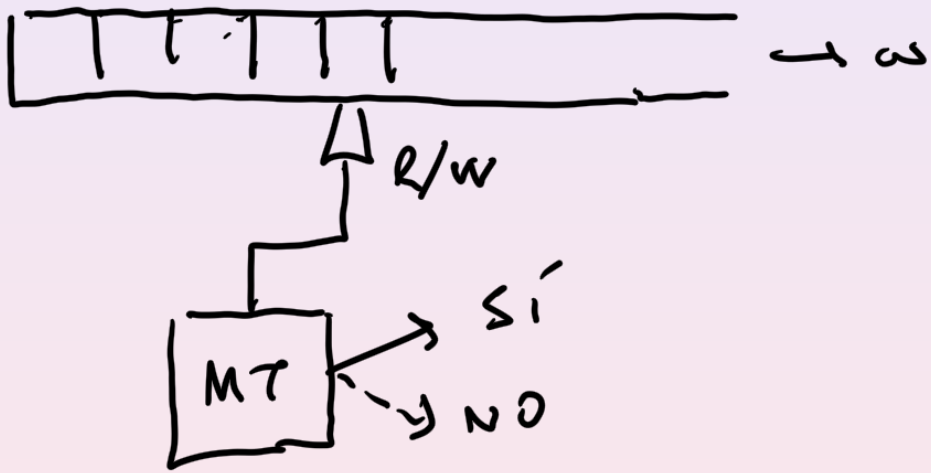
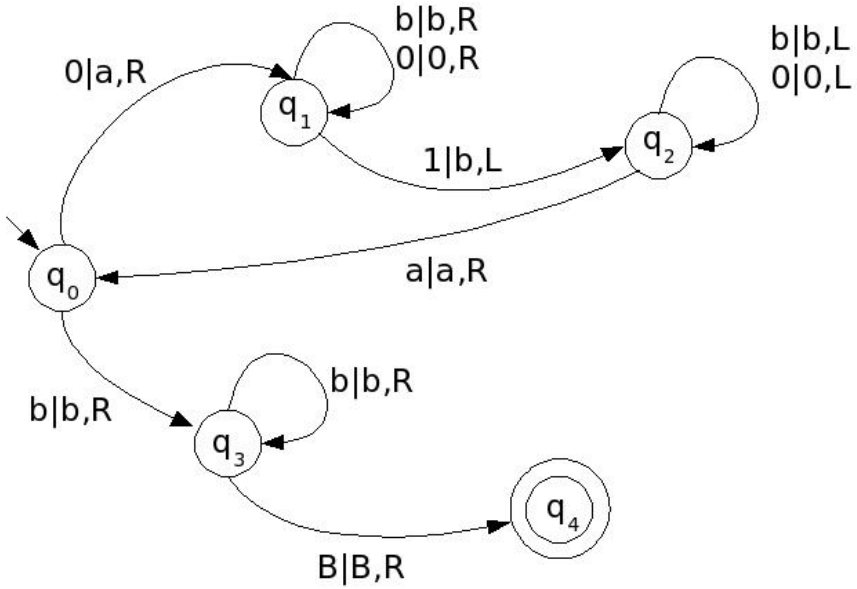

w
R/W
MT
SÍ
y NO


b|b,R
0|0,R
0|a,R
b|b,L
0|0,L
q1
q2
1|b,L
q0
a|a,R
b|b,R
b|b,R
q3
q4
B|B,R

Definition

Autómata linealmente acotado (ALA): es una máquina de Turing no-determinística que cumple con las siguientes condiciones

1. el alfabeto de entrada incluye dos símbolos ¢ y \$, utilizados como topes izquierdo y derecho, respectivamente
2. el ALA no se mueve ni a la izquierda de ¢ ni a la derecha de \$, ni los sobreescribe

Definition

El ALA $M = \langle Q, \Sigma, \Gamma, \delta, q_0, ¢, \$, F \rangle$ acepta el lenguaje L dado por

$$L = \left\{ w : w \in (\Sigma - \{¢, \$\})^* \text{ y } q_0 ¢ w \$ \overset{*}{\underset{M}{\vdash}} \alpha q \beta \text{ para } q \in F \right\}.$$

Theorem

Si L es un lenguaje dependiente del contexto (o sea generado por una gramática G, dependiente del contexto), entonces existe un ALA M tal que $L = \mathcal{L}(M)$.

Demostración.

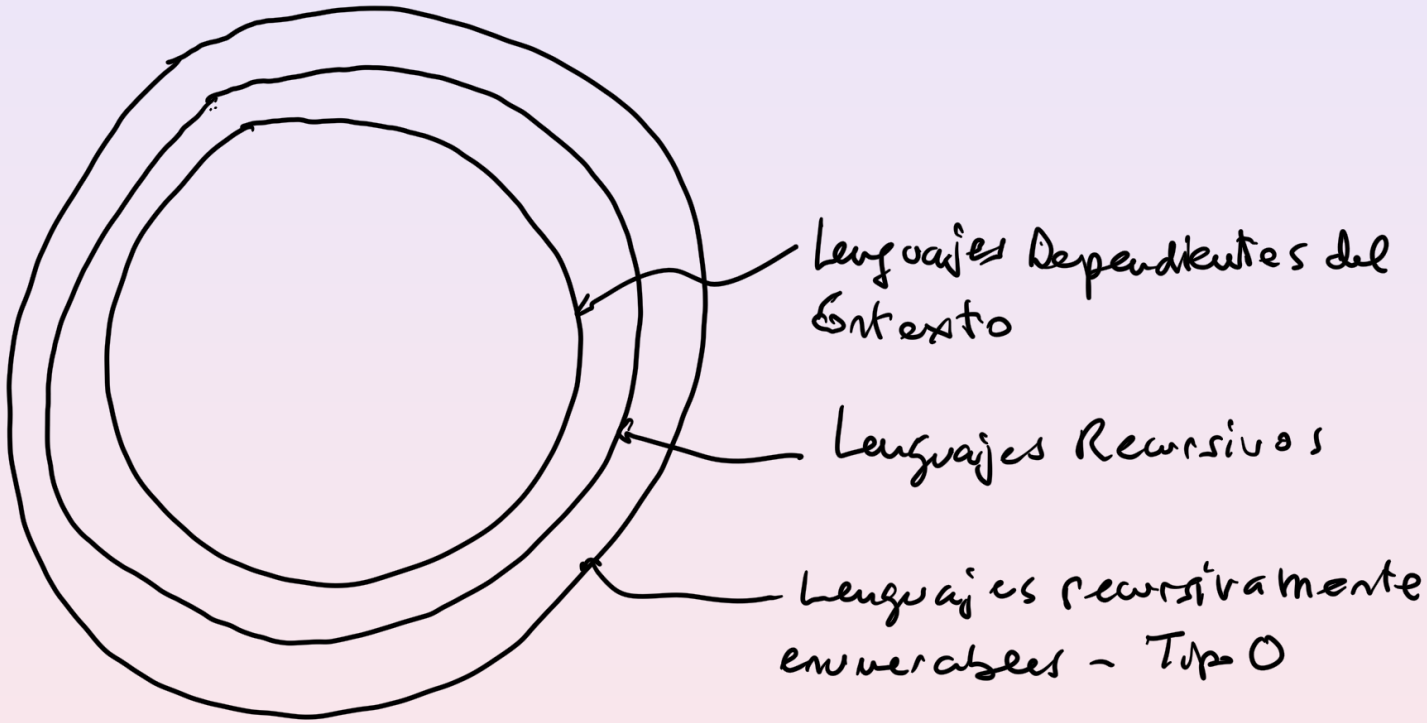
construimos un ALA M que posee dos cintas:

- la primera contiene la cadena de entrada $¢w\$$ que permanece inalterada
- la segunda se utiliza para generar las formas sentenciales de la derivación. En cualquier instante contiene la forma sentencial α de la derivación de la cadena de entrada. Se inicializa con el símbolo distinguido S.

Demostración.

El ALA M opera de la siguiente manera:

1. si $w = \lambda$, entonces M se detiene rechazando la cadena de entrada
2. seleccionar (en forma no-determinística) la posición i dentro de α
3. seleccionar (en forma no-determinística) la producción $\beta \to \gamma \in P$
4. si β aparece a partir de la posicoón i en α, reemplazar β por γ en α
5. si la nueva forma sentencial α es tal que $|\alpha| > |w|$ entonces M se detiene rechazando la cadena de entrada
6. comparar la nueva forma sentencial α resultante con la cadena de entrada w. Si $\alpha = w$ entonces: aceptar w, sino entonces volver al paso 2.


Lenguajes Dependientes del Contexto
Lenguajes Recursivos
Lenguajes recursivamente enumerables - Tipo 0

Theorem

Todo lenguaje dependiente del contexto es recursivo

Demostración.

Dada la gramática dependiente del contexto $G = \langle V_N, \Sigma, P, S \rangle$, podemos formular un algoritmo que determina, para toda cadena $w \in \Sigma^*$, si esta pertenece o no a $\mathcal{L}(G)$.

- Para esto construyamos un grafo finito en el cual existe un nodo por cada cadena de terminales y no terminales de G de longitud entre 1 y $|w|$.
- Pongamos un arco entre dos nodos cualesquiera sii $\alpha \underset{G}{\Rightarrow} \beta$, donde α y β son las cadenas correspondientes al nodo origen y destino, respectivamente.

En este grafo, $S \overset{*}{\underset{G}{\Rightarrow}} w$ sii existe un camino desde el nodo correspondiente s S hasta el nodo correspondiente s w. □

Lemma

Sea $M_1, M_2, \ldots$ una enumeración de un conjunto de máquinas de Turing que paran para todas las entradas. Siempre existe un lenguaje recursivo que no es aceptado por ninguna de ellas, o sea, siempre existe un lenguaje L tal que

- *L es recursivo*
- *y $\forall j, L \neq \mathcal{L}(M_j)$*

Demostración.

Consideremos el lenguaje $L \subseteq \{0,1\}^*$ definido por

$$L = \{w_i : w_i \notin \mathcal{L}(M_i)\},$$

formado por todas las cadenas w_i que son rechazadas por su correspondiente máquina M_i. □

Lemma

Sea $M_1, M_2, \ldots$ una enumeración de un conjunto de máquinas de Turing que paran para todas las entradas. Siempre existe un lenguaje recursivo que no es aceptado por ninguna de ellas, o sea, siempre existe un lenguaje L tal que

- *L es recursivo*
- *y $\forall j, L \neq \mathcal{L}(M_j)$*

Demostración.

Consideremos el lenguaje $L \subseteq \{0,1\}^*$ definido por

$$L = \{w_i : w_i \notin \mathcal{L}(M_i)\},$$

formado por todas las cadenas w_i que son rechazadas por su correspondiente máquina M_i. □

Lemma

Sea $M_1, M_2, \ldots$ una enumeración de un conjunto de máquinas de Turing que paran para todas las entradas. Siempre existe un lenguaje recursivo que no es aceptado por ninguna de ellas, o sea, siempre existe un lenguaje L tal que

- *L es recursivo*
- *y $\forall j, L \neq \mathcal{L}(M_j)$*

Demostración.

Consideremos el lenguaje $L \subseteq \{0,1\}^*$ definido por

$$L = \{w_i : w_i \notin \mathcal{L}(M_i)\},$$

formado por todas las cadenas w_i que son rechazadas por su correspondiente máquina M_i. □

Lemma

Sea $M_1, M_2, \ldots$ una enumeración de un conjunto de máquinas de Turing que paran para todas las entradas. Siempre existe un lenguaje recursivo que no es aceptado por ninguna de ellas, o sea, siempre existe un lenguaje L tal que

- *L es recursivo*
- *y $\forall j, L \neq \mathcal{L}(M_j)$*

Demostración.

Consideremos el lenguaje $L \subseteq \{0,1\}^*$ definido por

$$L = \{w_i : w_i \notin \mathcal{L}(M_i)\},$$

formado por todas las cadenas w_i que son rechazadas por su correspondiente máquina M_i. □

Lemma

Sea $M_1, M_2, \ldots$ una enumeración de un conjunto de máquinas de Turing que paran para todas las entradas. Siempre existe un lenguaje recursivo que no es aceptado por ninguna de ellas, o sea, siempre existe un lenguaje L tal que

- *L es recursivo*
- *y $\forall j, L \neq \mathcal{L}(M_j)$*

Demostración.

