University of Nevada, Las Vegas Computer Science 456/656 Spring 2022

Answers to Assignment 7: Due Friday December 9 2022

This version: Sun Dec 11 19:07:22 PST 2022

Name:____

You are permitted to work in groups, get help from others, read books, and use the internet. Turn in the assignment in the manner given to you by our grader, Janeen Sudiacal.

I have given answers to some of the problems, including all the true false problems; thus, the homework is shorter than it looks. But be sure to go over those answers.

If you notice any errors, email me **immediately.**

1. True or False. If the question is currently open, write "O" or "Open."

   (i) **T** The complement of every regular language is regular.

   (ii) **F** The complement of every context-free language is context-free.

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

   (iv) **O** The complement of any $\mathcal{NP}$ language is $\mathcal{NP}$.

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

   (vi) **T** The complement of every recursive language is recursive.

   (vii) **F** The complement of every recursively enumerable language is recursively enumerable.

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

   (ix) **F** The context-free membership problem is undecidable.

   (x) **T** The factoring problem, where inputs are written in binary notation, is co-$\mathcal{NP}$.

   (xi) **T** If L_1 reduces to L_2 in polynomial time, and if L_2 is $\mathcal{NP}$, and if L_1 is $\mathcal{NP}$-complete, then L_2 must be $\mathcal{NP}$-complete.

   (xii) **F** Given any context-free grammar G and any string $w \in L(G)$, there is always a unique leftmost derivation of w using G.

   (xiii) **F** For any deterministic finite automaton, there is always a unique minimal non-deterministic finite automaton equivalent to it.

   (xiv) **F** The question of whether two regular expressions are equivalent is known to be $\mathcal{NP}$-complete.

   (xv) **T** The halting problem is recursively enumerable.

   (xvi) **T** The union of any two context-free languages is context-free.

   (xvii) **F** The question of whether a given Turing Machine halts with empty input is decidable.

(xviii) **T** The class of languages accepted by non-deterministic finite automata is the same as the class of languages accepted by deterministic finite automata.

(xix) **F** The class of languages accepted by non-deterministic push-down automata is the same as the class of languages accepted by deterministic push-down automata.

(xx) **T** The class of languages accepted by non-deterministic Turing Machines is the same as the class of languages accepted by deterministic Turing Machines.

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

(xxii) **T** If L_1 reduces to L_2 in polynomial time, and if L_2 is $\mathcal{NP}$, then L_1 must be $\mathcal{NP}$.

(xxiii) **T** Let π be the ratio of the circumference of a circle to its diameter. The problem of whether the n^{th} digit of the decimal expansion of π for a given n is equal to a given digit is decidable.

(xxiv) **T** There cannot exist any computer program that can decide whether any two C++ programs are equivalent.

(xxv) **T** Every context-free language is in the class $\mathcal{P}$-TIME.

(xxvi) **T** Every regular language is in the class $\mathcal{NC}$

(xxvii) **T** The language of all binary numerals for multiples of 23 is regular.

(xxviii) **F** The language of all binary strings which are the binary numerals for prime numbers is context-free.

(xxix) **T** Every context-free grammar can be parsed by some non-deterministic top-down parser.

(xxx) **T** If anyone ever proves that $\mathcal{P} = \mathcal{NP}$, then all one-way encoding systems will be insecure.

(xxxi) **T** If a string w is generated by a context-free grammer G, then w has a unique leftmost derivation if and only if it has a unique rightmost derivation.

(xxxii) **T** A language L is in $\mathcal{NP}$ if and only if there is a polynomial time reduction of L to SAT.

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

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

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

(xxxvi) **T** The question of whether two context-free grammars generate the same language is undecidable.

(xxxvii) **T** There exists some proposition which is true but which has no proof.

(xxxviii) **T** The set of all binary numerals for prime numbers is in the class $\mathcal{P}$.

(xxxix) **T** If L_1 reduces to L_2 in polynomial time, and if L_2 is $\mathcal{NP}$, and if L_1 is $\mathcal{NP}$-complete, then L_2 must be $\mathcal{NP}$-complete.

(xl) **F** Given any context-free grammar G and any string $w \in L(G)$, there is always a unique leftmost derivation of w using G.

(xli) **F** For any deterministic finite automaton, there is always a unique minimal non-deterministic finite automaton equivalent to it.

