1

Strategy

Bound $|c_n x^n|$ below and each lower $|c_i x^i|$ above.

Outline

We break the polynomial into its first term and its remaining term. We show that for large enough x, the first term dominates the remaining terms, so their sum cannot be 0.

Proof

Let $|x| > (n+1)\frac{c_n}{c_{\max}}$. Then

$$|c_0 + \cdots + c_{n-1}x^{n-1}| \leq |c_0 + \cdots + c_1||x|^n \leq |c_{\max} + \cdots + c_{\max}||x^n| = (n+1)c_{\max}|x^n| < |c_n x^n|.$$

Thus

$$|c_0 + \cdots + c_n x^n| \geq |c_n x^n| - |c_0 + \cdots + c_{n-1}x^{n-1}| > 0$$

and x is not a root.

Thus any root x_0 must have $|x_0| < (n+1)\frac{c_{\max}}{|c_n|}$

2

Strategy

Reduce from the complement of the accepting problem.

Outline

We describe a mapping reduction from the accepting problem to the useless states problem by constructing a Turing machine that passes through all its states except q_{accept} on some input, but never accepts unless it does so simulating M on x.

Proof

Our language is

$$\{\langle M = (Q, \Sigma, \Gamma, \delta, q_0, q_a, q_r)\rangle \mid \text{ for some } q \in Q \text{ and all } w \in \Sigma^*, M \text{ is not in state } q \text{ after reading } w\}.$$

Consider the computable function mapping $\langle M, x\rangle$ to $\langle M_x\rangle$. M_x simulates M if the input is x. Otherwise, it transitions through all its non-halting states (say, by printing an extra tape symbol which causes the states to transition to the next state in some enumeration when they read it) and then halts.

Each state in M_x is reached on any non-x input except for q_{accept}. Thus, each non-accept state is not useless. The accept state is never reached except possibly on x, so it is useless iff it is reached on x. Thus M_x has a useless state iff q does not accept.

3

Strategy

We prove the forward direction by construction; the reverse direction follows from known results.

Outline

The reverse direction follows from the fact that A_{TM} is semidecidable. For the forward direction, given a semidecider M for L, our reduction maps w it to $\langle M, w\rangle$, since by definition of a semidecider M accepts w iff $w \in L$.

Proof

Let L be a language. Assume L is Turing-recognizable and let M be a a TM recognizing it, so $L(M) = L$. We will construct a reduction f from L to A_{TM} according to the map $w \mapsto \langle M, w\rangle$. By construction, M accepts a string w iff $w \in L$, so $\langle M, w\rangle \in A_{TM}$ iff $w \in L$, so f is membership-preserving. Clearly f is computable, so it is a valid reduction, completing the forward direction of the proof.

As for the reverse direction, we have shown in class that A_{TM} is Turing-recognizable and that a language that is mapping-reducible to a Turing-recognizable language is itself Turing-recognizable, so if $L \leq_m A_{TM}$, then L is Turing-recognizable, completing the proof.

4

Strategy

Assume the input is ww and use the extra information.

Outline

When reducing from the accepting problem to DOUBLESTACK by sending $\langle M, x\rangle$ to $\langle P\rangle$, we don't care what P decides on strings which are not of the form ww. Thus our proof that $\langle P\rangle$ is in DOUBLESTACK or its complement at the appropriate times need only consider the behavior of P on strings of the form ww.

But on strings of the form ww, we have extra information: we know there's a second copy of the input! So, when we "lose" information from the input by reading it and not pushing it onto the stack, we actually know that information is still there in the second half. This allows us to check all the steps in the configuration history with two sweeps through.

Proof

As in the proof for ALL_{PDA} in Sipser, we encode configuration histories with every other configuration reversed, and delimeters in between them. We now describe a fully deterministic PDA P which, provided its input is two copies of a string $w\#w\#$, determines if w is an accepting configuration history:

1. While reading the first configuration, determine if it is w. If not, move to a reject state. In this process, write it onto the stack.
2. While there is a configuration in the stack, read another from the input and make sure there is a valid transition between them. Then read another from the input and push it onto the stack.
3. If we ever reach a # instead of another configuration when we go to push, if there is an accepting configuration in the stack then mark in the states that the configuration history accepts.
4. Skip the first configuration, push the second configuration onto the stack, then repeat step 2.
5. If we reach a # and the end of the input, and there is a state in the stack, accept iff it is an accepting configuration history. If we reach a # and the end and there is no state in the stack, accept iff we previously marked seeing an accepting configuration.
6. If we ever reach the end of the input and not a # before it, move to a reject state.

By the note in the outline, we need only consider the behavior of P on inputs of the form ww. If the input is ww and w does not end with a #, it will be rejected by state 6. Thus we only need consider the behavior of P on inputs of the form ww. On such a behavior, step 1 will ensure that the first configuration is x and steps 3 and 5 will ensure that the last configuration accepts. Step 2 ensures that each even-numbered configuration has a transition to the next, and step 5 will ensure that each odd-numbered configuration has a transition to the next. Between these, this PDA accepts ww if and only if w is an accepting configuration history of M on x, followed by a #.

Thus P accepts some ww if and only if there is an accepting configuration history of M on x, i.e. if and only if M accepts x.

Thus $A_{TM} \leq$ DOUBLESTACK, so since A_{TM} is undecidable, so is DOUBLESTACK.