4.10 Regular Language Recognition

**Theorem 4.10.1** The languages \( L(G) \) and \( L(G) \cup \{ \epsilon \} \) generated by regular grammars \( G \) and recognized by finite-state machines are the same.

**Proof** Given a regular grammar \( G \), we construct a corresponding NFSM \( M \) that accepts exactly the strings generated by \( G \). Similarly, given a DFSM \( M \) we construct a regular grammar \( G \) that generates the strings recognized by \( M \).

From a regular grammar \( G = (N, T, R, S) \) with rules \( R \) of the form \( A \rightarrow a \) and \( A \rightarrow bC \) we create a grammar \( G' \) generating the same language by replacing a rule \( A \rightarrow a \) with rules \( A \rightarrow aB \) and \( B \rightarrow \epsilon \) where \( B \) is a new non-terminal unique to \( A \rightarrow a \). Thus, every derivation \( S \Rightarrow^*_G w, w \in T^* \), now corresponds to a derivation \( S \Rightarrow^*_{G'} wB \) where \( B \rightarrow \epsilon \). Hence, the strings generated by \( G \) and \( G' \) are the same.

Now construct an NFSM \( M_{G'} \) whose states correspond to the non-terminals of this new regular grammar and whose input alphabet is its set of terminals. Let the start state of \( M_{G'} \) be labeled \( S \). Let there be a transition from state \( A \) to state \( B \) on input \( a \) if there is a rule \( A \rightarrow aB \) in \( G' \). If state \( q \) is a final state of \( M \), add the rule \( q \rightarrow \epsilon \). Thus, if a string \( w \) is accepted by \( M_{G'} \), given the one-to-one correspondence between edges and rules, there is a derivation of \( w \) from \( S \) in \( G' \). Thus, the strings generated by \( G \) and the strings accepted by \( M_{G'} \) are the same.

Now assume we are given a DFSM \( M \) that accepts a language \( L_M \). Create a grammar \( G_M \) whose non-terminals are the states of \( M \) and whose start symbol is the start state of \( M \). \( G_M \) has a rule of the form \( q_1 \rightarrow aq_2 \) if \( M \) makes a transition from state \( q_1 \) to \( q_2 \) on input \( a \). If state \( q \) is a final state of \( M \), add the rule \( q \rightarrow \epsilon \). If a string is accepted by \( M \), that is, it causes \( M \) to move to a final state, then \( G_M \) generates the same string. Since \( G_M \) generates only strings of this kind, the language accepted by \( M \) is \( L(G_M) \). Now convert \( G_M \) to a regular grammar \( \bar{G}_M \) by replacing each pair of rules \( q_1 \rightarrow aq_2, q_2 \rightarrow \epsilon \) by the pair \( q_1 \rightarrow aq_2, q_1 \rightarrow a \), deleting all rules \( q \rightarrow \epsilon \) corresponding to unreachable final states \( q \), and deleting the rule \( S \rightarrow \epsilon \) if \( \epsilon \in L_M \). Then, \( L_M - \{ \epsilon \} = L(G_M) - \{ \epsilon \} = L(\bar{G}_M) \).

![Figure 4.27](image-url) A nondeterministic FSM that accepts a language generated by a regular language in which all rules are of the form \( A \rightarrow bC \) or \( A \rightarrow \epsilon \). A state is associated with each non-terminal, the start symbol \( S \) is associated with the start state, and final states are associated with non-terminals \( A \) such that \( A \rightarrow \epsilon \). This particular NFSM accepts the language \( L(G_4) \) of Example 4.9.4.