

## SOLUTIONS: Homework 7, CS301, McConnell, Spring '09

1. Either use the pumping lemma to show that the following language is not context-free, or show that it is by giving a context-free grammar for it.

$$\{a^i b^j a^i b^j \mid i, j \geq 0\}$$

**Solution:** *From the standpoint of your career, a minor lesson from this problem is how to use the pumping lemma, and the major lesson is how to apply principles of formal logic that are useful all over computer science. For instance, you need them to debug programs. I will post more commentary that what I expect in your solutions to help you understand these principles and get ready for the exam.*

*To prove that it isn't context-free, assume the opposite and obtain a contradiction.*

*Let's teaser apart what the pumping lemma says. If a language is context-free, then there exists an  $n$  such that*

- **For all** words in the language longer than  $n$ ,
- **there exist** substrings  $v$  and  $y$ , at least one of which is nonempty, and such that the number of characters from the beginning of  $v$  to the end of  $y$  is at most  $n$ , and
- **for all**  $k \geq 0$ , you can replace each of  $v$  and  $y$  with  $k$  copies of  $v$  and  $y$ , respectively.

*The proof shows that it never lies about a context-free language. Our contradiction will be to show that the pumping lemma lies about our language.*

*When somebody says **for all**, you can show that they aren't telling the truth by showing a single counterexample. This makes our job easy in this case. When they say **there exists**, we must show that **every possible** counterexample fails to be a counterexample. This makes our job harder.*

- *Since the pumping lemma says that it applies **for all** strings in the language, we get to pick a string, since it suffices to pick a single counterexample to catch it in a lie. Let's pick  $a^n b^n a^n b^n$ , where  $n$  is the value that the pumping lemma claims exists.*
- *Since the lemma says that **there exist**  $v$  and  $y$  with the claimed properties, let's show that there can't exist **any** such  $v$  and  $y$ . There is a huge number of choices for  $v$  and  $y$ , and we must show that **none** of them can work.*

*The string can be pumped if  $v$  is in the first block of  $a$ 's and  $y$  is in the second block of  $a$ 's, and  $|v| = |y|$ . Similarly, it can be pumped if  $v$  is in the first block of  $b$ 's and  $y$  is in the second block of  $b$ 's and  $|v| = |y|$ . It seems like we have failed.*

*Fortunately, the pumping lemma helps us out by saying that the number of characters from the beginning of  $v$  to the end of  $y$  is at most  $n$ . (The pumping lemma was designed to make it as easy as possible to catch in a lie. Even though it is a statement about context-free languages, its main purpose is to allow us to show that a language is not regular, by tripping it up.)*

*Because each block of  $a$ 's and  $b$ 's has length  $n$ , this constraint means that  $v$  and  $y$  are confined to two consecutive blocks. One of these consists of  $a$ 's and the other*

*b's. There is no way to pump one or two consecutive blocks without causing an imbalance with the number of a's and/or b's in the other two blocks. There can be no choice of  $v$  and  $y$  that can be pumped.*

*The pumping lemma lies about our string  $a^n b^n a^n b^n$ . Since it never lies about any string in a context-free language, our language can't be context-free.*

2. Either use the pumping lemma to show that the following language is not context-free, or give a grammar for it:

$\{a^i b a^i b \mid i \geq 1\}$ .

**Solution:**  $S \rightarrow Ab; A \rightarrow aAa|aba$

3. Use the pumping lemma to show that the following language is not context-free:

$\{a^i \mid i \text{ is a power of two that is greater than or equal to } 1\}$ .

**Solution:** *Suppose it's context-free. Then the pumping lemma applies. Let  $n$  be as in the pumping lemma. Let  $j$  be the smallest power of two greater than  $n$ . Let's trip it up with the word  $a^j$ . (We can choose this because it says that it applies **for all** words in the language.) Because it says that **for all**  $k \geq 0$ , we can replace  $v$  and  $y$  with  $k$  copies, we get to choose  $k$ . Let's choose  $k = 2$ . Because it says that at least one of  $|v|$  and  $|y|$  is nonempty and the number of characters between the beginning of  $v$  and the end of  $y$  is at most  $n$ ,  $1 \leq |v| + |y| \leq n$ . Pumping in an extra copy of  $v$  and  $y$  increases our string to length at most  $j + n$ , which is less than  $2j$  since we chose  $j > n$ .*

*The new word is not long enough to be the next larger string in the language, no matter which  $v$  and  $y$  we choose. The pumping lemma is not telling the truth about our string, so it's not context-free.*

4. Show that the intersection of two context-free languages is not necessarily context-free by giving context-free grammars for the following languages, and then showing that their intersection is not context-free:  $\{a^n b^m c^k \mid n = m\}$  and  $\{a^n b^m c^k \mid m = k\}$ .

