Lenguajes regulares

UNIVERSIDAD DE CHILE

Arturo Merino

Recuerdo / Plan

Recuerdo/Plan

- **Recuerdo:**

Recuerdo/Plan

- **Recuerdo:**
  - Mostrar equivalencia AFD-AFND

Recuerdo/Plan

- **Recuerdo:**
  - Mostrar equivalencia AFD-AFND
  - Operaciones y expresiones regulares

Recuerdo/Plan

- **Recuerdo:**
  - Mostrar equivalencia AFD-AFND
  - Operaciones y expresiones regulares
  - σ, ϵ, $\emptyset$, $R_1|R_2$, $(R)^*$, R_1R_2.

Recuerdo/Plan

- **Recuerdo:**
  - Mostrar equivalencia AFD-AFND
  - Operaciones y expresiones regulares
  - σ, ϵ, $\emptyset$, $R_1|R_2$, $(R)^*$, R_1R_2.

- **Plan:**

Recuerdo/Plan

- **Recuerdo:**
  - Mostrar equivalencia AFD-AFND
  - Operaciones y expresiones regulares
  - σ, ϵ, $\emptyset$, $R_1|R_2$, $(R)^*$, R_1R_2.

- **Plan:**
  - Equivalencia ER - Lenguajes regulares

Recuerdo/Plan

- **Recuerdo:**
  - Mostrar equivalencia AFD-AFND
  - Operaciones y expresiones regulares
  - σ, ϵ, $\emptyset$, $R_1|R_2$, $(R)^*$, 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.

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 $\mathcal{L}(R) = L$.

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 $\mathcal{L}(R) = L$.

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

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 $\mathcal{L}(R) = L$.

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

- Demostración sobre estructura recursiva...

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 $\mathcal{L}(R) = L$.

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

- Demostración sobre estructura recursiva...
- ¡Inducción estructural!

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 $\mathcal{L}(R) = L$.

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

- Demostración sobre estructura recursiva...
- ¡Inducción estructural!
- Nos da un algoritmo

Equivalencia ER - lenguaje regular

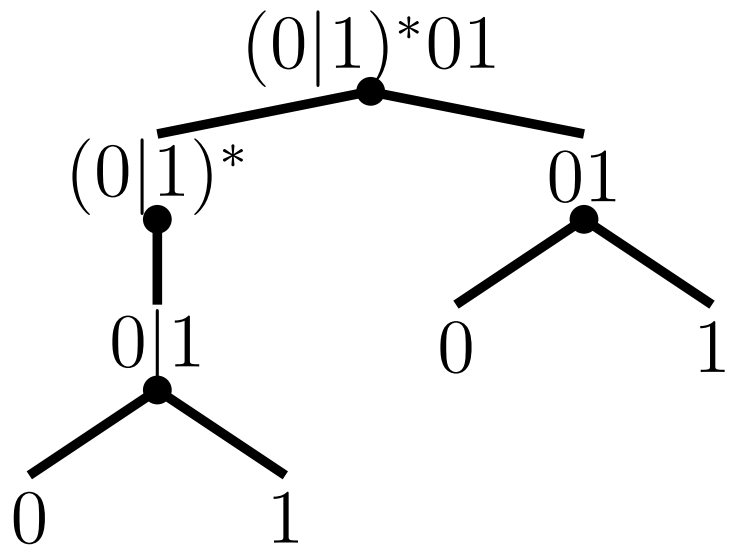
Teo.

