CS 682 (Spring 2001) - Solutions to Assignment 5

(1) If NP $\neq$ PSPACE prove that there exists under $\leq_m^p$-reductions PSPACE-incomplete languages in PSPACE $\setminus$ NP.

Proof. Let P_i, N_j, D_k be r.e. lists of all deterministic polynomial space machines, nondeterministic polynomial time machines, deterministic polynomial time machines, respectively.

Assume $NP \neq$ PSPACE and let

$$\Delta_1 = \{P_i | L(P_i) \in \text{PSPACE} \setminus \text{NP}\}, \qquad \Delta_2 = \{P_i | L(P_i) \text{ is PSPACE-complete}\}$$

Notice that $\Delta_2 \subseteq \Delta_1$ since every PSPACE-complete language is not in NP.

First we show that Δ_1 is Π_2-hard. To this aim, we construct a many-one reduction from $M_{\Sigma^*} = \{M_i | L(M_i) = \Sigma^*\}$, which is known to be Π_2-complete, to Δ_1. Let C be a PSPACE-complete set (we know that such a set exists), and suppose it can be computed using $p_C(|x|)$ space.

Let f be recursive such that $P_{f(i)}$ does the following on input x:

1. $P_{f(i)}$ sets up a bounded work tape of size $p_C(|x|)$.
2. It simulates M_i on all inputs, in lexicographical order, up to existing resources. Let y be the first input not accepted by M_i in this simulation.
3. Starting with $j = 1$ and going in order, possibly up to $j = |y|$, it checks for disagreement between $L(N_j)$ and $L(P_{f(i)})$, withing existing resources. To do that, all possible computations of N_j on each input $< x$ are simulated, and the existence of an accepting computation is checked against the computation of $P_{f(i)}$ on the same input (note that $P_{f(i)}$ here simulates itself on smaller inputs). If for some j a disagreement is not found, $P_{f(i)}$ accepts x iff $x \in C$ (which can be decided using the available tape). If a disagreement is found for j, $j + 1$ is then tested. If a disagreement has been found for all $1 \leq j \leq |y|$, x is rejected.

Now, suppose that $M_i \in M_{\Sigma^*}$. Then disagreements between N_j and $P_{f(i)}$ will eventually (i.e. for large enough values of x) be found for all j (note that if a disagreement for some j is found in a run for a certain input, a disagreement for that j will also be found all runs for bigger inputs, since at least the same resources are available). To show this, first notice that disagreements for all j will be checked eventually, since for large enough x there will be enough resources to show that M_i accepts all inputs up to some input of length j. Then, suppose that for some j a disagreement is never found. In that case, all large enough inputs x (enough to check disagreement for j) will be accepted iff they are in C. This means that $L(P_{f(i)})$ differs finitely from C. But then it is PSPACE-complete as well, and must disagree with $L(N_j)$. Since this disagreement will be found eventually, we have a contradiction. Consequently, $L(P_{f(i)}) \neq L(N_j)$ for all j, and $P_{f(i)} \in \Delta_1$.

On the other hand, suppose that $M_i \notin M_{\Sigma^*}$, and let y be the first string which M_i does not accept. For all large enough x, disagreements for $1 \leq j \leq |y|$, and only for those j, will be tested. By a similar argument to the above, for large enough x disagreements for all $1 \leq j \leq |y|$ will be found. Therefore all of those x will be rejected, making $L(P_{f(i)})$ finite (and in particular NP), and therefore $P_{f(i)} \notin \Delta_1$.

Thus Δ_1 is Π_2-hard. Next, we show that Δ_2 is Σ_2 First we show that there exists an r.e. list Γ of names for all PSPACE-complete languages. Let f be recursive such that $P_{f(i,j)}$ does the following on input x:

1. $P_{f(i,j)}$ sets up a bounded work tape of size $p_{i,j}(|x|)$, where $p_{i,j}$ is a polynomial of greater degree than the space polynomial of P_i, the time polynomial of D_j, and the polynomial p_C.

