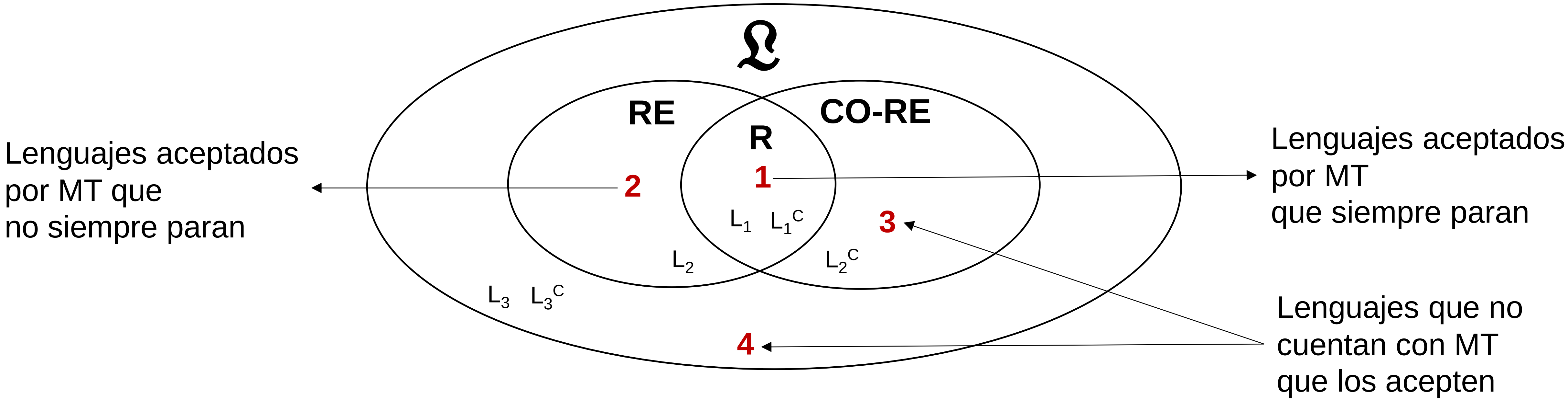
Clase teórica 3

Indecibilidad

Repaso

- **Hemos presentado la jerarquía de la computabilidad:**



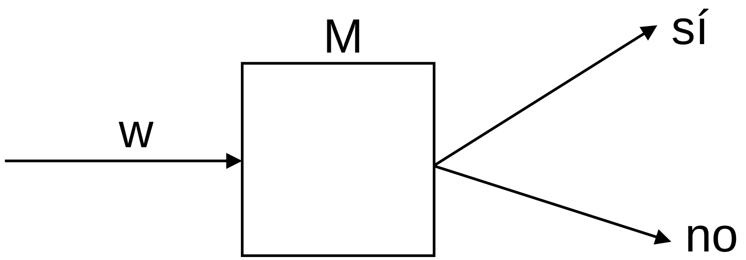
- Falta probar **R** $\subset$ **RE** $\subset \mathfrak{L}$:

  Hay lenguajes en RE – R (región 2): **hay problemas computables no decidibles**.

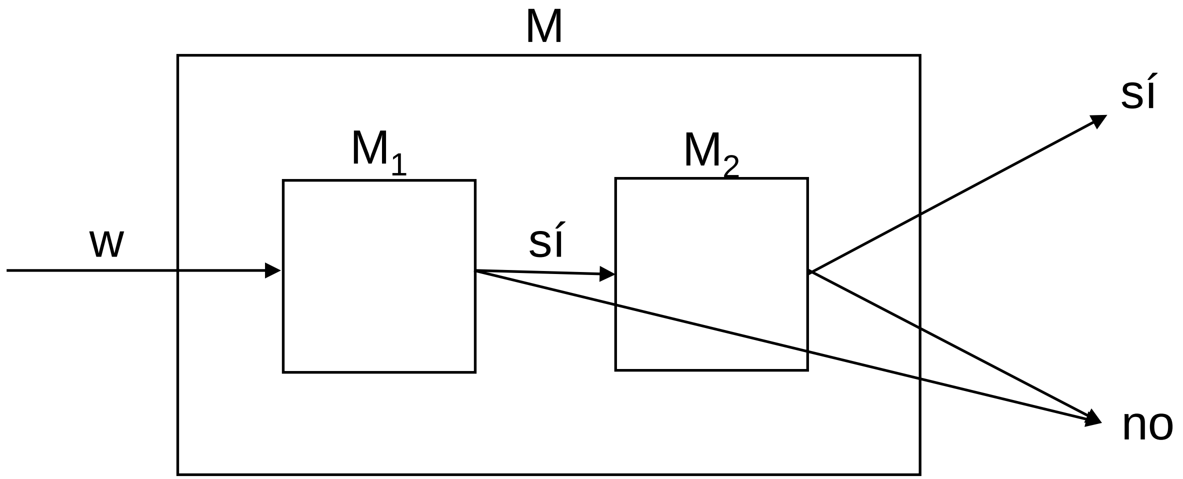
  Hay lenguajes en $\mathcal{L}$ – RE (regiones 3 y 4): **hay problemas no computables**.

