Autómatas finitos

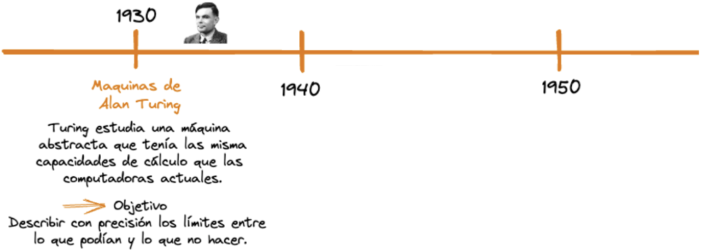
Teoría de Lenguajes

Sabrina Gisele Silvero

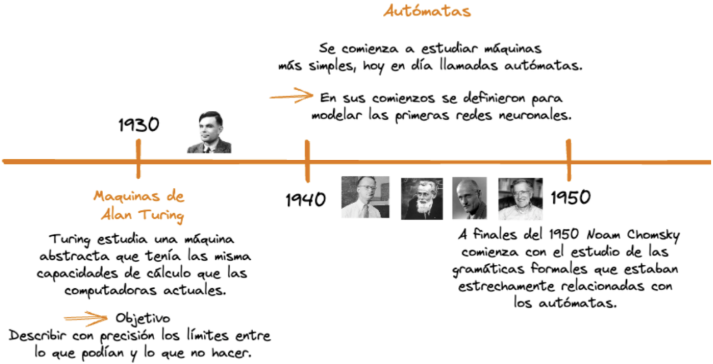
Departamento de Computación,
Facultad de Ciencias Exactas y Naturales,
Universidad de Buenos Aires

2do cuatrimestre 2023

¿Por qué estudiamos lo que estudiamos?



¿Por qué estudiamos lo que estudiamos?



Autómatas finitos

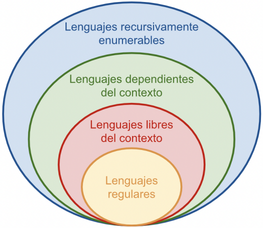
- La clase pasada vimos símbolos, alfabetos, cadenas y lenguajes

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**

Autómatas finitos

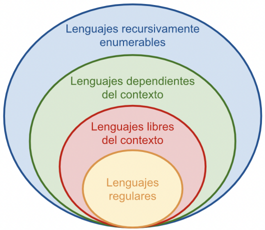
- La clase pasada vimos símbolos, alfabetos, cadenas y lenguajes
- Hoy vamos a ver una máquina abstracta que nos permite *reconocer* lenguajes: los **autómatas finitos**
- Estos reconocen exactamente una *clase* de lenguajes en particular: los **lenguajes regulares**



Jerarquía de Chomsky

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

$$L_1 = \{\alpha \mid \alpha \in \{0,1\}^* \text{ y } |\alpha|_0 \text{ es par } \}$$

Ejercicio 1

$$L_1 = \{\alpha \mid \alpha \in \{0,1\}^* \text{ y } |\alpha|_0 \text{ es par }\}$$

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

Ejemplos de cadenas:

Ejercicio 1

$$L_1 = \{\alpha \mid \alpha \in \{0, 1\}^* \text{ y } |\alpha|_0 \text{ es par } \}$$

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

Ejercicio 1

$$L_1 = \{\alpha \mid \alpha \in \{0,1\}^* \text{ y } |\alpha|_0 \text{ es par } \}$$

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 A_1 que reconoce L_1 es,

Ejercicio 1

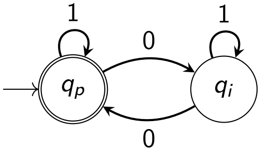
$$L_1 = \{\alpha \mid \alpha \in \{0,1\}^* \text{ y } |\alpha|_0 \text{ es par } \}$$

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 A_1 que reconoce L_1 es,



Definición de autómatas finitos

Un AFD es una tupla de la forma

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

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 A_1 es

$$A_1 = \langle \overbrace{\{q_p, q_i\}}^{Q}, \overbrace{\{0, 1\}}^{\Sigma}, \delta, \overbrace{q_p}^{\text{inicial}}, \overbrace{\{q_p\}}^{F} \rangle$$

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**,

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 escribir la tupla!

