ST0270 Lenguajes Formales y Compiladores
Lenguajes regulares y autómatas finitos

Andrés Sicard Ramírez

Universidad EAFIT

Semestre 2024-1

Preliminares

Referencias

Las referencias principales para estas diapositivas son (Kozen 2012, Lecturas 3–6, 8, 11).

Convenciones

(i) Los números asignados a los definiciones, ejemplos, ejercicios, páginas y teoremas en estas diapositivas corresponden a los números asignados en (Kozen 2012).

(ii) Los números naturales incluyen el cero, es decir, $\mathbb{N} = \{0, 1, 2, \dots\}$.

(iii) El conjunto potencia de un conjunto A es denotado 2^A.

(iv) Los términos «definición inductiva» y «definición recursiva» se usan como sinónimos.

Introducción

Correspondencia entre las gramáticas y los modelos de computación

Un tipo de gramática genera una clase de lenguajes y un modelo de computación reconoce una clase de lenguajes.

Tipo de gramática	Clase de lenguaje	Modelo de computación
3	Regulares	Autómatas finitos
2	Libres (independientes) de contexto	Autómatas a pila
1	Dependientes del contexto	Autómatas linealmente acotados
0	Recursivamente enumerables	Máquinas de Turing

Autómatas finitos

Descripción

*A **state** of a system is an instantaneous description of that system, a snapshot of reality frozen in time. A state gives all relevant information necessary to determine how the system can evolve from that point on. **Transitions** are changes of state; they can happen spontaneously or in response to external inputs. (Kozen 2012, pág. 14)*

*The **finite automaton** is a mathematical model of a system, with discrete inputs and outputs. The system can be in any one of a finite number of internal configurations or «states». The state of the system summarizes the information concerning past inputs that is needed to determine the behaviour the system on subsequent inputs. (Hopcroft y Ullman 1979, p. 13)*

Autómatas finitos deterministas

Definición

Un **autómata finito determinista** (AFD) (*deterministic finite automaton*) es una estructura

$$(Q, \Sigma, \delta, s, F),$$

donde

(i) Q es un conjunto finito, los **estados**,

(ii) Σ es un conjunto finito, el **alfabeto de entrada**,

(iii) $\delta : Q \times \Sigma \to Q$, la **función de transición**,

(iv) $s \in Q$, el **estado inicial**,

(v) $F \subseteq Q$, el conjunto de **estados finales** o **estados de aceptación**.

Especificaciones de AFDs

Los AFDs se puede especificar de varias formas equivalentes:

(i) Listando cada una de sus partes.

(ii) Diagrama de transición.

(iii) Tabla de transición.

Especificaciones de AFDs

Ejemplo 3.1

Especificamos un AFD de tres formas equivalentes:

(i) Listando cada una de sus partes

$$Q = \{0, 1, 2, 3\},$$

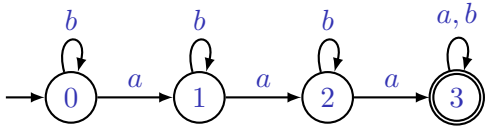
$$\Sigma = \{a, b\},$$

$$\begin{aligned} \delta(0, a) &= 1, \\ \delta(1, a) &= 2, \\ \delta(2, a) &= \delta(3, a) = 3, \\ \delta(q, b) &= q, q \in Q, \end{aligned}$$

$$s = 0,$$

$$F = \{3\}.$$

(ii) Diagrama de transición



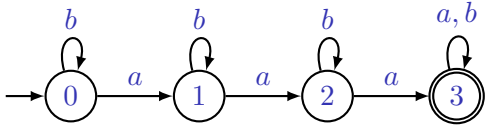
(iii) Tabla de transición

	a	b
$\rightarrow 0$	1	0
1	2	1
2	3	2
$3F$	3	3

Operación de un AFD

Ejemplo

Veamos como **acepta** o **rechaza** una palabra de entrada el AFD de la figura.



