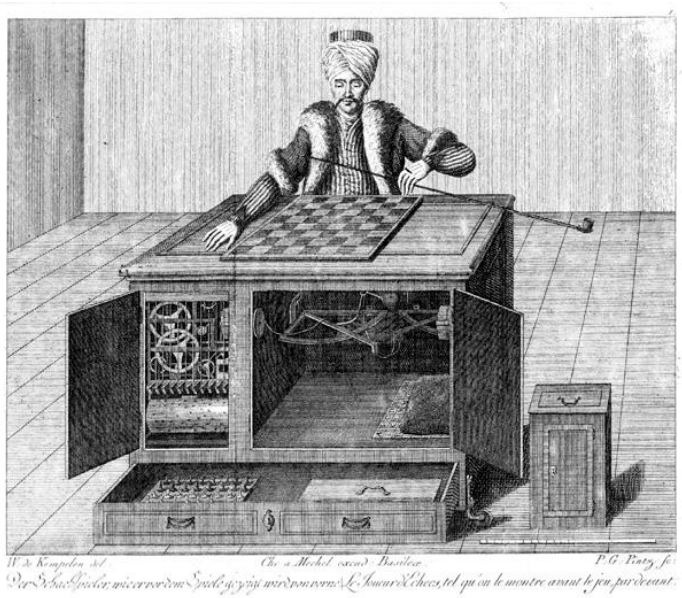
Autómatas

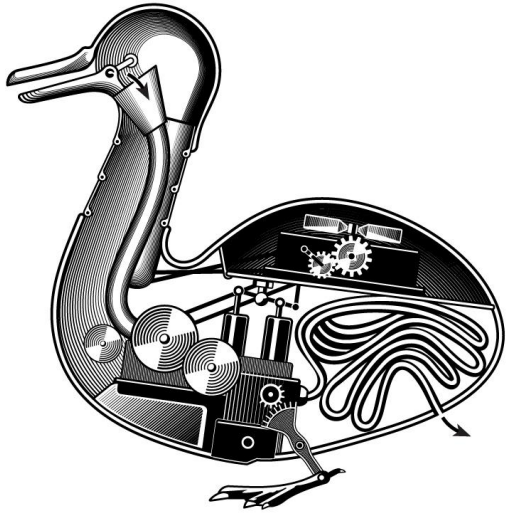
AUTÓMATA FINITO DETERMINISTA (AFD)

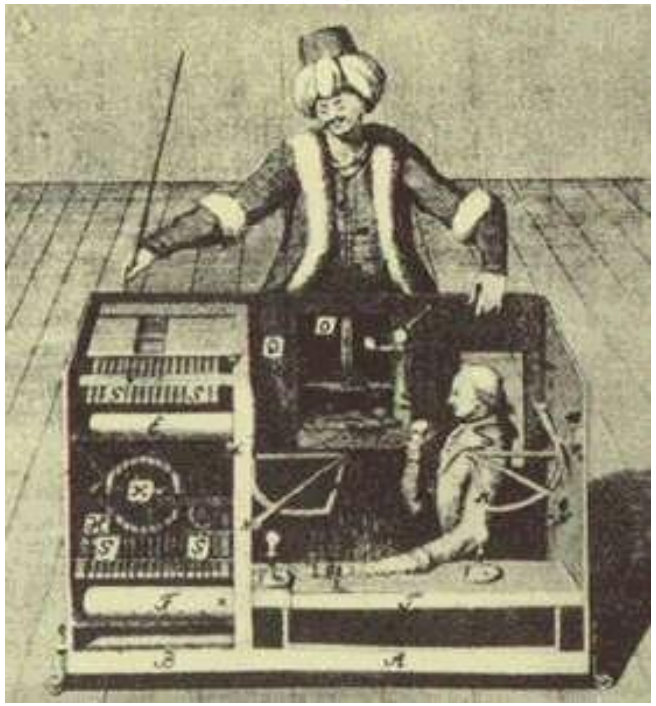
Contenido

Introducción

Un ejemplo de aplicación





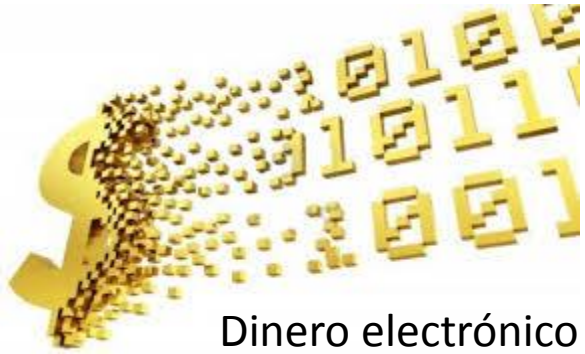



AUTOMATA
User Guide: Maintenance, Troubleshooting and Technical Specifications.
A film by GABE IBÁÑEZ.
We bring you the future.

Un ejemplo de autómatas: Dinero electrónico

Tienda

Cliente



Banco

Reglas

- Participantes: la cliente, la tienda, el banco.
- Solo hay un archivo de dinero electrónico.
- El cliente puede decidir transferir este archivo a la tienda la cual lo reenviará al banco (es decir, solicita al banco que emita un nuevo archivo que refleje que el dinero pertenece a la tienda y no más al cliente) y suministra los bienes al cliente.
- El cliente tiene la opción de cancelar el archivo (cancelar la compra); esto es, el cliente puede pedir al banco que devuelva el dinero a su cuenta, anulando la posibilidad de gastarlo.
- La interacción entre los participantes se limita a cinco sucesos:

Reglas

1. El cliente decide *pagar*. Es decir, el cliente envía el dinero a la tienda.
2. El cliente decide *cancelar* el pago. El dinero se envía al banco con un mensaje que indica que el dinero se ha añadido a la cuenta bancaria del cliente.
3. La tienda *suministra* los bienes al cliente.
4. La tienda *libra* el dinero. Es decir, el dinero se envía al banco con la solicitud de que su valor se asigne a la cuenta de la tienda.
5. El banco *transfiere* el dinero creando un nuevo archivo de dinero electrónico cifrado y se lo envía a la tienda.

El protocolo

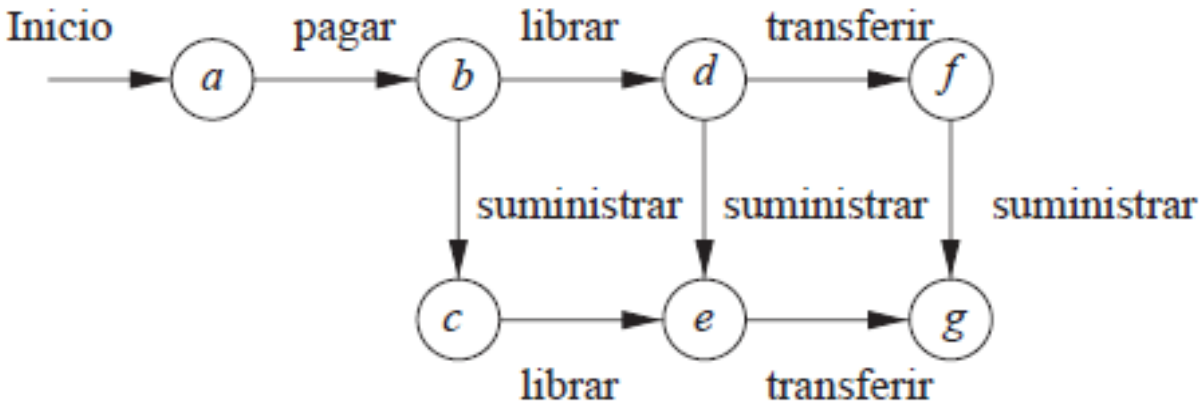
Los tres actores se deben comportar de forma responsable:

- La cliente no debe intentar copiar el archivo de dinero electrónico y emplearlo para pagar varias veces; tampoco debe pagar y cancelar el pago para recibir los bienes gratuitamente.
- El banco debe garantizar que dos tiendas no puedan liberar el mismo archivo de dinero; tampoco debe permitir que el mismo archivo de dinero sea a la vez cancelado por el cliente y liberado por la tienda.
- La tienda debe asegurarse de no suministrar bienes hasta haber recibido el dinero electrónico válido.

