University of Nevada, Las Vegas Computer Science 456/656 Fall 2020

Answers to Assignment 4: Due Monday November 9, 2020

Name:_____

You are permitted to work in groups, get help from others, read books, and use the internet. Your answers must be written in a pdf file and emailed to the graudate assistant, Shekhar Singh `shekhar.singh@unlv.edu` by 23:59 November 6. Your file must not exceed 10 megabytes, and must print out to at most 8 pages.

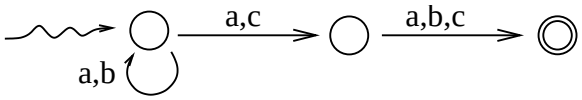
1. Suppose L is the regular language accepted by an NFA M with p states. Prove that L has [regular] pumping length p.

   Without loss of generality, M has no self-loop transition labeled λ, since that transition would be useless. Suppose $w \in L$ and $|w| \geq p$. There is a computation of M which accepts w in at last p steps. Each step traverses an arc of M, and hence must visit a state at least $p+1$ times. Since M has p states, the computation must visit some state, say q, at least twice. The portion of the computation between the two visitations of q corresponds to a non-empty substring of w, which is then a non-empty pumpable substring.

   Furthermore, note that we traverse a loop during the first p steps. Thus, we can be sure that $|xy| \leq p$.

   steps.

2. Find an NFA with three states whose equivalent minimal DFA has eight states.



3. Given alphabets Σ_1 and Σ_2, a *homomorphism* from Σ_1 to Σ_2 is a function $h : \Sigma_1 \to \Sigma_2^*$. For example, any Huffman code on an alphabet Σ is a homomorphism from Σ to the binary alphabet. If $h : \Sigma_1 \to \Sigma_2^*$ is a homomorphism and $w \in \Sigma_1^*$, we define $h(w)$ to be the string obtained by replacing each symbol x of w by $h(x)$. (For example, if h is the huffman code $a \mapsto 10$, $b \mapsto 1110$, $c \mapsto 1111$, $d \mapsto 110$, $e \mapsto 0$, then $h(bed) = 11100110$.) If $L \subseteq \Sigma_1^*$, let $h(L) = \{h(w) : w \in L\}$. If L is regular, is it always true that $h(L)$ is regular? Why do you believe that?

   Suppose that M is an NFA which accepts L. Define a new NFA M_2 which accepts $h(L)$ as follows by replacing the label, say a, of each arc with $h(a)$. It may be necessary to introduce additional states. For example, suppose $h(a) = b$, $h(b) = \lambda$, and $h(c) = ab$. An arc labeled a in M becomes an arc labeled b in M_2, an arc labeled b in M becomes an arc labeled λ in M_2, and an arc labeled c in M becomes an arc labeled a to a new state, which is not a final state, followed by an arc labeled c from that new state.

4. What is the minimum [regular] pumping length of the language of all decimal numerals for multiples of three? Exact answer, please. Warning: The empty string is not a decimal numeral.

   There is a DFA with 4 states which accepts the language, so by Problem 1, the minimum pumping length cannot be more than 4. Since the string 111 is in the language and does not have a pumpable substring, the minimum pumping length must be greater than 3. Thus, the minimum pumping length is 4.

5. We say that a set D of vertices of a graph G *dominates* G if every vertex of G is adjacent to some member of D. The Dominating Set problem is, given a graph G and a number k, does G have a dominationg set of order (that is, size) k? From what we've covered so far in class, we know that SAT, 3-SAT, Independent set, Partition, Subset Sum, and regular expression equivalence are all $\mathcal{NP}$-complete. Using that knowledge, prove that the dominating set problem is $\mathcal{NP}$-complete

   This is the proof I gave in class:

   `https://www.chegg.com/homework-help/questions-and-answers/goal-following-questions-show-dominating-set-problem-np-complete-reduction-3-sat-construct-q24999585`

6. What is the minimum [context-free] pumping length of $L = L_{\text{Dyck}} \backslash \{\lambda\}$?

   The answer is 3. L has no strings of length 3, so if $w \in L$ and $|w| \geq 3$, then $|w| \geq 4$, and w must have a pumpable substring ab.

7. Find a context-free language which is not accepted by any DPDA. Justify your answer.

   Let L be the set of palindromes over any alphabet with more than one symbol. L cannot be accepted by any DPDA, since the machine has no way of knowing when it has reached the middle of the string.

8. Consider the following problems:

   (a) The furniture placement problem. Given a room of certain dimensions, and given a set of pieces of furniture, it is possible to place all the furniture into the room? You are permitted to lower furniture through the ceiling with a crane.

