

Computabilidad y Complejidad

Tema 2: La máquina de Turing. Lenguajes recursivos. Lenguajes recursivamente enumerables. Funciones Turing-computables.

Índice:

1. Generalidades. Modelo básico de máquina de Turing.
2. Lenguajes reconocibles mediante máquinas de Turing. Lenguajes recursivos y recursivamente enumerables.
3. Funciones computables mediante máquinas de Turing.
4. Hipótesis de Church.
5. Técnicas para la construcción de máquinas de Turing.
6. Máquinas de Turing modificadas.
7. La máquina de Turing como enumerador de lenguajes.
8. Propiedades de los lenguajes recursivos y de los lenguajes recursivamente enumerables.

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.)
- ♦ Teoría de la computación (Brookshear, J. Glenn)

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.

El modelo básico (posteriormente estudiaremos otros) de máquina de Turing 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 (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.

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

$$f: Q \times \Gamma \longrightarrow Q \times \Gamma \times \{R, L\}$$

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.
-  $f(q, a) = (p, b, L)$, 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 .

La máquina de Turing con una entrada $x \in \Sigma^*$ 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 $|x|$ celdillas y quedando el resto con el símbolo B . El cabezal de la cinta se sitúa en la primera celdilla leyendo su símbolo asociado, a_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: $f(q_0, a_0)$, 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.

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:

$$\Gamma \cap Q = \emptyset$$

Σ = conjunto de símbolos no vacíos (de entrada)

Γ = conjunto de símbolos de cinta no vacíos

Q = conjunto de estados no vacíos

f = función de transición definida como:

$$Q \times \Sigma \rightarrow Q \times \Gamma \times (R/L)$$

B = símbolo que representa al "Blanco"

q_0 = estado inicial

F = conjunto de estados finales (de aceptación y no aceptación)

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.

Descripciones instantáneas

Para estudiar las computaciones realizadas por las máquinas de Turing vamos a introducir el concepto de descripción instantánea, 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_i 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

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

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

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

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

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

y de

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

se transita a

$$a_1 a_2 \dots a_{i-1} b p$$

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

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

y de

$$a_1a_2\ldots a_{i-1}q$$

se transita a

- ◆ $a_1a_2\ldots pa_{i-1}b$ si $b \neq B$,
- ◆ $a_1a_2\ldots pa_{i-1}$ si $b = B$ y $a_{i-1} \neq B$,
- ◆ $a_1a_2\ldots p$ si $b = B$ y $a_{i-1} = B$

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

$$q_0x$$

en particular para $x = \lambda$ es

$$q_0$$

Observación: Las descripciones instantáneas son palabras del lenguaje $\Gamma^*Q\Gamma^*$.

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

La relación binaria de alcanzabilidad directa entre descripciones instantáneas en una máquina de Turing $M = (\Sigma, \Gamma, Q, f, B, q_0, F)$

$$|-_M \subseteq \Gamma^* Q \Gamma^* \times \Gamma^* Q \Gamma^*$$

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

Ejemplo: Si $f(q, a_i) = (p, b, R)$:

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

Siempre que no exista posibilidad de confusión escribiremos $|-$ en lugar de $|-_M$.

De la misma manera que en el entorno gramatical a partir de la relación \Rightarrow , definimos las relaciones \Rightarrow^n , \Rightarrow^+ y \Rightarrow^* ; definimos ahora a partir de la relación $|-$, las relaciones $|-^n$, $|-^+$ y $|-^*$.

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

$$L(M) = \{x \in \Sigma^* \mid q_0 x \vdash^* \alpha p \beta, p \in F\}$$

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

Observación: Nótese que 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.

Observación: En lo que sigue 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.

Observación: Nótese que con este convenio dada una máquina de Turing arbitraria existe una máquina de Turing equivalente con a lo sumo un estado final.

Observación: 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

$$f' : \{q_0\} \times (\Sigma \cup \{B\}) \longrightarrow \{q_0\} \times (\Sigma \cup \{B\}) \times \{R, L\}$$

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

Ejercicio: Diseñe una máquina de Turing que reconozca el lenguaje $\{a^n b^n \mid n \geq 0\}$.

Lenguajes reconocibles mediante máquinas de Turing: Lenguajes recursivos y lenguajes recursivamente enumerables

Los lenguajes reconocibles mediante máquinas de Turing constituyen la clase de los lenguajes computables y coinciden, en la jerarquía de Chomsky, con la clase \mathcal{L}_0 .