Consideremos el lenguaje $L \subseteq \{0,1\}^*$ definido por

$$L = \{w_i : w_i \notin \mathcal{L}(M_i)\},$$

formado por todas las cadenas w_i que son rechazadas por su correspondiente máquina M_i. □

Lemma

Sea $M_1, M_2, \ldots$ una enumeración de un conjunto de máquinas de Turing que paran para todas las entradas. Siempre existe un lenguaje recursivo que no es aceptado por ninguna de ellas, o sea, siempre existe un lenguaje L tal que

- *L es recursivo*
- *y $\forall j, L \neq \mathcal{L}(M_j)$*

Demostración.

Consideremos el lenguaje $L \subseteq \{0,1\}^*$ definido por

$$L = \{w_i : w_i \notin \mathcal{L}(M_i)\},$$

formado por todas las cadenas w_i que son rechazadas por su correspondiente máquina M_i. □

Lemma

Sea $M_1, M_2, \ldots$ una enumeración de un conjunto de máquinas de Turing que paran para todas las entradas. Siempre existe un lenguaje recursivo que no es aceptado por ninguna de ellas, o sea, siempre existe un lenguaje L tal que

- *L es recursivo*
- *y $\forall j, L \neq \mathcal{L}(M_j)$*

Demostración.

Consideremos el lenguaje $L \subseteq \{0,1\}^*$ definido por

$$L = \{w_i : w_i \notin \mathcal{L}(M_i)\},$$

formado por todas las cadenas w_i que son rechazadas por su correspondiente máquina M_i. □

Lemma

Sea $M_1, M_2, \ldots$ una enumeración de un conjunto de máquinas de Turing que paran para todas las entradas. Siempre existe un lenguaje recursivo que no es aceptado por ninguna de ellas, o sea, siempre existe un lenguaje L tal que

- *L es recursivo*
- *y $\forall j, L \neq \mathcal{L}(M_j)$*

Demostración.

Consideremos el lenguaje $L \subseteq \{0,1\}^*$ definido por

$$L = \{w_i : w_i \notin \mathcal{L}(M_i)\},$$

formado por todas las cadenas w_i que son rechazadas por su correspondiente máquina M_i. □

Lemma

Sea $M_1, M_2, \ldots$ una enumeración de un conjunto de máquinas de Turing que paran para todas las entradas. Siempre existe un lenguaje recursivo que no es aceptado por ninguna de ellas, o sea, siempre existe un lenguaje L tal que

- *L es recursivo*
- *y $\forall j, L \neq \mathcal{L}(M_j)$*

Demostración.

Consideremos el lenguaje $L \subseteq \{0,1\}^*$ definido por

$$L = \{w_i : w_i \notin \mathcal{L}(M_i)\},$$

formado por todas las cadenas w_i que son rechazadas por su correspondiente máquina M_i. □

Lemma

Sea $M_1, M_2, \ldots$ una enumeración de un conjunto de máquinas de Turing que paran para todas las entradas. Siempre existe un lenguaje recursivo que no es aceptado por ninguna de ellas, o sea, siempre existe un lenguaje L tal que

- *L es recursivo*
- *y $\forall j, L \neq \mathcal{L}(M_j)$*

Demostración.

Consideremos el lenguaje $L \subseteq \{0,1\}^*$ definido por

$$L = \{w_i : w_i \notin \mathcal{L}(M_i)\},$$

formado por todas las cadenas w_i que son rechazadas por su correspondiente máquina M_i. □

Lemma

Sea $M_1, M_2, \ldots$ una enumeración de un conjunto de máquinas de Turing que paran para todas las entradas. Siempre existe un lenguaje recursivo que no es aceptado por ninguna de ellas, o sea, siempre existe un lenguaje L tal que

- *L es recursivo*
- *y $\forall j, L \neq \mathcal{L}(M_j)$*

Demostración.

Consideremos el lenguaje $L \subseteq \{0,1\}^*$ definido por

$$L = \{w_i : w_i \notin \mathcal{L}(M_i)\},$$

formado por todas las cadenas w_i que son rechazadas por su correspondiente máquina M_i. □

Lemma

Sea $M_1, M_2, \ldots$ una enumeración de un conjunto de máquinas de Turing que paran para todas las entradas. Siempre existe un lenguaje recursivo que no es aceptado por ninguna de ellas, o sea, siempre existe un lenguaje L tal que

- *L es recursivo*
- *y $\forall j, L \neq \mathcal{L}(M_j)$*

Demostración.

Consideremos el lenguaje $L \subseteq \{0,1\}^*$ definido por

$$L = \{w_i : w_i \notin \mathcal{L}(M_i)\},$$

formado por todas las cadenas w_i que son rechazadas por su correspondiente máquina M_i. □

Lemma

Sea $M_1, M_2, \ldots$ una enumeración de un conjunto de máquinas de Turing que paran para todas las entradas. Siempre existe un lenguaje recursivo que no es aceptado por ninguna de ellas, o sea, siempre existe un lenguaje L tal que

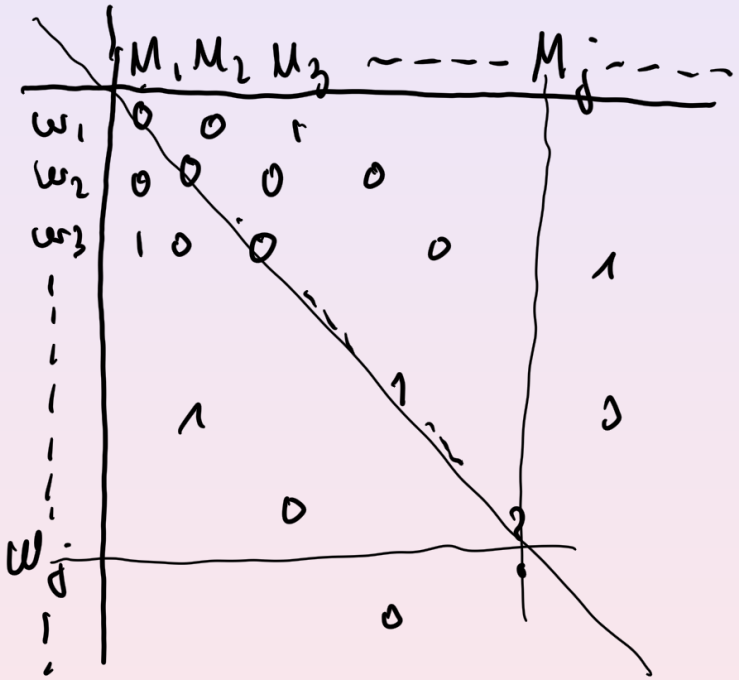
- *L es recursivo*
- *y $\forall j, L \neq \mathcal{L}(M_j)$*

Demostración.

