

Máquinas de Turing

Pablo Castro - UNRC

Un poco de historia

La idea de “algoritmo” se utilizó por mucho tiempo aún cuando las computadoras no existían:

- Unas tablas de arcillas encontradas en Bagdad del año 1600 antes de Cristo, describen el algoritmo de la división.
- La criva de Eratostenes es 200~ antes de Cristo, describe un procesos para encontrar números primos:
- El algoritmo de Euclides para encontrar el máximo común divisor

Más sobre algoritmos

Sin embargo la definición estos algoritmos es informal, por ejemplo:

Supongamos que queremos encontrar todos los números primos menores a un N dado, entonces:

- *Listar todos los números hasta n : $1, 2, 3, \dots, N$;*

- *Para todo $i = 2, \dots, \sqrt{n}$ hacer:*

Tachar todos los números desde $i + 1, \dots, n$ que son múltiplos de i .

- *Los números que quedan sin tachar son los números primos.*

El problema de Hilbert



David Hilbert planteó en 1900 los 23 problemas más importantes para los matemáticos de su tiempo

El problema número 10 de Hilbert era el siguiente:

“Existe un procedimiento que permita en una secuencia de pasos finitos determinar si un polinomio tiene una raíz integral.”

Por ejemplo: $6x^3yz^2 + 3xz^2$ es un polinomio de grado 3, se debe decir si hay valores enteros de x, y, z tal que:

Si existiera tal procedimiento se lo debe describir. Si no existiera, es más difícil se debe demostrar el por qué tal procedimiento es imposible.

Las Máquinas de Turing



El décimo problema de Hilbert inspiró a Alan Turing a tratar definir la idea de algoritmo para poder demostrar que existían que eran no computables

230

A. M. TURING

[Nov. 12,

ON COMPUTABLE NUMBERS, WITH AN APPLICATION TO
THE ENTSCHEIDUNGSPROBLEM

By A. M. TURING.

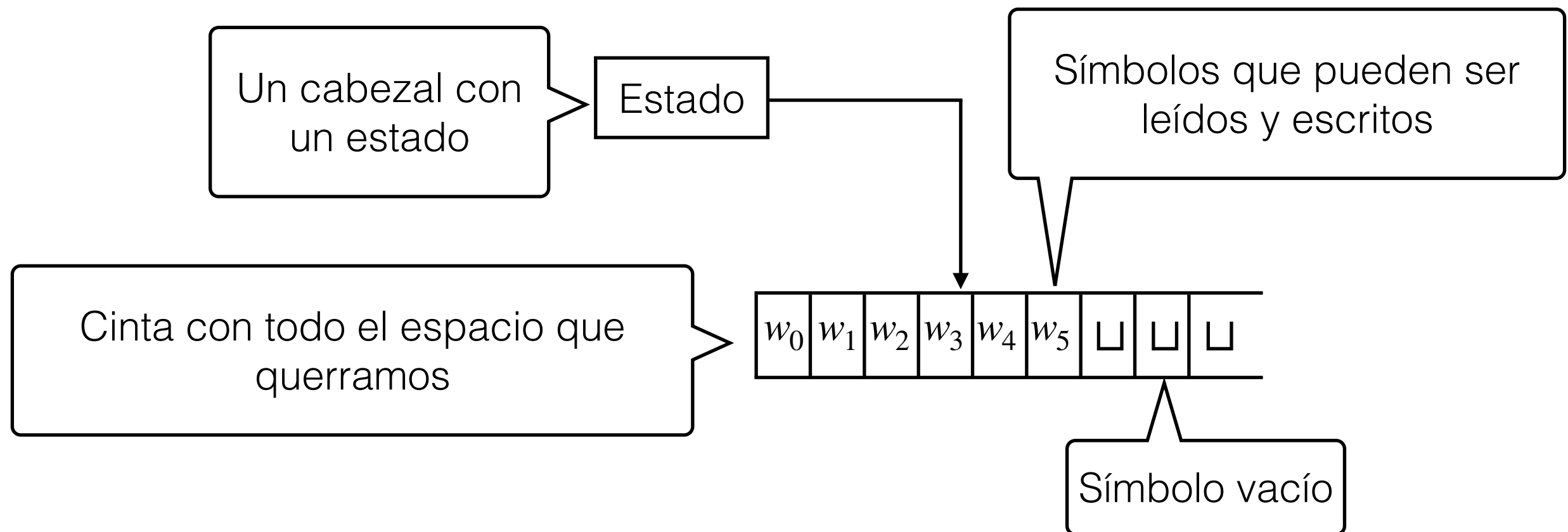
[Received 28 May, 1936.—Read 12 November, 1936.]

The “computable” numbers may be described briefly as the real numbers whose expressions as a decimal are calculable by finite means. Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of an integral variable or a real or computable variable, computable predicates, and so forth. The fundamental problems involved are, however, the same in each case, and I have chosen the computable numbers for explicit treatment as involving the least cumbersome technique. I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers. According to my definition, a number is computable if its decimal can be written down by a machine.

Es un artículo en 1936, Turing describe las máquinas de Turing, y demuestra que existen problemas no computables.

Las Máquinas de Turing

Las máquinas de Turing pueden pensarse como una formalización de la noción de computación:



En cada paso la máquina lee un símbolo, puede escribir un símbolo en la cinta y moverse a la izq. o der.

