

## Practica 4

- 1) Construir una máquina de Turing que escriba en la primera cinta las palabras de  $\{0,1\}^*$  en orden canónico separadas por un símbolo ";". Obviamente esta máquina nunca se detiene.

$q_0, B, B$

$q_1, ,, D, 0, S$

$q_1, B, 0$

$q_1, 0, D, 0, D$

$q_1, B, 1$

$q_1, 1, D, 1, D$

$q_1, B, B$

$q_3, ,, S, B, I$

$q_3, ,, 0$

$q_4, ,, S, 1, I$

$q_3, ,, 1$

$q_5, ,, S, 0, I$

$q_5, ,, B$

$q_1, ,, D, 0, S$

$q_5, ,, 0$

$q_4, ,, S, 1, I$

$q_5, ,, 1$

$q_5, ,, S, 0, I$

$q_4, ,, 0$

$q_4, ,, S, 0, I$

$q_4, ,, 1$

$q_4, S, 1, I$

$q_4, B$

$q_1, D, B, D$

2) Sean  $\Sigma = \{a, b\}$  y  $\mathcal{L}$  el conjunto de todos los lenguajes definidos sobre  $\Sigma$ . Diga si las siguientes afirmaciones son verdaderas o falsas:

a)  $\mathcal{L} - R = \emptyset$

Falso

$L_D \in \mathcal{L}$  pero no a  $R$ , por lo que si se hace  $\mathcal{L} - R$  por lo menos  $L_D$  seguirá estando en  $\mathcal{L}$

b)  $\Sigma^* \in R$

Verdadera

Existe una MT tal que acepta el lenguaje  $\Sigma^*$  ( $q_0 w \vdash q_A$ )

c)  $ab \in \Sigma^*$

Verdadera

$\Sigma^* = (\lambda, a, b, aa, ab, ba, bb, \dots)$

d)  $RE - R \neq \emptyset$

Verdadero

Por absurdo:  $RE - R = \emptyset$

$L_u \in (RE - R)$

$L_u \in \emptyset$  (ABSURDO)

$\therefore RE - R \neq \emptyset$

e)  $\emptyset \in RE$

Verdadero

Existe una MT que acepta el lenguaje  $\emptyset$  tal que ( $q_0 w \vdash q_R$ )  $\therefore \emptyset \in R$  y por definición  $\emptyset \in RE$

f)  $CO-R \subset CO-RE$

Verdadero

$L \in CO-R \Rightarrow (\text{def. Co-R})$

$L \in R \Rightarrow (\text{def. } R \text{ y RE})$

$L \in RE \Rightarrow (\text{def. Co-RE})$

$L \in \text{CO-RE}$

g)  $\{\lambda\} \in (\mathcal{L} - \text{CO-RE})$

Falso

Existe una MT tal que acepte el lenguaje  $\{\lambda\}$  y sea recursivo ( $q0w \vdash qA, w = \lambda$ ) o ( $q0w \vdash qR$ ))

$\therefore \{\lambda\} \in R \Rightarrow \{\lambda\} \in \text{CO-RE}$  (Teorema 3)

$\therefore \{\lambda\} \notin (\mathcal{L} - \text{CO-RE})$

h)  $\text{CO-RE} = \text{RE}$

Falso

Por absurdo:  $\text{CO-RE} - \text{RE} = \emptyset$

$L_D \in (\text{CO-RE} - \text{RE})$

$L_D \in \emptyset$  (ABSURDO)

$\therefore \text{CO-RE} \neq \text{RE}$

i)  $a \in R$

Falso

$R = \{\emptyset, \{\lambda\}, \{a\}, \{b\}, \{aa\}, \{ab\}, \dots\}$   $a \notin R$ .

j)  $\text{RE} \cup R = \mathcal{L}$

Falso

$\text{RE} \cup R = \text{RE}$

$L_D \in \mathcal{L}$  pero  $L_D \notin \text{RE}$

$\therefore \text{RE} \cup R \neq \mathcal{L}$

k)  $(\mathcal{L} - \text{RE}) = \text{CO-RE}$

Falso

Existe  $L = \{1w / w \in L_D\} \cup \{0w / w \notin L_D\}$  que  $\notin (\text{RE} \cup \text{CO-RE})$

