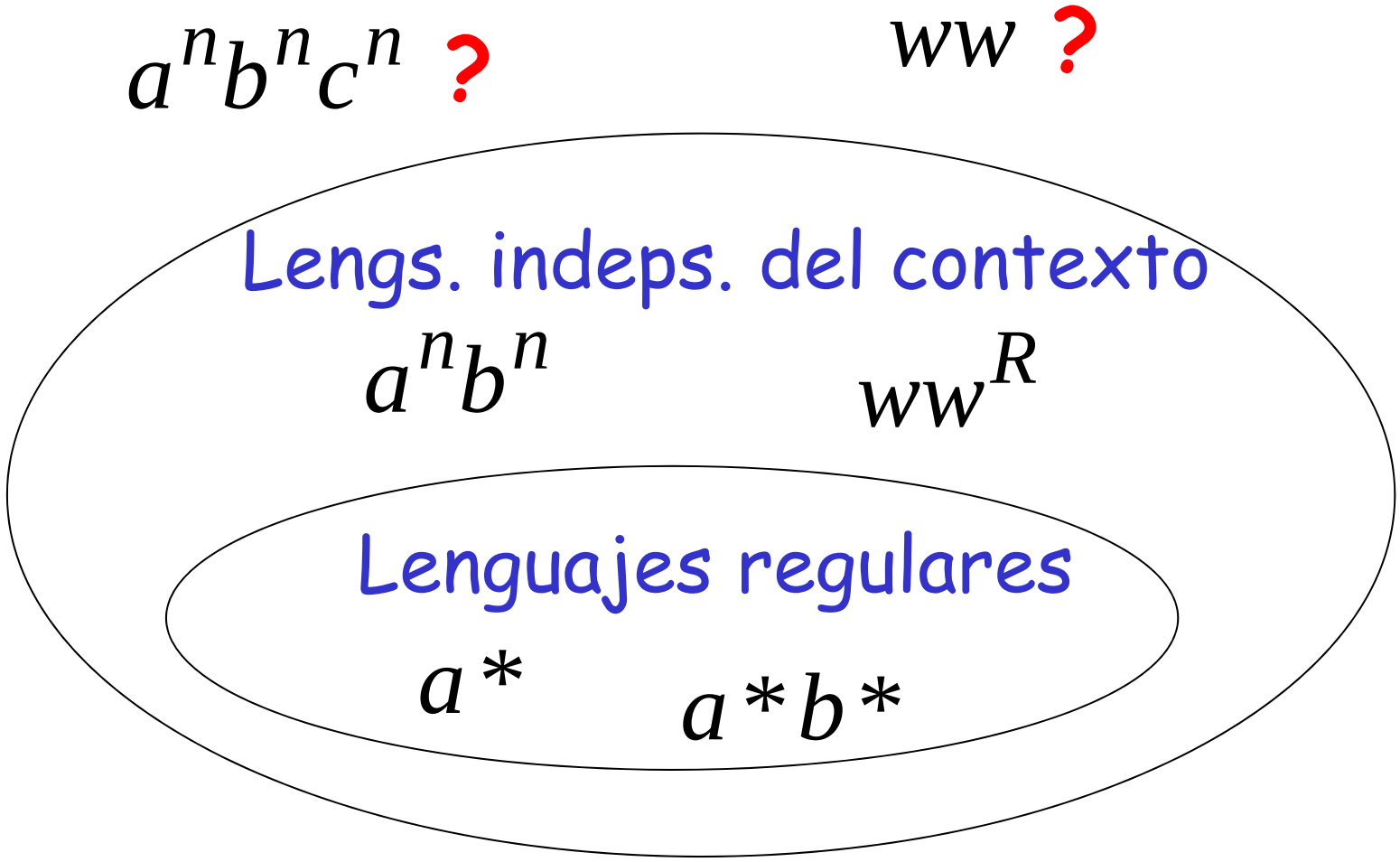
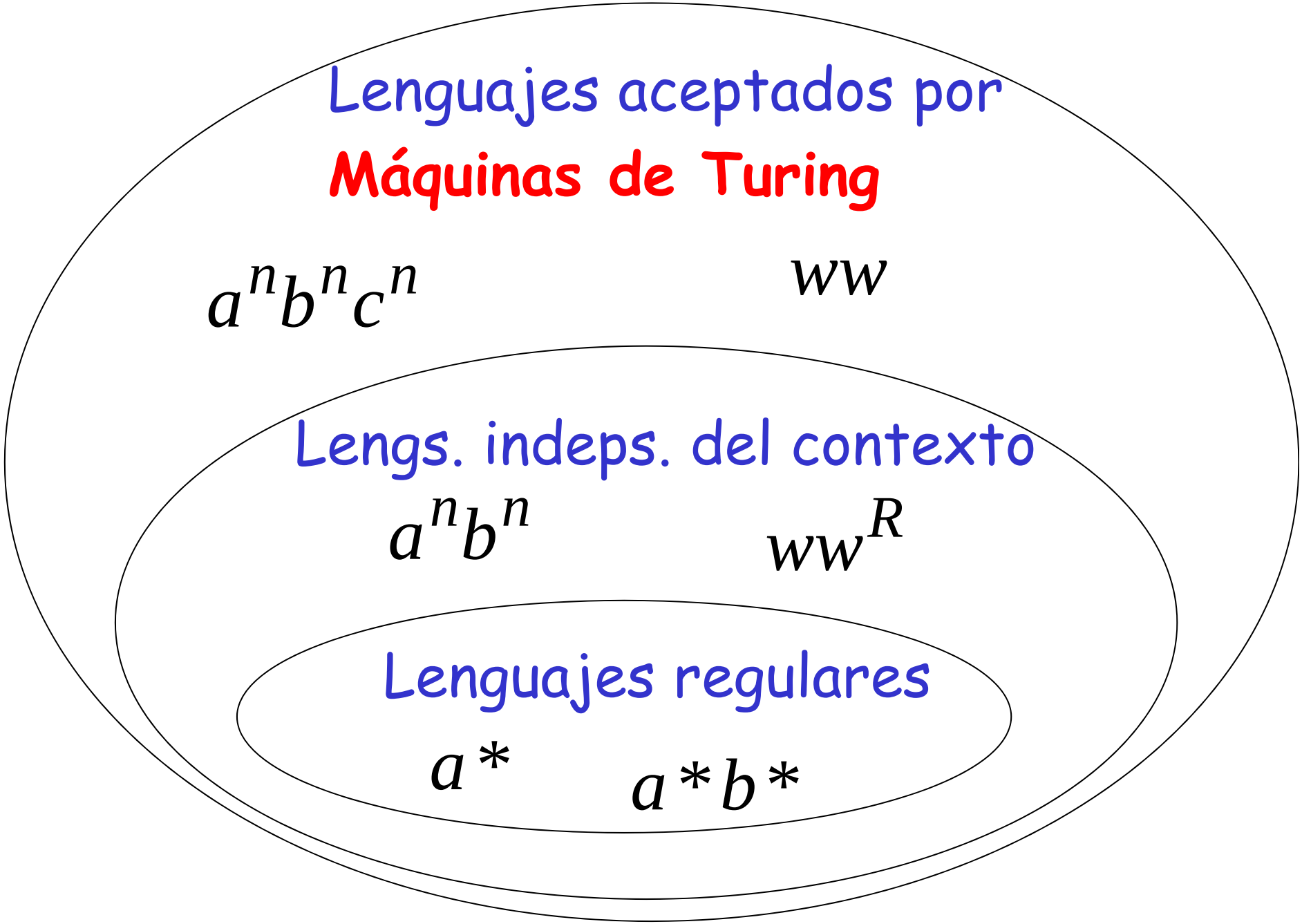
Máquinas de Turing, programas y tesis de Turing-Church

Elvira Mayordomo, Universidad de Zaragoza
Ilustraciones:
Costas Busch, Rensselaer Polytechnic Institute

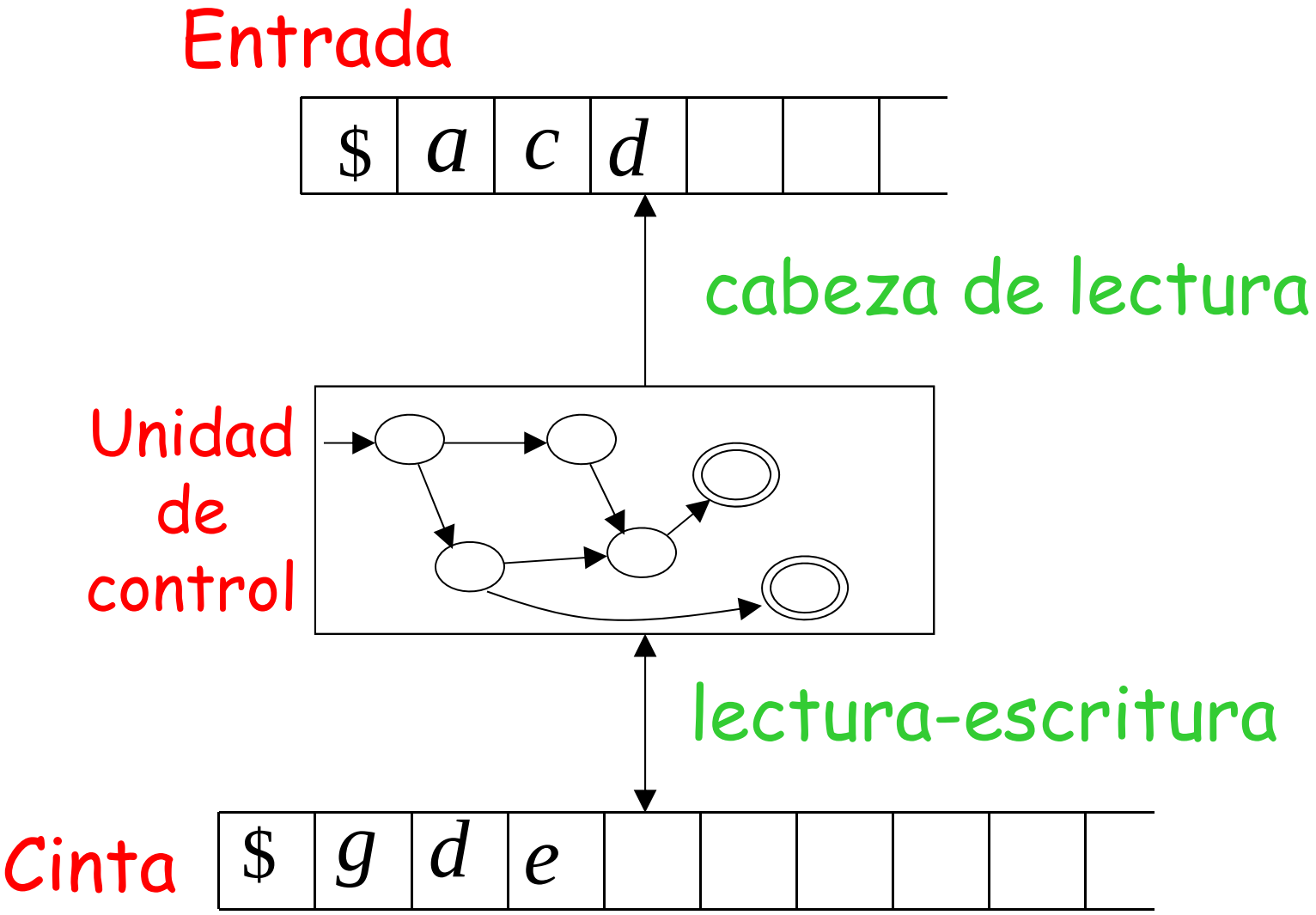
Máquinas de Turing

La jerarquía de lenguajes



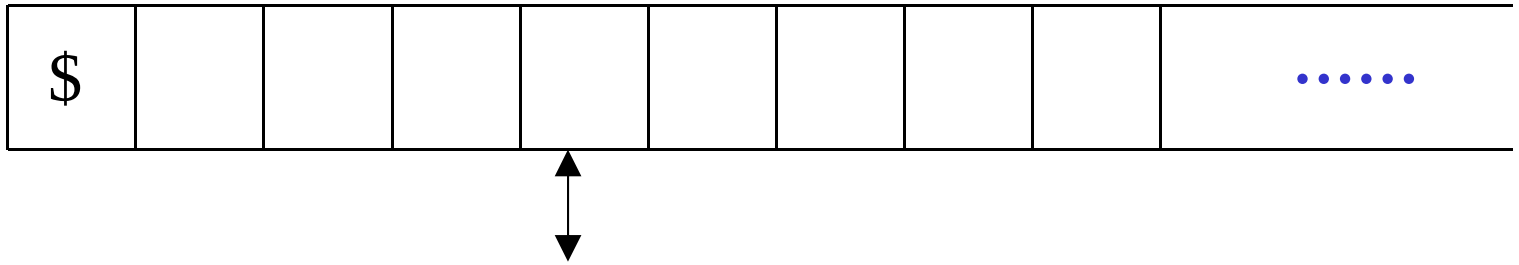

Lenguajes aceptados por
Máquinas de Turing
$a^n b^n c^n$
ww
Lengs. indeps. del contexto
$a^n b^n$
ww^R
Lenguajes regulares
$a*$
$a*b*$