Memoria (finita) del AFD:

- Estado 0: El autómata no ha visto ninguna a.
- Estado 1: El autómata ha visto una a.
- Estado 2: El autómata ha visto dos $a$es.
- Estado 3: El autómata ha visto tres o más $a$es.

Función de transición extendida para AFDs

Definición

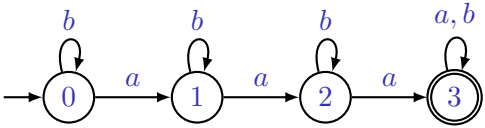
Sea $D = (Q, \Sigma, \delta, s, F)$ un AFD. La **función de transición extendida** (*extended transition function*)*, denotada $\hat{\delta}$, es recursivamente definida por:

$$\hat{\delta} : Q \times \Sigma^* \to Q$$
$$\hat{\delta}(q, \epsilon) \stackrel{\text{def}}{=} q,$$
$$\hat{\delta}(q, xa) \stackrel{\text{def}}{=} \delta(\hat{\delta}(q, x), a).$$

*El nombre «*extended transition function*» es tomado de (Hopcroft, Motwani et al. 2007).

Función de transición extendida para AFDs

Ejemplo



$$
\begin{aligned}
\hat{\delta}(0, baa) &= \hat{\delta}(0, \epsilon baa) \\
&= \delta(\hat{\delta}(0, \epsilon ba), a) \\
&= \delta(\delta(\hat{\delta}(0, \epsilon b), a), a) \\
&= \delta(\delta(\delta(\hat{\delta}(0, \epsilon), b), a), a) \\
&= \delta(\delta(\delta(0, b), a), a) \\
&= \delta(\delta(0, a), a) \\
&= \delta(1, a) \\
&= 2 \notin F
\end{aligned}
$$

Lenguajes aceptados por los AFDs

Definiciones

Sea $D = (Q, \Sigma, \delta, s, F)$ un AFD y sea $x \in \Sigma^*$.

(i) La palabra x es **aceptada** por el autómata D ssi $\hat{\delta}(s, x) \in F$.

Lenguajes aceptados por los AFDs

Definiciones

Sea $D = (Q, \Sigma, \delta, s, F)$ un AFD y sea $x \in \Sigma^*$.

(i) La palabra x es **aceptada** por el autómata D ssi $\hat{\delta}(s, x) \in F$.

(ii) La palabra x es **rechazada** por el autómata D ssi $\hat{\delta}(s, x) \notin F$.

Lenguajes aceptados por los AFDs

Definiciones

Sea $D = (Q, \Sigma, \delta, s, F)$ un AFD y sea $x \in \Sigma^*$.

(i) La palabra x es **aceptada** por el autómata D ssi $\hat{\delta}(s, x) \in F$.

(ii) La palabra x es **rechazada** por el autómata D ssi $\hat{\delta}(s, x) \notin F$.

(iii) El **lenguaje aceptado** por el autómata D, denotado $\mathrm{L}(D)$, es el conjunto de palabras aceptadas por D, es decir,

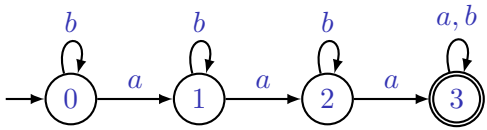
$$\mathrm{L}(D) \stackrel{\text{def}}{=} \left\{ x \in \Sigma^* \;\middle|\; \hat{\delta}(s, x) \in F \right\}.$$

Lenguajes aceptados por los AFDs

Ejemplo

Sea D el AFD de la figura. El lenguaje aceptado por el automata es

$$\mathrm{L}(D) = \{\, x \in \{a, b\}^* \mid x \text{ contiene al menos tres } a\text{es} \,\}.$$



Lenguajes regulares

Definición

Un lenguaje L es **regular** sii existe un AFD D tal que $L = \mathrm{L}(D)$.

Lenguajes regulares

Definición

Un lenguaje L es **regular** sii existe un AFD D tal que $L = \mathrm{L}(D)$.

Observación

Kozen (2012) no habla de «lenguajes regulares» sino de «conjuntos regulares».

