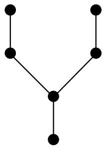
Introducción a la Lógica y la Computación

Mariana Badano   Héctor Gramaglia
Pedro Sánchez Terraf   Mauricio Tellechea
Guido Ivetta

FaMAF, 13 de octubre de 2021

Ejes de Contenidos

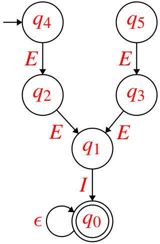
1. Estructuras Ordenadas



2. Lógica Proposicional

$$\dfrac{\dfrac{\dfrac{[\varphi \wedge \psi]_1}{\psi}\wedge E \qquad \dfrac{[\varphi \wedge \psi]_1}{\varphi}\wedge E}{\psi \wedge \varphi}\wedge I}{\varphi \wedge \psi \to \psi \wedge \varphi}\to I_1$$

3. Lenguajes y Autómatas



Parte 3: Lenguajes y Autómatas

Bibliografía

- Alejandro Tiraboschi y otros, *Introducción a la Lógica. Parte III: Lenguajes y Autómatas*.
- J. Hopcroft, R. Motwani y J. Ullman, *Introduction to Automata Theory, Languages and Computation* (2001; hay versión en castellano).

Contenidos estimados para hoy

1. Lenguajes y Autómatas
   - Ejemplo de autómata
   - Definición formal de lenguaje
   - Definición formal de autómata finito
   - Lenguajes aceptados y regulares
   - Caracterizando lenguajes aceptados
   - Autómatas finitos no deterministas

Autómatas

Los autómatas son un modelo matemático de dispositivos que computan.

Autómatas

Los autómatas son un modelo matemático de dispositivos que computan.
En una primera aproximación teórica, podemos clasificarlos de acuerdo a qué tipo de memoria tienen.

Autómatas

Los autómatas son un modelo matemático de dispositivos que computan.
En una primera aproximación teórica, podemos clasificarlos de acuerdo a qué tipo de memoria tienen.

Tipo de memoria

- Memoria sólo para el programa (finita, “registros”).
- Memoria tipo pila.
- Memoria tipo RAM.

Autómatas

Los autómatas son un modelo matemático de dispositivos que computan.
En una primera aproximación teórica, podemos clasificarlos de acuerdo a qué tipo de memoria tienen.

Tipo de memoria

- Memoria sólo para el programa (finita, “registros”).
- Memoria tipo pila.
- Memoria tipo RAM.

En los dos últimos casos, se supone que no hay límites de recursos (pero se usan sólo finitos para cada computación).

Lenguajes

Un **lenguaje** es una familia de strings (“cadenas”, “palabras”).

Lenguajes

Un **lenguaje** es una familia de strings (“cadenas”, “palabras”).

¿Por qué hablamos de lenguajes?

Un problema puede *codificarse* como un lenguaje.

Lenguajes

Un **lenguaje** es una familia de strings ("cadenas", "palabras").

¿Por qué hablamos de lenguajes?

Un problema puede *codificarse* como un lenguaje.
Ejemplo: Sea L_p el conjunto de sucesiones de números binarios que representan primos ($10 \in L_p, 101 \in L_p, 100 \notin L_p$).

Lenguajes

Un **lenguaje** es una familia de strings (“cadenas”, “palabras”).

¿Por qué hablamos de lenguajes?

Un problema puede *codificarse* como un lenguaje.
Ejemplo: Sea L_p el conjunto de sucesiones de números binarios que representan primos $(10 \in L_p, 101 \in L_p, 100 \notin L_p)$.
Decidir si n es primo se reduce a comprobar si su representación binaria está o no en L_p.

Ejemplo

Computar $f(x) := x^3$ usando un lenguaje.

$f : Nat \to Nat.$

Alfabeto $= \Sigma := \{0, 1, \ldots, 9, \#\}$

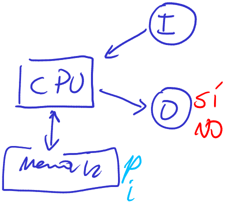
$L_3 = \{0\#0, 1\#1, 2\#8, 3\#27, 4\#64, \ldots\}$
$= \{\overline{n}\#\overline{n^3} : n \in Nat\}$

$n \in Nat$ $\qquad \overline{n}$ = expresión decimal de n $\in$ String.