En la terminología de la teoría de la computabilidad desarrollada a partir del modelo de la máquina de Turing esta clase se denomina de los lenguajes Recursivamente Enumerables y se denotará mediante \mathcal{L}_{REN} .

Así un lenguaje $L \subseteq \Sigma^*$ pertenece \mathcal{L}_{REN} si y sólo si existe una máquina de Turing $M = (\Sigma, \Gamma, Q, f, B, q_0, F)$, tal que $L = L(M)$, es decir, que opera del siguiente modo:

Para cada $x \in \Sigma^*$:

- La máquina M al procesar x se detiene después de un número finito de transiciones $(M(x) \downarrow)$ en un estado de aceptación, es decir, **aceptando**.
- La máquina M al procesar x se detiene después de un número finito de transiciones $(M(x) \downarrow)$ en un estado que no es de aceptación, es decir, **rechazando**.
- La máquina M al procesar x **no se detiene** $(M(x) \uparrow)$.

La posibilidad de que la máquina pueda comportarse como indica el último punto, es decir, sin detenerse, introduce una situación de incertidumbre indeseable respecto a un posible resultado final y, por tanto, con respecto a la computación en su totalidad.

Desafortunadamente la posibilidad de no detención no puede, en general, eliminarse. Aunque sí puede excluirse a costa de restringir la familia de lenguajes que de este modo pueden reconocerse, dando lugar a la clase de los lenguajes recursivos denotados por \mathcal{L}_R .

Así un lenguaje $L \subseteq \Sigma^*$ pertenece \mathcal{L}_R si y sólo si existe una máquina de Turing $M = (\Sigma, \Gamma, Q, f, B, q_0, F)$, tal que $L = L(M)$, operando del siguiente modo:

Para cada $x \in \Sigma^*$:

- La máquina M al procesar x se detiene después de un número finito de transiciones ($M(x) \downarrow$) en un estado de aceptación, es decir, **aceptando**.
- La máquina M al procesar x se detiene después de un número finito de transiciones ($M(x) \downarrow$) en un estado que no es de aceptación, es decir, **rechazando**.

De este modo

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

Ejemplo: El lenguaje incontextual $\{a^n b^n \mid n \geq 0\}$ es recursivo.

Más adelante veremos:

→ Lenguajes que no son recursivamente enumerables y que en consecuencia no pueden ser reconocidos por las máquinas de Turing y, equivalentemente, tampoco pueden ser generados mediante gramáticas.

→ Lenguajes que son recursivamente enumerables pero no son recursivos.

Todos los lenguajes recursivos son también recursivamente enumerables.

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

$$g: N^m \longrightarrow N^k$$

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

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

En este contexto, la máquina de Turing presenta la forma

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

La m -tupla, n_1, \dots, n_m , se le suministra a la máquina como una palabra de $\{0, 1\}^*$ según el formato

$$0^{n_1}1 \dots 10^{n_m}$$

la máquina comienza la computación y se detiene en la descripción instantánea

$$\alpha p 0^{i_1}1 \dots 10^{i_k}B\beta, \quad p \in Q, \quad \alpha, \beta \in \Gamma^*$$

diremos que el resultado de la computación es

$$0^{i_1}1 \dots 10^{i_k} \text{ representación de } (i_1, \dots, i_k)$$

en cualquier otro caso diremos que la función computada está indefinida para los argumentos: n_1, \dots, n_m .

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

Ejercicio: Diseñe una máquina de Turing que compute la siguiente función g (diferencia propia).

$$g: \mathbb{N}^2 \longrightarrow \mathbb{N}$$

estando definida como sigue:

$$g(n, m) = \begin{cases} n-m, & \text{si } n > m \\ 0, & \text{en otro caso} \end{cases}$$

Ejercicio: Bosqueje una máquina de Turing que compute la función $g(n, m) = nm$.

Ejercicio: Bosqueje una máquina de Turing que compute la función $g(n, m) = n/m$, donde $/$ es el operador de división entera.

Ejercicio: Bosqueje una máquina de Turing que compute la función $g(n, m) = n \% m$, donde $\%$ es el resto de la división entera.