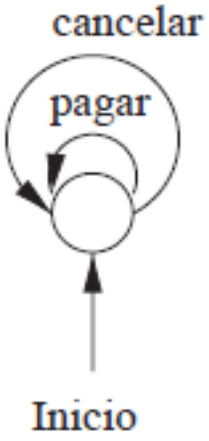
El protocolo

- Los protocolos pueden representarse de forma gráfica mediante un autómata finito
- Cada estado del autómata representa una situación en la que puede encontrarse uno de los participantes.
- Un estado “recuerda” que sucesos importantes han ocurrido y qué sucesos todavía no han tenido lugar.
- Las transiciones entre estados se producen cuando tiene lugar uno de los cinco sucesos descritos anteriormente.

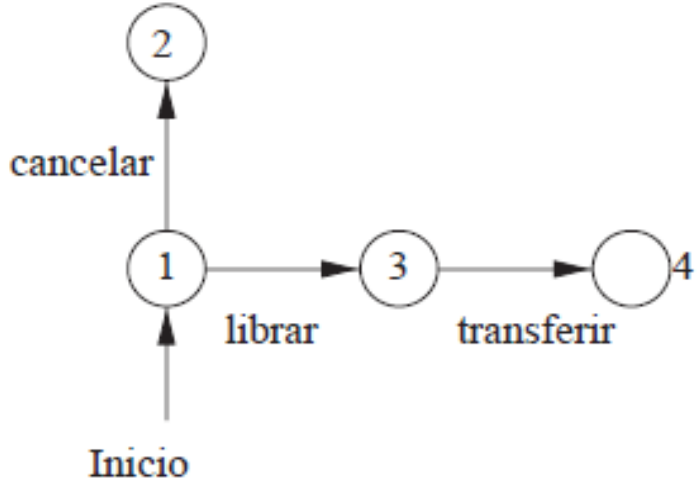
Los autómatas para cada actor



(a) Tienda

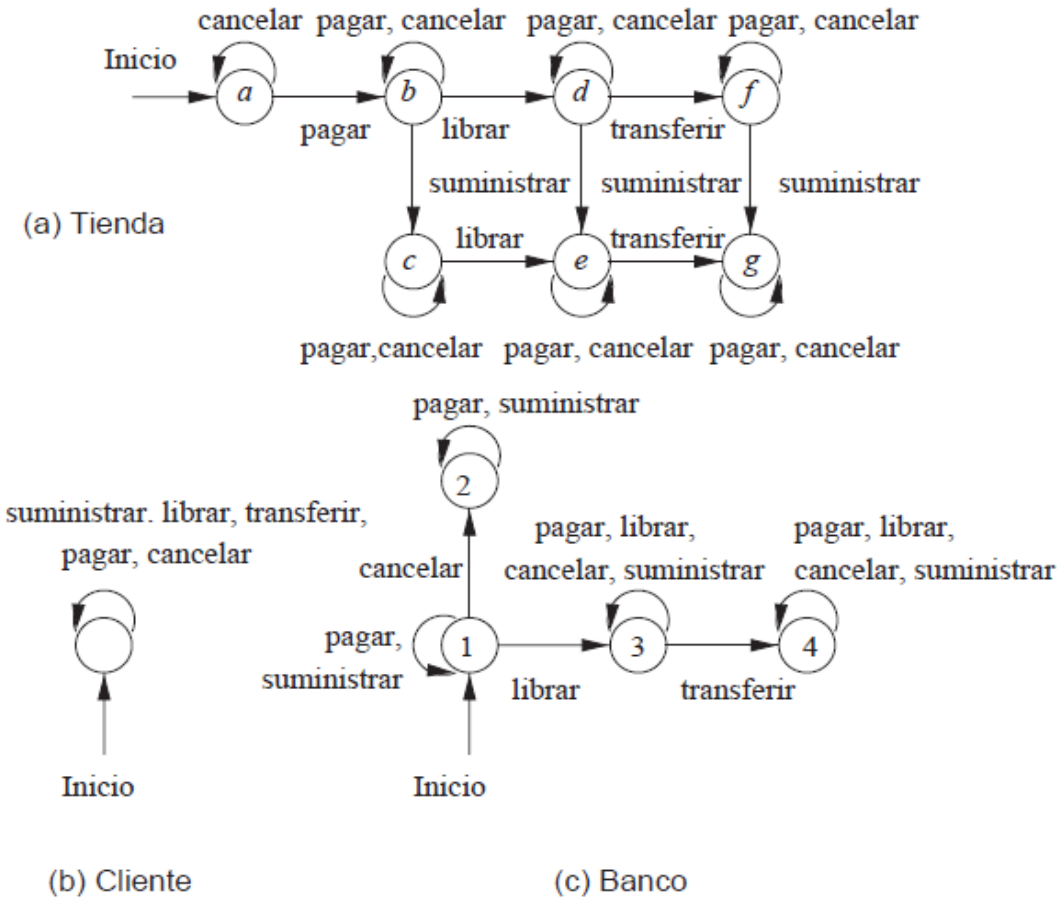


(b) Cliente

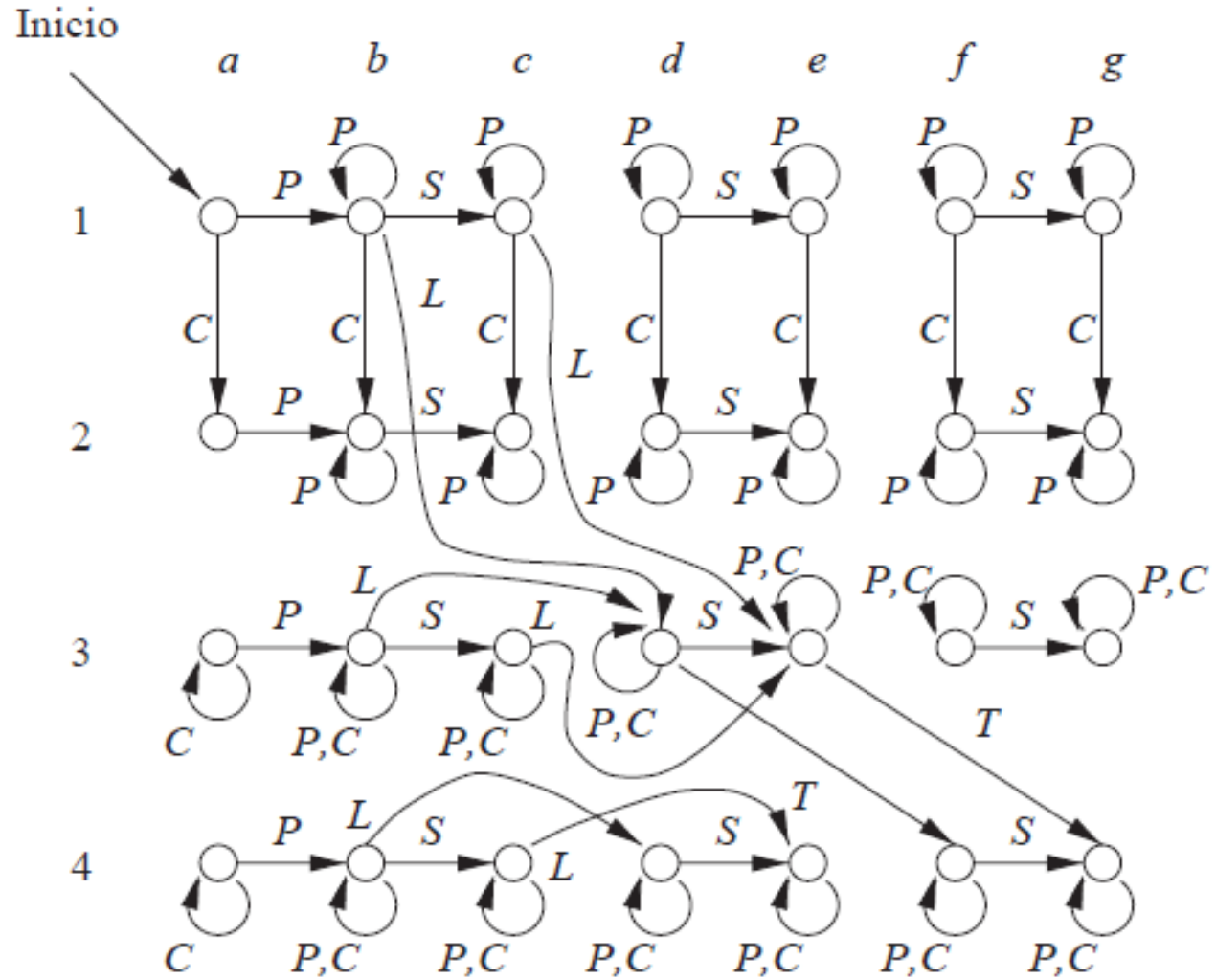


(c) Banco

Agregando transiciones adicionales (para evitar que el autómata detenga su ejecución): acción *cancelar.*



