δ which maps from $Q \times (\Sigma \cup \{\in\}) \times \Gamma$ to (**finite subset** of) $Q \times \Gamma^*$. A nondeterministic PDA accepts an input if a sequence of choices leads to some final state or causes PDA to empty its stack. Since, sometimes it has more than one choice to move further on a particular input ; it means, PDA guesses the right choice always, otherwise it will fail and will be in hang state.

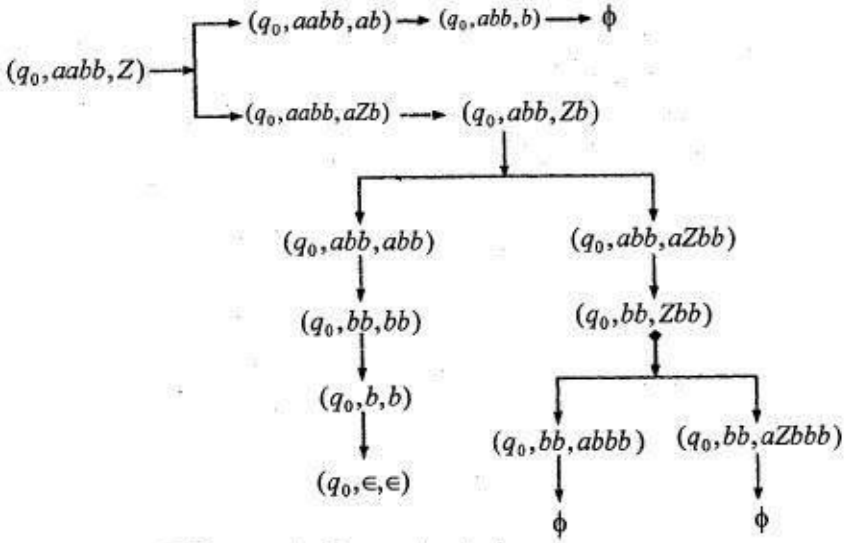
Example : consider a nondeterministic PDA $M = (\{q_0\},\{a,b\},\{a,b,Z\},\delta,q_0,Z,\phi)$, for the language $L = \{a^n b^n : n \geq 1\}$, where δ is defined as follows :

$\delta(q_0,\in,Z) = \{(q_0,ab),(q_0,aZb)\}$ (Two possible moves for input $\in$ on the tape and Z on the stack),

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

Check whether string $w = aabb$ is accepted or not ?

Solution : Initial configuration is $(q_0,aabb,Z)$. Following moves are possible :



Hence, $w = aabb$ is accepted by empty stack.

One thing is noticeable here that only one move sequence leads to empty store and other don't. In other words, we say that some move sequence(s) leads to accepting configuration and other lead to hang state.

6.3.2 Deterministic PDA (DPDA)

Deterministic PDA (DPDA) is just like DFA, which has *at most one choice* to move for certain input. A PDA $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ is deterministic if it satisfies both the conditions given as follows :

1. For any $q \in Q$, $a \in (\Sigma \cup \{\in\})$, and $Z \in \Gamma$, $\delta(q, a, Z)$ has at most one choice of move.
2. For any $q \in Q$, and $Z \in \Gamma$, if $\delta(q, \in, Z)$ is defined i.e. $\delta(q, \in, Z) \neq \phi$, then $\delta(q, a, Z) = \phi$ for all $a \in \Sigma$

Example : Consider a DPDA $M = (\{q_0, q_1\}, \{a, c\}, \{a, Z_0\}, \delta, q_0, Z_0, \phi)$ accepting the language $\{a^n c a^n : n \geq 1\}$, where δ is defined as follows :

$\delta(q_0, a, Z_0) = \{(q_0, aZ_0)\}$

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

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

$\delta(q_1, a, a) = \{(q_1, \in)\}$, and $\delta(q_1, \in, Z_0) = \{(q_1, \in)\}$

Check whether the string $w = aacaa$ is accepted by empty stack or not ?

Solution :

We see that in each transition DPDA has at most one move. Initial configuration is $(q_0, aacaa, Z_0)$. Following are the possible moves.

$$(q_0, aacaa, Z_0) \rightarrow (q_0, acaa, aZ_0) \rightarrow (q_0, caa, aaZ_0) \rightarrow (q_1, aa, aaZ_0)$$

$$\downarrow$$

$$(q_1, \in, \in) \leftarrow (q_1, \in, Z_0) \leftarrow (q_1, a, aZ_0)$$

Hence, the string $w = aacaa$ is accepted by empty stack.

As we have discussed in earlier chapters that DFA and NFA are equivalent with respect to the language acceptance, but the same is not true for the PDA.

For example, language $L = \{ww^R : w \in (a \cup b)^*\}$ is accepted by nondeterministic PDA, can not by any deterministic PDA. A nondeterministic PDA can not be converted into equivalent deterministic PDA, but all DCFLs which are accepted by DPDA, are also accepted by NPDA. So, we say that deterministic PDA is a proper subset of nondeterministic PDA. Hence, the power of nondeterministic PDA is more as compared to deterministic PDA.

Procedure to find whether PDA is deterministic or not

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA. The PDA is deterministic if

1. $\delta(q, a, Z)$ has only one element.
2. If $\delta(q, \in, Z)$ is not empty, then $\delta(q, a, Z)$ should be empty.

Both the conditions should be satisfied for the PDA to be deterministic. If one of the conditions fails, the PDA is non - deterministic.

Example 1 : Is the PDA to accept the language $L(M) = \{w\, Cw^R \mid w \in (a+b)^*\}$ is deterministic or not ?

