1. [6 points] Assume the NFA $A$ whose transition function is graphically represented below.

Consider the algorithm for transforming a FA into a regular expression, based on state elimination. Apply the following steps in the given order:

(a) eliminate state $q_1$ from $A$, and display the resulting automaton $A'$;
(b) eliminate state $q_3$ from $A'$, and display the resulting automaton $A''$;
(c) convert $A''$ into the equivalent regular expression $E_{q_2}$.

If you simplify any of the resulting regular expressions, add some discussion.

**Solution** Recall that, for every regular expression $R$, we have $\emptyset + R = R$, $\emptyset R = R \emptyset = \emptyset$, and $\epsilon R = R \epsilon = R$. We use these simplifications several times below.

(a) After the elimination of $q_1$ from $A$ we obtain the automaton $A'$, graphically represented as
(b) After the elimination of \( q_3 \) from \( A' \) we obtain the automaton \( A'' \), graphically represented as

![Automaton Graph]

(c) The automaton \( A'' \) has two states, with the initial and the final states representing distinct states. We then need to apply the expression \( E_q = (R + SU^*T)^*SU^* \).

Considering that in our case we have

\[
R = 11^*1 \\
S = 0 + (0 + 11^*0)0^*1 \\
U = 00^*1 \\
T = \emptyset
\]

we obtain the regular expression

\[
E_{q_2} = (11^*1 + (0 + (0 + 11^*0)0^*1)(00^*1)^*\emptyset)(0 + (0 + 11^*0)0^*1)(00^*1)^* \\
= (11^*1 + \emptyset)^*(0 + (0 + 11^*0)0^*1)(00^*1)^* \\
= (11^*1)^*(0 + (0 + 11^*0)0^*1)(00^*1)^*.
\]

2. [9 points] Consider the following languages, defined over the alphabet \( \Sigma = \{a, b\} \):

\[
L_1 = \{ba^mba^n b \mid m, n \geq 1, m < n\} \\
L_2 = \{ba^mb^n a \mid m, n \geq 1, m < n\} \\
L_3 = L_2L_1
\]

For each of the above languages, state whether it belongs to \( \text{REG} \), to \( \text{CFL} \setminus \text{REG} \), or else whether it is outside of \( \text{CFL} \). Provide a mathematical proof for all of your answers.

**Solution**

(a) \( L_1 \) belongs to the class \( \text{CFL} \setminus \text{REG} \).

We first show that \( L_1 \) is not a regular language, by applying the pumping lemma for this class. Let \( N \) be the pumping lemma constant for \( L_1 \). We choose the string \( w = ba^Nb^an+1b \in L_1 \) with \( |w| \geq N \), and consider all possible factorizations \( w = xyz \) satisfying the conditions \( |y| \geq 1 \) and \( |xy| \leq N \). We distinguish two cases.
Case 1: $y$ spans the leftmost occurrence of $b$ in $w$, and possibly more symbols from $w$. This means that $x = \epsilon$. We then choose $k = 0$ and obtain the string $w_0 = xy^0z = z$ which has fewer than 3 occurrences of symbol $b$, and therefore $w_0 \notin L_1$.

Case 2: $y$ does not span the leftmost occurrence of $b$ in $w$. Because of the condition $|xy| \leq N$, we have that $y$ can only contain occurrences of symbol $a$, with these occurrences placed to the left of the second occurrence of symbol $b$ in $w$. In this case, we choose $k = 2$ and obtain the string $w_2 = xy^2z$ which has the form $ba^{N+|y|}ba^{N+1}b$. Because of the condition $|y| \geq 1$, we have that $N + |y| \geq N + 1$, and therefore $w_2 \notin L_1$.

Since we have considered all possible factorizations for string $w$, we must conclude that $L_1$ is not a regular language.

As a second part of the answer, we need to show that $L_1$ belongs to the class CFL. Consider the CFG $G_1$ with productions:

$$S \to bAb$$
$$A \to aAa \mid aBa$$
$$B \to Ba \mid ba$$

It is not difficult to see that $L(G_1) = L_1$.

(b) $L_2$ belongs to the class REG.

To see this, we observe that we can rewrite the definition of this language as $L_2 = \{ba^n b \mid n \geq 3\}$.

