Tema 1: Lenguajes Regulares

Lección 1.2

Autómatas Finitos Deterministas (AFD)

Jorge Bernad

Índice

Simular una máquina expendedora

- Simulemos una máquina expendedora sencilla.
- Todos los productos en la máquina cuestan 1 euro.
- La máquina no puede devolver cambio. Es necesario echar el dinero justo.
- Solo acepta monedas de 1 euro.
- Tiene un botón que devuelve todo el dinero echado.
- Si se golpea la máquina, se bloquea (y no devuelve el dinero)

Simulación

Acciones usuario

El usuario puede realizar las siguientes acciones:

- Echar una moneda de 1 euro (M).
- Pulsar el botón de devolución (D)
- Golpear la máquina (G).

Estados de la máquina

La máquina puede estar en los siguientes estados:

- Al inicio, la máquina tiene 0 euros en el cajetín de servicio.
- Hay 1 euro en el cajetín de servicio. Para poder servir un producto, la máquina debería estar en este estado (estado final)
- Hay más de 1 euro en el cajetín.
- La máquina está bloqueada.

Objetivo de la simulación

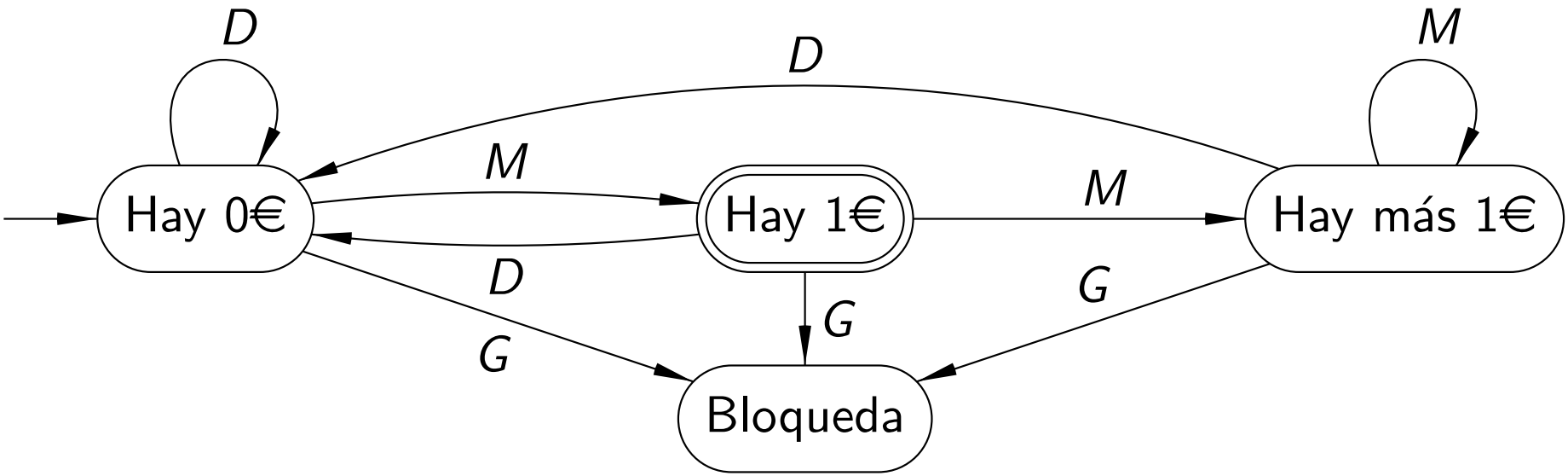
Objetivo

- Dada una serie de acciones que puede ejecutar un usuario, ¿la máquina puede expender el producto?.
- Esto es, ¿hay un 1 euro en el cajetín de servicio?
- Por ejemplo, si el usuario ha realizado las acciones:

$$MMDM$$

¿se puede servir el producto?

“Pintemos” el problema



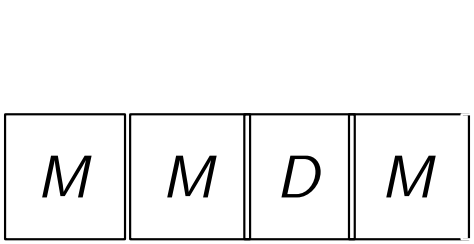
- Hemos pintado un Autómata Finito Determinista
- Tiene 4 estados.
- Las transiciones entre estados van marcados por las flechas y su etiqueta: si estás en el estado “Hay 0€” e introducen una moneda (M), pasa al estado “Hay 1€”.
- El estado inicial está marcado por una flecha.
- El estado final esta doblemente enmarcado.

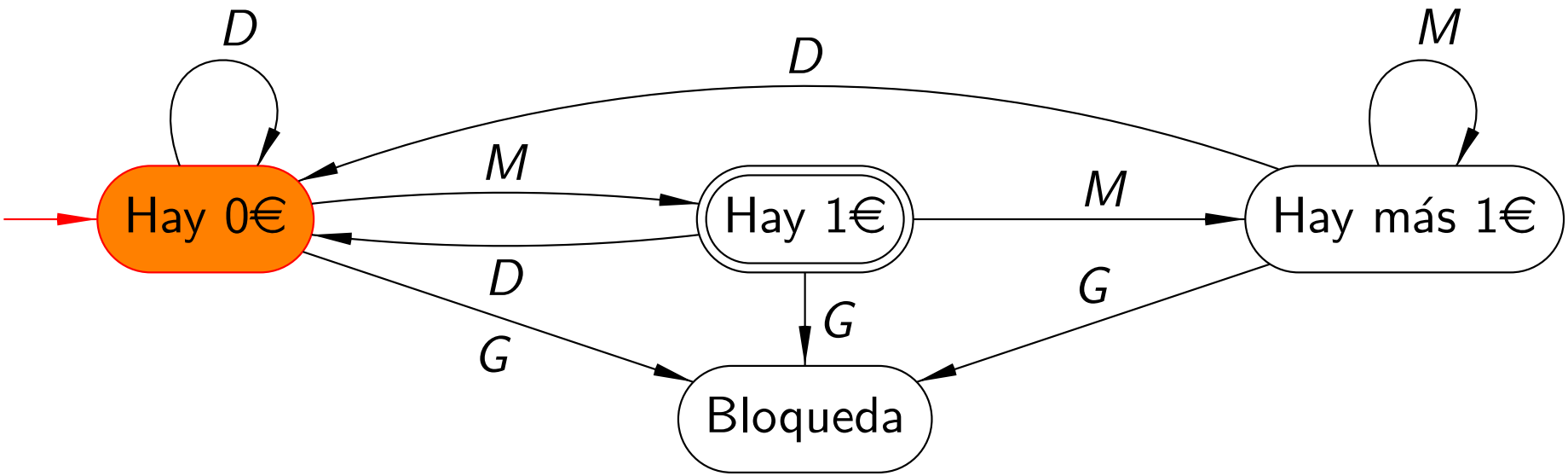
Relación con los lenguajes

- Sea $\Sigma = \{M, D, G\}$ un alfabeto.
- Una cadena w sobre Σ es aceptada por el autómata si al finalizar de investigar la cadena nos quedamos en un estado final.
- ¿Es aceptada la cadena $w = MMDMMMDM$?
- ¿Y la cadena $w = MMDGMDM$?

