CS 682 (Spring 2001) - Solutions to Assignment 6

(1) DEF: Let $A \subseteq \Sigma^*$ and $B \subseteq \Gamma^*$. We say that $A$ is recursive in $B$ iff there exists a total TM $M$ such that $A = L(M^B)$.

Prove that $\{M_i|\overline{M_i(-)}\}$ is recursive in

MING = $\{G_i|(\forall j)(j < i \rightarrow G_i \not\equiv G_j)\}$.

For extra credit show that MING is or is not an $\leq_m$-r.e. complete set.

Proof. First, notice that minimization of context-free grammars is recursive in MING. To minimize a given CFG $G_i$:

1. Check whether $G_i \in$ MING. If this is the case, we are done.
2. If not, make a list of all minimal $G_j$ with $j < i$. This is easily done by querying MING with each $G_j$ with $j < i$.
3. Simultaneously, search for disagreements between $L(G_i)$ and $L(G_j)$ for every $G_j$ on the list, until the search terminates for all but one $G_j$. This will happen, since exactly one grammar on the list is equivalent to $G_i$, and that is the minimized grammar of $G_i$.
4. Return $G_j$.

From this it follows that equivalence of context-free grammars is also recursive in MING. Given CFGs $G_i, G_j$, minimize both to $G'_i, G'_j$. Then $G_i \equiv G_j$ iff $G'_i = G'_j$.

Now we’re ready to decide $\Delta = \{M_i|\overline{M_i(\epsilon)}\}$ (with oracle MING). Let $f, g$ be recursive such that $M_{f(i)}(x)$ simulates $M_i(x)$ if $x = \epsilon$, accepting if $M_i$ halts, and rejects $x$ otherwise, and such that $L(G_{g(i)}) = \text{VALCOM}(M_i)$. Let $G_T$ be a grammar such that $L(G_T) = \Sigma^*$. Then

$M_i \in \Delta \iff M_i(\epsilon) \uparrow \iff \epsilon \notin L(M_{f(i)}) \iff L(G_{g(f(i))}) = \Sigma^* \iff G_{g(f(i))} \equiv G_T$

and the latter is recursive in MING.

(2) DEF: A set $C \subseteq \Sigma^*$ is sparse iff there is a $k$ such that for all $n$

$|\{x||x| \leq n \text{ and } x \in C\}| \leq n^k + k$.

Prove that there are no sparse complete sets, under $\leq^p_m$-reductions, for

EXSPACE = $\bigcup_{k \geq 1} \text{SPACE} \left[2^{n^k}\right]$.

Proof. Let $S$ be a sparse set, $k$ such that $|\{x||x| \leq n \text{ and } x \in S\}| \leq n^k + k$. We show that $S$ is not EXSPACE-complete by constructing an EXSPACE machine $M$ such that $L(M)$ is not poly-time reducible to $S$.

Let $D_l$ be the $l$-th poly-time machine, $p_l$ be the polynomial bound on the runtime of $D_l$. It is safe to assume that both can be computed in $\text{SPACE}[2^l]$. Let $M$ do the following on input $x$ of length $n$:
1. Set a bounded working tape of length $2^n$.
2. If $x$ is not of the form $l\#y$, reject $x$.
3. Compute $w = D_l(x)$.
4. For each string $l\#z < \text{lex} x$ of length $n$, compute $D_l(l\#z)$. If one of the values is $w$ reject $x$, otherwise accept.
5. If at any time during steps 2-4 the tape runs out, reject $x$.

Notice that for $x = l\#y$ of length $n$, completion of steps 2,3,4 would need at most $C \cdot p_l(n)$ space, since the result of $D_l$ on strings of length $n$ is not longer than $p_l(n)$. The algorithm needs $p_l(n)$ space to remember $w$, $p_l(n)$ more to simulate $D_l$ on the other strings, one at a time, and little more for bookkeeping. $C = 17$ should be more than enough.

First, $L(M)$ is in SPACE[$2^n$] by construction (steps 1 and 5), therefore in EXSPACE. Now, for a given $D_l$ let $n$ be such that

$$2^n > C \cdot p_l(n) \quad \text{and} \quad |\Sigma|^{n-l-1} > (p_l(n))^k + k.$$ 

All large enough $n$ would clearly do. Note that on any $x$ of length $n$, $M$ would complete its computation, without running out of tape, because of the left inequality.

There are two cases:

Case 1. $D_l$ is 1-1 on the set $A = \{l\#z||z| = n-l-1\}$. In that case, $M$ will accept all strings in $A$, since step 4 would never yield a matching value. However, $D_l$ maps $A$ to a set of $|\Sigma|^{n-l-1}$ strings of length at most $p_l(n)$. Since $S$ has, at most, only $(p_l(n))^k + k$ many strings of that length (less by the right inequality above), for some string $u \in A \subseteq L(M)$ it must be the case that $D_l(u) \notin S$. Therefore $D_l$ cannot reduce $L(M)$ to $S$.

Case 2. $D_l$ is not 1-1 on $A$. Then there is a string $w$ and a subset $B \subseteq A$ of cardinality $> 1$ such that $D_l(u) = w$ iff $u \in B$. Let $u_0$ be the $\text{lex}$-least element of $B$, $u_1$ be another element of $B$, different from $u_0$. $M$ accepts $u_0$ because $D_l(u) \neq w$ for every $u < \text{lex} u_0$ in $A$. $M$ rejects $u_1$ because $D_l(u_0) = w$ and $u_0 < \text{lex} u_1$. Since $u_0 \in L(M)$ and $u_1 \notin L(M)$ are mapped by $D_l$ to the same string $w$, $D_l$ cannot reduce $L(M)$ to anything, and to $S$ in particular.

Since by the above no $D_l$ reduces $L(M) \in \text{EXSPACE}$ to $S$, $S$ is not $\text{EXSPACE}$-complete. ■