Assignment #5

Gustavo Estrela de Matos
CSCE 433: Formal Languages and Automata

April 8, 2016

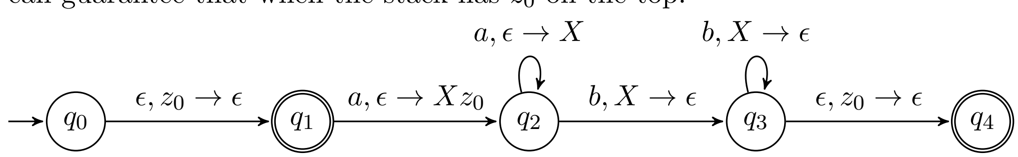
Question 1. For each of the following languages, construct a PDA to accept L.
(a) $L = \{a^n b^{2n} \mid n \geq 0\}$

(b) $L = \{a^i b^j c^k \mid i, j, k \geq 0$ **and** $i + k = j\}$

(c) $L = \{w \mid w$ **has twice as many** 0**'s as** 1**'s**$\}$

(a)

The idea for this language PDA is to have a state where the a's are read and for each of them we push 2 elements in the stack; and a also a state where the b's are read and for each of them an element is poped from the stack. Since we pushed 2 elements for every a and poped one for every b we can guarantee that when the stack has z_0 on the top.

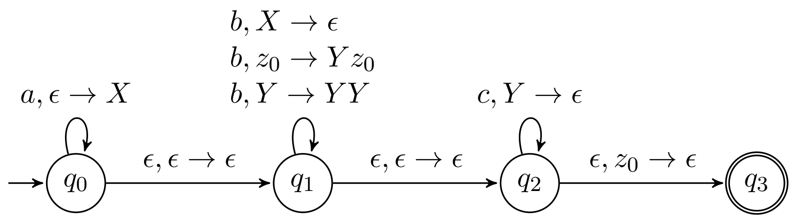


This PDA is deterministic.

(b)

Saying that $i + k = j$ is the same as saying that $k = j - i$, therefore we need to calculate $j - i$ while reading as and bs. To do that we are going to push an element X for every a and pop them for every b read. Once the stack have z_0 on the top we can start pushing Ys for every b read.

The number of Ys in the stack after reading all bs will be the number of cs needed to the input to be in L. If after reading all the bs we have a X on the top of the stack we know we already know that the string is not in L because having a X on the top means that $j < i$ thus there's no positive k such that $i + k = j$.



This PDA is nondeterministic.

(c)

To build this DFA we are going to use two symbols:

- Y: the number of Ys in the stack will represent the number of 0s needed for the string to be in L;
- X: the number of Xs in the stack will represent the number of 1s needed for the string to be in L;
  and the stack will keep the stack with only z_0 and either Xs or Ys.

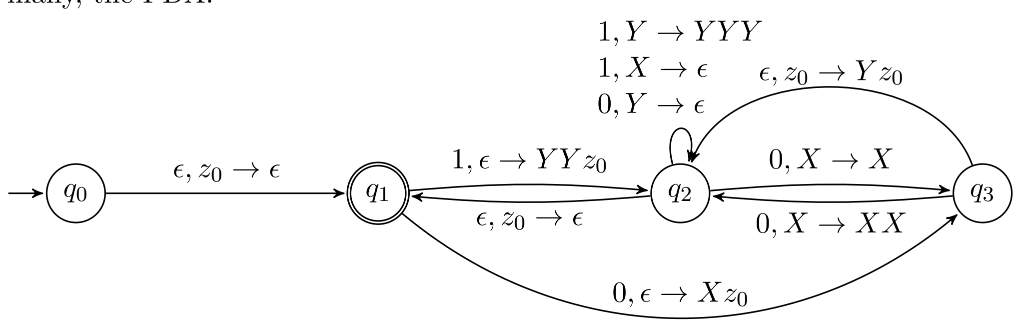
Our DFA will have at least two states: one to represent that the string so far read is in L and other state if there are symbols needed to be read for the string to be in L. If we are in the second state we need to have a feew rules:

- $1, Y \to YYY$: if needed 0s and read a 1 we need two more 0.
- $0, Y \to \epsilon$: if needed 0s and read a 0 we need one less 0.
- $1, X \to \epsilon$: if needed 1s and read a 1 we need one less 1.

But what if we read one more 0 when needing 1s? We would need to read one more 1 and one more 0. Since we don't want to have mixed Xs and Ys in our stack we will create another state that will represent our need for an amount of 1s and one more 0. From this state three things can happen:

- $1, X \to \epsilon$: if we read a 1 we would need one less 1;
- $\epsilon, z_0 \to Xz_0$: if we read all 1s we needed, then we only need to read one more 0;
- $0, X \to XX$: if we read a 0 then we only need to read one more 1 (for the 0 we read when entering and getting out of this state).

Finally, the PDA:



Question 2.

Question 3.

Question 4.