Practica 1 – Clases 1 – 2

MT de búsqueda: el algoritmo de la MT M devuelve una solución al problema. w es la entrada y M(w) es la salida. Son el tipo de problema más general que se puede formular son los problemas de búsqueda, donde la respuesta puede ser cualquier valor.

MT de decisión: el algoritmo de la MT M devuelve "si" (acepta) o "no" (rechaza)de acuerdo a si existe una solución al problema. Un problema computacional de decisión es aquel que dependiendo de ciertas condiciones predefinidas deberá responder "si" o "no" de acuerdo a la entrada que se reciba.

MT generadora: el algoritmo de la MT M genera un lenguaje.

Problemas y lenguajes serán sinónimos hasta nuevo aviso.

Un **lenguaje** es un conjuntos de cadenas de símbolos. La MT acepta (o reconoce) un lenguaje. A este se lo conoce como el lenguaje de la maquina y se anota L(M). Básicamente el lenguaje de una maquina es el conjunto de cadenas de símbolos que el lenguaje acepta.

Σ es un alfabeto o conjunto de símbolos. Σ = {w1 , w2 , w3 , w4 , ...}

Σ* es el lenguaje o conjunto de cadenas de símbolos generado a partir de Σ.

Σ* = {λ, w1 , w2 , w3 , ..., w1w1 , w1w2 , w1w3 , ..., w1w1w1 , w1w1w2 , ...}

Σ* es infinito. Sus cadenas son finitas. λ es la cadena vacía.

Todo lenguaje L que consideraremos será un subconjunto de Σ*, siendo Σ un único alfabeto que tomaremos como universal.

$L \subseteq \Sigma^*$.

Operaciones típicas entre lenguajes:

- Intersección: L1 ⋂ L2
- Unión: L1 ⋃ L2
- Diferencia: L1 – L2
- Complemento: LC, que con respecto al conjunto universal Σ*, equivale a (Σ* – L)
- Producto (o Concatenación): L1 . L2

¿Qué es una MT?

Formalmente, una MT M es una tupla (Q, Σ, δ, q0 , qA , qR):

- Q es el conjunto de estados de M.
- Σ es el alfabeto admitido por la cinta de M. Σ incluye al símbolo blanco B. Las cadenas de entrada no admiten blancos.
- q0 es el estado inicial de M.
- qA y qR son los estados finales de aceptación y rechazo de M, respectivamente.
- δ es la función de transición de M (la especificación de su comportamiento):

$$\delta : Q \times \Sigma \rightarrow (Q \cup \{qA , qR \}) \times \Sigma \times \{L, R, S\}$$

Dado un estado corriente de Q, y un símbolo corriente de Σ, la máquina pasa eventualmente a un nuevo estado de Q, o a qA , o a qR , modifica eventualmente el símbolo corriente de Σ, y se mueve un lugar a la derecha (R), a la izquierda (L), o no se mueve (S). Se detiene si en algún momento alcanza el estado qA o el estado qR.

La **tesis de Church-Turing** postula que todo aquello que es computable puede ser calculado por una Máquina de Turing o cualquier otro modelo de cómputo equivalente. Es importante mencionar que no se puede construir física o teóricamente ninguna máquina más potente que una MT.

Dos MT son **equivalentes** si ambas aceptan el mismo lenguaje. **Dos modelos de MT** son **equivalentes** si dada una MT de un modelo existe una MT equivalente del otro.

¿Qué debo poner para construir una MT?

Idea General, Construcción de la MT

Ejemplo:

L = {w | w es una cadena con cero o más símbolos a o b, y es un palíndromo o “capicúa”}

Comentario: w es un palíndromo si es igual a su reverso. Por ejemplo, w = aba es un palíndromo.

