Autómatas finitos

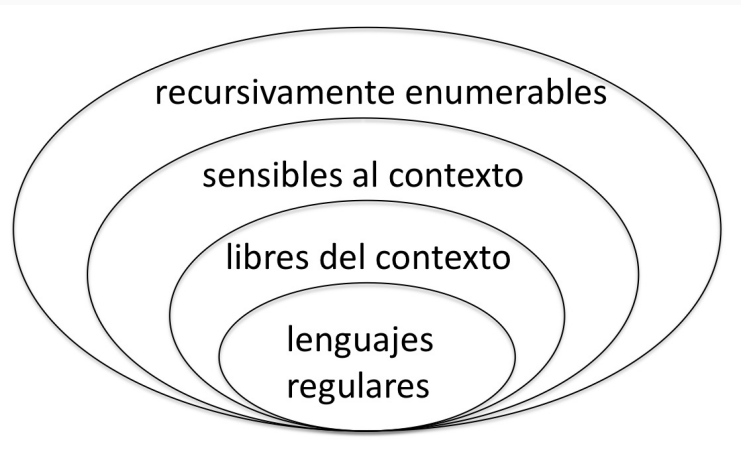
Lenguajes formales, autómatas y computabilidad

DC-UBA

2do cuatrimestre 2024

Autómatas finitos

- La clase pasada vimos símbolos, 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

- Consumen cadenas símbolo por símbolo y mantienen un estado interno.

Ejercicio 1 - Repaso y definiciones

Ejemplo motivador (Ej 1b de la práctica 1)

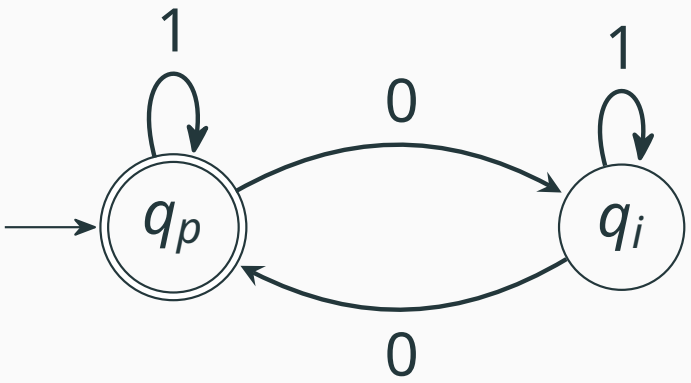
$$\mathcal{L}_1 = \{\alpha \mid \alpha \in \{0,1\}^* \text{ y } |\alpha|_0 \text{ es par } \}$$

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

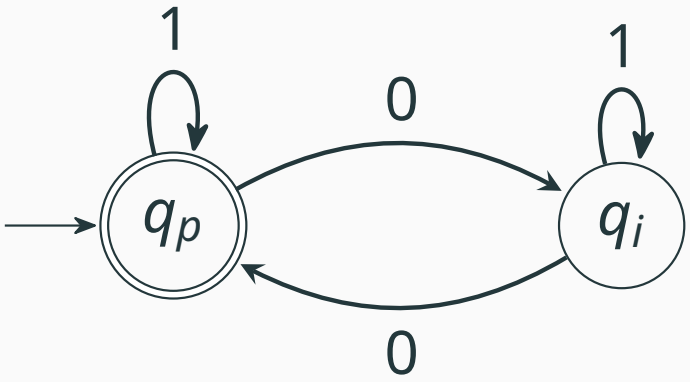
Ejemplos de cadenas:

- Que pertenecen: 1, 00, 010, 00100010
- Que no pertenecen: 0, 10010, 1110

El autómata $\mathcal{A}_1$ que reconoce $\mathcal{L}_1$ es,



Seguimientos de cadenas



- $q_p \xrightarrow[\mathcal{A}_1]{0} q_i \xrightarrow[\mathcal{A}_1]{1} q_i \xrightarrow[\mathcal{A}_1]{0} q_p$ ✓
- $q_p \xrightarrow[\mathcal{A}_1]{1} q_p \xrightarrow[\mathcal{A}_1]{0} q_i \xrightarrow[\mathcal{A}_1]{1} q_i$ ✗

Definición

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

Volviendo al ejemplo

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

$$\mathcal{A}_1 = \langle \{q_p, q_i\}, \{0, 1\}, \delta, q_p, \{q_p\} \rangle$$

y antes dimos una representación pictórica de δ, que más formalmente está dada por la siguiente tabla

