

Tema 2: Autómatas Finitos y Expresiones Regulares

Serafín Moral

Universidad de Granada

Octubre, 2012

- Autómata Finito Determinista
- Autómata Finito No-Determinista
- Autómata Finito con Transiciones Nulas
- Expresiones Regulares
- Gramáticas Regulares

Importancia de los autómatas finitos

Son importantes en las siguientes tareas:

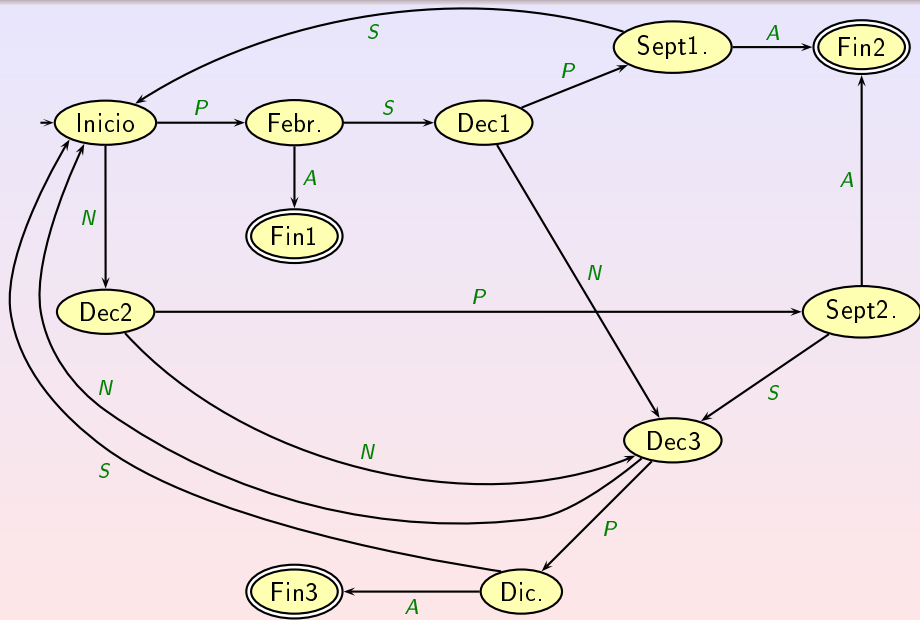
- Software para el diseño y verificación de circuitos digitales.
- Construcción de analizadores léxicos de compiladores.
- Software para analizar grandes conjuntos de textos para buscar palabras, estructuras u otros patrones (p.e. páginas web).
- Software para comprobar la corrección de cualquier tipo de sistemas que tengan un número finito de estados diferentes (p.e. protocolos de comunicación).

Vamos a diseñar un autómata que reconozca el paso de un alumno por una asignatura, por ejemplo, Modelos de Computación I.

El alfabeto de entrada contendrá los siguientes elementos:

- *P*: El alumno se presenta a un examen.
- *N*: El alumno no se presenta a un examen.
- *A*: El alumno aprueba un examen.
- *S*: El alumno suspende un examen.

Ejemplo Introdutorio



AUTOMATA FINITO DETERMINISTA

Un **autómata finito** es una quintupla $M = (Q, A, \delta, q_0, F)$ donde

- Q es un conjunto finito llamado **conjunto de estados**
- A es un alfabeto llamado **alfabeto de entrada**
- δ es una aplicación llamada **función de transición**

$$\delta: Q \times A \rightarrow Q$$

- q_0 es un elemento de Q , llamado **estado inicial**
- F es un subconjunto de Q , llamado conjunto de **estados finales**.

Sea el autómata $M = (Q, A, q_0, \delta, F)$, donde

- $Q = \{q_0, q_1, q_2\}$
- $A = \{a, b\}$
- La función de transición viene dada por:

$$\delta(q_0, a) = q_1, \quad \delta(q_0, b) = q_2,$$

$$\delta(q_1, a) = q_1, \quad \delta(q_1, b) = q_2,$$

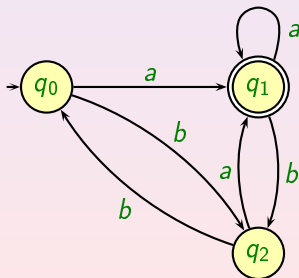
$$\delta(q_2, a) = q_1, \quad \delta(q_2, b) = q_0$$

- $F = \{q_1\}$

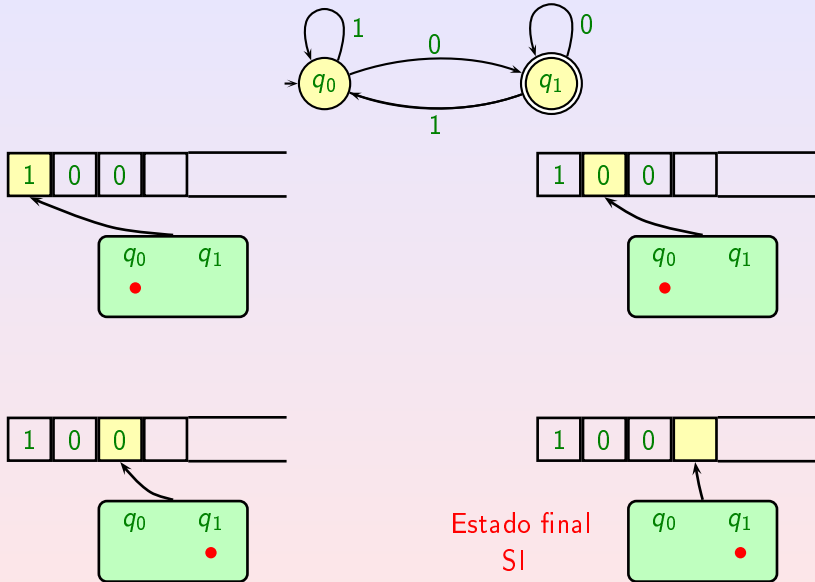
Diagrama de Transición

Es un grafo en el que:

- Hay un nodo por cada estado
- Por cada transición $\delta(q, a) = p$ hay un arco de q a p con la etiqueta a .
- El estado inicial está indicado con un ángulo entrante. Los estados finales están indicados con una doble circunferencia.



Cálculo Asociado. Trazas



Autómata $M = (Q, A, \delta, q_0, F)$

- **Descripción Instantánea o Configuración:**
Un elemento de $Q \times A^*$: (q, u) .
- **Configuración Inicial** para $u \in A^*$: (q_0, u)
- **Relación paso de cálculo** entre dos configuraciones:

$$((q, au) \vdash (p, u)) \Leftrightarrow \delta(q, a) = p$$

De una configuración sólo se puede pasar a lo máximo a una configuración en un paso de cálculo.

- **Relación de cálculo** entre dos configuraciones:

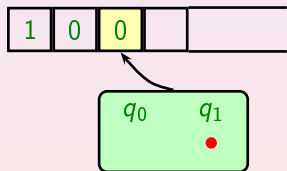
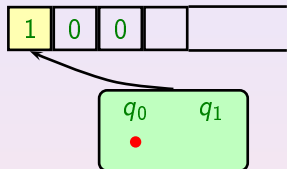
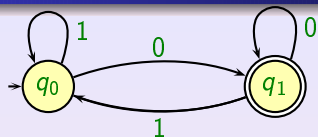
$((q, u) \stackrel{*}{\vdash} (p, v))$ si y solo si existe una sucesión de configuraciones C_0, \dots, C_n tales que $C_0 = (q, u)$, $C_n = (p, v)$ y $\forall i \leq n-1, C_i \vdash C_{i+1}$.

Lenguaje Aceptado por un Autómata Finito

$$L(M) = \{u \in A^* : (q_0, u) \stackrel{*}{\vdash} (q, \varepsilon), q \in F\}$$

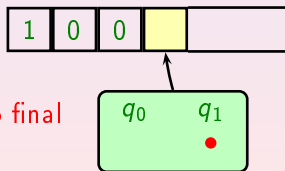
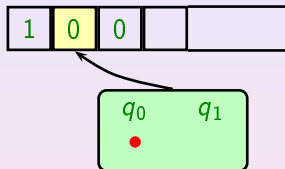
Las palabras de $L(M)$ se dicen **aceptadas por el autómata**.

Ejemplo. Cálculo Asociado



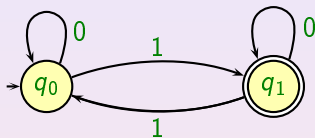
$$(q_0, 100) \vdash (q_0, 00) \vdash (q_1, 0) \vdash (q_1, \varepsilon)$$

$$(q_0, 100) \stackrel{*}{\vdash} (q_1, \varepsilon), 100 \text{ aceptada}$$



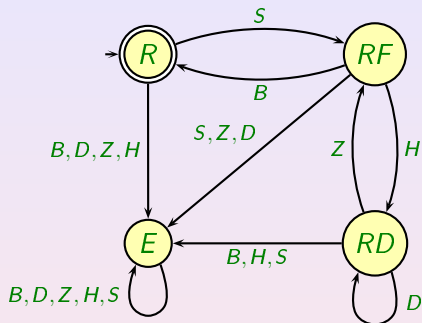
Estado final
SI

Ejemplo



Acepta el conjunto de palabras con un número impar de 1.

