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

Answers to Practice Problems for Final Examination May 8, 2024

These problems are taken from Assignment 7 and from final examinations for four earlier semesters.

Try as I might, I could not remove all duplicate questions, so just ignore the duplicates.

Problems Taken from Assignment 7

1. Prove that every decidable language is enumerated in canonical order by some machine.

   Let $L \subset \Sigma^*$ be a decidable language over an alphabet Σ. Let $w_1, w_2, \ldots$ be the canonical enumeration of Σ^*. The following program enumerates L in canonical order.

```
For all $i$ from 1 to $\infty$
    If $w_i \in L$ write $w_i$
```

2. Prove that every language that is enumerated in canonical order by some machine is decided by some other machine.

   Suppose some machine writes Let $w_1, w_2, \ldots$, the canonical order of a language L over an alphabet Σ. If L is finite, let w_n be the last item in that enumeration, otherwise, the enumeration is infinite. The following program decides whether a given $w \in \Sigma^*$ is a member of L.

```
Read $w$
For $i$ from 1 to $n$ if $L$ is finite, otherwise from 1 to $\infty$.
    If $w = w_i$
        Halt and accept $w$
    Else if $w \le w_i$ (canonical order)
        Halt and reject $w$


Halt and reject $w$
```

3. Prove that eny language accepted by any machine can be enumerated by some other machine.

   Let L be a language over an alphabet Σ and M a machine that accepts L. If $w \in L$, M will accept w in finitely many steps. Let $w_1, w_2, \ldots$ be the enumeration of Σ^* in canonical order. The following program enumerates L.

```
For $t$ from 1 to $\infty$
    For $i$ from 1 to $t$
        If $M$ accepts $w_i$ within $t$ steps
            If $w_i$ has not already been written
                write $w_i$
```

   This modified program will write each member of L exactly once.

4. Prove that any language which is enumerated by some machine is accepted by some other machine.

   Let M be a machine which writes an enumeration of L, $w_1, w_2, \ldots$ The following program will accept L.

```
Read $w$
For $i = 1$ to $\infty$
    If $w = w_i$
        Halt and accept $w$
```

5. I have repeatedly stated in class that no language that has parentheses can be regular. For that to be true, there must be parenthetical strings of arbitrary nesting depth. (If you don't know what nesting depth is, look it up.)

   Some programming languages have limitations on nesting depth. For example, I have read that ABAP has maximum nesting depth of 256. (Who would ever want to go that far!)

   The Dyck language is generated by the following context-free grammar. (As usual, to make grading easier, I use a and b for left and right parentheses.)

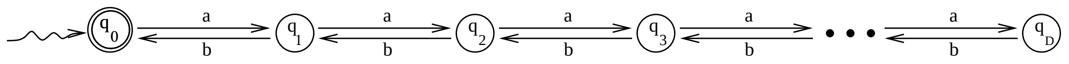
   1. $S \to aSbS$
   2. $S \to \lambda$

   (a) Use the pumping lemma to prove that the Dyck language is not regular.

   Let L be the Dyck language. Suppose that L is regular. Let p be a positive integer which is a pumping length of L. Let $w = a^p b^p$, which is member of L. Then, by the pumping lemma, we can choose strings x, y, z such that
   1. $w = xyz$
   2. $|xy| \leq p$
   3. $|y| \geq 1$
   4. For any integer $i \geq 0$, $xy^i z \in L$.
   By 2., xy is a prefix of $xyz = w = a^p b^p$ of length at most p, hence consists entirely of a's. Thus $y = a^k$, and $k > 0$ by 3. By 4. $xyyz = a^{p+k} b^p = a^{p+k} b^p \in L$, contradiction, since $p + k \neq p$.

   (b) Let D be any positive integer. Let L be the language consisting of all members of the Dyck language whose nesting depth does not exceed D. Prove that L is regular.

   The following DFA accepts L. It has one live state for each possible nesting depth of the current input, $D + 1$ live states altogether, together with a dead state not shown.



6. We know that context-free languages are exactly those which are accepted by push-down automata. We now define a new class of machines, which we call "limited push-down automata." An LPDA is exactly the same as a PDA, but with the restriction that the stack is never allowed to be larger than some given constant. What is the class of languages accepted by limited push-down automata? Think!

   Regular languages. An LPDA can be converted into an NFA, since there are a finite number of possible stack states.

7. Prove that every context-sensitive language is decidable. The way to do this is to start with an arbitrary context-sensitive grammar, using the definition I gave in class (that's not the only definition) namely that the right side of any production must be at least as long as the left side, and then design a program which decides whether any given string is generated by that grammar. (If the string has length n, the running time of your program could be very long, maybe an exponentially bounded function of n?)

