Solutions to Test 1

Problem 1. Why do we need to study automata? Provide two main reasons.

Solution to Problem 1.

- To help develop a general understanding of which general problems are solvable and which are not.
- To understand how program are compiled.

Problem 2–4. Let us consider the automaton that has two states: g (good student) and p (student on probation); g is the starting state, p is the final state. The only three symbols are A, B, and F.

- From g, A and B lead back to g, and F leads to p.
- From p, any symbol leads back to p.

Problem 2. Trace, step-by-step, how this finite automaton will check that the word ABF belongs to this language. Use the above tracing to find the parts x, y, and z of the word ABF corresponding to the Pumping Lemma. Check that the "pumped" word $xyyz$ will also be accepted by this automaton.

Solution to Problem 2. Let us first trace the word ABF:

- we start in the starting state g;
- we read the first symbol A and stay in g;
- we read B and stay in g;
- we read F and move to p.

We have read all the letters of the word, we are in the final state, so the word is accepted.

We can describe this transition as follows:

$$\begin{array}{ccccccc} | & A & | & B & | & F & | \\ g & & g & & g & & p \end{array}$$

In the derivation of the word ABF, the first pair of repeating states is the pair of the g states:

$$\begin{array}{ccccccc} | & A & | & B & | & F & | \\ \underline{g} & & \underline{g} & & g & & p \end{array}$$

So:

- x is what is before the first repetition, i.e., $x = \Lambda$;
- y is what is in between the repetitions, i.e., $y = A$; and
- z is what is after the second repetition, i.e., $z = BF$.

By repeating the part between the two repetitions we get the derivation of the word $xyyz = AABF$:

$$\begin{array}{ccccccccc} | & A & | & A & | & B & | & F & | \\ g & & g & & g & & g & & p \end{array}$$

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

- Q is the set of all the states,
- Σ is the alphabet, i.e., the set of all the symbols that this automaton can encounter;
- $\delta : Q \times \Sigma \to Q$ 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.

Solution to Problem 3. Here, $Q = \{g, p\}$, $\Sigma = \{A, B, F\}$, $q_0 = g$, $F = \{p\}$, and the function δ is described by the following table:

	g	p
A	g	p
B	g	p
F	p	p

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 ABF.

Solution to Problem 4. The corresponding grammar has variables G and P corresponding to the states of the automaton. The variable G corresponding to the starting state g is the starting variable. We have the following rules:

$$G \to AG;$$

$$G \to BG;$$

$$G \to FP;$$

$$P \to AP;$$

$$P \to BP;$$

$$P \to FP;$$

$$P \to \varepsilon.$$

The corresponding derivation is:

$$\underline{G} \to A\underline{G} \to AB\underline{G} \to ABF\underline{P} \to ABF.$$

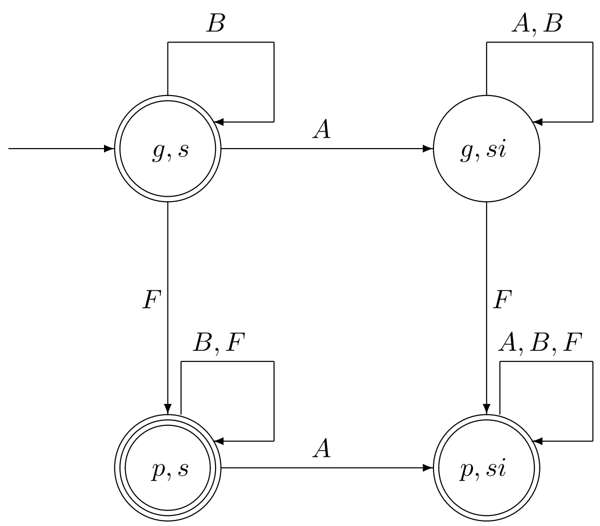
Problem 5. Let A_1 be the automaton described in Problem 2. Let A_2 be an automaton that accepts all the strings that do not contain As. This automaton has two states: the start state which is also final, and the sink state. The transitions are as follows:

- from the start state, B and F lead back to the start state, while A leads to the sink state;
- from the sink 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.

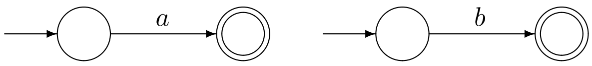
Solution to Problem 5.



