

AUTÓMATAS FINITOS

Lenguajes formales, autómatas y computabilidad

Departamento de Computación, FCEN, UBA

1er cuatrimestre 2025

Autómatas finitos

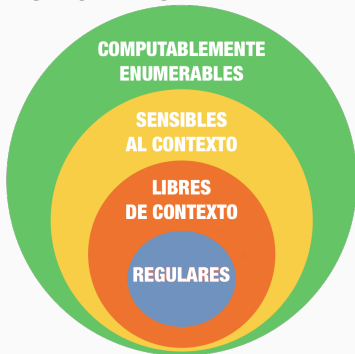
- La clase pasada vimos alfabetos, cadenas y lenguajes

Autómatas finitos

- La clase pasada vimos alfabetos, cadenas y lenguajes
- Hoy vamos a ver una máquina abstracta que nos permite *reconocer* lenguajes: los **autómatas finitos**

Autómatas finitos

- La clase pasada vimos alfabetos, cadenas y lenguajes
- Hoy vamos a ver una máquina abstracta que nos permite *reconocer* lenguajes: los **autómatas finitos**
- Estos reconocen exactamente una *clase* de lenguajes en particular: los **lenguajes regulares**.



Jerarquía de Chomsky-Schutzenberger

Ejercicio 1 - Repaso y definiciones

Ejercicio 1b de la práctica 2

Dar un autómata finito que reconozca \mathcal{L}_1 :

$$\mathcal{L}_1 = \{ \omega \mid \omega \in \{0, 1\}^* \text{ y } |\omega|_0 \text{ es par} \}$$

Ejercicio 1 - Repaso y definiciones

Ejercicio 1b de la práctica 2

Dar un autómata finito que reconozca \mathcal{L}_1 :

$$\mathcal{L}_1 = \{ \omega \mid \omega \in \{0, 1\}^* \text{ y } |\omega|_0 \text{ es par} \}$$

Cadenas sobre $\Sigma = \{0, 1\}$ con cantidad par de ceros.

Ejemplos de cadenas:

Ejercicio 1 - Repaso y definiciones

Ejercicio 1b de la práctica 2

Dar un autómata finito que reconozca \mathcal{L}_1 :

$$\mathcal{L}_1 = \{ \omega \mid \omega \in \{0, 1\}^* \text{ y } |\omega|_0 \text{ es par} \}$$

Cadenas sobre $\Sigma = \{0, 1\}$ con cantidad par de ceros.

Ejemplos de cadenas:

- Que pertenecen: 1, 00, 010, 00100010, λ ✓
- Que **no** pertenecen: 0, 10010, 1110, 101 ✗

Ejercicio 1 - Repaso y definiciones

Ejercicio 1b de la práctica 2

Dar un autómata finito que reconozca \mathcal{L}_1 :

$$\mathcal{L}_1 = \{ \omega \mid \omega \in \{0, 1\}^* \text{ y } |\omega|_0 \text{ es par} \}$$

Cadenas sobre $\Sigma = \{0, 1\}$ con cantidad par de ceros.

Ejemplos de cadenas:

- Que pertenecen: 1, 00, 010, 00100010, λ ✓
- Que **no** pertenecen: 0, 10010, 1110, 101 ✗

¿ \mathcal{L}_1 es regular?

Ejercicio 1 - Repaso y definiciones

Ejercicio 1b de la práctica 2

Dar un autómata finito que reconozca \mathcal{L}_1 :

$$\mathcal{L}_1 = \{ \omega \mid \omega \in \{0, 1\}^* \text{ y } |\omega|_0 \text{ es par} \}$$

Cadenas sobre $\Sigma = \{0, 1\}$ con cantidad par de ceros.

Ejemplos de cadenas:

- Que pertenecen: 1, 00, 010, 00100010, λ ✓
- Que **no** pertenecen: 0, 10010, 1110, 101 ✗

¿ \mathcal{L}_1 es regular? Sí, el autómata \mathcal{A}_1 que reconoce a \mathcal{L}_1 es:

Ejercicio 1 - Repaso y definiciones

Ejercicio 1b de la práctica 2

Dar un autómata finito que reconozca \mathcal{L}_1 :

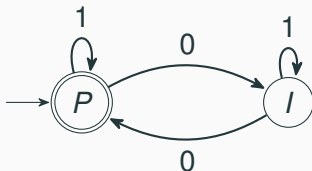
$$\mathcal{L}_1 = \{ \omega \mid \omega \in \{0, 1\}^* \text{ y } |\omega|_0 \text{ es par} \}$$

Cadenas sobre $\Sigma = \{0, 1\}$ con cantidad par de ceros.

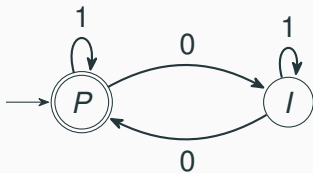
Ejemplos de cadenas:

- Que pertenecen: 1, 00, 010, 00100010, λ ✓
- Que **no** pertenecen: 0, 10010, 1110, 101 ✗

¿ \mathcal{L}_1 es regular? Sí, el autómata \mathcal{A}_1 que reconoce a \mathcal{L}_1 es:

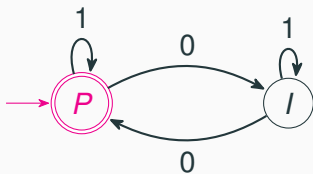


Casos de test



- $010 \in \mathcal{L}_1$:

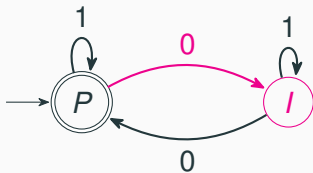
Casos de test



- $010 \in \mathcal{L}_1$:

P

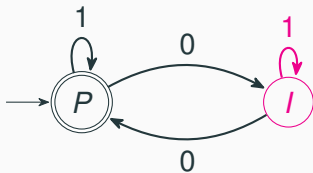
Casos de test



- $010 \in \mathcal{L}_1$:

$P \xrightarrow{0} I$

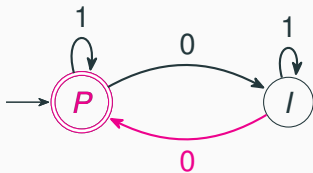
Casos de test



- $010 \in \mathcal{L}_1$:

$$P \xrightarrow{0} I \xrightarrow{1} I$$

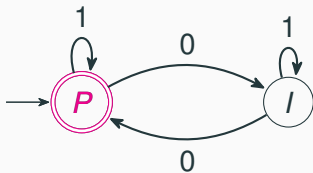
Casos de test



- $010 \in \mathcal{L}_1$:

$$P \xrightarrow{0} I \xrightarrow{1} I \xrightarrow{0} P$$

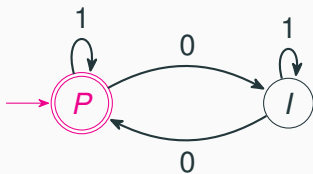
Casos de test



- $010 \in \mathcal{L}_1$:

$P \xrightarrow{0} I \xrightarrow{1} I \xrightarrow{0} P \checkmark$

Casos de test



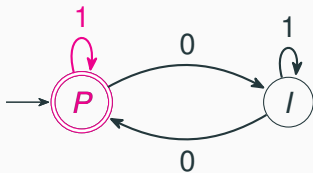
- $010 \in \mathcal{L}_1$:

$$P \xrightarrow{0} I \xrightarrow{1} I \xrightarrow{0} P \checkmark$$

- $101 \notin \mathcal{L}_1$:

P

Casos de test



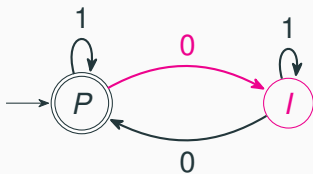
- $010 \in \mathcal{L}_1$:

$$P \xrightarrow{0} I \xrightarrow{1} I \xrightarrow{0} P \checkmark$$

- $101 \notin \mathcal{L}_1$:

$$P \xrightarrow{1} P$$

Casos de test



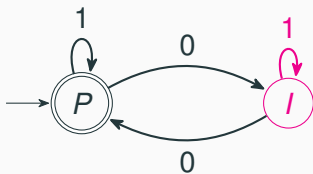
- $010 \in \mathcal{L}_1$:

$$P \xrightarrow{0} I \xrightarrow{1} I \xrightarrow{0} P \checkmark$$

- $101 \notin \mathcal{L}_1$:

$$P \xrightarrow{1} P \xrightarrow{0} I$$

Casos de test



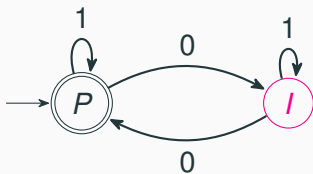
- $010 \in \mathcal{L}_1$:

$$P \xrightarrow{0} I \xrightarrow{1} I \xrightarrow{0} P \checkmark$$

- $101 \notin \mathcal{L}_1$:

$$P \xrightarrow{1} P \xrightarrow{0} I \xrightarrow{1} I$$

Casos de test



- $010 \in \mathcal{L}_1$:

$$P \xrightarrow{0} I \xrightarrow{1} I \xrightarrow{0} P \checkmark$$

- $101 \notin \mathcal{L}_1$:

$$P \xrightarrow{1} P \xrightarrow{0} I \xrightarrow{1} I \text{ ✗}$$

Autómata finito

Un AFD es una tupla de la forma

$$\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle,$$

donde:

