Universidad de La Laguna

Grado en Ingeniería Informática

Computabilidad y Algoritmia

Tema 4: Máquinas de Turing

F. de Sande

Curso 2024-2025

Indice

Indice

Introducción

Hasta ahora...

- Lenguajes regulares: autómatas finitos
- Lenguajes independientes del contexto: autómatas de pila
- **Modelos de computación:** reciben una cadena de símbolos de entrada, realizan ciertos movimientos en respuesta a los símbolos de la cadena y proporcionan una respuesta rudimentaria: *sí* o *no*

Introducción

Hasta ahora...

- Lenguajes regulares: autómatas finitos
- Lenguajes independientes del contexto: autómatas de pila
- **Modelos de computación:** reciben una cadena de símbolos de entrada, realizan ciertos movimientos en respuesta a los símbolos de la cadena y proporcionan una respuesta rudimentaria: *sí* o *no*

A partir de ahora...

- Hay lenguajes simples ($L = \{a^n b^n c^n \mid n \geq 0\}$) que no son independientes del contexto, y por ello ni los autómatas finitos ni los autómatas de pila pueden ser considerados modelos generales de computación
- Se estudiará en este tema un modelo de computación más general

Máquina de Turing

Definición

Una Máquina de Turing (MT) es una tupla $M \equiv (Q, \Sigma, \Gamma, q_0, \text{ƀ}, F, \delta)$:

Máquina de Turing

Definición

Una Máquina de Turing (MT) es una tupla $M \equiv (Q, \Sigma, \Gamma, q_0, \text{ß}, F, \delta)$:

- Q es el conjunto de estados (Q finito y $Q \neq \emptyset$)

Máquina de Turing

Definición

Una Máquina de Turing (MT) es una tupla $M \equiv (Q, \Sigma, \Gamma, q_0, \text{ƀ}, F, \delta)$:

- Q es el conjunto de estados (Q finito y $Q \neq \emptyset$)
- Γ es el alfabeto de cinta

Máquina de Turing

Definición

Una Máquina de Turing (MT) es una tupla $M \equiv (Q, \Sigma, \Gamma, q_0, \text{ƀ}, F, \delta)$:

- Q es el conjunto de estados (Q finito y $Q \neq \emptyset$)
- Γ es el alfabeto de cinta
- Σ es el alfabeto de entrada. Generalmente $\Sigma \subseteq \Gamma - \{\text{ƀ}\}$

Máquina de Turing

Definición

Una Máquina de Turing (MT) es una tupla $M \equiv (Q, \Sigma, \Gamma, q_0, \text{ƀ}, F, \delta)$:

- Q es el conjunto de estados (Q finito y $Q \neq \emptyset$)
- Γ es el alfabeto de cinta
- Σ es el alfabeto de entrada. Generalmente $\Sigma \subseteq \Gamma - \{\text{ƀ}\}$
- $q_0 \in Q$ es el estado inicial o de arranque

Máquina de Turing

Definición

Una Máquina de Turing (MT) es una tupla $M \equiv (Q, \Sigma, \Gamma, q_0, \text{ƀ}, F, \delta)$:

- Q es el conjunto de estados (Q finito y $Q \neq \emptyset$)
- Γ es el alfabeto de cinta
- Σ es el alfabeto de entrada. Generalmente $\Sigma \subseteq \Gamma - \{\text{ƀ}\}$
- $q_0 \in Q$ es el estado inicial o de arranque
- $F \subseteq Q$ es el conjunto de estados de aceptación

Máquina de Turing

Definición

Una Máquina de Turing (MT) es una tupla $M \equiv (Q, \Sigma, \Gamma, q_0, \text{ƀ}, F, \delta)$:

- Q es el conjunto de estados (Q finito y $Q \neq \emptyset$)
- Γ es el alfabeto de cinta
- Σ es el alfabeto de entrada. Generalmente $\Sigma \subseteq \Gamma - \{\text{ƀ}\}$
- $q_0 \in Q$ es el estado inicial o de arranque
- $F \subseteq Q$ es el conjunto de estados de aceptación
- ƀ$\in \Gamma$ es el símbolo blanco (ƀ$\notin \Sigma$)

Máquina de Turing

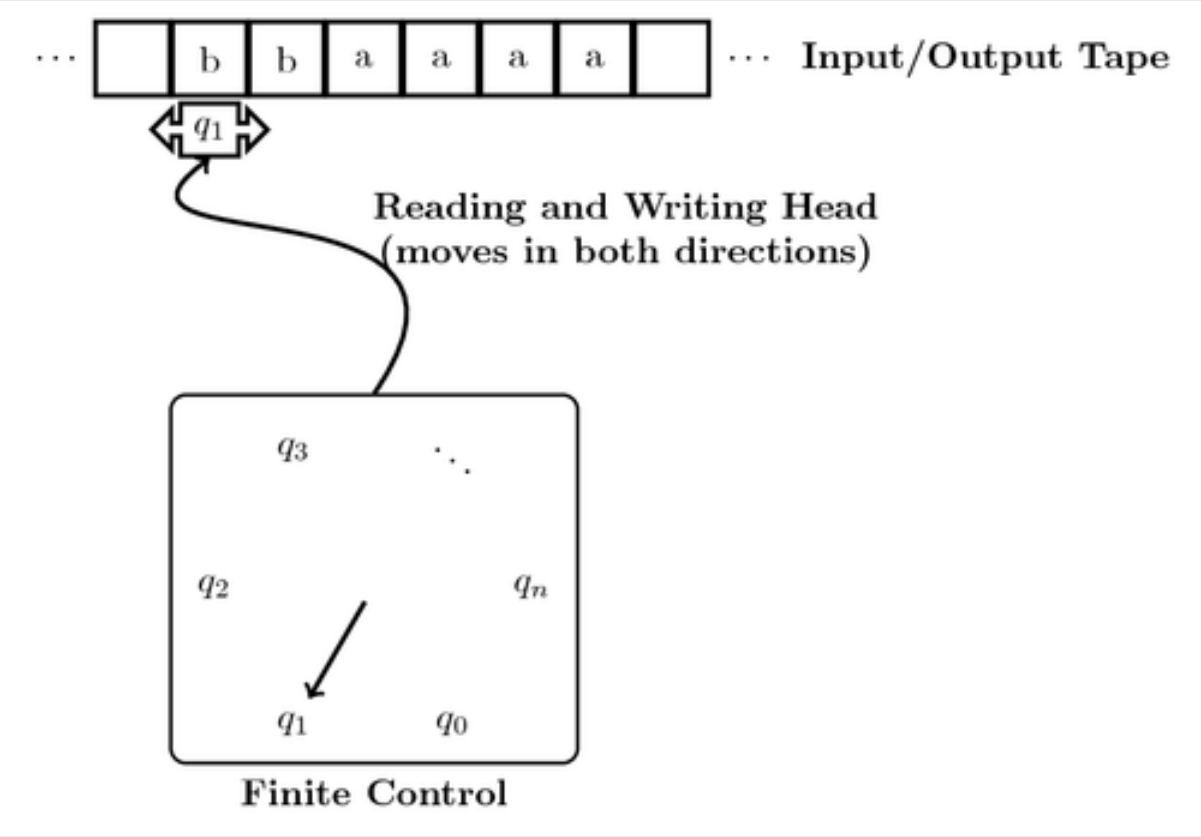
Definición

Una Máquina de Turing (MT) es una tupla $M \equiv (Q, \Sigma, \Gamma, q_0, \text{ƀ}, F, \delta)$:

- Q es el conjunto de estados (Q finito y $Q \neq \emptyset$)
- Γ es el alfabeto de cinta
- Σ es el alfabeto de entrada. Generalmente $\Sigma \subseteq \Gamma - \{\text{ƀ}\}$
- $q_0 \in Q$ es el estado inicial o de arranque
- $F \subseteq Q$ es el conjunto de estados de aceptación
- ƀ$\in \Gamma$ es el símbolo blanco (ƀ$\notin \Sigma$)
- δ es la función de transición:
  $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$
  $(q, a) \to (p, c, X)$
  con $p, q \in Q,\ a, c \in (\Sigma \cup \{\text{ƀ}\}),\ X \in \{L, R\}$

Características

Una MT dispone de una secuencia de celdas de almacenamiento que se extiende infinitamente en ambas direcciones: cinta infinita



Características

- Posee una cabeza de lectura/escritura que puede moverse sobre la cinta y por cada movimiento lee y escribe un símbolo
- Cada celda permite almacenar un único símbolo
- El valor inicial de todas las celdas de la cinta es es símbolo blanco (ƀ)
- A los contenidos de las celdas se puede acceder en cualquier orden
- Las transiciones dependen únicamente del estado actual y del contenido de la celda sobre la que se encuentre la cabeza de lectura/escritura

Funcionamiento

Consideremos la máquina de Turing definida mediante:

- $Q = \{q_1, q_2\}$
- $\Sigma = \{a, b\}$
- $\Gamma = \{a, b, \text{ƀ}\}$
- $F = \{q_2\}$
- $s = q_1$
- δ:

$$\delta(q_1, a) = (q_1, a, R)$$
$$\delta(q_1, b) = (q_1, a, R)$$
$$\delta(q_1, \text{ƀ}) = (q_2, \text{ƀ}, L)$$

Funcionamiento

Considérese la máquina de Turing definida mediante:

$$\delta(q_1, a) = (q_1, a, R)$$
$$\delta(q_1, b) = (q_1, a, R)$$
$$\delta(q_1, \text{b}) = (q_2, \text{b}, L)$$

Funcionamiento

Considérese la máquina de Turing definida mediante:

$\delta(q_1, a) = (q_1, a, R)$
$\delta(q_1, b) = (q_1, a, R)$
$\delta(q_1, \text{ƀ}) = (q_2, \text{ƀ}, L)$

El funcionamiento de la máquina es el siguiente:

- La máquina comienza sus operaciones en el estado q_1
- Si el contenido de la celda sobre la que se encuentra la cabeza de lectura/escritura es a, se aplica la transición $\delta(q_1, a) = (q_1, a, R)$:
  - La MT sobreescribirá el símbolo a que está en la cinta con otra a
  - La cabeza de lectura/escritura se moverá una posición a la derecha
  - La MT permanecerá en el estado q_1

Funcionamiento

Considérese la máquina de Turing definida mediante:

$$\delta(q_1, a) = (q_1, a, R)$$
$$\delta(q_1, b) = (q_1, a, R)$$
$$\delta(q_1, \text{ƀ}) = (q_2, \text{ƀ}, L)$$

Funcionamiento

Considérese la máquina de Turing definida mediante:

$\delta(q_1, a) = (q_1, a, R)$
$\delta(q_1, b) = (q_1, a, R)$
$\delta(q_1, \text{ƀ}) = (q_2, \text{ƀ}, L)$

El funcionamiento de la máquina es el siguiente:

- Cualquiera de las siguientes aes que haya en la cinta no se cambiará, sin embargo, las bes serán sustituidas por aes: $\delta(q_1, b) = (q_1, a, R)$
- Si la maquina de Turing encuentra un blanco (símbolo ƀ), se moverá una celda hacia la izquierda y pasará al estado final, q_2: $\delta(q_1, \text{ƀ}) = (q_2, \text{ƀ}, L)$
- Puesto que no hay ninguna transición definida desde el estado q_2, la MT detendrá su ejecución una vez que esté en ese estado

