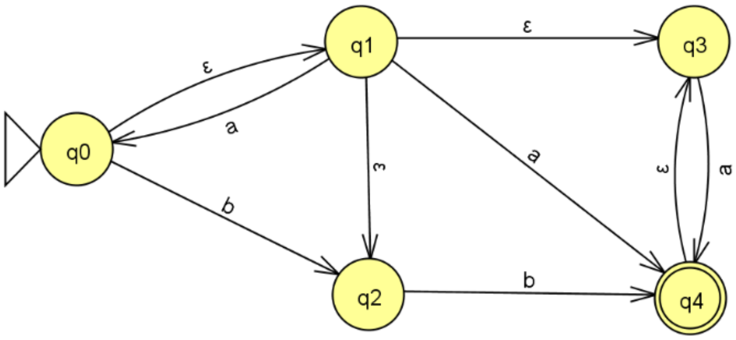
Exame de Época Normal / First Exam (2018/01/16)

Duration: 2h30 **Version A**
No consultation is allowed, other than the supplied document.
No electronic means are allowed (computer, cellphone, …).
Fraud attempts lead to the annulment of the exam for all participants.

Answer each group in separate sheets!
Write your full name and exam version in all sheets!

Group I: [4.5 Points] Finite Automata and Regular Expressions

Consider the following ε-NFA.



a) Determine the ε-closure of each of the ε-NFA states.

Answer:
ε-closure(q0) = {q0, q1, q2, q3}
ε-closure(q1) = {q1, q2, q3}
ε-closure(q2) = {q2}
ε-closure(q3) = {q3}
ε-closure(q4) = {q4, q3}

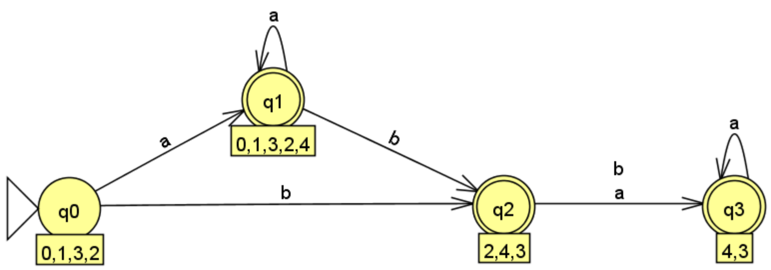
b) Obtain the equivalent DFA for the ε-NFA. Show the table of transitions and the state diagram of the DFA.

Answer:

Table of transitions for the DFA:

State	*a*	*b*
→{q0, q1, q2, q3}	*{q0, q1, q2, q3, q4}*	*{q2, q3, q4}*
**{q0, q1, q2, q3, q4}*	*{q0, q1, q2, q3, q4}*	*{q2, q3, q4}*
**{q3, q4}*	*{q3, q4}*	*Ø*
**{q2, q3, q4}*	*{q3, q4}*	*{q3, q4}*
Ø	*Ø*	*Ø*

State diagram of the DFA (not including the dead state):



The complete DFA includes the state Ø and the transitions δ(q3,b)=Ø, δ(Ø,b)=Ø and δ(Ø,a)=Ø

c) Minimize the obtained DFA. Show the table of distinguishable states and the state diagram for the minimized DFA.

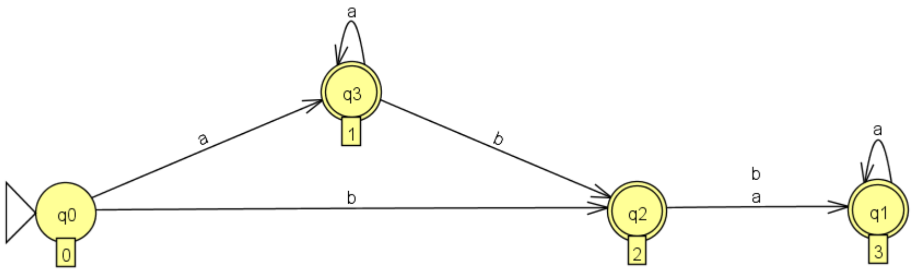
Answer:

Table with distinguishable states (generically we should include also the dead state of the DFA if that state is reachable, but here it is easy to conclude that the dead state is not equivalent to any of the DFA states):

