Problem 1. [10 points] Let \( \Sigma = \{ [a, b] : a, b \in \{0, 1\} \} \).

A string \( w \in \Sigma^* \) can then be understood as encoding two natural numbers in binary notation. More precisely, if

\[
w = \prod_{i=n-1}^{0} [a_i b_i] = [a_{n-1} b_{n-1}] \ldots [a_0 b_0],
\]

then the numbers encoded are

\[
\sum_{i=0}^{n-1} 2^i a_i \quad \text{and} \quad \sum_{i=0}^{n-1} 2^i b_i.
\]

Consider the language

\[
\text{LESS} = \left\{ \prod_{i=n-1}^{0} [a_i b_i] : \sum_{i=0}^{n-1} 2^i a_i < \sum_{i=0}^{n-1} 2^i b_i \right\}.
\]

Draw a diagram describing a DFA that recognizes LESS.

Solution:
Problem 2. [30 points] Let $\Sigma$ be an alphabet. A proper prefix of a string $w \in \Sigma^*$ is a string $x \in \Sigma^*$ for which there exists a non-empty string $y \in \Sigma^* \setminus \{\varepsilon\}$ such that $xy = w$. Similarly, a proper suffix of a string $w \in \Sigma^*$ is a string $y \in \Sigma^*$ for which there is a non-empty string $x \in \Sigma^* \setminus \{\varepsilon\}$ such that $xy = w$.

Given an arbitrary DFA $M = (Q, \Sigma, \delta, q_0, F)$, construct a DFA $M' = (Q', \Sigma, \delta', q'_0, F')$ by formally defining $Q'$, $\delta'$, $q'_0$ and $F'$ in such a way that

$$L(M') = \{w \in L(M) : \text{no proper prefix of } w \text{ is in } L(M)\}.$$ 

Construct also a DFA $M'' = (Q'', \Sigma, \delta'', q''_0, F'')$ by formally defining $Q''$, $\delta''$, $q''_0$ and $F''$ in such a way that

$$L(M'') = \{w \in L(M) : w \text{ is not a proper prefix of any string in } L(M)\}.$$ 

Let $A$ be a regular language. Is it always true that $\{w \in A : \text{no proper suffix of } w \text{ is in } A\}$ is regular? Prove that your answer is correct.

Solution: To define $M'$, let $Q' = Q \cup \{\bot\}$ (where $\bot \notin Q$), $q'_0 = q_0$, $F' = F$ and

$$\delta' : Q' \times \Sigma \to Q'$$

$$(q, a) \mapsto \begin{cases} \bot, & q \in F \cup \{\bot\} \\ \delta(q, a), & \text{otherwise.} \end{cases}$$

To define $M''$, let $Q'' = Q$, $q''_0 = q_0$, $\delta'' = \delta$ and

$$F'' = \{f \in F : \exists w \in \Sigma^* \setminus \{\varepsilon\} : \delta(f, w) \in F\}.$$ 

We will show that the language is indeed regular whenever $A$ is regular. First, however, note that whenever $x \in \Sigma^*$ is a proper suffix of $w \in \Sigma^* \setminus \{\varepsilon\}$, $x^\top$ is a proper prefix of $w^\top$ and vice-versa (recall Assignment 2 Problem 1). Therefore, we have

$$\{w \in A : \text{no proper suffix of } w \text{ is in } A\} =$$

$$\{w \in A : \exists x \in A : x \text{ is a proper suffix of } w\} =$$

$$\{w \in A : \exists x \in A : x^\top \text{ is a proper prefix of } w^\top\} =$$

$$\{w \in A^\top : \exists x \in A^\top : x \text{ is a proper prefix of } w\} =$$

$$\{w \in A^\top : \text{no proper prefix of } w \text{ is in } A^\top\},$$ 

which, from the first item and our knowledge that $A^\top$ is regular, is a regular language.
Problem 3. [20 points] Let \( A \) be a language over an alphabet \( \Sigma \). Define \( A_+ = \{ xay : x, y \in \Sigma^*, a \in \Sigma \text{ and } xy \in A \} \), i.e., \( A_+ \) is the language of strings that can be obtained by inserting a character into a string in \( A \). Is \( A_+ \) guaranteed to be regular whenever \( A \) is regular? Prove that your answer is correct.

Less detailed solution, part A: Let \( A \) and \( B \) be two languages. As to insert a character in a string from \( A \cup B \) we can either insert a character in a string from \( A \) or in a string from \( B \), we have \( (A \cup B)_+ = A_+ \cup B_+ \). Similarly, to insert a character in a string from \( AB \), we can put the character in the part from \( A \) or in the part from \( B \), so we have \( (AB)_+ = A_+B \cup AB_+ \). Finally, to insert a character in a string from \( A^* \) we need to insert it in one of the parts from \( A \), so \( (A^*)_+ = A^*_A_+A^* \).

