

SOLUTIONS: Homework 5, CS301, McConnell, Spring '09

Reading: Sections 3.1 and 3.2.

Corrections made 4/6 16:45

1. Consider the grammar $S \rightarrow bS|Sb|a$.

- (a) The context-free language it generates happens to be regular. Give a regular expression for it.

Solution: b^*ab^* .

- (b) Show that the grammar is ambiguous by giving two leftmost derivations for the string bab and their parse trees.

Solution:

$$S \Rightarrow bS \Rightarrow bSb \Rightarrow bab.$$

$$S \Rightarrow Sb \Rightarrow bSb \Rightarrow bab.$$

In one of the parse trees, the left-branching b is above the right-branching b , and in the other the right-branching b is above the left-branching one.

- (c) Rewrite the grammar so that it is not ambiguous. *Hint: introduce a new non-terminal symbol A .*

Solution:

$$S \rightarrow bS|A; A \rightarrow Ab|a.$$

The left-branching b 's are forced to be produced before the right-branching ones.

2. We have seen examples of the use of *setbuilder notation* in specifying a language, such as $\{a^n b^m c^n | m > 0 \text{ and } n \geq 0\}$, which is the set of words that consist of equal-sized blocks of a 's and c 's, separated by at least one b . Review your CS160/161 text or notes if you need to.

Consider the grammar $S \rightarrow aSb|aAb; A \rightarrow cAd|B; B \rightarrow aBb|e$, where S is the start symbol.

- (a) Specify the language it generates using setbuilder notation. *Hint: Specify the language generated if B is used as the start symbol, then use this to get a specification of the language generated if A is used as the start symbol, and then use this to get one for the language generated by S .*

Solution: B generates $\{a^n b^n | n \geq 0\}$. Therefore, A generates $\{c^m a^n b^n d^m | n, m \geq 0\}$. Therefore S generates $a^k c^m a^n b^n d^m b^k | n, m \geq 0 \text{ and } k \geq 1\}$.

- (b) Prove that the language it generates is not regular.

Solution: We'll suppose the opposite of what we want to prove and obtain a contradiction.

Suppose it's regular. Then there is a DFA that recognizes the language. Choose a word $a^k c^m a^n b^n d^m b^k$ such that k is larger than the number of states in this DFA. Since we have assumed the language is regular, the pumping lemma must apply. Some of the a 's in the initial segment can be pumped, but then we get a word where the number of a 's at the beginning doesn't match the number of b 's at the end, a contradiction.

- (c) Show that the grammar is ambiguous by giving derivations of $aabb$ that have two different parse trees.

Solution:

$$\begin{aligned} S &\Longrightarrow aAb \Longrightarrow aBb \Longrightarrow aaBbb \Longrightarrow aabb \\ S &\Longrightarrow aSb \Longrightarrow aaAbb \Longrightarrow aaBbb \Longrightarrow aabb. \end{aligned}$$

- (d) Construct an unambiguous grammar for the same language.

Solution: *The problem comes up when no c 's or d 's are generated, which makes it impossible to determine which a 's and b 's were generated directly by S and which were generated by B . We can modify the grammar so that, in this case, we force all the a 's and b 's to be generated by S and none by B . We make A generate B only if it has generated at least one c and d .*

$$\begin{aligned} S &\rightarrow aSb|aAb \\ A &\rightarrow cAd|cBd|e \\ B &\rightarrow aBb|e. \end{aligned}$$

3. Study Examples 3.3.1 and 3.3.3, and Figure 3.5. Following the format of Figure 3.5, give the operation of the automaton of Example 3.3.1 on the strings $aabb$, $aaabb$, and $abab$.

Solution:

$$aabb : (s_0, aabb, e), (s, abb, a), (s, bb, aa), (f, b, a), (f, e, e)$$

The state f is favorable and the stack is empty, so the string is accepted.

$aaabb : (s_0, aaabb, e), (s, aabb, a), (s, abb, aa), (s, bb, aaa), (f, b, aa), (f, e, a)$. We can't apply any transition and the stack isn't empty, so the string is not accepted.

$abab : (s_0, abab, e), (s, bab, a), (f, ab, e)$. We can't apply a transition from f on an empty stack, but we haven't finished reading the string. The string isn't accepted.