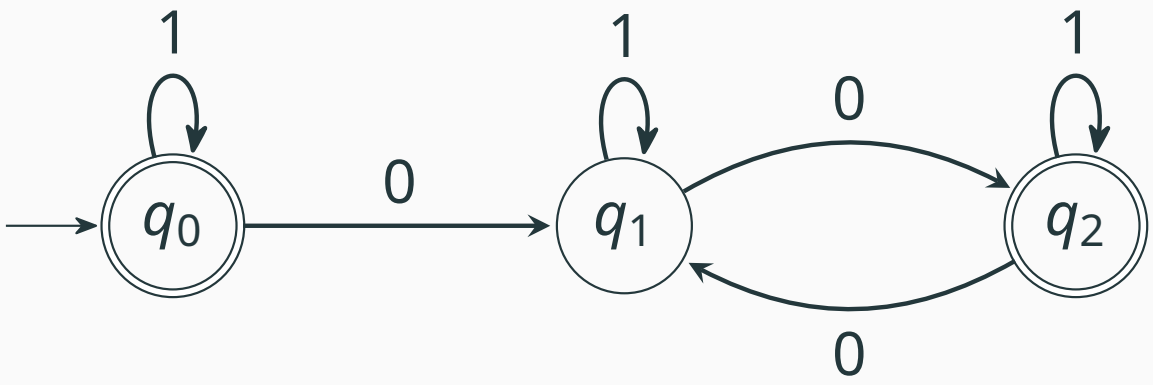
δ	0	1
q_p	q_i	q_p
q_i	q_p	q_i

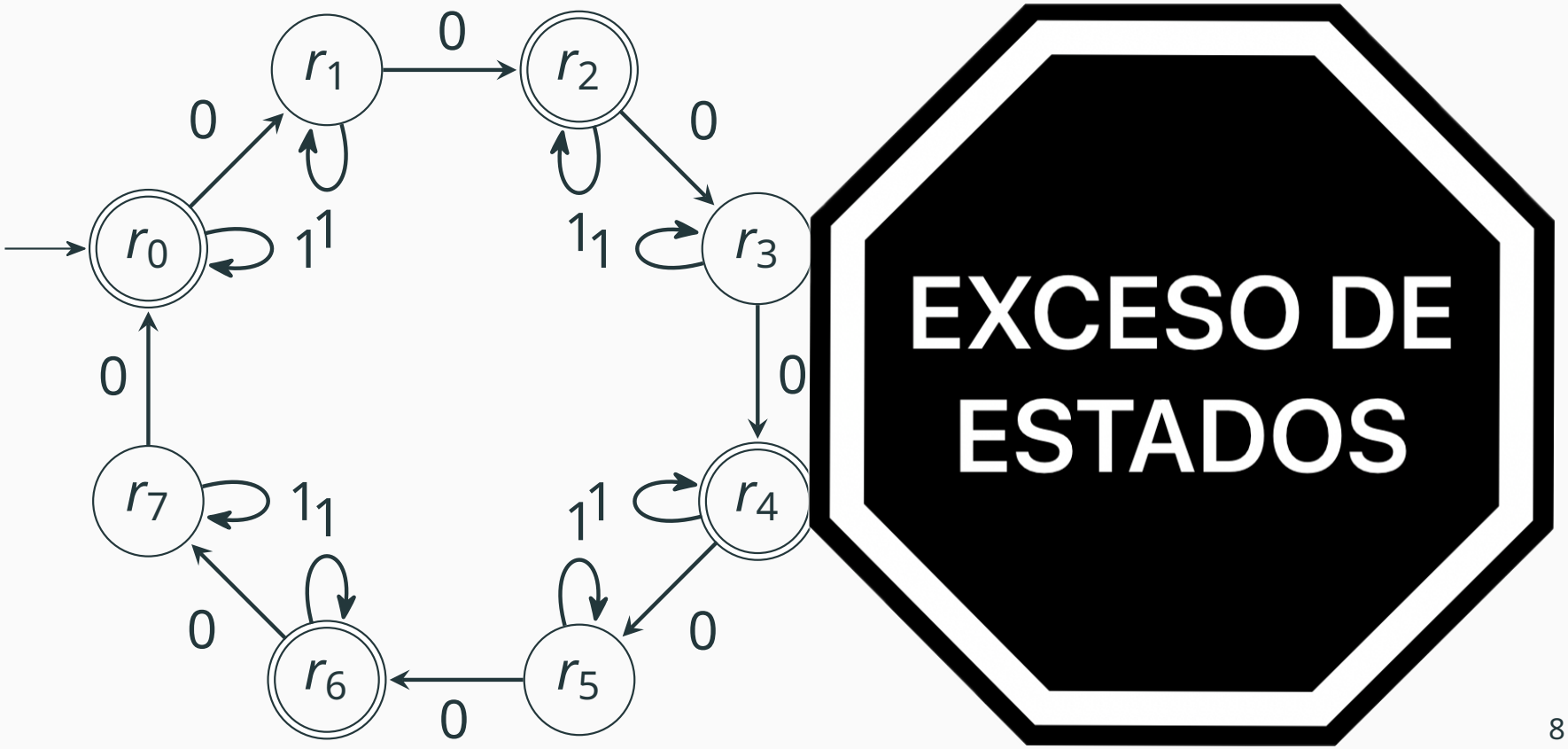
La tupla y los parciales

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

¡No olviden la tupla!

Hay otros?


1
1
1
0
0
q_0
q_1
q_2
0

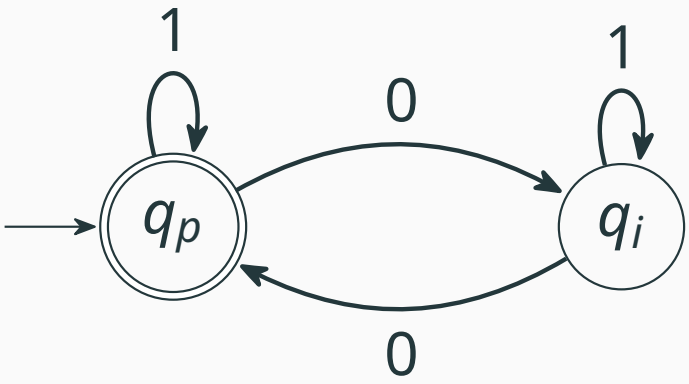

r_1
0
r_2
0
0
r_0
1
1
1
r_3
0
0
r_7
1
1
1
1
r_4
0
0
r_6
r_5
0
EXCESO DE ESTADOS

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? 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 autómata?

Lenguaje aceptado

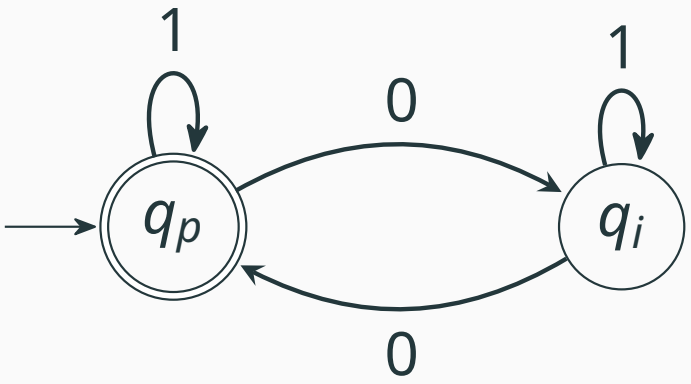
$$\alpha \in L(\mathcal{A}) \iff \exists q_f \in F \mid (q_0, \alpha) \vdash_{\mathcal{A}}^{*} (q_f, \lambda)$$

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. ¿Cuando necesitamos aplicarla cero veces? Cuando la entrada es λ.

