Tutorial 3

Exercise 1 (compulsory)

Consider the following claim:

> **Claim:** If L is a decidable language and $L' \subseteq L$, then L' is a decidable language too.

Is this claim true? If yes, prove it. If not, give a counter-example.

Solution:
This claim is wrong. Let $L = \Sigma^*$. This is surely a decidable language, however, any language L' is now a subset of L. Then any undecidable language L' (and we know that undecidable languages exist — e.g. the equality problem of CFG, or the acceptance problem of a TM) will satisfy the precondition of the claim (being a subset of L) but will break the conclusion as L' is not a decidable language.

Exercise 2 (compulsory)

Consider the following problem:

> "Does a given regular expression R over $\Sigma = \{0, 1\}$ describe a language, which contains at least one string starting with 11 and ending with 0?"

1. Express the problem as a language RU_{110}.
2. Prove that RU_{110} is decidable.

Solution:

1. The problem has one input, the regular expression R, and should therefore be expressed as the language

   $$RU_{110} \stackrel{\text{def}}{=}$$

   $\{\langle R\rangle \mid R \text{ is a reg. expr. over } \{0,1\} \text{ and there is } w \in L(R) \text{ such that } w = 11u0 \text{ for some } u \in \{0,1\}^*\}$

2. We prove that RU_{110} is decidable by constructing a decider for it. This decider first converts a regular expression R to an NFA. Next, it converts the NFA to a DFA. Finally, we examine the DFA to see if there is a path from the start state to an accept state that begins with 11 and ends with 0.

   The complete algorithm is now as follows:

   "On input $\langle R\rangle$:

   1. Convert R to an NFA N.
   2. Convert N to an DFA M.
   3. Let s_0 denote the unique state of M that can be reached from the start state of M by reading the string 11.
   4. Let T denote the set of all states from which we can reach an accept state of M by reading a 0.
   5. Check if there is a path in M from s_0 to some state in T (can be done e.g. by depth-first search).
   6. If there is such a path then *accept*, else *reject*."

Exercise 3 (compulsory)

Prove that the class of decidable languages is closed under union, concatenation and Kleene star.

Solution:

- Closure under union. Let L_1 and L_2 be two decidable languages. By definition there are deciders M_1 and M_2 such that $L(M_1) = L_1$ and $L(M_2) = L_2$. We construct the following 2-tape Turing machine M:

  1. "On input x:
  2. Copy x on the second tape.
  3. On the first tape run M_1 on x.
  4. If M_1 accepted then M *accepts* else continue with step 5.
  5. On the second tape run M_2 on x.
  6. If M_2 accepted then M *accepts* else M *rejects*."

  Now M is surely a decider because both M_1 and M_2 are deciders and $L(M) = L_1 \cup L_2$. Any 2-tape decider is equivalent to some single tape decider. Hence we have a decider for the union of L_1 and L_2.

- Closure under concatenation. Let L_1 and L_2 be two decidable languages. By definition there are deciders M_1 and M_2 such that $L(M_1) = L_1$ and $L(M_2) = L_2$. We construct the following nondeterministic 3-tape Turing machine M:

  1. "On input x:
  2. Nondeterministically split the input string into two parts $x = w_1w_2$ and copy w_1 on second tape and w_2 on the third tape.
  3. On the second tape run M_1 on w_1.
  4. If M_1 accepted then continue with step 5, else M *rejects*.
  5. On the third tape run M_2 on w_2.
  6. If M_2 accepted then M *accepts* else M *rejects*."

  Now M is surely a nondeterministic decider because both M_1 and M_2 are deciders and $L(M) = L_1 \circ L_2$. Any 3-tape nondeterministic decider is equivalent to some single tape deterministic decider. Hence we have a decider for the concatenation of L_1 and L_2.

