Assignment 5 solutions

Let $\Sigma = \{0, 1\}$ and assume that all languages and classes of languages considered in this assignment are over the alphabet Σ. Also assume that the word "polynomial" means "non-constant polynomial with nonnegative integer coefficients."

1. Suppose that $A \subseteq \Sigma^*$ is NP-complete, $B \subseteq \Sigma^*$ is in P, $A \cap B = \varnothing$, and $A \cup B \neq \Sigma^*$. Prove that $A \cup B$ is NP-complete.

Solution. First let us show that $A \cup B$ is in NP. Consider the following NTM. On input $x \in \Sigma^*$: if $x \in B$ (which can be checked deterministically in polynomial time), then accept; otherwise run a polynomial-time NTM for A on x and return its result. This NTM runs in polynomial-time and decides $A \cup B$.

Now it is enough to show that there is a polynomial-time mapping reduction from A to $A \cup B$, so let us do that. Choose an arbitrary $y \in \Sigma^* \setminus (A \cup B)$. Because $B \in \mathrm{P}$, the function f defined as

$$f(x) = \begin{cases} y & \text{if } x \in B \\ x & \text{if } x \notin B \end{cases} \tag{1}$$

is polynomial-time computable. We have:

- if $x \in A$, then $x \notin B$ and $f(x) = x \in A \cup B$;
- if $x \notin A$, then
  - either $x \in B$, in which case $f(x) = y \notin A \cup B$,
  - or $x \notin B$ and thus $f(x) = x \notin A \cup B$.

Hence, $A \leq_m^P (A \cup B)$ and $A \cup B$ is NP-complete.

2. For every language $A \subseteq \Sigma^*$ and function $f : \mathbb{N} \to \mathbb{N}$ satisfying $f(n) \geq n+1$ for all $n \in \mathbb{N}$, define a new language

$$\mathrm{pad}(A, f) = \left\{ x01^{f(|x|)-|x|-1} \;:\; x \in A \right\}.$$

The way to think about the language $\mathrm{pad}(A, f)$ is that it "pads" strings $x \in A$ with a tail of the form 01^m, which has the effect of artificially increasing the length of strings in the language: for each $x \in A$ there is a unique corresponding string $x01^m \in \mathrm{pad}(A, f)$ whose length is $f(|x|)$.

(a) Prove that for any polynomial p with $p(n) \geq n + 1$ we have

$$A \in \mathrm{DTIME}\left(2^{p(n)}\right) \quad \Leftrightarrow \quad \mathrm{pad}(A, p) \in \mathrm{DTIME}(2^n).$$

(b) Prove that for any polynomial p with $p(n) \geq n + 1$ we have

$$A \in \mathrm{PSPACE} \quad \Leftrightarrow \quad \mathrm{pad}(A, p) \in \mathrm{PSPACE}.$$

(c) Use the time-hierarchy theorem, along with parts (a) and (b), to prove

$$\mathrm{PSPACE} \neq \bigcup_{k \geq 1} \mathrm{DTIME}\left(2^{k \cdot n}\right).$$

(d) Repeat (b) and (c) with NP in place of PSPACE.

Solution. Let M be a (deterministic or nondeterministic) Turing machine and let p be a polynomial with $p(n) \geq n+1$. Consider the following Turing machine $P_{M,p}$. On input $y \in \Sigma^*$:

1. if $y \in 1^*$, reject;
2. obtain $x \in \Sigma^*$ and $m \in \mathbb{N}$ such that $y = x01^m$;
3. compute $p(|x|)$ and, if $|y| \neq p(|x|)$, reject;
4. run M on x and return the result.

One can see that, if M decides A, then $P_{M,p}$ decides $\mathrm{pad}(A,p)$. Also note that steps 1 through 3 take time and space polynomial in $|y|$. Consider another Turing machine $R_{M,p}$ which on input input $y \in \Sigma^*$ does the following:

1. compute $m = p(|x|) - |x| - 1$ and set $y = x01^m$;
2. run M on y and return the result.

If M decides $\mathrm{pad}(A,p)$, then $R_{M,p}$ decides A. Step 1 requires time and space polynomial in $|x|$.

