6 Push Down Automata

Syllabus

Push Down Automata : Deterministic (single stack) PDA, Equivalence between PDA and CFG.

6.1 Introduction to Push Down Automata (PDA)

Informally, pushdown automata can be viewed as finite automata with stack. An added stack provides memory and increases the capability of the machine.

A pushdown automata can do the followings :

1. Read input symbol [as in case of FA]
2. Perform stack operations.
   - 2.1 Push operation
   - 2.2 Pop operation
   - 2.3 Check empty condition of a stack through an initial stack symbol.
   - 2.4 Read top symbol of stack without a pop.
3. Make state changes.

PDA is more powerful than FA. A context-free language (CFL) can be recognized by a PDA. Only a subset of CFL that are regular can be recognized by finite automata.

- A context free language can be recognized by PDA.
- For every context-free language, there exists a PDA.
- The language of PDA is a context-free language.

Example : A string of the form $a^n b^n$ can not be handled by a finite automata. But the same can be handled by a PDA.

- Any machine recognizing a string of the form $a^n b^n$, must keep track of a's [first half of $a^n b^n$] as number of b's must be equal to the number of a's.
- First half of the string can be remembered through a stack.
- As the machine reads the first half of $a^n b^n$, it remembers it by pushing it on top of the stack. As shown in Fig. 6.1.1, after reading first 5 a's, the stack contains 5 a's.
- While reading the second half of the input string consisting of b's, the machine pops out an 'a' from the stack for every 'b' as input.

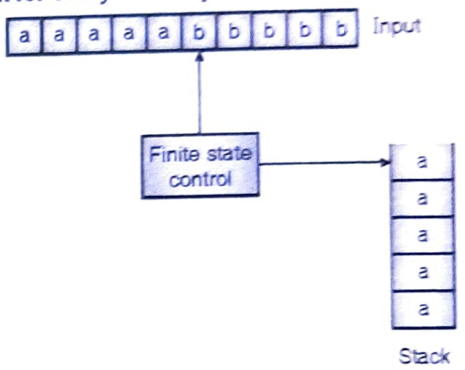


Fig. 6.1.1 : Stack after reading the first half of $a^5 b^5$

- After reading 5 b's, input will finish and the stack will become empty. This will indicate that the input string is of the form $a^n b^n$.
- The machine will have two states q_0 and q_1.

  State q_0 – while the machine is reading a's

  State q_1 – While the machine is reading b's.

  While in state q_1; an input 'a' is not allowed and hence there is a need for two states.
- A transition in PDA depends on :
  1. Current state
  2. Current input
  3. Top symbol of the stack.

A transition in PDA can be shown as a directed edge from the state q_i to q_j. While moving to state q_j, the machine can also perform stack operation. A transition edge from q_i to q_j should be marked with current input, current stack symbol (topmost symbol of the stack) and the stack operation. It is shown in Fig. 6.1.2.

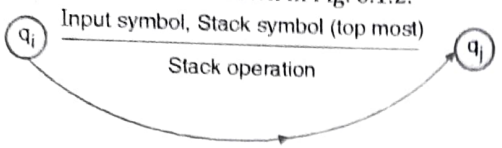


Fig. 6.1.2 : A transition from q_i to q_j

- A PDA uses three stack operations :
  1) **Pop** operation, it removes the top symbol from the stack.
  2) **Push** operation, it inserts a symbol onto the top of the stack.
  3) **Nop** operation, it does nothing to stack.
- The language $\{a^n b^n \mid n \geq 1\}$ can be accepted by the PDA of Fig. 6.1.3.

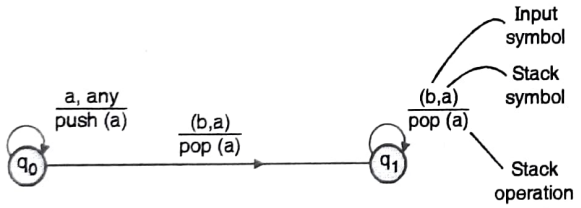


Fig. 6.1.3 : PDA for $a^n b^n$

- The state q_0 will keep track of the number of a's in an input string, by pushing symbol 'a' onto the stack for each input 'a'. A second state q_1 is used to pop an 'a' from the stack for each input symbol 'b'. Finally, after consuming the entire input the stack will become empty.

6.2 The Formal Definition of PDA

Q. Give formal definition of a Push Down Automata (PDA). MU - May 14, May 15, Dec. 18, 5 Marks

A pushdown automata M is defined as 7-tuple :

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

Where, Q = The set of states

Σ = Input alphabet

Γ = Stack symbols

δ = The transition function is a transition form $Q \times (\Sigma \cup \varepsilon) \times \Gamma$ to $Q \times \Gamma^*$

q_0 = $q_0 \in Q$ is the initial state

F = $F \subseteq Q$ is the set of final states

z_0 = An initial stack symbol

Transition function in detail

A transition function is a mapping from

$$Q \times (\Sigma \cup \varepsilon) \times \Gamma \text{ to } Q \times \Gamma^*$$

Where

$Q \times (\Sigma \cup \varepsilon) \times \Gamma$ implies that a transition is based on :

1. Current state
2. Next input (including ε)
3. Stack symbol (topmost)

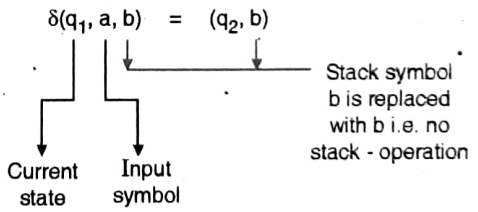
and

$Q \times \Gamma^*$ implies that :

1. The next state could be any state belonging to Q.
2. It can perform push, pop or no-operation (NOP) on stack.

In general, we can have the following types of transition behaviours.

1. Read input with no-operation on stack



- Left hand side of the transition, $\delta(q_1, a, b)$ implies :

  q_1 is the current state,

  a is the current input,

  b is the stack symbol (topmost).
- Right hand side of the transition, (q_2, b) implies that

  [The machine enters a new state q_2. The top symbol of the stack, which is b is replaced with b.]

Thus, the transition $\delta(q_1, a, b) = (q_2, b)$ does not modify a stack.

2. Pop operation

$\delta(q_1, a, b) = (q_2, \varepsilon)$

Stack symbol (top most) is replaced with ε

The above transition will erase the stack symbol (topmost). Replacing, b with ε amounts to erasing b from the stack top.

3. Push operation

$\delta(q_1, a, b) = (q_2, ab)$

Replacing the stack symbol (top most) b with ab will push a symbol a onto the stack

The above transition will perform a push operation. It will push 'a' onto the stack. Replacing 'b' with 'ab' amounts to push 'a' onto the stack.

The three operations for stack are shown in Table 6.2.1.

Table 6.2.1 : Transitions rules for stack operations

Stack before transition (assumed)	Transition rule	Stack after transition
1. b b a b z_0 ← top	$\delta(q_1, a, b) = (q_2, b)$	b b a b z_0 ← top; No - operation
2. b b a b z_0 ← top	$\delta(q_1, a, b) = (q_2, \varepsilon)$	b a b z_0 ← top; Pop - operation
3. b b a b z_0 ← top	$\delta(q_1, a, b) = (q_2, ab)$	a b b a b z_0 ← top; Push - operation

Example : Transition function for a PDA accepting a language

$L = \{\omega \in \{a, b\}^* \mid \omega \text{ is of the } a^n b^n \text{ with } n \geq 1\}$,

can be given in various forms :

1. Using a set of rules.
2. Using graphical representation (transition diagram)

Using a set of equations

$\delta(q_0, a, z_0) = (q_0, az_0)$

[First a of $a^n b^n$ is pushed onto the stack]

$\delta(q_0, a, a) = (q_0, aa)$ [Subsequent a's of $a^n b^n$ are pushed one by one onto the stack.]

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

[On Seeing the first b, the machine will make a move to q_1 with a POP]

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

[For each subsequent b, a POP operation is performed]

$\delta(q_1, \varepsilon, z_0) = (q_2, z_0)$ [After the end of input string, machine will transit to a final state]

Note : Failure transitions are not shown as it is a standard practice.

The transition rules mentioned above can be written in two different forms.

Form 1		Form 2
$\delta(q_0, a, z_0)$	$= (q_0, a\, z_0)$	$\Rightarrow ((q_0, az_0), (q_0, az_0))$
$\delta(q_0, a, a)$	$= (q_0, aa)$	$\Rightarrow ((q_0, a, a), (q_0, aa))$
$\delta(q_0, b, a)$	$= (q_1, \varepsilon)$	$\Rightarrow ((q_0, b, a), (q_1, \varepsilon))$
$\delta(q_1, b, a)$	$= (q_1, \varepsilon)$	$\Rightarrow ((q_1, b, a), (q_1, \varepsilon))$
$\delta(q_1, \varepsilon, z_0)$	$= (q_2, z_0)$	$\Rightarrow ((q_1, \varepsilon, z_0), (q_2, z_0))$

Using graphical representation (transition diagram)

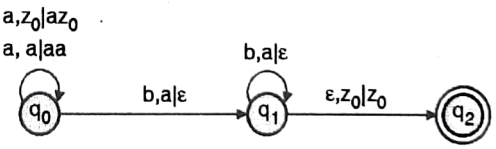
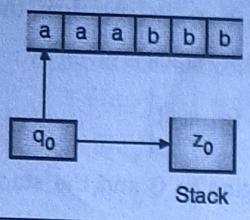
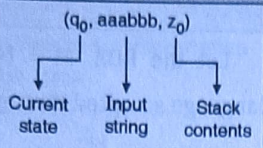
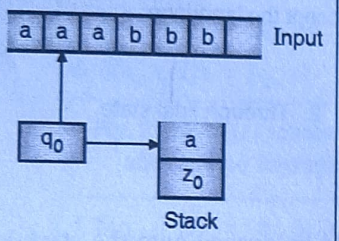
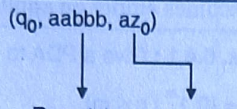
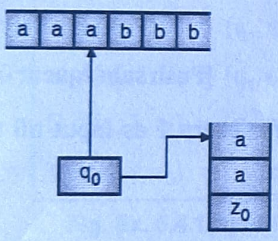
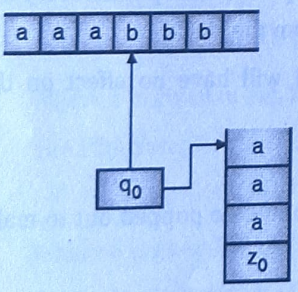
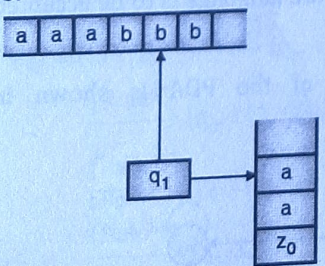


Fig. 6.2.1 : Transition diagram

6.3 Instantaneous Description of a PDA

Instantaneous behaviour is useful in showing the processing of a string by PDA. Processing of the input string a^3b^3 by the PDA of Fig. 6.2.1 is shown in Fig. 6.3.1.

Graphical representation (instantaneous behaviour)	Tuple format (instantaneous behaviour)
1. 	
2. 	
3. 	$(q_0, abbb, aaz_0)$
4. 	$(q_0, bbb, aaaz_0)$
5. 	(q_1, bb, aaz_0)

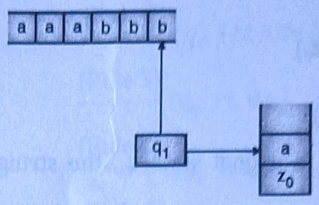
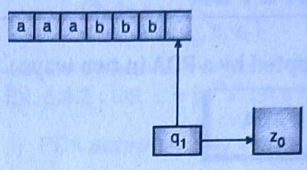
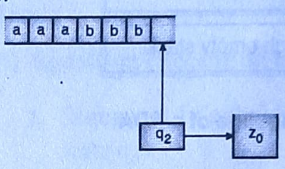
Graphical representation (instantaneous behaviour)	Tuple format (instantaneous behaviour)
6. 	(q_1, b, az_0)
7. 	(q_1, ε, z_0)
8. 	(q_2, ε, z_0)

Fig. 6.3.1 : Processing of a^3b^3 by the PDA of Fig. 6.2.1

The PDA for recognizing a string of the form a^nb^n can be described as:

M = ({ q_0,q_1,q_2}, {a,b}, {a,b,z_0}, δ, q_0, z_0, {q_2})

- Q, the set of states
- Σ, input symbols
- Γ, Stack symbols
- Transition function as given above
- Start state
- Initial stack symbols
- Set of final states

Ex. 6.3.1 : Suppose the PDA M = ({q_0, q_1}, {a, b, c}, {a, b, Z_0}, δ, q_0, Z_0, {q_1}) has the following transition function.

1. δ(q0, a, ε) = (q0, a)
2. δ(q0, b, ε) = (q0, b)
3. δ(q0, c, ε) = (q1, ε)
4. δ(q1, a, a) = (q1, ε)
5. δ(q1, b, b) = (q1, ε)

Show the acceptance of abbcbba by the above PDA through an instantaneous description.

Soln. :

$(q_0, abbcbba, z_0) \xrightarrow{\text{(Rule 1)}} (q_0, bbcbba, az_0)$

$\xrightarrow{\text{(Rule 2)}} (q_0, bcbba, baz_0)$

$\xrightarrow{\text{(Rule 2)}}$ $(q_0, cbba, bbaz_0)$

$\xrightarrow{\text{(Rule 3)}}$ $(q_1, bba, bbaz_0)$

$\xrightarrow{\text{(Rule 5)}}$ (q_1, ba, baz_0)

$\xrightarrow{\text{(Rule 5)}}$ (q_1, a, az_0)

$\xrightarrow{\text{(Rule 4)}}$ (q_1, ε, z_0)

Since, the machine is in a final state q_1, the string abbcbba is accepted.

6.4 The Language of a PDA

A language L can be accepted by a PDA in two ways :

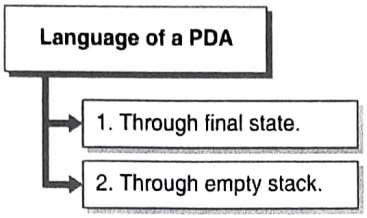


Fig. 6.4.1 : Language of a PDA

It is possible to convert between the two classes.

1. From final state to empty stack.
2. From empty stack to final state.

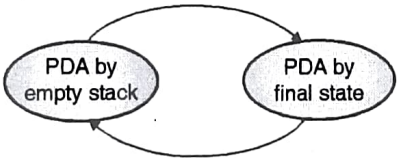


Fig. 6.4.1(a) : Equivalence of two PDAs

6.4.1 Acceptance by Final State

Q. Explain the concept, acceptance by final state of a Pushdown automata with suitable example.

MU - May 19, 2.5 Marks

Let the PDA, $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ then the language accepted by M through a final state is given by :

$$L(M) = \left\{ w \mid (q_0, w, z_0) \overset{*}{\underset{M}{\Rightarrow}} (q_1, \varepsilon, \alpha) \right\}$$

Where the state $q_1 \in F$. α, the final contents of the stack are irrelevant as a string is accepted through a final state.

6.4.2 Acceptance by Empty Stack

Q. Explain the concept, acceptance by empty stack of a Pushdown automata with suitable example.

MU - May 19, 2.5 Marks

Let the PDA, $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, \phi)$ then the language accepted through an empty stack is given by :

$$L(M) = \left\{ w \mid (q_0, w, z_0) \overset{*}{\underset{M}{\Rightarrow}} (q_1, \varepsilon, \varepsilon) \right\}$$

Where q_1 is any state belonging to Q and the stack becomes empty on application of input string w.

Ex. 6.4.1 : Give a PDA to accept the language

$L = \{0^n 1^m \mid n \leq m\}$

1. Through empty stack. 2. Through final state.

Soln. :

Algorithm :

1. Sequence of 0's should be pushed onto the stack in state q_0.

   $\delta(q_0, 0, z_0) = (q_0, 0z_0)$ [Push the first 0]

   $\delta(q_0, 0, 0) = (q_0, 00)$ [Push subsequent 0's]

2. A '0' should be popped for every 1 as input till the stack becomes empty.

   $\delta(q_0, 1, 0) = (q_1, \varepsilon)$

   [Pop on first 1 and change the state to q_1]

   $\delta(q_1, 1, 0) = (q_1, \varepsilon)$

   [Pop on subsequent 1 as input till every 0 is erased from the stack]

3. Subsequent 1's (m – n) will have no effect on the stack.

   $\delta(q_1, 1, z_0) = (q_1, z_0)$

4. Finally, the symbol Z_0 should be popped out to make the stack empty.

   $\delta(q_1, \varepsilon, z_0) = (q_1, \varepsilon)$

This step is required if the language is to be accepted through an empty stack.

