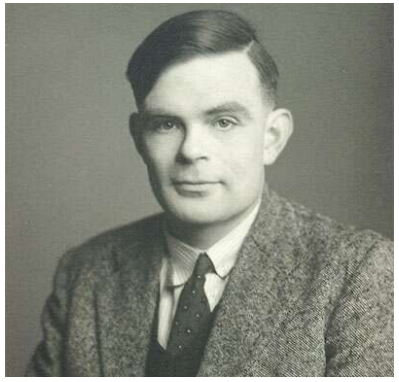
Computabilidad y Complejidad (CMC)

La máquina de Turing. Lenguajes recursivos. Lenguajes recursivamente enumerables. Funciones Turing-computables

Índice

Bibliografía básica recomendada

- Introducción a la teoría de autómatas, lenguajes y computación (Hopcroft, John E., Ullman, Jeffrey D.,Motwani, Rajeev)
- Elements of the theory of computation (Lewis, Harry R., Papadimitriou, Christos H.)



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

En respuesta al desafío de Hilbert, Alan Turing publica en 1936 un trabajo donde demuestra que hay problemas irresolubles mediante métodos algorítmicos

Con el objetivo de capturar la esencia de los procesos algorítmicos, Turing idealiza estos procesos proponiendo un mecanismo que captura los medios necesarios para la definición de cualquier algoritmo: la máquina de Turing

Fundamentalmente, en la máquina de Turing intervienen los tres elementos fundamentales necesarios para resolver cualquier cálculo de forma algorítmica: papel y lápiz (el medio físico para plasmar los progresos en la ejecución del algoritmo) y los estados mentales del calculador* durante la ejecución del algoritmo (que indican en cada momento qué acciones se deben tomar en función de la información disponible hasta ese momento).

*En 1936 el concepto de calculador(a) hacía referencia a la persona encargada de realizar cálculos

Generalidades. Modelo básico de máquina de Turing

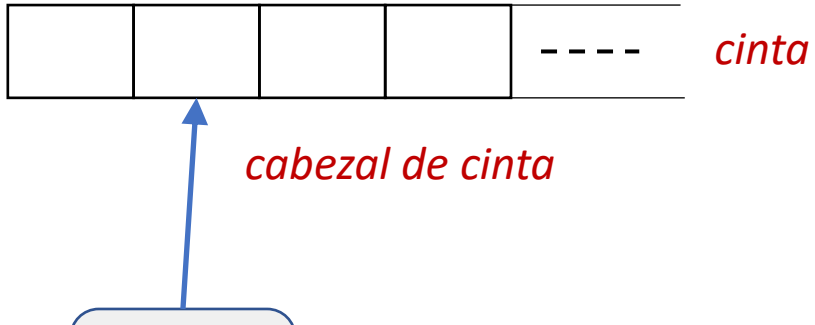
- La máquina de Turing es un modelo de computación universal estructural y operacionalmente muy sencillo que permite abordar el estudio de la Teoría de la Computabilidad, equivalentemente, tanto desde la Teoría de Lenguajes Formales como desde la Teoría de las Funciones Computables.

- Es capaz de realizar cualquier computación que se pueda hacer de modo efectivo por otros medios, y por tanto es capaz de reconocer cualquier lenguaje generado por cualquier gramática y de computar cualquier función computable definida en cualquiera de los sistemas formales de cálculo conocidos.

- En particular, el poder de computación del modelo gramatical es equivalente al del modelo de la máquina de Turing.

Generalidades. Modelo básico de máquina de Turing

El modelo básico de máquina de Turing (posteriormente estudiaremos otros) es un reconocedor de lenguajes, equivalentemente un computador de funciones, que se compone en esencia de los siguientes elementos:



Una **cinta de entrada** y de cálculo potencialmente infinita por la derecha y con un comienzo por la izquierda. Esta cinta se encuentra dividida en celdillas en las que necesariamente aparece escrito un símbolo del alfabeto de la cinta Γ; por defecto y como relleno se utiliza un símbolo denominado blanco y representado por B ($\in\Gamma$).

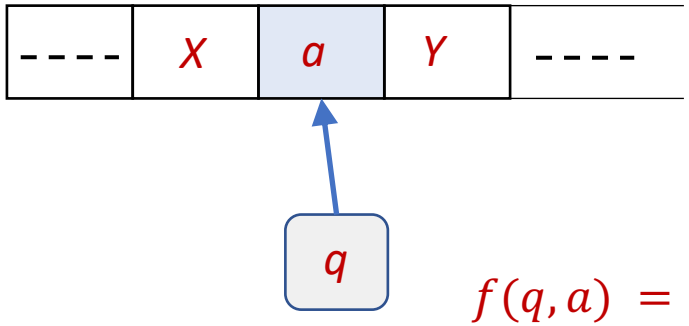
Un **cabezal de lectura/escritura**, que puede desplazarse por la cinta de celdilla en celdilla, una posición a la derecha (denotado por R) o a la izquierda (denotado por L), siempre leyendo el símbolo que hay escrito en cada una y escribiendo un símbolo a su vez (por tanto la lectura es siempre destructiva, si se requiere que el símbolo no se modifique hay que escribirlo de nuevo). En estos desplazamientos el cabezal no puede abandonar la cinta; así, si estando en la celdilla más a la izquierda de la cinta intentara un desplazamiento a la izquierda se abortaría la computación en curso.

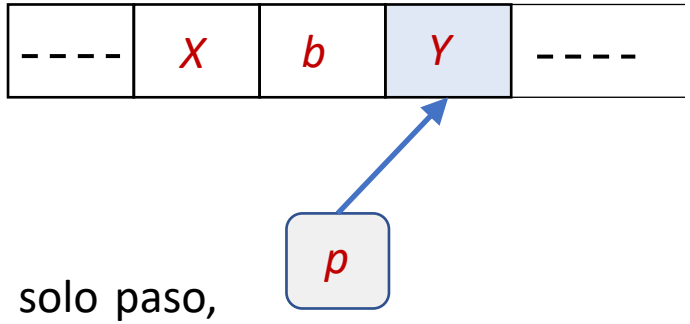
Un **control finito** que alberga un conjunto finito de estados, Q, y la función de transición, f, que determina, paso a paso, la evolución de la computación. En el conjunto de estados se encuentra un estado señalado como inicial (q_0) del cual arrancan todas las computaciones y un subconjunto denominado de estados finales o de aceptación ($F \subseteq Q$) que señalizan la aceptación de una palabra por parte de la máquina cuando uno de ellos es alcanzado.

Generalidades. Modelo básico de máquina de Turing

Además, la máquina tiene definido un alfabeto de entrada $\Sigma \subseteq \Gamma - \{B\}$, de modo que las palabras que constituyen las posibles entradas a la máquina pertenecen a Σ^*.

La función de transición es una función parcial definida como $f: Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$ de modo que, estando el cabezal de la máquina sobre una celdilla en la que se encuentra escrito el símbolo a y el control finito en el estado q, entonces:

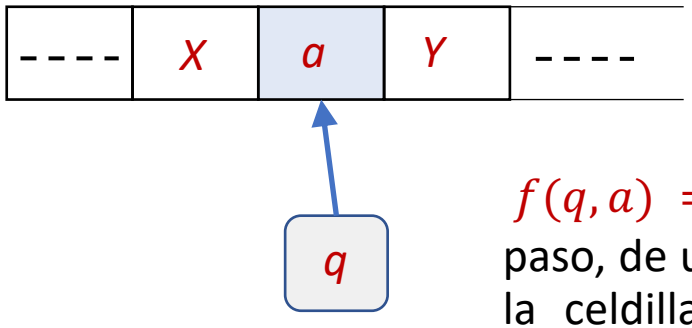


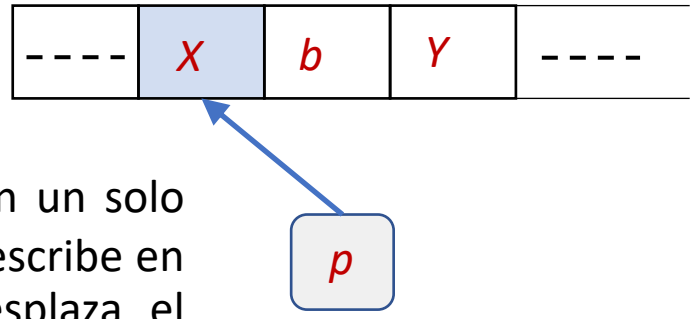


$f(q, a) = (p, b, R)$, significa que la máquina en un solo paso, de un modo atómico, transita al estado p, escribe en la celdilla en cuestión el símbolo b y se desplaza el cabezal a la celdilla contigua por la derecha.

Generalidades. Modelo básico de máquina de Turing

La función de transición es una función parcial definida como $f: Q \times \Gamma \longrightarrow Q \times \Gamma \times \{L, R\}$ de modo que, estando el cabezal de la máquina sobre una celdilla en la que se encuentra escrito el símbolo a y el control finito en el estado q, entonces:



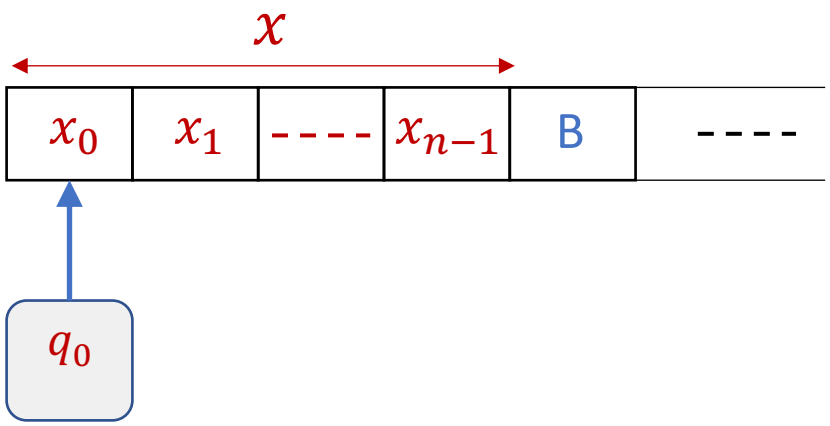


$f(q, a) = (p, b, L)$, significa que la máquina en un solo paso, de un modo atómico, transita al estado p, escribe en la celdilla en cuestión el símbolo b y se desplaza el cabezal a la celdilla contigua por la izquierda; siempre que la primera no sea la celdilla en la que comienza la cinta, en cuyo caso se aborta la computación sin realizarse la transición.

- Si $f(q, a)$ no está definida, entonces la máquina se detiene en la celdilla en cuestión, sin modificarse su símbolo y permaneciendo en el estado q.
- Un estado $q \in Q$ diremos que es de bloqueo si $f(q, a)$ no está definida para cada símbolo a de Γ.

Generalidades. Modelo básico de máquina de Turing

La máquina de Turing con una entrada $x_0x_1 \dots x_{n-1} = x \in \Sigma^*$ opera como sigue. La palabra x se escribe en la cinta de entrada, con cada símbolo en una celdilla, comenzando en la primera, ocupando por tanto $|x| = n$ celdillas y quedando el resto con el símbolo B. El cabezal de la cinta se sitúa en la primera celdilla leyendo su símbolo asociado, x_0, y el control finito en el estado inicial q_0. Realizándose, si es posible, como primer paso de la computación el indicado por la función de transición: $f(q_0, x_0)$, y continuándose de este modo.



En cada momento de una computación la posición del cabezal, el estado en curso y el contenido de la cinta condensa el resultado de los cálculos realizados.

Generalidades. Modelo básico de máquina de Turing

Formalmente, a partir de los elementos introducidos, se define una máquina de Turing M como

$$M = (\Sigma, \Gamma, Q, f, B, q_0, F)$$

- Supondremos, sin pérdida de generalidad, que $\Gamma \cap Q = \emptyset$

Dada una máquina de Turing $M = (\Sigma, \Gamma, Q, f, B, q_0, F)$, para cada $x \in \Sigma^*$,

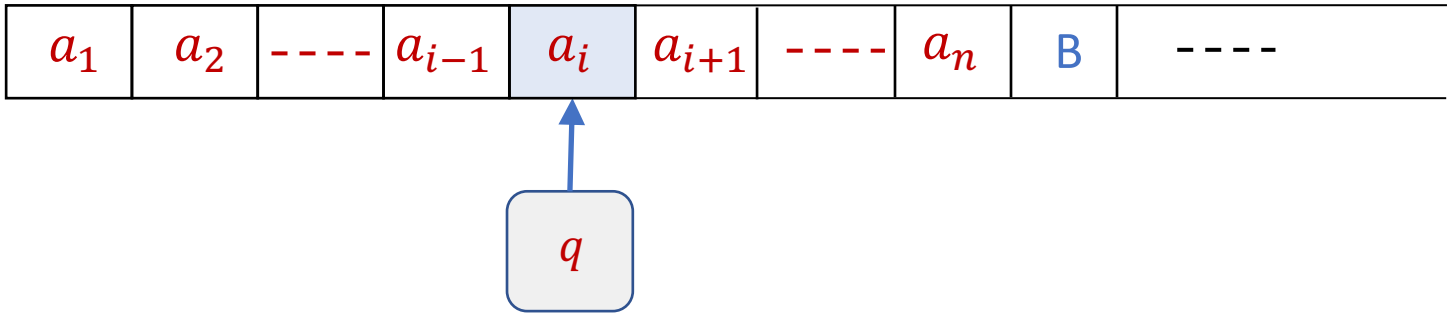
- mediante la notación $M(x)\downarrow$ denotaremos que la máquina M al procesar la palabra x termina la computación, deteniéndose indistintamente en un estado final o no.

- mediante la notación $M(x)\uparrow$ denotaremos que la máquina M al procesar la palabra x no se detiene.

Generalidades. Modelo básico de máquina de Turing

Descripciones instantáneas

Para estudiar las computaciones realizadas por las máquinas de Turing vamos a introducir el concepto de descripcion instantanea, que representa la configuración global en la que se encuentra la máquina en un instante dado de la computación.

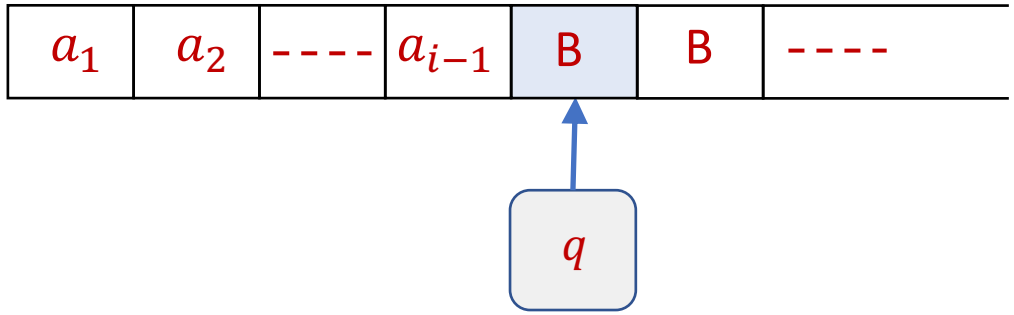


- Supóngase que la cinta tiene escrito desde la primera celdilla la siguiente secuencia de símbolos: $a_1 a_2 \dots a_{i-1} a_i a_{i+1} \dots a_n$ (siendo $a_n \neq B$, y estando el resto de las celdillas escritas con B).
- Supóngase, además, que la máquina se encuentra en el estado q y que el cabezal se encuentra sobre la celdilla del símbolo a_i.
- En estas condiciones la descripción instantánea asociada que representa esta configuración es

$$a_1 a_2 \dots a_{i-1} q a_i a_{i+1} \dots a_n$$

Generalidades. Modelo básico de máquina de Turing

Descripciones instantáneas

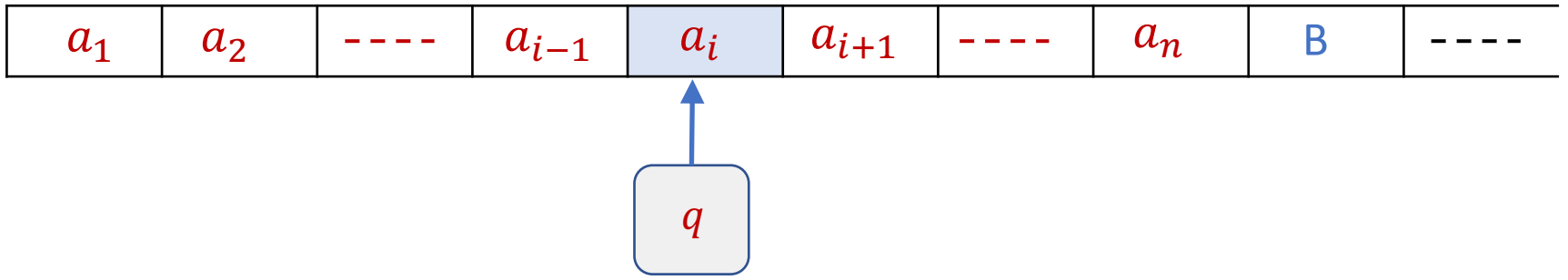


si $a_j = B$, $j = i, \ldots, n$, y el cabezal se encuentra sobre la celdilla del símbolo a_i, entonces la descripción instantánea asociada es

