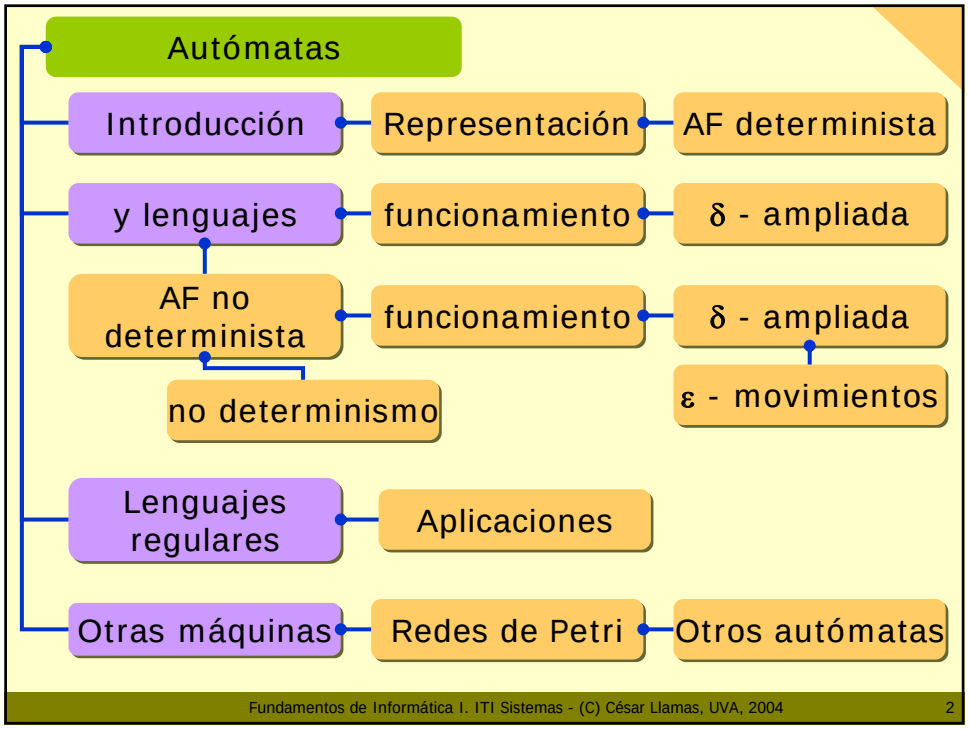
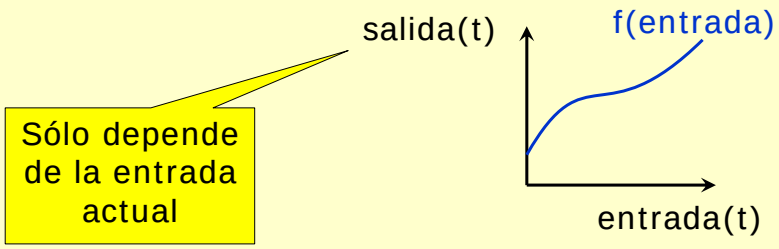
Autómatas


Autómatas
Introducción
Representación
AF determinista
y lenguajes
funcionamiento
δ - ampliada
AF no determinista
funcionamiento
δ - ampliada
ε - movimientos
no determinismo
Lenguajes regulares
Aplicaciones
Otras máquinas
Redes de Petri
Otros autómatas

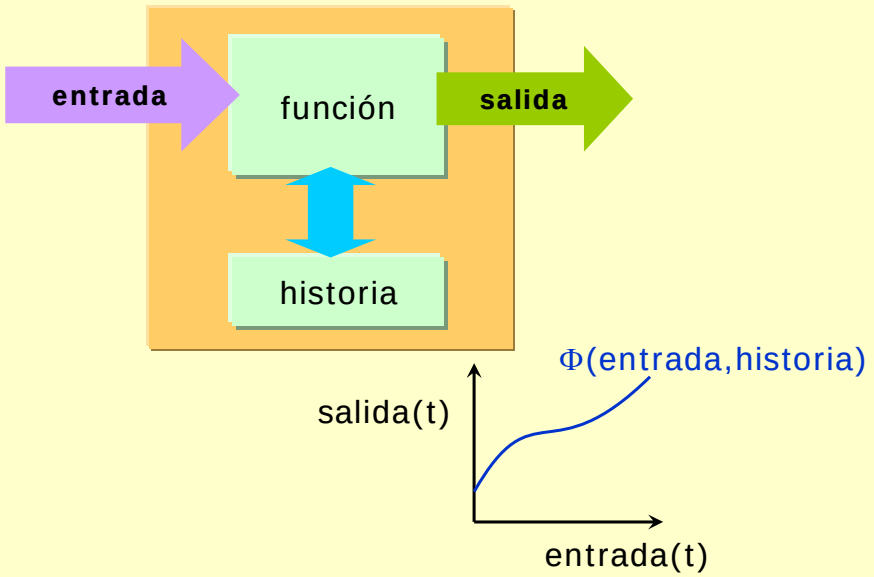
Autómatas

- Cualquier dispositivo autónomo
- Nos interesan autómatas sobre información.
  1. Leen un símbolo.
  2. Producen un símbolo.
  3. Vuelta al paso 1.
- Se asientan en la noción de estado
- Adoptan una forma procedural, más que declarativa.

