Chapter 5
Pushdown Automata

What are we studying in this chapter?

- Definition of the Pushdown automata
- The languages of a PDA
- Equivalence of PDA's and CFG's
- Deterministic Pushdown Automata.

- 7 hours

In the previous chapter we have seen how a context free language can be described using context free grammars. But, we have not encountered an automaton to recognize the context free languages as we had finite automata to recognize the regular languages. Now, the question is "Is there any need for pushdown automata when finite automata already exists?"

A DFA (or NFA) is not powerful enough to recognize many context-free language. Since the finite automaton have strict finite memories, it is not possible to construct those machines to accept the context free languages. The automaton to recognize the CFL may require additional amount of storage which will be used to store the data.

- For example, during parentheses matching, if we encounter '(', it is required to store the left parentheses on the stack and as we encounter right parentheses i.e., ')' we may have to pop the corresponding '(' from the stack. Thus, stack is used to remember the scanned symbols. This is evident as we design the automaton to accept CFL.

- Sometimes, it is required to count the symbols also. For example, if we have the language $L = \{a^n b^n \mid n \geq 0\}$ after reading n number of a's, we should see that n number of b's exist. So, an automaton that we construct to accept CFL's should be in a position to count at the same time should have sufficient memory to hold the string scanned.

Since the DFA's or NFA's can not count and can not store the input for future reference, we are forced to have a new machine called **Pushdown Automaton (PDA)**. Note that **PDA** is a Finite Automata with the exception that PDA has an extra stack. So, the definition of PDA is similar to the definition of NFA with slight changes. Let us take some typical examples and define the PDA.

Example 5.1: Consider the language $L = \{wCw^R \mid w \in (a + b)^*\}$ where w^R is reserve of w and $C \in \Sigma$ indicates middle of string.

It is clear from the language $L = wCw^R$ that if w is *abb* then w^R is bba which is reverse of w. The language generated will be abbCbba which is a palindrome. So, the language generates strings of palindromes.

Design: Let us device a method to accept a string of palindrome. In a given string, the letter C is the middle of the string. To start with, machine will be in the start state say q_0. Now keep pushing all input symbols on to the stack and stay in state q_0 till the input symbol C is encountered.

Immediately after scanning the input symbol C, change the state to q_1. Now, we have passed the middle of the string. If the given string is a palindrome, for each character encountered after the midpoint, there will be a corresponding character on the stack.

So, in state q_1, after scanning the input symbol, compare it with most recently pushed symbol on the stack. If they are same, discard both the symbols and scan the next symbol and compare it with the symbol on the stack and repeat the process and remain in state q_1. If there is a mismatch the machine halts and the string will be rejected. Once the last symbol in the input is matched with symbol on the stack, finally stack will be empty. Once the stack is empty, it is evident that the given string is a palindrome. So, change the state to q_2 which is an accepting state.

Note: It is clear from this example that, the PDA has set of states Q and set of input symbols Σ. Since it has stack also as a component, some symbols will be pushed on to the stack. So, the stack has stack alphabets denoted by Γ. The stack contains a special symbol Z_0 which is present in the stack initially (Note that this is the initial condition). It means that an empty stack contains Z_0 as the top of the stack.

5.1 Transitions

For the example 5.1 shown above, the transitions can be written as shown in table 5.1. Assume that q_0 is the start state and Z_0 is the initial symbol on the stack indicating stack is empty.

Since there is an ε-transition, the PDA is actually Non-deterministic. In a non-deterministic PDA, there will be ε-transitions or there may be multiple transitions from the same state on a given input when a specified symbol is there on the stack.

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

δ: $Q \times (\Sigma \cup \varepsilon) \times \Gamma$ to $Q \times \Gamma^*$

Table 5.1 Transitions

For example, consider the transitions

$$\delta(q_0, a, Z) = (q_1, bZ)$$
$$\delta(q_0, a, Z) = (q_2, cZ)$$

These transitions can also be written as

$$\delta(q_0, a, Z) = \{ (q_1, bZ), (q_2, cZ) \}$$

In the above transition, when the PDA is currently in state q_0, on scanning the input symbol *a* and when the top of the stack contains Z, the machine can perform one of the transitions defined i.e., either

1. The PDA can go to state q_1 after pushing the symbol *b* on to the stack

or

2. The PDA can go to state q_2 after pushing the symbol *c* on to the stack

Since there is an ε-transition, the PDA is actually Non-deterministic. In a non-deterministic PDA, there will be ε-transitions or there may be multiple transitions from the same state on a given input when a specified symbol is there on the stack.

$\delta(q_0, a, Z_0)$	=	(q_0, aZ_0)
$\delta(q_0, b, Z_0)$	=	(q_0, bZ_0)
$\delta(q_0, a, a)$	=	(q_0, aa)
$\delta(q_0, b, a)$	=	(q_0, ba)
$\delta(q_0, a, b)$	=	(q_0, ab)
$\delta(q_0, b, b)$	=	(q_0, bb)
$\delta(q_o, c, Z_0)$	=	(q_1, Z_0)
$\delta(q_0, c, a)$	=	(q_1, a)
$\delta(q_0, c, b)$	=	(q_1, b)
$\delta(q_1, a, a)$	=	(q_1, ε)
$\delta(q_1, b, b)$	=	(q_1, ε)
$\delta(q_1, \varepsilon, Z_0)$	=	(q_2, Z_0)

δ: $Q \times (\Sigma \cup \varepsilon) \times \Gamma$ to $Q \times \Gamma^*$

Table 5.1 Transitions

For example, consider the transitions

$$\delta(q_0, a, Z) = (q_1, bZ)$$
$$\delta(q_0, a, Z) = (q_2, cZ)$$

These transitions can also be written as

$$\delta(q_0, a, Z) = \{ (q_1, bZ), (q_2, cZ) \}$$

In the above transition, when the PDA is currently in state q_0, on scanning the input symbol *a* and when the top of the stack contains Z, the machine can perform one of the transitions defined i.e., either

1. The PDA can go to state q_1 after pushing the symbol *b* on to the stack

or

2. The PDA can go to state q_2 after pushing the symbol *c* on to the stack

5.4 Pushdown Automata

Unless otherwise specified, let us call **Push Down Automata (PDA)** as **Non Deterministic Push Down Automata (NPDA)**. Now, let us see "What is pushdown automata?" The formal definition of PDA is shown below:

Definition: A pushdown automata (PDA) is a seven tuple

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

where

Q - is set of finite states.
Σ - Set of input alphabets.
Γ - Set of stack alphabets.
δ - transition from $Q \times (\Sigma \cup \varepsilon) \times \Gamma$ to finite sub set of $Q \times \Gamma^*$
δ is called the transition function of M.
$q_0 \in Q$ is the start state of machine.
$Z_0 \in \Gamma$ is the initial symbol on the stack.
$F \subseteq Q$ is set of final states.

