Assignment 4 solutions

---

1. Let $\Sigma = \{0, 1\}$, and assume A and B are Turing-recognizable languages such that $A \cup B = \Sigma^*$. Prove that there exists a decidable language $C \subseteq \Sigma^*$ such that

$$A \cap \overline{B} \subseteq C \qquad \text{and} \qquad \overline{A} \cap B \subseteq \overline{C}.$$

Solution. Because A and B are Turing-recognizable languages, there must exist DTMs M_A and M_B such that $\mathrm{L}(M_A) = A$ and $\mathrm{L}(M_B) = B$. Define $C = \mathrm{L}(K)$ for a new DTM K having input alphabet Σ that operates in the following way:

On input x:
1. Set $t = 1$.
2. Simulate M_A on input x for t steps. If M_A accepts x within t steps, then *accept*.
3. Simulate M_B on input x for t steps. If M_B accepts x within t steps, then *reject*.
4. Set $t = t + 1$ and goto step 2.

We must now prove that C has the required properties.

First let us prove that C is decidable. It suffices to prove that K halts on all input strings, for then we will have that K decides C. The fact that K halts on all inputs follows from the assumption $A \cup B = \Sigma^*$, for there must therefore be some positive integer t for which either M_A or M_B accepts x within t steps. (It could be that they both accept x, but this is fine—we just want to be sure that K never runs forever.)

Now let us prove the two set containments above. For any string $x \in A \cap \overline{B}$ we must have that M_A accepts x (because $x \in A$) and M_B does not accept x (because $x \in \overline{B}$). It must therefore be that K accepts x, so $x \in C$. We have just proved that $A \cap \overline{B} \subseteq C$. The other containment is similar. For any string $x \in \overline{A} \cap B$ we must have that M_A does not accept x (because $x \in \overline{A}$) and M_B does accept x (because $x \in B$). It must therefore be that K rejects x, so $x \in \overline{C}$.

---

2. Let Σ be an alphabet, and assume that we have fixed a scheme for encoding every possible DTM M as a string $\langle M \rangle \in \Sigma^*$ in the usual way, and define a language $A \subseteq \Sigma^*$ as

$$A = \big\{ \langle M \rangle \,:\, M \text{ is a DTM that halts on at least one input string} \big\}.$$

Prove that A is not decidable.

Solution. To prove that A is undecidable, it suffices to prove $\text{HALT} \leq_m A$. For any DTM M and any string w over the input alphabet of M, define a new DTM M_w as follows:

On input x:

Ignore the input string x and run M on input w.

It is the case that M_w halts on *every* string if M halts on w, and M_w *never* halts if M does not halt on w.

Now choose a string $z \in \Sigma^*$ with $z \notin A$, and define a function

$$f(y) = \begin{cases} \langle M_w \rangle & \text{if } y = \langle M, w \rangle \text{ for a DTM } M \text{ and a} \\ & \text{string } w \text{ over the input alphabet of } M \\ z & \text{otherwise} \end{cases}$$

for all $y \in \Sigma^*$. This is a computable function, and we will now prove that it is a reduction from HALT to A.

Suppose that M is a DTM and w is a string over the input alphabet of M. These implications hold:

$$\langle M, w \rangle \in \text{HALT} \Rightarrow M_w \text{ halts on all input strings} \Rightarrow \langle M_w \rangle \in A \Rightarrow f(\langle M, w \rangle) \in A,$$
$$\langle M, w \rangle \notin \text{HALT} \Rightarrow M_w \text{ never halts} \Rightarrow \langle M_w \rangle \notin A \Rightarrow f(\langle M, w \rangle) \notin A.$$

For any string y that does not take the form $y = \langle M, w \rangle$, for some DTM M and a string w over the input alphabet of M, we have $y \notin \text{HALT}$ and $f(y) = z \notin A$. We have therefore proved that

$$y \in \text{HALT} \Leftrightarrow f(y) \in A$$

for all $y \in \Sigma^*$. Consequently, HALT $\leq_m A$, which completes the solution.