(a) If $A \in$ DTIME $\left(2^{p(n)}\right)$, then there exists a DTM M_A which for every $x \in \Sigma^*$ decides if $x \in A$ in time $O(2^{p(|x|)})$. The time cost of steps 1 through 3 in the DTM $P_{M_A,p}$ is negligible, therefore for every $y \in \Sigma^*$ it decides if $y \in \mathrm{pad}(A,p)$ in time $O(2^{p(|x|)})$, where $p(|x|) = |y|$, and $\mathrm{pad}(A,p) \in$ DTIME(2^n).

If $\mathrm{pad}(A,p) \in$ DTIME(2^n), then there exists a DTM $M_{\mathrm{pad}(A,p)}$ which for every $y \in \Sigma^*$ decides if $y \in \mathrm{pad}(A,p)$ in time $O(2^{|y|})$. Hence, for every $x \in \Sigma^*$ the DTM $R_{M_{\mathrm{pad}(A,p)},p}$ decides if $x \in A$ in time $O(2^{p(|x|)})$, and thus $A \in$ DTIME $\left(2^{p(n)}\right)$.

(b) If $A \in$ PSPACE, then there exists a DTM M_A deciding A and running in polynomial space. All the steps of the DTM $P_{M_A,p}$ require polynomial space and, because it decides $\mathrm{pad}(A,p)$, $\mathrm{pad}(A,p) \in$ PSPACE. In the same manner we show that $\mathrm{pad}(A,p) \in$ PSPACE implies $A \in$ PSPACE.

(c) Suppose the contrary:

$$\text{PSPACE} = \bigcup_{k\geq 1} \text{DTIME}\left(2^{k\cdot n}\right),$$

which also implies DTIME $(2^n) \subseteq$ PSPACE. By the time-hierarchy theorem, we have

$$\bigcup_{k\geq 1} \text{DTIME}\left(2^{k\cdot n}\right) \subsetneq \text{DTIME}\left(2^{n^2}\right) \subsetneq \text{DTIME}\left(2^{n^3}\right).$$

Choose an arbitrary language $A \in \text{DTIME}\left(2^{n^3}\right) \setminus \text{DTIME}\left(2^{n^2}\right)$, for which we have

$$A \notin \bigcup_{k\geq 1} \text{DTIME}\left(2^{k\cdot n}\right) = \text{PSPACE},$$

and let $p(n) = n^3 + 1$. Then $A \in \text{DTIME}\left(2^{n^3}\right)$ implies $\mathrm{pad}(A,p) \in$ DTIME(2^n) (part (a)), which implies $\mathrm{pad}(A,p) \in$ PSPACE (because DTIME$(2^n) \subseteq$ PSPACE), which then implies $A \in$ PSPACE (part (b)). So we have $A \notin$ PSPACE and $A \in$ PSPACE, which is a contradiction.

(d) We proceed exactly as in part (b). That is, if $A \in \mathrm{NP}$, then there exists a NTM M_A deciding A and running in polynomial time. All the steps of the NTM $P_{M_A,p}$ require polynomial time and, because it decides $\mathrm{pad}(A,p)$, $\mathrm{pad}(A,p) \in \mathrm{NP}$. Similarly: $\mathrm{pad}(A,p) \in \mathrm{NP} \Rightarrow A \in \mathrm{NP}$. Now when we have shown this, in the same way as in part (c) we get

$$\mathrm{NP} \neq \bigcup_{k \geq 1} \mathrm{DTIME}\left(2^{k \cdot n}\right).$$

3. Prove that the following two statements are equivalent (i.e., that each one implies the other).

Statement 1: For every language $B \in \mathrm{P}$ and every polynomial p, there exists a polynomial-time computable function $f : \Sigma^* \to \Sigma^*$ such that

$$\left(\exists y \in \Sigma^{p(|x|)}\right) [\langle x, y \rangle \in B] \quad \Leftrightarrow \quad \langle x, f(x) \rangle \in B$$

Statement 2: $\mathrm{P} = \mathrm{NP}$.