Definición Formal

Una máquina de Turing es una tupla:

$$\langle Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R \rangle$$

En donde:

- Q , es un conjunto finitos de estados,
- Σ , es el alfabeto de entrada conteniendo \sqcup
- Γ , es el alfabeto de la cinta, $\Sigma \subseteq \Gamma$
- $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$, es la función de transición,
- q_0 , es el estado inicial,
- q_A , es el estado de aceptación,
- q_R , es el estado de rechazo.

Intuición

Dada una máquina de Turing $M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R \rangle$ y un cadena de entrada w la máquina funciona de la siguiente manera:

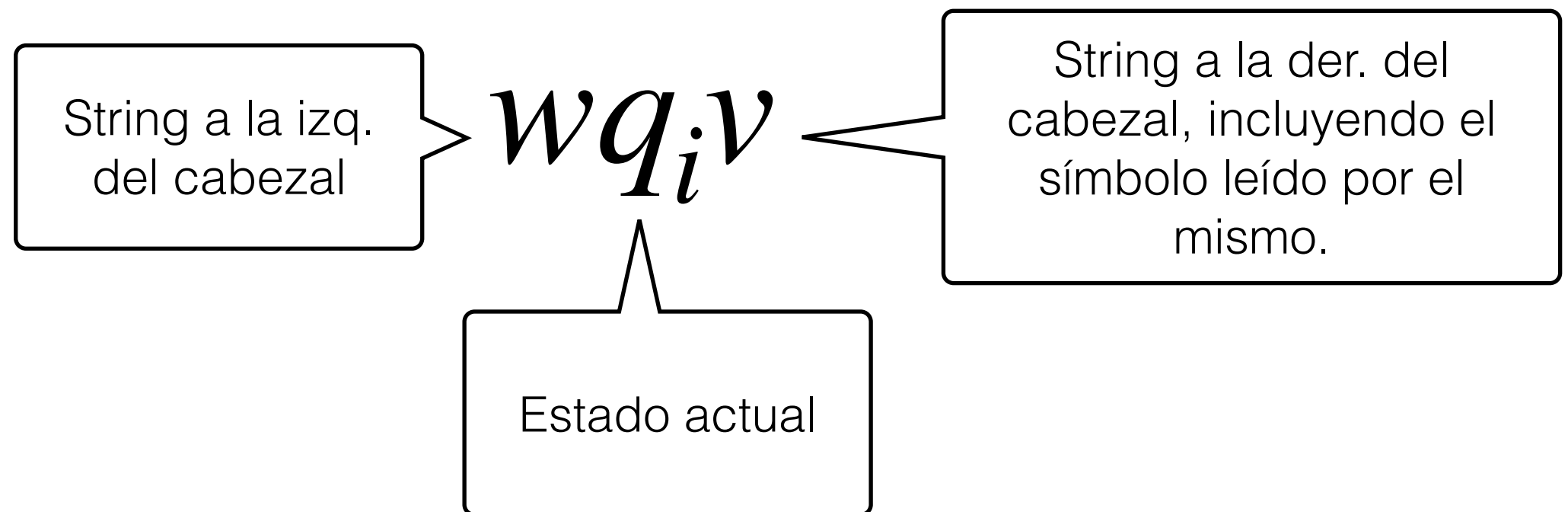
- *Se empieza con el estado inicial q_0 con el cabezal apuntando al primer carácter.*
- *Lee el carácter, y según δ esta lee, escribe, y se mueve a la izq. o der. y cambia su estado,*
- *Se repite el paso anterior hasta que se llega a q_A , se acepta la w , o q_R , se rechaza w .*

Una máquina de Turing puede: (i) aceptar una cadena, (ii) rechazarla al llegar al estado q_R , (iii) rechazarla al no terminar.

Configuraciones (Estados)

Podemos describir el funcionamiento de las máquinas de Turing usando las nociones de configuración/estados:

Una **configuración** tiene tres componentes:



Escribimos $wq_i v \rightarrow^M w'q_j v'$ si de la configuración $wq_i v$ pasamos a $w'q_j v'$ en la máquina M .

Aceptación

Una MT M acepta una cadena w si existe una secuencia de configuraciones C_0, C_1, \dots, C_n tal que:

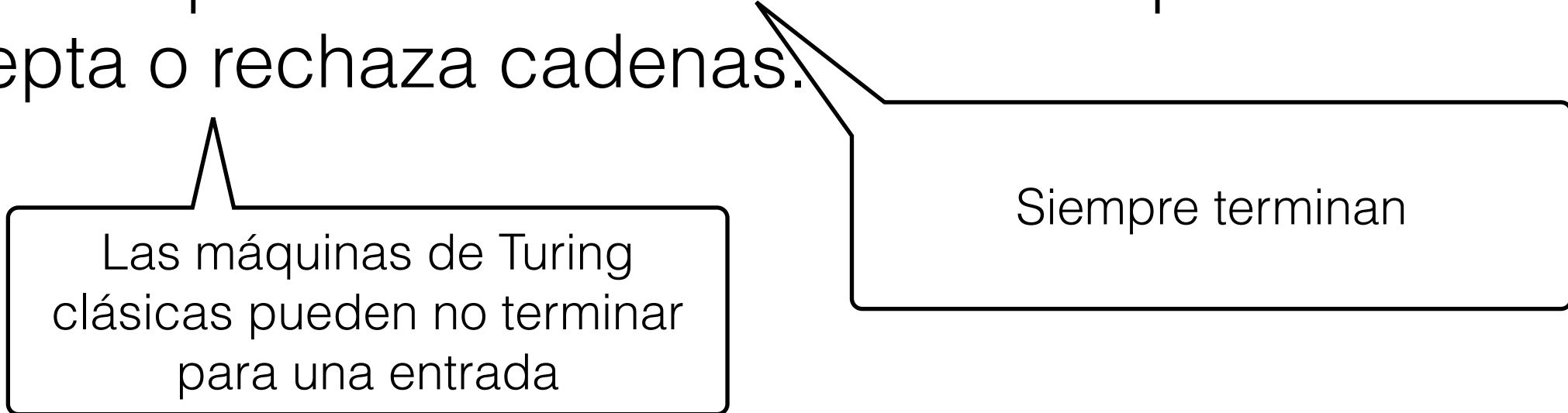
- $C_0 = q_{init}w$ es la configuración inicial.
- Para cada $1 \leq i \leq n : C_i \rightarrow C_{i+1}$
- C_n es una configuración de aceptación, es decir:
$$C_n = w'q_Aw''$$