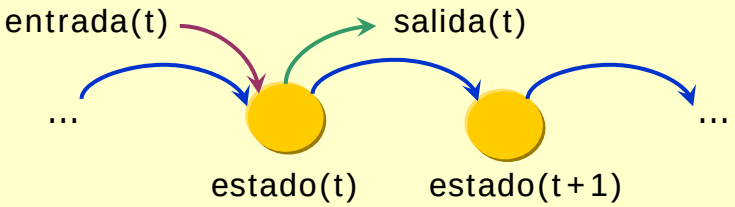
Máquina combinacional



Máquina secuencial



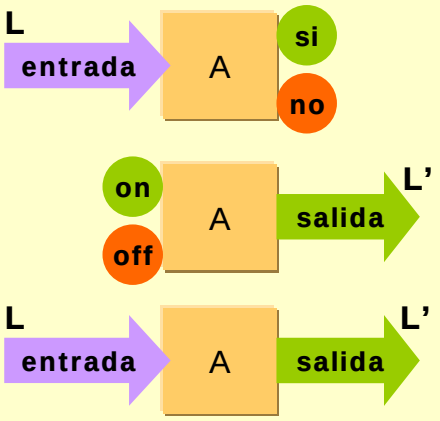
Noción de estado



- "configuración instantánea de un sistema"
  - El sistema presenta en cada momento uno de un conjunto de estados.
- La decisión sobre el conjunto de estados de un sistema es puramente arbitraria.
  - Ejemplo del sistema de la ventana.

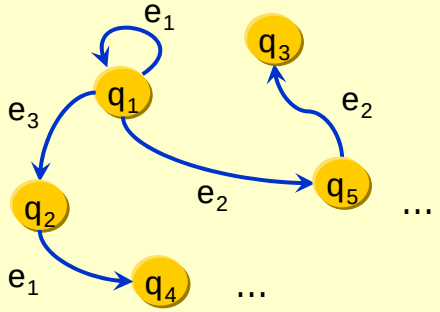
Autómatas y Lenguajes

- Tipos en términos de un lenguaje L:
  - Reconocedores
    Permiten especificar lenguajes (**L**)
  - Generadores
  - Traductores



Sistemas de transiciones etiquetadas

- Grafos dirigidos
  alfabeto de símbolos de transición (E)+
  alfabeto de símbolos de nodos (Q) +
  Relación {(q, e, q')}, donde
  - q, q' ∈ Q
  - e ∈ E

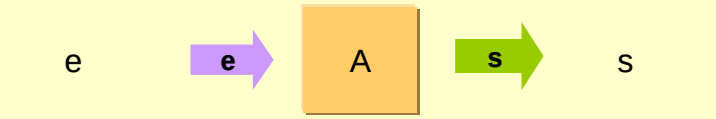


Representación de autómatas

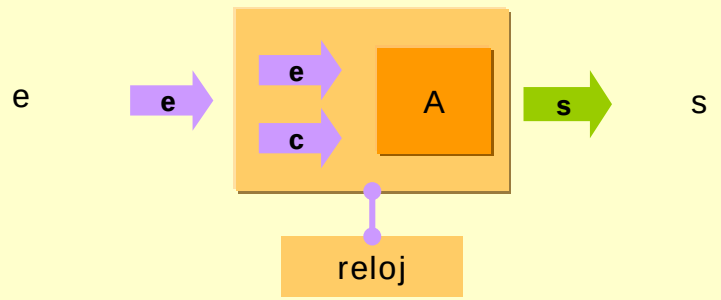
- E, alfabeto de símbolos de entrada
- S, alfabeto de símbolos de salida
- Q, alfabeto de símbolos de estado
- f, función de evolución
- g, función de salida

- Si Q es finito tenemos un AF.
- Si f y g son verdaderas funciones, tenemos un AFD.

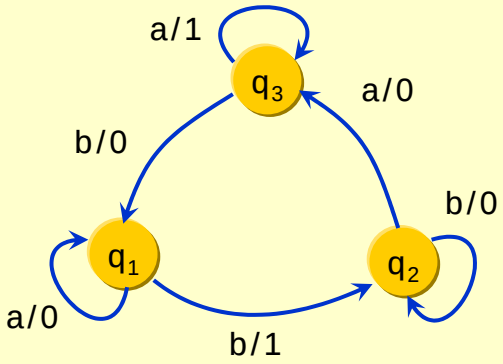
Autómatas y tiempo



+ tiempo

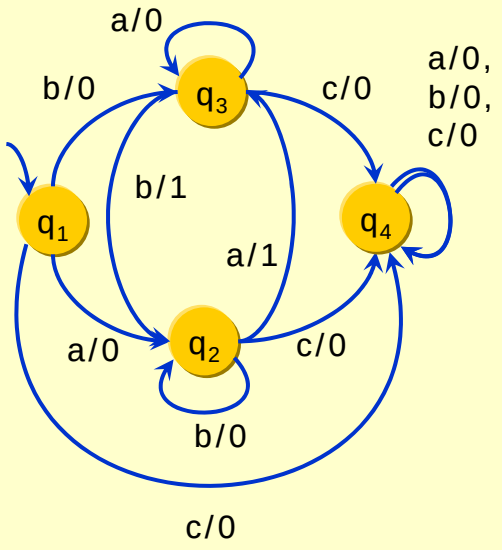


Ejemplo de autómata (1)



e_t	q_t	f( )	g( )
a	q_1	q_1	0
b	q_1	q_2	1
a	q_2	q_3	0
b	q_2	q_2	0
a	q_3	q_3	1
b	q_3	q_1	0