2. It checks for all inputs up to x if D_j reduces C to $L(P_i)$, namely, whether $y \in C \iff D_j(y) \in L(P_i)$ for all $y \leq x$. If this is the case, x is accepted iff $x \in L(P_i)$. If not, x is accepted iff $x \in C$.

Let $\Gamma = \{P_{f(i,j)} | i, j \in \omega\}$. Γ is r.e. since f is recursive. Note that $L(P_{f(i,j)})$ is PSPACE-complete for all i, j, since either the reduction fails for some x, and then $L(P_{f(i,j)})$ differs finitely from C, or the reduction holds for all x, and then $L(P_{f(i,j)}) = L(P_i)$, to which C can be reduced. Conversely, if $L(P_i)$ is PSPACE-complete, there exists a j such that D_j reduces C to $L(P_i)$. Then $L(P_i) = L(P_{f(i,j)})$ has a name in our set.

Now notice that

$$P_i \in \Delta_2 \qquad \iff \qquad (\exists k)(P_k \in \Gamma \wedge L(P_i) = L(P_k))$$

and this is a Σ_2 definition of Δ_2 since $P_k \in \Gamma$ is Σ_1 and $L(P_i) = L(P_k)$ is Π_1.

Finally, since Δ_1 is Π_2-hard and Δ_2 is Σ_2, the sets cannot be equal. Therefore $\Delta_1 \setminus \Delta_2$ is nonempty, i.e. there is some $L(P_i)$ in PSPACE $\setminus$ NP which is incomplete. ■

(2) Let NPR denote nondeterministic polynomial-time machines that can ask only polynomially many different questions to its oracle, counting the queries over all possible nondeterministic computation paths for a given input. Let NPR^A denote the family of languages accepted by such machines with oracle A. Prove that

$$\mathrm{P} = \mathrm{NP} \qquad \iff \qquad (\forall A)\left(\mathrm{P}^A = \mathrm{NPR}^A\right)$$

Proof. One direction is easy: if $\mathrm{P}^A = \mathrm{NPR}^A$ for all oracles A, then in particular $\mathrm{P}^\emptyset = \mathrm{NPR}^\emptyset$. But $\mathrm{P}^\emptyset = \mathrm{P}$ since any poly-time computation with the empty oracle can be simulated by a poly-time computation without any oracle, just by replacing each call to the oracle with an automatic "no" answer. Similarly $\mathrm{NPR}^\emptyset = \mathrm{NP}$, and we get $\mathrm{P} = \mathrm{NP}$.

Conversely, suppose that $\mathrm{P} = \mathrm{NP}$. For any oracle A it is easily shown that $\mathrm{P}^A \subseteq \mathrm{NPR}^A$: Every deterministic poly-time computation in A has only poly-many queries to A, and this can be simulated as a nondeterministic poly-time computation with only one path, the original deterministic computation, therefore with only poly-many queries (over "all paths").

It remains to show that $\mathrm{NPR}^A \subseteq \mathrm{P}^A$. Let R be an NPR machine, with polynomial bound p (on computation time and on number of queries). We want to give a deterministic poly-time machine D such that $L(D^A) = L(R^A)$. Let S be the set of finite sets s of strings x_i and corresponding answers a_i ("yes" or "no"). Those finite sets are easily encodable as strings, so we can treat S as a set of strings. The following sets are in NP, and therefore in P:

1. The set T_1 of all pairs $(x, s) \in \Sigma^* \times S$ such that the run of R on input x, using s as a partial oracle, has a path which makes a query not in s.

2. The set T_2 of all triplets $(x, s, y) \in \Sigma^* \times S \times \Sigma^*$ such that the run of R on input x, using s as a partial oracle, has a path which makes a query not in s with prefix y.

It is not hard to see why. If (x, s) is in T_1, we can guess a path of R on x, and verify that it is a legal computation path, using s to answer queries, that ends up asking a query not is s, all of this

in polynomial time. Similarly, if (x, s, y) is in T_2, we can guess the witnessing path and verify it in polynomial time.