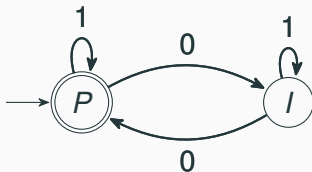
- Q es un conjunto de estados
- Σ es el alfabeto
- $\delta : Q \times \Sigma \rightarrow Q$ es la función de transición
- q_0 es el estado inicial
- $F \subseteq Q$ es el conjunto de estados finales

Volviendo al ejercicio

La tupla que describe a \mathcal{A}_1 es

$$\mathcal{A}_1 = \langle \{P, I\}, \{0, 1\}, \delta, \overset{\text{inicial}}{P}, \{P\} \rangle$$

y antes dimos una representación pictórica de $\delta : Q \times \Sigma \rightarrow Q$

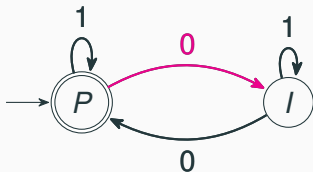


Volviendo al ejercicio

La tupla que describe a \mathcal{A}_1 es

$$\mathcal{A}_1 = \langle \{P, I\}, \{0, 1\}, \delta, \overset{\text{inicial}}{P}, \overset{F}{\{P\}} \rangle$$

y antes dimos una representación pictórica de $\delta : Q \times \Sigma \rightarrow Q$



También la podríamos definir con la siguiente tabla:

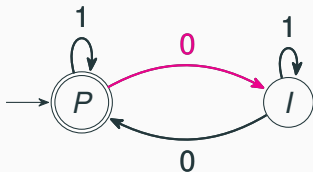
δ	0	1
P		
I		

Volviendo al ejercicio

La tupla que describe a \mathcal{A}_1 es

$$\mathcal{A}_1 = \langle \{P, I\}, \{0, 1\}, \delta, \overset{\text{inicial}}{P}, \overset{F}{\{P\}} \rangle$$

y antes dimos una representación pictórica de $\delta : Q \times \Sigma \rightarrow Q$



También la podríamos definir con la siguiente tabla:

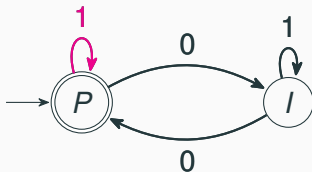
δ	0	1
P	I	
I		

Volviendo al ejercicio

La tupla que describe a \mathcal{A}_1 es

$$\mathcal{A}_1 = \langle \{P, I\}, \{0, 1\}, \delta, \overset{\text{inicial}}{P}, \overset{F}{\{P\}} \rangle$$

y antes dimos una representación pictórica de $\delta : Q \times \Sigma \rightarrow Q$



También la podríamos definir con la siguiente tabla:

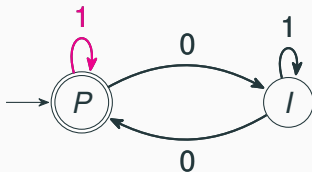
δ	0	1
P	I	
I		

Volviendo al ejercicio

La tupla que describe a \mathcal{A}_1 es

$$\mathcal{A}_1 = \langle \{P, I\}, \{0, 1\}, \delta, \overset{\text{inicial}}{P}, \overset{F}{\{P\}} \rangle$$

y antes dimos una representación pictórica de $\delta : Q \times \Sigma \rightarrow Q$



También la podríamos definir con la siguiente tabla:

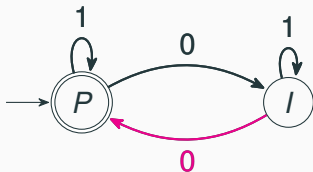
δ	0	1
P	I	P
I		

Volviendo al ejercicio

La tupla que describe a \mathcal{A}_1 es

$$\mathcal{A}_1 = \langle \{\overset{Q}{P}, I\}, \{\overset{\Sigma}{0}, 1\}, \overset{\text{inicial}}{P}, \overset{F}{P}, \{\overset{F}{P}\} \rangle$$

y antes dimos una representación pictórica de $\delta : Q \times \Sigma \rightarrow Q$



También la podríamos definir con la siguiente tabla:

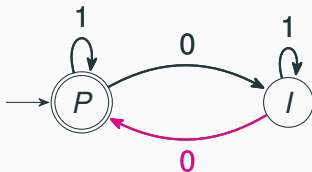
δ	0	1
P	I	P
I		

Volviendo al ejercicio

La tupla que describe a \mathcal{A}_1 es

$$\mathcal{A}_1 = \langle \{P, I\}, \{0, 1\}, \delta, \overset{\text{inicial}}{P}, \overset{F}{P} \rangle$$

y antes dimos una representación pictórica de $\delta : Q \times \Sigma \rightarrow Q$



También la podríamos definir con la siguiente tabla:

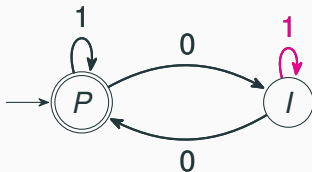
δ	0	1
P	I	P
I	P	

Volviendo al ejercicio

La tupla que describe a \mathcal{A}_1 es

$$\mathcal{A}_1 = \langle \{P, I\}, \{0, 1\}, \delta, \overset{\text{inicial}}{P}, \overset{F}{\{P\}} \rangle$$

y antes dimos una representación pictórica de $\delta : Q \times \Sigma \rightarrow Q$



También la podríamos definir con la siguiente tabla:

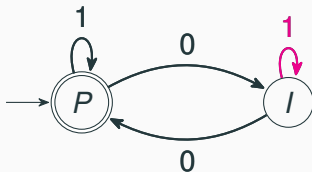
δ	0	1
P	I	P
I	P	

Volviendo al ejercicio

La tupla que describe a \mathcal{A}_1 es

$$\mathcal{A}_1 = \langle \{P, I\}, \{0, 1\}, \delta, \overset{\text{inicial}}{P}, \overset{F}{\{P\}} \rangle$$

y antes dimos una representación pictórica de $\delta : Q \times \Sigma \rightarrow Q$



También la podríamos definir con la siguiente tabla:

δ	0	1
P	I	P
I	P	I

La tupla y los parciales

Para los ejercicios alcanza con el dibujo para especificar δ , no es necesario que escriban la tabla. Pero, en especial en los **parciales**,

La tupla y los parciales

Para los ejercicios alcanza con el dibujo para especificar δ , no es necesario que escriban la tabla. Pero, en especial en los **parciales**,

¡No olviden la tupla!

La tupla y los parciales

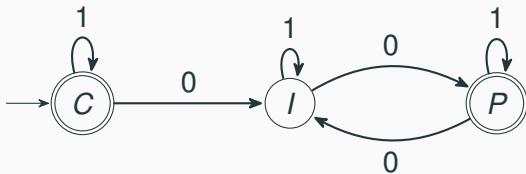
Para los ejercicios alcanza con el dibujo para especificar δ , no es necesario que escriban la tabla. Pero, en especial en los **parciales**,

¡No olviden la tupla!

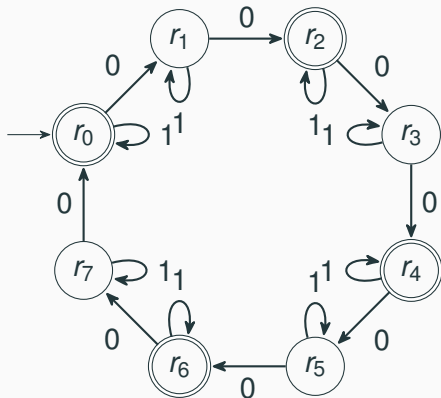
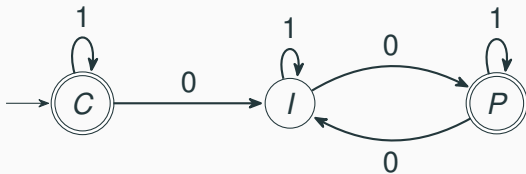


¿Hay otros autómatas?

