UNIVERSIDAD CARLOS III DE MADRID
FORMAL LANGUAGES AND AUTOMATA THEORY
GRADO EN INGENIERÍA INFORMÁTICA.
Final Exam – January 2011-12 17th January 2012

	LAST NAME(s) (Capital letters)			
	FIRST NAME (Capital letters)			
	NIA		DNI	

INSTRUCTIONS

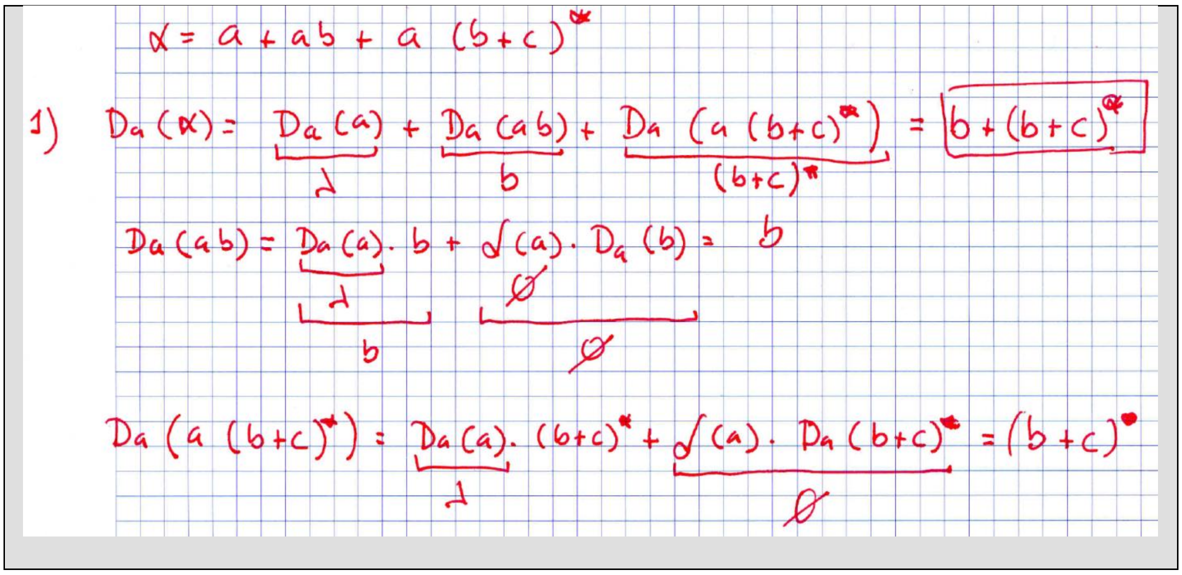
- **Solve each one of the problems starting in a new sheet.**
- **Do not forget to write your name and NIA using capital letters in every answer sheet.**
- **Pay attention to what it is asked in each question and/or problem, given that it is not the same: to explain, to list, to describe, to define, etc., always, sometimes, at least.**
- **Duration of the exam (TEST + PROBLEMS): 3 hours and 30 minutes.**

Problem 1 (1 point)

Given the alphabet $\Sigma=\{a,b,c,d\}$ and the regular expression: $\alpha = a + ab + a\,(b+c)^*$

Calculate the following derivatives:

- $D_a(\alpha)$
- $D_b(\alpha)$
- $D_c(\alpha)$
- $\delta(D_a(\alpha))$
- $\delta(D_b(\alpha))$
- $\delta(D_c(\alpha))$
- Given $\beta = D_a(\alpha)$, obtain $D_b(\beta)$

