11 Pushdown Automata

'Pushdown lethargy in a context-free environment, then is available an unlimited amount of resources—'

Introduction

The applications of regular languages are very important in the field of computer science. However, many useful regular languages which are not regular cannot be accepted by DFAs and NFAs. For example, the language consisting of nested, balanced parentheses is not regular and hence is not accepted by a DFA, where

$$L = \{\epsilon, (), ()(), (()), ((())), \cdots\}.$$

In fact, this language is useful in programming languages for the purpose of nesting expressions and program blocks–the reason for the non-acceptance of this language by a DFA being that the storage of information will be done in its current state only. Hence, for sufficiently long inputs, the machine loses track of the pattern of the parentheses suggesting that there exists a limit for the memory of DFAs.

A pushdown automaton *PDA* is similar to FSA, except that *PDA* has an **auxiliary stack** which provides an unlimited amount of memory. A language L is recognised by a pushdown automaton, iff L is **context-free**.

Thus, pushdown automaton accepts a rich class of languages (which may be regular or non-regular) with an unlimited memory capacity, in the form of stack. PDA contributes mainly in the area of parsing and compiler construction.

Definition:

There exist the following equivalent definitions for the concept of PDA:

a. PDA is a way to represent the language class called context-free languages. In other words, PDAs are abstract devices that recognise context-free languages.
b. PDA is a generalisation of FSA and a PDA changes from state to state, reading input symbols. Unlike FSA, transitions also update the stack either by popping symbols or by pushing them.

11.1 Components of PDA

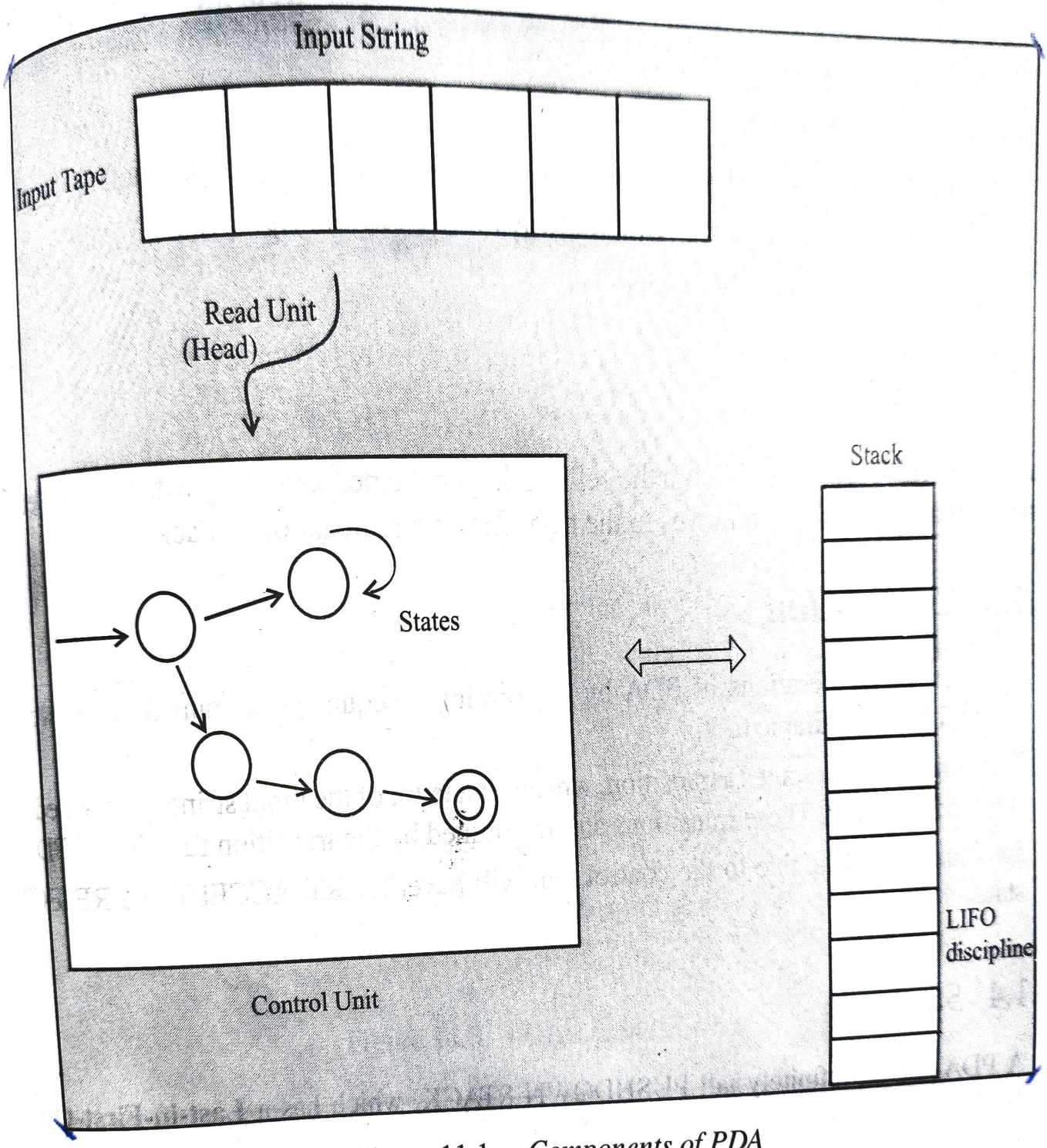


Figure 11.1. *Components of PDA*

The PDA is usually described as consisting of four components:

- Input Tape
- Read unit
- Control unit
- Stack (Memory unit)

11.1.1 *Input Tape*

- It is an infinitely long tape, on which input is written.
- The tape is divided into sequence of cells. Each cell begins from the left end and extends to the right, without an end.
- Each cell of the tape holds one input letter or a blank, $\in$.
- Input string is written on to the tape, prior to the beginning of the operation of the PDA.
- The tape has read-only head, which always moves to the right, one cell at a time. It reads a symbol and cannot go back.

11.1.2 *Read Unit*

Read unit of PDA reads words from the cells of the input tape, beginning with the first letter in the leftmost cell, and then moves to the right. However, it cannot go back.

11.1.3 *Control Unit*

- It governs the operations of PDA by performing a sequence of transitions between internal states available to it.
- The control unit executes a transition, whenever a letter of the input string is provided to it by the read unit. These transitions are determined by the transition function of PDA.
- Internal states available to the control unit will have START, ACCEPT and REJECT states.

11.1.4 *Stack*

- A PDA has an infinitely tall PUSHDOWN STACK, which has a **Last-in-First-Out (LIFO)** discipline.
- Stack always start with STACK empty.
- Stack can hold letters of STACK alphabet which can be the same as input alphabets.
- Usually, an initial stack symbol is placed on the top of the stack.
- The primitives, that are used to write to the stack, are:
  a. **Push** – Adds the input alphabet to the top of the STACK.
  b. **Pop** – Removes the top input alphabet from the top of the stack. If the stack is empty, then a basic pop does not change the state of the stack.
  c. **nop** – does nothing to the stack.

11.2 Description of PDA

There are different ways to describe the task of PDA, as follows:

11.2.1 *Transition Diagram*

In a directed graph, for an arc going from the vertex which corresponds to state p, to the vertex that corresponds to state q, the edge labelling can be represented in different forms as follows:

form-1: (input symbol, top input symbol of stack) / Operations on stack

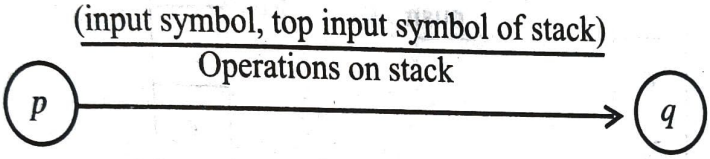


Figure 11.2. *Edge Label Format*

form-2: (p, input symbol, top symbol fo stack, operation on stack, q)

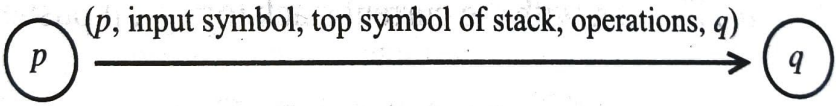


Figure 11.3. *Edge Label Format*

form-3: input symbol, Pop old stack symbol | Push new stack symbol

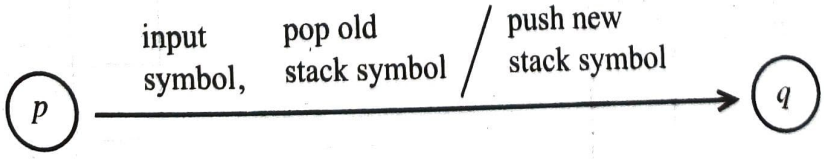


Figure 11.4. *Edge Label Format*

The transition diagram of PDA incorporates the following operations: If a PDA is in state p, the input symbol is any letter of a string w (the input symbol can be $\in$ also) and 's' is the top element of stack, then PDA

a. executes the stack operation (push|pop|nop)
b. moves to state q, and
c. if the input symbol $\neq \in$, then it goes to the right (i.e. the next cell of the tape).

EXAMPLE 11.2.1: (Edge labelling in transition diagram)

a.

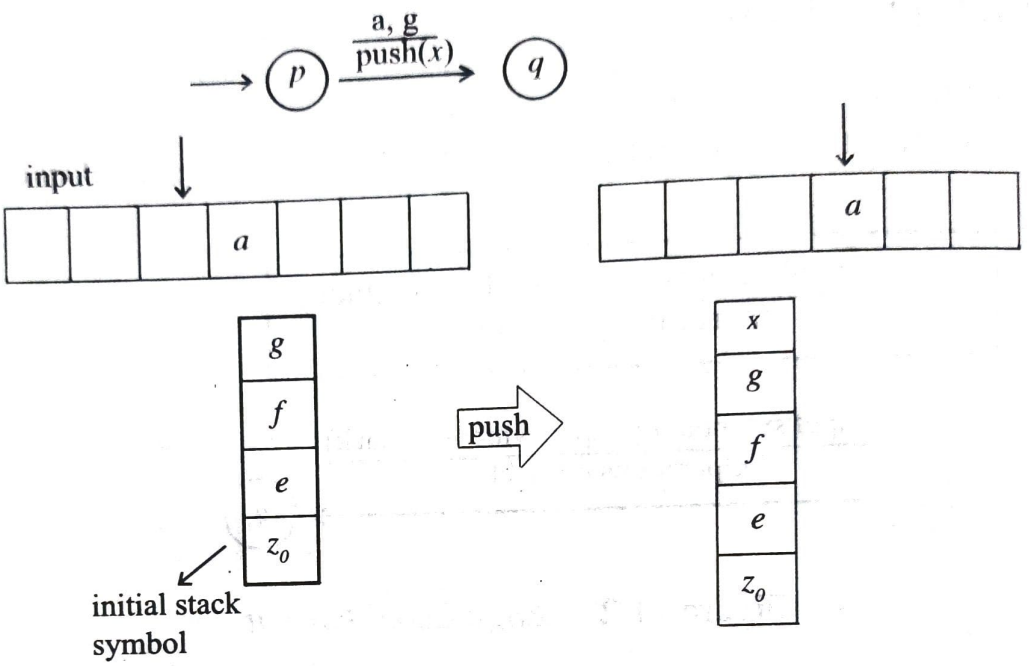


Figure 11.5. *Transition Diagram Showing Edge Lable Format-1*

$\left(\frac{a,g}{push(x)}\right)$ means, with the current state as p, the control goes to the state q by reading the input symbol 'a'. Further, with the current stack top 'g', it pushes x onto the stack.

b.

