Basic exercises

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

1. As an exercise on a programming course, the students are required to write a Java program that prints the numbers 1, . . . , 10 and halts.
   (a) It is possible to write an automatic grading program that gets as input all the solutions turned in by the students and tells the teacher, which work correctly and which don’t?
   (b) It is possible to write a helper program that would at least detect solutions that go into an infinite loop without producing any output?

   Justify your answers.
   In both parts, it is required that the checking program never gives an incorrect output, even if a student knows the source code and tries to mislead it in either direction.

2. [Part of Sipser Problems 3.15 and 3.16]
   (a) Show that the class of decidable languages is closed under union, intersection and complementation.
   (b) Show that the class of recognizable languages is closed under union and intersection. Why can’t you use the same construction as in part (a) for complementation?

3. [Sipser Exercise 3.4] Give a formal definition of an enumerator (see page 154). Consider it to be a type of two-tape Turing machine that uses its second tape as the printer. Include a definition of the enumerated language.

Continues on the next page!
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.

4. Analogously with the Turing machine acceptance problem $A_{TM}$, we define the DFA acceptance problem

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

(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.

5. 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 eventually 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.

6. 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.

7. Consider enumerators, as in Problem 3. Show the following.

(a) A language is Turing-recognisable if and only if some enumerator enumerates it.

(b) A language is decidable if and only if some enumerator enumerates it in lexicographic order.

(c) A language $L$ is decidable if and only if both $L$ and its complement $\overline{L}$ can be enumerated by an enumerator.