Homework 5 Solution

Formal Languages and Automata (CS322)

Last update: December 13, 2025

1. Let A and B be languages over $\{\mathtt{0}, \mathtt{1}\}$. Prove or disprove the following.

(a) If both A and B are Turing-decidable, then $A \leq_{\mathrm{m}} B$.

(b) If A is Turing-decidable and B is not, then $A \leq_{\mathrm{m}} B$.

(c) If B is Turing-decidable and A is not, then $A \leq_{\mathrm{m}} B$.

(d) If neither A nor B is Turing-decidable, then $A \leq_{\mathrm{m}} B$.

Solution. Let $\Sigma = \{\mathtt{0}, \mathtt{1}\}$.

(a) Answer: False

Consider $A = \emptyset$ and $B = \Sigma^*$. Note that both A and B are Turing-decidable. Suppose that $A \leq_{\mathrm{m}} B$. Then there exists a computable function $f \colon \Sigma^* \to \Sigma^*$ such that for all $w \in \Sigma^*$, $w \in A$ iff $f(w) \in B$. Consider $w = \epsilon$. However, $w \in A$ is false and $f(w) \in B$ is true. Hence a contradiction. Therefore, it is not always true that $A \leq_{\mathrm{m}} B$ when both A and B are Turing-decidable. □

(b) Answer: True

Since B is not Turing-decidable, $B \neq \emptyset$ and $B \neq \Sigma^*$. So take $w_1 \in B$ and $w_2 \in \Sigma^* \setminus B$. Define a function $f \colon \Sigma^* \to \Sigma^*$ by $f(w) = \begin{cases} w_1 & (w \in A) \\ w_2 & (w \notin A) \end{cases}$. Since A is Turing-decidable, f is computable. Also, for all $w \in \Sigma^*$, $w \in A$ iff $f(w) \in B$. Therefore, it is always true that $A \leq_{\mathrm{m}} B$ when A is Turing-decidable and B is not. □

(c) Answer: False

Consider $A = A_{\mathsf{TM}}$ (see Q2 for its definition) and $B = \emptyset$. Note that A is not Turing-decidable and B is Turing-decidable. Suppose that $A \leq_{\mathrm{m}} B$. Since B is Turing-decidable and $A \leq_{\mathrm{m}} B$, we must have that A is Turing-decidable, which is not true. Hence a contradiction. Therefore, it is not always true that $A \leq_{\mathrm{m}} B$ when B is Turing-decidable and A is not. □

(d) Answer: False

Consider $A = \overline{A_{\mathsf{TM}}}$ (where the overline denotes the complement) and $B = A_{\mathsf{TM}}$. Note that neither A nor B is Turing-decidable. Also, A is not Turing-recognizable and B is Turing-recognizable. Suppose that $A \leq_{\mathrm{m}} B$. Since B is Turing-recognizable and $A \leq_{\mathrm{m}} B$, we must have that A is Turing-recognizable, which is not true. Hence a contradiction. Therefore, it is not always true that $A \leq_{\mathrm{m}} B$ when neither A nor B is Turing-decidable. □

2. Recall two languages

$$E_{\mathsf{TM}} \coloneqq \{\langle M\rangle : M \text{ is a Turing machine and } L(M) = \emptyset\},$$
$$A_{\mathsf{TM}} \coloneqq \{\langle M, w\rangle : M \text{ is a Turing machine and } M \text{ accepts } w\}.$$

We proved the undecidability of E_{TM} by using the undecidability of A_{TM} in the lecture. Prove that there is no mapping reduction from A_{TM} to E_{TM}.

Solution. We first prove that if $A \leq_m B$ and if B is Turing-recognizable, then A is also Turing-recognizable. By the definition of mapping reducibility, there always exists a computable f so that $w \in A$ if and only if $f(w) \in B$. As B is Turing-recognizable, we can construct a Turing machine which accepts every $w \in A$ by first mapping w to $f(w)$ and then checking if $f(w) \in B$.

From the lecture, we know that $A \leq_m B$ if and only if $\overline{A} \leq_m \overline{B}$. Thus if $A_{TM} \leq_m E_{TM}$, then $\overline{A_{TM}}$ is Turing-recognizable as $\overline{E_{TM}}$ is Turing-recognizable. However, we proved in the lecture that A_{TM} is not Turing-recognizable, hence by contradiction, there is no mapping reduction from A_{TM} to E_{TM}. $\square$

3. By assigning integers and an order to states, tape symbols, and a transition function, we can represent a TM as a binary string, that is, we can fix an encoding of each TM. Then many binary strings are not a valid encoding of a TM at all. However, if we map those strings to some canonical trivial TM, e.g., the TM that immediately halts and rejects on any input, then we may assume that every binary string corresponds to a TM.

Let $M_1, M_2, \ldots$ be an enumeration of all Turing machines such that the i-th TM M_i corresponds to the TM whose encoding $\langle M_i \rangle$ is the i-th binary string w_i, where the binary strings are lexicographically ordered. Prove that the language

$$L_{\mathrm{d}} = \{w_i : i \geq 1 \text{ and } w_{2i} \notin L(M_i)\}$$

is undecidable by using the diagonal argument.

