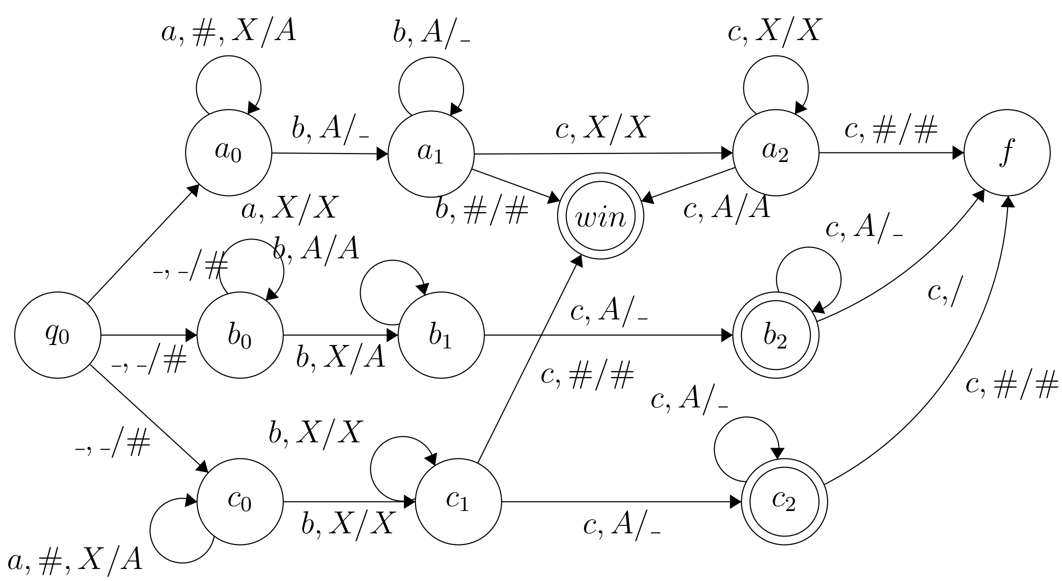
Tarea 3
Curso : Teoría de la Computación
Codigo : CC3102-1
Fecha : 8 de noviembre de 2017
Autor : Bastián
Mail : mail@gmail.com

1. Probar pertenencia a Lenguajes libres de contexto

1.1. $L_1 = \{a^m b^n c^p \,:\, m \neq n \vee n \neq p \vee m \neq p\}$

Dado a que L_1 es la unión de los lenguajes $\{m \neq n\}$, $\{n \neq p\}$ y $\{m \neq p\}$, los cuales son libres de contexto. Tenemos entonces que L_1 es libre de contexto por las propiedades de clausura de los lenguajes libres de contexto.



1.2. $L_2 = \{a^{n^2} \,:\, n \geq 0\}$

Usando el lema de bombeo para lenguajes libres de contexto notamos que claramente al dividir solo importa la cantidad de $a's$ que son bombeadas en $u = uv^ixy^iz$, si consideramos m a las $a's$ que no serán bombeadas, tendríamos que bombear una cantidad tal que $m + |v|^i + |y|^i = n^2$ para algún $n \geq 0$. Luego, si m n^2 entonces nos sirve cualquier i que rompa la igualdad de arriba, si no entonces para $i = 0$ obtenemos el resultado deseado

1.3. $L_3 = \{a^m b^n c^p d^q \ : \ n = q \vee m \leq p \vee m + n = p + q\}$

Siguiendo el procedimiento de la sección 1.1 solo nos falta notar que alguno de los lenguajes que componen al lenguaje L_3 **no es libre de contexto**. Dado que esto no sucede, tenemos que el lenguaje si es libre de contexto.

1.4. $L_4 = \{a^m b^n c^p \ : \ m \neq n \wedge n \neq p \wedge m \neq p\}$

Usando el *lema de bombeo para lenguajes libres de contexto*, diremos sin perdida de generalidad que existe un $N \in \mathbb{N}$ tal que cada palabra w se puede dividir en $w = uvxyz$. Escogemos la palabra $w = a^{N+m} b^{N+n} c^{N+p}$ con $n, m, p \in \mathbb{N}$ distintos entre si (para que pertenezcan a L_1), la cual claramente cumple $|w| \geq 3N \geq N$.

