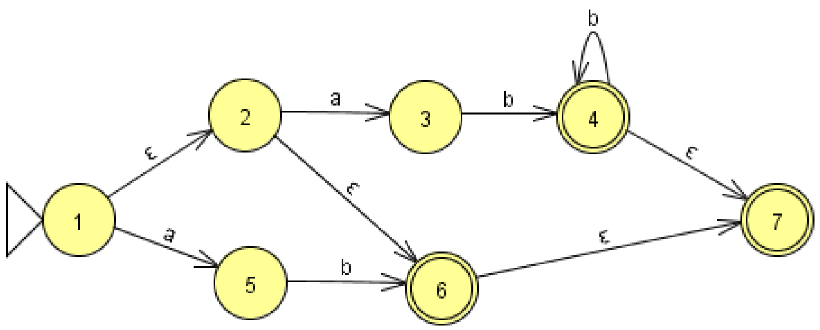
Exame de Época Normal / First Exam (2017/01/19)

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) Show the ε-closure for each of the states of the ε-NFA.

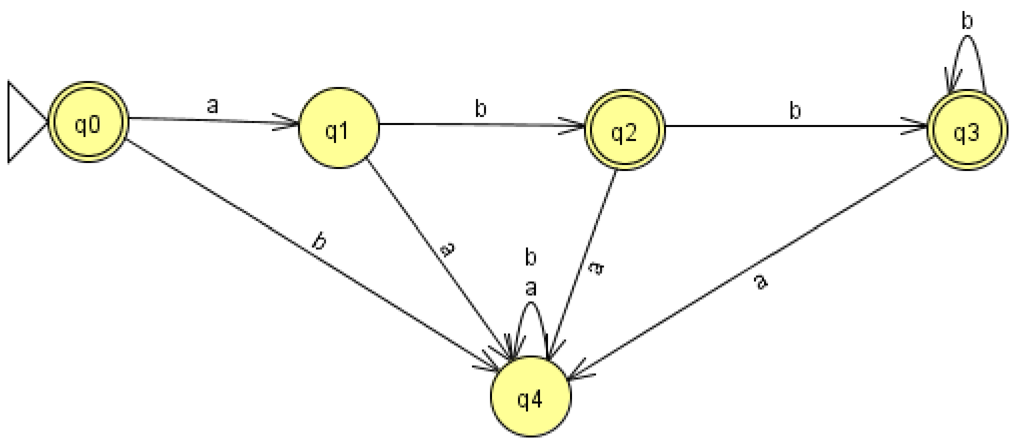
Answer:
ε-closure(1) = {1, 2, 6, 7}
ε-closure(2) = {2, 6, 7}
ε-closure(3) = {3}
ε-closure(4) = {4,7}
ε-closure(5) = {5}
ε-closure(6) = {6, 7}
ε-closure(7) = {7}

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

Answer:

state	a	b
→*{1,2,6,7}	{5, 3}	∅
{5, 3}	∅	{4,6,7}
*{4,6,7}	∅	{4,7}
*{4,7}	∅	{4,7}
∅	∅	∅

State diagram of the DFA with state ∅ labeled by "q4"



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

Answer:

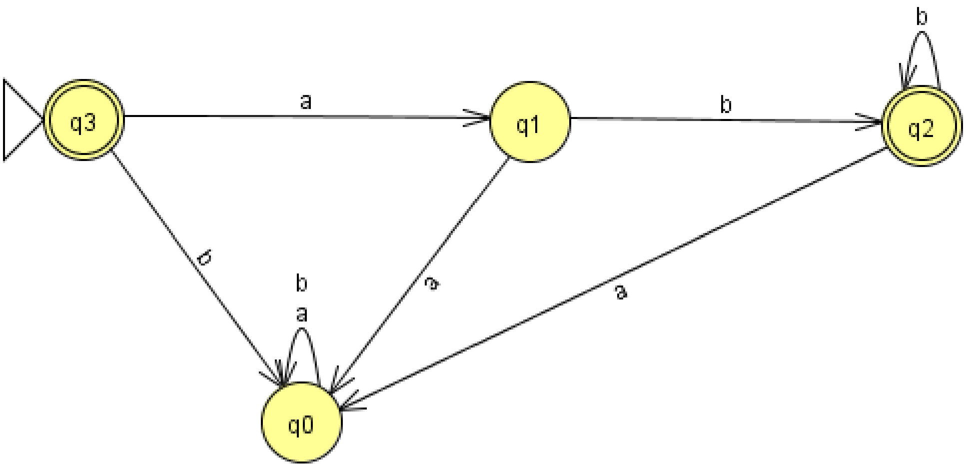
Table of distinguishable states:

q1	*x*			
**q2*	*x*	*x*		
**q3*	*x*	*x*		
q4	*x*	*x*	*x*	*x*
	**q0*	*q1*	**q2*	**q3*

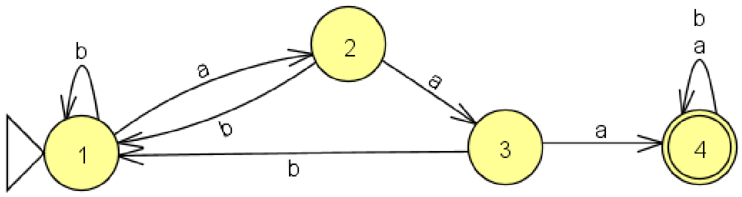
Equivalent states: {q2, q3}

Minimized DFA with the following correspondence between the states of the DFA above and the states of the minimized DFA:

{q2, q3} → q2
q4 → q0
q0 → q3
q1 → q1



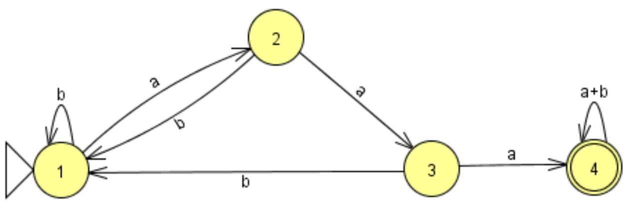
Consider the DFA below.



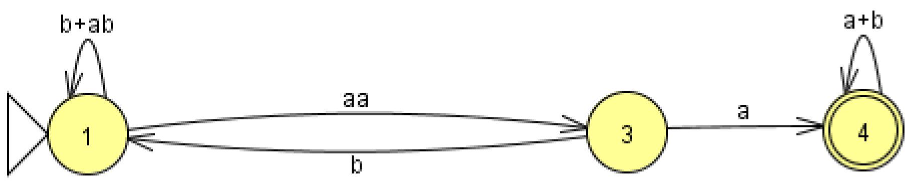
d) Obtain a regular expression for the language defined by the DFA using the state elimination method and considering the ordering of elimination 2→3 (i.e., first eliminate state 2 and then state 3). Show all the intermediate steps.

Answer:

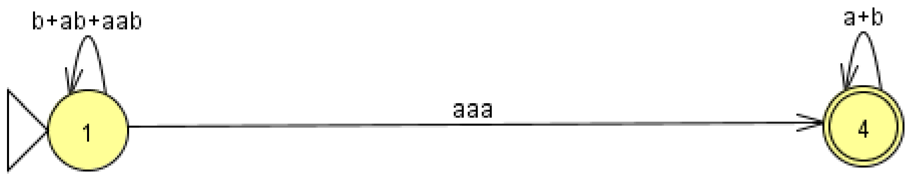
DFA for translating to RE using the state elimination method:



After elimination of state 2:



After elimination of state 3:



And the regular expression is:
*RE= (b+ab+aab)*aaa(a+b)**

e) Show the expressions for the terms $R_{12}^{(0)}$, $R_{11}^{(0)}$, $R_{22}^{(0)}$, $R_{23}^{(1)}$, $R_{12}^{(1)}$ and $R_{14}^{(1)}$ obtained by the path construction method to convert the DFA in a regular expression.