Solution. Suppose there exists a TM M that recognizes L, i.e., $L_d = L(M)$. We consider the following TM M'.

$$M'(w_i) = \begin{cases} \textsc{Accept} & \text{if } w_{i/2} \in L(M) \\ \textsc{Reject} & \text{otherwise} \end{cases}$$

On input w_i, M' first computes $i/2$ and checks whether it is an integer or not. If so, then M' computes $w_{i/2}$ (this is possible due to the above description of the enumeration of all TM's.) and simulates M on input $w_{i/2}$. The TM M' accepts if M accepts $w_{1/2}$ and rejects otherwise.

Since M' is a TM, M' must appear in the enumeration of all TM's, say M' is the k-th TM with $k \geq 1$ with the encoding $w_k = \langle M' \rangle$. Now, we ask whether w_k is in L_d.

- If $w_k \in L_d = L(M)$, then we have $w_{2k} \in L(M')$ by the definition of M'. This means that w_{2k} is accepted by $M_k = M'$. Then by the definition of L_d, w_k is not in L_d, which is a contradiction.
- If $w_k \notin L_d = L(M)$, then we have $w_{2k} \notin L(M')$ by the definition of M', that is, w_{2k} is not accepted by $M_k = M'$. Then by the definition of L_d, w_k is in L_d. This is also a contradiction.

$\square$

4. We say that an instance of the Post Correspondence Problem is restricted to Σ if the domino strings of the instance lie in Σ^*. In each of the following questions, you have to justify the correctness of the reduction.

(a) Show that the Post Correspondence Problem is undecidable even when the instances are restricted to an alphabet of size 2 by reducing an arbitrary instance to an instance restricted to an alphabet of size 2.

(b) Let $\mathsf{OVERLAP}_{\mathsf{CFG}} = \{\langle G_1, G_2\rangle : G_1, G_2 \text{ are CFGs and } L(G_1) \cap L(G_2) \neq \emptyset\}$. Give a reduction from the Post Correspondence Problem to $\mathsf{OVERLAP}_{\mathsf{CFG}}$. This establishes that $\mathsf{OVERLAP}_{\mathsf{CFG}}$ is undecidable.

(c) Let $\mathsf{REGULAR}_{\mathsf{CFG}} = \{\langle G\rangle : G \text{ is a CFG and } L(G) \text{ is regular}\}$. Give a reduction from the Post Correspondence Problem to $\overline{\mathsf{REGULAR}_{\mathsf{CFG}}}$. Argue that $\mathsf{REGULAR}_{\mathsf{CFG}}$ is undecidable.

Solution.

(a) Let $\Sigma = \{\sigma_1, \dots, \sigma_n\}$. Define $g : \Sigma \to \{0,1\}^*$ by $g(\sigma_i) = 0^i 1$, and extend to $h : \Sigma^* \to \{0,1\}^*$ by concatenation: $h(x_1 \cdots x_k) = g(x_1) \cdots g(x_k)$ (and $h(\varepsilon) = \varepsilon$). Given a PCP instance $P = \{(\alpha_1, \beta_1), \dots, (\alpha_m, \beta_m)\}$ over Σ, set $f(P) = \{(h(\alpha_1), h(\beta_1)), \dots, (h(\alpha_m), h(\beta_m))\}$, an instance over $\{0,1\}$; clearly f is computable.

We use two facts: (i) $h(uv) = h(u)h(v)$ for all $u, v \in \Sigma^*$. (ii) h is injective: the codewords $0^i 1$ are prefix-free, so any $h(w)$ has a unique parsing into blocks ending at each 1, hence $h(w_1) = h(w_2) \Rightarrow w_1 = w_2$.

Correctness. If $\alpha_{i_1} \cdots \alpha_{i_t} = \beta_{i_1} \cdots \beta_{i_t}$, then by (i) $h(\alpha_{i_1}) \cdots h(\alpha_{i_t}) = h(\beta_{i_1}) \cdots h(\beta_{i_t})$, so $f(P)$ is YES. Conversely, if $h(\alpha_{i_1}) \cdots h(\alpha_{i_t}) = h(\beta_{i_1}) \cdots h(\beta_{i_t})$, then by (i) $h(\alpha_{i_1} \cdots \alpha_{i_t}) = h(\beta_{i_1} \cdots \beta_{i_t})$, and by (ii) $\alpha_{i_1} \cdots \alpha_{i_t} = \beta_{i_1} \cdots \beta_{i_t}$, so P is YES.

Thus P is YES iff $f(P)$ is YES. Therefore, a decider for binary-PCP would decide PCP over arbitrary alphabets, contradicting the undecidability of PCP. Hence PCP is undecidable even over $\{0,1\}$.

(b) We reduce binary-PCP to $\mathsf{OVERLAP}_{\mathsf{CFG}}$.

Let $P = \{(\alpha_1, \beta_1), \dots, (\alpha_k, \beta_k)\}$ be an instance of binary-PCP where $\alpha_i, \beta_i \in \{0,1\}^*$. Define an alphabet $\Sigma = \{0,1\} \cup \{\sigma_1, \dots, \sigma_k\}$, where the σ_i's are fresh distinct symbols not in $\{0,1\}$.

Construct CFGs $G_1 = (\{A\}, \Sigma, R_1, A)$ and $G_2 = (\{B\}, \Sigma, R_2, B)$ with productions

$$R_1 = \{\, A \to \alpha_i A \sigma_i \;:\; i \in [k] \,\} \;\cup\; \{\, A \to \alpha_i \sigma_i \;:\; i \in [k] \,\},$$

$$R_2 = \{\, B \to \beta_i B \sigma_i \;:\; i \in [k] \,\} \;\cup\; \{\, B \to \beta_i \sigma_i \;:\; i \in [k] \,\}.$$

Let $f(P) = \langle G_1, G_2\rangle$. This mapping is computable.