Descripciones instantáneas

Cualquier configuración de una MT viene determinada por:

- El estado actual
- El contenido de la cinta
- La posición de la cabeza de lectura/escritura sobre la cinta

Descripciones instantáneas

Cualquier configuración de una MT viene determinada por:

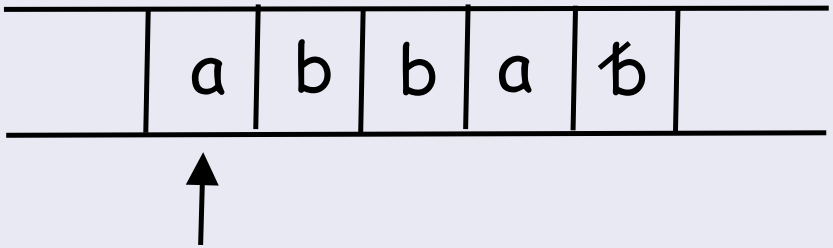
- El estado actual
- El contenido de la cinta
- La posición de la cabeza de lectura/escritura sobre la cinta

Se usarán dos notaciones para representar las configuraciones:

1. $(q_i, w_1 \underline{a} w_2)$:
   - q_i es el estado actual
   - $w_1 \in \Gamma^*$ es la cadena a la izquierda de la cabeza de L/E
   - $w_2 \in \Gamma^*$ es la cadena a la derecha de la cabeza de L/E
   - $a \in \Gamma$ es el símbolo sobre el que se encuentra la cabeza de L/E
2. $a_1 a_2 ... a_{k-1}\ q_i\ a_k ... a_n$:
   - $a_1 a_2 ... a_{k-1} = w_1$
   - $a_k = a$
   - $a_{k+1} a_{k+2} ... a_n = w_2$

Descripciones instantáneas

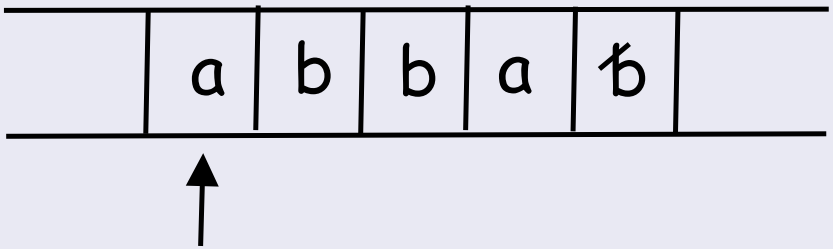
Ejemplo:



- Notación 1:
  $(q_1, \underline{a}bba) \vdash (q_1, a\underline{b}ba) \vdash (q_1, aa\underline{b}a) \vdash (q_1, aaa\underline{a}) \vdash (q_1, aaaa\underline{ƀ}) \vdash (q_2, aaa\underline{a})$
- Notación 2:
  $q_1abba \vdash aq_1bba \vdash aaq_1ba \vdash aaaq_1a \vdash aaaaq_1ƀ \vdash aaaq_2a$

Descripciones instantáneas

Ejemplo:



- Notación 1:
  $(q_1, \underline{a}bba) \vdash (q_1, a\underline{b}ba) \vdash (q_1, aa\underline{b}a) \vdash (q_1, aaa\underline{a}) \vdash (q_1, aaaa\underline{b}) \vdash (q_2, aaa\underline{a})$
- Notación 2:
  $q_1abba \vdash aq_1bba \vdash aaq_1ba \vdash aaaq_1a \vdash aaaaq_1b \vdash aaaq_2a$

- $\vdash$ denota el paso de una configuración a otra
- $\vdash^*$ y $\vdash^+$ tienen el significado usual de "cero o más" y "uno o más" pasos de cómputo

Descripciones instantáneas

Ejemplo:

$Q = \{q_1, q_2, q_3\}$
$\Sigma = \{a, b\}$
$\Gamma = \{a, b, \text{ƀ}\}$
$F = \{q_3\}$
$q_0 = q_1$

$\delta(q_1, a) = (q_1, a, L)$
$\delta(q_1, b) = (q_1, b, L)$
$\delta(q_1, \text{ƀ}) = (q_2, \text{ƀ}, R)$
$\delta(q_2, a) = (q_3, a, L)$
$\delta(q_2, b) = (q_3, b, L)$
$\delta(q_2, \text{ƀ}) = (q_3, \text{ƀ}, R)$

Descripciones instantáneas

Ejemplo:

$Q = \{q_1, q_2, q_3\}$
$\Sigma = \{a, b\}$
$\Gamma = \{a, b, \text{ƀ}\}$
$F = \{q_3\}$
$q_0 = q_1$

$\delta(q_1, a) = (q_1, a, L)$
$\delta(q_1, b) = (q_1, b, L)$
$\delta(q_1, \text{ƀ}) = (q_2, \text{ƀ}, R)$
$\delta(q_2, a) = (q_3, a, L)$
$\delta(q_2, b) = (q_3, b, L)$
$\delta(q_2, \text{ƀ}) = (q_3, \text{ƀ}, R)$

Esta MT examina su cinta hacia la izquierda hasta hallar el primer ƀ. Entonces para y se coloca sobre el ƀ:

$(q_1, \text{ƀ}aab\underline{a}bb\text{ƀ}) \vdash (q_1, \text{ƀ}aa\underline{b}abb\text{ƀ}) \vdash (q_1, \text{ƀ}a\underline{a}babb\text{ƀ}) \vdash (q_1, \text{ƀ}\underline{a}ababb\text{ƀ}) \vdash$
$(q_1, \underline{\text{ƀ}}aababb\text{ƀ}) \vdash (q_2, \text{ƀ}\underline{a}ababb\text{ƀ}) \vdash (q_3, \underline{\text{ƀ}}aababb\text{ƀ})$

Lenguaje aceptado por una MT

Sea una MT $M \equiv (Q, \Sigma, \Gamma, q_0, \text{ƀ}, F, \delta)$:

El lenguaje aceptado por M se denota como $L(M)$ y se define como:

$$L(M) = \{w \in \Sigma^* \mid q_0 w \vdash^* w_1 p w_2 \text{ para } p \in F, w_i \in \Gamma^*\}$$

- Una cadena $w \in \Sigma^*$ es aceptada si M alcanza un estado de aceptación p, y para

Lenguaje aceptado por una MT

Sea una MT $M \equiv (Q, \Sigma, \Gamma, q_0, \text{ƀ}, F, \delta)$:

El lenguaje aceptado por M se denota como $L(M)$ y se define como:

$$L(M) = \{w \in \Sigma^* \mid q_0 w \vdash^* w_1 p w_2 \text{ para } p \in F, w_i \in \Gamma^*\}$$

- Una cadena $w \in \Sigma^*$ es aceptada si M alcanza un estado de aceptación p, y para

Nota: Se ha supuesto que una MT siempre para (detiene su ejecución) al alcanzar un estado de aceptación puesto que no se definen transiciones en los estados $p \in F$

Parada de una máquina de Turing

Considérese una MT $M \equiv (Q, \Sigma, \Gamma, q_0, \text{ƀ}, F, \delta)$:

- Cuando $\delta(q, a)$ no está definida y la configuración de la MT es $(q, w_1 a w_2)$ es imposible que la MT cambie la configuración
- En estas circunstancias, se dice que la MT está **parada**
- Una MT puede estar parada en un estado $q \in F$ o en un estado $q \notin F$
- Para simplificar se asumirá que no se definirán transiciones para los estados de F, y consecuentemente
  - La MT se parará siempre que llegue a un estado de aceptación
- La secuencia de movimientos que conducen a una MT desde su estado de arranque a una configuración de parada se llama **computación**

Máquina de Turing que nunca para

Considérese una MT $M \equiv (Q, \Sigma, \Gamma, q_0, \text{ƀ}, F, \delta)$:

$Q = \{q_1, q_2\}$
$\Sigma = \{a, b\}$
$\Gamma = \{a, b, \text{ƀ}\}$
$F = \emptyset$
$q_0 = q_1$

$\delta(q_1, a) = (q_2, a, R)$
$\delta(q_1, b) = (q_2, b, R)$
$\delta(q_1, \text{ƀ}) = (q_2, \text{ƀ}, R)$
$\delta(q_2, a) = (q_1, a, L)$
$\delta(q_2, b) = (q_1, b, L)$
$\delta(q_2, \text{ƀ}) = (q_1, \text{ƀ}, L)$

Máquina de Turing que nunca para

Considérese una MT $M \equiv (Q, \Sigma, \Gamma, q_0, \text{ƀ}, F, \delta)$:

$Q = \{q_1, q_2\}$
$\Sigma = \{a, b\}$
$\Gamma = \{a, b, \text{ƀ}\}$
$F = \emptyset$
$q_0 = q_1$

$\delta(q_1, a) = (q_2, a, R)$
$\delta(q_1, b) = (q_2, b, R)$
$\delta(q_1, \text{ƀ}) = (q_2, \text{ƀ}, R)$
$\delta(q_2, a) = (q_1, a, L)$
$\delta(q_2, b) = (q_1, b, L)$
$\delta(q_2, \text{ƀ}) = (q_1, \text{ƀ}, L)$

- Si se estudia el cómputo que realiza esta MT:
  $q_1abw \vdash aq_2bw \vdash q_1abw \vdash aq_2bw \vdash ...$
- Esta MT nunca para: podría decirse que entra en un "*bucle infinito*"
- Esta situación se representa como:
  - $(q, w_1\underline{a}w_2) \vdash^* \infty$
  - $w_1qaw_2 \vdash^* \infty$

Diferencias entre las máquinas de Turing (MT) y los autómatas finitos (AF)

- Una MT puede leer o escribir información de la cinta, mientras que un AF sólo puede leer su "entrada"

Diferencias entre las máquinas de Turing (MT) y los autómatas finitos (AF)

- Una MT puede leer o escribir información de la cinta, mientras que un AF sólo puede leer su "entrada"
- La cabeza de lectura/escritura de una MT puede moverse a izquierda o derecha, mientras que un AF lee símbolos en una sola "dirección" (hacia la derecha)

Diferencias entre las máquinas de Turing (MT) y los autómatas finitos (AF)

- Una MT puede leer o escribir información de la cinta, mientras que un AF sólo puede leer su "entrada"
- La cabeza de lectura/escritura de una MT puede moverse a izquierda o derecha, mientras que un AF lee símbolos en una sola "dirección" (hacia la derecha)
- La cinta de una MT es infinita, la entrada en un AF es finita

Diferencias entre las máquinas de Turing (MT) y los autómatas finitos (AF)

- Una MT puede leer o escribir información de la cinta, mientras que un AF sólo puede leer su "entrada"
- La cabeza de lectura/escritura de una MT puede moverse a izquierda o derecha, mientras que un AF lee símbolos en una sola "dirección" (hacia la derecha)
- La cinta de una MT es infinita, la entrada en un AF es finita
- Los estados en los que una MT acepta o rechaza toman efecto inmediatamente, mientras que un AF debe consumir toda su entrada para terminar su cómputo