Una máquina de Turing



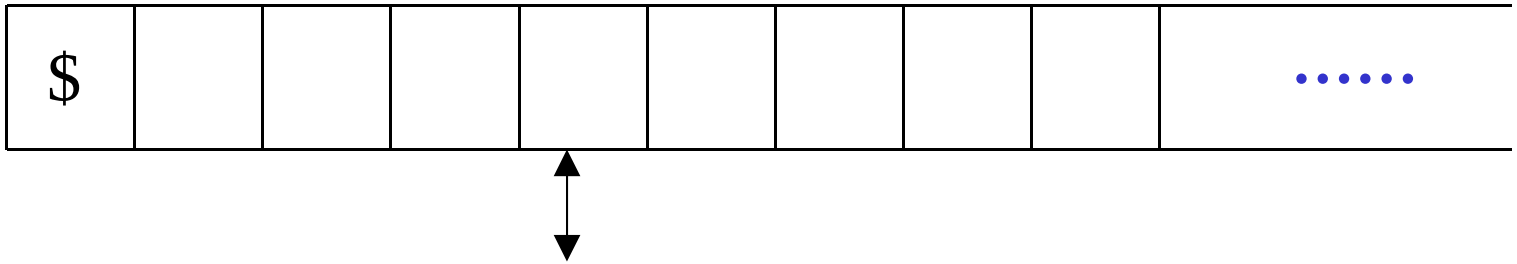
La cinta de memoria

Sin límite derecho – longitud infinita



cabeza de lectura-escritura

La cabeza se mueve a la Derecha o a la Izquierda o No se mueve



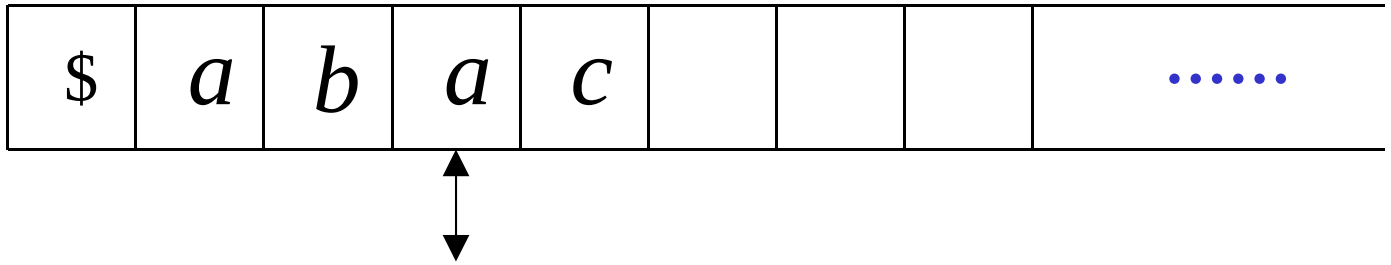
cabeza de lectura-escritura

La cabeza en cada paso de ejecución:

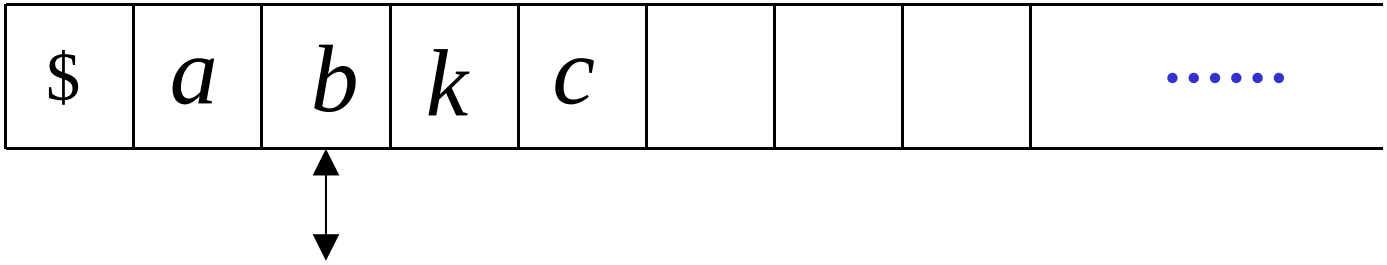
1. Lee un símbolo
2. Escribe un símbolo
3. Se mueve a Dcha o Izda o
   No se mueve

Ejemplo:

Tiempo 0

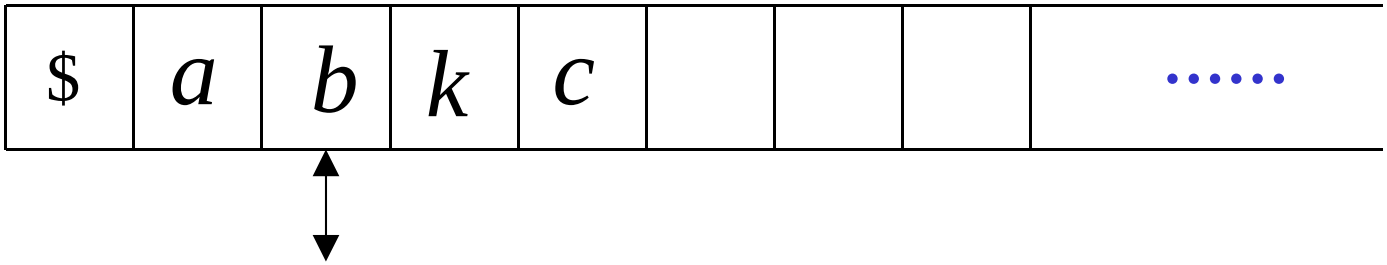


Tiempo 1

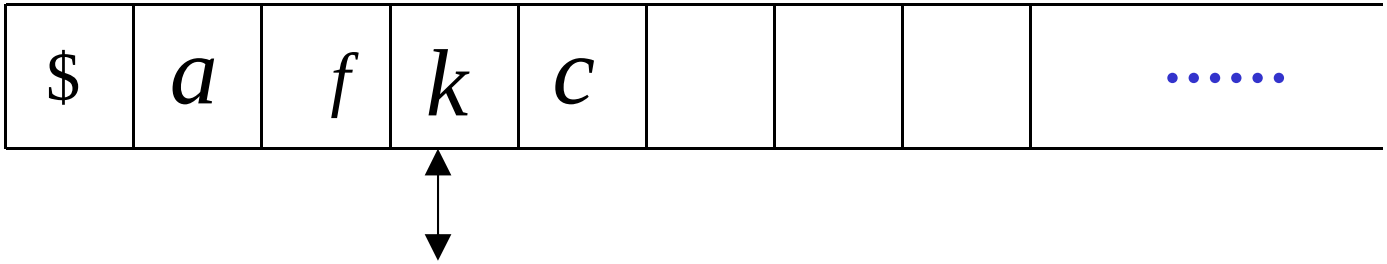


1. Lee a
2. Escribe k
3. Se mueve a la Izda

Tiempo 1

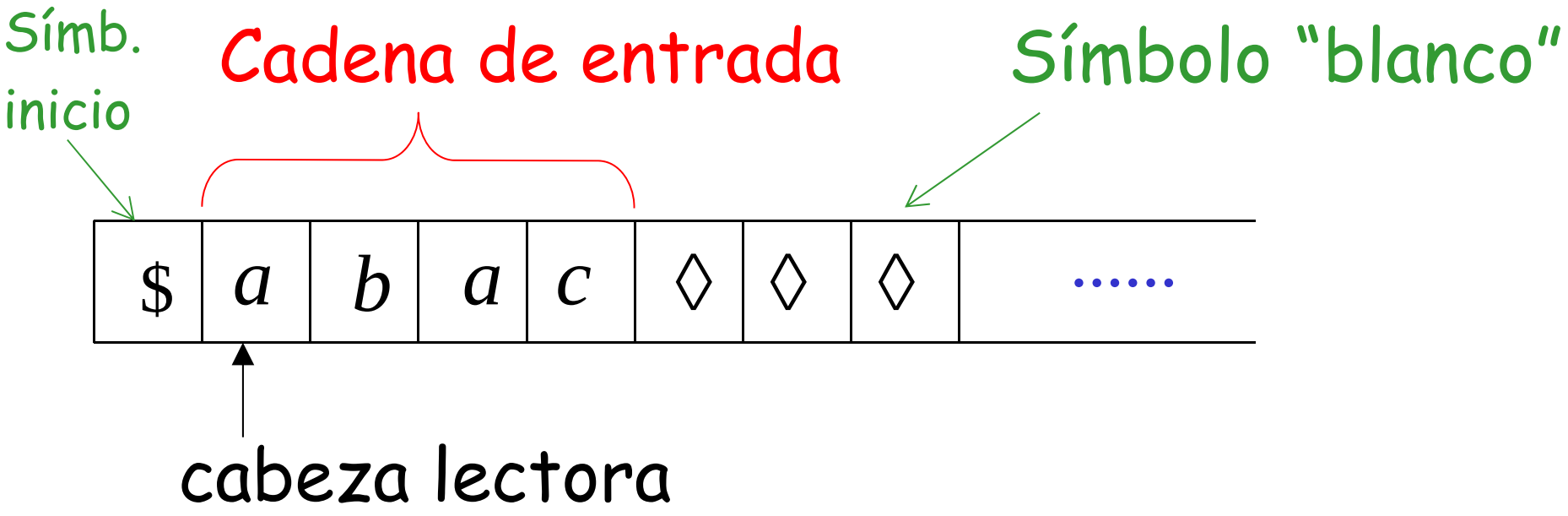


Tiempo 2



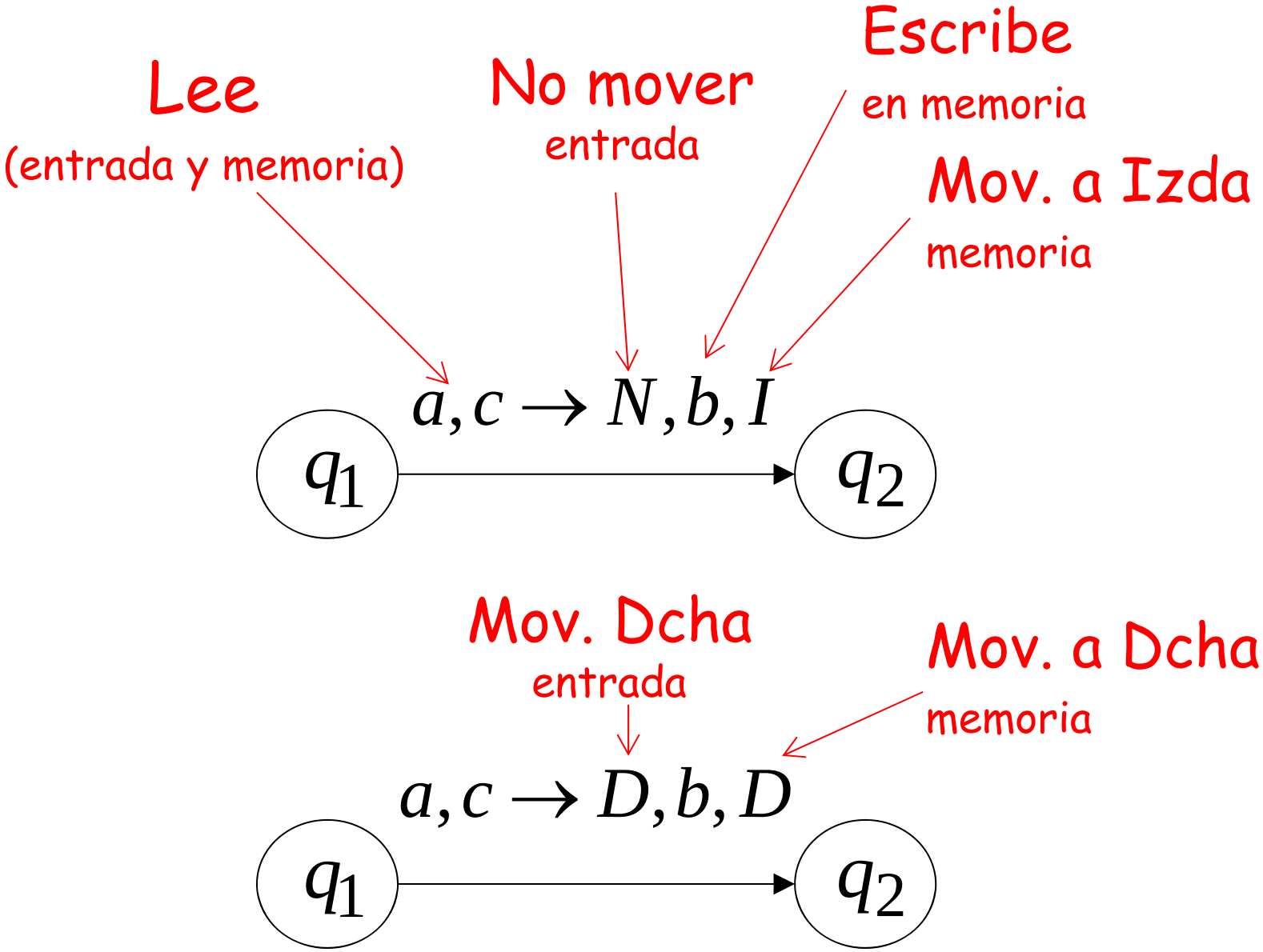
1. Lee b
2. Escribe f
3. Se mueve a la Dcha

