Exercise 1: The Halting Problem (2+2 points)

A Turing machine recognizes a language $L$ over the alphabet $\Sigma$, if it halts in the accepting state if and only if $s \in L$. Thus the set of Turing recognizable languages is the same as the set of Recursive Enumerable languages RE from the previous exercise. Commonly $L \in \text{RE}$ is also called semi-decidable. If a Turing machine recognizes $L$ and halts for every input string $s \in \Sigma^*$ then it decides $L$ and $L$ is called decidable. If no decider exists for a language $L$, then $L$ is called undecidable.

(a) Show that the Halting problem $H := \{\langle M, s \rangle \mid \text{Turing machine } M \text{ halts on input } s\}$ is semi-decidable\(^1\).

(b) Show that the Halting problem $H$ is undecidable.

Hint: You may use that we know that $U := \{\langle M, s \rangle \mid \text{Turing machine } M \text{ accepts input } s\}$ is an undecidable language from the lecture.

Solution

(a) A Turing machine $M_H$ that recognizes $H$ does the following. Let $\langle M, s \rangle$ bet the input of $M_H$ (it is easy to decide whether inputs encode a proper pair $\langle M, s \rangle$, i.e. are well-formed, thus we do not consider these). Turing machine $M_H$ simulates $M$ with input $s$. If the simulation of $M$ halts (in any state) then $M_H$ halts in the accepting state. Thus $\langle M, s \rangle$ is accepted if and only if $M$ halts on input $s$.

(b) Assume that $H$ is decidable. Then we have a Turing machine $D_H$ that decides $H$, i.e. that halts either in the accepting or in the rejecting state for every input. We construct a Turing machine $D_U$ from $D_H$ as follows.

Let $\langle M, s \rangle$ be the input of $D_U$. In the first part the Turing Machine $D_U$ simulates $D_H$ with the same input $\langle M, s \rangle$. If $D_H$ rejects $\langle M, s \rangle$ then $D_U$ enters the rejecting state as well and halts. If $D_H$ accepts $\langle M, s \rangle$ then $D_U$ simulates $M$ on $s$ in the second part. If $M$ accepts $s$ then so does $D_U$. If $M$ rejects $s$ then $D_U$ halts in the rejecting state as well.

If $M$ accepts $s$ it halts thus $D_H$ will accept $\langle M, s \rangle$. Therefore $D_U$ will enter the second part simulate $M$ on $s$ and accept since $M$ does so. If $M$ rejects $s$ and halts, $D_H$ will accept $\langle M, s \rangle$, but in the second part, when $D_U$ simulates $M$ on $s$, $D_U$ will halt in the rejecting state. Finally, if $M$ does not halt on input $s$, then $D_H$ rejects $\langle M, s \rangle$ and so does $D_U$.

Thus $D_U$ always halts on an input $\langle M, s \rangle$ and accepts it if and only if $M$ accepts $s$. Hence $D_U$ decides $U$ which is a contradiction to the fact that $U$ is undecidable.

\(^1\)Our definition of the halting problem deviates from the one on the lecture slides, but is also very common.
Exercise 2: Language Classes and Set Operations (2+3+1 points)

(a) Show that RE is not closed under complementation. I.e. for \( L \in \text{RE} \), \( \overline{L} := \Sigma^* \setminus L \) is in general not semi-decidable.

Hint: Use that with the results from exercise 1 you have a language which is undecidable but semi-decidable. What if \( \overline{H} \) were also semi-decidable?

(b) Argue why the set of decidable languages is closed under union, intersection and complement.

(c) Give a table showing whether or not the set of regular, context-free, decidable, semi-decidable languages are in general closed under the operations union, intersection and complement.

Solution

(a) Assume that for every \( L \in \text{RE} \), also \( \overline{L} \in \text{RE} \). Let \( M, \overline{M} \) be the Turing machines that recognize \( L \) and \( \overline{L} \) respectively. Using \( M \) and \( \overline{M} \) we construct a decider \( D_L \) that decides \( L \) as follows. For an input \( s \), \( D_L \) runs both Turing machines \( M \) and \( \overline{M} \) with input \( s \), simultaneously.

If \( M \) accepts \( s \) then \( D_L \) enters the accepting state and halts. If \( \overline{M} \) rejects \( s \) then so does \( D_L \). Thus \( D_L \) accepts exactly those inputs which are in \( L \) and rejects all others. Since either \( s \in L \) or \( s \in \overline{L} \) must be true, our decider \( D_L \) definitely halts and thus fulfills the requirements of a decider.

Now, consider the halting problem \( H \in \text{RE} \). As we did above we can construct \( D_H \) which decides \( H \). However we know from Exercise 1 that \( H \) is undecidable, a contradiction. Thus the assumption was wrong and in general for \( L \in \text{RE} \), \( L \notin \text{RE} \)

(b) Since a Turing machine \( M \) that decides a language \( L \) always halts either in the accepting state \( q_a \) or in the rejecting state \( q_r \), it suffices to swap those states to obtain a Turing machine that decides the complement \( \overline{L} \).

If we have two decidable languages \( L_1 \) and \( L_2 \) with deciders \( D_1, D_2 \), we can easily construct a decider \( D_\cup \) for the language \( L_1 \cup L_2 \) as follows. For input \( s \) the Turing machine \( D_\cup \) simulates \( D_1 \) and \( D_2 \) on \( s \). It doesn’t matter if the simulation of \( D_1 \) and \( D_2 \) is conducted consecutively or simultaneously, since both Turing machines \( D_1 \) and \( D_2 \) will halt eventually.