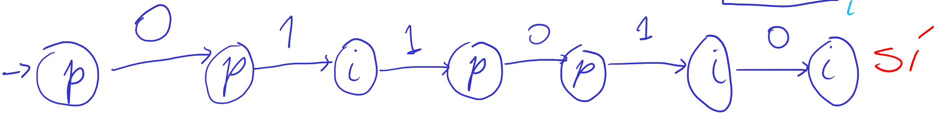
Autómatas finitos

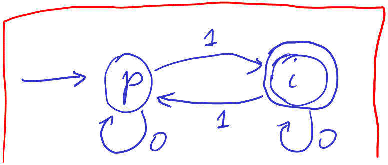
Decidir si una cadena binaria ('0' y '1') tiene una cantidad impar de 1s.

0110.-



011010





Ejemplo de autómata

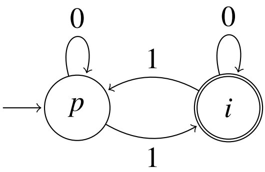
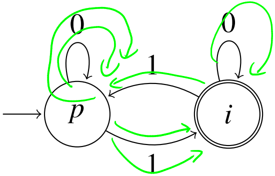


Diagrama de transición

Estados	Los "nodos" en el diagrama.
Transiciones	Son las flechas, tienen un *símbolo* como etiqueta.
Estado inicial	Se indica por la flecha que no tiene origen.
Estados finales	Se indican por un doble círculo.
Símbolos	Inputs que conoce el autómata.

Ejemplo de autómata



011010

Traza o ejecución

Dada una cadena, es la sucesión de estados por los que transita el autómata mientras la recorre, empezando con el estado inicial.

Aceptación

Una palabra es aceptada si su traza concluye en algún estado final.

Definición formal de lenguaje

Alfabeto Conjunto finito no vacío, habitualmente usamos Σ para referirnos a alfabetos. En el ejemplo anterior $\Sigma = \{0, 1\}$.

Definición formal de lenguaje

Alfabeto Conjunto finito no vacío, habitualmente usamos Σ para referirnos a alfabetos. En el ejemplo anterior $\Sigma = \{0, 1\}$.

Cadena Sucesión finita de símbolos de Σ.
A la sucesión de longitud 0 la denotamos con ϵ.
Dado una cadena α y un símbolo $x\alpha$ es la cadena que tiene como primer símbolo x y luego α.

Confundiremos x :: símbolo con (x) :: cadena.

$$x\alpha \to x \triangleright \alpha$$

$$x\alpha \to (x) ++ \alpha$$

Definición formal de lenguaje

Alfabeto Conjunto finito no vacío, habitualmente usamos Σ para referirnos a alfabetos. En el ejemplo anterior $\Sigma = \{0, 1\}$.

Cadena Sucesión finita de símbolos de Σ.
A la sucesión de longitud 0 la denotamos con ϵ.
Dado una cadena α y un símbolo $x\alpha$ es la cadena que tiene como primer símbolo x y luego α.

Potencias de Σ Definimos por recursión en $k \in \mathbb{N}$.

$$\Sigma^0 = \{\epsilon\}$$
$$\Sigma^{k+1} = \{x\alpha \mid \alpha \in \Sigma^k, x \in \Sigma\}$$
$$\Sigma^* = \bigcup_{n\in\mathbb{N}} \Sigma^n$$

Si $\Sigma \neq \varnothing$, Σ^* es infinito.

$$\alpha \in \Sigma^k \iff |\alpha| = k.$$

Lenguaje

Un **lenguaje** es un subconjunto de Σ^*.

Lenguaje

Un **lenguaje** es un subconjunto de Σ^*.

Ejemplos

L_p sucesiones binarias que representan números primos.

L_{py} cadenas ASCII que son programas de Python que compilan.