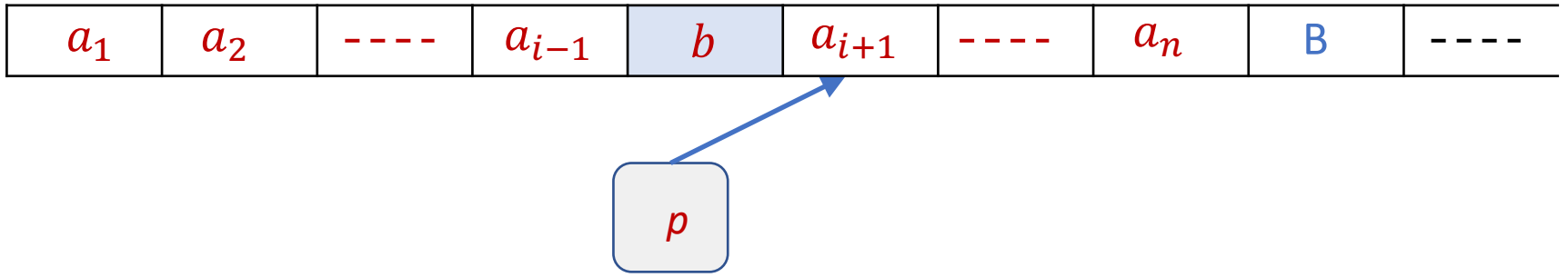
$$a_1 a_2 \ldots a_{i-1} q$$

Generalidades. Modelo básico de máquina de Turing

Descripciones instantáneas

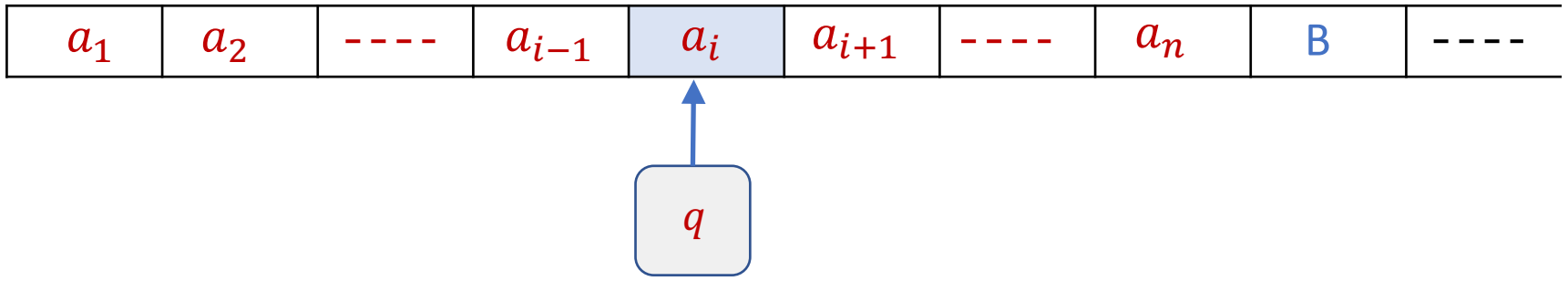


si $f(q, a_i) = (p, b, R)$, de la descripción instantánea $a_1 a_2 \dots a_{i-1} q a_i a_{i+1} \dots a_n$ se transita a la descripción instantánea $a_1 a_2 \dots a_{i-1} b p a_{i+1} \dots a_n$

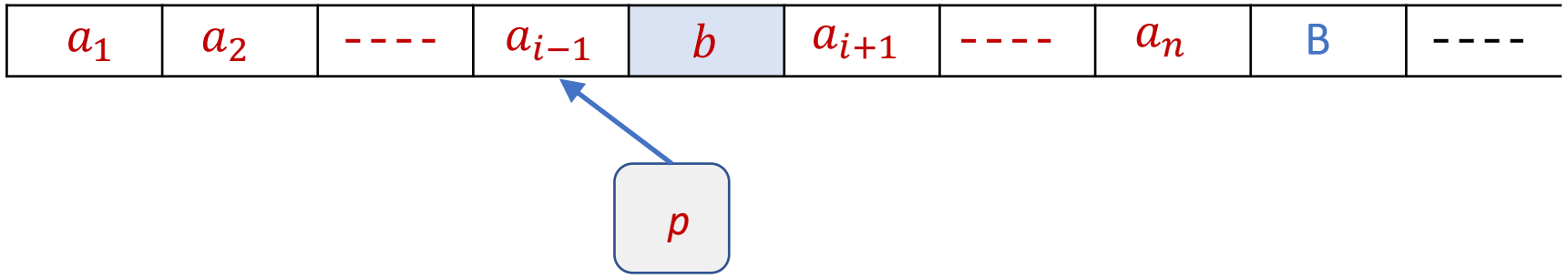


Generalidades. Modelo básico de máquina de Turing

Descripciones instantáneas

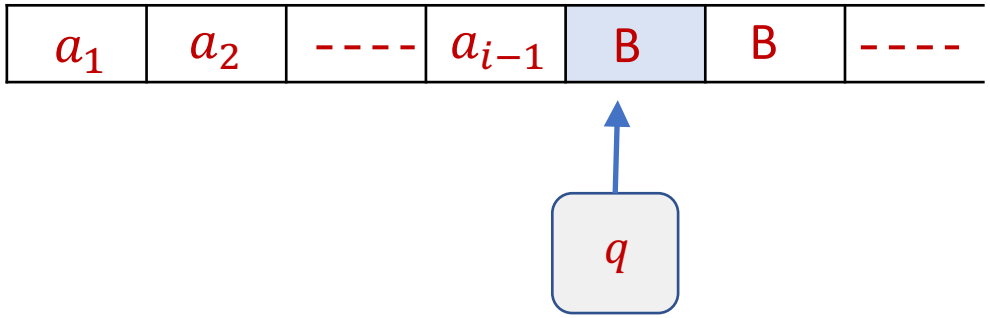


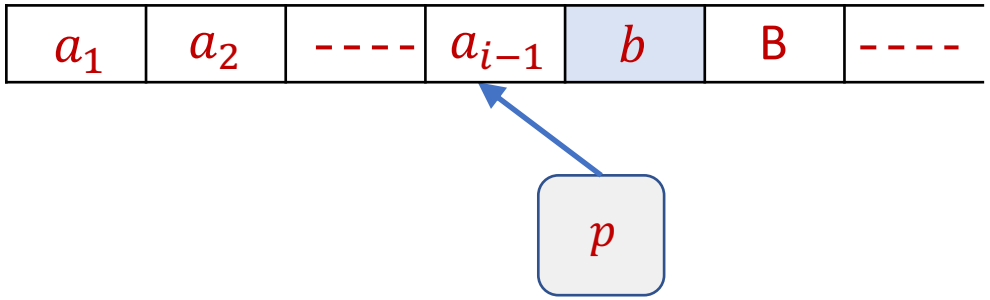
si $f(q, a_i) = (p, b, L)$, y siempre que $i > 1$, de la descripción instantánea $a_1 a_2 \dots a_{i-1} q a_i a_{i+1} \dots a_n$ se transita a la descripción instantánea $a_1 a_2 \dots p a_{i-1} b a_{i+1} \dots a_n$



Generalidades. Modelo básico de máquina de Turing

Descripciones instantáneas



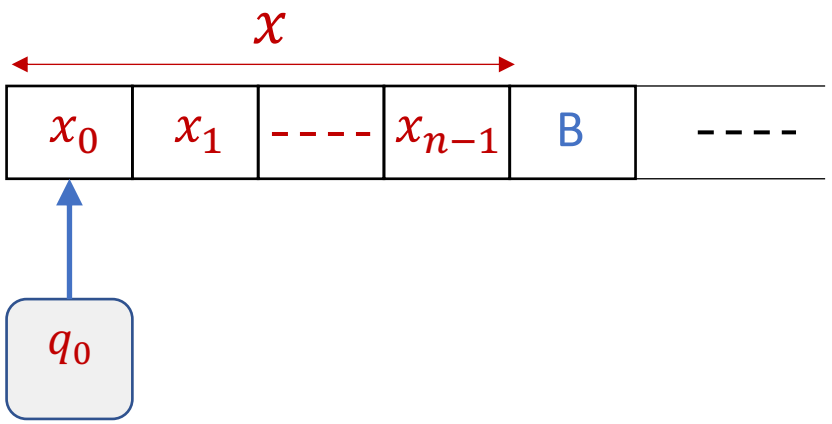


si $f(q, B) = (p, b, L)$, de la descripción instantánea $a_1 a_2 \dots a_{i-1} q$ se transita a

- la descripción instantánea $a_1 a_2 \dots p a_{i-1} b$ si $b \neq B$
- la descripción instantánea $a_1 a_2 \dots p a_{i-1}$ si $b = B$ y $a_{i-1} \neq B$
- la descripción instantánea $a_1 a_2 \dots a_{i-2} p$ si $b = B$ y $a_{i-1} = B$

Generalidades. Modelo básico de máquina de Turing

Descripciones instantáneas



La descripción instantánea inicial de la que parte una máquina con entrada x y estado inicial q_0 es q_0x

Si la entrada x es igual a la cadena vacía λ entonces las descripción instantánea inicial es q_0

Todas las descripciones instantáneas son palabras del lenguaje $\Gamma^*Q\Gamma^*$

Generalidades. Modelo básico de máquina de Turing

Relación de alcanzabilidad directa (o en un solo paso) entre descripciones instantáneas y sus derivadas

- La relación binaria de alcanzabilidad directa entre descripciones instantáneas en una máquina de Turing $M = (\Sigma, \Gamma, Q, f, B, q_0, F)$ se denota por $\vdash_M \subseteq \Gamma^* Q \Gamma^* \times \Gamma^* Q \Gamma^*$

- $I_1 \vdash_M I_2$ indica que de una descripción instantánea I_1 se puede alcanzar otra I_2 mediante la aplicación de una transición especificada por f en la máquina M.

Ejemplo: Si $f(q, a_i) = (p, b, R)$ entonces se produce la siguiente relación

$$a_1 a_2 \dots a_{i-1} q a_i a_{i+1} \dots a_n \vdash_M a_1 a_2 \dots a_{i-1} b p a_{i+1} \dots a_n$$

siempre que no exista posibilidad de confusión escribiremos $\vdash$ en lugar de $\vdash_M$

Generalidades. Modelo básico de máquina de Turing

Lenguaje reconocido por una máquina de Turing

Definimos el lenguaje reconocido por una máquina de Turing $M = (\Sigma, \Gamma, Q, f, B, q_0, F)$, denotado mediante $L(M)$, como

$$L(M) = \{x \in \Sigma^* : q_o x \vdash^* \alpha p \beta,\ p \in F,\ \alpha\beta \in \Gamma^*\}$$

dos máquinas de Turing M y M' son equivalentes si y sólo si L(M) = L(M').

- Según esta definición la aceptación de una palabra x no presupone que la palabra x se tenga que leer completamente ni que la máquina M se detenga.
- Supondremos, sin pérdida de generalidad, que todos los estados de F son de bloqueo. De este modo, a partir de ahora, la aceptación de una palabra implica la detención inmediata de la computación.
- Dada una máquina de Turing arbitraria existe una máquina de Turing equivalente con a lo sumo un estado final.
- En general, dada una máquina de Turing M, existe un número infinito de máquinas de Turing equivalentes a ella.

Generalidades. Modelo básico de máquina de Turing

Lenguaje reconocido por una máquina de Turing

$$L(M) = \{x \in \Sigma^* : q_o x \vdash^* \alpha p \beta,\ p \in F,\ \alpha\beta \in \Gamma^*\}$$

Nótese que si $q_0 \in F$, entonces la máquina comienza bloqueada y $L(M) = \Sigma^*$, siendo M equivalente a

$$M' = (\Sigma, \Sigma \cup \{B\}, \{q_0\}, f', B, q_0, \{q_0\})$$

con la función de transición f': $\{q_0\}\times(\Sigma \cup \{B\}) \rightarrow \{q_0\}\times(\Sigma \cup \{B\})\times\{R, L\}$ totalmente indefinida.

Por esto, sin pérdida de generalidad y mientras no se diga lo contrario, supondremos que $q_0 \notin F$.

Clases de lenguajes aceptados por la máquina de Turing

Lenguajes recursivamente enumerables

La clase de los lenguajes reconocibles mediante máquinas de Turing se denomina la clase de los lenguajes recursivamente enumerables. La denotaremos por $\mathcal{L}_{REN}$.

Los lenguajes recursivamente enumerables coinciden con los lenguajes generados por las gramáticas de tipo 0 en la Jerarquía de Chomsky (que a su vez constituye la clase de lenguajes $\mathcal{L}_0$).

Si consideramos un lenguaje $L \subseteq \Sigma^*$ de forma que $L = L(M)$ siendo $M = (\Sigma, \Gamma, Q, f, B, q_0, F)$ una máquina de Turing que, para cada cadena $x \in \Sigma^*$ la máquina se comporta de una de las tres siguientes maneras:

- La máquina M, al procesar x, se detiene después de un número finito de transiciones (lo denotaremos por $M(x)\downarrow$) en un estado final o de aceptación. Diremos que M acepta x y $x \in L(M)$.

- La máquina M, al procesar x, se detiene después de un número finito de transiciones, es decir $M(x)\downarrow$, en un estado que no es final. Diremos que M rechaza x y $x \notin L(M)$.

- La máquina M, al procesar x, nunca se detiene, es decir $M(x)\uparrow$. En cualquiera de sus configuraciones instantáneas, M siempre podrá aplicar una nueva transición. En este caso $x \notin L(M)$.

Clases de lenguajes aceptados por la máquina de Turing

Lenguajes recursivamente enumerables

- La posibilidad de que una máquina de Turing, al procesar una cadena de entrada, pueda no detenerse nunca, introduce una situación de incertidumbre indeseable respecto a un posible resultado final y, por tanto, con respecto a la computación en su totalidad.

- En esa situación la máquina está definiendo un procedimiento efectivo pero no un algoritmo, ya que no se garantiza una finalización del proceso.

En los años 50 del siglo XX el profesor Martin Davis, en una serie de conferencias y publicaciones empezó a referirse al Problema de la Parada (*The Halting Problem*) que se puede formular de la siguiente forma: "*Dada una máquina de Turing M y una cadena de entrada para la máquina x, ambas tomadas arbitrariamente, establezca si M se detendrá al procesar la cadena x.*"

El problema de la parada es irresoluble mediante técnicas algorítmicas (asumiendo la hipótesis de Church-Turing)

Clases de lenguajes aceptados por la máquina de Turing

Lenguajes recursivos

Si consideramos el conjunto de lenguajes reconocibles por máquinas de Turing que siempre se detienen (sea cual sea su cadena de entrada), restringimos la familia de los lenguajes recursivamente enumerables y damos lugar a una nueva clase de lenguajes que es la clase de lenguajes recursivos que denotaremos por $\mathcal{L}_R$.

Un lenguaje $L \subseteq \Sigma^*$ pertenece a $\mathcal{L}_R$ si y sólo si existe una máquina de Turing $M = (\Sigma, \Gamma, Q, f, B, q_0, F)$ de forma que $L(M) = L$ operando del siguiente modo ante cualquier cadena de entrada x

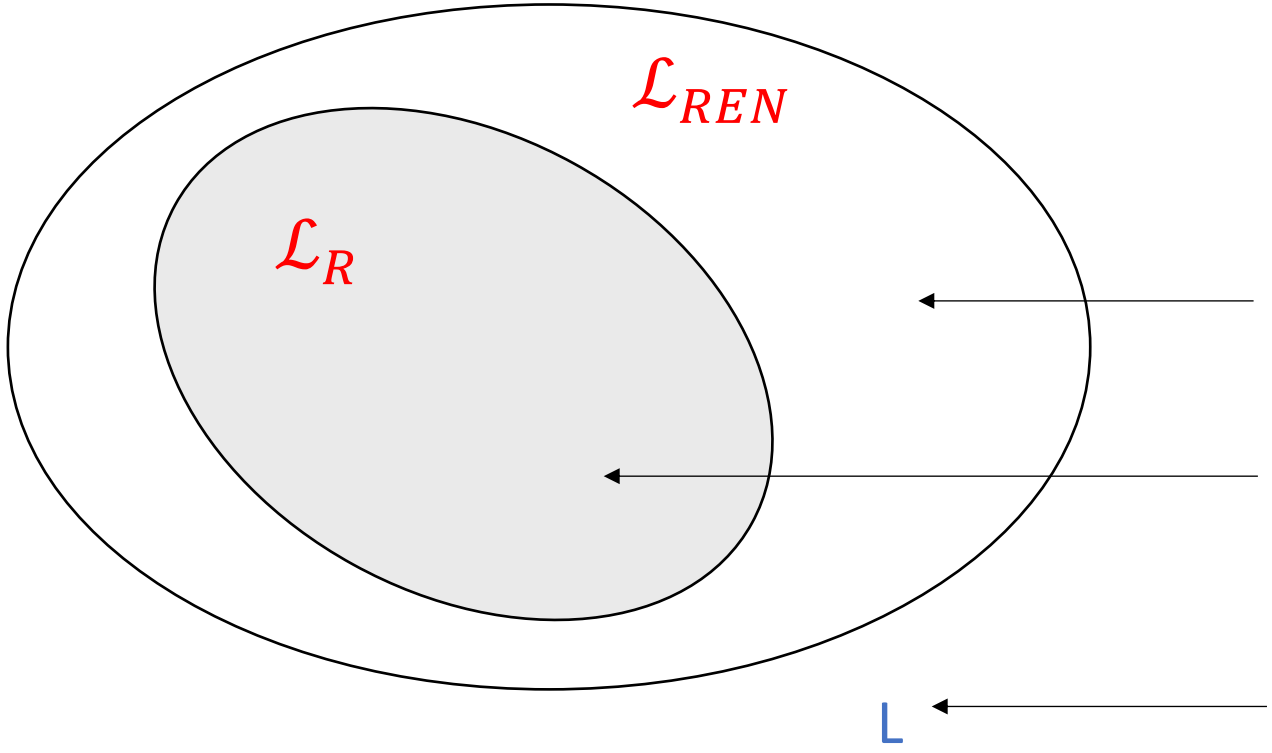
- La máquina M, al procesar x, se detiene después de un número finito de transiciones (lo denotaremos por $M(x)\downarrow$) en un estado final o de aceptación. Diremos que M acepta x y $x \in L(M)$.

- La máquina M, al procesar x, se detiene después de un número finito de transiciones, es decir $M(x)\downarrow$, en un estado que no es final. Diremos que M rechaza x y $x \notin L(M)$.

La clase de los lenguajes reconocibles mediante máquinas de Turing que siempre se detienen se denomina la clase de los lenguajes recursivos. La denotaremos por $\mathcal{L}_R$.