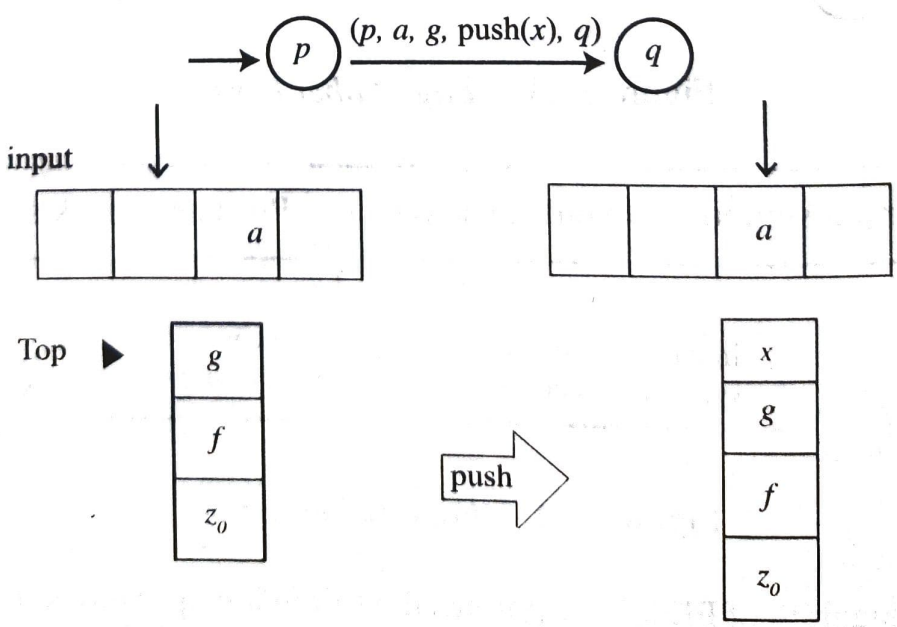


Figure 11.6. *Transition Diagram Showing Edge Label Format-2*

c.

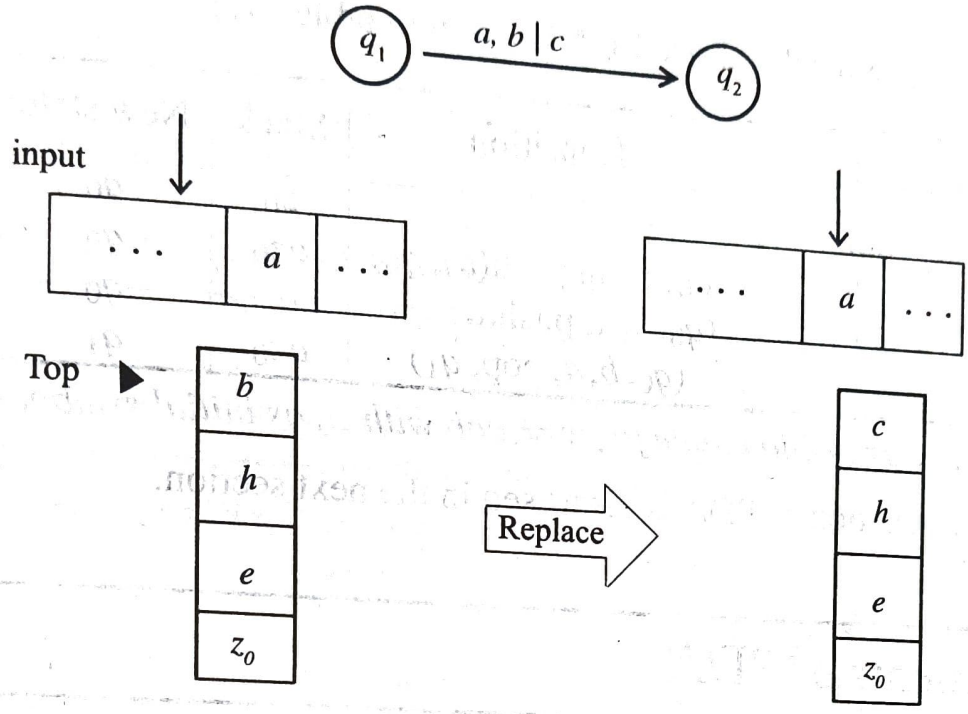


Figure 11.7. *Transition Diagram Showing Edge Label Format-3*

Here the PDA goes from q_1 to q_2 by reading an input symbol 'a', pops the top symbol from stack '(b)' and finally, pushes the new symbol 'c' onto stack.

11.2.2 *Transition Table*

The description of operation of a PDA, for a given input string, can be represented in a tabular format called transition table. The table format is shown below.

Unread input	Transition	Stack	New state

Table 11.1 *Transition table format*

EXAMPLE 11.2.2: Consider a PDA, whose task is described in the transition diagram shown in figure 11.8:

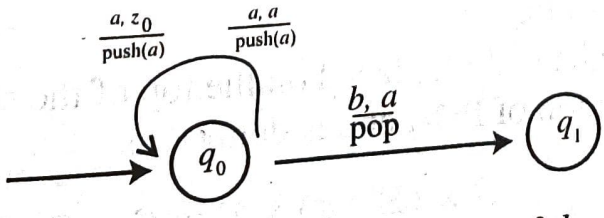


Figure 11.8. *Transition Diagram of the PDA*

The transition table for the above PDA is given in table 11.2.

Unread input	Transition	Stack	New state
aab	—	z_0	q_0
ab	$(q_0, a, z_0, \text{push}(a), q_0)$	az_0	q_0
b	$(q_0, a, a, \text{push}(a), q_0)$	aaz_0	q_0
$\in$	$(q_0, b, a, \text{pop}, q_1)$	az_0	q_1

Table 11.2 *Transition table for $w = aab$ with z_0 as initial symbol on stack*

The transition function of PDA is discussed in the next section.

11.3 Elements of PDA

A PDA constitutes the following seven characteristics:

a. A finite set Q, of states $q_0, q_1, \ldots, q_n$.
b. A finite input alphabet of letters $\Sigma = \{a, b, \cdots\}$, for forming the input string.
c. A finite set 'Γ' of stack symbols.
d. A transition function 'S', which tells how PDA goes from one step to the next.
e. An initial state q_0.
f. A symbol z_0, indicating the top of the stack.
g. A set of final states F.

11.3.1 *Ordered Seven-Tuple Specification of PDA*

Formally, a PDA is a seven-tuple.

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

where,

a. Q is the set of finite states of PDA.
b. Σ is a finite set of input symbols.
c. Γ is the finite set of stack symbols.
d. $q_0 \in Q$ is the initial state.
e. $F \sqsubseteq Q$ is the final state.
f. z_0 is the initial stack symbol, placed on the top of the stack.
g. δ is transition function of PDA and is defined as

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

11.3.2 *Transitions of PDA*

The transition of PDA can be represented in different ways, as follows:

Form-1:

δ(current state, current input symbol, current stack top) = (new state, new stack top).

Form-2:

(current state, current input symbol, current stack top, operation on stack, new state)

EXAMPLE 11.3.1: (Transition format)

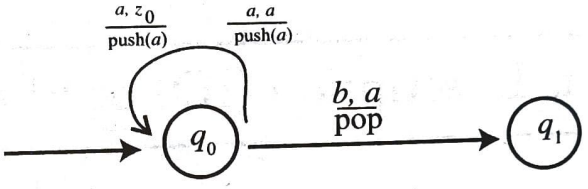


Figure 11.9. *Transition Diagram*

Transitions of the above diagram are represented using different forms as:

Form-1: $(q_0, a, z_0\ \text{push}(a), q_0)$
This means for the current state q_0, current input symbol a, if the current stack top is z_0, then push 'a' onto stack and remain in state q_0.

Form-2: $\delta(q_0, a, z_0) = (q_0, a)$
This means for the current state q_0, current input symbol 'a', if the current stack top is z_0, then new state and new stack top are represented as (q_0, a).

Similarly, other transitions of above diagram are:

Form-1: $(q_0, a, a\ \text{push}(a), q_0)$
Form-2: $\delta(q_0, a, a) = (q_0, a)$
Form-1: $(q_0, b, a, \text{pop}, q_1)$
Form-2: $\delta(q_0, b, a) = (q_1, e)$

where 'e' is used to indicate pop 'a' (current stack top) from the stack.

11.4 The Language Accepted by PDA

There are two ways to describe the acceptance of a language:

a. Define the language accepted to be the set of all inputs, for which some sequence of moves causes PDA to empty its stack. This language is referred to as **language accepted by empty stack.**

b. Designate some state as final state and define the accepted language as the set of all inputs, for which choice of moves causes PDA **to enter a final state**.

In other words a language is accepted by PDA, if the machine attains the form

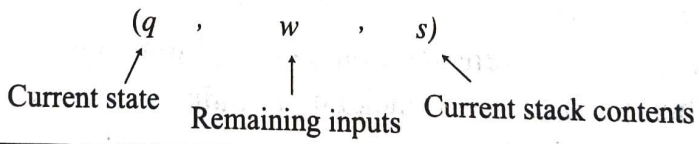
'(final state, end of input, empty stack)'.

Formally, let $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ be a PDA. The language accepted by M is

$$L(M) = \{w \in \Sigma^* | (q_0, w, Z) \vdash^* M(P, \in, K), P \in F \text{ and } K \in \Gamma\}.$$

11.5 Instantaneous Description (ID) of PDA

ID of a PDA is defined as a triple (q, w, s), where q is the current state, w is the remaining input and s is the current stack contents.



Formally, let $M = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ be a PDA, then the ID of a PDA is:

$$(q, aw, z\alpha) \vdash_m (p, w, B\alpha).$$

EXAMPLE 11.5.1: Consider the following transition diagram of a PDA.

