Algoritmia y Complejidad.

Curso 2023/24.

Ejercicio grupal M4 de una cinta.

Autores:

Cárdenas Palacios, Lucía

Cazorla Rodríguez, Rubén

Cotrina Santos, Joaquín

Martín Conejo, Ana

Índice:

Introducción:

El objetivo de esta memoria es construir una máquina de Turing determinista de una cinta que decida el lenguaje $L = \{w = \#x1\#x2 \dots \#xl \mid xi \in \{0,1\}^* \ \forall i,\ xi \neq xj \ \forall i \neq j\}$, es decir, subcadenas de ceros y/o unos separadas una de otras mediante símbolos #.

Como cuestión técnica, podemos pensar en la cadena vacía, $w = \varepsilon$, como parte del lenguaje o que al menos hay un símbolo # sin ningún otro símbolo detrás, es decir, $w = \#x_1$ donde $x_1 = \varepsilon$. Si bien la segunda posibilidad es claramente una cadena del lenguaje, la primera opción no está del todo clara. Para introducir la cadena vacía como cadena del lenguaje basta añadir a la función de transición la posibilidad $\delta(q_{ini}, \_) = (q_{acc}, \_, R)$.

La idea general de esta máquina de Turing va a ser considerar la primera subcadena $x1$ de w y compararla con las que tiene a su derecha, de una en una y carácter a carácter. Si hay coincidencia de $x1$ con xi rechazaremos w y, si no pasaremos a comparar $x1$ con $x(i+1)$ repitiendo el procedimiento. Cuando hayamos llegado a la última, procederemos a repetir todo este proceso comparando $x2$ con las subcadenas que están a su derecha y así, hasta que comparemos $x(l-1)$ con xl. Si todas ellas son distintas, entonces aceptaremos w como cadena del lenguaje. En cualquier otro caso, se rechazará.

Ejemplo:

Veamos un ejemplo: $w = \#001\#000\#100 \dots \#11\#101$ (léase por columnas)

$q_{ini}\#001\#000\#100 \dots \#11\#101$

$\$q_{primeraX}001\#000\#100 \dots \#11\#101$

$\$001q_{primeraX}\#000\#100 \dots \#11\#101$

$\$00q_{busca\$}1X000\#100 \dots \#11\#101$

$q_{busca\$}\$001X000\#100 \dots \#11\#101$

$\$q_{leer}001X000\#100 \dots \#11\#101$

$\$aq_{0}01X000\#100 \dots \#11\#101$

$\$a01Xq_{X0}000\#100 \dots \#11\#101$

$\$a01q_{busca\$}Xa00\#100 \dots \#11\#101$

…

$\$aabXaq_{borraX}a0\#100 \dots \#11\#101$

$\$aabq_{borraX}Xaa0\#100 \dots \#11\#101$

$\$aab\#q_{ponerX}aa0\#100 \dots \#11\#101$

$\$aab\#aa0q_{ponerX}\#100 \dots \#11\#101$

$\$aab\#aaq_{retroceso}0X100 \dots \#11\#101$

$q_{retroceso}\$001\#000X100 \dots \#11\#101$

…

$\$a01\#000\#100 \dots \#11Xq_{X0}101$

$\$a01\#000\#100 \dots \#11q_{borraX}X101$

$\$a01\#000\#100 \dots \#11\#q_{ponerX}101$

$\$aabXaaq_{X1}0\#100 \dots \#11\#101$

$\$a01\#000\#100 \dots \#11\#10q_{borra\$}1$

$q_{borra\$}\$001\#000\#100 \dots \#11\#101$

$\#q_{poner\$}001\#000\#100 \dots \#11\#101$

$\#001q_{poner\$}\#000\#100 \dots \#11\#101$

$\#001\$q_{primeraX}000\#100 \dots \#11\#101$

$\$a01\#000\#100 \dots \#11\#101q_{ponerX}$

…

$\#001\#000\#100 \dots \#11\$q_{primeraX}101$

$\#001\#000\#100 \dots \#11\$101q_{primeraX}$

$\#001\#000\#100 \dots \#11\$101\_q_{acc}$

Construcción de la máquina de Turing:

La información de lo que va haciendo la máquina de Turing debe quedar almacenada, bien en el estado, bien en la cinta a través del alfabeto de cinta. Para poder mantener indefinido el número l de subcadenas que hay en w, no podemos almacenar en el estado qué cadenas estamos comparando, ya que el número de estados es fijo e independiente de las entradas w. Por tanto, si el alfabeto del lenguaje viene dado por

$$\Sigma = \{0, 1, \#\}$$

vamos a utilizar un lenguaje de cinta

$$\Gamma = \{0, 1, \#, \_, a, b, \$, X\}$$

donde a es el símbolo que usamos para indicar que antes había un 0, b el símbolo que usamos para indicar que antes había un 1, _ es una celda vacía y $ y X son símbolos que indican que antes había un # siendo $ el indicativo de que la subcadena inmediatamente a su derecha y hasta el siguiente # es la primera subcadena que estamos comparando y X es el indicativo de éste está inmediatamente seguido de la subcadena con la que estamos comparando la subcadena anterior.

