University of Nevada, Las Vegas Computer Science 456/656 Spring 2024

Practice Problems for the Examination on April 10, 2024

1. True or False. T = true, F = false, and O = open, meaning that the answer is not known science at this time.

   (i) **F** Every subset of a regular language is regular.

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

   (iii) **T** If L is any $\mathcal{P}$–TIME language, there is a reduction of the Boolean circuit problem (CVP) to L which can be computed in polylogarithmic time using polynomially many processors.

   (iv) **T** The intersection of any two RE languages is RE.

   (v) **O** $\mathcal{P}$–TIME $= \mathcal{NC}$.

   (vi) **F** The context-free grammar equivalence problem is decidable.

   (vii) **T** The set of all positions of generalized checkers ($N \times N$ board for any N) from which black can win is decidable.

   (viii) **F** Every function that can be mathematically defined is bounded by some recursive function.

   (ix) **T** There are uncountably many languages over the binary alphabet.

   (x) **F** There are uncountably many RE languages over the binary alphabet.

   (xi) **O** If a language is both $\mathcal{NP}$ and co–$\mathcal{NP}$, it must be $\mathcal{P}$–TIME.

   (xii) **O** There is a $\mathcal{P}$–TIME algorithm which determines whether a given set of n positive integers has a subset whose total is n.

   (xiii) **T** If L_1 is $\mathcal{NP}$-complete and L_2 is $\mathcal{NP}$ and there is a $\mathcal{P}$–TIME reduction of L_1 to L_2, then L_2 must be $\mathcal{NP}$–COMPLETE.

   (xiv) **T** Nick's Class is closed under intersection. **T**

   (xv) **T** Every subset of any enumerable set is enumerable.

   (xvi) **F** Every subset of any recursively enumerable language is recursively enumerable.

   (xvii) **T** The computer language C++ has Turing power.

   (xviii) **T** Binary numeral multiplication is $\mathcal{NC}$.

   (xix) **T** There is an $\mathcal{P}$–SPACE algorithm which decides SAT.

   (xx) **O** Every dynamic program problem can be worked by polynomially many processors in polylogarithmic time.

   (xxi) **T** If an abstract Pascal machine can perform a computation in polynomial time, there must be some Turing machine that can perform the same computation in polynomial time.

(xxii) **F** Let L be any undecidable $\mathcal{RE}$ language, and let M_L be a machine which accepts L. For any string $w \in M$, let $F_L(w)$ be the number of steps M_L executes, if its input is w. If $w \notin M$, let $F_L(w) = 0$. Now define $T_L(n) = \{\max F(w) : w \in \Sigma^* \text{ and } |w| = n\}$. Then T_L is recursive.

(xxiii) **O** There is a polynomial time reduction of the subset sum problem to 2SAT.

(xxiv) **T** Every $\mathcal{P}$–TIME dynamic programming problem has an $\mathcal{NC}$ reduction to CVP.

(xxv) **T** If a language L is accepted by a non-deterministic machine, then L must be accepted by some deterministice machine.

(xxvi) **T** The language $\{a^n b^n c^n : n \geq 0\}$ is $\mathcal{NC}$.

(xxvii) **F** The set of all languages over the binary alphabet is countable.

(xxviii) **F** No set is larger than $\mathbb{R}$, the set of real numbers.

(xxix) **O** The furniture mover's problem is known to be $\mathcal{NP}$-complete. (Given a room with a door and given a collection of furniture that must be put into the room, can all the furniture be moved into the room through the door?)

(xxx) **F** The context-free grammar equivalence problem is co-$\mathcal{RE}$.

(xxxi) **T** Let $L = \{(G_1, G_2)\} : G_1$ and G_2 are not equivalent. Then L is recursively enumerable.

(xxxii) **T** The factoring problem for unary numerals is $\mathcal{P}$–TIME

(xxxiii) **T** The set of all binary numerals for prime numbers is in $\mathcal{P}$–TIME.

(xxxiv) **F** If L is a recursively enumerable language, there must be a machine which enumerates L in canonical order.

(xxxv) **F** The set of all positive real numbers is countable.

