Tema 2: Autómatas Finitos y Expresiones Regulares

Serafín Moral

Universidad de Granada

Octubre, 2020

Contenido

- 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:

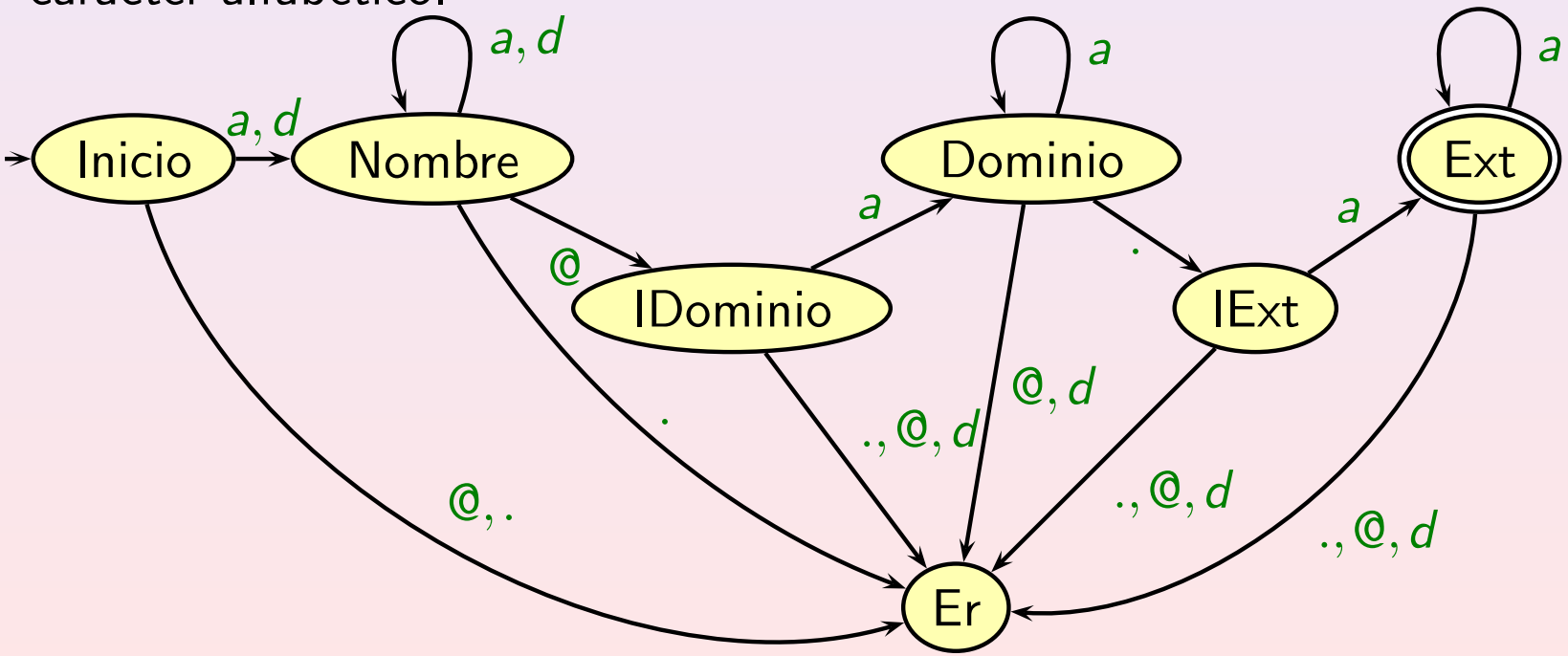
- Software para el diseño y verificación de circuitos digitales.
- Construcción de analizadores léxicos de compiladores.
- Software para analizar grandes conjuntos de textos para buscar palabras, estructuras u otros patrones (p.e. páginas web).
- Software para comprobar la corrección de cualquier tipo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

Ejemplo Introductorio

Supongamos que queramos reconocer palabras que son direcciones de correo electrónico del tipo nombre@dominio.exten, donde nombre es una palabra formada por dígitos y caracteres alfabéticos, y dominio y extensión son palabras formadas por símbolos alfabéticos (la realidad es más compleja).

¿Cómo podemos especificar un algoritmo que identifique las palabras que corresponden a este patrón?

Una idea es usar el siguiente diagrama en el que d es un dígito y a un carácter alfabético.



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

$$\delta : Q \times A \to Q$$

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

Ejemplo

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:

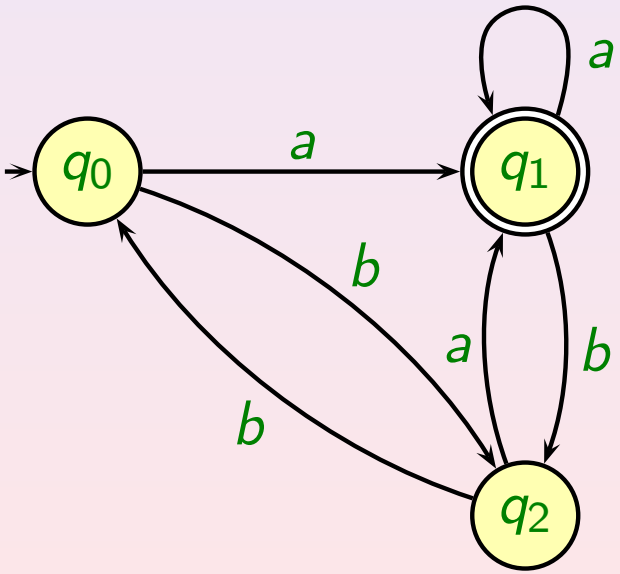
$$\begin{array}{ll} \delta(q_0, a) = q_1, & \delta(q_0, b) = q_2, \\ \delta(q_1, a) = q_1, & \delta(q_1, b) = q_2, \\ \delta(q_2, a) = q_1, & \delta(q_2, b) = q_0 \end{array}$$

- $F = \{q_1\}$

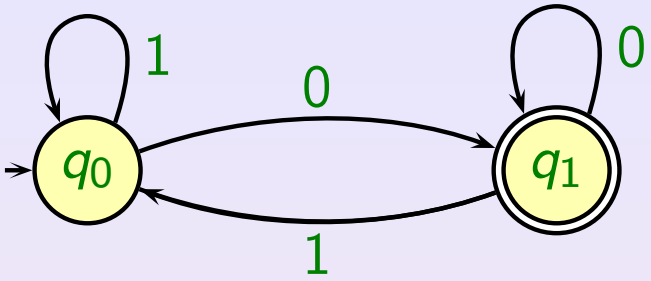
Diagrama de Transición

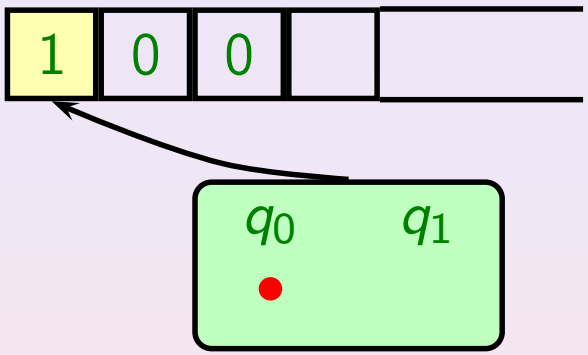
Es un grafo en el que:

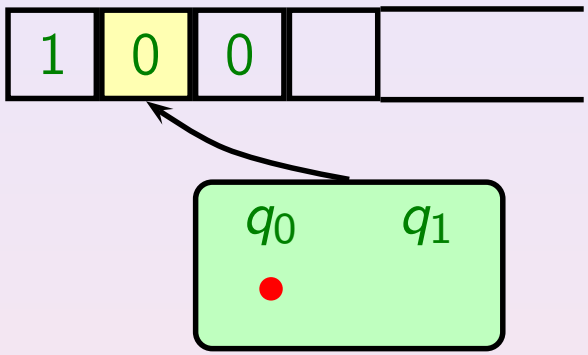
- Hay un nodo por cada estado
- Por cada transición $\delta(q, a) = p$ hay un arco de q a p con la etiqueta a.
- El estado inicial está indicado con un ángulo entrante. Los estados finales están indicados con una doble circunferencia.

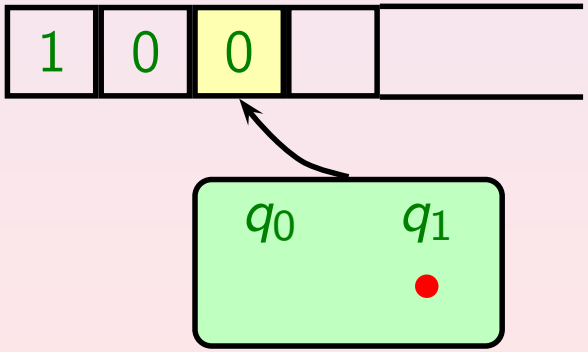


Cálculo Asociado. Traza

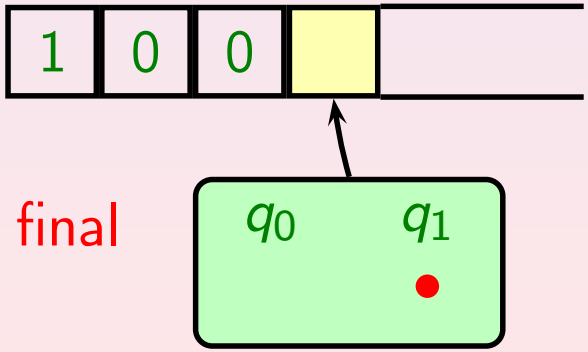








Estado final
SI



PROCESO DE CALCULO

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.

Proceso de Cálculo

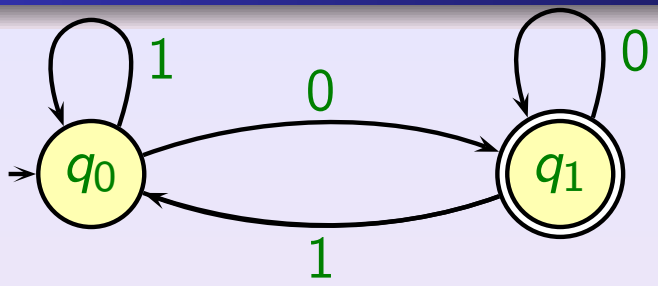
- **Relación de cálculo entre dos configuraciones:**
  $((q,u) \overset{*}{\vdash} (p,v))$ si y solo si existe una sucesión de configuraciones $C_0,\ldots,C_n$ tales que $C_0=(q,u), C_n=(p,v)$ y $\forall i \leq n-1, C_i \vdash C_{i+1}$.

Lenguaje Aceptado por un Autómata Finito

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

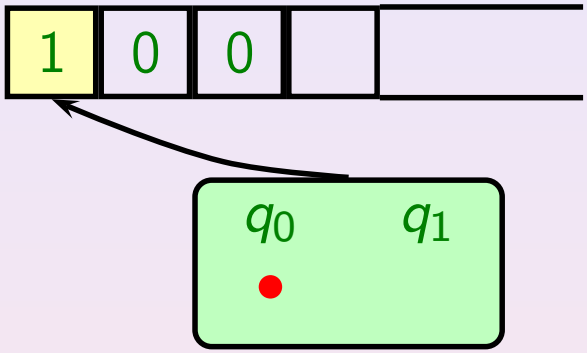
Las palabras de $L(M)$ se dicen aceptadas por el autómata.

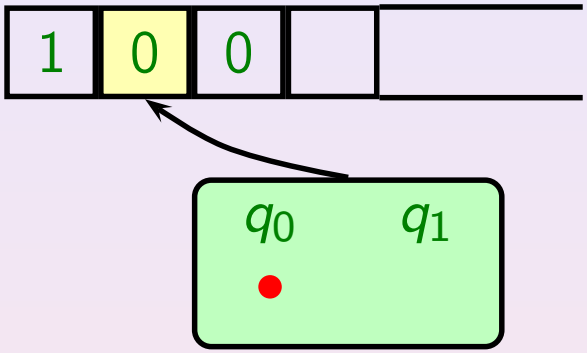
Ejemplo. Cálculo Asociado

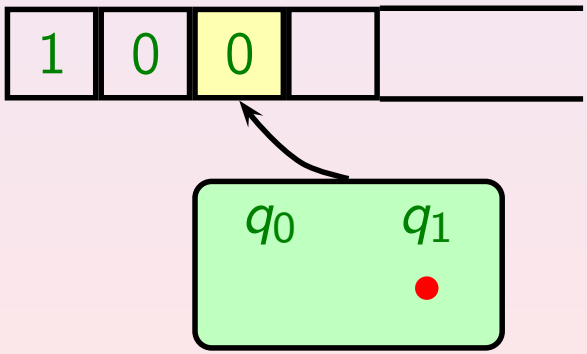


$(q_0, 100) \vdash (q_0, 00) \vdash (q_1, 0) \vdash (q_1, \varepsilon)$

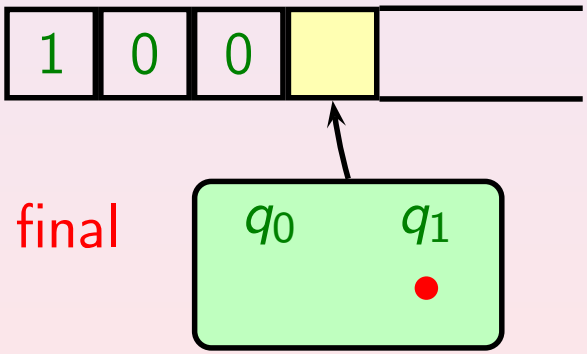
$(q_0, 100) \overset{*}{\vdash} (q_1, \varepsilon)$, 100 aceptada







Estado final
SI



Definición Alternativa del Lenguaje Aceptado por un Autómata Finito: Función de estado

Dado $M = (Q, A, \delta, q_0, F)$, definimos $\delta^* : Q \times A^* \to Q$, como:

- Si $q \in Q$,
  - $\delta^*(q, \varepsilon) = q$
  - $\delta^*(q, au) = \delta^*(\delta(q, a), u)$

$\delta^*(q, u)$ es el estado al que llegaría el autómata si parte del estado q y lee la palabra u.