La cadena de entrada



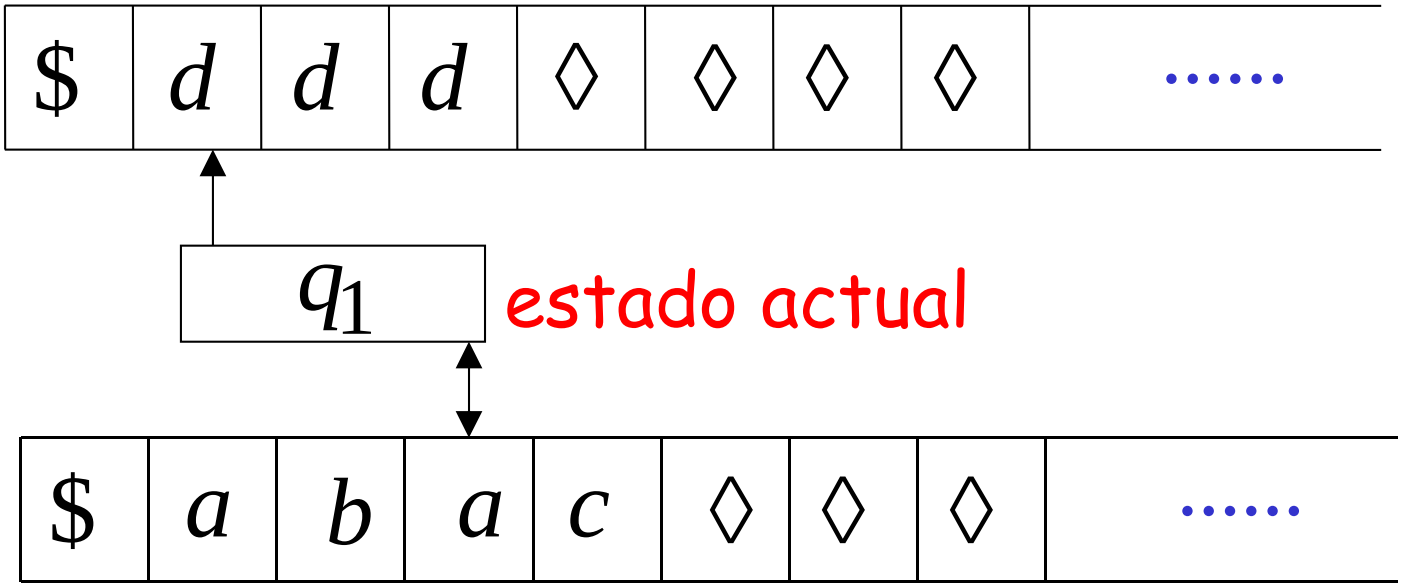
La cabeza empieza en la posición más a la izda de la cadena de entrada

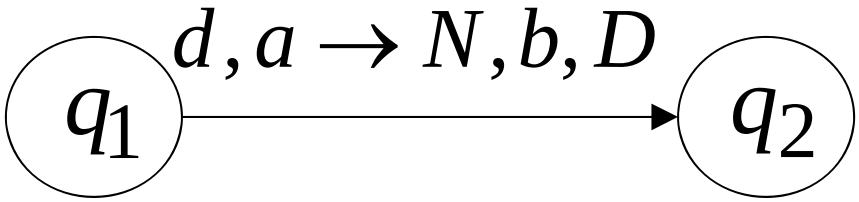
Estados y transiciones



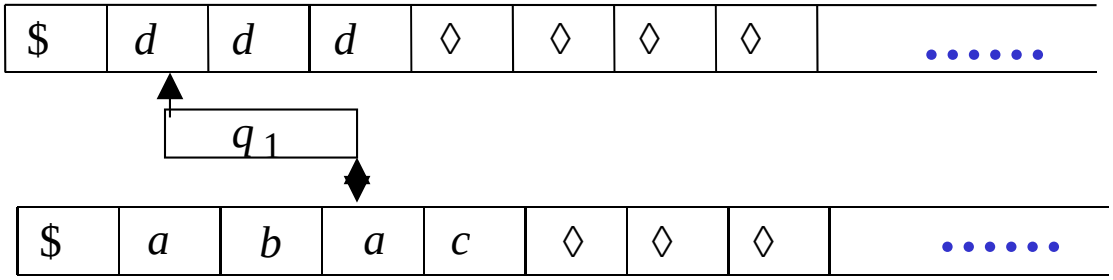
Ejemplo:

Tiempo 1

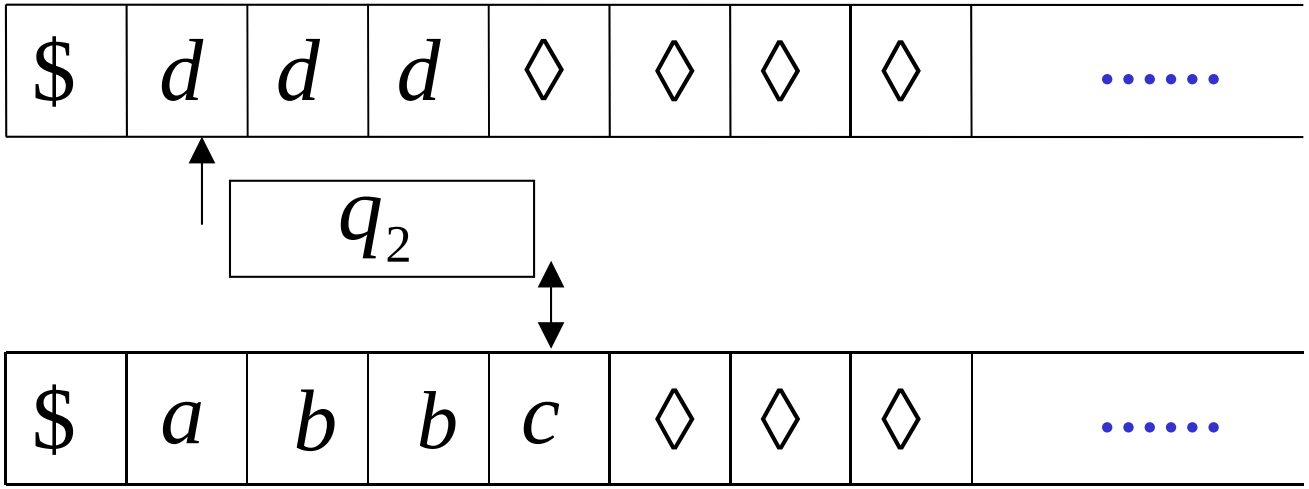


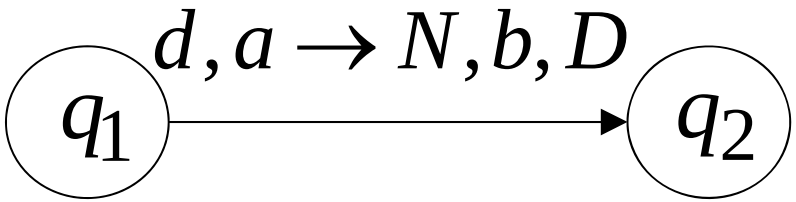


Tiempo 1

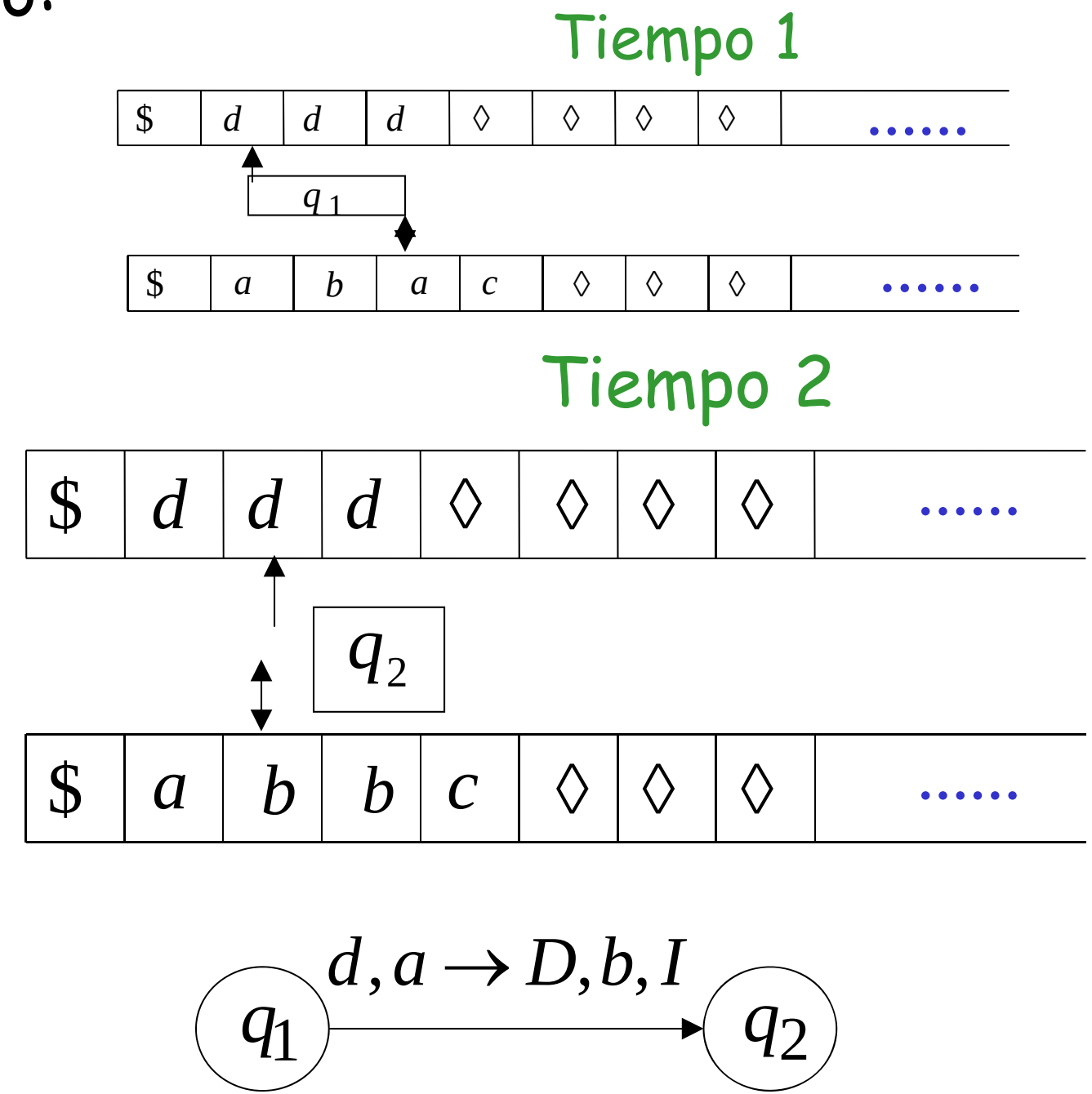


Tiempo 2

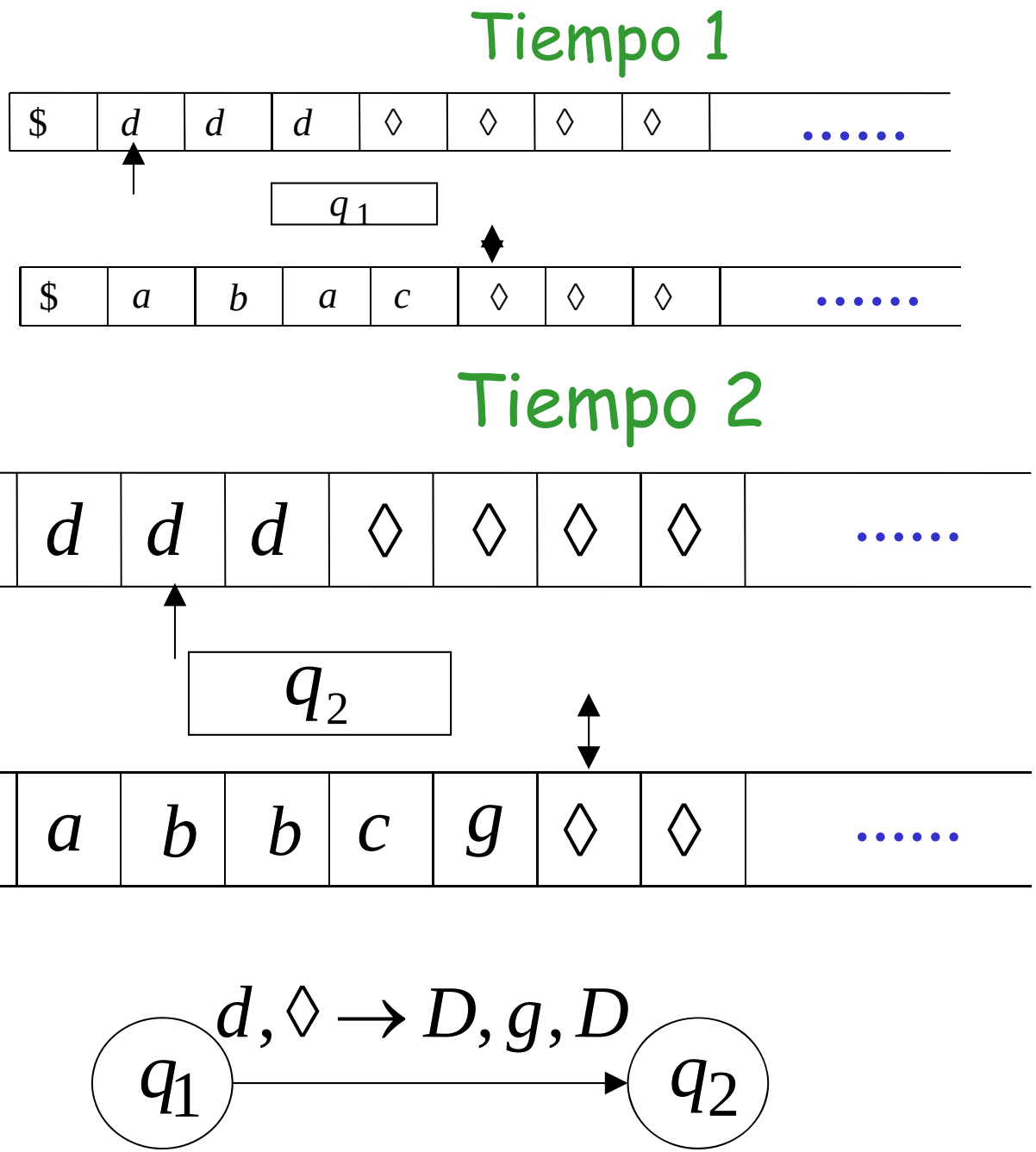




Ejemplo:

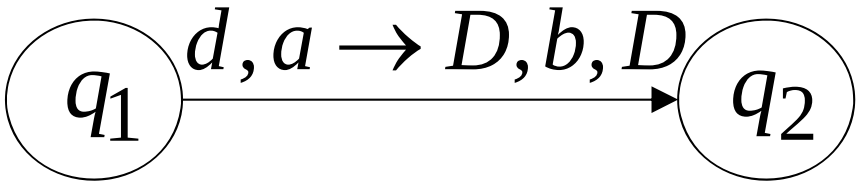


Ejemplo:


Tiempo 1
$ d d d ◊ ◊ ◊ ◊ .....
q_1
$ a b a c ◊ ◊ ◊ .....
Tiempo 2
$ d d d ◊ ◊ ◊ ◊ .....
q_2
$ a b b c g ◊ ◊ .....
$d, \Diamond \rightarrow D, g, D$
q_1
q_2

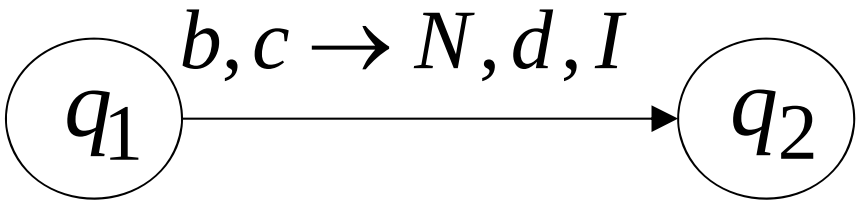
Definición formal de máquinas deTuring

Función de transición



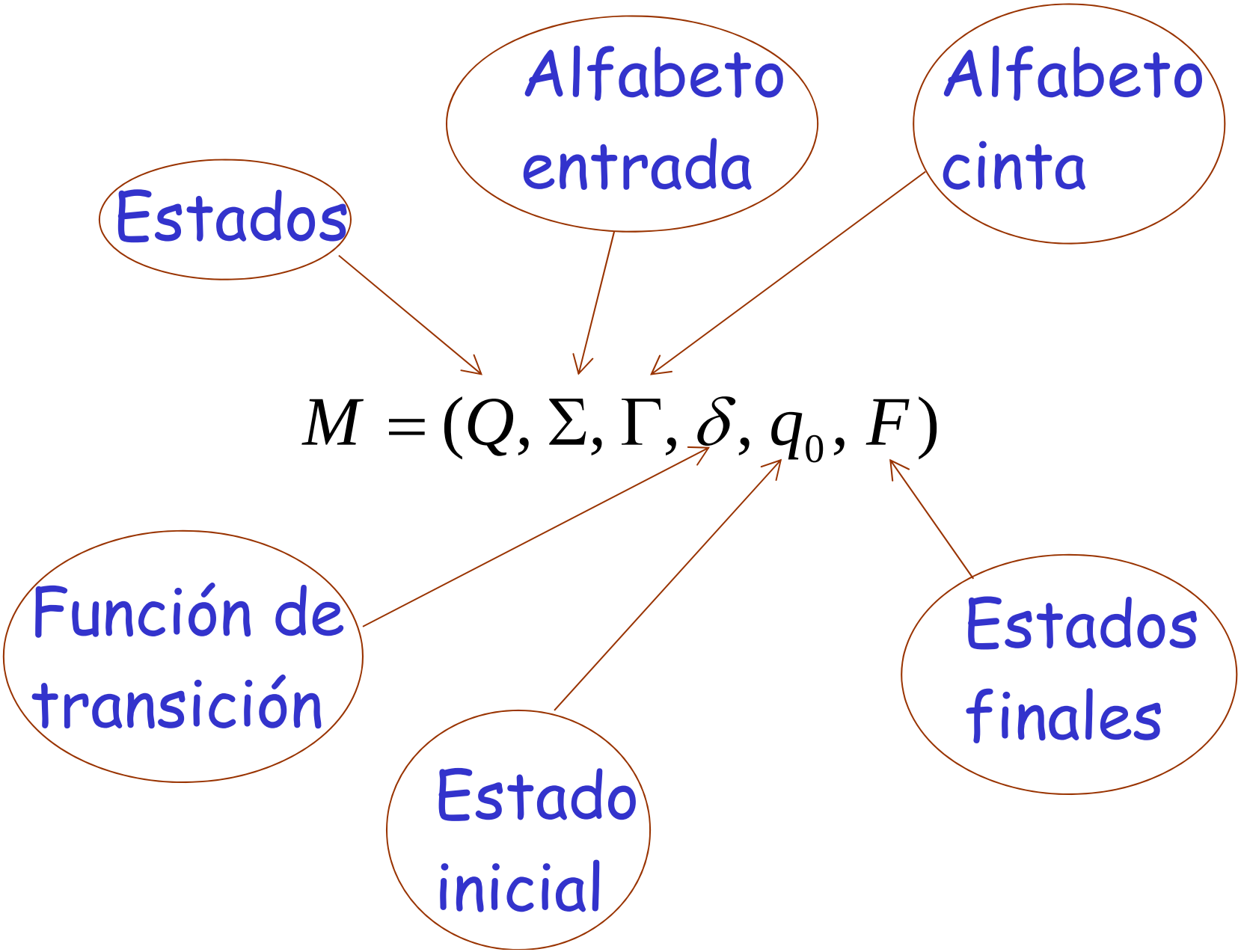
$$\delta(q_1, d, a) = (q_2, D, b, D)$$

Función de transición



$$\delta(q_1, b, c) = (q_2, N, d, I)$$

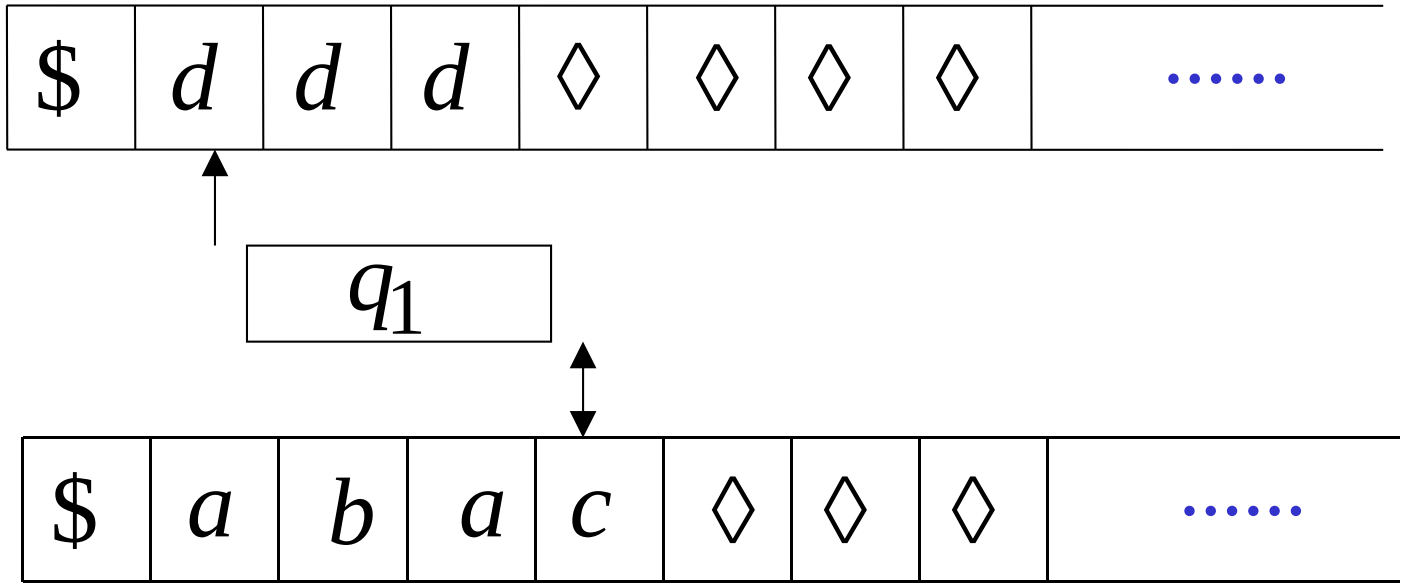
Máquina de Turing:

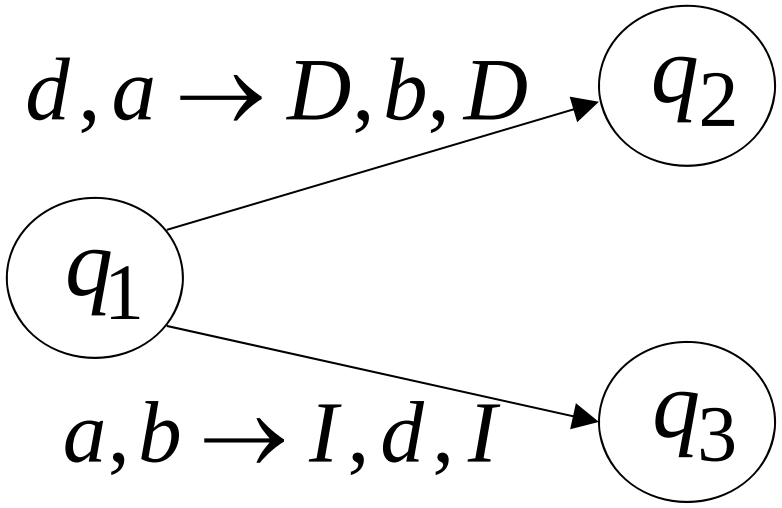


Parar

La máquina ***para*** si no hay transición posible desde la configuración actual

Ejemplo:

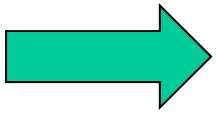


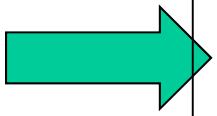


No hay transición posible

¡PARA!!!

