TEMA 1

INTRODUCCIÓN

OBJETIVOS

- Aclarar el **significado** y la **relación** existente entre los siguientes conceptos:

Teorías de Gramáticas

Lenguajes

Máquinas Abstractas (Autómatas)

LENGUAJES

- Lenguaje **natural**
- Lenguaje **artificial**

Conjunto de **palabras** (cadenas o sentencias) que están formadas por **símbolos** de un **alfabeto**.

Español → {casa, loro, ...}

GRAMÁTICAS

- Estructura del lenguaje → Formas **válidas** en que se pueden combinar los **símbolos** de un alfabeto.

GRAMÁTICAS

- **Lenguajes Formales**: van a obedecer a reglas **preestablecidas** , no evolucionan y se crean para un fin **específico**.
- **Lenguajes naturales**: utilizados por el **hombre** y cuyas reglas gramaticales han sido desarrolladas con **posterioridad**.

GRAMÁTICAS

- **Lenguajes Formales**: van a obedecer a reglas **preestablecidas** , no evolucionan y se crean para un fin **específico**.
- **Lenguajes naturales**: utilizados por el **hombre** y cuyas reglas gramaticales han sido desarrolladas con **posterioridad**.

MÁQUINA ABSTRACTA

Autómatas

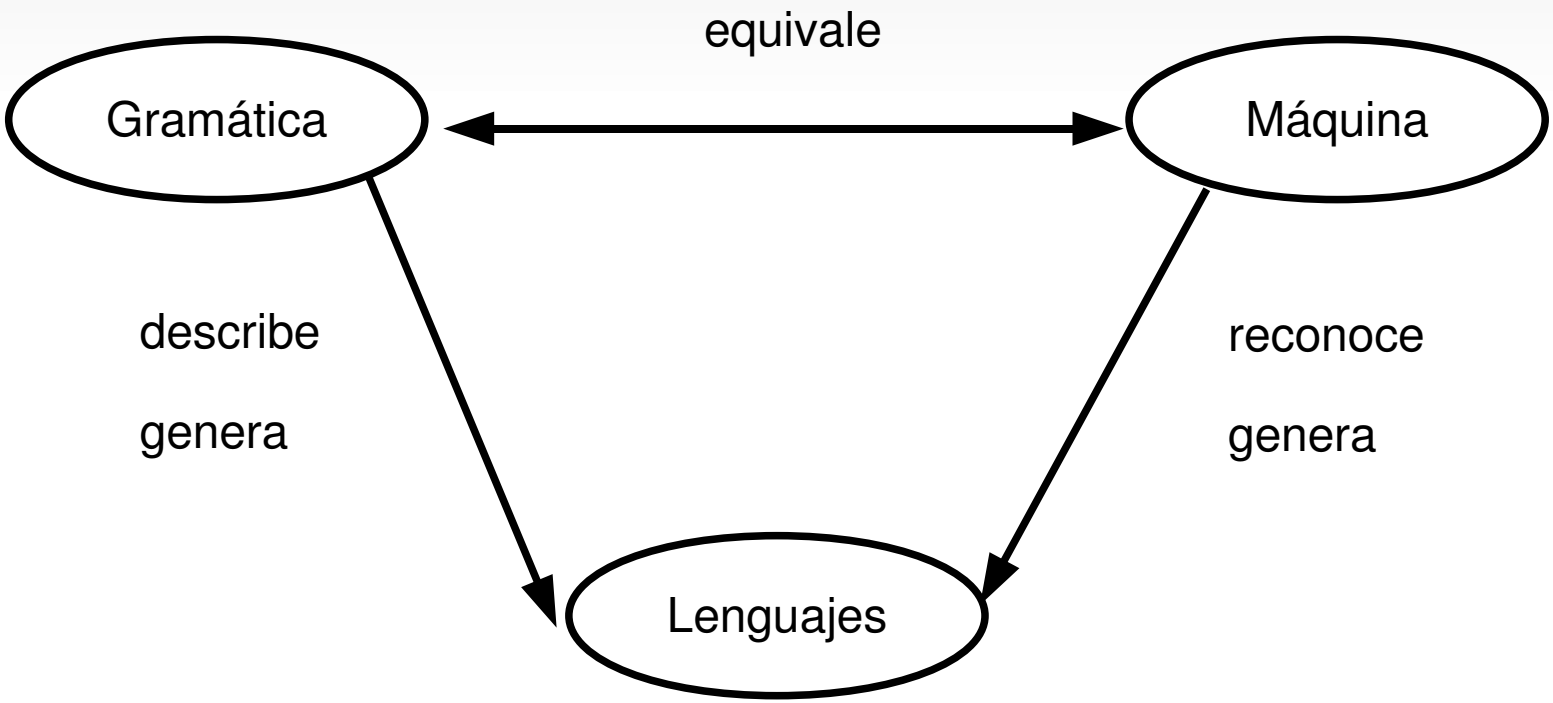
- Dispositivo teórico capaz de **recibir** y **transmitir** información.

(entrada + estado → salida)

(entrada + estado + memoria → salida)

CONEXIÓN

- Trabajos **iniciales** sobre **gramáticas** y **máquinas** abstractas tienen orígenes **distintos**.



GRAMÁTICA

- Origen → Estudios de **CHOMSKY** para la búsqueda de **formalización** de oraciones en lenguaje natural.

$$G3 \subseteq G2 \subseteq G1 \subseteq G0$$

GRAMÁTICA

- Origen → Estudios de **CHOMSKY** para la búsqueda de **formalización** de oraciones en lenguaje natural.

$$G3 \subseteq G2 \subseteq G1 \subseteq G0$$

¿Cuál es la más restrictiva?

GRAMÁTICA

- **Tipo 0**: Gramática **sin restricciones** o de estructura de frase.
- **Tipo 1**: Gramáticas **sensibles** al contexto.
- **Tipo 2**: Gramáticas **independientes** del contexto
- **Tipo 3**: Gramática **regulares**.

Un **lenguaje** se llama **del tipo i** (i= 0,1,2,3) si existe una **gramática** G del **tipo i** capaz de **generar** ese lenguaje.

MÁQUINAS

- Origen → Estudios de **SHANNON** y **MOORE**. A finales de los 50 relación útil gramática-autómata.

Se establece una **conexión** entre los lenguajes **generados** por ciertos tipos de **gramáticas** y los lenguajes **reconocibles** por ciertas **máquinas**

LENGUAJES- GRAMÁTICAS- MÁQUINAS

- Reglas de una Gramática para un Lenguaje:

Una **cadena** de símbolos **pertenece** al correspondiente **lenguaje** si la cadena se formó **obedeciendo** a esas reglas.

- Máquina aceptadora de lenguaje (cadena → lógico):

Máquina reconoce lenguaje **L** si **acepta** todas las sentencias que **pertenecen** a L y no acepta ninguna que **no pertenezca**

LENGUAJES- GRAMÁTICAS- MÁQUINAS

Gramáticas	Lenguajes	Máquinas
Sin restricciones o de Tipo 0	Sin restricciones o de Tipo 0	Máquina de Turing
Sensible al contexto o de Tipo 1	Sensible al contexto o de Tipo 1	Autómata linealmente acotado
Libre de contexto o de Tipo 2	Libre de contexto o de Tipo 2	Autómata a pila
Regular o de Tipo 3	Regular o de Tipo 3	Autómata Finito

TEMA 1

INTRODUCCIÓN

TEMA 2

LENGUAJES FORMALES

CONTENIDOS

- Definiciones básicas
- Operaciones con palabras.
- Operaciones con lenguajes.
- Definiciones complementarias.

DEFINICIONES BÁSICAS

- **Alfabeto** ($\sum$): conjunto no vacío **finito** de símbolos.

$\sum_1$={na, pa, la, bra}

- **Palabra**: secuencia **finita** de símbolos de un alfabeto.

lapa, nala, branala, ...

DEFINICIONES BÁSICAS

- **Longitud** de una palabra (|x|) : número de **símbolos** del **alfabeto** que la forman.

|branala| = ?

DEFINICIONES BÁSICAS

- **Longitud** de una palabra (**|x|**) : número de **símbolos** del **alfabeto** que la forman.

  Sobre $\sum_1$ |branala| = 3

  Sobre Alf. Español |branala|= 7

- **Palabra vacía** (**λ**): |λ|=0

- **Universo** de un alfabeto (**W(∑)**): Todas las palabras que se pueden formar con símbolos de ∑.

DEFINICIONES BÁSICAS

- Ejercicio:

Sobre $\sum_3 = \{z,l,m,cd,hv\}$

$x=zzlm \rightarrow |x|= ?$

$x'=cdhvzz \rightarrow |x'|= ?$

DEFINICIONES BÁSICAS

- Ejercicio:

Sobre $\sum_3 = \{z,l,m,cd,hv\}$

$x=zzlm \rightarrow |x|= 4$

$x'=cdhvzz \rightarrow |x'|= 4$

DEFINICIONES BÁSICAS

- **Longitud** de una palabra (**|x|**) : número de **símbolos** del **alfabeto** que la forman.

  Sobre $\sum_1$ |branala| = 3

  Sobre Alf. Español |branala|= 7

- **Palabra vacía** (**λ**): |λ|=0

- **Universo** de un alfabeto (**W(∑)**): Todas las palabras que se pueden formar con símbolos de ∑.

DEFINICIONES BÁSICAS

$W(\Sigma_1) = \{\lambda, na, pa, la, bra, napa, nala, ...\}$

- **Lenguaje** sobre un alfabeto **($L(\Sigma)$)**: cualquier subconjunto de $W(\Sigma)$.

$L_1(\Sigma_1) = \{nana, napa, lana\}$

$L_2(\Sigma_1) = \{\lambda, pana, palabra, pala\}$

OPERACIONES CON PALABRAS

- ***x*** e ***y*** son palabras, la **concatenación** x · y, es:

palabra formada por los **símbolos** de *x* seguidos de los símbolos de *y*

En $\sum_1$ si *x* = pa e *y* = labra, *x* · *y* = palabra

- La **potencia** i-ésima de *x* → **Concatenar** *i* veces *x*

$\sum_2$ ={0, 1} y x=10 → x^5=1010101010

¿Qué pasa con x^0?

OPERACIONES CON PALABRAS

- ***x*** e ***y*** son palabras, la **concatenación** x · y, es:

palabra formada por los **símbolos** de *x* seguidos de los símbolos de *y*

En $\sum_1$ si *x* = pa e *y* = labra, *x* · *y* = palabra

- La **potencia** i-ésima de *x* → **Concatenar** *i* veces *x*

$\sum_2$ ={0, 1} y x=10 → x^5=1010101010

¿Qué pasa con x^0? → $x^0 = \lambda$

OPERACIONES CON PALABRAS

- Ejercicio:

Sobre $\sum_3 = \{z, l, m, cd, hv\}$

$x=cdz$, $y=hvm \rightarrow xy=$?

Sobre $\sum_3$ $w=zlcd \rightarrow w^0=$?

$|w^1|=$?

$w^2=$?

$w^3=$?

OPERACIONES CON PALABRAS

- Ejercicio:

Sobre $\sum_3 = \{z, l, m, cd, hv\}$

$x=cdz$, $y=hvm \rightarrow xy= cdzhvm$

Sobre $\sum_3$ $w=zlcd \rightarrow w^0= \lambda$

$|w^1|= |zlcd| = 3$

$w^2= zlcdzlcd$

$w^3= zlcdzlcdzlcd$

OPERACIONES CON PALABRAS

- **<u>Reflexión</u>:** si $x=A_1A_2...A_n$ entonces la **palabra inversa** de x es: $x^{-1}=A_n...A_2A_1$

$$\text{En } \Sigma_1 \text{ si } x = \text{pala} \rightarrow x^{-1} = ?$$

OPERACIONES CON PALABRAS

- **<u>Reflexión</u>:** si $x=A_1A_2...A_n$ entonces la **palabra inversa** de x es: $x^{-1}=A_n...A_2A_1$

$$\text{En } \Sigma_1 \text{ si } x = \text{pala} \rightarrow x^{-1} = \text{lapa}$$

Sobre $\Sigma_3 = \{z, l, m, cd, hv\}$

$x = zllm \rightarrow x^{-1} = ?$

$x' = cdlz \rightarrow x'^{-1} = ?$

OPERACIONES CON PALABRAS

- **<u>Reflexión</u>:** si $x=A_1A_2...A_n$ entonces la **palabra inversa** de x es: $x^{-1}=A_n...A_2A_1$

En $\sum_1$ si x = pala $\rightarrow$ x^{-1} = lapa

Sobre $\sum_3$ = {z, l, m, cd, hv}

x= zllm $\rightarrow$ x^{-1} = mllz

x'= cdlz $\rightarrow$ x'^{-1} = zlcd

OPERACIONES CON LENGUAJES

- La **unión** de dos lenguajes contiene las palabras que pertenezcan a **cualquiera** de los dos lenguajes (O).

$L_1(\Sigma_1)$ = {nana, napa, lana}

$L_2(\Sigma_1)$ = {λ, nana, pana, palabra, pala}

$L_1(\Sigma_1)$ **U** $L_2(\Sigma_1)$= {λ, nana, pana, palabra, pala, napa, lana }

¿Elementos repetidos? ¿Orden?

OPERACIONES CON LENGUAJES

- La **unión** de dos lenguajes contiene las palabras que pertenezcan a **cualquiera** de los dos lenguajes (O).

$L_1(\Sigma_1)$ = {nana, napa, lana}

$L_2(\Sigma_1)$ = {λ, nana, pana, palabra, pala}

$L_1(\Sigma_1)$ **U** $L_2(\Sigma_1)$= {λ, napa, pana, palabra, pala, nana, lana }

¿Elementos repetidos? ¿Orden?

OPERACIONES CON LENGUAJES

- Ejercicio:

$\sum_1 = \{1, 2, 3, 4, 5, 6, 7, 8\}$

$\sum_2 = \{a, b, c, d, e, f, g, h\}$

$L_1(\sum_1) = \{1, 2, 3, 4, 5, 6, 7, 8\}$

$L_2(\sum_2) = \{a, b, c, d, e, f, g, h\}$

¿Esta coincidencia es posible?

?

OPERACIONES CON LENGUAJES

- Ejercicio:

$\sum_1 = \{1, 2, 3, 4, 5, 6, 7, 8\}$

$\sum_2 = \{a, b, c, d, e, f, g, h\}$

$L_1(\sum_1) = \{1, 2, 3, 4, 5, 6, 7, 8\}$

$L_2(\sum_2) = \{a, b, c, d, e, f, g, h\}$

¿Esta coincidencia es posible?

- Si porque son los lenguajes con palabras de longitud 1

OPERACIONES CON LENGUAJES

- Ejercicio:

$L_1(\sum_1) = \{1, 2, 3, 4, 5, 6, 7, 8\}$

$L_2(\sum_2) = \{a, b, c, d, e, f, g, h\}$

$L_1 \cup L_2 = ?$

OPERACIONES CON LENGUAJES

- Ejercicio:

$L_1(\sum_1) = \{1, 2, 3, 4, 5, 6, 7, 8\}$

$L_2(\sum_2) = \{a, b, c, d, e, f, g, h\}$

$L_1 \cup L_2 = \{1, 2, 3, 4, 5, 6, 7, 8, a, b, c, d, e, f, g, h\}$

$L_1 \cup L_2 = \{1, b, 3, a, 5, 6, 7, 8, 4, 2, c, d, e, f, g, h\}$

OPERACIONES CON LENGUAJES

- La **intersección** de dos lenguajes contiene las palabras que pertenezcan a los dos lenguajes (Y).

$$L_1(\Sigma_1) = \{nana, napa, lana\}$$

$$L_2(\Sigma_1) = \{\lambda, nana, pana, palabra, pala\}$$

$$L_1(\Sigma_1) \cap L_2(\Sigma_1) = \{nana\}$$

OPERACIONES CON LENGUAJES

- Ejercicio:

$L_1(\sum_1) = \{1, 2, 3, 4, 5, 6, 7, 8\}$

$L_2(\sum_2) = \{a, b, c, d, e, f, g, h\}$

$L_1 \cap L_2 = ?$

OPERACIONES CON LENGUAJES

- Ejercicio:

$L_1(\sum_1) = \{1, 2, 3, 4, 5, 6, 7, 8\}$

$L_2(\sum_2) = \{a, b, c, d, e, f, g, h\}$

$L_1 \cap L_2 = \varnothing$

OPERACIONES CON LENGUAJES

- La **resta** de $L_1(\Sigma_1)$ y $L_2(\Sigma_1)$ contiene las palabras que pertenecen a L_1 y no pertenecen a L_2.

$L_1(\Sigma_1)$ = {nana, napa, lana}

$L_2(\Sigma_1)$ = {λ, nana, pana, palabra, pala}

$L_1(\Sigma_1) - L_2(\Sigma_1)$= {napa, lana}

$L_2(\Sigma_1) - L_1(\Sigma_1)$= {λ, pana, palabra, pala}

OPERACIONES CON LENGUAJES

- La **concatenación** contiene las palabras formadas por la **concatenación de palabras** de L_1 y de L_2.

$$L_1(\Sigma_1) = \{nana, napa, lana\}$$

$$L_2(\Sigma_1) = \{\lambda, nana, pana, palabra, pala\}$$

$$L_1(\Sigma_1) \cdot L_2(\Sigma_1) = \{nana, napa, lana, nananana, nanapana, nanapalabra, ...\}$$

¿Por qué aparece ”napa” que sólo pertenece a L_1?

OPERACIONES CON LENGUAJES

- La **concatenación** contiene las palabras formadas por la **concatenación de palabras** de L_1 y de L_2.

$L_1(\Sigma_1)$ = {nana, napa, lana}

$L_2(\Sigma_1)$ = {λ, nana, pana, palabra, pala}

$L_1(\Sigma_1) \cdot L_2(\Sigma_1) = L_1(\Sigma_1)L_2(\Sigma_1)$ = {nana, napa, lana, nananana, nanapana, nanapalabra, ...}

¿Por qué aparece ”napa” que sólo pertenece a L_1?

Porque: $napa \cdot \lambda = napa$

OPERACIONES CON LENGUAJES

- Ejercicio:

$\sum_3 = \{z, l, m, cd, hv\}$

$L_1 = \{zl\}$

$L_2 = \{cd, m\}$

$L = L_1 \cdot L_2 = ?$

$L = L_2 \cdot L_1 = ?$

OPERACIONES CON LENGUAJES

- Ejercicio:

$\sum_3 = \{z, l, m, cd, hv\}$

$L_1 = \{zl\}$

$L_2 = \{cd, m\}$

$L = L_1 \cdot L_2 = \{zlcd, zlm\}$

$L = L_2 \cdot L_1 = \{cdzl, mzl\}$

OPERACIONES CON LENGUAJES

- La **potencia** i-ésima de un **lenguaje** es la concatenación *i* veces del lenguaje con **él mismo**.

$$L_1(\Sigma_2) = \{0, 1\}$$

$$L^2{}_1(\Sigma_2) = \{00, 01, 10, 11\}$$

¿Qué pasa con $L^0{}_1$?

OPERACIONES CON LENGUAJES

- La **potencia** i-ésima de un **lenguaje** es la concatenación *i* veces del lenguaje con **él mismo**.

$$L_1(\Sigma_2) = \{0, 1\}$$

$$L^2_1(\Sigma_2) = \{00, 01, 10, 11\}$$

¿Qué pasa con L^0_1?

$$L^0_1 = \{\lambda\}$$

OPERACIONES CON LENGUAJES

- Ejercicio:

$L = \{cd\}$

$L^0 = ?$

$L^1 = ?$

$L^2 = ?$

$L^3 = ?$

OPERACIONES CON LENGUAJES

- Ejercicio:

$L = \{cd\}$

$L^0 = \{\lambda\}$

$L^1 = \{cd\}$

$L^2 = \{cdcd\}$

$L^3 = \{cdcdcd\}$

OPERACIONES CON LENGUAJES

- Ejercicio:

$L = \{cdz, l\}$

$L^0 = ?$

$L^1 = ?$

$L^2 = ?$

$L^3 = ?$

OPERACIONES CON LENGUAJES

- Ejercicio:

$L = \{cdz, l\}$

$L^0 = \{\lambda\}$

$L^1 = \{cdz, l\}$

$L^2 = \{cdzcdz, cdzl, lcdz, ll\}$

$L^3 = \{cdzcdzcdz, cdzcdzl, cdzlcdz, cdzll, lcdzcdz, lcdzl, llcdz, lll\}$

OPERACIONES CON LENGUAJES

- La **clausura positiva** de un lenguaje **L** es la unión de todas sus **potencias** (**salvo** la **cero**)

$$L^{+} = U_{i=1}^{\infty} L^{i}$$

$$\Sigma^{+} = W(\Sigma) - \{\lambda\}$$

$L_1(\Sigma_2) = \{0, 1\}$

$L_1(\Sigma_2)^{+} = \{0, 1, 00, 01, 10, 11, 000, 001, ...\}$

OPERACIONES CON LENGUAJES

- Ejercicio:

$L_1(\Sigma_4) = \{m, a, b\}$

$L_1^+ = ?$

OPERACIONES CON LENGUAJES

- Ejercicio:

$L_1(\sum_4)$ = {m, a, b}

L_1^+= { m, a, b, mm, ma, mb, am, aa, ab, bm, ba, bb, mmm, mma, mmb, mam, maa, mab, mbm, mba, mbb, amm, ...}

OPERACIONES CON LENGUAJES

- El **cierre o clausura** de un lenguaje **L** es la unión de la **cadena vacía** a la **clausura positiva** del lenguaje.

$$L^* = L^+ \cup \{\lambda\} = U_{i=0}^{\infty} L^i$$

$$\Sigma^* = W(\Sigma)$$

$L_1(\Sigma_2) = \{0, 1\}$

$L_1(\Sigma_2)^* = \{\lambda, 0, 1, 00, 01, 10, 11, 000, 001, \ldots\}$

OPERACIONES CON LENGUAJES

- Ejercicio:

$L_1(\sum_4) = \{m, a, b\}$

$L_1^* = ?$

OPERACIONES CON LENGUAJES

- Ejercicio:

$L_1(\sum_4) = \{m, a, b\}$

$L_1^* = \{\lambda,\ m, a, b, mm, ma, mb, am, aa, ab, bm, ba, bb, mmm,$
$mma, mmb, mam, maa, mab, mbm, mba, mbb, amm, \ldots\}$

OPERACIONES CON LENGUAJES

- La **reflexión** de un lenguaje es la reflexión a cada una de las palabras.

$$L^{-1} = \{ x^{-1} \mid x \in L \}$$

$L_1(\Sigma_2) = \{0, 1, 00, 10\}$

$L_1(\Sigma_2)^{-1} = \{0, 1, 00, 01\}$

OPERACIONES CON LENGUAJES

- Ejercicio:

$\sum_5$ = {ma, pa, z, l}

$L_1(\sum_5)$ = {mal, pal, zma, paz}

L_1^{-1}= ?

OPERACIONES CON LENGUAJES

- Ejercicio:

Sea w=cca sobre el alfabeto $\sum$={a,b,c} indica el resultado de las siguientes operaciones:

a) w^0 =

b) w^{-1} =

c) $|w^3|$ =

OPERACIONES CON LENGUAJES

- Ejercicio:

Sea w=cca sobre el alfabeto $\sum=\{a,b,c\}$ indica el resultado de las siguientes operaciones:

a) $w^0 = \lambda$

b) $w^{-1} = acc$

c) $|w^3| = 9$

OPERACIONES CON LENGUAJES

- Ejercicio:

Sea $L_1=\{a, b, x\}$ y $L_2=\{ab\}$ calcula:

a) $L_1 - L_2$

b) $L_2 - L_1$

c) $L_1^2 - L_2^3$

OPERACIONES CON LENGUAJES

- Ejercicio:

Sea $L_1=\{a, b, x\}$ y $L_2=\{ab\}$ calcula:

a) $L_1 - L_2 = \{a, b, x\}$ (No hay elementos comunes)

b) $L_2 - L_1 = \{ab\}$ (No hay elementos comunes)

c) $L_1^2 - L_2^3 = \{aa, ab, ax, ba, bb, bx, xa, xb, xx\}$

$L_1^2 = \{aa, ab, ax, ba, bb, bx, xa, xb, xx\}$

$L_2^3 = \{ababab\}$

OPERACIONES CON LENGUAJES

- Ejercicio:

Sea Σ= {a,b,c,d} diseña un lenguaje que contenga todas las cadenas que empiecen en a y acaben en b.

L=

OPERACIONES CON LENGUAJES

- Ejercicio:

Sea $\Sigma = \{a,b,c,d\}$ diseña un lenguaje que contenga todas las cadenas que empiecen en a y acaben en b.

$$L = \{a\} \cdot \Sigma^* \cdot \{b\}$$

OPERACIONES CON LENGUAJES

- Ejercicio:

Sea $\Sigma = \{0,1,2\}$ diseña un lenguaje que contenga todas las cadenas de longitud 3 que sean múltiplos de 2, salvo la cadena 210 y la 222.

OPERACIONES CON LENGUAJES

- Ejercicio:

Sea $\Sigma = \{0,1,2\}$ diseña un lenguaje que contenga todas las cadenas de longitud 3 que sean múltiplos de 2, salvo la cadena 210 y la 222.

$$L = (\Sigma^2 \cdot \{0, 2\}) - \{210, 222\}$$

EJERCICIOS

Matrículas

La Dirección General de Tráfico desea construir un **sistema** que sea **capaz** de **determinar** si una **secuencia** de **símbolos** forma una **matrícula** española **válida** (antes del 2000). Se pide **diseñar el lenguaje** que servirá de base para dicho sistema.

Matrículas

L_M

$L_{A1} \rightarrow$ PROVINCIA – 6 DIGITOS (M-657261)

$L_{A2} \rightarrow$ PROVINCIA – 5 DIGITOS (CR-57261)

$L_{N1} \rightarrow$ PROV. – 4 DIGITOS – LETRA (CR-7635-A)

$L_{N2} \rightarrow$ PROV. – 4 DIGITOS – LETRAS (J-1249-AB)

¡Construir el lenguaje en función de otros más simples!

Matrículas

$$L_p = \{A, AB, AL, ..., ZA\}$$

$$L_D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

$$L_G = \{-\}$$

$$L_L = \{A, B, C, ..., Z\} - \{Ñ, Q, R\}$$

En L_L no se incluye (Ñ, Q, R)

Matrículas

$L_{A1} \rightarrow$ PROVINCIA – 6 DIGITOS (M-657261)

$$L_{A1} = L_P \cdot L_G \cdot L^6_D$$

$L_{A2} \rightarrow$ PROVINCIA – 5 DIGITOS (CR-65726)

$$L_{A2} = L_P \cdot L_G \cdot L^5_D$$

$L_A = L_{A1}\ U\ L_{A2} \rightarrow$ 5 y 6 dígitos

$L_A = ?$

Matrículas

$$L_{A1} \rightarrow \text{PROVINCIA} - 6 \text{ DIGITOS (M-657261)}$$

$$L_{A1} = L_P \cdot L_G \cdot L^6_D$$

$$L_{A1} \rightarrow \text{PROVINCIA} - 5 \text{ DIGITOS (CR-65726)}$$

$$L_{A2} = L_P \cdot L_G \cdot L^5_D$$

$$L_A = L_{A1} \cup L_{A2} \rightarrow 5 \text{ y } 6 \text{ dígitos}$$

$$L_A = L_P \cdot L_G \cdot (L^5_D \cup L^6_D)$$

Matrículas

$L_{N1} \rightarrow$ PROV. – 4 DIGITOS – LETRA (CR-7635-A)

$$L_{N1} = L_P \cdot L_G \cdot L^4_D \cdot L_G \cdot L_L$$

$L_{N2} \rightarrow$ PROV. – 4 DIGITOS – LETRAS (CR-7635-AB)

$$L_{N2} = L_P \cdot L_G \cdot L^4_D \cdot L_G \cdot L^2_L$$

$L_N = ?$

Matrículas

$L_{N1} \rightarrow$ PROV. – 4 DIGITOS – LETRA (CR-7635-A)

$$L_{N2} = L_P \cdot L_G \cdot L^4_D \cdot L_G \cdot L_L$$

$L_{N2} \rightarrow$ PROV. – 4 DIGITOS – LETRAS (CR-7635-AB)

$$L_{N2} = L_P \cdot L_G \cdot L^4_D \cdot L_G \cdot L^2_L$$

$$L_N = L_P \cdot L_G \cdot L^4_D \cdot L_G \cdot (L_L \cup L^2_L)$$

Matrículas

¿$L_N \rightarrow$ PROV. – 4 DIGITOS – LETRAS (CR-7635-AB)?

$$L_N = L_P \cdot L_G \cdot L^4_D \cdot L_G \cdot (L_L \, U \, L^2_L)$$

Sería válido si no hubiese combinaciones de letras no permitidas (BA, CA, ...)

$$L_F = \{BA, CA, DA, ...\}$$

$$L_M = L_A \, U \, (L_N - L_F)$$

Ejercicio 1

- Definir sobre el **alfabeto** $\sum=\{a,b\}$ el lenguaje formado por todas las palabras que **comienzan** por **“a”**.

L = ?

Ejercicio 1

- Definir sobre el **alfabeto** $\Sigma=\{a,b\}$ el lenguaje formado por todas las palabras que **comienzan** por **"a"**.

$$L = (\Sigma - \{b\}) \cdot \Sigma^*$$

$$L = \{a\} \cdot \Sigma^*$$

¿Por qué no Σ^+?

Ejercicio 2

- Definir sobre el **alfabeto** $\sum=\{a,b\}$ el lenguaje formado por todas las palabras que **contienen** la subcadena **“bb”**.

L =?

Ejercicio 2

- Definir sobre el **alfabeto** $\sum=\{a,b\}$ el lenguaje formado por todas las palabras que **contienen** la subcadena **“bb”**.

$$L = \sum^* \cdot \{bb\} \cdot \sum^*$$

Ejercicio 5

- Definir el lenguaje formado por todos los **números naturales**, **evitar** los números que contienen **ceros** a la **izquierda**, p.e. 000020, 02, 0000,...

L =?

Ejercicio 5

- Definir el lenguaje formado por todos los **números naturales**, **evitar** los números que contienen **ceros** a la **izquierda**, p.e. 000020, 02, 0000,...

$$L_Z = \{0\}$$

$$L_D = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

$$L = L_D \cdot (L_D \cup L_Z)^*$$

¿Por qué no clausura positiva?

Ejercicio 6

- Definir el **lenguaje** formado por todos los números **enteros**, **evitar** los números que contienen **ceros** a la **izquierda**, p.e. -000002, -002, 0002, 0000120,...

L =?

Ejercicio 6

- Definir el **lenguaje** formado por todos los números **enteros**, **evitar** los números que contienen **ceros** a la **izquierda**, p.e. -000002, -002, 0002, 0000120,...

$$L_Z = \{0\}$$

$$L_G = \{\lambda, -\}$$

$$L_D = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

$$L = L_G \cdot L_D \cdot (L_D \cup L_Z)^*$$

¿Y el 0?

Ejercicio 6

- Definir el **lenguaje** formado por todos los números **enteros**, **evitar** los números que contienen **ceros** a la **izquierda**, p.e. -000002, -002, 0002, 0000120,...

$$L_Z = \{0\}$$

$$L_G = \{\lambda, -\}$$

$$L_D = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

$$L = L_G \cdot ((L_D \cdot (L_D \cup L_Z)^*) \cup L_Z)$$

¿Y el 0?

Ejercicio 7

- Definir el conjunto de los números **divisibles** por **5**.

L =?

Ejercicio 7

- Definir el conjunto de los números **divisibles** por **5**.

$$L_D = \{1, 2, 3, 4, 6, 7, 8, 9\}$$

$$L_M = \{0, 5\}$$

$$L = (L_D \cup L_M)^* \cdot L_M$$

Ejercicio 13, 17 y 22

13. Definir el lenguaje formado por todas aquellas direcciones de correo electrónico bien formadas de los alumnos de la UCLM.

17. Definir el lenguaje formado sobre el alfabeto $\Sigma=\{a,b,c\}$ por todas aquellas cadenas de longitud 4 y que contienen exactamente una "b".

22. Definir el lenguaje sobre el alfabeto $\Sigma=\{0,1\}$ formado por todas las cadenas que tienen longitud par.

Ejercicio 13

$$L_D = \{@alu.uclm.es\}$$

$$L_H = \{0,1,..., 9\}$$

$$L_N = \{a,b,...,z\}$$

$$L_P = \{.\}$$

$$L = L_N^+ \cdot L_P \cdot L_N^+ \cdot (\{\lambda\} \cup L^1_H \cup L^2_H) \cdot L_D$$

Ejercicio 13

$$L_D = \{@alu.uclm.es\}$$

$$L_H = \{\lambda, 0,1,..., 9\}$$

$$L_N = \{a,b,...,z\}$$

$$L_P = \{.\}$$

$$L = L_N^+ \cdot L_P \cdot L_N^+ \cdot (L^1_H \cup L^2_H) \cdot L_D$$

Ejercicio 17

$$\sum=\{a,b,c\}$$

$$L = (\{b\} \cdot (\sum-\{b\})^3) \cup$$

$$(\sum-\{b\} \cdot \{b\} \cdot (\sum-\{b\})^2) \cup$$

$$((\sum-\{b\})^2 \cdot \{b\} \cdot \sum-\{b\}) \cup$$

$$((\sum-\{b\})^3) \cdot \{b\})$$

Ejercicio 22

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

$$L = (\sum^2)^n$$

1. Definir el lenguaje formado sobre el alfabeto ∑={a,b,c} por todas aquellas cadenas de longitud 4 y que contienen exactamente una "b". Utiliza el operador reflexión para alcanzar una solución.
2. Lenguaje formado por aquellas direcciones de correo electrónico de profesores. Juan.Ramirez@uclm.es Juan.LRamirez@uclm.es.
3. Sea el alfabeto ∑={0, 1, 2} define el lenguaje de cadenas en las que el primer y el último símbolo son diferentes.
4. Sea el alfabeto ∑={a, b, c, d} define el lenguaje de cadenas de longitud cero o mayor o igual a 3. No se permite en el inicio de las cadenas que aparezcan combinaciones de tres símbolos iguales. p.e. aaac
5. Define el lenguaje para generar códigos de un producto de la forma: números-letras. Donde números puede tener una longitud de 5 a 7 y letras comienza por una A seguida de dos consonantes en minúscula. Letras puede ser un campo vacío.

EJERCICIO

- Especifica un lenguaje para representar cuatro digitos sin ningún 5 y sin combinaciones seguidas de valores ”12” y ”23”.

EJERCICIO

- Especifica un lenguaje para representar cuatro digitos sin ningún 5 y sin combinaciones seguidas de valores ”12” y ”23”.

$$L_D = \{0, 1, \ldots, 9\} - \{5\}$$

$$L_E = ((L_D^2 - \{12, 23\}) \cdot L_D^2) \cap (L_D \cdot (L_D^2 - \{12, 23\}) \cdot L_D) \cap (L_D^2 \cdot (L_D^2 - \{12, 23\}))$$

TEMA 3

EXPRESIONES REGULARES

CONTENIDOS

- Contexto
- Definiciones
- Precedencia de operadores
- Propiedades Algebraicas
- Expresiones regulares extendidas
- Conclusiones

CONTEXTO

Gramáticas	Lenguajes	Máquinas
Sin restricciones o de Tipo 0	Sin restricciones o de Tipo 0	Máquina de Turing
Sensible al contexto o de Tipo 1	Sensible al contexto o de Tipo 1	Autómata linealmente acotado
Libre de contexto o de Tipo 2	Libre de contexto o de Tipo 2	Autómata a pila
Regular o de Tipo 3	**Regular o de Tipo 3**	**Autómata Finito**

CONTEXTO

Las **Expresiones Regulares(ER)** son fórmulas que representan de manera **concisa** Lenguajes Regulares.

~~Descripción Exhaustiva~~ → Resumen

CONTEXTO

Las **Expresiones Regulares** definen lenguajes aceptados por un **Autómata Finito**.

A partir de un **Autómata Finito** que acepta un **Lenguaje Regular**, éste se puede extraer generando su **Expresión Regular**.

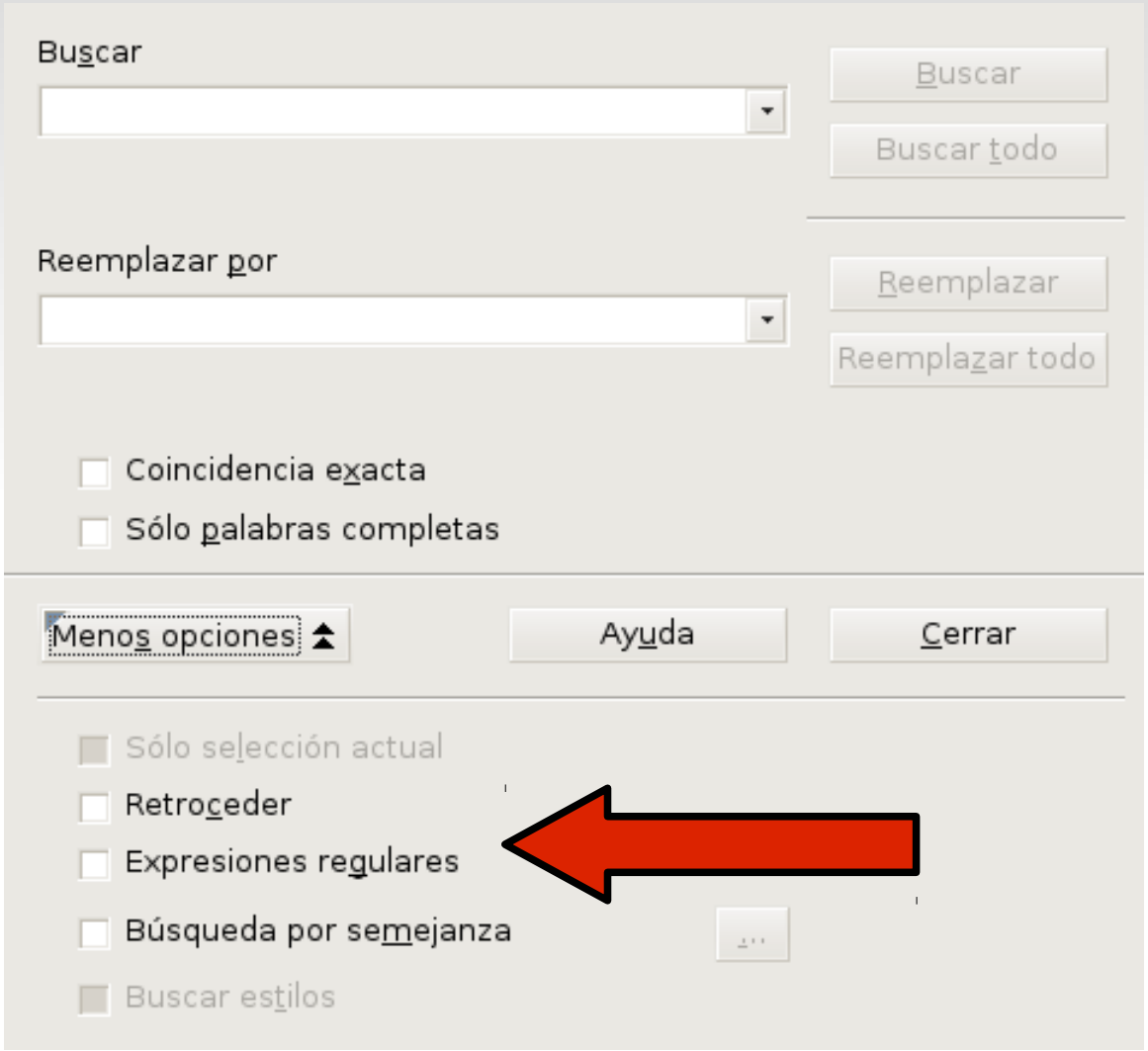
CONTEXTO

Además se **emplean** en:

- **Lenguajes** de programación
- **Utilidades** de programación
- **Software** en general

para describir **filtros + cadenas válidas.**

CONTEXTO


Buscar
Buscar
Buscar todo
Reemplazar por
Reemplazar
Reemplazar todo
Coincidencia exacta
Sólo palabras completas
Menos opciones
Ayuda
Cerrar
Sólo selección actual
Retroceder
Expresiones regulares
Búsqueda por semejanza
Buscar estilos

DEFINICIONES

Expresiones Regulares sobre un alfabeto $\sum$

- Ø es una ER.
- λ es una ER.
- Cada símbolo de $\sum$ es una ER.
- α y β son ER → α+β es una ER
- α y β son ER → α·β (αβ) es una ER
- α es una ER → α* es una ER
- α es una ER → (α) es una ER

DEFINICIONES

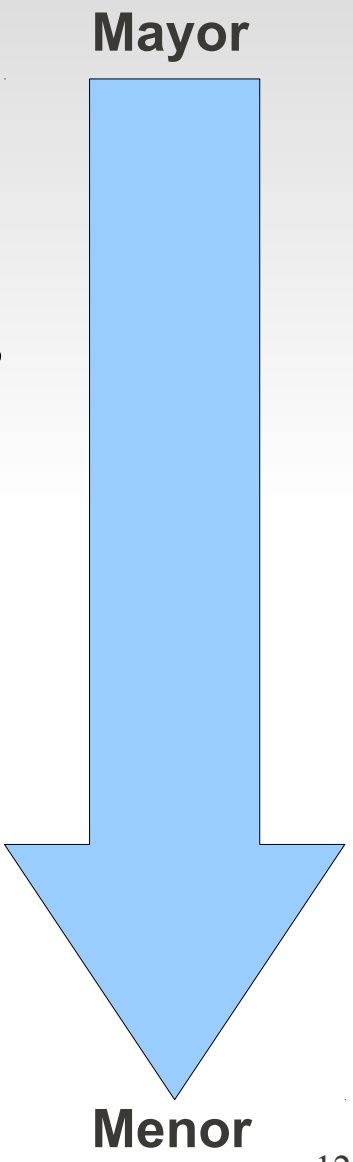
Lenguaje representado por una ER

Toda ER α **define** recursivamente un **lenguaje** L mediante las siguientes **reglas**:

- $\alpha = \emptyset \rightarrow L(\alpha) = \emptyset$
- $\alpha = \lambda \rightarrow L(\alpha) = \{\lambda\}$
- $\alpha = a$ y $a \in \sum \rightarrow L(\alpha) = \{a\}$
- α y β son ER → $L(\alpha+\beta) = L(\alpha) \cup L(\beta)$
- α y β son ER → $L(\alpha \cdot \beta) = L(\alpha) \cdot L(\beta)$
- α es ER → $L(\alpha^*) = L(\alpha)^*$
- α es ER → $L((\alpha)) = L(\alpha)$

PRECEDENCIA OPERADORES

- **Paréntesis**
- **Clausura** → aplicada a la ER más pequeña que quede a su izquierda
- **Concatenación**
- **Unión** (+)



PRECEDENCIA OPERADORES

- **Sea la expresión 01*+1, ¿Qué lenguaje representa?**

$1^* \rightarrow \{\lambda, 1, 11, 111, \ldots\}$

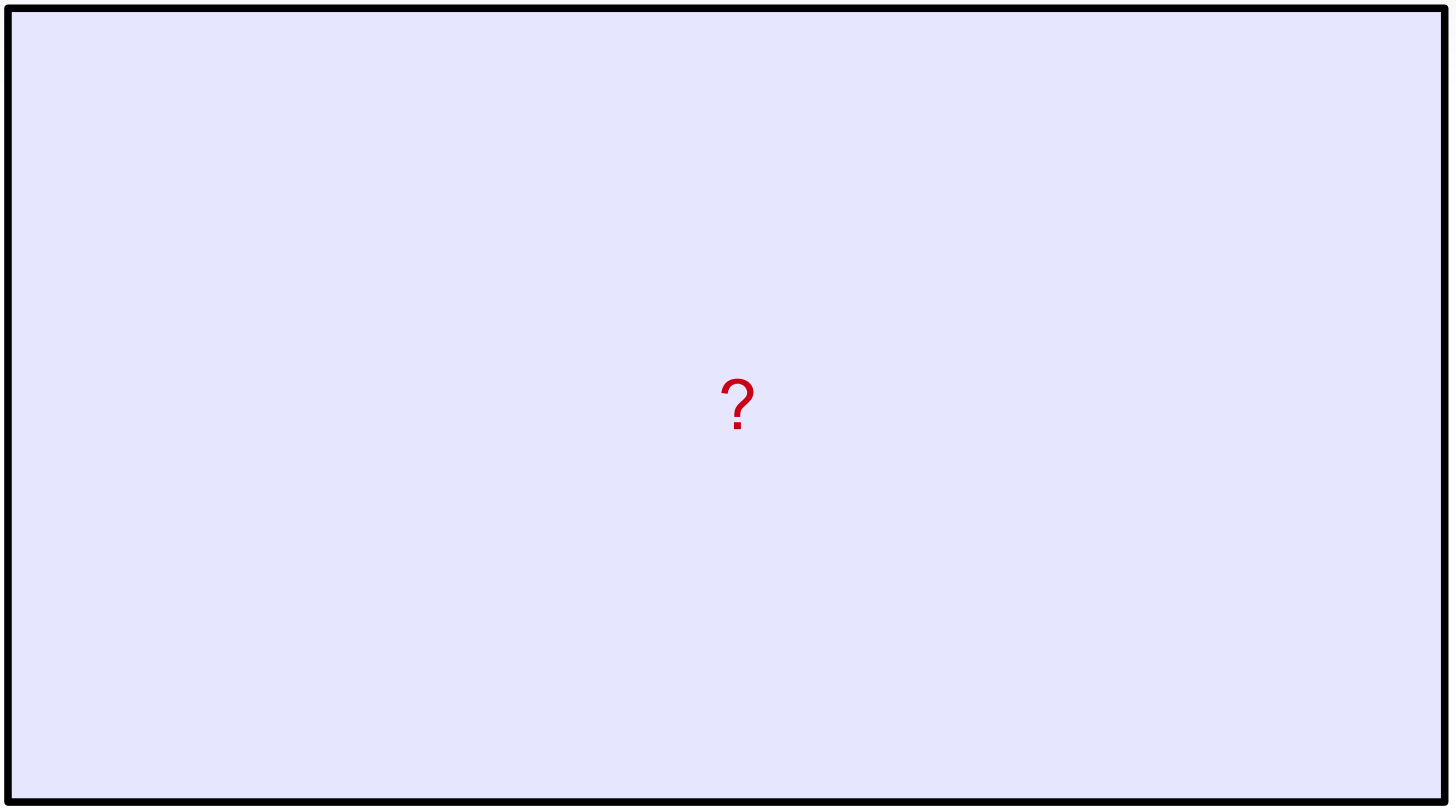
$01^* \rightarrow \{0, 01. 011, 0111, ..\}$

$01^* + 1 \rightarrow \{0, 1, 01. 011, 0111, ..\}$

”La cadena 1 más todas las cadenas que comienzan por cero y están seguidas por cualquier número de unos”

PRECEDENCIA OPERADORES

- **Sea la expresión (0+1)*+1, ¿Qué lenguaje representa?**



PRECEDENCIA OPERADORES

- **Sea la expresión (0+1)*+1, ¿Qué lenguaje representa?**

$(0+1) \rightarrow \{0, 1\}$

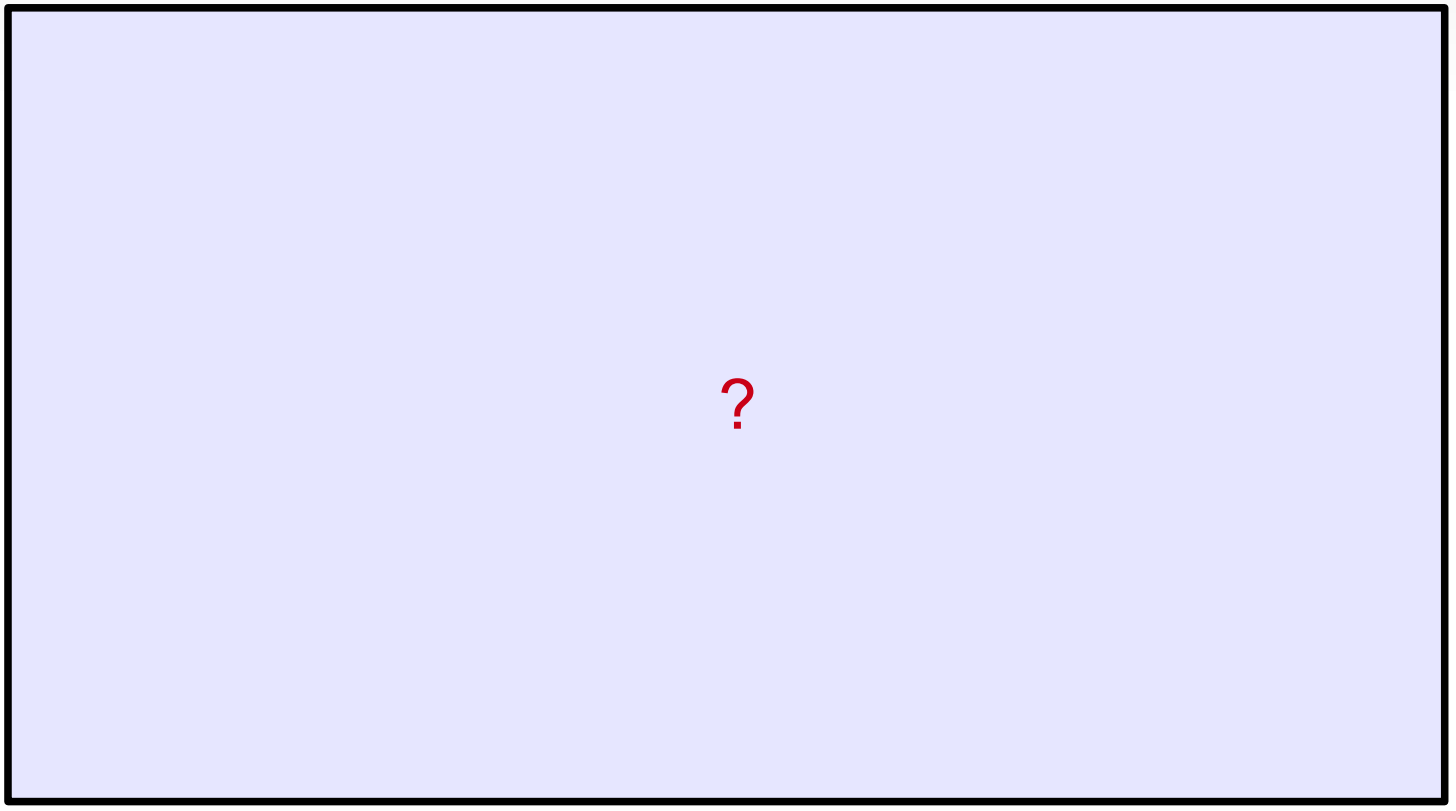
$(0+1)^* \rightarrow \{\lambda, 0, 1, 01, 10, 11, 0101, 00000, \ldots\}$

$(0+1)^* + 1 \rightarrow \{\lambda, 0, 1, 01, 10, 11, 0101, 00000, \ldots\}$

”Cadena de longitud mayor o igual a cero que contienen cualquier combinación ceros o unos”

PRECEDENCIA OPERADORES

- **Sea la expresión 0+(1*+1), ¿Qué lenguaje representa?**



PRECEDENCIA OPERADORES

- **Sea la expresión 0+(1*+1), ¿Qué lenguaje representa?**

$1^* \rightarrow \{\lambda, 1, 11, 111, \ldots\}$

$(1^* + 1) \rightarrow \{\lambda, 1, 11, 111, \ldots\}$

$0+(1^* + 1) \rightarrow \{\lambda, 0, 1, 11, 111, \ldots\}$

"La cadena cero y cadenas de longitud mayor o igual que cero que contienen solamente unos"

PROPIEDADES ALGEBRAICAS

- Tema 2 → **Más** de un resultado para **generar** un **mismo** lenguaje.
- Con las ER sucede igual:

$$\alpha \text{ y } \beta \text{ son equivalentes si } L(\alpha) = L(\beta)$$

Las **propiedades algebraicas** van a **permitir** la **simplificación** de ER para obtener la **solución** más **sencilla**.

EJERCICIO 1

- Dado el alfabeto $\sum = \{0,1\}$, definir las expresiones regulares para los siguientes lenguajes:

EJERCICIO 1(a)

- Dado el alfabeto $\sum = \{0,1\}$, definir las expresiones regulares para los siguientes lenguajes:

Lenguaje de todas las cadenas de longitud par.

?

EJERCICIO 1(a)

- Dado el alfabeto $\sum = \{0,1\}$, definir las expresiones regulares para los siguientes lenguajes:

Lenguaje de todas las cadenas de longitud par.

(00 + 01 + 10 + 11)*

EJERCICIO 1(a)

- Dado el alfabeto $\sum = \{0,1,2\}$, definir las expresiones regulares para los siguientes lenguajes:

Lenguaje de todas las cadenas de longitud par.

?

EJERCICIO 1(a)

- Dado el alfabeto $\sum = \{0,1,2\}$, definir las expresiones regulares para los siguientes lenguajes:

Lenguaje de todas las cadenas de longitud par.

(00 + 01 +02 + 10 + 11 + 12 + 20 +21 + 22)*

EJERCICIO 1(b)

- Dado el alfabeto $\sum = \{0,1\}$, definir las expresiones regulares para los siguientes lenguajes:

Lenguaje de todas las cadenas de longitud impar.

?

EJERCICIO 1(b)

- Dado el alfabeto $\sum = \{0,1\}$, definir las expresiones regulares para los siguientes lenguajes:

Lenguaje de todas las cadenas de longitud impar.

$(00 + 01 + 10 + 11)^* (0 + 1)$

EJERCICIO 1(b)

- Dado el alfabeto $\sum = \{0,1,2\}$, definir las expresiones regulares para los siguientes lenguajes:

Lenguaje de todas las cadenas de longitud impar.

?

EJERCICIO 1(b)

- Dado el alfabeto $\sum = \{0,1,2\}$, definir las expresiones regulares para los siguientes lenguajes:

Lenguaje de todas las cadenas de longitud impar.

$((0+1+2)(0+1+2))^*(0+1+2)$

EJERCICIO 1(c)

- Dado el alfabeto $\sum = \{0,1\}$, definir las expresiones regulares para los siguientes lenguajes:

Lenguaje de todas las cadenas que tienen al menos un 1.

?

EJERCICIO 1(c)

- Dado el alfabeto $\sum = \{0,1\}$, definir las expresiones regulares para los siguientes lenguajes:

Lenguaje de todas las cadenas que tienen al menos un 1.

$(0 + 1)^* \; 1(0 + 1)^*$

EJERCICIO 1(c)

- Dado el alfabeto $\sum = \{0,1,2\}$, definir las expresiones regulares para los siguientes lenguajes:

Lenguaje de todas las cadenas que tienen al menos un 2.

?

EJERCICIO 1(c)

- Dado el alfabeto $\sum = \{0,1,2\}$, definir las expresiones regulares para los siguientes lenguajes:

Lenguaje de todas las cadenas que tienen al menos un 2.

(0+1+2)* 2 (0+1+2)*

EJERCICIO 1(d)

- Dado el alfabeto $\sum = \{0,1\}$, definir las expresiones regulares para los siguientes lenguajes:

Lenguaje de todas las cadenas cuya longitud es igual o menor que 4.

?

EJERCICIO 1(d)

- Dado el alfabeto $\sum = \{0,1\}$, definir las expresiones regulares para los siguientes lenguajes:

Lenguaje de todas las cadenas cuya longitud es igual o menor que 4.

$$(0 + 1 + \lambda)(0 + 1 + \lambda)(0 + 1 + \lambda)(0 + 1 + \lambda)$$

EJERCICIO 1(d)

- Dado el alfabeto $\sum = \{0,1\}$, definir las expresiones regulares para los siguientes lenguajes:

Lenguaje de todas las cadenas cuya longitud es mayor que 6

?

EJERCICIO 1(d)

- Dado el alfabeto $\sum = \{0,1\}$, definir las expresiones regulares para los siguientes lenguajes:

Lenguaje de todas las cadenas cuya longitud es mayor que 6

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

CUESTIONARIO 1

1. Define un lenguaje con las siguientes características:

   Cadenas formadas por dígitos y vocales minúsculas de tal forma que sean:

- Cadenas de longitud 6 que empiecen por un dígito y terminen por una vocal.
- Cadenas de longitud 3 que no empiecen ni terminen por dígitos y que no contengan las subcadena ae.

EJERCICIO 1(I)

- Dado el alfabeto $\sum = \{0,1,2\}$, definir las expresiones regulares para los siguientes lenguajes:

Lenguaje de todas las cadenas en los que la primera cifra es diferente de la última.

?

EJERCICIO 1(I)

- Dado el alfabeto $\sum = \{0,1,2\}$, definir las expresiones regulares para los siguientes lenguajes:

Lenguaje de todas las cadenas en los que la primera cifra es diferente de la última.

0(0+1)*1 + 0(0+1)*2 + 1(0+1)*0 + 1(0+1)*2 + 2(0+1)*0 + 2(0+1)*1

ER EXTENDIDAS

- $+ \longrightarrow \alpha^{+} \longrightarrow \alpha\alpha^{*}$

Al menos aparece una vez la ER α.

- [ ]

Expresar rangos.

[a-zA-Z]

a+b+c+...z+A+B+...+Z

ER EXTENDIDAS

- .

Cualquier carácter

- ?

$$r? \rightarrow r+\lambda$$

- ^

Cualquier carácter que **no** esté **contenido** en la expresión a la que **precede**

```
[^a-zA-Z]
```

CONCLUSIONES

- Las **ER** se rigen por una serie de **normas** y hay una construcción para cualquier patrón de caracteres.
- **No** existe un **lenguaje estándar** para las ER

DIFERENTES **DIALECTOS**

- Todos los **dialectos** siguen los **mismos principios**.

EJERCICIO 1(f)

- Dado el alfabeto $\sum = \{0,1\}$, definir las expresiones regulares para los siguientes lenguajes:

Lenguaje de todas las cadenas que no contienen la subcadena 110

?

EJERCICIO 1(f)

- Dado el alfabeto $\sum = \{0,1\}$, definir las expresiones regulares para los siguientes lenguajes:

Lenguaje de todas las cadenas que no contienen la subcadena 110

(0+10)*1*

EJERCICIO 1(g)

- Dado el alfabeto $\sum = \{0,1\}$, definir las expresiones regulares para los siguientes lenguajes:

Lenguaje de todas las cadenas en las que la subcadena 00 aparece como mucho dos veces

?

EJERCICIO 1(g)

- Dado el alfabeto $\sum = \{0,1\}$, definir las expresiones regulares para los siguientes lenguajes:

Lenguaje de todas las cadenas en las que la subcadena 00 aparece como mucho dos veces

¿(01)*00(01)*00 ?

EJERCICIO 1(g)

- Dado el alfabeto $\sum = \{0,1\}$, definir las expresiones regulares para los siguientes lenguajes:

Lenguaje de todas las cadenas en las que la subcadena 00 aparece como mucho dos veces

1* + 1*001* + 1*001*001*

EJERCICIO 1(g)

- Dado el alfabeto $\sum = \{0,1\}$, definir las expresiones regulares para los siguientes lenguajes:

Lenguaje de todas las cadenas en las que la subcadena 00 aparece como mucho dos veces

$1^*(00+\lambda)1^*(00+\lambda)1^*$

EJERCICIO 1(g)

- Dado el alfabeto $\sum = \{0,1\}$, definir las expresiones regulares para los siguientes lenguajes:

Lenguaje de todas las cadenas en las que la subcadena 00 aparece como mucho dos veces

$(01+1)^*(00+\lambda)(10+1)^*1^*(00+\lambda)(10+1)^*$

EJERCICIO 2(a)

- Dado el alfabeto $\sum = \{a,b\}$, definir las expresiones regulares para los siguientes lenguajes:

$L = \{0^n 1^m \mid n+m \text{ es impar } \}$

?

EJERCICIO 2(a)

- Dado el alfabeto $\sum = \{a,b\}$, definir las expresiones regulares para los siguientes lenguajes:

$L = \{0^n 1^m \mid n+m \text{ es impar } \}$

$(00)^*(11)^*1 + (00)^*0(11)^*$

¿Qué pasa con las cadenas que terminan en 0?

EJERCICIO 2(b)

- Dado el alfabeto $\sum = \{a,b\}$, definir las expresiones regulares para los siguientes lenguajes:

$$L = \{0^n\ 1^m \mid n \geq 1,\ m \geq 1,\ nm \geq 4\}$$

?

EJERCICIO 2(b)

- Dado el alfabeto $\sum = \{a,b\}$, definir las expresiones regulares para los siguientes lenguajes:

$$L = \{0^n\ 1^m \mid n \geq 1,\ m \geq 1,\ nm \geq 4\}$$

$$00^+\ 11^+ + 0^+\ 1111^+ + 0000^+\ 1^+$$

EJERCICIO 2(e)

- Dado el alfabeto $\sum = \{a,b\}$, definir las expresiones regulares para los siguientes lenguajes:

$$L = \{vwz \mid v, w, z \in \{0, 1\}^*, |w| = 2\}$$

?

EJERCICIO 2(e)

- Dado el alfabeto $\sum = \{a,b\}$, definir las expresiones regulares para los siguientes lenguajes:

$$L = \{vwz \mid v, w, z \in \{0, 1\}^*, |w| = 2\}$$

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

EJERCICIO 2(e)

- Dado el alfabeto $\sum = \{a,b\}$, definir las expresiones regulares para los siguientes lenguajes:

$$L = \{vwv \mid v, w \in \{0, 1, 2\}^*, |v| = 1\}$$

EJERCICIO 2(e)

- Dado el alfabeto $\sum = \{a,b\}$, definir las expresiones regulares para los siguientes lenguajes:

$$L = \{vwv \mid v, w \in \{0, 1, 2\}^*, |v| = 1\}$$

$$0(0 + 1 + 2)^*0 + 1(0 + 1 + 2)^*1 + 2(0 + 1 + 2)^*2$$

EJERCICIO 2(f)

- Dado el alfabeto $\sum = \{a,b\}$, definir las expresiones regulares para los siguientes lenguajes:

$$L = \{w \in \{0, 1\}^* \;\; 0 = |w| \bmod 4 \}$$

?

EJERCICIO 2(f)

- Dado el alfabeto $\sum = \{a,b\}$, definir las expresiones regulares para los siguientes lenguajes:

$$L = \{w \in \{0, 1\}^*,\ 0 = |w| \bmod 4\ \}$$

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

EJERCICIO 2(g)

- Dado el alfabeto $\sum = \{a,b\}$, definir las expresiones regulares para los siguientes lenguajes:

$L = \{w \in \{0, 1\}^* \mid w$ tiene un número de 1s en $w \equiv 0 \bmod 2 \}$

?

EJERCICIO 2(g)

- Dado el alfabeto $\sum = \{a,b\}$, definir las expresiones regulares para los siguientes lenguajes:

$L = \{w \in \{0, 1\}^* \mid w$ tiene un número de 1s en $w \equiv 0 \bmod 2 \}$

$(0^* 10^* 10^*)^*$

EJERCICIO 2(g)

- Dado el alfabeto $\sum = \{a,b\}$, definir las expresiones regulares para los siguientes lenguajes:

$L = \{w \in \{0, 1\}^* \mid w$ tiene un número de 1s en $w \equiv 0 \bmod 2 \}$

$(0^* 10^* 10^*)^* + 0^*???$

EJERCICIO

- Sea $\sum = \{a,b\}$ define la expresión regular para el lenguaje: $L = \{w \mid |w| \text{ es impar y } |w|>1 \}$:

$$(a + b)\,((a + b)\,(a + b))^*$$

EJERCICIO

- Sea $\sum$ ={a,b} define la expresión regular para el lenguaje: L = {w | |w| es impar y |w|>1 }:

$ab^+ + b^+a$

EJERCICIO

- Sea $\sum = \{a,b\}$ define la expresión regular para el lenguaje: $L = \{w \mid |w| \text{ es impar y } |w|>1 \}$:

$$(a + b)\ ((a+b)\ (a+b))^{+}$$

EJERCICIO

- Sea $\sum = \{a,b\}$ define la expresión regular para el lenguaje: $L = \{w \mid |w| \text{ es par y } |w|>2 \}$

$$(aa + bb + ab + ba)^{+}$$

EJERCICIO

- Sea $\sum = \{a,b\}$ define la expresión regular para el lenguaje: $L = \{w \mid |w| \text{ es par y } |w|>2 \}$

$$(a + b)\,(a + b)^* - (a+b)$$

EJERCICIO

- Sea $\sum = \{a,b\}$ define la expresión regular para el lenguaje: $L = \{w \mid |w| \text{ es par y } |w|>2 \}$

$$((a + b)(a+b))\ ((a+b)\ (a+b))^{+}$$

TEMA 4

AUTÓMATAS FINITOS

INTRODUCCIÓN

Gramáticas	Lenguajes	Máquinas
Sin restricciones o de Tipo 0	Sin restricciones o de Tipo 0	Máquina de Turing
Sensible al contexto o de Tipo 1	Sensible al contexto o de Tipo 1	Autómata linealmente acotado
Libre de contexto o de Tipo 2	Libre de contexto o de Tipo 2	Autómata a pila
Regular o de Tipo 3	**Regular o de Tipo 3**	**Autómata Finito**

INTRODUCCIÓN

- **AFD**: Autómatas finitos **deterministas**
- **AFND**: Autómatas finitos **no deterministas**
- **AFND-λ**: Autómatas finitos **no deterministas** con transiciones **vacías**.

INTRODUCCIÓN

Clasificación en función del tipo de **control**:

- **Deterministas**, el autómata únicamente puede estar en un estado en un momento determinado.
- **No Deterministas**, el autómata puede estar en varios estados simultáneamente.

Los no deterministas permiten **describir** más **eficientemente** determinados **problemas**.

INTRODUCCIÓN

- Los **autómatas finitos no deterministas** pueden ser transformados a **autómatas finitos deterministas.**
- Existe otro tipo de autómata finito que **amplía** los **no deterministas** permitiendo la **transición** de un estado a otro **espontáneamente** por medio de transiciones con la **cadena vacía**. Son los autómatas finitos **no deterministas** con **transiciones vacías**.

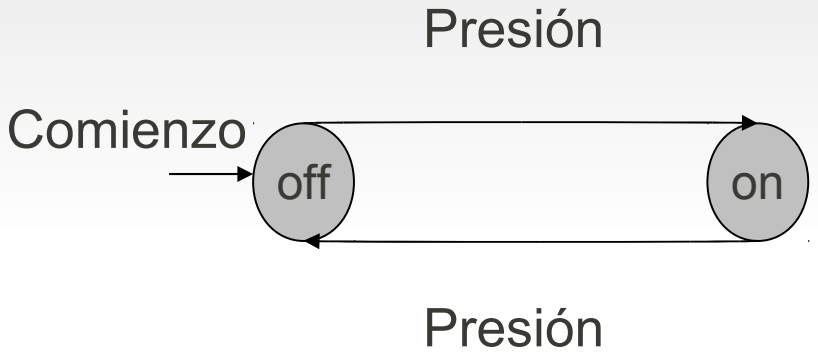
INTRODUCCIÓN

- Estos últimos **aceptan** el **mismo** tipo de **lenguajes** (los regulares).
- También existe una **equivalencia** entre las expresiones regulares y los autómatas finitos

INTRODUCCIÓN

- Sistema: **Interruptor**.
- El sistema debe recordar si está **conectado** (ON) o **desconectado** (OFF) Estados
- El usuario interactúa con el sistema ejerciendo presión o pulsando un interruptor Instrucciones
- Si está en **OFF** y es presionado el sistema pasa al estado **ON**.
- Si está en **ON** y es presionado el sistema pasa al estado **OFF**.

→ **Control**



1. CONCEPTOS GENERALES

- Un Autómata Finito **transita** entre estados de un conjunto finito de estados **según reciben símbolos** que forman la **palabra de entrada**.
- Su **salida** se **limita** a **dos** valores:

1. **Aceptación** de la palabra de entrada
2. **No Aceptación** de la palabra de entrada.

1. CONCEPTOS GENERALES

Tres tipos de estados:

1. Estado **Inicial**.

2. Estados **Intermedios**.

3. Estado/**os** **Final/es** → Si **finaliza** la **recepción** de símbolos de entrada y el autómata está en uno de estos estados, se considera que hay **ACEPTACIÓN**.

