Máquina de Turing

A continuación presentamos la última máquina abstracta que estudiamos en la asignatura: la *Máquina de Turing*. Pero antes haremos una relación entre las máquinas para comprender sus características, recursos y relación entre ellas.

En la unidad 1 se presentó la jerarquía de máquinas abstractas donde se puede apreciar que a partir del Autómata Finito incorporando recursos a cada máquina, éstas incrementan su capacidad de cómputo. Hasta este momento hemos estudiado dos máquinas Abstractas a nivel práctico: Autómatas Finitos y Autómatas con Pila, de las cuales el AF aumenta su capacidad de cómputo al incorporase una memoria de tipo LIFO convirtiéndose en un AP.

También, si un Autómata Finito posee la capacidad de poder decidir sobre el movimiento del cabezal (*izquierda (I), neutro (N), derecha (D) y la parada de la máquina (P)*) se convierte en un *Autómata Finito Determinista Bidireccional* (AFDB).

Un *AFDB* se define como:

$$AFDB = (\Sigma_E, \Gamma, Q, q_0, A, f)$$

$$\text{dónde: } \Gamma = \Sigma_E \cup \{\vdash, \dashv\} \text{ y donde } f: Q \times \Gamma \rightarrow Q \times \{I, N, D, P\}$$

La cinta de entrada presenta límites que se definieron en un nuevo alfabeto de cinta Γ, formado por el alfabeto de entrada Σ_E y dos símbolos especiales destinados a demarcar los extremos del medio de lectura ($\vdash$ y $\dashv$). Así, una cadena α a ser procesada es representada en la cinta por $\vdash\alpha\dashv$. Estos símbolos se los conoce como **BOT** (*Begin Of Tape*) y **EOT** (*End Of Tape*).

Los AFBD no son parte del estudio de la asignatura a nivel práctico pero son la base para las nuevas máquinas que se presentan en esta unidad.

Si le permitimos al Autómata Finito Bidireccional la posibilidad de grabar sobre la cinta se obtiene una nueva máquina abstracta llamada ***Autómata Linealmente Acotado* (*ALA*)**. La capacidad de grabar implica la ampliación del alfabeto de cinta Γ con la incorporación de símbolos auxiliares reunidos en un nuevo alfabeto Ω.

El *Autómata Linealmente Acotado* queda definido como:

$$ALA = (\Sigma_E, \Gamma, Q, q_0, A, f)$$

$$\text{dónde: } \Gamma = \Sigma_E \cup \{\vdash, \dashv\} \cup \Omega \text{ y donde } f: Q \times \Gamma \rightarrow Q \times \Gamma \times \{I, N, D, P\}$$

Los componentes de la *MT* son los mismos del *ALA*, con la única excepción del alfabeto de cinta, que debe ser redefinido ya que se eliminaron las marcas de inicio y fin de cinta. Además, debe observarse que en la definición de la máquina de Turing se incluye el símbolo $\cancel{b}$, que por defecto ocupa el resto de la cinta no utilizada.

Una *Máquina de Turing* queda definido como:

$$MT = (\Sigma_E, \Gamma, Q, q_0, A, f, \cancel{b})$$

Σ_E **:** Alfabeto de símbolos de entrada

Γ **:** Alfabeto de cinta, $\Gamma = \Sigma_E \cup \Omega \cup \{\cancel{b}\}$

Q **:** Conjunto finito, no vacío, de estados posibles

q_0 **:** Estado inicial de operación, $q_0 \in Q$

A **:** Conjunto de estados de aceptación, $A \subseteq Q$

f **:** Función de transición:

$$f\colon \overbrace{Q \times \Gamma}^{\text{Estado actual y lectura del símbolo de la cadena de entrada}} \rightarrow \underbrace{Q \times \Gamma \times \{I, N, D, P\}}_{\text{Próximo Estado, grabación en la cinta y movimiento del cabezal para la lectura de próximo símbolo de la cadena}}$$

Cómo se puede apreciar, si a un ALA se le suprimen las cotas de la cadena se permite una cinta infinita y se convierte en una Máquina de Turing.

En el diseño de Autómatas Finitos nos centrábamos solo en las transiciones de estados y la lectura de la cadena. Ahora con la MT no solo tenemos que determinar el próximo estado si no que debemos conocer qué grabar en la cinta y cuál es el próximo símbolo a leer en la cadena de entrada, son nuevos recursos que hay que aprender a gestionar.

Representación de las transiciones

Las transiciones en una MT en un grafo se definen de la siguiente manera:

$$\underbrace{\mathbf{a}}_{\text{1era Parte}}\ /\ \overbrace{\mathbf{b,\ m}}^{\text{2da Parte}}$$

Con **a** y **b** $\in$ **Γ*** y **m**$\in\{I, N, D, P\}$

<u>1era Parte:</u> se corresponde con la lectura de un símbolo de la cadena de entrada, pertenece al alfabeto de cinta porque se puede leer $\Gamma = \Sigma_E \cup \Omega \cup \{\not b\}$, es decir, un símbolo que pertenece al alfabeto de entra, o un blanco en la cinta, o un símbolo auxiliar.

<u>2da Parte:</u> en este momento definimos el símbolo a grabar sobre la cinta que puede ser el mismo símbolo que se leyó u otro distinto (un símbolo que pertenece al alfabeto de entrada, o un blanco en la cinta, o un símbolo auxiliar) y determinamos el movimiento del cabezal, estos pueden ser:

I (izquierda): próximo símbolo a leer es el que se encuentra inmediatamente a la izquierda del símbolo actual donde se encuentra posicionado el cabezal

D (derecha): próximo símbolo a leer es el que se encuentra inmediatamente a la derecha del símbolo actual donde se encuentra posicionado el cabezal

N (neutro): próximo símbolo a leer es el mismo en el que se encuentra actualmente

P (parada): no hay más movimiento de cabezal, se utiliza al finalizar el autómata

Actividades Prácticas

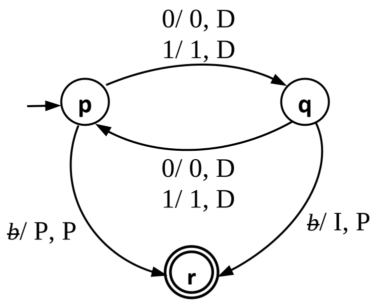
Nota: se aconseja construir todos los ejercicios en el simulador JFLAP y comprobar el funcionamiento de las máquinas para una mejor comprensión. Además, se presentan en cada ejercicio una solución, no es la única, si usted ha creado otro diseño de máquina y tiene dudas al respecto, consulte con su profesor de práctico.