Ejemplo de autómata (1)



e_t	q_t	f( )	g( )
a	q_1	q_4	0
b	q_1	q_3	0
c	q_1	q_4	0
a	q_2	q_3	1
b	q_2	q_2	0
c	q_2	q_4	0
a	q_3	q_3	0
b	q_3	q_2	1
c	q_3	q_4	0
a	q_4	q_4	0
b	q_4	q_4	0
b	q_4	q_4	0

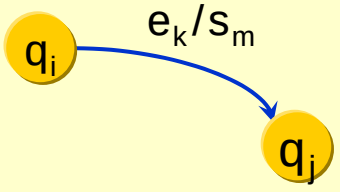
Autómata finito determinista ejecución

- s: variable de estado (actual)
- e: variable de símbolo de entrada (actual)
- o: variable de símbolo de salida (actual)
- s ← estado inicial

Mientras haya símbolos de entrada

1. e ← lee_símbolo();
2. [s, o] ← [f(e,s), g(e,s)];
3. emite_símbolo(o);

Representación de Mealy

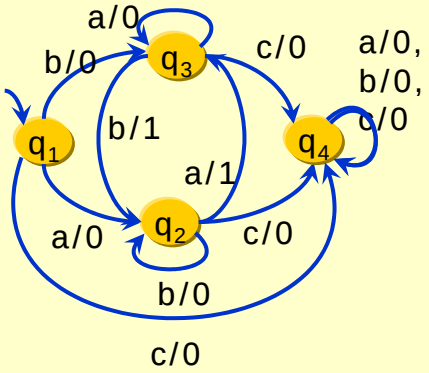


e_t	q_t	f( )	g( )
⋮			
e_k	q_i	q_j	s_m
⋮			

$f: E \times Q \rightarrow Q$

$g: E \times Q \rightarrow S$

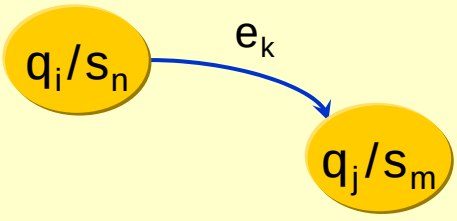
Operación del autómata de Mealy



Variables:
e: entrada
s: estado

```
s ← q1;  // define el estado inicial
mientras (e ← lee_entrada()) {
    emite( g(e, s) );
    s ← f(e, s);
}
```


Representación de Moore

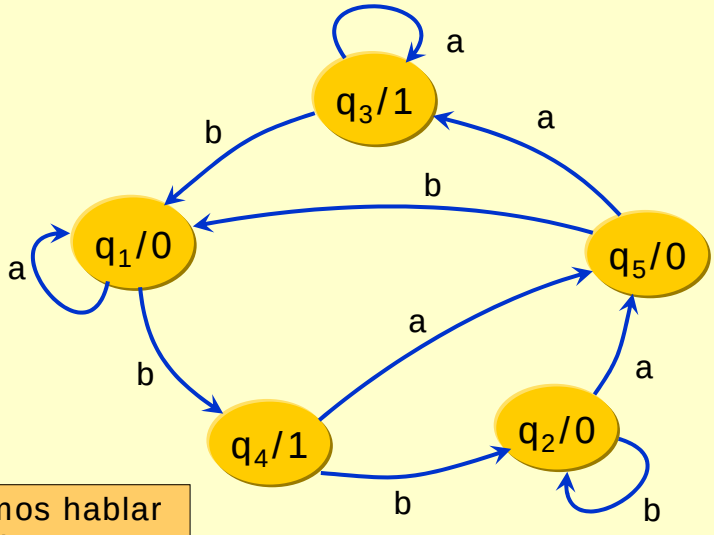


$f: E \times Q \rightarrow Q$

$g: Q \rightarrow S$

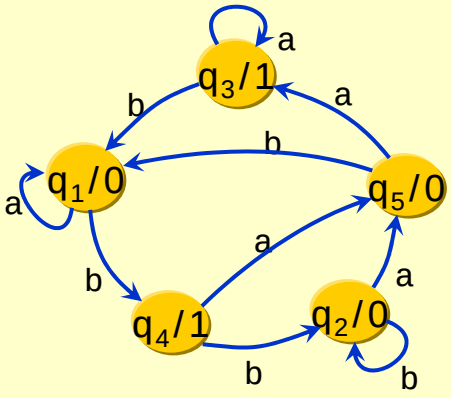
e_t	q_t	$f( )/g( )$
⋮		
e_k	q_i	q_j/s_m
⋮		

Ejemplo en la representación de Moore



¿Podemos hablar del elemento vacío (ε)?