Comunicaciones Correctas



R: Estado de espera

RD: Recepción de datos

S: Comienza recepción

H: Cabecera de fichero

D: Datos

RF: Estado de recepción de ficheros

E: Error

B: Fin de recepción

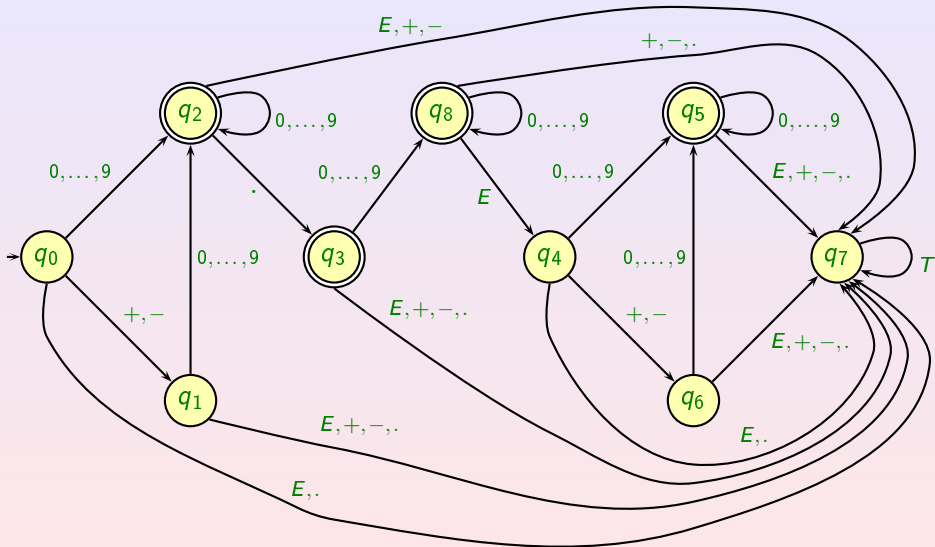
Z: Fin de fichero

Gramática $G = (V, T, P, S)$, donde

- $T = \{+, -, E, 0, 1, \dots, 9, .\}$
- $V = \{< \text{Signo} >, < \text{Digito} >, < \text{Natural} >, < \text{Entero} >, < \text{Real} >\}$
- $S = < \text{Real} >$
- P contiene las siguientes producciones
 - $< \text{Signo} > \rightarrow + | -$
 - $< \text{Digito} > \rightarrow 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9$
 - $< \text{Natural} > \rightarrow < \text{Digito} > | < \text{Digito} > < \text{Natural} >$
 - $< \text{Entero} > \rightarrow < \text{Natural} > | < \text{Signo} > < \text{Natural} >$
 - $< \text{Real} > \rightarrow < \text{Entero} > | < \text{Entero} > .$
 - $< \text{Real} > \rightarrow < \text{Entero} > . < \text{Natural} >$
 - $< \text{Real} > \rightarrow < \text{Entero} > . < \text{Natural} > E < \text{Entero} >$

Autómata Finito

T es el conjunto de los símbolos terminales.



Autómatas Finitos No Deterministas

Un autómata finito no determinista es una quintupla

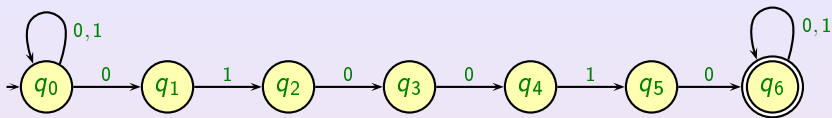
$M = (Q, A, \delta, q_0, F)$ en la que

- Q es un conjunto finito llamado **conjunto de estados**
- A es un alfabeto llamado **alfabeto de entrada**
- δ es una aplicación llamada **función de transición**

$$\delta: Q \times A \rightarrow \wp(Q)$$

- q_0 es un elemento de Q , llamado **estado inicial**
- F es un subconjunto de Q , llamado conjunto de **estados finales**.

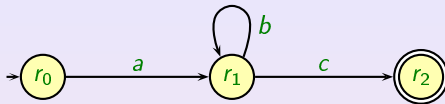
Ejemplo



- Se pueden usar también diagramas de transición.
- Puede haber estados que para una entrada tenga dos transiciones. Por ejemplo, q_0 cuando lee 0 puede quedarse en q_0 o pasar a q_1 .
- También puede haber estados que para una entrada no tengan ninguna transición: desde q_1 no se puede leer el 0 .
- Acepta el conjunto de las palabras que tienen a 010010 como subcadena: palabras que se **pueden** leer pasando de q_0 a un estado final.

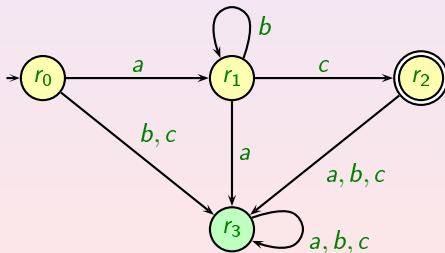
Ejemplos

También es no-determinista:



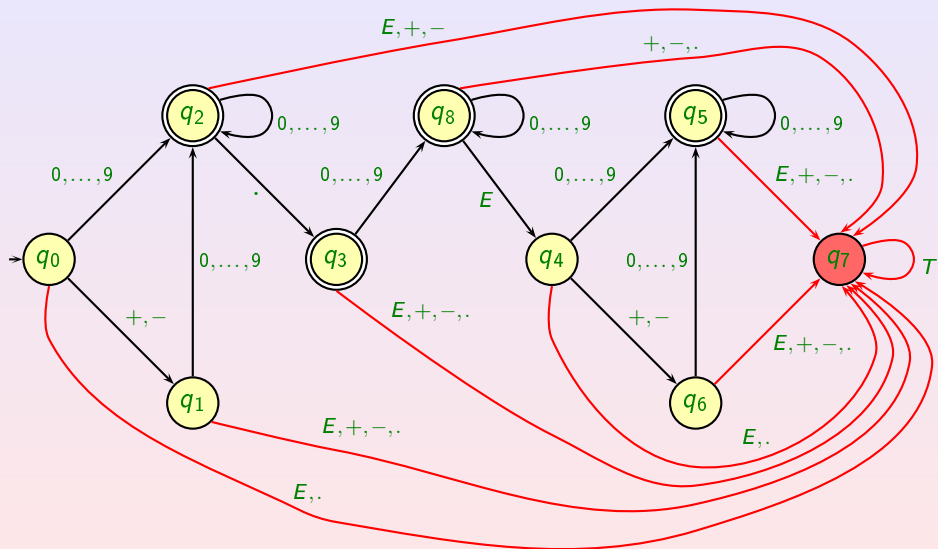
Acepta cadenas formadas por una a , una sucesión de b y una c .

Se puede transformar en uno determinista que acepte el mismo lenguaje añadiéndole un **estado de error** donde vayan todas las transiciones no definidas en el autómata anterior:



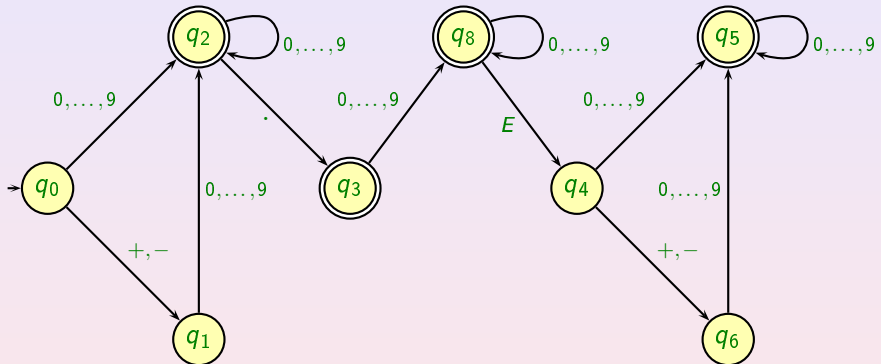
Ejemplo: Constantes reales

Autómata determinista:



Ejemplo: Constantes reales

Autómata no determinista:



Autómata no determinista $M = (Q, A, \delta, q_0, F)$

- **Descripción Instantánea o Configuración:**
Un elemento de $Q \times A^*$: (q, u) .
- **Configuración Inicial** para $u \in A^*$: (q_0, u)
- **Relación paso de cálculo** entre dos configuraciones:

$$((q, au) \vdash (p, u)) \Leftrightarrow p \in \delta(q, a)$$

De una configuración se puede pasar a varias configuraciones distintas en un paso de cálculo, e incluso a ninguna.

- **Relación de cálculo** entre dos configuraciones:

$((q, u) \stackrel{*}{\vdash} (p, v))$ si y solo si existe una sucesión de configuraciones C_0, \dots, C_n tales que $C_0 = (q, u)$, $C_n = (p, v)$ y $\forall i \leq n-1, C_i \vdash C_{i+1}$.

Lenguaje Aceptado por un AF no-determinista

$$L(M) = \{u \in A^* : \exists q \in F, (q_0, u) \stackrel{*}{\vdash} (q, \varepsilon)\}$$

Las palabras de $L(M)$ se dicen **aceptadas** por el autómata.

Definiciones

Dado un autómata $M = (Q, A, \delta, q_0, F)$, definimos la función δ^* de la siguiente forma:

- Si $B \subseteq Q$,

$$\delta^*(B, a) = \bigcup_{q \in B} \delta(q, a)$$

- Si $B \subseteq Q$,
 - $\delta^*(B, \epsilon) = B$
 - $\delta^*(B, au) = \delta^*(\delta^*(B, a), u)$
- $\delta^*(q, u) = \delta^*({q}, u)$

$\delta^*(B, u)$ es igual a todos los estados a los que se puede llegar desde cualquiera de los estados de B después de leer la palabra u .