- **Hicimos:**

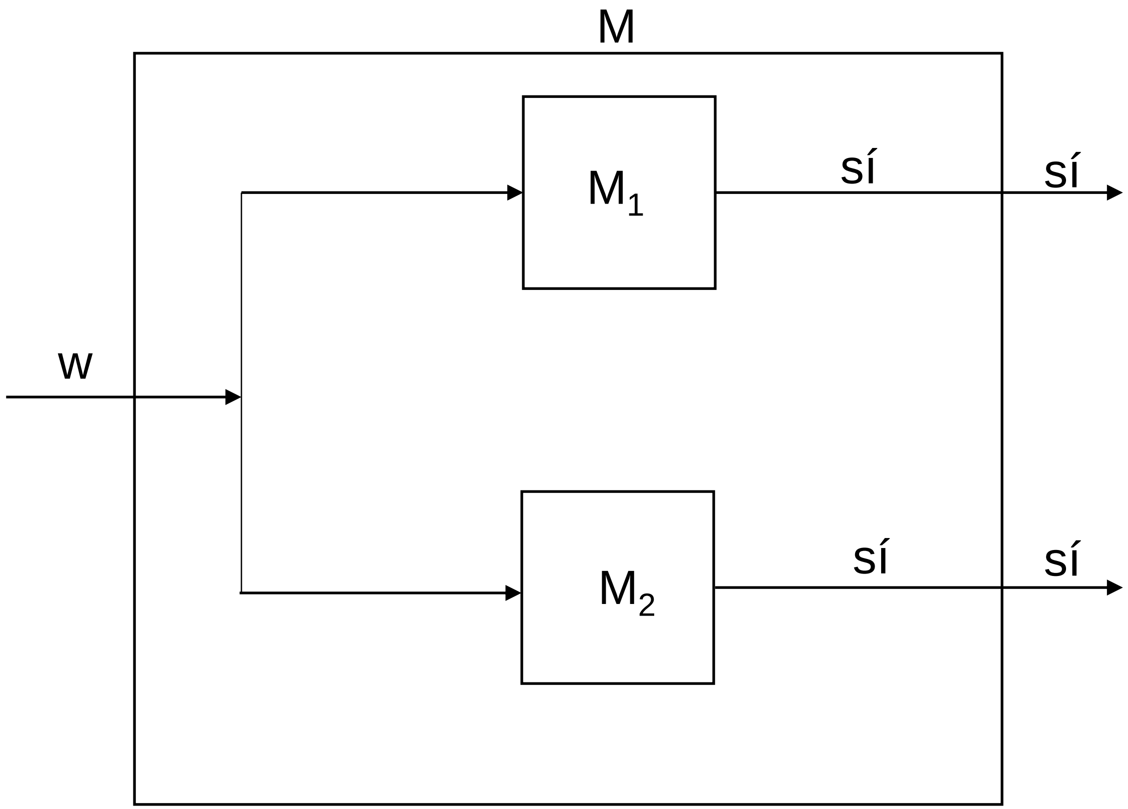
Construcción de MT para probar **pertenencia a R o RE**. Por ejemplo:



M decide el lenguaje de las cadenas $a^n b^n$, con $n \geq 1$



M decide la intersección de los lenguajes que deciden M_1 y M_2



M acepta la unión de los lenguajes que aceptan M_1 y M_2 (que pueden no parar)

- **Haremos:**

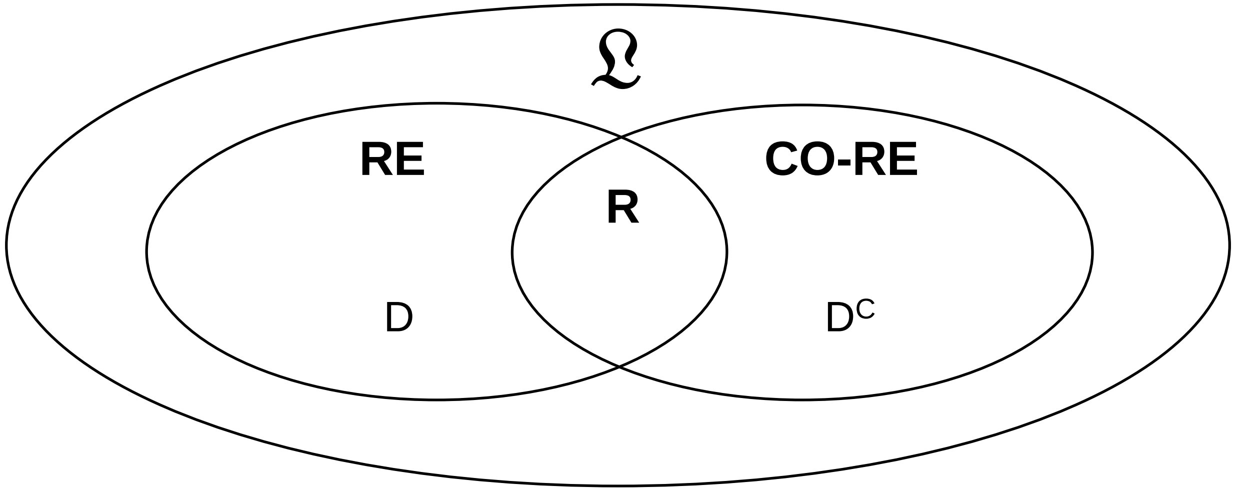
  Uso del método de **diagonalización** para probar **no pertenencia a R o RE.**

  (construyendo MT no se puede probar que un lenguale **NO** pertenece a una clase de lenguajes).

- Vamos a encontrar:

  Un primer lenguaje **D** en RE – R, para probar **R** $\subset$ **RE.**

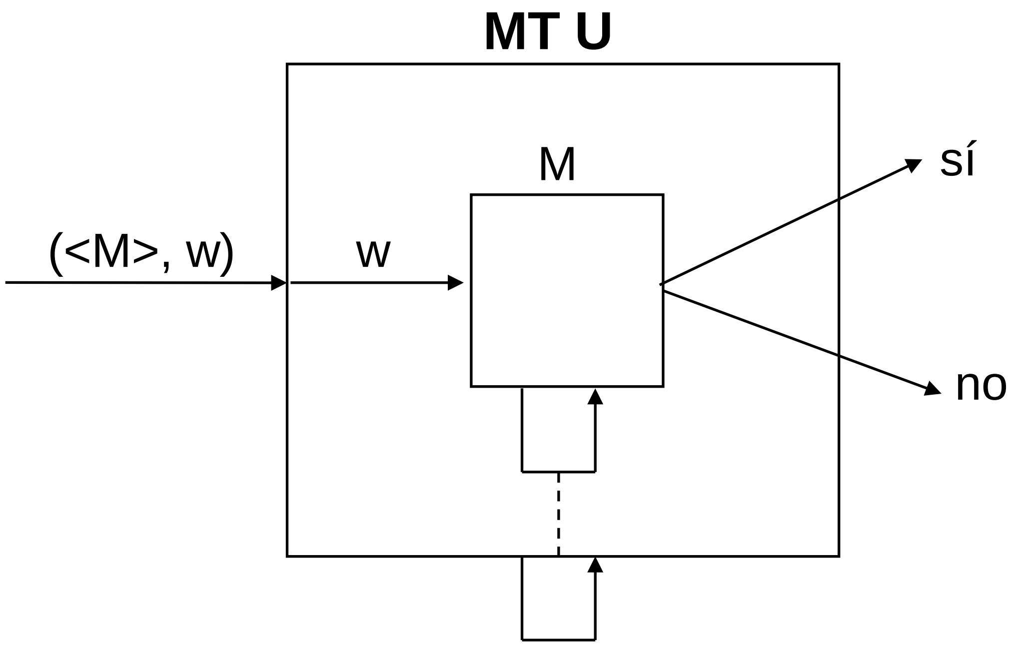
  Un primer lenguaje $\mathbf{D^C}$ en CO-RE – R, para probar **RE** $\subset \mathfrak{L}$**.**

