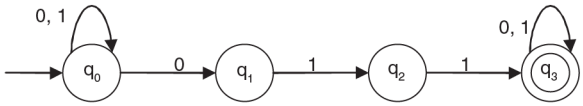
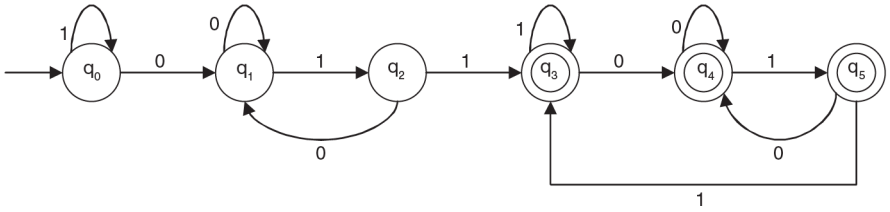
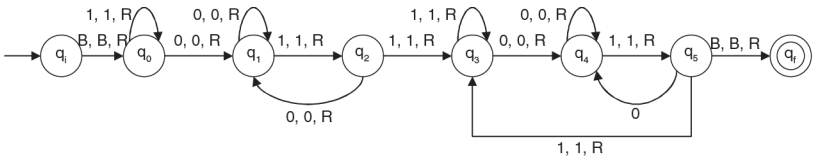
The NFA for accepting $(0,1)^*011(0,1)^*$ is



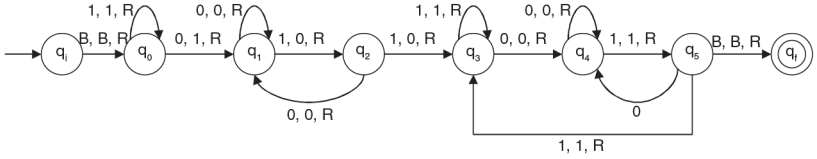
The equivalent DFA is



The converted TM is



The modified TM is (converting 011 by 100) as follows. The modifications are denoted in bold.



8.5 Two-stack PDA and Turing Machine

Two-stack PDA is already discussed in the pushdown automata chapter. It is also discussed that a PDA cannot be only two stack but may be more than two stacks $\left(a^n b^n c^n d^n, n \geq 0\right)$. In real, the language accepting power of a PDA increases by adding extra stacks. Here, the following question arises 'is two-stack (or more than two stacks) PDA as strong as the TM?'. In this section, we shall learn about a theorem proposed by an American artificial intelligence scientist Marvin Minsky called the Minsky theorem which answers this question.

8.5 .1 Minsky Theorem

Any language accepted by a two-stack PDA can also be accepted by some TM and vice versa.

8.5 .1 .1 General Minsky Model

A general Minsky model is shown in the following figure.

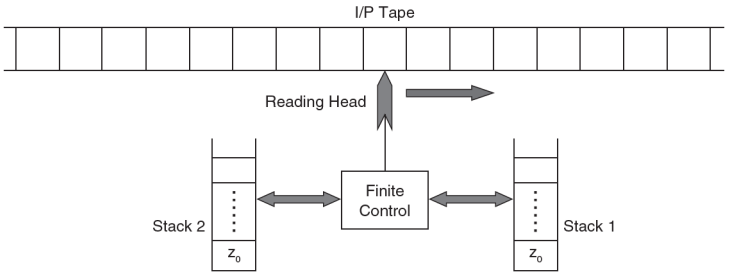


Fig. 8.4 *Two Stack PDA*

The model in Fig. 8.4 is a two-stack PDA containing an input tape, a reading head, and finite control as it was in PDA. The additionals are two stacks S_1 and S_2. Let the language be $a^n b^n c^n$ for $n \geq 1$. Scanning 'a', the symbols (say X) are put into stack S_1. With the stack top °X ' in S_1, b ' is scanned and the symbols (say Y) are put into stack S_2. At the time of traversing ' c ', the stack symbols must be X (for S_1) and Y (For S_2). These are popped for traversing 'c'.

