Harvard University
Computer Science 121

Problem Set 6

Due Friday, November 5, 2010 at 1:20 PM.
Submit a single PDF (lastname+ps6.pdf) of your solutions to cs121+ps6@seas.harvard.edu
Late problem sets may be turned in until Monday, November 8, 2010 at 1:20 PM with a 20% penalty.
See syllabus for collaboration policy.

PROBLEM 1 (15 points)

(A) Write a general grammar that generates $\{ww : w \in \{a,b\}^*\}$. Explain in words what each rule in your grammar does.
(B) Demonstrate a derivation of this grammar for the string $abbabb$.

(A)

$$S \to aAS|bBS|T$$

$$Aa \to aA$$

$$Ab \to bA$$

$$Ba \to aB$$

$$Bb \to bB$$

$$AT \to Ta$$

$$BT \to Tb$$

$$T \to e$$

S generates aA and bB paired. A's and B's all walk to the right, but they can't cross over each other, so they stay in the same order. T walks to the left and converts all of them and vanishes when done.

Another solution:
Our general approach is to generate the string ww^R, that is all even length palindromes, and then have our general grammar reverse the order of w^R, thus yielding ww.

These first two lines generate us strings $w\#Xw^R\$$. We use the # to mark the boundary between the strings and \$ to mark the end of the string. X is a walker variable that we will use to move characters in w^R past the \$ and reverse the string.

$$S \to T\$$$

$$T \to \#X|aTa|bTb$$

These rules take the letter bordering X and move it gradually along until it hits the end of the string, i.e. the \$.

$$Xaa \to aXa$$

$$Xab \to bXa$$

$$Xba \to aXb$$

$$Xbb \to bXb$$

Once we hit the end, we change X to the symbol Y so we can walk back down the string to the # and start again

$$Xa\$ \to \$Xa$$

$$Xb\$ \to \$Xb$$

Now we walk back to the # sign.

$$\$X \to Y\$$$

$$aY \to Ya$$

$$bY \to Yb$$

$$\#Y \to \#X$$

We finish once all of the characters have been moved past the \$.

$$\#X\$ \to \varepsilon$$

(B) We exhibit a derivation using the first solution:

$$S \Rightarrow aAS \Rightarrow aAbBS \Rightarrow abABS \Rightarrow abABbBS \Rightarrow abAbBBS \Rightarrow abbABBS \Rightarrow abbABBT$$
$$\Rightarrow abbABTb \Rightarrow abbATbb \Rightarrow abbTabb \Rightarrow abbabb$$

.

PROBLEM 2 (10 points)

Define $\textsc{Prefix}(L) = \{x \mid xy \in L \text{ for some } y \in \Sigma^*\}$.

Show that if L is r.e., then $\textsc{Prefix}(L)$ is r.e.

(A)

Solution 1:

We use a recognizer M for L to construct a recognizer R for $\textsc{Prefix}(L)$:

On input x, R will dovetail the computation of M on all strings xy, $y \in \Sigma^*$, accepting if M accepts any xy. This will accept if and only if x is a prefix of some string in L, so R recognizes $\textsc{Prefix}(L)$.

Solution 2:

Since L is recognizable, there exists an enumerator E for L. We construct a recognizer R for $\textsc{Prefix}(L)$ as follows:

On input x, R runs E, and for every string w E outputs, accepts if x is a prefix of w. This will accept if and only if x is a prefix of some string in L, so R recognizes $\textsc{Prefix}(L)$.

Solution 3:

We construct an enumerator R for $\textsc{Prefix}(L)$ using an enumerator E for L.

R: run E. For every string w output by E, output all prefixes of w.

This will enumerate exactly the strings in $\textsc{Prefix}(L)$, which proves that $\textsc{Prefix}(L)$ is recognizable.

PROBLEM 3

Show that every infinite r.e. language has an infinite recursive subset. (Hint: apply an ordering to Σ^*).