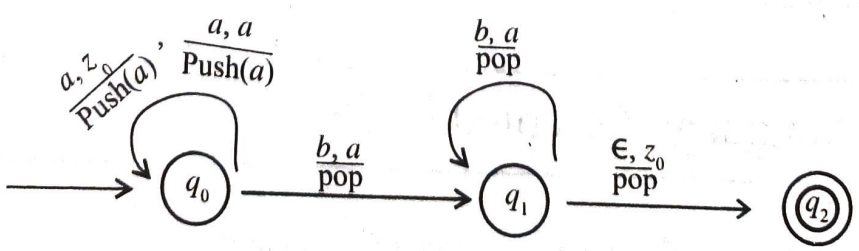


Figure 11.10. *Transition Diagram PDA*

Consider the input string $w = aaabbb$ and z_0 as the current stack top.

Unread input	Transition	Stack	New state
$aaabbb$	–	z_0	q_0
$aabbb$	$(q_0, a, z_0, \text{push}(a), q_0)$	az_0	q_0
$abbb$	$(q_0, a, a, \text{push}(a), q_0)$	aaz_0	q_0
bbb	$(q_0, a, a, \text{push}(a), q_0)$	$aaaz_0$	q_0
bb	$(q_0, b, a, \text{pop}, q_1)$	aaz_0	q_1
b	$(q_1, b, a, \text{pop}, q_1)$	az_0	q_1
$\in$	$(q_1, b, a, \text{pop}, q_1)$	z_0	q_1
–	$(q_1, \in, z_0, \text{nop}, q_2)$	z_0	q_2

Table 11.3 *Transition table*

ID is: $(q_0, aaabbb, z_0) \vdash (q_0, aabbb, az_0)$

$\vdash (q_0, abbb, aaz_0)$

$\vdash (q_0, bbb, aaaz_0)$

$\vdash (q_1, bb, aaz_0)$

$\vdash (q_1, b, az_0)$

$\vdash (q_2, z_0) \Rightarrow (q_0, aaabbb, z_0) \overset{*}{\underset{M}{\vdash}} (q_2, z_0)[-2pt]$

Since, the final state q_2 is reached, stack is empty and the input symbol is consumed (input is $\in$). Thus, the language is accepted by PDA.

11.6 Design of PDAs

The basic design strategy for PDA is as follows:

a. Understand the language properties, for which the PDA has to be designed.
b. Determine the state and alphabet set required.
c. Identify the initial, accepting and dead states of PDA.
d. Decide on the stack symbols required.
e. Determine the initial stack symbol from the stack symbol set.
f. For each state, decide on the transition to be made for each character of the input string.
g. For each state transition, decide on the stack operation to be performed.
h. Obtain the transition diagram and table for PDA.
i. Test, the PDA obtained, on short strings.

EXAMPLE 11.6.1: Design a PDA to accept the language

$$L = \{wCw^R | w \in (0+1)^*\}$$

by an empty stack.

or

Design a PDA that accepts the language of odd palindrome.

Design Strategy: The PDA must operate in the following way. As it reads the first half of its input, it remains in its initial state q_0 and pushes all the symbols from the input string onto the stack. When the machine sees a 'C' symbol in the input string, it switches from state q_0 to state q_1 without operating on stack. Now this causes the removal of the top symbol on the stack, provided that it is same as the next input symbol. If the input symbol does not match the top symbol on the stack, then no further operation is possible. If automaton attains the form

'(final state, end of input or $\in$, empty stack)'

then the input is of the form wCw^R and PDA has accepted the input string. On the other hand, if automaton detects a mismatch between input and stack symbols, or if input is exhausted before the stack is emptied, then it does not accept the string.

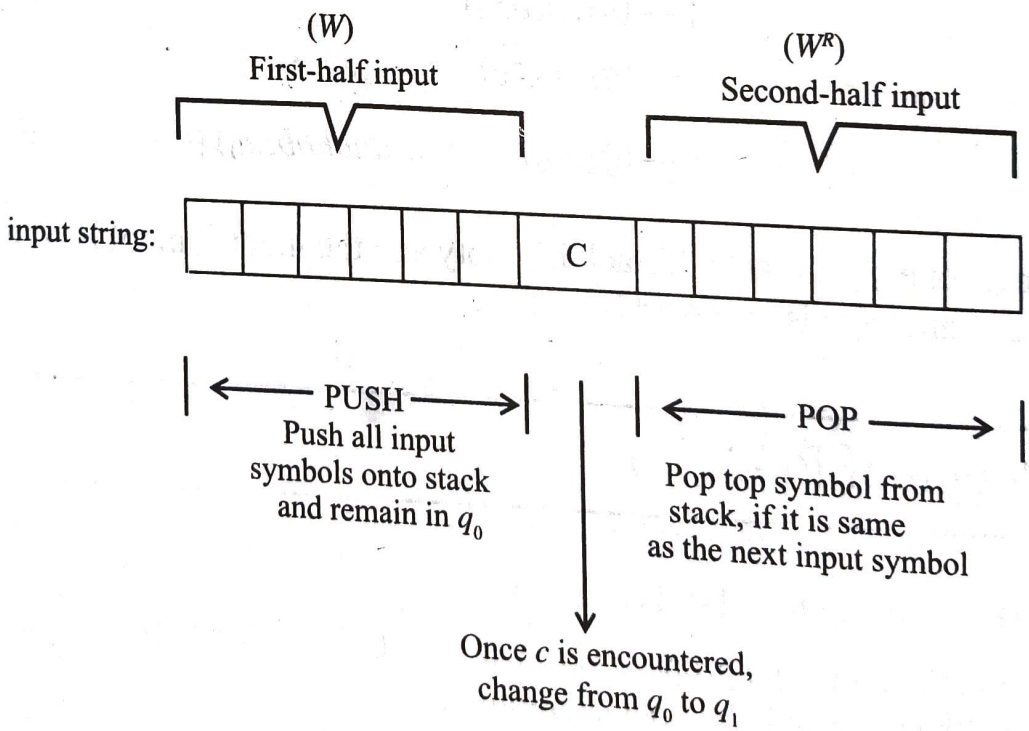


Figure 11.11. *Design View of PDA*

Following are the transitions required:

i. Read each symbol of the input string and push onto stack before C is encountered.
   a. $(q_0, \in, z_0, \text{nop}, q_0)$
   b. $(q_0, 0, z_0, \text{push}(0), q_0)$

c. $(q_0, 1, z_0, \text{push}(1), q_0)$
d. $(q_0, 0, 0, \text{push}(0), q_0)$
e. $(q_0, 1, 1, \text{push}(1), q_0)$
f. $(q_0, 1, 0, \text{push}(1), q_0)$
g. $(q_0, 0, 1, \text{push}(0), q_0)$

ii. Once 'c' is encountered, change to q_1 without performing any operation.
h. $(q_0, C, 0, \text{nop}, q_1)$
i. $(q_0, C, 1, \text{nop}, q_1)$
j. $(q_0, C, z_0, \text{nop}, q_1)$

iii. Read the next input symbol in the second-half and pop from the stack, if there is a match.
k. $(q_1, 0, 0, \text{pop}, q_1)$
l. $(q_1, 1, 1, \text{pop}, q_1)$

iv. Once the stack is empty or the input is ϵ, change to q_2.
m. $(q_1, \epsilon, z_0, \text{nop}, q_2)$

Thus, PDA $= \{\{q_0, q_1, q_2\}, \{0, 1\}\{0, 1, z_0\}, \delta, q_0, z_0, q_2\}$, where δ is given by:

Transition diagram:

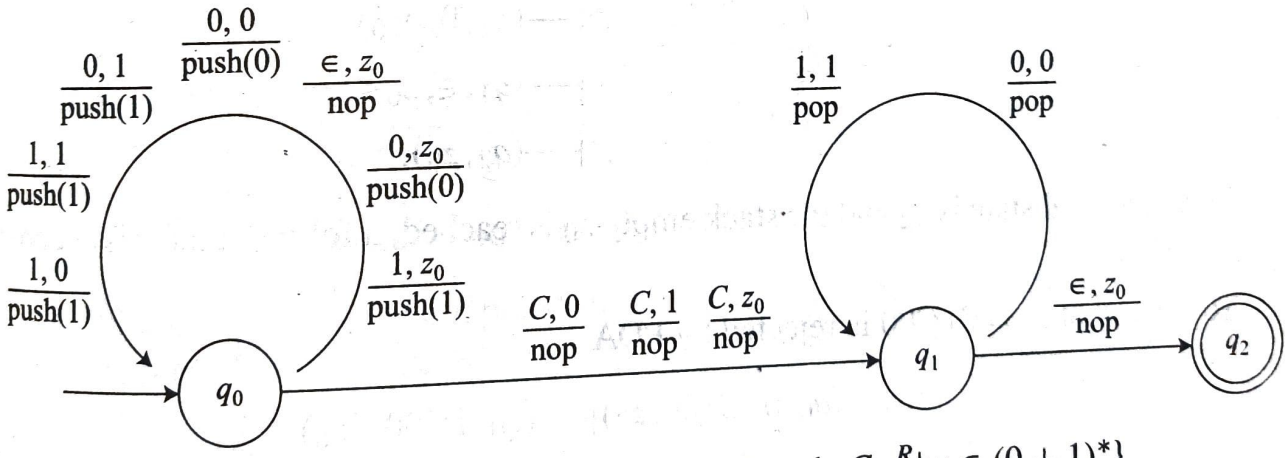


Figure 11.12. *Transition Diagram,* $L = \{wCw^R | w \in (0+1)^*\}$

PDA action for the input string:

Consider the input string $w = 001C100$, whose description is given by:

Transition Table:

Unread input	Transition	Stack	New State
001 C 100	–	z_0	q_0
01 C 100	$(q_0, 0, z_0, \text{push}(0), q_0)$	$0z_0$	q_0

1 C 100	$(q_0, 0, 0, \text{push}(0), q_0)$	$00z_0$	q_0
C 100	$(q_0, 1, 0, \text{push}(1), q_0$	$100z_0$	q_0
100	$(q_0, C, 1, \text{nop}, q_1)$	$100z_0$	q_1
00	$(q_1, 1, 1, \text{pop}, q_1)$	$00z_0$	q_1
0	$(q_1, 0, 0, \text{pop}, q_1)$	$0z_0$	q_1
$\in$	$(q_1, 0, 0, \text{pop}, q_1)$	z_0	q_1
–	$(q_1, \in, z_0, \text{nop}, q_2)$	z_0	q_2

Table 11.4 *Transition table for* $L = \{wCw^R | w \in (0+1)^*\}$

a. To show that $w = 001C100$ is accepted by PDA ID is:

$$\begin{aligned}\text{ID is: } (q_0, 001C100, z_0) &\vdash (q_0, 01C100, 0z_0)\\ &\vdash (q_0, 1C100, 00z_0)\\ &\vdash (q_0, C100, 100z_0)\\ &\vdash (q_1, 100, 100z_0)\\ &\vdash (q_1, 00, 00z_0)\\ &\vdash (q_1, 0, 0z_0)\\ &\vdash (q_1, \in, z_0)\\ &\vdash (q_2, z_0).\end{aligned}$$

Since the final state is q_2 and the stack empty z_0 is reached, it follows that w is accepted by PDA.

b. To show that $w = 01C00$ is rejected by PDA

$$\begin{aligned}\text{ID: } (q_0, 01C100, z_0) &\vdash (q_0, 1C00, 0z_0)\\ &\vdash (q_0, C00, 10z_0)\\ &\vdash (q_1, 00, 10z_0)\end{aligned}$$

The transition is not defined for the above configuration $(q_1, 0, 1)$, hence the string w is rejected by PDA.