Propiedad

Es inmediato probar que el lenguaje aceptado por un autómata finito no-determinista se puede expresar como:

$$L(M) = \{u \in A^* : \delta^*(q_0, u) \cap F \neq \emptyset\}$$

Equivalencia Aut. Deterministas \leftrightarrow No-Deterministas

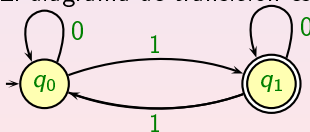
Equivalencia

Un lenguaje L puede ser aceptado por un autómata finito determinista si y solo si L puede ser aceptado por un autómata finito no determinista.

Propiedad 1: Aut. Deterministas \rightarrow No-Deterministas

Todo lenguaje L aceptado por un autómata determinista es aceptado también por un autómata no-determinista

Esto se comprueba considerando que todos los autómatas deterministas son también autómatas no-deterministas, en los que $\delta(q, a)$ tiene siempre un y sólo un estado. Ambos autómatas tienen el mismo cálculo asociado y aceptan el mismo lenguaje. El diagrama de transición es el mismo:



Así, si L es aceptado por un autómata determinista es aceptado también por un autómata no-determinista: aquel que tiene el mismo diagrama.

Aut. No Determinista \rightarrow Aut. Determinista

Propiedad 2: Aut. No-Deterministas \rightarrow Deterministas

Todo lenguaje L aceptado por un autómata no determinista es aceptado también por un autómata determinista

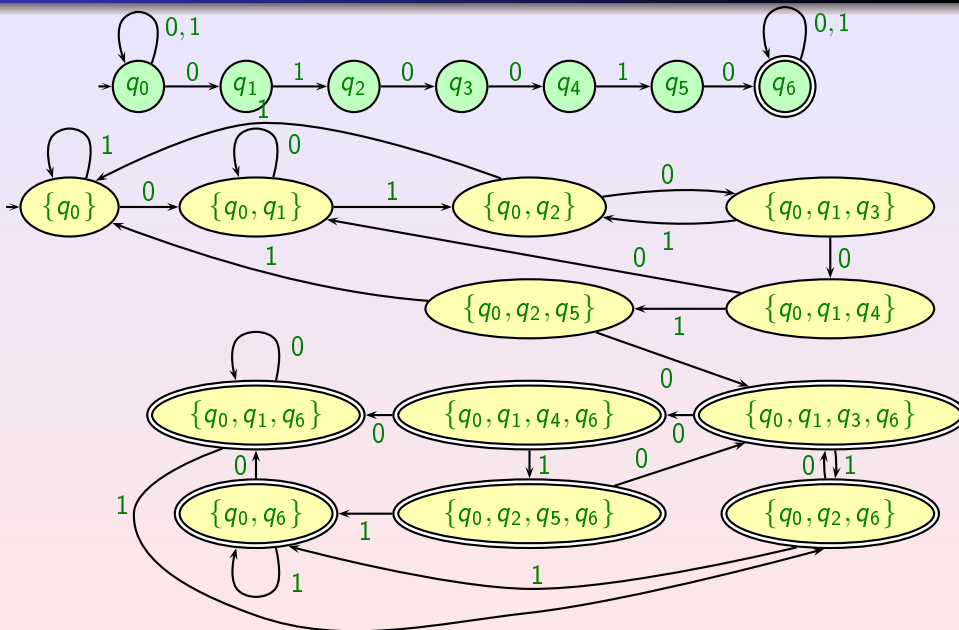
Dado un AFND $M = (Q, A, \delta, q_0, F)$ se llama **autómata determinista asociado** a M , al autómata $\bar{M} = (\bar{Q}, A, \bar{\delta}, \bar{q}_0, \bar{F})$ dado por

- $\bar{Q} = \wp(Q)$
- $\bar{q}_0 = \{q_0\}$
- $\bar{\delta}(B, a) = \delta^*(B, a) = \bigcup_{q \in B} \delta(q, a)$
- $\bar{F} = \{B \in \wp(Q) \mid B \cap F \neq \emptyset\}$

Idea básica: Dado un autómata no determinista se le hace corresponder uno determinista que recorre todos los caminos al mismo tiempo.

Un autómata no-determinista y su determinista asociado aceptan el mismo lenguaje

Ejemplo



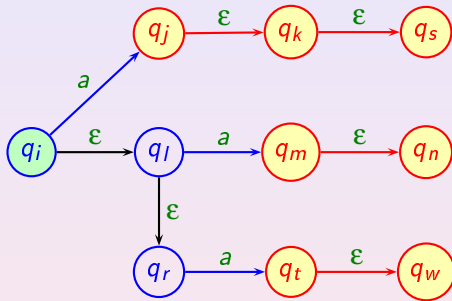
AF No Deterministas con Transiciones Nulas

Un **autómata finito no determinista con transiciones nulas** es una quintupla $M = (Q, A, \delta, q_0, F)$ en la que

- Q es un conjunto finito llamado **conjunto de estados**
- A es un alfabeto llamado **alfabeto de entrada**
- δ es una aplicación llamada **función de transición**

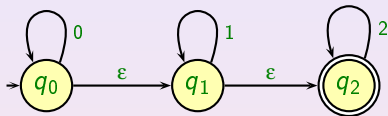
$$\delta : Q \times (A \cup \{\epsilon\}) \rightarrow \wp(Q)$$

- q_0 es un elemento de Q , llamado **estado inicial**
- F es un subconjunto de Q , llamado conjunto de **estados finales**.



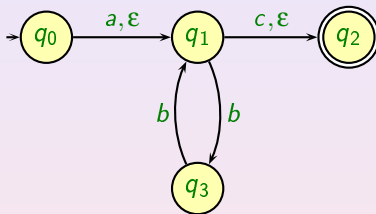
Desde el estado q_i se puede llegar a los estados:
 $q_j, q_k, q_s, q_m, q_n, q_t, q_w$ después de leer una a .

Ejemplo



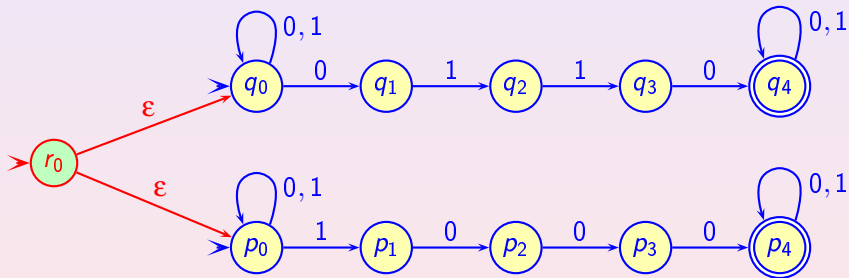
El lenguaje aceptado es $L = \{0^i 1^j 2^k : i, j, k \geq 0\}$.

Ejemplo



El lenguaje generado es $L = \{a^i b^{2j} c^k : i, k = 0, 1 \text{ y } j \geq 0\}$.

Conjunto de palabras que tienen a 0110 o a 1000 como subcadena.



Para un **Autómata Finito con Transiciones Nulas**

$$M = (Q, A, \delta, q_0, F)$$

- **Descripción Instantánea o Configuración:**
Un elemento de $Q \times A^*$: (q, u) .
- **Configuración Inicial** para $u \in A^*$: (q_0, u)
- **Relación paso de cálculo** entre dos configuraciones:
 $((q, u) \vdash (p, v))$ si y solo si se da una de las condiciones
 - $((u = av) \wedge p \in \delta(q, a))$ (caso: $((q, av) \vdash (p, v))$)
 - $((u = v) \wedge p \in \delta(q, \epsilon))$ (caso: $((q, v) \vdash (p, v))$)

De una configuración se puede pasar a varias configuraciones distintas en un paso de cálculo, e incluso a ninguna.

- **Relación de cálculo** entre dos configuraciones:

$((q, u) \stackrel{*}{\vdash} (p, v))$ si y solo si existe una sucesión de configuraciones C_0, \dots, C_n tales que $C_0 = (q, u)$, $C_n = (p, v)$ y $\forall i \leq n-1, C_i \vdash C_{i+1}$.

Lenguaje Aceptado por un ϵ -AF no-determinista

$$L(M) = \{u \in A^* : \exists q \in F, (q_0, u) \stackrel{*}{\vdash} (q, \epsilon)\}$$

Las palabras de $L(M)$ se dicen **aceptadas** por el autómata.

Propiedad

L es aceptado por un autómata finito determinista si y solo si es aceptado por un autómata no-determinista con transiciones nulas.

Propiedad 1: Autómata determinista \rightarrow Autómata No-Determinista con transiciones nulas

Dado un autómata finito determinista M existe un autómata no determinista con transiciones nulas \overline{M} que acepta el mismo lenguaje: $L(M) = L(\overline{M})$

Es inmediato: sería un autómata en el que para cada símbolo del alfabeto de entrada hay siempre una opción y para cada estado $\delta(q, \epsilon) = \emptyset$.