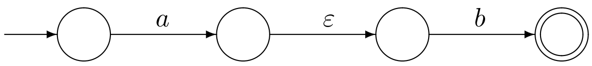
Problem 6. Use the general algorithm that we learned in class to design a non-deterministic finite automaton that recognizes the language $ab \cup a^*$:

- first, describe the automata for recognizing a and b;
- then, combine them into the automata for recognizing the concatenation ab and the Kleene star a^*;
- finally, combine the automata for ab and a^* into an automaton for recognizing the desired union of the two languages.

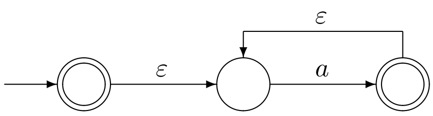
Solution to Problem 6. We start with the standard non-deterministic automata for recognizing the words a and b:



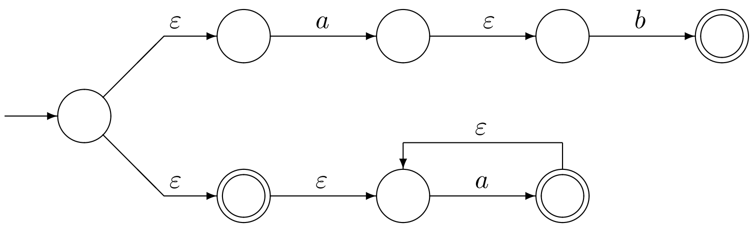
Then, we use the general algorithm for the concatenation to design a non-deterministic automaton for recognizing the language ab:



Now, we apply a standard algorithm for the Kleene star, and we get the following non-deterministic automaton for a^*:

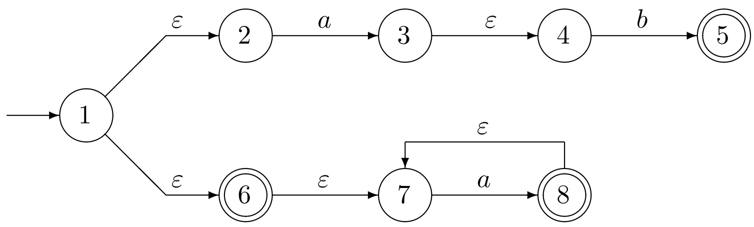


Now, we use the algorithm for union for combine them:

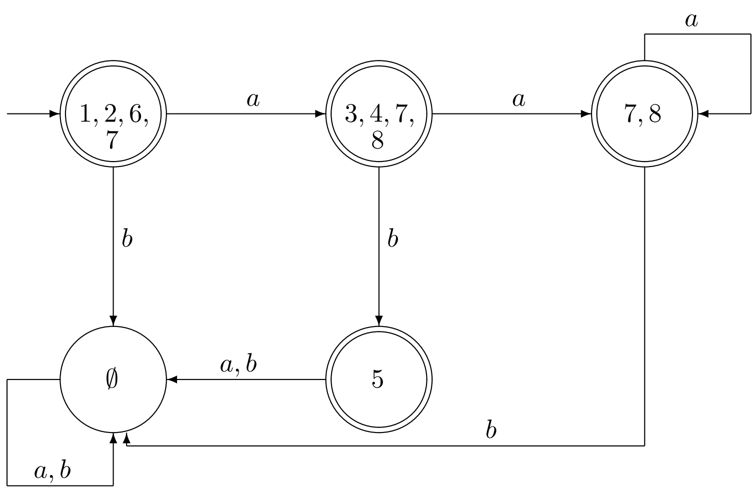


Problem 7. Use the general algorithm to transform the resulting non-deterministic finite automaton into a deterministic one.