Cómo se ejecuta un autómata

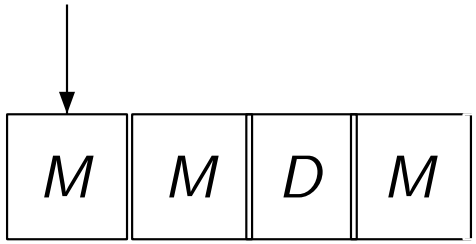
Configuración incial

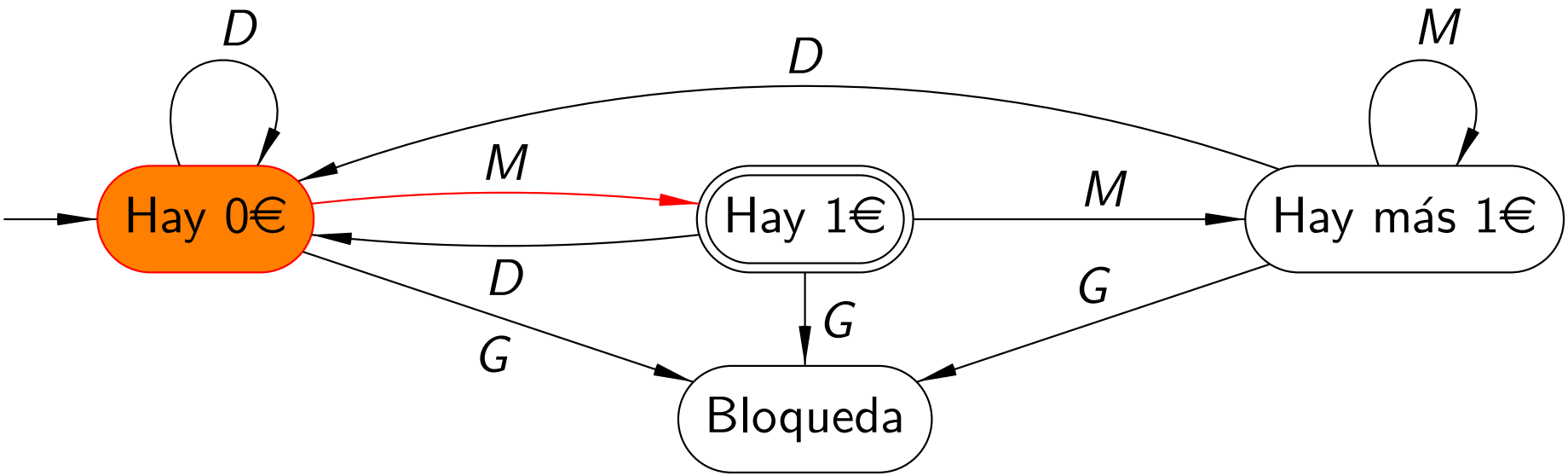




Cómo se ejecuta un autómata

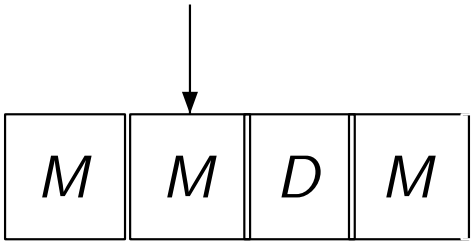
Leer siguiente símbolo

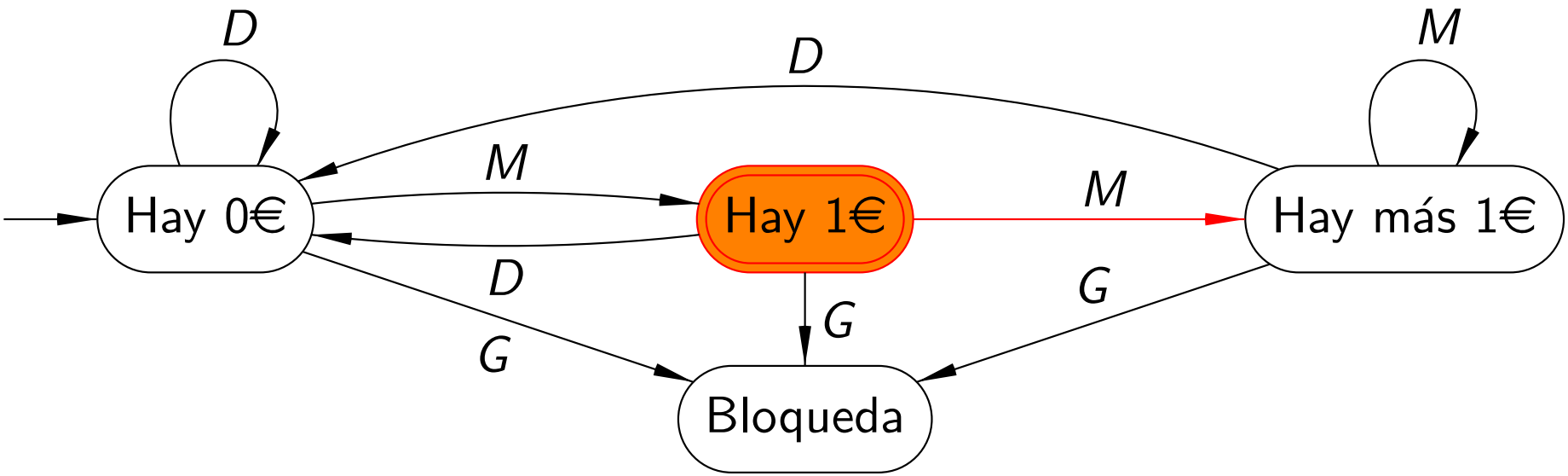




Cómo se ejecuta un autómata

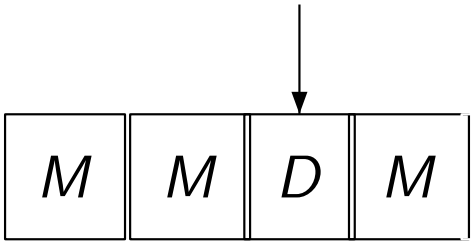
Leer siguiente símbolo

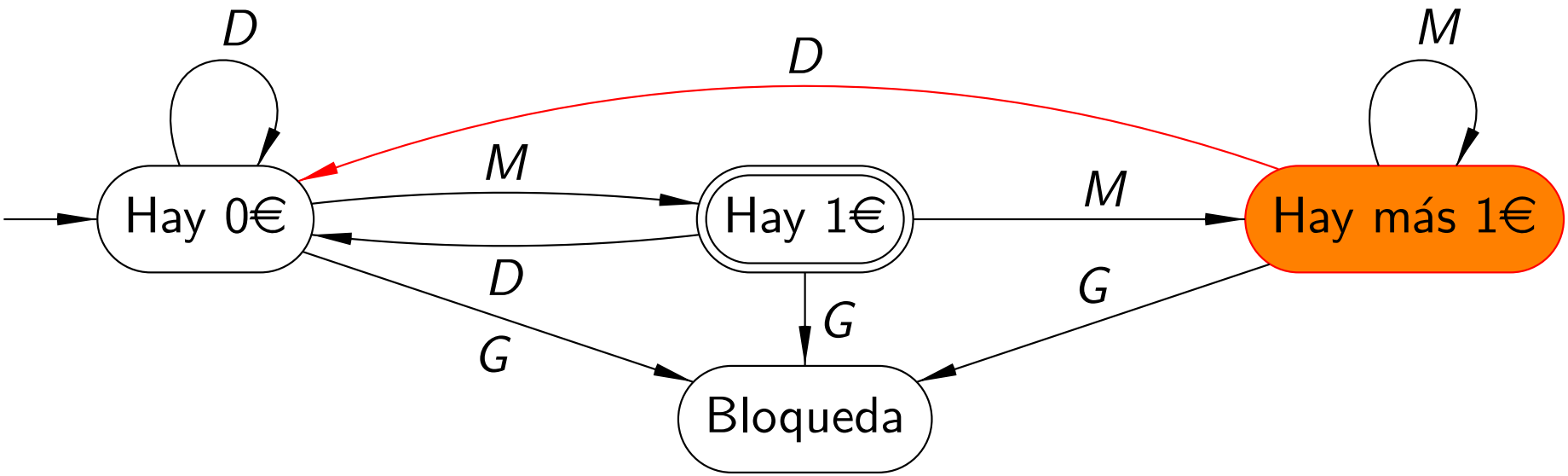




Cómo se ejecuta un autómata

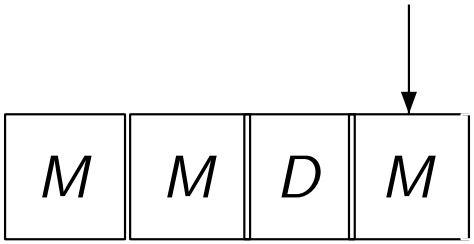
Leer siguiente símbolo

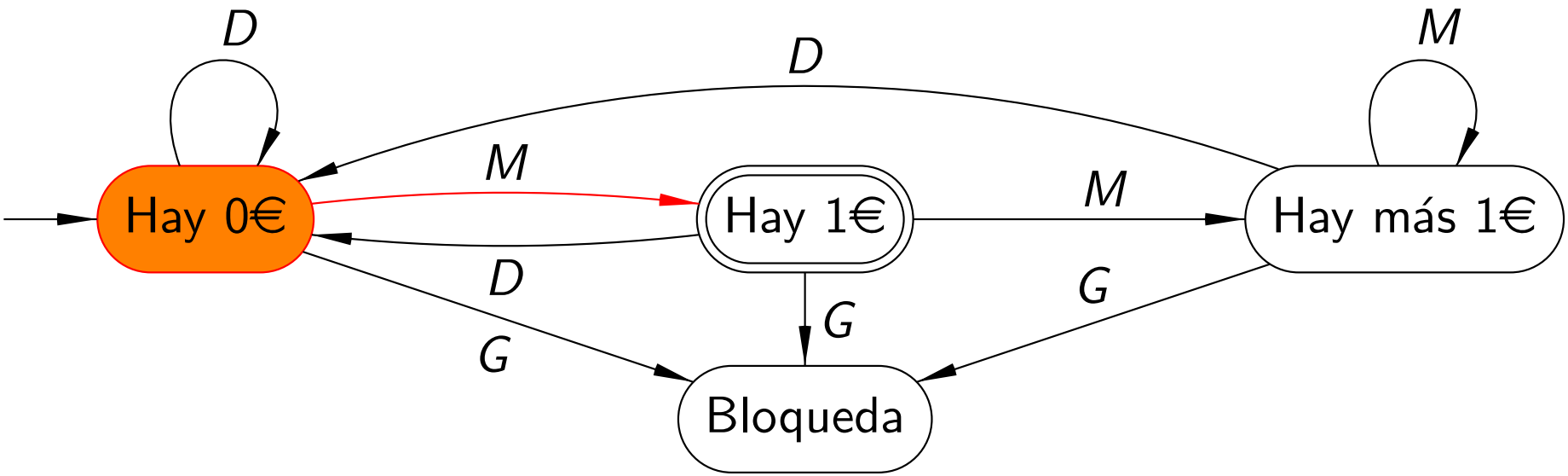




Cómo se ejecuta un autómata

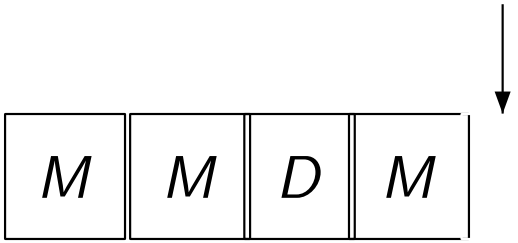
Leer siguiente símbolo

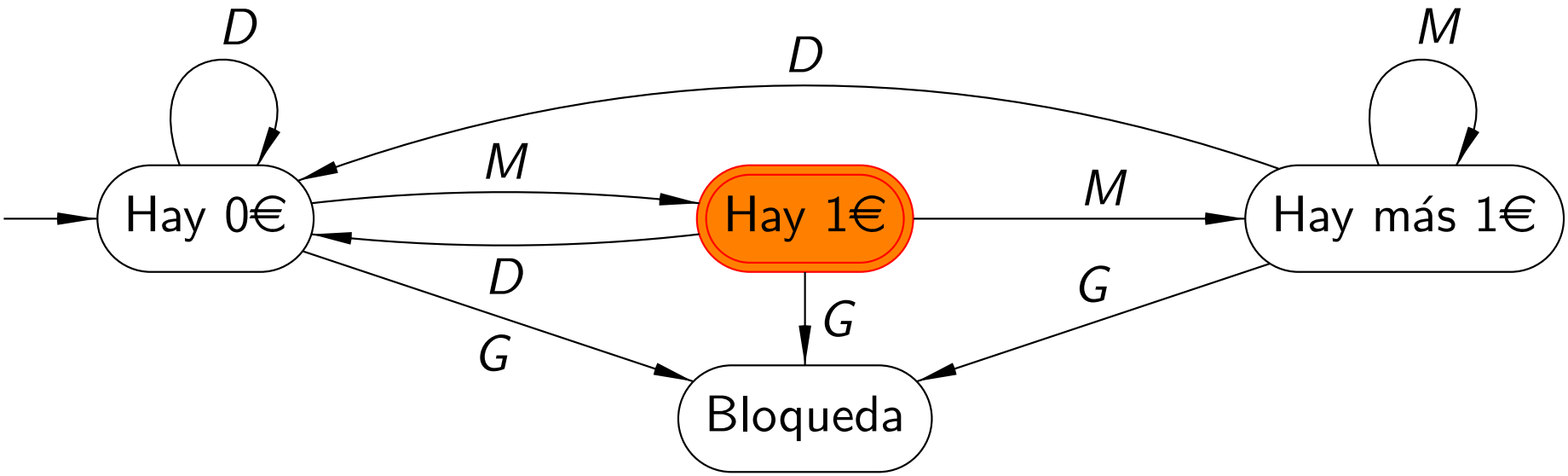




Cómo se ejecuta un autómata

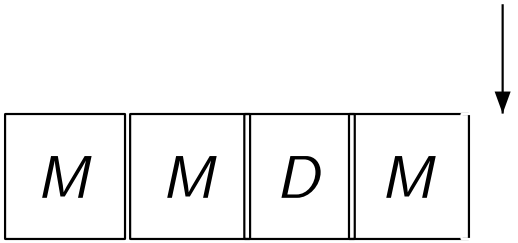
Fin de lectura



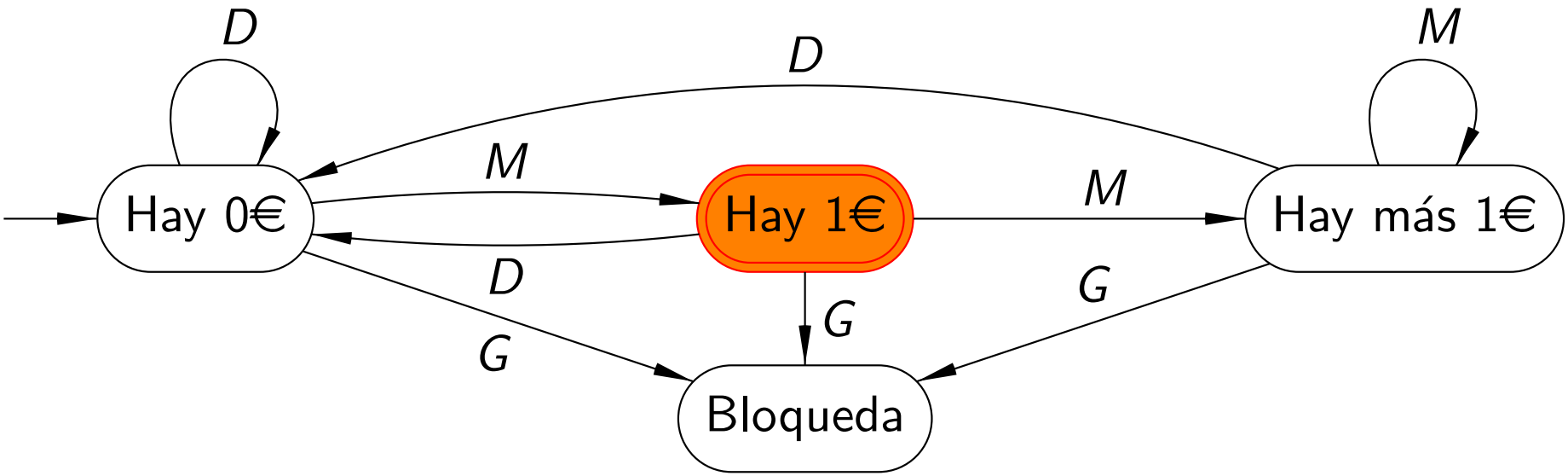


Cómo se ejecuta un autómata

Fin de lectura

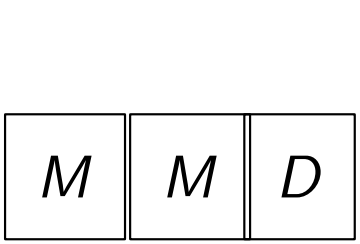


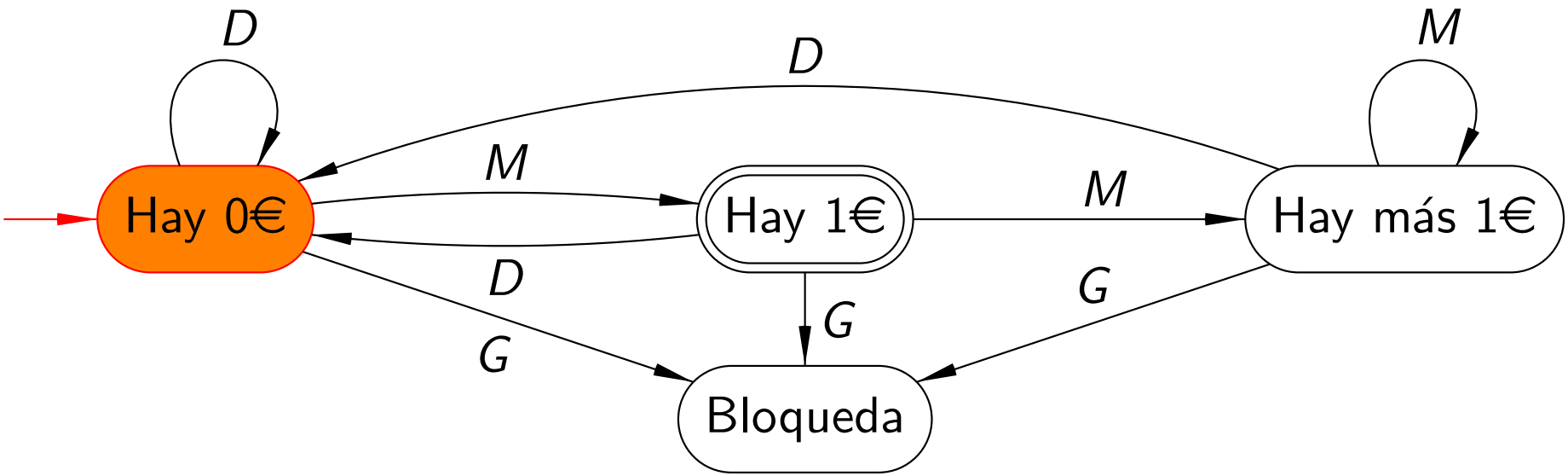
ACEPTADA



Cómo se ejecuta un autómata

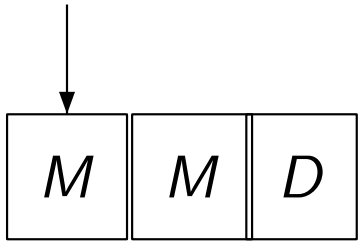
Configuración incial

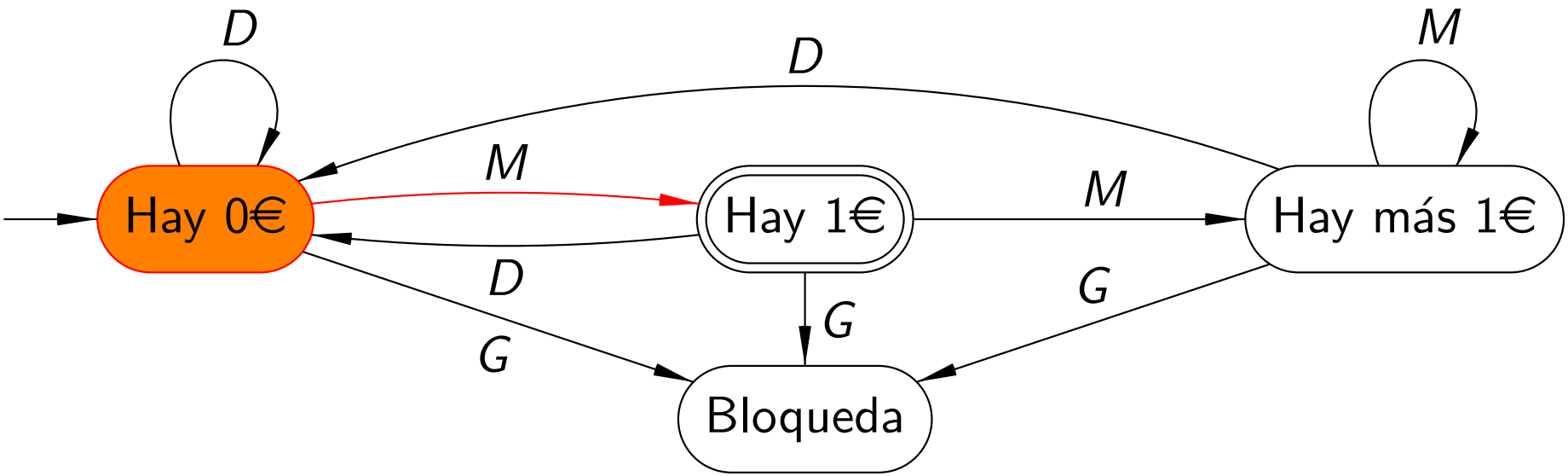




Cómo se ejecuta un autómata

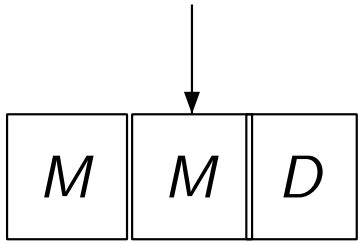
Leer siguiente símbolo

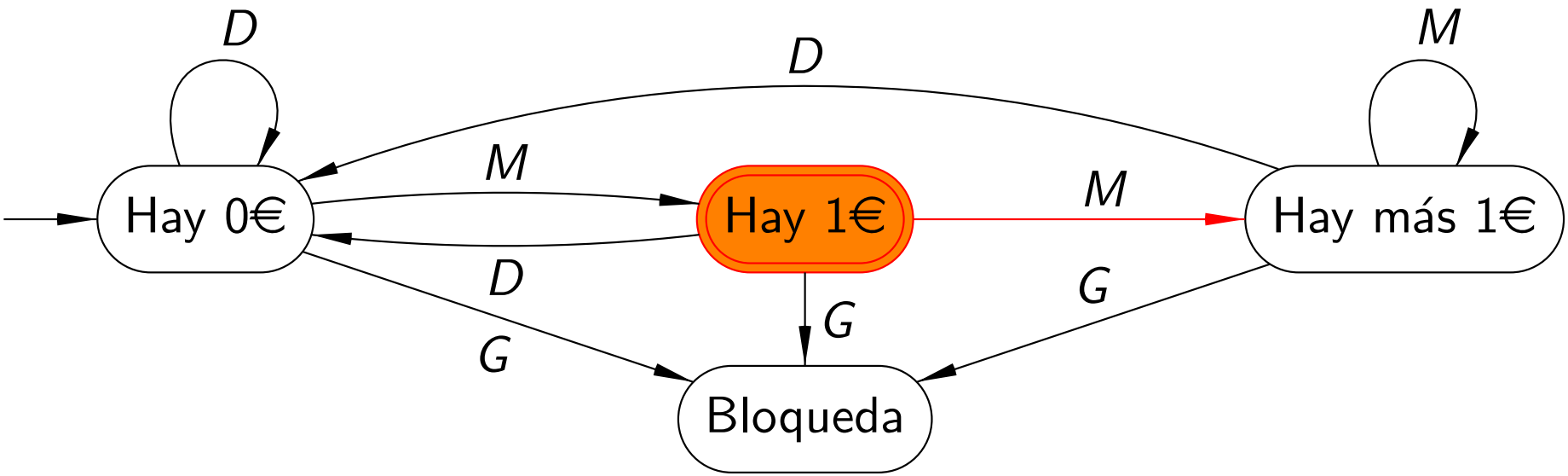




Cómo se ejecuta un autómata

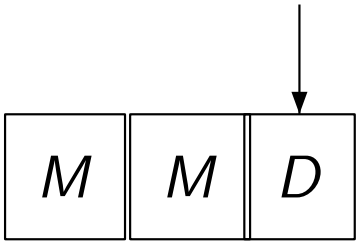
Leer siguiente símbolo

