

# Lenguajes regulares



UNIVERSIDAD  
DE CHILE

Arturo Merino



**Recuerdo / Plan**

# Recuerdo/Plan

- **Recuerdo:**

- Mostrar equivalencia AFD-AFND
- Operaciones y expresiones regulares
- $\sigma, \epsilon, \emptyset, R_1|R_2, (R)^*, R_1R_2$ .

- **Plan:**

- Equivalencia ER - Lenguajes regulares
- Recapitulación sobre lenguajes regulares

# Equivalencia ER-AFD-AFND

# Equivalencia ER - lenguaje regular

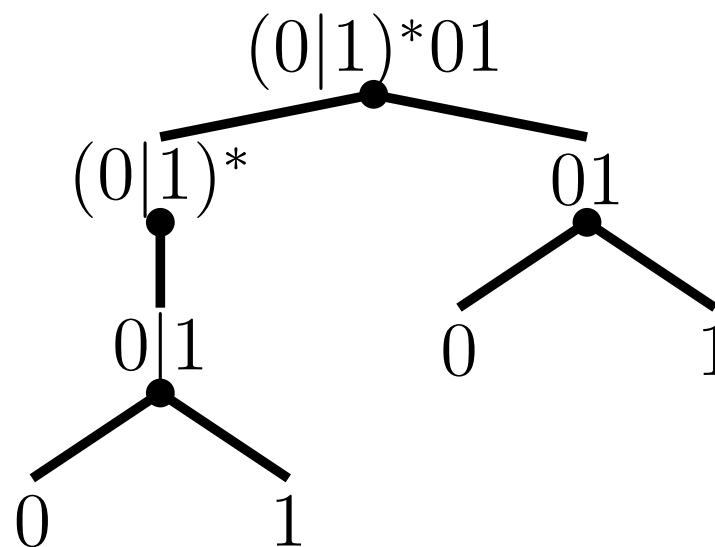
Teo.

Sea  $L \subseteq \Sigma^*$  un lenguaje.

$L$  es regular si y sólo si existe una expresión regular  $R$  tal que  $\mathcal{L}(R) = L$ .

( $\Leftarrow$ ) Para toda expresión regular  $R$ , PDQ  $\mathcal{L}(R)$  es regular.

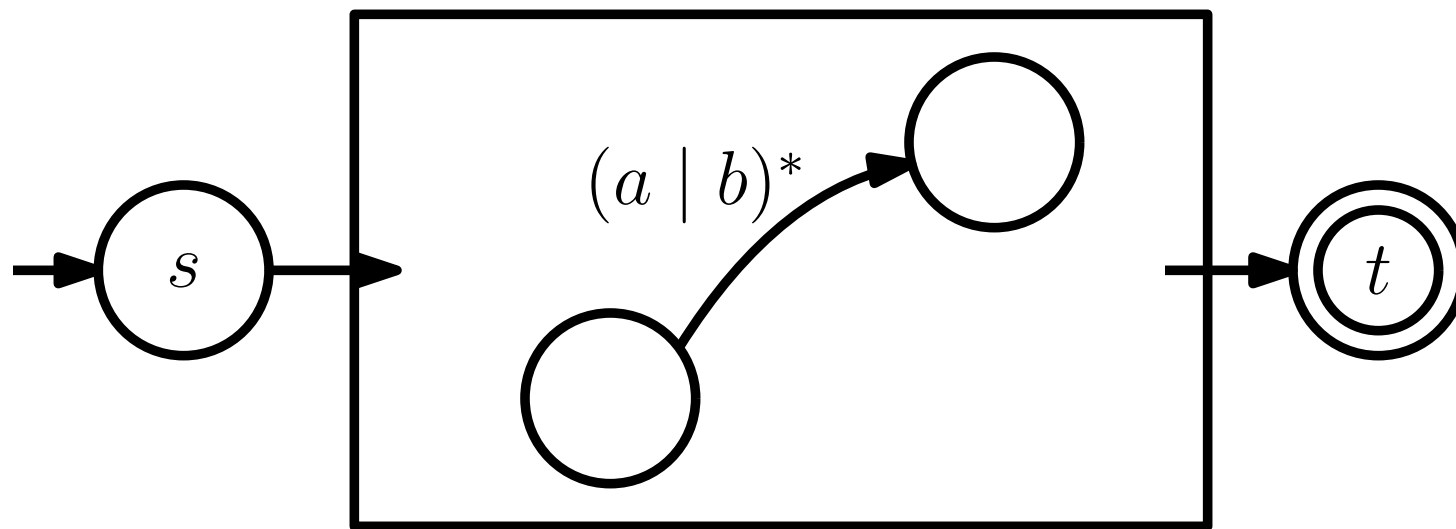
- Demostración sobre estructura recursiva...
- ¡Inducción estructural!
- Nos da un algoritmo
- **Ej:**  $(0|1)^*01$