Dada una máquina de Turing M podemos definir el lenguaje que esta reconoce: $\mathcal{L}(M) = \{w \mid M \text{ acepta } w\}$.

Un lenguaje L se dice **Turing reconocible** ssi $\mathcal{L}(M) = L$ para alguna máquina de Turing M .

Máquinas de Decisión

Una máquina de decisión es una maquina de Turing que solo acepta o rechaza cadenas.



Las máquinas de Turing clásicas pueden no terminar para una entrada

Siempre terminan

Un lenguaje se dice **decidible** ssi existe una máquina de decisión que reconoce el lenguaje.

Veremos que hay lenguajes que **no son decidibles**.

También lenguajes que son **Turing reconocibles pero no decidibles**.

Una máquina de decision permite reconocer si algo es una respuesta o no a un problema.

Ejemplo de Lenguaje Reconocible pero no decidable

Consideremos el siguiente lenguaje: $L = \{\phi \mid \vdash \phi\}$

Las formulas de primer orden que son válidas (teoremas).

Podemos pensar una MT que reconozca este lenguaje.

1. Se empieza con la formula codificada en la cinta,
2. Después de la secuencia inicial, se escribe un #
3. Utilizando las reglas y axiomas se empiezan a escribir las posibles demostraciones
4. Si aparece ϕ se acepta.

Si ϕ es demostrable, alguna vez aparecerá, pero sino la máquina se ejecuta para siempre. Se puede demostrar que este lenguaje no es decidable.

Describiendo Máquinas de Turing

Ejemplo: Consideremos el lenguaje: $A = \{w\#w \mid w \in \{0,1\}^*\}$, pensemos una MT para reconocer este lenguaje.

Dado un input w , M procede de la siguiente forma:

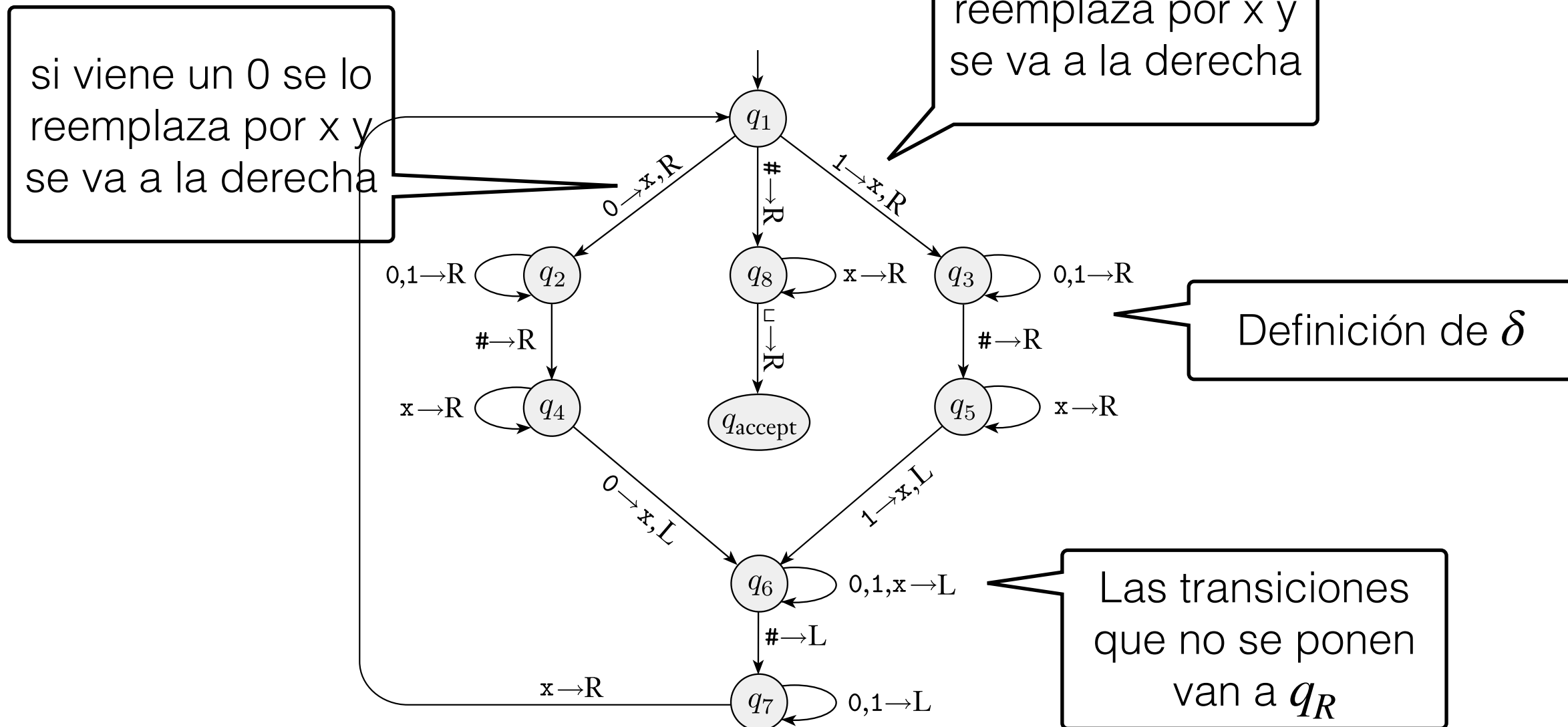
- Dada la configuración $wq_i b v$ reemplazamos b por x y avanzamos hasta que encontramos el primer símbolo después de $\#$ diferente de x . Si es igual que b lo reemplazamos por x , y volvemos al comienzo, sino rechazamos.
- En configuración $w\#q_j v$ nos fijamos que todos los símbolos son x , en ese caso aceptamos, sino rechazamos la cadena.