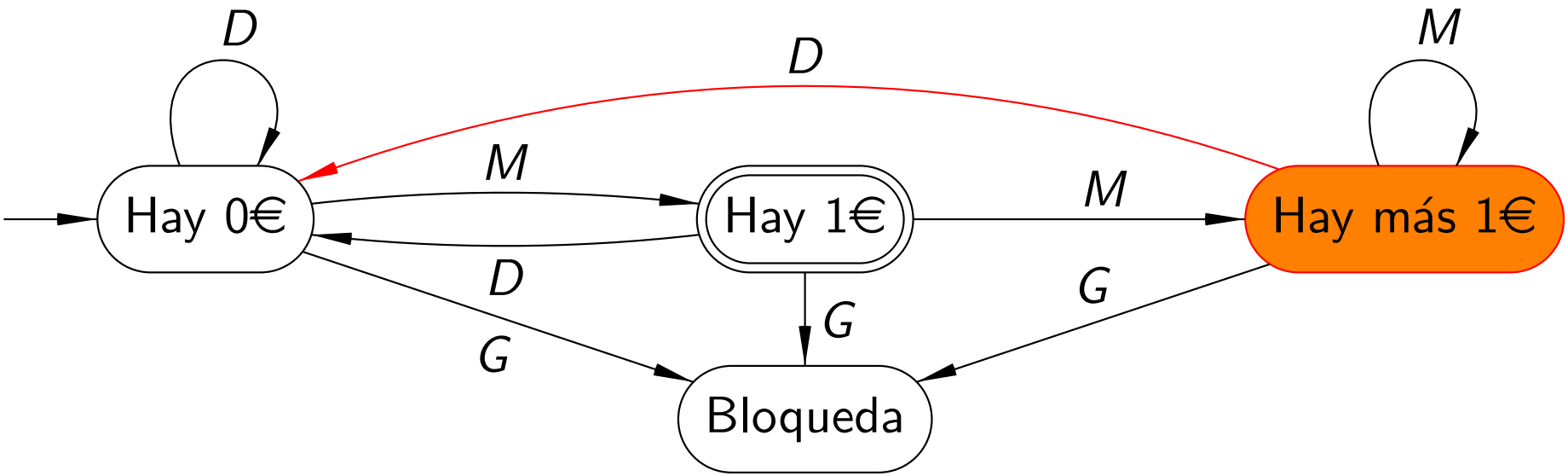




Cómo se ejecuta un autómata

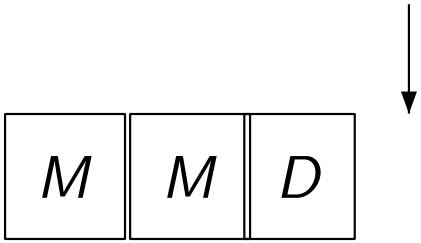
Leer siguiente símbolo

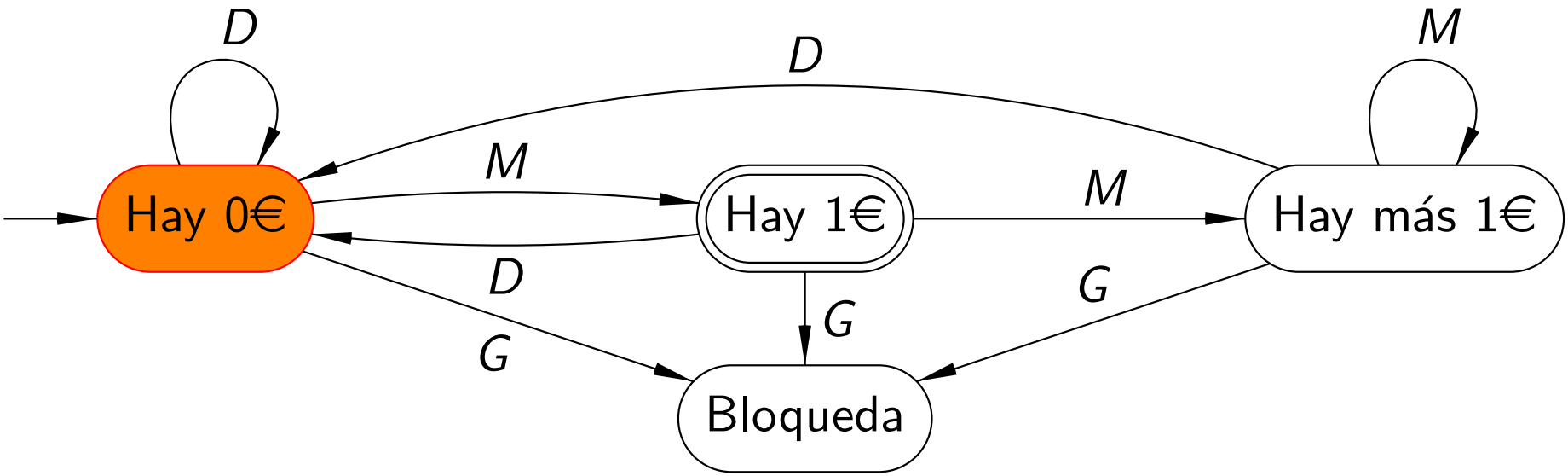




Cómo se ejecuta un autómata

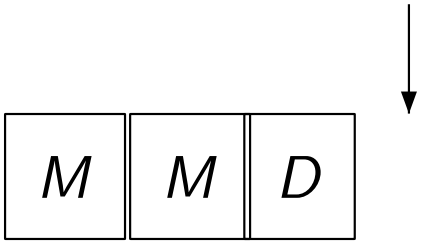
Fin de lectura



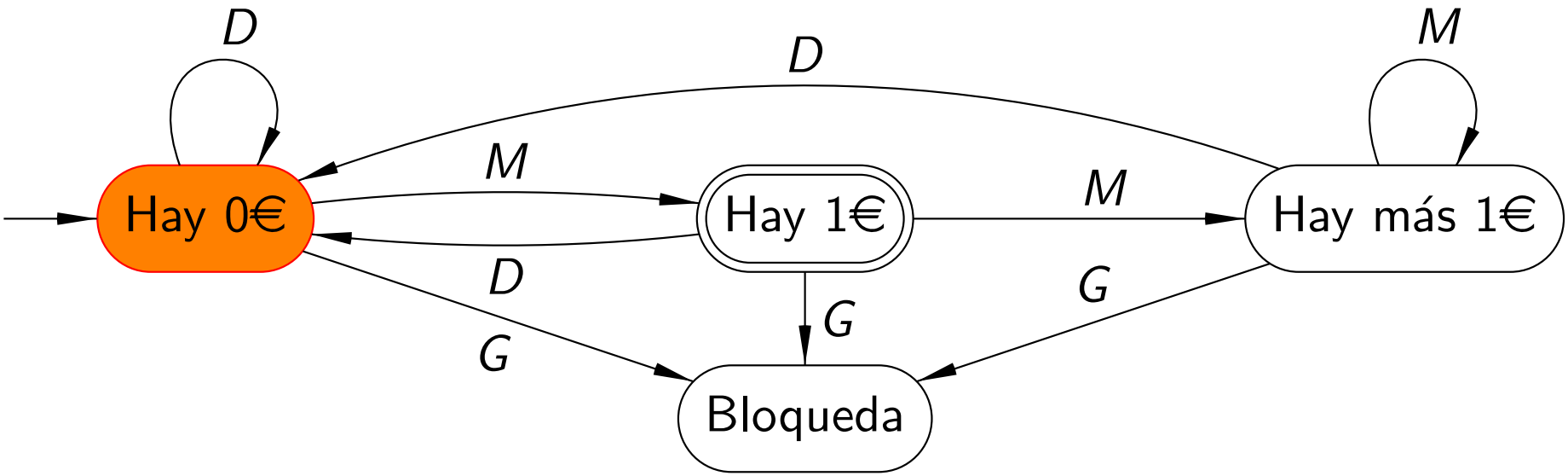


Cómo se ejecuta un autómata

Fin de lectura

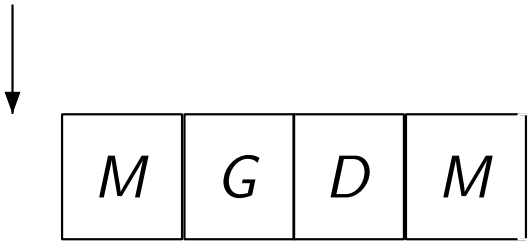


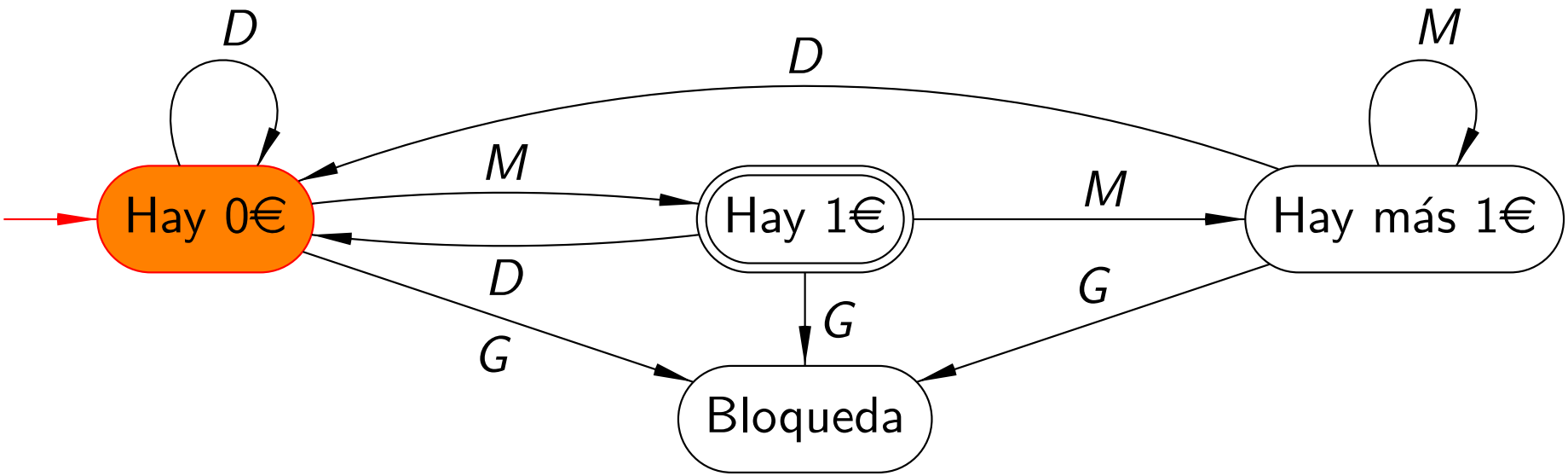
RECHAZADA



Cómo se ejecuta un autómata

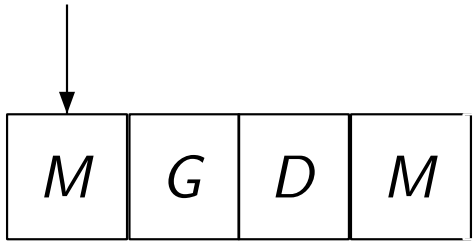
Configuración incial

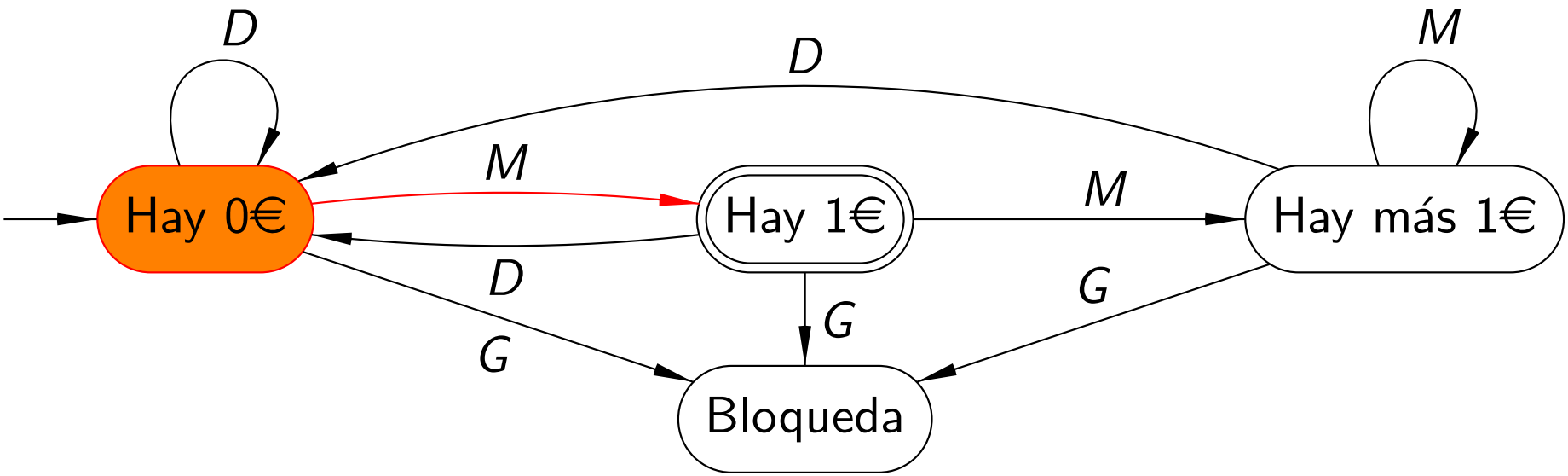




Cómo se ejecuta un autómata

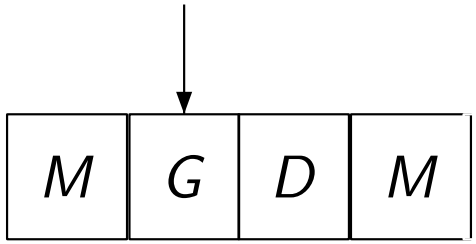
Leer siguiente símbolo

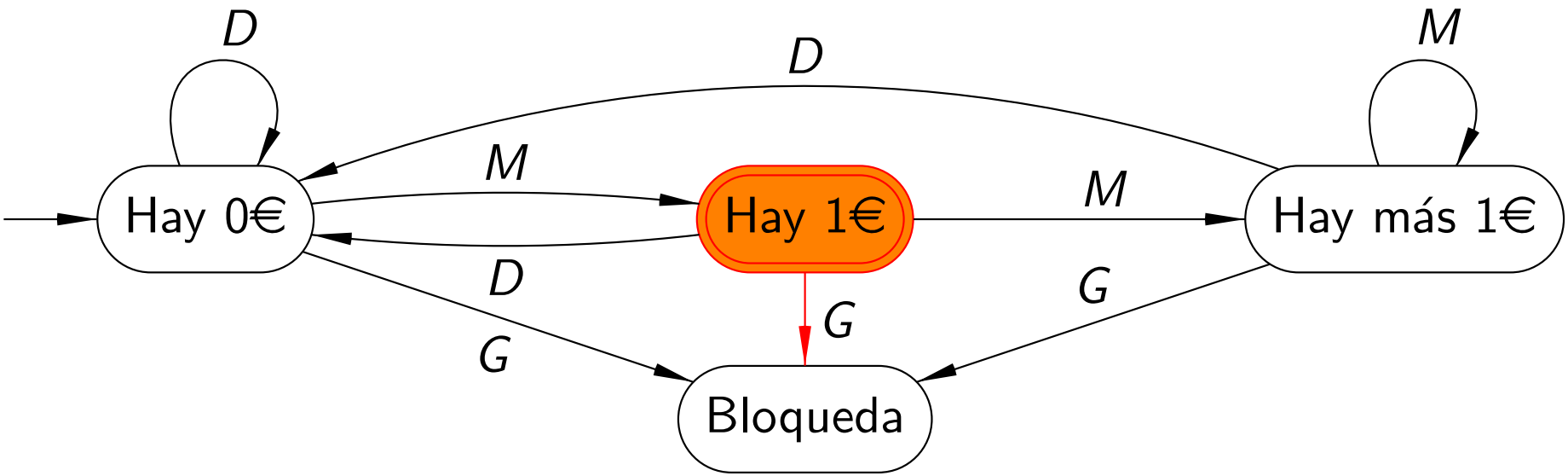




Cómo se ejecuta un autómata

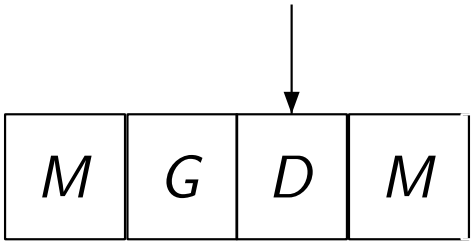
Leer siguiente símbolo

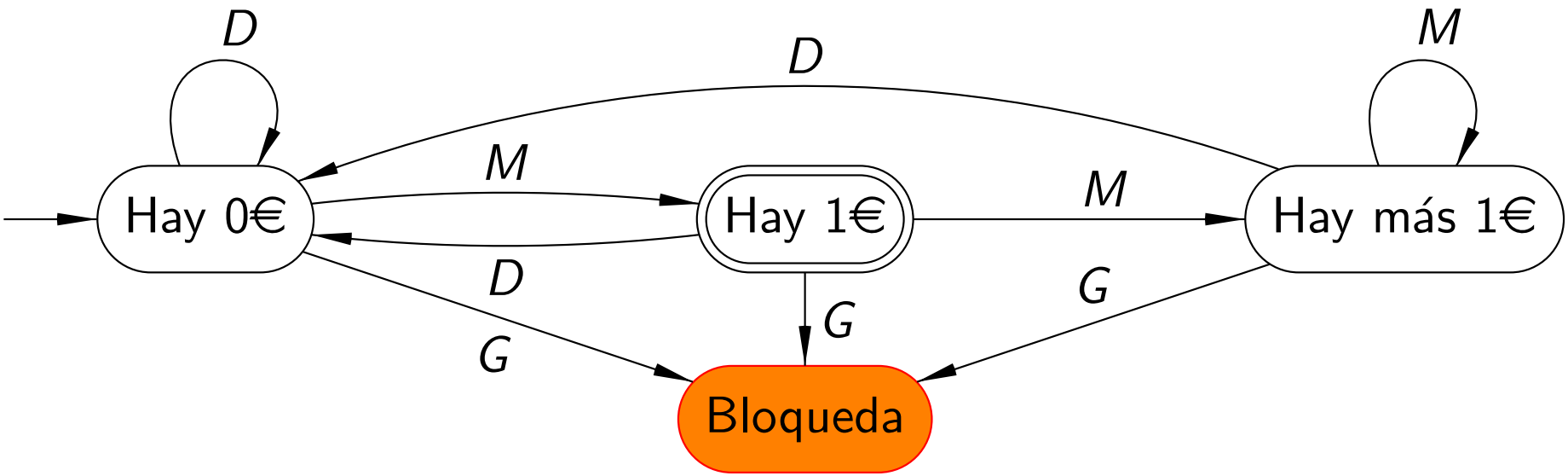




Cómo se ejecuta un autómata

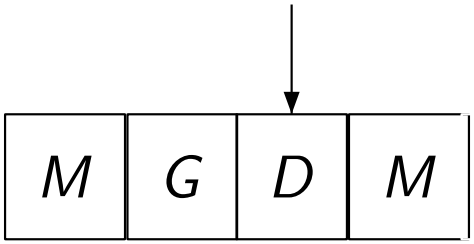
No se puede seguir



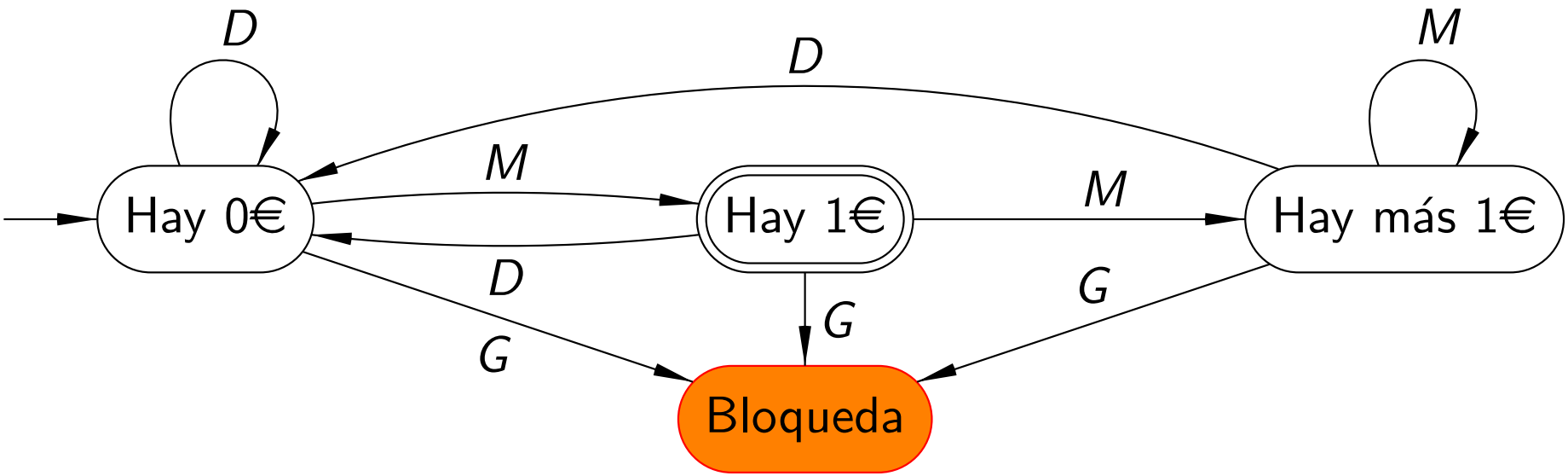


Cómo se ejecuta un autómata

No se puede seguir



RECHAZADA

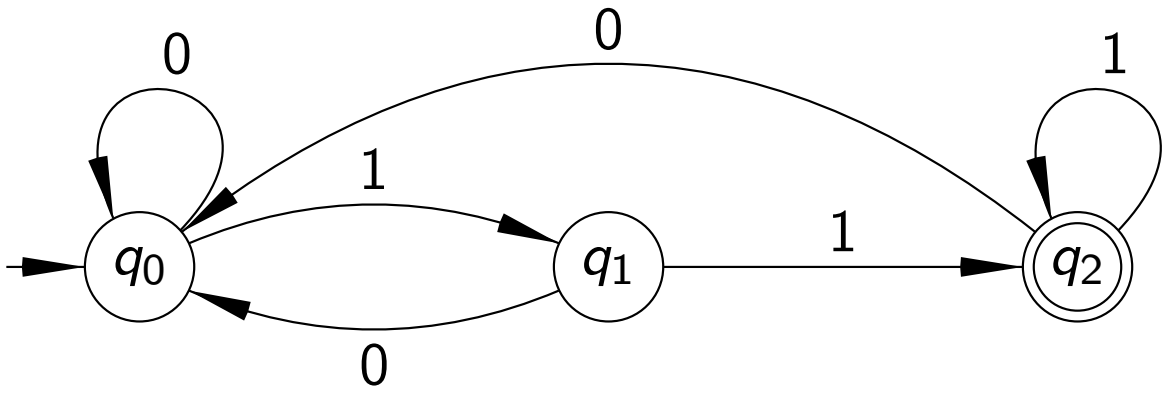


Lenguaje reconocido por el autómata

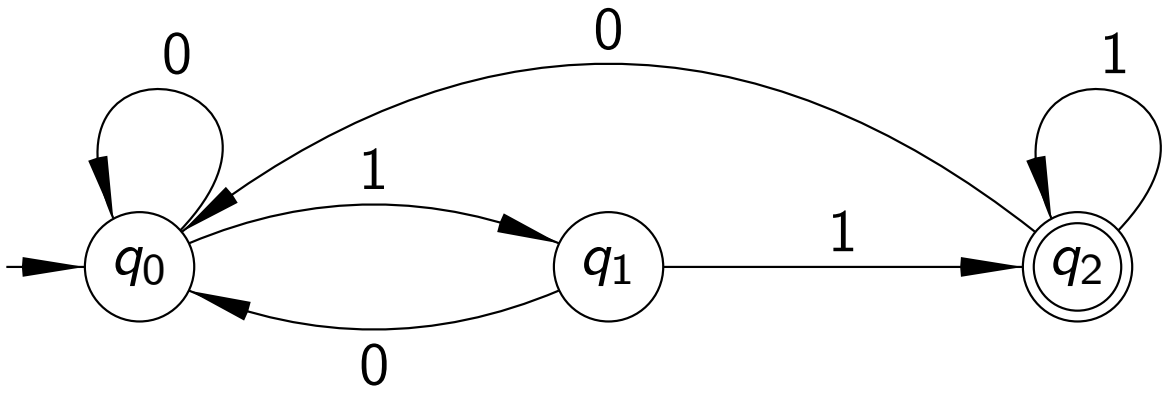
- El lenguaje reconocido por el autómata es el conjunto de cadenas que son aceptadas.

$$
\begin{aligned}
L = \{w \in \Sigma^* \mid & w \text{ tal que hay 1€ en el cajetín,} \\
