University of Nevada, Las Vegas Computer Science 456/656 Fall 2021

Answers to Assignment 3. Due 11:30 AM Monday September 20, 2021

Name:_____

You are permitted to work in groups, get help from others, read books, and use the internet.

Do not hand this homework in. It will not be graded, but I will go over it in class on September 20.

1. Starting on page 138 of the sixth edition of your textbook or on page 137 of the fifth edition, work problems 9(b), 9(e), 12(c),and 15. All the grammars I give here are unambiguous.

   9b in sixth edition, not in fifth edition. $\{a^n b^m : n = m - 1\}$

   $S \to Ab$
   $A \to aAb$
   $A \to \lambda$

   9c in sixth edition, 7c in fith edition. This one wasn't assigned, but I got confused and thought it was, since "c" and "e" look alike to me on the screen. $L = \{a^n b^m : n \neq 2m\}$

   Let $L_1 = \{a^n b^m : n > 2m\}$ and $L_2 = \{a^n b^m : n < 2m\}$. Then $L = L_1 + L_2$. Pick variables S_1 and S_2, and define productions such that $L_1 = L(S_1)$ and $L_2 = L(S_2)$, then use the productions $S \to S_1 \mid S_2$ to get both cases.

   The exercise requests any CF grammar which generates L, but just to show you it can be done, I give an unambiguous grammar.

   $S \to S_1$
   $S_1 \to aA$
   $A \to aA$
   $A \to aaCb$
   $A \to \lambda$
   $C \to aaCb \quad C \to \lambda$
   $S \to S_2$
   $S_2 \to Bb$
   $S_2 \to aBb$
   $B \to Bb$
   $B \to aaCb$
   $B \to \lambda$

   It would have been simpler to have an ambiguous grammar, but my decision to define an unambiguous grammar imposed discipline, and made it easier for me to prove that I had generated all strings of L and no others. I used a technique to maintain this discipline, which involved drawing a graph where one axis measures n and the other m.

   9e in the sixth edition, 7e in the fifth edition: $L = \{w \in \{a, b\} : n_a(w) \neq n_b(w)\}$ L is the union of two CF languages, one more $n_a > n_b$ and the other the opposite. To avoid confusion, I will start by giving you a CF grammar for $\{w \in \{a, b\} : n_a(w) = n_b(w)\}$. This grammar is ambiguous, although an unambiguous CF grammar exists.

$S \to SS$
$S \to aSb$
$S \to bSa$
$S \to \lambda$

Now for L itself, we have the ambiguous grammar:

$S \to A|B$
$A \to AA|EaE$
$B \to BB|EbE$
$E \to EE|aEb|bEa|\lambda$

(12c) $\left\{a^n b^m c^k \ : \ k = n + m\right\}$

$S \to aSc$
$S \to B$
$B \to bBc$
$B \to \lambda$

(15) $\left\{a^n w w^R b^n \ : \ w \in \Sigma^* \ and \ n \geq 1\right\}$

$S \to aAb$
$A \to aAb$
$A \to B$
$B \to aBa$
$B \to bBb$
$B \to lambda$

2. I will give the proof by contradiction that $\sqrt{2}$ is irrational in class. Prove by contradiction that $\sqrt{3}$ is irrational.

   Suppose $\sqrt{3}$ is rational. Then $\sqrt{3} = \dfrac{p}{q}$, where the fraction is reduced to the lowest terms. Squaring both sides, we obtain $3 = \dfrac{p^2}{q^2}$ Clearing the fraction we have $3q^2 = p^2$, and thus p^2 is a multiple of 3, which implies that p is a multiple of 3. Write $p = 3k$, where k is an integer. Then $3q^2 = p^2 = 9k^2$. Dividing by 3, we have $q^2 = 3k^2$, that is, q^2 is also a multiple of 3, which imples that q is a multiple of 3.
   This contradicts the hypothesis that the fraction $\dfrac{p}{q}$ is reduced to the lowest terms. Thus $\sqrt{3}$ must not be rational.

3. You have probably seen the easy (high school) proof that the sum of the first n positive integers is $\frac{n(n+1)}{2}$. That formula was made famous by Johann Carl Friedrich Gauss, who came up with the proof at age 10 to work a class asignment in his head in less than a minute while the other pupils were busy for an hour.

He was not the first to find the formula, which was well-known at the time. The teacher knew it, but didn't tell the kids. There is also a proof by induction, which I will give in class.

Use induction to prove that the sum of the squares of the first n positive integers is $\frac{n(n+1)(2n+1)}{6}$.