The actions (i.e., transitions) performed by the PDA depends on

1. The current state.
2. The next input symbol
3. The symbol on top of the stack.

The action performed by the machine consists of

1. Changing the states from one state to another
2. Replacing the symbol on the stack.

Note: In general, the transition function accepts three parameters namely a state, an input symbol and stack symbol and returns a new state after changing the top of the stack i.e. the transition function has the form:

δ (state, input_symbol, stack_symbol) = (next_state, stack_symbol)

Example 5.2: What does each of the following transitions represent?

a. $\delta(p, a, Z) = (q, aZ)$
b. $\delta(p, a, Z) = (q, \varepsilon)$
c. $\delta(p, a, Z) = (q, r)$
d. $\delta(p, \varepsilon, Z) = (q, r)$
e. $\delta(p, \varepsilon, \varepsilon) = (q, Z)$
f. $\delta(p, \varepsilon, Z) = (q, \varepsilon)$

The transition

$$\delta(p, a, Z) = (q, aZ)$$

means that the PDA in current state p after scanning the input symbol $a \in \Sigma$ and if $Z \in \Gamma$ is on top of the stack, then the PDA enters into new state q pushing $a \in \Sigma$ on to the stack. The transition

$$\delta(p, a, Z) = (q, \varepsilon)$$

means that in state p, on scanning the input symbol a, and when top of this stack is Z, the machine enters into state q and the topmost symbol Z is deleted from the stack. The transition

$$\delta(p, a, Z) = (q, r)$$

means that in state p, on scanning the input symbol a and when top of the stack is Z, the machine enters into state q and topmost symbol Z on the stack is replaced by r. The transition

$$\delta(p, \in, Z) = (q, r)$$

means that in state p, on scanning the empty string and when top of the stack is Z, the machine enters into q and topmost symbol Z on the stack is replaced by r. The transition

$$\delta(p, \varepsilon, \varepsilon) = (q, Z)$$

means that in state p, on scanning the empty string and when top of the stack empty, the machine enters into q pushing the symbol Z on the stack . The transition

$$\delta(p, \varepsilon, Z) = (q, \varepsilon)$$

means that in state p, on scanning the empty string and when top of the stack is Z, the machine enters into q and topmost symbol Z on the stack is deleted.

Note: δ is a transition function with three arguments. The first two are same as NFA i.e., the state and either ε or a symbol from the input alphabet. The third argument is the symbol on top of the stack. As the input symbol is *consumed* when the transition(or function) is applied, the symbol on top of the stack may be deleted or it

may be altered or some times one or more items may be pushed on to the stack (depends on the problem being solved).

The move of a machine can be defined as follows.

Definition: Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA. The move of machine M

$$\delta(p, a, Z) = (q, \alpha)$$

imply that, when the PDA is currently in state p, if the scanned input symbol is *a* and the top of the stack is Z, the PDA enters into new state q and pushes Z on the top of the stack. The symbol on top of the stack and the recently pushed symbol is denoted by α. Here $p, q \in Q, a \in \Sigma, Z \in \Gamma$ and $\alpha \in \Gamma^*$.

5.2 Graphical representation of a PDA

The various actions performed by a PDA in state *q* on an input symbol *a* and when the top of the stack represented by Z can be represented using two methods:

1. Using δ notation (as discussed in previous section)
2. Using graphical representation

Now, let us discuss how a PDA is represented graphically using the notations that have been used to represent an FA in which:

a. The states of the PDA correspond to the nodes in a graph and are represented using circles
b. The start state of the PDA is denoted by an arrow labeled *start*
c. The nodes of the graph represented by two concentric circles are the final states of the PDA
d. If there is a transition of the form

$$\delta(p, a, Z) = (q, \alpha)$$

then, there will be an arc from state *p* to state *q* and the arc is labeled with

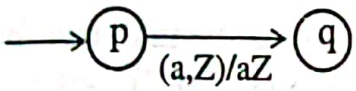
$$a, Z/\alpha$$

indicating *a* is the current symbol, Z is the symbol on top of the stack and α represent the top of the stack along with the recently pushed symbol.

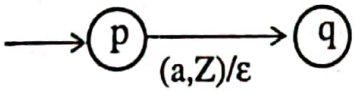
Example 5.3: For each of the transitions obtain the corresponding transition diagrams?

a. $\delta(p, a, Z) = (q, aZ)$
b. $\delta(p, a, Z) = (q, \varepsilon)$
c. $\delta(p, a, Z) = (q, r)$
d. $\delta(p, \varepsilon, Z) = (q, r)$
e. $\delta(p, \varepsilon, \varepsilon) = (q, Z)$
f. $\delta(p, \varepsilon, Z) = (q, \varepsilon)$

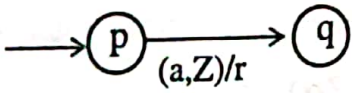
The transition $\delta(p, a, Z) = (q, aZ)$ means that the PDA in current state p after scanning the input symbol $a \in \Sigma$ and if $Z \in \Gamma$ is on top of the stack, then the PDA enters into new state q pushing $a \in \Sigma$ on to the stack. The graphical representation is shown below:



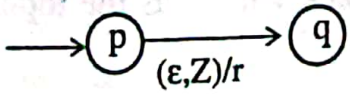
The transition $\delta(p, a, Z) = (q, \varepsilon)$ means that in state p, on scanning the input symbol a, and when top of this stack is Z, the machine enters into state q and the topmost symbol Z is deleted from the stack. The corresponding graphical representation is:



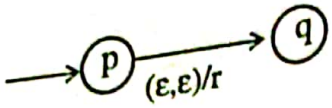
The transition $\delta(p, a, Z) = (q, r)$ means that in state p, on scanning the input symbol a and when top of the stack is Z, the machine enters into state q and topmost symbol Z on the stack is replaced by r. The corresponding graphical notation is shown below:



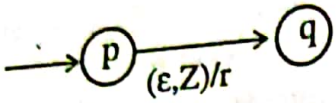
The transition $\delta(p, \varepsilon, Z) = (q, r)$ means that in state p, on scanning the empty string and when top of the stack is Z, the machine enters into q and topmost symbol Z on the stack is replaced by r and the equivalent graphical representation is shown below:



The transition $\delta(p, \varepsilon, \varepsilon) = (q, Z)$ means that in state p, on scanning the empty string and when top of the stack empty, the machine enters into q pushing the symbol Z on the stack. The equivalent graphical representation is shown below:



The transition $\delta(p, \varepsilon, Z) = (q, \varepsilon)$ means that in state p, on scanning the empty string and when top of the stack is Z, the machine enters into q and top most symbol Z on the stack is deleted. The graphical representation for this is shown below:



5.3 Instantaneous description

The current configuration of PDA at any given instant can be described by an *instantaneous description* (in short we can call ID). An ID gives the current state of the PDA, the remaining string to be processed and the entire contents of the stack. Thus, an instantaneous description ID can be defined as shown below.

Definition: Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA. An ID (instantaneous description) is defined as 3-tuple or a triple

$$(q, w, \alpha)$$

where q is the current state, *w* is the string to be processed and α is the current contents of stack. The leftmost symbol in the string α is on top of the stack and rightmost symbol in α is at the bottom of the stack.

Let the current configuration of PDA be

$$(q, aw, Z\alpha)$$

It means

- q - is the current state
- aw - is the string to be processed
- $Z\alpha$ - is the current contents of the stack with Z as the topmost symbol on the stack.

If the transition defined is $\delta(q, a, Z) = (p, \beta)$ then the new configuration obtained will be

$$(p, w, \beta\alpha)$$

The move from the current configuration to next configuration is given by

$$(q, aw, Z\alpha) \vdash (p, w, \beta\alpha)$$

This can be read as "the configuration $(q, aw, Z\alpha)$ derives $(p, w, \beta\alpha)$ in one move". If an arbitrary number of moves are used to move from one configuration to another configuration then the moves are denoted by the symbol $\vdash^*$ and $\vdash^+$.

For example,

$$(q, aw, Z\alpha) \vdash^* (p, w, \beta\alpha)$$

means that the current configuration of the PDA will be $(q, aw, Z\alpha)$ and after applying zero or more number of transitions, the PDA enters into new configuration $(p, w, \beta\alpha)$.

Note: The instantaneous description $(q, aw, Z\alpha) \vdash^+ (p, w, \beta\alpha)$ indicates that the configuration $(p, w, \beta\alpha)$ is obtained from the configuration $(q, aw, Z\alpha)$ by applying one or more transitions.

5.4 Acceptance of a language by PDA

There are two cases wherein a string *w* is accepted by a PDA.

