Computer Science 456/656 Fall 2020 Practice for Examination November 16, 2018

Some of these problems are copied from the practice2. Those deal with issues that were not on Test 2.

1. All the problems of Assignment 3.

2. All the problems of Assignment 4.

3. True or False.T = true, F = false, and O = open, meaning that the answer is not known to science at this time. The first 42 questions should be familiar to you.

   (i) **F** Let L be the language over $\{a, b, c\}$ consisting of all strings which have more a's than b's and more b's than c's. There is some PDA that accepts L.

   (ii) **T** The language $\{a^n b^n \mid n \geq 0\}$ is context-free.

   (iii) **F** The language $\{a^n b^n c^n \mid n \geq 0\}$ is context-free.

   (iv) **T** The language $\left\{a^i b^j c^k \mid j = i + k\right\}$ is context-free.

   (v) **T** The intersection of any three regular languages is context-free.

   (vi) **T** If L is a context-free language over an alphabet with just one symbol, then L is regular.

   (viii) **T** The set of strings that your high school algebra teacher would accept as legitimate expressions is a context-free language.

   (ix) **T** Every language accepted by a non-deterministic machine is accepted by some deterministic machine.

   (x) **T** The problem of whether a given string is generated by a given context-free grammar is decidable.

   (xi) **T** If G is a context-free grammar, the question of whether $L(G) = \emptyset$ is decidable.

   (xii) **F** Every language generated by an unambiguous context-free grammar is accepted by some DPDA.

   (xiii) **T** The language $\{a^n b^n c^n d^n \mid n \geq 0\}$ is recursive.

   (xiv) **T** The language $\{a^n b^n c^n \mid n \geq 0\}$ is in the class $\mathcal{P}$-TIME.

   (xv) **O** There exists a polynomial time algorithm which finds the factors of any positive integer, where the input is given as a binary numeral.

   (xvi) **F** Every undecidable problem is $\mathcal{NP}$-complete.

   (xvii) **F** Every problem that can be mathematically defined has an algorithmic solution.

   (xviii) **F** The intersection of two undecidable languages is always undecidable.

   (xix) **T** Every $\mathcal{NP}$ language is decidable.

(xx) **T** The intersection of two $\mathcal{NP}$ languages must be $\mathcal{NP}$.

(xxii) **O** $\mathcal{NC} = \mathcal{P}$.

(xxiii) **O** $\mathcal{P} = \mathcal{NP}$.

(xxiv) **O** $\mathcal{NP} = \mathcal{P}$-SPACE

(xxv) **O** $\mathcal{P}$-SPACE = EXP-TIME

(xxvi) **O** EXP-TIME = EXP-SPACE

(xxvii) **T** The traveling salesman problem (TSP) is $\mathcal{NP}$-complete.

(xxviii) **T** The knapsack problem is $\mathcal{NP}$-complete.

(xxix) **T** The language consisting of all satisfiable Boolean expressions is $\mathcal{NP}$-complete.

(xxx) **T** The Boolean Circuit Problem is in $\mathcal{P}$.

(xxxi) **O** The Boolean Circuit Problem is in $\mathcal{NC}$.

(xxxii) **F** If L_1 and L_2 are undecidable langugages, there must be a recursive reduction of L_1 to L_2.

(xxxiv) **T** The language consisting of all strings over $\{a, b\}$ which have more a's than b's is context-free.

(xxxv) **T** 2-SAT is $\mathcal{P}$-TIME.

(xxxvi) **O** 3-SAT is $\mathcal{P}$-TIME.

(xxxvii) **T** Primality is $\mathcal{P}$-TIME.

(xxxviii) **F** There is a $\mathcal{P}$-TIME reduction of the halting problem to 3-SAT.

(xxxix) **T** Every context-free language is in $\mathcal{P}$.

(xl) **T** Every context-free language is in $\mathcal{NC}$.

(xli) **T** Addition of binary numerals is in $\mathcal{NC}$.

(xliii) **F** Every language generated by a general grammar is recursive.

(xliv) **F** The problem of whether two given context-free grammars generate the same language is decidable.