Lenguajes regulares

Ejemplo

Sea L el conjunto de palabras sobre $\{a, b\}$ que tienen un número par de $a$es y tienen un número par de $b$es. El lenguaje L es regular.

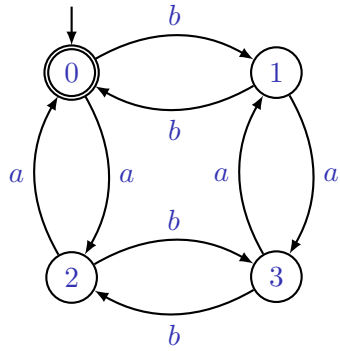
Lenguajes regulares

Ejemplo

Sea L el conjunto de palabras sobre $\{a, b\}$ que tienen un número par de $a$es y tienen un número par de $b$es. El lenguaje L es regular.

Memoria (finita) del AFD:

- Estado 0: El número de $a$es vistos es par y el número de $b$es vistos es par.
- Estado 1: El número de $a$es vistos es par pero el número de $b$es vistos es impar.
- Estado 2: El número de $b$es vistos es par pero el número de $a$es vistos es impar.
- Estado 3: El número de $a$es vistos es impar y el número de $b$es vistos es impar.



Lenguajes regulares

Pregunta

Sea Σ un alfabeto. ¿Es $\emptyset$ un lenguaje regular?

Lenguajes regulares

Pregunta

Sea Σ un alfabeto. ¿Es $\emptyset$ un lenguaje regular? ¿Lo es Σ^*?

Algunas propiedades de clausura de los lenguajes regulares

Sean L y L' lenguajes regulares. Los siguientes lenguajes también son regulares:

$$\begin{array}{rl} \sim L & \text{(complemento)} \\ L \cup L' & \text{(unión)} \\ L \cap L' & \text{(intersección)} \end{array}$$

Clausura para la complementación

Teorema

Si L es un lenguaje regular entonces $\sim L$ también es un lenguaje regular.

Clausura para la complementación

Teorema

Si L es un lenguaje regular entonces $\sim L$ también es un lenguaje regular.

Demostración

Un AFD M' que acepta $\sim L$ se puede construir intercambiando los estados de aceptación y no aceptación de un AFD M que acepta L. Es decir, si

$$L = \mathrm{L}(M), \text{ donde } M = (Q, \Sigma, \delta, s_0, F),$$

entonces

$$\sim L = \mathrm{L}(M'), \text{ donde } M' = (Q, \Sigma, \delta, s_0, Q - F).$$

Clausura para la intersección

Teorema

Si L_1 y L_2 son lenguajes regulares entonces $L_1 \cap L_2$ también es un lenguaje regular.

Clausura para la intersección

Teorema

Si L_1 y L_2 son lenguajes regulares entonces $L_1 \cap L_2$ también es un lenguaje regular.

Demostración (la construcción del producto)

En el tablero (Lema 4.1 y Teorema 4.2).

Clausura para la unión

Teorema

Si L_1 y L_2 son lenguajes regulares entonces $L_1 \cup L_2$ también es un lenguaje regular.

Clausura para la unión

Teorema

Si L_1 y L_2 son lenguajes regulares entonces $L_1 \cup L_2$ también es un lenguaje regular.

Demostración

Los lenguajes regulares son cerrados para la intersección y el complemento y

$$L_1 \cup L_2 = \sim(\sim L_1 \cap \sim L_2) \quad \text{(propiedad de De Morgan)}.$$

Autómatas finitos no deterministas

Introducción

- Una computación es «no determinista» cuando el siguiente estado de ésta no es completamente determinado por el estado actual.

Autómatas finitos no deterministas

Introducción

- Una computación es «no determinista» cuando el siguiente estado de ésta no es completamente determinado por el estado actual.
- El no determinismo es una abstracción importante en Ciencias de la Computación.