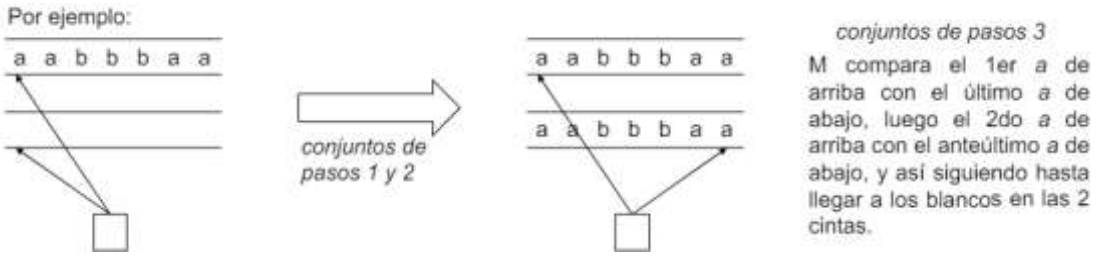
Queremos construir una MT que acepte L.

Idea general: Una MT M con 2 cintas que hace:

1. Copia la entrada, de la cinta 1 a la cinta 2.

2. Vuelve el cabezal de la cinta 1 a la izquierda y deja el cabezal de la cinta 2 a la derecha.

3. Se desplaza a la derecha en la cinta 1 y a la izquierda en la cinta 2, comparando cada vez los símbolos apuntados. Si los pares de símbolos comparados son siempre iguales, acepta. Si no, rechaza.



Construccion:

Definición de la MT M = (Q, Σ, δ, q0 , qA , qR):

Estados Q = {q0 , q1 , q2}

q0: copia de la cinta 1 a la 2;

q1: reposicionamiento en la cinta 1;

q2: comparación de las 2 cintas

Alfabeto Σ = {a, b, B}

	a, a	a, b	a, B	b, a	b, b	b, B	B, a	B, b	B, B
q_0			q_0, a, R, a, R			q_0, b, R, b, R			q_1, B, L, B, L
q_1	q_1, a, L, a, S	q_1, a, L, b, S		q_1, b, L, a, S	q_1, b, L, b, S		q_2, B, R, a, S	q_2, B, R, b, S	q_2, B, S, B, S
q_2	q_2, a, R, a, L	q_R, a, S, b, S		q_R, b, S, a, S	q_2, b, R, b, L				q_A, B, S, B, S

Las celdas en blanco son casos de rechazo de la MT.

Existen **MT con K cintas**, por teorema: Dada una MT M1 con K cintas, existe una MT M2 equivalente, es decir que acepta el mismo lenguaje, con 1 cinta. Comentario: si M1 acepta una cadena en h pasos, M2 lo hace en unos h^2 pasos

La primera cinta solo tiene la entrada y la MT en un solo paso puede modificar el estado corriente, los símbolos corrientes de todas las cintas, y moverse distinto en cada cinta.

Existen **MT no determinísticas (MTN)** en donde para un mismo par (q, w), la máquina puede responder de más de una manera. Una MTN acepta sii al menos una computación acepta. Las MTN sirven para expresar abreviadamente el comportamiento de las MT determinísticas (MTD). Para simular una MTN con una MTD hay que recorrer en el peor caso todas sus computaciones. Asumiendo que toda computación hace a lo sumo h pasos, ejecutar secuencialmente todas las computaciones requiere unos c^h pasos, siendo c el grado del no determinismo (retardo exponencial).

Jerarquía computabilidad

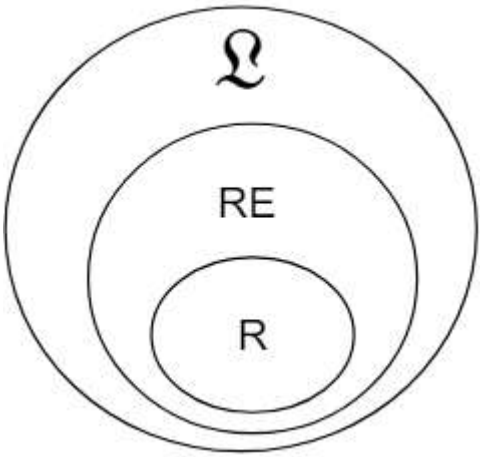
Un lenguaje es **recursivo** (R) si y solo si existe una Maquina de Turing **ML** que lo acepta y siempre se detiene.