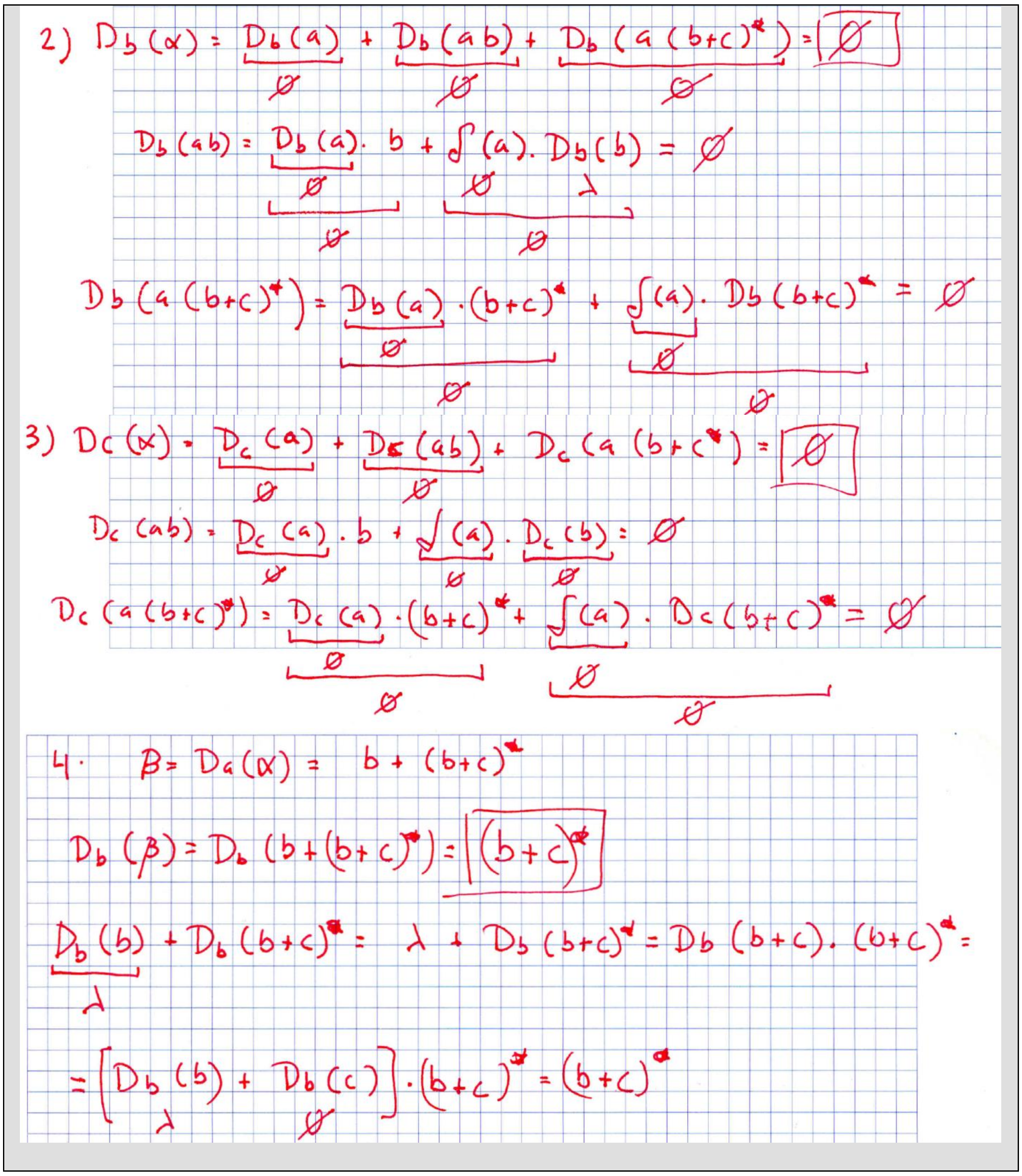


$\alpha = a + ab + a\,(b+c)^*$

1) $$D_a(\alpha) = \underbrace{D_a(a)}_{\lambda} + \underbrace{D_a(ab)}_{b} + \underbrace{D_a(a(b+c)^*)}_{(b+c)^*} = \boxed{b+(b+c)^*}$$

$$D_a(ab) = \underbrace{\underbrace{D_a(a)}_{\lambda}.\,b}_{b} + \underbrace{\underbrace{\delta(a)}_{\emptyset}.\,D_a(b)}_{\emptyset} = b$$

$$D_a(a(b+c)^*) = \underbrace{D_a(a)}_{\lambda}.\,(b+c)^* + \underbrace{\delta(a).\,D_a(b+c)^*}_{\emptyset} = (b+c)^*$$



2) $D_b(\alpha) = \underbrace{D_b(a)}_{\emptyset} + \underbrace{D_b(ab)}_{\emptyset} + \underbrace{D_b(a(b+c)^*)}_{\emptyset} = \boxed{\emptyset}$