Aceptar

Aceptar Entrada  Si la máquina para en un estado final

Rechazar Entrada  Si la máquina para en un estado no final

o

Si la máquina no para nunca

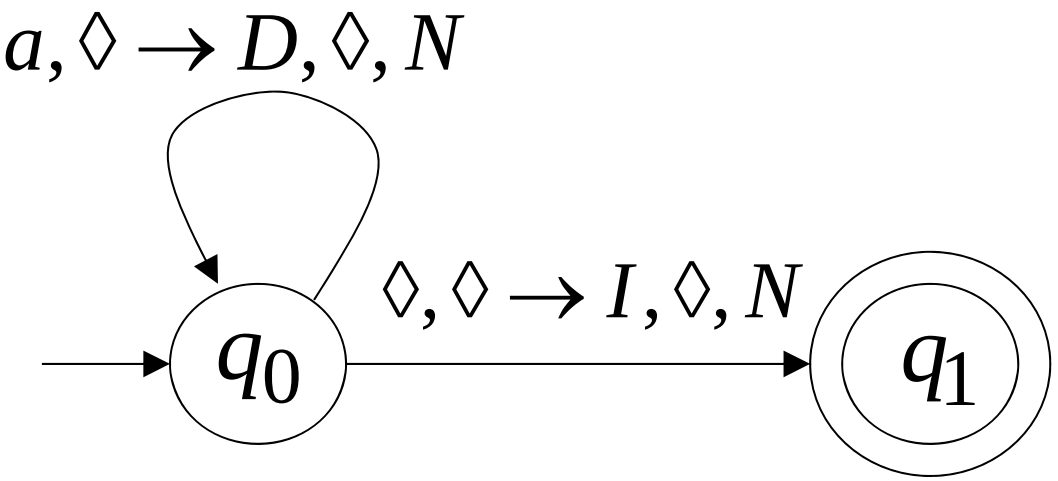
El lenguaje aceptado

Para una máquina de Turing M

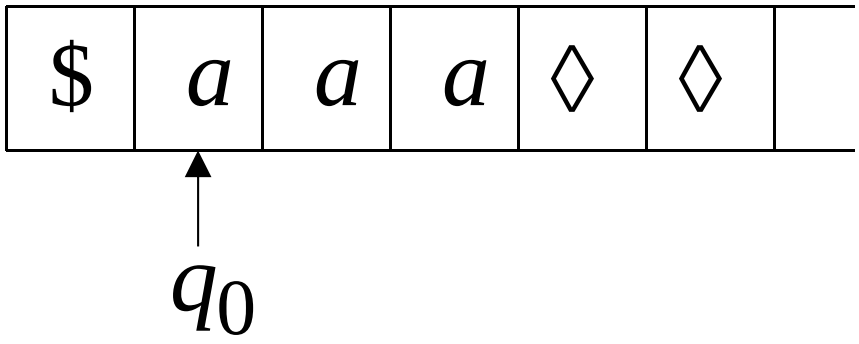
$$L(M) = \left\{ w : \begin{array}{l} \text{desde el estado } q_0, \text{ con } w \text{ en la entrada,} \\ \text{la máquina para en un estado final} \end{array} \right\}$$

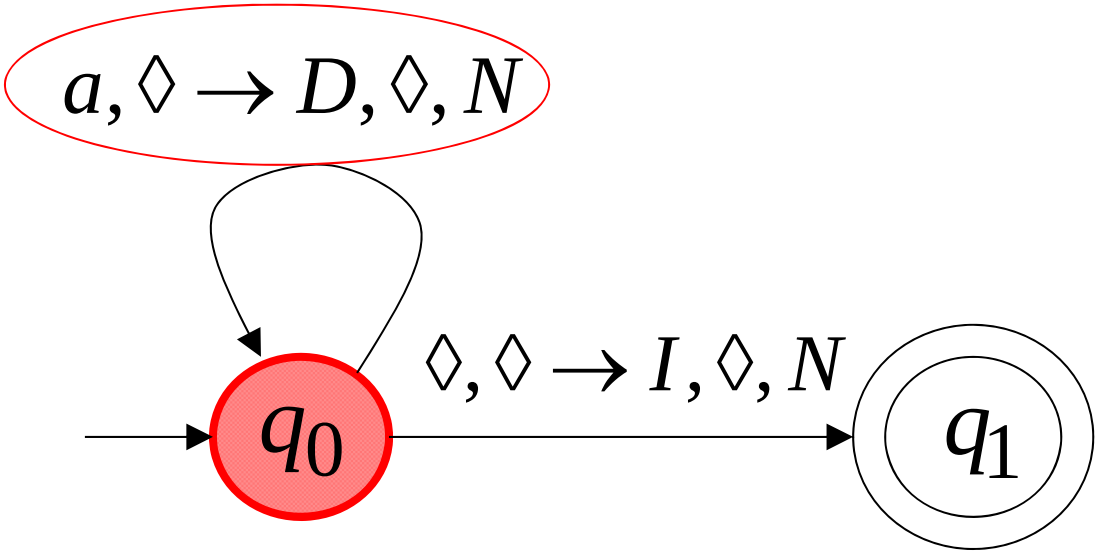
Un ejemplo de máquina de Turing

Una máquina de Turing que acepta: a^*

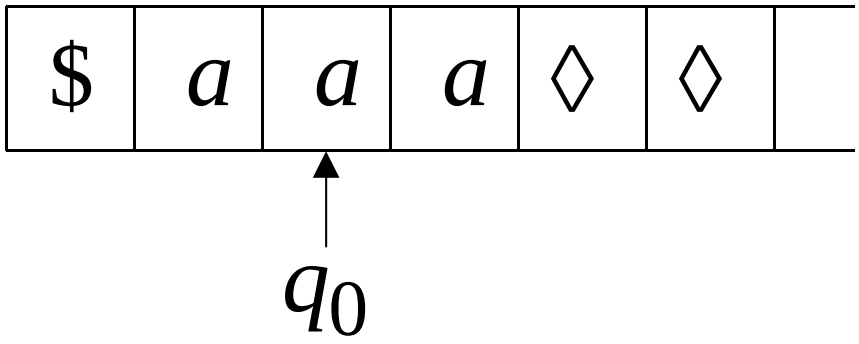


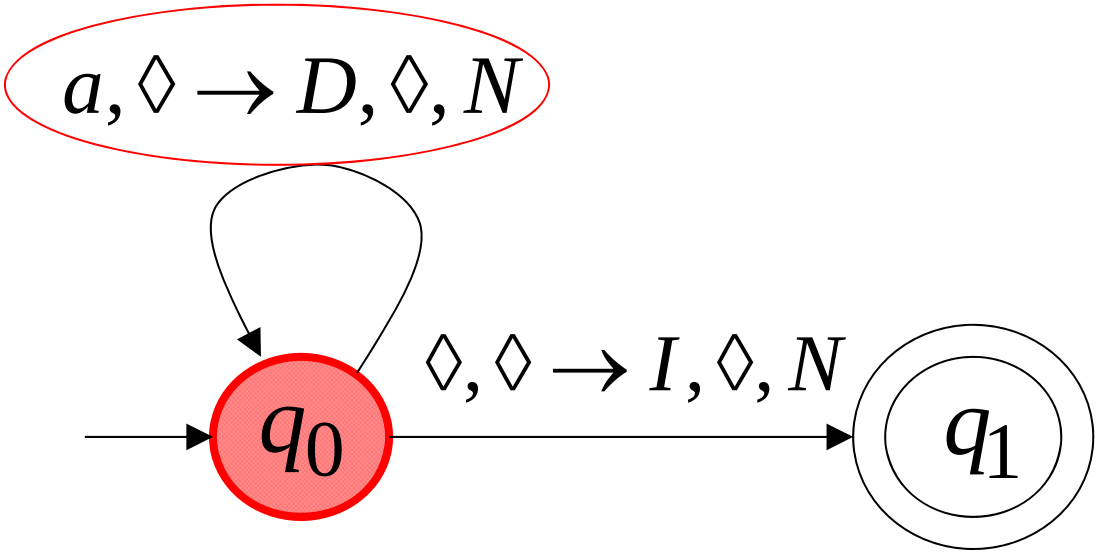
Tiempo 0


$
a
a
a
◊
◊
q_0

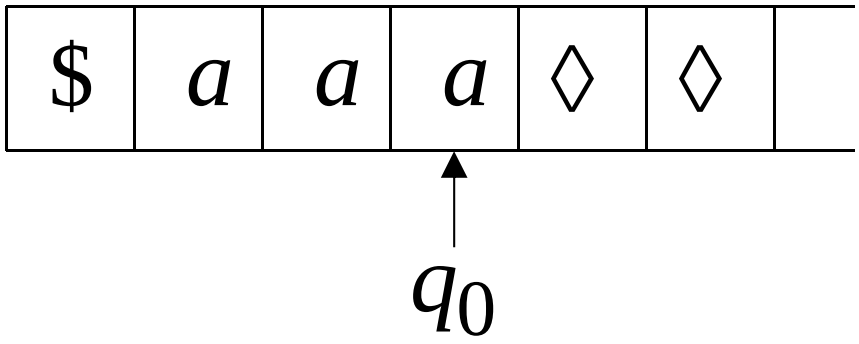

$a, \Diamond \rightarrow D, \Diamond, N$
$\Diamond, \Diamond \rightarrow I, \Diamond, N$
q_0
q_1

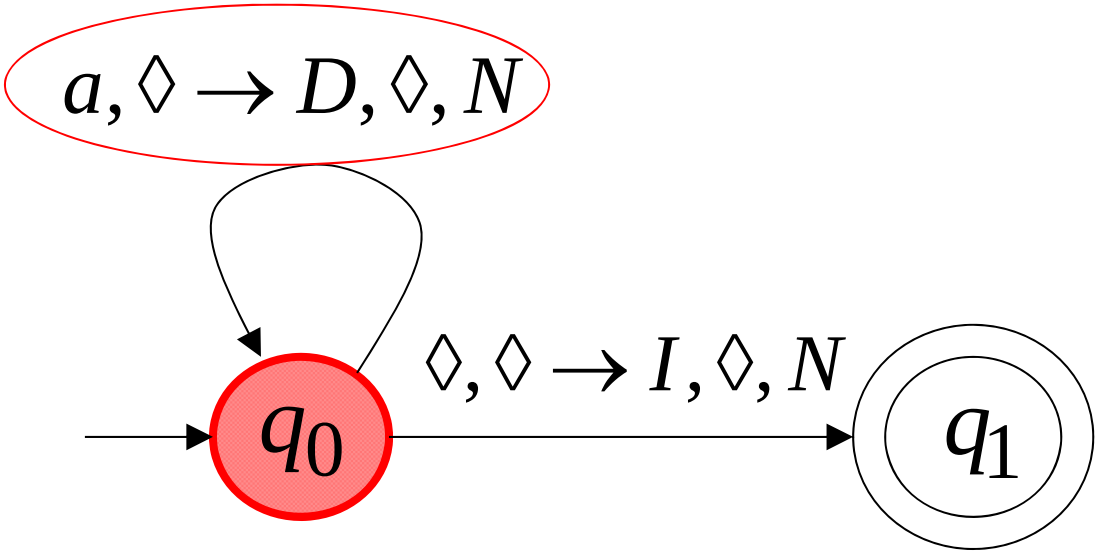
Tiempo 1


$
a
a
a
◊
◊
q_0

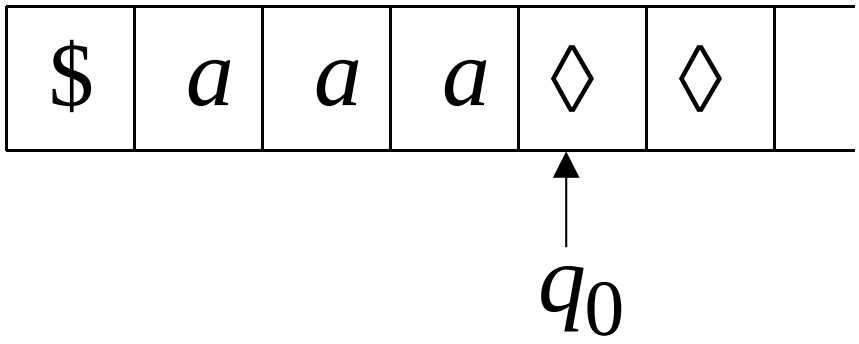

$a, \Diamond \rightarrow D, \Diamond, N$
$\Diamond, \Diamond \rightarrow I, \Diamond, N$
q_0
q_1

