

Teoría de Autómatas y Lenguajes Formales

Ejercicios de Máquinas de Turing

Autores:

Araceli Sanchis de Miguel
Agapito Ledezma Espino
Jose A. Iglesias Martínez
Beatriz García Jiménez
Juan Manuel Alonso Weber

* Algunos ejercicios están basados en enunciados de los siguientes libros:

- Enrique Alfonseca Cubero, Manuel Alfonseca Cubero, Roberto Moriyón Salomón. *Teoría de autómatas y lenguajes formales*. McGraw-Hill (2007).
- Manuel Alfonseca, Justo Sancho, Miguel Martínez Orga. *Teoría de lenguajes, gramáticas y autómatas*. Publicaciones R.A.E.C. (1997).
- Pedro Isasi, Paloma Martínez y Daniel Borrajo. *Lenguajes, Gramáticas y Autómatas. Un enfoque práctico*. Addison-Wesley (1997).



1. Diseñar una Máquina de Turing que calcule el complemento a 1 de un número binario. (Es decir, que sustituya los 0's por 1's y los 1's por 0's).
2. Diseñar una Máquina de Turing que obtenga el sucesor de un número en codificación unaria. Considerar en la codificación unaria que el 0 se representa por la cadena vacía, el 1 por 1, el 2 por 11, etc.
3. Diseñar una Máquina de Turing que obtenga el predecesor de un número en codificación unaria. Considerar la codificación unaria del 0 igual que en el ejercicio 2.
4. Diseñar una Máquina de Turing que calcule la paridad de un número binario. Es decir, si el número de 1's de la cadena es par, se añade un 0 al final, y si es impar, se añade un 1.
5. Diseñar una Máquina de Turing que sea un contador unario de caracteres del lenguaje con alfabeto $\Sigma = \{a,b,c\}$. Es decir, se deben devolver tantos 1's como caracteres haya en la palabra de entrada. Considerar la codificación unaria del 0 igual que en el ejercicio 2.
6. Diseñar una Máquina de Turing que haga una copia de una cadena de símbolos $\{A,B,C\}$. Por ejemplo, para la entrada "bAABCAB" devuelve en la cinta "bAABCAABCAb", donde 'b' representa el blanco.
7. Diseñar una Máquina de Turing que tome como entrada una cadena con M 1's y N A's ($M \leq N$), y cambia las M primeras A's por B's. Por ejemplo, para la entrada "b11AAAAAb" devuelve en la cinta "b11BBAAAb", donde 'b' representa la celda de la cinta vacía.
8. Diseñar una Máquina de Turing que tome como entrada dos palabras formadas por los símbolos del alfabeto $\{0,1,2\}$, separadas por el símbolo $\{\#\}$, y comprueba si son iguales. Por ejemplo, para la entrada b2101#2101b devuelve que sí son iguales, donde 'b' representa la celda de la cinta vacía.
9. Diseñar una Máquina de Turing que obtenga el sucesor de un número binario.
10. Diseñar una Máquina de Turing que obtenga el antecesor de un número binario.

SOLUCIONES

1. Diseñar una Máquina de Turing que calcule el complemento a 1 de un número binario. (Es decir, que sustituya los 0's por 1's y los 1's por 0's).

Solución:

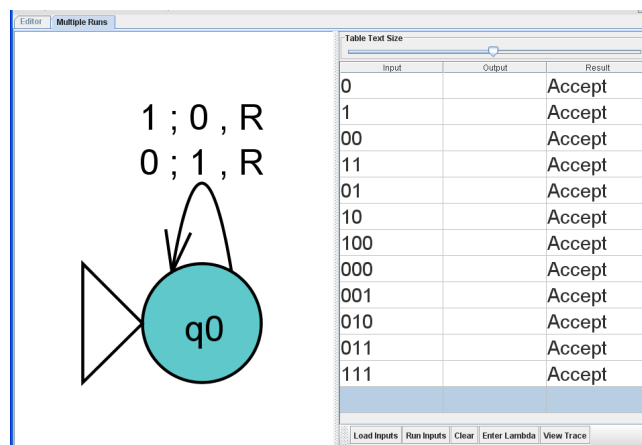
Algoritmo: Recorrer la cinta de izquierda a derecha, sustituyendo 1's por 0's y viceversa.

Definición de la MT

Necesitamos implementar una Máquina de Turing que se comporte como transductor, porque genera una salida en la cinta.

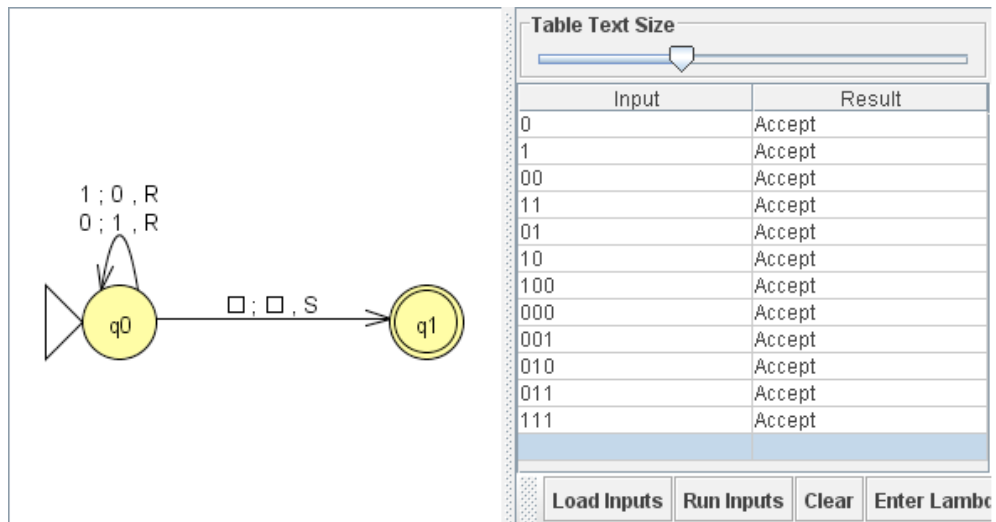
Tendremos un solo estado, que será el inicial, q_0 , sobre el que se realizan todas las transiciones

$MT_1 = (\{0,1\}, \{0,1,\square\}, \square, \{q_0\}, q_0, f, \{\Phi\})$, donde f :



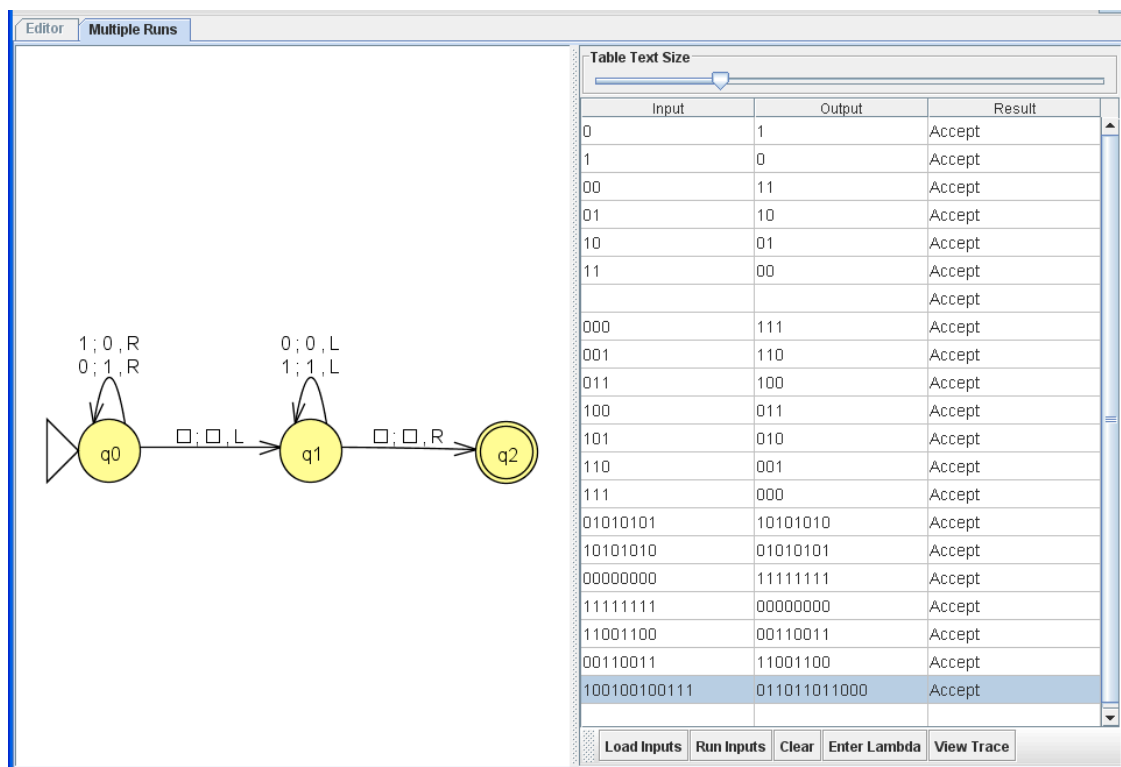
Si además se exige que el transductor termine en un estado final y pare, si la entrada es correcta, es decir, una simple secuencia de ceros y unos, la solución sería:

$MT_{1,2} = (\{0,1\}, \{0,1,\square\}, \square, \{q_0, q_1\}, q_0, f, \{q_1\})$, donde f :



Si además del estado final se exige que la cabeza de la pila termine al inicio de la palabra (para que se pueda ver el contenido de la cinta en la herramienta Jflap), se obtiene la siguiente máquina de Turing:

$MT_{1.3} = (\{0,1\}, \{0,1,\square\}, \square, \{q_0, q_1, q_2\}, q_0, f, \{q_2\})$, donde f :



2. Diseñar una Máquina de Turing que obtenga el sucesor de un número en codificación unaria. Considerar en la codificación unaria que el 0 se representa por la cadena vacía, el 1 por 1, el 2 por 11, etc.

Solución:

La representación de un número en unario, según las indicaciones del enunciado es:

$$\begin{array}{ccc} n & \Rightarrow & n+1 \text{ "unos"} \\ \text{(decimal)} & & \text{(unario)} \end{array}$$

Así:

$$0 \Rightarrow \text{cadena vacía}$$

$$1 \Rightarrow 1$$

$$2 \Rightarrow 11$$

$$3 \Rightarrow 111$$

$$4 \Rightarrow 1111$$

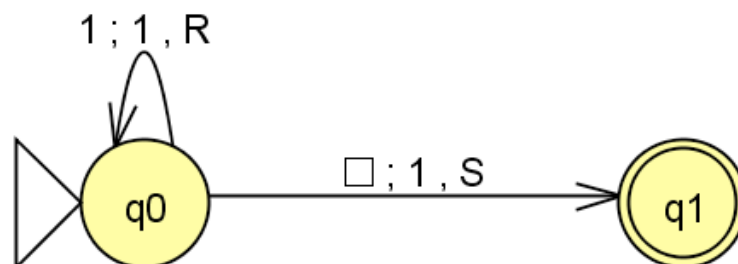
$$5 \Rightarrow 11111$$

...

Algoritmo: Sucesor unario: Recorrir número hasta el final [transición recursiva en q_0], y allí añadir un 1 adicional, a la derecha de la secuencia de 1's [transición de q_0 a q_1].

Definición de la MT

$MT_2 = (\{1\}, \{1, \square\}, \square, \{q_0, q_1\}, q_0, f, \{q_1\})$, donde f :



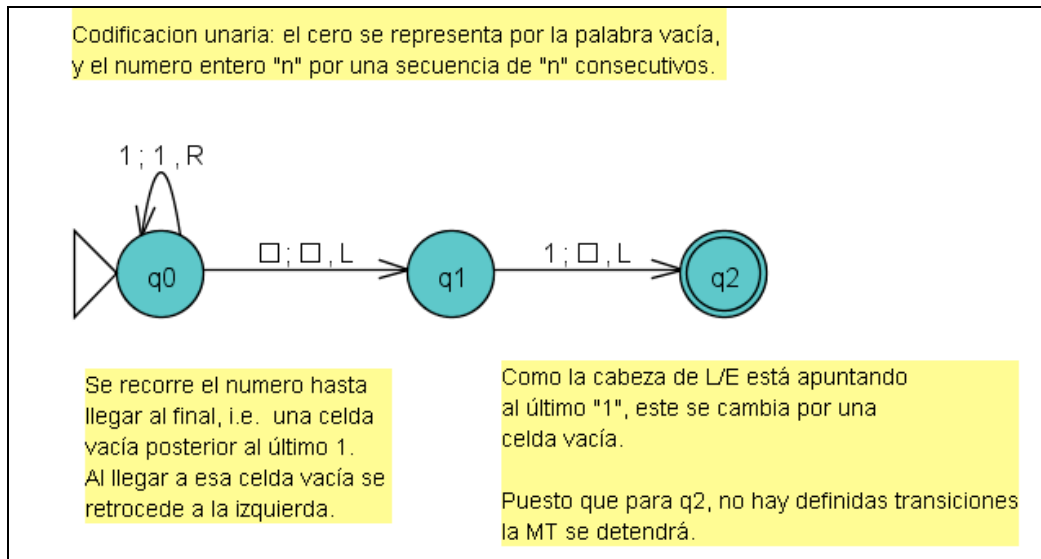
3. Diseñar una Máquina de Turing que obtenga el predecesor de un número en codificación unaria. Considerar la codificación unaria del 0 igual que en el ejercicio 2.

Solución:

Algoritmo: Predecesor unario: Recorrer número hasta el final [transición recursiva en q_0], situarse en el último dígito [transición q_0 a q_1 , moviendo puntero a la izquierda], y escribir un blanco encima para borrarlo [transición q_1 a q_2].

Definición de la MT

$MT_{3.1} = (\{1\}, \{1, \square\}, \square, \{q_0, q_1, q_2\}, q_0, f, \{q_2\})$, donde f :

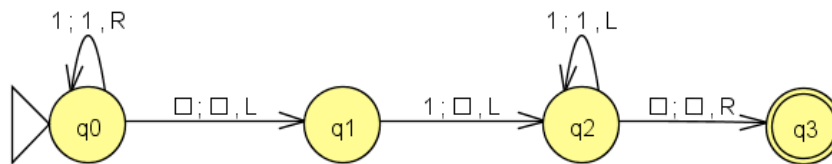


Si además se quiere que la cabeza lectora de la máquina de Turing apunte al primer símbolo de la palabra, se necesita una transición recursiva en q_2 , para recorrer la cinta hacia la izquierda, y un nuevo estado final, q_3 , al que transitar al llegar al símbolo de celda vacía, quedando la siguiente MT:

$MT_{3.2} = (\{1\}, \{1, \square\}, \square, \{q_0, q_1, q_2, q_3\}, q_0, f, \{q_3\})$, donde f :

■ representa la celda vacía (lo que en teoría era el símbolo "b")

Codificación unaria: el cero se representa por la palabra vacía, y el número entero "n" por una secuencia de "n" consecutivos.



Se recorre el número hasta llegar al final, i.e. una celda vacía posterior al último 1. Al llegar a esa celda vacía se retrocede a la izquierda.

Como la cabeza de L/E está apuntando al último "1", este se cambia por una celda vacía.

Las transiciones de q1 a q2, de q2 a q2 y de q2 a q3 es para llevar la cabeza lectora al primer 1 por la izquierda para que JFLAP muestre en Input-Multiple Run (transducer) el resultado de la cinta correctamente.

Puesto que para q3 no hay definidas transiciones la MT se detendrá.