Definición Formal

Podemos definir M formalmente:

$$\Sigma = \{0,1,\#\} \quad \Gamma = \{0,1,\#,x,\sqcup\}$$

M_1 :



Otro Ejemplo

Veamos un ejemplo de una máquina de Turing que acepta el lenguaje: $A = \{0^{2^n} \mid n \geq 0\}$

- Reemplazar el primer 0 por \sqcup , si no hay más símbolos se acepta,
- Si hay un 0 más se lo reemplaza por x,
- Para todos los 0's a la derecha, reemplazar el de al lado (si es cero) por x
- Si se llega al fin y no hay más ceros y todos x, se acepta. En otro caso se repite el paso (3)

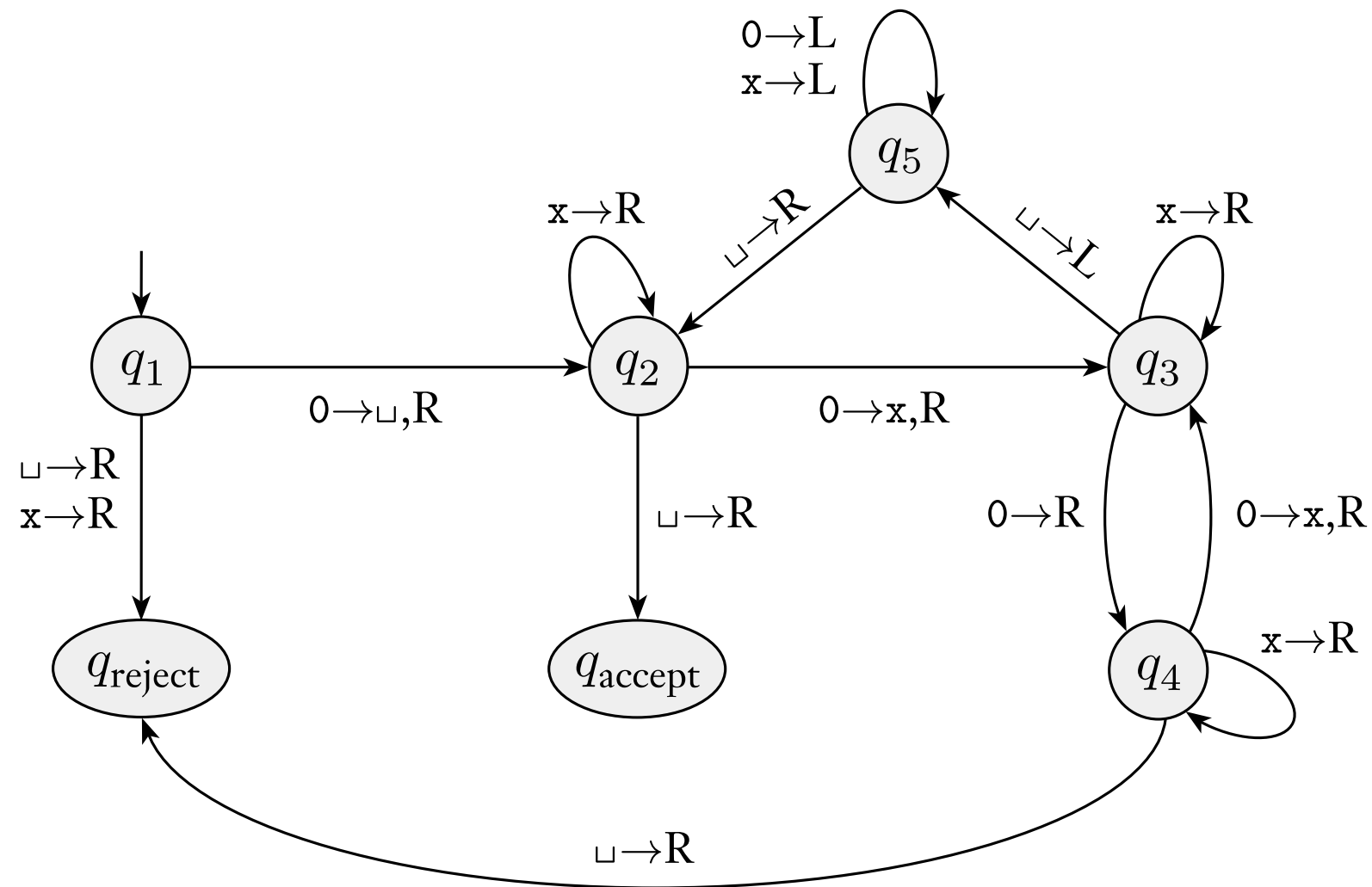
Definición Formal