(xlii) **O** The question of whether two regular expressions are equivalent is $\mathcal{NP}$-complete.

(xliii) **F** No language which has an ambiguous context-free grammar can be accepted by a DPDA.

(xliv) **T** The union of any two context-free languages is context-free.

(xlv) **F** The question of whether a given Turing Machine halts with empty input is decidable.

(xlvi) **T** The class of languages accepted by non-deterministic finite automata is the same as the class of languages accepted by deterministic finite automata.

(xlvii) **F** The class of languages accepted by non-deterministic push-down automata is the same as the class of languages accepted by deterministic push-down automata.

(xlviii) **T** The intersection of any two regular languages is regular.

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

(l) **T** If L_1 reduces to L_2 in polynomial time, and if L_2 is $\mathcal{NP}$, then L_1 must be $\mathcal{NP}$.

(li) **T** Let $F(0) = 1$, and let $F(n) = 2^{F(n-1)}$ for $n > 0$. Then F is recursive.

(lii) **T** Every language which is accepted by some non-deterministic machine is accepted by some deterministic machine.

(liii) **F** The language of all regular expressions over the binary alphabet is a regular language.

(liv) **T** Let π be the ratio of the circumference of a circle to its diameter. (That's the usual meaning of π you learned in school.) The problem of whether the n^{th} digit of π, for a given n, is equal to a given digit is decidable.

(lv) **T** There cannot exist any computer program that decides whether any two given C++ programs are equivalent.

(lvi) **F** An undecidable language is necessarily $\mathcal{NP}$-complete.

(lvii) **T** Every context-free language is in the class $\mathcal{P}$-TIME.

(lviii) **T** Every regular language is in the class $\mathcal{NC}$

(lix) **F** Every function that can be mathematically defined is recursive.

(lx) **T** The language of all binary strings which are the binary numerals for multiples of 23 is regular.

(lxi) **F** The language of all binary strings which are the binary numerals for prime numbers is context-free.

(lxii) **F** Every bounded function from integers to integers is Turing-computable. (We say that f is *bounded* if there is some B such that $|f(n)| \leq B$ for all n.)

(lxiii) **F** The language of all palindromes over $\{0, 1\}$ is inherently ambiguous.

(lxiv) **F** Every context-free grammar can be parsed by some deterministic top-down parser.

(lxv) **T** Every context-free grammar can be parsed by some non-deterministic top-down parser.

(lxvi) **F** Commercially available parsers cannot use the LALR technique, since most modern programming languages are not context-free.

(lxvii) **F** The boolean satisfiability problem is undecidable.

(lxviii) **T** If anyone ever proves that $\mathcal{P} = \mathcal{NP}$, then all public key/private key encryption systems will be known to be insecure.

(lxix) **T** If a string w is generated by a context-free grammer G, then w has a unique leftmost derivation if and only if it has a unique rightmost derivation.

(lxx) **F** If a machine outputs a sequence of fractions which converges to a real number x, then x must be a recursive real number.

(But, do you see why?)

2. What class of machines accepts the class of context free languages?

   See Problem 21.

3. What class of machines accepts the class of recursively enumerable languages?

   Turing machines (TMs).

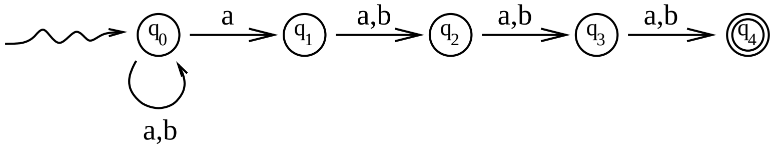
4. Using the context-free grammar with start symbol S and productions listed below, write two different leftmost derivations (not parse trees) of the string *iibwaanea*

$S \to a$
$S \to bLn$
$S \to wS$
$S \to iS$
$S \to iSeS$
$L \to \lambda$
$L \to SL$