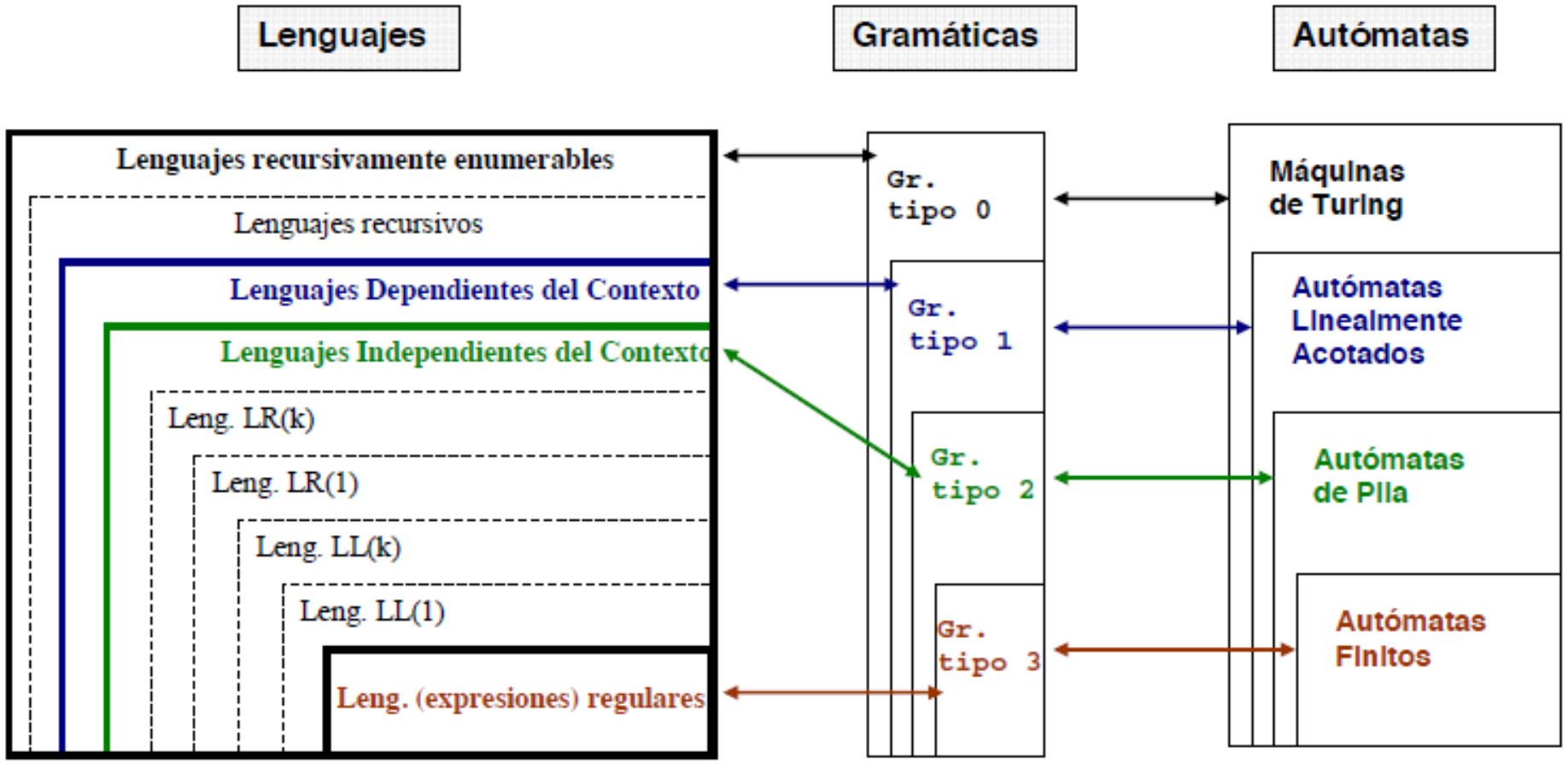
Un autómata para el sistema completo



P = Pagar
S = Suministrar
C = Cancelar
T = Transferir
L = Librar

El autómata producto para la tienda y el banco

Contexto de nuestro curso de Teoría Computacional



Jerarquía de lenguajes, gramáticas y autómatas

Componentes de un compilador

- **Analizador léxico (scanner)**: Lee los caracteres del programa fuente, uno a uno, del archivo de entrada y va formando grupos de caracteres con alguna relación entre sí (*tokens*).

  Tipos de *tokens*:

  Cadenas específicas, como las palabras reservadas (*if, while,* ...), signos de puntuación (., ,, =, ...), operadores aritméticos (+,*, ...) y lógicos (*AND, OR,* ...), etc. Habitualmente, las cadenas específicas no tienen asociado ningún valor, sólo su tipo.

  Cadenas no específicas, como los identificadores o las constantes numéricas o de texto. Las cadenas no específicas siempre tienen tipo y valor. Por ejemplo, si **dato** es el nombre de una variable, el tipo del token será *identificador* y su valor será *dato*.

- Los autómatas finitos son usados para el diseño de analizadores léxicos.
- El analizador léxico es una subrutina del analizador sintáctico.

Componentes de un compilador

- **Analizador sintáctico (*parser*)**: recibe como entrada los *tokens* generados por el analizador léxico y comprueba si estos tokens van llegando en el orden correcto.
- Si no hay errores, la salida del analizador sintáctico es un árbol semántico.
- Si el programa no es correcto se generarán los mensajes de error correspondientes.
- Los autómatas de pila son usados en el diseño de analizadores sintácticos.

Componentes de un compilador

- **Analizador semántico**: su propósito es determinar si el significado de las diferentes instrucciones del programa es válido.
- Para conseguirlo tendrá que calcular y analizar información asociada a las sentencias del programa:
  - Determinar el tipo de los resultados intermedios de las expresiones.
  - Comprobar que los argumentos de un operador pertenecen al conjunto de los operandos posibles.
  - Comprobar que los operandos son compatibles entre sí, etc.
- La salida de esta fase de compilación es un árbol semántico. Éste es una ampliación de un árbol sintáctico en el que cada rama del árbol ha adquirido, además, el significado que debe tener el fragmento de programa que representa.
- Esta fase del análisis es más difícil de formalizar que las dos anteriores y se utilizarán para ello las gramáticas atribuidas.

Componentes de un compilador

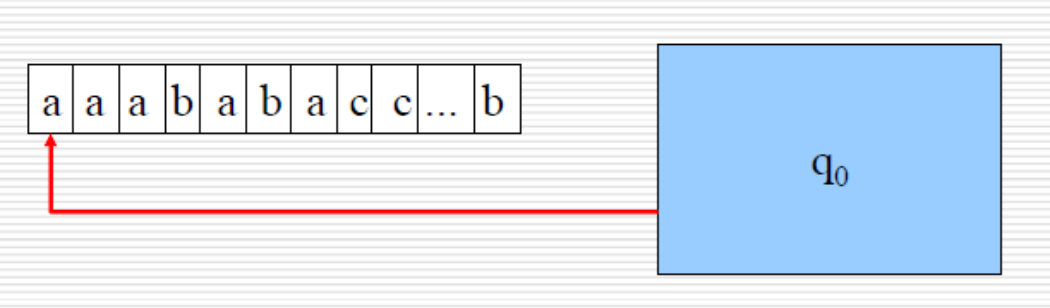
- Otras fases de compilación:
  - Generación de código intermedio.
  - Optimización de código.
  - Generación de código objeto.
  - Tabla de símbolos
  - Control de errores.