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

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

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

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



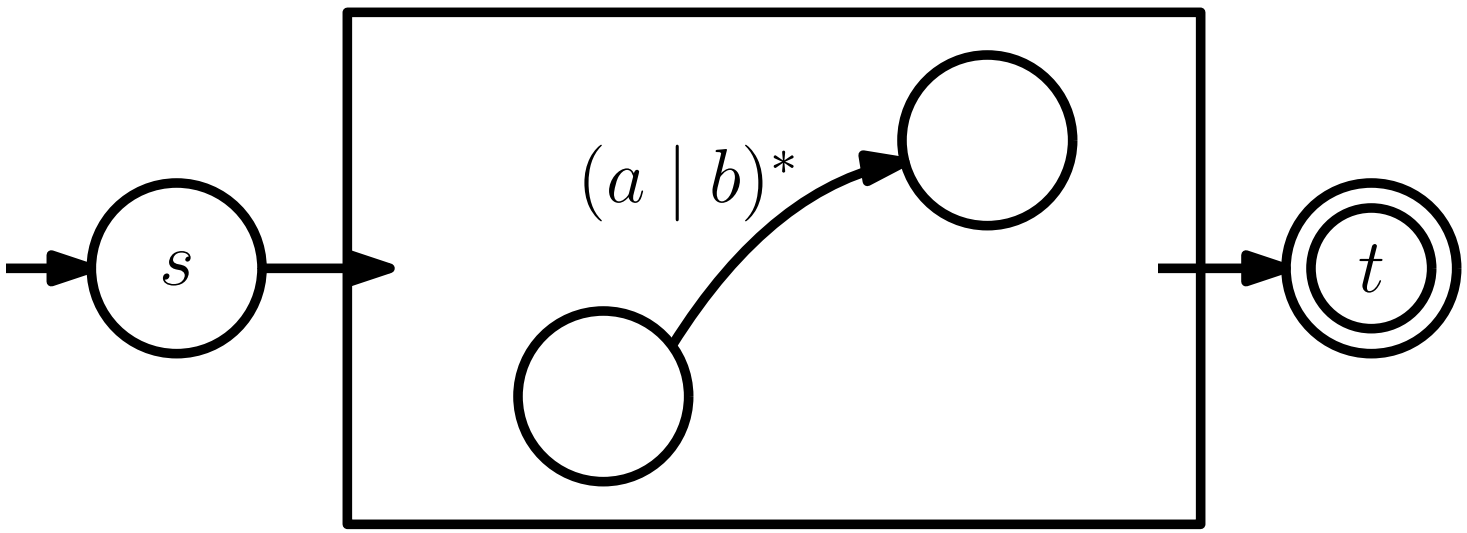
De AFD a ER

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

De AFD a ER

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

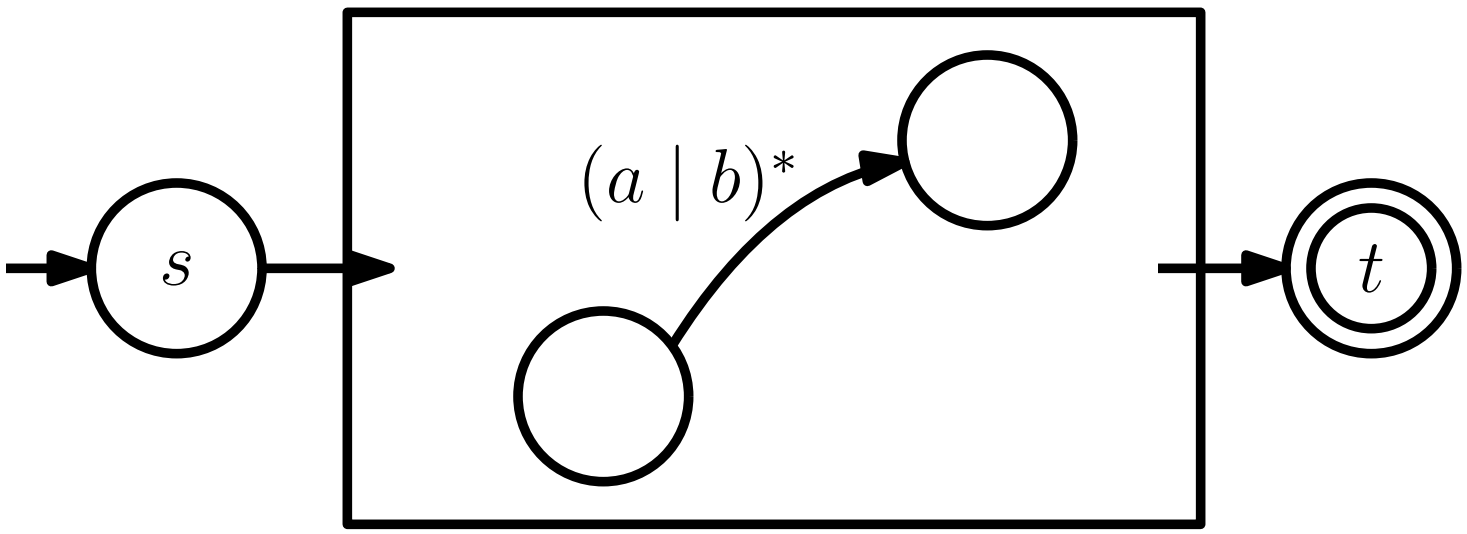
- **Idea:** autómatas con expresiones regulares



De AFD a ER

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

- **Idea:** autómatas con expresiones regulares

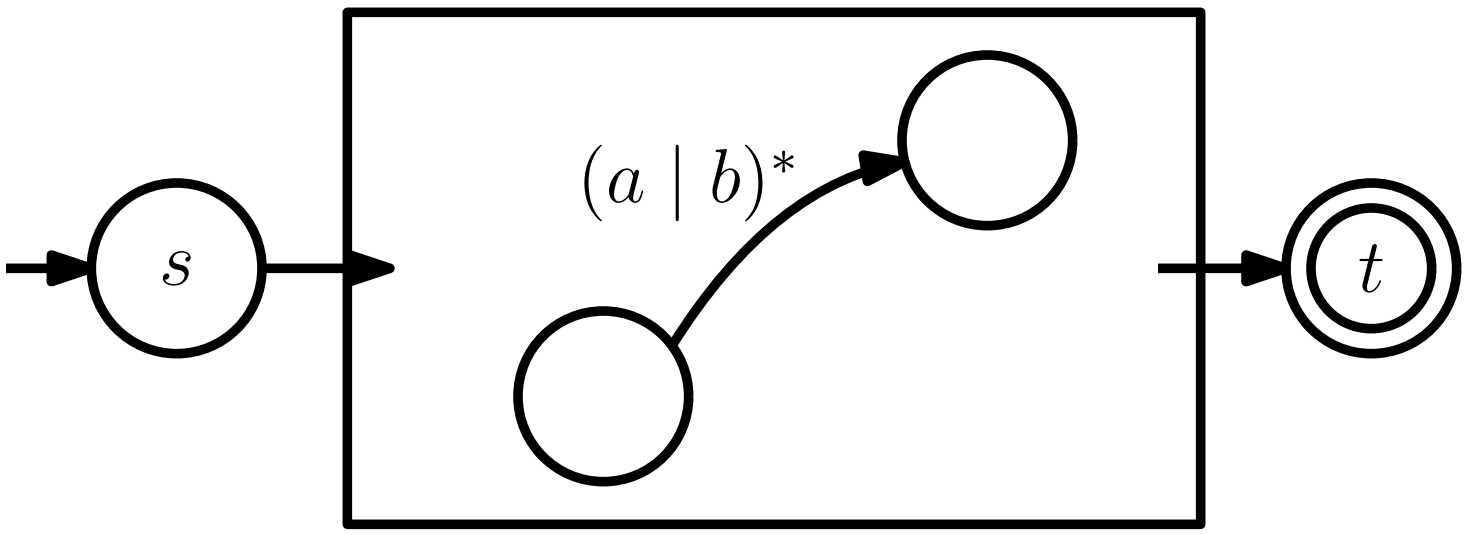


- Eliminar los estados intermedios uno a uno.

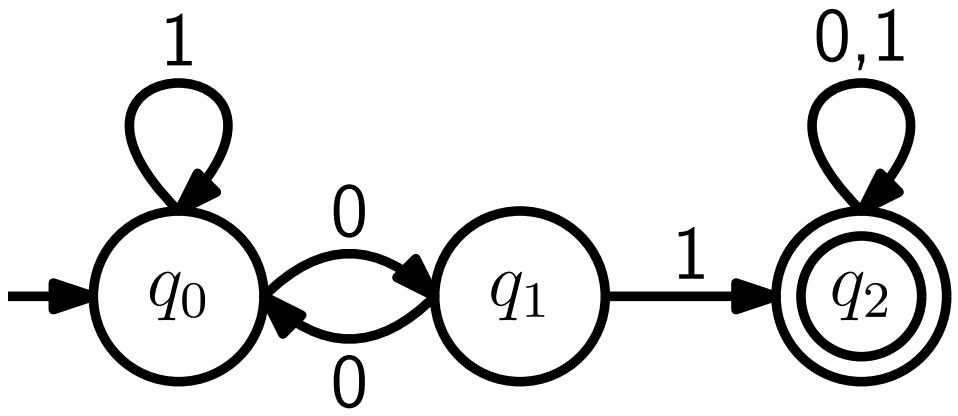
De AFD a ER

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

Autómata generalizado

Def.