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 escribir la tupla!

Configuraciones instantáneas

¿Cómo formalizamos que un autómata **acepte** una cadena? [1]

- Vamos a definir *configuraciones instantáneas*, una tupla[2] 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.

[1] El formalismo que usamos en la práctica es distinto que el de la teórica

[2] En TLeng nos encantan las tuplas ;)

Configuraciones instantáneas

¿Cómo formalizamos que un autómata **acepte** una cadena? [1]

- Vamos a definir *configuraciones instantáneas*, una tupla[2] 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.

[1] El formalismo que usamos en la práctica es distinto que el de la teórica

[2] En TLeng nos encantan las tuplas ;)

Configuraciones instantáneas

¿Cómo formalizamos que un autómata **acepte** una cadena? [1]

- Vamos a definir *configuraciones instantáneas*, una tupla[2] 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.

[1] El formalismo que usamos en la práctica es distinto que el de la teórica

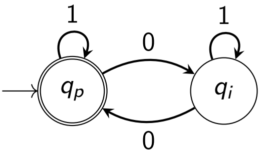
[2] En TLeng nos encantan las tuplas ;)

Configuraciones instantáneas

¿Cómo formalizamos que un autómata **acepte** una cadena? [1]

- Vamos a definir *configuraciones instantáneas*, una tupla[2] 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:



$$
\begin{aligned}
(q_p, 010) &\vdash_{A_1} (q_i, 10) \\
&\vdash_{A_1} (q_i, 0) \\
&\vdash_{A_1} (q_p, \lambda)
\end{aligned}
$$

[1] El formalismo que usamos en la práctica es distinto que el de la teórica

[2] En TLeng nos encantan las tuplas ;)

Configuraciones instantáneas

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

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

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

Relación de transición entre configuraciones

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

$$(q_i, a.\alpha) \vdash_A (q_j, \alpha) \iff \delta(q_i, a) = q_j$$

Cuando está claro cual es el autómata, podemos omitirlo y usar $\vdash$ *en lugar de* $\vdash_A$

Pertenencia al lenguaje

$$\alpha \in L(A) \iff \exists q_f \in F \mid (q_0, \alpha) \vdash_A^* (q_f, \lambda)$$

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 $\vdash^*$ es la clausura de Kleene de la relación $\vdash$. Es decir, aplicarla cero o más veces. ¿Cuando necesitamos aplicarla cero veces?

Pertenencia al lenguaje

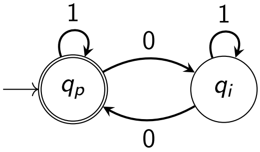
$$\alpha \in L(A) \iff \exists q_f \in F \mid (q_0, \alpha) \vdash_A^* (q_f, \lambda)$$

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 $\vdash^*$ es la clausura de Kleene de la relación $\vdash$. Es decir, aplicarla cero o más veces. ¿Cuando necesitamos aplicarla cero veces? Cuando la entrada es λ, que sería aceptada solo si el estado inicial también es final.

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

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

$$L_2 = \{01\alpha \mid \alpha \in \{0, 1\}^*\}$$

¿Qué tenemos que recordar?

Ejercicio 2

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

$$L_2 = \{01\alpha \mid \alpha \in \{0, 1\}^*\}$$

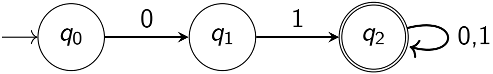
¿Qué tenemos que recordar? Si vimos un 0 y luego un 1

Ejercicio 2

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

$$L_2 = \{01\alpha \mid \alpha \in \{0, 1\}^*\}$$

¿Qué tenemos que recordar? Si vimos un 0 y luego un 1
Proponemos el siguiente autómata A_2 para L_2,



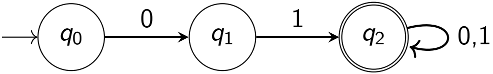
¿Qué problema tiene?