Operación del autómata de Moore



Variables:
e: entrada
s: estado

```
s ← q1;  // define el estado inicial
mientras (e ← lee_entrada()) {
    emite( g(s) );
    s ← f(e, s);
}
```

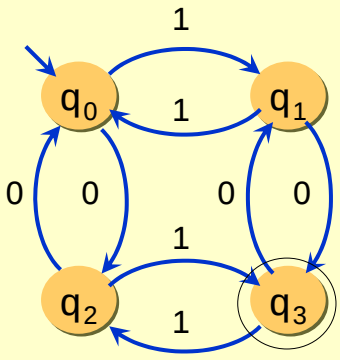
Autómatas aceptores de lenguajes

$A = \langle Q, \Sigma, \delta, q_0, F \rangle$

- Q, alfabeto de símbolos de estado
  - q_0 es el estado inicial
- F, alfabeto, no vacío, de estados aceptores, extraído de Q.
- Σ, alfabeto de símbolos de entrada.
- δ, función de transición $Q \times \Sigma \rightarrow Q$.

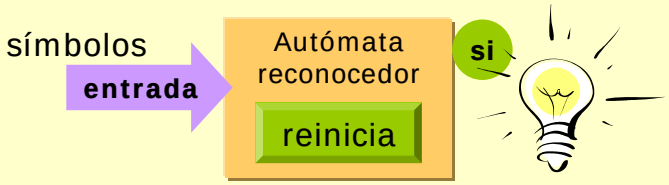
Autómatas aceptores ejemplo

$A = \langle Q, \Sigma, \delta, q_0, F \rangle$,
con $\Sigma=\{0,1\}$,
$Q=\{q_0,q_1,q_2,q_3\}$, y
$F=\{q_3\}$



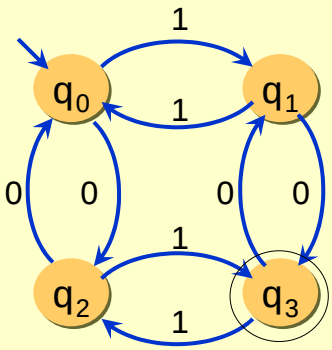
e_t	q_t	q_{t+1}
1	q_0	q_1
0	q_0	q_2
1	q_1	q_0
0	q_1	q_3
1	q_2	q_3
0	q_2	q_0
1	q_3	q_2
0	q_3	q_1

Autómatas aceptores ejecución



1. Pulsar reinicia
2. Mientras haya símbolos, hacer:
   1. Alimentar símbolo
   2. Si está en un estado de F, encender bombilla

Autómatas aceptores ejemplo



Traza

e_t	q_{t+1}	
-	q_0	
0	q_2	x
0	q_0	x
1	q_1	x
0	q_3	x ←
1	q_2	y
1	q_3	y ←
0	q_1	
1	q_0	
1	q_1	
1	q_0	

Define un lenguaje

x e y ∈ L(A)

Autómatas aceptores diseño

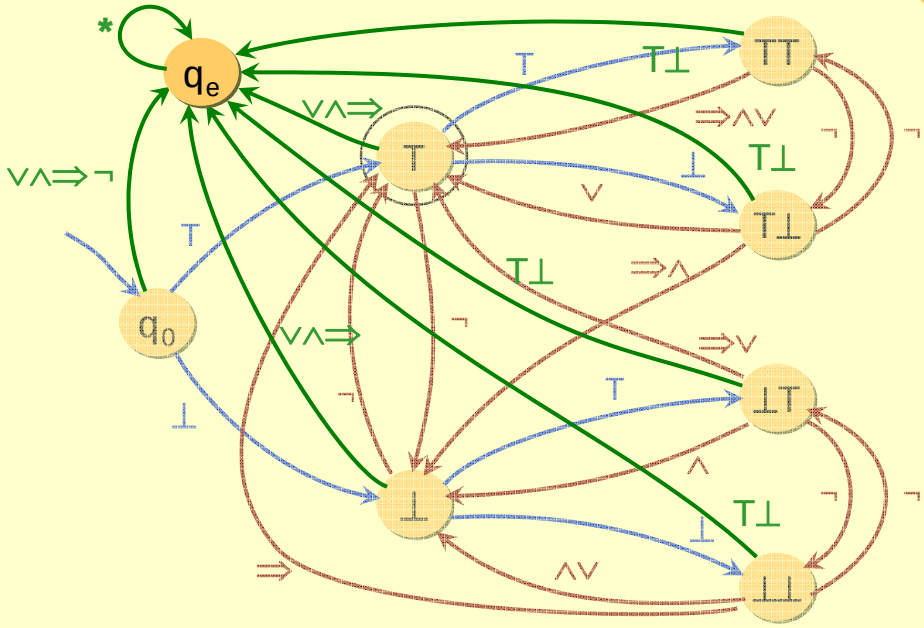
1. Concretar el lenguaje (propiedades)
2. Abstraer criterios de clasificación
3. Enumerar un conjunto de estados suficiente
4. Dibujar cada transición del autómata
5. Convencerse de su validez (i. e. cada cadena del lenguaje es un modelo del autómata)

