2nd Lt David Crow

ENG/20M

CSCE 532 Homework 5

Problem 4.13: Let . Show that is decidable.

If , then . This is valid under the definition of union because regular languages are closed under union (by Theorem 1.25), and we know and are regular languages because they are each recognized by some regular expression (by Theorem 1.54).

With this knowledge, we now construct the Turing Machine as follows:

“On input , where and are regular expressions:

1. Construct NFAs and such that and using the procedure given in the proof of Lemma 1.55.
2. Construct an NFA such that using the procedure given in the proof of Theorem 1.45.
3. Construct DFAs and such that and using the procedure given in the proof of Theorem 1.39.
4. Run TM as described in the proof of Theorem 4.5 on input . If accepts, accept; otherwise, reject.”

Because accepts input when and rejects otherwise, and because we run

on input such that and , accepts when

and rejects otherwise. By the definition of subset, then, accepts when and rejects otherwise. Thus, is decidable.

Problem 4.28: Let . Show that is decidable. (Hint: An elegant solution to this problem uses the decider for .)

Let be the language consisting of all strings containing . In other words, a string is in if and only if can be generated by the regular expression , where is the alphabet of . We know is regular by Theorem 1.54.

The proof for Problem 2.18 tells us that the intersection of a regular language and a CFL is itself a CFL, so let be a CFL. Theorem 2.9 tells us that there exists some CFG such that .

Let’s create a Turing Machine in the following way:

“On input where is a CFG and is a string:

1. Construct a CFG such that , where is the language described above.
2. Run TM as described in the proof of Theorem 4.8 on input . If accepts, reject; otherwise, accept.”

accepts when rejects and rejects otherwise. In other words, accepts if the intersection of and is nonempty. Because is nonempty only when some string of the form is in both and , we know that accepts only when is a substring of

. Thus, decides .

Problem 4.30: Let be a Turing-recognizable language consisting of descriptions of Turing machines, , where every is a decider. Prove that some decidable language is not decided by any decider whose description appears in . (Hint: You may find it helpful to consider an enumerator for .)

We know is a Turing-recognizable language, so, by Theorem 3.21, there exists an enumerator such that enumerates . We now construct the Turing Machine as follows:

“On input such that :

1. Allow to print until it prints , the -th description of a Turing Machine.
2. Run TM on input . If accepts, reject; otherwise, accept.”

Clearly, decides some language . Assume there exists in some description of a Turing Machine such that . However, as described, does not behave as does for input . This holds for all possible . In other words, behaves opposite of every for at least one input (specifically, the -th input). We’ve now contradicted our previous assumption that for some , so we conclude that is not decided by any decider for which the description appears in .