Answer:

$R_{12}^{(0)}=a$

$R_{11}^{(0)}=b+\varepsilon$

$R_{22}^{(0)}= \varepsilon$

$R_{23}^{(1)}= R_{23}^{(0)}+ R_{21}^{(0)}(R_{11}^{(0)})^* R_{13}^{(0)}=a+b(b+\varepsilon)^*\varnothing=a$

$R_{12}^{(1)}= R_{12}^{(0)}+ R_{11}^{(0)}(R_{11}^{(0)})^* R_{12}^{(0)}=a+(b+\varepsilon)(b+\varepsilon)^*a=a+b^*a=b^*a$

$R_{14}^{(1)}=R_{14}^{(0)}+ R_{11}^{(0)}(R_{11}^{(0)})^* R_{14}^{(0)}= \varnothing+(b+\varepsilon)(b+\varepsilon)^*\varnothing=\varnothing$

Consider the existence of a DFA of a regular language L:

f) Suppose that we intend to implement L1 = L:/{a} which must accept all the strings accepted by L, but without the a's that can exist, e.g., if L={abcaa, bb, ab, ccaaaa} then L1={bc, bb, b, cc}. Indicate a method that can be applied to the DFA of L to obtain a DFA that implements L1= L:/{a}.

Answer:
Two methods:
(a) Substitute all transitions of type q(p,a) by q(p, ε) and then transform the resultant ε-NFA in a DFA.
(b) Transform the DFA in a resultant DFA using the symbol "a" in a-closure and eliminating the transitions with "a" in a similar way of ε transitions using ε-closure in the translation of ε-NFA to DFA.

The insertion of grades in SIGARRA can be done with the use of a file formatted with the grades of the students for a given course:

g) Obtain a regular expression to validate the format of the information for a student, which consists of the student number, name, and grade, separated by ':', as shown in the examples below. Use the symbols 'M' to represent an uppercase letter, 'm' for a lowercase letter (both include letters with accentuated), 'D' for a digit, 'E' for a space. Indicate the alphabet used in the regular expression and in case you use an extra symbol indicate its meaning. The student number must have 9 digits, in which the first 4 digits form a number between 1970 and 2019. The grade information can be RFC, RFF, or a number between 0 and 20, corresponding to the grade obtained. Examples:

197560430 : Dionísio Adalberto da Silva Côrte-Real : RFC
200540075 : Leonilde Maria do Ouro dos Anjos e Ramos da Árvore : 12

Answer:
S = {0,1,2,7,8,9,D, M, m, E, T, P, R, F, C}, T = '-', P=':'
RE = (19(7+8+9)D + 20(0+1)D) DDDDD EPE Mm((E+T)(M+m)m*)* EPE (RFC+RFF+D+1D+20)*

Group II: [2 Pts] Properties of Regular Languages

Prove for each of the following languages if it is a regular language or not. If it is, show a finite automaton that represents the language.

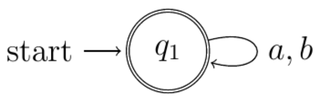
a) L = {xwywz | x,y,z ∈ {a, b}* and w ∈ {c, d}*}.

This language is not regular. Using the Pumping Lemma to perform a proof by contradiction, we start by choosing a string belonging to L. Let us choose the string $c^m a c^m$*, belonging to L (x and y will be empty strings, w=*c^m *and y=a). The string has a length of 2m+1, which respects the condition set forth by the pumping lemma of having a length equal to or greater than m.*
We now have to check all possible divisions of the string in three parts (p, q and r) such that |pq| <= m, |q|>0. Given these constraints, q will have only c's. Putting it in a more formal manner:
$p = c^s$, $s>=0$
$q = c^t$, $t>0$
$r = c^{m-s-t}ac^m$
This division sums all possible divisions of the string in three parts, respecting the conditions set by the Lemma. For k=0, the generated string (pq^0r*) will no longer belong to L:*
$pq^0r = pr = c^s\, c^{m-s-t}ac^m = c^{m-t}ac^m$
As we know that t >0, it means that the left part of the string will have fewer c's than the right part of the string (at least one c), which means that this string doesn't belong to L.
Having started from a string belonging to L and proven that there is no division of the string that allows for the generation of strings belonging to L for any value of k>=0, we prove that L is not regular.

