Theoretical Computer Science (Bridging Course)

Dr. G. D. Tipaldi
F. Boniardi
Winter semester 2014/2015

University of Freiburg
Department of Computer Science

Exercise Sheet 7

Due: 18th December 2014

Exercise 7.1 (Decidable Languages)

Let L and L' be decidable languages. Prove the following properties.

(a) The complement $\overline{L}$ is decidable.

Solution: Let $L = L(M)$ for some Turing Machine M that always halts. We construct a TM $\overline{M}$ that $\overline{L} = L(\overline{M})$ as follows

1. The accepting states of M are made nonaccepting states of $\overline{M}$ with no transitions, i.e., in these states $\overline{M}$ will halt without accepting.
2. $\overline{M}$ has a new accepting state, say r; there are no transitions from r.
3. For each combination of a nonaccepting state of M and a tape symbol of M such that M has no transition, add a transition to the accepting state r.

Since M is guaranteed to halt, we know that $\overline{M}$ is also guaranteed to halt. Moreover, $\overline{M}$ accepts exactly those strings that M does not accept. Thus, $\overline{M}$ accepts $\overline{L}$

(b) The union $L \cup L'$ is decidable.

Solution: Since L and L' are decidable, there exist Turing Machines M and M' that decide L and L' respectively. Thus, we can construct a non-deterministic Turing Machine $M_\cup$ that runs M and M' in parallel. It is easy to see that such TM decides the language $L \cup L'$.

Exercise 7.2 (Decidable Languages)

Show that the following languages are decidable:

(a) $EQ_{DFA\_RE} = \{\langle D, R\rangle \mid D \text{ is a DFA and } R \text{ is a regular expression and } L(D) = L(R)\}$

Solution: We know from the lecture that $EQ_{DFA} = \{\langle A, B\rangle \mid A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$ is decidable. Let M be a Turing machine that decides this language. We construct a new TM M' that uses M to decide EQ_{DFA_RE} as follows:

M' = "On input $\langle D, R\rangle$, where D is a DFA and R is a regular expression:

1. Convert R to an equivalent DFA A by using the procedure for this conversion given in Theorem 1.28.
2. Run M on $\langle D, A\rangle$.
3. If M accepts, accept; if M rejects, reject."

Since M accepts iff $L(D) = L(A)$, and since $L(A) = L(R)$ this procedure is correct.

(b) $A_{\epsilon CFG} = \{\langle G\rangle \mid G \text{ is a CFG that generates } \epsilon\}$

Solution: We know from the lecture that $A_{CFG} = \{\langle G, w\rangle \mid G \text{ is a CFG that generates input string } w\}$ is decidable. Let M be a Turing machine that decides this language. We can obviously construct a TM M' that decides $A_{\epsilon CFG}$ as follows:

M' = "On input $\langle G\rangle$, where G is a CFG:

1. Run M on $\langle G, \epsilon \rangle$.
2. If M accepts, accept; if M rejects, reject."

(c) $ALL_{DFA} = \{\langle A \rangle \mid A \text{ is a DFA that recognizes } \Sigma^*\}$

Solution: A DFA A recognizes Σ^* iff all states that are reachable from the initial state are goal states. This can easily be checked by a Turing machine. Alternatively, we can use that EQ_{DFA} is decidable. Let M be a TM that decides this language. We can obviously construct a TM M' that decides ALL_{DFA} as follows:

$M' =$ "On input $\langle A \rangle$, where A is a DFA:

1. Create a DFA B that consists only of the initial state q_0 which is a goal state. For each symbol of the alphabet there is a transition from q_0 to q_0.
2. Run M on $\langle A, B \rangle$.
3. If M accepts, accept; if M rejects, reject."

M' decides ALL_{DFA}: M accepts iff $L(A) = L(B)$, and by construction $L(B) = \Sigma^*$.

Exercise 7.3 (Undecidable Languages)

Consider the problem of determining whether a two-tape Turing machine ever writes a non-blank symbol on its second tape, i.e.

$$N = \{\langle M, w \rangle \mid M \text{ is a two-tape Turing machine which writes a non-blank symbol onto its second tape when it runs on } w\}.$$

Show that N is undecidable. *Hint*: Use a reduction from A_{TM}.
Solution: Proof by contradiction: assume N is decidable and let D be a decider for N:

$$D(\langle M, w \rangle) = \begin{cases} \textit{accept} & \text{if } M \text{ running on } w \text{ writes a non-blank on its second tape} \\ \textit{reject} & \text{if } M \text{ running on } w \text{ does not write a non-blank on its second tape} \end{cases}$$

Use D to define a TM H as follows:
On input $\langle M, w \rangle$ where M is an arbitrary one-tape Turing machine and $w \in \Sigma^*$,

(1) Create a TM M' that differs from M only in being a two-tape Turing machine that does not use the second tape except for one case: If M accepts the input, it writes a non-blank symbol on the second tape. This can be done by a simple change of the transition function: Whenever the transition function δ of M maps the accept state of M for a tape symbol to the empty set (meaning that there is no transition and M halts), the transition function δ' of M' writes instead the non-blank on the second tape.

(2) If $D(\langle M', w \rangle) = \textit{accept}$, *accept*; if $D(\langle M', w \rangle) = \textit{reject}$, *reject*.

This means

$$H(\langle M, w \rangle) = \begin{cases} \textit{accept} & \text{if } M \text{ accepts } w \\ \textit{reject} & \text{if } M \text{ does not accept } w \end{cases}$$

So, H decides A_{TM} which is known to be undecidable. Hence, N cannot be decidable.