UNIVERSIDAD
NACIONAL
DE COLOMBIA

Conceptos previos: gramáticas y autómatas

Felipe Restrepo Calle
ferestrepoca@unal.edu.co

Departamento de Ingeniería de Sistemas e Industrial
Facultad de Ingeniería
Universidad Nacional de Colombia
Sede Bogotá

1. Gramáticas

2. Autómatas

Concepto de gramática formal

¿Cómo se pueden definir las palabras/frases que pertenecen a un determinado lenguaje?

- **Enumerando**
  - ➡ ¿algunos lenguajes son infinitos?
- **Descripción informal**:
  - ➡ a veces complicado y demasiado impreciso

Concepto de gramática formal

Necesitamos un **formalismo** para definir los lenguajes (las frases que pertenecen a un lenguaje).

➡ **Gramáticas formales**

- Una gramática describe **de forma inequívoca** la estructura de las frases y palabras de un lenguaje a través del uso de reglas definidas.
- Proporcionan un **mecanismo para generar todas las palabras** que pertenecen a un determinado lenguaje (también se llaman *gramáticas generadoras*).

Concepto de gramática formal

Ejemplo: español

La sintaxis del español se define mediante reglas:

1. Una oración consta de sujeto y predicado y termina con un punto.
2. Un sujeto es una frase nominal.
3. Una frase nominal es un grupo nominal seguido de un calificativo (que puede faltar)
4. ...

Concepto de gramática formal

Ejemplo: español

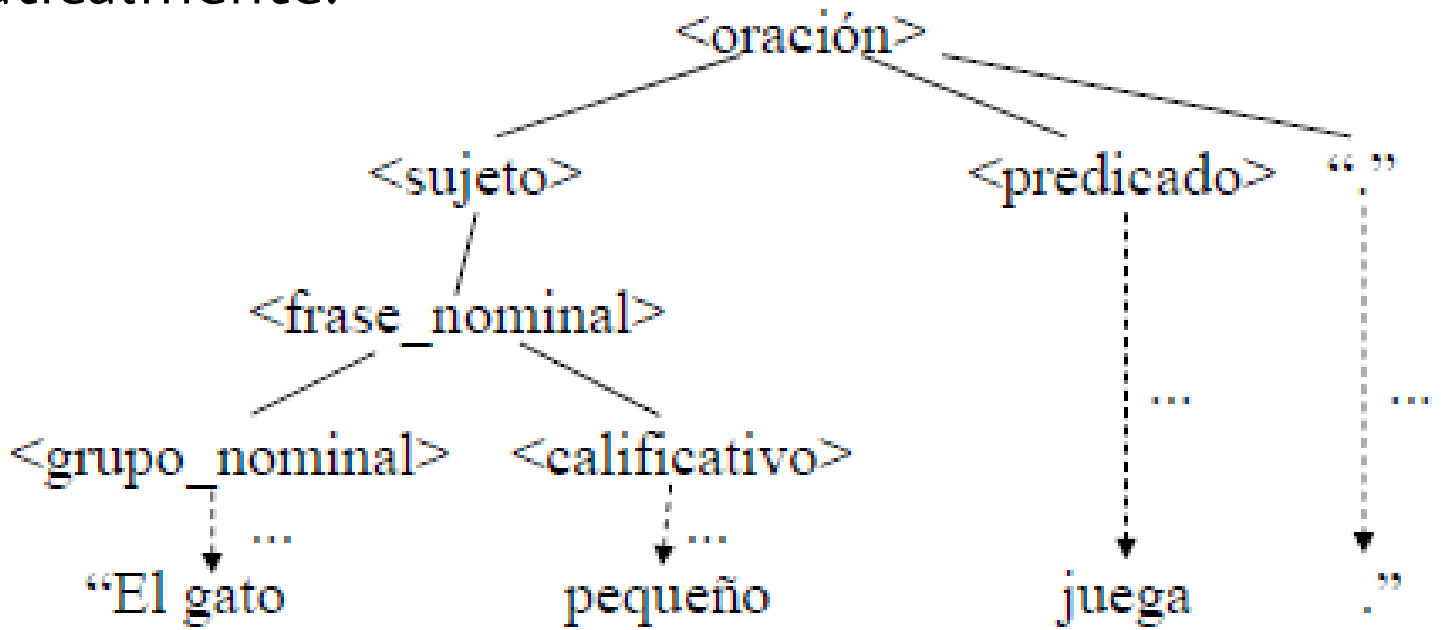
Podemos usar *producciones* para representar estas reglas:

<oración>::=<sujeto> <predicado>.
<sujeto>::=<frase_nominal>
<frase_nominal>::=<grupo_nominal> <calificativo>
<frase_nominal>::=<grupo_nominal>
...

Concepto de gramática formal

Ejemplo: español

Podemos analizar si una frase en español es correcta gramaticalmente:



Gramática BNF

La notación formal usada para describir la gramática de un lenguaje es la **Forma Backus-Naur (BNF).**

$$G = (V_N ; V_T ; S ; P)$$

- V_N es el conjunto de símbolos **no terminales** (variables)
- V_T es el conjunto de símbolos **terminales**
- S es el **símbolo inicial** de la gramática
- P es el conjunto de **reglas** de la gramática

Gramática BNF

Ejemplo:

$$G = (V_N ; V_T ; S ; P)$$

- $V_N = \{S\}$ no terminales
- $V_T = \{0, 1\}$ terminales
- S símbolo inicial
- $P = \{$ reglas

$$S \rightarrow 0\ S\ 0 \mid 1\ S\ 1 \mid 0 \mid 1$$

$\}$

¡Números capicúas con 0 y 1!

0
101
00100
100101001
...