More detailed solution, part A: Let \( A \) and \( B \) be languages. We have

\[
(A \cup B)_+ = \{ xay : x, y \in \Sigma^*, a \in \Sigma, xy \in A \cup B \} = \{ xay : x, y \in \Sigma^*, a \in \Sigma, xy \in A \} \cup \{ xay : x, y \in \Sigma^*, a \in \Sigma, xy \in B \} = A_+ \cup B_+.
\]

Next note that whenever \( x, y \in \Sigma^* \) and \( xy \in AB \), there is a prefix of \( x \) in \( A \) or a suffix of \( y \) in \( B \), so

\[
(AB)_+ = \{ xay : x, y \in \Sigma^*, a \in \Sigma, xy \in AB \} = \{ wxay : w \in A, x, y \in \Sigma^*, a \in \Sigma, xy \in B \} \cup \{ xayw : w \in B, x, y \in \Sigma^*, a \in \Sigma, xy \in A \} = AB_+ \cup A_+B.
\]

Let us now show that \( (A^*)_+ = A^*_A_+A^* \). First, take a string \( w \in A^*_A_+A^* \), which, by definition, can be written as \( w_0 \cdots w_{m-1}xayw_0' \cdots w_{n-1}' = w_0, \ldots, w_{m-1}, w_0', \ldots, w_{n-1}' \in A \setminus \{ \varepsilon \} \), \( x, y \in \Sigma^* \), \( a \in \Sigma \) and \( xy \in A \). Then \( w_0 \cdots w_{m-1}xayw_0' \cdots w_{n-1}' \in A^* \), so \( w \in (A^*)_+ \). Next, to show the other inclusion, take a string \( w \in A^* \), which, by definition, can be written as \( w_0 \cdots w_{n-1} \in A \setminus \{ \varepsilon \} \). If \( x, y \in \Sigma^* \) satisfy \( xy = w \), then there must be an \( i \in \{ 0, \ldots, n-1 \} \) and strings \( x', y' \in \Sigma^* \) such that \( x = w_0 \cdots w_{i-1}x' \), \( y = y'w_{i+1} \cdots w_{n-1} \) and \( x'y' = w_0 \cdots w_{i-1}x'y'w_{i+1} \cdots w_{n-1} \in A^*_A_+A^* \).

Solution, part B: As every regular language can be described by a regular expression, it is enough to show that every regular language \( R \) has a regular expression \( R_+ \) such that \( L(R_+) = L(R)^+ \) (note: \( R_+ \) is just notation). We prove this by induction on \( R \) considering all possible cases of \( R \)'s definition:

- \( R = \emptyset \) or \( R = w \in \Sigma^* \): \( L(R) \) is finite and so is then \( L(R)_+ \), so we construct \( R_+ \) as a disjunction of the strings in \( L(R)_+ \);
- \( R = R_0 \cup R_1 \): by the induction hypothesis, there are regular expressions \( R_0+ \) and \( R_1+ \) such that \( L(R_0+) = L(R_0) \cup L(R_1) \) and \( L(R_1+) = L(R_1) \). We then let \( R_+ = R_0+R_1+ \) so we have \( L(R_+) = L(R_0) \cup L(R_1) + L(R_0) + L(R_1) = L(R_0+) + L(R_1+) = L(R_0) + L(R_1) = L(R) \);
- \( R = R_0R_1 \): similarly to the first case, we obtain from the induction hypothesis two regular expressions \( R_0+ \) and \( R_1+ \) such that \( L(R_0+) = L(R_0) \) and \( L(R_1+) = L(R_1) \). We then let \( R_+ = R_0R_1+R_0R_1+ \), which gives us \( L(R_+) = L(R_0)L(R_1) + L(R_0) + L(R_1) = L(R_0) + L(R_1) + L(R_0) + L(R_1) = L(R_0) + L(R_1) = L(R) \);
- \( R = R_0^* \): again we use the induction hypothesis, this time obtaining a regular expression \( R_0^* \) with \( L(R_0^*) = L(R_0) \). Here we let \( R_+ = R_0^*R_0^* \) since then \( L(R_+) = L(R_0)^*L(R_0) = L(R_0)^*L(R_0) + L(R_0)^* + L(R_0)^* = (L(R_0)^*)_+ = (L(R_0))^+ = L(R)_+ \).
Problem 4. [10 points] Show a context-free grammar describing the language $\{0^a1^b0^{a+b} : a, b \in \mathbb{N}\}$.

Solution:

$$
\begin{align*}
S & \rightarrow 0S0 | X \\
X & \rightarrow \varepsilon | 1X0
\end{align*}
$$
Problem 5. [15 points] Prove that the language \( \{ x \# y : x, y \in \{0, 1\}^* \text{ and } x \neq y \} \) over the alphabet \( \{0, 1, \#\} \) is not regular and show a context-free grammar describing it.