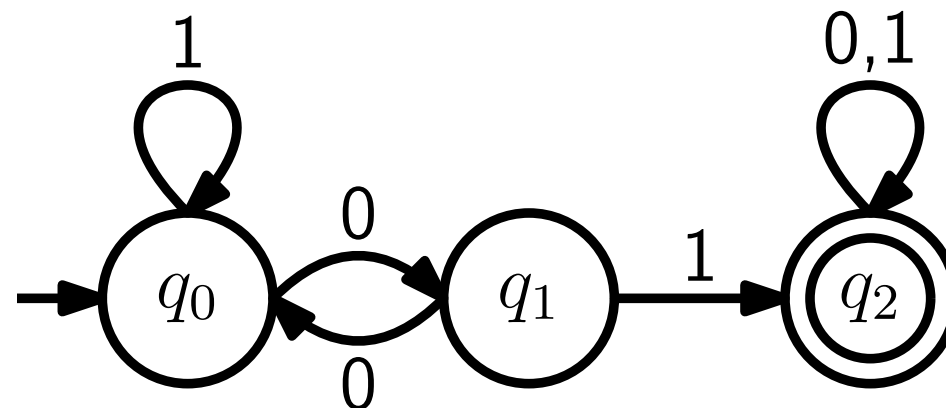
## De AFD a ER

( $\implies$ ) PDQ Todo AFD  $M$ , tiene una expresión regular equivalente

- **Idea:** autómatas con expresiones regulares



- Eliminar los estados intermedios uno a uno.
- Ej:



# Autómata generalizado

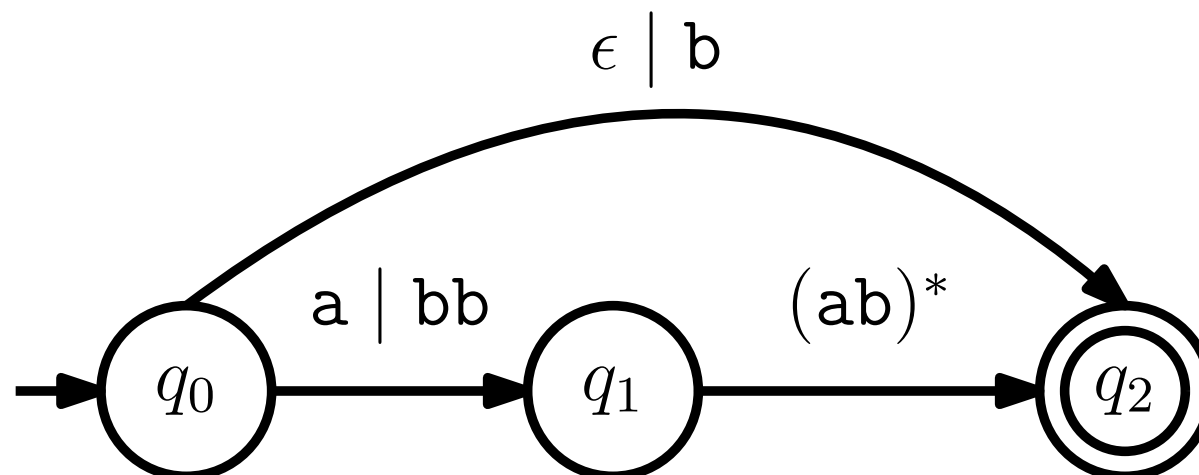
## Def.

