Theory of Computation

Spring 2023, Homework #3 Solution

1.

The proof is done by reducing A_{TM} to S_{TM}.

Suppose S_{TM} is decidable and M_S decides it. Consider the following TM

M_A = " On input $\langle M, w \rangle$ where M is a TM and w is a string, use $\langle M \rangle$ and w to construct

M_1 = "On input x:

(1) If $x = 01$, accept.

(2) Otherwise, Run M on w. Accept if M accepts. Reject if M rejects. Loop if M loops. "

Run M_S on $\langle M_1 \rangle$.

1) If M_S accepts, accept.

2) If M_S rejects, reject. "

M_A accepts $\langle M, w \rangle \Longrightarrow M_S$ accepts $\langle M_1 \rangle \Longrightarrow \langle M_1 \rangle \in S_{TM} \Longrightarrow M$ accepts w and $L(M_1) = \Sigma^*$.

M_A rejects $\langle M, w \rangle \Longrightarrow M_S$ rejects $\langle M_1 \rangle \Longrightarrow \langle M_1 \rangle \notin S_{TM} \Longrightarrow M$ does not accept w and M_1 accepts 01 but does not accept $10 = (01)^R$.

Therefore M_A decides A_{TM}. However, A_{TM} is undecidable and this is a contradiction. So S_{TM} is undecidable.

2.

(Proof for the $\leftarrow$ part)

Let B be a Turing-decidable language such that $A = \{x \mid \text{there exists } y \text{ such that } \langle x, y \rangle \in B\}$.

Let M_B be the TM that decides B. Consider the following TM

M_A = "On input w,

1) Run M_B on w. If M_B accepts, accept. (In this case, $y = \epsilon$ since $w \in B$.)

2) For $l = 1, 2, \cdots, n, \cdots$, do the following loop:

For every $y \in \Sigma^*$ and $|y| = l$, run M_B on $\langle w, y \rangle$. If M_B accepts, accept.

Otherwise continue with the loop."

We can conclude that:

(a) For any w accepted by M_A, there exists y such that $\langle w, y \rangle \in B$. So $w \in A$.

(b) By definition, if the input $w \in A$, there exists y such that $\langle w, y \rangle \in B$. If $\langle w, \epsilon \rangle \in B$, then step 1) in M_A accepts. Otherwise, step 2) in M_A goes through all strings and will eventually find the corresponding y and accepts. This means $w \in L(M_A)$ if $w \in A$.

(c) If $w \notin A$, M_A loops forever in step 2) and never accepts. This means $w \notin L(M_A)$ if $w \notin A$.

Based on (a) and (b), $L(M_A) = A$ so A is Turing-recognizable.

(Proof for the $\rightarrow$ part)

Let M_A be a TM that recognizes A. Defined a language $C = \{\langle x, y\rangle \mid x$ is accepted by M_A in at most $|y|$ steps. $\}$. Consider

$M_C =$ " On input $\langle x, y\rangle$ where x and y are strings,

1) Simulate running M_A on x ***one step at a time***.
2) If M_A accepts, accept.
3) If M_A rejects or does not halt after $|y|$ steps, reject."

Clearly, $L(M_C) = C$. Since M_C always halts, C is decidable. We then show that $A = \{x \mid$ there exists y such that $\langle x, y\rangle \in C\}$.

Consider the following cases:

(a) For $x \in A$, M_A accepts x in finite number of steps. Let n be the number of steps. Clearly a string y where $|y| \geq n$ will result in $\langle x, y\rangle$ being accepted by M_C. That is, there exists y such that $\langle x, y\rangle \in C$.

(b) For $x \notin A$, clearly $\langle x, y\rangle \notin C$ for any y. That is, there are no y such that $\langle x, y\rangle \in C$.

Based on (a) and (b), $x \in A \iff$ there exists y such that $\langle x, y\rangle \in C$. Hence $A = \{x \mid$ there exists y such that $\langle x, y\rangle \in C\}$. Since we already proved that C is decidable, C matches the definition of B in the original questions.

3 (a).

Ans: No.

Let $A = \{0^n1^n \mid n \in N\}$, which is not a regular language. Let $B = \{0\}$, which is a regular language. Let $\Sigma = \{0,1\}$ be the alphabet for both A and B. Define $f : \Sigma^* \rightarrow \Sigma^*$ where $f(w) = \begin{cases} 0 & \text{if } w \in A \\ 1 & \text{if } w \notin A \end{cases}$. Since A is CFL so A is decidable. Therefore f is computable (since we can use A's decider to test w and output 0 or 1 accordingly).

If $w \in A$, then $f(w) = 0 \in B$. If $w \notin A$, then $f(w) = 1 \notin B$. So $A \leq_m B$.

3 (b).

Consider $\Sigma = \{0,1\}$. Let $B = \{\langle 1, M\rangle \mid \langle M\rangle \in E_{TM}\} \cup \{\langle 0, M\rangle \mid \langle M\rangle \notin E_{TM}\}$. Then $\overline{B} = \{\epsilon\} \cup \{\langle 0, M\rangle \mid \langle M\rangle \in E_{TM}\} \cup \{\langle 1, M\rangle \mid \langle M\rangle \notin E_{TM}\}$. Note $\epsilon \notin E_{TM}$.