Clases de lenguajes aceptados por la máquina de Turing

Relación entre los lenguajes recursivamente enumerables y lenguajes recursivos



$$\mathcal{L}_R \subset \mathcal{L}_{REN} = \mathcal{L}_0$$

Existen lenguajes recursivamente enumerables que no son recursivos

Todos los lenguajes recursivos son lenguajes recursivamente enumerables

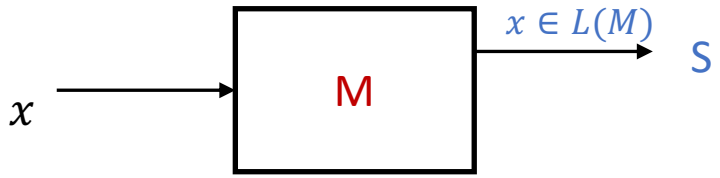
Existen lenguajes que no son recursivamente enumerables (no pueden ser aceptados por máquinas de Turing ni generados por gramáticas formales)

Clases de lenguajes aceptados por la máquina de Turing

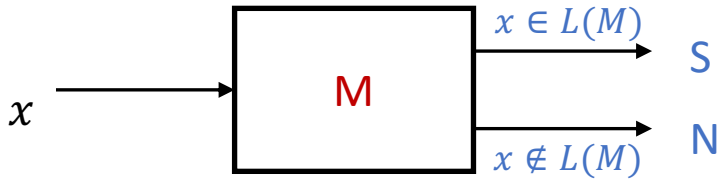
Representación mediante esquemas

Cuando queramos representar una máquina de Turing podremos utilizar esquemas que representen su funcionamiento.

Consideremos una máquina M y su lenguaje asociado L(M)



A la máquina M se le proporciona la cadena x como entrada y si $x \in L(M)$ entonces M para aceptando. Si $x \notin L(M)$ entonces M puede parar rechazando o no parar. Este esquema es válido para cualquier lenguaje L(M) recursivamente enumerable



A la máquina M se le proporciona la cadena x como entrada y si $x \in L(M)$ entonces M para aceptando. Si $x \notin L(M)$ entonces M para rechazando. Este esquema es válido sólo cuando L(M) es recursivo.

Funciones computables mediante máquinas de Turing

La máquina de Turing además de como un reconocedor de lenguajes puede utilizarse como un computador de funciones permitiendo desarrollar la teoría de las Funciones Computables.

Sin pérdida de generalidad, para funciones numéricas, únicamente consideraremos funciones (parciales) de la forma

$$g: \mathbb{N}^m \longrightarrow \mathbb{N}^k$$

esto es funciones, para la m-tupla de números naturales, $n_1, n_2, \dots, n_m$, definidas de la forma

$$g(n_1, n_2, \dots, n_m) = (i_1, i_2, \dots, i_k)$$

donde $i, i_2, \dots, i_k$ son números naturales.

En la teoría de funciones computables es relativamente sencillo pasar de dominios y rangos de números naturales a dominios y rangos de números enteros (positivos y negativos). También lo es para pasar a trabajar sobre números racionales. Sin embargo, las computación con números reales $\mathbb{R}$ es una materia de investigación avanzada debido a la precisión infinita que se necesita de forma intrínseca y las implicaciones que tiene la precisión en diversos aspectos de cálculo algorítmico.

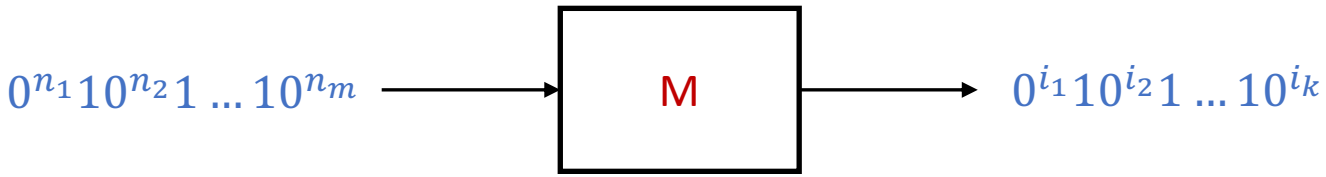
Funciones computables mediante máquinas de Turing

Dado que la máquina va a funcionar tomando como entrada valores naturales y proporcionando como salida valores naturales. Un primer aspecto a considerar es cómo se van a representar esos valores en la cinta de la máquina. Para ello se proporciona una posible codificación mediante un alfabeto binario:

- El número natural n se representa como la cadena 0^n
- La tupla de valores naturales $(n_1, n_2, \dots, n_m)$ se representa como la cadena $0^{n_1}10^{n_2}1 \dots 10^{n_m}$

$M = (\{0,1\}, \Gamma, Q, f, B, q_0, \emptyset)$

(obsérvese que en la máquina no se definen estados finales ya que no se aceptan/rechazan cadenas)



$g(n_1, n_2, \dots, n_m) = (i_1, i_2, \dots, i_k)$

Funciones computables mediante máquinas de Turing

$M = (\{0,1\}, \Gamma, Q, f, B, q_0, \emptyset)$ $\quad g(n_1, n_2, \dots, n_m) = (i_1, i_2, \dots, i_k)$

La máquina comienza la computación con la descripción instantánea $q_0 0^{n_1} 1 0^{n_2} 1 \dots 1 0^{n_k}$ y se detiene en la descripción instantánea $\alpha p 0^{i_1} 1 0^{i_2} 1 \dots 1 0^{i_k} \mathrm{B} \beta$ con $p \in Q$ (alternativamente podría definirse la descripción instantánea última como $\mathrm{B} \dots \mathrm{B} \alpha p \beta$ con $\alpha\beta = 0^{i_1} 1 0^{i_2} 1 \dots 1 0^{i_k}$, es decir, el contenido de la cinta se limita a los valores de la función escritos de forma compacta siendo el resto celdillas en blanco).

En cualquier otro caso diremos que la función computada está indefinida para los argumentos de entrada $n_1, n_2, \dots, n_m$

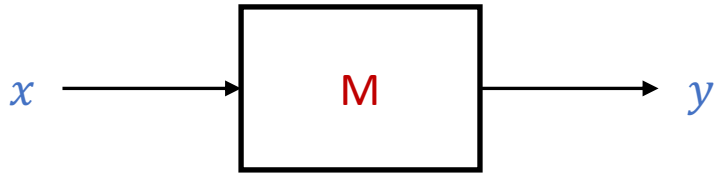
A las funciones computables de esta forma mediante máquinas de Turing las denominaremos funciones numéricas Turing-computables

Funciones computables mediante máquinas de Turing

La máquina puede también calcular funciones alfabéticas,

$$g: \Sigma^* \longrightarrow \Delta^*$$

siendo Σ y Δ dos alfabetos (que pueden ser el mismo) que no contienen el símbolo blanco.



$$g(x) = y$$

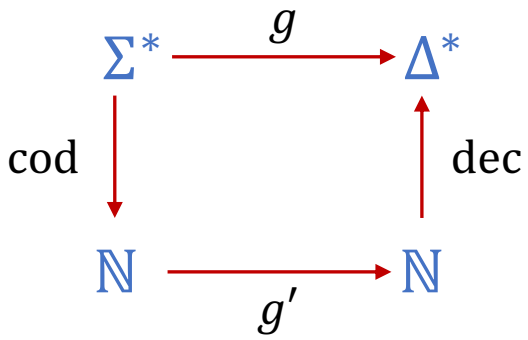
En este caso, la máquina M calcula la función g. También podemos decir que la máquina M transduce la cadena x en la cadena y de acuerdo con la transducción g. A las funciones computables de esta forma mediante máquinas de Turing las denominaremos funciones alfabéticas Turing-computables

Funciones computables mediante máquinas de Turing

Relación entre las funciones numéricas y las funciones alfabéticas

- Toda función numérica puede verse como un caso de función alfabética debido a la codificación sobre el alfabeto $\{0,1\}$ impuesta sobre las tuplas de valores naturales.
- Recíprocamente toda función alfabética Turing-computable puede computarse por medio de una función numérica y una función de codificación y otra de decodificación, todas ellas Turing-computables.

De modo que el siguiente diagrama conmuta



- Podemos considerar cod como un esquema de codificación efectivo que transforma las cadenas en valores naturales
- Podemos considerar dec como un esquema de decodificación efectivo que transforma los valores naturales en cadenas.

En lo referente a cuestiones computables, podemos, sin pérdida de generalidad, ceñirnos a las funciones computables numéricas de la forma $\mathbb{N} \longrightarrow \mathbb{N}$ o, equivalentemente, de la forma $\{0^*\} \longrightarrow \{0^*\}$

Funciones computables mediante máquinas de Turing

Funciones características

A cada lenguaje $L \subseteq \Sigma^*$ pueden asociársele funciones (parciales) de la forma

$$\eta_{L,\Sigma}: \Sigma^* \longrightarrow \{0\}^*$$

de modo que $L = \{x \in \Sigma^*: \eta_{L,\Sigma}(x) = \lambda\}$

Proposición: Sea el lenguaje $L \subseteq \Sigma^*$. Entonces $L \in \mathcal{L}_{REN}$ si y sólo si existe una función $\eta_{L,\Sigma}$ Turing-computable

Funciones computables mediante máquinas de Turing

Funciones características

A cada lenguaje $L \subseteq \Sigma^*$ pueden asociársele funciones totales de la forma

$$\varphi_{L,\Sigma}: \Sigma^* \longrightarrow \{0\}^*$$

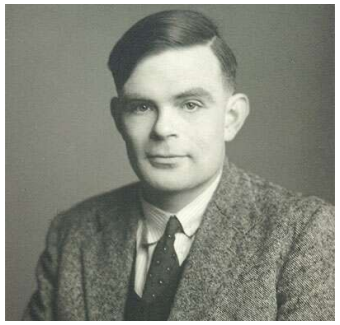
de modo que $\forall x \in \Sigma^*$

- $\varphi_{L,\Sigma}(x) = \lambda$, si $x \in L$
- $\varphi_{L,\Sigma}(x) \in \{0\}^+$, si $x \notin L$

Proposición: Sea el lenguaje $L \subseteq \Sigma^*$. Entonces $L \in \mathcal{L}_R$ si y sólo si existe una función $\varphi_{L,\Sigma}$ Turing-computable

Proposición: Sea el lenguaje $L \subseteq \Sigma^*$. Si $L \in \mathcal{L}_{REN} - \mathcal{L}_R$, entonces cada función $\varphi_{L,\Sigma}$ no es Turing-computable

La hipótesis de Church



- La hipótesis de Church, también conocida como la tesis de Church-Turing, enuncia que toda función computable es Turing-computable, esto es, identifica la noción intuitiva de computabilidad con la de Turing-computabilidad.
- La hipótesis se ha demostrado para todos los sistemas formales de computación conocidos.
- Se mantiene como hipótesis debido al componente informal que tiene la noción de computabilidad hasta que ésta se formaliza en un modelo específico de computación.
- Existen modelos de computación que superan la barrera de Turing y que habilitan la hipercomputación ó super-computación. Estos modelos no son razonables desde un punto de vista clásico (i.e. máquinas de Turing con oráculos, máquinas de números reales con precisión infinita, ...)

Técnicas de construcción de máquinas de Turing

Las técnicas para la construcción de máquinas de Turing que a continuación se exponen son las que utilizaremos en nuestro desarrollo y consisten básicamente en unos procedimientos y operaciones que, a modo de herramientas de alto nivel, facilitan la definición de las máquinas de Turing complejas sin alterar su modelo.

Por ejemplo

- Parametrización de estados
- Cintas con varias bandas o sectores
- Desplazamientos en el contenido de la cinta
- Definición de subrutinas
- ...

Técnicas de construcción de máquinas de Turing

Parametrización de estados

Esta técnica consiste en representar los estados mediante n-tuplas de n-parámetros, para un n dado, de modo que cada estado adopta la representación

$$[p_1, p_2, \dots, p_n], p_i \in \mathcal{P}_i$$

donde $\mathcal{P}_i,\ i = 1,2, \dots, n$, es el conjunto finito de valores que puede adoptar el parámetro p_i.

Así, en la máquina de Turing $M = (\Sigma, \Gamma, Q, f, B, q_0, F)$, $Q \subseteq \{[p_1, p_2, \dots, p_n]: p_i \in \mathcal{P}_i, i = 1,2, \dots, n\}$.

Ejemplo

Se puede utilizar la técnica de parametrización para almacenar en el control finito un estado de un conjunto Q y los dos últimos símbolos que se han leído de la cinta con un alfabeto Γ.

$M = (\Sigma, \Gamma, Q', f, B, q_0, F)$ donde $Q' = Q \times \Gamma \times \Gamma$, es decir un estado de Q' tendría la forma $[q, A, B]$ con $q \in Q, \quad A, B \in \Gamma$

Se puede definir $f([q, A, B], C) = ([p, B, C], C, R)$ (leemos el símbolo C en la cinta, lo almacenamos en el nuevo estado cambiando también q por p, reescribimos el símbolo C en la cinta y avanzamos la cabeza de cinta a la derecha)

Técnicas de construcción de máquinas de Turing

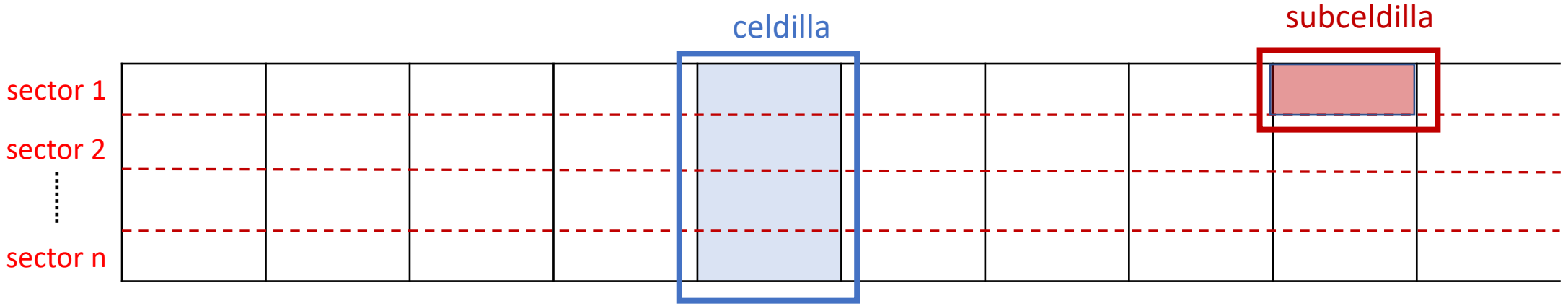
Cinta con varias bandas o sectores (I)

Esta técnica consiste en imaginar la cinta dividida horizontalmente en n bandas o sectores de modo que cada celdilla corta a cada sector en la correspondiente subceldilla, componiéndose ésta, en consecuencia, de una columna de n subceldillas.

Cada sector tiene su correspondiente alfabeto $\Gamma_i, i = 1, \ldots n$. El símbolo B está en cada Γ_i.

El alfabeto de la cinta Γ está, de este modo, definido como $\Gamma = \Gamma_1 \times \cdots \times \Gamma_n$ de forma que cada símbolo de la cinta puede verse como una n-tupla de símbolos de los correspondientes sectores.

El blanco de la cinta es, por tanto, $\mathcal{B}$= [B,...,B].



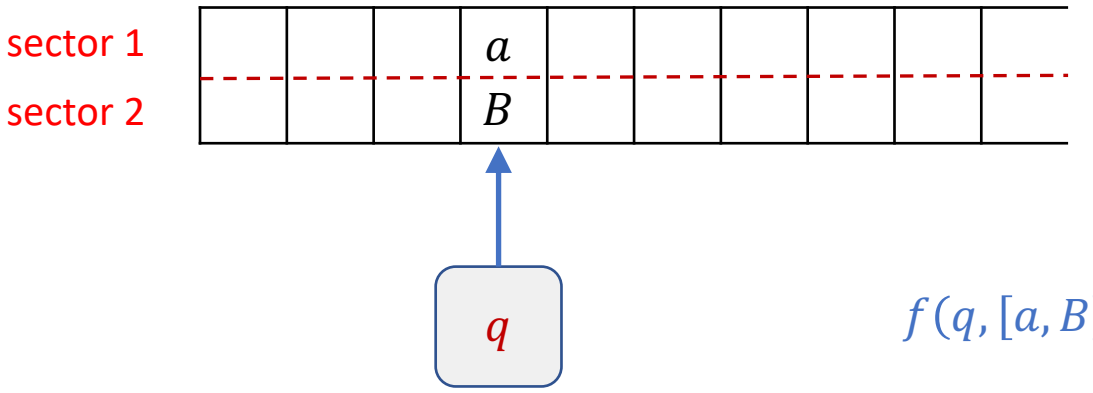
Técnicas de construcción de máquinas de Turing

Cinta con varias bandas o sectores (II)

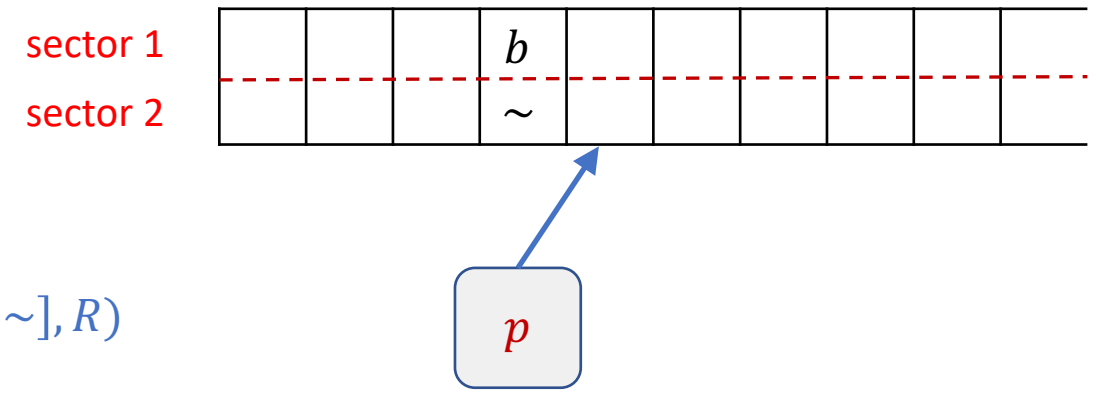
Para introducir la palabra de entrada se fija un sector, dígase el i-ésimo, de modo que el alfabeto de entrada de la máquina queda como $\Sigma = \{[B, \dots, a_i, \dots, B]: \ a_i \in \Sigma_i\}$ donde cada $[B, \dots, a_i, \dots, B]$ puede verse como a_i y $\Sigma_i \subseteq \Gamma_i - \{\mathcal{B}\}$