Ejemplo de máquina de Turing

Diseñar una MT M tal que $L(M) = L[a*]$ sobre $\{a, b\}$

$Q = \{q_1, q_2\}$
$\Sigma = \{a, b\}$
$\Gamma = \{a, b, \text{ƀ}\}$
$F = \{q_2\}$
$q_0 = q_1$

$\delta(q_1, a) = (q_1, a, R)$
$\delta(q_1, \text{ƀ}) = (q_2, \text{ƀ}, R)$

Ejemplo de máquina de Turing

Diseñar una MT M tal que $L(M) = L[a*]$ sobre $\{a, b\}$

$Q = \{q_1, q_2\}$
$\Sigma = \{a, b\}$
$\Gamma = \{a, b, \text{ƀ}\}$
$F = \{q_2\}$
$q_0 = q_1$

$\delta(q_1, a) = (q_1, a, R)$
$\delta(q_1, \text{ƀ}) = (q_2, \text{ƀ}, R)$

Hay dos posibilidades para rechazar una cadena $w \in \Sigma^*$:

1. M para en un estado que no es de aceptación
2. M no para nunca (entrando en un "bucle infinito")

Ejemplo de máquina de Turing

Diseñar otra MT M tal que $L(M) = L[a*]$ sobre $\{a, b\}$

$Q = \{q_1, q_2, q_3\}$
$\Sigma = \{a, b\}$
$\Gamma = \{a, b, \text{ƀ}\}$
$F = \{q_3\}$
$q_0 = q_1$

$\delta(q_1, a) = (q_1, a, R)$
$\delta(q_1, b) = (q_2, b, R)$
$\delta(q_1, \text{ƀ}) = (q_3, \text{ƀ}, R)$
$\delta(q_2, a) = (q_2, a, R)$
$\delta(q_2, b) = (q_2, b, R)$
$\delta(q_2, \text{ƀ}) = (q_2, \text{ƀ}, R)$

Ejemplo de máquina de Turing

Diseñar otra MT M tal que $L(M) = L[a*]$ sobre $\{a, b\}$

$Q = \{q_1, q_2, q_3\}$
$\Sigma = \{a, b\}$
$\Gamma = \{a, b, \text{b}\}$
$F = \{q_3\}$
$q_0 = q_1$

$\delta(q_1, a) = (q_1, a, R)$
$\delta(q_1, b) = (q_2, b, R)$
$\delta(q_1, \text{b}) = (q_3, \text{b}, R)$
$\delta(q_2, a) = (q_2, a, R)$
$\delta(q_2, b) = (q_2, b, R)$
$\delta(q_2, \text{b}) = (q_2, \text{b}, R)$

Si esta MT lee un símbolo b, pasa al estado q_2 donde se moverá indefinidamente hacia la derecha

Máquina de Turing que reconoce un CFL

Diseñar una MT M tal que $L(M) = \{a^n b^n \mid n \geq 1\}$

Máquina de Turing que reconoce un CFL

Diseñar una MT M tal que $L(M) = \{a^n b^n \mid n \geq 1\}$

- M cambia la primera a por c: $\delta(q_1, a) = (q_2, c, R)$

Máquina de Turing que reconoce un CFL

Diseñar una MT M tal que $L(M) = \{a^n b^n \mid n \geq 1\}$

- M cambia la primera a por c: $\delta(q_1, a) = (q_2, c, R)$
- Luego se desplaza hacia la derecha hasta hallar la primera b y cambiarla por d:
  $\delta(q_2, a) = (q_2, a, R)$
  $\delta(q_2, d) = (q_2, d, R)$
  $\delta(q_2, b) = (q_3, d, L)$

Máquina de Turing que reconoce un CFL

Diseñar una MT M tal que $L(M) = \{a^n b^n \mid n \geq 1\}$

- M cambia la primera a por c: $\delta(q_1, a) = (q_2, c, R)$
- Luego se desplaza hacia la derecha hasta hallar la primera b y cambiarla por d:
  $\delta(q_2, a) = (q_2, a, R)$
  $\delta(q_2, d) = (q_2, d, R)$
  $\delta(q_2, b) = (q_3, d, L)$
- A continuación se mueve a la izquierda hasta posicionarse en la c de más a la derecha:
  $\delta(q_3, d) = (q_3, d, L)$
  $\delta(q_3, a) = (q_3, a, L)$
  $\delta(q_3, c) = (q_1, c, R)$

Máquina de Turing que reconoce un CFL

Diseñar una MT M tal que $L(M) = \{a^n b^n \mid n \geq 1\}$

- Si se acaban los símbolos a, M estará situada sobre un símbolo d en el estado q_1

Máquina de Turing que reconoce un CFL

Diseñar una MT M tal que $L(M) = \{a^n b^n \mid n \geq 1\}$

- Si se acaban los símbolos a, M estará situada sobre un símbolo d en el estado q_1
- Bastaría con comprobar que todos los símbolos b han sido sustituidos por d:
  $\delta(q_1, d) = (q_4, d, R)$
  $\delta(q_4, d) = (q_4, d, R)$
  $\delta(q_4, \text{ƀ}) = (q_5, \text{ƀ}, L)$

Máquina de Turing que reconoce un CFL

Diseñar una MT M tal que $L(M) = \{a^n b^n \mid n \geq 1\}$

- Si se acaban los símbolos a, M estará situada sobre un símbolo d en el estado q_1
- Bastaría con comprobar que todos los símbolos b han sido sustituidos por d:
  $\delta(q_1, d) = (q_4, d, R)$
  $\delta(q_4, d) = (q_4, d, R)$
  $\delta(q_4, \text{ƀ}) = (q_5, \text{ƀ}, L)$
- El resto de componentes de M se definen como:
  $Q = \{q_1, q_2, q_3, q_4, q_5\}$
  $\Sigma = \{a, b\}$
  $\Gamma = \{a, b, c, d, \text{ƀ}\}$
  $F = \{q5\}$
  $q_0 = q_1$

Máquina de Turing que reconoce un CFL

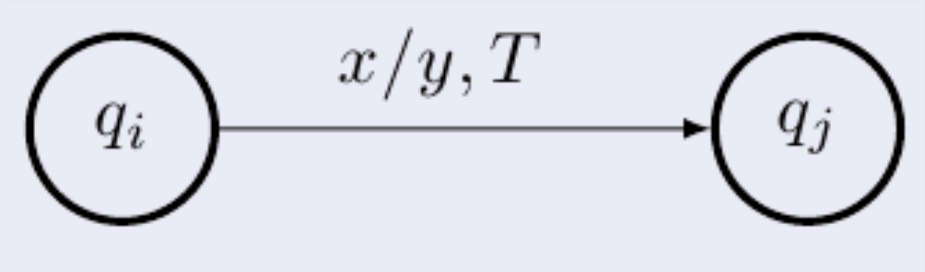
MT que reconoce $L = \{a^n b^n \mid n \geq 1\}$

turingmachine.io
Fichero de entrada

Diagrama de transiciones

Una MT puede ser representada gráficamente mediante un diagrama de transiciones

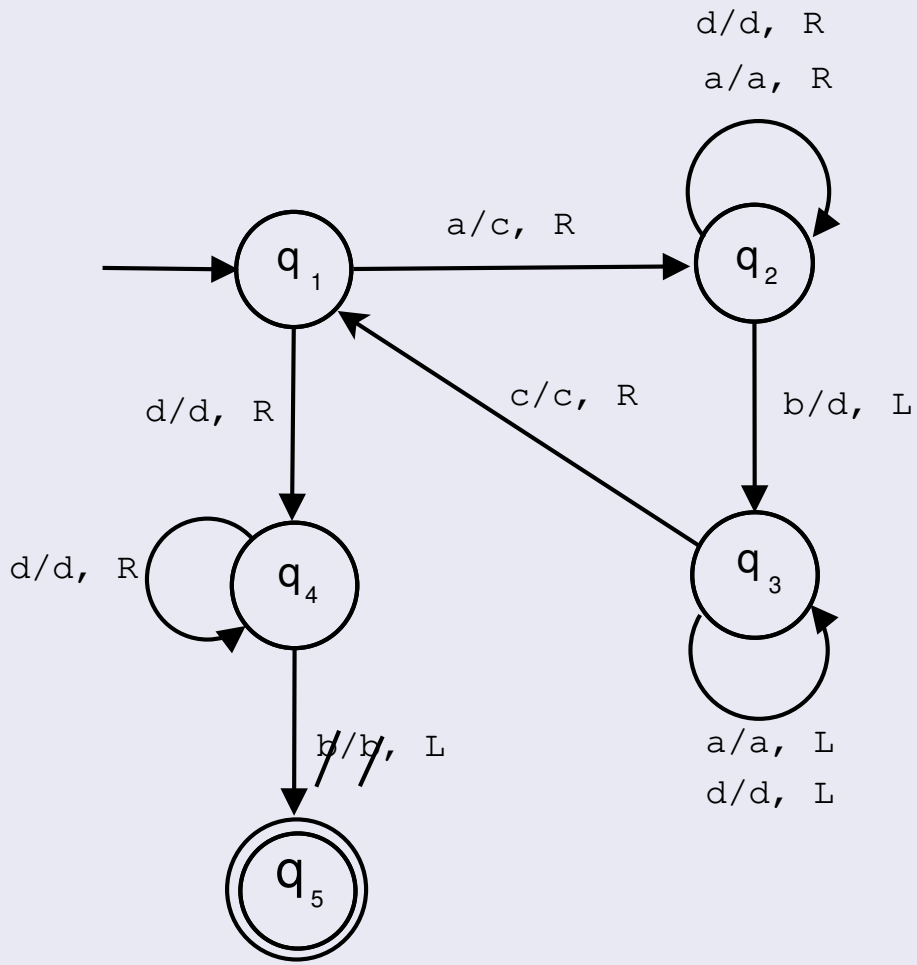
Sea $M \equiv (Q, \Sigma, \Gamma, q_0, \text{ƀ}, F, \delta)$
Una transición $\delta(q_i, x) = (q_j, y, T)$ con $T \in \{L, R\}$ se representa mediante el diagrama:



- El estado de arranque de M se indica tal como se hace con los diagramas de transiciones de los AFs
- Los estados de aceptación se representan usando doble trazo

Diagrama de transiciones

Ejemplo: MT que reconoce $L = \{a^n b^n \mid n \geq 1\}$



Transformación de cadenas

Una MT tiene la capacidad de transformar una cadena de entrada (secuencia de símbolos sobre su cinta) en una cadena de salida (secuencia resultante al finalizar el cómputo):

- Modelo abstracto de computador

Ejemplo: MT que complementa cadenas de $\{a, b\}$

$M \equiv (Q, \Sigma, \Gamma, q_0, \text{ƀ}, F, \delta)$ se define como:

$Q = \{q_1, q_2, q_3\}$ $\quad F = \{q_3\}$

$\Sigma = \{a, b\}$ $\quad q_0 = q_1$

$\Gamma = \{a, b, \text{ƀ}\}$

δ	a	b	ƀ
q_1	q_1, b, R	q_1, a, R	$q_2, \text{ƀ}, L$
q_2	q_2, a, L	q_2, b, L	$q_3, \text{ƀ}, R$