Un *autómata finito generalizado (AFG)* es una 5-tupla $M = (Q, \Sigma, \delta, q_0, q_{\texttt{fin}})$, donde

- Q es un conjunto finito de *estados*

Autómata generalizado

Def.

Un *autómata finito generalizado (AFG)* es una 5-tupla $M = (Q, \Sigma, \delta, q_0, q_{\texttt{fin}})$, donde

- Q es un conjunto finito de *estados*
- Σ es un conjunto finito llamado *alfabeto*

Autómata generalizado

Def.

Un *autómata finito generalizado (AFG)* es una 5-tupla $M = (Q, \Sigma, \delta, q_0, q_{\texttt{fin}})$, donde

- Q es un conjunto finito de *estados*
- Σ es un conjunto finito llamado *alfabeto*
- $\delta \colon (Q \setminus \{q_{\texttt{fin}}\}) \times (Q \setminus \{q_0\}) \to \texttt{RegExp}(\Sigma)$ la *función de transición*.

Autómata generalizado

Def.

Un *autómata finito generalizado (AFG)* es una 5-tupla $M = (Q, \Sigma, \delta, q_0, q_{\texttt{fin}})$, donde

- Q es un conjunto finito de *estados*
- Σ es un conjunto finito llamado *alfabeto*
- $\delta \colon (Q \setminus \{q_{\texttt{fin}}\}) \times (Q \setminus \{q_0\}) \to \texttt{RegExp}(\Sigma)$ la *función de transición*.
- $q_0 \in Q$ es el *estado inicial*, y

Autómata generalizado

Def.

Un *autómata finito generalizado (AFG)* es una 5-tupla $M = (Q, \Sigma, \delta, q_0, q_{\texttt{fin}})$, donde

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

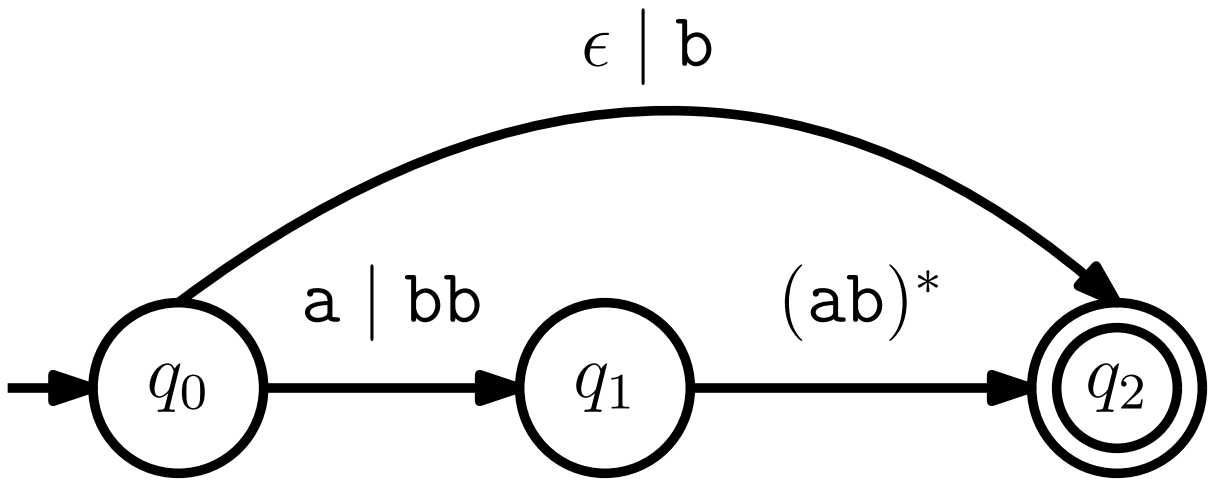
Autómata generalizado

Def.

Un *autómata finito generalizado (AFG)* es una 5-tupla $M = (Q, \Sigma, \delta, q_0, q_{\texttt{fin}})$, donde

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

- **Ej:**



Computación

Def.

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

Computación

Def.

Sea $M = (Q, \Sigma, \delta, q_0, q_{\texttt{fin}})$ 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

Computación

Def.

Sea $M = (Q, \Sigma, \delta, q_0, q_{\texttt{fin}})$ 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)

Computación

Def.

Sea $M = (Q, \Sigma, \delta, q_0, q_{\texttt{fin}})$ 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)

Computación

Def.

Sea $M = (Q, \Sigma, \delta, q_0, q_{\texttt{fin}})$ 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)
- $r_m = q_{\texttt{fin}}$ (terminamos en aceptación)

Computación

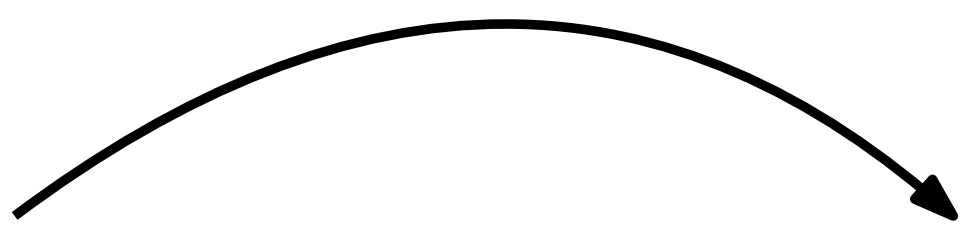
Def.

Sea $M = (Q, \Sigma, \delta, q_0, q_{\texttt{fin}})$ 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)
- $r_m = q_{\texttt{fin}}$ (terminamos en aceptación)

- La secuencia **certifica**.