Solution. Assume Statement 1. Let A be an arbitrary language in NP, which means that there exist a polynomial p and a polynomial-time decidable language B such that

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

By the equivalence above and Statement 1, there exist a polynomial-time computable function $f : \Sigma^* \to \Sigma^*$ such that

$$x \in A \quad \Leftrightarrow \quad \left(\exists y \in \Sigma^{p(|x|)}\right) [\langle x, y \rangle \in B] \quad \Leftrightarrow \quad \langle x, f(x) \rangle \in B.$$

Because $g(x) = \langle x, f(x) \rangle$ is a polynomial-time computable function, $A \leq_m^P B$, and $B \in \mathrm{P}$ implies $A \in \mathrm{P}$. Hence $\mathrm{P} = \mathrm{NP}$ and Statement 2 holds.

Now, assume Statement 2, that is, $\mathrm{P} = \mathrm{NP}$. Let B be an arbitrary language P and p be an arbitrary polynomial. Consider the following NTM M. On input $\langle x, v \rangle \in \Sigma^* \times \Sigma^*$:

1. calculate $p(|x|)$ and, if $|v| > p(|x|)$, reject;
2. if there is $w \in \Sigma^{p(|x|)-|v|}$ (choose w nondeterministically) such that $\langle x, vw \rangle \in B$, then accept;
3. (otherwise) reject.

M runs in nondeterministic polynomial-time, therefore $L(M) \in \mathrm{NP} = \mathrm{P}$ and there exists a DTM M' which decides $L(M)$ in deterministic polynomial-time. Now we define f to be a function computed by the following DTM H. On input $x \in \Sigma^*$:

1. if $\langle x, \varepsilon \rangle \notin L(M')$, return any $u \in \Sigma^{p(|x|)}$ (say, $u = 0^{p(|x|)}$);
2. initialize $y := \epsilon$;
3. while $|y| < p(|x|)$ do:
   (a) if $\langle x, y0 \rangle \in L(M')$, set $y := y0$;
   (b) otherwise set $y := y1$;

4. return y.

Because for any string z of length polynomial in $|x|$ we can check deterministically in polynomial time if $\langle x, z\rangle \in L(M')$, the algorithm above runs in polynomial time, and thus f is polynomial-time computable. If there is no $y \in \Sigma^{p(|x|)}$ such that $\langle x, y\rangle \in B$, then $\langle x, \varepsilon\rangle \notin L(M')$ and therefore $\langle x, f(x)\rangle \notin B$. While, if there is such y, then H gives us first such y lexicographically. That is, H constructs such y bit by bit, one bit per iteration of step 3, starting from the leftmost bit. Therefore Statement 1 holds.

4. [Bonus problem] Prove that if there exists a set $S \subseteq \mathbb{N}$ for which the language $\{1^n : n \in S\}$ is NP-complete, then P = NP. (Hint: thinking about the previous problem may help with this one.)

Solution. Suppose there exists a set S for which the language $L = \{1^n : n \in S\}$ is NP-complete. Then, there exists a polynomial-time mapping reduction f from SAT to L. Without loss of generality we can assume that $\mathrm{range}(f) \subseteq 1^*$ (either $S = \mathbb{N}$, which is a trivial case, or we can concatenate f with another polynomial-time mapping which maps all the strings in $\Sigma^* \setminus 1^*$ to a fixed $u \in 1^* \setminus L$). For any Boolean formula (in conjunctive normal form) ψ with at least one variable and $a \in \{0, 1\}$ let ψ_a be a Boolean formula obtained from ψ by fixing the value of the first variable to be a. Also without loss of generality, for the encoding of ψ_a is at most as long as the encoding of ψ and this encoding can be obtained in polynomial-time (given the encoding of ψ and a).

We will use f to construct a deterministic polynomial-time algorithm solving SAT. Suppose that as an input we have a Boolean formula $\phi(x_1, \ldots, x_m)$. Consider a complete binary tree T of height m such that each its node is labeled by some Boolean formula ψ and the string $f(\psi) \in 1^*$ in the following way

- the root of T has the label $\langle \phi, f(\phi)\rangle$;
- if a node has label $\langle \mathrm{true}, f(\mathrm{true})\rangle$ or $\langle \mathrm{false}, f(\mathrm{false})\rangle$, then all its children has that same label;
- if $\langle \psi, f(\psi)\rangle$ is the label of a node such that ψ is a Boolean formula with at least one variable, then its children have labels $\langle \psi_0, f(\psi_0)\rangle$ and $\langle \psi_1, f(\psi_1)\rangle$.