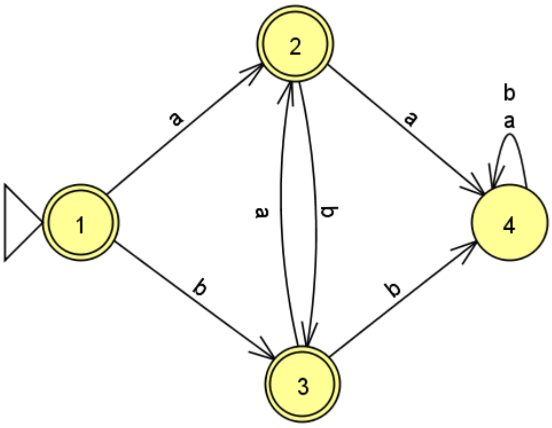
*q1	X		
*q2	X	q1=q3 and q2=q3 X	
*q3	x	q1=q3 and q2= Ø X	q3=q3 and q3= Ø X
	q0	*q1	*q2

All the states are distinguishable.

State diagram of the DFA:



Consider the following DFA.



d) Obtain a regular expression for the language defined by the DFA using the state elimination method. Show all the intermediate steps.

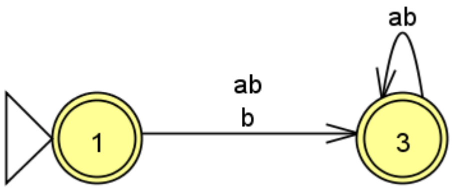
Answer:

Without modifying the DFA to a generalized FA we need to determine 3 regular expressions, one per final state, and then use the OR of them.

State 4 represents a dead state and thus can be eliminated.

Considering first the regular expressions RE13 and RE1

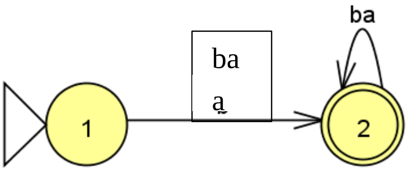
Eliminate state 2:



Considering RE1, State 1 is the final, Eliminate State 3:
RE1 = ε

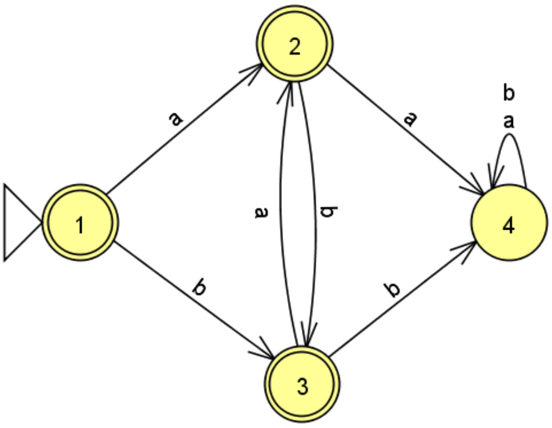
Considering RE13, State 3 is the final:
*RE13 = (ab+b)(ab)**

Consider RE12, State 2 is the final, Eliminate State 3,:



*RE12 = (ba+a)(ba)**

The regular expression of the presented DFA is given by:
RE^1 = RE1+RE13+RE12 = ε + (ab+b)(ab) + (ba+a)(ba)**



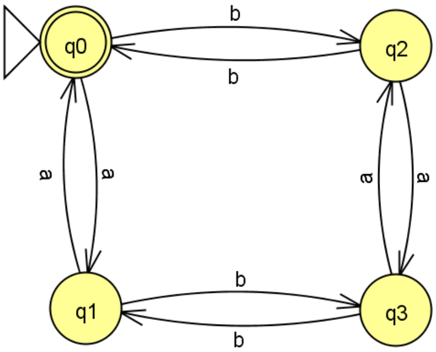
e) Show the regular expressions for the terms $R_{24}^{(0)}$, $R_{11}^{(0)}$, $R_{12}^{(0)}$, $R_{44}^{(0)}$, $R_{21}^{(0)}$, $R_{24}^{(1)}$, $R_{12}^{(1)}$ and $R_{24}^{(1)}$ obtained by the method of path constructions for the conversion of DFAs to regular expressions.