- Get the final state from the start state.
- Get an empty stack from the start state

In the first case, we say that the language is accepted by a **final state** and in the second case we say that the language is accepted by an **empty stack** or **null stack**. The formal definitions to accept the string by a final state and by an empty stack are defined as follows.

Definition: Let $M = (Q, \Sigma, \Gamma, \delta, q_0, Z_0, F)$ be a PDA. The language L(M) accepted by a final state is defined as

$$L(M) = \{w \mid (q_0, w, Z_0) \vdash^* (p, \varepsilon, \alpha)\}$$

for some $\alpha \in \Gamma^*$, $p \in F$ and $w \in \Sigma^*$. It means that the PDA, currently in state q_0, after scanning the string w enters into a final state p. Once the machine is in state p, the input symbol should be ε and the contents of the stack are irrelevant. Any thing can be there on the stack. The only point to remember here is that when all the symbols in string w have been read and when the machine is in the final state the final contents of the stack are irrelevant.

We can also define a language N(M) accepted by an empty stack (Null stack) as

$$N(M) = \{w \mid (q_0, w, Z_0) \vdash^* (p, \varepsilon, \varepsilon)\}$$

for $w \in \Sigma^*$, $q_0, p \in Q$. It means that when the string w is accepted by an empty stack, the final state is irrelevant, the input should be completely read and the stack should be empty. Here the state p is not the final state, only thing is that the string w should be completely read and stack should be empty.

5.5 Construction of PDA

Now, so far we have seen some concepts on PDA. In this section, we shall see how PDA's can be constructed.

Example 5.4: Obtain a PDA to accept the language $L(M) = \{wCw^R \mid w \in (a + b)^*\}$ where W^R is reveres of W by a final state.

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

$$w = abb$$

then reverse of w denoted by w^R will be

$$w^R = bba$$

and the language L will be wCw^R i.e.,

$$abbCbba$$

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

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

Step1: Input symbols can be *a* or *b*.
Let q_0 be the initial state and Z_0 be the initial symbol on the stack. In state q_0 and when top of the stack is Z_0, whether the input symbol is *a* or *b* push it on to the stack, and remain in q_0. The transitions defined for this can be of the form

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

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

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

Step 2: Input symbol is C.
Now, if the next input symbol is C, the top of the stack may be *a* or *b*. Another possibly is that, in state q_0, the first symbol itself can be C. In this case *w* is null string and Z_0 will be on the stack. In all these cases, the input symbol is C i.e., the symbol which is present in the middle of the string. So, change the state to q_1 and do not alter the contents of the stack. The transitions defined for this can be of the form

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

Now, we have passed the middle of the string.

Step3: Input symbols can be *a* or *b*.
To be a palindrome, for each input symbol there should be a corresponding symbol (same as input symbol) on the stack. So, whenever the input symbol is same as symbol on the stack, remain in state q_1 and delete that symbol from the stack and repeat the process. The transitions defined for this can be of the form

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

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

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

So, the PDA M to accept the language

$$L(M) = \{wCw^R \mid w \in (a,b)^*\}$$

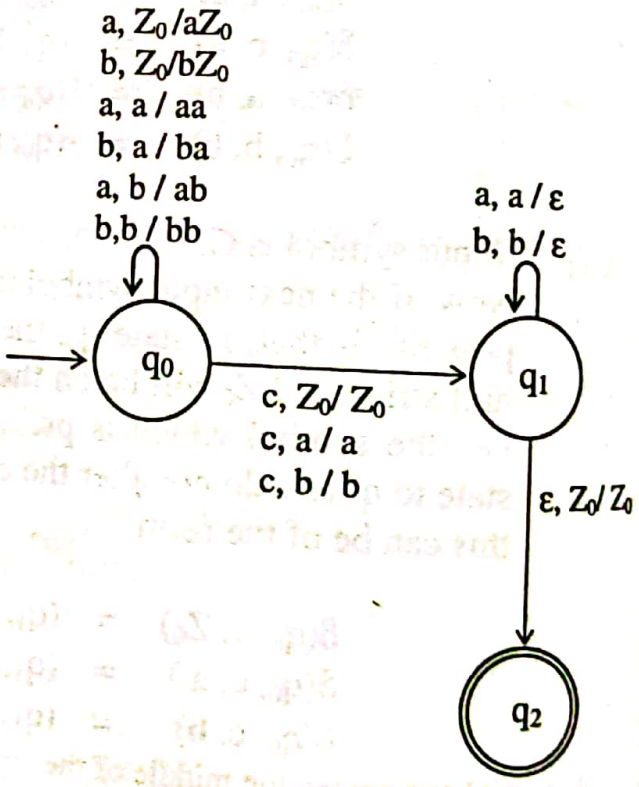
along with transition graph is given by

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

where

$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{a, b, C\}$
$\Gamma = \{a, b, Z_0\}$
δ : is shown below.

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



$q_0 \in Q$ is the start state of machine.
$Z_0 \in \Gamma$ is the initial symbol on the stack.
$F = \{q_2\}$ is the final state.

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

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

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

So, the PDA M to accept the language

$$L(M) = \{wCw^R \mid w \in (a,b)^*\}$$

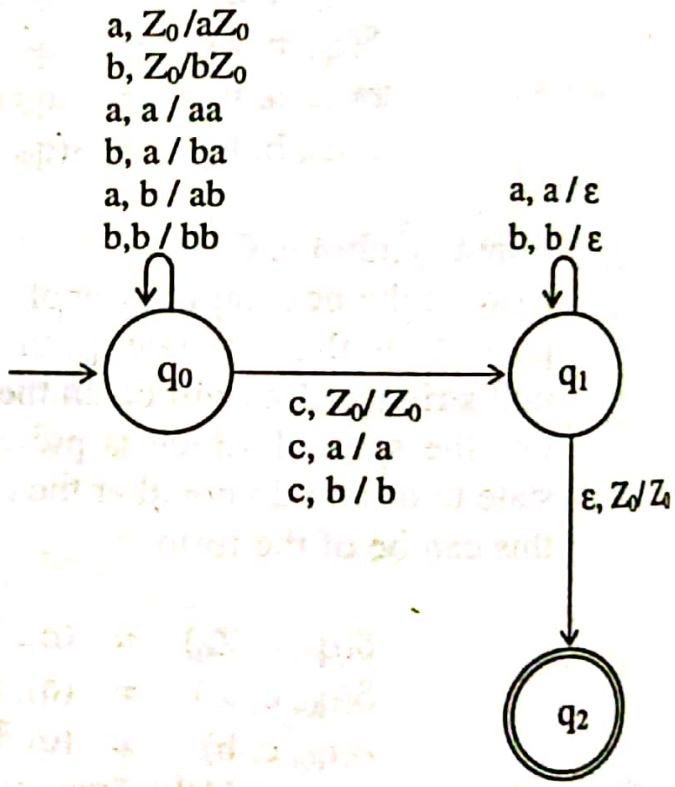
along with transition graph is given by

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

where

$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{a, b, C\}$
$\Gamma = \{a, b, Z_0\}$
δ : is shown below.

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



$q_0 \in Q$ is the start state of machine.
$Z_0 \in \Gamma$ is the initial symbol on the stack.
$F = \{q_2\}$ is the final state.

To accept the string: The sequence of moves made by the PDA for the string aabCbaa is shown below.

Initial ID
$(q_0, aabCbaa, Z_0)$ $\vdash (q_0, abCbaa, aZ_0)$
$\vdash (q_0, bCbaa, aaZ_0)$
$\vdash (q_0, Cbaa, baaZ_0)$
$\vdash (q_1, baa, baaZ_0)$
$\vdash (q_1, aa, aaZ_0)$
$\vdash (q_1, a, aZ_0)$
$\vdash (q_1, \varepsilon, Z_0)$
$\vdash (q_2, \varepsilon, Z_0)$
(Final Configuration)

Since q2 is the final state and input string is ε in the final configuration, the string

aabCbaa

is accepted by the PDA.