Transformación de cadenas

Ejemplo: MT que complementa cadenas de $\{a, b\}$

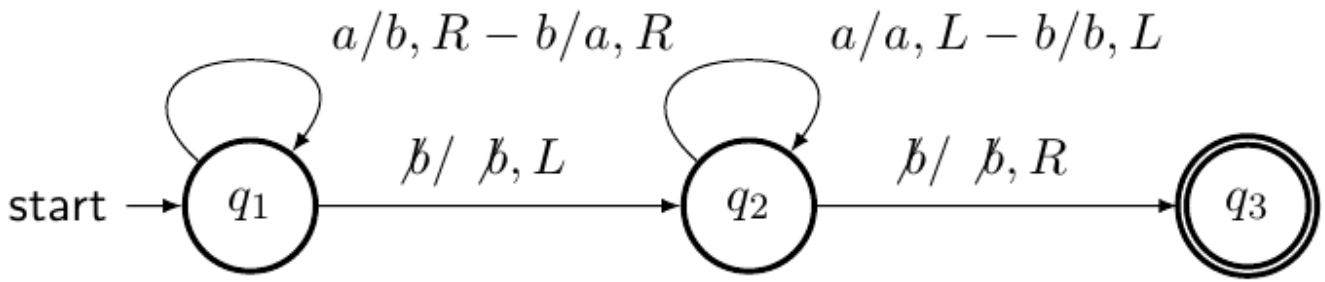
$M \equiv (Q, \Sigma, \Gamma, q_0, \text{ƀ}, F, \delta)$ se define como:

$Q = \{q_1, q_2, q_3\}$ $\quad F = \{q_3\}$

$\Sigma = \{a, b\}$ $\quad q_0 = q_1$

$\Gamma = \{a, b, \text{ƀ}\}$

δ	a	b	ƀ
q_1	q_1, b, R	q_1, a, R	$q_2, \text{ƀ}, L$
q_2	q_2, a, L	q_2, b, L	$q_3, \text{ƀ}, R$



Función Turing computable

Definición

Una **función de cadena** es una función $f\colon \Sigma^* \to \Sigma^*$

Función Turing computable

Definición

Una **función de cadena** es una función $f\colon \Sigma^* \to \Sigma^*$

Definición

Se dice que una función de cadena es Turing computable (o computable) si existe una MT $M \equiv (Q, \Sigma, \Gamma, q_0, \text{ƀ}, F, \delta)$ para la cual:

$$q_0 w \ \vdash^* \ q_f u$$

para algún $q_f \in F$ cuando $f(w) = u$

Función Turing computable

Definición

Una **función de cadena** es una función $f\colon \Sigma^* \to \Sigma^*$

Definición

Se dice que una función de cadena es Turing computable (o computable) si existe una MT $M \equiv (Q, \Sigma, \Gamma, q_0, \text{b}, F, \delta)$ para la cual:

$$q_0 w \ \vdash^* \ q_f u$$

para algún $q_f \in F$ cuando $f(w) = u$

Aunque la computabilidad de Turing se ha definido sólo para funciones de cadena, la definición se puede extender a las funciones integrables

Función Turing computable

Ejemplo

Supongamos $\Sigma = \{a, b\}$ y representemos los enteros positivos mediante cadenas de "aes":

- El entero positivo n estaría representado por a^n
- La función suma $f(n, m) = n + m$ se puede implementar mediante la transformación de $a^n b a^m$ en $a^{n+m} b$

δ	a	b	ƀ
q_1	q_1, a, R	q_2, a, R	
q_2	q_2, a, R		q_3, ƀ, L
q_3	q_4, b, L		
q_4	q_4, a, L		q_5, ƀ, R

- Esta MT desplaza el símbolo b hacia la derecha, después de a^{n+m}
- Cuando termina ($q_5 \in F$), la MT sitúa su cabeza de L/E sobre el símbolo a situado más a la izquierda

Combinación de MT

Definición

Sean $M_1 \equiv (Q_1, \Sigma, \Gamma, q_{01}, \text{ƀ}, F_1, \delta_1)$ y $M_2 \equiv (Q_2, \Sigma, \Gamma, q_{02}, \text{ƀ}, F_2, \delta_2)$ dos MT sobre el mismo Σ, con el mismo Γ y con $Q_1 \cap Q_2 = \emptyset$, definimos la **combinación de M_1 y M_2** como $M_1M_2 \equiv (Q, \Sigma, \Gamma, q_0, \text{ƀ}, F, \delta)$ con:

$$Q_1 \cup Q_2 \qquad q_0 = q_{01} \qquad F = F_2$$

$$\delta(q,a) = \begin{cases} \delta_1(q,a) & \text{si } q \in Q_1 \text{ y } \delta_1(q,a) \neq (p,t,X)\ \forall p \in F_1 \\ (q_{02},t,X) & \text{si } q \in Q_1 \text{ y } \delta_1(q,a) = (p,t,X) \text{ con } p \in F_1 \\ \delta_2(q,a) & \text{si } q \in Q_2 \end{cases}$$

- M_1M_2 se comporta como M_1 hasta que alcanza un estado de F_1
- En ese momento, cambia al estado q_{02} y se comporta como M_2 hasta el final del cómputo

Combinación de MT

Ejemplo

Consideremos una MT M_1:

δ	a	ƀ
q_1	q_2, a, R	q_2, ƀ, R
q_2	q_2, a, R	q_3, ƀ, L
q_3	q_4, a, R	q_4, ƀ, R

Consideremos una MT M_2:

δ	a	ƀ
q_1	q_2, a, R	q_2, a, R

Combinación de MT

Ejemplo

Consideremos una MT M_1:

δ	a	ƀ
q_1	q_2, a, R	q_2, ƀ, R
q_2	q_2, a, R	q_3, ƀ, L
q_3	q_4, a, R	q_4, ƀ, R

Consideremos una MT M_2:

δ	a	ƀ
q_1	q_2, a, R	q_2, a, R

- M_1 busca el primer ƀ a la derecha de la posición inicial de su cabeza L/E
- M_2 escribe un símbolo a y para
- Al combinar M_1 y M_2 de modo que una computación de M_1 vaya seguida de una de M_2 obtendremos una MT M_1M_2 que busca el primer símbolo ƀ a la derecha de la posición inicial de su cabeza L/E y lo sustituye por una a

MT con posibilidad de no moverse

- Originalmente definimos:
  $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R\}$
- De forma alternativa podríamos definir:
  $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, S\}$
- En este caso, la MT tendría la posibilidad de no mover su cabeza de L/E en cada paso de cómputo

MT con posibilidad de no moverse

- Originalmente definimos:
  $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$
- De forma alternativa podríamos definir:
  $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R, S\}$
- En este caso, la MT tendría la posibilidad de no mover su cabeza de L/E en cada paso de cómputo

Este tipo de MT se puede simular por una MT de las definidas originalmente

MT con múltiples pistas

- En una MT con múltiples pistas, cada celda no contiene un único símbolo, sino un vector de símbolos:

$$\delta : Q \times \Gamma^k \rightarrow Q \times \Gamma^k \times \{L, R\}$$

- En cada movimiento:
  - La MT lee un vector de símbolos
  - La MT cambia de estado
  - La MT escribe un nuevo vector
  - La MT mueve su cabeza de L/E

MT con múltiples pistas

Ejemplo: MT multi-pista para sumar números binarios

$$\delta(q_1, a) = \begin{cases} (q_1, a, R) & \text{si } a \neq (\not b, \not b, \not b) \\ (q_2, a, L) & \text{si } a = (\not b, \not b, \not b) \end{cases}$$

$$\begin{array}{ll}
\delta(q_2, (0,0,\not b)) = (q_2, (0,0,0), L) & \delta(q_3, (0,0,\not b)) = (q_2, (0,0,1), L) \\
\delta(q_2, (0,1,\not b)) = (q_2, (0,1,1), L) & \delta(q_3, (0,1,\not b)) = (q_3, (0,1,0), L) \\
\delta(q_2, (1,0,\not b)) = (q_2, (1,0,1), L) & \delta(q_3, (1,0,\not b)) = (q_3, (1,0,0), L) \\
\delta(q_2, (1,1,\not b)) = (q_3, (1,1,0), L) & \delta(q_3, (1,1,\not b)) = (q_3, (1,1,1), L) \\
\delta(q_2, (\not b,\not b,\not b)) = (q_4, (\not b,\not b,0), S) & \delta(q_3, (\not b,\not b,\not b)) = (q_4, (\not b,\not b,1), S)
\end{array}$$

$\not b$	1	0	1	$\not b$
$\not b$	0	1	0	$\not b$
$\not b$	$\not b$	$\not b$	$\not b$	$\not b$

MT multi-cinta

- La MT tiene varias cintas, cada una con su cabeza de L/E
- En un sólo movimiento, la MT:
  - Cambia de estado, dependiendo del estado actual y del contenido de todas las cintas
  - Escribe un nuevo símbolo en cada una de sus cintas
  - Mueve, de forma independiente, cada cabeza de L/E a izquierda o derecha ($\{L, R\}$)
- Para una MT con n cintas:
  $\delta : Q \times \Gamma^n \rightarrow Q \times \Gamma^n \times \{L, R, S\}^n$
  Es decir, $\delta(q, (a_1, a_2, ..., a_n)) = (p, (b_1, b_2, ..., b_n), (X_1, X_2, ..., X_n))$

Este tipo de MT se puede simular por una MT tal como se definió originalmente

MT multi-cinta

Ejemplo: MT con dos cintas C_1 y C_2 que reconoce $L = \{a^n b^n \mid n \geq 1\}$

- La cadena a analizar la colocamos en la cinta C_1
- Suponemos que el estado inicial es q_1 y que inicialmente la cabeza de L/E de C_1 está situada sobre el símbolo del extremo izquierdo de la cadena
- Mientras encuentra símbolos a los deja en C_1 y los copia en C_2: $\delta(q_1, (a,\text{ƀ})) = (q_1, (a, a), (R, R))$
- Cuando encuentra la primera b la deja en C_1 y se mueve a la izquierda (L) en C_2: $\delta(q_1, (b,\text{ƀ})) = (q_2, (b,\text{ƀ}), (S, L))$
- A continuación, se van emparejando las b de C_1 con las a copiadas en C_2: $\delta(q_2, (b, a)) = (q_2, (b, a), (R, L))$
- Cuando encuentra (ƀ,ƀ) pasa a q_3 y acaba: $\delta(q_2, (\text{ƀ},\text{ƀ})) = (q_3, (\text{ƀ}, \text{ƀ}), (R, L))$

MT con cinta multidimensional

- En este caso la cinta de la MT es N-dimensional
- Una posibilidad será una cinta bidimensional, que se extiende infinitamente hacia la izquierda, derecha, arriba y abajo, de modo que la función de transición estará definida por:
  $\delta : Q \times \Gamma \rightarrow Q \times \Gamma \times \{L, R, U, D\}$

