

# Propiedades de Decisión de los Lenguajes Regulares

Alan Reyes-Figueroa

Teoría de la Computación

(Aula 08) 14.agosto.2023

Discusión general de “Propiedades”

El Lema de Bombeo

Pertenencia, Vacuidad, Finitud, Etc.

# Propiedades Clases de Lenguajes

- ◆ Una *clase de lenguajes* es un conjunto de lenguajes.
  - ▶ Ejemplo: los lenguajes regulares.
  - ▶ Veremos muchos otros ejemplos.
- ◆ Las clases de lenguajes tienen dos tipos importantes de propiedades:
  1. Propiedades de decisión.
  2. Propiedades de cerradura.

# Representación de Lenguajes

- ◆ Pueden ser formales o informales.
- ◆ **Ejemplo** (formal): representar un lenguaje mediante su *regex* o DFA que lo define.
- ◆ **Ejemplo**: (informal): mediante un enunciado prosaico o lógico acerca de sus cadenas:
  - ◆  $\{0^n1^n: n \text{ es un entero no-negativo}\}$
  - ◆ “El conjunto de las cadenas que consisten de un cierto número de 0’s seguido del mismo número de 1’s.”

# Propiedades de Decisión

- ◆ Una *propiedad de decisión* para una clase de lenguajes es un algoritmo que toma una descripción formal del lenguaje (e.g., un DFA) y nos dice cuándo cierta propiedad de ese lenguaje se cumple o no.
- ◆ **Ejemplo:** Es el lenguaje  $L$  vacío?  
¿Son los lenguajes  $L$  y  $M$  iguales?

# Por qué prop. de decisión?

- ◆ Cuando hablamos de ejemplos de protocolos representados por DFAs, vimos que propiedades de un buen protocolo se relacionan con el lenguaje.
- ◆ **Ejemplo:** “El protocolo termina?” = “Es el lenguaje finito?”
- ◆ **Ejemplo:** “Puede el protocolo fallar?” = “Es el lenguaje no vacío?”

# Por qué prop. de decisión?

- ◆ Otra razón es que para un lenguaje, nos gustaría tener la “menor” representación posible, e.g., DFA con estados mínimos o la menor *regex*.
- ◆ Si no podemos decidir “Son estos dos lenguajes el mismo?”
  - ◆ i.e., dos DFA's definen el mismo lenguaje?Entonces no podemos hallar el “menor”.

# Propiedades de cerradura

- ◆ Una *propiedad de cerradura* de una clase de lenguajes establece que lenguajes en dicha clase, una operación (e.g., unión) entre ellos produce otro lenguaje en la misma clase.
- ◆ **Ejemplo:** los lenguajes regulares son cerrados mediante unión, concatenación, y la cerradura de Kleene.
  - ◆ Usar la representación *regex*.

# Por qué prop. de cerradura?

1. Ayudan a construir representaciones.
2. Ayudan a mostrar (de manera informal) que un cierto lenguajes no pertenece a una clase.



