

**Def.: orden canónico para  $\Sigma^*$ :** se listan todas las palabras en orden según su tamaño con las palabras del mismo tamaño en orden lexicográfico.

Ej:  $\Sigma = \{0, 1\}$ , el orden canónico es:

$\lambda, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, 0000, 0001 \dots$   
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17

Obsérvese que si  $w$  es un string de  $\{0,1\}^*$ , la posición  $i$  que ocupa en el orden canónico se escribe en binario como  $1w$ . Decimos entonces que  $w$  es el  $i$ -ésimo string y por ello lo denotamos  $w_i$

Por ejemplo el string  $\lambda$  ocupa la posición 1 ( $1\lambda$ ) el string 01 ocupa la posición 5 (101), el string 0000 la posición 16 (10000)

**Pregunta 1:** ¿Puede una MT generar las palabras de  $\Sigma^*$  en orden canónico?

Rta.: Sí. Idea: ir sumando 1 en binario, cuando el resultado necesita un bit más, se ponen todos los bits en cero y se vuelve al proceso de sumar uno en binario.

**Ejercicio de deber:** construir una MT que escriba en la primera cinta las palabras de  $\{0, 1\}^*$  en orden canónico separadas por “;”

**Pregunta 2:** Si  $L$  es un lenguaje recursivo ( $L \in R$ ) ¿Puede una MT generar todas las palabras de  $L$  en orden canónico?

Rta.: Sí, porque existirá alguna MT  $M$  que reconoce  $L$  y siempre se detiene. Se construye una MT  $M'$  que va generando en una cinta los strings de  $\Sigma^*$  en orden canónico, simula  $M$  sobre cada string generado y si  $M$  lo acepta  $M'$  lo escribe en la cinta 1.

**Pregunta 3:** si  $L$  es un lenguaje recursivamente enumerable ( $L \in \text{RE}$ ) ¿Puede una MT generar todas las palabras de  $L$ ?

Rta.: Sí. Se generan todos los pares  $(i, j)$  en orden de su suma,  $i+j$ , y entre los de igual suma en orden creciente de  $i$  (ver ejercicio de la práctica 1)

**$(1, 1); (1, 2); (2, 1); (1, 3); (2, 2); (3, 1), \dots$**

Por cada par  $(i, j)$  generado se simulan  $j$  pasos de la MT  $M$  que reconoce el lenguaje  $L$  ( $L=L(M)$ ), sobre  $w_i$  ( $i$ -ésimo string de  $\Sigma^*$  en orden canónico). Si  $M$  acepta  $w_i$  en esos  $j$  pasos  $\Rightarrow$  se escribe  $w_i$  en la cinta 1.

**Pregunta 4:** ¿Puede codificarse una MT como un string de un alfabeto de 2 símbolos?

Rta.: Sí. Se puede hacer de muchas formas, acá se muestra una.

Ej: se quiere codificar  $M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R \rangle$

$Q = \{q_0, q_1, \dots, q_K\}$   $\Sigma = \{a_1, a_2, \dots, a_L\}$   $\Gamma = \{B, a_1, a_2, \dots, a_L, a_{L+1}, \dots, a_n\}$

Se puede codificar M con un alfabeto binario  $\{0, 1\}$  de la sgte. forma:

Estados:  $q_A = 1$      $q_R = 11$      $q_0 = 111$      $q_1 = 1111$     ...     $q_i = 1^{(i+3)}$

Símbolos:  $B = 1$ ,     $a_1 = 11$      $a_2 = 111$     ...     $a_i = 1^{(i+1)}$

Movimiento del Cabezal:  $D = 1$      $I = 11$      $S = 111$

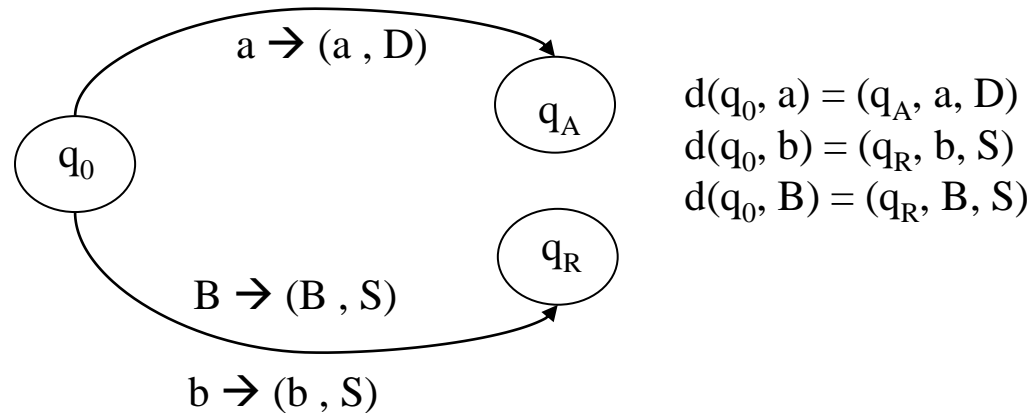
Función de Transición: cada transición se codifica como una quintupla de elementos separados por un símbolo 0. Ej.:

$\delta(q_0, a_2) = (q_1, a_4, D)$  se codifica como

$(q_0, a_2, q_1, a_4, D) = (11101110111101111101)$   
 $q_0, \quad a_2, \quad q_1, \quad a_4, \quad D$

M se codifica como una sucesión de quintuplas separadas por 00.

Ej.:  $M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R \rangle$      $Q = \{q_0\}$      $\Sigma = \{a, b\}$      $\Gamma = \{B, a, b\}$



Ej.:  $M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R \rangle$      $Q = \{q_0\}$      $\Sigma = \{a, b\}$      $\Gamma = \{B, a, b\}$

$q_A = 1$	$q_R = 11$	$q_0 = 111$
$B = 1$	$a = 11$	$b = 111$
$D = 1$	$I = 11$	$S = 111$

$\langle M \rangle = 1110110101101001110111011011101110011101011010111$   
 $\delta(q_0, a) = (q_A, a, D)$      $\delta(q_0, b) = (q_R, b, S)$      $\delta(q_0, B) = (q_R, b, S)$

Note que existen otros posible códigos para  $M$ . Las tres transiciones pueden listarse en distinto orden ( $3!$  formas distintas) por lo tanto hay 6 códigos distintos para  $M$

**Pregunta 5:** ¿ $L_{\langle M \rangle} \in R$ ?

Con  $L_{\langle M \rangle} = \{w \in \{0, 1\}^* / w \text{ es el código bien formado de una MT}\}$

Rta.: Sí, para demostrarlo hay que construir una MT que realice un chequeo sintáctico y siempre se detenga.

**Ejercicio para el lector:** proponer una forma de codificar una MT del modelo original (con conjunto de estados finales  $F$ ) y construir una MT que acepte  $L_{\langle M \rangle}$  para esa codificación.

El código binario de una MT  $M$  se denotará con  $\langle M \rangle$

**Def.** Se denomina  $i$ -ésima máquina de Turing y se denota  $M_i$  a aquella MT cuyo código es  $w_i$  ( $i$ -ésimo string binario en orden canónico), es decir  $\langle M_i \rangle = w_i$

Si  $w_i$  no es un código válido de MT se toma  $M_i$  como la máquina de Turing que se detiene inmediatamente rechazando cualquier input ( $L(M_i) = \{ \}$ ). Así, todo string  $w_i$  de  $\{0,1\}^*$  se corresponde con una MT  $M_i$

**Def.:** se define el lenguaje Diagonal  $L_D$  (primer ejemplo de lenguaje no recursivamente enumerable que se verá) de la siguiente manera:

$$L_D = \{ w_i \in \Sigma^* / w_i \notin L(M_i) \}$$

siendo  $\Sigma = \{0,1\}$ ;  $w_i$  el  $i$ -ésimo string en orden canónico de  $\Sigma^*$  y  $M_i$  la  $i$ -ésima MT



$$L_D = \{w_i \in \Sigma^* / w_i \notin L(M_i)\}$$

**Teorema:**  $L_D \notin \text{RE}$

**Dem.:** Por reducción al absurdo, se asume que  $L_D \in \text{RE} \Rightarrow$  existirá una MT  $M$  y un natural  $k$  para el cual  $M = M_k$  y  $L_D = L(M_k)$ .

Considerando  $w_k$  el  $k$ -ésimo string de  $\Sigma^*$ , hay dos posibilidades, o bien  $w_k \in L_D$  o bien  $w_k \notin L_D$ .

- a)  $w_k \in L_D \Rightarrow w_k \notin L(M_k) \Rightarrow$  (por def.  $L_D$ )  
 $\Rightarrow w_k \notin L_D$  (porque  $L(M_k) = L_D$ )  
(contradicción)
- b)  $w_k \notin L_D \Rightarrow w_k \in L(M_k)$  (por def.  $L_D$ )  
 $\Rightarrow w_k \in L_D$  (porque  $L(M_k) = L_D$ )  
(contradicción)

En ambos casos se llega a una contradicción y por ende no puede existir una  $M_k$  que reconozca al lenguaje  $L_D$ , por lo tanto  $L_D$  no puede ser recursivamente enumerable.

**Por lo tanto  $L_D \notin \text{RE}$ .**