Consideremos el lenguaje $L \subseteq \{0,1\}^*$ definido por

$$L = \{w_i : w_i \notin \mathcal{L}(M_i)\},$$

formado por todas las cadenas w_i que son rechazadas por su correspondiente máquina M_i. □



$0 \Leftrightarrow w_k \notin L(w_i)$

$1 \Leftrightarrow w_k \in L(w_k)$

Demostración (cont.)

Vemos que el lenguaje L así definido es recursivo ya que, para toda cadena w_i, ésta pertenecerá a L sii es rechazada por su correspondiente MT M_i, y la MT M_i siempre para.

Por lo tanto, siempre se puede decir si una cadena pertenece o no pertenece al lenguaje L. □

Demostración (cont.)

Vemos que el lenguaje L así definido es recursivo ya que, para toda cadena w_i, ésta pertenecerá a L sii es rechazada por su correspondiente MT M_i, y la MT M_i siempre para.

Por lo tanto, siempre se puede decir si una cadena pertenece o no pertenece al lenguaje L. □

Demostración (cont.)

Supongamos ahora que este lenguaje L es aceptado por alguna de las MT de la enumeración $M_1, M_2, \ldots$.

Supongamos que esta máquina es la M_j, o sea $L = \mathcal{L}(M_j)$.

Entonces, la cadena w_j pertenece o no pertenece al lenguaje L?

Utilizando la definición del lenguaje L y que $L = \mathcal{L}(M_j)$ tenemos que

$$w_j \in L \Leftrightarrow w_j \notin \mathcal{L}(M_j) \Leftrightarrow w_j \notin L.$$

Esta contradicción surge de considerar que existe un índice j tal que $L = \mathcal{L}(M_j)$.

Por lo tanto L no es ninguno de los lenguajes aceptados por las máquinas $M_1, M_2, \ldots$. □

Demostración (cont.)

Supongamos ahora que este lenguaje L es aceptado por alguna de las MT de la enumeración $M_1, M_2, \ldots$.
Supongamos que esta máquina es la M_j, o sea $L = \mathcal{L}(M_j)$.
Entonces, la cadena w_j pertenece o no pertenece al lenguaje L?
Utilizando la definición del lenguaje L y que $L = \mathcal{L}(M_j)$ tenemos que

$$w_j \in L \Leftrightarrow w_j \notin \mathcal{L}(M_j) \Leftrightarrow w_j \notin L.$$

Esta contradicción surge de considerar que existe un índice j tal que $L = \mathcal{L}(M_j)$.

Por lo tanto L no es ninguno de los lenguajes aceptados por las máquinas $M_1, M_2, \ldots$. □

Demostración (cont.)

Supongamos ahora que este lenguaje L es aceptado por alguna de las MT de la enumeración $M_1, M_2, \ldots$.
Supongamos que esta máquina es la M_j, o sea $L = \mathcal{L}(M_j)$.
Entonces, la cadena w_j pertenece o no pertenece al lenguaje L?
Utilizando la definición del lenguaje L y que $L = \mathcal{L}(M_j)$ tenemos que

$$w_j \in L \Leftrightarrow w_j \notin \mathcal{L}(M_j) \Leftrightarrow w_j \notin L.$$

Esta contradicción surge de considerar que existe un índice j tal que $L = \mathcal{L}(M_j)$.

Por lo tanto L no es ninguno de los lenguajes aceptados por las máquinas $M_1, M_2, \ldots$. □

Demostración (cont.)

Supongamos ahora que este lenguaje L es aceptado por alguna de las MT de la enumeración $M_1, M_2, \ldots$.
Supongamos que esta máquina es la M_j, o sea $L = \mathcal{L}(M_j)$.
Entonces, la cadena w_j pertenece o no pertenece al lenguaje L?

Utilizando la definición del lenguaje L y que $L = \mathcal{L}(M_j)$ tenemos que

$$w_j \in L \Leftrightarrow w_j \notin \mathcal{L}(M_j) \Leftrightarrow w_j \notin L.$$

Esta contradicción surge de considerar que existe un índice j tal que $L = \mathcal{L}(M_j)$.

Por lo tanto L no es ninguno de los lenguajes aceptados por las máquinas $M_1, M_2, \ldots$. □

Demostración (cont.)

Supongamos ahora que este lenguaje L es aceptado por alguna de las MT de la enumeración $M_1, M_2, \ldots$.
Supongamos que esta máquina es la M_j, o sea $L = \mathcal{L}(M_j)$.
Entonces, la cadena w_j pertenece o no pertenece al lenguaje L?
Utilizando la definición del lenguaje L y que $L = \mathcal{L}(M_j)$ tenemos que

$$w_j \in L \Leftrightarrow w_j \notin \mathcal{L}(M_j) \Leftrightarrow w_j \notin L.$$

Esta contradicción surge de considerar que existe un índice j tal que $L = \mathcal{L}(M_j)$.

Por lo tanto L no es ninguno de los lenguajes aceptados por las máquinas $M_1, M_2, \ldots$. □

Demostración (cont.)

Supongamos ahora que este lenguaje L es aceptado por alguna de las MT de la enumeración $M_1, M_2, \ldots$.
Supongamos que esta máquina es la M_j, o sea $L = \mathcal{L}(M_j)$.
Entonces, la cadena w_j pertenece o no pertenece al lenguaje L?
Utilizando la definición del lenguaje L y que $L = \mathcal{L}(M_j)$ tenemos que

$$w_j \in L \Leftrightarrow w_j \notin \mathcal{L}(M_j) \Leftrightarrow w_j \notin L.$$

Esta contradicción surge de considerar que existe un índice j tal que $L = \mathcal{L}(M_j)$.