D^A acts on input x as follows:

1. Set s to the empty set.
2. If $(x, s) \notin T_1$, go to step 6.
3. Set y to the empty string.
4. If for some $a \in \Sigma$ $(x, s, ya) \in T_2$, set y to ya and repeat step 4.
5. Query A with y, add y and the answer to s, go to step 2.
6. s has all the needed information. Simulate R on x with oracle s (this is in NP, therefore in P), and return answer.

The algorithm should be quite self-explanatory, so no formal proof will be presented here.

Intuitively, D^A collects into s all possible queries in the run of R^A on x, querying each string when it is found. Since there are at most $p(|x|)$ queries, each of length at most $p(|x|)$, s is of size $O(p(|x|)^2)$ (allowing for the answers and encoding markers). The algorithm advances query by query, deciding T_1 for each query, and letter by letter within each query, possibly deciding T_2 for every letter in Σ. This gives a polynomial runtime. ■

(3) Prove that there exists a recursive oracle A such that

$$\mathrm{P}^A \neq \mathrm{NP}^A \neq \mathrm{PSPACE}^A$$

The rough idea is to consider two problems: EX^A, which contains strings w that have a witness of length exactly $|w|$ in A, and ALL^A, which contains strings w such that all strings of length $|w|$ are in A, i.e. that have exponentially many witnesses. These two problems are obviously in NP^A and PSPACE^A, respectively. We will construct A inductively such that its witnesses "diagonalize" against all machines (deterministic for EX^A, non-deterministic for ALL^A). That is, we explicitly add or remove witnesses to prove the given machine wrong. We must apply some care to ensure that witnesses added later will not change the behavior of previously considered machines. In order to achieve this, we "spread" the witnesses far apart, i.e. we add large gaps in between them. The details follow below.

Let $D_1, D_2, \ldots$ and $N_1, N_2, \ldots$ be enumerations of all deterministic and non-deterministic machines with polynomial runtime. For each j, let $p_{2j-1}(n)$ be a polynomial runtime bound for D_j, and $p_{2j}(n)$ a polynomial runtime bound for N_j. Furthermore, let n_i be numbers such that $2^n > p_i(n)$ for all $n \geq n_i$ and $n_{i+1} > p_i(n_i)$. Such n_i can certainly be found inductively by just choosing "large enough" numbers at each step. To start the induction, set $A_0 := \emptyset$.

If $i \geq 1$ is odd, let $j = (i+1)/2$ be the index of the deterministic machine under consideration. Let S_i be the set of strings queried by $D_j^{A_{i-1}}$ on input 1^{n_i} (i.e. D_j uses oracle A_{i-1}), and s_i any string of length exactly n_i not in S_i. Such an s_i exists, because $|S_i| \leq p_i(n_i) < 2^{n_i} \leq |\{s \mid |s| = n_i\}|$. If $D_j^{A_{i-1}}$ accepts 1^{n_i}, let $A_i := A_{i-1}$. Otherwise, let $A_i := A_{i-1} \cup \{s_i\}$.

If $i \geq 1$ is even, let $j = i/2$, and $A'_{i-1} := A_{i-1} \cup \{w \mid n_i \leq |w| < n_{i+1}\}$, i.e. add all strings with lengths between n_i and n_{i+1}. If $N_j^{A'_{i-1}}$ rejects input 1^{n_i}, let $A_i := A'_{i-1}$. Otherwise, let W_i be the

set of strings queried during some accepting computation of $N_j^{A'_{i-1}}$ on 1^{n_i}. As $|W_i| \leq p_i(n_i) < 2^{n_i} \leq |\{w \mid |w| = n_i\}|$, there is some w_i of length exactly n_i that is not in W_i. For this w_i, set $A_i := A'_{i-1} \setminus \{w_i\}$. Finally, define $A := \bigcup_i A_i$.

