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 programs are compiled.
Problem 2–4. Let us consider the automaton that has two states: $n$ (the student is in normal mood) and $h$ (the student is happy); $n$ is the starting state, $h$ is the final state. The only two symbols are $g$ (the student got a good grade on this test) and $t$ (time passed).

- From $n$, $g$ leads to $h$, and $t$ leads back to $n$.
- From $h$, $g$ leads back to $h$ and $t$ leads to $n$.

Problem 2. Trace, step-by-step, how this finite automaton will check that the word $gtg$ belongs to this language. Use the tracing to find the parts $x$, $y$, and $z$ of the word $gtg$ 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 how the automaton will accept the word $gtg$:

- we start in the starting state $n$;
- we read the first symbol $g$ and move to $h$;
- we read $t$ and move back to $n$;
- we read $g$ and go back to $h$.

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

Let us now trace how the automaton will accept the word $gtg$:

$$
\begin{array}{c|c|c|c|c|c|c|c}
\text{ } & g & t & g & t & g & t & g \\
\hline
n & h & n & h & n & h & n & h \\
\end{array}
$$

In this derivation, the first pair of repeating states is the pair of the $n$ states: $x$ is what is before the first repetition, i.e., $x = \Lambda$; $y$ is what is in between the repetitions, i.e., $y = gt$; and $z$ is what is after the second repetition, i.e., $z = g$.

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

$$
\begin{array}{c|c|c|c|c|c|c|c}
\text{ } & g & t & g & t & g & t & g \\
\hline
n & h & n & h & n & h & n & h \\
\end{array}
$$
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.

Solution to Problem 3. Here, \( Q = \{n, h\} \), \( \Sigma = \{g, t\} \), \( q_0 = n \), \( F = \{h\} \), and the function \( \delta \) is described by the following table:

\[
\begin{array}{c|cc}
   & n & h \\
\hline
  g & h & h \\
  t & n & n \\
\end{array}
\]
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 $gtg$.

Solution to Problem 4. The corresponding grammar has variables $N$ and $H$ corresponding to the states of the automaton. The variable $N$ corresponding to the starting state $n$ is the starting variable. We have the following rules:

$$
N \rightarrow gH;
N \rightarrow tN;
H \rightarrow gH;
H \rightarrow tN;
H \rightarrow \epsilon.
$$

The corresponding derivation is:

$$
N \rightarrow gH \rightarrow gtN \rightarrow gtgH \rightarrow gtg.
$$
**Problem 5.** Let $A_1$ be the automaton described in Problem 2. Let $A_2$ be an automaton that accepts all the strings that contain at least one symbol $g$ – indicating good news. This automaton has two states: the starting state $s$, and the final state $f$. The transitions are as follows:

- from the start state, $t$ lead back to the start state, while $g$ leads to the final state $f$;
- from the final state $f$, 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 \((g \cup t)^* g\):

- first, describe the automata for recognizing \(g\) and \(t\);
- then, combine them into the automata for recognizing the union \(g \cup t\), and the Kleene star \((g \cup t)^*\);
- finally, combine the automata for \((g \cup t)^*\) and \(g\) into an automaton for recognizing the desired composition of the two languages.

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

- the language \(g\) – that consists of a single word \(g\), and
- the language \(t\) – that consists of a single word \(t\):

Then, we use the general algorithm for the union to design a non-deterministic automaton for recognizing the language \(g \cup t\):

Now, we apply a standard algorithm for the Kleene star, and we get the following non-deterministic automaton for \((g \cup t)^*\):
Now, we use the algorithm for concatenation for combine them: final states of the automaton for $A$ are no longer final, and from each of them, we add a jump to the starting state of the automaton for $(g \cup t)^* g$: 

![Diagram](image-url)
Problem 7. Use the general algorithm to transform the resulting non-deterministic finite automaton into a deterministic one.

Solution to Problem 7. Let us first enumerate the states of the resulting non-deterministic automaton.

In the beginning, before we see any symbol, we are in state 3, and we can jump to 1, 4, 5, and 7. So, the resulting state is \{1, 3, 4, 5, 7\}.