El funcionamiento de la máquina, en general, es el siguiente, desde un estado inicial, q_{ini}, al encontrarnos #, lo cambiamos por $ y pasamos al estado $q_{primeraX}$, que avanza hacia la derecha hasta ver el primer #. Si no hay, se acepta w como cadena del lenguaje. Si hay #, lo cambia por X y pasamos a $q_{busca\$}$, que retrocede hacia la izquierda hasta encontrar el $ (lo tiene que haber porque al estado $q_{primeraX}$ solo se llega tras cambiar un # por un $).

Una vez que el cabezal lee \$, pasamos buscar el primer carácter 0 o 1 que haya a la derecha del \$ desde el estado q_{leer}. Este estado ignora los posibles caracteres a y b. Si encuentra un 0 o un 1 pasaremos al estado q_0 o q_1 respectivamente, cambiando el carácter hallado por a o por b respectivamente. Si encuentra # o X pasaremos al estado q_f que indica que la subcadena izquierda llegó a su fin y no posee más caracteres susceptibles de ser comparados o que incluso no tenía ninguno inicialmente (caso de tener w dos # seguidos inicialmente). A priori, puede parecer que hay una posibilidad más, que es encontrar una casilla en blanco estando en el estado q_{leer}, pero eso no es posible porque las lecturas se producen tras haber situado el \$ y la X y siempre estando el cabezal a la derecha del \$, lo que imposibilita encontrar una celda vacía.

La lectura de final de cadena tiene un problema técnico si las subcadenas que se están comparando son consecutivas, es decir, no hay ningún # entre \$ y X. Estando en el estado q_{leer} y no encontrar ningún 0 ni ningún 1, si hallamos # podemos pasar a q_f y mover el cabeza a la derecha, porque el cabezal estará en X o éste seguirá a su derecha. Pero si el final de la subcadena izquierda viene dado por el símbolo X porque las subcadenas comparadas son consecutivas, hay que mover el cabezal a la izquierda, para asegurarse de que, desde q_f moviéndonos a la derecha, lleguemos a encontrar el símbolo X. Eso hace que, desde el estado q_f el cabezal pueda situarse sobre \$ si hay una subcadena vacía.

Desde cualquiera de los estados q_0, q_1 o q_f, avanza el cabezal hacia la derecha ignorando todos los caracteres hasta haber encontrado la X. Una vez leído X nos movemos a su derecha pasando al estado q_{X0}, q_{X1} o q_{Xf} respectivamente. Estando en uno de estos estados, nos movemos a la derecha ignorando posibles a's o b's. Si encontramos un 0 estando en q_{X0} o un 1 estando en q_{X1}, lo cambiamos por una a o b respectivamente y pasamos al estado $q_{busca\$}$ moviéndonos hacia la izquierda en busca de una siguiente lectura. Si encontramos un carácter distinto al leído en la primera cadena, ya sabemos que son distintas y pasamos al proceso de comparar la cadena de la izquierda con la siguiente subcadena a la derecha de la que actualmente tiene una X. Ese proceso comienza con el estado q_{borraX} yendo hacia la izquierda.

Si estuviésemos en q_{Xf} y encontramos tras las posibles a's y b's un 0 o un 1, sabemos que las cadenas son distintas y, por tanto, pasaríamos a q_{borraX} yendo hacia la izquierda. Si por el contrario encontramos # o _ entonces las cadenas son idénticas y hay que rechazar w.

Este proceso describe la comparación de dos subcadenas. Cuando se ha determinado que son distintas, nos encontramos en el estado q_{borraX} que se encarga de inicializar el cambio de la subcadena derecha a la inmediatamente posterior a la comparada. El estado q_{borraX} se mueve hacia la izquierda hasta que encuentra X, que lo vuelve a poner #, se mueve a su derecha y pasamos al estado q_{ponerX}. En este estado nos movemos hacia la derecha buscando el siguiente #. Al encontrarlo, lo cambiamos por X, nos movemos a su izquierda y pasamos al estado $q_{retroceso}$ que se mueve a la izquierda buscando \$ pero, a

diferencia de $q_{busca\$}$, va cambiando los a's por 0's y los b's por 1's para poder realizar la siguiente comparación de subcadenas.