(xxxvi) **T** Let L be a recursive language over an alphabet Σ, and M a machine that decides L. For any n, let $F(n)$ be the maximum number of steps M needs to decide whether a given string in Σ^* of length n is in L. Then F must be recursive.

(xxxvii) **F** Let L be a recursively enumerable language over an alphabet Σ, and M a machine that accepts L. For any n, let $G(n)$ be the maximum number of steps M needs to accept any string in L of length n. Then G must be recursive.

(xxxviii) **T** For any alphabet Σ, the set of all recursively enumerable languages over Σ is countable.

(xxxix) **T** If L is a context-free language over the unary alphabet, then L must be regular.

(xl) **F** The union of any two undecidable languages is undecidable.

(xli) **T** co-$\mathcal{P}$–TIME= $\mathcal{P}$–TIME.

(xlii) **T** Every finite language is decidable.

(xliii) **T** Every context-free language is in Nick's class.

(xliv) **F** 2SAT is known to be $\mathcal{NP}$-complete.

(xlv) **T** The complement of any $\mathcal{P}$-TIME language is $\mathcal{P}$-TIME.

(xlvi) **T** The complement of any $\mathcal{P}$–SPACE language is $\mathcal{P}$–SPACE.

The jigsaw puzzle problem is, given a set of various polygons, and given a rectangular table, is it possibe to assemble those polygons to exactly cover the table?

(xlvii) **T** The jigsaw puzzle problem is known to be $\mathcal{NP}$ complete.

(xlviii) **F** The jigsaw puzzle problem is known to be $\mathcal{P}$–SPACE complete.

(xlix) **T** The furniture mover's problem is known to be $\mathcal{P}$–SPACE complete.

(l) **T** The complement of any recursive language is recursive.

(li) **T** The complement of any undecidable language is undecidable.

(lii) **F** Every undecidable language is either $\mathcal{RE}$ or co-$\mathcal{RE}$.

(liii) **T** For any infinite countable sets A and B, there is a 1-1 correspondence between A and B.

(liv) **T** A language L is recursively enumerable if and only if there is a machine which accepts L.

(lv) **T** Every $\mathcal{NP}$ language is reducible to the independent set problem in polynomial time.

(lvi) **T** If a Boolean expression is satisfiable, there is a polynomial time proof that it is satisfiable.

(lvii) **F** If a language L is recursively enumerable, there is a proof that L is recursively enumerable.

(lviii) **F** If a language L is co-$\mathcal{RE}$, there is a proof that L is co-$\mathcal{RE}$

(lix) **T** The Post correspondence problem is undecidable.

2. Fill in the blanks.

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

   (b) The class of RE languages is generated by the class of unrestricted (or general, or phase-structgure) grammars.

   (c) A language is **decidable** (or recursive) if and only if it is both RE and co-RE.

   (d) The class of push-down-automata accepts the class of **context-free** languages.

   (e) The class of Turing machines accepts the class of **recursively enumerable** languages.

   (f) A language L is **decidable** if and only if there is a machine which enumerates L in canonical order.

3. Here is a list of problems or languages. For each problem, enter **T** if it is known to be $\mathcal{NP}$–complete, **F** if it is not known to be $\mathcal{NP}$–complete.

   (a) **T** SAT

   (b) **F** 2-SAT

   (c) **T** 3-SAT

(d) **T** 4-SAT

(e) **F** Rush Hour

(f) **F** The Boolean circuit problem.

(g) **F** Integer factoring, using binary numerals.

(h) **T** Tiling, i.e., covering a big polygon exactly with the members of a set of smaller polygons.

(i) **F** Given a room with a door and some pieces of furniture, move them all into the room through the door.

(j) **T** Given a set of trucks, each with a given capacity, and given a set of items, can all the items fit into the trucks?

4. Give a definition of "unrestricted grammar." An unrestricted grammar consists of
   1. An alphabet of terminals, Σ.
   2. An alphabet of variables, Γ.
   3. A start symbol $S \in \Gamma$.
   4. A finite set of productions each of which is of the form lhs $\rightarrow$ rhs, where lhs, rhs $\in (\Sigma + \Gamma)^*$, and where lhs is not the empty string.

5. Give a definition of the class $\mathcal{P}$–SPACE.

   The class of all languages L such that L is decided by a program whose available memory is a polynomial function of the length of the input string.