Solution :

The transitions defined for this machine are

$$\delta(q_0, a, Z_0) = (q_0, aZ_0)$$
$$\delta(q_0, b, Z_0) = (q_0, bZ_0)$$
$$\delta(q_0, a, a) = (q_0, aa)$$
$$\delta(q_0, b, a) = (q_0, ba)$$
$$\delta(q_0, a, b) = (q_0, ab)$$
$$\delta(q_0, b, b) = (q_0, bb)$$
$$\delta(q_0, C, Z_0) = (q_1, Z_0)$$
$$\delta(q_0, C, a) = (q_1, a)$$
$$\delta(q_0, C, b) = (q_1, b)$$
$$\delta(q_1, a, a) = (q_1, \in)$$
$$\delta(q_1, b, b) = (q_1, \in)$$
$$\delta(q_1, \in, Z_0) = (q_2, Z_0)$$

The PDA should satisfy the two conditions shown in the procedure to be deterministic.

1. $\delta(q, a, Z)$ has only one element : Note that in the transitions, for each $q \in Q$, $a \in \Sigma$ *and* $Z \in \Gamma$, there is only one component defined and the first condition is satisfied.
2. The second condition states that if $\delta(q, \in, Z)$ is not empty, then $\delta(q, a, Z)$ should be empty i.e., if there is an $\in$ – transition, (in this case it is $\delta(q_1, \in, Z_0)$) , then there should not be any transition from the state q_1 when top of the stack is Z_0 which is true.

Since, the PDA satisfies both the conditions, the PDA is deterministic.

Example 2 : Is the PDA corresponding to the language $L = \{ a^n b^n | n \geq 1 \}$ by a final state is deterministic or not ?

Solution :

The transitions defined for this machine are

$$\delta(q_0, a, Z_0) = (q_0, aZ_0)$$
$$\delta(q_0, a, a) = (q_0, aa)$$
$$\delta(q_0, b, a) = (q_1, \in)$$
$$\delta(q_1, b, a) = (q_1, \in)$$
$$\delta(q_1, \in, Z_0) = (q_2, \in)$$

The first condition to be deterministic is $\delta(q, a, Z)$ should have only one component. In this case, for each $q \in Q$, $a \in \Sigma$ and $Z \in \Gamma$, there exists only one definition. So, the first condition is satisfied.

To satisfy the second condition, consider the transition

$$\delta(q_1, \in, Z_0) = (q_2, \in)$$

Since the transition is defined, the transition $\delta(q_1, a, Z_0)$ where $a \in \Sigma$ should not be defined which is true. Since both the conditions are satisfied, the given PDA is deterministic.

Example 3 : Is the PDA to accept the language $L(M) = \{w | w \in (a+b)^* \text{ and } n_a(w) = n_b(w)$ is deterministic or not ?

Solution : The transitions defined for this machine are

$$\delta(q_0, a, Z_0) = (q_0, aZ_0)$$
$$\delta(q_0, b, Z_0) = (q_0, bZ_0)$$
$$\delta(q_0, a, a) = (q_0, aa)$$
$$\delta(q_0, b, b) = (q_0, bb)$$
$$\delta(q_0, a, b) = (q_0, \in)$$
$$\delta(q_0, b, a) = (q_0, \in)$$
$$\delta(q_0, \in, Z_0) = (q_1, Z_0)$$

The first condition to be deterministic is $\delta(q, a, Z)$ should have only one component. In this case, for each $q \in Q$, $a \in \Sigma$ and $Z \in \Gamma$, there exists only one component. So, the first condition is satisfied.

To satisfy the second condition, consider the transition

$$\delta(q_0, \in, Z_0) = (q_1, Z_0)$$

Since this transition is defined, the transition $\delta(q_0, a, Z_0)$ where $a \in \Sigma$ should not be defined. But, there are two transitions

$$\delta(q_0, a, Z_0) = (q_0, aZ_0)$$
$$\delta(q_0, b, Z_0) = (q_0, bZ_0)$$

defined from q_0 when top of the stack is Z_0. Since the second condition is not satisfied, the given PDA is non - deterministic PDA.

Example 4 : Give a deterministic PDA for the language $L = \{a^n cb^{2n} : n \geq 1\}$ over the alphabet $\Sigma = \{a, b, c\}$. Specify the acceptance state.

Solution :

Let DPDA $M = (\{q_a, q_b, q_f\}, \{a, b, c\}, \{a, Z_0\}, \delta, q_a, Z_0, \{q_f\})$ accepts the given language L. We analyze that in the given L, in each word, double of number of a's at the starting is equal to the number of b's at the end and no restriction apposed on the c in relation with the number of a's or b's. So, one a is read on the tape and stored as aa on the stack and when b's encounted then matched with a's on the stack.

In state q_a all a's are read and stored on the stack and when c is read then DPDA moves in the state q_b and in the state q_b, b's are read on the tape and matched with a's on the stack. When no symbol is on the tape then DPDA moves to the final state q_f.

The transition function δ is defined as follows:

$\delta(q_a, a, Z_0) = \{(q_a, aaZ_0)\}$ (To store the first a as aa on the stack),

$\delta(q_a, a, a) = \{(q_a, aaa)\}$ (To store the remaining a's as double on the stack),

$\delta(q_a, c, a) = \{(q_b, a)\}$ (To read c and move to state q_b),

$\delta(q_b, b, a) = \{(q_b, \in)\}$ (To match the b's and a's stored on the stack), and

$\delta(q_b, \in, Z_0) = \{(q_f, Z_0)\}$ (To reach the final state)

The acceptance state is q_f and DPDA M is shown in below figure.

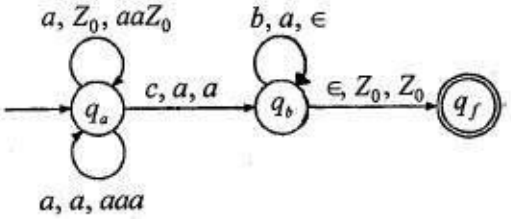


FIGURE : PDA accepting $\{a^n cb^{2n} : n \geq 1\}$

6.4 ACCEPTANCE OF LANGUAGE BY PDA

The language can be accepted by a Push Down Automata using two approaches.

