Problem 1:

\[(aa/a)(aa/a)(b/a)(ab/abab)\]

Problem 2:

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

\[M''' = \text{"On input } <M, w>:\]
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 \(<M', x_1x_1x_2x_2...x_mx_mw>\). 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 \(t_1t_2...t_n = b_1b_2...b_n\), then the string \(t_1t_2...t_na_na_{n-1}...a_1\) has two different parse trees in \(G\). If \(G\) is ambiguous, then some string \(t_{i_1}t_{i_2}...t_{i_m}a_{i_m}a_{i_{m-1}}...a_{i_1}\) 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 \(i_1i_2...i_m\) 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 the 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.

$$M'' = \text{"On input } <M, w>:\n1. \text{Create a new TM } M' \text{ which on input } w' \text{ first simulates } M \text{ on } w \text{ and then simulates } M_1 \text{ on } w'.
2. \text{Run } M'' \text{ on } <M'>. \text{ If } M \text{ halts on } w, \text{ then } M' \text{ is identical to } M_1. \text{ Otherwise, } M' \text{ recognizes the empty language. Accept if } M'' \text{ accepts."}$$