Transition behaviour of the PDA is shown in Fig. Ex. 6.4.1(a to d)

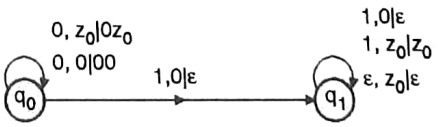


Fig. Ex. 6.4.1(a) : Transition diagram for acceptance through an empty stack

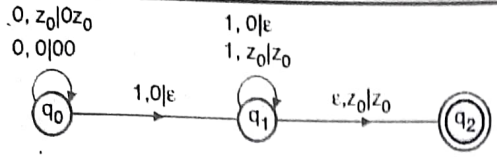


Fig. Ex. 6.4.1(b) : Transition diagram for acceptance through a final state

1. $\delta(q_0, 0, z_0) = (q_0, 0z_0)$
2. $\delta(q_0, 0, 0) = (q_0, 00)$
3. $\delta(q_0, 1, 0) = (q_1, \varepsilon)$
4. $\delta(q_1, 1, 0) = (q_1, \varepsilon)$
5. $\delta(q_1, 1, z_0) = (q_1, z_0)$
6. $\delta(q_1, \varepsilon, z_0) = (q_1, \varepsilon)$

Fig. Ex. 6.4.1(c) : State transition rules for acceptance through an empty stack

1. $\delta(q_0, 0, z_0) = (q_0, 0z_0)$
2. $\delta(q_0, 0, 0) = (q_0, 00)$
3. $\delta(q_0, 1, 0) = (q_1, \varepsilon)$
4. $\delta(q_1, 1, 0) = (q_1, \varepsilon)$
5. $\delta(q_1, 1, z_0) = (q_1, z_0)$
6. $\delta(q_1, \varepsilon, z_0) = (q_2, z_0)$

Fig. Ex. 6.4.1(d) : State transition rules for acceptance through a final state

The PDA accepting through an empty stack is given by :

$$M = \left(\{q_0, q_1\}, \{0, 1\}, \{0, 1, z_0\}, \delta, q_0, z_0, \phi\right)$$

Where δ is given in Fig. Ex. 6.4.1(c).

The PDA accepting through final state is given by :

$$M = \left(\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, z_0\}, \delta, q_0, z_0, \{q_2\}\right)$$

Where δ is given in Fig. Ex. 6.4.1(d).

Example : Processing of string 00111 by the PDA

Case I : Acceptance through empty stack.

$$(q_0, 00111, z_0) \xrightarrow{\text{(Rule 1)}} (q_0, 0111, 0z_0)$$
$$\xrightarrow{\text{(Rule 2)}} (q_0, 111, 00z_0)$$
$$\xrightarrow{\text{(Rule 3)}} (q_1, 11, 0z_0)$$
$$\xrightarrow{\text{(Rule 4)}} (q_1, 1, z_0)$$
$$\xrightarrow{\text{(Rule 5)}} (q_1, \varepsilon, z_0)$$
$$\xrightarrow{\text{(Rule 6)}} (q_1, \varepsilon, \varepsilon)$$

Case II : Acceptance through final state :

$$(q_0, 00111, z_0) \xrightarrow{\text{(Rule 1)}} (q_0, 0111, 0z_0)$$
$$\xrightarrow{\text{(Rule 2)}} (q_0, 111, 00z_0)$$
$$\xrightarrow{\text{(Rule 3)}} (q_1, 11, 0z_0)$$
$$\xrightarrow{\text{(Rule 4)}} (q_1, 1, z_0)$$
$$\xrightarrow{\text{(Rule 5)}} (q_1, \varepsilon, z_0)$$
$$\xrightarrow{\text{(Rule 6)}} (q_2, \varepsilon, z_0)$$

Ex. 6.4.2 : Let $L = \{a^m b^n \mid n < m\}$. Construct,

(i) PDA accepting L by empty store.

(ii) PDA accepting L by final state.

Soln. :

(i) Algorithm (Accepting L by empty store)

1. Sequence of a's should be pushed onto the stack in state q_0.
$$\delta(q_0, a, z_0) = (q_0, az_0)$$
$$\delta(q_0, a, a) = (q_0, aa)$$
2. An 'a' should be popped for every b as input till the end of input.
$$\delta(q_0, b, a) = (q_1, \varepsilon)$$
$$\delta(q_1, b, a) = (q_1, \varepsilon)$$
3. Additional m – n a's should be popped from the stack.
$$\delta(q_1, \varepsilon, a) = (q_1, \varepsilon)$$
4. Finally, the symbol z_0 should be popped out to make the state empty.
$$\delta(q_1, \varepsilon, z_0) = (q_1, \varepsilon)$$

Transition diagram is given in Fig. Ex. 6.4.2 and transition table in Table Ex. 6.4.2

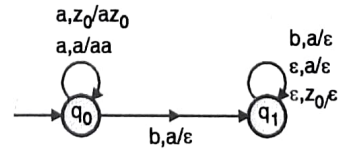


Fig. Ex. 6.4.2

Table Ex. 6.4.2

$\delta(q_0,a,z_0)$	=	(q_0,az_0)
$\delta(q_0,a,a)$	=	(q_0,aa)
$\delta(q_0,b,a)$	=	(q_1,ε)
$\delta(q_1,b,a)$	=	(q_1,ε)
$\delta(q_1,\varepsilon,a)$	=	(q_1,ε)
$\delta(q_1,\varepsilon,z_0)$	=	(q_1,ε)

$M = (\{q_0, q_1\}, \{a, b\}, \{a, b, z_0\}, \delta, q_0, z_0, \phi)$

ii) Accepting L through final state

Following modifications are required to accept through a final state.

In step 3, when the input ends with stack containing m-n a's, the machine should enter the final state.

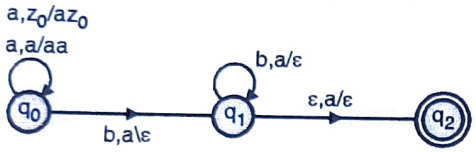


Fig. Ex. 6.4.2(a)

Table Ex. 6.4.2(a)

$\delta(q_0,a,z_0)$	=	(q_0,az_0)
$\delta(q_0,a,a)$	=	(q_0,aa)
$\delta(q_0,b,a)$	=	(q_1,ε)
$\delta(q_1,b,a)$	=	(q_1,ε)
$\delta(q_1,\varepsilon,a)$	=	(q_2,ε)

$M = (\{q_0, q_1\}, \{a, b\}, \{a, b, z_0\}, \delta, q_0, z_0, \{q_2\})$

Ex. 6.4.3 : Design a PDA for accepting the set of all strings over {a, b} with an equal number of a's and b's. The string should be accepted both by

(1) Final state (2) Empty stack.

Soln. :

- Stack will be used to store excess of a's over b's or excess of b's over a's out of input seen so far.
- The status of stack on input abaabbbbaa is shown in the Fig. Ex. 6.4.3.

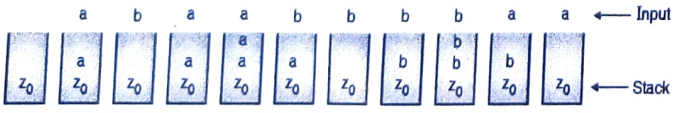
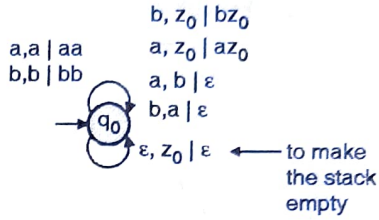


Fig. Ex. 6.4.3 : Stack preserving excess of a's over b's or b's over a's

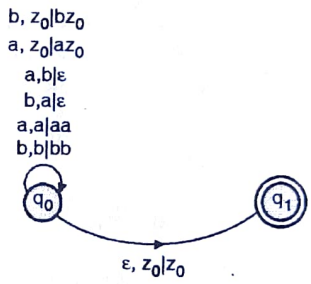
Algorithm : At any point of time, there could be one the six situations :

1. Stack contains z_0(topmost) and the input is a
   $\delta(q_0, a, z_0) = (q_0, az_0)$ [Extra a is pushed]
2. Stack contain z_0(topmost) and the input is b
   $\delta(q_0, b, z_0) = (q_0, bz_0)$ [Extra b is pushed]
3. Stack contains a(topmost) and the input is a.
   $\delta(q_0, a, a) = (q_0, aa)$ [Excess a's will increase by 1]
4. Stack contains a(topmost) and the input is b
   $\delta(q_0, b, a) = (q_0, \varepsilon)$ [Excess a's will decrease by 1]
5. Stack contains b(topmost) and the input is a.
   $\delta(q_0, a, b) = (q_0, \varepsilon)$ [Excess b's will decrease by 1]
6. Stack contains b(topmost) and the input is b.
   $\delta(q_0, b, b) = (q_0, bb)$ [Excess b's will increase by 1]

The PDA is shown in Fig. Ex. 6.4.3(a,b).



(a) Transition diagram for the PDA accepting through an empty stack



(b) Transition diagram for the PDA accepting through a final state

Fig. Ex. 6.4.3

$\delta(q_0, a, z_0) = (q_0, az_0)$	$\delta(q_0, a, z_0) = (q_0, az_0)$
$\delta(q_0, b, z_0) = (q_0, bz_0)$	$\delta(q_0, b, z_0) = (q_0, bz_0)$
$\delta(q_0, a, b) = (q_0, \varepsilon)$	$\delta(q_0, a, b) = (q_0, \varepsilon)$
$\delta(q_0, b, a) = (q_0, \varepsilon)$	$\delta(q_0, b, a) = (q_0, \varepsilon)$
$\delta(q_0, a, a) = (q_0, aa)$	$\delta(q_0, a, a) = (q_0, aa)$
$\delta(q_0, b, b) = (q_0, bb)$	$\delta(q_0, b, b) = (q_0, bb)$
$\delta(q_0, \varepsilon, z_0) = (q_0, \varepsilon)$	$\delta(q_0, \varepsilon, z_0) = (q_1, z_0)$

Fig. Ex. 6.4.3(c) : Transition rules for the PDA accepting through empty stack

Fig. Ex. 6.4.3(d) : Transition rules for the PDA accepting through final state

Fig. Ex. 6.4.3 (a) to (d) shows state transition behaviour of PDA of Example 6.4.3.

The PDA accepting through empty stack is given by :

$$M = (\{q_0\}\ \{a, b\}\ \{a, b, z_0\}, \delta, q_0, z_0, \phi)$$

Where, δ is given in Fig. Ex. 6.4.3(c).

The PDA accepting through final state is given by :

$$M = (\{q_0, q_1\}, \{a, b\}, \{a, b, z_0\}, \delta, q_0, z_0, \{q_1\}),$$

Where, δ is given in Fig. Ex. 6.4.3(d).

Ex. 6.4.4 : Design a PDA to accept $(ab)^n(cd)^n$.

Soln. : To solve this problem, we can take a stack symbol x. For every 'ab', one x will be pushed on top of the stack. After reading $(ab)^n$, the stack should contain n number of x's. These x's will be matched with $(cd)^n$. For every 'cd' one x will be popped.

The transitions for the PDA accepting through an empty stack are given in Fig. Ex. 6.4.4.

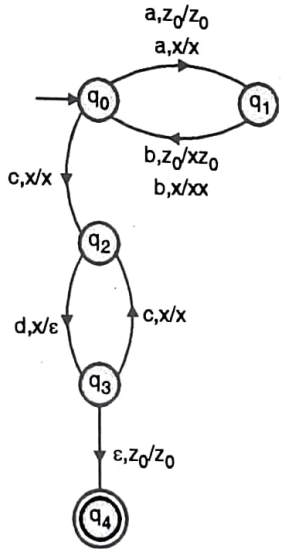


Fig. Ex. 6.4.4

– PDA accepts through the final state q_4.

The PDA $M = \{Q, \Sigma, \Gamma, \delta, q_0, z_0, F\}$

Where, $Q = \{q_0, q_1, q_2, q_3, q_4\}$

$\Sigma = \{a, b, c, d\}$

$\Gamma = \{x, z_0\}$

The transition function δ is given by,

$\delta(q_0, a, z_0) = (q_1, z_0)$

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

$\delta(q_1, b, z_0) = (q_0, x\, z_0)$

$\delta(q_1, b, x) = (q_0, xx)$

$\delta(q_0, c, x) = (q_2, x)$

$\delta(q_2, d, x) = (q_3, \varepsilon)$

$\delta(q_3, c, x) = (q_2, x)$

$\delta(q_2, \varepsilon, z_0) = (q_4, z_0)$

q_0 is initial state,

z_0 is initial stack symbol.

Set of final states $F = \{q_4\}$

Ex. 6.4.5 : Design a PDA to accept $(bdb)^n c^n$

Soln. : To solve this problem, we can take a stack symbol x. For every 'bdb', one x will be pushed on top of the stack. After reading $(bdb)^n$, the stack should contain n number of x's. These x's will be matched with c's.

The transitions for the PDA accepting through an empty stack are given in Fig. Ex.6.4.5.

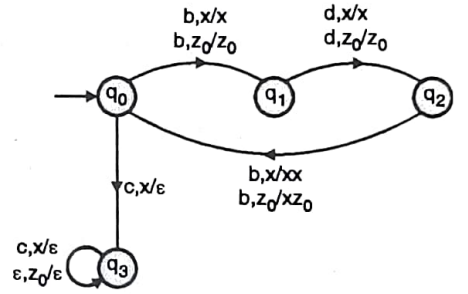


Fig. Ex. 6.4.5

– A cycle through $q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_0$ traces a group of bdb.

The PDA $M = \{Q, \Sigma, \Gamma, \delta, q_0, z_0, \phi\}$

Where, $Q = \{q_0, q_1, q_2, q_3\}$, $\Sigma = \{b, d, c\}$, $\Gamma = \{x, z_0\}$

q_0 is the initial state, z_0 is initial stack symbol.

The transition function δ is given by,

$\delta(q_0, b, z_0) = (q_1, z_0)$

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

$\delta(q_1, d, z_0) = (q_2, z_0)$

$\delta(q_1, d, x) = (q_2, x)$

$\delta(q_2, b, z_0) = (q_0, xz_0)$

$\delta(q_2, b, x) = (q_0, xx)$

$\delta(q_0, c, x) = (q_3, \varepsilon)$

$\delta(q_3, c, x) = (q_3, \varepsilon)$

$\delta(q_3, \varepsilon, z_0) = (q_3, \varepsilon)$ Accept through empty stack.

Ex. 6.4.6 : Construct pushdown automata for the following language :

L = {the set of strings over alphabet {a, b} with exactly twice as many a's as b's}

Soln. :

To solve this example, we can take three stack symbols :

$$\Gamma = (x, y, z_0)$$

Where, x stands for 2 a's,

y stands for a 'b',

z_0 is initial stack symbol.

- Stack will be used to store excess of x's over y's or excess of y's over x's.
- Since, a 'x' will be pushed onto the stack after 2 a's, after first 'a' the machine will transit to q_1 to remember that one 'a' has already come and the second 'a' in q_1 will complete 2 a's.
- Status of the stack and the state of the machine is shown in Fig. Ex. 6.4.6. Input applied to the machine is abaabbaaaaabaab.

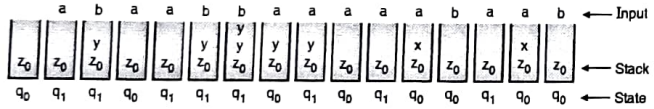


Fig. Ex. 6.4.6 : Status of stack for input abaabbaaaaabaab

Implementation of PDA using final state :

Let the PDA, M = $\{Q, \Sigma, \Gamma, \delta, q_0, z_0, F\}$

Where, $Q = \{q_0, q_1, q_2\}$, $\Sigma = \{a, b\}$, $\Gamma = \{x, y, z_0\}$

q_0 is initial state, z_0 is initial stack symbol.