Ejercicio 2

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

$$L_2 = \{01\alpha \mid \alpha \in \{0,1\}^*\}$$

¿Qué tenemos que recordar? Si vimos un 0 y luego un 1
Proponemos el siguiente autómata A_2 para L_2,

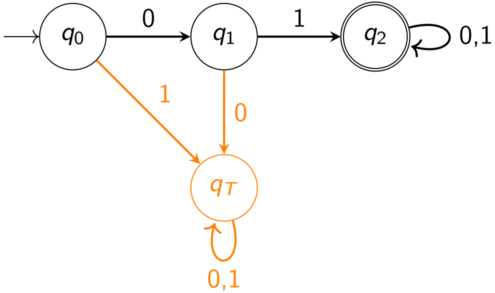


¿Qué problema tiene? ¡La función de transición está incompleta!

δ	0	1
q_0	q_1	?
q_1	?	q_2
q_2	q_2	q_2

Ejercicio 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

$$A_2 = \langle \{q_0, q_1, q_2, q_5\}, \{0,1\}, \delta, q_0, \{q_2\} \rangle$$

Ejercicio 3

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

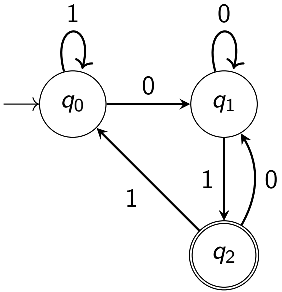
$$L_3 = \{\alpha 01 \mid \alpha \in \{0, 1\}^*\}$$

Ejercicio 3

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

$$L_3 = \{\alpha 01 \mid \alpha \in \{0, 1\}^*\}$$

A_3:



Significado intuitivo de cada estado:

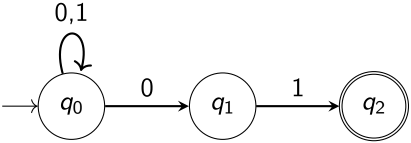
- q_0: "La cadena termina en 1^*"
- q_1: "La cadena termina en 0^+"
- q_2: "La cadena termina en 01"

Ejercicio 3 - Alternativa

El lenguaje L_3 es muy parecido a 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".

Ejercicio 3 - Alternativa

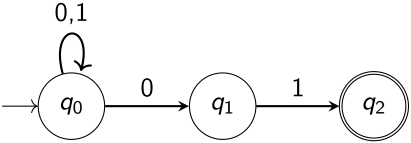
El lenguaje L_3 es muy parecido a L_2, pero el autómata se ve más complicado. Nos gustaría que el formalismo nos permita expresar algo como "Puede venir cualquier cadena, siempre y cuando termine con 01". Proponemos A_3',



Pero

Ejercicio 3 - Alternativa

El lenguaje L_3 es muy parecido a 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". Proponemos 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!**

AFND-λ

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

[3] No confundir con la cadena λ, es una notación

AFND-λ

Los autómatas que veníamos viendo hasta ahora eran **determinísticos**, en cada momento tenían una sola acción posible. 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 $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ pero cambia la función de transición:

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

- En lugar de un solo estado, devuelve un conjunto
- Además de transiciones por un símbolo de la entrada, podemos transicionar [3] por λ sin consumir ningún símbolo

[3] No confundir con la cadena λ, es una notación

AFND-λ

Los autómatas que veníamos viendo hasta ahora eran **determinísticos**, en cada momento tenían una sola acción posible. 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 $A = \langle Q, \Sigma, \delta, q_0, F \rangle$ pero cambia la función de transición:

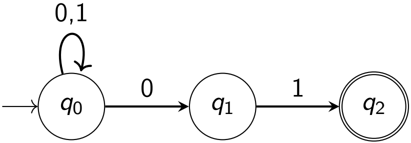
$$\delta : Q \times (\Sigma \cup \lambda) \rightarrow P(Q)$$

- En lugar de un solo estado, devuelve un conjunto
- Además de transiciones por un símbolo de la entrada, podemos transicionar [3] por λ sin consumir ningún símbolo

Diferencia con bibliografía

En la teórica y bibliografía van a ver que se diferencia entre AFNDs con y sin transiciones λ (a veces llamadas ϵ), pero para nosotros va a ser lo mismo.