(xlv) **T** The language of all fractions (using base 10 numeration) whose values are less than π is decidable. (A *fraction* is a string. "314/100" is in the language, but "22/7" is not.)

(xlvi) **T** There exists a polynomial time algorithm which finds the factors of any positive integer, where the input is given as a unary ("caveman") numeral.

(xlvii) **T** For any two languages L_1 and L_2, if L_1 is undecidable and there is a recursive reduction of L_1 to L_2, then L_2 must be undecidable.

(xlviii) **F** For any two languages L_1 and L_2, if L_2 is undecidable and there is a recursive reduction of L_1 to L_2, then L_1 must be undecidable.

(xlix) As you may have learned, there is a formal language which can be used to write any mathematical proposition as well as any proof of any mathematical proposition, and an algorithm exists that can check the correctness of such a proof. In 1978, Jack Milnor https://en.wikipedia.org/wiki/John_Milnor told me that in the future no proof will be accepted unless it can be verified by a computer.

F If P is a mathematical proposition that can be written using a string of length n, and P has a proof, then P must have a proof whose length is $O(2^{2^n})$.

(li) **F** Every bounded function is recursive.

(lii) **O** If L is $\mathcal{NP}$ and also co-$\mathcal{NP}$, then L must be $\mathcal{P}$.

(liii) **T** Recall that if $\mathcal{L}$ is a class of languages, co-$\mathcal{L}$ is defined to be the class of all languages that are not in $\mathcal{L}$. Let $\mathcal{RE}$ be the class of all recursively enumerable languages. If L is in $\mathcal{RE}$ and also L is in co-$\mathcal{RE}$, then L must be decidable.

(liv) **T** Every language is enumerable.

(lv) **F** If a language L is undecidable, then there can be no machine that enumerates L.

(lvi) **T** There exists a mathematical proposition that can be neither proved nor disproved.

(lvii) **T** There is a non-recursive function which grows faster than any recursive function.

(lviii) **T** There exists a machine[1] that runs forever and outputs the string of decimal digits of π (the well-known ratio of the circumference of a circle to its diameter).

(lix) **F** For every real number x, there exists a machine that runs forever and outputs the string of decimal digits of x.

(lx) **O** **Rush Hour**, the puzzle sold in game stores everywhere, generalized to a board of arbitrary size, is $\mathcal{NP}$–complete.

(lxi) **O** There is a polynomial time algorithm which determines whether any two regular expressions are equivalent.

(lxii) **T** If two regular expressions are equivalent, there is a polynomial time proof that they are equivalent.

(lxiii) **F** $\mathcal{P}$–TIME = EXP–TIME.

(lxiv) **F** $\mathcal{P}$–SPACE = EXP–SPACE.

(lxv) **T** Multiplication of integers is in $\mathcal{NC}$.

(lxvi) **T** Every context-free language is in $\mathcal{NC}$.

(lxviii) **T** Any language generated by an unrestricted [general] grammar is recursively enumerable.

(lxix) **O** If there is a polynomial time reduction of L_1 to L_2, and L_1 is $\mathcal{P}$-complete and L_2 is $\mathcal{P}$, then L_2 must be $\mathcal{P}$-complete.

[1] As always in automata theory, "machine" means abstract machine, a mathematical object whose memory and running time are **not** constrained by the size and lifetime of the known (or unknown) universe, or any other physical laws. If we want to discuss the kind of machine that exists (or could exist) physically, we call it a "physical machine."

For the following four questions, let Σ_1 and Σ_2 be alphabets, and $h : \Sigma_1 \to \Sigma_2^*$, a homomorphism.

(lxx) **T** If $L \subseteq \Sigma_1^*$ is regular, then $h(L)$ is regular.

(lxxi) **T** If $L \subseteq \Sigma_2^*$ is regular, then $h^{-1}(L)$, defined to be $\{w \in \Sigma_1^* : h(w) \in L\}$, is regular.

(lxxii) **T** If $L \subseteq \Sigma_1^*$ is context-free, then $h(L)$ is context-free.

(lxxiii) **T** If $L \subseteq \Sigma_1^*$ is decidable, then $h(L)$ is decidable.