In fact, if we wanted to build a DFA to recognize this language, it would be impossible to do so, as it would be necessary to have na infinite number of states to memorize the sub-string w, in order to guarantee that this sub-string would repeat itself ahead in the string.

b) L = {xwywz | x,w,y, z ∈ {a, b}*}.

This language is regular. As all the substrings of the string are in the same alphabet and define (a+b) we can simplify to (a+b)*, represented by the following DFA:*



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

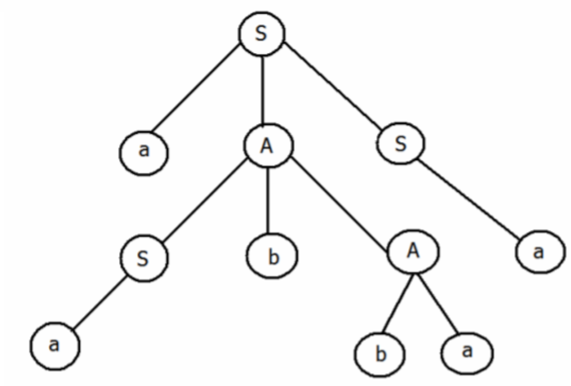
S → aAS | a
A → SbA | SS | ba

Consider the CFG G on the left, in which S is the initial variable.

a) Show an analysis tree and a leftmost derivation for the string: aabbaa.

Answer:

Leftmost derivation:
S ⇒ aAS ⇒ aSbAS ⇒ aabAS ⇒ aabbaS ⇒ aabbaa



b) Is the CFG G ambiguous? Justify. If it is ambiguous, modify the grammar in order it is non-ambiguous.

Answer:
The grammar is ambiguous as there is at least a string w with two or more distinct syntactic (analysis) trees.
Example: w=aaaaaaa

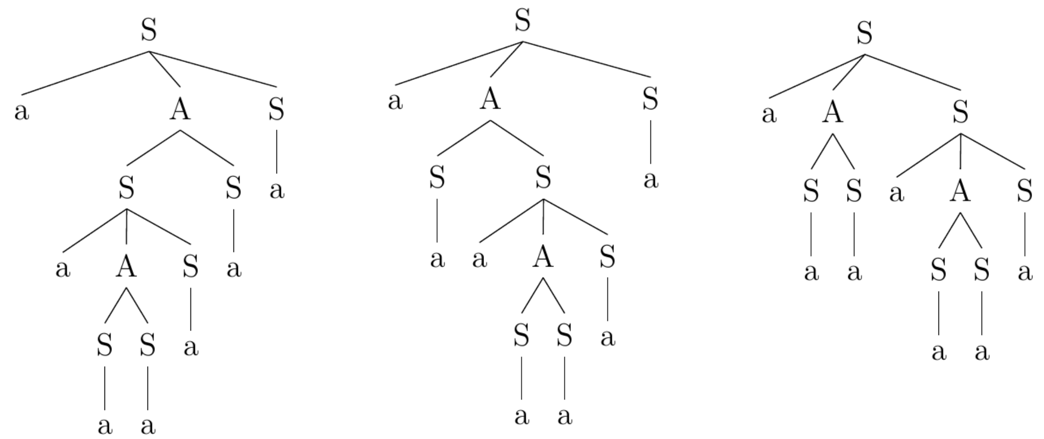
Two of the possible leftmost derivations for w[1]*:*
S⇒aAS⇒aSSS⇒aaASSS⇒aaSSSSS⇒aaaSSSS⇒aaaaSSS⇒aaaaaSS⇒aaaaaaS ⇒aaaaaaa

S⇒aAS⇒aSSS⇒aaSS⇒aaaASS⇒aaaSSSS⇒aaaaSSS⇒aaaaaSS⇒aaaaaaS⇒aaaaaaa

A possibility to eliminate the non-ambiguity is using the following CFG:
S → A1 | A1 b T1 A2 | A2 bb T1 A2
T1 → A3 bb T1 | A2 b T1 | e
A2 → aa | aaa A2
A1 → a | aaa A1
A3 → aaa | aaa A3

c) Suppose that we intend to represent the language L given by G with a CFG in which the analysis trees are binary trees. Show a CFG for L.