$F = \{q_2\}$

Transition function δ is given by :

1. $\delta(q_0, a, z_0) = (q_1, z_0)$] First input could be either a or b
2. $\delta(q_0, b, z_0) = (q_0, yz_0)$]
3. $\delta(q_0, a, x) = (q_1, x)$ [goes to q_1 to remember first a of aa]
4. $\delta(q_0, a, y) = (q_1, y)$ [goes to q_1 to remember first a of 2 a's]
5. $\delta(q_0, b, x) = (q_0, \varepsilon)$ [Excess x is removed]
6. $\delta(q_0, b, y) = (q_0, yy)$ [Excess y increases by 1]
7. $\delta(q_1, a, z_0) = (q_0, xz_0)$ [Push a x for 2 a's]
8. $\delta(q_1, a, x) = (q_0, xx)$ [Push a x for 2 a's]
9. $\delta(q_1, a, y) = (q_0, \varepsilon)$ [Two a's will remove a 'y']
10. $\delta(q_1, b, z_0) = (q_1, yz_0)$ [Excess y is saved]
11. $\delta(q_1, b, x) = (q_1, \varepsilon)$ [Excess x is removed]
12. $\delta(q_1, b, y) = (q_1, yy)$ [Excess y increases by 1]
13. $\delta(q_0, \varepsilon, z_0) = (q_2, z_0)$ [goes to final state to accept the string]

Example : Processing of string abaabbaaaaabaab by the PDA

$(q_0, abaabbaaaaabaab, z_0) \xrightarrow{\text{(Rule 1)}}$

$(q_1, baabbaaaaabaab, z_0)$

$\xrightarrow{\text{(Rule 10)}}$ $(q_1, aabbaaaaabaab, yz_0)$

$\xrightarrow{\text{(Rule 9)}}$ $(q_0, abbaaaaabaab, z_0)$

$\xrightarrow{\text{(Rule 1)}}$ $(q_1, bbaaaaabaab, z_0)$

$\xrightarrow{\text{(Rule 10)}}$ $(q_1, baaaaabaab, yz_0)$

$\xrightarrow{\text{(Rule 12)}}$ $(q_1, aaaaabaab, yyz_0)$

$\xrightarrow{\text{(Rule 9)}}$ $(q_0, aaaabaab, yz_0)$

$\xrightarrow{\text{(Rule 4)}}$ $(q_1, aaabaab, yz_0)$

$\xrightarrow{\text{(Rule 9)}}$ $(q_0, aabaab, z_0)$

$\xrightarrow{\text{(Rule 1)}}$ $(q_1, abaab, z_0)$

$\xrightarrow{\text{(Rule 7)}}$ $(q_0, baab, xz_0)$

$\xrightarrow{\text{(Rule 5)}}$ (q_0, aab, z_0)

$\xrightarrow{\text{(Rule 1)}}$ (q_1, ab, z_0)

$\xrightarrow{\text{(Rule 7)}}$ (q_0, b, xz_0)

$\xrightarrow{\text{(Rule 5)}}$ (q_0, ε, z_0)

$\xrightarrow{\text{(Rule 13)}}$ (q_2, ε, z_0)

Ex. 6.4.7 : Give the transition table for PDA recognizing the following language

$L = \{ a^n x \mid n \geq 0 \text{ and } x \in \{a, b\}^* \text{ and } |x| \leq n \}$

Soln. :

Algorithm

- Sequence of initial a's should be pushed onto the stack in state q_0.
- For every symbol in x, an 'a' should be erased from the stack.

1. Push the first 'a' using the transition $\delta(q_0, a, z_0) = (q_0, az_0)$
2. Push subsequent a's using the transition $\delta(q_0, a, a) = (q_0, aa)$.
3. On the first 'b' as input, the machine will transit to q_1 with a pop operation using the transition $\delta(q_0, b, a) = (q_1, \varepsilon)$.
4. On subsequent a's or b's in x, an 'a' should be erased from the stack.

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

5. After the end of input string, contents of stack should be erased one by one.

$$\delta(q_1, \varepsilon, a) = (q_1, \varepsilon)$$
$$\delta(q_1, \varepsilon, z_0) = (q_1, \varepsilon)$$

The PDA is given by :

$$M = (\{q_0, q_1\}, \{a, b\}, \{a, z_0\}, \delta, q_0, z_0, \phi)$$

Where, δ is given below :

$$\delta(q_0, a, z_0) = (q_0, az_0)$$
$$\delta(q_0, a, a) = (q_0, aa)$$
$$\delta(q_0, \varepsilon, z_0) = (q_0, \varepsilon)$$

[To accept a null string]

$$\left.\begin{aligned}\delta(q_0, b, a) &= (q_1, \varepsilon)\\ \delta(q_1, a, a) &= (q_1, \varepsilon)\\ \delta(q_1, b, a) &= (q_1, \varepsilon)\end{aligned}\right]$$

erase, everything from the stack

$$\delta(q_1, \varepsilon, a) = (q_1, \varepsilon)$$
$$\delta(q_1, \varepsilon, z_0) = (q_1, \varepsilon)$$

Ex. 6.4.8 : Let $L = \{ a^n b^n c^m d^m \mid n, m \geq 1 \}$ find a PDA that accepts L.

Soln. :

Algorithm

1. Sequence of a's should be pushed onto the stack.
2. For every b as input, an 'a' should be erased from the stack.
3. Sequence of c's should be pushed onto the stack.
4. For every d as input, a 'c' should be erased from the stack.

The PDA is given by :

$$M = (\{q_0, q_1, q_2, q_3\}, \{a, b, c, d\}, \{a, c, z_0\}, \delta, q_0, z_0, \phi)$$

Where the transition function δ is given below :

$\delta(q_0, a, z_0) = (q_0, az_0)$ [Push the first a]

$\delta(q_0, a, a) = (q_0, aa)$ [Push remaining a's]

$\delta(q_0, b, a) = (q_1, \varepsilon)$ [Erase an 'a' on first b]

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

[Erase remaining a's on subsequent b's]

$\delta(q_1, c, z_0) = (q_2, cz_0)$ [First c is pushed]

$\delta(q_2, c, c) = (q_2, cc)$ [Subsequent c's are pushed]

$\delta(q_2, d, c) = (q_3, \varepsilon)$ [On first d, machine transits to q_3 with a pop]

$\delta(q_3, d, c) = (q_3, \varepsilon)$ [For every d, a 'c' is erased]

$\delta(q_3, \varepsilon, z_0) = (q_3, \varepsilon)$

[String is accepted through empty stack]

Ex. 6.4.9 : Let $L = \{ a^i b^j c^k \mid i, j, k \geq 0 \text{ and } i + j = k \}$

(i) Find a PDA accepting through final state.

(ii) Find a PDA accepting through empty stack.

Soln. :

Algorithm

1. For every input symbol 'a', a symbol x is pushed onto the stack.
2. For every input symbol 'b', a symbol x is pushed onto the stack.
3. For every input symbol 'c', x is erased from the stack.

(i) PDA accepting through final state is given by

$$M = (\{q_0, q_1, q_2, q_3\}, \{a, b, c\}, \{x, z_0\}, q_0, z_0, \delta, \{q_3\})$$

Where the transition function δ is :

1. $\delta(q_0, a, z_0) = (q_0, xz_0)$ [x is pushed for the first a]
2. $\delta(q_0, a, x) = (q_0, xx)$ [x is pushed for every subsequent a]
3. $\delta(q_0, b, z_0) = (q_1, xz_0)$ [First b without a's]

4. $\delta(q_0, b, x) = (q_1, xx)$ [First b after a's]
5. $\delta(q_1, b, x) = (q_1, xx)$ [Subsequent b's]
6. $\delta(q_1, c, x) = (q_2, \varepsilon)$ [x is erased for the first c]
7. $\delta(q_2, c, x) = (q_2, \varepsilon)$ [x is erased for subsequent c's]
8. $\delta(q_2, \varepsilon, z_0) = (q_3, z_0)$ [Accept through q_3]
9. $\delta(q_0, \varepsilon, z_0) = (q_3, z_0)$ [Accept a null string]

(ii) PDA accepting through empty stack is given by,

$$M = \left(\{q_0, q_1, q_2\}, \{a, b, c\}, \{x, z_0\}, q_0, z_0, \delta, \phi \right),$$

where the transition function δ is :

1. $\delta(q_0, a, z_0) = (q_0, xz_0)$
2. $\delta(q_0, a, x) = (q_0, xx)$
3. $\delta(q_0, b, z_0) = (q_1, xz_0)$
4. $\delta(q_0, b, x) = (q_1, xx)$
5. $\delta(q_1, b, x) = (q_1, xx)$
6. $\delta(q_1, c, x) = (q_2, \varepsilon)$
7. $\delta(q_2, c, x) = (q_2, \varepsilon)$
8. $\delta(q_2, \varepsilon, z_0) = (q_2, \varepsilon)$ [Stack is made empty]
9. $\delta(q_0, \varepsilon, z_0) = (q_0, \varepsilon)$ [Accept a null string]

Ex. 6.4.10 : Construct a PDA accepting $\{a^n b^m a^n \mid m, n \geq 1\}$ by null store. **MU - May 14, 10 Marks**

Soln. :

Algorithm

1. The sequence of a's should be pushed onto the stack in state q_0

$$\delta(q_0, a, z_0) = (q_0, az_0)$$
$$\delta(q_0, a, a) = (q_0, aa)$$

2. On first b, the machine moves to q_1 and remains there for b's. b's will have no effect on the stack.
3. For every 'a', an 'a' is erased from the stack.

The PDA accepting through empty stack is given by

$$M = \left(\{q_0, q_1, q_2\}, \{a, b\}, \{a, z_0\}, \delta, q_0, z_0, \phi \right)$$

Where the transition function δ is :

1. $\delta(q_0, a, z_0) = (q_0, az_0)$ [First 'a' is pushed]
2. $\delta(q_0, a, a) = (q_0, aa)$ [Subsequent a's are pushed]
3. $\delta(q_0, b, a) = (q_1, a)$ [Input symbols b's are skipped]
4. $\delta(q_1, b, a) = (q_1, a)$
5. $\delta(q_1, a, a) = (q_2, \varepsilon)$ [An a is erased on first a of last a's]
6. $\delta(q_2, a, a) = (q_2, \varepsilon)$ [An a is erased on subsequent a's of last a's]
7. $\delta(q_2, \varepsilon, z_0) = (q_2, \varepsilon)$ [Accepting through empty stack]

Ex. 6.4.11 : Design a PDA for accepting a language

$$L = \left\{ WcW^T \mid W \in \{a, b\}^* \right\}$$

Soln. : W^T stands for reverse of W. A string of the form WcW^T is an odd length palindrome with the middle character as c.

Algorithm

If the length of the string is 2n + 1, then the first n symbols should be matched with the last n symbols in the reverse order. A stack can be used to reverse the first n input symbols.

– Status of the stack and state of the machine is shown in Fig. Ex. 6.4.11. Input applied is abbcbba.

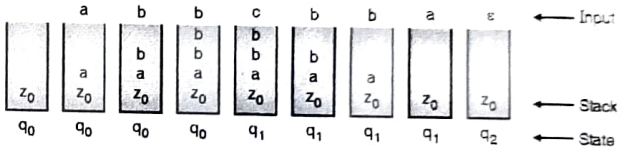


Fig. Ex. 6.4.11 : A PDA on input abbcbba

The PDA accepting through final state is given by

$$M = \left(\{q_0, q_1, q_2\}, \{a, b, c\}, \{a, b, z_0\}, \delta, q_0, z_0, \{q_2\} \right)$$

here the transition function δ is given below :

1. $\delta(q_0, a, \varepsilon) = (q_0, a)$
2. $\delta(q_0, b, \varepsilon) = (q_0, b)$

(1, 2) First n symbols are pushed onto the stack

3. $\delta(q_0, c, \varepsilon) = (q_1, \varepsilon)$ [State changes on c]
4. $\delta(q_1, a, a) = (q_1, \varepsilon)$
5. $\delta(q_1, b, b) = (q_1, \varepsilon)$

(4, 5) Last n symbols are matched with first n symbols in reverse order

6. $\delta(q_1, \varepsilon, z_0) = (q_2, z_0)$ [Accepted through final state]

A transition of the form $\delta(q_0, a, \varepsilon) = (q_0, a)$ implies that always push a, irrespective of stack symbol.

Ex. 6.4.12 : Design a PDA to check whether a given string over {a, b} ends in abb.

Soln. : Strings ending in abb form a regular language. A regular language is accepted by DFA. A PDA for a regular language can be constructed in two steps :

1. Design a DFA.
2. Convert DFA to PDA by appending no-stack-operation to every transition in DFA.

Step 1 : DFA for the given language is :

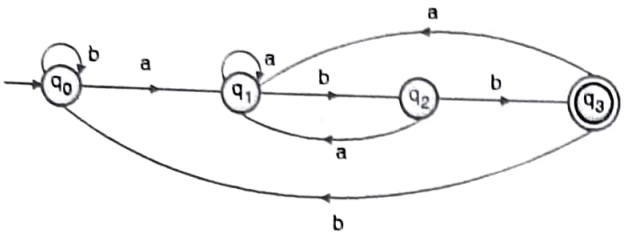


Fig. Ex. 6.4.12 : A DFA for string ending in abb

Step 2 : From DFA to PDA

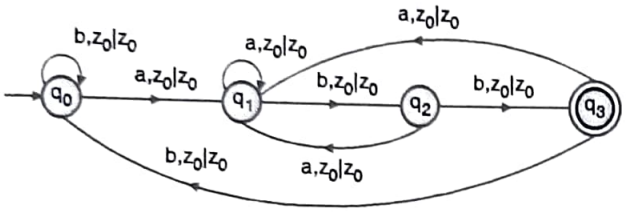


Fig. Ex. 6.4.12(a) : A PDA constructed from DFA

- In every transition of DFA, a stack operation z_0/z_0 is added. z_0/z_0 implies no-stack operation.

The PDA accepting strings ending in abb through final state is given by :

$$M = \left(\{q_0, q_1, q_2, q_3\}, \{a, b\}, \{z_0\}, \delta, q_0, z_0, \{q_3\} \right)$$

Where the transition function δ is given below,

$$\delta(q_0, b, z_0) = (q_0, z_0)$$
$$\delta(q_0, a, z_0) = (q_1, z_0)$$
$$\delta(q_1, a, z_0) = (q_1, z_0)$$
$$\delta(q_1, b, z_0) = (q_2, z_0)$$
$$\delta(q_2, a, z_0) = (q_1, z_0)$$
$$\delta(q_2, b, z_0) = (q_3, z_0)$$
$$\delta(q_3, a, z_0) = (q_1, z_0)$$
$$\delta(q_3, b, z_0) = (q_0, z_0)$$

Ex. 6.4.13 : Construct deterministic PDA to recognize $a^n abb^n$, $n > 0$ over {a, b} **MU - Dec. 16, 10 Marks**

Soln. :

The smallest string of the above language will be a^2b^2. The PDA can be constructed by pushing leading is on the stack and then matching than with the trailing b's.

The transitions rules for the PDA :

$\delta(q_0, a, z_0) = (q_1, az_0)$ [First 'a' is pushed]

$\delta(q_1, a, a) = (q_2, aa)$ [Second 'a' is pushed]

$\delta(q_2, a, a) = (q_2, aa)$ [Subsequent a's are pushed]

$\delta(q_2, b, a) = (q_3, \varepsilon)$ [First 'b' is matched]

$\delta(q_3, b, a) = (q_3, \varepsilon)$ [Subsequent b's are matched]

$\delta(q_3, \varepsilon, z_0) = (q_3, \varepsilon)$

↓

Accept through as empty stack

The PDA is given by,

$$M = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \{a_1, z_0\}, \$, q_0, z_0, \phi)$$

Ex. 6.4.14 : Design a PDA to accept the language $\{L = a^m b^m c^n \mid m, n, > = 1\}$ **MU - May 17, 10 Marks**

Soln. :

The transitions for PDA are :

1. $\delta(q_0, a, z_0) = (q_1, az_0)$
2. $\delta(q_1, a, a) = (q_1, aa)$
3. $\delta(q_1, b, a) = (q_2, \in)$
4. $\delta(q_2, b, a) = (q_2, \in)$
5. $\delta(q_2, c, z_0) = \delta(q_3, z_0)$
6. $\delta(q_3, c, z_0) = \delta(q_3, z_0)$
7. $\delta(q_3, \in, z_0) = \delta(q_3, \in)$

Accept through the empty stack.

Ex. 6.4.15 : Design a PDA for CFL that checks the well formedness of parenthesis i.e. the language L of all "balanced" string of two types of parenthesis say "( )" and "[ ]". Trace the sequence of moves made corresponding to input string (([ ])[ ]). **MU - Dec. 18, Dec. 19, 10 Marks**

Soln. :

The transition function of the PDA is given below :

1. $\delta(q_0, (, z_0) = (q_0, (z_0)$
2. $\delta(q_0, (, () = (q_0, (()$
3. $\delta(q_0, (, [) = (q_0, ([)$

Push the opening bracket '('

4. $\delta(q_0, [, z_0) = (q_0, [z_0)$
5. $\delta(q_0, [, [) = (q_0, [[)$
6. $\delta(q_0, [,]) = (q_0, [])$

Push the opening bracket '['

7. $\delta(q_0,), [) = (q_0, \varepsilon)$
8. $\delta(q_0,],]) = (q_0, \varepsilon)$

POP an opening bracket for a closing bracket.

9. $\delta(q_0, \varepsilon, z_0) = (q_f, z_0)$

Accept through a final state.

