7 Pushdown Automata

Introduction

Pushdown automata (in short *PDA*) is the machine format of the context-free language. It is the same as finite automata with the attachment of an auxiliary amount of storage as stack. Already, we have discussed the limitations of finite automata. Pushdown automata is designed to remove those limitations. Remember the example of $a^n b^n$, which is not recognizable by the finite automata due to the limitation of memory, but pushdown automata can memorize the number of occurrences of 'a' and match it with the number of occurrences of 'b' with the help of the stack. In this chapter, we shall learn about the mathematical representation of *PDA*, acceptance by a *PDA*, deterministic *PDA*, non-deterministic *PDA*, conversion of *CFG* to *PDA* and *PDA* to *CFG*, and the graphical representation of *PDA*.

7.1 Basics of PDA

7.1.1 Definition

A PDA consists of a 7-tuple

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

where

Q denotes a finite set of states.

Σ denotes a finite set of input symbols.

Γ denotes a finite set of pushdown symbols or stack symbols.

δ denotes the transitional functions.

q_0 is the initial state of the PDA ($q_0 \in Q$).

z_0 is the stack bottom symbol.

F is the fi nal state of the PDA.

In PDA, the transitional function δ is in the form

$$Q \times (\Sigma \cup \lambda) \times \Gamma \rightarrow (Q, \Gamma)$$

(The PDA is in some state. With an input symbol from the input tape and with the stack top symbol, the PDA moves to one right. It may change its state and push some symbol in the stack or pop symbol from the stack top.)

7.1.2 Mechanical Diagram of the PDA (Fig. 7.1)

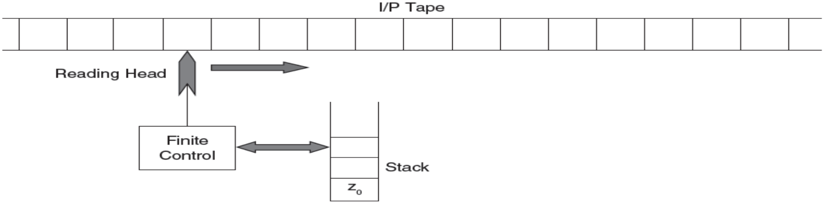

Fig. 7.1

Mechanical diagram of PDA is given in Fig. 7.1. The components of it are described below.

- **Input tape:** The input tape contains the input symbols. The tape is divided into a number of squares. Each square contains a single input character. The string placed in the input tape is traversed from left to right. The two end sides of the input string contain an infinite number of blank symbols.
- **Reading head:** The head scans each square in the input tape and reads the input from the tape. The head moves from left to right. The input scanned by the reading head is sent to the finite control of the PDA.
- **Finite control:** The finite control can be considered as a control unit of a PDA. An automaton always resides in a state. The reading head scans the input from the input tape and sends it to the finite control. A two-way head is also added with the finite control to the stack top. Depending on the input taken from the input tape and the input from the stack top, the finite control decides in which state the PDA will move and which stack symbol it will push to the stack or pop from the stack or do nothing on the stack.

- **Stack:** A stack is a temporary storage of stack symbols. Every move of the PDA indicates one of the following to the stack

– One stack symbol may be added to the stack (push)
– One stack symbol may be deleted from the top of the stack (pop)

The stack is the component of the PDA which differentiates it from the fi nite automata. In the stack, there is always a symbol z0 which denotes the bottom of the stack.

7.1.2.1 Why Pushdown?

Push is an operation related to the stack. By this operation, one symbol is added to the stack top.

In finite automata, the states act as a form of primitive memory. The states memorize the nonterminals encountered at the time of derivation of a string. So, only the state is suitable for traversing a regular expression as in the case of finite automata. Now, let us consider a case of $L = \{a^n b^n, wheren \geq 1\}$.It is not a regular expression but a context-free language. Here, n is any number. In the string, there is an equal number of 'a' and 'b'. In the string, all 'a's appear before 'b'. So, the machine for traversing $a^n b^n$ has to remember n number of a's to traverse an equal number of 'b'. n is any number. Thus, to memorize the number of a's in the string, the machine requires an infinite number of states, i.e., the machine will not be a finite state machine.

To remove this bottleneck, we need to add an auxiliary memory in the form of a stack. For each occurrence of 'a' in the string L, one symbol is pushed into the stack. For each occurrence of b (after finishing 'a'), one symbol will be popped from the stack. By this process, the matching of 'a' and 'b' will be done.

By adding a stack to the finite automata, the PDA is generated.

7.1.2.2 Instantaneous Description (ID)

It describes the configuration of the PDA at a given instance. ID remembers the information of the state and the stack content at a given instance of time.

An ID is a format of triple (q, w, k), where $q \in Q$ (finite set of states), $w \in \Sigma$ (finite set of input alphabets), and $k \in \Gamma$ (fi nite set of stack symbols).

7.2 Acceptance by a PDA

There are two ways to declare a string to be accepted by a PDA:

1. Accepted by an empty stack (store)
2. Accepted by the final state

7.2.1 Accepted by an Empty Stack (Store)

We know in each PDA there is a stack attached. In the stack at the bottom, there is a symbol called the stack bottom symbol, say z_0. In each move of the PDA, one symbol called the stack symbol is either pushed in or popped from the stack. But the symbol z_0 still remains in the stack.

