COT 4210 Quiz #2 Part B: PDAs and Pumping Lemma for CFGs Solutions

2/23/2021

1) (12 pts) Consider designing a PDA to accept the following language of strings over the alphabet {0, 1}:

L = { w | w is a palindrome **with length 1 or greater**, with leading 0s allowed, whose binary value is divisible by 3 }

Note that with this definition, some strings in L are 11, 0110, and 10101. Examples of strings not in L are 110, 101, and 1001001. 110 is not in the language because 110 is not a palindrome. Both 101 and 1001001 are palindromes, but their numerical values in binary, 5 and 73, respectively, are not divisible by 3. **Your PDA should NOT accept ε.**

The design of this PDA is likely to require 9 states. Describe the conceptual meaning of each of the 9 states. In each description, explain both the property of the string read in so far as well as the meaning of the stack contents.

Solution

The key items to think about are that we always need to know the current modulus of the string read in, and we need to non-deterministically need to guess whether or not we more than halfway through the string. We need separate states to know if we are pushing or popping from the stack, and we also need to be able to test if the stack is empty, so we'll have to use the dollar sign (extra tape alphabet character) to place at the bottom of the stack. Here are the states:

q0: start state, nothing has been read in and we haven't even marked the bottom of the stack

q7: nothing has been read in, but we have just pushed a dollar sign onto the stack to mark the bottom of it

q1: we have started reading in our string and currently the value of the string is 1 mod 3 and we are still in the first half of the string (so we pushed 1 onto the stack in route to this state)

q2: we have started reading in our string and currently the value of the string is 2 mod 3 and we are still in the first half of the string.

q3: we have started reading in our string and currently the value of the string is 0 mod 3 and we are still in the first half of the string.

q4: the current value of our string is 1 mod 3 and we are in the second half of the string (so when we transition from here, we pop off items from the stack)

q5: the current value of our string is 2 mod 3 and we are in the second half of the string (so when we transition from here, we pop off items from the stack)

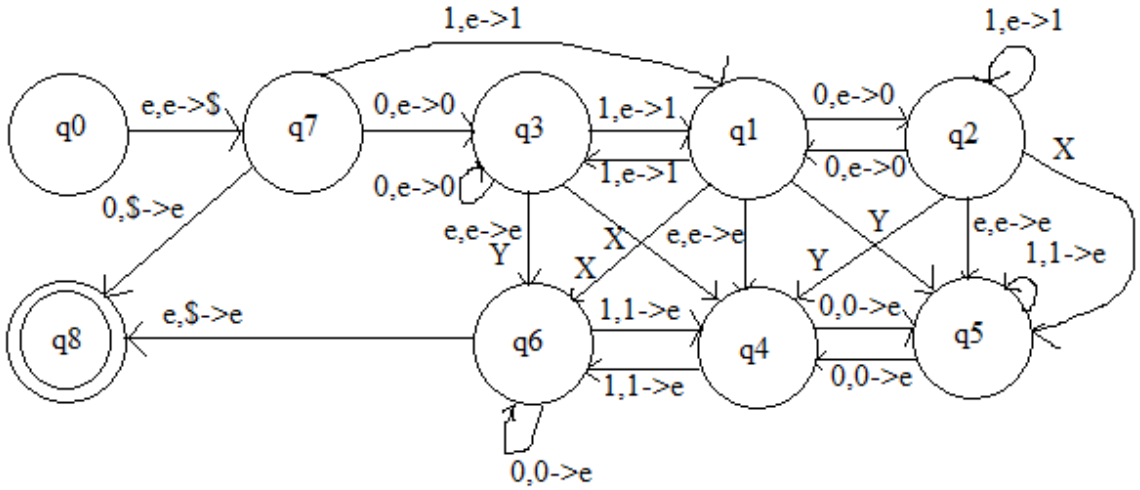
q6: the current value of our string is 0 mod 3 and we are in the second half of the string (so when we transition from here, we pop off items from the stack)

q8: this is our accept state, to have gotten here, we must have popped off the dollar sign en route from state q6 (where the current value is 0 mod 3).