L_I cadenas con una cantidad impar de unos.

$L_=$ cadenas con igual cantidad de ceros que de unos.

Lenguaje

Un **lenguaje** es un subconjunto de Σ^*.

Ejemplos

L_p sucesiones binarias que representan números primos.

L_{py} cadenas ASCII que son programas de Python que compilan.

L_I cadenas con una cantidad impar de unos.

$L_=$ cadenas con igual cantidad de ceros que de unos.

Casos especiales

$\emptyset$ el lenguaje vacío.

$\{\epsilon\}$ el lenguaje "unidad".

ϵ :: String $\neq \emptyset$ :: set

Σ considerado a los símbolos como cadenas.

Σ^* todas las palabras del alfabeto Σ.

Más vocabulario

Concatenación pegar dos palabras del mismo alfabeto. Formalmente por recursión en una de las palabras:

$$\epsilon\beta := \beta$$
$$(x\alpha)\beta := x(\alpha\beta)$$

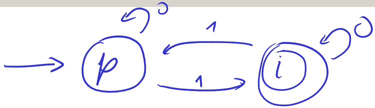
segmento

Subcadena α es *subcadena* de β si existen γ y γ' tales que $\beta = \gamma\alpha\gamma'$.

Prefijo α es *prefijo* de β si $\beta = \alpha\gamma'$.

Sufijo α es *sufijo* de β si $\beta = \gamma\alpha$.

Autómatas finitos deterministas (DFA)



Un **autómata finito determinista** $\mathbb{A}$ consta de los siguientes componentes:

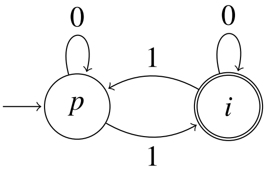
1. Un conjunto finito de estados Q.
2. Un alfabeto Σ.
3. Una función de transición (que codifica las aristas de la representación gráfica) $\delta : Q \times \Sigma \to Q$.
4. Un estado inicial $q_0 \in Q$.
5. Un conjunto de estados finales $F \subseteq Q$.

Usaremos la notación $\mathbb{A} = (Q, \Sigma, \delta, q_0, F)$ para indicar los componentes del autómata $\mathbb{A}$.

Formalizando el autómata anterior

$\delta(p, 0) = p.$

$p \xrightarrow{0} p$



$Q = \{p, i\}$

$\Sigma = \{0, 1\}$

$q_0 = p$

$F = \{i\}$

δ	0	1
p	p	i
i	i	p

Lenguaje de un autómata y lenguajes regulares

La función de transición δ puede ser extendida a $\hat{\delta}\colon Q \times \Sigma^* \to Q$ para que acepte palabras:

$$\hat{\delta}(q, \epsilon) := q$$

$$\hat{\delta}(q, x\alpha) := \hat{\delta}(\underline{\delta(q, x)}, \alpha)$$

$$q \xRightarrow{\varepsilon} q$$

$$q \xRightarrow{x\alpha} q'$$

sii

$$\exists r : q \xrightarrow{x} r \xRightarrow{\alpha} q'$$

Lenguaje de un autómata y lenguajes regulares

La función de transición δ puede ser extendida a $\hat{\delta}\colon Q \times \Sigma^* \to Q$ para que acepte palabras:

$$\hat{\delta}(q,\epsilon) := q \qquad\qquad \hat{\delta}(q,\epsilon) := q$$

$$\hat{\delta}(q,x\alpha) := \hat{\delta}(\delta(q,x),\alpha) \qquad\qquad \hat{\delta}(q,\beta x) := \delta(\hat{\delta}(q,\beta),x)$$

$$q \xRightarrow{\beta x} q'$$

sii

$$\exists r: \quad q \xRightarrow{\beta} r \xrightarrow{x} q'$$

Lenguaje de un autómata y lenguajes regulares

La función de transición δ puede ser extendida a $\hat{\delta}\colon Q \times \Sigma^* \to Q$ para que acepte palabras:

$$\hat{\delta}(q, \epsilon) := q \qquad \hat{\delta}(q, \epsilon) := q$$

$$\hat{\delta}(q, x\alpha) := \hat{\delta}(\delta(q, x), \alpha) \qquad \hat{\delta}(q, \beta x) := \delta(\hat{\delta}(q, \beta), x)$$