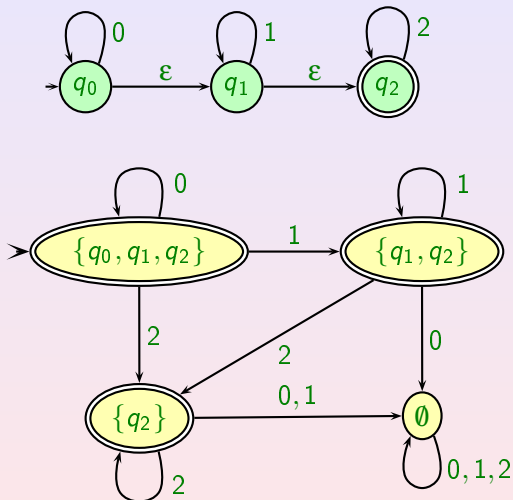
Propiedad 2: Autómata determinista con transiciones nulas \rightarrow Autómata Determinista

Dado un autómata finito no determinista con transiciones nulas M existe un autómata finito determinista \overline{M} que acepta el mismo lenguaje: $L(M) = L(\overline{M})$

La idea intuitiva de la demostración es construir \overline{M} como en el caso de la construcción de un autómata determinista a partir de uno no-determinista: los estados de \overline{M} son subconjuntos del conjunto Q de estados de M y un estado $P \subset Q$ de \overline{M} representa todos los estados en los que puede estar el autómata no determinista con transiciones nulas M .

Primero, veremos un ejemplo.

Ejemplo



Definición: Clausura

Dado un autómata no determinista con transiciones nulas

$M = (Q, A, \delta, q_0, F)$, definimos:

Clausura de un estado q :

$$Cl(q) = \{p : \exists p_1, \dots, p_n, p_1 = q, p_n = p, \quad p_i \in \delta(p_{i-1}, \epsilon) \ (i = 2, \dots, n)\}$$

Clausura de un conjunto de estados P : $Cl(P) = \bigcup_{q \in P} Cl(q)$

Si $B \subseteq Q$, se define la función δ^* :

$$\delta^*(B, a) = \bigcup_{q \in B} \delta(q, a)$$

Construcción formal de un autómata determinista

Dado un autómata no determinista con transiciones nulas

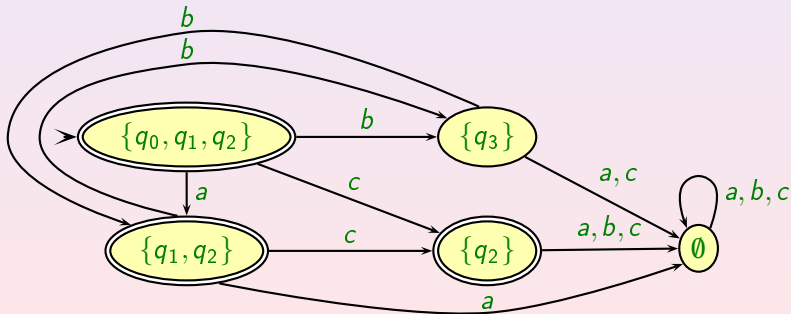
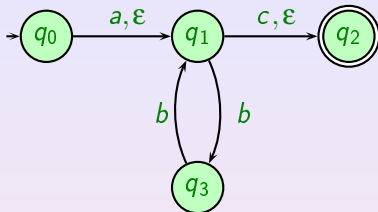
$M = (Q, A, \delta, q_0, F)$, el **Autómata Finito Determinista**

$\overline{M} = (\overline{Q}, A, \overline{\delta}, \overline{q_0}, \overline{F})$ dado por

- $\overline{Q} = \wp(Q)$
- $\overline{\delta}(P, a) = Cl(\delta^*(P, a)) = Cl(\bigcup_{q \in P} \delta(q, a))$
- $\overline{q_0} = Cl(q_0)$
- $\overline{F} = \{P : P \cap F \neq \emptyset\}$

acepta el mismo lenguaje que M .

Ejemplo



EXPRESIONES REGULARES

Si A es un alfabeto, una **expresión regular** sobre este alfabeto se define de la siguiente forma:

- \emptyset es una expresión regular que denota el lenguaje vacío.
- ϵ es una expresión regular que denota el lenguaje $\{\epsilon\}$.
- Si $a \in A$, a es una expresión regular que denota el lenguaje $\{a\}$
- Si r y s son expresiones regulares denotando los lenguajes R y S entonces definimos las siguientes operaciones:
 - **Unión:** $(r + s)$ es una expresión regular que denota el lenguaje $R \cup S$.
 - **Concatenación:** (rs) es una expresión regular que denota el lenguaje RS .
 - **Clausura:** r^* es una expresión regular que denota el lenguaje R^* .

Ejemplos

$$A = \{0, 1\}$$

- **00** *El conjunto {00}*
- **$01^* + 0$** *Conjunto de palabras que empiezan por 0 y después tienen una sucesión de unos.*
- **$(1 + 10)^*$** *Conjunto de palabras en las que los ceros están precedidos siempre por unos*
- **$(0 + 1)^* 011$** *Conjunto de palabras que terminan en 011*
- **$0^* 1^*$** *Conjunto de palabras formadas por una sucesión de ceros seguida de una sucesión de unos. Ambas sucesiones pueden ser vacías*
- **$00^* 11^*$** *Conjunto de palabras formadas por una sucesión de ceros seguida de una sucesión de unos. Ninguna de las sucesiones puede ser vacía*

A $r^* r$ se le denota como r^+ . La última expresión regular quedaría $0^+ 1^+$

Ejemplos - Alfabeto $\{0,1\}$

Construir una expresión regular para las palabras en las que el número de ceros es par.

$$1^*(01^*01^*)^*$$

Construir una expresión regular para las palabras que contengan a 0110 como subcadena.

$$(0+1)^*0110(0+1)^*$$

Construir una expresión regular para el conjunto de palabras que empiezan por 000 y tales que esta subcadena sólo se encuentra al principio de la palabra.

$$(000)(1+10+100)^*$$

Construir una expresión regular para el conjunto de palabras que tienen a 000 o a 101 como subcadena

$$(0+1)^*(000+101)(0+1)^*$$

Propiedades de las Expresiones Regulares

$$\textcircled{1} \quad r_1 + r_2 = r_2 + r_1$$

$\textcircled{2}$

$$r_1 + (r_2 + r_3) = (r_1 + r_2) + r_3$$

$$\textcircled{3} \quad r_1(r_2r_3) = (r_1r_2)r_3$$

$$\textcircled{4} \quad r\varepsilon = r$$

$$\textcircled{5} \quad r\emptyset = \emptyset$$

$$\textcircled{6} \quad r + \emptyset = r$$

$$\textcircled{7} \quad \varepsilon^* = \varepsilon$$

$$\textcircled{8} \quad r_1(r_2 + r_3) = r_1r_2 + r_1r_3$$

$$\textcircled{9} \quad (r_1 + r_2)r_3 = r_1r_3 + r_2r_3$$

$$\textcircled{10} \quad r^+ + \varepsilon = r^*$$

$$\textcircled{11} \quad r^* + \varepsilon = r^*$$

$$\textcircled{12} \quad (r + \varepsilon)^* = r^*$$

$$\textcircled{13} \quad (r + \varepsilon)^+ = r^*$$

$$\textcircled{14} \quad (r_1^* + r_2^*)^* = (r_1 + r_2)^*$$

Preguntas sobre Expresiones Regulares

¿Verdadero o falso?

- 1 Si r y s son expresiones regulares, tenemos que siempre se verifica que $(rs)^* = r^*s^*$
- 2 Si r y s son expresiones regulares, tenemos que siempre se verifica que $(r + s)^* = r^* + s^*$
- 3 Si r_1 y r_2 son expresiones regulares, tales que su lenguaje asociado contiene la palabra vacía, entonces $(r_1r_2)^* = (r_2r_1)^*$.
- 4 Si r y s son expresiones regulares, tenemos que siempre se verifica que $(r + \epsilon)^+ = r^*$
- 5 Si r y s son expresiones regulares, tenemos que siempre se verifica que $r(r + s)^* = (r + s)^*r$
- 6 Si r_1 y r_2 son expresiones regulares, entonces $r_1^*r_2^* \subseteq (r_1r_2)^*$, en el sentido de que los lenguajes asociados están incluidos.
- 7 Si r_1 , r_2 y r_3 son expresiones regulares, entonces $(r_1 + r_2)^*r_3 = r_1^*r_3 + r_2^*r_3$.
- 8 Si r_1 y r_2 son expresiones regulares entonces: $(r_1^*r_2^*)^* = (r_1 + r_2)^*$

Preguntas sobre Expresiones Regulares

¿Verdadero o falso?

- 1 Si r_1 y r_2 son expresiones regulares, entonces $(r_1.r_2)^* = (r_1 + r_2)^*$.
- 2 Si r es una expresión regular, entonces $r^*r^* = r^*$.
- 3 Si r es una expresión regular, entonces $r\emptyset = r + \emptyset$.
- 4 Si r es una expresión regular, entonces se verifica que $r^*\epsilon = r^+\epsilon$.
- 5 Si r_1 y r_2 son expresiones regulares, entonces siempre $r_1(r_2r_1)^* = (r_1r_2)^*r_1$.
- 6 Si r y s son expresiones regulares, entonces $(r^*s^*)^* = (r + s)^*$.
- 7 Si r es una expresión regular, entonces $(rr)^* \subseteq r^*$.
- 8 Si r_1 y r_2 son expresiones regulares, tales que su lenguaje asociado contiene la palabra vacía, entonces $(r_1r_2)^* = (r_1 + r_2)^*$.
- 9 Si r_1, r_2, r_3 son expresiones regulares, entonces $r_1(r_2^* + r_3^*) = r_1r_2^* + r_1r_3^*$.

Equivalencia Autómatas - Expresiones Regulares

- La familia de los lenguajes aceptados por los autómatas finitos coincide con la familia de lenguajes que pueden representarse mediante expresiones regulares. Es decir se verifica la siguiente propiedad

Propiedad

Un lenguaje es aceptado por un autómata finito determinista si y solo si puede representarse mediante una expresión regular.