---

3. Define two languages as follows:

$$\begin{aligned} \text{E}_{\text{DTM}} &= \{\langle M \rangle \,:\, M \text{ is a DTM with } \text{L}(M) = \varnothing\} \\ \text{DISJ}_{\text{DTM}} &= \{\langle M_1, M_2 \rangle \,:\, M_1 \text{ and } M_2 \text{ are DTMs with } \text{L}(M_1) \cap \text{L}(M_2) = \varnothing\}. \end{aligned}$$

   We already discussed E_{DTM} in lecture. The language DISJ_{DTM} contains all encodings $\langle M_1, M_2 \rangle$ of pairs of DTMs whose corresponding languages are *disjoint*.

   (a) Prove that $\text{E}_{\text{DTM}} \leq_m \text{DISJ}_{\text{DTM}}$.
   (b) Prove that $\text{DISJ}_{\text{DTM}} \leq_m \text{E}_{\text{DTM}}$.

   (When answering this question you should assume that E_{DTM} and DISJ_{DTM} are languages over the same alphabet Σ.)

Solution. (a) Define function $f : \Sigma^* \to \Sigma^*$ as

$$f(\langle M \rangle) = \langle M, M \rangle,$$

for every DTM M. (For any string x that does not encode a DTM, define $f(x)$ to be any fixed string that is not contained in DISJ_{DTM}.) The function f is computable, and for every DTM M we have

$$\langle M \rangle \in \text{E}_{\text{DTM}} \Leftrightarrow \langle M, M \rangle \in \text{DISJ}_{\text{DTM}} \Leftrightarrow f(\langle M \rangle) \in \text{DISJ}_{\text{DTM}}.$$

It follows that $\text{E}_{\text{DTM}} \leq_m \text{DISJ}_{\text{DTM}}$.

(b) Define function $f : \Sigma^* \to \Sigma^*$ so that for any two DTMs M_1 and M_2, we have

$$f(\langle M_1, M_2 \rangle) = \langle M_3 \rangle,$$

where M_3 is a DTM operating as follows:

On input x:
1. Run M_1 on input x.
2. Run M_2 on input x.
3. If both M_1 and M_2 accept x, then *accept*, else *reject*.

Observe that $\mathrm{L}(M_3) = \mathrm{L}(M_1) \cap \mathrm{L}(M_2)$. (This holds regardless of whether or not M_1 and M_2 halt on all inputs.)

The function f is computable, and we have

$$\langle M_1, M_2 \rangle \in \mathrm{DISJ}_{\mathrm{DTM}} \Leftrightarrow \langle M_3 \rangle \in \mathrm{E}_{\mathrm{DTM}} \Leftrightarrow f(\langle M_1, M_2 \rangle) \in \mathrm{E}_{\mathrm{DTM}}.$$

Therefore $\mathrm{DISJ}_{\mathrm{DTM}} \leq_m \mathrm{E}_{\mathrm{DTM}}$.

---

4. Prove that there does not exist a DTM M that simultaneously satisfies both of the following two properties:

   (i) If K is a DTM that halts on all input strings over its alphabet and $\mathrm{L}(K)$ is infinite, then M accepts $\langle K \rangle$.

   (ii) If K is a DTM that halts on all input strings over its alphabet and $\mathrm{L}(K)$ is finite, then M does not accept $\langle K \rangle$.

Solution. Suppose the contrary: there does exist such a DTM M. We will use this assumption to prove that

$$\mathrm{DIAG} = \{\langle T \rangle \,:\, T \text{ is a DTM and } \langle T \rangle \notin L(T)\}$$

is Turing-recognizable (which we know is false). First, for an arbitrary DTM T, define a DTM K_T as follows (for an arbitrary choice of an alphabet Σ):

On input $x \in \Sigma^*$:

1. Run T on $\langle T \rangle$ for $|x|$ steps.
2. If T accepts $\langle T \rangle$ within $|x|$ steps, then *reject*, otherwise *accept*.

