

# Lenguajes formales

**Contexto:** Teoría de Autómatas y Lenguajes Formales

- Veremos a los **lenguajes** desde el punto de vista de su aplicación a **problemas** de computación
- Lenguajes con **estructuras sintácticas más o menos complicadas** se asocian a **problemas más o menos complicados** de computar.

# Lenguajes formales

Los Lenguajes son conjuntos de **sentencias** (*strings o cadenas*) construidas a partir de un conjunto finito de símbolos (el alfabeto). Cada una de las sentencias de un lenguaje es una **secuencia finita** de estos símbolos.

# Lenguajes formales

## Sintaxis vs. Semántica

•**Sintaxis**: Principios y procesos que permiten **combinar los símbolos** para formar las sentencias de un lenguaje particular. Corresponde a la pregunta ¿Es gramaticalmente correcto?

•**Semántica**: Mecanismo subyacente a través del cual se le asigna un **significado** a las sentencias de un lenguaje particular. Corresponde a las preguntas ¿Qué significa esta sentencia? ¿Qué sentencia tiene sentido?

Es claro que para que una sentencia tenga sentido es conveniente que sea sintácticamente correcta.

Para trabajar con **lenguajes formales** sólo hace falta observar la **sintaxis** ( las formas ).

# Lenguajes formales

Desde el punto de vista sintáctico (es decir en el contexto de los lenguajes formales) existen dos cuestiones importantes:

**La generación:** Gramáticas para generar sentencias sintácticamente correctas.

**El reconocimiento o aceptación:** Autómatas capaces de reconocer si una sentencia es sintácticamente correcta para un determinado lenguaje. Las MT son un ejemplo de un tipo particular de autómata.

# Lenguajes formales

## Clases de lenguajes (Chomsky)

Según el tipo de autómatata que lo acepte o gramática que lo genere:

- Lenguajes Regulares
- Lenguajes Libres de contexto
- Lenguajes Sensibles al contexto
- Lenguajes Recursivos y Recursivos Enumerables

Máquinas de Turing

A diagram showing a yellow rectangular box with a black border containing the text 'Máquinas de Turing'. A vertical black arrow points upwards from the top center of this box to the underlined text 'Lenguajes Recursivos y Recursivos Enumerables' in the list above.

# Definiciones

**Definición. Alfabeto:** Diremos que un conjunto finito  $\Sigma$  es un alfabeto

si  $\Sigma \neq \emptyset$  y  $(\forall x)(x \in \Sigma \rightarrow x \text{ es un símbolo indivisible})$

Ejemplos       $\Sigma = \{a,b\}$ ,  $\Sigma = \{0,1\}$ ,  $\Sigma = \{a,b,\dots,z\}$  son alfabetos  
 $\Sigma = \{0,1,00,01\}$   $\Sigma = \{sa,ca,casa\}$  no lo son

# Definiciones

**Definicion. Palabra:** Se dice que  $w$  es una palabra (cadena, sentencia o string) sobre  $\Sigma$  si  $w$  es una **secuencia finita** de símbolos **de  $\Sigma$**

Ejemplos: si  $\Sigma = \{0,1\}$ , entonces:

0011, 101, 1 son palabras sobre  $\Sigma$

# Definiciones

**Definicion. Longitud de una palabra:** Se denota  $|w|$ , es el número de símbolos que contiene  $w$ .

Por ejemplo:  $|\text{perro}|=5$   $|010|=3$

**Nota:** notaremos con  $\Sigma^*$  al conjunto de todas las palabras formadas por símbolos de  $\Sigma$  incluida la cadena nula (o vacía) que tiene longitud cero y denotaremos con  $\lambda$  ( $|\lambda| = 0$ )

Ejemplo:  $\Sigma = \{a,b\}$

$\Sigma^* = \{\lambda, a, b, aa, ab, ba, bb, aaa...\}$



# Definiciones

**Concatenación:** La notación utilizada para denotar la concatenación de dos palabras  $w$  y  $v$  es  $w.v$  (o simplemente  $wv$ ).

