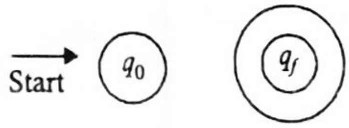
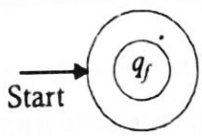
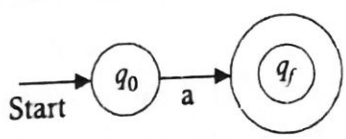
2.8 REGULAR SETS AND REGULAR EXPRESSIONS

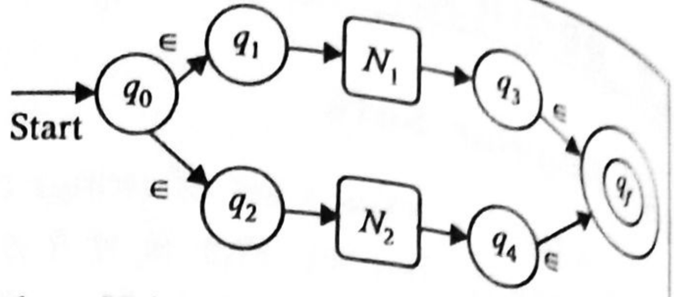
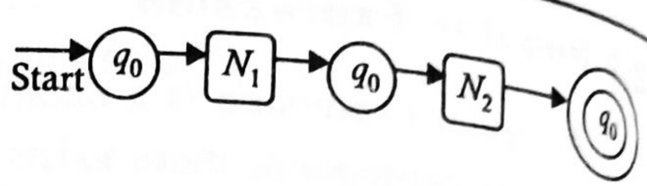
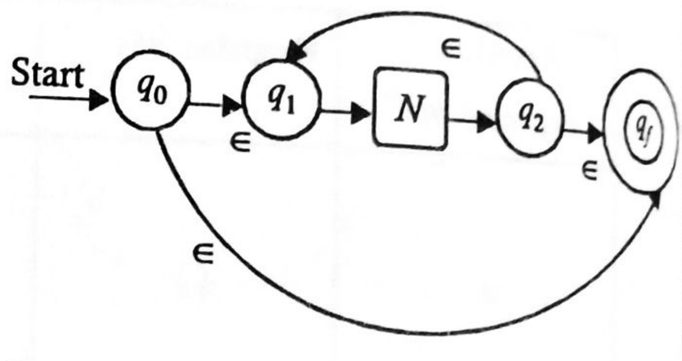
2.8.1 Regular Sets

A regular set is a set of strings for which there exists some finite automata accepting that set. That is, if R is a regular set, then $R = L(M)$ for some finite automata M. Similarly, if M is a finite automata, then $L(M)$ is always a regular set.

2.8.2 Regular Expression

A regular expression is a notation to specify a regular set. Hence, for every regular expression, there exists a finite automata that accepts the language specified by the regular expression. Similarly, for every finite automata M, there exists a regular expression notation specifying $L(M)$. Regular expressions and the regular sets they specify are shown in the following table.

Regular expression	Regular Set	Finite automata
ϕ	{ }	
$\in$	{ $\in$ }	
Every a in Σ is a regular expression	$\{a\}$	

$r_1 + r_2$ or $r_1 \mid r_2$ is a regular expression,	$R_1 \cup R_2$ (Where R_1 and R_2 are regular sets corresponding to r_1 and r_2, respectively)	 where N_1 is a finite automata accepting R_1, and N_2 is a finite automata accepting R_2
$r_1 . r_2$ is a regular expression,	$R_1.R_2$ (Where R_1 and R_2 are regular sets corresponding to r_1 and r_2, respectively)	 where N_1 is a finite automata accepting R_1, and N_2 is finite automata accepting R_2
r^* is a regular expression,	R^* (where R is a regular set corresponding to r)	 where N is a finite automata accepting R.

Hence, we only have three regular-expression operators: | or + to denote union operations,. for concatenation operations, and * for closure operations. The precedence of the operators in the decreasing order is: *, followed by., followed by | . For example, consider the following regular expression:

$$a.\ (a + b)^*.\ b.b$$

To construct a finite automata for this regular expression, we proceed as follows: the basic regular expressions involved are a and b, and we start with automata for a and automata for b. Since brackets are evaluated first, we initially construct the automata for $a + b$ using the automata for a and the automata for b, as shown in Figure 2.25.