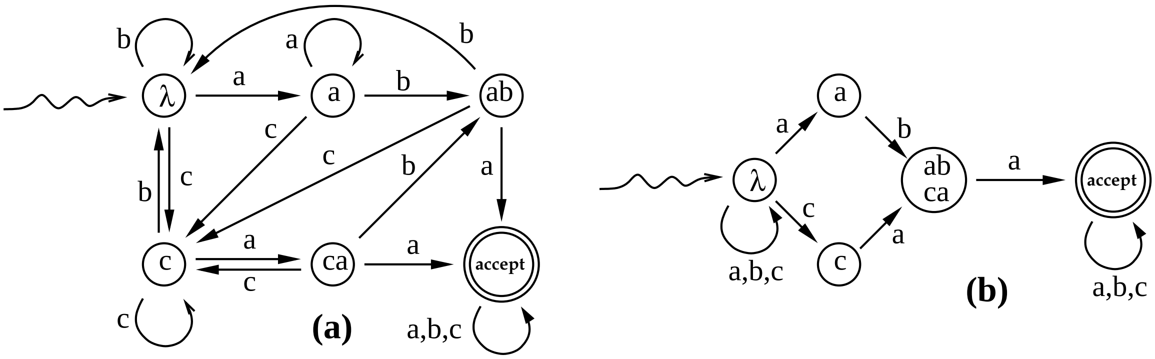
$S \Rightarrow iS \Rightarrow iiSeS \Rightarrow iibLneS \Rightarrow iibSLneS \Rightarrow iibwSLneS \Rightarrow iibwaLneS \Rightarrow iibwaSLneS \Rightarrow iibwaaLneS \Rightarrow iibwaaneS \Rightarrow iibwaanea$

$S \Rightarrow iSeS \Rightarrow iiSeS \Rightarrow iibLneS \Rightarrow iibSLneS \Rightarrow iibwSLneS \Rightarrow iibwaLneS \Rightarrow iibwaSLneS \Rightarrow iibwaaLneS \Rightarrow iibwaaneS \Rightarrow iibwaanea$

5. Draw an NFA with five states which accepts the language described by the regular expression $(a+b)^*a(a+b)(a+b)(a+b)$



6. Draw a DFA which accepts the language L over the alphabet $\{a, b, c\}$ consisting of all strings which contain either *aba* or *caa* as a substring. (My answer has six states.)



(a) shows a minimal DFA for L, while (b) shows an NFA. L is described by the regular expression $(a+b+c)^*(aba+caa)(a+b+c)^*$

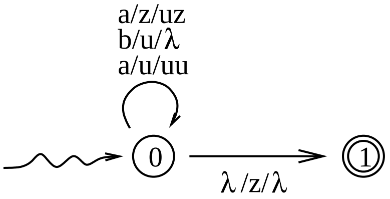
(To think about: what do the labels on the states mean?)

7. Find a context-free grammar which generates the language $L = \left\{a^i b^j c^k \,:\, i = j \text{ or } i = k\right\}$

L is the union of two context-free languages which have straightforward grammars. S_1 generates $\left\{a^i b^i c^k\right\}$, while S_2 generates $\left\{a^i b^j c^i\right\}$.

$S \to S_1 \,|\, S_2$
$S_1 \to S_1 c \,|\, T$
$T \to aTb \,|\, \lambda$
$S_2 \to aS_2c \,|\, B$
$B \to bB \,|\, \lambda$

8. Draw a state diagram for a PDA that accepts the Dyck language. (For ease of grading, use a and b instead of "(" and ")")



9. Every language we have discussed this semester falls into at least one of these categories.

   (a) $\mathcal{NC}$.

   (b) $\mathcal{P}$ but not known to be $\mathcal{NC}$.

   (c) $\mathcal{NP}$ but not known to be $\mathcal{P}$ and not known to be $\mathcal{NP}$-complete.

   (d) Co-$\mathcal{NP}$ but not known to be $\mathcal{P}$.

   (e) Known to be $\mathcal{NP}$-complete.

   (f) Recursive, but not known to be $\mathcal{NP}$.

(g) RE (Recursively enumerable), but not recursive.

(h) Co-RE, but not recursive.

(i) Neither RE nor co-RE.

State which of the above categories each of the languages below falls into.

(ii) (e) The 0-1 Traveling Salesman Problem.

(iii) (h) The diagonal language.

(iv) (e) L_{sat}, the set of satisfiable boolean expressions.

(v) (a) The language described by the regular expression a^*b^*.

(vi) (b) The Boolean circuit problem.

(vii) (b) Dynamic programming where all each subproblem can be worked in polynomial time and has Boolean output.

(viii) (a) The language $\{a^nb^n \,:\, n \geq 0\}$.

