

CS 273, Spring 2008

Exam 2 Solutions

Problem 1: Short Answer (12 points)

Answer “yes” or “no” to the following questions. No explanations are required. (All the strings in this problem use the same, fixed alphabet.)

- (a) Is the language $\{ww^Rw \mid w \in \{a, b\}^*\}$ a context-free language?

Solution:

No. A context-free language can only generate matched pairs, not matched triples.

- (b) If L is a non-regular language over Σ^* , and h is a homomorphism, then $h(L)$ must also be non-regular. Is this statement correct?

Solution:

No. Suppose that h mapped all characters to the empty string. Then $h(L)$ would be regular no matter what L is.

- (c) Suppose all the words in language L are no more than 1024 characters long. Then L must be regular. Is this statement correct?

Solution:

Yes. There's only a finite set of strings with ≤ 1024 characters. So L is finite and therefore regular.

- (d) If L_1 and L_2 be two languages, the **xor** of the two languages is

$$L_1 \oplus L_2 = (L_1 \setminus L_2) \cup (L_2 \setminus L_1) = \left\{ w \mid \begin{array}{l} w \in L_1 \text{ and } w \notin L_2 \\ \text{or} \\ w \notin L_1 \text{ and } w \in L_2 \end{array} \right\}.$$

If L_1 and L_2 are both context-free, then $L_1 \oplus L_2$ must also be context-free. Is this statement correct?

Solution:

No. Subtracting one context-free language from another does not necessarily result in a context-free language.

- (e) If L is a language, its prefix language is

$$P(L) = \left\{ w \mid \text{there exists } x \text{ s.t. } wx \in L \right\}.$$

If L is context-free, then the language $P(L)$ is context free. Is this statement correct?

Solution:

Yes. You can build a PDA for $P(L)$ which guesses the right contents for x .

- (f) A PDA that is allowed to enter the accept state only if the stack is empty is a **strict PDA**. There are context-free languages for which no strict PDA exists. Is this statement correct?

Solution:

No. If you have a PDA recognizing a language L , you can create a modified PDA which recognizes L and empties its stack before accepting.

Problem 2: Grammar design (8 points)

Let $\Sigma = \{a, b\}$.

Let $J = \left\{ w_1 \# w_2 \# \dots \# w_{n-1} \# w_n \mid n \geq 2, w_i \in \Sigma^* \text{ for all } i, \text{ and for some } i, |w_i| = |w_{i+1}| \right\}$

In other words, an element of J is a list of at least two strings of a's and b's, separated by #'s. In the list, some pair of adjacent strings have the same length. E.g. J contains $b\#aa\#bbb\#aba$ but not $a\#bbb\#b$.

Give a context-free grammar whose language is J . Be sure to indicate what its start symbol is.

Solution:

Here's a grammar for J with start symbol S . The variable X generates an adjacent pair of strings with matching lengths. The variable Y generates an arbitrary string from Σ^* .

$S \rightarrow Y\#S \mid S\#Y \mid X$

$Y \rightarrow aY \mid bY \mid \epsilon$

$X \rightarrow aXa \mid aXb \mid bXa \mid bXb \mid \#$

Problem 3: PDA design (8 points)

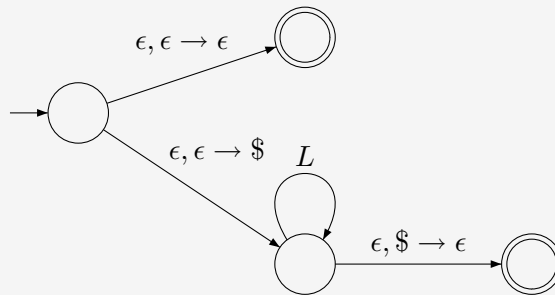
Let

$$J = \left\{ w \in \{a, b\}^* \mid \begin{array}{l} w \text{ contains only a's} \\ \text{or} \\ w \text{ has an equal number of a's and b's} \end{array} \right\}.$$

For example, J contains ϵ , aaa , and $aabbba$. But $abbba$ is not in J .

