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.

q0,B,B
q1,;,D,0,S

q1,B,0
q1,0,D,0,D

q1,B,1
q1,1,D,1,D

q1,B,B
q3,;,S,B,I

q3,;,0
q4,;,S,1,I

q3,;,1
q5,;,S,0,I

q5,;,B
q1,;,D,0,S

q5,;,0
q4,;,S,1,I

q5,;,1
q5,;,S,0,I

q4,;,0
q4,;,S,0,I

q4,;,1

q4,;,S,1,I

q4,;,B
q1,;,D,B,D

2) Sean Σ ={a,b} y $\mathcal{L}$ el conjunto de todos los lenguajes definidos sobre Σ. 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 $\Sigma *$ (q0w $\vdash$ qA)

c) $ab \in \Sigma *$
   Verdadera
   $\Sigma *$ = (λ,a,b,aa,ab,ba,bb,…)

d) $RE - R \neq \emptyset$
   Verdadero
   Por absurdo: $RE - R = \emptyset$
   $Lu \in (RE - R)$
   $Lu \in \emptyset$ (ABSURDO)
   ∴ $RE - R \neq \emptyset$

e) $\emptyset \in RE$
   Verdadero
   Existe una MT que acepta el lenguaje ∅ tal que (q0w $\vdash$ qR) ∴ $\emptyset \in R$ y por definición $\emptyset \in RE$

f) CO-R ⊂ CO-RE
   Verdadero
   L ∈ CO-R ⇒ (def. Co-R)

Ŀ ∈ R ⇒ (def. R y RE)

Ŀ ∈ RE ⇒ (def. Co-Re)

L ∈ CO-RE

g) {λ} ∈ (ℒ – CO-RE)

Falso

Existe una MT tal que acepte el lenguaje {λ} y sea recursivo (q0w ⊢ qA, w =λ) o (q0w ⊢ qR))

∴ {λ} ∈ R ⇒ {λ} ∈ CO-RE (Teorema 3)

∴ {λ} ∉ (ℒ – CO-RE)

h) CO-RE = RE

Falso

Por absurdo: CO-RE - RE = ∅

L_D ∈ (CO-RE - RE)

L_D ∈ ∅ (ABSURDO)

∴ CO-RE ≠ RE

i) a ∈ R

Falso

R = {∅, {λ}, {a}, {b}, {aa}, {ab}, …) *a ∉ R.*

*"a" no es un lenguaje. El conjunto R se compone de lenguajes (conjuntos de cadenas), no de cadenas individuales (*a es una cadena y {a} es el lenguaje que contiene la cadena a)

La diferencia entre "a ∉ R" y "{a} ∈ R" se debe a la forma en que se definen los lenguajes y las cadenas en la teoría de la computabilidad.

<u>a ∉ R (a no pertenece a R):</u>

En el contexto de la teoría de la computabilidad, el conjunto de lenguajes recursivos (R) está formado por lenguajes que son decidibles por una máquina de Turing, es decir, hay una máquina de Turing que puede aceptar o rechazar cada cadena de entrada en un tiempo finito. Un lenguaje se considera recursivo si hay una MT que siempre se detiene y decide si una cadena dada está en el lenguaje o no.

La notación "a ∉ R" significa que la cadena "a" no pertenece a ningún lenguaje recursivo. Esto implica que no existe una máquina de Turing que pueda tomar la cadena "a" como entrada y siempre detenerse en un tiempo finito para decidir si "a" está en el lenguaje o no. En otras palabras, "a" es una cadena que no es decidible por ninguna máquina de Turing.

<u>{a} ∈ R ({a} pertenece a R):</u>

Por otro lado, cuando tienes un conjunto que contiene una sola cadena, como "{a}", esta notación se refiere al lenguaje que contiene solo esa cadena. En este caso, el lenguaje es

{a}. El hecho de que "{a} ∈ R" significa que el lenguaje que contiene solo la cadena "a" es un lenguaje recursivo. Esto implica que existe una máquina de Turing que puede decidir en tiempo finito si una cadena dada es igual a "a" o no.

j) RE ∪ R = $\mathcal{L}$

Falso

RE ∪ R = RE

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

∴ RE ∪ R ≠ $\mathcal{L}$

k) ($\mathcal{L}$ – RE) = CO-RE

Falso

Existe *L* = {1w / w ∈ L_D } ∪ {0w / w ∉ L_D } que ∉ (RE ∪ CO-RE)