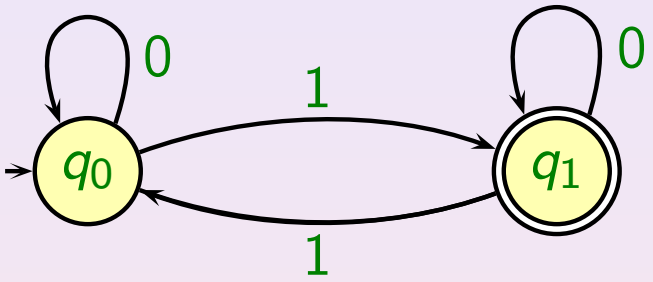
Propiedad

Es inmediato probar que el lenguaje aceptado por un autómata finito se puede expresar como:

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

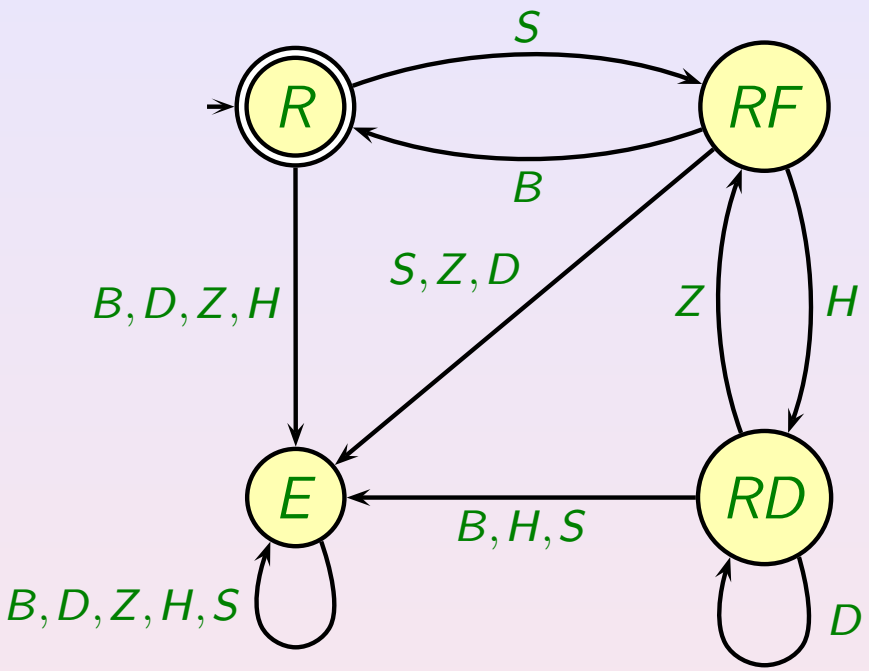
También podremos aplicar δ^* a conjuntos de estados, $\delta^* : \mathcal{P}(Q) \times A^* \to \mathcal{P}(Q)$: $\delta^*(P, u) = \bigcup_{q \in P} \delta^*(q, u)$.

Ejemplo



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

Comunicaciones Correctas



R: Estado de espera

RF: Estado de recepción de ficheros

RD: Recepción de datos

E: Error

S: Comienza recepción

B: Fin de recepción

H: Cabecera de fichero

Z: Fin de fichero

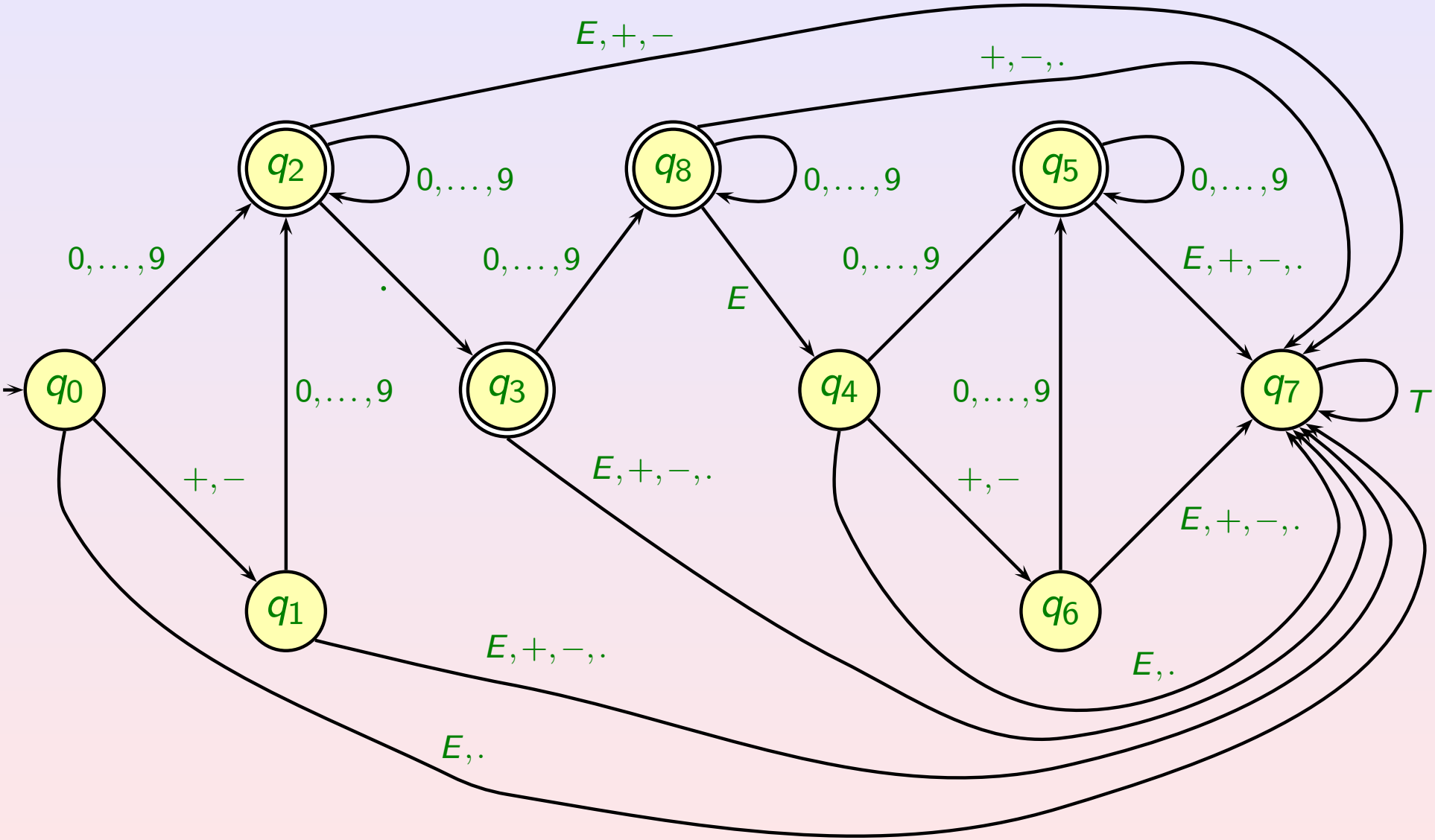
D: Datos

Constantes Reales

Gramática $G = (V, T, P, S)$, donde

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

T es el conjunto de los símbolos terminales.



Autómatas Finitos No Deterministas

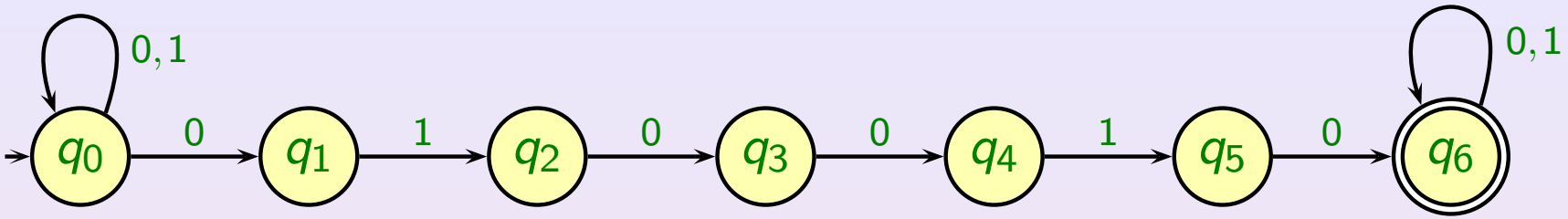
Un autómata finito no determista es una quintupla $M = (Q, A, \delta, q_0, F)$ en la que

- 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

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

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

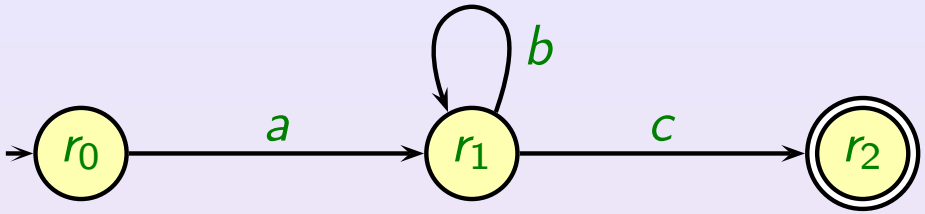
Ejemplo



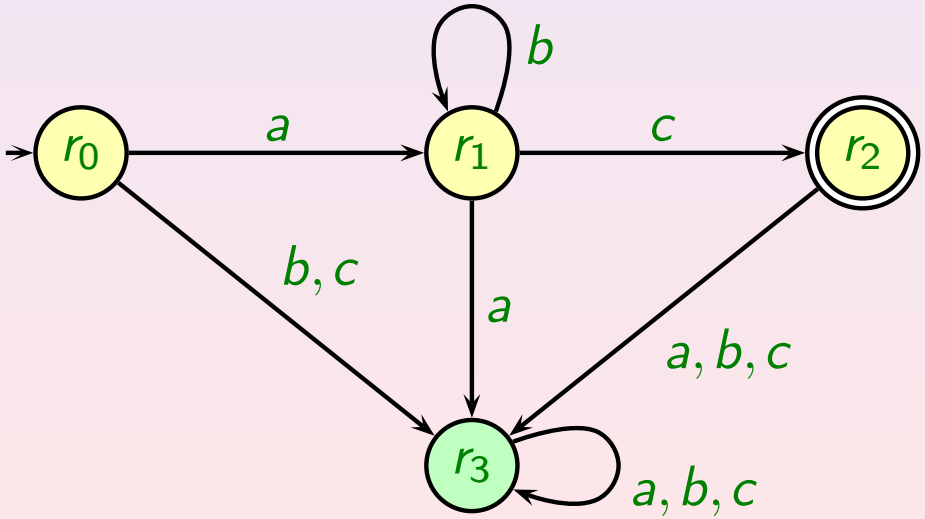
- Se pueden usar también diagramas de transición.
- Puede haber estados que para una entrada tenga dos transiciones. Por ejemplo, q_0 cuando lee 0 puede quedarse en q_0 o pasar a q_1.
- También puede haber estados que para una entrada no tengan ninguna transición: desde q_1 no se puede leer el 0.
- Acepta 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:

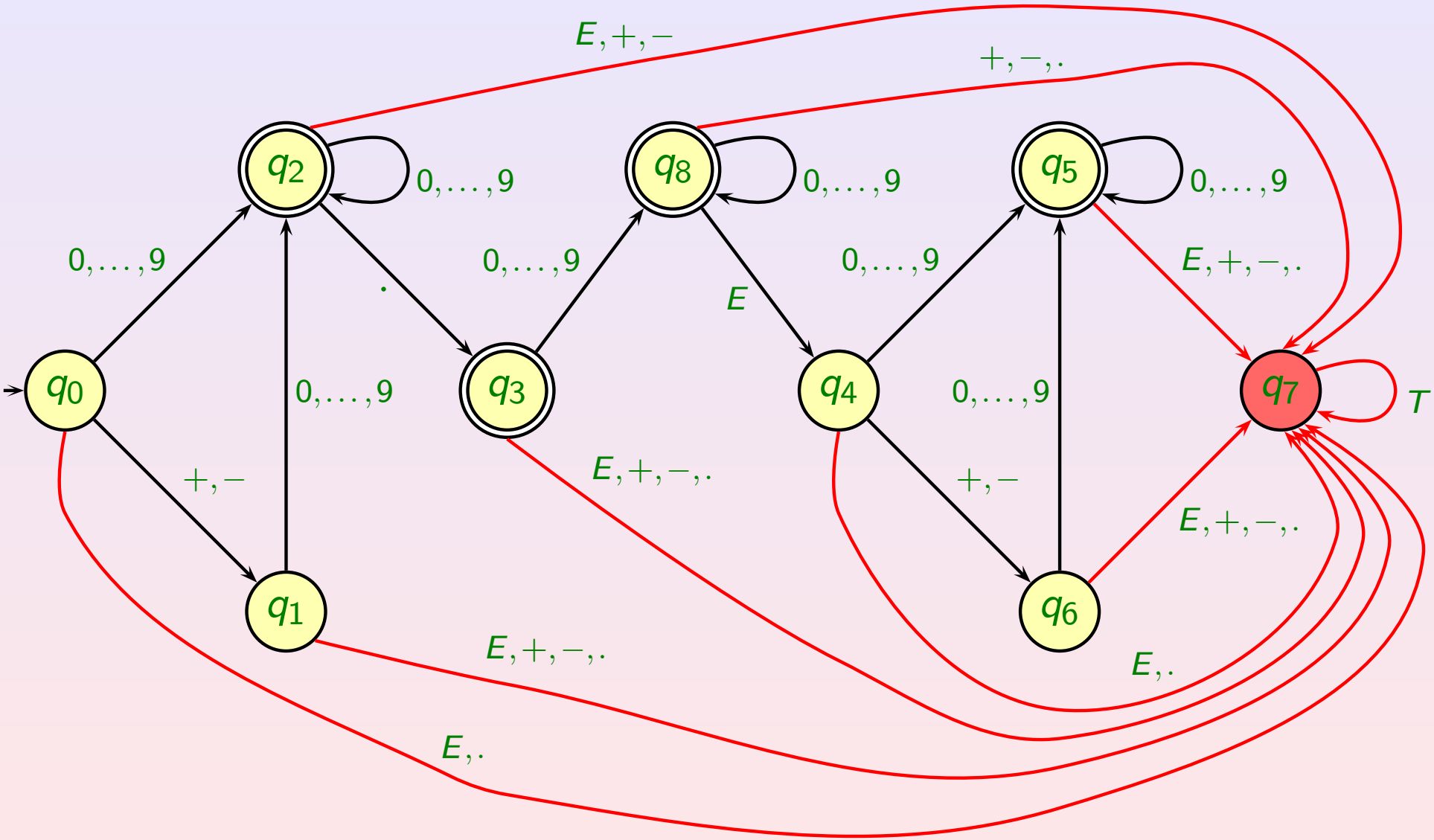


Acepta cadenas formadas por una a, una sucesión de b y una c.
Se puede transformar en uno determinista que acepte el mismo lenguaje añadiéndole un estado de error donde vayan todas las transiciones no definidas en el autómata anterior:



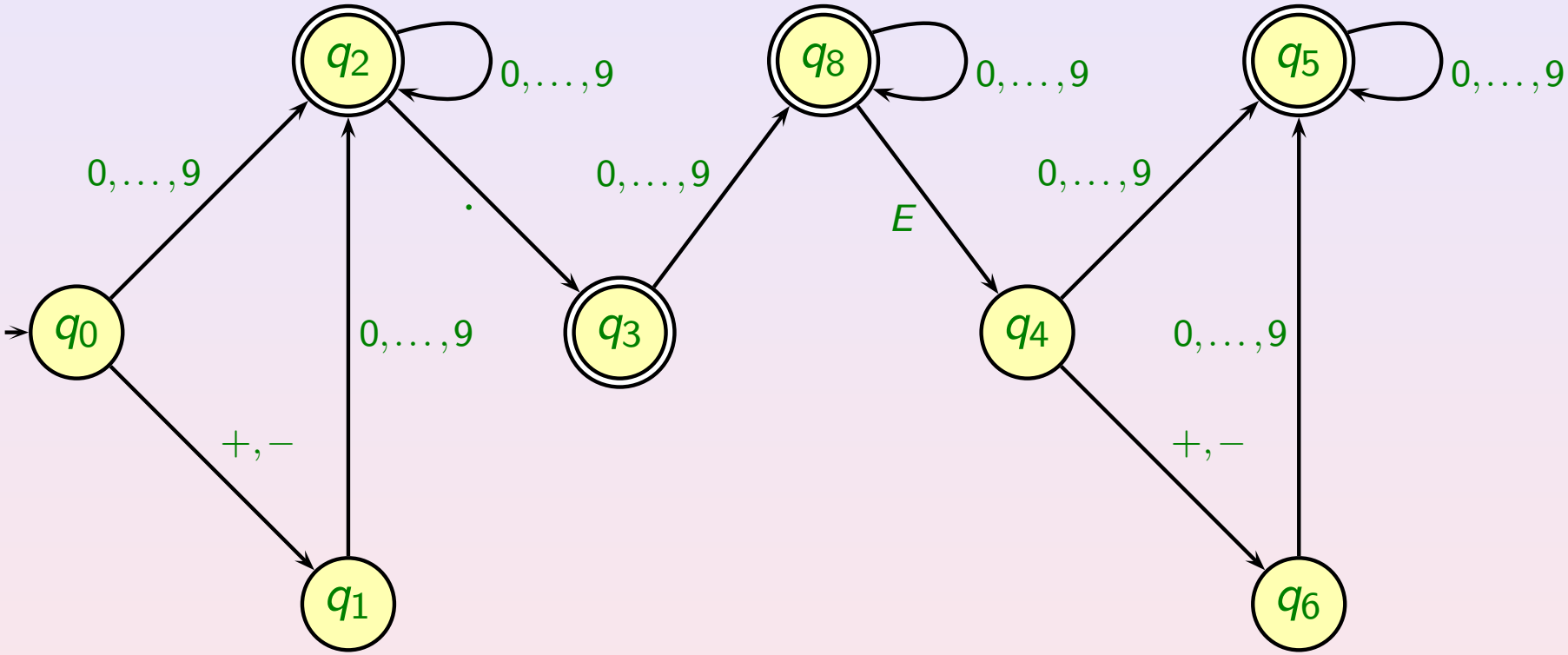
Ejemplo: Constantes reales

Autómata determinista:



Ejemplo: Constantes reales

Autómata no determinista:



PROCESO DE CALCULO

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.

Proceso de Cálculo

- **Relación de cálculo entre dos configuraciones:**
  $((q,u) \overset{*}{\vdash} (p,v))$ si y solo si existe una sucesión de configuraciones $C_0,\ldots,C_n$ 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

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

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

Función de Estado: lenguaje aceptado

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

$\delta^*(B, u)$ 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-Determistas

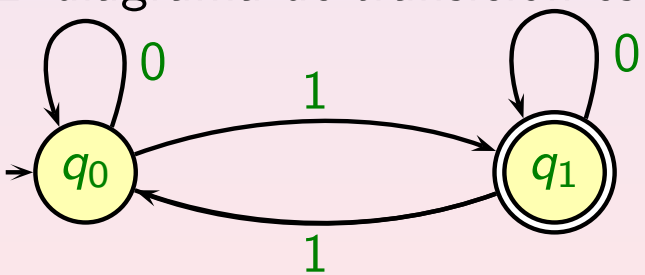
Equivalencia

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-Determistas

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

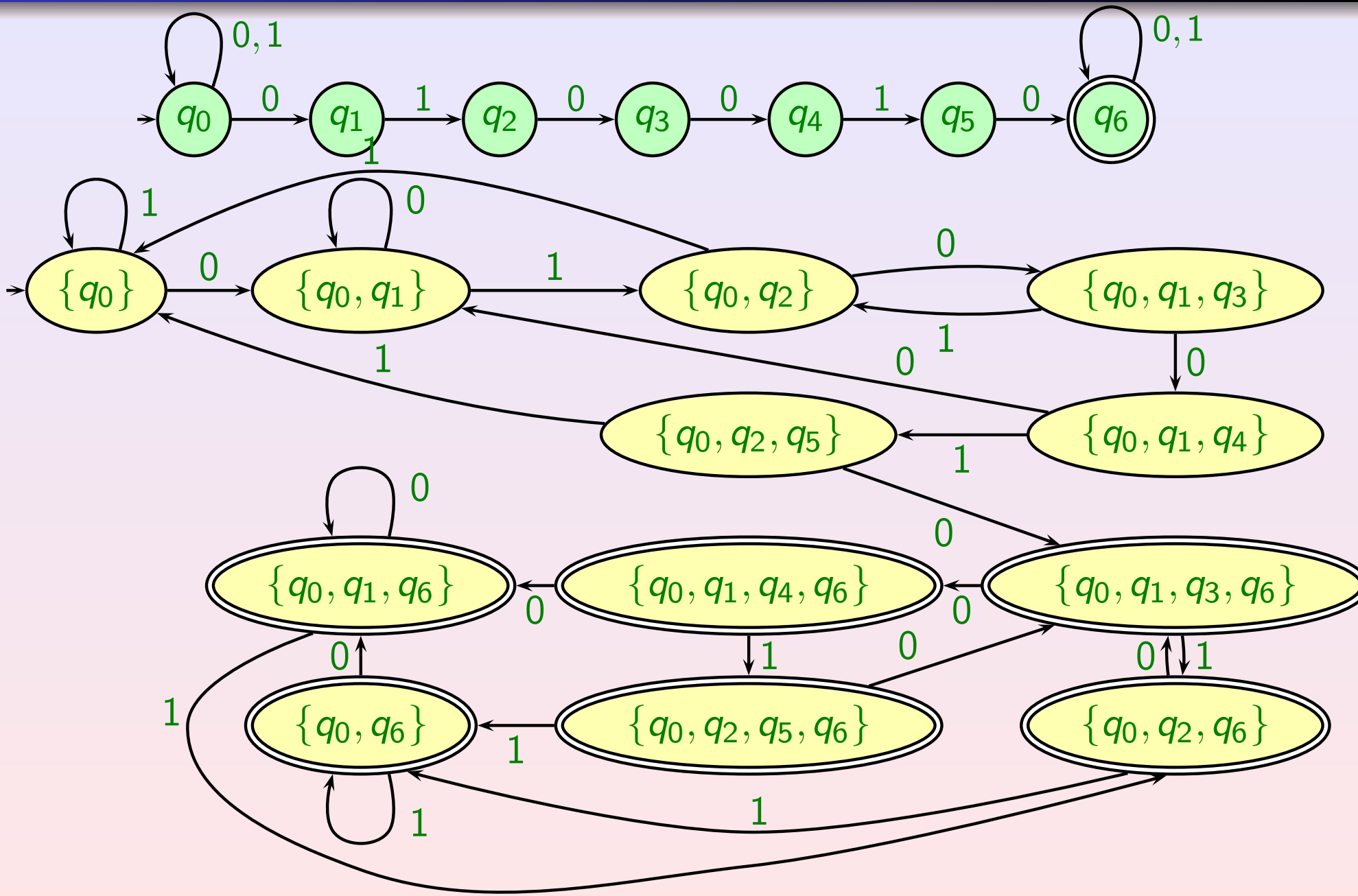
Dado un AFND $M = (Q, A, \delta, q_0, F)$ se llama autómata determinista asociado a M, al autómata $\bar{M} = (\bar{Q}, A, \bar{\delta}, \bar{q}_0, \bar{F})$ dado por

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

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


0, 1
q0
0
q1
1
q2
0
q3
0
q4
1
q5
0
q6
0, 1
1
{q0}
0
{q0, q1}
1
{q0, q2}
0
{q0, q1, q3}
1
0
{q0, q2, q5}
{q0, q1, q4}
1
0
{q0, q1, q6}
{q0, q1, q4, q6}
{q0, q1, q3, q6}
{q0, q6}
{q0, q2, q5, q6}
{q0, q2, q6}

AF No Deterministas con Transiciones Nulas

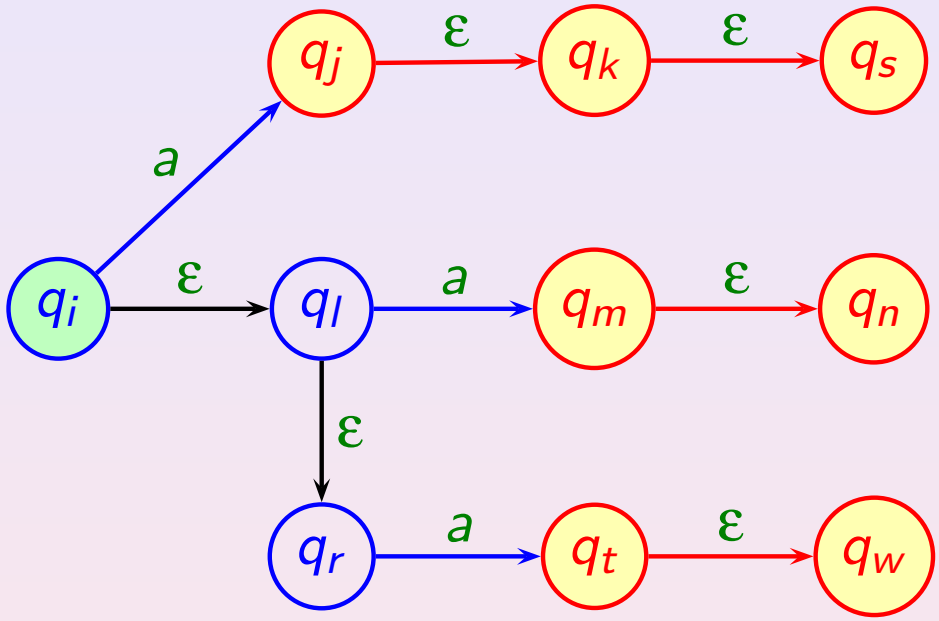
Un autómata finito no determinista con transiciones nulas es una quintupla $M = (Q, A, \delta, q_0, F)$ en la que

- 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

$$\delta : Q \times (A \cup \{\varepsilon\}) \to \wp(Q)$$

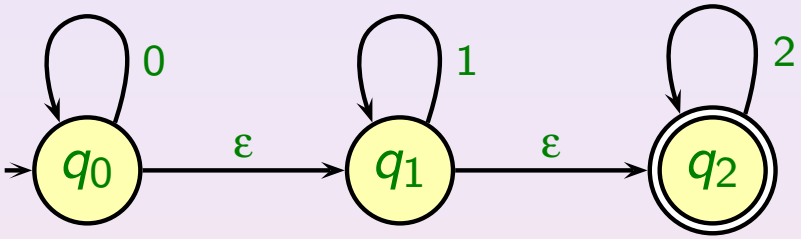
- q_0 es un elemento de Q, llamado estado inicial
- F es un subconjunto de Q, llamado conjunto de estados finales.