Solution to Problem 7. First, we enumerate the states:

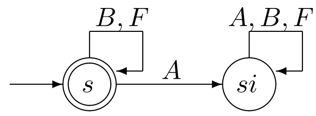


Then, we get the following deterministic automaton:



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

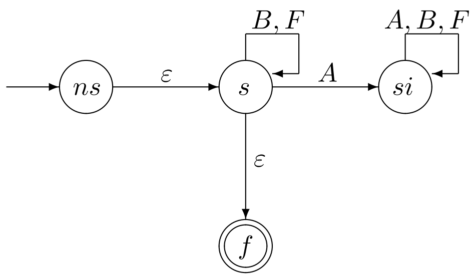
Solution to Problem 8–9. We start with the described automaton:



According to the general algorithm, first we add a new start state ns and a new final state f, and we add jumps:

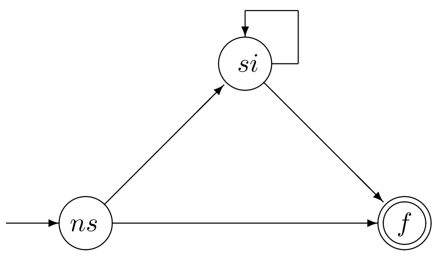
- from the new start state ns to the old start state, and
- from each old final state to the new final state f.

As a result, we get the following automaton.



Then, we need to eliminate the two intermediate states s and si one by one. We can start with eliminating s or with eliminating si. Let us show what happens in both cases.

First version, when we first eliminate the state s. First, we draw all possible arrows:



Now, to find expressions to place at all these arrows, we will use the general formula

$$R'_{i,j} = R_{i,j} \cup (R_{i,k} R^*_{k,k} R_{k,j}),$$

where k is the state that we are eliminating, i.e., in this case, the state $k = s$.

By applying this formula, and by using simplification formulas described in the lecture, we get the following results:

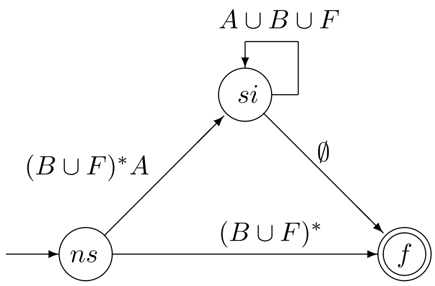
$$R'_{ns,si} = R_{ns,si} \cup (R_{ns,s} R^*_{s,s} R_{s,si}) = \emptyset \cup (\Lambda (B \cup F)^* A) = \emptyset \cup (B \cup F)^* A = (B \cup F)^* A;$$

$$R'_{ns,f} = R_{ns,f} \cup (R_{ns,s} R^*_{s,s} R_{s,f}) = \emptyset \cup (\Lambda (B \cup F)^* \Lambda) = \emptyset \cup (B \cup F)^* = (B \cup F)^*;$$

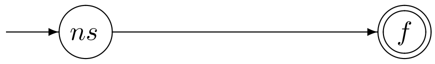
$$R'_{si,si} = R_{si,si} \cup (R_{si,s} R^*_{s,s} R_{s,si}) = A \cup B \cup F \cup (\emptyset \ldots) = A \cup B \cup F \cup \emptyset = A \cup B \cup F;$$

$$R'_{si,f} = R_{si,f} \cup (R_{si,s} R^*_{s,s} R_{s,f}) = \Lambda \cup (\emptyset \ldots) = \Lambda \cup \emptyset = \Lambda.$$

Thus, the 3-state a-automaton takes the following form:



Now, all that remains to do is to go from here to the 2-state a-automaton by eliminating the remaining state si:



The final expression is the corresponding expression for $R'_{ns,f}$:

$$R'_{ns,f} = R_{ns,f} \cup (R_{ns,si} R^*_{si,si} R_{si,f}) =$$

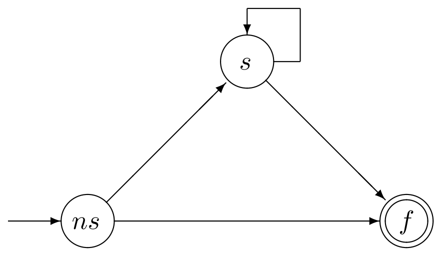
$$(B \cup F)^* \cup ((B \cup F)^* A (B \cup F \cup A)^* \emptyset) = (B \cup F)^* \cup \emptyset =$$

$$(B \cup F)^*.$$

The formula on the previous line is a regular expression corresponding to the original automaton.

First version – answer: $(B \cup F)^*$.