Ejemplo

Podemos utilizar una cinta con dos sectores donde en el sector superior se carga la cadena de entrada y en el sector inferior marcamos aquellos símbolos de entrada que hayamos modificado durante la computación con un símbolo especial $\sim$



$$f(q, [a, B]) = (p, [b, \sim], R)$$



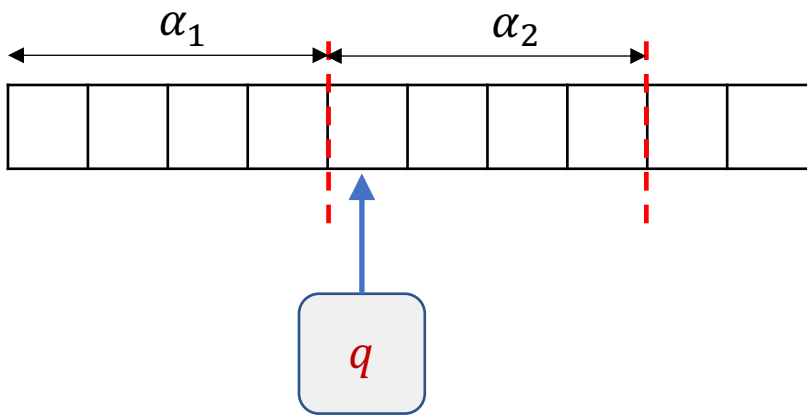
Técnicas de construcción de máquinas de Turing

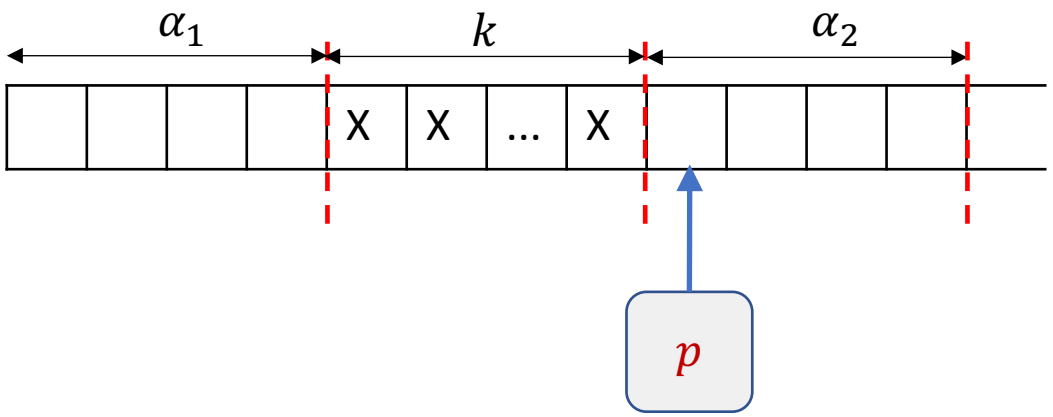
Desplazamientos en el contenido de la cinta (I)

Esta técnica permite trasladar el contenido de la cinta un número de celdillas predeterminado k a la derecha.

Supóngase que la máquina de Turing tiene un contenido de cinta α desde su inicio hasta que comienza el relleno con blancos. Supondremos que α originalmente no contiene blancos ni símbolos X.

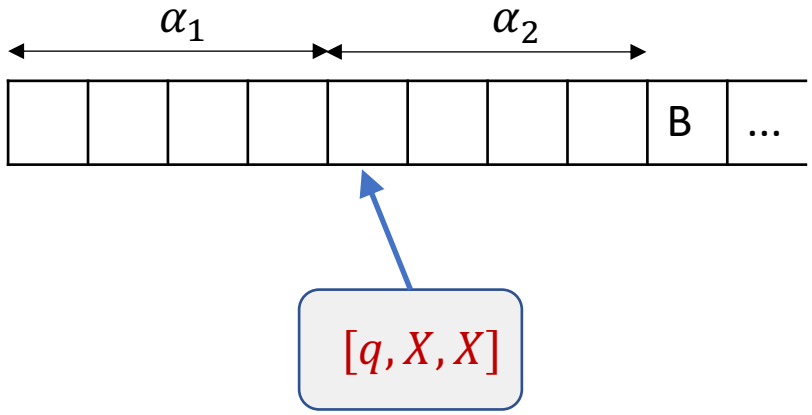
Sea el contenido de la cinta $\alpha = \alpha_1\alpha_2$ $(\alpha_1 \neq \lambda)$, y éste se desea transformar en $\alpha = \alpha_1 X^k \alpha_2$, desplazando por consiguiente el sufijo α_2 k posiciones, donde k es una constante mayor que 0, e insertando en el hueco la palabra de relleno X^k, dejando al final del proceso el cabezal ubicado en el último símbolo X.

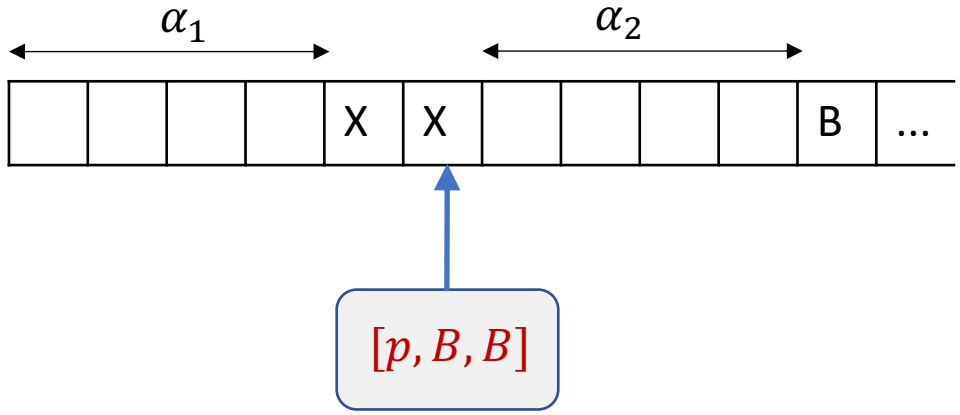




Técnicas de construcción de máquinas de Turing

Desplazamientos en el contenido de la cinta (II)





Un modo de hacerlo, por ejemplo para k = 2, es el siguiente:

- El desplazamiento comienza en el estado $[q, X, X]$ estando el cabezal sobre el primer símbolo de α_2 y termina en el estado $[p, B, B]$ con el cabezal ubicado en el último símbolo X.
- La porción de la función de transición asociada a esta tarea es la siguiente:
  - $f([q, A_1, A_2], A) = [q, A_2, A], A_1, R)$ para $A_1 \neq B \neq A_2$
  - $f([q, A_1, B], B) = ([p, B, B], A_1, L)$ para $A_1 \neq B$
  - $f([p, B, B], A) = ([p, B, B], A, L)$ para $A \neq X$
- A continuación la máquina continuará operando de acuerdo al resto de la función de transición

Técnicas de construcción de máquinas de Turing

Definición de subrutinas

- El concepto de subrutinas en máquinas de Turing coincide con el mismo concepto utilizado en lenguajes de programación de alto nivel, es decir fracciones de código que se pueden invocar en distintos puntos de ejecución con paso de parámetros y recursividad.

- El concepto de código en el caso de máquinas de Turing se corresponde con los movimientos definidos en la función de siguiente movimiento f.

- El paso de parámetros se puede realizar utilizando la técnica de almacenamiento en el control finito.

- Para asegurar el flujo de control de las subrutinas a la ejecución principal y viceversa, se pueden almacenar los estados actuales y de llamada en el almacenamiento del control finito.

- Las pilas de recursividad necesarias para almacenar las sucesivas llamadas de una subrutina a sí misma, se pueden habilitar en un sector de la cinta (en este caso, el alfabeto de ese sector se correspondería con el conjunto de estados de llamada posibles).

Máquinas de Turing modificadas

- El modelo básico de máquina de Turing, hasta el momento considerado, puede ser modificado de diferentes maneras sin que se modifique su poder computacional.

- Pueden plantearse modificaciones estructurales (cambiar el número de cintas, el número de cabezas de cinta, la disposición espacial de las celdillas en la cinta, etc.)

- También pueden plantearse modificaciones operativas (máquinas no deterministas, máquinas probabilistas, con restricciones en los movimientos, etc.).

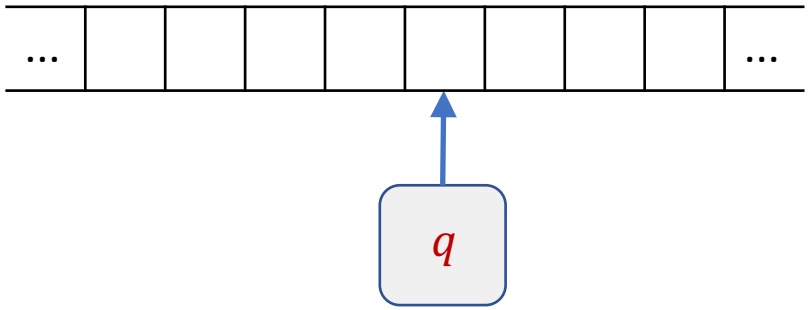
- Las modificaciones que planteamos, mantienen la capacidad computacional del modelo básico. No consideraremos otras modificaciones que superan ese límite computacional (por ejemplo, máquinas de Turing con *oráculos*) o que lo reducen (por ejemplo, máquinas de Turing con movimientos de cabeza de cinta sólo a la derecha).

Máquinas de Turing modificadas

Máquinas de Turing con la cinta infinita en ambos sentidos (I)

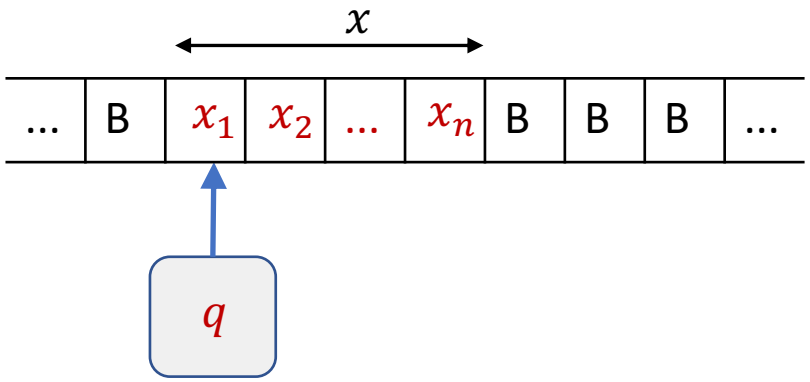
Este modelo de máquina de Turing es en todo similar al modelo básico salvo en la naturaleza de su cinta que en este caso presenta la extensión de prolongarse indefinidamente en ambos sentidos.

En este modelo de máquina no cabe la posibilidad de que la computación se aborte por intentar el cabezal un desplazamiento hacia la izquierda a partir de la primera celdilla de la cinta.



Máquinas de Turing modificadas

Máquinas de Turing con la cinta infinita en ambos sentidos (II)



Para analizar una palabra x, ésta se escribe comenzando en cualquier celdilla de la cinta (ya que no existe ninguna privilegiada) colocándose el cabezal en la celdilla en la que ésta comienza, caso en que $x \neq \lambda$. Las celdillas a la derecha e izquierda de x se encuentran rellenadas con blancos.

Si $x = \lambda$, entonces la cinta queda con todas las celdillas con blancos y el cabezal está inicialmente en cualquiera de ellas.

La definición de descripción instantánea y sus relaciones asociadas se definen de modo análogo; así como el lenguaje aceptado por la máquina.

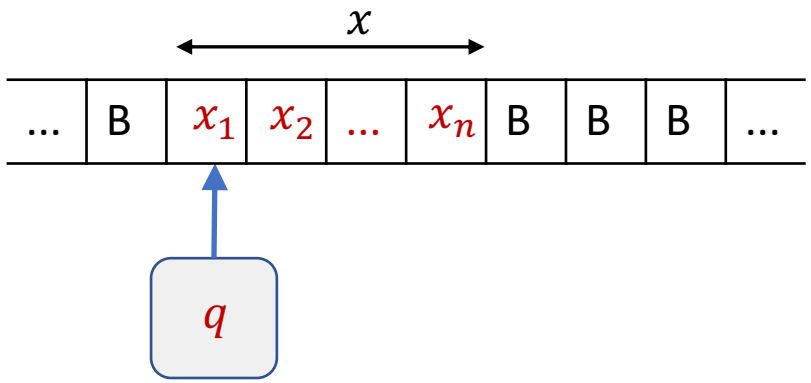
La descripción instantánea considera el contenido de cinta desde el símbolo no blanco más a la izquierda hasta el símbolo no blanco más a la derecha.

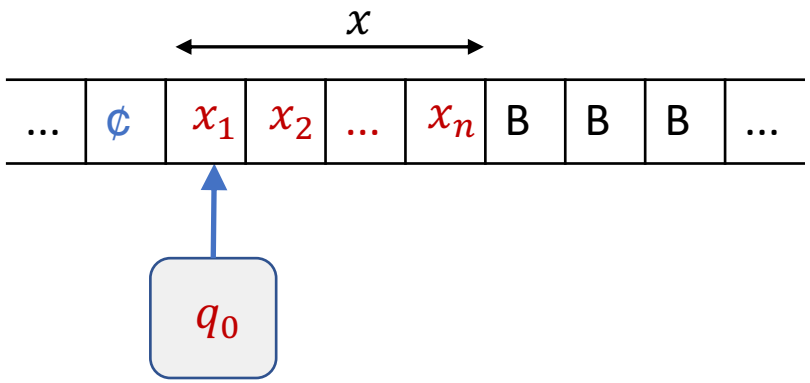
Máquinas de Turing modificadas

Máquinas de Turing con la cinta infinita en ambos sentidos (III)

Teorema: Si el lenguaje L es aceptado por una máquina de Turing con cinta acotada por la izquierda, entonces también es aceptado por una máquina de Turing con cinta infinita en ambos sentidos.

La simulación de una máquina de Turing con cinta acotada por la izquierda mediante una máquina de Turing con cinta infinita en ambos sentidos se puede hacer marcando un límite de cinta ficticio con un símbolo especial (por ejemplo, ¢) antes de comenzar la simulación de movimientos. Ante ese símbolo especial, la máquina no tiene definido movimiento por lo que quedaría bloqueada sin posibilidad de realizar más movimientos simulando el comportamiento de la máquina con cinta limitada.



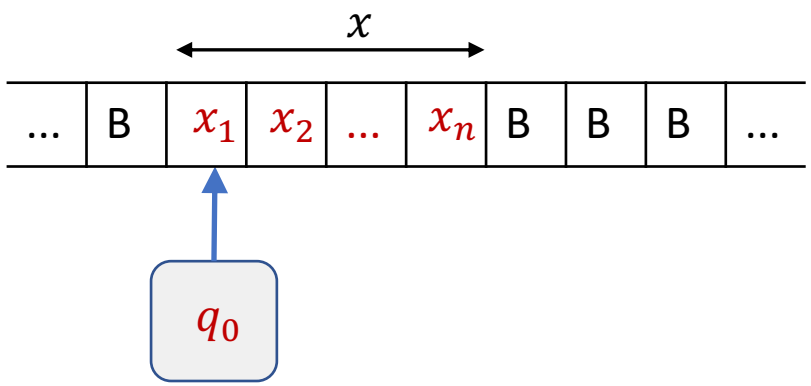


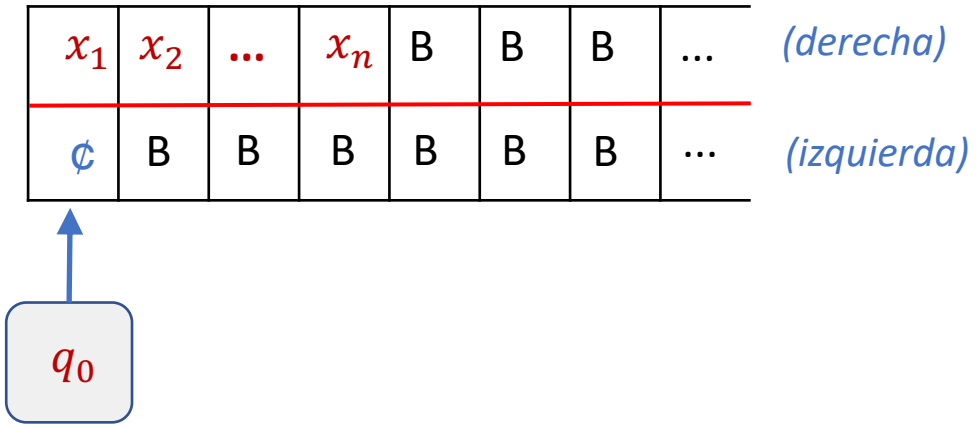
Máquinas de Turing modificadas

Máquinas de Turing con la cinta infinita en ambos sentidos (IV)

Teorema: Si el lenguaje L es aceptado por una máquina de Turing con cinta infinita en ambos sentidos, entonces también es aceptado por una máquina de Turing con cinta acotada por la izquierda.

La simulación de una máquina de Turing con cinta cinta infinita en ambos sentidos mediante una máquina de Turing con cinta acotada por la izquierda se puede hacer utilizando dos sectores que simulen la parte de cinta hacia la derecha desde la posición inicial en el sector superior y la parte de cinta hacia la izquierda desde la posición inicial en el sector inferior.



