Introducción a la Lógica y la Computación

Mariana Badano Héctor Gramaglia Pedro Sánchez Terraf
M. Clara Gorín Matías Steinberg

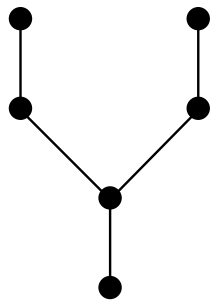
(basadas en filminas de Miguel Pagano)

FaMAF, 30 de octubre de 2020

UNC Universidad Nacional de Córdoba
1613 - 2013 400 AÑOS

Ejes de Contenidos

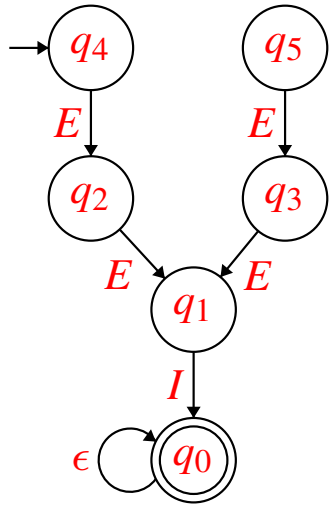
1. Estructuras Ordenadas



2. Lógica Proposicional

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

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

- Sólo memoria para el “programa”.
- 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

- Sólo memoria para el “programa”.
- 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 ($01 \in L_p, 101 \in L_p, 100 \notin L_p$).

Lenguajes

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

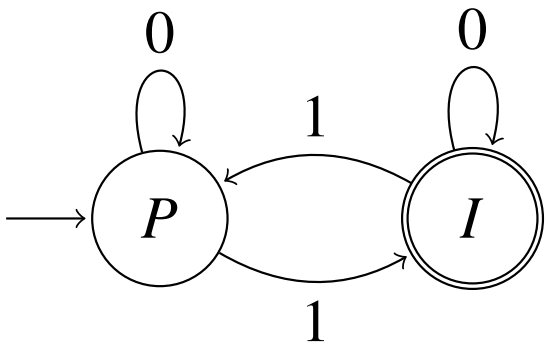
¿Por qué hablamos de lenguajes?

Un problema puede *codificarse* como un lenguaje.
Ejemplo: Sea L_p el conjunto de sucesiones de números binarios que representan primos ($01 \in L_p, 101 \in L_p, 100 \notin L_p$).
Decidir si n es primo se reduce a comprobar si su representación binaria está o no en L_p.

Autómatas finitos

Discutamos en la pizarra.

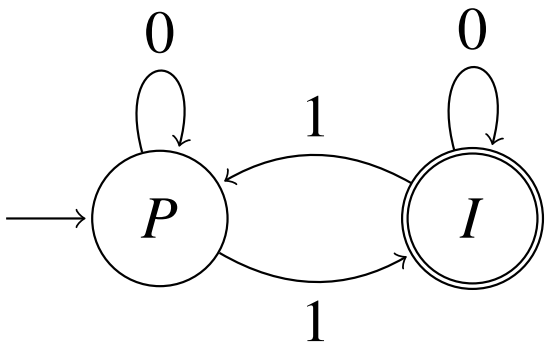
Ejemplo de autómata



Vocabulario

Estados	Son los nodos del grafo.
Transiciones	Son las aristas, 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	Acciones que conoce el autómata.

Ejemplo de autómata



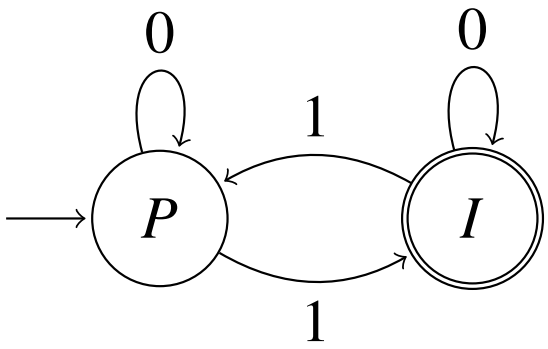
Estados

Cada estado representa cierta información *relevante* del cómputo: con el estado P representamos que hemos recibido una cantidad par de unos; mientras que I indica un número impar de unos.

Etiquetas

Señalan las acciones que pueden producir un cambio de estado. Por ejemplo, que llegue un 0 no altera la paridad de unos que recibimos.

Ejemplo de autómata



Ejecución

Una sucesión de acciones (es decir una sucesión de símbolos) produce una sucesión de cambios de estado.

Aceptación

Una palabra es aceptada si la ejecución desde el estado inicial nos lleva a 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.

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 $k \in \mathbb{N}$.

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

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_{id} cadenas ASCII que son nombres de variables válidas.

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_{id} cadenas ASCII que son nombres de variables válidas.

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".

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:

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

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 $\beta = \gamma\alpha$.