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

∴ ($\mathcal{L}$ – RE) ≠ CO-RE, ($\mathcal{L}$ – RE) = *L* ∪ CO-RE

l) {a} ∈ RE

Verdadero

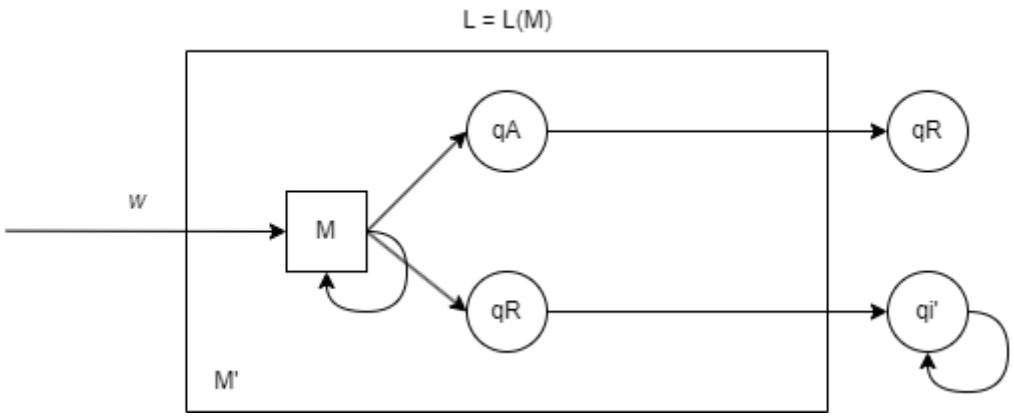
{a} ∈ R ⇒ (def. R y RE)

{a} ∈ RE

3) Si L∈(RE - R)

a) ¿Existirá alguna máquina de Turing que rechace parando en qR si su entrada está en L y rechace loopeando si su entrada no está en L?

Si, sabiendo que tenemos una maquina que reconoce L y que a veces rechaza loopeando, podemos construir una maquina tomando la anterior y todas las δ que terminan en estado qR las redirijo a un loopeo y todas las que terminan en qA las redirijo a qR



b) ¿Existirá alguna máquina de Turing que rechace loopeando si su entrada está en L y rechace parando en qR si su entrada no está en L?

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

Demostración por absurdo

- M (máquina que reconoce a L)
    - Si el string pertenece a L se detiene en qA
- Mb (máquina del inciso b que rechaza loopeando si su entrada está en L y rechace parando en qR si su entrada no está en L)
    - Si el string no pertenece a L se detiene en qR

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 qA o la segunda se va a detener en qR. 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 $L \in R$ → 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 | Existe alguna Máquina de Turing M 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 qA, 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?

Dado que para cada máquina de Turing que reconoce un lenguaje recursivamente enumerable, hay una representación finita de su descripción, y por lo tanto, podemos contar todas las máquinas de Turing posibles que reconocen lenguajes RE, por lo tanto el conjunto de lenguajes RE es un conjunto infinito contable. Se podría mapear la codificación de una Máquina de Turing a un número natural.

c) ¿$\mathcal{L}$ – RE es un conjunto infinito contable?
- La unión de dos conjuntos contables da otro conjunto contable

($\mathcal{L}$ – RE) U RE es contable → Absurdo

($\mathcal{L}$ – RE) U 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 ∈ $\mathcal{L}$, 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 ∈ $\mathcal{L}$ tal que sea infinito no contable.

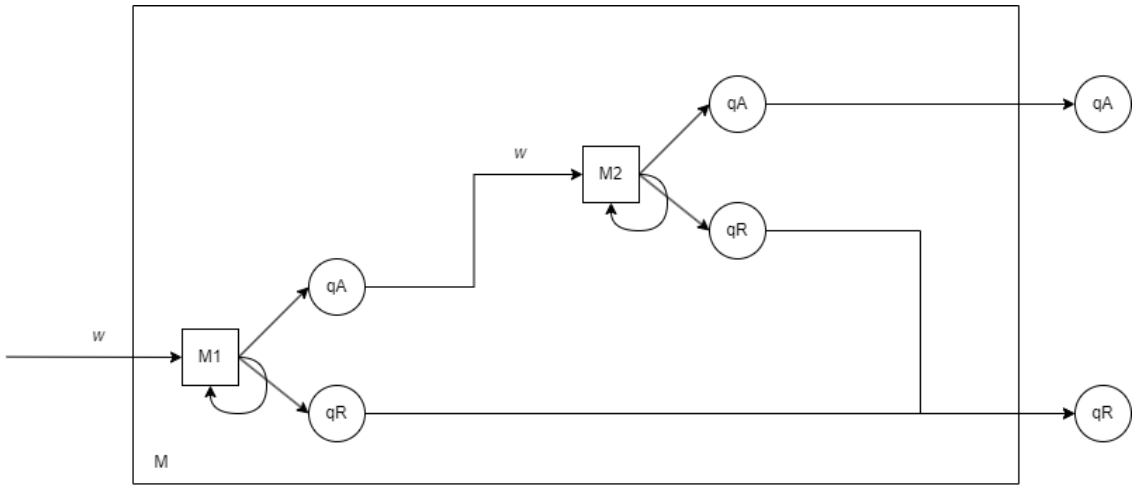
Demostración:
Sabemos que la unión de dos conjuntos contables da otro conjunto contable. También sabemos que $\Sigma*$ es infinito contable.
Supongamos que existe un L infinito no contable, esto llevaría a que $L \cup L^C$ sea incontable (ya que en la unión, si al menos uno es incontable, la unión es incontable).
$L \cup L^C = \Sigma*$ ∴ $\Sigma*$ es incontable → Absurdo, ya que partimos sabiendo que $\Sigma*$ es infinito contable

