Automata, Spring 2025, Final Exam

Problem 1. *Finite automata and regular languages.*

Problem 1a. Design a finite automaton for recognizing words that contain letter q but do not contain letter k. Assume that we are dealing with words that consist of only three letters: a, k, and q. The easiest is to have three states:

- the starting state d meaning that the word does not contain any of letter k or q;
- the final state f meaning that the word contains letter q and does not contain letter k;
- the error state e meaning that the word contains letter k; this state is a sink.

If you see a in any state you stay in this state. If you see k in any state, you go to e. If you see q in states d or f, you go to f. Show, step-by-step, how your automaton will accept the word aq.

Problem 1b. Use the general algorithm to design automata for recognizing union and intersection of the automaton A from Problem 1a with an automaton that accepts all the words that contain letter a. A natural way to design this new automaton B is to have 2 states: the starting state n in which the word does not have letter a, and the final state f in which the word has letter a. You need to describe transitions between these states.

Problem 1c. On the example of the automaton from the Problem 1a, show how the word aq can be represented as xyz in accordance with the pumping lemma.

Problem 1d. Use the general algorithm to describe a regular expression corresponding to the finite automaton from the Problem 1a.

Problem 1e-f. One of the possible way to describe the resulting language is by a regular expression $a^*q(a \cup q)^*$. Use the general algorithm to transform this regular expression into a finite automaton: first a non-deterministic one, then a deterministic one.

Problem 2. *Beyond finite automata: pushdown automata and context-free grammars*

Problem 2a. Prove that the set of all palindromes accepted by the automaton from Problem 1a is not regular and therefore, cannot be recognized by a finite automaton.

Problem 2b. Use the general algorithm to transform the finite automaton from the Problem 1a into a context-free grammar (CFG). Show, step-by-step, how this CFG will generate the word aq.

Problem 2c. For the context-free grammar from the Problem 2b, show how the word aq can be represented as $uvxyz$ in accordance with the pumping lemma.

Problem 2d. Use the general algorithm to translate the CFG from 2b into Chomsky normal form.

Problem 2e. Use the general algorithm to translate the CFG from 2b into a push-down automaton. Show, step-by-step, how this automaton will accept the word aq.

Problem 2f. A finite automaton is a particular case of a pushdown automaton. Use the general algorithm of transforming a pushdown automaton into a context-free grammar to show how the acceptance of the word aq by the finite automaton from Problem 1a can be translated into a derivation of this word in the context-free grammar.

Problem 3. *Beyond pushdown automata: Turing machines*

Problem 3a. Use the general algorithm to design a Turing machine that accepts exactly all the words accepted by a finite automaton from Problem 1a. Show, step-by-step, how this Turing machine will accept the word aq. Describe, for each step, how the state of the tape can be represented in terms of states of two stacks.

Problem 3b. Design Turing machines for computing $a - 2$ for numbers $a \geq 2$ in unary and in binary codes. Trace both Turing machines for $a = 4$.

Problem 4. *Beyond Turing machines: computability*

Problem 4a. Formulate Church-Turing thesis. Is it a mathematical theorem? Is it a statement about the physical world?

Problem 4b. Prove that the halting problem is not algorithmically solvable.

Problem 4c. Not all algorithms are feasible, but, unfortunately, we do not have a perfect definition is feasibility. Give a current formal definition of feasibility and an explanation of what practically feasible means, and give two examples:

- an example of an algorithm's running time which is feasible according to the current definition but not practically feasible, and
- an example of an algorithm's running time which is practically feasible but not feasible according to the current definition.

These examples should be different from what we studied in class and what is posted in solutions.

Problem 4d. Briefly describe what is P, what is NP, what is NP-hard, and what is NP-complete. Can an optimization problem be NP-complete? Is P equal to NP?

Problem 4e. When someone proves that a problem is NP-complete, what are positive and negative consequences of this? Give an example of an NP-complete problem: what is given, and what we want to find.

Problem 4f. Give definitions of a decidable (recursive) language and of a semi-decidable (recursively enumerable, Turing-recognizable) language. Give an example of a decidable language and an example of a language which is semi-decidable but not decidable.