Autómatas finitos

- Un autómata finito es un conjunto de estados y un control que se mueve de un estado a otro en respuesta a entradas externas.
- La salida de un autómata finito está limitada a dos valores: *aceptado* y *no aceptado*, que pueden indicar si la cadena que se ha recibido como entrada es o no válida.
- Los autómatas finitos se pueden clasificar 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.
- Utilizamos ambos para reconocer lenguajes regulares (indicar si una palabra pertenece a un lenguaje), sin embargo los **No deterministas** permiten describir más eficientemente determinados problemas.

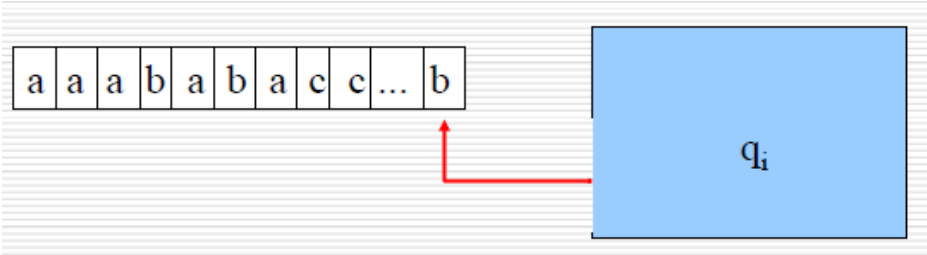
Autómata finito determinista (AFD)

- ¿Cómo procesa las entradas un AFD?
  - La entrada a un AF es un conjunto de símbolos tomados del alfabeto de entrada Σ, no hay límite en tamaño de la cadena.
  - Existe un “puntero” que en cada momento apunta a una posición de la cadena de entrada.
  - El autómata está siempre en un estado de *Q,* inicialmente se encuentra en el estado q0.



Autómata finito determinista (AFD)

- ¿Cómo procesa las entradas un AFD?
  - En cada paso el autómata lee un símbolo de la entrada y según el estado en el que se encuentre, cambia de estado y pasa a leer otro símbolo.



  - Y así sucesivamente hasta que se terminen de leer todos los símbolos de la cadena de entrada.
  - Si en ese momento el AF está en un estado $\mathbf{q_i}$ de **F** (conjunto de estados finales), se dice que **acepta** la cadena, en caso contrario la rechaza (**no acepta**).

AFD – Definición formal

Un AFD se define como una quíntupla:

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

donde:

∑ es el alfabeto de entrada.

Q es el conjunto finito y no vacío de los estados del Autómata.

δ es la función de transición que indica en qué situaciones el Autómata pasa de un estado a otro, se define:

$$\delta : Q \times \Sigma \rightarrow Q$$

$q_0 \subset Q$ es el estado inicial

F ***Q*** es el conjunto de estados finales de aceptación ($F \neq 0$)

Representación de AFD

Tablas de transición

- Es una representación clásica de una función con dos argumentos.
- En las filas se colocarán los estados y en las columnas los símbolos del alfabeto de entrada.
- Cada intersección **fila (estado q) - columna (carácter *a*)** corresponde al estado **δ(q, *a*)**.
- El estado inicial se representa con $\longrightarrow$
- El estado o estados finales con un *****

Ejemplo:

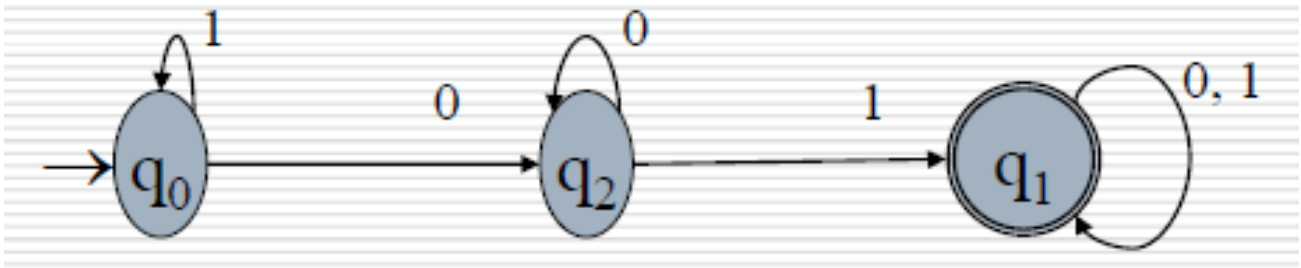
	0	1
$\longrightarrow q_0$	q_2	q_0
$^*q_1$	q_1	q_1
q_2	q_2	q_1

Representación de AFD

Diagramas de transición

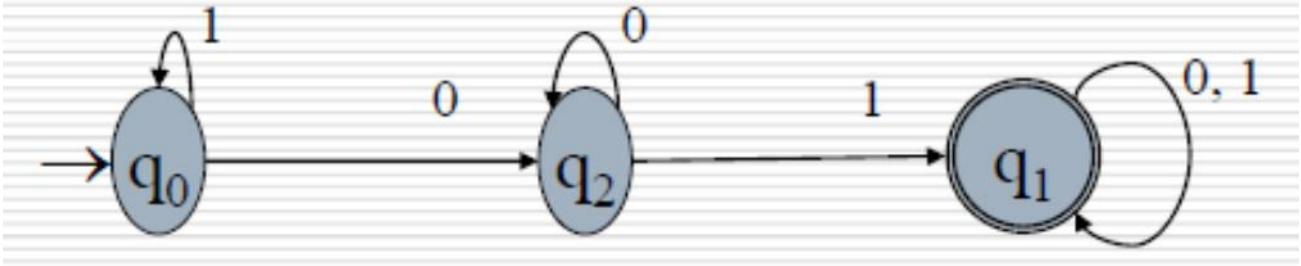
- Es un grafo en el que los vértices representan los distintos estados y los arcos las transiciones entre los estados.
- Cada arco va etiquetado con el símbolo que corresponde a dicha transición.
- El estado inicial se representa con $\longrightarrow$
- Los estados finales se representan con un doble círculo.

Ejemplo:



Representación de AFD

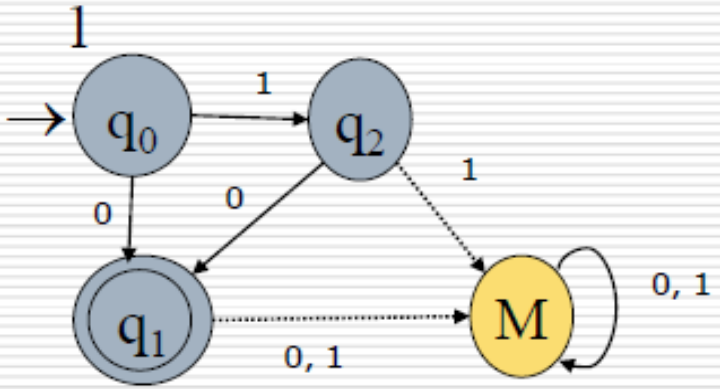
- Determinismo porque:
  - No existen transiciones λ
  - ∀ q∈Q , ∀ a∈Σ ∃ una única **δ(q,a)** :
    - Una sola arista etiquetada con **a** para cada símbolo;
    - Para cada entrada en la tabla un solo estado



Representación de AFD

- En el caso que falten transiciones para algunas entradas, lo resolvemos incluyendo un nuevo estado llamado de **absorción** o **muerto**, al cual llegan todas las transiciones no definidas.
- Ejemplo:

	0	1
→ q_0	q_1	q_2
*q_1	-	-
q_2	q_1	-



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

Ejemplo AFD

Sea el siguiente AFD: ∑ = {a, b} Q = {p, q, r} q0 = p F = {q}
donde δ se define de la siguiente forma:

δ (p, a) = q δ (p, b) = r
δ (q, a) = q δ (q, b) = r
δ (r, a) = r δ (r, b) = r

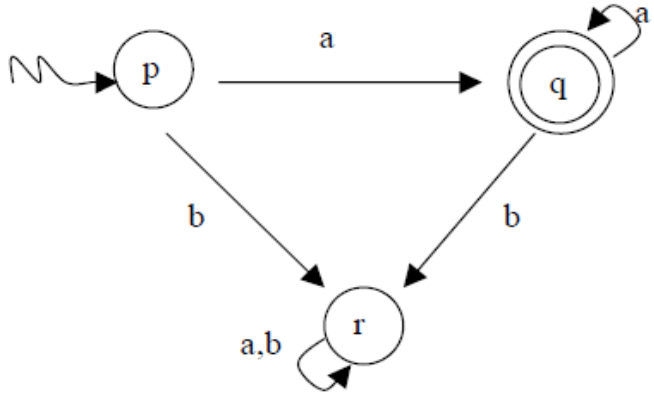


Diagrama de transición

Tabla de transición El AFD se representaría mediante la siguiente tabla que representa los valores de la función de transición.

		a	b
$\rightarrow$	p	q	r
$*$	q	q	r
	r	r	r

¿Qué cadenas del alfabeto ∑ pueden ser identificadas por el AFD?