Ejercicios propuestos de Máquinas de Turing (MT)

Ejercicio 6

Comprobar que la función que cumple la siguiente máquina de Turing es verificar si una cadena formada por bits, es de longitud par o impar. Al finalizar graba una **P** si la cadena resultó ser par o una **I** en caso contrario. La cadena no tiene longitud fija.

Solución



f	0	1	ƀ
→p	q,0,D	q,1,D	r,P,P
q	p,0,D	p,1,D	r,I,P
*r			

Figura 6.24 y tabla 6.12: Grafo y función de transición de la máquina de Turing del ejercicio 6.

MT = ({0,1}, {0,1,I,P,ƀ}, {p,q,r}, p, {r}, f, ƀ)

A continuación vamos a comprobar dos cadenas, una par y una impar para que puedan apreciar el estado en que se encuentra en cada lectura de símbolo, el movimiento del cabezal y la grabación en la cinta:

Cadena PAR α = 101101

00 | ƀ 1 0 1 1 0 1 ƀ |
p

01 | ƀ 1 0 1 1 0 1 ƀ |
q

02 | ƀ 1 0 1 1 0 1 ƀ |
p

03 | ƀ 1 0 1 1 0 1 ƀ |
q

04 | ƀ 1 0 1 1 0 1 ƀ |
p

05 | ƀ 1 0 1 1 0 1 ƀ |
q

06 | ƀ 1 0 1 1 0 1 ƀ |
p

07 | ƀ 1 0 1 1 0 1 ***P*** ƀ |
r

Cadena IMPAR α = 00110

00 | ƀ 0 0 1 1 0 ƀ |
p

01 | ƀ 0 0 1 1 0 ƀ |
q

02 | ƀ 0 0 1 1 0 ƀ |
p

03 | ƀ 0 0 1 1 0 ƀ |
q

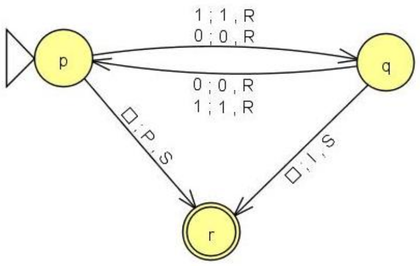
04 | ƀ 0 0 1 1 0 ƀ |
p

05 | ƀ 0 0 1 1 0 ƀ |
q

06 | ƀ 0 0 1 1 0 ***I*** ƀ |
r

En este ejercicio el movimiento del cabezal siempre fue hacia la derecha ya que no necesita realizar otro movimiento pero aprendimos a grabar símbolos en la cinta, los 0(ceros) y 1(unos) no se modificaron pero se incorporaron dos símbolos auxiliares **Ω = {P, I}**.

Veamos el grafo construido usando JFLAP, observe que utiliza un punto y coma para separar las partes de la transición, el símbolo blanco se representa por un cuadrado vacío y los movimiento están escritos en inglés (Right, Left, Stop).



MT = ({0,1}, {0,1,I,P,ƀ}, {p,q,r}, p, {r}, *f*, ƀ)

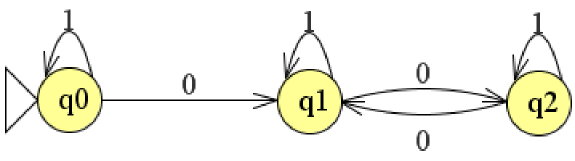
Ejercicio 7

Construir una *MT* que verifique en una cadena formada por bits, si la cantidad de **0s** (ceros) es par o impar y grabe la letra **P** si es par o **I** si es impar, al finalizar la cadena. La cadena no tiene longitud fija.

↓
Ejemplo: Entrada ƀ0100110ƀ
Salida ƀ0100110**P**ƀ

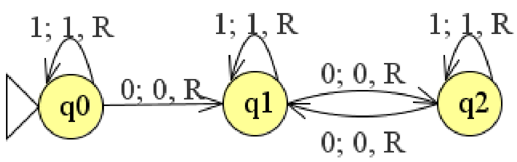
Solución Ejercicio 7

Este tipo de lenguaje a reconocer ya ha sido parte de estudio en otras máquinas como AF por ejemplo y la lógica es la misma: solo debo transitar con la lectura de 0 (ceros) para poder determinar la paridad de los mismos:

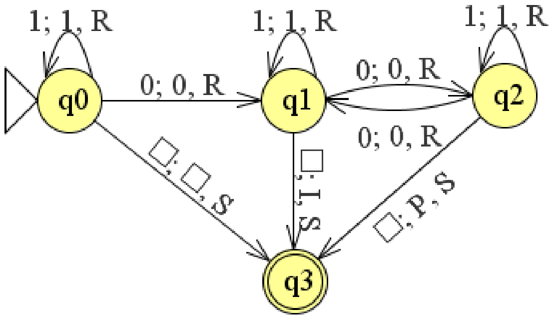


La MT tiene otros recursos que debemos analizar: símbolo que grabamos en la cinta y el próximo símbolo a leer. Con respecto a la grabación de la cinta al igual que el ejercicio anterior los 0(ceros) y los 1(unos) no se modifican y se incorporan dos símbolos auxiliares **Ω = {P, I}**. Con respecto al movimiento del cabezal solo necesitamos recorrer toda la cadena hacia la derecha, por lo tanto el movimiento siempre será hacia la derecha.

Con todo lo analizado hasta ahora, comencemos a diseñar la MT:



Pero obviamente no está completo, la MT tiene que grabar una I si la cadena es impar o una P si la cadena es par y también el estado de aceptación:



MT = ({0,1}, {0, 1, P, I, ƀ} {q0,q1,q2,q3}, q0, {q3}, f, ƀ)

La lectura de la cadena puede finalizar en los tres estados:

- Si la lectura de la cadena finaliza en el estado **q0** significa que nunca transitó a otro estado porque la cadena solo está formada por símbolos 1(unos)
- Si la lectura de la cadena finaliza en el estado **q1** significa que la cantidad de ceros es impar y transita al estado q3 grabando una **I** sobre el primer blanco.
- Si la lectura de la cadena finaliza en el estado **q2** significa que la cantidad de ceros es par y transita al estado q3 grabando una **P** sobre el primer blanco.