Computación

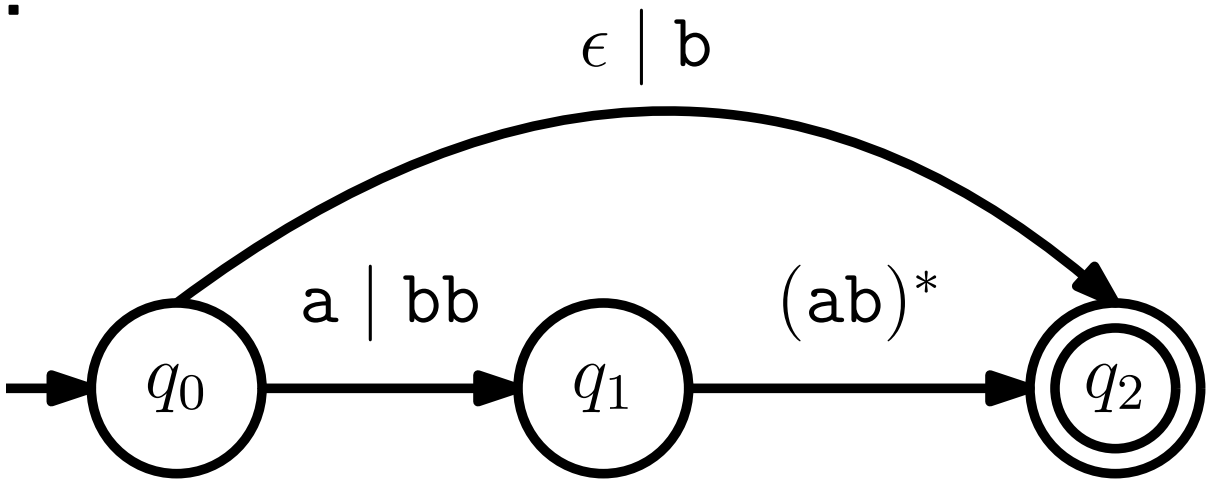
Def.

Sea $M = (Q, \Sigma, \delta, q_0, q_{\texttt{fin}})$ 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)
- $r_m = q_{\texttt{fin}}$ (terminamos en aceptación)

- La secuencia **certifica**.
- **Ej:**



$w = \texttt{bbabababababababababab}$

De AFND a ER

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

De AFND a ER

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

- Todo lenguaje regular tiene un AFND.

De AFND a ER

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

- Todo lenguaje regular tiene un AFND.
- ¿De AFND a AFG?

De AFND a ER

($\Longrightarrow$) 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?

De AFND a ER

($\Longrightarrow$) 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?

De AFND a ER

($\Longrightarrow$) 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.

Equiv. Equiv. Equiv.

De AFND a ER

($\Longrightarrow$) 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.

De AFND a ER

($\Longrightarrow$) 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.

De AFND a ER

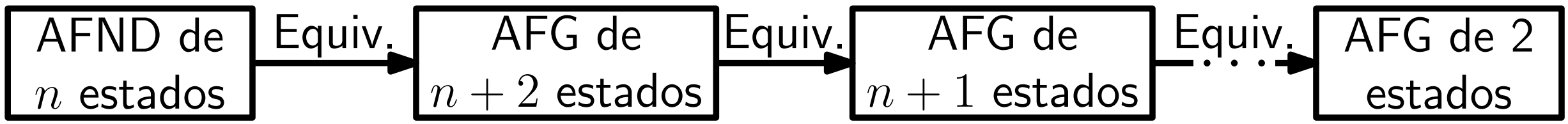
($\Longrightarrow$) PDQ Todo AFD M, tiene una expresión regular equivalente

- 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.

De AFND a ER

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

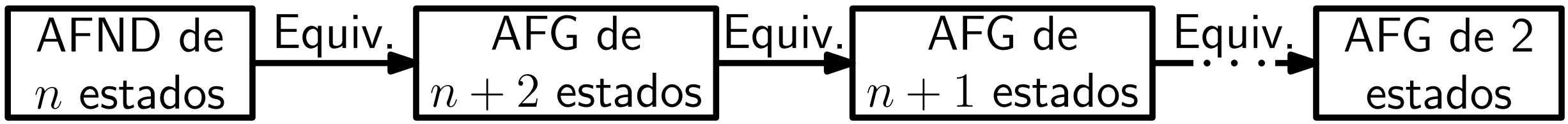
- Todo lenguaje regular tiene un AFND.
- ¿De AFND a AFG?
  - ¿Único estado final? ¿Estado final sin aristas salientes?
  - ¿Estado inicial sin aristas entrantes?
- Se arregló agregando (a lo más) dos estados.



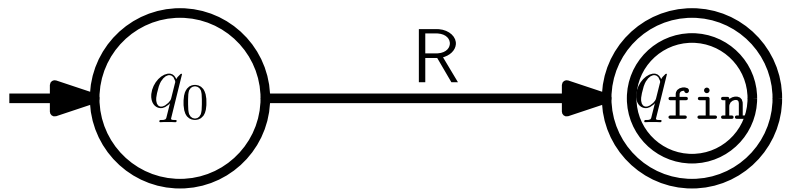
De AFND a ER

($\Longrightarrow$) 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.



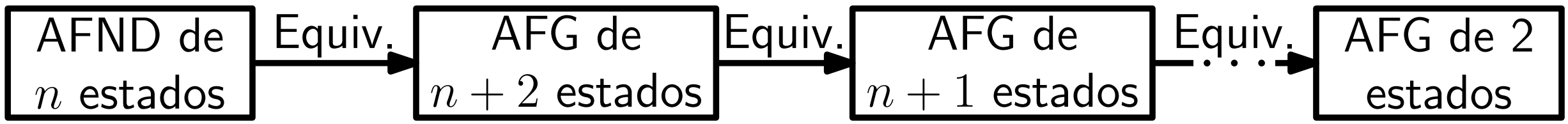
- ¿Cómo se ve un AFG de dos estados?



