

# CS 273, Fall 2007

## Exam 2 Solutions

### Problem 1: Short Answer (12 points, about 10 min)

The answers to these problems should be short and not complicated. For some, “yes” or “no” is sufficient.

(a) Are context-free languages closed under intersection?

**Solution:** No.

(b) Let  $\Sigma = \{a, b\}$ . Is the language  $\{w\#x\#y \mid w, x, y \in \Sigma^* \text{ and } |w| = |x| = |y|\}$  context-free?

**Solution:** No. This language requires checking that three numbers are all equal. If you intersect it with  $a^*\#b^*\#a^*$ , and then get rid of the  $\#$  characters with a homomorphism, you get  $a^n b^n a^n$ . We can use the pumping lemma to show that  $a^n b^n a^n$  is not context-free, which

(c) Suppose that language  $L_1$  is regular and  $L_2$  is not regular. Can we safely conclude that  $L_1 \cup L_2$  is not regular?

**Solution:** No. This isn't safe reasoning. Suppose that  $L_1 = \Sigma^*$ . Then  $L_1 \cup L_2$  will be regular no matter what  $L_2$  is.

(d) Let  $G$  be a grammar and  $w$  a string. Suppose that  $w$  has more than one derivation. Does this imply that  $G$  is ambiguous? Why or why not?

**Solution:** No. A grammar is ambiguous if some string has two parse trees or, equivalently, two leftmost derivations. But there could be many derivations (not all leftmost) corresponding to a single parse tree.

(e) Is the language  $\{a^i b^j \mid i, j \geq 0\}$  context-free? Why or why not?

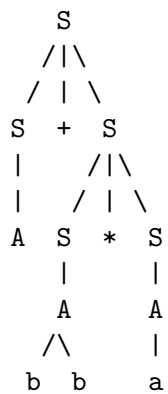
**Solution:** Yes. It's just  $a^* b^*$  which is regular and, thus, context-free.

(f) Let  $G$  be a grammar with start symbol  $S$  and the following rules. Give a parse tree for the string  $a + bb * a$

$$S \rightarrow S + S \mid S * S \mid A$$

$$A \rightarrow a \mid bb$$

**Solution:**



**Problem 2: Grammar design (8 points, about 10 min)**

Let  $J = \{ww^R \mid w \in \{a,b\}^*, |w| \geq 1\}$ , where  $w^R$  is the reversal of the string  $w$ . Notice that  $J$  does not contain the empty string.

Define the language  $L$  by  $L = \{a^n \# x_1 \# x_2 \# \dots \# x_n \mid n \geq 1, x_i \in J \text{ for all } i\}$ .

That is, any string in  $L$  starts with an initial  $a^n$ , which is a counter telling you how many items from  $J$  to expect in the rest of the string. For example,  $a\#abba$  and  $aa\#babbab\#bb$  are both in  $L$ . But none of the following are in  $L$ :  $\epsilon$ ,  $a\#abba\#bbbb$ ,  $a\#abab$ ,  $a\#aba$ .

(a) Give a context-free grammar whose language is  $J$ .

**Solution:** The grammar has the start symbol  $T$  and rules:

$$T \rightarrow aTa \mid bTb \mid aa \mid bb$$

Common mistakes were including  $\epsilon$  on the righthand side (which makes  $J$  include the empty string) or  $a$  and  $b$  (which makes  $J$  include odd-length strings).

(b) Give a context-free grammar whose language is  $L$ .

**Solution:** The grammar has start symbol  $S$ . It contains the rules for  $T$  given in part (a) plus:

$$S \rightarrow aS\#T \mid a\#T$$

It's important that a single recursive step generates an  $a$  and a  $T$  at the same time, so the number of  $a$ 's will match the number of  $T$ 's.