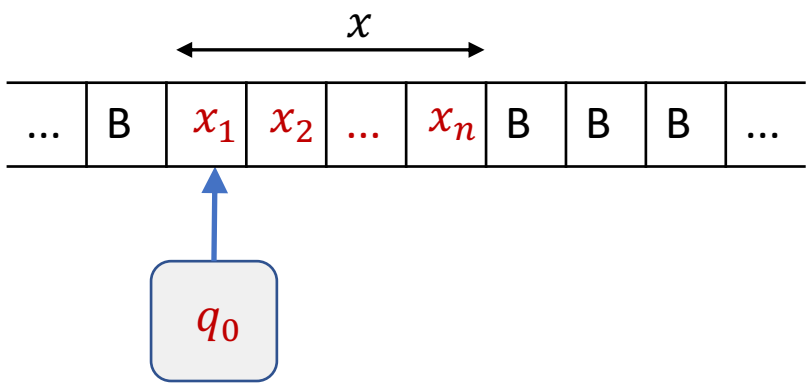


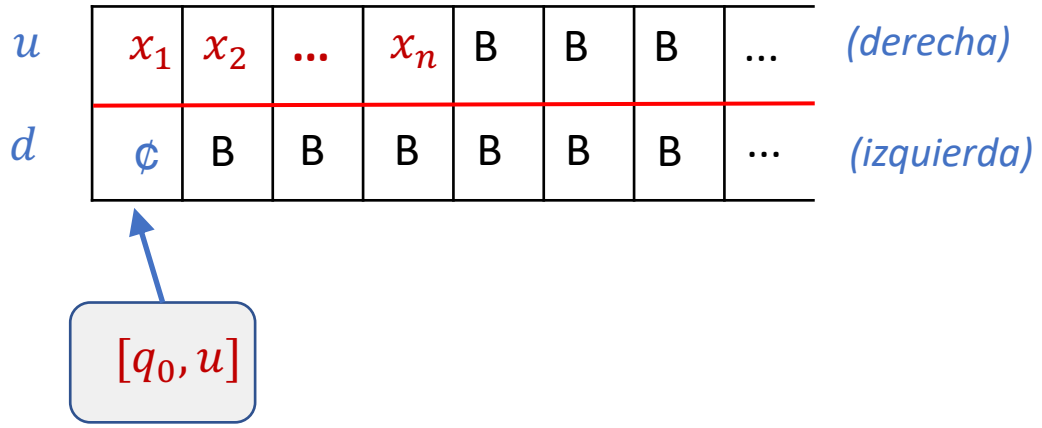
Máquinas de Turing modificadas

Máquinas de Turing con la cinta infinita en ambos sentidos (V)

Teorema: Si el lenguaje L es aceptado por una máquina de Turing con cinta infinita en ambos sentidos, entonces también es aceptado por una máquina de Turing con cinta acotada por la izquierda.

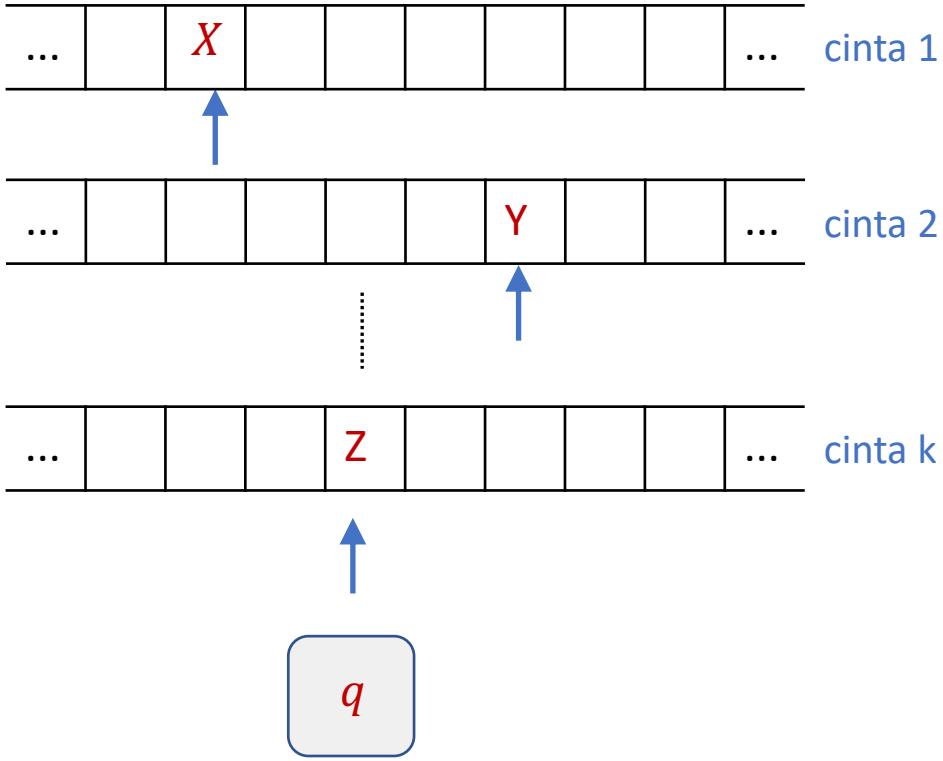
Durante la simulación de una máquina de Turing con cinta cinta infinita en ambos sentidos mediante una máquina de Turing con cinta acotada por la izquierda, el control finito almacena en qué sector se está realizando la simulación (u indica sector superior y d indica sector inferior). Esto facilita la simulación de movimientos (en el sector inferior se invierte la direccionalidad R/L) y también facilita el cambio de sector cuando se visite la celda con contenido [X, ¢].





Máquinas de Turing modificadas

Máquinas de Turing multicinta (I)



En este modelo la máquina de Turing dispone de *k* cintas independientes infinitas en ambos sentidos, cada una con su propio cabezal que puede manejarse de modo independiente.

En cada transición este modelo de máquina lee el símbolo de cada cinta y a partir del estado en que se encuentra el control finito, a partir de lo que especifique la función de transición, puede:

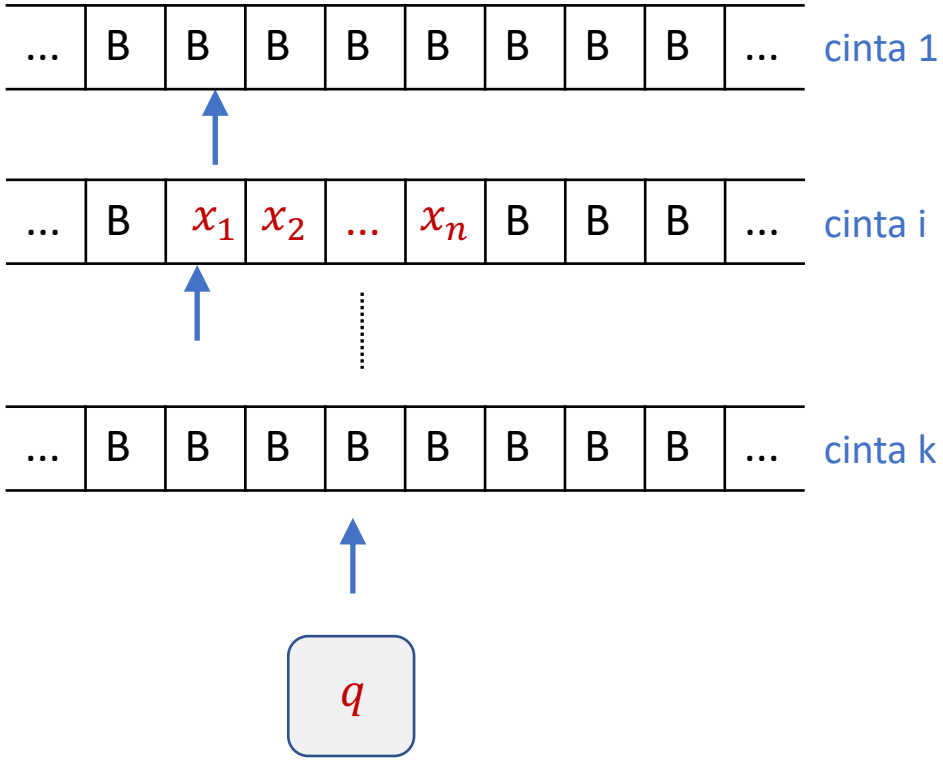
- cambiar de estado,
- escribir un símbolo en la correspondiente celdilla de cada cinta, y
- desplazar cada cabezal, en cada cinta, de modo independiente.

$$f(q, X_1, X_2, \dots, X_k) = (p, Y_1, Y_2, \dots, Y_k, m_1, m_2, \dots, m_k)$$

$$q, p \in Q$$
$$X_i, Y_i \in \Gamma \ 1 \leq i \leq k$$
$$m_i \in \{L, R\} \ 1 \leq i \leq k$$

Máquinas de Turing modificadas

Máquinas de Turing multicinta (II)



Hay que tener en cuenta que en este modelo y en cada transición algunos de los cabezales pueden permanecer inmóviles.

De las k cintas una, dígase la i-ésima, está designada como cinta de entrada, así el alfabeto de entrada de la máquina es $\Sigma_i \subseteq \Gamma_i - \{B\}$, siendo Γ_i el alfabeto de cinta utilizado en la cinta i. Sobre ella se escribe la entrada $x_1 x_2 \ldots x_n$ como en el modelo anterior; el resto de cintas se encuentran con todas las celdillas con el símbolo blanco.

La definición de descripción instantánea y las relaciones de alcanzabilidad correspondiente se realiza de modo análogo al de una máquina estándar; así como el lenguaje aceptado por la máquina.

Máquinas de Turing modificadas

Máquinas de Turing multicinta (III)

Teorema: Si el lenguaje L es aceptado por una máquina de Turing con cinta acotada por la izquierda, entonces también es aceptado por una máquina de Turing multicinta.

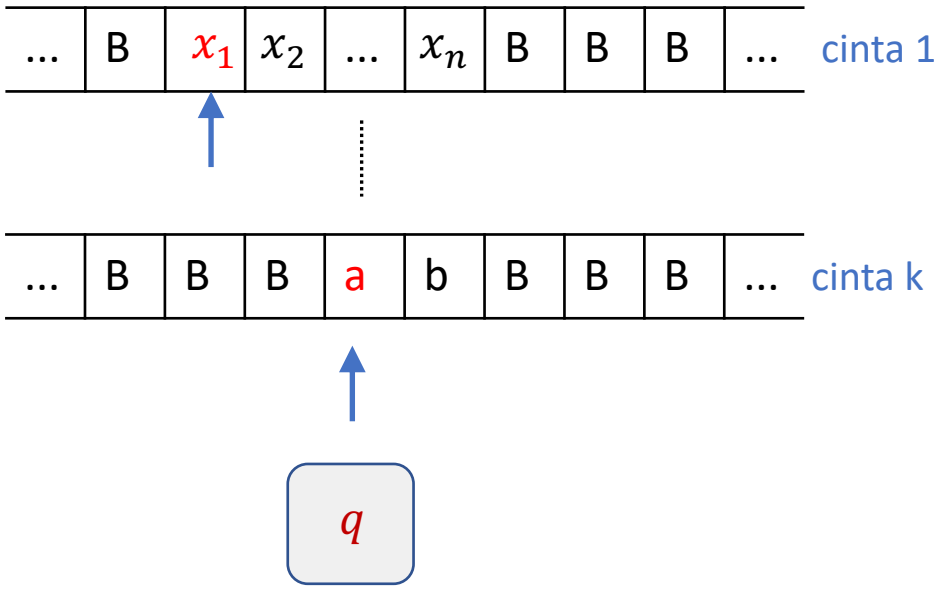
La veracidad del teorema queda establecida al considerar que el modelo de máquina de Turing estándar es un caso restringido del modelo multicinta donde se limita a 1 el número de cintas y el límite por la izquierda se puede simular mediante la misma técnica empleada al simular máquinas de Turing estándar mediante máquinas de Turing con cinta infinita en ambos sentidos.

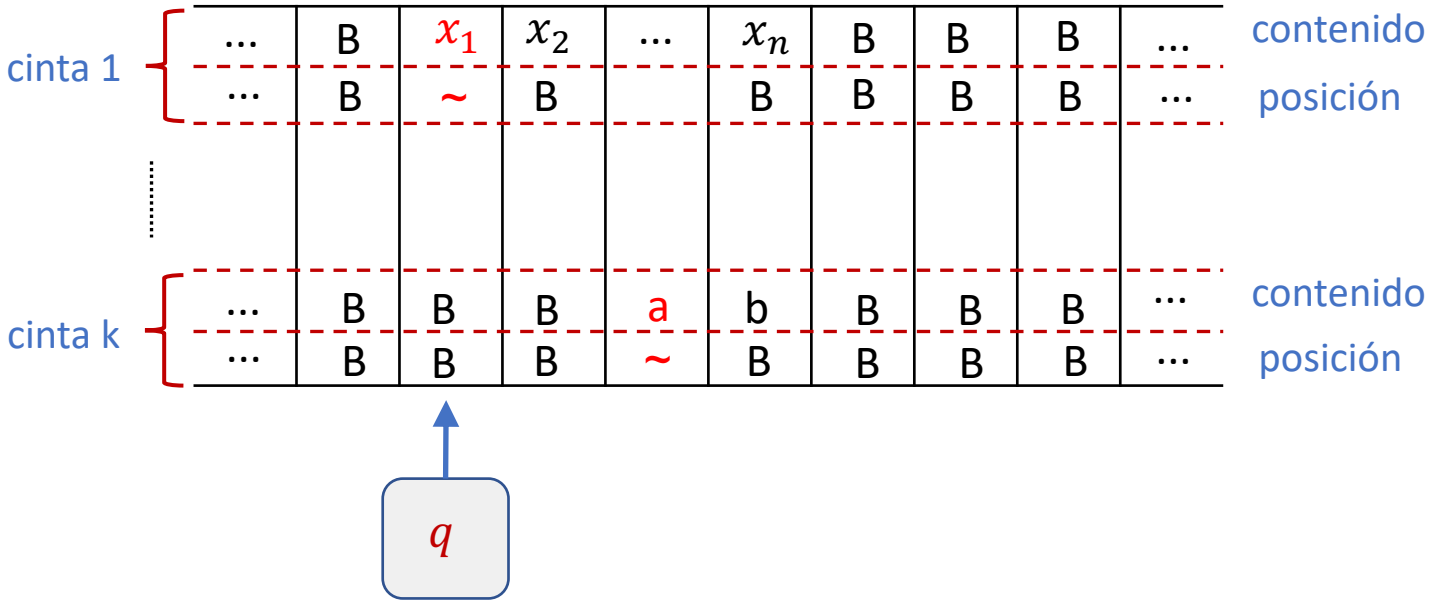
Máquinas de Turing modificadas

Máquinas de Turing multicinta (IV)

Teorema: Si el lenguaje L es aceptado por una máquina de Turing multicinta, entonces también es aceptado por una máquina de Turing con cinta acotada por la izquierda.

Podemos obviar el límite de la cinta por la izquierda. Para la simulación de la máquina multicinta utilizaremos una máquina con una sola cinta y varios sectores. Por cada cinta de la máquina multicinta utilizaremos dos sectores: en uno de ellos alojaremos el contenido de la cinta y en el otro sector pondremos una marca que indica la posición de la cabeza.

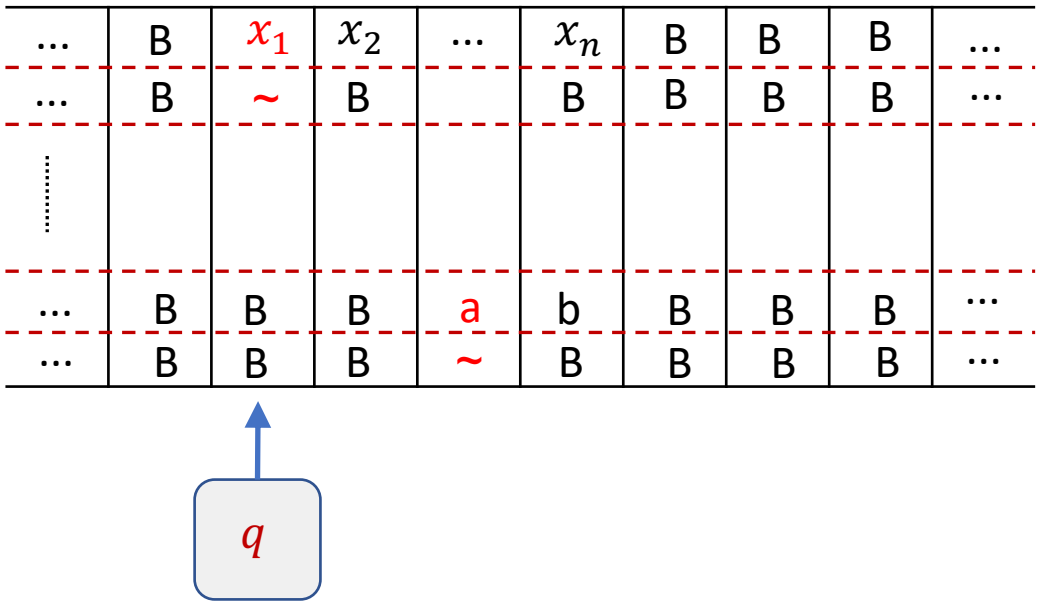




Máquinas de Turing modificadas

Máquinas de Turing multicinta (V)

Teorema: Si el lenguaje L es aceptado por una máquina de Turing multicinta, entonces también es aceptado por una máquina de Turing con cinta acotada por la izquierda.



Para la simulación de un movimiento de la máquina multicinta, la máquina estándar sitúa su cabeza de cinta en la celda con la marca de posición más a la izquierda. Conforme avanza hacia la derecha y va recorriendo marcas, almacena los contenidos marcados en su control finito. Una vez ha recopilado todas las marcas, inicia su recorrido hacia la izquierda para cambiar los contenidos y mover las marcas a la izquierda o la derecha. Una vez que ha realizado todos los cambios puede iniciar de nuevo otro ciclo de simulación

$[q, x_1, \ldots, a]$ (el control finito almacena el estado y los contenidos de las k cintas)

La simulación de i movimientos de la máquina multicinta puede suponer realizar i^2 movimientos en la máquina estándar, lo cual supone una pérdida de eficiencia cuadrática con respecto a la medida de complejidad computacional temporal.

Máquinas de Turing modificadas

Máquinas de Turing indeterministas (I)

El modelo de máquina de Turing indeterminista coincide con el modelo o básico, o equivalente con el modelo con cinta infinita en ambos sentidos, en todos los aspectos salvo en uno fundamental: la definición de la función de transición; en este modelo ésta se define como

$$f: Q \times \Gamma \longrightarrow \mathcal{P}(Q \times \Gamma \times \{L, R\})$$

donde $\mathcal{P}(A)$ denota el conjunto potencia del conjunto A, es decir el conjunto de todos los subconjuntos posibles formados por elementos del conjunto A.

De esta forma, $(\forall\, q \in Q)(\forall\, a \in \Gamma)\ (|f(q,a)| \text{ es finito })$.

Esto implica que
- $f(q,a) = \emptyset$, o
- $f(q,a) = \{(q_i, b_i, m_i) \mid i = 1, \ldots, n_{q,a}\}$

Es posible que la máquina indeterminista, para un estado y un símbolo, tenga definido un número finito de movimientos que puede aplicar (es decir, un cambio de estado, una escritura de símbolo y un movimiento de la cabeza de cinta a izquierda o derecha). La máquina, cuando se encuentre en la configuración instantánea correspondiente a ese estado y ese símbolo, selecciona uno de los movimientos posibles. A priori, es imposible predecir qué movimiento selecciona la máquina y decimos que la máquina selecciona el movimiento de forma no determinista.