Autómatas finitos no deterministas

Introducción

- Una computación es «no determinista» cuando el siguiente estado de ésta no es completamente determinado por el estado actual.
- El no determinismo es una abstracción importante en Ciencias de la Computación.
- No determinismo y diseño eficiente de programas:

> *The famous* $\mathrm{P} = \mathrm{NP}$ *problem—whether all problems solvable in nondeterministic polynomial time can be solved in deterministic polynomial time—is a major open problem in computer science and arguably one of the most important open problems in all of mathematics. (Kozen 2012, pág. 25)*

Autómatas finitos no deterministas

No determinismo y autómatas finitos

- El no determinismo no incrementa el poder computacional (o potencia expresiva) de los autómatas finitos.

Autómatas finitos no deterministas

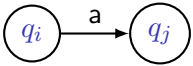
No determinismo y autómatas finitos

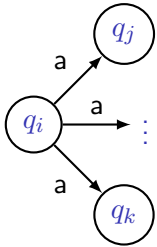
- El no determinismo no incrementa el poder computacional (o potencia expresiva) de los autómatas finitos.
- El procesamiento de una entrada por parte de un autómata finito no determinista se puede pensar en términos de adivinar y verificar.

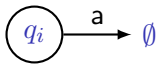
Autómatas finitos no deterministas

No determinismo y autómatas finitos

- El no determinismo no incrementa el poder computacional (o potencia expresiva) de los autómatas finitos.
- El procesamiento de una entrada por parte de un autómata finito no determinista se puede pensar en términos de adivinar y verificar.
- La transición desde un estado y un símbolo puede ser a: un estado, varios estados o ningún estado.





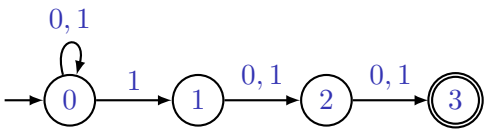


Autómatas finitos no deterministas

Ejemplo

El autómata no determinista (AFN) de la figura acepta el lenguaje

$$\{\, x \in \{0,1\}^* \mid \text{ el tercero símbolo desde la derecha es } 1 \,\}.$$



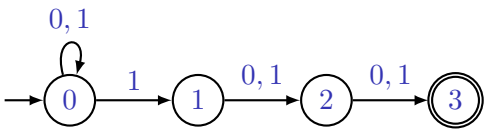
- Estado 0: El autómata supone que no ha visto el tercer símbolo desde la derecha.
- Estado 1: El autómata supone que 1 es el tercer símbolo desde la derecha.
- Estado 2: Los dos últimos símbolos vistos son $1, 0$ o $1, 1$.
- Estado 3: Los tres últimos símbolos vistos son $1, 0, 0$, $1, 0, 1$, $1, 1, 0$ o $1, 1, 1$.

Autómatas finitos no deterministas

Ejemplo

El autómata no determinista (AFN) de la figura acepta el lenguaje

$$\{\, x \in \{0,1\}^* \mid \text{ el tercero símbolo desde la derecha es } 1 \,\}.$$



- Estado 0: El autómata supone que no ha visto el tercer símbolo desde la derecha.
- Estado 1: El autómata supone que 1 es el tercer símbolo desde la derecha.
- Estado 2: Los dos últimos símbolos vistos son $1,0$ o $1,1$.
- Estado 3: Los tres últimos símbolos vistos son $1,0,0$, $1,0,1$, $1,1,0$ o $1,1,1$.

Veamos como el autómata adivina y verifica cuando procesa la entradas 110 y 11.

La construcción de subconjuntos

Introducción

- La construcción de subconjuntos será empleada para demostrar que para cada AFN existe un AFD equivalente, es decir, que acepta el mismo lenguaje.

La construcción de subconjuntos

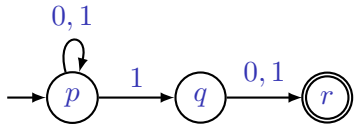
Introducción

- La construcción de subconjuntos será empleada para demostrar que para cada AFN existe un AFD equivalente, es decir, que acepta el mismo lenguaje.
- Dado un AFN se construirá un AFD equivalente donde sus estados serán subconjuntos de estados del AFN.