Un lenguaje es **recursivamente numerables no recursivo** (RE – R) si y solo si existe una Maquina de Turing **ML** que lo acepta o en su defecto loopea sobre un input (nunca se detiene).

Un lenguaje es **no recursivamente numerable** si y solo si no existe una Maquina de Turing **ML** que lo acepte.

- Un lenguaje L es recursivo ($L \in R$) sii existe una MT ML que lo acepta y para siempre.
  Para toda cadena w del conjunto universal de cadenas Σ^*:
  - Si $w \in L$, entonces ML a partir de w para en su estado qA
  - Si $w \notin L$, entonces ML a partir de w para en su estado Qr
- Un lenguaje L es recursivamente numerable ($L \in RE$) sii existe una MT ML que lo acepta.
  Para toda cadena w del conjunto universal de cadenas Σ^*:

- Si w ∈ L, entonces ML a partir de w para en su estado qA
- Si w ∉ L, entonces ML a partir de w para en su estado qR o no para



Se cumple por definición que $R \subseteq RE \subseteq \mathfrak{Q}$

$R \subseteq RE$ → se puede probar por definición ya que en R están los lenguajes que tienen MT que los aceptan y siempre se detienen y en RE los lenguajes que tienen MT que los aceptan. Por lo tanto R es un subconjunto de RE.

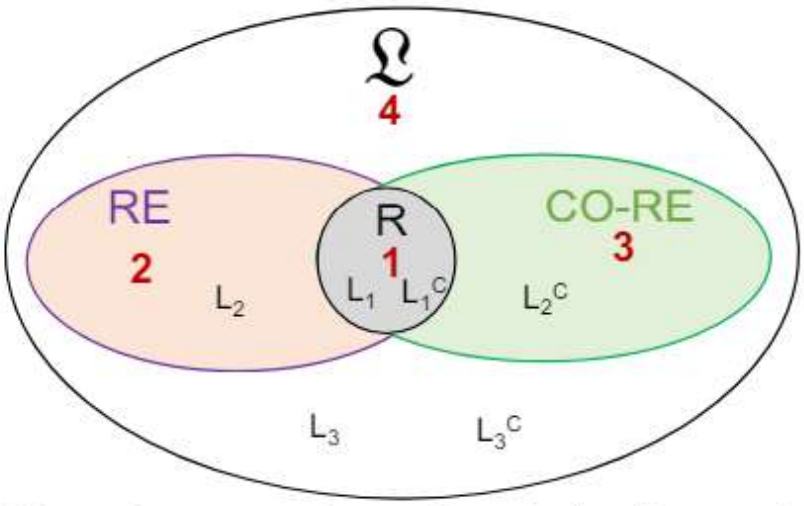
$RE \subseteq \mathfrak{Q}$ → se puede probar por definición ya que $\mathfrak{Q}$ es el conjunto de **todos** los lenguajes definidos sobre el alfabeto Σ y RE es el conjunto de los lenguajes que tienen MT que los aceptan, por lo tanto RE es un subconjunto de $\mathfrak{Q}$.

Propiedades R

- Si $L \in R$, entonces $L^C \in R$, tal que L^C es el complemento de L. (lema 1)
  - $L^C = (\Sigma^* - L)$, o en otras palabras, L^C tiene las cadenas que no tiene L.
- Si $L1 \in R$ y $L2 \in R$, entonces $L1 \cap L2 \in R$, tal que $L1 \cap L2$ es la intersección de L1 y L2 . (lema 2)
  - Definición: $L1 \cap L2$ tiene las cadenas que están en L1 y L2.
- También se cumple que si $L1 \in R$ y $L2 \in R$, entonces $L1 \cup L2 \in R$