Ejercicio: Bosqueje una máquina de Turing que compute la siguiente función g .

$$g: \mathbb{N} \longrightarrow \mathbb{N}$$

estando definida como sigue:

$$g(n) = \begin{cases} \lfloor \lg_2 n \rfloor, & \text{si } n \geq 1 \\ 0, & \text{en otro caso} \end{cases}$$

De modo análogo se definen las funciones alfabéticas Turing-computables. En este caso las funciones alfabéticas son sintácticamente de la forma

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

Donde Δ y E son dos alfabetos.

Ejercicio: Bosqueje una máquina de Turing que compute el homomorfismo

$$h: \{0,1\}^* \longrightarrow \{0,1\}^*$$

con $h(0) = 00$ y $h(1) = 11$.

Ejercicio: Bosqueje una máquina de Turing que compute la función

$$f: \{a,b\}^* \longrightarrow \{a,b\}^*$$

de modo que

- $f(\lambda) = \lambda$
- $\forall e \in \{a,b\}: f(e) = e$
- $\forall e, e' \in \{a,b\}, \forall x \in \{a,b\}^*: f(exe') = e'f(x)e$

Claramente toda función numérica puede verse como un caso particular de función alfabética.

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

$$\begin{array}{ccc}
 & g & \\
 \Delta^* & \longrightarrow & E^* \\
 \text{cod} \downarrow & & \uparrow \text{dec} \\
 N & \xrightarrow{g'} & N
 \end{array}$$

Por tanto, en lo referente a cuestiones computables, podemos, sin pérdida de generalidad, ceñirnos a la funciones computables numéricas de la forma

$$N \longrightarrow N$$

Equivalentemente, también podemos restringirnos a funciones computables de $\{0\}^*$ en $\{0\}^*$.

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 $L \subseteq \Sigma^*$. $L \in \mathcal{L}_{\text{REN}}$ si y sólo si existe una función $\eta_{L, \Sigma}$ Turing computable.

A cada lenguaje $L \subseteq \Sigma^*$ pueden asociársele funciones totales definidas 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 $L \subseteq \Sigma^*$. $L \in \mathcal{L}_R$ si y sólo si existe una función $\varphi_{L,\Sigma}$ Turing computable.

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

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.

Técnicas para la 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.

Parametrización de los 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

$$[p_1, \dots, p_n], \quad p_i \in P_i$$

donde $P_i, i=1, \dots, n$, es el conjunto finito de valores que puede adoptar el parámetro p_i .

Así, en la máquina de Turing: $Q \subseteq \{[p_1, \dots, p_n] / p_i \in P_i, i=1, \dots, n\}$.

En clase veremos ejemplos de su uso.

Ejercicio: Bosqueje una máquina de Turing que reconozca el lenguaje $\{xcx/x \in \{0,1\}^*\}$.

Cinta con varias bandas o sectores

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 Γ_i , $i=1, \dots, n$. El símbolo B está en cada Γ_i .

El alfabeto de la cinta Γ está, de este modo, definido como

$$\Gamma = \Gamma_1 \times \dots \times \Gamma_n$$

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, $B = [B, \dots, B]$.

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

$$\Sigma_i \subseteq \Gamma_i - \{B\}$$

En clase veremos ejemplos de su uso.

Ejercicio: Bosqueje una máquina de Turing que reconozca el lenguaje $\{xx/x \in \{0,1\}^*\}$.

Desplazamientos en el contenido de la cinta

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 $\alpha = \alpha_1\alpha_2$ ($\alpha_2 \neq \lambda$), y éste se desea transformar en $\alpha = \alpha_1X^k\alpha_2$, 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 .

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 $[q', B, B]$ 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 $A_1 \neq B \neq A_2$
- $f([q, A_1, B], B) = ([q', B, B], A_1, L)$ para $A_1 \neq B$
- $f([q', B, B], A) = ([q', B, B], A, L)$ para $A \neq X$

A continuación la máquina continuará operando de acuerdo al resto de la función de transición.

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. Algunas de estas posibles modificaciones son las que a continuación siguen.

Máquinas de Turing con la cinta infinita en ambos sentidos

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.

De este modo 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.

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 rellenas 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.

Teorema: Si el lenguaje \mathbb{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.

Teorema: Si el lenguaje \mathbb{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.

Máquinas de Turing multicintas

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.

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 $\Sigma_i \subseteq \Gamma_i - \{B\}$. Sobre ella se escribe la entrada 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 sus relaciones asociadas se definen de modo análogo; así como el lenguaje aceptado por la máquina.