- If in the state \{1, 3, 4, 5, 7\}, we see letter \(t\), we can go to 2 and from there, jump to 1, 4, 5, and 7. Thus, the resulting state is \{1, 2, 4, 5, 7\}.

- If in the state \{1, 3, 4, 5, 7\}, we see letter \(g\), we can go to 8 and from there, jump to 1, 4, 5, and 7. We can also go to 6. Thus, the resulting state is \{1, 4, 5, 6, 7, 8\}. This state contains the final state 6 and is, thus, final.

- If in the state \{1, 2, 4, 5, 7\}, we see letter \(t\), we can go to 2 and from there, jump to 1, 4, 5, and 7. Thus, the resulting state is the same state \{1, 2, 4, 5, 7\}.

- If in the state \{1, 2, 4, 5, 7\}, we see letter \(g\), we can go to 8 and from there, jump to 1, 4, 5, and 7. We can also go to 6. Thus, the resulting state is \{1, 4, 5, 6, 7, 8\}.

- If in the state \{1, 4, 5, 6, 7, 8\}, we see letter \(t\), we can go to 2 and from there, jump to 1, 4, 5, and 7. Thus, the resulting state is \{1, 2, 4, 5, 7\}.

- If in the state \{1, 4, 5, 6, 7, 8\}, we see letter \(g\), we can go to 8 and from there, jump to 1, 4, 5, and 7. We can also go to 6. Thus, the resulting state is the same state \{1, 4, 5, 6, 7, 8\}.

Thus, we arrive at the following deterministic automaton.

\[ \text{Diagram} \]
Problem 8–9. Use a general algorithm to transform the finite automaton from Problem 2 into the corresponding regular expression. Start with eliminating the state $n$.

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

![Automaton Diagram]

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 $nf$.

As a result, we get the following automaton.

![Enhanced Automaton Diagram]

Then, we need to eliminate the two intermediate states $n$ and $g$ one by one. We first eliminate the state $n$. First, we draw all possible arrows:

![Intermediate States Diagram]

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

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

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

10
By applying this formula, and by using simplification formulas described in the lecture, we get the following results:

\[
R'_{ns,h} = R_{ns,h} \cup (R_{ns,n}R'_{n,n}R_{n,h}) = \emptyset \cup (\Lambda t^* g) = \emptyset \cup t^* g = t^* g;
\]

\[
R'_{ns,nf} = R_{ns,nf} \cup (R_{ns,n}R'^*_{n,n}R_{n,nf}) = \emptyset \cup (\Lambda t^* \emptyset) = \emptyset \cup \emptyset = \emptyset;
\]

\[
R'_{h,h} = R_{h,h} \cup (R_{h,n}R'^*_{n,n}R_{n,h}) = g \cup (tt^* g);
\]

\[
R'_{n,nf} = R_{h,nf} \cup (R_{h,n}R'^*_{n,n}R_{n,nf}) = \Lambda \cup (tt^* \emptyset) = \Lambda \cup \emptyset = \Lambda.
\]

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 \( h \):

The final expression is the corresponding expression for \( R'_{ns,nf} \):

\[
R'_{ns,nf} = R_{ns,nf} \cup (R_{ns,h}R'_{h,h}R_{n,nf}) = \\
\emptyset \cup (t^* g(g \cup tt^* g)^* \Lambda) = \emptyset \cup (t^* g(g \cup tt^* g)^*) = \\
t^* g(g \cup tt^* g)^*.
\]

The formula on the previous line is a regular expression corresponding to the original automaton.
Problem 10. Prove that the language $L$ of all the words that have more $g$’s than $t$’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 $xy^i z \overset{\text{def}}{=} xy \ldots yz$, in which $y$ is repeated $i$ times, also belongs to the language $L$.

Let us take the word $w = t^p g^{p+1} = t \ldots tg \ldots g$, in which first $t$ is repeated $p$ times, then $g$ 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 $\text{len}(xy) \leq p$. 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 $t$’s. So, the word $y$ only has $t$’s.

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