COS 487
Fall 2003

Problem Set 6 Solutions

Problem 1:

(aa/a)(aa/a)(b/a)(ab/abab)

Problem 2:

Suppose *M''* decides *L* = {<*M*> | *M* is a TM and *M* has no useless states}. Let be an instance of an A_{TM} problem. Then the following machine decides A_{TM}, a contradiction.

M''' = "On input <*M*, *w*>:
1. Create *M'* from *M* by adding a transition to and from every state q_i in *M* on a new symbol x_i.
2. Run *M''* on <*M'*, $x_1x_1x_2x_2\ldots x_mx_mw$>. Accept if *M''* accepts."

Problem 3:

Observe that given an instance of PCP over a unary alphabet, a match exists only if there exists a domino with the same number of symbols on the top and bottom (a singleton match) or if there exists a domino with more symbols on the top and another domino with more symbols on the bottom. Constructing a TM which checks these two conditions is straightforward.

Problem 4:

The PCP problem can be reduced to the binary PCP problem, so the binary PCP problem is undecidable. Encode each symbol x_i as 01^i.

Problem 5:

The reduction works if *P* has a match iff *G* is ambiguous. If *P* has a match $t_1t_2\ldots t_n = b_1b_2\ldots b_n$, then the string $t_1t_2\ldots t_na_na_{n-1}\ldots a_1$ has two different parse trees in *G*. If *G* is ambiguous, then some string $t_{i1}t_{i2}\ldots t_{in}a_{in}a_{in-1}\ldots a_{i1}$ has two different parse trees. But any string has only one parse tree from *T* (by examination of the sequence of symbols a_i), and the same holds for *B*. So the string can be derived from *T* as well as *B*, and the sequence $i_1i_2\ldots i_n$ form the indices of a match of *P*.

Problem 6:

Let *P'* be *P* if *P* contains the TM which rejects all inputs, and let *P'* be he complement of *P* otherwise. We reduce the Halting Problem to *P'*, showing that *P* is undecidable in both cases.

Suppose *M''* decides *P*. Let $<M_1>$ be a machine in *P*. Then the following machine decides the Halting Problem.

M''' = "On input $<M, w>$:

1. Create a new TM *M'* which on input *w'* first simulates *M* on *w* and then simulates M_1 on *w'*.

2. Run *M''* on $<M'>$. If *M* halts on *w*, then *M'* is identical to M_1. Otherwise, *M'* recognizes the empty language. Accept if *M''* accepts."