1. CONCEPTOS GENERALES

REPRESENTACIÓN DE AUTÓMATAS

- **Diagrama de transiciones**: es un grafo dirigido, donde cada **nodo** representa un **estado** del autómata. Cada **arco** etiquetado (desde un estado a otro) representa la **transición** entre los dos estados, siendo la **etiqueta** el **símbolo** del **alfabeto** asociado a la transición.

1. CONCEPTOS GENERALES

REPRESENTACIÓN DE AUTÓMATAS

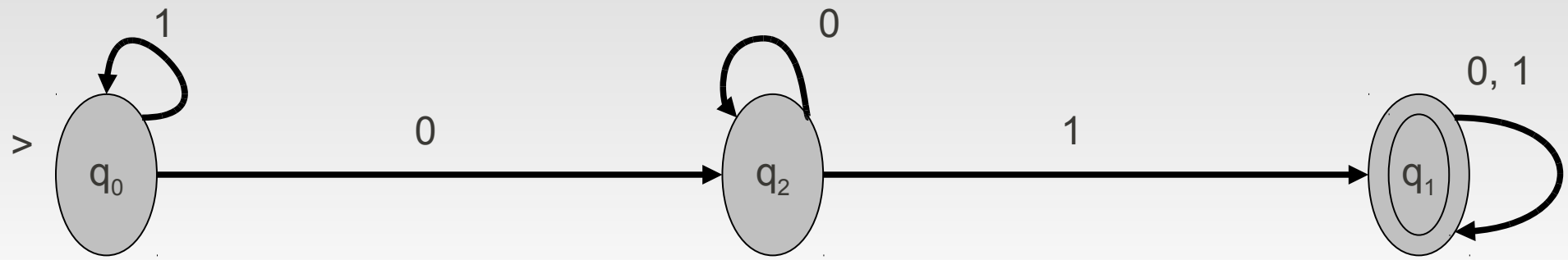
- **Tabla de transición**: Es una **tabla** que representa a la **función** de **transición (δ)**, donde las **filas** representan a los **estados** y las **columnas** a los **símbolos** del alfabeto de entrada. Cada **casilla** de la tabla representa el valor asociado a **δ(q,a)**.

2. AFD

$$AFD = (Q, \Sigma, \delta, q_0, F)$$

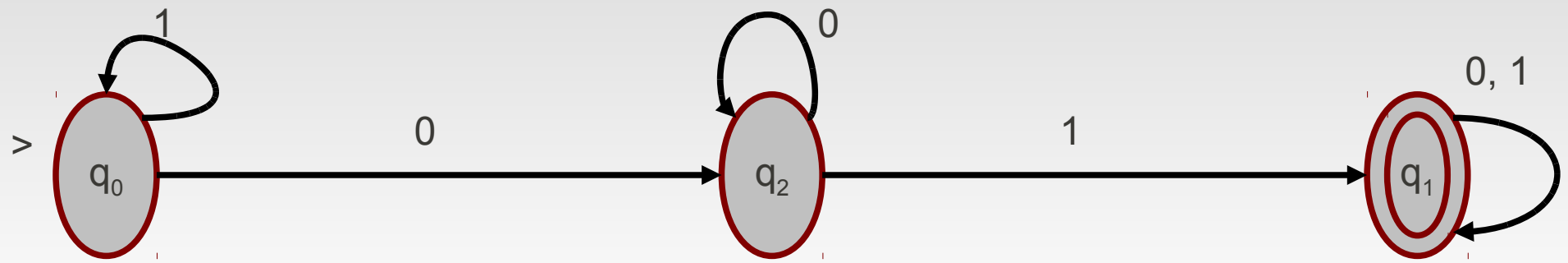
- $Q \rightarrow$ conjunto de estados
- $\Sigma \rightarrow$ alfabeto de símbolos de entrada
- $\delta \rightarrow$ función de transición entre estados
- $q_0 \rightarrow$ estado inicial
- $F \rightarrow$ conjunto de estados finales

2.1. Representación AFD



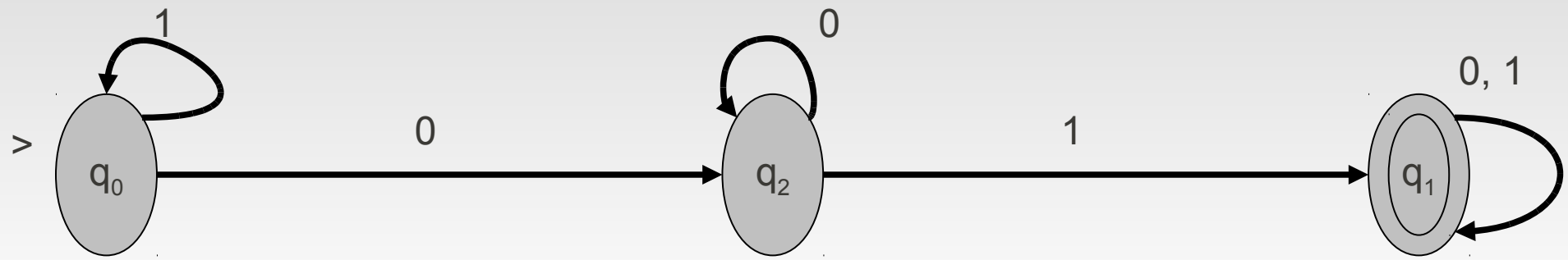
- Q → conjunto de estados
- ∑ → alfabeto de símbolos de entrada
- δ → función de transición entre estados
- q_0 → estado inicial
- F → conjunto de estados finales

2.1. Representación AFD



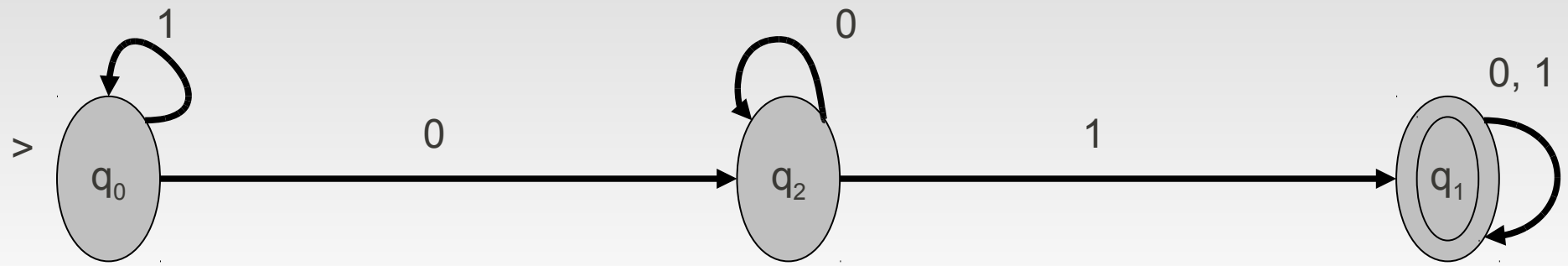
- Q → conjunto de estados
- ∑ → alfabeto de símbolos de entrada
- δ → función de transición entre estados
- q_0 → estado inicial
- F → conjunto de estados finales

2.1. Representación AFD



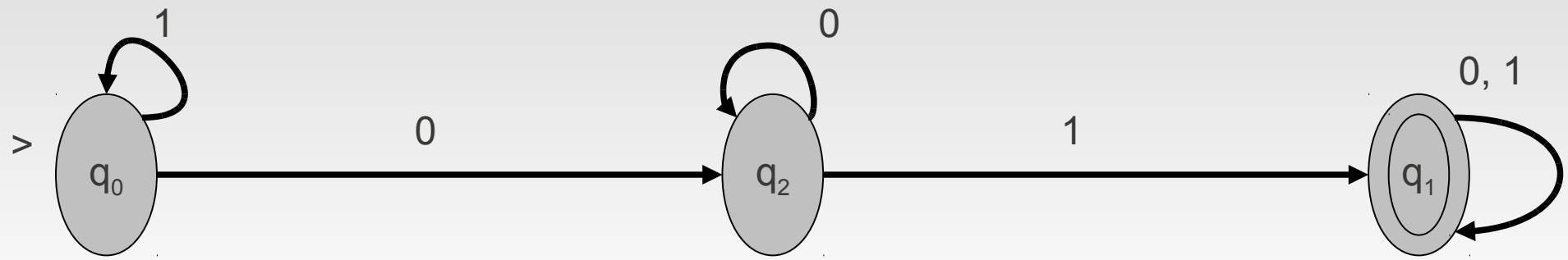
- Q → conjunto de estados
- ∑ → alfabeto de símbolos de entrada → {0, 1}
- δ → función de transición entre estados
- q_0 → estado inicial
- F → conjunto de estados finales

2.1. Representación AFD



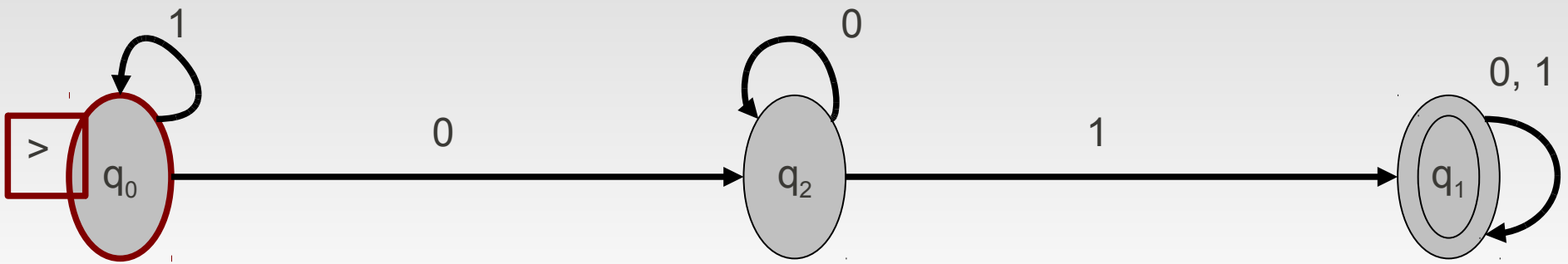
- Q → conjunto de estados
- Σ → alfabeto de símbolos de entrada
- $\delta: Q \times \Sigma \rightarrow Q$; $\delta(q_0,0) = q_2$; $\delta(q_1,1) = ?$
- q_0 → estado inicial
- F → conjunto de estados finales

2.1. Representación AFD



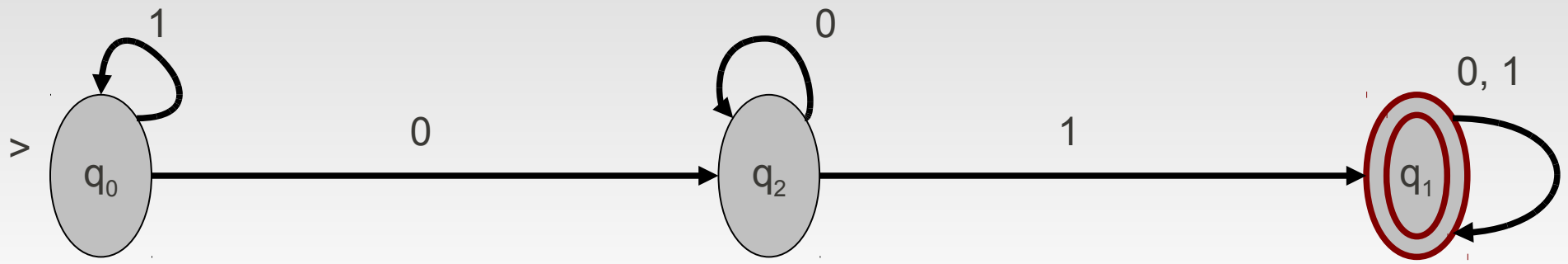
- Q → conjunto de estados
- ∑ → alfabeto de símbolos de entrada
- δ: Q x ∑ → Q; $\delta(q_0,0) = q_2$; $\delta(q_1,1) = q_1$
- q_0 → estado inicial
- F → conjunto de estados finales

2.1. Representación AFD



- Q → conjunto de estados
- ∑ → alfabeto de símbolos de entrada
- δ → función de transición entre estados
- $q_0 \in Q$ → Se marca con un ángulo o flecha ¿Por qué?
- F → conjunto de estados finales

2.1. Representación AFD



- Q → conjunto de estados
- Σ → alfabeto de símbolos de entrada
- δ → función de transición entre estados
- q_0 → estado inicial
- $F \subseteq Q$ ¿Por qué $F \subseteq Q$ y no $F \in Q$?

2.1. Representación AFD

	0	1
q_0	q_2	q_0
q_1	q_1	q_1
q_2	q_2	q_1

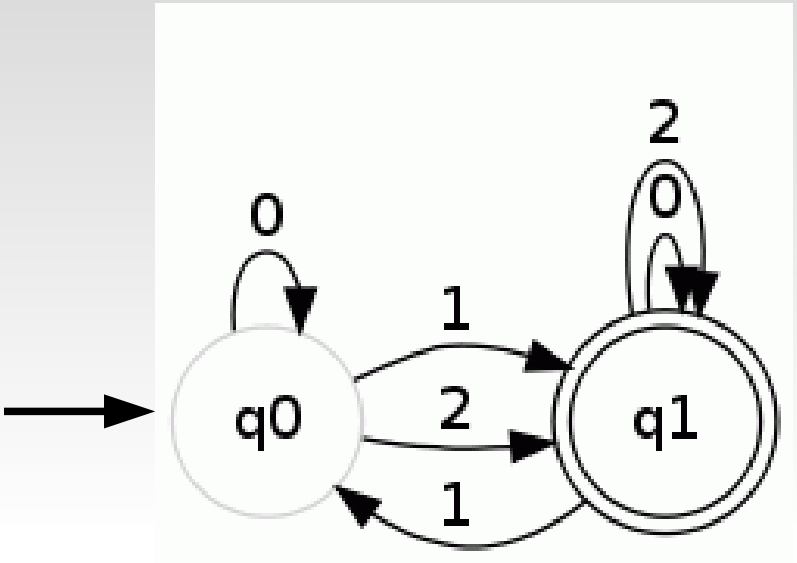
¿Qué información de la definición de autómata faltaría?

2.1. Representación AFD

	0	1
→ q_0	q_2	q_0
* q_1	q_1	q_1
q_2	q_2	q_1

¿Qué información de la definición de autómata faltaría?

2.1. Representación AFD



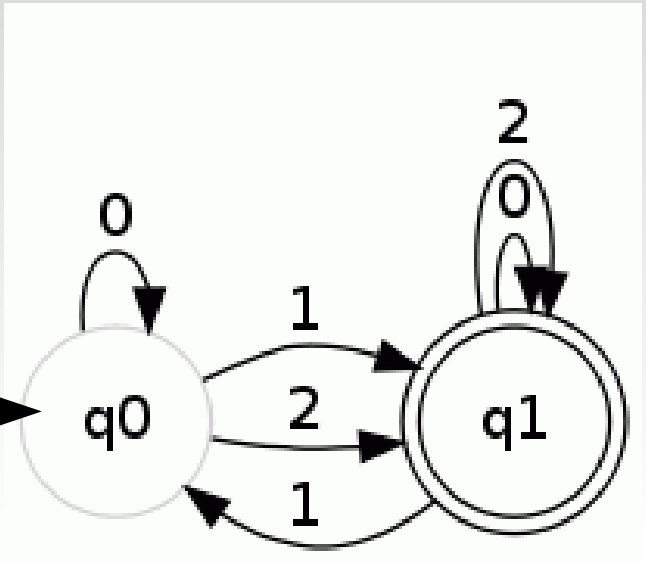
Q: ?

Σ: ?

F: ?

$\delta(q_0)$: ?

2.1. Representación AFD



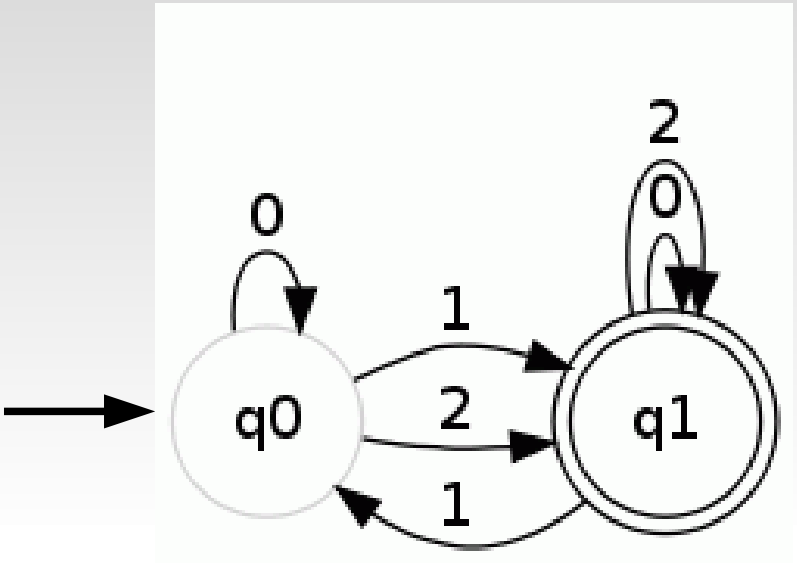
Q: ?

Σ: ?

F: ?

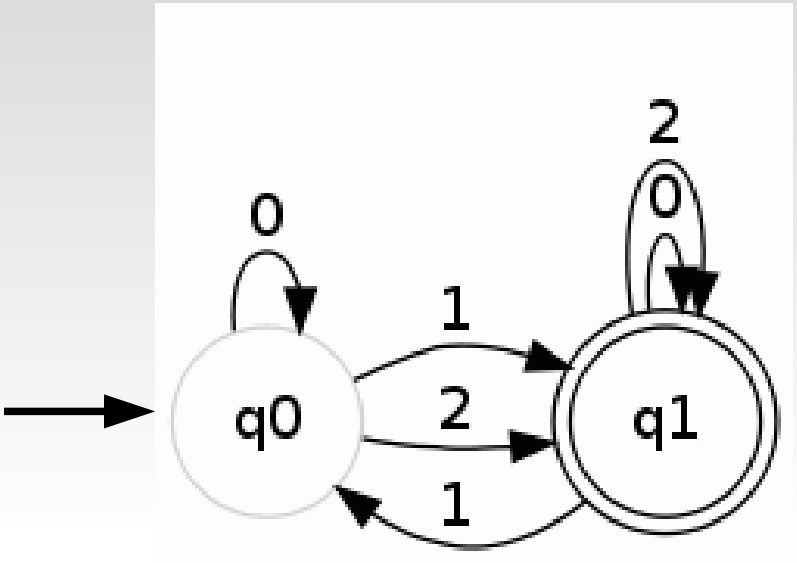
$\delta(q_{0,}1)$: ?

2.1. Representación AFD



¿Tabla de Transición?

2.1. Representación AFD



	0	1	2
>q0	q0	q1	q1
*q1	q1	q0	q1

2.2. Lenguaje aceptado por un AFD

- Sea AFD = $(Q, \sum, \delta, q_0, F)$ se define la **función de transición sucesiva** (δ^*) aplicable a cadenas:

$$\delta^*: (Q \times \sum^* \rightarrow Q)$$

(i) $\forall q \in Q,\ \delta^*(q,\lambda)=q$

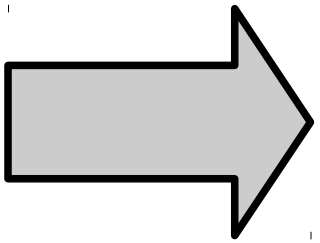
(ii) $\forall y \in \sum^*,\ a \in \sum,\ q \in Q,\ \delta^*(q,ya)=\delta(\delta^*(q,y),a)$

2.2. Lenguaje aceptado por un AFD

- Una **palabra** $w \in \Sigma^*$ **es aceptada** por AFD = $(Q, \Sigma, \delta, q_0, F) \leftrightarrow \delta^*(q_0, w) \in F$:

¿ 110 es aceptada por AFD_0?

AFD_0



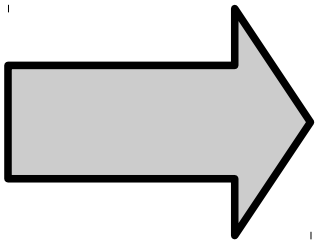
	0	1
→ q_0	q_2	q_0
* q_1	q_1	q_1
q_2	q_2	q_1

2.2. Lenguaje aceptado por un AFD

- Una **palabra** $w \in \Sigma^*$ **es aceptada** por AFD = $(Q, \Sigma, \delta, q_0, F)$ <--> $\delta^*(q_0,w) \in F$:

$$\delta^*(q_0,110)=\delta(\delta^*(q_0,11),0)=\delta(\delta(\delta^*(q_0,1),1),0)=$$
$$=\delta(\delta(\delta(\delta^*(q_0,\boldsymbol{\lambda}),1),1),0)=$$

AFD$_0$



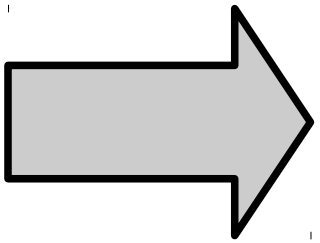
	0	1
→ q_0	q_2	q_0
* q_1	q_1	q_1
q_2	q_2	q_1

2.2. Lenguaje aceptado por un AFD

- Una **palabra** $w \in \Sigma^*$ **es aceptada** por AFD = $(Q, \Sigma, \delta, q_0, F) \leftrightarrow \delta^*(q_0, w) \in F$:

$$=\delta(\delta(\delta(\delta^*(q_0,\boldsymbol{\lambda}),1),1),0)=\delta(\delta(\delta(q_0,1),1),0)=$$

$$\delta(\delta(q_0,1),0)=\delta(q_0,0)=q_2 \text{ no pertenece a F}$$

AFD_0 

	0	1
→ q_0	q_2	q_0
* q_1	q_1	q_1
q_2	q_2	q_1

2.2. Lenguaje aceptado por un AFD

- Una **palabra** $w \in \Sigma^*$ **es aceptada** por AFD = $(Q, \Sigma, \delta, q_0, F) \leftrightarrow \delta^*(q_0,w) \in F$:

$\delta^*(q_0,110)$

1. $\delta^*(q_0,\lambda)=q_0$
2. $\delta^*(q_0,1)=\delta(q_0,1)=q_0$
3. $\delta^*(q_0,11)=\delta(q_0,1)=q_0$
4. $\delta^*(q_0,110)=\delta(q_0,0)=q_2$

	0	1
→ q_0	q_2	q_0
* q_1	q_1	q_1
q_2	q_2	q_1

2.2. Lenguaje aceptado por un AFD

- Una **palabra** $w \in \Sigma^*$ **es aceptada** por AFD = $(Q, \Sigma, \delta, q_0, F) \leftrightarrow \delta^*(q_0, w) \in F$:

$\delta^*(q_0, 0110)$

	0	1
→ q_0	q_2	q_0
* q_1	q_1	q_1
q_2	q_2	q_1

2.2. Lenguaje aceptado por un AFD

- Una **palabra** $w \in \Sigma^*$ **es aceptada** por AFD = $(Q, \Sigma, \delta, q_0, F)$ <--> $\delta^*(q_0,w) \in F$:

$\delta^*(q_0,0110)$

1. $\delta^*(q_0,\lambda)=q_0$
2. $\delta^*(q_0,0)=\delta(q_0,0)=q_2$
3. $\delta^*(q_0,01)=\delta(q_2,1)=q_1$
4. $\delta^*(q_0,011)=\delta(q_1,1)=q_1$
4. $\delta^*(q_0,0110)=\delta(q_1,0)=q_1$

	0	1
→ q_0	q_2	q_0
* q_1	q_1	q_1
q_2	q_2	q_1

2.2. Lenguaje aceptado por un AFD

- Dado un AFD se llama **lenguaje aceptado** por dicho autómata al conjunto de **palabras** de $\sum^*$ que acepta.

$$L(AFD)=\{x \in \sum^* / \delta^*(q_0,x) \in F\}$$

3. AFND

$$AFND = (Q, \Sigma, \delta, q_0, F)$$

- $Q \rightarrow$ conjunto de estados
- $\Sigma \rightarrow$ alfabeto de símbolos de entrada
- $\delta \rightarrow$ **función de transición entre estados**
- $q_0 \rightarrow$ estado inicial
- $F \rightarrow$ conjunto de estados finales

3. AFND

$$AFND = (Q, \Sigma, \delta, q_0, F)$$

- $Q \rightarrow$ conjunto de estados
- $\Sigma \rightarrow$ alfabeto de símbolos de entrada
- $\delta: Q \times \Sigma \rightarrow \mathbf{P(Q)}; \delta(q,a) = \{q_i, ..., q_k\}$
- $q_0 \rightarrow$ estado inicial
- $F \rightarrow$ conjunto de estados finales

3. AFND

$$AFND = (Q, \Sigma, \delta, q_0, F)$$

- $Q \rightarrow$ conjunto de estados
- $\Sigma \rightarrow$ alfabeto de símbolos de entrada
- $\delta: Q \times \Sigma \rightarrow P(Q)$; ¿$\delta(q,a) = \Phi$?
- $q_0 \rightarrow$ estado inicial
- $F \rightarrow$ conjunto de estados finales

3. AFND

$$AFND = (Q, \Sigma, \delta, q_0, F)$$

- $Q \rightarrow$ conjunto de estados
- $\Sigma \rightarrow$ alfabeto de símbolos de entrada
- $\delta: Q \times \Sigma \rightarrow P(Q)$; ¿$\delta(q,a) = \phi$?
- $q_0 \rightarrow$ estado inicial
- $F \rightarrow$ conjunto de estados finales

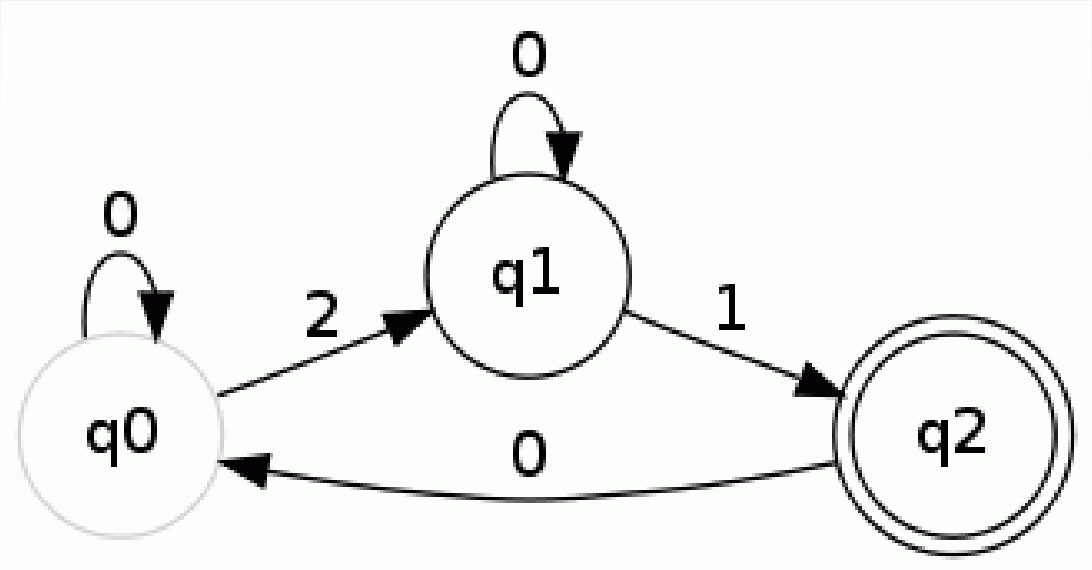
3. AFND

Diferencias con AFDs

- Puede haber **más** de **un estado activo** en cada instante.
- En un estado y con un único símbolo de entrada se puede **transitar** a **más** de **un estado**.
- Desde un estado con ciertos símbolos **no se puede transitar** a ningún estado → **Cadena no reconocida**

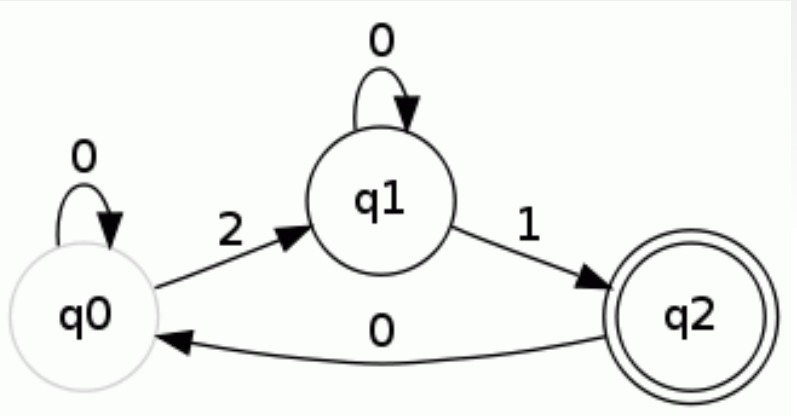
3.1. Representación AFND

- Dibuja la tabla de transición para el AFND representado mediante el diagrama de transición:



3.1. Representación AFND

- Dibuja la tabla de transición para el AFND representado mediante el diagrama de transición:



	0	1	2
>q0	q0	[ ]	q1
q1	q1	q2	[ ]
*q2	q0	[ ]	[ ]

3.2. Lenguaje aceptado por un AFND

- Sea AFND = $(Q, \sum, \delta, q_0, F)$ se define la **función de transición sucesiva** (δ^*) aplicable a cadenas:

$$\delta^*: (Q \times \sum^* \rightarrow \mathbf{P(Q)})$$

(i) $\forall q \in Q$, $\delta^*(q,\lambda) = \mathbf{\{q\}}$

(ii) $\forall y \in \sum^*$, $a \in \sum$, $\mathbf{p}$, $q \in Q$, $\delta^*(q,ya) = \mathbf{U\ \delta(p,a)}$

$\mathbf{p \in \delta^*(q,ya)}$

3.2. Lenguaje aceptado por un AFND

$\delta^*(q_0, 00101)$

	0	1
→ q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	ϕ	$\{q_2\}$
$^*q_2$	ϕ	ϕ

3.2. Lenguaje aceptado por un AFND

	0	1
→ q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	ϕ	$\{q_2\}$
$^*q_2$	ϕ	ϕ

$\delta^*(q_0, 00101)$

1. $\delta^*(q_0, \lambda)=\{q_0\}$
2. $\delta^*(q_0, 0)= \delta(q_0, 0)= \{q_0, q_1\}$
3. $\delta^*(q_0, 00)= \delta(q_0, 0) \cup \delta(q_1, 0)= \{q_0, q_1\} \cup \phi= \{q_0, q_1\}$
4. $\delta^*(q_0, 001)= \delta(q_0, 1) \cup \delta(q_1, 1)= \{q_0\} \cup \{q_2\}= \{q_0, q_2\}$
5. $\delta^*(q_0, 0010)= \delta(q_0, 0) \cup \delta(q_2, 0)= \{q_0, q_1\} \cup \phi= \{q_0, q_1\}$
6. $\delta^*(q_0, 00101)= \delta(q_0, 1) \cup \delta(q_1, 1)= \{q_0\} \cup \{q_2\}= \{q_0, q_2\}$

3.2. Lenguaje aceptado por un AFND

$\delta^*(q_0, 1101)$

	0	1
→ q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	ϕ	$\{q_2\}$
$^*q_2$	ϕ	ϕ

3.2. Lenguaje aceptado por un AFND

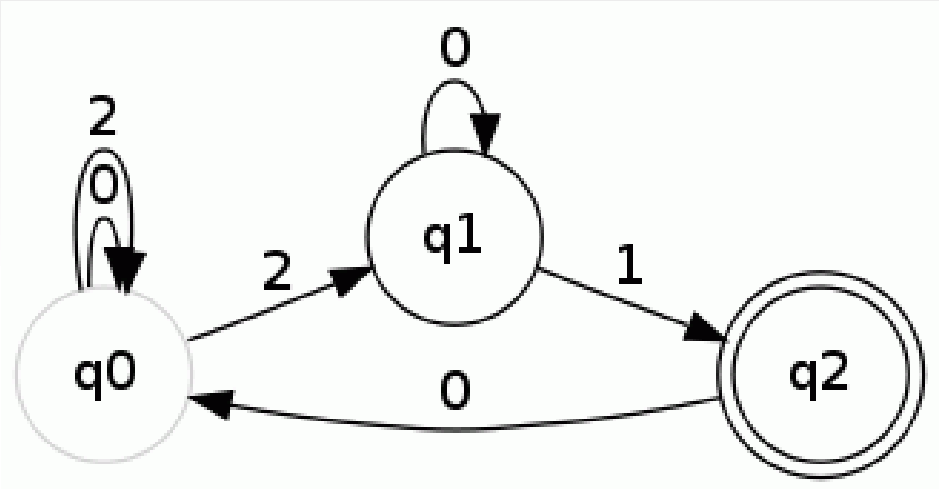
$$\delta^*(q_0, 1101)$$

1. $\delta(q_0,1)=\{q_0\}$
2. $\delta(q_0,1)=\{q_0\}$
3. $\delta(q_0,0)=\{q_0, q_1\}$
4. $\delta(q_0,1) \cup \delta(q_1,1)= \{q_0\} \cup \{q_2\} =\{q_{0,} q_2\}$

3.2. Lenguaje aceptado por un AFND

¿Cadenas 22, 21021 ?

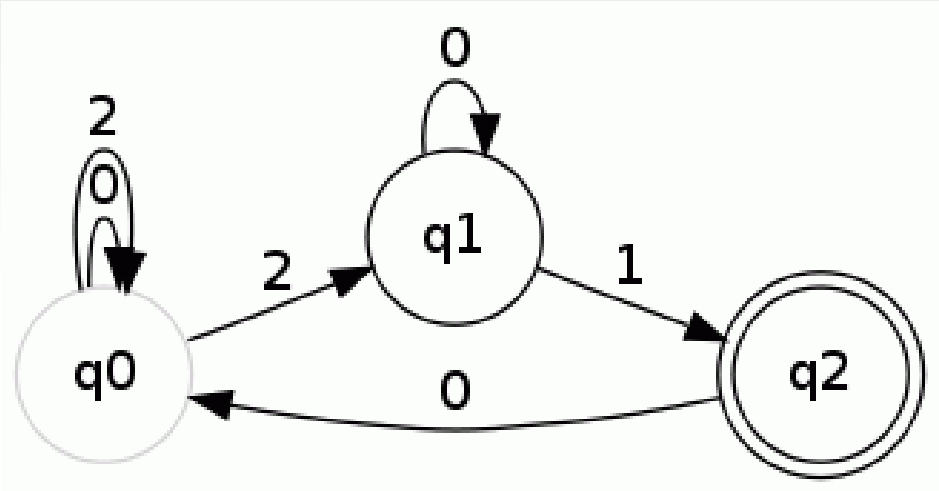
	0	1	2
>q0	{q0}	[ ]	{q0, q1}
q1	{q1}	{q2}	[ ]
*q2	{q0}	[ ]	[ ]



3.2. Lenguaje aceptado por un AFND

¿Cadenas 22, 21021 ?

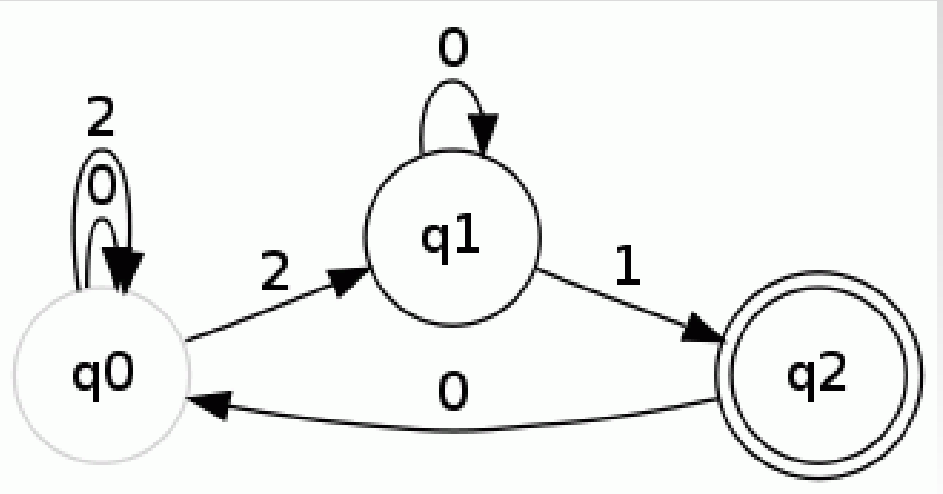
	0	1	2
>q0	{q0}	[ ]	{q0, q1}
q1	{q1}	{q2}	[ ]
*q2	{q0}	[ ]	[ ]



$\delta^*(q_0,22)$:

1. $\delta(q_0,2)=\{q_0,q_1\} \rightarrow$
2. $\delta(q_0,2) \cup \delta(q_1,2) = \{q_0,q_1\} \cup [] = \{q_0,q_1\}$

3.2. Lenguaje aceptado por un AFND



¿Cadenas 22, 21021 ?

$\delta^*(q_0,21021)$:

1. $\delta(q_0,2)=\{q_0,q_1\} \rightarrow$
2. $\delta(q_0,1) \cup \delta(q_1,1) = [] \cup \{q_2\} = \{q_2\} \rightarrow$
3. $\delta(q_2,0) = \{q_0\} \rightarrow$
4. $\delta(q_0,2)=\{q_0,q_1\} \rightarrow$
5. $\delta(q_0,1) \cup \delta(q_1,1) = [] \cup \{q_2\} = \{q_2\}$

3.2. Lenguaje aceptado por un AFND

- Una **palabra** $w \in \Sigma^*$ **es aceptada** por AFND = $(Q, \Sigma, \delta, q_0, F)$ <--> $\boldsymbol{\delta^*(q_0,w) \cap F \neq \phi}$
- Dado un AFND se llama **lenguaje aceptado** por dicho autómata al conjunto de **palabras** de Σ^* que acepta.

$$L(AFND)=\{x \in \Sigma^* / \delta^*(q_0,x) \cap F \neq \phi\}$$

4. AFND-λ

Autómatas finitos **no deterministas** con transiciones **vacías**.

$$AFND\text{-}\lambda = (Q, \Sigma, \delta, q_0, F)$$

- $Q \rightarrow$ conjunto de estados
- $\Sigma \rightarrow$ alfabeto de símbolos de entrada
- $\delta: Q \times (\Sigma \cup \lambda) \rightarrow P(Q)$;
- $q_0 \rightarrow$ estado inicial
- $F \rightarrow$ conjunto de estados finales

4.1. Representación AFND-λ

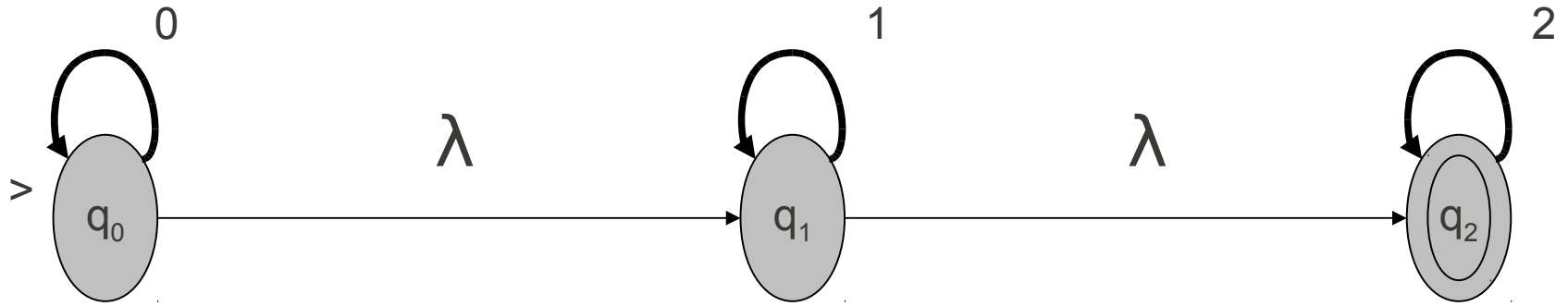
Dibuja el diagrama de transición

	λ	0	1	2
→ q_0	$\{q_1\}$	$\{q_0\}$	$\varnothing$	$\varnothing$
q_1	$\{q_2\}$	$\varnothing$	$\{q_1\}$	$\varnothing$
$^*q_2$	$\varnothing$	$\varnothing$	$\varnothing$	$\{q_2\}$

4.1. Representación AFND-λ

Dibuja el diagrama de transición

	λ	0	1	2
→ q_0	$\{q_1\}$	$\{q_0\}$	∅	∅
q_1	$\{q_2\}$	∅	$\{q_1\}$	∅
$^*q_2$	∅	∅	∅	$\{q_2\}$



4.2. λ-Clausura

- **Clausura de λ respecto de un estado** ($\lambda(q)$): conjunto de estados a los que se puede **llegar** desde q utilizando **transiciones** etiquetadas con λ. Se incluye también al propio estado q.

$\lambda(q_0)$?

4.2. λ-Clausura

- **Clausura de λ respecto de un estado** ($\lambda(q)$): conjunto de estados a los que se puede **llegar** desde q utilizando **transiciones** etiquetadas con λ. Se incluye también al propio estado q.

$$\lambda(q_0) = \{q_0, q_1, q_2\}$$

4.2. λ-Clausura

- **Clausura de λ respecto a un conjunto de estados** ($\lambda(S)$; $S \subseteq Q$):

$$\lambda(S) = \bigcup_{q \in S} \lambda(q)$$

4.3. Lenguaje aceptado por un AFND-λ

- Sea AFND-λ=(Q, Σ, δ, q_0, F) se define la **función de transición sucesiva** (δ*) aplicable a cadenas:

$$\delta^*: Q \times \Sigma^* \rightarrow P(Q)$$

(i) $\delta^*(q, \lambda) = \lambda(q), \forall q \in Q$

(ii) $\delta^*(q, ya) = \lambda \left(\bigcup_{p \in \delta^*(q, y)} \delta(p, a) \right), \forall q \in Q, y \in \Sigma^*, a \in \Sigma$

4.3. Lenguaje aceptado por un AFND-λ

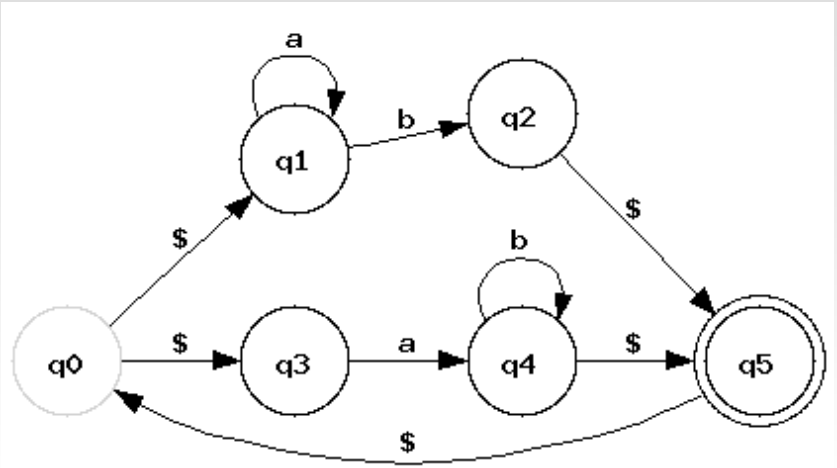
- Una **palabra** $w \in \Sigma^*$ **es aceptada** por AFND-λ = $(Q, \Sigma, \delta, q_0, F)$ <--> $\boldsymbol{\delta^*(q_0,w) \cap F \neq \phi}$

- Dado un AFND-λ se llama **lenguaje aceptado** por dicho autómata al conjunto de **palabras** de Σ^* que acepta.

$$L(\text{AFND-}\lambda)=\{x \in \Sigma^* / \delta^*(q_0,x) \cap F \neq \phi\}$$

4.3. Lenguaje aceptado por un AFND-λ

Lenguaje: (a*b +ab*)+



δ	a	b	λ	λ(qi)
>q0	ϕ	ϕ	{q1,q3}	{q0,q1,q3}
q1	{q1}	{q2}	{}	{q1}
q2	ϕ	ϕ	{q5}	{q2,q5,q0,q1,q3}
q3	{q4}	ϕ	ϕ	{q3}
q4	ϕ	{q4}	{q5}	{q4,q5,q0,q1,q3}
*q5	ϕ	ϕ	{q0}	{q5,q0,q1,q3}

λ(q0)={q0,q1,q3}

λ(q1)={q1}

λ(q2)={q2,q5,q0,q1,q3}

λ(q3)={q3}

λ(q4)={q4,q5,q0,q1,q3}

λ(q5)={q5,q0,q1,q3}

4.3. Lenguaje aceptado por un AFND-λ

$$\delta^*(q_0, aab)$$

1. $\delta(q0,a) = \lambda(\delta(q0,a) \cup \delta(q1,a) \cup \delta(q3,a)) = \lambda(\phi \cup \{q1\} \cup \{q4\}) = \{q1,q4,q5,q0,q3\}$

2. $\lambda(\delta(q1,a) \cup \delta(q4,a) \cup \delta(q5,a) \cup \delta(q0,a) \cup \delta(q3,a)) = \lambda(\{q1,q4\}) = \{q1, q4, q5, q0, q3\}$

3. $\lambda(\delta(q1,b) \cup \delta(q4,b) \cup \delta(q5,b) \cup \delta(q0,b) \cup \delta(q3,b)) = \lambda(\{q2,q4\}) = \{q2,q5,q0,q1,q3, q4\}$

4.3. Lenguaje aceptado por un AFND-λ

$\delta^*(q_0, ba)$

4.3. Lenguaje aceptado por un AFND-λ

$\delta^*(q_0, ba)$

1. $\lambda(\delta(q_0,b) \cup \delta(q_1,b) \cup \delta(q_3,b))=\lambda(\phi \cup \{q_2\} \cup \phi)=\lambda(\{q_2\})=\{q_0, q_1, q_2, q_3, q_5\}$
2. $\lambda(\delta(q_0,a) \cup \delta(q_1,a) \cup \delta(q_2,a) \cup \delta(q_3,a) \cup \delta(q_5,a))=\lambda(\phi \cup \{q_1\} \cup \phi \cup \{q_4\} \cup \phi)=\lambda(\{q_1,q_4\})=\{q_1, q_4, q_5, q_0, q_3\}$

4.3. Lenguaje aceptado por un AFND-λ

Calcula la λ-clausura para cada uno de los estados del autómata

	λ	0	1	2
→ q_0	$\{q_1\}$	$\{q_0\}$	∅	∅
q_1	$\{q_2\}$	∅	$\{q_1\}$	∅
$^*q_2$	∅	∅	∅	$\{q_2\}$

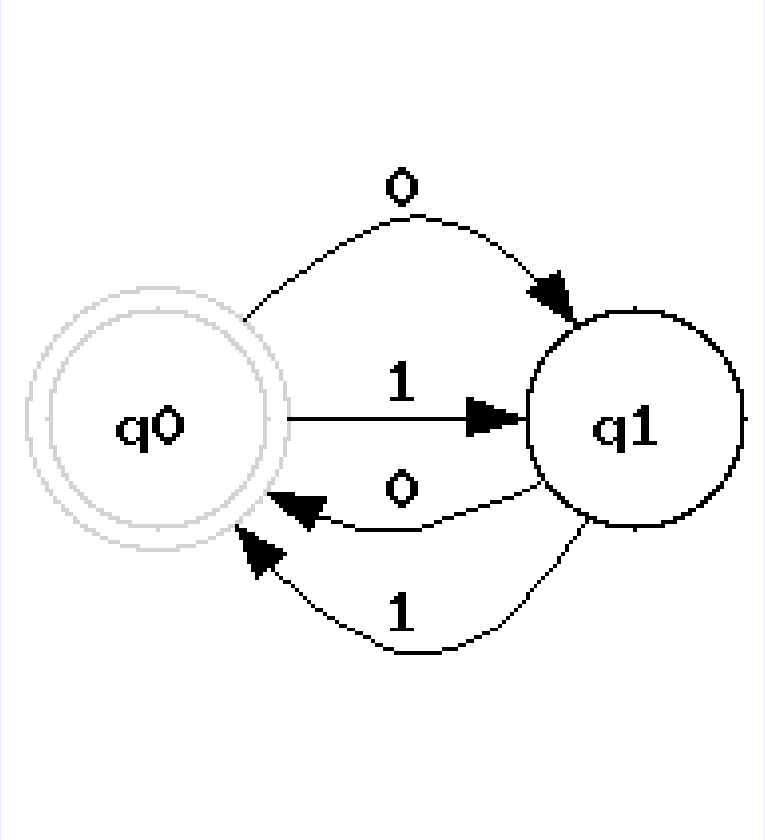
4.3. Lenguaje aceptado por un AFND-λ

$\delta^*(q_0, 1210)$

	λ	0	1	2
→ q_0	$\{q_1\}$	$\{q_0\}$	∅	∅
q_1	$\{q_2\}$	∅	$\{q_1\}$	∅
$^*q_2$	∅	∅	∅	$\{q_2\}$

5. EJERCICIOS

1.a Lenguaje de todas las cadenas de longitud par.

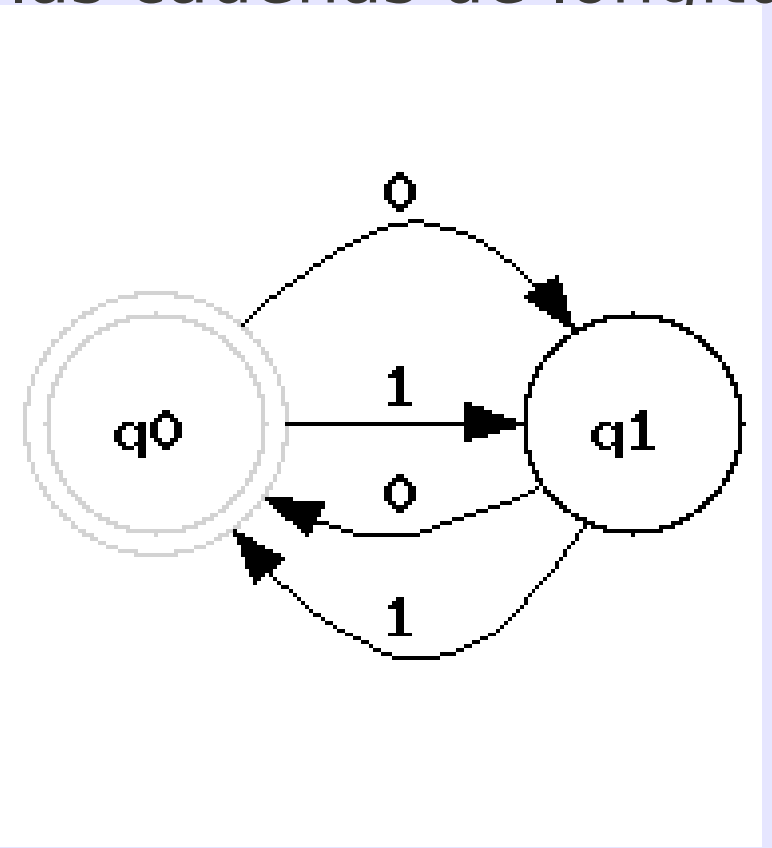


5. EJERCICIOS

1.a Lenguaje de todas las cadenas de longitud par.

¿Palabra vacía?

¿Estados Finales?

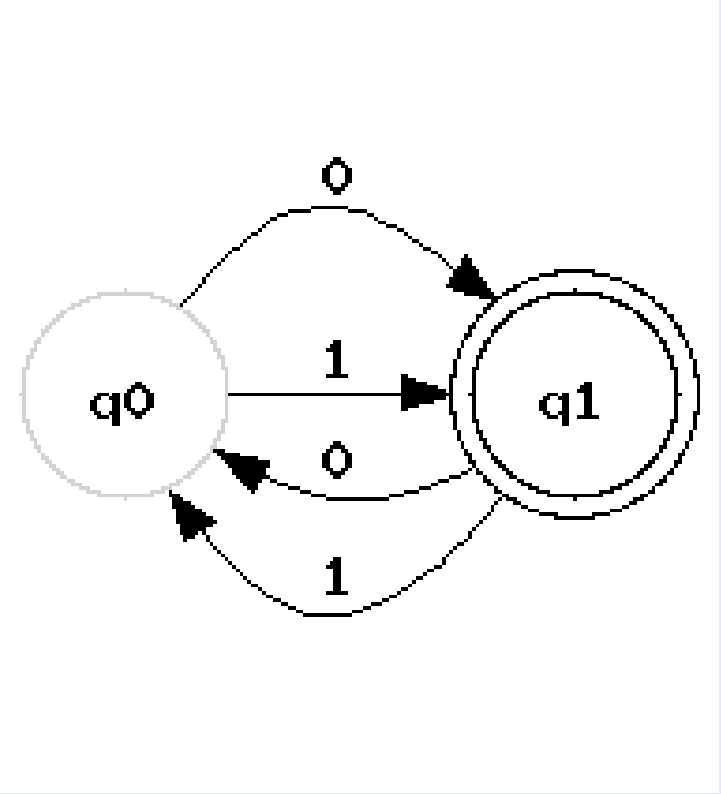


5. EJERCICIOS

1.b) Lenguaje de todas las cadenas de longitud impar.

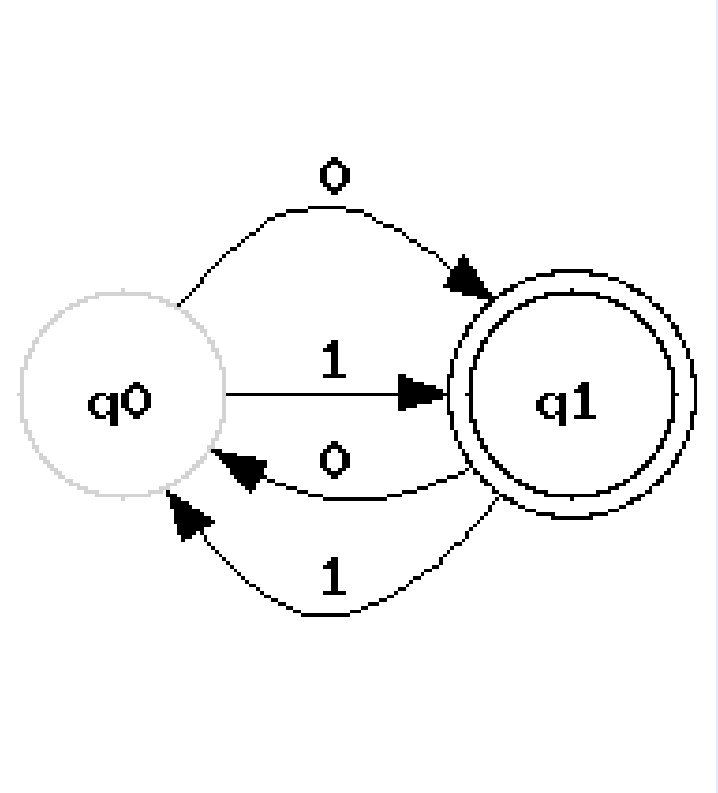
5. EJERCICIOS

1.b) Lenguaje de todas las cadenas de longitud impar.



5. EJERCICIOS

1.b) Lenguaje de todas las cadenas de **longitud** impar.



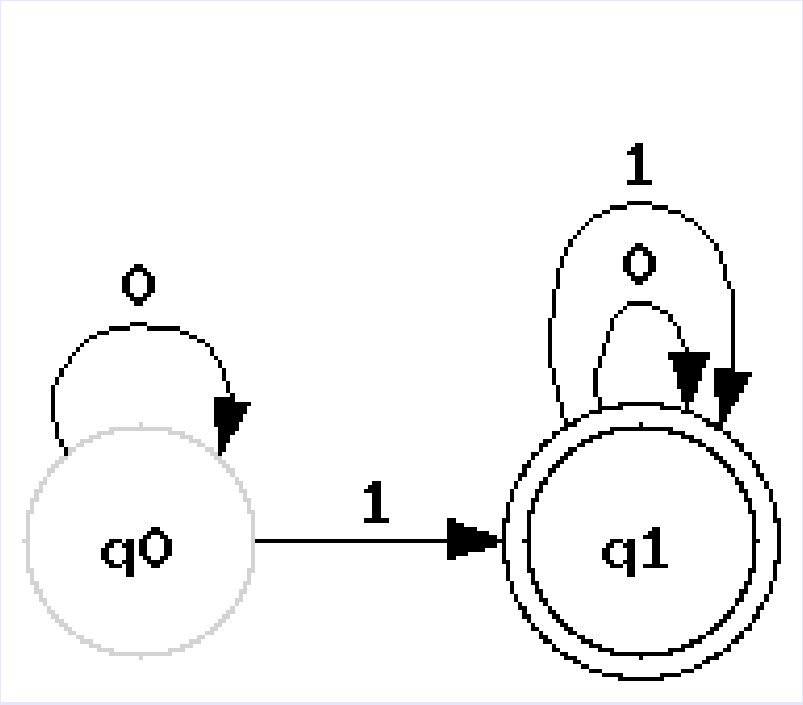
Longitud: contar el número de transiciones.

5. EJERCICIOS

1.c) Lenguaje de todas las cadenas que al menos tienen un 1.

5. EJERCICIOS

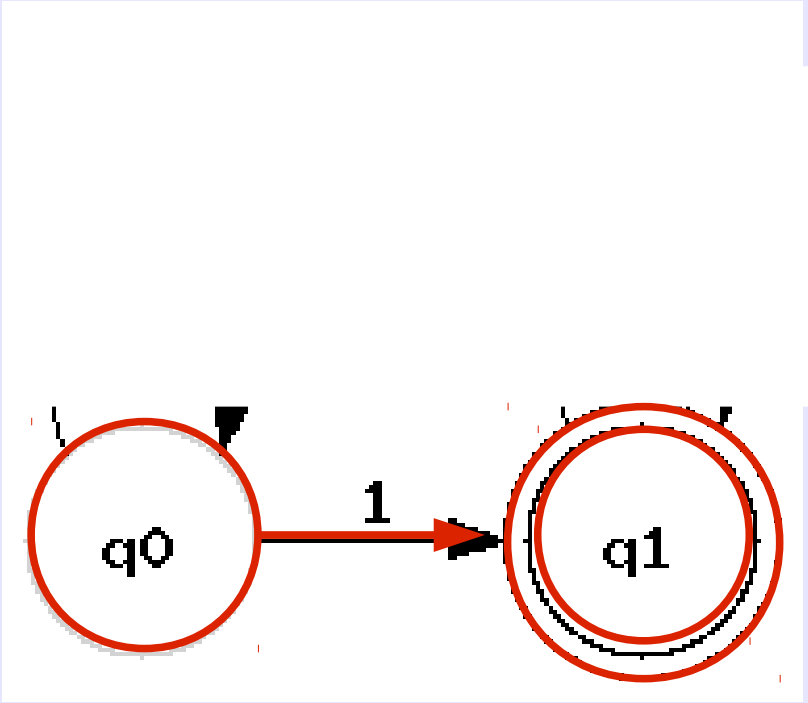
1.c) Lenguaje de todas las cadenas que al menos tienen un 1.



5. EJERCICIOS

1.c) Lenguaje de todas las cadenas que al menos **tienen** un 1.

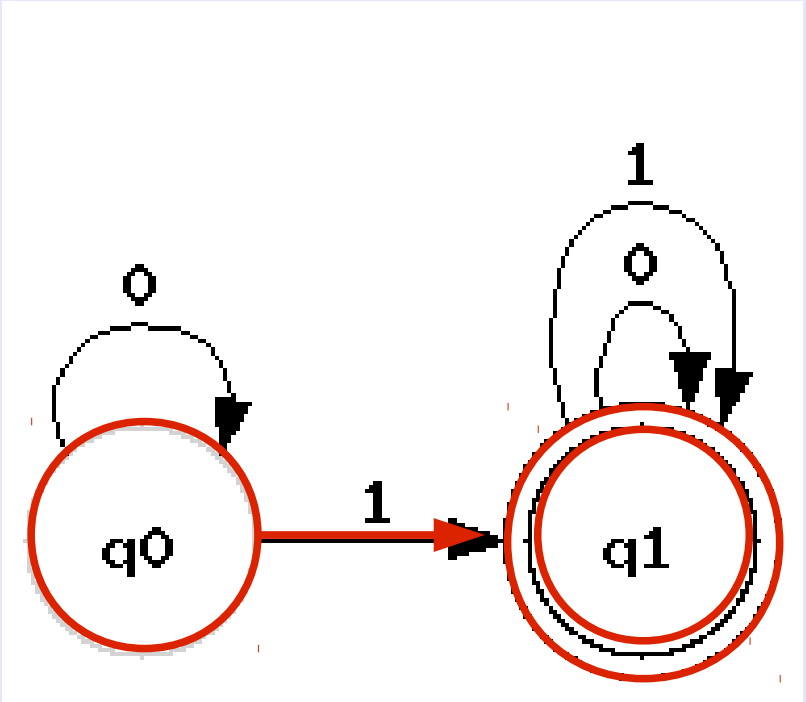
¿Palabra vacía?



5. EJERCICIOS

1.c) Lenguaje de todas las cadenas que al menos tienen un 1.

¿Palabra vacía?

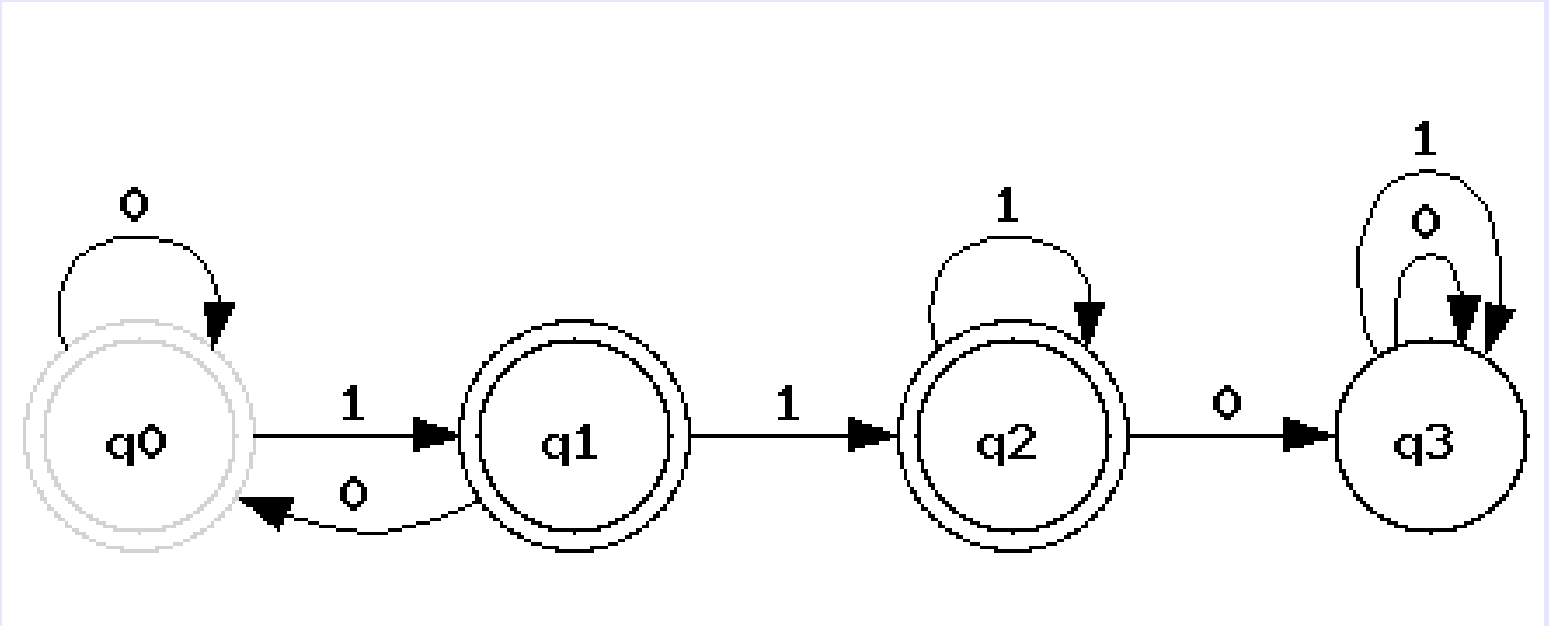


5. EJERCICIOS

1.f). Lenguaje de todas las cadenas que no contienen a la subcadena 110.

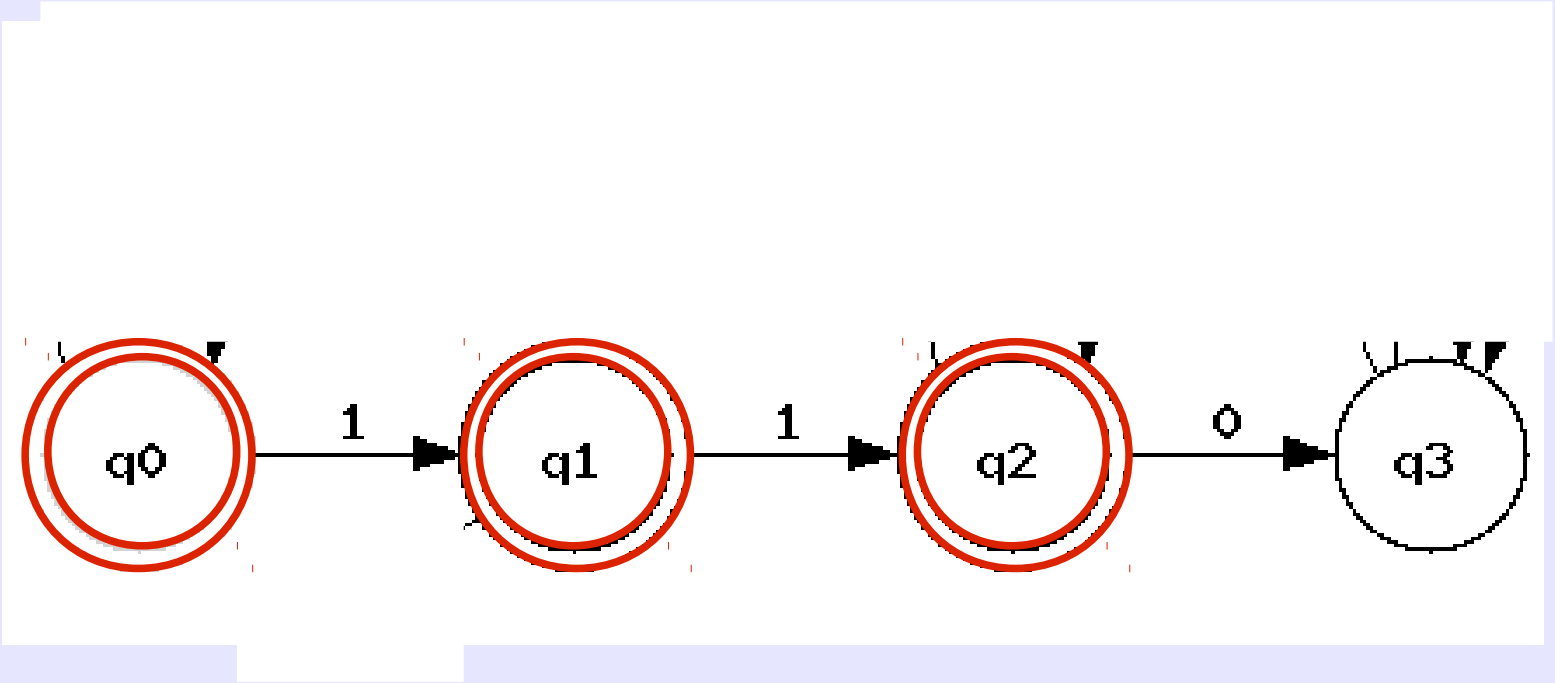
5. EJERCICIOS

1.f). Lenguaje de todas las cadenas que no contienen a la subcadena 110.