# Ejemplo: propiedad de cerradura

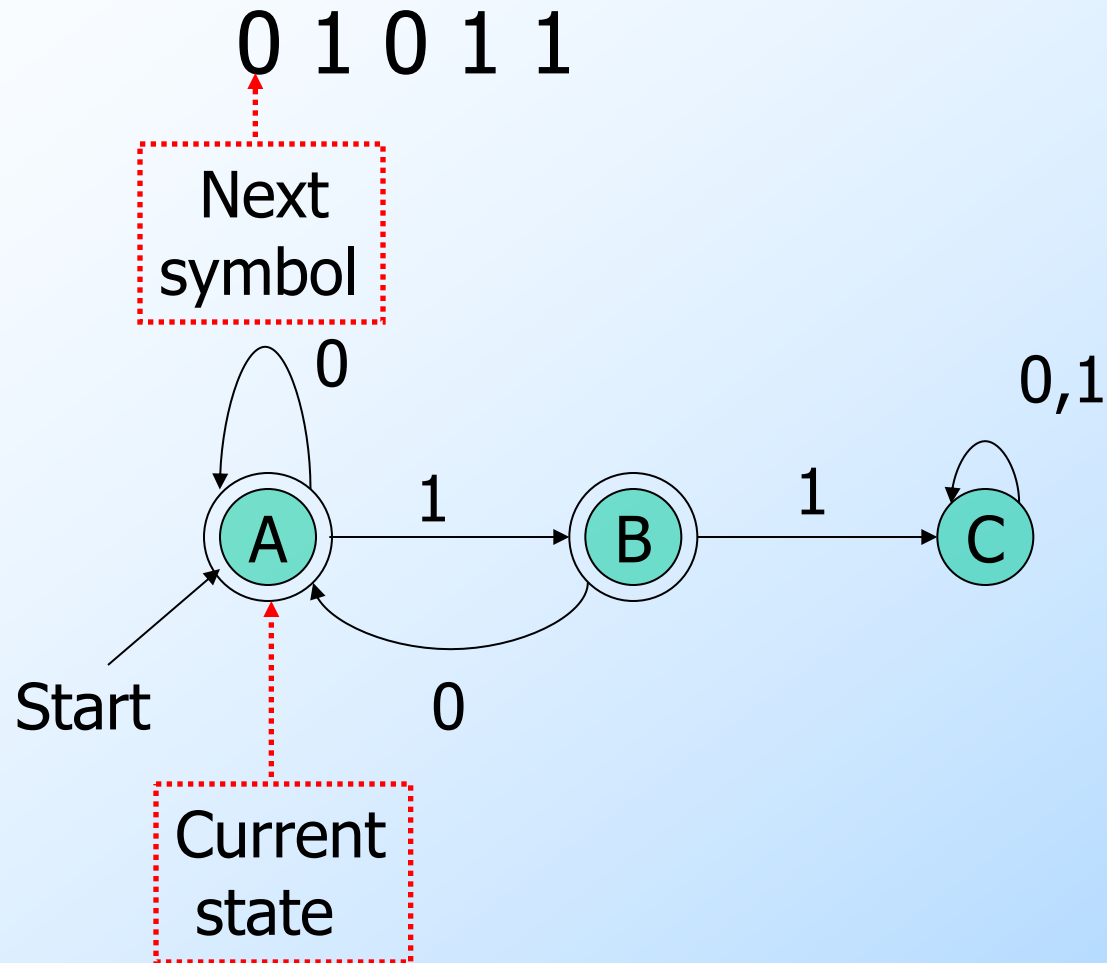
Veremos en un momento que el lenguaje  $L_1 = \{0^k 1^k \mid k \geq 0\}$  no es regular.

- ◆  $L_2 = \{\text{cadenas de 0's y 1's con el mismo número de 0's y de 1's}\}$  tampoco es regular (pero es más difícil de probar).
- ◆ Lenguajes regulares son cerrados bajo  $\cap$ .
- ◆ Si  $L_2$  fuera regular, entonces  $L_2 \cap L(\mathbf{0^*1^*}) = L_1$  sería regular, pero no lo es.