**Solution:** The first language is the concatenation of  $D = \{a^n b^n \mid n \geq 0\}$  and  $C = c^*$ .  $S_1 \rightarrow DC; D \rightarrow aDb|e; C \rightarrow cC|e;$

The second language is the concatenation of  $A = a^*$  and  $E = \{b^m c^m \mid m \geq 0\}$ .  $S_2 = AE; A = aA|e; E = bEc|e.$

Their intersection is  $\{a^n b^n c^n \mid n \geq 0\}$ , which we've shown is not context-free.

The claim that the "intersection of two context-free languages is context free" is a shorthand for the more formal statement, "**For all** context-free languages  $L_1$  and  $L_2$ ,  $L_1 \cap L_2$  is context-free." We've found a single counterexample, so the claim is false.

The claim that "**There exist** two context-free languages  $L_1$  and  $L_2$  such that  $L_1 \cap L_2$  is context-free" is a true statement. It's true if there is a single example, rather than a single counterexample. Any two regular languages (which are context-free) serve as an example. (Why?)

5. Use the construction of Theorem 3.5.1 to find context-free grammars for the following languages:

- (a)  $\{a^n b^n a^m b^m \mid n \geq 1, m \geq 0\}$ . *Hint: this is the concatenation of the language  $\{a^n b^n \mid n \geq 1\}$  and the language  $\{a^m b^m \mid m \geq 0\}$ .*

**Solution:**  $S \rightarrow S_1 S_2; S_1 \rightarrow a S_1 b | ab; S_2 \rightarrow a S_2 b | \epsilon$

- (b)  $\{a^n b^m c^k \mid n, m, k \geq 1 \text{ and } (n = m \text{ or } m = k)\}$  *Hint: this is the union of two languages, one where  $n = m$  and the other where  $m = k$ . The first of these is the concatenation of  $\{a^n b^n \mid n \geq 1\}$  and the language  $cc^*$ .*

**Solution:**  $S \rightarrow S_1 | S_2;$

$S_1 \rightarrow DC; D \rightarrow aDb | ab; C \rightarrow cC | c;$

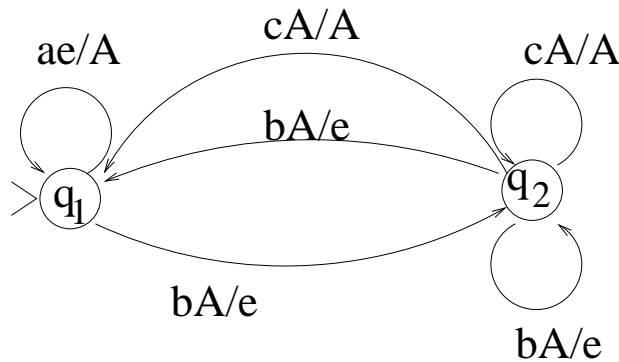
$S_2 \rightarrow AE; A \rightarrow aA | a; E \rightarrow bEc | bc;$

- (c)  $\{a^n b^n \mid n \geq 1\}^*$ . (Notice the Kleene star.)

**Solution:**  $S_1 \rightarrow e | S_1 S_1 | A; A \rightarrow aAb | ab$

6. In my posting on constructing a context-free grammar from a PDA, study the relationship between the example PDA near the end and the rules I generate for it. Make guesses about the general strategy. Refer to earlier in the handout for confirmation of your guesses, or for an explanation.