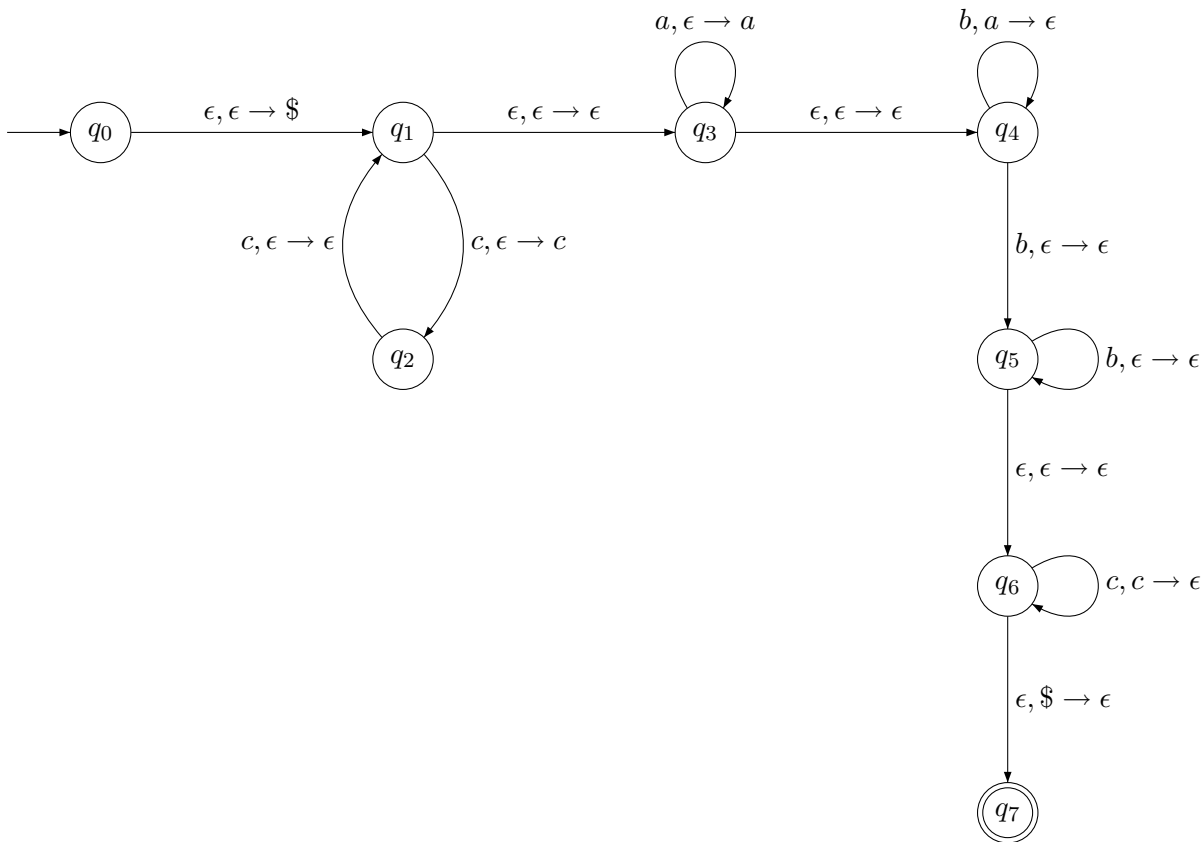
These are succinct solutions. Many people got full credit for slightly longer grammars, e.g. with some auxiliary variables and/or with their answer to (a) copied into their answer for  $B$ .

**Problem 3: PDA design (8 points, about 12 min)**

Let  $L = \{c^{2n}a^ib^jc^n \mid n \geq 0, j > i \geq 0\}$ . Notice that  $n$  and  $i$  can be zero, but  $j$  must be greater than or equal to 1.

Give the state diagram for a PDA whose language is  $L$ . Include brief comments explaining the design of your PDA, to help us understand how it works.