La concatenación es asociativa pero no conmutativa:

$$(v.w).x = v.(wx) \quad v.w \neq w.v$$

Se cumple que:

$$|v.w| = |v| + |w|$$

La cadena vacía es el elemento neutro para la concatenación  $\lambda.w = w.\lambda = w$

# Definiciones

**Definición.** Sea una palabra  $w \in \Sigma^*$  y un número natural  $i$ , se define la **potencia  $i$ -ésima** de  $w$  como:

$$\begin{aligned} w^0 &= \lambda \\ w^{(i+1)} &= w.w^i \end{aligned} \quad (\forall i) \ (i \geq 0)$$

Ejemplo: si  $w = ab$ ,  $w^3 = ababab$

# Definiciones

**Definición.** Se denomina **lenguaje definido sobre  $\Sigma$**  a cualquier **subconjunto de  $\Sigma^*$**

Ejemplo: si  $\Sigma = \{0,1\}$

$\emptyset$  ,  $\Sigma^*$  ,  $\{\lambda\}$  ,  $\{w \in \Sigma^* / w \text{ comienza con } 1\}$  son lenguajes sobre  $\Sigma$

Si  $L$  es un lenguaje sobre  $\Sigma$ , su complemento también lo es (Complemento de  $L$  respecto de  $\Sigma^*$  es  $\bar{L} = \Sigma^* - L$ )

# Definiciones

**Nota:** Llamaremos  $\mathcal{L}$  al conjunto de todos los lenguajes definidos sobre el alfabeto  $\Sigma$ , es decir:

$$\mathcal{L} = \rho(\Sigma^*)$$

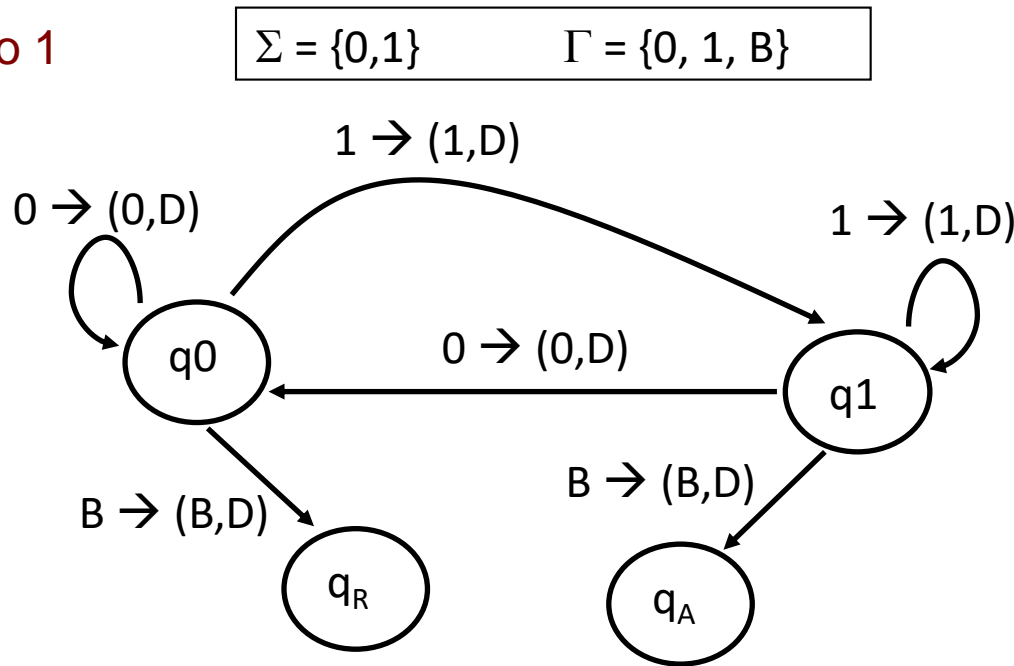
# Máquina de Turing como reconocedoras de cadenas de símbolos

Para este tipo de máquina de Turing vamos a considerar dos estados de parada:

- $q_A$  : el estado de aceptación.
- $q_R$  : el estado de rechazo.
- Se define  $\delta$  de manera completa, considerando todos los estados de  $Q$ . Sin embargo, ni  $q_A$  ni  $q_R$  pertenecen a  $Q$ .