Existe un lenguaje en  $\mathcal{L} - \text{RE}$  que no está en  $\text{CO-RE}$

$\therefore (\mathcal{L} - \text{RE}) \neq \text{CO-RE}, (\mathcal{L} - \text{RE}) = L \cup \text{CO-RE}$

l)  $\{a\} \in \text{RE}$

Verdadero

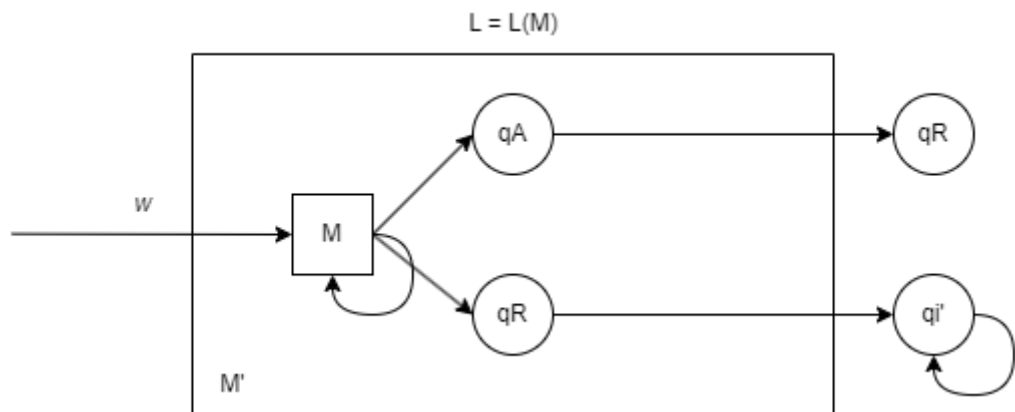
$\{a\} \in R \Rightarrow (\text{def. } R \text{ y } RE)$

$\{a\} \in RE$

3) Si  $L \in (RE - R)$

- a) ¿Existirá alguna máquina de Turing que rechace parando en  $q_R$  si su entrada está en  $L$  y rechace looppeando si su entrada no está en  $L$ ?

Si, sabiendo que tenemos una maquina que reconoce  $L$  y que a veces rechaza looppeando, podemos construir una maquina tomando la anterior y todas las  $\delta$  que terminan en estado  $q_R$  las redirijo a un looppeo y todas las que terminan en  $q_A$  las redirijo a  $q_R$



- b) ¿Existirá alguna máquina de Turing que rechace looppeando si su entrada está en  $L$  y rechace parando en  $q_R$  si su entrada no está en  $L$ ?

No, no se puede crear una máquina que, a partir de la maquina original, rechace looppeando si su entrada está en  $L$  y rechace parando en  $q_R$  si su entrada no está en  $L$ , porque la maquina original cuando la entrada no esta en  $L$  se puede quedar looppeando, no siempre para en  $q_R$ . Si rechaza looppeando no hay forma de identificar que la maquina entro en un loop como para rechazarla parando en  $q_R$

Demostración por absurdo

- $M$  (máquina que reconoce a  $L$ )
  - Si el string pertenece a  $L$  se detiene en  $q_A$
- $M_b$  (máquina del inciso b que rechaza looppeando si su entrada está en  $L$  y rechace parando en  $q_R$  si su entrada no está en  $L$ )
  - Si el string no pertenece a  $L$  se detiene en  $q_R$

Podría construir una nueva maquina si yo recibo un string lo hago pasar por estas dos máquinas paralelamente. O la primer maquina se va a detener en  $q_A$  o la segunda se va a detener en  $q_R$ . Para todo string alguna de las dos máquinas se va a detener.

Si tenemos garantizado que alguna de las maquinas se va a detener, podemos construir una nueva maquina que usa las dos máquinas mencionadas, que seguro se detiene y que me puede decir si un string pertenece o no a  $L$ .