It is then easy to see that the regular expression $R = baaaa^*b$ generates $L_2$.

(c) $L_3$ belongs to the class CFL\REG.

The easy part here is to show that $L_3$ is in CFL. We have already seen that $L_2$ is in REG and therefore in CFL, and we have already shown that $L_1$ is in CFL. Since $L_3 = L_2L_1$, and since the class CFL is closed under concatenation, we conclude that $L_3$ is in CFL.

We now prove that $L_3$ is not a regular language, again by applying the pumping lemma for this class. Let $N$ be the pumping lemma constant for $L_3$. We choose the string $w = ba^3bba^Nba^{N+1}b \in L_3$ with $|w| \geq N$, and consider all possible factorizations $w = xyz$ satisfying the conditions $|y| \geq 1$ and $|xy| \leq N$. We observe that string $w$ has three runs of symbols $a$: the first of length 3, the second of length $N$, and the third of length $N + 1$. We call these three runs block 1, block 2, and block 3, respectively. We distinguish three cases.

Case 1: $y$ spans at least one occurrence of $b$ from $w$. We then choose $k = 0$ and obtain the string $w_0 = xy^0z = xz$ which has fewer than 5 occurrences of symbol $b$, and therefore $w_0 \notin L_3$.

Case 2: $y$ spans zero occurrence of $b$ and a few occurrences of symbol $a$ from block 1 only. We choose $k = 0$ and obtain the string $w_0 = xy^0z = xz$ which has the form $ba^{3-|y|}ba^Nba^{N+1}b$. Because of the condition $|y| \geq 1$, we have $3 - |y| < 3$, and therefore $w_0 \notin L_3$.

Case 3: $y$ spans zero occurrence of $b$ and a few occurrences of symbol $a$ from block 2 only. We choose $k = 2$ and obtain the string $w_2 = xy^2z$ which has the form $ba^3bba^{N+|y|}ba^{N+1}b$. Because of the condition $|y| \geq 1$, we have that $N + |y| \geq N + 1$, and therefore $w_2 \notin L_3$.

Since we have considered all possible factorizations for string $w$, we must conclude that $L_3$ is not a regular language.

We observe that the above proof showing that $L_3$ is not in REG is a little bit involved. There is an alternative, simpler way of proving that $L_3$ is not a regular language. Assume by now
that $L_3$ is a regular language. From known properties of regular languages, it follows that $L_3^R$ is also a regular language, where $R$ is the string reversal operator, extended to languages as usual. Observing that we have $L_3^R = L_1^R L_2^R$, the language $L_3^R$ can be rewritten as

$$L_3^R = \{ba^m ba^n bba^p b \mid m, n \geq 1, m > n, p \geq 3\}$$

We can now apply the pumping lemma to $L_3^R$, resulting in a proof that is very similar to the proof for $L_1$, consisting only of two cases. We then find that $L_3^R$ is not a regular language, and we must therefore conclude that $L_3$ cannot be regular as well.

3. [6 points] With reference to the membership problem for context-free languages, answer the following two questions.

(a) Specify the dynamic programming algorithm reported in the textbook for the solution of this problem.

(b) Consider the CFG $G$ in Chomsky normal form defined by the following rules:

- $S \rightarrow CD$
- $C \rightarrow AC' \mid c$
- $C' \rightarrow CB$
- $A \rightarrow a$
- $B \rightarrow b$
- $D \rightarrow DD \mid d$

Assuming as input the CFG $G$ and the string $w = aacbbdddd$, trace the application of the algorithm in (a).

Solution

(a) The required dynamic programming algorithm is reported in Section 7.4.4 of the textbook.

(b) On input $w$ and $G$, the algorithm constructs the table reported below.
4. [5 points] Assess whether the following statements are true or false. Provide motivations for all of your answers.

(a) Let $L_1, L_3$ be in REG (the class of regular languages) and let $L_2$ be in CFL. Then the language $L_1L_2L_3$ is always in REG.

(b) Let $L_1, L_3$ be in REG and let $L_2$ be in CFL. Then the language $L_1L_2L_3$ is always in CFL.