$$Q = \{q_0, q_1, q_2, q_3, q_4\}$$

$$\Sigma = \{0\}$$

$$\Gamma = \{0, x, \sqcup\}$$

M_2 :



Definición Formal (cont.)

También podemos definir δ con una tabla:

	q_0			q_1			q_2			q_3			q_4			q_5		
	write	move	next	write	move	next	write	move	next	write	move	next	write	move	next	write	move	next
Simbo lo																		
0	□	R	q_2	x	R	q_3	0	R	q_3	0	R	q_4	x	R	q_4	0	L	q_5
□	□	R	q_R	□	R	q_A	□	R	q_A	□	L	q_5	□	L	q_R	□	R	q_2
x	□	R	q_R	x	R	q_2	x	R	q_2	x	R	q_3	x	R	q_4	x	L	q_5

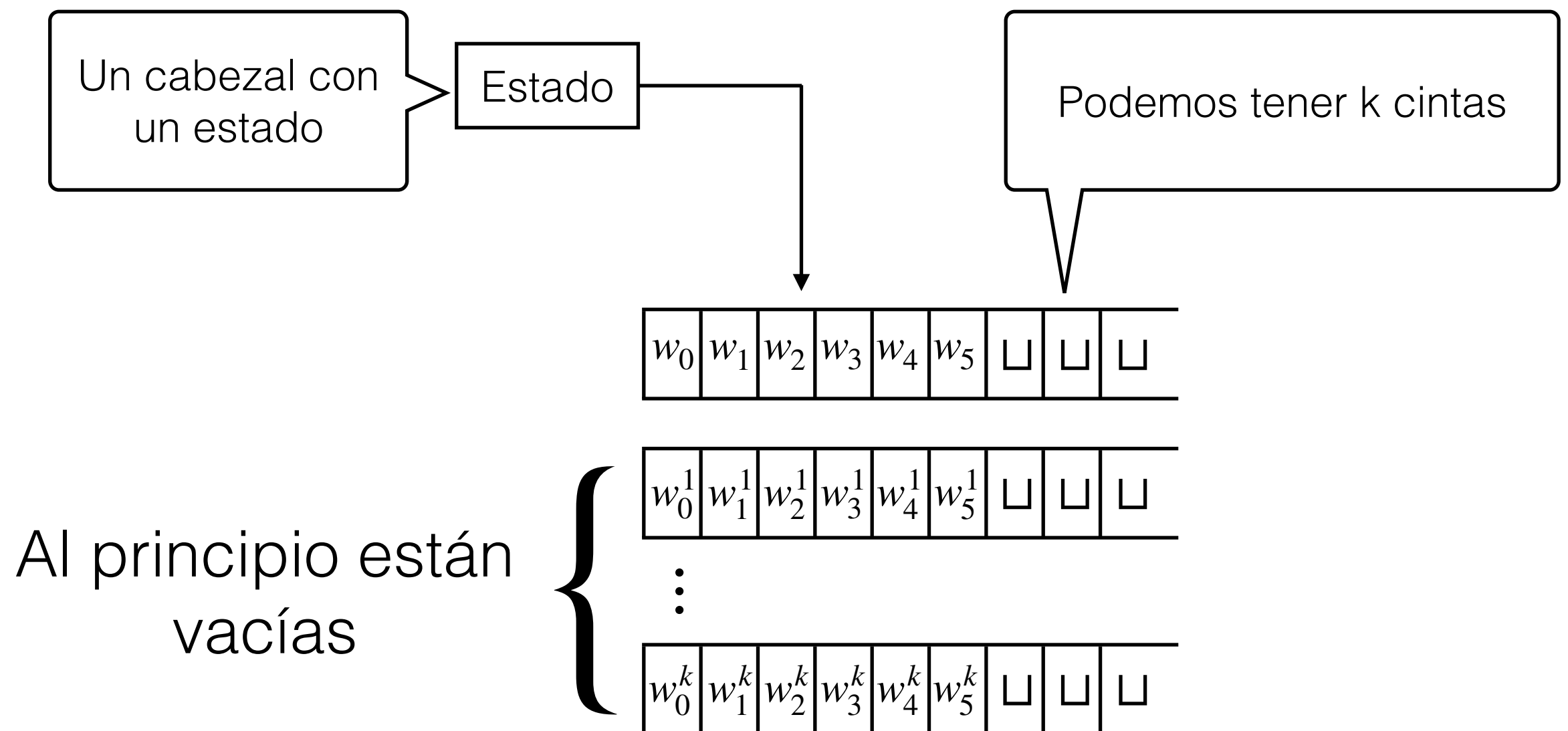
Para máquinas de Turing muy grandes es impracticables.