& \text{y no se ha golpeado la máquina}\} \\
= \{w \in \Sigma^* \mid & w \text{ no contiene ninguna } G\text{, termina en } M \\
& \text{y si } |w| > 1\text{, el penúltimo es } D\}
\end{aligned}
$$

¿Qué lenguaje reconoce sobre el alfabeto $\Sigma = \{0, 1\}$?



¿Qué lenguaje reconoce sobre el alfabeto $\Sigma = \{0, 1\}$?



$$L = \{w \in \Sigma^* \mid w \text{ termina en } 11\}$$

Otro ejercicio

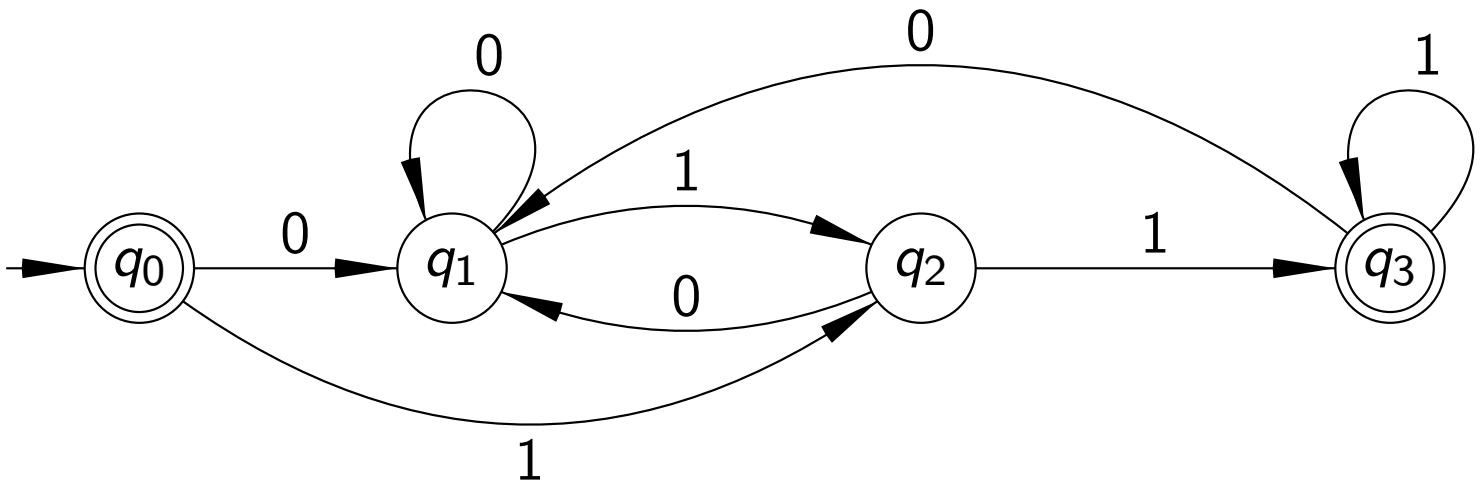
Crear un autómata que reconozca el lenguaje

$$L = \{w \in \Sigma^* \mid w \text{ termina en } 11 \text{ o es igual a } \epsilon\}$$

Otro ejercicio

Crear un autómata que reconozca el lenguaje

$$L = \{w \in \Sigma^* \mid w \text{ termina en } 11 \text{ o es igual a } \epsilon\}$$



Definición formal Autómata Finito Determinista

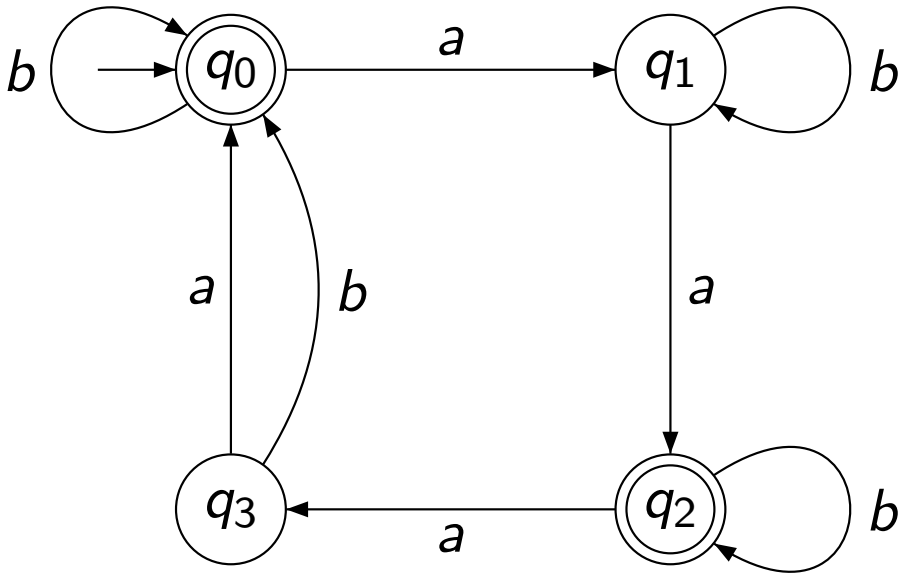
Definición

Un **Autómata Finito Determinista (AFD)** es una 5-tupla $M = (Q, \Sigma, \delta, q_0, F)$ tal que

- Q es el conjunto finito de estados.
- Σ es el alfabeto de entrada.
- $\delta : Q \times \Sigma \to Q$ es la **función de transición**. $\delta(q, a) = q'$ quiere decir que si estoy en el estado q y leo el símbolo a voy al estado q'.
- $q_0 \in Q$ es el **estado inicial**.
- $F \subseteq Q$ es el conjunto de los **estados finales** o de aceptación.

Representación AFD

- Lo más usual es la representación gráfica



Representación AFD