Se deben analizar todos los casos posibles, los cuales son:

1. Si v e y están compuestos solo de símbolos a's entonces se puede escoger un i tal que $N + m - |v \cdot y| + i \cdot |v \cdot y| = N + n$ o $N + m - |v \cdot y| + i \cdot |v \cdot y| = N + p$
2. Si v e y están compuestos solo de símbolos b's entonces es análogo al primer caso
3. Si v e y están compuestos solo de símbolos c's entonces es análogo al primer caso
4. Si v está compuesto solo por símbolos a's e y está compuesto solo por b's

2. Equivalencia entre AC y MT

Dado es trivial ver que una MT puede simular a un AC no ahondaremos en eso, un ejemplo es una MT de doble cinta que es infinita hacia los dos lados.

Ahora, un AC puede simular a una MT guardando los elementos a la *izquierda del cabezal* en un stack, y los que se encuentran a la *derecha del cabezal* en el otro. En cada stack, los elementos más cercanos a el cabezal del MT se encuentran más cerca del *top* del stack, mientras que los más lejanos se encuentran en el fondo de este.

Así mostramos que cada una puede simular a la otra, completando la demostración.

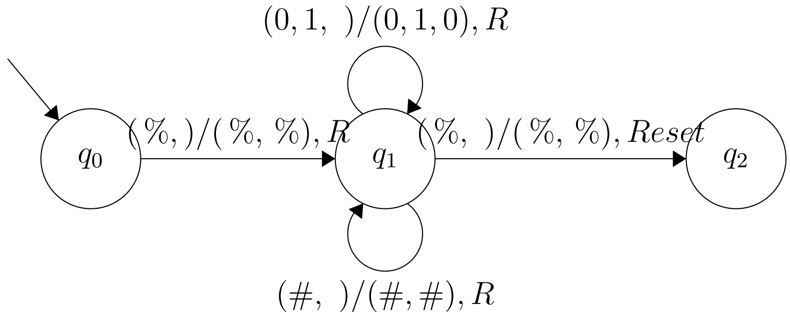
3. Ecuación de `NAND`

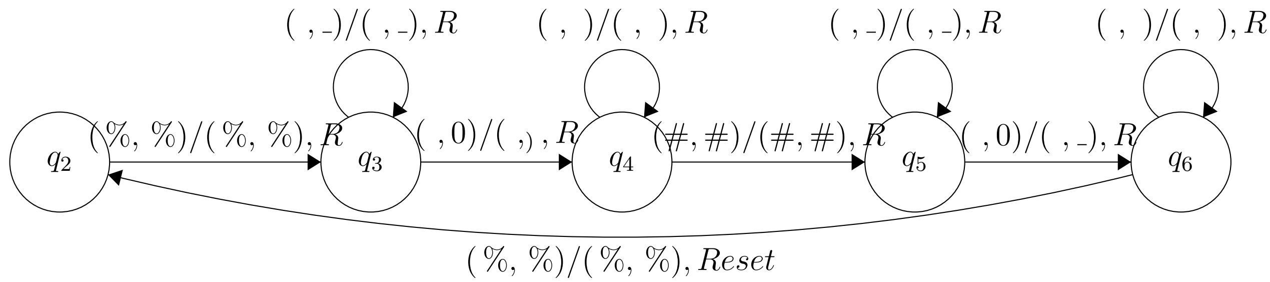
Se tiene una máquina de Turing que en la primera pasada por cada numero en x escribe un 0 en la segunda cinta, luego copia el signo # para seguir escribiendo un 0 en la segunda lista por cada elemento de y, para luego simular en la segunda cinta a una MT que verifica $|x| = |y|$.

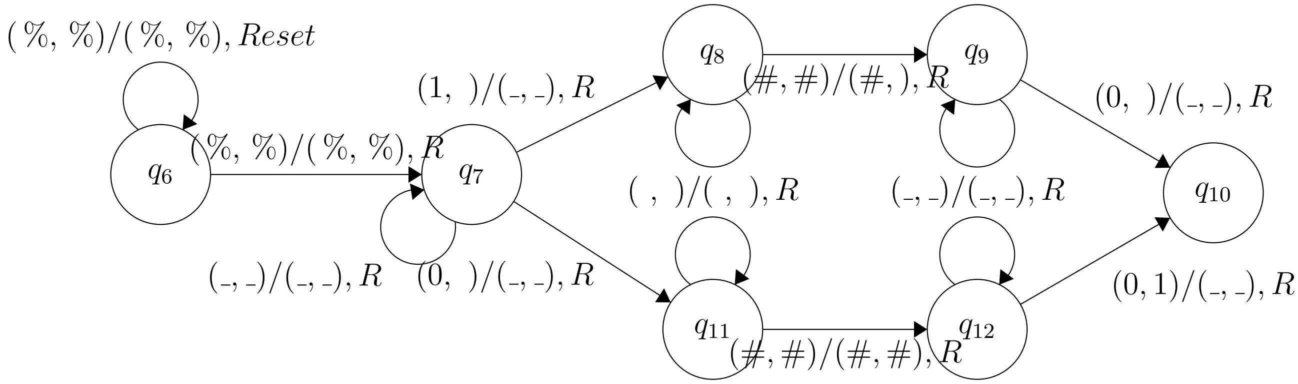
A partir de esto, la MT pasa a verificar que $z = x \uparrow$, comparando x_0 y y_0 con z_0 y así sucesivamente $|z|$ veces.