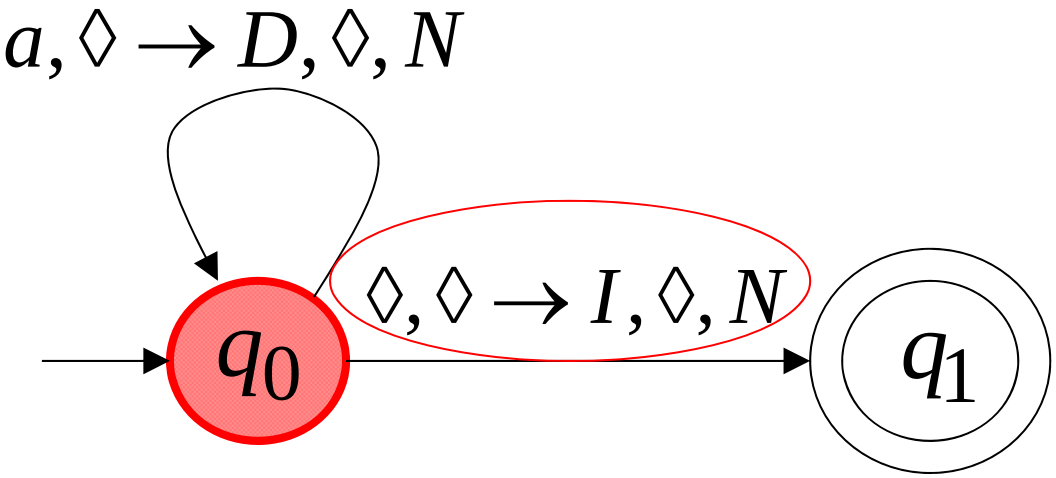
Tiempo 2


$
a
a
a
◊
◊
q_0

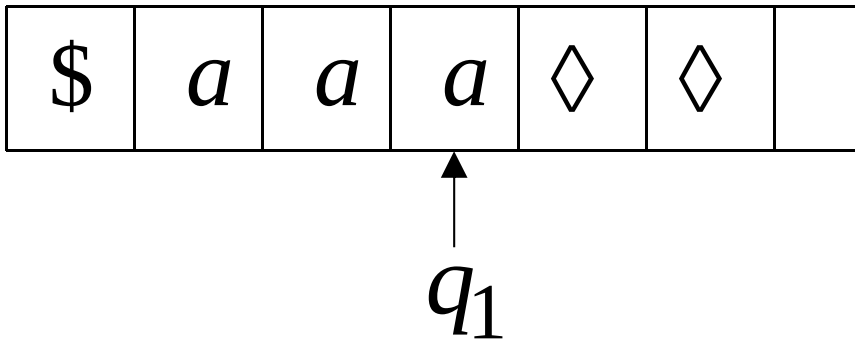

$a, \lozenge \rightarrow D, \lozenge, N$
$\lozenge, \lozenge \rightarrow I, \lozenge, N$
q_0
q_1

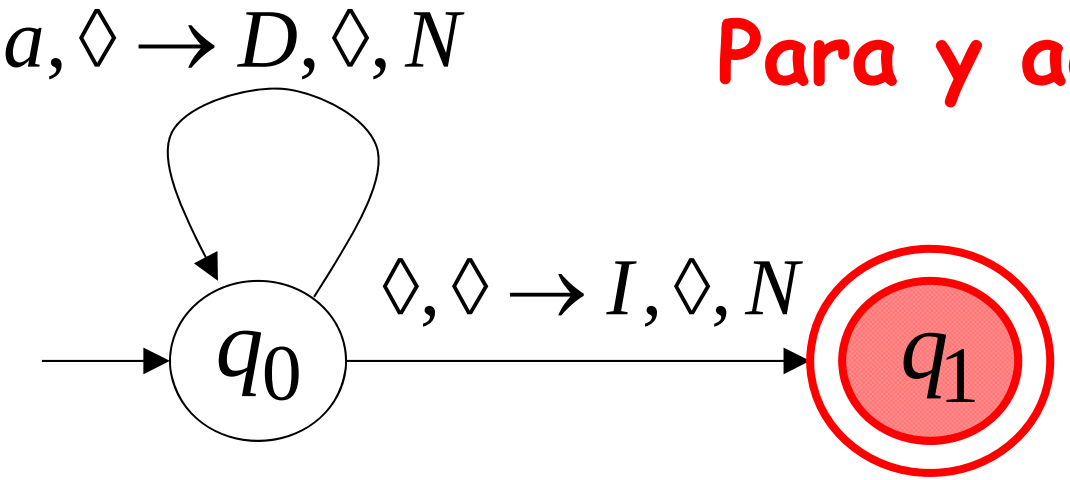
Tiempo 3


$
a
a
a
◊
◊
q_0


$a, \lozenge \rightarrow D, \lozenge, N$
$\lozenge, \lozenge \rightarrow I, \lozenge, N$
q_0
q_1