Ejemplo: intérprete de expresiones lógicas cerradas

- Notación postfija:
  $(((\bot \Rightarrow \bot) \wedge T) \vee (\neg \bot)) \Rightarrow T)$
  debe escribirse como
  $\bot \bot \Rightarrow T \wedge \bot \neg \vee T \Rightarrow$
- Los símbolos se van proporcionando ordenadamente

$\bot \bot \Rightarrow T \wedge \bot \neg \vee T \Rightarrow$

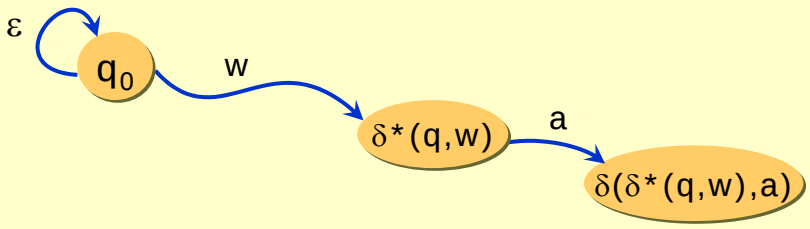
Intérprete de L. Propoposicional en notación posfija



Autómatas aceptores delta-ampliada

δ^*: $Q \times \Sigma^* \rightarrow Q$, para conocer el estado al que nos lleva una cadena completa.
La cadena de estados es única.

1. $\delta^*(q,\varepsilon)=q$
2. $\delta^*(q,wa)= \delta(\delta^*(q,w),a)$



Autómatas aceptores delta-ampliada

- Nociones:
  - Función de transición ampliada
  - Cadena aceptada por un autómata
  - Lenguaje aceptado por un autómata

- Los autómatas finitos deterministas aceptan conjuntos de cadenas regulares
  → Lenguajes regulares.

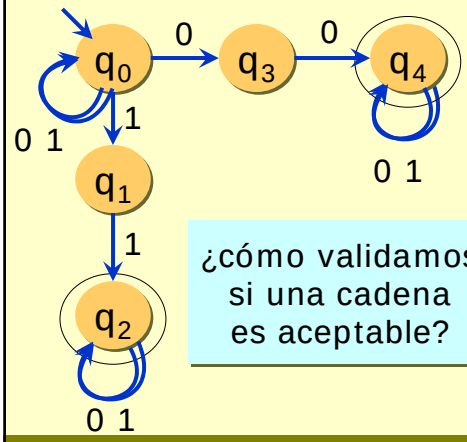
Autómatas finitos no deterministas (AFND)

- Peculiaridad:
  $\delta: Q \times \Sigma \rightarrow 2^Q$.
- Para ciertos estados, frente a una misma entrada, son posibles diversas transiciones.
- Problema gordo:
  ¿por cuál nos decidimos?
- Respuesta gorda:
  Por todos a la vez.
- Respuesta refinada:
  Por el que sea preciso, pero no necesariamente.
- ... Hay que meditarlo un poco más.

AFND ejemplo

$A=\langle\{q_0, q_1, q_2, q_3, q_4\}, \{0, 1\}, \delta, q_0, \{q_2, q_4\}\rangle$

Q Σ F

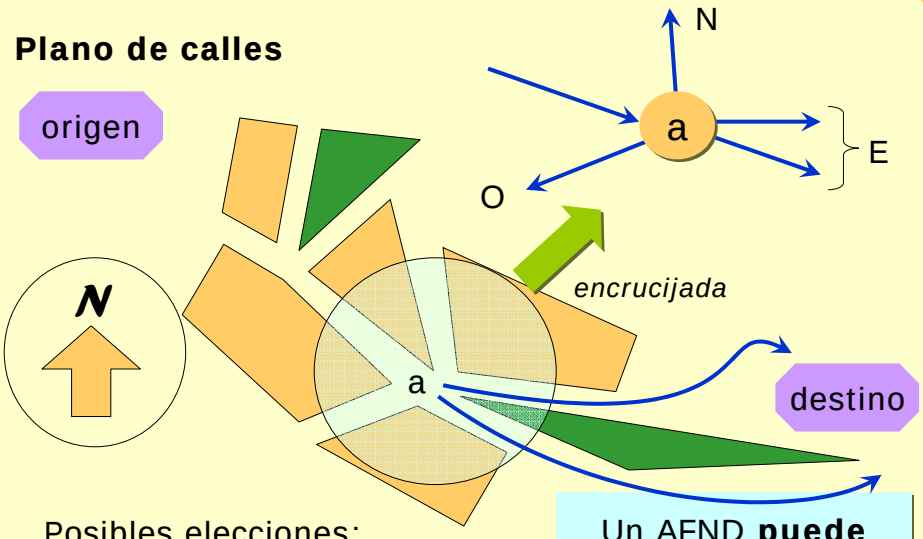


¿cómo validamos si una cadena es aceptable?

estado	entrada 0	entrada 1
q_0	$\{q_0, q_3\}$	$\{q_0, q_1\}$
q_1	$\varnothing$	$\{q_2\}$ ←
q_2	$\{q_2\}$ ←	$\{q_2\}$ ←
q_3	$\{q_4\}$ ←	$\varnothing$
q_4	$\{q_4\}$ ←	$\{q_4\}$ ←