La construcción de subconjuntos

Ejemplo 5.1

El siguiente AFN N



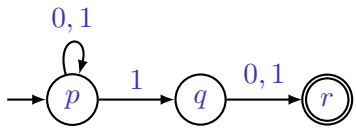
acepta el lenguaje

$$L(N) = \{\, x \in \{0,1\}^* \mid \text{ el segundo símbolo desde la derecha es } 1 \,\}.$$

La construcción de subconjuntos

Ejemplo 5.1

El siguiente AFN N



acepta el lenguaje

$$L(N) = \{\, x \in \{0,1\}^* \mid \text{ el segundo símbolo desde la derecha es } 1 \,\}.$$

Vamos a construir un AFD D equivalente al AFN N vía la construcción de subconjuntos.

(continua en la próxima diapositiva)

La construcción de subconjuntos

Ejemplo 5.1 (continuación)

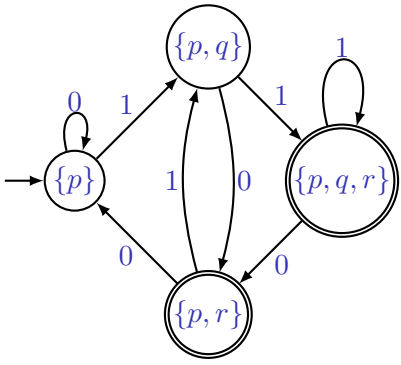
(i) Construcción del AFD D vía la construcción de subconjuntos.

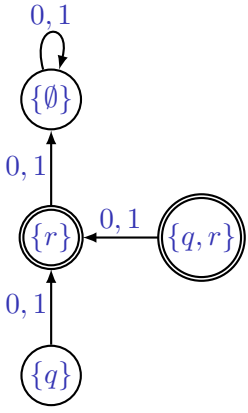
La construcción de subconjuntos

Ejemplo 5.1 (continuación)

(i) Construcción del AFD D vía la construcción de subconjuntos.

	0	1
$\emptyset$	$\emptyset$	$\emptyset$
$\to \{p\}$	$\{p\}$	$\{p,q\}$
$\{q\}$	$\{r\}$	$\{r\}$
$\{r\}F$	$\emptyset$	$\emptyset$
$\{p,q\}$	$\{p,r\}$	$\{p,q,r\}$
$\{p,r\}F$	$\{p\}$	$\{p,q\}$
$\{q,r\}F$	$\{r\}$	$\{r\}$
$\{p,q,r\}F$	$\{p,r\}$	$\{p,q,r\}$



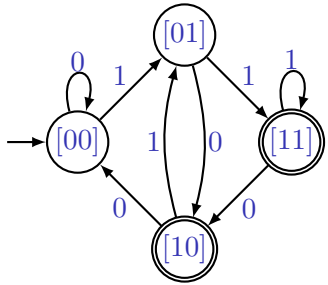


(continua en la próxima diapositiva)

La construcción de subconjuntos

Ejemplo 5.1 (continuación)

(ii) Removiendo estados inalcanzables y renombrados estados



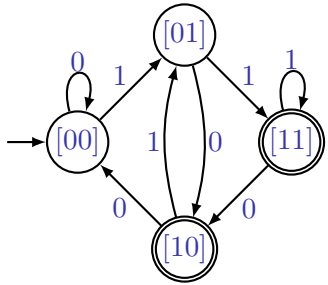
$$[00] = \{p\},$$
$$[01] = \{p, q\},$$
$$[10] = \{p, r\},$$
$$[11] = \{p, q, r\}.$$

Estado $[ab]$: Los dos últimos símbolos vistos son a y b.

La construcción de subconjuntos

Ejemplo 5.1 (continuación)

(ii) Removiendo estados inalcanzables y renombrados estados



$$[00] = \{p\},$$
$$[01] = \{p, q\},$$
$$[10] = \{p, r\},$$
$$[11] = \{p, q, r\}.$$

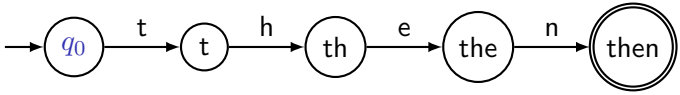
Estado $[ab]$: Los dos últimos símbolos vistos son a y b.