Answer:
$R_{24}^{(0)} = a$
$R_{11}^{(0)} = \varepsilon$
$R_{12}^{(0)} = a$
$R_{44}^{(0)} = a+b+\varepsilon$
$R_{21}^{(0)} = \varnothing$
$R_{12}^{(1)} = R_{12}^{(0)} + R_{11}^{(0)}(R_{11}^{(0)})^* R_{12}^{(0)} = a+\varepsilon(\varepsilon)^*a = a$
$R_{24}^{(1)} = R_{24}^{(0)} + R_{21}^{(0)}(R_{11}^{(0)})^* R_{14}^{(0)} = a + \varnothing(\varepsilon)^*\varnothing = a$
$[R_{24}^{(0)} = a\ R_{14}^{(0)} = \varnothing]$

Suppose that we need a DFA for the language L = {w ∈ {a,b,c}* | no. of a's is even, no. of b's is even and the no. of c's is even}. The DFA for the language L1 = {w ∈ {a,b}* | no. of a's is even and the no. of b's is even} is given (see the DFA below).

☐

[1] Possible RE = ε+a+(b+ab)(ab)*(ε+a)



f) Explain how you can determine a DFA for L based on the DFA of L1 and considering the closure operations of the regular languages.

Answer:
We can consider the DFA for L1, include in each state self c-transitions (we obtain L1'), another DFA for representing L2 = {w ∈ {a,b} | no. of b's is even and the no. of c's is even} (which can be obtained by simply modifying the DFA on the left substituting the a's by c's, include in each state self a-transitions (we obtain L2'). Then one can do the intersection of L1' and L2' (e.g., via the product of the two DFAs) and we obtain the DFA for L.*

I.e., this is the intersection of:
L1' = {w ∈ {a,b,c} | no. of a's is even, no. of b's is even}*
With
L2'= {w ∈ {a,b,c} | no. of b's is even and the no. of c's is even}*
to obtain L and based on the DFA given for L1.

We intend to specify as a regular expression the language representing registers of persons. Each register uses a line and there might exist 0 or more registers in the same file.

g) Show a regular expression that allows the validation of the format of data assuming that each line consists of: name of the person, followed by the birthday date, separated by ':', as we can see in the examples below. Use the symbols 'M' to represent an uppercase letter, 'm' for a lowercase letter (both including letters with accents), 'D' for a digit, 'E' for a space and 'R' for a new line. Indicate the alphabet used in the expression and in the case you use extra symbols explain their meaning. The birthday dates must be represented by: DD/MM/AAAA or DD-MM-AAAA, and the leftmost zeros can be omitted and the fields DD and MM may have values from 1 to 31 and from 1 to 12, respectively. Examples:

Dionísio Adalberto da Silva Côrte-Real : 01/04/1999
Leonilde Maria do Ouro dos Anjos e Ramos da Árvore : 4/08/2000

Answer:
S = {D, M, m, E, T, P, R, B}[2]*, T = '-', P=':', B='/', d=D\'0'*

RE = (
Mm((E+T)(M+m)m*)* EPE (d+0d+1D+2D+30+31) (T+B)*
(0d+d+1(0+1+2)) (T+B) (D +DD+DDD+DDDD) R
*)**
(
ε +
Mm((E+T)(M+m)m*)* EPE (d+0d+1D+2D+30+31) (T+B)*
(0d+d+1(0+1+2)) (T+B) (D +DD+DDD+DDDD)
*)**

As they may have considered that the leftmost zeros that can be omitted ae only for the DD field

☐

2 Note that you can substitute the special symbols by the symbols they represente.

RE = (
Mm((E+T)(M+m)m*)* EPE (d+0d+1D+2D+30+31) (T+B)*
(0d+10+11+12) (T+B) (DDDD) R
*)**
(
ε +
Mm((E+T)(M+m)m*)* EPE (d+0d+1D+2D+30+31) (T+B)*
(0d+d+1(0+1+2)) (T+B) (D +DD+DDD+DDDD)
*)**

Group II: [2 Pts] Properties of Regular Languages

Prove that if each of the following languages is regular or not. In case a language is regular, show a DFA, NFA or ε-NFA to recognize the language. Note: if necessary, assume that the language $\{a^n b^n \mid n \geq 1\}$ has been proved as a non-regular language.

a) L = {w | w ∈ {o,n,y,e,s}* and w has the same number of "no" and "yes" substrings}.

