CSE396, Spring 2026 **Problem Set 10 (the last)** **Due Tue. 5/5, 11:59pm**

This homework is due on **Tuesday**, May 5. Some parts, including the last few Tophat questions, depend on the Thursday 4/30 lecture—whose reading was specified on Assignment 9. The reading for the last lecture, on May 5, is to the end of Chapter 7. The lecture will *skim* the NP-completeness examples about graphs from section 7.5, and may not mention the "Subset Sum" problem at all. That lecture will conclude with a course flyover that will pick up some examples that were skipped (notably in section 5.1) and the "big picture" of complexity classes today.

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

(1) Using *TopHat*, the "Worksheet" titled **Spr 26 HW10 Online Part**. Ten questions, each worth 2 points, for 20 total. They continue being advance practice for the language classification problem (1) on the final exam.

(2) Consider context-free grammars G over an alphabet Σ that includes the character 2 as well as $0, 1$. Here are three decision problems we can frame about them:

(a) Is $L(G) \subseteq \{0,1\}^*$? That is, does G fail to derive any strings that have the character 2?

(b) Is $L(G) \cap \{0,1\}^* \neq \emptyset$? That is, does G derive any strings that omit the character 2?

(c) Is $L(G) \supseteq \{0,1\}^*$? That is, does G derive all strings that omit the character 2?

Although these problems sound similar, the first two are in P whereas the last one is *undecidable*. Give polynomial-time algorithms for (a) and (b), using loops similar to those given in lecture to show that the problems of whether $\epsilon \in L(G)$ and whether $L(G) \neq \emptyset$ are polynomial-time decidable.

Then summarize why (c) is undecidable. Note that when the ALL_{CFG} problem was shown to be undecidable, it was for grammars G' using large alphabets called Γ' that came from both the Q and Γ of one-tape Turing machines M—but since each individual Γ' is finite, it can be re-coded over binary strings in a way that carries through to PDAs and grammars. Thus you can get a grammar G'' such that ALL_{CFG} is true for G' over Γ' exactly when $L(G'') = \{0,1\}^*$. Your summary just needs to demonstrate your understanding of this and transfer it to the context of part (c). (The footnote at the end of this problem set is not directly relevant. 30 pts. total: 24 for the algorithms and 6 for (c).)

(3) Suppose you are choosing from among n books for a school's summer reading list—or maybe for the syllabus of a large lit survey course. You ask a bunch of colleagues for suggestions and get recommendations of three basic types:

- *Definite:* Choose this book. Or: *don't* choose this book.
- *Positive:* Choose at least one of these two-or-three books.
- *Negative:* Don't choose all of these two-or-three books—that would be too much in this category. Not choosing any of them is fine too.

So the INSTANCE is a list of n possible books and a number m of recommendations of the above kinds. Your QUESTION is, can you choose a list of books that obeys each recommendation? There is no condition on the size $\ell \leq n$ of your list.

Prove that the decision problem framed by this question is NP-complete. You may want to connect your proof directly to the kind of formulas used in the Thu. 4/30 lecture to show that 3SAT is NP-complete, not just use the general 3SAT problem. (24 pts., for 74 on the set)

Footnote about grammars and undecidability. The text in the latter half of section 5.1 "pulls a fast one" on page 226 in the middle of proving that the ALL_{CFG} problem is undecidable. We can give the tweaked name V'_M to what it defines there: valid accepting computations by the one-tape TM M *in which every second ID is written in reverse.* (That the text uses # rather than my comma between IDs, and labels IDs $C_1, C_2, C_3, \ldots$ rather than my $I_0(x), I_1, I_2, \ldots$ are only cosmetic differences.)

- The reason the text does this is that it pictures a nondeterministic PDA guessing which j causes $I_j \vdash_M I_{j+1}$ to fail, like I did, and then follows through with the NPDA copying I_j onto its stack.
- The reversal then allows it to compare I_{j+1} char-by-char while popping from I_j on its stack, but with the mindset of verifying that the 2-or-3 char difference between the IDs does **not** follow by an instruction of M.
- That is intuitively leveraging the fact that the complement of the palindromes language (with-or-without the middle marked with a # character) is also a CFL. (The unmarked language PAL is the nicest example of a CFL whose complement is also a CFL but it is still not a DCFL; the proof that it is not a DCFL is beyond our scope.)
- I was able to avoid this tweak because we saw (on homework) that the complement of the double-word language is a CFL. I especially wanted to keep things simple because I was doing this on the chalkboard.

The V'_M tweak does, however, have one further consequence worth mentioning. V'_M itself is *not* as CFL. It can't be: whether a CFL is empty is decidable, whereas $V'_M = \emptyset \iff L(M) = \emptyset$ just as for V_M. Intuitively, a PDA fails to recognize V'_M because while it can use its stack to verify $I_0 \vdash I_1$, at that moment it has erased its stack entirely and has nothing left to compare I_1 against I_2. However, it could go on to verify $I_2 \vdash_M I_3$ and then $I_4 \vdash_M I_5$ and so on. A second DPDA could handle the skipped-over $I_1 \vdash_M I_2$ and $I_3 \vdash_M I_4$ etc. cases. The upshot is this:

Theorem: The language V'_M is the intersection of two DCFLs. In consequence, the problem of whether the intersection of two DCFLs is empty is undecidable.

We can give this problem a name: $E_{\cap DCFL}$. It follows that $E_{\cap CFL}$ is undecidable, and likewise $NE_{\cap CFL}$ which is just its complement.

Moving on, the problem $ALL_{\cap CFL}$ is undecidable "by restriction": $A_1 \cap A_2 = \Sigma^* \iff$ both A_1 and A_2 equal Σ^*. So we can restrict the instance A_1 to be given by the simple grammar $G_1 = S \rightarrow 0S \mid 1S \mid \epsilon$ which we already know has $L(G_1) = \{0, 1\}^*$, so the question boils down to whether a given CFG G_2 gives $L(G_2) = \{0, 1\}^*$, which is just ALL_{CFG}. But the same token makes $ALL_{\cap DCFL}$ decidable, because ALL_{DCFL} is decidable (in polynomial time, in fact).