Si estando en el estado q_{ponerX} no encontrara un # significa que la subcadena derecha era la última de w, por lo que encontraríamos un carácter _ y de ahí deberíamos movernos a su izquierda, pasar al estado $q_{borra\$}$. Desde el estado $q_{borra\$}$ nos vamos moviendo a la izquierda buscando el $ cambiando los a's por 0's y los b's por 1's. Cuando encontramos el $ lo cambiamos a #, nos movemos a su derecha y pasamos al estado $q_{poner\$}$ que busca el primer # y lo cambia por $, se mueve a su derecha y pasamos al estado $q_{primeraX}$, con lo que nos encontramos como al principio pero con la primera subcadena ya comparada con todas las subcadenas a su derecha, es decir, se ha terminado un ciclo completo de comparación. Es importante destacar que cuando entramos en el estado $q_{poner\$}$ es obligatorio tener algún # a la derecha porque hemos llegado a este estado desde $q_{borra\$}$, que a su vez viene de q_{ponerX}, y éste viene de q_{borraX}, lo que garantiza que hay un # a la derecha del $ que se está borrando.

Falta explicar el final de la máquina de Turing, pero queda claro cómo va a ser, una vez descrito el ciclo anterior. Cuando $ esté reemplazando al penúltimo #, X al último y hayamos terminado de comparar esas dos subcadenas, nos encontraremos en el estado q_{borraX} con el cabezal sobre el último carácter de w. Yendo hacia la izquierda, se cambia X por #, pasamos a q_{ponerX} que intenta cambiar un # a la derecha por X pero no encuentra ninguno, con lo que pasamos a $q_{borra\$}$, que pone el $ de vuelta como un # y el último # de w lo pone como $ pasando al estado $q_{primeraX}$. Pero como ya no quedan más # en w, encontramos un carácter _ y la cadena w es aceptada porque todas las subcadenas son efectivamente distintas.

El proceso que hemos explicado, queda explicitado en la siguiente tabla de transiciones:

$\delta(q_{ini}, \#) = (q_{primeraX}, \$, R)$

$\delta(q_{primeraX}, \_) = (q_{acc}, \_, R)$

$\delta(q_{primeraX}, \#) = (q_{busca\$}, X, L)$

$\delta(q_{primeraX}, 0) = (q_{primeraX}, 0, R)$

$\delta(q_{primeraX}, 1) = (q_{primeraX}, 1, R)$

$\delta(q_{busca\$}, 0) = (q_{busca\$}, 0, L)$

$\delta(q_{busca\$}, 1) = (q_{busca\$}, 1, L)$

$\delta(q_{busca\$}, a) = (q_{busca\$}, a, L)$

$\delta(q_{busca\$}, b) = (q_{busca\$}, b, L)$

$\delta(q_{busca\$}, \#) = (q_{busca\$}, \#, L)$

$\delta(q_{busca\$}, \$) = (q_{leer}, \$, R)$

$\delta(q_{leer}, a) = (q_{leer}, a, R)$

$\delta(q_{leer}, b) = (q_{leer}, b, R)$

$\delta(q_{leer}, 0) = (q_0, a, R)$

$\delta(q_{leer}, 1) = (q_1, b, R)$

$\delta(q_{leer}, \#) = (q_f, \#, R)$

$\delta(q_{leer}, X) = (q_{Xf}, X, R)$

$\delta(q_0, 0) = (q_0, 0, R)$

$\delta(q_0, 1) = (q_0, 1, R)$

$\delta(q_0, \#) = (q_0, \#, R)$

$\delta(q_0, X) = (q_{X0}, X, R)$

$\delta(q_1, 0) = (q_1, 0, R)$

$\delta(q_1, 1) = (q_1, 1, R)$

$\delta(q_1, \#) = (q_1, \#, R)$

$\delta(q_1, X) = (q_{X1}, X, R)$

$\delta(q_{X0}, a) = (q_{X0}, a, R)$

$\delta(q_{X0}, b) = (q_{X0}, b, R)$

$\delta(q_{X0}, 0) = (q_{busca\$}, a, L)$

$\delta(q_{X0}, 1) = (q_{borraX}, 1, L)$

$\delta(q_{X0}, \#) = (q_{borraX}, \#, L)$

$\delta(q_{X0}, \_) = (q_{borraX}, \_, L)$

$\delta(q_{X1}, a) = (q_{X1}, a, R)$

$\delta(q_{X1}, b) = (q_{X1}, b, R)$

$\delta(q_{X1}, 1) = (q_{busca\$}, b, L)$

$\delta(q_{X1}, 0) = (q_{borraX}, 0, L)$

$\delta(q_{X1}, \#) = (q_{borraX}, \#, L)$

$\delta(q_{X1}, \_) = (q_{borraX}, \_, L)$

$\delta(q_f, 0) = (q_f, 0, R)$

$\delta(q_f, 1) = (q_f, 1, R)$

$\delta(q_f, \#) = (q_f, \#, R)$

$\delta(q_f, \$) = (q_f, \$, R)$

$\delta(q_f, X) = (q_{Xf}, X, R)$

$\delta(q_{Xf}, a) = (q_{Xf}, a, R)$

$\delta(q_{Xf}, b) = (q_{Xf}, b, R)$

$\delta(q_{Xf}, 0) = (q_{borraX}, 0, L)$

$\delta(q_{Xf}, 1) = (q_{borraX}, 1, L)$

$\delta(q_{borraX}, 0) = (q_{borraX}, 0, L)$

$\delta(q_{borr\quad}, 1) = (q_{borraX}, 1, L)$

$\delta(q_{borraX}, a) = (q_{borraX}, a, L)$

$\delta(q_{borraX}, b) = (q_{borraX}, b, L)$

$\delta(q_{borraX}, X) = (q_{ponerX}, \#, R)$

$\delta(q_{ponerX}, 0) = (q_{ponerX}, 0, R)$