Máquinas de Turing modificadas

Máquinas de Turing indeterministas (II)

Así, si $f(q,a) = \{(q_i, b_i, m_i) \mid i = 1, \dots, n_{q,a}\}$ al leer el símbolo a estando el control finito en el estado q la máquina puede optar, de un modo indeterminista, en ejecutar una de las transiciones (q_i, b_i, m_i) $i = 1, \dots, n_{q,a}$.

Esto conlleva que, dada una descripción instantánea I, ésta pueda tener más de una descripción instantánea siguiente I'. Así, en general,

$$I \vdash I_k \quad k = 1, \dots, n$$

Dando lugar, para una entrada dada, en lugar de una secuencia de computación, como en el caso determinista, a un árbol de computaciones.

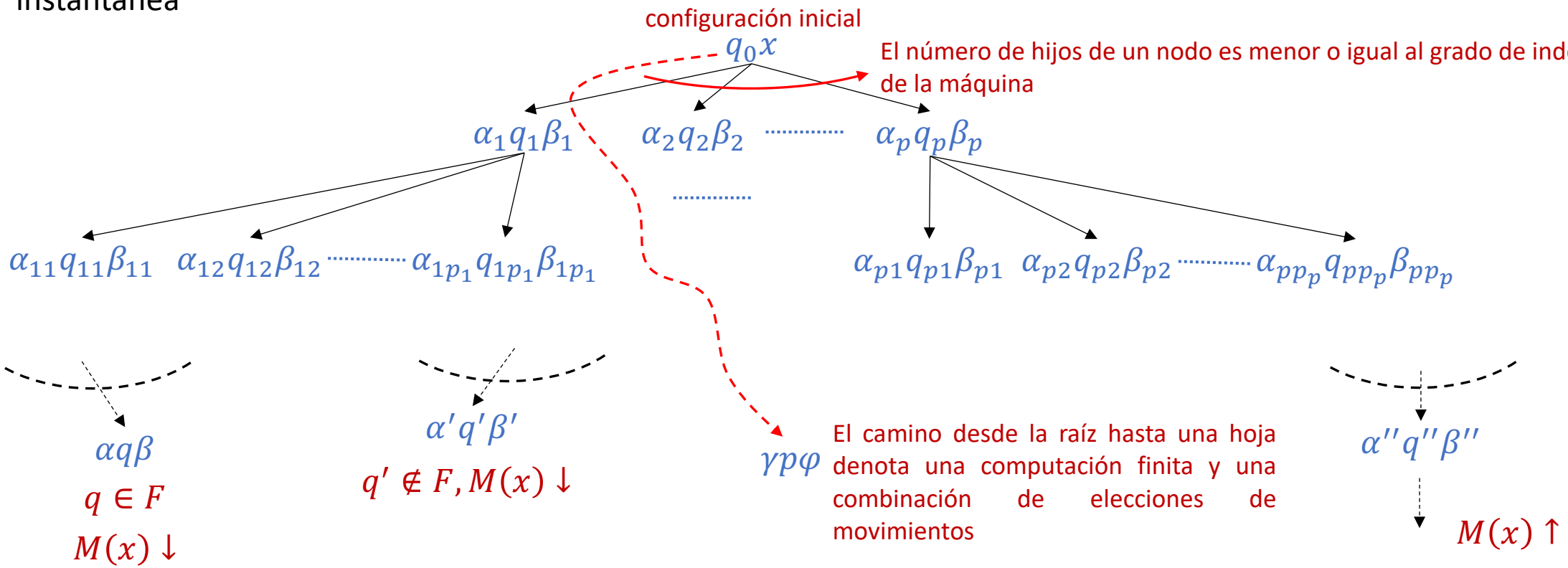
Denominaremos grado de indeterminismo de una máquina indeterminista al valor máximo $n_{q,a}$ $\forall\, q \in Q\,, \forall\, a \in \Gamma$. Es decir, el número máximo de movimientos definidos en su función de transición, sea cual sea el estado de su control finito y el símbolo analizado en la cinta.

Obsérvese que la máquina de Turing estándar puede ser considerada una máquina de Turing indeterminista con grado de indeterminismo 1.

Máquinas de Turing modificadas

Máquinas de Turing indeterministas (III)

Para cada entrada x en la máquina, se puede definir un árbol de computación que permite analizar todas las posibles combinaciones de elección de movimientos durante la computación, En ese árbol cada nodo es una configuración instantánea



Máquinas de Turing modificadas

Máquinas de Turing indeterministas (IV)

El lenguaje aceptado por el modelo indeterminista se define igual que en el modelo básico, esto es, $L(M) = \{x \in \Sigma^*: q_{0x} \vdash^* \alpha p \beta, p \in F\}$. Esto significa que existe una combinación de movimientos en la máquina que, desde la configuración inicial conduce a la máquina a una configuración de aceptación)

Aunque, no obstante, ahora debe notarse que pueden darse los casos en los que para una palabra del lenguaje aceptado por la máquina puedan darse también computaciones:

- que terminen rechazando, y
- otras que no terminen.

Teorema: Si el lenguaje L es aceptado por una máquina de Turing determinista, entonces también es aceptado por una máquina de Turing indeterminista.

Para la demostración de este resultado basta con considerar que la máquina determinista es, a su vez, una máquina indeterminista con grado de indeterminismo 1.

Máquinas de Turing modificadas

Máquinas de Turing indeterministas (V)

Teorema: Si el lenguaje L es aceptado por una máquina de Turing indeterminista, entonces también es aceptado por una máquina de Turing indeterminista.

Consideremos que la máquina de Turing indeterminista M tiene un grado de indeterminismo r.

Definiremos una secuencia de computación de tamaño p, como una secuencia finita de valores enteros $i_1, i_2, \dots, i_p$ con $i_j \in \{1, .., r\}\ 1 \leq j \leq p$. Dado una grado de indeterminismo r, las secuencias de computación se pueden generar en orden canónico (primero las de menor tamaño y para tamaños idénticos, en orden creciente numérico).

En la máquina M, consideramos una enumeración de movimientos para cada estado y símbolo, de la forma

$$f(q, a) = \{(q_i, b_i, m_i) \mid i = 1, \dots, n_{q,a}\}$$

(obsérvese que $n_{q,a} \leq r$). Por lo tanto, cada movimiento lo podemos asociar con un valor entero.

Una secuencia de computación $i_1, i_2, \dots, i_p$, diremos que es plausible para la máquina M y la configuración inicial $q_0 x$ si en el paso de computación j-ésimo se puede aplicar el movimiento i_j.

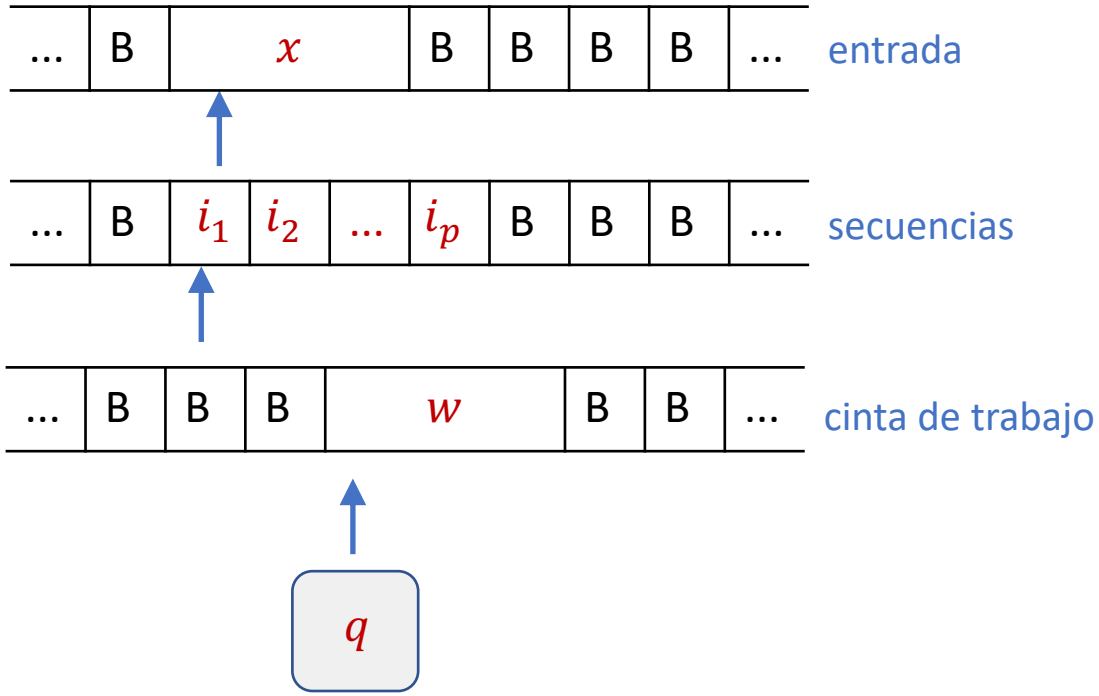
Máquinas de Turing modificadas

Máquinas de Turing indeterministas (VI)

Teorema: Si el lenguaje L es aceptado por una máquina de Turing indeterminista, entonces también es aceptado por una máquina de Turing indeterminista.

Podemos comprobar la aceptación de una cadena de entrada en una máquina indeterminista mediante una máquina multicinta determinista con la siguiente estructura de cintas:

- Una cinta donde se alojará la entrada x
- Una cinta donde se generan secuencias de computación en orden numérico creciente en un sistema con r valores de dígito posibles.
- Una cinta de trabajo



Máquinas de Turing modificadas

Máquinas de Turing indeterministas (VII)

Teorema: Si el lenguaje L es aceptado por una máquina de Turing indeterminista, entonces también es aceptado por una máquina de Turing indeterminista.

Para comprobar si una máquina indeterminista puede aceptar una cadena de entrada, la máquina multicinta determinista realizará el siguiente flujo de acciones:

1. Copiar la cadena de entrada x en la cinta de trabajo
2. Generar una secuencia de computación (aplicando el orden canónico)
3. Simular los movimientos de la secuencia de computación aplicándolos en la cinta de trabajo, almacenando el estado de la máquina indeterminista en el control finito

Si la máquina puede finalizar la secuencia de computación en un estado de aceptación, entonces para y acepta la cadena de entrada. En el caso de que la secuencia de computación no sea plausible o de que finalice en un estado que no es de aceptación, la máquina volvería a repetir la secuencia de acciones anterior, continuando por la siguiente secuencia de computación.

La aceptación de una cadena de entrada en la máquina indeterminista mediante i movimientos puede suponer pasar por r^i configuraciones distintas en la máquina determinista multicinta, siendo r el grado de indeterminismo, lo cual supone una pérdida de eficiencia exponencial con respecto a la medida de complejidad computacional temporal.

La máquina de Turing como enumerador de lenguajes (generadores de Turing)

La máquina de Turing además de reconocedor de lenguajes puede utilizarse como generador o enumerador de lenguajes.

Cuando actúa como generador de lenguajes se usa en su versión multicinta. En esta situación carece de cinta de entrada y se designa una de las cintas como cinta de salida.

La cinta de salida tiene las restricciones siguientes:

- su cabezal no puede retroceder,
- y se desplaza una celdilla a la derecha si y sólo si se realiza una operación de escritura de un símbolo diferente de B.

La máquina de Turing como enumerador de lenguajes (generadores de Turing)

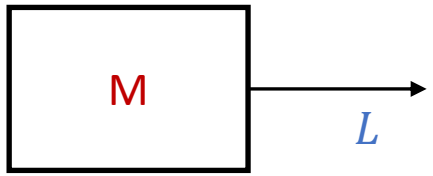
La cinta de salida tiene como alfabeto de cinta Γ_S y como alfabeto de salida Σ_S, cumpliéndose $\Sigma_S = \Gamma_S - \{B, \#\}$ donde el símbolo # es un símbolo especial de separación.

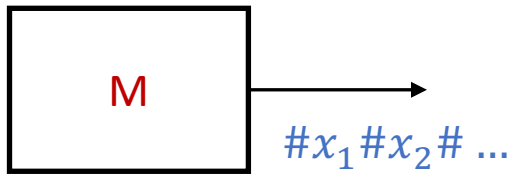
Inicialmente la máquina arranca con todas las cintas con símbolos en blanco, y carece de estados finales, esto es $F = \emptyset$.

El lenguaje, en esta condiciones, generado por un generador de Turing M se denota por G(M) y se define como

G(M) = { $x \in \Sigma_S^*$: la palabra $\#x\#$ aparece alguna vez escrita en la cinta de salida}

Podemos utilizar los siguientes esquemas para denotar un generador de Turing donde $L = G(M)$ y $x_i \in G(M)$





La máquina de Turing como enumerador de lenguajes (máquinas generadoras)

No existe ninguna restricción en cuanto al número de veces que una misma palabra puede aparecer en la cinta de salida.

Dos generadores de Turing M y M' son equivalentes si y sólo si G(M) = G(M').

Propiedad: Dado un generador de Turing M existe un generador equivalente de Turing M' que genera cada palabra una sola vez.

El generador M' dispondrá de una cinta adicional a las del generador M. Esta cinta adicional actuará como cinta de salida de M'. El generador M' simulará los movimientos de M y, cada vez que M escriba en su cinta de salida, M' comprobará en esa cinta si la cadena había aparecido previamente (recuérdese que en esa cinta la máquina M' puede hacer movimientos a la izquierda). Sólo en el caso de que la cadena no se haya escrito previamente, se escribe en la cinta de salida de M'.

La máquina de Turing como enumerador de lenguajes (máquinas generadoras)

Caracterización de los lenguajes recursivamente enumerables mediante generadores de Turing

La clase de los lenguajes recursivamente enumerables se ha definido a partir de la máquina de Turing actuando como reconocedor. Seguidamente la caracterizaremos a partir de generadores de Turing.

Teorema: Un lenguaje es recursivamente enumerable si y sólo si es generado por un generador de Turing

Para la demostración del teorema, bastará con establecer la veracidad de las dos siguientes afirmaciones:

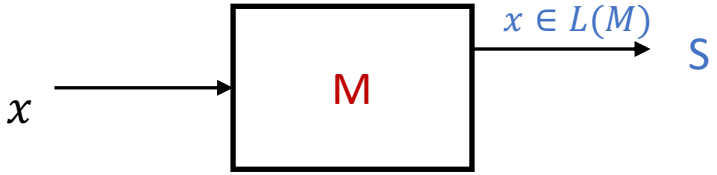
- para cada lenguaje aceptado por una máquina de Turing, existe un generador de Turing que lo genera.
- para cada lenguaje generado por un generador de Turing, existe una máquina de Turing que lo acepta.

La máquina de Turing como enumerador de lenguajes (máquinas generadoras)

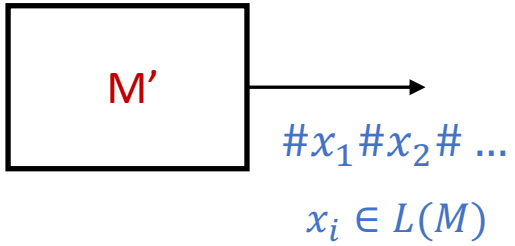
Caracterización de los lenguajes recursivamente enumerables mediante generadores de Turing

Para cada lenguaje aceptado por una máquina de Turing, existe un generador de Turing que lo genera.

Para la demostración del anterior enunciado partiendo de un esquema aceptor



... debemos construir un esquema generador

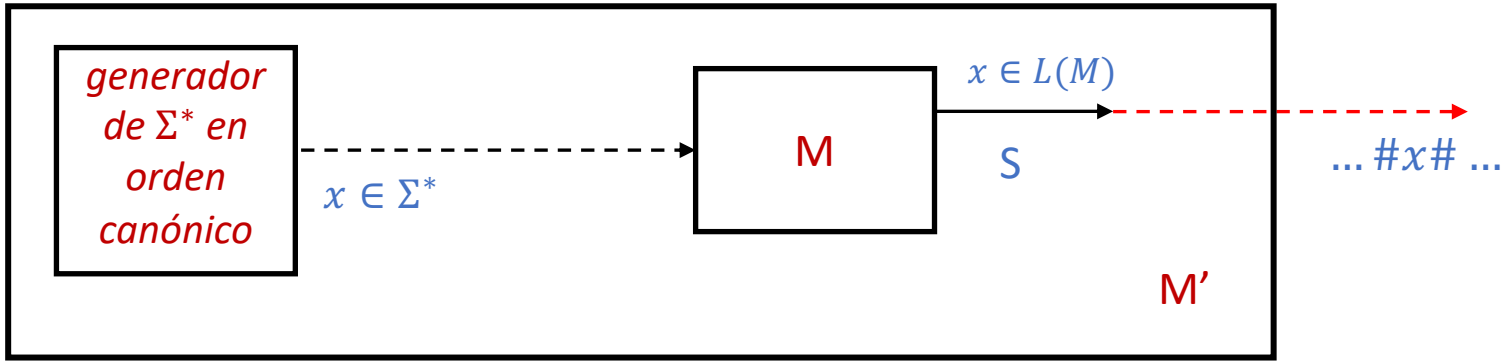


tal que $G(M') = L(M)$

La máquina de Turing como enumerador de lenguajes (máquinas generadoras)

Caracterización de los lenguajes recursivamente enumerables mediante generadores de Turing

Una primera aproximación para construir M' a partir de M, podría ser la que se sigue en este esquema



Los esquemas dentro del esquema de M' pueden definirse como subrutinas dentro de la máquina.

La idea subyacente es comprobar qué cadenas de Σ^*acepta M. Cada una de las cadenas aceptadas es escrita en la cinta de salida de M'

Si M, ante una cadena de entrada x, no finaliza su computación, entonces M' no llegará a comprobar el resto de las (infinitas) cadenas de Σ^* y L(M) podría no llegar a generarse de forma completa.