(ix) (c) Factorization of a binary numeral.

(x) (a) Dynamic programming where all subproblems have Boolean solutions, and each subproblem can be computed using only the output of the immediately previous subproblem.

(xi) (a) Multiplication of two binary numerals.

(xii) (e) The 0-1 traveling salesman problem.

(xiii) (f) The set of configurations of Rush Hour which are solvable.

(xiv) (a) Factorization of a unary numeral.

(xv) (g) The halting problem.

(xvi) (e) The clique problem.

(xvii) (b) Primality, where the input is written in binary.

(xviii) (a) The language generated by a given context-free grammar.

(xix) (a) The language of all finite monotone increasing sequences of arabic numerals for positive integers. (For example, "1,5,23,41,200,201" is a member of that language.)

(xx) (a) The language accepted by a given DFA.

(xxi) (h) The context-free grammar equivalence problem.

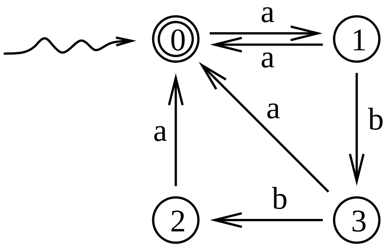
(xxii) (e) The independent set problem.

(xxiii) (h) The language consisting of all C++ programs P such that P does not halt if given P as its input file.

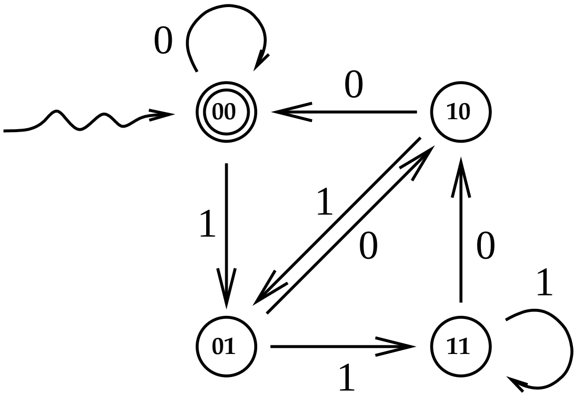
(xxiv) (e) 3-CNF-SAT.

(xxv) (a) The language described by the regular expression $\texttt{a(b+bc)}^*\texttt{(}\lambda\texttt{+a)(ba)}^*$

10. Draw the state diagram for a DFA that accepts the language described by the regular expression `(a(`λ`+b+bb)a)`*



11. Let L be the language of all binary numerals for positive integers which are multiples of 4. Thus, for example, the binary numerals for 0, 4, 8, 12, 16, 20 ... are in L. We allow a binary numeral to have leading zeros; thus (for example) $0011100 \in L$, since it is a binary numeral for 28. Draw a DFA with four states which accepts L.



12. What does it mean to say that machines M_1 and M_2 are *equivalent*?

    It means that given the same input, M_1 and M_2 will give the same output.

13. Prove that every decidable language can be enumerated in canonical order by some machine.

    Let L be a decidable language over an alphabet Σ. let M be a machine that decides L. let $w_1, w_2, \ldots$ be all the strings in Σ^* in canonical order. The following program decides L:

```
For all $i$ from 1 to $\infty$
    If($M$ accepts $w_i$)
        Write $w_i$
```

    The outer loop will run forever, since the conditional statement can always be evaluated in finite time.

14. Repeat of Problem 23

15. Give a $\mathcal{P}$-time reduction of the subset sum problem to the partition problem.

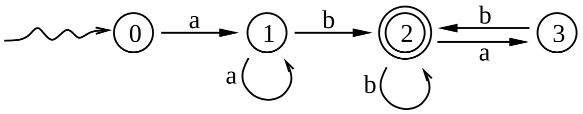
    An instance of the subset sum problem is a number K and a set of n objects of positive weight. A solution to that instance is a subset of the objects whose total weight is K. Let S be the sum of the weights of the objects. Without loss of generality, $K \leq S$, since otherwsie there can be no solution.

Given that instance, we define an instance of the partition problem to consist of the n objects together with two additional objects, one of weight $K+1$ and the other of weight $S-K+1$. The sum of the weights of thos $n+2$ objects is $2S+2$. That instance has a solution if there is the objects can be partitioned into two equal weight sets, that is, each set must have weight $S+1$.