Claim. P is a YES-instance of binary-PCP iff $L(G_1) \cap L(G_2) \neq \emptyset$.

($\Rightarrow$) If P has a solution $i_1 \cdots i_m$ such that $\alpha_{i_1} \cdots \alpha_{i_m} = \beta_{i_1} \cdots \beta_{i_m}$, then G_1 derives

$$A \Rightarrow \alpha_{i_1} A \sigma_{i_1} \Rightarrow \cdots \Rightarrow \alpha_{i_1} \cdots \alpha_{i_m} \sigma_{i_m} \cdots \sigma_{i_1},$$

and G_2 derives

$$B \Rightarrow \beta_{i_1} B \sigma_{i_1} \Rightarrow \cdots \Rightarrow \beta_{i_1} \cdots \beta_{i_m} \sigma_{i_m} \cdots \sigma_{i_1}.$$

Since the α- and β-concatenations are equal, the same string

$$w = \alpha_{i_1} \cdots \alpha_{i_m} \sigma_{i_m} \cdots \sigma_{i_1} = \beta_{i_1} \cdots \beta_{i_m} \sigma_{i_m} \cdots \sigma_{i_1}$$

lies in both languages, so the intersection is nonempty.

($\Leftarrow$) Suppose $w \in L(G_1) \cap L(G_2)$. By construction, any derivation in G_1 must use productions indexed by some sequence $i_1, \dots, i_m$ and yields a string of the form

$$w = \alpha_{i_1} \cdots \alpha_{i_m} \sigma_{i_m} \cdots \sigma_{i_1}.$$

Similarly, any string in $L(G_2)$ has the form

$$w = \beta_{j_1} \cdots \beta_{j_t} \sigma_{j_t} \cdots \sigma_{j_1}.$$

Because $\sigma_1, \ldots, \sigma_k$ are distinct and not in $\{0,1\}$, the suffix over $\{\sigma_i\}$ uniquely determines the index sequence. Hence $m = t$ and $i_\ell = j_\ell$ for all ℓ, so for the common w we get

$$\alpha_{i_1} \cdots \alpha_{i_m} = \beta_{i_1} \cdots \beta_{i_m},$$

which is a PCP solution. Thus P is YES.

Therefore $P \in \mathrm{PCP}_2$ (the YES-instances of binary-PCP problem) iff $f(P) \in \mathsf{OVERLAP_{CFG}}$, so $\mathrm{PCP}_2 \leq_m \mathsf{OVERLAP_{CFG}}$. Since from 4(a) we have that $\mathrm{PCP} \leq_m \mathrm{PCP}_2$, we also have that $\mathrm{PCP} \leq_m \mathsf{OVERLAP_{CFG}}$.

(c) Let $P = \{(\alpha_1, \beta_1), \ldots, (\alpha_k, \beta_k)\}$ be a PCP_2 instance, where $\alpha_i, \beta_i \in \{0,1\}^*$. Let $\Sigma = \{0,1\} \cup \{\sigma_i : i \in [k]\}$. Define the homomorphisms $h_\alpha, h_\beta : \{\sigma_i\}^* \to \{0,1\}^*$ by $h_\alpha(\sigma_i) = \alpha_i$ and $h_\beta(\sigma_i) = \beta_i$. Recall the CFGs from 4(b):

$$G_1 = (\{A\}, \Sigma, R_1, A), \quad R_1 = \{A \to \alpha_i A \sigma_i : i \in [k]\} \cup \{A \to \alpha_i \sigma_i : i \in [k]\},$$

$$G_2 = (\{B\}, \Sigma, R_2, B), \quad R_2 = \{B \to \beta_i B \sigma_i : i \in [k]\} \cup \{B \to \beta_i \sigma_i : i \in [k]\}.$$

Then

$$L(G_1) = \{\, h_\alpha(z)\, z^R : z \in \{\sigma_i\}^+ \,\}, \qquad L(G_2) = \{\, h_\beta(z)\, z^R : z \in \{\sigma_i\}^+ \,\}.$$

Hence

$$L(G_1) \cap L(G_2) = \{\, h_\alpha(z)\, z^R : z \in \{\sigma_i\}^+ \text{ and } h_\alpha(z) = h_\beta(z) \,\},$$

so P has a PCP solution iff $L(G_1) \cap L(G_2) \neq \emptyset$; we have these from 4(b). Before proceeding, we make the following claim, proof of which is provided shortly after:

Claim A. *A word $w \in \Sigma^*$ is in $\overline{L(G_1)}$ iff either:*

(A) w has some index symbol before a later bit (i.e. $\exists\ \sigma_i$ occurring before a later $0/1$), or

(B) $w = xy$ with $x \in \{0,1\}^*$ and $y \in \{\sigma_1, \ldots, \sigma_k\}^+$, but $x \neq h_\alpha(y^R)$.

Moreover, $\overline{L(G_1)}$ is context-free and a grammar for it is computable in polynomial time. Same claim and its proof holds for G_2 once we replace α with β. ◇

By Claim A, for each $i \in \{1,2\}$ we can compute in polynomial time a CFG G_i' such that $L(G_i') = \overline{L(G_i)} = \Sigma^* \setminus L(G_i)$. Let G be a CFG for the union $L(G_1') \cup L(G_2')$ (obtainable by a fresh start symbol with two start rules). Then

$$L(G) = L(G_1') \cup L(G_2') = \overline{L(G_1)} \cup \overline{L(G_2)} = \overline{L(G_1) \cap L(G_2)}.$$

Define $f(P) = \langle G \rangle$. Clearly f is computable in polynomial time.