(1) To prove B is undecidable: Let $f : \Sigma^* \rightarrow \Sigma^*$, $f(w) = 1w$. Clearly f is computable. If $w \in E_{TM}, f(w) = 1w \in B$. If $w \notin E_{TM}, f(w) = 1w \notin B$. So f is the reduction of E_{TM} to B. Since E_{TM} is undecidable, B is undecidable.

(2) To prove $B \leq_m \overline{B}$: Let $g : \Sigma^* \rightarrow \Sigma^*$ where $g(w) = \begin{cases} 0x & \text{if } w = 1x, x \in \Sigma^* \\ 1x & \text{if } w = 0x, x \in \Sigma^* \\ 0 & \text{if } w = \epsilon \end{cases}$. Clearly g is computable. First consider the case when $|w| \geq 1$. Since $g(w)$ flips the first alphabet of w, $w \in B \iff g(w) \in \overline{B}$. Then consider the case when $w = \epsilon$. $\epsilon \notin B$ but $g(\epsilon) = 0 \in B$. So $g(\epsilon) \notin \overline{B}$. Therefore g is the reduction of B to $\overline{B}$.

4.

Let A and B be two disjoint co-Turing-recognizable languages. Then there exists two Turing machines, $M_{\overline{A}}$ and $M_{\overline{B}}$, that recognize $\overline{A}$ and $\overline{B}$ respectively. Consider Turing machine

M = "On input w:

Run both $M_{\overline{A}}$ and $M_{\overline{B}}$ on the input w in parallel.

At each step:

(1) If $M_{\overline{A}}$ accepts, reject.

(2) Else if $M_{\overline{B}}$ accepts, accept.

(3) Else continue to the next step."

We then prove that (i) $L(M)$ is decidable and (ii) $L(M)$ separates A and B.

i) Since A and B are disjoint, $\overline{A} \cup \overline{B} = \{\Sigma^*\}$. That is, for any input $w \in \Sigma^*$, w is accepted by either $M_{\overline{A}}$, $M_{\overline{B}}$, or both. Since M stops as soon as either $M_{\overline{A}}$ or $M_{\overline{B}}$ accepts w, M halts on all inputs. So $L(M)$ is a decidable language.

ii) For any input w to be accepted by M, it must be accepted by $M_{\overline{B}}$. So $L(M) \subseteq \overline{B}$, which implies $B \subseteq \overline{L(M)}$. Similarly, for any input w to be rejected by M, it must be accepted by $M_{\overline{A}}$. So $\overline{L(M)} \subseteq \overline{A}$, which implies $A \subseteq L(M)$. Hence $L(M)$ separates A and B.

5.

Let A and B be two languages where $A, B \in \text{NP}$. So there exist nondeterministic polynomial time Turing machines M_A and M_B that decide A and B, respectively.

(I) NP is closed under union: Consider a NTM

M = "On input w,

(1) Run M_A on w, if M_A accepts, accept.

(2) Run M_B on w, if M_B accepts, accept.

(3) Reject. "

Clearly $L(M) = L(M_A) \cup L(M_B) = A \cup B$. Since M_A and M_B are both deciders, M always halts. Finally, since steps (1) and (2) can both be done in polynomial time (w.r.t. $|w|$), M is a polynomial time decider. Therefore $L(M) \in \text{NP}$.

(II) NP is closed under intersection: Consider a NTM

M = "On input w,

(1) Run M_A on w, if M_A rejects, reject.

(2) Run M_B on w, if M_B rejects, reject.

(3) Accept. "

Clearly $L(M) = L(M_A) \cap L(M_B) = A \cap B$. Since M_A and M_B are both deciders, M always halts. Finally, since steps (1) and (2) can both be done in polynomial time (w.r.t. $|w|$), M is a polynomial time decider. Therefore $L(M) \in$ NP.

(III) NP is closed under concatenation: Consider a NTM

M = "On input w,

(1) Nondeterministically split w into two substrings w_1 and w_2, where $w = w_1 \cdot w_2$.

(2) Run M_A on w_1, if M_A rejects, reject.

(3) Run M_B on w_2, if M_B rejects, reject.

(4) Accept. "

Clearly $L(M) = L(M_A) \cdot L(M_B) = A \cdot B$. Since M_A and M_B are both deciders, M always halts. Finally, since steps (1), (2) and (3) can all be done in polynomial time (w.r.t. $|w|$), M is a polynomial time decider. Therefore $L(M) \in$ NP.

(IV) NP is closed under Kleene star: Consider a NTM

M = "On input w,

(1) If $w = \epsilon$, accept.

(2) Nondeterministically choose a number m where $1 \leq m \leq |w|$.

(3) Nondeterministically split w into m substrings: $w = w_1 \cdot w_2 \cdot \cdots \cdot w_m$.

(4) For $i = 1,2,\cdots,m$, run M_A on w_i , if M_A rejects, reject.

(5) Accept. "

Clearly $L(M) = (L(M_A))^* = A^*$. Since M_A is a decider, M always halts. Suppose M_A decides A in time $O(n^k)$. Step (4) takes $O(n \cdot n^k) = O(n^{k+1})$. So M is still a polynomial time decider. Therefore $L(M) \in$ NP.