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.

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.

Máquinas de Turing indeterministas

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

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

de modo que

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

esto es:

- $f(q, a) = \emptyset, 0$
- $f(q, a) = \{(q_i, b_i, m_i) / i = 1, \dots, n_{q, a}\}$

Así, si

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

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

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

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.

El lenguaje aceptado por este modelo se define igual que en el modelo básico, esto es,

$$L(M) = \{x \in \Sigma^* \mid q_0x \vdash^* \alpha p \beta, p \in F\}$$

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.

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

La máquina de Turing como enumerador de lenguajes

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 \mathbb{B} .

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 estas condiciones, generado por un generador de Turing M se denota por $G(M)$ y se define como

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

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.

Órdenes canónicos

Antes de abordar las caracterizaciones de los lenguajes recursivamente enumerables y de los lenguajes recursivos mediante generadores definiremos la noción de orden canónico sobre las palabras definidas a partir de un alfabeto.

Sea Δ un alfabeto definimos un orden canónico sobre Δ^* como sigue.

Sea primeramente una enumeración arbitraria de sus símbolos

$$a_1, \dots, a_n$$

el orden canónico asociado a las palabras de Δ^* se define de menor a mayor como sigue:

- 1) se ordenan por longitud creciente, y
- 2) para una misma longitud se ordenan por orden lexicográfico a partir de la enumeración previamente establecida.

Ejemplo: Sea $\Delta = \{1, 0\}$ y sea la enumeración: 0,1. El orden canónico asociado es

$$\lambda, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, 110, 111, 0000, \dots$$

Observación: Dado un alfabeto Δ hay $|\Delta|!$ órdenes canónicos diferentes sobre Δ^* .

Ejercicio: Sea $\Delta = \{1, 0\}$ y sea la enumeración: $0, 1$. Bosqueje un generador de Turing que enumere Δ^* en el orden canónico especificado.

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.

Observación: Un lenguaje es generado por un generador de Turing si y sólo si es generado por una gramática.

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 un orden canónico.

Ejercicio resuelto: Sean dos lenguajes $L_1, L_2 \subseteq \Sigma^*$, se define la operación P del modo que sigue:

$$x \in P(L_1, L_2) \Leftrightarrow \exists u, v, w \in \Sigma^* : x = uvw \wedge u \in L_1 \wedge w \in L_2.$$

- I . ¿Es la familia de los lenguajes recursivos cerrada para la operación P?
- II . ¿Es la familia de los lenguajes recursivamente enumerables cerrada para la operación P?

EL lenguaje definido por $P(L_1, L_2)$ se caracteriza porque sus palabras tienen un prefijo en L_1 , un sufijo en L_2 , y el segmento restante en Σ^* . Así $P(L_1, L_2) = L_1 \Sigma^* L_2$.

Demostraremos que las clases de I y II son cerradas para la operación P demostrando que son cerradas para la concatenación, ya que Σ^* es un lenguaje regular y por tanto recursivo y recursivamente enumerable.

I La clase de los lenguajes recursivos es cerrada para la concatenación. Sean L y L' , dos lenguajes recursivos; existen dos máquinas de Turing M y M' que se detienen para cada entrada con $L(M) = L$ y $L(M') = L'$. Seguidamente definimos una máquina de Turing M'' que se detiene para cada entrada y $L(M'') = LL'$. M'' opera como sigue. Tiene una cinta de entrada con dos sectores, en uno de ellos se escribe la palabra de entrada x y en el otro un símbolo de marcaje \checkmark . Este símbolo recorrerá los símbolos de xB (hasta el blanco B inmediatamente a la derecha de x). Para una posición del símbolo \checkmark , en el rango especificado, se factorizará a x como uv , siendo u el prefijo de x que termina en el símbolo anterior al marcado y v el sufijo de x restante.

Si M'' recibe como entrada $x = \lambda$, aplicará la entrada λ a M , si M rechaza M'' también rechaza, si M acepta aplicará λ a M' , aceptando o rechazando según ésta lo haga.

Si M'' recibe como entrada $x \neq \lambda$, pondrá \checkmark sobre el primer símbolo de x y aplicará la entrada u a M , si M acepta aplicará v a M' ; si M' acepta, M'' acepta a x . Si se produce rechazo, bien por parte de M o de M' , se desplaza la marca \checkmark una posición a la derecha y vuelve a repetir el proceso, siempre que la marca está dentro del rango de posiciones. Si la marca se sale del rango de posiciones M'' termina rechazando a x .