También podemos indicar quiénes son los estados, el alfabeto, la tabla de transiciones, el estado inicial y los estados finales.

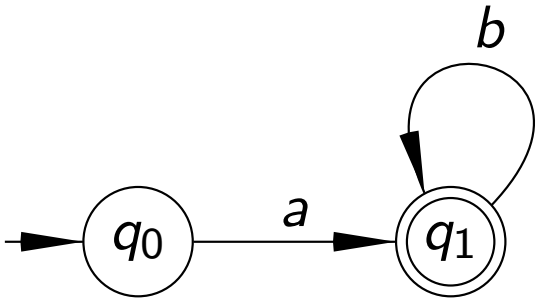
1. $Q = \{q_0, q_1, q_2, q_3\}$
2. $\Sigma = \{a, b\}$
3. La tabla de transiciones

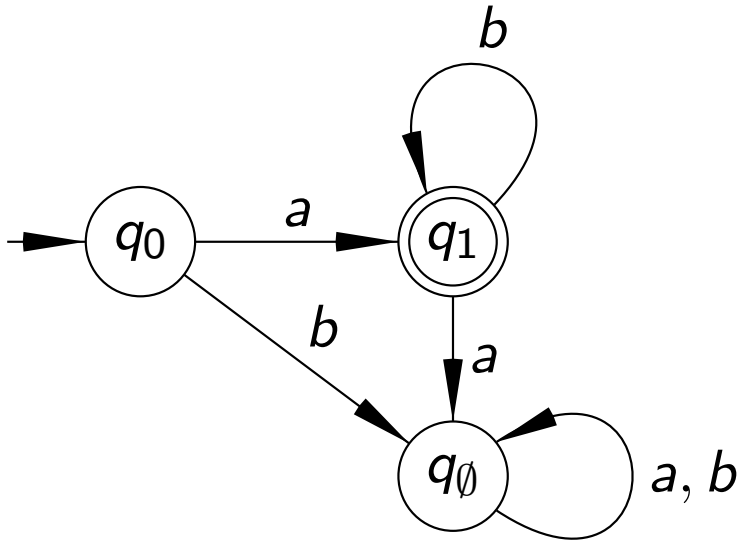
δ	a	b
q_0	q_1	q_0
q_1	q_2	q_1
q_2	q_3	q_2
q_3	q_0	q_0

4. Estado inicial q_0 (si se llama q_0 no hace falta decirlo)
5. $F = \{q_0, q_2\}$

Observaciones

En la definición de AFD es necesario que para cada estado estén definidas todas las transiciones posibles para cada símbolo del alfabeto. Para nosotros será equivalente





Definición formal computación

Definición

Un AFD $M = (Q, \Sigma, \delta, q_0, F)$ **acepta** una cadena $w = a_1 a_2 \cdots a_n$, $a_i \in \Sigma$, si existe una secuencia de estados $r_0, r_1, \ldots, r_n$ cumpliendo

1. $r_0 = q_0$
2. $\delta(r_i, a_{i+1}) = r_{i+1}$, $0 \leq i < n$
3. $r_n \in F$

Función transición extendida δ^*

Definición

Sea $M = (Q, \Sigma, \delta, q_0, F)$ un AFD. La función extendida de transición $\delta^* : Q x \Sigma^* \rightarrow Q$ se define recursivamente:

- $\delta^*(q, \epsilon) = q$
- Si $w \in \Sigma^*$, $w = w'a$,

$$\delta^*(q, w) = \delta(\delta^*(q, w'), a)$$

Intuitivamente, $\delta^*(q, w)$ es el estado al que llegamos partiendo del estado q y leyendo la cadena w.