Correctness Claim. *P is YES if and only if $\langle G \rangle \in \overline{\mathsf{REGULAR_{CFG}}}$.* ◇

Proof of Correctness Claim. ($\Leftarrow$ no-solution case) If P has no solution then $L(G_1) \cap L(G_2) = \emptyset$, so $L(G) = \overline{\emptyset} = \Sigma^*$, which is regular. Thus $\langle G \rangle \notin \overline{\mathsf{REGULAR_{CFG}}}$.

($\Rightarrow$ solution case) Assume P has a solution $z \in \{\sigma_i\}^+$ with $h_\alpha(z) = h_\beta(z)$, and set $u = h_\alpha(z) \in \{0,1\}^*$. Then for every $m \geq 1$, $w_m := u^m\, (z^R)^m \in L(G_1) \cap L(G_2)$, because $h_\alpha(z^m) = u^m = h_\beta(z^m)$. We claim $L(G_1) \cap L(G_2)$ is not regular. Suppose it were regular with pumping length p. Pick m such that $|u^m| > p$, and write $w_m = xyz$ with $|xy| \leq p$ and $|y| \geq 1$. Since the first $|u^m|$ symbols are in $\{0,1\}$, y consists only of $0/1$ symbols, so xz has the maximal σ-suffix (suffix with symbols only from $\{\sigma_1, \ldots, \sigma_k\}$) $(z^R)^m$. But any string in $L(G_1) \cap L(G_2)$ with maximal σ-suffix $(z^R)^m$ must be of the form $h_\alpha(s)s^R$ with $s^R = (z^R)^m$, i.e. $s = z^m$, hence its $0/1$-prefix must be $h_\alpha(z^m) = u^m$. Pumping down changes the $0/1$-prefix (since $|y| \geq 1$), so xz cannot equal $u^m(z^R)^m$ and therefore $xz \notin L(G_1) \cap L(G_2)$, contradicting the pumping lemma. Thus $L(G_1) \cap L(G_2)$ is nonregular.

If $L(G)$ were regular, then its complement $\overline{L(G)} = L(G_1) \cap L(G_2)$ would also be regular, a contradiction. Hence $L(G)$ is not regular and $\langle G \rangle \in \overline{\mathsf{REGULAR_{CFG}}}$. □

Therefore $\mathrm{PCP}_2 \leq_m \overline{\mathsf{REGULAR}_{\mathsf{CFG}}}$. Since PCP_2 is undecidable, $\overline{\mathsf{REGULAR}_{\mathsf{CFG}}}$ is undecidable, and so $\mathsf{REGULAR}_{\mathsf{CFG}}$ is undecidable as well.

Proof of Claim A. We build a CFG $G_1' = (V, \Sigma, R, S)$ for $\overline{L(G_1)}$ by splitting into (A) and (B). Let $V = \{S, T, U, N, E\}$, where U generates Σ^*, N generates $\{0,1\}^*$, and E generates $\{\sigma_1, \ldots, \sigma_k\}^*$. Let R consist of the following productions:

$$\begin{aligned}
&S \to U\,\sigma_i\,U\,a\,U \quad (i \in [k],\ a \in \{0,1\}) \ \mid \ T \\
&U \to xU\ (x \in \Sigma) \ \mid \ \varepsilon \\
&N \to 0N \mid 1N \mid \varepsilon \\
&E \to \sigma_i E\ (i \in [k]) \ \mid \ \varepsilon \\
&T \to \alpha_i\,T\,\sigma_i \quad (i \in [k]) \\
&T \to aN \quad (a \in \{0,1\}) \\
&T \to E\,\sigma_i \quad (i \in [k] \text{ such that } \alpha_i \neq \varepsilon) \\
&T \to w\,E\,\sigma_i \quad (i \in [k],\ w \text{ strict prefix of } \alpha_i) \\
&T \to w\,N\,E\,\sigma_i \quad (i \in [k],\ w \text{ minimal non-prefix of } \alpha_i).
\end{aligned}$$

Claim. $L(G_1') = \overline{G_1}$. ◇

Proof of Claim. Case (A): A word is *not* of the form $\{0,1\}^*\{\sigma_i\}^*$ iff it contains some σ_i followed later by a bit. Exactly those strings are generated by $S \to U\sigma_i U a U$ (arbitrary prefix, then some σ_i, then later a bit).

Case (B): $w = xy$ but $x \neq h_\alpha(y^R)$. The rules $T \to \alpha_i T \sigma_i$ "peel" a matching pair (α_i, σ_i) from the outside of a *correct* word $h_\alpha(z)z^R$. Thus any derivation that ever uses one of the *non-recursive* T-rules marks the *first* point where the required equality $x = h_\alpha(y^R)$ fails:

- $T \to wE\sigma_i$ with w strict prefix Of α_i produces x that ends too early, so x is a strict prefix of $h_\alpha(y^R)$; and then E puts an arbitrary $z \in \{\sigma_1, \ldots, \sigma_k\}^*$.
- $T \to wNE\sigma_i$ with w a minimal non-prefix produces a first mismatch symbol (after a common strict prefix); and then NE produces an arbitrary $z \in \{0,1\}^*\{\sigma_1, \ldots, \sigma_k\}^*$.
- $T \to aN$ produces strings extra bits beyond what matching would allow, i.e. x is longer than any $h_\alpha(y^R)$ compatible with the suffix.
- $T \to E\sigma_i$ produces strings whose bit-prefix is empty while the rightmost tile σ_i forces a nonempty α_i on the left when $\alpha_i \neq \varepsilon$.