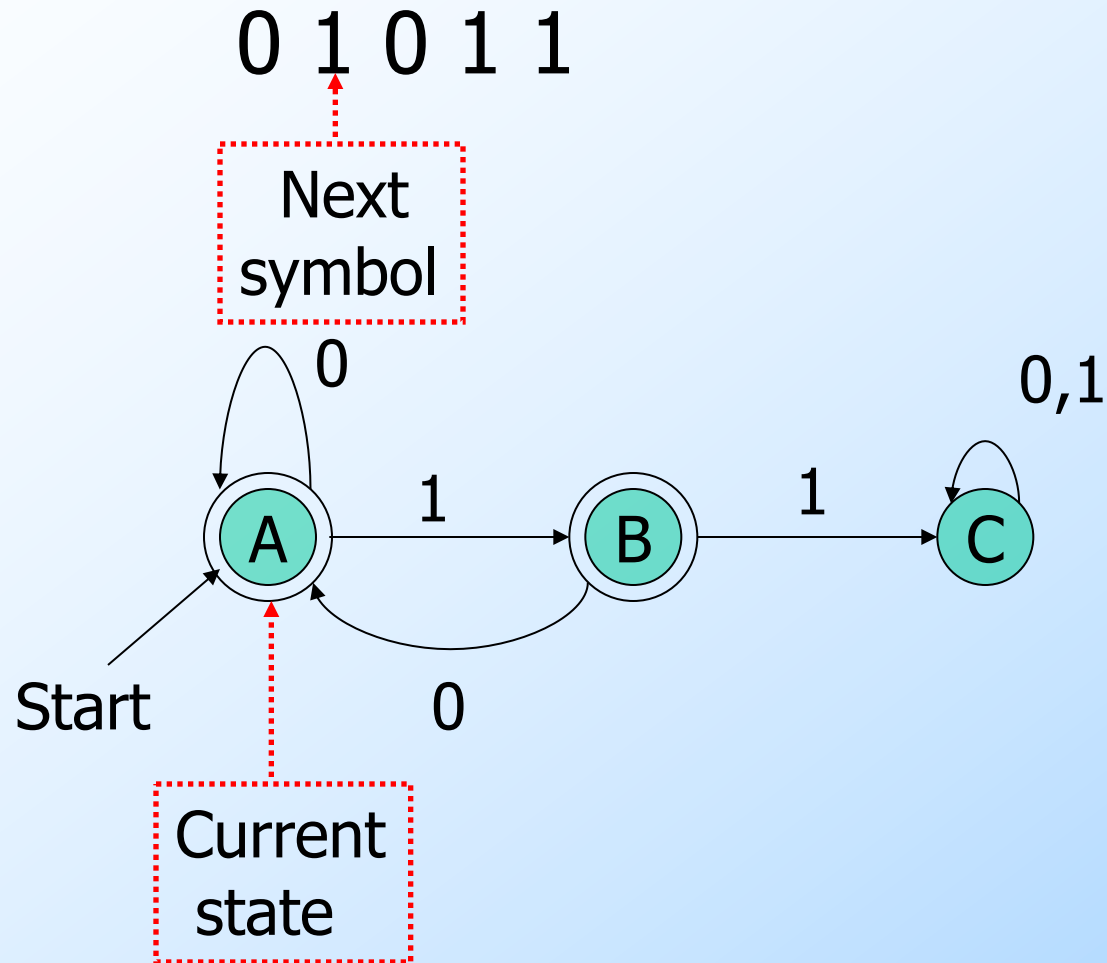
# Pertenencia

- ◆ Nuestra primera propiedad de decisión es la pregunta: “está la cadena  $w$  en el lenguaje regular  $L$ ?”
- ◆ Asumir que  $L$  es representado por un DFA, denotado  $M$ .
- ◆ Simular la acción del autómata  $M$  tomando como input en la secuencia de símbolos de  $w$ .