Simulation of PDA for the input string (([])[])

$(q_0, (([])[]), z_0) \xrightarrow{\text{Rule 1}} (q_0, ([])[]), (z_0)$

$\xrightarrow{\text{Rule 2}} (q_0, [])[]), ((z_0)$

$\xrightarrow{\text{Rule 5}} (q_0,])[]), [((z_0)$

$\xrightarrow{\text{Rule 8}} (q_0,)[]), ((z_0)$

$\xrightarrow{\text{Rule 7}} (q_0, []), (z_0)$

$\xrightarrow{\text{Rule 5}} (q_0,]), [(z_0)$

$\xrightarrow{\text{Rule 8}} (q_0,), (z_0)$

$\xrightarrow{\text{Rule 7}} (q_0, \varepsilon, z_0)$

$\xrightarrow{\text{Rule 9}} (q_f, \varepsilon, z_0)$

Ex. 6.4.16 : Construct a PDA for $L = \{a^n bc^m \mid n, m \geq 1 \text{ and } n < m\}$. **MU - May 19, 10 Marks**

Soln. :

The transitions for the PDA are given below :

1. $\delta(q_0, a, z_0) = (q_1, az_0)$
2. $\delta(q_1, a, a) = (q_1, aa)$
3. $\delta(q_1, b, a) = (q_2, \varepsilon)$
4. $\delta(q_2, c, a) = (q_3, \varepsilon)$
5. $\delta(q_3, c, a) = (q_3, \varepsilon)$
6. $\delta(q_3, c, z) = (q_4, \varepsilon)$

↓

accept through final state

6.5 Non-deterministic PDA (NPDA)

There are two types of push down automata :

1. DPDA (Deterministic PDA)
2. NPDA (Non-deterministic PDA)

A NPDA provides non-determinism to PDA.

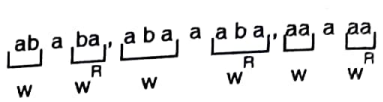
In a DPDA there is only one move in every situation. Where as, in case of NPDA there could be multiple moves under a situation.

- DPDA is less powerful than NPDA. Every context free language can not be recognized by a DPDA but it can be recognized by NPDA. The class of language a DPDA can accept lies in between a regular language and CFL. A palindrome can be accepted by NPDA but it can not be accepted by a DPDA.

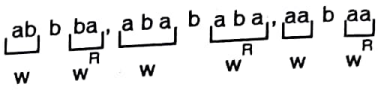
Ex. 6.5.1 : Design a PDA for detection of odd palindrome over {a, b}.

Soln. : An odd palindrome will be of the form

1. waw^R



2. wbw^R



If the length of w is n then a palindrome of odd length is :

First n characters are equal to the last n characters in reverse order with middle character as 'a' or 'b'.

Algorithm

There is no way of finding the middle position of a string by a PDA, therefore the middle position is fixed non-deterministically.

1. First n characters are pushed onto the stack, where n is non-deterministic.
2. The n characters on the stack are matched with the last n characters of the input string.
3. n is decided non-deterministically. Every character out of first n characters should be considered for two cases :
   (a) It is not the middle character – push the current character using the transition :

$\delta(q_0, a, \varepsilon) \Rightarrow (q_0, a)$

$\delta(q_0, b, \varepsilon) \Rightarrow (q_0, b)$

(b) It is a middle character - go for matching of second half with the first half.

$\delta(q_0, a, \varepsilon) \Rightarrow (q_1, \varepsilon)$

$\delta(q_0, b, \varepsilon) \Rightarrow (q_1, \varepsilon)$

The status of the stack and the state of the machine is shown in the Fig. Ex. 6.5.1. Input applied is ababa.

- Left child → current input is taken as the middle character
- Right child → current input is not a middle character.

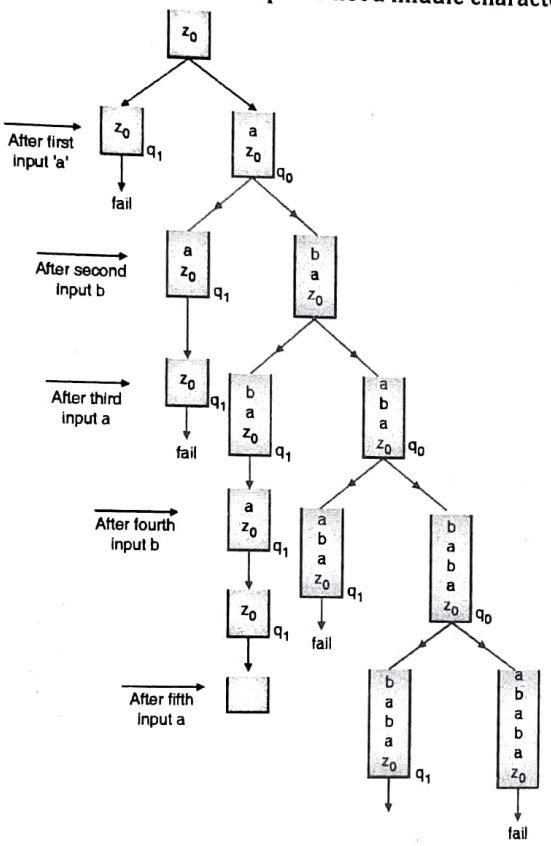


Fig. Ex. 6.5.1 : Processing of string by the PDA. String is taken as "ababa"

The transition table for the PDA is given below,

$\delta(q_0, a, \varepsilon) \Rightarrow \{(q_1, \varepsilon), (q_0, a)\}$

↓

ε - indicates that irrespective of the current stack symbol, perform the transition.

$\delta(q_0, b, \varepsilon) \Rightarrow \{(q_1, \varepsilon), (q_0, b)\}$

$\delta(q_1, a, a) \Rightarrow \{(q_1, \varepsilon)\}$

$\delta(q_1, b, b) \Rightarrow \{(q_1, \varepsilon)\}$

$\delta(q_1, \varepsilon, z_0) \Rightarrow \{(q_1, \varepsilon)\}$

[Accept through an empty stack]

Where, The set of states $Q = \{q_0, q_1\}$

The set input alphabet $\Sigma = \{a, b\}$

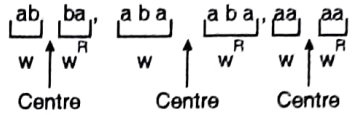
The set of stack symbols $\Gamma = \{a, b, z_0\}$

Starting state $= q_0$

Initial stack symbol $= z_0$

Ex. 6.5.2 : Design a PDA for detection of even palindrome over {a, b). **MU- May 15, 10 Marks**

Soln. : An even palindrome will be of the form ww^R



If the length of w is n then a palindrome of even length is :

First n characters are equal to the last n characters in the reverse order.

- The character immediately before the middle position will be identical to the character immediately after the middle position.

Algorithm

There is no way of finding the middle position by a PDA; therefore the middle position is fixed non-deterministically.

1. First n characters are pushed onto the stack. n is non-deterministic.
2. The n characters on the stack are matched with the last n characters of the input string.
3. n is decided non-deterministically. Every character out of first n characters, whose previous character is same as itself should be considered for two cases :

(a) It is first character of the second half.

Pop the current stack symbol using the transitions :

$\delta(q_0, a, a) \Rightarrow (q_1, \varepsilon)$

$\delta(q_0, b, b) \Rightarrow (q_1, \varepsilon)$

Must be identical

(b) It belongs to first half.

Push the current input

$\delta(q_0, a, \varepsilon) \Rightarrow (q_0, a)$

$\delta(q_0, b, \varepsilon) \Rightarrow (q_0, b)$

4. n is decided non-deterministically. Every character out of first n characters, whose previous character is not same as itself should be pushed onto the stack.

- Push the current symbol using the transitions :

$\delta(q_0, a, b) \Rightarrow (q_0, ab)$

$\delta(q_0, b, a) \Rightarrow (q_0, ba)$

The transition table for the PDA is given below :

$\delta(q_0, a, z_0) \Rightarrow \{(q_0, az_0)\}$

$\delta(q_0, b, z_0) \Rightarrow \{(q_0, bz_0)\}$

$\delta(q_0, a, a) \Rightarrow \{(q_0, aa)\ (q_1, \varepsilon)\}$

$\delta(q_0, a, b) \Rightarrow \{(q_0, ab)\}$

$\delta(q_0, b, a) \Rightarrow \{(q_0, ba)\}$

$\delta(q_0, b, b) \Rightarrow \{(q_0, bb), (q_1, \varepsilon)\}$

$\delta(q_1, a, a) \Rightarrow \{(q_1, \varepsilon)\}$

$\delta(q_1, b, b) \Rightarrow \{(q_1, \varepsilon)\}$

$\delta(q_1, \varepsilon, z_0) \Rightarrow \{(q_1, \varepsilon)\}$

[Accept through an empty stack]

Where,the set of states Q = $\{q_0, q_1\}$

the set of input symbols Σ = {a, b}

the set of stack symbols Γ = $\{a, b, z_0\}$

Starting state = q_0

Initial stack symbol = z_0

Ex. 6.5.3 : Design a PDA for detection of palindromes over {a, b}.

Soln. :

A palindrome will be of the form :

1. ww^R – even palindrome
2. waw^R
3. wbw^R– odd palindrome

If the length of w is n then a palindrome is :

First n characters are equal to the last n characters in the reverse order with the middle character as :

(1) a [For odd palindrome]

(2) b [For odd palindrome]

(3) ε [For even palindrome]

The transition table for the PDA is given below :

$\delta(q_0, a, z_0) \Rightarrow \{(q_1, z_0), (q_0, az_0)\}$

$\delta(q_0, b, z_0) \Rightarrow \{(q_1, z_0), (q_0, bz_0)\}$

$\delta(q_0, a, a) \Rightarrow \{(q_0, aa)\ (q_1, a), (q_1, \varepsilon)\}$

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

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

$\delta(q_0, b, b) \Rightarrow \{(q_0, bb), (q_1, b), (q_1, \varepsilon)\}$

$\delta(q_1, a, a) \Rightarrow \{(q_1, \varepsilon)\}$

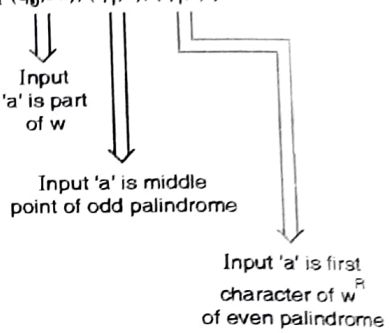
$\delta(q_1, b, b) \Rightarrow \{(q_1, \varepsilon)\}$

$\delta(q_1, \varepsilon, z_0) \Rightarrow \{(q_1, \varepsilon)\}$

[Accept through an empty stack]

Details of Important transitions

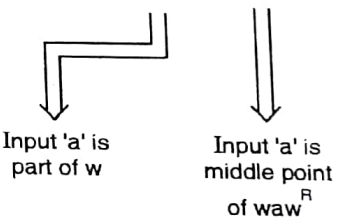
The transaction, $\delta(q_0,a,a) \Rightarrow \{ (q_0,aa), (q_1,a), (q_1,\varepsilon) \}$



The transition rule for $\delta(q_0, a, a)$, must consider the three cases :

1. Input 'a' is part of w of the palindrome.
2. Input 'a' is middle character of waw^R
3. Input 'a' is the first character of w^R.

The transaction, $\delta(q_0,a,b) \Rightarrow \{ (q_0, ab), (q_1, b) \}$



Ex. 6.5.4 : Construct PDA for the language

$L = \{a^i b^j c^k \mid i \neq j \text{ or } j \neq k\}$

Soln. : Algorithm

Machine has to non-deterministically figure out which of the two conditions will be satisfied by the string :

1. $i \neq j$, by entering the state q_1 on first input.
2. $j \neq k$, by entering the state q_2 on first input.

$\delta(q_0, a, z_0) = \{(q_1, aa), (q_2, z_0)\}$

In state q_1

1. Initial a's will be pushed.
2. For every input symbol b, an 'a' should be erased from the stack.
3. If a's and b's are not equal i.e. either some a's are left on the stack or some b's are still in the input then the string should be accepted.

In state q_2

1. Initial a's will be skipped.
2. For every input symbol b, a 'b' should be pushed.
3. For every input symbol c, a 'ab' should be erased from the stack.
4. If b's and c's are not equal i.e. either some b's are left on the stack or some c's are still in the input then the string should be accepted.

Transitions for PDA

$$\delta(q_0, a, z_0) = \{(q_1, aa), (q_2, z_0)\}$$
$$\delta(q_1, a, a) = \{(q_1, aa)\}$$
$$\delta(q_1, b, a) = \{(q_3, \varepsilon)\}$$
$$\delta(q_3, b, z_0) = \{(q_f, z_0)\}$$
$$\delta(q_3, c, a) = \{(q_f, \varepsilon)\}$$
$$\delta(q_3, b, a) = \{(q_3, \varepsilon)\}$$
$$\delta(q_2, a, z_0) = \{(q_2, \varepsilon)\}$$
$$\delta(q_2, b, z_0) = \{(q_4, bz_0)\}$$
$$\delta(q_4, b, b) = \{(q_4, bb)\}$$
$$\delta(q_4, c, b) = \{(q_4, \varepsilon)\}$$
$$\delta(q_4, \varepsilon, b) = \{(q_f, \varepsilon)\}$$
$$\delta(q_f, b, \varepsilon) = \{(q_f, \varepsilon)\}$$
$$\delta(q_f, c, \varepsilon) = \{(q_f, \varepsilon)\}$$
$$\delta(q_f, \varepsilon, \varepsilon) = \{(q_f, \varepsilon)\}$$

6.6 Push Down Automata and Context Free Language

The class of languages accepted by pushdown automata is exactly the class of context-free languages. The following three classes of languages are same :

1. Context Free Language defined by CFG.
2. Languages accepted by PDA by final state.
3. Languages accepted by PDA by empty stack.

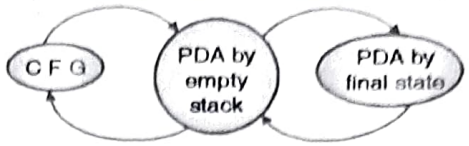


Fig. 6.6.1 : Equivalence of PDA and CFG

- It is possible to find a PDA for a CFG
- It is possible to find a CFG for a PDA.

6.6.1 Construction of PDA from CFG

From a given CFG G = (V, T, P, S), we can construct a PDA, M that simulates the leftmost derivation of G.

The PDA accepting L(G) by empty stack is given by :

$$M = (\{q\}, T, V \cup T, \delta, q, S, \phi) \quad [M \text{ is a PDA for } L(G)]$$

Where δ is defined by :

1. For each variable $A \in V$, include a transition,
   $\delta(q, \varepsilon, A) \Rightarrow \{(q, \alpha) \mid A \rightarrow \alpha \text{ is a production in } G\}$
2. For each terminal $a \in T$, include a transition
   $\delta(q, a, a) \Rightarrow \{(q, \varepsilon)\}$

Ex. 6.6.1 : Find a PDA for the given grammar

$S \rightarrow 0S1 \mid 00 \mid 11$

Soln. :

The equivalent PDA, M is given by :

$$M = (\{q\}, \{0, 1\}, \{0, 1, S\}, \delta, q, S, \phi),$$

where δ is given by :

$$\delta(q, \varepsilon, S) = \{(q, 0S1), (q, 00), (q, 11)\}$$
$$\delta(q, 0, 0) = \{(q, \varepsilon)\}$$
$$\delta(q, 1, 1) = \{(q, \varepsilon)\}$$

Ex. 6.6.2 : Convert the grammar

$S \rightarrow 0S1 \mid A$

$A \rightarrow 1A0 \mid S \mid \varepsilon$

to PDA that accepts the same language by empty stack.

Soln. :

Step 1 : for each variable $A \in V$, include a transition

$$\delta(q, \varepsilon, A) \Rightarrow \{(q, \alpha) \mid A \rightarrow \alpha \text{ is a production in } G\}$$
$$\therefore \quad \delta(q, \varepsilon, S) \Rightarrow \{(q, 0S1), (q, A)\}$$
$$\delta(q, \varepsilon, A) \Rightarrow \{(q, 1A0), (q, S), (q, \varepsilon)\}$$

Step 2 : For each terminal $a \in T$, include a transition

$$\delta(q, a, a) \Rightarrow (q, \varepsilon)$$
$$\therefore \quad \delta(q, 0, 0) = \{(q, \varepsilon)\}$$
$$\delta(q, 1, 1) = \{(q, \varepsilon)\}$$

Therefore, the PDA is given by :

$M = (\{q\}, \{0, 1\}, \{S, A, 0, 1\}, \delta, q, S, \phi)$

where δ is :

$\delta(q, \varepsilon, S) = \{(q, 0S1), (q, A)\}$

$\delta(q, \varepsilon, A) = \{(q, 1A0), (q, S), (q, \varepsilon)\}$

$\delta(q, 0, 0) = \{(q, \varepsilon)\}$

$\delta(q, 1, 1) = \{(q, \varepsilon)\}$

Ex. 6.6.3 : Let G be the grammar given by

$S \rightarrow aABB \mid aAA,\ A \rightarrow aBB \mid a,\ B \rightarrow bBB \mid A.$

Construct NPDA that accepts the language generated by this grammar.

