Automata, Fall 2025, Final Exam

Problem 1. *Finite automata and regular languages.*

Problem 1a. Design a finite automaton for recognizing words that contain no more than one letter f. Assume that we are dealing with words that consist of letters a, b, c, d, and f. The easiest is to have three states:

- the starting state s – which is also final – meaning that the word does not contain any letters f;
- the intermediate state t – which is also final – meaning that the word contains exactly one letter f; and
- the error state e meaning that the word contains two or more letters f; this state is a sink.

If you see f in the state s, you go to i; if you see f in the state t, you go to e. If you see any other letter, you remain in the same state. Show, step-by-step, how your automaton will accept the word fab.

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 contains only letters a. A natural way to design this new automaton B is to have 2 states: the starting state s – – which is also final – in which the word has only letters a, and the error state e in which the word has other letters. You need to describe transitions between these states.

Problem 1c. On the example of the automaton from the Problem 1a, show how the word fab 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. Start with eliminating the errors state e, then eliminate t, and finally eliminate s.

Problem 1e-f. One of the possible way to describe the language of the all the words in the alphabet $\{a, b, c, d, f\}$ accepted by Automaton B is by a regular expression a^*. 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). For simplicity, ignore the letters c and d. Show, step-by-step, how this CFG will generate the word fab.

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

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 fab.

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 fab 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. For simplicity, ignore the letters c and d. Show, step-by-step, how this Turing machine will accept the word fab. 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 $n + 2$ for numbers a in unary and in binary codes. Trace both Turing machines for $n = 4$.

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.