Math 225B–Final Project

Edward Kim

May 10, 2016

Problem 1

Proof: Following the hint provided, suppose we had a function f such that f maps x to the least stage such that the approximation to A is correct on all numbers n such that $n < x$. Since A is a Δ_2^0 subset of $\mathbb{N}$, it follows that it can be computed from the halting set $\emptyset'$. We recall the halting set can compute the least stage s where:

$$n \in A \Leftrightarrow (\forall t \geq s)\ g(n,t) = 1$$
$$n \notin A \Leftrightarrow (\forall t \geq s)\ g(n,t) = 0$$

Thus, $f(x)$ can retrieve this information directly from A for all inputs $n \leq x$ and it follows that f is computable from A. Now suppose, we have a function g computable from a set X such that g dominates f on all inputs $n \in \mathbb{N}$. However, we observe that g computes A by the following scheme: we compute $A(n)$ by running the program that approximates A at input n for $g(n+1)$ stages. In other words, let $y = f(n+1)$, then by definition of f:

$$n \in A \Leftrightarrow (\forall t \geq y)\ g(n,t) = 1$$
$$n \notin A \Leftrightarrow (\forall t \geq y)\ g(n,t) = 0$$

From $g(n+1) \geq f(n+1)$, it follows that running the program for $g(n+1)$ stages will result in the correct output. Thus, A can be computed by X through g. $\square$

Problem 2

Proof: We provide a priority construction where the negative requirements are injured finitely often. We give the requirements below:

$$P_e : \{e\}^D \neq G$$

and

$$N_e : \{e\}^G \neq D$$

For the positive requirements, we hold on a potential witness x and observe if $\{e\}^D(x) \downarrow = 0$ on stage s. If so, we enumerate x into G_{s+1}. If not, no action is required since either $\{e\}^D \downarrow \neq 0$ or $\{e\}^D(x) \uparrow$. Hence, we say that P_e requires attention at stage s if for some potential witness x

$$\{e\}_s^{D_s}(x) \downarrow = 0$$

The negative requirements will require us to restrain values from being enumerated into G. Thus, we shall define the following length function:

$$L(e,s) = \max\{x : \forall n \leq x \ \{e\}_s^{G_s}(n) \downarrow = D_s(n)\}$$

We similarly define the restraint function $r(e,s)$ which prohibits any values smaller than the largest value used in the computation of all $\{e\}_s^{G_s}(n) \downarrow$ for some stage s where $n \leq L(e,s)$. Lastly, we give the priority list:

$$P_1, N_1, P_2, N_2, ...$$

and outline the construction:

Stage 0: Let $G_0 = \emptyset$. For each positive requirement P_e, let the first picked potential witness be e. For each negative requirement N_e, let the restraint functions be initialized $r(e,0) = -1$.
Stage s: We find the least positive requirement P_i that requires attention at this stage. We enumerate the witness x into G_s. We require all postive requirements P_j where $j \geq i$ to find new witnesses x_j where $x_j \geq r(e,s)$ for $e \leq i$ and $x_j \notin G_s$.

We see that all of the positive requirements are eventually met. All there is left to verify is that our negative requirements are met.

Claim: $\{e\}^G \neq D$ for all e.

Proof: Suppose that $\{e\}^G = D$ for some e. Then we see that $\lim_s L(e,s) = \infty$. Thus, for all input n, there must exist a stage s such that $L(e,s) > n$ and N_e is never injured after stage s. By definition of L, $\{e\}_s^G(n) \downarrow = D(n)$. Thus, D is recursive which contradicts our original assumption.$\square$
$\square$

Problem 3

Proof: We use a priority argument where we enumerate W one element x per stage ($x \in W_{s+1} - W_s$). We only enumerate the element x into one of the sets A and B and the elements must respect the negative requirements of higher priority. We outline the positive and negative criterion below:

$$P : \text{For every } x \in W_{s+1} - W_s,\ x \in A_s \text{ or } x \in B_s$$
$$N_{[e,A]} : \{e\}^A \neq D$$
$$N_{[e,B]} : \{e\}^B \neq D$$

Let us define $L^A(e,s)$ as we did in Problem 2. That is $L^A(e,s)$ the largest input x in which the computations of $\{e\}_s^A$ and D_s agree on inputs $n \leq x$. We define the restraint function $r^A(e,s)$ simiarly as in Problem 2 as well. Functions $L^B(e,s)$ and $r^B(e,s)$ are defined similarly with B in place of A.

Stage 0: $A = \emptyset$, $B = \emptyset$
Stage s: Let $x \in W_{s+1} - W_s$. Choose the highest negative priority $N_{[e,i]}$ where $i = A, B$ and enumerate x on into the other set. In other words, if $i = A$, then enumerate x into B_{s+1} and if $i = B$, then enumerate x into A_{s+1}.

