CS381, Homework #12 Solutions *Question 1*

Prove that the intersection of two CFGs is undecidable.

We shall prove this fact with a reduction from the halting problem to the intersection of two CFGs. Recall from class, that for some Turing Machine M, on some input x we can write the valid computations of M as $G = L(C_1) \cap L(C_2) = k_1 \vdash (k_2)^R \vdash ... \vdash k_n$. If you do not remember how to do this, review your course notes.

Now observe, that if G (which is the intersection of two context free languages) is non-empty, then it must contain some finite length string, which means that the Turing Machine M has some sequence of valid computations such that it halts on input x. We can make sure that when we formulate our intersection, we allow the machine to end in any state (and not just final states) as long as no input remains and there are no epsilon transitions from the state in k_n. Thus, it may be that a string x is not accepted by M, but if G is nonempty, it can be assured that M halts on x. If the intersection is empty on the other hand, then there is no such sequence, and the Turing Machine will run forever. This gives us a way to determine if a machine if a Turing Machine M halts on input x using the intersection of two CFLs.

This conclusion though creates a contradiction, for if this intersection problem were decidable, we could solve the halting problem using it, and since we know that the halting problem is in fact undecidable, it cannot be that the intersection of two context free languages is decidable.

CS 381/481 Homework 12

#2 (Exercise 9.3.4)

a) Let $L_2 = \{\langle M \rangle : |L(M)| \geq 2\}$. Here, $\langle M \rangle$ denotes a string encoding of a Turing machine M. L_2 is RE (i.e., recognizable).

We can build a Turing machine T that accepts (recognizes) L_2. For $k = 1, 2, 3, \ldots$, T simulates M on w_1 to w_k for at most k steps, and see if M accepts. (This is called *dovetailing.*) T keeps a count of how many strings has been accepted by M, and accepts if two has been found.

b) Let $L_\infty = \{\langle M \rangle : |L(M)| = \infty\}$. L_∞ is not RE (i.e., unrecognizable).

We can also show that if L_∞ is recognizable, then the halting problem is decidable. The proof is similar to that of Rice's theorem, with a slight twist.

Let R be a recognizer for L_∞. We now build a decider H for the halting problem. $H(\langle M, x \rangle)$ accepts if M halts on input x, and rejects otherwise.

$H(\langle M, x \rangle)$:

- Construct a machine M' such that M' does the following.

  $M'(w)$:

  - Simulate $M(x)$ for at most $k = |w|$ steps.
  - If M did not halt within k steps, then accept w.

- Simulate $M(x)$ and $R(\langle M' \rangle)$ in "parallel". That is, simulate $M(x)$ for one step, then simulate $R(\langle M' \rangle)$ for one step, and so on.
- Accept if $M(x)$ ever halts.
- Reject if $R(\langle M' \rangle)$ ever accepts.

The crux of this solution is the construction of M'. Observe that if $M(x)$ does not halt, then M' will accept all strings, so $L(M') = \Sigma^*$, which is infinite. Otherwise, if $M(x)$ does halt after k steps, then $L(M') = \{w : |w| < k\}$, which is a finite set!

By running $M(x)$ and $R(\langle M' \rangle)$ in parallel, H can always decide whether M halts on input x. Here's how: Suppose it does halt. Then the simulation of $M(x)$ will eventually halt and we accept. Suppose $M(x)$ does not halt. Then $L(M') = \Sigma^*$, so

our recognizer R for L_∞ is required to halt at some point and accept $\langle M' \rangle$. At this point, H can halt and reject.

c) Let $L_{\text{CFL}} = \{\langle M \rangle : L(M) \text{ is context-free}\}$. L_{CFL} is not RE.

Same proof as part b), except we build M' as follows.

$M'(w)$:

- Simulate $M(x)$.
- Accept w if it's of the form $a^n b^n c^n$.

If $M(x)$ does not halt, then $L(M') = \emptyset$, which is context-free. Otherwise, if $M(x)$ does halt, then $L(M') = \{a^n b^n c^n\}$, which is not context-free.

d) Let $L_R = \{\langle M \rangle : L(M) = L(M)^R\}$. L_R is not RE.

Same format as part b). Construct M' as follows.

$M'(w)$:

- Simulate $M(x)$.
- Accept w if it's of the form $a^n b^n$.

If $M(x)$ does not halt, then $L(M') = \emptyset$, and thus $L(M') = L(M')^R$ trivially. Otherwise, $L(M') = \{a^n b^n\}$, so $L(M') \neq L(M')^R$.

CS 381 HW 12
Solutions

Exercise 9.3.5 Show that L is RE, but not recursive.

RE:
Let M be a turing machine that accepts L. For each set of inputs (M1,M2,k), the turing machine M could simulate all strings w on machines M1 and M2. If both M1 and M2 accept w, then w is in the intersection of L(M1) and L(M2). Repeat this for all possible strings w. Once k strings are found, halt and accept.

This is RE because a TM M halts and accepts on all accepted inputs.

RE, but not recursive:
The easiest way to show this is by using Rice's thereom. "Every nontrivial property of the RE languages is undecidable." Since we've already shown that L is RE, just show that this property is nontrivial. One way to do this is to show one input that is accepted, and one that is not accepted.

For example:
Accepted: (a*,a*,5)
Not Accepted: (a*, b*, 5)

Since this is undecideable and RE, L is RE, but not recursive.

CS 381 Homework #12 Problem 4

Question 9.3.6

a)

After making *m* transitions (not *m+1* as suggested by the hint), the TM will have been in *m+1* different states. These states cannot all be different. Thus, we can find some repeating state, and the moves of the TM look like $[q_0] \vdash^* q \vdash^* q \vdash^* \ldots$, where the central $\vdash^*$ represents at least one move. Note that we assume the tape remains blank; if not then we know the TM eventually prints a nonblank. However, if it enters a loop without printing a nonblank, then it will remain forever in that loop and never print a nonblank. Thus, we can decide whether the TM ever prints a nonblank by simulating it for *M* moves, and saying "yes" if and only if it prints a nonblank during that sequence of moves.

b)

To decide whether a TM never makes a move to the left, we simply look at the code for the TM and check its transitions for any moves to the left. If there are none, we report "yes".