University of Nevada, Las Vegas Computer Science 456/656 Fall 2019

Answers to Assignments 6 and 7: Due November 13, 2019

1. True or False. T = true, F = false, and O = open, meaning that the answer is not known science at this time. In the questions below, $\mathcal{P}$ and $\mathcal{NP}$ denote $\mathcal{P}$-TIME and $\mathcal{NP}$-TIME, respectively.

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

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

   (iii) 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.

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

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

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

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

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

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

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

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

   (xii) T The clique problem is $\mathcal{NP}$-complete.

   (xiii) T The traveling salesman problem is $\mathcal{NP}$-hard.

   (xiv) T The union of two $\mathcal{NP}$ languages must be $\mathcal{NP}$.

   (xv) F OR O The intersection of two $\mathcal{NP}$-complete languages must be $\mathcal{NP}$-complete.

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

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

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

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

   (xx) O EXP-TIME = EXP-SPACE

   (xxi) T There is a deterministic parser for any context-free grammar.

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

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

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

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

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

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

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

(xxix) T 2-SAT is $\mathcal{P}$–TIME.

(xxx) O 3-SAT is $\mathcal{P}$–TIME.

(xxxi) T Primality, where the input is written in binary, is $\mathcal{P}$-TIME.

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

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

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

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

(xxxvi) O Every context-sensitive language is in $\mathcal{P}$.

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

(xxxviii) T Every language generated by a general grammar is recursively enumerable.

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

(xl) T The problem of whether two given context-free grammars generate the same language is co–$\mathcal{RE}$.

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

(xlii) 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.)

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

(xliv) 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.

(xlv) 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.

(xlvi) O OR F If L is any $\mathcal{NP}$ language, there must be a $\mathcal{P}$–TIME reduction of the partition problem to L.

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

(xlviii) 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.

(xlix) T Every language is enumerable.

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

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

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

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

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

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

(lvi) O There is a well-defined function f on positive integers, where:

$f(n) = 0$ if $n = 1$

$f(n) = 1 + f(n/2)$ if n is even

$f(n) = 1 + f(3n + 1)$ if n is odd and greater than 1.

For example, $f(1) = 0$, $f(2) = 1$, $f(3) = 7$, $f(4) = 2$, $f(5) = 5$, $f(6) = 8$, $f(7) = 16$, ...

Hint: look on the internet for "Collatz."

(lvii) F The *busy beaver* function is recursive.

(lviii) F The Post correspondence problem is $\mathcal{NP}$–COMPLETE.

2. Suppose x and y are positive integers, and their binary numerals $\langle x \rangle$ and $\langle y \rangle$ each have length n. Then $\langle xy \rangle$, the binary numeral of their product, has length at most $2n$. Explain how the problem of computing $\langle xy \rangle$ from $\langle x \rangle$ and $\langle y \rangle$ is in the class $\mathcal{NC}$.

   Using the grade-school algorithm for multiplication, $\langle xy \rangle$ is is obtained by taking the sum of n n-bit strings, each of which is either a copy of $\langle x \rangle$ shifted some number of places, or a string of zeros. By adding pairs, we obtain $n/2$ $(n+1)$-bit strings. We add those in pairs to obtain $n/4$ strings, and so forth, until we have $\langle xy \rangle$, one binary string of length $2n$. There are $\log n$ of these steps, each of which takes $O(\log n)$ time using n^2 processors. Thus, $\langle xy \rangle$ can be computed in $O(\log^2 n)$ time uisng n^2 processors.

3. Let L be the language generated by the context-free grammar below. What is the minimum pumping length of L? (Note that this grammar does not contain the production $S \to iS$.) Hint: read `http://web.cs.unlv.edu/larmore/Courses/CSC456/pumping.pdf`

   $S \to wS$

   $S \to iSeS$

   $S \to a$

The answer is 3.

Because we are using w in the grammar, we will replace w in the statement of the pumping lemma by s. Case 1. The string contains w. Then w can be eliminated, or replaced with w^i for any i.