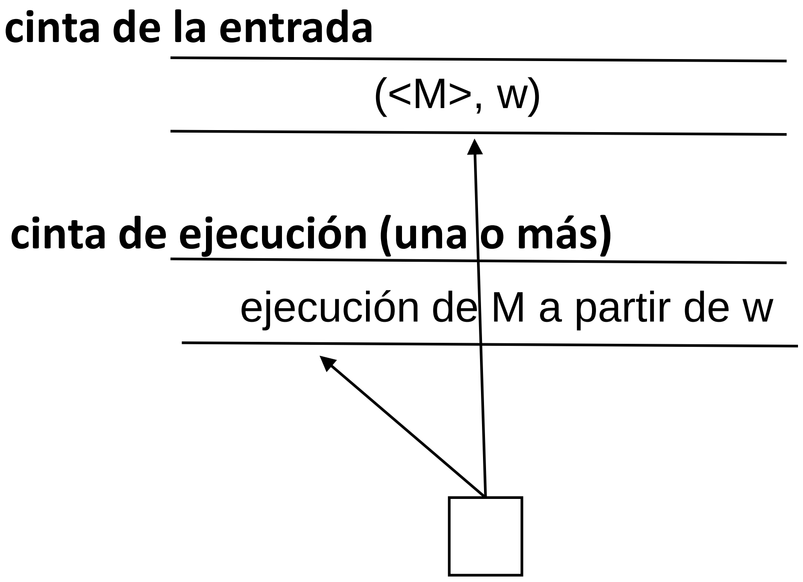


- Para ello, primero tenemos que introducir la **máquina de Turing universal.**

Máquina de Turing universal

- Una máquina de Turing universal (MT U) es una máquina de Turing capaz de ejecutar cualquier otra (noción de **programa almacenado**, Turing 1936). El esquema más general es:





La MT U recibe como entrada una **MT M** (codificada mediante una cadena <M>) y una **cadena w**, y **ejecuta M a partir de w**.

Codificación de una máquina de Turing y una cadena de entrada

- Hay varias maneras. P.ej., usando números en binario (cadenas de 1 y 0). Entre otras características:

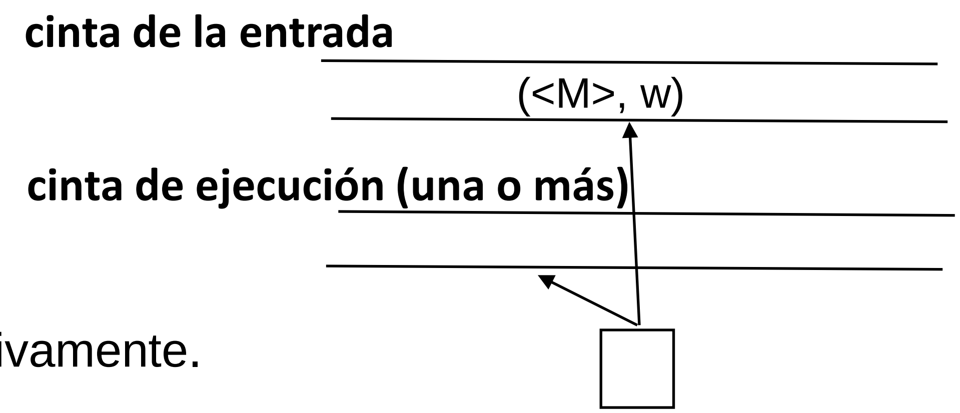
  - El código del estado inicial q_0 es 1,

    el código del estado q_A es 2,

    y el código del estado q_R es 3.

  - El código del símbolo blanco es 1.

  - Los movimientos R, L y S se codifican con 1, 2 y 3, respectivamente.



- <u>Ejemplo</u>:

  Si M es: $\delta(q_0, s) = (q_0, t, L)$, $\delta(q_0, t) = (q_0, s, L)$, $\delta(q_0, B) = (q_A, B, S)$,

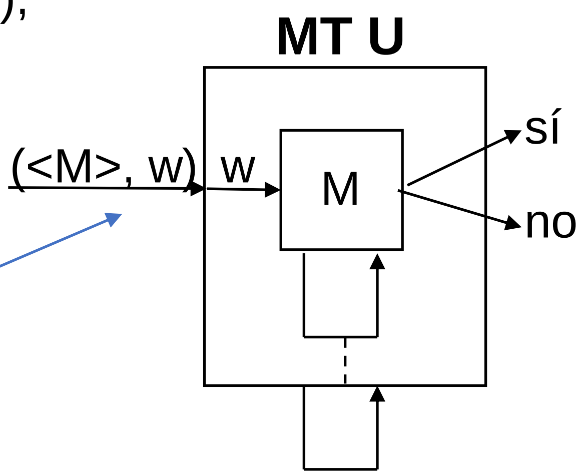
  y el código del símbolo s es 2 y el código del símbolo t es 3,

  queda: **<M> = (1,2,1,3,2),(1,3,1,2,2),(1,1,2,1,3).**

  Y si la cadena w es: st,

  queda: **(<M>, w) = (1,2,1,3,2),(1,3,1,2,2),(1,1,2,1,3)#2,3**

  (para simplificar la escritura, usamos w en lugar de <w>).



Enumeración de las máquinas de Turing

- Codificar las máquinas de Turing permite enumerarlas, por ejemplo en el **orden canónico:**

  1) Los códigos se ordenan de menor a mayor longitud.

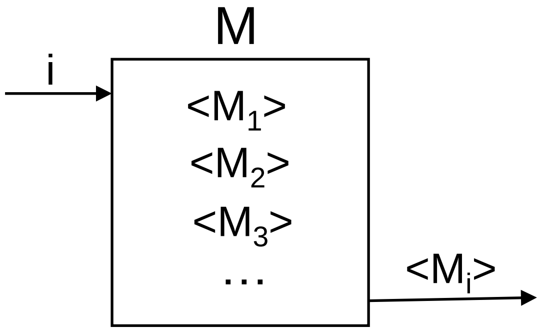
  2) Los códigos con longitudes iguales se ordenan según el orden alfanumérico.

- Por ejemplo, las primeras cadenas según el orden canónico, con ceros, unos, paréntesis y comas, son:

λ														
0	1	(	)	,										
00	01	0(	0)	0,	10	11	1(	1)	1,	(0	(1	((	…	
000	001	00(	00)	00,	010	011	01(	01)	01,	0(0	0(1	0((	…	etc.

- La siguiente MT M obtiene la MT i-ésima (MT M_i) según el orden canónico. Dado i, la MT M hace:

1. Hace n := 0.
2. Genera la siguiente cadena v según el orden canónico.
3. Valida que v sea el código de una MT. Si no lo es, vuelve al paso 2.
4. Si n = i, devuelve v **(v es el código de la MT M_i)** y para.
   Si n ≠ i, hace n := n + 1 y vuelve al paso 2.