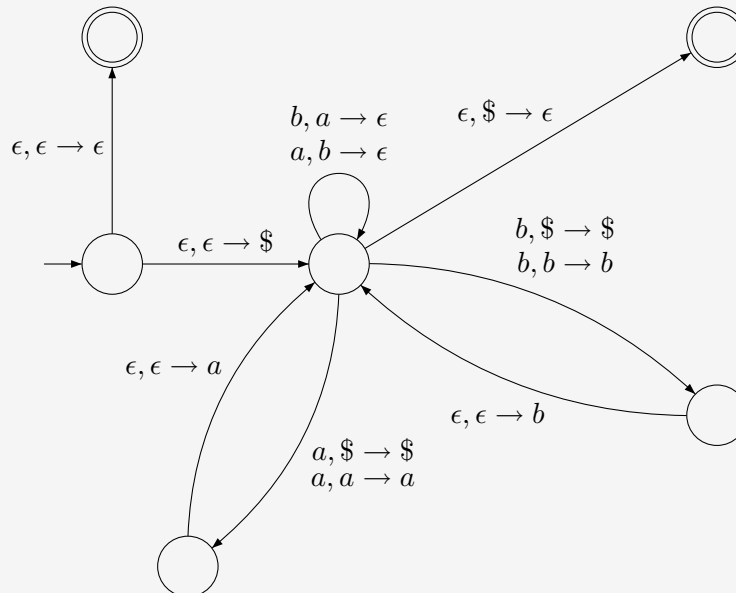
Give the state diagram for a PDA whose language is J . Include brief comments explaining the design of your PDA, to help us understand how it works.

Solution:



The upper branch accepts strings containing only a's. The lower branch accepts the others, using the stack to keep characters that aren't yet matched. The arc labelled L contains the following transitions: $(b, a \rightarrow \epsilon)$, $(a, b \rightarrow \epsilon)$, $(a, \$ \rightarrow a\$)$, $(b, \$ \rightarrow b\$)$, $(a, a \rightarrow aa)$, $(b, b \rightarrow bb)$. This solution uses the multiple-push feature (not forbidden by the instructions).

Here's a variation that doesn't use multiple pushes:



Problem 4: Short Answer II (8 points)

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

- (a) Suppose we know that the language $B = \{a^n b^n c^n \mid n \geq 2\}$ is not context-free. Let L be the language $L = \{a^n b c^n b d^n \mid n \geq 0\}$. 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 the homomorphism that maps a to itself, b to ϵ , c to b , and d to c . Then $h(L) = \{a^n b^n c^n \mid n \geq 0\}$ must be context-free because context-free languages are closed under homomorphisms.

Let K be the set of all strings of length ≥ 2 . K is regular. (It's easy to show a DFA recognizing K but also we've done enough with regular languages that this should be obvious.) Then $h(L) \cap K$ must be context-free, by closure under intersection with a regular language.

But $h(L) \cap K$ is just B , which we know not to be regular. So we have a contradiction. Therefore, L must not have been regular.

A more intuitive way to do the second step is to subtract a (small finite) regular language from $h(L)$. Technically, we haven't shown that context-free languages are closed under subtraction of a regular language, but it's a very easy exercise to show that this is true. So this line of reasoning is also ok.

- (b) Define what it means for a grammar G to be in Chomsky Normal Form (CNF).

A grammar is in Chomsky normal form every rule has one of the following forms:

Solution:

- $A \rightarrow BC$ where B and C are variables that aren't the start symbol.
- $A \rightarrow a$ where a is a terminal.
- $S \rightarrow \epsilon$ where S is the start symbol.

Problem 5: Pumping Lemma (8 points)

Suppose $\Sigma = \{a, b\}$ and let $L = \{a^n w \mid w \in \Sigma^* \text{ and } |w| = n\}$. That is, L contains even-length strings whose first half contains only a's. Prove that L is not regular by filling in the missing parts of the following pumping lemma proof.

Suppose that L were regular. Let p be the constant given by the pumping lemma.

Consider the string $w_p =$

Solution:

$w_p = a^p b^p$. $w_p = a^p b a^{p-1}$ will also work. However, it doesn't work to use $w_p = a^p w$, without constraining what's in w . What if w were a^p ?

Because $w_p \in L$ and $|w_p| \geq p$, there must exist strings x , y , and z such that $w_p = xyz$, $|xy| \leq p$, $|y| > 0$, and $xy^i z \in L$ for every $i \geq 0$.

Solution:

The trick here to pump down. Since $xy^i z \in L$ for every $i \geq 0$, the string xz should be in L . Since $|xy| \leq p$, y contains only a's. So if $|y| = k$, then xz has the form $a^{p-k} b^p$. Since $|y| > 0$, $k \geq 1$. So $p - k < p$. This means that, even if the length of xz happens to be even, its first half will contain some b's. Therefore xz cannot be in L .

[It's not sufficient to state that the number of a's and b's don't match. The language L doesn't actually require them to match: a string in L could contain more a's than b's.]

[Also, it doesn't work to pump up, e.g. consider the string $xyyz$. If you add more a's to the start of the string, the variable w can just absorb them. Your adversary gets to pick y , so he can ensure that it's even length.