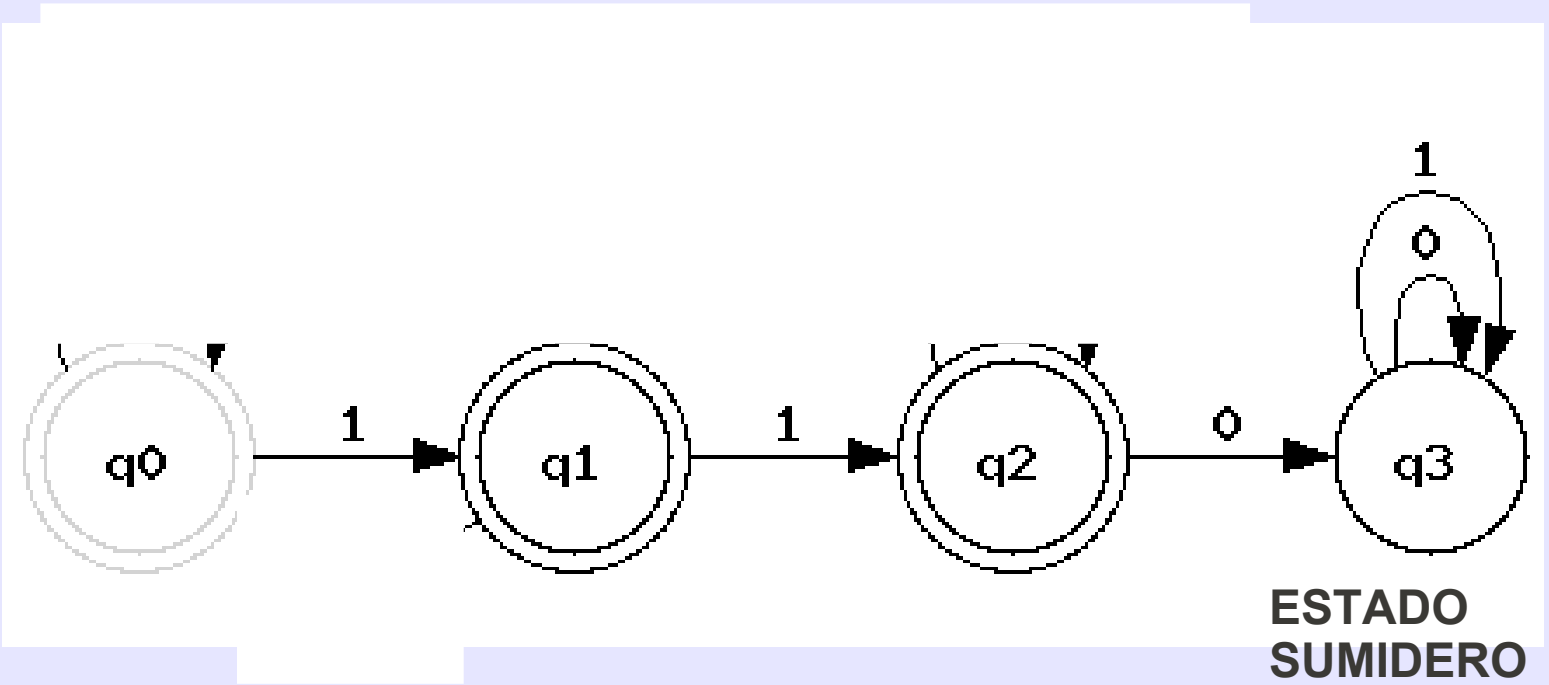
5. EJERCICIOS

1.f). Lenguaje de todas las cadenas que **no contienen** a la subcadena 110.



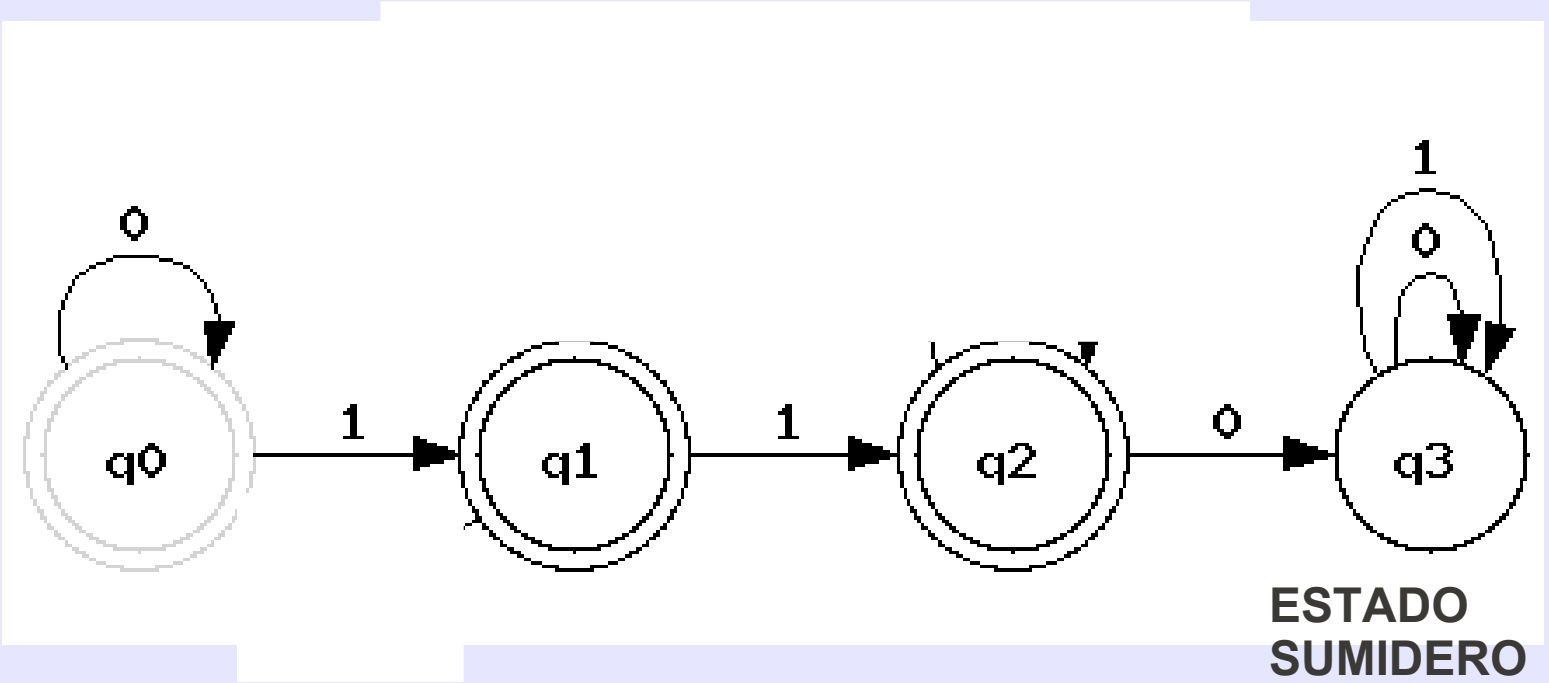
5. EJERCICIOS

1.f). Lenguaje de todas las cadenas que **no contienen** a la subcadena 110.



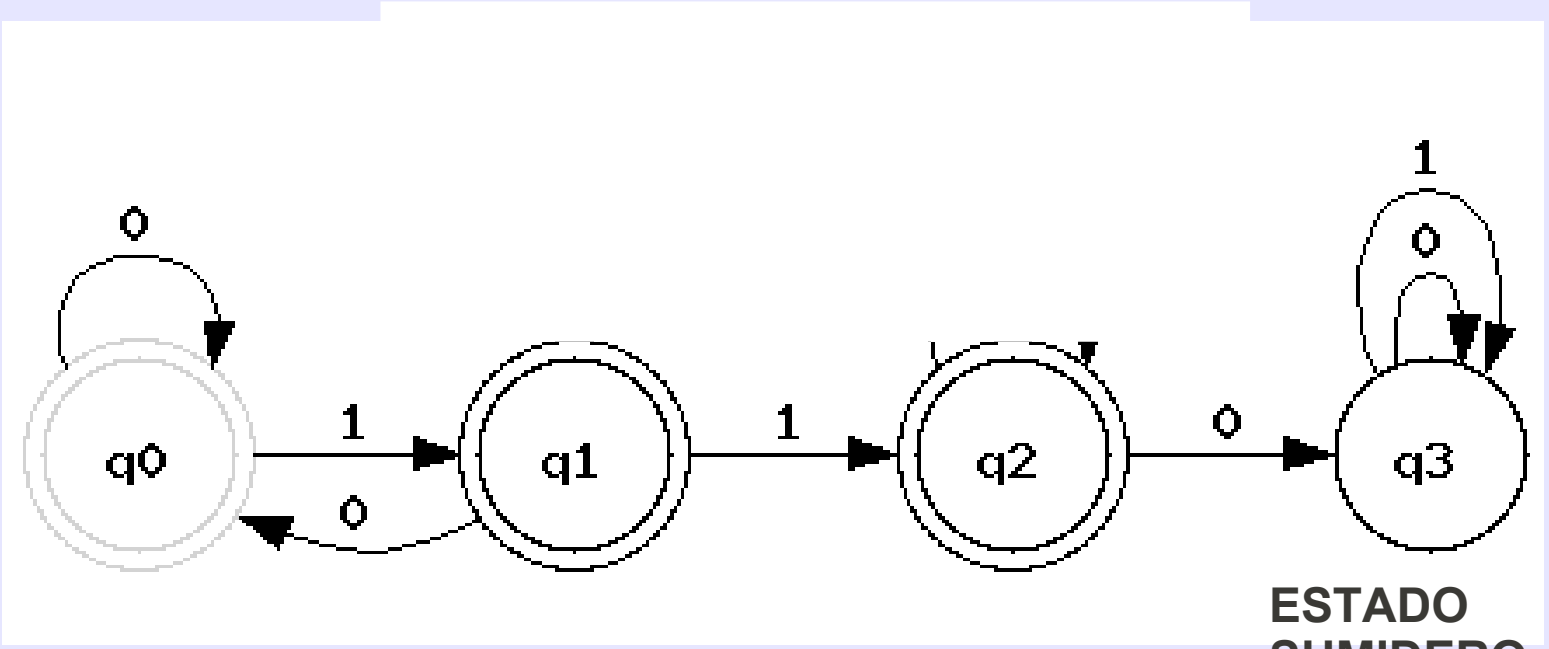
5. EJERCICIOS

1.f). Lenguaje de todas las cadenas que **no contienen** a la subcadena 110.



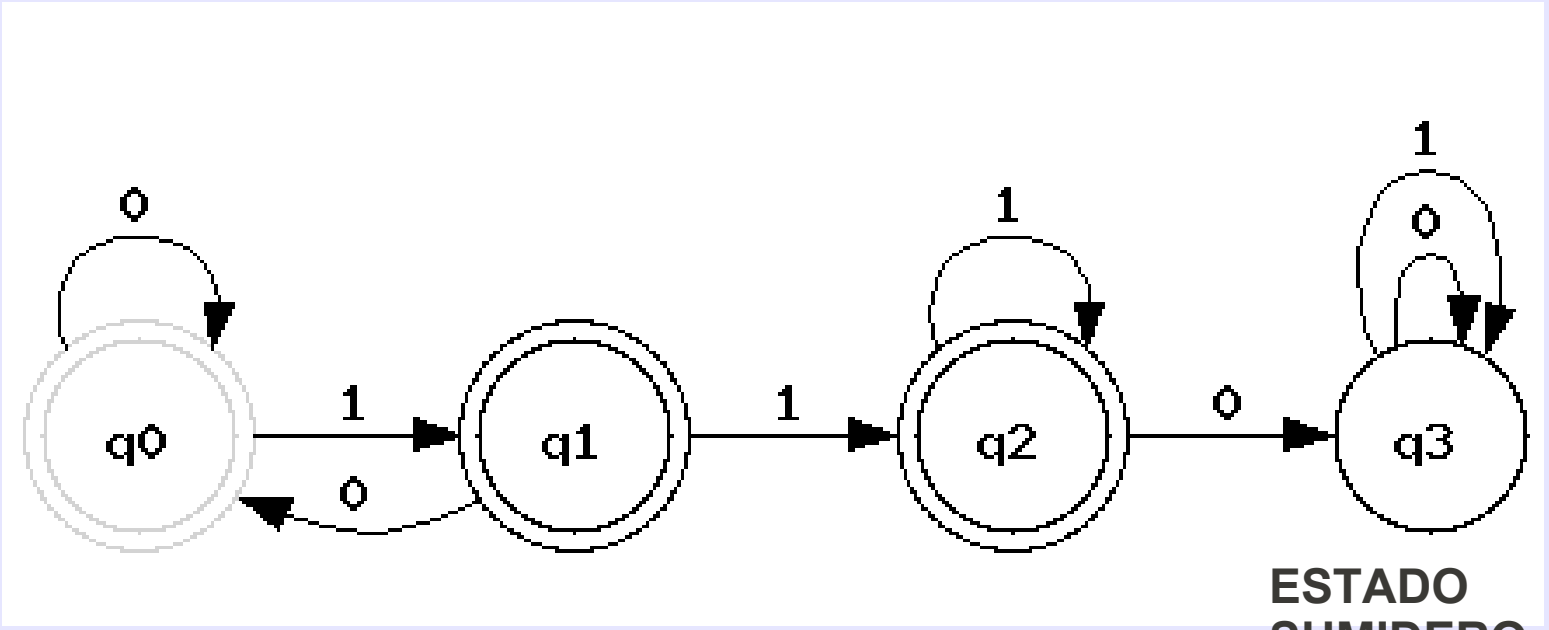
5. EJERCICIOS

1.f). Lenguaje de todas las cadenas que **no contienen** a la subcadena 110.



5. EJERCICIOS

1.f). Lenguaje de todas las cadenas que **no contienen** a la subcadena 110.

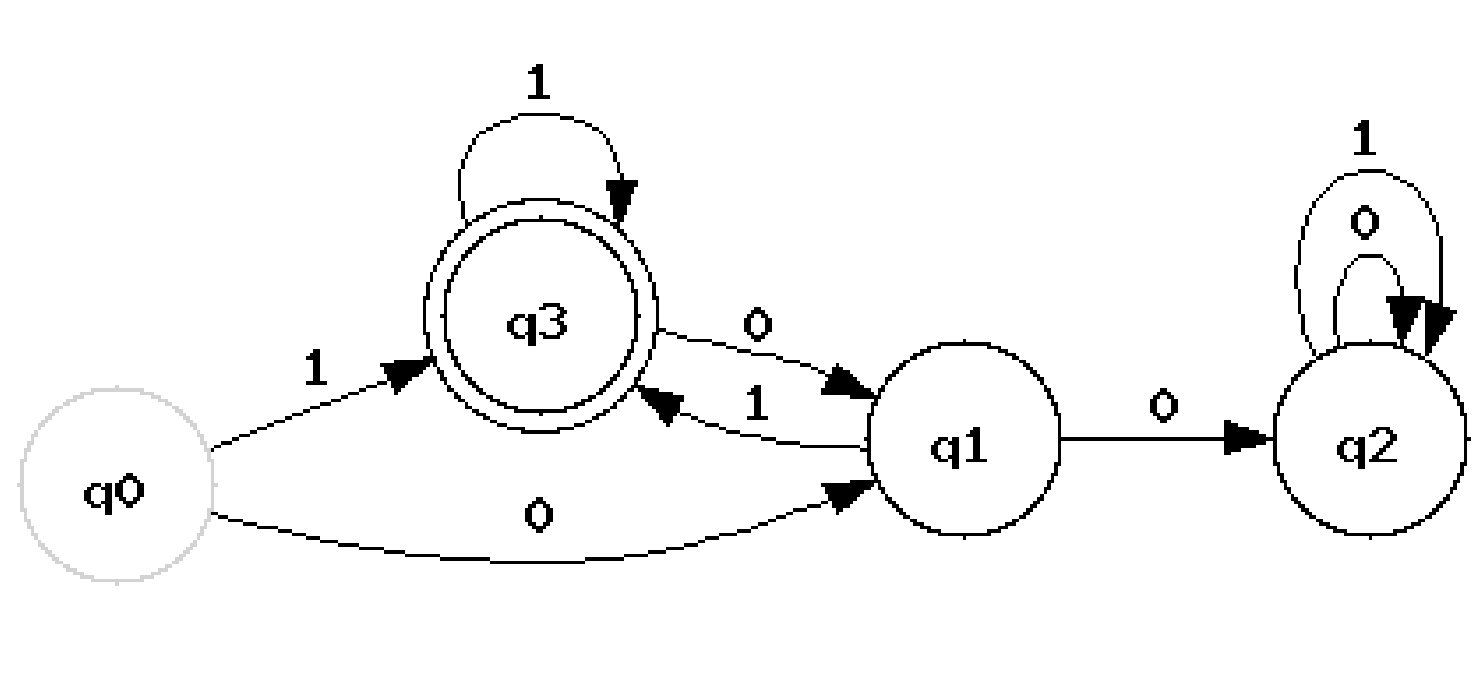


5. EJERCICIOS

1.e). Lenguaje de todas las cadenas que terminen con un 1 y no contenga a la subcadena 00.

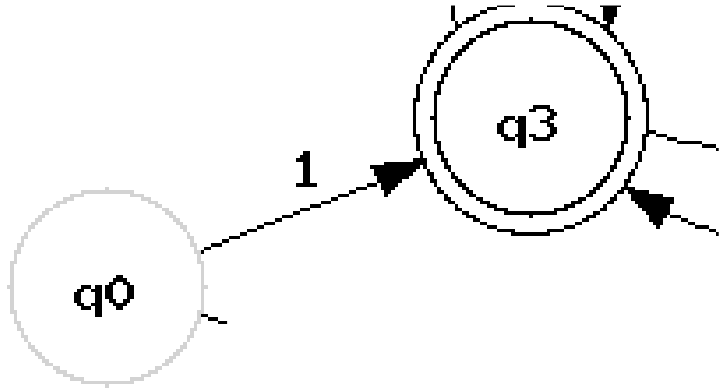
5. EJERCICIOS

1.e). Lenguaje de todas las cadenas que terminen con un 1 y no contenga a la subcadena 00.



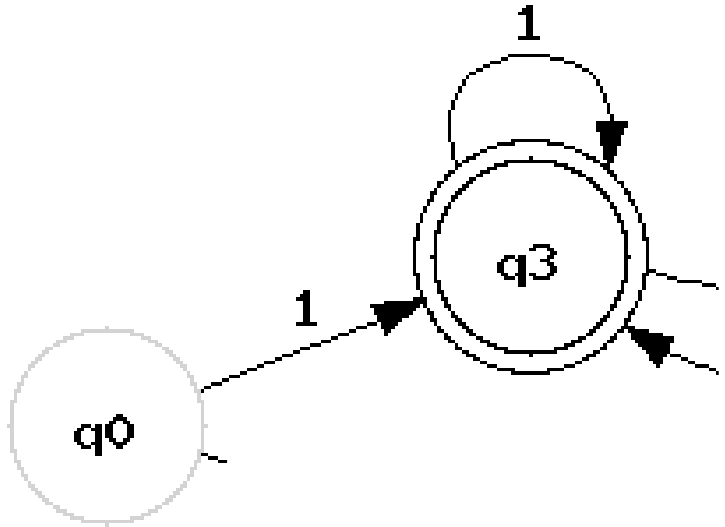
5. EJERCICIOS

1.e). Lenguaje de todas las cadenas que **terminen con un 1** y no contenga a la subcadena 00.



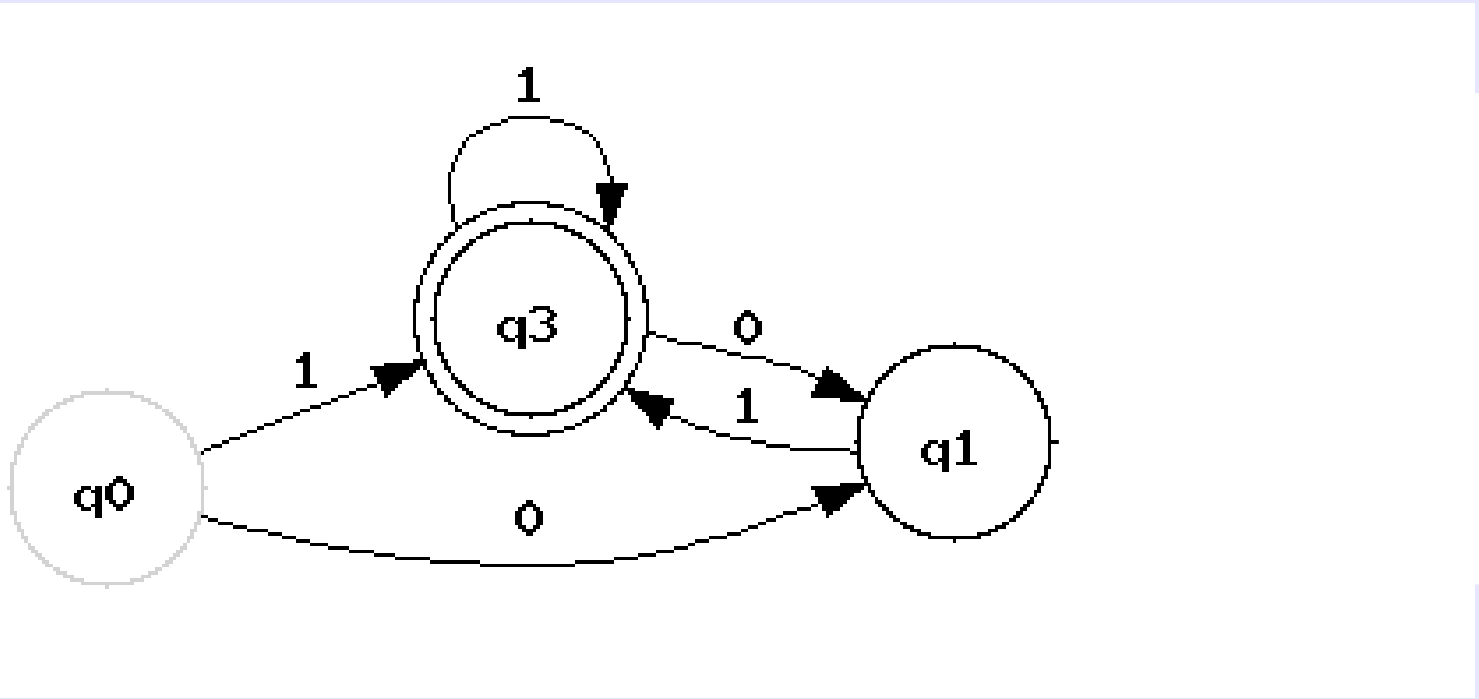
5. EJERCICIOS

1.e). Lenguaje de todas las cadenas que **terminen con un 1** y no contenga a la subcadena 00.



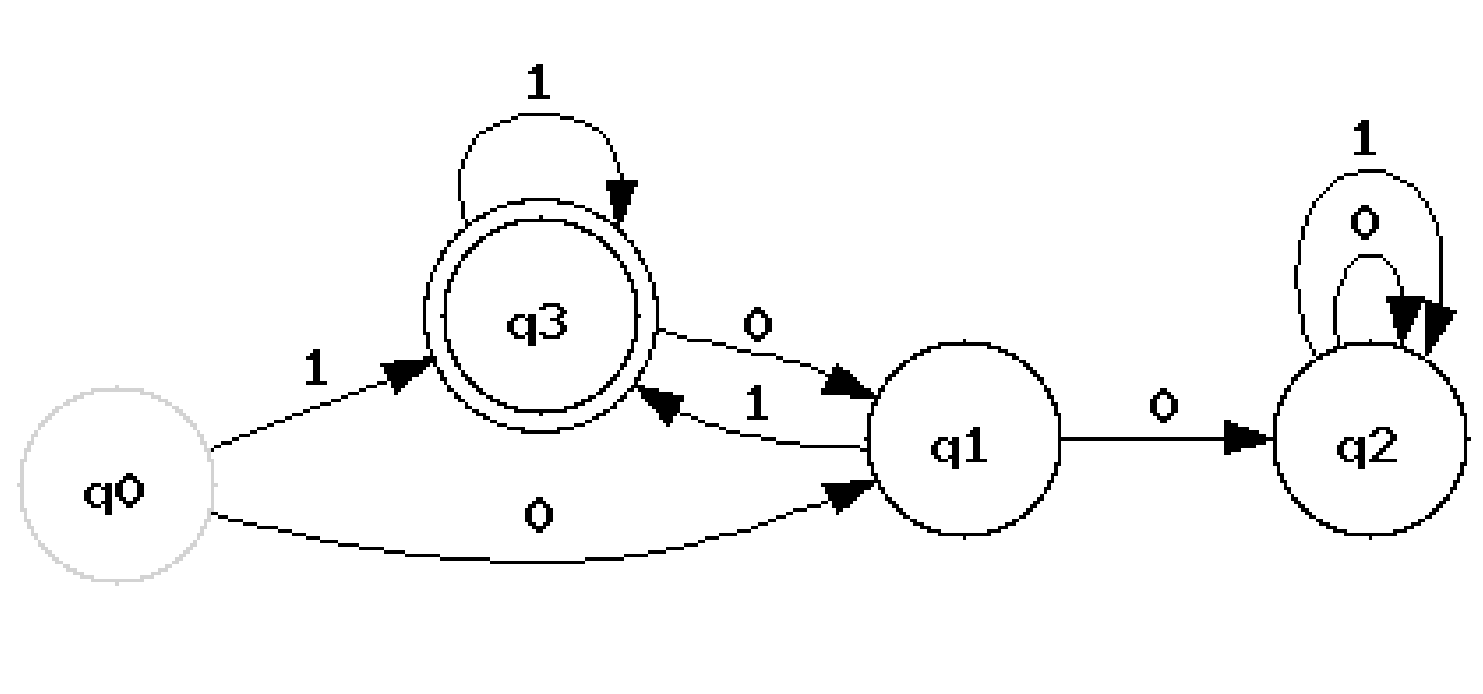
5. EJERCICIOS

1.e). Lenguaje de todas las cadenas que **terminen con un 1** y no contenga a la subcadena 00.



5. EJERCICIOS

1.e). Lenguaje de todas las cadenas que terminen con un 1 y **no contenga a la subcadena 00**.

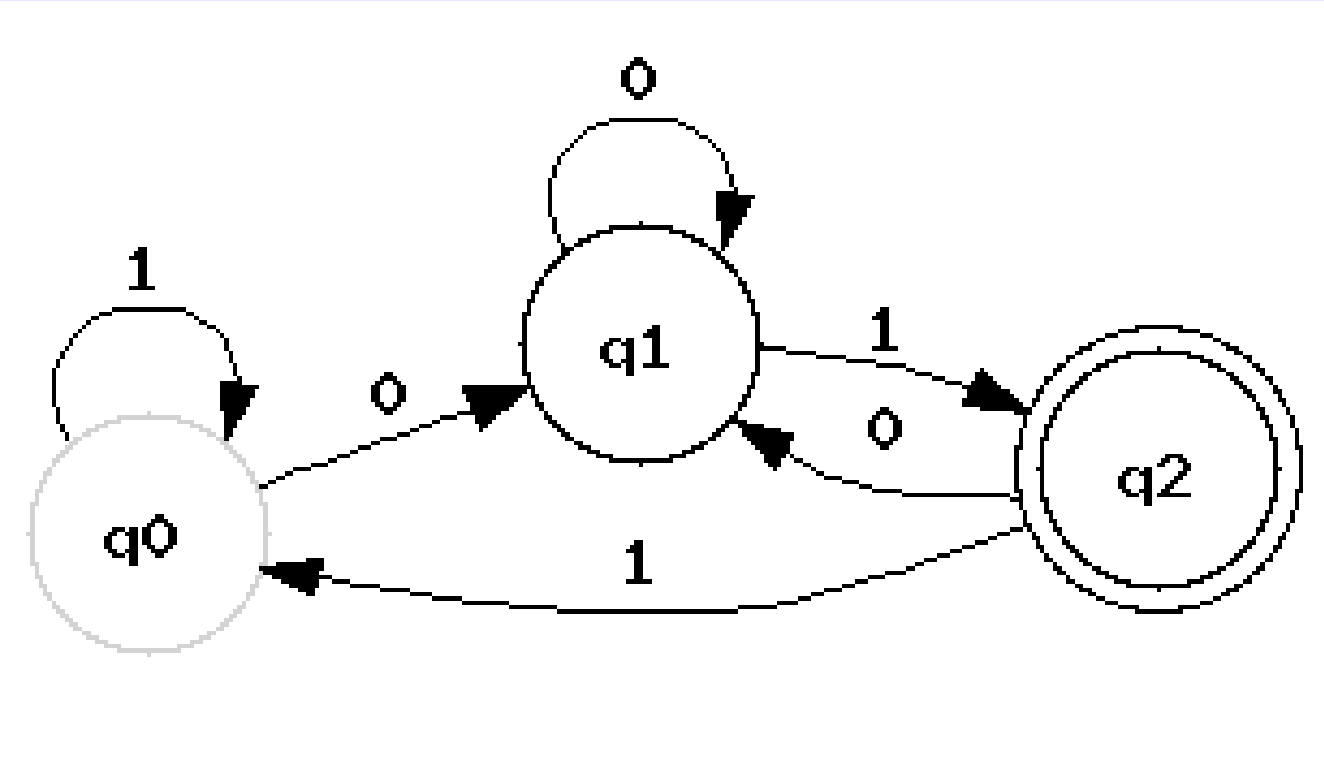


5. EJERCICIOS

1.h) Lenguaje de todas las cadenas que acaban en 01.

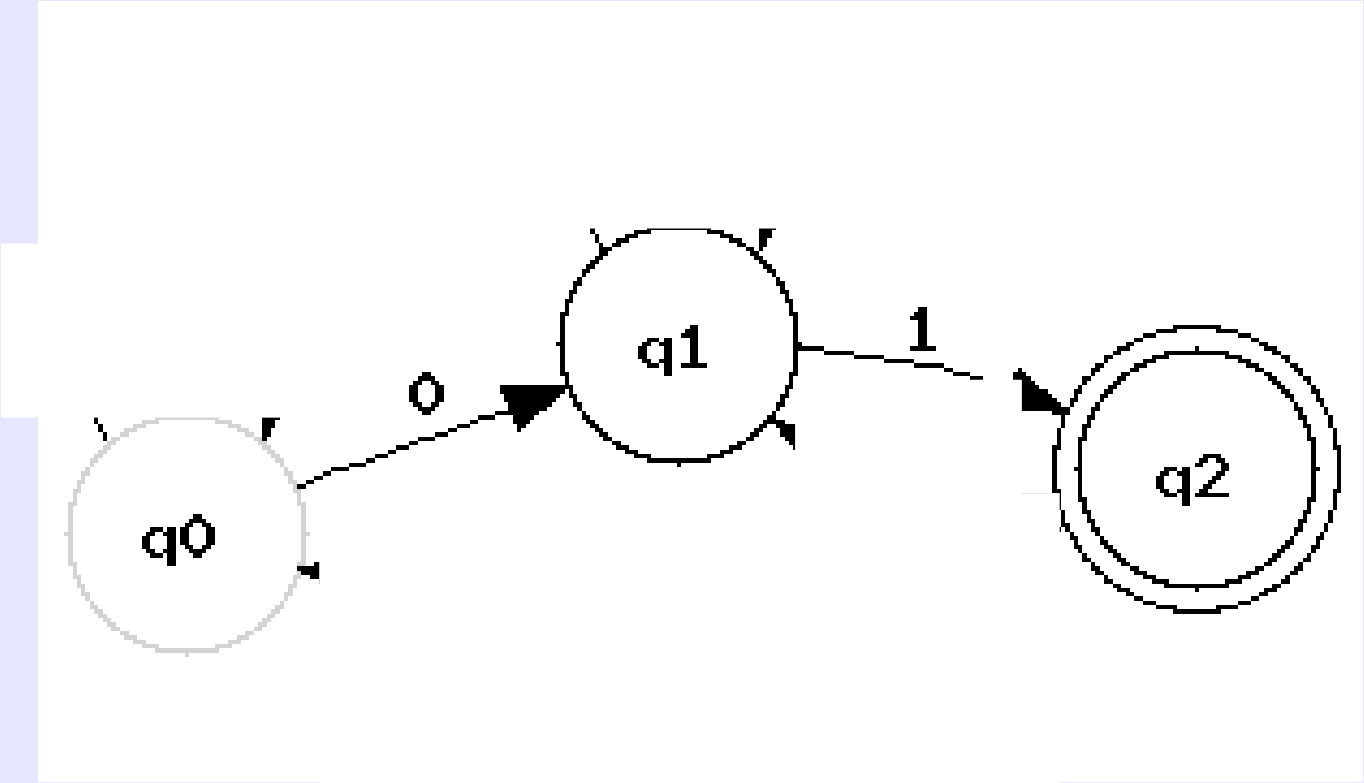
5. EJERCICIOS

1.h) Lenguaje de todas las cadenas que acaban en 01.



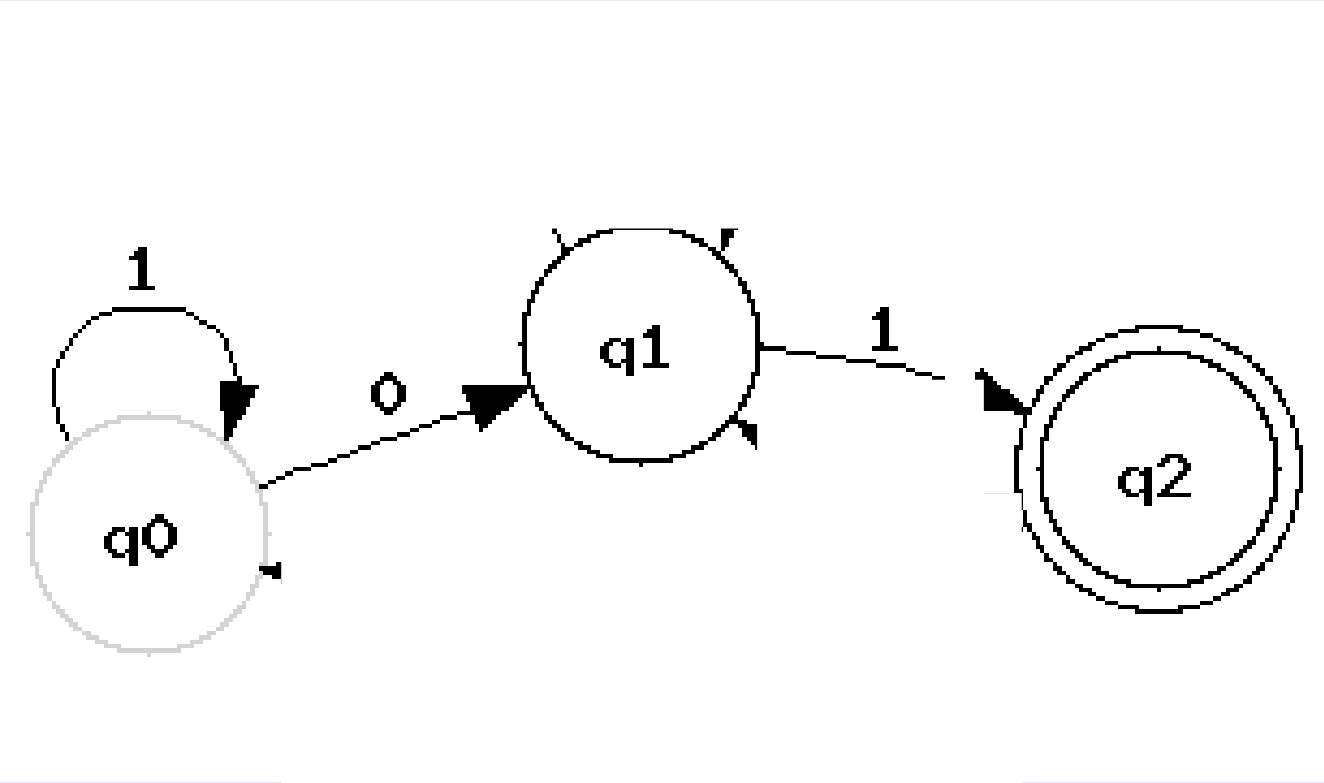
5. EJERCICIOS

1.h) Lenguaje de todas las cadenas que **acaban en 01**.



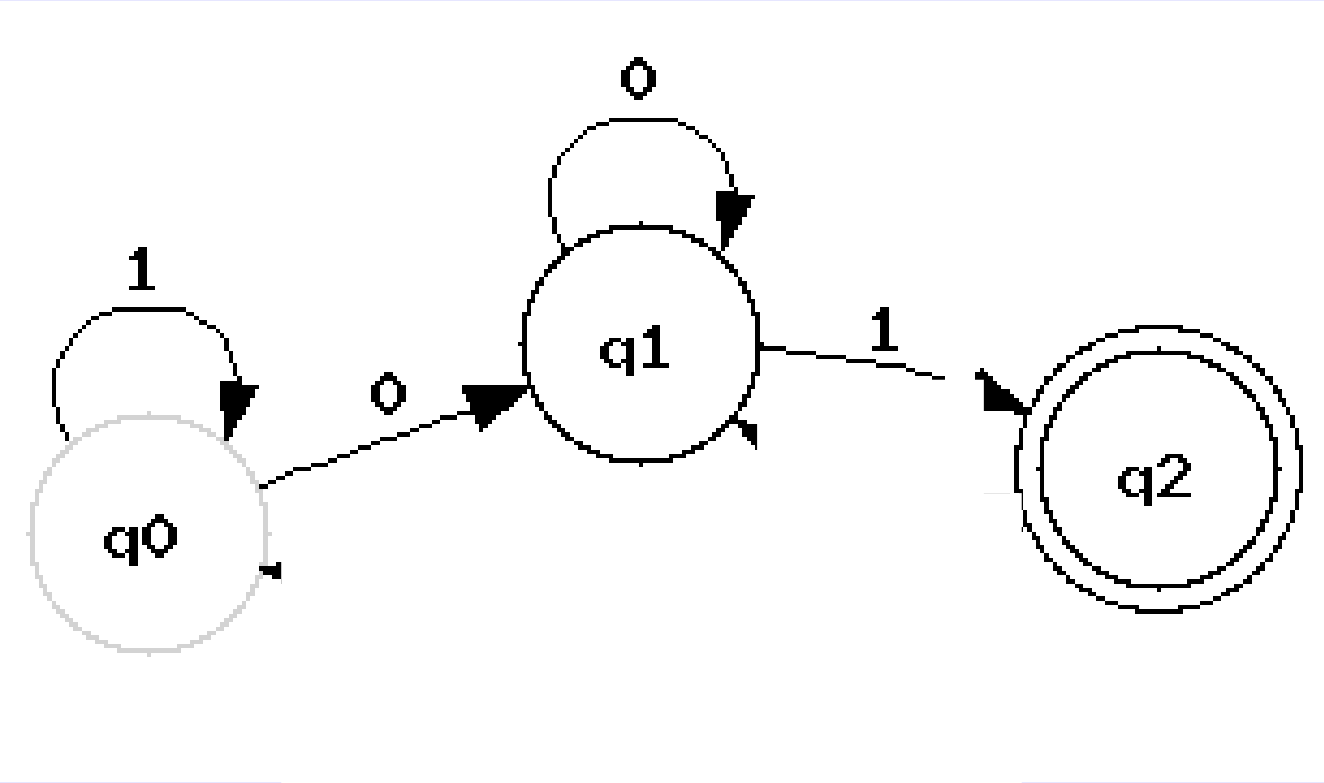
5. EJERCICIOS

1.h) Lenguaje de todas las cadenas que **acaban en 01**.



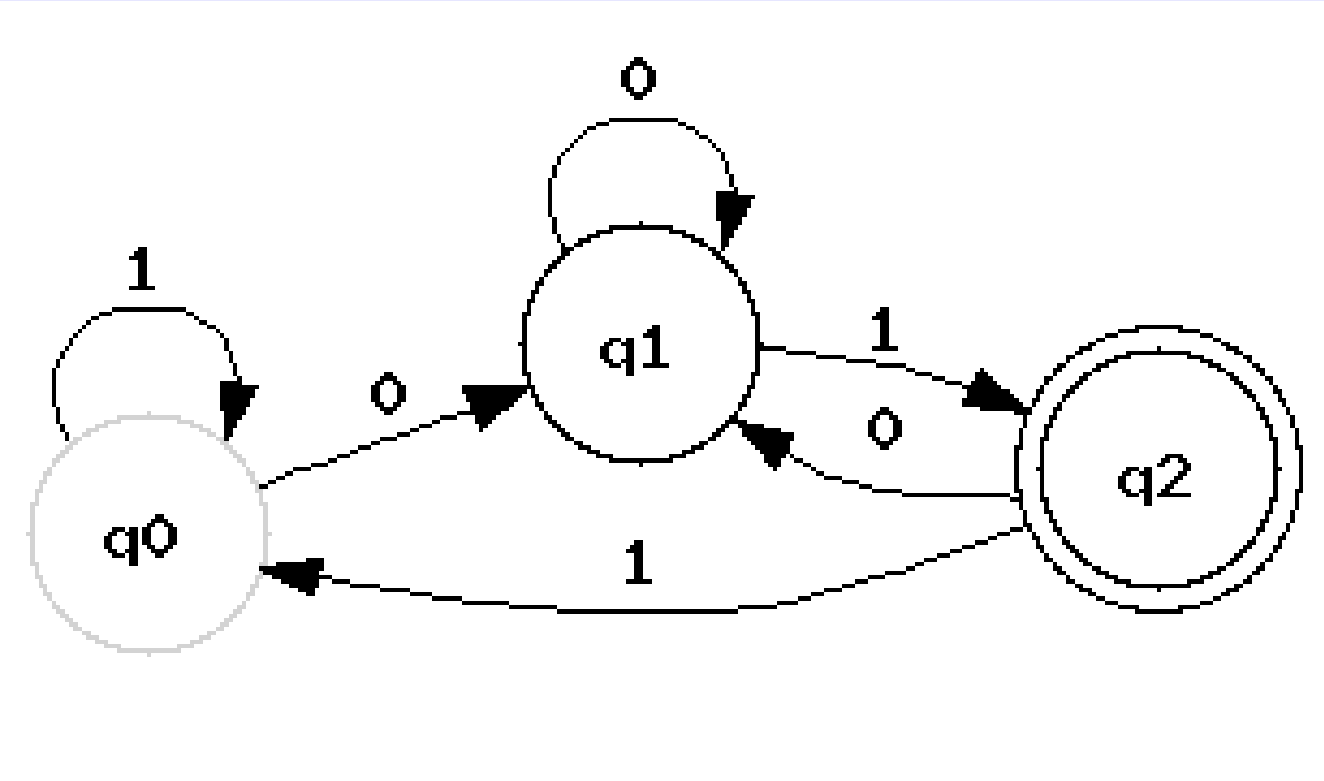
5. EJERCICIOS

1.h) Lenguaje de todas las cadenas que **acaban en 01**.



5. EJERCICIOS

1.h) Lenguaje de todas las cadenas que **acaban en 01**.

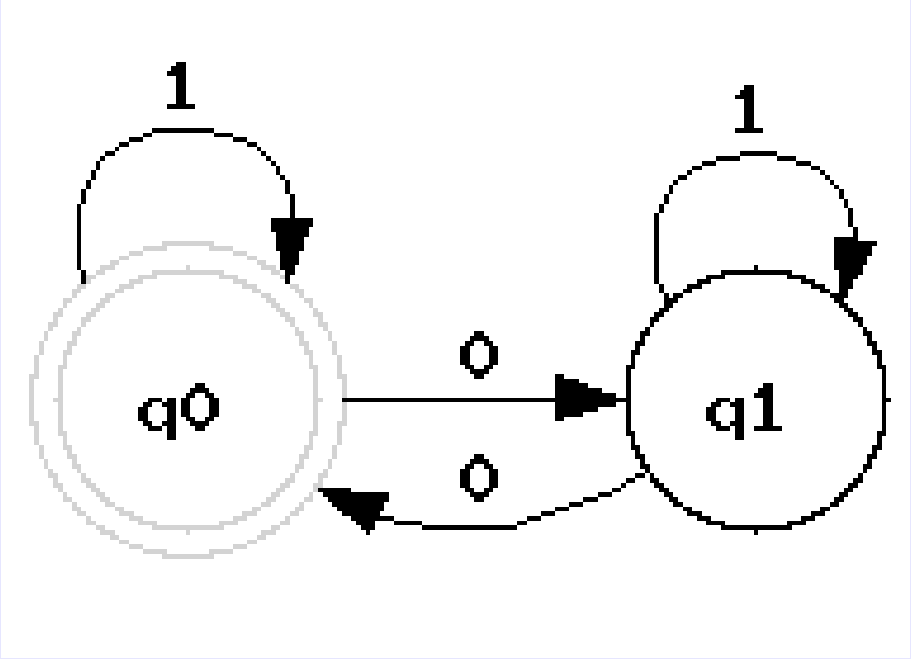


5. EJERCICIOS

1.i) Lenguaje de todas las cadenas que contienen un número par de ceros.

5. EJERCICIOS

1.i) Lenguaje de todas las cadenas que contienen un número par de ceros.

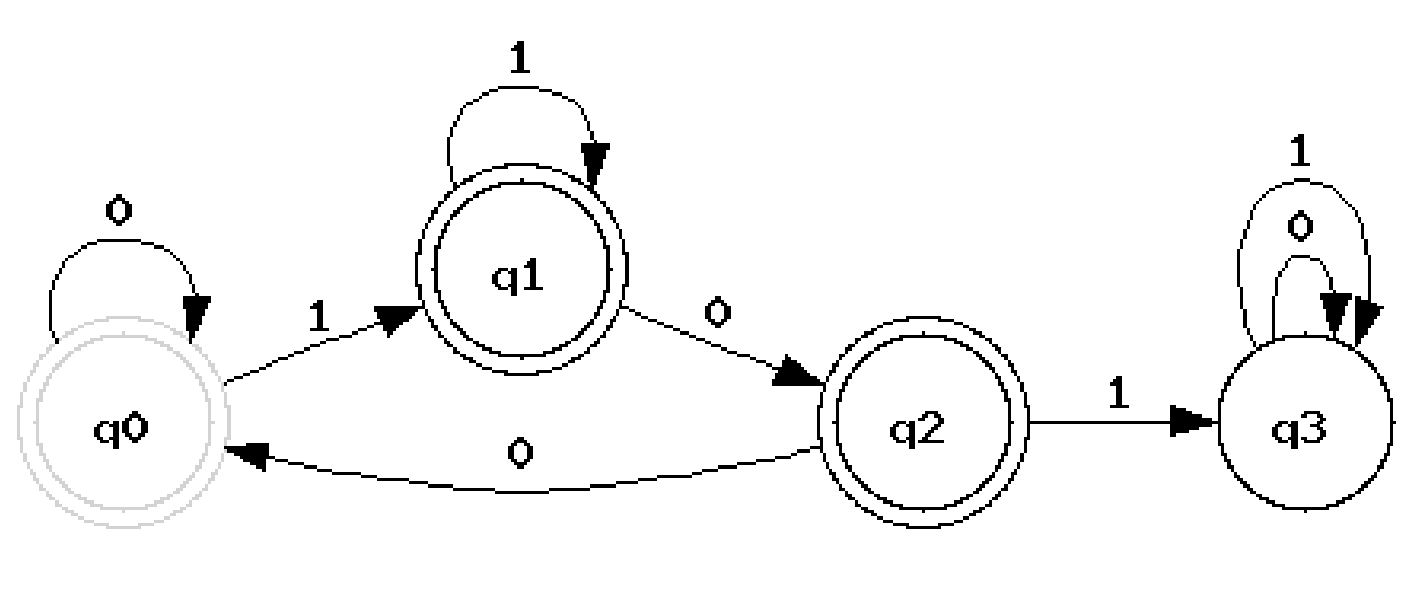


5. EJERCICIOS

1.j) Lenguaje de todas las cadenas que no contienen la subcadena 101.

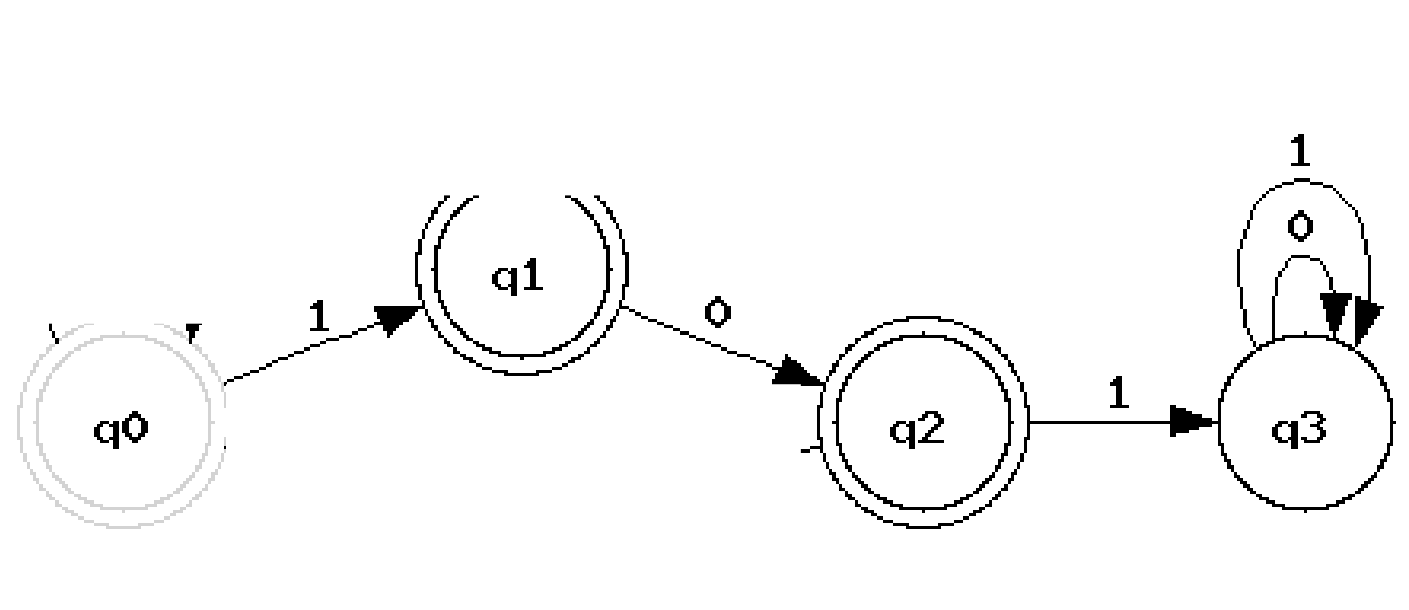
5. EJERCICIOS

1.j) Lenguaje de todas las cadenas que no contienen la subcadena 101.



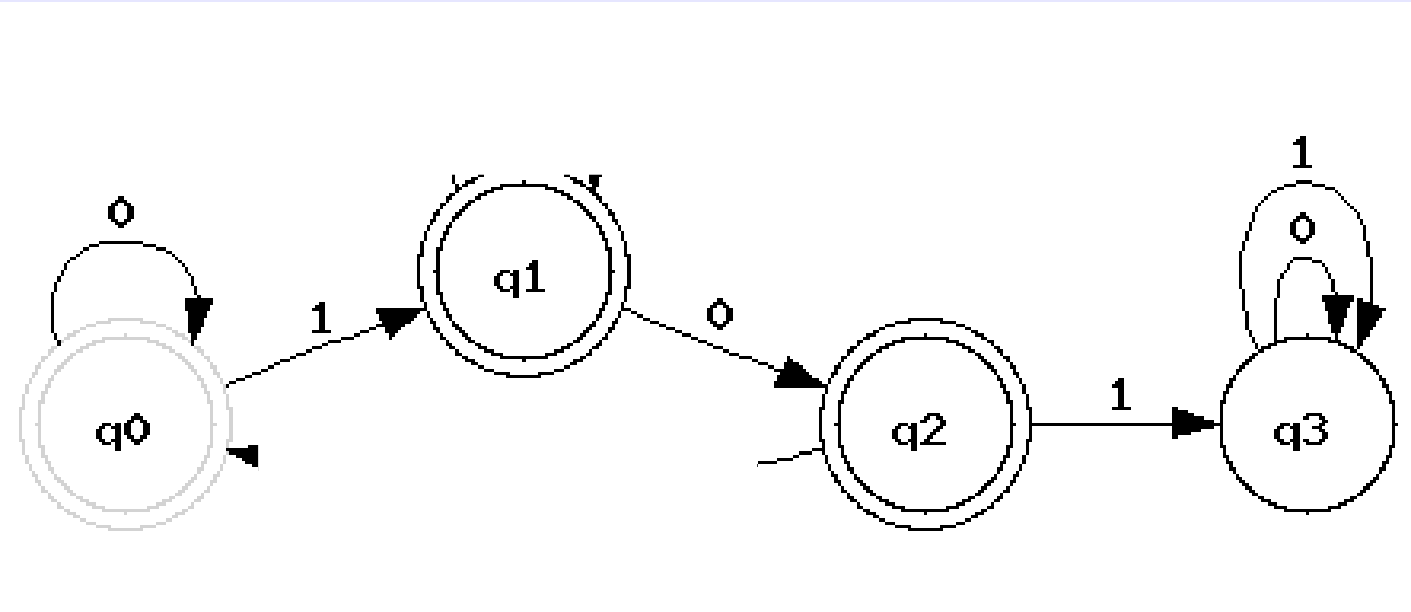
5. EJERCICIOS

1.j) Lenguaje de todas las cadenas que **no contienen la subcadena 101.**



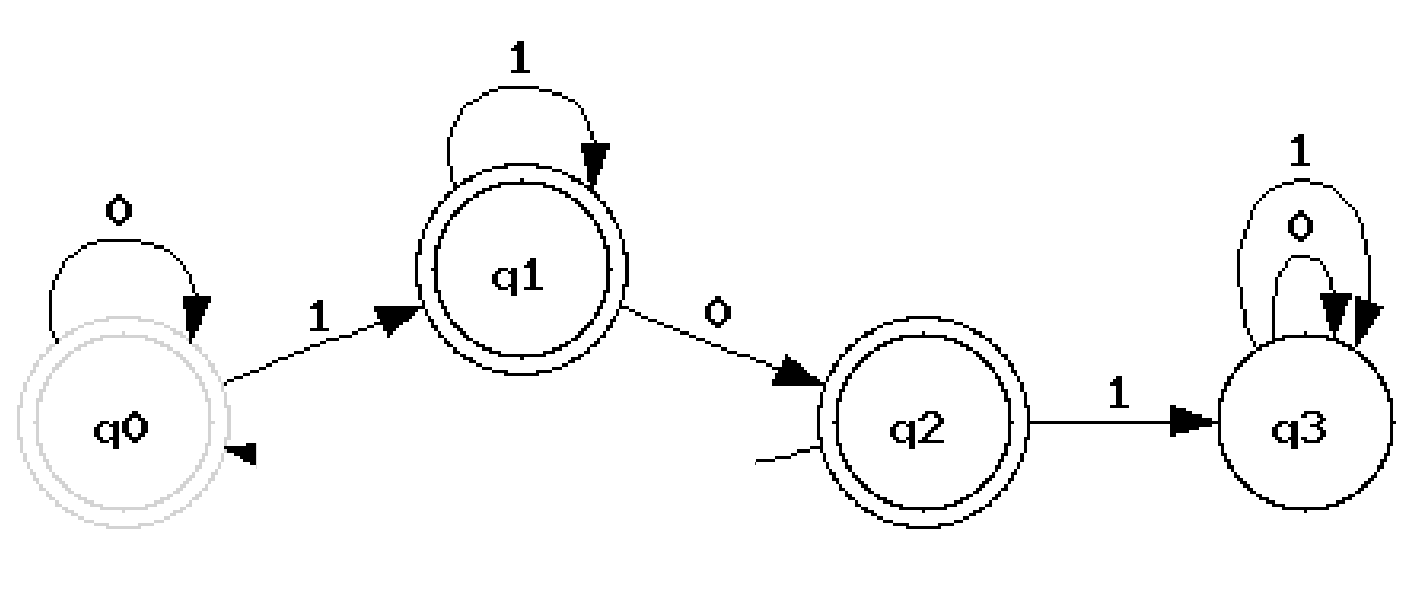
5. EJERCICIOS

1.j) Lenguaje de todas las cadenas que **no contienen la subcadena 101.**



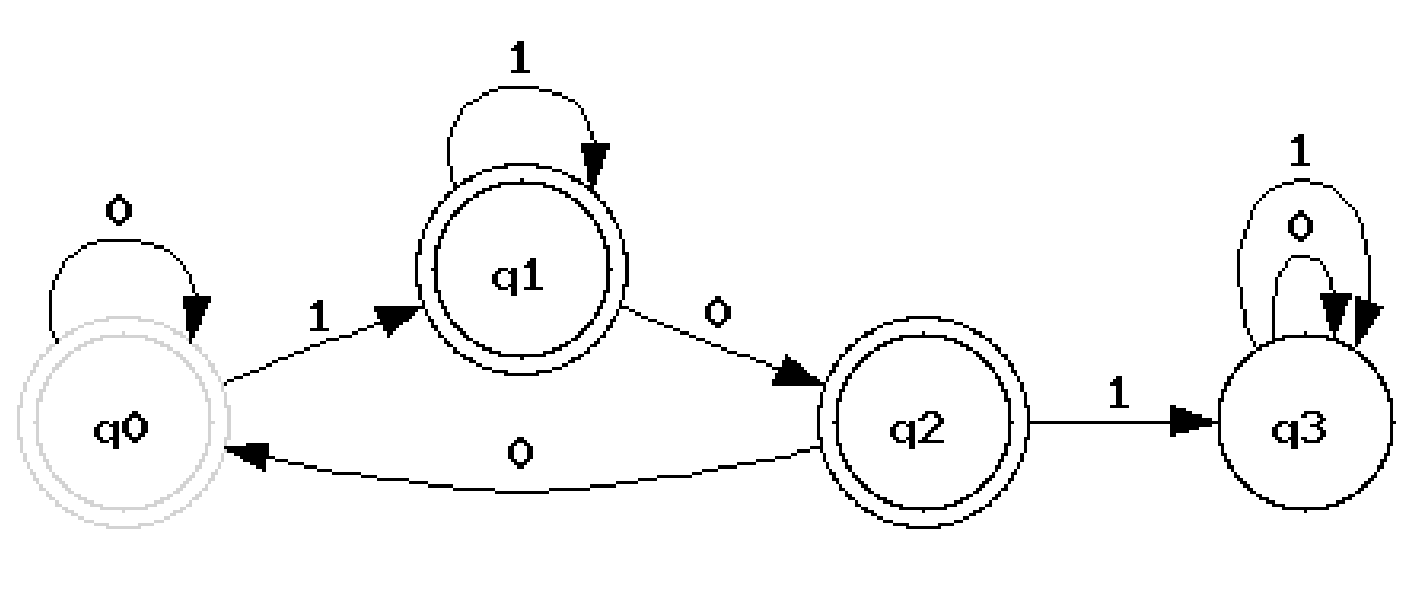
5. EJERCICIOS

1.j) Lenguaje de todas las cadenas que **no contienen la subcadena 101.**



5. EJERCICIOS

1.j) Lenguaje de todas las cadenas que **no contienen la subcadena 101.**



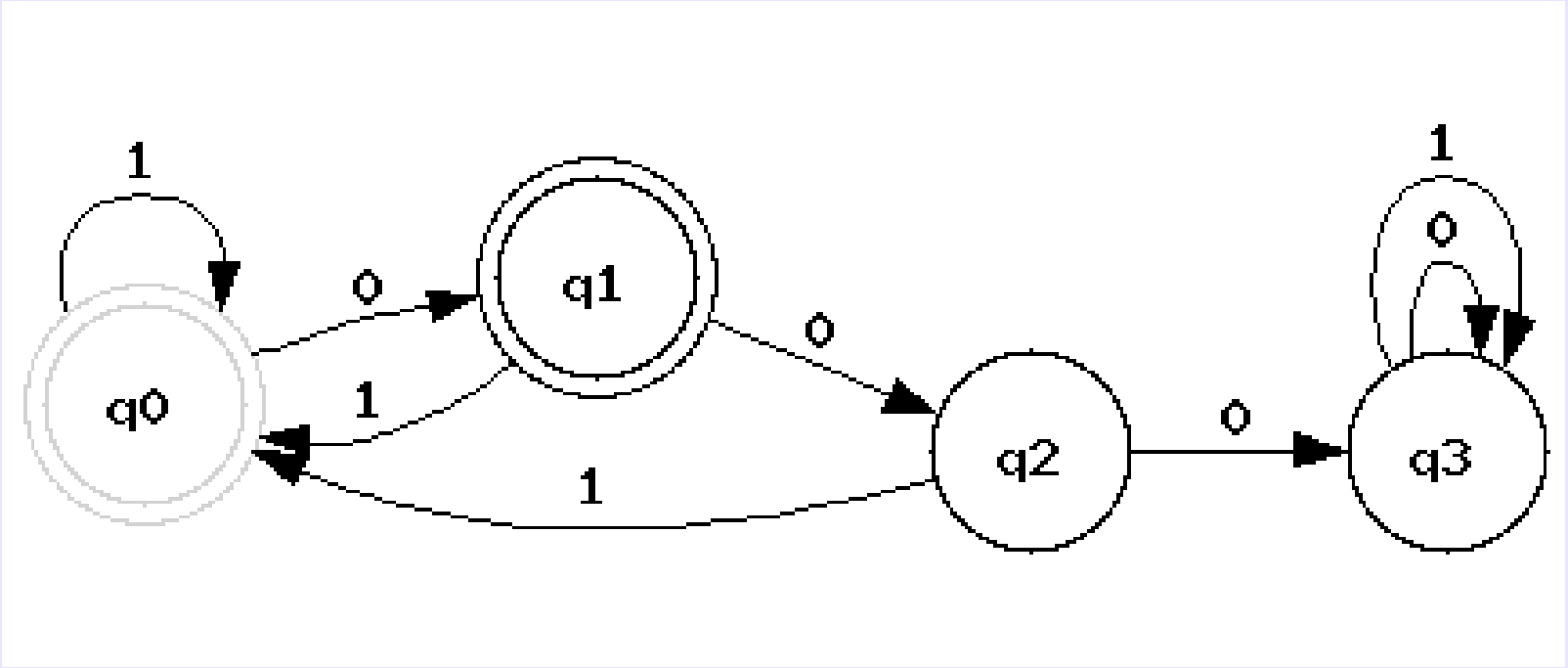
5. EJERCICIOS

1.k) Lenguaje de todas las cadenas en los que la subcadena 00 va seguida siempre por un

1.

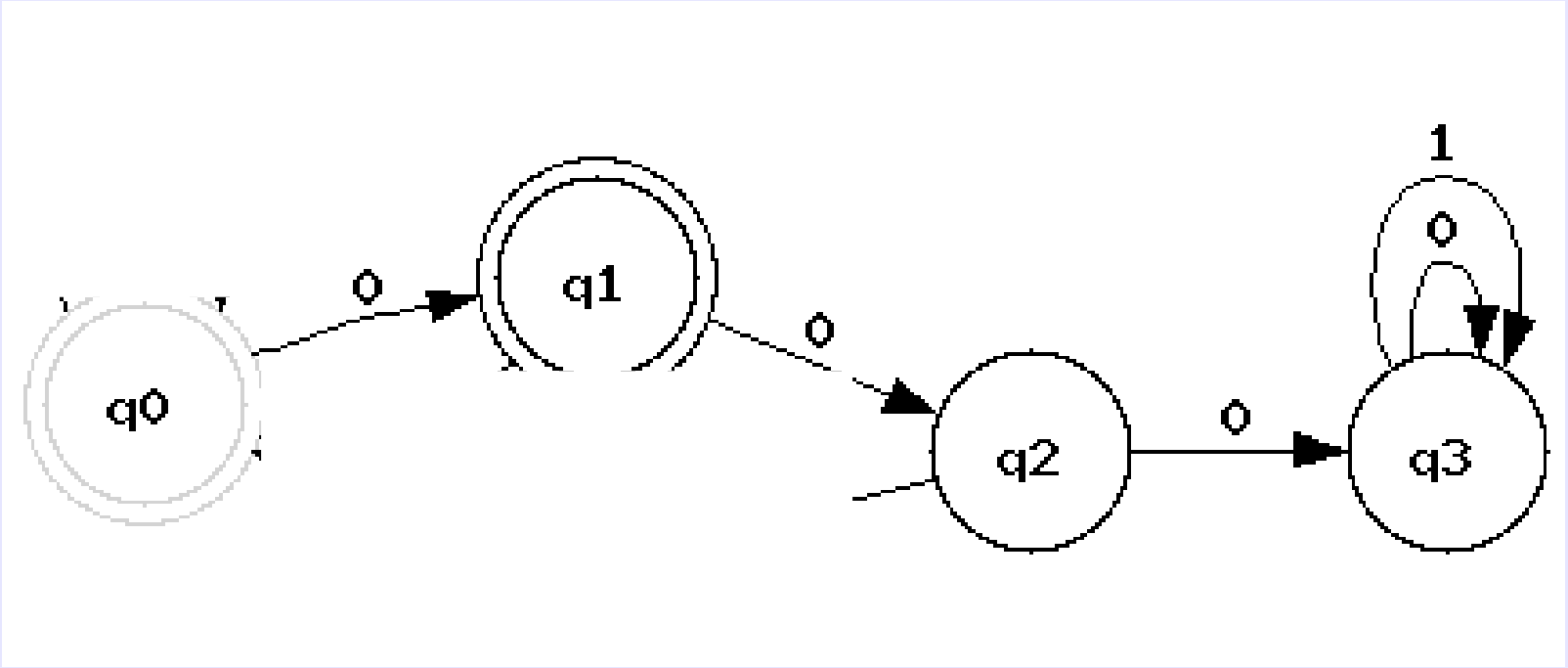
5. EJERCICIOS

1.k) Lenguaje de todas las cadenas en los que la subcadena 00 va seguida siempre por un 1.



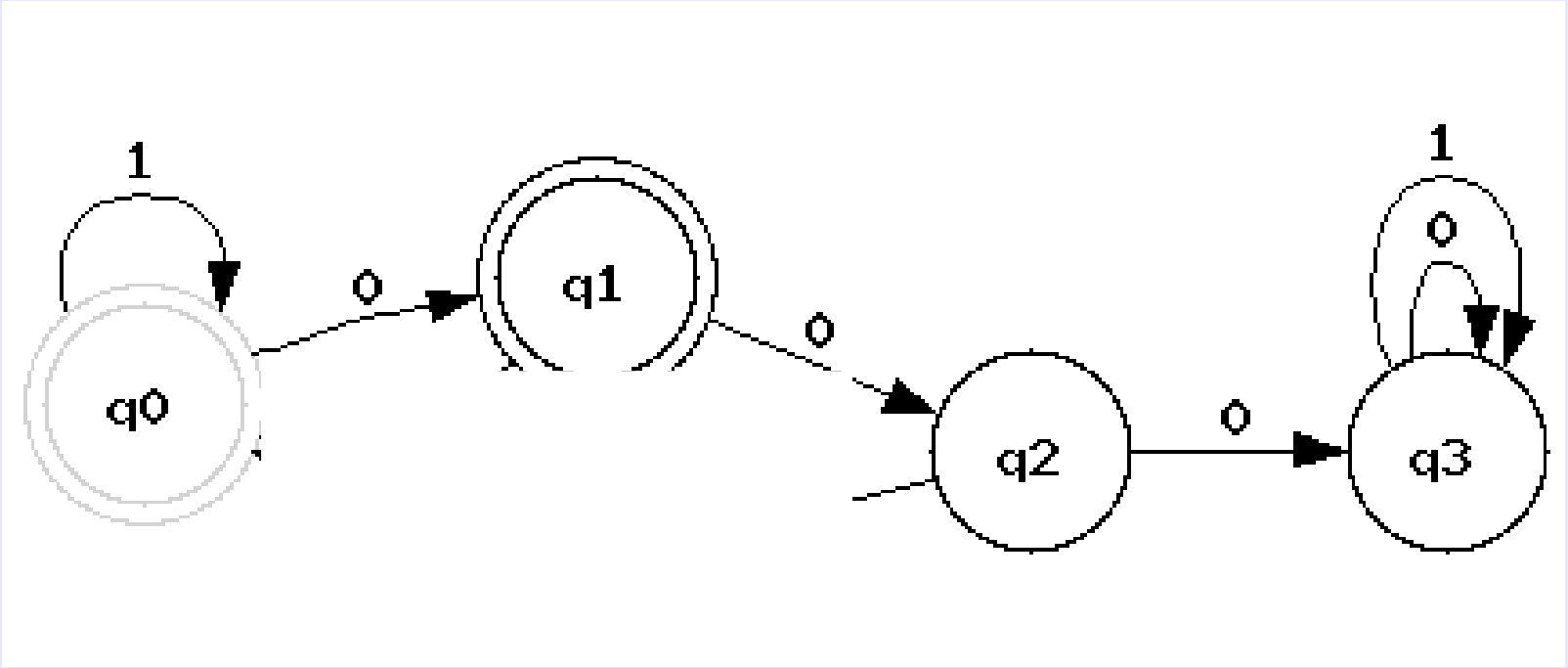
5. EJERCICIOS

1.k) Lenguaje de todas las cadenas en los que la subcadena 00 va seguida siempre por un 1.