- Esto se demostrará comprobando:
 - Dada una expresión regular, existe un autómata que acepta el mismo lenguaje que el representado por la expresión regular.
 - Dado un autómata finito existe siempre una expresión regular que representa el lenguaje aceptado por el autómata.
- La primera transformación es más útil, ya que inicialmente los lenguajes se representan mediante expresiones regulares y después necesitamos algoritmos (autómatas) que reconozcan estos lenguajes.

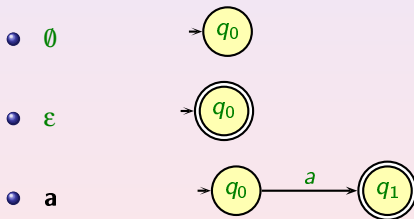
Expresión Regular \rightarrow Autómata

Dada una expresión regular existe un autómata finito que acepta el lenguaje asociado a esta expresión regular.

Vamos a demostrar que existe un AFND con transiciones nulas. A partir de él se podría construir el autómata determinista asociado.

La construcción del autómata va a ser recursiva.

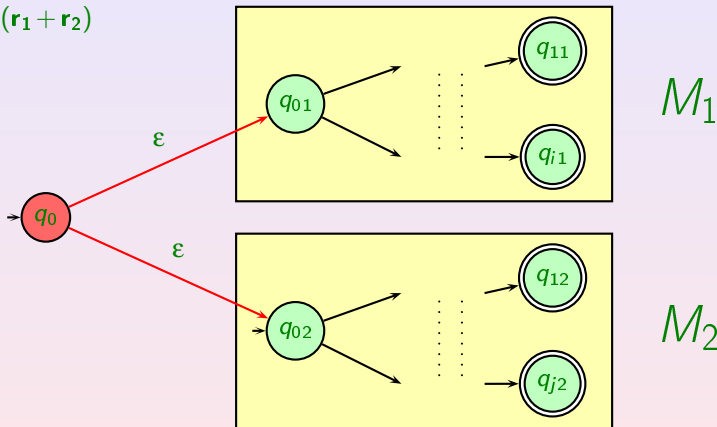
Para las expresiones regulares iniciales tenemos los siguiente autómatas:



Autómatas Compuestos: Unión

Si M_1 es el autómata que acepta el mismo lenguaje que el representado por r_1 y M_2 el que acepta el mismo lenguaje que el de r_2 , entonces

- **Unión** ($r_1 + r_2$)



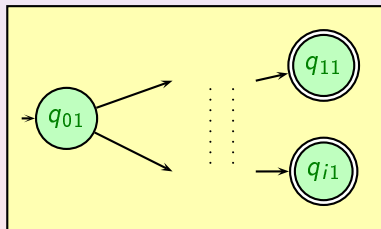
Unión: Expresión Matemática

En lenguaje matemático, la unión se puede expresar de la siguiente forma. Si $M_1 = (Q_1, A, \delta_1, q_0^1, F_1)$ y $M_2 = (Q_2, A, \delta_2, q_0^2, F_2)$ con $Q_1 \cap Q_2 = \emptyset$, entonces el autómata que acepta la **unión** es $M = (Q, A, \delta, q_0, F)$ donde

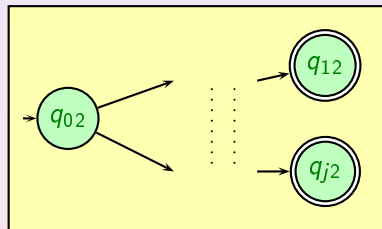
- $Q = Q_1 \cup Q_2 \cup \{q_0\}$ donde $q_0 \notin (Q_1 \cup Q_2)$ es un nuevo estado.
- δ viene definida por
 - $\delta(q, a) = \delta_1(q, a)$ si $q \in Q_1, a \in A$
 - $\delta(q, \epsilon) = \delta_1(q, \epsilon)$ si $q \in Q_1$
 - $\delta(q, a) = \delta_2(q, a)$ si $q \in Q_2, a \in A$
 - $\delta(q, \epsilon) = \delta_2(q, \epsilon)$ si $q \in Q_2$
 - $\delta(q_0, a) = \emptyset$ si $a \in A$
 - $\delta(q_0, \epsilon) = \{q_0^1, q_0^2\}$
- $F = F_1 \cup F_2$

Autómatas Compuestos: Concatenación

- **Concatenación:** El autómata para la expresión $(r_1 r_2)$ es



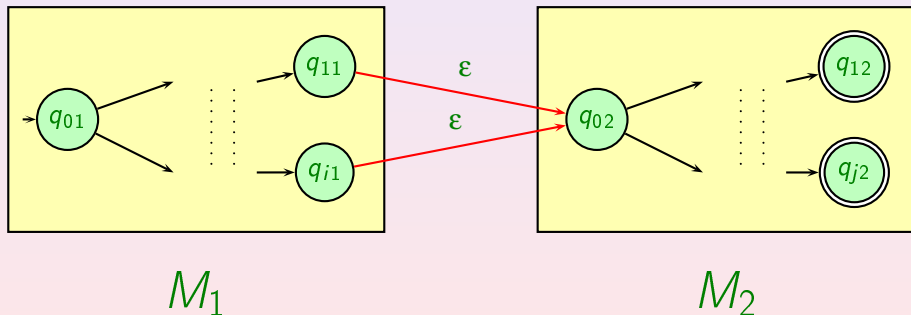
M_1



M_2

Autómatas Compuestos: Concatenación

- Concatenación:** El autómata para la expresión $(r_1 r_2)$ es



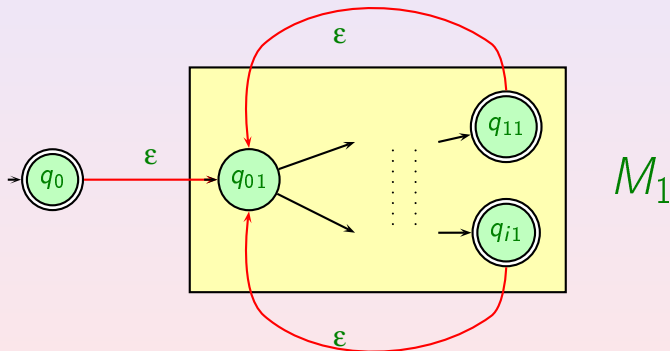
Concatenación: Expresión Matemática

En lenguaje matemático, la concatenación se puede expresar de la siguiente forma. Si $M_1 = (Q_1, A, \delta_1, q_0^1, F_1)$ y $M_2 = (Q_2, A, \delta_2, q_0^2, F_2)$ con $Q_1 \cap Q_2 = \emptyset$, entonces el autómata que acepta la concatenación es $M = (Q, A, \delta, q_0, F)$ donde:

- $Q = Q_1 \cup Q_2$.
- δ viene definida por
 - $\delta(q, a) = \delta_1(q, a)$ si $q \in Q_1 \setminus F_1, a \in A$
 - $\delta(q, \epsilon) = \delta_1(q, \epsilon)$ si $q \in Q_1 \setminus F_1$
 - $\delta(q, a) = \delta_1(q, a)$ si $q \in F_1, a \in A$
 - $\delta(q, \epsilon) = \delta_1(q, \epsilon) \cup \{q_0^2\}$ si $q \in F_1$
 - $\delta(q, a) = \delta_2(q, a)$ si $q \in Q_2$
 - $\delta(q, \epsilon) = \delta_2(q, \epsilon)$ si $q \in Q_2$
- $q_0 = q_0^1$
- $F = F_2$

Autómatas Compuestos: Clausura

- **Clausura:** El autómata para r_1^* es



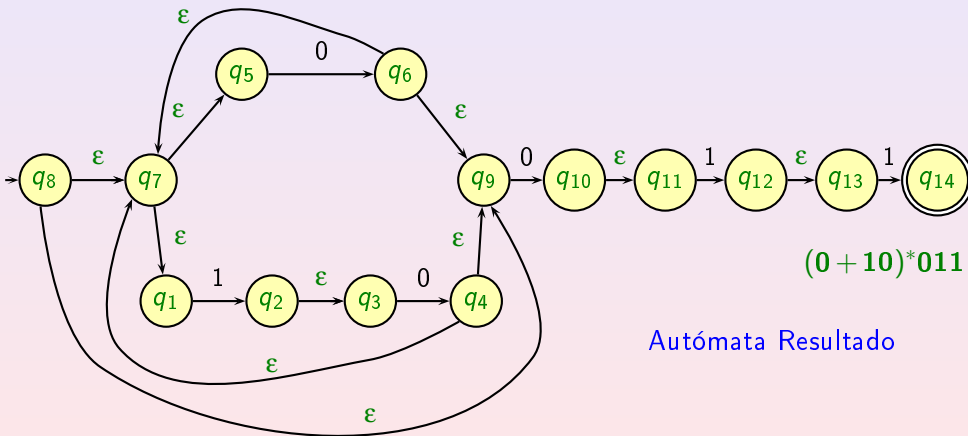
Clausura: Expresión Matemática

En lenguaje matemático: si $M_1 = (Q_1, A, \delta_1, q_0^1, F_1)$, entonces el autómata que acepta la **clausura** es $M = (Q, A, \delta, q_0, F)$ donde:

- $Q = Q_1 \cup \{q_0\}$, donde $q_0 \notin Q_1$.
- δ viene definida por
 - $\delta(q, a) = \delta_1(q, a)$ si $q \in Q_1 \setminus F_1, a \in A$
 - $\delta(q, \varepsilon) = \delta_1(q, \varepsilon)$ si $q \in Q_1 \setminus F_1$
 - $\delta(q, a) = \delta_1(q, a)$ si $q \in F_1, a \in A$
 - $\delta(q, \varepsilon) = \delta_1(q, \varepsilon) \cup \{q_0\}$ si $q \in F_1$
 - $\delta(q_0, a) = \emptyset$ si $a \in A$
 - $\delta(q_0, \varepsilon) = \{q_0^1\}$
- q_0
- $F = F_1 \cup \{q_0\}$

Ejemplo

Encontrar un autómata que acepte el mismo lenguaje que el asociado a la expresión regular $(0 + 10)^*011$



$(0 + 10)^*011$

Autómata Resultado

Si L es aceptado por un autómata finito determinista, entonces puede venir expresado mediante una expresión regular.