To reject the string: The sequence of moves made by the PDA for the string aabCbab is shown below.

Initial ID
$(q_0, aabCbab, Z_0)$ $\vdash (q_0, abCbab, aZ_0)$
$\vdash (q_0, bCbab, aaZ_0)$
$\vdash (q_0, Cbab, baaZ_0)$
$\vdash (q_1, bab, baaZ_0)$
$\vdash (q_1, ab, aaZ_0)$
$\vdash (q_1, b, aZ_0)$
(Final Configuration)

Since the transition $\delta(q_1, b, a)$ is not defined, the string

aabCbab

is not a palindrome and the machine halts and the string is rejected by the PDA.

Note: The same problem can be converted to accept the language by an empty stack. Only the change is, instead of the final transition namely

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

replace it by the transition

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

When a language is accepted by an empty stack, finally the stack should not contain any thing including Z_0. Note that q_1 is not a final state. There is no final state.

Example 5.5: Obtain a PDA to accept the language $L = \{a^n b^n \mid n \geq 1\}$ by a final state.

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

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

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

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

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

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

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

and delete the corresponding *a* from the stack. The transitions defined for this can be of the form

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

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

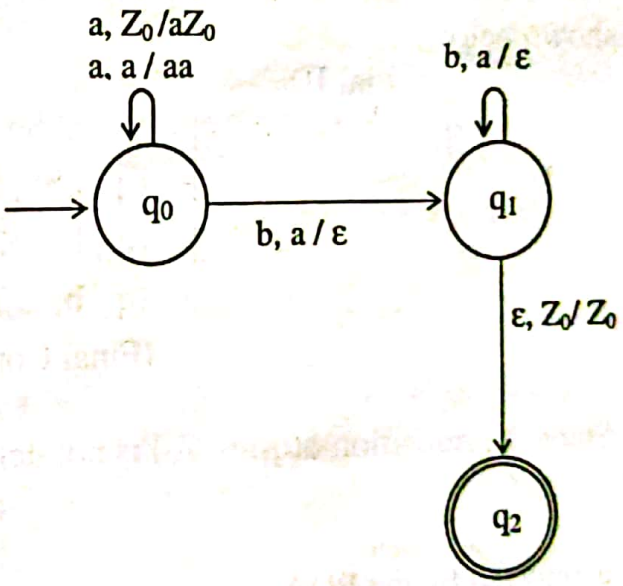
$$\delta(q_1, \varepsilon, Z_0) = (q_2, Z_0)$$

So, the PDA to accept the language

$$L \triangleq \{a^n b^n \mid n \geq 1\}$$

along with transition diagram is given by

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



where

$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{a, b\}$
$\Gamma = \{a, Z_0\}$
δ : is shown below.

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

$q_0 \in Q$ is the start state of machine.
$Z_0 \in \Gamma$ is the initial symbol on the stack.
$F = \{q_2\}$ is the final state.

To accept the string: The sequence of moves made by the PDA for the string aaabbb is shown below.

Initial ID
$(q_0, aaabbb, Z_0)$ $\vdash (q_0, aabbb, aZ_0)$
$\vdash (q_0, abbb, aaZ_0)$
$\vdash (q_0, bbb, aaaZ_0)$
$\vdash (q_1, bb, aaZ_0)$
$\vdash (q_1, b, aZ_0)$
$\vdash (q_1, \varepsilon, Z_0)$
$\vdash (q_2, \varepsilon, Z_0)$
(Final Configuration)

Since q2 is the final state and input string is ε in the final configuration, the string
aaabbb

is accepted by the PDA.

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

Initial ID
$(q_0, aabbb, Z_0)$ $\vdash (q_0, abbb, aZ_0)$
$\vdash (q_0, bbb, aaZ_0)$
$\vdash (q_1, bb, aZ_0)$
$\vdash (q_1, b, Z_0)$
(Final Configuration)

Since the transition $\delta(q_1, b, Z_0)$ is not defined, the string
aabbb

is rejected by the PDA.

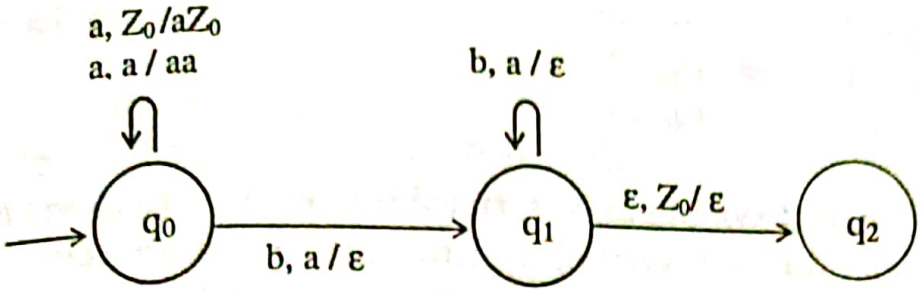
Note: By changing the final transition from

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

to

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

the PDA accepted by an empty stack is obtained. Note that q_2 is not the final state. The corresponding transition diagram accepting by an empty stack is shown below:



Example 5.6: Obtain a PDA to accept the language L(M) = {w | w ∈ (a+b)* and $n_a(w) = n_b(w)$ by a final state i.e., number of a's in string w should be equal to number of b's in w.

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

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

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

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

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

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

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

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

So, the PDA to accept the language

$$L = \{w \mid n_a(w) = n_b(w)\}$$

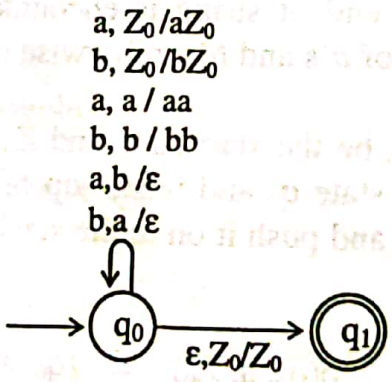
is given by

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

where

$Q = \{q_0, q_1\}$
$\Sigma = \{a, b\}$
$\Gamma = \{a, b, Z_0\}$
δ : is shown below.

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



$q_0 \in Q$ is the start state of machine.
$Z_0 \in \Gamma$ is the initial symbol on the stack.
$F = \{q_1\}$ is the final state.

To accept the string: The sequence of moves made by the PDA for the string abbbaa is shown below.

Initial ID
$(q_0, abbbaa, Z_0)$ $\vdash (q_0, bbbaa, aZ_0)$
$\vdash (q_0, bbaa, Z_0)$
$\vdash (q_0, baa, bZ_0)$
$\vdash (q_0, aa, bbZ_0)$
$\vdash (q_0, a, bZ_0)$
$\vdash (q_0, \varepsilon, Z_0)$
$\vdash (q_1, \varepsilon, Z_0)$
(Final Configuration)

Since q_1 is the final state and input string is ε in the final configuration, the string

abbbaa

is accepted by the PDA.

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

Initial ID
$(q_0, aabbb, Z_0)$ $\vdash (q_0, abbb, aZ_0)$
$\vdash (q_0, bbb, aaZ_0)$
$\vdash (q_0, bb, aZ_0)$
$\vdash (q_0, b, Z_0)$
$\vdash (q_0, \varepsilon, bZ_0)$
(Final Configuration)

Since the transition $\delta(q_0, \varepsilon, b)$ is not defined, the string

aabbb

is rejected by the PDA.

