

Homework 5 Solutions, CS 132, Winter 2005

March 10, 2005

- 11.15 Show that the decision problem **WritesNonBlank**: Given a TM T , does it ever write a nonblank symbol on its tape when started with a blank tape? is solvable, by providing a decision algorithm.

Suppose T has n non-halting states. Within n moves, T will have halted or entered some non-halting state q for the second time. Since there is no input on the tape, if T enters any state twice it is in a loop and will either continue infinitely down the tape to the right without ever writing a symbol or continue to the left and crash. This can easily be determined by recording where on the tape T enters each state. Say q has been encountered twice, first on the i th cell and then on the j th cell. If $j > i$ then T is in an infinite loop. If $i > j$ then it will continue to enter state q at cells $2j - i, 3j - 2i, 4j - 3i, \dots$ until it crashes at the beginning of the tape. Thus in the first n moves T will write a non-blank symbol or else it will have repeated a state or crashed, in which case it never will.

1. (11.18) Give a solution to the correspondence system or show that none exists.

a. $\alpha_1 = 100, \alpha_2 = 101, \alpha_3 = 110, \beta_1 = 10, \beta_2 = 01, \beta_3 = 1010$.

Any viable solution must begin with 1, which means the next two must be 2, which then requires 1 again, but no correspondence is possible from here.

b. $\alpha_1 = 1, \alpha_2 = 01, \alpha_3 = 0, \alpha_4 = 001, \beta_1 = 10, \beta_2 = 101, \beta_3 = 101, \beta_4 = 0$.

One solution is $\alpha_1\alpha_4\alpha_2 = \beta_1\beta_4\beta_2$.

- 11.21 c. Show that the following decision problem is unsolvable. **CFGEqualReg**: Given a CFG G and a regular language R , is $L(R) = R$?

We will show **CFGGeneratesAll** \leq **CFGEqualReg**.

The input to **CFGGeneratesAll** is a grammar G over some alphabet Σ ; the input to **CFGEqualReg** is a grammar G' and a regular language R . We must give a computable F so that G is a yes instance of **CFGGeneratesAll** if and only if $F(G) = (G', R)$ is a yes-instance of **CFGEqualReg**.

1. We define $F(G) = (G, \Sigma^*)$. Recall that Σ^* is a regular language.
2. F makes no modification to G and needs only denote Σ^* , not enumerate it, so F is computable.
3. If G is a yes-instance of **CFGGeneratesAll** then $L(G) = \Sigma^*$ and so (G, Σ^*) is a yes-instance of **CFGEqualReg**. Likewise if $L(G') = \Sigma^*$ then, since $L(G) = L(G')$, G is a yes-instance of **CFGGeneratesAll**.

- 12.1 Let $f : \mathcal{N} \rightarrow \mathcal{N}$ be the function defined as follows: $f(n)$ is the maximum number of moves an n -state TM with tape alphabet $\{0, 1\}$ can make if it starts with input 1^n and eventually halts. Show that f is not computable.

Suppose f is computable, and let T_f be the TM that computes f . We can create a TM $T' = T_f T_1$ where T_1 is a TM which makes a single move and then halts. T' has a fixed number of states, call this number m . By definition of the function f , T' makes fewer than $f(m)$ moves on input 1^m . But since T_f writes m to the tape it must make at least m moves, meaning T' must make at least $m + 1$. Contradiction.

12.5 Show that if $f : \mathcal{N} \rightarrow \mathcal{N}$ is a total function, then f is computable if and only if the decision problem: Given $n, C \in \mathcal{N}$, is $f(n) > C$? is solvable.

If f is a computable total function, then we can solve the decision problem by computing $f(n)$ and comparing it to C .

If the decision problem is solvable, then we can compute $f(n)$ by successively solving the problem $f(n) > C$ for $C = 0, 1, \dots$; we output the first C such that the $f(n) > C$ is false and then immediately halt.

Another way of saying this is

$$f(n) = \mu C [f(n) > C \text{ is false}].$$

Since $[f(n) > C \text{ is false}]$ is a total function and all μ -recursive functions are computable this shows that f is computable.

11.20 (Extra Credit) Show that the special case of **PCP** in which the alphabet has only one symbol is solvable.

Given an instance $(\alpha_1, \beta_1), \dots, (\alpha_n, \beta_n)$ of **PCP** in which $\alpha_i, \beta_i \in \{1\}^*$. For each i , let $d_i = |\alpha_i| - |\beta_i|$. If $d_i = 0$ for some i , then clearly it is a yes-instance. Likewise if $d_i > 0$ or $d_i < 0$ for all i then clearly it is a no-instance.

Otherwise, there is an i and j such that $d_i = p > 0$ and $d_j = -q < 0$. In this case we claim $\alpha_i^q \alpha_j^p = \beta_i^q \beta_j^p$ is a solution. Note that since we have only one symbol in our alphabet, strings differ only by length. The length of the alphas is

$$q|\alpha_i| + p|\alpha_j| = (|\beta_j| - |\alpha_j|)|\alpha_i| + (|\alpha_i| - |\beta_i|)|\alpha_j| = |\alpha_i||\beta_j| - |\alpha_j||\beta_i| =$$

$$(|\beta_j| - |\alpha_j|)|\beta_i| + (|\alpha_i| - |\beta_i|)|\beta_j| = q|\beta_i| + p|\beta_j|.$$

Thus $\alpha_i^q \alpha_j^p = \beta_i^q \beta_j^p$.

We can use this property to construct a simple decision algorithm: we check if there exists i such that $d_i = 0$ or j, k such that $d_j > 0$ and $d_k < 0$.

11.34 Is the decision problem: Given a CFG G and a string x , is $L(G) = \{x\}$? solvable or unsolvable.

It is solvable. Here is an algorithm to solve it. First test x for membership in $L(G)$. If $x \notin L(G)$ then $L(G) \neq \{x\}$. If $x \in L(G)$ then construct a PDA M accepting $L(G)$ using the method in Section 7.4. Since $L_1 = \{z \in \Sigma^* | z \neq x\}$ is regular, the proof of Theorem 8.4 provides an algorithm for constructing another PDA M_1 to accept $L \cap L_1$. Section 7.5 describes an algorithm to produce a CFG G_1 generating $L(M_1)$. In Section 8.3 there is a decision algorithm to decide whether $L(G_1) = \emptyset$. $L(G) = \{x\}$ if and only if $L(G_1) = \emptyset$.