4. State the pumping lemma for regular languages. (Your answer must be correct in its structure, not just the words you use. Even if all the correct words are there, you could get no credit if you get the logic wrong.)

5. State the pumping lemma for context-free languages. (Your answer must be correct in its structure, not just the words you use. Even if all the correct words are there, you could get no credit if you get the logic wrong.)

   This is given in a previous document.

6. Give a context-free grammar for the language of all strings over $\{a, b, c\}$ of the form $a^n bc^n$ for $n \geq 0$.
   $S \to aSc$
   $S \to b$

7. The following context-free grammar G is ambiguous. Give an equivalent unambiguous grammar.

   1. $E \to E + E$
   2. $E \to E - E$
   3. $E \to E * E$
   4. $E \to - E$
   5. $E \to (E)$
   6. $E \to a$
   7. $E \to b$
   8. $E \to c$

   This is given in a previous document.

8. Let L be the language generated by the Chomsky Normal Form (CNF) grammar given below.

   (i) $S \to a$

   (ii) $E \to a$

   (iii) $S \to LA$

   (iv) $E \to LA$

   (v) $L \to ($

   (vi) $A \to ER$

   (vii) $R \to)$

   (viii) $S \to PE$

   (ix) $E \to PE$

   (x) $S \to EE$

   (xi) $E \to EE$

   (xii) $P \to EQ$

   (xiii) $Q \to +$

   This is given in a previous document.

   Use the CYK algorithm to prove that the string $a(a+a)$ is a member of L. Use the figure below for your work.

   This is given in a previous document.

9. Consider the NFA whose transition diagram is in Figure 1 below. where the input alphabet is $\{a, b, c\}$. Draw the transition diagram of an equivalent minimal DFA. Show your steps.

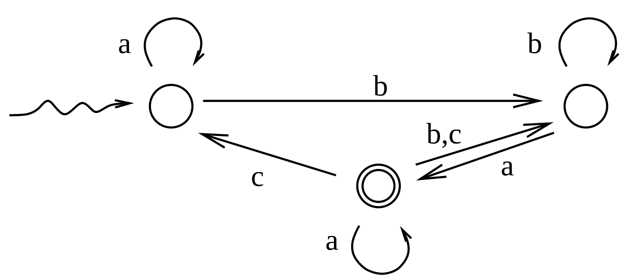

Figure 1: Find a minimal DFA equivalent to this NFA

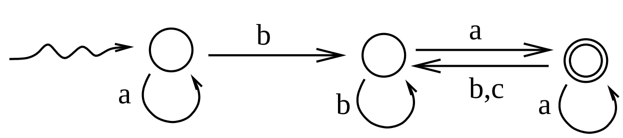

Figure 2: Minimal DFA

10. Let $L = \left\{w \in \{a,b\}^* \mid \#_a(w) = 2\#_b(w)\right\}$, here $\#_a(w)$ denotes the number of instances of the symbol a in the string w. For example, $aaababaaabba \in L$, because that string has the twice as many a's as b's. Give a context-free grammar for L. Your grammar may be ambiguous.

    This is given in a previous document.

11. Give a conext-sensitive grammar for $\{a^n b^n c^n \,:\, n \geq 1\}$

    This is given in a previous document.

12. Give a definition of each of these $\mathcal{NP}$-complete languages/problems.

    (i) 3-SAT

    (ii) Independent Set

    (iii) Dominating Set

    (iv) Subset Sum

    (v) Partition

    These problems are all defined in documents I have written that are linked to on the assignment web page.

13. Give a general (unrestricted) grammar for the language consisting of all string of 1's of length a power of 2, that is, $\left\{1^{2^n}\right\}$

    $S \to L1R$
    $L \to LD$
    $D1 \to 11D$
    $DR \to R$
    $L \to \lambda$
    $R \to \lambda$

14. State the Church-Turing thesis. Why it is important?

    Any computation that can be done by any machine can be done by some Turing Machine.

    This is important because it gives us a way to prove that something cannot be computed. All we have to prove, is that no Turing machine can compute it.

15. Expect a reduction problem involving $\mathcal{NP}$-completeness.