Sea el autómata $M = (Q, A, \delta, q_1, F)$, $Q = \{q_1, \dots, q_n\}$ y q_1 es el estado inicial.

Sea R_{ij}^k el conjunto de las cadenas de A^* que permiten pasar del estado q_i al estado q_j y no pasa por ningún estado intermedio de numeración mayor que k (q_i y q_j si pueden tener numeración mayor que k).

De forma más específica $u = a_1 \dots a_m$ está en R_{ij}^k si y solo si todo estado $q_{i_l} = \delta^*(q_i, a_1 \dots a_l)$ donde $1 \leq l \leq m-1$ es tal que $i_l \leq k$.

Cálculo de R_{ij}^k

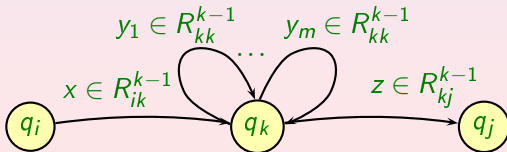
R_{ij}^k se puede calcular de forma recursiva:

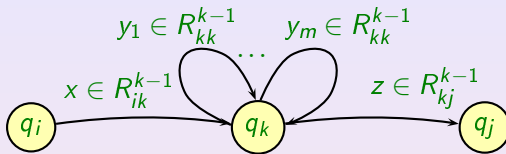
$$R_{ij}^0 = \begin{cases} \{a : \delta(q_i, a) = q_j\} & \text{si } i \neq j \\ \{a : \delta(q_i, a) = q_i\} \cup \{\epsilon\} & \text{si } i = j \end{cases}$$

Para $k \geq 1$, tenemos que R_{ij}^k está compuesto de dos tipos de palabras:

- Palabras que para ir de q_i a q_j no pasan por q_k : pertenecen a R_{ij}^{k-1}
- Palabras que para ir de q_i a q_j pasan por q_k .

Una palabra de este lenguaje está compuesta de tres partes:





Como la palabra $y_1 \dots y_m \in \left(R_{kk}^{k-1}\right)^*$, entonces la palabra completa está en

$$R_{ik}^{k-1} \left(R_{kk}^{k-1}\right)^* R_{kj}^{k-1}$$

Uniendo las dos partes, obtenemos:

$$R_{ij}^k = R_{ij}^{k-1} \cup R_{ik}^{k-1} \left(R_{kk}^{k-1}\right)^* R_{kj}^{k-1}$$

Asociando expresión regular r_{ij}^k a R_{ij}^k

Expresión regular asociada a $R_{ij}^k \longrightarrow r_{ij}^k$

Para $k = 0$ es inmediato.

$$r_{ij}^0 = \begin{cases} a_1 + \dots + a_l & \text{si } i \neq j \\ a_1 + \dots + a_l + \varepsilon & \text{si } i = j \end{cases}$$

donde $\{a_1, \dots, a_l\}$ es el conjunto $\{a : \delta(q_i, a) = q_j\}$.

Si este conjunto es vacío la expresión regular sería:

$$r_{ij}^0 = \begin{cases} \emptyset & \text{si } i \neq j \\ \varepsilon & \text{si } i = j \end{cases}$$

Cálculo de las expresiones r_{ij}^k , calculadas las r_{ij}^{k-1}

$$R_{ij}^k = R_{ij}^{k-1} \cup R_{ik}^{k-1} (R_{kk}^{k-1})^* R_{kj}^{k-1} \longrightarrow r_{ij}^k = r_{ij}^{k-1} + r_{ik}^{k-1} (r_{kk}^{k-1})^* r_{kj}^{k-1}$$

Expresión Regular del lenguaje aceptado por el autómata

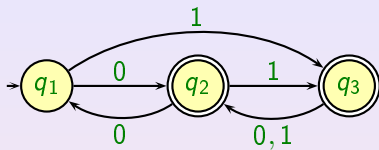
$$L(M) = \bigcup_{q_j \in F} R_{1j}^n$$

Por tanto, $L(M)$ viene denotado por la expresión regular

$$r_{1j_1}^n + \dots + r_{1j_k}^n$$

donde $F = \{q_{j_1}, \dots, q_{j_k}\}$ y q_1 es el estado inicial.

Ejemplo



$$r_{11}^0 = \varepsilon$$

$$r_{12}^0 = 0$$

$$r_{13}^0 = 1$$

$$r_{21}^0 = 0$$

$$r_{22}^0 = \varepsilon$$

$$r_{23}^0 = 1$$

$$r_{31}^0 = \emptyset$$

$$r_{32}^0 = 0 + 1$$

$$r_{33}^0 = \varepsilon$$

Ejemplo

$$\left\{ \begin{array}{l} r_{11}^0 = \varepsilon \\ r_{12}^0 = 0 \\ r_{13}^0 = 1 \\ r_{21}^0 = 0 \\ r_{22}^0 = \varepsilon \\ r_{23}^0 = 1 \\ r_{31}^0 = \emptyset \\ r_{32}^0 = 0 + 1 \\ r_{33}^0 = \varepsilon \end{array} \right.$$

$$\left\{ \begin{array}{l} r_{11}^1 = r_{11}^0 + r_{11}^0 (r_{11}^0)^* r_{11}^0 = \varepsilon + \varepsilon(\varepsilon)^* \varepsilon = \varepsilon \\ r_{12}^1 = r_{12}^0 + r_{11}^0 (r_{11}^0)^* r_{12}^0 = 0 + \varepsilon(\varepsilon)^* 0 = 0 \\ r_{13}^1 = r_{13}^0 + r_{11}^0 (r_{11}^0)^* r_{13}^0 = 1 + \varepsilon(\varepsilon)^* 1 = 1 \\ r_{21}^1 = r_{21}^0 + r_{21}^0 (r_{11}^0)^* r_{11}^0 = 0 + 0(\varepsilon)^* \varepsilon = 0 \\ r_{22}^1 = r_{22}^0 + r_{21}^0 (r_{11}^0)^* r_{12}^0 = \varepsilon + 0(\varepsilon)^* 0 = \varepsilon + 00 \\ r_{23}^1 = r_{23}^0 + r_{21}^0 (r_{11}^0)^* r_{13}^0 = 1 + 0(\varepsilon)^* 1 = 1 + 01 \\ r_{31}^1 = r_{31}^0 + r_{31}^0 (r_{11}^0)^* r_{11}^0 = \emptyset + \emptyset(\varepsilon)^* \varepsilon = \emptyset \\ r_{32}^1 = r_{32}^0 + r_{31}^0 (r_{11}^0)^* r_{12}^0 = 0 + 1 + \emptyset(\varepsilon)^* 0 = 0 + 1 \\ r_{33}^1 = r_{33}^0 + r_{31}^0 (r_{11}^0)^* r_{13}^0 = \varepsilon + \emptyset(\varepsilon)^* 1 = \varepsilon \end{array} \right.$$

Ejemplo

$$\left\{ \begin{array}{l} r_{11}^1 = \varepsilon \\ r_{12}^1 = 0 \\ r_{13}^1 = 1 \\ r_{21}^1 = 0 \\ r_{22}^1 = \varepsilon + 00 \\ r_{23}^1 = 1 + 01 \\ r_{31}^1 = \emptyset \\ r_{32}^1 = 0 + 1 \\ r_{33}^1 = \varepsilon \end{array} \right\} \left\{ \begin{array}{l} r_{11}^2 = r_{11}^1 + r_{12}^1 (r_{22}^1)^* r_{21}^1 = \varepsilon + 0(\varepsilon + 00)^* 0 = (00)^* \\ r_{12}^2 = r_{12}^1 + r_{12}^1 (r_{22}^1)^* r_{22}^1 = 0 + 0(\varepsilon + 00)^* (\varepsilon + 00) = 0(00)^* \\ r_{13}^2 = r_{13}^1 + r_{12}^1 (r_{22}^1)^* r_{23}^1 = 1 + 0(\varepsilon + 00)^* (1 + 01) = 0^* 1 \\ r_{21}^2 = r_{21}^1 + r_{22}^1 (r_{22}^1)^* r_{21}^1 = 0 + (\varepsilon + 00)(\varepsilon + 00)^* 0 = (00)^* 0 \\ r_{22}^2 = r_{22}^1 + r_{22}^1 (r_{22}^1)^* r_{22}^1 = \varepsilon + 00 + (\varepsilon + 00)(\varepsilon + 00)^* (\varepsilon + 00) = (00)^* \\ r_{23}^2 = r_{23}^1 + r_{22}^1 (r_{22}^1)^* r_{23}^1 = 1 + 01 + (\varepsilon + 00)(\varepsilon + 00)^* (1 + 01) = 0^* 1 \\ r_{31}^2 = r_{31}^1 + r_{32}^1 (r_{22}^1)^* r_{21}^1 = \emptyset + (0 + 1)(\varepsilon + 00)^* 0 = (0 + 1)(00)^* 0 \\ r_{32}^2 = r_{32}^1 + r_{32}^1 (r_{22}^1)^* r_{22}^1 = 0 + 1 + (0 + 1)(\varepsilon + 00)^* (\varepsilon + 00) = (0 + 1)(00)^* \\ r_{33}^2 = r_{33}^1 + r_{32}^1 (r_{22}^1)^* r_{23}^1 = \varepsilon + (0 + 1)(\varepsilon + 00)^* (1 + 01) = \varepsilon + (0 + 1)0^* 1 \end{array} \right.$$