EXAMPLE 11.6.2: Construct a PDA to accept $L = \{ww^R | w \in (0+1)^*\}$

or

Build a PDA that accepts the language of even palindrome.

Design Strategy: The PDA must operate in the following way. As it reads the first half of its input, it remains in its initial state q_0 and pushes all the symbols from the input string

onto the stack. When the machine sees the first symbol of the string w^R, then it is the end of first half of its inputs. Now, the machine changes from state q_0 to state q_1 without operating on stack. For the second-half of inputs, each input symbol has to be matched with the top symbol on the stack. If there is no match, then no further operation is possible and we say that the input is rejected by the machine. If there is a match, then the top symbol on the stack is removed and then the next input string is compared. This process continues and we say that the input is of the form ww^R and L is accepted.

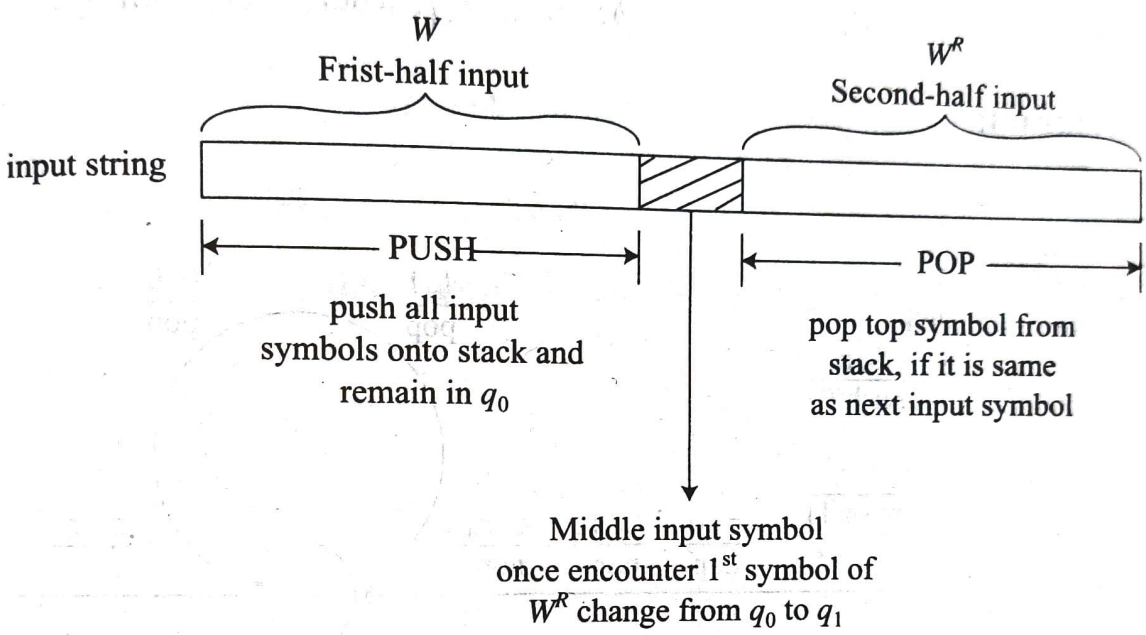


Figure 11.13. *Design View of PDA*

Following are the transitions required:

i. Read each symbol of the input string and push onto stack, before the first symbol of the string w^R is encountered.

   a. $(q_0, \in, z_0, \text{nop}, q_0)$
   b. $(q_0, 0, z_0, \text{push}(0), q_0)$
   c. $(q_0, 1, z_0, \text{push}(1), q_0)$
   d. $(q_0, 0, 0, \text{push}(0), q_0)$
   e. $(q_0, 1, 1, \text{push}(1), q_0)$
   f. $(q_0, 1, 0, \text{push}(1), q_0)$
   g. $(q_0, 0, 1, \text{push}(0), q_0)$

ii. Once the first symbol of the string w^R is encountered, perform the following to change from q_0 to q_1.

   h. $(q_0, \in, 0, \text{nop}, q_1)$
   i. $(q_0, \in, 1, \text{nop}, q_1)$
   j. $(q_0, \in, z_0, \text{nop}, q_1)$

iii. Read the next input symbol in the second-half and pop from the stack, if there is a match.

k. $(q_1, 0, 0, \text{pop}, q_1)$

l. $(q_1, 1, 1, \text{pop}, q_1)$

iv. Once the stack is empty or the input is '$\in$', then change to q_2.

m. $(q_1, \in, z_0, \text{nop}, q_2)$

Thus, $PDA = \{\{q_0, q_1, q_2\}, \{0, 1\}, \{0, 1, z_0\}, \delta, q_0, z_0, q_2\}$, where δ is given by:

Transition diagram:

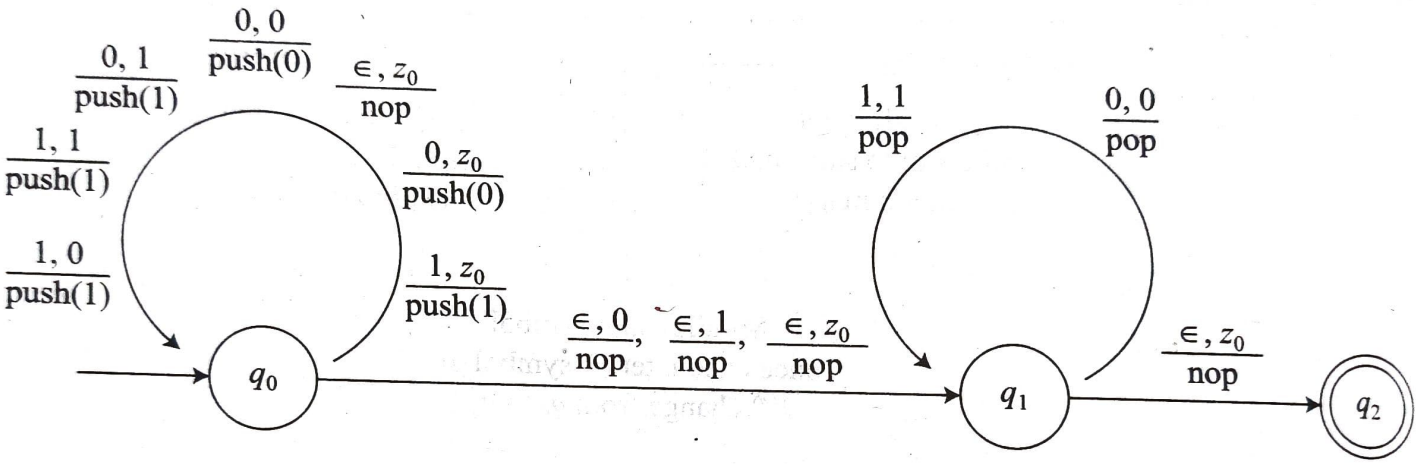


Figure 11.14. *Transition Diagram*

$$L = \{ww^R | w \in (0+1)^*\}$$

PDA Action for the input string:

Consider the input string $w = 0110$, whose description is given by:

Transition table:

Unread input	Transition	Stack	New State
0110	–	z_0	q_0
110	$(q_0, 0, z_0, \text{push}(0), q_0)$	$0z_0$	q_0
10	$(q_0, 1, 0, \text{push}(1), q_0)$	$10z_0$	q_0
10	$(q_0, \in, 1, \text{nop}, q_1)$	$10z_0$	q_1
0	$(q_0, 1, 1, \text{pop}, q_1)$	$0z_0$	q_1
$\in$	$(q_0, 0, 0, \text{pop}, q_1)$	z_0	q_1
–	$(q_0, \in, z_0, \text{nop}, q_2)$	z_0	q_2

Table 11.5 *Transition table for* $L = \{ww^R | w \in (0+1)^*\}$

a. To show that $w = 0110$ is accepted by PDA

$$\begin{aligned}\text{ID: } (q_0, 0110, z_0) &\vdash (q_0, 110, 0z_0)\\ &\vdash (q_0, 10, 10z_0)\\ &\vdash (q_1, 10, 10z_0)\\ &\vdash (q_1, 0, 0z_0)\\ &\vdash (q_1, \in, z_0)\\ &\vdash (q_2, z_0).\end{aligned}$$

Since the final state is q_2 and stack empty z_0 is reached, it follows that w is accepted by PDA.

b. To show that $w = 0100$ is rejected by PDA

$$\begin{aligned}\text{ID: } (q_0, 0100, z_0) &\vdash (q_0, 100, 0z_0)\\ &\vdash (q_0, 00, 10z_0)\\ &\vdash (q_1, 00, 10z_0)\end{aligned}$$

The transition is not defined for the above configuration, hence the string w is rejected by PDA.

EXAMPLE 11.6.3: Obtain a PDA to accept the language $L = \{a^n b^n | n \geq 1\}$.

Design Strategy: The PDA must operate in the following way. As it reads the input symbol 'a', it pushes the symbol onto stack top and remains in the state q_0. This process is continued if the next input symbol is also 'a'. Once an input symbol 'b' is encountered, then the machine switches from q_0 to q_1 and this causes the removal of the top symbol on the stack. In the state q_1, for every 'b' encountered in the input symbol, pop operation (with stack top as 'a') is performed. If the machine reaches the final state or it is the end of input or empty stack, then the input is in the form $a^n b^n$.