We now prove that the instance of the subset sum problem has a solution if and only if our instance of the partition problem has a solution. Suppose there is a subset of the set of original weights whose total weight is K. Then, that set combined with the one new object of weight $S-K+1$ is a set whose weight is half the total.

Conversely, suppose that the instance of the partition problem has a solution. The two new weights cannot both be in either half, since their total weight is $S+2$. Thus one of the two halves has the new object of weight $S-K+1$. The remaining objects in that set must have total weight K.

16. Draw the state diagram for a minimal DFA that accepts the language described by the regular expression $\mathtt{a^*a(b+ab)^*}$



17. Write a regular expression for the language accepted by the NFA shown in Figure 1.

$b^*(a+c+ba^*c)(a+c+bb^*(a+c)+bb^*ba^*c)^*$

This expression is not the only solution, and may not be the shortest one.

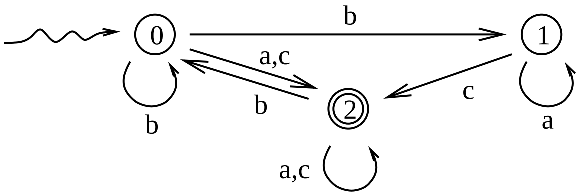


Figure 1: The NFA for Problems 17 and 32.

18. Let L be the language of all binary numerals for positive integers equivalent to 2 modulo 3. Thus, for example, the binary numerals for 2, 5, 8, 11, 14, 17 ... are in L. We allow a binary numeral to have leading zeros; thus (for example) $001110 \in L$, since it is a binary numeral for 14. Draw a minimal DFA which accepts L.

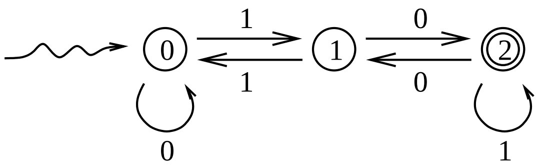


Figure 2: DFA which accepts the language of binary numerals equivalent to 2 modulo 3

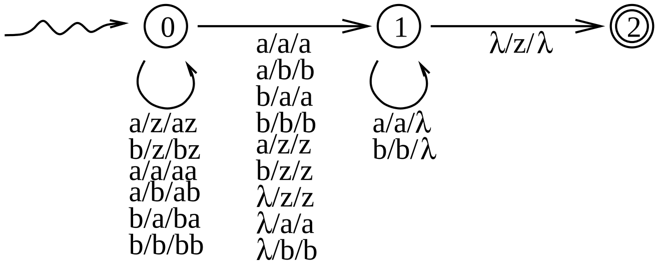


Figure 3: PDA which accepts by final state the palindromes over $\{\mathbf{a}, \mathbf{b}\}$

19. Design a PDA that accepts the language of all palindromes over the alphabet $\{a, b\}$.

20. Consider the context-free grammar with start symbol S and productions given below. Write a leftmost derivation of the string $bswsbwsnn$.

    $S \to s$
    $S \to bLn$
    $S \to wS$
    $L \to \lambda$
    $L \to SL$

    $S \Rightarrow bLn \Rightarrow bSLn \Rightarrow bsSLn \Rightarrow bswSLn \Rightarrow bswsLn \Rightarrow bswsSLn$
    $\Rightarrow bswsbLnLn \Rightarrow bswsbSLnLn \Rightarrow bswsbwSLnLn$
    $\Rightarrow bswsbwsLnLn \Rightarrow bswsbwsnLn \Rightarrow bswsbwsnn$

21. What class of machines accepts the class of context free languages?

    Push down Automata (PDAs)

22. What class of machines accepts the class of regular languages?

    Finite Automata (DFAs or NFAs).

23. What is the Church-Turing Thesis, and why is it important?

    Every computation that can be done by any machine can be done by a Turing machine. This is important because, if we can prove that no Turing machine can perform a certain computation, we know that no machine can perform that computation. Turing machines are simple, making these proofs easier.

24. What does it mean to say that a language can be recursively enumerated in *canonical order*? What is the class of languages that can be so enumerated?

    If u and v are strings, we say that u comes before v in the canonical the canonical order if either $|u| < |v|$, or $|u| = |v|$ and u comes before v in lexical [alphabetic] order.