   Let G be a context-sensitive grammar, and $L = L(G)$. Let Σ be the terminal alphabet of G, that is, $L \subseteq \Sigma^*$. Let Γ be the variable alphabet of G, and $S \in \Gamma$ the start symbol of G. By definition, a *sentential form* of G is any string which is derived from S in any number of steps, using productions of G. Let $\mathfrak{S} \subseteq (\Sigma + \Gamma)^*$ be set of all sentential forms of G. Note that $L = \Sigma^* \cap \mathfrak{S}$. We let $\mathfrak{S}_m$ be the set of all sentential forms of G of length m.

   Let $w \in \Sigma^*$, and let $n = |w|$. The following algorithm decides whether $w \in L$.

   Note $\{S\} \in \mathfrak{S}_1 \subseteq \Sigma + \Gamma$. **corrected.** For each m from 2 to n, compute $\mathfrak{S}_m$ from $\mathfrak{S}_i$ for all $i < m$, by repeatedly using the following construction. If G has the production $\alpha \to \beta$, and if $\phi\,\alpha\,\psi \in \mathfrak{S}$ for some $\phi, \psi \in (\Sigma + \Gamma)^*$, then $\phi\beta\psi \in \mathfrak{S}$. By the non-decreasing rule of CS productions, $|\phi\,\alpha\,\psi| \leq |\phi\,\beta\,\psi|$. Finally, $w \in L$ if and only if $w \in \mathfrak{S}$.

   The above algorithm is EXP–SPACE, since G could have exponentially many sentential forms of length n.

Problems Taken from the Final Examination of Fall 2021

1. True/False/Open

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

   (ii) **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.

   (iii) **T** The intersection of any regular language with any context-free language is context-free.

   (iv) **F** The intersection of any two context-free languages is context-free.

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

   (vi) **T** The language $\{a^n b^n c^n : n \text{ is prime}\}$ is decidable.

   (vii) **T** If a language L is EXP–SPACE complete, L must be decidable.

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

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

   (x) **F** EXP-TIME = $\mathcal{P}$-TIME.

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

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

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

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

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

(xvi) **T** The problem of whether two given context-free grammars are equivalent is co-$\mathcal{RE}$.

(xvii) **T** The language of all fractions (using base 10 numeration) whose values are less than π is decidable. (Recall that a fraction is a string consisting of two numerals separated by a slash, "/".)

(xviii) **T** If L is $\mathcal{RE}$ and co-$\mathcal{RE}$, then L is decidable.

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

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

(xxi) **T** If a Boolean expression is satisfiable, there is a $\mathcal{P}$–TIME proof that it's satisfiable.

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

(xxiii) **F** Every subset of any recursively enumerable set is recursively enumerable.

(xxiv) **T** The binary numeral factorization problem is co-$\mathcal{NP}$.

2. Every language, or problem, falls into exactly one of these categories. For each of the languages, write a letter indicating the correct category.

   A Known to be $\mathcal{NC}$.
   B Known to be $\mathcal{P}$–TIME, but not known to be $\mathcal{NC}$.
   C Known to be $\mathcal{NP}$, but not known to be $\mathcal{P}$–TIME and not known to be $\mathcal{NP}$–complete.
   D Known to be $\mathcal{NP}$–complete.
   E Known to be $\mathcal{P}$–SPACE but not known to be $\mathcal{NP}$
   F Known to be EXP–TIME but not nown to be $\mathcal{P}$–SPACE.
   G Known to be EXP–SPACE but not nown to be EXP–TIME.
   H Known to be decidable, but not nown to be EXP–SPACE.
   K $\mathcal{RE}$ but not decidable.
   L co-$\mathcal{RE}$ but not decidable.
   M Neither $\mathcal{RE}$ nor co-$\mathcal{RE}$.

   (a) **K** All C++ programs which halt with no input.

   (b) **A** All base 10 numerals for perfect squares.

   (c) **L** All context-free grammars that generate the Dyck language.

   (d) **E Corrected!** All configurations of RUSH HOUR from which it's possible to win.

   (e) **D** All satisfiable Boolean expressions.

(f) **B** All binary numerals for composite integers. (Composite means not prime.)

3. Give a definition of each term.

   (a) Accept. (That is, what does it mean for a machine M to accept a language L over an alphabet Σ.)

   If M is given an input string $w \in \Sigma^*$, it will halt and accept w if and only if $w \in L$.