Ejercicio 8

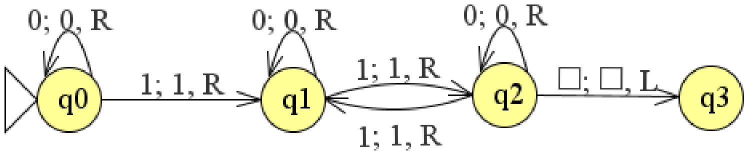
Diseñar una *MT* que verifique en una cadena formada por bits, si la cantidad de **1s** (unos) es par o impar. En caso de que sea par, al finalizar la comprobación, negar la cadena (cambiar ceros por unos y viceversa), en caso contrario, la máquina solo se detiene. La cadena no tiene longitud fija.

↓
Ejemplo: Entrada: ƀ101011ƀ
Salida: ƀ**010100**ƀ (cadena negada)

Solución Ejercicio 8

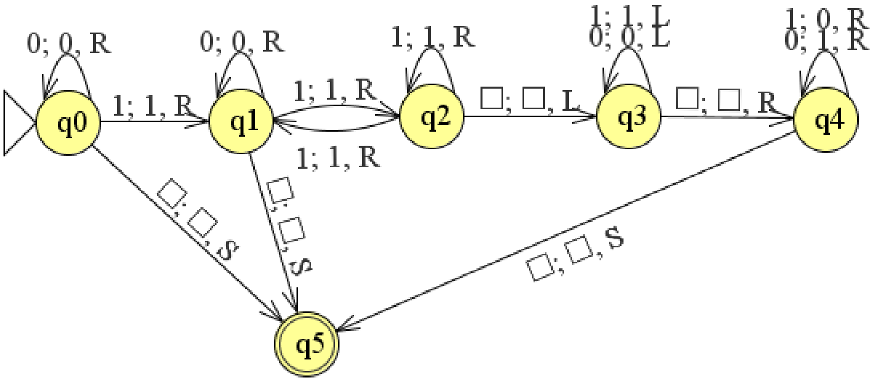
Con respecto a la verificación de la paridad de 1(unos) es similar al ejercicio anterior solo cambiando los 0(ceros) por 1(uno), pero ya no se pide grabar los símbolos auxiliares de par o impar ahora se debe ***negar*** la cadena si la cantidad de 1(unos) es par, esto significa que al terminar de recorrer toda la cadena para verificar la paridad se debe volver a leer la cadena para cumplir con la negación.

Primer paso:



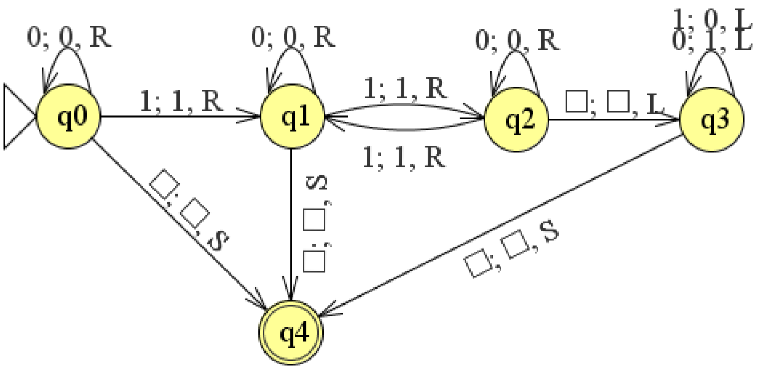
Cuando se verifica la paridad par de 1(unos), desde el estado q2 al estado q3 se lee el primer blanco que significa que se terminó de leer la cadena entonces incorporamos un movimiento a la izquierda para poder volver a leer la cadena, se pude resolver de dos maneras:

- mover el cabezal hacia la izquierda por cada símbolo que lea hasta leer el primer blanco, significa que recorrimos toda la cadena y comenzar a negar la cadena desde el primer símbolo hacia la derecha hasta que llegue al primer blanco al final de la cadena:



MT = ({0, 1}, {0, 1, ~~b~~} {q0, q1, q2, q3, q4, q5}, q0, {q5}, f, ~~b~~)

- a medida que vuelvo a la izquierda voy negando la cadena, en este caso la máquina finaliza su trabajo en el primer blanco a la izquierda:



MT = ({0, 1}, {0, 1, ~~b~~} {q0,q1,q2,q3,q4}, q0, {q4}, f, ~~b~~)

Presentamos dos soluciones viables para un mismo problema y ambas son correctas.

Ejercicio 9

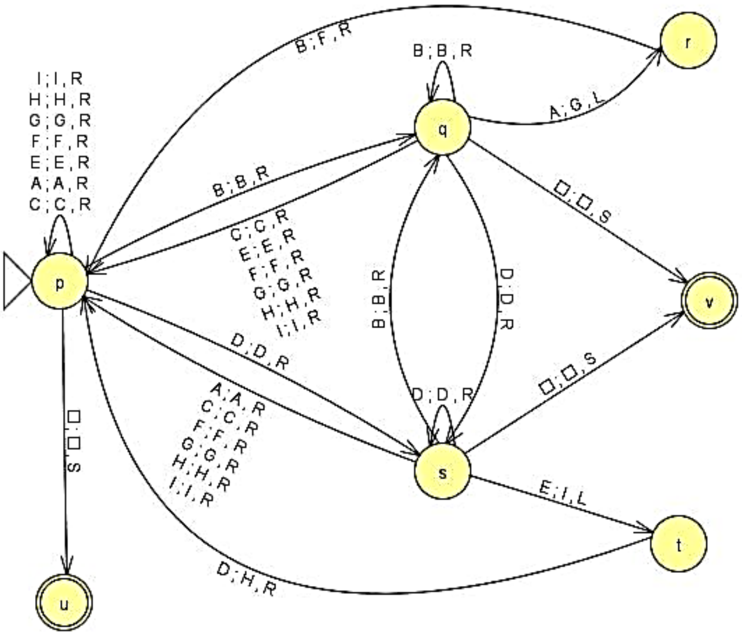
Construir una *MT* que opere sobre el alfabeto **Σ = {A, B, C, D, E, F, G, H, I}** y sustituya las subtiras en una cadena dada según se indica:

BA por **FG** y **DE** por **HI**

↓

Ejemplo: Entrada ~~b~~CFH**BA**AG**DE**~~b~~

Salida ~~b~~CFH**FG**AG**HI**~~b~~



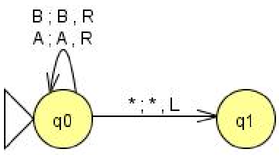
MT=({A,B,C,D,E,F,G,H,I},{ A,B,C,D,E,F,G,H,I,~~b~~},{p,q,r,s,t,u,v},p,{u,v},f,~~b~~)

Ejercicio 10

Diseñar una *MT* que cree la imagen refleja de la entrada después de un ***** (asterisco), dejando inalterada al finalizar la cadena original.