A string w may be declared accepted by an empty stack after processing all the symbols of w, if the stack is empty after reading the rightmost input character of the string w. In mathematical notation, we can say that if $M = Q, \Sigma, \Gamma, \delta, q_0, z_0, F$ is a PDA, the string w is declared accepted by empty stack if

$$\{w \in \Sigma^*/(q_0, w, z_0) \rightarrow^* (q, \lambda, \lambda) for some q \in Q\}$$

$\{w \in \Sigma^*/(q_0, w, z_0) \rightarrow^* (q, \lambda, \lambda) for some q \in Q\}$

In general, we can say that a string is accepted by a PDA by empty stack if both the following conditions are fulfilled:

1. The string is finished (totally traversed)
2. The stack is empty

7.2.2 Accepted by the Final State A string w may be declared

accepted by the final state if after total traversal of the string w the PDA enters into its final state. In mathematical notation, we can say that if $M = \{Q, \Sigma, \Gamma, \delta, q_0, z_0, F\}$ is a PDA, the string w is declared accepted by the final state if

$$\{w \in \Sigma^*/(q_0, w, z_0) \rightarrow^* (q_f, \lambda, z_0) for q_f \in F and z_0 is the stack bottom symbol\}$$

In general, we can say that a string is accepted by a PDA by the final state if both the following conditions are fulfilled:

1. The string is finished (totally traversed)
2. The PDA enters into its final state

[Although there are two ways to declare a string to be accepted by a PDA, it can be proved that both the ways are equivalent. In general, for both the cases all the steps are the same, except the last step. In the last step, it is differentiated whether the string is accepted by the empty stack or by the final state.

In general, we can say that a string is accepted by a PDA by the final state if both the following conditions are fulfilled:

1. The string is finished (totally traversed)
2. The PDA enters into its final state

[Although there are two ways to declare a string to be accepted by a PDA, it can be proved that both the ways are equivalent. In general, for both the cases all the steps are the same, except the last step. In the last step, it is differentiated whether the string is accepted by the empty stack or by the final state.

if in the last step $\delta(q_i, \lambda, z_0) \rightarrow (q_i, \lambda)$, then it is accepted by the empty stack

if $\delta(q_i, \lambda, z_0) \rightarrow (q_f, z_0)$, then it is accepted by the fi nal state.]

Theorem 7.1: A language is accepted by a PDA by the empty stack if and only if the language is accepted by a PDA by the final state.

Proof: Let $M_1 = \{Q, \Sigma, \Gamma, \delta, q_0, z_0, F\}$ be a PDA. Let it accept a language L by the final state. Let there exist another PDA M_2, which accepts the same language L by the empty stack.

Theorem 7.1: A language is accepted by a PDA by the empty stack if and only if the language is accepted by a PDA by the final state.

Proof: Let $M_1 = \{Q, \Sigma, \Gamma, \delta, q_0, z_0, F\}$ be a PDA. Let it accept a language L by the final state. Let there exist another PDA M_2, which accepts the same language L by the empty stack.

Let $M_2 = \{Q', \Sigma', \Gamma', \delta', q_0', z_0', F'\}$,

where $Q' = Q \cup \{q_0', q_e\}, where q_0', q_e \in Q$

$\Sigma' = \Sigma$

$\Gamma' = \Gamma \cup \{z_0'\}, where z_0' \in \Gamma$

The transitional function δ' is defined as follows:

a) $\delta'(q_0', \lambda, z_0') contains (q_0, z_0 z_0')$
b) $\delta'(q_i, a, z)$ is the same as $\delta(qi, a, z)$ for all $q_i \in Q$, $a \in \Sigma \cup \{\lambda\}$, and $z \in \Gamma$
c) $\delta'(q_i, \lambda, z)$ contains (q_e, λ) for $q_i \in F$ and for all $z \in \Gamma \cup \{z_0'\}$
d) $\delta'(q_e, \lambda, z)$ contains (q_e, λ) for all $z \in \Gamma \cup \{z_0'\}$

[Here (a) describes that M2 enters in the initial configuration of M_1 with z_0' as the leftmost push down symbol. (b) describes to make M_2 simulate the behaviour of M_1, thus reading an input word w if M_1 reads w and enters a final state. Once w is read and M_2 enters a final state of M_1, the effect of (c) and (d) are to empty the pushdown store.]

Here, $w \in M_1$ if and only if

The transitional function δ' is defined as follows:

a) $\delta'(q_0', \lambda, z_0') contains (q_0, z_0 z_0')$
b) $\delta'(q_i, a, z)$ is the same as $\delta(qi, a, z)$ for all $q_i \in Q$, $a \in \Sigma \cup \{\lambda\}$, and $z \in \Gamma$
c) $\delta'(q_i, \lambda, z)$ contains (q_e, λ) for $q_i \in F$ and for all $z \in \Gamma \cup \{z_0'\}$
d) $\delta'(q_e, \lambda, z)$ contains (q_e, λ) for all $z \in \Gamma \cup \{z_0'\}$

[Here (a) describes that M2 enters in the initial configuration of M_1 with z_0' as the leftmost push down symbol. (b) describes to make M_2 simulate the behaviour of M_1, thus reading an input word w if M_1 reads w and enters a final state. Once w is read and M_2 enters a final state of M_1, the effect of (c) and (d) are to empty the pushdown store.]

Here, $w \in M_1$ if and only if

$$(q_0, w, z_0) \Rightarrow^* (q_f, \lambda, z_0)$$

where q_f is the final state of the machine M_1.

By rule (a), we can write

$$\begin{aligned} (q_0', w, z_0') &\Rightarrow (q_0, w, z_0' z_0) \\ &\Rightarrow^* (q_f, \lambda, z_0' z_0) -- \text{ by rule (b) where } q_f \text{ is the final state} \\ &\Rightarrow (q_e, \lambda, \lambda) -- \text{ by rules (c) and (d)} \end{aligned}$$

Conversely, let $M_2 = (Q', \Sigma', \Gamma', \delta', q_0', z_0', \phi)$ be a PDA accepting L by the empty store.Then,

$$M_1 = \} Q, \Sigma, \Gamma, \delta, q_0, z_0, F \}$$