Seguimientos de cadenas



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

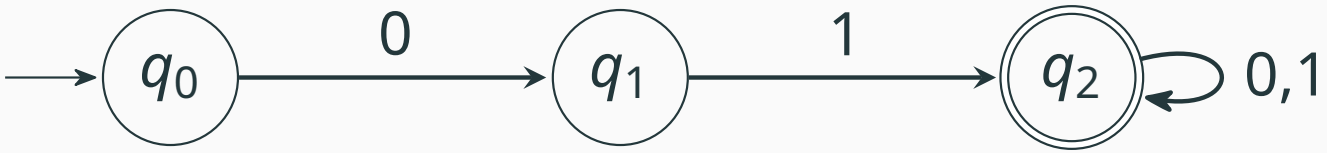
Ejercicio 2

Enunciado

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

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

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

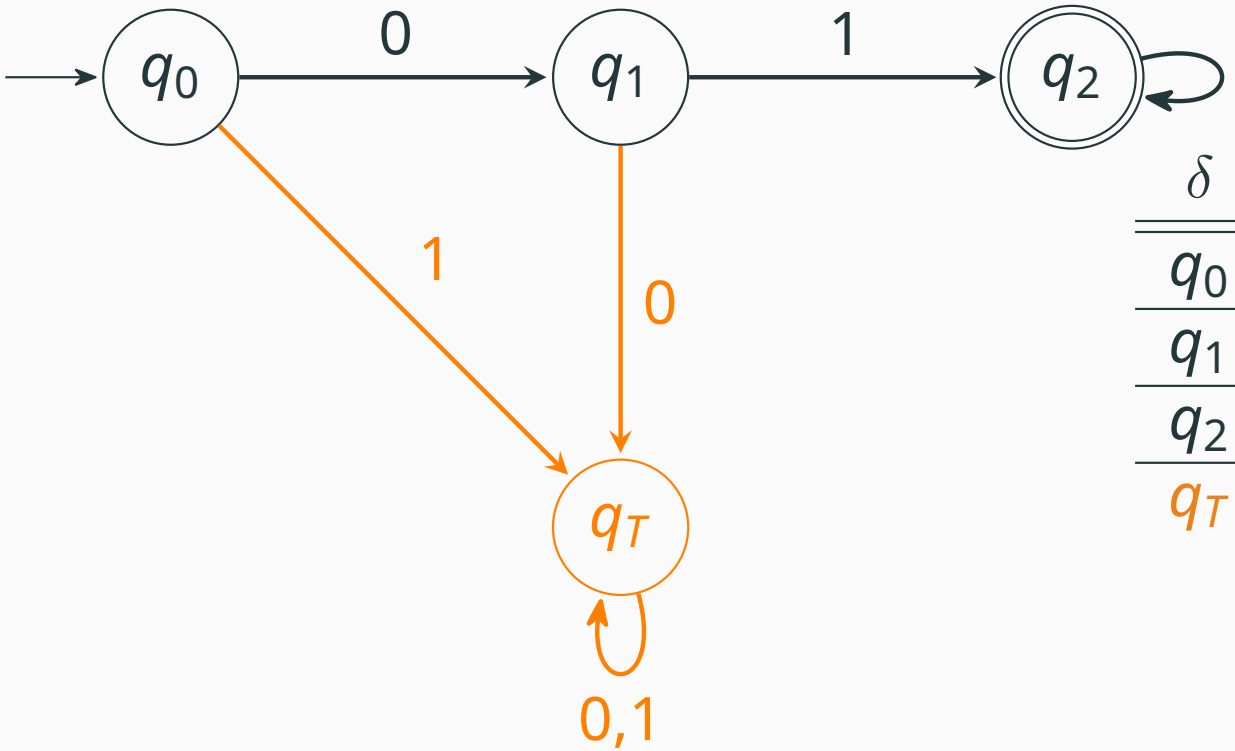


¿Qué problema tiene?
¡La función de transición está incompleta!

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



δ	0	1
q_0	q_1	q_T
q_1	q_T	q_2
q_2	q_2	q_2
q_T	q_T	q_T

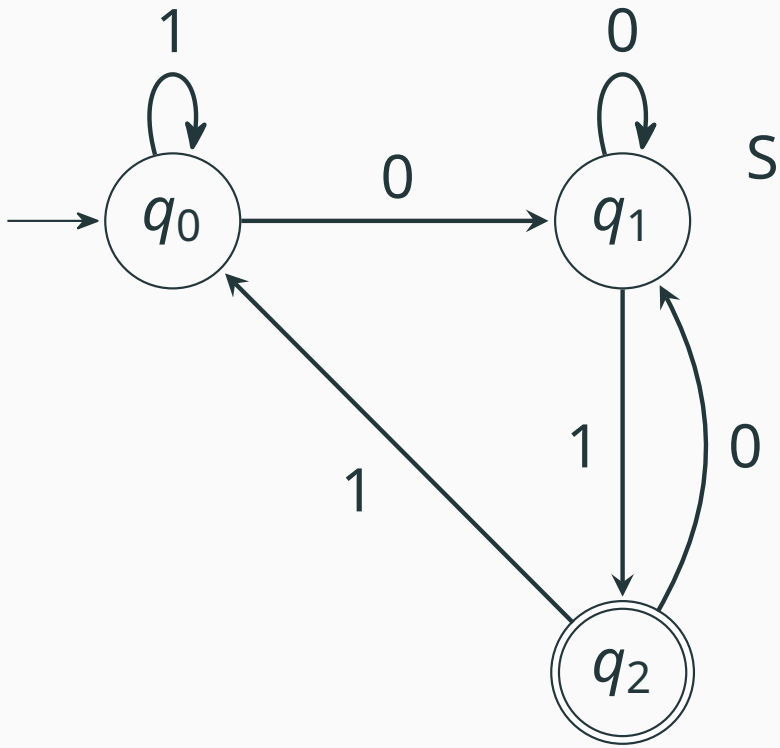
$$\mathcal{A}_2 = \langle \{q_0, q_1, q_2, q_T\}, \{0, 1\}, \delta, q_0, \{q_2\} \rangle$$

Ejercicio 3

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

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

$\mathcal{A}_3$:



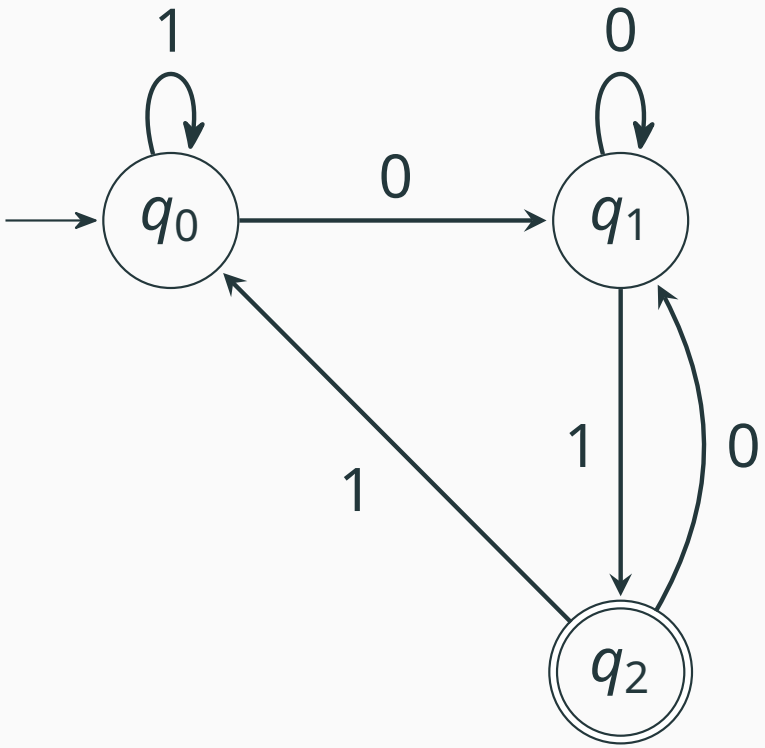
Significado intuitivo de cada estado:

- 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

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

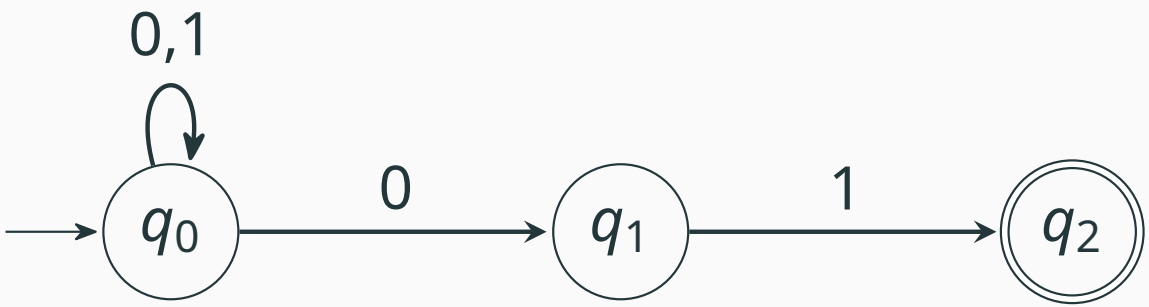
$\mathcal{A}_3$:



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

Alternativa

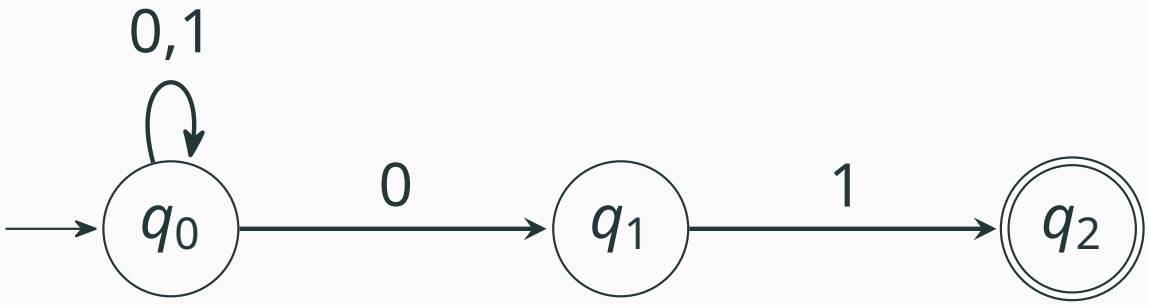
El lenguaje $\mathcal{L}_3$ es muy parecido a $\mathcal{L}_2$, pero el autómata se ve más complicado. Nos gustaría que el formalismo nos permita expresar algo como “Puede venir cualquier cadena, siempre y cuando termine con 01”, al igual que para $\mathcal{L}_2$ nos permitía decir “Si arranca por 01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

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

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**.

AFND-λ

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:

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

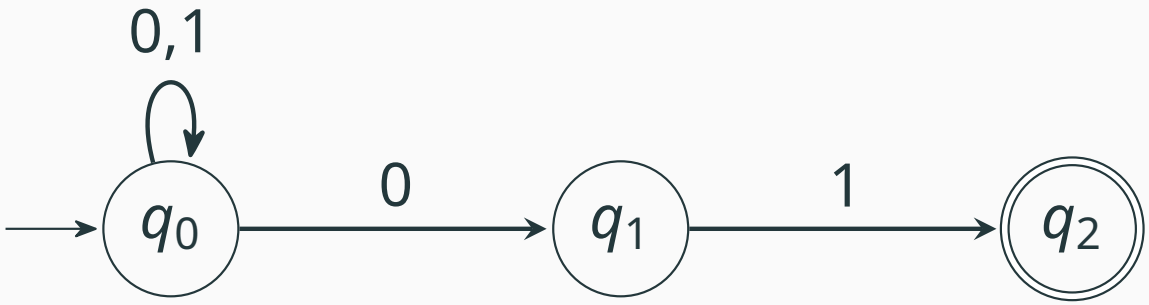
- En lugar de un solo estado, devuelve un conjunto
- Además de transiciones por un símbolo de la entrada, podemos transicionar por λ [1] sin consumir ningún símbolo

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.