Consider the following PDA:



- (a) Each nonterminal that we need to generate for a PDA is of the form  $[q, e, r]$  or  $[q, B, r]$ , where  $B$  is any stack symbol, and  $q$  and  $r$  are any states and needn't be distinct. (In the case of the above PDA, the only stack symbol is  $A$ .)

Use a cartesian product to generate a list of the possible possible nonterminals that can appear in the grammar for it. Use the notation the handout uses for the nonterminals. If you list some nonterminals that aren't actually used, you won't be counted off, as long as you list all the ones that are used.

**Solution:**  $S, [q_1, e, q_1], [q_1, e, q_2], [q_2, e, q_1], [q_2, e, q_2], [q_1, A, q_1], [q_1, A, q_2], [q_2, A, q_1], [q_2, A, q_2],$

- (b) Some of the rules say a successful computation corresponds to a walk that starts in  $q_1$  with an empty stack and ends in any state with an empty stack. Write these rules, using  $S$  as the start symbol.

**Solution:**  $S \rightarrow [q_1, e, q_1] \mid [q_1, e, q_2].$

- (c) One of the rules generates the language of all strings that can be consumed during a successful walk that begins by taking the loop that reads  $a$  at  $q_1$ , starting with an empty stack, and ends at  $q_1$  with an empty stack. Write this rule, using our notation for the nonterminals.

**Solution:**  $[q_1, e, q_1] \rightarrow a[q_1, A, q_1].$

- (d) One of the rules generates the language of all strings that can be consumed during a successful walk that begins by taking the loop that reads  $a$  at  $q_1$ , starting with an empty stack, and ends at  $q_2$  with an empty stack. Write this rule, using our notation for the nonterminals.

**Solution:**  $[q_1, e, q_2] \rightarrow a[q_1, A, q_2]$ .

- (e) One of the rules reflects the fact that you can get from  $q_1$  to  $q_1$  starting with an empty stack and ending with an empty stack by taking no action. Write this rule.

**Solution:**  $[q_1, e, q_1] \rightarrow e$ .

- (f) One of the rules generates the language of all strings that can be consumed during a successful walk that begins at  $q_2$  with just an  $A$  on the stack, ends at  $q_2$  with an empty stack, and begins with the transition from  $q_2$  to  $q_1$  that reads  $b$ .

**Solution:**  $[q_2 A q_2] \rightarrow b[q_1 e q_2]$ .

- (g) Some of the rules generate the language of all strings that can be consumed during a successful walk that begins at  $q_1$  with a single  $A$  on the stack, and ends at  $q_2$  with an empty stack. Write these rules, using our notation.

**Solution:**  $[q_1, A, q_2] \rightarrow a[q_1, A, q_1][q_1, A, q_2] \mid a[q_1, A, q_2][q_2, A, q_2]$ .

7. Here's the gist of section 3.8, with a few modifications to make it easier. Recall that the *configuration* of a PDA consists of its current state, the current string of letters on its stack, and the string of remaining letters to read. A *move* is an action that changes the configuration. A *deterministic* PDA is a one-at-a-time PDA that starts out with a special symbol,  $Z$ , on the bottom of the stack, has at most one move that it can take for any configuration, and accepts on an empty stack. Which move it can take depends on the next input letter and/or the symbol at the top of the stack.

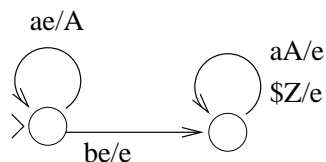
If it finds itself in a configuration that doesn't allow it any moves, it halts. It accepts if it halts on an empty stack with nothing left on the input.

Deterministic PDA's (DPDA's) are the PDA's of interest in the theory of compiler design.

A problem arises when some string  $x$  in the language it is supposed to accept is a prefix of another string,  $xy$ , that is also in the language. When it has read  $x$ , how does it know whether it's supposed to halt? If there are no more characters, trying to read a character to find this out is not an allowed move.

To avoid this, we append a special character,  $\$$ , to the end of each string placed on the input. This makes sure that no string that can be placed on the input can be a prefix of any other. It accepts  $w$  if and only if it halts with an empty stack after reading  $w\$$ .

