2nd Lt David Crow

ENG/20M

CSCE 532 Homework 6

Problem 5.9 (complete proof): Let . Show that is undecidable.

We use the undecidability of to prove the undecidability of by reducing to . Let’s assume for the purpose of obtaining a contradiction that TM decides . We construct TM to decide , where operates as follows:

“On input such that is an encoding of a TM and is a string:

1. Construct the following TM:

“On input such that is a string:

1. If , run on input . If accepts, accept; otherwise, reject.
2. If , reject.”
3. Run on input . If accepts, accept; otherwise, reject.”

Clearly, decides , but Theorem 4.11 tells us that is undecidable, so we’ve reached a contradiction. We assumed that decides , so our assumption is false. Thus, is undecidable.

Problem 5.17: Show that the Post Correspondence Problem is decidable over the unary alphabet

.

In this alphabet, our dominos all have the form . This means some domino can only differ from some other domino if or or both. We can construct the following TM to decide this version of the PCP:

“On input such that is a collection of dominos:

1. If for all , reject.
2. If for all , reject.
3. Accept.”

We reject in step 1 and in step 2 because there is no sequence of dominos such that the length of the top string is equal to the length of the bottom string. We accept otherwise. If we have some domino such that , we have a match.

If such a does not exist, we can select a domino such that and a domino such that . Let and let . Any sequence of dominos and dominos has in the top row and s in the bottom row. Clearly, then, both the top and bottom row have s, so we have a match.

Our TM either accepts or rejects on every input, so decides the PCP for a unary alphabet.

Problem 5.22: Show that is Turing-recognizable iff .

To prove this, we must prove both of the following:

1. is Turing-recognizable
2. is Turing-recognizable

Proving (1) is trivial. The proof for Theorem 4.11 tells us that is Turing-recognizable. By Theorem 5.28, then, is Turing-recognizable if .

To prove (2), note that, if is Turing-recognizable, then there exists some TM that recognizes . This means that receives input and accepts if . We now construct a new TM as follows:

“On input such that is a TM and is a string:

1. Run on input . If accepts, accept. If rejects, reject.”

Because is equivalent to , we’ve shown that if is Turing-recognizable.

We’ve now proven both directions, so we’ve also proven the biconditional.

Problem 5.35: Say that a variable in CFG is *necessary* if it appears in every derivation of some string . Let .

1. Show that is Turing-recognizable.

We know we can enumerate every string by generating every string in (where is the alphabet of ) and printing if .

We can construct a TM to recognize in the following way:

“On input such that is a CFG and is a variable in the alphabet of :

1. Construct a CFG from by deleting every production in where is variable on the left-hand side and by deleting every instance of in the right-hand side of every production.
2. Generate a new string .
3. Run on input . If rejects, accept. Otherwise, return to stage 2.”

In stage 3, we see that we accept if the CFG does not recognize . Because we know that recognizes , and because we constructed by deleting only the instances of from , we know that is necessary for generating , so we accept.

If we don’t accept, we have to test a new string. Because the set of strings in is infinite, we can’t guarantee that will halt. Given infinite time, will eventually generate a string such that is necessary in ’s derivation, *but only* *if such a string exists*. In other words, will always accept if is necessary in . We now know that is Turing-recognizable.

1. Show that is undecidable.

Let’s assume that decides . We now construct the following TM:

“On input such that is a TM and is a string:

1. If does not simulate a CFG , reject.
2. If for some variable in the set of variables in , reject.
3. Construct as described in problem 5.35a.
4. Run on input . Output whatever outputs.”

Clearly, then, if a decider for exists, then a decider for must also exist. However, we know that is undecidable by Theorem 5.1. Because our only assumption was that exists, our assumption must be false. Thus, is undecidable.