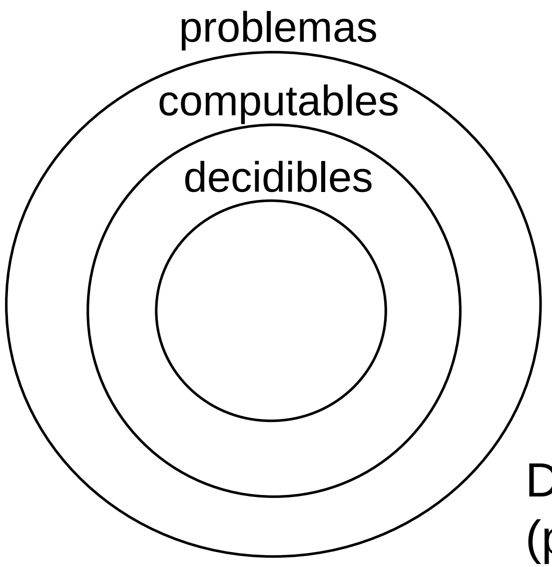
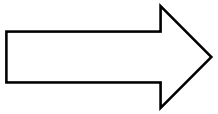
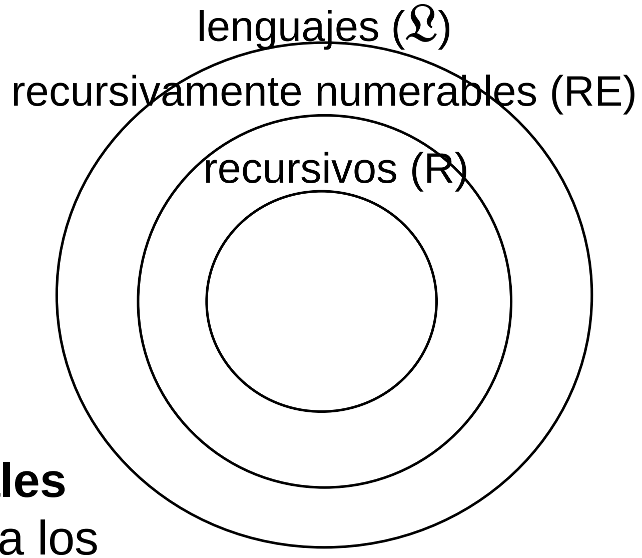
Clase 2. Jerarquía de la Computabilidad.







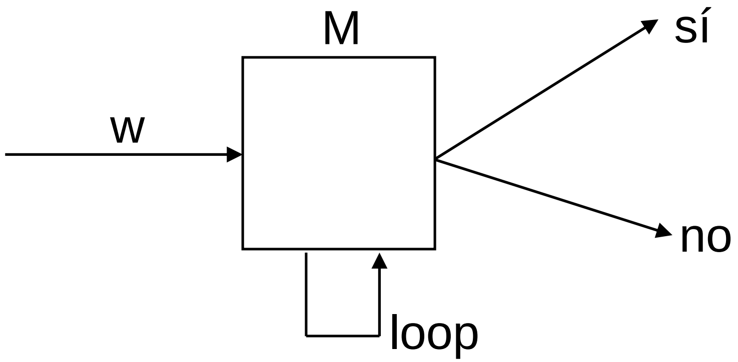
De los **problemas generales** (problemas de búsqueda) a los **problemas de decisión** (representados por lenguajes)

𝔏 : conjunto de los lenguajes

RE : conjunto de los lenguajes recursivamente numerables (o enumerables)

R : conjunto de los lenguajes recursivos

Iremos probando formalmente las distintas fronteras del mapa de la computabilidad



Nuestro artefacto para el estudio de la computabilidad: MT con K cintas.

L(M) = {w | M acepta w}. La MT M acepta, o reconoce, el lenguaje L(M).

- Se asume un alfabeto universal de símbolos: $\Sigma = \{a_1, a_2, a_3, \ldots\}$
- Σ^* es el conjunto de todas las cadenas finitas formadas con símbolos de Σ
- $\mathfrak{L}$ es el conjunto de todos los lenguajes formados con cadenas de Σ^*:

  $\mathfrak{L} = P(\Sigma^*)$, es decir que $\mathfrak{L}$ es el conjunto de partes de Σ^*

alfabeto universal Σ

a_1
a_2
a_3
…….

conjunto Σ^* de cadenas

$\lambda \;\; a_1 \;\; a_2 \;\; a_3 \;\ldots$
$a_1a_1 \;\; a_1a_2 \;\; a_1a_3 \;\ldots$
$a_1a_1a_1 \;\; a_1a_1a_2 \ldots$
………….

$\mathfrak{L}$ tiene todos los lenguajes de $P(\Sigma^*)$

$L_1 \;\; L_2 \;\; L_3$
$L_4 \;\; L_5 \;\; L_6 \;\; L_7$
$L_8 \;\; L_9 \;\; L_{10} \;\; L_{11}$
…………

- Todo L_i de $\mathfrak{L}$ es un subconjunto de Σ^*, p.ej. $\{a_1, a_5a_{10}a_8, a_2a_2a_2a_{20}a_2\}$
- Todo lenguaje L_i representa un **problema de decisión**
- Ejemplo de lenguaje L_i : {G | G es un grafo que tiene un camino del vértice v_1 al vértice v_n}

Un lenguaje L es **recursivamente numerable** ($L \in RE$) si y sólo si existe una MT M_L que **lo acepta**, es decir $L(M_L) = L$. Por lo tanto, para toda cadena w de Σ^*:

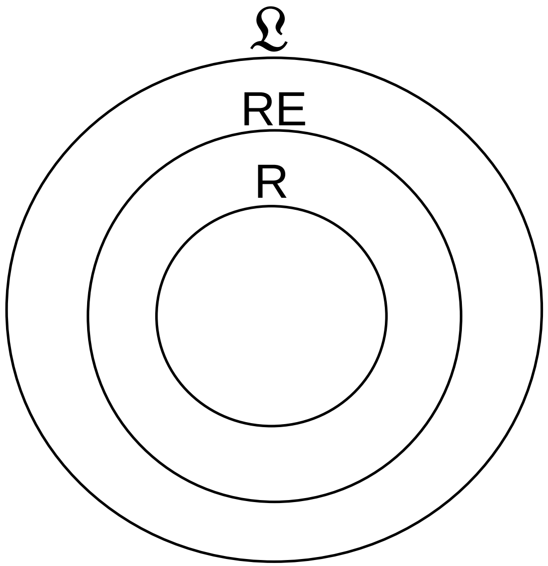
Si $w \in L$, entonces M_L a partir de w para en su estado q_A

Si $w \notin L$, entonces M_L a partir de w para en su estado q_R o no para

Un lenguaje L es **recursivo** ($L \in R$) si y sólo si existe una MT M_L que **lo acepta y para siempre** (también se puede decir directamente que **lo decide**). Por lo tanto, para toda cadena w de Σ^*:

Si $w \in L$, entonces M_L a partir de w para en su estado q_A

Si $w \notin L$, entonces M_L a partir de w para en su estado q_R



Se cumple por definición que $R \subseteq RE \subseteq \mathfrak{L}$ (ejercicio)

Probaremos entre esta clase y la que viene que $R \subset RE \subset \mathfrak{L}$

Algunas propiedades de la clase R

R es **cerrada** con respecto a las operaciones de complemento, intersección, unión y concatenación. Es decir:

- Si $L \in R$, entonces $L^C \in R$, siendo L^C el complemento de L con respecto a Σ^*, con:

  $$L^C = \{w \mid w \in \Sigma^* \wedge w \notin L\}, \text{ o en otras palabras: } L^C = \Sigma^* - L$$

- Si $L_1 \in R$ y $L_2 \in R$, entonces $L_1 \cap L_2 \in R$

- Si $L_1 \in R$ y $L_2 \in R$, entonces $L_1 \cup L_2 \in R$

- Si $L_1 \in R$ y $L_2 \in R$, entonces $L_1 . L_2 \in R$, siendo $L_1 . L_2$ el lenguaje concatenación o producto de L_1 con L_2, con:

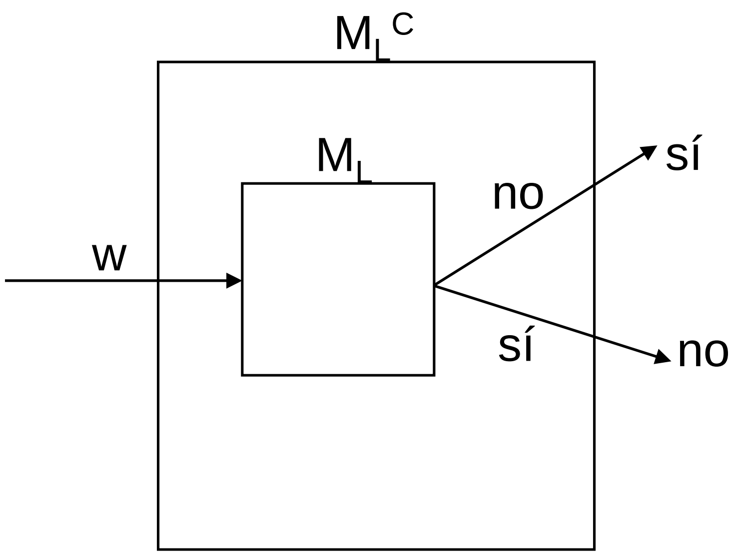
  $$L_1 . L_2 = \{w \mid w = w_1 w_2, \text{ con } w_1 \in L_1 \text{ y } w_2 \in L_2\}$$

Lema 1. Si $L \in R$, entonces $L^C \in R$

Prueba.

<u>1. Idea general</u>.

Dada una MT M_L que acepta L y para siempre (hipótesis), la idea es construir una MT $M_L{}^C$ que acepte L^C y pare siempre.



En la nueva MT se permutan los estados finales de la MT original

2. Construcción.

Si $M_L = (Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R)$,
entonces $M_L^C = (Q, \Sigma, \Gamma, \delta', q_0, q_A, q_R)$,
tal que δ y δ´ son idénticas salvo que con los estados q_A y q_R **permutados**

Formalmente:

Para todos los estados q y q´, símbolos a y a´, y movimientos d de {L, R, S}:

- Si $\delta(q, a) = (q', a', d)$, siendo $q' \neq q_A$ y q_R, entonces $\delta'(q, a) = (q', a', d)$
- Si $\delta(q, a) = (q_A, a', d)$, entonces $\delta'(q, a) = (q_R, a', d)$
- Si $\delta(q, a) = (q_R, a', d)$, entonces $\delta'(q, a) = (q_A, a', d)$

3. Prueba de correctitud de la construcción.

- **M_L^C para siempre:**

  M_L para siempre y M_L^C sólo difiere de M_L en que para en el estado opuesto

- **$L(M_L^C) = L^C$** (vamos a probarlo por doble inclusión de conjuntos):