$\delta(q_{ponerX}, 1) = (q_{ponerX}, 1, R)$

$\delta(q_{ponerX}, a) = (q_{ponerX}, a, R)$

$\delta(q_{ponerX}, b) = (q_{ponerX}, b, R)$

$\delta(q_{ponerX}, \#) = (q_{retroceso}, X, L)$

$\delta(q_{ponerX}, \_) = (q_{borra\$}, \_, L)$

$\delta(q_{retroceso}, 0) = (q_{retroceso}, 0, L)$

$\delta(q_{retroceso}, 1) = (q_{retroceso}, 1, L)$

$\delta(q_{retroceso}, a) = (q_{retroceso}, 0, L)$

$\delta(q_{retroceso}, b) = (q_{retroceso}, 1, L)$

$\delta(q_{retroceso}, \#) = (q_{retroceso}, \#, L)$

$\delta(q_{retroceso}, \$) = (q_{leer}, \$, R)$

$\delta(q_{borra\$}, 0) = (q_{borra\$}, 0, L)$

$\delta(q_{borra\$}, 1) = (q_{borra\$}, 1, L)$

$\delta(q_{borra\$}, a) = (q_{borra\$}, 0, L)$

$\delta(q_{borra\$}, b) = (q_{borra\$}, 1, L)$

$\delta(q_{borra\$}, \#) = (q_{borra\$}, \#, L)$

$\delta(q_{borra\$}, \$) = (q_{poner\$}, \#, R)$

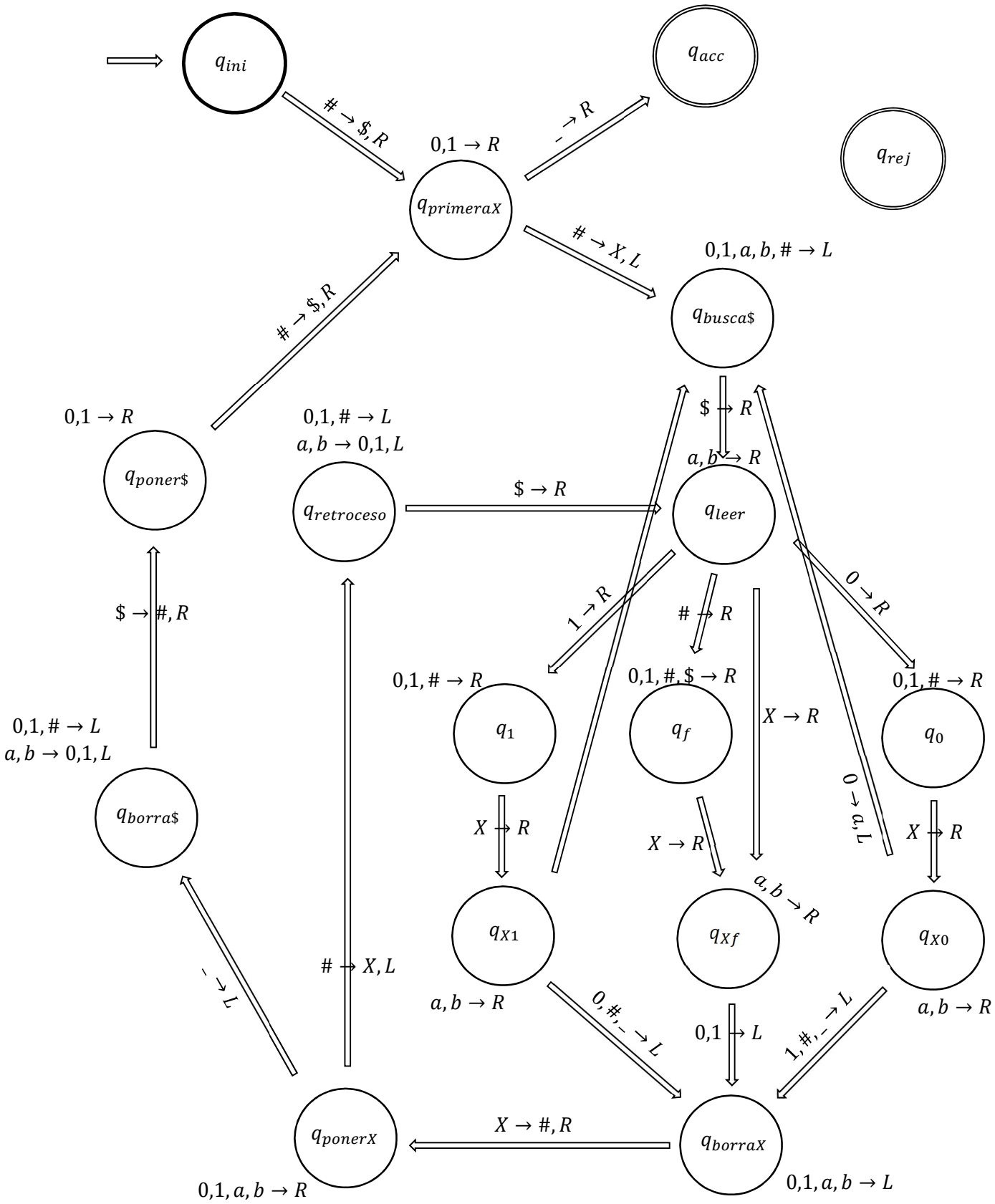
$\delta(q_{poner\$}, 0) = (q_{poner\$}, 0, R)$

$\delta(q_{poner\$}, 1) = (q_{poner\$}, 1, R)$

$\delta(q_{poner\$}, \#) = (q_{primeraX}, \$, R)$

Máquina de estados:

La función de transición se ve más claramente en el diagrama de la máquina de estados, donde cualquier transición no explicitada, lleva al estado de rechazo.



Demostración por inducción:

Veamos que la mT acepta las cadenas del lenguaje por inducción sobre $l \geq 1$. Sea $w = \#x_1$, entonces tendremos $q_{ini}\#x_1 \rightarrow \$q_{primeraX}x_1 \overset{*}{\rightarrow} \$x_1 q_{primeraX\_} \rightarrow \$x_1\_q_{acc}$. Supongamos que la mT acepta todas las cadenas con k subcadenas distintas dos a dos, $w = \#x_1 \dots \#x_k$. Consideremos una cadena $w = \#x_1 \dots \#x_k\#x_{k+1}$ del lenguaje, es decir, todas las subcadenas son distintas dos a dos. Entonces tendremos

$$q_{ini}\#x_1 \dots \#x_k\#x_{k+1} \rightarrow \$q_{primeraX}x_1 \dots \#x_k\#x_{k+1} \overset{*}{\rightarrow} \$x_1 q_{primeraX}\# \dots \#x_k\#x_{k+1}$$

$$\rightarrow \$x_1 q_{busca\$}\sigma X x_2 \dots \#x_k\#x_{k+1} \overset{*}{\rightarrow} q_{busca\$}\$x_{1,1} \dots x_{1,n_1}Xx_{2,1} \dots x_{2,n_2}\# \dots \#x_k\#x_{k+1}$$

$$\rightarrow \$q_{lee}x_{1,1} \dots x_{1,n_1}Xx_{2,1} \dots x_{2,n_2}\# \dots \#x_k\#x_{k+1}$$