Por lo tanto L no es ninguno de los lenguajes aceptados por las máquinas $M_1, M_2, \ldots$. □

Demostración (cont.)

Supongamos ahora que este lenguaje L es aceptado por alguna de las MT de la enumeración $M_1, M_2, \ldots$.
Supongamos que esta máquina es la M_j, o sea $L = \mathcal{L}(M_j)$.
Entonces, la cadena w_j pertenece o no pertenece al lenguaje L?
Utilizando la definición del lenguaje L y que $L = \mathcal{L}(M_j)$ tenemos que

$$w_j \in L \Leftrightarrow w_j \notin \mathcal{L}(M_j) \Leftrightarrow w_j \notin L.$$

Esta contradicción surge de considerar que existe un índice j tal que $L = \mathcal{L}(M_j)$.

Por lo tanto L no es ninguno de los lenguajes aceptados por las máquinas $M_1, M_2, \ldots$. □

Demostración (cont.)

Supongamos ahora que este lenguaje L es aceptado por alguna de las MT de la enumeración $M_1, M_2, \ldots$.
Supongamos que esta máquina es la M_j, o sea $L = \mathcal{L}(M_j)$.
Entonces, la cadena w_j pertenece o no pertenece al lenguaje L?
Utilizando la definición del lenguaje L y que $L = \mathcal{L}(M_j)$ tenemos que

$$w_j \in L \Leftrightarrow w_j \notin \mathcal{L}(M_j) \Leftrightarrow w_j \notin L.$$

Esta contradicción surge de considerar que existe un índice j tal que $L = \mathcal{L}(M_j)$.

Por lo tanto L no es ninguno de los lenguajes aceptados por las máquinas $M_1, M_2, \ldots$. □

Demostración (cont.)

Supongamos ahora que este lenguaje L es aceptado por alguna de las MT de la enumeración $M_1, M_2, \ldots$.
Supongamos que esta máquina es la M_j, o sea $L = \mathcal{L}(M_j)$.
Entonces, la cadena w_j pertenece o no pertenece al lenguaje L?
Utilizando la definición del lenguaje L y que $L = \mathcal{L}(M_j)$ tenemos que

$$w_j \in L \Leftrightarrow w_j \notin \mathcal{L}(M_j) \Leftrightarrow w_j \notin L.$$

Esta contradicción surge de considerar que existe un índice j tal que $L = \mathcal{L}(M_j)$.

Por lo tanto L no es ninguno de los lenguajes aceptados por las máquinas $M_1, M_2, \ldots$. □

Demostración (cont.)

Supongamos ahora que este lenguaje L es aceptado por alguna de las MT de la enumeración $M_1, M_2, \ldots$.
Supongamos que esta máquina es la M_j, o sea $L = \mathcal{L}(M_j)$.
Entonces, la cadena w_j pertenece o no pertenece al lenguaje L?
Utilizando la definición del lenguaje L y que $L = \mathcal{L}(M_j)$ tenemos que

$$w_j \in L \Leftrightarrow w_j \notin \mathcal{L}(M_j) \Leftrightarrow w_j \notin L.$$

Esta contradicción surge de considerar que existe un índice j tal que $L = \mathcal{L}(M_j)$.

Por lo tanto L no es ninguno de los lenguajes aceptados por las máquinas $M_1, M_2, \ldots$. □

Lemma

Existe un lenguaje recursivo que no es dependiente del contexto

Demostración.

Se trata de demostrar que podemos encontrar una enumeración de máquinas de Turing que paran en todas las entradas, correspondientes a cada uno de los lenguajes dependientes del contexto, definidos sobre $\{0, 1\}^*$.

- Codifiquemos todas las gramáticas sensitivas al contexto por cadenas binarias. Por ejemplo: codificando 0, 1, $,$, $\rightarrow$, $\{$, $\}$, $($, $)$ como 10, 100, $\ldots$, 10^8, y codificando el k-ésimo no-terminal como 10^{k+8}.

Demostración.

- al codificar cada una de las gramáticas dependientes del contexto mediante cadenas de 1s y 0s podemos enumerarlas: $G_1, G_2, \ldots$.
- existe un algoritmo que nos permite obtener una MT que para para todas las entradas a partir de un gramática dependiente del contexto.
  Aplicando este algoritmo a cada una de mas gramáticas $G_1, G_2, \ldots$, obtenemos una sucesión $M_1, M_2, \ldots$ de MTs que paran en para todas las entradas, tales que

$$\forall i, \mathcal{L}(M_i) = \mathcal{L}(G_i).$$

Demostración.

- al codificar cada una de las gramáticas dependientes del contexto mediante cadenas de 1s y 0s podemos enumerarlas: $G_1, G_2, \ldots$.
- existe un algoritmo que nos permite obtener una MT que para para todas las entradas a partir de un gramática dependiente del contexto.
  Aplicando este algoritmo a cada una de mas gramáticas $G_1, G_2, \ldots$, obtenemos una sucesión $M_1, M_2, \ldots$ de MTs que paran en para todas las entradas, tales que

$$\forall i, \mathcal{L}(M_i) = \mathcal{L}(G_i).$$

Demostración.

- al codificar cada una de las gramáticas dependientes del contexto mediante cadenas de 1s y 0s podemos enumerarlas: $G_1, G_2, \ldots$.
- existe un algoritmo que nos permite obtener una MT que para para todas las entradas a partir de un gramática dependiente del contexto.

  Aplicando este algoritmo a cada una de mas gramáticas $G_1, G_2, \ldots$, obtenemos una sucesión $M_1, M_2, \ldots$ de MTs que paran en para todas las entradas, tales que

$$\forall i, \mathcal{L}(M_i) = \mathcal{L}(G_i).$$

Demostración.