  $w \in L(M_L^C) \leftrightarrow$ por definición:

  con input w, M_L^C para en $q_A \leftrightarrow$ por construcción:

  con input w, M_L para en $q_R \leftrightarrow$ por definición:

  $w \notin L(M_L) \leftrightarrow$ por definición:

  $w \notin L \leftrightarrow$ por definición:

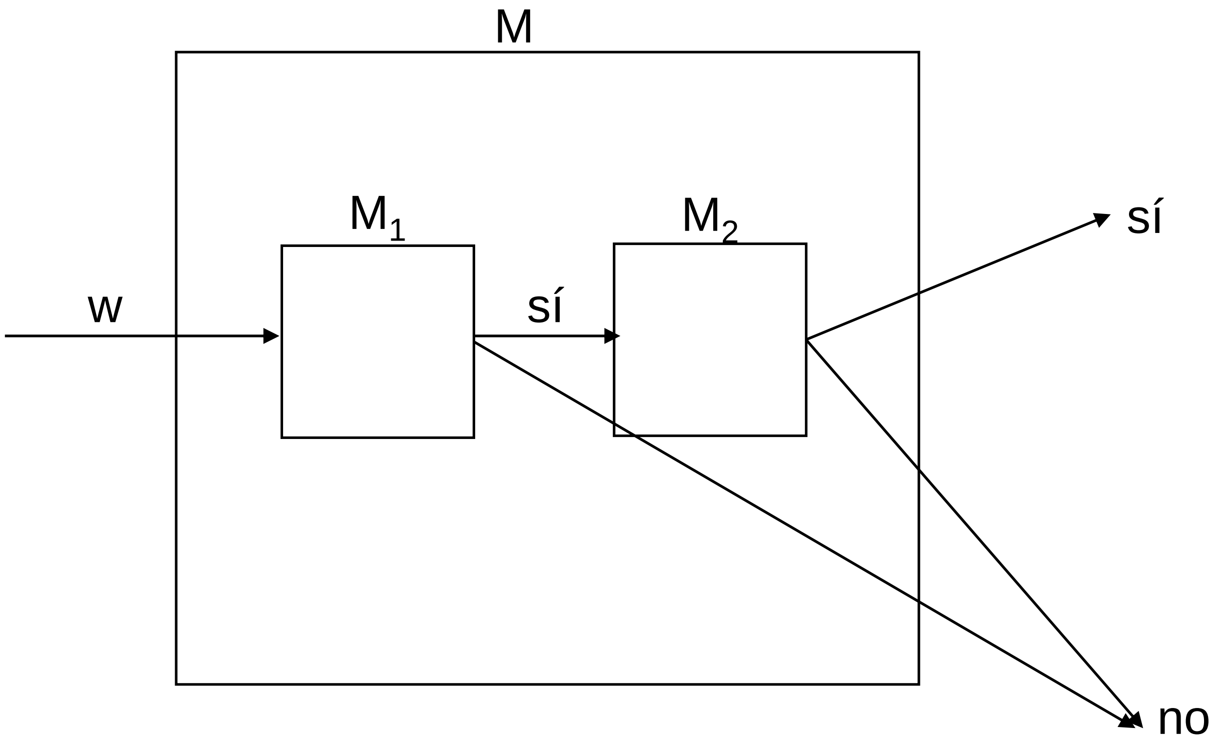
  $w \in L^C$

Lema 2. Si $L_1 \in R$ y $L_2 \in R$, entonces $L_1 \cap L_2 \in R$

Prueba.

1. Idea general.

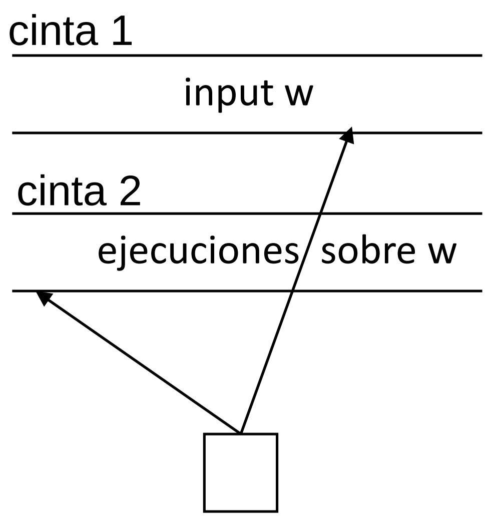
Dadas dos MT M_1 y M_2 que aceptan L_1 y L_2 y paran siempre (hipótesis), la idea es construir a partir de ellas una MT M que acepte $L_1 \cap L_2$ y pare siempre.



La nueva MT ejecuta secuencialmente (eventualmente) las dos MT originales

2. Construcción.

M tiene 2 cintas. Con el input w en la cinta 1, M hace:



1. Copia w en la cinta 2.
2. Ejecuta M_1 sobre w en la cinta 2. Si M_1 para en q_R, entonces M para en q_R.
3. Borra el contenido de la cinta 2 y copia w en la cinta 2.
4. Ejecuta M_2 sobre w en la cinta 2. Si M_2 para en q_A (q_R), entonces M para en q_A (q_R).

- Los pasos 2 y 4 pueden entenderse como invocaciones a subrutinas (que no son más que las funciones de transición δ_1 de M_1 y δ_2 de M_2 incluidas adecuadamente en la función de transición δ de M). Queda como ejercicio indicar cómo se implementaría copiar w en la cinta 2 y borrar el contenido de la cinta 2.
- También quedan como ejercicios:
  - ✓ Probar la correctitud de la construcción: (a) M para siempre. (b) $L(M) = L_1 \cap L_2$.
  - ✓ Probar las otras propiedades de clausura de R mencionadas anteriormente.