Ejercicios:

1. Implementar en Python las máquinas de Turing. Utilizar su implementación para programar los ejemplos dados, y testearlos.
2. Para la MT M_2 dar la secuencia de configuraciones cuando se empieza con:
 - I. 0
 - II. 00
 - III. 000
 - IV. 000000
3. Dar definiciones de máquinas de Turing para **decidir** los siguientes lenguajes:
 1. $\{w \mid w \text{ contiene un número igual de 0's y 1's}\}$
 2. $\{w \mid w \text{ contiene dos veces mas 0's que 1's}\}$
4. Una máquina de Turing con cinta doblemente infinita son máquinas de Turing en las cuales las cintas son infinitas tanto para la izquierda como para la derecha. Mostrar que estas máquinas reconocen los mismos lenguajes que las máquinas de Turing.

MTs con muchas cintas

Las máquinas de Turing con muchas cintas son una generalización de las MTs, que permiten procesar las cosas de forma más simple.



Definición Formal

Podemos definir formalmente las MT con muchas cintas:

$$\langle Q, \Sigma, \{\Gamma_i\}_{0 \leq i \leq k}, \delta, q_0, q_A, q_R \rangle$$

En donde:

- Γ_i es el alfabeto de la i-ésima cinta
- $\delta : Q \times \Gamma_0 \times \dots \times \Gamma_{k-1} \rightarrow Q \times \{L, R\}^k$ es la función de transición

Equivalencia

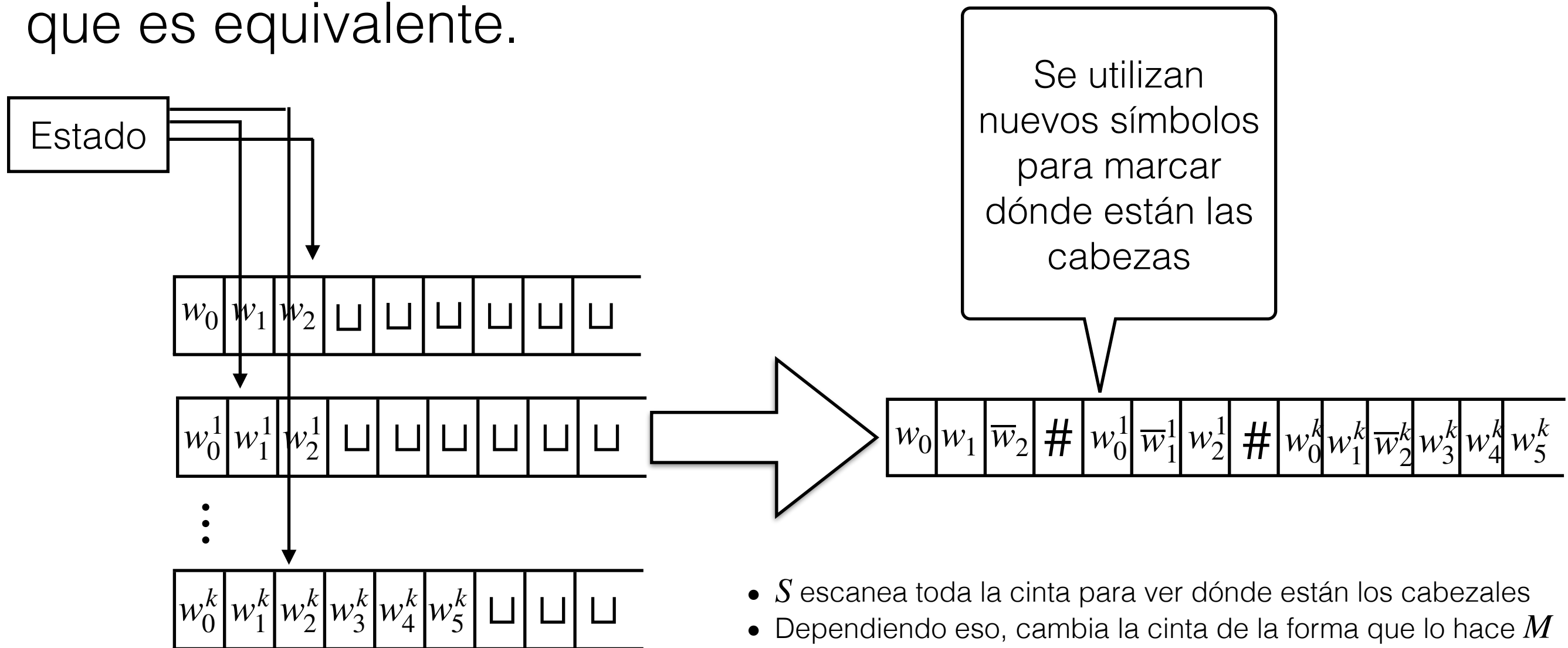
Decimos que dos máquinas de Turing M, M' son equivalentes cuando, para cualquier palabra w

- M acepta la palabra ssi M' acepta la palabra,
- M rechaza w ssi M' rechaza w
- M no termina ssi M' no termina