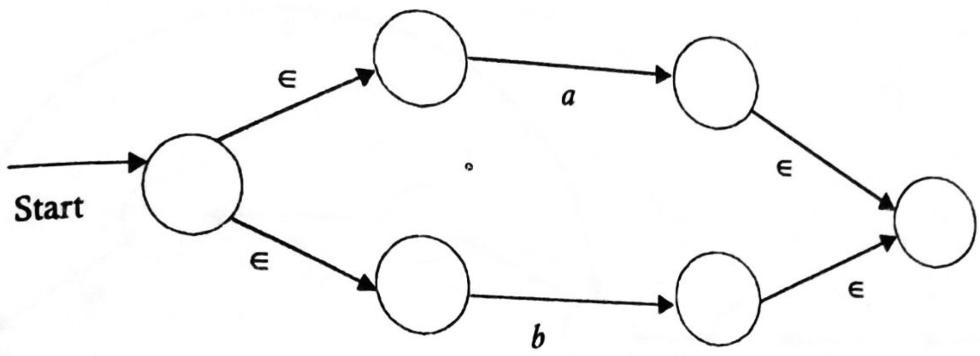


FIGURE 2.25 Transition diagram for $(a + b)$.

Since closure is required next, we construct the automata for $(a + b)^*$, using the automata for $a + b$, as shown in Figure 2.26.

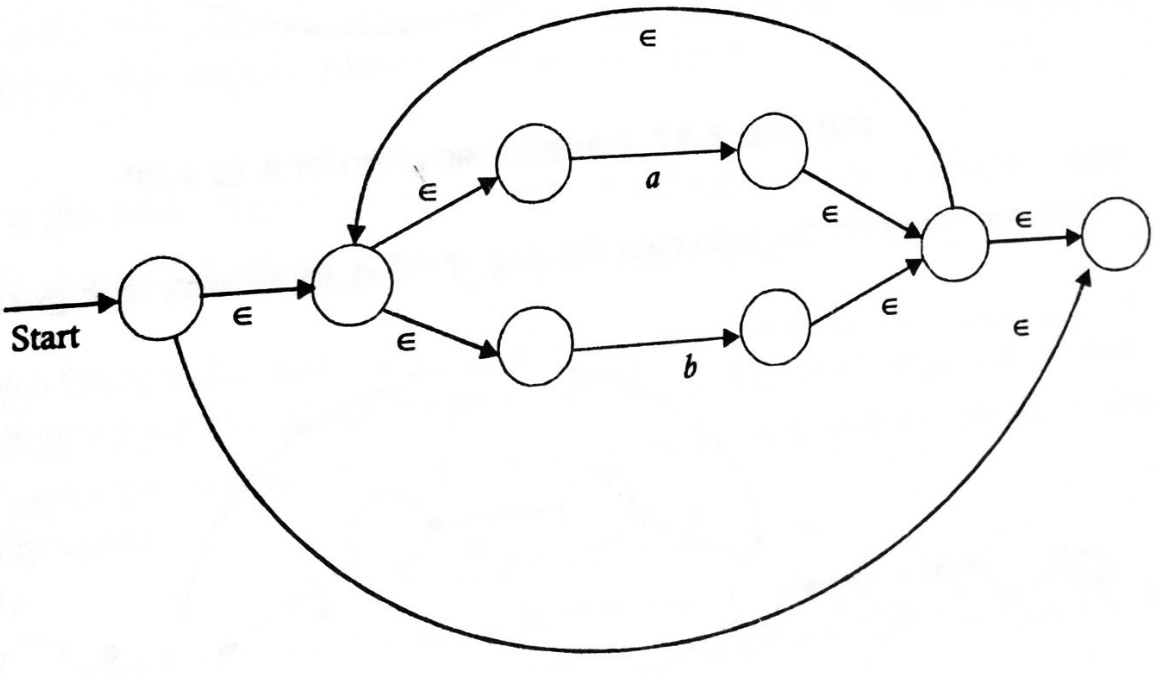


FIGURE 2.26 Transition diagram for $(a + b)^*$.

The next step is concatenation. We construct the automata for $a. (a + b)^*$ using the automata for $(a + b)^*$ and a, as shown in Figure 2.27.