Answer:
If we intersect the language given by the regular expression $(yes)^+(no)^+$ with L we obtain the language given by $(yes)^n(no)^n$, i.e.:

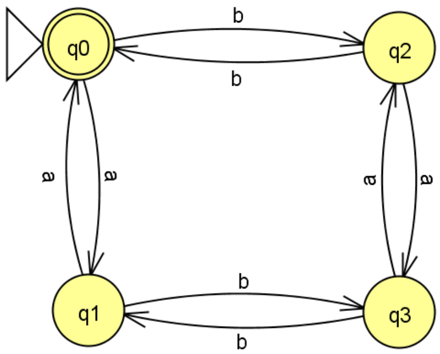
Lang($(yes)^+(no)^+$) ∩ L = L1=$\{x^n y^n \mid x =$ "yes", $y=$"no" and $n \geq 1\}$

If we use homomorphism and apply:
h(yes)=a
h(no)=b
to L1 we obtain the language $\{a^n b^n \mid n \geq 1\}$ which we assume that has been proved as a non-regular language.

From the closure properties of the regular languages we can conclude that L is a non-regular language, because if it was a regular language, the intersection with a regular language and then the application of homomorphism over the resultant language would be a regular language and it is not.

b) L = {w | w ∈ {a, b, c}* and contains an even number of a's and an odd number of b's}.

Answer:
L is a regular language as we can provide a DFA for L.
For example, if we take the DFA given for the even number of 0's and even number of 1's:



We can see that state q2 is the state with an even number of a's and an odd number of b's. if we move the final state of this DFA from state q0 to state q2 and if we include in each of the states a self-transition with 'c' we obtain a DFA for L and thus L is a regular language.

Group III: [4.5 Pts] Context-Free Grammars (CFG) and Push-Down Automata (PDA)

$S \rightarrow \varepsilon \mid (S)S$

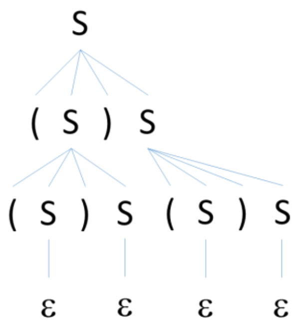
Consider the CFG G in the left.

a) Show the syntax tree and a letfmost derivation for the string: (())().

Answer:

Leftmost derivation:
S ⇒ (S)S ⇒ ((S)S)S ⇒ (()S)S ⇒ (())S ⇒ (())(S)S ⇒ (())()S ⇒ (()
)()

Syntax Tree:



b) The CFG G is ambiguous? Justify. If it is ambiguous, modify the grammar in order to eliminate the ambiguity.

Answer:
The CFG is not ambiguous. None of the strings of the language can have more than one syntax tree. The recursivity in the CFG is to derive nested (X) or sequences of (X)(X).

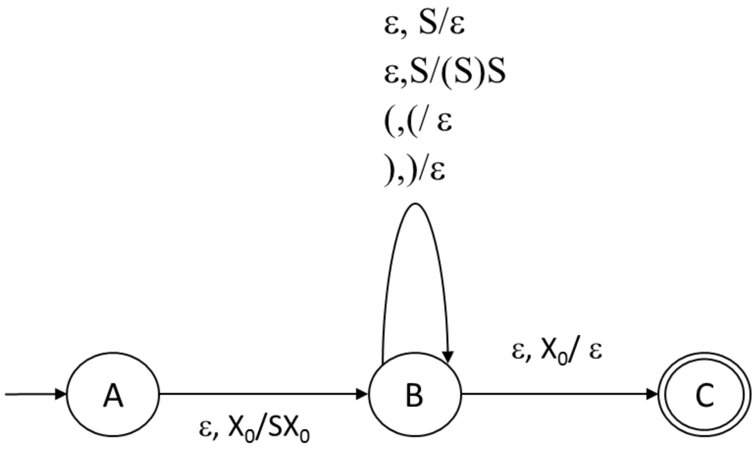
c) Suppose one wants to represent the L language given by G with a CFG for which all the syntax trees are binary trees. Indicate the new CFG for L.

Answer:
The following is a possible CFG for L, with binary syntax trees:

$S \to \varepsilon \mid TS$
$T \to U)$
$U \to (S$

d) Indicate a PDA accepting by final state of G.

Answer:
PDA begins with X0 on the stack:



e) The previous PDA is deterministic or non-deterministic? Justify your answer.