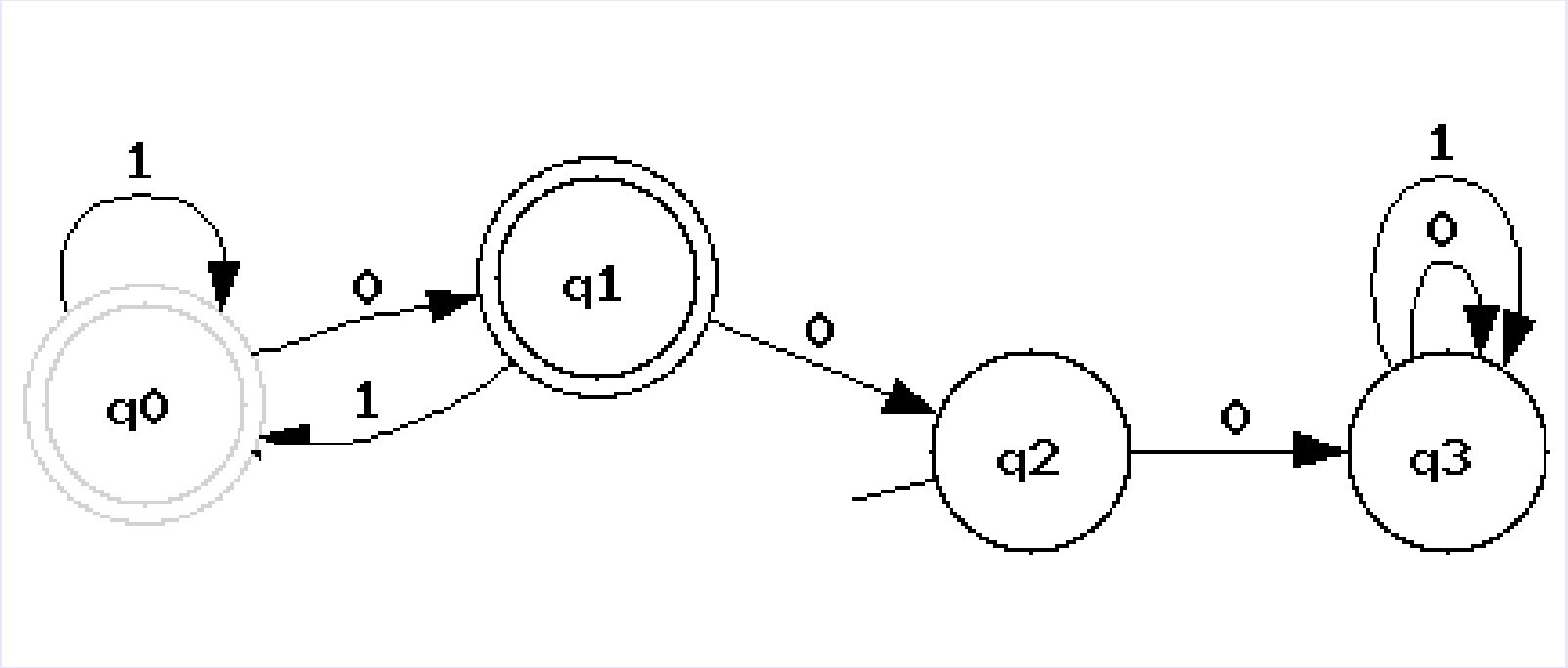
5. EJERCICIOS

1.k) Lenguaje de todas las cadenas en los que la subcadena 00 va seguida siempre por un 1.



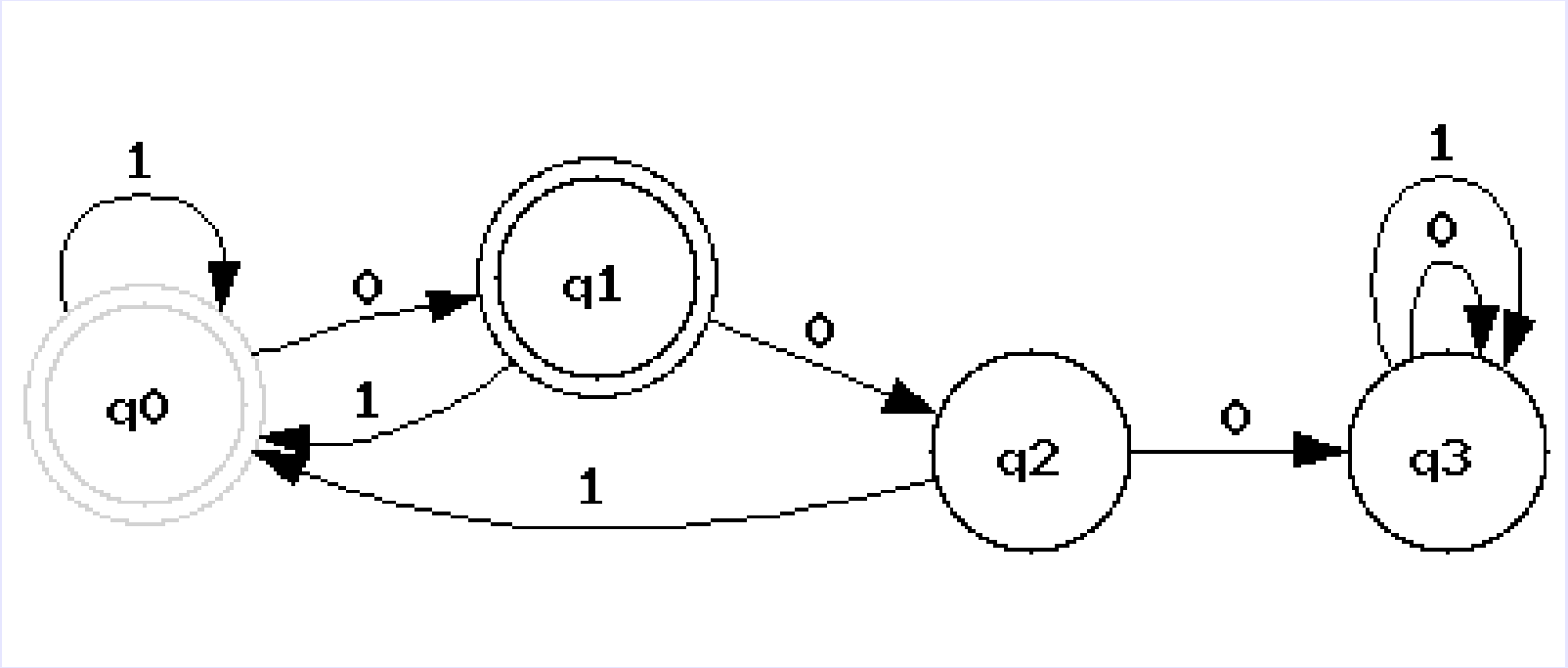
5. EJERCICIOS

1.k) Lenguaje de todas las cadenas en los que la subcadena 00 va seguida siempre por un 1.



5. EJERCICIOS

1.k) Lenguaje de todas las cadenas en los que la subcadena 00 va seguida siempre por un 1.



5. EJERCICIOS

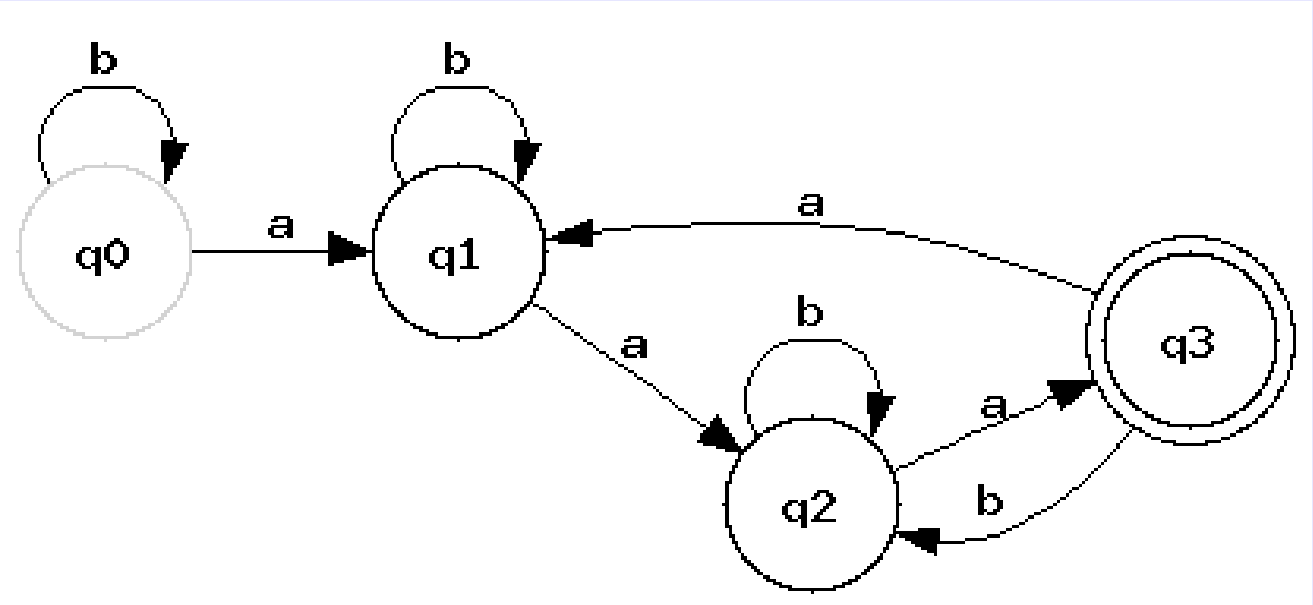
2.h Número de aes congruente con 1 mod 3:

a mod 3 = 1

5. EJERCICIOS

2.h Número de aes congruente con 1 mod 3:

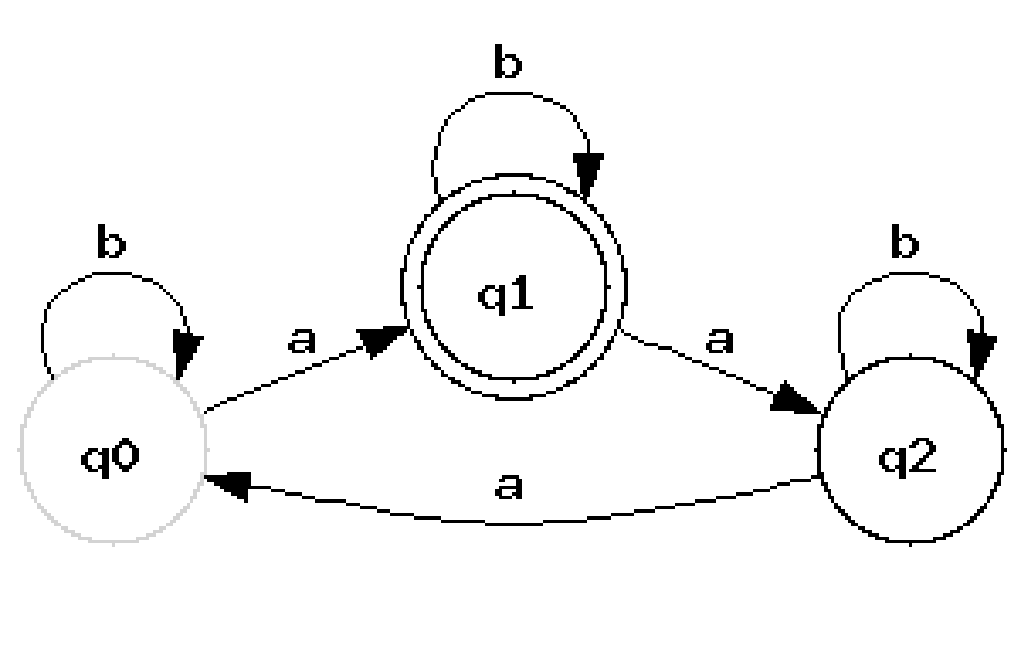
a mod 3 = 1



5. EJERCICIOS

2.h Número de aes congruente con 1 mod 3:

a mod 3 = 1

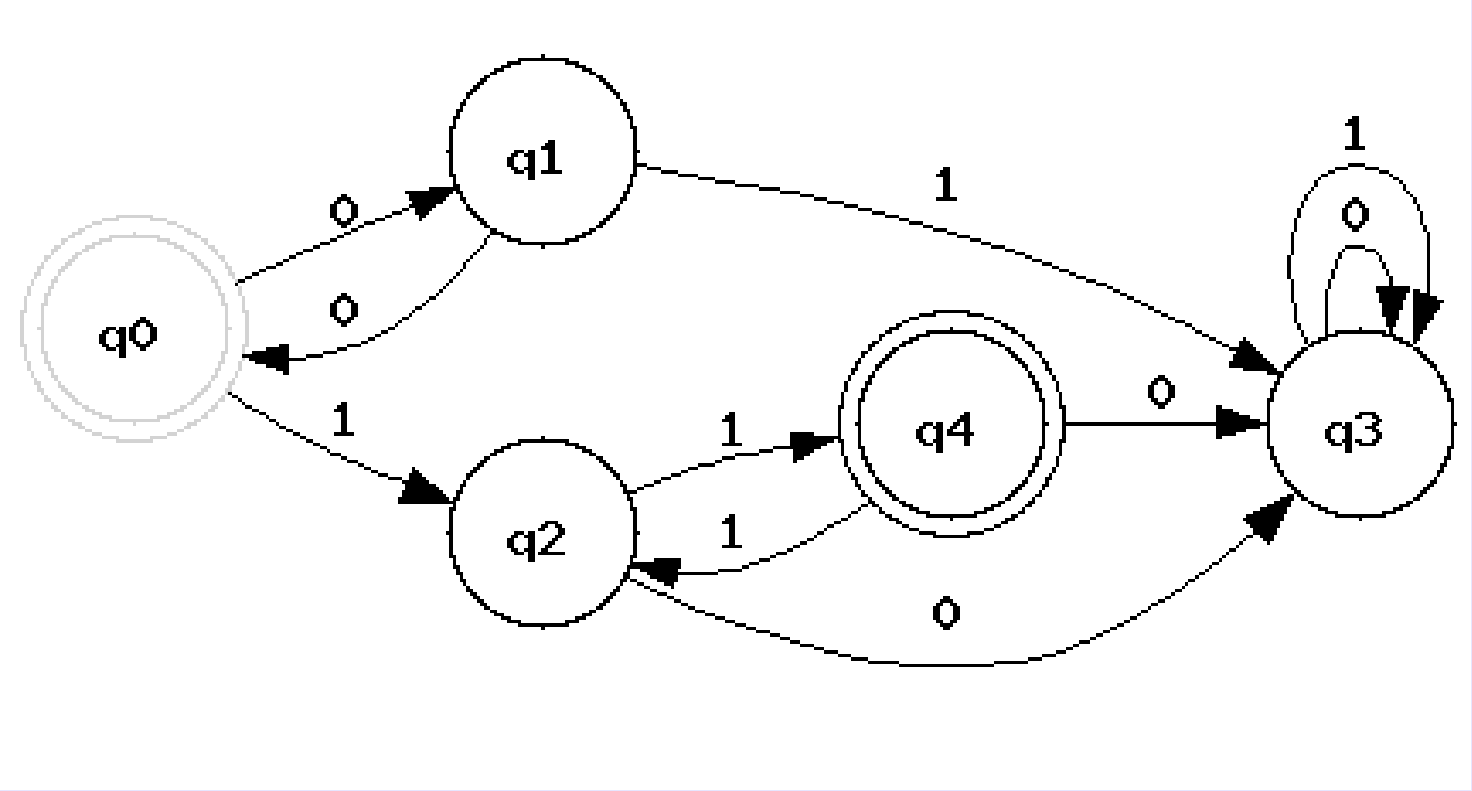


5. EJERCICIOS

(00)*(11)*

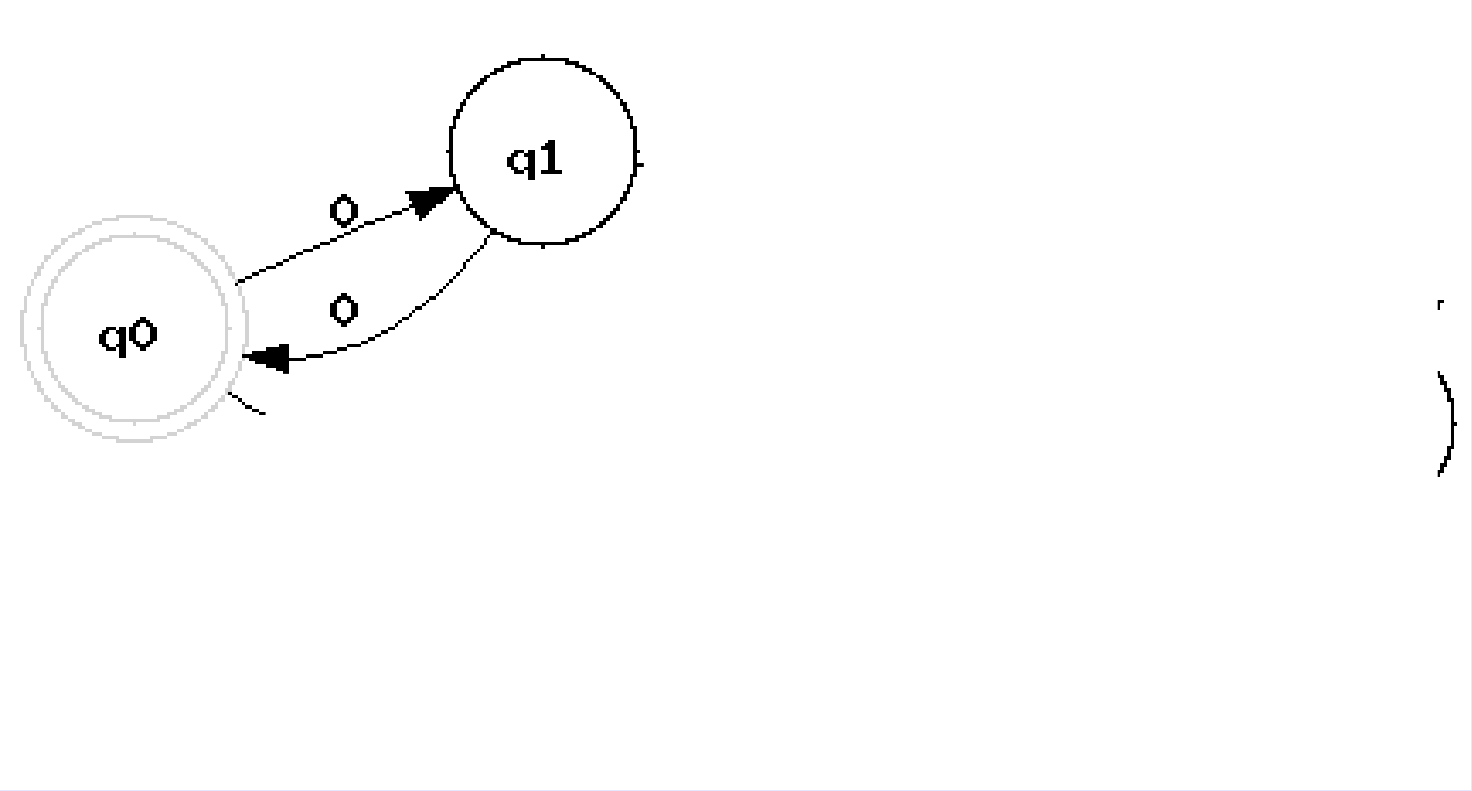
5. EJERCICIOS

(00)*(11)*



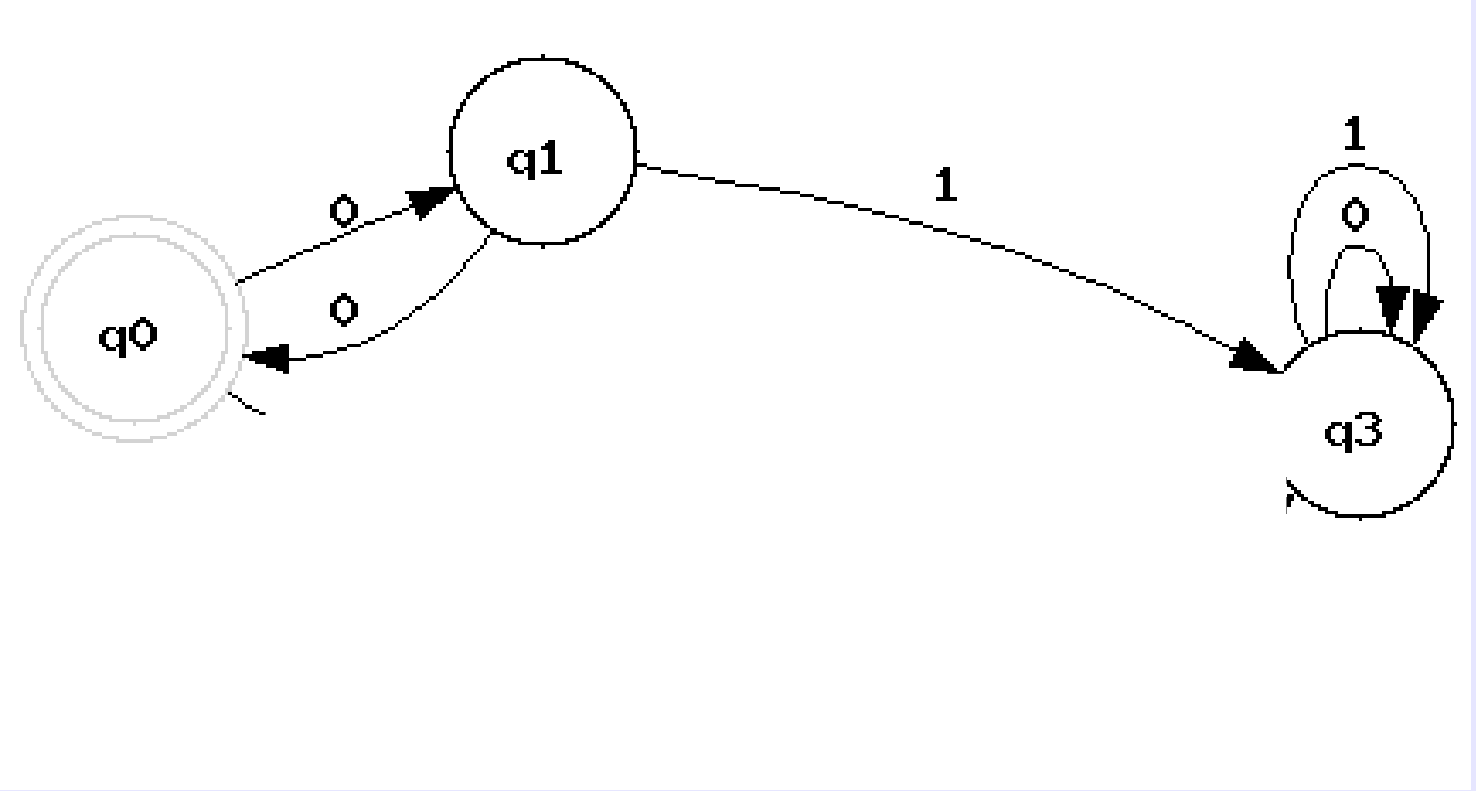
5. EJERCICIOS

(00)*(11)*



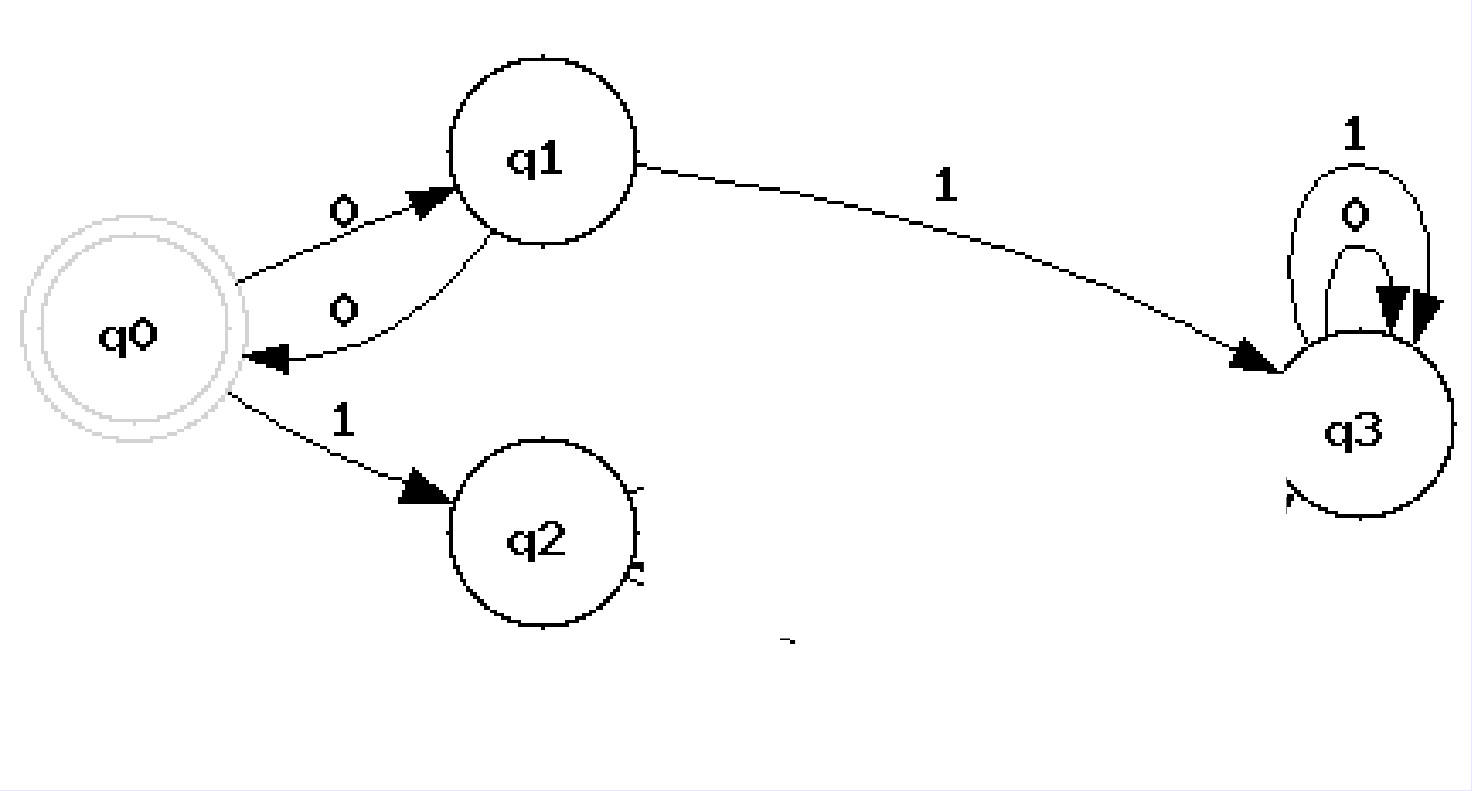
5. EJERCICIOS

(00)*(11)*



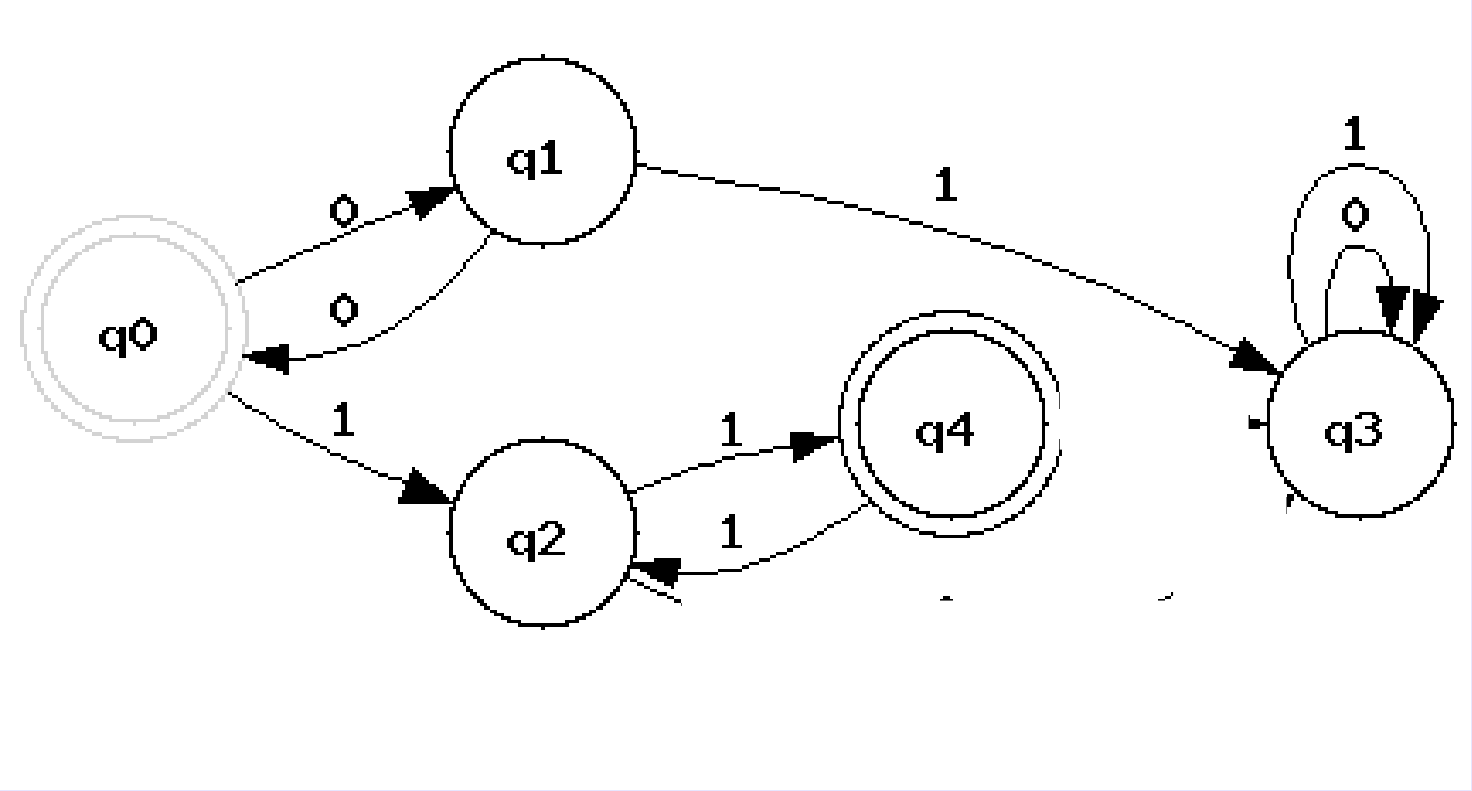
5. EJERCICIOS

(00)*(11)*



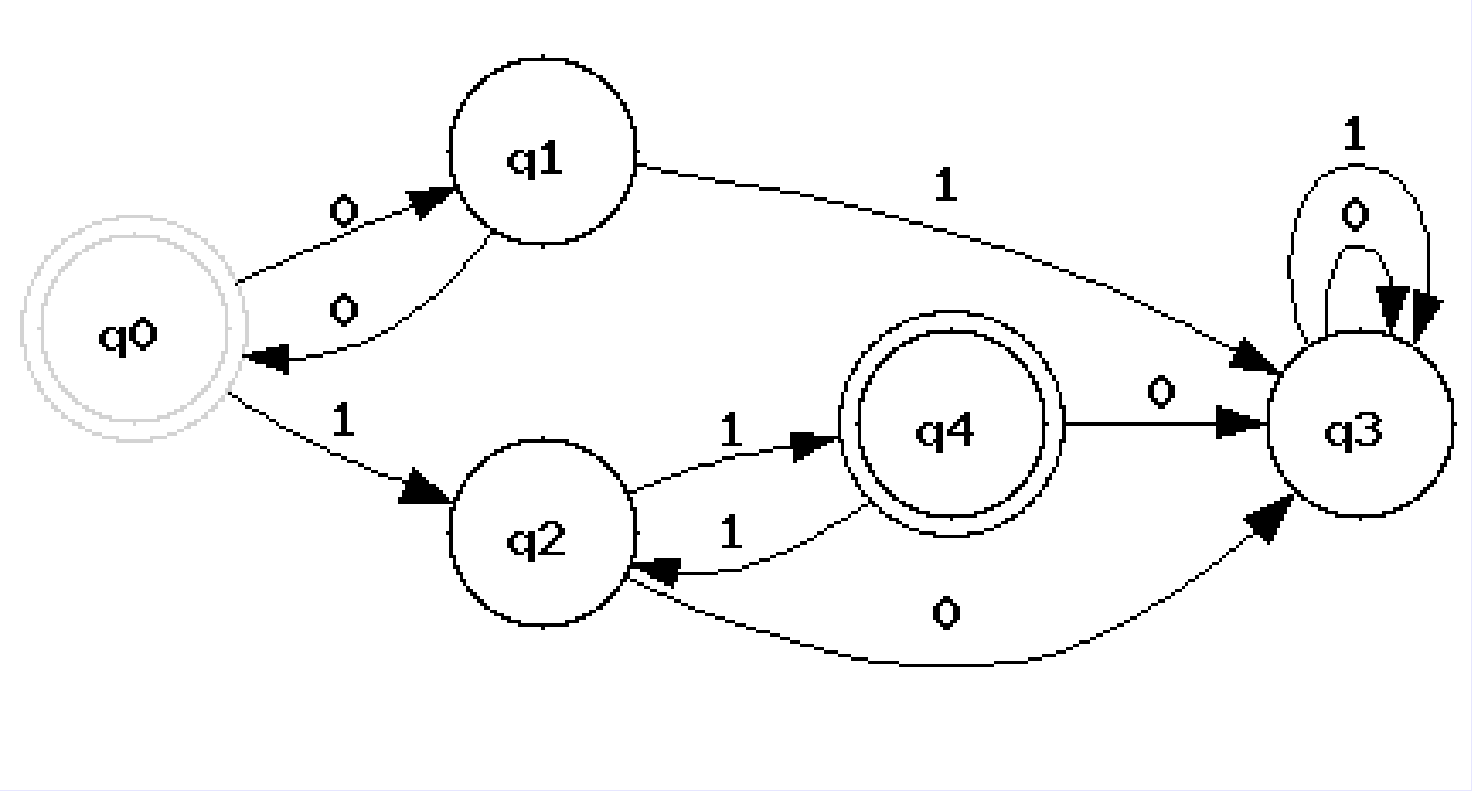
5. EJERCICIOS

(00)*(11)*



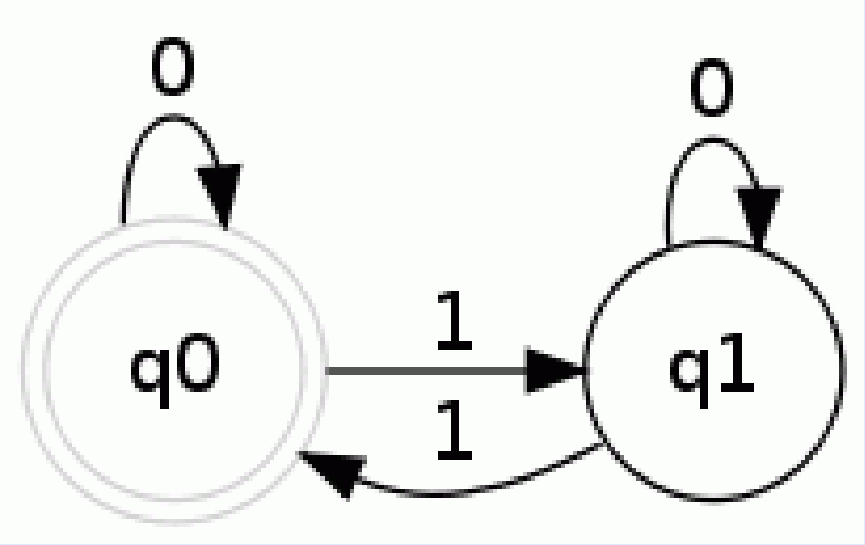
5. EJERCICIOS

(00)*(11)*



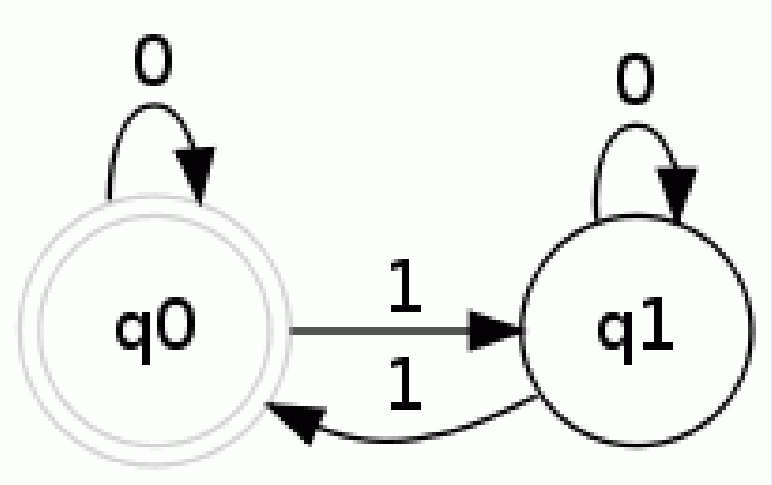
5. EJERCICIOS

2.) Explicar cual es el lenguaje asociado a cada uno de los siguientes autómatas.



5. EJERCICIOS

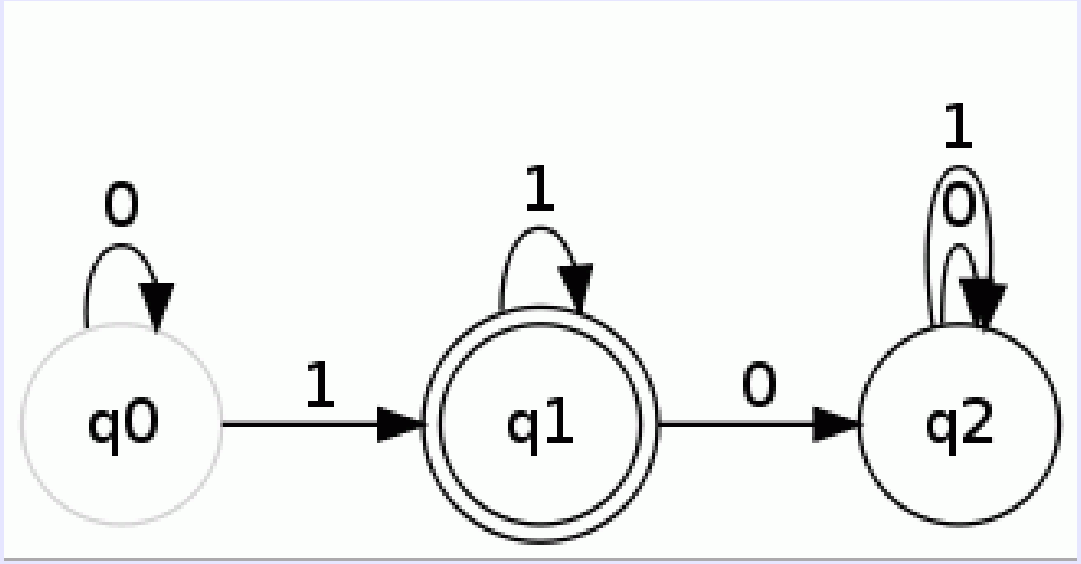
2.) Explicar cual es el lenguaje asociado a cada uno de los siguientes autómatas.



L= { x, contiene un número par de unos }

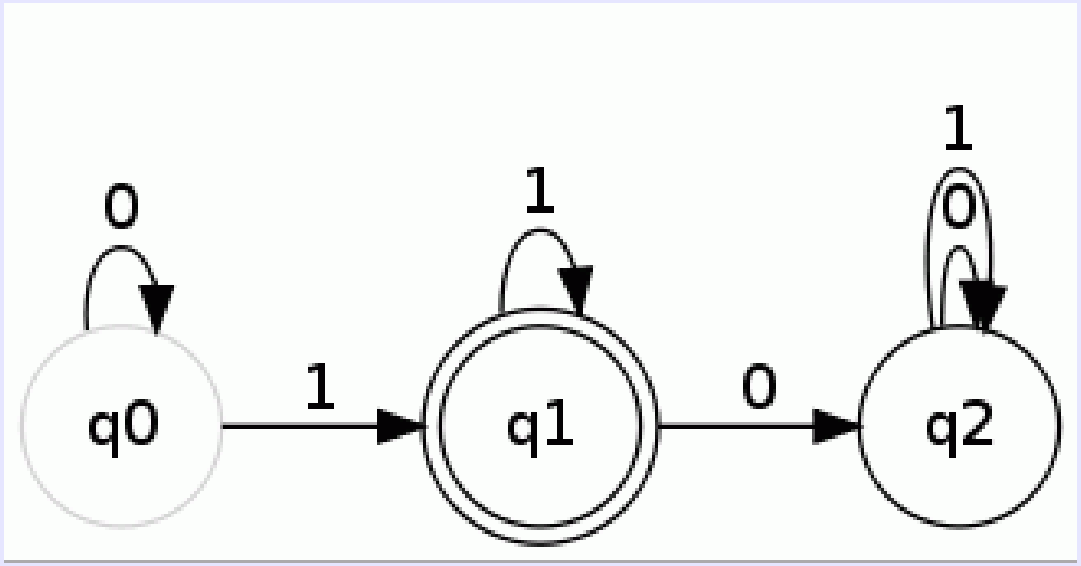
5. EJERCICIOS

2.) Explicar cual es el lenguaje asociado a cada uno de los siguientes autómatas.



5. EJERCICIOS

2.) Explicar cual es el lenguaje asociado a cada uno de los siguientes autómatas.

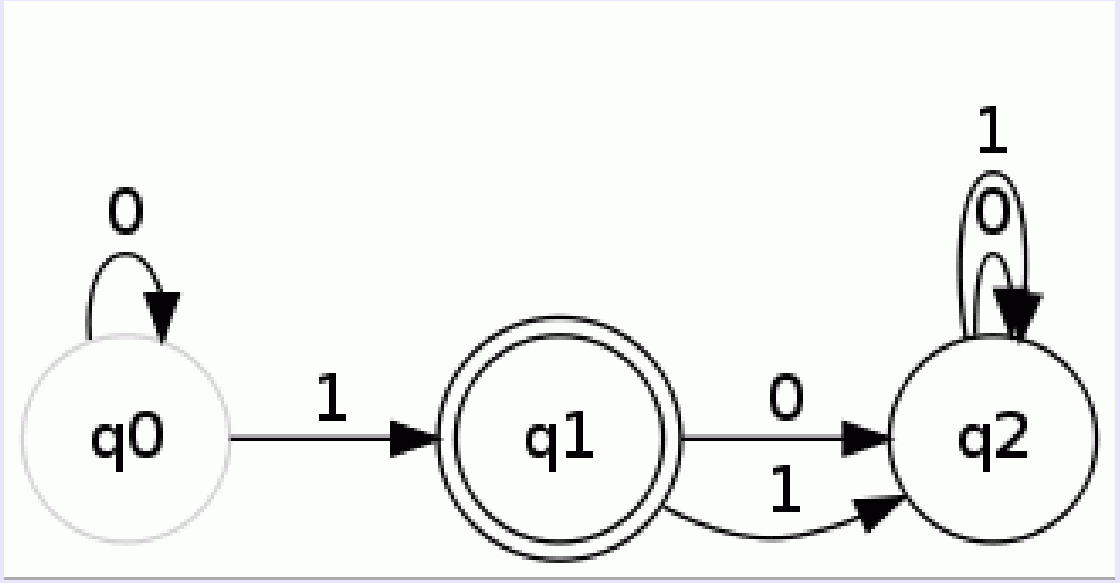


$L = 0^*1^+$

L = {x | x no contiene la cadena 10}

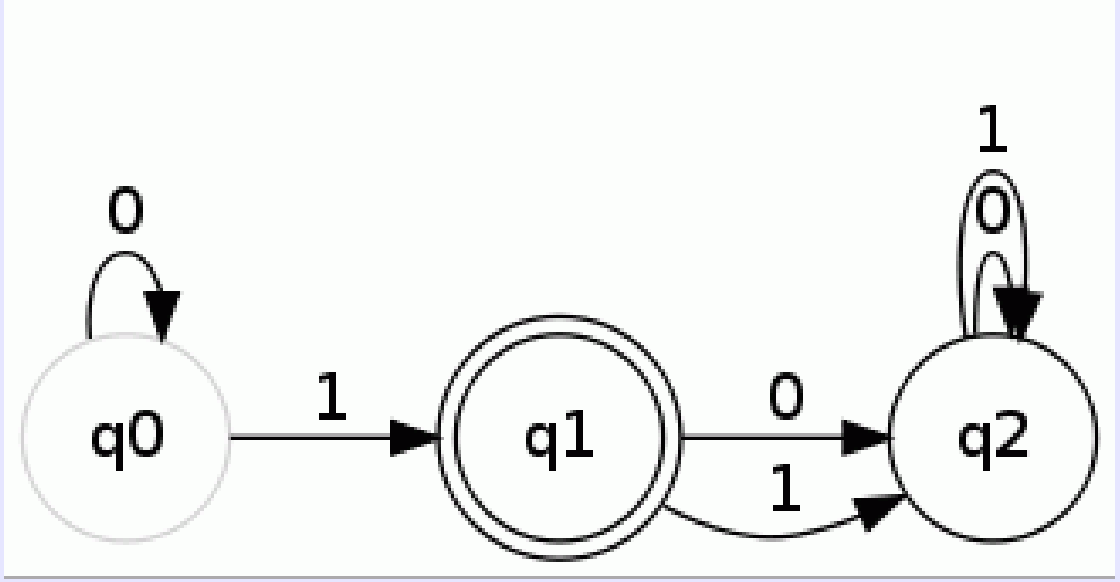
5. EJERCICIOS

2.) Explicar cual es el lenguaje asociado a cada uno de los siguientes autómatas.



5. EJERCICIOS

2.) Explicar cual es el lenguaje asociado a cada uno de los siguientes autómatas.

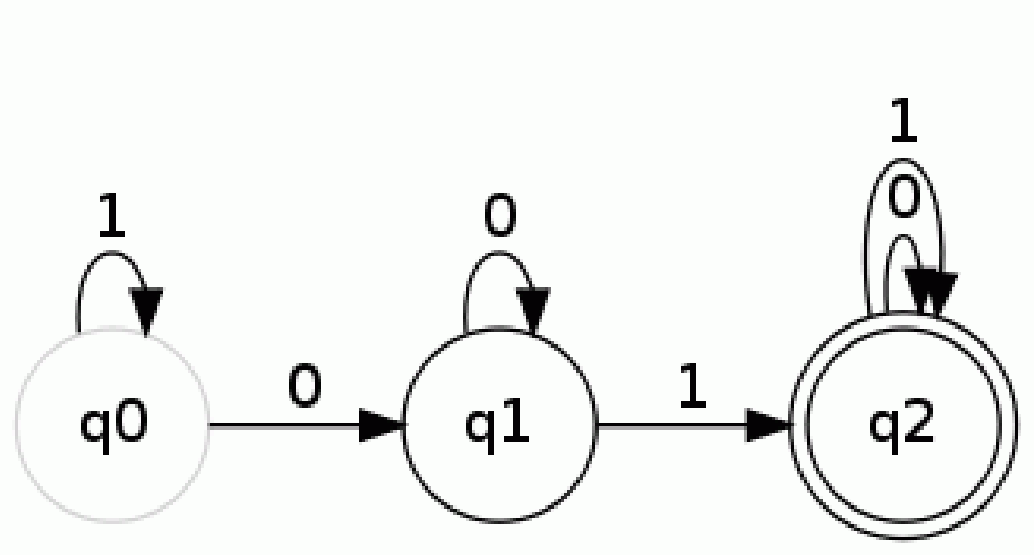


L= 0*1

L={x | x no contiene las subcadenas 10 y 11}

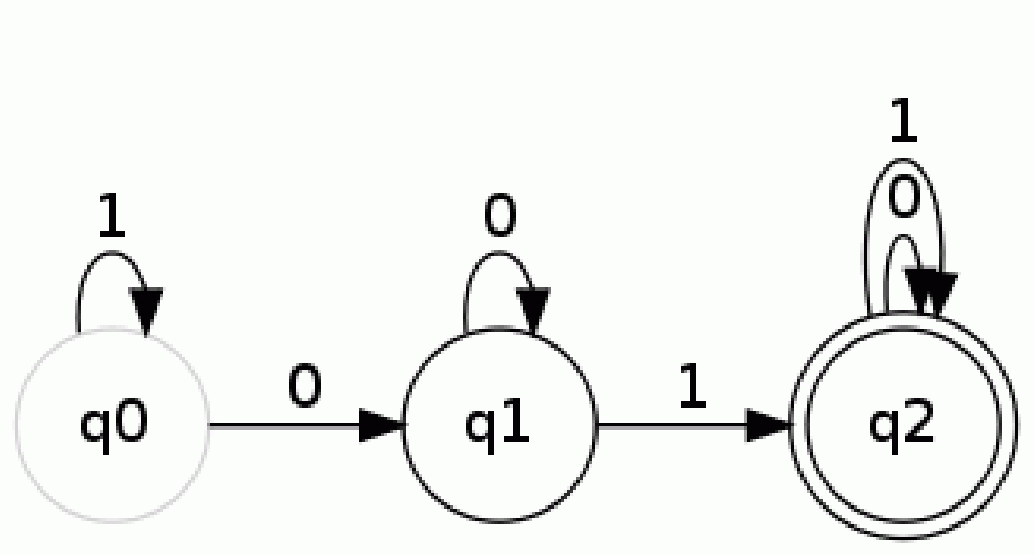
5. EJERCICIOS

2.) Explicar cual es el lenguaje asociado a cada uno de los siguientes autómatas.



5. EJERCICIOS

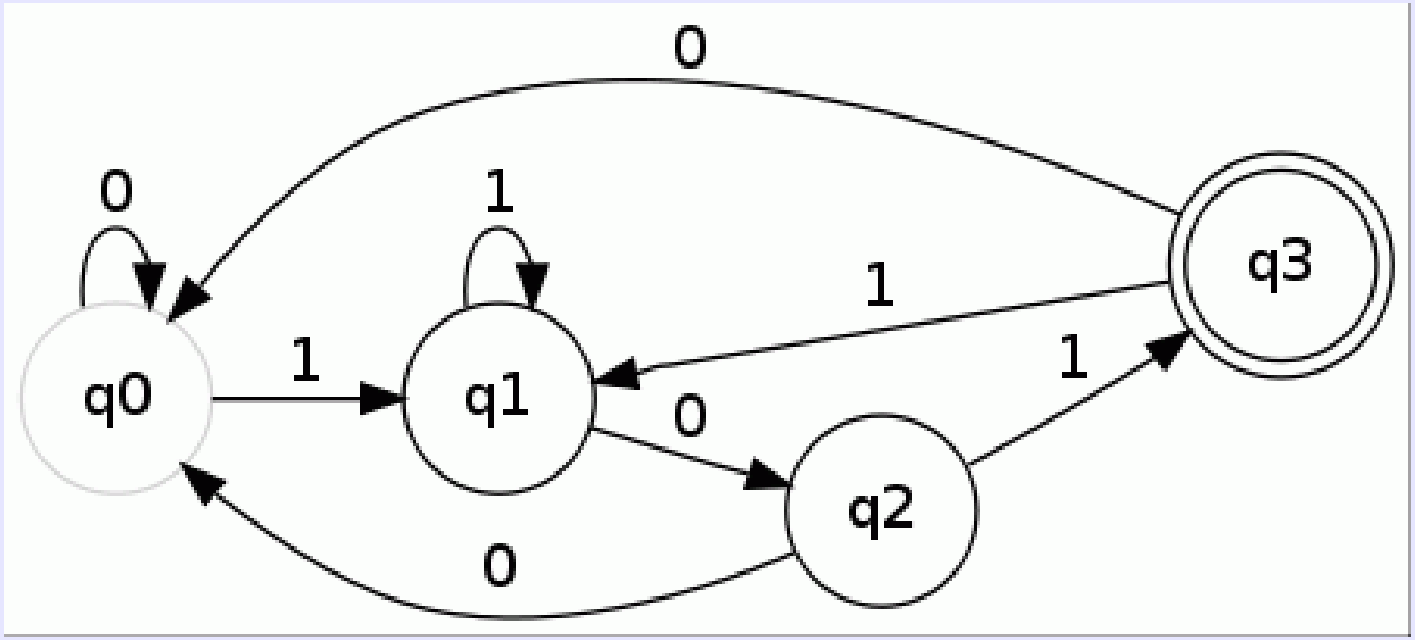
2.) Explicar cual es el lenguaje asociado a cada uno de los siguientes autómatas.



L = {x | x contiene la subcadena 01}

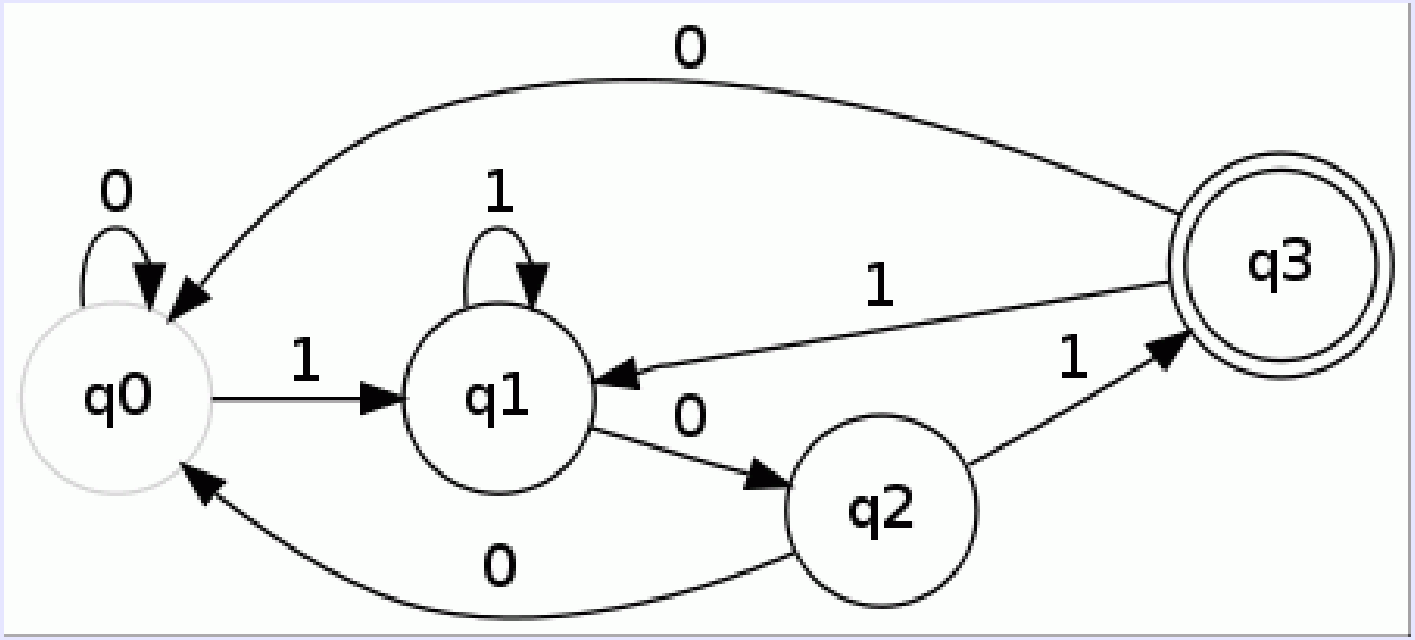
5. EJERCICIOS

2.) Explicar cual es el lenguaje asociado a cada uno de los siguientes autómatas.



5. EJERCICIOS

2.) Explicar cual es el lenguaje asociado a cada uno de los siguientes autómatas.



L = {x | x termina con la subcadena 101

ESTA MAL Q3,0 ES Q2}

CUESTIONARIO 2

1. Sea $\sum$ ={A,B,C} define la expresión regular para el lenguaje cuyas cadenas empiezan por A y terminan en B o en C.

2. Sea $\sum$ ={0, 1} define la expresión regular para el lenguaje: L = {vwz | v, w, z Є {0, 1}* con |w| par, |w|>2 y |v|>0 }

CUESTIONARIO 3

1)

	a	b	c	λ
>q0	q1,q2	ϕ	ϕ	q2
q1	q2	q2	ϕ	q3
*q2	q3	ϕ	q2	ϕ
q3	q3	ϕ	ϕ	q0

a) Dibujar el diagrama de transiciones

Indica **detallando las operaciones:**

b). Si la cadena aacb es aceptada por el autómata ($\delta^*(q0, aacb)$)

2. Diseña un AFD que reconozca el lenguaje $0^*1^*2^+$

TEMA 4-2

EQUIVALENCIA ENTRE AUTÓMATAS FINITOS

INTRODUCCIÓN

1. Equivalencia entre Autómatas finitos

- 1.1. Paso de un AFND-λ a un AFND
- 1.2. Paso de un AFND a un AFD.

2. Minimización de Autómatas.

EQUIVALENCIAS

AFND-$\lambda \rightarrow$ AFND $\rightarrow$ AFD

”Si L es un lenguaje reconocido por un AFND-λ entonces existe un AFD que también lo reconoce”

1) $M_i \rightarrow M_n$

2) $M_n \rightarrow M_d$

AFND-λ a AFND

ENTRADA: un AFND-λ $M_l = (Q_l , \Sigma_l, \delta_l, q_{0l}, F_l)$.

SALIDA: un AFND $M_n = (Q_n, \Sigma, \delta_n, q_{0n} , F_n)$ equivalente a M_l

1. $q_{0n} = q_{0l}$; $Q_n = Q_l$; Inicializar $F_n = F_l$
2. Para todo $q \in Q_l$
    1. Si $\lambda(q)$ contiene a algún estado final de F_l , añadir q a F_n
3. Para cada $q \in Q_l$

    Para cada $a \in \Sigma$

    Obtener $\delta_n (q, a)$ donde $\delta_n (q, a) = \delta_l^* (q, a) =$

    $= \delta_l^* (q,\lambda a) = \lambda (\cup \delta_l (p,a))$

    $p \in \delta_l^* (q,\lambda) = \lambda(q)$

AFND-λ a AFND

ENTRADA: un AFND-λ $M_l = (Q_l , \Sigma_l, \delta_l, q_{0l}, F_l)$.

SALIDA: un AFND $M_n = (Q_n, \Sigma, \delta_n, q_{0n} , F_n)$ equivalente a M_l

1. $q_{0n} = q_{0l}$; $Q_n = Q_l$; Inicializar $F_n = F_l$
2. Para todo $q \in Q_l$
    1. Si λ(q) contiene a algún estado final de F_l , añadir q a F_n
3. Para cada $q \in Q_l$

    Para cada $a \in \Sigma$

    Obtener $\delta_n (q, a)$ donde $\delta_n (q, a) = \delta_l^* (q, a) =$

    $= \delta_l^* (q, \lambda a) = \lambda (\cup \delta_l (p, a))$

    $p \in \delta_l^* (q, \lambda) = \lambda(q)$

$\Sigma = \Sigma_l - \{\lambda\}$

AFND-λ a AFND

ENTRADA: un AFND-λ $M_l = (Q_l , \Sigma_l, \delta_l, q_{0l}, F_l)$.

SALIDA: un AFND $M_n = (Q_n, \Sigma, \delta_n, q_{0n} , F_n)$ equivalente a M_l

1. $q_{0n} = q_{0l}$; $Q_n = Q_l$; Inicializar $F_n = F_l$
2. Para todo $q \in Q_l$
    1. Si λ(q) contiene a algún estado final de F_l , añadir q a F_n
3. Para cada $q \in Q_l$

    Para cada $a \in \Sigma$

    Obtener $\delta_n (q, a)$ donde $\delta_n (q, a) = \delta_l^* (q, a) =$

    $= \delta_l^* (q,\lambda a) = \lambda (\cup \delta_l (p,a))$

    $p \in \delta_l^* (q,\lambda) = \lambda(q)$

MISMO ESTADO INICIAL

AFND-λ a AFND

ENTRADA: un AFND-λ $M_l = (Q_l , \Sigma_l, \delta_l, q_{0l}, F_l)$.

SALIDA: un AFND $M_n = (Q_n, \Sigma, \delta_n, q_{0n} , F_n)$ equivalente a M_l

1. $q_{0n} = q_{0l}$; $Q_n = Q_l$ Inicializar $F_n = F_l$
2. Para todo $q \in Q_l$
    1. Si λ(q) contiene a algún estado final de F_l , añadir q a F_n
3. Para cada $q \in Q_l$

    Para cada $a \in \Sigma$

    Obtener $\delta_n (q, a)$ donde $\delta_n (q, a) = \delta_l^* (q, a) =$

    $= \delta_l^* (q,\lambda a) = \lambda (\cup \delta_l (p,a))$

    $p \in \delta_l^* (q,\lambda) = \lambda(q)$

TIENEN EL MISMO CONJUNTO DE ESTADOS

AFND-λ a AFND

ENTRADA: un AFND-λ $M_l = (Q_l , \Sigma_l, \delta_l, q_{0l}, F_l)$.

SALIDA: un AFND $M_n = (Q_n, \Sigma, \delta_n, q_{0n} , F_n)$ equivalente a M_l

1. $q_{0n} = q_{0l}$; $Q_n = Q_l$; Inicializar $F_n = F_l$
2. Para todo $q \in Q_l$
    1. Si λ(q) contiene a algún estado final de F_l , añadir q a F_n
3. Para cada $q \in Q_l$

    Para cada $a \in \Sigma$

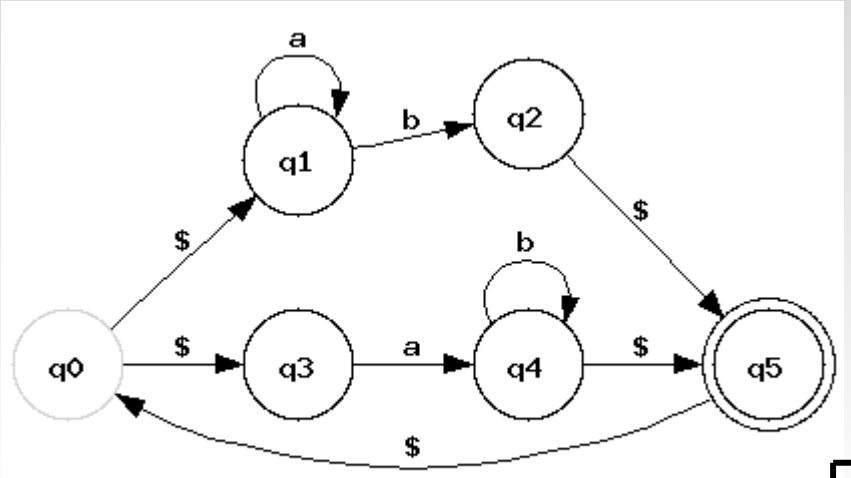
    Obtener $\delta_n (q, a)$ donde $\delta_n (q, a) = \delta_l^* (q, a) =$

    $= \delta_l^* (q,\lambda a) = \lambda (\cup \delta_l (p,a))$

    $p \in \delta_l^* (q,\lambda) = \lambda(q)$

EN PRINCIPIO LOS ESTADOS FINALES SON LOS MISMOS

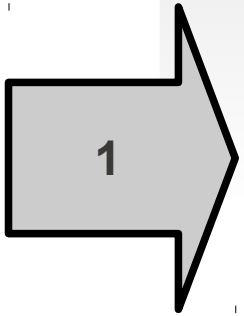
AFND-λ a AFND



δ	a	b	λ
>q0	{}	{}	{q1,q3}
q1	{q1}	{q2}	{}
q2	{}	{}	{q5}
q3	{q4}	{}	{}
q4	{}	{q4}	{q5}
*q5	{}	{}	{q0}

AFND-λ a AFND

δ	a	b	λ
>q0	{}	{}	{q1,q3}
q1	{q1}	{q2}	{}
q2	{}	{}	{q5}
q3	{q4}	{}	{}
q4	{}	{q4}	{q5}
*q5	{}	{}	{q0}



δ	a	b
>q0		
q1		
q2		
q3		
q4		
*q5		

AFND-λ a AFND

ENTRADA: un AFND-λ $M_l = (Q_l , \Sigma_l, \delta_l, q_{0l}, F_l)$.

SALIDA: un AFND $M_n = (Q_n, \Sigma, \delta_n, q_{0n} , F_n)$ equivalente a M_l

1. $q_{0n} = q_{0l}$; $Q_n = Q_l$; Inicializar $F_n = F_l$
2. Para todo $q \in Q_l$
   1. Si λ(q) contiene a algún estado final de F_l , añadir q a F_n
3. Para cada $q \in Q_l$

   Para cada $a \in \Sigma$

   Obtener $\delta_n (q, a)$ donde $\delta_n (q, a) = \delta_l^* (q, a) =$

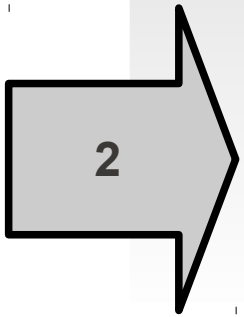
   $= \delta_l^* (q,\lambda a) = \lambda (\cup \delta_l (p,a))$

   $p \in \delta_l^* (q,\lambda) = \lambda(q)$

MODIFICO EL CONJUNTO DE ESTADOS FINALES

AFND-λ a AFND

δ	a	b	λ
>q0	{}	{}	{q1,q3}
q1	{q1}	{q2}	{}
q2	{}	{}	{q5}
q3	{q4}	{}	{}
q4	{}	{q4}	{q5}
*q5	{}	{}	{q0}



δ	a	b
>q0		
q1		
*q2		
q3		
*q4		
*q5		

λ(q0)={q0,q1,q3}

λ(q1)={q1}

λ(q2)={q2,q5,q0,q1,q3}

λ(q3)={q3}

λ(q4)={q4,q5,q0,q1,q3}

λ(q5)={q5,q0,q1,q3}

AFND-λ a AFND

ENTRADA: un AFND-λ $M_l = (Q_l , \Sigma_l, \delta_l, q_{0l}, F_l)$.

SALIDA: un AFND $M_n = (Q_n, \Sigma, \delta_n, q_{0n} , F_n)$ equivalente a M_l

1. $q_{0n} = q_{0l}$; $Q_n = Q_l$; Inicializar $F_n = F_l$
2. Para todo $q \in Q_l$
    1. Si λ(q) contiene a algún estado final de F_l , añadir q a F_n
3. Para cada $q \in Q_l$
    Para cada $a \in \Sigma$

Obtener $\delta_n (q, a)$ donde $\delta_n (q, a) = \delta_l^* (q, a) =$

$= \delta_l^* (q,\lambda a) = \lambda (\cup \delta_l (p,a))$

$p \in \delta_l^* (q,\lambda) = \lambda(q)$

PARA CADA ESTADO Y PARA CADA SÍMBOLO

AFND-λ a AFND

ENTRADA: un AFND-λ $M_l = (Q_l , \Sigma_l, \delta_l, q_{0l}, F_l)$.

SALIDA: un AFND $M_n = (Q_n, \Sigma, \delta_n, q_{0n} , F_n)$ equivalente a M_l

1. $q_{0n} = q_{0l}$; $Q_n = Q_l$; Inicializar $F_n = F_l$
2. Para todo $q \in Q_l$
    1. Si λ(q) contiene a algún estado final de F_l , añadir q a F_n
3. Para cada $q \in Q_l$

    Para cada $a \in \Sigma$

    Obtener $\delta_n (q, a)$ donde $\delta_n (q, a) = \delta_l^* (q, a) =$

    $= \delta_l^* (q,\lambda a) = \lambda (\cup \delta_l (p,a))$

    $p \in \delta_l^* (q,\lambda) = \lambda(q)$

OBTENER LA FUNCIÓN DE TRANSICIÓN

AFND-λ a AFND

ENTRADA: un AFND-λ $M_l = (Q_l , \Sigma_l, \delta_l, q_{0l}, F_l)$.