Gramática BNF

Ejemplo:

$$G_2 = (V_N ; V_T ; S ; P)$$

- $V_N = \{S, A\}$ no terminales
- $V_T = \{a, b\}$ terminales
- S símbolo inicial
- $P = \{$ reglas

$$S \longrightarrow Ab$$

$$A \longrightarrow aAb \mid \varepsilon$$

$\}$

$\Rightarrow L(G_2) = \{a^n b^{n+1} \mid n \geq 0\}$

b

abb

aabbb

aaaaabbbbbb

...

Gramática BNF

Ejemplo:

$$G_3 = (V_N ; V_T ; S ; P)$$

- $V_N = \{S, A\}$ no terminales
- $V_T = \{a, b\}$ terminales
- S símbolo inicial
- $P = \{$ reglas

$S \longrightarrow Ab$

$A \longrightarrow aAb \mid ab$

$\}$

abb

aabbb

aaaaabbbbbb

...

$\Rightarrow L(G_3) = \{a^n b^{n+1} \mid n > 0\}$

Gramática BNF

Ejemplo:

$$G_4 = (V_N ; V_T ; S ; P)$$

- $V_N = \{S, A\}$ no terminales
- $V_T = \{a, b\}$ terminales
- S símbolo inicial
- $P = \{$ reglas

  $S \rightarrow Ab$

  $A \rightarrow ab \mid a$

  $\}$

abb

ab

$\Rightarrow L(G_4) = \{abb \mid ab\}$

Gramática BNF

Ejemplo:

$$G_5 = (V_N ; V_T ; S ; P)$$

- $V_N = \{S, A\}$ no terminales
- $V_T = \{a, b\}$ terminales
- S símbolo inicial
- $P = \{$ reglas

$S \longrightarrow Ab$

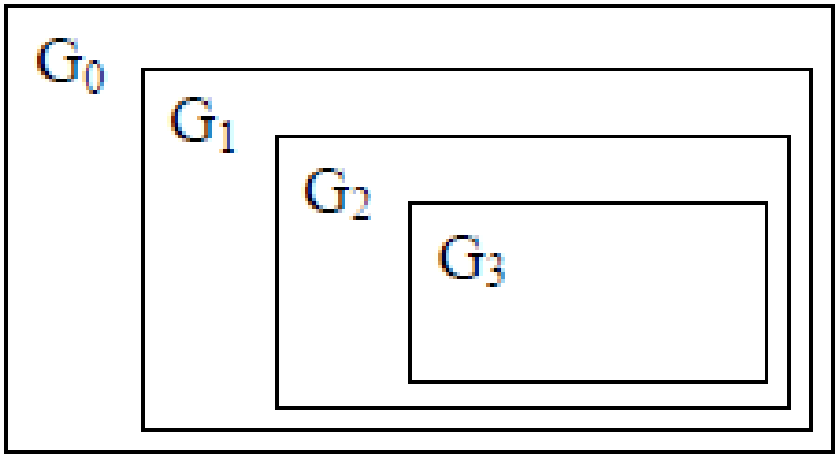
$A \longrightarrow Aab \mid a$

$\}$

ab
aabb
aababb
aababababb
...

$\Rightarrow L(G_5) = \{a(ab)^n b \mid n \geq 0\}$

Jerarquía de gramáticas de Chomsky (1956)



Todas las gramáticas posibles:

$$G_3 \subset G_2 \subset G_1 \subset G_0$$

G_3: Gramáticas regulares
G_2: Gramáticas independientes del contexto
G_1: Gramáticas sensibles al contexto
G_0: Gramáticas recursivamente enumerables o sin restricciones

Jerarquía de gramáticas

Según la forma de las reglas, las gramáticas son:

- **Regulares (tipo 3)**:
  - En la parte izquierda sólo hay un *no terminal*
  - En la parte derecha puede haber:

    no terminal ⟶ **terminal**

    no terminal ⟶ **terminal** *no terminal*

    no terminal ⟶ **ε** (cadena vacía)
  - Generan los lenguajes regulares
  - Autómata finito

Jerarquía de gramáticas

Según la forma de las reglas, las gramáticas son:

- **Independientes del contexto (GIC - tipo 2)**:
  - En la parte izquierda sólo hay un *no terminal*
  - En la derecha no hay restricciones
  - La mayor parte de los lenguajes de programación están generados por este tipo de gramáticas
  - Autómata de pila

Jerarquía de gramáticas

Según la forma de las reglas, las gramáticas son:

- **Dependientes del contexto (tipo 1)**:
  - En la izquierda puede haber *terminales* y *no terminales*, pero al menos debe haber un *no terminal*
  - La longitud de la parte derecha debe ser mayor o igual que la de la izquierda
  - Autómata linealmente acotado
- **No restringidas (tipo 0)**
  - Máquina de Turing

Gramáticas independientes del contexto – GIC (tipo 2)

Ejemplo:

$$S \rightarrow 0B \mid 1A$$
$$A \rightarrow 0 \mid 0S \mid 1AA$$
$$B \rightarrow 1 \mid 1S \mid 0BB$$

Representa el lenguaje formado por las cadenas que contienen igual número de ceros que de unos.

Gramáticas independientes del contexto – GIC (tipo 2)

Ejemplo:

$$S \rightarrow 0S1 \mid 01$$

¿Qué lenguaje genera?

Cadenas del tipo 0^n1^n, con $n > 0$

Gramáticas independientes del contexto – GIC (tipo 2)

Lenguajes independientes del contexto, LIC: son los lenguajes generados por GIC.