- al codificar cada una de las gramáticas dependientes del contexto mediante cadenas de 1s y 0s podemos enumerarlas: $G_1, G_2, \ldots$.
- existe un algoritmo que nos permite obtener una MT que para para todas las entradas a partir de un gramática dependiente del contexto.
  Aplicando este algoritmo a cada una de mas gramáticas $G_1, G_2, \ldots$, obtenemos una sucesión $M_1, M_2, \ldots$ de MTs que paran en para todas las entradas, tales que

$$\forall i, \mathcal{L}(M_i) = \mathcal{L}(G_i).$$

Demostración.

- al codificar cada una de las gramáticas dependientes del contexto mediante cadenas de 1s y 0s podemos enumerarlas: $G_1, G_2, \ldots$.
- existe un algoritmo que nos permite obtener una MT que para para todas las entradas a partir de un gramática dependiente del contexto.
  Aplicando este algoritmo a cada una de mas gramáticas $G_1, G_2, \ldots$, obtenemos una sucesión $M_1, M_2, \ldots$ de MTs que paran en para todas las entradas, tales que

$$\forall i, \mathcal{L}(M_i) = \mathcal{L}(G_i).$$

Demostración.

- al codificar cada una de las gramáticas dependientes del contexto mediante cadenas de 1s y 0s podemos enumerarlas: $G_1, G_2, \ldots$.
- existe un algoritmo que nos permite obtener una MT que para para todas las entradas a partir de un gramática dependiente del contexto.
  Aplicando este algoritmo a cada una de mas gramáticas $G_1, G_2, \ldots$, obtenemos una sucesión $M_1, M_2, \ldots$ de MTs que paran en para todas las entradas, tales que

$$\forall i, \mathcal{L}(M_i) = \mathcal{L}(G_i).$$

Demostración (cont.)

- Luego, aplicando el lema anterior a esta enumeración a las MTs $M_1, M_2, \ldots$ obtenidas a partir de las graáticas $G_1, G_2, \ldots$, tenemos entonces que existe un lenguaje recursivo L, dado por,

$$L = \{w_i : w_i \notin \mathcal{L}(M_i) = \mathcal{L}(G_i) \text{ con } G_i \text{ GDC sobre } \{0,1\}^*\},$$

que no es dependiente del contexto.

Demostración (cont.)

- Luego, aplicando el lema anterior a esta enumeración a las MTs $M_1, M_2, \ldots$ obtenidas a partir de las graáticas $G_1, G_2, \ldots$, tenemos entonces que existe un lenguaje recursivo L, dado por,

$$L = \{w_i : w_i \notin \mathcal{L}(M_i) = \mathcal{L}(G_i) \text{ con } G_i \text{ GDC sobre } \{0,1\}^*\},$$

que no es dependiente del contexto.

Demostración (cont.)

- Luego, aplicando el lema anterior a esta enumeración a las MTs $M_1, M_2, \ldots$ obtenidas a partir de las graáticas $G_1, G_2, \ldots$, tenemos entonces que existe un lenguaje recursivo L, dado por,

$$L = \{w_i : w_i \notin \mathcal{L}(M_i) = \mathcal{L}(G_i) \text{ con } G_i \text{ GDC sobre } \{0,1\}^*\},$$

que no es dependiente del contexto.

Demostración (cont.)

- Luego, aplicando el lema anterior a esta enumeración a las MTs $M_1, M_2, \ldots$ obtenidas a partir de las graáticas $G_1, G_2, \ldots$, tenemos entonces que existe un lenguaje recursivo L, dado por,

$$L = \{w_i : w_i \notin \mathcal{L}(M_i) = \mathcal{L}(G_i) \text{ con } G_i \text{ GDC sobre } \{0,1\}^*\},$$

que no es dependiente del contexto.

Demostración (cont.)

- Luego, aplicando el lema anterior a esta enumeración a las MTs $M_1, M_2, \ldots$ obtenidas a partir de las graáticas $G_1, G_2, \ldots$, tenemos entonces que existe un lenguaje recursivo L, dado por,

$$L = \{w_i : w_i \notin \mathcal{L}(M_i) = \mathcal{L}(G_i) \text{ con } G_i \text{ GDC sobre } \{0,1\}^*\},$$

que no es dependiente del contexto.

□

Demostración (cont.)

- Luego, aplicando el lema anterior a esta enumeración a las MTs $M_1, M_2, \ldots$ obtenidas a partir de las graáticas $G_1, G_2, \ldots$, tenemos entonces que existe un lenguaje recursivo L, dado por,

$$L = \{w_i : w_i \notin \mathcal{L}(M_i) = \mathcal{L}(G_i) \text{ con } G_i \text{ GDC sobre } \{0,1\}^*\},$$

que no es dependiente del contexto.

□

Demostración (cont.)

- Luego, aplicando el lema anterior a esta enumeración a las MTs $M_1, M_2, \ldots$ obtenidas a partir de las graáticas $G_1, G_2, \ldots$, tenemos entonces que existe un lenguaje recursivo L, dado por,

$$L = \{w_i : w_i \notin \mathcal{L}(M_i) = \mathcal{L}(G_i) \text{ con } G_i \text{ GDC sobre } \{0,1\}^*\},$$

que no es dependiente del contexto.

Demostración (cont.)

- Luego, aplicando el lema anterior a esta enumeración a las MTs $M_1, M_2, \ldots$ obtenidas a partir de las graáticas $G_1, G_2, \ldots$, tenemos entonces que existe un lenguaje recursivo L, dado por,

$$L = \{w_i : w_i \notin \mathcal{L}(M_i) = \mathcal{L}(G_i) \text{ con } G_i \text{ GDC sobre } \{0,1\}^*\},$$

que no es dependiente del contexto.

Demostración (cont.)

- Luego, aplicando el lema anterior a esta enumeración a las MTs $M_1, M_2, \ldots$ obtenidas a partir de las graáticas $G_1, G_2, \ldots$, tenemos entonces que existe un lenguaje recursivo L, dado por,

$$L = \{w_i : w_i \notin \mathcal{L}(M_i) = \mathcal{L}(G_i) \text{ con } G_i \text{ GDC sobre } \{0,1\}^*\},$$

que no es dependiente del contexto.

Demostración (cont.)