Si podemos construir una maquina así, entonces tengo una maquina que reconoce  $L$  y siempre se detiene, eso significa que  $L \in R \rightarrow$  absurdo, partimos de que  $L \in (RE - R)$  y llegamos a la conclusión de que  $L \in R$ .

c) De existir, que lenguaje reconocería esta máquina de Turing.

Reconocería el  $\emptyset$  porque rechazaría los que  $L(M)$  acepta y también rechazaría los que  $L(M)$  rechaza.

4) Sea  $L = \{w \mid \text{Existe alguna Máquina de Turing } M \text{ que acepta } w\}$  ¿ $L \in R$ ? Justifique.

Para que una palabra quede afuera de  $L$ , tendría que existir una máquina de Turing que no acepte esa palabra. Todo los Strings de mi alfabeto van a tener al menos una máquina que lo acepte. No hay un String que va a quedar por fuera de  $L$ .

Para todo  $w$ , existe alguna Máquina de Turing  $M$  que acepta  $w$ .

-  $L = \Sigma^*$ .

$\therefore L \in R$ , porque puedo hacer una Maquina de Turing que me lee el primer símbolo carácter y automáticamente pare en  $q_A$ , esta maquina reconoce el lenguaje  $\Sigma^*$  y como lo reconoce y siempre se detiene podemos decir que  $\Sigma^* \in R$ .

5) Conteste y justifique:

a) ¿ $\mathcal{L}$  es un conjunto infinito contable?

No, es incontable ya que  $\mathcal{L} = \rho(\Sigma^*)$ , y ya está demostrado que  $\rho(\Sigma^*)$  es incontable.

b) ¿RE es un conjunto infinito contable?

RE es infinito contable ya que cada lenguaje de este conjunto puede asociarse con una máquina de Turing que lo reconozca, y el conjunto de máquinas de Turing es contable. Aunque un lenguaje puede ser aceptado por varias máquinas de Turing, aun podemos contar ordenadamente estas máquinas, demostrando que el conjunto de lenguajes recursivamente enumerables es infinito contable.

c) ¿ $\mathcal{L} - \text{RE}$  es un conjunto infinito contable?

- La unión de dos conjuntos contables da otro conjunto contable

$(\mathcal{L} - \text{RE}) \cup \text{RE}$  es contable  $\rightarrow$  Absurdo

$(\mathcal{L} - \text{RE}) \cup \text{RE} = \mathcal{L}$  y este es incontable, para que la unión sea incontable (que es lo que debería ser) lo que esta a la izquierda tiene que ser incontable (sabemos que RE si o si es contable).

d) Existe algún lenguaje  $L \in \mathcal{L}$ , tal que L sea infinito no contable

Todo lenguaje es subconjunto de  $\Sigma^*$ , dado que ya sabemos que  $\Sigma^*$  es infinito contable, un subconjunto de el va a ser a lo sumo infinito contable, por lo que no existe algún lenguaje L que  $\in \mathcal{L}$  tal que sea infinito no contable.

Demostración:

Sabemos que la unión de dos conjuntos contables da otro conjunto contable. También sabemos que  $\Sigma^*$  es infinito contable.

Supongamos que existe un L infinito no contable, esto llevaría a que  $L \cup L^c$  sea incontable (ya que en la unión, si al menos uno es incontable, la unión es incontable).

$L \cup L^c = \Sigma^* \therefore \Sigma^*$  es incontable  $\rightarrow$  Absurdo, ya que partimos sabiendo que  $\Sigma^*$  es infinito contable