4. Come up with context-free grammars for the following languages

- (a) $\{a^n b^n a^m b^m | n, m \geq 0\}$. You must find one that uses only two nonterminals, S and A . *Hint: Note that the language is the concatenation of $\{a^k b^k | k \geq 0\}$ with itself. Come up with a grammar for this language with start symbol A , and use what you've done as part of your final answer, which should have start symbol S .*

Solution:

$$\begin{aligned} S &\rightarrow AA \\ A &\rightarrow aAb|e. \end{aligned}$$

- (b) $\{a^n b^{n+m} b^m | n, m \geq 0\}$. You must find one that uses only three nonterminals, S , A , and B . *Hint: The language is the concatenation of $\{a^i b^i | i \geq 0\}$ and $\{b^j a^j | j \geq 0\}$. Come up with a grammar with start symbol A for the first language, a grammar with start symbol B for the second language, and use what you've done to get a grammar that has start symbol S .*

Solution:

$$\begin{aligned} S &\rightarrow AB \\ A &\rightarrow aAb|e \\ B &\rightarrow bBa|e \end{aligned}$$

- (c) $\{a^n b^m | n, m \geq 0 \text{ and } n \leq 2m\}$. *Hint: Come up with a grammar for $\{a^n b^m | n, m \geq 0 \text{ and } n = 2m\}$ and modify what you've developed.*

Solution: (revised 4/6)

$$\begin{aligned} S &\rightarrow Sb|A \\ A &\rightarrow aaAb|aAb|e \end{aligned}$$

5. Understanding why all regular languages are context free (Proof 1).

We have seen that there exist some context-free languages that are non-regular. We left open the question of whether there are regular languages that aren't context free. We'll now show that every regular language is context-free, which will mean that the regular languages are a **proper** subset of the context-free languages.

Proceed as follows.

- (a) Show that the one-word languages $\{\{c\} | c \in \Sigma\}$ are context-free. Then show that if L_1 and L_2 are context-free languages, then so are $L_1 \cup L_2$ and the concatenation $L_1 L_2$. Then show that if L is a context-free language, then so is L^* .

Explain why, if you succeed in showing this, it will prove that all regular languages are context-free.

Solution: *By definition, every regular language can be constructed by a sequence of these operations. You've proven that none of these operations can generate a non-context-free language. Therefore, every regular language is context-free.*

- (b) Let me get you started in proving the claim by showing that the union of two context-free languages is context-free. If L_1 and L_2 are context-free, then L_1 is generated by a grammar G_1 and L_2 is generated by a grammar G_2 . Rename any nonterminals in G_2 that appear in G_1 so that the two grammars share no nonterminals. Let S_1 be the start symbol of G_1 and S_2 be the start symbol of G_2 . Take the union of the rules in G_1 and G_2 , and create a new start symbol S and two new rules $S \rightarrow S_1 | S_2$. S can generate a word if and only if it is generated by S_1 or by S_2 , in other words, if and only if it is in $L_1 \cup L_2$.

Solution: *The set of rules $\{S \rightarrow c | c \in \Sigma\}$ generate all languages consisting of a one-letter word.*

The rest of the proof is given as the proof of Theorem 3.5.1 on page 92.

6. Understanding why all regular languages are context-free (Proof 2).

Here is a context-free grammar for the DFA of Figure 2.7, where S is the starting symbol:

$$\begin{aligned} S &\rightarrow aQ|bT \\ Q &\rightarrow bR|aS \\ R &\rightarrow aT|bQ \\ T &\rightarrow bS|aR|e \end{aligned}$$

Compare this grammar to the DFA and figure out how I came up with it.

- (a) Show that you understand how to generalize the trick from DFA's to context-free grammars by giving a grammar for the language recognized by the NFA of Figure 2.20.

Solution:

$$S \rightarrow aQ|bR$$

$$Q \rightarrow bS|R$$

$$R \rightarrow aT|T$$

$$T \rightarrow bT|e$$

- (b) Come up with a general algorithm for generating a context-free grammar from an NFA. Let me get you started.

- For each state x , create a nonterminal X .
- For each transition (x, y) on letter c , generate a rule of the form [fill this in yourself.]

Solution: $X \rightarrow cY$

- For each transition (x, y) on e , generate a rule of the form [fill this in yourself.]

Solution: $X \rightarrow Y$

- For each accept state X , generate a rule of the form [fill this in yourself.]

Solution: $X \rightarrow e$.

- If s is the start state make [fill this in] be the starting symbol.

Solution: S .

(A way to prove that this construction works is to show by induction on the length of a word w that $w \in L_x$ if and only if $w \in L_X$. From this it follows that $L_s = L_S$, that is, that the language accepted by the NFA is the same as the one generated by the grammar. If you've come up with the strategy, you've shown that you understand the induction, even if you're still having trouble expressing it.)