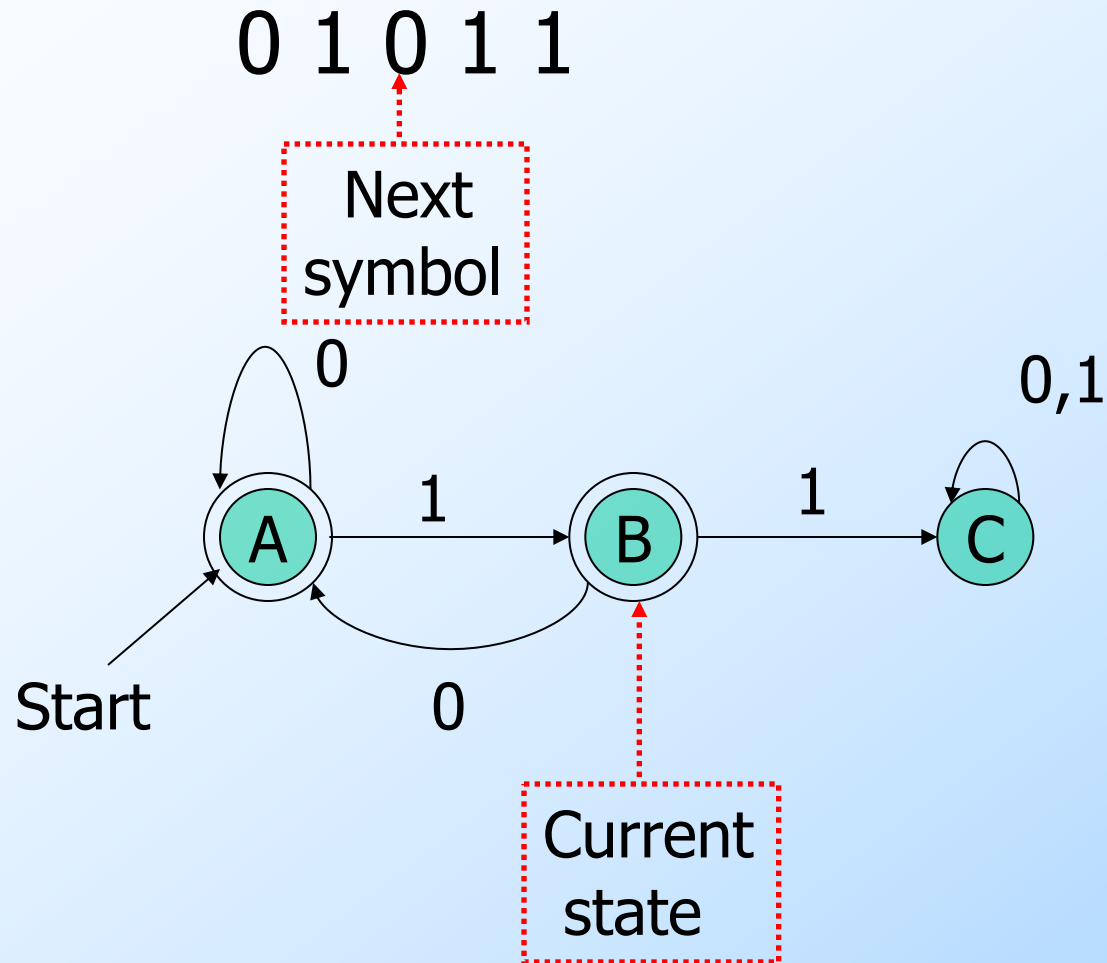
# Ejemplo: Pertenencia



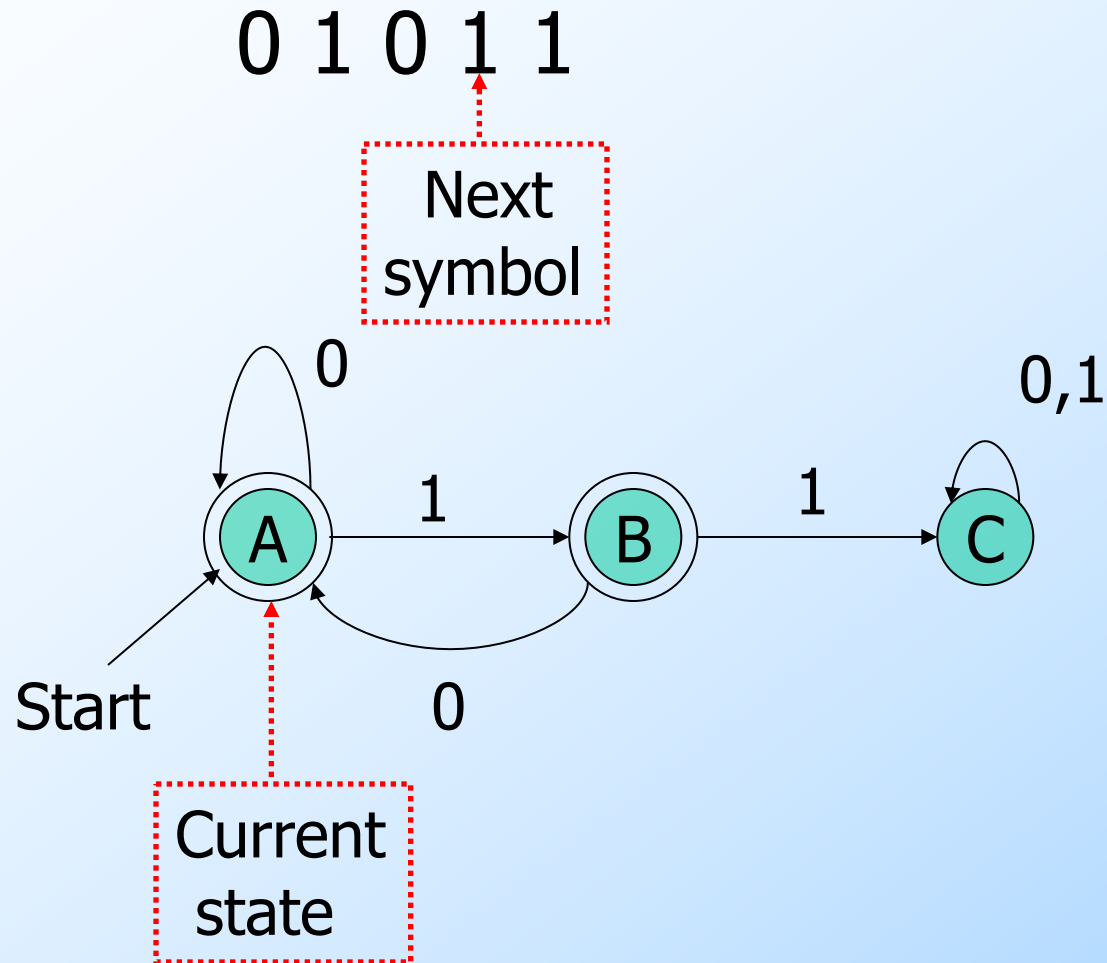
# Ejemplo: Pertenencia



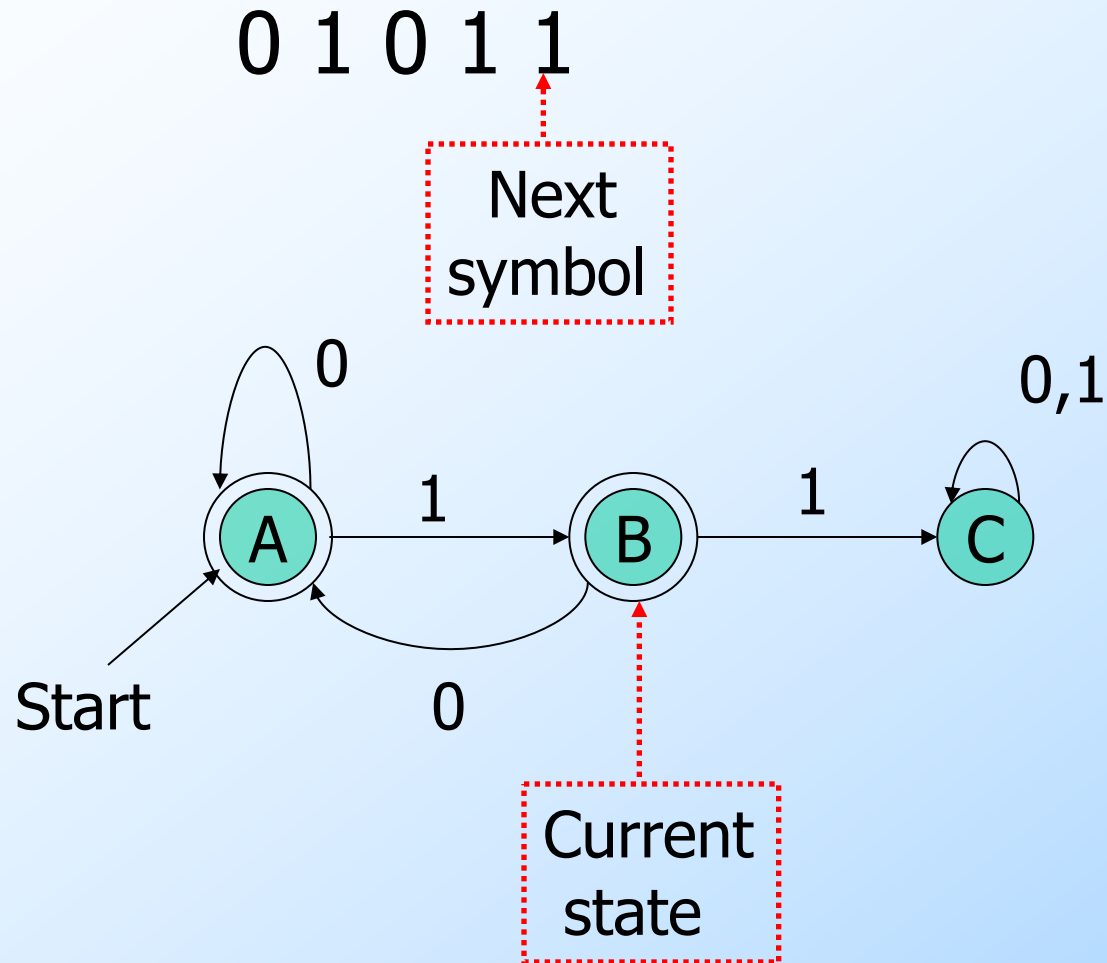