6. Give a context-sensitive grammar for $L = \{a^n b^n c^n \ : \ n \geq 1\}$.

   There are many correct answers. Here is one:
   1. $S \rightarrow abc|aAbc$
   2. $Ab \rightarrow bA$
   3. $Ac \rightarrow Bcc$
   4. $bB \rightarrow Bb$
   5. $aB \rightarrow aab|aaAb$

7. Consider the following CF grammar G with start symbol E, and an LALR parser for G. The grammar generates algebraic expressions, where the only operations are subtraction and multiplication. The precedence of operators is as used in algebra and in programming languages.

   1. $E \rightarrow E -_2 E_3$
   2. $E \rightarrow E *_4 E_5$
   3. $E \rightarrow \mathbf{id}_6$

	id	$-$	$*$	\$	E
0	$s6$				1
1		$s2$	$s4$	HALT	
2	$s6$				3
3		$r1$	$s4$	$r1$	
4	$s6$				5
5		$r2$	$r2$	$r2$	
6		$r3$	$r3$	$r3$	

   (a) The entries in row 2 are left blank. Fill them in.

(b) The entries in row 3 are left blank. Fill them in.

(c) The entries in row 6 are left blank. Fill them in.

8. Explain how it is possible to find the maximum of an array of n integers in $O(\log n)$ time using $O(n/logn)$ processors. You don't need to draw a diagram, but it might help.

   We assume that $n = 2^k$ for some k. Otherwise, pad the array with numbers smaller than the first number. For k steps, pair the items of the current array and pick the maximum of those two, resulting in a list half as long. Keep going until there is only one in the list: that will be the maximum of the original list. The number of processors needed in $O(n)$, while the time complexity is $O(k) = O(\log n)$, hence the algorithm is $\mathcal{NC}$.

9. Prove that the halting problem is undecidable.

   Proof: By contradiction. Assume the language HALT $= \{\langle M\rangle w : M \text{ halts with input } w\}$ is decidable. Then the following program P exists:

   Read $\langle M\rangle$
   If ($\langle M\rangle\langle M\rangle \in$HALT) run forever
   Else halt

   We now run P with input $\langle P\rangle$. If $\langle P\rangle\langle P\rangle \in$HALT, then the program will run forever, contradiction, but if $\langle P\rangle\langle P\rangle \notin$HALT, the program will halt, contradiction. Thus, in any case, there is a contradiction, showing that our assumption was false, meaning that HALT is undecidable. ▌

10. Which of the following conditions is true if and only if a real number x is recursive? Yes, No, or Open for each.

    (a) **T** There is a program which, given n, finds the n^{th} digit after the decimal point of the decimal expansion of x.

    (b) **T** There is a machine which, given any positive integer q, computes an integer p such that $\frac{p}{q} \leq x < \frac{p+1}{q}$.

    (c) **F** There is a polynomial P with integral coefficients such that $P(x) = 0$. (For example: $5x^3 - 2x^2 + 9x - 4$ is a polynomial with integral coefficients.)

    The reason is that only algebraic numbers are roots to integral polynomials, and thus π is not. But π is a recursive real number.

    (d) **F** There is a mathematical definition of x.

    No explanation here; I might give it in class. But I still expect you to know that not all mathematically defined real numbers are recursive.

11. Prove that every recursively enumerable language is accepted by some deterministic machine.

    Proof: Let L be an $\mathcal{RE}$ language. Then some deterministic machine write an enumeration $w_1, w_2, \ldots$ of L. The following program accepts L.

Read w.
For all i from 1 to ∞
If($w = w_i$) Halt and accept

If $w \in L$, the program will eventually accept w, otherwise, it will never halt. ∎

12. (a) What is a one-way function?

    f is a one-way function if f can be computed in polynomial time, but there is no polynomial time algorithm to invert f. That is, give $f(x)$, there is no polynomial time algorithm to find some y such that $f(y) = f(x)$.

    (b) Does any one-way function exist?

    That is an open problem.

14. Determine whether the following Boolean expression is satisfiable. If so, give a satisfying assignment.

    $(a + b) * (a + c) * (!a + e) * (!b + d) * (!c + !d) * (!d + !e)$

    $a = 1$ $d = 0$ $e = 1$ $b = 0$, c arbitrary

15. Using the fact that 3SAT is $\mathcal{NP}$–complete, prove that the independent set problem is $\mathcal{NP}$-complete.