Opciones



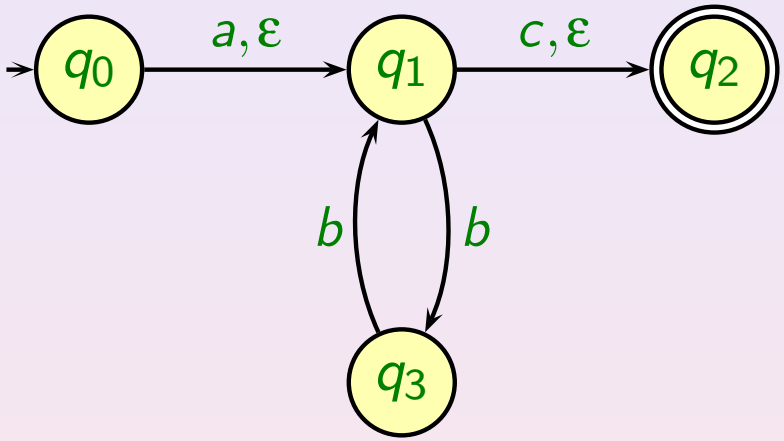
Desde el estado q_i se puede llegar a los estados:
$q_j, q_k, q_s, q_m, q_n, q_t, q_w$ después de leer una a.

Ejemplo



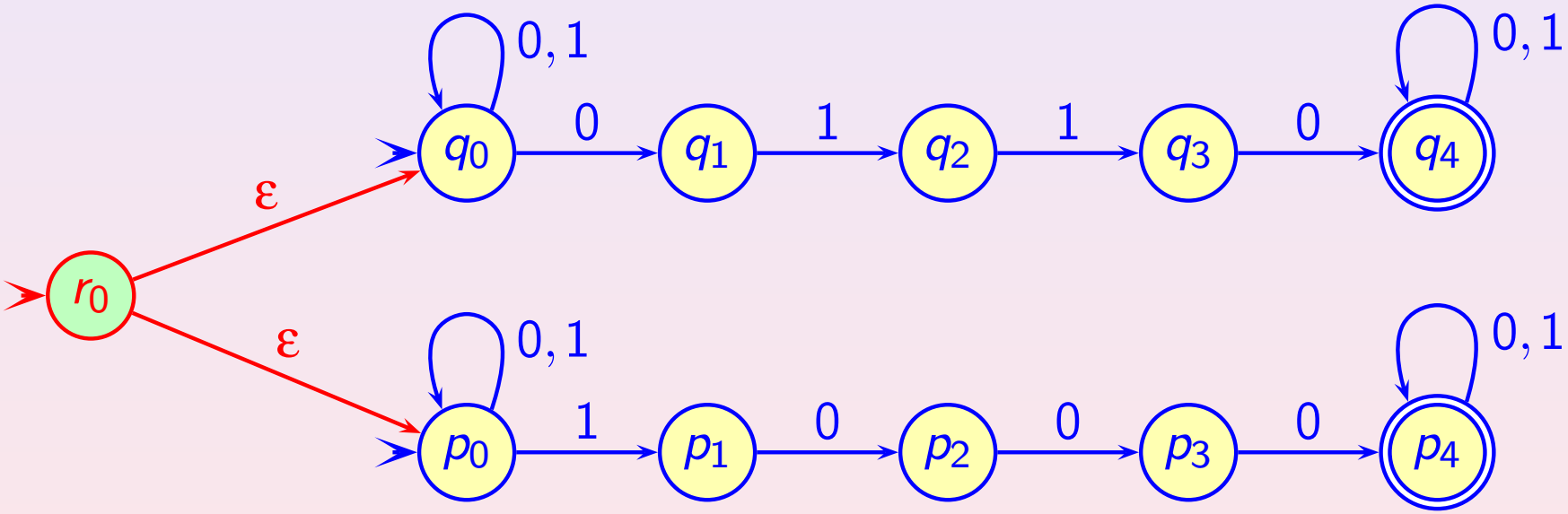
El lenguaje aceptado es $L = \{0^i 1^j 2^k : i, j, k \geq 0\}$.

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.



PROCESO DE CALCULO

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: $((q, av) \vdash (p, v))$)*
  - $((u = v) \wedge p \in \delta(q, \varepsilon))$ *(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.

Proceso de Cálculo

- **Relación de cálculo entre dos configuraciones:**
  $((q,u) \overset{*}{\vdash} (p,v))$ si y solo si existe una sucesión de configuraciones $C_0,\ldots,C_n$ 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

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

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

Función de estados

Definición: Clausura

Dado un autómata no determinista con transiciones nulas $M = (Q, A, \delta, q_0, F)$, definimos:

Clasura de un estado q:

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

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

Función de estados: δ^*

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

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

Si $B \subseteq Q$ y $u \in A^*$, $\delta^*(B, u)$ se define de forma recursiva:

$$\delta^*(B, \varepsilon) = Cl(B)$$
$$\delta^*(B, au) = \delta^*(\delta^*(B, a), u))$$

Finalmente, $\delta^*(p, u) = \delta^*(\{p\}, u)$. $\delta^*(Cl(p), u)$

Lenguaje Aceptado: Función de estados

Propiedad

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

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

AFD $\leftrightarrow$ ε-AFND

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: $L(M) = L(\overline{M})$

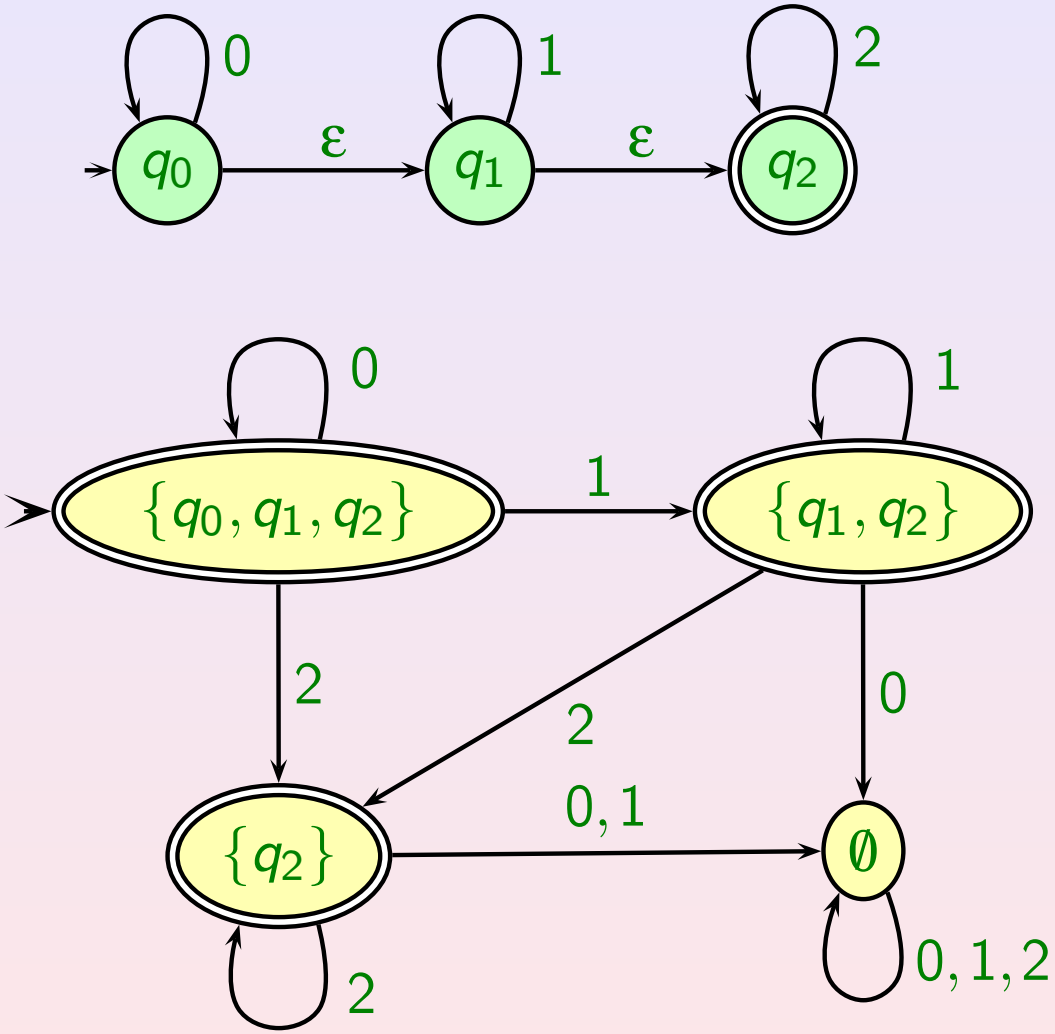
Es inmediato: sería un autómata en el que para cada símbolo del alfabeto de entrada hay siempre una opción y para cada estado $\delta(q, \varepsilon) = \emptyset$.

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 una 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


0
1
2
q_0
ε
q_1
ε
q_2
0
1
$\{q_0, q_1, q_2\}$
1
$\{q_1, q_2\}$
2
2
0
0, 1
$\{q_2\}$
$\emptyset$
2
0, 1, 2

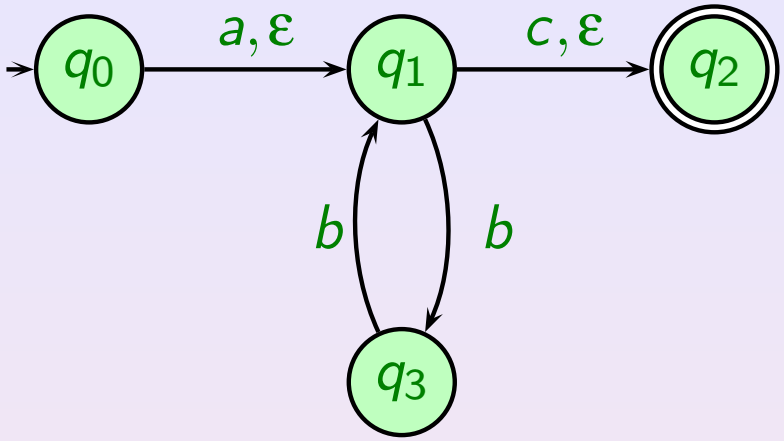
Construcción formal de un autómata determinista

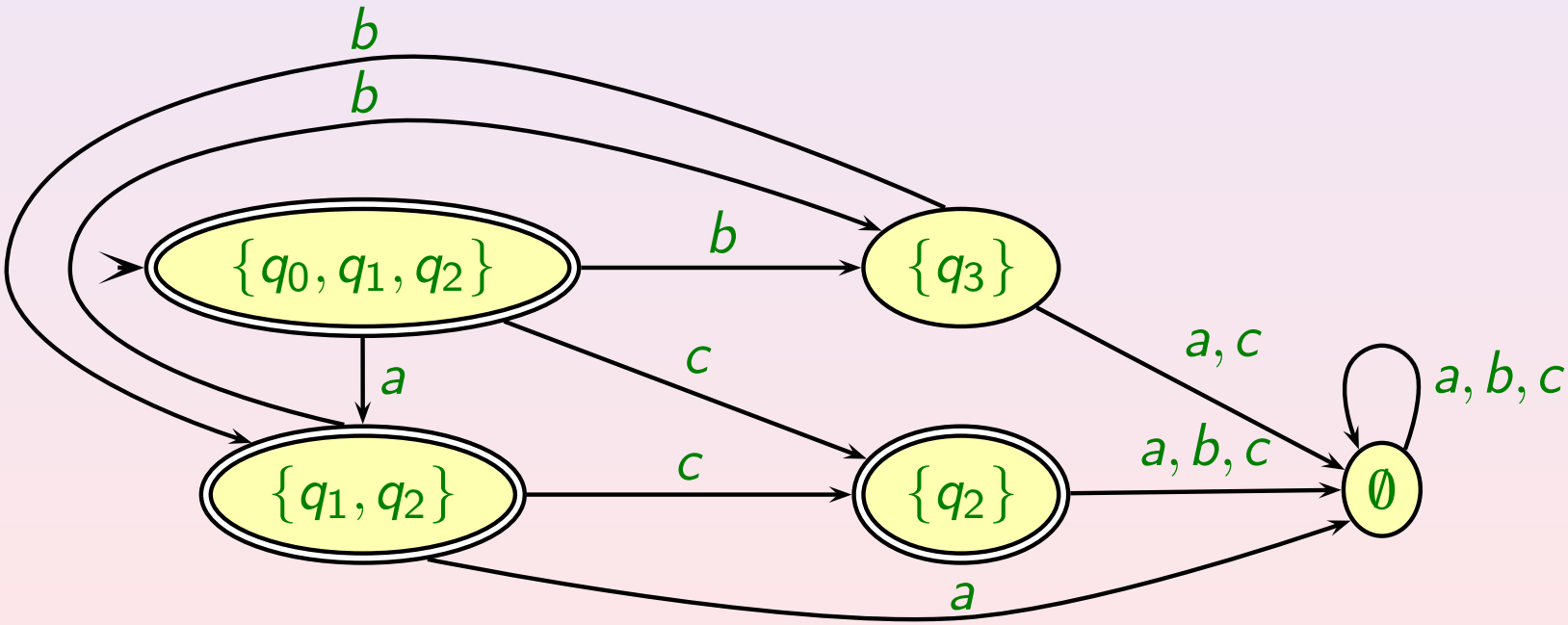
Dado un autómata no determinista con transiciones nulas $M = (Q, A, \delta, q_0, F)$, el **Autómata Finito Determinista** $\overline{M} = (\overline{Q}, A, \overline{\delta}, \overline{q_0}, \overline{F})$ dado por

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

acepta el mismo lenguaje que M.