Therefore every string derived from T is in (B), hence in $\overline{L(G_1)}$.

Conversely, take any $w = xy$ with $x \in \{0,1\}^*$, $y \in \{\sigma_i\}^+$, and $x \neq h_\alpha(y^R)$. Let $y^R = \sigma_{i_1} \cdots \sigma_{i_m}$. Compare x with $\alpha_{i_1} \cdots \alpha_{i_m} = h_\alpha(y^R)$ from left to right and let j be the first tile where matching fails (possibly because one side ends). Then we derive w by applying $T \Rightarrow \alpha_{i_1} T \sigma_{i_1} \Rightarrow \cdots \Rightarrow \alpha_{i_{j-1}} T \sigma_{i_{j-1}}$ and finally choosing the appropriate base rule among the four bullets above to realize that earliest failure at tile i_j. Hence $w \in L(G_1')$.

Thus $L(G_1') = \overline{L(G_1)}$, so $\overline{L(G_1)}$ is context-free. □

For each i, the set of strict prefixes of α_i and the set of minimal non-prefixes of α_i have size $O(|\alpha_i|)$ and can be listed in $O(|\alpha_i|)$ time. Hence $|R| = O(k + \sum_i |\alpha_i|)$ and G_1' is produced in time polynomial in the PCP instance size. □

□

5. A *property* is a subset $\mathcal{P} \subseteq \{\langle M \rangle : M \text{ is a Turing machine}\}$ of the set of (the encodings of) all Turing machines. A property $\mathcal{P}$ is

- *semantic* if, for any two Turing machines M_1 and M_2, $L(M_1) = L(M_2)$ implies that both $\langle M_1 \rangle, \langle M_2 \rangle$ are contained in $\mathcal{P}$ or neither are, and
- *nontrivial* if $\mathcal{P}$ is neither the empty set nor the set of (the encodings of) all Turing machines.

The *Rice's theorem* states that every nontrivial semantic property is undecidable, and the following is the (sketch of) its proof:

> Let $\mathcal{P}$ be a non-trivial semantic property. Suppose to the contrary that $\mathcal{P}$ is decidable with its decider D.
> We may assume that $\underline{(1)\ \langle M_\emptyset \rangle \notin \mathcal{P}}$, where $M_\emptyset$ is a Turing machine that rejects every string. $\underline{(2) \text{ Fix a Turing machine } M_0 \text{ such that } \langle M_0 \rangle \in \mathcal{P}}$. Now, for a Turing machine M and a string w, define a new Turing machine N as follows:
> `On input` x`, ignore the input` x `and run` M `on` w.
> - `If` M `rejects` w `then reject` x.
> - `If` M `accepts` w `then run` M_0 `on` x `and accept` x `if` M_0 `does.`
>
> Then $\underline{(3)\ D \text{ accepts } \langle N \rangle \text{ if and only if } \langle M, w \rangle \in A_{TM}}$. Thus, this gives a decider of A_{TM}, a contradiction.

Answer the following questions.

(a) Explain why we may assume that (1) holds, and also why such M_0 in (2) exists.

(b) Verify the statement in (3).

(c) By using Rice's theorem, show that the language

$$DEC_{TM} := \{\langle M \rangle : M \text{ is a Turing machine and } L(M) \text{ is decidable}\}$$

is undecidable.

(d) Does Rice's theorem apply to the following (undecidable) language? Justify your answer.

$$REJ_{TM} := \{\langle M \rangle : M \text{ is a Turing machine that rejects } \texttt{1001}.\}$$

Solution.

(a) Regarding (1), suppose that $\langle M_\emptyset \rangle \in \mathcal{P}$. Consider

$$\overline{\mathcal{P}} := \{\langle M \rangle : M \text{ is a Turing Machine}\} \setminus \mathcal{P}.$$

Clearly, $\langle M_\emptyset \rangle \notin \overline{\mathcal{P}}$. We claim that $\overline{\mathcal{P}}$ is a nontrivial semantic property. Since $\mathcal{P}$ is nontrivial, there is a Turing machine M' such that $\langle M' \rangle \notin \mathcal{P}$. Then $\langle M' \rangle \in \overline{\mathcal{P}}$; since $\langle M_\emptyset \rangle \notin \mathcal{P}$, $\overline{\mathcal{P}}$ is nontrivial. Moreover, since $\mathcal{P}$ is symantic, so is $\overline{\mathcal{P}}$.

The item (2) follows since $\mathcal{P}$ is nontrivial.

(b) Suppose M accepts w. Then by construction, N accepts x if and only if M_0 does and $L(N) = L(M_0)$. Since $\langle M_0 \rangle \in \mathcal{P}$, $\langle N \rangle \in \mathcal{P}$ as well. Now, suppose M does not accepts w. Then N rejects every string, which implies that $L(N) = \emptyset = L(M_\emptyset)$. Thus, $\langle N \rangle \notin \mathcal{P}$.