It holds that K_T always halts, and that $L(K_T)$ is infinite if and only if T does not accept $\langle T \rangle$. The function $f(\langle T \rangle) = \langle K_T \rangle$ is a computable function.

Now consider a DTM D defined as follows:

On input $\langle T \rangle$:

1. Compute $\langle K_T \rangle = f(\langle T \rangle)$.
2. Run M on $\langle K_T \rangle$.

Given that K_T always halts, it holds that M accepts $\langle K_T \rangle$ if and only if $L(K_T)$ is infinite, and therefore D accepts $\langle T \rangle$ if and only if T does not accept $\langle T \rangle$. Therefore $L(D) = \mathrm{DIAG}$, which contradicts the fact that DIAG is not Turing-recognizable.

5. Let $\Sigma = \{0,1\}$ and let $A, B \subseteq \Sigma^*$ be languages.

   (a) Prove that if A and B are both in NP, then the union $A \cup B$ is also in NP.

   (b) Prove that if A is NP-complete, B is in P, $A \cap B = \varnothing$, and $A \cup B \neq \Sigma^*$, then $A \cup B$ is NP-complete.

Solution. (a) Under the assumption that A is in NP, there must exist a polynomially bounded time-constructible function $p : \mathbb{N} \to \mathbb{N}$ and a language $C \in \mathrm{P}$ such that

$$x \in A \Leftrightarrow \left(\exists y \in \Sigma^{p(|x|)}\right)\big[\langle x, y\rangle \in C\big]$$

for every $x \in \Sigma^*$. Similarly, under the assumption that B is in NP, there must exist a polynomially bounded time-constructible function $q : \mathbb{N} \to \mathbb{N}$ and a language $D \in \mathrm{P}$ such that

$$x \in B \Leftrightarrow \left(\exists z \in \Sigma^{q(|x|)}\right)\big[\langle x, z\rangle \in D\big]$$

for every $x \in \Sigma^*$.

Define $r : \mathbb{N} \to \mathbb{N}$ as $r(n) = p(n) + q(n)$ for each $n \in \mathbb{N}$, which is a polynomially bounded time-constructible function. Also define a language E as follows:

$$E = \big\{\langle x, yz\rangle \,:\, y \in \Sigma^{p(|x|)}, z \in \Sigma^{q(|x|)} \text{ and } \big(\langle x, y\rangle \in C \text{ or } \langle x, z\rangle \in D\big)\big\}.$$

Given that C and D are in P, it is straightforward to decide E in polynomial time, so $E \in \mathrm{P}$. It holds that

$$x \in A \cup B \Leftrightarrow \left(\exists w \in \Sigma^{r(|x|)}\right)\big[\langle x, w\rangle \in E\big],$$

and therefore $A \cup B \in \mathrm{NP}$.

(b) Given that A is NP-complete and B is in P, it holds that both A and B are in NP, and thus $A \cup B \in \mathrm{NP}$ by part (a).

It remains to show that $A \cup B$ is NP-hard, which follows if we can prove that $C \leq_m^p A \cup B$ for some NP-hard language C. As A is NP-complete, and therefore NP-hard, it is enough to show that $A \leq_m^p A \cup B$.

Define a function f as

$$f(x) = \begin{cases} y & \text{if } x \in B \\ x & \text{if } x \notin B, \end{cases}$$

where y is any fixed string that is not in $A \cup B$. (Such a string must exist because $A \cup B \neq \Sigma^*$.) Given that $B \in \mathrm{P}$, we have that f is polynomial-time computable.

If $x \in A$, then $f(x) = x$ (because $A \cap B = \varnothing$), and therefore $f(x) \in A \cup B$.

If $x \notin A$, then there are two cases: $x \in B$ and $x \notin B$. If $x \in B$, then $f(x) = y \notin A \cup B$. If $x \notin B$, then $f(x) = x$, and because $x \notin A$ and $x \notin B$ we have $f(x) \notin A \cup B$.

Thus,

$$x \in A \Leftrightarrow f(x) \in A \cup B,$$

and therefore $A \leq_m^p A \cup B$.