SALIDA: un AFND $M_n = (Q_n, \Sigma, \delta_n, q_{0n} , F_n)$ equivalente a M_l

1. $q_{0n} = q_{0l}$; $Q_n = Q_l$; Inicializar $F_n = F_l$
2. Para todo $q \in Q_l$
    1. Si λ(q) contiene a algún estado final de F_l , añadir q a F_n
3. Para cada $q \in Q_l$

    Para cada $a \in \Sigma$

    Obtener $\delta_n (q, a)$ donde $\delta_n (q, a) = \delta_l^* (q, a) =$

$$= \delta_l^* (q, \lambda a) = \lambda \left(\cup \delta_l (p, a) \right)$$

$$p \in \delta_l^* (q, \lambda) = \lambda(q)$$

DOS LAMBDA CLAUSURAS

AFND-λ a AFND

$p \in \delta l^*(q0,\lambda)=\lambda(q0)= \{q0,q1,q3\}$

$\delta n(q0,a)=\delta l^*(q0,a) =\lambda\ (\cup\ \delta l(p,a))= \lambda\ (\delta l(q0,a) \cup \delta l(q1,a) \cup \delta l(q3,a)=$

$\lambda\ (\{\} \cup \{q1\} \cup \{q4\})= \{q1,q4,q5,q0,q1,q3\}$

$p \in \delta l^*(q0,\lambda)=\lambda(q0)=\{q0,q1,q3\}$

$\delta n(q0,b)=\delta l^*(q0,b) =\lambda\ (\cup\ \delta l(p,b))= \lambda\ (\delta l(q0,b) \cup \delta l(q1,b) \cup \delta l(q3,b)=$

$\lambda\ (\{\} \cup \{q2\} \cup \{\})= \{q2,q5,q0,q1,q3\}$

$p \in \delta l^*(q1,\lambda)=\lambda(q1)=\{q1\}$

$\delta n(q1,a)=\delta l^*(q1,a) =\lambda\ (\cup\ \delta l(p,a))= \lambda\ (\delta l(q1,a))= \lambda\ (\{q1\})= \{q1\}$

$p \in \delta l^*(q1,\lambda)=\lambda(q1)=\{q1\}$

$\delta n(q1,b)=\delta l^*(q1,b) =\lambda\ (\cup\ \delta l(p,b))= \lambda\ (\delta l(q1,b))= \lambda\ (\{q2\})= \{q2,q5,q0,q1,q3\}$

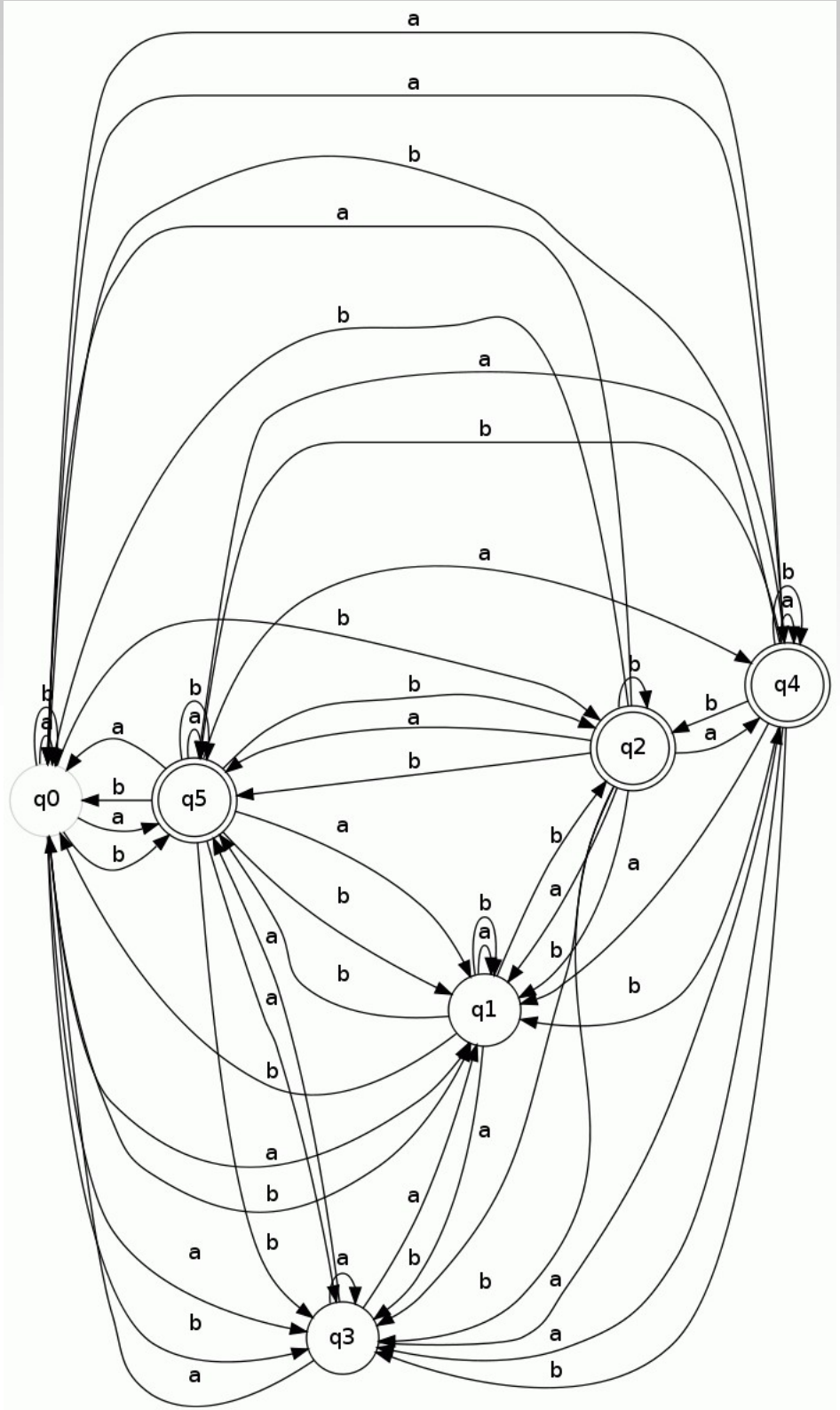
$p \in \delta l^*(q2,\lambda)=\lambda(q2)=\{q2,q5,q0,q1,q3\}$

$\delta n(q2,a)=\delta l^*(q2,a) =\lambda\ (\cup\ \delta l(p,a))= \lambda\ (\delta l(q2,a) \cup \delta l(q5,a) \cup \delta l(q0,a) \cup \delta l(q1,a)$

$\cup\ \delta l(q3,a)) =\ \lambda\ (\{\} \cup \{\}\ \cup \{\} \cup \{q1\} \cup \{q4)= \{q4,q5,q0,q1,q3\}$

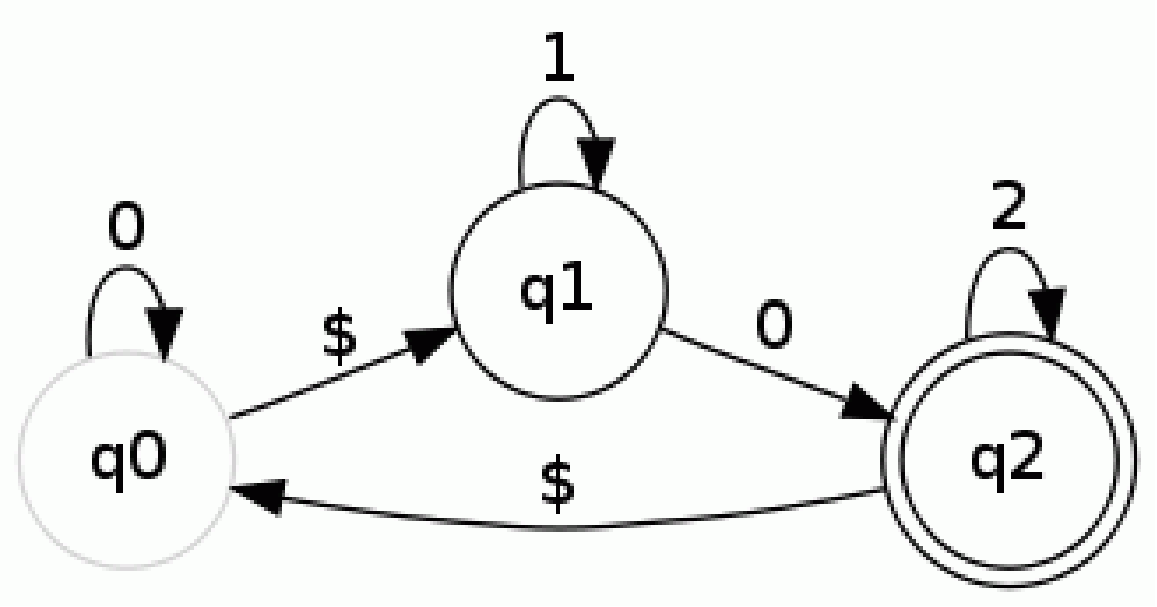
AFND-λ a AFND

δn	a	b
>q0	{q1,q4,q5,q0,q3}	{q2,q5,q0,q1,q3}
q1	{q1}	{q2,q5,q0,q1,q3}
*q2	{q4,q5,q0,q1,q3}	{q2,q5,q0,q1,q3}
q3	{q4,q5,q0,q1,q3}	∅
*q4	{q4,q5,q0,q1,q3}	{q2,q4,q5,q0,q1,q3}
*q5	{q4,q5,q0,q1,q3}	{q2,q5,q0,q1,q3}


q0
q1
q2
q3
q4
q5

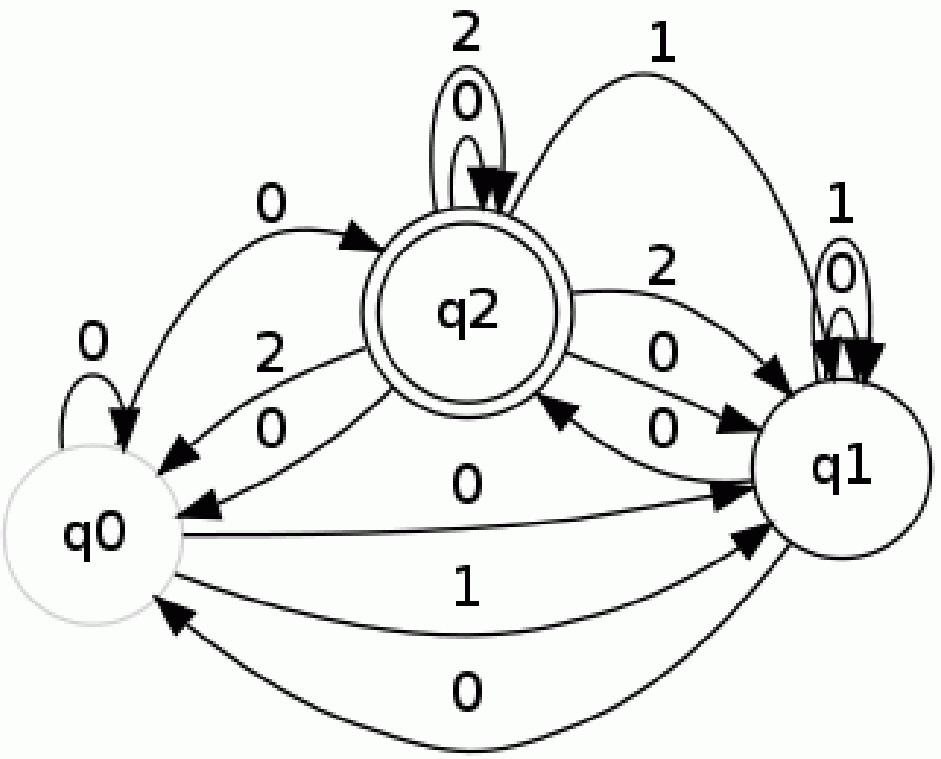
AFND-λ a AFND

	0	1	2	$
>q0	q0	[ ]	[ ]	q1
q1	q2	q1	[ ]	[ ]
*q2	[ ]	[ ]	q2	q0



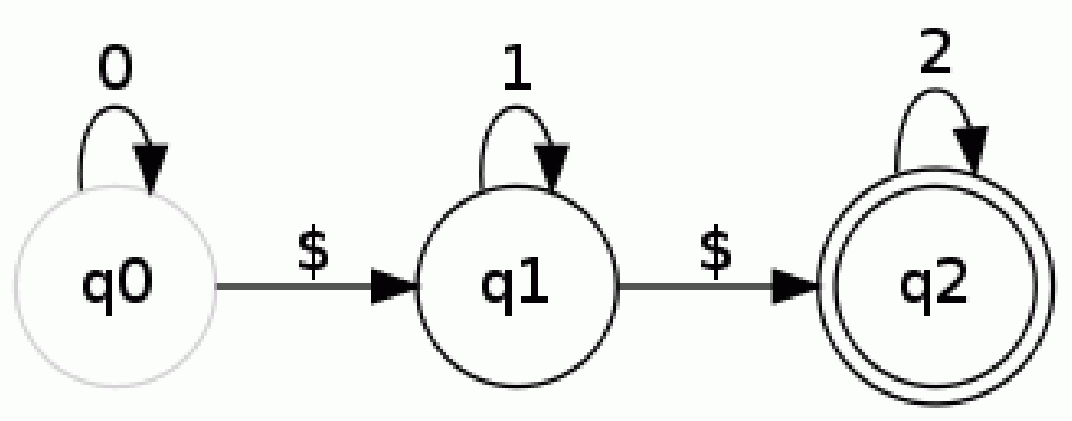
AFND-λ a AFND

	0	1	2
>q0	q0, q1, q2	q1	[ ]
q1	q2, q0, q1	q1	[ ]
*q2	q0, q1, q2	q1	q2, q0, q1



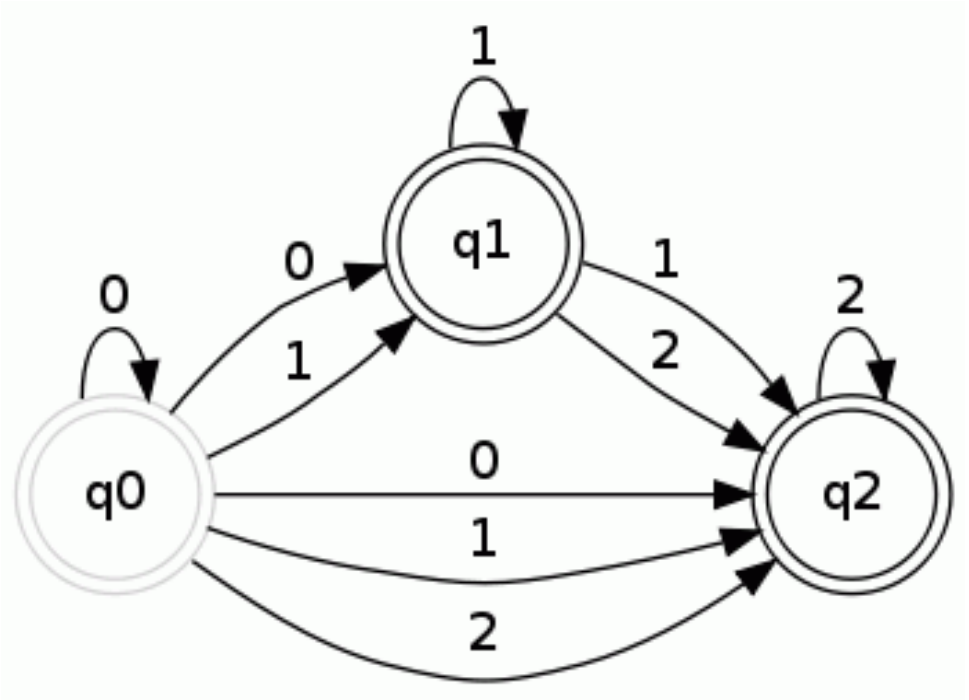
AFND-λ a AFND

	λ	0	1	2
→ q_0	$\{q_1\}$	$\{q_0\}$	$\varnothing$	$\varnothing$
q_1	$\{q_2\}$	$\varnothing$	$\{q_1\}$	$\varnothing$
$^*q_2$	$\varnothing$	$\varnothing$	$\varnothing$	$\{q_2\}$



AFND-λ a AFND

	0	1	2
→ $^{*}q_0$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$^{*}q_1$	$\varnothing$	$\{q_1, q_2\}$	$\{q_2\}$
$^{*}q_2$	$\varnothing$	$\varnothing$	$\{q_2\}$



AFND-λ a AFND

```
automaton prueba{

     states q0,q1,q2;
    alphabet 0,1,2;
     initial q0;
     final q2;

     transition{

          q0,$=q1; q0,0=q0; q1,$=q2;  q1,1=q1;   q2,2=q2;
     }
}


pruebaAFND=FAtoNDFA(prueba);
print(pruebaAFND);
```

AFND A AFD

ENTRADA: un AFND $M_n = (Q_n, \Sigma, \delta_n, q_{0n}, F_n)$

SALIDA: un AFD $M_d = (Q_d, \Sigma, \delta_d, q_{0d}, F_d)$ equivalente a M_n

1. $q_{0d} = \{q_{0n}\}$; $Qd = \{\}$; $EstNoDef = \{q_{0d}\}$

2. Mientras $EstNoDef \neq \emptyset$ hacer
   (a) Sacar un estado S de EstNoDef; $EstNoDef = EstNoDef - \{S\}$
   (b) $Q_d = Q_d \cup \{S\}$
   (c) Para todo $a \in \Sigma$ hacer
   $\delta_d (S, a) = P = U_{q \in S} \delta_n (q, a)$
   Si $P \notin Q_d$ entonces $EstNoDef = EstNoDef \cup \{P\}$

3. $F_d = \{R \in Q_d \mid R \cap F_n \neq \emptyset\}$

LA ÚNICA COINCIDENCIA INICIAL ES EL ALFABETO Y EL ESTADO INICIAL

AFND A AFD

ENTRADA: un AFND $M_n = (Q_n, \Sigma, \delta_n, q_{0n}, F_n)$

SALIDA: un AFD $M_d = (Q_d, \Sigma, \delta_d, q_{0d}, F_d)$ equivalente a M_n

1. $q_{0d} = \{q_{0n}\}$; Qd = {}; EstNoDef = $\{q_{0d}\}$

2. Mientras EstNoDef $\neq \emptyset$ hacer
   (a) Sacar un estado S de EstNoDef; EstNoDef = EstNoDef - {S}
   (b) $Q_d = Q_d \cup \{S\}$
   (c) Para todo $a \in \Sigma$ hacer
   $\delta_d (S, a) = P = U_{q \in S} \delta_n (q, a)$
   Si $P \notin Q_d$ entonces EstNoDef = EstNoDef $\cup$ {P}

3. $F_d = \{R \in Q_d \mid R \cap F_n \neq \emptyset\}$

EstNoDef = {{q0}}

Q_d = { }

AFND A AFD

ENTRADA: un AFND $M_n = (Q_n, \Sigma, \delta_n, q_{0n}, F_n)$

SALIDA: un AFD $M_d = (Q_d, \Sigma, \delta_d, q_{0d}, F_d)$ equivalente a M_n

1. $q_{0d} = \{q_{0n}\}$; Qd = {}; EstNoDef = $\{q_{0d}\}$

2. Mientras EstNoDef $\neq \emptyset$ hacer
   (a) Sacar un estado S de EstNoDef; EstNoDef = EstNoDef - {S}
   (b) $Q_d = Q_d \cup \{S\}$
   (c) Para todo a $\in \Sigma$ hacer
   $\delta_d (S, a) = P = U_{q \in S} \delta_n (q, a)$
   Si P $\notin Q_d$ entonces EstNoDef = EstNoDef $\cup$ {P}

3. $F_d = \{R \in Q_d \mid R \cap F_n \neq \emptyset\}$

EstNoDef = {}

Q_d = {}

S = {q0}

AFND A AFD

ENTRADA: un AFND $M_n = (Q_n, \Sigma, \delta_n, q_{0n}, F_n)$

SALIDA: un AFD $M_d = (Q_d, \Sigma, \delta_d, q_{0d}, F_d)$ equivalente a M_n

1. $q_{0d} = \{q_{0n}\}$; Qd = {}; EstNoDef = $\{q_{0d}\}$

2. Mientras EstNoDef $\neq \emptyset$ hacer
   (a) Sacar un estado S de EstNoDef; EstNoDef = EstNoDef - {S}
   (b) $Q_d = Q_d \cup \{S\}$
   (c) Para todo a $\in \Sigma$ hacer
   $\delta_d (S, a) = P = U_{q \in S} \delta_n (q, a)$
   Si P $\notin Q_d$ entonces EstNoDef = EstNoDef $\cup$ {P}

3. $F_d = \{R \in Q_d \mid R \cap F_n \neq \emptyset\}$

	a	b
> **{q0}**		

EstNoDef = {}

Q_d = {{q0}}

S = {q0}

AFND A AFD

ENTRADA: un AFND $M_n = (Q_n, \Sigma, \delta_n, q_{0n}, F_n)$

SALIDA: un AFD $M_d = (Q_d, \Sigma, \delta_d, q_{0d}, F_d)$ equivalente a M_n

1. $q_{0d} = \{q_{0n}\}$; Qd = {}; EstNoDef = $\{q_{0d}\}$

2. Mientras EstNoDef $\neq \emptyset$ hacer
   (a) Sacar un estado S de EstNoDef; EstNoDef = EstNoDef - {S}
   (b) $Q_d = Q_d \cup \{S\}$
   (c) Para todo $a \in \Sigma$ hacer
   $\delta_d (S, a) = P = U_{q \in S} \delta_n (q, a)$
   Si $P \notin Q_d$ entonces EstNoDef = EstNoDef $\cup$ {P}

3. $F_d = \{R \in Q_d \mid R \cap F_n \neq \emptyset\}$

EstNoDef = {}

Q_d = {{q0}}

S = {q0}

AFND A AFD

δn	a	b
>q0	{q1,q4,q5,q0,q3}	{q2,q5,q0,q1,q3}
q1	{q1}	{q2,q5,q0,q1,q3}
*q2	{q4,q5,q0,q1,q3}	{q2,q5,q0,q1,q3}
q3	{q4,q5,q0,q1,q3}	∅
*q4	{q4,q5,q0,q1,q3}	{q2,q4,q5,q0,q1,q3}
*q5	{q4,q5,q0,q1,q3}	{q2,q5,q0,q1,q3}

	a	b
> {q0}	{q1,q4,q5,q0,q3}	

AFND A AFD

ENTRADA: un AFND $M_n = (Q_n, \Sigma, \delta_n, q_{0n}, F_n)$

SALIDA: un AFD $M_d = (Q_d, \Sigma, \delta_d, q_{0d}, F_d)$ equivalente a M_n

1. $q_{0d} = \{q_{0n}\}$; Qd = {}; EstNoDef = $\{q_{0d}\}$

2. Mientras EstNoDef $\neq \emptyset$ hacer
   (a) Sacar un estado S de EstNoDef; EstNoDef = EstNoDef - {S}
   (b) $Q_d = Q_d \cup \{S\}$
   (c) Para todo a $\in \Sigma$ hacer
   $\delta_d (S, a) = P = U_{q \in S} \delta_n (q, a)$
   Si $P \notin Q_d$ entonces EstNoDef = EstNoDef $\cup$ {P}

3. $F_d = \{R \in Q_d \mid R \cap F_n \neq \emptyset\}$

EstNoDef = {{q1,q4,q5,q0,q3}}

Q_d = {{q0}}

S = {q0}

P = {q1,q4,q5,q0,q3}

AFND A AFD

ENTRADA: un AFND $M_n = (Q_n, \Sigma, \delta_n, q_{0n}, F_n)$

SALIDA: un AFD $M_d = (Q_d, \Sigma, \delta_d, q_{0d}, F_d)$ equivalente a M_n

1. $q_{0d} = \{q_{0n}\}$; Qd = {}; EstNoDef = $\{q_{0d}\}$

2. Mientras EstNoDef $\neq \emptyset$ hacer
   (a) Sacar un estado S de EstNoDef; EstNoDef = EstNoDef - {S}
   (b) $Q_d = Q_d \cup \{S\}$
   (c) Para todo a $\in \Sigma$ hacer
   $\delta_d (S, a) = P = U_{q \in S} \delta_n (q, a)$
   Si P $\notin Q_d$ entonces EstNoDef = EstNoDef $\cup$ {P}

3. $F_d = \{R \in Q_d \mid R \cap F_n \neq \emptyset\}$

EstNoDef = {{q1,q4,q5,q0,q3}}

Q_d = {{q0}}

S = {q0}

P = {q1,q4,q5,q0,q3}

AFND A AFD

δn	a	b
>q0	{q1,q4,q5,q0,q3}	{q2,q5,q0,q1,q3}
q1	{q1}	{q2,q5,q0,q1,q3}
*q2	{q4,q5,q0,q1,q3}	{q2,q5,q0,q1,q3}
q3	{q4,q5,q0,q1,q3}	∅
*q4	{q4,q5,q0,q1,q3}	{q2,q4,q5,q0,q1,q3}
*q5	{q4,q5,q0,q1,q3}	{q2,q5,q0,q1,q3}

	a	b
> {q0}	{q1,q4,q5,q0,q3}	{q2,q5,q0,q1,q3}

AFND A AFD

ENTRADA: un AFND $M_n = (Q_n, \Sigma, \delta_n, q_{0n}, F_n)$

SALIDA: un AFD $M_d = (Q_d, \Sigma, \delta_d, q_{0d}, F_d)$ equivalente a M_n

1. $q_{0d} = \{q_{0n}\}$; Qd = {}; EstNoDef = $\{q_{0d}\}$

2. Mientras EstNoDef $\neq \emptyset$ hacer
   (a) Sacar un estado S de EstNoDef; EstNoDef = EstNoDef - {S}
   (b) $Q_d = Q_d \cup \{S\}$
   (c) Para todo $a \in \Sigma$ hacer
   $\delta_d(S, a) = P = U_{q \in S}\, \delta_n(q, a)$
   Si $P \notin Q_d$ entonces EstNoDef = EstNoDef $\cup$ {P}

3. $F_d = \{R \in Q_d \mid R \cap F_n \neq \emptyset\}$

EstNoDef = {{q1,q4,q5,q0,q3}, {q2,q5,q0,q1,q3}}

Q_d = {{q0}}

S = {q0}

P = {q2,q5,q0,q1,q3}

AFND A AFD

ENTRADA: un AFND $M_n = (Q_n, \Sigma, \delta_n, q_{0n}, F_n)$

SALIDA: un AFD $M_d = (Q_d, \Sigma, \delta_d, q_{0d}, F_d)$ equivalente a M_n

1. $q_{0d} = \{q_{0n}\}$; Qd = {}; EstNoDef = $\{q_{0d}\}$

2. Mientras EstNoDef $\neq \emptyset$ hacer
   (a) Sacar un estado S de EstNoDef; EstNoDef = EstNoDef - {S}
   (b) $Q_d = Q_d \cup \{S\}$
   (c) Para todo $a \in \Sigma$ hacer
   $\delta_d(S, a) = P = U_{q \in S}\, \delta_n(q, a)$
   Si $P \notin Q_d$ entonces EstNoDef = EstNoDef $\cup$ {P}

3. $F_d = \{R \in Q_d \mid R \cap F_n \neq \emptyset\}$

EstNoDef = {{q1,q4,q5,q0,q3}, {q2,q5,q0,q1,q3}}

Q_d = {{q0}}

S = {q0}

P = {q2,q5,q0,q1,q3}

AFND A AFD

ENTRADA: un AFND $M_n = (Q_n, \Sigma, \delta_n, q_{0n}, F_n)$

SALIDA: un AFD $M_d = (Q_d, \Sigma, \delta_d, q_{0d}, F_d)$ equivalente a M_n

1. $q_{0d} = \{q_{0n}\}$; Qd = {}; EstNoDef = $\{q_{0d}\}$

2. Mientras EstNoDef $\neq \emptyset$ hacer
   (a) Sacar un estado S de EstNoDef; EstNoDef = EstNoDef - {S}
   (b) $Q_d = Q_d \cup \{S\}$
   (c) Para todo a $\in \Sigma$ hacer
   $\delta_d (S, a) = P = U_{q \in S} \delta_n (q, a)$
   Si P $\notin Q_d$ entonces EstNoDef = EstNoDef $\cup$ {P}

3. $F_d = \{R \in Q_d \mid R \cap F_n \neq \emptyset\}$

EstNoDef = {{q2,q5,q0,q1,q3}}

Q_d = {{q0}, {q1,q4,q5,q0,q3}}

S = {q1,q4,q5,q0,q3}

AFND A AFD

ENTRADA: un AFND $M_n = (Q_n, \Sigma, \delta_n, q_{0n}, F_n)$

SALIDA: un AFD $M_d = (Q_d, \Sigma, \delta_d, q_{0d}, F_d)$ equivalente a M_n

1. $q_{0d} = \{q_{0n}\}$; Qd = {}; EstNoDef = $\{q_{0d}\}$

2. Mientras EstNoDef $\neq \emptyset$ hacer
   (a) Sacar un estado S de EstNoDef; EstNoDef = EstNoDef - {S}
   (b) $Q_d = Q_d \cup \{S\}$
   (c) Para todo $a \in \Sigma$ hacer
   $\delta_d(S, a) = P = U_{q \in S}\, \delta_n(q, a)$
   Si $P \notin Q_d$ entonces EstNoDef = EstNoDef $\cup$ {P}

3. $F_d = \{R \in Q_d \mid R \cap F_n \neq \emptyset\}$

EstNoDef = {{q2,q5,q0,q1,q3}}

Q_d = {{q0}, {q1,q4,q5,q0,q3}}

S = {q1,q4,q5,q0,q3}

AFND A AFD

δn	a	b
>q0	{q1,q4,q5,q0,q3}	{q2,q5,q0,q1,q3}
q1	{q1}	{q2,q5,q0,q1,q3}
*q2	{q4,q5,q0,q1,q3}	{q2,q5,q0,q1,q3}
q3	{q4,q5,q0,q1,q3}	∅
*q4	{q4,q5,q0,q1,q3}	{q2,q4,q5,q0,q1,q3}
*q5	{q4,q5,q0,q1,q3}	{q2,q5,q0,q1,q3}

δ*({q1,q4,q5,q0,q3}, a) = $\delta(q_1, a) \cup \delta(q_4, a) \cup \delta(q_5, a) \cup \delta(q_0, a) \cup \delta(q_3, a) = \{q_1\} \cup \{q4,q5,q0,q1,q3\} \cup \{q4,q5,q0,q1,q3\} \cup \{q1,q4,q5,q0,q3\} \cup \{q4,q5,q0,q1,q3\} = \{q4,q5,q0,q1,q3\}$

AFND A AFD

δn	a	b
>q0	{q1,q4,q5,q0,q3}	{q2,q5,q0,q1,q3}
q1	{q1}	{q2,q5,q0,q1,q3}
*q2	{q4,q5,q0,q1,q3}	{q2,q5,q0,q1,q3}
q3	{q4,q5,q0,q1,q3}	∅
*q4	{q4,q5,q0,q1,q3}	{q2,q4,q5,q0,q1,q3 }
*q5	{q4,q5,q0,q1,q3}	{q2,q5,q0,q1,q3}

	a	b
> {q0}	{q1,q4,q5,q0,q3}	{q2,q5,q0,q1,q3}
{q1,q4,q5,q0,q3}	{q4,q5,q0,q1,q3}	

AFND A AFD

ENTRADA: un AFND $M_n = (Q_n, \Sigma, \delta_n, q_{0n}, F_n)$

SALIDA: un AFD $M_d = (Q_d, \Sigma, \delta_d, q_{0d}, F_d)$ equivalente a M_n

1. $q_{0d} = \{q_{0n}\}$; Qd = {}; EstNoDef = $\{q_{0d}\}$

2. Mientras EstNoDef $\neq \emptyset$ hacer
   (a) Sacar un estado S de EstNoDef; EstNoDef = EstNoDef - {S}
   (b) $Q_d = Q_d \cup \{S\}$
   (c) Para todo a $\in \Sigma$ hacer
   $\delta_d (S, a) = P = U_{q \in S} \delta_n (q, a)$
   Si **$P \notin Q_d$** entonces EstNoDef = EstNoDef $\cup$ {P}

3. $F_d = \{R \in Q_d \mid R \cap F_n \neq \emptyset\}$

EstNoDef = {{q2,q5,q0,q1,q3}}

Q_d = {{q0}, {q1,q4,q5,q0,q3}}

S = {q1,q4,q5,q0,q3}

P = {q4,q5,q0,q1,q3}

AFND A AFD

δ_n	a	b
>q0	{q1,q4,q5,q0,q3}	{q2,q5,q0,q1,q3}
q1	{q1}	{q2,q5,q0,q1,q3}
*q2	{q4,q5,q0,q1,q3}	{q2,q5,q0,q1,q3}
q3	{q4,q5,q0,q1,q3}	∅
*q4	{q4,q5,q0,q1,q3}	{q2,q4,q5,q0,q1,q3}
*q5	{q4,q5,q0,q1,q3}	{q2,q5,q0,q1,q3}

δ_d	a	b
>{q0} I	{q1, q4, q5, q0, q3} II	{q2, q5, q0, q1, q3} III
{q1, q4, q5, q0, q3} II	{q1, q4, q5, q0, q3} II	{q2, q5, q0, q1, q3,q4} IV
{q2, q5, q0, q1, q3} III	{q1, q4, q5, q0, q3} II	{q2, q5, q0, q1, q3} III
{q2, q5, q0, q1, q3,q4} IV	{q1, q4, q5, q0, q3} II	{q2, q5, q0, q1, q3,q4} IV

AFND A AFD

ENTRADA: un AFND $M_n = (Q_n, \Sigma, \delta_n, q_{0n}, F_n)$

SALIDA: un AFD $M_d = (Q_d, \Sigma, \delta_d, q_{0d}, F_d)$ equivalente a M_n

1. $q_{0d} = \{q_{0n}\}$; Qd = {}; EstNoDef = $\{q_{0d}\}$

2. Mientras EstNoDef $\neq \emptyset$ hacer
   (a) Sacar un estado S de EstNoDef; EstNoDef = EstNoDef - {S}
   (b) $Q_d = Q_d \cup \{S\}$
   (c) Para todo $a \in \Sigma$ hacer
   $\delta_d (S, a) = P = U_{q \in S} \delta_n (q, a)$
   Si $P \notin Q_d$ entonces EstNoDef = EstNoDef $\cup$ {P}

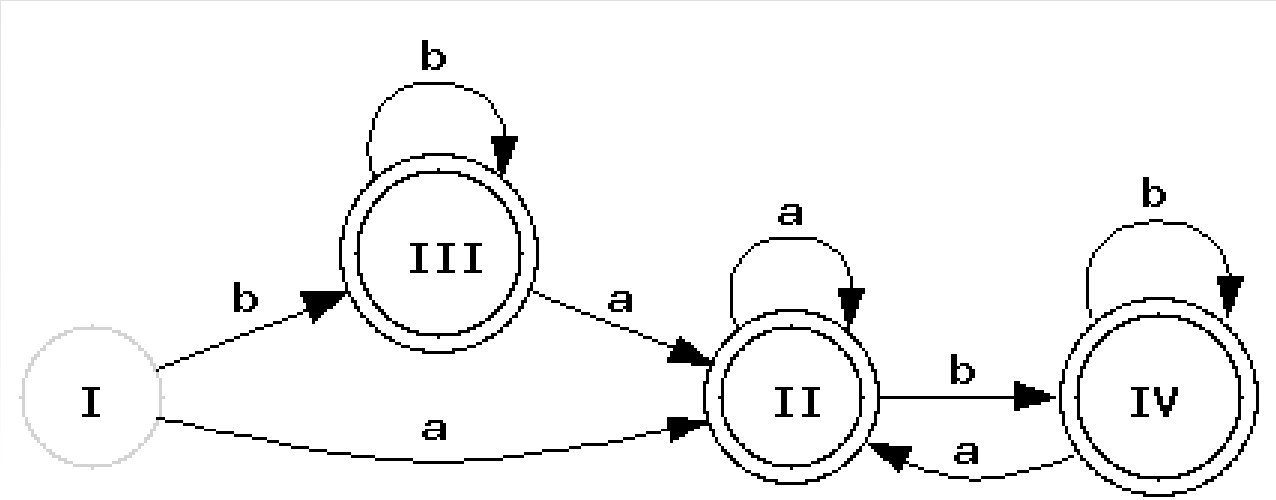
3. $F_d = \{R \in Q_d \mid R \cap F_n \neq \emptyset\}$

AFND A AFD

δ_n	a	b
>q0	{q1,q4,q5,q0,q3}	{q2,q5,q0,q1,q3}
q1	{q1}	{q2,q5,q0,q1,q3}
*q2	{q4,q5,q0,q1,q3}	{q2,q5,q0,q1,q3}
q3	{q4,q5,q0,q1,q3}	∅
*q4	{q4,q5,q0,q1,q3}	{q2,q4,q5,q0,q1,q3}
*q5	{q4,q5,q0,q1,q3}	{q2,q5,q0,q1,q3}

δ_d	a	b
>{q0} I	{q1, q4, q5, q0, q3} II	{q2, q5, q0, q1, q3} III
*{q1, q4, q5, q0, q3} II	{q1, q4, q5, q0, q3} II	{q2, q5, q0, q1, q3,q4} IV
*{q2, q5, q0, q1, q3} III	{q1, q4, q5, q0, q3} II	{q2, q5, q0, q1, q3} III
*{q2, q5, q0, q1, q3,q4} IV	{q1, q4, q5, q0, q3} II	{q2, q5, q0, q1, q3,q4} IV

AFND A AFD



δ_d	a	b
>{q0} I	{q1, q4, q5, q0, q3} II	{q2, q5, q0, q1, q3} III
*{q1, q4, q5, q0, q3} II	{q1, q4, q5, q0, q3} II	{q2, q5, q0, q1, q3,q4} IV
*{q2, q5, q0, q1, q3} III	{q1, q4, q5, q0, q3} II	{q2, q5, q0, q1, q3} III
*{q2, q5, q0, q1, q3,q4} IV	{q1, q4, q5, q0, q3} II	{q2, q5, q0, q1, q3,q4} IV

AFND A AFD

	0	1	2
→ $^{*}q_0$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$^{*}q_1$	$\varnothing$	$\{q_1, q_2\}$	$\{q_2\}$
$^{*}q_2$	$\varnothing$	$\varnothing$	$\{q_2\}$

Al inicio se deja **sólo la primera fila** con el estado que sigue siendo inicial. Ya no es el estado q0 sino un conjunto con un elemento que es el estado q0. Las **transiciones** son **idénticas** que las del autómata origen. Por ahora **no hay estados finales**.

	0	1	2
→ $\{q_0\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$

AFND A AFD

	0	1	2
→ $^{*}q_0$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$^{*}q_1$	$\varnothing$	$\{q_1, q_2\}$	$\{q_2\}$
$^{*}q_2$	$\varnothing$	$\varnothing$	$\{q_2\}$

Cada **transición nueva** que aparezca será un **nuevo estado.** Como hay tres transiciones distintas como mínimo el autómata tendrá tres estados más.

Añadimos un nuevo estado y calculamos las transiciones.

	0	1	2
→ $\{q_0\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$

AFND A AFD

	0	1	2
→ $^{*}q_0$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$^{*}q_1$	$\varnothing$	$\{q_1, q_2\}$	$\{q_2\}$
$^{*}q_2$	$\varnothing$	$\varnothing$	$\{q_2\}$

Añadimos un nuevo estado.

	0	1	2
→ $\{q_0\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$\{q_1, q_2\}$			

Nos surge un caso no visto anteriormente ni reflejado en el Algoritmo. **$\delta^*(\{q1,q2\}, 0) = \delta(q_1, 0) \cup \delta(q_2, 0) = \varnothing$**

Esto no puedo ocurrir porque queremos obtener un AFD

AFND A AFD

	0	1	2
→ $^{*}q_0$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$^{*}q_1$	$\varnothing$	$\{q_1, q_2\}$	$\{q_2\}$
$^{*}q_2$	$\varnothing$	$\varnothing$	$\{q_2\}$

Solución: se añade un **estado sumidero o de error.**

	0	1	2
→ $\{q_0\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$\{q_1, q_2\}$	$\mathbf{q_e}$	$\{q_1, q_2\}$	$\{q_2\}$

AFND A AFD

	0	1	2
→ $^{*}q_0$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$^{*}q_1$	$\varnothing$	$\{q_1, q_2\}$	$\{q_2\}$
$^{*}q_2$	$\varnothing$	$\varnothing$	$\{q_2\}$

Solución: se añade un estado sumidero o de error.

	0	1	2
→ $\{q_0\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$\{q_1, q_2\}$	q_e	$\{q_1, q_2\}$	$\{q_2\}$
$\{q_2\}$	q_e	q_e	$\{q_2\}$

¿Todo lo que aparece en las transiciones está como estado?

AFND A AFD

	0	1	2
→ $^{*}q_0$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$^{*}q_1$	$\varnothing$	$\{q_1, q_2\}$	$\{q_2\}$
$^{*}q_2$	$\varnothing$	$\varnothing$	$\{q_2\}$

FALTA EL ESTADO DE ERROR O **SUMIDERO**

	0	1	2
→ $\{q_0\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$\{q_1, q_2\}$	q_e	$\{q_1, q_2\}$	$\{q_2\}$
$\{q_2\}$	q_e	q_e	$\{q_2\}$
q_e	q_e	q_e	q_e

AFND A AFD

	0	1	2
→ $^{*}q_0$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$^{*}q_1$	$\varnothing$	$\{q_1, q_2\}$	$\{q_2\}$
$^{*}q_2$	$\varnothing$	$\varnothing$	$\{q_2\}$

FALTARÍA VER VEN CUALES SON LOS **ESTADOS FINALES.**

	0	1	2
→ $^{*}\{q_0\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$^{*}\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$^{*}\{q_1, q_2\}$	q_e	$\{q_1, q_2\}$	$\{q_2\}$
$^{*}\{q_2\}$	q_e	q_e	$\{q_2\}$
q_e	q_e	q_e	q_e

AFND A AFD

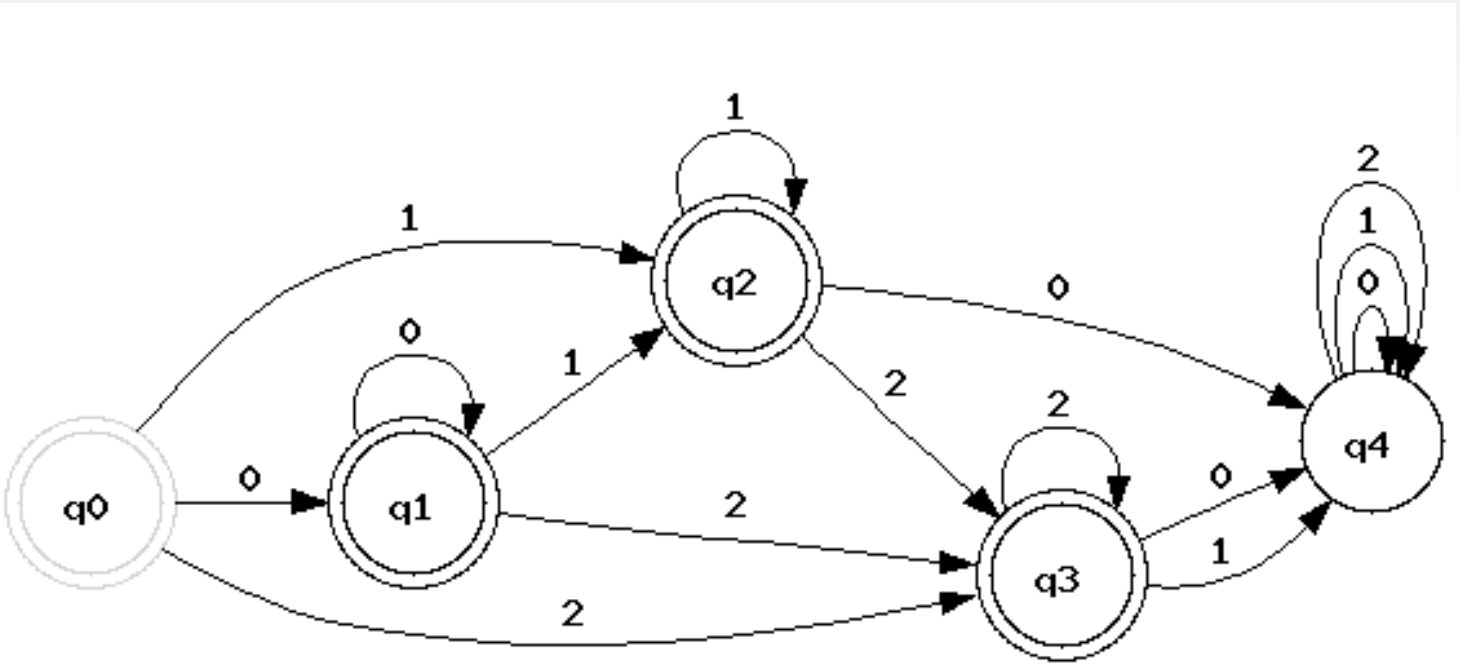
	0	1	2
→ $*q_0$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$*\{q_0, q_1, q_2\}$	$\{q_0, q_1, q_2\}$	$\{q_1, q_2\}$	$\{q_2\}$
$*\{q_1, q_2\}$	q_e	$\{q_1, q_2\}$	$\{q_2\}$
$*\{q_2\}$	q_e	q_e	$\{q_2\}$
q_e	q_e	q_e	q_e

POR ÚLTIMO RENOMBRAMOS LOS ESTADOS:

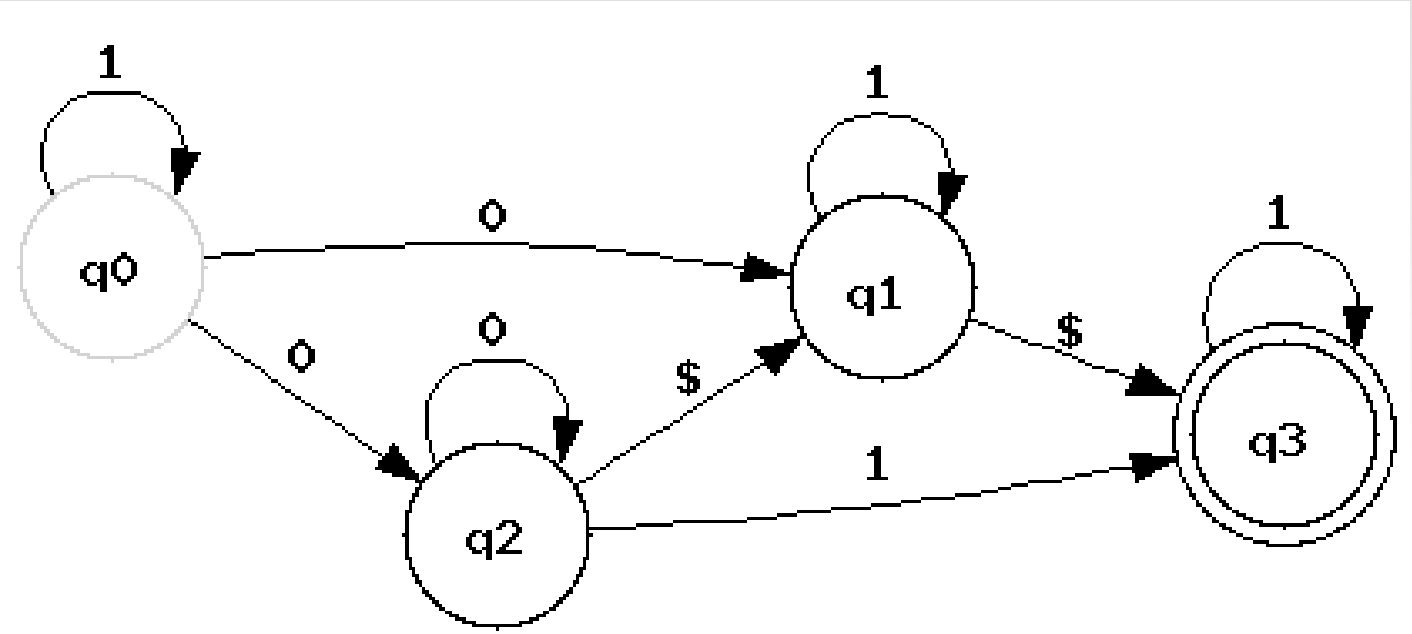
Antiguo:		Nuevo:	V	0	1	2
[q0]	----->	q0	>*q0	q1	q2	q3
[q0, q1, q2]	----->	q1	*q1	q1	q2	q3
[q1, q2]	----->	q2	*q2	q4	q2	q3
[q2]	----->	q3	*q3	q4	q4	q3
[qe]	----->	q4	q4	q4	q4	q4

AFND A AFD

V	0	1	2
>*q0	q1	q2	q3
*q1	q1	q2	q3
*q2	q4	q2	q3
*q3	q4	q4	q3
q4	q4	q4	q4

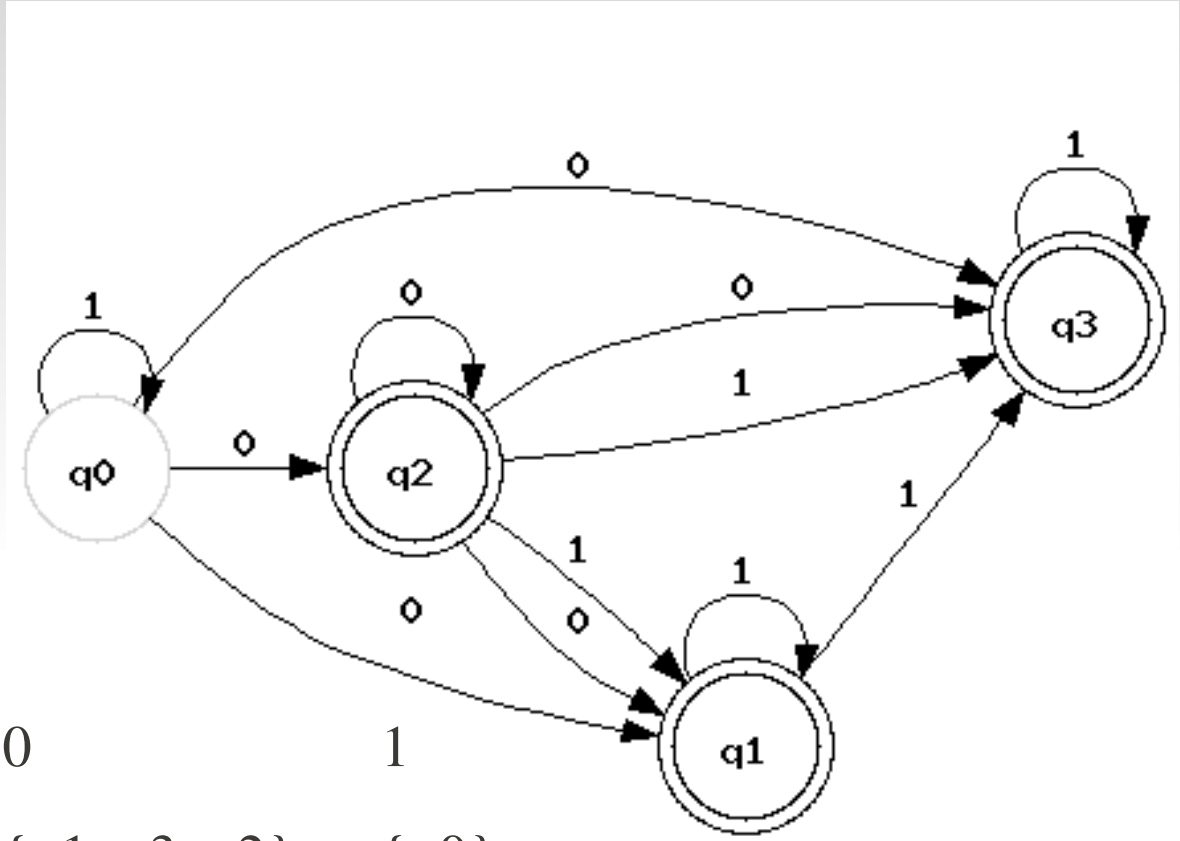


AFND-λ A AFD



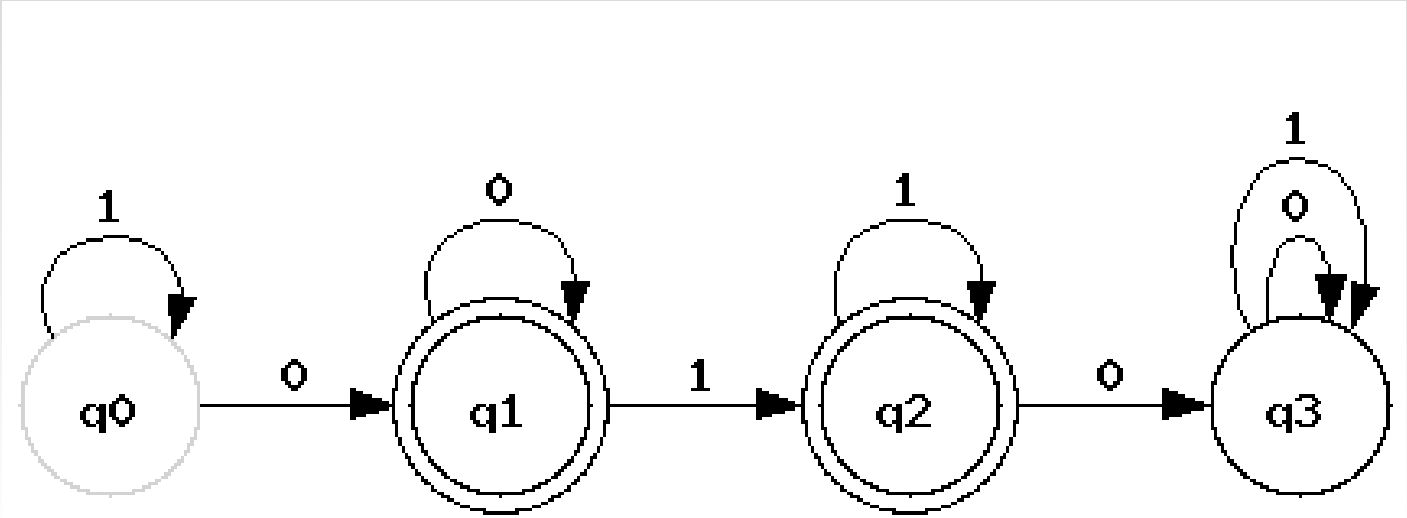
V	0	1	$
>q0	{q1, q2}	{q0}	[ ]
q1	[ ]	{q1}	{q3}
q2	{q2}	{q3}	{q1}
*q3	[ ]	{q3}	[ ]

AFND-λ A AFND



V	0	1
>q0	{q1, q3, q2}	{q0}
*q1	[ ]	{q1, q3}
*q2	{q2, q1, q3}	{q3, q1}
*q3	[ ]	{q3}

AFND-λ A AFD



V	0	1
>{q0}	{q1, q3, q2}	{q0}
*{q1,q3,q2}	{q2, q1, q3}	{q1, q3}
*{q1,q3}	qe	{q1, q3}
qe	qe	qe

Antiguo: ------> Nuevo:

{q0} ------> q0

{q1, q3, q2} ------> q1

{q1, q3} ------> q2

qe ------> q3

V	0	1
>q0	q1	q0
*q1	q1	q2
*q2	q3	q2
q3	q3	q3

EQUIVALENCIA

Antes de continuar hay que reflexionar sobre la definición de **equivalencia**:

TODOS LOS AUTÓMATAS OBTENIDOS POR LOS ALGORITMOS VISTOS ANTERIORMENTE TIENEN QUE **RECONOCER EL MISMO LENGUAJE**

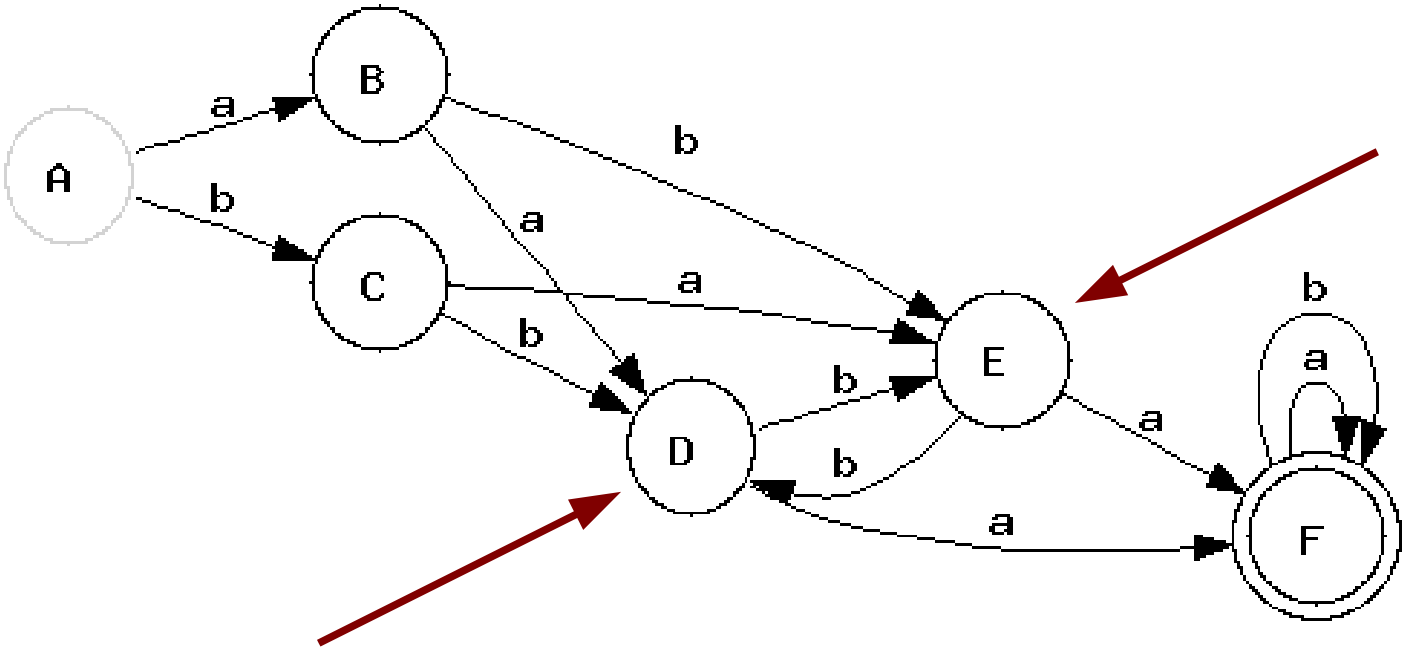
Minimización

Eliminar un estado siempre que ya **exista** otro que realiza la **misma función**: **cualquier** palabra de **entrada** da el **mismo resultado** partiendo de un estado o de otro.

- Si eliminamos un estado → **Redireccionamos** sus **entradas** al estado **equivalente** o indistinguible.

Minimización

- Dos estados son **equivalentes** si para la misma palabra de entrada:
  - ambos llegan a un estado final
  - ninguno llega a un estado final.



Algoritmo de Minimización

ENTRADA: un AFD $M_d = (Q_d, \Sigma, \delta_d, q_{0d}, F_d)$

SALIDA: un AFD Mínimo $M_m = (Q_m, \Sigma, \delta_m, q_{0m}, F_m)$ equivalente a M_d

1. $i=0$, $Q/E_0 = \{ c_1, c_2 \}$ donde $c_1 = Q_d - F$ y $c_2 = F$

2. Para cualquier par $p,q \in c_j/E_i$ hacer
   Si $\forall a \in \Sigma$ se cumple que $\delta(p,a) \in c_i$ y $\delta(q,a) \in c_i$ entonces $p \; E_{i+1} \; q$
   en caso contrario $p \; E_{i+1} \; q$ (no equivalente)
3. Calcular Q/E_{i+1} poniendo en la misma clase a los pares equivalentes y en clases distintas a los no equivalentes.
   Si $Q/E_{i+1} \neq Q/E_i$, i++, volver a 2
   en caso contrario $Q/E_i = Q/E = Q_m$,
4. Obtener M_m
   $Q_m = Q/E_i$
   $q_{om} = c_i \in Q_m \; / \; q_{od} \in c_i$
   $F_m = \{ ci \in Q_m \mid q_f \in c_i \; , \; q_f \in F_d \}$
   $\delta_m (c_i , a) = c_j$ / si $q_i \in c_i$ y $q_j \in c_j$ y $\delta_d (q_i, a) = q_j$

Algoritmo de Minimización

- Mediante clases de equivalencia:

$Q/E_0=\{Q-\{F\}, F\}=\{C_0,C_1\}$

$Q/E_1=\{$En cada clase buscar equivalencias$\}$

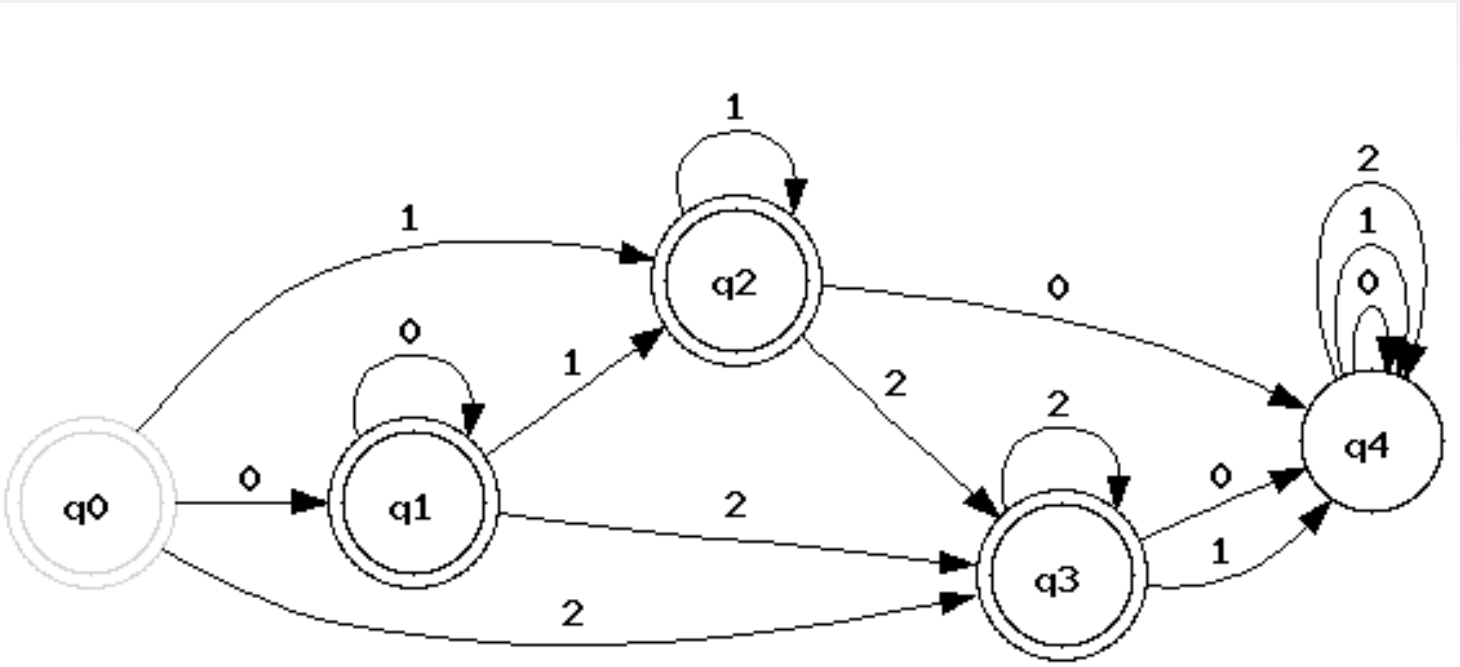
Ej:

$Q/E_0= \{\{q0,q2\}, \{q3\}\}$

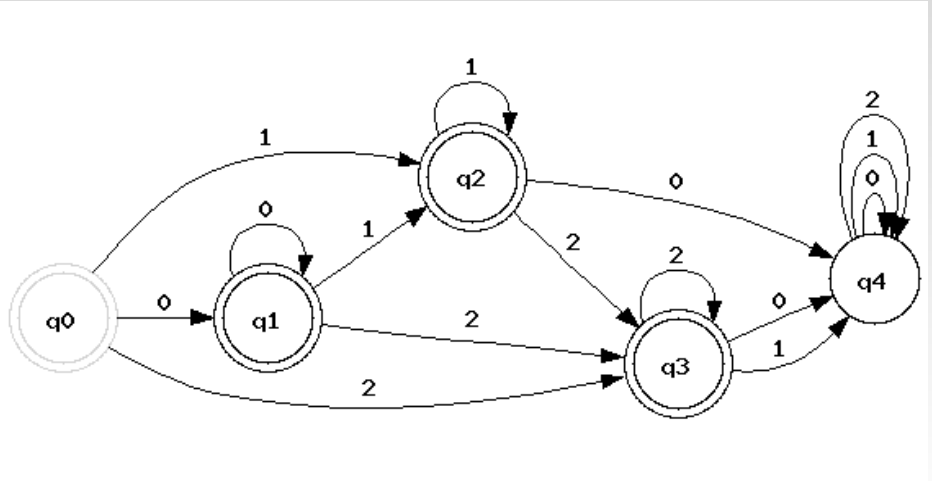
- q0 y q2 son equivalentes si para cada símbolo acaban en estados que estén en la misma clase de equivalencia (sea o no a la que ellos pertenecen)
- Si no son equivalentes parto las clases, generando nuevas.