Note: To accept the language by an empty stack, the final state is irrelevant where as the next input symbol to be scanned should be ε and stack should to be empty. Even Z_0 should not be there on the stack. So, to obtain the PDA to accept equal number of *a*'s and *b*'s using an empty stack, change only the last transition in example 5.5. The last transition

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

can be changed as

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

So, the PDA to accept the language

$$L = \{w \mid n_a(w) = n_b(w)\}$$

by an empty stack is given by

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

where

$Q = \{q_0, q_1\}$
$\Sigma = \{a, b\}$
$\Gamma = \{a, b, Z_0\}$
δ : is shown below.

$$\begin{aligned}
\delta(q_0, a, Z_0) &= (q_0, aZ_0)\\
\delta(q_0, b, Z_0) &= (q_0, bZ_0)\\
\delta(q_0, a, a) &= (q_0, aa)\\
\delta(q_0, b, b) &= (q_0, bb)\\
\delta(q_0, a, b) &= (q_0, \varepsilon)\\
\delta(q_0, b, a) &= (q_0, \varepsilon)\\
\delta(q_0, \varepsilon, Z_0) &= (q_1, \varepsilon)
\end{aligned}$$

$q_0 \in Q$ is the start state of machine.
$Z_0 \in \Gamma$ is the initial symbol on the stack.
$F = \{\phi\}$. Note that PDA is accepted by an empty stack

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

Initial ID
$(q_0, aabbab, Z_0)$ $\vdash (q_0, abbab, aZ_0)$
$\vdash (q_0, bbab, aaZ_0)$
$\vdash (q_0, bab, aZ_0)$
$\vdash (q_0, ab, Z_0)$
$\vdash (q_0, b, aZ_0)$
$\vdash (q_0, \varepsilon, Z_0)$
$\vdash (q_1, \varepsilon, \varepsilon)$
(Final Configuration)

Since the next input symbol to be scanned is ε and the stack is empty, the string aabbab is accepted by an empty stack.

Note: A PDA accepting by a final state and by an empty stack are equivalent. They can be obtained from each other i.e., if we know a PDA by a final state, we can obtain a PDA by an empty stack and vice versa.

Example 5.7: Obtain a PDA to accept a string of balanced parentheses. The parentheses to be considered are (,), [,]

Note 1: Some of the valid strings are:

[( )( )([ ])], ε, [ ][ ]( )

and invalid strings are:

[ ) ( ) [ ],) (, [ )

Note 2: ε (null string) is valid

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

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

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

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

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

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

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

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

Step 5: When top of the stack is Z_0, it indicates that so far all the parentheses have been matched. At this point, on ε-transition, the PDA enters into state q0 and all the steps from step1 are repeated. The transition for this can be of the form

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

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

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

where

$Q = \{q_0, q_1\}$
$\Sigma = \{(,), [,]\}$
$\Gamma = \{(, [, Z_0\}$
δ : is shown below.

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

$q_0 \in Q$ is the start state of machine.
$Z_0 \in \Gamma$ is the initial symbol on the stack.
$F = \{q_0\}$. Note that even ε is accepted by PDA and is valid.

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

Initial ID
$(q_0, [(\,)(\,)(\,[\,]\,)\,], Z_0)$ $\vdash$ $(q_1, (\,)(\,)(\,[\,]\,)\,], [Z_0)$
$\vdash$ $(q_1, \;)(\,)(\,[\,]\,)\,], ([Z_0)$
$\vdash$ $(q_1, \;\;(\,)(\,[\,]\,)\,], [Z_0)$
$\vdash$ $(q_1, \;\;\;)(\,[\,]\,)\,], ([Z_0)$
$\vdash$ $(q_1, \;\;\;\;(\,[\,]\,)\,], [Z_0)$
$\vdash$ $(q_1, \;\;\;\;\;[\,]\,)\,], ([Z_0)$
$\vdash$ $(q_1, \;\;\;\;\;\;]\,)\,], [([Z_0)$
$\vdash$ $(q_1, \;\;\;\;\;\;\;)\,], ([Z_0)$
$\vdash$ $(q_1, \;\;\;\;\;\;\;\;], [Z_0)$
$\vdash$ (q_1, ε, Z_0)
$\vdash$ (q_0, ε, Z_0)
(Final Configuration)

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

Example 5.8: Obtain a PDA to accept the language $L = \{w \mid w \in (a, b)^*$ and $n_a(w) > n_b(w)\}$

Note: The solution is similar to that of the problem discussed in example 5.5 in which we are accepting strings of *a*'s and *b*'s of equal numbers. When we encounter end of the input i.e., ε and top of the stack is Z_0, it has equal number of *a*'s and *b*,s. But, what we want is a machine to accept more number of *a*'s than *b*'s. For this, only change to be made is that when we encounter ε (i.e., end of the input), if top of the stack contains at least one *a*, then change the final state to q1 and do not alter the contents of the stack. The transition defined remains same as problem shown in example 5.5, except the last transition. The last transition is of the form

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

So, the PDA to accept the language

$$L = \{w \mid n_a(w) > n_b(w)\}$$

is given by

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

where

$Q = \{q_0, q_1\}$
$\Sigma = \{a, b\}$
$\Gamma = \{a, b, Z_0\}$
δ : is shown below.

$$\begin{aligned}
\delta(q_0, a, Z_0) &= (q_0, aZ_0)\\
\delta(q_0, b, Z_0) &= (q_0, bZ_0)\\
\delta(q_0, a, a) &= (q_0, aa)\\
\delta(q_0, b, b) &= (q_0, bb)\\
\delta(q_0, a, b) &= (q_0, \varepsilon)\\
\delta(q_0, b, a) &= (q_0, \varepsilon)\\
\delta(q_0, \varepsilon, a) &= (q_1, a)
\end{aligned}$$

$q_0 \in Q$ is the start state of machine.
$Z_0 \in \Gamma$ is the initial symbol on the stack.
$F = \{q_1\}$ is the final state.

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

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

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

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

to

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

So, the PDA to accept the language

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

is given by

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

where

$Q = \{q_0, q_1\}$
$\Sigma = \{a, b\}$
$\Gamma = \{a, b, Z_0\}$
δ : is shown below.

$$\begin{aligned}
\delta(q_0, a, Z_0) &= (q_0, aZ_0)\\
\delta(q_0, b, Z_0) &= (q_0, bZ_0)\\
\delta(q_0, a, a) &= (q_0, aa)\\
\delta(q_0, b, b) &= (q_0, bb)\\
\delta(q_0, a, b) &= (q_0, \varepsilon)\\
\delta(q_0, b, a) &= (q_0, \varepsilon)\\
\delta(q_0, \varepsilon, b) &= (q_1, b)
\end{aligned}$$

$q_0 \in Q$ is the start state of machine.
$Z_0 \in \Gamma$ is the initial symbol on the stack.
$F = \{q_1\}$ is the final state.

Example 5.9: Obtain a PDA to accept the language $L = \{a^n b^{2n} \mid n \geq 1\}$

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

General Procedure Since *n* number of *a*'s should be followed by 2*n* number of *b*'s, for each *a* in the input, push two *a*'s on to the stack. Once we encounter *b*'s, we should see that for each *b* in the input, there should be corresponding *a* on the stack.

When the input pointer reaches the end of the string, the stack should be empty. If stack is empty, it indicates that the string scanned has n number of a's followed by $2n$ number of b's.

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

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

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

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

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

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

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

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

So, the PDA to accept the language

$$L = \{a^n b^{2n} \mid n \geq 1\}$$

is given by

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

where

$Q = \{q_0, q_1, q_2\}$
$\Sigma = \{a, b\}$
$\Gamma = \{a, Z_0\}$
δ : is shown below.

$\delta(q_0, a, Z_0) = (q_0, aaZ_0)$
$\delta(q_0, a, a) = (q_0, aaa)$
$\delta(q_0, b, a) = (q_1, \varepsilon)$
$\delta(q_1, b, a) = (q_1, \varepsilon)$
$\delta(q_1, \varepsilon, Z_0) = (q_2, \varepsilon)$

$q_0 \in Q$ is the start state of machine.
$Z_0 \in \Gamma$ is the initial symbol on the stack.
$F = \{q_2\}$ is the final state.