Prueba de que R ⊂ RE ⊂ $\mathfrak{L}$

- La siguiente tabla T representa el comportamiento de **todas las MT M** con respecto a **todas las cadenas w**:

todas las cadenas

	T	w_0	w_1	w_2	w_3	w_4	…..
fila 0	M_0	1	0	1	1	1	…..
fila 1	M_1	1	0	0	1	0	…..
fila 2	M_2	0	0	1	0	1	…..
fila 3	M_3	0	1	1	1	1	…..
fila 4	M_4	0	1	1	1	0	…..
	…..	…..	…..	…..	….	…..	…..

todas las MT

$T(M_i, w_j) = 1$ si M_i **acepta** w_j; $T(M_i, w_j) = 0$ si M_i **rechaza** w_j (los valores son de ejemplo)

- La fila 0 = (1, 0, 1, 1, 1, …), representa el lenguaje $L(M_0) = \{w_0, w_2, w_3, w_4. …\}$.
- La fila 1 = (1, 0, 0, 1, 0, …), representa el lenguaje $L(M_1) = \{w_0, w_3, …\}$.
- La fila 2 = (0, 0, 1, 0, 1, …), representa el lenguaje $L(M_2) = \{w_2, w_4. …\}$.
- Etc.

- Por lo tanto, las filas representan **todos los lenguajes aceptados por MT**, es decir el conjunto: **RE.**

- Consideremos en particular la **diagonal** de la tabla T:

	T	w_0	w_1	w_2	w_3	w_4	.....
fila 0	M_0	**1**	0	1	1	1	.....
fila 1	M_1	1	**0**	0	1	0	.....
fila 2	M_2	0	0	**1**	0	1	.....
fila 3	M_3	0	1	1	**1**	1	.....
fila 4	M_4	0	1	1	1	**0**	.....
	.....	.....	.....	.....	....	.....	.....

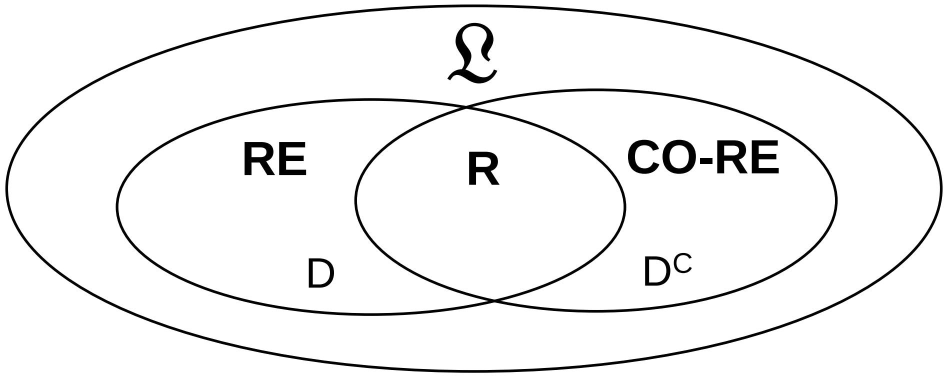
- La diagonal es **(1, 0, 1, 1, 0, …)**, y representa el lenguaje **D = {w_i | M_i acepta w_i}.**
  En el ejemplo: D = {w_0, w_2, w_3, …}.

- Y la diagonal con los 1 y 0 invertidos es **(0, 1, 0, 0, 1, …)**, y representa el lenguaje **D^C = {w_i | M_i rechaza w_i}.**
  En el ejemplo: D^C = {w_1, w_4, …}.

- Vamos a probar:

  (1) **D está en RE**
  (2) **D^C no está en RE**
  Y por lo tanto: (3) **D no está en R** ¿Por qué?

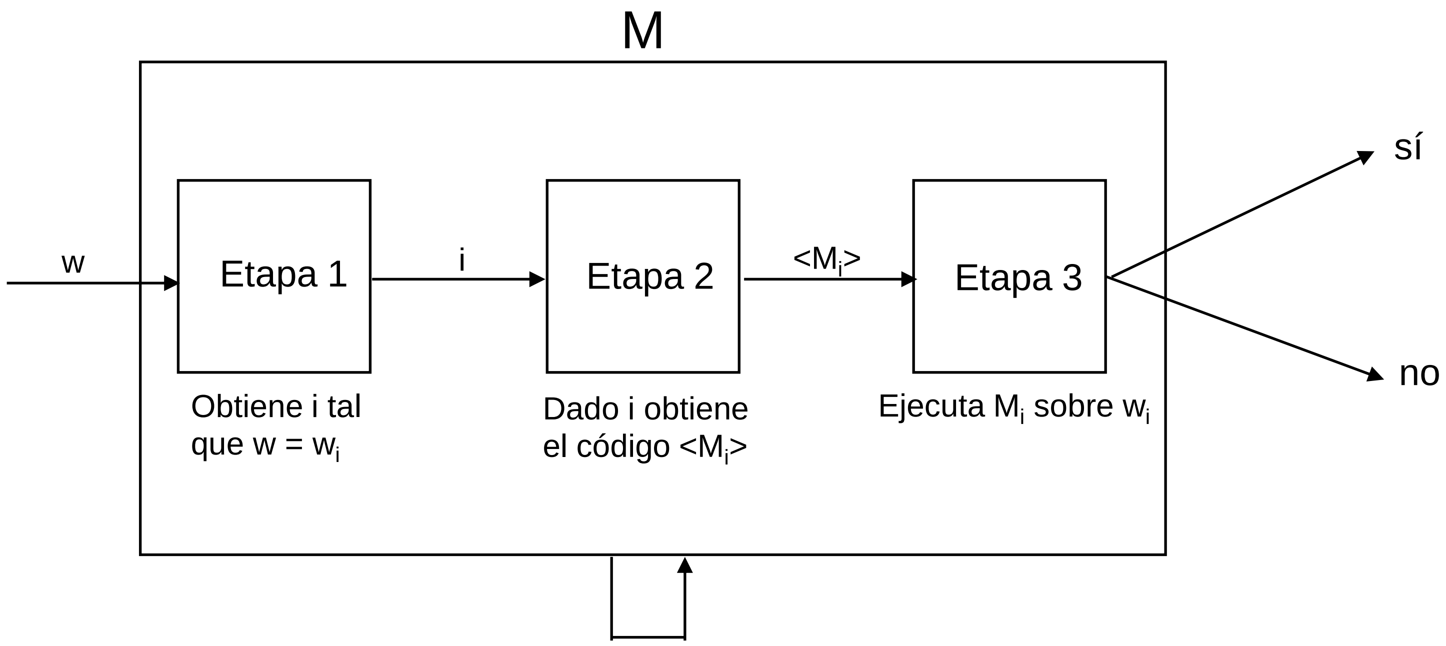
  Porque si D está en R, D^C está en RE.