Este tipo de MT se puede simular por una MT tal como se definió originalmente

MT no determinista

- Se elimina el requerimiento de que la regla de transición sea una función
- Para un estado y símbolo actual, puede existir un número finito de movimientos a elegir

Ejemplo

Si la MT tuviera la transición: $\delta(q_1, a) = \{(q_1, b, R), (q_2, a, L)\}$
Podríamos tener los movimientos siguientes:

- $(q_1, abb\underline{a}b) \vdash (q_1, abbb\underline{b})$
- $(q_1, abb\underline{a}b) \vdash (q_2, ab\underline{b}ab)$

Este tipo de MT se puede simular por una MT tal como se definió originalmente

Modificaciones de la MT

- Se puede demostrar que todas las variantes que hemos introducido con respecto a la MT tal como se definió originalmente, tienen la misma capacidad de cómputo que la MT original
- Todo cómputo que pueda realizarse con una MT "de la forma original" puede ser también realizado por cualquiera de las variantes
- Alternativamente, si un cómputo puede realizarse con cualquiera de las variantes, también puede ser llevado a cabo por una MT definida "de la forma original"
- La finalidad es que, si una de las variantes resulta más adecuada para resolver un determinado problema, se pueda utilizar para simplificar

Máquina de Turing universal

Definición

Una MT universal (M_U) es una MT que, a partir de una descripción adecuada de una MT (M) y una cadena de entrada ($w \in \Sigma^*$), simula el comportamiento de M ante la cadena w

Codificación de una MT arbitraria $M \equiv (Q, \Sigma, \Gamma, q_0, \text{b}, F, \delta)\}$

- Se ha de hallar una manera de describir M, puesto que esa descripción se dará a M_U como entrada
- Las entradas para M_U serán (M, w)
- Se codificará M utilizando el alfabeto $\Sigma = \{0, 1\}$ y estableciendo algunos convenios

Máquina de Turing universal

Codificación de una MT arbitraria $M \equiv (Q, \Sigma, \Gamma, q_0, \text{ƀ}, F, \delta)\}$

1. Se hará que $|F| = 1$ (siempre se puede conseguir) y supondremos que $Q = \{q_1, q_2, ..., q_n\}$ siendo $q_0 = q_1$ y $F = \{q_n\}$
2. Asumiremos $\Gamma = \{a_1, a_2, ..., a_m\}$ siendo $a_1 = \text{ƀ}$
3. Se representará q_i mediante 1^i (p. ej. $q_5 = 11111$)
4. Se representará a_i mediante 1^i (p. ej. $a_3 = 111$)
5. Se representará $L = 1$ y $R = 11$
6. Utilizando el símbolo 0 como separador se representará:
   $\delta(q_2, a_3) = (q_3, a_2, R)$ como 01101110111011011
   $\delta(q_5, a_1) = (q_2, a_3, L)$ como 01111101011011101
   De este modo se puede especificar completamente la función δ de M
7. Una MT arbitraria M tiene una codificación representada por una cadena de ceros y unos de longitud finita
8. Dada una codificación, ésta se puede decodificar unívocamente

Máquina de Turing universal

Implementación

M_U se puede implementar como una MT con tres cintas (C_1, C_2, C_3) cuyo alfabeto Σ contenga $\{0, 1\}$:

1. C_1 contiene la codificación de M y tiene su cabeza de L/E situada sobre el primer 0 de la codificación
2. C_2 contiene la codificación (con $\{0, 1\}$) del contenido de la cinta de M, con la cabeza de L/E situada sobre el primer 1 del símbolo actual
3. C_3 almacena el estado actual de M e inicialmente contiene:
   ...ƀ$\underline{1}$ ƀ...
   Es decir, la máquina está en su estado de arranque, q_0

Máquina de Turing universal

Implementación

El modo de funcionamiento de M_U es el siguiente:

1. M_U analiza y compara el contenido de las cintas C_2 y C_3 (símbolo y estado)
2. Busca en $C1$ una transición para ese par (símbolo-estado) hasta encontrarla o finalizar la búsqueda
3. Si no encuentra una transición, M_U para (como lo haría M)
4. En caso contrario, M_U simula M modificando convenientemente el contenido de las tres cintas
5. Si M para ante w, M_U parará cuando tenga como entrada (M, w)

Indice

Lenguajes aceptados por máquinas de Turing

Definición

Un lenguaje L se dice que es **recursivamente enumerable (LRE)** si existe una MT $M \equiv (Q, \Sigma, \Gamma, q_0, \text{ƀ}, F, \delta)$ tal que $L = L(M)$

Lenguajes aceptados por máquinas de Turing

Definición

Un lenguaje L se dice que es **recursivamente enumerable (LRE)** si existe una MT $M \equiv (Q, \Sigma, \Gamma, q_0, \text{ƀ}, F, \delta)$ tal que $L = L(M)$

Definición

Un lenguaje L es **recursivo (LREC)** si existe al menos una MT que reconoce las cadenas de L y para con toda cadena de entrada.
Los lenguajes recursivos son un subconjunto de los lenguajes recursivamente enumerables

Lenguajes aceptados por máquinas de Turing

Definición

Un lenguaje L se dice que es **recursivamente enumerable (LRE)** si existe una MT $M \equiv (Q, \Sigma, \Gamma, q_0, \text{ƀ}, F, \delta)$ tal que $L = L(M)$

Definición

Un lenguaje L es **recursivo (LREC)** si existe al menos una MT que reconoce las cadenas de L y para con toda cadena de entrada.
Los lenguajes recursivos son un subconjunto de los lenguajes recursivamente enumerables

Para que una MT reconozca un lenguaje no necesita parar sobre cualquier cadena de entrada: basta que pare en un estado $q \in F$ para aquellas cadenas $w \in L$.

Si un L es un LREC, existe al menos una MT M tal que, si se le pasa a M una $w \in L$, entonces M para y acepta w. Si se le pasa una $w \notin L$, entonces M para y rechaza w

Máquina de Turing a partir de un DFA

Sea $M \equiv (Q, \Sigma, q_0, F, \delta)$ un DFA

Se puede construir una MT $M' \equiv (Q', \Sigma', \Gamma, \text{ƀ}, q_0', F', \delta')$ tal que $L(M') = L(M)$:

- $Q' = Q \cup \{q'\}$ $(q' \notin Q)$
- $\Sigma' = \Sigma$
- $\Gamma = \Sigma \cup \{\text{ƀ}\}$
- $F' = \{q'\}$
- $\delta'(q, a) = (\delta(q, a), a, R) \quad \forall q \in Q, \ \forall a \in \Sigma$
- $\delta'(q, \text{ƀ}) = (q', \text{ƀ}, S) \quad \forall q \in F$

Máquina de Turing a partir de un DFA

Sea $M \equiv (Q, \Sigma, q_0, F, \delta)$ un DFA

Se puede construir una MT $M' \equiv (Q', \Sigma', \Gamma, \text{ƀ}, q_0', F', \delta')$ tal que $L(M') = L(M)$:

- $Q' = Q \cup \{q'\}$ $(q' \notin Q)$
- $\Sigma' = \Sigma$
- $\Gamma = \Sigma \cup \{\text{ƀ}\}$
- $F' = \{q'\}$
- $\delta'(q, a) = (\delta(q, a), a, R) \quad \forall q \in Q, \ \forall a \in \Sigma$
- $\delta'(q, \text{ƀ}) = (q', \text{ƀ}, S) \quad \forall q \in F$

La MT analiza su entrada de izquierda a derecha y con cada símbolo cambia de estado del mismo modo que lo hace el DFA.

Aceptará la cadena si al llegar al final de la misma encuentra un ƀ y se encuentra en un estado de aceptación

Lenguajes regulares y lenguajes independientes del contexto

Teorema

Si $L \subseteq \Sigma^*$ es un lenguaje regular, entonces L es un lenguaje recursivo

Lenguajes regulares y lenguajes independientes del contexto

Teorema

Si $L \subseteq \Sigma^*$ es un lenguaje regular, entonces L es un lenguaje recursivo

$L = \{a^n b^n \mid n \geq 1\}$ es un lenguaje recursivo (se han estudiado varias MT que lo reconocen) pero hemos estudiado asimismo que L no es regular

Así pues, hay lenguajes recursivos que no son regulares: $L_{REG} \subset L_{REC}$

Lenguajes regulares y lenguajes independientes del contexto

Teorema

Si $L \subseteq \Sigma^*$ es un lenguaje regular, entonces L es un lenguaje recursivo

$L = \{a^n b^n \mid n \geq 1\}$ es un lenguaje recursivo (se han estudiado varias MT que lo reconocen) pero hemos estudiado asimismo que L no es regular

Así pues, hay lenguajes recursivos que no son regulares: $L_{REG} \subset L_{REC}$

Teorema

Si $L \subseteq \Sigma^*$ es un lenguaje independiente del contexto, entonces L es un lenguaje recursivo

Lenguajes recursivos

Teorema

Si L_1 y L_2 son lenguajes recursivos, entonces $L_1 \cap L_2$ también lo es

Lenguajes recursivos

Teorema

Si L_1 y L_2 son lenguajes recursivos, entonces $L_1 \cap L_2$ también lo es

Demostración

- Sean M_1 y M_2 las MT que reconocen L_1 y L_2 respectivamente, y que paran ante cualquier cadena:
  - Si $w \in L(M_i)$, M_i parará en un estado de aceptación
  - Si $w \notin L(M_i)$, M_i parará en un estado que no es de aceptación
- Sea M una MT con 2 cintas.
  Se coloca w en la cinta 1 y M hará una copia de ella en la cinta 2

Lenguajes recursivos

Teorema

Si L_1 y L_2 son lenguajes recursivos, entonces $L_1 \cap L_2$ también lo es

Demostración

- M emulará a M_1 usando como entrada el contenido de la cinta 1 como entrada hasta que M_1 pare
  - Si M_1 para en un estado que no es de aceptación, entonces M también para y rechaza w
  - Si M_1 acepta w, entonces M emulará ahora a M_2 usando la copia de la cinta 2
- Si M_2 rechaza w, entonces M rechaza la cadena
- Si M_2 acepta w, entonces M también acepta w

Lenguajes recursivos

Teorema

Si L_1 y L_2 son lenguajes recursivos, entonces $L_1 \cap L_2$ también lo es

Demostración

- Así pues, se tiene que $w \in L(M) \Leftrightarrow w \in L(M_1)$ y $w \in L(M_2)$, y por lo tanto $L(M) = L(M_1) \cap L(M_2)$
- Por otra parte, M para ante cualquier cadena de entrada, de modo que $L(M)$ es recursivo