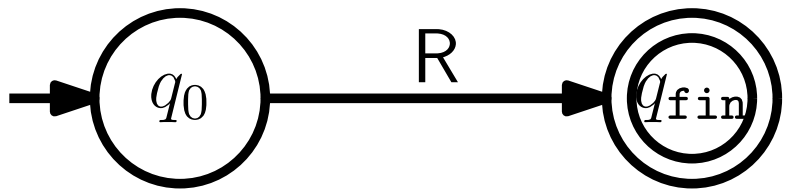
De AFND a ER

($\Longrightarrow$) 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

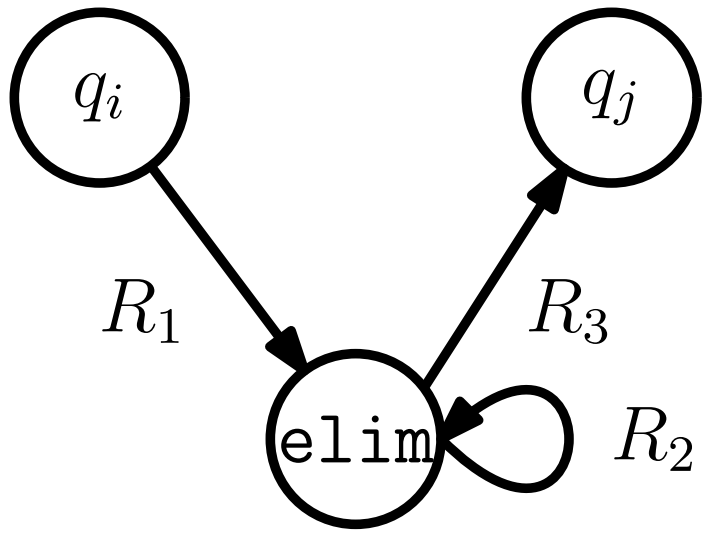
- Eliminar estado `elim`

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`

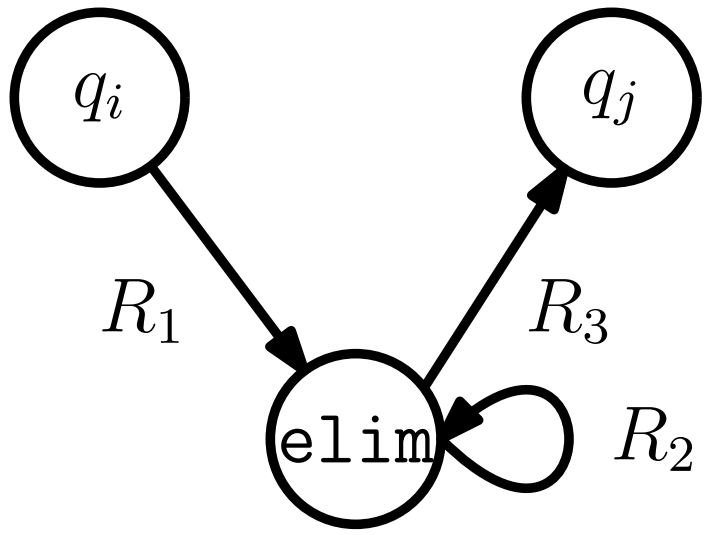
Eliminación de estados

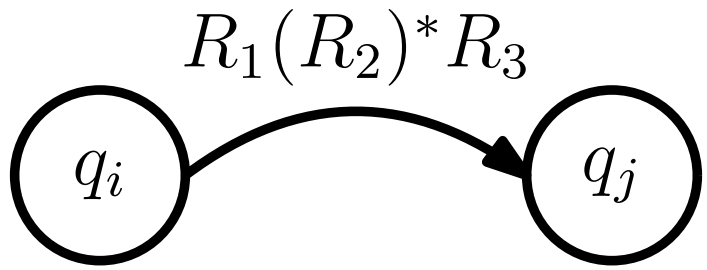
- Eliminar estado `elim`
- Actualizar q_i y q_j si hay conexión entre q_i a `elim` y de q_j a `elim`
  - No hay conexión de q_i a q_j



Eliminación de estados

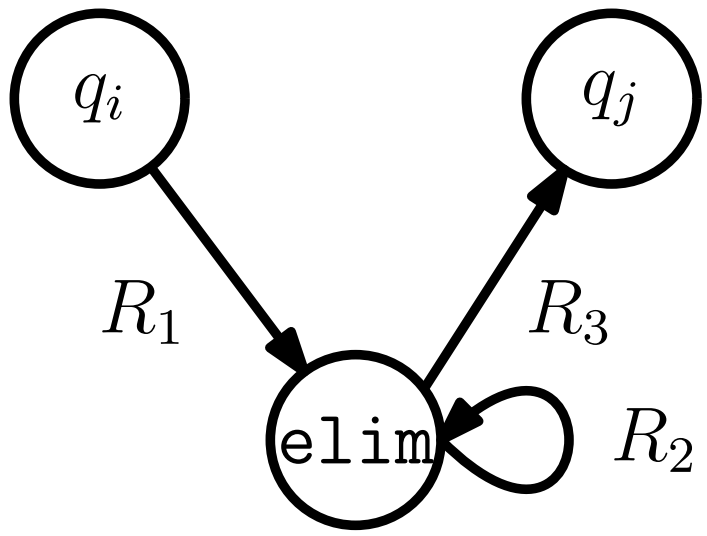
- Eliminar estado `elim`
- Actualizar q_i y q_j si hay conexión entre q_i a `elim` y de q_j a `elim`
  - No hay conexión de q_i a q_j