↓
Ejemplo: Entrada ~~b~~ABB*~~b~~
Salida ~~b~~ABB***BBA**~~b~~

Esta MT ejecutora debe copiar la imagen refleja a la derecha de la misma, el cabezal como en toda máquina abstracta se encuentra ubicado en el primer símbolo de la cadena y no tiene longitud fija, si queremos comenzar a copiar desde el primer símbolo debo copiarlo a la derecha del *(asterisco) *pero en qué posición de la cinta?* desconozco la longitud de la cadena y como estamos creando la imagen refleja, éste símbolo debe ser el último de la nueva cadena, por lo tanto debo empezar por el último símbolo de la cadena que va a ser el primero de la nueva cadena a la derecha del *(asterisco), por lo tanto como primer paso debo mover el cabezal hacia la derecha hasta el *(asterisco) y volver un lugar a la izquierda de esta manera dejo posicionado el cabezal sobre el último símbolo de la palabra que será el primera símbolo que voy a crear:

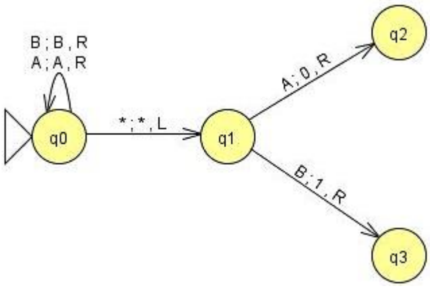


El símbolo que puedo leer es una A o una B recordar que siempre se diseña una máquina abstracta para un lenguaje:

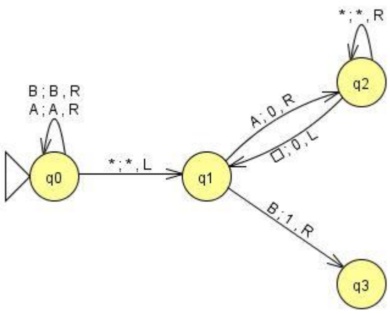
~~b~~ B B A A * ~~b~~ ~~b~~ ~~b~~ ~~b~~

q1

Este símbolo debe ser copiado en el primer lugar después del *(asterisco) por lo tanto debemos mover el cabezal hacia la derecha hasta el primer blanco y en esa posición de la cinta grabar; el símbolo *(asterisco) separa la cadena original de la nueva cadena, ahora bien cuando grabe el símbolo debe volver a la izquierda para seguir copiando, este mecanismo la MT lo va a realizar tantas veces como símbolos tenga la cadena, ahora bien, *¿cómo sabe la máquina qué símbolo ya fue grabado?* La única forma es marcar con un símbolo auxiliar cada vez que copie un símbolo de la cadena por ejemplo los símbolos A los marco con un 0(cero) y los símbolos B con un 1(uno):



Los símbolos se deben grabar al final de la cadena por lo tanto el cabezal va hacia la derecha hasta el primer blanco que encuentre y graba, ahora bien, *¿qué símbolo graba?* Puede grabar el símbolo original (ceros o unos) o símbolos auxiliares, pero hay que tener cuidado al volver y debe cambiarse de estado cuando lee un *(asterisco) o la forma más simple es que el cabezal lea siempre el mismo símbolo, en este caso el auxiliar:



En el estado q2 por ahora se encuentra solo con el *(asterisco) e inmediatamente con el primer blanco y graba el símbolo auxiliar 0:

ƀ B B A 0 * ƀ ƀ ƀ ƀ
q2

ƀ B B A 0 * ƀ ƀ ƀ ƀ
q2

ƀ B B A 0 * 0 ƀ ƀ ƀ
q1

Y debe volver el cabezal a la izquierda hasta encontrar el próximo símbolo a copiar. Siguiendo la cadena de ejemplo va a volver a leer el *(asterisco), luego el símbolo auxiliar 0(cero) y a la izquierda el próximo símbolo a copiar que en este caso es una A (pero podría haber sido una B):

ƀ B B A 0 * 0 ƀ ƀ ƀ
q1

ƀ B B A 0 * 0 ƀ ƀ ƀ
q1

ƀ B B 0 0 * 0 ƀ ƀ ƀ
q2

Y la MT va a volver a transitar por los mismos estados copiando el siguiente símbolo:

ƀ B B 0 0 * 0 ƀ ƀ ƀ
q2

ƀ B B 0 0 * 0 ƀ ƀ ƀ
q2

ƀ B B 0 0 * 0 ƀ ƀ ƀ
q2

~~b~~ B B 0 0 * 0 0 ~~b~~ ~~b~~
q1

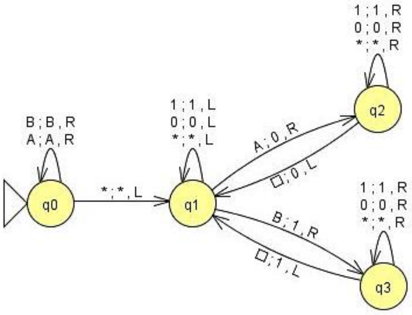
~~b~~ B B 0 0 * 0 0 ~~b~~ ~~b~~
q1

~~b~~ B B 0 0 * 0 0 ~~b~~ ~~b~~
q1

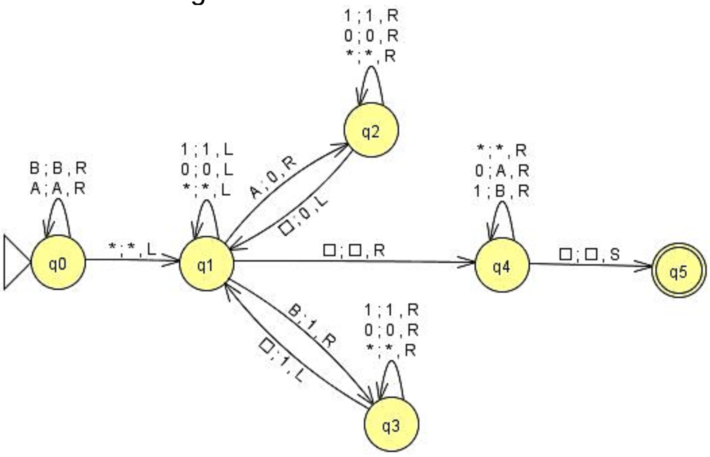
~~b~~ B B 0 0 * 0 0 ~~b~~ ~~b~~
q1

~~b~~ B B 0 0 * 0 0 ~~b~~ ~~b~~
q1

Exactamente igual debe realizarse con la lectura de símbolos B:

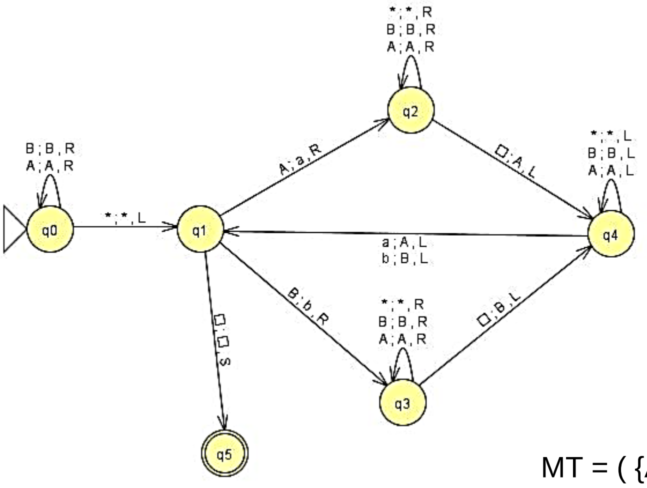


Cuando finalice de copiar todos los símbolos de la cadena, el cabezal encontrará el primer blanco a la izquierda de la cadena en este momento volvemos el cabezal hacia la derecha volviendo a dejar la cadena con sus símbolos originales como así también la nueva cadena:



MT = ({A,B,*}, {A,B, 0, 1, *, ~~b~~}, {q0,q1,q2,q3,q4,q5}, q0, {q5}, f, ~~b~~)

A continuación proponemos otra solución para el mismo problema:

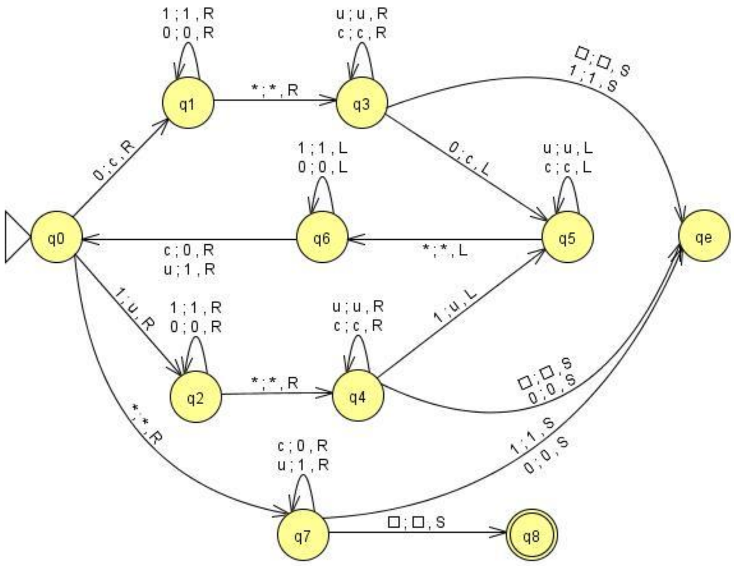


MT = ({A,B,*}, {A,B, a, b, *, ~~b~~}, {t,p,q,r,s,z}, t, {z}, *f*, ~~b~~)

Ejercicio 11

Diseñar una *MT* que verifique si dos palabras separadas por un ***** (asterisco) son iguales (en este caso, acepta la cadena de entrada).

↓
Ejemplo: ƀ**0010*****0010**ƀ

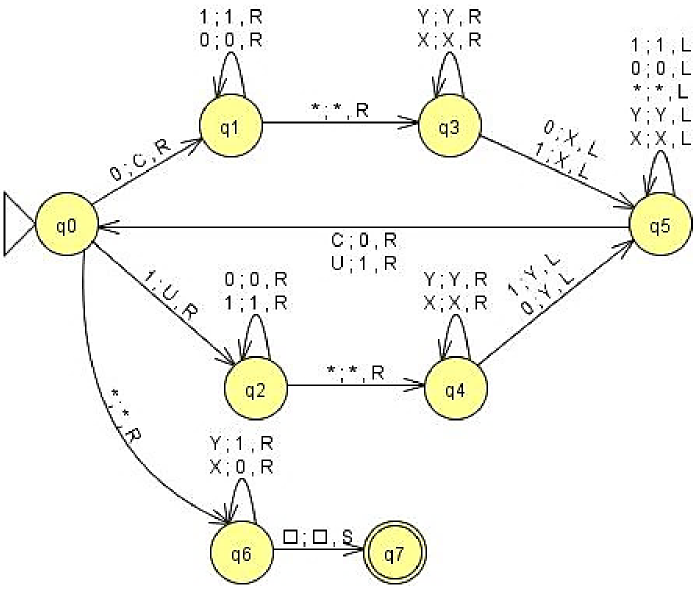


MT = ({0,1,*}, {c, u, 0, 1, *, ƀ}, {q0,q1,q2,q3,q4,q5,q6,q7,q8,qe}, q0, {q8}, f, ƀ)

Ejercicio 12

Diseñar una *MT* que copie el primer nibble de un byte sobre el segundo, que se encuentra separado por un ***** (asterisco).