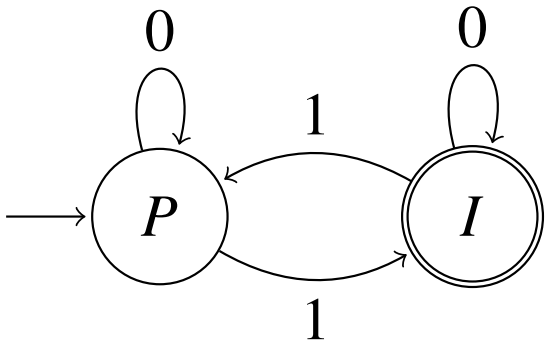
Autómatas finitos deterministas (AFD)

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



$$Q = \{P, I\}$$
$$\Sigma = \{0, 1\}$$
$$q_0 = P$$
$$F = \{I\}$$

δ	0	1
P	P	I
I	I	P

Observaciones

El tipo de la función de transición δ fuerza a que de cada estado salga una (y sólo una) arista por cada símbolo del alfabeto.

Observaciones

El tipo de la función de transición δ fuerza a que de cada estado salga una (y sólo una) arista por cada símbolo del alfabeto.
Por eso hablamos de autómatas deterministas: si el autómata está en un estado sabemos que dado un símbolo de entrada pasará a un único estado siguiente.

Observaciones

El tipo de la función de transición δ fuerza a que de cada estado salga una (y sólo una) arista por cada símbolo del alfabeto.
Por eso hablamos de autómatas deterministas: si el autómata está en un estado sabemos que dado un símbolo de entrada pasará a un único estado siguiente.
Generalizando: dada una palabra existe un único camino desde el estado inicial que consume toda la palabra.

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:

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

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:

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

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

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:

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

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:

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

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)

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

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

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.

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

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 $L_{\mathbb{A}} = L_{\mathbb{A}'}$.

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

Autómatas finitos no deterministas (AFN)

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

Autómatas finitos no deterministas (AFN)

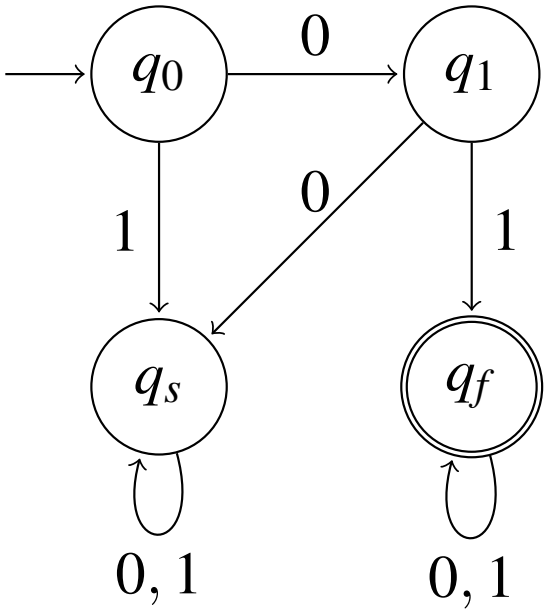
Los autómatas no deterministas son más permisivos que los AFD: permite que para estado haya cero, una o más transiciones por cada símbolo.
Dada una palabra, puede haber más de un camino desde el estado inicial. Por ello, cambiamos la condición de aceptación.

Autómatas finitos no deterministas (AFN)

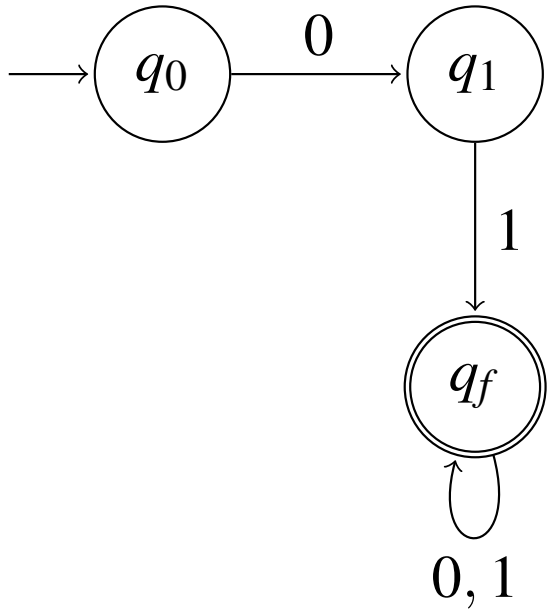
Los autómatas no deterministas son más permisivos que los AFD: permite que para estado haya cero, una o más transiciones por cada símbolo.
Dada una palabra, puede haber más de un camino desde el estado inicial. Por ello, cambiamos la condición de aceptación.
Una palabra es aceptada por un AFN si *existe 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 $\delta : Q \times \Sigma \to \mathscr{P}(Q)$.
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 $\delta : Q \times \Sigma \to \mathscr{P}(Q)$.
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.

Aceptación para NFA

Como $\delta(p, q)$ es un conjunto de estados no podemos definir la extensión de δ tan sencillamente.

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

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

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

Aceptación para NFA

Como $\delta(p, q)$ es un conjunto de estados no podemos definir la extensión de δ tan sencillamente.

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

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

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

Definición

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

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