Ejemplo


q_0
a, ε
q_1
c, ε
q_2
b
b
q_3


b
b
$\{q_0, q_1, q_2\}$
b
$\{q_3\}$
a, c
a, b, c
a
c
$\{q_1, q_2\}$
c
$\{q_2\}$
a, b, c
$\emptyset$
a

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

Ejemplos

$A = \{0, 1\}$

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

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

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

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

$$\mathbf{1^*(01^*01^*)^*}$$

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

$$\mathbf{(0+1)^*0110(0+1)^*}$$

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.

$$\mathbf{(000)(1+10+100)^*}$$

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

$$\mathbf{(0+1)^*(000+101)(0+1)^*}$$

Propiedades de las Expresiones Regulares

1. $r_1 + r_2 = r_2 + r_1$
2. $r_1 + (r_2 + r_3) = (r_1 + r_2) + r_3$
3. $r_1(r_2 r_3) = (r_1 r_2) r_3$
4. $r\varepsilon = r$
5. $r\emptyset = \emptyset$
6. $r + \emptyset = r$
7. $\varepsilon^* = \varepsilon$
8. $r_1(r_2 + r_3) = r_1 r_2 + r_1 r_3$
9. $(r_1 + r_2) r_3 = r_1 r_3 + r_2 r_3$
10. $r^+ + \varepsilon = r^*$
11. $r^* + \varepsilon = r^*$
12. $(r + \varepsilon)^* = r^*$
13. $(r + \varepsilon)^+ = r^*$
14. $\left(r_1^* + r_2^*\right)^* = (r_1 + r_2)^*$

Preguntas sobre Expresiones Regulares

¿Verdadero o falso?

1. Si $\mathbf{r}$ y $\mathbf{s}$ son expresiones regulares, tenemos que siempre se verifica que $(\mathbf{rs})^* = \mathbf{r}^*\mathbf{s}^*$ Falso : $(rs)^* = rsrsrs\ldots$, $r^*s^* = r\ldots rs\ldots s$
2. Si $\mathbf{r}$ y $\mathbf{s}$ son expresiones regulares, tenemos que siempre se verifica que $(\mathbf{r}+\mathbf{s})^* = \mathbf{r}^* + \mathbf{s}^*$ Falso
3. Si $\mathbf{r}_1$ y $\mathbf{r}_2$ son expresiones regulares, tales que su lenguaje asociado contiene la palabra vacía, entonces $(\mathbf{r}_1\mathbf{r}_2)^* = (\mathbf{r}_2\mathbf{r}_1)^*$. V
4. Si $\mathbf{r}$ y $\mathbf{s}$ son expresiones regulares, tenemos que siempre se verifica que $(\mathbf{r}+\varepsilon)^+ = \mathbf{r}^*$ V
5. Si $\mathbf{r}$ y $\mathbf{s}$ son expresiones regulares, tenemos que siempre se verifica que $\mathbf{r}(\mathbf{r}+\mathbf{s})^* = (\mathbf{r}+\mathbf{s})^*\mathbf{r}$ F. La de la dcha puede empezar por r ó s y la dcha. solo r.
6. Si $\mathbf{r}_1$ y $\mathbf{r}_2$ son expresiones regulares, entonces $\mathbf{r}_1^*\mathbf{r}_2^* \subseteq (\mathbf{r}_1\mathbf{r}_2)^*$, en el sentido de que los lenguajes asociados están incluidos. F
7. Si $\mathbf{r}_1$, $\mathbf{r}_2$ y $\mathbf{r}_3$ son expresiones regulares, entonces $(\mathbf{r}_1+\mathbf{r}_2)^*\mathbf{r}_3 = \mathbf{r}_1^*\mathbf{r}_3 + \mathbf{r}_2^*\mathbf{r}_3$. F
8. Si $\mathbf{r}_1$ y $\mathbf{r}_2$ son expresiones regulares entonces: $(\mathbf{r}_1{}^*\mathbf{r}_2{}^*)^* = (\mathbf{r}_1+\mathbf{r}_2)^*$ V

Preguntas sobre Expresiones Regulares

¿Verdadero o falso?

1. Si $\mathbf{r_1}$ y $\mathbf{r_2}$ son expresiones regulares, entonces $(\mathbf{r_1}.\mathbf{r_2})^* = (\mathbf{r_1} + \mathbf{r_2})^*$. ✗
2. Si $\mathbf{r}$ es una expresión regular, entonces $\mathbf{r}^*\mathbf{r}^* = \mathbf{r}^*$. ✓
3. Si $\mathbf{r}$ es una expresión regular, entonces $\mathbf{r}\emptyset = \mathbf{r} + \emptyset$. ✗
4. Si $\mathbf{r}$ es una expresión regular, entonces se verifica que $\mathbf{r}^*\varepsilon = \mathbf{r}^+\varepsilon$ ✗ ($\mathbf{r}^* \neq \mathbf{r}^+$)
5. Si $\mathbf{r_1}$ y $\mathbf{r_2}$ son expresiones regulares, entonce siempre $\mathbf{r_1}(\mathbf{r_2}\mathbf{r_1})^* = (\mathbf{r_1}\mathbf{r_2})^*\mathbf{r_1}$ ✓
6. Si $\mathbf{r}$ y $\mathbf{s}$ son expresiones regulares, entonces $(\mathbf{r}^*\mathbf{s}^*)^* = (\mathbf{r} + \mathbf{s})^*$. ✓
7. Si $\mathbf{r}$ es una expresión regular, entonces $(\mathbf{rr})^* \subseteq \mathbf{r}^*$. ✓
8. Si r_1 y r_2 son expresiones regulares, tales que su lenguaje asociado contiene la palabra vacía, entonces $(\mathbf{r_1}\mathbf{r_2})^* = (\mathbf{r_1} + \mathbf{r_2})^*$. ✓
9. Si $\mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3}$ son expresiones regulares, entonces $\mathbf{r_1}(\mathbf{r_2}^* + \mathbf{r_3}^*) = \mathbf{r_1}\mathbf{r_2}^* + \mathbf{r_1}\mathbf{r_3}^*$. ✓

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.

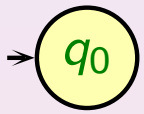
Dada una expresión regular existe un audómata finito que acepta el lenguaje asociado a esta expresión regular.

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

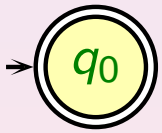
La construcción del autómata va a ser recursiva.

Para las expresiones regulares iniciales tenemos los siguiente autómatas:

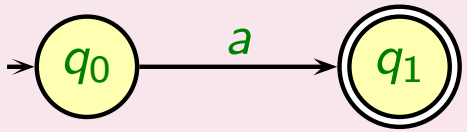
- $\emptyset$



- ε



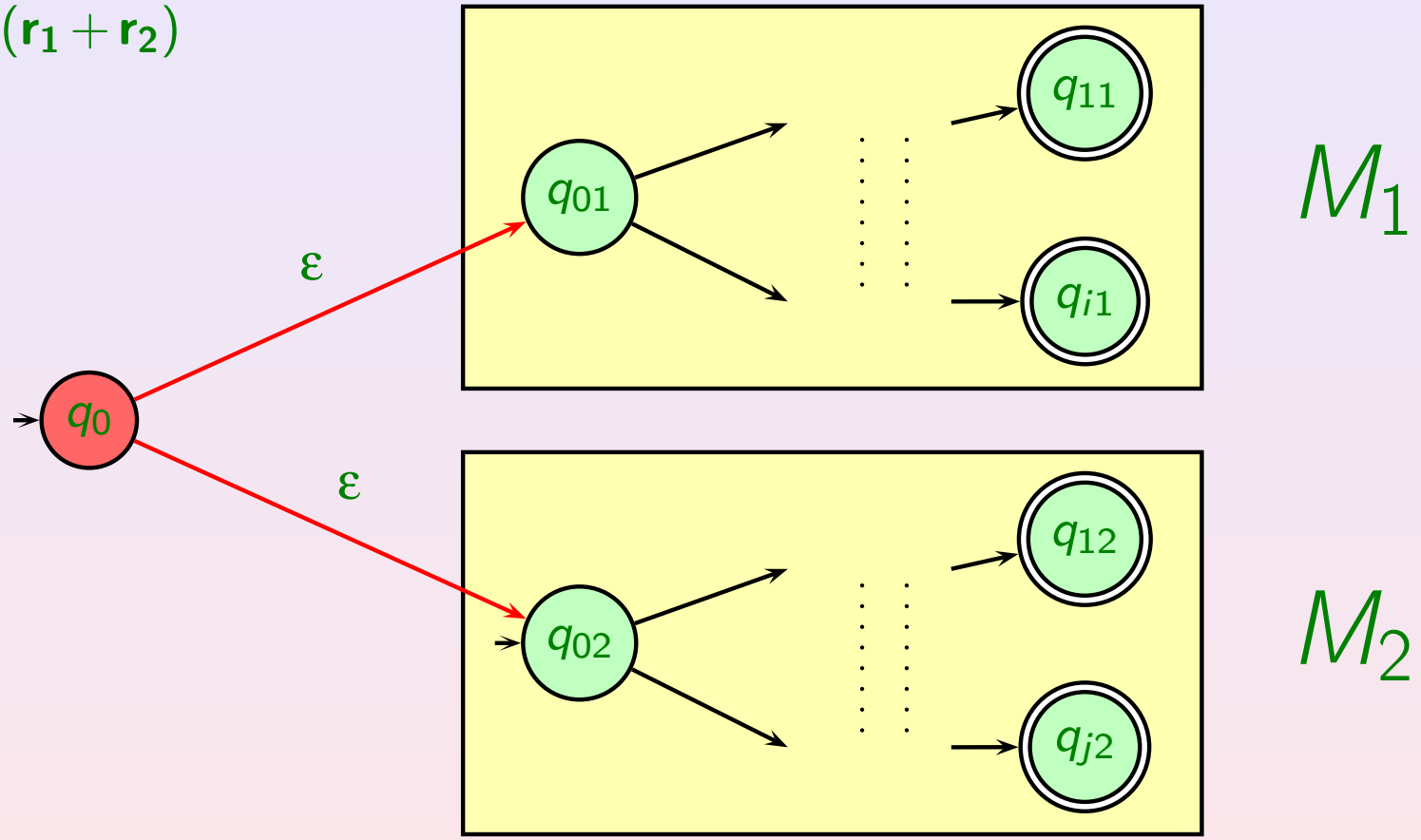
- **a**



Autómatas Compuestos: Unión

Si M_1 es el autómata que acepta el mismo lenguaje que el representado por $\mathbf{r_1}$ y M_2 el que acepta el mismo lenguaje que el de $\mathbf{r_2}$, entonces

- **Unión** $(\mathbf{r_1} + \mathbf{r_2})$



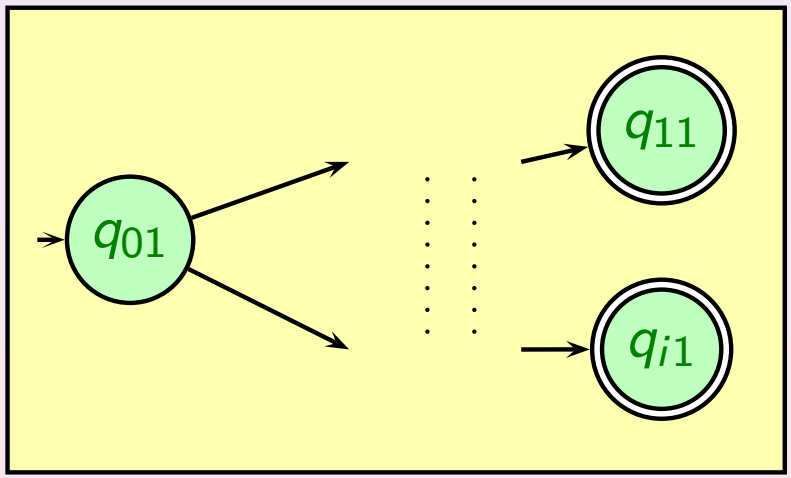
Unión: Expresión Matemática

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

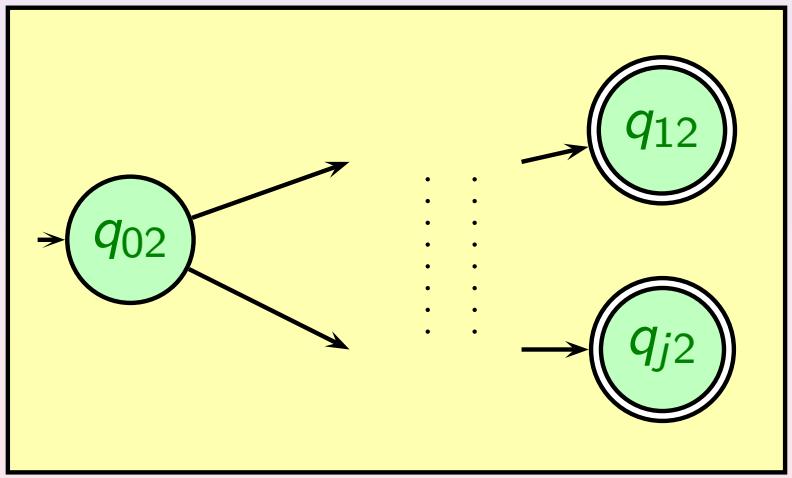
- $Q = Q_1 \cup Q_2 \cup \{q_0\}$ donde $q_0 \notin (Q_1 \cup Q_2)$ es un nuevo estado.
- δ viene definida por
  - $\delta(q, a) = \delta_1(q, a)$ si $q \in Q_1$, $a \in A$
  - $\delta(q, \varepsilon) = \delta_1(q, \varepsilon)$ si $q \in Q_1$
  - $\delta(q, a) = \delta_2(q, a)$ si $q \in Q_2$, $a \in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