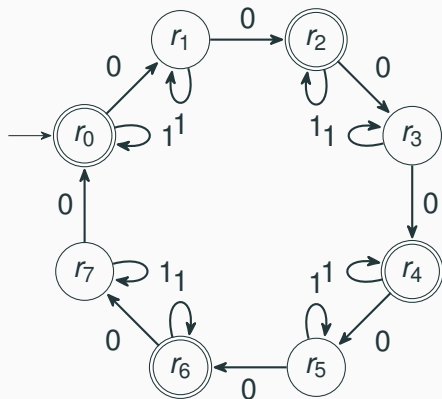
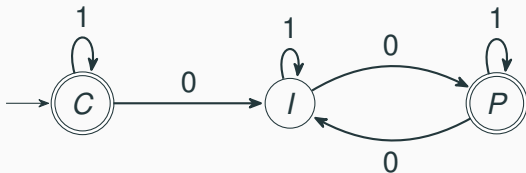
¿Hay otros autómatas?



¿Hay otros autómatas?



¿Hay otros autómatas?



Configuraciones instantáneas

¿Cómo formalizamos que un autómata **acepte** una cadena?

- Vamos a definir *configuraciones instantáneas*, una tupla compuesta por el estado actual y lo que resta de consumir de la cadena. Representan una *foto* del proceso de reconocimiento de una cadena en un instante dado.

Configuraciones instantáneas

¿Cómo formalizamos que un autómata **acepte** una cadena?

- Vamos a definir *configuraciones instantáneas*, una tupla compuesta por el estado actual y lo que resta de consumir de la cadena. Representan una *foto* del proceso de reconocimiento de una cadena en un instante dado.
- Luego, el autómata va a *transicionar* entre configuraciones a medida que consume la cadena.

Configuraciones instantáneas

¿Cómo formalizamos que un autómata **acepte** una cadena?

- Vamos a definir *configuraciones instantáneas*, una tupla compuesta por el estado actual y lo que resta de consumir de la cadena. Representan una *foto* del proceso de reconocimiento de una cadena en un instante dado.
- Luego, el autómata va a *transicionar* entre configuraciones a medida que consume la cadena.

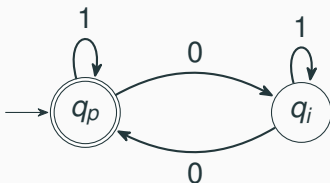
Ejemplo: $\alpha = 010$

Configuraciones instantáneas

¿Cómo formalizamos que un autómata **acepte** una cadena?

- Vamos a definir *configuraciones instantáneas*, una tupla compuesta por el estado actual y lo que resta de consumir de la cadena. Representan una *foto* del proceso de reconocimiento de una cadena en un instante dado.
- Luego, el autómata va a *transicionar* entre configuraciones a medida que consume la cadena.

Ejemplo: $\alpha = 010$



$$\begin{aligned}(q_p, 010) &\vdash_{\mathcal{A}_1} (q_i, 10) \\ &\vdash_{\mathcal{A}_1} (q_i, 0) \\ &\vdash_{\mathcal{A}_1} (q_p, \lambda)\end{aligned}$$

Formalizando

Dado $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$, definimos:

Configuraciones instantáneas

$$(q, \alpha) \in Q \times \Sigma^*$$

donde q es el estado actual y α es lo que resta por consumir de la cadena de entrada.

¿Cómo podemos formalizar la transición entre ellas?

Formalizando

Dado $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$, definimos:

Configuraciones instantáneas

$$(q, \alpha) \in Q \times \Sigma^*$$

donde q es el estado actual y α es lo que resta por consumir de la cadena de entrada.

¿Cómo podemos formalizar la transición entre ellas? Podemos pasar de un estado a otro consumiendo un símbolo solo si δ nos dice que existe tal transición

Relación de transición entre configuraciones

$$(q_i, a.\alpha) \vdash_{\mathcal{A}} (q_j, \alpha) \iff$$

Formalizando

Dado $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$, definimos:

Configuraciones instantáneas

$$(q, \alpha) \in Q \times \Sigma^*$$

donde q es el estado actual y α es lo que resta por consumir de la cadena de entrada.

¿Cómo podemos formalizar la transición entre ellas? Podemos pasar de un estado a otro consumiendo un símbolo solo si δ nos dice que existe tal transición

Relación de transición entre configuraciones

$$(q_i, a.\alpha) \vdash_{\mathcal{A}} (q_j, \alpha) \iff \delta(q_i, a) = q_j$$

Pertenencia al lenguaje

Con todo lo que vimos, ¿Se les ocurre cómo definir el **lenguaje aceptado** por un autómatata?

Pertenencia al lenguaje

Con todo lo que vimos, ¿Se les ocurre cómo definir el **lenguaje aceptado** por un autómata?

Lenguaje aceptado

$$\alpha \in L(\mathcal{A}) \iff \exists q_f \in F \mid (q_0, \alpha) \vdash_{\mathcal{A}}^* (q_f, \lambda)$$

α pertenece al lenguaje aceptado si partiendo de la configuración inicial (q_0, α) se puede consumir toda la cadena llegando a un estado final. Es decir, llegar a la configuración (q_f, λ) con $q_f \in F$.

Recordatorio

Recordemos que \vdash^* quiere decir aplicar \vdash cero o más veces.
¿Cuándo necesitamos aplicarla cero veces?

Pertenencia al lenguaje

Con todo lo que vimos, ¿Se les ocurre cómo definir el **lenguaje aceptado** por un autómata?

Lenguaje aceptado

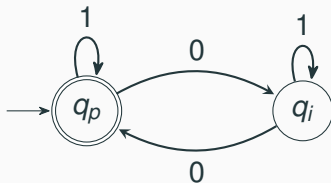
$$\alpha \in L(\mathcal{A}) \iff \exists q_f \in F \mid (q_0, \alpha) \vdash_{\mathcal{A}}^* (q_f, \lambda)$$

α pertenece al lenguaje aceptado si partiendo de la configuración inicial (q_0, α) se puede consumir toda la cadena llegando a un estado final. Es decir, llegar a la configuración (q_f, λ) con $q_f \in F$.

Recordatorio

Recordemos que \vdash^* quiere decir aplicar \vdash cero o más veces. ¿Cuándo necesitamos aplicarla cero veces? Cuando la entrada es λ .

Seguimientos de cadenas



- $(q_p, 010) \vdash_{A_1} (q_i, 10) \vdash_{A_1} (q_i, 0) \vdash_{A_1} (q_p, \lambda)$ ✓
- $(q_p, 101) \vdash_{A_1} (q_p, 01) \vdash_{A_1} (q_i, 1) \vdash_{A_1} (q_i, \lambda)$ ✗

Ejercicio 2

Enunciado

Cadenas sobre $\Sigma = \{0, 1\}$ que comienzan con 01.

$$\mathcal{L}_2 = \{01\alpha \mid \alpha \in \{0, 1\}^*\}$$

¿Qué tenemos que recordar?

Ejercicio 2

Enunciado

Cadenas sobre $\Sigma = \{0, 1\}$ que comienzan con 01.

$$\mathcal{L}_2 = \{01\alpha \mid \alpha \in \{0, 1\}^*\}$$

¿Qué tenemos que recordar? Si vimos un 0 y luego un 1

Ejercicio 2

Enunciado

Cadenas sobre $\Sigma = \{0, 1\}$ que comienzan con 01.

$$\mathcal{L}_2 = \{01\alpha \mid \alpha \in \{0, 1\}^*\}$$

¿Qué tenemos que recordar? Si vimos un 0 y luego un 1
Proponemos el siguiente autómata \mathcal{A}_2 para \mathcal{L}_2 ,



¿Qué problema tiene?