(1) Primero probaremos D = {w_i | M_i acepta w_i} está en RE:

- La siguiente MT M acepta D. Dada una entrada w, M hace:

1. Encuentra i tal que w = w_i en el orden canónico (genera cadenas en el orden canónico hasta encontrar w).
2. Encuentra el código <M_i> de la MT M_i (genera códigos de MT en el orden canónico hasta llegar a <M_i>).
3. Ejecuta M_i sobre w_i y acepta sii M_i acepta.



Por lo tanto, se cumple D = {w_i | M_i acepta w_i} ∈ RE

(2) Ahora probaremos $D^C = \{w_i \mid M_i \text{ rechaza } w_i\}$ no está en RE:

	T	w_0	w_1	w_2	w_3	w_4	.....
fila 0	M_0	**1**	0	1	1	1	.....
fila 1	M_1	1	**0**	0	1	0	.....
fila 2	M_2	0	0	**1**	0	1	.....
fila 3	M_3	0	1	1	**1**	1	.....
fila 4	M_4	0	1	1	1	**0**	.....
	.....	.....	.....	.....	....	.....	**.....**

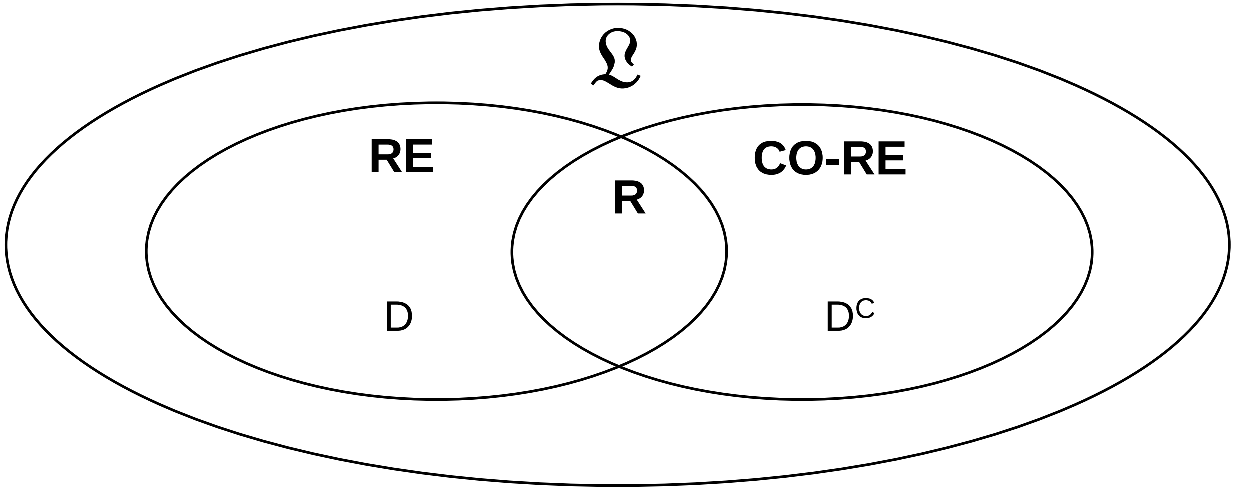
- Vimos que la diagonal con los 1 y 0 invertidos es **(0, 1, 0, 0, 1, …),** y que representa $D^C = \{w_i \mid M_i \text{ rechaza } w_i\}$.
- Notar que dicha diagonal **es diferente de todas las filas**:

  Es diferente de la fila 0 = (1, 0, 1, 1, 1,…), que representa el lenguaje $L(M_0)$, en el primer elemento.
  Es diferente de la fila 1 = (1, 0, 0, 1, 0,…), que representa el lenguaje $L(M_1)$, en el segundo elemento.
  Es diferente de la fila 3 = (0, 0, 1, 0, 1,…), que representa el lenguaje $L(M_2)$, en el tercer elemento.
  Etc.

- Y como las filas representan todos los lenguajes de RE, **D^C es diferente de todos los lenguajes de RE.**

Por lo tanto, se cumple $D^C = \{w_i \mid M_i \text{ rechaza } w_i\} \notin RE$

- Así encontramos dos primeros lenguajes fuera de R (el lenguaje D^C por **diagonalización**):



$D = \{w_i \mid M_i \text{ acepta } w_i\}$

$D^C = \{w_i \mid M_i \text{ rechaza } w_i\}$

- A partir de estos primeros lenguajes se pueden encontrar otros lenguajes fuera de R y RE, con un método más sencillo, la **reducción** (lo veremos en la clase siguiente).

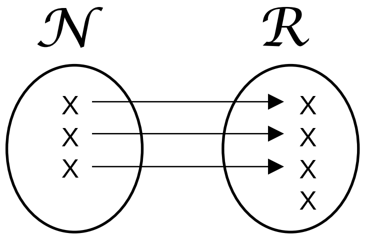
- La diagonalización es muy útil para encontrar **primeros lenguajes en un conjunto**.

- Consiste básicamente en encontrar un lenguaje **separador.** Por ejemplo, el lenguaje D^C actúa como separador entre los conjuntos RE y $\mathcal{L}$ (no pertenece a RE y pertenece a $\mathcal{L}$).

Anexo

Otro ejemplo de diagonalización

- Se prueba por diagonalización que el conjunto $\mathcal{R}$ de los números reales es más grande que el conjunto $\mathcal{N}$ de los números naturales (enteros positivos), es decir que **los números reales no se pueden enumerar**:



Hay más números reales que números naturales.
En símbolos: **$|\mathcal{R}| > |\mathcal{N}|$.**
(|C| denota el tamaño o la cardinalidad del conjunto C).

Supongamos que no se cumple $|\mathcal{R}| > |\mathcal{N}|$ (**llegaremos a una contradicción**). Sea, por ej., la siguiente enumeración de **todos** los números reales entre el 0 y el 1 (alcanza con considerar este intervalo):

0,**1**287…
0,8**5**50…
0,13**8**0…
0,275**1**…
etc.