(c) We claim that DEC_{TM} is a nontrivial symantic property. That DEC_{TM} is symentic directly follows from the definition of the language. Moreover, if we let U a recognizer for A_{TM}, then $\langle M_\emptyset \rangle \in DEC_{TM}$ and $\langle U \rangle \notin DEC_{TM}$. This shows that DEC_{TM} is nontrivial. Therefore, we conclude that DEC_{TM} is undecidable by Rice's theorem.

(d) The answer is **No**. To see why, let M_∞ be a Turing machine that does not halt for every input. Then $L(M_\emptyset) = L(M_\infty) = \emptyset$. However, $\langle M_\emptyset \rangle \in REJ_{TM}$ and $\langle M_\infty \rangle \notin REJ_{TM}$, so REJ_{TM} is not a semantic property. Therefore, we cannot apply Rice's theorem to REJ_{TM}.

$\square$

6. Answer the following questions.

> VERTEX COVER
>
> **Input:** An undirected graph $G = (V, E)$ and a non-negative integer k.
>
> **Question:** Is there a set $X \subseteq V(G)$ of size at most k such that every edge in G has at least one endpoint in X?

> FEEDBACK ARC SET
>
> **Input:** A directed graph G and a non-negative integer k.
>
> **Question:** Does there exist a set X of at most k arcs of G such that $G - X$ is acyclic, i.e., it does not contain any directed cycles?

> DOMINATING SET
>
> **Input:** An undirected graph $G = (V, E)$ and a non-negative integer k.
>
> **Question:** Does there exist a set $X \subseteq V$ of size at most k such that for every vertex $v \in V$, either $v \in X$ or v has a neighbor in X?

(a) Show that VERTEX COVER is polynomial-time many-one reducible to FEEDBACK ARC SET.

(b) Show that DOMINATING SET is polynomial-time many-one reducible to BIPARTITE DOMINATING SET. Here, BIPARTITE DOMINATING SET is the same as DOMINATING SET, but it takes an (undirected) bipartite graph and a non-negative integer k as input.

Solution.

(a) We construct a polynomial time computable function f such that for all $\langle G, k\rangle$,

$$\langle G, k\rangle \in \text{VERTEX COVER} \Leftrightarrow f(\langle G, k\rangle) = \langle G' = (V', E'), k'\rangle \in \text{FEEDBACK ARC SET}.$$

Let $\langle G = (V, E), k\rangle$ be an instance of VERTEX COVER. We let k' be the same as k and the reduction f generates G' as follows.

(i) For each vertex $v \in V$, put two vertices v_1, v_2 in V', and add an arc (v_1, v_2) in E', which will be called a *vertex-arc*.

(ii) For every edge $e = \{u, v\} \in E$, put two arcs (u_2, v_1) and (v_2, u_1) in E', which will be called *edge-arcs*.

The reduction is polynomial time computable as $2|V|$ vertices and $2|E| + |V|$ edges are added in G'.

We now show the correctness of the above reduction.

($\Leftarrow$): Assume $\langle G' = (V', E'), k'\rangle \in$ FEEDBACK ARC SET. Then there is a feedback arc set $X' \subseteq E'$ of size at most k in G'. If E' contains an edge-arc (u_2, v_1) with $u \neq v$, put the vertex-arc (v_1, v_2) in E' instead. As every $v_1 \in V'$ has only one outgoing arc (v_1, v_2), every cycle containing (u_2, v_1) must contain (v_1, v_2). Hence, this new set X', which consists of only vertex-arcs, is also a feedback arc set of size at most k in G'.

We claim that $X = \{v : (v_1, v_2) \in E'\}$ is a vertex cover of size at most k in G. Suppose that there is an edge $(x, y) \in E$ in G such that neither x nor y is in X. This means that neither (x_1, x_2) nor (y_1, y_2) is in X'. Then there is a cycle $x_1x_2y_1y_2x_1$ in G', where no arcs are hit by X', which is a contradiction to that X' is a feedback arc set in G'.

($\Rightarrow$): Assume $\langle G, k\rangle \in$ VERTEX COVER. Then there exists a vertex cover $X \subseteq V(G)$ of size at most k. We claim that $X' = \{(v_1, v_2) : v \in X\}$ is a feedback arc set of size at most k in G'. Suppose that there is a cycle C in $G' - X'$. Due to construction, every vertex-arc is followed by an edge-arc in every cycle in G'. So, C must contain at least two vertex-arcs (x_1, x_2) and (y_1, y_2) such that x and y are adjacent in G. Since both x and y are not in X, it is a contradiction to that X is a vertex cover in G.

(b) We construct a polynomial time computable function f such that for all $\langle G, k\rangle$,

$$\langle G, k\rangle \in \text{DOMINATING SET} \Leftrightarrow f(\langle G, k\rangle) = \langle G' = (V', E'), k'\rangle \in \text{BIPARTITE DOMINATING SET}.$$

Let $\langle G = (V, E), k\rangle$ be an instance of DOMINATING SET. We let $k' = k + 1$, and $G' = (V_1 \cup V_2, E')$ is constructed as follows.

(i) Add a copy of every vertex $v \in V$ to both V_1 and V_2. Denote by v_1 and v_2 the copies of v in V_1 and V_2, respectively.

(ii) For every edge $\{u, v\} \in E$, add edges $\{u_1, v_2\}$ and $\{v_1, u_2\}$. Additionally, add the edge $\{u_1, u_2\}$ for every vertex $u \in V$.

(iii) Add a vertex z_1 to V_1 and a vertex z_2 to V_2. Put the edge $\{z_1, z_2\}$ and also the edge $\{u_1, z_2\}$ for all $u \in V$.