Ejercicio 2

Enunciado

Cadenas sobre $\Sigma = \{0, 1\}$ que comienzan con 01.

$$\mathcal{L}_2 = \{01\alpha \mid \alpha \in \{0, 1\}^*\}$$

¿Qué tenemos que recordar? Si vimos un 0 y luego un 1
Proponemos el siguiente autómata \mathcal{A}_2 para \mathcal{L}_2 ,



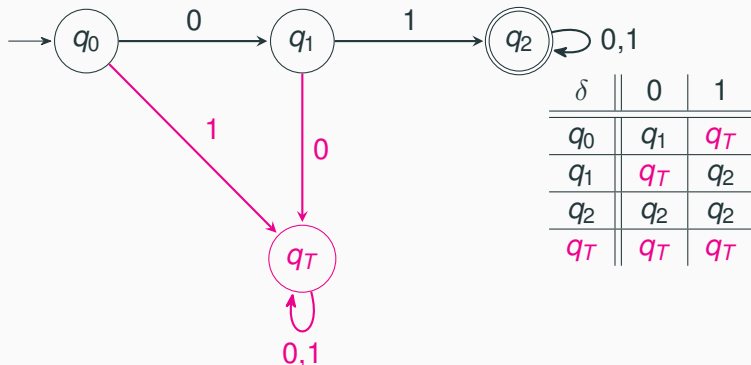
¿Qué problema tiene?

¡La función de transición está incompleta!

δ	0	1
q_0	q_1	?
q_1	?	q_2
q_2	q_2	q_2

Estado trampa

Para que el autómata quede bien definido, vamos a completarlo con un **estado trampa** (o de *error*), al que van todas las transiciones no definidas y cicla sobre sí mismo con todos los símbolos del alfabeto



$$\mathcal{A}_2 = \langle \{q_0, q_1, q_2, q_T\}, \{0, 1\}, \delta, q_0, \{q_2\} \rangle$$

Ejercicio 3

Ejercicio 3

Cadenas sobre $\Sigma = \{0, 1\}$ que terminan con 01.

$$\mathcal{L}_3 = \{\alpha 01 \mid \alpha \in \{0, 1\}^*\}$$

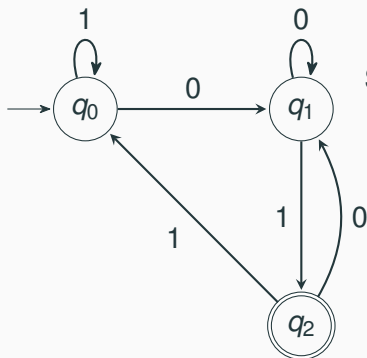
Ejercicio 3

Ejercicio 3

Cadenas sobre $\Sigma = \{0, 1\}$ que terminan con 01.

$$\mathcal{L}_3 = \{\alpha 01 \mid \alpha \in \{0, 1\}^*\}$$

\mathcal{A}_3 :



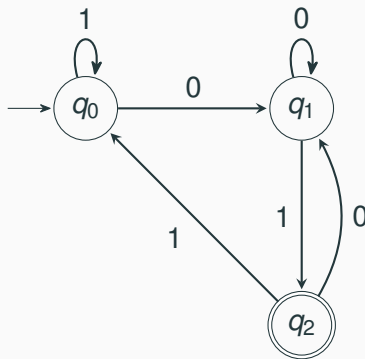
Significado intuitivo de cada estado:

- q_0 : “La cadena no termina en 0 ni en 01”
- q_1 : “La cadena termina en 0”
- q_2 : “La cadena termina en 01”

Casos de test

$$\mathcal{L}_3 = \{\alpha 01 \mid \alpha \in \{0, 1\}^*\}$$

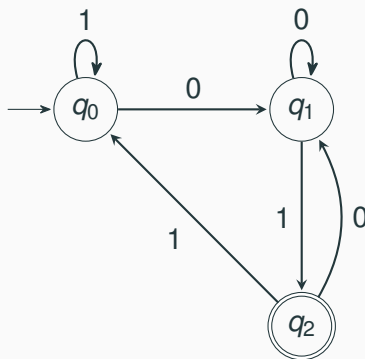
\mathcal{A}_3 :



Casos de test

$$\mathcal{L}_3 = \{\alpha 01 \mid \alpha \in \{0, 1\}^*\}$$

\mathcal{A}_3 :



- 0, 1, 010, 011, 1111010 ✗
- 01, 0001, 010101, 1100001 ✓

El lenguaje \mathcal{L}_3 es muy parecido a \mathcal{L}_2 , pero el autómata se ve más complicado. Nos gustaría que el formalismo nos permita expresar algo como “Puede venir cualquier cadena, siempre y cuando termine con 01”, al igual que para \mathcal{L}_2 nos permitía decir “Si arranca por 01 puede seguir cualquier cadena”.

Alternativa

El lenguaje \mathcal{L}_3 es muy parecido a \mathcal{L}_2 , pero el autómata se ve más complicado. Nos gustaría que el formalismo nos permita expresar algo como “Puede venir cualquier cadena, siempre y cuando termine con 01”, al igual que para \mathcal{L}_2 nos permitía decir “Si arranca por 01 puede seguir cualquier cadena”.

Proponemos \mathcal{A}'_3 ,

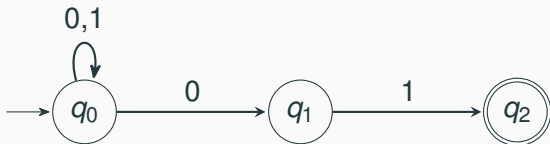


Pero

Alternativa

El lenguaje \mathcal{L}_3 es muy parecido a \mathcal{L}_2 , pero el autómata se ve más complicado. Nos gustaría que el formalismo nos permita expresar algo como “Puede venir cualquier cadena, siempre y cuando termine con 01”, al igual que para \mathcal{L}_2 nos permitía decir “Si arranca por 01 puede seguir cualquier cadena”.

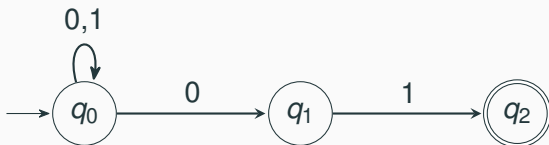
Proponemos \mathcal{A}'_3 ,



Pero no cuadra con la definición que vimos antes, $\delta(q_0, 0)$ tiene más de una opción: q_0 y q_1 . ¡No es **determinístico**!

Seguimientos

A'_3 :



Para la cadena $\alpha = 101 \in \mathcal{L}_3$ tenemos dos caminos posibles

Seguimientos

A'_3 :



Para la cadena $\alpha = 101 \in \mathcal{L}_3$ tenemos dos caminos posibles

- $q_0 \xrightarrow[{\mathcal{A}'_3}]{1} q_0 \xrightarrow[{\mathcal{A}'_3}]{0} q_1 \xrightarrow[{\mathcal{A}'_3}]{1} q_2$ ✓
- $q_0 \xrightarrow[{\mathcal{A}'_3}]{1} q_0 \xrightarrow[{\mathcal{A}'_3}]{0} q_0 \xrightarrow[{\mathcal{A}'_3}]{1} q_0$ ✗

Alcanza con que *exista al menos un* recorrido desde el inicial a un estado final cuya etiqueta sea la cadena para que pertenezca al lenguaje. **No importa que haya otros que no sean exitosos.**

Los autómatas que veníamos viendo hasta ahora eran **determinísticos**, en cada momento tenían una sola acción posible. \mathcal{A}'_3 es un autómata finito **no determinístico**, que en cada paso puede tener más de una alternativa para elegir.

Los autómatas que veníamos viendo hasta ahora eran **determinísticos**, en cada momento tenían una sola acción posible. \mathcal{A}'_3 es un autómata finito **no determinístico**, que en cada paso puede tener más de una alternativa para elegir. Al igual que los AFDs, son una tupla $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$ pero cambia la función de transición:

$$\delta : Q \times (\Sigma \cup \lambda) \rightarrow \mathcal{P}(Q)$$

- En lugar de un solo estado, devuelve un conjunto
- Además de transiciones por un símbolo de la entrada, podemos transicionar por λ ¹ sin consumir ningún símbolo

¹No confundir con la cadena λ , es una notación

Los autómatas que veníamos viendo hasta ahora eran **determinísticos**, en cada momento tenían una sola acción posible. \mathcal{A}'_3 es un autómata finito **no determinístico**, que en cada paso puede tener más de una alternativa para elegir. Al igual que los AFDs, son una tupla $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$ pero cambia la función de transición:

$$\delta : Q \times (\Sigma \cup \lambda) \rightarrow \mathcal{P}(Q)$$

- En lugar de un solo estado, devuelve un conjunto
- Además de transiciones por un símbolo de la entrada, podemos transicionar por λ ¹ sin consumir ningún símbolo

Diferencia con bibliografía

Se suele diferenciar entre AFNDs con y sin transiciones λ (a veces llamadas ϵ), pero para nosotros va a ser lo mismo.