Autómatas Compuestos: Concatenación

- **Concatenación:** El autómata para la expresión $(\mathbf{r_1 r_2})$ es



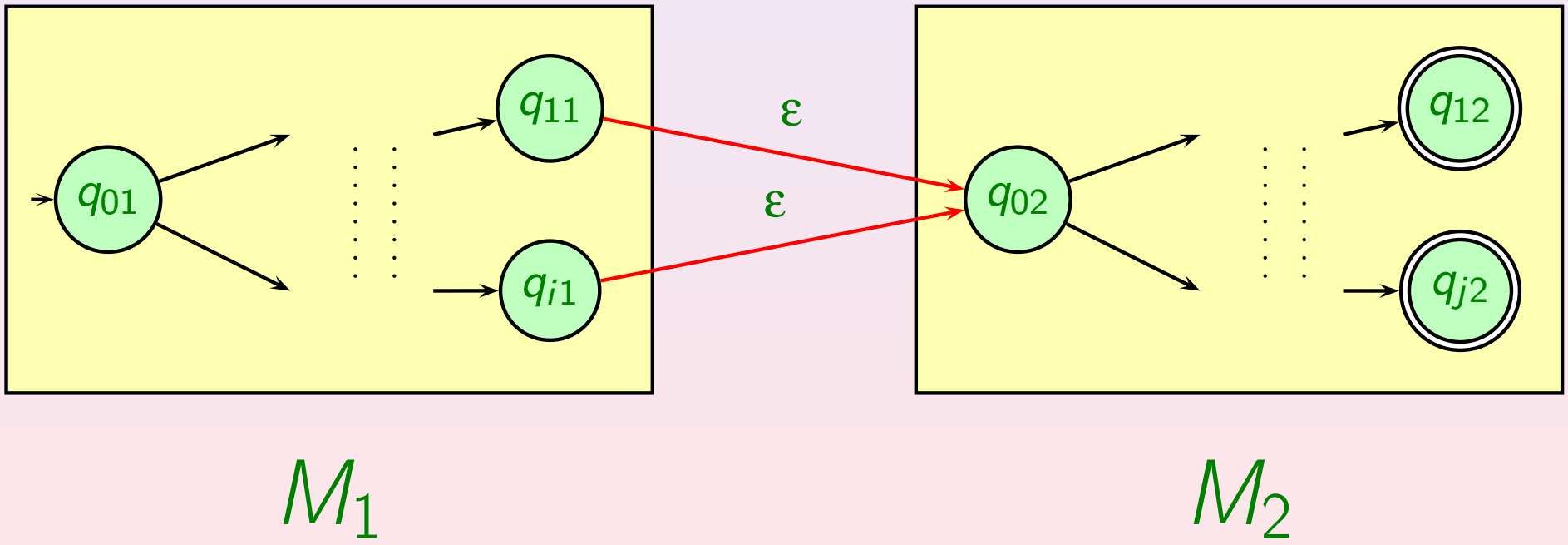
M_1



M_2

Autómatas Compuestos: Concatenación

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



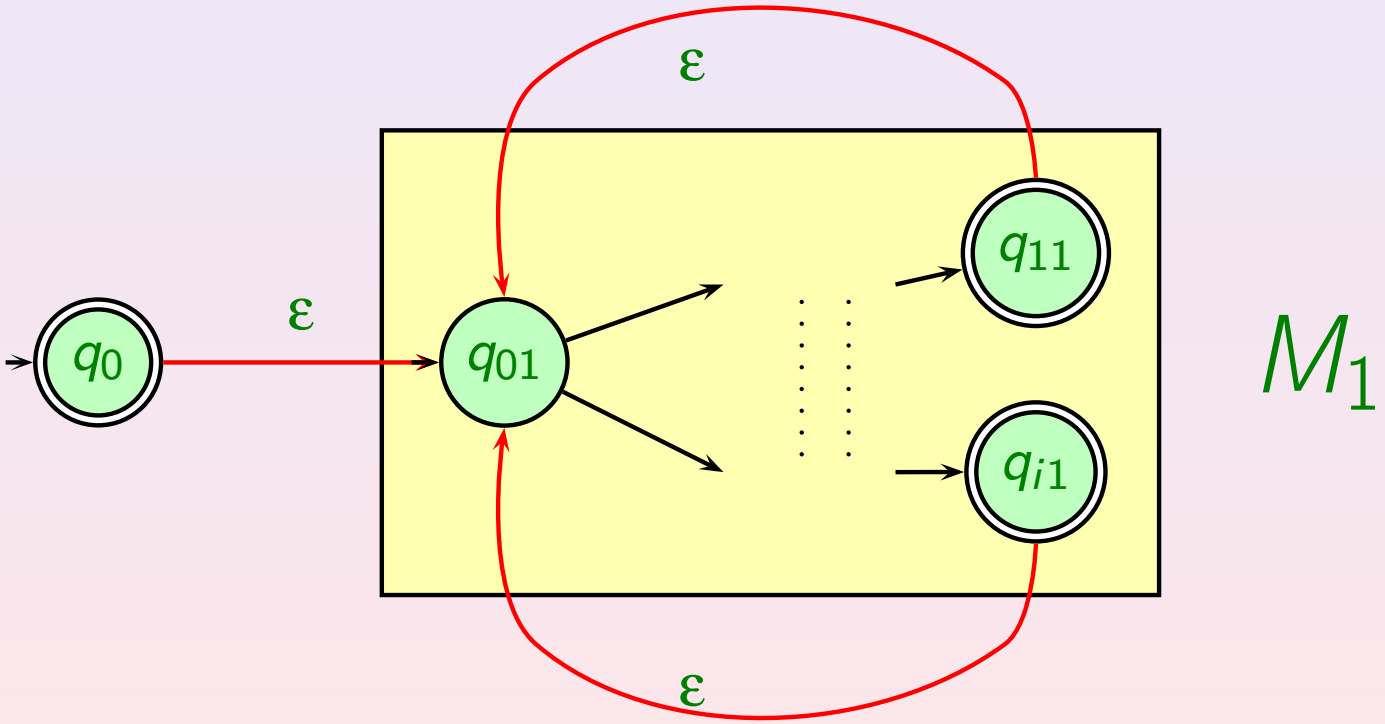
Concatenación: Expresión Matemática

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 $q \in Q_1 \setminus F_1$, $a \in A$
  - $\delta(q, \varepsilon) = \delta_1(q, \varepsilon)$ si $q \in Q_1 \setminus F_1$
  - $\delta(q, a) = \delta_1(q, a)$ si $q \in F_1, a \in A$
  - $\delta(q, \varepsilon) = \delta_1(q, \varepsilon) \cup \{q_0^2\}$ si $q \in F_1$
  - $\delta(q, a) = \delta_2(q, a)$ si $q \in Q_2$
  - $\delta(q, \varepsilon) = \delta_2(q, \varepsilon)$ si $q \in Q_2$
- $q_0 = q_0^1$
- $F = F_2$

Autómatas Compuestos: Clausura

- **Clausura:** El autómata para $\mathbf{r_1^*}$ es



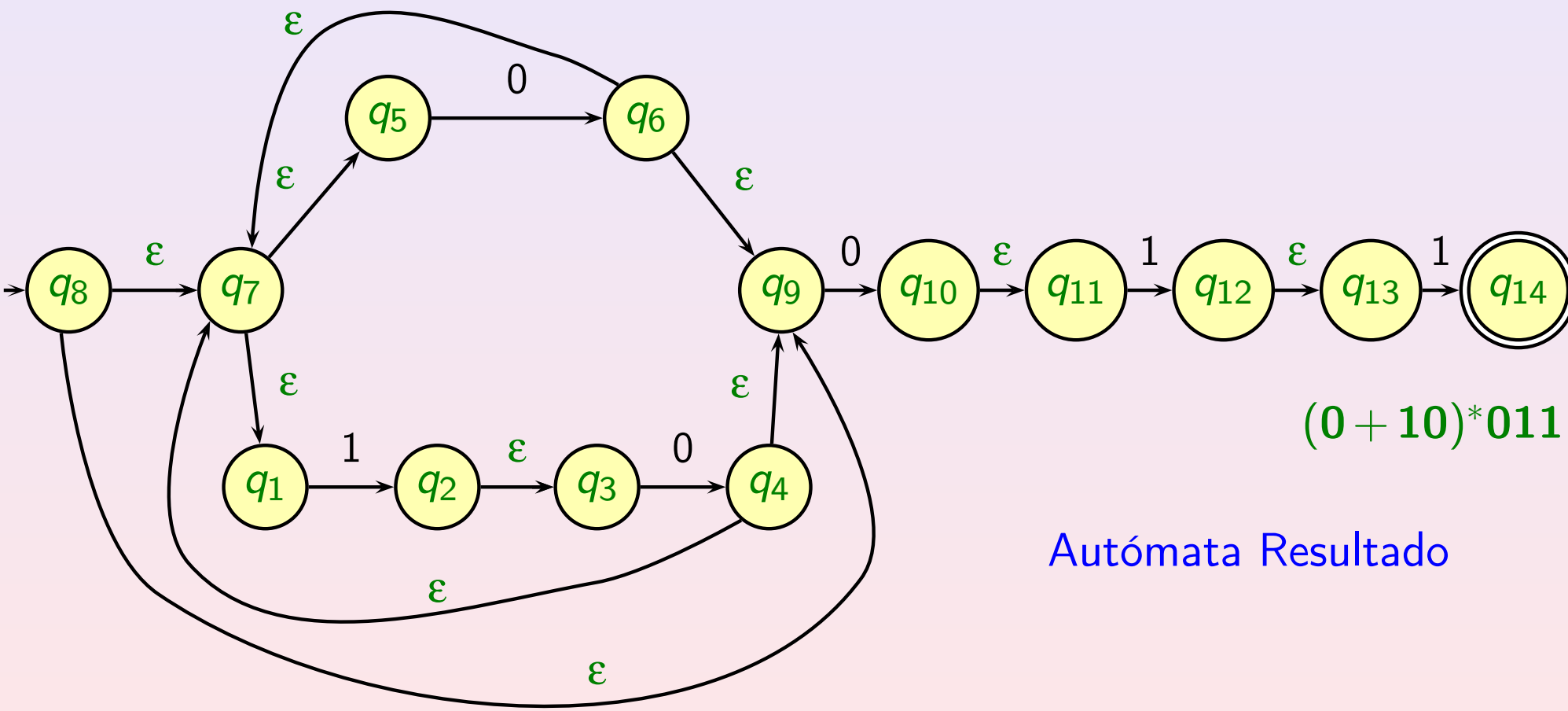
Clausura: Expresión Matemática

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 $(\mathbf{0}+\mathbf{10})^*\mathbf{011}$



$(\mathbf{0}+\mathbf{10})^*\mathbf{011}$

Autómata Resultado

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

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

Sea R_{ij}^k el conjunto de las cadenas de A^* que premiten 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 $u = a_1 \ldots a_m$ está en R_{ij}^k si y solo si todo estado $q_{i_l} = \delta^*(q_i, a_1 \ldots a_l)$ donde $1 \leq l \leq m-1$ es tal que $i_l \leq k$.