La máquina de Turing como enumerador de lenguajes (máquinas generadoras)

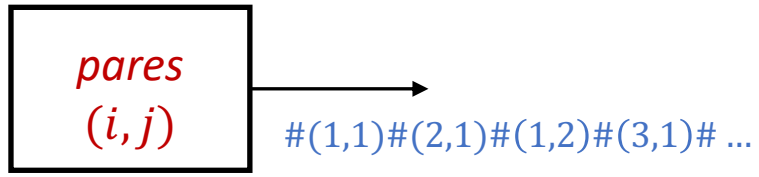
Caracterización de los lenguajes recursivamente enumerables mediante generadores de Turing

Generadores de pares de números naturales

$j \downarrow$ \ $i \rightarrow$	1	2	3	4	5	...
1	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	...
2	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	...
3	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	...
4	(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	...
5	(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	...
...	...	...	...	...	...	...

La enumeración de los pares de números naturales de la forma (i,j) se puede realizar de forma efectiva siguiendo el orden creciente de los valores $i+j$. Para aquellos pares con idéntico valor $i+j$ entonces se puede establecer un orden creciente con el valor j (o equivalentemente decreciente con el valor i). Esta enumeración se corresponde con las trayectorias diagonales marcadas en la tabla.

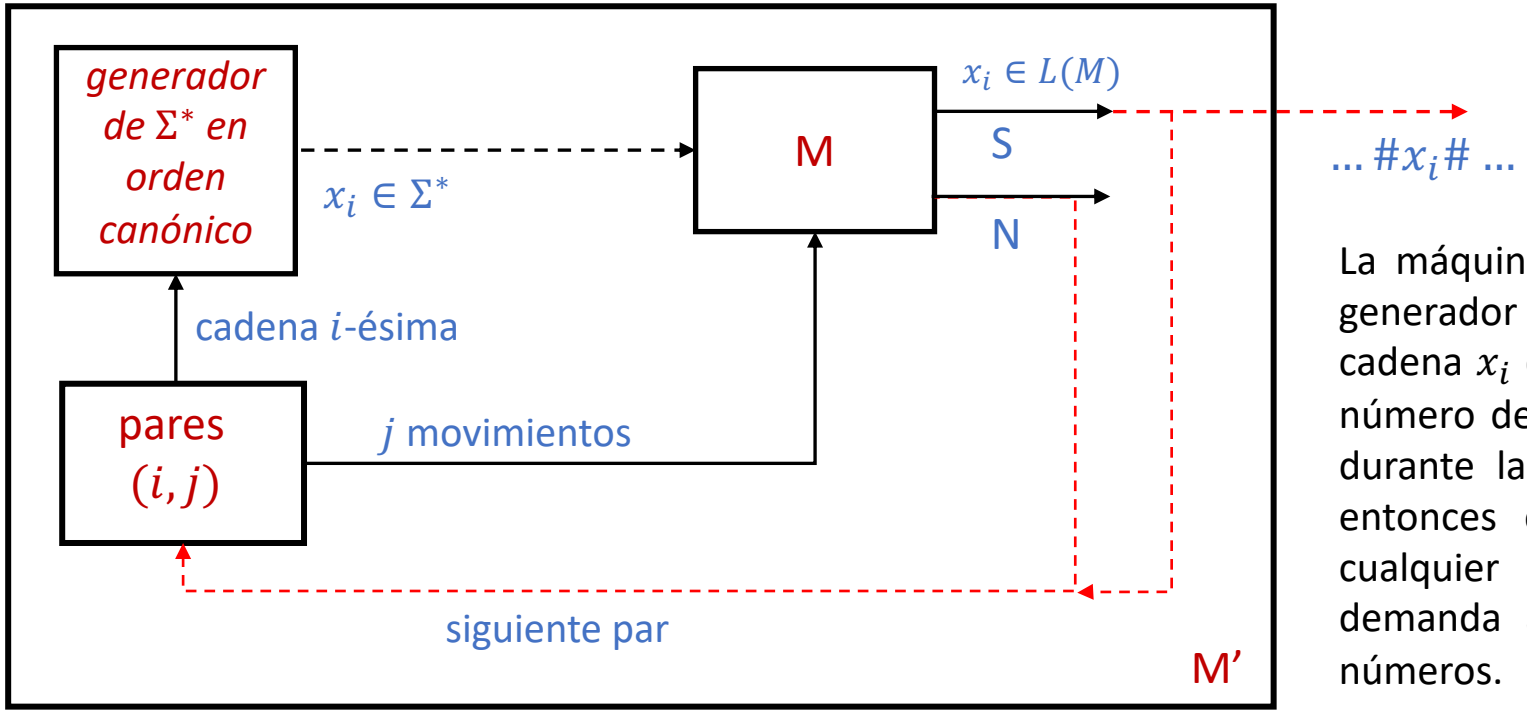
Podemos definir un generador de Turing que, siguiendo la enumeración establecida, genere todos los pares de números naturales debidamente codificados.



La máquina de Turing como enumerador de lenguajes (máquinas generadoras)

Caracterización de los lenguajes recursivamente enumerables mediante generadores de Turing

Una segunda aproximación para construir M' a partir de M, podría ser la que se sigue en este esquema



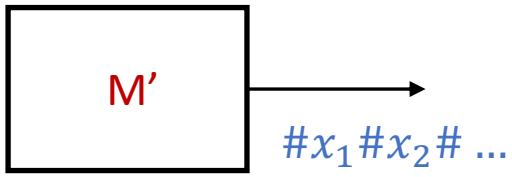
La máquina M' actúa poniendo en funcionamiento un generador de pares (i, j). La componente i denota la cadena x_i de Σ^* mientras que la componente j indica el número de movimientos a simular en la máquina M. Si durante la simulación de M, se acepta la cadena x_i, entonces esta se escribe en la cinta de salida. En cualquier caso, una vez finalizada la simulación se demanda al generador de pares el siguiente par de números.

La máquina de Turing como enumerador de lenguajes (máquinas generadoras)

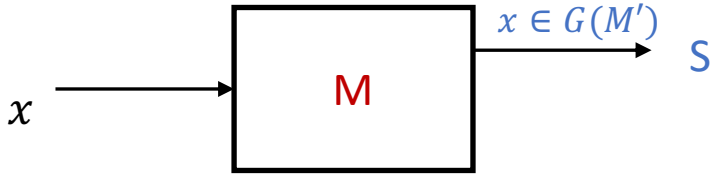
Caracterización de los lenguajes recursivamente enumerables mediante generadores de Turing

Para cada lenguaje generado por un generador de Turing, existe una máquina de Turing que lo acepta.

Para la demostración del anterior enunciado partiendo de un esquema generador



... debemos construir un esquema aceptor

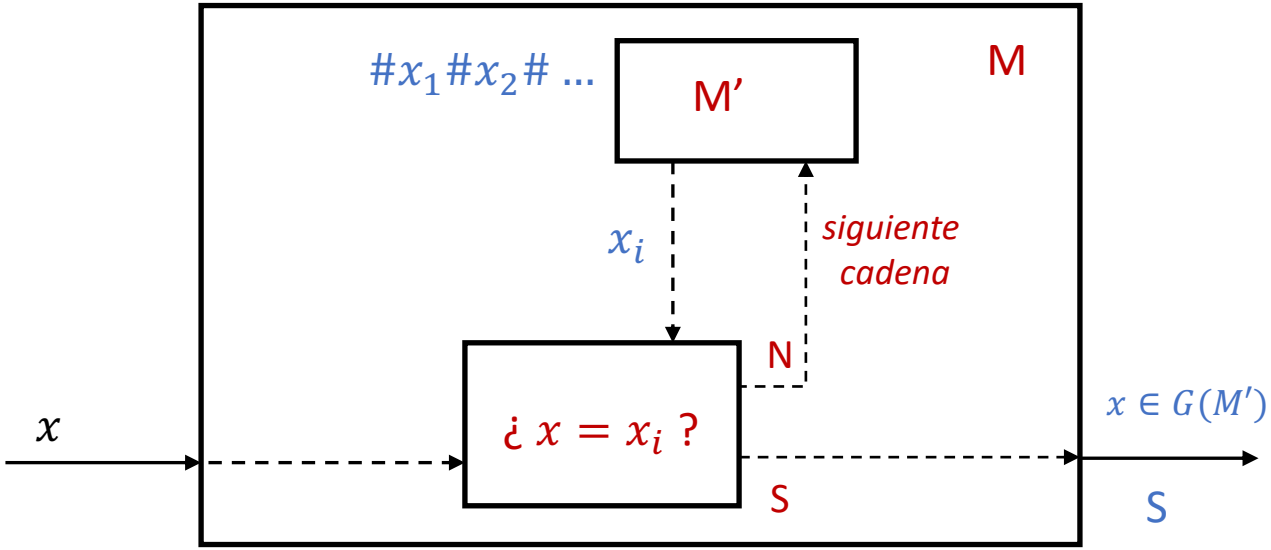


tal que $G(M') = L(M)$

La máquina de Turing como enumerador de lenguajes (máquinas generadoras)

Caracterización de los lenguajes recursivamente enumerables mediante generadores de Turing

Un esquema para la demostración del anterior enunciado es el siguiente



La máquina M pone a funcionar el generador de Turing M' y, cada vez que éste genera una cadena en su cinta de salida, compara la cadena generada con su cadena de entrada x. Si ambas cadenas coinciden entonces M para y acepta su cadena de entrada ya que pertenece al lenguaje $G(M')$. En caso de que las cadenas no coincidan, entonces M cede el control a M' a la espera de que vuelva a generar otra cadena en su cinta de salida.

En definitiva, M sólo acepta las cadenas generadas por M' y, por lo tanto, $L(M) = G(M')$.

La máquina de Turing como enumerador de lenguajes (máquinas generadoras)

Caracterización de los lenguajes recursivos mediante generadores de Turing

La clase de los lenguajes recursivos se ha definido a partir de la máquina de Turing actuando como reconocedor. Seguidamente la caracterizaremos a partir de generadores de Turing.

Teorema: Un lenguaje es recursivo si y sólo si existe un generador de Turing que enumere sus palabras en orden canónico

Para la demostración del teorema, bastará con establecer la veracidad de las dos siguientes afirmaciones:

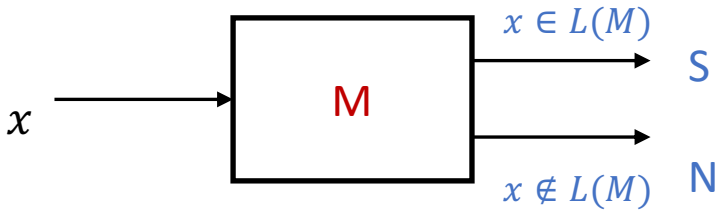
- para cada lenguaje aceptado por una máquina de Turing que siempre finaliza su computación, existe un generador de Turing que lo genera en orden canónico.
- para cada lenguaje generado por un generador de Turing en orden canónico, existe una máquina de Turing que lo acepta y siempre finaliza sus computaciones.

La máquina de Turing como enumerador de lenguajes (máquinas generadoras)

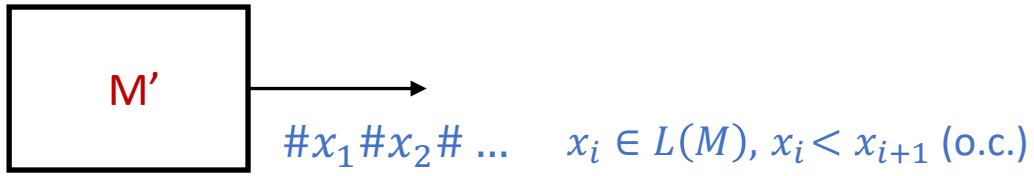
Caracterización de los lenguajes recursivos mediante generadores de Turing

Para cada lenguaje aceptado por una máquina de Turing que siempre finaliza sus computaciones, existe un generador de Turing que lo genera en orden canónico.

Para la demostración del anterior enunciado partiendo de un esquema aceptor



... debemos construir un esquema generador

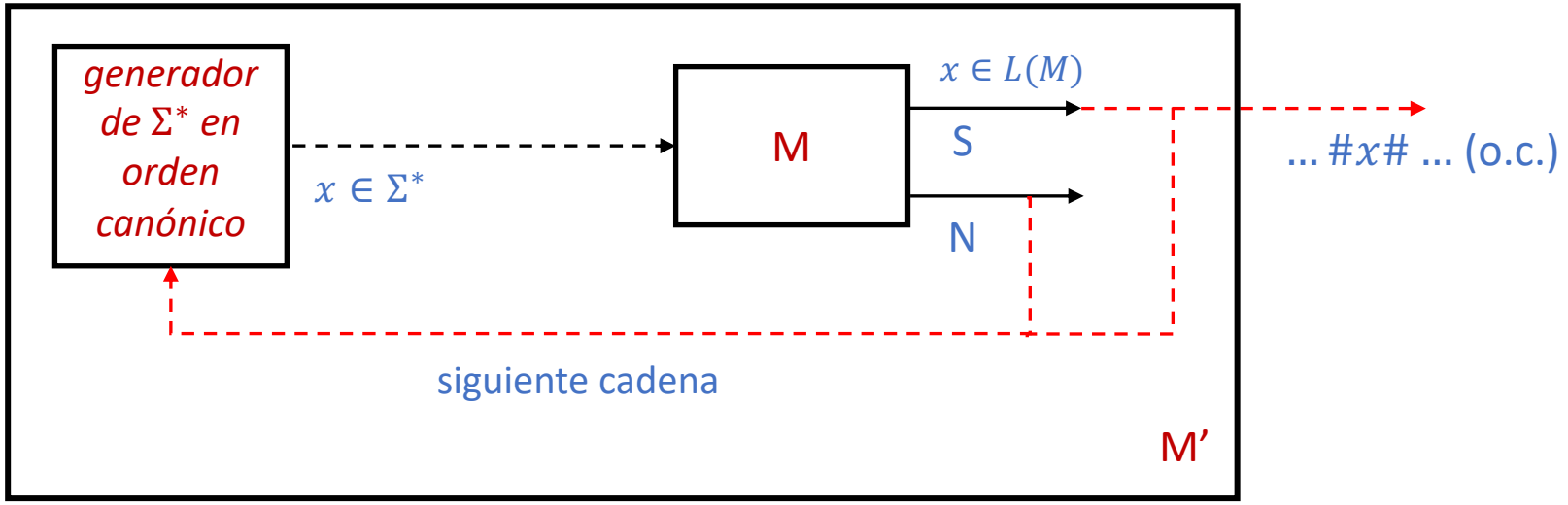


tal que $G(M') = L(M)$

La máquina de Turing como enumerador de lenguajes (máquinas generadoras)

Caracterización de los lenguajes recursivos mediante generadores de Turing

Un esquema para la demostración del anterior enunciado es el siguiente



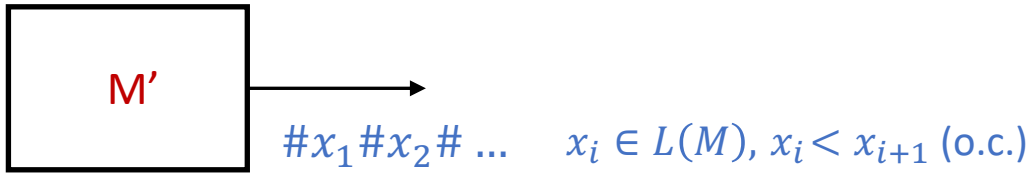
La idea subyacente es comprobar qué cadenas de Σ^*acepta M. Cada una de las cadenas aceptadas es escrita en la cinta de salida de M'. Dado que se pueden ir probando las cadenas en orden canónico, entonces también se escribirán en ese orden.

La máquina de Turing como enumerador de lenguajes (máquinas generadoras)

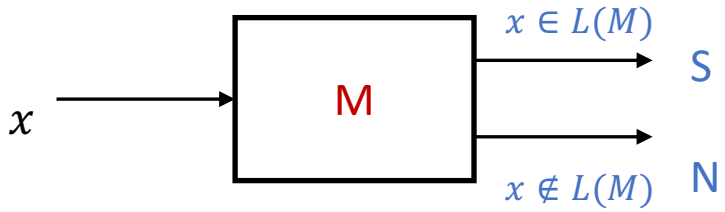
Caracterización de los lenguajes recursivos mediante generadores de Turing

Para cada lenguaje generado en orden canónico por un generador de Turing existe una máquina de Turing que lo acepta y siempre finaliza sus computaciones.

Para la demostración del anterior enunciado partiendo de un esquema generador (en orden canónico



... debemos construir un esquema aceptor

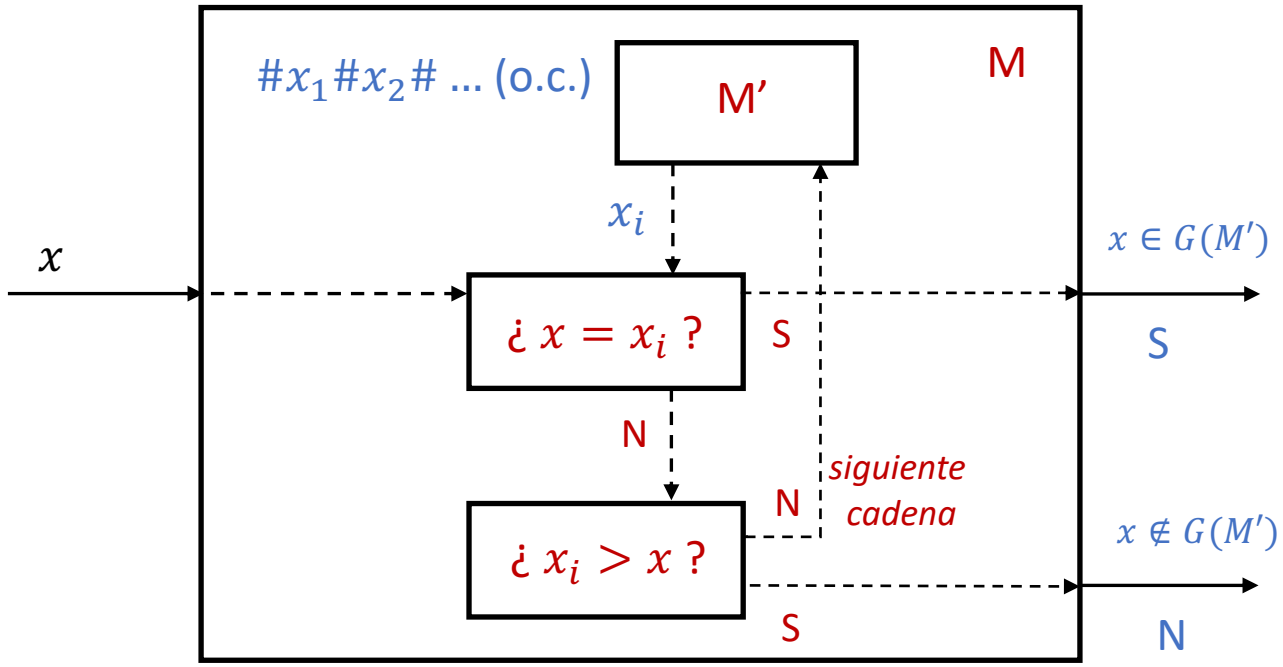


tal que $G(M') = L(M)$

La máquina de Turing como enumerador de lenguajes (máquinas generadoras)

Caracterización de los lenguajes recursivos mediante generadores de Turing

Un esquema para la demostración del anterior enunciado es el siguiente



La máquina M pone a funcionar el generador de Turing M' y, cada vez que éste genera una cadena en su cinta de salida, compara la cadena generada con su cadena de entrada x. Si ambas cadenas coinciden entonces M para y acepta su cadena de entrada ya que pertenece al lenguaje $G(M')$. En caso de que las cadenas no coincidan, entonces se comprueba si la cadena generada es mayor en orden canónico que la cadena de entrada. En caso afirmativo, la cadena no se acepta ya que existe la certeza de que ya no aparecerá en la generación de M' ya que se sigue el orden canónico. Si no fuera este el caso M cede el control a M' a la espera de que vuelva a generar otra cadena en su cinta de salida.