# Ejemplo: Pertenencia



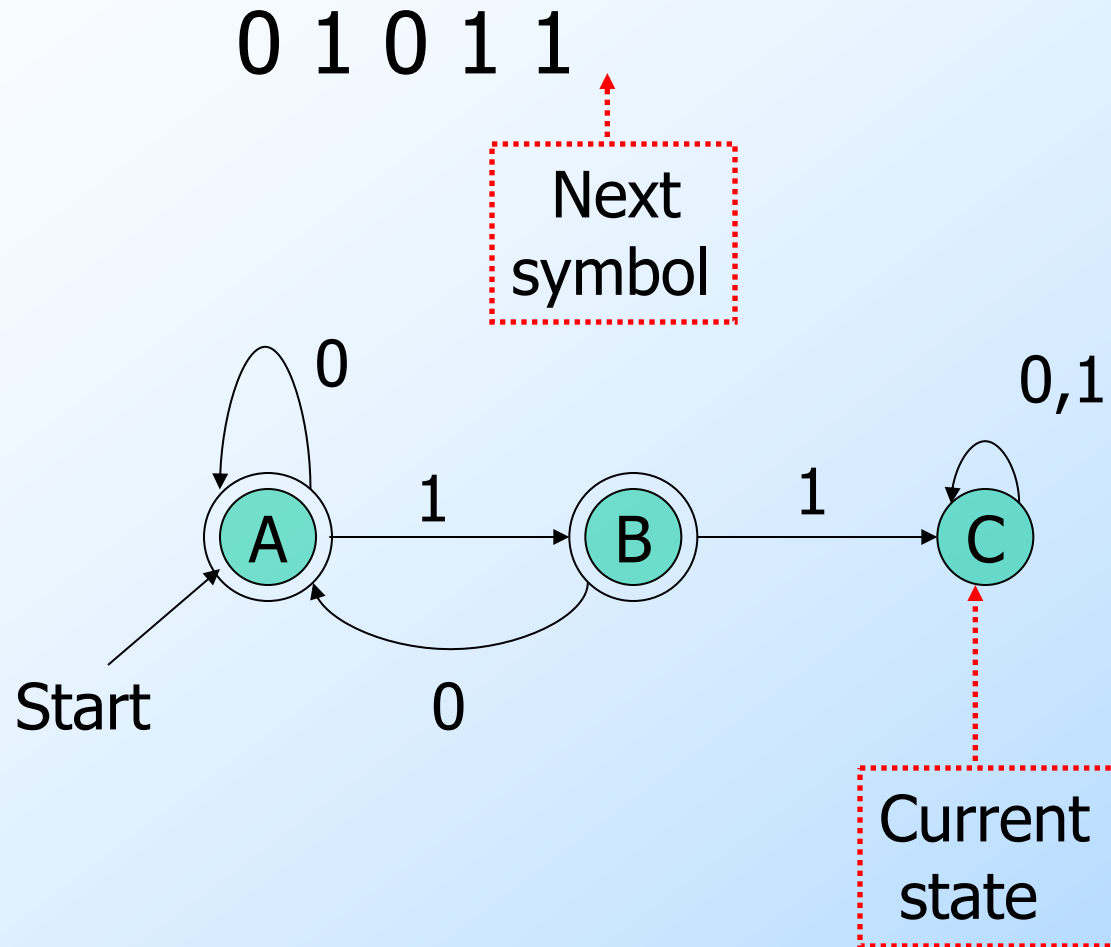
# Ejemplo: Pertenencia



# Ejemplo: Pertenencia



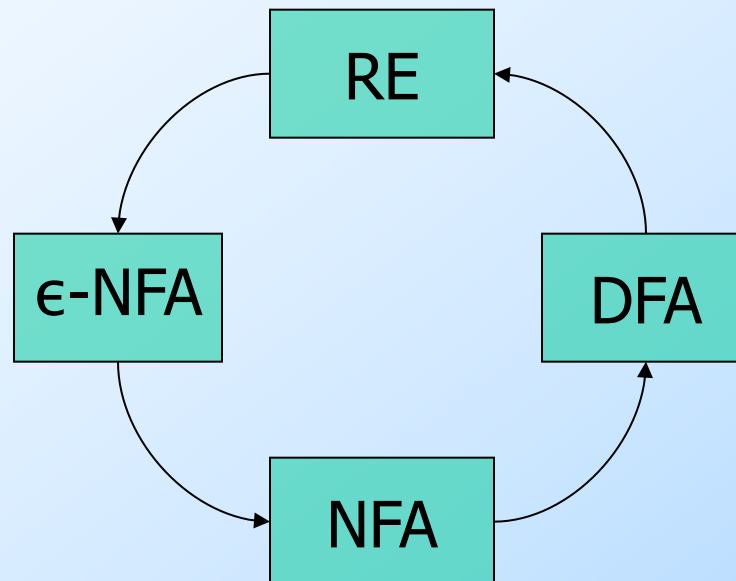
# Ejemplo: Pertenencia





# Y si $L$ no es representado por un DFA?

- ◆ Recordemos que tenemos un ciclo de equivalencias para representar lenguajes regulares:



# El problema de vacuidad

- ◆ Dado un lenguaje regular  $L$ , queremos saber si  $L$  contiene alguna palabra.
- ◆ Suponga que  $L$  se representa con un DFA.
- ◆ Construir el grafo de transiciones.
- ◆ Calcular el conjunto de estados alcanzables desde el estado inicial  $q_0$ .
- ◆ Si algún estado final  $q_f$  es alcanzable, sí hay palabras, caso contrario  $L = \{\}$ .

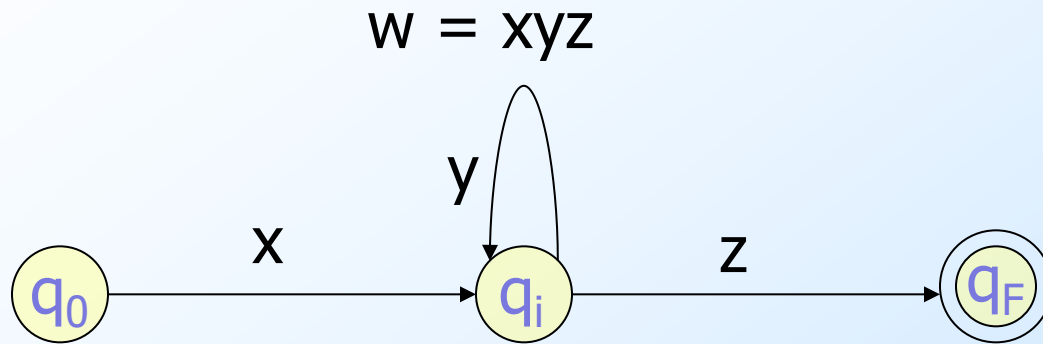
# El problema de la finitud

- ◆ Dado un lenguaje regular  $L$ , es infinito?
- ◆ Comenzar con un DFA para  $L$ .
- ◆ **Idea clave:** si el DFA tiene  $n$  estados, y el lenguaje contiene cualquier cadena de longitud  $n$  o mayor, entonces  $L$  es infinito.
- ◆ Caso contrario, el lenguaje es finito.
  - ▶ Limitado a cadenas de longitud menor a  $n$ .

# Prueba de la **idea clave**

- ◆ Si un DFA de  $n$  estados acepta una cadena  $w$  de longitud  $n$  o mayor, entonces debe haber un estado que aparece al menos dos veces en el trayecto de  $w$  desde el estado inicial  $q_0$  al estado final  $q_F$  de  $w$ .
- ◆ Esto ya que hay al menos  $n+1$  estados a lo largo del trayecto de  $w$ .

# Prueba de la idea clave



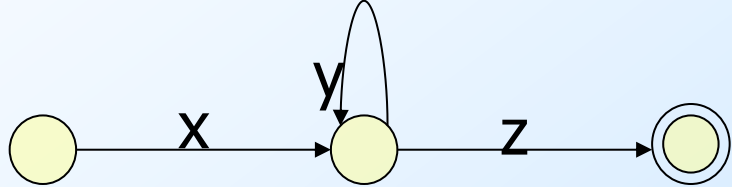
→  $xy^iz$  está en el lenguaje, para todo  $i \geq 0$ .

Como  $y$  no es la cadena  $\epsilon$ , todas las cadenas  $xy^iz$  (hay infinitas de ellas) están en  $L$ .

# Finitud

- ◆ Aún no tenemos un algoritmo.
- ◆ Hay un número infinito de cadenas de longitud  $> n$ . No podemos testarlas todas.
- ◆ **Segunda idea clave:** si hay una cadena de longitud  $\geq n$  (= número de estados) en  $L$ , entonces debe haber una cadena de longitud entre  $n$  y  $2n-1$ .