Ejemplo

$$\begin{aligned}r_{11}^2 &= (00)^* \\r_{12}^2 &= 0(00)^* \\r_{13}^2 &= 0^*1 \\r_{21}^2 &= (00)^*0 \\r_{22}^2 &= (00)^* \\r_{23}^2 &= 0^*1 \\r_{31}^2 &= (0+1)(00)^*0 \\r_{32}^2 &= (0+1)(00)^* \\r_{33}^2 &= \varepsilon + (0+1)0^*1\end{aligned}$$

Finalmente la expresión regular para el lenguaje aceptado es:

$$\begin{aligned}r_{12}^3 + r_{13}^3 &= r_{12}^2 + r_{13}^2(r_{33}^2)^*r_{32}^2 + r_{13}^2 + r_{13}^2(r_{33}^2)^*r_{33}^2 = \\0(00)^* + 0^*1(\varepsilon + (0+1)0^*1)^*(0+1)(00)^* + 0^*1 + 0^*1(\varepsilon + (0+1)0^*1)^*(\varepsilon + (0+1)0^*1) &= \\0(00)^* + 0^*1((0+1)0^*1)^*(0+1)(00)^* + 0^*1 + 0^*1((0+1)0^*1)^* &\end{aligned}$$

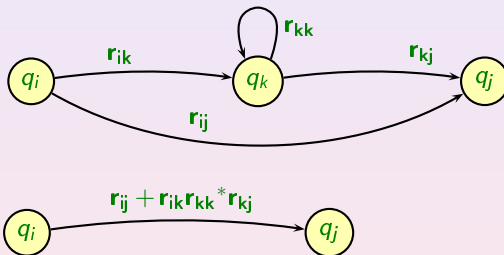
Método alternativo: Eliminar estados

- Para cada estado final $q \in F$ construir un autómata finito equivalente que tenga sólo el estado inicial y ese final y con transiciones que vienen etiquetadas por expresiones regulares.
- A cada uno de esos autómatas se le asocia de forma sencilla una expresión regular.
- Si r_q es la expresión regular del autómata que corresponde al estado final $q \in F$, entonces la expresión regular del autómata $\sum_{q \in F} r_q$ (la suma de todas las expresiones regulares que se obtienen para los distintos estados finales).
- Hay que tener en cuenta que si el estado inicial es final, habrá que construir un autómata equivalente con un solo estado y r_q será la etiqueta de la transición que vaya de ese estado a él mismo.
- Para construir un autómata equivalente con expresiones regulares en las transiciones y con sólo estado inicial y final se procede con un proceso de eliminación de estados intermedios que comienza con el autómata de partida.

Eliminación de estado intermedio

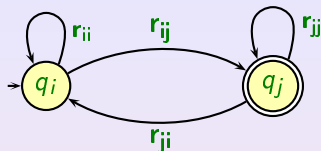
Para eliminar todo estado intermedio q_k , la etiqueta r_{ij} de cada transición de q_i a q_j se actualiza con

$$r_{ij} \leftarrow r_{ij} + r_{ik} r_{kk}^* r_{kj}$$

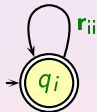


Hay que recalcular todas las todas las transiciones posibles del nuevo autómata. Si una transición no existe, entonces se supone que es vacía \emptyset y hay que tener en cuenta que $\emptyset^* = \epsilon$

Asociar expresión regular a autómata simplificado

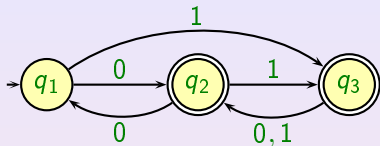


Se le asocia: $(r_{ii} + r_{ij}r_{jj}^*r_{ji})^*r_{ij}r_{jj}^*$

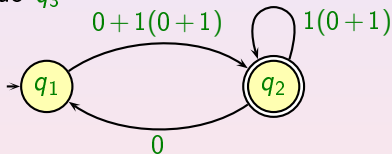


Se le asocia: r_{ii}^*

Ejemplo



Obtenemos la expresión regular r_2 para el estado final q_2 eliminando q_3

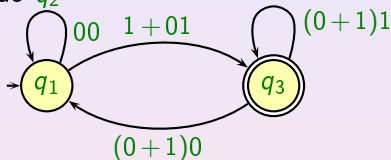


Expresión regular:

$$r_2 = (0 + 1(0 + 1))(1(0 + 1))^*0^*(0 + 1(0 + 1))(1(0 + 1))^*$$

Ejemplo

Obtenemos la expresión regular r_3 para el estado final q_3 eliminando q_2



Expresión regular:

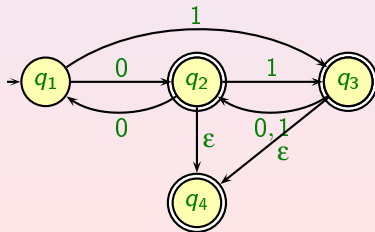
$$r_3 = (00 + (1 + 01)((0 + 1)1)^*(0 + 1)0)^*(1 + 01)((0 + 1)1)^*$$

Expresión regular final

$$r_2 + r_3 = (0 + 1(0 + 1))(1(0 + 1))^*0)^*(0 + 1(0 + 1))(1(0 + 1))^* + (00 + (1 + 01)((0 + 1)1)^*(0 + 1)0)^*(1 + 01)((0 + 1)1)^*$$

El método de eliminación con autómatas no deterministas

- El método de eliminación que hemos visto vale con autómatas no deterministas y con transiciones nulas. No hace falta pasar el autómata a determinista.
- En vez de hacer un proceso de eliminación para cada estado final, se puede diseñar uno equivalente con un solo estado final y así aplicar la eliminación de estados intermedios solo para ese estado:
 - Se añade un estado nuevo estado que es el único final
 - Los antiguos estados finales dejan de serlo y se unen con transiciones nulas con el nuevo estado final.



Expresiones Regulares en Unix - Estándar IEEE POSIX

Expresión Regular

Caracteres Normales

+, *

|

[...]

[a - b]

[^...]

?

{*nombre*}

{*n*}

{*n, m*}

*

"..."

^, \$

.

\n

Significado

Ellos mismos

Superíndices +, *

La unión de los lenguajes

Cualquier símbolo entre corchetes

todos los caracteres entre *a* y *b*

El complementario de [...]

0 ó 1 repetición de lo anterior

Se substituye la e.r. *nombre*

n repeticiones de la anterior e.r.

entre *n* y *m* repeticiones de la anterior e.r.

El carácter *

Los caracteres entre comillas literalmente

Principio, fin de línea

Cualquier carácter excepto el salto de línea

salto de línea

Estructura de un fichero lex

```
nombre1      er1
nombre2      er2
nombrei      eri
              decla global1
              decla global2

%%

              decla local1
              decla local2

er1           accion1;
er2           accion2;
er3           accion3;
%%
definiciones de funciones en C
```

Variables y Procedimientos

- `yylex()` – Programa que reconoce las expresiones regulares y ejecuta las acciones.
- `main()` – Programa principal. Por defecto sólo llama a `yylex()`. Se puede redefinir después de los últimos%%
- `yywrap()` – Función que se ejecuta cuando `yylex()` encuentra un fin de fichero. Si devuelve 1 (lo único que hace la versión por defecto) `yylex()` termina. Si devuelve un 0, `yylex()` sigue leyendo de la entrada.
- `yyin` – Fichero de entrada (`stdin` por defecto)
- `yyout` – Fichero de salida (`stdout` por defecto)
- `yytext` – Variable que contiene la cadena reconocida por `yylex()`
- `yylen` – Longitud de la cadena reconocida

Ejemplo

```
car          [a-zA-Z]
digito       [0-9]
signo        (\-|\+ )
suc          ({digito}+)
enter        ({signo}?{suc})
real1        ({enter}\.{digito}*)
real2        ({signo}?\.{suc})

int          ent=0, real=0, ident=0, sumaent=0;

%%
int          i;
{enter}      {ent++; sscanf(yytext,"%d",&i); sumaent += i;
              printf("Numero entero%s\n",yytext);}
({real1}|{real2}) {real++; printf("Num. real%s\n",yytext);}
{car}({car}|{digito})* {ident++; printf("Identificador%s\n",yytext);}
.\|n        {;}
%%
yywrap()
{printf("Numero de Enteros%d, reales%d, ident%d,
        Suma de Enteros%d",ent,real,ident,sumaent); return 1;}
```

- 1 Crear fichero ejemplo con el contenido anterior
- 2 Ejecutar `lex` con el fichero creado:
`lex ejemplo`
- 3 Compilar el programa que crea `lex`:
`gcc lex.yy.c -o prog -ll`
- 4 ejecutar el programa `prog`

Aplicaciones de las Expresiones Regulares

- Para búsqueda de patrones (buscar direcciones, enlaces o números de teléfono en páginas web).
- Fueron centrales en el desarrollo de Unix:
K. Thompson (1968) Regular expressions search algorithms. *Comm. ACM.* 11, 419–422.