   (b) Decide. (That is, what does it mean for a machine M to decide a language L over an alphabet Σ.)

   If M is given an input string $w \in \Sigma^*$, it will halt. If $w \in L$, M will accept w, if not, it will reject w.

   (c) Canonical order of a language L.

   In the canonical order, if $u, v \in L$, we say $u < v$ if either $|u| < |v|$, or $|u| = |v|$ and u comes before v in alphabetic ordering.

4. Find a minimal DFA equivalent to the NFA shown in Figure 1.

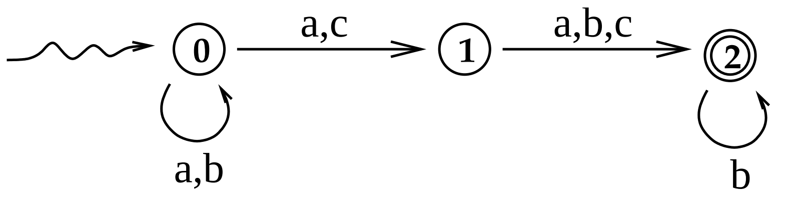


Figure 1: NFA for Problems 4 and 5

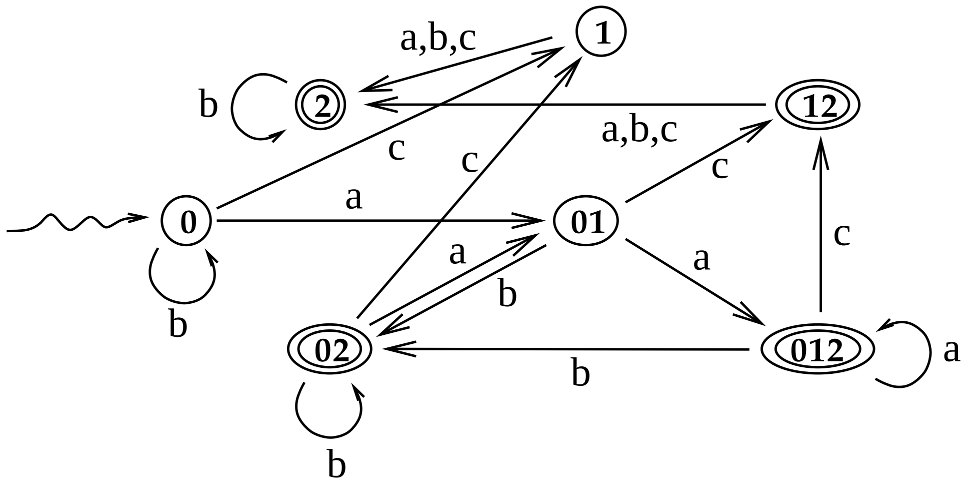


Figure 2: DFA Equivalent to NFA in Figure 1

5. Give a regular grammar with **no more than three variables** for the language accepted by the machine in Figure 1.

   $S \to aS|bS|aA|cA$

$A \to aB|bB|cB$
$B \to bB|\lambda$ **corrected**

6. Give a regular expression for the language accepted by the machine in Figure 3

   $a^*b(ca^*b + a + ac^*b)^*$

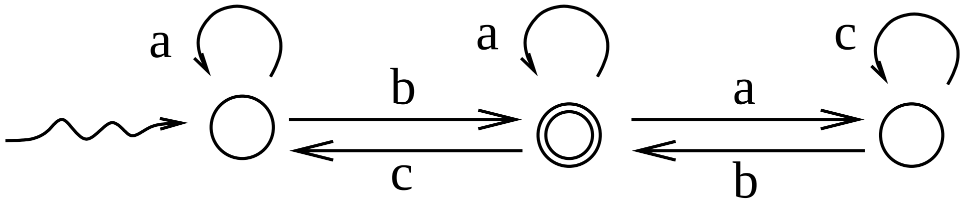


Figure 3: NFA for problem 6.

7. Which class of languages does each of these machine classes accept?

   (a) Deterministic finite automata. **Regular languages.**

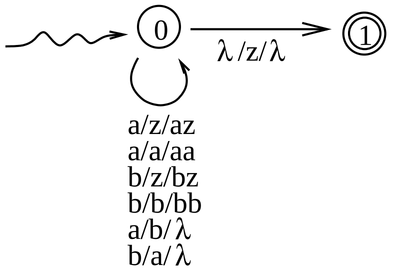
   (b) Non-deterministic finite automata. **Regular languages.**

   (c) Push-down automata. **Context-free languages.**

   (d) Turing Machines. **Recursively enumerable languages.**

8. Let $L = \{w \in \{a,b\}^* : \#_a(w) = \#_b(w)\}$, which is generated by the following context-free grammar.
   1. $S \to aSbS$
   2. $S \to bSaS$
   3. $S \to \lambda$

   Draw a PDA which accepts L.