1. **Acceptance by Final State :** The PDA accepts its input by consuming it and then it enters in the final state.

2. **Acceptance by empty stack :** On reading the input string from initial configuration for some PDA, the stack of PDA gets empty.

6.4.1 Equivalence of Empty Store and Final state acceptance

Theorem:

If $M_1 = (Q_1, \Sigma, \Gamma_1, \delta_1, p_1, Z_1, \phi)$ is a PDA accepting CFL L by empty store then there exists PDA $M_2 = (Q_2, \Sigma, \Gamma_2, \delta_2, p_2, Z_2, \{q_f\})$ which accepts L by final state.

Proof :

First we construct PDA M_2 based on PDA M_1 and then we prove that both accept L.

Step 1 : Construction of PDA M_2 based on given PDA M_1

Σ is same for both PDAs. We add a new initial state and a new final state with given PDA M_1.

So, $Q_2 = Q_1 \cup \{p_2 \cup q_f\}$

The stack alphabet Γ_2 of PDA M_2 contains one additional symbol Z_2 with Γ_1.

So, $\Gamma_2 = \Gamma_1 \cup \{Z_2\}$

The transition function δ_2 contains all the transitions of given PDA M_1 and two additional transitions (R_1 *and* R_3) as defined as follows :

$R_1 : \delta_2(p_2, \in, Z_2) = \{(p_1, Z_1 Z_2)\}$,

$R_2 : \delta_2(q, a, Z) = \delta_1(q, a, Z)$ for all (q, a, Z) in $Q_1 \times (\Sigma \cup \{\in\}) \times \Gamma_1$
(the original transitions of M_1), and

$R_3 : \delta_2(q, \in, Z_2) = \{(q_f, \in)\}$ for all $q \in Q_1$

By the R_1, M_2 moves from its initial ID $(p_2, \in, Z_2)$ to the initial ID of M_1. By R_2, M_2 uses all the transitions of M_1 after reaching the initial ID of M_1 and by using R_3 M_2 reaches the final state q_f.

The block diagram is shown in below figure.

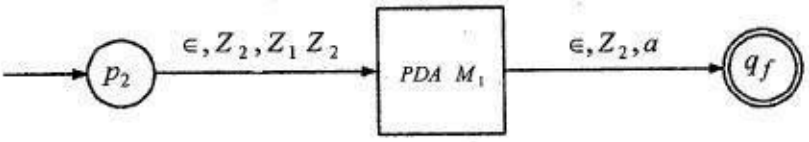


FIGURE : Block diagram of PDA M_2

Step 2 : The language accepted by PDA M_1 and PDA M_2

The behaviors of M_1 and M_2 are same except the two by $\in$ - moves defined by R_1 *and* R_3.

Let string $w \in L$ and accepted by M_1, then

$(p_1, w, Z_1) \vdash_{M_1}^{*} (q, \in, \in)$ where $q \in Q_1$ **(Result 1)**

For M_2, the initial ID is (p_2, w, Z_2) and it can be written as $(p_2, \in w \in, Z_2)$. So,

$(p_2, \in w \in, Z_2) \vdash_{M_2} (p_1, w, Z_1 Z_2)$ (This initial ID of M_1)

$\vdash_{M_2}^{*} (q, \in, Z_2)$ (by R_2 and Result 1)

$\vdash_{M_2}^{*} (q_f, \in, \alpha)$ $\alpha \in \Gamma_2^{*}$ (By R_3)

Thus, if M_1 accepts w, then M_2 also accepts it.

It means $L(M_2) \subseteq L(M_1)$ **(Result 2)**

Let string $w \in L$ and accepted by PDA M_2, then

$(p_2, \in w \in, Z_2) \vdash_{M_2} (p_1, w, Z_1 Z_2)$ (By R_1) **(Result 3)**

$\vdash_{M_2}^{*} (q, \in, Z_2)$ (By R_2) **(Result 4)**

$\vdash_{M_2} (q_f, \in, \alpha)$ $\alpha \in \Gamma_2^{*}$ (By R_3)

Note : The Result 3 is the initial ID of M_1. The Result 4 shows the empty store for M_1 if symbol Z_2 is not there.

For M_1, the initial ID is (p_1, w, Z_1)

So, $(p_1, w, Z_1) \vdash_{M_2}^{*} (q, \in, \in)$, where $q \in Q_1$ (By Result 3 and Result 4) Thus, if M_2 accepts w, then M_1 also accepts it.

It means, $L(M_1) \subseteq L(M_2)$ **(Result 5)**

Therefore, $L = L(M_2) = L(M_1)$ (From Result 2 and Result 5)

Hence, the statement of theorem is proved.

Example: Consider a nondeterministic PDA $M_1 = (\{q_0\}, \{a,b\}, \{a,b,S\}, \delta, q_0, S, \phi)$ which accepts the language $L = \{a^n b^n : n \geq 1\}$ by empty store, where δ is defined as follows :

$\delta(q_0, \in, S) = \{(q_0, ab), (q_0, aSb)\}$ (Two possible moves),

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

Construct an equivalent PDA M_2 which accepts L in final state and check whether string $w = aabb$ is accepted or not ?

Solution : Following moves are carried out by PDA M_1 in order to accept $w = aabb$:

$$(q_0, aabb, S) \vdash (q_0, aabb, aSb)$$
$$\vdash (q_0, abb, Sb)$$
$$\vdash (q_0, abb, abb)$$
$$\vdash (q_0, bb, bb)$$
$$\vdash (q_0, b, b)$$
$$\vdash (q_0, \in, \in)$$

Hence, $(q_0, aabb, S) \vdash_{M_1}^{*} (q_0, \in, \in)$

Therefore, $w = aabb$ is accepted by M_1.