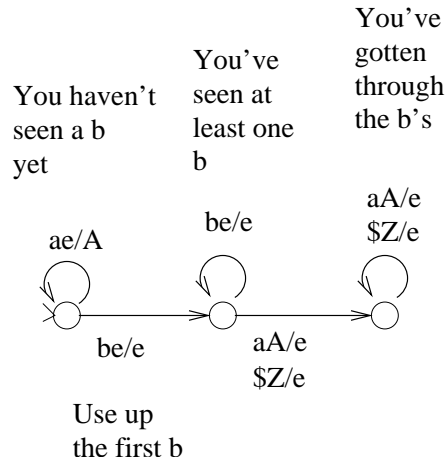
Here's an example of a DPDA for the language  $\{a^n b a^n \mid n \geq 0\}$ :



The end of string symbol, \$, and the bottom of stack symbol, Z, help to keep any words not in the language from being accepted, while allowing at most one transition at each state for any possible configuration. Whenever a string is not in the language, it finds itself with no allowed move even though its stack is nonempty or there are characters left to read on the input.

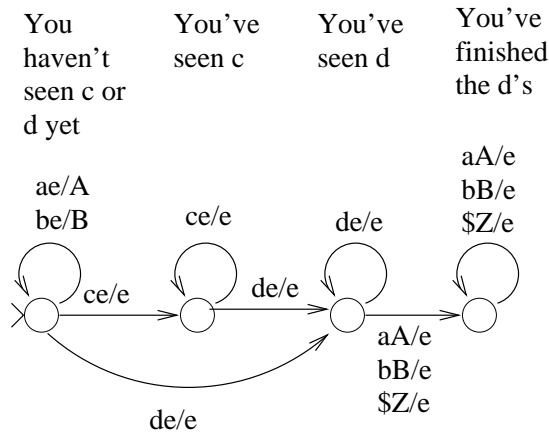
- (a) Modify this PDA to get a DPDA for the language  $\{a^n b^m a^n \mid n \geq 0 \text{ and } m \geq 1\}$ .

**Solution:**

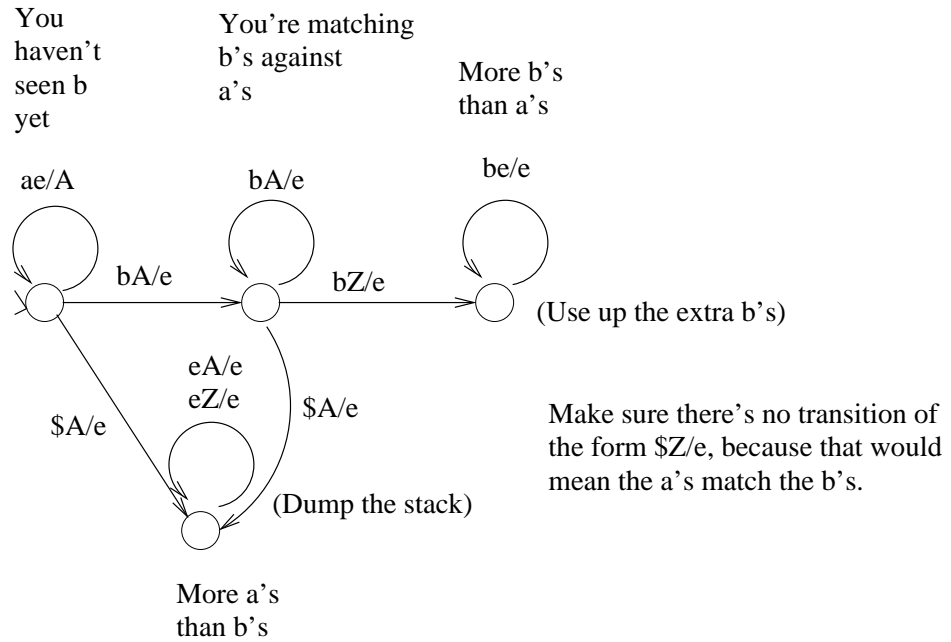


- (b) Draw one for the language  $\{wc^k d^j w^R \mid w \in \{a, b\}^* \text{ and } k \geq 0 \text{ and } j \geq 1\}$  (Recall that  $w^R$  is the reverse of  $w$ .)