Tiempo 4

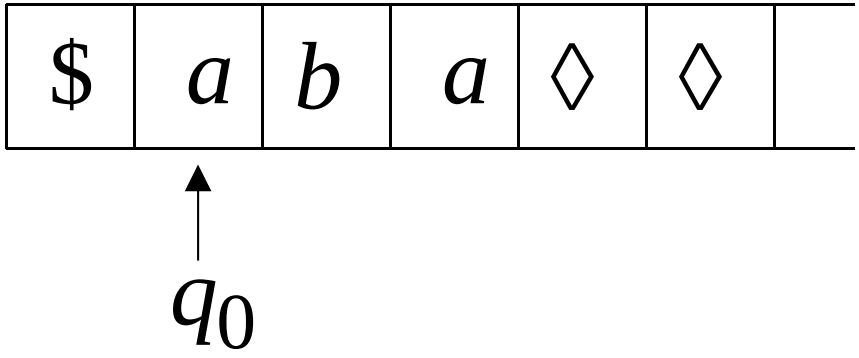


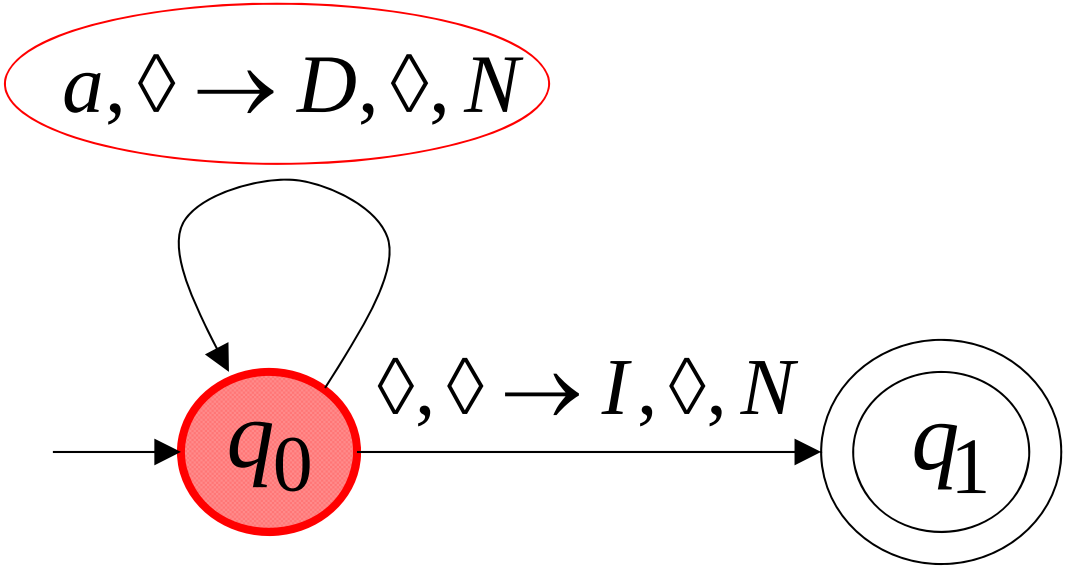


Para y acepta

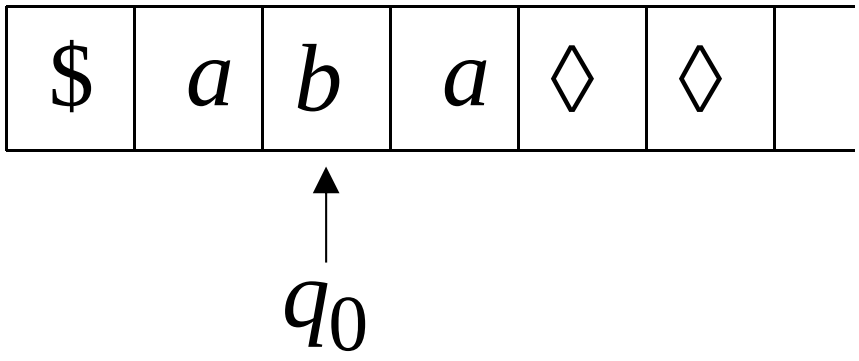
Ejemplo de rechazo

Tiempo 0



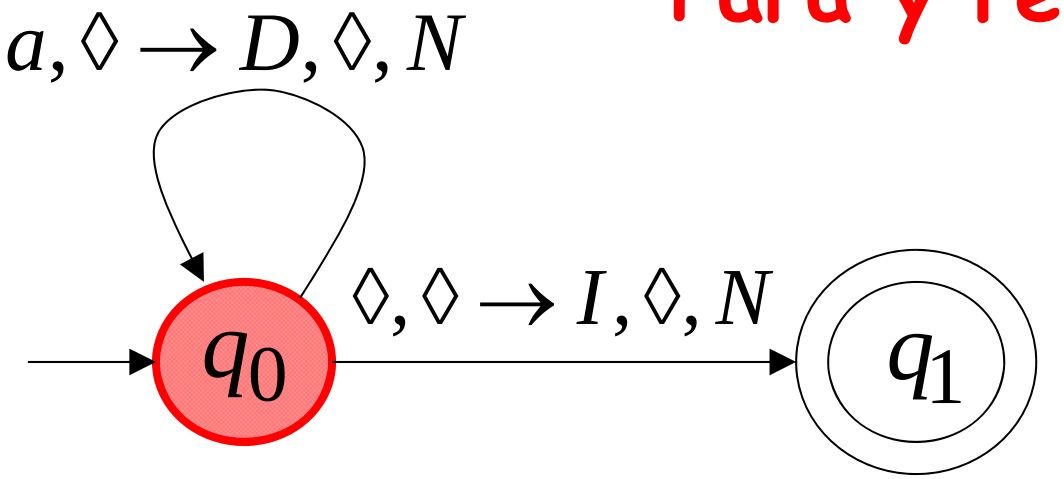


Tiempo 1



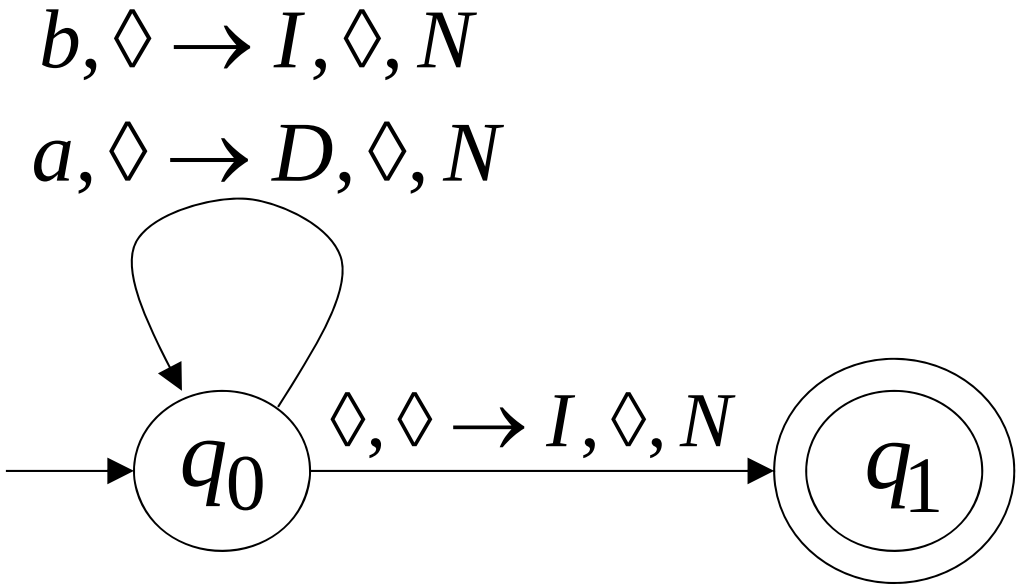
No hay transición posible

Para y rechaza

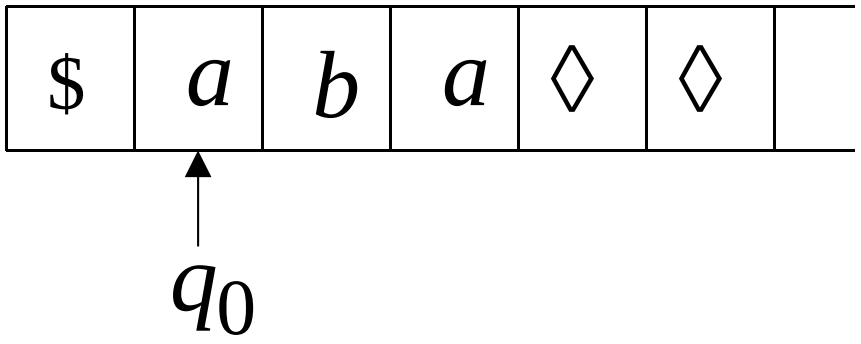


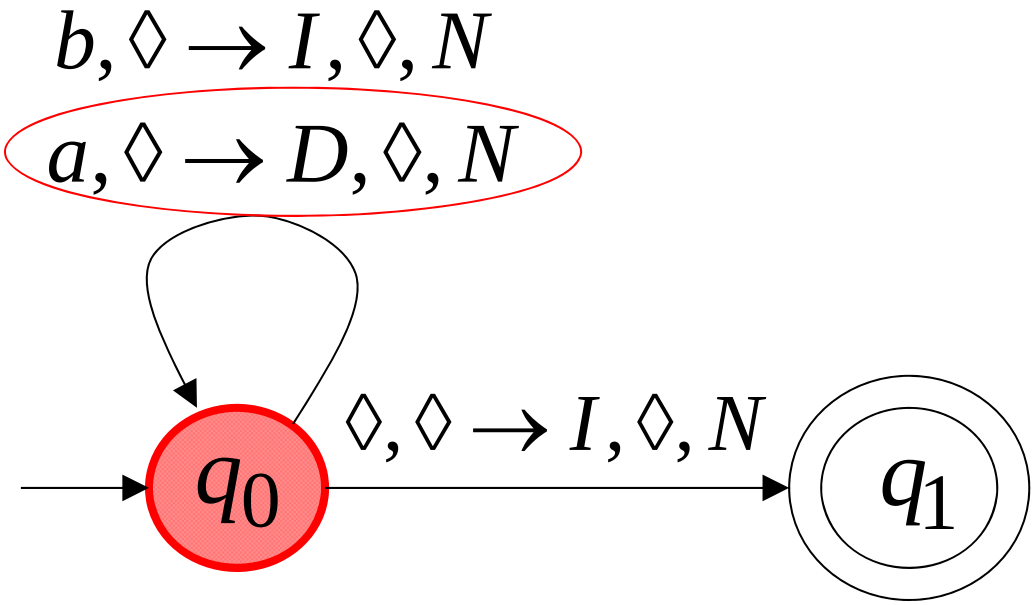
Ejemplo con bucle infinito

Una máquina de Turing para el lenguaje a^*

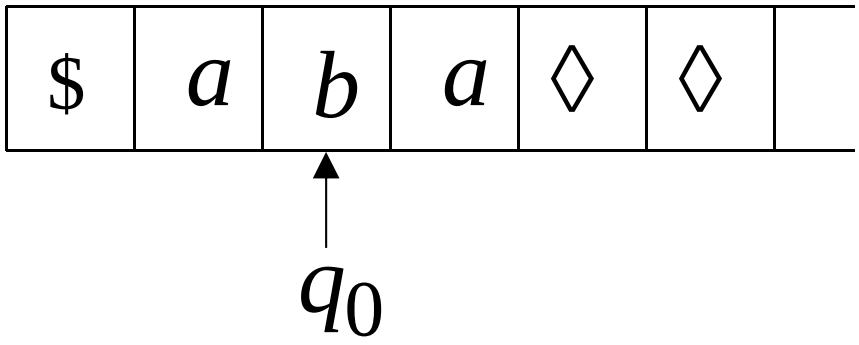


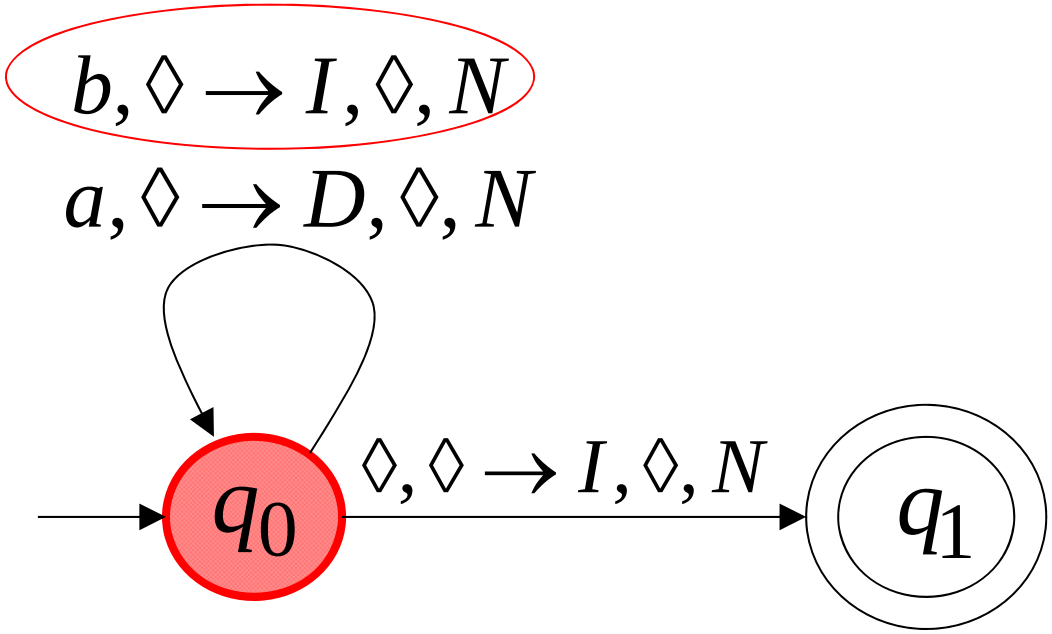
Tiempo 0


$
a
b
a
◊
◊
q_0

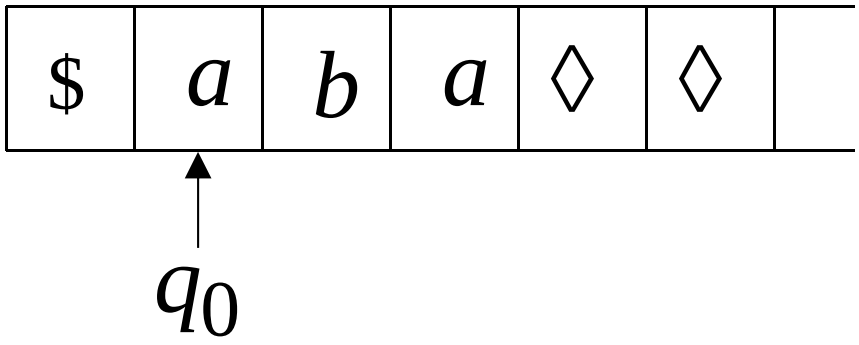

$b, \Diamond \rightarrow I, \Diamond, N$
$a, \Diamond \rightarrow D, \Diamond, N$
$\Diamond, \Diamond \rightarrow I, \Diamond, N$
q_0
q_1

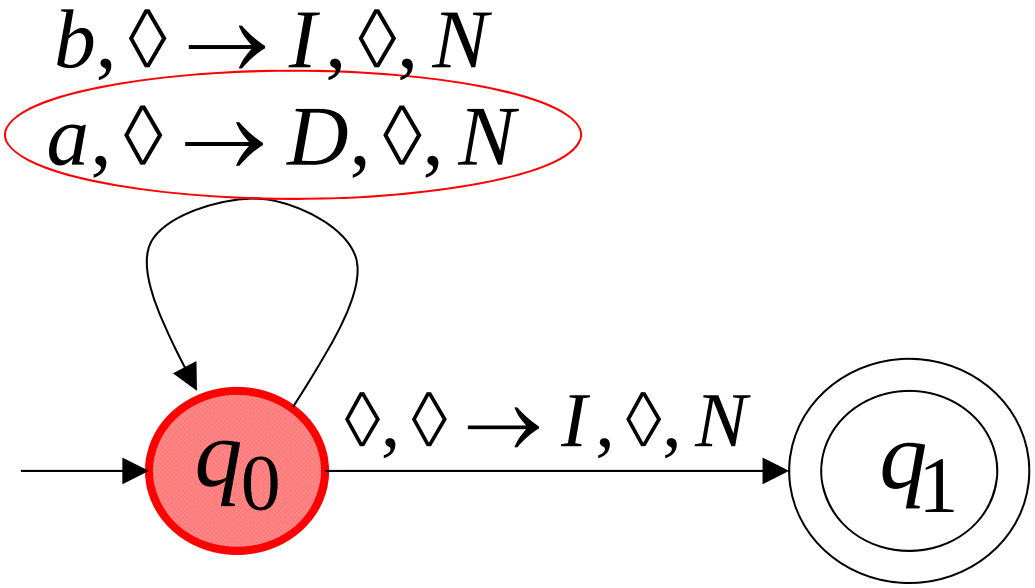
Tiempo 1


$
a
b
a
◊
◊
q_0

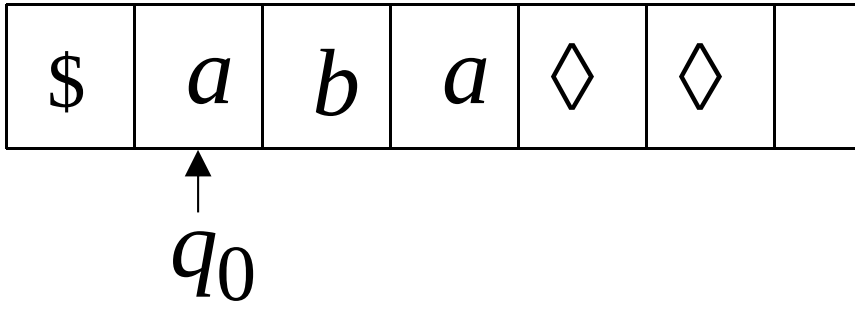

$b, \Diamond \rightarrow I, \Diamond, N$
$a, \Diamond \rightarrow D, \Diamond, N$
$\Diamond, \Diamond \rightarrow I, \Diamond, N$
q_0
q_1