Algunas propiedades de la clase RE

RE es **cerrada** con respecto a las operaciones de intersección, unión y concatenación (no con respecto al complemento, como veremos luego). Es decir:

- Si $L_1 \in RE$ y $L_2 \in RE$, entonces $L_1 \cap L_2 \in RE$
- Si $L_1 \in RE$ y $L_2 \in RE$, entonces $L_1 \cup L_2 \in RE$
- Si $L_1 \in RE$ y $L_2 \in RE$, entonces $L_1 . L_2 \in RE$

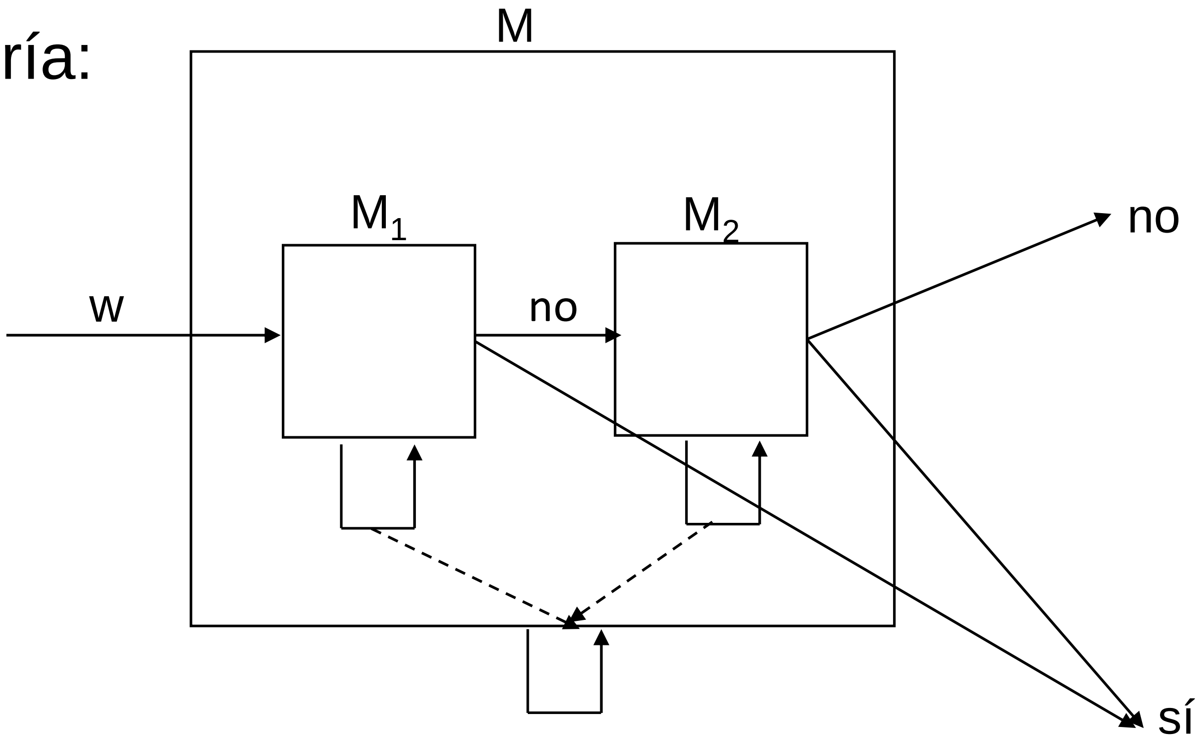
    - Si $L \in RE$, no necesariamente $L^C \in RE$

Lema 3. Si $L_1 \in RE$ y $L_2 \in RE$, entonces $L_1 \cup L_2 \in RE$

Prueba.

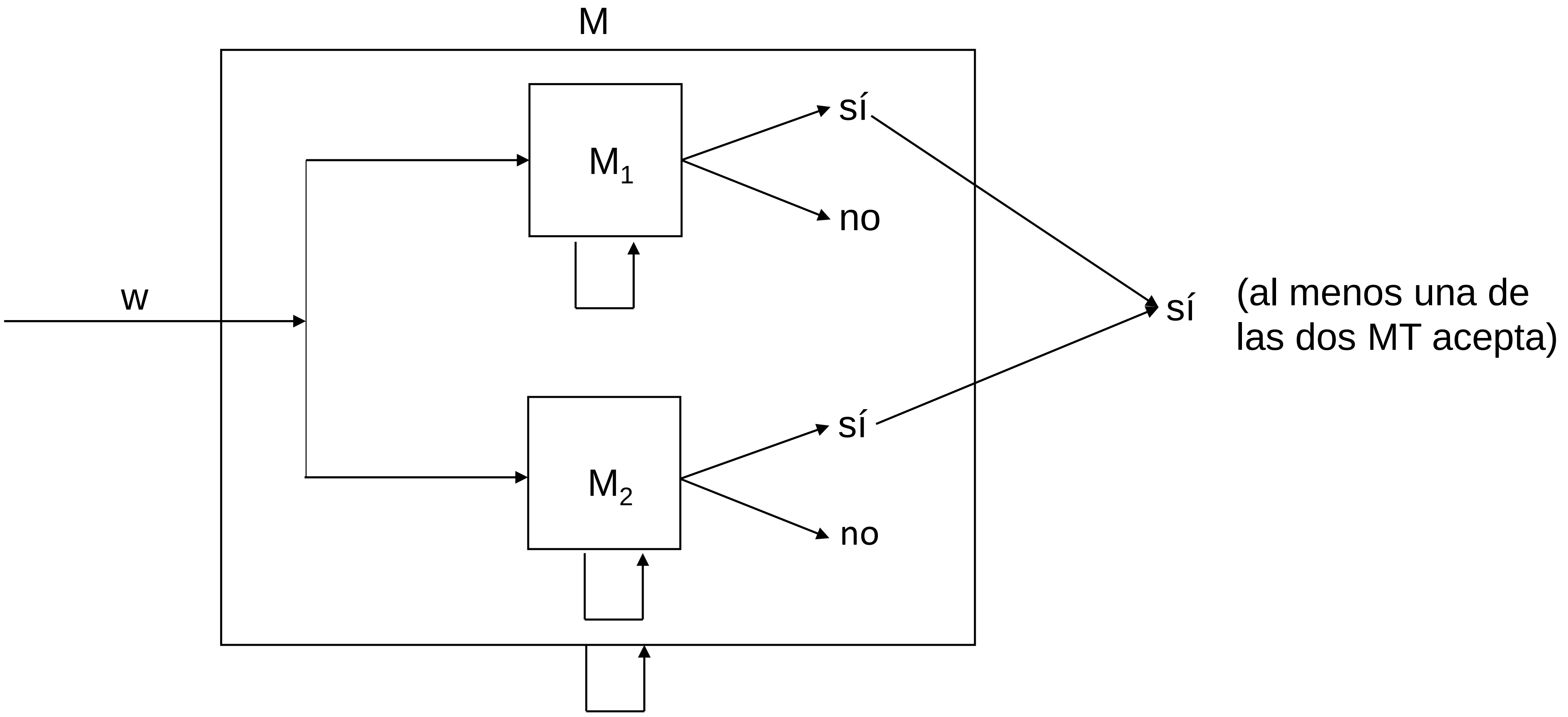
1. Idea general.