[1] No 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? Hay dos casos: consumiendo un símbolo o por λ

Relación de transición entre configuraciones

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

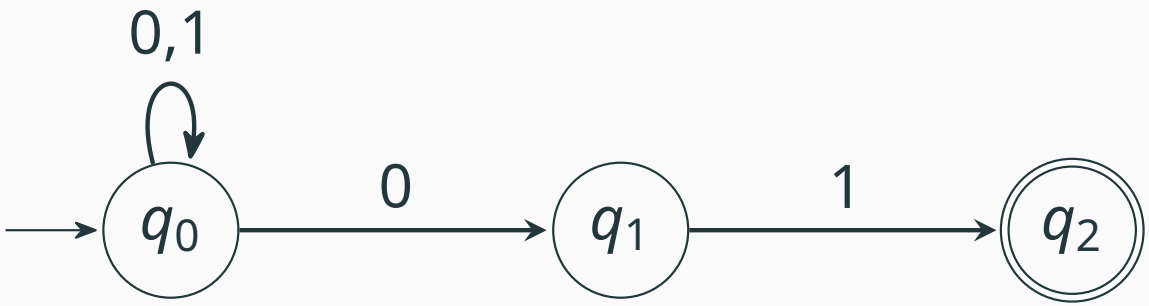
La definición del lenguaje es idéntica

Lenguaje aceptado

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

Seguimientos

A'_3:



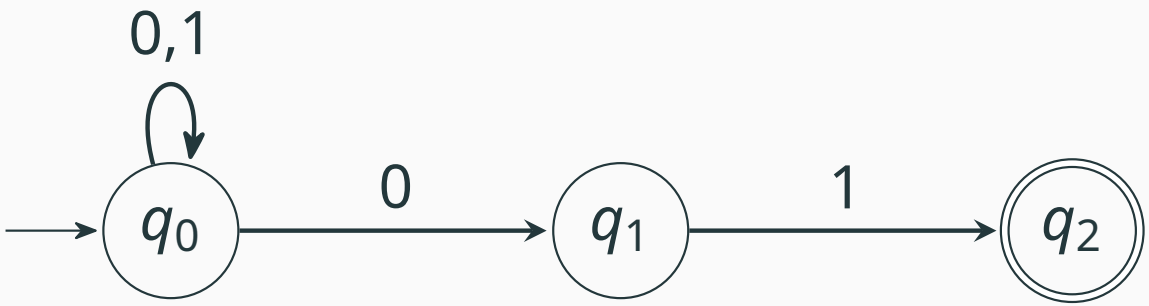
Para la cadena $\alpha = 101 \in L_3$ tenemos dos caminos posibles

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

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

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

  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.

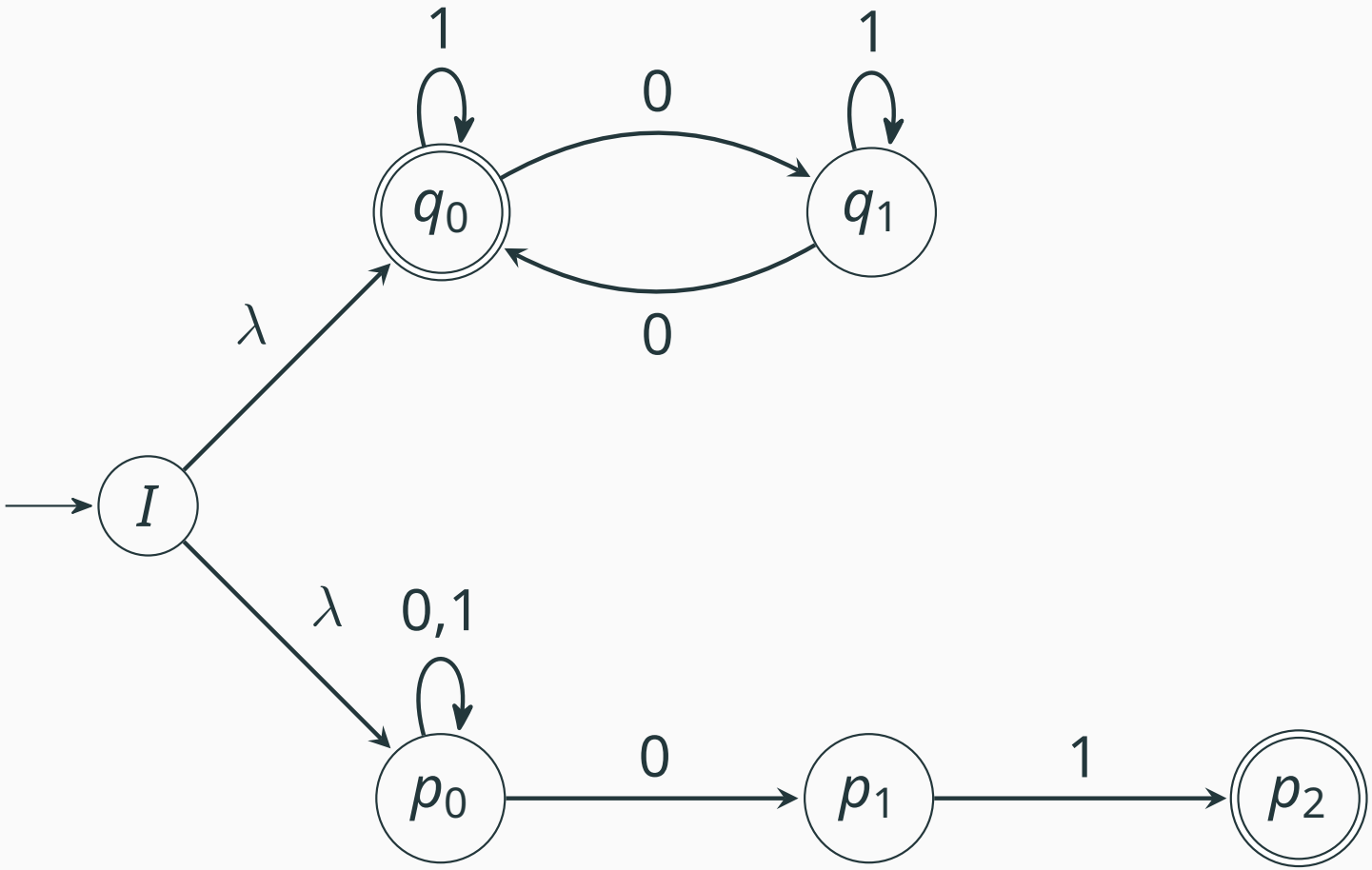
Ejercicio 4 - Unión

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 $\mathcal{L}_4$? Cadenas que terminen en 01 **o** tengan cantidad par de 0s.