Ejemplo de autómata no determinista

Plano de calles



Posibles elecciones:
norte, sur, este, oeste

Un AFND **puede** modelar el problema

AFND características

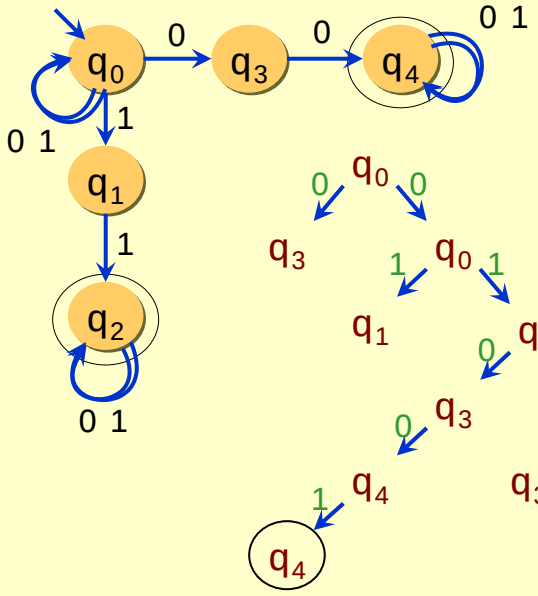
- Los AFND permiten modelar problemas donde se dan elecciones.
- Es posible definir el lenguaje aceptable por el autómata como:
  «*el conjunto de cadenas para las que existe al menos un camino para ir del estado inicial a uno de los estados aceptores*».
- Existe un procedimiento sistemático para saber si una cadena es aceptada por un AFND.
- La clase de lenguajes aceptables por los AFND es la misma que los AFD (lenguajes regulares)

AFND cálculo de aceptación

- Registraremos las transiciones en un árbol.

1. Inicializamos el autómata y registramos q0 como el nodo raíz.
2. Para el símbolo actual de la cadena ramificamos el árbol en términos de las posibles transiciones.
3. Repetimos 2. con la cadena de símbolos hasta que:
   1. No hay más ramas: la cadena no es aceptada.
   2. No hay más símbolos:
      1. Si estamos en un estado aceptor: la cadena es aceptada.
      2. Si no estamos en un estado aceptor: la cadena no es aceptada.

AFND cálculo de aceptación - ejemplo



Problema:
«01001»

AFND cálculo de aceptación (más formal)

Acepta ([estados] , [símbolos]) {

1. Si [símbolos] = ∅

   Si ∃ p ∈ [estados] | p ∈ F
   entonces: regresa Éxito
   si no: regresa Fracaso

2. Si [estados] = ∅
   entonces: regresa Fracaso
   si no:

   1.e ← [símbolos].cabeza;

   2.Para cada s ∈ [estados]

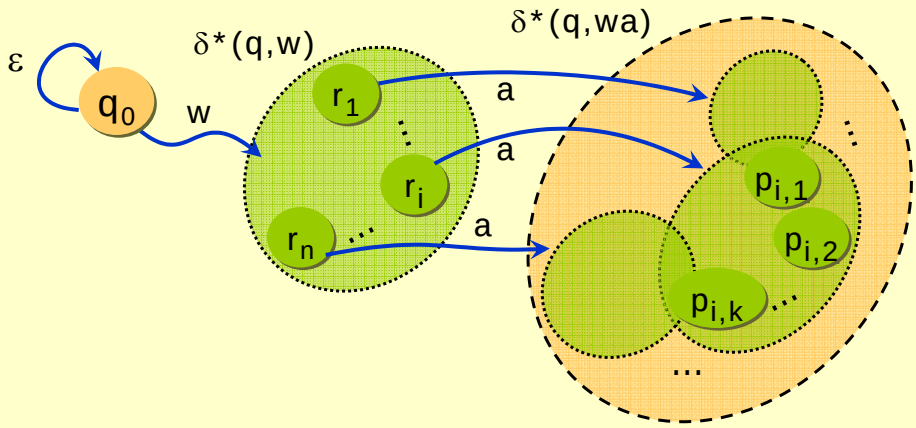
   1. [nuevos_estados] ← delta(s, e)
   2. regresa Acepta([nuevos_estados] , [símbolos].cola)

}

Autómatas aceptores delta-ampliada

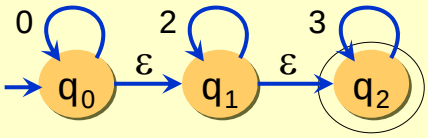
δ^*: $Q \times \Sigma^* \rightarrow 2^Q$, da el conjunto de estados donde lleva una cadena. Puede haber varias cadenas de estados.