Dadas por hipótesis dos MT M_1 y M_2 que aceptan L_1 y L_2 (no necesariamente paran siempre), una idea sería:



¡Problema! Si M_2 acepta w y M_1 no para sobre w, entonces M no acepta w (**error**).

La idea es ejecutar **“en paralelo”** M_1 y M_2. Si en algún paso alguna de ellas acepta, entonces la MT M construida acepta.



2. Construcción.

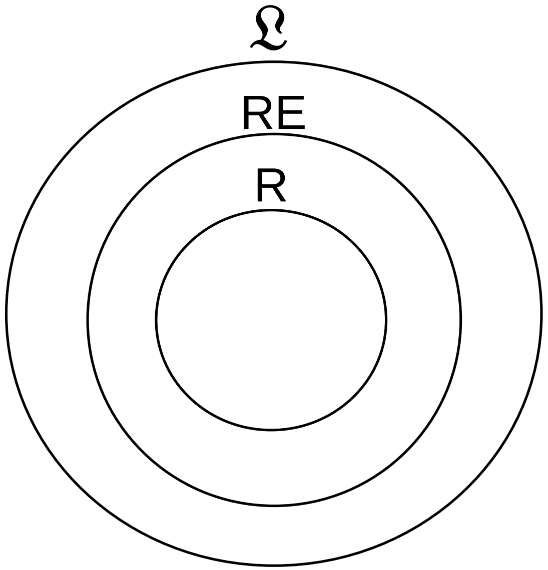
M tiene 4 cintas. En la cinta 1 tiene el input w. En las cintas 2 y 3 ejecuta M_1 y M_2. En la cinta 4 tiene un contador i de pasos:

1. Copia w en las cintas 2 y 3, y en la cinta 4 hace i := 1.
2. Ejecuta a lo sumo i pasos de M_1 sobre w en la cinta 2, y a lo sumo i pasos de M_2 sobre w en la cinta 3. Si M_1 o M_2 aceptan, M acepta.
3. Borra el contenido de las cintas 2 y 3, copia w en ellas desde la cinta 1, suma 1 a i en la cinta 4, y vuelve al paso 2.

cinta 1	w
cinta 2	ejecución de M_1
cinta 3	ejecución de M_2
cinta 4	contador i de pasos

- M acepta o no para. M se puede optimizar rechazando en (2) cuando las 2 MT rechazan.
- En c/iteración M ejecuta las dos MT desde el paso 1. M se puede optimizar ejecutando en c/iteración sólo el paso siguiente (debería memorizar los estados y posiciones corrientes).
- Queda como ejercicio indicar cómo sumar 1 a i y cómo ejecutar i pasos de M_1 y M_2.
- También quedan como ejercicios probar la correctitud de la construcción (es decir la igualdad $L(M) = L_1 \cup L_2$) y las otras propiedades de clausura de RE mencionadas antes.

Jerarquía de la computabilidad



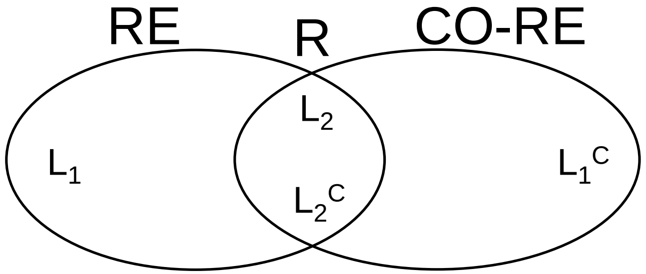
Ya se indicó que R $\subseteq$ RE $\subseteq \mathfrak{L}$.

Probaremos entre esta clase y la que viene que R $\subset$ RE $\subset \mathfrak{L}$.