$D_b(ab) = \underbrace{\underbrace{D_b(a)}_{\emptyset}\cdot b}_{\emptyset} + \underbrace{\underbrace{\delta(a)}_{\emptyset}\cdot \underset{\lambda}{D_b(b)}}_{\emptyset} = \emptyset$

$D_b(a(b+c)^*) = \underbrace{\underbrace{D_b(a)}_{\emptyset}\cdot(b+c)^*}_{\emptyset} + \underbrace{\underbrace{\delta(a)}_{\emptyset}\cdot D_b(b+c)^*}_{\emptyset} = \emptyset$

3) $D_c(\alpha) = \underbrace{D_c(a)}_{\emptyset} + \underbrace{D_c(ab)}_{\emptyset} + D_c(a(b+c^*) = \boxed{\emptyset}$

$D_c(ab) = \underbrace{D_c(a)}_{\emptyset}\cdot b + \underbrace{\delta(a)}_{\emptyset}\cdot\underbrace{D_c(b)}_{\emptyset} = \emptyset$

$D_c(a(b+c)^*) = \underbrace{\underbrace{D_c(a)}_{\emptyset}\cdot(b+c)^*}_{\emptyset} + \underbrace{\underbrace{\delta(a)}_{\emptyset}\cdot D_c(b+c)^*}_{\emptyset} = \emptyset$

4. $\beta = D_a(\alpha) = b + (b+c)^*$

$D_b(\beta) = D_b(b+(b+c)^*) = \boxed{(b+c)^*}$

$\underbrace{D_b(b)}_{\lambda} + D_b(b+c)^* = \lambda + D_b(b+c)^* = D_b(b+c)\cdot(b+c)^* =$

$= \left[\underset{\lambda}{D_b(b)} + \underset{\emptyset}{D_b(c)}\right]\cdot(b+c)^* = (b+c)^*$

Problem 2 (2 points)

We want to design a device for a diving chronometer that prevents its involuntary use. The device includes three buttons: *a*, *b* and *c*.

> The button *a* moves the pointer 10 minutes forward; the button *b* moves the pointer 20 minutes forward, and the *c*, 30 minutes.

To start the chronometer it is necessary to complete 60 minutes by means of pressing 3 buttons (the same or not) from the initial state (0 minutes).

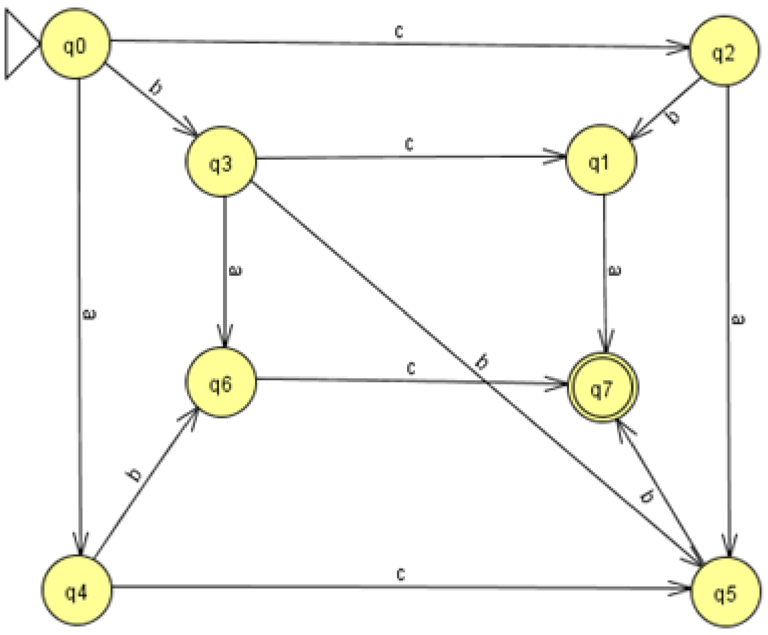
It is required:

a. Construct a Finite Automata that indicates that the pointer has been moved 60 minutes forward by pressing 3 buttons. Explain in detail.

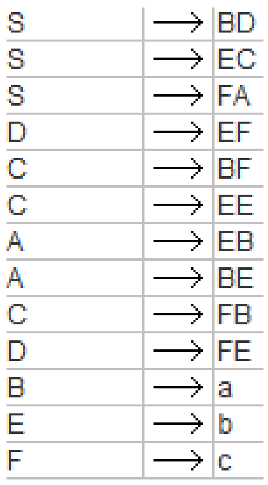
b. Obtain a grammar corresponding to the language accepted by the previous automaton, expressed in Chomsky Normal Form. Explain in detail.