\( D_\cup \) accepts \( s \) if either \( D_1 \) or \( D_2 \) accept \( s \) and rejects if both \( D_1 \) and \( D_2 \) reject \( s \). The resulting machine \( D_\cup \) accepts exactly those strings in \( L_1 \cup L_2 \) and will halt eventually since \( D_1 \) and \( D_2 \) do so.

Finally, using DeMorgans law, we see that \( L_1 \cap L_2 = \overline{L_1 \cup L_2} \). The right term is decidable since we have already shown this for complement and union of decidable languages. Thus \( L_1 \cap L_2 \) is decidable.

(c) After showing closure properties of the above language classes in this and previous exercises we are finally able to subsume our results in the following Table.

<table>
<thead>
<tr>
<th>Language-class:</th>
<th>Regular</th>
<th>Context-Free</th>
<th>Decidable</th>
<th>Recursive enumerable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Union</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Complement</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>x</td>
</tr>
<tr>
<td>Intersection</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Exercise 3: Relation between Language Classes (2+2+2 points)

(a) Give a Venn diagram showing the relation between the set of regular, context-free, decidable and semi-decidable languages.

(b) Give an explanation why some of these sets are contained in others.
    
    **Hint:** Argue with the according automaton models that represent these language classes.

(c) Show that the subset relations are proper, e.g. by giving a language which is contained in the respective superset but not in the subset.
    
    **Hint:** You can cite results from previous exercises.

Solution

(a) We denote the given sets of languages by $RGL$, $CFL$, $DEC$, $RE$. The according Venn-diagram is given in Figure 1.

(b) We have $RGL \subset CFL \subset DEC \subset RE$. This is due to the fact that we can simulate an automaton of a given language class with the respective automaton type of the 'higher' class. So a NFA can easily be transformed into a PDA by just ignoring the stack (not pushing or popping anything and always reading $\varepsilon$). Hence any regular language is also recognized by a PDA.

A similar argument holds for PDAs and deciders (Turing machines or algorithms that halt for every input). Since a PDA always halts, it decides its corresponding language, thus it is a decider. Therefore context-free languages are decidable. Since a decider is a Turing machine, a decidable language is by definition also recursive enumerable.

(c) On exercise sheet 1 we have shown that $L_1 := \{a^nb^n \mid n \geq 0\} \notin RGL$, but from the lecture we know that $L_1 \in CFL$, therefore $RGL \subsetneq CFL$. Furthermore we know that $L_2 := \{ww^R \mid w \in \{a, b\}^*, u \in \{c\}^*, |u| = |w| \geq 1\} \in DEC$ from the 6th exercise sheet, however with the Pumping lemma for context-free languages it can be shown that $L_2 \notin CFL$, thus $CFL \subsetneq DEC$. Finally with the Halting Problem above we have a language with $H \in RE$ but $H \notin DEC$ and hence $DEC \subsetneq RE$.

---

![Figure 1: Venn-Diagram of the Language classes given in exercise 2 (a).](image-url)
Exercise 4: Decidable Languages (2+2+0 points)

(a) Show that \{\langle A \rangle \mid A \text{ is a CFG that can generate } \varepsilon \} is decidable.

(b) Show that \{\langle A \rangle \mid A \text{ is a DFA and } L(A) = \Sigma^* \} is decidable.

(c) Show that \{\langle R, S \rangle \mid R, S \text{ are regular expressions and } L(R) \subseteq L(S) \} is decidable. (This is a voluntary exercise, no points will be rewarded.)

Solution

(a) We have shown in the video lecture that testing membership for any given CFG is a decidable problem. Let \( M \) be such a turing machine that can test membership for a given CFG (e.g. a Turing machine which implements the Cocke-Younger-Kasami Algorithm from exercise sheet 4 does the trick). We can construct a Turing machine \( M' \), such that upon input \( A \) where \( A \) is a CFG, it will run \( M \) with input \( \langle A, \varepsilon \rangle \). If \( M \) accepts the input, then \( M' \) accepts the input \( A \) as well, otherwise \( M' \) rejects. Since \( M \) will give definite answer in finite time, we know \( M' \) will give definite answer in finite time as well. Hence, we know that the given language is decidable.

(b) Let \( B \) be a DFA such that the language generated by \( B \) is \( \Sigma^* \). That is, \( L_B = \Sigma^* \). (It is easy to see that this is always possible for any given alphabet \( \Sigma \).) We have shown in the video lecture that testing equivalence for two DFA is a decidable problem. Let \( M \) be such a turing machine that can test equivalence for two DFAs. We can construct a Turing machine \( M' \), such that upon input \( A \) where \( A \) is a DFA, it will run \( M \) with input \( \langle A, B \rangle \). If \( M \) accepts the input, then \( M' \) accepts the input \( \langle A \rangle \) as well, otherwise \( M' \) rejects. Since \( M \) will give definite answer in finite time, we know \( M' \) will give definite answer in finite time as well.

Another approach is to give an algorithm \( A \) which decides the problem directly. Algorithm \( A \) does the following. Interpret the input DFA \( \langle A \rangle \) as graph and conduct a breadth first search (BFS) which finishes in finite time, since a DFA (as the name says) has a finite number of states. If every state, which the BFS reaches is accepting, algorithm \( A \) accepts input DFA \( \langle A \rangle \). If the BFS finds a state which is non-accepting then \( A \) rejects. If we accept, every input \( s \) transfers DFA \( A \) into an accepting state, thus \( \forall s \in \Sigma^* : s \in L_A \Rightarrow A = \Sigma^* \). If \( A \) rejects \( \langle A \rangle \) it is possible to find a string \( s \in \Sigma^* \) for which the DFA \( A \) halts in a non-accepting state, thus \( s \notin L_A \neq \Sigma^* \). A Turing machine implementing \( A \) decides the given language.