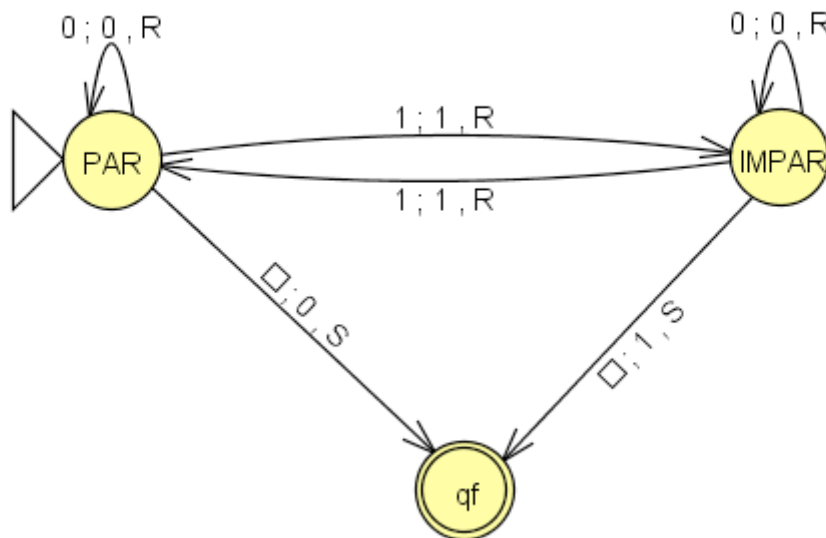
4. Diseñar una Máquina de Turing que calcule la paridad de un número binario. Es decir, si el número de 1's de la cadena es par, se añade un 0 al final, y si es impar, se añade un 1.

Solución:

Algoritmo: Recorrer de izquierda a derecha, recordando si se lleva un nº de 1's par o impar (situándose en un estado diferente), para añadir al final 0 ó 1, respectivamente.

Definición de la MT

$MT_4 = (\{0,1\}, \{0,1,\square\}, \square, \{PAR, IMPAR, qf\}, PAR, f, \{qf\})$, donde f :



Definición de estados:

El estado “PAR”, representa que se ha leído un número de **1's par** (considerando el 0 par).

El estado “IMPAR”, representa que se ha leído un número de **1's impar**.

El estado “qf” es el estado final, al que se llega sólo al final, tras añadir el 1 ó 0 de paridad de 1's.

Definición de transiciones:

Mientras se recorre la cadena, la máquina de Turing transita entre los estados PAR o IMPAR, dependiendo de la cantidad de 1's de la subcadena leída hasta el momento. En cualquiera de los dos estados:

- si se lee un 1, se cambia de estado, porque ha cambiado la paridad del número.
- si se lee un 0, se mantiene en el mismo estado, porque no ha cambiado la paridad.
- si se lee un blanco, se transita al estado final y se para, tras escribir un dígito distinto según el estado actual de la máquina:
 - PAR (nº de 1's par): escribir un 0, para mantener la paridad existente.
 - IMPAR (nº de 1's impar): escribir un 1, para conseguir un número de 1's par.

5. Diseñar una Máquina de Turing que sea un contador unario de caracteres del lenguaje con alfabeto $\Sigma = \{a,b,c\}$. Es decir, se deben devolver tantos 1's como caracteres haya en la palabra de entrada. Considerar la codificación unaria del 0 igual que en el ejercicio 2.

Solución:

Dado que no se especifica si se debe mantener la cadena original de a's, b's y c's, se pueden asumir diversas interpretaciones, como las tres que se presentan a continuación:

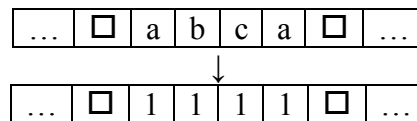
- a) No mantener la palabra de entrada.
- b) Mantener la posición de la palabra de entrada, pero sustituida por marcas, y a continuación, el contador.
- c) Mantener la palabra de entrada, y a continuación, el contador.

La solución de cada interpretación se presenta a continuación:

a) Versión a

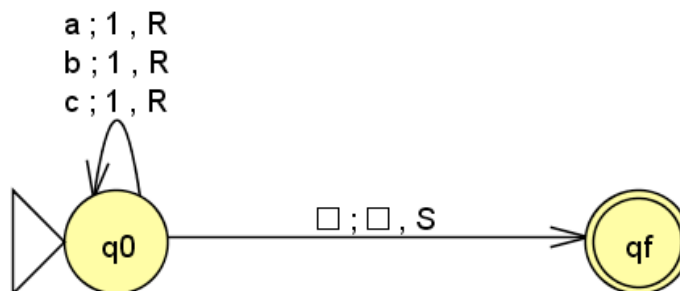
Algoritmo: Recorrer la palabra de entrada, de izquierda a derecha, y sustituir cada a, b y c por un 1. Al llegar al blanco, parar.

Ejemplo:



Definición de la MT

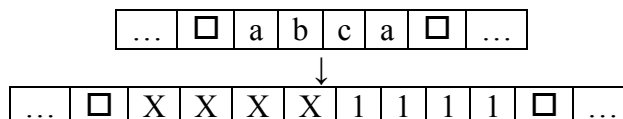
$MT_{5,a} = (\{a,b,c\}, \{1, \square\}, \square, \{q_0, q_f\}, q_0, f, \{q_f\})$, donde f:



b) Versión b

Algoritmo: Recorrer la palabra de entrada, marcando cada letra con una X, y escribir un 1 por cada una al final de la palabra.

Ejemplo:



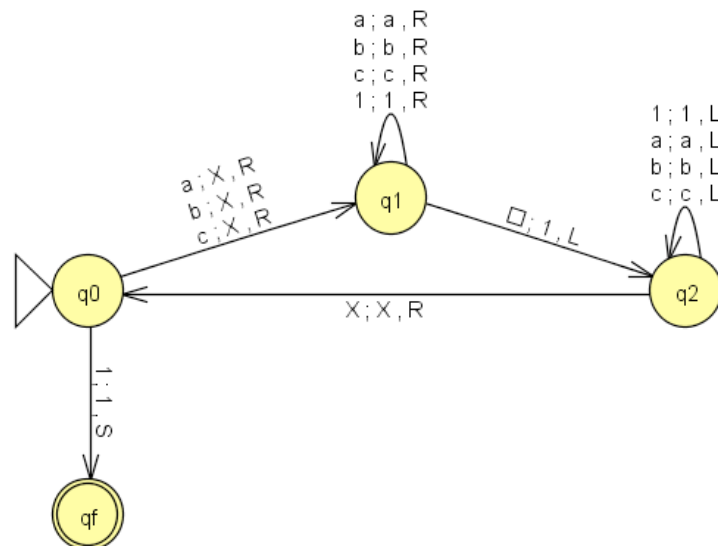
Detalladamente:

- 1.- Marcar con una X el primer carácter del alfabeto (a, b o c) no marcado [transición de q0 a q1].
- 2.- Recorrer la palabra hacia la derecha hasta encontrar un blanco [transiciones recursivas en estado q1].
- 3.- Escribir 1 en el blanco localizado [transición de q1 a q2].
- 4.- Retroceder hasta la marca X [transiciones recursivas en estado q2].
- 5.- Avanzar una posición a la derecha, y repetir desde el paso 1 [transición de q2 a q0], hasta que se recorra toda la palabra de entrada, es decir, se lea un 1, en lugar de a, b o c, y entonces se para la máquina [transición de q0 a qf].

Definición de la MT

$MT_{5,b} = (\{a,b,c\}, \{1,X,\square\}, \square, \{q_0, q_1, q_2, q_f\}, q_0, f, \{q_f\})$, donde f:

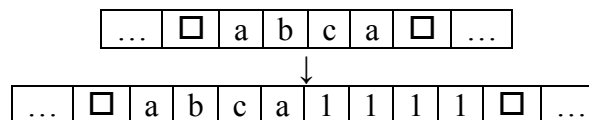
* se añaden 2 estados y muchas transiciones con respecto a la versión anterior.



c) Versión c

Algoritmo: Recorrer la palabra de entrada, marcando cada letra con un símbolo específico, escribir un 1 al final de la palabra, retroceder hasta la última marca, y sustituir por la letra original.