To accept the string: The sequence of moves made by the PDA for the string aabbbb is shown below.

Initial ID
$(q_0, aabbbb, Z_0) \vdash (q_0, abbbb, aaZ_0)$
$\vdash (q_0, bbbb, aaaaZ_0)$
$\vdash (q_0, bbb, aaaZ_0)$
$\vdash (q_1, bb, aaZ_0)$
$\vdash (q_1, b, aZ_0)$
$\vdash (q_1, \varepsilon, Z_0)$
$\vdash (q_2, \varepsilon, Z_0)$
(Final Configuration)

Since q_2 is the final state and input string is ε in the final configuration, the string

aabbbb

is accepted by the PDA.

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

Initial ID
$(q_0, aabbb, Z_0) \vdash (q_0, abbb, aaZ_0)$
$\vdash (q_0, bbb, aaaZ_0)$
$\vdash (q_0, bb, aaZ_0)$
$\vdash (q_0, b, aZ_0)$
$\vdash (q_0, \varepsilon, aZ_0)$
(Final Configuration)

Since the transition $\delta(q_0, \varepsilon, a)$ is not defined, the string

aabbb

is rejected by the PDA.

Example 5.10: Obtain a PDA to accept the language $L = \{ww^R \mid w \in (a + b)^*\}$ by a final state

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

$w = abb$

then reverse of w denoted by w^R will be

$w^R = bba$

and the language L will be ww^R i.e.,

abbbba

which is a string of palindrome.

So, we have to construct a PDA which accepts a palindrome consisting of *a*'s anb *b*'s. This problem is similar to the problem discussed in example 5.3. Only difference is that in example 5.3, an extra symbol C acts as a pointer to the middle string. But, here there is no way to find the mid point for the string.

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

Step1: Let q_0 be the initial state and Z_0 be the initial symbol on the stack. In state q_0 and when top of the stack is Z_0, whether the input symbol is *a* or *b* push it on to the stack, and remain in q_0. The transitions defined for this can be of the form

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

Note: In state q_0, if we encounter ε, the empty string has to be accepted and we should be in a position to reach the final state and so, we may have one more transition of the form

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

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

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

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

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

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

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

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

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

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

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

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

where

$Q = \{q_0, q_1, q_2\}$

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

$\Gamma = \{a, b, Z_0\}$

δ : is shown below.

	$\delta(q_0, \varepsilon, Z_0)$	=	(q_1, Z_0)
	$\delta(q_0, a, Z_0)$	=	(q_0, aZ_0)
	$\delta(q_0, b, Z_0)$	=	(q_0, bZ_0)
3	$\delta(q_0, a, a)$	=	(q_0, aa)
	$\delta(q_0, b, a)$	=	(q_0, ba)
	$\delta(q_0, a, b)$	=	(q_0, ab)
6	$\delta(q_0, b, b)$	=	(q_0, bb)
7	$\delta(q_0, a, a)$	=	(q_1, ε)
8	$\delta(q_0, b, b)$	=	(q_1, ε)
	$\delta(q_1, a, a)$	=	(q_1, ε)
	$\delta(q_1, b, b)$	=	(q_1, ε)
	$\delta(q_1, \varepsilon, Z_0)$	=	(q_2, Z_0)

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

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

$F = \{q2\}$ is the final state.

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

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

Note that once the following transitions are applied

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

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

To accept the string: The sequence of moves made by the PDA for the string aabbaa is shown below.

Initial ID

$(q_0, aabbaa, Z_0)$

$(q_0, abbaa, aZ_0)$ $(q_1, aabbaa, Z_0)$ $(q_2, aabbaa, Z_0)$ (reject)

$(q_0, bbaa, aaZ_0)$ $(q_1, bbaa, Z_0) \rightarrow$ $(q_2, bbaa, Z_0)$ (reject)

$(q_0, baa, baaZ_0)$

(q_1, aa, aaZ_0) $(q_0, aa, bbaaZ_0) \rightarrow$ $(q_0, a, abbaaZ_0)$ $(q_0, \varepsilon, abbaaZ_0)$ (reject)

(q_1, a, aZ_0) $(q_1, \varepsilon, bbaaZ_0)$ (reject)

$(q_1, \varepsilon, Z_0) \rightarrow$ (q_2, ε, Z_0) (accept)

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

5.6 Deterministic and Non-deterministic PDA

In the example 5.10, we have seen that there can be multiple transitions defined from a state. The PDA defined in example 5.10, is clearly a non-deterministic PDA. Now, let us see the difference between *deterministic* PDA and *non-deterministic* PDA.

Definition: Let M = (Q, Σ, Γ, δ, q_0, Z_0, F) be a PDA. The PDA is deterministic if

1. δ(q, a, Z) has only one element.
2. If δ(q, ε, Z) is not empty, then δ(q, a, Z) should be empty.

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

In the PDAs discussed in the examples 5.3 to 5.10, let us see what are deterministic PDAs and what are non-deterministic PDAs.

Example 5.11: **Is the PDA to accept the language L(M) = {wCw^R | w∈ (a + b)*} discussed in example 5.4 is deterministic?**

The transitions defined for this machine are obtained in example 5.4 and are reproduced for the sake of convenience.

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

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

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

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

Example 5.12: **Is the PDA corresponding to the language $L = \{a^n b^n \mid n \geq 1\}$ by a final state is deterministic?**

The transitions defined for this machine discussed in the example 5.5 are reproduced here for the sake of convenience.

$$\begin{aligned}
\delta(q_0, a, Z_0) &= (q_0, aZ_0) \\
\delta(q_0, a, a) &= (q_0, aa) \\
\delta(q_0, b, a) &= (q_1, \varepsilon) \\
\delta(q_1, b, a) &= (q_1, \varepsilon) \\
\delta(q_1, \varepsilon, Z_0) &= (q_2, \varepsilon)
\end{aligned}$$

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

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

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

Example 5.13: **Is the PDA to accept the language $L(M) = \{w \mid w \in (a+b)^*$ and $n_a(w) = n_b(w)$ is deterministic?**

The transitions defined for this machine discussed in the example 5.6 are reproduced here for the sake of convenience.

$$\begin{aligned}
\delta(q_0, a, Z_0) &= (q_0, aZ_0) \\
\delta(q_0, b, Z_0) &= (q_0, bZ_0) \\
\delta(q_0, a, a) &= (q_0, aa)
\end{aligned}$$

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

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

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

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

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

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

Example 5.14: **Is the PDA to accept the language consisting of balanced parentheses is deterministic?**

The transitions defined for this machine discussed in the example 5.7 are reproduced here for the sake of convenience.