   The answer is that the problem is $\mathcal{NP}$-complete. The problem is clearly $\mathcal{NP}$, since if the furniture fits, we can verify that fact in polynomial time. We reduce the partition problem to the furniture moving problem. Supose $x_1, x_2 \ldots x_n$ is an instance of the partition problem. We can assume each number is an integer. Let $S = \sum_{i=1}^{n}$. For each i, let f_i be a rectangle of dimension $\frac{1}{4} \times x_i$, and let the room R be a $\frac{1}{4} \times \frac{S}{2}$ rectangle. Note that the total area of the furniture equals the area of the room, hence a solution would give an exact fit, which implies that each f_i must fit into R with its long side parallel to the long side of R. The top half of the room is a $\frac{1}{8} \times \frac{S}{2}$ rectangle and holds exactly half, measured by area, of the furniture. Thus the set of x_i such that f_i is in the top half has sum $\frac{S}{2}$.

   (b) The furniture moving problem. Given a room of certain dimensions, with a given door, and given a set of pieces of furniture, is it possible to move all the furniture into the room through the door?

   Refer to the Euler diagram of complexity classes handed out earlier, what is the smallest of those complexity classes that is known to contain the furniture placement problem, and what is the smallest of those complexity class that is known to contain the furniture moving problem? I am not asking for proofs.

   The answer is $\mathcal{P}$–SPACE complete. I will not give a proof, but you should recognize that the furniture moving problem is a special case of the sliding block problem, which is knwon to be $\mathcal{P}$–SPACE complete.

9. Let $L = \{a^n b^n c^n : n \geq 1\}$. Let G be the context-sensitive grammar with productions:
$S \to abc$
$ab \to aaAbb$
$Ab \to bA$
$Ac \to cc$
Does G generate L? Justify your answer.

   There is a proof that this is a grammar for L, but I would not expect you to write that proof. Instead, we first note that no string generated by that grammar can contain substring ba, ca, or cb, hence any string in the language is of the form $a^i b^j c^k$, and we give derivations for $aabbcc$ and $aaabbbccc$.
$S \Rightarrow abc \Rightarrow aaAbbc \Rightarrow aabAbc \Rightarrow aabbAc \Rightarrow aabbcc$
$S \Rightarrow abc \Rightarrow aaAbbc \Rightarrow aabAbc \Rightarrow aabbAc \Rightarrow aabbcc \Rightarrow aaaAbbbcc \Rightarrow aaabAbbcc \Rightarrow aaabbAbcc \Rightarrow aaabbbAcc \Rightarrow aaabbbccc$

   To prove that $i = j = k$ for any $w = a^i b^j c^k$ generated by G, we define two invariants:

   (a) Inv_1: $\#_a = \#_b$

   (b) Inv_2: $\#_a = \#_c + \#_A$

   Both invariants hold at the beginning of each derivation, since $\#_a = \#_b = \#_c = \#_A = 0$ for S. We can check, using the production rules, that any step of a derivation maintains both invariants. Thus, at the end of the derivation both invariants hold, and we have $\#_a = \#_b$ by Inv_1 and $\#_b = \#_c$ by Inv_2 since $\#_A = 0$.

10. Prove that the context-free grammar equivalence problem is co-RE.

    The grammar equivalence problem corresponds to the language of all strings of the form $\langle G_1 \rangle \langle G_2 \rangle$ such that G_1 and G_2 are context-free grammars. and that $L(G_1) = L(G_2)$. We can assume that the binary alphabet is the terminal alphabet of both grammars. To show that language is in co-R.E., we need to show that its complement is in R.E.
    Let L be that complement. Suppose $w \in L$. There are two cases.
    Case 1: $w = \langle G_1 \rangle \langle G_2 \rangle$, where G_1 and G_2 are context-free grammars and $L(G_1) \neq L(G_2)$. Choose a string There must be a string in one language but not the other. The following program accepts L:

    For all $w \in \{0.1\}^*$ in canonical order
        Use the CYK algorithm to decide whether $w \in L(G_1)$
        Use the CYK algorithm to decide whether $w \in L(G_2)$
        If the two answers are different HALT.

    Case 2: w is not of the form $\langle G_1 \rangle \langle G_2 \rangle$ for context-free grammars G_1 and G_2. (w could be any nonsense string.) The fact that w is not of that form can be easily determined, and we are done.

11. Is every decidable language context-sensitive? You may give an answer you find on the internet. If you do that, give the url.

    The following Wikipedia page states that not every decidable language is context-sensitive:
    `https://en.wikipedia.org/wiki/Nondeterministic_Turing_machine`