La equivalencia de máquinas de Turing es un problema no decidable

Equivalencia entre Modelos

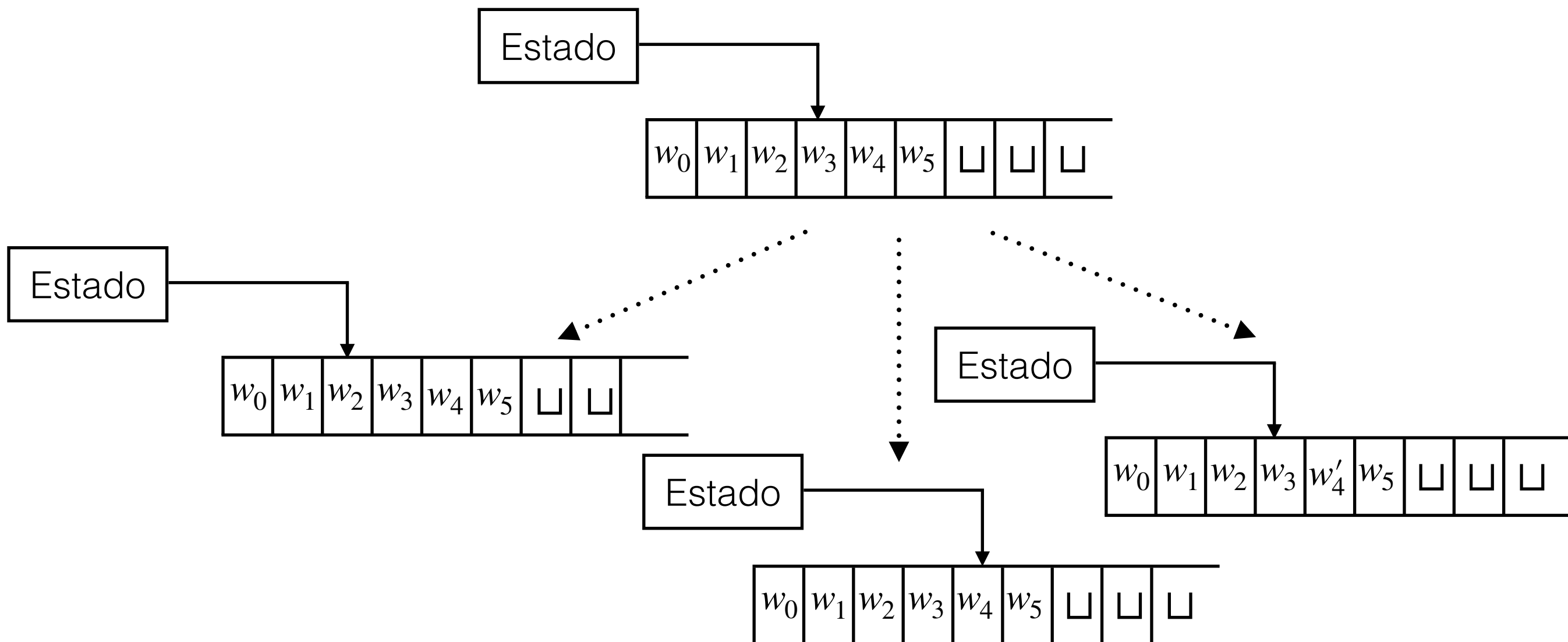
Teorema : Toda MT con k cintas tiene una MT con una cinta que es equivalente.



- S escanea toda la cinta para ver dónde están los cabezales
- Dependiendo eso, cambia la cinta de la forma que lo hace M
- Si es necesario tener más espacio se corren todos los símbolos para la derecha.

Máquinas de Turing No-Deterministas

Las máquinas de Turing no-deterministas permiten considerar ejecuciones no deterministas, es decir, en cada configuración tenemos muchos posibles sucesores



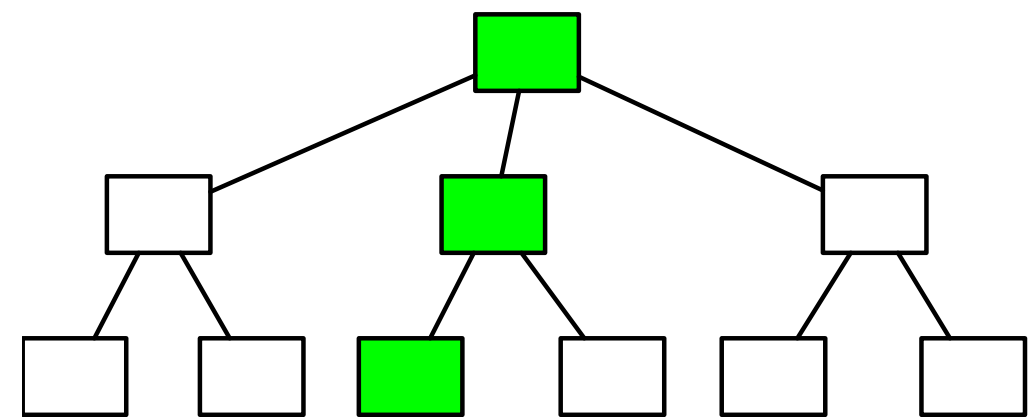
Definición Formal

En la definición formal lo único que cambia es la función de transición:

$$\delta : Q \times \Gamma \rightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\})$$

Notar que:

- Cada configuración tiene solo una cantidad finita de posibles sucesores.
- Se acepta una cadena cuando hay al menos una rama del árbol de ejecución que la acepta.
- No es un modelo realista de computación.