↓
Ejemplo: Entrada ƀ**1001***0011ƀ
Salida ƀ1001***1001**ƀ



MT = ({0,1,*}, {C, U, X, Y, 0, 1, *, ƀ}, {q0,q1,q2,q3,q4,q5,q6,q7}, q0, {q7}, f, ƀ)

Ejercicio 13

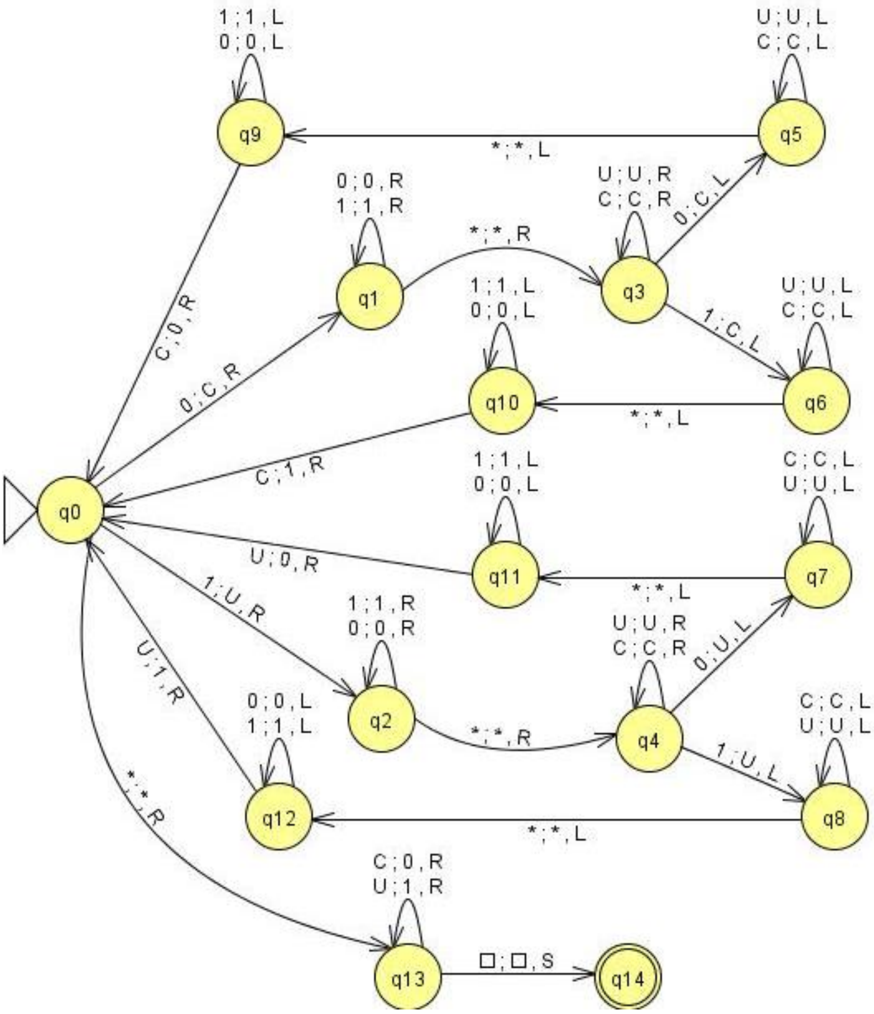
Construir una *MT* que intercambie el primer nibble por el segundo, estando ambos separados por un ***** (asterisco).

↓

Ejemplo: Entrada ~~b~~1001*0011~~b~~
Salida ~~b~~**0011***1001~~b~~

Vamos a proponer dos estrategias distintas para la solución:

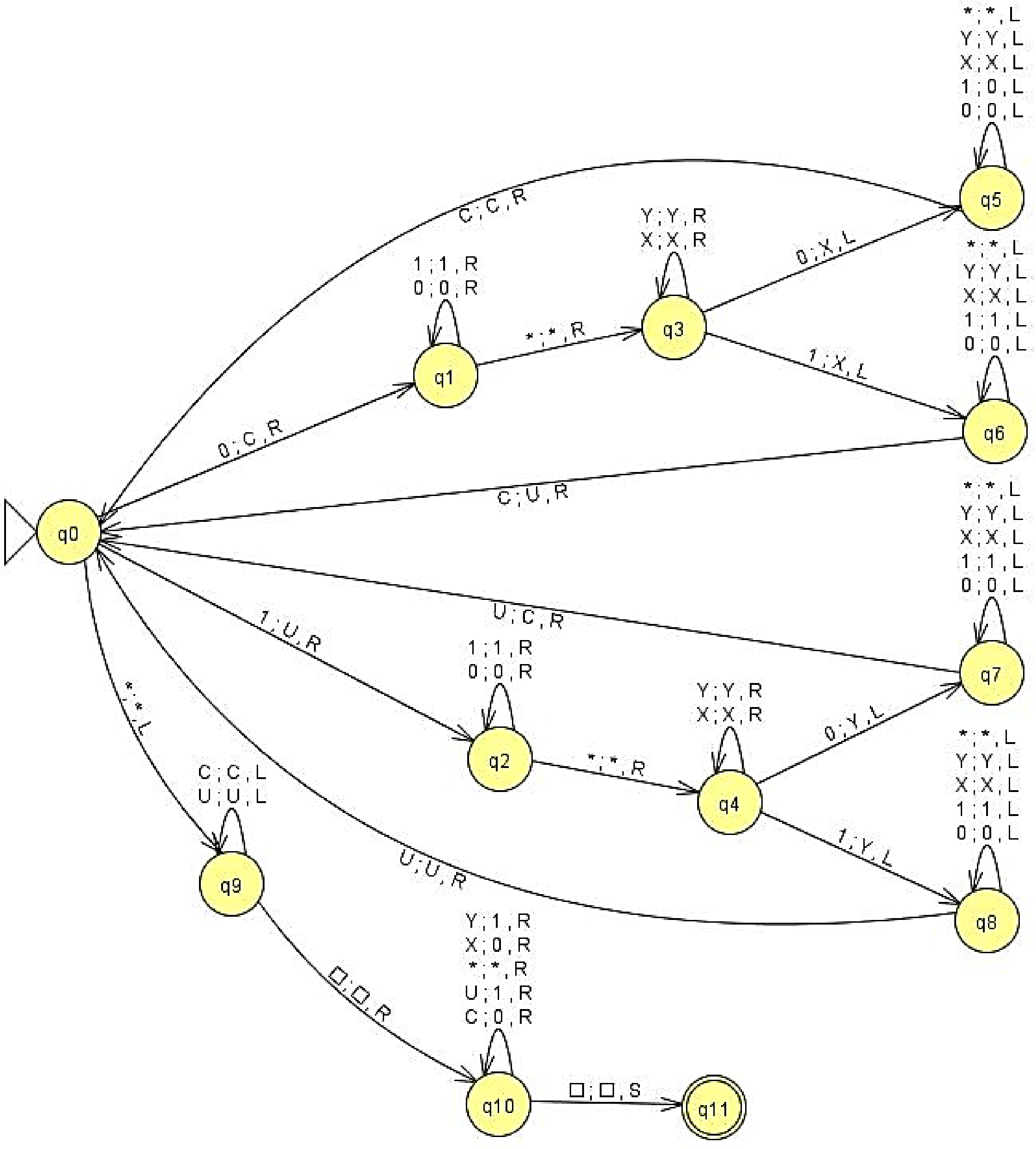
Estrategia 1: Con 2 símbolos auxiliares



MT = ({0,1,*}, {C,U,0,1,*,~~b~~}, {q0,q1,q2,q3,q4,q5,q6,q7,q8,q9,q10,q11,q12,q13,q14}, q0, {q14}, f, ~~b~~)

Atención: q5 y q7 pueden ser un mismo estado
q6 y q8 pueden ser un mismo estado

Estrategia 2: Con 4 símbolos auxiliares:



MT = ({0,1,*}, {C,U,X,Y,0,1,*,ƀ}, {q0,q1,q2,q3,q4,q5,q6,q7,q8,q9,q10,q11}, q0, {q11}, f, ƀ)

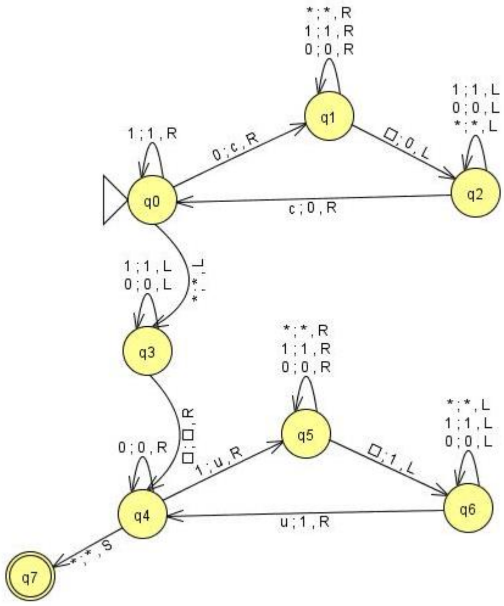
Ejercicio 14

Construir una *MT* que reordene en orden ascendente una cadena de bits de cualquier longitud y la grabe sobre la cinta a partir de un ***** (asterisco).

↓
Ejemplo: Entrada ~~b~~0110100*~~b~~
Salida ~~b~~0110100***0000111**~~b~~

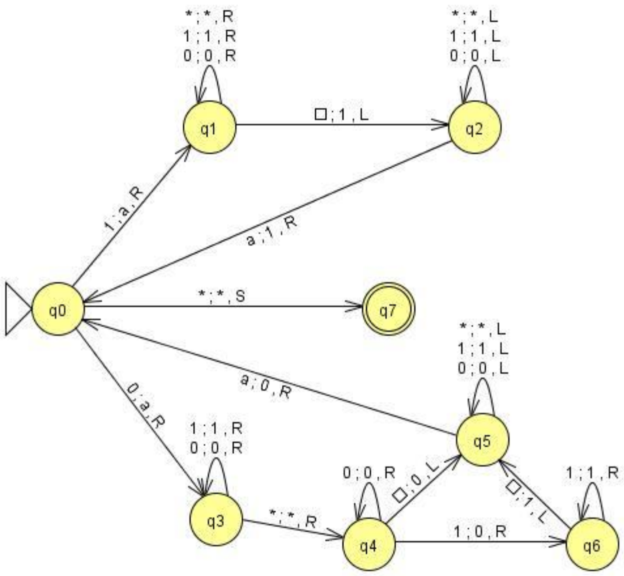
Vamos a proponer dos estrategias distintas para la solución:

Estrategia 1: Efectúo 2 recorridos de la cadena. En el primero grabo los ceros a la derecha del * (asterisco) y en el segundo grabo los unos a la derecha de los ceros.



MT = ({0,1,*}, {C,U,0,1,*,~~b~~}, {q0,q1,q2,q3,q4,q5,q6,q7}, q0, {q7}, f, ~~b~~)

Estrategia 2: Grabo los ceros y los unos en la medida que van apareciendo, en la posición correcta.

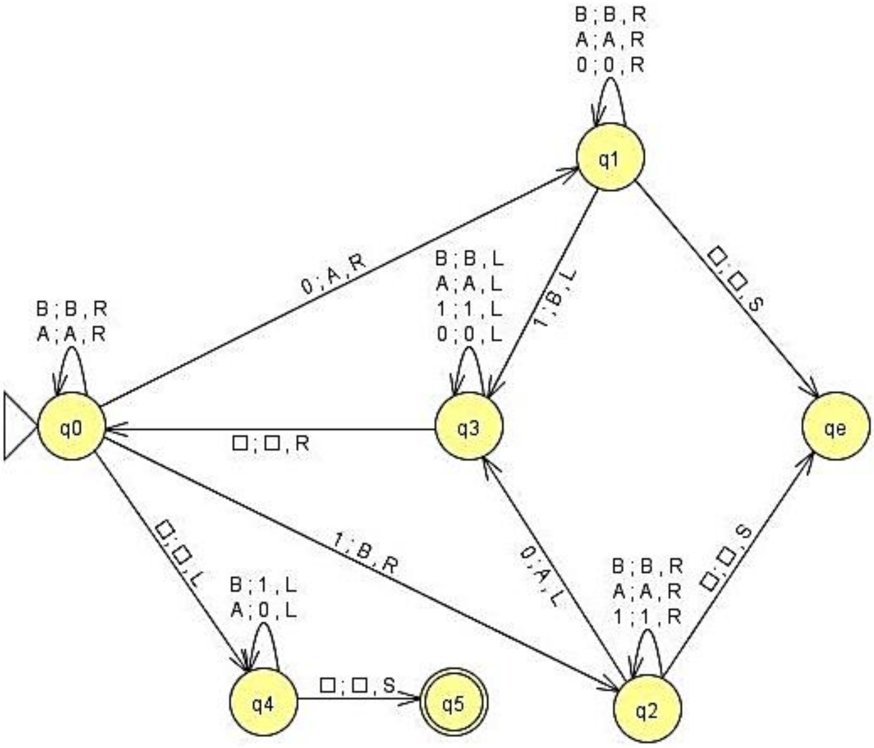


MT = ({0,1,*}, {a,0,1,*,~~b~~}, {q0,q1,q2,q3,q4,q5,q6,q7}, q0, {q7}, f, ~~b~~)

Ejercicio 15

Construir una *MT* que verifique en una cadena formada por bits, que haya la misma cantidad de **0s** (ceros) que de **1s** (unos). La cadena no tiene longitud fija.