16. State the Church Turing thesis. Why is it important?

    Any computation which can be done by any machine can be done by some Turing machine.

    The importance of this is that if we can prove that no Turing machine can do a given computation, then no machine can do that compuation.

17. Prove that every language which can be enumerated in canonical order by some machine is recursive.

    Proof: Supose that some machine finds a canonical order enumeration of a language L, namely $w_1, w_2, \ldots$. Then the following program decides L.

    Read w
    for i from 1 to ∞
    if ($w_i = w$) halt and accept
    else if $w_i > w$ (in the canonical ordering) halt and reject

    The program always halts, hand hence decides L. ∎

18. Prove that every recursive language can be enumerated in canonical order by some machine.

    Proof: Let $L \subseteq \Sigma^*$ be a recursive language. Let $w_1, w_2, \ldots$ be a canonical order enumeration of Σ^*. Then the following program enumerates L in canonical order.

    for i from 1 to ∞
    if($w_i \in L$) write w_i

    The program enumerates L in canonical order. ∎

19. Prove that every recursively enumerable language is accepted by some machine.

Proof: Let L be an $\mathcal{RE}$ language. There is some machine that enumerates L. Let $w_1, w_2 \ldots$ be that enumeration. The following program accepts L.

Read a string w
For all i from 1 to ∞
  If($w = w_i$) Halt and accept ∎

20. Prove that every language accepted by a machine is recursively enumerable.

Proof: Let M be a machine which accepts $L \subseteq \Sigma^*$ for some alphabet Σ. Let $w_1, x_2 \ldots$ be the canonical order enumeration of Σ^*, which is easily computable. The following program enumerates L.

For t from 1 to ∞
  For i from 1 to t
    If (M accepts w_i within its first t steps)
      Write w_i ∎

21. Why is the question of whether $\mathcal{NC} = \mathcal{P}$–TIME so important nowadays?

Because parallel computing is becoming more common, and problems are getting larger. It would be good if the work for really large programs could be efficiently distributed amonn many processors. This is sometimes true, but if $\mathcal{P}$–TIME $= \mathcal{NC}$, it would always be true.

22. Prove that every regular languge is $\mathcal{NC}$.

The proof is in Handouts/regularNC.pdf. It is much too long to put on a test, but I still want you to know that all regular languages are $\mathcal{NC}$.

25. Which of these languages (problems) are **known** to be $\mathcal{NP}$-complete? If a language, or problem, is known to be $\mathcal{NP}$-complete, fill in the first circle. If it is either known not be be $\mathcal{NP}$-complete, or if whether it is $\mathcal{NP}$-complete is not known at this time, fill in the second circle.

⊗	○	Boolean satisfiability.
○	⊗	2SAT.
⊗	○	3SAT.
⊗	○	Subset sum problem.
○	⊗	Generalized checkers, i.e. on a board of arbitrary size.
⊗	○	Traveling salesman problem.
○	⊗	Rush Hour: https://www.youtube.com/watch?v=HI0rlp7tiZ0
⊗	○	Dominating set problem.
○	⊗	Strong connectivity of directed graphs.
○	⊗	Circuit value problem, CVP.
○	⊗	C++ program equivalence.
⊗	○	Partition.
○	⊗	Regular language membership problem.
⊗	○	Block sorting.

26. State the pumping lemma for regular languages.

    For any regular language L, there is a positive number p such that for any $w \in L$ of length at least p there exist strings x, y, z such the the following statements hold:
    1. $w = xyz$
    2. $|xy| \leq p$
    3. $|y| \geq 1$
    4. For any integer $i \geq 0$ $xy^iz \in L$.

27. State the pumping lemma for context-free languages.

    For any context-free language L, there is a positive number p such that for any $w \in L$ of length at least p there exist strings u, v, x, y, z such the the following statements hold:
    1. $w = uvxyz$
    2. $|vxy| \leq p$
    3. $|v| + |y| \geq 1$
    4. For any integer $i \geq 0$ $uv^ixy^iz \in L$.