(iii) El AFD D acepta el mismo lenguaje que el AFN N.

La construcción de subconjuntos

Ejemplo (caso especial)

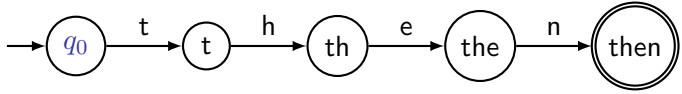
Un AFN aceptando la palabra «then»:



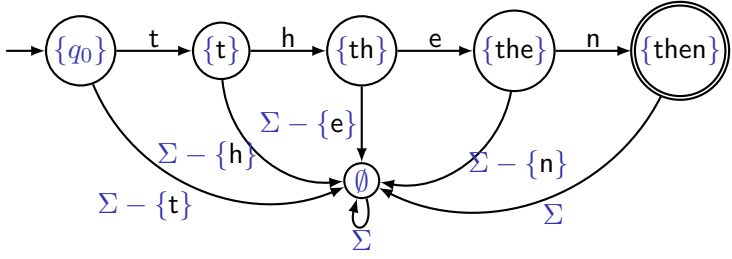
La construcción de subconjuntos

Ejemplo (caso especial)

Un AFN aceptando la palabra «then»:



El AFD equivalente obtenido por la construcción de subconjuntos:



Autómatas finitos no deterministas

Definición

Un **autómata finito no determinista** (AFN) (*nondeterministic finite automaton*) es una estructura

$$(Q, \Sigma, \Delta, S, F),$$

donde

(i) Q es un conjunto finito, los **estados**,

(ii) Σ es un conjunto finito, el **alfabeto de entrada**,

(iii) $\Delta : 2^Q \times \Sigma \to 2^Q$, la **función de transición**,

(iv) $S \subseteq Q$, el **conjunto de estados iniciales**,

(v) $F \subseteq Q$, el conjunto de **estados finales** o **estados de aceptación**.

Función de transición extendida para AFNs

Definición

Sea $N = (Q, \Sigma, \Delta, S, F)$ un AFN. La **función de transición extendida** (*extended transition function*)*, denotada $\hat{\Delta}$, es recursivamente definida por:

$$\hat{\Delta} : 2^Q \times \Sigma^* \to 2^Q$$

$$\hat{\Delta}(A, \epsilon) \stackrel{\text{def}}{=} A,$$

$$\hat{\Delta}(A, xa) \stackrel{\text{def}}{=} \bigcup_{q \in \hat{\Delta}(A,x)} \Delta(q, a).$$

*El nombre «*extended transition function*» es tomado de (Hopcroft, Motwani et al. 2007).

Función de transición extendida para AFNs

Lema 6.1

Para cualquier $x, y \in \Sigma^*$ y para cualquier $A \subseteq Q$,

$$\hat{\Delta}(A, xy) = \hat{\Delta}(\hat{\Delta}(A, x), y).$$

Función de transición extendida para AFNs

Lema 6.1

Para cualquier $x, y \in \Sigma^*$ y para cualquier $A \subseteq Q$,

$$\hat{\Delta}(A, xy) = \hat{\Delta}(\hat{\Delta}(A, x), y).$$

Demostración (por inducción en y)

En el tablero.

Lenguajes aceptados por los AFNs

Definiciones

Sea $N = (Q, \Sigma, \Delta, S, F)$ un AFN y sea $x \in \Sigma^*$.

(i) La palabra x es **aceptada** por N ssi $\hat{\Delta}(S, x) \cap F \neq \emptyset$.

Lenguajes aceptados por los AFNs

Definiciones

Sea $N = (Q, \Sigma, \Delta, S, F)$ un AFN y sea $x \in \Sigma^*$.

(i) La palabra x es **aceptada** por N ssi $\hat{\Delta}(S, x) \cap F \neq \emptyset$.