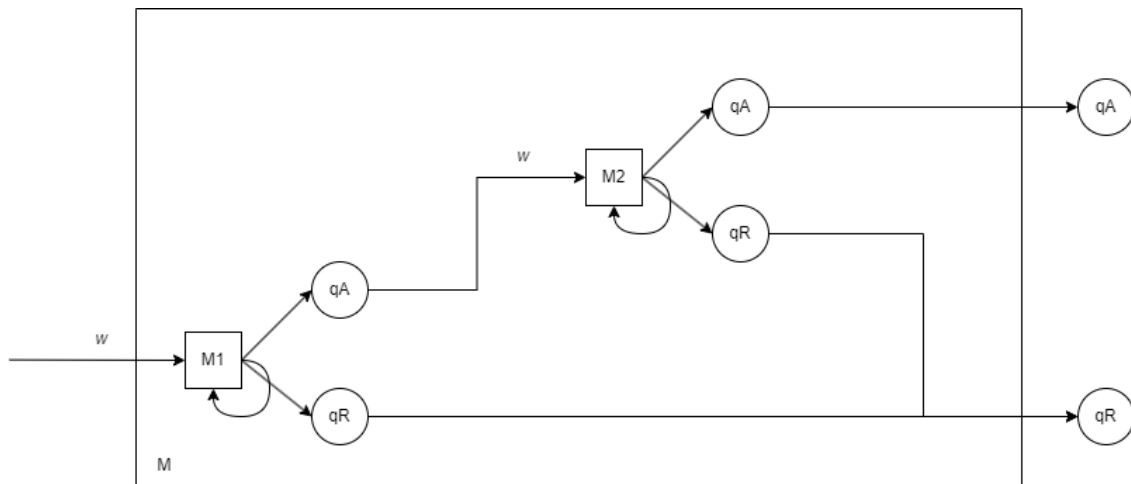
6) Sea  $L$  un lenguaje definido sobre  $\Sigma$ . Demostrar que:

a)  $L^c \notin R \Rightarrow L \notin R$

Si  $L^c \notin R \Rightarrow L \notin \text{CO-R}$  (definición CO-R)

$\Rightarrow L \notin R$  (En el teorema 1 y 2 de la teoría se llega a que  $R = \text{CO-R}$ )

b)  $(L1 \in RE) \text{ AND } (L2 \in RE) \Rightarrow L1 \cap L2 \in RE$



- $L1 = L(M1)$
- $L2 = L(M2)$
- $L = L(M)$

-  $L = L1 \cap L2$

M reconoce solo los inputs que son reconocidos por M1 y M2, es decir, reconoce los inputs que son reconocidos por ambas maquinas (la intersección).

Si se trata de un input que rechaza uno pero el otro no, M no lo va a aceptar. Si se trata de un input ambas maquinas, M lo va a rechazar. Por lo tanto  $L = L1 \cap L2$

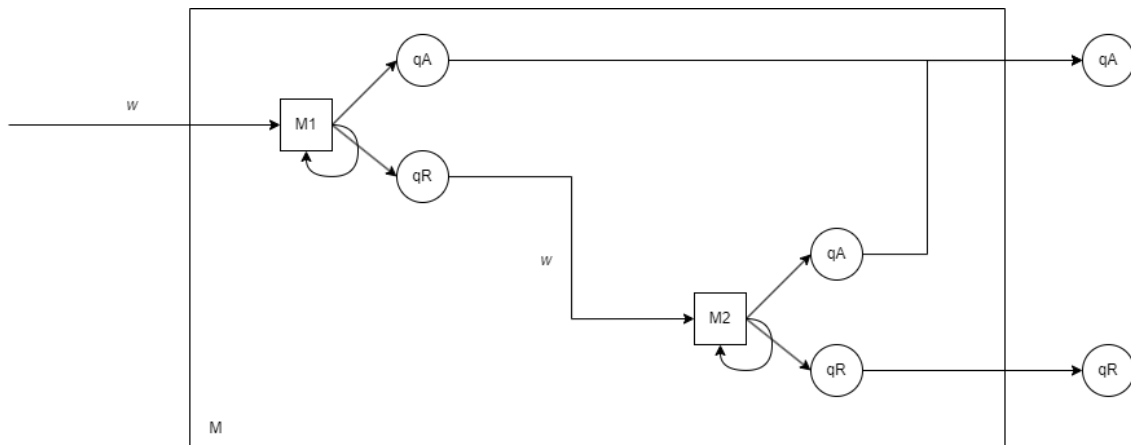
-  $L \in RE$

Como  $L = L(M)$ , existe una máquina que lo acepta.