# Máquina de Turing como reconocedoras de cadenas de símbolos

Ejemplo 1

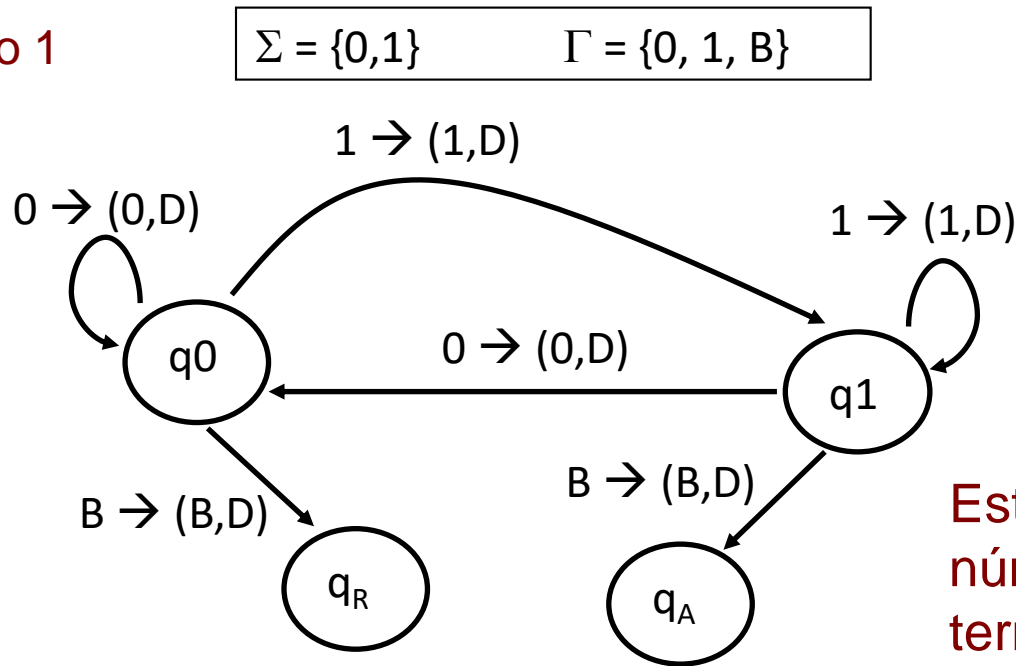


¿Qué strings reconoce esta máquina?

Es decir ¿para qué cadenas de símbolos la máquina se detiene en  $q_A$ ?

# Máquina de Turing como reconocedoras de cadenas de símbolos

Ejemplo 1



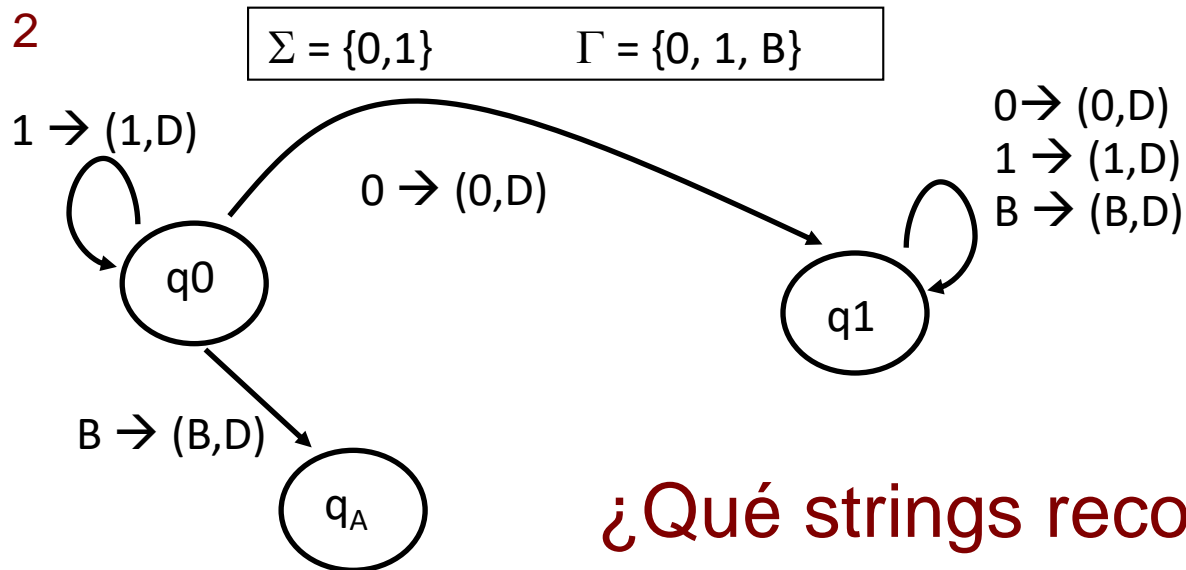
Esta máquina reconoce  
números binarios  
terminados en 1  
 $L = \{w1, w \in \Sigma^*\}$

