University of Nevada, Las Vegas Computer Science 456/656 Fall 2023

Answers to Assignment 7: Due Sunday November 19, 2023

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

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

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

   (c) **F** If L_1 is a regular language and L_2 is a context-free language, then $L_1 \cap L_2$ is context-free.

   (d) **T** If there is a recursive reduction of L_1 to L_2, where L_1 is an undecidable language, then L_2 must be undecidable.

   (e) **O** The factoring problem is in $\mathcal{P}$.

   (f) **T** If L is a recursive language, there must be a machine which enumerates L in canonical order.

   (g) **T** If there is a machine which enumerates a language L in canonical order, then L must be recursive.

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

   (i) **T** The set of all recursive real numbers is countable.

   (j) **F** For any alphabet Σ, the set of all languages over Σ is countable.

   (k) **T** For any alphabet Σ, the set of all recursive languages over Σ is countable.

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

   (m) **F** Every subset of a recursive language is recursive.

2. Give a definition of a *recursive* real number. (There is more than one correct definition.)

   A recursive real number x is defined by any one of the following:

   - There is a machine that writes the decimal (or binary, or any base) expansion of x.
   - If $d(n)$ is the n^{th} digit of the decimal (or whatever) expansion of x, then d is a recursive function.
   - The question of whether x is less than any given fraction is decidable.

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.

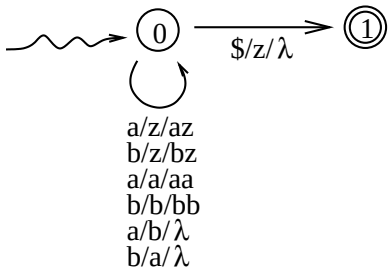
- ⊗ ○ Boolean satisfiability.
- ○ ⊗ 2SAT.
- ⊗ ○ 3SAT.
- ⊗ ○ 4SAT.
- ⊗ ○ Subset sum problem.
- ○ ⊗ Generalized checkers, *i.e.* on a board of arbitrary size.
- ⊗ ○ Independent set problem.
- ⊗ ○ Traveling salesman problem.
- ○ ⊗ Regular expression equivalence.
- ○ ⊗ C++ program equivalence.
- ○ ⊗ Rush Hour: `https://www.youtube.com/watch?v=HI0rlp7tiZ0`
- ○ ⊗ Circuit value problem, CVP.
- ○ ⊗ Regular grammar equivalence.
- ⊗ ○ Dominating set problem.
- ⊗ ○ Partition.

3. State the pumping lemma for regular languages.

For any regular language L,
there exists an integer p such that
for any $w \in L$ of length at least p,
there exist strings x, y, z, such that the following statements hold:

1. $w = xyz$
2. $|xy| \leq p$
3. $|y| \geq 1$
4. for any integer $i \geq 0$, $xy^iz \in L$

5. Let L be the language over $\{a, b\}$ consisting of all strings which have the same number of a's as b's, such as aabb, abba, aaabbb, bbbaaa, ... Design a DPDA which accepts L.



6. Give a polynomial time reduction of the subset sum problem to partition.

   Let $\alpha = (\gamma, K)$ be an instance of the subset sum problem, where $\gamma = (x_1, x_2, \ldots x_n)$. That instance has a solution if and only if γ has a subsequence whose sum is K.

   We construct an instance β of the partition problem which has a solution if and only if α has a solution.

   Let $S = \sum_{i=1}^{n} x_i$. If $S < K$ then α has no solution, and we let $\beta = (1)$, which has no solution.

   Suppose $K \leq S$. Let $\beta = (x_1, x_2, \ldots x_n, K+1, S-K+1)$. The sum of that sequence is $2S+2$. Thus a solution to β is a subsequence δ of β whose sum is $S+1$. If α has a solution then there is a subsequence δ of γ whose sum is K. Then the concatenation of δ with $S-K+1$ is a subsequence of β whose sum is $S+1$, hence a solution of β

   Conversely, suppse ϵ is a subsequence of β whose sum is $S+1$. β cannot contain both $K+1$ and $S-K+1$, since their total is $S+2$. Similarly, β must contain either $K+1$ or $S-K+1$, since the sum of γ is less than $S+1$. Thus, ϵ must contain either $K+1$ or $S-K+1$.

   Suppose ϵ contains $S-K+1$. Deletion of $S-K+1$ from ϵ gives a subsequence of γ whose sum is K. Alternatively, ϵ contains $K+1$. Then the complement of ϵ is a subseqence of β sum is $S+1$.