Note: It is not required to define drain states into the designed FA.

The required DFA (without drain state) is the following:



The grammar in CNF corresponding to the language accepted by the DFA is:



S	$\longrightarrow$	BD
S	$\longrightarrow$	EC
S	$\longrightarrow$	FA
D	$\longrightarrow$	EF
C	$\longrightarrow$	BF
C	$\longrightarrow$	EE
A	$\longrightarrow$	EB
A	$\longrightarrow$	BE
C	$\longrightarrow$	FB
D	$\longrightarrow$	FE
B	$\longrightarrow$	a
E	$\longrightarrow$	b
F	$\longrightarrow$	c

Problem 3 (2 points)

Construct a Push-Down Automaton by empty stack to recognize the language L:

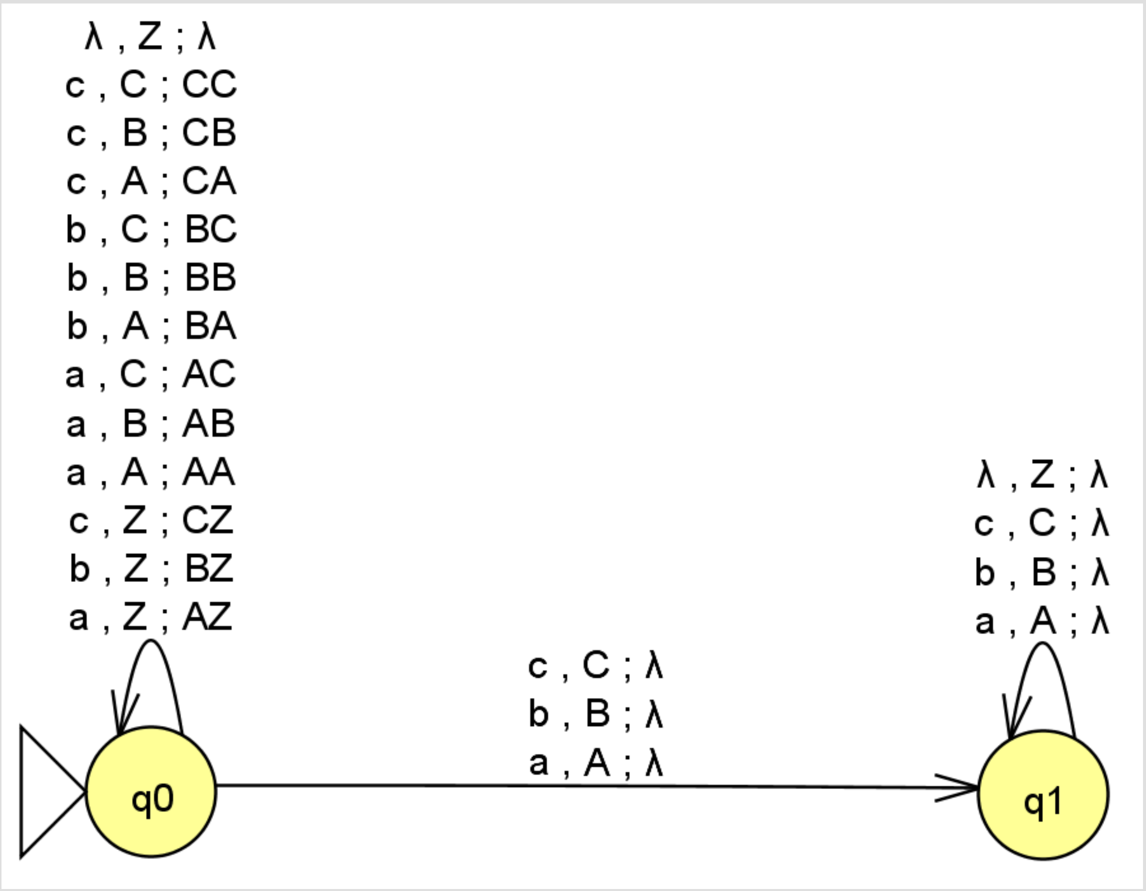
L = { ZZ^{-1} / Z = (a+b+c)* } where Z^{-1} represents the opposite order of the element in the expression Z.