Observar que:

- No hay links que salen de los estados  $q_A$  y  $q_R$ .
- $\delta$  está definida para todos los demás casos

# Máquina de Turing como reconocedoras de cadenas de símbolos

## Ejemplo 2



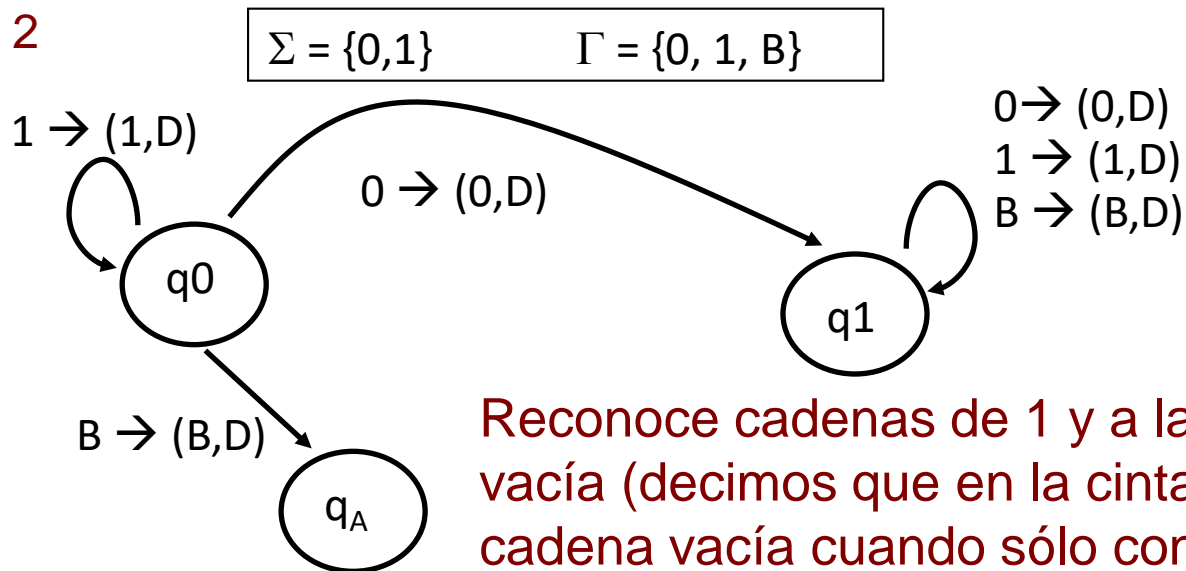
¿Qué strings reconoce esta máquina?

Es decir ¿para qué cadenas de símbolos la máquina se detiene en  $q_A$ ?



# Máquina de Turing como reconocedoras de cadenas de símbolos

## Ejemplo 2



Reconoce cadenas de 1 y a la cadena vacía (decimos que en la cinta está la cadena vacía cuando sólo contiene símbolos B)