The reduction can be computed in time $\mathcal{O}(|V| + |E|)$.

We now show the correctness of the above reduction.

($\Rightarrow$): Assume $\langle G, k\rangle \in$ DOMINATING SET. Then there exists a dominating set $X \subseteq V$ in G with $|X| \leq k$. We claim that

$$X' := \{v_1 \in V_1 \mid v \in X\} \cup \{z_2\}$$

is a dominating set in G'. By construction, z_2 dominates z_1 and all vertices in V_1. For each vertex $u \in V \setminus X$, there exists a vertex $v \in X$ with $\{u, v\} \in E$. Due to construction, the edge $\{v_1, u_2\} \in E'$ exists and thus u_2 is dominated. Furthermore, for every vertex $u \in X$, the edge $\{u_1, u_2\} \in E'$ ensures that the copies of dominating vertices in V_2 are also dominated. Hence, X' dominates all vertices in G'.

($\Leftarrow$): Assume $\langle G', k + 1\rangle \in$ BIPARTITE DOMINATING SET. Then there is a dominating set X' in G' with $|X'| \leq k + 1$. If $z_1 \in X'$, replace it with z_2. If $z_1 \notin X'$, then z_2 must already be in X' as otherwise z_1 would not be dominated. Hence, all vertices in $V_1 \cup \{z_1, z_2\}$ are dominated by z_2.

Define the vertex set

$$X := \{v \in V \mid v_1 \in X' \text{ or } v_2 \in X'\}.$$

We claim that X is a dominating set in G. Let $u \in V \setminus X$. Since u_2 is dominated by X' and $u_2 \notin X'$, there exists $v_1 \in X'$ with $\{v_1, u_2\} \in E'$. Thus, $v \in X$ and, by construction, $\{u, v\} \in E$. Hence, u is dominated in G.

$\square$

7. Consider the following variant of k-SAT for an integer $k \geq 1$.

> Not-All-Equal k-SAT
>
> **Input:** A set V of Boolean variables and a collection $\mathcal{C}$ of clauses each of which consists of k distinct variables.
>
> **Question:** Is there a truth assignment for V so that each clause in $\mathcal{C}$ has at least one true literal and at least one false literal?

For example, let $V_1 = \{x_1, x_2, x_3, x_4\}$ and $\mathcal{C}_1 = \{\{x_1, x_2, \neg x_2\}, \{x_1, x_3, \neg x_4\}, \{\neg x_1, \neg x_2, x_4\}\}$. Then $(V_1, \mathcal{C}_1)$ is a Yes-instance for Not-All-Equal 3-SAT since we can take $(x_1, x_2, x_3, x_4) = (\mathtt{T}, \mathtt{T}, \mathtt{F}, \mathtt{T})$, where `T` and `F` stand for `True` and `False` respectively. However, the truth assignment $(x_1, x_2, x_3, x_4) = (\mathtt{F}, \mathtt{F}, \mathtt{T}, \mathtt{T})$ does not imply that $(V_1, \mathcal{C}_1)$ is a Yes-instance since the clause $\{\neg x_1, \neg x_2, x_4\}$ consists of only true literals.

(a) Prove that Not-All-Equal k-SAT problem is NP-complete for each $k \geq 3$. (Hint: Give a reduction from k-SAT to Not-All-Equal $(k+1)$-SAT.)

(b) Assuming $\mathsf{P} \neq \mathsf{NP}$, prove or disprove that Not-All-Equal 2-SAT problem is NP-hard.

(c) Show that the (special instance of) Set Splitting problem is NP-hard by giving a polynomial-time many-one reduction from Not-All-Equal 3-SAT.

> Set Splitting
>
> **Input:** A finite set V and a family $\mathcal{F} \subseteq 2^V$ where $|F| \leq 3$ for each $F \in \mathcal{F}$.
>
> **Question:** Is there a partition (V_1, V_2) of V such that $F \cap V_i \neq \emptyset$ for each $i \in [2]$ and $F \in \mathcal{F}$?

Remark. To prove (b), suggest a polynomial-time many-one reduction; to disprove it, you can either provide (the idea of) a polynomial-time algorithm or suggest a polynomial-time many-one reduction.

Solution. We also write Not-All-Equal k-SAT by NAE k-SAT for simplicity. We say a truth assignment for $(V, \mathcal{C})$ is *NAE-satisfying* if every clause in $\mathcal{C}$ has at least one true literal and at least one false literal, and say $(V, \mathcal{C})$ is *NAE-satisfiable* if it has an NAE-satisfying assignment.

(a) Clearly, the problem NAE k-SAT is in NP for each $k \geq 3$. To show the hardness, we show that NAE k-SAT is NP-hard when $k \geq 4$. Fix an integer $k \geq 3$ and let $(V, \mathcal{C})$ be an instance for k-SAT. We introduce a new variable $x \notin V$ and let $V' := V \cup \{x\}$ and $\mathcal{C}' := \{C \cup \{x\} : C \in \mathcal{C}\}$. We claim that $(V, \mathcal{C})$ is a Yes-instance for k-SAT if and only if $(V', \mathcal{C}')$ is a Yes-instance for NAE $(k+1)$-SAT. If $\varphi : V \to \{\mathtt{T}, \mathtt{F}\}$ is a satisfying assignment for $(V, \mathcal{C})$, then it is easy to check that the assignment $\varphi : V' \to \{\mathtt{T}, \mathtt{F}\}$ defined by