¹No confundir con la cadena λ , es una notación

Volviendo al ejercicio



La tupla que describe a \mathcal{A}'_3 es

$$\mathcal{A}'_3 = \langle \{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_2\} \rangle$$

donde δ está dada por la siguiente tabla,

δ	0	1	λ
q_0	$\{q_0, q_1\}$	$\{q_0\}$	\emptyset
q_1	\emptyset	$\{q_2\}$	\emptyset
q_2	\emptyset	\emptyset	\emptyset

AFNDs no hacen trampa

δ	0	1	λ
q_0	$\{q_0, q_1\}$	$\{q_0\}$	\emptyset
q_1	\emptyset	$\{q_2\}$	\emptyset
q_2	\emptyset	\emptyset	\emptyset

Trampas en AFNDs

Observen que **en AFNDs no es necesario completar con un estado trampa**. La imagen de δ son los conjuntos y asumimos que si una transición no está en el dibujo, va a \emptyset

Relación \vdash y lenguaje

Las configuraciones instantáneas son las mismas que para AFDs, ¿pero cómo es la relación entre ellas?

Relación \vdash y lenguaje

Las configuraciones instantáneas son las mismas que para AFDs, ¿pero cómo es la relación entre ellas? Hay dos casos: consumiendo un símbolo o por λ

Relación de transición entre configuraciones

$$(q_i, a.\alpha) \vdash_{\mathcal{A}} (q_j, \alpha) \iff q_j \in \delta(q_i, a)$$

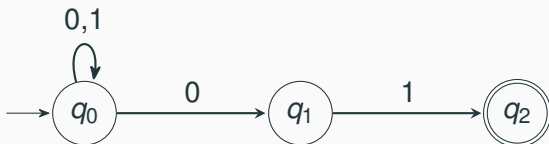
$$(q_i, \alpha) \vdash_{\mathcal{A}} (q_j, \alpha) \iff q_j \in \delta(q_i, \lambda)$$

La definición del lenguaje es idéntica

Lenguaje aceptado

$$\alpha \in L(\mathcal{A}) \iff \exists q_f \in F \mid (q_0, \alpha) \vdash_{\mathcal{A}}^* (q_f, \lambda)$$

A'_3 :



Para la cadena $\alpha = 101 \in L_3$ tenemos dos caminos posibles

Seguimientos

A'_3 :



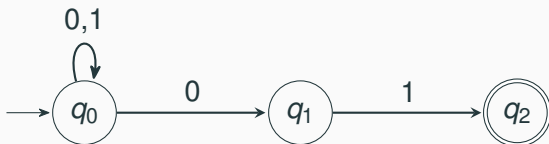
Para la cadena $\alpha = 101 \in L_3$ tenemos dos caminos posibles

- $(q_0, 101) \vdash_{\mathcal{A}'_3} (q_0, 01) \vdash_{\mathcal{A}'_3} (q_1, 1) \vdash_{\mathcal{A}'_3} (q_2, \lambda)$ ✓
- $(q_0, 101) \vdash_{\mathcal{A}'_3} (q_0, 01) \vdash_{\mathcal{A}'_3} (q_0, 1) \vdash_{\mathcal{A}'_3} (q_0, \lambda)$ ✗

Por la definición de lenguaje aceptado, alcanza con que *exista al menos una* secuencia de configuraciones que lleve a un estado final consumiendo toda la cadena para que pertenezca al lenguaje. **No importa que otras secuencias no lleven a aceptar.**

Más seguimientos

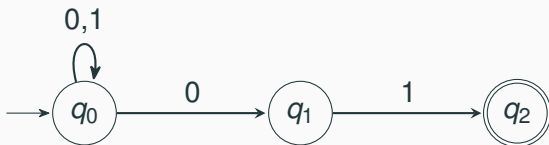
\mathcal{A}'_3 :



Para la cadena $\beta = 010 \notin \mathcal{L}_3$ tenemos tres caminos posibles

Más seguimientos

\mathcal{A}'_3 :



Para la cadena $\beta = 010 \notin \mathcal{L}_3$ tenemos tres caminos posibles

- $(q_0, 010) \vdash_{\mathcal{A}'_3} (q_0, 10) \vdash_{\mathcal{A}'_3} (q_0, 0) \vdash_{\mathcal{A}'_3} (q_1, \lambda)$ ✗
 $(q_0, 010) \vdash_{\mathcal{A}'_3} (q_0, 10) \vdash_{\mathcal{A}'_3} (q_0, 0) \vdash_{\mathcal{A}'_3} (q_0, \lambda)$ ✗

Consumen toda la cadena pero no llegan a un estado final

- $(q_0, 010) \vdash_{\mathcal{A}'_3} (q_1, 10) \vdash_{\mathcal{A}'_3} (q_2, 0)$ ✗

Llega a un estado final **pero no consume toda la cadena**

Aclaración sobre lenguaje aceptado

Lenguaje aceptado

$$\alpha \in L(\mathcal{A}) \iff \exists q_f \in F \mid (q_0, \alpha) \vdash_{\mathcal{A}}^* (q_f, \lambda)$$

¡Hay que consumir toda la cadena!

Un autómata finito solo acepta una cadena si puede consumirla toda y terminar en un estado final. No alcanza solo con llegar a un estado final, **tiene que consumir toda la cadena**.

Esto suele generar confusión sobre todo con autómatas no determinísticos.

Ejercicio 4 - Unión

Ejercicio 4

$\mathcal{L}_4 = \mathcal{L}_1 \cup \mathcal{L}_3$. Con \mathcal{L}_1 = cadenas con cantidad par de 0s y \mathcal{L}_3 = cadenas que terminan en 01. ¿Qué significa \mathcal{L}_4 ?

Ejercicio 4 - Unión

Ejercicio 4

$\mathcal{L}_4 = \mathcal{L}_1 \cup \mathcal{L}_3$. Con \mathcal{L}_1 = cadenas con cantidad par de 0s y \mathcal{L}_3 = cadenas que terminan en 01. ¿Qué significa \mathcal{L}_4 ?
Cadenas que terminen en 01 \bullet tengan cantidad par de 0s.

Pista:

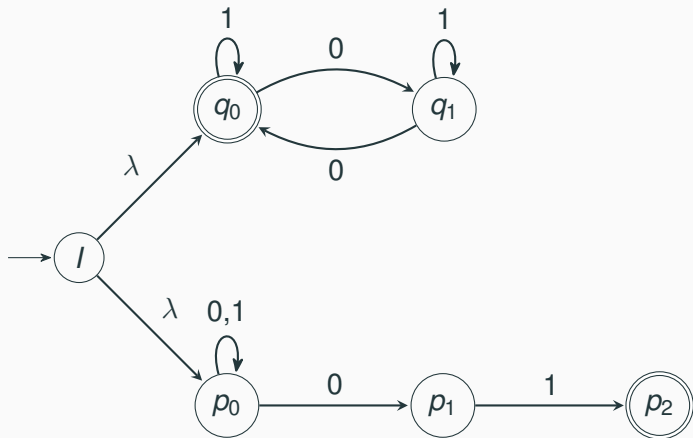
Ejercicio 4 - Unión

Ejercicio 4

$\mathcal{L}_4 = \mathcal{L}_1 \cup \mathcal{L}_3$. Con \mathcal{L}_1 = cadenas con cantidad par de 0s y \mathcal{L}_3 = cadenas que terminan en 01. ¿Qué significa \mathcal{L}_4 ?
Cadenas que terminen en 01 \bullet tengan cantidad par de 0s.

Pista: ¿Cómo podemos usar \mathcal{A}_1 y \mathcal{A}'_3 ?

Ejercicio 4 - Unión



$$\mathcal{A}_4 = \langle \{I, q_0, q_1, p_0, p_1, p_2\}, \{0, 1\}, \delta, I, \{q_0, p_2\} \rangle$$

Ejercicio 5 - Complemento

Ejercicio 5.a

- a. Con $\mathcal{L}_2 =$ cadenas que comienzan por 01,
 $\mathcal{L}_2^c =$

Ejercicio 5 - Complemento

Ejercicio 5.a

- a. Con $\mathcal{L}_2 =$ cadenas que comienzan por 01,
 $\mathcal{L}_2^c =$ cadenas que no comienzan por 01

Pista:

Ejercicio 5 - Complemento

Ejercicio 5.a

- a. Con \mathcal{L}_2 = cadenas que comienzan por 01,
 \mathcal{L}_2^c = cadenas que no comienzan por 01

Pista: ¿Cómo podemos usar \mathcal{A}_2 ?



Ejercicio 5 - Complemento

Ejercicio 5.a

- a. Con \mathcal{L}_2 = cadenas que comienzan por 01,
 \mathcal{L}_2^c = cadenas que no comienzan por 01

Pista: ¿Cómo podemos usar \mathcal{A}_2 ?