Soln. :

The equivalent PDA, M is given by :

$M = (\{q\}, \{a, b\}, \{a, b, S, A, B\}, \delta, q, S, \phi)$

Where, δ is given by :

$\delta(q, \varepsilon, S) \Rightarrow \{(q, aABB), (q, aAA)\}$

$\delta(q, \varepsilon, A) \Rightarrow \{(q, aBB), (q, a)\}$ For each production in given grammar

$\delta(q, \varepsilon, B) \Rightarrow \{(q, bBB), (q, A)\}$

$\delta(q, a, a) \Rightarrow (q, \varepsilon)$

$\delta(q, b, b) \Rightarrow (q, \varepsilon)$ | For each terminal in T

Ex. 6.6.4 : Construct a PDA equivalent to the following CFG.

$S \rightarrow 0BB$

$B \rightarrow 0S \mid 1S \mid 0$

Test if 010^4 is in the language

Soln. : The equivalent PDA, M is given by

$M = (\{q\}, \{0, 1\}, \{0, 1, S, B\}, \delta, q, S, \phi)$,

where δ is given by

$\delta(q, \varepsilon, S) \Rightarrow \{(q, 0BB)\}$ | For each production in the given grammar

$\delta(q, \varepsilon, B) \Rightarrow \{(q, 0S), (q, 1S), (q, 0)\}$

$\delta(q, 0, 0) \Rightarrow \{(q, \varepsilon)\}$

$\delta(q, 1, 1) \Rightarrow \{(q, \varepsilon)\}$ | For each terminal

Acceptance of 010^4 by M :

$\delta(q, 010000, S) \vdash (q, 010000, 0BB)$ [$\delta(q, \varepsilon, S) = (q, 0BB)$]

$\vdash (q, 10000, BB)$ [$\delta(q, 0, 0) = (q, \varepsilon)$]

$\vdash (q, 10000, 1SB)$ [$\delta(q, \varepsilon, B) = (q, 1S)$]

$\vdash (q, 0000, SB)$ [$\delta(q, 1, 1) = (q, \varepsilon)$]

$\vdash (q, 0000, 0BBB)$ [$\delta(q, \varepsilon, S) = (q, 0BB)$]

$\vdash (q, 000, BBB)$ [$\delta(q, 0, 0) = (q, \varepsilon)$]

$\vdash (q, 000, 0BB)$ [$\delta(q, \varepsilon, B) = (q, 0)$]

$\vdash (q, 00, BB)$ [$\delta(q, 0, 0) = (q, \varepsilon)$]

$\vdash (q, 00, 0B)$ [$\delta(q, \varepsilon, B) = (q, 0)$]

$\vdash (q, 0, B)$ [$\delta(q, 0, 0) = (q, \varepsilon)$]

$\vdash (q, 0, 0)$ [$\delta(q, \varepsilon, B) = (q, 0)$]

$\vdash (q, \varepsilon, \varepsilon)$ [$\delta(q, 0, 0) = (q, \varepsilon)$]

Thus the string 010^4 is accepted by M using an empty stack.

$\therefore\ 010^4 \in L$

Ex. 6.6.5 : Design a PDA to recognize the language generated by the following grammar :

$S \rightarrow S + S \mid S * S \mid 4 \mid 2$

Show the acceptance of the input string

2 + 2 * 4 by this PDA.

Soln. :

The equivalent PDA, M is given by :

$M = (\{q\}, \{+, *, 4, 2\}, \{+, *, 4, 2, S\}, \delta, q, S, \phi)$

Where δ is given by :

$\delta(q, \varepsilon, S) \Rightarrow \{(q, S + S), (q, S * S), (q, 4), (q, 2)\}$ | For every production in G

$\delta(q, +, +) = \{(q, \varepsilon)\}$

$\delta(q, *, *) = \{(q, \varepsilon)\}$

$\delta(q, 2, 2) = \{(q, \varepsilon)\}$

$\delta(q, 4, 4) = \{(q, \varepsilon)\}$ | for every terminal in T.

Acceptance of 2 + 2 * 4 by this PDA

$\delta(q, 2 + 2 * 4, S)$ $\delta(q, \varepsilon\ S) \Rightarrow (q, S + S)$

$\longrightarrow (q, 2 + 2 * 4, S + S)$

$\delta(q, \varepsilon, S) \Rightarrow (q, 2)$

$\longrightarrow (q, 2 + 2 * 4, 2 + S)$

$\delta(q, 2, 2) \Rightarrow (q, \varepsilon)$

$\longrightarrow (q, + 2 * 4, + S)$

$\delta(q, +, +) \Rightarrow (q, \varepsilon)$

$\longrightarrow (q, 2 * 4, S)$

$\delta(q, \varepsilon, S) \Rightarrow (q, S * S)$

$\longrightarrow (q, 2 * 4, S * S)$

$\delta(q, \varepsilon, S) \Rightarrow (q, 2)$

$\longrightarrow (q, 2 * 4, 2 * S)$

$\delta(q, 2, 2) \Rightarrow (q, \varepsilon)$

$\longrightarrow (q, * 4, * S)$

$\delta(q, *, *) \Rightarrow (q, \varepsilon)$

$\longrightarrow (q, 4, S)$

$\delta(q, \varepsilon, S) = (q, 4)$

$\longrightarrow (q, 4, 4)$

$\delta(q, 4, 4) = (q, \varepsilon)$

$\longrightarrow (q, \varepsilon, \varepsilon)$

Ex. 6.6.6 : Convert the following expression grammar to PDA

$I \rightarrow a \mid b \mid Ia \mid Ib \mid I0 \mid I1$

$E \rightarrow I \mid E * E \mid E * E \mid (E)$

Soln. : The equivalent PDA, M is given by,

$$M = (\{q\},\{0, 1, a, b, *, +, (,)\},\{0, 1, a, b, *, +, (,), I, E\}, \delta, q, E, \phi)$$

where, δ is given by,

$\delta(q, \varepsilon, E) = \{(q, I), (q, E * E), (q, E + E), (q, (E))\}$

$\delta(q, \varepsilon, I) = \{(q, a), (q, b), (q, Ib), (q, Ia), (q, I0), (q, I1)\}$

$\delta(q, 0, 0) = \{(q, \varepsilon)\}$

$\delta(q, 1, 1) = \{(q, \varepsilon)\}$

$\delta(q, a, a) = \{(q, \varepsilon)\}$

$\delta(q, b, b) = \{(q, \varepsilon)\}$

$\delta(q, +, +) = \{(q, \varepsilon)\}$

$\delta(q, *, *) = \{(q, \varepsilon)\}$

$\delta(q, (, () = \{(q, \varepsilon)\}$

$\delta(q,),)) = \{(q, \varepsilon)\}$

Ex. 6.6.7 : Design a PDA corresponding to the grammar $S \rightarrow aSa \mid bSb \mid \in$ **MU - Dec. 14, 10 Marks**

Ans. : The equivalent PDA, M is given by :

$$M = (\{q\}, \{a,b\}, \{a,b,S\}, \delta, q, S, \phi)$$

Where δ is given by :

$\delta(q, \in, S) = \{(q, aSa), (q, bSb), (q, \in)\}$

$\delta(q, a, a) = (q, \in)$

$\delta(q, b, b) = (q, \in)$

Ex. 6.6.8 : Construct a PDA for the following Context Free Grammar (CFG).

$S \rightarrow CBAA \quad A \rightarrow 0A0 \mid 0$

$B \rightarrow 0B \mid 0 \quad C \rightarrow 0C1 \mid 1C0 \mid \varepsilon$ **MU - May 19. 5 Marks**

Soln. :

The equivalent PDA, M is given by :

$$M = (\{q\}, \{0, 1\}, \{0, 1, S, A, B, C\}, \{\delta, q, S, \phi\})$$

where δ is given by

$\delta(q, \varepsilon, S) = \{(q, CBAA)\}$

$\delta(q, \varepsilon, A) = \{(q, 0A0), (q, 0)\}$

$\delta(q, \varepsilon, B) = \{(q, 0B), (q, 0)\}$

$\delta(q, \varepsilon, C) = \{(q, 0C1), (q, 1C0), (q, \varepsilon)\}$

$\delta(q, 0, 0) = \{(q, \varepsilon)\}$

$\delta(q, 1, 1) = \{(q, \varepsilon)\}$

Ex. 6.6.9 : Show that if L is generated by a CFG then there exists a PDA accepting L.

Soln. :

Proof : Let G = (V, T, P, S) be a Context Free Grammar and a PDA, M is constructed as given below :

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

where δ is defined by :

1. For each variable $A \in V$, include a transition,

   $\delta(q, \varepsilon, A) \Rightarrow \{(q, \alpha) \mid A \rightarrow \alpha$ is a production in G $\}$

2. For each terminal $a \in T$, include a transition

   $\delta(q, a, a) \Rightarrow \{(q, \varepsilon)\}$

– The transitions of M are designed to simulate a leftmost derivation of a string.

1. The transition of the form

$\delta(q, \varepsilon, A) \Rightarrow (q, \alpha)$

is for expansion of the topmost variable of the stack.

2. The transition of the form

$\delta(q, a, a) \Rightarrow (q, \varepsilon)$

is for removing terminals from the stack so that a variable is exposed for further expansion.

To prove that a PDA constructed using above rules is equivalent to G.

We can prove that a word $w \in L(M)$ if and only if $w \in L(G)$.

where, L(M) is the language of PDA

and L(G) is the language of the given CFG.

Let us take a word $w \in L(G)$. The word w can be derived using the leftmost derivation.

$S \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow .... \Rightarrow \alpha_n = w$

where, $\alpha_i \rightarrow \alpha_{i+1}$

is obtained by single application of leftmost derivation, using a production in grammar G. We will show that for each α_i there is a unique configuration of PDA. α_n corresponds to the configuration of PDA accepting w.

Let $\alpha_i = x_i \beta_i$

Where, $x_i \in T^*$ and $B_i \in (V \cup T)^*$

The string α_i corresponds to a unique configuration of PDA given by a pair :

$$(y_i, \beta_i)$$

The corresponding diagram is shown in Fig. Ex. 6.6.8.

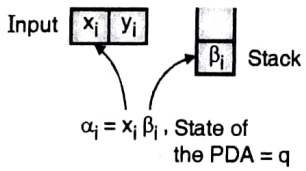
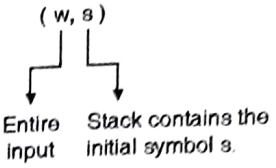


Fig. Ex. 6.6.8 : Unique configuration of PDA for α_i

- x_i portion of the input string w has already been scanned.
- The y_i is the portion of the input string, yet to be scanned. The stack contains β_i
- The theorem can be proved by induction on i of α_i.

Base case : i = 0

$\alpha_0 = S$, the configuration of PDA is given by



Induction step

We have to show that the correspondence is preserved for j = i + 1 also, if the correspondence for α_i is assumed.

Without the loss of generality, we can assume that the given grammar is in GNF.

If $(i + 1)_{th}$ input symbol is a_{i+1} and the first variable of β_i is V_i then the top of the stack contains V_i. The variable V_i can be expanded using the production.

$$V_i \Rightarrow a_{i+1} y$$

Similarly, the top symbol of the stack can be replaced with a_{i+1} y and then a_{i+1} can be popped out using the transition

$$\delta (q, a_{i+1}, a_{i+1}) \Rightarrow (q, \varepsilon)$$

Thus, there is one-to-one correspondence between the strings α_i and the configuration of PDA. Thus a string $w \in L(G)$ will be accepted by the PDA M.

6.6.2 Construction of CFG from PDA

We can find the Context Free Grammar G for any PDA, M such that

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

i.e., we can construct an equivalent CFG for a PDA.

The variables of the CFG, so constructed will be of the form :

[pXq], where $p, q \in Q$ and $X \in \Gamma$

Let the PDA is given by :

$$M = (Q, \Sigma, \Gamma, \delta, q_0, z, \phi)$$

Where, z is the initial stack symbol.

Then an equivalent CFG is given by

$$G = (V, \Sigma, P, S) \text{ where}$$

$$V = \{S, [p^x q] \mid p, q \in Q \text{ and } X \in \Gamma\}$$

Example : If $Q = \{q_0, q_1\}$ and $\Gamma = \{a, b, z\}$ then the possible set of variables in the corresponding CFG is given by :

1. S
2. $[q_0\ ^{a}\ q_0], [q_0\ ^{a}\ q_1], [q_1\ ^{a}\ q_0], [q_1\ ^{a}\ q_1]$
3. $[q_0\ ^{b}\ q_0], [q_0\ ^{b}\ q_1], [q_1\ ^{b}\ q_0], [q_1\ ^{b}\ q_1]$
4. $[q_0\ ^{z}\ q_0], [q_0\ ^{z}\ q_1], [q_1\ ^{z}\ q_0], [q_1\ ^{z}\ q_1]$

Set of productions for the equivalent CFG

1. Add the following productions for the start symbol S.

   $S \rightarrow [q_0\ ^{z}\ q_i]$ for each $q_i \in Q$, where z is the start symbol
2. For each transition of the form

   $\delta(q_i, a, B) \Rightarrow (q_j, C)$

   Where,

   (a) $q_i, q_j \in Q$

   (b) a belongs to $(\Sigma \cup \varepsilon)$

   (c) B and C belong to $(\Gamma \cup \varepsilon)$

   Then for each $q \in Q$, we add the production :

   $$[q_i\ ^{B}\ q] \rightarrow a\ [q_j\ ^{C}\ q]$$
3. For each transition of the form

   $\delta(q_i, a, B) \Rightarrow (q_j, C_1\ C_2)$

   Where,

   (a) $q_i, q_j \in Q$

   (b) a belongs to $(\Sigma \cup \varepsilon)$

   (c) B, C_1 and C_2 belongs to Γ

   then for each $p_1, p_2 \in Q$, we add the production

   $$[q_i\ ^{B}\ p_1] \rightarrow a\ [q_j\ ^{C_1}\ p_2]\ [p_2\ ^{C_2}\ p_1]$$

Ex. 6.6.10 : Convert PDA to CFG. PDA is given by

M = ({ p, q}, {0, 1}, {x, z}, δ, q, z), transition function δ is defined by :

$\delta(q, 1, z) \Rightarrow \{(q, xz)\}$

$\delta(q, 1, x) \Rightarrow \{(q, xx)\}$

$\delta(q, \varepsilon, x) \Rightarrow \{(q, \varepsilon)\}$

$\delta(q, 0, x) \Rightarrow \{(p, x)\}$

$\delta(p, 1, x) \Rightarrow \{(p, \varepsilon)\}$

$\delta(p, 0, z) \Rightarrow \{(q, z)\}$

Soln. :

Step 1 : Add productions for the start symbol.

$S \rightarrow [q\ ^{z}\ q]$

$S \rightarrow [q\ ^{z}\ p]$

Step 2 : Add productions for $\delta(q, 1, z) \Rightarrow \{(q, xz)\}$

$[q\ ^{z}\ q] \rightarrow 1\ [q\ ^{x}\ q]\ [q\ ^{z}\ q]$

$[q\ ^{z}\ q] \rightarrow 1\ [q\ ^{x}\ p]\ [p\ ^{z}\ q]$

$[q\ ^{z}\ p] \rightarrow 1\ [q\ ^{x}\ q]\ [q\ ^{z}\ p]$

$[q\ ^{z}\ p] \rightarrow 1\ [q\ ^{x}\ p]\ [p\ ^{z}\ p]$

Step 3 : Add productions For $\delta(q, 1, x) \Rightarrow \{(q, xx)\}$

$[q\ ^{x}\ q] \rightarrow 1\ [q\ ^{x}\ q]\ [q\ ^{x}\ q]$

$[q\ ^{x}\ q] \rightarrow 1\ [q\ ^{x}\ p]\ [p\ ^{x}\ q]$

$[q\ ^{x}\ p] \rightarrow 1\ [q\ ^{x}\ q]\ [q\ ^{x}\ p]$

$[q\ ^{x}\ p] \rightarrow 1\ [q\ ^{x}\ p]\ [p\ ^{x}\ p]$

Step 4 : Add productions for $\delta(q, \varepsilon, x) \Rightarrow \{(q, \varepsilon)\}$

$[q\ ^{x}\ q] \rightarrow \varepsilon$

Step 5 : Add productions for $\delta(q, 0, x) \Rightarrow \{(p, x)\}$

$[q\ ^{x}\ q] \rightarrow 0\ [p\ ^{x}\ q]$

$[q\ ^{x}\ p] \rightarrow 0\ [p\ ^{x}\ p]$

Step 6 : Add productions for $\delta(p, 1, x) \Rightarrow \{(p, \varepsilon)\}$

$[p\ ^{x}\ p] \rightarrow 1$

Step 7 : Add productions for $\delta(p, 0, z) \Rightarrow \{(q, z)\}$

$[p\ ^{z}\ q] \rightarrow 0\ [q\ ^{z}\ q]$

$[p\ ^{z}\ p] \rightarrow 0\ [q\ ^{z}\ p]$

Step 8 : Renaming of variables :

Original name	New name
$[q\ ^{z}\ q]$	A
$[q\ ^{z}\ p]$	B
$[p\ ^{z}\ q]$	C
$[p\ ^{z}\ p]$	D
$[q\ ^{x}\ q]$	E
$[q\ ^{x}\ p]$	F
$[p\ ^{x}\ q]$	G
$[p\ ^{x}\ p]$	H

The set of productions can be written as :

$S \rightarrow A \mid B$

$A \rightarrow 1EA \mid 1FC$

$B \rightarrow 1EB \mid 1FD$

$E \rightarrow IEE \mid 1FG$

$F \rightarrow 1EF \mid 1FH$

$E \rightarrow \varepsilon$

$E \rightarrow 0G$

$F \rightarrow 0H$

$H \rightarrow 1$

$C \rightarrow 0A$

$D \rightarrow 0B$

Step 9 : Simplification of grammar

Symbol G does not come on the left side of the production, hence it can be eliminated.