Si L es regular → L es LIC (al revés no es cierto)

Ejemplo:

$$S \rightarrow aSb \mid aSa \mid bSa \mid bSb \mid a \mid b \mid \varepsilon$$

No es una gramática regular, pero L(G) es regular

Ejercicios de GIC para conjuntos sobre $\Sigma = \{0, 1\}$

1. Números binarios capicúa (cadena binaria palíndroma)

2. Cadenas que no son capicúa (o palíndromas)

Ejercicios de GIC para conjuntos sobre $\Sigma = \{0, 1\}$

3. Cadenas del tipo $0^{n-1}1^n$, con $n > 0$

4. Cadenas del tipo 0^n10^{2n}, con $n \geq 0$

1. Gramáticas

2. Autómatas

Autómatas Finitos

Los **autómatas finitos** sirven para describir **lenguajes regulares**.

Son **máquinas de estados** que tienen acceso a una secuencia de símbolos de entrada (mediante una cabeza lectora).

Según su forma de funcionamiento se dividen en:

- **Deterministas (AFD):** dada una entrada, existe un único estado al que se puede llegar.
- **No deterministas (AFND o indeterministas - AFI):** dada una entrada, puede estar en más de un estado a la vez, pueden tener más de un estado inicial, pueden tener transiciones vacías (sin consumir entrada).

Autómatas Finitos Deterministas (AFD)

- Se encuentra **en cada momento en un estado determinado y puede transitar a otro estado**. Para ello:
  - Se lee la cinta y se avanza la cabeza lectora.
  - En función del símbolo leído y del estado actual, el autómata transita a otro estado.

- Un AFD **detiene el procesamiento cuando no le quedan más símbolos** en la entrada.

Definición de AFD

Un AFD es una quíntupla:

$$A=(Q, \sum, f, q_0, F)$$

donde:

- Q: es el conjunto de estados.
- $\sum$: es el alfabeto de entrada.
- $f: Q \times \sum \rightarrow Q$ es la función (total) de transición.
- $q_0 \in Q$ es el estado inicial.
- $F \subseteq Q$ es el conjunto de estados finales.

Representación de AFD

Existen 3 tipos de representación:

a) Diagrama (grafo) de transiciones

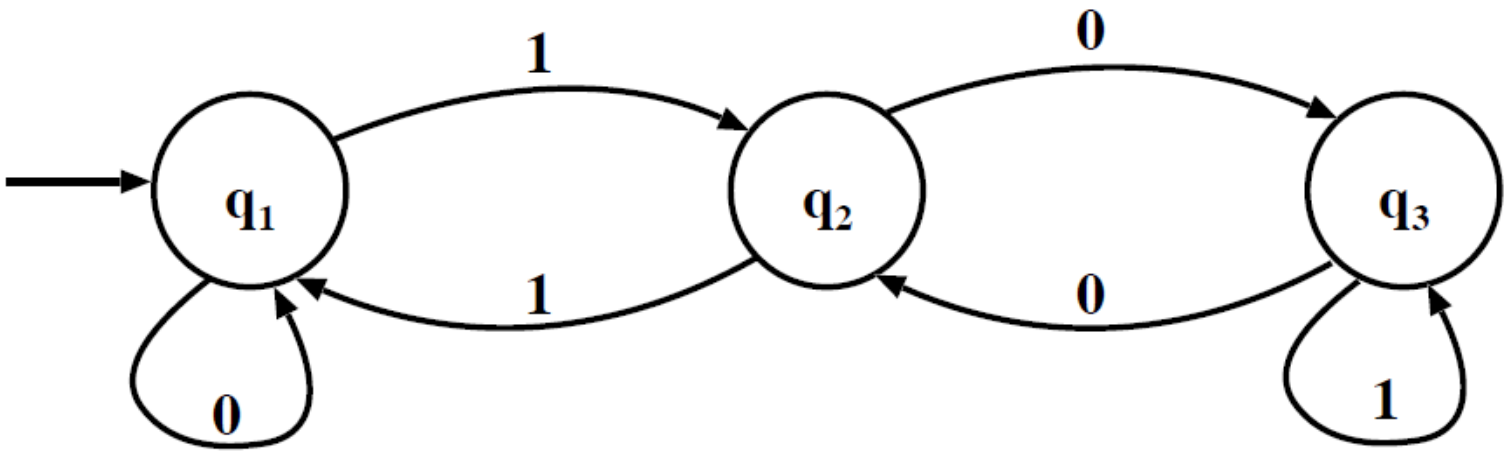
b) Tabla de transiciones

c) Función de transición

Representación de AFD: Diagrama de Transiciones

Grafo dirigido etiquetado que, dado un alfabeto ∑, consta de:

- un conjunto de nodos Q (estados del autómata)
- un subconjunto de arcos $E_a \subset Q \times Q$, $\forall a \in \sum$ (transiciones de estado del autómata)
- existe una flecha dirigida al **estado inicial q_1**



Representación de AFD: Tabla de Transiciones

Contiene los cambios de estado que se producen al procesar un símbolo de la cadena de entrada:

		0	1
$\longrightarrow$	q_1	q_1	q_2
	q_2	q_3	q_1
	q_3	q_2	q_3

Representación de AFD: Función de Transición

La tabla anterior es equivalente a un conjunto de funciones, una por cada columna, que asocia a cada estado de partida otro de llegada:

$$f: Q \times \sum \rightarrow Q$$

En el ejemplo:

$f(q_1, 0) = q_1$ $\quad$ $f(q_1, 1) = q_2$

$f(q_2, 0) = q_3$ $\quad$ $f(q_2, 1) = q_1$

$f(q_3, 0) = q_2$ $\quad$ $f(q_3, 1) = q_3$

Representación de AFD: Función de Transición

Un AFD realiza transiciones entre estados de forma que un estado de llegada es a su vez estado de partida para la siguiente transición.