- Closure under Kleene star. Let L_1 be a decidable language. By definition there is a decider M_1 such that $L(M_1) = L_1$. We construct the following nondeterministic 2-tape Turing machine M:

  1. "On input x:
  2. Nondeterministically select a nonempty left-most part of the input x which has not been read yet and copy it on the second tape
  3. On the second tape run M_1 on the present string.
  4. If M_1 accepted and the whole input x was processed, then M *accepts*. If M_1 accepted and some suffix of x still has to be processed then clean the second tape and continue with step 2. If M_1 rejected then M *rejects*.

  Now M is surely a nondeterministic decider because M_1 is a decider and $L(M) = L_1^*$. Any 2-tape nondeterministic decider is equivalent to some single tape deterministic decider. Hence we have a decider for the Kleene star of L_1.

Exercise 4 (compulsory)

Prove that the class of recognizable languages is closed under intersection, concatenation and Kleene star.

Solution:

- Closure under intersection. Notice that (unlike for union) there is no problem in running the machine M_1 first and M_2 second because they both have to accept (and hence terminate). This means that the same construction as for the intersection of decidable languages presented in Lecture 3 will work also for recognizable languages.
- Closure under concatenation and Kleene star. Again, try to go carefully through the proofs in the previous exercise and realize that the same constructions will work also for recognizable languages. Even if the for some splittings of the input string x the recognizers may now loop, if there exists a good splitting then the nondeterminism will simply find it.

Exercise 5 (compulsory)

Consider the following two proofs that try to show that recognizable languages are closed under complement.

Proof 1:

> Let L be a recognizable language. By definition there is a Turing machine M such that $L(M) = L$. Consider the following machine M':
>
> 1. On input x:
> 2. Simulate M on x.
> 3. If M accepted then M' *rejects*. If M rejected then M' *accepts.*
>
> Now we can see that $L(M') = \overline{L}$. Hence $\overline{L}$ is recognizable.

Proof 2:

> Let L be a recognizable language. By definition there is a Turing machine M such that $L(M) = L$. Consider the following machine M':
>
> 1. On input x:
> 2. Simulate M on x.
> 3. If M accepted then M' *rejects*. If M rejected or looped on x then M' *accepts.*
>
> Now we can see that $L(M') = \overline{L}$. Hence $\overline{L}$ is recognizable.

As we already know, recognizable languages are not closed under complement and hence there must be an error in both proofs. Locate the errors by underlining them and explain (e.g. by giving a counter example) where is the problem.

Solution:
Proof 1:

> Let L be a recognizable language. By definition there is a Turing machine M such that $L(M) = L$. Consider the following machine M':
>
> 1. On input x:
> 2. Simulate M on x.

3. If M accepted then M' *rejects*. If M rejected then M' *accepts.*

Now we can see that $L(M') = \overline{L}$. Hence $\overline{L}$ is recognizable.

The presented construction does not recognize the complement of the language L. Consider a TM M over $\Sigma = \{a\}$ with three states q_0, q_{accept} and q_{reject} such that $\delta(q_0, a) = (q_0, a, R)$ and $\delta(q_0, \sqcup) = (q_0, a, R)$. On any given input x the machine M will always loop (keep moving the head to the right) and hence $L(M) = \emptyset$. Because M' run on x will first try to simulate M on x, it will never terminate either and hence $L(M') = \emptyset$, which is surely not equal to the complement of the empty language.

Proof 2:

Let L be a recognizable language. By definition there is a Turing machine M such that $L(M) = L$. Consider the following machine M':

1. On input x:
2. Simulate M on x.
3. If M accepted then M' *rejects*. If M rejected or looped on x then M' *accepts.*

Now we can see that $L(M') = \overline{L}$. Hence $\overline{L}$ is recognizable.

The problem here is that the machine M' has no way to verify whether the machine M on x loops or not. Hence conditions of the type "if a Turing machine M on an input x loops then do something" cannot be a part of any correct definition of a Turing machine because this test cannot be done algorithmically.

Exercise 6 (optional and challenging)

Problem 4.28 on page 212 (in international edition) or Problem 4.14 on page 211 (in standard edition).