Existen instrucciones como **grep**: '*Global (Search for) Regular Expressions and Print*'

- *Common Applications of Regular Expressions* By Richard Lowe
en
<http://www.4guysfromrolla.com/webtech/120400-1.shtml>
contiene 4 aplicaciones de expresiones regulares, desde verificación de direcciones de correo electrónico a dividir un documento en secciones para incorporarlo en una base de datos.
- En <http://www.webreference.com/js/column5/> podeis ver el uso de expresiones regulares en navegadores.

- *Lineales por la derecha.-* Cuando todas las producciones tienen la forma

$$A \rightarrow uB$$

$$A \rightarrow u$$

- *Lineales por la izquierda.-* Cuando todas las producciones tienen la forma

$$A \rightarrow Bu$$

$$A \rightarrow u$$

Gramática Lineal por la Derecha:

$S \rightarrow 0A$, $A \rightarrow 10A$, $A \rightarrow \epsilon$

Expresión Regular

$0(10)^*$

Gramática Lineal por la Izquierda:

$S \rightarrow S10$, $S \rightarrow 0$

Gramática Regular \rightarrow Autómata

Si L es un lenguaje generado por una gramática regular, entonces existe un autómata finito determinista que lo reconoce.

L es un lenguaje generado por la gramática $G = (V, T, P, S)$ lineal por la derecha. AFND con movimientos nulos que acepta L : $M = (Q, T, \delta, q_0, F)$ donde

- $Q = \{[\alpha] : (\alpha = S) \vee (\exists A \in V, u \in T^*, \text{ tales que } A \rightarrow u\alpha \in P)\}$
- $q_0 = [S]$
- $F = \{[\epsilon]\}$
- δ viene definida por
 - Si A es una variable: $\delta([A], \epsilon) = \{[\alpha] : (A \rightarrow \alpha) \in P\}$
 - Si $a \in T$ y $\alpha \in (T^* V \cup T^*)$, entonces

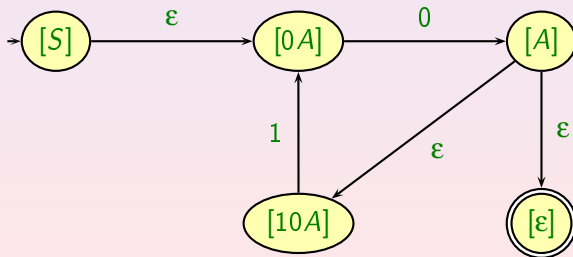
$$\delta([a\alpha], a) = [\alpha]$$

Ejemplo

Sea la gramática:

$$S \rightarrow 0A, \quad A \rightarrow 10A, \quad A \rightarrow \epsilon$$

El autómatá que se obtiene es el siguiente:



Gramáticas Lineales por la Izquierda

Gramática lineal por la izquierda, $G = (V, T, P, S)$

1. Consideraremos la gramática $G' = (V, T, P', S)$ donde

$$P' = \{A \rightarrow \alpha : A \rightarrow \alpha^{-1} \in P\}$$

Es inmediato que $L(G') = L(G)^{-1}$.

2. Sea M' el autómata finito no-determinista que acepta el lenguaje $L(G')$.

3. Calcular M a partir de M' invirtiendo el autómata:

- Dejar sólo un estado final (ocurre siempre en nuestro caso).
- Invertir las transiciones
- Intercambiar el estado inicial y el final.

El lenguaje aceptado por M es: $L(M')^{-1} = L(G')^{-1} = (L(G)^{-1})^{-1} = L(G)$

Ejemplo

$S \rightarrow S10$, $S \rightarrow 0$

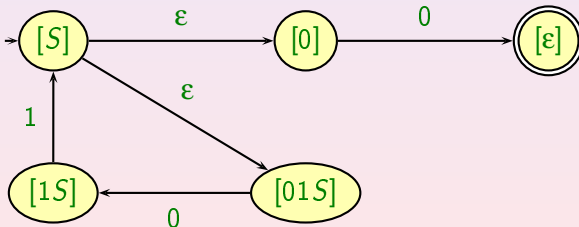
Para construir un AFND con transiciones nulas que acepte este lenguaje se dan los siguientes pasos:

- Invertir la parte derecha de las producciones:

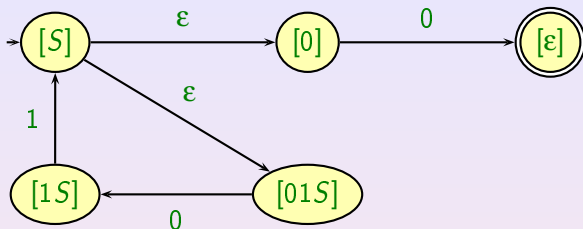
$S \rightarrow 01S$

$S \rightarrow 0$

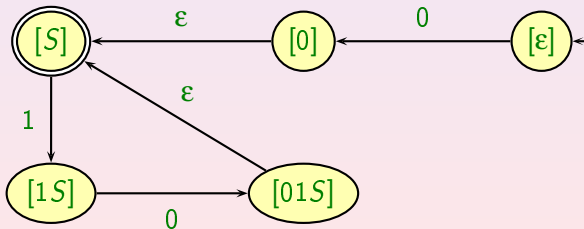
- Construir el AFND con transiciones nulas asociado



Ejemplo Cont.



- Invertimos el autómata



Autómata \rightarrow Gramática lineal

Si L es aceptado por un Autómata Finito Determinístico entonces L puede generarse mediante una gramática lineal por la derecha y por una lineal por la izquierda.

Sea $L = L(M)$ donde $M = (Q, A, \delta, q, F)$ es un autómata finito determinista.

La gramática lineal por la derecha es $G = (Q, A, P, q_0)$ donde las variables son los estados, la variable inicial es q_0 y P contiene las producciones,

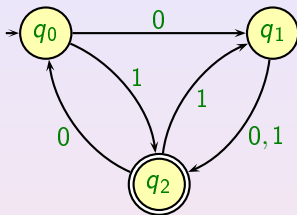
$$p \rightarrow aq, \quad \text{si } \delta(p, a) = q$$

$$p \rightarrow \epsilon, \quad \text{si } p \in F$$

Para el caso de una gramática lineal por la izquierda, invertimos el autómata, construimos la gramática lineal por la derecha asociada e invertimos la parte derecha de las producciones.

Ejemplo: Gramática Lineal por la Derecha

Consideremos el autómata:



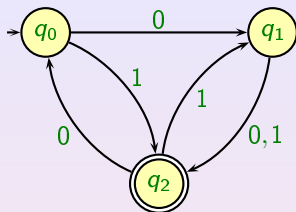
La gramática es (variable inicial q_0):

$$q_0 \rightarrow 0q_1, \quad q_0 \rightarrow 1q_2, \quad q_1 \rightarrow 0q_2, \quad q_1 \rightarrow 1q_2$$

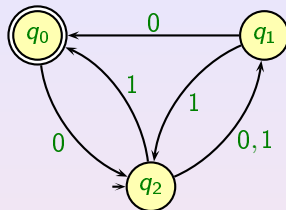
$$q_2 \rightarrow 0q_0, \quad q_2 \rightarrow 1q_1, \quad q_2 \rightarrow \varepsilon$$

Ejemplo: Gramática Lineal por la Izquierda

Autómata de Partida:



Invertimos el autómata:



La gramática asociada a este autómata es (variable inicial q_2):

$$q_1 \rightarrow 0q_0, \quad q_2 \rightarrow 1q_0, \quad q_2 \rightarrow 0q_1, \quad q_2 \rightarrow 1q_1$$

$$q_0 \rightarrow 0q_2, \quad q_1 \rightarrow 1q_2, \quad q_0 \rightarrow \varepsilon$$

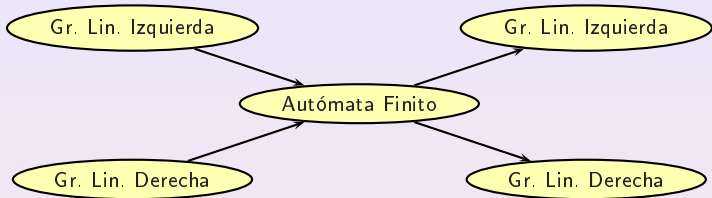
Invertimos la parte derecha de las producciones:

$$q_1 \rightarrow q_00, \quad q_2 \rightarrow q_01, \quad q_2 \rightarrow q_10, \quad q_2 \rightarrow q_11$$

$$q_0 \rightarrow q_20, \quad q_1 \rightarrow q_21, \quad q_0 \rightarrow \varepsilon$$

Equivalencia Gramáticas Lineales por la Derecha y Lineales por la izquierda

Hemos demostrado:



Componiendo dos de estas transformaciones se puede conseguir pasar de una gramática lineal por la izquierda a una lineal por la derecha que genere el mismo lenguaje y, de la misma forma, pasar de una gramática lineal por la derecha a una lineal por la izquierda que genere el mismo lenguaje.

Resultado

Dada una gramática lineal por la izquierda existe una gramática lineal por la derecha que genere el mismo lenguaje.

Dada una gramática lineal por la derecha existe una gramática lineal por la izquierda que genere el mismo lenguaje.