Ejemplo:



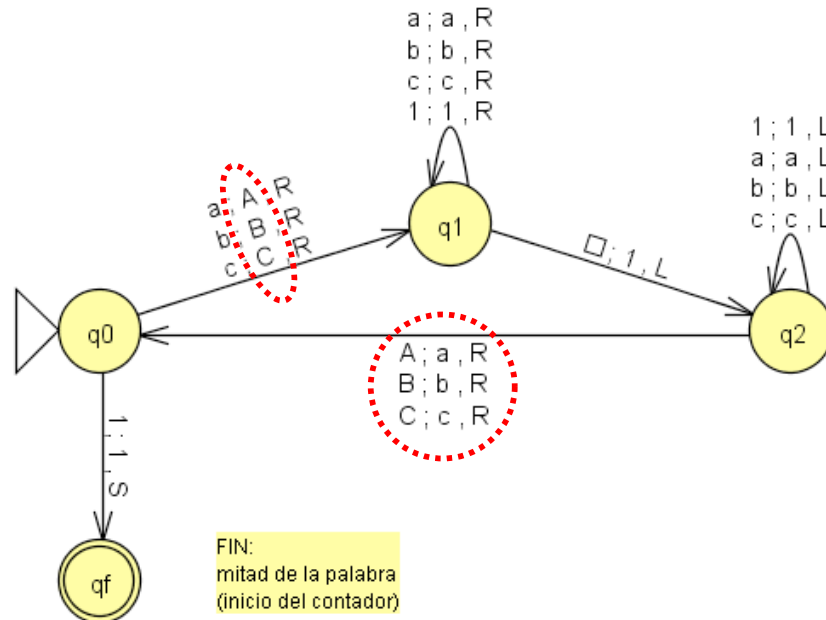
Detalladamente (con cambios respecto al algoritmo anterior resaltados en rojo):

- 1.- Marcar con una A, B o C el primer carácter del alfabeto no marcado, respectivamente para a, b o c [transición de q0 a q1].
- 2.- Recorrer la palabra hacia la derecha hasta encontrar un blanco [transiciones recursivas en estado q1].
- 3.- Escribir 1 en el blanco localizado [transición de q1 a q2].
- 4.- Retroceder hasta la **marca A, B o C** [transiciones recursivas en estado q2].

5.- Sustituir marca por su carácter original correspondiente ($A \rightarrow a$, $B \rightarrow b$ o $C \rightarrow c$), y avanzar a la derecha [transición de q_2 a q_0]. Repetir desde paso 1, hasta que en la siguiente casilla se lea un 1, y entonces se para la máquina [transición de q_0 a q_f].

Definición de la MT

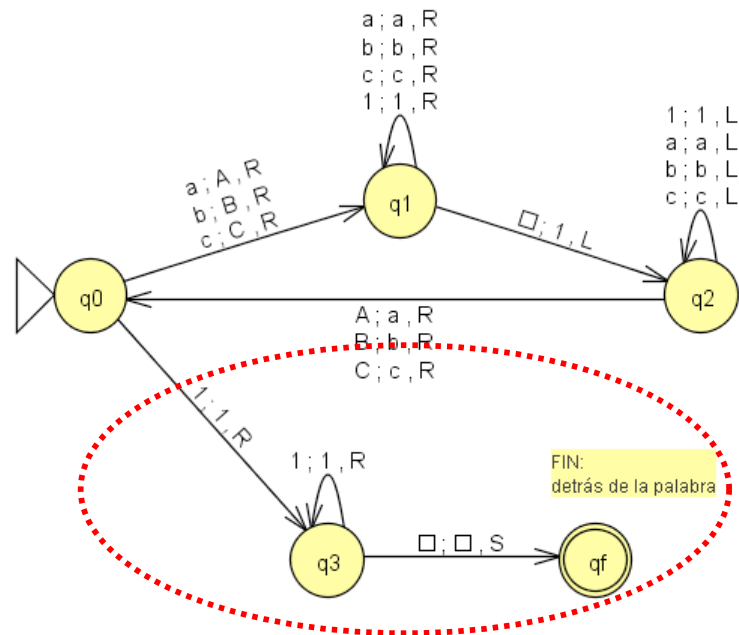
$MT_{5,c} = (\{a,b,c\}, \{a,b,c,1,A,B,C,\square\}, \square, \{q_0,q_1,q_2,q_f\}, q_0, f, \{q_f\})$, donde f (con cabeza de la pila a mitad de la palabra, donde comienza el contador):



Si se requiere que la cabeza de la pila acabe en una posición concreta, las variantes serían las siguientes, para el fin y el inicio de la palabra:

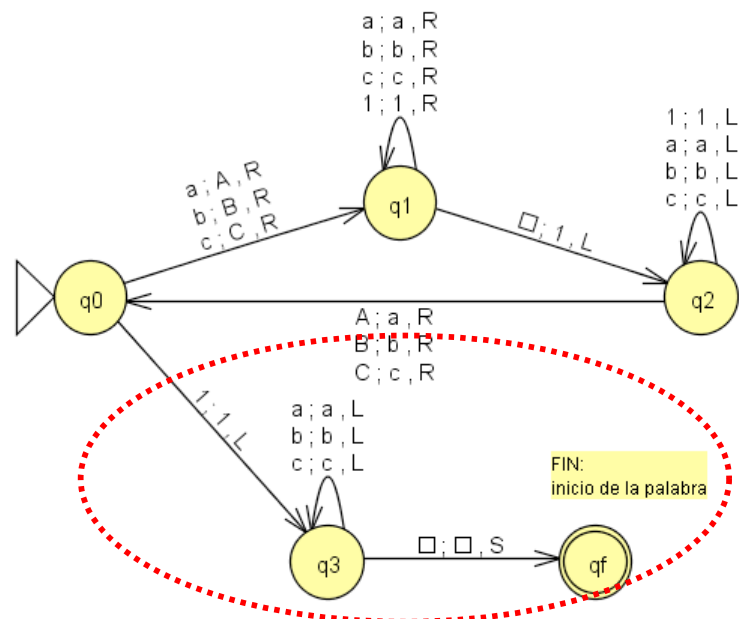
$MT_{5,c,bis} = (\{a,b,c\}, \{1,A,B,C,\square\}, \square, \{q_0,q_1,q_2,q_3,q_f\}, q_0, f, \{q_f\})$, donde f , con cabeza de la pila al final de la palabra:

* en el fragmento resaltado se recorre la cinta hacia delante, saltando los 1's del contador, hasta llegar al primer blanco.



donde f, con cabeza de la pila al inicio de la palabra (como se necesita en JFLAP para ver los resultados):

* en el fragmento resaltado, se recorre la cinta hacia atrás, saltando las a's, b's y c's, hasta llegar al primer blanco. También se podría avanzar una posición al llegar al blanco, para parar en el primer símbolo del alfabeto, y no en el blanco anterior.



6. Diseñar una Máquina de Turing que haga una copia de una cadena de símbolos {A,B,C}. Por ejemplo, para la entrada “bAABCab” devuelve en la cinta “bAABCAAABCab”, donde ‘b’ representa el blanco.

Solución:

Algoritmo: *Copia símbolos de una cadena, con alfabeto {A,B,C}, sin carácter de separación.*

Se diseña la máquina de Turing como un transductor, porque tiene que dar una salida en la cinta: la cadena original seguida de la cadena copiada, sin ningún separador. Así, aunque se utilicen marcas durante el proceso de copiado, luego se deberán eliminar, para que la cinta sólo contenga símbolos del alfabeto de entrada.

