CSCI 2210: Theory of Computation

Problem Set 6 Solutions

Problem 1. Show that the following languages are decidable.

(a) $C_{regex} = \{\langle R, S\rangle \mid R \text{ and } S \text{ are regular expressions and } L(R) \subseteq L(S)\}$

Solution. There are a few facts that we can combine to build a Turing machine that decides C_{regex}.

- $\mathsf{E}_{\mathsf{DFA}}$ is decidable, so there is a TM that decides it. Let's call this TM M_{e_dfa}.
- We have procedures to obtain a DFA from a regular expression (Regex $\to$ NFA $\to$ DFA).
- Regular languages are closed under complement and intersection, and there are procedures to obtain the corresponding DFAs.

Using M_{e_dfa} as a subroutine and the other facts listed above, we can define the following Turing machine that decides C_{regex}.

M_{c_regex} = "On input $\langle R, S\rangle$:
  1. Convert regular expressions R, S to equivalent DFAs D_R, D_S.
  2. Construct a DFA D that recognizes the language $L(D_R) \cap \overline{L(D_S)}$.
  3. Run M_{e_dfa} on $\langle D\rangle$.
  4. If M_{e_dfa} accepts, **accept**. Otherwise, **reject**."

Observe that if $L(R) \subseteq L(S)$, then $L(R) \cap \overline{L(S)} = \emptyset$. If the intersection is not $\emptyset$, then there was some $w \in L(R)$, that was not in $L(R)$, and we would have $L(R) \not\subseteq L(S)$

◊

(b) $\mathsf{EQ}_{\mathsf{DFA/REX}} = \{\langle D, R\rangle \mid D \text{ is a DFA, } R \text{ is regular expression, and } L(D) = L(R)\}$

Solution. We know that $\mathsf{EQ}_{\mathsf{DFA}}$ is decidable, so let M_{eq_dfa} be the TM that decides it. In $\mathsf{EQ}_{\mathsf{DFA/REX}}$, rather than two DFAs, we have one DFA and one regular expression but we can convert any regular expression into a DFA (as mentioned in the previous problem). The following Turing machine decides $\mathsf{EQ}_{\mathsf{DFA/REX}}$.

M_{eq_dfarex} = "On input $\langle D, R\rangle$:
  1. Convert regular expression R to an equivalent DFA D_R.
  2. Run M_{eq_dfa} on $\langle D, D_R\rangle$.
  3. If M_{eq_dfa} accepts, **accept**. Otherwise, **reject**."

◊

Problem 2. Consider the following facts:

- Regular languages are closed under intersection.
- Regular expression $\Sigma^* w \Sigma^*$ describes all strings over alphabet Σ that have w as a substring.

Show that language L is decidable.

$$L = \{\langle D, u\rangle \mid D \text{ is a DFA and there exists } w \in L(D) \text{ such that } u \text{ is a substring of } w\ \}$$

Solution. Using the fact that $\mathsf{E}_{\mathsf{DFA}}$ is decidable, we have TM M_{e_dfa} that decides it. We can define the following Turing machine to decide L.

M_{dfa_u} = "On input $\langle D, u\rangle$:

1. Construct regular expression $\Sigma^* u \Sigma^*$.
2. Convert regular expression to DFA D_u.
3. Construct a DFA $D_\cap$ that recognizes $L(D_u) \cap L(D)$.
4. Run M_{e_dfa} on $\langle D_\cap\rangle$.
5. If M_{e_dfa} accepts, **reject**. Otherwise, **accept**."

Observe that $L(D_u) \cap L(D)$ is the set of all strings w that have u as a substring and are accepted by DFA D. If $L(D_u) \cap L(D) = \emptyset$, then there is no string accepted by D that contains u as substring. If $L(D_u) \cap L(D) \neq \emptyset$, then there is at least one string with the property.

$\diamondsuit$

Problem 3. Consider languages $L_1 \subseteq L_2 \subseteq L_3$. Knowing that L_1 and L_3 are decidable, can one conclude that L_2 is also decidable? Justify your answer

Solution. L_2 may or may not be decidable. Consider Σ the alphabet for L_2, whatever that language might be. Let $L_1 = \emptyset$ and $L_3 = \Sigma^*$. Both L_1, L_3 are regular languages, and therefore also decidable. Now consider $L_2 = \mathsf{A}_{\mathsf{TM}}$. We have $L_1 \subseteq L_2 \subseteq L_3$ but A_{TM} is not decidable. Similarly, if $L_2 = \{ab, ba\}$, we'd still have $L_1 \subseteq L_2 \subseteq L_3$ but now L_2 is decidable.

$\diamondsuit$

Problem 4. Show that the following languages are undecidable.

(a) $\mathsf{REV}_{\mathsf{TM}} = \{\langle M\rangle \mid M \text{ is a TM such that } L(M) = (L(M))^R\}$. *Recall that* $L^R = \{w^R \mid w \in L\}$.

Solution. We can show that $\mathsf{REV}_{\mathsf{TM}}$ is undecidable by showing a reduction $\mathsf{A}_{\mathsf{TM}} \leq_m \mathsf{REV}_{\mathsf{TM}}$ that maps each pair $\langle M, w\rangle$ to $\langle M'\rangle$, where M' is the following Turing Machine:

M' = "On input x:

1. If $x \neq ab$ and $x \neq ba$, **reject**.

2. If $x = ab$, **accept**.
3. If $x = ba$,
4. Run M on w.
5. If M accepts, **accept**. Otherwise, **reject**."

Suppose that M_{rev_tm} can decide $\mathsf{REV_{TM}}$. We can then construct a Turing machine M_{a_tm} as follows:

M_{a_tm} = "On input $\langle M, w\rangle$:
1. Use M and w to construct M' as described above.
2. Run M_{rev_tm} on $\langle M'\rangle$.
3. If M_{rev_tm} accepts, **accept**. Otherwise, **reject**."