[1] *Below are three possible syntactic trees for w:*

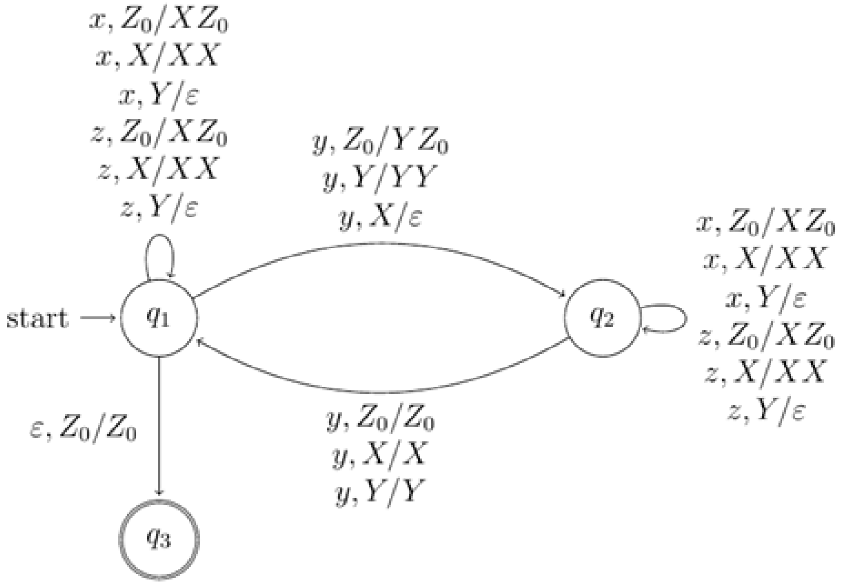


As the grammars in the Chomsky Normal Form follow this property, we can translate the CFG to the CNF.
For this particular CFG, we only need to modify all the productions to be of the form: A→BC or A→a.
Hence, we obtain the CFG on the right:

S → FC | a
A → SD | SS | BF
C → AS
D → BA
B → b
F → a

d) Indicate a PDA accepting by final state to recognize the language of the strings in the alphabet {x, y, z} in which the total number of y's is twice of the sum of x's and z's.

A possible strategy can use the stack to maintain a counter of symbols. Using two states (one q_1 for the even number of y's and other q_2 for the odd number), and the symbol Y to represent excess of y's and X to represent the lack of y's, we can obtain the following PDA:
PDA = ({q_1,q_2,q_3}, {x,y,z}, {Z_0, X, Y}, δ, q_1, Z_0, {q_3})



e) The previous PDA is deterministic or not? Justify your answer.

The PDA is non-deterministic. A transition from state q_1 to q_3 with Z_0 in the top of the stack and any input symbol (ε) causes non-determinism due to the existence of other transitions from q_1 with Z_0 in the top of the stack.

f) Indicate the sequence of instantaneous descriptions to process the string: yxyzyy.

(q_1, yxyzyy, Z_0) ├ (q_2, xyzyy, YZ_0) ├ (q_2, yzyy, Z_0) ├ (q_1, zyy, Z_0) ├ (q_1, yy, XZ_0) ├ (q_2, y, Z_0) ├ (q_1, ε, Z_0) ├ (q_3, ε, Z_0)