7. Give a polynomial time reduction of 3SAT to the independent set problem.

   Let $E = C_1 * C_2 * \cdots * C_k$ be a Boolean expression in 3-CNF form. Each C_i is a clause which is the disjunction of three terms, each of which is either a variable or the negation of a variable. The reduction maps E onto a graph G, which $3k$ vertices, each labeled by one of the $3k$ terms of E. Each clause C_i corresponds to the 3-clique Q_i whose vertices are labeled by the three terms of C_i. The remaining edges of G are between cliques. These edges connect any two vertices of G which have contradictory labels. We show that E is satisfiable if and only if G has an independent set of size k.

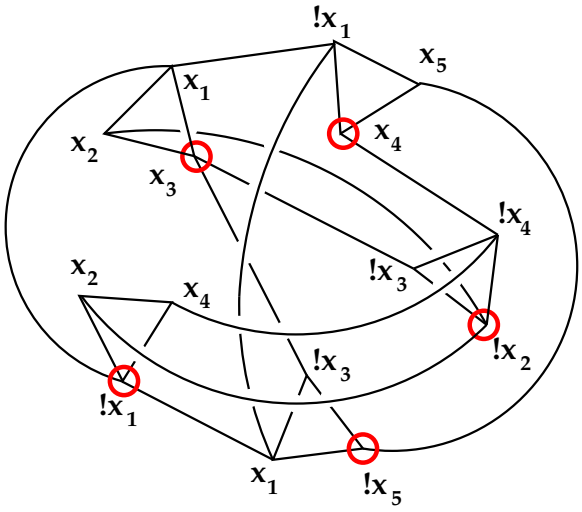
   Suppose E has a satisfying assignment. Pick one term of each clause which is true under the assignment, we call that the *satisfying* term of that clause. Let I be the set of vertices labeled by the certifying terms, k vertices altogether. Any two members of I cannot be connected by a clique edge, since they lie in different cliques. They also cannot be connected by an edge between cliques, since the labels of the ends of that edge cannot both be true under the assignment. Thus G has an independent set of size k.

   Conversely, suppose G has an independent set I of size k. There must No two of these vertices can lie in the same clique, hence there must be one in each clique. Let v_i be the sole member of $I \cap Q_i$. Now pick an assignment such that the label of each member of I is assigned true. This is possible because no two labels of members of I can contradict. If there is a variable whose value has not been assigned, assign that variable the value true. (Actually, that assignment is arbitrary.) This assignment satisfies E, since each clause has a certifying term.

   Example. Let $E = (x_1 + x_2 + x_3) * (!x_1 + x_4 + x_5) * (!x_2 + !x_3 + !x_4) * (x_1 + !x_3 + !x_5) * (!x_1 + x_2 + x_4)$

   The following assignment satisfies E.
   $x_1 = 0$
   $x_2 = 0$
   $x_3 = 1$
   $x_4 = 1$
   $x_5 = 0$

In the figure below, the vertices of the corresponding independent set of G are circled in red.



8. Prove that a language is recursively enumerable, $\mathcal{RE}$, if and only if it is accepted by some machine.

This is really two theorems.

1. If a language L is enumerated by some machine, then L is accepted by some machine.

Proof: Let $w_1, w_2, \ldots$ be an enumeration of L written by some machine. Then the following program accepts L.

```
read w
for i = 1 to ∞
  if (w = w_i)
    write "1" and halt
```

2. If a language L is accepted by some machine M, then some machine enumerates L.

Proof: Recall that any machine computation consists of a sequence of steps. Let Σ be the alphabet of L. Let $w_1, w_2, \ldots$ be the canonical order enumeration of Σ^*. The following program enumerates L.

```
for t = 1 to ∞
  for i = 1 to t
    if M accepts w_i within t steps
      write w_i
```

9. Consider G, the following context-free grammar 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. $E \to x_{13}$

   What follows is an ACTION table followed by a GOTO table for an LALR parser for G.

   (a) Which entry guarantees that negation has higher priority than multiplication?

   Row 9, column "$*$"

   (b) Oops! I somehow forgot to fill in the column headed by "$-$"! Fill it in. Hint: unlike the others, this column contains no empty cells. Remember that the minus sign is used for both subtraction and negation.

   Here is the solution.