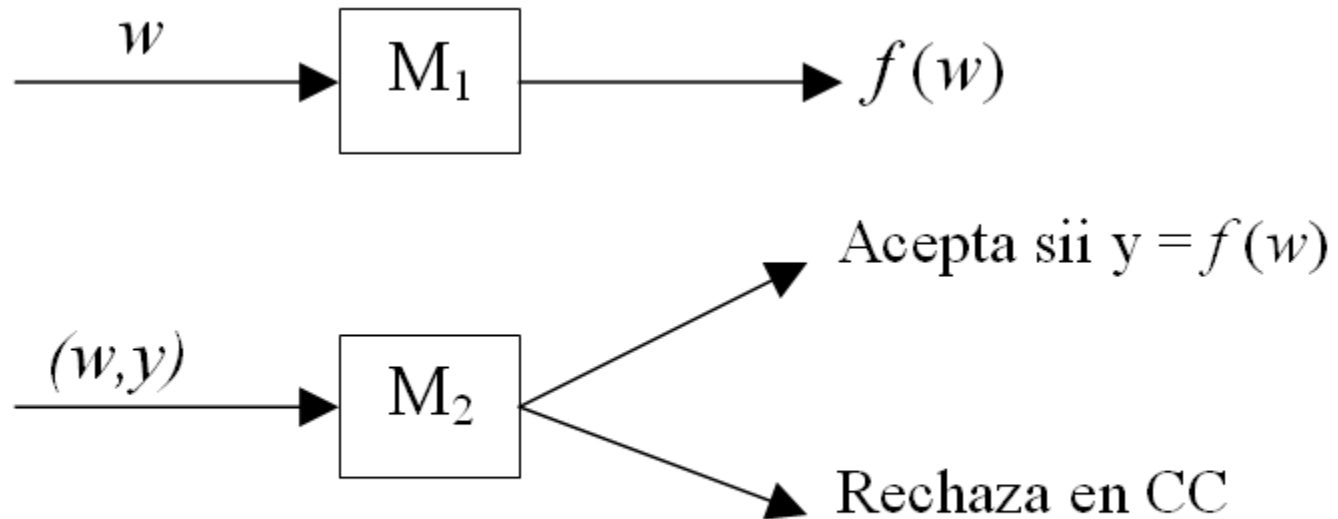
$L = \{1^n, n \geq 0\}$  ( $\lambda \in L, n = 0$ )

Observar que:

- No hay transiciones al estado  $q_R$  (es perfectamente válido)
- Se rechazan las cadenas "loopeando" (lazo o loop infinito), es decir, la máquina no se detiene nunca.

# Máquina de Turing como reconocedoras de cadenas de símbolos

Para el estudio de la computabilidad podemos quedarnos únicamente con las máquinas de Turing reconocedoras sin perder generalidad. Intuitivamente:



$M_2$  es una máquina de Turing reconocedora

Si se puede computar  $f(w)$  se puede reconocer el lenguaje de los pares  $(w, f(w))$  y viceversa

# Formalización del modelo de Máquina de Turing $q_A, q_R$

$M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R \rangle$  con  $q_A, q_R \notin Q$

Donde:

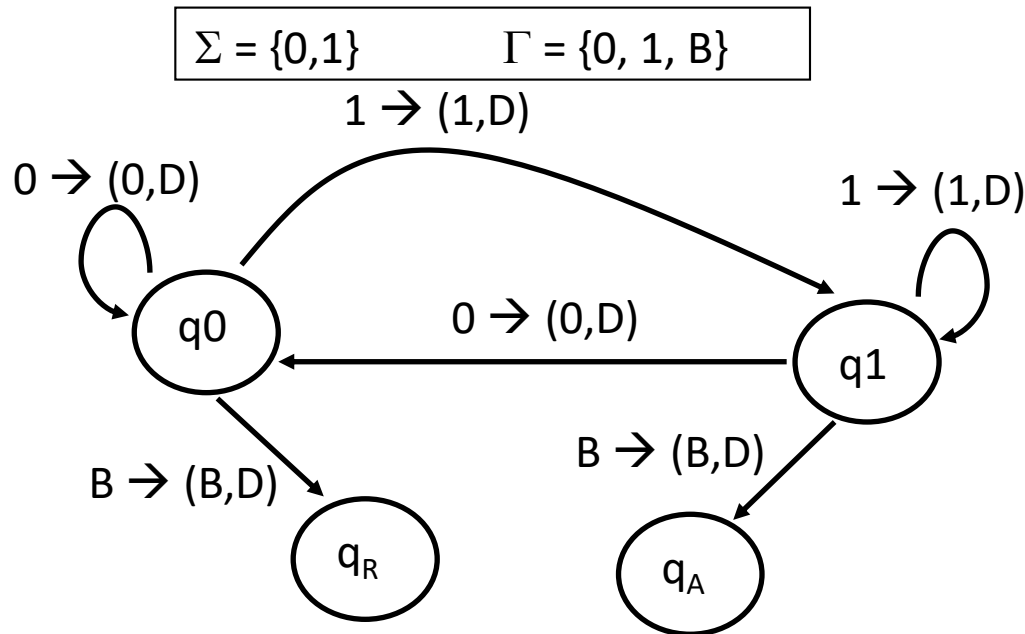
$Q, \Sigma, \Gamma$  y  $q_0$  se definen como en el caso de las MT de cómputo