(A)

Let L be an infinite Turing-recognizable language, and let M be a Turing machine that enumerates it (to find out more on enumerators, refer to Section 3.2 in Sipser). Consider the language $L' = \{w_1, w_2, w_3 \ldots\}$ where:

- w_1 is the first string enumerated by M.
- For $i > 1$, w_i is the first string enumerated by M that is lexicographically larger than w_{i-1}.

Given the enumerator M, it is trivial to create an enumerator M' that will enumerate L' in lexicographic order: just wait till M outputs a string lexicographically larger than the previous one. But we saw in lecture that a language L is decidable if and only if it is enumerable in lexicographic order. Therefore, L' is a decidable subset of L

Extra note: It's easiest to think of this problem in terms of integers, which are computationally equivalent to strings in lexicographic order. So if M enumerates $5, 4, 2, 3, 7, 4, 8, 0, 7, 11, 9, 3, 19, 25, 17, 2, \ldots$ then the first few elements in L' would be $5, 7, 8, 11, 19, 25$. Since L is infinite, L' must also be infinite, or else we would have that L contains only numbers less than some k, implying that it is finite (contradiction).

A reminder of why L' is decidable: Say we are given n. We run the above algorithm, and look at the elements added to L'. If we see n, we accept. If we see a number greater than n, we reject. This works because we have a listing of L' in increasing order, so if we see a number greater than n, we know we will never see n and can safely conclude that n is not in L'. Note that the same argument does not apply to L, since we don't have a listing (or enumeration) of L in increasing (or for that matter, any useful) order.

PROBLEM 4 (8+10 points)

(A) Let $L = \{\langle D, k\rangle : D \text{ is a DFA that accepts exactly } k \text{ strings, where } k \in \mathbb{N} \cup \{\infty\}\}$. Show that L is recursive. (*Hint:* Show how to find a p such that if D accepts any string of length at least p, then D accepts infinitely many strings).

(B) Let $L = \{\langle M\rangle : M \text{ is a TM and } L(M) \text{ contains a string with no } a\text{'s}\}$. Show that L is r.e.

(A)

From the pumping lemma (Sipser p. 78), we know that for $p =$ number of states in a DFA, if the DFA accepts any string of length at least p, it must accept infinitely many strings.

The following is a decider for L:

$D =$ On input $\langle D, k\rangle$:

1. Let p be the number of states of D. Simulate D on all strings w, where $p \leq |w| \leq 2p$.
2. If D accepts any string of that length, the language is infinite, so accept if $k = \infty$, and reject if $k \neq \infty$.
3. If D rejects all strings of that length, then simulate D on all strings of length less than p. Count the number of accepting strings, and let that number be c. If $k = c$, then accept.
4. Otherwise, reject.

Now if D accepts some string of length greater or equal to p, then D accepts infinitely many strings. If D rejects all strings of length between p and $2p$, then by the pumping lemma, we can deduce that D rejects all string of length greater or equal to p. (If D accepts w and $|w| \geq 2p$, then, letting w be the shortest string accepted by D of length at least p, we see that $|w| \leq 2p$ because otherwise we can pump down w to get a shorter string.)

Note: It was also possible to do cycle detection on the DFA graph directly to tell whether the language was infinite. If you do that, you need to be careful that you only look for reachable cycles from which it is also possible to reach an accept state.

(B)

We'll construct a recognizer R for L. Given input $\langle M\rangle$, R will simulate M on all strings in $(\Sigma - \{a\})^*$ in parallel by using dovetailing, accepting if any of those computations accept. Thus R will eventually accept $\langle M\rangle$ if M a string that does not contain an a, and will run forever otherwise.