Observe that if $\langle M, w\rangle \in \mathsf{A_{TM}}$ then M accepts w and $L(M') = \{ab, ba\}$, so $\langle M'\rangle \in \mathsf{REV_{TM}}$ and M_{a_tm} accepts M, w. Conversely, if $\langle M, w\rangle \notin \mathsf{A_{TM}}$, then $L(M') = \{ab\}$ and $\langle M'\rangle \notin \mathsf{REV_{TM}}$. In this case, M_{a_tm} rejects M, w. This shows that M_{a_tm} decides $\mathsf{A_{TM}}$. This is a contradiction, so there can't be a Turing machine that decides M_{rev_tm}. Therefore, M_{rev_tm} is undecidable.

$\diamondsuit$

(b) $\mathsf{HALT_{any}} = \{\langle M\rangle \mid M \text{ is a TM that halts on at least one string}\}$

Solution. We can show that $\mathsf{HALT_{any}}$ is undecidable by showing a reduction $\mathsf{A_{TM}} \leq_m \mathsf{HALT_{any}}$ that maps each pair $\langle M, w\rangle$ to $\langle M'\rangle$, where M' is the following Turing Machine:

M' = "On input x:
1. Run M on w.
2. If M accepts, **accept**. Otherwise, **loop**."

Suppose that M_{any_halt} can decide $\mathsf{HALT_{any}}$. We can then construct a Turing machine M_{a_tm} as follows:

M_{a_tm} = "On input $\langle M, w\rangle$:
1. Use M and w to construct M' as described above.
2. Run M_{any_halt} on $\langle M'\rangle$.
3. If M_{any_halt} accepts, **accept**. Otherwise, **reject**."

If $\langle M, w\rangle \in \mathsf{A_{TM}}$ then M halts and accepts on input w and we can say thar M' will halt for any input x. Note that M' ignores its own input and uses the result of running M on w as the criterion for accepting any string x. Since M' halts on all strings, $\langle M'\rangle \in \mathsf{HALT_{any}}$ and M_{a_tm} accepts M, w.

Conversely, if $\langle M, w\rangle \notin \mathsf{A_{TM}}$ then M doesn't accept w. In this scenario, we have two cases: (1) M loops on w, which then implies that M' will loop for any string x; (2) M rejects w, in which case M' is also made to loop for any string x. So $\langle M'\rangle \notin \mathsf{HALT_{any}}$

and M_{a_tm} rejects M, w. This shows that M_{a_tm} decides $\mathsf{A_{TM}}$. This is a contradiction, so there is no Turing machine that decides $\mathsf{HALT_{any}}$.

$\diamondsuit$

Problem 5. Show that the semidecidable languages are closed under intersection but NOT closed under complement.

Solution.

Closed under intersection. This argument is very similar to the one used to show that decidable languages are closed under intersection.

For any two semidecidable languages L_1 and L_2, let M_1 and M_2 be Turing machines that recognize them. We construct $M_\cap$ that recognizes $L_1 \cap L_2$:

$M_\cap$ = "On input w:
1. Run M_1 on w.
2. If M_1 rejects, **reject**.
3. Else,
4. Run M_2 on w.
5. If M_2 accepts, **accept**. Otherwise, **reject**."

$M_\cap$ accepts w if both M_1 and M_2 accept it. If M_1 or M_2 reject w, then $M_\cap$ should also reject, and if M_1 doesn't halt or M_2 doesn't halt, then $M_\cap$ also doesn't halt. Therefore, $M_\cap$ recognizes $L_1 \cap L_2$ because it accepts w, when $w \in L_1 \cap L_2$ and it reject or doesn't halt when $w \notin L_1 \cap L_2$.

NOT closed under complement. Suppose that semidecidable languages are closed under complement. So for any L that is semidecidable, we have that $\overline{L}$ must also be semidecidable. But if both L and $\overline{L}$ are semidecidable, we have that L is also decidable. This would imply that all semidecidable languages are also decidable. This is a contradiction, since we know that $\mathsf{A_{TM}}$ is semidecidable but not decidable.

$\diamondsuit$

Problem 6. Show that if L is semidecidable and $L \leq_m \overline{L}$, then L is decidable.

Solution. We want to show that L is decidable and one way to do that is by showing that L and $\overline{L}$ are both semidecidable. We are given that L is semidecidable, so it remains to show that $\overline{L}$ is also semidecidable.

First, we are going to argue that if $A \leq_m B$, then $\overline{A} \leq_m \overline{B}$. Assuming $A \leq_m B$ is true, we know there is a function f such that, $w \in A$ if and only if $f(w) \in B$. That is equivalent to say that $w \notin A$ if and only if $f(w) \notin B$, which is equivalent to $w \in \overline{A}$ if and only if $f(w) \in \overline{B}$. Therefore, we can conclude $\overline{A} \leq_m \overline{B}$.

Now, by using the fact above, from $L \leq_m \overline{L}$, we also have $\overline{L} \leq_m L$. Given that L is semidecidable and $\overline{L} \leq_m L$, we can conclude that $\overline{L}$ is also semidecidable, and therefore L must be decidable.

$\diamondsuit$