[3] No confundir con la cadena λ, es una notación



La tupla que describe a A'_3 es

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

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

$$\begin{aligned}(q_i, a.\alpha) \vdash_A (q_j, \alpha) &\iff q_j \in \delta(q_i, a)\\ (q_i, \alpha) \vdash_A (q_j, \alpha) &\iff q_j \in \delta(q_i, \lambda)\end{aligned}$$

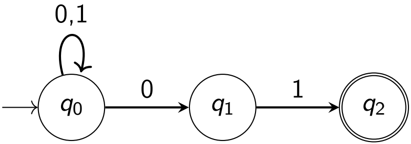
La definición del lenguaje es idéntica

Lenguaje aceptado

$$\alpha \in L(A) \iff \exists q_f \in F \mid (q_0, \alpha) \vdash_A^* (q_f, \lambda)$$

Seguimientos

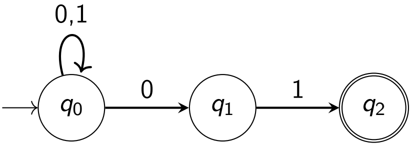
A'_3:



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

Seguimientos

A'_3:



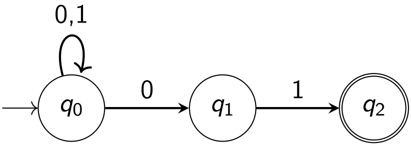
Para la cadena $\alpha = 101 \in L_3$ tenemos dos caminos posibles

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

Por la definición de lenguaje aceptado, alcanza con que *existe 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

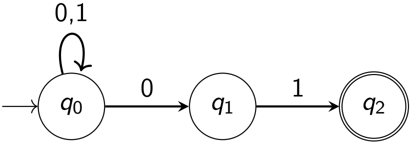
A_3':



Para la cadena $\beta = 010 \notin L_3$ tenemos tres caminos posibles

Más seguimientos

A_3':



Para la cadena $\beta = 010 \notin L_3$ tenemos tres caminos posibles

- $(q_0, 010) \vdash_{A_3'} (q_0, 10) \vdash_{A_3'} (q_0, 0) \vdash_{A_3'} (q_1, \lambda)$ ✗
  $(q_0, 010) \vdash_{A_3'} (q_0, 10) \vdash_{A_3'} (q_0, 0) \vdash_{A_3'} (q_0, \lambda)$ ✗

  Consumen toda la cadena pero no llegan a un estado final

- $(q_0, 010) \vdash_{A_3'} (q_1, 10) \vdash_{A_3'} (q_2, 0)$ ✗

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

Lenguaje aceptado

$$\alpha \in L(A) \iff \exists q_f \in F \mid (q_0, \alpha) \vdash_A^* (q_f, \lambda)$$

¡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.

¿Nos tomamos un break?

Ejercicio 4 - Unión

$L_4 = L_1 \cup L_3$. Con L_1 = cadenas que terminan en 01 y L_3 = cadenas con cantidad par de 0s. ¿Qué significa L_4?

Ejercicio 4 - Unión

$L_4 = L_1 \cup L_3$. Con $L_1 =$ cadenas que terminan en 01 y $L_3 =$ cadenas con cantidad par de 0s. ¿Qué significa L_4? Cadenas que terminen en 01 **o** tengan cantidad par de 0s.

Pista:

Ejercicio 4 - Unión

$L_4 = L_1 \cup L_3$. Con $L_1 =$ cadenas que terminan en 01 y $L_3 =$ cadenas con cantidad par de 0s. ¿Qué significa L_4? Cadenas que terminen en 01 **o** tengan cantidad par de 0s.

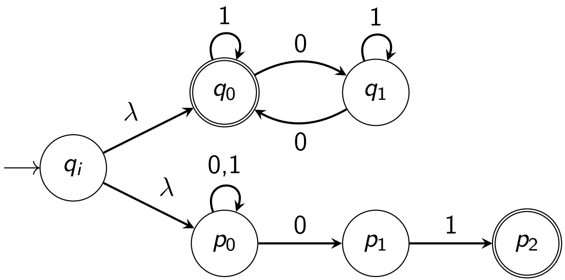
Pista: ¿se les ocurre alguna forma de resolverlo usando A_1 y A'_3?