$$\varphi'(v) = \begin{cases} \varphi(v) & v \in V, \\ \mathtt{F} & v = x \end{cases}$$

is a NAE-satisfying assignment for $(V', \mathcal{C}')$. Conversly, let $\psi : V' \to \{\mathtt{T}, \mathtt{F}\}$ is an NAE-satisfying assignment for $(V', \mathcal{C}')$. If $\psi(x) = \mathtt{F}$, then the restriction of ψ onto V gives a satisfying assignment for $(V, \mathcal{C})$. Otherwise, let $\psi' : V \to \{\mathtt{T}, \mathtt{F}\}$ be an assignment defined as

$$\psi'(v) = \begin{cases} \mathtt{T} & \psi(v) = \mathtt{F}, \\ \mathtt{F} & \psi(v) = \mathtt{T}. \end{cases}$$

Then ψ' is a satisfying assignment for $(V, \mathcal{C})$. Since k-SAT is NP-hard for each $k \geq 3$, the reduction shows that NAE k-SAT is NP-hard for each $k \geq 4$.

Unfortunately, this approach does not show that NAE 3-SAT is NP-hard since 2-SAT is in P. Instead, we use a standard reduction argument from NAE 4-SAT as follows: Let $(V, \mathcal{C})$ be an instance for NAE 4-SAT, where $\mathcal{C} = \{C_1, \dots, C_m\}$ and $C_i = \{\ell_i^1, \ell_i^2, \ell_i^3, \ell_i^4\}$ where each ℓ_i^j is either $v \in V$ or its negation. Now, let $z_1, \dots, z_m$ be new variables not in V; let $V' = V \cup \{z_1, \dots, z_m\}$ and let $\mathcal{C}' = \{C_i^+, C_i^- : i \in [m]\}$, where

$$C_i^+ \coloneqq \{\ell_i^1, \ell_i^2, z_i\}, \quad C_i^- \coloneqq \{\neg z_i, \ell_i^3, \ell_i^4\}.$$

Then it is routine to check that $(V, \mathcal{C})$ is NAE-satisfiable if and only if $(V', \mathcal{C}')$ is NAE-satisfiable. This shows that NAE 3-SAT is NP-hard.

(b) The problem NAE 2-SAT is **polynomial-time solvable**, that is, it is **not NP-hard**. To see this, we give a reduction from NAE 2-SAT to 2-Coloring, where the latter problem has a polynomial-time algorithm. Let $(V, \mathcal{C})$ be an instance for NAE 2-SAT. Let G be a graph where the vertices of G are the variables in V and their negations, and two vertices are adjacent in G if and only if either they form a clause in $\mathcal{C}$ or one is the negation of the other.

Observe that $|V(G)| = 2|V|$ and $|E(G)| = |V| + |\mathcal{C}|$. We claim that $(V, \mathcal{C})$ is NAE-satisfiable if and only if G is 2-colorable. Let $\varphi : V \to \{\mathtt{T}, \mathtt{F}\}$ be an NAE-satisfying assignment for $(V, \mathcal{C})$. Now, color the vertices of G according to the truth value of the corresponding literal. That is, we consider the function $f : V(G) \to \{\mathtt{T}, \mathtt{F}\}$ where

$$f(v) = \begin{cases} \mathtt{T} & \varphi(v) = \mathtt{T} \text{ and } v \text{ is positive, or } \varphi(v) = \mathtt{F} \text{ and } v \text{ is negative;} \\ \mathtt{F} & \varphi(v) = \mathtt{F} \text{ and } v \text{ is positive, or } \varphi(v) = \mathtt{T} \text{ and } v \text{ is negative.} \end{cases}$$

If there is an edge $uv \in E(G)$ for which $f(u) = f(v)$, this implies that there is a clause C in $\mathcal{C}$ where two literals in C is assigned the same truth value, a contradiction. Thus, G is 2-colorable when $(V, \mathcal{C})$ is NAE-satisfiable. Indeed, the converse can be shown by using the same argument: Given a 2-coloring $f : V(G) \to \{1, 2\}$ of G, we define a truth assignment $\varphi : V \to \{\mathtt{T}, \mathtt{F}\}$ where

$$\varphi(x) = \begin{cases} \mathtt{T} & f(v) = 1, \\ \mathtt{F} & f(v) = 2. \end{cases}$$

Then it is routine to check that φ gives an NAE-satisfying assignment for $(V, \mathcal{C})$. This shows that there is a polynomial-time many-to-one reduction from NAE 2-SAT to 2-Coloring. Therefore, we conclude that NAE 2-SAT is polynomial-time solvable.

(c) The idea is essentially same as given in (b); we consider V_1 and V_2 as the set of true and false literals respectively. Given an instance $(V, \mathcal{C})$ for NAE 3-SAT, define a set system $(V', \mathcal{F})$ as follows:

$$V' \coloneqq V \cup \{\neg v : v \in V\};$$
$$\mathcal{F} \coloneqq \{\{v, \neg v\} : v \in V\} \cup \mathcal{C}.$$

Then $|V'| = 2|V|$ and $|\mathcal{F}| = |V| + |\mathcal{C}|$. By applying a similar argument given in (b), one can check that $(V, \mathcal{C})$ is NAE-satisfiable if and only if $(V', \mathcal{F})$ is a Yes-instance for Set Splitting. Therefore, we conclude that Set Splitting is NP-hard.

$\square$