Claramente M'' se detiene para cualquier entrada y $L(M'') = LL'$.

II La clase de los lenguajes recursivamente enumerables es cerrada para la concatenación. Sean L y L' , dos lenguajes recursivamente enumerables; existen dos máquinas de Turing M y M' con $L(M) = L$ y $L(M') = L'$. Seguidamente definimos una máquina de Turing M'' con $L(M'') = LL'$. M'' opera como sigue. M'' integra dos máquinas \underline{M} y \underline{M}' , que reciben como entrada una palabra z y un contador j que limita el número máximo de movimientos que cada una puede realizar al procesar z simulando, respectivamente a M y a M' .

Si M'' recibe como entrada $x = \lambda$, entonces en una cinta aparte inicializa el contador j a 1, y aplica λ y j a \underline{M} , si ésta rechaza incrementa el contador j en uno y vuelve a repetir la operación; si acepta, aplica λ y j a \underline{M}' , si ésta acepta M'' acepta, si no incrementa el contador j en uno y se vuelve a repetir la operación completa con el nuevo valor de j .

Si M'' recibe como entrada $x \neq \lambda$, entonces arranca un generador triangular controlado de pares (i,j) , que inicialmente genera el par $(1,1)$ y luego genera los pares sucesivos, de uno en uno, cada vez que recibe la señal de control de avance. Ante un par (i,j) , si $i > |x| + 1$, envía al generador la señal de control, en otro caso factoriza x como uv donde u es el prefijo de x de longitud $i - 1$, y v es el sufijo de x restante; seguidamente aplica u y j a \underline{M} , si ésta rechaza envía al generador la señal de control y vuelve a repetirse la operación con el nuevo par (i,j) ; si acepta, aplica v y j a \underline{M}' , si ésta acepta M'' acepta, si no se envía al generador la señal de control y vuelve a repetirse la operación completa con el nuevo par (i,j) .

Claramente $L(M'') = LL'$: $x \in LL' \Leftrightarrow (\exists u \in L)(\exists v \in L')(x = uv)$; $u \in L \Leftrightarrow u \in L(M) \Leftrightarrow$

$(\exists n)(\underline{M} \text{ acepta a } u \text{ en } n \text{ pasos}); v \in L' \Leftrightarrow v \in L(M') \Leftrightarrow (\exists m)(\underline{M}' \text{ acepta a } v \text{ en } m \text{ pasos}); \text{ sea } k = \max(n,m); x \in LL' \Leftrightarrow \text{ en } M'' \text{ se genera el par } (|u| + 1, k); \text{ como el par } (|u| + 1, k) \text{ se genera en alguna etapa se concluye que: } x \in LL' \Leftrightarrow x \in L(M'').$

Ejercicio resuelto: Sean $L \subseteq \{a,b\}^*$, se define la operación

$$P(L) = \{x \in L / (\exists y \in L) (|y| = |x|_a)\}.$$

I. ¿Es la operación P de cierre en la clase de los lenguajes recursivos?

Si L es recursivo, entonces también lo es $P(L)$. En primer lugar de la observación $P(L) \subseteq L$, se deduce que si L es finito, entonces también lo es $P(L)$, y en consecuencia también es recursivo. Supóngase ahora que L es infinito. Para establecer que $P(L)$ es recursivo definiremos una máquina de Turing M que lo reconozca y que se detenga para cada entrada. Puesto que L es recursivo existe una máquina de Turing M^\bullet que reconoce a L y se detiene para cada entrada y también existe un generador G que genera a L en orden canónico. La máquina M opera del siguiente modo. Para una entrada x la aplica a la máquina M^\bullet , si ésta rechaza M también lo hace. Si M^\bullet acepta, entonces se cuenta el número de veces que en la cadena x aparece el símbolo a , sea este número n , y se arranca el generador G ; cada vez que G genera una palabra se comprueba si su longitud

coincide con n . Si coincide se acepta x . Si no se genera la siguiente palabra. En estas condiciones ocurrirá que, en alguna etapa de la generación, o bien aparece una palabra de longitud n (en cuyo caso se acepta x) o de longitud mayor que n en cuyo caso se rechaza x .