(ii) La palabra x es **rechazada** por N ssi $\hat{\Delta}(S, x) \cap F = \emptyset$.

Lenguajes aceptados por los AFNs

Definiciones

Sea $N = (Q, \Sigma, \Delta, S, F)$ un AFN y sea $x \in \Sigma^*$.

(i) La palabra x es **aceptada** por N ssi $\hat{\Delta}(S, x) \cap F \neq \emptyset$.

(ii) La palabra x es **rechazada** por N ssi $\hat{\Delta}(S, x) \cap F = \emptyset$.

(iii) El **lenguaje aceptado** por el autómata N, denotado $L(N)$, es el conjunto de palabras aceptadas por N, es decir,

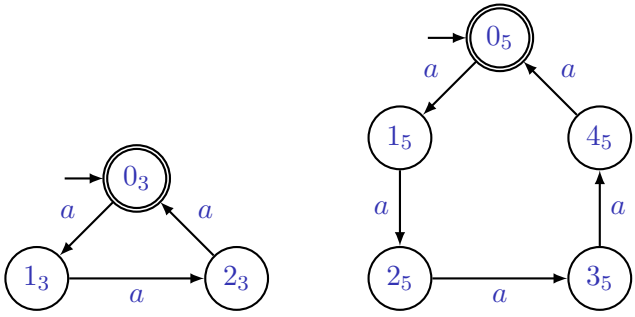
$$\mathrm{L}(N) \stackrel{\text{def}}{=} \Big\{ x \in \Sigma^* \;\Big|\; \hat{\Delta}(S, x) \cap F \neq \emptyset \Big\}.$$

Lenguajes aceptados por los AFNs

Ejemplo 5.2 (un AFN con dos estados iniciales)

Sea N el AFN de la figura. El lenguaje aceptado por el autómata es

$$\mathrm{L}(N) = \{\, x \in \{a\}^* \mid |x| \text{ es divisible por } 3 \text{ o por } 5 \,\}.$$



La construcción de subconjuntos

Definición

Sea un AFN $N = (Q_N, \Sigma, \Delta_N, S_N, F_N)$. La **construcción de subconjuntos** especifica un AFD $D = (Q_D, \Sigma, \delta_D, s_D, F_D)$ donde

La construcción de subconjuntos

Definición

Sea un AFN $N = (Q_N, \Sigma, \Delta_N, S_N, F_N)$. La **construcción de subconjuntos** especifica un AFD $D = (Q_D, \Sigma, \delta_D, s_D, F_D)$ donde

$$
\begin{aligned}
Q_D &\stackrel{\text{def}}{=} 2^{Q_N}, \\
\delta_D(A, a) &\stackrel{\text{def}}{=} \hat{\Delta}_N(A, a), \text{ para todo } A \subseteq Q_N \text{ y } a \in \Sigma, \\
s_D &\stackrel{\text{def}}{=} S_N, \\
F_D &\stackrel{\text{def}}{=} \{\, A \subseteq Q_N) \mid A \cap F_N \neq \emptyset \,\}.
\end{aligned}
$$

La construcción de subconjuntos

Lema 6.3

Para cualesquiera $A \subseteq Q$ y $x \in \Sigma^*$,

$$\hat{\delta}_D(A, x) = \hat{\Delta}_N(A, x).$$

La construcción de subconjuntos

Lema 6.3

Para cualesquiera $A \subseteq Q$ y $x \in \Sigma^*$,

$$\hat{\delta}_D(A, x) = \hat{\Delta}_N(A, x).$$

Demostración (por inducción en y)

En el tablero.

La construcción de subconjuntos

Teorema 6.4

Sea D el AFD obtenido a partir de un AFN N vía la construcción de subconjuntos, entonces $\mathrm{L}(D) = \mathrm{L}(N)$.

La construcción de subconjuntos

Teorema 6.4

Sea D el AFD obtenido a partir de un AFN N vía la construcción de subconjuntos, entonces $\mathrm{L}(D) = \mathrm{L}(N)$.