Convención del trampa implícito

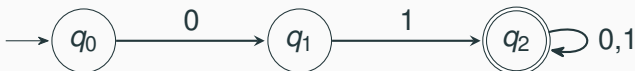
Vamos a tomar el estado trampa como **implícito** cuando un autómata sea determinístico (no haya más de una opción) pero tenga δ indefinida para algunas transiciones.

Ejercicio 5 - Complemento

Ejercicio 5.a

- a. Con $\mathcal{L}_2 =$ cadenas que comienzan por 01,
 $\mathcal{L}_2^c =$ cadenas que no comienzan por 01

Pista: ¿Cómo podemos usar \mathcal{A}_2 ?

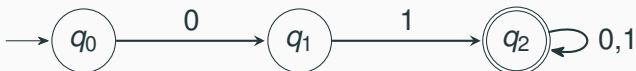


Ejercicio 5 - Complemento

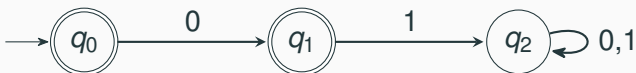
Ejercicio 5.a

- a. Con \mathcal{L}_2 = cadenas que comienzan por 01,
 \mathcal{L}_2^c = cadenas que no comienzan por 01

Pista: ¿Cómo podemos usar \mathcal{A}_2 ?



¡**Invertimos** los estados finales!



Pero

Ejercicio 5 - Complemento

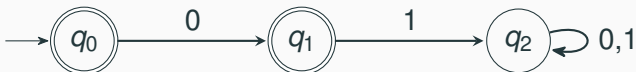
Ejercicio 5.a

- a. Con \mathcal{L}_2 = cadenas que comienzan por 01,
 \mathcal{L}_2^c = cadenas que no comienzan por 01

Pista: ¿Cómo podemos usar \mathcal{A}_2 ?



¡Invertimos los estados finales!

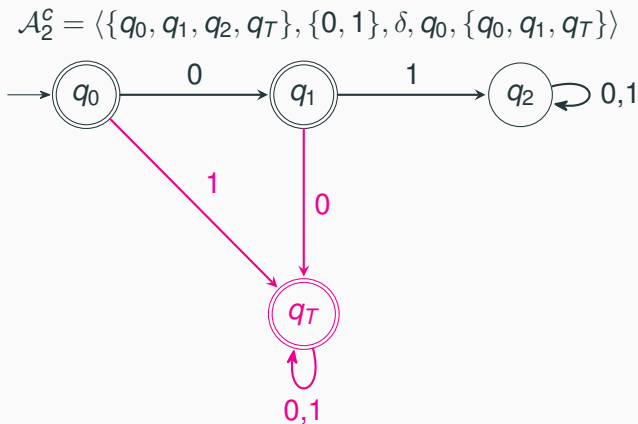


Pero $111 \notin \mathcal{L}_2$ (no arranca con 01) y, ¡no la reconoce! **El autómata tiene que estar completo**, sino perdemos cadenas.

Ejercicio 5 - Complemento

Ejercicio 5.a

a. \mathcal{L}_2^c = cadenas que no comienzan por 01



Ejercicio 5 - Complemento

Ejercicio 5.b

a. \mathcal{L}_3^c = cadenas que no terminan con 01

Ejercicio 5 - Complemento

Ejercicio 5.b

a. \mathcal{L}_3^c = cadenas que no terminan con 01

Candidato:



Ejercicio 5 - Complemento

Ejercicio 5.b

a. \mathcal{L}_3^c = cadenas que no terminan con 01

Candidato:



¡No funciona! Aceptamos cadenas demás como 01 (en particular en este caso aceptamos Σ^*). Para cada cadena que aceptábamos, había caminos que no aceptaban. Entonces si invertimos los finales, esos caminos pueden pasar a ser de aceptación (excepto que se traben) y aceptamos cadenas que no deberíamos.

El autómata tiene que ser determinístico.

Ejercicio 5 - Complemento

Ejercicio 5.b

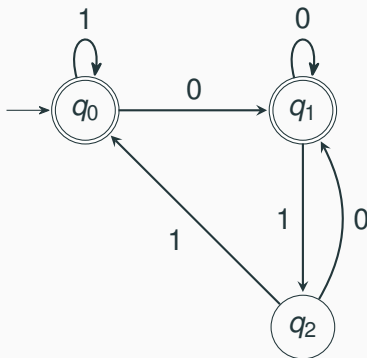
b. \mathcal{L}_3^c = cadenas que no terminan con 01

Ejercicio 5 - Complemento

Ejercicio 5.b

b. \mathcal{L}_3^c = cadenas que no terminan con 01

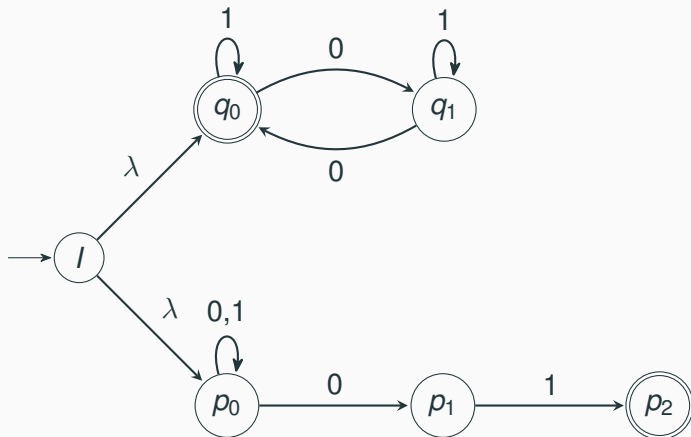
$$\mathcal{A}_3^c = \langle \{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_0, q_1\} \rangle$$



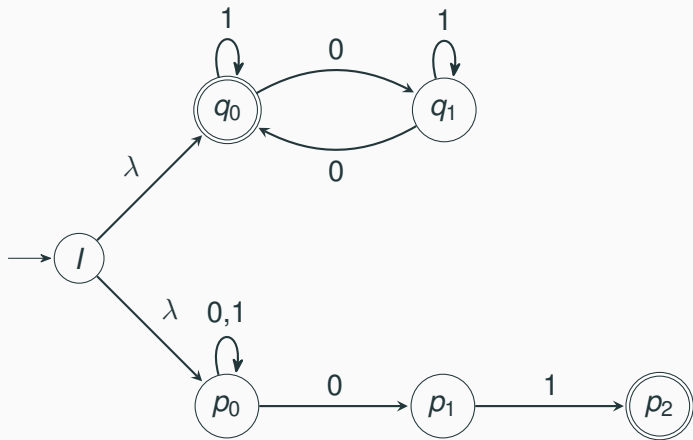
Ejercicio 6 - Reversa

Ejercicio 6

\mathcal{L}_4^r , con $\mathcal{L}_4 = \mathcal{L}_1 \cup \mathcal{L}_3$, cadenas que terminan en 01 o tienen cantidad par de 0s



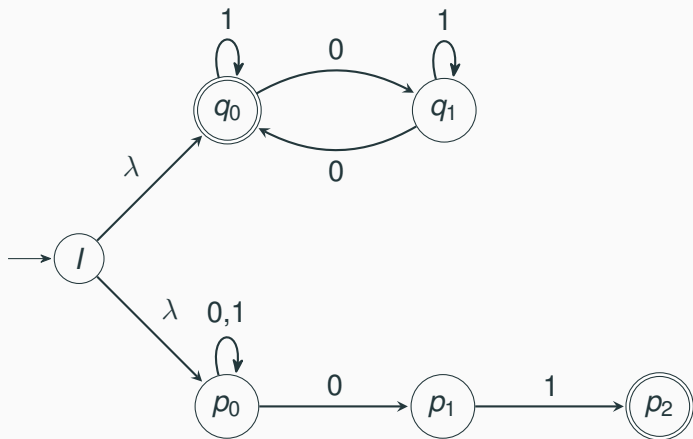
Ejercicio 6 - Reversa



$$\mathcal{A}_4 = \langle \{I, q_0, q_1, p_0, p_1, p_2\}, \{0, 1\}, \delta, I, \{q_0, p_2\} \rangle$$