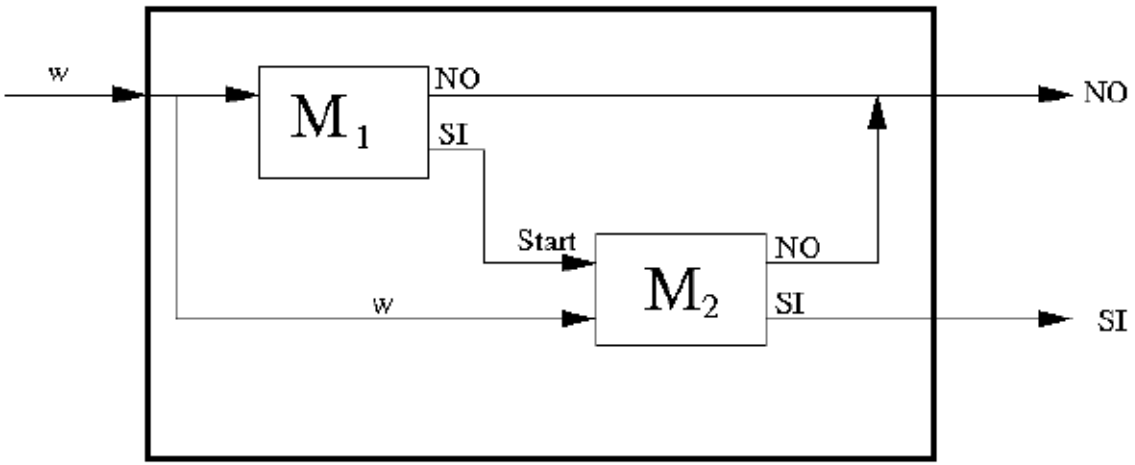
Lenguajes recursivos

Teorema

Si L_1 y L_2 son lenguajes recursivos, entonces $L_1 \cap L_2$ también lo es

Demostración

- Así pues, se tiene que $w \in L(M) \Leftrightarrow w \in L(M_1)$ y $w \in L(M_2)$, y por lo tanto $L(M) = L(M_1) \cap L(M_2)$
- Por otra parte, M para ante cualquier cadena de entrada, de modo que $L(M)$ es recursivo



Lenguajes recursivos

Teorema

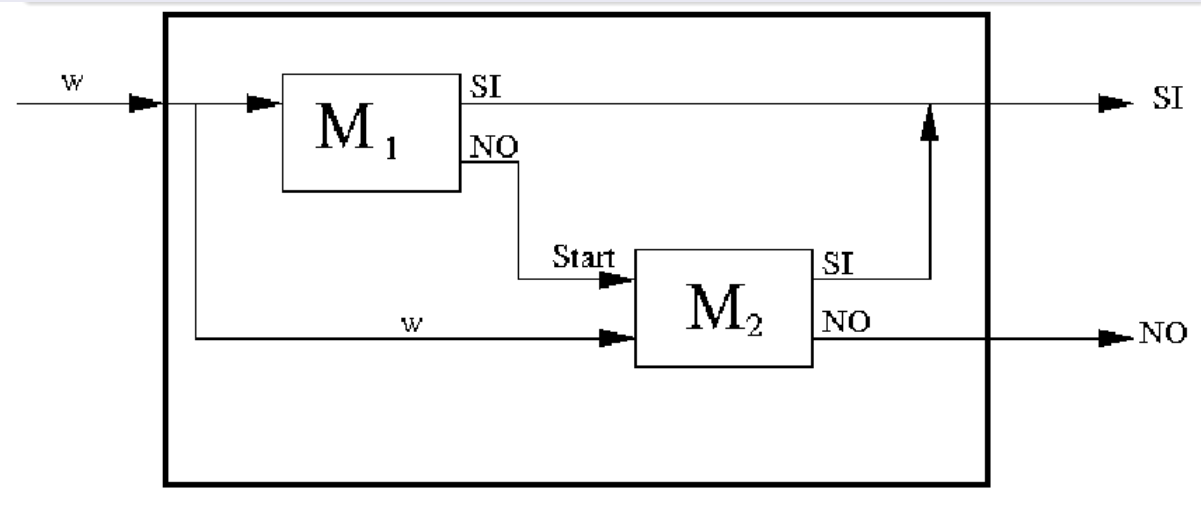
Si L_1 y L_2 son lenguajes recursivos, entonces $L_1 \cup L_2$ también lo es

Lenguajes recursivos

Teorema

Si L_1 y L_2 son lenguajes recursivos, entonces $L_1 \cup L_2$ también lo es

Demostración



Lenguajes recursivos

Teorema

Si L es un lenguaje recursivo, entonces $\overline{L}$ también lo es

Lenguajes recursivos

Teorema

Si L es un lenguaje recursivo, entonces $\overline{L}$ también lo es

Demostración

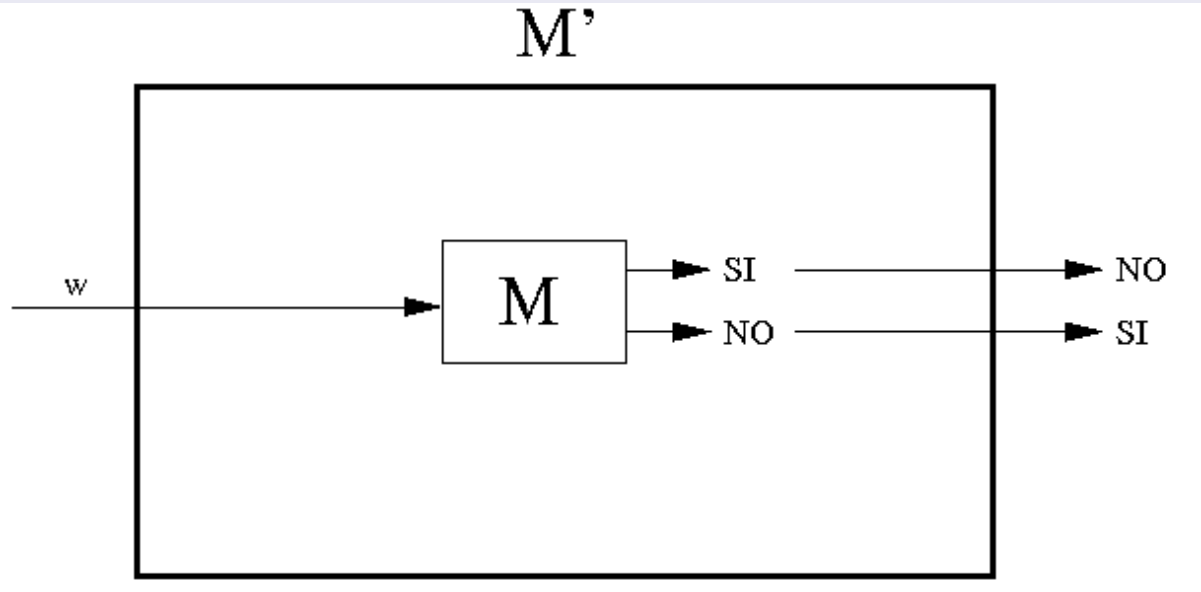
- Si L es recursivo, existe una MT $M \equiv (Q, \Sigma, \Gamma, \delta, F, q_0, \text{ƀ})$ que acepta L y que para ante cualquier entrada
- Sea $M' \equiv (Q, \Sigma, \Gamma, \delta, Q - F, q_0, \text{ƀ})$:
  - Si $w \in L$, M parará en un estado de aceptación y M' parará en un estado que no es de aceptación
  - Si $w \notin L$, M parará en un estado que no es de aceptación y M' parará en un estado de aceptación
  - Así pues, $L(M') = \Sigma^* - L = \overline{L}$
- Puesto que M para ante cualquier cadena de entrada, M' también
- Así pues se concluye que $\overline{L}$ es recursivo

Lenguajes recursivos

Teorema

Si L es un lenguaje recursivo, entonces $\overline{L}$ también lo es

Demostración



Lenguajes recursivamente enumerables

Si L es un lenguaje recursivo, entonces $\overline{L}$ también lo es

En general, esta propiedad no es cierta para lenguajes recursivamente enumerables

Lenguajes recursivamente enumerables

Si L es un lenguaje recursivo, entonces $\overline{L}$ también lo es

En general, esta propiedad no es cierta para lenguajes recursivamente enumerables

Teorema

Existe un lenguaje recursivamente enumerable L para el cual $\overline{L}$ no es recursivamente enumerable

Lenguajes recursivamente enumerables

Cadenas y máquinas de Turing con alfabeto Σ

- Sea Σ un alfabeto
- Σ^* es numerable, de modo que se puede enumerar: $\Sigma^* = \{w_1, w_2, w_3, ...\}$
- Puesto que las MT con alfabeto Σ se pueden codificar como cadenas binarias, y hay un número numerable de cadenas binarias (no todas representan codificaciones válidas de MTs), hay un número numerable de $MT(\Sigma) : MT(\Sigma) = \{M_1, M_2, M_3, M_4, ...\}$

Lenguajes recursivamente enumerables

Cadenas y máquinas de Turing con alfabeto Σ

- Sea Σ un alfabeto
- Σ^* es numerable, de modo que se puede enumerar: $\Sigma^* = \{w_1, w_2, w_3, ...\}$
- Puesto que las MT con alfabeto Σ se pueden codificar como cadenas binarias, y hay un número numerable de cadenas binarias (no todas representan codificaciones válidas de MTs), hay un número numerable de $MT(\Sigma) : MT(\Sigma) = \{M_1, M_2, M_3, M_4, ...\}$

El lenguaje Diagonal

Sea $L = \{w_i \in \Sigma^* \mid w_i \text{ es aceptada por } M_i\}$

Lenguajes recursivamente enumerables

Cadenas y máquinas de Turing con alfabeto Σ

- Sea Σ un alfabeto
- Σ^* es numerable, de modo que se puede enumerar: $\Sigma^* = \{w_1, w_2, w_3, ...\}$
- Puesto que las MT con alfabeto Σ se pueden codificar como cadenas binarias, y hay un número numerable de cadenas binarias (no todas representan codificaciones válidas de MTs), hay un número numerable de $MT(\Sigma) : MT(\Sigma) = \{M_1, M_2, M_3, M_4, ...\}$

El lenguaje Diagonal

Sea $L = \{w_i \in \Sigma^* \mid w_i \text{ es aceptada por } M_i\}$

Teorema

L es recursivamente enumerable. $\overline{L}$ no lo es

Lenguajes recursivamente enumerables

Demostración: estudiemos en primer lugar que L es recursivamente enumerable

Lenguajes recursivamente enumerables

Demostración: estudiemos en primer lugar que L es recursivamente enumerable

- Para demostrarlo, se construirá una MT, M que acepte L
- M será una composición de varias máquinas auxiliares:
  - Sea $w \in \Sigma^*$. M genera las cadenas $w_1, w_2, w_3, ...$ hasta hallar un $i \in \mathbb{N} \mid w = w_i$
  - A continuación, M genera (codificada) la i-ésima $MT(\Sigma)$, M_i
  - M pasa el par (M_i, w_i) a la MT Universal, M_U, que simulará M_i con $w_i = w$ como entrada
  - Si M_i para y acepta w_i entonces M para y acepta w, de modo que M reconoce L
  - Puede ocurrir que M_i pare sin aceptar w_i o que M_i no pare ante w_i
  - En ambos casos, M rechaza la cadena w
  - Por lo tanto, $w \in L(M) \Leftrightarrow w_i \in L(M_i)$
  - Así pues, L es recursivamente enumerable puesto que se ha construido una MT, M que reconoce L

Lenguajes recursivamente enumerables

Demostración: veamos por reducción al absurdo que $\overline{L}$ no es recursivamente enumerable