We prove that every negative requirement is injured finitely many times and that r^A and r^B converge through induction. For any $[e,i]$, let us assume that after stage s', all priorities $[g,j] < [e,i]$ are never injured and $r^A(g)$ and $r^B(g)$ converge for stages $s \geq s'$. Let us find a stage t where $N_{[e,i]}$ is never injured after stage t. To find this, we have to ensure that B does not enumerate anymore elements after this stage into any of the restraints of higher priority. This ensures that no elements enumerated from B can injure $N_{[e,i]}$ since all of the higher priority restraints are completely settled. Take a value $r > \max\{r^j(g) \mid [g,j] < [e,i]\}$. Let t be the stage such that $B_t \upharpoonright r = B \upharpoonright r$. As mentioned previously, $N_{[e,i]}$ cannot be injured after this point and it follows that $r^A(e)$ and $r^B(e)$ converge through the proof of the claim in Problem 2. $\square$

Problem 4

Proof: We will first conclude that every branch in the aforementioned tree must split into two infinite branches. We shall proceed by contradiction and assume that there exists an isolated branch B in T. We recursively compute B as follows: let $\sigma \subset B$. We can compute the next branch B will take by computing strings of T longer than $|\sigma|$ and extending σ. Let $\sigma * 0$ or $\sigma * 1$ represent strings σ concatenated with 0 and 1 respectively. Since B is an isolated branch, there must exist a level L such that all strings extending one of the branches must eventually terminate within a finite amount of stages. The next step of the path will traverse the branch which has not terminated within the level L. However, this implies that f is a recursive path in T which contradicts our original assumption about the infinite paths in T. Thus, every branch in T must split into two infinite branches. Thus, given a set $X \subseteq \mathbb{N}$, we can trace a path P in T such that if a real number z_i codes a set recursive in X, then $z_i \not\subset P$ (follows from the observation that only a countable number of these types of real numbers can exist). Hence, this path P is an infinite path such that $X \not\geq_T P$. We give a simple argument for the uncountability of the infinite paths in T. Suppose that there were a countable number of infinite paths in T, then we can code these paths into a countable set of real numbers X'. However, by the result above, there must exist an infinte path P' in T such that $X' \not\geq_T P'$. However, by construction, P' cannot be any of the infinite paths in X' which contradicts the definition of X'. Hence, the uncountability of the infinite paths in T follows. $\square$

Problem 5

Proof: The intuition behind the proof of this proposition is to create two sets A and B such that $\{e\}^A \neq X$ and $\{e\}^B \neq X$ for all e but the join $A \oplus B$ encodes enough information about X to ensure that $X \leq_T A \oplus B$. We provide the following finite extension construction by constructing the characteristic function of A and B through finite initial segments $\{f_s\}_{s\in\omega}$ and $\{g_s\}_{s\in\omega}$ respectively:

<u>Stage 0:</u> $f_{A_0} = \emptyset$, $f_{B_0} = \emptyset$.

<u>Stage $s+1 = 2e$:</u> We shall ensure that $\{e\}^A \neq X$. We proceed by finite extension and query the following Σ^0_1 sentence to a $\emptyset'$-oracle :

$$(\exists\sigma)(\exists\tau)(\exists x)(\exists s)(\exists v)(\exists w)[f_s \subset \sigma, \tau \ \& \ \{e\}^{\sigma}_s(x) = y \ \& \ \{e\}^{\tau}_s(x) = z \ \& \ y \neq z] \tag{1}$$

This ensures that at least one of the finite extensions σ, τ disagrees with X on witness x. We then use a X-oracle to find if $\{e\}^{\sigma}_s(x) \neq X(x)$ or $\{e\}^{\tau}_s(x) \neq X(x)$. Without the loss of generality, we then let σ be the extension that disagrees with X. In addition, we extend g_s by copying the $|\sigma| - |f_s|)$ bits that were added to f_s. This can be recursively done by appending $\sigma(n)$ for $|f_s| + 1 \leq n \leq |\sigma|$ to the end of g_s in order. We then code the necessary information into A and B to code X. Let $n = |\sigma|$. If $X(e) = 0$, then let $f_{s+1}(n) = 0$ and $g_{s+1}(n) = 1$. If $X(e) = 1$, then let $f_{s+1}(n) = 1$ and $g_{s+1}(n) = 0$. By construction, we see that both f_{s+1} and g_{s+1} are of the same length.

Stage $s + 1 = 2e + 1$: We shall now ensure that $\{e\}^B \neq X$. Repeat the above steps except replace f with g and A with B.

Claim: $A \oplus B \leq_T X$
Proof: This claim follows from our construction procedure above. Each nonzero stage queries a $\emptyset'$-oracle and a X-oracle. By our assumption that $X \geq_T \emptyset'$, we see that the construction is done X-recursively. $\square$

Claim: $X \leq_T A \oplus B$
Proof: We see that, by construction, the differences between A and B code the characteristic function for X. We can recursively check for any difference between A and B. If a difference is detected at index n, we consider two cases. If $A(n) = 1$, then $X(n) = 1$. If $B(n) = 1$, then $X(n) = 0$. $\square$

$\square$