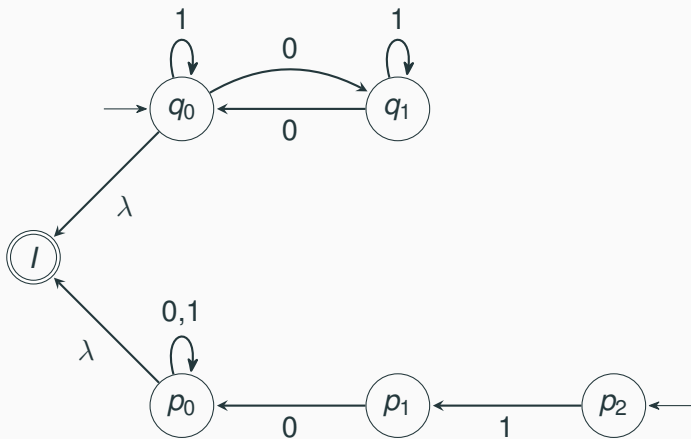
Solución:

Ejercicio 6 - Reversa

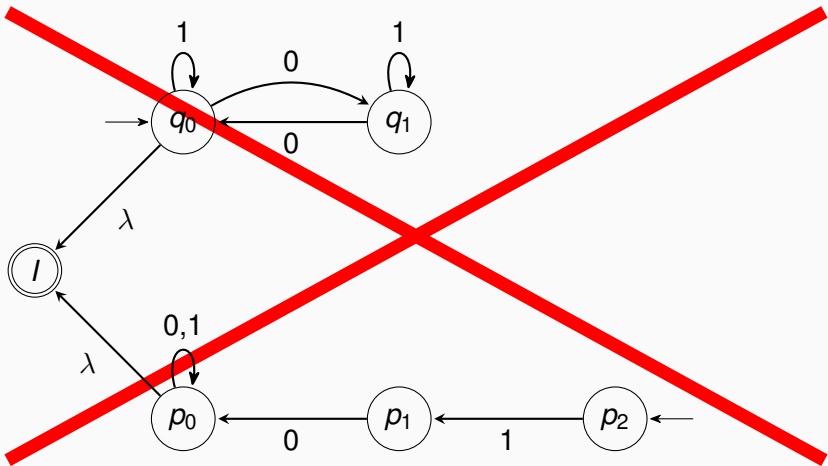


Solución: Ejecutamos el autómata al revés: damos vuelta las flechas e invertimos los finales e iniciales.

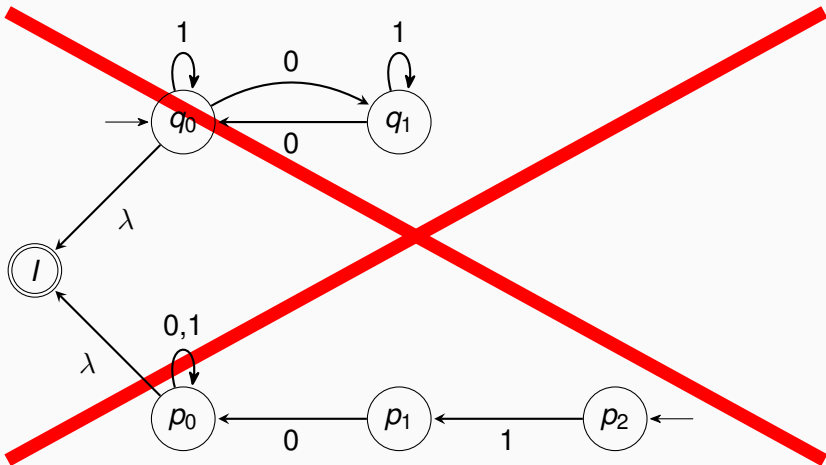
Ejercicio 6 - Reversa



Ejercicio 6 - Reversa

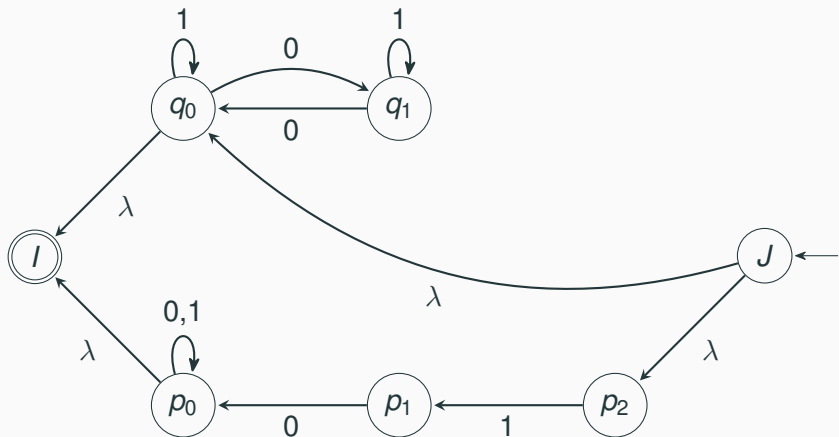


Ejercicio 6 - Reversa



¡No podemos tener mas de un estado inicial! Agregamos un estado inicial con transiciones λ

Ejercicio 6 - Reversa



$$\mathcal{A}_4^r = \langle \{I, J, q_0, q_1, p_0, p_1, p_2\}, \{0, 1\}, \delta, J, \{I\} \rangle$$

Reversa

Sea $\mathcal{A} = \langle Q, \Sigma, \delta, q_0, F \rangle$ un AFND- λ y $\mathcal{A}^r = \langle Q', \Sigma, \delta^r, q'_0, F' \rangle$ tal que:

- $Q' = Q \cup \{q'_0\}$ (nuevo inicial)
- $\delta^r(q'_0, \lambda) = F$ (arrancar por los finales)
- $q_2 \in \delta'(q_1, a) \iff q_1 \in \delta(q_2, a)$ (dar vuelta flechas)
- $F' = \{q_0\}$ (terminar en el estado que era inicial)

Luego $(L(\mathcal{A}))^r = L(\mathcal{A}^r)$

Paso intermedio

Queremos demostrar primero que:

$$(\forall q, p \in Q)((q, \alpha) \vdash_{\mathcal{A}}^* (p, \lambda) \iff (p, \alpha^r) \vdash_{\mathcal{A}^r}^* (q, \lambda))$$

Paso intermedio

Queremos demostrar primero que:

$$(\forall q, p \in Q)((q, \alpha) \vdash_{\mathcal{A}}^* (p, \lambda) \iff (p, \alpha^r) \vdash_{\mathcal{A}^r}^* (q, \lambda))$$

Vamos usar inducción estructural sobre α

Caso base

$\alpha = \lambda$:

Queremos probar que

$$(q, \lambda) \vdash_{\mathcal{A}}^* (r, \lambda) \iff (p, \lambda) \vdash_{\mathcal{A}^r}^* (q, \lambda)$$

Caso base

$\alpha = \lambda$:

Queremos probar que

$$(q, \lambda) \vdash_{\mathcal{A}}^* (r, \lambda) \iff (p, \lambda) \vdash_{\mathcal{A}^r}^* (q, \lambda)$$

Vamos a hacer induccion en la cantidad de transiciones. Es decir

$$(q, \lambda) \vdash_{\mathcal{A}}^i (p, \lambda) \iff (p, \lambda) \vdash_{\mathcal{A}^r}^i (q, \lambda)$$

- Caso base, $i = 1$:

- Caso base, $i = 1$:

$$(q, \lambda) \vdash_{\mathcal{A}}^1 (p, \lambda)$$

$$\stackrel{\text{def } \vdash}{\iff} p \in \delta(q, \lambda)$$

$$\stackrel{\text{def } \delta^r}{\iff} q \in \delta^r(p, \lambda)$$

$$\stackrel{\text{def } \vdash}{\iff} (p, \lambda) \vdash_{\mathcal{A}^r}^1 (q, \lambda)$$

Demostración reversa

- Paso inductivo. Vale para n , queremos ver que vale para $n + 1$:

Demostración reversa

- Paso inductivo. Vale para n , queremos ver que vale para $n + 1$:

$$(q, \lambda) \vdash_{\mathcal{A}}^{n+1} (p, \lambda)$$

$$\stackrel{\text{def } \vdash^*}{\iff} \exists s \mid (q, \lambda) \vdash_{\mathcal{A}}^1 (s, \lambda) \wedge (s, \lambda) \vdash_{\mathcal{A}}^n (p, \lambda)$$

$$\stackrel{\text{def } \vdash}{\iff} \exists s \mid s \in \delta(q, \lambda) \wedge (s, \lambda) \vdash_{\mathcal{A}}^n (p, \lambda)$$

$$\stackrel{\text{def } \delta^r}{\iff} \exists s \mid q \in \delta^r(s, \lambda) \wedge (s, \lambda) \vdash_{\mathcal{A}}^n (p, \lambda)$$

$$\stackrel{\text{HI}}{\iff} \exists s \mid q \in \delta^r(s, \lambda) \wedge (p, \lambda) \vdash_{\mathcal{A}^r}^n (s, \lambda)$$

$$\stackrel{\text{def } \vdash}{\iff} \exists s \mid (s, \lambda) \vdash_{\mathcal{A}^r}^1 (q, \lambda) \wedge (p, \lambda) \vdash_{\mathcal{A}^r}^n (s, \lambda)$$

$$\stackrel{\text{reordenando}}{\iff} \exists s \mid (p, \lambda) \vdash_{\mathcal{A}^r}^n (s, \lambda) \wedge (s, \lambda) \vdash_{\mathcal{A}^r}^1 (q, \lambda)$$

$$\stackrel{\text{def } \vdash^{n+1}}{\iff} (p, \lambda) \vdash_{\mathcal{A}^r}^{n+1} (q, \lambda)$$