Since

Solution:

wz

is *not* in L , we have a contradiction. Therefore, L must not have been regular.

Problem 6: Formal notation (8 points)

Let $M = (Q, \Sigma, \Gamma, \delta, q_0, F)$ be a PDA recognizing the language L , here L is defined over an alphabet Σ . Define the language L' to be

$$L' = \{x\#y \mid x, y \in L\}.$$

Describe how to construct a PDA recognizing L' . (You can safely assume that $\#$ is not in the alphabets Σ and Γ used by M and L .)

- (a) Describe the ideas behind your construction in words and/or pictures.

Solution:

We will make two copies of M , connected together by a transition that reads the $\#$. Having two copies is critical, so the PDA can check that the input contains exactly two strings from L .

- (b) When you read the $\#$ from the input, or shortly after you read it, you will need to do something about whatever is left on the stack from reading the x part of the input string. Do you need to push anything onto the stack or pop anything off? If so, what? Namely, describe what your PDA does upon reading $\#$.

Solution:

You have to do something with the stack, otherwise the second copy of M may get confused when it processes y . One option is to empty the stack when you read $\#$. This requires pushing a stack-bottom symbol before running the first copy of M , because we have no idea whether M used one. You'll need a couple extra states, one at the start and one between the two copies of M .

We are using a second option, which is slightly simpler: push $\#$ onto the stack. Because M doesn't use $\#$, this will force the PDA to reject if the second copy of M tries to read too deep into the stack.

- (c) Give the details of your construction in formal notation. That is, for the new PDA recognizing L' , specify the set of states, the initial and final states, the stack alphabet, the details of the transition function. The input alphabet for the new machine will be $\Sigma \cup \{\#\}$.

Solution:

First, let $M'(Q', \Sigma, \Gamma, \delta', q'_0, F')$ be a copy of M . Then the new PDA will be $N = (Q \cup Q', \Sigma \cup \{\#\}, \Gamma \cup \{\#\}, \gamma, q_0, F')$. The transition function γ is defined by:

$$\gamma(q, a, t) = \delta(q, a, t) \text{ if } q \in Q, a \in \Sigma, t \in \Gamma$$

$$\gamma(q, a, t) = \delta'(q, a, t) \text{ if } q \in Q', a \in \Sigma, t \in \Gamma$$

$$\gamma(q, \#, \epsilon) = \{(q'_0, \#)\} \text{ if } q \in F$$

$$\gamma(q, a, t) = \emptyset \text{ for all other inputs}$$

Problem 7: Induction (8 points)

Let $\Sigma = \{\mathbf{a}, \mathbf{b}\}$. Given any string w in Σ^* , let $A(w)$ be the number of \mathbf{a} 's in w and $B(w)$ be the number of \mathbf{b} 's in w .

Suppose that grammar G has the following rules:

$$S \rightarrow \mathbf{aSb} \mid \mathbf{aSS} \mid \mathbf{ab},$$

where S is the start symbol. Use induction on the derivation length to prove that $A(w) \geq B(w)$ for any string w in $L(G)$.

Solution:

Base: If w is derived using only a one-step derivation, then $w = \mathbf{ab}$, which contains at least as many \mathbf{a} 's as \mathbf{b} 's.

Induction: Suppose that $A(x) \geq B(x)$ for any string x in $L(G)$ which has a derivation of length $< k$. Let w be a string in $L(G)$ with a derivation of length k . Consider the first step in this derivation.

Case 1: The first step in the derivation is $S \rightarrow \mathbf{ab}$. Then we are back in the base case. [Or, you could reasonably have left out this case.]

Case 2: The first step in the derivation is $S \rightarrow \mathbf{aSb}$. Then $w = \mathbf{axb}$, where x is derived from S using $k - 1$ steps. By the inductive hypothesis, $A(x) \geq B(x)$. But then $A(w) = 1 + A(x) \geq 1 + B(x) = B(w)$.

Case 3: The first step in the derivation is $S \rightarrow \mathbf{aSS}$.

Then $w = \mathbf{axz}$, where x is derived from S using less than k steps and z is also derived from S using less than k steps. By the inductive hypothesis, $A(x) \geq B(x)$ and $A(z) \geq B(z)$. But then $A(w) = 1 + A(x) + A(z) \geq B(x) + B(z) = B(w)$.

Notice that the two S 's on the righthand side of the rule might expand into different strings of terminals. A common mistake was to assume that \mathbf{aSS} would expand into \mathbf{aww} and then apply the inductive hypothesis to w .