AFD como reconocedor de un lenguaje

A) Dado el lenguaje L = { x | x en {0, 1} , x comienza con 0}, es decir todas las cadenas binarias que comienzan con 0; el AFD que lo reconoce es el siguiente:

M = (Q , Σ, f, q_0, F), Σ= {0,1}, F = {q_1}, Q={q_0,q_1,q_2}

Función de transición f

$f(q_0,0) = q_1$

$f(q_0,1) = q_2$

$f(q_1,0) = q_1$

$f(q_1,1) = q_1$

$f(q_2,0) = q_2$

$f(q_2,1) = q_2$

Tabla de transiciones

f	0	1
--> q_0	q_1	q_2
*q_1	q_1	q_1
q_2	q_2	q_2

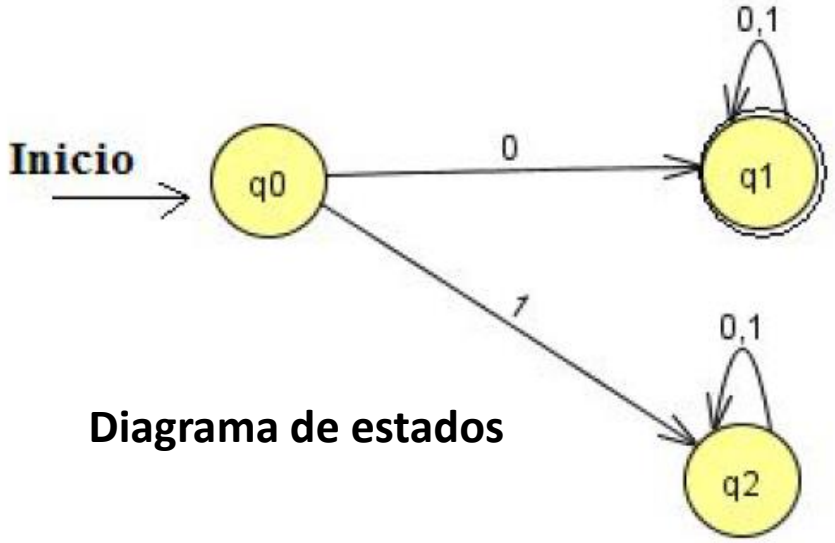


Diagrama de estados

Ejemplo de AFD reconocedor

1) Especificamos formalmente un AFD que acepte únicamente todas las cadenas de ceros y unos que contengan la secuencia **01** en cualquier posición de la cadena. Podemos escribir este lenguaje como sigue:

L = { *w* | *w* tiene la forma ***x*01*y*** para algunas cadenas, *x* e *y* constan sólo de ceros y unos }

O también como:

L = {***x*01*y*** | *x* e *y* son cadenas cualesquiera formadas por 0s y 1s }

Ejemplos de cadenas de este lenguaje: 01, 11010, 100011, etc.

Ejemplos de cadenas que no son de este lenguaje: 0, 111, 1100, etc.

Ejemplo de AFD reconocedor

2) ¿Qué componentes de este AFD (que aceptará el lenguaje L) conocemos?

Recordemos la quíntupla del AFD = $(Q , \Sigma, \delta, q_0, F)$

De aquí ya conocemos:

El alfabeto de entrada: $\sum = \{ 0, 1 \}$

El estado inicial q_0 que pertenece al conjunto de estados ***Q***.

Tenemos entonces que completar los otros estados de Q, así como definir a δ y a F.

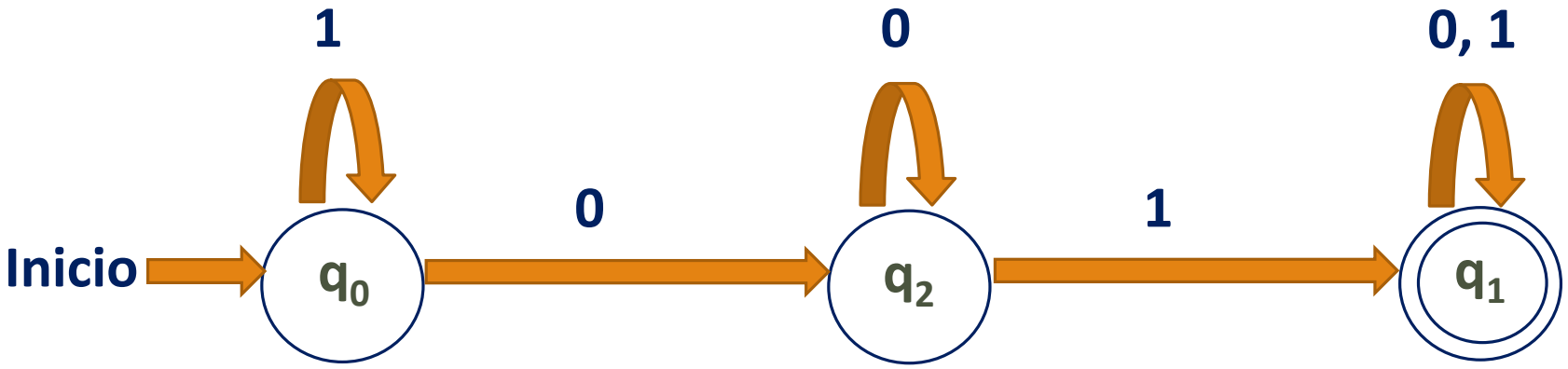
Ejemplo de AFD reconocedor

3) Definiendo los estados de nuestro AFD

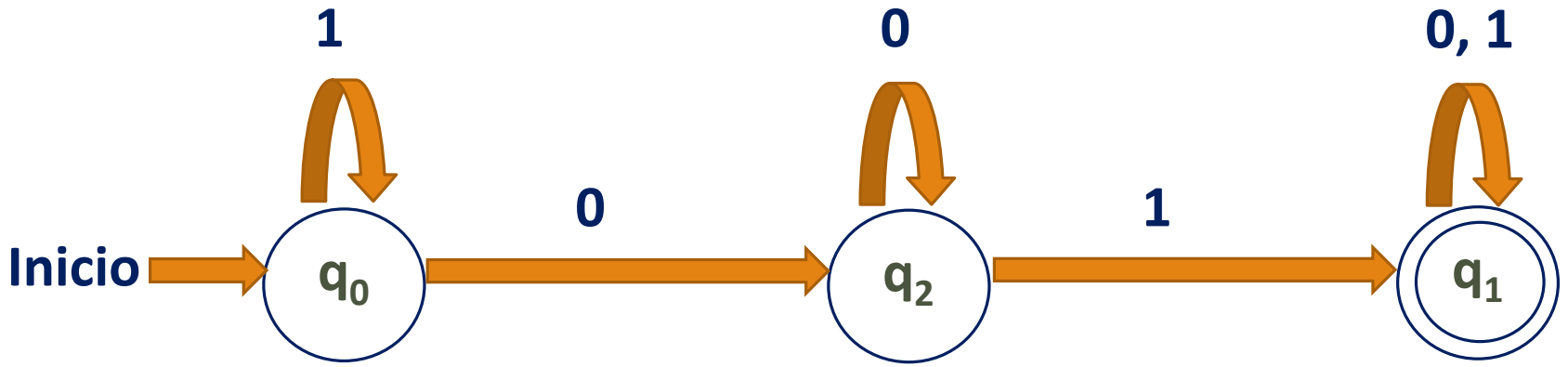
Este AFD tiene que recordar las entradas que ha leído hasta el momento. Para decidir si 01 es una subcadena de la entrada, tiene que recordar:

- a) ¿Ya ha leído una subcadena 01? Si es así, aceptará entonces cualquier secuencia de entradas futuras: es decir, **ya está en el (los) estado(s) de aceptación** (conjunto **F**).
- b) ¿Todavía no ha leído la secuencia 01, pero la entrada más reciente ha sido un 0, de manera que si ahora lee un 1, habrá leído la subcadena 01 y podrá aceptar cualquier cosa que lea de ahí en adelante?
- c) ¿Todavía no ha leído la secuencia 01, pero la última entrada no existe (acaba de iniciarse) o ha sido un 1? En este caso, el AFD no puede aceptar la entrada hasta que no lea un 0 seguido inmediatamente de un 1. **Este será el estado inicial q_0** .

