Solutions for Problem Set 9
CS 373: Theory of Computation

Assigned: April 11, 2013   Due on: April 18, 2013

Problem 1. [Category: Comprehension+Proof] For strings $u, v \in \Sigma^*$, we will say $u < v$ to denote that u is less than v in the lexicographic order. An enumerator N is said to enumerate strings in lexicographic order iff for any strings $u, v \in \mathbf{E}(N)$, if $u < v$ then N prints u before v. In this problem, you are required to prove that a language is decidable iff some enumerator enumerates the language in lexicographic order.

1. Let M be a Turing machine that decides the language L. Show that there is enumerator N such that $\mathbf{E}(N) = L$ and N enumerates the words in L in lexicographic order. **[5 points]**
2. Let N be an enumerator that enumerates strings in lexicographic order. If $\mathbf{E}(N)$ is finite then $\mathbf{E}(N)$ is regular and, therefore, decidable. Prove that if $\mathbf{E}(N)$ is infinite then there is a Turing machine M that decides $\mathbf{E}(N)$. **[5 points]**

Solution:

1. The enumerator N for $\mathbf{L}(M) = L$, will run M on every string in lexicographic order, and output a string if M accepts. Here is a pseudocode for N.

   ```
   for w = ϵ, 0, 1, 00, 01, 10, 11, 000, ... do
       simulate M on w
       if M accepts w then write the word 'w'
           on output tape
   ```

   Observe that the above pseudo-code enumerates every string in $\mathbf{L}(M)$ only because M is guaranteed to halt on every input.

2. Consider an enumerator N such that $\mathbf{E}(N)$ is infinite, and the strings in $\mathbf{E}(N)$ are enumerated in lexicographic order. The decider M, on input w, will run N until either N outputs w or a string greater that w (in lexicographic order). Since $\mathbf{E}(N)$ is infinite one of these will happen in finite time. The pseudo-code for M is as follows.

   ```
   On input w
       Run N.  Every time N writes a word 'x'
       compare x with w.
       If x = w then accept and halt
       else if x > w then reject and halt
       else continue simulating N
   ```

■

Problem 2. [Category: Comprehension+Design] Show that

$$\mathsf{Inf}_{\mathsf{CFG}} = \{\langle G\rangle \mid G \text{ is a CFG such that } \mathbf{L}(G) \text{ is infinite}\}$$

is decidable by outlining an algorithm that decides this problem; you need not prove that your algorithm is correct. *Hint:* You may find it useful to look at the solution for problem 1 in Discussion 12 (or problem 4.10 in the textbook) and think about the pumping lemma for CFGs. **[10 points]**

Solution: Given a grammar G, the algorithm needs to check if $\mathbf{L}(G)$ is infinite. The algorithm will first convert G into Chomsky Normal Form. This algorithm will take exponential time (because of the step removing nullable variables), but will definitely terminate. Let the resulting grammar be G_1. From the proof of the pumping lemma for CFGs, we know that if $\mathbf{L}(G_1)$ contains some word of size $2^{|G_1|}$ (where $|G_1|$ denotes the number of variables in G') then $\mathbf{L}(G_1)$ contains infinite many strings. Thus, the algorithm will check if some word of length $\geq 2^{|G_1|}$ is in $\mathbf{L}(G_1)$; we will now describe how this can be done.

Observe that the collection of all words of length at most $2^{|G_1|}$ is a finite language, and hence regular. Since regular languages are closed under complementation, the language consisting of all strings of length $\geq 2^{|G_1|}$, denoted by $\Sigma^{\geq 2^{|G_1|}}$, regular. The algorithm needs to check if $\mathbf{L}(G_1) \cap \Sigma^{\geq 2^{|G_1|}} \neq \emptyset$. Observe that $\mathbf{L}(G_1) \cap \Sigma^{\geq 2^{|G_1|}}$ is context-free, since context-free languages are closed under interesection with regular languages. Thus, the algorithm will construct a CFG G_2 recognizing $L_1 = \mathbf{L}(G_1) \cap \Sigma^{\geq 2^{|G_1|}}$, by converting G_1 to a PDA, constructing a PDA recognizing L_1 by the proof given in class, and then converting that PDA back to a CFG. Now we need to check if $\mathbf{L}(G_2)$ is empty. This can be done by checking of the start symbol of G_2 is generating.

Putting all the steps together gives us a decision procedure for $\mathsf{Inf}_{\mathsf{CFG}}$ ■

Problem 3. [Category: Comprehension+Design+Proof] Disjoint languages A and B are said to be *recursively separable* if there is a decidable language L such that $A \subseteq L$ and $B \subseteq \overline{L}$. Prove that if A and B disjoint languages such that $\overline{A}$ and $\overline{B}$ are recursively enumerable then A and B are recursively separable. **[10 points]**

Solution: Let $M_{\overline{A}}$ be a Turing machine recognizing $\overline{A}$ and let $M_{\overline{B}}$ be a Turing machine recognizing $\overline{B}$. Since $A \cap B = \emptyset$, $\overline{A} \cup \overline{B} = \Sigma^*$. We will define the decidable language L that separates A and B, by giving the pseudo-code of a program that decides it. Consider the program M as follows

```
On input w
    for i = 1 to ∞ do
        Simulate M_Ā on w for i steps
        Simulate M_B̄ on w for i steps
        if either M_Ā or M_B̄ accept within i steps then break
    if M_Ā accepts w then reject
    if M_B̄ accepts w then accept
```

Observe that since $\overline{A} \cup \overline{B} = \Sigma^*$, for some value of i, either $M_{\overline{A}}$ or $M_{\overline{B}}$ will accept w, and so M will terminate on all inputs. Take $L = \mathbf{L}(M)$, which is decidable.

To complete the proof, we need to show that $A \subseteq L$ and $B \subseteq \overline{L}$. Let $w \in A$. Then $M_{\overline{A}}$ will not accept w, and so $M_{\overline{B}}$ will accept w. Hence, M will accept w, which shows that $A \subseteq \mathbf{L}(M) = L$. On the other hand, if $w \in B$ then $M_{\overline{B}}$ will not accept w. Again (since $\mathbf{L}(M_{\overline{A}}) \cup \mathbf{L}(M_{\overline{B}}) = \Sigma^*$) $M_{\overline{A}}$ will accept w, and so M will reject w. Thus, $B \subseteq \overline{L}$. This completes the proof. ■