Construction of PDA M_2 based on given PDA M_1

Let PDA $M_2 = (Q_2, \Sigma, \Gamma_2, \delta_2, p_2, Z_2, \{q_f\})$, where

$Q_2 = \{q, q_f, p_2\}$; $\Sigma = \{a, b\}$

$\Gamma_2 = \{a, b, S, Z_2\}$, and transition function δ_2 is defined as follows :

$\delta_2(p_2, \in, Z_2) = \{(q, SZ_2)\}$ (Using R_1)

$\delta_2(q, \in, S) = \{(q, ab), (q, aSb)\}$ (Using R_2)

$\delta_2(q, a, a) = \{(q, \in)\}$ (Using R_2)

$\delta_2(q, b, b) = \{(q, \in)\}$ (Using R_2)

$\delta_2(q, \in, Z_2) = \{(q_f, \in)\}$ (Using R_3)

Following moves are carried out by PDA M_2 in order to accept $w = aabb$:

$$(p_2, aabb, Z_2) \vdash (q, aabb, SZ_2)$$
$$\vdash (q, aabb, aSbZ_2)$$
$$\vdash (q, abb, SbZ_2)$$
$$\vdash (q, abb, abbZ_2)$$
$$\vdash (q, bb, bbZ_2)$$
$$\vdash (q, b, bZ_2)$$
$$\vdash (q, \in, Z_2)$$
$$\vdash (q_f, \in, Z_2)$$

Hence, $(p_2, aabb, Z_2) \overset{*}{\underset{M_2}{\vdash}} (q_f, \in, Z_2)$

The PDA M_2 is shown in below figure.

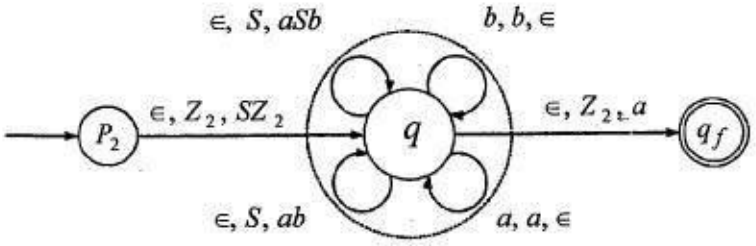


FIGURE : PDA M_2 accepting $\{a^n b^n : n \geq 1\}$

6.4.2 Equivalence of Final state and Empty Store acceptance

Theorem :

If $M_1 = (Q_1, \Sigma, \Gamma, \delta_1, q_0, Z_0, F)$ is a PDA accepting CFL L by final state then there exists PDA $M_2 = (Q_2, \Sigma, \Gamma, \delta_2, q_0, Z_0, \phi)$ which accepts L by empty store.

Proof :

First we construct PDA M_2 based on PDA M_1 and then we prove that both accept L.

Step 1 : Construction of PDA M_2 based on given PDA M_1

Σ, Γ, initial state q_0, and initial symbol on the stack Z_0 are same for both PDAs. We add a new state with given PDA M_1. All final states of PDA M_1 are converted into non-final states.

So, $Q_2 = Q_1 \cup \{E\}$ (where E is new added state)

The transition function δ_2 contains all the transitions of given PDA M_1 and additional transitions (R_2 *and* R_3) defined as follows :

$R_1 : \delta_2(q, a, Z) = \delta_1(q, a, Z)$ for all (q, a, Z) in $Q_1 \times (\Sigma \cup \{\in\}) \times \Gamma_1$ (the original transitions of M_1),

$R_2 : \delta_2(p, \in, \alpha) = \{(E, \alpha)\}$ for all $p \in F$ and $\alpha \in \Gamma^*$, and

$R_3 : \delta_2(E, \in, \alpha) = \{(E, \in)\}$ for all $\alpha \in \Gamma^*$ and $E \in Q_2$

By R_1, PDA M_2 uses all the transitions of M_1 and reaches the final state if acceptability is there. By R_2, M_2 reaches state E and after reaching state E, by R_3, erases all the stack symbols. R_3 provides a loop incase of stack is not empty. The block diagram is shown in below figure.

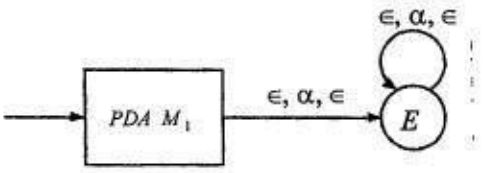


FIGURE : Block diagram of PDA M_2

Step 2 : The language accepted by PDA M_1 and PDA M_2

The behaviors of PDA M_1 and PDA M_2 are the same except the two $\in$ - moves defined in R_2 and R_3.

Let string $w \in L$ and accepted by PDA M_1, then

$(q_0, w, Z_0) \vdash_{M_1}^{*} (q_f, \in, \alpha)$, where $q_f \in F$ and $\alpha \in \Gamma^*$ **(Result 1)**

For PDA M_2, the initial ID is (q_0, w, Z_0).

So, $(q_0, w, Z_0) \vdash_{M_1}^{*} (q_f, \in, \alpha)$, where $q_f \in Q_2$ and $\alpha \in \Gamma^*$ (By R_1 and Result 1)

$\vdash_{M_1}^{*} (E, \in, \alpha)$ (By R_2)

$\vdash_{M_1}^{*} (E, \in, \in)$ (By R_3)

Thus, if M_1 accepts w, then M_2 also accepts it.

It means, $L(M_2) \subseteq L(M_1)$ **(Result 2)**

Let string $w \in L$ and accepted by M_2, then

For PDA M_2, the initial ID is (q_0, w, Z_0). So

$(q_0, w, Z_0) \vdash_{M_1}^{*} (q_f, \in, \alpha)$ for some $q_f \in Q_2$ and $\alpha \in \Gamma^*$ **(Result 3)**

$\vdash_{M_1}^{*} (E, \in, \alpha)$ (By R_2)

$\vdash_{M_1}^{*} (E, \in, \in)$ (By R_3)

For M_1, the initial ID is (q_0, w, Z_0).