Proof. First, notice that A is recursive, since the construction can be simulated with a TM, and in each stage the only strings added to A are ones of length greater than those already in it. To decide whether a given string is in A, all we have to do is to simulate the construction, until some string of greater length is added to A. If the given string is not already in A at that time, it will never be.

To prove that this construction actually works, note first that $A_0 \subseteq A_1 \subseteq A_2 \subseteq \ldots$, i.e. no strings are ever removed. Also note that any string added in stage i is longer than all strings in A_{i-1}.

Let us first prove that $\mathrm{P}^A \neq \mathrm{NP}^A$. Consider the problem $\mathrm{EX} := \{1^n \mid \exists w : |w| = n \text{ and } w \in A\}$. Clearly, $\mathrm{EX} \in \mathrm{NP}^A$, because a non-deterministic machine can guess w in polynomial time and query the oracle to confirm that $w \in A$. Now, assume for contradiction that $\mathrm{EX} \in \mathrm{P}^A$, i.e. that EX is accepted by some machine D_j in deterministic time $p_{2j-1}(n)$. Let $i = 2j - 1$ be the stage of the construction that dealt with D_j. Consider the string 1^{n_i}, with n_i defined above, and let S be the set of all strings queried by D_j^A on input 1^{n_i}. As all strings added in phases later then i have length at least $n_{i+1} > p_i(n_i)$, D_j cannot have queried any of them. And by definition, s_i was not queried either, so $S \cap A = S \cap A_{i-1}$. Since D_j is deterministic and obtains the same answers on all queries from oracles A and A_{i-1}, the outcome of the computation must be the same for both oracles. However, if $D_i^{A_{i-1}}$ accepted 1^{n_i}, then by construction, no w with $|w| = n_i$ is in A, otherwise, there is one. In either case, $D_j^A(1^{n_i})$ accepts if and only if $1^{n_i} \notin \mathrm{EX}$, a contradiction. Hence, $\mathrm{EX} \notin \mathrm{P}^A$.

To prove $\mathrm{NP}^A \neq \mathrm{PSPACE}^A$ consider the problem $\mathrm{ALL} := \{1^n \mid \forall w : |w| = n \text{ implies } w \in A\}$. $\mathrm{ALL} \in \mathrm{PSPACE}$, because a machine can enumerate all w with $|w| = n$ one by one in polynomial space and query the oracle. Assume for contradiction that $\mathrm{ALL} \in \mathrm{NP}^A$, i.e. ALL is accepted by N_j in time $q_{2j}(n)$. Let $i = 2j$, and consider the string 1^{n_i}. If $N_j^{A'_{i-1}}$ does not accept 1^{n_i}, then neither does N_j^A. This is because $A \cap \{w \mid |w| \leq p_i(n_i)\} = A'_{i-1} \cap \{w \mid |w| \leq p_i(n_i)\}$, i.e. all queries are answered the same way by oracles A and A'_{i-1} (since only strings up to length $p_i(n_i)$ can be tested). Now, by construction, $A_i = A'_{i-1}$, so in particular $\{w \mid |w| = n_i\} \subseteq A$, and $1^{n_i} \in \mathrm{ALL}$. Otherwise, consider the accepting computation of $N_j^{A'_{i-1}}$ on 1^{n_i} that was considered in the construction. Since w_i was never queried, this is also an accepting computation for $N_j^{A_i}$. As above, we have that $A \cap \{w \mid |w| \leq p_i(n_i)\} = A_i \cap \{w \mid |w| \leq p_i(n_i)\}$, i.e. all queries by N_j have the same result for oracles A and A_i. Therefore, N_j must also accept 1^{n_i}. However, we explicitly removed w_i, so $w_i \notin A, |w_i| = n_i$, implying that $1^{n_i} \notin \mathrm{ALL}$. Taking both cases together, we get that N_j^A accepts 1^{n_i} iff $1^{n_i} \notin \mathrm{ALL}$, a contradiction to N_j accepting ALL. Hence, $\mathrm{ALL} \notin \mathrm{NP}^A$, completing the proof. ■