CS 280: Homework 8 Solutions

Date of Handout: 24 Nov 99

Problem 1: (5 points) Assume that all possible Turing machines are written down in the sequence $M_1, M_2, M_3, \ldots$. Assume that all strings of finite length are written down without any repetitions in the sequence $w_1, w_2, w_3, \ldots$ Prove that the language

$$L_d = \{w_i | M_i \textit{ does not accept } w_i\}$$

is not r.e. (This is Lemma 8.1 in the text and was discussed in class.)

Assume that L_d is accepted by a Turing machine M. This Turing machine must appear as M_i for some positive integer i in the sequence of Turing machines. By definition of L_d, if $w_i \in L(M_i)$ then $w_i \notin L_d$ and if $w_i \notin L(M_i)$ then $w_i \in L_d$. Thus $L(M_i) \neq L_d$ and the assumption that L_d is accepted by a Turing machine is false.

In words: L_d differs from the language accepted by the ith Turing machine on the ith string. Therefore, there is no Turing machine which accepts L_d.

Problem 2: Assume there is an algorithm to decide if two Turing machines M_1 and M_2 accept exactly the same language. Use that algorithm to construct

(a) (10 points) an algorithm to decide if the language accepted by a Turing machine M is empty,

(b) (10 points) an algorithm to decide if a Turing machine M accepts a string w (this is L_u in the text).

(5 points) Argue that no algorithm can tell if two C programs perform the same task in possibly different ways.

Assume that algorithm A can decide if $L(M_1) = L(M_2)$ when it is input M_1 and M_2.

(a): Construct a Turing machine M_2 which rejects every string and hence has $L(M_2) = \phi$. For example, you can construct M_2 by taking the start state to be the reject state. Feed M and M_2 as inputs to algorithm A. If $L(M) = L(M_2)$, conclude that M accepts the empty language. If $L(M) \neq L(M_2)$, conclude that the language accepted by M is not empty.

(b): Construct a Turing machine M' which works as follows. M' starts off by keeping its input aside. Then M' simulates M on w. If M never halts on w, then M' never halts on any input and thus accepts only the empty language. If M rejects w, then M' rejects its input whatever it is. Thus, in this case too, M' accepts only the empty language. However, if M accepts w then M' accepts its input whatever it is. Thus, in this case, $L(M') = \Sigma^*$.

If the action of a Turing machine M on a string w is thought of as a function call $M(w)$, this construction can be written down as follows:

```
M'(w')
{
result = M(w);
if result == REJECT
return REJECT;
else if result == ACCEPT
return ACCEPT;
}
```

This makes it clear that M' goes into an infinite loop on any input w' if M goes into an infinite loop on w; that M' accepts any input w' if M accepts w; and that M' rejects any input w' if M rejects w. The language accepted by M' is the null language when M does not accept w and Σ^* when M accepts w.

Feed M' and a Turing machine M_2 which accepts the null language to algorithm A. If $L(M') = L(M_2)$ then M does not accept w. If $L(M') \neq L(M_2)$, then M accepts w.

The constructions in (a) and (b) are different ways of showing that it is undecidable whether two Turing machines accept the same language. If it were decidable, the two constructions give algorithms to decide the acceptance of a string by a Turing machine and the problem of whether the language accepted by a Turing machine is empty. But both these problems are undecidable.

Given two Turing machines, they can be converted into two C programs easily. The C program can store a variable that will give the state of the Turing machine, an array that corresponds to the tape of the Turing machine and an integer that gives the position of the head of the Turing machine on the tape. Thus, if it is possible to compare and decide if two C programs do the same task, it is possible to decide if two Turing machines accept the same language by translating the Turing machines into C programs and then using the decision procedure that compares C programs. But it is impossible to decide using an algorithm if two Turing machines accept the same language. So it must be impossible to decide if two C programs do the same task.

Problem 3: (10 points) Assume there is an algorithm to decide if a Turing machine M accepts a string w (this is L_u in the text). Use this to give an algorithm to decide if a Turing machine M halts on the input w (this is the halting problem).

Assume that algorithm A can decide if a Turing machine M accepts a string w.

Construct M' as follows:

```
M'(w)
{
result = M(w);
if result == REJECT or result == ACCEPT
return ACCEPT;
}
```

Clearly, M' accepts w if and only if M halts on w. Feed M' and w as inputs to algorithm A. If it says that M' accepts w then M halts on w. If it says that M' does not accept w, then M does not halt on w.