Un *autómata finito generalizado (AFG)* es una 5-tupla

$M = (Q, \Sigma, \delta, q_0, q_{\text{fin}})$ , donde

- $Q$  es un conjunto finito de *estados*
- $\Sigma$  es un conjunto finito llamado *alfabeto*
- $\delta: (Q \setminus \{q_{\text{fin}}\}) \times (Q \setminus \{q_0\}) \rightarrow \text{RegExp}(\Sigma)$  la *función de transición*
- $q_0 \in Q$  es el *estado inicial*, y
- $q_{\text{fin}} \in Q$  es el *estado de aceptación (o final)*.

• Ej:



# Computación

## Def.

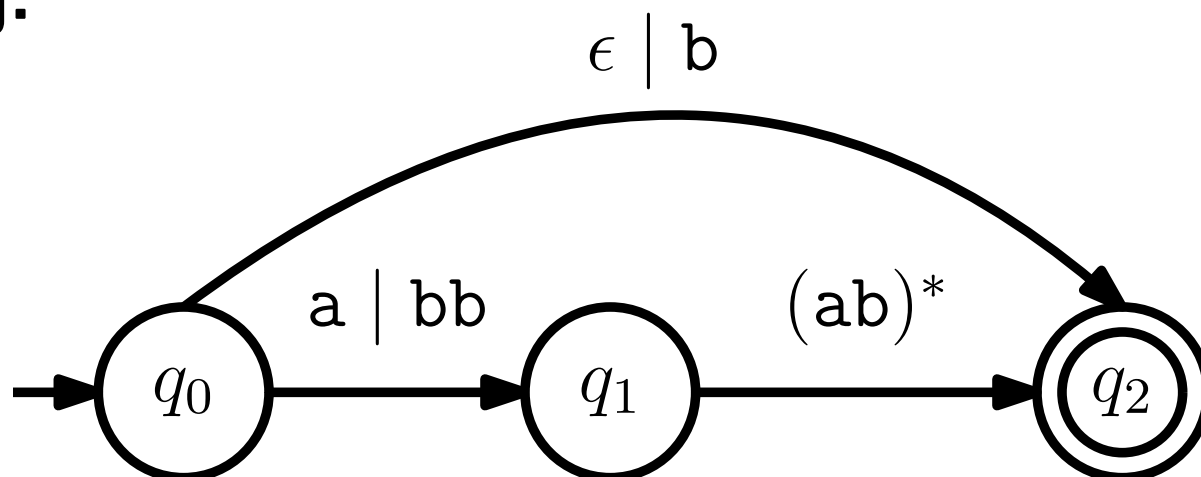
Sea  $M = (Q, \Sigma, \delta, q_0, q_{\text{fin}})$  un AFND y  $w \in \Sigma^*$  un string.

Diremos que  $M$  *acepta*  $w$  si **podemos escribir**  $w = x_1 x_2 \dots x_m$  **donde cada**  $x_i \in \Sigma^*$  **y existen estados**  $r_0, \dots, r_m \in Q$  **tal que**

- $r_0 = q_0$  (partimos en el estado inicial)
- $x_i \in \mathcal{L}(\delta(r_{i-1}, r_i))$  para todo  $i \in \{1, \dots, m\}$  (transiciones válidas)
- $r_m = q_{\text{fin}}$  (terminamos en aceptación)

- La secuencia **certifica**.

- Ej:

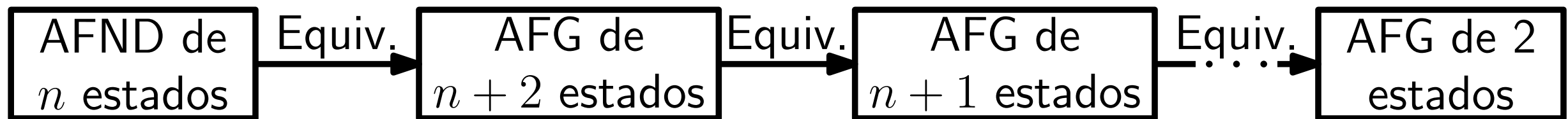


$w = \text{bbabababababababab}$

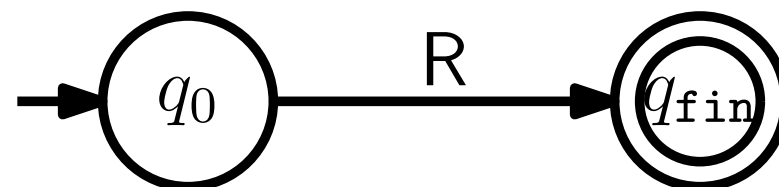
## De AFND a ER

( $\implies$ ) PDQ Todo AFD  $M$ , tiene una expresión regular equivalente

- Todo lenguaje regular tiene un AFND.
- ¿De AFND a AFG?
  - ¿Único estado final? ¿Estado final sin aristas salientes?
  - ¿Estado inicial sin aristas entrantes?
- Se arregló agregando (a lo más) dos estados.



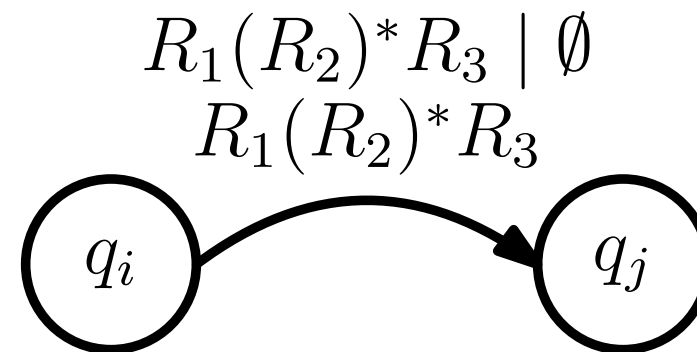
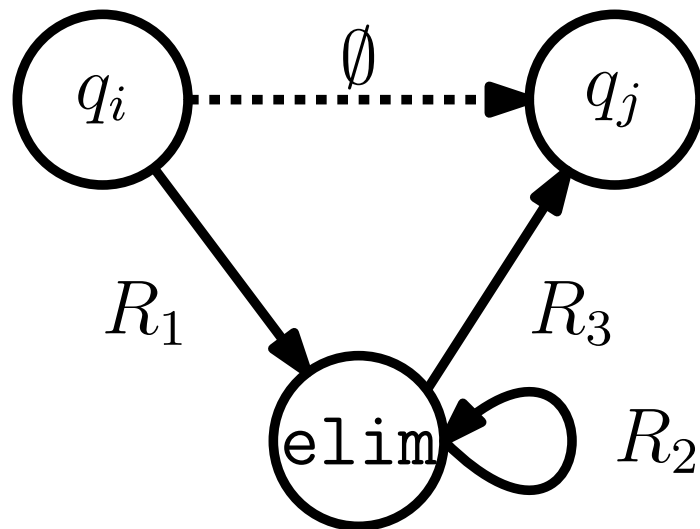
- ¿Cómo se ve un AFG de dos estados?



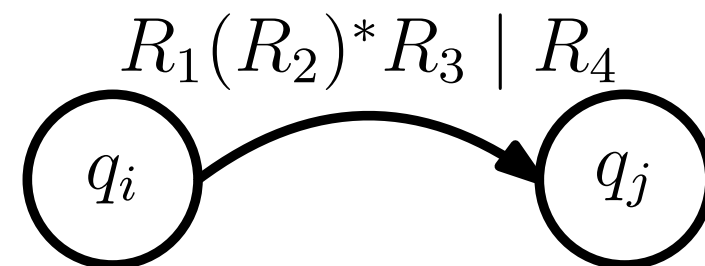
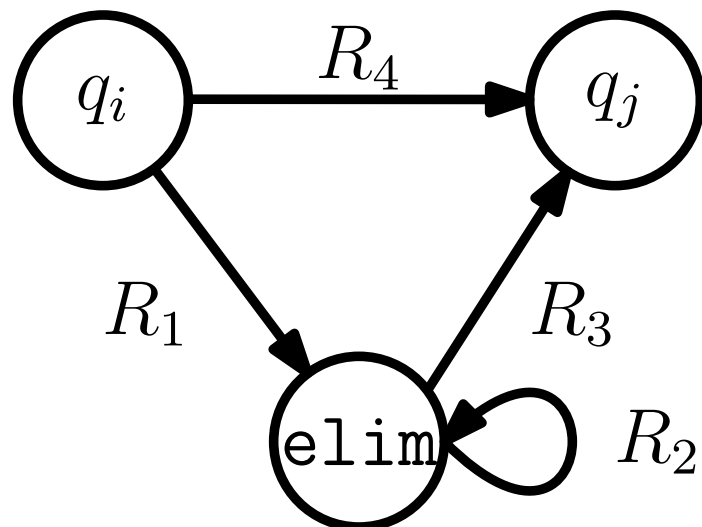
- $R$  era la expresión buscada.