Solution: If the language were regular, there would be a DFA \( M = (Q, \{0, 1, \#\}, \delta, q_0, F) \) recognizing it. Define \( q_i = \overline{\delta(q_0, 0^i)} \) for all \( i \in \mathbb{N} \) (note that this agrees with \( q_0 \)). Let now \( i, j \in \mathbb{N} \) with \( i \neq j \). Since \( 0^i \# 0^i \) is not in the language but \( 0^j \# 0^i \) is, we have that \( \overline{\delta(q_i, \#1^i)} \notin F \) but \( \overline{\delta(q_j, \#1^i)} \in F \), so \( q_i \neq q_j \). Therefore \( Q \) is an infinite set, a contradiction.

Let us write a context-free grammar to describe the language (the textual description is just commentary). We start by introducing a variable \( X \) that derives all strings in \( \{0, 1\}^* \):

\[
X \rightarrow \varepsilon | 0X | 1X
\]

Next let us introduce a variable \( A \) that derives zeroes and ones:

\[
A \rightarrow 0 | 1
\]

If \( x, y \in \{0, 1\}^* \) and \( x \neq y \), at least one of the three cases below occurs:

- \( |x| < |y| \);
- \( |x| > |y| \); or
- There is a natural number \( k < \min\{|x|, |y|\} \) such that the characters at (0-based) index \( k \) in \( x \) and \( y \) differ.

Let us introduce a variable \( L \) that derives \( A^m \# A^n \) with \( m < n \) to handle the first case:

\[
L \rightarrow L' A \\
L' \rightarrow \# | L' A | AL'A
\]

Similarly, we introduce a variable \( R \) that derives \( A^m \# A^n \) with \( m > n \) to handle the second case:

\[
R \rightarrow AR' \\
R' \rightarrow \# | AR' | AR'A
\]

In the third case, for some \( k \in \mathbb{N} \), \( x \in \Sigma^k \{0\} \Sigma^* \) and \( y \in \Sigma^k \{1\} \Sigma^* \) or vice-versa. Let us introduce a variable \( S_0 \) that derives strings of the form \( A^k 0X \# A^k 1X \):

\[
S_0 \rightarrow S_0' 1X \\
S_0' \rightarrow 0X \# | AS_0'
\]

We then introduce a variable \( S_1 \) to derive \( A^k 1X \# A^k 0X \):

\[
S_1 \rightarrow S_1' 0X \\
S_1' \rightarrow 1X \# | AS_1'
\]

To close it off, we introduce the starting symbol branching into all of the cases:

\[
S \rightarrow L | R | S_0 | S_1
\]
Problem 6. [15 points] Prove that the language \{xy : x, y \in \{0, 1\}^*, |x| = |y| \text{ but } x \neq y\} is not regular and show a context-free grammar describing it.

Solution: Similar to the previous one. If the language were regular, there would be a DFA \( M = (Q, \{0, 1\}, \delta, q_0, F) \) recognizing it. Define \( q_i = \delta(q_0, 0^i) \) for all \( i \in \mathbb{N} \) (note that this agrees with \( q_0 \)). Let now \( i, j \in \mathbb{N} \) with \( i \neq j \). Since \( 0^i1^i \) is in the language but \( 0^j1^j \) is not, we have that \( \delta(q_i, 1^i) \in F \) but \( \delta(q_j, 1^j) \notin F \), so \( q_i \neq q_j \). Therefore \( Q \) is an infinite set, a contradiction.

Again the explanation on how to construct the grammar is only commentary. Let \( x, y \in \{0, 1\}^* \) be two distinct strings of same size and let \( m \) be the (0-based) index of a character in \( x \) that differs from the corresponding character in \( y \). Then \( x \in \Sigma^m \{0\} \Sigma^n \) and \( y \in \Sigma^m \{1\} \Sigma^n \), where \( n = |x| - m - 1 = |y| - m - 1 \), or vice-versa. But then \( xy \in \Sigma^m \{0\} \Sigma^{m+n} \{1\} \Sigma^n \) or \( xy \in \Sigma^m \{1\} \Sigma^{m+n} \{0\} \Sigma^n \). This problem is thus similar to Problem 4.

Let us first once again introduce a variable to represent terminals:

\[
A \rightarrow 0 \mid 1
\]

Then let us introduce variables to handle this first case:

\[
S_0 \rightarrow XY \\
X \rightarrow 0 \mid AXA \\
Y \rightarrow 1 \mid AYA
\]

The second case can be handled similarly:

\[
S_1 \rightarrow YX
\]

We then join the cases with the starting variable:

\[
S \rightarrow S_0 \mid S_1
\]