The formula holds if $n = 1$, since $1^2 = \dfrac{1(1+1)(2*1+1)}{6}$. Suppose the formula holds for a given n. Then

$$\begin{aligned} 1^2 + 2^2 + \cdots + n^2 + (n+1)^2 &= \frac{n(n+1)(2n+1)}{6} + (n+1)^2 \\ &= \frac{2n^3 + 3n^2 + n}{6} + \frac{6(n+1)^2}{6} \\ &= \frac{2n^3 + 3n^2 + n + 6n^2 + 12n + 6}{6} \\ &= \frac{2n^3 + 9n^2 + 13n + 6}{6} \\ &= \frac{(n+1)(n+2)(2n+3)}{6} \end{aligned}$$

and we are done.

The sum of the cubes of the first n positive integers is $\frac{n^2(n+1)^2}{4}$.

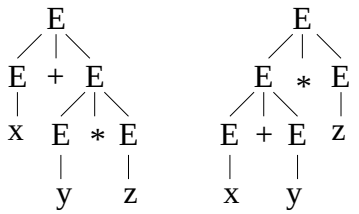
The context-free grammar G_1 given below generates an algebraic language L, where the strings of L are C++ expressions with variables and operators. There are only three C++ variables, x, y, and z, and there are only four operations, addition, subtraction, negation, and multiplication. The start symbol is E (for "expression") instead of the usual S, and the alphabet of L is $\{x, y, z, +, -, *, (,)\}$

(a) Prove that G_1 is ambiguous by giving two different left-most derivations of the string $x + y * z$ and the corresponding derivation trees.

$E \Rightarrow E + E \Rightarrow x + E \Rightarrow x + E * E \Rightarrow x + y * E \Rightarrow x + y * z$

$E \Rightarrow E * E \Rightarrow E + E * E \Rightarrow x + E * E \Rightarrow x + y * E \Rightarrow x + y * z$

1. $E \to E + E$
2. $E \to E - E$
3. $E \to E * E$
4. $E \to -E$
5. $E \to (E)$
6. $E \to x$
7. $E \to y$
8. $E \to z$



(b) When I input a certain string $w \in L$ into my top-down parser, it outputs 352674513687, the encoding of a left-most derivation of w. We get $w = (x - y) * -(x * z + y)$.

(c) The grammar G_1 is does not "respect" the C++ precedence of operators. (What does that mean?) Give an unambiguous grammar G_2 for L which respects C++ precedence rules. Use three grammar variables: E for "expression," T for "term," and F for "factor." I will help get you started:

1. $E \to E + T$
2. $E \to E - T$
3. $E \to T$
4. $T \to T * F$
5. $T \to F$
6. $F \to -F$
7. $F \to (E)$
8. $F \to x$
9. $F \to y$
10. $F \to z$

4. Fill in the blanks.

(a) If M_1 is an NFA with n states, then M_1 is equivalent to a DFA M_2 with no more than 2^n states.

(b) The CYK algorithm determines whether a given string of length n is generated by a given context-free grammar in $O(n^3)$ time.

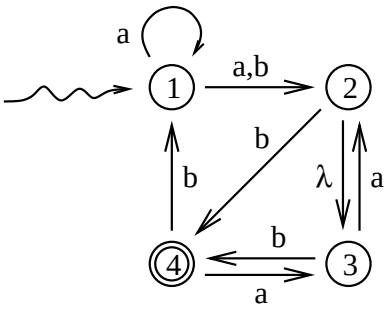
(c) A derivation tree of a string w generated by context-free grammar G can be labeled by writing, next to each variable symbol in the tree, the number of a production of G. The encoding of a left-most derivation of w is obtained by visiting the internal nodes of the corresponding derivation tree in preorder.

5. State the pumping lemma for regular languages by filling in the blanks.

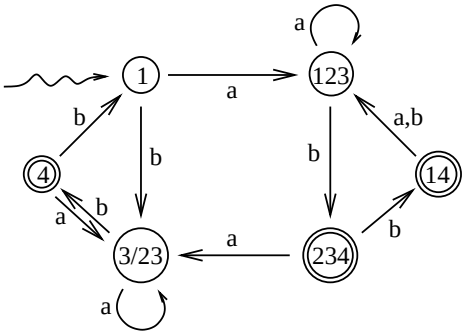
   If L is a regular language, there exists an integer p, which we call the pumping length of L, such that for every $w \in L$ such that $|w| \geq p$, there exist strings x, y and z such that the following four statements hold:

   1. w = xyz
   2. $|y| > 0$
   3. $|xy| \leq p$
   4. For any $i \geq 0$, the string xy^iz is a member of L.