**Solution:**



The first and last transitions put an marker on the stack and check that it's still there, ensuring that the numbers of  $c$ 's match. States  $q_1$  and  $q_2$  push a single  $c$  onto the stack for each pair of input  $c$ 's. State  $q_3$  pushes incoming  $a$ 's onto the stack. State  $q_4$  compares incoming  $b$ 's to the  $a$ 's on the stack. State  $q_5$  reads any extra  $b$ 's and the transition from  $q_4$  into  $q_5$  ensures that at least one extra  $b$  has been read. Finally, state  $q_6$  compares incoming  $c$ 's to the  $c$ 's on the stack.

#### Problem 4: Short Answer II (8 points, about 10 min)

The answers to these problems should be short and not complicated.

(a) We know that the language  $B = \{a^n b^n c^n \mid n \geq 0\}$  is not context-free. Let  $L$  be the language  $\{a^n b c^n b d^{n-1} \mid n \geq 1\}$ . Prove that  $L$  is not context-free using closure properties and the fact that  $B$  is not context-free.

**Solution:**

Suppose that  $L$  were context-free. Let  $h$  be a homomorphism that maps  $a$  to itself,  $b$  to  $\epsilon$ ,  $c$  to  $b$ , and  $d$  to  $c$ .  $h(L)$  must be context-free, because CFLs are closed under homomorphism. And  $h(L) = \{a^n b^n c^{n-1} \mid n \geq 1\}$ .

Now, let  $L' = (h(L) \circ \{c\})$ .  $L'$  must be context-free, because CFLs are closed under concatenation.  $L'$  is the set  $\{a^n b^n c^n \mid n \geq 1\}$ , which is almost what we want, but missing the empty string. So let  $L'' = L' \cup \{\epsilon\}$ . This is context-free since  $L'$  is context free and CFLs are closed under union.

But  $L''$  is just  $B$ , which we know not to be context-free. So we have a contradiction. so  $L$  must not have been context-free.

(b) Prove that context-free languages are closed under the star (\*) operation. Specifically, given a grammar  $G = (V, \Sigma, R, S)$ , construct a grammar  $G'$  such that  $L(G') = (L(G))^*$ .

**Solution:** Let  $X$  be some new variable not in  $V$ . Then  $G' = (V \cup \{X\}, \Sigma, R', X)$  where  $R'$  contains all the rules in  $R$  plus the rule  $X \rightarrow SX \mid \epsilon$ .

We have never been very precise about the format of the rule set  $R$ , so I accepted any reasonably clear notation for expressing the idea of adding the extra rule to  $R$ .

A new start symbol is definitely needed, to avoid generating extra strings beyond what's in the star of the language. For example, if your original grammar  $G$  is  $S \rightarrow aSb \mid \epsilon$ , the strings in  $L(G)^*$  should contain  $b^i$  substrings right next to the corresponding  $a^i$  substrings. If you try to make the star of  $L(G)$  by adding the rule  $S \rightarrow SS$ , you can get derivations like

$$S \rightarrow aSb \rightarrow aSSb \rightarrow aabSb \rightarrow aababb$$

### Problem 5: Pumping Lemma (8 points, about 10 min)

Let  $\Sigma = \{a, b\}$  and let  $L$  be the language  $\{ww^R \mid w \in \Sigma^*\}$ , where  $w^R$  is the reversed version of the string  $w$ . Use the pumping lemma to show that  $L$  is not regular.

**Solution:** Suppose that  $L$  were regular. Let  $p$  be its pumping length, as guaranteed by the pumping lemma.

Consider the string  $w = a^p b b a^p$ .  $w$  is clearly in  $L$  and its length is  $\geq p$ . So, by the pumping lemma, there exist strings  $x, y, z$  such that

- a)  $w = xyz$ ,
- b)  $|xy| \leq p$ ,
- c)  $|y| \geq 1$ , and
- d)  $xy^i z \in L$  for all  $i \geq 0$ .

