

582206 Models of Computation (Autumn 2007)

Exercise 12 (4–7 December)

Basic exercises

Solve these by yourself. If there is anything unclear you can ask about it during the exercise session.

1. Three statements are given below. What can you deduce about their truth values using only the fact that the halting problem is undecidable, plus the related basic definitions? (Thus, for each statement the alternatives are “true”, “false” and “cannot be deduced from the given information”).
 - (a) There is an algorithm that for any given program P can decide the inputs on which P will halt.
 - (b) There is a program P such that no algorithm can decide the inputs on which P will halt.
 - (c) For all programs P it is the case that no algorithm can decide the inputs on which P will halt.

(An additional question: if the truth value of some of the above statement could not be deduced from the given information alone, figure out its truth value using any other necessary information given on the course.)

Discussion problems

Read the following problems and make sure you are familiar with the necessary basic concepts. You are not expected to solve the problems by yourself; we shall discuss them together.

2. Analogously with the Turing machine acceptance problem A_{TM} , we define the DFA acceptance problem *hyväksymisongelma*

$$A_{\text{DFA}} = \{ \langle A, w \rangle \mid A \text{ is a DFA that accepts the string } w \}.$$

(You can see the textbook for ideas of how this could be encoded, but the details are not important here.) Clearly A_{DFA} is decidable, but is it regular? In other words, is there a “universal DFA” that is able to simulate any other DFA? Justify your answer.

3. We are given the task of implementing a tool for program checking. According to the specification, the user can load a Java program into the tool. After that, the user can choose one line of code in the Java program, and give an input. The tool should tell, whether the Java program with the given input would ever execute the chosen line of code.

Can such a tool be implemented? Will your answer change, if instead of the user specifying the input to the Java program, the task is changed to deciding whether the chosen line of code would be executed on *any* possible input? Justify your answer.

4. Suppose that Church-Turing thesis turns out to be wrong, and someone manufactures a device that can efficiently solve the Turing machine halting problem. Call such a device the *halting oracle*. With access to the halting oracle, we can devise *superalgorithms*, which are like regular algorithms but may additionally make queries to the halting oracle.

It is of course trivial to write a superalgorithm to decide the Turing halting problem. But is there a superalgorithm that can decide the halting problem for *superalgorithms*? Justify your answer.