Lenguaje reconocido por un AFD

Definición

Sea $M = (Q, \Sigma, \delta, q_0, F)$ un AFD, denotaremos por $L = L(M)$ al lenguaje

$$L(M) = \{w \mid M \text{ acepta } w\} = \{w \mid \delta^*(q_0, w) \in F\}$$

Diremos que el lenguaje L es **reconocido** por M.

Lenguajes regulares

Definición

Un lenguaje L es **regular** si es reconocido por algún AFD.

Ejemplos de lenguajes regulares

- Los números múltiplos de 3 en binario (o en cualquier otro sistema de numeración).
- Las cadenas que tienen un número par de aes.
- Las cadenas cuya quinta letra es una b.
- ¿Es Σ^* regular?
- ¿Es $L = \{\epsilon\}$ regular?
- ¿Es $L = \emptyset$ regular?

AFD equivalentes

- ¿Cuándo dos AFDs reconocen el mismo lenguaje?
- Si dos AFDs reconocen el mismo lenguaje, elegiremos el que menos estados tiene.
- Su implementación será más simple y eficaz.
- Buscamos entre todos los AFDs que reconocen el mismo lenguaje, aquellos que tengan el menor número de estados.

AFD mínimo

Definición

Un AFD es **mínimo** si tienen el menor número de estados entre todos los que reconocen el mismo lenguaje.

Teorema

Sea M y M' dos AFDs que reconocen el mismo lenguaje L y son mínimos. Entonces $M = M'$.

- Por tanto, el AFD mínimo que reconoce un lenguaje es único.

Minimización de un AFD

Definición

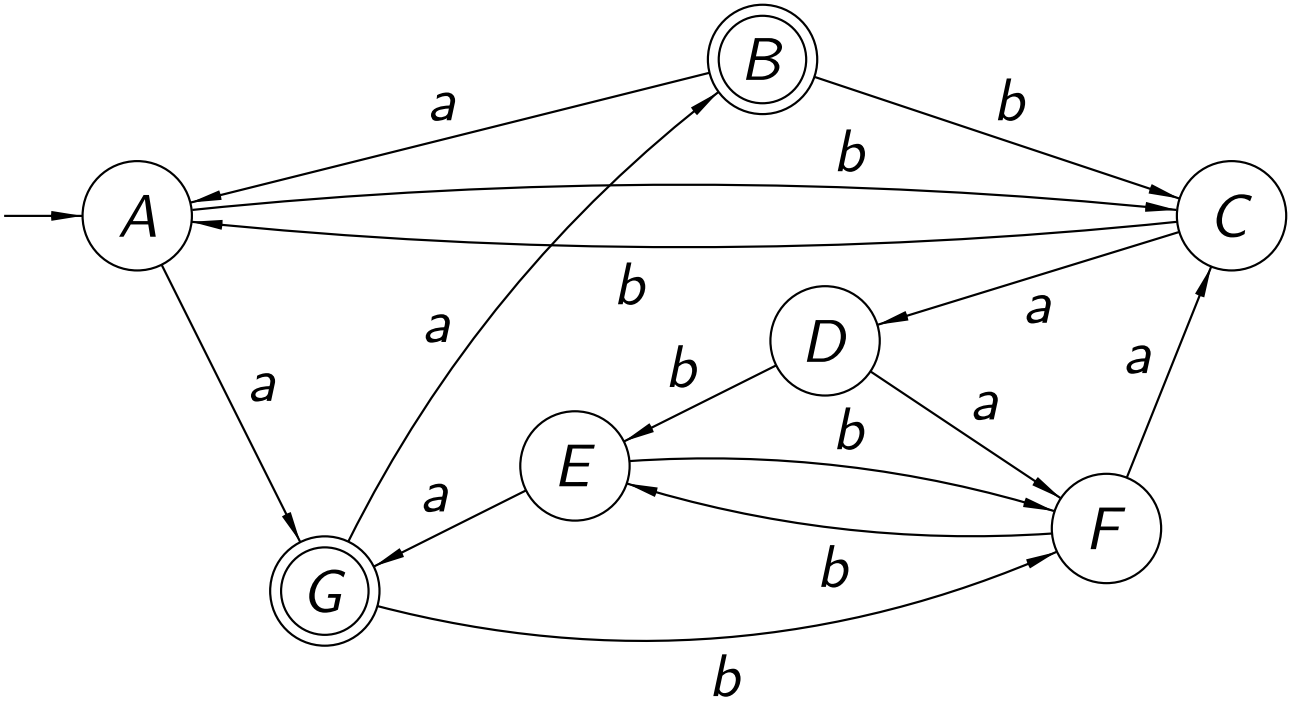
Dos estados q, q' de un AFD M se dicen **indistinguibles** si para toda cadena w, se cumple:

- $\delta(q, w) \in F$ y $\delta(q', w) \in F$
- $\delta(q, w) \notin F$ y $\delta(q', w) \notin F$

Los estados se dicen **distinguibles**, en caso contrario.

- En la minimización de un AFD buscaremos estados indistinguibles.
- Los estados indistinguibles se convertirán en un único estado en el AFD mínimo.

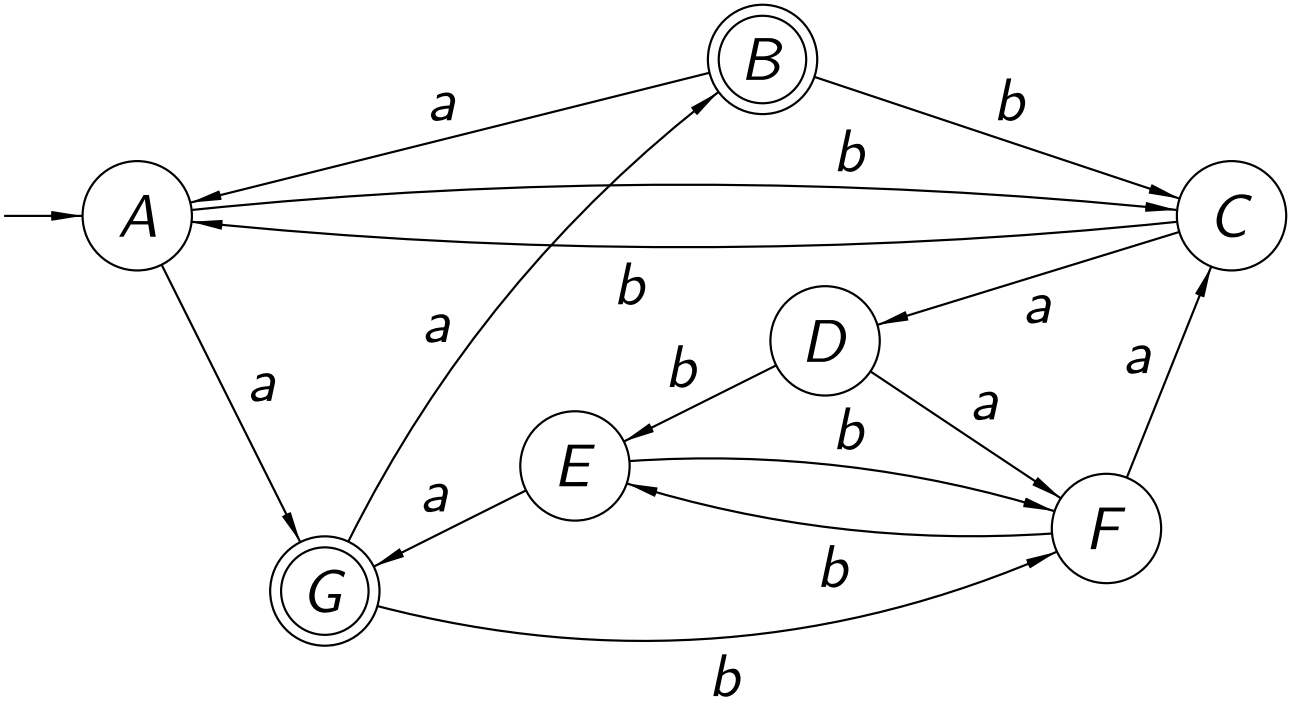
Minimización de un AFD: proceso



B	0					
C		0				
D		0				
E		0				
F		0				
G	0		0	0	0	0
	A	B	C	D	E	F

- **Paso 0**: se marcan distinguibles estados finales y no finales.
- Distinguibles mediante cadena ϵ

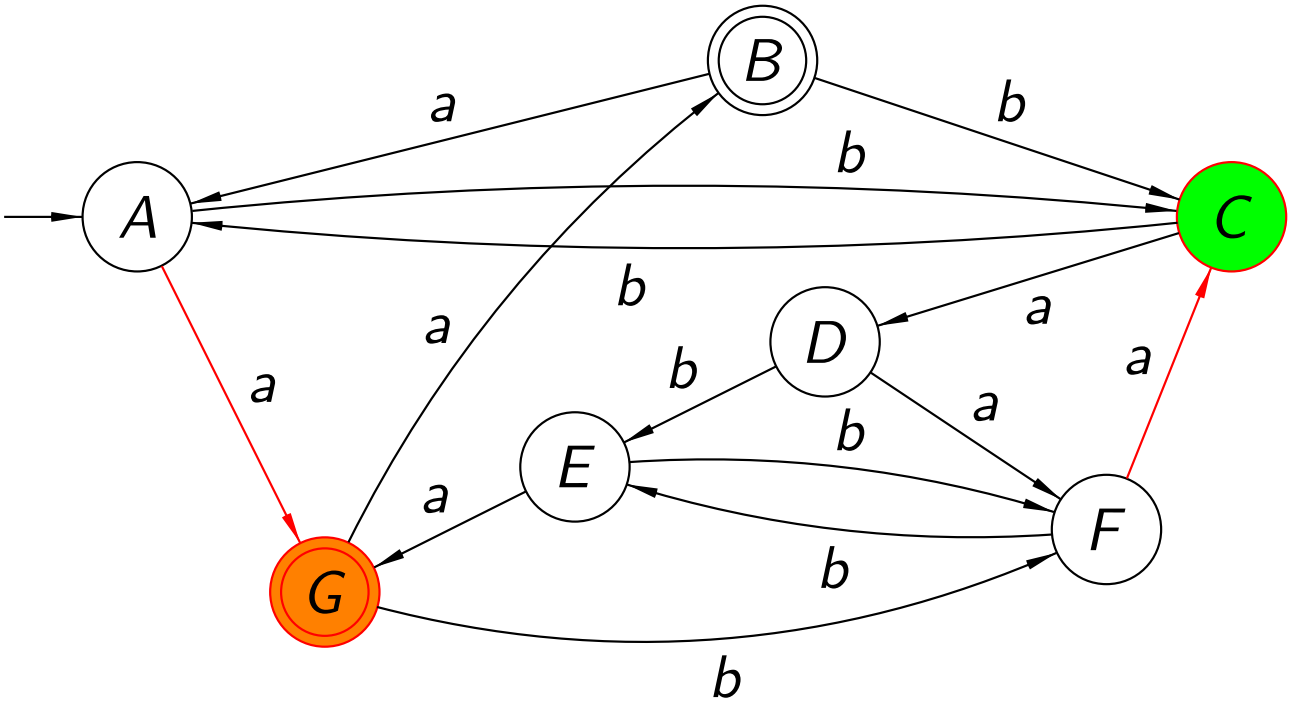
Minimización de un AFD: proceso



B	0					
C		0				
D		0				
E		0				
F		0				
G	0		0	0	0	0
	A	B	C	D	E	F

- **Paso 1**: se marcan distinguibles si existe transición que lleve a estados marcados como distinguibles en el paso anterior.
- Distinguibles mediante cadenas longitud 1