A two-stack PDA is capable of accepting $\left(a^n b^n c^n d^n, n \geq 1\right)$, $\left(a^n b^n c^n d^n e^n, n \geq 1\right)$, and so on, with the additional stack. Accepting these types of languages is not possible for a single-stack PDA but possible for a TM. By this process, the two-stack PDA simulates a TM. The following example shows the acceptance of $\left(a^n b^n c^n d^n, n \geq 1\right)$ using the two-stack PDA.

Example **8.22** Design a two-stack PDA for $\left(a^n b^n c^n d^n, n \geq 1\right)$

Solution: Traversing 'a', X are put into stack S. Traversing 'b' with the stack top 'X' in S $_1$, X ' is popped fromS_1, and 'Y' is pushed into S_2. Traversing 'c' with the stack top 'Y' in S_2, 'Y' is popped from S_2, and Z ' is pushed into S_1. Traversing d′, Z ' is popped from S_1. The transitional functions are

$\delta\left(q_0, \lambda, z_1, z_2\right) \rightarrow \left(q_1, z_1, z_2\right)$ (z_1 and z_2 are stack bottom symbols of S_1 and S_2, respectively.)

$\delta\left(q_1, a, z_1, z_2\right) \rightarrow \left(q_1, Xz_1, z_2\right)$

$\delta\left(q_1, a, X, z_2\right) \rightarrow \left(q_1, XX, z_2\right)$

$\delta\left(q_1,\ b, X, z_2\right) \rightarrow \left(q_2, \lambda, Yz_2\right)$

$\delta\left(q_2,\ b, X, Y\right) \rightarrow \left(q_2, \lambda, YY\right)$

$\delta\left(q_2, c, z_1, Y\right) \rightarrow \left(q_3, Zz_1, \lambda\right)$

$\delta\left(q_3, c, Z, Y\right) \rightarrow \left(q_3, ZZ, \lambda\right)$

$\delta\left(q_3,\ d, Z, z_2\right) \rightarrow \left(q_4, \lambda, z_2\right)$

$\delta\left(q_4,\ d, Z, z_2\right) \rightarrow \left(q_4, \lambda, z_2\right)$

$\delta\left(q_4, \lambda, z_1, z_2\right) \rightarrow \left(q_f, z_1, z_2\right)$ // accepted by the final state

What We Have Learned So Far

1. The Turing machine was proposed by A.M. Turing in 1936 as a model of any possible combination. Any computational process carried out by the present day's computer can be done on the Turing machine.

2. The Turing machine is the machine format of unrestricted language, i.e., all types of languages are accepted by the Turing machine.

3. Undecidable problems are not solved by the computer as no Turing machine can be developed for these problems.

4. The mathematical description of the Turing machine consists of 7 touples $\big(Q, \Sigma, \Gamma, \delta, q_0,\ B,\ F\big)$ where Q is the finite set of states, Σ is the finite set of input alphabets, Γ is the finite set of allowable tape symbol, δ is the transitional function, q_0 is the initial state, B is the blank symbol, and F is the final state.

5. The transitional functions of the Turing machine are in the form $Q \times \Gamma \to (Q \times \Gamma \times \{L, R, H\})$. That is, from one state by getting one input from the input tape the machine moves to a state, writing a symbol on the tape, and moves to left, right, or halts.

6. The mechanical diagram of the Turing machine consists of the input tape, the finite control and the read-write head.

7. Upon execution of a transitional function, the machine goes to some state, writes a symbol in the cell of the input tape from where the input symbol was scanned, and moves the reading head to the left or right or halts.

8. The instantaneous description (ID) of a Turing machine remembers the contents of all cells from the rightmost to at least the leftmost, the cell currently being scanned by the read-write head, and the state of the machine at a given instance of time.