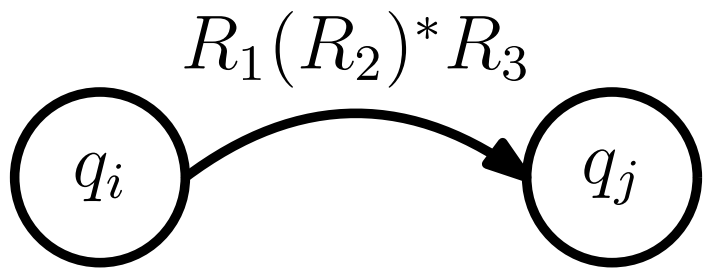




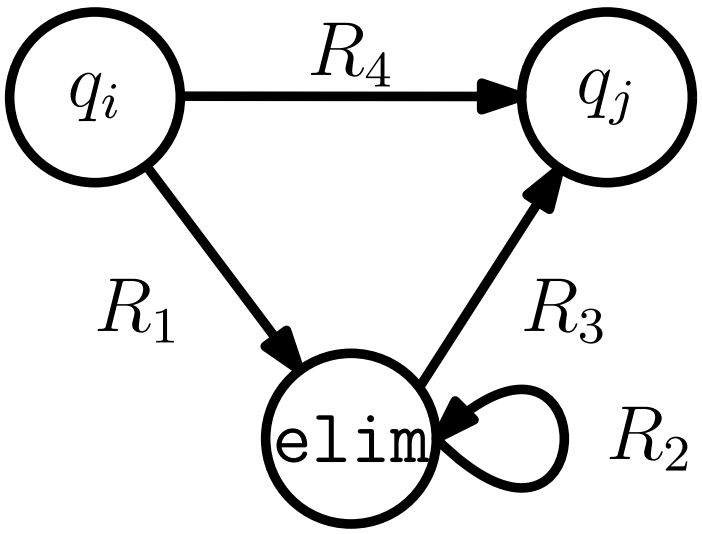
Eliminación de estados

- 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



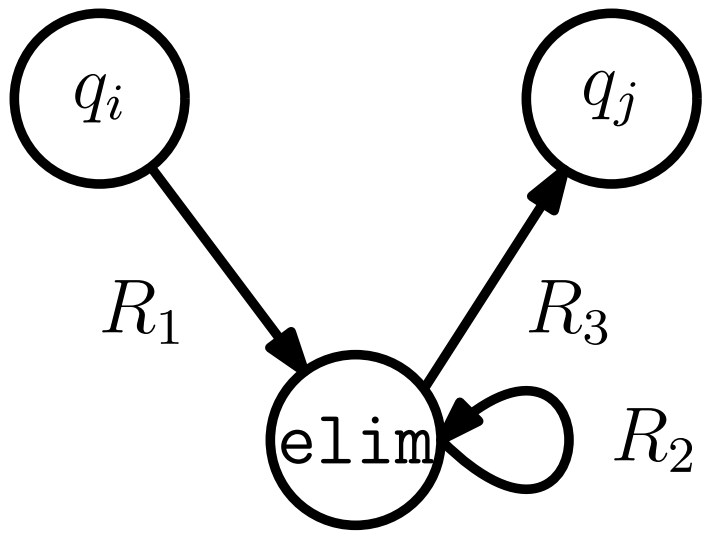


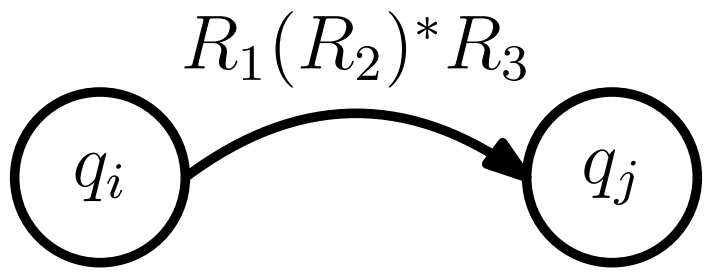
  - Hay conexión de q_i a q_j



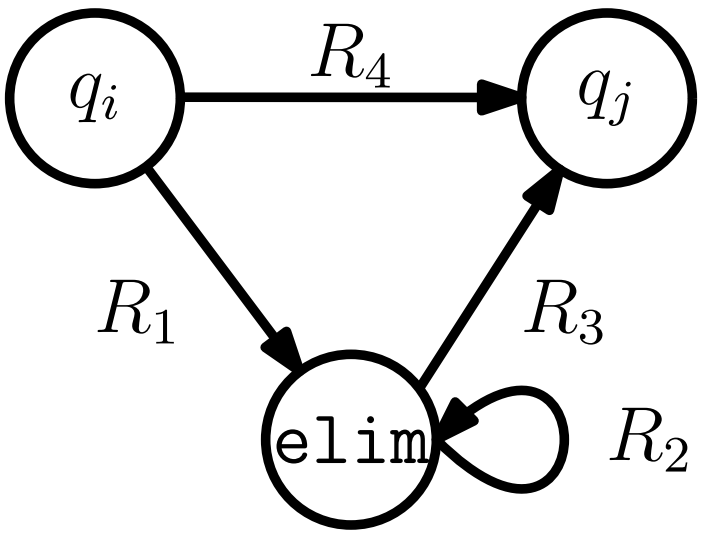
Eliminación de estados

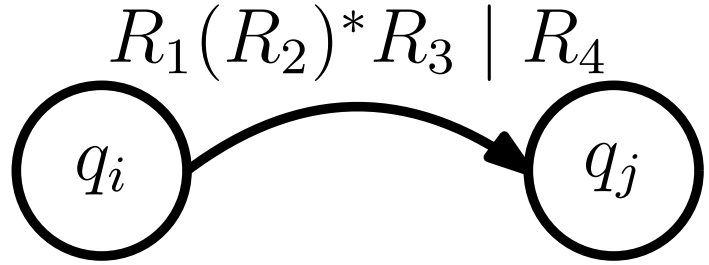
- Eliminar estado `elim`
- Actualizar q_i y q_j si hay conexión entre q_i a `elim` y de q_j a `elim`
  - No hay conexión de q_i a q_j