Propiedades RE

- Si $L1 \in RE$ y $L2 \in RE$, entonces $L1 \cup L2 \in RE$, tal que $L1 \cup L2$ es la unión de L1 y L2. (lema 3)
  - Definición: $L1 \cup L2$ tiene las cadenas que están en L1 o L2
- También se cumple que si $L1 \in RE$ y $L2 \in RE$, entonces $L1 \cap L2 \in RE$
- Por otro lado, $L \in RE$ NO IMPLICA que $L^C \in RE$ (diferencia con R)



CO-RE tiene los complementos de los lenguajes de RE

Región 1 (los lenguajes más "fáciles"):R es la clase de los lenguajes recursivos. Si L1 está en R, entonces también $L1^C$ está en R.

Región 2. Clase de los lenguajes (RE – R). Si L2 está en RE, entonces $L2^C$ está en CO-RE.

Región 3. Clase de los lenguajes (CO-RE – R). Si L2 está en CO-RE, entonces $L2^C$ está en RE.

Región 4 (los lenguajes más "difíciles"). Clase de los lenguajes $\mathfrak{Q}$ – (RE ∪ CO-RE). Si L3 está en la clase, también está $L3^C$.

R = RE ⋂ CO-RE. (lema 4):

a) $R \subseteq RE \cap CO\text{-}RE$: Si $L \in R$, entonces:
   - $L \in RE$ por definición.
   - $L^C \in R$ por el Lema 1, y entonces $L^C \in RE$ por definición.
   - Por lo tanto, $L \in RE$ y $L^C \in RE$, es decir $L \in RE$ y $L \in CO\text{-}RE$, es decir $L \in RE \cap CO\text{-}RE$.

b) $RE \cap CO\text{-}RE \subseteq R$: Si $L \in RE \cap CO\text{-}RE$, entonces:
   - $L \in RE$ y $L \in CO\text{-}RE$.
   - $L \in RE$ y $LC \in RE$.
   - Existe una MT M que acepta L y existe una MT M^C que acepta L^C.
   - Demostración: existen una MT M que acepta L y una MT M^C que acepta L^C. Construimos una MT M´ que ejecuta "en paralelo" M y M^C:
     - Se cumple que M´ acepta L y para siempre:
       - Para todo w, $w \in L$ o $w \in L^C$. Así, M acepta w o M^C acepta w, respectivamente. Por lo tanto, M´ para siempre.
       - M´ acepta L porque acepta una cadena w sii M acepta w.

Los únicos lenguajes de la clase CO-RE que tienen MT que los aceptan son los lenguajes que pertenecen a R. Todo lenguaje que este en CO-RE – RE no posee una Máquina de Turing que lo acepta, si no que en realidad existe una MT que acepta a su complemento (y no necesariamente para).

Σ* es recursivo ya que existe una Maquina de Turing que lo acepta y siempre se detiene, esta es aquella que luego de leer el primer símbolo de cualquier cadena acepta. Esta MT siempre se detiene.

∅ es recursivo ya que existe una Maquina de Turing que lo acepta y siempre se detiene, esta es aquella que luego de leer el primer símbolo de cualquier cadena rechaza, haciendo que el lenguaje de la maquina sea vacío. Esta MT siempre se detiene.

Un lenguaje finito es recursivo porque se puede construir una Máquina de Turing que lo acepte si la entrada coincide con alguno de los elementos del lenguaje o la rechace en el caso contrario.