	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	

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

$\$_0$	$x + x * -(-x * x)\$$		
$\$_0 x_1 3$	$+x * -(-x * x)\$$		$s13$
$\$_0 E_1$	$+x * -(-x * x)\$$	6	$r6$
$\$_0 E_1 +$	$x * -(-x * x)\$$	6	$s2$
$\$_0 E_1 +_2 x_{13}$	$* - (-x * x)\$$	6	$s13$
$\$_0 E_1 +_2 E_3$	$* - (-x * x)\$$	66	$r6$
$\$_0 E_1 +_2 E_3 *_4$	$-(-x * x)\$$	66	$s4$
$\$_0 E_1 +_2 E_3 *_4 -_8$	$(-x * x)\$$	66	$s8$
$\$_0 E_1 +_2 E_3 *_4 -_8 (_{10}$	$-x * x)\$$	66	$s8$
$\$_0 E_1 +_2 E_3 *_4 -_8 (_{10} -_8$	$x * x)\$$	66	$s8$
$\$_0 E_1 +_2 E_3 *_4 -_8 (_{10} -_8 x_{13}$	$*x)\$$	66	$s13$
$\$_0 E_1 +_2 E_3 *_4 -_8 (_{10} -_8 E_9$	$*x)\$$	666	$r6$
$\$_0 E_1 +_2 E_3 *_4 -_8 (_{10} E_{11}$	$*x)\$$	6664	$r4$
$\$_0 E_1 +_2 E_3 *_4 -_8 (_{10} E_{11} *_4$	$x)\$$	6664	$s4$
$\$_0 E_1 +_2 E_3 *_4 -_8 (_{10} E_{11} *_4 x_{13}$	$)\$$	6664	$s13$
$\$_0 E_1 +_2 E_3 *_4 -_8 (_{10} E_{11} *_4 E_5$	$)\$$	66646	$r6$
$\$_0 E_1 +_2 E_3 *_4 -_8 (_{10} E_{11}$	$)\$$	666462	$r2$
$\$_0 E_1 +_2 E_3 *_4 -_8 (_{10} E_{11})_{12}$	$\$$	666462	$s12$
$\$_0 E_1 +_2 E_3 *_4 -_8 E_9$	$\$$	6664625	$r5$
$\$_0 E_1 +_2 E_3 *_4 E_5$	$\$$	66646254	$r4$
$\$_0 E_1 +_2 E_3$	$\$$	666462542	$r2$
$\$_0 E_1$	$\$$	6664625421	$r1$

HALT 6664625421

10. Consider the following problem. Given binary numerals $\langle u \rangle$ and $\langle v \rangle$ of length n, decide whether $u < v$. Give an $\mathcal{NC}$ algorithm for solving this problem.

    This problem is fairly easy, but has been eliminated from Assignment 7, since I have not covered enough on parallel algorithms.

11. Prove that a language is enumerable in canonical order by some machine if and only if it is decidable.

    This is really two theorems.

    1. If a language L is enumerated in canonical order by some machine, then L is decidable.

    Proof: Let $w_1, w_2, \ldots$ be an enumeration of L in canonical order. written by some machine. Then the following program decides L.

```
read w
for i = 1 to ∞
  if (w = w_i)
    write "1" and halt
  else if (w > w_i)  using the canonical order to define "<"
    write "0" and halt
```

2. If a language L is decidable, then some machine enumerates L in canonical order.

Proof: Let Σ be the alphabet of L. Let $w_1, w_2, \ldots$ be the canonical order enumeration of Σ^*. The following program enumerates L in canonical order.

for i = 1 to ∞
  if ($w_i \in L$) (recall that L is decidable)
    write w_i

12. Consider the Chomsky Normal Form grammar G given below.

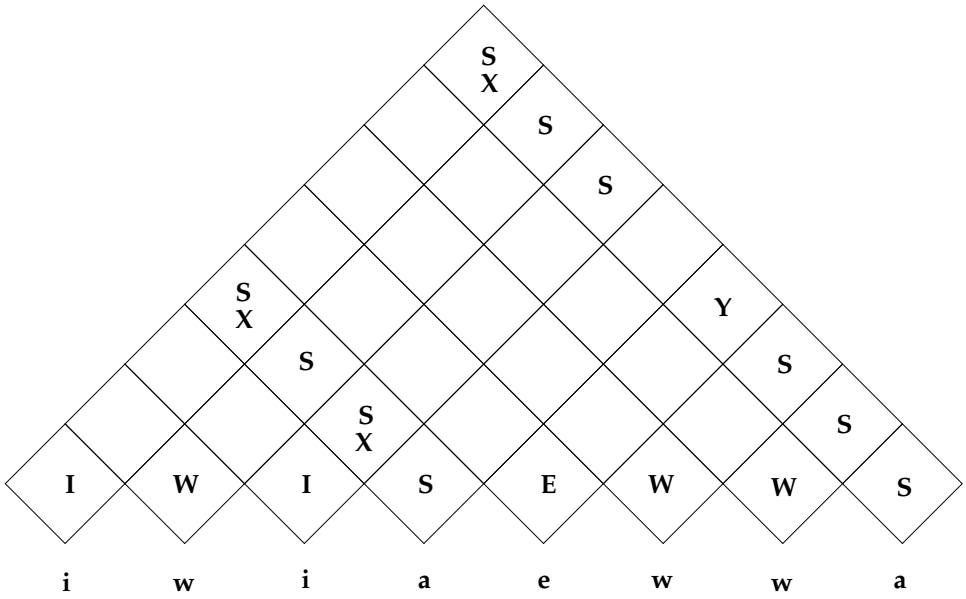
$S \to IS$
$S \to WS$
$S \to XY$
$X \to IS$
$Y \to ES$
$S \to a$
$I \to i$
$W \to w$
$E \to e$

