Problem 1–4. Let us consider an automaton for recognizing words that have only $a$’s and $b$’s – and that have at least one symbol. For simplicity, let us only consider the alphabet consisting of only three symbols: $a$, $b$, and $c$. This automaton has 3 states:

- the start state $s$;
- the sink state $si$, and
- the final state $f$.

The transitions are as follows:

- from the start state $s$, symbols $a$ and $b$ lead to $f$, symbol $c$ leads to $si$;
- from $f$, symbols $a$ and $b$ lead back to $f$, while $c$ leads to $si$; and
- from $si$, every symbol leads to $si$.

Problem 1. Trace, step-by-step, how this finite automaton will check whether the following two words belong to this language:

- the word $aba$ (which this automaton should accept) and
- the word $acb$ (which this automaton should reject).

Problem 2. Use the above tracing to find the parts $x$, $y$, and $z$ of the word $aba$ corresponding to the Pumping Lemma. Check that the “pumped” word $xyyz$ will also be accepted by this automaton.

Problem 3. Write down the tuple $(Q, \Sigma, \delta, q_0, F)$ corresponding to this automaton:

- $Q$ is the set of all the states,
- $\Sigma$ is the alphabet, i.e., the set of all the symbols that this automaton can encounter;
- $\delta : Q \times \Sigma \rightarrow Q$ is the function that describes, for each state $q$ and for each symbol $s$, the state $\delta(q, s)$ to which the automaton that was originally in the state $q$ moves when it sees the symbol $s$ (you do not need to describe all possible transitions this way, just describe two of them);
• \( q_0 \) is the staring state, and
• \( F \) is the set of all final states.

**Problem 4.** Use a general algorithm that we had in class to generate a context-free grammar corresponding to this automaton. Show how this grammar will generate the word \( aba \).

**Problem 5.** Let \( A_1 \) be the automaton described in Problem 1–3. Let \( A_2 \) be an automaton that accepts all the strings that contain at least one \( a \). This automaton has two states: the start state and the sink state which is also final. The transitions are as follows:
- from the start state, \( b \) or \( c \) lead back to the start state,
- \( a \) leads to the final state;
- from the final state, any symbol leads back to this state.

Use the algorithm that we had in class to describe the following two new automata:
- the automaton that recognizes the union \( A_1 \cup A_2 \) of the two corresponding languages, and
- the automaton that recognizes the intersection of the languages \( A_1 \) and \( A_2 \).

**Problem 6.** Use the general algorithm that we learned in class to design a non-deterministic finite automaton that recognizes the language \( (a \cup b)(a \cup b)^* \):
- first, describe the automata for recognizing \( a \) and \( b \);
- then, combine them into the automata for recognizing the union \( a \cup b \), and the Kleene star \( (a \cup b)^* \);
- finally, combine the automata for \( a \cup b \) and \( (a \cup b)^* \) into an automaton for recognizing the desired composition of the two languages.

**Problem 7.** Use the general algorithm to transform the resulting non-deterministic finite automaton into a deterministic one.

**Problem 8–9.** Use a general algorithm to transform the finite automaton \( A_2 \) from Problem 5 into the corresponding regular expression.

**Problem 10.** Prove that the language \( L \) of all the words that have more \( a \)'s than \( b \)'s and no \( c \)'s is not regular.