Ejercicio 4 - Unión

$L_4 = L_1 \cup L_3$. Con L_1 = cadenas que terminan en 01 y L_3 = cadenas con cantidad par de 0s. ¿Qué significa L_4? Cadenas que terminen en 01 **o** tengan cantidad par de 0s.

Pista: ¿se les ocurre alguna forma de resolverlo usando A_1 y A_3'?

$$A_4 = \langle \{q_i, q_0, q_1, p_0, p_1, p_2\}, \{0, 1\}, \delta, q_i, \{q_0, p_2\} \rangle$$



Ejercicio 5 - Complemento

a. Con $L_2 =$ cadenas que comienzan por 01,
   $L_2^c =$

Ejercicio 5 - Complemento

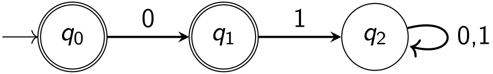
a. Con $L_2 =$ cadenas que comienzan por 01,
   $L_2^c =$ cadenas que no comienzan por 01

Pista: ¿Cómo podemos usar A_2?

Ejercicio 5 - Complemento

a. Con $L_2 =$ cadenas que comienzan por 01,
$L_2^c =$ cadenas que no comienzan por 01

Pista: ¿Cómo podemos usar A_2? Invertimos los estados finales.
Candidato con el *trampa implícito*:



Convención del trampa implícito

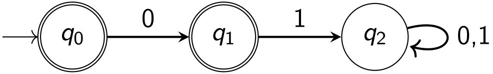
Vamos a tomar el estado trampa como implícito cuando un autómata sea determinístico (no haya más de una opción) pero tenga δ indefinida para algunas transiciones. Pero para esta transformación, no hay que olvidar agregarlo.

Pero

Ejercicio 5 - Complemento

a. Con $L_2 =$ cadenas que comienzan por 01,
$L_2^c =$ cadenas que no comienzan por 01

Pista: ¿Cómo podemos usar A_2? Invertimos los estados finales.
Candidato con el *trampa implícito*:



Convención del trampa implícito

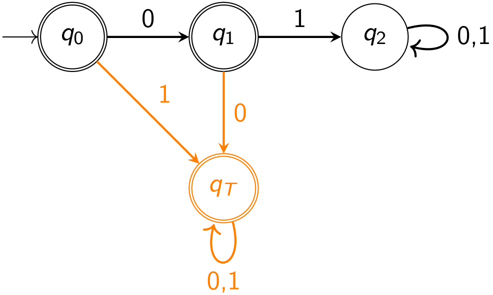
Vamos a tomar el estado trampa como implícito cuando un autómata sea determinístico (no haya más de una opción) pero tenga δ indefinida para algunas transiciones. Pero para esta transformación, no hay que olvidar agregarlo.

Pero $111 \notin L_2$ (no arranca con 01) y, ¡no la reconoce! **El autómata tiene que estar completo**, sino perdemos cadenas.

Ejercicio 5 - Complemento

a. L_2^c = cadenas que no comienzan por 01

$$A_2^c = \langle \{q_0, q_1, q_2, q_T\}, \{0,1\}, \delta, q_0, \{q_0, q_1, q_T\} \rangle$$



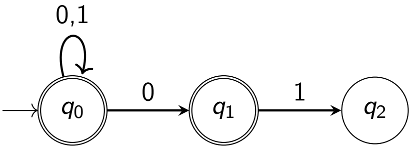
Ejercicio 5 - Complemento

a. L_3^c = cadenas que no terminan con 01

Ejercicio 5 - Complemento

a. L_3^c = cadenas que no terminan con 01

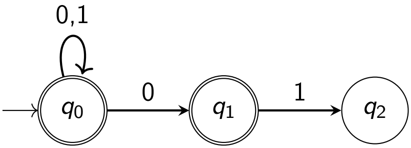
Candidato:



Ejercicio 5 - Complemento

a. $L_3^c =$ 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. Sino aceptamos cadenas demás.