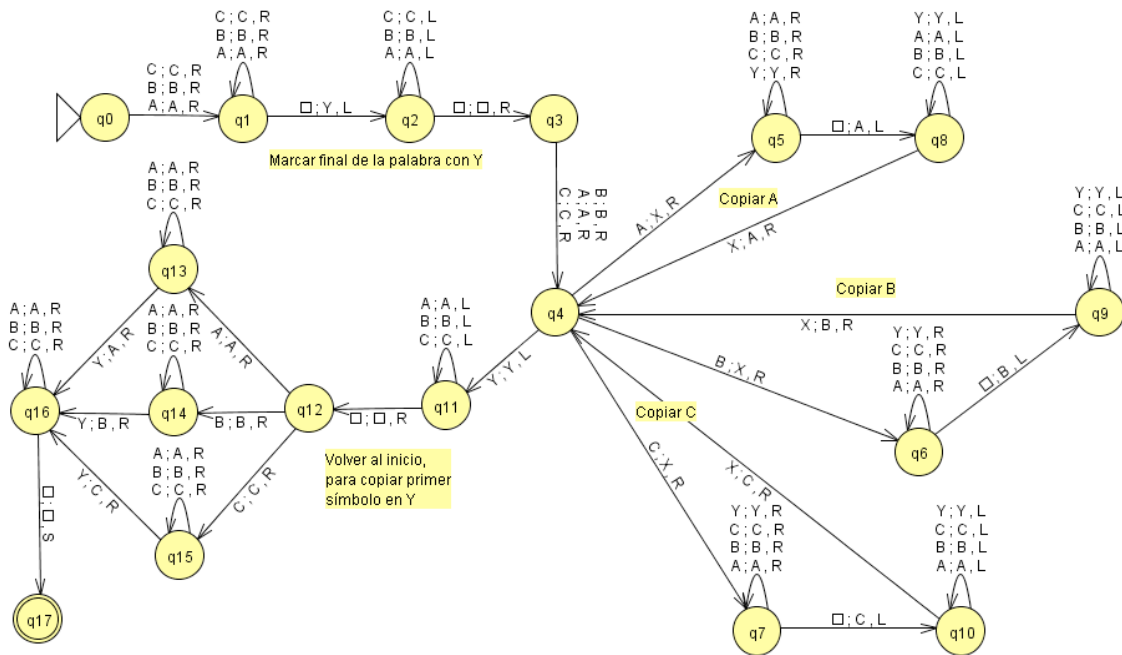
La idea es ir copiando uno a uno cada carácter, utilizando una rama de estados diferente para cada símbolo del alfabeto, de forma que se pueda “guardar en memoria” de qué símbolo se trata, al no existir variables explícitamente en las máquinas de Turing.

Se deja una marca X en el símbolo de la palabra original, para saber dónde regresar después, y se recorre la cadena hasta el final, sustituyendo el primer blanco encontrado por el símbolo representado por la rama de estados por la que se ha llegado. Entonces, se retrocede hasta encontrar la X, sustituyéndola también por el símbolo representado en la rama de estados por la que se ha llegado. Se avanza a la derecha, y se empieza otra vez el proceso marcando con una X el próximo símbolo.

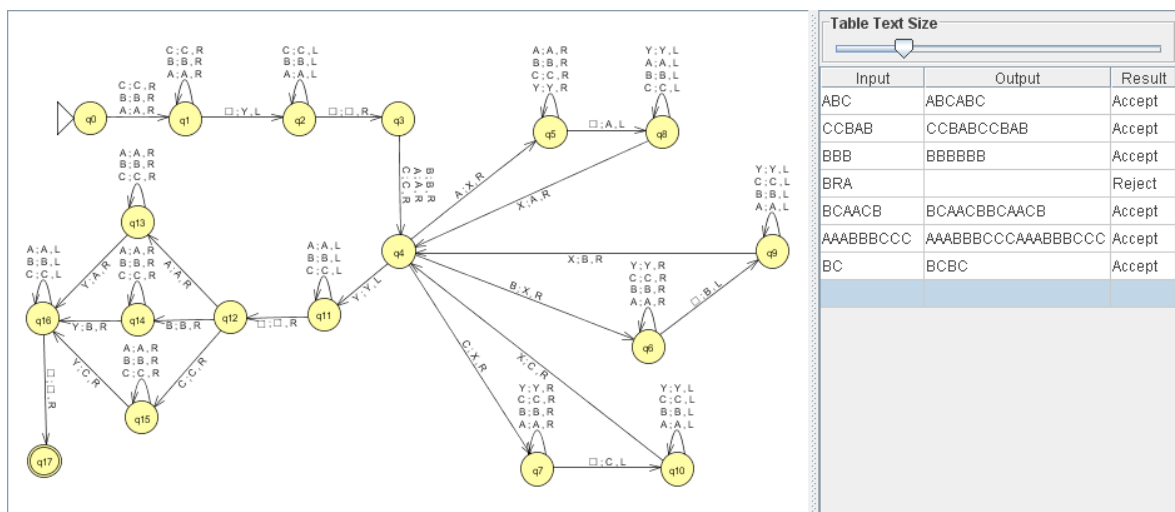
Al no haber marca de separación entre las dos cadenas, al inicio, se recorrerá la palabra hasta el final, escribiendo tras el último símbolo una Y. Ese será el lugar donde parar de copiar caracteres. Además, al final, cuando se hayan copiado todos los demás, ahí habrá que copiar el primer símbolo de la palabra. Por lo que, al inicio hay que saltarse el primer símbolo, y empezar marcando con X el segundo. Al llegar a la Y, se retrocede hasta el blanco inicial, y se copia el siguiente carácter en la Y.

Definición de la MT

$MT_6 = (\{A,B,C\}, \{A,B,C,X,Y,\square\}, \square, \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9, q_{10}, q_{11}, q_{12}, q_{13}, q_{14}, q_{15}, q_{16}, q_{17}\}, q_0, f, \{q_{17}\})$, donde f (con cabeza de la pila al final de la palabra):



Comprobación con algunas palabras. Para poder ver el resultado del transductor, se han modificado las transiciones finales, para acabar apuntando al inicio de la palabra en vez de al final (transiciones de estados q16 y q17):



7. Diseñar una Máquina de Turing que tome como entrada una cadena con M 1's y N A's ($M \leq N$), y cambia las M primeras A's por B's. Por ejemplo, para la entrada "b11AAAAb" devuelve en la cinta "b11BBAAAb", donde 'b' representa la celda de la cinta vacía.

Solución:

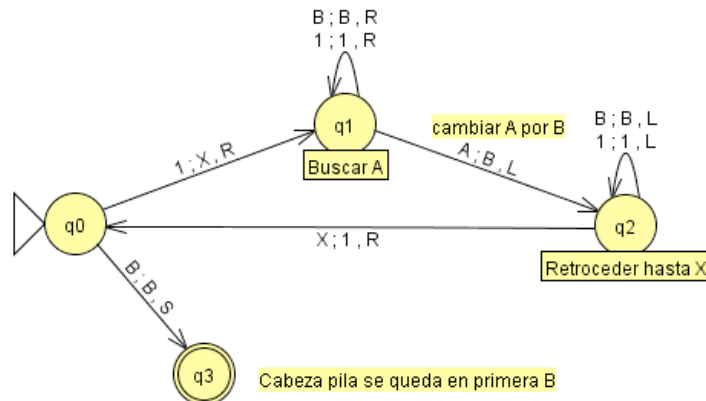
Algoritmo: Cambiar A's iniciales por B's, tantas como 1's.

Ir cambiando A por B, dejando marcado con una X el 1 que estoy sustituyendo en el inicio de la palabra, para saber por dónde voy al regresar. En ese momento, se vuelve a sustituir la X por un 1, y se avanza a la derecha, para repetir el proceso para el próximo uno. Al final, si se quiere acabar con el puntero de la cinta al final de la palabra, se deben avanzar todas las B's y todas las A's restantes de las palabras, hasta llegar al blanco.

Definición de la MT

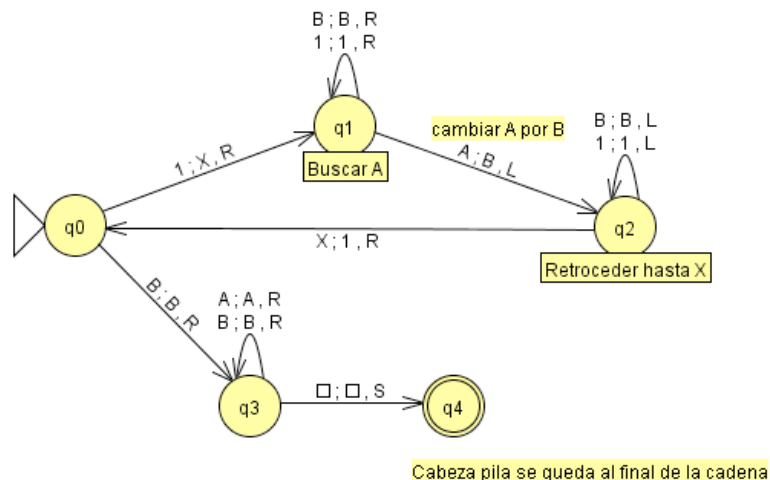
- V1: si la cabeza se queda apuntando a la primera B.

$MT_{7,1} = (\{1, A, B\}, \{1, A, B, X, \square\}, \square, \{q_0, q_1, q_2, q_3\}, q_0, f, \{q_3\})$, donde f:



- V2: si la cabeza se queda apuntando al final de la cadena.