The equivalent set of productions is :

$S \rightarrow A \mid B$

$A \rightarrow 1EA \mid 1FC$

$B \rightarrow 1EB \mid 1FD$

$E \rightarrow 1EE \mid \varepsilon$

$F \rightarrow 1EF \mid 1FH \mid 0H$

$H \rightarrow 1$

$C \rightarrow 0A$

$D \rightarrow 0B$

Ex. 6.6.11 : Give the CFG generating the language accepted by the following PDA :

$M = (\{q_0, q_1\}, \{0, 1\}, \{z_0, x\}, \delta, q_0, z_0, \phi)$ when δ is given below

$\delta(q_0, 1, z_0) = \{(q_0, xz_0)\}$

$\delta(q_0, 1, x) = \{(q_0, xx)\}$

$\delta(q_0, 0, x) = \{(q_1, x)\}$

$\delta(q_0, \varepsilon, z_0) = \{(q_0, \varepsilon)\}$

$\delta(q_1, 1, x) = \{(q_1, \varepsilon)\}$

$\delta(q_1, 0, z_0) = \{(q_0, z_0)\}$

Soln. :

Step 1 : Add productions for the start symbol

$S \rightarrow [q_0 \, {}^{z_0} \, q_0]$

$S \rightarrow [q_0 \, {}^{z_0} \, q_1]$

Step 2 : Add productions for $\delta(q_0, 1, z_0) = \{(q_0, xz_0)\}$

$[q_0 \, {}^{z_0} \, q_0] \rightarrow 1 \, [q_0 \, {}^{x} \, q_0] \, [q_0 \, {}^{z_0} \, q_0]$

$[q_0 \, {}^{z_0} \, q_0] \rightarrow 1 \, [q_0 \, {}^{x} \, q_1] \, [q_1 \, {}^{z_0} \, q_0]$

$[q_0 \, {}^{z_0} \, q_1] \rightarrow 1 \, [q_0 \, {}^{x} \, q_0] \, [q_0 \, {}^{z_0} \, q_1]$

$[q_0 \, {}^{z_0} \, q_1] \rightarrow 1 \, [q_0 \, {}^{x} \, q_1] \, [q_1 \, {}^{z_0} \, q_1]$

Step 3 : Add productions for $\delta(q_0, 1, x) \Rightarrow \{(q_0, xx)\}$

$[q_0 \, {}^{x} \, q_0] \rightarrow 1 \, [q_0 \, {}^{x} \, q_0] \, [q_0 \, {}^{x} \, q_0]$

$[q_0 \, {}^{x} \, q_0] \rightarrow 1 \, [q_0 \, {}^{x} \, q_1] \, [q_1 \, {}^{x} \, q_0]$

$[q_0 \, {}^{x} \, q_1] \rightarrow 1 \, [q_0 \, {}^{x} \, q_0] \, [q_0 \, {}^{x} \, q_1]$

$[q_0 \, {}^{x} \, q_1] \rightarrow 1 \, [q_0 \, {}^{x} \, q_1] \, [q_1 \, {}^{x} \, q_1]$

Step 4 : Add productions for $\delta(q_0, 0, x) \Rightarrow \{(q_1, x)\}$

$[q_0 \, {}^{x} \, q_0] \rightarrow 0 \, [q_1 \, {}^{x} \, q_0]$

$[q_0 \, {}^{x} \, q_1] \rightarrow 0 \, [q_1 \, {}^{x} \, q_1]$

Step 5 : Add productions for $\delta(q_0, \varepsilon, z_0) = \{(q_1, \varepsilon)\}$

$[q_0 \, {}^{z_0} \, q_1] \rightarrow \varepsilon$

Step 6 : Add production for $\delta(q_1, 1, x) \Rightarrow \{(q_1, \varepsilon)\}$

$[q_1 \, {}^{x} \, q_1] \rightarrow 1$

Step 7 : Add productions for $\delta(q_1, 0, z_0) \Rightarrow \{(q_0, z_0)\}$

$[q_1 \, {}^{z_0} \, q_0] \Rightarrow 0 \, [q_0 \, {}^{z_0} \, q_0]$

$[q_1 \, {}^{z_0} \, q_1] \Rightarrow 0 \, [q_0 \, {}^{z_0} \, q_1]$

Ex. 6.6.12 : Consider the PDA with the following moves :

$\delta(q_0, a, z_0) = \{(q_0, az_0)\}$

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

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

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

$\delta(q_1, \varepsilon, z_0) = \{(q_1, \varepsilon)\}$

Obtain CFG equivalent to PDA. **MU - May 19, 10 Marks**

Soln. :

Step 1 : Add productions for the start symbol.

$S \rightarrow [q_0 \, {}^{z_0} \, q_0]$

$S \rightarrow [q_0 \, {}^{z_0} \, q_1]$

Step 2 : Add productions for $(q_0, a, a) = \{(q_0, aa)\}$

$[q_0 \, {}^{a} \, q_0] \rightarrow a \, [q_0 \, {}^{a} \, q_0] \, [q_0 \, {}^{a} \, q_0]$

$[q_0 \, {}^{a} \, q_0] \rightarrow a \, [q_0 \, {}^{a} \, q_1] \, [q_1 \, {}^{a} \, q_0]$

$[q_0 \, {}^{a} \, q_1] \rightarrow a \, [q_0 \, {}^{a} \, q_0] \, [q_0 \, {}^{a} \, q_1]$

$[q_0 \, {}^{a} \, q_1] \rightarrow a \, [q_0 \, {}^{a} \, q_1] \, [q_1 \, {}^{a} \, q_1]$

Step 3 : Add productions for $\delta(q_0, b, a) = \{(q_1, \varepsilon)\}$

$[q_0\ ^{a}\ q_1] \rightarrow b$

Step 4 : Add productions for $\delta(q_1, b, a) = \{(q_1, \varepsilon)\}$

$[q_1\ ^{a}\ q_1] \rightarrow b$

Step 5 : Add productions for $\delta(q_1, \varepsilon, z_0) \rightarrow \{(q_1, \varepsilon)\}$

$[q_1\ ^{z_0}\ q_1] \rightarrow \varepsilon$

Ex. 6.6.13 : For the PDA ({ q_0, q_1 }, { 0,1 }, {0,1,z_0 }, δ, q_0, z_0, ϕ) where δ is

$\delta(q_0, \varepsilon, z_0) = \{(q_1, \varepsilon)\}$

$\delta(q_0, 0, z_0) = \{(q_0, 0z_0)\}$

$\delta(q_0, 0, 0) = \{(q_0, 00)\}$

$\delta(q_0, 1, 0) = \{(q_0, 10)\}$

$\delta(q_0, 1, 1) = \{(q_0, 11)\}$

$\delta(q_0, 0, 1) = \{(q_1, \varepsilon)\}$

$\delta(q_1, 0, 1) = \{(q_1, \varepsilon)\}$

$\delta(q_1, 0, 0) = \{(q_1, \varepsilon)\}$

$\delta(q_1, \varepsilon, z_0) = \{(q_1, \varepsilon)\}$

Obtain CFG accepted by the above PDA and simplify the CFG and describe the language it accepts.

Soln. :

Sr. No.	PDA transition	Corresponding productions
1.	Productions due to start symbol S.	$S \rightarrow [q_0\ ^{z_0}\ q_0]$
		$S \rightarrow [q_0\ ^{z_0}\ q_1]$
2.	$\delta(q_0, \varepsilon, z_0) = (q_1, \varepsilon)$	$[q_0\ ^{z_0}\ q_1] \rightarrow \varepsilon$
3.	$\delta(q_0, 0, z_0) = (q_0, 0z_0)$	$[q_0\ ^{z_0}\ q_0] \rightarrow 0\,[q_0\ ^{0}\ q_0]\,[q_0\ ^{z_0}\ q_0]$
		$[q_0\ ^{z_0}\ q_0] \rightarrow 0\,[q_0\ ^{0}\ q_1]\,[q_1\ ^{z_0}\ q_0]$
		$[q_0\ ^{z_0}\ q_1] \rightarrow 0\,[q_0\ ^{0}\ q_0]\,[q_0\ ^{z_0}\ q_1]$
		$[q_0\ ^{z_0}\ q_1] \rightarrow 0\,[q_0\ ^{0}\ q_1]\,[q_1\ ^{z_0}\ q_1]$
4.	$\delta(q_0, 0, 0) = (q_0, 00)$	$[q_0\ ^{0}\ q_0] \rightarrow 0\,[q_0\ ^{0}\ q_0\ ]\,[q_0\ ^{0}\ q_0]$
		$[q_0\ ^{0}\ q_0] \rightarrow 0\,[q_0\ ^{0}\ q_1]\,[q_1\ ^{0}\ q_0]$
		$[q_0\ ^{0}\ q_1] \rightarrow 0\,[q_0\ ^{0}\ q_0]\,[q_0\ ^{0}\ q_1]$
		$[q_0\ ^{0}\ q_1] \rightarrow 0\,[q_0\ ^{0}\ q_1]\,[q_1\ ^{0}\ q_1]$
5.	$\delta(q_0, 1, 0) = (q_0, 10)$	$[q_0\ ^{0}\ q_0] \rightarrow 1\,[q_0\ ^{1}\ q_0]\,[q_0\ ^{0}\ q_0]$
		$[q_0\ ^{0}\ q_0] \rightarrow 1\,[q_0\ ^{1}\ q_1]\,[q_1\ ^{0}\ q_0]$
		$[q_0\ ^{0}\ q_1] \rightarrow 1\,[q_0\ ^{1}\ q_0]\,[q_0\ ^{0}\ q_1]$
		$[q_0\ ^{0}\ q_1] \rightarrow 1\,[q_0\ ^{1}\ q_1]\,[q_1\ ^{0}\ q_1]$
6.	$\delta(q_0, 1, 1) = (q_0, 11)$	$[q_0\ ^{1}\ q_0] \rightarrow 1\,[q_0\ ^{1}\ q_0]\,[q_0\ ^{1}\ q_0]$
		$[q_0\ ^{1}\ q_0] \rightarrow 1\,[q_0\ ^{1}\ q_1]\,[q_1\ ^{1}\ q_0]$
		$[q_0\ ^{1}\ q_1] \rightarrow 1\,[q_0\ ^{1}\ q_0]\,[q_0\ ^{1}\ q_1]$
		$[q_0\ ^{1}\ q_1] \rightarrow 1\,[q_0\ ^{1}\ q_1]\,[q_1\ ^{1}\ q_1]$
7.	$\delta(q_0, 0, 1) = (q_1, \varepsilon)$	$[q_0\ ^{1}\ q_1] \rightarrow 0$
8.	$\delta(q_1, 0, 1) = (q_1, \varepsilon)$	$[q_1\ ^{1}\ q_1] \rightarrow 0$
9.	$\delta(q_1, 0, 0) = (q_1, \varepsilon)$	$[q_1\ ^{0}\ q_1] \rightarrow 0$
10.	$\delta(q_1, \varepsilon, z_0) = (q_1, \varepsilon)$	$[q_1\ ^{z_0}\ q_1] \rightarrow \varepsilon$

Simplification of grammar

We can rename the variables as given below :

$[q_0\ ^{z_0}\ q_0]$ - A, $[q_0\ ^{z_0}\ q_1]$ - B, $[q_1\ ^{z_0}\ q_0]$ - C, $[q_1\ ^{z_0}\ q_1]$ - D

$[q_0\ ^{z_0}\ q_0]$ - E, $[q_0\ ^{0}\ q_1]$ - F, $[q_1\ ^{0}\ q_0]$ - G, $[q_1\ ^{0}\ q_1]$ - H

$[q_0\ ^{1}\ q_0]$ - I, $[q_0\ ^{1}\ q_1]$ - J, $[q_1\ ^{1}\ q_0]$ - K, $[q_1\ ^{1}\ q_1]$ - L

With the above substitutions, the resulting set of productions can be written as :

$S \rightarrow A \mid B$

$B \rightarrow \varepsilon$

$A \rightarrow 0EA \mid 0FC$

$B \rightarrow 0EB \mid 0FD$

$E \rightarrow 0EE \mid 0FG$

$F \rightarrow 0EF \mid 0FH$

$E \rightarrow 1IE \mid 1JG$

$F \rightarrow 1IF \mid 1JH$

$I \rightarrow 1II \mid 1JK$

$J \rightarrow 1IJ \mid 1JL$

$J \rightarrow 0$

$L \rightarrow 0$

$H \rightarrow 0$

$D \rightarrow \varepsilon$

1. Removing ε-productions :

Nullable set = {D, B, S}

ε-productions are removed with resulting set of productions as given below :

$S \rightarrow A \mid B$

$A \rightarrow 0EA \mid 0FC$

$B \rightarrow 0EB \mid 0FD \mid 0E \mid 0F$

$E \rightarrow 0EE \mid 0FG \mid 1IE \mid 1JG$

$F \rightarrow 0EF \mid 0FH \mid 1IF \mid 1JH$

$H \rightarrow 0$

$I \rightarrow 1II \mid 1JK$

$J \rightarrow 1IJ \mid 1JL \mid 0$

$L \rightarrow 0$

2. Removing non-generating symbols

Set of productions after elimination of non-generating symbols {A, C, D, E, G, I} is given below :

$S \rightarrow B$

$B \rightarrow 0F$

$F \rightarrow 0FH \mid 1JH$

$H \rightarrow 0$

$J \rightarrow 1JL \mid 0$

$L \rightarrow 0$

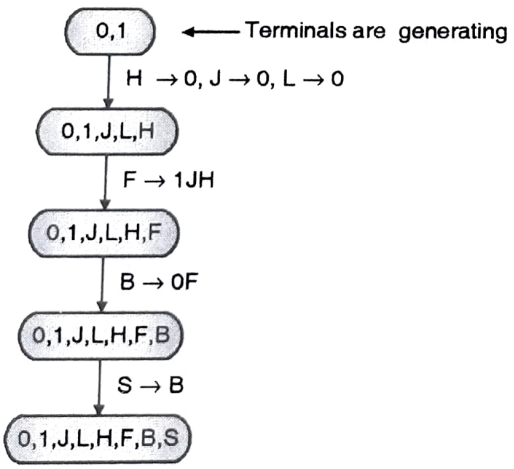


Fig. Ex. 6.6.13

3. The unit production $S \rightarrow B$ should be removed. The set of productions after elimination of the unit production $S \rightarrow B$ is given below :

$S \rightarrow 0F$

$F \rightarrow 0FH \mid 1JH$

$H \rightarrow 0$

$J \rightarrow 1JL \mid 0$

$L \rightarrow 0$

Language accepted by the PDA

The language accepted by the PDA is given by :

$$L = \{0^n, 1^m 0^{n+m} \mid n, m \geq 1\} \cup \varepsilon$$

Ex. 6.6.14 : Construct a PDA accepting $\{a^n b^m a^n \mid m, n > 1\}$ by null store.

From the PDA construct the corresponding CFG.

Soln. :

Algorithm

1. Sequence of a's should be pushed onto the stack in state q_0.

$$\delta(q_0,a,z_0) = (q_0,az_0)$$

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

2. Sequence of input b's should be skipped. These b's will have no effect on the stack.

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

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

3. Initial a's which are on the stack should be matched with the trailing a's in the input. An 'a' should be popped for every 'a' as input till the end of input.

$$\delta(q_1,a,a) = (q_2, \varepsilon)$$

$$\delta(q_2,a,a) = (q_2, \varepsilon)$$

4. Finally, the symbol z_0 should be popped but to make the stack empty.

$$\delta(q_2, \varepsilon, z_0) = (q_2, \varepsilon)$$

Transition diagram and the transition table are given below.