Following are the transitions required:

i. Read each symbol of the input string i.e., 'a' and push onto stack, until 'b' is encountered.
   a. $(q_0, a, z_0, \text{push}(a), q_0)$
   b. $(q_0, a, a, \text{push}(a), q_0)$

ii. Once 'b' is encountered, change to q_1 by performing pop operations.
   c. $(q_0, b, a, \text{pop}, q_1)$
   d. $(q_1, b, a, \text{pop}, q_1)$

iii. Once the stack is empty or input is $\in$, change to q_2.
   e. $(q_1, \in, z_0, \text{nop}, q_2)$

Thus, $PDA = \{\{q_0, q_1, q_2\}, \{a, b\}, \{a, z_0\}, \delta, q_0, z_0, q_2\}$, where δ is given by:

Transition diagram:

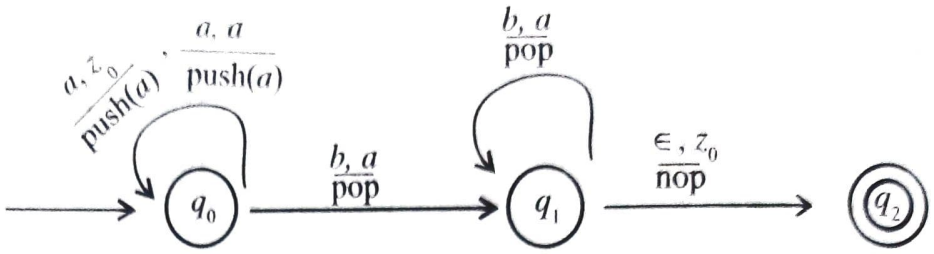


Figure 11.15. *Transition Diagram* $L = \{a^n b^n | n \geq 1\}$

PDA action for the input string:

Consider the input string $w = aaabbb$, whose description is given by:

Transition table:

Unread input	Transition	Stack	New State
$aaabbb$	–	z_0	q_0
$aabbb$	$(q_0, a, z_0, \text{push}(a), q_0)$	az_0	q_0
$abbb$	$(q_0, a, a, \text{push}(a), q_0)$	aaz_0	q_0
bbb	$(q_0, a, a, \text{push}(a), q_0)$	$aaaz_0$	q_0
bb	$(q_0, b, a, \text{pop}, q_1)$	aaz_0	q_1
b	$(q_1, b, a, \text{pop}, q_1)$	az_0	q_1
$\in$	$q_1, b, a, \text{pop}, q_1)$	z_0	q_1
–	$(q_1, \in, z_0, \text{nop}, q_2)$	z_0	q_2

Table 11.6 *Transition table for* $L = \{a^n b^n | n \geq 1\}$

To show that $aaabbb$ is accepted by PDA:

$$\begin{aligned}\text{ID: } (q_0, aaabbb, z_0) &\vdash (q_0, aabbb, az_0)\\ &\vdash (q_0, abbb, aaz_0)\\ &\vdash (q_0, bbb, aaaz_0)\\ &\vdash (q_1, bb, aaz_0)\\ &\vdash (q_1, b, az_0)\\ &\vdash (q_1, \in, z_0)\\ &\vdash (q_2, z_0).\end{aligned}$$

Since the final state is q_2 and stack empty z_0 is reached, it follows that w is accepted by PDA.

EXAMPLE 11.6.4: Design a PDA to accept the language $L = \{w \in \{a, b\}^* | n_a = n_b\}$.

Design strategy: The PDA must operate in the following way. To start with, irrespective of what is there in input symbol, input symbol is pushed onto the stack. Next, if the input symbol is same as the top of the stack, the symbol is pushed onto stack, otherwise popped from stack. This process continues and if the automata reaches the end of input ($\in$) or empty stack, then the input is of the form $n_a = n_b$ and L is accepted by the PDA.

Following are the transitions required:

i. Read each symbol of the input string, irrespective of what the input string is and push it onto stack.
   a. $(q_0, a, z_0, \text{push}(a), q_0)$.
   b. $(q_0, b, z_0, \text{push}(b), q_0)$.

ii. Push onto stack the input symbol, if it is same as the top of the stack.
   c. $(q_0, a, a, \text{push}(a), q_0)$
   d. $(q_0, b, b, \text{push}(b), q_0)$

iii. If the input symbol is not the same as that of the stack top, then pop.
   e. $(q_0, a, b, \text{pop}, q_0)$
   f. $(q_0, b, a, \text{pop}, q_0)$.

iv. Once the stack is empty or the input is $\in$, change to q_1.
   g. $(q_0, \in, z_0, \text{nop}, q_1)$

Thus, PDA = $\{\{q_0, q_1\}, \{a, b\}, \{a, b, z_0\}, \delta, q_0, z_0, q_1\}$, where δ is given by:

Transition diagram:

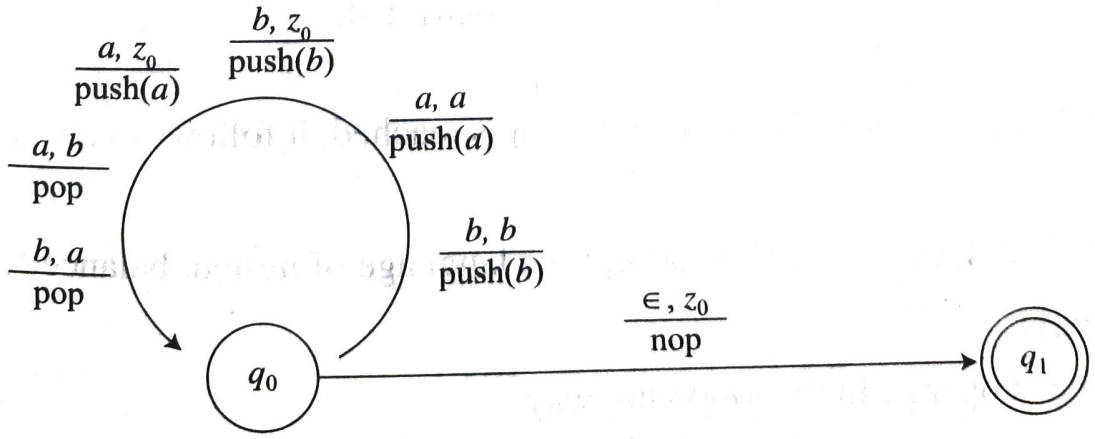


Figure 11.16. *Transition Diagram,* $L = \{w \in (a, b)^* | n_a = n_b\}$

PDA action for the input string:

Consider the input string $w = abbbabaa$, whose description is given by:

Transition table:

Unread input	Transition	Stack	New state
$abbbabaa$	–	z_0	q_0
$bbbabaa$	$(q_0, a, z_0, \text{push}(a), q_0)$	az_0	q_0
$bbabaa$	$(q_0, b, a, \text{pop}, q_0)$	z_0	q_0
$babaa$	$(q_0, b, z_0, \text{push}(b), q_0)$	bz_0	q_0
$abaa$	$(q_0, b, b, \text{push}(b), q_0)$	bbz_0	q_0
baa	$(q_0, a, b, \text{pop}, q_0)$	bz_0	q_0
aa	$(q_0, b, b, \text{push}(b), q_0)$	bbz_0	q_0
a	$(q_0, a, b, \text{pop}, q_0)$	bz_0	q_0
$\in$	$(q_0, a, b, \text{pop}, q_0)$	z_0	q_0
–	$(q_0, \in, z_0, \text{nop}, q_1)$	z_0	q_1

Table 11.7 *Transition table for* $L = \{w \in \{a, b\}^* | n_a = n_b\}$

To show that $w = abbbabaa$ is accepted by PDA:

$$\begin{aligned}\text{ID: } (q_0, abbbabaa, z_0) &\vdash (q_0, bbbabaa, az_0)\\ &\vdash (q_0, bbabaa, z_0)\\ &\vdash (q_0, babaa, bz_0)\\ &\vdash (q_0, abaa, bbz_0)\\ &\vdash (q_0, baa, bz_0)\\ &\vdash (q_0, aa, bbz_0)\\ &\vdash (q_0, a, bz_0)\\ &\vdash (q_0, \in, z_0)\\ &\vdash (q_1, z_0).\end{aligned}$$

Since the final state is q_1 and the stack empty z_0 is reached, it follows that w is accepted by PDA.

EXAMPLE 11.6.5: Design a PDA to accept the language of nested, balanced parentheses.

Design strategy:

The PDA must operate in the following way:

- Push all the symbols of type '(' onto the stack.

- If the input symbol of type ')' is encountered, then remove all the symbols from the stack until the stack becomes empty.

Following are the transitions required:

i. To read each symbol of the input string of type '(' and push onto stack, until ')' is encountered.
   a. $(q_0, (, z_0, \text{push}(\,(), q_0)$
   b. $(q_0, (, (, \text{push}(\,(), q_0)$

ii. Once b is encountered, change to q_1 by performing pop operations.
   c. $(q_0,), (, \text{pop}, q_1)$
   d. $(q_1,), (, \text{pop}, q_1)$

iii. If the input is $\in$, change to final state q_2.
   e. $(q_1, \in, z_0, \text{nop}, q_2)$

Thus, $PDA = \{\{q_0, q_1, q_2\}, \{(,)\}, \{(, z_0\}, \delta, q_0, z_0, q_2\}$, where δ is given by:

Transition diagram:

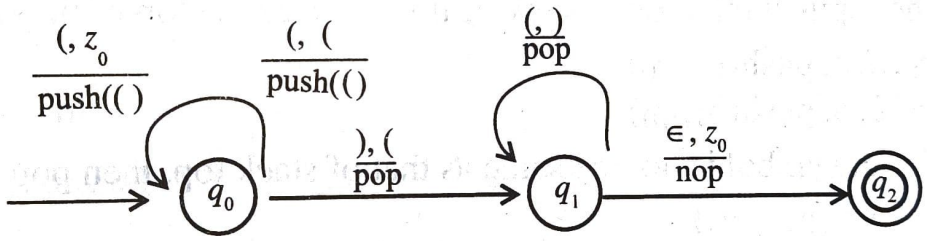


Figure 11.17. *Transition Diagram to Accept Balanced Parenthesis*

PDA action for the input string

To show that $w = ((()))$ is accepted by PDA:

$$\begin{aligned}\text{ID:}\quad (q_0, ((())), z_0) &\vdash (q_0, (())), (z_0)\\ &\vdash (q_0, ())), ((z_0)\\ &\vdash (q_0,))), (((z_0)\\ &\vdash (q_1,)), ((z_0)\\ &\vdash (q_1,), (z_0)\\ &\vdash (q_1, \in, z_0)\\ &\vdash (q_2, z_0)\end{aligned}$$

Since the final state is q_2 and the stack empty z_0 is reached, it follows that w is accepted by PDA.

EXAMPLE 11.6.6: Design a PDA to accept $L = \{w|w \in (a,b)^*\}$, such that;

i. $n_a(w) > n_b(w)$ and

ii. $n_a(w) < n_b(w)$

Design strategy: The PDA must operate in the following way. To start with, irrespective of what is there in the input symbol, the input symbol is pushed onto the stack. Next, if the input symbol is same as the top of the stack, then the symbol is pushed onto stack, otherwise pop from stack. This process is continued.

i. To show that PDA accepts more number of a's than b's i.e $n_a(w) > n_b(w)$.