El **lenguaje del autómata** $\mathbb{A}$ son las palabras que comenzando en el estado inicial terminan (según $\hat{\delta}$) en uno final:

$$L_{\mathbb{A}} := \{\alpha \in \Sigma^* \mid \hat{\delta}(q_0, \alpha) \in F\}$$

$$= \{\alpha \in \Sigma^* \mid \exists r \in F : p_0 \xRightarrow{\alpha} r\}$$

Lenguaje de un autómata y lenguajes regulares

La función de transición δ puede ser extendida a $\hat{\delta}\colon Q \times \Sigma^* \to Q$ para que acepte palabras:

$$\hat{\delta}(q,\epsilon) := q \qquad\qquad \hat{\delta}(q,\epsilon) := q$$

$$\hat{\delta}(q,x\alpha) := \hat{\delta}(\delta(q,x),\alpha) \qquad\qquad \hat{\delta}(q,\beta x) := \delta(\hat{\delta}(q,\beta),x)$$

El **lenguaje del autómata** $\mathbb{A}$ son las palabras que comenzando en el estado inicial terminan (según $\hat{\delta}$) en uno final:

$$L_{\mathbb{A}} := \{\alpha \in \Sigma^* \mid \hat{\delta}(q_0,\alpha) \in F\}$$

Dado un lenguaje cualquiera L, diremos que L es **regular** si existe un autómata finito determinista $\mathbb{A}$ tal que $L = L_{\mathbb{A}}$.

Cómo caracterizar $L_{\mathbb{A}}$

Definición (Lenguaje de un estado)

$L_q := \{\alpha \in \Sigma^* \mid \hat{\delta}(q_0, \alpha) = q\}$ $= \{\alpha \in \Sigma^* : q_0 \xRightarrow{\alpha} q\}$

Probamos $\alpha \in L_q$ si y sólo si α cumple cierta propiedad, que está determinada por la información que representa cada estado.

Cómo caracterizar $L_{\mathbb{A}}$

Definición (Lenguaje de un estado)

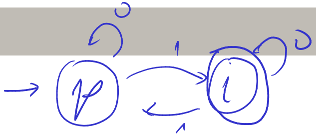
$$L_q := \{\alpha \in \Sigma^* \mid \hat{\delta}(q_0, \alpha) = q\}$$

Probamos $\alpha \in L_q$ si y sólo si α cumple cierta propiedad, que está determinada por la información que representa cada estado.

Probamos simultáneamente todos los casos por inducción en el largo de la palabra.

Cómo caracterizar $L_{\mathbb{A}}$

Definición (Lenguaje de un estado)



$$L_q := \{\alpha \in \Sigma^* \mid \hat{\delta}(q_0, \alpha) = q\}$$

Probamos $\alpha \in L_q$ si y sólo si α cumple cierta propiedad, que está determinada por la información que representa cada estado.

Probamos simultáneamente todos los casos por inducción en el largo de la palabra.

Volviendo al ejemplo

$\alpha \in L_p \iff \alpha$ tiene una cantidad par de unos. $|\alpha|_1$ es par

$\alpha \in L_i \iff \alpha$ tiene una cantidad impar de unos. $|\alpha|_1$ es impar.

Cómo caracterizar $L_{\mathbb{A}}$

Definición (Lenguaje de un estado)

$L_q := \{\alpha \in \Sigma^* \mid \hat{\delta}(q_0, \alpha) = q\}$

Probamos $\alpha \in L_q$ si y sólo si α cumple cierta propiedad, que está determinada por la información que representa cada estado.

Probamos simultáneamente todos los casos por inducción en el largo de la palabra.

Volviendo al ejemplo

$\alpha \in L_p \iff \alpha$ tiene una cantidad par de unos.

$\alpha \in L_i \iff \alpha$ tiene una cantidad impar de unos.

no es suficiente ver sólo sólo

La caracterización del lenguaje del autómata se deduce de la caracterización de los estados finales.