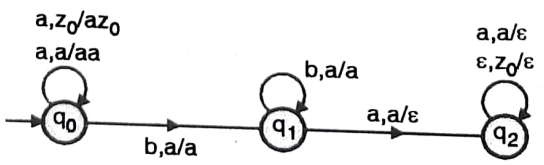


Fig. Ex. 6.6.14 : Transition diagram

Transition table

$\delta(q_0,a,z_0) = (q_0,a,z_0)$

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

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

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

$\delta(q_1,a,a) = (q_2,\varepsilon)$

$\delta(q_2,a,a) = (q_2,\varepsilon)$

$\delta(q_2,\varepsilon,z_0) = (q_2,\varepsilon)$

The PDA M $= (\{q_0,q_1,q_2\},\{a,b\},\{a,b,z_0\},\delta,q_0,z_0,\phi)$

PDA to CFG

Step 1 : Add productions for the start symbol

$S \rightarrow [q_0\ ^{z_0}\ q_0]$

$S \rightarrow [q_0\ ^{z_0}\ q_1]$

$S \rightarrow [q_0\ ^{z_0}\ q_2]$

Step 2 : Add productions for $\delta(q_0,a,z_0) = (q_0,az_0)$

$[q_0\ ^{z_0}\ q_0] \rightarrow a\ [q_0\ ^{a}\ q_0]\ [q_0\ ^{z_0}\ q_0]$

$[q_0\ ^{z_0}\ q_0] \rightarrow a\ [q_0\ ^{a}\ q_1]\ [q_1\ ^{z_0}\ q_0]$

$[q_0\ ^{z_0}\ q_0] \rightarrow a\ [q_0\ ^{a}\ q_2]\ [q_2\ ^{z_0}\ q_0]$

$[q_0\ ^{z_0}\ q_1] \rightarrow a\ [q_0\ ^{a}\ q_0]\ [q_0\ ^{z_0}\ q_1]$

$[q_0\ ^{z_0}\ q_1] \rightarrow a\ [q_0\ ^{a}\ q_1]\ [q_1\ ^{z_0}\ q_1]$

$[q_0\ ^{z_0}\ q_1] \rightarrow a\ [q_0\ ^{a}\ q_2]\ [q_2\ ^{z_0}\ q_1]$

$[q_0\ ^{z_0}\ q_2] \rightarrow a\ [q_0\ ^{a}\ q_0]\ [q_0\ ^{z_0}\ q_2]$

$[q_0\ ^{z_0}\ q_2] \rightarrow a\ [q_0\ ^{a}\ q_1]\ [q_1\ ^{z_0}\ q_2]$

$[q_0\ ^{z_0}\ q_2] \rightarrow a\ [q_0\ ^{a}\ q_2]\ [q_2\ ^{z_0}\ q_2]$

Step 3 : Add productions for $\delta(q_0,a,a) = (q_0,aa)$

$[q_0\ ^{a}\ q_0] \rightarrow a\ [q_0\ ^{a}\ q_0]\ [q_0\ ^{a}\ q_0]$

$[q_0\ ^{a}\ q_0] \rightarrow a\ [q_0\ ^{a}\ q_1]\ [q_1\ ^{a}\ q_0]$

$[q_0\ ^{a}\ q_0] \rightarrow a\ [q_0\ ^{a}\ q_2]\ [q_2\ ^{a}\ q_0]$

$[q_0\ ^{a}\ q_1] \rightarrow a\ [q_0\ ^{a}\ q_0]\ [q_0\ ^{a}\ q_1]$

$[q_0\ ^{a}\ q_1] \rightarrow a\ [q_0\ ^{a}\ q_1]\ [q_1\ ^{a}\ q_1]$

$[q_0\ ^{a}\ q_1] \rightarrow a\ [q_0\ ^{a}\ q_2]\ [q_2\ ^{a}\ q_1]$

$[q_0\ ^{a}\ q_2] \rightarrow a\ [q_0\ ^{a}\ q_0]\ [q_0\ ^{a}\ q_2]$

$[q_0\ ^{a}\ q_2] \rightarrow a\ [q_0\ ^{a}\ q_1]\ [q_1\ ^{a}\ q_2]$

$[q_0\ ^{a}\ q_2] \rightarrow a\ [q_0\ ^{a}\ q_2]\ [q_2\ ^{a}\ q_2]$

Step 4 : Add productions for $\delta(q_0,b,a) = (q_1,a)$

$[q_0\ ^{a}\ q_0] \rightarrow b\ [q_1\ ^{a}\ q_0]$

$[q_0\ ^{a}\ q_1] \rightarrow b\ [q_1\ ^{a}\ q_1]$

$[q_0\ ^{a}\ q_2] \rightarrow b\ [q_1\ ^{a}\ q_2]$

Step 5 : Add productions for $\delta(q_1,b,a) = (q_1,a)$

$[q_1\ ^{a}\ q_0] \rightarrow b\ [q_1\ ^{a}\ q_0]$

$[q_1\ ^{a}\ q_1] \rightarrow b\ [q_1\ ^{a}\ q_1]$

$[q_1\ ^{a}\ q_2] \rightarrow b\ [q_1\ ^{a}\ q_2]$

Step 6 : Add productions for $\delta(q_1,a,a) = (q_2,\varepsilon)$

$[q_1\ ^{a}\ q_2] \rightarrow a$

Step 7 : Add productions for $\delta(q_2,a,a) = (q_2,\varepsilon)$

$[q_2\ ^{a}\ q_2] \rightarrow a$

Step 8 : Add productions for $\delta(q_2,\varepsilon,z_0) = (q_2,\varepsilon)$

$[q_2\ ^{z_0}\ q_2] \rightarrow \varepsilon$

Ex. 6.6.15 : Design a PDA and then corresponding CFG for the language that accepts the simple palindrome.

$L = \{xcx^R \mid x \in \{a,b\}^*\}$

Soln. : Transitions for the PDA are given by :

$\delta(q_0, a, z) = (q_0, az)$

$\delta(q_0, b, z) = (q_0, bz)$

$\delta(q_0, c, z) = (q_0, \varepsilon)$

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

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

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

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

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

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

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

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

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

- z is the initial stack symbol.
- q_0 is the initial state.
- String is accepted through an empty stack.

Productions for the corresponding CFG are given below :

Step 1 : Add productions for the start symbol.

$S \rightarrow [q_0 \, {}^{z} \, q_0]$

$S \rightarrow [q_0 \, {}^{z} \, q_1]$

Step 2 : Add productions for $\delta(q_0, a, z) = (q_0, az)$

$[q_0 \, {}^{z} \, q_0] \rightarrow a \, [q_0 \, {}^{a} \, q_0] \, [q_0 \, {}^{z} \, q_0]$

$[q_0 \, {}^{z} \, q_0] \rightarrow a \, [q_0 \, {}^{a} \, q_1] \, [q_1 \, {}^{z} \, q_0]$

$[q_0 \, {}^{z} \, q_1] \rightarrow a \, [q_0 \, {}^{a} \, q_0] \, [q_0 \, {}^{z} \, q_1]$

$[q_0 \, {}^{z} \, q_1] \rightarrow a \, [q_0 \, {}^{a} \, q_1] \, [q_1 \, {}^{z} \, q_1]$

Step 3 : Add productions for $\delta(q_0, c, z) = (q_0, \varepsilon)$

$[q_0 \, {}^{z} \, q_0] \rightarrow c$

Step 4 : Add productions for $\delta(q_0, a, a) = (q_0, aa)$

$[q_0 \, {}^{a} \, q_0] \rightarrow a \, [q_0 \, {}^{a} \, q_0] \, [q_0 \, {}^{a} \, q_0]$

$[q_0 \, {}^{a} \, q_0] \rightarrow a \, [q_0 \, {}^{a} \, q_1] \, [q_1 \, {}^{a} \, q_0]$

$[q_0 \, {}^{a} \, q_1] \rightarrow a \, [q_0 \, {}^{a} \, q_0] \, [q_0 \, {}^{a} \, q_1]$

$[q_0 \, {}^{a} \, q_1] \rightarrow a \, [q_0 \, {}^{a} \, q_1] \, [q_1 \, {}^{a} \, q_1]$

Step 5 : Add productions for $\delta(q_0, a, b) = (q_0, ab)$

$[q_0 \, {}^{b} \, q_0] \rightarrow a \, [q_0 \, {}^{a} \, q_0] \, [q_0 \, {}^{b} \, q_0]$

$[q_0 \, {}^{b} \, q_0] \rightarrow a \, [q_0 \, {}^{a} \, q_1] \, [q_1 \, {}^{b} \, q_0]$

$[q_0 \, {}^{b} \, q_1] \rightarrow a \, [q_0 \, {}^{a} \, q_0] \, [q_0 \, {}^{b} \, q_1]$

$[q_0 \, {}^{b} \, q_1] \rightarrow a \, [q_0 \, {}^{a} \, q_1] \, [q_1 \, {}^{b} \, q_1]$

Step 6 : Add productions for $\delta(q_0, b, a) = (q_0, ba)$

$[q_0 \, {}^{a} \, q_0] \rightarrow b \, [q_0 \, {}^{b} \, q_0] \, [q_0 \, {}^{a} \, q_0]$

$[q_0 \, {}^{a} \, q_0] \rightarrow b \, [q_0 \, {}^{b} \, q_1] \, [q_1 \, {}^{a} \, q_0]$

$[q_0 \, {}^{a} \, q_1] \rightarrow b \, [q_0 \, {}^{b} \, q_0] \, [q_0 \, {}^{a} \, q_1]$

$[q_0 \, {}^{a} \, q_1] \rightarrow b \, [q_0 \, {}^{b} \, q_1] \, [q_1 \, {}^{a} \, q_1]$

Step 7 : Add productions for $\delta(q_0, b, b) = (q_0, bb)$

$[q_0 \, {}^{b} \, q_0] \rightarrow b \, [q_0 \, {}^{b} \, q_0] \, [q_0 \, {}^{b} \, q_0]$

$[q_0 \, {}^{b} \, q_0] \rightarrow b \, [q_0 \, {}^{b} \, q_1] \, [q_1 \, {}^{b} \, q_0]$

$[q_0 \, {}^{b} \, q_1] \rightarrow b \, [q_0 \, {}^{b} \, q_0] \, [q_0 \, {}^{b} \, q_1]$

$[q_0 \, {}^{b} \, q_1] \rightarrow b \, [q_0 \, {}^{b} \, q_1] \, [q_1 \, {}^{b} \, q_1]$

Step 8 : Add productions for $\delta(q_0, c, a) = (q_1, a)$

$[q_0 \, {}^{a} \, q_0] \rightarrow c \, [q_1 \, {}^{a} \, q_0]$

$[q_0 \, {}^{a} \, q_1] \rightarrow c \, [q_1 \, {}^{a} \, q_1]$

Step 9 : Add productions for $\delta(q_0, c, b) = (q_1, b)$

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

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

Step 10 : Add productions for $\delta(q_1, a, a) = (q_1, \varepsilon)$

$[q_1 \, {}^{a} \, q_1] \rightarrow a$

Step 11 : Add productions for $\delta(q_1, b, b) = (q_1, \varepsilon)$

$[q_1 \, {}^{b} \, q_1] \rightarrow b$

Step 12 : Add production for $\delta(q_1, \varepsilon, z) = (q_1, \varepsilon)$

$[q_1 \, {}^{z} \, q_1] \rightarrow \varepsilon$

Step 13 : Add productions for $\delta(q_0, b, z) = (q_0, bz)$

$[q_0 \, {}^{z} \, q_0] \rightarrow b \, [q_0 \, {}^{b} \, q_0] \, [q_0 \, {}^{z} \, q_0]$

$[q_0 \, {}^{z} \, q_0] \rightarrow b \, [q_0 \, {}^{b} \, q_1] \, [q_1 \, {}^{z} \, q_0]$

$[q_0 \, {}^{z} \, q_1] \rightarrow b \, [q_0 \, {}^{b} \, q_0] \, [q_0 \, {}^{z} \, q_1]$

$[q_0 \, {}^{z} \, q_1] \rightarrow b \, [q_0 \, {}^{b} \, q_1] \, [q_1 \, {}^{z} \, q_1]$

Ex. 6.6.16 : For the PDA

$(\{q_0, q_1\}, (0, 1), \{ z_0, x\}, \delta, q_0, z_0, \phi)$ where δ is

$\delta(q_0, 1, z_0) = \{(q_0, xz_0)\}$

$\delta(q_0, 1, x) = \{(q_0, xx)\}$

$\delta(q_0, 0, x) = \{(q_1, x)\}$

$\delta(q_0, \varepsilon, z_0) = \{(q_0, \varepsilon)\}$

$\delta(q_1, 1, x) = \{(q_1, \varepsilon)\}$

$\delta(q_1, 0, z_0) = \{(q_0, z_0)\}$

Obtain CFG accepted by the above PDA and simplify the CFG and describe the language it accepts.

Soln. :

Sr. No.	PDA transition	Corresponding productions
1.	Productions due to start symbol S.	$S \rightarrow [q_0 \, {}^{z_0} \, q_0]$
		$S \rightarrow [q_0 \, {}^{z_0} \, q_1]$
2.	$\delta(q_0, 1, z_0) = (q_0, xz_0)$	$[q_0 \, {}^{z_0} \, q_0] \rightarrow 1 \, [q_0 \, {}^{x} \, q_0] \, [q_0 \, {}^{z_0} \, q_0]$

Sr. No.	PDA transition	Corresponding productions
		$[q_0\ ^{Z_0}\ q_0] \rightarrow 1\ [q_0\ ^{X}\ q_1]\ [q_1\ ^{Z_0}\ q_0]$
		$[q_0\ ^{Z_0}\ q_1] \rightarrow 1\ [q_0\ ^{X}\ q_0]\ [q_0\ ^{Z_0}\ q_1]$
		$[q_0\ ^{Z_0}\ q_1] \rightarrow 1\ [q_0\ ^{X}\ q_1]\ [q_1\ ^{Z_0}\ q_1]$
3.	$\delta(q_0, 1, X) = (q_0, XX)$	$[q_0\ ^{X}\ q_0] \rightarrow 1\ [q_0\ ^{X}\ q_0]\ [q_0\ ^{X}\ q_0]$
		$[q_0\ ^{X}\ q_0] \rightarrow 1\ [q_0\ ^{X}\ q_1]\ [q_1\ ^{X}\ q_0]$
		$[q_0\ ^{X}\ q_1] \rightarrow 1\ [q_0\ ^{X}\ q_0]\ [q_0\ ^{X}\ q_1]$
		$[q_0\ ^{X}\ q_1] \rightarrow 1\ [q_0\ ^{X}\ q_1]\ [q_1\ ^{X}\ q_1]$
4.	$\delta(q_0, 0, X) = (q_1, X)$	$[q_0\ ^{X}\ q_0] \rightarrow 0[q_1\ ^{X}\ q_0]$
		$[q_0\ ^{X}\ q_1] \rightarrow 0[q_1\ ^{X}\ q_1]$
5.	$\delta(q_0, \varepsilon, Z_0) = (q_0, \varepsilon)$	$[q_0\ ^{Z_0}\ q_0] \rightarrow \varepsilon$
6.	$\delta(q_1, 1, X) = (q_1, \varepsilon)$	$[q_1\ ^{X}\ q_1] \rightarrow 1$
7.	$\delta(q_1, 0, Z_0) = (q_1, Z_0)$	$[q_1\ ^{Z_0}\ q_0] \rightarrow 0\ [q_0\ ^{Z_0}\ q_0]$
		$[q_1\ ^{Z_0}\ q_1] \rightarrow 0\ [q_0\ ^{Z_0}\ q_1]$

Simplification of grammar

We can rename the variables as given below :

$[q_0\ ^{Z_0}\ q_0]$ – A, $[q_0\ ^{Z_0}\ q_1]$ – B, $[q_1\ ^{Z_0}\ q_0]$ – C, $[q_1\ ^{Z_0}\ q_1]$ – D

$[q_0\ ^{X}\ q_0]$ – E, $[q_0\ ^{X}\ q_1]$ – F, $[q_1\ ^{X}\ q_0]$ – G, $[q_1\ ^{X}\ q_1]$ – H

With the above substitutions, the resulting set of productions can be written as :

$S \rightarrow A \mid B$

$A \rightarrow 1EA \mid 1FC$

$B \rightarrow 1EB \mid 1FD$

$E \rightarrow 1EE \mid 1FG$

$F \rightarrow 1EF \mid 1FH$

$E \rightarrow 0G$

$F \rightarrow 0H$

$A \rightarrow \varepsilon$

$H \rightarrow 1$

$C \rightarrow 0A$

$D \rightarrow 0B$

1. Removing ε-production.

Nullable set = {S, A}

ε-productions are removed with resulting set of productions as given below :

$S \rightarrow A \mid B$

$A \rightarrow 1EA \mid 1FC \mid 1E$

$B \rightarrow 1EB \mid 1\ FD$

$C \rightarrow 0A \mid 0$

$D \rightarrow 0B$

$E \rightarrow 1EE \mid 1\ FG \mid 0G$

$F \rightarrow 1EH \mid 1FH \mid 0H$

$H \rightarrow 0$

2. Removing non-generating symbols

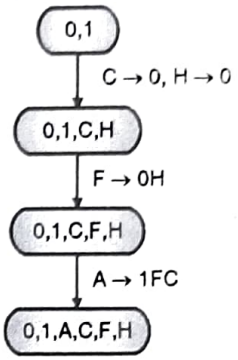


Fig. Ex. 6.6.16

Following symbols are non-generating = (B, D, E, G)

Set of productions after elimination of non-generating symbols :

$S \rightarrow A$

$A \rightarrow 1FC$

$C \rightarrow 0A \mid 0$

$F \rightarrow 1FH \mid 0H$

$H \rightarrow 0$

3. Unit production $S \rightarrow A$ is removed, the resulting set of productions :

$S \rightarrow 1FC$

$A \rightarrow 1FC$

$C \rightarrow 0A \mid 0$

$F \rightarrow 1FH \mid 0H$

$H \rightarrow 0$

The language accepted by the PDA is given by :

$$L = \{1^n01^n0 \ldots..1^n0 \mid n \geq 1\} \cup \varepsilon$$

Ex. 6.6.17 : Show that if a language L is accepted by a PDA then there exists a CFG generating L.