Second version, when we first eliminate the state *si*. First, we draw all possible arrows:



Now, to find expressions to place at all these arrows, we will use the general formula

$$R'_{i,j} = R_{i,j} \cup (R_{i,k} R^*_{k,k} R_{k,j}),$$

where k is the state that we are eliminating, i.e., in this case, the state $k = si$.

By applying this formula, and by using simplification formulas described in the lecture, we get the following results:

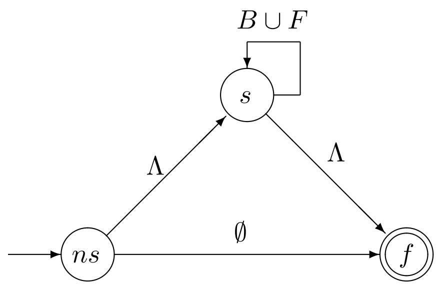
$$R'_{ns,s} = R_{ns,s} \cup (R_{ns,si} R^*_{si,si} R_{si,s}) = \Lambda \cup (\emptyset \ldots) = \Lambda \cup \emptyset = \Lambda;$$

$$R'_{ns,f} = R_{ns,f} \cup (R_{ns,si} R^*_{si,si} R_{si,f}) = \emptyset \cup (\emptyset \ldots) = \emptyset \cup \emptyset = \emptyset;$$

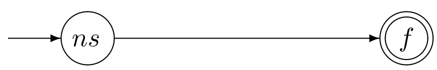
$$R'_{s,s} = R_{s,s} \cup (R_{s,si} R^*_{si,si} R_{si,s}) = (B \cup F) \cup (A(A \cup B \cup F)^* \emptyset) = (B \cup F) \cup \emptyset = B \cup F;$$

$$R'_{s,f} = R_{s,f} \cup (R_{s,si} R^*_{si,si} R_{si,f}) = \Lambda \cup (A(A \cup B \cup F)^* \emptyset) = \Lambda \cup \emptyset = \Lambda.$$

Thus, the 3-state a-automaton takes the following form:



Now, all that remains to do is to go from here to the 2-state a-automaton by eliminating the remaining state s:



The final expression is the corresponding expression for $R'_{ns,f}$:

$$R'_{ns,f} = R_{ns,f} \cup (R_{ns,s} R^*_{s,s} R_{s,f}) =$$
$$\emptyset \cup (\Lambda(B \cup F)^* \Lambda) =$$
$$\emptyset \cup (B \cup F)^* =$$
$$(B \cup F)^*.$$

The formula in the previous line is also a regular expression corresponding to the original automaton.

Problem 10. Prove that the language L of all the words that have fewer a's than b's is not regular.

Solution to Problem 10. We will prove it by contradiction. Let us assume that the language L is regular, and let us show that this assumption leads to a contradiction.

Since this language is regular, according to the Pumping Lemma, there exists an integer p such that every word from L whose length $\text{len}(w)$ is at least p can be represented as a concatenation $w = xyz$, where:

- y is non-empty;
- the length $\text{len}(xy)$ does not exceed p, and
- for every natural number i, the word $xy^iz \stackrel{\text{def}}{=} xy \ldots yz$, in which y is repeated i times, also belongs to the language L.

Let us take the word

$$w = a^p b^{p+1} = a \ldots ab \ldots b,$$

in which first a is repeated p times, then b is repeated $p+1$ times. The length of this word is $p + p + 1 = 2p + 1 > p$. So, by pumping lemma, this word can be represented as $w = xyz$ with $\text{len}(xy) \le p$. This word starts with xy, and the length of xy is smaller than or equal to p. Thus, xy is among the first p symbols of the word w – and these symbols are all a's. So, the word y only has a's.

Thus, when we go from the word $w = xyz$ to the word $xyyz$, we add at least one a, and we do not add any b's. So, in the word $xyyz$, there are now at least $p + 1$ letters a. Since there are $p + 1$ letters a, this means that the number of a's is no longer smaller than the number of b's. Thus, the word $xyyz$ cannot be in the language L, since by definition L only contains words which have more a's than b's.

On the other hand, by Pumping Lemma, the word $xyyz$ must be in the language L. So, we proved two opposite statements:

- that this word *is not* in L and
- that this word *is* in L.

This is a contradiction.

The only assumption that led to this contradiction is that L is a regular language. Thus, this assumption is false, so L is not regular.