9. The grammar below is an ambiguous CF grammar and is parsed by the LALR parser whose ACTION and GOTO tables are shown below. Write a computation of the parser for the input string *iiwaea*.

1. $S \to i_2 S_3$
2. $S \to i_2 S_3 e_4 S_5$
3. $S \to w_6 S_7$
4. $S \to a_8$

	a	i	e	w	\$	S
0	$s8$	$s2$		$s6$		1
1					halt	
2	$s8$	$s2$		$s6$		3
3			$s4$		$r1$	
4	$s8$	$s2$		$s6$		5
5			$r2$		$r2$	
6	$s8$	$s2$		$s6$		7
7			$r3$		$r3$	
8			$r4$		$r4$	

$\$_0$	$iiwaea\$$		
$\$_0 i_2$	$iwaea\$$	$s8$	
$\$_0 i_2 i_2$	$waea\$$	$s8$	
$\$_0 i_2 i_2 w_6$	$aea\$$	$s8$	
$\$_0 i_2 i_2 w_6 a_8$	$ea\$$	$s8$	
$\$_0 i_2 i_2 w_6 S_7$	$ea\$$	$r4$	4
$\$_0 i_2 i_2 S_3$	$ea\$$	$r3$	43
$\$_0 i_2 i_2 S_3 e_4$	$a\$$	$s4$	43
$\$_0 i_2 i_2 S_3 e_4 a_8$	\$	$s8$	43
$\$_0 i_2 i_2 S_3 e_4 S_5$	\$	$r4$	434
$\$_0 i_2 S_3$	\$	$r2$	4342
$\$_0 S_1$	\$	$r1$	43421
halt			

Problems Taken from the Final Examination of Spring 2022

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** There is some PDA that accepts 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.

   (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 $\{a^i b^j c^k \mid j = i + k\}$ is context-free.

   (v) **T** The intersection of any regular language with any context-free language is context-free.

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

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

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

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

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

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

   (xiii) **F** The intersection of any two undecidable languages is undecidable.

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

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

(xvi) **O** There exists a $\mathcal{P}$-TIME algorithm which finds a maximum independent set in any graph G.

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

(xxi) **F** EXP-SPACE = $\mathcal{P}$-SPACE.

(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 $\mathcal{P}$–TIME.

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

(xxix) **T** Primality, using binary numerals, is $\mathcal{P}$-TIME.

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

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

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

(xxxiii) **F** Every language generated by an unrestricted grammar is recursive.

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

(xxxvi) **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.

(xxxvii) **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.

(xxxviii) **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 $O(2^{2^n})$.

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

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

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

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

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

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

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

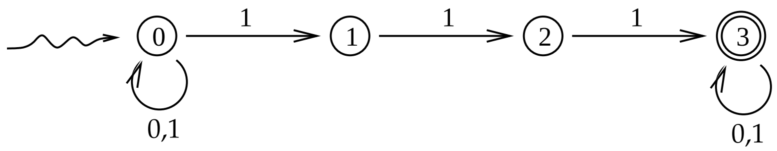
(xlviii) **O** There is a polynomial time algorithm which determines whether any two regular expressions are equivalent.

(xlix) **T** If L is a context-free language which contains the empty string, then $L\backslash\{\lambda\}$ must be context-free.

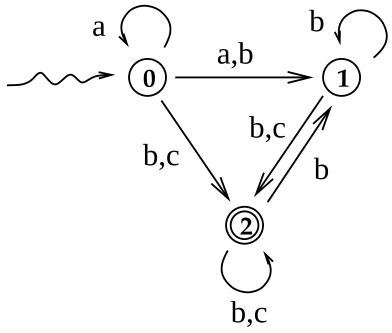
(l) **T** The computer language Pascal has Turing power.

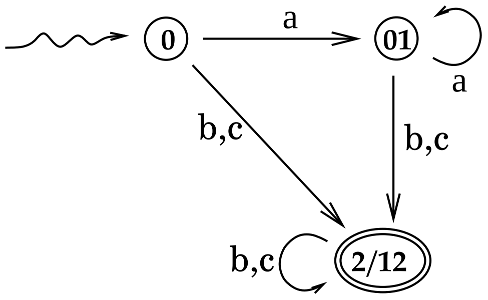
(li) **T** The binary integer factorization problem is co-$\mathcal{NP}$.