Conceptually, we need 6 states to keep track of the mod value and first versus second half. The other three states are for managing the stack and acceptance.

Grading: 9 pts total for having a state for each mod value and both halves of the string, 3 pts for the other three states.

Here is a full diagram of the PDA with these nine states filled in **(not necessary for a student solution and also doesn't earn full credit.)**



X = 1,e->e
Y = 0, e->e

A few things to note here: the drawing is very complex, which is precisely why I didn't require students to draw it under the time pressure. It's relatively easy to come up with the conceptual states (and this is the fundamental key to the design), but very time consuming to work out all of the transitions. But, once one understands WHAT the states stand for, the transitions mostly write themselves. Here is a high-level description of what is going on: states q3, q1 and q2 act as the "DFA" for processing the first half of the string, pushing on the character read to the stack while the state keeps track of the mod value (q3 is mod 0). States q6, q4 and q5 do the same for the second half of the string (popping what's on the stack off). The key transitions are the non-deterministic transitions between {q3,q1,q2} and {q6,q4,q5}. When we hit the "middle" of the string, we either have an "odd" character that has no match or we don't. One possible transition, for even length strings, is to not make any changes to the stack AND not to read anything in (e, e->e). In doing this transition, the mod value doesn't change, so these are the vertical transitions down. But, if a palindrome is odd length, we must read in this middle character WITHOUT touching the stack, AND adjust the mod value while we consume the character. This necessitates six transitions from the states {q3,q1,q2} to the states {q6,q4,q5}. In these transitions, we read a character (0 or 1), but make no changes to the stack (e->e). We also move from a state in the top to the appropriate state in the bottom based on the character read in.

2) (10 pts) Use the pumping lemma for context free languages to prove that the following language over the alphabet {a, b} is NOT context free:

$$L = \{a^n b^{2^n} \mid n \in Z^+\}$$

Note: Recall that the set Z^+ is the set of positive integers.

Solution

We will show that L does NOT satisfy the pumping lemma for context free languages to prove that L is not context free.

Let p be the pumping length of L. Then, according to the pumping lemma, any string of length p or greater in L satisfies the pumping lemma. Consider the string s = $a^p b^{2^p}$ This string belongs to L and is longer than length p. According to the pumping lemma, the string can be partitioned into five parts s = uvxyz such that |vxy| ≤ p, |v| + |y| > 0 and uv^ixy^iz is in L for all non-negative integer values of i.

Let us consider all possible placements of v and y to prove that no such partition exists:

Case 1: v and y contain the letter a only: uv^2xy^2z will contain MORE than p a's because |v| + |y| > 0, but will still contain exactly 2^p b's. Thus, this string will NOT be in L.

Case 2: v and y contain the letter b only: uv^2xy^2z will contain MORE than 2^p b's because |v| + |y| > 0, but will still contain exactly p a's. Thus, this string will NOT be in L.

Case 3: Either v or y contain both a's and b's: uv^2xy^2z will contain alternating a's and b's so the string won't be in L.

Case 4: The key non-trivial case is where v contains 1 or more a's only and y contains 1 or more b's only. All other cases have already been covered.
Again, let's consider the string uv^2xy^2z. This string must have at least p+1 a's and the maximum number of b's it can have is $2^p + p$. If we can prove that $2^p + p < 2^{p+1}$, then we have proven that this string is not in L as desired.

Note that for all positive integers, $p < 2^p$. Thus, it follows that

$2^p + p < 2^p + 2^p = 2(2^p) = 2^{p+1}$, as desired, completing the proof.

Since L does NOT satisfy the pumping lemma for context free grammars for the string s = $a^p b^{2^p}$, we can conclude that L is NOT context free.

Grading: 1 pt choice of string (if no string is clearly chosen 0 out of 10)
1 pt if string is length p or greater and in L
1 pt case 1, 1 pt case 2
2 pts case 3
4 pts case 4

3) (3 pts) Olivia Rodrigo's hit song Driver's License is about the singer getting her license to do this. (Note: it's really about something else, but at this point, hopefully you know how to answer these questions!)

Drive =) **Grading: Give to All**