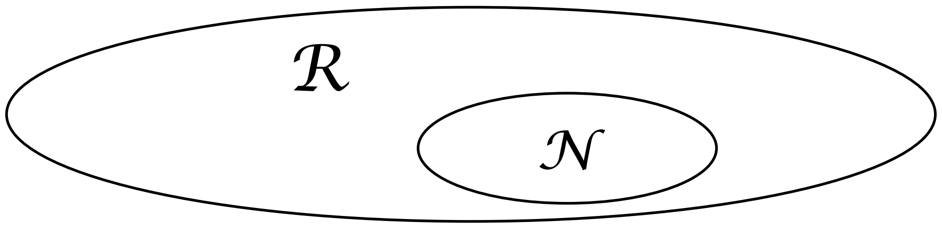
Y sea el número coloreado **0,1581…** Sumando 1 a sus decimales obtenemos **0,2692…** Veamos que este número entre el 0 y el 1 no está en la enumeración (**contradicción de acuerdo a lo que asumimos**):

0,**2**692… difiere de 0,**1**287… en el primer decimal.
0,2**6**92… difiere de 0,8**5**50… en el segundo decimal.
0,26**9**2… difiere de 0,13**8**0… en el tercer decimal.
Etc.

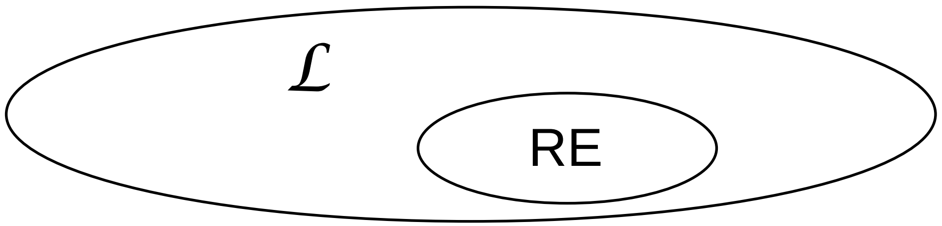
Por lo tanto, $|\mathcal{R}| > |\mathcal{N}|$

¡OTRA MALA NOTICIA!

- El conjunto $\mathcal{R}$ de los números reales es **mucho más grande** que el conjunto $\mathcal{N}$ de los números naturales:



y como $|\mathcal{R}| = |\mathcal{L}|$ y $|\mathcal{N}| = |\text{RE}|$, entonces $\mathcal{L}$ **es mucho más grande** que RE:



¡Hay muchos más problemas que problemas computables!

- **$|\mathcal{N}| = |\text{RE}|$** porque las MT se pueden enumerar, y entonces también los lenguajes de RE se pueden enumerar.

- **$|\mathcal{R}| = |\mathcal{L}|$** porque:

  Se prueba que $|\mathcal{R}| = |P(\mathcal{N})|$, siendo $P(\mathcal{N})$ el conjunto de partes de $\mathcal{N}$ (el conjunto de todos sus subconjuntos).

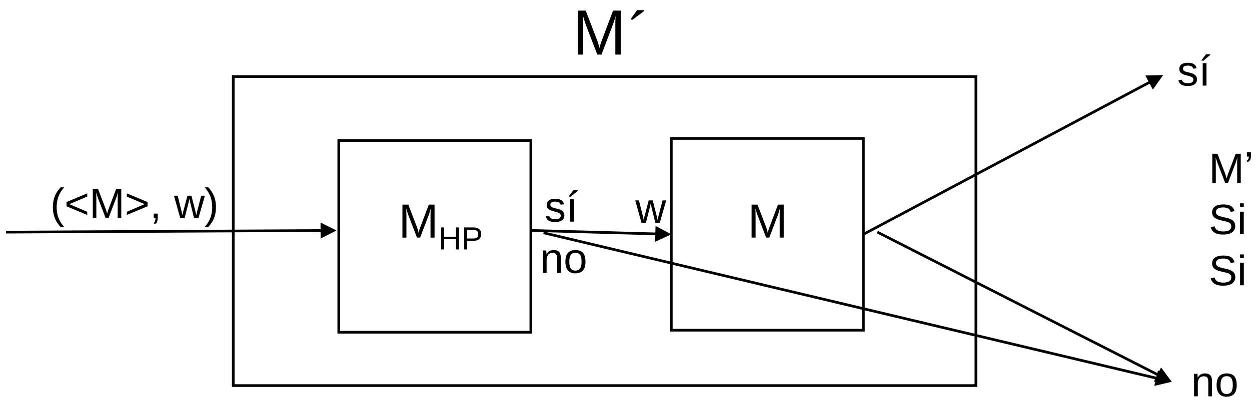
  $|\mathcal{N}| = |\Sigma^*|$, porque las cadenas de Σ^* se pueden enumerar. Así, $|P(\mathcal{N})| = |P(\Sigma^*)|$.

  $P(\Sigma^*) = \mathcal{L}$, porque $\mathcal{L}$ es el conjunto de todos los subconjuntos de Σ^*. Así, $|P(\Sigma^*)| = |\mathcal{L}|$.

  En consecuencia: **$|\mathcal{R}| = |P(\mathcal{N})| = |P(\Sigma^*)| = |\mathcal{L}|$.**

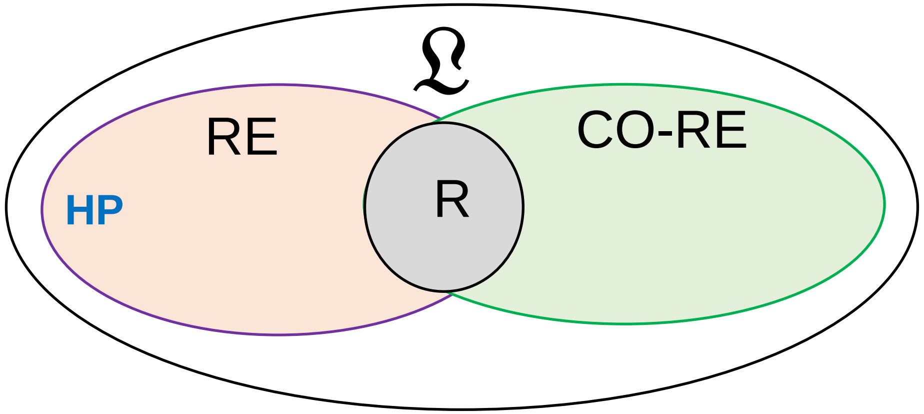
Nueva visita al *halting problem*

- HP = {(<M>, w) | M para a partir de w} **está entre los lenguajes más difíciles de RE**.
- Una forma de verlo es probando que **si HP fuera recursivo, entonces todo lenguaje de RE sería recursivo**:
- Sea M_{HP} una MT que decide HP y M una MT que reconoce algún lenguaje L de RE. La siguiente MT M´ decide L:



M’ primero decide si M para a partir de w.
Si M no para, M´ rechaza (M no acepta w).
Si M para, M´ ejecuta M a partir de w y responde como M.