Solved Problems

1. Consider the Turing machine's description in the following table below. Draw the computation sequence of the input string 00 . [WBUT2008]

Present State	Tape Symbol :: b	Tape Symbol :: 0	Tape Symbol : 1
Q_1	$1Lq_2$	$0Rq_1$	–
Q_2	bRq_3	$0Lq_2$	$1Lq_2$
Q_3	–	bRq_4	bRq_5
Q_4	$0Rq_5$	$0Rq_4$	$1Rq_4$
Q_5	$0Lq_2$	–	–

Solution: $\big(Q_1,\ B\_00\ B\big) \to \big(Q_1, 00\ B\big) \to \big(Q_1, 00\ B\big) \to \big(Q_2, 001\big) \to \big(Q_2, \underline{0}01\big) \to \big(Q_2, \underline{B}001\big) \to \big(Q_3,\ B001\big) \to \big(Q_4,\ B01\big) \to^{-} \big(Q_4,\ B01\big) \to \overline{(Q_4},\ B01\ B\Big) \to^{-} \big(Q_5,\ B01\bar{0}\ B\big) \to \big(Q_2, \overline{B0}100\big) \to \big(Q_2,\ B0\underline{1}00\big) \to \big(Q_2,\ B\_100\big) \to \big(Q_2, \underline{B}0100\big) \to \big(Q_3,\ B\_100\big) \to \big(Q_4,\ B0\underline{1}00\big) \to \big(Q_4,\ B0100\big) \to \big(Q_4,\ B0\bar{1}00\big) \to \big(Q_4,\ B\bar{0}100\underline{B}\big) \to \big(Q_5, \overline{B}01000\underline{B}\big) \to \big(Q_2,\ B010000\big)$ (By this process, it will continue as there is no halt in the Turing Machine.)

2. Design a Turing Machine to test a string of balanced parenthesis. Show an ID for ()(()) Solution: There are two types of parentheses-open parentheses and close parentheses. Balanced parenthesis means that for each open parenthesis there is a close parenthesis. It may be nested like ((())) or may be non-nested like (()()) or ()()() or any other form. But for each open parenthesis, there is

a close parenthesis. [The definition that a balanced parenthesis means an equal number of open and close parentheses is wrong.

As an example,)( ) ( ) (is not a balanced parenthesis.] The string must start with an open parenthesis and after that open parenthesis or close parenthesis can appear, but for any position of the string, the number of open parenthesis is greater than or equal to the number of close parenthesis. The Turing Machine is designed as follows. Start from the leftmost symbol and traverse right to find the first close parenthesis. The transitional function is

$$\delta\left(q_0, ^{'}\ (^{\circ}\right) \rightarrow \left(q_0, (, \mathrm{R}\right)$$

Upon getting the first close parenthesis, replace it by 'X', change the state to q_1, and move left to find the open parenthesis for the replaced close parenthesis. The transitional function is

$$\delta\left(q_0, ^{\circ})^{\prime}\right) \rightarrow \left(q_1, \mathrm{X}, \mathrm{L}\right)$$

Getting the open parenthesis, replace it by 'X' and change the state to q_0. The transitional function is

$$\delta\left(q_1, ^{\circ}\ (^{\circ}\right) \rightarrow \left(q_0, \mathrm{X}, \mathrm{R}\right)$$

Then, traverse towards the right to find the close parenthesis. Here, the machine may have to traverse X, which is the replaced symbol for close parenthesis. The transitional function for traversing this is

$$\delta\left(q_0, \mathrm{X}^{\circ}\right) \rightarrow \left(q_0, \mathrm{X}, \mathrm{R}\right)$$

For a nested parenthesis like (()), the machine has to traverse X, which is the replaced symbol of the open parenthesis at the time of finding the open parenthesis. The transitional function for traversing this is

$$\delta\left(q_1, ^{\circ}\mathrm{X}^{\circ}\right) \rightarrow \left(q_1, \mathrm{X}, \mathrm{L}\right)$$

Traversing right, if B appears as an input, then there is no parenthesis (open or close) left at the right side. Then, traverse left to find if any parenthesis is left at the left side or not. The transitional function is

$$\delta\left(q_0,\ \mathrm{B}^{\mathrm{c}}\right) \rightarrow \left(q_2,\ \mathrm{B},\ \mathrm{L}\right)$$

At the time of left traversing, the machine has to traverse X by the following transitional function

$$\delta\left(q_2, \mathrm{X}^{\circ}\right) \rightarrow \left(q_2,\ \mathrm{B},\ \mathrm{L}\right)$$

At the left hand side if it gets B, the machine halts.

$$\delta\left(q_2, ^{\circ}\mathrm{B}^{\circ}\right) \rightarrow \left(q_3,\ \mathrm{B}, \mathrm{H}\right)$$