2. Find an NFA with at most 4 states which accepts the language of binary strings which contain the substring 111.



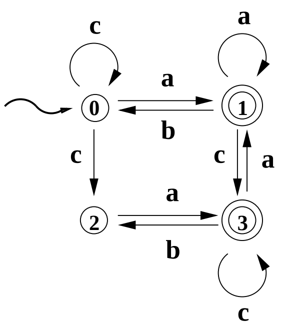
3. Construct a minimal DFA equivalent to the NFA shown below.



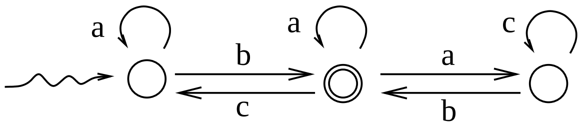


4. Find an NFA which accepts the language generated by this grammar.

$$
\begin{aligned}
S &\to aA|cS|cC \\
A &\to aA|bS|cB|\lambda \\
B &\to aA|cB|bC|\lambda \\
C &\to aB
\end{aligned}
$$



5. Give a regular expression which describes the language accepted by this NFA.

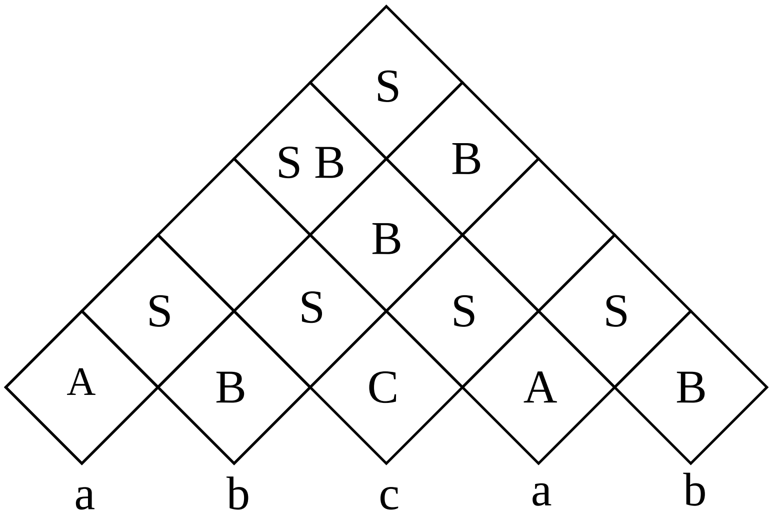


$a^*b(ca^*b + a + (ac^*b)^*$

6. Use the CYK algorithm to decide whether *abcab* is generated by the CNF grammar:

$S \to AB \,|\, BC \,|CA$
$A \to a$
$B \to SA \,|\, SS \,|\, b$
$C \to c$

by filling in the matrix.



Since the start symbol S appears in the top cell, the string *abcab* is generated by the grammar.

7. The LALR parser given for this grammar:
1. $E \to E -_2 E_3$
2. $E \to E *_4 E_5$
3. $E \to x_6$
contains errors, meaning that it might parse a string in a manner that would be considered incorrect by your programming instructor. Find those errors and correct them.

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

incorrect table

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

corrected table

8. Prove that the grammar given in Problem 7 is ambiguous by giving two different leftmost derivations for some string. (If you simply give two different parse trees, you have not answered the question.)

There are several correct answers for this problem: strings $x - x - x$, $x * x * x$, $x * x - x$ and $x - x * x$ are each ambigous. Here's one correct answer.

$E \Rightarrow E + E \Rightarrow E * E + E \Rightarrow x * E + E \Rightarrow x * x + E \Rightarrow x * x + x$

$E \Rightarrow E * E \Rightarrow x * E \Rightarrow x * E + E \Rightarrow x * x + E \Rightarrow x * x + x$

9. State the pumping lemma for regular languages.

For any regular language L
There exists a positive 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. $|xy| \le p$
3. $|y| \ge 1$
4. For any integer $i \ge 0$, $xy^iz \in L$

10. Give the verifier-certificate definition of the class $\mathcal{NP}$.

    A language L is $\mathcal{NP}$ if there exists a machine M and an integer k such that:
    1. For any $w \in L$ there exists a string c such that M accepts the string w, c in time $O(|w|^k)$
    2. For any $w \notin L$ and any string c, M does not accept the string x, c.

11. What is the importance nowadays of $\mathcal{NC}$?

    Nowadays, many of the most powerful computers have many processors working simultaneously. Given a large task, it would is helpful if the task can be divided up among the processors so as to run much faster. If the task is $\mathcal{NC}$, this can be done.