If the automaton reaches the form of 'end of input $(\in)$', and if the top of the stack contains atleast one a, then change to state q_1 and perform no operation.

Following are the transitions required:

a. Read each symbol of the input string, irrespective of what the input string is and push it onto stack.
   - $(q_0, a, z_0, \text{push}(a), q_0)$
   - $(q_0, b, z_0, \text{push}(b), q_0)$

b. Push the input symbol onto stack, if it is same as the top of the stack.
   - $(q_0, a, a, \text{push}(a), q_0)$
   - $(q_0, b, b, \text{push}(b), q_0)$

c. If the input symbol is not the same as that of stack top, then pop.
   - $(q_0, a, b, \text{pop}, q_0)$
   - $(q_0, b, a, \text{pop}, q_0)$.

d. If the automata reaches the form of 'end of input' and if the stack top contains atleast one a, then change to state q_1.
   - $(q_0, \in, a, \text{nop}, q_1)$.

Thus, *PDA* for $L = \{w|w \in (a,b)^*|n_a(w) > n_b(w)\}$ is

$$PDA = \{\{q_0, q_1, \{a,b\}, \{a,b,z_0\}, \delta, q_0, z_0, q_1\}.$$

ii. To show that PDA accepts more number of b's than a's i.e., $n_a(w) < n_b(w)$.
If the automata reaches the form of 'end of input $(\in)$', and if the top of the stack contains atleast one b, then change to state q_1 and perform no operation.

The transitions required are:

a. Read each symbol of the input string, irrespective of what the input string is and push onto stack.
   - $(q_0, a, z_0, \text{push}(a), q_0)$.
   - $(q_0, b, z_0, \text{push}(b), q_0)$.

b. Push the input symbol onto stack, if it is same as the top of the stack.
   - $(q_0, a, a, \text{push}(a), q_0)$.
   - $(q_0, b, b, \text{push}(b), q_0)$.

c. If the input symbol is not the same as that of stack top, then POP.
   - $(q_0, a, b, \text{pop}, q_0)$.
   - $(q_0, b, a, \text{pop}, q_0)$.

d. If the automata reaches the form of 'end of input' and if stact top contains atleast one b, then change to state q_1.
   - $(q_0, \in, b, \text{nop}, q_1)$.

Thus, PDA for $L = \{w | w \in (a,b)^* | n_a(w) < n_b(w)\}$ is

$$PDA = \{\{q_0, q_1\}, \{a, b\}, \{a, b, z_0\}, \delta, q_0, z_0, q_1\}.$$

11.7 Determinism and Nondeterminism

Based on the processing of input string by the machine, a PDA can be:

- Deterministic PDA
- Nondeterministic PDA

11.7.1 *Deterministic PDA*

A PDA is deterministic, if each input string can only be processed by the machine in only one way, i.e., for the same input symbol and same stack symbol, there must be only one choice.

Formally, a PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ is deterministic if

i. $\delta(q, a, z)$ has only one element
ii. $\delta(q, \in, z)$ is not empty, then $\delta(q, a, z)$ should be empty.

If conditions (i) and (ii) are satisfied, then the PDA is deterministic, otherwise PDA is nondeterministic.

EXAMPLE 11.7.1: A PDA for simple nested parentheses strings is deterministic $L() = \{ \in$ $, (), ()(), (()), ((())), ()()(), \cdots \}$.

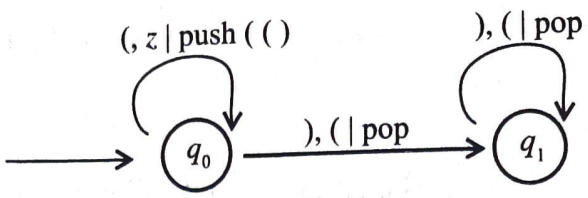


Figure 11.18. *Transition Diagram to Demonstrate Deterministic PDA*

In the above diagram, it is clear that for each input string machine processes in one way. For example, $\delta(q_0, (, z_0) = (q_0, ( )$ or $(q_0, (, z_0, \text{Push}(( ), q_0)$.

EXAMPLE 11.7.2: The PDA for $L = \{a^n b^n | n \geq 1\}$ is deterministic.

11.7.2 *Nondeterministic PDA*

A PDA is nondeterministic, if there is some string that can be processed by it in more than one way.

There are two types of nondeterministic PDA that may occur (two types of moves):

a. First kind of nondeterminism occurs, when a state emits two or more edges labelled with the same input symbol and same stack symbol.

Formally, for the same input symbol and same stack symbol there exists number of choices. This is represented as:

$$\delta(q, a, z) = \{(p_1, \alpha_1), (p_2, \alpha_2), \ldots, (p_n, \alpha_n)\} \text{ where}$$
p_i and q are states, $a \in \Sigma$, z is a stack symbol and $\alpha_i \in \top^*$.

EXAMPLE 11.7.3: First kind of non-determinism

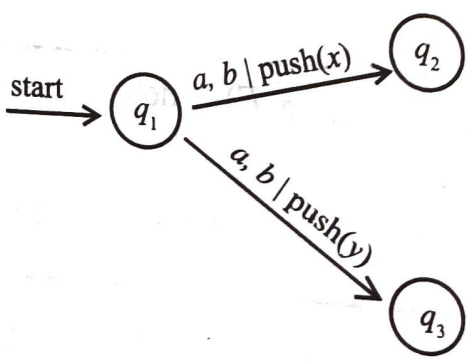


Figure 11.19. *Transition Diagram to Demonstrate Nondeterminism for a State that Emits Two Edges Labelled with Same Input and Same State Symbol*

In the above figure, it is seen that for the same input 'a' and for the same stack top 'b', there are two choices:

- $\delta(q_1, a, b) = (q_2, x)$ or $(q_1, a, b, \text{push}(x), q_2)$
- $\delta(q_1, a, b) = (q_3, y)$ or $(q_1, a, b, \text{push}(y), q_3)$

b. Second kind of nondeterminism occurs, when a state emits two edges labelled with the same stack symbol, where one input symbol is '$\in$' and the other input symbol is not.

Formally, $\delta(q_1 \in, z) = \{(p_1, \alpha_1), (p_2, \alpha_2), \ldots, (p_n, \alpha_n)\}$
and $\delta(q, a, z) = \{(p_1, \alpha_1), (p_2, \alpha_2), \ldots, (p_n, \alpha_n)\}$.

EXAMPLE 11.7.4: Second kind of non-determinism

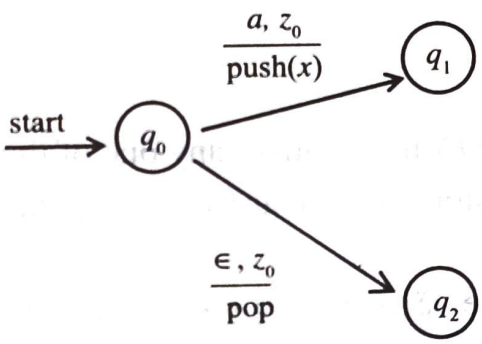


Figure 11.20. *Transition Diagram to Demonstrate Non-determinism for a State that Emits Two Edges, One Labelled with 'a' and the Other with ∈, with the Same Stack Symbol*

In the above figure, for the same stack symbol, there are two choices based on the different inputs:

- $(q_0, a, z_0, \text{push}(x), q_1)$
- $(q_0, \in, z_0, \text{nop}, q_2)$

EXAMPLE 11.7.5: Examples of nondeterministic PDA

a. Language $L = \{ww^R | w \in (a, b)^*\}$

A language of PALINDROME with even length of words, that reads the same in forward as well as backward directions:

$$\text{PALINDROME} = \{\in, aa, bb, aaaa, abba, baab, bbbb, \cdots\}$$

b. Language $L = \{w \in \{a, b\}^* | n_a = n_b\}$.

Note-1:

- for finite automata, nondeterminism **does not** increase the power of machines.
- for PDA's nondeterminism **does** increase the power of machines.

EXAMPLE 11.7.6: Show that the PDA, that accepts the language $L = \{w \in \{a, b\}^* | n_a = n_b\}$, is nondeterministic.

Solution: The transitions required to design a PDA for $L = \{w \in \{a, b\}^* | n_a = n_b\}$

i. $(q_0, a, z_0, \text{push}(a), q_0)$
ii. $(q_0, b, z_0, \text{push}(b), q_0)$
iii. $(q_0, a, a, \text{push}(a), q_0)$
iv. $(q_0, b, b, \text{push}(b), q_0)$
v. $(q_0, a, b, \text{pop}, q_0)$
vi. $(q_0, b, a, \text{pop}, q_0)$
vii. $(q_0, \in, z_0, \text{nop}, q_1)$

To be nondeterministic, the PDA must satisfy any one of the following two conditions:

Condition-1: For the same input symbol and the same stack symbol, there exists number of choices,

$$\text{i.e. } \delta(q, a, z) = \{(p_1, \alpha_1), \ldots (p_n, \alpha_n)\}.$$

Since there is only one transition for the same input symbol and the same stack symbol, condition-1 is not satisfied.

Condition-2: For the same stack symbol, where one input symbol is '$\in$' and the other input symbol is not, there exist the following conditions:

$$\delta(q, \in, z) = \{(p_1, \alpha_1), \ldots (p_n, \alpha_n)\}$$

$$\text{and} \quad \delta(q, a, z) = \{(p_1, \alpha_1), \ldots (p_n, \alpha_n)\}.$$

The given PDA has the following transition for the same stack symbol, with only one input symbol as '$\in$':

$$(q_0, \in, z_0, \text{nop}, q_1)$$
$$(q_0, a, z_0, \text{push}(a), q_0)$$
$$(q_0, b, z_0, \text{push}(b), q_0)$$

Hence condition-2 is satisfied and the given PDA

$$L = \{w \in \{a, b\}^* | n_a = n_b\}$$

is nondeterministic.

EXAMPLE 11.7.7: Show that the PDA, that accepts the Language $L = \{ww^R | w \in (a+b)^*\}$, is nondeterministic.