- Luego, aplicando el lema anterior a esta enumeración a las MTs $M_1, M_2, \ldots$ obtenidas a partir de las graáticas $G_1, G_2, \ldots$, tenemos entonces que existe un lenguaje recursivo L, dado por,

$$L = \{w_i : w_i \notin \mathcal{L}(M_i) = \mathcal{L}(G_i) \text{ con } G_i \text{ GDC sobre } \{0,1\}^*\},$$

que no es dependiente del contexto.

Definition

Una máquina de Turing (MT) es un autómata M definido por

$$M = \langle Q, \Sigma, \Gamma, \delta, q_0, B, F \rangle ,$$

donde

- Q es un conjunto finito de estados
- Γ es un conjunto finito de símbolos de cinta
- $B \in \Gamma$ es blanco
- $\Sigma \subset \Gamma$ es el conjunto de símbolos de entrada.$B \notin \Sigma$
- $\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, R\}$ es la función de transición
- $q_0 \in Q$ es el estado inicial
- $F \subseteq Q$ es el conjunto de estados finales

Definition

Una configutación instanánea de una máquina de Turing está dada por $\alpha_1 q \alpha_2$, donde $q \in Q$ y $\alpha_1, \alpha_2 \in \Gamma^*$, donde

- α_1 es el contenido de la cinta desde su primera posición hasta la posición a la izquierda de la cabeza lectora-escritora, y
- α_2 es el contenido de la cinta desde la posición de dicha cabeza hasta la posición del símbolo distinto de B más a la derecha en la cinta o hasta la posición a la izquierda de la cabeza, la que quede más a la derecha.

Definition

Las transiciones de una máquna de Turing M son tales que, si M está en una configuración dada por $X_1 \dots X_{i-1} q X_i \dots X_n$, o sea, con la cabeza lectora-escritora en la posición i, entonces es cierto que:

- si $\delta(q, X_i) = (p, Y, L)$ entonces

$$X_1 \dots X_{i-1} q X_i \dots X_n \underset{M}{\vdash} X_1 \dots X_{i-2} p X_{i-1} Y \dots X_n,$$

si $1 < i \leq n-1$, y

$$X_1 \dots\dots X_n q B \underset{M}{\vdash} X_1 \dots\dots p X_n Y,$$

si $i = n+1$. Si $i = 1$, no hay movimiento posible hacia la izquierda.

Definition

- si $\delta(q, X_i) = (p, Y, R)$ entonces

$$X_1 \dots qX_iX_{i+1} \dots X_n \underset{M}{\vdash} X_1 \dots YpX_{i+1} \dots X_n,$$

si $1 \leq i \leq n-1$, y

$$X_1 \dots\dots X_nqBB \underset{M}{\vdash} X_1 \dots\dots X_nYpB,$$

si $i = n+1$

Definition

El lenguaje aceptado por una máquina de Turing M se define como

$$\mathcal{L}(M) = \left\{ w \in \Sigma^* : \ q_0 w \overset{*}{\underset{M}{\vdash}} \alpha_1 p \alpha_2 \text{ con } p \in F \text{ y } \alpha_1, \alpha_2 \in \Gamma^* \right\}.$$

Theorem

Sea M una MT no-determinística. Sea L el lenguaje reconocido por M, o sea $L = \mathcal{L}(M)$.
Existe una MT determinística M' tal que $L = \mathcal{L}(M')$.

Demostración.

- Para cada estado existe una cantidad de transiciones que parten de él. Llamemos r a la máxima cantidad de transiciones que parten de un estado en una MT M dada.
- Para cada estado, numeremos las transiciones que parten de él entre 1 y r, como máximo.
- Entonces toda sequencia finita de enteros entre 1 y r puede ser interpretada como una secuencia de transiciones en la MT M, partiendo desde el estado inicial q_0. Tener en cuenta que, algunas de estas secuencias no serán ejecutables por no existir alguna o varias de sus transiciones.

Theorem

Sea M una MT no-determinística. Sea L el lenguaje reconocido por M, o sea $L = \mathcal{L}(M)$.
Existe una MT determinística M' tal que $L = \mathcal{L}(M')$.

Demostración.

- Para cada estado existe una cantidad de transiciones que parten de él. Llamemos r a la máxima cantidad de transiciones que parten de un estado en una MT M dada.
- Para cada estado, numeremos las transiciones que parten de él entre 1 y r, como máximo.
- Entonces toda sequencia finita de enteros entre 1 y r puede ser interpretada como una secuencia de transiciones en la MT M, partiendo desde el estado inicial q_0. Tener en cuenta que, algunas de estas secuencias no serán ejecutables por no existir alguna o varias de sus transiciones.

Theorem

Sea M una MT no-determinística. Sea L el lenguaje reconocido por M, o sea $L = \mathcal{L}(M)$.
Existe una MT determinística M' tal que $L = \mathcal{L}(M')$.

Demostración.

- Para cada estado existe una cantidad de transiciones que parten de él. Llamemos r a la máxima cantidad de transiciones que parten de un estado en una MT M dada.
- Para cada estado, numeremos las transiciones que parten de él entre 1 y r, como máximo.
- Entonces toda sequencia finita de enteros entre 1 y r puede ser interpretada como una secuencia de transiciones en la MT M, partiendo desde el estado inicial q_0. Tener en cuenta que, algunas de estas secuencias no serán ejecutables por no existir alguna o varias de sus transiciones.

Demostración.

- Construyamos M' con 3 cintas:
  - la primera contendrá la cadena de entrada, la cual no será alterada,
  - la segunda contendrá la secuencia de enteros entre 1 y r que se analiza,
  - la tercera servirá para simular la MT no-determinística M.
- generemos secuencias de enteros con valores entre 1 y r en orden creciente de longitud y alfabético hasta aceptar la cadena de la cinta 1.
  - para cada secuencia, borremos el contenido de la cinta 3, y luego copiemos el contenido de la cinta 1 en la cinte 3. Simulemos la MT M en la cinta 3 ejecutando la secuencia de transiciones en curso especificada en la cinta 2
  - aceptar la cadena en la cinta 1 si se acepta esa cadena en la simulación.

Theorem

Si para la gramática sin restricciones $G = \langle V_N, V_T, P, S \rangle$ *es* $L = \mathcal{L}(G)$*, entonces existe una Máquina de Turing (MT)* M *tal que* $L = \mathcal{L}(M)$*.*

Demostración.