Demostración reversa

Demostramos así entonces que

$$(q, \lambda) \vdash_{\mathcal{A}}^i (p, \lambda) \iff (p, \lambda) \vdash_{\mathcal{A}^r}^i (q, \lambda).$$

y en particular que

$$(q, \lambda) \vdash_{\mathcal{A}}^* (p, \lambda) \iff (p, \lambda) \vdash_{\mathcal{A}^r}^* (q, \lambda).$$

Con esto queda probado el caso base.

Paso inductivo

Vale para ω

Queremos probar que vale para $\alpha = a\omega$ ($\alpha^r = \omega^r a$):

$$(q, a\omega) \vdash_{\mathcal{A}}^* (p, \lambda) \iff (p, \omega^r a) \vdash_{\mathcal{A}^r}^* (q, \lambda)$$

Demostración reversa

Paso inductivo:

$$\begin{aligned} & (q, a\omega) \vdash_{\mathcal{A}}^* (p, \lambda) \\ & \stackrel{\text{def } \vdash^*}{\iff} \exists s \mid (q, a\omega) \vdash_{\mathcal{A}} (s, \omega) \wedge (s, \omega) \vdash_{\mathcal{A}}^* (p, \lambda) \\ & \stackrel{\text{def } \vdash}{\iff} \exists s \mid s \in \delta(q, a) \wedge (s, \omega) \vdash_{\mathcal{A}}^* (p, \lambda) \\ & \stackrel{\text{def } \delta^r}{\iff} \exists s \mid q \in \delta^r(s, a) \wedge (s, \omega) \vdash_{\mathcal{A}}^* (p, \lambda) \\ & \stackrel{\text{HI}}{\iff} \exists s \mid q \in \delta^r(s, a) \wedge (p, \omega^r) \vdash_{\mathcal{A}^r}^* (s, \lambda) \\ & \stackrel{\text{reordenando}}{\iff} \exists s \mid (p, \omega^r) \vdash_{\mathcal{A}^r}^* (s, \lambda) \wedge q \in \delta^r(s, a) \\ & \stackrel{\text{def } \vdash}{\iff} \exists s \mid (p, \omega^r) \vdash_{\mathcal{A}^r}^* (s, \lambda) \wedge (s, a) \vdash_{\mathcal{A}^r}^* (q, \lambda) \\ & \stackrel{\omega^r \cdot a \wedge \lambda \cdot a}{\iff} \exists s \mid (p, \omega^r a) \vdash_{\mathcal{A}^r}^* (s, a) \wedge (s, a) \vdash_{\mathcal{A}^r}^* (q, \lambda) \\ & \stackrel{\text{def } \vdash^*}{\iff} (p, \omega^r a) \vdash_{\mathcal{A}^r}^* (q, \lambda) \end{aligned}$$

Demostración reversa

Paso inductivo:

$$\begin{aligned} & (q, a\omega) \vdash_{\mathcal{A}}^* (p, \lambda) \\ & \xLeftrightarrow{\text{def } \vdash^*} \exists s \mid (q, a\omega) \vdash_{\mathcal{A}} (s, \omega) \wedge (s, \omega) \vdash_{\mathcal{A}}^* (p, \lambda) \\ & \xLeftrightarrow{\text{def } \vdash} \exists s \mid s \in \delta(q, a) \wedge (s, \omega) \vdash_{\mathcal{A}}^* (p, \lambda) \\ & \xLeftrightarrow{\text{def } \delta^r} \exists s \mid q \in \delta^r(s, a) \wedge (s, \omega) \vdash_{\mathcal{A}}^* (p, \lambda) \\ & \xLeftrightarrow{\text{HI}} \exists s \mid q \in \delta^r(s, a) \wedge (p, \omega^r) \vdash_{\mathcal{A}^r}^* (s, \lambda) \\ & \xLeftrightarrow{\text{reordenando}} \exists s \mid (p, \omega^r) \vdash_{\mathcal{A}^r}^* (s, \lambda) \wedge q \in \delta^r(s, a) \\ & \xLeftrightarrow{\text{def } \vdash} \exists s \mid (p, \omega^r) \vdash_{\mathcal{A}^r}^* (s, \lambda) \wedge (s, a) \vdash_{\mathcal{A}^r}^* (q, \lambda) \\ & \xLeftrightarrow{\omega^r \cdot a \wedge \lambda \cdot a} \exists s \mid (p, \omega^r a) \vdash_{\mathcal{A}^r}^* (s, a) \wedge (s, a) \vdash_{\mathcal{A}^r}^* (q, \lambda) \\ & \xLeftrightarrow{\text{def } \vdash^*} (p, \omega^r a) \vdash_{\mathcal{A}^r}^* (q, \lambda) \end{aligned}$$

Entonces $(\forall q, p \in Q)((q, \alpha) \vdash_{\mathcal{A}}^* (p, \lambda) \iff (p, \alpha^r) \vdash_{\mathcal{A}^r}^* (q, \lambda))$

Recordemos

Queríamos demostrar que $(L(\mathcal{A}))^r = L(A^r)$

Demostración reversa

Recordemos

Queríamos demostrar que $(L(\mathcal{A}))^r = L(A^r)$

$$(L(\mathcal{A}))^r = L(A^r)$$

$$\begin{array}{c} \text{aplicamos reverso} \\ \iff \end{array} ((L(\mathcal{A}))^r)^r = (L(A^r))^r$$

$$\begin{array}{c} (\mathcal{L}^r)^r = \mathcal{L} \\ \iff \end{array} L(\mathcal{A}) = (L(A^r))^r$$

$$\iff (w \in L(\mathcal{A})) \iff w \in (L(A^r))^r$$

Demostración reversa

Entonces

$$w \in L(\mathcal{A})$$

$$\stackrel{\text{def}}{\iff} \exists q_f \in F \mid (q_0, w) \vdash_{\mathcal{A}}^* (q_f, \lambda)$$

$$\stackrel{\text{propiedad}}{\iff} \exists q_f \in F \mid (q_f, w^r) \vdash_{\mathcal{A}^r}^* (q_0, \lambda)$$

$$\stackrel{\delta^r(q'_0, \lambda) = F}{\iff} \exists q_f \in F \mid (q'_0, w^r) \vdash_{\mathcal{A}^r} (q_f, w^r) \wedge (q_f, w^r) \vdash_{\mathcal{A}^r}^* (q_0, \lambda)$$

$$\stackrel{\text{def } \vdash^*}{\iff} w^r \in L(\mathcal{A}^r)$$

$$\stackrel{\text{def}}{\iff} w \in (L(\mathcal{A}^r))^r$$



Operaciones en general

Unión

Dados \mathcal{A}_1 y \mathcal{A}_2 AFs, para obtener $L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$ agregar un nuevo estado inicial con transiciones λ a los iniciales de \mathcal{A}_1 y \mathcal{A}_2 .

Complemento

Dado un AFD **completo**, invertir los finales: $F' = F \setminus Q$

Reversa

Dado un AFND- λ , obtener $\mathcal{A}' = \langle Q', \Sigma, \delta', q'_0, F' \rangle$ tal que:

- $Q' = Q \cup \{q'_0\}$ (nuevo inicial)
- $\delta'(q'_0, \lambda) = F$ (arrancar por los finales)
- $q_2 \in \delta'(q_1, a) \iff q_1 \in \delta(q_2, a)$ (dar vuelta flechas)
- $F' = \{q_0\}$ (terminar en el estado que era inicial)

Vimos,

- AFDs, AFNDs y sus definiciones formales
- La importancia de que cada estado tenga un propósito claro
- Problemas que son más sencillos de resolver con AFNDs
- Algunos autómatas para operaciones entre lenguajes:
Unión, complemento, reversa (más en la práctica)

Ya pueden hacer toda la práctica 2