12. What complexity class contains sliding block problems?

    $\mathcal{P}$–SPACE. (The answer $\mathcal{P}$–SPACE complete is incorrect, since some, but not all, sliding block problems are $\mathcal{P}$–SPACE complete.)

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

    Let $x_1, x_2, \ldots x_n, K$ be an instance of the subset sum problem. (That means, does there exist a subsequence of $x_1, \ldots x_n$ whose sum is K?)

    We create an instance of the partition problem which has a solution if and only if the instance of the subset sum problem has a solution. Let $S = x_1 + x_2 + \cdots x_n$. We can assume $K \leq S$, since otherwise it is trivial that there is no solution. Our instance of the partition problem is the sequence $x_1, x_2, \ldots x_n, K + 1, S - K + 1$. The sum of that sequence is $2S + 2$. It has a subsequence which totals $S + 1$ if and only if the origintal problem has a solution.

14. Label each of the following sets as countable or uncountable.

    (i) **countable.** The set of integers.
    (ii) **countable.** The set of rational numbers.
    (iii) **uncountable.** The set of real numbers.
    (iv) **uncountable.** The set of binary languages.
    (v) **countable.** The set of co-$\mathcal{RE}$ binary languages.
    (vi) **uncountable.** The set of undecidable binary languages.
    (vii) **uncountable.** The set of functions from integers to integers.
    (viii) **countable.** The set of recursive real numbers.

Problems Taken from the Final Examination of Fall 2022

1. True/False/Open

    (ii) **T** The intersection of any two $\mathcal{NP}$ languages is $\mathcal{NP}$. **Think!**

    (vii) **T** The independent set problem is $\mathcal{P}$-TIME.

(viii) **T** If S is a recursive set of positive integers, then $\sum_{n\in S} 2^{-n}$ must be a recursive real number.

(ix) **T** Multiplication of matrices with binary numeral entries is $\mathcal{NC}$.

(x) **T** Equivalence of regular expressions is decidable.

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

(xii) **T** Equivalence of context-free grammars is co–$\mathcal{RE}$.

(xiii) **T** The language consisting of all fractions whose values are less than the natural logarithm of 5.0 is recursive.

(xvi) **T** Every sliding block problem is $\mathcal{P}$–SPACE.

(xviii) **T** If L is any $\mathcal{P}$–TIME language, there is an $\mathcal{NC}$ reduction of L to CVP, the Boolean circuit problem.

(xix) **T** There is a polynomial time algorithm for checking whether an integer is prime.

(xx) Every finite language is regular.

(xxi) **T** If L is a $\mathcal{P}$–TIME language, there is a Turing Machine which decides L in polynomial time.

(xxii) **T** If anyone ever finds a polynomial time algorithm for any $\mathcal{NP}$–complete language, then $\mathcal{P} = \mathcal{NP}$.

(xxiii) **T** RSA encryption is believed to be secure because it is believed that the factorization problem for integers is very hard.

(xxiv) **F** If S is a recursively enumerable set of positive integers, then $\sum_{n\in S} 2^{-n}$ must be a recursive real number.

2. Every language, or problem, falls into exactly one of these categories. For each of the languages, write a letter indicating the correct category.

   A Known to be $\mathcal{NC}$.
   B Known to be $\mathcal{P}$–TIME, but not known to be $\mathcal{NC}$.
   C Known to be $\mathcal{NP}$, but not known to be $\mathcal{P}$–TIME and not known to be $\mathcal{NP}$–complete.
   D Known to be $\mathcal{NP}$–complete.
   E Known to be $\mathcal{P}$–SPACE but not known to be $\mathcal{NP}$
   F Known to be decidable, but not known to be $\mathcal{P}$–SPACE.
   G $\mathcal{RE}$ but not decidable.
   H co-$\mathcal{RE}$ but not decidable.
   I Neither $\mathcal{RE}$ nor co-$\mathcal{RE}$.

   (a) **H** All C++ programs which do not halt if given themselves as input.

   (b) **A** All base 10 numerals for perfect squares.

   (c) **A** The Dyck language.

   (d) **H** $\{\langle G\rangle : L(G)$ is the Dyck language.$\}$

(e) **D** The Jigsaw problem. (That is, given a finite set of two-dimensional pieces, can they be assembled into a rectangle, with no overlap and no spaces.)

(f) **C** Factorization of binary numerals.