- Sea $L = \{w_i \in \Sigma^* \mid w_i \text{ es aceptada por } M_i\}$
- Si $\overline{L}$ es recursivamente enumerable, ha de ser aceptado por una $MT(\Sigma)$
- Sea M_j la MT que reconoce $\overline{L}$: $\overline{L} = L(M_j)$
- Considérese la cadena $w_j \in \Sigma^*$:
  - Si $w_j \in L(M_j)$ entonces $w_j \in L \Rightarrow w_j \notin \overline{L} = L(M_j) \Rightarrow w_j \notin L(M_j)$
  - Si $w_j \notin L(M_j)$ entonces $w_j \notin L \Rightarrow w_j \in \overline{L} = L(M_j) \Rightarrow w_j \in L(M_j)$
- En ambos casos se alcanza un absurdo, así que se concluye que $\overline{L}$ no es recursivamente enumerable

Lenguajes recursivamente enumerables

Teorema

Si L_1 y L_2 son lenguajes recursivamente enumerables, entonces $L_1 \cup L_2$ también lo es

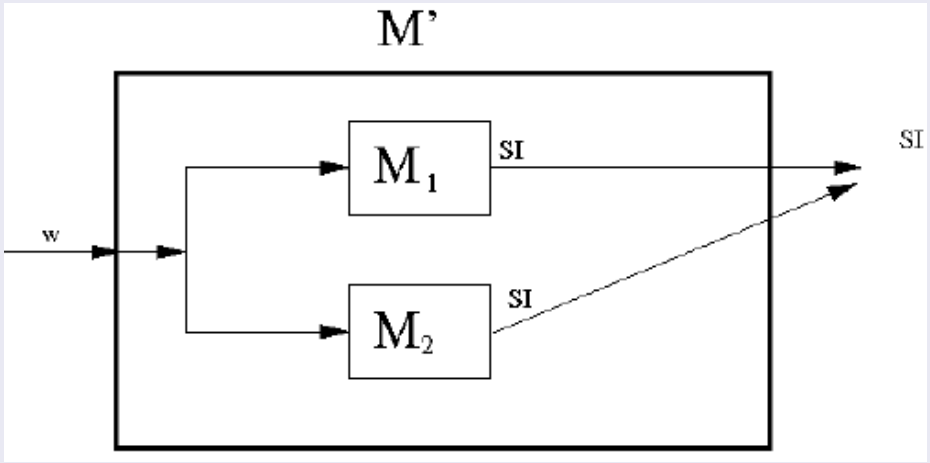
Lenguajes recursivamente enumerables

Teorema

Si L_1 y L_2 son lenguajes recursivamente enumerables, entonces $L_1 \cup L_2$ también lo es

Demostración

La construcción que se hizo anteriormente no funciona en este caso porque las M_i pueden no parar ante la cadena de entrada. Alternativa: M simula en paralelo M_1 y M_2 sobre cintas separadas. Si cualquiera de ellas acepta, M' también



Lenguajes recursivos y recursivamente enumerables

Teorema

Si un lenguaje L y su complementario $\overline{L}$ son ambos recursivamente enumerables, entonces L (y por tanto $\overline{L}$) es recursivo

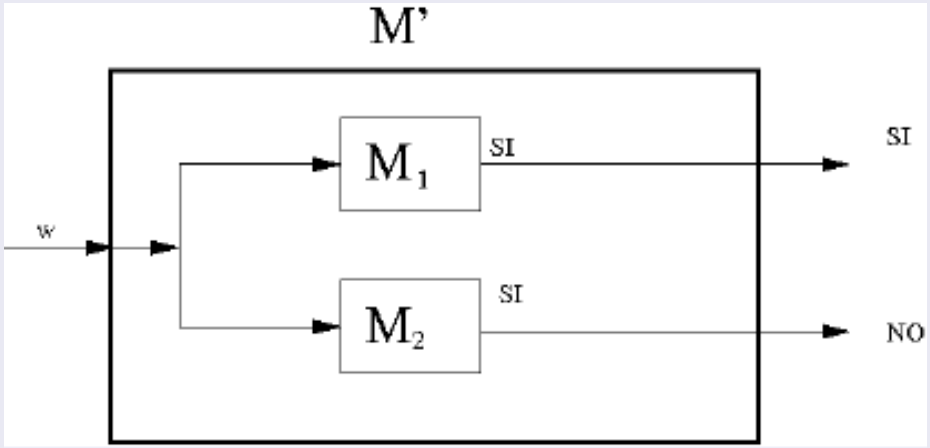
Lenguajes recursivos y recursivamente enumerables

Teorema

Si un lenguaje L y su complementario $\overline{L}$ son ambos recursivamente enumerables, entonces L (y por tanto $\overline{L}$) es recursivo

Demostración

- Sean M_1 y M_2 las MT tales que $L = L(M_1), \overline{L} = L(M_2)$. Se construirá una MT M' que siempre para ante cualquier cadena de entrada y tal que $L(M') = L$:



Lenguajes recursivos y recursivamente enumerables

Dados un par de lenguajes complementarios L y $\overline{L}$, de los teoremas anteriores se deduce que sólo caben las siguientes posibilidades:

Lenguajes recursivos y recursivamente enumerables

Dados un par de lenguajes complementarios L y $\overline{L}$, de los teoremas anteriores se deduce que sólo caben las siguientes posibilidades:

1. Tanto L como $\overline{L}$ son recursivos

Lenguajes recursivos y recursivamente enumerables

Dados un par de lenguajes complementarios L y $\overline{L}$, de los teoremas anteriores se deduce que sólo caben las siguientes posibilidades:

1. Tanto L como $\overline{L}$ son recursivos
2. Ni L ni $\overline{L}$ son recursivamente enumerables

Lenguajes recursivos y recursivamente enumerables

Dados un par de lenguajes complementarios L y $\overline{L}$, de los teoremas anteriores se deduce que sólo caben las siguientes posibilidades:

1. Tanto L como $\overline{L}$ son recursivos
2. Ni L ni $\overline{L}$ son recursivamente enumerables
3. Uno de los dos es recursivamente enumerable pero no recursivo; el otro no es recursivamente enumerable

Gramáticas no restringidas

Gramáticas que se han estudiado:

- Gramáticas regulares:
  - Lado izquierdo de la producción: un único símbolo no terminal
  - Lado derecho de la producción: cadena de terminales seguida por un único símbolo no terminal
- Gramáticas independientes del contexto:
  - Lado izquierdo de la producción: un único símbolo no terminal
  - Lado derecho de la producción: secuencia de terminales y no terminales

Gramáticas no restringidas

Gramáticas que se han estudiado:

- Gramáticas regulares:
  - Lado izquierdo de la producción: un único símbolo no terminal
  - Lado derecho de la producción: cadena de terminales seguida por un único símbolo no terminal
- Gramáticas independientes del contexto:
  - Lado izquierdo de la producción: un único símbolo no terminal
  - Lado derecho de la producción: secuencia de terminales y no terminales

Definición

Una *gramática no restringida* (o gramática estructurada por frases) $G \equiv (N, \Sigma, S, P)$ tiene producciones de la forma $\alpha \rightarrow \beta$, donde $\alpha \in (N \cup \Sigma)^+$ y $\beta \in (N \cup \Sigma)^*$

Gramáticas no restringidas

Obsérvese que cualquier gramática regular o independiente del contexto es además una gramática no restringida

Gramáticas no restringidas

Obsérvese que cualquier gramática regular o independiente del contexto es además una gramática no restringida

Teorema

Si G es una gramática no restringida (de tipo-0), entonces $L(G)$ es un lenguaje recursivamente enumerable

Gramáticas no restringidas

Obsérvese que cualquier gramática regular o independiente del contexto es además una gramática no restringida

Teorema

Si G es una gramática no restringida (de tipo-0), entonces $L(G)$ es un lenguaje recursivamente enumerable

Teorema

Un lenguaje $L \subseteq \Sigma^*$ es recursivamente enumerable $\Leftrightarrow L = L(G)$ para alguna gramática G no restringida (de tipo-0)

Gramáticas sensibles al contexto

Definición

Una *gramática sensible al contexto* $G \equiv (N, \Sigma, S, P)$ tiene producciones de la forma $\alpha \rightarrow \beta$, donde $\alpha, \beta \in (N \cup \Sigma)^+$ y $|\alpha| \leq |\beta|$

Gramáticas sensibles al contexto

Definición

Una *gramática sensible al contexto* $G \equiv (N, \Sigma, S, P)$ tiene producciones de la forma $\alpha \rightarrow \beta$, donde $\alpha, \beta \in (N \cup \Sigma)^+$ y $|\alpha| \leq |\beta|$

Teorema

El conjunto de los lenguajes sensibles al contexto contiene al conjunto de los lenguajes independientes del contexto

Gramáticas sensibles al contexto

Definición

Una *gramática sensible al contexto* $G \equiv (N, \Sigma, S, P)$ tiene producciones de la forma $\alpha \rightarrow \beta$, donde $\alpha, \beta \in (N \cup \Sigma)^+$ y $|\alpha| \leq |\beta|$

Teorema

El conjunto de los lenguajes sensibles al contexto contiene al conjunto de los lenguajes independientes del contexto

Teorema

Si L es un lenguaje sensible al contexto, entonces L es recursivo

Gramáticas sensibles al contexto

Definición

Una *gramática sensible al contexto* $G \equiv (N, \Sigma, S, P)$ tiene producciones de la forma $\alpha \to \beta$, donde $\alpha, \beta \in (N \cup \Sigma)^+$ y $|\alpha| \leq |\beta|$

Teorema

El conjunto de los lenguajes sensibles al contexto contiene al conjunto de los lenguajes independientes del contexto