$MT_{7,2} = (\{1, A, B\}, \{1, A, B, X, \square\}, \square, \{q_0, q_1, q_2, q_3, q_4\}, q_0, f, \{q_4\})$, donde f:



- 8. Diseñar una Máquina de Turing que tome como entrada dos palabras formadas por los símbolos del alfabeto $\{0,1,2\}$, separadas por el símbolo $\{\#\}$, y compruebe si son iguales. Por ejemplo, para la entrada $b2101\#2101b$ devuelve que sí son iguales, donde 'b' representa la celda de la cinta vacía.**

Solución:

Algoritmo: *Comparar dos cadenas, con símbolos $\{0,1,2\}$, separadas por $\#$.*

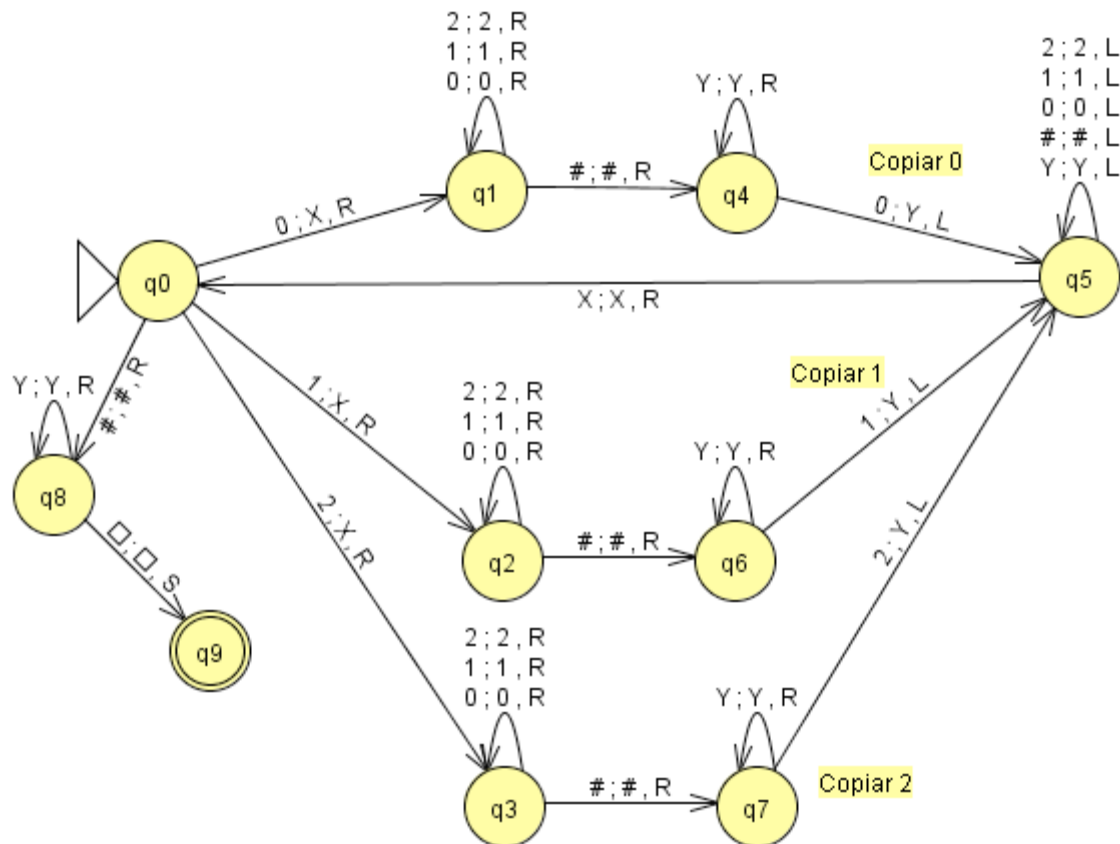
Se diseña la máquina de Turing como un reconocedor, que acaba en estado final cuando las palabras de entrada son iguales, y en uno no final cuando no lo son. Se considera que el contenido de la cinta puede modificarse, dado que no es un transductor y no se pide explícitamente que escriba una salida.

Así, se irá comparando carácter a carácter. Para cada símbolo de la palabra de entrada que se esté comparando, inicialmente se marca con una X, para saber hasta dónde hay que regresar para comprobar el siguiente símbolo. Entonces, en función del símbolo del alfabeto que sea, se avanza a través de estados diferentes, para recordar si se tiene que comparar en la segunda palabra con un 0, con un 1 ó con un 2; es decir, el equivalente a una variable en memoria de un lenguaje de programación de alto nivel. Se avanza hasta la $\#$, y se recorre la parte de la segunda palabra ya comparada, marcada convenientemente con Y's. Al llegar al primer carácter distinto de Y, se compara con el carácter analizado en ese momento de la primera palabra, (almacenado "en memoria" en función del estado en el que la máquina de Turing se encuentre). Si es igual, se marca con una Y, se retrocede hasta la primera X encontrada, y se repite el proceso desde el carácter a su derecha. Si no es igual, se queda en ese estado, porque no habrá transición definida.

Cuando la palabra finalice, se habrá llegado al carácter $\#$. Entonces, se recorren todas las Y's, para asegurarse que la segunda palabra no es más larga que la primera y no han sobrado símbolos, y cuando se llegue a un blanco, se transita al estado final, que verifica que las dos palabras son iguales. Si las palabras son iguales, en la cinta quedará una cadena de X's de igual longitud a la de las palabras, seguida de un $\#$ y seguida de tantas Y's como X's, y apuntando al carácter blanco después de la última Y.

Definición de la MT

$MT_8 = (\{0,1,2,\#\}, \{0,1,2,\#,X,Y,\square\}, \square, \{q_0,q_1,q_2,q_3,q_4,q_5,q_6,q_7,q_8,q_9\}, q_0, f, \{q_9\})$,
donde f (con cabeza de la pila al final de la palabra):



Comprobando con algunas palabras:

Diagrama de la Máquina de Turing MT_8 con estados q_0 a q_9 . El diagrama muestra transiciones basadas en la lectura de un símbolo y la escritura de otro, con acciones de movimiento de la cabeza (R para Right, L para Left). Se destacan tres copias de la palabra: "Copiar 0" (q0 a q5), "Copiar 1" (q0 a q6) y "Copiar 2" (q0 a q7). El estado q_9 es el estado final.

Table Text Size

Input	Result
012#012	Accept
11#11	Accept
210210#210210	Accept
22001122#221012	Reject
22001122#22001122	Accept
#	Accept
210#21010	Reject

Load Inputs Run Inputs Clear Enter Lambda View Trace

9. Diseñar una Máquina de Turing que obtenga el sucesor de un número binario.

Solución:

En primer lugar, se escribe un conjunto de números binarios consecutivos, para intentar buscar regularidades en los números binarios que permitan idear un algoritmo para resolver el problema de forma general.

Entrada		Salida (sucesor)	
Decimal	Binario	Binario	Decimal
0	0	1	1
1	1	10	2
2	10	11	3
3	11	100	4
4	100	101	5
5	101	110	6
6	110	111	7
7	111	1000	8
8	1000	1001	9
9	1001	1010	10
10	1010	1011	11

Tras observar detenidamente por filas las dos columnas centrales sombreadas de la tabla anterior, se puede concluir que:

- si la entrada acaba en 0, su sucesor se obtiene cambiando dicho último 0 por un 1.
- si la entrada acaba en 1, la obtención del sucesor es más complicada:
 - se debe recorrer el número del final al inicio, sustituyendo todos los 1's por 0's, hasta que aparezca un 0 (o en su defecto un blanco) y cambiarlo por un 1.

Como no se especifica nada en el enunciado, se puede asumir que la cinta al final sólo contendrá el sucesor, sin preservar la entrada inicial.

Basándose en esta generalización, se puede pasar a describir el algoritmo y definir la máquina de Turing formalmente.