# Eliminación de estados

- Eliminar estado `elim`
- Actualizar  $q_i$  y  $q_j$  si hay conexión entre  $q_i$  a `elim` y de  $q_j$  a `elim`
  - No hay conexión de  $q_i$  a  $q_j$



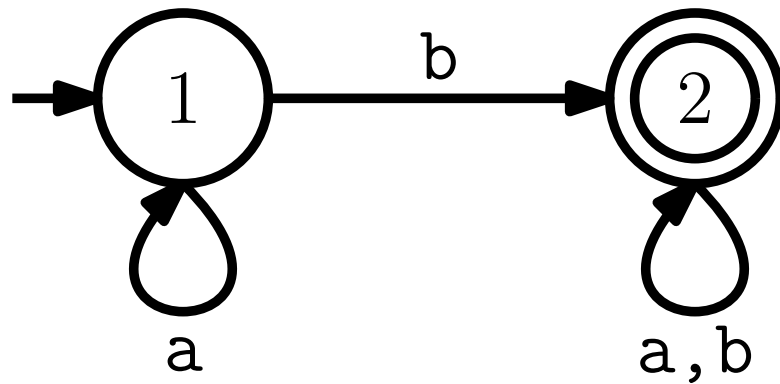
- Hay conexión de  $q_i$  a  $q_j$



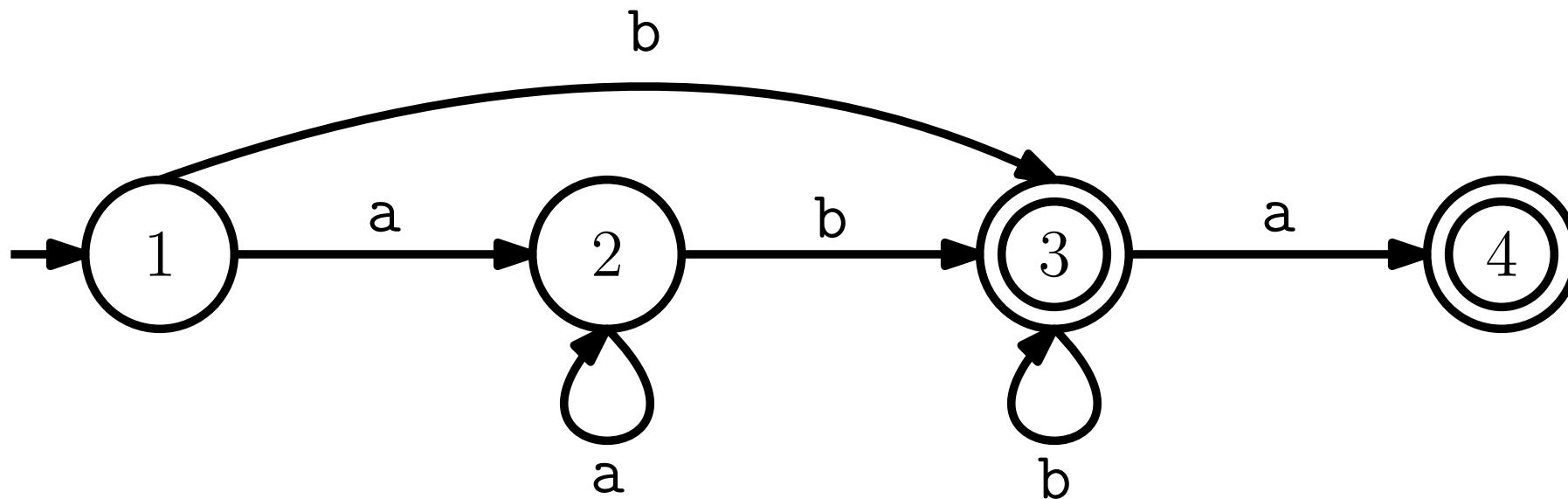
- Mismo caso si uno interpreta no hay conexión como  $\emptyset$ .