Cada una de estas tres condiciones puede representarse mediante un estado.



c) b) a)



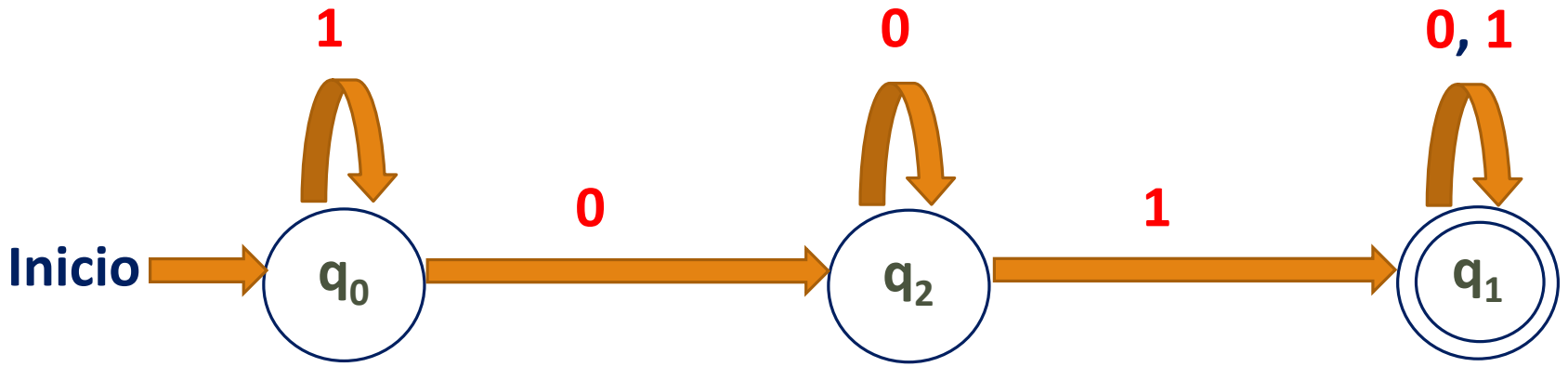
c) b) a)

Función de transición δ

$\delta(q_0, 1) = q_0$
$\delta(q_0, 0) = q_2$

$\delta(q_2, 1) = q_1$
$\delta(q_2, 0) = q_2$

$\delta(q_1, 1) = q_1$
$\delta(q_1, 0) = q_1$



c) b) a)

Función de transición δ

$\delta(q_0, 1) = q_0$
$\delta(q_0, 0) = q_2$

$\delta(q_2, 1) = q_1$
$\delta(q_2, 0) = q_2$

$\delta(q_1, 1) = q_1$
$\delta(q_1, 0) = q_1$

Conjunto Q = { q_0, q_1, q_2 }

Conjunto F = { q_1 }

AFD = ({ q_0, q_1, q_2 }, {0, 1}, δ, q_0, { q_1 })

Función de transición extendida (ejecución de un AFD)

Para la ejecución formal de un AFD se utiliza la definición de la función de transición extendida sobre cadenas:

$$\hat{\delta}: Q \times \Sigma^* \rightarrow Q$$

Caso Base: $\hat{\delta}(q, \varepsilon) = q$

Es decir, si nos encontramos en el estado ***q*** y no leemos ninguna entrada, entonces permanecemos en el mismo estado ***q***.

Recursividad: $\hat{\delta}(q, w) = \delta(\hat{\delta}(q, x), a)$

Suponiendo que ***w*** es una cadena de la forma *xa*; donde ***a*** es el último símbolo de ***w*** y ***x*** es la cadena formada por todos los símbolos excepto el último. Por ejemplo, ***w*** = 1101 se divide en ***x*** = 110 y ***a*** = 1

Ejemplo de función de transición extendida

Deseamos diseñar un AFD que acepte el lenguaje

$$L = \{w \mid w \text{ tiene un número par de ceros y un número par de unos}\}$$

La tarea de los estados de este AFD es la de contar el número de ceros y el número de unos contando en módulo 2. Es decir, el estado se emplea para recordar si el número de ceros es par o impar hasta el momento y también para recordar si el número de unos leídos hasta el momento es par o impar. Existen por tanto cuatro estados que pueden interpretarse de la manera siguiente:

Ejemplo de función de transición extendida

q_0: tanto el número de ceros como el de unos leídos hasta el momento es par.

q_1: el número de ceros leídos hasta el momento es par, pero el de unos es impar.

q_2: el número de unos leídos hasta el momento es par, pero el de ceros es impar.

q_3: tanto el número de ceros como el de unos leídos hasta el momento es impar.

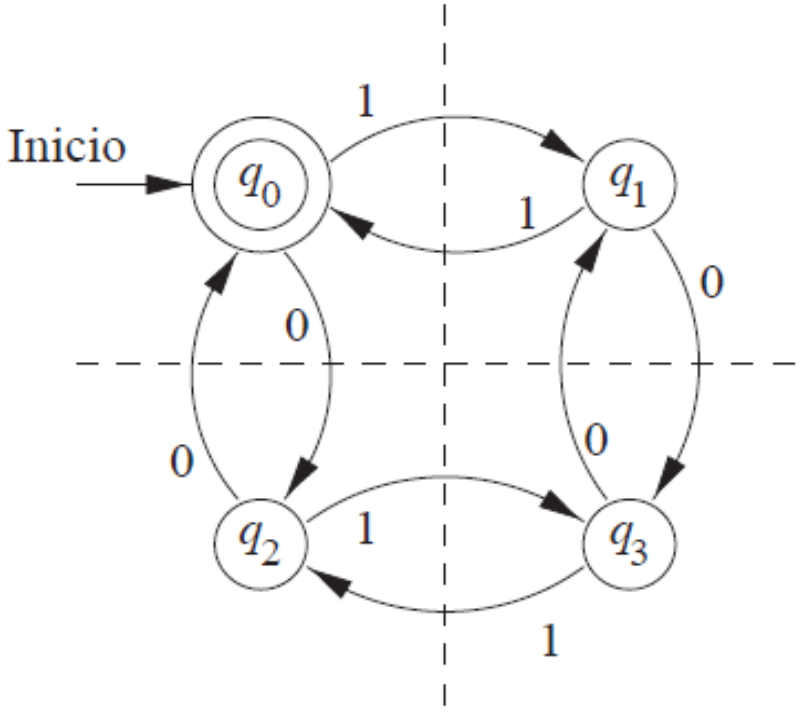
El estado q_0 es tanto el estado incial como el único estado de aceptación. Es el estado inicial porque antes de leer ninguna entrada, la cantidad de ceros y unos leídos hasta el momento es igual a cero y cero es par. Es el único estado de aceptación porque describe de forma exacta la condición para que una secuencia de ceros y unos pertenezca al lenguaje L.