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

Ejercicio 4 - Unión



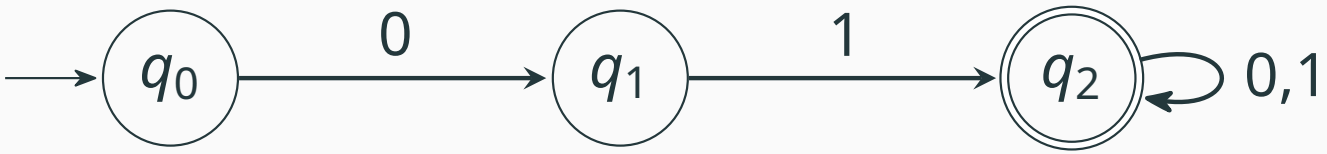
$$\mathcal{A}_4 = \langle \{I, q_0, q_1, p_0, p_1, p_2\}, \{0, 1\}, \delta, I, \{q_0, p_2\} \rangle$$

Ejercicio 5 - Complemento

Ejercicio 5.a

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

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



Convención del trampa implícito

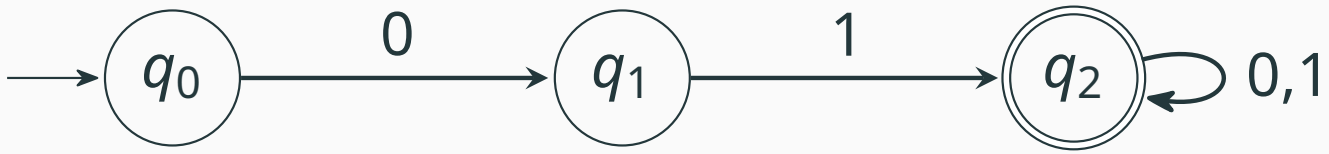
Vamos a tomar el estado trampa como **implícito** cuando un autómata sea determinístico (no haya más de una opción) pero tenga δ indefinida para algunas transiciones.

Ejercicio 5 - Complemento

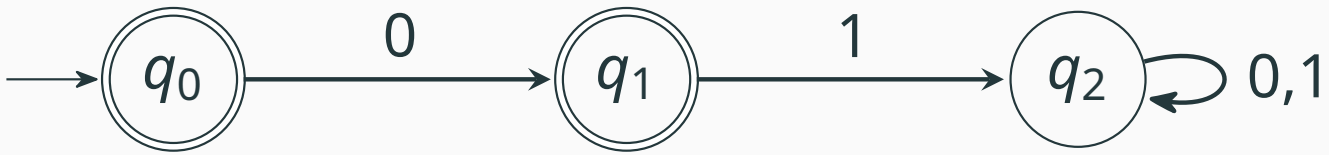
Ejercicio 5.a

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

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



¡**Invertimos** los estados finales!



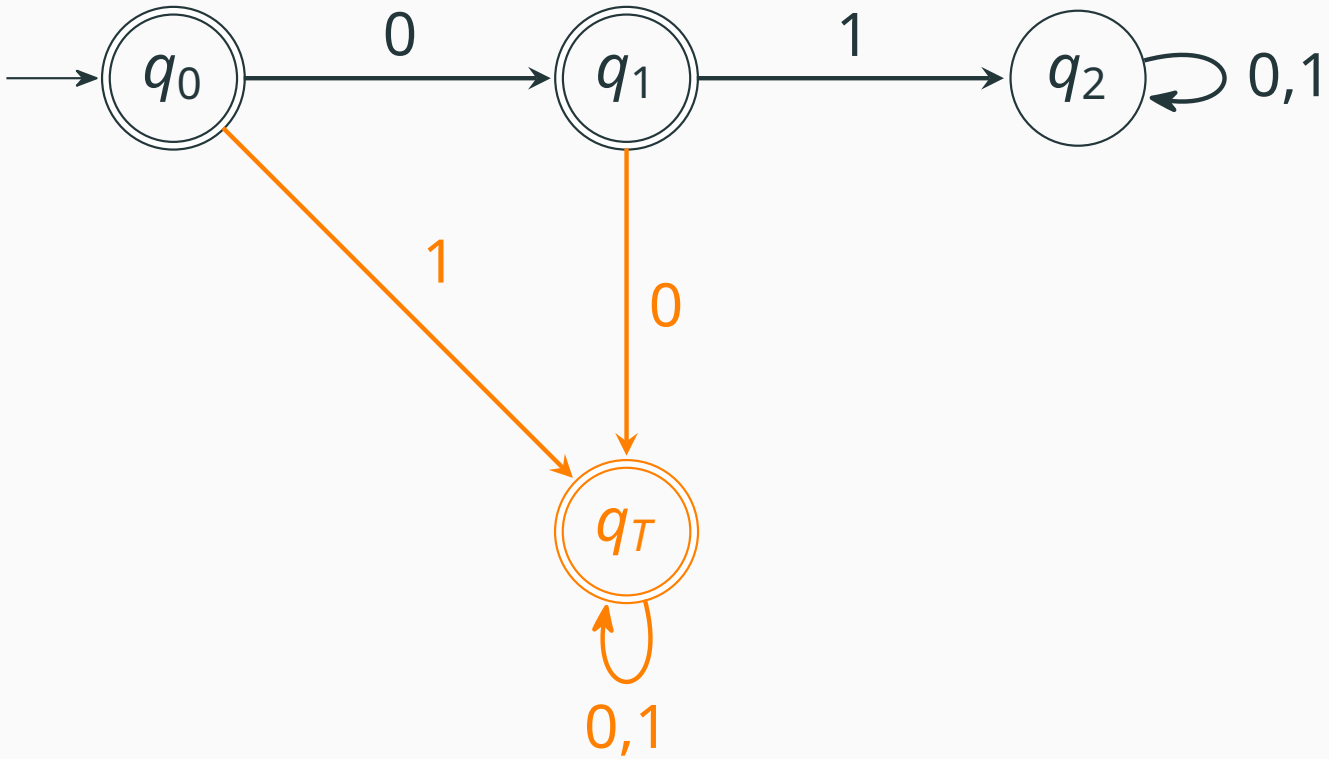
Pero $111 \notin \mathcal{L}_2$ (no arranca con 01) y, ¡no la reconoce! **El autómata tiene que estar completo**, sino perdemos cadenas.

