

Problems 2-4 are from old exams. **Try to do this set without help (if you can), as it's good practice for the next exam.**

1. (a) (Sipser 5.18) Show that PCP is still undecidable when $\Sigma = \{0, 1\}$.

Reduce from PCP by encoding each symbol that appears in binary (i.e., encode each $\#, q_i$, etc.) You must be careful that each code-word is the same length so that, for instance, two code words together won't form a third.

Formally, Let I be an instance of PCP over alphabet $\Sigma = a_1, a_2, a_3, \dots, a_n$, convert it to an instance, I' , over alphabet $\{0, 1\}$ by encoding each symbol to a distinct $\lceil \log(n) \rceil$ -bit string. It should be clear that I has a solution if and only if I' does.

- (b) (HU 8.15, Sipser 5.17) Show PCP is decidable when $\Sigma = \{0\}$.

Notice that the instance of PCP has a solution if and only if there are dominos $\left[\frac{w_1}{x_1} \right]$ and $\left[\frac{w_2}{x_2} \right]$ such that $|w_1| \geq |x_1|$ and $|w_2| \leq |x_2|$, a fact easily checked by a TM. Without being overly formal, the following observations should suffice:

- First, notice that if $x_1 = 000$ and $w_1 = 00000$, that you might as well change x_1 to ϵ and w_1 to 00 : The only thing that's relevant is the difference between their lengths.
- Second, if $w_1 = 0^i$, $x_2 = 0^j$, $w_2 = x_1 = \epsilon$, then $(w_1)^i(w_2)^j = (x_1)^i(x_2)^j$.

2. Design a Turing machine, M , with input alphabet $\{0, 1\}$ to compute the function $f(x) = xp_x$. For example, $f(011) = 0110$ and $f(010) = 0101$.

$M = (\{q_0, \dots, q_7, q_A, q_R\}, \{0, 1\}, \{0, 1, B\}, \delta, q_0, q_A, q_R)$. Transition in ()'s are optional.

δ	0	1	B	Comments
q_0	q_1, B, R	q_4, B, R	$(q_7, 0, R)$	blank out and remember leading digit
q_1	$q_1, 0, R$	$q_2, 1, R$	$q_5, 0, L$	leading 0, even number of 1's so far
q_2	$q_2, 0, R$	$q_1, 1, R$	$q_5, 1, L$	leading 0, odd number of 1's so far
q_3	$q_3, 0, R$	$q_4, 1, R$	$q_6, 0, L$	leading 1, even number of 1's so far
q_4	$q_4, 0, R$	$q_3, 1, R$	$q_6, 1, L$	leading 1, odd number of 1's so far
q_5	$q_5, 0, L$	$q_5, 1, L$	$q_7, 0, R$	go to start and write leading 0
q_6	$q_6, 0, L$	$q_6, 1, L$	$q_7, 1, R$	go to start and write leading 1
q_7	$q_8, 0, L$	$q_8, 1, L$	(q_8, B, L)	jog left and accept
q_A				the accept state

3. Define

$$L = \{\langle M_1, M_2 \rangle : L(M_1) \cap L(M_2) \neq \emptyset\}$$

(I.e., $\langle M_1, M_2 \rangle \in L$ if there is a string, ω , that M_1 and M_2 both accept.) Like any language, we know L falls into one of four possibilities: (1) L is decidable (2) L is recognizable but not decidable (3) \overline{L} recognizable but not decidable (4) neither L nor \overline{L} are recognizable.

- Which category does L fall into?
- Prove your answer.

L is recognizable but not decidable. To see that L is not decidable, construct a machine, M^* , accepting Σ^* . Then, $f(\langle M \rangle) = \langle M, M^* \rangle$ reduces $\overline{L_\emptyset} \times L$. To see that L is r.e., a machine, M_L , simulates both M_1 and M_2 on w_i for j steps for all possible (i, j) by dovetailing. If both accept some w_i , accept. $L(M_L) = L$, since some string w is accepted by both machines if and only if an (i, j) exists making M_L accept.

A is recognizable but not decidable. To see that it's recognizable, a machine, M , accepting A dovetails over all pairs (w, t) , running $M_1(w)$ and $M_2(w)$ for t steps. If both accept, accept. To see that L is undecidable, reduce $A_{\text{TM}} \times A$. (Note that reducing $A_{\text{TM}} \times \overline{A}$ will not work since we've just shown A is recognizable.) Choose the reduction $f(\langle M, w \rangle) = \langle M', M' \rangle$, where $M'(x)$ ignores it's input and simulates $M(w)$. Clearly, f is computable. Furthermore, f is a reduces $A_{\text{TM}} \times A$ since

$$\begin{cases} L(M') \cap L(M') = \Sigma^* & , \text{ if } w \in L(M) \\ L(M') \cap L(M') = \emptyset & , \text{ if } w \notin L(M) \end{cases}$$

and hence $\langle M, w \rangle \in A_{\text{TM}} \iff f(\langle M, w \rangle) = \langle M', M' \rangle \in A$.

4. Of the following three problems, one is in NP, one is in P, and one is not known to be in either class. Specify which is in each category, and give a brief but **clear** explanation. You need not go into detail about any algorithm, but you should explain clearly how non-determinism is (or cannot be) used.

- $L = \{(G, k) : \text{Graph } G \text{ has exactly } 10 \text{ } k\text{-vertex cliques}\}$
- $L = \{(G, k) : \text{Graph } G \text{ has exactly } k \text{ } 10\text{-vertex cliques}\}$
- $L = \{(G, k) : \text{Graph } G \text{ has at least } k \text{ } k\text{-vertex cliques}\}$

- I doubt it's in NP or co-NP, since I know of no short general proof that $(G, k) \in L$ or $(G, k) \notin L$.
- $\in P$: There are only $\binom{|V|}{10} \leq |V|^{10}$ cliques of size 10. So checking them all takes polynomial time.
- $\in NP$: A non-deterministic algorithm can guess k cliques, and verify they are present in G .