Solution: The transitions required to design a PDA for $L = \{ww^R | w \in (a+b)^*\}$ are:

i. $(q_0, \in z_0, \text{nop}, q_0)$
ii. $(q_0, 0, z_0, \text{push}(0), q_0)$
iii. $(q_0, 1, z_0, \text{push}(1), q_0)$
iv. $(q_0, 0, 0, \text{push}(0), q_0)$
v. $(q_0, 1, 1, \text{push}(1), q_0)$
vi. $(q_0, 1, 0, \text{push}(1), q_0)$
vii. $(q_0, 0, 1, \text{push}(0), q_0)$
viii. $(q_0, \in, 0, \text{nop}, q_1)$
ix. $(q_0, \in, 1, \text{nop}, q_1)$
x. $(q_0, \in, z_0, \text{nop}, q_1)$
xi. $(q_1, 0, 0, \text{nop}, q_1)$
xii. $(q_1, 1, 1, \text{pop}, q_1)$
xiii. $(q_1, \in, z_0, \text{nop}, q_2)$

To be nondeterministic, the PDA must satisfy any one of the following two conditions:

Condition 1: For the same input symbol and the same stack symbol, there exists a number of choices

$$\text{i.e., } \delta(q, a, z) = \{(P_1, \alpha_1) \cdots (P_n, \alpha_n)\}.$$

For the same input symbol and the same stack symbol, the given PDA has the following:
The transitions (viii) and (ix) are used by the PDA to change over from state q_0 to q_1, once the first symbol of the string w^R is encountered (which could be either 0 or 1). Thus, the transitions can also be written as

$$(q_0, 0, 0, \text{nop}, q_1) \quad \ldots (\text{viii}^*)$$

$$(q_0, 1, 1, \text{nop}, q_1) \quad \ldots (\text{ix}^*).$$

Now from transitions (iv) and (viii*), it is clear that

$$\delta(q_0, 0, 0) = \{(q_0, 00), (q_1, \text{nop})\}.$$

From transitions (v) and (ix*), it is seen that

$$\delta(q_0, 1, 1) = \{(q_0, 11), (q_1, \text{nop})\}.$$

Thus, for the same input symbol and the same stack symbol, there are number of choices. Hence condition-1 is satisfied and the given PDA is nondeterministic.

EXAMPLE 11.7.8: Show that the PDA, that accepts the language $L = \{a^n b^n | n \geq 1\}$, is deterministic.

Solution: The transitions required to design a PDA for $L = \{a^n b^n | n \geq 1\}$ are:

i. $(q_0, a, z_0, \text{push}(a), q_0)$
ii. $(q_0, a, a, \text{push}(a), q_0)$
iii. $(q_0, b, a, \text{pop}, q_1)$
iv. $(q_1, b, a, \text{pop}, q_1)$
v. $(q_1, \in, z_0, \text{nop}, q_2)$

To be deterministic, the PDA must satisfy the following two conditions:

Condition 1: For the same input symbol and the same stack symbol, there must be only one choice,

$$\text{i.e. } \delta(q, a, z) = (p, \alpha).$$

Since there is only one transition for the same input symbol and the same stack symbol for the given PDA, condition-1 is satisfied.

Condition-2: For the same stack symbol, where one input symbol is $\in$ and the other input symbol is not, it is observed that

$$\delta(q, \in, z) = (p, \alpha) \text{ is defined,}$$

$$\text{and } \delta(q, a, z) = (p, \alpha) \text{ is not defined.}$$

For the given PDA, $(q_1, \in, z_0, \text{nop}, q_2)$ is defined and $(q_1, a, z_0, \text{nop}, q_2)$ is not defined.

Hence, both condition-1 and condition-2 are satisfied, from which it follows that the given PDA, $L = \{a^n b^n | n \geq 1\}$ is deterministic.

EXAMPLE 11.7.9: Show that the PDA, that accepts the language consisting of balanced parentheses, is deterministic.

Solution: The transitions required to design a PDA for balanced parentheses are

i. $(q_0, (, z_0, \text{push}((), q_0)$
ii. $(q_0, (, (, \text{push}((), q_0)$
iii. $(q_0,), (, \text{pop}, q_1)$

iv. $(q_1,), (, \text{pop}, q_1)$

v. $(q_1, \in, z_0, \text{nop}, q_2)$

To be deterministic, the PDA must satisfy the following two conditions:

Condition-1: For the same input symbol and the same stack symbol, there must be only one choice.

$$\text{i.e., } \delta(q, a, z) = (p, \alpha).$$

Since there is only one transition for the same input symbol and the same stack symbol, condition-1 is satisfied for the given PDA.

Condition-2: For the same stack symbol, where one input symbol is $\in$ and other input symbol is not,

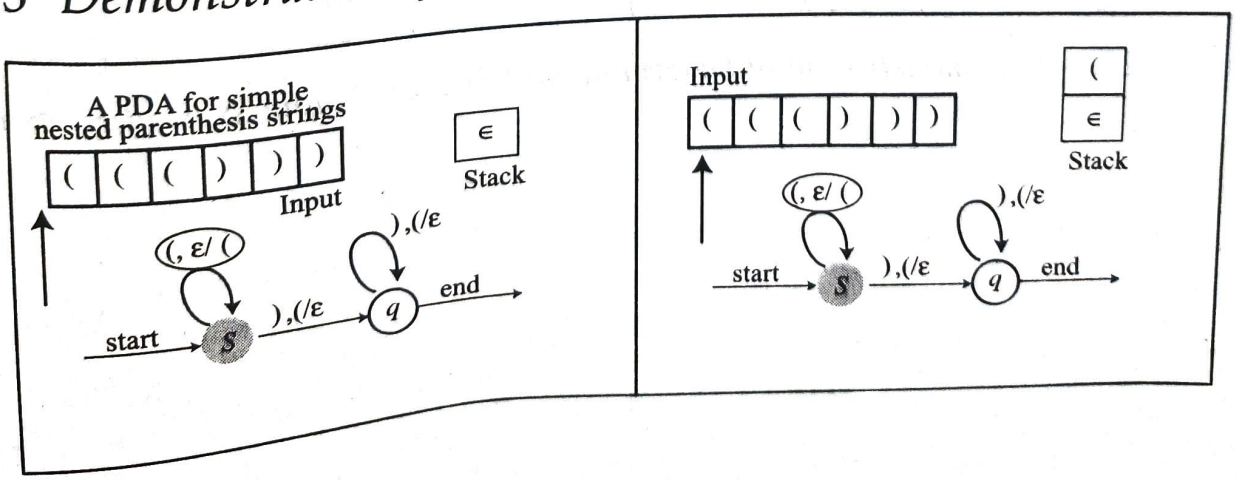
$$\delta(q, \in, z) = (p, \alpha) \text{ is defined}$$

$$\text{and } \delta(q, a, z) = (p, \alpha) \text{ is not defined.}$$

For the given PDA, $(q_1, \in, z_0, \text{nop}, q_2)$ is defined and $(q_1, a, z_0, \text{nop}, q_2)$ is not defined.

Hence both condition-1 and condition-2 are satisfied and the given PDA is deterministic.

11.7.3 *Demonstration of Deterministic PDA*



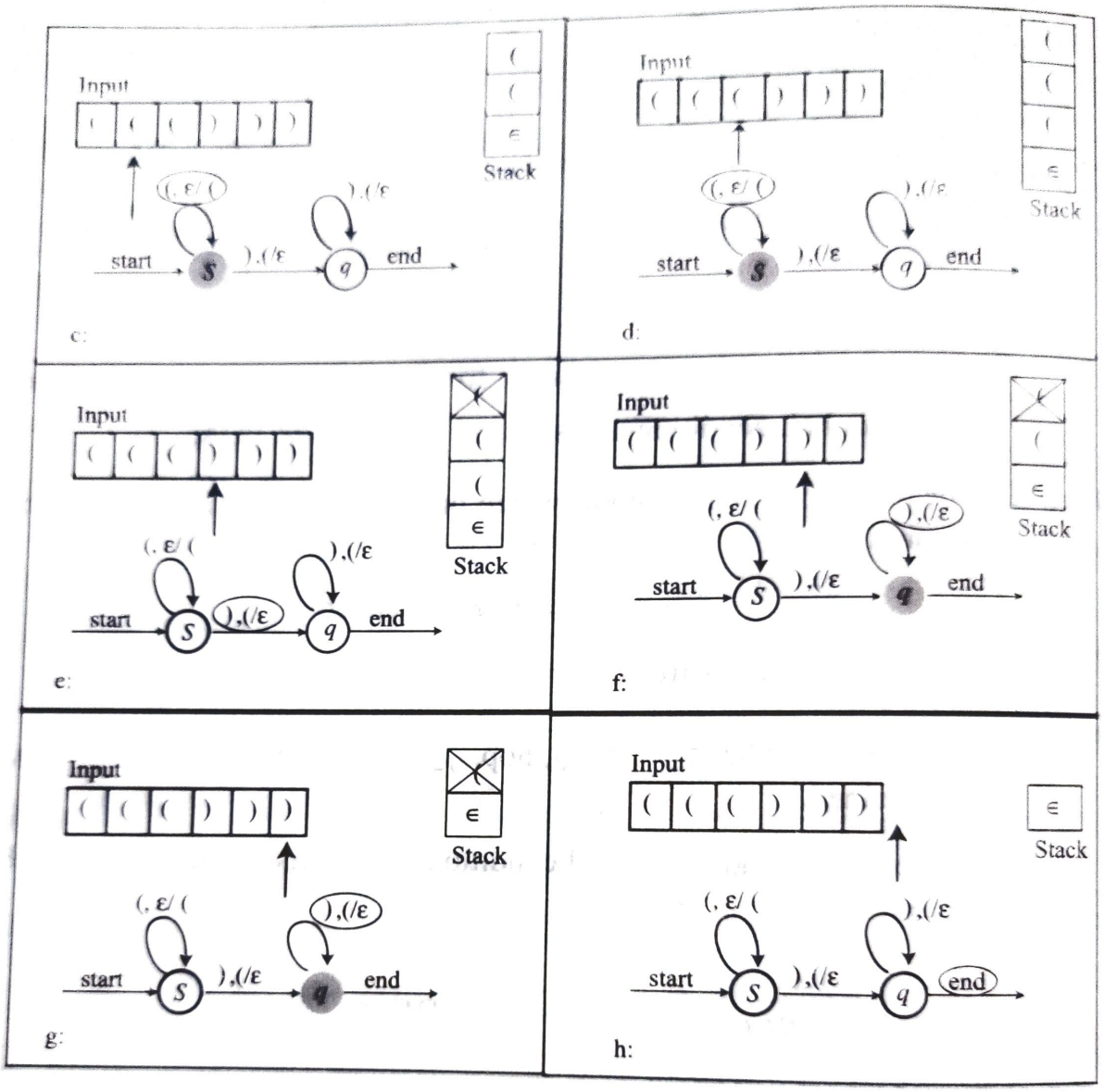


Figure 11.21.*a-h: Demonstration of Deterministic PDA for Simple, Nested Parentheses*