Soln. :

Proof : For simplicity, we will consider a normalized PDA with following properties :

1. There is a single final state q_f
2. It empties the stack before accepting
3. Each transition either pushes a symbol onto the stack or performs a **pop** operation, but not both.

A transition,

$\delta(q_i, x, b) \Rightarrow (q_j, c)$ [pop b and push c]

can be replaced by couple of transitions.

1. $\delta(q_i, x, b) \Rightarrow \delta(q_{temp}, \varepsilon)$
2. $\delta(q_{temp}, \varepsilon, \varepsilon) \Rightarrow \delta(q_j, c)$

A new state q_{temp} has been introduced.

Similarly, a transition that neither pushes nor pops anything can be replaced with two transitions :

1. Transition pushing a dummy stack symbol
2. Transition popping the dummy stack symbol.

Let us consider a normalized PDA,

$N = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$, where $F = \{ q_f \}$

- A word w will be accepted by N if it starts in q_0 and ends up in q_f with an empty stack.
- The language $L_{p, q}$, where $p, q \in Q$, is defined as consisting of those strings that start at state p with an empty stack and ends up in q with an empty stack.

Ex. 6.6.18 : Describe the language L(M) in English for push down automation.

$M = (K, \Sigma, \Gamma, \Delta, S, F)$

where, $k = \{S, F\}$

$F = \{F\}$

$\Sigma = \{a, b\}$

$T = \{a\}$

$\Delta = \{((S, a, \varepsilon), (S, a)), ((S, b, \varepsilon), (S, a)),$
$((S, a, \varepsilon), (F, \varepsilon)), ((F, a, a),$
$(F, \varepsilon)), ((F, b, a), (F, \varepsilon))\}$

Soln. :

It recognizes a string of the form

$(a + b)^m a(a + b)^n \mid n \leq m$ and $n, m \geq 0$

- The middle character 'a' is determined non-deterministically.
- For every symbol from the first $(a + b)^m$ of $(a + b)^m a (a + b)^n$ an 'a' is pushed onto the stack using the given two moves.

  $((S, a, \varepsilon),(S,a)),((S,b,\varepsilon),(S,a))$
- The middle a of the input string $(a + b)^m a (a + b)^n$ takes the PDA to state F. 'a' is fixed non-deterministically.
- For every symbol from $(a + b)^n$ of $(a + b)^m a (a + b)^n$ an a is erased from the stack in state F.

6.7 Deterministic Push Down Automata (DPDA)

In a DPDA there is only one move in every situation. A DPDA is less powerful than NPDA. Every context free language cannot be accepted by a DPDA. For example, a string of the form ww^R can not be processed by a DPDA. The class of a language a DPDA can accept lies in between a regular language and CFL.

A DPDA is defined as :

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

$\delta(q, a, x)$ has one move for any $q \in Q, X \in \Gamma$ and $a \in \Sigma$.

6.7.1 Regular Language and DPDA

We can always design a DPDA for a regular language. A DPDA can be designed for regular language in two steps :

Step 1 : Construct an equivalent DFA for the given regular language.

Step 2 : For every transition $\delta(q_i, a) = q_j$ in FA (where $q_i, q_j \in Q$ and $a \in \Sigma$), we can write an equivalent transition for DPDA.

$\delta(q_i, a, z_0) = \{(q_j, z_0)\}$.

The move for PDA involves neither a **Push** nor a **Pop** operation.

Ex. 6.7.1 : Design a DPDA for a binary number divisible by 3.

Soln. :

Step 1 : Construction of DFA.

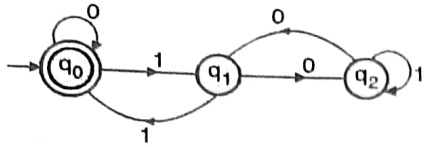


Fig. Ex. 6.7.1 : Construction of DFA

Step 2 : The DPDA is given by :

$M = (\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, 1, z_0\}, \delta, q_0, z_0, \{q_0\})$

where the transition function δ is :

$$\delta(q_0, 0, z_0) = (q_0, z_0)$$
$$\delta(q_0, 1, z_0) = (q_1, z_0)$$
$$\delta(q_1, 0, z_0) = (q_2, z_0)$$
$$\delta(q_1, 1, z_0) = (q_0, z_0)$$
$$\delta(q_2, 0, z_0) = (q_1, z_0)$$
$$\delta(q_2, 1, z_0) = (q_2, z_0)$$

Ex. 6.7.2 : Show that if L is accepted by a PDA in which no symbols are removed from the stack, then L is regular.

Soln. : Every regular language is accepted by some FA. Every transition of an FA can be converted into a PDA move without any push or pop operation.

Let there be a transition $\delta(q_i, a) = q_j$ in the FA. Where,

$q_i, q_j \in Q$ and $a \in \Sigma$.

The FA move $\delta(q_i, a) = q_j$ can be converted into an equivalent PDA move as given below :

$$\delta(q_i, a, z_0) = \{(q_j, z_0)\}$$

This move for the PDA involves neither a **Push** nor a **Pop** operation.

Ex. 6.7.3 : Prove "Let L be a language accepted by deterministic PDA, then the complement of L, can also be accepted by a DPDA".

Soln. : Let the DPDA for the given language L is

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

We can construct M′ from M, such that M′ accepts L′.

$$M' = (Q, \Sigma, \Gamma, \delta, q_0, z_0, Q\text{-}F)$$

i.e. an accepting state in M becomes a non-accepting state in M′ and a non-accepting state in M becomes an accepting state in M′.

Proof that M′ accepts L′

Case I : Let us take a string $\omega \in L$.

On application of ω, the machine M will reach a final state.

$(q_0, \omega, z_0) \vdash_M^* (p, -)$, where $p \in F$.

Since the state of $p \notin Q\text{-}F$, it will not be accepted by M′.

Case II : Let us take a string $\omega \in \Sigma^*$ but $\omega \notin L$. i.e. $\omega \in L'$.

On application of ω, the machine M will reach a non-final state.

$(q_0, \omega, z_0) \vdash_M^* (r, -)$, where $r \notin F$.

Since, the state $r \in Q\text{-}F$, it will be accepted by M′.

Ex. 6.7.4 : Enlist the difference between PDM and FSM.

Soln. :

1. PDM is more powerful than FSM.
2. PDM has additional memory in the form of a stack.
3. PDM can handle CFL but FSM can not handle CFL.

6.8 Application of PDA

PDA is a machine for CFL. A string belonging to a CFL can be recognized by a PDA. PDA is extensively used for parsing. PDA is an abstract machine; it can also used for giving proofs of lemma on CFL.

6.9 Parsing

We can always find a method for determining whether a particular string is generated by a CFG. Parsing a string is nothing but finding a derivation of the string in the given grammar G. A great deal of work has been done in finding an efficient algorithm for parsing. These algorithms depend on specific properties of the grammar.

PDA is a machine for CFG. A PDA can be used for parsing. A PDA can be enhanced to record its moves, so that the sequence of moves leading to an acceptance of the string can be remembered.

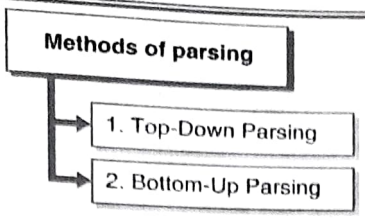


Fig. 6.9.1 : Methods of parsing

6.9.1 Top-Down Parsing

A top down parser for a given grammar G tries to derive a string through a sequence of derivations starting with the start symbol.

A top down parser, normally uses leftmost derivation to derive an input string.

- **Recursive-descent parser** is a general top down parser.
- A recursive descent parser may require backtracking and repeated scan of input string. working of a recursive descent parser is being explained with the help of an example.

Example : Consider the grammar

$S \rightarrow aXb$

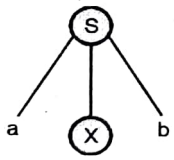
$X \rightarrow ab \mid b$

and the input string abb.

The parse tree of abb can be constructed as given below :

Step 1 : We create a parse tree with single node S. S is the start symbol.

Step 2 : We use the first production $S \rightarrow aXb$ to expand the node



The leftmost leaf, labelled a, matches the first symbol of input string abb. Input pointer is advanced to the next symbol b of abb.

Step 3 : Now, X is expanded using $X \rightarrow ab$ and if it fails to generate remaining input symbols, we backtrack and try the next production $X \rightarrow b$.

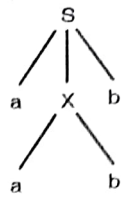


Fig. 6.9.1(a)

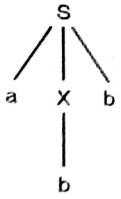


Fig. 6.9.2

The first tree will not generate the input string abb. The leftmost symbol 'a' of subtree, rooted at X does not match the next input symbol 'b'. We must backtrack and try the next production $X \rightarrow b$, as shown in Fig. 6.9.2.

- A recursive-descent parser may enter an infinite loop.
- We can write a predictive parser (without backtracking) by modifying the grammar :
  1. Left recursion should be eliminated.
  2. Apply left factoring to the grammar.

We can easily write a program for a recursive descent parser.

- There must be a function corresponding to each variable.
- A global variable 'position' points to the current input character.
- If the current input character matches the terminal in the production then the next input character is read by advancing the variable 'position' to the next location.

Ex. 6.9.1 : Write a program for recursive descent parser for the following grammar.

E = E + T | T

T = T * F | F

F = (E) | a | b

Soln. :

Step 1 : Removing left recursion from the grammar we get :

$E = TE_1$

$E_1 = +TE_1 \mid \varepsilon$

$T = FT_1$

$T_1 = *FT_1 \mid \varepsilon$

$F = (E) \mid a \mid b$

Step 2 : The program will have a function for each variable.

Function corresponding to $E = TE_1$

```
E()
{T();
E1();
}
```

Function corresponding to $E_1 = + T E_1 | \varepsilon$

```
E1()
{   if (input [position] == '+')
 {  match();
    T();
    E1();
 }
}
```

/*The function match ( ) will increment the current position pointer for input string by 1.*/

Function corresponding to $T = F T_1$

```
T()
{  F();
   T1();
}
```

Function corresponding to $T_1 = *F T_1 | \varepsilon$

```
T1()
{ if(input [position] == '*')
{ match();
  F();
 T1();
 }
}
```

Function corresponding to F = (E) | a | b

```
F()
 {
if(input[position] == 'C')
{ match();
     E ();
if (input[position] == ')')
     match();
else
     error =1;
 }
 else
if(input [position] == 'a' || input [position]
  == 'b')
 match();
}
 // main program
 char input [30];
int error = 0, position = 0;
 void main( )
  {
  printf("\n enter a string :");
  gets(input);
  E();
  if(error == 1 || input [position]! = '\0')
     printf("\n invalid string");
  else
 printf("\n valid string");
  }
```

6.9.2 Bottom-Up Parsing

In bottom up parsing, the source string is reduced to the start symbol of the grammar. The bottom up parsing is also known as shift-reduce parsing.

- A shift reduce parser constructs a parse tree by beginning at the leaves and then working up towards the root.
- A shift reduce parser can be constructed using a stack.

Implementation of shift reduce parser

- A stack is taken to hold grammar symbols.
- An array is taken to store input string to be parsed.
- Initially, the stack is empty.

At each step the parser can take one of the following two steps :

1. **Reduce :** The process of reduction is shown in the Fig. 6.9.3.

If there is a production X → α and the string α is found at the top end of the stack then α should be replaced by X.

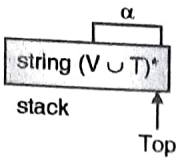


Fig. 6.9.3

2. If the reduction step cannot be carried out then the next input symbol is shifted on the stack.

The parser repeats this cycle of shift-reduce until it has detected an error or until the stack contains the start symbol and the input is empty.

Ex. 6.9.2 : Consider the grammar

E → E + T | T

T → T * F | F

F → (E) | id

and the input string id + id * id. Show the working of the shift reduce parser.

Soln. :

	Stack	Input	Action
1.	empty	id + id *id	Shift
2.	id	+ id * id	Reduce by F → id
3.	F	+ id * id	Reduce by T → F
4.	T	+ id * id	Reduce by E → T
5.	E	+ id * id	Shift
6.	E +	id * id	Shift
7.	E + id	* id	Reduce by F → id
8.	E + F	* id	Reduce by T → F
9.	E + T	* id	Shift
10.	E + T*	id	Shift
11.	E + T * id	–	Reduce by F → id
12.	E + T * F	–	Reduce by T → T * F
13.	E + T	–	Reduce by
14.	E	–	Accept

6.10 University Questions and Answers

May 2014

Q. 1 Give formal definition of a Push Down Automata (PDA). **(Refer Section 6.2)** **(5 Marks)**

Q. 2 Construct a PDA accepting $\{a^n b^m a^n \mid m, n \geq 1\}$ by null store. **(Refer Ex. 6.4.10)** **(10 Marks)**

Dec. 2014

Q. 3 Design a PDA corresponding to the grammar S → aSa | bSb | ∈ **(Refer Ex. 6.6.7)** **(10 Marks)**

May 2015

Q. 4 Design a PDA for detection of even palindrome over {a, b}. **(Refer Ex. 6.5.2)** **(10 Marks)**

Q. 5 Give formal definition of a Push Down Automata (PDA). **(Refer Section 6.2)** **(5 Marks)**

Dec. 2016

Q. 6 Construct deterministic PDA to recognize $a^n abb^n$, n > 0 over {a, b} **(Refer Ex. 6.4.13)** **(10 Marks)**

May 2017

Q. 7 Design a PDA to accept the language

$\{L = a^m b^m c^n \mid m, n, > = 1\}$

(Refer Ex. 6.4.14) **(10 Marks)**

Dec. 2018

Q. 8 Design a PDA for CFL that checks the well formedness of parenthesis i.e. the language L of all "balanced" string of two types of parenthesis say "( )" and "[ ]". Trace the sequence of moves made corresponding to input string (([ ])[ ]). **(Refer Ex. 6.4.15)** **(10 Marks)**

Q. 9 Give formal definition of a Push Down Automata (PDA). **(Refer Section 6.2)** **(5 Marks)**

May 2019

Q. 10 Explain the concept, acceptance by final state of a Pushdown automata with suitable example.

(Refer Section 6.4.1) **(2.5 Marks)**

Q. 11 Explain the concept, acceptance by empty stack of a Pushdown automata with suitable example.

(Refer Section 6.4.2) **(2.5 Marks)**

Q. 12 Construct a PDA for $L = \{a^n bc^m \mid n, m \geq 1 \text{ and } n < m\}$.

(Refer Ex. 6.4.16) **(10 Marks)**

Q. 13 Construct a PDA for the following Context Free Grammar (CFG).

$S \rightarrow CBAA \quad A \rightarrow 0A0 \mid 0$

$B \rightarrow 0B \mid 0 \quad C \rightarrow 0C1 \mid 1C0 \mid \varepsilon$

(Refer Ex. 6.6.8) **(5 Marks)**

Q. 14 Consider the PDA with the following moves :

$\delta(q_0, a, z_0) = \{(q_0, az_0)\}$

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

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

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

$\delta(q_1, \varepsilon, z_0) = \{(q_1, \varepsilon)\}$

Obtain CFG equivalent to PDA.

(Refer Ex. 6.6.12) **(10 Marks)**

Dec. 2019

Q. 15 Design a PDA for CFL that checks the well formedness of parenthesis i.e. the language L of all "balanced" string of two types of parenthesis say "( )" and "[ ]". Trace the sequence of moves made corresponding to input string (([ ])[ ]).

(Refer Ex. 6.4.15) **(10 Marks)**

❑❑❑