Ejemplo:

$$f(q_1, 101) = (q_2, 01) = (q_3, 1) = q_3$$

Determinista: ¡sólo hay una opción en cada momento!

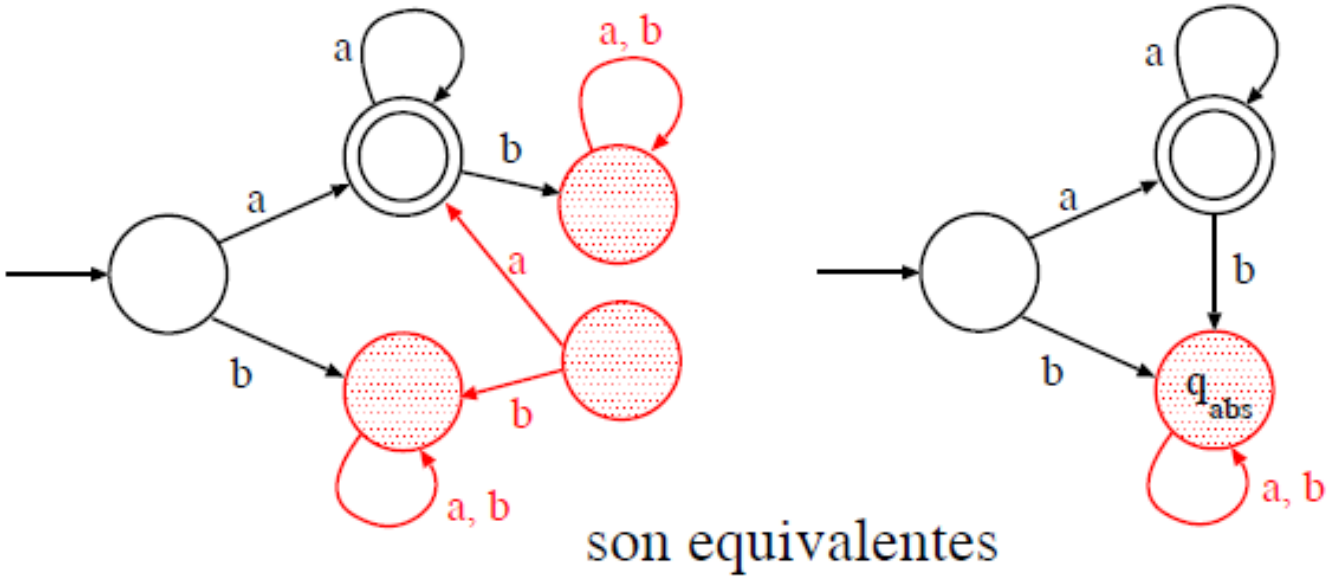
Grafos sencillos que NO corresponden a AFDs

- Carecen de transiciones para determinados estados y símbolos del alfabeto
- Algunas transiciones no están etiquetadas
- No tiene estado inicial (un AFD si puede carecer de estados finales)

Minimización de AFD: nodo útil

Nodo útil: es aquel que es utilizado en el proceso de reconocimiento de alguna cadena.

Los nodos no útiles o **inútiles** pueden ser eliminados sin que L(M) cambie.

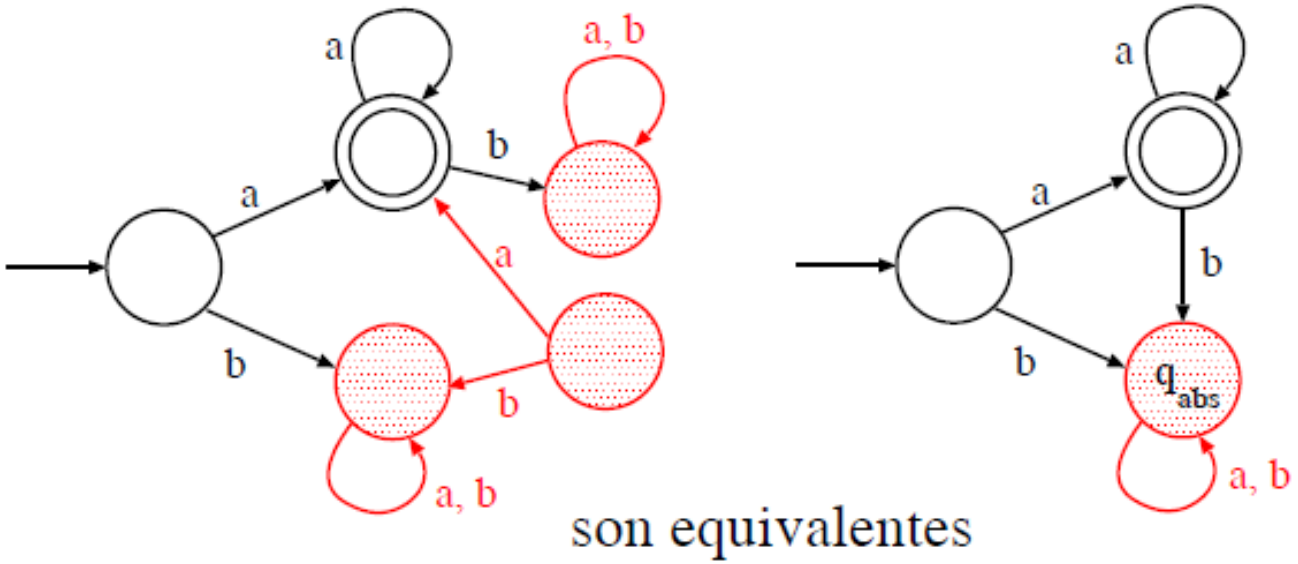


son equivalentes

Minimización de AFD

La **minimización de un autómata** pasa por la minimización del conjunto de estados. Hay 2 formas:

- **Eliminando estados inaccesibles**
- **Combinando estados equivalentes**



Uso de los AFD

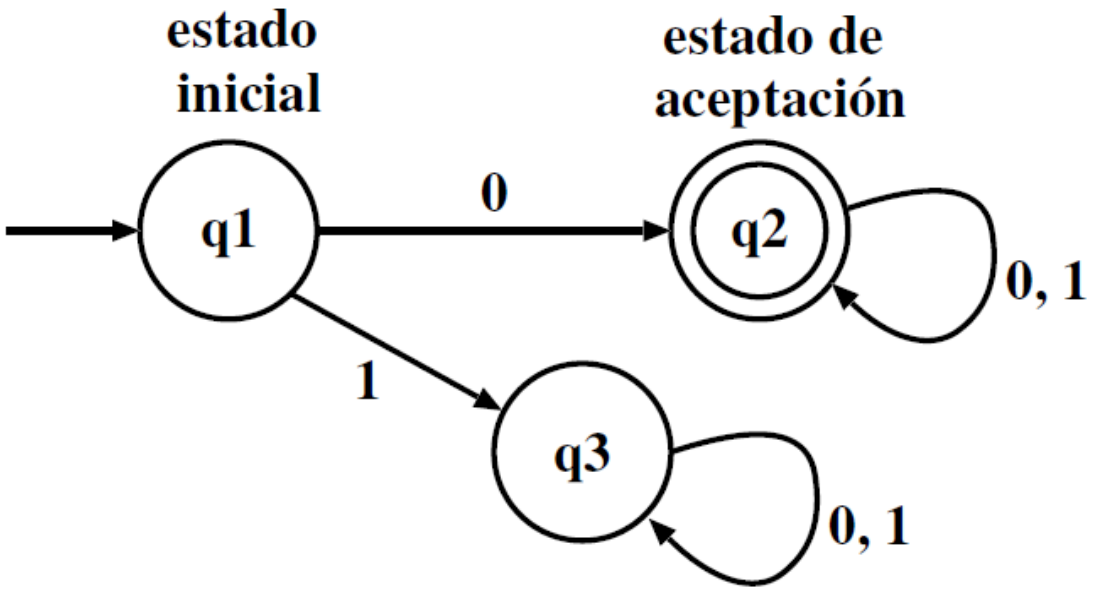
Según la tarea, se pueden usar como:

- **Clasificadores binarios** (clasificación de cadenas en dos clases)
- **Traductores** (clasificación en multitud de grupos)

Uso de los AFD: Clasificador

Un AFD puede servir para discriminar palabras, observando el estado final en el que se encuentra.

Ejemplo:



Sólo acepta cadenas de ceros y unos que empiecen con 0

Uso de los AFD: Clasificador

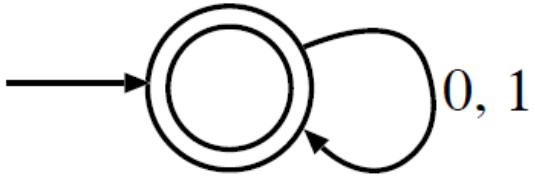
El **lenguaje aceptado por un AFD** es:

$$L(M) = \{w \in \sum^* : f(q_1, w) \in F\}$$

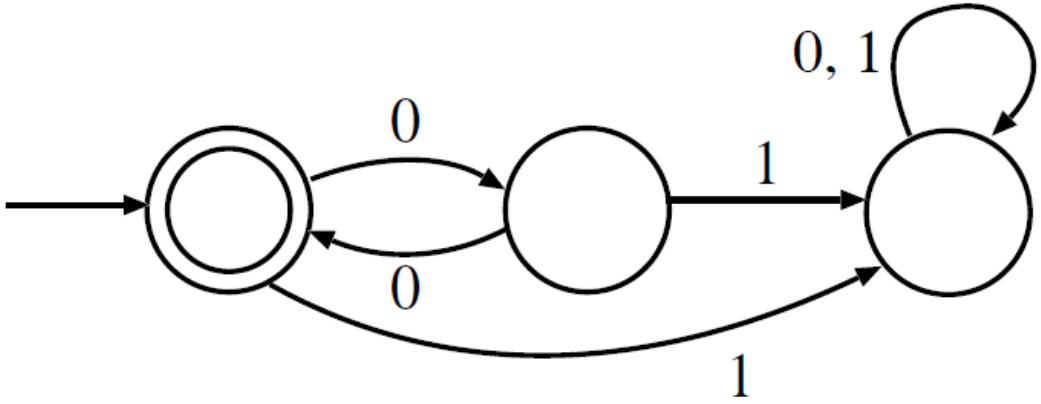
Todo lenguaje reconocible mediante un AFD se llama **lenguaje regular**.

Uso de los AFD: Clasificador

Si $\sum$ = {0, 1}, ¿cuál es el lenguaje aceptado por los siguientes AFD?



Cadenas binarias (incluye la cadena vacía)



Cadenas únicamente con un número par de ceros (incluye la cadena vacía)

Uso de los AFD: Clasificador

Aplicaciones:

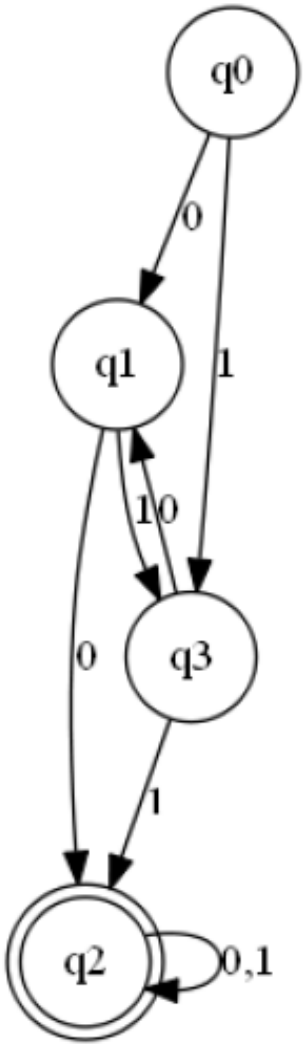
- Análisis léxico en programas (búsqueda de palabras correctas)
- Búsqueda de patrones en textos

Ejemplo:

A partir de $\sum$ = { 0, 1, . . ., 9, E, +, -, . }, sólo algunas palabras serán literales numéricos válidos en lenguaje C.