(a) Show that G is ambiguous by giving two different **leftmost** derivations for the string $iiaea$.

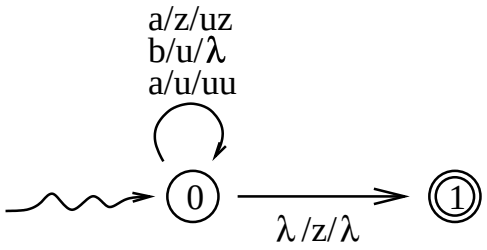
$S \Rightarrow IS \Rightarrow iS \Rightarrow iWS \Rightarrow iwS \Rightarrow iwXY \Rightarrow iwISY \Rightarrow iwiSY \Rightarrow iwiaY \Rightarrow iwiaES \Rightarrow iwiaeS$
$\Rightarrow iwiaeWS \Rightarrow iwiaewS \Rightarrow iwiaewWS \Rightarrow iwiaewwS \Rightarrow iwiaewwa$

$S \Rightarrow XY \Rightarrow iSY \Rightarrow iWSY \Rightarrow iwISY \Rightarrow iwiSY \Rightarrow iwiaY \Rightarrow iwiaES \Rightarrow iwiaES \Rightarrow iwiaeS$
$\Rightarrow iwiaeWS \Rightarrow iwiaewS \Rightarrow iwiaewWS \Rightarrow iwiaewwS \Rightarrow iwiaewwa$

(b) Use the CYK algorithm to prove that $iwiaewwa \in L(G)$.

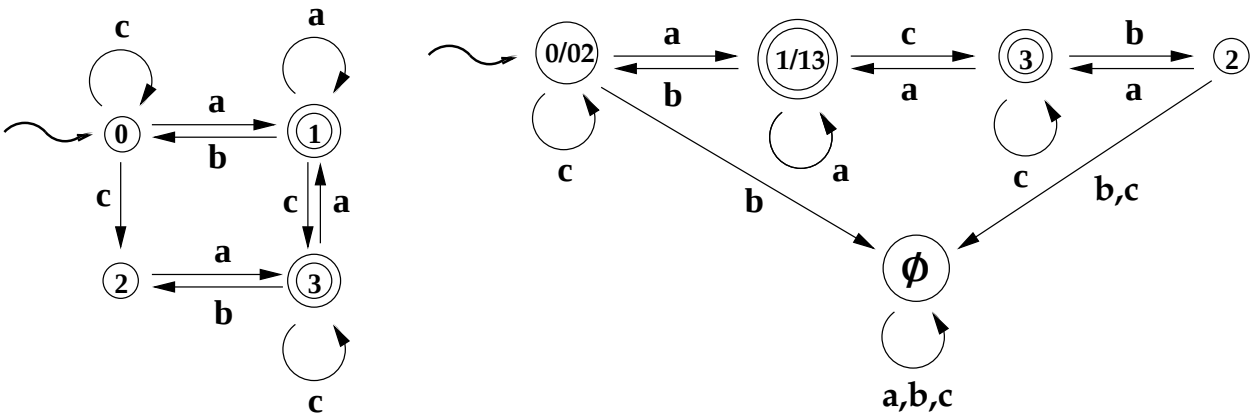


13. Let L be the language accepted by the DPDA shown below. What is L? You can either describe L in a **few** words, or give a context-free grammar for L.