Ejercicio 5 - Complemento

Ejercicio 5.a

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

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

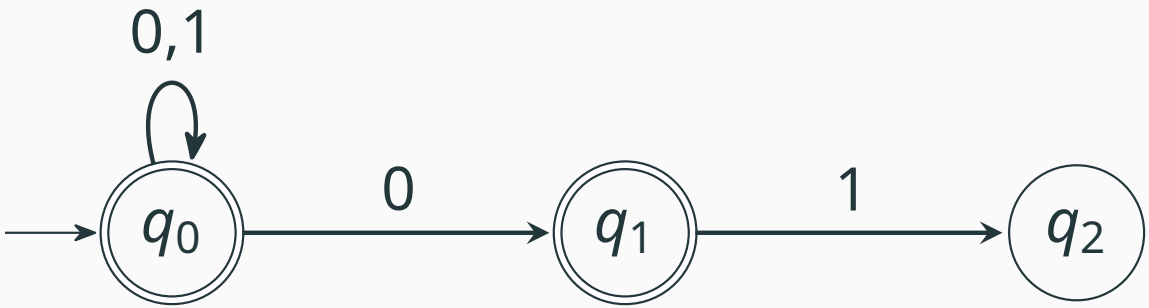


Ejercicio 5 - Complemento

Ejercicio 5.b

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

Candidato:



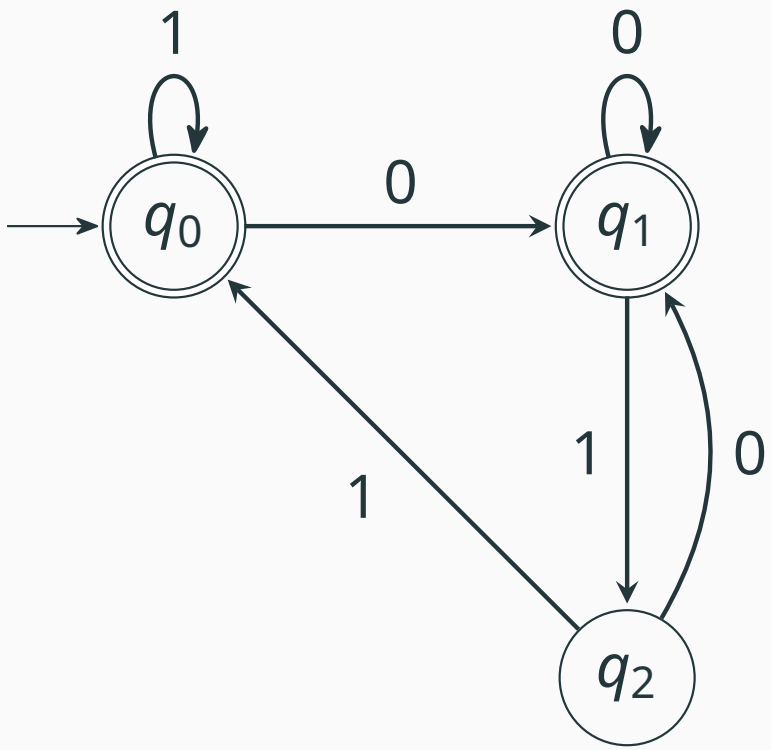
¡No funciona! Aceptamos cadenas demás como 01 (en particular en este caso aceptamos Σ^*). Para cada cadena que aceptábamos, había caminos que no aceptaban. Entonces si invertimos los finales, esos caminos pueden pasar a ser de aceptación (excepto que se traben) y aceptamos cadenas que no deberíamos.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