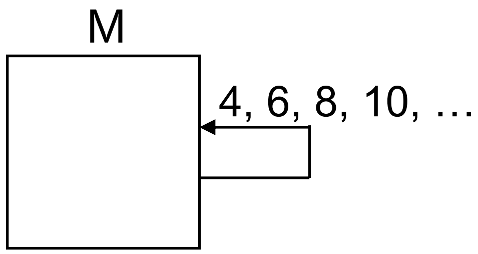
HP está en la frontera de RE, lo más lejos posible de R.
Se dice que es **RE-completo,** identifica el grado de dificultad de RE, representa el **problema general de la indecibilidad**.



- Siguiendo con lo anterior, si HP estuviera en R, es decir, si existiera una MT M_{HP} que decide HP, entonces muchos enunciados matemáticos se probarían fácilmente. Por ejemplo:

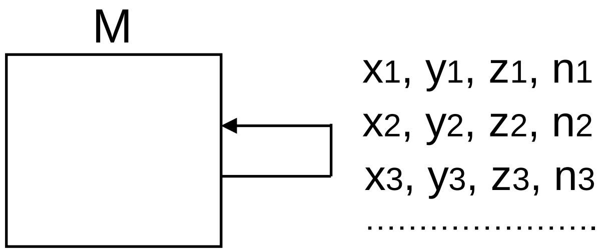
Conjetura de Goldbach (no resuelta al día de hoy)
Todo número par mayor que 2 es la suma de 2 números primos:
$4 = 2 + 2$, $6 = 3 + 3$, $8 = 3 + 5$, $10 = 5 + 5$, $12 = 5 + 7$, $14 = 7 + 7$, …

Resolución:
- Se construye una MT M que va probando con 4, 6, 8, 10, etc., y que para sii encuentra un par que no cumple la conjetura.
- Se ejecuta M_{HP} a partir de <M> y se acepta la conjetura sii M_{HP} rechaza.



Ultimo Teorema de Fermat (resuelto por A. Wiles en 1995)
En el marco de los números naturales, no existe $n > 2$ tal que $x^n + y^n = z^n$ (salvo las soluciones triviales con los valores 1 y 0).

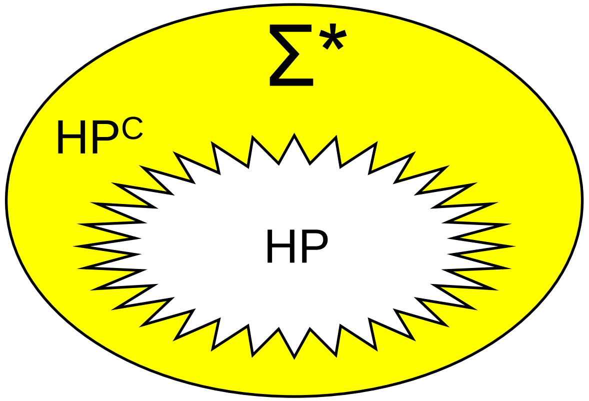
Resolución:
- Se construye una MT M que va probando con todas las combinaciones de x, y, z, n, y que para sii encuentra una combinación que cumple la igualdad.
- Se ejecuta M_{HP} a partir de <M> y se acepta el enunciado sii M_{HP} rechaza.



Relación entre la computabilidad y el tamaño de un lenguaje

- **HP $\subseteq$ Σ*, y HP es más difícil que Σ***

  HP ∉ R y Σ* ∈ R

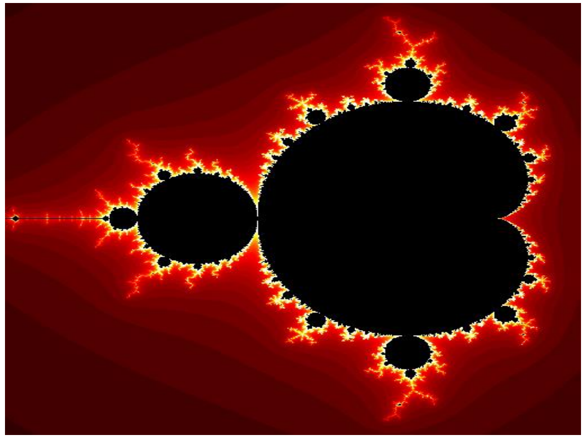


- **HP^C $\subseteq$ Σ*, y HP^C es más difícil que Σ***

  HP^C ∉ RE y Σ* ∈ R

- La computabilidad de un lenguaje (o problema) tiene más que ver con su definición, su **contorno** (si consideramos su representación gráfica en el plano), que con su **tamaño**.

CONJUNTO DE MANDELBROT

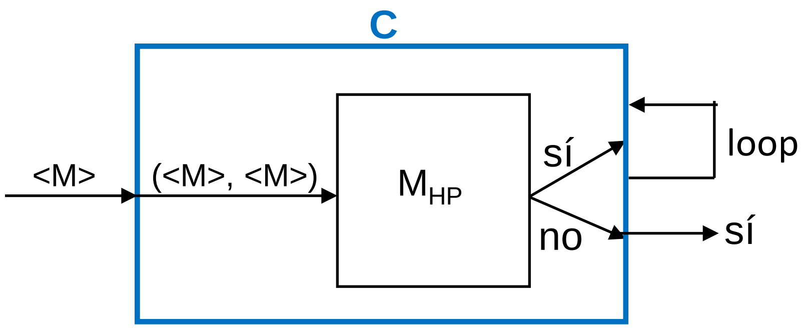


- Un ejemplo que ilustra muy bien la relación entre computabilidad y contorno es el Conjunto de Mandelbrot, perteneciente a CO-RE – R.

La prueba de Turing de que el lenguaje HP no es recursivo

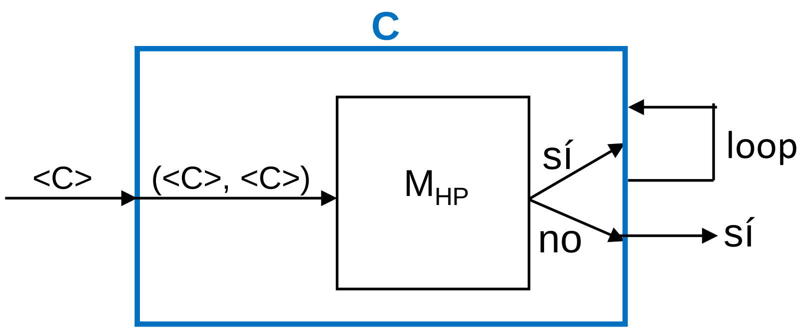
- Supondremos que HP = {<M>, w) | M para a partir de w} ∈ R y **llegaremos a una contradicción**.