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

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

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

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

Example 5.15: **Is the PDA to accept the language $L = \{w \mid w \in (a, b)^*$ and $n_a(w) > n_b(w)\}$ is deterministic?**

The transitions defined for this machine discussed in the example 5.8 are reproduced here for the sake of convenience.

$$\begin{aligned}
\delta(q_0, a, Z_0) &= (q_0, aZ_0)\\
\delta(q_0, b, Z_0) &= (q_0, bZ_0)\\
\delta(q_0, a, a) &= (q_0, aa)\\
\delta(q_0, b, b) &= (q_0, bb)\\
\delta(q_0, a, b) &= (q_0, \varepsilon)\\
\delta(q_0, b, a) &= (q_0, \varepsilon)\\
\delta(q_0, \varepsilon, a) &= (q_1, a)
\end{aligned}$$

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

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

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

$$\begin{aligned}
\delta(q_0, a, a) &= (q_0, aa)\\
\delta(q_0, b, a) &= (q_0, \varepsilon)
\end{aligned}$$

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

Example 5.16: **Is the PDA to accept the language $L = \{a^n b^{2n} \mid n \geq 1\}$ is deterministic?**

The transitions defined for this machine discussed in the example 5.9 are reproduced here for the sake of convenience.

$$\begin{aligned}
\delta(q_0, a, Z_0) &= (q_0, aaZ_0) \\
\delta(q_0, a, a) &= (q_0, aaa) \\
\delta(q_0, b, a) &= (q_1, \varepsilon) \\
\delta(q_1, b, a) &= (q_1, \varepsilon) \\
\delta(q_1, \varepsilon, Z_0) &= (q_2, \varepsilon)
\end{aligned}$$

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

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

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

Example 5.17: **Is the PDA to accept the language $L = \{ww^R \mid w \in (a + b)^*\}$ is deterministic?**

The transitions defined for this machine discussed in the example 5.10 are reproduced here for the sake of convenience.

$$\begin{aligned}
\delta(q_0, a, Z_0) &= (q_0, aZ_0) \\
\delta(q_0, b, Z_0) &= (q_0, bZ_0) \\
\delta(q_0, a, a) &= (q_0, aa), (q_1, \varepsilon) \\
\delta(q_0, b, a) &= (q_0, ba) \\
\delta(q_0, a, b) &= (q_0, ab) \\
\delta(q_0, b, b) &= (q_0, bb), (q_1, \varepsilon) \\
\delta(q_1, a, a) &= (q_1, \varepsilon) \\
\delta(q_1, b, b) &= (q_1, \varepsilon) \\
\delta(q_1, \varepsilon, Z_0) &= (q_2, Z_0)
\end{aligned}$$

The first condition to be deterministic is $\delta(q, a, Z)$ should have only one component. But, there are two transitions each having two components viz.,

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

So, the first condition fails. To be deterministic, both the conditions shown in the definition should be satisfied. Since one of the conditions is not met, the PDA is *non-deterministic*.

Exercises:

1. What are the demerits of regular languages when compared to context free languages?
2. What are the demerits of DFA (or NFA) when compared with PDA?
3. Why FAs are less powerful than the PDAs?
4. What is the difference between NFA and PDA?
5. What is a PDA? Explain with an example.
6. What does each of the following transitions represent?
   a. $\delta(p, a, Z) = (q, aZ)$
   b. $\delta(p, a, Z) = (q, \varepsilon)$
   c. $\delta(p, a, Z) = (q, r)$
   d. $\delta(p, \varepsilon, Z) = (q, r)$
   e. $\delta(p, \varepsilon, \varepsilon) = (q, Z)$
   f. $\delta(p, \varepsilon, Z) = (q, \varepsilon)$
7. How the transition / move of a PDA defined?
8. What is an instantaneous description? Explain with respect to PDA.
9. When a language is accepted by a final state and when a language is accepted by an empty stack?

10. Obtain a PDA to accept the language $L(M) = \{wCw^R \mid w \in (a + b)^*\}$ where W^R is reveres of W. Show the sequence of moves made by the PDA for the strings aabCbaa, aabCbab.
11. Obtain a PDA to accept the language $L = \{a^n b^n \mid n \geq 1\}$ by a final state.
12. Obtain a PDA to accept the language $L(M) = \{w \mid w \in (a+b)^*$ and $n_a(w) = n_b(w)$ i.e., number of a's in string w should be equal to number of b's in w.
13. Obtain a PDA to accept a string of balanced parentheses. The parentheses to be considered are (,), [,], { and }.
14. Obtain a PDA to accept the language $L = \{w \mid w \in (a, b)^*$ and $n_a(w) > n_b(w)\}$
15. Obtain a PDA to accept the language $L = \{a^n b^{2n} \mid n \geq 1\}$
16. Obtain a PDA to accept the language $L = \{ww^R \mid w \in (a + b)^*\}$
17. When the PDA is deterministic and when it is called non-deterministic?
18. Is the PDA to accept the language $L(M) = \{wCw^R \mid w \in (a + b)^*\}$ is deterministic?
19. Is the PDA corresponding to the language $L = \{a^n b^n \mid n \geq 1\}$ by a final state is deterministic?
20. Is the PDA to accept the language $L(M) = \{w \mid w \in (a+b)^*$ and $n_a(w) = n_b(w)$ is deterministic?
21. Is the PDA to accept the language consisting of balanced parentheses is deterministic?
22. Is the PDA to accept the language $L = \{w \mid w \in (a, b)^*$ and $n_a(w) > n_b(w)\}$ is deterministic?
23. Is the PDA to accept the language $L = \{a^n b^{2n} \mid n \geq 1\}$ is deterministic?
24. Is the PDA to accept the language $L = \{ww^R \mid w \in (a + b)^*\}$ is deterministic?

6.2.3 From Empty Stack to Final State

We shall show that the classes of languages that are $L(P)$ for some PDA P is the same as the class of languages that are $N(P)$ for some PDA P. This class is also exactly the context-free languages, as we shall see in Section 6.3. Our first construction shows how to take a PDA P_N that accepts a language L by empty stack and construct a PDA P_F that accepts L by final state.

Theorem 6.9: If $L = N(P_N)$ for some PDA $P_N = (Q, \Sigma, \Gamma, \delta_N, q_0, Z_0)$, then there is a PDA P_F such that $L = L(P_F)$.

PROOF: The idea behind the proof is in Fig. 6.4. We use a new symbol X_0, which must not be a symbol of Γ; X_0 is both the start symbol of P_F and a marker on the bottom of the stack that lets us know when P_N has reached an empty stack. That is, if P_F sees X_0 on top of its stack, then it knows that P_N would empty its stack on the same input.

We also need a new start state p_0, whose sole function is to push Z_0, the start symbol of P_N, onto the top of the stack and enter state q_0, the start state of P_N. Then, P_F simulates P_N, until the stack of P_N is empty, which P_F detects because

[2] The N in $N(P)$ stands for "null stack," a synonym for "empty stack."

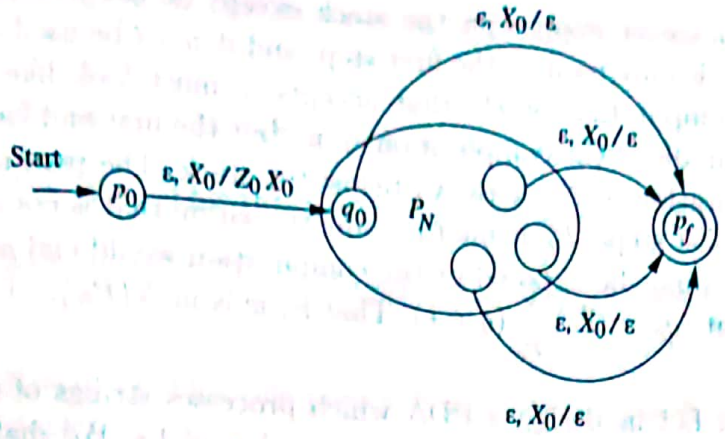


Figure 6.4: P_F simulates P_N and accepts if P_N empties its stack

it sees X_0 on the top of the stack. Finally, we need another new state, p_f, which is the accepting state of P_F; this PDA transfers to state p_f whenever it discovers that P_N would have emptied its stack.

The specification of P_F is as follows:

$$P_F = (Q \cup \{p_0, p_f\}, \Sigma, \Gamma \cup \{X_0\}, \delta_F, p_0, X_0, \{p_f\})$$

where δ_F is defined by:

1. $\delta_F(p_0, \epsilon, X_0) = \{(q_0, Z_0X_0)\}$. In its start state, P_F makes a spontaneous transition to the start state of P_N, pushing its start symbol Z_0 onto the stack.