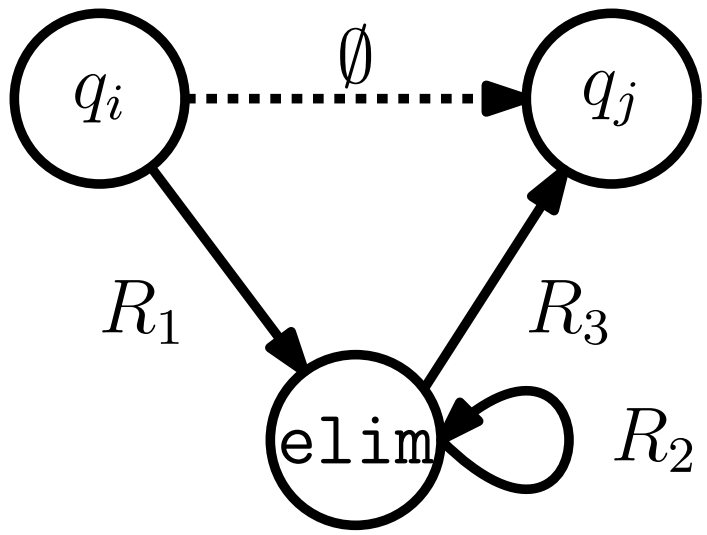
  - Hay conexión de q_i a q_j

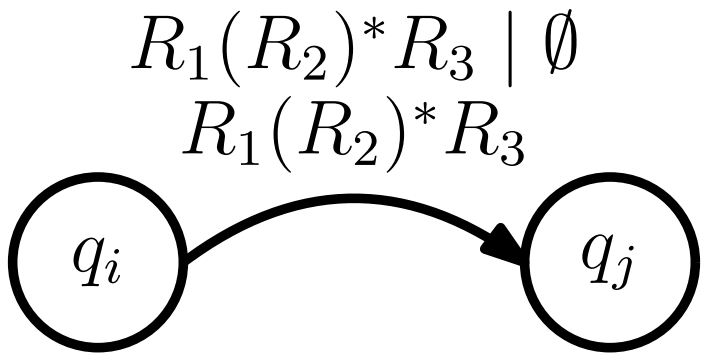




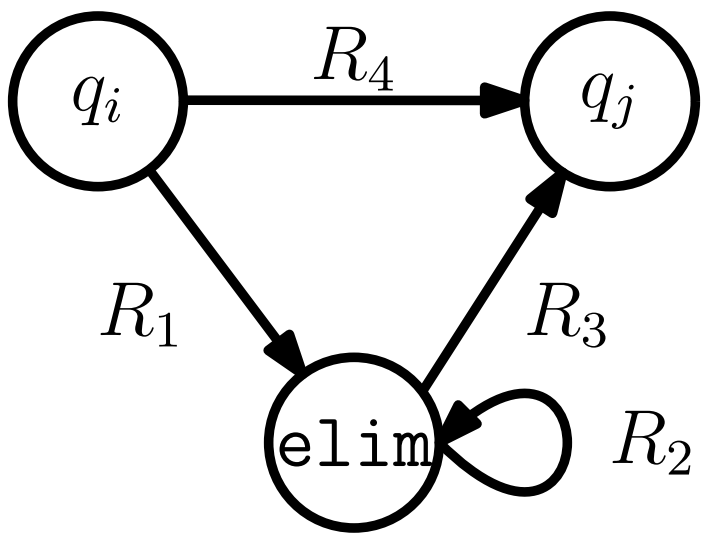
Eliminación de estados

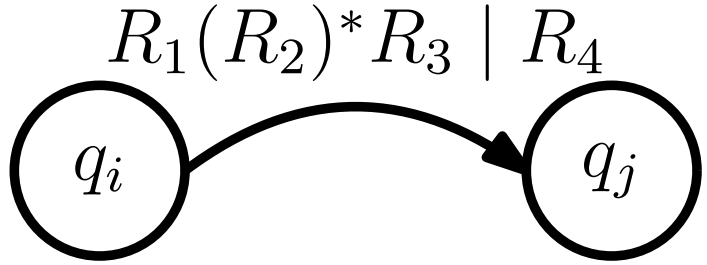
- Eliminar estado `elim`
- Actualizar q_i y q_j si hay conexión entre q_i a `elim` y de q_j a `elim`
  - No hay conexión de q_i a q_j





  - Hay conexión de q_i a q_j

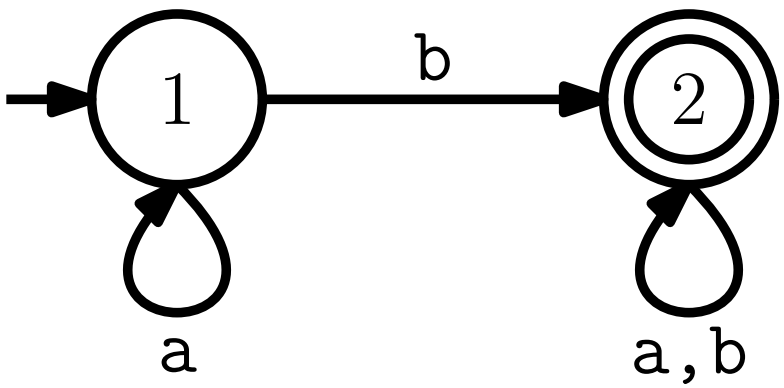




- Mismo caso si uno interpreta no hay conexión como $\emptyset$.