En definitiva, M sólo acepta las cadenas generadas por M' y, por lo tanto, $L(M) = G(M')$.

Obsérvese que este esquema pudiera no ser efectivo si el lenguaje generado por M' fuera finito y la cadena de entrada fuera mayor que cualquiera perteneciente al lenguaje. En este caso, podemos mantener que $G(M')$ es recursivo por ser finito.

Propiedades de los lenguajes recursivamente enumerables y de los lenguajes recursivos

Dada una clase de lenguajes $\mathcal{L}$ una propiedad hace referencia a cualquier operación que podamos definir sobre los lenguajes de la clase y que proporciona como resultado un lenguaje. Diremos que la operación es n-aria si para su aplicación es necesario contar con n lenguajes de partida.

Una operación P es de cierre para la clase de lenguajes $\mathcal{L}$ si el resultado de su aplicación sobre lenguajes de $\mathcal{L}$ tomados arbitrariamente es siempre un lenguaje de la clase $\mathcal{L}$. De forma equivalente, diremos que P es una propiedad de cierre para $\mathcal{L}$ o, alternativamente, diremos que $\mathcal{L}$ es cerrada bajo P. Basta con un caso de aplicación de P donde el resultado no pertenezca a $\mathcal{L}$ para considerar que P no es una propiedad de cierre.

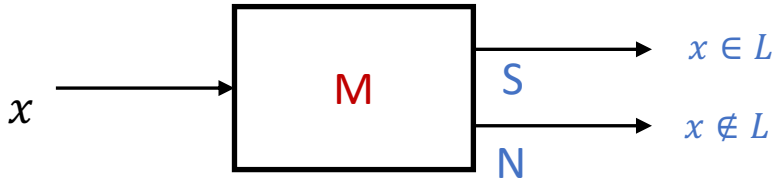
Una operación P de cierre para $\mathcal{L}$ diremos que es efectiva si existe un algoritmo que nos permite construir un modelo de aceptación ó generación del lenguaje resultado de la operación. En el caso de los lenguajes recursivamente enumerables el algoritmo construiría máquinas de Turing aceptoras o generadores de Turing que aceptasen o generasen los lenguajes obtenidos por la aplicación de la operación. En el caso de los lenguajes recursivos además se debería garantizar la parada en las máquinas aceptoras o el orden canónico en los generadores de Turing.

Propiedades de los lenguajes recursivamente enumerables y de los lenguajes recursivos

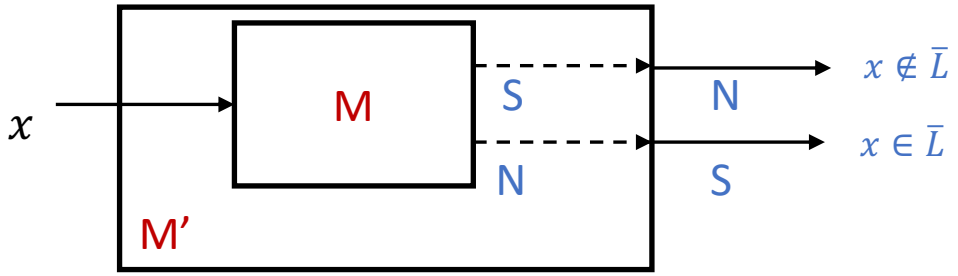
En esta sección veremos algunas de las propiedades más inmediatas tanto de los lenguajes recursivos como recursivamente enumerables. Algunas de esta propiedades son especialmente relevantes cuando se aplican a cuestiones, representadas mediante lenguajes, en el marco de la decidibilidad.

Proposición: Los lenguajes recursivos son cerrados para el complemento.

Supongamos que el lenguaje recursivo L es aceptado por la máquina M con el siguiente esquema



Entonces el siguiente esquema para una máquina M' que siempre para y acepta $\bar{L}$ sería el siguiente

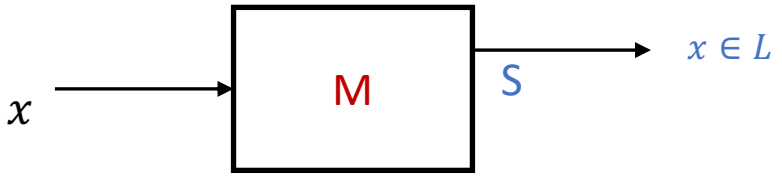


Obsérvese que M' siempre finaliza su computación dado que M siempre finaliza. Por lo tanto podemos concluir que el lenguaje $\bar{L}$ es recursivo siempre que L lo sea y, en consecuencia el complemento es una propiedad de cierre para la clase de los lenguajes recursivos.

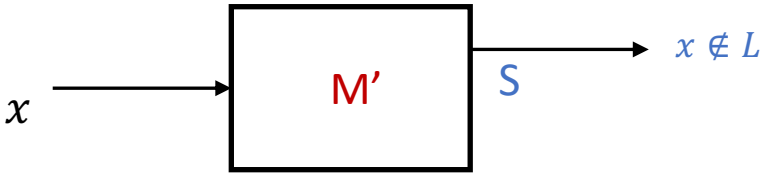
Propiedades de los lenguajes recursivamente enumerables y de los lenguajes recursivos

Proposición: Los lenguajes recursivamente enumerables no son cerrados para el complemento.

Consideremos el lenguaje recursivamente enumerable pero no recursivo L que es aceptado por la máquina M con el siguiente esquema



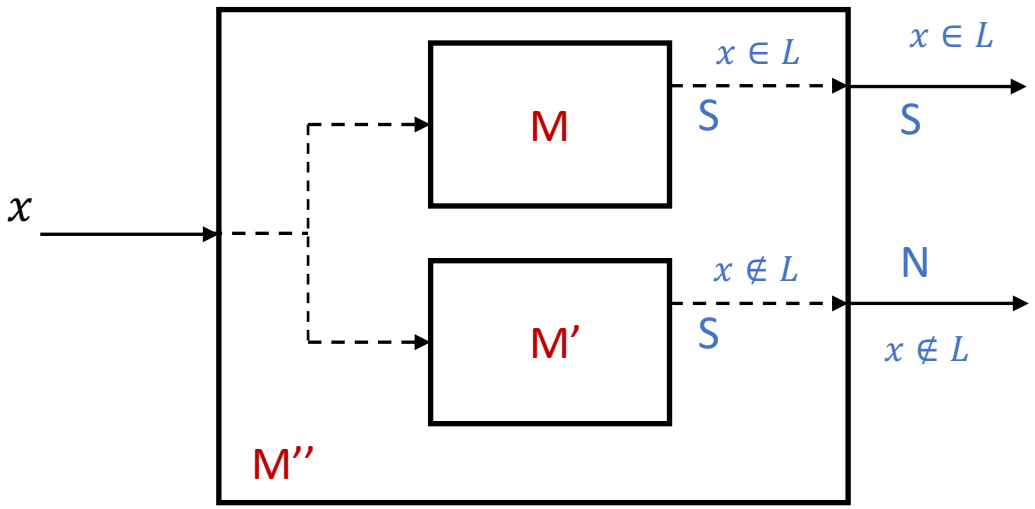
Supongamos que $\bar{L}$ también fuera recursivamente enumerable. En consecuencia debe existir una máquina M' con el siguiente esquema



Propiedades de los lenguajes recursivamente enumerables y de los lenguajes recursivos

Proposición: Los lenguajes recursivamente enumerables no son cerrados para el complemento.

A partir de M y M' podemos construir una máquina M'' con el siguiente esquema



Obsérvese que M'' acepta exactamente L. Además, M'' siempre finaliza su computación ya que ésta depende de la computación de M y M'. En cuanto una de las dos máquinas finaliza aceptando la cadena, entonces M'' también finaliza.

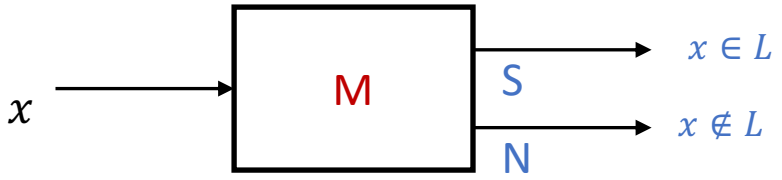
En consecuencia M'' es una máquina que acepta L y siempre finaliza su computación, por lo que podemos concluir que L sería un lenguaje recursivo. La anterior conclusión es una contradicción con nuestra afirmación de partida (L no es recursivo) y, por lo tanto la máquina M' no puede existir.

Como conclusión de la anterior, podemos afirmar que el complemento no es una propiedad de cierre para la clase de los lenguajes recursivamente enumerables.

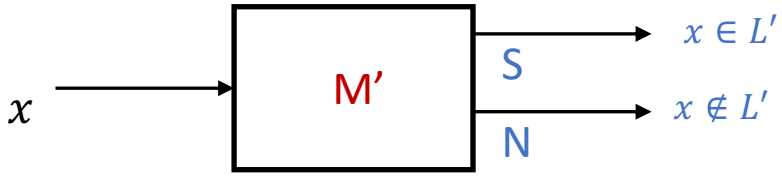
Propiedades de los lenguajes recursivamente enumerables y de los lenguajes recursivos

Proposición: Los lenguajes recursivos son cerrados para la unión y la intersección

Consideremos el lenguaje recursivo L que es aceptado por la máquina M con el siguiente esquema



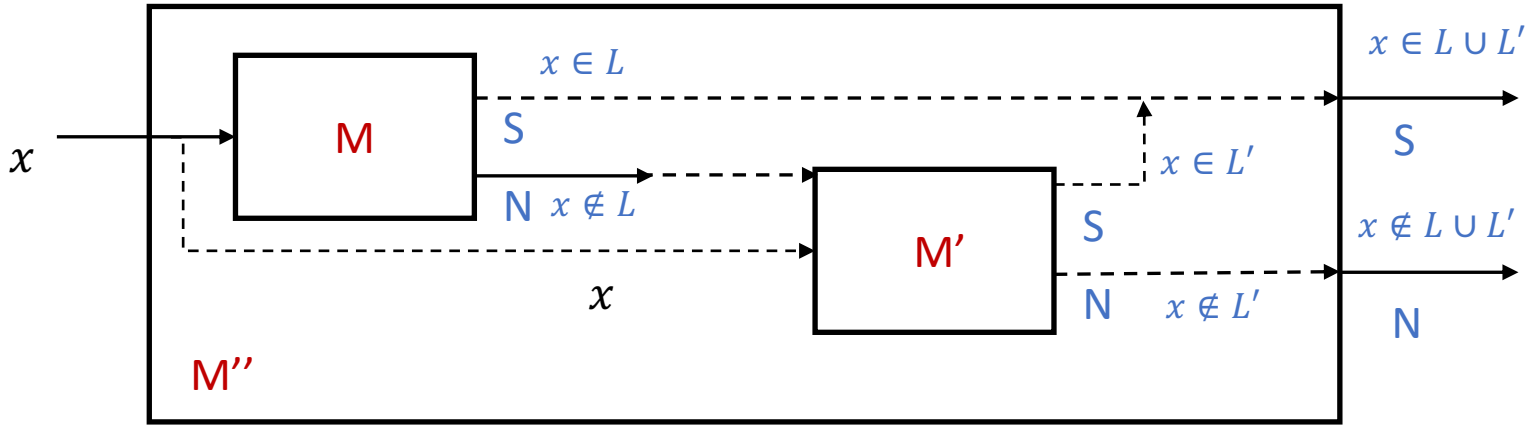
De igual forma, consideremos el lenguaje recursivo L' aceptado por la máquina M' con el siguiente esquema



Propiedades de los lenguajes recursivamente enumerables y de los lenguajes recursivos

Proposición: Los lenguajes recursivos son cerrados para la unión y la intersección

A partir de M y M' podemos construir una máquina M'' con el siguiente esquema



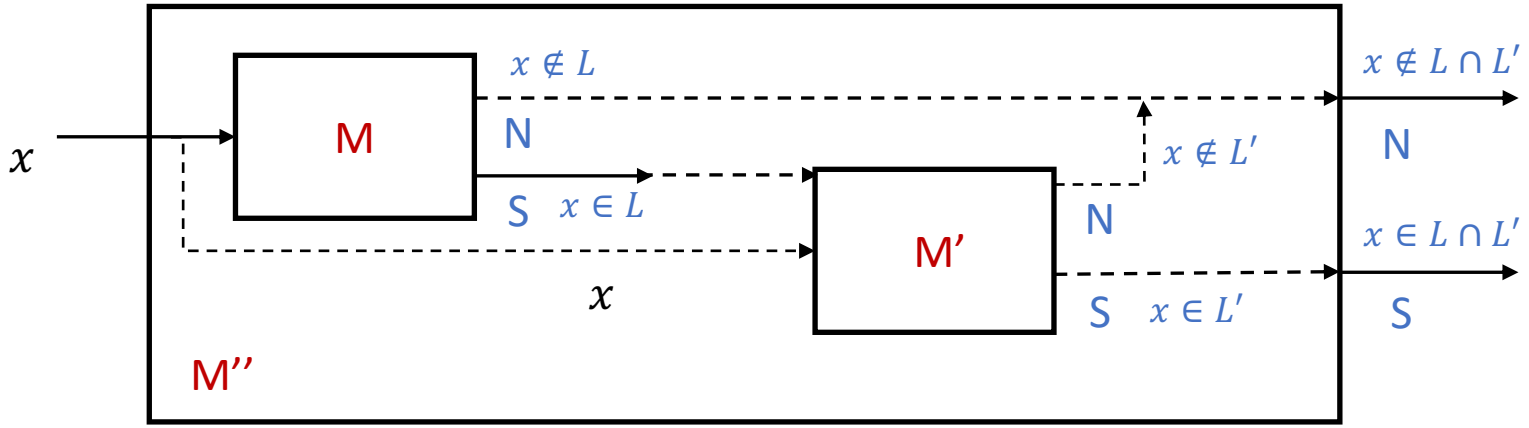
Dada una cadena de entrada x la máquina M'' simula a la máquina M y si M para aceptando x entonces M'' también para y acepta. En el caso que M pare y no acepte x, entonces M'' simula a la máquina M' dándole como entrada la misma cadena x. Si M' para y acepta x, entonces M'' para y acepta. Si M' para y no acepta x, entonces M'' también para y rechaza.

Obsérvese que M'' acepta el lenguaje $L \cup L'$ y garantiza siempre la finalización de su computación. En consecuencia $L \cup L'$ es recursivo y, por lo tanto, la unión es una operación de cierre para los lenguajes recursivos.

Propiedades de los lenguajes recursivamente enumerables y de los lenguajes recursivos

Proposición: Los lenguajes recursivos son cerrados para la unión y la intersección

También a partir de M y M' podemos construir una máquina M'' con el siguiente esquema



Dada una cadena de entrada x la máquina M'' simula a la máquina M y si M para rechazando x entonces M'' también para y rechaza. En el caso que M pare y acepte x, entonces M'' simula a la máquina M' dándole como entrada la misma cadena x. Si M' para y acepta x, entonces M'' para y acepta. Si M' para y no acepta x, entonces M'' también para y rechaza.

Obsérvese que M'' acepta el lenguaje $L \cap L'$ y garantiza siempre la finalización de su computación. En consecuencia $L \cap L'$ es recursivo y, por lo tanto, la intersección es una operación de cierre para los lenguajes recursivos.

Propiedades de los lenguajes recursivamente enumerables y de los lenguajes recursivos

Proposición: Los lenguajes recursivamente enumerables son cerrados para la unión y la intersección.

Proposición: Los lenguajes recursivos son cerrados para la concatenación, clausura, clausura positiva y reverso.

Proposición: Los lenguajes recursivamente enumerables son cerrados para la concatenación, clausura, clausura positiva y reverso

Proposición: Los lenguajes recursivos no son cerrados para homomorfismos.

Proposición: Sea un homomorfismo $h: \Delta^* \longrightarrow \Gamma^*$, de forma que $(\forall a \in \Delta) h(a) \neq \lambda$. Los lenguajes recursivos son cerrados para este tipo de homomorfismos

Propiedades de los lenguajes recursivamente enumerables y de los lenguajes recursivos

Proposición: Los lenguajes recursivos son cerrados para homomorfismos inversos.

Proposición: Los lenguajes recursivamente enumerables son cerrados para homomorfismos y homomorfismo inversos.

Proposición: Si un lenguaje y su complemento son recursivamente enumerables, entonces ambos son recursivos.

Observación

Un lenguaje y su complemento pueden estar en uno de los tres siguientes casos:

- Ambos son recursivamente enumerables, equivalentemente: ambos son recursivos.
- Uno es recursivamente enumerable y el otro no.
- Ambos no son recursivamente enumerables.