28. Give a polynomial time reduction of 3SAT to the independent set problem. (Pictures help.)

    This reduction is given in the proof of Theorem 6 of Handouts/rducNP2.pdf.

29. Prove that any recursively enumerable language is accepted by some machine.

    Duplicate of previous problem.

30. Consider G, the following context-free gammar with start symbol E. Stack states are indicated.
    1. $E \to E_{1,11} +_2 E_3$
    2. $E \to E_{1,11} -_4 E_5$
    3. $E \to E_{1,3,5,11} *_6 E_7$
    4. $E \to -_8 E_9$

5. $E \to ({}_{10}E_{11})_{12}$
6. $E \to x_{13}$

(a) Below are the tables of an LALR parser for G. Fill in the missing colums.

	x	$+$	$-$	$*$	(	)	\$	E
0	s13		s8		s10			1
1		s2	s4	s6			**halt**	
2	s13		s8		s10			3
3		r1	r1	s6		r1	r1	
4	s13		s8		s10			5
5		r2	r2	s6		r2	r2	
6	s13		s8		s10			7
7		r3	r3	r3		r3	r3	
8	s13		s8		s10			9
9		r4	r4	r4		r4	r4	
10	s13		s8		s10			11
11		s2	s4	s6		s12		
12		r5	r5	r5		r5	r5	
13		r6	r6	r6		r6	r6	

(b) Give a complete computation of the parser if the input string is $x - x * -(-x + x)$.

31. Fill in the following table, showing which operations are closed for each class of languages. In each box, write **T** if it is known that that language class is closed under that operation, **F** it is known that that class is not closed under that operation, and **O** if neither of those is known.

language class	union	intersection	concatenation	Kleene closure	complementation
$\mathcal{NC}$	T	T	T	T	T
context-free	T	F	T	T	F
$\mathcal{NP}$	T	T	T	T	O
recursive	T	T	T	T	T
co-$\mathcal{RE}$	T	T	?	?	F
undecidable	F	F	?	?	T

32. Consider the following well-known complexity classes:

$\mathcal{NC} \subseteq \mathcal{P}$–TIME$\subseteq \mathcal{NP} \subseteq \mathcal{P}$–SPACE $\subseteq$**EXP**–TIME $\subseteq$**EXP**–SPACE

(a) Which of the above complexity classes is the smallest class which is known to contain SAT, the Boolean satisfiability problem?

$\mathcal{NP}$

(b) Which of the above complexity classes is the smallest class which is known to contain the connectivity problem for graphs?

$\mathcal{P}$–TIME

(c) Which of the above complexity classes is the smallest class which is known to contain the context-free language membership problem?

$\mathcal{NC}$

(d) Which of the above complexity classes is the smallest class which is known to contain every sliding block problem?

$\mathcal{P}$–SPACE

(e) Which of the above complexity classes is the smallest class which is known to contain integer matrix multiplication?

$\mathcal{NC}$

(f) We say that a computer program is straight-line if no portion of the code can be executed more than once. That implies that the code contains no loops or recursion, and no GOTO from one line of the code to an earlier line. Which of the above complexity classes is the smallest class which contains the problem of determining whether the output of a straight-line program is zero?

$\mathcal{P}$–TIME

33. Let L be the simple algebraic language with three operators, subtraction, multiplication, and negation, and with only one variable. Write an annotated context-free grammar for L, annotated with stack states as in the handout, and write the ACTION and GOTO tables for an LALR parser.
    1. $E \to E +_2 E_3$
    2. $E \to E -_4 E_5$
    3. $E \to E *_6 E_7$
    4. $E \to -_8 E_9$
    5. $E \to x_{10}$

	x	$+$	$-$	$*$	\$	E
0	$s10$		$s8$			1
1		$s2$	$s4$	$s6$	**halt**	
2	$s10$		$s8$			3
3		$r1$	$r1$	$s6$	$r1$	
4	$s10$		$s8$			5
5		$r2$	$r2$	$s6$	$r2$	
6	$s10$		$s8$			7
7		$r3$	$r3$	$r3$	$r3$	
8	$s10$		$s8$			9
9		$r4$	$r4$	$r4$	$r4$	
10		$r5$	$r5$	$r5$	$r5$	