So, $(q_0, w, Z_0) \vdash_{M_1}^{*} (q_f, \in, \alpha)$, where $q_f \in Q_2$ and $\alpha \in \Gamma^*$ **(By Result 3)**

Thus, if M_2 accepts w, then M_1 also accepts it.

It means $L(M_1) \subseteq L(M_2)$ **(Result 4)**

Therefore, $L = L(M_2) = L(M_1)$ (From Result 2 and Result 4)

Hence, the statement of theorem is proved.

Example :

Consider a PDA $M_1 = (\{q_0, q_1, q_2\}, \{a,c\}, \{a, Z_0\}, \delta_1, q_0, Z_0, \{q_2\})$, convert it into PDA M_2 whose acceptance is by empty store. Check the acceptability of string $w = aacaa$.

The transition function δ_1 is defined as follows :

$\delta_1(q_0, a, Z_0) = \{(q_0, aZ_0)\}$,

$\delta_1(q_0, a, a) = \{(q_0, aa)\}$,

$\delta_1(q_0, c, a) = \{(q_1, a)\}$

$\delta_1(q_1, a, a) = \{(q_1, \in)\}$ and

$\delta_1(q_1, \in, Z_0) = \{(q_2, Z_0)\}$

Solution :

Following moves are carried out by M_1 in order to check acceptability of string $w = aacaa$

$$(q_0, aacaa, Z_0) \vdash (q_0, acaa, aZ_0)$$

$$\vdash (q_0, caa, aaZ_0)$$

$$\vdash (q_1, aa, aaZ_0)$$

$$\vdash (q_1, a, aZ_0)$$

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

$$\vdash (q_2, \in, Z_0)$$

Hence, $(q_0, aacaa, Z_0) \overset{*}{\underset{M_1}{\vdash}} (q_2, \in, Z_0)$

Therefore, PDA M_1 accepts the string $aacaa$.

The transition function δ_2 for M_2 is defined as follows :

$\delta_2(q_0, a, Z_0) = \{(q_0, aZ_0)\}$ (Using R_1),

$\delta_2(q_0, a, a) = \{(q_0, aa)\}$ (Using R_1),

$\delta_2(q_0, c, a) = \{(q_1, a)\}$ (Using R_1),

$\delta_2(q_1, a, a) = \{(q_1, \in)\}$ (Using R_1)

$\delta_2(q_1, \in, Z_0) = \{(q_2, Z_0)\}$ (Using R_1)

$\delta_2(q_2, \in, a) = \{(E, a)\}$ (Using R_2)

$\delta_2(q_2, \in, c) = \{(E, c)\}$ (Using R_2)

$\delta_2(q_2, \in, Z_0) = \{(E, Z_0)\}$ (Using R_2)

$\delta_2(E, \in, a) = \{(E, \in)\}$ (Using R_3)

$\delta_2(E, \in, c) = \{(E, \in)\}$ (Using R_3), and

$\delta_2(E, \in, Z_0) = \{(E, \in)\}$ (Using R_3)

Following moves are carried out by M_2 in order to check the acceptability of the string $w = aacaa$

$$(q_0, aacaa, Z_0) \vdash (q_0, acaa, aZ_0)$$
$$\vdash (q_0, caa, aaZ_0)$$
$$\vdash (q_1, aa, aaZ_0)$$
$$\vdash (q_1, a, aZ_0)$$
$$\vdash (q_1, \in, aZ_0)$$
$$\vdash (q_2, \in, Z_0)$$
$$\vdash (E, \in, Z_0)$$
$$\vdash (E, \in, \in)$$

Hence, $(q_0, aacaa, Z_0) \overset{*}{\underset{M_2}{\vdash}} (E, \in, \in)$

6.5 PUSHDOWN AUTOMATA AND CFL

6.5.1 PDA FROM CFG

It is quite easy to get a PDA from the context free grammar. This is possible only from a CFG which is in GNF. So, given any grammar, first obtain the grammar in GNF and then obtain PDA. The steps to be followed to convert a grammar to its equivalent PDA are shown below.

1. Convert the grammar into GNF
2. Let q_0 be the start state and Z_0 is the initial symbol on the stack. Without consuming any input, push the start symbol S onto the stack and change the state to q_1. The transition for this can be

$$\delta(q_0, \in, Z_0) = (q_1, SZ_0)$$

3. For each production of the form

$$A \rightarrow a\alpha$$

introduce the transition $\delta(q_1, a, A) = (q_1, \alpha)$

4. Finally, in state q_1, without consuming any input, change the state to q_f which is an accepting state. The transition for this can be of the form

$$\delta(q_1, \in, Z_0) = (q_f, Z_0)$$

Example 1 : For the grammar

S → aABC
A → aB | a
B → bA|b
C → a

Obtain the corresponding PDA

Solution :

Let q_0 be the start state and Z_0 the initial symbol on the stack.

Step 1 : Push the start symbol S on to the stack and change the state to q_1. The transition for this can be of the form

$$\delta(q_0, \in, Z_0) = (q_1, SZ_0)$$

Step 2 : For each production $A \rightarrow a\alpha$ introduce the transition

$$\delta(q_1, a, A) = (q_1, \alpha)$$

This can be done as shown below.

Production			Transition
S	→	aABC	$\delta(q_1, a, S) = (q_1, ABC)$
A	→	aB	$\delta(q_1, a, A) = (q_1, B)$
A	→	a	$\delta(q_1, a, A) = (q_1, \in)$
B	→	bA	$\delta(q_1, b, B) = (q_1, A)$
B	→	b	$\delta(q_1, b, B) = (q_1, \in)$
C	→	a	$\delta(q_1, a, C) = (q_1, \in)$

Step 3 : Finally in state q_1, without consuming any input change the state to q_f which is an accepting state

$$\text{i. e., } \delta(q_1, \in, Z_0) = (q_f, Z_0)$$

So, the PDA M is given by $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$

where $Q = \{q_0, q_1, q_f\}$; $\Sigma = \{a, b\}$; $\Gamma = \{S, A, B, C, Z_0\}$

δ : is shown below.

$$\delta(q_0, \in, Z_0) = (q_1, SZ_0)$$
$$\delta(q_1, a, S) = (q_1, ABC)$$
$$\delta(q_1, a, A) = (q_1, B)$$
$$\delta(q_1, a, A) = (q_1, \in)$$
$$\delta(q_1, b, B) = (q_1, A)$$
$$\delta(q_1, b, B) = (q_1, \in)$$
$$\delta(q_1, a, C) = (q_1, \in)$$
$$\delta(q_1, \in, Z_0) = (q_f, Z_0)$$

$q_0 \in Q$ is the start state of machine ; $Z_0 \in \Gamma$ is the initial symbol on the stack.