HI
- $\alpha \in L_p \iff \alpha$ tiene una cantidad par de unos $\iff |\alpha|_1$ es par.
- $\alpha \in L_i \iff \alpha$ tiene una cantidad impar de unos $\iff |\alpha|_1$ es impar.

Prueba: por inducción en α. $|\varepsilon|_1 = 0$

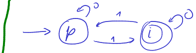
$\boxed{\alpha = \varepsilon}$:

$\varepsilon \underset{T}{\in} L_p \iff |\varepsilon|_1 \underset{T}{\text{es}}$ par

$\varepsilon \in L_i \iff |\varepsilon|_1$ es impar.

F F

✓



$\boxed{\beta = \alpha x}$ Casos: $x=0$ ó $x=1$ $\quad q_0 \xRightarrow{\alpha x} p$

$\boxed{x=0}$: $\beta \in L_p \iff \alpha 0 \in L_p. \iff \hat{\delta}(q_0, \alpha 0) = p$

$\hat{\delta}(q_0, \alpha 0) = \delta(\hat{\delta}(q_0, \alpha), 0) = p. \iff \hat{\delta}(q_0, \alpha) = p$

$\iff \alpha \in L_p \iff |\alpha|_1$ es par $\iff |\alpha 0|_1$ es par.

$\Updownarrow$

$|\beta|_1$ es par.

Falta el 2º "sii", y el caso $\boxed{x=1}$.

Equivalencia de Autómatas

Definición

Dos autómatas $\mathbb{A}, \mathbb{A}'$ son **equivalentes** si $L_{\mathbb{A}} = L_{\mathbb{A}'}$.

Equivalencia de Autómatas

Definición

Dos autómatas $\mathbb{A}, \mathbb{A}'$ son **equivalentes** si $L_{\mathbb{A}} = L_{\mathbb{A}'}$.

DFA

A continuación veremos otra clase de autómatas tales que todo ~~AFD~~ es equivalente a uno de nuestra clase (y viceversa).

UNC Universidad Nacional de Córdoba 1613 - 2013 400 AÑOS

Autómatas finitos no deterministas (NFA)

Los autómatas no deterministas son más permisivos que los DFA: permite que para estado haya cero, una o más transiciones por cada símbolo.

Autómatas finitos no deterministas (NFA)

Los autómatas no deterministas son más permisivos que los DFA: permite que para estado haya cero, una o más transiciones por cada símbolo.

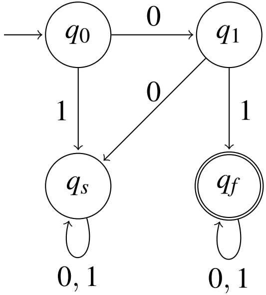
Dada una palabra, puede haber más de un camino desde el estado inicial. Por ello, cambiamos la condición de aceptación.

Autómatas finitos no deterministas (NFA)

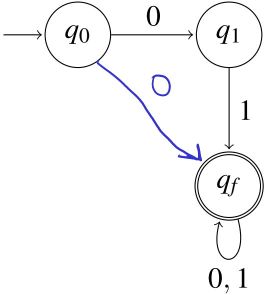
Los autómatas no deterministas son más permisivos que los DFA: permite que para estado haya cero, una o más transiciones por cada símbolo.
Dada una palabra, puede haber más de un camino desde el estado inicial. Por ello, cambiamos la condición de aceptación.
Una palabra es aceptada por un NFA si *existe un camino* que la consume y termina en un estado final.

Ejemplo: palabras que empiezan con 01

Determinista



No determinista



Formalizando NFA

Un **autómata finito no determinista** $\mathbb{A}$ consta de los siguientes componentes:

1. Un conjunto finito de estados Q.
2. Un alfabeto Σ.
3. Una función de transición $\delta \colon Q \times \Sigma \to \mathscr{P}(Q)$.
4. Un estado inicial $q_0 \in Q$.
5. Un conjunto de estados finales $F \subseteq Q$.