Ahora ya sabemos cómo especificar el AFD para el lenguaje L. Así

$$A = (\{q_0, q_1, q_2, q_3\}, \{0, 1\}, \delta, q_0, \{q_0\})$$

Ejemplo de función de transición extendida

El diagrama de estados y la tabla de transición para nuestro AFD A se muestran a continuación:



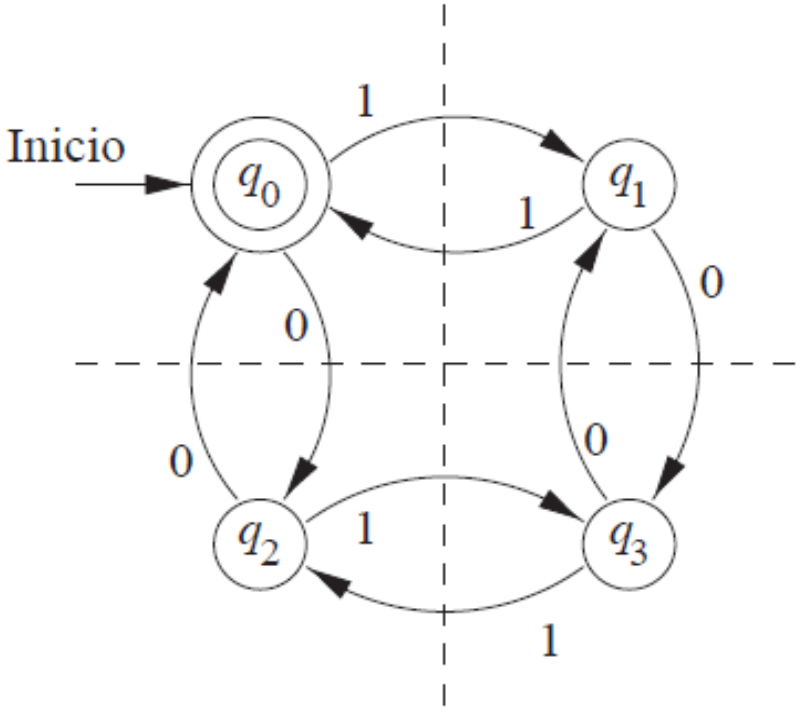
	0	1
$* \rightarrow q_0$	q_2	q_1
q_1	q_3	q_0
q_2	q_0	q_3
q_3	q_1	q_2

Ejemplo de función de transición extendida

Comprobemos ahora nuestra función de transición extendida con la palabra 110101 .

La comprobación supone calcular $\hat{\delta}(q_0, w)$ para cada prefijo w de 110101, comenzando por ε y aumentando progresivamente el tamaño. El resumen de este cálculo es:

- $\hat{\delta}(q_0, \varepsilon) = q_0$.
- $\hat{\delta}(q_0, 1) = \delta(\hat{\delta}(q_0, \varepsilon), 1) = \delta(q_0, 1) = q_1$.
- $\hat{\delta}(q_0, 11) = \delta(\hat{\delta}(q_0, 1), 1) = \delta(q_1, 1) = q_0$.
- $\hat{\delta}(q_0, 110) = \delta(\hat{\delta}(q_0, 11), 0) = \delta(q_0, 0) = q_2$.
- $\hat{\delta}(q_0, 1101) = \delta(\hat{\delta}(q_0, 110), 1) = \delta(q_2, 1) = q_3$.
- $\hat{\delta}(q_0, 11010) = \delta(\hat{\delta}(q_0, 1101), 0) = \delta(q_3, 0) = q_1$.
- $\hat{\delta}(q_0, 110101) = \delta(\hat{\delta}(q_0, 11010), 1) = \delta(q_1, 1) = q_0$.



Lenguaje aceptado por un AFD

Sea $M = (Q, \Sigma, f, q0, F)$ un AFD, el lenguaje **L** aceptado por **M** esta definido por:

$$L(M) = \{ w \mid w \in \Sigma^* \ y \ f(q_0, w) \in F \}$$

Nota: el lenguaje de **M**, es un conjunto de cadenas **w** que llevan al autómata **M** desde q_0 a un estado de aceptación. Si **L** es un **L(M)** para algún AFD **M**, entonces decimos que **L** es un "Lenguaje Regular".

Nota importante: Para poder verificar si la cadena **w** está (es aceptada) o no está(no es aceptada) en **L** se debe procesar por completo.

Ejercicios con AFD

Para Σ = {0,1} construya un AFD que acepte los siguientes lenguajes:

a) El conjunto de cadenas que tienen el mismo número de unos y de ceros: 0101, 000111, …

b) El conjunto de cadenas en las que el numero de ceros es divisible por 3.

d) El conjunto de cadenas que terminan con 01.

f) El conjunto de cadenas que comienzan con 10

e) El conjunto de cadenas que tienen un número impar de unos y de ceros.

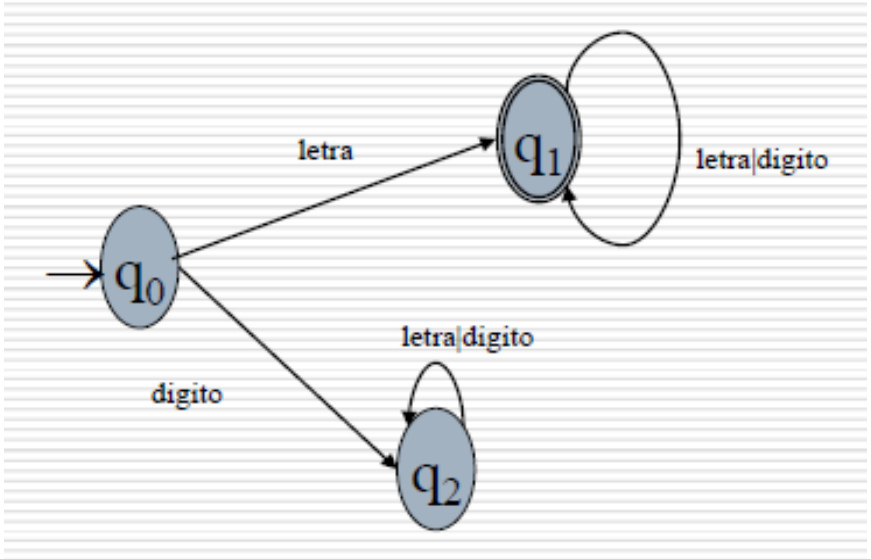
Simulación algorítmica de un AFD

- Entrada: cadena de entrada ***x*** que termina con un carácter fin de cadena o fin de archivo (FDC).
- Salida: La respuesta ACEPTADA si el autómata reconoce; NO ACEPTADA en caso contrario
- Método: aplicar δ al estado al cual hay una transición desde el estado ***q*** a un carácter de entrada ***c***

```
Función reconocer()
  q=q0
  c= leer_carácter()
  mientras c != FDC
      q=δ(q,c)
      c= leer_carácter()
  fmientras
  Si q ε F entonces
        Devolver(ACEPTADA)
  Sino
        Devolver(NO ACEPTADA)
  fsi
```

Simulación algorítmica de un AFD

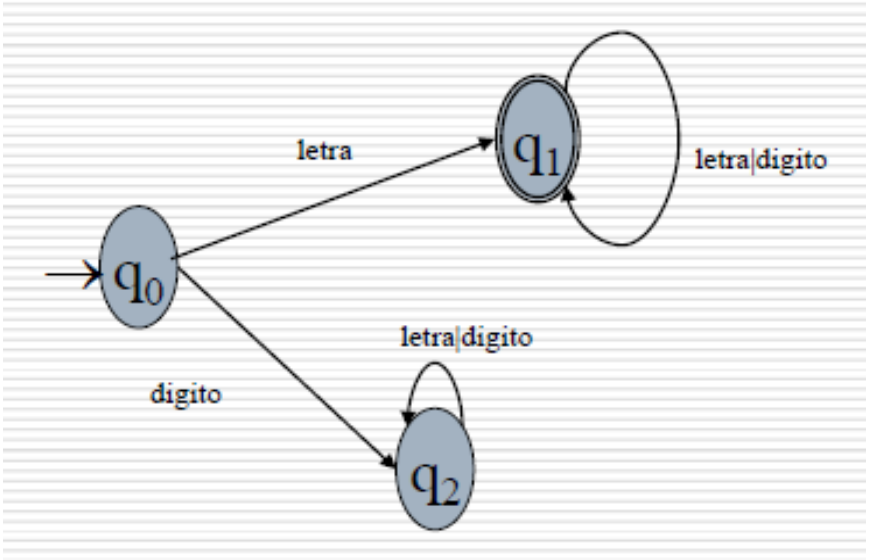
Ejemplo: AFD que reconozca el nombre de variables empezando por letra y seguida de letras o dígitos.