Uso de los AFD: Clasificador

Ejemplo: dibujar un AFD que reconozca cadenas que, como mínimo, contengan dos ceros consecutivos o dos unos consecutivos para **Σ**= {0,1}



Uso de los AFD: Traductor

Son autómatas capaces de **generar una cadena de salida** a partir de una cadena de entrada.

Un traductor secuencial es un AFD en el que:

- Se ha incorporado:
  - Alfabeto de salida (**Λ** – lambda mayúscula)
  - Función de salida (**λ** – lambda minúscula)
- No se define el subconjunto de estados de aceptación F

$$M=(Q, \sum, \Lambda, f, \lambda, q_0)$$

Uso de los AFD: Traductor

Dependiendo de la forma de **λ**:

- **Máquinas de Moore (λ : Q → Λ)**
- **Máquinas de Mealy (λ : Q × ∑ → Λ)**

Uso de los AFD: Traductor

Dada $w_{in} = a_1a_2a_3 \ldots$ que produce transiciones en:

$$M: q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} q_3 \ldots$$

- En las **máquinas de Moore**, $w_{out} = \boldsymbol{\lambda}(q_2)\ \boldsymbol{\lambda}(q_3) \ldots$

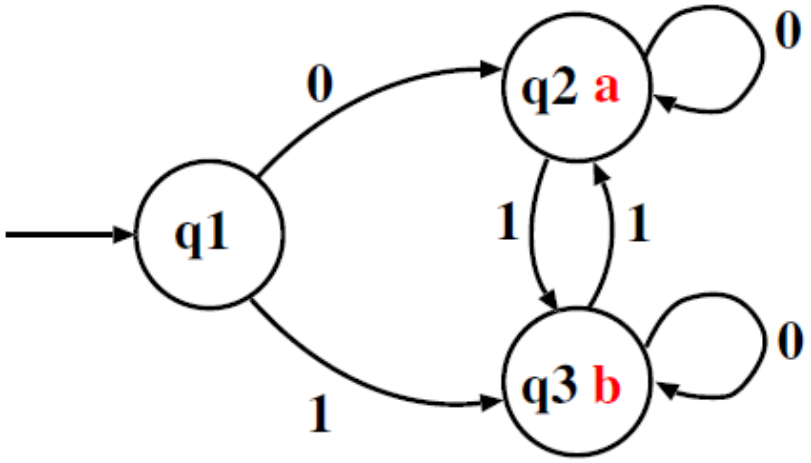
  El símbolo de salida es función del estado en que se encuentra M. Después que M cambie de estado, se detecta un nuevo símbolo.

- En las **máquinas de Mealy**, $w_{out} = \boldsymbol{\lambda}_{a1}(q_1)\ \boldsymbol{\lambda}_{a2}(q_2)\ \boldsymbol{\lambda}_{a3}(q_3)\ldots$

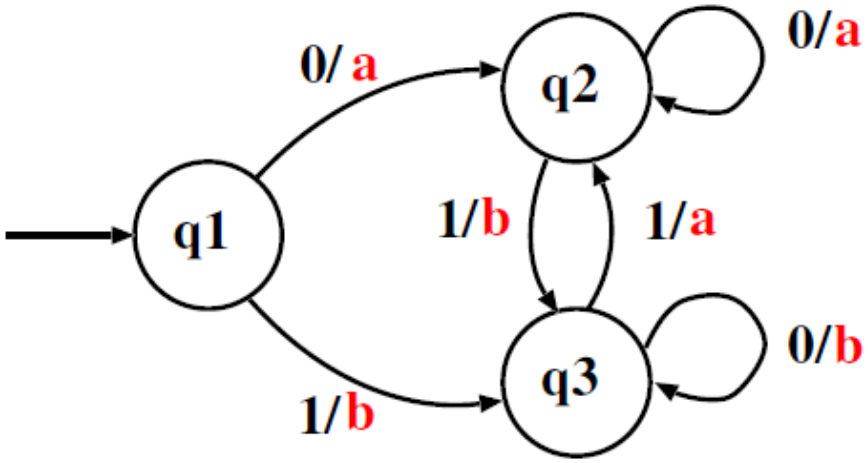
  El símbolo de salida es función del estado y de la transición y varía simultáneamente con M cuando este cambia de estado.

Uso de los AFD: Traductor

Ejemplo: máquina de Moore y de Mealy que traduzca cadenas de $\sum^*$ = {0, 1} a cadenas de Λ^* = {a, b} de manera que se genere una "b" si el último número de unos leídos es impar, y "a" si dicho número es par.



máquina de Moore



máquina de Mealy

Uso de los AFD: Traductor

Ejercicio: construir una máquina de Moore capaz de ir calculando la suma parcial de los números {0, 1, 2, 3} que va recibiendo, y producir como salida el módulo 5 de dicha suma parcial.

Por ejemplo, si se recibe la cadena 3 2 1 3 0, las sumas parciales serán: 3, 5, 6, 9 y 9, y la salida que deberá producir la máquina será 3 0 1 4 4.