Since the first  $p$  characters of  $w$  are  $a$ 's and  $|xy| \leq p$ ,  $y$  and  $x$  contain only  $a$ 's. Then  $xy^2z = a^{p+|y|} b b a^p$ . Since  $|y| \geq 1$ ,  $p + |y| \neq p$ . So the number of  $a$ 's in the two halves of the strings don't match and, thus,  $xy^2z$  can't be in  $L$ . This contradicts condition (d) of the pumping lemma.

Since we have found a contradiction,  $L$  must not have been regular.

**Problem 6: Formal notation (8 points, about 10 min)**

Let  $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$  be a PDA and  $N = (R, \Sigma, \gamma, r_0, G)$  be a DFA. We can use the product construction to build a PDA  $M'$  which recognizes  $L(M) \cap L(N)$ .

Specifically,  $M' = (Q \times R, \Sigma, \Gamma, \delta', (q_0, r_0), F')$

(a)  $F' =$

**Solution:**  $F \times G$

(b) First, let's consider the transitions where the PDA  $M$  reads an input character. If  $c \in \Sigma$ ,  $s \in \Gamma_\epsilon$ ,  $q \in Q$ , and  $r \in R$ , then  $\delta'((q, r), c, s) =$

**Solution:**  $\{((p, \gamma(r, c)), t) \mid (p, t) \in \delta(q, c, s)\}$

The output is a set, because we are constructing a PDA. The elements of the set need to be pairs: (state, stack symbol). The "state" is a state of the new PDA, which is a pair of states, one from each of the two input automata. Notice that the output of  $\delta$  is a set, but the output of  $\gamma$  is a single value.

Mistakes on this part were very, very common. The answer  $\delta(q, c, s) \times \gamma(r, c)$  is on the right track, but the elements of the output set are in the wrong format. It has the first state grouped with the new stack symbol  $((q', s'), r')$  rather than the two state grouped together  $((q', r'), s')$ . Also the output of  $\gamma$  is a single state (not a set).

(c) We also need to handle the transitions where the PDA  $M$  changes state without reading any input characters. Suppose  $s \in \Gamma_\epsilon$ ,  $q \in Q$ , and  $r \in R$ , then  $\delta'((q, r), \epsilon, s) =$

**Solution:**  $\{((p, r), t) \mid (p, t) \in \delta(q, \epsilon, s)\}$ .

Because we aren't reading any input characters, the DFA can't change state. So its state has to remain  $r$ .

Answers to (c) were graded based on how well you had adapted your answer to (b), as long as your answer to (b) was coherent enough.

**Problem 7: Induction (8 points, about 10 min)**

Let  $G$  be a grammar with the following rules:

$$S \rightarrow SS \mid bb \mid aSa$$

Use induction to prove that every string in  $L(G)$  has even length. Your induction variable should be the number of steps in a derivation (or something very similar such as the number of nodes in a parse tree). Your induction variable should **not** be the length of the string.

**Solution:** Proof by induction on the length of a leftmost derivation for the string.

Base case: if a string of terminals  $w$  is derived in a single step, that means that the string must be  $bb$ . This string clearly has even length.

Induction: Suppose that the claim is true for all strings that can be derived in  $\leq k$  steps. Let  $w$  be a string whose derivation is  $k + 1$  steps long. Consider the first step in the derivation.

Case 1: the first step uses the rule  $S \rightarrow aSa$ . Then the other  $k$  steps of the derivation must expand the new  $S$  into some string  $x$ . And then  $w = axa$ . Because  $x$  was derived using  $\leq k$  steps, the induction hypothesis tells us that  $|x|$  is even. Therefore,  $|w| = |x| + 2$  must be even.

Case 2: the first step uses the rule  $S \rightarrow SS$ . Then the rest of the derivation expands the first  $S$  into some string  $x$  and the second  $S$  into some string  $y$ , where  $w = xy$ . The derivation for  $x$  must be  $\leq k$  steps, so the induction hypothesis implies that  $|x|$  is even. Similarly,  $|y|$  is even. This means that  $|w| = |x| + |y|$  must also be even.