Automata, Fall 2024, Final Exam, Section 2

Problem 1. *Finite automata and regular languages.*

Problem 1a. Design a finite automaton for recognizing odd binary numbers. The easiest is to have two states:

- the starting state s meaning that the last-read digit was not 1, and
- the final state f meaning that the last-read digit was 1.

If you see 1 in any state you move to f, if you see 0 in any state you move to s. Show, step-by-step, how your automaton will accept the word 101.

Problem 1b. Use the general algorithm to design automata for recognizing union and intersection of the automaton from Problem 1a with an automaton B that accepts all binary numbers that start with 0. A natural way to design this new automaton B is to have 3 states: the starting state s, the error state e, and the final state f. You need to describe transitions between these states.

Problem 1c. On the example of the automaton from the Problem 1a, show how the word 101 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 $(0 \cup 1)^*1$. 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 binary numbers in which there are more 1s than 0s 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 101.

Problem 2c. For the context-free grammar from the Problem 2b, show how the word 101 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 101.

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 101 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. Design a Turing machine that accepts all odd binary numbers. *Hint:* a binary number is odd if and only if its least significant digit is 1.

Problem 3b-c. Use the 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 101. 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 $a = 3$.

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.