Group IV: [4 Pts] Turing Machine

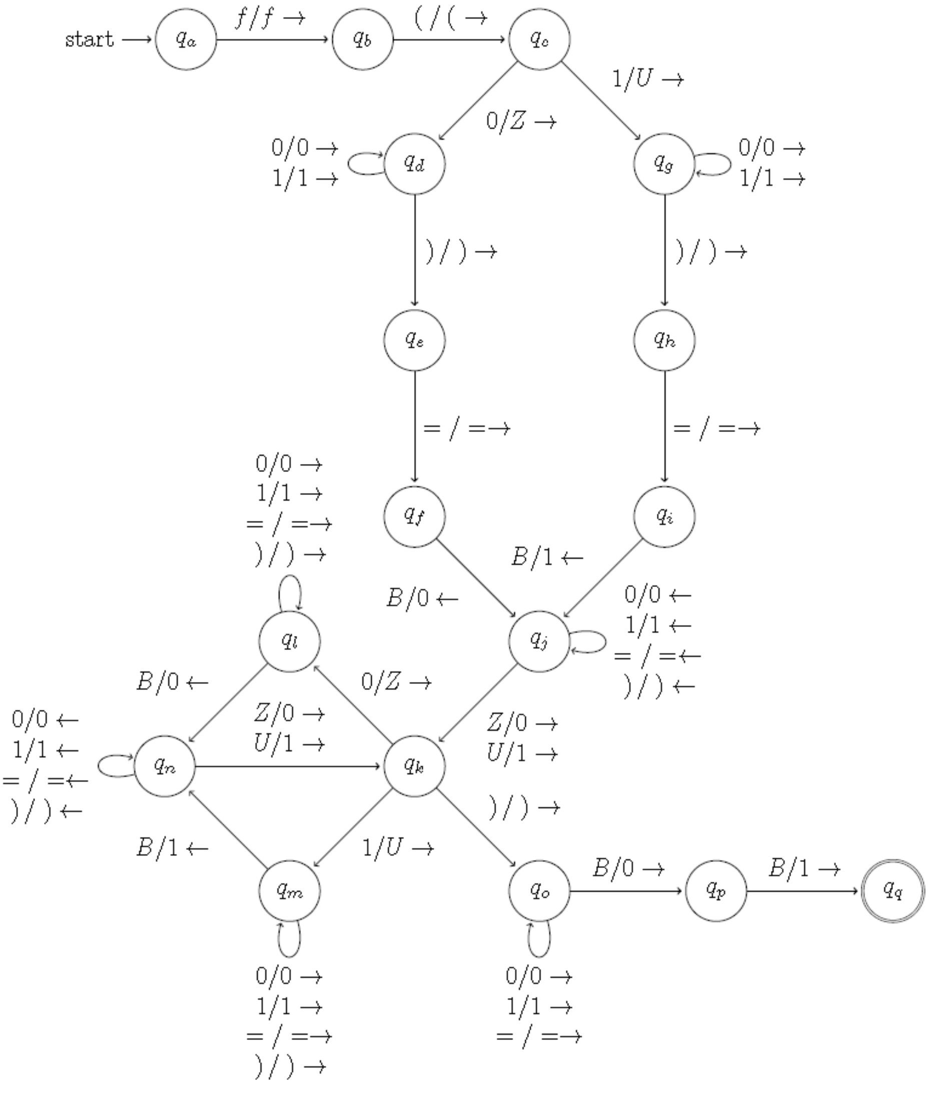
We intend a Turing machine to perform the operation 4×a+1, in which "a" is a natural number represented in binary. The input in the tape uses the format: "f(", followed by the binary number and ")=", and the result must be placed immediately after the symbol "=" of the input string (the input string must be maintained).

a) Describe a strategy for a deterministic Turing Machine that implements this operations.