	letra	digito
→ q_0	q_1	q_2
* q_1	q_1	q_1
q_2	q_2	q_2

Simulación algorítmica de un AFD

```
Función reconocer_id() //Por el diagrama
estado=0
s= leer_simbolo()
Mientras s != FDC
    Caso estado sea
        0: Si s=letra entonces estado=1
           sino
              Si s=digito entonces estado=2
              sino
                  Error() //salir a la rutina de error léxico
              fsi
           fsi
        1: Si s=letra or digito entonces estado=1
           sino
                  Error() //salir a la rutina de error léxico
           fsi
        2: Si s=letra or digito entonces estado=2
           sino
                  Error() //salir a la rutina de error léxico
           fsi
     fcaso
s= leer_simbolo()
fmientras
Si estado=1 entonces
      Devolver(SI)
sino
     Devolver(NO)
fsi
```



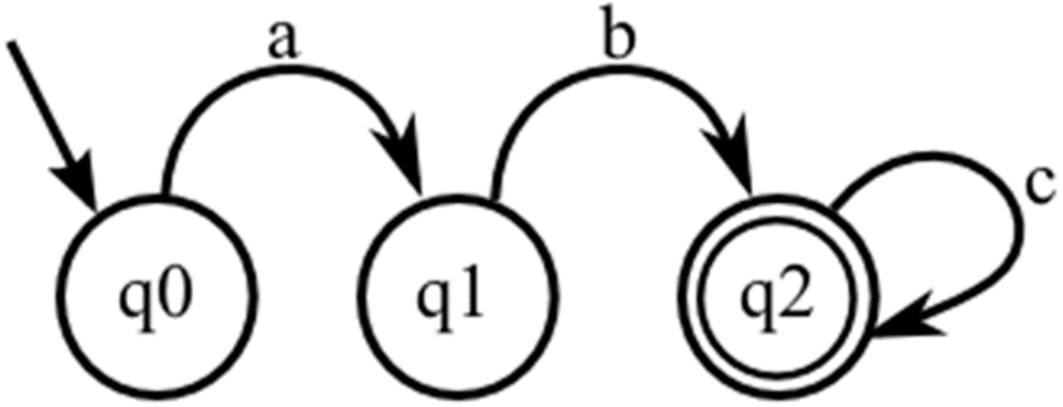
Simulación algorítmica de un AFD

```
Función reconocer_id() //por la tabla
  estado=0
  s= leer_simbolo()
  Mientras s != FDC
     Caso s sea
           letra: entrada=letra
           digito: entrada=digito
           otro: Error() //salir a la rutina de error léxico
     fcaso
     estado= f(estado,entrada)
     s= leer_simbolo()
  fmientras
  Si estado=1 entonces
    Devolver(SI)
  sino
    Devolver(NO)
  fsi
```

	letra	digito
→ q_0	q_1	q_2
*q_1	q_1	q_1
q_2	q_2	q_2

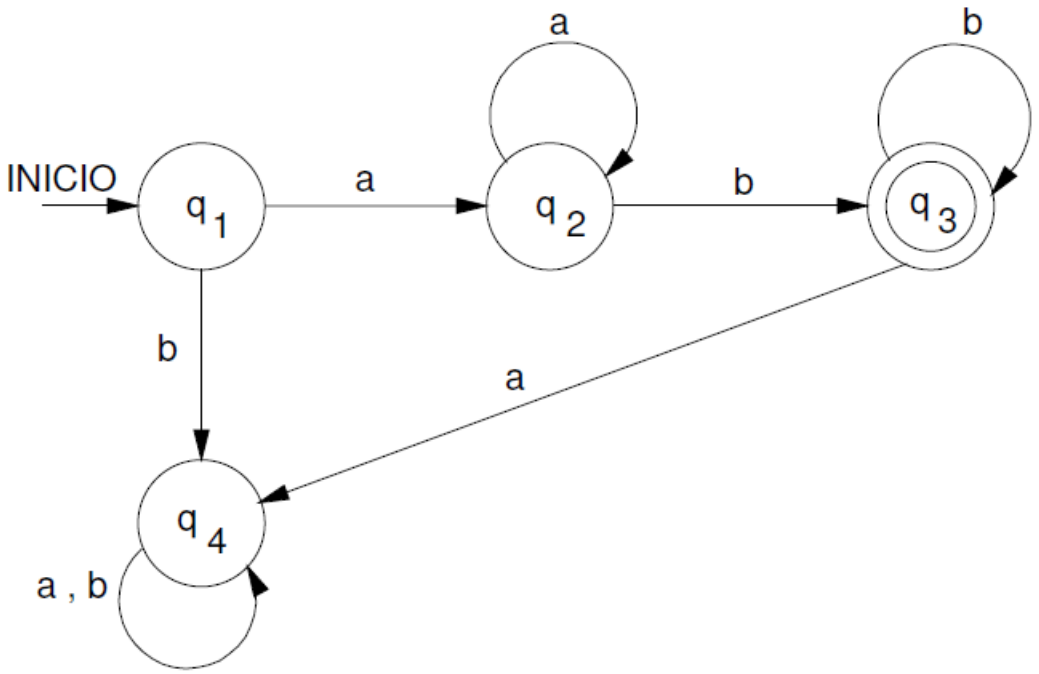
AFD y Expresiones regulares

Un autómata finito es una maquina de estados capaz de reconocer un **lenguaje regular**, si el autómata es capaz de alcanzar en su configuración final un estado del conjunto F.



abc*

Ejemplo 1



L(A)= {ab, aab,…,abbb, …, aabb,…}
L(A) = { $a^n b^m$ | n >= 1, m >= 1 }

La expresión regular que denota el lenguaje es a^+b^+ o también aa^*bb^*.

Ejemplo 2

Los identificadores de C son: cadenas de letras, dígitos y guiones bajos.

*letra_* -> [A-Za- z_]

dígito -> [0-9]

id -> *letra_* (*letra_* | *dígito*)*

El autómata finito es el siguiente:

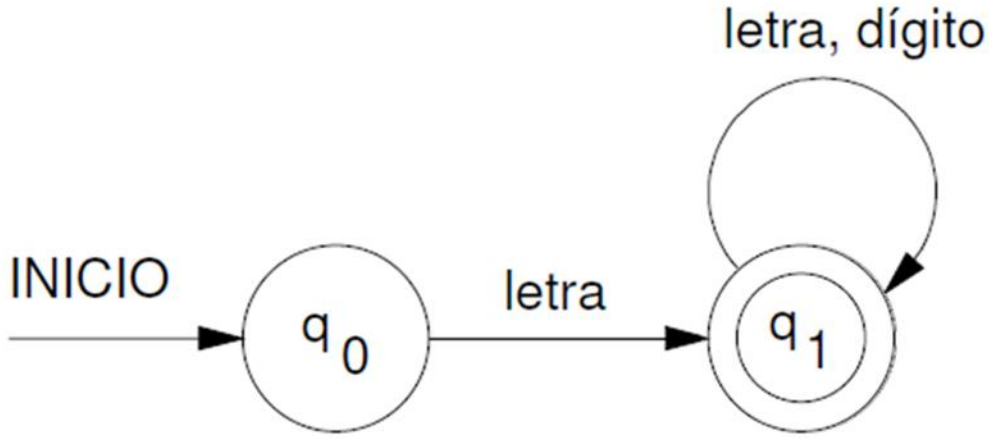
$AFD = \{ \sum=\{a,b,...,z,0,1,...9,\$\}, Q=\{q_0, q_1, q_2\}, f, q_0, F=\{q_2\}\}$

$ = fin de cadena

Ejemplo 2 - continuación

Función *f*:

f	<letra>	<dígito>	$
q_0	q_1	-	-
q_1	q_1	q_1	q_2
q_2	-	-	-



Diseña autómatas finitos capaces de reconocer los siguientes lenguajes definidos por la expresiones regulares:

1. a^+b^*
2. $ab(cd)^+e$
3. $(a^+b^+)cd$
4. $abz?b^+$
5. $(cg)^*gato^+ \mid cd$