Answer:
The PDA is non-deterministic. For the symbol S in the top of the stack and for any input symbol in the input there are always two transitions: to pop S without pushing any symbol or to pop S and push (S)S.

f) Indicate a sequence of instantaneous descriptions that result in the acceptance of the string: ().

Answer:
(A,(), X0) |- (B,(),SX0) |- (B,(), (S)SX0) |- (B,),S)SX0) |- (B,),)SX0) |- (B,ε,SX0) |- (B,ε,X0) |- (C,ε,ε)

g) Considering that given a language L, the set of prefixes of L, denoted by Pref (L), contains all the prefixes of the strings in L (i.e., a string x belongs to Pref (L) when it exists another string y and xy belongs to L), indicate a grammar for Pref (L(G)).

Answer:
$S \to \varepsilon \mid (S)S \mid (S$

Group IV: [4 Pts] Turing Machine

We intend to implement a Turing Machine ™ to recognize the strings of the language in the alphabet {a,b,c} in the format $\{ab^na^pc^{(n-p)}, n,p>0, n \geq p\}$.

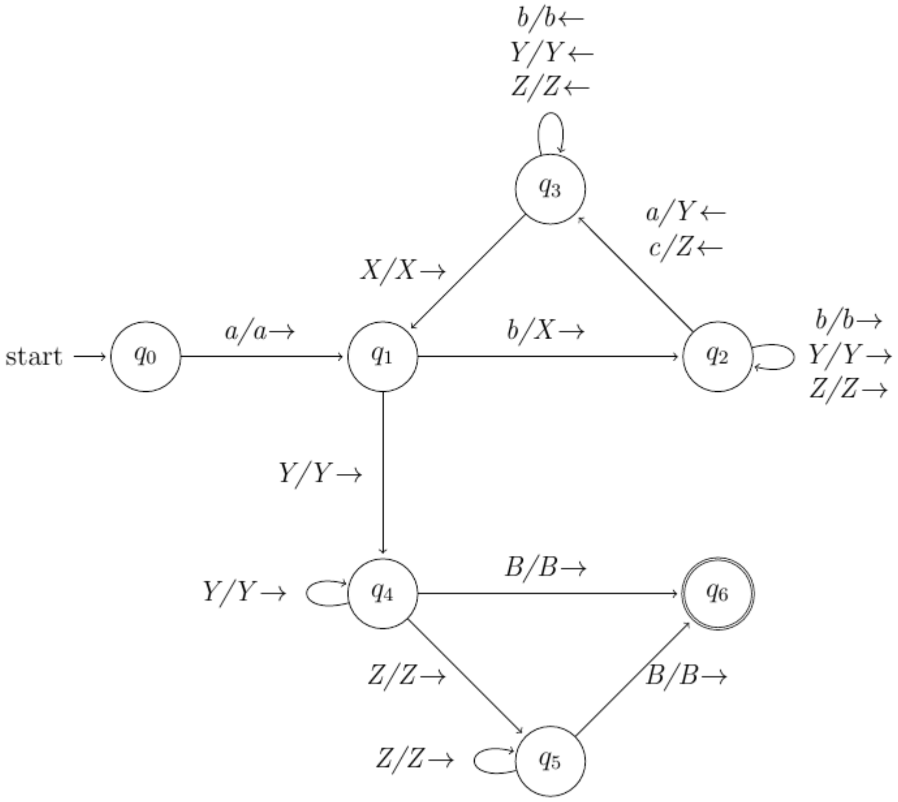
Ex.: the strings abbaa and abbbaac belong to the language while the strings abbbaaac and abbbaca do not belong. Note: it is not needed to maintain the input string in the end of the computations.

a) Describe a strategy to implement a TM to perform the recognition of strings of the language.

Answer:
There are as many b's as characters to the right of the b's (sum of a's and c's). A possible strategy to implement the TM is to start by recognizing the first 'a' in the string, followed by a cycle where we match each 'b' to either an 'a' or a 'c' to the right of the b's, marking the processed 'b' as an X, the processed 'a' as a Y and the processed 'c' as a Z. When there are no more b's to process, we make verifications to the string format, ensuring that there was at least one 'a' (replaced by a Y) in the original string (at the same time, this also ensures that there was at least one 'b'); that if there were c's in the original string, they all come after the a's; and that there is no excessive number of a's or c's.

b) Draw a possible TM.