# Ejemplos

- Convertir a ER



- Convertir a ER



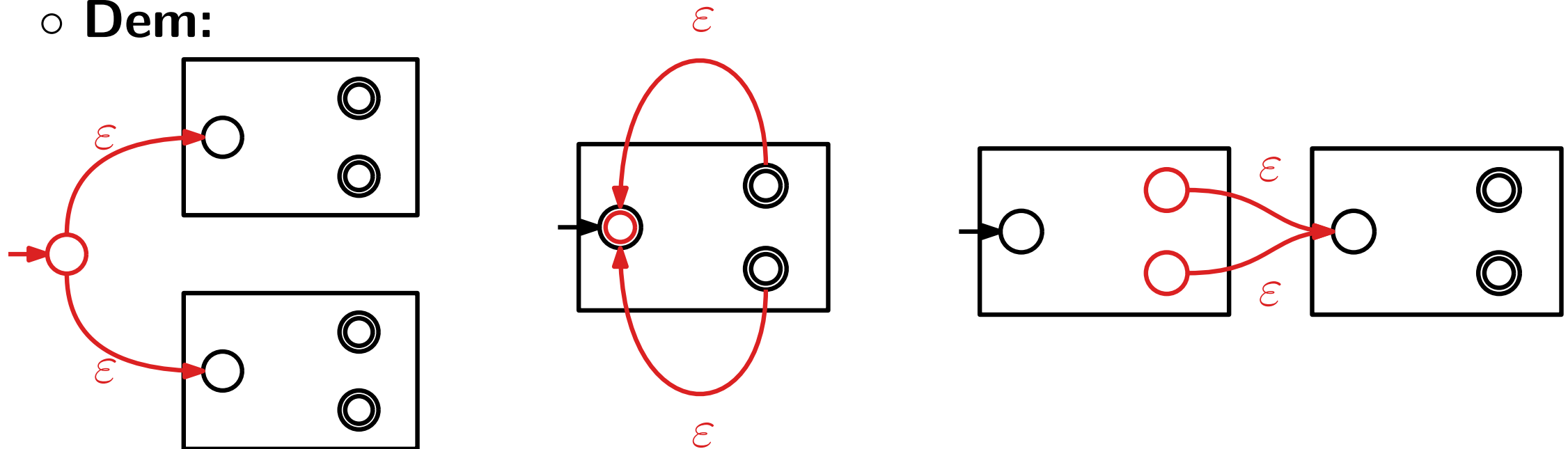
# Recapitulación sobre lenguajes regulares

## Tres perspectivas

- **Lenguaje regular:** aceptado por AFD, AFND, ER.
- $\text{AFND} \rightarrow \text{AFD}$ .
  - Quitar las transiciones  $\varepsilon$  vía alcanzables.
  - Construcción en  $2^Q$  simulando todos los posibles estados.
- $\text{ER} \rightarrow \text{AFND}$ .
  - Usar la definición recursiva de las ER.
- $\text{AFD}, \text{AFND} \rightarrow \text{ER}$ 
  - Vía AFG, partir agregando nuevos estados iniciales y finales.
  - Eliminar estados intermedios uno a uno.
- $\text{ER} \rightarrow \text{AFD}$ 
  - De ER a AFND y de AFND a ER.
- $\text{AFND} \rightarrow \text{AFD}$

# Mostrando que lenguajes son regulares

- Diseñar AFD, AFND o ER.
- + propiedades de cerradura
- Complemento: **no**.
  - **Dem:** cambiar aceptación/no aceptación en un AFD.
- Intersección y unión: **y/o**.
  - **Dem:** correr los AFD en paralelo vía el producto.
- Operaciones regulares: unión, concatenación y estrella.
  - **Dem:**



**Dudas/Preguntas**

# Wrap-Up

- **Próx. clase:**
  - Lenguajes no regulares