Note: remember that if you just run an arbitrary TM M on some string, it may not halt. Therefore dovetailing is crucial here–if we just run M on all strings in Σ^* in order, it may loop on the first one and never get to the others. As a reminder, dovetailing computations of M over some set of strings means running the M for 1 step on the first string, then for 2 steps on the first two strings, and so on. Eventually, this will get to all strings and all numbers of steps, so if M accepts any string w after k steps, this method will terminate.

PROBLEM 5 (8+8 points)

Consider $L = \{\langle M\rangle : M$ is a TM that accepts no strings shorter than 42 characters in length$\}$
(A) Prove that L is not recursive.
(B) Prove that L is not r.e.

(A) Observe that $L = \{\langle M\rangle : L(M) \in \{L : L \text{ is Turing recognizable and } L \cap \{w : |w| < 42\} = \emptyset\}\}$, since any language that can be written $L(M)$ for some Turing Machine M is necessarily Turing recognizable. Observe that $\{L : L \text{ is r.e. and } L \cap \{w : |w| < 42\} = \emptyset\}$ is a set of Turing recognizable languages by construction, non-empty, since it contains $\emptyset$, and not the set of all Turing recognizable languages, since it does not contain $\{e\}$. Thus, by Rice's Theorem, it is undecidable.

(B) $\overline{L}$ is Turing recognizable, since we can build a Turing Machine $M_{\overline{L}}$ that recognizes $\overline{L}$. On input $\langle M\rangle$, $M_{\overline{L}}$ will simulate M on each string shorter than 42 characters in length for one step, then for two steps, then for three steps, and so on, halting if any of those simulations ever halt and otherwise continuing to simulate. On any input which is not the encoding of a Turing Machine, $M_{\overline{L}}$ will halt. To see that $M_{\overline{L}}$ does in fact recognize $\overline{L}$, suppose that $\langle M\rangle \in \overline{L}$. Then $L(M)$ contains at least one string shorter than 42 characters in length, and so there is some such string on which M halts. Let w be such a string and let q be the number of states that M requires to halt. By construction, unless it halts sooner, $M_{\overline{L}}$ will eventually simulate M on w for q steps and thus halt, so $\langle M\rangle \in L(M_{\overline{L}})$. Conversely, suppose that $\langle M\rangle \notin \overline{L}$; then there are no strings shorter than 42 characters in length on which M halts. By construction, $M_{\overline{L}}$ will halt only if a simulation of M on such a string halts, so $M_{\overline{L}}$ will never halt on $\langle M\rangle$. Thus $\langle M\rangle \notin L(M_{\overline{L}})$. But now we have shown that $\overline{L} = L(M_{\overline{L}})$, so $\overline{L}$ is Turing recognizable as was to be shown.

Since $\overline{L}$ is Turing recognizable but L is not decidable, L is not Turing recognizable.

PROBLEM 6 (Challenge!! 3 points)

Show that there is an infinite co-recognizable language L such that it does not have an infinite subset $L' \subseteq L$ that is recognizable.

(A) We are basically looking for a $\overline{L}$ such that $\overline{L}$ is Turing recognizable, has an infinite complement, and intersects every infinite Turing recognizable set. Here is how we build $\overline{L}$. We start every TM going on every input (using a dovetailing process). We put into $\overline{L}$ the first string accepted by the ith TM whose length is greater than $2i$; if $L(M_i)$ is infinite, we are guaranteed that M_i will produce such a string; if it isn't we don't care anyway. In this way, $\overline{L}$ will intersect every infinite Turing recognizable set (since every Turing recognizable set is recognized by some TM, and we are doing the above process for every TM). L is infinite because for every $k \in \mathbb{N}$, $\overline{L}$ contains at most half of the numbers from 1 to $2k$. $\overline{L}$ is also Turing recognizable, since we have just described a procedure to enumerate $\overline{L}$, so L is co-recognizable. And because $\overline{L}$ intersects every infinite Turing recognizable set, then every infinite Turing recognizable set has an element that is not a member of L, so there is no way that any infinite Turing recognizable set can be a subset of L.