2. For all states q in Q, inputs a in Σ or $a = \epsilon$, and stack symbols Y in Γ, $\delta_F(q, a, Y)$ contains all the pairs in $\delta_N(q, a, Y)$.

3. In addition to rule (2), $\delta_F(q, \epsilon, X_0)$ contains (p_f, ϵ) for every state q in Q.

We must show that w is in $L(P_F)$ if and only if w is in $N(P_N)$.

(If) We are given that $(q_0, w, Z_0) \overset{*}{\underset{P_N}{\vdash}} (q, \epsilon, \epsilon)$ for some state q. Theorem 6.5 lets us insert X_0 at the bottom of the stack and conclude $(q_0, w, Z_0X_0) \overset{*}{\underset{P_N}{\vdash}} (q, \epsilon, X_0)$. Since by rule (2) above, P_F has all the moves of P_N, we may also conclude that $(q_0, w, Z_0X_0) \overset{*}{\underset{P_F}{\vdash}} (q, \epsilon, X_0)$. If we put this sequence of moves together with the initial and final moves from rules (1) and (3) above, we get:

$$(p_0, w, X_0) \underset{P_F}{\vdash} (q_0, w, Z_0X_0) \overset{*}{\underset{P_F}{\vdash}} (q, \epsilon, X_0) \underset{P_F}{\vdash} (p_f, \epsilon, \epsilon) \tag{6.1}$$

Thus, P_F accepts w by final state.

(Only-if) The converse requires only that we observe the additional transitions of rules (1) and (3) give us very limited ways to accept w by final state. We must use rule (3) at the last step, and we can only use that rule if the stack of P_F contains

only X_0. No X_0's ever appear on the stack except at the bottommost position. Further, rule (1) is only used at the first step, and it *must* be used at the first step.

Thus, any computation of P_F that accepts w must look like sequence (6.1). Moreover, the middle of the computation — all but the first and last steps — must also be a computation of P_N with X_0 below the stack. The reason is that, except for the first and last steps, P_F cannot use any transition that is not also a transition of P_N, and X_0 cannot be exposed or the computation would end at the next step. We conclude that $(q_0, w, Z_0) \overset{*}{\underset{P_N}{\vdash}} (q, \epsilon, \epsilon)$. That is, w is in $N(P_N)$. □

6.2.4 From Final State to Empty Stack

Now, let us go in the opposite direction: take a PDA P_F that accepts a language L by final state and construct another PDA P_N that accepts L by empty stack. The construction is simple and is suggested in Fig. 6.7. From each accepting state of P_F, add a transition on ϵ to a new state p. When in state p, P_N pops its stack and does not consume any input. Thus, whenever P_F enters an accepting state after consuming input w, P_N will empty its stack after consuming w.

To avoid simulating a situation where P_F accidentally empties its stack without accepting, P_N must also use a marker X_0 on the bottom of its stack. The marker is P_N's start symbol, and like the construction of Theorem 6.9, P_N must start in a new state p_0, whose sole function is to push the start symbol of P_F on the stack and go to the start state of P_F. The construction is sketched in Fig. 6.7, and we give it formally in the next theorem.

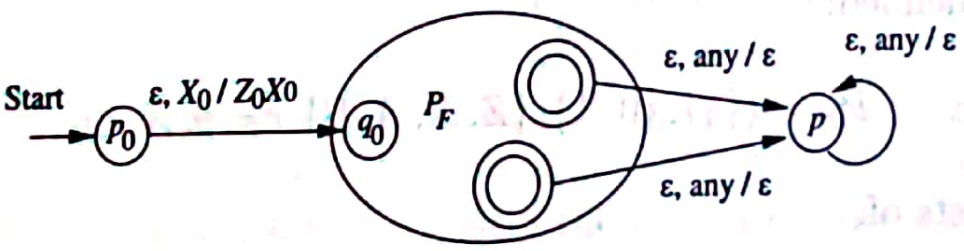


Figure 6.7: P_N simulates P_F and empties its stack when and only when P_N enters an accepting state

Theorem 6.11: Let L be $L(P_F)$ for some PDA $P_F = (Q, \Sigma, \Gamma, \delta_F, q_0, Z_0, F)$. Then there is a PDA P_N such that $L = N(P_N)$.

PROOF: The construction is as suggested in Fig. 6.7. Let

$$P_N = (Q \cup \{p_0, p\}, \Sigma, \Gamma \cup \{X_0\}, \delta_N, p_0, X_0)$$

where δ_N is defined by:

1. $\delta_N(p_0, \epsilon, X_0) = \{(q_0, Z_0X_0)\}$. We start by pushing the start symbol of P_F onto the stack and going to the start state of P_F.
2. For all states q in Q, input symbols a in Σ or $a = \epsilon$, and Y in Γ, $\delta_N(q, a, Y)$ contains every pair that is in $\delta_F(q, a, Y)$. That is, P_N simulates P_F.

3. For all accepting states q in F and stack symbols Y in Γ or $Y = X_0$, $\delta_N(q, \epsilon, Y)$ contains (p, ϵ). By this rule, whenever P_F accepts, P_N can start emptying its stack without consuming any more input.

4. For all stack symbols Y in Γ or $Y = X_0$, $\delta_N(p, \epsilon, Y) = \{(p, \epsilon)\}$. Once in state p, which only occurs when P_F has accepted, P_N pops every symbol on its stack, until the stack is empty. No further input is consumed.

Now, we must prove that w is in $N(P_N)$ if and only if w is in $L(P_F)$. The ideas are similar to the proof for Theorem 6.9. The "if" part is a direct simulation, and the "only-if" part requires that we examine the limited number of things that the constructed PDA P_N can do.

(If) Suppose $(q_0, w, Z_0) \overset{*}{\underset{P_F}{\vdash}} (q, \epsilon, \alpha)$ for some accepting state q and stack string α. Using the fact that every transition of P_F is a move of P_N, and invoking Theorem 6.5 to allow us to keep X_0 below the symbols of Γ on the stack, we know that $(q_0, w, Z_0X_0) \overset{*}{\underset{P_N}{\vdash}} (q, \epsilon, \alpha X_0)$. Then P_N can do the following:

$$(p_0, w, X_0) \underset{P_N}{\vdash} (q_0, w, Z_0X_0) \overset{*}{\underset{P_N}{\vdash}} (q, \epsilon, \alpha X_0) \overset{*}{\underset{P_N}{\vdash}} (p, \epsilon, \epsilon)$$

The first move is by rule (1) of the construction of P_N, while the last sequence of moves is by rules (3) and (4). Thus, w is accepted by P_N, by empty stack.

(Only-if) The only way P_N can empty its stack is by entering state p, since X_0 is sitting at the bottom of stack and X_0 is not a symbol on which P_F has any moves. The only way P_N can enter state p is if the simulated P_F enters an accepting state. The first move of P_N is surely the move given in rule (1). Thus, every accepting computation of P_N looks like

$$(p_0, w, X_0) \underset{P_N}{\vdash} (q_0, w, Z_0X_0) \overset{*}{\underset{P_N}{\vdash}} (q, \epsilon, \alpha X_0) \overset{*}{\underset{P_N}{\vdash}} (p, \epsilon, \epsilon)$$

where q is an accepting state of P_F.

Moreover, between ID's (q_0, w, Z_0X_0) and $(q, \epsilon, \alpha X_0)$, all the moves are moves of P_F. In particular, X_0 was never the top stack symbol prior to reaching ID $(q, \epsilon, \alpha X_0)$.[3] Thus, we conclude that the same computation can occur in P_F, without the X_0 on the stack; that is, $(q_0, w, Z_0) \overset{*}{\underset{P_F}{\vdash}} (q, \epsilon, \alpha)$. Now we see that P_F accepts w by final state, so w is in $L(P_F)$. □