CS401 - Problem Set 5

CW: Write up no. 4

1. The $\leq_p$ symbol tells us something about the *time* required to convert one type of problem to another. We used this symbol in our definition of **PSPACE**-HARD. It seems like it might make more sense to worry about the space used by our conversion procedure, rather than time. In this problem, you'll investigate what would happen if we were to use a space-based reduction definition.

   Suppose we instead defined $L' \in$ **PSPACE**-HARD if for every $L \in$ **PSPACE**, $L \leq_{ps} L'$, where $\leq_{ps}$ means there exists a function f computable in polynomial *space* such that $f(x) \in L'$ iff $x \in L$. Consider the following: let $L \in$ **PSPACE**, and let L^* be any language except for the empty set, or the set of all binary strings. Then there is a string w such that $w \in L^*$ and a string y such that $y \notin L^*$. Consider the function $f_L : L \to L^*$:

$$f_L(x) = \begin{cases} w & \text{if } x \in L \\ y & \text{if } x \notin L. \end{cases} \tag{1}$$

   (a) Explain why f_L is computable in polynomial space.

   (b) Explain why this shows it is not a good idea to use polynomial space reductions to define the concept of **PSPACE**-HARD.

   (c) More generally, why is it good to require the conversion be done using less powerful resources than the resources of the class for which you are defining a HARD version? For example, we used polynomial time conversion to define **NP**-HARD, and where polynomial time is less powerful than **NP**. Then we used polynomial time conversion to define **PSPACE**-HARD, where again polynomial time is less powerful than **PSPACE**.

2. Crumbling Graph is a game played by two players on a directed graph $G = (V, E)$ with one vertex labelled as the starting vertex. Player 1 starts with a token at the starting vertex. Then Player 1 and Player 2 alternate taking turns, where Player 1 can moved their token from their current vertex along one outgoing edge to a new vertex. Player 2 can remove one outgoing edge from the vertex where Player 1's token currently sits. The game ends when Player 1 can no longer move. Let L be the following language:

$$L = \{\langle G, s, T\rangle : \text{Player 1 can make at least } T \text{ moves in the Crumbling Graph} \tag{2}$$
$$\text{game on the graph } G \text{ with starting vertex } s\}. \tag{3}$$

   Describe a recursive algorithm to decide if an input x is in L that uses as little space as possible. You don't need to prove your algorithm is correct, but it should be correct!

3. **[Moved to next week's pset!]** As in class, let $\varphi_i(C_1, C_2) = 1$ (true) if there is a path of length at most 2^i from configuration C_1 to configuration C_2 in the configuration graph of a TM, and 0 otherwise. Explain why

$$\begin{aligned}\varphi_i(C, C') &\equiv \exists C^* : \varphi_{i-1}(C, C^*) \wedge \varphi_{i-1}(C^*, C') \\ &\equiv \exists C^* \forall D, D' \left(\big((D = C) \wedge (D' = C^*)\big) \vee \big((D = C^*) \wedge (D' = C')\big) \right) \rightarrow \varphi_{i-1}(D, D').\end{aligned} \tag{4}$$

4. Prove **NP** $\subseteq$**PSPACE**.