Sea M_{HP} una MT que decide HP. Utilizando M_{HP} podemos construir la siguiente MT C:



Si M_{HP} responde que M para desde <M>, C entra en loop
Si M_{HP} responde que M no para desde <M>, C para
Es decir, C "le lleva la contra" a M_{HP}

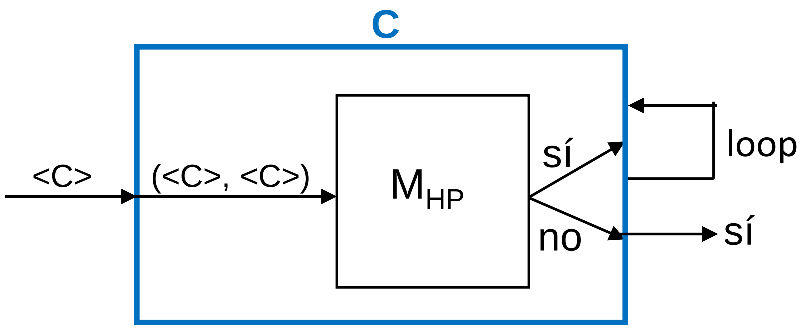
Veamos qué sucede cuando la entrada de C es su propio código <C>:



Si M_{HP} responde que C para desde <C>, C entra en loop (**absurdo**)
Si M_{HP} responde que C no para desde <C>, C para (**absurdo**)

Así, C no puede existir, y como se construyó a partir de M_{HP} , tampoco M_{HP} puede existir. **Por lo tanto, HP ∉ R.**

- En este caso, la diagonalización se basa en una **autorreferencia**:



Dos niveles de "discurso":
C como MT universal que ejecuta C desde <C>.
C como MT común que se ejecuta desde <C>.

- La autorreferencia es muy útil para probar algunos enunciados. Por ejemplo, la indecibilidad en la aritmética. En este caso, una fórmula ψ expresa tanto una propiedad de la aritmética como una propiedad de los números.

- Se caracteriza por encontrar **paradojas** (como la que vimos en la prueba de Turing). Otras clásicas son:

Paradoja del mentiroso. Si Juan dice: "estoy mintiendo", entonces:
si Juan miente, dice la verdad, y si en cambio dice la verdad, miente.

Paradoja del barbero. El barbero de un pueblo sólo afeita a quienes no se afeitan a sí mismos.
¿El barbero se afeita a sí mismo? Sí sí, entonces no, y si no, entonces sí.

Paradoja de Russell. Sea U el conjunto de todos los conjuntos que no son elementos de sí mismos.
¿El conjunto U es elemento de sí mismo? Si sí, entonces no, y si no, entonces sí.

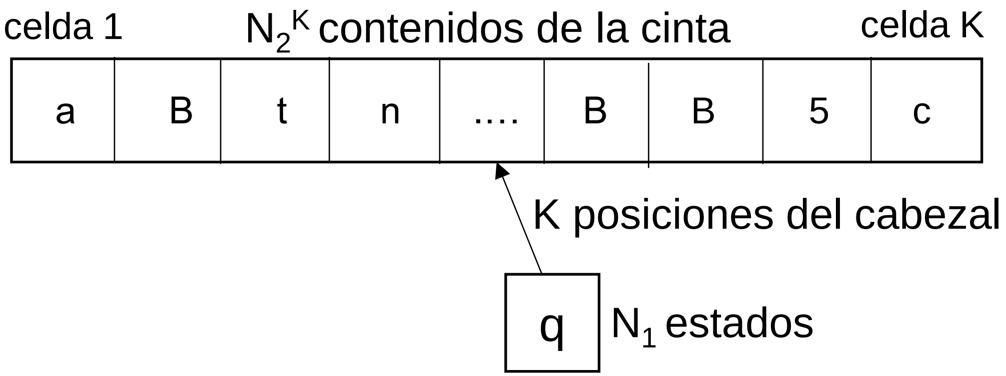
Clase práctica 3

¿Cómo burlar al *halting problem*? (1)

- No existe una MT que pueda decidir, **para todos los pares (<M>, w),** si M para a partir de w.

- Pero **para ciertos casos particulares**, el *halting problem* es decidible.

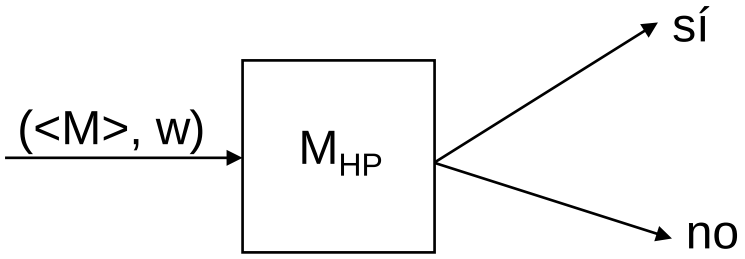
- Un ejemplo es cuando una MT M siempre ocupa un **espacio acotado de celdas**:
  Sea M una MT con una cinta, N_1 estados, N_2 símbolos, y que recorre a lo sumo K celdas para toda entrada w.
  Entonces, si M loopea, en algún momento **repite una configuración** (posición de cabezal, estado, cinta):



Total: $N = K.N_1.N_2^K$ configuraciones distintas

Así, la siguiente MT M_{HP} decide si M para a partir de una cadena w:

- Ejecuta M a partir de w.
- Si al cabo de N pasos M para, acepta. En caso contrario, rechaza.



¿Cómo burlar al *halting problem*? (2)

- ¿Cómo detectar si una MT M acepta al menos una cadena?

Tenemos que tener cuidado en cómo construimos una MT M´ que chequee si M acepta una cadena.
No sirve que M´ ejecute M sobre la 1ra cadena, luego sobre la 2da, luego sobre 3ra, … (según el orden canónico), porque M puede no detenerse en algunos casos.

Solución:

1. Hacer i := 1.
2. Ejecutar i pasos de M sobre todas las cadenas de longitud a lo sumo i.
3. Si M acepta alguna vez, aceptar.
4. Si no, hacer i := i + 1 y volver al paso 2.

pasos	**cadenas**
1	λ w_0 w_1 w_2 w_3 …
2	λ w_0 w_1 w_2 … w_0w_0 w_0w_1 w_0w_2 …
3	λ w_0 w_1 … w_0w_0 w_0w_1 … $w_0w_0w_0$ $w_0w_0w_1$ …
…	……………………………………………………………..

Por ejemplo, si la primera cadena que M acepta tiene 80 símbolos y M la acepta en 120 pasos, entonces en la iteración 120 la MT M´ la va a encontrar.