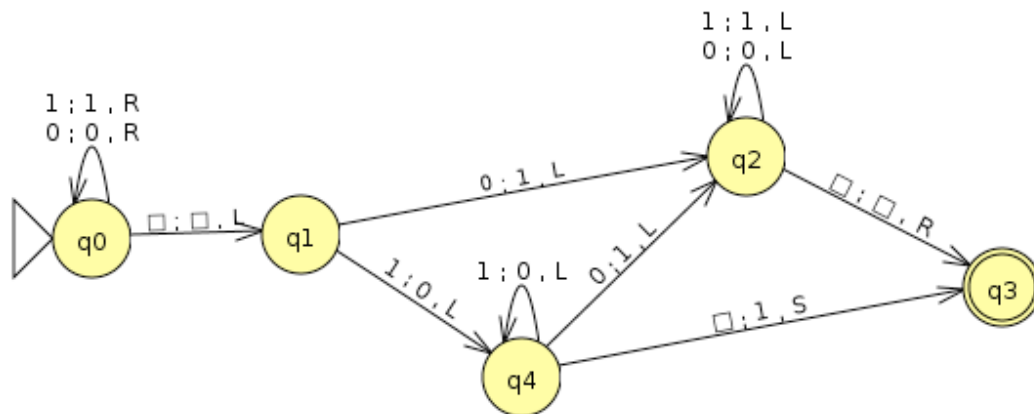
Algoritmo (detallado):

1. Recorrer el número hasta el final, buscando el primer blanco a la derecha de los 1's y 0's [transiciones recursivas en estado q0]. Al llegar, mover a la izquierda una posición [transición de q0 a q1], para situarse en el último dígito del número binario [llegada a estado q1].
2. En la posición final:
 - 2.1. si es un 0, se cambia por un 1 [transición de q1 a q2], y se va al paso 5.
 - 2.2. si es un 1, ir recorriendo el número hacia atrás, sustituyendo 1's por 0's [transición de q1 a q4 y transiciones recursivas en q4], hasta que aparezca un 0 (ir a paso 3) o un blanco (ir a paso 4).
3. Si recorriendo hacia atrás se llega a un 0, se sustituye por un 1 [transición de q4 a q2], y se va al paso 5.

4. Si recorriendo hacia atrás se llega a un blanco, se sustituye por un 1 [transición de q_4 a q_3], y la máquina se para, porque ya se tiene el sucesor y la cabeza lectora está al inicio del número.
5. Tras obtener el sucesor, pero tener la cabeza lectora en el medio del número, se transita al estado q_2 , donde se recorre el número hacia la izquierda [transiciones recursivas en q_2], hasta encontrar un blanco, moviéndose una posición a la derecha [transición a q_3], para finalizar con la cabeza lectora al inicio del número sucesor binario.

Definición de la MT

$MT_9 = (\{0,1\}, \{0,1,\square\}, \square, \{q_0, q_1, q_2, q_3, q_4\}, q_0, f, \{q_3\})$, donde f (con cabeza de la pila al inicio del número):



Comprobando con algunas palabras:

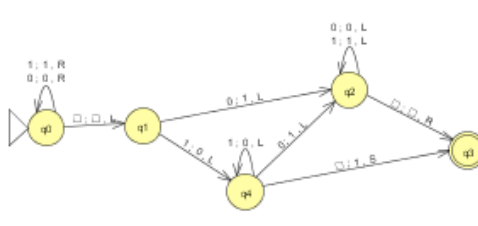


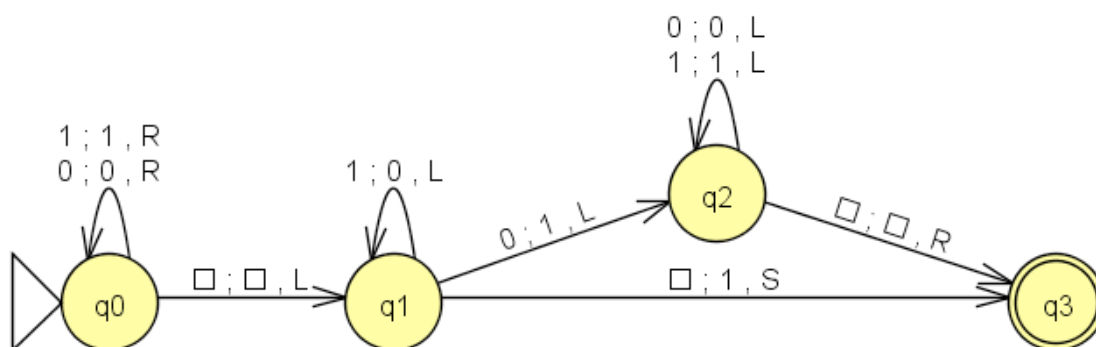
Table Text Size

Input	Output	Result
0	1	Accept
1	10	Accept
10	11	Accept
11	100	Accept
100	101	Accept
101	110	Accept
110	111	Accept
111	1000	Accept
1000	1001	Accept
1001	1010	Accept
1010	1011	Accept

Aunque la solución presentada es perfectamente válida, esto no indica que sea la única correcta. Pueden existir otras máquinas de Turing que también sean capaces de calcular el sucesor de un número binario, con más estados y transiciones, o incluso con menos. O también, se puede pensar un algoritmo válido, definir todos los elementos de la máquina de Turing correspondiente, y a posteriori descubrir una simplificación de la misma.

Por ejemplo, al observar el diagrama de transiciones que cumple el algoritmo propuesto inicialmente, podemos darnos cuenta que: la transición del estado q_4 al q_2 es igual que la del estado q_1 a q_2 . Además, la transición de q_1 a q_4 es la misma que la recursiva en q_4 , es decir, sustituir 1 por 0 y mover la cabeza lectora a la izquierda. Esta transición se puede cumplir de “0 a n ” veces, dependiendo de la existencia de 1’s o no al final de la cadena; por lo que no se exige un estado intermedio para asegurar que se ejecuta esa transición al menos una vez. Así, para poder definir una máquina de Turing sin el estado q_4 y ninguna de sus transiciones, sólo falta verificar que la transición con el blanco de q_4 a q_3 , se puede pasar de q_1 a q_3 . Esta transformación también es válida, dado que puede que no exista ningún 0 en el número binario de entrada, y no sea necesario transitar de q_1 a q_2 por la rama superior de la máquina de Turing. Por lo tanto, otra máquina de Turing, con menos estados y transiciones, que también resuelve este problema sería la siguiente:

$MT_{9,2} = (\{0,1\}, \{0,1,\square\}, \square, \{q_0, q_1, q_2, q_3\}, q_0, f, \{q_3\})$, donde f (con cabeza de la pila al inicio del número):



También se comprueba que la máquina de Turing obtiene la salida correcta para todas las palabras verificadas anteriormente:

Table Text Size		
Input	Output	Result
0	1	Accept
1	10	Accept
10	11	Accept
11	100	Accept
100	101	Accept
101	110	Accept
110	111	Accept
111	1000	Accept
1000	1001	Accept
1001	1010	Accept
1010	1011	Accept

En este caso, el algoritmo también se simplificaría, no existiendo el paso 2.1, sino sólo el 2.2, independientemente de que el último dígito sea un 1 ó un 0.

10. Diseñar una Máquina de Turing que obtenga el antecesor de un número binario.

Solución:

Este problema está relacionado con el anterior. En este caso, hay que encontrar regularidades entre un número binario y su **antecesor**. De nuevo, como en el ejercicio anterior, recurrimos a una tabla que nos relacione dicho par de números.

Entrada		Salida (antecesor)	
Decimal	Binario	Binario	Decimal
1	1	0	0
2	10	1	1
3	11	10	2
4	100	11	3
5	101	100	4
6	110	101	5
7	111	110	6
8	1000	111	7
9	1001	1000	8
10	1010	1001	9
11	1011	1010	10

Tras observar detenidamente por filas las dos columnas centrales sombreadas de la tabla anterior, y teniendo en cuenta la similitud con el problema anterior, se puede concluir el algoritmo general para obtener el antecesor binario:

Algoritmo: Recorrer el número de derecha a izquierda, cambiando 0's por 1's, hasta encontrar el primer 1 (el menos significativo), que se cambia por 0.

Como no se especifica nada en el enunciado, se puede asumir que la cinta al final sólo contendrá el sucesor, sin preservar la entrada inicial.

Se asumen como entradas números naturales, sin el 0 porque su antecesor sería un número negativo, sin representación definida en el enunciado para los binarios negativos.