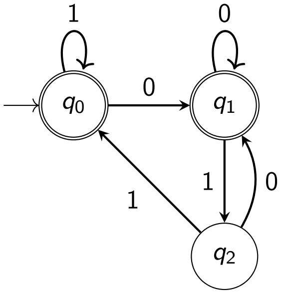
Ejercicio 5 - Complemento

b. $L_3^c =$ cadenas que no terminan con 01

Ejercicio 5 - Complemento

b. L_3^c = cadenas que no terminan con 01

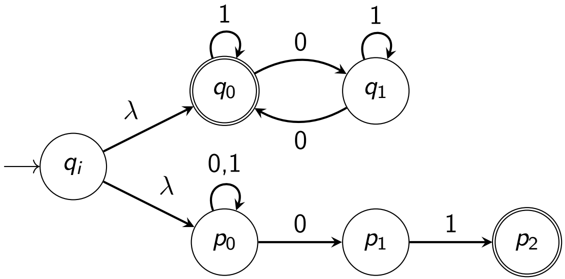
$$A_3^c = \langle \{q_0, q_1, q_2\}, \{0, 1\}, \delta, q_0, \{q_0, q_1\} \rangle$$



Ejercicio 6 - Reversa

L_4^r, con $L_4 = L_1 \cup L_3$, cadenas que terminan en 01 o tienen cantidad par de 0s

$$A_4 = \langle \{q_i, q_0, q_1, p_0, p_1, p_2\}, \{0, 1\}, \delta, q_i, \{q_0, p_2\} \rangle$$

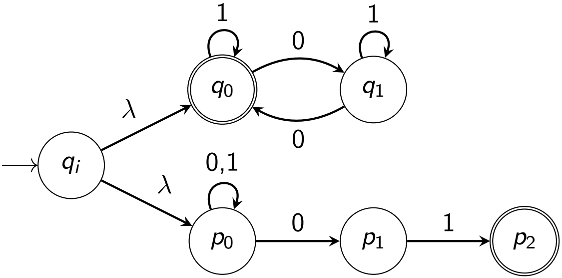


Solución:

Ejercicio 6 - Reversa

L_4^r, con $L_4 = L_1 \cup L_3$, cadenas que terminan en 01 o tienen cantidad par de 0s

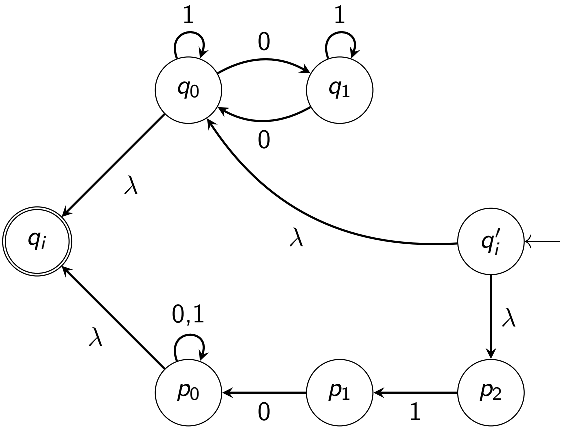
$$A_4 = \langle \{q_i, q_0, q_1, p_0, p_1, p_2\}, \{0,1\}, \delta, q_i, \{q_0, p_2\} \rangle$$



Solución: Ejecutamos el autómata al revés: damos vuelta las flechas e invertimos los finales e iniciales. Como no podemos tener más de un inicial, agregamos uno con transiciones λ

Ejercicio 6 - Reversa

$$A_4^r = \langle \{q_i, q_i', q_0, q_1, p_0, p_1, p_2\}, \{0, 1\}, \delta, q_i', \{q_i\} \rangle$$



Operaciones vistas

Unión

Dados A_1 y A_2 AFs, para obtener $L(A_1) \cup L(A_2)$ agregar un nuevo estado inicial con transiciones λ a los iniciales de A_1 y A_2.

Complemento

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

Reversa

Dado un AFND-λ, obtener $A' = \langle Q', \Sigma, \delta', q_0', F' \rangle$ 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 1 :D

¿Preguntas?