1. $\delta^*(q,\varepsilon)=\{q\}$
2. $\delta^*(q,wa)=\{p \mid$ para algún estado r en $\delta^*(q,w)$, $p \in \delta(r,a)\}$



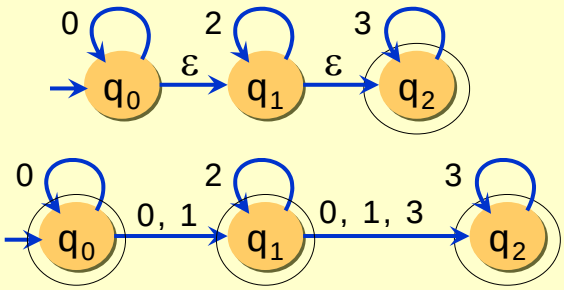
AFND con ε-movimientos

- Se admite, en ellos, las transiciones en ausencia de símbolo de entrada.
- Ej.: $A=\langle\{q_0, q_1, q_2\}, \{0, 1, 2\}, \delta, q_0, \{q_2\}\rangle$



	entradas			
estado	0	1	2	ε
q_0	$\{q_0\}$	∅	∅	$\{q_1\}$
q_1	∅	$\{q_1\}$	∅	$\{q_2\}$
q_2	∅	∅	$\{q_2\}$	∅

AFND con ε-movimientos AFND con ε-cierre



	entradas		
estado	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 cuestiones finales

- Los AFND permiten reflejar problemas con elección (No determinismo interno).
- El AFND que enciende siempre la bombilla cuando la entrada es aceptable solo es posible mediante simulación.
  - Funcionaría realmente, a cambio de que no le inspeccionáramos mientras funciona.
- El AFND real, en funcionamiento, podría no llegar a aceptar una cadena aceptable.

No determinismo y programación

- Es posible diseñar procesos informáticos utilizando No determinismo.
  - Podemos hacerles funcionar mediante una máquina que recorriera todas las elecciones posibles y se detuviera cuando tuviera la solución al proceso.
  - En el caso más simple un programa no determinista reflejaría un modelo de un sistema.

Expresiones regulares

- Definición de lenguaje regular a partir de conjunto regular.
  - Definición de conjunto + operación interna (aquí, la concatenación)

1. Conjuntos por enumeración de cadenas.
2. Conjuntos por unión de conjuntos.
3. Conjuntos por intersección de conjuntos.
4. Conjuntos por composición de conjuntos por la operación interna (⊗). Aquí, la concatenación de cada elemento del primer conjunto con los del segundo.

Conjuntos regulares ejemplo

- A={a, b, c}, B={ε, aa, bb}
- L = A∪B = {a, b, c, aa, bb, ε}
- L' = A∩B = ∅
  B = L ∩B = ...
- L'' = A⊗B = AB =
  {a, aaa, abb, b, baa, bbb, c, caa, cbb}
- ... y demás composiciones:
  (A∪B)A, A(A∪B), (A∪B)(A∩B)L', ...

Conjuntos regulares cierre de Kleene

- Sea Σ = {a, b, c},
  - L^0 = {ε} // cadenas de longitud 0
  - L^1 = {a, b, c} // cadenas de longitud 1
  - L^2 = {aa, ab, ba, ac, ca, bb, bc, cb, cc}
  - ...
  - L^i = {cadenas de longitud i} = L^1L^{i-1}
  - L^* = {todas las cadenas de talla numerable}

$$L^* = \bigcup_{i=0}^{\infty} L^i$$

Conjuntos regulares expresiones regulares

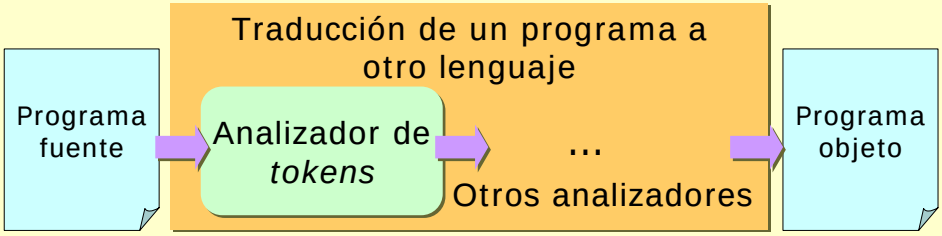
- Las expresiones regulares son fórmulas que se refieren a lenguajes regulares son más cómodas.
  1. $\varnothing$ representa el conjunto vacío
  2. ε representa $\{\varepsilon\}$
  3. **a** representa el conjunto con la cadena $\{a\}$
  4. Si r y s representan los conjuntos R y S
     1. (r+s) indica la unión de R y S
     2. (rs) indica la composición de R con S
     3. (r*) indica el cierre de Kleene de R.

Expresiones regulares ejemplo

- R= {a,b,aa,ab}, y S = {c}
  - (r+s) representa {a, b, aa, ab, c}
  - (rs) representa {ac, bc, aac, abc}
  - (r*) representa {ε, a, b, aa, ab, aaa, aab, baa, ...}, es decir equivale a (t*), donde t representa a T= {a,b}.
  - (s (r+s)) representa {ca, cb, caa, cab, cc}
  - (((s*) r) s) representa un conjunto como {ac, cac, ccac, bc, cbc, ccbc, ...}

Conjuntos regulares aplicación

- La mayoría de las palabras que forman parte de un documento en un lenguaje formal pueden definirse para que se ajusten a un lenguaje regular.
  - Limitamos la forma de construir identificadores de variables, funciones, etc, para que sean de un lenguaje regular, y así poder reconocerlas fácilmente.