# Pueba de la 2ª idea clave

- ◆ Recordemos: 
- ◆ Podemos elegir  $y$  como el primer ciclo a lo largo del trayecto de  $w$ .
- ◆ Así,  $|xy| \leq n$ ; en particular,  $1 \leq |y| \leq n$ .
- ◆ Luego, si  $w$  es de longitud  $2n$  o mayor, hay una cadena de menor longitud en  $L$  que aún es de longitud  $n$  o más.
- ◆ Reducir hasta obtener algo en  $[n, 2n-1]$ .

# Completamos el algoritmo de infinitud

- ◆ Verificar la pertenencia a  $L$  de todas las cadenas de longitudes entre  $n$  y  $2n-1$ .
  - ▶ Si alguna es aceptada, entonces  $L$  es infinito. Caso contrario,  $L$  es finito.
- ◆ El peor algoritmo posible.
- ◆ **Mejor idea:** buscar la existencia de ciclos entre el estado inicial  $q_0$  y final  $q_F$ .



# Búsqueda de Ciclos

1. Eliminar los estados que no son alcanzables desde el estado inicial  $q_0$ .
2. Eliminar los estados que no llegan al estado final  $q_F$ .
3. Verificar si el grafo de transiciones remanente posee algún ciclo.

# El Lema de Bombeo

## *(Pumping Lemma)*

- ◆ En lo anterior casi hemos probado, de forma accidental, un resultado que es muy útil para mostrar que ciertos lenguajes no son regulares.
- ◆ Este es llamado el *lema de bombeo para lenguajes regulares*.

# Lema de Bombeo

Número de  
estados del  
DFA para L

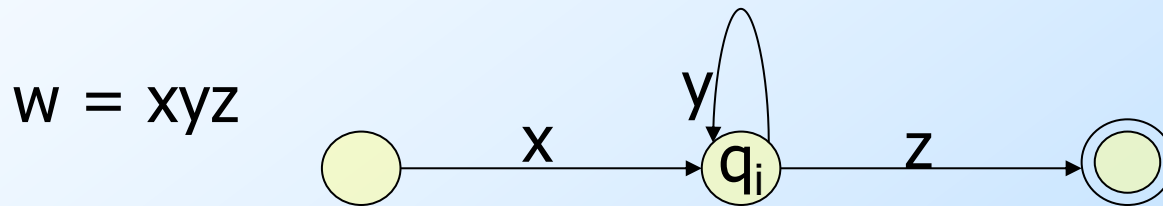
Para todo lenguaje regular L,  
existe un entero  $n \geq 1$ , tal que  
para toda cadena  $w \in L$  de longitud  $\geq n$   
podemos escribir  $w = xyz$ , donde:

1.  $|xy| \leq n$ .
2.  $|y| > 0$ .
3. Para todo  $i \geq 0$ ,  $xy^iz \in L$ .

y corresponde  
al primer ciclo en  
el trayecto de w

# Prueba de Lema de Bombeo

1. Tomamos  $n = \text{número de estados del DFA para } L$
2. Tomamos  $w \in L$  de longitud  $\geq n$
3. Por el principio de las casillas en el trayecto de  $w$  hay  $\geq n+1$  estados, de modo que al menos un estado  $q_i$  se repite.
4. Tome  $y$  la subcadena de  $w$  del ciclo en  $q_i$



(Observe que  $y$  tiene longitud al menos 1, no es la cadena vacía).

5.  $\rightarrow xy^iz$  está en el lenguaje, para todo  $i \geq 0$ .
6. Como  $y$  no es la cadena  $\epsilon$ , todas las cadenas de la forma  $xy^iz$  (hay infinitas de ellas) están en  $L$ .

# Ejemplo: Lema de Bombeo

Vamos a mostrar que  $L = \{0^k1^k: k \geq 1\}$  no es un lenguaje regular.

- ◆ Suponga que sí es. Entonces existe un  $n \geq 1$  para  $L$  que cumple el lema de bombeo.
- ◆ Tome  $w = 0^n1^n \in L$ . Podemos escribir  $w = xyz$ , donde  $|xy| \leq n$ ,  $|y| > 0$ . Esto implica que:
  1.  $y \neq \varepsilon$ .
  2.  $x, y$  consisten sólo de 0's
- ◆ Pero, por el lema de bombeo, la cadena  $xyyz \in L$ .
- ◆ Pero esta cadena tiene  $(n+1)$  0's y  $n$  1's. Absurdo!