CSE396, Spring 2019 Problem Set 9 Due Thu. 5/2, 11:59pm

Reading:

The rest of the reading is: Chapter 5, sections 5.1 and 5.3, and Chapter 7, but taking the proof of the NP-completeness of SAT to be the "alternative proof" in Chapter 9. Chapter 6 is skipped, likewise section 5.2. The "computation histories" part of 5.1 may be skimmed in Thursday's lecture or held over to next Tuesday.

I have already jumped over the subject of countability and uncountability in chapter 4, but I recommend *skimming* it now as a way to reinforce your understanding of *functions* and for reviewing "diagonalization" this way: Consider any function f that associates each point a in a set A with a *subset* $f(a)$ of A. The question is, does every subset of A get associated? If A is finite, of size k say, then the answer is "obviously not"—because A has $2^k > k$ subsets but there are only k points to go around. But we can show this more expressly by actually demonstrating a set that doesn't get associated:

$$D_f = \{a \in A : a \text{ is not in the set } f(a)\}.$$

If D_f did get associated—that is, $D_f = f(d)$ for some "set code" $d \in A$—then considering whether $d \in D_f$ or not leads to a contradiction either way. So D_f is not in the range of f. Now the point of Cantor's Theorem is that this logic works the same way even when A is infinite—and when $A = \mathbb{N}$ it leads to the inference that the power set of $\mathbb{N}$ (which under the correspondence between numbers and strings we can think of as the class of *all* languages, decidable and undecidable) is uncountable. But uncountability isn't driving the bus—diagonalization is—so uncountability isn't covered here.

The thing to realize about Chapter 5 is that it is titled "Reducibility" but Sipser defers the key concept until section 5.3. This makes it like watching a movie where the title character doesn't show on screen until you've already finished all the popcorn. This is despite the text foreshadowing it all through Chapter 4. I will put it up-front, so Tuesday's lecture will cover the definitions in section 5.3 before tackling the examples in section 5.1. The chain of reductions will then start by reviewing how the end of last Tuesday's lecture, after defining K_{TM} to be the complement of D_{TM}, essentially reduced K_{TM} to A_{TM} **via** the simple reduction function $f(w) = \langle w, w \rangle$. It will continue by reducing A_{TM} to NE_{TM} (defined as $\{\langle M \rangle : L(M) \neq \emptyset\}$, which is essentially the complement of E_{TM} as defined in the text), and then reducing A_{TM} to the historical Halting Problem.

The *TopHat* portion is lengthier but important for final-exam (p)review. It is still coded as questions with one right answer and a single attempt. However, here are some hints and reminders about facts related to Chapters 2 and 4 that would be given if it had been coded as having 2 attempts:

- The complement of a CFL need not be a CFL, *so* there is no general way to transform a CFG G into a CFG G' such that $L(G') = \sim L(G)$.
- The problem E_{CFG} of whether $L(G) = \emptyset$ is decidable, but the problem ALL_{CFG} of whether $L(G) = \{0,1\}^*$ is undecidable.

- A PDA P can be converted into an equivalent CFG and vice-versa. This plus the decidability of the A_{CFG} problem via Chomsky normal form makes the A_{PDA} problem decidable too. (In fact, though the text doesn't say this, the PDA-to-CFG process is also computationally efficient, so that when we hit Chapter 7 in the last week we will "officially" say that A_{CFG} and A_{PDA} belong to polynomial time, which is stronger than saying all individual CFLs belong to polynomial time.)
- If the answer to "Is $L(M) = \emptyset$" is *no*, this doesn't mean $L(M) = \Sigma^*$. (Although, we will bring this about for certain very special Turing machines M that we construct when doing reductions.)
- We cannot do Cartesian product of two PDAs. Intuitively this is because they might "fight" over control of the single stack a PDA is allowed. But we can do Cartesian product of a PDA and a DFA, so long as the Boolean operation involved is not something like symmetric difference which would involve complementing the PDA (which can't be done in general unless the PDA is deterministic). In particular, this is AOK for $\cap$ as well as $\cup$, so the intersection of a CFL and a regular language is still a CFL.

Homework—part online and all *individual work*—due **Thu.** 5/2, 11:59pm.

(1) Using *TopHat*, the "Worksheet" titled **Spr'19 HW9.1**. There are 10 questions, each worth 2 points, for 20 total. All are unique-answer questions with 1 attempt given. See list of hints above.

(2) Give a decision procedure for the following computational problem:

INSTANCE: A DFA M.

QUESTION: Does $L(M)$ equal its reversal?

You should begin by morphing the DFA M into an NFA N such that $L(N) = L(M)^R$ by making each arc go the other way, making $F_N = \{s_M\}$, and making N have a new start state with ϵ-arcs to all the old final states of M. Ultimately your algorithm should produce a DFA M' such that deciding the E_{DFA} problem on M' tells you the answer about M—it will channel the text's Theorem 4.5 but you should give the steps of how. (18 pts.)

(3) Examine the following decision problem:

INSTANCE: A one-tape Turing machine M and an input x to M.

QUESTION: Does M on input x ever erase a non-blank char by overwriting it by the blank?

Show that it is undecidable. Your answer should include giving your understanding of what lectures said about the ability to code one-tape TMs so they only write blanks over non-blank chars during a final "good housekeeping" stage before accepting—this is prefatory to explaining why deciding this problem would achieve the impossible by deciding A_{TM}. (You need not use a "reduction" *per-se*. 15 pts., for 53 total on the set.)