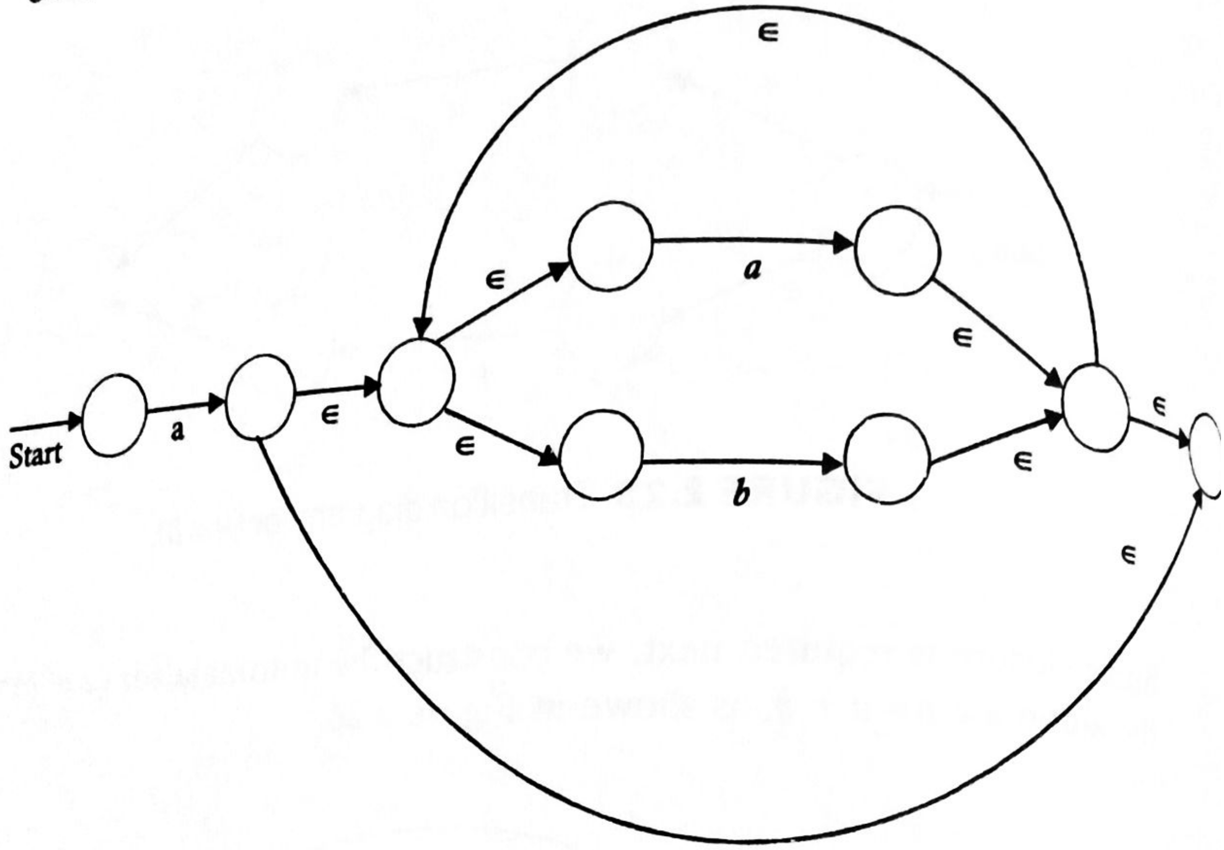


FIGURE 2.27 Transition diagram for $a.\ (a + b)^*$.

Next we construct the automata for $a.(a + b)^*.b$, as shown in Figure 2.28.