Answer:



c) Indicate the computing trace when the input to the TM is: abbac.

Answer:
0abbac |- a1bbac |- aX2bac |- aXb2ac |- aX3bYc |- a3XbYc |- aX1bYc |- aXX2Yc |- aXXY2c |- aXX3YZ |- aX3XYZ |- aXX1YZ |- aXXY4Z |- aXXYZ5B |- aXXYZB6B

Group V: [5 Pts] Statements about Languages (V/F: 20%, justification: 80%; wrong answer = reduction of 50%)

Indicate, justifying succinctly (with 1 or 2 sentences or a counter example), whether each of the following statements is True or False.

a) If we show based on the pumping lemma for regular languages that a language defined by a subset of strings of a given language L is non-regular, it is immediately proved that L is non-regular.

Answer: *FALSE* *Let's suppose the language L1=$\{a^n b^n \mid n \geq 0\}$, which is a non-regular language. L1 is a language consisting of a subset of strings of the regular language L(a*b*). Thus, being L1 a non-regular languages does not imply that L(a*b*) is a non-regular language.*

b) The equality $L1 \cap L2 = \Sigma^* \setminus ((\Sigma^* \setminus L1) \cup (\Sigma^* \setminus L2))$ is wrong (note: "\" stands for the subtraction operation).

Answer: *TRUE (assuming that the alphabets of L1 and L2 are also Σ)* *Note that: $\Sigma^* \setminus L1 = \overline{L1}$ and $\Sigma^* \setminus L2 = \overline{L2}$* *So, $\Sigma^* \setminus ((\Sigma^* \setminus L1) \cup (\Sigma^* \setminus L2)) = \Sigma^* \setminus (\overline{L1} \cup \overline{L2}) = \overline{\overline{L1} \cup \overline{L2}}$* *$= L1 \cap L2$* *And the equality is true.*

c) The language L= $\{0^i 1 0^j 1 0^k \mid k < i+j\}$ is a context-free language (CFL);

Answer: *TRUE* *We can think of a PDA that process the first sequence of 0's and push them to the stack, then process a '1', then the second sequence of 0's and push them to the stack, then process a 1 and for the third sequence of 0's pop a '0' from the stack for each '0' in the input. If all the symbols in the input string are processed (consumed) and there still exist at least an 'a' in the stack then the PDA accepts it.* *If there is a PDA for L then there exists a CFG for L and this means that L is a CFL.*

d) If A is an NFA then L(A), i.e., the language represented by A, é is a context-free language (CFL).

Answer: *TRUE* *A language given by an NFA is a regular language and the regular languages are a subset of the CFLs. So, L(A) is a CFL.*

e) The intersection of two context-free languages does not result always in a context-free language.

Answer: *TRUE* *The intersection is not a closure operation over CFLs. For example the intersection of the two CFLs:*

$L1= \{a^n b^n c^k \mid n, k \geq 0\}$ and
$L2= \{a^k b^n c^n \mid n, k \geq 0\}$ is
$L3=\{a^n b^n c^n \mid n \geq 0\}$, which is not a CFL.

f) We can determine a regular expression that represents the language of a PDA by converting the PDA to a DFA and then use the state elimination conversion method.

Answer:
FALSE
A PDA can be only converted to a DFA if the language represented by the PDA is a regular language. If the language represented by a PDA is a non-regular language then it does not exist any DFA and regular expression representing that language.

g) If a context-free grammar (CFG) is ambiguous, then the PDA that implements the language of that grammar must be always non-deterministic.

Answer:
FALSE
The following CFG is ambiguous and we can think of a non-deterministic PDA to implement the language:
$S \rightarrow SS \mid a \mid \varepsilon$
This CFG represents the language L(a), which is a regular language, with a DFA and also with a non-deterministic PDA.*

h) If the tape of a Turing Machine (TM) is finite then the languages that that TM recognizes can be recognized by deterministic PDAs.

Answer:
TRUE
If the tape of a Turing Machine (TM) is finite means that this TM only recognizes finite languages. As every finite language is a regular language, we can recognize it by a DFA and thus by a deterministic PDA. (note that to obtain a deterministic PDA from a DFA we only need to extend the transitions with ",Z /Z" considering 'Z' as the symbol in the stack in the beginning).