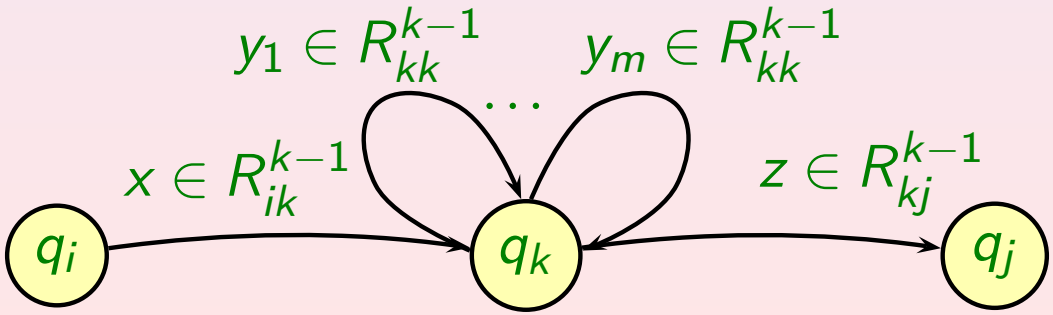
Cálculo de R_{ij}^k

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

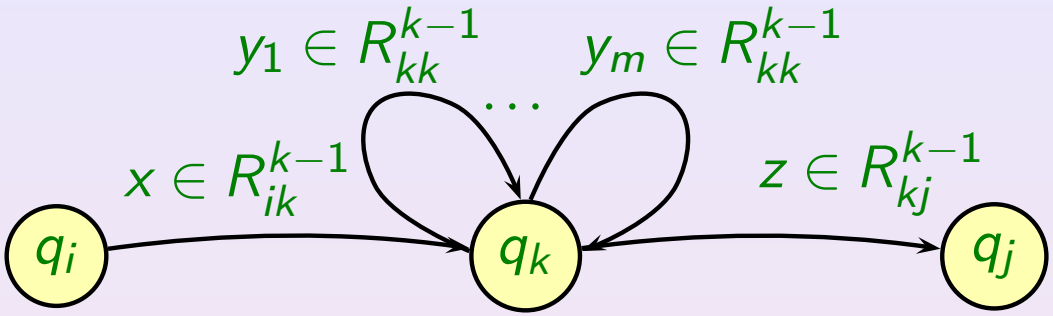
$$R_{ij}^0 = \begin{cases} \{a : \delta(q_i, a) = q_j\} & \text{si } i \neq j \\ \{a : \delta(q_i, a) = q_i\} \cup \{\varepsilon\} & \text{si } i = j \end{cases}$$

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:



Cálculo de R_{ij}^k



Como la palabra $y_1 \ldots y_m \in \left(R_{kk}^{k-1}\right)^*$, 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 $\mathbf{r_{ij}^k}$ a R_{ij}^k

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

Para $k = 0$ es inmediato.

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

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

Si este conjunto es vacío la expresión regular sería:

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

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

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

Expresión regular del lenguaje $L(M)$

Expresión Regular del lenguaje aceptado por el autómata

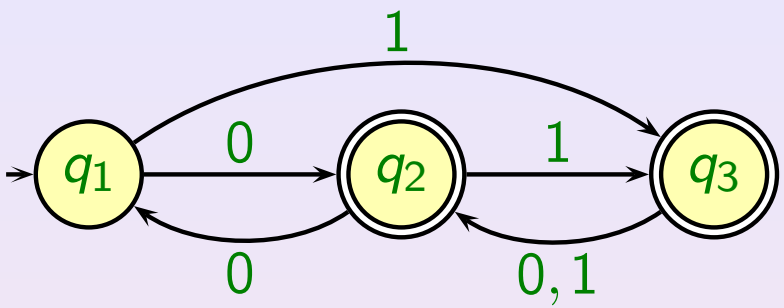
$$L(M) = \bigcup_{q_j \in F} R_{1j}^n$$

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

$$\mathbf{r_{1j_1}^n + \ldots + r_{1j_k}^n}$$

donde $F = \{q_{j_1}, \ldots, q_{j_k}\}$ 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

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

Ejemplo

$$\begin{cases} r_{11}^1 = \varepsilon \\ r_{12}^1 = 0 \\ r_{13}^1 = 1 \\ r_{21}^1 = 0 \\ r_{22}^1 = \varepsilon + 00 \\ r_{23}^1 = 1 + 01 \\ r_{31}^1 = \emptyset \\ r_{32}^1 = 0 + 1 \\ r_{33}^1 = \varepsilon \end{cases}$$

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

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:

$$r_{12}^3 + r_{13}^3 = r_{12}^2 + r_{13}^2 (r_{33}^2)^* r_{32}^2 + r_{13}^2 + r_{13}^2 (r_{33}^2)^* r_{33}^2 =$$

$$0(00)^* + 0^*1(\varepsilon + (0+1)0^*1)^*(0+1)(00)^* + 0^*1 + 0^*1(\varepsilon + (0+1)0^*1)^*(\varepsilon + (0+1)0^*1) =$$

$$0(00)^* + 0^*1((0+1)0^*1)^*(0+1)(00)^* + 0^*1 + 0^*1((0+1)0^*1)^*$$

Método Alternativo: Ecuaciones Expresiones Regulares

Ecuación de Expresiones Regulares

Si $\mathbf{r_1}, \ldots, \mathbf{r_n}$ son variables representando expresiones regulares, una ecuación para la expresión regular $\mathbf{r_i}$ es una expresión de la forma:

$$\mathbf{r_i} = \alpha_i + \beta_{i1}\mathbf{r_1} + \cdots \beta_{in}\mathbf{r_n}$$

donde $\alpha_i, \beta_{i1}, \ldots, \beta_{in}$ son expresiones regulares concretas.

Solución

Si $\mathbf{r_1}, \ldots, \mathbf{r_n}$ son variables representando expresiones regulares y tenemos una ecuación para cada expresión regular, una solución es una asignación de una e.r. concreta a cada variable $\mathbf{r_i}$ que satisfaga todas las ecuaciones.

Método de resolución

- Si tenemos la ecuación

$$\mathbf{r_i} = \alpha_i + \beta_{i1}\mathbf{r_1} + \cdots + \beta_{in}\mathbf{r_n}$$

y $\mathbf{r_i}$ no aparece en la parte derecha entonces se sustituye cada aparición de $\mathbf{r_i}$ en las otras ecuaciones por $\alpha_i + \beta_{i1}\mathbf{r_1} + \cdots + \beta_{in}\mathbf{r_n}$ y se aplican las reglas de cálculo con e.r. para dejarla como una nueva ecuación donde ha desaparecido la variable $\mathbf{r_i}$.

Método de resolución

- Si en la parte derecha aparece $\mathbf{r_i}$:

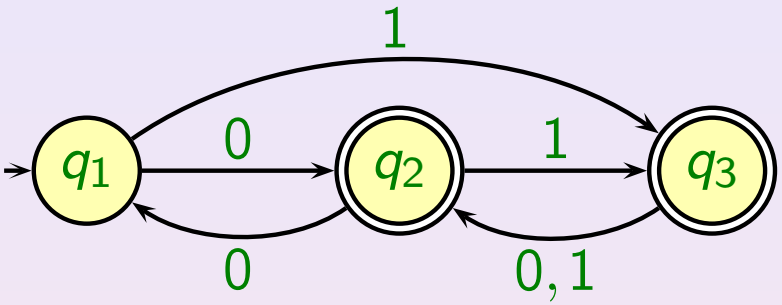
$$\mathbf{r_i} = \alpha + \beta \mathbf{r_i}$$

  donde en α pueden aparecer otras variables, pero no en β, se aplica el lema de Arden para despejar $\mathbf{r_i}$.
  Si β no incluye la palabra vacía, entonces la única solución de la anterior e.r. es $\mathbf{r_i} = \beta^* \alpha$.
  Ahora ya se puede sustituir $\mathbf{r_i}$ en el resto de las ecuaciones.
- Así se continúa hasta encontrar el valor de una de las variables. Sustituyendo en las ecuaciones usadas, podemos encontrar el valor de las otras variables.

De Autómata Finito a Sistema de Ecuaciones

- Se puede aplicar a autómatas no deterministas, pero no con transiciones nulas (puede dar lugar a ecuaciones en las que no exista una única solución).
- Hay una variable e.r. $\mathbf{q_i}$ por cada estado q_i: indica el conjunto de palabras tales que leyéndose desde el estado q_i permiten llegar a un estado final.
- Por cada estado q_i se añade una ecuación con $\mathbf{q_i}$ en la parte izquierda y en la que la derecha es una suma que se calcula añadiendo los siguientes términos:
  - Si q_i es final se suma ε
  - Si $q_j \in \delta(q_i, a)$, entonces se suma $a\mathbf{q_j}$
- Una vez resuelto el sistema, la expresión regular del lenguaje asociado al autómata es $\mathbf{q_0}$, donde q_0 es el estado inicial.

Ejemplo



Ecuaciones:

$$\mathbf{q_1} = 0\mathbf{q_2} + 1\mathbf{q_3}$$
$$\mathbf{q_2} = \varepsilon + 0\mathbf{q_1} + 1\mathbf{q_3}$$
$$\mathbf{q_3} = \varepsilon + 0\mathbf{q_2} + 1\mathbf{q_2} = \varepsilon + (0+1)\mathbf{q_2}$$

Resolución

$$\mathbf{q_1} = 0\mathbf{q_2} + 1\mathbf{q_3}$$
$$\mathbf{q_2} = \varepsilon + 0\mathbf{q_1} + 1\mathbf{q_3}$$
$$\mathbf{q_3} = \varepsilon + (0+1)\mathbf{q_2}$$

Sustituímos $\mathbf{q_3}$ en la primera y segunda ecuaciones y nos queda:

$$\mathbf{q_1} = 0\mathbf{q_2} + 1(\varepsilon + (0+1)\mathbf{q_2})$$
$$\mathbf{q_2} = \varepsilon + 0\mathbf{q_1} + 1(\varepsilon + (0+1)\mathbf{q_2})$$

Reorganizando,

$$\mathbf{q_1} = 1 + (0 + 1(0+1))\mathbf{q_2}$$
$$\mathbf{q_2} = \varepsilon + 1 + 0\mathbf{q_1} + 1(0+1)\mathbf{q_2}$$

Resolución

$$\mathbf{q_1} = 1 + (0 + 1(0 + 1))\mathbf{q_2}$$
$$\mathbf{q_2} = \underbrace{\varepsilon + 1 + 0\mathbf{q_1}}_{\alpha} + \underbrace{1(0 + 1)}_{\beta}\mathbf{q_2}$$

En la ecuación segunda se aplica el lema de Arden:

$$\mathbf{q_1} = 1 + (0 + 1(0 + 1))\mathbf{q_2}$$
$$\mathbf{q_2} = (1(0 + 1))^*(\varepsilon + 1 + 0\mathbf{q_1})$$

Resolución

$$\mathbf{q_1} = 1 + (0 + 1(0 + 1))\mathbf{q_2}$$
$$\mathbf{q_2} = (1(0 + 1))^*(\varepsilon + 1 + 0\mathbf{q_1})$$

Reorganizamos:

$$\mathbf{q_1} = 1 + (0 + 1(0 + 1))\mathbf{q_2}$$
$$\mathbf{q_2} = (1(0 + 1))^*(\varepsilon + 1) + (1(0 + 1))^*0\mathbf{q_1})$$

Sustituimos $\mathbf{q_2}$ en la primera ecuación y tenemos:

$$\mathbf{q_1} = 1 + (0 + 1(0 + 1))((1(0 + 1))^*(\varepsilon + 1) + (1(0 + 1))^*0\mathbf{q_1}))$$

Resolución

Partimos de:

$$\mathbf{q_1} = 1 + (0 + 1(0 + 1))((1(0 + 1))^*(\varepsilon + 1) + (1(0 + 1))^* 0\mathbf{q_1}))$$

Reorganizando:

$$\mathbf{q_1} = 1 + (0 + 1(0 + 1))(1(0 + 1))^*(\varepsilon + 1)) + (0 + 1(0 + 1))(1(0 + 1))^* 0)\mathbf{q_1}$$

Aplicando el lema de Arden:

$$\mathbf{q_1} = ((0 + 1(0 + 1))(1(0 + 1))^* 0))^* (1 + (0 + 1(0 + 1))(1(0 + 1))^*(\varepsilon + 1)))$$

que es una e.r. del lenguaje aceptado por el autómata (ya que q_1 es el estado inicial).

Expresiones Regulares en Unix - Estándar IEEE POSIX

Expresión Regular	Significado
Caracteres Normales	Ellos mismos
$+,*$	Superíndices $+,*$
$\|$	La unión de los lenguajes
$[\ldots]$	Cualquier símbolo entre corchetes
$[a-b]$	todos los caracteres entre a y b
$[\hat{}\ldots]$	El complementario de $[\ldots]$
?	0 ó 1 repetición de lo anterior
$\{nombre\}$	Se substituye la e.r. *nombre*
$\{n\}$	n repeticiones de la anterior e.r.
$\{n,m\}$	entre n y m repeticiones de la anterior e.r.
$\backslash *$	El carácter $*$
"..."	Los caracteres entre comillas literalmente
$\hat{}$,$	Principio, fin de línea
.	Cualquier carácter excepto el salto de línea
\n	salto de línea

Estructura de un fichero lex

```
nombre1     er1
nombre2     er2
nombrei     eri
            declaglobal1
            declaglobal2
%%
            declalocal1
            declalocal2
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()
- yyleng – 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(ïdentificador%s\n",yytext);}
.|\n                            {;}
%%
yywrap()
        {printf("Numero de Enteros%d, reales%d, ident%d,
                    Suma de Enteros%d",ent,real,ident,sumaent); return 1;}
```

Procedimiento

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 intrucciones como **grep**: *'Global (Searh for) Regular Expressions and Print'*

Aplicaciones de las Expresiones Regulares

- *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.