Algoritmo de Minimización

V	0	1	2
>*q0	q1	q2	q3
*q1	q1	q2	q3
*q2	q4	q2	q3
*q3	q4	q4	q3
q4	q4	q4	q4



Algoritmo de Minimización



Q/E_0={{q4}, {q0,q1,q2,q3}}

δ*(q0,0), δ*(q0,1) y δ*(q0,2) ∈ C1

δ*(q1,0), δ*(q1,1) y δ*(q1,2) ∈ C1

δ*(q2,0) ∈ C0; δ*(q2,1) y δ*(q2,2) ∈ C1

δ*(q3,0) y δ*(q3,1) ∈ C0 y δ*(q3,2) ∈ C1

Q/E_1={{q4}, {q0,q1}, {q2} ,{q3}}

δ*(q0,0) ∈ C1, δ*(q0,1) ∈ C2 y δ*(q0,2) ∈ C3

δ*(q1,0) ∈ C1, δ*(q1,1) ∈ C2 y δ*(q1,2) ∈ C3

Q/E_2={{q4}, {q0,q1}, {q2} ,{q3}} **NO MÁS DIVISIONES**

Q/E_n={{q4}, {q0,q1}, {q2} ,{q3}}

Algoritmo de Minimización

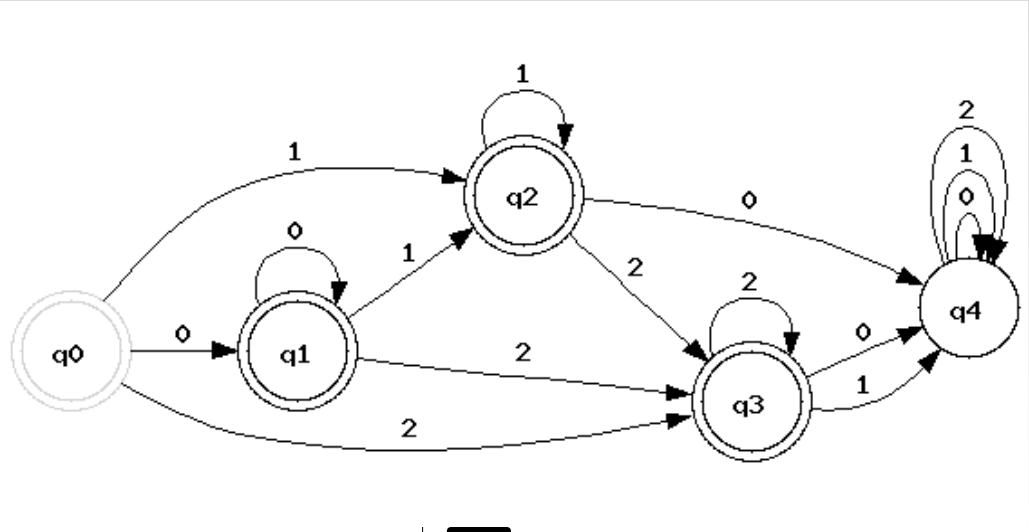
ENTRADA: un AFD $M_d = (Q_d, \Sigma, \delta_d, q_{0d}, F_d)$

SALIDA: un AFD Mínimo $M_m = (Q_m, \Sigma, \delta_m, q_{0m}, F_m)$ equivalente a M_d

1. $i=0$, $Q/E_0 = \{ Q_d\text{-}F, F \}$ donde $c_1 = Q_d\text{-}F$ y $c_2 = F$

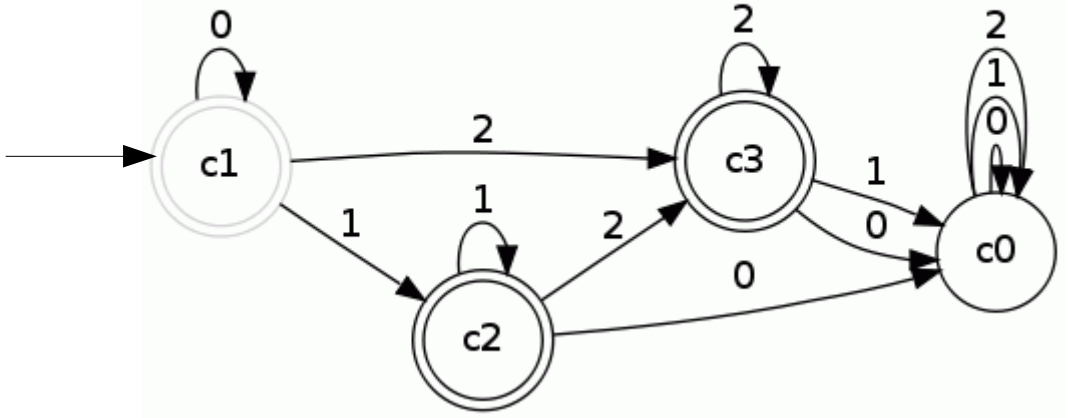
2. Para cualquier par $p,q \in c_j/E_i$ hacer
   Si $\forall a \in \Sigma$ se cumple que $\delta(p,a) \in c_i$ y $\delta(q,a) \in c_i$ entonces $p\ E_{i+1}\ q$
   en caso contrario $p\ E_{i+1}\ q$ (no equivalente)
3. Calcular Q/E_{i+1} poniendo en la misma clase a los pares equivalentes y en clases distintas a los no equivalentes.
   Si $Q/E_{i+1} \neq Q/E_i$, i++, volver a 2
   en caso contrario $Q/E_i = Q/E = Q_m$,
4. **Obtener M_m**
   $Q_m = Q/E_i \rightarrow$ **TANTOS ESTADOS COMO CLASES DE EQUIVALENCIA**
   $q_{om} = c_i \in Q_m \ / \ q_{od} \in c_i$
   $F_m = \{ ci \in Q_m \mid q_f \in c_i , q_f \in F_d \}$
   $\delta_m (c_i , a) = c_j$ / si $q_i \in c_i$ y $q_j \in c_j$ y $\delta_d (q_i, a) = q_j$

Algoritmo de Minimización



$Q/E_n = \{\{q4\}, \{q0,q1\}, \{q2\}, \{q3\}\}$

C0 C1 C2 C3



Algoritmo de Minimización

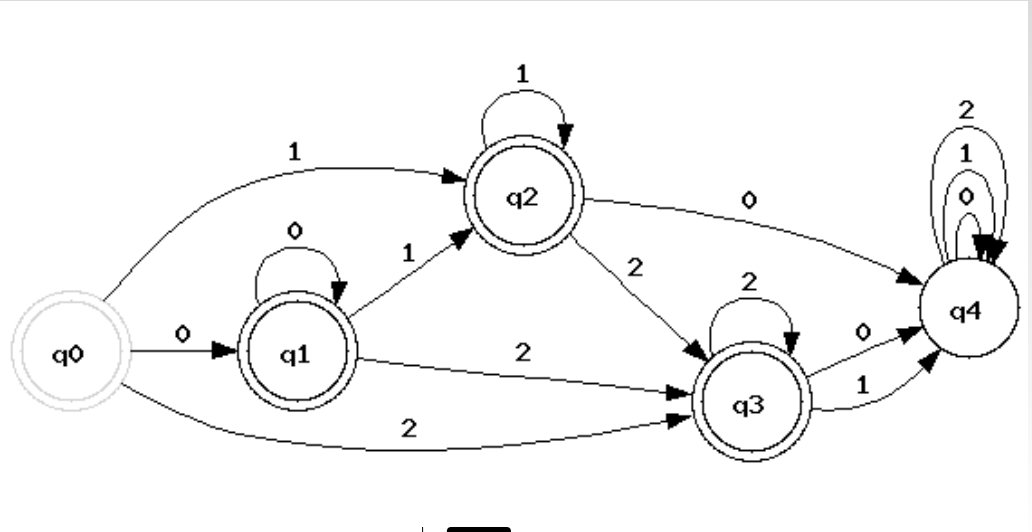
ENTRADA: un AFD $M_d = (Q_d, \Sigma, \delta_d, q_{0d}, F_d)$

SALIDA: un AFD Mínimo $M_m = (Q_m, \Sigma, \delta_m, q_{0m}, F_m)$ equivalente a M_d

1. $i=0$, $Q/E_0 = \{ Q_d\text{-}F, F \}$ donde $c_1 = Q_d\text{-}F$ y $c_2 = F$

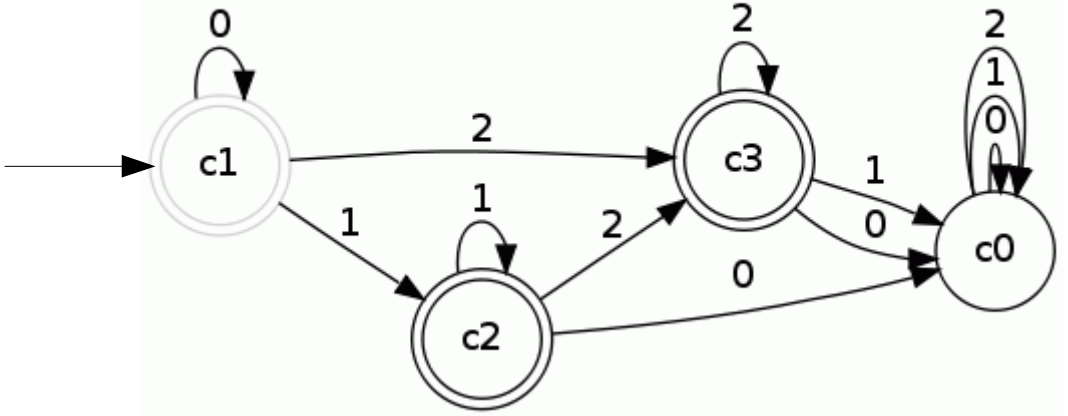
2. Para cualquier par $p,q \in c_j/E_i$ hacer
   Si $\forall a \in \Sigma$ se cumple que $\delta(p,a) \in c_i$ y $\delta(q,a) \in c_i$ entonces $p\ E_{i+1}\ q$
   en caso contrario $p\ E_{i+1}\ q$ (no equivalente)
3. Calcular Q/E_{i+1} poniendo en la misma clase a los pares equivalentes y en clases distintas a los no equivalentes.
   Si $Q/E_{i+1} \neq Q/E_i$, i++, volver a 2
   en caso contrario $Q/E_i = Q/E = Q_m$,
4. **Obtener M_m**
   $Q_m = Q/E_i$
   $q_{om} = c_i \in Q_m$ / $q_{od} \in c_i \rightarrow$ Estado Inicial clase que contiene el inicial del autómata origen
   $F_m = \{ ci \in Q_m \mid q_f \in c_i , q_f \in F_d \} \rightarrow$ Estados Finales: clases que contienen algún estado final del autómata origen
   $\delta_m (c_i , a) = c_j$ / si $q_i \in c_i$ y $q_j \in c_j$ y $\delta_d (q_i, a) = q_j$

Algoritmo de Minimización



$Q/E_n = \{\{q4\}, \{q0,q1\}, \{q2\}, \{q3\}\}$

C0 C1 C2 C3



Algoritmo de Minimización

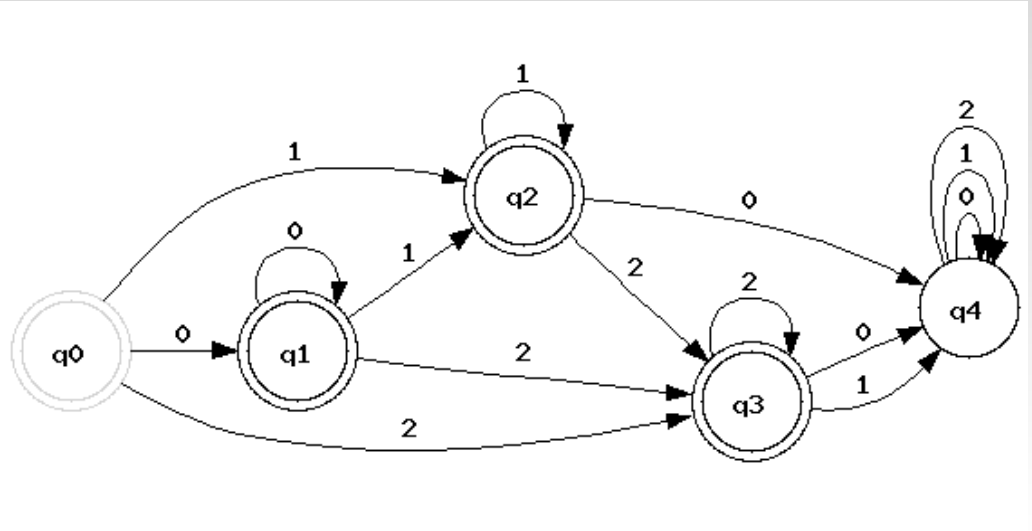
ENTRADA: un AFD $M_d = (Q_d, \Sigma, \delta_d, q_{0d}, F_d)$

SALIDA: un AFD Mínimo $M_m = (Q_m, \Sigma, \delta_m, q_{0m}, F_m)$ equivalente a M_d

1. $i=0$, $Q/E_0 = \{ Q_d\text{-}F, F \}$ donde $c_1 = Q_d\text{-}F$ y $c_2 = F$

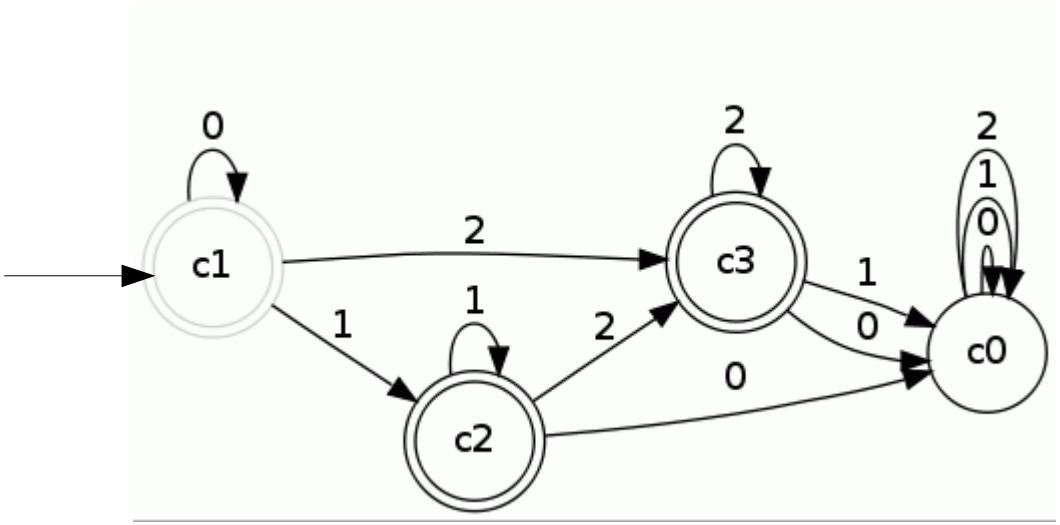
2. Para cualquier par $p,q \in c_j/E_i$ hacer
   Si $\forall a \in \Sigma$ se cumple que $\delta(p,a) \in c_i$ y $\delta(q,a) \in c_i$ entonces $p\ E_{i+1}\ q$
   en caso contrario $p\ E_{i+1}\ q$ (no equivalente)
3. Calcular Q/E_{i+1} poniendo en la misma clase a los pares equivalentes y en clases distintas a los no equivalentes.
   Si $Q/E_{i+1} \neq Q/E_i$, i++, volver a 2
   en caso contrario $Q/E_i = Q/E = Q_m$,
4. **Obtener M_m**
   $Q_m = Q/E_i$
   $q_{om} = c_i \in Q_m \ / \ q_{od} \in c_i$
   $F_m = \{ ci \in Q_m \mid q_f \in c_i , q_f \in F_d \}$
   $\delta_m (c_i , a) = c_j$ / si $q_i \in c_i$ y $q_j \in c_j$ y $\delta_d (q_i, a) = q_j$ → **RECALCULAR LAS TRANSICIONES: IMPORTANTE ES UN AFD**

Algoritmo de Minimización

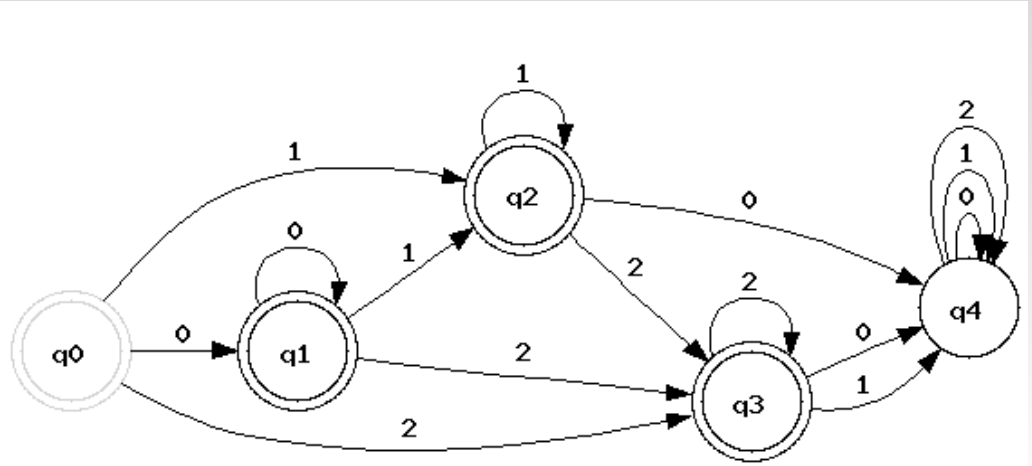


$Q/E_n = \{\{q4\}, \{q0,q1\}, \{q2\}, \{q3\}\}$

$\delta_m (c_1, 2) = c_3$ / si $q_0 \in c_1$ y $q_3 \in c_3$ y $\delta_d (q_0, 2) = q_3$



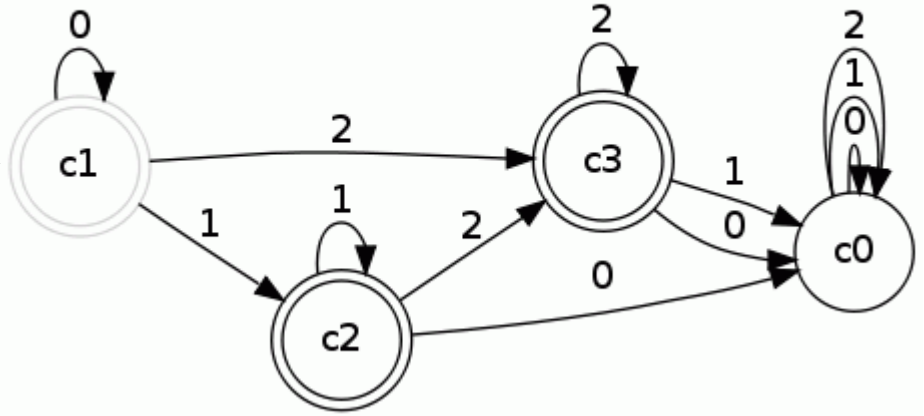
Algoritmo de Minimización



$Q/E_n = \{\{q4\}, \{q0,q1\}, \{q2\}, \{q3\}\}$

$\delta_m (c_1, 2) = c_3$ / si $q_0 \in c_1$ y $q_3 \in c_3$ y $\delta_d (q_0, 2) = q_3$

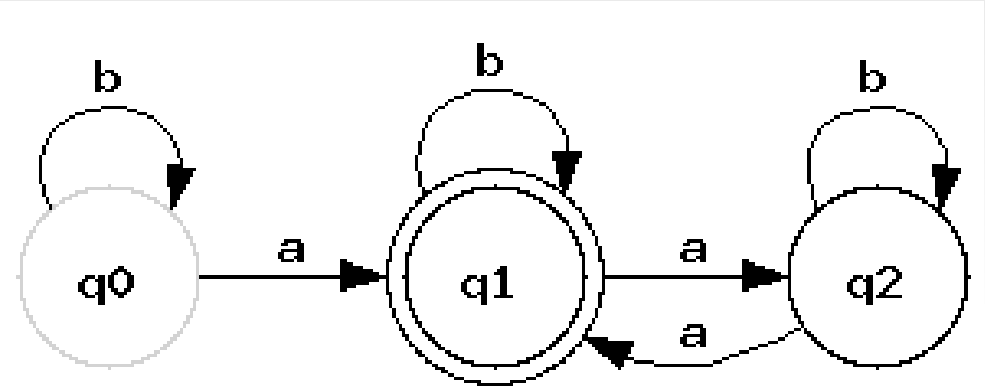
EJERCICIOS PROPUESTO: CALCULAR TODAS LAS TRANSICIONES



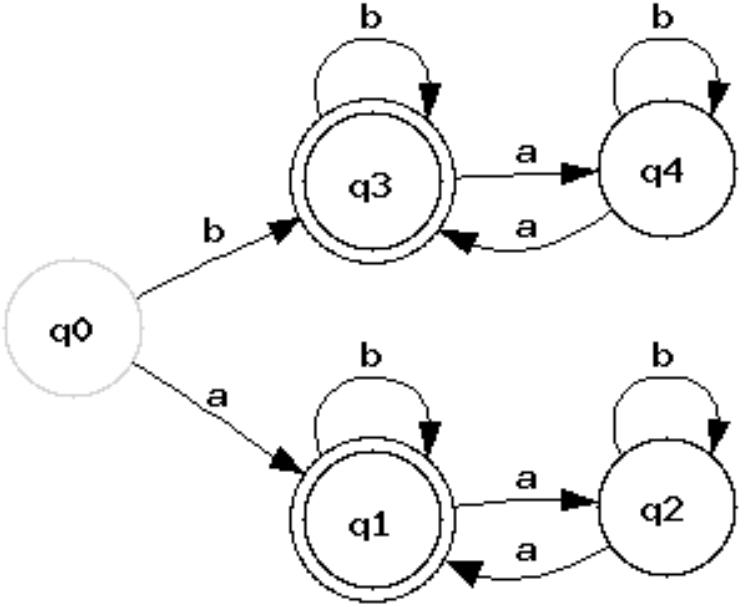
Ejercicios propuestos

Minimiza los siguientes autómatas.

AFD3:



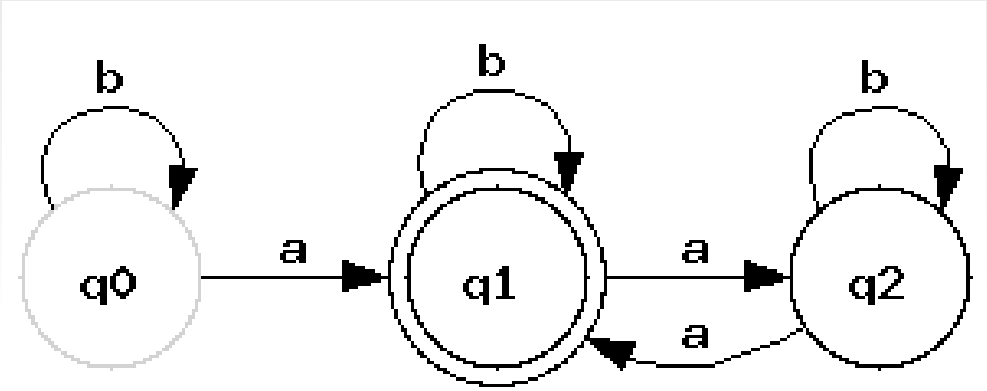
AFD4:



Ejercicios propuestos

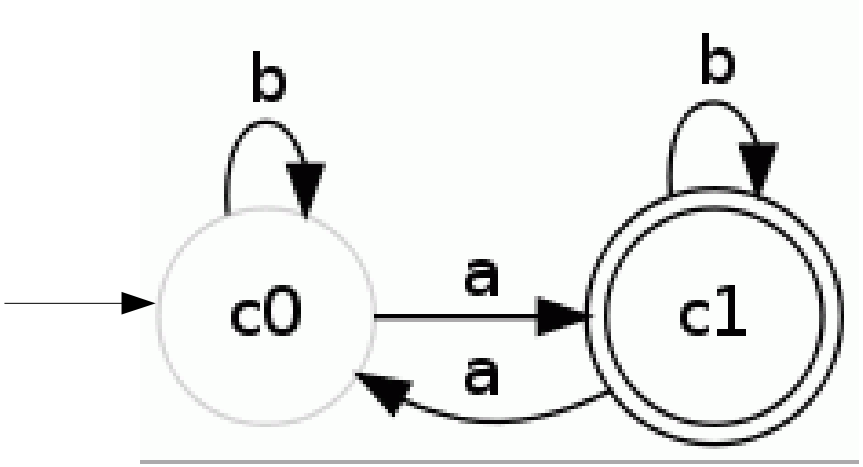
Soluciones:

AFD3:



C0: {q0,q2}

C1: {q1}



Ejercicios propuestos

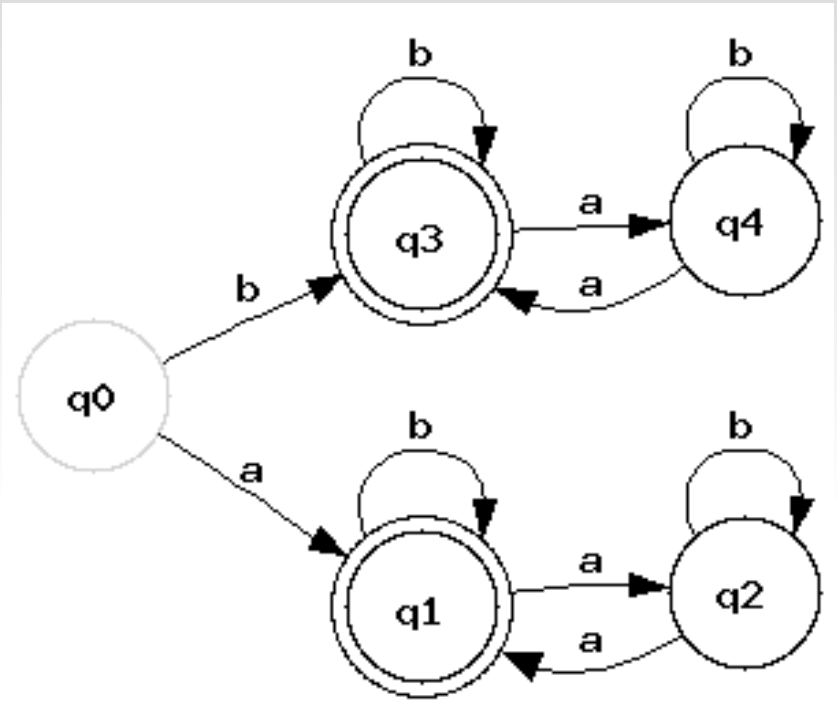
Soluciones:

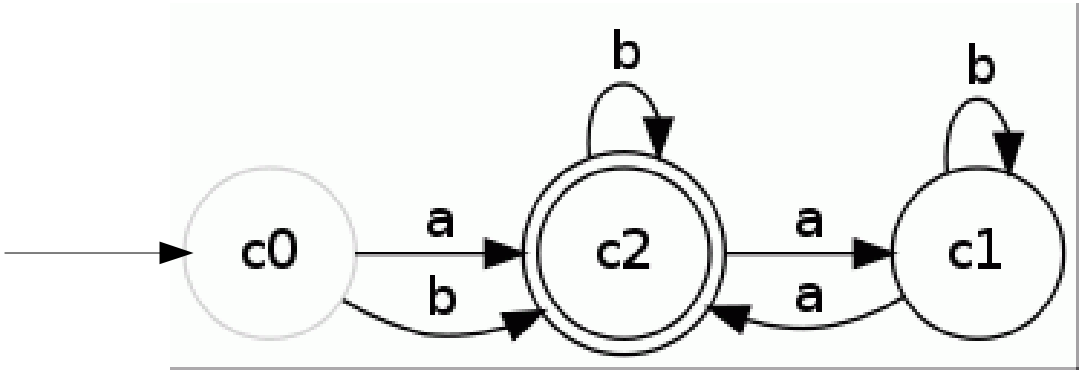
AFD4:

C0: {q0}

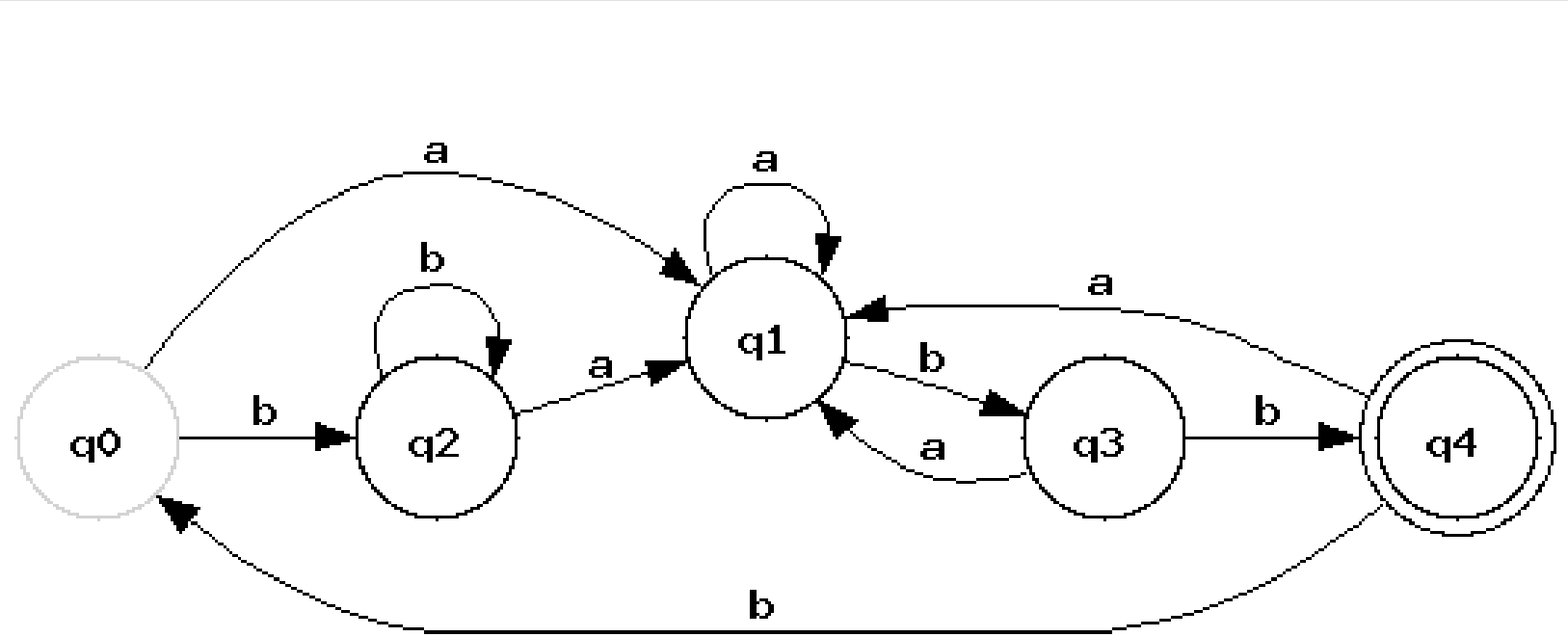
C1: {q4, q2}

C2: {q3, q1}





Ejercicios propuestos



Ejercicios propuestos

Solución:

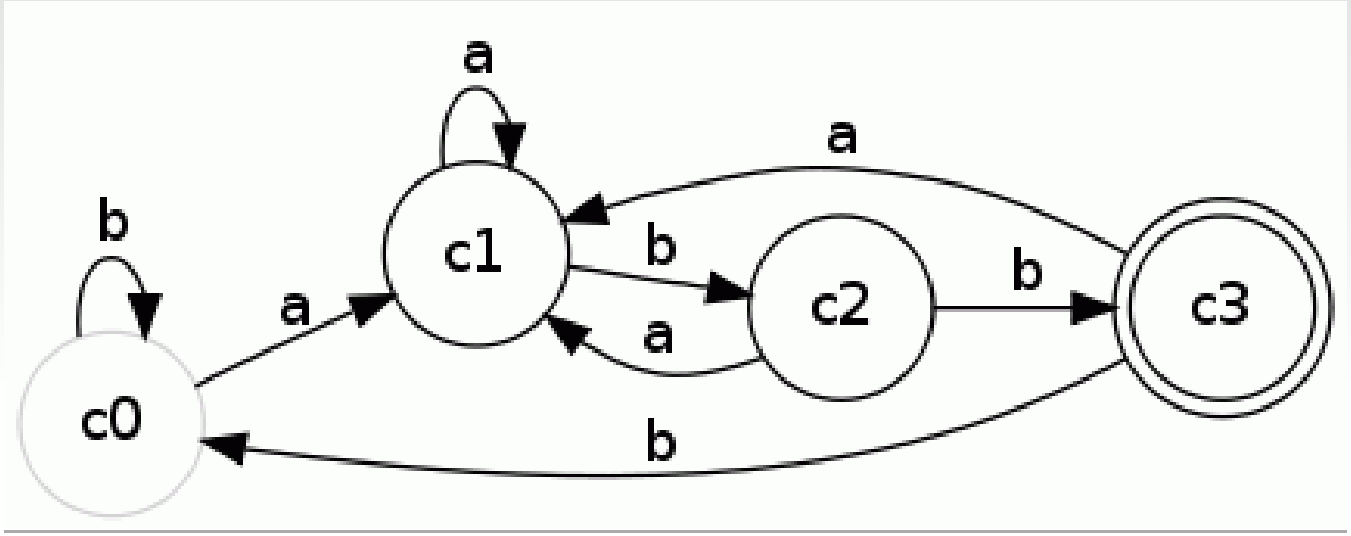
Q/E3=Q/E2

c0: {q0,q2}

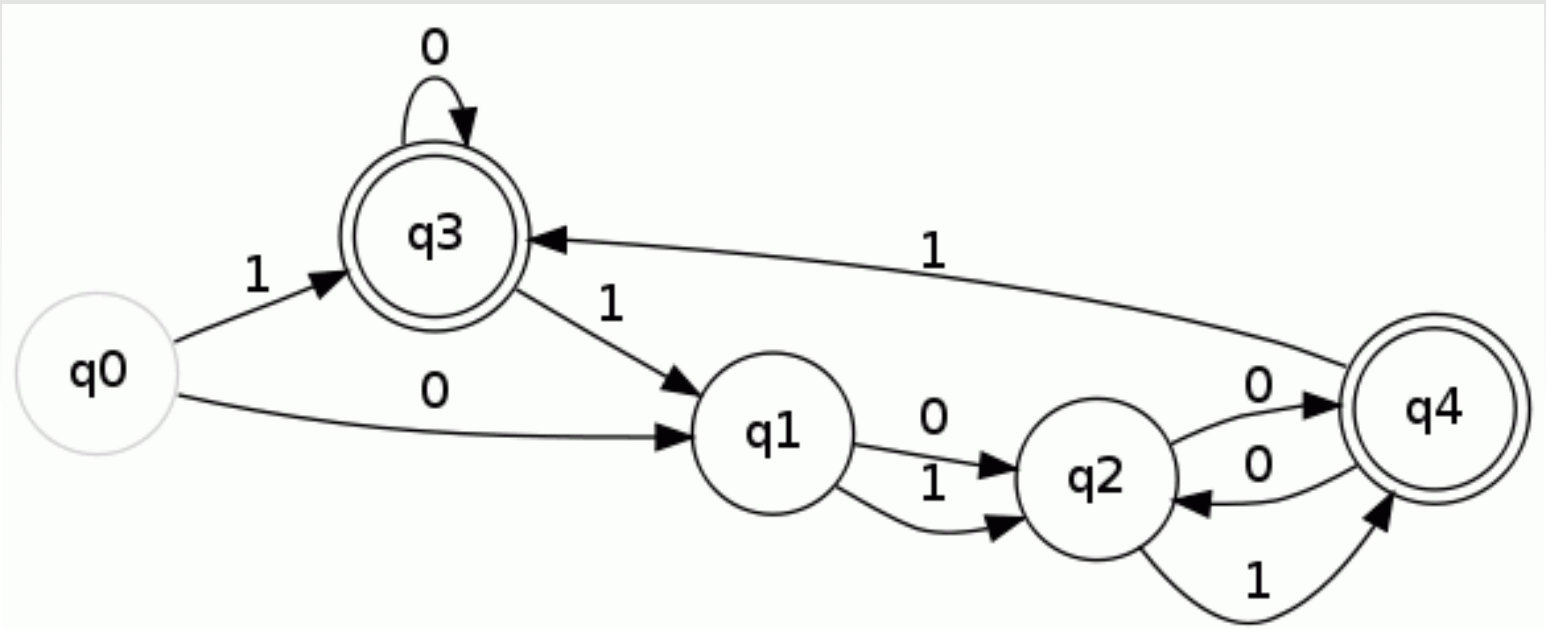
c1={q1}

c2={q3}

c3={q4}



Ejercicios propuestos



TEMA 4-2

EQUIVALENCIA ENTRE AUTÓMATAS FINITOS

Ejercicios propuestos

CÓDIGO SELFA AUTÓMATAS

```
automaton automata{

states c0,c1,c2,c3;

alphabet 0,1,2;

initial c1;

final c1,c2,c3;

transition{

c1,1=c2;

c1,0=c1;

c1,2=c3;


c2,1=c2;

c2,0=c0;

c2,2=c3;


c3,0=c0;

c3,1=c0;

c3,2=c3;

c0,0=c0;

c0,1=c0;

c0,2=c0;}

}print(automata);
```

Ejercicios propuestos

CÓDIGO SELFA e Ejercicio final minimización

```
automaton automata{

states c0,c1,c2,c3;

alphabet a,b;

initial c0;

final c3;

transition{

c0,a=c1;

c0,b=c0;

c1,a=c1;

c1,b=c2;

c2,a=c1;

c2,b=c3;

c3,b=c0;

c3,a=c1;

}

}


print(automata);
```

TEMA 5

EQUIVALENCIA ENTRE AUTÓMATAS FINITOS Y EXPRESIONES REGULARES

5.1 Generación de un AFND-λ a partir de una ER

5.2 Generación de una ER a partir de un AFND-λ

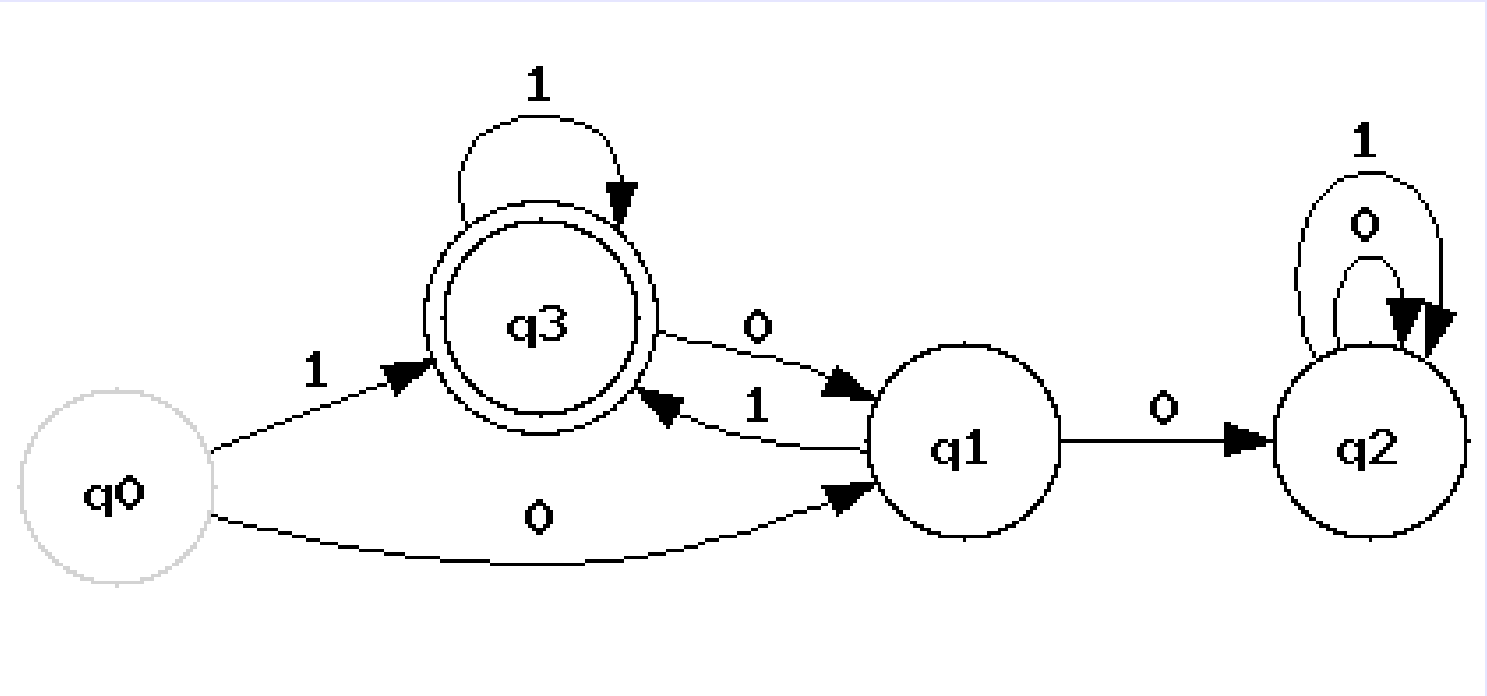
INTRODUCCIÓN

En el tema anterior hemos descrito de manera **informal** el proceso de:

- **Lenguaje** regular expresado en lenguaje natural → **AFD** que lo reconoce.

INTRODUCCIÓN

1.e). Lenguaje de todas las cadenas que terminen con un 1 y no contenga a la subcadena 00.



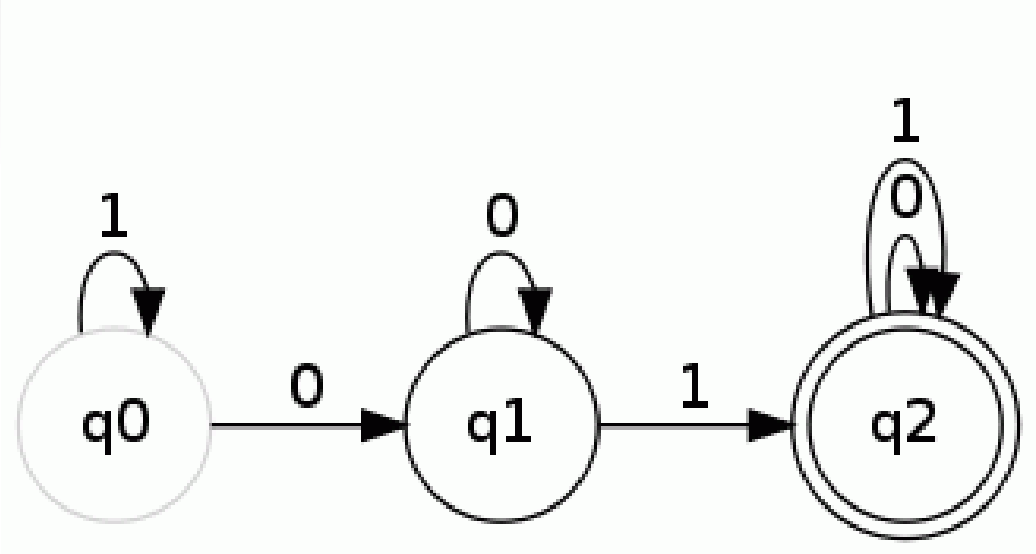
INTRODUCCIÓN

En el tema anterior hemos descrito de manera **informal** el proceso de:

- **Lenguaje** regular expresado en lenguaje natural → **AFD** que lo reconoce.
- **AF** → **Lenguaje** regular expresado en lenguaje natural que es reconocido por el autómata.

INTRODUCCIÓN

2.) Explicar cual es el lenguaje asociado a cada uno de los siguientes autómatas.



L = {x | x contiene la subcadena 01}

INTRODUCCIÓN

En este tema:

- Planteamos un conjunto de **algoritmos** que permiten realizar estas transformaciones **no** de manera **intuitiva** sino de manera *formal*.
- Sólo trabajamos con **Expresiones Regulares** (No Lenguaje Natural)
- Trabajamos con autómatas **finitos no deterministas con transiciones vacías.**

INTRODUCCIÓN

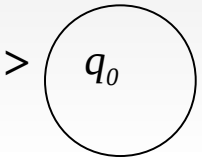
Se pueden diseñar algoritmos para estas transformaciones porque:

- Todo lenguaje definido por un **autómata finito** también puede definirse por una **expresión regular**.
- Todo lenguaje definido por una **expresión regular** puede definirse mediante uno de estos **autómatas**.

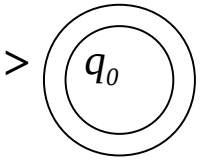
5.1 Generación de un AFND-λ a partir de una ER

Caso Base:

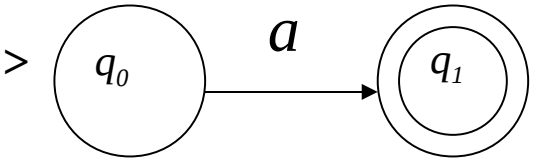
Si ER = ∅ entonces M es:



Si ER = λ entonces M es:

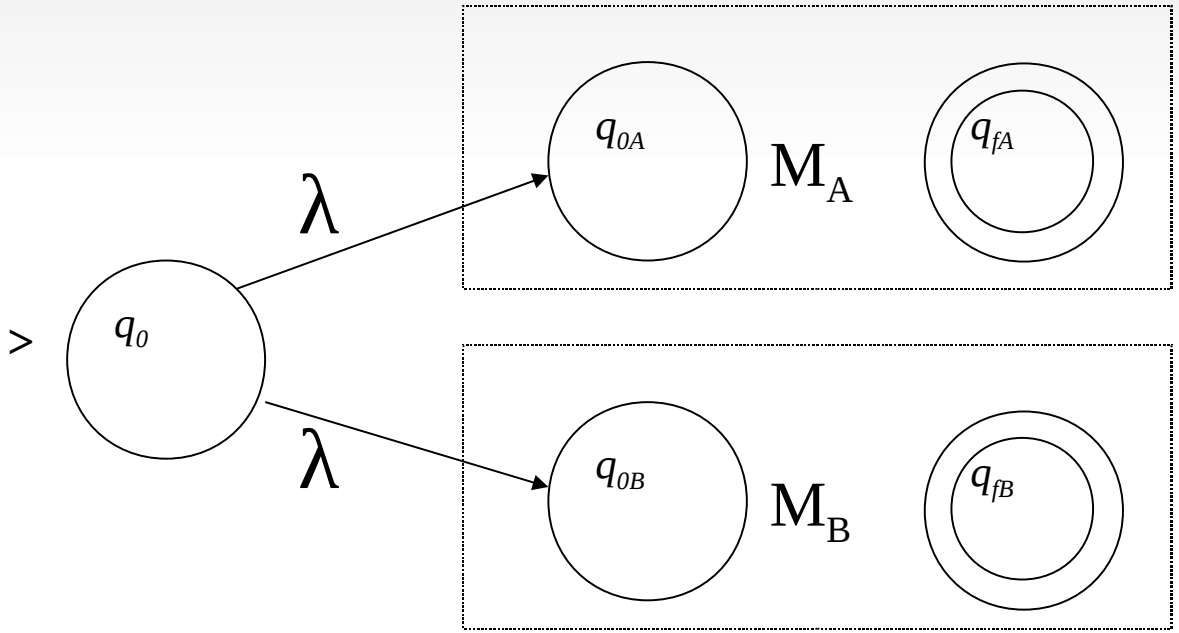


Si ER = a entonces M es:



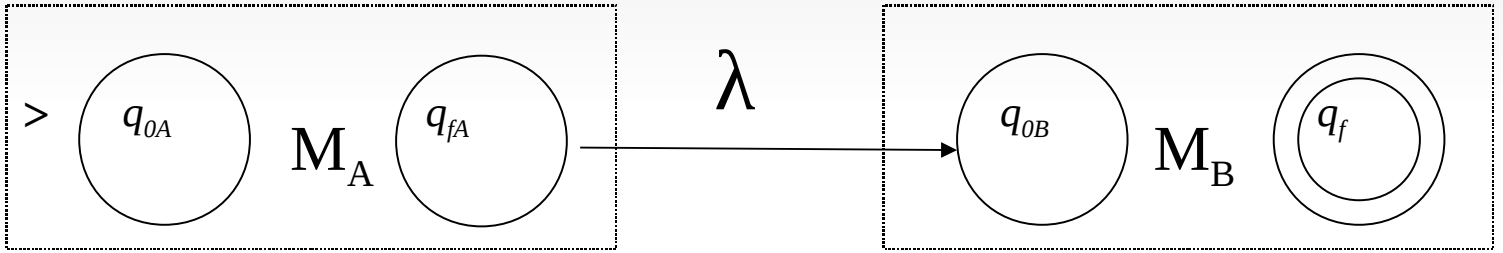
5.1 Generación de un AFND-λ a partir de una ER

R=A+B:



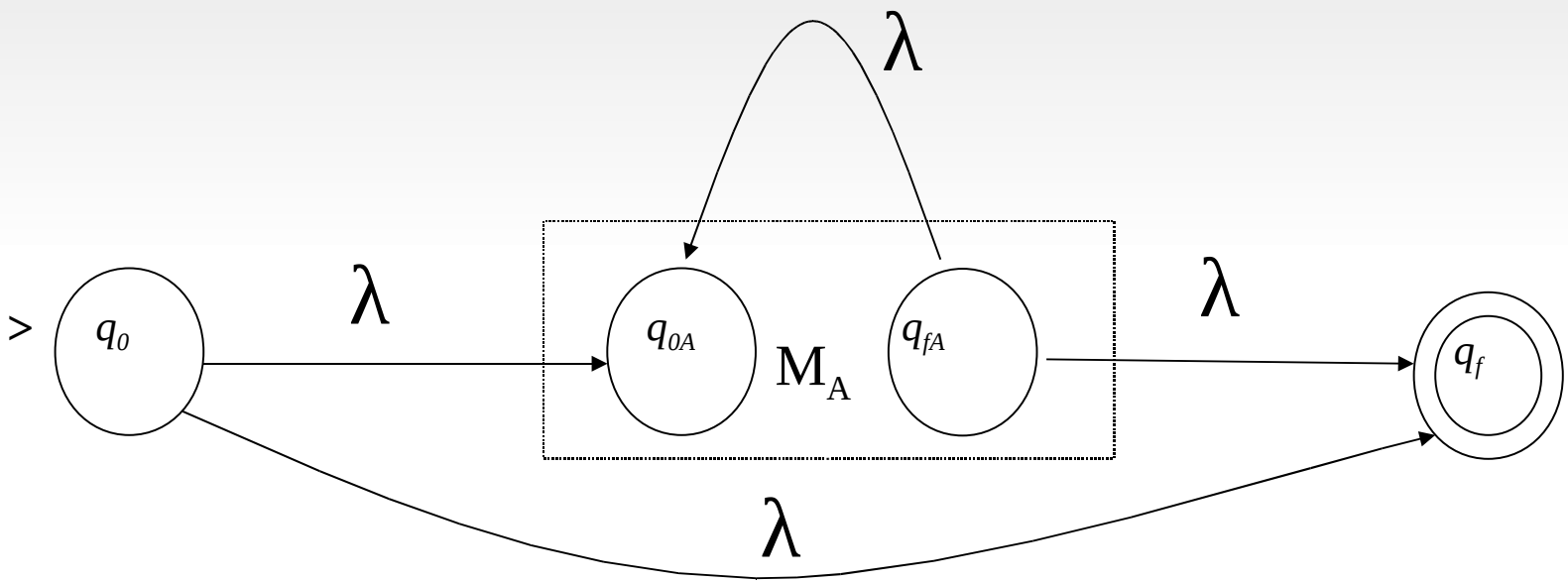
5.1 Generación de un AFND-λ a partir de una ER

R=AB:



5.1 Generación de un AFND-λ a partir de una ER

R=A*

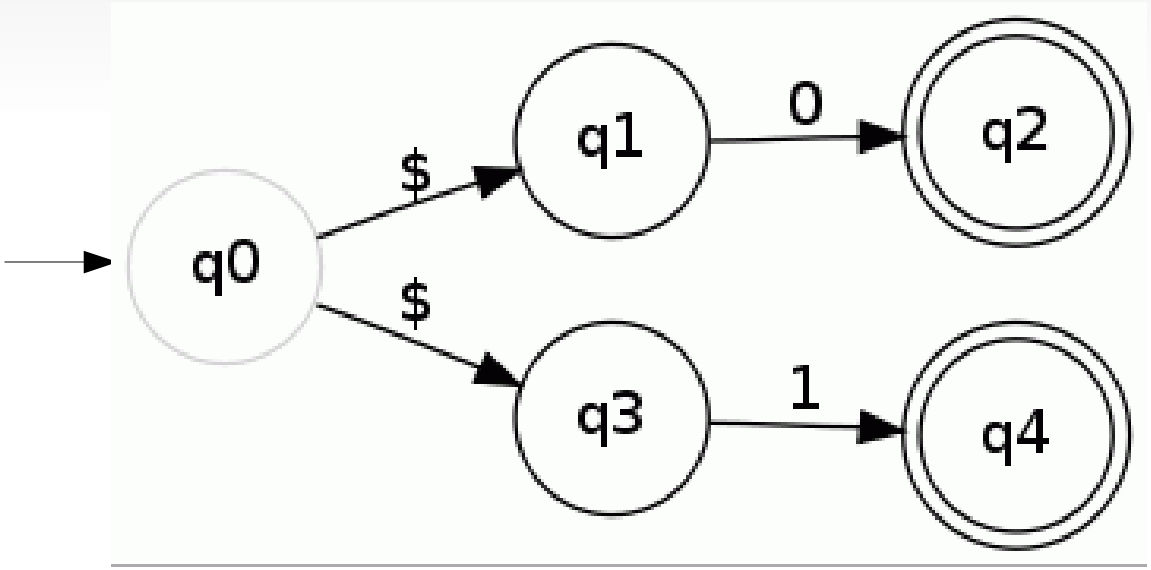


5.1 Generación de un AFND-λ a partir de una ER

EJEMPLO: (0+1)*1(0+1)

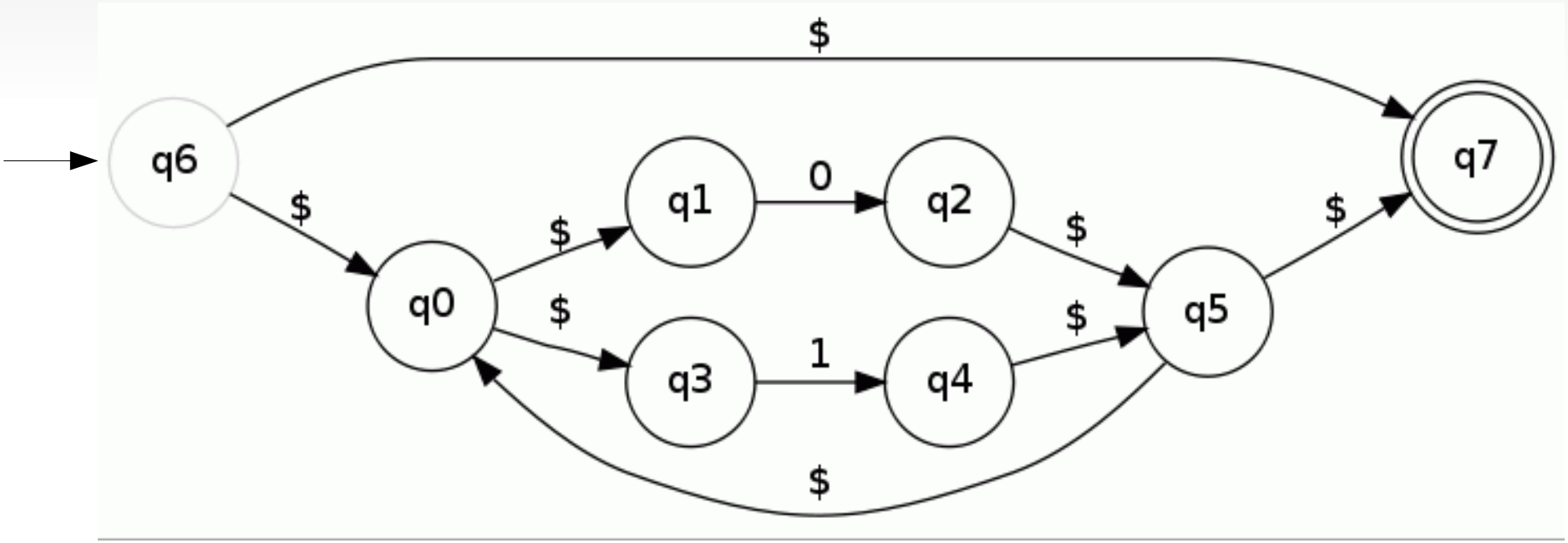
5.1 Generación de un AFND-λ a partir de una ER

EJEMPLO: (0+1)*1(0+1)



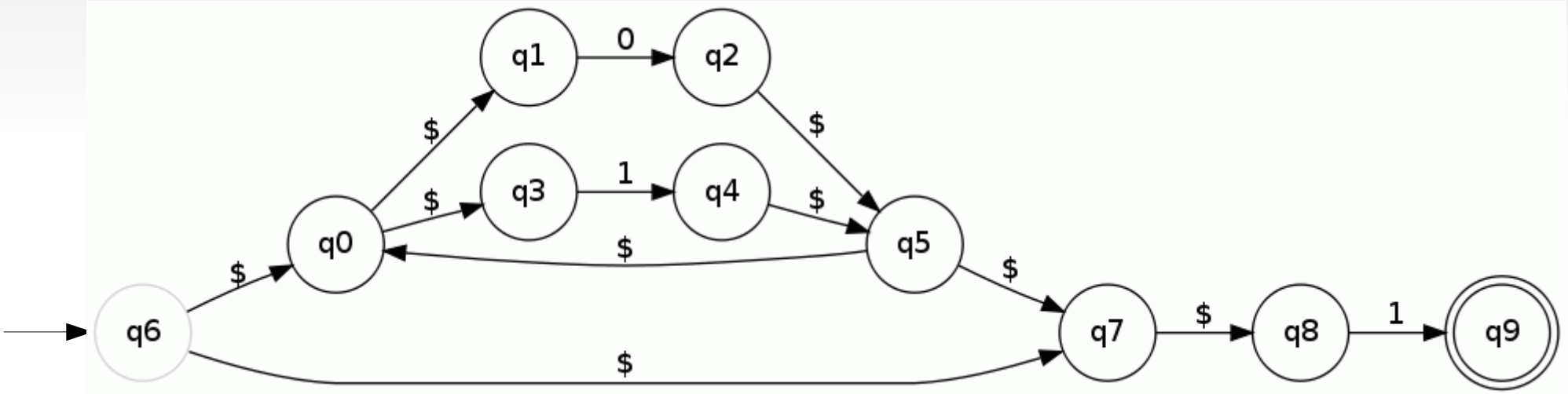
5.1 Generación de un AFND-λ a partir de una ER

EJEMPLO: (0+1)*1(0+1)



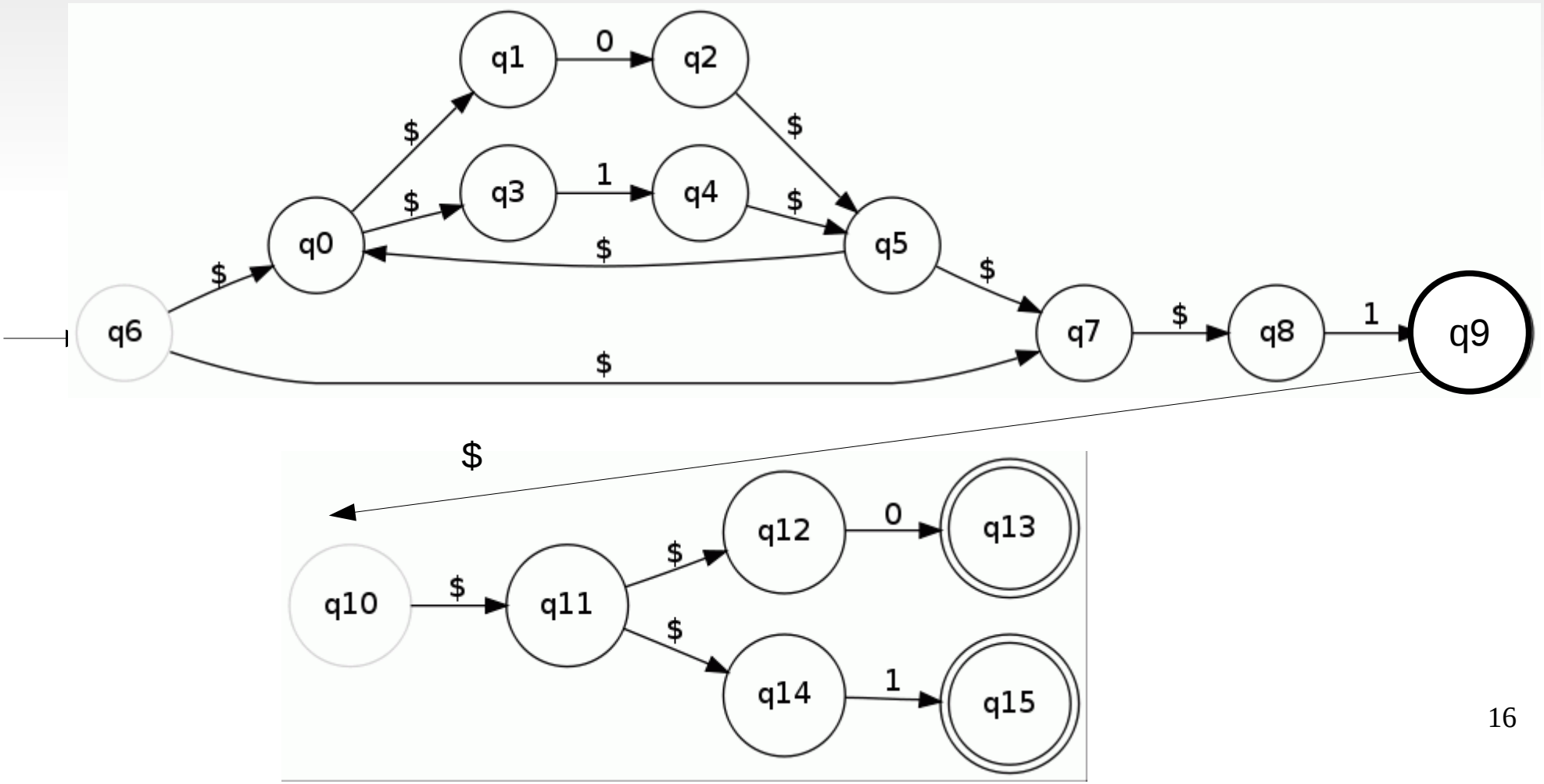
5.1 Generación de un AFND-λ a partir de una ER

EJEMPLO: (0+1)*1(0+1)



5.1 Generación de un AFND-λ a partir de una ER

EJEMPLO: (0+1)*1(0+1)



Convertir las siguientes expresiones regulares a AFND-λ

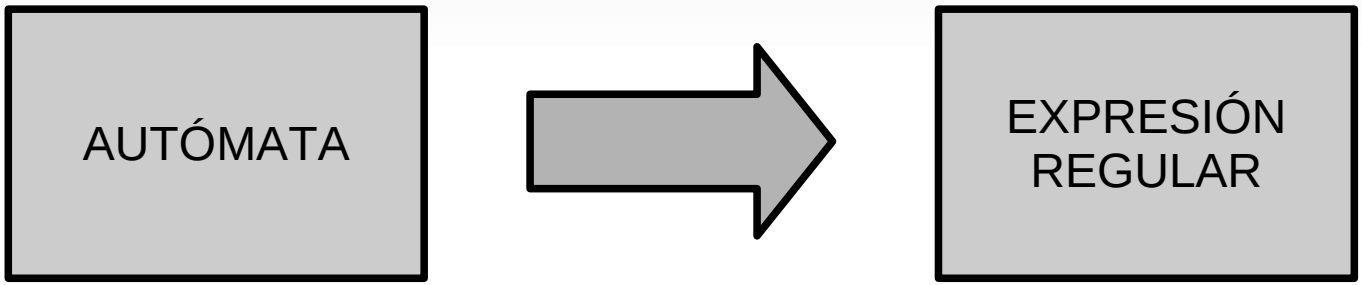
a) 01*

b) (0+1)01

c) 00(0+1)*

5.2 Generación de una ER a partir de un AFND-λ

Teorema: Para cada Autómata Finito M existe una expresión regular R tal que **L(M)= L(R)**

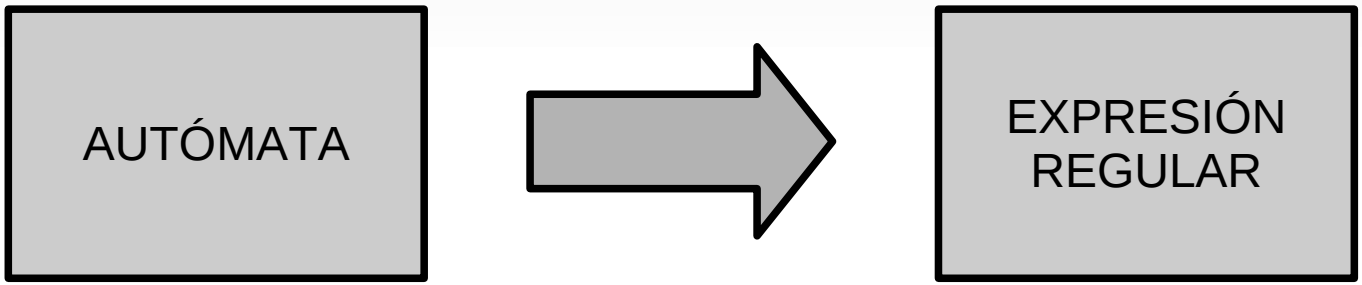


Tres métodos:

1. Método de las r^n_{ij} (Hopcroft)
2. Eliminación de Estados (Hopcroft, Linz)
3. Ecuaciones Características (Isasi)

5.2 Generación de una ER a partir de un AFND-λ

Teorema: Para cada Autómata Finito M existe una expresión regular R tal que **L(M)= L(R)**



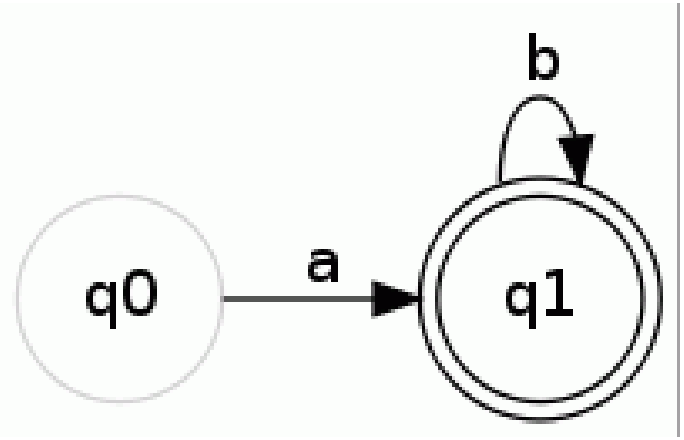
Tres métodos:

1. Método de las r^n_{ij} (Hopcroft)
2. Eliminación de Estados (Hopcroft, Linz)
3. **Ecuaciones Características (Isasi)**

Definición de Ecuación Característica

Cada estado del autómata tiene una ecuación característica que describe las distintas formas de llegar desde dicho estado a un estado final. $q_i \rightarrow x_i =$

x_i se define como el conjunto de palabras que M **reconocería** si se ejecuta a partir del estado **q_i**.



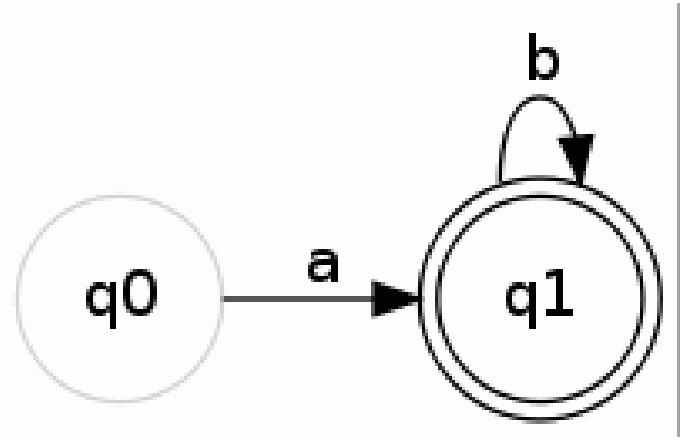
$x_0 = ax_1$

$x_1 = b^+ + \lambda = b^*$

Definición de Ecuación Característica

Cada estado del autómata tiene una ecuación característica que describe las distintas formas de llegar desde dicho estado a un estado final. $q_i \rightarrow x_i =$

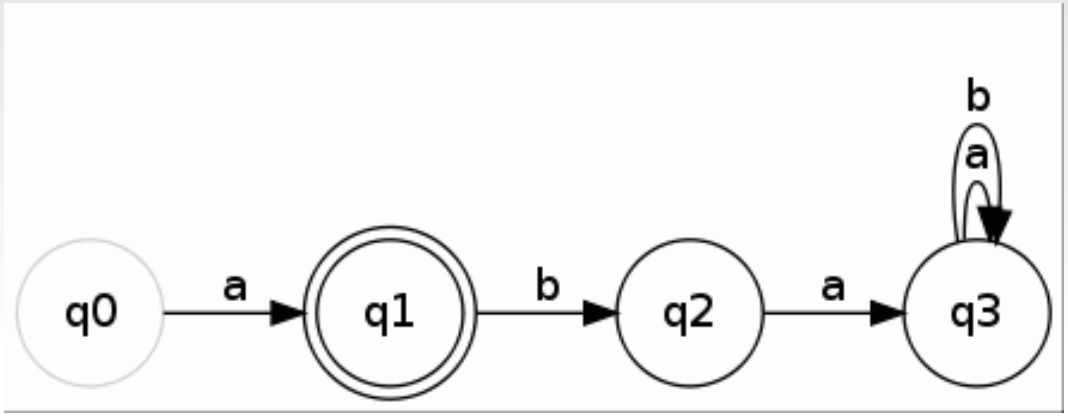
x_0 por tanto define a las palabras reconocidas por el AF



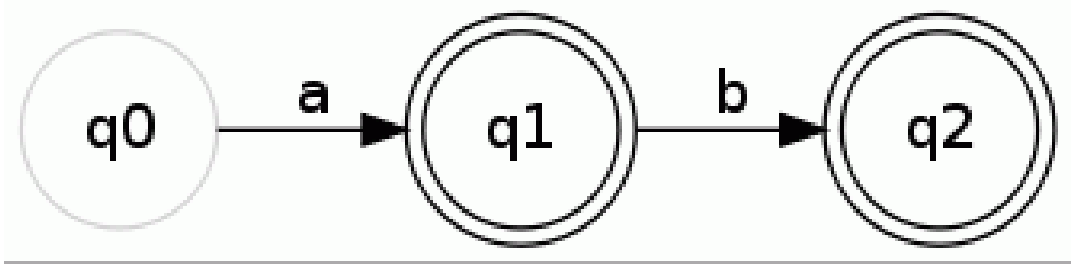
$L(M) = x_0 = ab^*$

Definición de Ecuación Característica

¿Cuál sería el conjunto de palabras reconocido desde q3?

