Automata, Spring 2023, Final Exam

Problem 1. *Finite automata and regular languages.*

Problem 1a. Design a finite automaton for recognizing words that contain at least one letter. Assume that the input strings contain only symbols a and b. The easiest is to have two states:

- the starting state s indicating that we have not yet read any letters; and
- the state f indicating that we have already read at least one letter.

You just need to describe transitions between these states, and which states are final. Show, step-by-step, how your automaton will accept the word *aba*.

Problem 1b. Explain why in most computers binary numbers are represented starting with the lowest possible digit.

Problem 1c. On the example of the above automaton, show how the word *aba* can be represented as xyz in accordance with the pumping lemma.

Problem 1d. Use a general algorithm to describe a regular expression corresponding to the finite automaton from the Problem 1a. Start by eliminating the state s. (If you are running out of time, it is Ok not to finish, just eliminate the first state.)

Problem 1e-f. The resulting language can be described by a regular expression $(a \cup b)(a \cup b)^*$. Use a 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 it is not possible to have an automaton that, given a sequence of student's grades – such as ABA – checks whether this student's GPA is larger than 3.5 or not. *Hint:* for a student who only gets As and Bs, GPA is larger than 3.5 if and only if this student has more As than Bs.

Problem 2b. Use a 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 *aba*.

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

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

Problem 2e. Use a general algorithm to translate the CFG from 2b into an appropriate push-down automaton. Explain, step-by-step, how this automaton will accept the word *aba*.

Problem 2f. Use the general stack-based algorithms to show:

- how the compiler will transform a Java expression $5 - 8 - 23$ into inverse Polish (postfix) notation, and
- how it will compute the value of this expression.

Problem 3. *Beyond pushdown automata: Turing machines*

Problem 3a. According to medical doctors, a healthy lifestyle is when you have the same number of hours:

- to work (we will denote an hour when a person worked by t, from *trabajar*),
- to eat and have other types of fun (c, from *comer*), and
- to sleep (d, from *dormir*).

In a single day, this means 8 hours for each of these three categories; for a month, it means 240 hours of each, etc. Participants in an experiment write down every hour what they are doing.

- Some of the resulting sequences are healthy, e.g., *tcdtcd*.
- Other sequences are not healthy, e.g., *tttcd*.

Prove that the language L of all healthy sequences is not context-free (and therefore, cannot be recognized by a pushdown automaton).

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

Problem 3d-e. Design Turing machines for computing $a + 2$ in unary and in binary codes. Trace both Turing machines for $a = 1$.

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, explain what is means to be practically feasible, 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 must be different from what we studied in class.

Problem 4d. Briefly describe what is P, what is NP, what is NP-hard, and what is NP-complete. Is P equal to NP?

Problem 4e. Give an example of an NP-complete problem: what is given, and what we want to find.

Problem 4f. Give definitions of a recursive (decidable) language and of a recursively enumerable (Turing-recognizable) language.