↓
Ejemplo: ƀ**00100111**ƀ (la misma cantidad de **0** que de **1**)

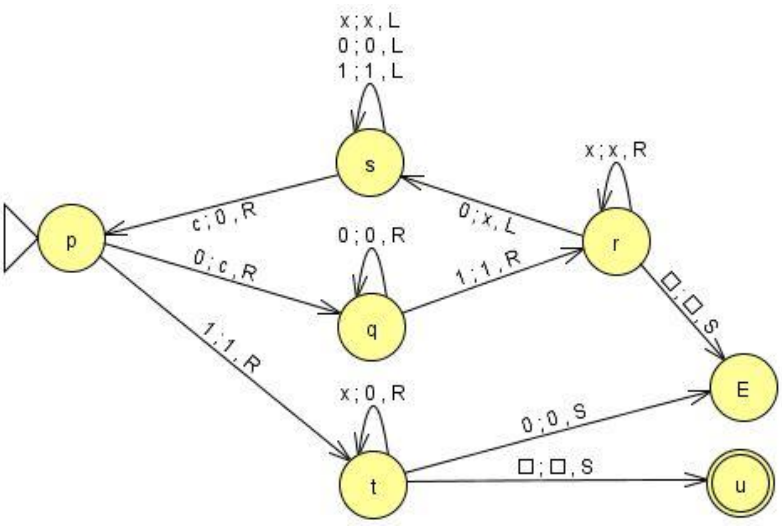


MT = ({0,1,*}, {A,B,0,1,ƀ}, {q0,q1,q2,q3,q4,q5,qe }, q0, {q5}, f, ƀ)

Ejercicio 16

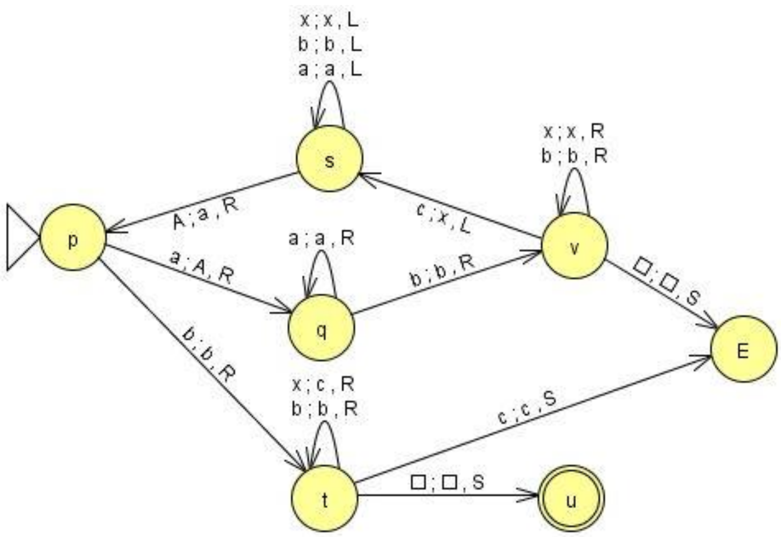
Construir una MT para reconocer cada uno de los lenguajes presentados en el ejercicio 4 del Capitulo5 (autómatas con pila).

a) L1 = { 0^n 1 0^n / n>=1 , Σ = {0,1}} Ejemplo de cadena aceptada α = ƀ00100ƀ



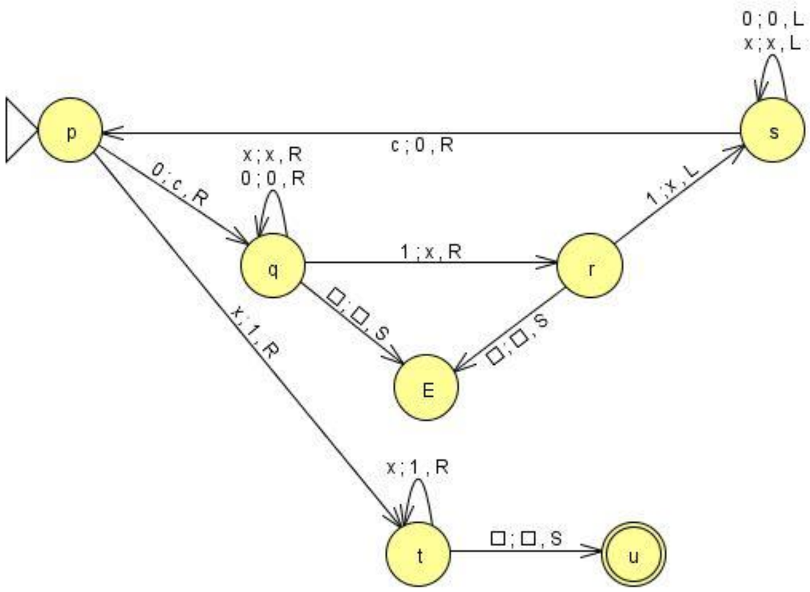
MT16a= ({0,1} , {0,1,c,x, ƀ }, {p,q,r,s,t,u,E }, p, {u}, *f*, ƀ)

b) L2 = { $a^p\ b^n\ c^p$ / p>= 1 , n>=1 , Σ = {a,b,c}} Ejemplo de cadena aceptada α = ~~**b**~~aaabbccc~~**b**~~



MT16b= ({a,b,c} , {a, b, c, A, x, ~~*b*~~ }, {p, q, t ,s, v, E ,u }, p, {u}, *f*, ~~*b*~~)

c) L3 = { $0^n\ 1^{2n}$ / n>=1 , Σ = {0,1}} Ejemplo de cadena aceptada α = ~~*b*~~001111~~*b*~~



MT16c= ({0,1} , {0, 1, c, x, ~~*b*~~ }, {p, q, r ,s, t, E ,u }, p, {u}, *f*, ~~*b*~~)