Case 2. s does not contain w and has length at least 3. Then s must contain i and e. Consider the bottom-most i in the parse tree. That i must be followed by ae. Write $s = uiaex$, and let $y = z = \lambda$. Then ux, $uiaeiaex$, *etc.* are in L.

4. Explain to me why $\mathcal{NP}$–TIME $\subseteq \mathcal{P}$–SPACE.

   Let $L \in \mathcal{NP}$–TIME. Then there is some NTM M and some k such that M accepts any member $w \in L$ within n^k steps for $n = |w|$. Let g be the guide string of such a computation, a string of length at most n^k.

   If $|w| = n$, generate each guide string of length n^k in canonical order. Emulate M with input w using each guide string, erasing all memory except the last guide string each time, using $O(n^k)$ space. If $w \in L$, there will eventually be an emulation which accepts w, if $w \notin L$, no emulation will accept w. Thus $\mathcal{NP}$–TIME $\subseteq \mathcal{P}$–SPACE.

5. Recall that a fraction is a string. If x is any real number, let LESS_x be the set of fractions whose values are less than x, and let MORE_x be the set of fractions whose values are more than x.

   (a) Is it true that, for every real number x, LESS_x is decidable?

   (b) Is it true that, for every real number x, MORE_x is decidable?

   (c) Is there a real number x such that LESS_x is decidable but MORE_x is not decidable?

   (d) Is there a real number x such that LESS_x is recursively enumerable but MORE_x is not recursively enumerable?

   Hint: If L is a language over the unary alphabet $\{1\}$, let $x_L = \sum_{i=0}^{\infty} 2^{-a_i}$, where $a_i = 1$ if $1^i \in L$, and $a_i = 0$ if $1^i \notin L$. Equivalently, we write $x_L = \sum_{1^i \in L} 2^{-i}$. Note that $x_L = 0$ if $L = \emptyset$, $x_L = 2$ if $L = \{1\}^*$, and $0 < x_L < 2$ for all other choices of L. Depending on whether L is decidable, or whether L is recursivly enumerable, is LESS_{x_L} decidable? Recursively enumerable?

   LESS_x and MORE_x are (trivially) decidable of x is rational, thus, without loss of generality, x is irrational. This implies that MORE_x is the complement of LESS_x, which implies that MORE_x is decidable if and only if LESS_x is decidable.

   As you might have guessed, LESS_{x_L} is decidable if and only if L is decidable, and LESS_{x_L} is R.E. if and only if L is R.E. Let L be any recursively enumerable, but undecidable, language over the unary alphabet, and let $x = x_L$. Thus the answers to (a), (b), and (c) are no. Since LESS_x is R.E. but not decidable, MORE_x is not R.E. The answer to (d) is thus yes.

6. Let L be the following language.

   (a) If $\mathcal{P} = \mathcal{NP}$, then $L = \{1\}$.

   (b) If $\mathcal{P} \neq \mathcal{NP}$, then $L = \{0\}$.

   Is L decidable? Explain your answer.

   L has only one string, hence is finite, hence is decidable.

7. Find a general grammar which generates $\left\{a^{2^n}\right\}$.

   Let $\Sigma = \{a\}$ and $V = \{S, A, B, C\}$, with productions
   $S \to AaB$
   $A \to AC$
   $A \to \lambda$
   $Ca \to aaC$
   $CB \to B$
   $B \to \lambda$

   For example:
   $S \Rightarrow AaB \Rightarrow aB \Rightarrow a$
   $S \Rightarrow AaB \Rightarrow ACaB \Rightarrow AaaCB \Rightarrow AaaB \Rightarrow aaB \Rightarrow aa$
   $S \Rightarrow AaB \Rightarrow ACaB \Rightarrow AaaCB \Rightarrow AaaB \Rightarrow ACaaB \Rightarrow AaaCaB \Rightarrow AaaaaCB \Rightarrow AaaaaB \Rightarrow aaaaB \Rightarrow aaaa$