CSCI 338, Homework #7, Sample Solutions

#1. Problem 5.1.

Assume EQCFG is decidable. Let TM X be the Turing Machine that decides EQCFG.

We can construct grammar GALL, that generates Σ\* over any set of n terminals. This grammar would contain the following n+1 production rules: S 🡪 t1S | t2S | … | tnS | ε where ti is a terminal symbol.

We can then construct TM Y to decide ALLCFG as follows:
 Y = “On input <G>, where G is a CFG
 1. Run X on input <G, GALL> where GALL is defined as above.
 2. If X accepts, then accept. If X rejects, then reject.”

However, Theorem 5.13 proved that ALLCFG is undecidable. A contradiction has been reached! Therefore, our initial assumption is false and EQCFG is undecidable.

#2. Problem 5.3. An infinite number of solutions exist. Two possible solutions are:

|  |  |  |  |
| --- | --- | --- | --- |
| aa | aa | b | ab |
| a | a | a | abab |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ab | ab | aba | b | b | aa | aa |
| abab | abab | b | a | a | a | a |

#3. Problem 5.4. No.

Consider language A = {On1n | n >= 0} over the binary alphabet.

Consider language B = {1} over the binary alphabet.

A computable function can be designed that works as follows

* If w is an element of A, then f(w) = 1
* If not, then f(w) = 0

A is not a regular language, but B is.

#4. LBA Problem. Ryan Darnell’s solution:

**; Machine receives an input in the form L{0,1}+R.**

**; It should increment by one given the current input.**

**; Machine starts in state 0**

**; State 0: Check to see if Left format**

**0 L L r 1**

**0 \* \* r reject**

**; State 1: Validate Binary String**

**1 \_ \_ r reject ; empty or invalid format**

**1 0 0 r 1**

**1 1 1 r 1**

**1 R R l 2 ; move on to increment**

**1 \* \* r reject**

**; State 2: Increment**

**2 0 1 \* accept**

**2 1 0 l 2**

**2 L L \* accept ; input string reached L. Stop**

**; State accept: Move head back to L**

**accept L L \* halt-accept**

**accept \* \* l accept**

**; State reject: Move head to R**

**reject R R \* halt-reject**

**reject \_ \_ \* halt-reject ; no R at end of tape**

**reject \* \* r reject**

#5. Problem A (Θ(n^2)) is mapping reducible to Problem B in constant time.

1. False. A problem (i.e. Θ(n^2)) can’t be transformed into an easier one (i.e. Θ(n)) .
2. True. A problem can be transformed into one of the same complexity.
3. True. A problem can be transformed into a harder one.

#6. Problem A is mapping reducible to Problem B (Θ(n^2)) in constant time.

1. True. A problem can be transformed into a harder one.
2. True. A problem can be transformed into one of the same complexity.
3. False. A problem (i.e. Θ(n^3)) can’t be transformed into an easier one (i.e. Θ(n^2)).