The formula ϕ is satisfiable if and only if there is a node in T labeled by $\langle \psi, f(\psi)\rangle$ such that $f(\psi) \in L$, which is the case if and only if there is a node labeled by $\langle \mathrm{true}, f(\mathrm{true})\rangle$.

The problem is that there are exponentially many nodes, and we cannot in polynomial-time traverse all of them, therefore we must somehow decide if T contains $\langle \psi, f(\psi)\rangle$ satisfying $f(\psi) \in L$ by traversing only polynomial number of nodes. Notice that for any node labeled by $\langle \psi, f(\psi)\rangle$, if $f(\psi) \notin L$, then there is no descendant of this node labeled by $\langle \xi, f(\xi)\rangle$ such that $f(\xi) \in L$.

This is where we take advantage of the fact that $\mathrm{range}(f) \subseteq 1^*$ and f is polynomial-time computable, which implies that there exists a polynomial p such that $|f(\phi)| \leq p(|\langle \phi\rangle|)$. Due to our assumption that "smaller Boolean formulas have shorter encodings", for every label $\langle \psi, f(\psi)\rangle$ of a node in T we have $f(\psi) \in \{1^n : n \in [0 \mathinner{..} p(|\langle \phi\rangle|)]\}$, which means that in our tree T we have at most $p(|\langle \phi\rangle|) + 1$ different values of the function f. Taking everything above into account, the algorithm M deciding SAT is as follows. On input $\langle \phi\rangle$:

1. initialize $A_L := \{f(\mathrm{true})\}$ and $A_{\overline{L}} := \{f(\mathrm{false})\}$ (the values of f known to be in L and $\overline{L}$, respectively);
2. run R on $\langle \phi\rangle$ and accept if and only if R accepts,

where we define R to be a recursive function (having access to the global variables A_L and $A_{\overline{L}}$) which on the encoding of a Boolean formula $\langle\psi\rangle$ as an input does the following:

1. if $f(\psi) \in A_L$, accept;
2. if $f(\psi) \in A_{\overline{L}}$, reject;
3. run R on $\langle\psi_0\rangle$:

   if R accepts $\langle\psi_0\rangle$, then update $A_L := A_L \cup \{f(\psi)\}$ and accept;
4. if $f(\psi) \in A_{\overline{L}}$, reject;
5. run R on $\langle\psi_1\rangle$:

   if R accepts $\langle\psi_1\rangle$, then update $A_L := A_L \cup \{f(\psi)\}$ and accept;

   otherwise update $A_{\overline{L}} := A_{\overline{L}} \cup \{f(\psi)\}$ and reject.

It is quite easy to see that the algorithm M works correctly.

Regarding the running time, we can see that M does depth-first traversal of the tree T. If the algorithm has either just started or just updated either A_L or $A_{\overline{L}}$, then the following happens. It backs up the tree till it reaches a node $\langle\psi, f(\psi)\rangle$ such that $f(\psi) \notin A_L \cup A_{\overline{L}}$, and then proceeds to yet unvisited child of this node. One can see that from this point the algorithm will never make two consecutive moves up the tree T and, therefore, visit more than two nodes in any given depth of T before updating either A_L or $A_{\overline{L}}$. Since there are only polynomially many values the function f can take, the algorithm M will visit no more than polynomially many nodes of T, and run in polynomial time. Hence, M solves SAT, an NP-complete problem, in deterministic polynomial-time, and P = NP.

Alternate Solution. (Prepared by John Watrous.) I had in mind a slightly different solution to the bonus problem, based on the same basic idea as Ansis's solution but with different details. Suppose that p is a polynomial and $B \in \mathrm{P}$. We will specify a polynomial-time computable function f such that

$$\left(\exists y \in \Sigma^{p(|x|)}\right)[\langle x, y\rangle \in B] \quad \Leftrightarrow \quad \langle x, f(x)\rangle \in B.$$

Given that p and B are chosen arbitrarily, this implies $\mathrm{P} = \mathrm{NP}$ by problem 3.