(c) The class RE defined over the alphabet $\Sigma = \{0, 1\}$ is closed under complementation.

(d) The class $\mathcal{P}$ of languages over the alphabet $\Sigma = \{0, 1\}$ that can be recognized in polynomial time by a TM is closed under complementation.

Solution

(a) False. Consider as a counterexample the regular languages $L_1 = L_3 = \{\epsilon\}$ and the context-free language $L_2 = \{a^n b^n \mid n \geq 1\}$. Observe that $L_1L_2L_3 = L_2$, and we know that $L_2$ is not a regular language.

(b) True. We know that a language in REG is also a language in CFL. We also know that the class CFL is closed under concatenation. Therefore $L' = L_1L_2$ must be in CFL, and $L'L_3 = L_1L_2L_3$ must be in CFL.

(c) False. As a counterexample consider the language $L_{ne}$ in RE, defined in the textbook. Consider also the language $L_e$, which is the complement of $L_{ne}$ with respect to $\Sigma^*$. We now that $L_e$ is not in RE.

(d) True. Consider an arbitrary language $L \in \mathcal{P}$. By the definition of the class $\mathcal{P}$, there exists a TM $M$ such that $L(M) = L$, and $M$ stops after a polynomial number of steps in the size of its input $w$. We can then construct a TM $M'$ that, given as input a string $w$, simulates $M$ on $w$. When the simulation stops in a state $q$, that is, when there is no next move for $M$, $M'$ moves to a final state if $q$ is not a final state for $M$, and $M'$ moves to a non-final state if $q$ is a final state.
for $M$. It is easy to see that $L(M') = \overline{L}$ and that $M'$ runs in polynomial time. We therefore conclude that $P$ is closed under complementation.

5. **[7 points]** Let $R$ be the string reversal operator, extended to languages as usual. Consider the following property of the RE languages defined over the alphabet $\Sigma = \{0, 1\}$

\[
P = \{L \mid L \in \text{RE}, L \cap L^R = \emptyset\}
\]

where the condition $L \cap L^R = \emptyset$ means that for every string $w \in L$, $w^R$ does not belong to $L$. Define $L_P = \{\text{enc}(M) \mid L(M) \in P\}$.

(a) Use Rice’s theorem to show that $L_P$ is not in REC.

(b) State whether $L_P$ is in $\text{RE} \setminus \text{REC}$ or else outside of RE.

**Solution**

(a) We have to show that property $P$ is not trivial.

- $P \neq \emptyset$. Consider the language $L = \{1100\}$. Since $L$ is finite, $L$ is also in RE. Observe that $L^R = \{0011\}$ and $L \cap L^R = \emptyset$. Therefore $L \in P$.

- $P \neq \text{RE}$. Consider the language $L = \{1100, 0011\}$. Since $L$ is finite, $L$ is also in RE. Observe that $L \cap L^R = L \neq \emptyset$, and therefore $L \notin P$.

(b) We now show that $L_P$ is not in RE. The most convenient way to do this is to consider the complement language $L_P^c = L_{\overline{P}}$, where $\overline{P}$ is the complement of class $P$ with respect to RE and can be specified as

\[
\overline{P} = \{L \mid L \in \text{RE}, L \cap L^R \neq \emptyset\}
\]

We specify a nondeterministic TM $N$ such that $L(N) = L_{\overline{P}}$. Since every nondeterministic TM can be converted into a standard TM, this shows that $L_{\overline{P}}$ is in RE. Our nondeterministic TM $N$ takes as input the encoding of a TM $M$ and performs the following steps.

- $N$ nondeterministically guesses a string $w \in \Sigma^*$ and checks that $w \in L(M)$ and $w^R \in L(M)$ are both satisfied.

- If the previous step terminates and is successful, $N$ ends the computation in a final state. In all other cases, $N$ ends the computation in a non-final state or runs for ever.

It is not difficult to see that $L(N) = L_{\overline{P}}$.

Since $L_{\overline{P}}$ is in RE, if its complement language $L_P$ were in RE as well, then we would conclude that both languages are in REC, from a theorem in Chapter 9 of the textbook. But we have already shown in (a) that $L_P$ is not in REC. We must therefore conclude that $L_P$ is not in RE.