    We say that a language L can be enumerated in canonical order if there is some machine with no input which outputs all the strings of L in canonical order, and does not output any other strings.

    A language L can be enumerated in canonical order if and only if it is recursive [decidable].

25. Repeat of Problem 12.

26. Give a definition of the language class $\mathcal{NP}$–TIME. Two definitions were given in class.

(a) A language L is in the class $\mathcal{NP}$–TIME if there is some non-deterministic machine which accepts L in polynomial time.

(b) A language L is in the class $\mathcal{NP}$–TIME if there is some deterministic machine which, for each $w \in L$, verifies that $w \in L$ in polynomial time if given another string, called a witness or certificate for w.

27. Give the definition of a *polynomial time reduction* of a language L_1 to another language L_2.

Let Σ_1, σ_2 be the alphabets of L_1 and L_2, respectively. A polynomial time reduction of L_1 to L_2 is a function $R : \Sigma_1^* \to \Sigma_2^*$ which is computable in polynomial time such that, for any $w \in \Sigma_1^*$, $w \in L_1$ if and only if $R(w) \in L_2$.

28. Give a definition of $\mathcal{NP}$*-complete* language.

A language L is $\mathcal{NP}$-complete if

(a) L is in the class $\mathcal{NP}$–TIME

(b) For any language L_2 in the class $\mathcal{NP}$-TIME, there is a polynomial time reduction of L_2 to L.

29. Give a definition of a *decidable* language.

We say that $L \subseteq \Sigma^*$ is decidable if there is a machine M which, given $w \in \Sigma^*$, halts with input w, outputs 1 if $w \in L$ and outputs 0 if $w \notin L$.

30. We say a binary string w over is *balanced* if w has the same number of 1's as 0's. Let L be the set of balanced binary strings. Give a context-free grammar for L.

Here is a simple ambiguous context-free grammar for L.
$S \to SS$
$S \to aSb$
$S \to bSa$
$S \to \lambda$

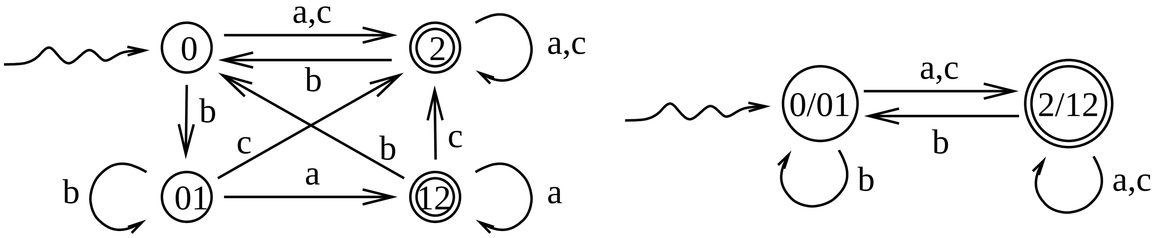
Here is an unambiguous grammar for L.
$S \to aAbS$
$S \to bBaS$
$S \to \lambda$
$A \to aAbA$
$A \to \lambda$
$B \to bBaB$
$B \to \lambda$

31. Give a Chomsky normal form (CNF) grammar for the language of all palindromes of odd length over the alphabet $\{a, b\}$.

$S \to a$　　$C \to SA$
$S \to b$　　$D \to SB$
$S \to AC$　　$A \to a$
$S \to BD$　　$B \to b$

The grammar is unambiguous, although that was not a requirement. Unlike for the previous problem, I do not believe there is a simpler ambiguous answer.

32. Construct a minimal DFA equivalent to the NFA shown in Figure 1.



The first figure is the DFA obtained by the powerset method. The second figure is obtained by minimization.

33. Consider the context-free grammar G, with start symbol S and productions as follows:

$S \to s$
$S \to bLn$
$S \to iS$
$S \to iSeS$
$L \to \epsilon$
$L \to LS$

Prove that G is ambiguous by giving two different leftmost derivations for some string.

$S \Rightarrow iSeS \Rightarrow iiSeS \Rightarrow iiaeS \Rightarrow iiaea$
$S \Rightarrow iS \Rightarrow iiSeS \Rightarrow iiaeS \Rightarrow iiaea$

34. The grammar below is an unambiguous CF grammar for the Dyck language, and is parsed by the given LALR parser. Write a computation of the parser for the input string $aabb$.