6) Sea L un lenguaje definido sobre Σ. Demostrar que:
   a) $L^c \notin R \Rightarrow L \notin R$

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

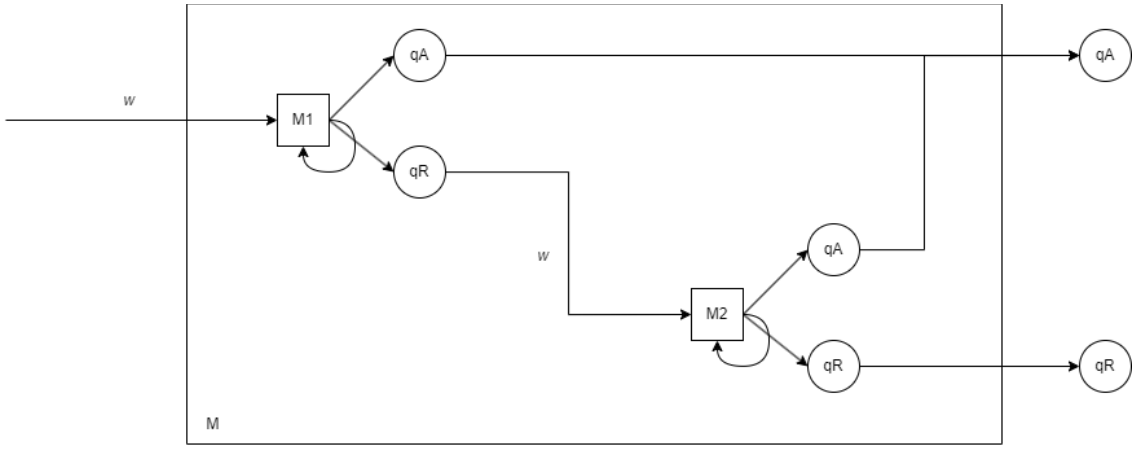
$\Rightarrow L \notin R$ (En el teorema 1 y 2 de la teoría se llega a que R = CO-R)

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



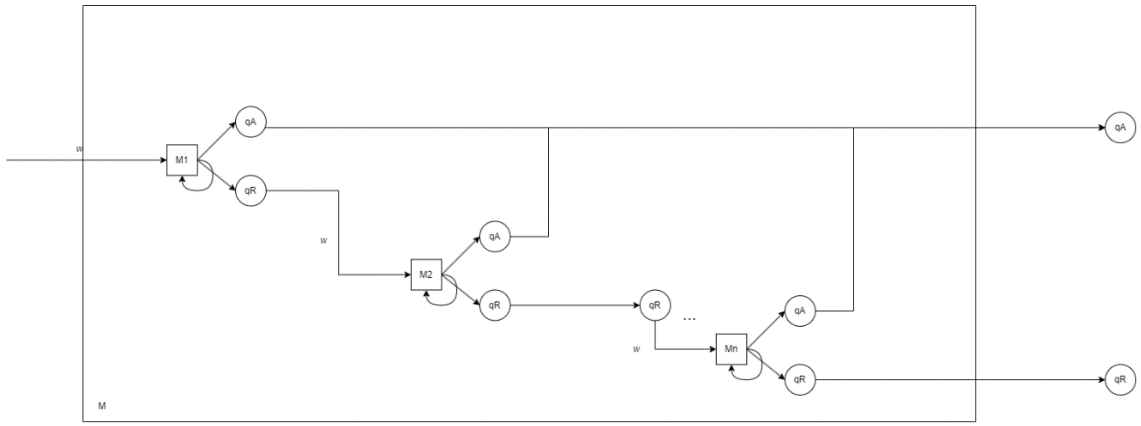
(consultar)

c) (L1 ∈ RE) AND (L2 ∈ RE) ⇒ L1 ∪ L2 ∈ RE



(consultar)

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



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)

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

Vale

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

$L_D \notin RE$

$L_D \cap L_D^c = \emptyset$

$\emptyset \in RE$

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

No vale

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

$L_D \notin RE$

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

$\Sigma* \in RE$

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

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.