Ejercicio 5 - Complemento

Ejercicio 5.b

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

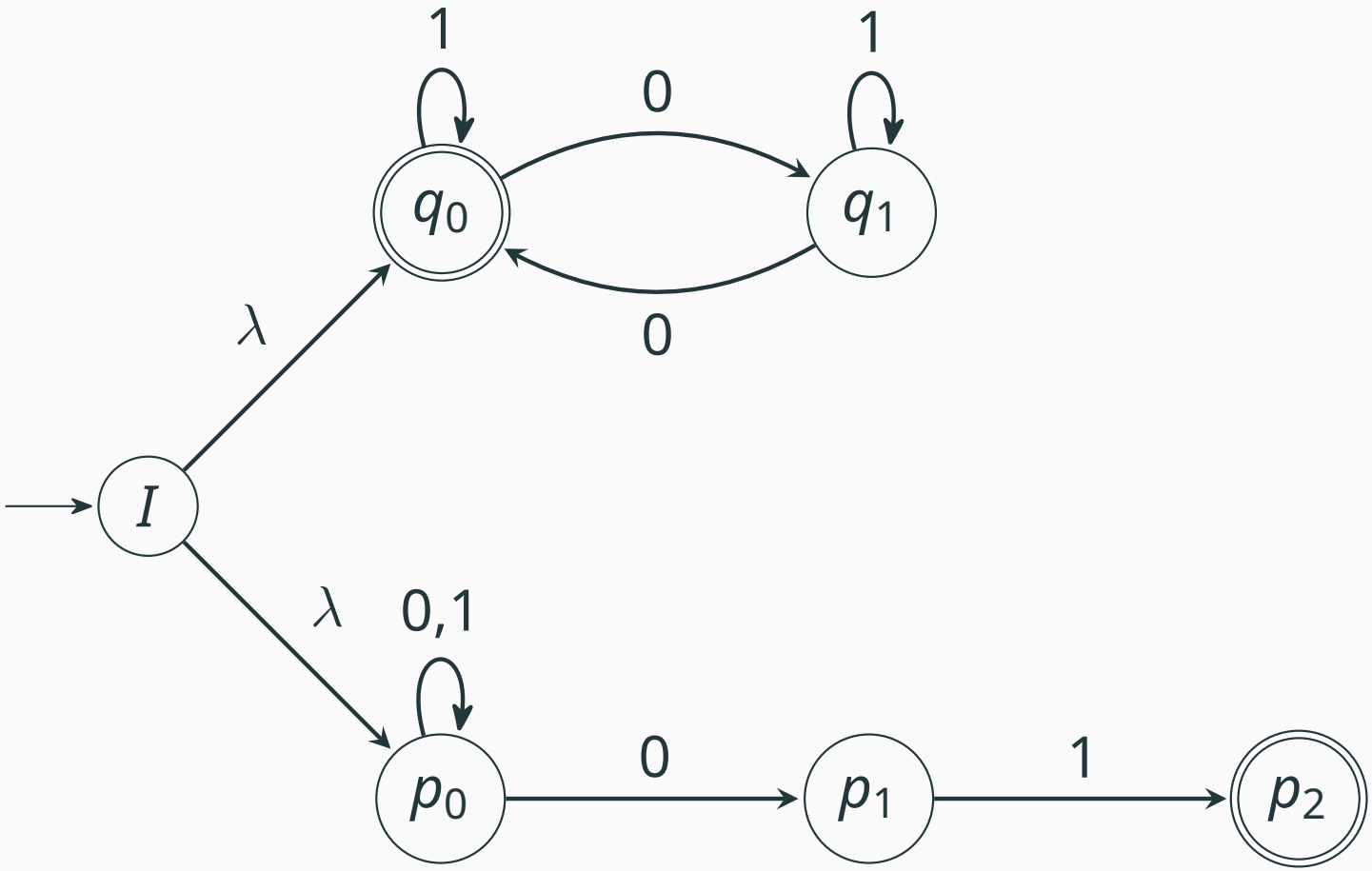
$$\mathcal{A}_3^c = \langle \{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_0, q_1\} \rangle$$



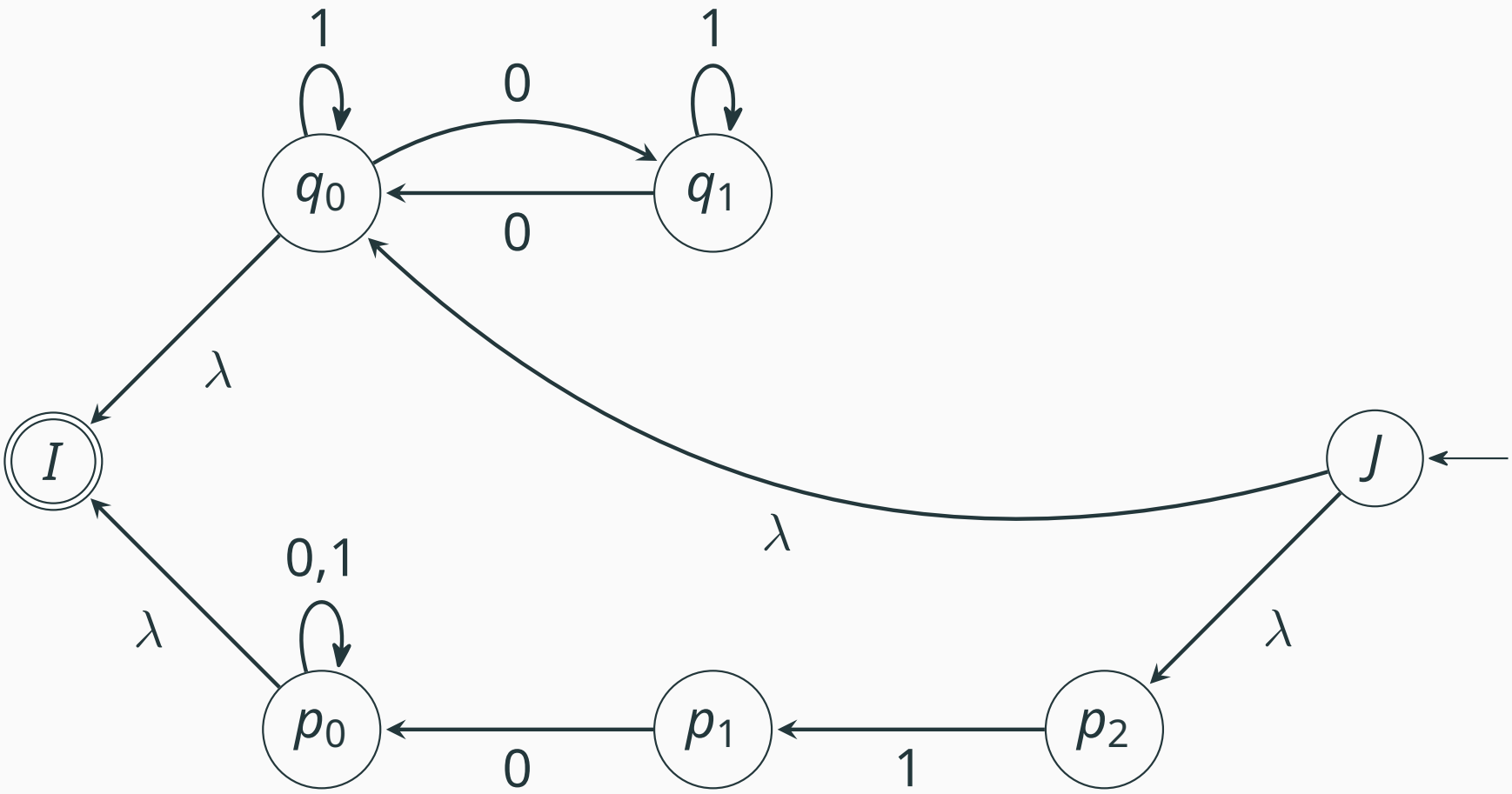
Ejercicio 6 - Reversa

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



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

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 AF**D 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 con iniciales)

Conclusiones

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