CSE396, Spring 2018 Problem Set 9 Due Thu. 5/3, 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 sections 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 passed over the subject of countability and uncountability in chapter 4, but you should read it as a way of reinforcing your understanding of *functions* (and the properties of being one-to-one and surjective) which will be key in chapter 5.

Note that the title of Chapter 5 is "Reducibility" but the key concept is not defined until section 5.3. The text regards it as already having been implicit in chapter 4, and my lectures noted how the way the algorithms for E_{DFA} and E_{CFG} were applied to other problems exemplified the "positive side of" reductions. The "negative side"—to prove undecidability and not-c.e./not-co-c.e. by contradiction—is the subject in Chapter 5. I like to put it up-front, so Tuesday's lecture will cover the definitions in section 5.3 before tackling the examples in section 5.1. One thing to be aware of is that the diagonal language D is *implicit* in the text's treatment of the undecidability of the halting problem (in whose proof it is counterfactually referred to as a "machine"); my lecture last Tuesday made it *explicit* but really covered the same logic. I will use what is actually the old standard name "K" for its complement, namely the set of programs that *do* accept their own code. Then K is likewise undecidable. The chain of reductions will then start $K \leq_m A_{TM} \leq_m NE_{TM}$ plus reducing A_{TM} to the historical Halting Problem.

Also review these notes in conjunction with previous lectures and with questions 1–6 of this HW9 on "TopHat": Although the intersection of two DCFls need not even be a CFL, the DCFLs *are* closed under any Boolean operation with a regular set. For instance, let M_1 be a DPDA and let M_2 be a DFA. Using the Cartesian product *idea*, we can combine the code of M_2 into that of M_1 so that *whenever* the DPDA M_1 makes an R-move upon reading a character c on the input tape, the DFA M_2 also gets stepped forward on c. If M_1 stays stationary, M_2 does not get stepped but is held paused. Provided M_1 has been edited so that it always reads all of its input x (without being stuck in a loop, that is), M_2 is able to read all of x as well. Then the combined machine M_3 gives an answer according to the Boolean combination of final states q_1, q_2 reached by M_1 and M_2 in the same way as when we had two DFAs, e.g. $(q_1 \in F_1 \wedge q_2 \in F_2)$ for intersection, $(q_1 \in F_1 \vee q_2 \in F_2)$ for union, and $(q_1 \in F_1 \text{ XOR } q_2 \in F_2)$ for symmetric difference. (The main reason why this doesn't work for two DPDAs is that they might "clash" both for control of the stack, one wanting to push where the other wants to pop. Also note that if M_1 is an NPDA then the idea still works for $\cap$ and $\cup$ but not for symmetric difference—indeed, it *can't* work for symmetric difference because $L \triangle \Sigma^* = \tilde{L}$ but the CFLs are not closed under complements.)

Assignment 9, due Thu. 5/3 at 11:59pm with the previous terms and "stretchy" allowances.

(1) This is the "HW9 Online Worksheet" on *TopHat*, worth 20 pts. as usual. It 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 in instances analogous to Q2–Q6 in particular.

(2) Say that a state q in an NFA $N = (Q, \Sigma, \delta, s, F)$ is "live" if there is a string $y \in \Sigma^*$ and state $r \in F$ such that N can process y from q to r. To keep things simple, we will suppose that N has no ϵ-arcs.

(a) Give a decision procedure to identify all live states. Pseudocode like in lectures and the text is fine, but your code sketch must include one while-loop (or for-loop) with a flag for change/no-change. (12 pts.)

(b) Briefly explain why the NFA N' obtained by deleting all non-live states than s (note: there may be ones that aren't so obviously a "dead state") and their incoming edges is equivalent to N, i.e., $L(N') = L(N)$. (If s is not live then N' has just the rejecting state s and no arcs. 6 pts.)

(c) Also say q is "reachable" if there is a string that N can process from s to q. Consider N'' obtained by similarly stripping out unreachable states. Then prove that $L(N)$ is infinite if and only if the graph of N'' has a cycle. NB: a self-loop counts as a cycle. (6 pts. for the $\Longrightarrow$ part, 3 pts. for $\Longleftarrow$).

(d) Conclude by stating a decision procedure for the decision problem, given an NFA N, is $L(N)$ infinite? (6 pts., for 30 on the problem)

It must be repeated that you are not allowed to trawl the Internet looking for solutions to (parts of) this problem. But you *are* allowed to lift things from the text that may act as guides—here we ask you to note and cite them as such. If you have had this or a similar problem in CSE331 or a similar course, please let us know and we will let you use it—again with citation.

(3) Consider the following decision problem:

INSTANCE: A DFA $M = (Q, \Sigma, \delta, s, F)$ with $\Sigma = \{a, b\}$ and a string $w \in \Sigma^*$.
QUESTION: Is there a string $y \in \Sigma^*$ such that wy is a palindrome and M accepts y?

(a) Give in pseudocode a decision procedure for this problem. (18 pts. For a hint, the most important machine for you to consider and modify is one not stated in the problem, namely an NPDA N recognizing the language of palindromes. Note that y need not equal w^R; you have to consider cases like $w = abab$ and $y = a$ or $y = bbaba$ too.)

(b) What happens if the final condition condition is changed to "M accepts wy?" Tweak your answer accordingly (6 pts., for 24 on the problem and 74 on the set)

For a footnote, one could ask what happens when M is allowed to be a DPDA. Then I right now do not know whether the problem is decidable. Some questions like that, most notably whether two DPDAs accept the same language (EQ_{DPDA} in the text's naming scheme), have taken humanity decades to answer. EQ_{DPDA} was proved decidable after the text was written and the proof is over 100 journal pages long.