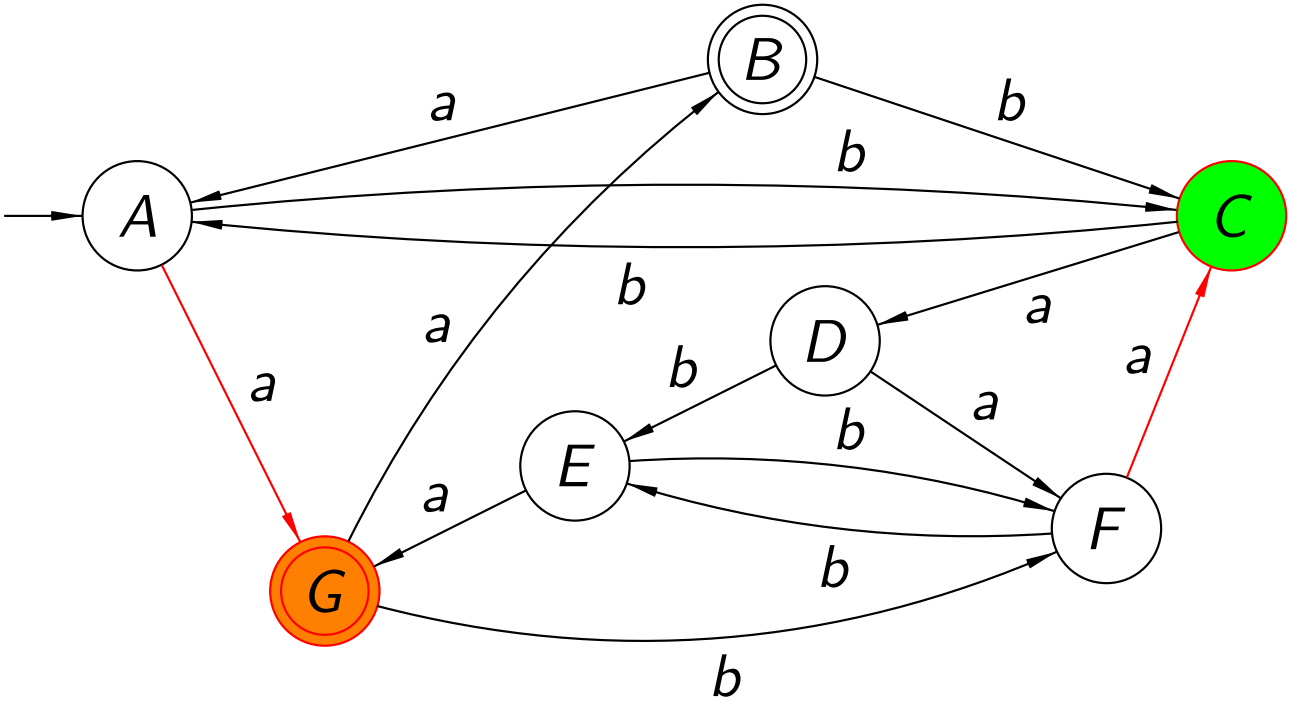
Minimización de un AFD: proceso



B	0					
C		0				
D		0				
E		0				
F		0				
G	0		0	0	0	0
	A	B	C	D	E	F

- **Paso 1**: se marcan distinguibles si existe transición que lleve a estados marcados como distinguibles en el paso anterior.
- Distinguibles mediante cadenas longitud 1

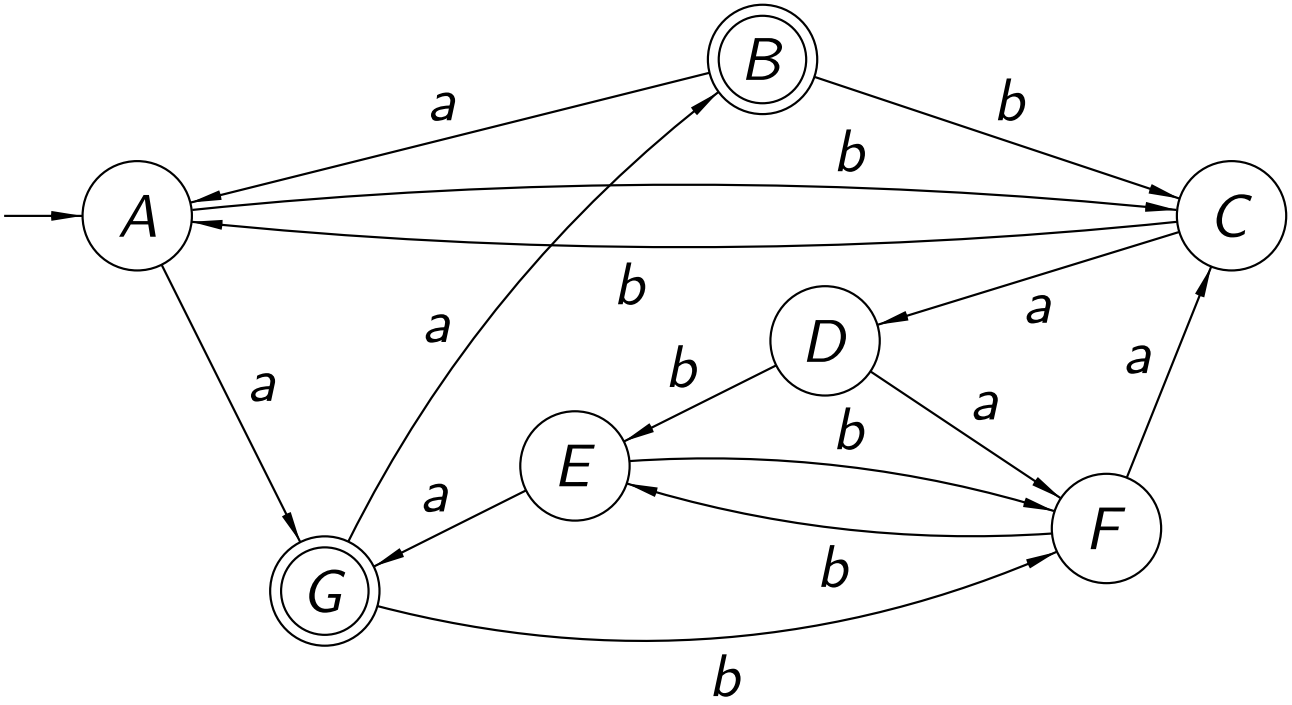
Minimización de un AFD: proceso



B	0					
C		0				
D		0				
E		0				
F	1	0				
G	0		0	0	0	0
	A	B	C	D	E	F

- **Paso 1**: se marcan distinguibles si existe transición que lleve a estados marcados como distinguibles en el paso anterior.
- Distinguibles mediante cadenas longitud 1

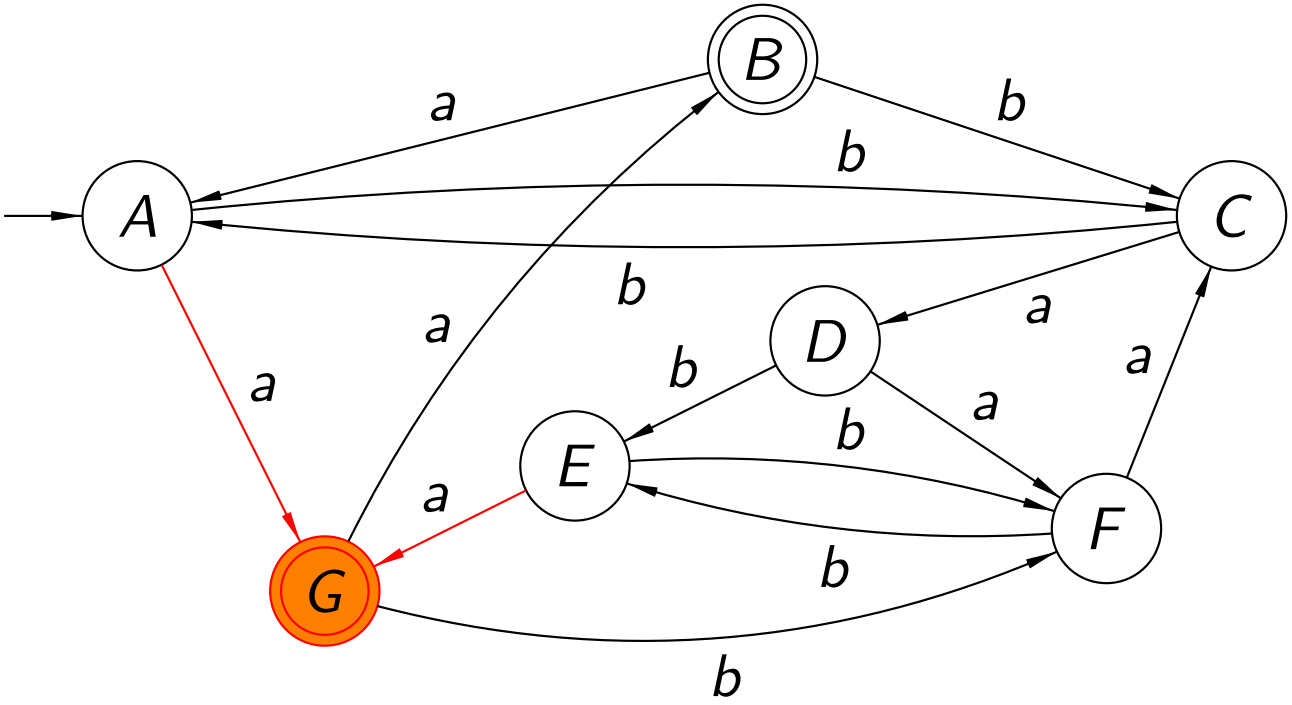
Minimización de un AFD: proceso



B	0					
C		0				
D		0				
E		0				
F	1	0				
G	0		0	0	0	0
	A	B	C	D	E	F

- **Paso 1**: se marcan distinguibles si existe transición que lleve a estados marcados como distinguibles en el paso anterior.
- Distinguibles mediante cadenas longitud 1

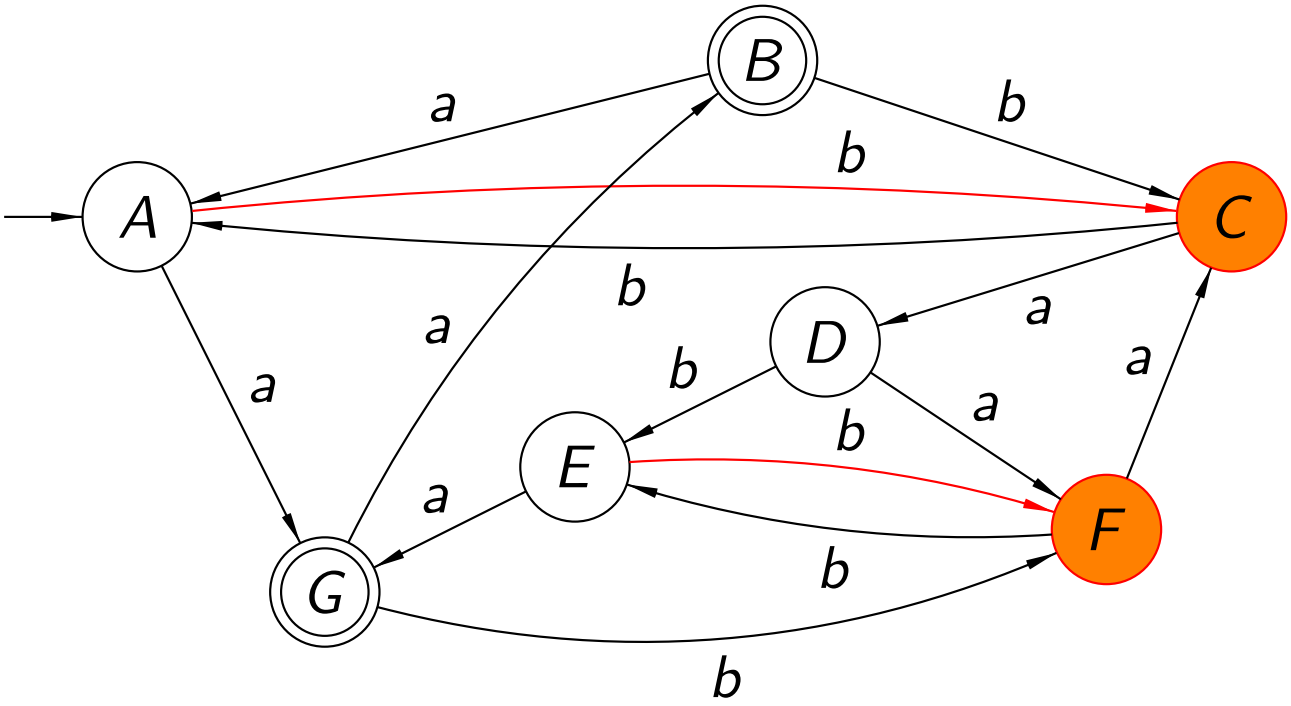
Minimización de un AFD: proceso



B	0					
C		0				
D		0				
E		0				
F	1	0				
G	0		0	0	0	0
	A	B	C	D	E	F

- **Paso 1**: se marcan distinguibles si existe transición que lleve a estados marcados como distinguibles en el paso anterior.
- Distinguibles mediante cadenas longitud 1