(Example, the words *abccba, baab,* and *cc* are included in the language).

1. Describe formally the PDA_E, also detailing if it is a deterministic or a non-deterministic automaton and the reasons why.
2. Explain which is the shortest word accepted and show the acceptance of a word in L with length equal to 4.
3. Transform the PDA_E into an equivalent PDA_F, also showing the acceptance of the same word of the previous section by the latter automaton.

1. A PDA_E that recognizes the language L could be as follows:

PDA_E = ({a,b,c}, {Z,A,B,C},Z,{q0, q1}, f, ϕ), where f is:



It can be observed that this PDA_E is Non-Deterministic, given that there are several possible transitions from q0 for the same input and the same symbol on the top of the stack. These transitions are:

1. Input symbol to be read: a. Symbol on the top of the stack: A
2. Input symbol to be read: b. Symbol on the top of the stack: B
3. Input symbol to be read: c. Symbol on the top of the stack: C

In these cases, if the PDA is still reading the first half of the word (Z), it stores the corresponding symbol in the stack. If the PDA is reading the second half of the word (Z^{-1}), it removes the element on the stack (extract this element and λ is added). For this reason, it is not possible to develop a Determinist PDA_E for this language because it is not known when symbols of the first or second part of the word are being read. In addition, the PDA_E recognizes the empty word (λ), since it is also included in the language.

2. The shortest word accepted is λ.

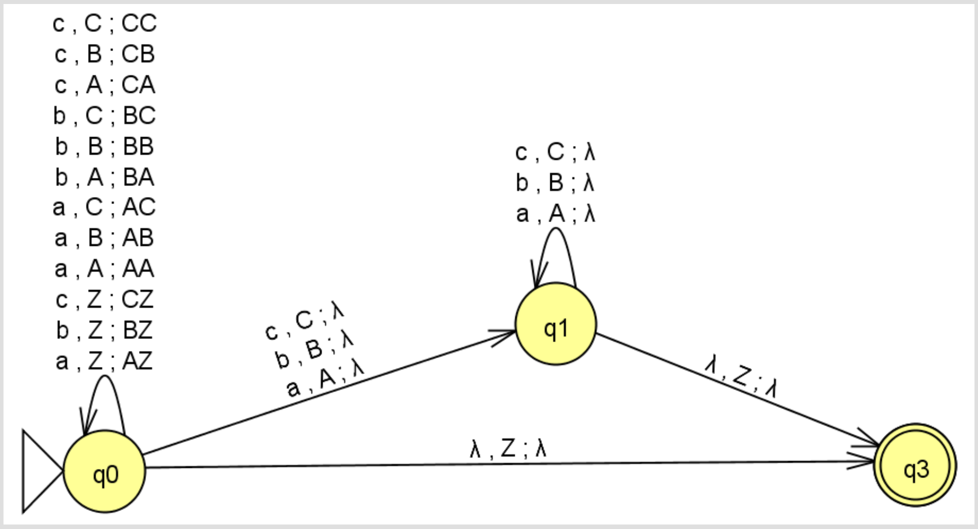
Acceptance of the word: acca (length 4):

Transition:	Element of the transition function used:
(q0, acca, Z) |-- (q0, cca, AZ)	f(q0, a, Z) = (q0, AZ)
(q0, cca, AZ) |-- (q0, ca, CAZ)	f(q0, c, A) = (q0, CA)
(q0, ca, CAZ) |-- (q1, a, AZ)	f(q0, c, C) = (q1, λ)
(q1, a, AZ) |-- (q1, λ, Z)	f(q1, a, A) = (q1, λ)
(q1, λ, Z) |-- (q1, λ, λ)	f(q1, λ, Z) = (q1, λ)

It can be observed that the first and second transitions incorporate the first two input symbols on the stack. The following two transitions extract the symbols on the stack. This way, the stack is empty once the input word has been completely read → **word accepted.**

2. To transform the PDA_E into its equivalent PDA_F, the corresponding algorithm incorporates a final state and the corresponding transitions as follows:

PDA_F = ({a,b,c}, {Z,A,B,C},Z,{q0, q1}, f, q3), where f:



Acceptance of the word: acca (length 4):