Ahora pasa a comparar $x_{1,i}$ con $x_{2,i}$. Mientras haya coincidencia vuelve al \$ y realiza una nueva comparación hasta que encuentra una lectura diferente entre $x_{1,i}$ y $x_{2,i}$ o una cadena termina antes que la otra. Esto va a ocurrir porque de lo contrario ambas subcadenas serían iguales en contra de la suposición $w \in L$. Cuando encuentre dicha discrepancia (bien leyó 0 pero encuentra 1 o #, bien leyó 1 pero encuentra 0 o #, o bien leyó # y encuentra 0 o 1) pasa a cambiar X a la siguiente subcadena a la derecha, es decir,

$$\overset{*}{\rightarrow} \$x_{1,1} \dots x_{1,n_1}Xx_{2,1} \dots x_{2,j}q_{borraX}x_{2,j+1} \dots x_{2,n_2}\# \dots \#x_k\#x_{k+1}$$

$$\overset{*}{\rightarrow} \$x_{1,1} \dots x_{1,n_1}q_{borraX}Xx_{2,1} \dots x_{2,n_2}\# \dots \#x_k\#x_{k+1}$$

$$\rightarrow \$x_{1,1} \dots x_{1,n_1}\#q_{ponerX}x_{2,1} \dots x_{2,n_2}\# \dots \#x_k\#x_{k+1} \quad \overset{*}{\rightarrow} \$x_1\#x_2 q_{ponerX}\# \dots \#x_k\#x_{k+1}$$

$$\rightarrow \$x_1\#x_2 q_{retroceso}\sigma X \dots \#x_k\#x_{k+1}$$

Retrocedemos hasta encontrar \$ cambiando a's por 0's y b's por 1's. Cuando encontremos \$ los caracteres de cada x_i estarán como al principio, salvo que el primer # será un \$ y el tercero será una X. Procedemos ahora a comparar x_1 con $x_3, \dots, x_{l-1}$ de forma totalmente análoga. Cuando lleguemos a la comparación con x_l también habrá una discrepancia entre x_1 y x_l por ser $w \in L$. Cuando se encuentre, pasaremos, como ocurría con las otras subcadenas a cambiar X por # y pasar al estado q_{ponerX}, pero esta vez encuentra _ en vez de #, que es el indicativo de que x_1 ha sido comparada con todas las subcadenas a su derecha. Pasamos al estado $q_{borrar\$}$ que restaura el \$ por un # a la par que restaura las a's a 0's y las b's a 1's.

$\overset{*}{\rightarrow} \$x_1\#x_2\#\ldots\#x_k\#x_{k+1}q_{ponerX-} \rightarrow \$x_1\#x_2\#\ldots\#x_k\#x_{k+1,1}q_{borrar\$}x_{k+1,n_{k+1}-}$

$\overset{*}{\rightarrow} q_{borrar\$}\$x_1\#x_2\#\ldots\#x_k\#x_{k+1} \rightarrow \#q_{poner\$}x_1\#x_2\#\ldots\#x_k\#x_{k+1}$

$\overset{*}{\rightarrow} \#x_1q_{poner\$}\#x_2\#\ldots\#x_k\#x_{k+1} \rightarrow \#x_1\$q_{primeraX}x_2\#\ldots\#x_k\#x_{k+1}$

Y estamos en una situación que terminará siendo de aceptación por hipótesis de inducción. Por tanto, la cadena será aceptada.

Supongamos ahora que w es una cadena que no es del lenguaje, es decir, hay al menos dos subcadenas que son exactamente iguales. Lo primero que observamos es que, para este caso, se necesita $l \geq 2$, es decir, L admite todas las cadenas de la forma $w = \#x_1$.

Si tenemos únicamente dos subcadenas y ambas son iguales, tenemos

$q_{ini}\#x_1\#x_2 \rightarrow \$q_{primeraX}x_1\#x_2 \overset{*}{\rightarrow} \$x_1q_{primeraX}\#x_2 \rightarrow \$x_{1,1}q_{busca\$}x_{1,n_1}Xx_2$

$\overset{*}{\rightarrow} q_{busca\$}\$x_{1,1}\ldots x_{1,n_1}Xx_{2,1}\ldots x_{2,n_2} \rightarrow \$q_{lee}x_{1,1}\ldots x_{1,n_1}Xx_{2,1}\ldots x_{2,n_2}$

Comparará término a término hasta llegar al final, en el que encontramos un blanco y se rechaza:

$\overset{*}{\rightarrow} \$\sigma_{1,1}\ldots\sigma_{1,n_1}q_{lee}X\sigma_{2,1}\ldots\sigma_{2,n_2} \rightarrow \$\sigma_{1,1}\ldots\sigma_{1,n_1}Xq_{Xf}\sigma_{2,1}\ldots\sigma_{2,n_2}$

$\overset{*}{\rightarrow} \$\sigma_{1,1}\ldots\sigma_{1,n_1}X\sigma_{2,1}\ldots\sigma_{2,n_2}q_{Xf-} \rightarrow \$\sigma_{1,1}\ldots\sigma_{1,n_1}X\sigma_{2,1}\ldots\sigma_{2,n_2-}q_{rej}$

Supongamos que la mT rechaza todas las cadenas $w = \#x_1 \ldots \#x_k$ que no son del lenguaje. Consideremos una cadena, $w = \#x_1 \ldots \#x_k\#x_{k+1}$, con $k+1$ subcadenas, que no es del lenguaje.

Entonces, al igual que hicimos en el comienzo de esta demostración, se irá comparando x_1 con cada una de las subcadenas a su derecha. Si hay alguna subcadena que es igual a x_1, supongamos que la primera a la derecha de x_1 es x_i. Por tanto, si $|x_1| = n_1$ y $|x_i| = n_i$ tendremos que $n_1 = n_i$ y que $x_{1,s} = x_{i,s}$ para todo $s = 1, \ldots, n_1$.

Se irá comparando x_1 con cada subcadena a su derecha. Al llegar a la comparación de x_1 con x_i tendremos

$q_{ini}\#x_1 \ldots \#x_i \ldots \#x_{k+1} \rightarrow \$q_{primeraX}x_1 \ldots \#x_i \ldots \#x_{k+1}$

$\overset{*}{\rightarrow} \$q_{lee}x_{1,1} \dots x_{1,n_1}\# \dots Xx_{i,1} \dots x_{i,n_i}\# \dots \#x_{k+1} \overset{*}{\rightarrow} \$\sigma_{1,1} \dots \sigma_{1,n_1}q_{lee}\# \dots X\sigma_{i,1} \dots \sigma_{i,n_i}\# \dots \#x_{k+1}$

$\rightarrow \$\sigma_{1,1} \dots \sigma_{1,n_1}\#q_f \dots X\sigma_{i,1} \dots \sigma_{i,n_i}\# \dots \#x_{k+1} \overset{*}{\rightarrow} \$\sigma_{1,1} \dots \sigma_{1,n_1}\# \dots Xq_{Xf}\sigma_{i,1} \dots \sigma_{i,n_i}\# \dots \#x_{k+1}$

$\overset{*}{\rightarrow} \$\sigma_{1,1} \dots \sigma_{1,n_1}\# \dots X\sigma_{i,1} \dots \sigma_{i,n_i}q_{Xf}\# \dots \#x_{k+1} \rightarrow \$\sigma_{1,1} \dots \sigma_{1,n_1}\# \dots X\sigma_{i,1} \dots \sigma_{i,n_i}\#q_{rej} \dots \#x_{k+1}$

Si x_1 fuese distinto a todas las subcadenas de su derecha, llegaríamos a la situación

$q_{ini}\#x_1 \dots \#x_k\#x_{k+1} \overset{*}{\rightarrow} \#x_1\$q_{primeraX}x_2\# \dots \#x_k\#x_{k+1}$

Es decir, comparar x_2 con las subcadenas de su derecha. Pero si $w \notin L$ y x_1 es distinta que todas las subcadenas de su derecha, entonces $w' = \#x_2 \dots \#x_{k+1} \notin L$, por lo que llegaríamos a rechazo por hipótesis de inducción, al tener w' k subcadenas.

Complejidad de la máquina de Turing:

En cuanto a la complejidad, la complejidad espacial es claramente n, siendo $|w| = n$, ya que, en el peor de los casos hay que recorrer toda la cadena (caso de aceptación, por ejemplo).