Así la máquina M acepta al lenguaje $P(L)$ y se detiene para cada entrada. Por tanto, $P(L)$ es recursivo.

II. ¿Es la operación P de cierre en la clase de los lenguajes recursivamente enumerables?

Si L es recursivamente enumerable, entonces también lo es $P(L)$. Para establecer que $P(L)$ es recursivamente enumerable definiremos una máquina de Turing M que lo reconozca. Puesto que L es recursivo existe una máquina de Turing M^* que reconoce a L y también existe un generador G que genera a L . La máquina M opera del siguiente modo. Para una entrada x la aplica a la máquina M^* , si ésta acepta, entonces, y sólo entonces, se cuenta el número de veces que en la cadena x aparece el símbolo a , sea este número n , y se arranca el generador G ; cada vez que G genera una palabra se comprueba si su longitud coincide con n . Si coincide se acepta x . Si no se genera la siguiente palabra.

Así la máquina M acepta al lenguaje $P(L)$.

Por tanto, $P(L)$ es recursivamente enumerable.

Ejercicio resuelto: Sea Σ un alfabeto y $L \subseteq \Sigma^*$, se define

$$P(L) = \{x \in \Sigma^* \mid (\exists y \in \Sigma^*) ((xy \in L) \wedge (|x| = |y|))\}.$$

1. Si L es recursivo ¿lo es también $P(L)$?
2. Si L es recursivamente enumerable ¿lo es también $P(L)$?

1. Si L es recursivo, entonces también lo es $P(L)$; para su demostración construiremos una MT M^\blacksquare que lo reconozca y que se detenga para cada entrada. Puesto que L es recursivo existe una MT M que lo reconoce y que se detiene para cada entrada. La máquina M^\blacksquare opera como sigue. Cuando se le aplica una entrada x , arranca un generador canónico G_x de todas las palabras de longitud $|x|$ de Σ^* , este generador comienza generando la primera palabra en el orden canónico de longitud $|x|$ y seguidamente continuará la generación, de palabra en palabra, cada vez que reciba una señal de control \checkmark . Cuando se le solicite mediante esta señal que genere la siguiente palabra y ya no haya más indicará a la máquina M^\blacksquare que se detenga rechazando. Cada vez que G_x genere una palabra y , ésta se concatenará a x formando la palabra $z = xy$ que se aplicará a la máquina M ; si M acepta, entonces M^\blacksquare se detendrá aceptando. Si M rechaza se enviará a G_x una señal \checkmark . En estas condiciones la máquina M^\blacksquare , por construcción, se detiene para cada entrada y reconoce a $P(L)$.

2. Si L es recursivamente enumerable, entonces $P(L)$ también lo es; para su demostración construiremos un generador de Turing \mathbf{M} que lo genere. Puesto que L es recursivamente enumerable existe un generador de Turing M que lo genera. El generador \mathbf{M} opera como sigue. Comienza arrancando el generador M que actuará, controlado por una señal de continuación \checkmark , de modo similar al generador G_x del apartado anterior. Cada vez que M genera una palabra z ésta, si se puede (longitud par), se divide en dos mitades, y se enumera la primera en la cinta de salida, enviándose en cualquier caso (tanto si se puede dividir como si no) la señal \checkmark al generador M . La división de z , en xy con $|x| = |y|$, puede realizarse utilizando una cinta infinita en ambos sentidos con dos sectores uno para z y el otro para marcar conveniente con dos marcas (\times , \boxtimes), la primera y segunda mitad. (También puede hacerse directamente sin el uso de marcas.) Así, por construcción, $G(\mathbf{M}) = P(L)$.

Propiedades de los lenguajes recursivos y de los lenguajes recursivamente enumerables

En esta sección veremos algunas de las propiedades más inmediatas tanto de los lenguajes recursivos como recursivamente enumerables. Algunas de estas 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.

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

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

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

Proposición: Los lenguajes recursivos son cerrados para la concatenación, potencia^{*}, potencia⁺ y reverso.

Proposición: Los lenguajes recursivamente enumerables son cerrados para la concatenación, potencia^{*}, potencia⁺ y reverso.

Proposición: Los lenguajes recursivos no son cerrados para homomorfismos.

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

Para los no borradores

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.

Lenguaje finito es recursivo y recursivamente enumerable

Lrecursivo y recursivamente enumerable está contenido en Sigma*