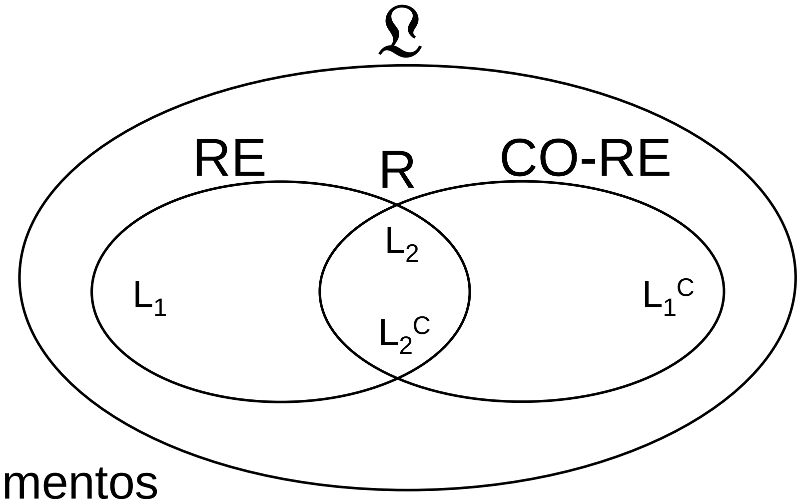
Como ayuda, definimos primero: CO-RE = $\{L \mid L \in \mathfrak{L} \wedge L^C \in \text{RE}\}$.

Es decir, CO-RE es el conjunto de los lenguajes tales que sus complementos están en RE.

Vamos a probar primero que se cumple **R = RE ∩ CO-RE**. Es decir que **sólo en la clase R vale que si contiene el lenguaje L también contiene su complemento L^C**. Llegaremos a lo siguiente:



y en definitiva a:



Así se dispondrán los lenguajes y sus complementos

Lema 4. R = RE ∩ CO-RE

Prueba.

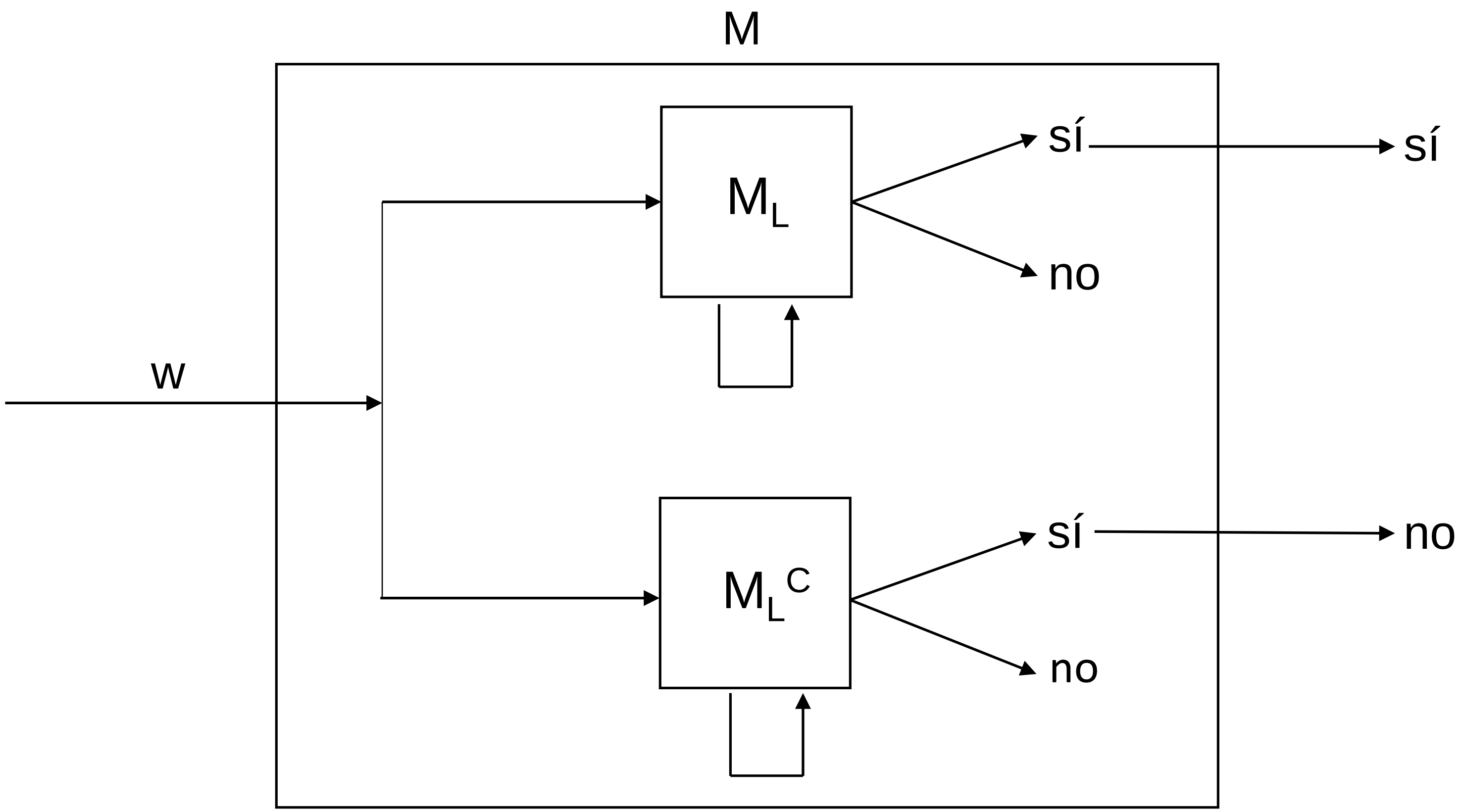
- La inclusión **R ⊆ RE ∩ CO-RE** se prueba fácilmente:

R ⊆ RE: se cumple por definición (ya visto antes)
R ⊆ CO-RE: L ∈ R → L^C ∈ R (por Lema 1) → L^C ∈ RE → L ∈ CO-RE

- Probaremos ahora la otra inclusión: **RE ∩ CO-RE ⊆ R**:

Hay que probar que si L ∈ RE ∩ CO-RE, entonces L ∈ R
En otras palabras: si L ∈ RE y L ∈ CO-RE, entonces L ∈ R
Lo que es lo mismo que: si L ∈ RE y L^C ∈ RE, entonces L ∈ R
Vamos a construir una MT M que decida L (que lo acepte y pare siempre), a partir de la hipótesis de que existen MT M_L y M_L^C que aceptan, resp., L y L^C:

Idea general



Con:

$L(M_L) = L$

$L(M_L^C) = L^C$

Y así:

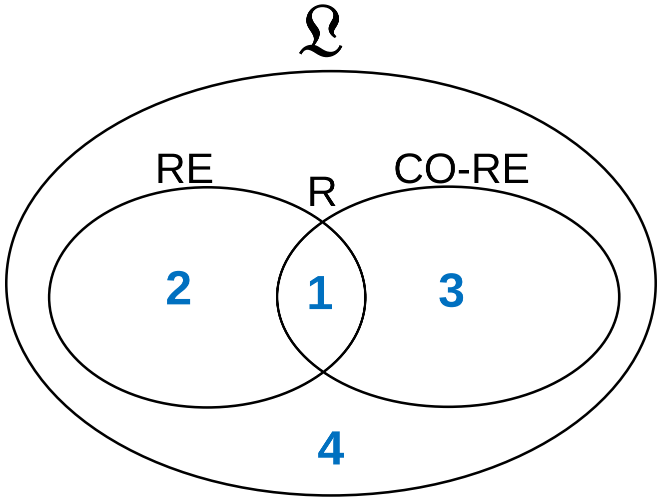
$L(M) = L$

Que existan las MT M_L y M_L^C que aceptan L y L^C, respectivamente, es **suficiente información** para obtener una MT M que decide L: **dado un input w, $w \in L$ o bien $w \in L^C$, y por lo tanto M_L acepta w (en cuyo caso M acepta) o bien M_L^C acepta w (en cuyo caso M rechaza).** De este modo, ejecutando ambas MT "en paralelo" se logra lo buscado.

Queda como ejercicio la construcción de M, y la prueba de que: (a) M para siempre, (b) L(M) = L

- Probaremos que:

  R $\subset$ RE (hay lenguajes recursivamente numerables que no son recursivos)
  RE $\subset \mathfrak{L}$ (hay lenguajes que no son recursivamente numerables)
  RE $\neq$ CO-RE (RE no es cerrada con respecto al complemento)
  RE $\cup$ CO-RE $\neq \mathfrak{L}$ (hay lenguajes fuera de RE $\cup$ CO-RE)



- Así las cosas, dado un lenguaje L veremos que existen 3 posibilidades:

1. **L y L^C están en R**. La clase R es la clase de lenguajes o problemas de menor "dificultad" (**región 1**) desde el punto de vista de la computabilidad.

2. **L está en RE y L^C está en CO-RE**. Los conjuntos RE – R y CO-RE – R le siguen en "dificultad" a R en la jerarquía de la computabilidad (**regiones 2 y 3**).

3. **Tanto L como L^C están fuera de RE $\cup$ CO-RE** (¡no existen MT ni para uno ni para el otro!). La región $\mathfrak{L}$ – (RE $\cup$ CO-RE) es la de mayor "dificultad" (**región 4**), es la que tiene a los lenguajes más "difíciles", y además la más amplia de la jerarquía.

Algunos ejemplos clásicos de problemas de decisión fuera de R

- Dada una ecuación diofántica (polinomio con coeficientes enteros), por ejemplo $x^3 + y^3 = z^3$, **¿tiene una solución con números enteros?** Está en RE – R.

- Teselación del plano: dado un conjunto finito de formas poligonales, **¿se puede cubrir el plano con ellas?** Está en RE – R.

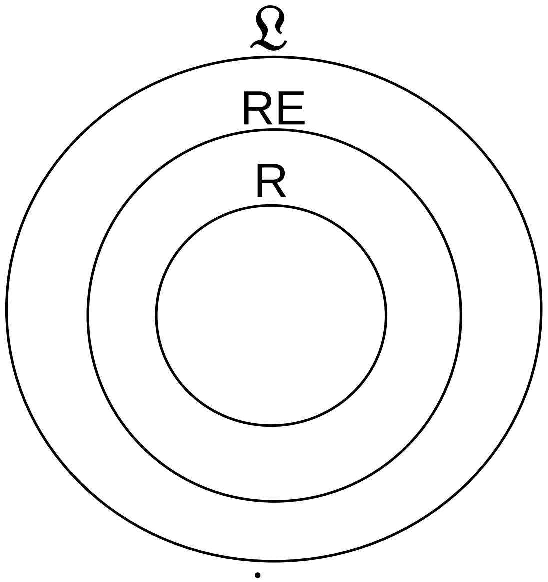
- Problema de Correspondencia de Post (PCP): dado un conjunto de pares de cadenas de 1 y 0, por ejemplo {(1101,11),(00110,0100),(1110,1100),(0101,1010)}, **¿hay una forma de disponer los pares (se pueden repetir) tal que concatenando las partes de la izquierda y las partes de la derecha se obtenga una misma cadena?** Está en RE – R.