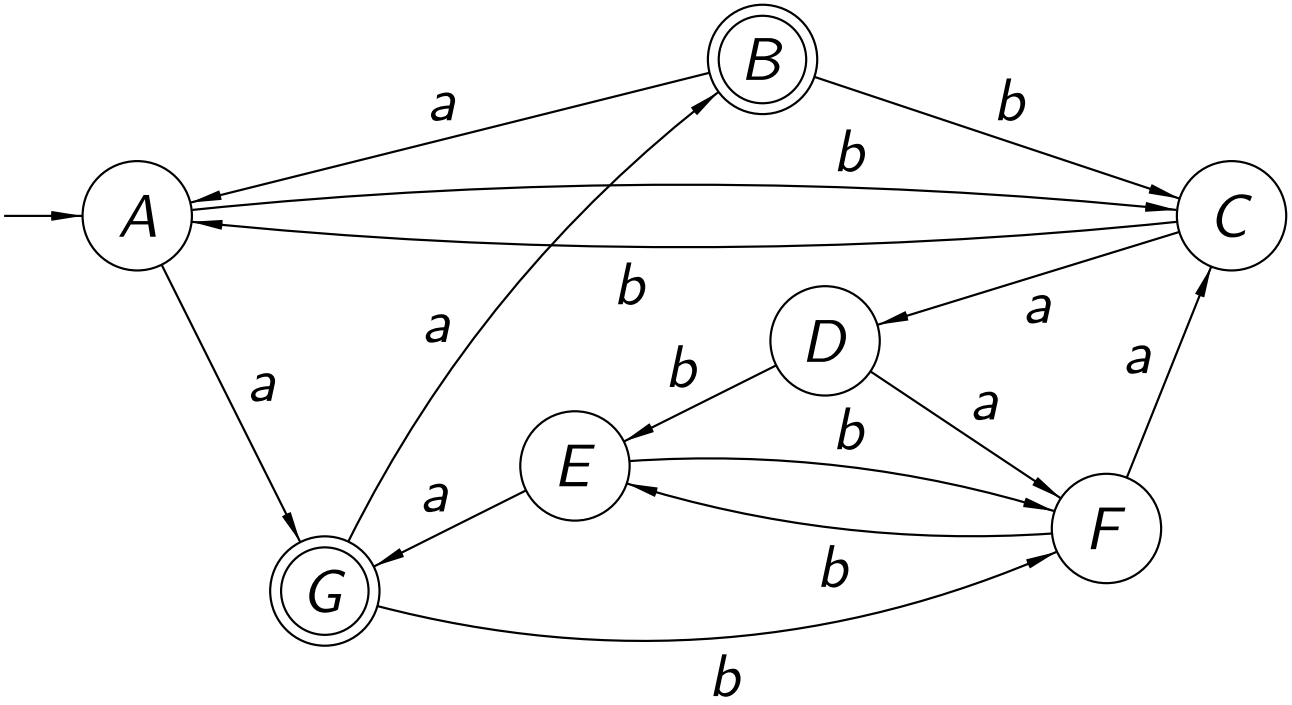
Minimización de un AFD: proceso



B	0					
C		0				
D		0				
E		0				
F	1	0				
G	0		0	0	0	0
	A	B	C	D	E	F

- **Paso 1**: se marcan distinguibles si existe transición que lleve a estados marcados como distinguibles en el paso anterior.
- Distinguibles mediante cadenas longitud 1

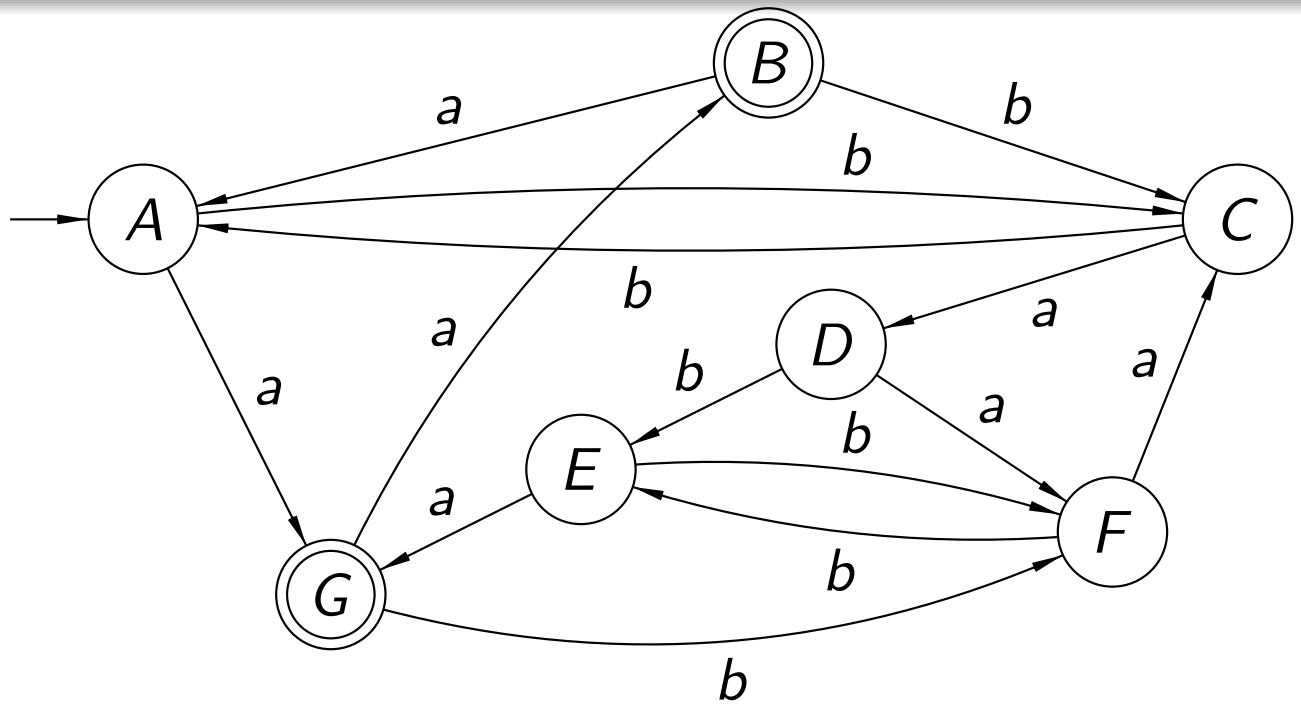
Minimización de un AFD: proceso



B	0					
C	1	0				
D	1	0				
E		0	1	1		
F	1	0			1	
G	0	1	0	0	0	0
	A	B	C	D	E	F

- **Paso 1**: se marcan distinguibles si existe transición que lleve a estados marcados como distinguibles en el paso anterior.
- Distinguibles mediante cadenas longitud 1

Minimización de un AFD: proceso



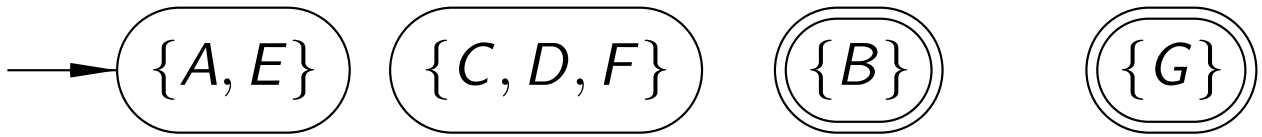
B	0					
C	1	0				
D	1	0				
E		0	1	1		
F	1	0			1	
G	0	1	0	0	0	0
	A	B	C	D	E	F

- **Paso 2**: se marcan distinguibles si existe transición que lleve a estados marcados como distinguibles en el paso anterior.
- Distinguibles con cadenas longitud 2. No aparecen nuevas marcas.
- **Fin de minimización**.

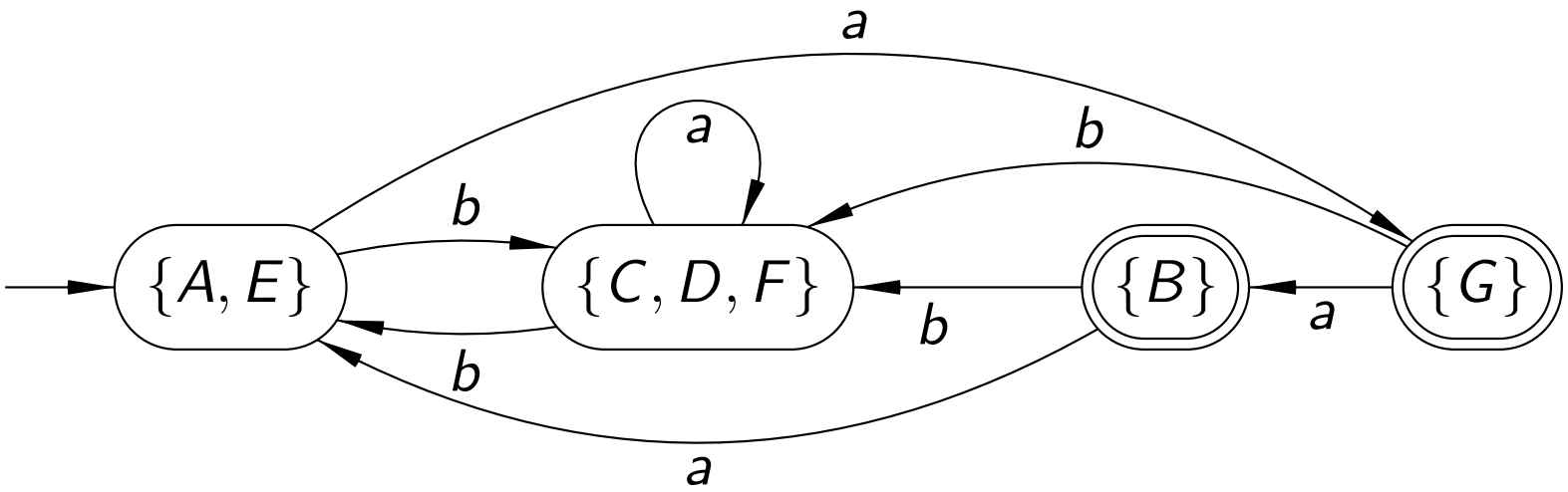
Minimización de un AFD: estado inicial y finales

B	0					
C	1	0				
D	1	0				
E		0	1	1		
F	1	0			1	
G	0	1	0	0	0	0
	A	B	C	D	E	F

- Los huecos en la tabla se corresponden con estados indistinguibles. El resto son distinguibles.
- Estados en AFD mínimo: $\{A, E\}$, $\{C, D, F\}$, $\{B\}$, $\{G\}$
- Estado inicial de AFD mínimo: el que contenga el estado inicial del original.
- Estados finales: los que estén formados por estados finales en el original.



Minimización AFD: función de transición



- Función transición δ' de AFD mínimo: si R, R' son estados del AFD mínimo, hay transición de R a R' por el símbolo a,

$$\delta'(R, a) = R', \text{ si } \delta(R, a) \in R'$$

Conclusión

- ¿Cómo saber si dos AFDs reconocen el mismo lenguaje (son equivalentes)?
- Minimizamos los dos AFDs y comprobamos si son iguales.

Implementación de un AFD

- ¿Cómo implementar en un lenguaje de programación un AFD?

```
int estado = 0;
while ((c = leerSig()) != FIN) {
    switch(estado) {
    case 0:
        if (c == 'a') then estado = 1; else estado = 3;
    case 1:
        if (c == 'a') then estado = 0; else estado = 2;
     ...
    }
}
//si estado final
if (estado == 2 || estado == 4 || ...)
    return true;
else
    return false;
```

Implementación de un AFD

- ¿Cuántas veces se ejecutará el bucle while?
- Tantas veces como símbolos tenga la entrada.
- Esto es, es lineal en el tamaño de la entrada.
- ¿Siempre obtendremos un buen algoritmo con un AFD?

Implementación de un AFD

- Hacer un programa que tenga como entrada una cadena de caracteres y nos devuelva cierto si el carácter 100 antes del final de la cadena es una 'a'.
- El lenguaje L es regular

$$L = \{w \in \Sigma^* \,|w \text{ tal que símbolo en posición 100 antes del final es } a\}$$

- Existe un AFD que reconoce L.
- ¿Cuántos estados tendrá el mínimo AFD?

Contar estados

- Debe tener al menos 2^{100} estados.
- Supongamos que tiene menos estados.
- El número de cadenas con 100 caracteres (suponiendo que el alfabeto es $\Sigma = \{a, b\}$) es 2^{100}.
- Como el AFD tiene menos de 2^{100} estados, hay dos cadenas $w_1 \neq w_2$ de longitud 100, que al finalizar la computación acaban en el mismo estado.

Contar estados

- Como w_1 y w_2 son distintas, tienen un símbolo diferente. Por ejemplo, el quinto.
- $w_1 = ababa....$, $w_2 = ababb...$
- $w_1' = w_1aaaa$: símbolo en la posición 100 antes del final es a
- $w_2' = w_2aaaa$: símbolo en la posición 100 antes del final es b
- Si el AFD reconoce L, w_1' debe ser aceptada y w_2' rechazada.
- Al acabar w_1 y w_2 están en el mismo estado, por tanto, al acabar w_1' y w_2' están en el mismo estado.
- Contradicción por suponer que el AFD tiene menos de 2^{100} estados.

Conclusión

- Por tanto, el programa que implementase el AFD debería tener al menos 2^{100} estados (tantos como partículas en el universo).
- Completamente inviable!!
- Aunque un problema se pueda resolver con un AFD, no siempre es eficaz (la mayoría de las veces, sí).
- Existen herramientas que generan automáticamente el código de un AFD.
- En práctica utilizaréis Flex para generar el código de los autómatas en C.

Lenguajes no regulares

- No todos los lenguajes son regulares.
- En los siguientes temas veremos ejemplos de lenguajes no regulares.
- Intenta crear un AFD que reconozca $L = \{a^n b^n \mid n \in \mathbb{N}\}$

Referencias

- Kelly: capítulo 2, secciones 2.3-2.4
- Minimización: apuntes de la profesora Elvira Mayordomo
- Sipser: capítulo 1, sección 1.1