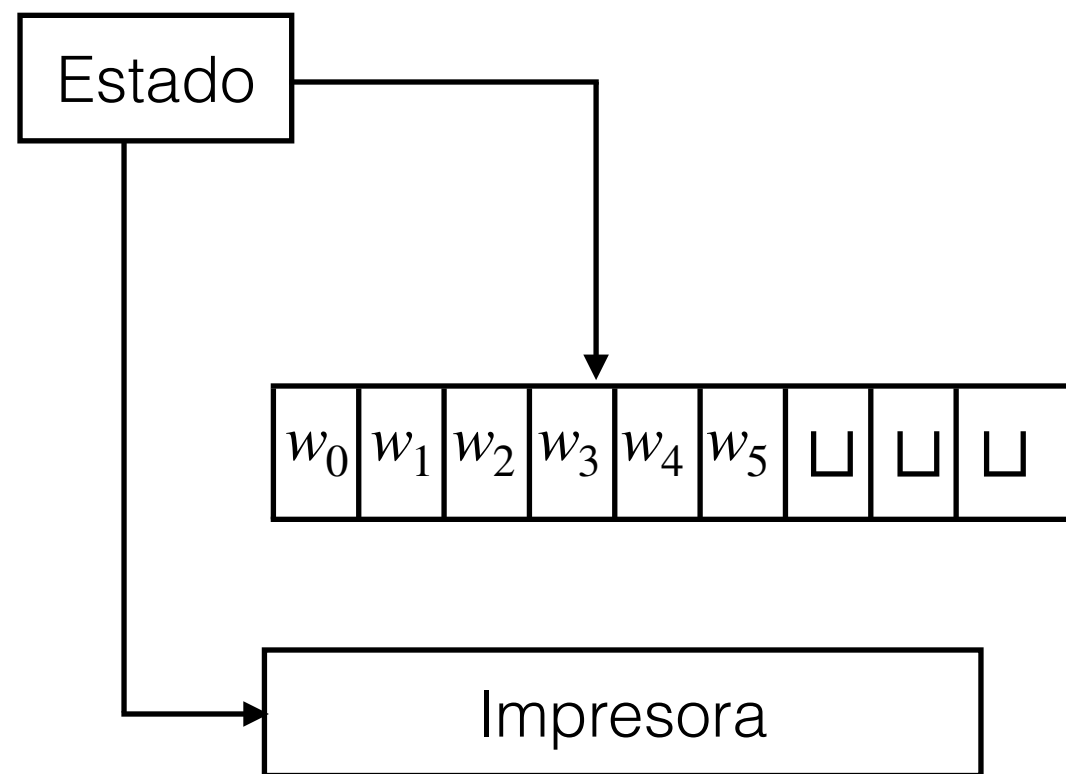
Las MT no-determinísticas son equivalentes a la MT

Teorema: Para cualquier máquina de Turing no-determinista existe una máquina de Turing determinista que es equivalente.

Prueba: La idea es que dada MT N , no determinística, construimos una máquina M determinística que acepta una cadena si y solo si N acepta la cadena. La idea es recorrer el árbol de ejecución de N con un BFS, si la cadena es aceptada por N en algún momento encontraremos la ejecución que la acepta. Si la N tiene una ejecución que no termina y no acepta la cadena entonces M no va a terminar.

Enumeradores

Los enumeradores son máquinas de Turing que, intuitivamente tienen una impresora.



- La impresora sirve para mostrar un output
- Formalmente es una cinta más
- Sirve para investigar los lenguajes que pueden ser computados por una computadora

Formalización

Un enumerador es una máquina de Turing con 2 cintas:

$$\langle Q, \Sigma, \Gamma_0, \Gamma_1, q_0, q_{init}, q_A, q_R, q_{print} \rangle$$

- Γ_0 es el alfabeto de la cinta,
- Γ_1 es el alfabeto de la impresora,
- q_{print} es un estado de impresión cuando se llega se imprime y se limpia la cinta de impresión.

Una lenguaje se dice que es (recursivamente) enumerado por un enumerador si existe un enumerador tal que imprime toda las palabras de lenguaje

Lenguajes Reconocibles y Enumerados

Teorema: Un lenguaje es Turing reconocible ssi existe un enumerador que lo enumera.

Prueba: Primero demostramos que si una enumerador E enumera un leng. A existe una MT M que la reconoce.

M con dos cintas actúa de la siguiente forma:

- Ejecutamos E , cada vez que imprimimos un string w , comparamos con w si son iguales se acepta,

Ahora dado una MT M definimos E como sigue:

Enumeradores (cont.)

El enumerador E se construye de la siguiente forma:

- Se ignora el input
- Se repite para $i=1,2,3,4,\dots$
 - Se corre M i pasos para cada input $s, s', s'', s''' \dots$
 - Si alguno se acepta se lo imprime

Cualquier cadena aceptada será impresa en algún momento.

Tesis de Church-Turing



Podemos pensar en diferentes formalizaciones de la noción de computación, todas ellas hasta ahora resultaron equivalentes. Esto se conoce como la tesis de Church-Turing

Tesis de Church-Turing: Todos los modelos de la noción de computación son equivalentes

Ejercicios:

1. Demostrar que la clase de lenguajes decidibles es cerrada bajo las operaciones de:
 - A. Union,
 - B. Concatenación,
 - C. Complemento,
 - D. Intersección,
 - E. iteración.
2. Demostrar que la clase de lenguajes reconocibles es cerrada bajo las operaciones de:
 - F. Union,
 - G. Concatenación,
 - H. Intersección,
 - I. Iteración.
3. Implementar maquinas de Turing No-Deterministas
4. Implementar los Eumeradores.