Tiempo 2

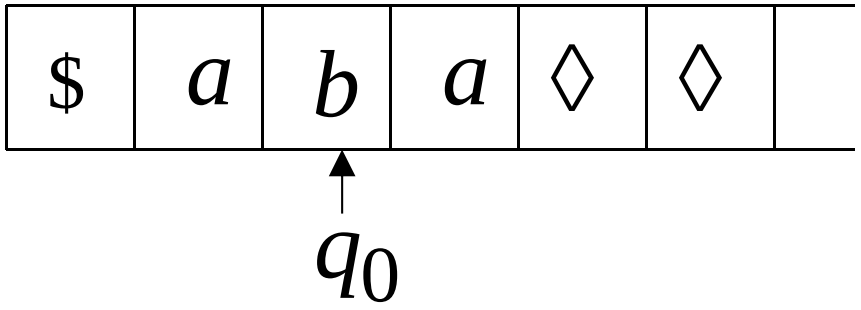

$
a
b
a
◊
◊
q_0


$b, \Diamond \rightarrow I, \Diamond, N$
$a, \Diamond \rightarrow D, \Diamond, N$
$\Diamond, \Diamond \rightarrow I, \Diamond, N$
q_0
q_1

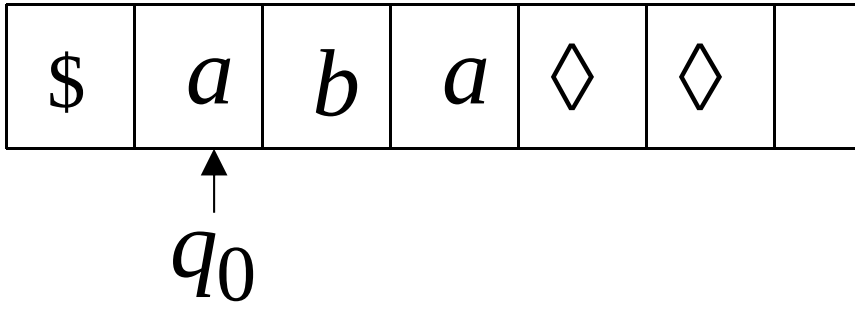
Tiempo 2



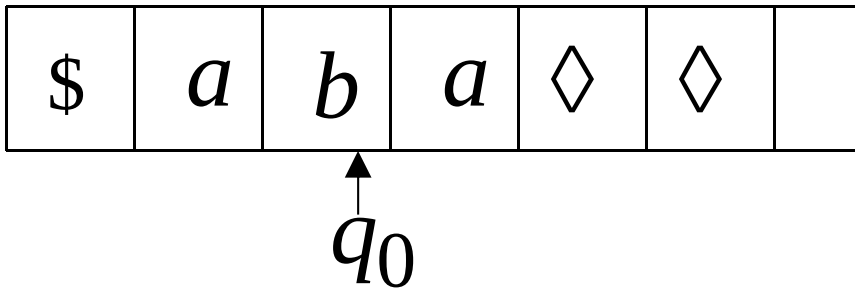
Tiempo 3



Tiempo 4



Tiempo 5



Bucle infinito

Por el bucle infinito:

- No se puede alcanzar el estado final
- La máquina nunca para
- No se acepta la entrada

Máquinas que paran siempre

Una máquina de Turing M para siempre si para cualquier cadena w, M con entrada w para

Lenguajes decidibles

Un lenguaje L es decidible si es el aceptado por una máquina de Turing M que para siempre

$w \in L \Rightarrow$ M para en un estado final

$w \notin L \Rightarrow$ M para en un estado no final

Lenguajes semidecidibles

Para una máquina de Turing M

$$L(M) = \left\{ w : \begin{array}{l} \text{desde el estado } q_0, \text{ con } w \text{ en la entrada,} \\ \text{la máquina para en un estado final} \end{array} \right\}$$

Un lenguaje es semidecidible si es aceptado por una máquina de Turing

Variaciones de máquinas de Turing

- Más de una cinta de memoria
- Memoria ilimitada también por la izda
- Entrada en la cinta de memoria
- Memoria en varias dimensiones
- Máquina no determinista

¡¡Todas equivalentes!!

Programas

Programas …

Consideramos programas sintácticamente correctos:

Tipos de datos predefinidos

booleano, carácter, natural, entero, real, cadena

Instrucciones condicionales

si <condición> **entonces**
  <secuencia de acciones>
sino
  <secuencia de acciones>
fsi

Instrucciones iterativas

mientrasQue <condición> **hacer**
  <secuencia de acciones>
fmq

Programas …

```
Procedimientos y funciones
procedimiento <nombre>(ent <parámetros_1>:<tipo_1>;
                       sal <parámetros_2>:<tipo_2>;
                       e/s <parámetros_3>:<tipo_3> ... )
…
función <nombre>(<parám_1>:<tipo_1>; <parám_2>:<tipo_2> ...)
devuelve <tipo_fun>
<declaraciones locales de constantes, tipos, variables,
proced., funciones...>
principio
  <secuencia de acciones>
  devuelve <valor_de_tipo_fun>   {tras devolver el valor la
función termina}
fin
…
```

Y el resto de la sintaxis habitual

Programas ideales …

El único añadido es que la memoria es ilimitada, es decir

- No hay nunca errores por "overflow" (de una variable o de un puntero)

De esta forma un programa real es un programa ideal, y un programa ideal se puede implementar en un computador real si tiene suficiente memoria

Codificación

Cualquier dato elemental (booleano, carácter, natural, entero, etc.) puede codificarse como una cadena, y varias cadenas en una sola (por ej. con un símbolo extra #)

Luego cualquier programa es equivalente a:

```
procedimiento ejemplo (ent w:cadena; sal z:cadena)
```

Nos interesan especialmente …

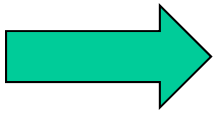
Un programa o algoritmo de decisión es el que tiene salida de tipo tpresultado

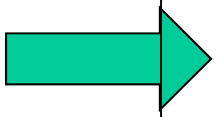
```
tipo  tpresultado = (acepta,rechaza)

procedimiento ejemplo (ent w:cadena; sal z:tpresultado)
```

Parar

Un programa con una entrada ***para*** si termina su ejecución

Aceptar

Aceptar Entrada  Si el programa para y devuelve acepta

Rechazar Entrada  Si el programa para y no devuelve acepta

o

Si el programa no para nunca

El lenguaje aceptado

Para un programa p

$$L(p) = \left\{ w : \begin{array}{l} \text{el programa } p \text{ con entrada } w \\ \text{para y devuelve } \textit{acepta} \end{array} \right\}$$

Ejemplo de programa que no para

Programas que paran siempre

Una programa p para siempre si para cualquier cadena w, p con entrada w para

Tesis de Turing-Church

Teorema

Las máquinas de Turing y los programas son equivalentes

Son equivalentes quiere decir

Para cualquier máquina M
existe un programa p tal que:

- L(M)=L(p)
- M y p paran con las mismas entradas

y viceversa

para cualquier programa p
existe una máquina M tal que L(M)=L(p)
y M y p paran con las mismas entradas

Demostración

1) Para cualquier máquina M

existe un programa p tal que L(M)=L(p)

y M y p paran con las mismas entradas

Escribir un programa que simule una máquina de Turing … (sencillo)

Demostración

1) Para cualquier programa p
existe una máquina M tal que L(M)=L(p)
y M y p paran con las mismas entradas

Escribir una máquina de Turing que simule una programa … (complicado)

Primero hacer una máquina que simule cada tipo de instrucción (condicional, bucle)

Luego combinarlas

Lenguajes semidecidibles

Para un programa p

$$L(p) = \left\{ w : \begin{array}{l} \text{el programa } p \text{ con entrada } w \\ \text{para y devuelve } \textit{acepta} \end{array} \right\}$$

Un lenguaje es semidecidible si es aceptado por un programa

Lenguajes decidibles

Un lenguaje L es decidible si es el aceptado por un programa que para siempre

$w \in L \Rightarrow$ p para y devuelve acepta

$w \notin L \Rightarrow$ p para y no devuelve acepta

Tesis de Turing-Church: (1930)

Cualquier modelo de cálculo que se pueda implementar físcamente es equivalente a las máquinas de Turing

Tesis de Turing-Church:

Se cumple para los modelos conocidos:

- Programas (imperativos)
- Programas lógicos
- Programas funcionales
- Máquinas paralelas
- ....

Tesis = conjetura. No hay demostración

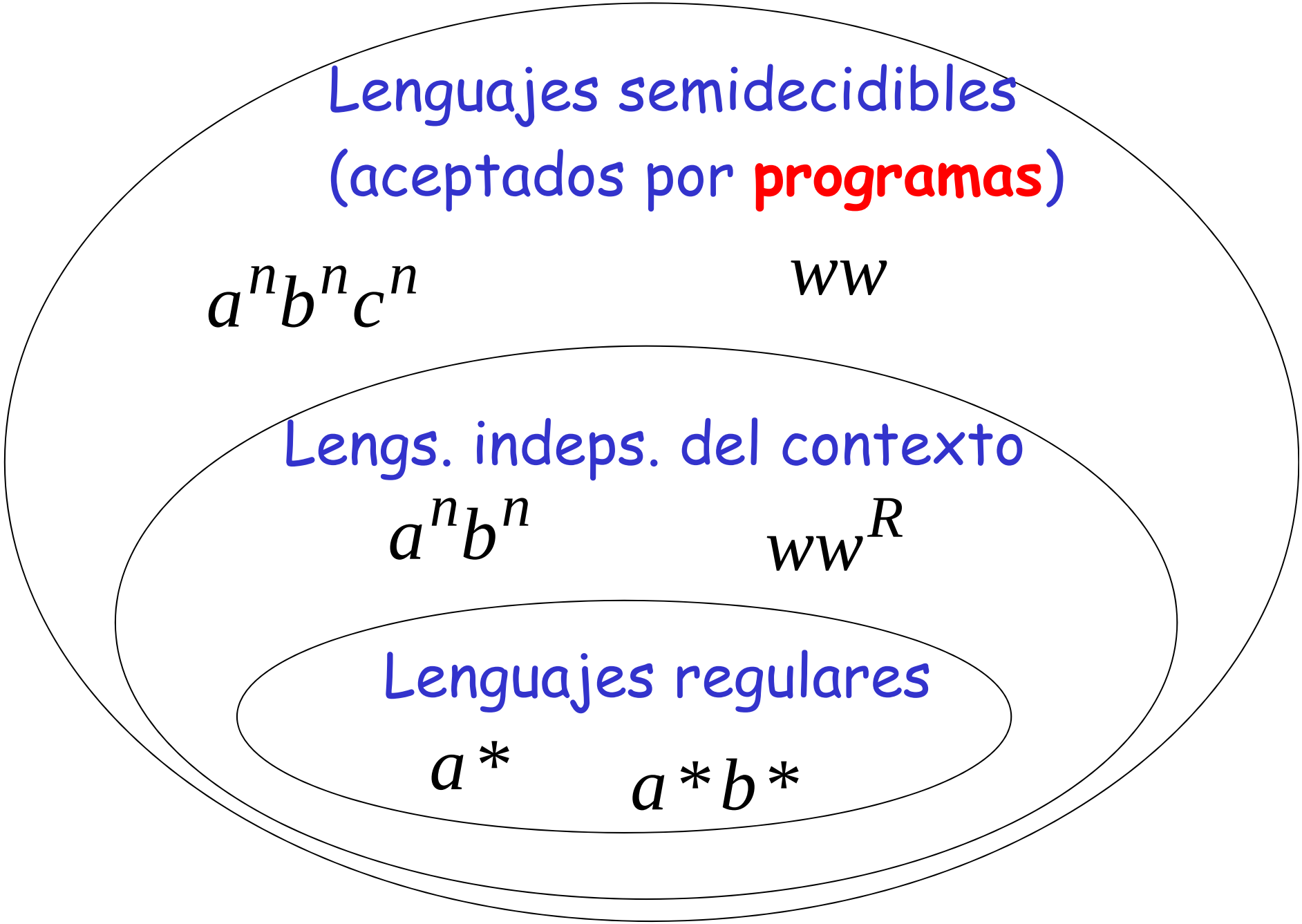
Los programas son máquinas de Turing

Cuando decimos:

Existe un algoritmo que hace x

Es lo mismo que:

Existe una máquina de Turing que hace x


Lenguajes semidecidibles
(aceptados por programas)
$a^n b^n c^n$
ww
Lengs. indeps. del contexto
$a^n b^n$
ww^R
Lenguajes regulares
$a*$
$a*b*$