- En la lógica de primer orden, dada una fórmula como por ejemplo $(\forall x)\ (\exists y)\ P(x,y,8)$, **¿acaso la fórmula es válida?** Está en RE – R.

- En la teoría o axiomática de la aritmética, dada una fórmula como por ejemplo $(\forall x)\ (\forall y)\ (\forall z)\ f(x,y,z) = g(x,y,z)$, **¿acaso la fórmula es verdadera?** ¡Está en $\mathfrak{L}$ - (RE ∪ CO-RE)!

- Halting Problem (Problema de la Parada de las Máquinas de Turing): dada una MT M y un input w, **¿acaso M para a partir de w?** Está en RE - R.

La prueba central de la clase que viene, para corroborar la jerarquía:



será la de R $\subset$ RE, **encontrando un lenguaje L de RE – R**.

Así también valdrá RE $\subset \mathfrak{L}$, porque deberá ser $L^C \notin$ RE (si $L^C \in$ RE, como L $\in$ RE, entonces L $\in$ R), **por lo que habremos encontrado un lenguaje L^C de $\mathfrak{L}$ – RE**.

Deberemos recurrir a una técnica distinta. Para probar que L $\in$ R o L $\in$ RE vimos que basta con **construir una MT**. En cambio, para probar que L $\notin$ R o L$\notin$ RE tendremos que utilizar otra técnica (usaremos **diagonalización** o **reducción**).

Ejercicio (Clase Práctica). Probar que la clase R es cerrada con respecto a la operación de concatenación, es decir que si $L_1 \in R$ y $L_2 \in R$, entonces también $L_1 . L_2 \in R$.

Idea general.

El lenguaje $L_1 . L_2$ contiene todas las cadenas $w = w_1w_2$, tales que $w_1 \in L_1$ y $w_2 \in L_2$

Sea M_1 una MT que decide el lenguaje L_1 y M_2 una MT que decide el lenguaje L_2. Hay que construir una MT M que decida el lenguaje $L_1 . L_2$. Dado un input w con n símbolos, M hace:

1. M ejecuta M_1 a partir de los primeros 0 símbolos de w, y M_2 a partir de los últimos n símbolos de w
2. Si en ambos casos se acepta, entonces M acepta
3. Si no, M hace lo mismo que en (1) pero ahora con el 1er símbolo de w y los últimos (n – 1) de w
4. Si en ambos casos se acepta, entonces M acepta
5. Si no, mientras M no acepte, M repite el paso (1) con:
   2 y (n – 2) símbolos de w
   3 y (n – 3) símbolos de w
   y así siguiendo hasta llegar a n y 0 símbolos de w (si M nunca acepta, entonces rechaza)

Queda como ejercicio la construcción de M y la verificación de su correctitud

Ejercicio (Clase Práctica). Probar que también la clase RE es cerrada con respecto a la operación de concatenación, es decir que si $L_1 \in$ RE y $L_2 \in$ RE, entonces también $L_1 \,.\, L_2 \in$ RE.

Idea general.

Tal como se hizo con los lenguajes recursivos, se tiene que construir una MT M que reconozca $L_1 \,.\, L_2$ ejecutando sobre un input w (de n símbolos) determinadas MT M_1 y M_2 (MT que reconocen L_1 y L_2, respectivamente, las cuales ahora pueden loopear en casos negativos),
primero a partir de 0 y n símbolos de w,
después a partir de 1 y n – 1 símbolos de w,
y así siguiendo hasta llegar a n y 0 símbolos de w, aceptando eventualmente.

La diferencia con el caso de los lenguajes recursivos está en que ahora, teniendo en cuenta los posibles loops de M_1 y M_2, M debe ejecutarlas "en paralelo":

M primero debe hacer ejecuciones de 1 paso de M_1 y M_2 con todas las posibles particiones de w,
luego ejecuciones de 2 pasos,
luego ejecuciones de 3 pasos,
y así siguiendo hasta eventualmente aceptar

Queda como ejercicio la construcción de M y la verificación de su correctitud

Ejercicio (Clase Práctica). Probar que la clase RE es cerrada con respecto a la operación de unión, permitiendo como solución una MT no determinística (MTN).

Idea general y construcción.

Dados dos lenguajes L_1 y L_2 de RE, aceptados por MT M_1 y M_2,
con $M_1 = (Q_1, \Sigma_1, \Gamma_1, \delta_1, q_{10}, q_A, q_R)$ y $M_2 = (Q_2, \Sigma_2, \Gamma_2, \delta_2, q_{20}, q_A, q_R)$,
vamos a construir una MTN M que acepta $L_1 \cup L_2$:

Sea q_0 un estado que no está en Q_1 ni en Q_2. La MTN M es:

$$M = (Q_1 \cup Q_2 \cup \{q_0\}, \Sigma = \Sigma_1 \cup \Sigma_2, \Gamma = \Gamma_1 \cup \Gamma_2, \Delta, q_0, q_A, q_R), \text{ tal que:}$$

$$\Delta = \delta_1 \cup \delta_2 \cup \{(q_0, a, q_{10}, a, S), (q_0, a, q_{20}, a, S)\}, \text{ considerando todos los símbolos a de } \Sigma$$

Es decir, al comienzo la MTN M pasa no determinísticamente a la configuración inicial de M_1 o M_2, y después se comporta como ellas.

Queda como ejercicio la verificación de la correctitud de la construcción de la MTN M