1 $S \to a_2 S_3 b_4 S_5$

2 $S \to \lambda$

	a	b	\$	S
0	$s2$		$r2$	1
1			HALT	
2	$s2$	$r2$		3
3		$s4$		
4	$s2$	$r2$	$r2$	5
5	$r1$	$r1$	$r1$	

$\$_0$	:	$aabb$		
$\$_0 a_2$	:	abb	$s2$	
$\$_0 a_2 a_2$	:	bb	$s2$	
$\$_0 a_2 a_2 S_3$	:	bb	$r2$	2
$\$_0 a_2 a_2 S_3 b_4$	:	b	$s4$	2
$\$_0 a_2 a_2 S_3 b_4 S_5$	:	b	$r2$	22
$\$_0 a_2 S_3$	:	b	$r1$	221
$\$_0 a_2 S_3 b_4$	:		$b4$	221
$\$_0 a_2 S_3 b_4 S_5$	:		$r2$	2212
$\$_0 S_1$	:		$r1$	22121
$\$_0 S_1$	:		HALT	22121

The output is 22121.

35. The following ambiguous grammar, with start symbol E, generates certain expressions. The parser illustrated below is intended to parse a string in accordance with C++ rules. However, the action table has one error. Identify that error, and fix it.

$E \rightarrow x_2$
$E \rightarrow E - {}_3E_4$
$E \rightarrow -{}_5E_6$
$E \rightarrow ({}_7E_8)_9$

	x	$-$	(	)	\$	S
0	$s2$	$s5$	$s7$			1
1		$s3$			HALT	
2		$r1$		$r1$	$r1$	
3	$s2$	$s5$	$s7$			4
4		$r2$		$r2$	$r2$	
5	$s2$	$s5$	$s7$			6
6		$r3$		$r3$	$r3$	
7	$s2$	$s5$	$s7$			8
8		$s3$		$s9$		
9		$r4$		$r4$	$r4$	

36. For each of the following languages, state whether the language is regular, context-free but not regular, context-sensitive but not context-free, or not context-sensitive.

    (a) **Regular.** The set of all strings over the alphabet $\{a, b\}$ of the form $a^n b^m$.

    (b) **Context-free, not regular.** The set of all strings over the alphabet $\{a, b\}$ of the form $a^n b^n$.

    (c) **Context-sensitive, not context-free.** The set of all strings over the alphabet $\{a, b, c\}$ of the form $a^n b^n c^n$.

    (d) **Context-free, not regular.** The set of all strings over the alphabet $\{a, b, c\}$ which are **not** of the form $a^n b^n c^n$.

37. Draw a minimal DFA which accepts the language L over the alphabet $\Sigma = \{a, b, c\}$ consisting of all strings which contain either *aba* or *caa* as a substring.

    See Problem 6.

38. State the pumping lemma for regular languages accurately. If you have all the right words but in the wrong order, that means you truly do not understand the lemma, and you might get no partial credit at all.

    For any regular language L
    there is an integer p such that
    for any $w \in L$ such that $|w| \geq p$
    there exist strings x, y, z such that
    $w = xyz$,
    $|xy| \leq p$,
    y is not the empty string and
    for any integer $i \geq 0$ $xy^i z \in L$

39. These are reduction problems. I could give one of them on the test. The proof should be very informal.

(a) Find a $\mathcal{P}$-time reduction of 3-CNF-SAT to the independent set problem.

Let B be a Boolean expression in 3-CNF form. That is, B is the conjunction of clauses $C_1, \ldots C_K$ and, for each i, $C_i = t_{i,1} + t_{i,2} + t_{i,3}$, where each term $t_{i,j}$ is either a variable or the negation of a variable. We define a graph $G_B = (V, E)$ as follows:

- $V = \{v_{i,j} \, : \, 1 \leq i \leq K, \; 1 \leq j \leq 3\}$.
- $E = \{\{v_{i,j}, v_{i',j'}\} \, : \, i = i'$ or $t_{i,j} * t_{i',j'}$ is a contradiction$\}$

We now show that G_B has an independent set of K vertices if and only if B is satisfiable. Let $I \subseteq V$ be an independent set or order K. Since G is the exact union of K 3-cliques, exactly one member of I lies in each clique. Let $X = \{x_1, \ldots x_m\}$ be the variables that appear in B, either positively or negatively. We define an assignment $\alpha : X \to \{0, 1\}$ as follows.