Lenguajes regulares Ejemplos de aplicación

- *Tokens* en Algol:
  - letra: **A** + **B** + ... + **Z** + **a** + **b** + ... + **z**
  - dígito: **0** + **1** + ... + **9**
  - identificador: (letra)((letra + digito)*)
    como “a”, “abc”, “a45”, ...
  - nnatural: (digito)(digito*)
    como “3”, “443”, ...
  - nentero: (**+** + **–** + ε)(nnatural)
    como “+4”, “-83”, “15”, ...

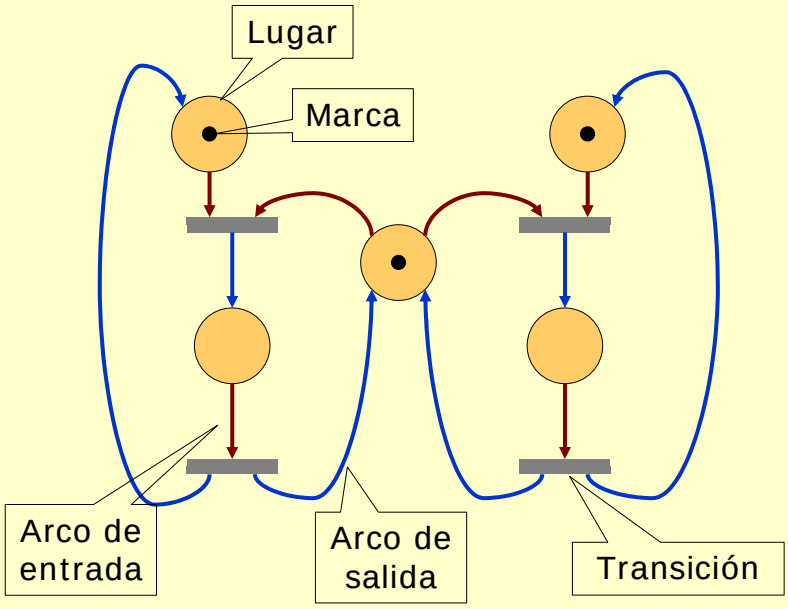
Lenguajes regulares Ejemplos de aplicación

- En Fortran 77 los identificadores no podían sobrepasar el tamaño de 6 letras:
  - identificador: (letra)(ε + letra + digito)^5
- Número en punto flotante:
  - (ε + **+** + **–**)(
    ((digito+(digito*))**.**(digito*)) +
    ((digito*)**.**(digito+(digito*)))
)**E**(**+** + **–**)(digito (digito*))

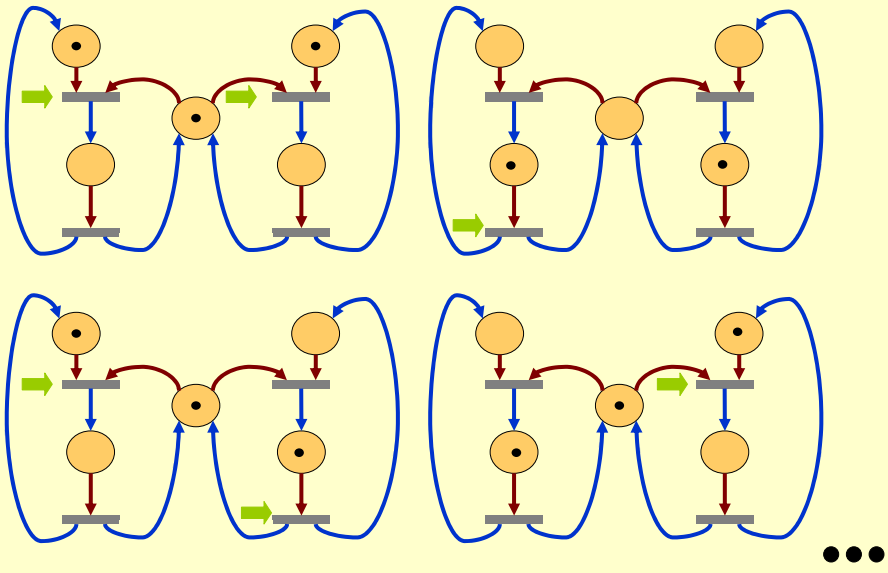
Otras máquinas de estado

- Razones para otras máquinas:
  - El nivel expresivo es demasiado elemental no admitiendo otros lenguajes más complejos
  - Se precisa modelar el comportamiento de sistemas estocásticos.
  - El modelo del sistema debe variar (ej. número de estados, ...)

Redes de Petri



Redes de Petri simulación



Autómatas estocásticos

- la función de transición y la función de salida pueden ser estocásticas:
  - Hay una probabilidad asociada a las transiciones desde un estado a otro.
  - Hay una probabilidad asociada a la emisión de un símbolo desde un estado, para cierto símbolo de entrada.
- En los autómatas borrosos, la función de transición y salida están, también, parametrizadas por valores, pero no existe una medida de probabilidad.

Autómatas con aprendizaje