$F = \{q_f\}$ is the final state

Note that the terminals grammar G will be input symbols in PDA and the non - terminals will be the stack symbols in PDA.

The derivation from the grammar is shown below

$$\begin{aligned} S &\Rightarrow aABC \\ &\Rightarrow aaBBC \\ &\Rightarrow aabBC \\ &\Rightarrow aabbC \\ &\Rightarrow aabba \end{aligned}$$

The string aabba is derived from the start symbol S. The same string should be accepted by PDA also. The moves made by the PDA are shown below.

Initial ID			
$(q_0, aabba, Z_0)$	$\vdash$	$(q_1, aabba, SZ_0)$	By Rule 1
	$\vdash$	$(q_1, abba, ABCZ_0)$	By Rule 2
	$\vdash$	$(q_1, bba, BBCZ_0)$	By Rule 3

$\vdash$ (q_1, ba, BCZ_0) By Rule 6

$\vdash$ (q_1, a, CZ_0) By Rule 6

$\vdash$ $(q_1, \in, Z_0)$ By Rule 7

$\vdash$ $(q_f, \in, Z_0)$ By Rule 8

(Final Configuration)

Since q_f is the final state and input string is $\in$ in the final configuration, the string **aabba** is accepted by the PDA.

Example 2 : Construct PDA M equivalent to the CFG $S \rightarrow 0BB, B \rightarrow 1S \mid 0S \mid 0$ and check whether 010000 is in $N(M)$ or not ?

Solution : Let PDA $M = (\{q_0\},\{a,b\},\{S,B,0,1\},\delta,q_0,S,\phi)$, δ be defined as follows :

$\delta(q_0,\in,S) = \{(q_0,0BB)\}$ (For the production $S \rightarrow 0BB$),

$\delta(q_0,\in,B)=\{(q_0,1S)\}$ (For the production $B \rightarrow 1S$),

$\delta(q_0,\in,B) =\{(q_0,0S)\}$ (For the production $B \rightarrow 0S$),

$\delta(q_0,\in,B) = \{(q_0,0)\}$ (For the production $B \rightarrow 0$),

$\delta(q_0,0,0) = \{(q_0,\in)\}$ (For terminal 0),

$\delta(q_0,1,1) = \{(q_0,\in)\}$ (For terminal 1)

For string $w = 010000, M$ has following moves :

$(q_0,010000\ ,S) \vdash (q_0,\underline{0}10000\ ,\underline{0}BB)$

$\vdash (q_0,10000\ ,\underline{B}B)$

$\vdash (q_0,\underline{1}0000\ ,\underline{1}SB)$

$\vdash (q_0,0000\ ,\underline{S}B)$

$\vdash (q_0,\underline{0}000\ ,\underline{0}BBB)$

$\vdash (q_0,000\ ,\underline{B}BB)$

$\vdash (q_0,\underline{0}00\ ,\underline{0}BB)$

$\vdash (q_0,00\ ,\underline{B}B)$

$$\vdash (q_0, \underline{0}0, \underline{0}B)$$

$$\vdash (q_0, 0, \underline{B})$$

$$\vdash (q_0, \underline{0}, \underline{0})$$

$$\vdash (q_0, \in, \in)$$

Hence, $010000 \in N(M)$.

6.5.2 Construction of CFG from Given PDA

As per our discussion, the CFG and PDA has a strong relationship. As we have seen in the previous section that we can construct a PDA from given CFG. Similarly we can obtain a CFG from given PDA.

Theorem : If $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, \phi)$ is a PDA which accepts the language L, then there is a CFG $G = (V, T, P, S)$ such that $L(G) \subseteq N(M) = L$.

Proof : $V = \{S\} \cup \{[p, Z, q] : p, q \in Q \text{ and } Z \in \Gamma\}$, T is same for both P includes the following productions :

$P_1 : S \to [q_0, Z_0, q]$ is in P for every $q \in Q$

$P_2 : [p, Z, q] \to a$ is in P for every $p, q \in Q, a \in T \cup \{\in\}$, and $Z \in \Gamma$ such that $\delta(p, a, Z) = \{(q, \in)\}$, and

$P_3 : [p, Z, q_{m+1}] \to a\,[q_1, B_1, q_2]\,[q_2, B_2, q_3] \ldots [q_m, B_m, q_{m+1}]$ is in P for every $p, q_i \in Q, a \in (T \cup \{\in\})$, and $Z, B_i \in \Gamma$, where $1 \le i \le m$ such that $\delta(p, a, Z) = \{(q_1, B_1 B_2 \ldots B_m)\}$.

If $m = 0$, then $[p, Z, q_1] \to a$

Example 1 : Consider the PDA $M = (\{q_0, q_1\}, \{a, b\}, \{a, Z_0\}, \delta, q_0, Z_0, \phi)$ accepting the language $L = \{a^n b^m a^n : m, n \ge 1\}$, which has the following transition function.

$\delta(q_0, a, Z_0) = \{(q_0, aZ_0)\}$,

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

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

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

$\delta(q_1, a, a) = \{(q_1, \in)\}$, *and*

$\delta(q_1, \in, Z_0) = \{(q_1, \in)\}$

Construct a CFG G which generates the same language.

