TDDD65
Introduction to the Theory of Computation
2018-10-22

Materials allowed: A dictionary from English to any other language is allowed. No other books, notes etc. are allowed and no electronic equipment (calculators, computer, mobile phones etc.) are allowed.

Grading: The maximum number of points is 30 and 15 points are required to pass the examination. At least 15 p is required for grade 3, at least 20 p is required for grade 4 and at least 25 p is required for grade 5.

Please observe the following:

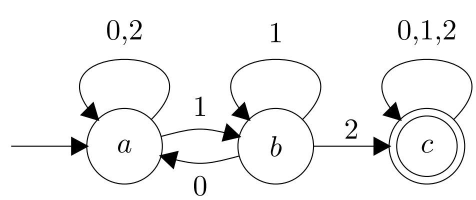
- Use only one side of each paper.
- Each problem must be solved on a separate paper (or several papers, if necessary. Subproblems of a problem (a, b, c etc.) may be solved on the same page.
- Properly justify all your answers. If you give only an answer without justification, you may get zero points even if the answer is correct.
- Make sure your answers are readable.
- Try to leave space for comments on every page.

Good luck!

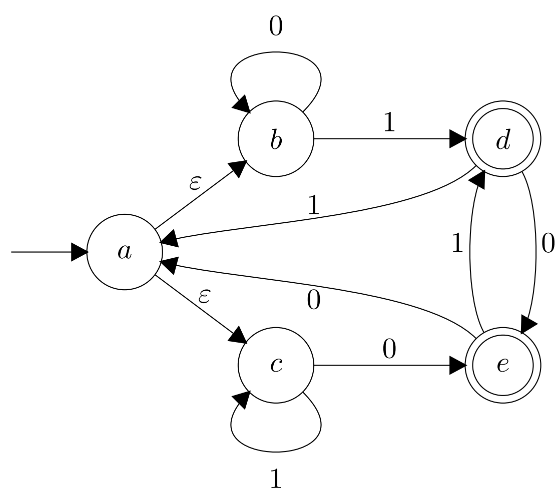
Problems

1. Assume the alphabet $\Sigma = \{0, 1, 2\}$. A string $x_1 x_2 \ldots x_n$ over Σ^* is in numerical order if $x_i \leq x_{i+1}$ for all i where $1 \leq i < n$. For example, the strings 0001222 and 11122 are in numerical order, but the strings 001102 and 1012 are not in numerical order. Draw the state transition diagram for a DFA that accepts exactly those non-empty strings over Σ^* that are *not* in numerical order. (4 p)

2. Convert the following DFA to a regular expression using the GNFA method. (4 p)



3. Convert the following NFA to an equivalent DFA, using the standard method (i.e. the subset construction method). (4 p)



4. Define the language $L = \{0^k 1 0^m 2 0^n \mid k, m \geq 0 \text{ and } n \leq k + m\}$ over the alphabet $\Sigma = \{0, 1, 2\}$. (6 p)

   (a) Prove that L is not regular by using the pumping lemma for regular languages.

   (b) Prove that L is a CFL by providing a CFG for it.

5. Prove that if two languages L_1 and L_2 are both decidable, then the language $L_1 L_2$ is also decidable. Recall that $L_1 L_2 = \{uw \mid u \in L_1 \text{ and } w \in L_2\}$ (4 p)

6. The concept of NP-completeness can be generalised to any complexity class X such that a language L is X-complete if $L \in X$ and $L' \leq_m^p L$ for all languages $L' \in X$ (i.e. NP-completeness is the special case where $X = NP$). (8 p)

   Recall that the complexity class coNP is defined to contain all languages L such that the complement language $\overline{L}$ is in NP. For instance, coNP contains the UNSAT problem, that asks if a boolean formula is not satisfiable, since NP contains the SAT problem. Also this concept can be generalized to any complexity class X such that co$X$ contains all languages L such that $\overline{L}$ is in X.

   (a) We can obviously define the concept of P-completeness for the class P. Is this a useful concept or not?

   (b) We can obviosly define the class coP. Is this a useful class or not?

(c) It is not currently known if NP=coNP or not. Suppose we could prove that P=NP. Would this have any implications for whether NP=coNP?

(d) A language is called NP-intermediate if it is in NP, but it is not in P and not NP-complete. The class NPI consists of the NP-intermediate problems. What would happen to this class if we could prove that P=NP?

It is essential that you explain your answers. Just an answer gives zero points!