7. The grammar below is an ambiguous CF grammar with start symbol E, and is parsed by the LALR parser whose ACTION and GOTO tables are shown here. The ACTION table is missing actions for the second column, when the next input symbol is the "minus" sign. Fill it in. Remember the C++ precedence of operators. (Hint: the column has seven different actions: s2, s4, r1, r2, r3, r4, and r5, some more than once, and has no blank spaces.)

1. $E \to E -_2 E_3$
2. $E \to -_4 E_5$
3. $E \to E *_6 E_7$
4. $E \to (_8 E_9)_{10}$
5. $E \to x_{11}$

	x	$-$	$*$	(	)	\$	S
0	$s13$	$s4$		$s8$			1
1		$s2$	$s6$			*halt*	
2	$s13$	$s4$		$s8$			3
3		$r1$	$s6$		$r1$	$r1$	
4	$s11$	$s4$		$s8$			5
5		$r2$	$r2$		$r2$	$r2$	
6	$s11$	$s4$		$s8$			7
7		$r3$	$r3$		$r3$	$r3$	
8	$s11$	$r5$	$s4$	$s8$			9
9		$s2$	$s6$		$s6$		
10		$r4$	$r4$	$r4$	$r4$	$r4$	
11		$r5$	$r5$		$r5$	$r5$	

HERE HERE HERE

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

If $e = C_1 * C_2 * \cdots * C_k$ is any Boolean expression such that each C_i is the disjunction of three terms, each of which is either a variable or the negation of a variable, there is a graph G such that G has an independent set of order k if and only if e has a satisfying assignment. Each $C_i = t_{i,1} + t_{i,2} + {}_{i,3}$ Let the vertices of G be the set $V_{i,j}$ for all $1 \le i \le k$ and $j = 1, 2, 3$. There is an edge from $V_{i,j}$ to $V_{i',j'}$ if either $i = i'$ or $t_{i,j} * t_{i',j'}$ is a contradiction.

Problems Taken from the Final Examination of Spring 2023

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.

(iv) **F** The set of palindromes over $\{a, b\}$ is accepted by some DPDA.

(v) **T** The language $\{a^n b^n c^n \mid n \ge 0\}$ is in the class $\mathcal{NC}$.

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

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

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

(xi) **T** The Boolean Circuit Problem (also known as the CVP problem) is in $\mathcal{P}$.

(xiii) **T** co-$\mathcal{P} = \mathcal{P}$.

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

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

(xvi) **T** The set of binary numerals for prime numbers is $\mathcal{P}$-TIME.

(xvii) **F** Every language generated by an unrestricted (general) grammar is recursive.

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

(xx) **T** There is a mathematical proposition that is true but cannot be proved true.

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

(xxii) **T** There is a Turing machine which, when turned on, runs forever, writing the decimal expansion of π (the well-known ratio of the circumference of a circle to its diameter).

(xxiii) **T** The binary integer factorization problem is co-$\mathcal{NP}$.

(xxiv) **T** If L is $\mathcal{NP}$, there is a polynomial time reduction of L to the subset sum problem.

(xxv) **T** The intersection of any two $\mathcal{NP}$ languages is $\mathcal{NP}$.

(xxvi) **T** The intersection of any two co-$\mathcal{NP}$ languages is co-$\mathcal{NP}$.

(xxvii) **T** The intersection of any two co-$\mathcal{RE}$ languages is co-$\mathcal{RE}$.

(xxviii) **T** Multiplication of matrices with binary numeral entries is $\mathcal{NC}$.

(xxix) **T** Every recursively enumerable language is generated by an unrestricted (general) grammar.

(xxx) **T** Equivalence of context-free grammars is co–$\mathcal{RE}$.

(xxxi) **F** The language of all true mathematical statements is recursively enumerable.

(xxxii) **T** The language of all **provably** true mathematical statements is recursively enumerable.

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

(xxxiv) **T** If there exists a polynomial time algorithm for any $\mathcal{NP}$–complete problem, then $\mathcal{P} = \mathcal{NP}$.

(xxxv) **T** RSA encryption is accepted as secure by most experts, because they believe that the factorization problem for binary numerals is very hard.

(xxxvi) **F** The language of all $\langle G_1\rangle\langle G_2\rangle$ such that G_1 and G_2 are CF grammars which are **not** equivalent is co-$\mathcal{RE}$.

(xxxvii) **T** A real number x is recursive if and only if the set of fractions whose values are greater than x is recursive (decidable).

(xl) **O** If there is a solution to a given instance of any sliding block problem, there must be a solution of polynomial length.

(xli) **T** If the Boolean circut problem (CVP) is $\mathcal{NC}$, then $\mathcal{P} = \mathcal{NC}$.