Gramáticas Regulares ó tipo 3

- *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

$$A \rightarrow Bu$$

$$A \rightarrow u$$

Ejemplo

Gramática Lineal por la Derecha:

$S \to 0A, \quad A \to 10A, \quad A \to \varepsilon$

Expresión Regular

$$0(10)^*$$

Gramática Lineal por la Izquierda:

$$S \to S10, \quad S \to 0$$

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

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

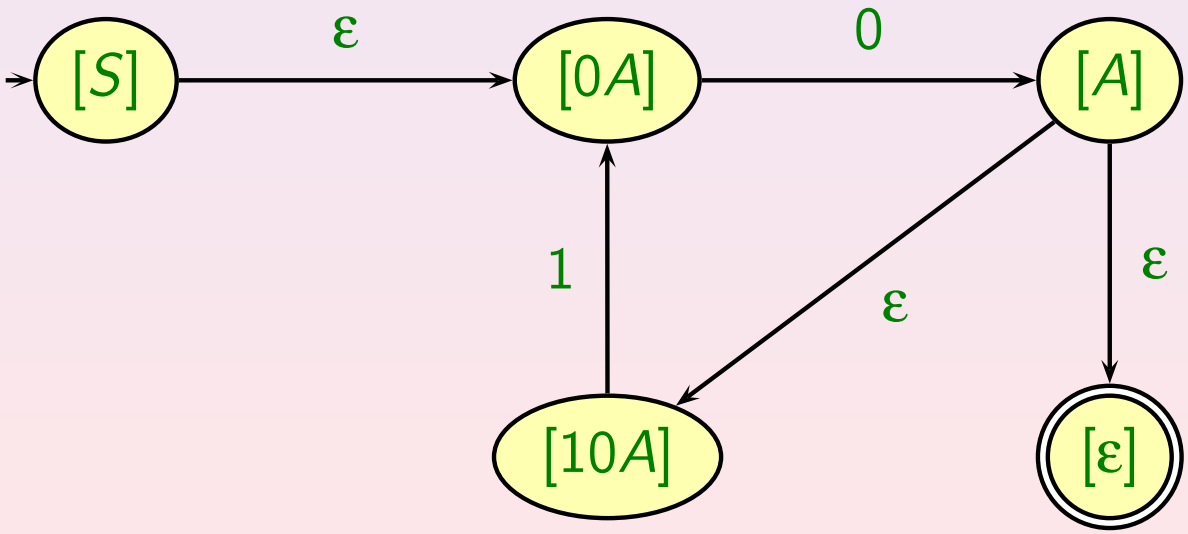
$$\delta([a\alpha], a) = [\alpha]$$

Ejemplo

Sea la gramática:

$$S \to 0A, \quad A \to 10A, \quad A \to \varepsilon$$

El autómata 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

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

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.

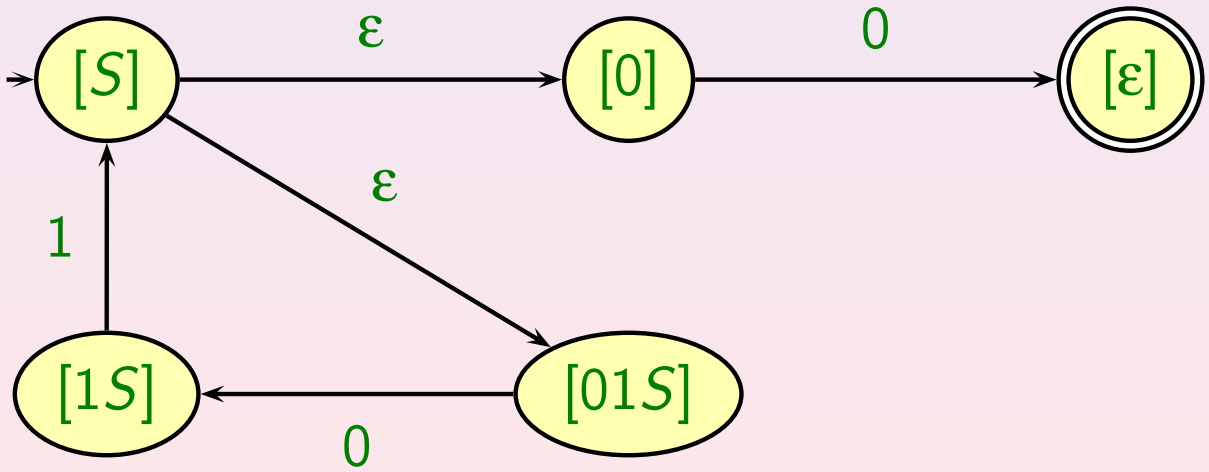
El lenguaje aceptado por M es: $L(M')^{-1} = L(G')^{-1} = \left(L(G)^{-1}\right)^{-1} = L(G)$

Ejemplo

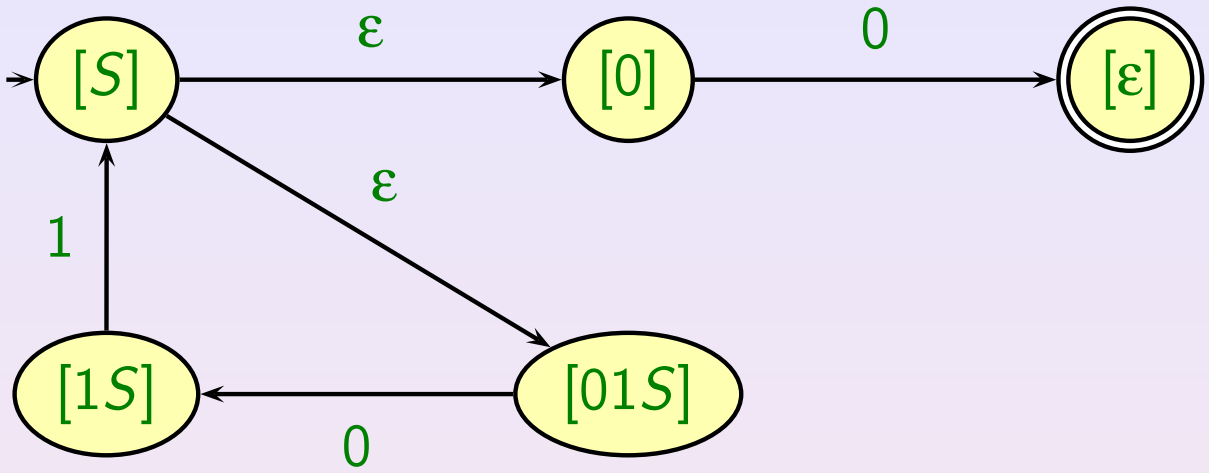
$S \to S10, \qquad S \to 0$

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

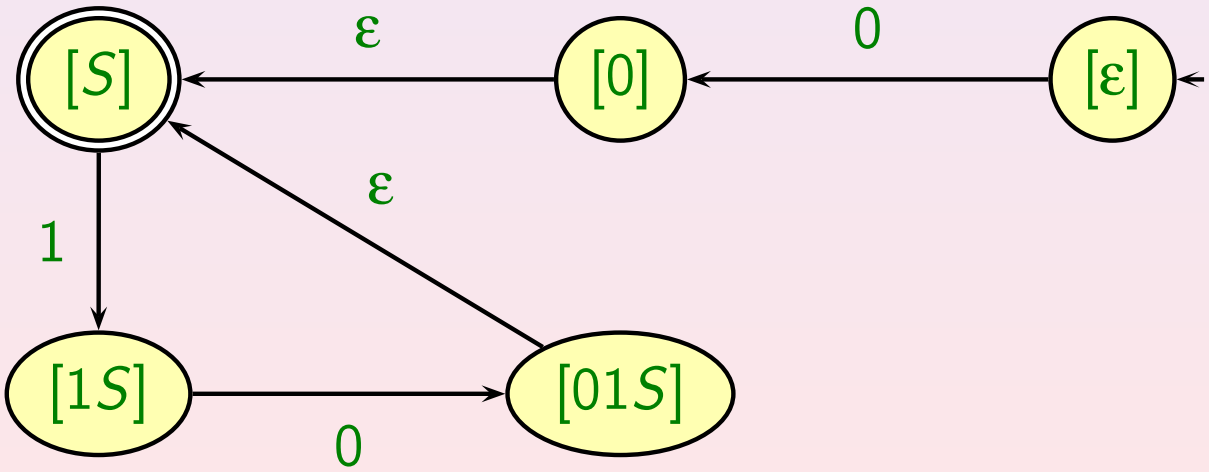
- Invertir la parte derecha de las producciones:
  $S \to 01S$
  $S \to 0$
- Construir el AFND con transiciones nulas asociado



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,

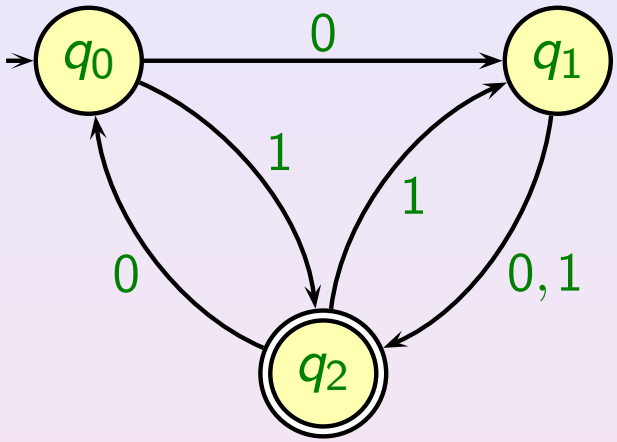
$$p \rightarrow aq, \quad \text{si } \delta(p, a) = q$$

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

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

Ejemplo: Gramática Lineal por la Derecha

Consideremos el autómata:



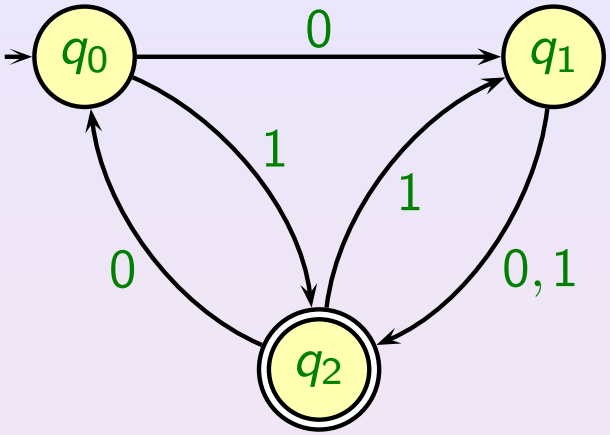
La gramática es (variable inicial q_0):

$$q_0 \to 0q_1, \qquad q_0 \to 1q_2, \qquad q_1 \to 0q_2, \qquad q_1 \to 1q_2$$

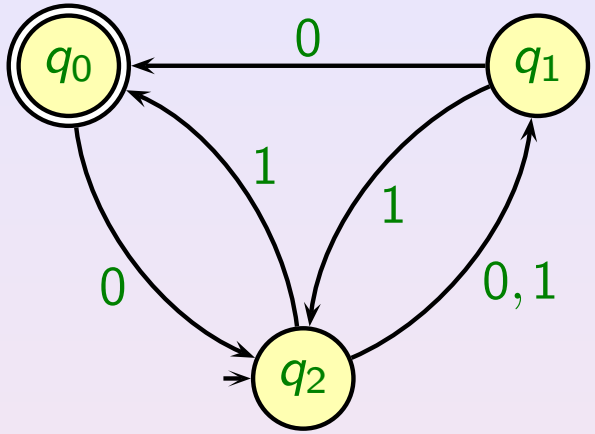
$$q_2 \to 0q_0, \qquad q_2 \to 1q_1, \qquad q_2 \to \varepsilon$$

Ejemplo: Gramática Lineal por la Izquierda

Autómata de Partida:



Invertimos el autómata:



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

$$q_1 \to 0q_0, \qquad q_2 \to 1q_0, \qquad q_2 \to 0q_1, \qquad q_2 \to 1q_1$$

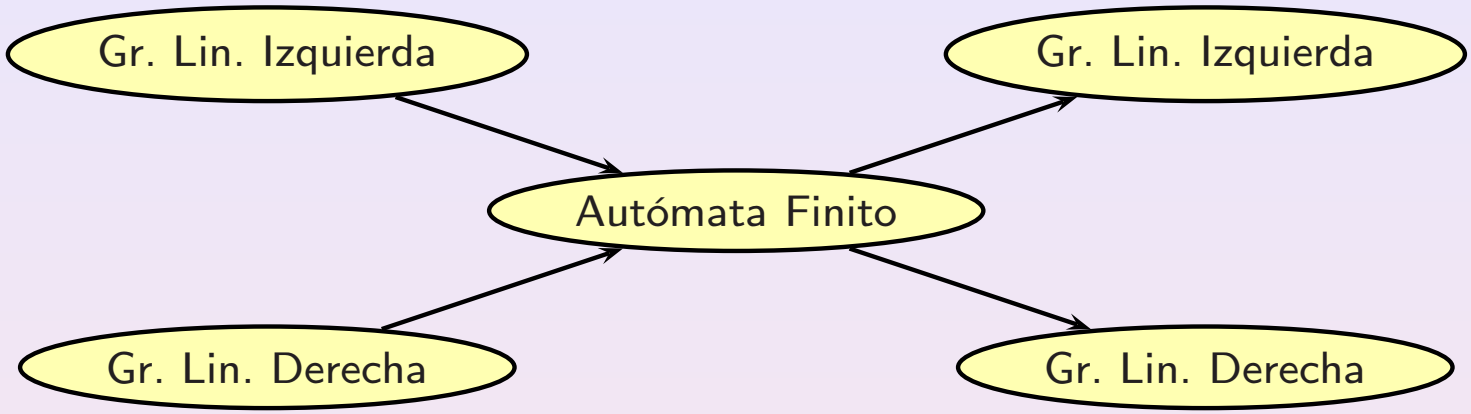
$$q_0 \to 0q_2, \qquad q_1 \to 1q_2, \qquad q_0 \to \varepsilon$$

Invertimos la parte derecha de las producciones:

$$q_1 \to q_00, \qquad q_2 \to q_01, \qquad q_2 \to q_10, \qquad q_2 \to q_11$$
$$q_0 \to q_20, \qquad q_1 \to q_21, \qquad q_0 \to \varepsilon$$

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.