Si M1 o M2 se quedan loopando M se va a quedar loopando.

(consultar)

c)  $(L1 \in RE) \text{ AND } (L2 \in RE) \Rightarrow L1 \cup L2 \in RE$



- $L1 = L(M1)$
- $L2 = L(M2)$
- $L = L(M)$

-  $L = L1 \cup L2$

M reconoce los inputs que son reconocidos por M1 o M2, es decir, reconoce los inputs que son reconocidos por alguna de las dos máquinas (la unión). Si se trata de un input que es rechazado por una maquina pero no por la otra, M lo va a aceptar. Si se trata de un input rechazado por ambas maquinas, M lo va a rechazar. Por lo tanto  $L = L1 \cup L2$

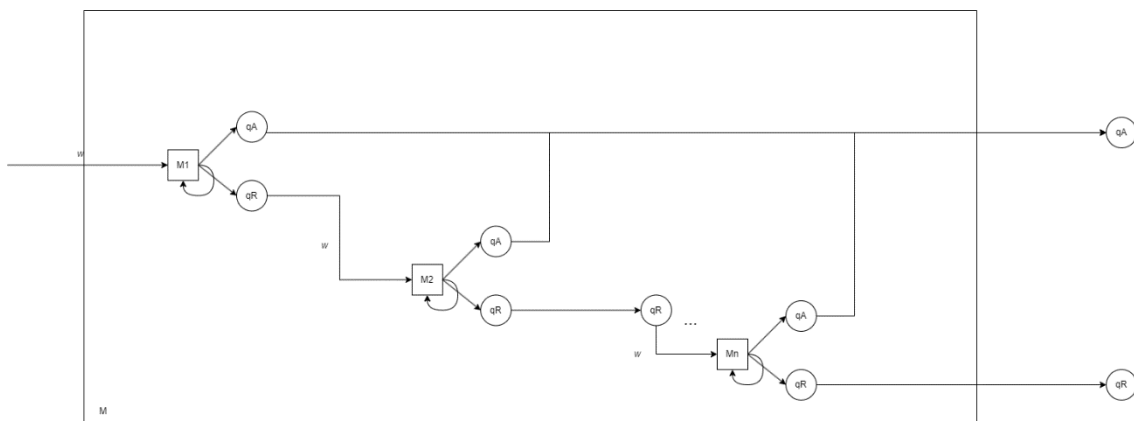
-  $L \in RE$

Como  $L = L(M)$ , existe una máquina que lo acepta.

Si M1 o M2 se quedan loopando M se va a quedar loopando.

(consultar)

- d) La unión de un número finito de lenguajes recursivamente enumerables es un lenguaje recursivamente enumerable.





- $L_1 = L(M_1)$
- $L_2 = L(M_2)$
- $L_n = L(M_n)$
- $L = L(M)$

$$- L = L_1 \cup L_2 \cup \dots \cup L_n$$

M reconoce los inputs que son reconocidos por alguna de todas máquinas (la unión). Si se trata de un input que es rechazado por una máquina pero no por alguna otra, M lo va a aceptar. Si se trata de un input rechazado por todas las máquinas, M lo va a rechazar. Por lo tanto  $L = L_1 \cup L_2 \cup \dots \cup L_n$

$$- L \in RE$$

Como  $L = L(M)$ , existe una máquina que lo acepta.

Si alguna de las máquinas se queda looppeando M se va a quedar looppeando.

7) Para los casos a), b) y c) del punto anterior ¿valen las recíprocas? Justifique

$$a) L \notin R \Rightarrow L^c \notin R$$

$L \notin R \Rightarrow L \notin CO-R$  (En el teorema 1 y 2 de la teoría se llega a que  $R = CO-R$ )

$$\Rightarrow L^c \notin R \text{ (definición CO-R)}$$

Vale

$$b) L_1 \cap L_2 \in RE \Rightarrow (L_1 \in RE) \text{ AND } (L_2 \in RE)$$

$$L_D \notin RE$$

$$L_D \cap L_D^c = \emptyset$$

$$\emptyset \in RE$$

$$\therefore L_D \cap L_D^c \in RE$$

No vale

$$c) L_1 \cup L_2 \in RE \Rightarrow (L_1 \in RE) \text{ AND } (L_2 \in RE)$$

$$L_D \notin RE$$

$$L_D \cup L_D^c = \Sigma^*$$

$\Sigma^* \in RE$

$\therefore L_D \cup L_D^c \in RE$

No vale

- 8) Si  $L$  es un subconjunto de un lenguaje recursivamente enumerable, ¿Puede afirmarse entonces que  $L$  es recursivamente enumerable? Justifique.