En lo que se refiere a la complejidad temporal vamos considerar que las subcadenas son más o menos del mismo tamaño, $|x_i| = \frac{|w|}{l} = \frac{n}{l}$, lo que no está muy alejado de la realidad en el peor caso. Entonces tenemos, desde que $ está marcado:

- Cambiar # por X y volver al $: $2\frac{n}{l}$ pasos.
- $2\left(\frac{n}{l}+1\right)$ para comparar el primer carácter y volver, $2\left(\frac{n}{l}+2\right)$ para el segundo, y así hasta $2\left(\frac{n}{l}+\frac{n}{l}\right)$ para el último. En total tenemos $2\left(\frac{n^2}{l^2}+\frac{\frac{n}{l}\left(\frac{n}{l}+1\right)}{2}\right) = 3\frac{n^2}{l^2}+\frac{n}{l}$ pasos para la comparación de x_1 con x_2.
- Al comparar x_1 con otra subcadena x_i tenemos más o menos lo mismo, salvo que por cada comparación de carácter hay que recorrer desde x_1 hasta x_{i-1} en vez de solo x_1 como antes. Así, ahora queda $2\left((i-1)\frac{n}{l}+1\right)+2\left((i-1)\frac{n}{l}+2\right)+\cdots+2\left((i-1)\frac{n}{l}+\frac{n}{l}\right)$, que nos da un número de pasos igual a $2\left[(i-1)\frac{n^2}{l^2}+\frac{\frac{n}{l}\left(\frac{n}{l}+1\right)}{2}\right] = (2i-1)\frac{n^2}{l^2}+\frac{n}{l} \leq 2i\frac{n^2}{l^2}+\frac{n}{l}$.
- Al terminar la comparación, volvemos a X, la cambiamos por X y cambiamos el # a la derecha de la cadena por X, lo que nos lleva $2\frac{n}{l}$ pasos que hay que sumar. Tenemos pues $2i\frac{n^2}{l^2}+3\frac{n}{l}$.
- Para hallar el número de pasos de comparar x_1 con todas las subcadenas a su derecha hay que sumar en i, desde 2 hasta l, es decir, tenemos $\sum_{i=2}^{l}\left(2i\frac{n^2}{l^2}+3\frac{n}{l}\right)$, que nos da la cantidad $2\frac{n^2}{l^2}\cdot\frac{l(l+1)}{2}+3n$, (un poco menos porque no está el sumando $i=1$, pero no importa porque con la notación O no importa un coeficiente u otro y esta cantidad es mayor que la real en cualquier caso). Como el segundo sumando es de un grado menos que el primero, podemos acotar y decir que un ciclo completo tiene algo menos de $2\frac{n^2(l+1)}{l}$ pasos.
- Ahora habría que sumar los pasos necesarios para cambiar $ por # y poner el siguiente # como $ y el siguiente # como X, lo que nos da unos n pasos. De nuevo, lo ignoramos por ser un sumando de grado n, menor al término que llevamos, n^2.
- Solo faltaría repetir el ciclo de comparar una cadena cualquiera con las que tiene a su derecha. La diferencia está en que en cada ciclo queda una subcadena menos que comparar, así que tenemos un total de $2\left(\frac{n^2(l+1)}{l}+\frac{n^2 l}{l}+\frac{n^2(l-1)}{l}+\cdots+\frac{n^2 2}{l}\right)$, que nos da un total de $2\frac{n^2}{l}\cdot\frac{(l+1)(l+2)}{2}$ pasos.

A primera vista vemos que $l \leq n$ ya que no puede haber más símbolos # que la longitud de la propia cadena w, de donde obtenemos complejidad temporal $O(n^3)$.

Observaciones:

- Sobre el diseño de esta máquina de Turing, es claro que, una vez comparadas dos cadenas, es más eficiente cambiar el # a la derecha de la cadena por una X y, al volver hacia el \$, cambiar la antigua X por un #, pero salían más casos límite a tener en cuenta que complicaban el diseño y el orden de complejidad temporal no se ve alterado, por lo que se ha decidido dejar así.
- En la expresión final que nos queda al estudiar la complejidad, que esencialmente es $n^2 l$, hemos acotado $l \leq n$, lo que es obvio. No obstante, hay una relación entre l y n del tipo $l \log l \sim n$ de la que no hemos sido capaces de obtener algo provechoso. A esa relación se puede llegar mediante el siguiente razonamiento:

  Dado un número k, si tenemos dos caracteres (0 y 1), hay un máximo de 2^k cadenas distintas. Contando el #, saldrían $2^k(k+1)$ caracteres, es decir, n, siendo el número de # igual al número de cadenas, es decir, 2^k sería l. El peor caso sería una cadena vacía, 2 con un carácter, 4 con dos carácter, etc. Allá donde paremos, la suma de cadenas (luego de símbolos #) es menor que la siguiente potencia de 2. Esto nos lleva a una cantidad más o menos de $2^{k+1}(k+1)$ caracteres y 2^{k+1} símbolos #. De cualquier manera, no hemos sido capaces de obtener una acotación adecuada de l en términos de n a partir de $l \log l \sim n$.