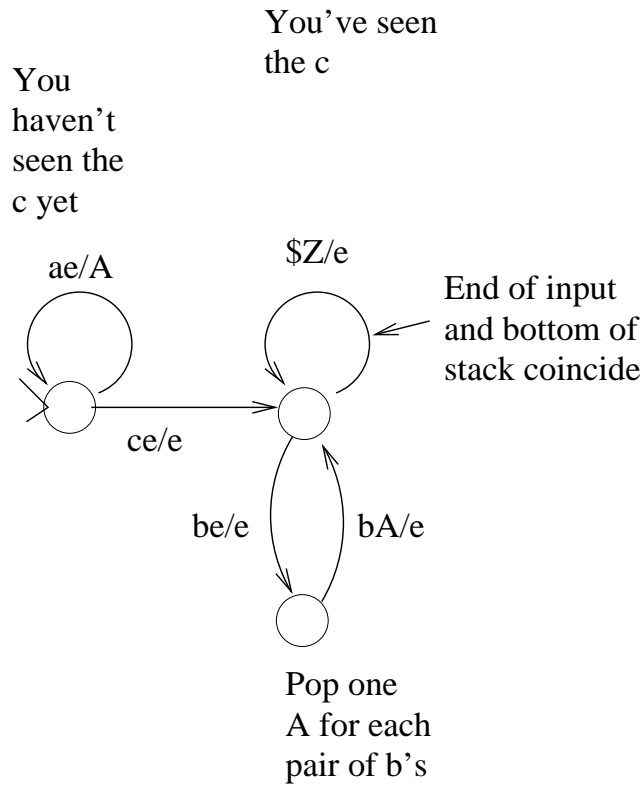
**Solution:**



- (c) Draw one for  $\{a^m b^n \mid m, n \geq 0 \text{ and } m \neq n\}$ .



(d) Draw one for  $\{a^m cb^{2m} \mid m \geq 0\}$ .



8. A *deterministic context-free language* is one that's accepted by a deterministic PDA. Given what we know about NFA's and DFA's, you may suspect that the deterministic context-free languages are just the context-free languages. Let's see if this is so.

A fact that we will not prove is that the complement of every deterministic context-free language is a deterministic context-free language. The trick is based on the usual

strategy: show how to convert a deterministic PDA for the language into one for the complement. It's more complicated than the way we turned a DFA for a regular language into one for the complement, however. One reason for this is that there is nothing in the definition of a DPDA that says that it can't get into an infinite loop or otherwise fail to halt, just as there is nothing in the definition of Java that says a Java program has to halt. (A Java program is also deterministic.) A string that a DPDA doesn't halt on is not in the language it accepts. Turning a DPDA into one for the complement has to deal with how to make the new DPDA accept these strings.

Now look at Theorem 3.8.2 and its proof.

The proof claims that  $\{a^n b^m c^k | n \neq m \text{ or } m \neq k\}$  is "obviously context-free." Show this by giving a grammar for this language. *Hint: refer to the tricks of problem 5.*

**Solution:** *This is the union of the languages  $L_1 = \{a^n b^m c^k | n < m\}$ ,  $L_2 = \{a^n b^m c^k | n > m\}$ ,  $L_3 = \{a^n b^m c^k | m < k\}$  and  $L_4 = \{a^n b^m c^k | m > k\}$ .*

*Here are  $a^*$ ,  $b^*$ ,  $c^*$ :*

$A \rightarrow aA|e; B \rightarrow bB|e; C \rightarrow cC|e.$

*Here are  $\{a^n b^n | n \geq 0\}$  and  $\{b^n c^n | n \geq 0\}$   $D_1 \rightarrow aD_1 b|e; D_2 \rightarrow bD_2 c|e.$*

*Here is our union of four languages:*

$S \rightarrow D_1 b B C; b$ 's exceed  $a$ 's, can be followed by some  $c$ 's

$S \rightarrow a A D_1 C; a$ 's exceed  $b$ 's, can be followed by some  $c$ 's

$S \rightarrow A D_2 c C; c$ 's exceed  $b$ 's, can be preceded by some  $a$ 's

$S \rightarrow A b B D_2; b$ 's exceed  $c$ 's, can be preceded by some  $a$ 's