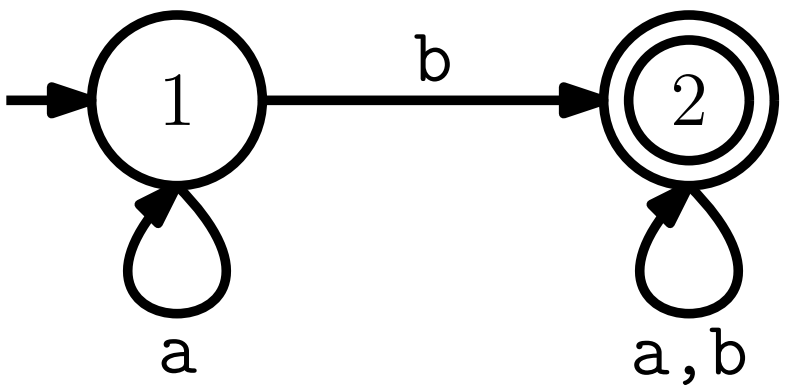
Ejemplos

- Convertir a ER

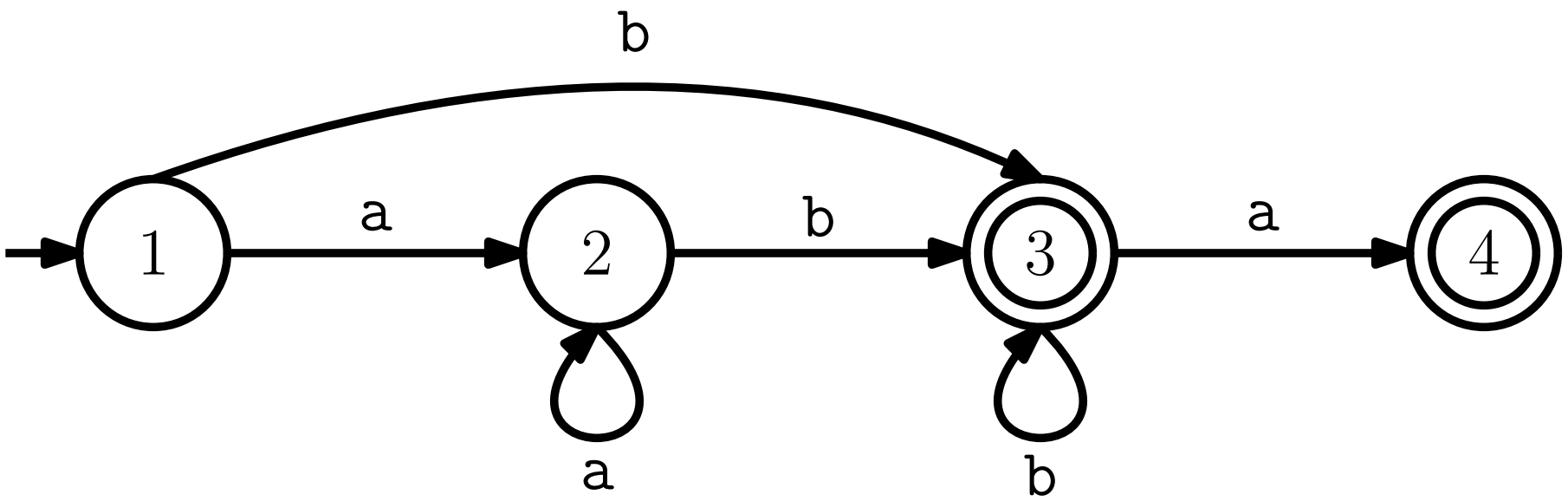


Ejemplos

- Convertir a ER



- Convertir a ER



Recapitulación sobre lenguajes regulares

Tres perspectivas

- **Lenguaje regular:** aceptado por AFD, AFND, ER.

Tres perspectivas

- **Lenguaje regular:** aceptado por AFD, AFND, ER.
- AFND $\rightarrow$ AFD.

Tres perspectivas

- **Lenguaje regular:** aceptado por AFD, AFND, ER.
- AFND $\rightarrow$ AFD.
  - Quitar las transiciones ε vía alcanzables.

Tres perspectivas

- **Lenguaje regular:** aceptado por AFD, AFND, ER.
- AFND $\rightarrow$ AFD.
  - Quitar las transiciones ε vía alcanzables.
  - Construcción en 2^Q simulando todos los posibles estados.

Tres perspectivas

- **Lenguaje regular:** aceptado por AFD, AFND, ER.
- AFND $\rightarrow$ AFD.
  - Quitar las transiciones ε vía alcanzables.
  - Construcción en 2^Q simulando todos los posibles estados.
- ER $\rightarrow$ AFND.

Tres perspectivas

- **Lenguaje regular:** aceptado por AFD, AFND, ER.
- AFND $\rightarrow$ AFD.
  - Quitar las transiciones ε vía alcanzables.
  - Construcción en 2^Q simulando todos los posibles estados.
- ER $\rightarrow$ AFND.
  - Usar la definición recursiva de las ER.

Tres perspectivas

- **Lenguaje regular:** aceptado por AFD, AFND, ER.
- AFND $\rightarrow$ AFD.
  - Quitar las transiciones ε vía alcanzables.
  - Construcción en 2^Q simulando todos los posibles estados.
- ER $\rightarrow$ AFND.
  - Usar la definición recursiva de las ER.
- AFD, AFND $\rightarrow$ ER

Tres perspectivas

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

Tres perspectivas

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

Tres perspectivas

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

Tres perspectivas

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

Tres perspectivas

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

Mostrando que lenguajes son regulares

- Diseñar AFD, AFND o ER.

Mostrando que lenguajes son regulares

- Diseñar AFD, AFND o ER.
- $+$ propiedades de cerradura

Mostrando que lenguajes son regulares

- Diseñar AFD, AFND o ER.
- $+$ propiedades de cerradura
- Complemento: **no**.

Mostrando que lenguajes son regulares

- Diseñar AFD, AFND o ER.
- $+$ propiedades de cerradura
- Complemento: **no**.
  - **Dem:** cambiar aceptación/no aceptación en un 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**.

Mostrando que lenguajes son regulares

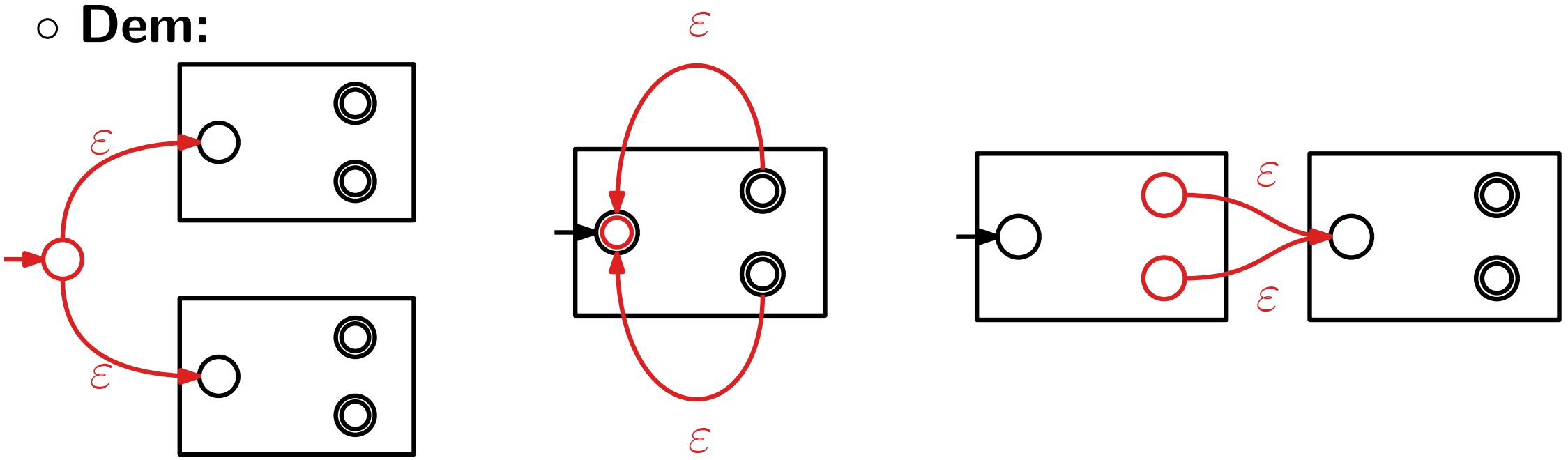
- Diseñar AFD, AFND o ER.
- $+$ propiedades de cerradura
- Complemento: **no**.
  - **Dem:** cambiar aceptación/no aceptación en un AFD.
- Intersección y unión: **y/o**.
  - **Dem:** correr los AFD en paralelo vía el producto.

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.

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:**

Wrap-Up

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