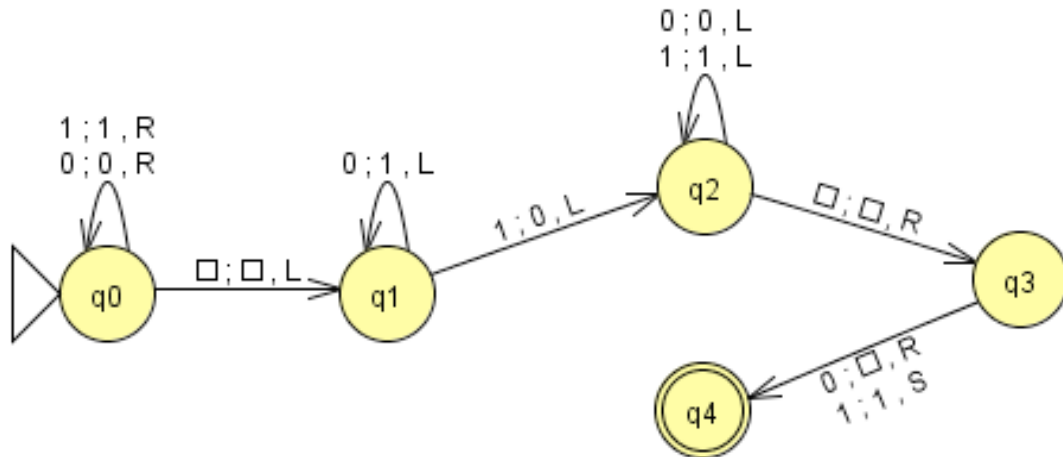
Algoritmo (detallado):

1. Recorrer el número hasta el final, buscando el primer blanco a la derecha de los 1's y 0's [transiciones recursivas en estado q0]. Al llegar, mover a la izquierda una posición [transición de q0 a q1], para situarse en el último dígito del número binario [llegada a estado q1].
2. Desde la posición final, ir recorriendo el número hacia atrás, sustituyendo 0's por 1's [transición recursiva en q1], hasta que aparezca un 1 (el menos significativo), que se cambia por un 0 [transición de q1 a q2]. Ya se tiene el antecesor escrito en la cinta.
3. A continuación, para tener la cabeza lectora al inicio del número, se recorre el número hacia la izquierda [transiciones recursivas en q2], hasta encontrar un blanco delante del número, moviéndose una posición a la derecha [transición a q3].

4. Para finalizar, se comprueba si existe un “0” no significativo al inicio del número, y se borra, sustituyéndolo por un blanco [transición con 0 de q_3 a q_4]. Si el primer dígito del número es un 1, se mantiene el dígito y la posición de la cabeza lectora.

Definición de la MT

$MT_{10} = (\{0,1\}, \{0,1,\square\}, \square, \{q_0, q_1, q_2, q_3, q_4\}, q_0, f, \{q_4\})$, donde f (con cabeza de la pila al inicio del número):



En resumen, se tiene el mismo diagrama que en el cálculo del sucesor, con las siguientes diferencias:

- Cambio de 0 por 1 y 1 por 0 en la transición recursiva del estado q_1 , y en la transición de q_1 a q_2 .
- Inclusión de un estado adicional, q_4 , para borrar el 0 no significativo a la izquierda que haya podido quedar tras la transformación. Si es un 1, no se modifica ni se mueve la cabeza lectora.

Comprobamos con una serie de números binarios la correcta obtención del antecesor, excepto para la entrada de número binario “1” para la que no se obtiene salida:

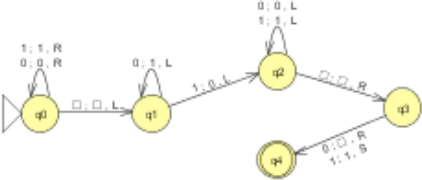


Table Text Size

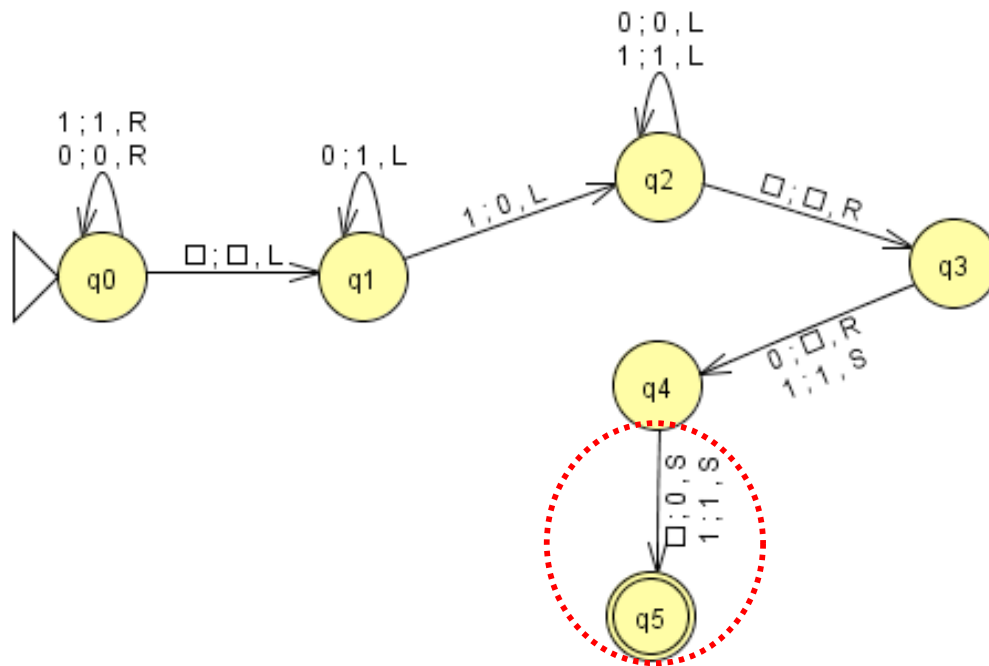
Input	Output	Result
1		Accept
10	1	Accept
11	10	Accept
100	11	Accept
101	100	Accept
110	101	Accept
111	110	Accept
1000	111	Accept
1001	1000	Accept
1010	1001	Accept
1011	1010	Accept

Buttons: Load Inputs, Run Inputs, Clear, Enter Lambda, View Trace

Esta verificación con la herramienta Jflap nos permite detectar que cuando la entrada es “1” y le corresponde un antecesor “0”, la máquina de Turing se queda con la cinta completamente en blanco. Para evitarlo, se necesita una transición adicional, antes de llegar al estado final, para verificar que si sólo hay un 0, no se borre, porque no se trata de un dígito no significativo a la izquierda, sino del único dígito del número. Así, se añade un nuevo estado q5, y una transición hasta él que verifique si no hay más dígitos, y en ese caso vuelva a escribir el 0 eliminado en la transición de q3 a q4 (que sí es necesario cuando existen 1's detrás de dicho 0 no significativo).

La nueva máquina de Turing quedaría de la siguiente forma (con los cambios marcados en rojo):

$MT_{10.2} = (\{0,1\}, \{0,1,\square\}, \square, \{q_0, q_1, q_2, q_3, q_4, q_5\}, q_0, f, \{q_5\})$, donde f (con cabeza de la pila al inicio del número):



Ahora la comprobación de las palabras verifica que se obtiene la salida correcta en la cinta de la máquina de Turing para el número binario “1”:

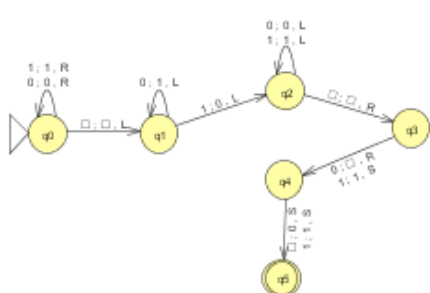


Table Text Size

Input	Output	Result
1	0	Accept
10	1	Accept
11	10	Accept
100	11	Accept
101	100	Accept
110	101	Accept
111	110	Accept
1000	111	Accept
1001	1000	Accept
1010	1001	Accept
1011	1010	Accept

Load Inputs
Run Inputs
Clear
Enter Lambda
View Trace