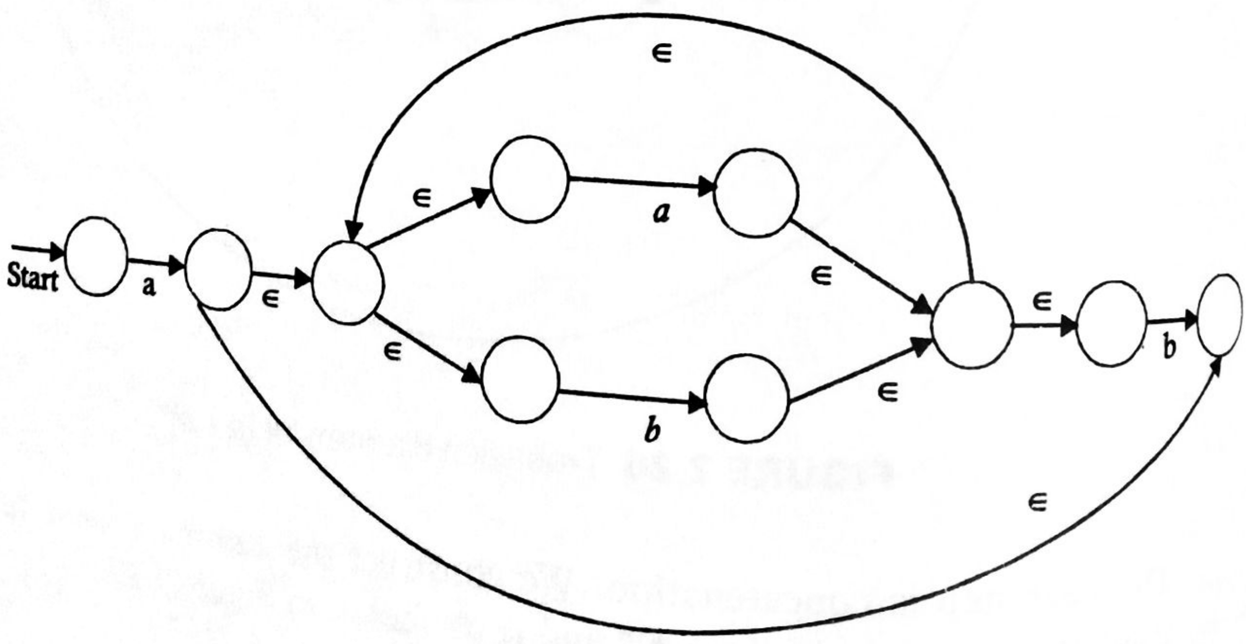


FIGURE 2.28 Automata for $a.(a + b)^*\ .b$.

Finally, we construct the automata for $a.(a + b)^*.b.b$ (Figure 2.29).

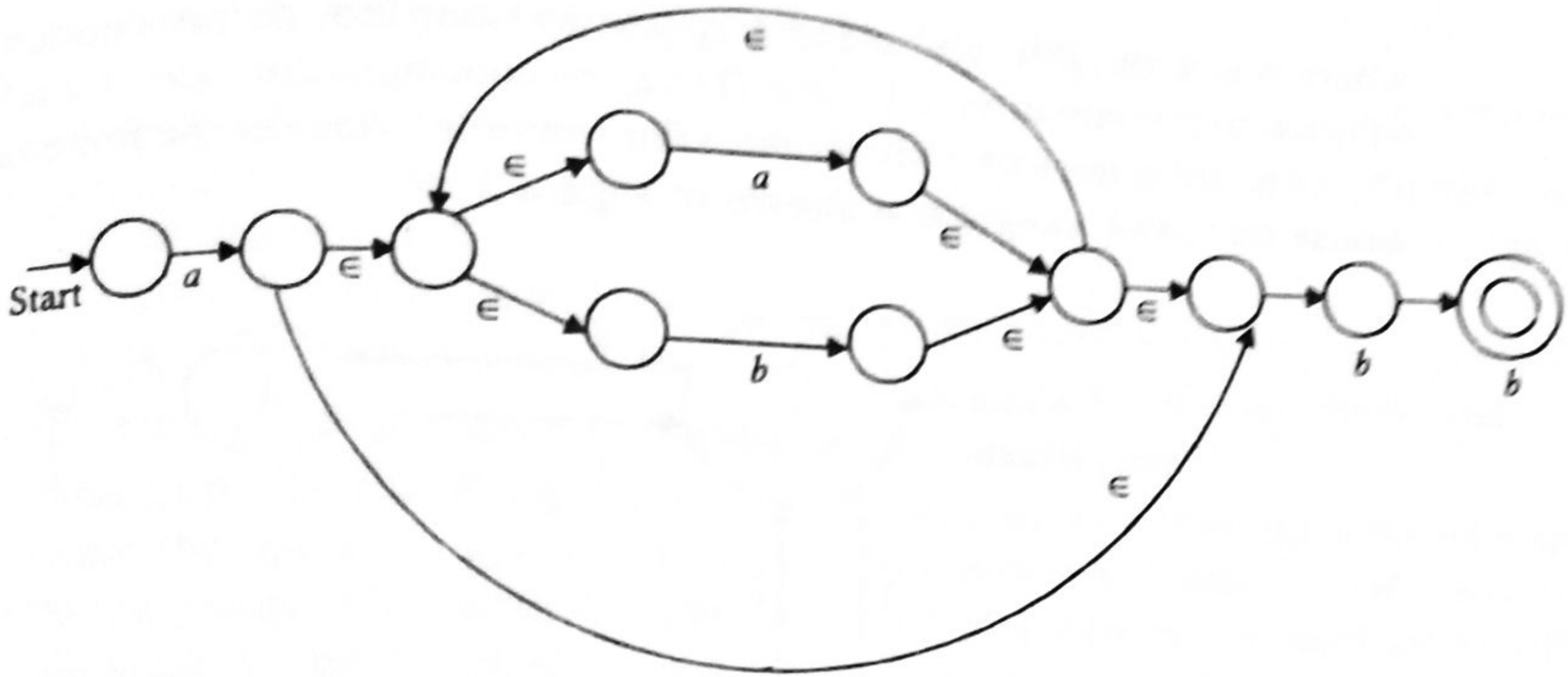


FIGURE 2.29 Automata for $a.(a + b)^*.b.b.$

This is an NFA with ∈-moves, but an algorithm exists to transform the NFA to a DFA. So, we can obtain a DFA from this NFA.