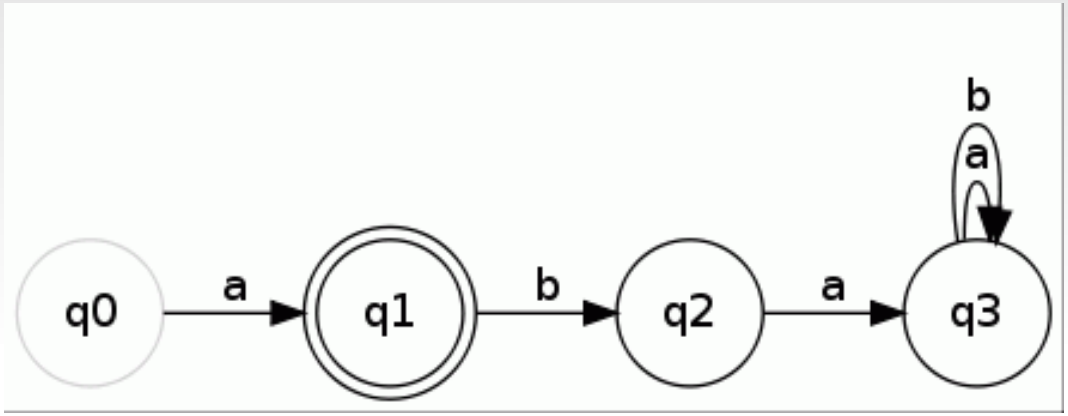


¿Cuál sería el conjunto de palabras reconocido desde q2?



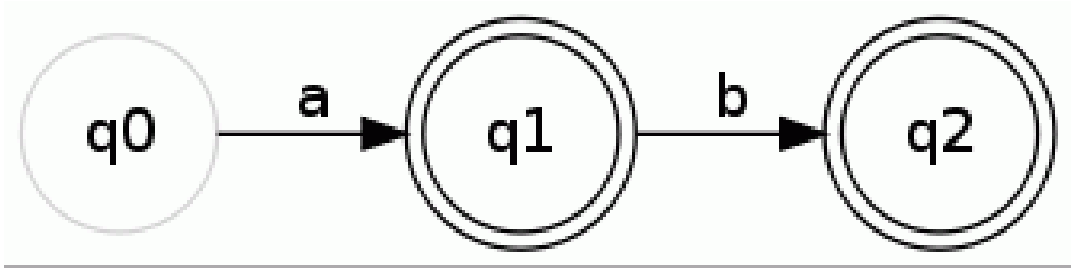
Definición de Ecuación Característica

¿Cuál sería el conjunto de palabras reconocido desde q3?



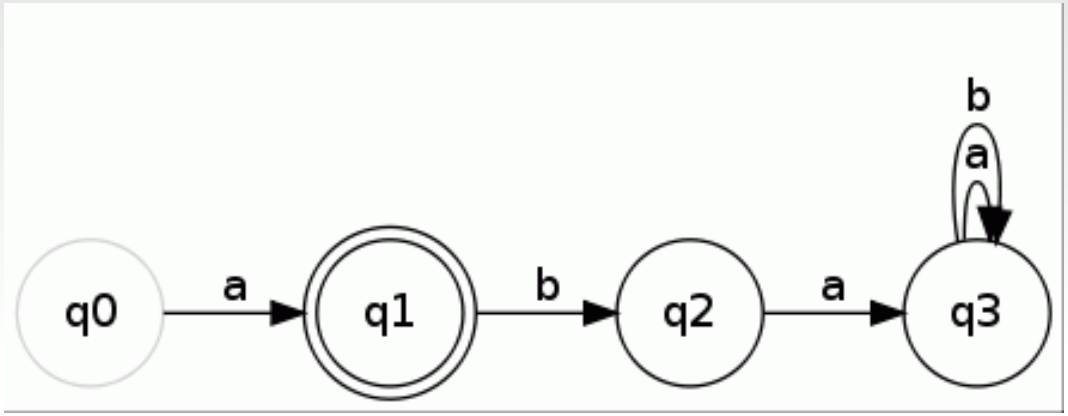
$X_3 = \emptyset$

¿Cuál sería el conjunto de palabras reconocido desde q_2?



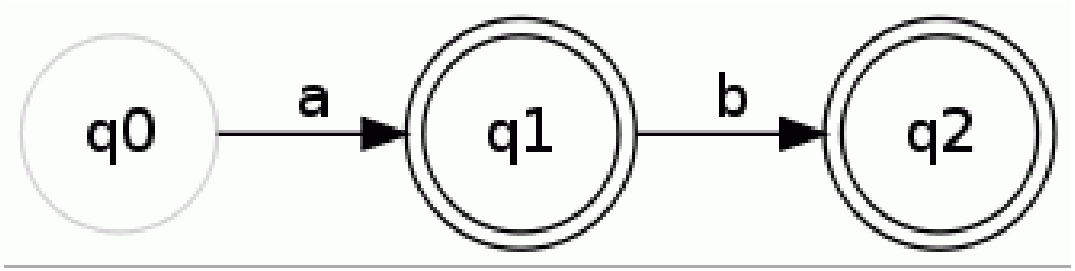
Definición de Ecuación Característica

¿Cuál sería el conjunto de palabras reconocido desde q3?



$X_3 = \emptyset$

¿Cuál sería el conjunto de palabras reconocido desde q_2?



$x_2 = \lambda$

Definición de Ecuación Característica

La **ecuación característica** para el estado q_i es la siguiente:

$$x_i = b_j x_j + b_k x_k + \ldots + b_w x_w + a_i$$

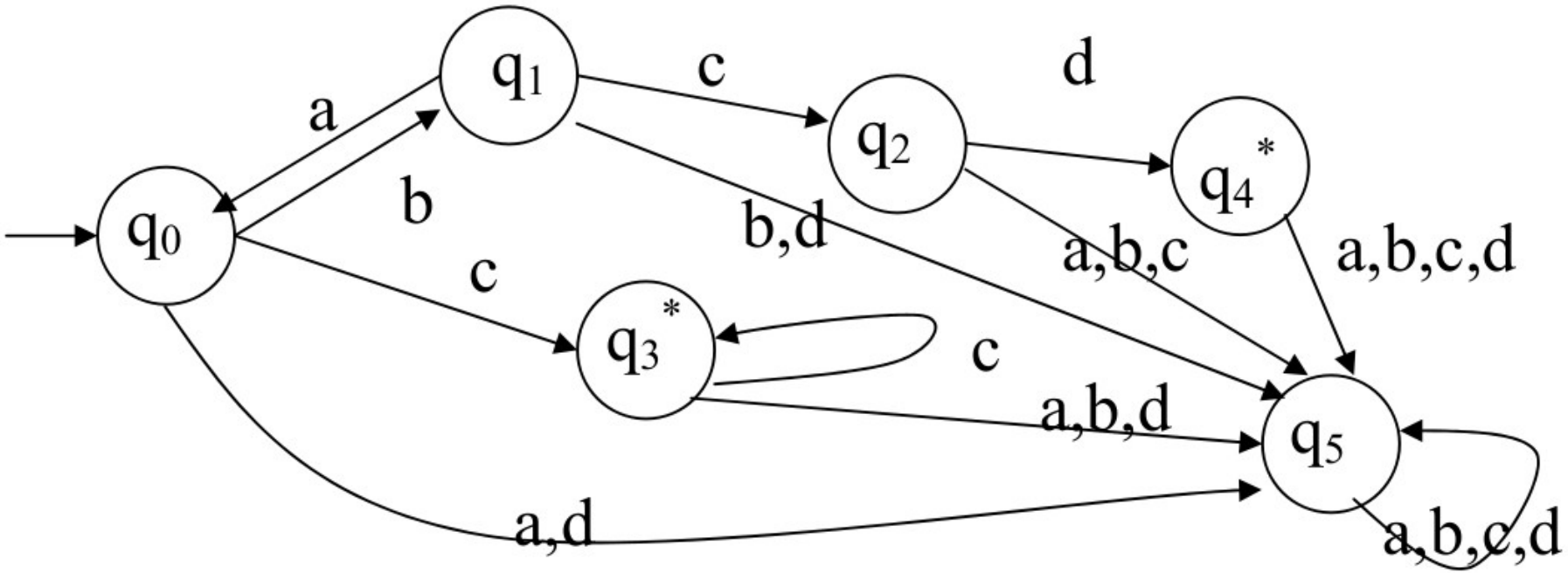
Donde:

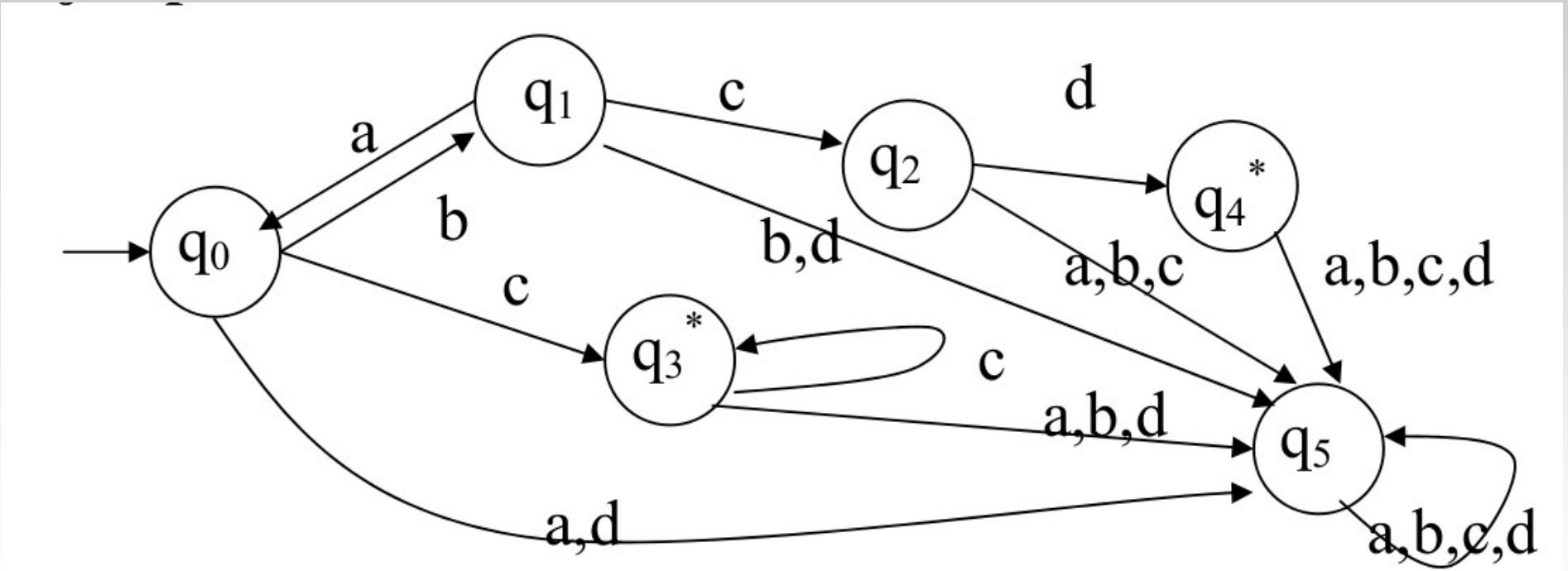
- La expresión $b_k x_k$ forma parte de la ecuación **si y sólo si** existe una transición del estado q_i al estado q_k para el símbolo de entrada b_k (Si $\delta(q_i, a) = q_j$ entonces **a**x_j también pertenece a x_i)
- a_i es una expresión tal que $a_i = \lambda$ si q_i es un estado final y $a_i = \emptyset$ en otro caso. (Si $qi \in F$ entonces $\lambda \in xi$ ....)

EJERCICIOS

Extrae las ecuaciones características del siguiente autómata

(Nota: q3 y q4 son los estados finales.)





$x0 = bx1 + cx3 + ax5 + dx5 = bx1 + cx3 + (a + d)x5$

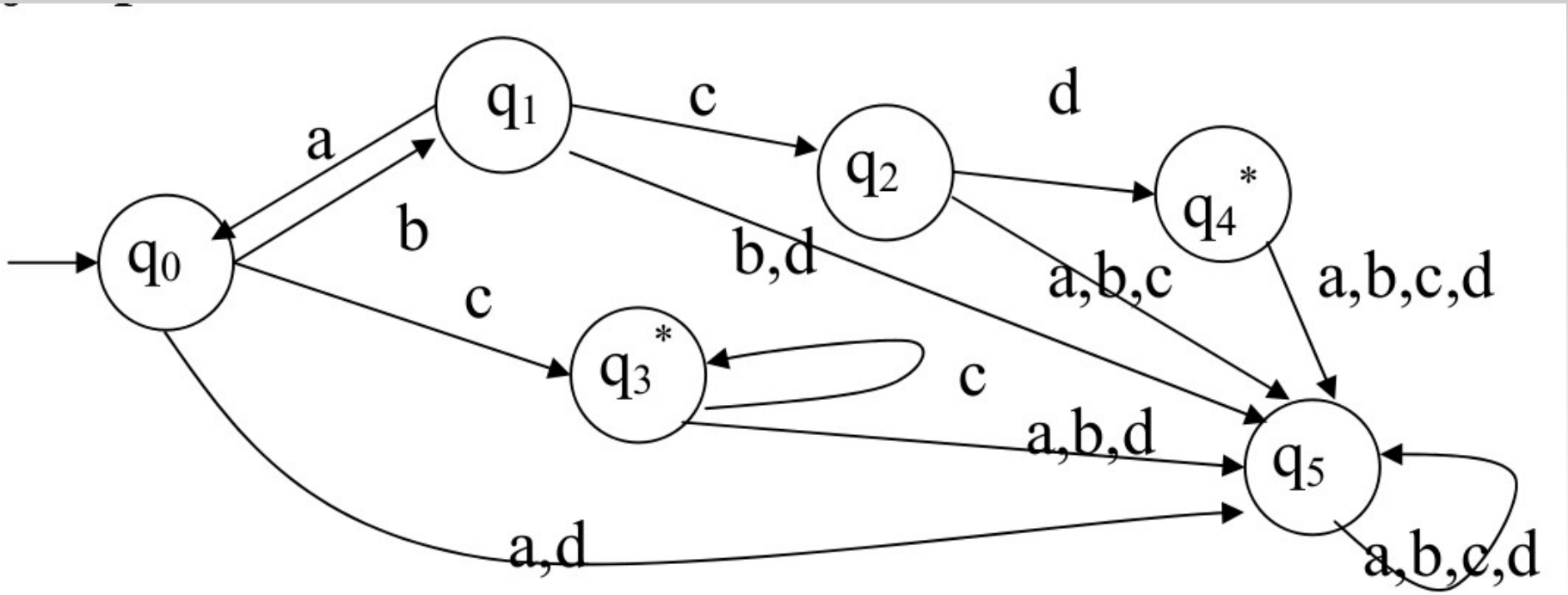
$x1 = ax0 + cx2 + (b+d)x5$

$x2 = dx4 + (a+b+c)x5$

$x3 = cx3 + (a+b+d)x5 + \lambda$

$x4 = (a+b+c+d)x5 + \lambda$

$x5 = (a+b+c+d)x5$



$x0 = bx1 + cx3 + ax5 + dx5 = bx1 + cx3 + (a + d)x5 + \emptyset$

$x1 = ax0 + cx2 + (b+d)x5 + \emptyset$

$x2 = dx4 + (a+b+c)x5 + \emptyset$

$x3 = cx3 + (a+b+d)x5 + \lambda$

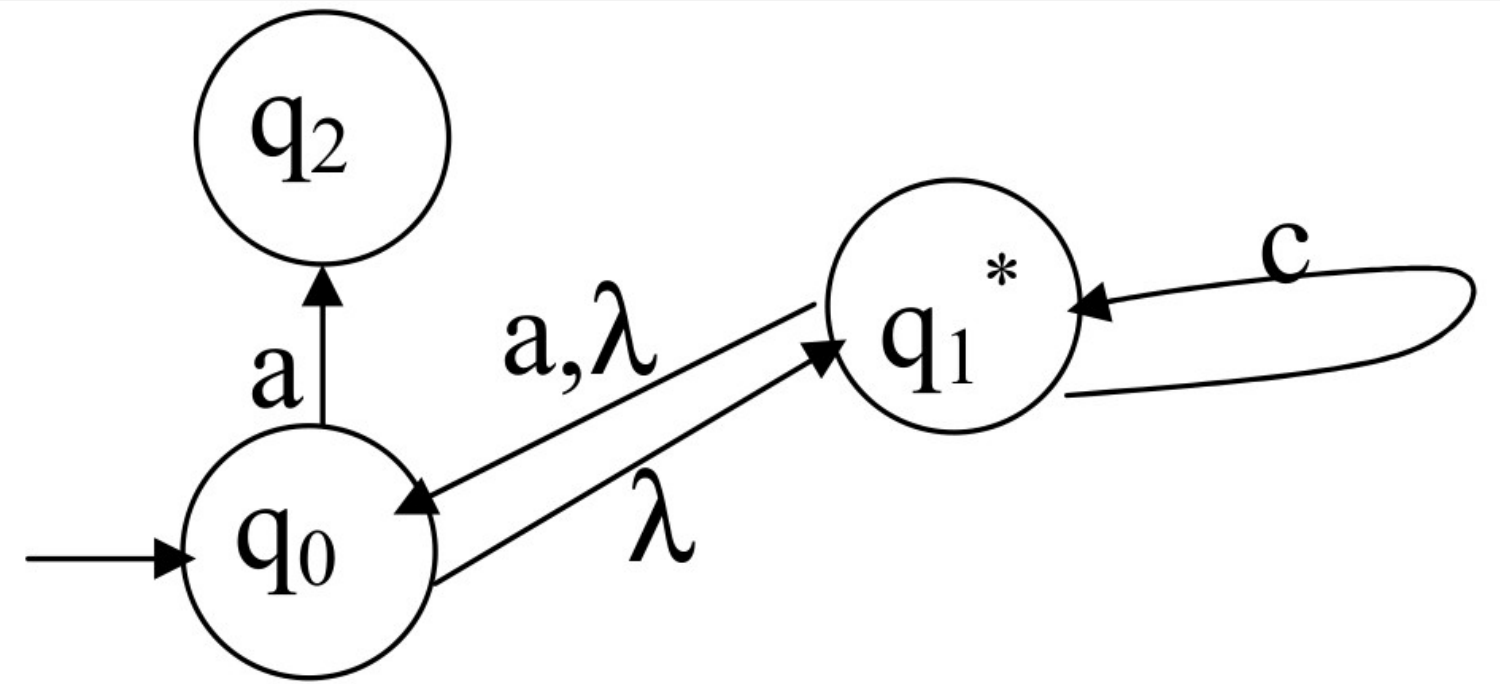
$x4 = (a+b+c+d)x5 + \lambda$

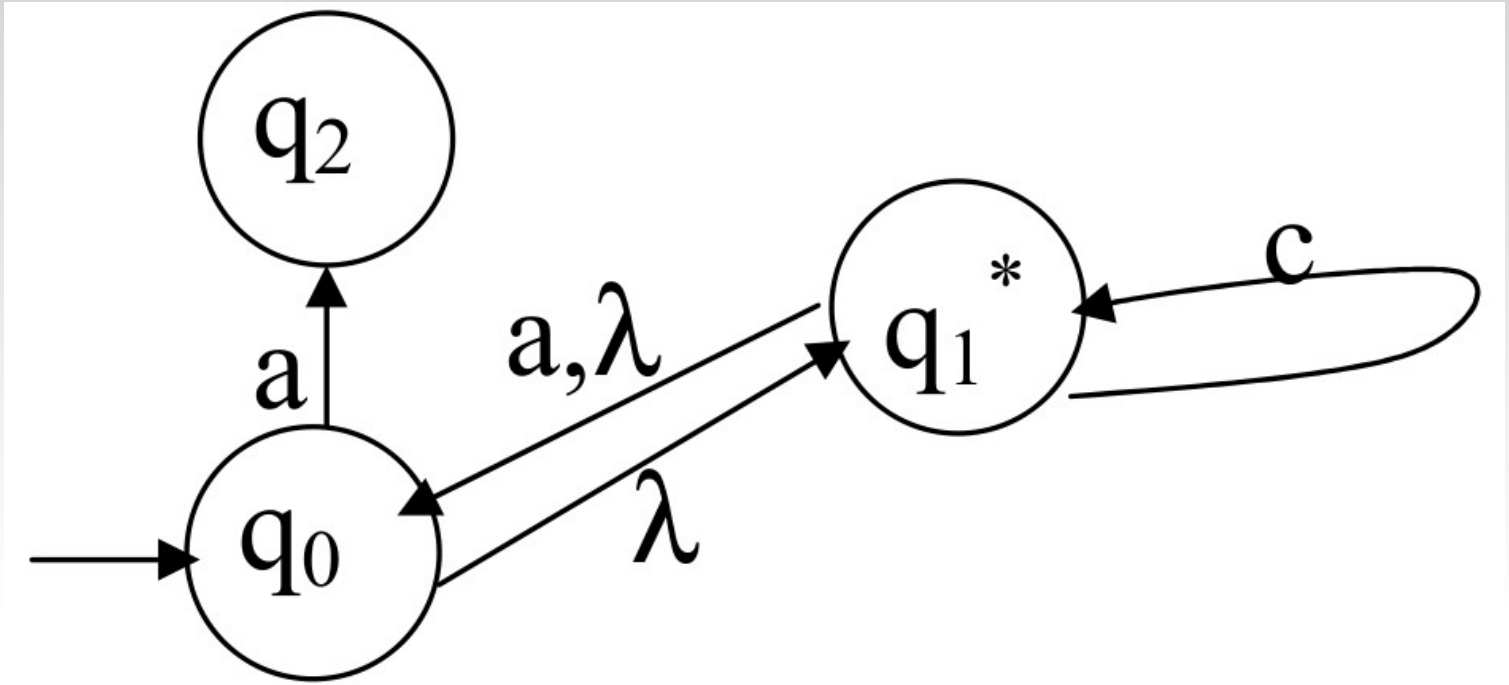
$x5 = (a+b+c+d)x5 + \emptyset$

EJERCICIOS

Extrae las ecuaciones características del siguiente autómata

(Nota: q1 es el estado final)

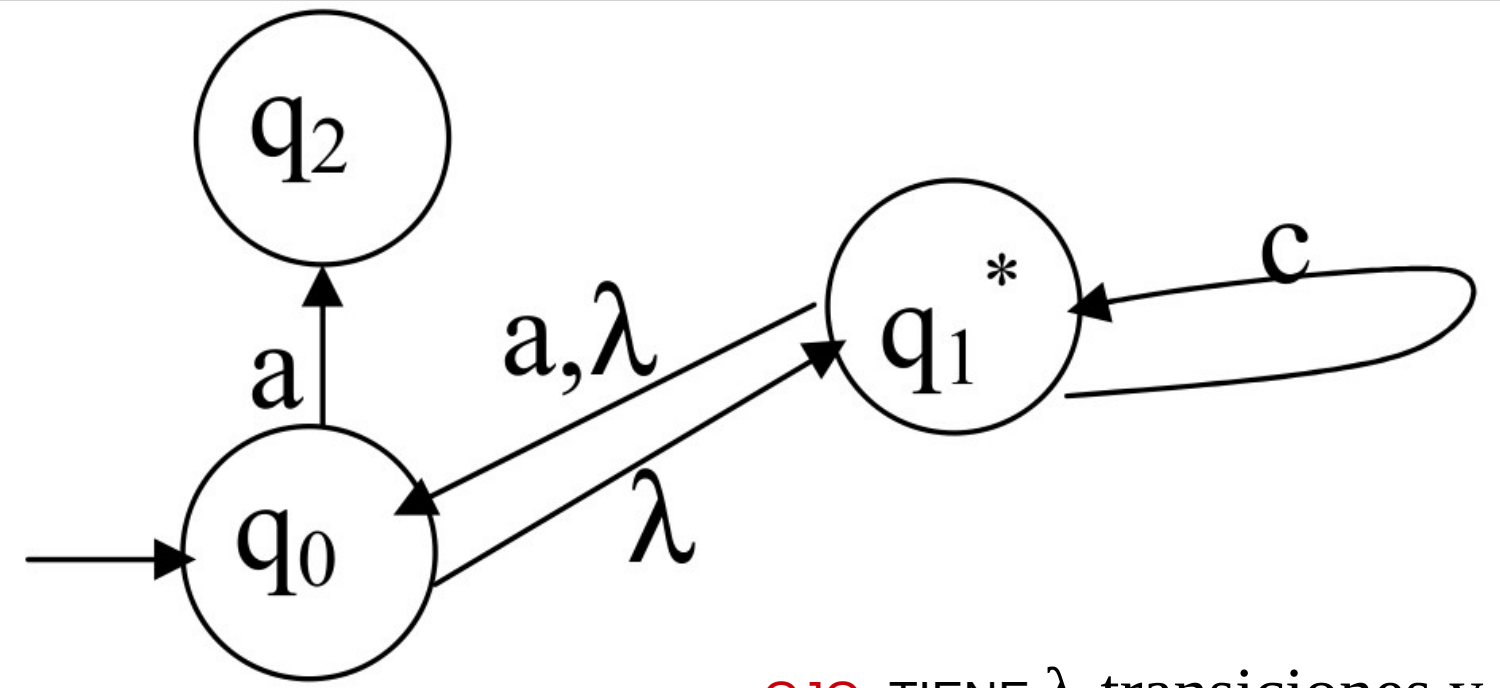




x0 = ax2 + λx1

x1 = cx1 + (a+λ)x0 + λ

x2= Ø



OJO: TIENE λ-transiciones y puede incumplir la regla general de resolución que vamos a ver más adelante.

$x0 = ax2 + \lambda x1$

$x1 = cx1 + (a+\lambda)x0 + \lambda$

$x2 = \emptyset$

Definición de Ecuación Característica

Se va a formar un sistema de ecuaciones donde las incógnitas son x_i

despejando x_0 para obtener las palabras que pertenecen a L(M) con la siguiente regla principal:

- Sea X una variable y A y B expresiones regulares:

Si X=A X + B y λ $\notin$ L(A), entonces X = A* B

Ejemplos de resolución de ecuaciones:

- X=abX → X=abX + Ø =(ab)*Ø = Ø
- X=abX + λ → X=(ab)*λ=(ab)*
- X=abX + cX → X=(ab+c)X=(ab+c)X+Ø=(ab+c)*Ø=Ø

EJERCICIOS

Obtén la expresión regular R tal que L(M)= L(R)

$X_0 = 1X_0 + 0X_1$

$X_1 = 1X_1 + 1X_0 + 0X_2$

$X_2 = 0X_0 + \lambda$

No hay estado sumidero y por tanto ninguna variable lleva al conjunto vacío.

EJERCICIOS

Obtén la expresión regular R tal que L(M)= L(R)

x0= bx1 + cx3 + ax5 + dx5 = bx1 + cx3 + (a + d)x5 + Ø

x1 = ax0 + cx2 + (b+d)x5 + Ø

x2 = dx4 + (a+b+c)x5 + Ø

x3 = cx3 + (a+b+d)x5 + λ

x4 =(a+b+c+d)x5 +λ

x5 = (a+b+c+d)x5 + Ø

x5 es estado sumidero y por tanto simplifica el resto de ecuaciones

EJERCICIOS

Obtén la expresión regular R tal que L(M)= L(R)

$X_0 = aX_0 + bX_1 + cX_3$

$X_1 = aX_1 + bX_3 + cX_2 + \lambda$

$X_2 = aX_1 + bX_3 + cX_2$

$X_3 = aX_3 + bX_3 + cX_3$

Aquí no se ha añadido el " + Ø" a los estados no finales pero es como si estuviese.

EJERCICIOS

Obtén la expresión regular R tal que L(M)= L(R)

$X_0 = 0X_1 + 1X_0 + \lambda$

$X_1 = 0X_2 + 1X_0 + \lambda$

$X_2 = 0X_2 + 1X_2 + \emptyset$

TEMA 5

EQUIVALENCIA ENTRE AUTÓMATAS FINITOS Y EXPRESIONES REGULARES

5.1 Generación de un AFND-λ a partir de una ER

5.2 Generación de una ER a partir de un AFND-λ

TEMA 5

automaton automata{

states q0,q1,q2,q3,q4,q5,q6;

alphabet 0,1;

initial q0;

final q5;

transition{

q0,$=q1;

q0,$=q3;

q3,1=q4;

q1,0=q2;

q2,$=q5;

q4,$=q5;

}

}

print(automata);

TEMA 5

automaton automata{

states q0,q1,q2,q3,q4,q5,q6,q7;

alphabet 0,1;

initial q6;

final q7;

transition{

q6,$=q0;

q0,$=q1;

q0,$=q3;

q3,1=q4;

q1,0=q2;

q2,$=q5;

q4,$=q5;

q5,$=q7;

q6,$=q7;

q5,$=q0;

}

}

print(automata);

automaton automata{

states q0,q1,q2,q3,q4,q5,q6,q7,q8,q9;

alphabet 0,1;

initial q6;

final q9;

transition{

q6,$=q0;

q0,$=q1;

q0,$=q3;

q3,1=q4;

q1,0=q2;

q2,$=q5;

q4,$=q5;

q5,$=q7;

q6,$=q7;

q5,$=q0;

q7,$=q8;

q8,1=q9;

}

}

print(automata);

TEMA 5

automaton automata{

states q10,q11,q12,q13,q14,q15,q16;

alphabet 0,1;

initial q10;

final q16;

transition{

q10,$=q11;

q11,$=q12;

q11,$=q14;

q14,1=q15;

q12,0=q13;

q13,$=q16;

q15,$=q16;

}

}

print(automata);

TEMA 6

PROPIEDADES DE LOS LENGUAJES REGULARES

PROPIEDADES DE CIERRE

Nos permiten **construir reconocedores** para un lenguaje **a partir** de **otros reconocedores** de otros lenguajes, p.e. “la intersección de dos lenguajes regulares es también regular.”

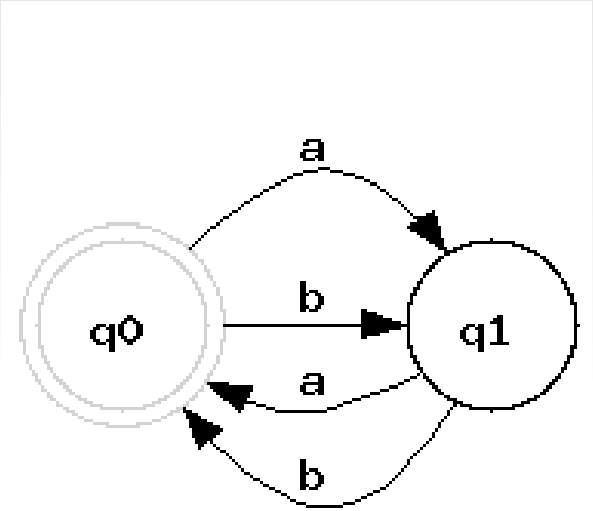
PROPIEDADES DE CIERRE

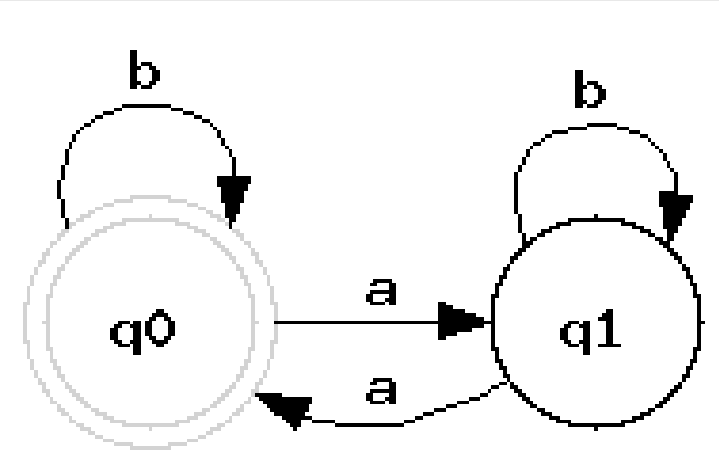
- **Unión**, **clausura** y **concatenación** ya visto.
- **Complemento**: Cambiar estados finales por no finales.
- **Intersección**: Mediante algoritmo.
- **Diferencia**: Aplicando **intersección** entre un lenguaje y el **complemento** del otro.

PROPIEDADES DE CIERRE

Diseñar un autómata para el alfabeto {a, b} en el que las palabras sean de longitud par y además contienen un número par de a's

PROPIEDADES DE CIERRE





PROPIEDADES DE CIERRE

ENTRADA: $M_A=(Q_A,\Sigma, q_{0A},\delta_A,F_A)$ y $M_B=(Q_B,\Sigma, q_{0B}, \delta_B,F_B)$.

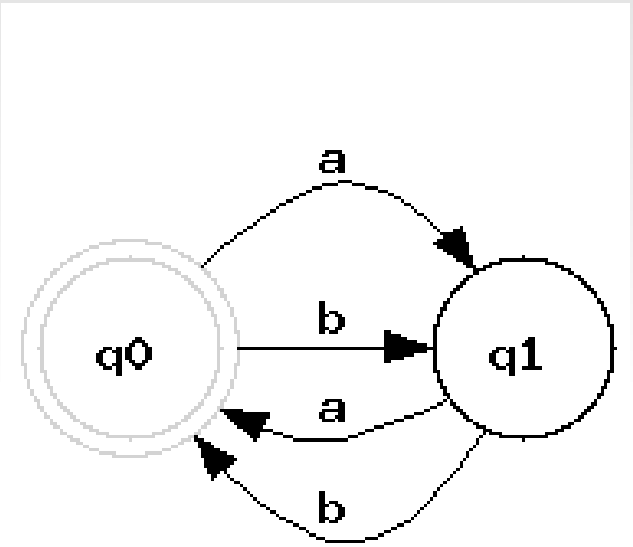
SALIDA: $M =(Q,\Sigma, q_0, \delta, F)$, tal que $L(M)= L_A \cap L_B$

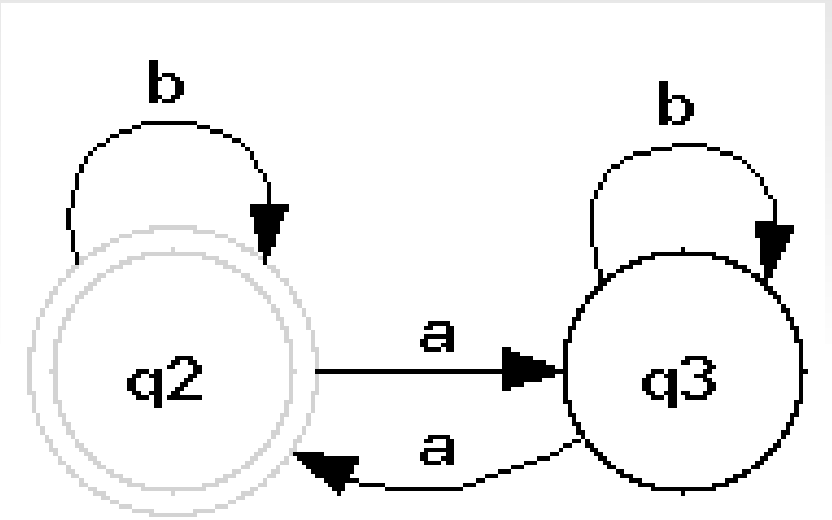
1. $q_0 = (q_{0A}, q_{0B})$; $Q = \{\}$; $EstNoDef = \{(q_{0A}, q_{0B})\}$

2. Mientras $EstNoDef \neq \emptyset$ hacer
   (a) Sacar un estado (p,q) de EstNoDef; EstNoDef = EstNoDef - {(p,q) }
   (b) $Q = Q \cup \{(p,q)\}$
   (c) Para todo $a \in \Sigma$ hacer
   $\delta ((p,q) , a) = (\delta_A (p, a), \delta_B (q, a)) = (p',q')$
   Si $(p',q') \notin Q$ entonces $EstNoDef = EstNoDef \cup \{(p',q') \}$

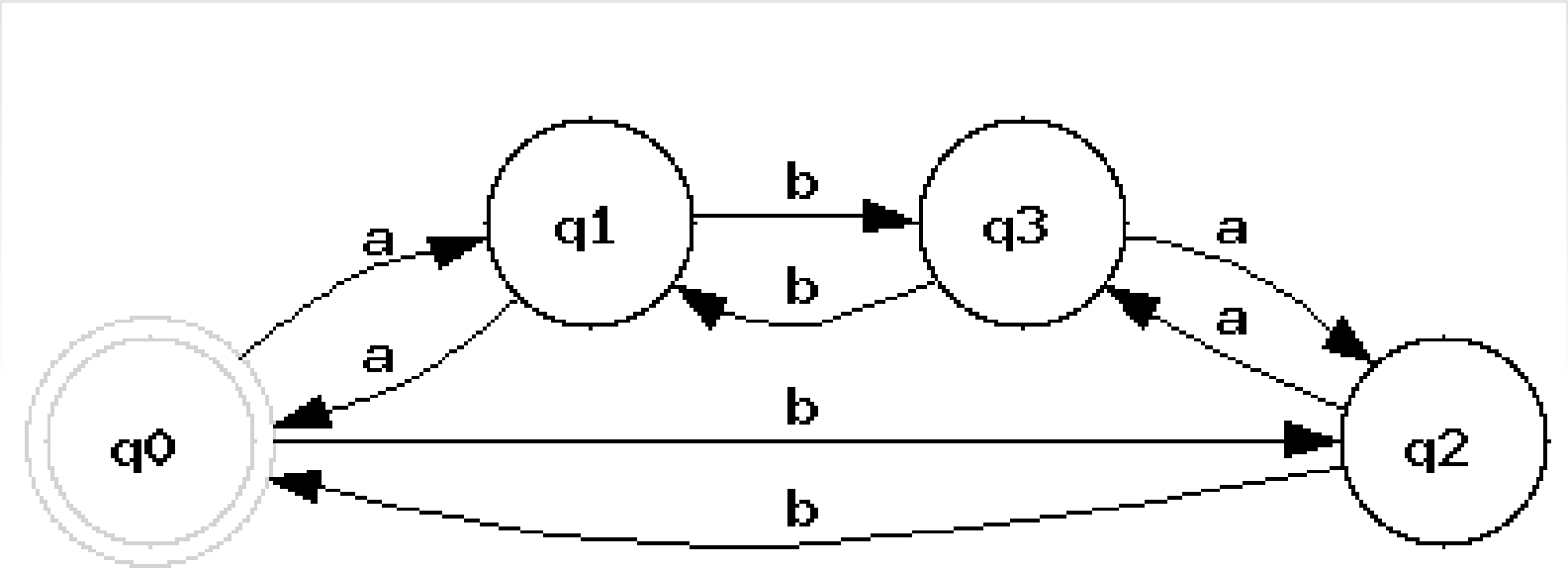
3. $F = \{(p,q) \in Q \mid p \in F_A \text{ y } q \in F_B\}$

PROPIEDADES DE CIERRE





PROPIEDADES DE CIERRE



ALGORITMOS DE DECISIÓN

- Son algoritmos que nos permiten responder cuestiones importantes sobre lenguajes regulares.

¿Es un lenguaje regular **vacío**?

¿Es un lenguaje regular **infinito**?

¿**Pertenece** una determinada **cadena** a un **lenguaje** dado?

Dadas dos descripciones de un lenguaje ¿describen el mismo lenguaje? **Equivalencia** de lenguajes.

ALGORITMOS DE DECISIÓN

¿Es un lenguaje regular **vacío**?

- Pensamos en que el lenguaje lo representamos mediante un autómata, por tanto:

*Un lenguaje es **vacío** si no existe un camino (cualquiera) desde el estado inicial a un estado final (cualquiera)*

ALGORITMOS DE DECISIÓN

¿Es un lenguaje regular **vacío**?

- Pensamos en que el lenguaje lo representamos mediante un autómata, por tanto:

¡MUCHO CUIDADO!

No quiere decir que el autómata no tenga estados o no tenga transiciones

ALGORITMOS DE DECISIÓN

¿Es un lenguaje regular **infinito**?

*Un lenguaje es **infinito** si se detecta*

algún ciclo en el autómata

ALGORITMOS DE DECISIÓN

¿**Pertenece** una determinada **cadena** a un **lenguaje** dado?

*Ver si al finalizar el análisis de la cadena nos encontramos en un **estado final.***

ALGORITMOS DE DECISIÓN

¿**Equivalencia** de Lenguajes?

$L_A = L_B$ *si se cumple que:*

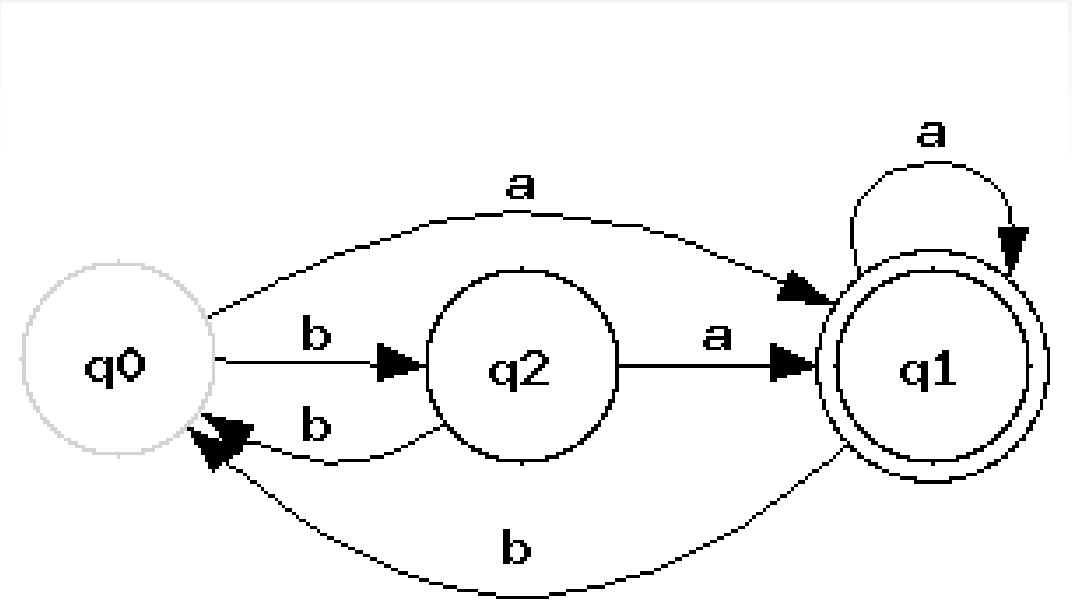
$$i)\ L_A \cap \overline{L_B} = \phi$$

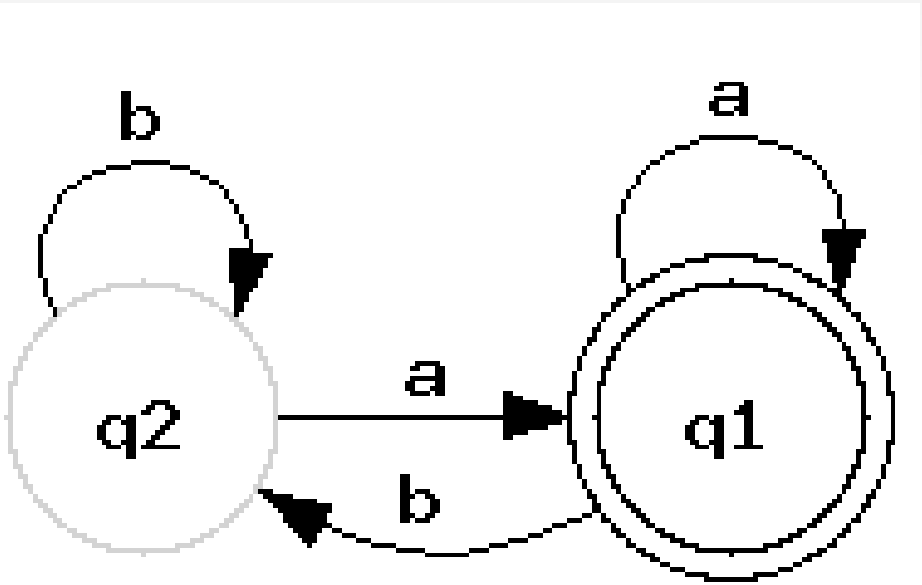
$$ii)\ \overline{L_A} \cap L_B = \phi$$

Es decir, si los dos autómatas resultantes de ambas operaciones reconocen el **lenguaje vacío.**

ALGORITMOS DE DECISIÓN

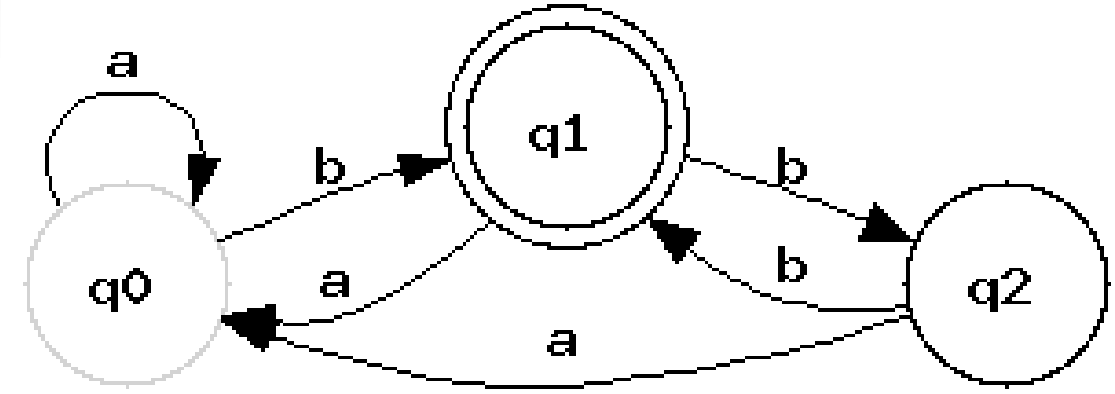
- Ejercicio: Determina si los siguientes lenguajes son equivalentes.

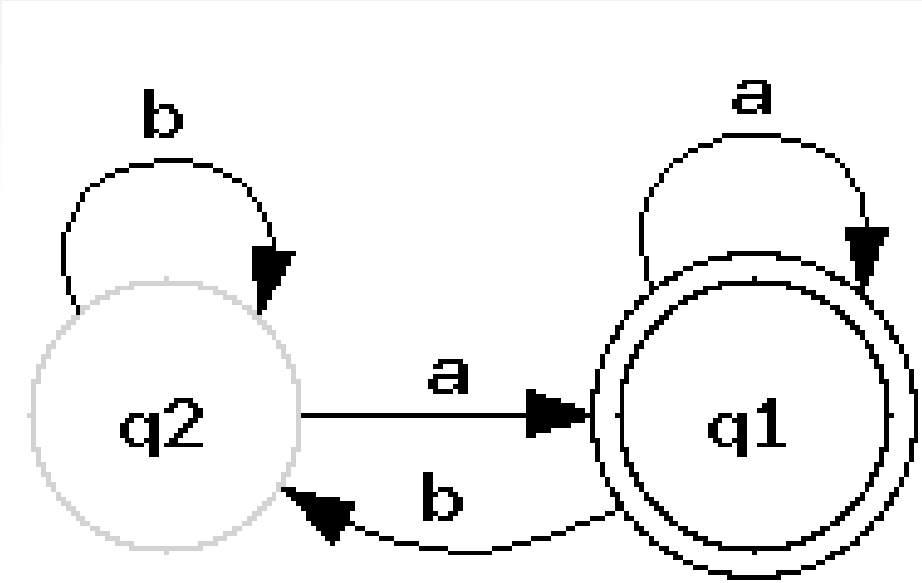




ALGORITMOS DE DECISIÓN

- Ejercicio: Determina si los siguientes lenguajes son equivalentes.





LEMA DE BOMBEO

Gramáticas	Lenguajes	Máquinas
Sin restricciones o de Tipo 0	Sin restricciones o de Tipo 0	Máquina de Turing
Sensible al contexto o de Tipo 1	Sensible al contexto o de Tipo 1	Autómata linealmente acotado
Libre de contexto o de Tipo 2	Libre de contexto o de Tipo 2	Autómata a pila
Regular o de Tipo 3	Regular o de Tipo 3	Autómata Finito

LEMA DE BOMBEO

(***Pumping Lemma***)

Teorema que nos permite **probar** que ciertos **lenguajes** no son **regulares**.

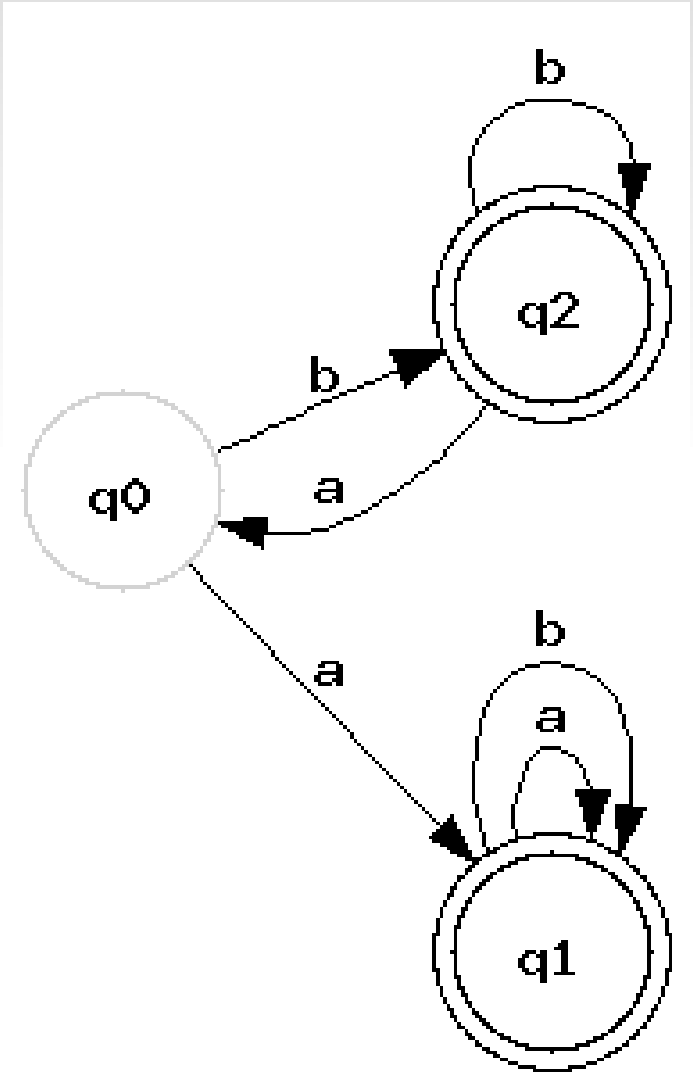
- Todo lenguaje finito es regular.
- No todo lenguaje infinito es regular.

LEMA DE BOMBEO

Suponiendo que cierto **lenguaje es regular**, entonces **forzosamente** dicho lenguaje debe **contener** palabras en que una **subcadena** se **repite** cualquier número de veces.

Es decir, hay **palabras** del lenguaje en que podemos **insertar repetidamente** (bombear) una **subcadena** (***v*** en el teorema) sin que el **autómata** se de cuenta.

LEMA DE BOMBEO



- Secuencias de caracteres distintas que llevan a un mismo estado se consideran indistinguibles. Ej:
  - bab
  - bbbbbb
- LIMITACIÓN AFs → sólo distingue aceptables de no aceptables.

LEMA DE BOMBEO

- Ejercicio: Construye un AFD que reconozca el lenguaje:

$$\{a^n b^n\}$$

LEMA DE BOMBEO

- Ejercicio: Construye un AFD que reconozca el lenguaje:

$$\{a^n b^n\}$$

No se puede porque lenguaje **no** es **regular**

- Esto lo demuestro mediante el lema de bombeo:

→ cadenas uvw donde v se repite:

Caso 1: si v sólo contienes as → más as que bs

Caso 2: si v sólo contiene bs → más bs que as

Caso 3: si v contiene as y bs → aa...bb...; si esto se repite entonces palabra **desordenada**.

TEMA 6

PROPIEDADES DE LOS LENGUAJES REGULARES

TEMA 7

GRAMÁTICAS

INTRODUCCIÓN

1. Definición de gramática formal.
2. Jerarquía de Chomsky.
3. Gramáticas Regulares .
4. Derivación directa y sucesiva.
5. Lenguaje generado por una gramática.
6. Árbol de derivación.
7. Ambigüedad de una gramática.

1. Gramática Formal

Una gramática es una *cuádrupla* $G = (\Sigma_N, \Sigma_T, P, S)$ donde:

- Σ_N : conjunto o alfabeto de símbolos no terminales (variables).

- Σ_T: conjunto o alfabeto de símbolos terminales (constantes).

  Debe cumplirse que $\Sigma_N \cap \Sigma_T = \varnothing$. Notamos $\mathbf{\Sigma} = \Sigma_N \cup \Sigma_T$ *a*lfabeto de la gramática.

- $S \in \Sigma_N$: símbolo inicial o axioma de la gramática.
- *P*: conjunto de producciones o reglas de reescritura.

- Observemos que $\alpha = \alpha_1 A \alpha_2$ debe contener al menos un símbolo no terminal, que puede estar *rodeado* de un *contexto*.

  - $P = \{\alpha \rightarrow \beta = \alpha_1 A \alpha_2 \rightarrow \beta \ / \ \alpha_1, \alpha_2 \in \Sigma^*, A \in \Sigma_N, \beta \in \Sigma^* \}$

DEFINICIÓN

$G = (\Sigma_N, \Sigma_T, P, S)$

P={NonZero $\rightarrow$ 1 | 2 | 3 |… | 9
Digito $\rightarrow$ 0 | 1 | 2 | 3 |… | 9
Cualquiera $\rightarrow \varepsilon$ | Digito Cualquiera
Nat $\rightarrow$ 0 | NonZero Cualquiera
Ex $\rightarrow$ Nat | **(**Ex**)** | Ex **+** Ex | Ex ***** Ex }

Σ_T= {0,1,2,… ,9,(,),+,*}

Σ_N = {Nat,NonZero,Cualquiera,Digito,Ex}

S = Ex

DEFINICIÓN

< *frase* > → < *sujeto* > < *predicado* >

< *sujeto* > → < *articulo* > < *sustantivo* >

< *articulo* > → *el* | *la*

< *sustantivo* > → *perro* | *luna*

< *predicado* > → < *verbo* >

< *verbo* > → *brilla* | *corre*

2. Jerarquía de Chosmky (1)

Sea $G = (\Sigma_N , \Sigma_T, P, S)$ una gramática formal. Clasificamos el lenguaje asociado a la gramática atendiendo a la estructura de esta de la siguiente forma:

Gramáticas de TIPO 0

Coincide con la definición de gramática formal, donde la única restricción es que en la parte izquierda de la producción debe haber al menos un No terminal.

El conjunto de produccionesse define:

$$P = \{\alpha_1 A \alpha_2 \rightarrow \beta \,/\, \alpha_1, \alpha_2 \in \Sigma^*, A \in \Sigma_N , \beta \in \Sigma^* \}$$

JERARQUÍA DE CHOMSKY

$A \rightarrow aABC \mid abC$

$cB \rightarrow BC$

$bB \rightarrow bb$

$bC \rightarrow b$

2. Jerarquía de Chosmky (2)

Gramáticas de TIPO 1 (Dependientes del contexto)

La parte izquierda y derecha de una producción debe tener una parte común y sólo puede anularse el símbolo inicial (S → λ)

$$S \rightarrow aSBC \mid aBC$$
$$aB \rightarrow ab$$
$$bB \rightarrow bb$$
$$bC \rightarrow bc$$
$$cC \rightarrow cc$$

2. Jerarquía de Chosmky (2)

Gramáticas de TIPO 2 (Libres del contexto)

La parte izquierda de una producción es un símbolo No terminal

$P = \{(S \rightarrow \lambda) \text{ o } A \rightarrow v \;/\; A \in \Sigma_N, v \in \Sigma^+\}$

$$S \rightarrow aSb \mid ab$$

2. Jerarquía de Chosmky (2)

Gramáticas de TIPO 3 (Regulares o lineales)

Lineales por la derecha:

$P = \{(S \rightarrow \lambda) \text{ o } A \rightarrow aB \text{ o } A \rightarrow a \;/\; A,B \in \Sigma_N \,,\, a \in \Sigma_T\}$

Lineales por la izquierda:

$P = \{(S \rightarrow \lambda) \text{ o } A \rightarrow Ba \text{ o } A \rightarrow a \;/\; A,B \in \Sigma_N \,,\, a \in \Sigma_T\}$

Lineal por la izquierda:

$A \rightarrow B1 \mid 1$

$B \rightarrow A0$

Lineal por la derecha:

$A \rightarrow 1B \mid 1$

$B \rightarrow 0A$

Se cumple que $\mathbf{G_3 \subset G_2 \subset G_1 \subset G_0}$

3. Gramáticas Regulares

- Una **Gramática Regular** (GR) es una gramática formal (GF) cuyas **producciones** son de la siguiente forma:

$$P = \{A \rightarrow \beta \,/\, A \in \Sigma_N ,$$

$$\beta = a \; o \; \beta = a\, B \; o \; \beta = \lambda ,$$

$$donde \; a \in \Sigma_T ,\; B \in \Sigma_N \}$$

3. Gramáticas Regulares

- Una **Gramática Regular** (GR) es una gramática formal (GF) cuyas **producciones** son de la siguiente forma:

$$P = \{A \rightarrow \beta \,/\, A \in \Sigma_N ,$$

$$\beta = a \text{ o } \beta = a\,B \text{ o } \boxed{\beta = \lambda},$$

$$donde\ a \in \Sigma_T,\ B \in \Sigma_N\}$$

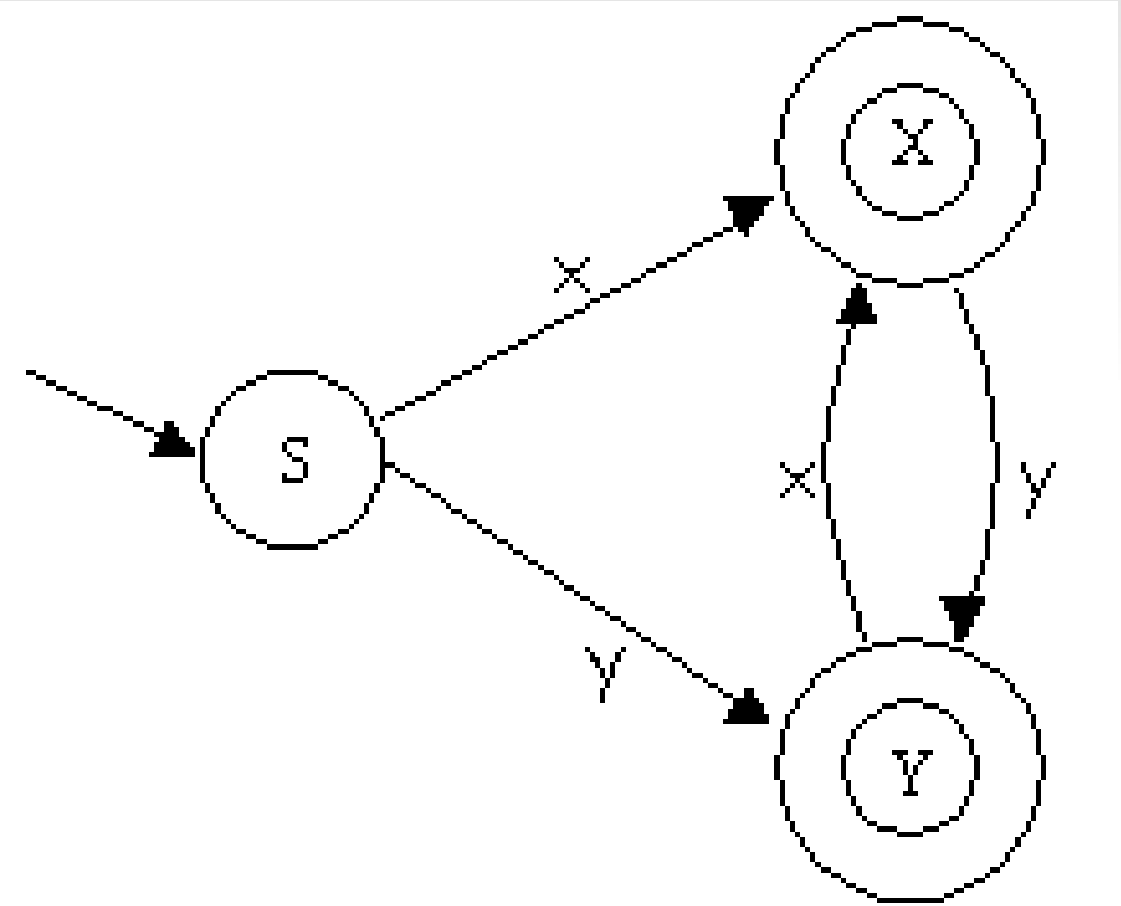
3. Gramáticas Regulares

AFD → GR

$$M = (Q, \Sigma, \delta, q_0, F) \rightarrow G=(\Sigma_N, \Sigma_T, P, S)$$

1. $\Sigma_N = Q$;
2. $\Sigma_T = \Sigma$;
3. $S = q_0$
4. $P = \{q \rightarrow ap / \delta(q,a)=p\} \cup \{q \rightarrow \lambda / q \in F\}$

3. Gramáticas Regulares



3. Gramáticas Regulares

- **Terminales**: x, y
- **No Terminales**: S, X, Y
- **Símbolo Inicial**: S
- **Producciones**:

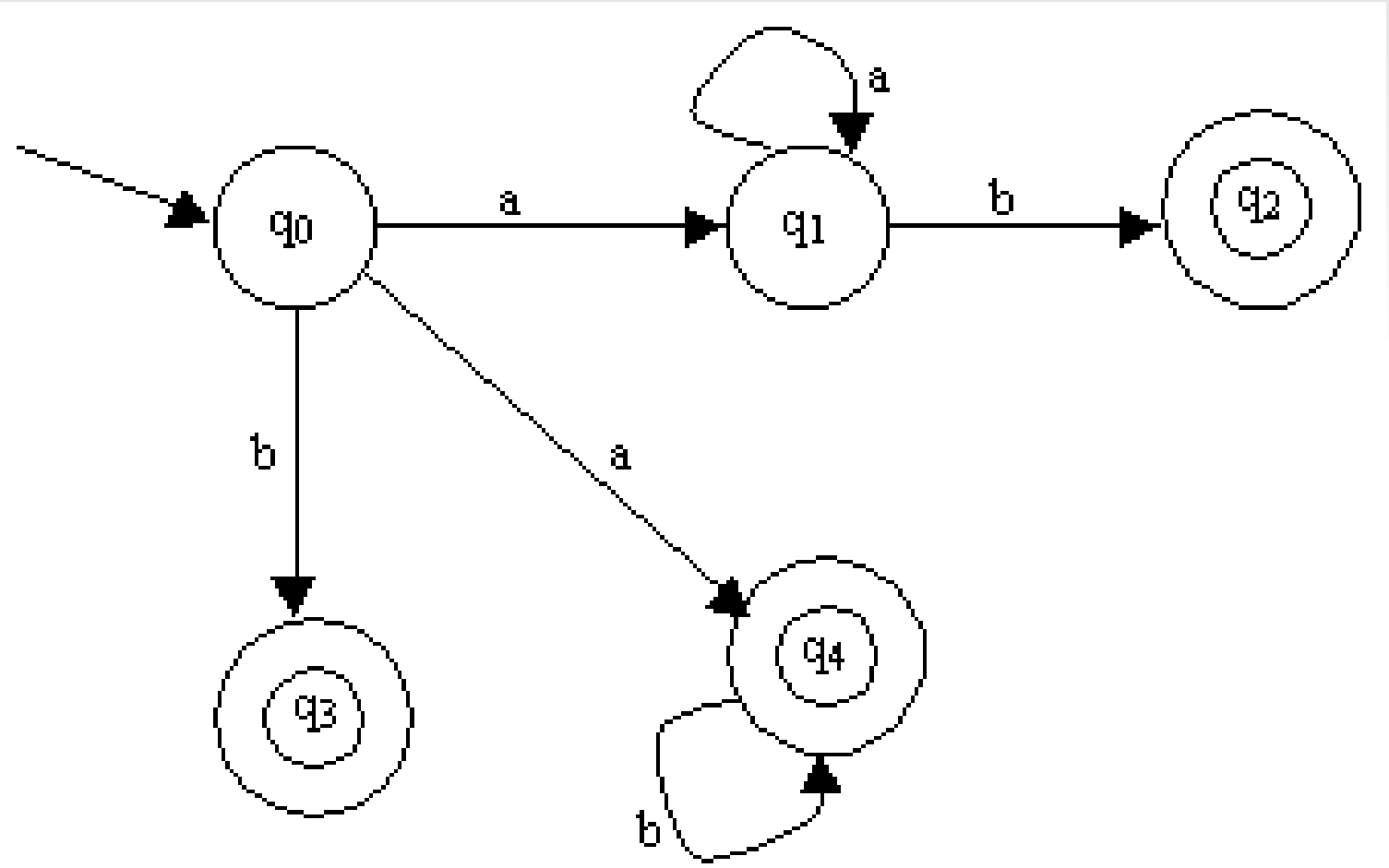
$$S \rightarrow xX \mid yY$$

$$X \rightarrow yY \mid \lambda$$

$$Y \rightarrow xX \mid \lambda$$

3. Gramáticas Regulares

EJERCICIO PROPUESTO AF → GR



3. Gramáticas Regulares

GR LINEAL POR LA DERECHA → AFND-λ

$$G=(\Sigma_N, \Sigma_T, P, S) \rightarrow M = (Q, \Sigma, \delta, q_0, F)$$

$Q= \Sigma_N \cup \{z\}$

$\Sigma = \Sigma_T$

$q_0 = S$

$F = \{z\}$

- Para cada producción $A \rightarrow aB$ se define la transición $\delta(A,a) = B$
- Para cada producción $A \rightarrow a$ se define la transición $\delta(A,a) = z$
- Para cada producción $A \rightarrow \lambda$ se define la transición $\delta(A,\lambda) = z$

3. Gramáticas Regulares

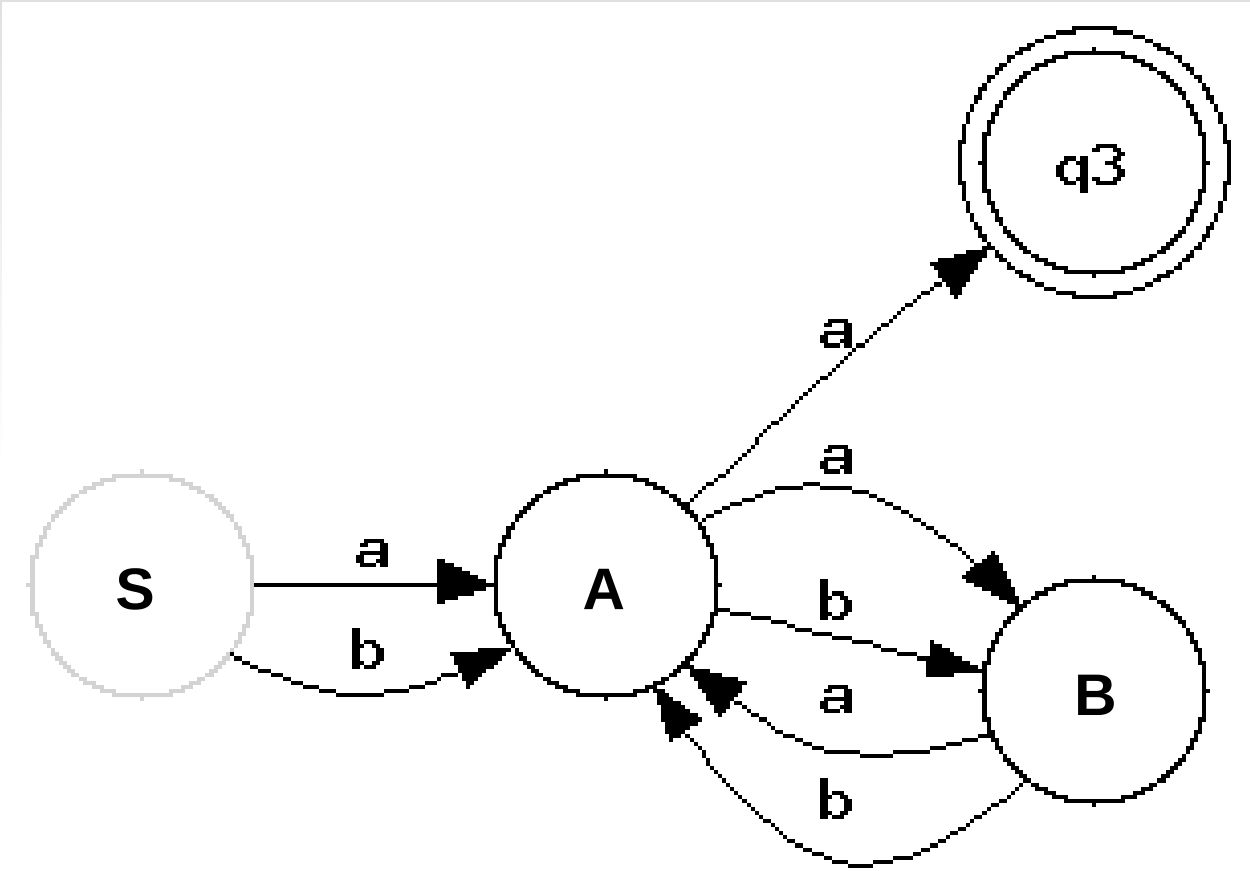
- **Terminales**: a, b,
- **No Terminales**: S, A, B,
- **Símbolo Inicial**: S
- **Producciones**:

$$S \rightarrow aA \mid bA$$

$$A \rightarrow aB \mid bB \mid a$$

$$B \rightarrow aA \mid bA$$

3. Gramáticas Regulares



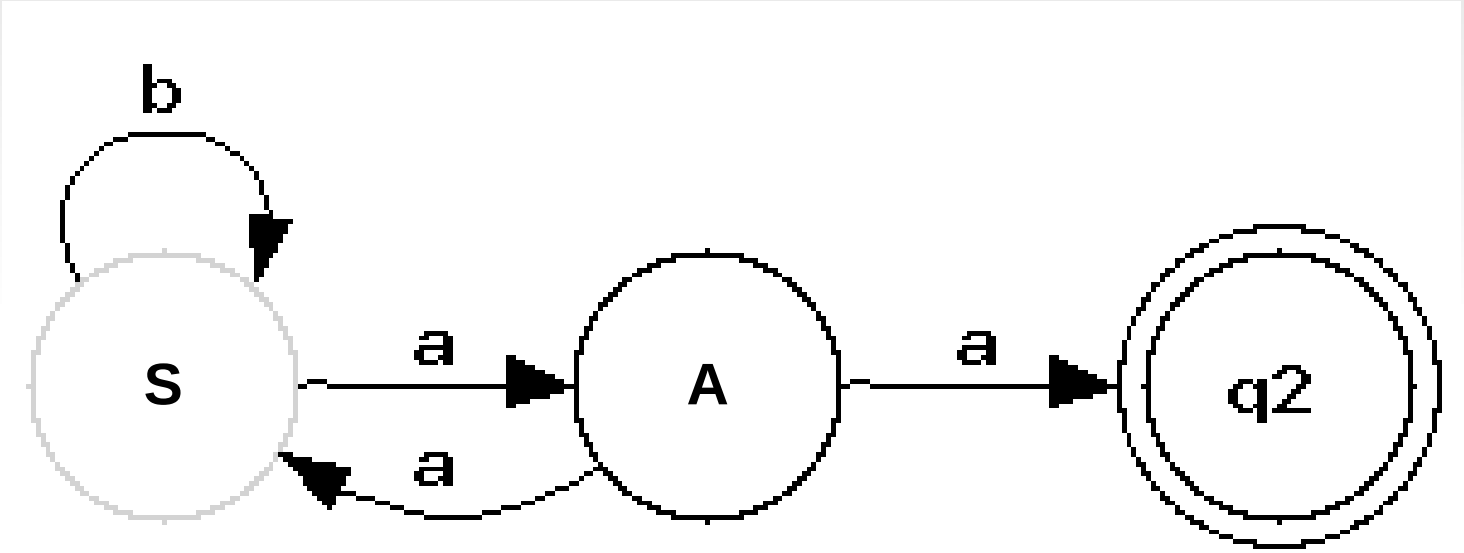
3. Gramáticas Regulares

- **Terminales**: a, b,
- **No Terminales**: S, A.
- **Símbolo Inicial**: S
- **Producciones**:

$$S \rightarrow b\,S \mid a\,A$$

$$A \rightarrow a\,S \mid a$$

3. Gramáticas Regulares



4. Derivación Directa y Sucesiva

- Sea $G = (\Sigma_N, \Sigma_T, P, S)$ una gramática, y $\alpha, \beta, \delta, \varphi, \rho, \ldots$ palabras de Σ^*. Entonces: β se deriva de α en un paso de derivación, o es una **derivación directa** de α, notado como $\alpha \Rightarrow_G \beta$, si existen dos cadenas $\varphi 1, \varphi 2 \in \Sigma^*$ y una producción $\delta \rightarrow \rho \in P$, tales que $\alpha = \varphi 1 \delta \varphi 2$ y $\beta = \varphi 1 \rho \varphi 2$.

$$\boldsymbol{\varphi 1 \delta \varphi 2 \Rightarrow_G \varphi 1 \rho \varphi 2}$$

4. Derivación Directa y Sucesiva

- Notamos como $\Rightarrow^*_G$ al cierre reflexivo y transitivo de $\Rightarrow_G$. Es decir, $\alpha \Rightarrow^*_G \beta$ (β es **derivable** o se deriva de α) si existe una sucesión de cadenas intermedias $\varphi 1, \varphi 2, \ldots, \varphi n$, tales que $\alpha = \varphi 1 \Rightarrow_G \varphi 2 \ldots \Rightarrow_G \varphi n = \beta$. Si $n = 0$ entonces $\alpha = \beta$.
- Una derivación $\alpha \Rightarrow^* G \beta$ se llama más a la **derecha**, si en cada paso de derivación directa se expande el símbolo no terminal más a la derecha. (análogo izquierda)
- $x \in \Sigma^*$ es una **forma sentencia**l de G si $S \Rightarrow^*_G x$ (se deriva del axioma S). Si además $x \in \Sigma_T^*$, entonces se dice que x es una **sentencia**.