6. Design a minimal DFA equivalent to the following NFA.



	a	b
1	123	23
3	23	4
4	3	1
23	23	4
123	123	234
234	23	14
14	123	123

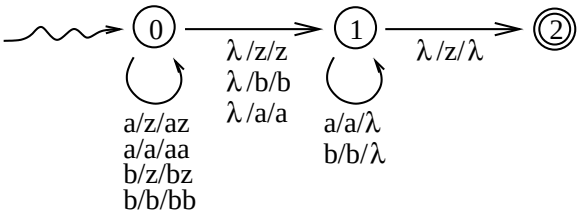


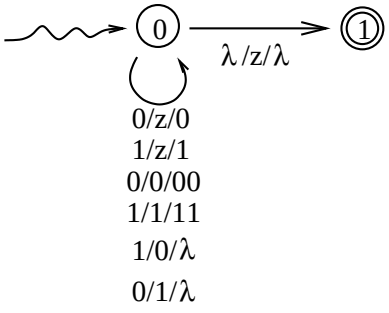
7. (a) Design a PDA which accepts the language L of all even length palindromes over $\{a, b\}$,

   $S \to aSa$
   $S \to bSb$
   $S \to \lambda$

   which is generated by this context-free grammar. You may describe your machine either by writing the transitions or by drawing a figure.

   (b) Let L be the language consisting of all binary strings with equally many zeros and ones. For example, 0100011011 and 1001 are in L, but 01101001100101101 is not. Design a DPDA which accepts L. You may describe your machine either by writing the transitions or by drawing a figure.





8. Let $L = \{a^n b^n c^n : n \geq 0\}$. As you know, L is not context-free.

   (a) Give two context-free languages whose intersection is L.

   $L_1 = \{a^n b^n c^m\}$ and $L_2 = \{a^n b^m c^m\}$.

   (b) Give a context-sensitive grammar for L.

   $S \to abc$
   $ab \to aAb$
   $aAb \to aabbA$
   $Ab \to bA$
   $Ac \to cc$
   I think this grammar is easier to understand than the one on page 301 of your textbook. Here's how to generate $a^4b^4c^4$.

   $S \Rightarrow abc \Rightarrow aAbc \Rightarrow aabbAc \Rightarrow aaAbbAc \Rightarrow aaabbAbAc \Rightarrow aaaAbbbAbAc \Rightarrow aaabAbbAbAc \Rightarrow aaabbAbAbAc \Rightarrow aaabbbAAbAc \Rightarrow aaabbbAbAAc \Rightarrow aaabbbbAAAc \Rightarrow aaabbbbAAcc \Rightarrow aaabbbbAccc \Rightarrow aaabbbbcccc$

9. Give a context-free grammar for the complement of L. If w is not in L, there must be one of only six reasons:

   1. w contains the substring ba.
   2. w contains the substring cb.

   If neither of those holds, then $w = \{a^i b^j c^k\}$ Then either $i \neq j$ or $j \neq k$, or more explicitly, $i < j$, $i > j$, $j < k$, or $j > k$.

   We can then define six CF languages which have these six properties.

   Let L_1 be the set of all strings over $\{a, b, c\}$ which contain the substring ba.

   Let L_2 be the set of all strings over $\{a, b, c\}$ which contain the substring cb.

   Let $L_3 = \{a^i b^j c^k : i < j\}$

   Let $L_4 = \{a^i b^j c^k : i > j\}$

   Let $L_5 = \{a^i b^j c^k : j < k\}$

   Let $L_6 = \{a^i b^j c^k : j > k\}$

   The complement of L is then $L_1 \cup L_2 \cup L_3 \cup L_4 \cup L_5 \cup L_6$. L_1 and L_2 are regular, hence CF. Each of the other four has an obvious CF grammar. For example, $L_3 = L(S_3)$ where

   $S_3 \to aAC$
   $A \to aA$
   $A \to aAb$
   $A \to \lambda$
   $C \to cC$
   $C \to \lambda$

10. 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** 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.

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

(v) **T** The intersection of any three regular languages is regular.

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

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

(viii) **F** The complement of any context-free language is context-free.

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

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

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

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

(xiii) **T** If G is a context-free grammar, the question of whether $L(G) = \emptyset$ is decidable.

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

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

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

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

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

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

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

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

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

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

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

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

(xxix) **F** If G is a context-free grammar, the question of whether $L(G) = \Sigma^*$ is decidable, where Σ is the terminal alphabet of G.

(xxx) **T** The language of all fractions (using base 10 numeration) whose values are less than π is decidable. (A *fraction* is a string. "314/100" is in the language, but "22/7" is not.)

(xxxi) **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.

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

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

(xxxiv) **T** There is an uncomputable function which grows faster than any computable function.

(xxxv) **T** There exists a machine that runs forever and outputs the decimal digits of π (the well-known ratio of the circumference of a circle to its diameter).

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

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