Transition:	Element of the transition function used:
(q0, acca, Z) |-- (q0, cca, AZ)	f(q0, a, Z) = (q0, AZ)
(q0, cca, AZ) |-- (q0, ca, CAZ)	f(q0, c, A) = (q0, CA)
(q0, ca, CAZ) |-- (q1, a, AZ)	f(q0, c, C) = (q1, λ)
(q1, a, AZ) |-- (q1, λ, Z)	f(q1, a, A) = (q1, λ)
(q1, λ, Z) |-- (q2, λ, λ)	f(q1, λ, Z) = (q2, λ)

Problem 4 (2 points)

Construct a Turing Machine to enumerate the complete set of binary numbers in the tape. A blank symbol (□) must be used as a separator between each one of the numbers. The Turing Machine begins with a 0 in the tape and completes the list of numbers from right to left, i.e., the contents in the tape will be the following successively:

□0□ → □1□0□ → □10□1□0□ → □11□10□1□0□ → …

where □ represents an empty cell in the tape.

The designed TM must be explained in detail.

Note: The operation of the TM could be considered as follows:

1. Copy the current number at the left.
2. Increase this number by one unit.
3. Repeat the process.

As it is described in the problem, the required Turing Machine could follow these steps:

Step 1: Copy the current number at the left.

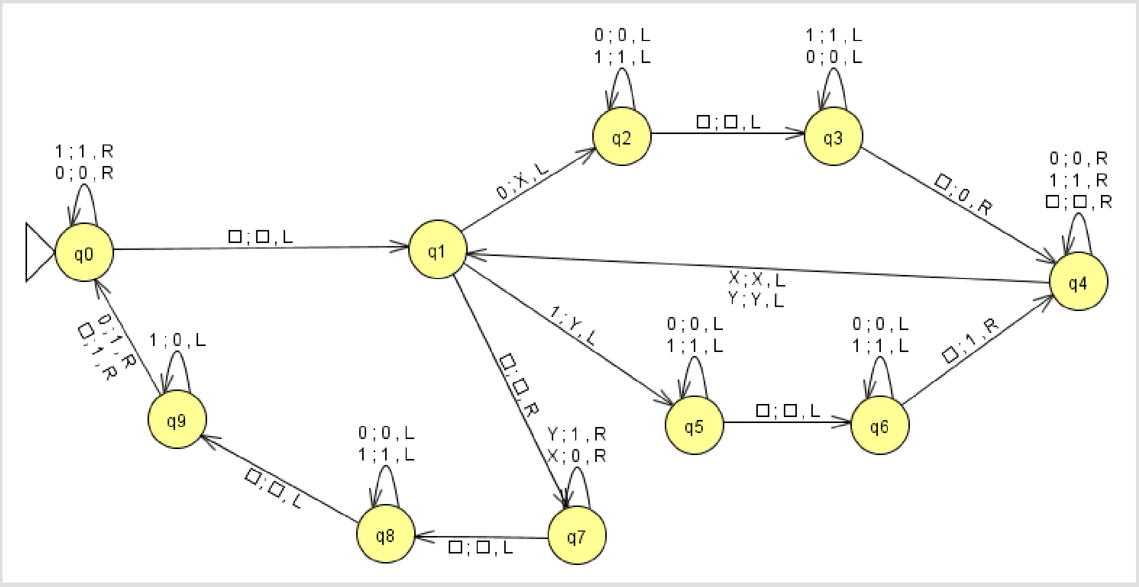
- Take the right-most position of the current number.
- If this digit is a 0, replace it with an X. If it is a 1, replace it with a Y.
- Go to the left and copy it (0 or 1).
- Repeat the previous process until the number has been completely read.
- Recover the initial number (transform Xs' s into 0' s and Y's into 1's).

Step 2: Increase this number by one unit.

- Take the right-most position of the current number.
- If this digit is a 0, replace it with a 1 and the process is finished.
- If this digit is a 1, replace it with a 0 and go to the bit at the left.
- Repeat this process until finding a 0 or the blank at the left, that would be transformed into a 1.

Step 3: Repeat the previous steps.

A Turing Machine that follows these steps is:



The states q1, q2, q3, q4, and q5 copy the current binary number to the left. The states q1, q7, q8, q9 and q0 restore the initial number (transform Xs' into 0's and Y’s into 1's) and increase by one unit the copied number. As it can be observed, it is a TM that never stops (i.e, final states are not included).