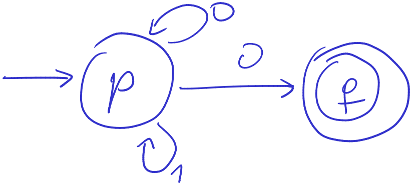
Formalizando NFA

Un **autómata finito no determinista** $\mathbb{A}$ consta de los siguientes componentes:

1. Un conjunto finito de estados Q.
2. Un alfabeto Σ.
3. Una función de transición $\delta\colon Q \times \Sigma \to \mathscr{P}(Q)$.
4. Un estado inicial $q_0 \in Q$.
5. Un conjunto de estados finales $F \subseteq Q$.

El único cambio está en la función de transición.

Ejemplo de NFA (1/2)



$\Sigma = \{0, 1\}$

$$\delta : Q \times \Sigma \longrightarrow \mathcal{P}(Q)$$

δ	0	1
p	$\{p, f\}$	$\{p\}$
f	$\emptyset$	$\emptyset$

Aceptación para NFA

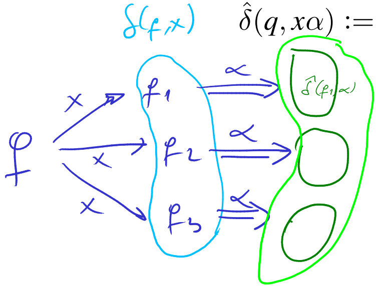
Ahora $\delta(p,x)$ es un conjunto: no se puede usar la composición de funciones directamente.

$$\delta(Q , \Sigma) \in \mathcal{P}(Q)$$

$$\hat{\delta} : Q \times \Sigma^* \to \mathcal{P}(Q)$$

$$\hat{\delta}(q,\epsilon) := \{q\}$$

$$\hat{\delta}(q,x\alpha) := \bigcup_{q' \in \delta(q,x)} \underline{\hat{\delta}(q',\alpha)}$$



$$\hat{\delta}(p, x\alpha)$$

$$p \overset{x\alpha}{\Longrightarrow} p' \quad \text{sii} \quad \exists r : p \xrightarrow{x} r \overset{\alpha}{\Longrightarrow} p'$$

Aceptación para NFA

Ahora $\delta(p,x)$ es un conjunto: no se puede usar la composición de funciones directamente.

$$\hat{\delta}\colon Q \times \Sigma^* \to \mathscr{P}(Q)$$

$$\hat{\delta}(q,\epsilon) := \{q\}$$

$$\hat{\delta}(q,x\alpha) := \bigcup_{q' \in \delta(q,x)} \hat{\delta}(q',\alpha)$$

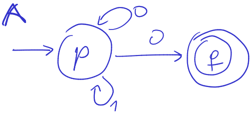
Definición

El **lenguaje del autómata** $\mathbb{A}$ son las palabras que comenzando en el estado inicial terminan (según $\hat{\delta}$) en uno final:

$$L_{\mathbb{A}} := \{\alpha \in \Sigma^* \mid \hat{\delta}(q_0,\alpha) \cap F \neq \emptyset\}.$$

$$L_A := \{\alpha \in \Sigma^* \mid \exists r \in F : p_0 \xRightarrow{\alpha} r\}$$

Ejemplo de NFA (2/2)



$F = \{q\}$ $\Sigma = \{0, 1\}$

$q_0 = p.$

$L_A = \{\alpha \in \Sigma^* \mid \exists\, r \in F : p \overset{\alpha}{\Longrightarrow} r\}$

$= \{\alpha \in \{0,1\}^* \mid p \overset{\alpha}{\Longrightarrow} q\}$

$(0) = 0\varepsilon$

$p \overset{\varepsilon}{\Longrightarrow} p$ $\varepsilon \notin L_A$

$p \overset{(0)}{\Longrightarrow} q'$ sii $\exists\, r : p \xrightarrow{0} r \overset{\varepsilon}{\Longrightarrow} q'$

$p \xrightarrow{0} q' \Longrightarrow q'$

$(0) \in L_A. \iff q' \in \{p, q\} \Longleftarrow$