Construyamos una MT no-determinística de dos cintas:

- la primera contiene la cadena de entrada w
- la segunda contiene la forma sentencial α de la derivación de la cadena de entrada. Se inicializa con el símbolo distinguido S.

Demostración.

La MT M opera de la siguiente manera:

1. seleccionar (en forma no-determinística) la posición i dentro de α
2. seleccionar (en forma no-determinística) la producción $\beta \to \gamma \in P$
3. si β aparece a partir de la posición i en α, reemplazar β por γ en α
4. comparar la nueva forma sentencial α resultante con la cadena de entrada w. Si $\alpha = w$ entonces: aceptar w, sino volver al paso 1.

Theorem

Si una la Máquina de Turing $M = \langle Q, \Sigma, \Gamma, \delta, q_0, B, F \rangle$ acepta el lenguaje L, o sea, $L = \mathcal{L}(M)$, entonces existe una gramática sin restricciones G que genera el mismo lenguaje ($L = \mathcal{L}(G)$).

Demostración.

La idea es que G genere dos copias de alguna representación de la cadena de entrada, y que luego simule la operación de la MT M sobre una de ellas. Si esta simulación resulta en una aceptación, entonces la primera copia (que es una representación de la cadena de entrada y que está aún intacta) se convierte en la cadena de entrada propiamente dicha. □

La gramática $G = \langle V_N, \Sigma, P, A_1 \rangle$, con $V_N = ((\Sigma \cup \{\lambda\}) \times \Gamma) \cup \{A_1, A_2, A_3\}$ y conjunto de producciones P dado por

1. $A_1 \to q_0 A_2$
2. $A_2 \to [a, a]\, A_2$ para cada $a \in \Sigma$
3. $A_2 \to A_3$
4. $A_3 \to [\lambda, B]\, A_3$
5. $A_3 \to \lambda$
6. $q\,[a, X] \to [a, Y]\, p$, para todo $a \in \Sigma \cup \{\lambda\}$, $q \in Q$ y $X, Y \in \Gamma$, tales que $\delta\,(q, X) = (p, Y, R)$
7. $[b, Z]\, q\,[a, X] \to p\,[b, Z]\,[a, Y]$, para todo $a, b \in \Sigma \cup \{\lambda\}$, $q \in Q$ y $X, Y, Z \in \Gamma$, tales que $\delta\,(q, X) = (p, Y, L)$
8. $[a, X]\, q \to qaq$, $q\,[a, X] \to qaq$, $q \to \lambda$, para todo $a \in \Sigma \cup \{\lambda\}$, $q \in F$ y $X \in \Gamma$.

utilizando las reglas 1 y 2 se puede generar

$$A_1 \stackrel{*}{\Rightarrow} q_0 \, [a_1, a_1] \ldots [a_n, a_n] \, A_2,$$

luego utilizando la regla 3 se generan los símbolos correspondientes a los espacios en blanco necesarios para el análisis de la cadena de entrada en la MT M

$$A_1 \stackrel{*}{\Rightarrow} q_0 \, [a_1, a_1] \ldots [a_n, a_n] \, [\lambda, B]^m \, A_3.$$

Utilizando las reglas 6 y 7 se simula la operación de M sobre las segundas componentes, dejando intactas las primeras componentes.

Puede demostrarse que

$$q_0 a_1 \dots a_n \overset{*}{\underset{M}{\vdash}} X_1 \dots X_{r-1} q X_r \dots X_s \quad \Rightarrow$$

$$q_0 \left[a_1, a_1\right] \dots \left[a_n, a_n\right] \left[\lambda, B\right]^m \overset{*}{\underset{G}{\Rightarrow}} \left[a_1, X_1\right] \dots \left[a_{r-1}, X_{r-1}\right] q \left[a_r, X_r\right] \dots \left[a_{n+m}\right.$$

con $a_1, \dots, a_n \in \Sigma$, $a_{n+1} = \dots = a_{n+m} = \lambda$, $X_1, \dots, X_{n+m} \in \Gamma$, y $X_{s+1} = \dots = X_{n+m} = B$. Lo que deseamos probar es cierto cuando la cantidad de transiciones en M es cero, por que en este caso tenemos $r = 1$ y $s = n$, o sea

$$q_0 a_1 \dots a_n \overset{0}{\underset{M}{\vdash}} q a_1 \dots a_n \quad \Rightarrow$$

$$q_0 \left[a_1, a_1\right] \dots \left[a_n, a_n\right] \left[\lambda, B\right]^m \overset{*}{\underset{G}{\Rightarrow}} q_0 \left[a_1, a_1\right] \dots \left[a_n, a_n\right] \left[\lambda, B\right]^m,$$

ya que antecedente y consecuente son verdaderos.

Tomemos ahora el caso de k transiciones

$$q_0 a_1 \dots a_n \overset{k-1}{\underset{M}{\vdash}} X_1 \dots X_{r-1} q X_r \dots X_s \underset{M}{\vdash} Y_1 \dots Y_{t-1} p Y_t \dots Y_u,$$

por hipótesis inductiva tenemos que

$$q_0 [a_1, a_1] \dots [a_n, a_n] [\lambda, B]^m \overset{*}{\underset{G}{\Rightarrow}} [a_1, a_1] \dots [a_{r-1}, X_{r-1}] q [a_r, X_r] \dots [a_{n+r}$$

Si en el paso k el movimiento es hacia la derecha, o sea $\delta(q, X_r) = (p, Y_r, R)$, tenemos que $t = r + 1$, y por la regla 6 sabemos que $q [a_r, X_r] \to [a_r, Y_r] p \in P$, por lo que

$$[a_1, a_1] \dots [a_{r-1}, X_{r-1}] q [a_r, X_r] \dots [a_{n+m}, X_{n+m}] \overset{*}{\underset{G}{\Rightarrow}}$$
$$[a_1, a_1] \dots [a_r, Y_r] p [a_{r+1}, X_{r+1}] \dots [a_{n+m}, X_{n+m}] =$$
$$[a_1, a_1] \dots [a_{t-1}, Y_{t-1}] p [a_t, Y_r] \dots [a_{n+m}, Y_{n+m}]$$