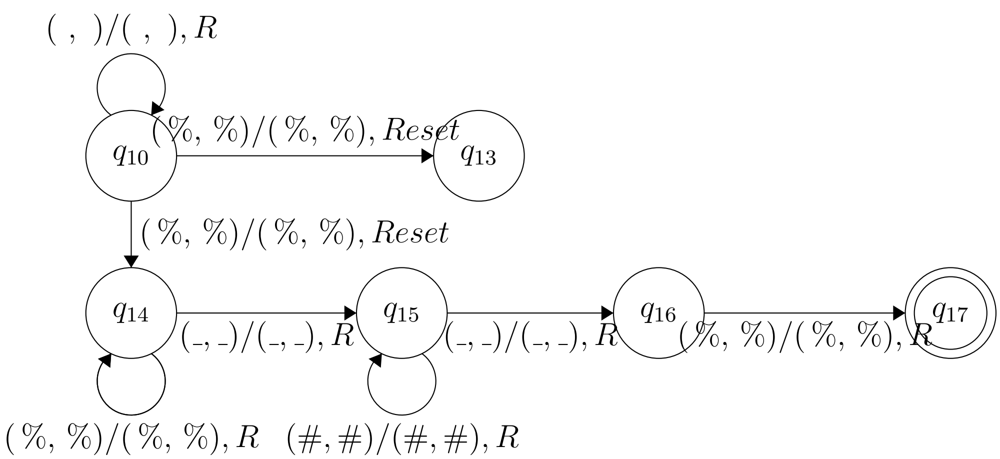
Para el diagrama de estados se asume que el input original está rodeado por el caracter especial % para simplificar la vida. Además de dejar un espacio cada vez que no importa el caracter y escribir ␣ cuando se borra el contenido.

Los estados con nombre repetido son el mismo y se ha diagramado así adrede para legibilidad.


(0, 1,)/(0, 1, 0), R
q0
(%,)/(%, %), R
q1
(%,)/(%, %), Reset
q2
(#,)/(#, #), R


(, _)/(, _), R
(,)/(,), R
(, _)/(, _), R
(,)/(,), R
q2
(%, %)/(%, %), R
q3
(, 0)/(,) , R
q4
(#, #)/(#, #), R
q5
(, 0)/(, _), R
q6
(%, %)/(%, %), Reset


(%, %)/(%, %), Reset
q6
(%, %)/(%, %), R
q7
(1,)/(_, _), R
(_, _)/(_, _), R
(0,)/(_, _), R
q8
(#, #)/(#,), R
q9
(,)/(,), R
(_, _)/(_, _), R
(0,)/(_, _), R
q10
q11
(#, #)/(#, #), R
q12
(0, 1)/(_, _), R


(,)/(,), R
q10
(%, %)/(%, %), Reset
q13
(%, %)/(%, %), Reset
q14
(_, _)/(_, _), R
q15
(_, _)/(_, _), R
q16
(%, %)/(%, %), R
q17
(%, %)/(%, %), R
(#, #)/(#, #), R

4. Clausura de los lenguajes Turing-Reconocibles

Los lenguajes Turing-reconocibles son aquellos que para todo string finito es capaz de responder $\{``Acepta'', ``Rechaza'', ``Loopea''\}$

4.1. Concatenación

Al "pegar" dos autómatas simplemente reaccionamos según los resultados que entrega, de la siguiente forma:

$M_1 \cdot M_2$	Acepta	Rechaza	Loopea
Acepta	Acepta	Rechaza	Loopea
Rechaza	Rechaza	Rechaza	Loopea
Loopea	Loopea	Loopea	Loopea

Luego, dado a que no se producen otros resultados más que {"Acepta","Rechaza","Loopea"}

4.2. Intersección

Simplemente utilizamos una Maquina M que simula a las máquinas M_1 y M_2, y entrega (al igual que en la pregunta anterior) el resultado según la respuesta de las máquinas M_1 y M_2.

$M_1 \cap M_2$	Acepta	Rechaza	Loopea
Acepta	Acepta	Rechaza	Loopea
Rechaza	Rechaza	Rechaza	Loopea
Loopea	Loopea	Loopea	Loopea