Solution : Let CFG $G = (V, T, P, S)$, where

$V = \{[q_0, Z_0, q_0], [q_0, Z_0, q_1], [q_1, Z_0, q_0], [q_1, Z_0, q_1], [q_0, a, q_0], [q_0, a, q_1], [q_1, a, q_0], [q_1, a, q_1], S\}$

(**NOTE :** The number of states are two and stack symbols are two, then number of combination of these in triple form is $2 \times 2 \times 2 = 2^3 = 8$).

$\Sigma = \{a, b\}$, S is the start symbol, and P consists of following production rules :

Using construction rule P_1 :

$S \to [q_0, Z_0, q_0]$ and $S \to [q_0, Z_o, q_1]$

For transition $\delta(q_0, a, Z_0) = \{(q_0, aZ_0)\}$ **:**

$[q_0, Z_o, q_0] \to a[q_0, a, q_0][q_0, Z_o, q_0]$ (Using construction rule P_3)

$[q_0, Z_o, q_0] \to a[q_0, a, q_1][q_1, Z_0, q_0]$ (Using construction rule P_3)

$[q_0, Z_0, q_1] \to a[q_0, a, q_0][q_0, Z_0, q_1]$ (Using construction rule P_3)

$[q_0, Z_0, q_1] \to a[q_0, a, q_1][q_1, Z_0, q_1]$ (Using construction rule P_3)

For transition $\delta(q_0, a, a) = \{(q_0, aa)\}$ **:**

$[q_0, a, q_0] \to a[q_0, a, q_0][q_0, a, q_0]$ (Using construction rule P_3)

$[q_0, a, q_0] \to a[q_0, a, q_1][q_1, a, q_0]$ (Using construction rule P_3)

$[q_0, a, q_1] \to a[q_0, a, q_0][q_0, a, q_1]$ (Using construction rule P_3)

$[q_0, a, q_1] \to a[q_0, a, q_1][q_1, a, q_1]$ (Using construction rule P_3)

For transition $\delta(q_0, b, a) = \{(q_1, a)\}$ **:**

$[q_0, a, q_0] \to b[q_1, a, q_0]$ (Using construction rule P_3)

$[q_0, a, q_1] \to b[q_1, a, q_1]$ (Using construction rule P_3)

For transition $\delta(q_1, b, a) = \{(q_1, a)\}$ **:**

$[q_1, a, q_0] \to b[q_1, a, q_0]$ (Using construction rule P_3)

$[q_1, a, q_1] \to b[q_1, a, q_1]$ (Using construction rule P_3)

For transition $\delta(q_1, a, a) = \{(q_1, \in)\}$ **:**

$[q_1, a, q_1] \to a$ (Using construction rule P_2)

For transition $\delta(q_1, \in, Z_0) = \{(q_1, \in)\}$ **:**

$[q_1, Z_0, q_1] \to \in$ (Using construction rule P_2)

Example 2 : Construct CFG G for the PDA given as follows :

$\delta(q_0, a, Z_0) = \{(q_0, aZ_0)\}$

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

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

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

$\delta(q_0, \in, Z_0) = \{(q_0, \in)\}$

Solution :

Let $G = (V, T, P, S)$, $V = \{[q_0, Z_0, q_0], [q_0, a, q_0]\}$, $\Sigma = \{a, b, c\}$, P consists of following production rules :

$S \rightarrow [q_0, Z_0, q_0]$,

For transition $\delta(q_0, a, Z_0) = \{(q_0, aZ_0)\}$:

$[q_0, Z_0, q_0] \rightarrow a\,[q_0, a, q_0]\,[q_0, Z_0, q_0]$

For the transition $\delta(q_0, a, a) = \{(q_0, aa)\}$:

$[q_0, a, q_0] \rightarrow a[q_0, a, q_0]\,[q_0, a, q_0]$

For the transition $\delta(q_0, c, a) = \{(q_0, a)\}$:

$[q_0, a, q_0] \rightarrow c[q_0, a, q_0]$

For the transition $\delta(q_0, b, a) = \{(q_0, \in)\}$:

$[q_0, a, q_0] \rightarrow b$

For the transition $\delta(q_0, \in, Z_0) = \{(q_0, \in)\}$:

$[q_0, Z_0, q_0] \rightarrow \in$

Example 3 : Let $M = (\{q_0, q_1\}, \{a, b\}, \{c, Z_0\}, \delta, q_0, Z_0, \phi)$ is a PDA and δ is defined as follows:

$\delta(q_0, a, Z_0) = \{(q_0, cZ_0)\}$,

$\delta(q_0, a, c) = \{(q_0, cc)\}$,

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

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

$\delta(q_1, \in, c) = \{(q_1, \in)\}$,

$\delta(q_1, \in, Z_0) = \{(q_1, \in)\}$

Construct CFG G generating $N(M)$.

Solution :

Let $G = (V, T, P, S)$, where

1. V contains elements from the set

   $\{S,[q_0,c,q_0],[q_0,c,q_1],[q_0,Z_0,q_0],[q_0,Z_0,q_1],[q_1,c,q_1],[q_1,c,q_0],\ [q_1,Z_0,q_1],[q_1,Z_0,q_0],$

2. $\Sigma = \{a, b\}$,
3. S is the starting symbol, and
4. P includes the following production rules.