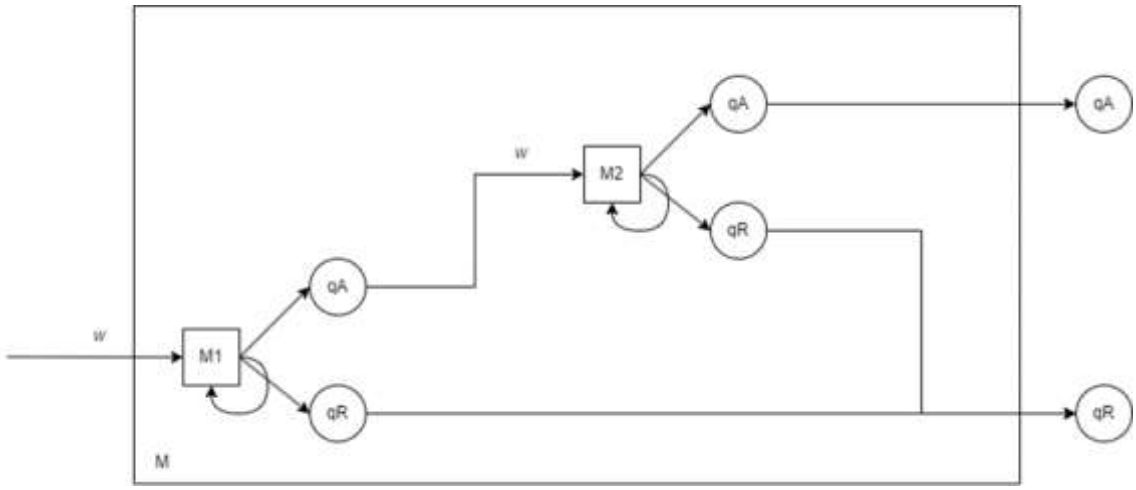
L1 ∈ CO-RE y L2 ∈ CO-RE, entonces (L1 ⋂ L2) ∈ CO-RE:

- Si L1 ∈ CO-RE existe un $L1^C$ que ∈ RE. Si L2 ∈ CO-RE existe un $L2^C$ que ∈ RE.
- $L1^C$ ⋃ $L2^C$ ∈ RE por lema 2
- Aplicando la Ley De Morgan:
    - $L1^C$ ⋃ $L2^C$ ∈ RE = $(L1 ⋂ L2)^C$ ∈ RE
    - Como $(L1 ⋂ L2)^C$ ∈ RE entonces (L1 ⋂ L2) ∈ CO-RE (por definición)

Prueba de la intersección para saber como se debe demostrar en el parcial

1. Idea general. 2. Construcción de la MT M. 3. Verificación de correctitud.

1. <u>Idea general</u>
   Dadas dos MT M1 y M2 que respectivamente aceptan L1 y L2, la idea es construir una MT M que acepte L1 ⋂ L2 (L = L1 ⋂ L2).



2. <u>Construcción</u>

- M tiene 2 cintas.
- Dada la entrada w en la cinta 1, M hace:
  1. Copia w en la cinta 2.
  2. Ejecuta M1 sobre w en la cinta 2. Si M1 para en qR , entonces M para en qR. Si M1 loopea entonces M loopeara (rechaza). Si M1 para en qA, entonces:
     - Borra el contenido de la cinta 2 y copia de nuevo w en la cinta 2.
     - Ejecuta M2 sobre w en la cinta 2. Si M2 para en qR , entonces M para en qR. Si M2 loopea entonces M loopeara (rechaza). Si M2 para en qA, entonces M para en qA

3. <u>Prueba de correctitud de la construcción.</u>

   L = L1 ⋂ L2

   M reconoce solo las entradas que son reconocidas por M1 y M2, es decir, reconoce las entradas que son reconocidas por ambas maquinas (la intersección). Si se trata de una entrada que es rechazada por una maquina pero no por la otra, M no la va a aceptar. Si se trata de una entrada rechazada por ambas maquinas, M la va a rechazar. Por lo tanto L = L1 ⋂ L2

   L ∈ RE

   Como L = L(M), existe una máquina que lo acepta. Si M1 o M2 se quedan loopeando M se va a quedar loopeando.

HP = {(<M>, w) | M para a partir de w} pertenece al conjunto RE – R. <M> es el código de una MT.

En algunas MT se puede ejecutar en paralelo que es simplemente ejecutar alternativamente un paso de cada una de las MT y luego ir incrementando los pasos.