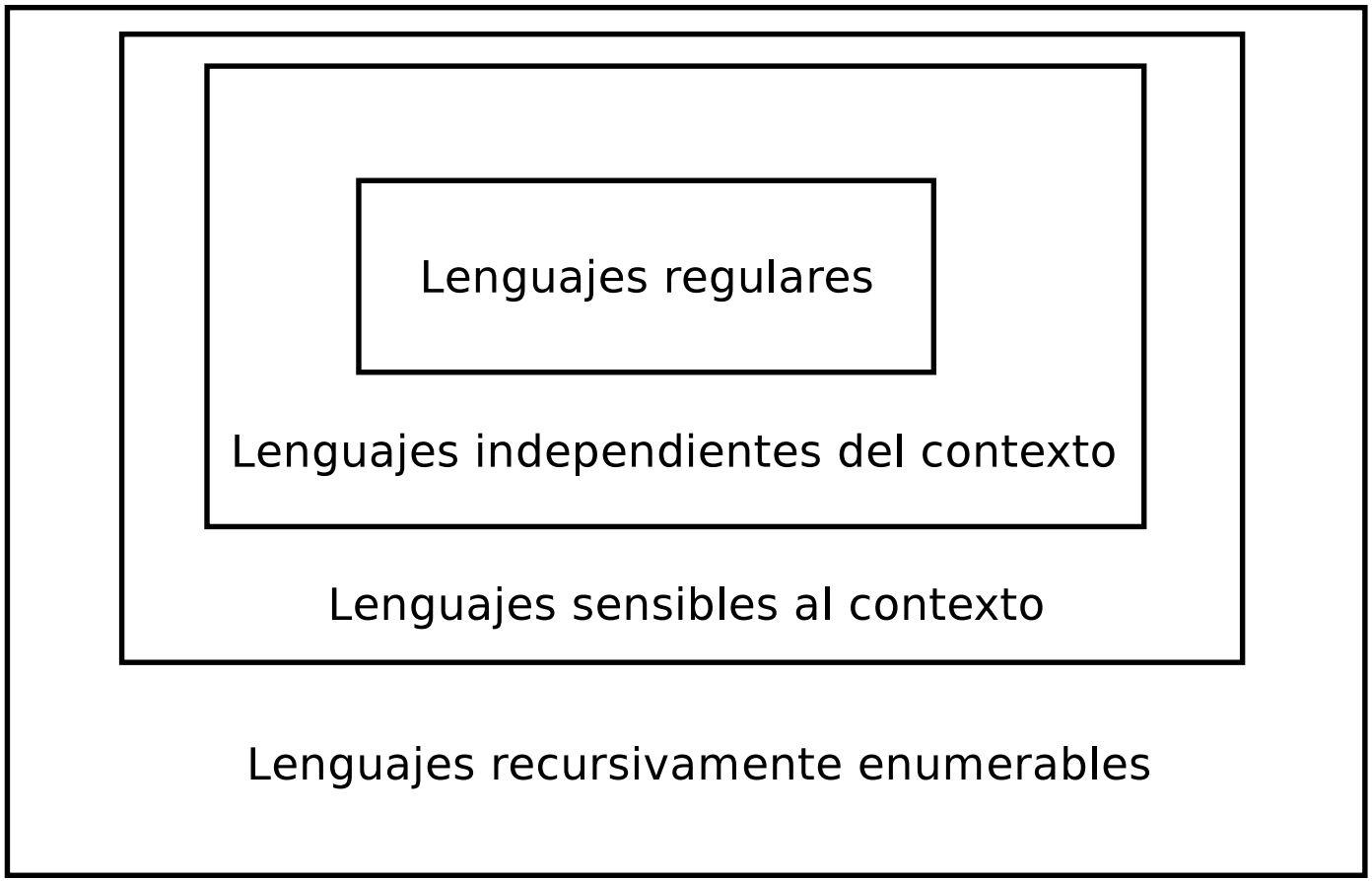
Teorema

Si L es un lenguaje sensible al contexto, entonces L es recursivo

Teorema

Hay lenguajes recursivos que no son sensibles al contexto

Jerarquía de Chomsky



Indice

Simulación de un computador con una MT

Un computador real es una tripleta

$Computador = \{Procesador, Memoria, DispositivosIO\}$

RunProgram::

while *Conectado and Encendido* **do**

    Ejecuta (PC);
    PC := Next(PC);

Cuestiones:

- Procesador = (PC, Programa)
- El Programa se almacena en Memoria (colección de instrucciones)
- PC siempre apunta a la siguiente instrucción a ejecutar

Simulación de un computador con una MT

- Memoria MT: cinta infinita
- E/S MT: cabeza lectora que puede leer/escribir símbolos y moverse por la cinta
- Procesador MT: dispositivo de control que procesa los símbolos referenciados por la cabeza lectora conforme ésta se mueve

Simulación de un computador con una MT

- Memoria MT: cinta infinita
- E/S MT: cabeza lectora que puede leer/escribir símbolos y moverse por la cinta
- Procesador MT: dispositivo de control que procesa los símbolos referenciados por la cabeza lectora conforme ésta se mueve

Cuestiones

- ¿Cuales son las similitudes con un computador real?
- ¿Y las diferencias?

Máquina de Turing

Mecanismo abstracto de cómputo capaz de ejecutar algoritmos paso a paso

- La noción de "paso computacional" está claramente definida. Esto permite precisar sin ambigüedades el concepto de "tiempo de computación" (número de pasos de un cómputo)
- La noción de "espacio de almacenamiento" está claramente presentada, por medio de las celdas individuales en la cinta
- Estos dos recursos, "tiempo" y "espacio", aparecen en la máquina de Turing de una manera muy realista y permiten analizar los efectos de imponer limitaciones sobre ellos, lo cual es muy adecuado en las investigaciones sobre complejidad computacional

Tesis de Church-Turing

Toda función efectivamente computable es Turing-computable

Church, Alonzo (April 1936). *An Unsolvable Problem of Elementary Number Theory*. American Journal of Mathematics. 58 (2): 345–363. doi:10.2307/2371045

Tesis de Church-Turing

Toda función efectivamente computable es Turing-computable

Church, Alonzo (April 1936). *An Unsolvable Problem of Elementary Number Theory*. American Journal of Mathematics. 58 (2): 345–363. doi:10.2307/2371045

- Cualquier procedimiento algorítmico que pueda ser implementado por un humano, un grupo de humanos o un ordenador, puede ser realizado por alguna máquina de Turing
- Se le llama tesis porque no se trata de una sentencia matemáticamente demostrable, debido a la carencia de una definición precisa de procedimiento algorítmico (o algoritmo)

Procedimiento algorítmico o algoritmo

Noción intuitiva

Secuencia de instrucciones simples para llevar a cabo alguna tarea:

- Hallar números primos
- Cálculo del máximo común denominador
- ...

Procedimiento algorítmico o algoritmo

Noción intuitiva

Secuencia de instrucciones simples para llevar a cabo alguna tarea:

- Hallar números primos
- Cálculo del máximo común denominador
- ...

Procedimiento

- Ha de constar de un número finito de instrucciones exactas, estando cada instrucción especificada por un número finito de símbolos
- Debe producir el resultado deseado, si se ejecuta sin error, en un número finito de pasos
- Puede ser realizado por un humano sin asistencia de ninguna máquina, utilizando solo lápiz y papel
- No requiere "trucos" ni ingenio por parte del humano

Problemas resolubles e irresolubles

Algoritmos y máquinas de Turing

- La máquina de Turing modela un proceso, ya sea el reconocimiento de una cadena o la codificación de una secuencia de símbolos en su cinta
- Un algoritmo es un proceso que siempre ha de terminar en un número finito de pasos
- Una máquina de Turing que pare ante cualquier entrada es un modelo abstracto de algoritmo
- Para un lenguaje $L \subseteq \Sigma^*$ que sea recursivo existe un algoritmo M que permite determinar si $w \in L$ para cualquier $w \in \Sigma^*$
- Sin embargo, el problema ¿$w \in L$? en el caso de un lenguaje L recursivamente enumerable pero no recursivo es un ejemplo de problema irresoluble: no existe un algoritmo que lo resuelva

Problemas resolubles e irresolubles

Función características de un lenguaje

Dado un lenguaje $L \subseteq \Sigma^*$, se define su función característica $\chi_L : \Sigma^* \to \{0, 1\}$ como:

$$\chi_L(w) = \begin{cases} 1 & \text{si } w \in L \\ 0 & \text{si } w \notin L \end{cases}$$

- Una función f es *Turing computable* si existe una MT que computa $f(w)\ \forall w$ del dominio de f
- De este modo tenemos que χ_L es Turing computable si L es un lenguaje recursivo
- Por el contrario hemos estudiado que la función χ_L para un lenguaje L que sea recursivamente enumerable pero no recursivo no es computable, puesto que hay cadenas de Σ^* para las que la MT que reconoce L no para

Problemas resolubles e irresolubles

El problema de la parada (*Halting Problem*)

Sea M una máquina de Turing arbitraria, con alfabeto de entrada Σ, y sea $w \in \Sigma^*$, el problema de la parada Π_{HP} se define como:

¿Parará M ante la cadena w?

- Una instancia de Π_{HP} es un par $< M, w >$
- Una solución al problema sería un algoritmo (una MT) que ante cualquier codificación de una instancia del problema respondiera **SI/NO** de forma correcta

Problemas resolubles e irresolubles

- Considérese un alfabeto Σ
- Hemos estudiado que Σ^* es enumerable: $\Sigma^* = \{w_1, w_2, w_3, w_4, ...\}$
- Puesto que las MT con alfabeto Σ se pueden codificar como cadenas binarias, y hay un número numerable de cadenas binarias (no todas representan codificaciones válidas de MTs), hay un número numerable de $MT(\Sigma) : MT(\Sigma) = \{M_1, M_2, M_3, M_4, ...\}$
- Considérese $L = \{w_i \in \Sigma^* \mid w_i$ no es aceptada por $M_i\}$ (el complementario del lenguaje diagonal)
- Demostraremos por reducción al absurdo que L no es recursivamente enumerable

Problemas resolubles e irresolubles

$L = \{w_i \in \Sigma^* \mid w_i$ no aceptada por $M_i\}$ no es recursivamente enumerable

- Sea M_k la MT que reconoce L: $L = L(M_k)$
- Consideremos la cadena $w_k \in \Sigma^*$:
  - Si $w_k \in L$, w_k no es aceptada por M_k, así que $w_k \notin L(M_k) = L$
  - Si $w_k \notin L$, w_k es aceptada por M_k , así que $w_k \in L(M_k) = L$
- Por lo tanto, no existe una MT que reconozca L y, por ello, L no es recursivamente enumerable

Problemas resolubles e irresolubles

Teorema: El problema de la parada es irresoluble

- Por reducción al absurdo: veremos que si Π_{HP} es resoluble, podríamos construir una MT M_L que reconozca el lenguaje anterior: $L = \{w_i \in \Sigma^* \mid w_i \text{ no es aceptada por } M_i\}$
- Supongamos que Π_{HP} es resoluble y que, por tanto, existe un algoritmo que resuelve el problema: una MT M_{HP} que para ante cualquier entrada y que ante descripciones codificadas de una MT M y una cadena de entrada w determina si M para ante w

Problemas resolubles e irresolubles

Teorema: El problema de la parada es irresoluble

- Sea $w \in \Sigma^*$, M_L enumera $w_1, w_2, w_3, w_4, ...$ hasta hallar un $k \in \mathbb{N} \mid w = w_k$
- M_L genera la MT M_k
- M_L pasa $< M_k, w_k >$ a la M_{HP}
- Si M_L obtiene que M_k no para ante w_k, entonces M_L para y acepta $w_k = w$
- Si M_L obtiene que M_k para ante w_k, entonces M_L pasa $< M_k, w_k >$ a M_U

Problemas resolubles e irresolubles

Teorema: El problema de la parada es irresoluble

- Se Sabe que M_U parará (así lo ha indicado M_{HP}) indicando si $w_k \in L(M_k)$
- Si $w_k \in L(M_k)$ entonces M_L para y rechaza $w = w_k$
- Si $w_k \notin L(M_k)$ entonces M_L para y acepta $w = w_k$
- Por lo tanto, $w \in L(M_L) \Leftrightarrow w \in L$ de modo que $L = L(M_L)$ y esto contradice el hecho de que L no es recursivamente enumerable

IMPORTANTE

Estas transparencias se utilizan ÚNICAMENTE como guía para el profesorado durante las clases.

Estas transparencias NO son un material completo y autocontenido para el uso del alumnado.