11.7.4 Demonstration of Nondeterministic PDA

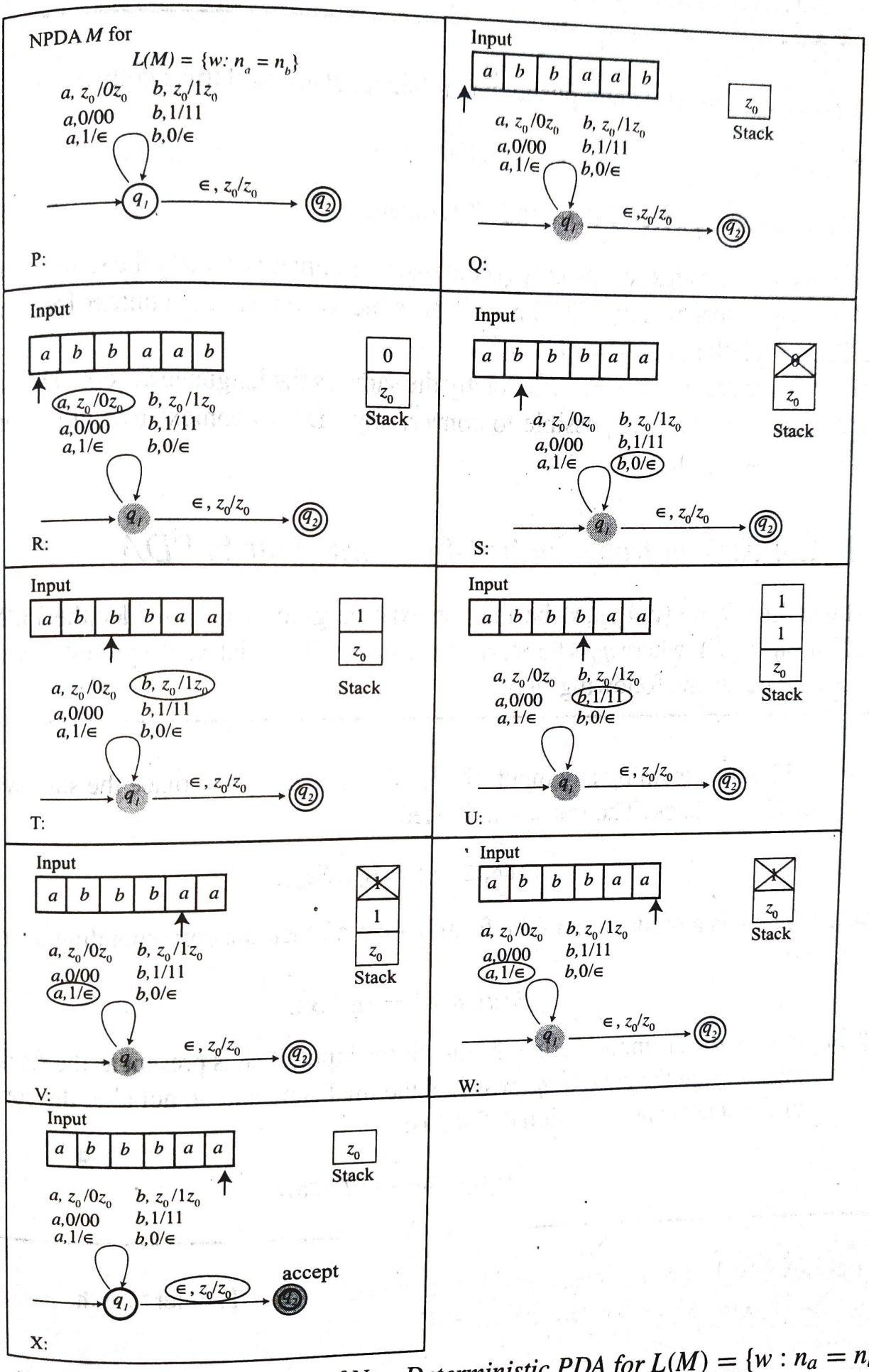


Figure 11.22. *P-X: Demonstration of Non-Deterministic PDA for* $L(M) = \{w : n_a = n_b\}$

11.8 Equivalence of PDA and CFL

The context-free grammar G and pushdown automata P are said to be equivalent iff

$$L(G) = L(P).$$

In other words, equivalence of PDA and CFL means:

a. the class of languages accepted by context-free grammar is exactly the same as the class of languages accepted by PDA i.e., it is possible to convert any context-free grammar to PDA, such that, $L(G) = L(P)$
b. the language accepted by PDA is exactly the same as the language accepted by context-free grammar, i.e., it is possible to convert any PDA to context-free grammar, such that, $L(P) = L(G)$.

11.8.1 *Conversion from Context-Free Grammar to PDA*

Procedure: Let $G = (v, \Sigma, p, s)$ be the context-free grammar. Consider the PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$, where q_0 is the start state and z_0 is the initial stack symbol. Conversion from 'G' to P needs the following steps:

Step-1: Without consuming any input, change the state to q_1 and place the start symbol of G onto stack. The transition defined is:

$$\delta(q_0, \in, z_0) = (q_1, Sz_0).$$

Step-2: If there is a production of the form $A \to a \propto$, then the corresponding transition is:

$$\delta(q_1, a, A) = (q_1, \propto).$$

Step-3: In state q_1, on encountering the end of the input, if z_0 is present as the stack top, then change the state to q_2 which is the final state and do not alter the contents of the stack. The transition defined is:

$$\delta(q_1, \in, z_0) = (q_2, z_0).$$

Note-2: Apply the above steps, only if the grammar is in GNF, otherwise the grammar G has to be first converted into GNF.

EXAMPLE 11.8.1: Construct a PDA for the grammar

$$S \to aAA$$
$$A \to aS|bS|a.$$

Solution: The grammar

$$S \to aAA$$
$$A \to aS$$
$$A \to bS$$
$$A \to a$$

is in GNF.

The equivalent PDA, $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ where

$$Q = \{q_0, q_1, q_2\}, \quad \Sigma = \{a, b\}, \quad \Gamma = \{S, A, z_0\}.$$

'δ' is defined as follows:

- Push the starting symbol 'S' onto the stack

$$\delta(q_0, \in, z_0) = (q_1, Sz_0).$$

-

Grammar	PDA
$S \to aAA$	$\delta(q_1, a, S) = (q_1, AA)$
$A \to aS$	$\delta(q_1, a, A) = (q_1, S)$
$A \to bS$	$\delta(q_1, b, A) = (q_1, S)$
$A \to a$	$\delta(q_1, a, A) = (q_1, \in)$

- Change q_1 to final state q_2

$$\delta(q_1, \in, z_0) = (q_2, z_0).$$

EXAMPLE 11.8.2: Construct a PDA for the grammar

$$S \to aA$$
$$A \to aABD|bB|a$$
$$B \to b$$
$$D \to d.$$

Solution: The grammar

$$S \to aA$$
$$A \to aABD$$
$$A \to bB$$
$$A \to a$$
$$B \to b$$
$$D \to d$$

is in GNF.

The equivalent PDA, $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ where

$$Q = \{q_0, q_1, q_2\}, \quad \Sigma = \{a, b, d\}, \quad \Gamma = \{S, A, B, D, z_0\}.$$

δ is defined as follows:

- Push, the starting symbol 'S' onto the stack

$$\delta(q_0, \in, z_0) = (q_1, Sz_0).$$

-

Grammar	PDA
$S \to aA$	$\delta(q_1, a, S) = (q_1, A)$
$A \to aABD$	$\delta(q_1, a, A) = (q_1, ABD)$
$A \to bB$	$\delta(q_1, b, A) = (q_1, B)$
$A \to a$	$\delta(q_1, a, A) = (q_1, \in)$
$B \to b$	$\delta(q_1, b, B) = (q_1, \in)$
$D \to d$	$\delta(q_1, d, D) = (q_1, \in)$

- Change q_1 to the final state q_2:

$$\delta(q_1, \in, z_0) = (q_2, z_0).$$

EXAMPLE 11.8.3: Construct the PDA for the grammar

$$S \to aSbb|a.$$

Solution: The given grammar is not in GNF. The GNF of this grammar is

$$S \to aSA|a$$
$$A \to bB$$
$$B \to b.$$

The equivalent PDA $P = (Q, \Sigma, \Gamma, \delta, q_0, z_0, F)$ where

$$Q = \{q_0, q_1, q_2\}, \quad \Sigma = \{a, b\}, \quad F = \{S, A, B, z_0\}.$$

δ is defined as

- Push the starting symbol 'S' onto stack

$$\delta(q_0, \in, z_0) = (q_1, Sz_0).$$

-

Grammar	PDA
$S \rightarrow aSA$	$\delta(q_1, a, S) = (q_1, SA)$
$S \rightarrow a$	$\delta(q_1, a, S) = (q_1, \in)$
$A \rightarrow bB$	$\delta(q_1, b, A) = (q_1, B)$
$B \rightarrow b$	$\delta(q_1, b, B) = (q_1, \in)$

- Change q_1 to the final state q_2, so that

$$\delta(q_1, \in z_0) = (q_2, z_0).$$

11.9 Exercises

1. What is a PDA? Explain with an example.
2. Explain the components of PDA, using a neat diagram.
3. Explain why finite automata are less powerful than PDAs.
4. Explain how PDAs are represented using transition diagrams.
5. Present the formal definition of PDA, with examples.
6. Formally define the concepts of string and language acceptance for PDAs.
7. Design PDAs to accept the following languages over $\Sigma = \{0, 1\}$:
   a. 0^*
   b. $\{0^i 1^i 0^j 1^j | i, j \geq 0\}$
   c. $\{0^{2i} 1^i | i \geq 1\}$
   d. $\{0^m 1^n | m \neq n\}$
8. Obtain the ID of a PDA to accept the language of balanced parentheses, for the following inputs:
   a. $\in$
   b. ((( )
   c. ( )))
   d. ((( )))

9. Design a PDA to accept the following languages:
   a. $\{xx|x \in \{0,1\}^*\}$
   b. $\{x|x \in \{0,1\}^* \text{ and } x = x^R\}$
   c. $\{0^m 1^n | 3n \leq m \leq 7n\}$
10. Define a deterministic and nondeterministic PDA with examples.
11. Show that the PDA $L = \{a^n b^{2n} | n \geq 1\}$ is deterministic.
12. Consider a PDA (M) with n states and m input alphabet symbols. What is the maximum possible number of rejecting conputations, that m could have on an input of length k?
13. Prove that, for a context-free grammar G, there is an equivalent pushdown automata P such that $L(G) = L(P)$.
14. For the grammar

$$S \rightarrow aABC$$
$$A \rightarrow aB|a$$
$$B \rightarrow bA|b$$
$$C \rightarrow a,$$

obtain the corresponding PDA.

15. Obtain the PDA for the CFG given below:

$$S \rightarrow aABB|aAA$$
$$A \rightarrow aBB|a$$
$$B \rightarrow bBB|A$$
$$C \rightarrow a.$$