CSE396, Spr'18 Problem Set 10 (the last) Due Thu. 5/10, 11:59pm

The **Final Exam** is **Tuesday, May 15, 11:45–2:45pm** in **121 Cooke Hall**, which is *not the lecture room*. That room needs some special ground rules because the seats have only a foldover for writing: You will be allowed to have *one notes binder* but the textbook is **not** allowed, and *all electronic devices* are forbidden. You are requested to leave your backpack in your car trunk or in some other secure area; if you must bring it into the exam room then it *must be stowed along the side of the room* (and not in front where it would block the small blackboard).

The last week of class will finish up some material from Chapter 5 and then do much of Chapter 7, taking however the "alternative proof" in Chapter 9 of the Cook-Levin Theorem. Chapter 6 is skipped, likewise sections 5.2. This assignment is longer and harder (on *TopHat* too but that aspect is necessary for preparation on the final exam. Some material for the last problem will not be covered formally until Tuesday's lecture, but the previous mentions of running time in lecture have laid groundwork for it. Thursday's lecture next week will include the proof of the Cook-Levin theorem and then facts to fill in the "landscape" of language classes covered in the course. See `https://www.cse.buffalo.edu/~regan/cse396/CSE396S17/CSE396ps10.pdf` for lecture notes on the Cook-Levin proof.

Assignment 10, due Thu. 5/10 at 11:59pm with the usual terms and "stretchy" allowances.

(1) This is the "HW10 Online Worksheet" on *TopHat*, worth 20 pts. as usual. It likewise prepares for the first problem to come on the final exam about classifying languages. But owing to current *TopHat* limitations it has stricter terms about (no) partial credit than the problem on the final will have. Pay especial attention to the difference between c.e. (but undecidable) and co-c.e. (but undecidable), as it corresponds to the difference between a logical $\exists$ quantifier and a logical $\forall$ quantifier.

(2) The *Turing Kit* shows Turing machine tapes that are infinite in both directions but other sources stipulate a sharp left edge at cell 0. Some of those sources say that an attempted left move in cell 0 becomes a stay move, but let us say instead that the TM then *hangs*. There is a simple way to simulate a virtual two-way infinite tape by a one-way infinite tape that is guaranteed not to hang: implement cell $+m$ on *Turing Kit* as cell $2m$ and cell $-m$ as cell $2m - 1$. Every time the *Turing Kit* machine moves a head R move your head R twice and similarly translate L to L twice, unless the move crosses cell 0, in which case you give-or-take an extra step to shift between the "odd track" and the "even track." There is only a roughly-factor-of-2 time slowdown.

Nevertheless, a Turing machine with one-sided tapes that isn't doing this emulation can still hang. It's a reasonable metaphor for real-world "hangs"—where unlike an infinite loop, you know immediately that something has gone wrong. We'd like to make both our Turing machines and our device drivers hang-free, but... So we have this decision problem:

- INSTANCE: A Turing machine M coded for a single one-way-infinite tape.
- QUESTION: Does there exist an input x such that $M(x)$ hangs?

Prove by reduction from any of A_{TM}, K_{TM}, or NE_{TM} (your choice) that this problem is undecidable. Show also that its language *is* computably enumerable. (*Hints:* A technical detail which you should reference in your proof: there is a guaranteed no-hang universal Turing machine U that the machine M' you create in your reduction can call for the "Simulate M on..." body of your code. Your M' will have other code that could be accompanied by the Kingston Trio recording of "Tom Dooley" [you can Google that]. 18 + 6 = 24 pts.)

(3) Consider the following decision problem:

- INSTANCE: Two Turing machines M_1 and M_2.
- QUESTION: Are their languages complementary, i.e., $L(M_1) \cup L(M_2) = \Sigma^*$, and also $L(M_1) \cap L(M_2) = \emptyset$?

Prove that the language of this problem is neither c.e. nor co-c.e. (12 pts.—there are shortcut answers to this question but you must explain fully why they work.)

(4) Consider the following decision problem:

- INSTANCE: A regular expression R over $\Sigma = \{0, 1\}$ that does not have any uses of the star operation; and a number n.
- QUESTION: Is there a binary string x of length n such that x does *not* match R?

Show that the language of this decision problem belongs to the class NP. You will need to argue that the relation "x matches R" belongs to P, i.e., is decidable in deterministic polynomial time. The most convincing way to do so (IMHO—and this works even when R *does* have stars) is to use the theorem in class that converted a regular expression into an equivalent NFA N_R and note the size of the NFA you get in terms of the size of R. Then argue you can simulate N_R on x *without* converting N_R into an equivalent DFA, by instead keeping track of which states of N_R it could possibly be in as you process each bit of x. Give a time bound in terms of the length n of x and the number m of states in N_R; you may also regard m as "order-of" the size of R. (18 pts. for a fully-reasoned answer, making 54 on the handwritten portion and 74 overall points on the set.)

Non-graded extra: Show that this problem is in fact NP-complete by a polynomial-time reduction from 3SAT. Although this is not graded, a key will be given out—actually, this is part of the explanation that will be revealed on *TopHat* to one of the questions on it, and you may use this as a hint toward answering that question.