The Dyck language, where "(" is repaced by a and ")" by b. Here is a CF grammar for L:
$S \to aSbS$
$S \to \lambda$

14. Find a minimal DFA equivalent to the NFA shown below.



15. 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
regular	T	T	T	T	T
context-free	T	F	T	T	F
$\mathcal{P}$–TIME	T	T	T	T	T
$\mathcal{NP}$	T	T	T	T	O
co-$\mathcal{NP}$	T	T	T	T	O
recursive	T	T	T	T	T
$\mathcal{RE}$	T	T	T	T	F
co-$\mathcal{RE}$	T	T	T	T	F
undecidable	F	F	F	F	T

Here is a proof that the class of undecidable languages is not closed under Kleene closure. Let U be an undecidable set of integers. That means, there is no recursive function $F(n)$ which is true if and only if $n \in U$. Now let $L \subset \{1\}^*$ consist of all strings 1^n such that $n \in U$, as well as the string $\{1\}$. Then L is undecidable, but $L^* = \{1\}^*$, the set of all unary strings, which is clearly deciable.

16. Prove that the halting problem is undecidable.

By contradiction. Suppose HALT is decidable. Let D be a machine which takes any machine description $\langle M \rangle$ as input, and runs forever if M halts with input $\langle M \rangle$, and otherwise halts. Does D with input $\langle D \rangle$ halt? If D halts with input $\langle D \rangle$, the D enters an infinte loop, contradition. But if D does not halt with input D, then it halts, contradiction.

17. Consider the following well-known complexity classes.

$$\mathcal{NC} \subseteq \mathcal{P} - \text{TIME} \subseteq \mathcal{NP} - \text{TIME} \subseteq \mathcal{P} - \text{SPACE} \subseteq \mathbf{EXP} - \text{TIME} \subseteq \mathbf{EXP} - \text{SPACE}$$

The *mover's problem* is, given a room with a door and pieces of furniture of various shapes and sizes, can the furniture be moved into the room through the door?

The *crane operator's problem* is, given a room and pieces of furniture of various shapes and sizes, can the furniture be placed into the room after the roof is removed?

For both furniture problems, we assume that no piece of furniture can ever be fully or partially on top of another.

(a) Which of the above complexity classes is the smallest class which is known to contain the crane operator's problem?

$\mathcal{NP}$–TIME

(b) Which of the above complexity classes is the smallest class which is known to contain the mover's problem?

$\mathcal{P}$–SPACE

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

$\mathcal{NC}$

(d) Which of the above complexity classes is the smallest class which is known to contain the circuit valuation problem, which is the problem of determining the output of a Boolean circuit with given inputs?

$\mathcal{P}$–TIME

(e) *Generalized checkers* is the game of checkers played on an $n \times n$ board. (The standard game uses an 8×8 board.) Which of the above complexity classes is the smallest class which is known to contain the problem of determining whether the first player to move, from a given configuration, can win?

EXP–TIME

18. f is a *one-way function* if $f(x)$ can be computed in polynomial time for any string x, but there is no polynomial time randomized algorithm which can invert f; that is, given $f(x)$, find, with high probability, a string x' such that $f(x) = f(x')$. Such a function would be useful in cryptography.

    The formal definition is given at `https://en.wikipedia.org/wiki/One-way_function` There are some functions that are generally believed to be one-way, but no one knows for sure. Prove that if $\mathcal{P} = \mathcal{NP}$, no one-way function exists.

    There is no possibility that I will ask for this proof on the exam. I expect you to know that if any one-way function exists, $\mathcal{P} \neq \mathcal{NP}$, and the fact that $\mathcal{P} = \mathcal{NP}$ would destroy any public key private key cryptography system, such as RSA. I will give a proof in class after the exam. Intuitively, if f is a 1-way function and $y = f(x)$ is given for some unknown string x, then, if someone tells you x, you can verify that $y = f(x)$ in polynomial time, therefore, if $\mathcal{P} = \mathcal{NP}$, you should be able to find x in polynomial time. The correct proof is more complex, but that is the basic idea.

19. The binary factorization problem is, given a binary numeral for an integer n, and another "benchmark" numeral for an integer a, determine whether n has a factor greater than 1 and less than a. Prove that the binary integer factorization problem is in $\mathcal{NP}$. (It is also in co-$\mathcal{NP}$, but that is harder to prove.)

    If such a factor p exists, it can be checked in polynomial time, just by dividing n by p. In this case, p is a certificate. The problem is in $\mathcal{NP}$ by one of the definitions of $\mathcal{NP}$.

    For the exam, you should also know that if the factoring problem is $\mathcal{P}$–TIME, RSA encryption can be broken in polynomial time, but that still does not imply that $\mathcal{P} = \mathcal{NP}$.