The multiplication of a binary number by 4 is equivalent to shift left the number by two, which can be done by adding two 0's to the right of the least-

significant bit. The sum of 1 to the value obtained can be done by substituting the least-significant bit of 4xa by 1. Thus, 4xa+1 can be obtained by adding "01" to the right of the least-significant bit of a.
As we like the result in the right of the "=" symbol we start by copying the number between "(" and ")" to the right of "=" and then to add "01" to the rightmost of the copied number.

b) Draw the respective Turing Machine.



c) Show the sequence of instantaneous descriptions when the input in the tape is: f(1)=.

q_af(1)= ├ fq_b(1)= ├ f(q_c1)= ├ f(Uq_g)= ├ f(U)q_h= ├ f(U)=q_iB ├ f(U)q_j=1 ├ f(Uq_j) =1 ├ f(q_jU) =1 ├ f(1q_k) =1 ├ f(1)q_o=1 ├ f(1)=q_o1 ├ f(1)=1q_oB ├ f(1)=10q_pB ├ f(1)=101q_qB

Group V: [5 Pts] Statements about Languages (T/F: 20%; justification: 80%))

Indicate, justifying succinctly, whether each of the following statements is True or False.

a) A language L defined by a CFG can never be a regular language.

False. CFGs represent the CFLs and CFLs include RLs. Thus, there are CFLs that are RLs. Example S $\rightarrow$ a is a CFG of a CFL and can be represented by an FA (DFA, NFA, or ε-NFA) and thus it is an RL.

b) For a CFG G there exists always a non-deterministic PDA which defines L(G).

True. There is a method to convert any CFG to a PDA (deterministic or non-deterministic depending on the input CFG) and we can easily transform a deterministic PDA into a non-deterministic one by include a transition that does not affect the recognized language and adding non-determinism.

c) All regular expressions can be implemented with PDAs.

True. Any regular expression RE has a DFA that implements L(RE). From the DFA of L(RE) we can obtain a PDA recognizing L(RE) by using the state diagram of the DFA and adding the "Z/Z" to each transition (i.e., $\delta(q,\alpha) = p$ is translated to $\delta(q,\alpha, Z) = (p,Z)$) and assuming our PDA starts with Z in the stack.

d) If a CFG is ambiguous, then the PDA for that CFG, and obtained by the algorithm to convert CFG→PDA, is non-deterministic.

True. If the CFG is ambiguous there is at least one string (w) that is accepted by two or more distinct derivations. As the conversion method to translate a CFG to a PDA "emulates" the possible derivations, there will be also two or more sequences of instantaneous descriptions of the PDA to accept w.

e) A language L is a CFL if and only if exists a CFG G non-ambiguous such that L=L(G).

False. There are CFLs inherently ambiguous and thus without a non-ambiguous CFG.

f) In order to prove that a CFG G is ambiguous we have to find a string $w \in L(G)$ for which there exist at least two distinct syntactic trees.

True. A CFG is ambiguous if there is at least a sting w for which there are two or more distinct derivations (leftmost or rightmost) to accept w and this implies that there are two or more syntactic trees (also known as parse trees or analysis trees) for w.

g) When we eliminate the ambiguity of grammars that represent languages of arithmetic expressions we obtain a grammar that satisfies the priority of the arithmetic operators.

False. Eliminating the ambiguity of a CFG does not imply to write a CFG that satisfies the priority of the arithmetic operators. For example, consider the ambiguous CFG for simple arithmetic expressions with + and x operators:

$E \rightarrow E + E \mid E \times E \mid I$

$I \rightarrow a \mid \ldots \mid 1 \mid \ldots \mid aI \mid 1I \ldots$

A non-ambiguous CFG for the language defined by the above language is:

$E \rightarrow I + E \mid I \times E \mid I$

$I \rightarrow a \mid \ldots \mid 1 \mid \ldots \mid aI \mid 1I \ldots$

This CFG, however, does not satisfy the priority of the operators as the operators are not placed in different levels of the syntactic trees according to their priority. For example the expression axa+a is interpreted as ax(a+a) and not as (axa)+a.

h) If the tape of a Turing Machine is finite then the languages recognized by that machine can be recognized by a DFA.

True. An hypothetical Turing Machine with a finite tape will deal with a finite number of strings and will accept finite languages and/or will calculate operations over numbers represented by a known finite-length strings. So, any language (as it is finite) can be recognized by a DFA. [it is possible to enumerate all the possible accepted strings and represent them as a DFA]