2. (a) Give a context-sensitive grammar for $\{a^n b^n c^n : n > 0\}$

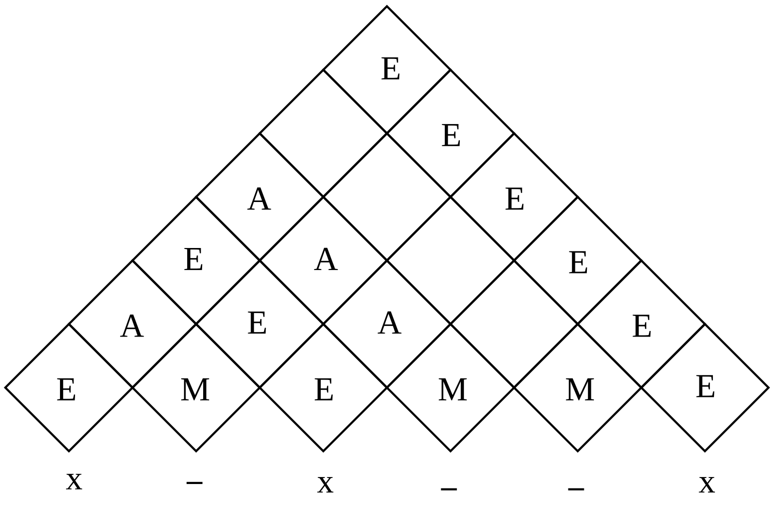
   (b) Using that grammar, give a derivation of the string *aaabbbccc*.

3. Illustrate an NFA which accepts the language generated by this grammar.

$S \to aA|cS|cC$
$A \to aA|bS|cB|\lambda$
$B \to aA|cB|bC|\lambda$
$C \to aB$

4. Use the CYK algorithm to decide whether $x - x - -x$ is generated by the CNF grammar below, by filling in the matrix.

$E \to ME$
$A \to EM$
$E \to AE$
$M \to -$
$E \to x$



The grammar generates $x - x - -x$, since the start symbol appears in the top cell.

1

6. Label each of the following sets as countable or uncountable.

   (a) **countable** The set of integers.

   (b) **countable** The set of rational numbers.

   (c) **uncountable** The set of real numbers.

   (d) **uncountable** The set of binary languages.

   (e) **countable** The set of co-RE binary languages.

   (f) **uncountable** The set of undecidable binary languages.

(g) **uncountable** The set of functions from integers to integers.

(h) **countable** The set of recursive real numbers.

(i) **countable** The set of $\mathcal{P}$–SPACE languages over the binary alphabet.

(j) **uncountable** The set of functions from the integers to the binary alphabet $\{0,1\}$.

8. Consider the CF grammar below. The ACTION and GOTO tables are given below, except that six actions are mising, indicated by question marks. Fill in the missing actions (below the question marks). The actions of your table must be consistent with the precedence of operators in C++.

1. $E \to E -_2 E_3$
2. $E \to -_4 E_5$
3. $E \to E *_6 E_7$
4. $E \to x_8$

	x	$-$	$*$	$\$$	E
0	$s8$	$s4$			1
1		$s2$	$s6$	HALT	
2	$s8$	$s4$			3
3		?	?	$r1$	
4	$s8$	$s4$			5
5		?	?	$r2$	
6	$s8$	$s4$			7
7		?	?	$r3$	
8	$s8$	$r4$	$r4$	$r4$	

9. Each of the languages listed below falls into one of these categories. Indicate which for each language.

A Known to be $\mathcal{NC}$.
B Known to be $\mathcal{P}$–TIME, but not known to be $\mathcal{NC}$.
C Known to be $\mathcal{NP}$, but not known to be $\mathcal{P}$–TIME and not known to be $\mathcal{NP}$–complete.
D Known to be $\mathcal{NP}$–complete.
E Known to be $\mathcal{P}$–SPACE but not known to be $\mathcal{NP}$
F Known to be decidable, but not known to be $\mathcal{P}$–SPACE.
G $\mathcal{RE}$ but not decidable.
H co-$\mathcal{RE}$ but not decidable.
K Neither $\mathcal{RE}$ nor co-$\mathcal{RE}$.

(a) bf D SAT.

(b) bf D SAT. 3-SAT.

(c) **B** 2-SAT.

(d) **D** The Independent Set problem.

(e) **D** The Subset Sum Problem.

(f) **H** The set of all (descriptions of) grammars which generate the Dyck language. That is, all $\langle G \rangle$ such that $L(G)$ is the Dyck language.

10. Prove that the halting problem is undecidable.