In addition to section 4.3, we consider here a different method to find an equivalent regular expression for a finite automaton. This method is still based on the $L(p,q,k)$ construction of the proof of Theorem 4.5, but is easier to apply since the amount of bookkeeping is reduced.

At the intermediate stages of the construction we will be dealing with generalized finite automata which may have arrows labelled by regular expressions. A regular expression labelling an arrow from state $p$ to state $q$ represents a set of input strings which may lead the automaton from $p$ to $q$. Observe that every symbol from the input alphabet as well as $\Lambda$ can be considered as a regular expression. Consequently, all FA, NFA, and NFA-$\Lambda$ can be seen as generalized finite automata!

Any finite automaton or NFA or NFA-$\Lambda$ can be modified in such a way that the resulting NFA-$\Lambda$ has an initial state without incoming arcs and a single final (accepting) state, and no outgoing arcs from that accepting state, and still accepts the same language (see Exercise 4.26).

Thus from here on, we can assume that the input to our construction is an NFA-$\Lambda$ $M = (Q, \Sigma, q_0, \{q_f\}, \delta)$ with no incoming arcs for its initial state $q_0$, with a single final state $q_f$, and no arrows leading out of $q_f$.

We describe a method to eliminate states from $M$ and change the labels of the arrows in such a way that in the end only $q_0$ and $q_f$ remain. Moreover, if $L(M) \neq \emptyset$, then there will be one arc left arc left, leading from $q_0$ to $q_f$ and labelled with a regular expression $r$ such that $L(M) = ||r||$ the regular language defined by $r$.

\[
\text{NFA-$\Lambda$ } M \quad \Rightarrow \quad ||r|| = L(M)
\]
First we replace all parallel edges as follows:

\[ p \xrightarrow{r_1} q \xrightarrow{r_0} r \quad \Rightarrow \quad p \xrightarrow{r_1 + r_2} q \]

Next, if \( q \) is a state such that \( q \neq q_0 \) and \( q \neq q_f \), then for all states \( p \) and \( s \) such that \( p \neq q \) and \( s \neq q \) (we allow \( p = s \)), and such that there is an arrow from \( p \) to \( q \) and one from \( q \) to \( s \), the automaton is modified as indicated below (we distinguish 4 cases), after which \( q \) is deleted.

1) \[ p \xrightarrow{r_0} s \quad \Rightarrow \quad p \xrightarrow{r_0 + r_1 r_2 r_3} s \]

2) \[ p \xrightarrow{r_0} s \quad \Rightarrow \quad p \xrightarrow{r_1 r_2 r_3} s \]

3) \[ p \xrightarrow{r_0} s \quad \Rightarrow \quad p \xrightarrow{r_0 + r_1 r_3} s \]

4) \[ p \xrightarrow{r_0} s \quad \Rightarrow \quad p \xrightarrow{r_1 r_3} s \]

Observe that the new regular expression labelling the arrow from \( p \) to \( s \) describes the same regular language as the original regular expression leading
directly from \( p \) to \( s \) combined (+) with the regular expression leading from \( p \) to \( s \) via \( q \).

Note furthermore that the cases 2, 3, 4 are each a special case of 1 (with \( r_0 = \emptyset \) and/or \( r_2 = \emptyset \)).

In this way we can systematically remove all states different from \( q_0 \) and from \( q_f \). If at this point, there is no arrow leading from \( q_0 \) to \( q_f \) then in \( M \) already \( q_f \) is not reachable from \( q_0 \) and \( L(M) = \emptyset \). Otherwise, there is an arrow leading from \( q_0 \) to \( q_f \) labelled with a regular expression \( r \) and we have that \( ||r|| = L(M) \), because the regular expressions introduced describe the words corresponding to the original path labels.

Finally, we note that the order in which states are removed is not fixed, but may be arbitrarily chosen. In general, different orders lead to different (but equivalent) expressions. Some orders may be more efficient than others.

**EXAMPLE 4.10**

As an example, we apply the method to the FA \( M \) of Example 4.10 (see Figure 4.18). First we bring \( M \) in the desired form with initial state \( q_i \) without incoming arcs and the single final state \( q_f \) with no outgoing arcs.

Next, we remove state \( q_3 \). Observe that the pairs \( q_2, q_1 \) and \( q_2, q_3 \) are the only pairs of states of which the first one has an outgoing arrow leading to \( q_3 \) and the second one has an incoming arrow from \( q_3 \).
Then we remove state $q_2$. We only have to consider the pairs $q_1, q_1$ and $q_1, q_f$.

\[
\begin{array}{c}
\sim q_0 \\
\Lambda \\
\Lambda + b(bb)^* \Lambda \\
2 + b(bb)^*(a + ba)
\end{array}
\quad \xrightarrow{q_f}
\]

Finally we remove state $q_1$; only $q_0$ and $q_f$ need to be considered.

\[
\sim q_0 \Lambda (a + b(bb)^*(a + ba))^* (\Lambda + b(bb)^* \Lambda) \xrightarrow{q_f}
\]

We are done, a regular expression for $L(M)$ is as indicated. It is easy to see that this expression can immediately be simplified to the equivalent: $(a + b(bb)^*(a + ba))^* (\Lambda + b(bb)^*)$.

By the way, we have chosen first $q_3$, then $q_2$, and finally $q_1$, because in that way we minimized at each step the number of pairs to be considered (see for yourself what happens when the states are removed in the order $q_1$, $q_2$, $q_3$).

\[\text{versie 10 oktober 2005}\]