Uso de los AFD: Traductor

Solución: Moore

Uso de los AFD: Traductor

Ejercicio: construir una máquina de Mealy que, para una cadena binaria de entrada, produzca como salida:

a) una *a* si ninguno de los dos últimos símbolos leídos es un 1;

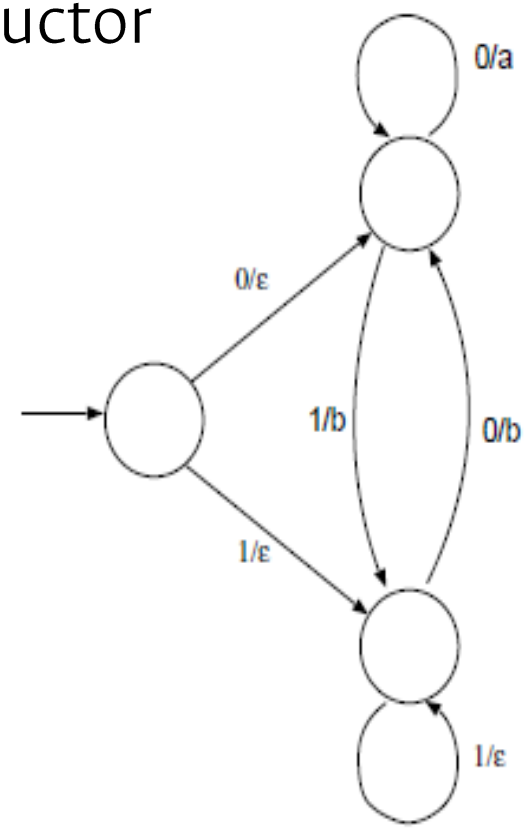
b) una *b* si exclusivamente uno de los dos últimos símbolos leídos es un 1;

c) la cadena vacía en otro caso.

Ej: la salida correspondiente a la entrada 01101001 sería bbbbab (=εbεbbbab).

Uso de los AFD: Traductor

Solución: Mealy



Máquina de Mealy

Autómatas Finitos No Deterministas (AFND)

Son una generalización de los AFDs donde se permite:

- que de un nodo parta más de una transición con el mismo símbolo
- que exista más de un estado inicial
- que existan ε–transiciones (transiciones vacías)

Autómatas Finitos No Deterministas (AFND)

Propiedades:

- puede existir más de un camino para una cadena (puede estar en más de un estado simultáneamente)
- todo AFND puede convertirse en un AFD (el problema será el número de estados necesarios)
- si $M_{AFD} \equiv M_{AFND} \rightarrow L(M_{AFD}) = L(M_{AFND})$
- Suelen ser más compactos y fáciles de diseñar que los AFD

Definición de AFND

Un AFND es:

$$M=(Q, \sum, f, Q_0, F)$$

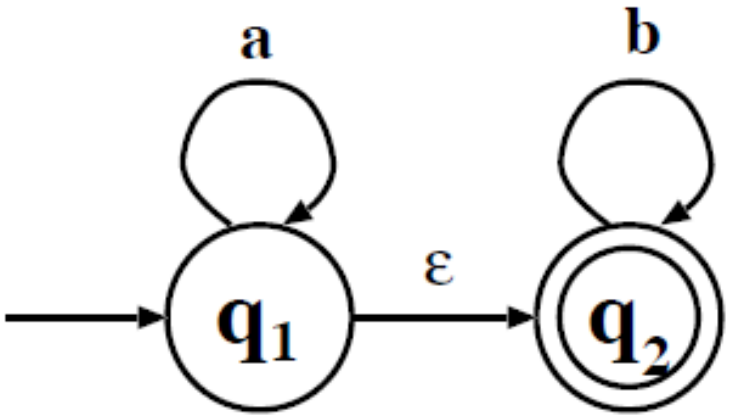
donde:

- $Q_0 \subseteq Q$ es un conjunto de estados iniciales
- $f: Q \times (\sum \cup \{\varepsilon\}) \rightarrow P(Q)$ es la función de transición (permite hacer transiciones con cualquier símbolo y con ε desde un estado a un conjunto de estados)
- Los demás elementos $(Q, \sum, F)$ tienen el mismo significado que en los AFD

Clausura nula

La clausura nula $C_\varepsilon(q)$ del estado q es el conjunto de estados accesibles directamente mediante transiciones-ε.

Si para algún q, $C_\varepsilon(q) \neq \{q\}$ el AFND contiene transiciones nulas.



δ	a	b	ε
q_1	q_1		q_2
q_2		q_2	

Ejemplo AFND

$$AFND_1=(\{q_0, q_1, q_2\}, \{0, 1\}, f, \{q_0\}, \{q_2\})$$

Con f:

$f(q_0, 0) = \{q_0, q_1\}$ $f(q_1, 0) = \emptyset$ $f(q_2, 0) = \emptyset$

$f(q_0, 1) = \{q_0\}$ $f(q_1, 1) = \{q_2\}$ $f(q_2, 1) = \emptyset$

$f(q_0, \varepsilon) = \emptyset$ $f(q_1, \varepsilon) = \{q_0, q_1\}$ $f(q_2, \varepsilon) = \emptyset$

Tabla de transiciones:

AFND	0	1	ε
→ q_0	$\{q_0,q_1\}$	$\{q_0\}$	
q_1		$\{q_2\}$	$\{q_0,q_1\}$
* q_2			

Ejemplo AFND

$$AFND_1=(\{q_0, q_1, q_2\}, \{0, 1\}, f, \{q_0\}, \{q_2\})$$

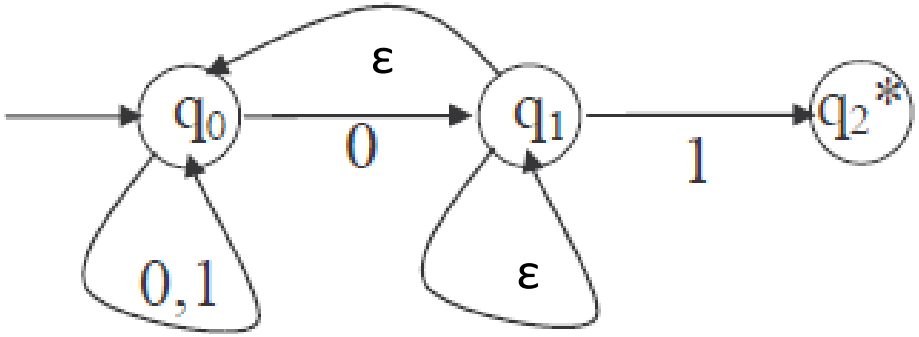
Con f:

$f(q_0, 0) = \{q_0, q_1\}$ $f(q_1, 0) = \emptyset$ $f(q_2, 0) = \emptyset$

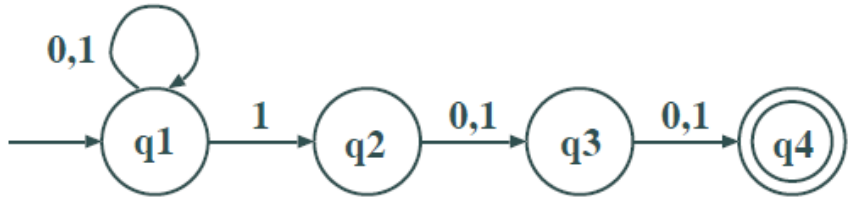
$f(q_0, 1) = \{q_0\}$ $f(q_1, 1) = \{q_2\}$ $f(q_2, 1) = \emptyset$

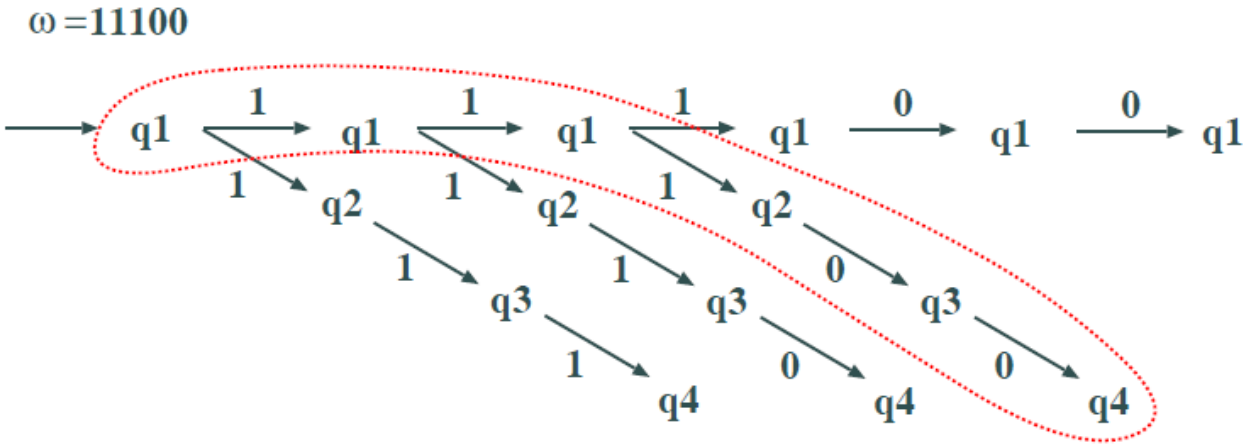
$f(q_0, \varepsilon) = \emptyset$ $f(q_1, \varepsilon) = \{q_0, q_1\}$ $f(q_2, \varepsilon) = \emptyset$

Diagrama de transiciones:



Ejemplo de procesamiento en un AFND



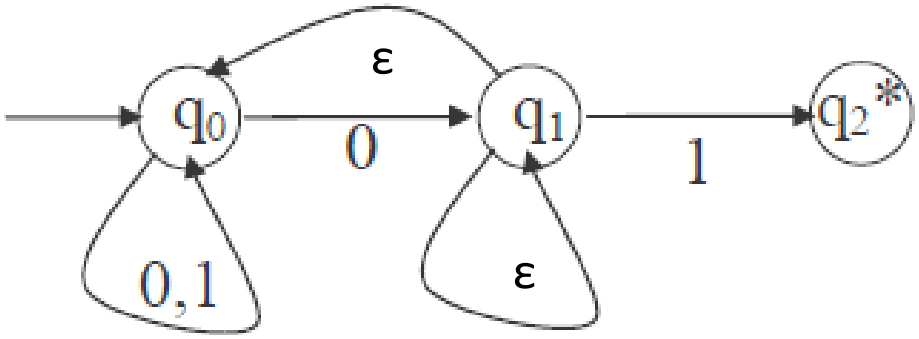


Con la cadena 11100 hemos llegado a dos estados del autómata, uno de aceptación y otro no, por lo tanto, la cadena es aceptada.

Lenguaje aceptado por un AFND

Un AFND acepta todas las palabras para las que puede transitar desde un estado inicial a un estado final.

Ejemplo: el AFND del ejemplo anterior



Acepta las palabras {x ∈ {0, 1}* | x termina en 01}

L(M)= {x = ab | a ∈ {0, 1}*, b=01}

Ejercicios AFND

Construir un AFND con cuatro estados que reconozca el lenguaje L siguiente:

$$L = \{a^n \mid n \geq 0\} \cup \{b^n a \mid n \geq 1\}$$

Ejercicios AFND

¿Cuál es el lenguaje reconocido por el siguiente autómata?