   For variable S

   $P_1: S \rightarrow [q_0, Z_0, q_0]$, and

   $P_2: S \rightarrow [q_0, Z_0, q_1]$

   For variable $[q_0,Z_0,q_0]$ using transition $\delta(q_0,a,Z_0) = \{(q_0,cZ_0)\}$

   $P_3[q_0,Z_0,q_0] \rightarrow a\,[q_0,c,q_0]\,[q_0,Z_0,q_0]$ and

   $P_4[q_0,Z_0,q_0] \rightarrow a\,[q_0,c,q_1]\,[q_1,Z_0,q_0]$

   For variable $[q_0,Z_0,q_1]$ $P_5[q_0,Z_0,q_1] \rightarrow a[q_0,c,q_0]\,[q_0,Z_0,q_1]$ and

   $P_6[q_0,Z_0,q_1] \rightarrow a\,[q_0,c,q_1][q_1,Z_0,q_1]$

   For variable $[q_0,c,q_0]$ using transition $\delta(q_0,a,c) = \{(q_0,cc)\}$:

   $P_7[q_0,c,q_0] \rightarrow a[q_0,c,q_0][q_0,c,q_0]$, and

   $P_8[q_0,c,q_0] \rightarrow a[q_0,c,q_1][q_1,c,q_0]$

   For variable $[q_0,c,q_1]$:

   $P_9:[q_0,c,q_1] \rightarrow a[q_0,c,q_0][q_0,c,q_1]$ and

   $P_{10}:[q_0,c,q_1] \rightarrow a[q_0,c,q_1]\,[q_1,c,q_1]$

   For variable $[q_0,c,q_1]$ using transition $\delta(q_0,b,c) = \{(q_1,\in)\}$:

   $P_{11}[q_0,c,q_1] \rightarrow b$

   For variable $[q_1,Z_0,q_1]$ using transition $\delta(q_1,\in,Z_0) = \{(q_1,\in)\}$:

   $P_{12}[q_1,Z_0,q_1] \rightarrow \in$

   For variable $[q_1,c,q_1]$ using transition $\delta(q_1,\in,c) = \{(q_1,\in)\}$:

   $P_{13}[q_1,c,q_1] \rightarrow \in$

   For variable $[q_1,c,q_1]$ using transition $\delta(q_1,b,c) = \{(q_1,\in)\}$:

   $P_{14}[q_1,c,q_1] \rightarrow b$

We have no production for the variable $[q_1, Z_0, q_0]$ and $[q_1, c, q_0]$. So, these are discarded with their associated productions.

So, now we have following productions.

$P_1: S \rightarrow [q_0, Z_0, q_0]$,

$P_2: S \rightarrow [q_0, Z_0, q_1]$,

$P_3: [q_0, Z_0, q_0] \rightarrow a[q_0, c, q_0][q_0, Z_0, q_0]$,

$P_5: [q_0, Z_0, q_1] \rightarrow a[q_0, c, q_0][q_0, Z_0, q_1]$,

$P_6: [q_0, Z_0, q_1] \rightarrow a[q_0, c, q_1][q_1, Z_0, q_1]$,

$P_7: [q_0, c, q_0] \rightarrow a[q_0, c, q_0][q_0, c, q_0]$,

$P_9: [q_0, c, q_1] \rightarrow a[q_0, c, q_0][q_0, c, q_1]$,

$P_{10}: [q_0, c, q_1] \rightarrow a[q_0, c, q_1][q_1, c, q_1]$,

$P_{11}: [q_0, c, q_1] \rightarrow b$,

$P_{12}: [q_1, Z_0, q_1] \rightarrow \in$,

$P_{13}: [q_1, c, q_1] \rightarrow \in$,

$P_{14}: [q_1, c, q_1] \rightarrow b$.

In Production P_3, and P_7, variables $[q_0, Z_0, q_0]$ and $[q_0, c, q_0]$ have right and left recursion respectively but have no terminating production i.e. no terminal from these variables. So, all the productions that include these variables including P_3 and P_7 are discarded.

Now, we have following productions included in P:

$P_2: S \rightarrow [q_0, Z_0, q_1]$,

$P_6: [q_0, Z_0, q_1] \rightarrow a[q_0, c, q_1][q_1, Z_0, q_1]$,

$P_{10}: [q_0, c, q_1] \rightarrow a[q_0, c, q_1][q_1, c, q_1]$,

$P_{11}: [q_0, c, q_1] \rightarrow b$,

$P_{12}: [q_1, Z_0, q_1] \rightarrow \in$,

$P_{13}: [q_1, c, q_1] \rightarrow \in$,

$P_{14}: [q_1, c, q_1] \rightarrow b$.

Theorem : Let L_1 be a context - free language and L_2 be a regular language. Then show that $L_1 \cap L_2$ is context - free.

Proof :

Let $M_1 = (Q, \Sigma, \Gamma, \delta_1, q_0, Z, F_1)$ be an NPDA which accepts L_1 and $M_2 = (P, \Sigma, \Gamma, \delta_2, q_0, F_2)$ be a DFA that accepts L_2. We construct a pushdown automaton $M = (\hat{Q}, \Sigma, \Gamma, \hat{\delta}, \hat{q}_0, Z, \hat{F})$ which simulates the parallel action of M_1 and M_2, whenever a symbol is read from the input string, $\hat{M}$ simultaneously executes the move M_1 and M_2.

Let $\hat{Q} = Q \times P$, $\hat{q}_0 = (q_0, p_0)$, $\hat{F} = F_1 \times F_2$

and define $\hat{\delta}$ such that, $((q_k, p_2), x) \in \hat{\delta}((q_i, p_j), a, b)$, if and only if,

$(q_k, x) \in \delta_1(q_i, a, b)$ and $\delta_2(p_j, a) = p_l$

In this, we also require that if $a = \in$, then $p_j = p_l$. In other words, the states of $\hat{M}$ are labeled with pairs (q_i, p_j), representing the respective states in which M_1 and M_2 can be after reading a certain input string. It is a straightforward induction argument to show that,

$((q_0, p_0), w, Z) \overset{*}{\vdash} ((q_r, p_s), x)$,

with $q_r \in F_1$ and $p_s \in F_2$

if and only if, ${}^{(q_0, w, z)}M_1^{(q_r, x)}$ and $\hat{\delta}(p_0, w) = p_s$.

Therefore, a string is accepted by M' if and only if it is accepted by M_1 and M_2 that is, if it is in $L(M_1) \cap L(M_2) = L_1 \cap L_2$.