4. Derivación Directa y Sucesiva

- (1) S → ASB
- (2) A →b
- (3) aaA → aaBB
- (4) S →d
- (5) A → aA
- (6) B → dcd

S → abddcd (Deriv. Izquierda)

4. Derivación Directa y Sucesiva

- (1) S → ASB
- (2) A → b
- (3) aaA → aaBB
- (4) S → d
- (5) A → aA
- (6) B → dcd

$$S \rightarrow abddcd$$

$$S \xrightarrow{(1)} ASB \xrightarrow{(5)} aASB \xrightarrow{(2)} abSB \xrightarrow{(4)} abdB \xrightarrow{(6)} abddcd$$

5. Lenguaje generado por una Gramática

El lenguaje generado por una gramática (L(G)) es el conjunto de todas las sentencias obtenidas a partir de la gramática.

$$L(G) = \{ x \in \Sigma_T^* \,/\, S \Rightarrow_G^* x \}$$

Dos gramáticas G_1 y G_2 son ***equivalentes*** cuando los lenguajes que generan son iguales: $L(G_1) = L(G_2)$

Ejemplo: Enteros. G_1=({0,1,2,3,4,5,6,7,8,9},{E,D},E,P)

E → DE | D
D → 0|1|2|3|4|5|6|7|8|9

E → 0D |1D |2D|3D|4D|5D|6D|7D |8D|9D
0|1|2|3|4|5|6|7|8|9

Ejemplo: L=Palíndromos con $|w| \geq 1$

A → 1B1 | 0B0 |0 | 1
B → A | λ

A → 1A1 | 0A0 | 00 | 11| 0 | 1

Ejemplo: L=Palíndromos con $|w| \geq 0$

A → 1A1 | 0A0 | 0 | 1 | λ

6. Árbol de Derivación

- Dada una gramática G=(N,T,P,S), un árbol de derivación o análisis para una palabra $x \in \Sigma_T^*$ es un árbol donde:
  - Los nodos son símbolos de $\Sigma \cup \{\lambda\}$.
  - La raíz es S.
  - Los nodos interiores son símbolos no terminales (de Σ_N).
  - Las hojas son símbolos terminales Σ_T y $\{\lambda\}$.
  - Si los hijos de un nodo con etiqueta A tienen etiquetas x1,x2, ..., xn, entonces existe en P una producción $A \rightarrow x1x2...xn$.

6. Árbol de Derivación

- **Terminales**: 0, 1, 2
- **No Terminales**: A, B,
- **Símbolo Inicial**: A
- **Producciones**:

$$A \rightarrow 0B \mid 2$$

$$B \rightarrow 0A \mid 1$$

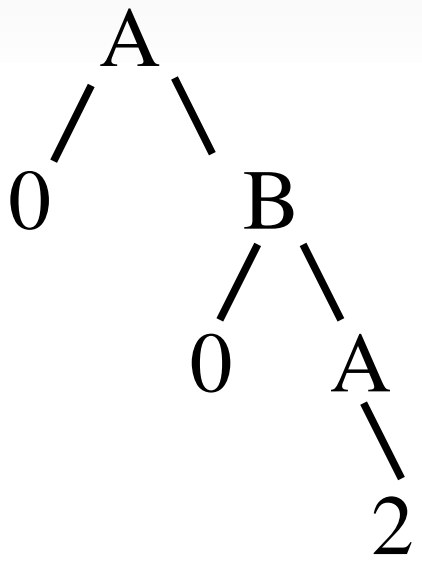
(i) Obtener derivaciones para las palabras 002 y 0001.

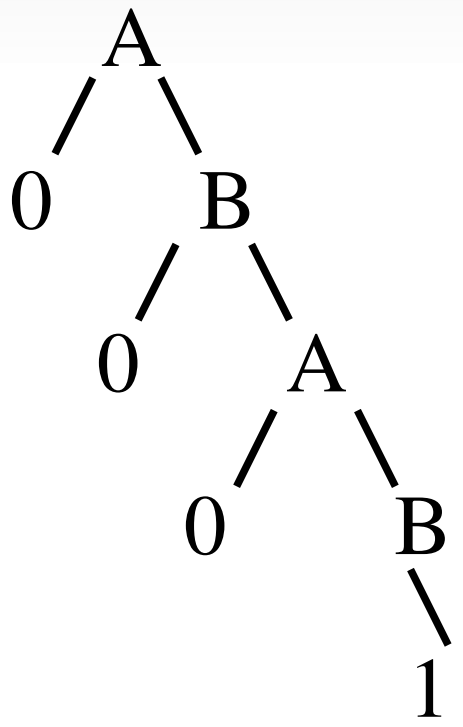
(ii) Describir el árbol de derivación para cada una de ellas.

(iii) Obtener el lenguaje que genera

6. Árbol de Derivación

- $A \rightarrow 0B \rightarrow 00A \rightarrow 002$
- $A \rightarrow 0B \rightarrow 00A \rightarrow 000B \rightarrow 0001$





6. Árbol de Derivación

(iii) La obtención del lenguaje se hace analizando las palabras obtenidas desde el axioma.

$A \rightarrow 2$

$A \rightarrow 0B \rightarrow 01$

$A \rightarrow 0B \rightarrow 00A \rightarrow 002$

$A \rightarrow 0B \rightarrow 00A \rightarrow 000B \rightarrow 0001$

$A \rightarrow 0B \rightarrow 00A \rightarrow 000B \rightarrow 0000A \rightarrow 00002$

$\mathbf{L_1=\{0^n2 \mid n\ par\}}$ **y** $\mathbf{L_2=\{0^m1 \mid m\ impar\}} \rightarrow \mathbf{L_G = L_1\ U\ L_2}$

6. Árbol de Derivación

$E \rightarrow E + E \mid E * E \mid x \mid y$

Obtener el árbol de derivación para la sentencia

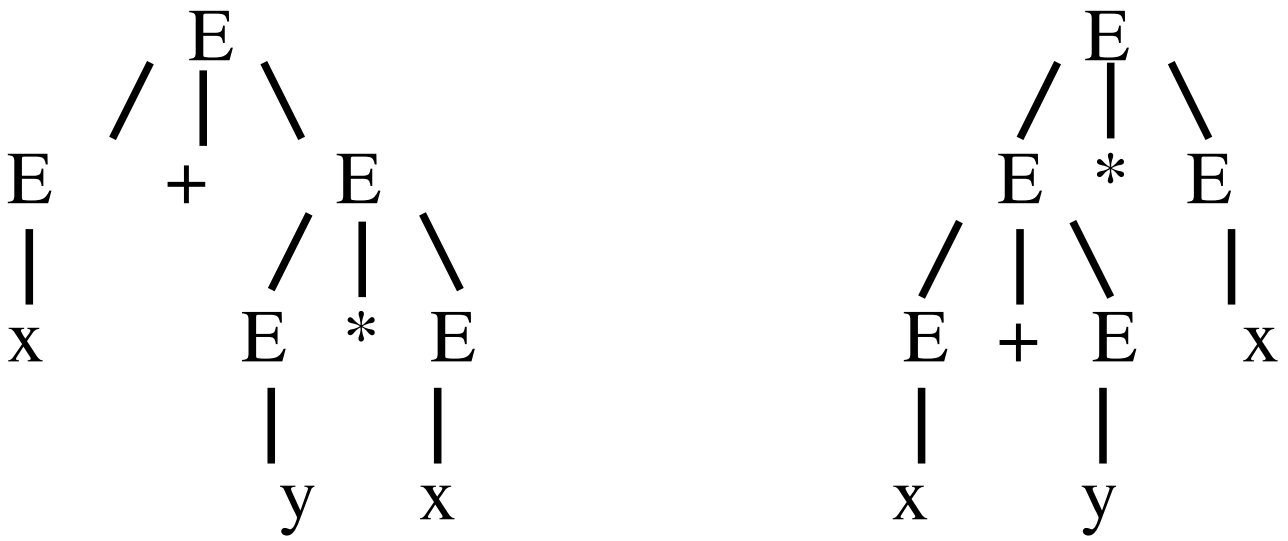
”x + y * x”

6. Árbol de Derivación

$E \rightarrow E + E \mid E * E \mid x \mid y$

Obtener el árbol de derivación para la sentencia

”x + y * x”



7. Ambigüedad de una Gramática

Una GLC es *ambigua* si existe una cadena $w \in$ L(G) para la cual existen dos o más árboles de derivación. En caso de que toda cadena $w \in$ L(G) tenga un único árbol de derivación, la gramática es no ambigua.

Ejemplo: la gramática $S \rightarrow aS \mid Sa \mid a$ es ambigua porque *aa* tiene dos derivaciones por la izquierda

$S \Rightarrow aS \Rightarrow aa$ $\qquad$ $S \Rightarrow Sa \Rightarrow aa$

Esta gramática genera el lenguaje a^+ que también es el lenguaje generado por la gramática no ambigua $S \rightarrow aS \mid a$.

AMBIGÜEDAD

- Una **sentencia** es ambigüa si tiene más de una derivación (o árbol de derivación).
- Una **gramática** es ambigüa si tiene al menos una sentencia ambigüa.
- Un **lenguaje** es ambigüo si es generado por una gramática ambigüa.

EJERCICIOS

Define una GLC para los siguientes lenguajes:

a) $L = \{ww^{-1} \mid w \in \{a, b\}^* \}$

EJERCICIOS

Define una GLC para los siguientes lenguajes:

a) $L = \{ww^{-1} \mid w \in \{a, b\}^* \}$

$S \rightarrow aSa \mid bSb \mid \lambda$

EJERCICIOS

Define una GLC para los siguientes lenguajes:

b) $L = \{wxw^{-1} \mid w \in \{a, b\}^* \text{ y } x \in \{a, b\}\}$

EJERCICIOS

Define una GLC para los siguientes lenguajes:

b) $L = \{wxw^{-1} \mid w \in \{a, b\}^* \text{ y } x \in \{a, b\}\}$

$S \rightarrow aSa \mid bSb \mid a \mid b$

EJERCICIOS

Define una GLC para los siguientes lenguajes:

e) $L = \{(a + b)^n \mid n \geq 1\}$

EJERCICIOS

Define una GLC para los siguientes lenguajes:

e) $L = \{(a + b)^n \mid n \geq 1\}$

$S \rightarrow aS \mid bS \mid a \mid b$

EJERCICIOS

Define una GLC para los siguientes lenguajes:

f) $L = \{a^i b^j \mid i, j \geq 1\}$

EJERCICIOS

Define una GLC para los siguientes lenguajes:

f) $L = \{a^i b^j \mid i, j \geq 1\}$

$S \rightarrow AB$
$A \rightarrow aA \mid a$
$B \rightarrow bB \mid b$

EJERCICIOS

Define una GLC para los siguientes lenguajes:

f) $L = \{a^i b^j \mid i, j \geq 1\}$

¿Es Regular?

¿Se puede generar con una GR?

EJERCICIOS

Define una GLC para los siguientes lenguajes:

f) $L = \{a^i b^j \mid i, j \geq 1\}$

$S \rightarrow aS \mid aB$

$B \rightarrow bB \mid b$

EJERCICIOS

Define una GLC para los siguientes lenguajes:

h) $L = \{a^n b^{2n} \mid n \geq 1\}$

EJERCICIOS

Define una GLC para los siguientes lenguajes:

h) $L = \{a^n b^{2n} \mid n \geq 1\}$

$S \rightarrow aSbb \mid abb$

EJERCICIOS

Define una GLC para los siguientes lenguajes:

j) $L = \{(a(b + c))^n a \mid n \geq 0\}$

EJERCICIOS

Define una GLC para los siguientes lenguajes:

j) $L = \{(a(b + c))^n a \mid n \geq 0\}$

$S \rightarrow abS \mid acS \mid a$

EJERCICIOS

Define una GLC para los siguientes lenguajes:

j) $L = \{(a(b + c))^n a \mid n \geq 0\}$

¿Se podría generar con una GR?

EJERCICIOS

Define una GLC para los siguientes lenguajes:

j) $L = \{(a(b + c))^n a \mid n \geq 0\}$

$S \rightarrow aB \mid a$

$B \rightarrow bS \mid cS$

EJERCICIOS

Define una GLC para los siguientes lenguajes:

l) $L = \{a^n b^n c^k \mid n \geq 1, k \geq 1\}$

EJERCICIOS

Define una GLC para los siguientes lenguajes:

l) $L = \{a^n b^n c^k \mid n \geq 1, k \geq 1\}$

$S \rightarrow Sc \mid Ac$

$A \rightarrow aAb \mid ab$

EJERCICIOS

Define una GLC para los siguientes lenguajes:

ñ) $L = \{a^i b^j c^{i+j} \mid i,j \geq 0\}$

EJERCICIOS

Define una GLC para los siguientes lenguajes:

ñ) $L = \{a^i b^j c^{i+j} \mid i,j \geq 0\}$

$S \rightarrow aSc \mid B$

$B \rightarrow bBc \mid \lambda$

TEMA 8

SIMPLIFICACIÓN DE GRAMÁTICAS LIBRES DE CONTEXTO

Recursividad

- Una producción es recursiva si el mismo símbolo no terminal aparece en los dos lados de la producción:

$$\mathbf{A \rightarrow xAy}$$

$A \rightarrow 0A0$

$B \rightarrow B10$

$C \rightarrow 111C$

Recursividad

- Una **gramática** es **recursiva** si tiene al menos una producción recursiva.
- Recursividad por la **Izquierda**: $A \rightarrow Ay$ $(y \in \sum^*)$
- Recursividad por la **Derecha**: $A \rightarrow xA$ $(x \in \sum^*)$
- Recursividad por la **izquierda** de **más** de **un paso**:

$$A \rightarrow Bx;\ B \rightarrow Ay \text{ es como si: } A \rightarrow^* Ayx$$

GRM. LIBRES DEL CONTEXTO

Para construir un lenguaje de programación las **gramáticas regulares** tienen **limitaciones**:

Paréntesis bien balanceados

((())) ↔ $a^n b^n$; (()())

Sin embargo, con GLC si podemos generar ese lenguaje:

$S \rightarrow SS$

$S \rightarrow (S)$

$S \rightarrow \lambda$

GRM. LIBRES DEL CONTEXTO

Los **autómatas** que **reconocen** los lenguajes generados por las GLC son los **Autómatas a Pila**

*”Las GLC y los Autómatas a Pila son importantes en el mundo de la computación ya que permiten la **descripción** de la mayoría de **lenguajes de programación**.”*

GRM. LIBRES DEL CONTEXTO

Múltiples Gramáticas pueden generar el **mismo lenguaje**, estudiaremos:

*"**Preparar** gramática a efectos de ser tratada **eficientemente** por el autómata que reconozca el lenguaje generado por la gramática"*

LIMPIAR LA GRAMÁTICA

GRAMÁTICA LIMPIA

Una GLC está **limpia** si y sólo si **NO** tiene:

1. Producciones y símbolos **inútiles**: no generativos (Alg. 1)

y no accesibles. (Alg. 2).

2. Producciones **nulas**. (Alg. 3 y Alg. 4)

3. Producciones **unitarias**. (Alg. 5)

¡CUIDADO! $L_G = L_{G'}$!

Eliminación de símbolos no generativos

Algorítmo 1: Eliminación de No terminales que no derivan en cadenas de terminales. Eliminación también de las producciones que los contienen.

ENTRADA: $G=(\Sigma_N, \Sigma_T, P, S)$ una GLC.

SALIDA: $G'=(\Sigma_N', \Sigma_T, P', S)$ tal que $L(G)=L(G')$ y se cumpla que: $\forall \quad A \in \Sigma_N'$, se tenga que $A \Rightarrow^* w$, para algún $w \in \Sigma_T^*$.

MÉTODO:

- Inicializar Σ_N' con todos los no terminales A para los que $A \rightarrow w$ sea una producción de G, con $w \in \Sigma_T^*$.
- Inicializar P' con todas las producciones $A \rightarrow w$ para las cuales $A \in \Sigma_N'$, y $w \in \Sigma_T^*$.
- Repetir

  Añadir a Σ_N' todos los no terminales A para los cuales $A \rightarrow w$, para algún $w \in (\Sigma_N' \cup \Sigma_T)^*$ que sea una producción de P y añadirla a P'.

  Hasta que no se puedan añadir mas no terminales a Σ_N'

Símbolos No generativos

S →AB
S →Ab
A → aC
B → bCa
B → DbE
C → b
D → Fb
E → ca
F → aD

Σ_N'={}

P' ={}

Símbolos No generativos

S →AB
S →Ab
A → aC
B → bCa
B → DbE
C → b
D → Fb
E → ca
F → aD

Σ_N'={C, E}

P' ={producciones en rojo}

Símbolos No generativos

S →AB
S →Ab
A → aC
B → bCa
B → DbE
C → b
D → Fb
E → ca
F → aD

Σ_N'={C, E, A, B}

P' ={producciones en rojo}

Símbolos No generativos

S →AB
S →Ab
A → aC
B → bCa
B → DbE
C → b
D → Fb
E → ca
F → aD

Σ_N'={C, E, A, B, S}

P' ={producciones en rojo}

Símbolos No generativos

S →AB
S →Ab
A → aC
B → bCa
~~B → DbE~~
C → b
~~D → Fb~~
E → ca
~~F → aD~~

Σ_N'={C, E, A, B, S}

P' ={producciones en rojo}

Símbolos No generativos

Elimina los símbolos no generativos de la siguiente gramática.

- Terminales: a, b, c,
- No Terminales: S, A, B, D, E,
- Símbolo Inicial: S
- Producciones:

S ---> A B
A ---> a D | a
B ---> b D | b
D ---> E
E ---> D

Símbolos No generativos

```
grammar simplif{

    terminal a,b,c;
    nonterminal S,A,B,D,E;
    axiom S;
    productions{
        S:=A B;
        A:= a D | a;
        B:= b D | b;
        D := E;
        E := D;
    }
}

print(simplif);
gramsimplificad = remnongen(simplif);
```

Símbolos No generativos

Elimina los símbolos no generativos de la siguiente gramática.

- Terminales: a, b,
- No Terminales: A, B, S,
- Símbolo Inicial: S
- Producciones:

S ---> A B
A ---> a
B ---> b

Eliminación de símbolos no accesibles

Algoritmo 2: Eliminación de símbolos no accesibles desde el símbolo inicial de la gramática. Eliminación también de las producciones que los contienen).

ENTRADA: $G=(\Sigma_N, \Sigma_T, P, S)$ una GLC.

SALIDA: $G'=(\Sigma_N', \Sigma_T', P', S)$ tal que $L(G)=L(G')$ y se cumpla que:

$\forall A \in \Sigma_N'$, y $\forall a \in \Sigma_T'$, se tenga que $S \Rightarrow^* \alpha_1 A \alpha_2$.y $S \Rightarrow^* \beta_1 a \beta_2$,

MÉTODO:

- Inicializar: $\Sigma_N'=\{ S \}$, $P' = \varnothing$ y $\Sigma_T' = \varnothing$.
- Repetir

  $\forall A \in \Sigma_N'$, Si $A \rightarrow w$ es una producción de P entonces:

  - Introducir $A \rightarrow w$ en P'.
  - Para todo no terminal B definido en w, introducir B en Σ_N'
  - Para todo terminal a definido en w, introducir a en Σ_T'

  Hasta que no se puedan añadir mas no terminales a Σ_N'

Símbolos No Accesibles

S →AB
S →Ab
A → aC
B → bCa
C → b
E→ ca

Σ_N'={S}

Σ_T'={}

P' ={}

Símbolos No Accesibles

S →AB
S →Ab
A → aC
B → bCa
C → b
E→ ca

Σ_N'={S, A, B}

Σ_T'={b}

P' ={producciones en rojo}

Símbolos No Accesibles

S →AB
S →Ab
A → aC
B → bCa
C → b
E→ ca

Σ_N'={S, A, B, C}

Σ_T'={b, a}

P' ={producciones en rojo}

Símbolos No Accesibles

S →AB
S →Ab
A → aC
B → bCa
C → b
E→ ca

Σ_N'={S, A, B, C}

Σ_T'={b, a}

P' ={producciones en rojo}

Símbolos No Accesibles

S →AB
S →Ab
A → aC
B → bCa
C → b
E→ ca

Σ_N'={S, A, B, C}

Σ_T'={b, a}

P' ={producciones en rojo}

Símbolos No Accesibles

S →AB
S →Ab
A → aC
B → bCa
C → b
~~E → ca~~

Σ_N'={S, A, B, C}

Σ_T'={b, a}

P' ={producciones en rojo}

Se suprimen los símbolos E y c porque no son accesibles desde S.

Símbolos No Accesibles

Elimina los símbolos no accesibles de la siguiente gramática.

$S \rightarrow Aba$
$A \rightarrow BDb \mid EB$
$B \rightarrow CD$
$C \rightarrow ab \mid aA$
$D \rightarrow b$
$E \rightarrow Sa$

Símbolos No Accesibles

Elimina los símbolos no accesibles de la siguiente gramática.

S → Aba
A → BDb | EB
B → CD
C → ab | aA
D → b
E → Sa

Σ_N'={S, A, B, D, E, C}

Σ_T'={b,a}

P' ={producciones en rojo}

No se suprime ningún símbolo

Correcta eliminacion de símbolos inútiles

1. Eliminar los símbolos no generadores.
2. Eliminar los símbolos no accesibles.

Si estos algoritmos se aplican en orden inverso entonces **no se garantiza** que se eliminen todos los símbolos inútiles.

Producciones anulables

La presencia de reglas que producen el **lenguaje vacío** puede ser fuente de dificultades:

1. **Ambigüedad**.
2. Se pueden llegar a obtener **derivaciones** arbitrariamente **largas**. Por ejemplo, para G:

$S \rightarrow SS \mid (S) \mid \lambda$

La obtención de () podría ser:

$S \rightarrow SS \rightarrow SSS \rightarrow \ldots \rightarrow SSS \rightarrow SS \rightarrow S \rightarrow (S) \rightarrow ()$

Obtención de símbolos anulables

Algoritmo 3: Obtención de símbolos anulables (símbolos No terminales que derivan en λ).

ENTRADA: $G=(\Sigma_N, \Sigma_T, P, S)$ una GLC.

SALIDA: $M \subseteq \Sigma_N$ tal que $\forall A \in M$ se cumple que $A \Rightarrow^* \lambda$

MÉTODO:

1. Inicializar M con los No terminales para los cuales existe una producción λ ($A \rightarrow \lambda$) .
2. Repetir
   Para cada $A \in (\Sigma_N - M)$ tal que $A \rightarrow w$, con $w \in M^+$,
   Introducir $A \rightarrow w$ en M.
   Hasta que no se añadan mas no terminales a M.

Símbolos anulables

$S \rightarrow AD \mid B$

$A \rightarrow CDE$

$B \rightarrow CE$

$C \rightarrow S \mid a \mid \lambda$

$D \rightarrow A \mid b$

$E \rightarrow S \mid a \mid \lambda$

$M=\{C, E\}$

Símbolos anulables

S → AD| B

A → CDE

B→CE

C→S | a | λ

D →A | b

E → S | a | λ

M={C, E, B}

Símbolos anulables

S → AD| B

A → CDE

B→CE

C→S | a | λ

D →A | b

E → S | a | λ

M={C, E, B, S}

Símbolos anulables

Obtener los símbolos anulables de la siguiente gramática

S ---> A B b
S ---> A B C
C ---> a b c
C ---> A B
A ---> a A
A ---> λ
B ---> b B
B ---> λ

Símbolos anulables

Obtener los símbolos anulables de la siguiente gramática

S ---> A B b
S ---> A B C
C ---> a b c
C ---> A B
A ---> a A
A ---> λ
B ---> b B
B ---> λ

M= {A, B, C, S}

Eliminación de producciones anulables

Algoritmo 4: Eliminación de producciones anulables.

ENTRADA: $G=(\Sigma_N, \Sigma_T, P, S)$ una GLC.

SALIDA: $G' = (\Sigma_N', \Sigma_T, P', S)$ tal que $L(G) = L(G')$ y se cumpla que: $\forall A \in \Sigma_N'$, A no deriva en λ. Si $\lambda \in L(G)$ entonces $S \rightarrow \lambda$.

MÉTODO:

- Obtener el conjunto de anulables M (algorítmo 3).
- Inicializar $P' = \varnothing$
- Para cada producción $B \rightarrow x_1 x_2 \ldots x_n \in P$ hacer

  Incluir en P' un conjunto de producciones con la estructura $B \rightarrow y_{1y} y_2 \ldots y_n$ donde:
  - si x_i no es anulable ($x_i \notin M$) entonces $y_i = x_i$
  - si x_i es anulable ($x_i \in M$) entonces se crea una producción donde $y_i = x_i$ y otra donde y_i no aparece
  - Si todos los x_i son anulables, NO añadir $B \rightarrow \lambda$
- Si $\lambda \in L(G)$ ($S \in M$) entonces añadir $S \rightarrow \lambda$ a P'

Eliminar Prod. anulables

S → AD
S → B
A → CDE
B→CE
C→S
C → a
C → λ
D →A
D → b
E → S
E → a
E → λ

M={C, E, B, S}

Σ_N'={}

P' ={producciones en rojo}

Eliminar Prod. anulables

S → AD
S → B entonces S →B y S →λ
A → CDE
B→CE
C→S
C → a
C → λ
D →A
D → b
E → S
E → a
E → λ

M={C, E, B, S}

Σ_N'={}

P' ={producciones en rojo}

Eliminar Prod. anulables

S → AD
S → B entonces S →B y ~~S → λ~~
A → CDE entonces A → CDE | CD | DE | D
B→CE
C→S
C → a
C → λ
D →A
D → b
E → S
E → a
E → λ

M={C, E, B, S}

Σ_N'={}

P' ={producciones en rojo}

Eliminar Prod. anulables

S → AD
S → B entonces S →B y ~~S → λ~~
A → CDE entonces A → CDE | CD | DE | D
B→CE entonces B → CE | C | E
C→S entonces C → S
C → a
~~C → λ~~
D →A
D → b
E → S entonces E → S
E → a
~~E → λ~~

M={C, E, B, S}

Σ_N'={}

P' ={producciones en rojo}

Eliminar Prod. anulables

Añadimos $S \rightarrow \lambda$

$S \rightarrow AD$

$S \rightarrow B$

$A \rightarrow CDE \mid CD \mid DE \mid D$

$B \rightarrow CE \mid C \mid E$

$C \rightarrow S$

$C \rightarrow a$

$D \rightarrow A$

$D \rightarrow b$

$E \rightarrow S$

$E \rightarrow a$

$M=\{C, E, B, S\}$

$\Sigma_N' = \{\Sigma_N\}$

P' ={producciones en rojo}

Eliminar Prod. anulables

Eliminar las producciones anulables de la siguiente gramática

S ---> A B b
S ---> A B C
C ---> a b c
C ---> A B
A ---> a A
A ---> λ
B ---> b B
B ---> λ

M= {A, B, C, S}

Eliminar Prod. anulables

Eliminar las producciones anulables de la siguiente gramática

M= {A, B, C, S}

S ---> A B b entonces S → ABb | Bb | Ab | b

S ---> A B C ent. S→ABC | BC | AC | AB | A | B | C

C ---> a b c

C ---> A B entonces C → AB | A | B

A ---> a A entonces A → aA | a

~~A ---> λ~~

B ---> b B entonces B → bB | b

~~B ---> λ~~

Añadir S → λ

Eliminación de producciones unitarias

Algoritmo 5: Eliminación de producciones unitarias.

ENTRADA: $G=(\Sigma_N, \Sigma_T, P, S)$ una GLC sin producciones λ.

SALIDA: $G' = (\Sigma_N', \Sigma_T', P', S')$ tal que $L(G) = L(G')$ y se cumpla que:

$\forall A \in \Sigma_N'$, no existen producciones con la forma $A \rightarrow B$.

MÉTODO:

- Para cada $A \in \Sigma_N$, obtener el conjunto Unitarios (A)

  $\text{Unitarios}(A)= \{B \in \Sigma_N / A \Rightarrow^+ B\}$

- Inicializar P' = P- {Producciones Unitarias}
- Para cada $A \in \Sigma_N$ para el cual $\text{Unitarios}(A) \neq \emptyset$

  Para cada $B \in$ Unitarios(A)

  Para cada producción no unitarios $B \rightarrow w \in P'$

  Añadir $A \rightarrow w$ a P'

Eliminar Prod. unitarias

Terminales: a, b,
No Terminales: A, B, C,
Simbolo Inicial: A
Producciones (P):
A ---> A B b
A ---> B
B ---> C
B ---> a b
C ---> a a
C ---> b b

Eliminar Prod. unitarias

Terminales: a, b,
No Terminales: A, B, C,
Simbolo Inicial: A
Producciones (P):
A ---> A B b
A ---> B
B ---> C
B ---> a b
C ---> a a
C ---> b b

UNITARIOS(A)={B, C}
UNITARIOS(B)={C}
UNITARIOS(C)={ϕ}

Eliminar Prod. unitarias

Terminales: a, b,
No Terminales: A, B, C,
Simbolo Inicial: A
Producciones (P'):
A ---> A B b
~~A ---> B~~
~~B ---> C~~
B ---> a b
C ---> a a
C ---> b b

UNITARIOS(A)={B, C}
UNITARIOS(B)={C}
UNITARIOS(C)={ϕ}

Eliminar Prod. unitarias

Terminales: a, b,
No Terminales: A, B, C,
Simbolo Inicial: A
Producciones (P'):
A --> A B b $A \rightarrow a\ b$
~~A ---> B~~
~~B ---> C~~
B ---> a b
C ---> a a
C ---> b b

UNITARIOS(A)={B, C}
UNITARIOS(B)={C}
UNITARIOS(C)={ϕ}

Eliminar Prod. unitarias

Terminales: a, b,
No Terminales: A, B, C,
Simbolo Inicial: A
Producciones (P'):

A --> A B b
~~A ---> B~~
~~B ---> C~~
B ---> a b
C ---> a a
C ---> b b

A → a b
A → a a
A → b b

UNITARIOS(A)={B, C}
UNITARIOS(B)={C}
UNITARIOS(C)={ϕ}

Eliminar Prod. unitarias

Terminales: a, b,
No Terminales: A, B, C,
Simbolo Inicial: A
Producciones (P'):

A --> A B b	$A \rightarrow a\ b$
~~A ---> B~~	$A \rightarrow a\ a$
~~B ---> C~~	$A \rightarrow b\ b$
B ---> a b	$B \rightarrow a\ a$
C ---> a a	$B \rightarrow b\ b$
C ---> b b	

UNITARIOS(A)={B, C}
UNITARIOS(B)={C}
UNITARIOS(C)={ϕ}

Eliminar Prod. unitarias

Terminales: a, b,
No Terminales: S, A, B, C,
Simbolo Inicial: S
Producciones:

* S ---> A a b
* S ---> A C
* S ---> B
* A ---> C
* A ---> a
* B ---> C
* B ---> a b
* C ---> a a
* C ---> b b

Eliminar Prod. unitarias

Terminales: a, b,
No Terminales: S, A, B, C,
Simbolo Inicial: S
Producciones (**P'**):

* S ---> A a b
* S ---> A C
* ~~S ---> B~~
* ~~A ---> C~~
* A ---> a
* ~~B ---> C~~
* B ---> a b
* C ---> a a
* C ---> b b

U(S) = {B , C}
U(A) = {C}
U(B) = {C}

S ---> a b
S ---> a a
S ---> b b
A ---> a a
A ---> b b
B ---> a a
B ---> b b

Forma Normal de Chomsky

Una gramática **está en FNC** si:

- No contiene producciones λ (excepto $S \rightarrow \lambda$ si $\lambda \in L(G)$)
- Sus producciones son de la forma:
  - $A \rightarrow a$
  - $A \rightarrow BC$, con $a \in \Sigma_T$, $A,B,C \in \Sigma_N$.
- La primera acción a realizar es limpiar la gramática: Eliminación de símbolos inútiles (no generativos y no accesibles); Eliminación de producciones nulas; Eliminación de producciones unitarias.

Forma Normal de Chomsky(FNC)

- Para cada producción de la forma $A \rightarrow w$, $w= x_1x_2\ldots x_n$,
  $w \in (\Sigma_N \cup \Sigma_T)^+$ con $n \geq 2$
  Para cada x_i, si x_i es un símbolo terminal (x_i=a)
  - Añadir la producción $C_a \rightarrow a$
  - Sustituir x_i por C_a en la producción $A \rightarrow x_1x_2\ldots x_n$
- Para cada producción con la forma $A \rightarrow B_1B_2\ldots B_m$, con $m \geq 3$
  - Añadir (m-2) no terminales ($D_1D_2\ldots D_{m-2}$)
  - La producción $A \rightarrow B_1B_2\ldots B_m$ se sustituye por las producciones:
    $A \rightarrow B_1D_1$
    $D_1 \rightarrow B_2D_2$
    ….
    $D_{m-2} \rightarrow B_{m-1}B_m$

Forma Normal de Chomsky

Terminales: a, b,
No Terminales: S, A, B,
Simbolo Inicial: S
Producciones:
S ---> b A
S ---> a B
A ---> b A A
A ---> a S
A ---> a
B ---> a B B
B ---> b S
B ---> b

La gramática está limpia.

Producciones en FNC

Forma Normal de Chomsky

Terminales: a, b,
No Terminales: S, A, B
Simbolo Inicial: S
Producciones:
S ---> b A
S ---> a B
A ---> b A A
A ---> a S
A ---> a
B ---> a B B
B ---> b S
B ---> b

La gramática está limpia.

Producciones no FNC Longitud 2

Forma Normal de Chomsky

Terminales: a, b,
No Terminales: S, A, B, C
Simbolo Inicial: S
Producciones:
S ---> C A
S ---> a B
A ---> b A A
A ---> a S
A ---> a
B ---> a B B
B ---> b S
B ---> b

La gramática está limpia.

Producciones no FNC Longitud 2

$C \rightarrow b$

Forma Normal de Chomsky

Terminales: a, b,
No Terminales: S, A, B, C
Simbolo Inicial: S
Producciones:
S ---> C A
S ---> a B
A ---> b A A
A ---> a S
A ---> a
B ---> a B B
B ---> b S
B ---> b

La gramática está limpia.

Producciones no FNC Longitud 2

$C \rightarrow b$

Forma Normal de Chomsky

Terminales: a, b,
No Terminales: S, A, B, C
Simbolo Inicial: S
Producciones:
S ---> C A
S ---> a B
A ---> b A A
A ---> a S
A ---> a
B ---> a B B
B ---> C S
B ---> b

$C \rightarrow b$

La gramática está limpia.

Producciones no FNC Longitud 2

Forma Normal de Chomsky

Terminales: a, b,
No Terminales: S, A, B, C
Simbolo Inicial: S
Producciones:
S ---> C A
S ---> a B
A ---> b A A
A ---> a S
A ---> a
B ---> a B B
B ---> C S
B ---> b

$C \rightarrow b$

La gramática está limpia.

Producciones no FNC Longitud 2

Forma Normal de Chomsky

Terminales: a, b,
No Terminales: S, A, B, C, D
Simbolo Inicial: S
Producciones:

S ---> C A
S ---> D B
A ---> b A A
A ---> D S
A ---> a
B ---> a B B
B ---> C S
B ---> b

$C \rightarrow b$

$D \rightarrow a$

La gramática está limpia.

Producciones no FNC Longitud 2

Forma Normal de Chomsky

Terminales: a, b,
No Terminales: S, A, B, C, D, E
Simbolo Inicial: S
Producciones:

S ---> C A
S ---> D B
A ---> b A A
A ---> D S
A ---> a
B ---> a B B
B ---> C S
B ---> b

$C \rightarrow b$
$D \rightarrow a$
$E \rightarrow AA$

La gramática está limpia.

Producciones no FNC Longitud 3

Forma Normal de Chomsky

Terminales: a, b,
No Terminales: S, A, B, C, D, E
Simbolo Inicial: S
Producciones:

S ---> C A
S ---> D B
A ---> C E
A ---> D S
A ---> a
B ---> a B B
B ---> C S
B ---> b

$C \rightarrow b$
$D \rightarrow a$
$E \rightarrow AA$

La gramática está limpia.

Producciones no FNC Longitud 3

Forma Normal de Chomsky

Terminales: a, b,
No Terminales: S, A, B, C, D, E, F
Simbolo Inicial: S

La gramática está limpia.

Producciones:

Producciones no FNC Longitud 3

S ---> C A
S ---> D B
A ---> C E
A ---> D S
A ---> a
B ---> a B B
B ---> C S
B ---> b

C → b
D → a
E → AA
F → BB

Forma Normal de Chomsky

Terminales: a, b,
No Terminales: S, A, B, C, D, E, F
Simbolo Inicial: S

La gramática está limpia.

Producciones:

Producciones no FNC Longitud 3

S ---> C A
S ---> D B
A ---> C E
A ---> D S
A ---> a
B ---> D F
B ---> C S
B ---> b

C → b
D → a
E → AA
F → BB

Forma Normal Chomsky

Terminales: a, b,
No Terminales: S, A, B, C,
Simbolo Inicial: S
Producciones (**P'**):

* S ---> A a b
* S ---> A C
* S ---> a b
* S ---> a a
* S ---> b b
* A ---> a
* A ---> a a
* A ---> b b
* B ---> a b
* B ---> a a
* B ---> b b
* C ---> b b
* C ---> a a

Forma Normal Chomsky

Terminales: a, b,
No Terminales: S, A, B, C,
Simbolo Inicial: S
Producciones (**P'**):

* F ---> D E
* S ---> A F
* S ---> A C
* S ---> E E
* S ---> D D
* S ---> D E
* A ---> a
* A ---> E E
* A ---> D D
* B ---> D E
* B ---> E E
* B ---> D D
* C ---> D D
* C ---> E E
* D ---> a
* E ---> b

Forma Normal Chomsky

Terminales: a, b,
No Terminales: S, A, B,
Simbolo Inicial: S
Producciones:

* S ---> b A
* S ---> a B
* A ---> b A A
* A ---> a S
* A ---> a
* B ---> a B B
* B ---> b S
* B ---> b

Forma Normal Chomsky

Terminales: a, b,
No Terminales: S, C, A, D, B, E, F,
Simbolo Inicial: S
Producciones:

* S ---> C A
* S ---> D B
* A ---> a
* E ---> A A
* A ---> C E
* A ---> D S
* B ---> b
* F ---> B B
* B ---> D F
* B ---> C S
* C--> b
* D → a

Eliminación de la recursión a izquierda

Dada una GLC *G*, se dice que una producción es recursiva a izquierda si tiene la forma:

$$A \rightarrow A\,\alpha, \text{ donde } A \in \Sigma_N\ ,\ \alpha \in (\Sigma_N\ \cup \Sigma_T)^*$$

Algoritmo:

Para cada símbolo no terminal *A* con recursividad a izquierda

$$A \rightarrow A\alpha_1 \mid \ldots \mid A\alpha_n \mid \beta_1 \mid \ldots \mid \beta_m$$

Crear un nuevo no terminal Z_i y sustituir el anterior conjunto de producciones por el siguiente:

$$A \rightarrow \beta_1 \mid \ldots \mid \beta_m \mid \beta_1 Z_i \mid \ldots \mid \beta_m Z_i$$

$$Z_i \rightarrow \alpha_1 \mid \ldots \mid \alpha_n \mid \alpha_1 Z_i \mid \ldots \mid \alpha_n Z_i$$

El. Recursividad Izqda.

S → Sa|Sb|cA
A → Aa|a|e

- **Eliminando la recursividad izquierda para S**:
  S → cA | cAZ1
  Z1 → a | b | aZ1 | bZ1
  A → Aa|a|e
- **Eliminando la recursividad izquierda para A**:

El. Recursividad Izqda.

S → Sa|Sb|cA
A → Aa|a|e

- **Eliminando la recursividad izquierda para S:**
  S → cA | cAZ1
  Z1 → a | b | aZ1 | bZ1
  A → Aa|a|e
- **Eliminando la recursividad izquierda para A**:
  S → cA | cAZ1
  Z1 → a | b | aZ1 | bZ1
  A → a| aZ2 | e | eZ2
  Z2 → a | aZ2

Forma Normal de Greibach (FNG)(1)

Una gramática está en Forma Normal Greibach si sus producciones son de la forma:

$A \rightarrow a\,\alpha$, donde $A \in \Sigma_N$, $a \in \Sigma_T$ y $\alpha \in \Sigma_N^*$. (si $\lambda \in L(G)$ $S \rightarrow \lambda$)

MÉTODO GENERAL:

- Partimos de una gramática en FNC ($A \rightarrow a$ o $A \rightarrow BC$)
- Obtenemos un nuevo conjunto de producciones P' tal que:
  - $\Sigma_N = \{A_1, A_2, \ldots, A_n\}$ con $A_1 = S$
  - $P' = \{A_r \rightarrow A_s\,\alpha$ con $r < s$ o bien

    $A_r \rightarrow a\,\alpha$ (FNG) $\}$
- Para cada A_i, desde i=n-1 hasta 1

  Para la producción $A_i \rightarrow A_j\,\alpha$ (i<j), sustituir A_j por su definición

Forma Normal de Greibach(FNG)(2)

MÉTODO:

1. Renombrar los No terminales $\Sigma_N = \{A_1, A_2, \ldots, A_n\}$ donde $A_1 = S$ y obtener las nuevas producciones con estos No terminales.
2. Para cada No terminal A_i desde i= 1hasta n hacer:

   (2.1) Si A_i contiene produciones con recursión a izquierda, eliminar la recursión a Izquierda de A_i.

   (2.2) Para cada producción de A_i , $A_i \rightarrow A_j \alpha$, i>j

   (a) Eliminar $A_i \rightarrow A_j \alpha$, siendo $A_j \rightarrow \beta_1 \mid \beta_2 \mid \ldots \mid \beta_m$

   (b) Para cada β_k , k=1 …m

   Añadir $A_i \rightarrow \beta_k \alpha$ al conjunto de producciones

   (c) Para cada producción $A_i \rightarrow \beta_k \alpha$ (k=1..m)

   Si $\beta_k = A_r \alpha'$

   Si i < r condicción satisfecha

   Si i > r volver al paso (a) con $A_i \rightarrow A_r \alpha' \alpha$

   Si i=r Eliminar la recursión a Izquierda y volver a (2.2) con el nuevo A_i

Forma Normal de Greibach(FNG)(3)

3. Para cada No terminal A_i desde i= n-1 hasta 1 hacer:
   Para cada producción de A_i , $A_i \rightarrow A_j \alpha$
   (a) Eliminar $A_i \rightarrow A_j \alpha$, siendo $A_j \rightarrow \beta_1 \mid \beta_2 \mid \ldots \mid \beta_m$
   (b) Para cada β_k , k=1 …m
   Añadir $A_i \rightarrow \beta_k \alpha$ al conjunto de producciones

4. Para cada No terminal Z_i (no terminal nuevo al aplicar R.I.)
   Para cada producción de Z_i , $Z_i \rightarrow A_j \alpha$,
   (a) Eliminar $Z_i \rightarrow A_j \alpha$, siendo $A_j \rightarrow \beta_1 \mid \beta_2 \mid \ldots \mid \beta_m$
   (b) Para cada β_k , k=1 …m
   Añadir $Z_i \rightarrow \beta_k \alpha$ al conjunto de producciones

Forma Normal de Greibach

Terminales: a, b,
No Terminales: A, B, C,
Simbolo Inicial: A Está en FNC
Producciones:

- A ---> B C
- B ---> C A
- B ---> a
- C ---> A B
- C ---> b

Forma Normal de Greibach

Terminales: a, b,
No Terminales: A1, A2, A3
Simbolo Inicial: A1
Producciones:

- A1 ---> A2 A3
- A2 ---> A3 A1
- A2 ---> a
- A3 ---> A1 A2
- A3 ---> b

Simbolo "A" sustituido por "A1"
Simbolo "B" sustituido por "A2"
Simbolo "C" sustituido por "A3"

Forma Normal de Greibach

Terminales: a, b,
No Terminales: A1, A2, A3
Simbolo Inicial: A1
Producciones:

- A1 ---> A2 A3
- A2 ---> A3 A1
- A2 ---> a
- A3 ---> A1 A2
- A3 ---> b

Forma Normal de Greibach

Terminales: a, b,
No Terminales: A1, A2, A3
Simbolo Inicial: A1
Producciones:

- A1 ---> A2 A3
- A2 ---> A3 A1
- A2 ---> a
- A3 ---> A2 A3 A2
- A3 ---> b

Forma Normal de Greibach

Terminales: a, b,
No Terminales: A1, A2, A3
Simbolo Inicial: A1
Producciones:

- A1 ---> A2 A3
- A2 ---> A3 A1
- A2 ---> a
- A3 ---> A2 A3 A2
- A3 ---> b

Forma Normal de Greibach

Terminales: a, b,
No Terminales: A1, A2, A3
Simbolo Inicial: A1
Producciones:

- A1 ---> A2 A3
- A2 ---> A3 A1
- A2 ---> a
- A3 ---> A3 A1 A3 A2
- A3 ---> b

Forma Normal de Greibach

Terminales: a, b,
No Terminales: A1, A2, A3
Simbolo Inicial: A1
Producciones:

- A1 ---> A2 A3
- A2 ---> A3 A1
- A2 ---> a
- A3 ---> A3 A1 A3 A2
- A3 ---> a A3 A2
- A3 ---> b

Forma Normal de Greibach

Terminales: a, b,
No Terminales: A1, A2, A3, Z1
Simbolo Inicial: A1

Recursividad Izqda.

Producciones:

- A1 ---> A2 A3
- A2 ---> A3 A1
- A2 ---> a
- ~~A3 ---> A3 A1 A3 A2~~
- A3 ---> a A3 A2 | a A3 A2 Z1
- A3 ---> b | b Z1
- Z1 ---> A1 A3 A2 | A1 A3 A2 Z1

Forma Normal de Greibach

Producciones:

- A1 ---> A2 A3
- ~~A2 ---> A3 A1~~
- A2 ---> a A3 A2 A1
- A2 ---> a
- A3 ---> a A3 A2
- A3 ---> a A3 A2 Z1
- A3 ---> b
- A3 ---> b Z1
- Z1 ---> A1 A3 A2
- Z1 ---> A1 A3 A2 Z1

Llego al tercer paso del algoritmo

Forma Normal de Greibach

Producciones:

- A1 ---> A2 A3
- ~~A2 ---> A3 A1~~
- A2 ---> a A3 A2 A1
- A2 ---> a A3 A2 Z1 A1
- A2 ---> a
- A3 ---> a A3 A2
- A3 ---> a A3 A2 Z1
- A3 ---> b
- A3 ---> b Z1
- Z1 ---> A1 A3 A2
- Z1 ---> A1 A3 A2 Z1

Forma Normal de Greibach

Producciones:

- A1 ---> A2 A3
- ~~A2 ---> A3 A1~~
- A2 ---> a A3 A2 A1
- A2 ---> a A3 A2 Z1 A1
- A2 ---> a
- A3 ---> a A3 A2
- A3 ---> a A3 A2 Z1
- A3 ---> b
- A3 ---> b Z1
- Z1 ---> A1 A3 A2
- Z1 ---> A1 A3 A2 Z1

Forma Normal de Greibach

Producciones:

- A1 ---> A2 A3
- ~~A2 ---> A3 A1~~
- A2 ---> a A3 A2 A1
- A2 ---> a A3 A2 Z1 A1
- A2 ---> b A1
- A2 ---> a
- A3 ---> a A3 A2
- A3 ---> a A3 A2 Z1
- A3 ---> b
- A3 ---> b Z1
- Z1 ---> A1 A3 A2
- Z1 ---> A1 A3 A2 Z1

Forma Normal de Greibach

Producciones:

- A1 ---> A2 A3
- ~~A2 ---> A3 A1~~
- A2 ---> a A3 A2 A1
- A2 ---> a A3 A2 Z1 A1
- A2 ---> b A1
- A2 ---> b Z1 A1
- A2 ---> a
- A3 ---> a A3 A2
- A3 ---> a A3 A2 Z1
- A3 ---> b
- A3 ---> b Z1
- Z1 ---> A1 A3 A2
- Z1 ---> A1 A3 A2 Z1

Forma Normal de Greibach

Producciones:

- ~~A1 ---> A2 A3~~
- A2 ---> a A3 A2 A1
- A2 ---> a A3 A2 Z1 A1
- A2 ---> b A1
- A2 ---> b Z1 A1
- A2 ---> a
- A3 ---> a A3 A2
- A3 ---> a A3 A2 Z1
- A3 ---> b
- A3 ---> b Z1
- Z1 ---> A1 A3 A2
- Z1 ---> A1 A3 A2 Z1

Forma Normal de Greibach

Producciones:

- ~~A1 ---> A2 A3~~
- A1 ----> a A3 A2 A1 A3
- A2 ---> a A3 A2 A1
- A2 ---> a A3 A2 Z1 A1
- A2 ---> b A1
- A2 ---> b Z1 A1
- A2 ---> a
- A3 ---> a A3 A2
- A3 ---> a A3 A2 Z1
- A3 ---> b
- A3 ---> b Z1
- Z1 ---> A1 A3 A2
- Z1 ---> A1 A3 A2 Z1

Forma Normal de Greibach

Producciones:

- ~~A1 ---> A2 A3~~
- A1 ----> a A3 A2 A1 A3
- A1 ---> a A3 A2 Z1 A1 A3
- A1 ---> b A1 A3
- A1 ---> b Z1 A1 A3
- A1 ---> a A3
- A2 ---> a A3 A2 A1
- A2 ---> a A3 A2 Z1 A1
- A2 ---> b A1
- A2 ---> b Z1 A1
- A2 ---> a
- A3 ---> a A3 A2
- A3 ---> a A3 A2 Z1
- A3 ---> b
- A3 ---> b Z1
- Z1 ---> A1 A3 A2
- Z1 ---> A1 A3 A2 Z1

Forma Normal de Greibach

Producciones:

- ~~A1 ---> A2 A3~~
- A1 ----> a A3 A2 A1 A3
- A1 ---> a A3 A2 Z1 A1 A3
- A1 ---> b A1 A3
- A1 ---> b Z1 A1 A3
- A1 ---> a A3
- A2 ---> a A3 A2 A1
- A2 ---> a A3 A2 Z1 A1
- A2 ---> b A1
- A2 ---> b Z1 A1
- A2 ---> a
- A3 ---> a A3 A2
- A3 ---> a A3 A2 Z1
- A3 ---> b
- A3 ---> b Z1
- Z1 ---> A1 A3 A2
- Z1 ---> A1 A3 A2 Z1

Forma Normal de Greibach

- A1 ----> a A3 A2 A1 A3
- A1 ---> a A3 A2 Z1 A1 A3
- A1 ---> b A1 A3
- A1 ---> b Z1 A1 A3
- A1 ---> a A3
- A2 ---> a A3 A2 A1
- A2 ---> a A3 A2 Z1 A1
- A2 ---> b A1
- A2 ---> b Z1 A1
- A2 ---> a
- A3 ---> a A3 A2
- A3 ---> a A3 A2 Z1
- A3 ---> b
- A3 ---> b Z1
- Z1 ---> A1 A3 A2
- Z1 ---> A1 A3 A2 Z1

Paso 4 del Algoritmo

Forma Normal de Greibach

- A1 ----> a A3 A2 A1 A3
- A1 ---> a A3 A2 Z1 A1 A3
- A1 ---> b A1 A3
- A1 ---> b Z1 A1 A3
- A1 ---> a A3
- A2 ---> a A3 A2 A1
- A2 ---> a A3 A2 Z1 A1
- A2 ---> b A1
- A2 ---> b Z1 A1
- A2 ---> a
- A3 ---> a A3 A2
- A3 ---> a A3 A2 Z1
- A3 ---> b
- A3 ---> b Z1
- Z1 ---> A1 A3 A2
- Z1 ---> A1 A3 A2 Z1

Forma Normal de Greibach

- A1 ----> a A3 A2 A1 A3
- A1 ---> a A3 A2 Z1 A1 A3
- A1 ---> b A1 A3
- A1 ---> b Z1 A1 A3
- A1 ---> a A3
- A2 ---> a A3 A2 A1
- A2 ---> a A3 A2 Z1 A1
- A2 ---> b A1
- A2 ---> b Z1 A1
- A2 ---> a
- A3 ---> a A3 A2
- A3 ---> a A3 A2 Z1
- A3 ---> b
- A3 ---> b Z1
- Z1 ---> a A3 A2 A1 A3 A3 A2
- Z1 ---> a A3 A2 Z1 A1 A3 A3 A2
- Z1 ---> b A1 A3 A3 A2
- Z1 ---> b Z1 A1 A3 A3 A2
- Z1 ---> a A3 A3 A2
- Z1 ---> a A3 A2 A1 A3 A3 A2 Z1
- Z1 ---> a A3 A2 Z1 A1 A3 A3 A2 Z1
- Z1 ---> b A1 A3 A3 A2 Z1
- Z1 ---> b Z1 A1 A3 A3 A2 Z1
- Z1 ---> a A3 A3 A2 Z1

Forma Normal de Greibach

Terminales: a, b,
No Terminales: A, B, C,
Simbolo Inicial: A
Producciones:

* A ---> B C
* B ---> C A
* B ---> a
* C ---> A B
* C ---> b

Forma Normal de Greibach

Terminales: a, b,
No Terminales: A, B, C,
Simbolo Inicial: A
Producciones:

* A1 ---> A2 A3
* A2 ---> A3 A1
* A2 ---> a
* A3 ---> A1 A2
* A3 ---> b

Simbolo "A" sustituido por "A1"
Simbolo "B" sustituido por "A2"
Simbolo "C" sustituido por "A3"

Forma Normal de Greibach

Terminales: a, b,
No Terminales: A, B, C,
Simbolo Inicial: A
Producciones:

* A1 ---> A2 A3
* A2 ---> A3 A1
* A2 ---> a
* A3 ---> A1 A2
* A3 ---> b

Simbolo "A" sustituido por "A1"
Simbolo "B" sustituido por "A2"
Simbolo "C" sustituido por "A3"

Forma Normal de Greibach

Terminales: a, b,
No Terminales: A, B, C,
Simbolo Inicial: A
Producciones:

* A1 ---> A2 A3
* A2 ---> A3 A1
* A2 ---> a
* A3 ---> A1 A2
* A3 ---> b

Simbolo "A" sustituido por "A1"
Simbolo "B" sustituido por "A2"
Simbolo "C" sustituido por "A3"

Forma Normal de Greibach

Terminales: a, b,
No Terminales: A, B, C,
Simbolo Inicial: A
Producciones:

* A1 ---> A2 A3
* A2 ---> A3 A1
* A2 ---> a
* A3 ---> A2 A3 A2
* A3 ---> b

Simbolo "A" sustituido por "A1"
Simbolo "B" sustituido por "A2"
Simbolo "C" sustituido por "A3"

Forma Normal de Greibach

Terminales: a, b,
No Terminales: A, B, C,
Simbolo Inicial: A
Producciones:

* A1 ---> A2 A3
* A2 ---> A3 A1
* A2 ---> a
* A3 ---> A2 A3 A2
* A3 ---> b

Simbolo "A" sustituido por "A1"
Simbolo "B" sustituido por "A2"
Simbolo "C" sustituido por "A3"

Forma Normal de Greibach

Terminales: a, b,
No Terminales: A, B, C,
Simbolo Inicial: A
Producciones:

* A1 ---> A2 A3
* A2 ---> A3 A1
* A2 ---> a
* A3 ---> A3 A1 A3 A2
* A3 ---> a A3 A2
* A3 ---> b

Simbolo "A" sustituido por "A1"
Simbolo "B" sustituido por "A2"
Simbolo "C" sustituido por "A3"

Forma Normal de Greibach

Terminales: a, b,
No Terminales: A, B, C,
Simbolo Inicial: A
Producciones:

* A1 ---> A2 A3
* A2 ---> A3 A1
* A2 ---> a
* A3 ---> A3 A1 A3 A2
* A3 ---> a A3 A2
* A3 ---> b

Simbolo "A" sustituido por "A1"
Simbolo "B" sustituido por "A2"
Simbolo "C" sustituido por "A3"

Forma Normal de Greibach

Terminales: a, b,
No Terminales: A, B, C,
Simbolo Inicial: A
Producciones:

* A1 ---> A2 A3
* A2 ---> A3 A1
* A2 ---> a
* A3 ---> A3 A1 A3 A2
* A3 ---> a A3 A2
* A3 ---> b

RECURSIVIDAD

Simbolo "A" sustituido por "A1"
Simbolo "B" sustituido por "A2"
Simbolo "C" sustituido por "A3"

Forma Normal de Greibach

Terminales: a, b,
No Terminales: A, B, C,
Simbolo Inicial: A
Producciones:

* A1 ---> A2 A3
* A2 ---> A3 A1
* A2 ---> a
* ~~A3 ---> A3 A1 A3 A2~~ ~~RECURSIVIDAD~~
* A3 ---> a A3 A2 | a A3 A2 Z1
* A3 ---> b | bZ1
* Z1 → A1 A3 A2 | A1 A3 A2 Z1

Simbolo "A" sustituido por "A1"
Simbolo "B" sustituido por "A2"
Simbolo "C" sustituido por "A3"

Forma Normal de Greibach

Terminales: a, b,
No Terminales: A, B, C,
Simbolo Inicial: A
Producciones:

* A1 ---> A2 A3
* A2 ---> A3 A1
* A2 ---> a
* A3 ---> a A3 A2 | a A3 A2 Z1
* A3 ---> b | bZ1
* Z1 → A1 A3 A2 | A1 A3 A2 Z1

Forma Normal de Greibach

Terminales: a, b,
No Terminales: A, B, C,
Simbolo Inicial: A
Producciones:

* A1 ---> A2 A3
* A2 ---> A3 A1
* A2 ---> a
* A3 ---> a A3 A2 | a A3 A2 Z1
* A3 ---> b | bZ1
* Z1 → A1 A3 A2 | A1 A3 A2 Z1

Forma Normal de Greibach

Terminales: a, b,
No Terminales: A, B, C,
Simbolo Inicial: A
Producciones:

* A1 ---> A2 A3
* A2 ---> A3 A1
* A2 ---> a
* A3 ---> a A3 A2 | a A3 A2 Z1
* A3 ---> b | bZ1
* Z1 → A1 A3 A2 | A1 A3 A2 Z1

Forma Normal de Greibach

Terminales: a, b,
No Terminales: A, B, C,
Simbolo Inicial: A
Producciones:

* A1 ---> A2 A3
* A2 ---> a A3 A2 A1 | a A3 A2 Z1 A1
* A2 → b A1 | b Z1 A1
* A2 ---> a
* A3 ---> a A3 A2 | a A3 A2 Z1
* A3 ---> b | bZ1
* Z1 → A1 A3 A2 | A1 A3 A2 Z1

Forma Normal de Greibach

Terminales: a, b,
No Terminales: A, B, C,
Simbolo Inicial: A
Producciones:

* A1 ---> A2 A3
* A2 ---> a A3 A2 A1 | a A3 A2 Z1 A1
* A2 → b A1 | b Z1 A1
* A2 ---> a
* A3 ---> a A3 A2 | a A3 A2 Z1
* A3 ---> b | bZ1
* Z1 → A1 A3 A2 | A1 A3 A2 Z1

Forma Normal de Greibach

Terminales: a, b,
No Terminales: A, B, C,
Simbolo Inicial: A
Producciones:

* A1 ---> a A3 A2 A1 A3 | a A3 A2 Z1 A1 A3
* A1 ---> b A1 A3 | b Z1 A1 A3
* A1 ---> a A3
* A2 ---> a A3 A2 A1 | a A3 A2 Z1 A1
* A2 → b A1 | b Z1 A1
* A2 ---> a
* A3 ---> a A3 A2 | a A3 A2 Z1
* A3 ---> b | bZ1
* Z1 → A1 A3 A2 | A1 A3 A2 Z1

Forma Normal de Greibach

Terminales: a, b,
No Terminales: A, B, C,
Simbolo Inicial: A
Producciones:

* A1 ---> a A3 A2 A1 A3 | a A3 A2 Z1 A1 A3
* A1 ---> b A1 A3 | b Z1 A1 A3
* A1 ---> a A3
* A2 ---> a A3 A2 A1 | a A3 A2 Z1 A1
* A2 → b A1 | b Z1 A1
* A2 ---> a
* A3 ---> a A3 A2 | a A3 A2 Z1
* A3 ---> b | bZ1
* Z1 → A1 A3 A2 | A1 A3 A2 Z1

Forma Normal de Greibach

* A1 ---> a A3 A2 A1 A3 | a A3 A2 Z1 A1 A3
* A1 ---> b A1 A3 | b Z1 A1 A3
* A1 ---> a A3
* A2 ---> a A3 A2 A1 | a A3 A2 Z1 A1
* A2 → b A1 | b Z1 A1
* A2 ---> a
* A3 ---> a A3 A2 | a A3 A2 Z1
* A3 ---> b | bZ1
* Z1 → a A3 A2 A1 A3 A3 A2 | a A3 A2 A1 A3 A3 A2 Z1
* Z1 → a A3 A2 Z1 A1 A3 A3 A2 | a A3 A2 Z1 A1 A3 A3 A2 Z1
* Z1 → b A1 A3 A3 A2 | b A1 A3 A3 A2 Z1
* Z1 → b Z1 A1 A3 A3 A2 | b Z1 A1 A3 A3 A2 Z1
* Z1 → a A3 A3 A2 | a A3 A3 A2 Z1

Forma Normal de Greibach

* A1 ---> a A3 A2 A1 A3 | a A3 A2 Z1 A1 A3
* A1 ---> b A1 A3 | b Z1 A1 A3
* A1 ---> a A3
* A2 ---> a A3 A2 A1 | a A3 A2 Z1 A1
* A2 → b A1 | b Z1 A1
* A2 ---> a
* A3 ---> a A3 A2 | a A3 A2 Z1
* A3 ---> b | bZ1
* Z1 → a A3 A2 A1 A3 A3 A2 | a A3 A2 A1 A3 A3 A2 Z1
* Z1 → a A3 A2 Z1 A1 A3 A3 A2 | a A3 A2 Z1 A1 A3 A3 A2 Z1
* Z1 → b A1 A3 A3 A2 | b A1 A3 A3 A2 Z1
* Z1 → b Z1 A1 A3 A3 A2 | b Z1 A1 A3 A3 A2 Z1
* Z1 → a A3 A3 A2 | a A3 A3 A2 Z1

CUESTIONARIO

- A) L = {$a^m b^n c$, m par y n impar}
- B) Lenguaje de los números binarios enteros negativos de longitud mayor o igual a 3 (-1010).

Pasar a FNC

S ---> a c C | A c

A ---> a A | c C | $

B ---> b B | $

C ---> b a | c

Pasar a FNC

S ---> c

F ---> D C

S ---> B F

S ---> A D

A ---> a

A ---> B A

A ---> D C

C ---> E B

C ---> c

B ---> a

D ---> c

E ---> b