$\delta: Q \times \Gamma \rightarrow Q \cup \{q_A, q_R\} \times \Gamma \times \{D, I\}$ , completa

$q_A$  es el estado de aceptación,  $q_R$  es el estado de rechazo

**$M$  se detiene  $\Leftrightarrow M$  pasa al estado  $q_A$  o  $q_R$**

# Ejemplo (revisitado)



$$M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R \rangle$$

$$Q = \{q_0, q_1\} \quad \Sigma = \{0, 1\} \quad \Gamma = \{B, 0, 1\}$$

$$\delta: Q \times \Gamma \rightarrow Q \cup \{q_A, q_R\} \times \Gamma \times \{D, I\}$$

$$\delta(q_0, 0) = (q_0, 0, D) \quad \delta(q_0, 1) = (q_1, 1, D) \quad \delta(q_0, B) = (q_R, B, D)$$

$$\delta(q_1, 0) = (q_0, 0, D) \quad \delta(q_1, 1) = (q_1, 1, D) \quad \delta(q_1, B) = (q_A, B, D)$$

# Ejemplo (revisitado)

Otra forma de especificar la función de transición  $\delta$  es con una tabla

$\delta$	0	1	B
q0	$(q0, 0, D)$	$(q1, 1, D)$	$(q_R, B, D)$
q1	$(q0, 0, D)$	$(q1, 1, D)$	$(q_A, B, D)$

## Ejemplo - Traza

$\delta$	0	1	B
q0	(q0,0,D)	(q1,1,D)	(q <sub>R</sub> ,B,D)
q1	(q0,0,D)	(q1,1,D)	(q <sub>A</sub> ,B,D)

Traza de ejecución (todos los movimientos) de la MT que reconoce secuencias binarias terminadas en 1 para la entrada  $w=0101$

$$q_0 0 1 0 1 \vdash_M 0 q_0 1 0 1 \vdash_M 0 1 q_1 0 1 \vdash_M 0 1 0 q_0 1 \vdash_M 0 1 0 1 q_1 B \vdash_M 0 1 0 1 B q_A B$$

Se puede escribir entonces:

$$q_0 0 1 0 1 \vdash_M^* 0 1 0 1 B q_A B$$

# Lenguaje Aceptado por una MT

Definición: El lenguaje aceptado por una MT

$M = \langle Q, \Sigma, \Gamma, \delta, q_0, q_A, q_R \rangle$  es:

$$L(M) = \{ w \in \Sigma^* / q_0 w \vdash_M^* \alpha q_A \beta, \alpha \beta \in \Gamma^* \}$$

Dicho de otra forma:

$$w \in L(M) \Leftrightarrow \text{con entrada } w, M \text{ para en } q_A$$

Nota: Para las cadenas que no pertenecen a  $L(M)$

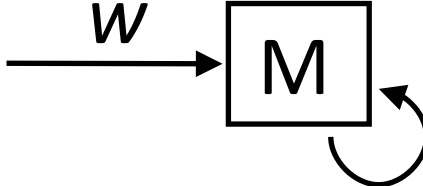
la MT  $M$  para en  $q_R$  o loopea

# Lenguaje Aceptado por una MT

Por cada entrada, hay 3 casos posibles:

1)   $M$  se detiene en  $q_A \Rightarrow w \in L(M)$

2)   $M$  se detiene en  $q_R \Rightarrow w \notin L(M)$

3)   $M$  no se detiene  $\Rightarrow w \notin L(M)$

Observar que: una MT  $M$  reconocedora realiza una partición de  $\Sigma^*$  en dos conjuntos:  $L(M)$  y  $\overline{L(M)}$ , donde  $\overline{L(M)} = \Sigma^* - L(M)$



# Lenguaje Aceptado por una MT

## Ejercicios:

a) Construir una máquina de Turing  $M$  tal que  $L(M) = \{0^n 1^n / n \geq 1\}$  y describir los movimientos de la máquina (traza de computación) para las entradas  $w_1=0011$  y  $w_2=011$

b) Construir una máquina de Turing que busque en la cinta el patrón “abab” y se detenga si y sólo si encuentra ese patrón.  $\Sigma = \{a,b,c\}$