No, no puede afirmarse.

Contraejemplo:

$\Sigma^*$  es recursivamente enumerable.  $L_D$  es subconjunto de  $\Sigma^*$  pero  $L_D$  no es recursivamente enumerable.

- 9) Dado  $L_1$ , un lenguaje recursivo cualquiera

$$L_2 = \{ \langle M \rangle \mid L(M) = L_1^c \}$$

$$L_3 = \{ \langle M \rangle \mid L(M) = L_1^c \text{ y } M \text{ siempre se detiene} \}$$

Determine si  $(L_2 - L_3) = \emptyset$ . Justifique su respuesta.

$L_2$  son el conjunto de codificaciones de Maquinas de Turing que aceptan  $L_1^c$  y  $L_3$  son el conjunto de codificaciones de Maquinas de Turing que aceptan a  $L_1^c$  y siempre se detienen.

Para probar que  $(L_2 - L_3) \neq \emptyset$  basta con un **contraejemplo**

Supongamos que se tiene  $\Sigma = \{a,b\}$ ,  $L_1$  son todos los inputs que comienzan con  $a$ , por lo tanto  $L_1^c$  serian todos los inputs que comienzan con  $b$ .

Existe una  $\langle M \rangle$  que si el primer símbolo es un  $b$ , lo acepta, pero si no, se queda loopando. Esta codificación de máquina de Turing pertenece a  $L_2$  puesto que acepta a  $L_1^c$  pero se queda loopando, por lo que no pertenece a  $L_3$ .

Vemos que  $L_2 - L_3 \neq \emptyset$  (no son iguales)

- 10) Sean los lenguajes  $L = \{ \langle M \rangle \mid M \text{ siempre se detiene} \}$  y  $LR = \{ \langle M \rangle \mid L(M) \in R \}$ .

Cuál es la afirmación correcta:

- a)  $L \subset LR$
- b)  $L \supset LR$
- c)  $L = LR$

L son el conjunto de codificaciones de Maquinas de Turing que se detienen

LR son el conjunto de codificaciones de Maquinas de Turing que reconocen lenguajes que pertenecen a R.

Esto no quiere decir que las  $\langle M \rangle$  que pertenecen a LR siempre se detengan.

Que un lenguaje sea recursivo no quiere decir que todas las maquinas que lo aceptan se detienen siempre.

Aun así, LR contiene a las que siempre se detienen (porque son las que básicamente hacen que sus  $L(M) \in R$ ).

Respuesta  $L \subset LR$

*(consultar)*

11) Encuentre una justificación para cada una de las siguientes afirmaciones

a)  $\emptyset \in RE$

$\emptyset \in R$ , puesto que se puede construir una máquina de Turing que rechace todos los inputs (en el primer paso), el lenguaje que reconoce esa maquina es el  $\emptyset$ . Por definición si  $\emptyset \in R$ ,  $\emptyset \in RE$

b) Si L es un lenguaje formado por una sola palabra, entonces  $L \in R$

Si un lenguaje L consiste en una sola palabra w, entonces podemos construir una máquina de Turing que acepte w y rechace cualquier otra entrada. Dado que esta máquina de Turing siempre se detiene y decide L, esto significa que L está en la clase de lenguajes recursivos (R).

c) Si L es un lenguaje finito, entonces  $L \in R$

Si se tiene un lenguaje finito L puede ser reconocido por una máquina de Turing que simplemente acepta cualquier entrada que sea igual a una de las palabras del lenguaje finito y rechaza todas las demás. Dado que esta máquina de Turing siempre se detiene  $L \in R$ .