- If $x_\ell = t_{i,j}$ where $v_{i,j} \in I$, $\alpha(x_\ell) = 1$.
- If $!x_\ell = t_{i,j}$ where $v_{i,j} \in I$, $\alpha(x_\ell) = 0$.
- Otherwise, we choose $\alpha(x_\ell)$ arbitrarily.

The assignment α is well-defined. Suppose $\alpha(x_\ell)$ is assigned both true and false. Then $t_{i,j} = x_\ell$ and $t_{i',j'} =!x_\ell$ for some i, i', j, j' where $v_{i,j}, v_{i',j'} \in I$ But this is impossible, since there is an edge which connects those two vertices. Under the assignment, at least one term of each clause of B is true, hence B is satisfiable.

Conversely, suppose α is a satisfying assignment of B. For each clause, at least one term is assigned true. Choose D to be a set consisting of one term assigned true for each clause. a total of K terms. Let $I \subseteq V$ be the vertices of G corresponding to D. Then $|I| = K$. There is exactly one member of I in each 3-clique. Two members of I are connected by an edge, the corresponding members of I contradict each other, since α assigns just one Boolean value to each variable, hence I is a K-independent set.

(b) Give a $\mathcal{P}$-time reduction of the subset sum problem to the partition problem.

See Problem 15.

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

Let L be a recursively enumerable language; let M be a machine that enumerates L. We let our machine be a program which reads a string w and halts if and only if $w \in L$. Let $w_1, w_2, \ldots$ be the strings of L in the order they are enumerated by M.

Here is the program:

```
Read $w$
For $i = 1$ to $\infty$
  If $(w = w_i)$ HALT
```

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

Let M be a Turing machine that accepts a language L over an alphabet Σ. Let $w_1, w_2, \ldots$ be an enumeration of Σ^* in canonical order. (That enumeration is easy to compute.) The following program runs forever, enumerating L.

```
For ($t = 1, 2, \ldots$)  // infinite loop
  For ($i = 1, 2, \ldots t$)  // finite loop
    If ($M$ accepts $w_i$ within $t$ steps and $w_i$ has not yet been written)
      WRITE $w_i$.
```

If w_i is written, then M clearly accepts w_i, hence $w_i \in L$. Conversely, if $w_i \in L$, then M accepts L at its t^{th} step for some t, and hence will be written during the t^{th} iteration of the outer loop.

42. Prove that the halting problem is undecidable.

Assume the halting problem is decidable. Let D be a machine which executes the following program.

```
Read a machine description $\langle M \rangle$.
If ($M$ halts with input $\langle M \rangle$)
  Loop forever
Else halt.
```

Since the halting problem is decidable, the program will not get stuck evaluating the condition of the "if" statement.

Now run D with input $\langle D \rangle$.

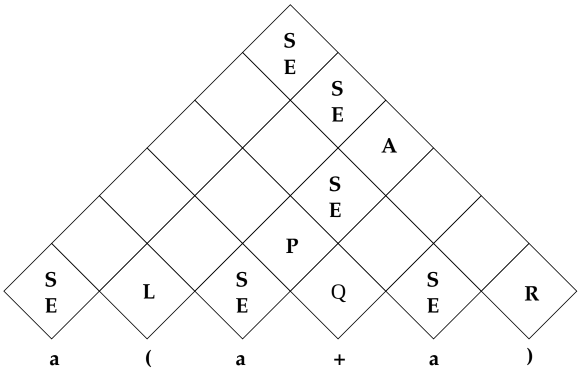
If D accepts $\langle D \rangle$, then D will not accept $\langle D \rangle$ since it will loop forever. If D does not accept $\langle D \rangle$, then D halts, and hence accepts $\langle D \rangle$. We now have a contradiction, hence the machine D cannot exist, hence the halting problem is undecidable.

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

$S \to a$
$E \to a$
$S \to LA$
$E \to LA$
$L \to ($
$A \to ER$
$R \to)$

$S \to PE$
$E \to PE$
$S \to EE$
$E \to EE$
$P \to EQ$
$Q \to +$

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



We conclude that $a(a+a)a \in L$ since the topmost cell contains the start symbol.