Define a language

$$C = \{\langle x, u\rangle \,:\, \text{there exists } v \in \Sigma^* \text{ such that } |uv| = p(|x|) \text{ and } \langle x, uv\rangle \in B\}$$

The language C is clearly in NP, so under the assumption in the problem statement we have that there exists a Karp reduction g from C to $\{1^n \,:\, n \in S\}$. We will use this reduction to "grow" a polynomial-size collection of possible candidate outputs $f(x)$ from a given input x, and then select from this collection by testing membership in B. Consider the following DTM:

On input $x \in \Sigma^*$:
1. Set $Q_0 \leftarrow \{\varepsilon\}$.
2. For $k \leftarrow 1, \ldots, p(|x|)$ do the following:
   a. Set $Q_k \leftarrow \varnothing$.
   b. For each string $u \in Q_{k-1}$ and each $b \in \Sigma$:
      If $g(\langle x, ub\rangle) \in L(1^*)$ and $g(\langle x, ub\rangle) \neq g(\langle x, w\rangle)$ for every string $w \in Q_k$, then add ub to Q_k.
3. Search through the strings $y \in Q_{p(|x|)}$. If one is found such that $\langle x, y\rangle \in B$, then output y, and otherwise output $1^{p(|x|)}$.

Let $x \in \Sigma^*$ be a given input string, and suppose first that there exists a string $y \in \Sigma^{p(|x|)}$ such that $\langle x, y\rangle \in B$. We must argue that our function f satisfies $\langle x, f(x)\rangle \in B$. We claim that the following property holds at the end of each iteration of the loop in step 2:

> If there exists a string $u \in Q_{k-1}$ such that $\langle x, u\rangle \in C$, then there must exist a string $w \in Q_k$ such that $\langle x, w\rangle \in C$.

The reason for this is simple: if $\langle x, u\rangle \in C$ and $u \in Q_{k-1}$, then it must hold that $\langle x, ub\rangle \in C$ for at least one choice of $b \in \Sigma$ (possibly both). If $\langle x, ub\rangle \in C$, then $g(\langle x, ub\rangle) \in \{1^n \,:\, n \in S\} \subseteq L(1^*)$. If we add ub to Q_k, then the claimed property is obviously satisfied. If we don't, then there must already exist a string $w \in Q_k$ such that $g(\langle x, w\rangle) = g(\langle x, ub\rangle) \in \{1^n \,:\, n \in S\}$ and therefore $\langle x, w\rangle \in C$. So, the property holds either way.

Now, given that the above property holds for each k, and given that the string $\varepsilon \in Q_0$ satisfies $\langle x, \varepsilon\rangle \in C$ (as we have assumed that there exists a string $y \in \Sigma^{p(|x|)}$ such that $\langle x, y\rangle \in B$), it follows that there must exist a string $y \in Q_{p(|x|)} \subseteq \Sigma^{p(|x|)}$ such that $\langle x, y\rangle \in C$, which is equivalent to $\langle x, y\rangle \in B$ when $|y| = p(|x|)$. Searching through $Q_{p(|x|)}$ and testing each candidate will obviously find such a string as required.

If, on the other hand, it holds that there are no strings $y \in \Sigma^{p(|x|)}$ such that $\langle x, y\rangle \in B$, then our algorithm clearly does not find such a string: it then outputs $1^{p(|x|)}$ (or any other string of length $p(|x|)$ would do), which satisfies $\langle x, 1^{p(|x|)}\rangle \notin B$ as required.

It remains to observe that our DTM runs in polynomial time. Given that g is polynomial-time computable, it must hold that $|g(\langle x, w\rangle)|$ is bounded by a polynomial for every choice of w with $|w| \leq p(|x|)$. There can therefore be at most polynomially many distinct outputs $g(\langle x, w\rangle) \in L(1^*)$, and therefore at most polynomially many strings in each set Q_k. This implies that our DTM runs in polynomial time as required.