12) Demuestre que si el Halting Problem (HP) es un lenguaje recursivo entonces podría construirse una máquina de Turing que acepte el lenguaje universal Lu,

y que se detenga para todo  $w \in \Sigma^*$  ¿Qué puede decir entonces sobre la recursividad de HP?

$$L_u = \{ \langle M \rangle, w \mid M \text{ acepta } w \}$$

$$HP = \{ \langle M \rangle, w \mid M \text{ se detiene con input } w \}$$

Se puede probar que existe la reducción  $L_u \leq HP$ , y como  $L_u \notin R$  será cierto  $HP \notin R$ .

### **Demostración:**

MT  $M_f$  computa la función  $f$  de reducibilidad

$$M_f(\langle M \rangle, w) = \langle M' \rangle, w$$

Si  $\langle M \rangle, w$  no es un par válido o  $\langle M \rangle$  no es un código válido de MT borra la cinta. Caso contrario busca en las quintuplas de  $\langle M \rangle$  el estado  $q_R$  y lo reemplaza por un nuevo estado  $q$ . Después agrega las quintuplas  $(q, x, q, x, S)$  por cada símbolo  $x$  del alfabeto de la cinta de  $M$ . Así la máquina  $M'$  construida por  $M_f$  entra en loop cuando  $M$  para en  $q_R$ .

### **Probar:**

a)  $f$  es computable? Si, ya que  $M_f$  siempre se detiene debido a que la entrada es finita y luego de recorrerla agrega un número finito de quintuplas y se detiene

b)  $\langle M \rangle, w \in L_u \Leftrightarrow \langle M' \rangle, w \in HP$ ?

i.  $\langle M \rangle, w \in L_u \Rightarrow \langle M' \rangle, w \in HP$

$$\langle M \rangle, w \in L_u \Rightarrow M \text{ acepta } w$$

$$\Rightarrow M \text{ para en } q_A$$

$$\Rightarrow M' \text{ para en } q_A \text{ (por construcción)}$$

$$\Rightarrow M' \text{ se detiene con input } w$$

$$\Rightarrow \langle M' \rangle, w \in HP$$

ii.  $\langle M \rangle, w \notin L_u \Rightarrow \langle M' \rangle, w \notin HP$

- Si  $\langle M \rangle, w$  no es un par válido o  $\langle M \rangle$  no es un código válido de máquina de Turing  $\Rightarrow \langle M' \rangle, w = \lambda \Rightarrow \langle M' \rangle, w \notin HP$

- Caso contrario,  $M$  rechaza  $w \Rightarrow M$  loopea o para en  $q_R$  con input  $w \Rightarrow M'$  loopea en  $q$  con input  $w \Rightarrow \langle M' \rangle, w \notin HP$

13) Demuestre que  $LN \in RE$

$$LN = \{ \langle M \rangle \mid L(M) \neq \emptyset \}$$

$LN$  esta formado por codificaciones de máquinas de Turing que reconocen algún lenguaje.

Se puede construir una máquina de Turing  $M_{LN}$  que reciba como entrada un  $\langle M \rangle$ . Esta máquina de Turing va a simular la ejecución de  $\langle M \rangle$  con todas las posibles cadenas de entrada en orden canónico haciendo  $i$  pasos de  $\langle M \rangle$  en cada cinta (hay una cinta por entrada) con una ejecución en paralelo. Si  $\langle M \rangle$  acepta una cadena, entonces  $M_{LN}$  va a aceptar a  $\langle M \rangle$ . Si  $\langle M \rangle$  rechaza todas las cadenas (rechazo con loop), entonces se rechaza  $\langle M \rangle$  (rechaza a los que no pertenecen a  $LN$ , es decir, los que no reconocen ningún lenguaje, en este caso, la maquina nunc se va a detern por lo que se puede concluir además que  $LN \notin R$ )

Se construyo una máquina de Turing ( $M_{LN}$ ) que acepta  $LN$  por lo tanto  $LN \in RE$

(consultar)