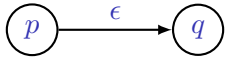
Demostración

En el tablero.

Transiciones-ϵ

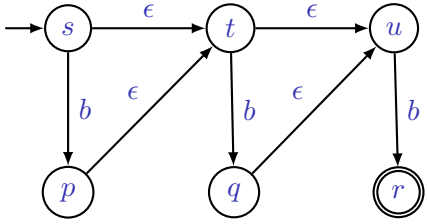
Definición

Una **transición-ϵ** (ϵ-*transition*) es una transición de un estado p a un estado q con la palabra vacía.



Transiciones-ϵ

Ejemplo 6.5



El lenguaje aceptado por el autómata es $\{b, bb, bbb\}$.

Algunas propiedades adicionales de clausura de los lenguajes regulares

Sean L y L' lenguajes regulares. Los siguientes lenguajes también son regulares:

$\sim L$	(complemento)
$L \cup L'$	(unión)
$L \cap L'$	(intersección)
LL'	(concatenación)
L^*	(clausura de Kleene)

Clausura para la concatenación

Teorema

Si L_1 y L_2 son lenguajes regulares entonces $L_1 L_2$ también es un lenguaje regular.

Clausura para la concatenación

Teorema

Si L_1 y L_2 son lenguajes regulares entonces $L_1 L_2$ también es un lenguaje regular.

Demostración

En el tablero.

Clausura para clausura de Kleene

Teorema

Si L es un lenguaje regular entonces L^* también es un lenguaje regular.

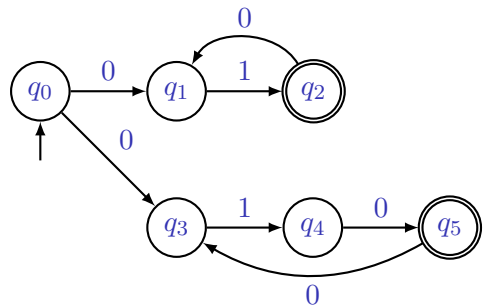
Demostración

En el tablero.

Expresiones regulares

$$(\mathbf{01})(\mathbf{01})^* + (\mathbf{010})(\mathbf{010})^*$$

Descripción algebraica



Descripción de una máquina

Expresiones regulares

Introducción

- Especificación algebraica y declarativa para los lenguajes regulares.

Expresiones regulares

Introducción

- Especificación algebraica y declarativa para los lenguajes regulares.
- Las expresiones regulares son usadas por ejemplo en comandos de búsqueda (*p. ej.* grep en Linux), en los generadores de analizadores lexicográficos (*p. ej.* lex) y en los *domain specific languages (DSL's)*.

Expresiones regulares

Definición

Las **expresiones regulares** sobre un alfabeto Σ es el conjunto más pequeño definido inductivamente por:

Expresiones regulares

Definición

Las **expresiones regulares** sobre un alfabeto Σ es el conjunto más pequeño definido inductivamente por:

(i) Paso base

- Si $a \in \Sigma$ entonces $\boldsymbol{a}$ es una expresión regular.
- $\boldsymbol{\epsilon}$ y $\boldsymbol{\emptyset}$ son expresiones regulares.

Expresiones regulares

Definición

Las **expresiones regulares** sobre un alfabeto Σ es el conjunto más pequeño definido inductivamente por:

(i) Paso base

- Si $a \in \Sigma$ entonces $\boldsymbol{a}$ es una expresión regular.
- $\boldsymbol{\epsilon}$ y $\boldsymbol{\emptyset}$ son expresiones regulares.

(ii) Paso inductivo
Si α and β son expresiones regulares entonces $\alpha + \beta$, $\alpha \cdot \beta$, α^* y (α) son expresiones regulares.

(continua en la próxima diapositiva)

Expresiones regulares

Definición (continuación)

Las **expresiones regulares** ER sobre un alfabeto Σ son definidas por las siguientes reglas de inferencia:

$$\frac{a \in \Sigma}{\boldsymbol{a} \in \mathrm{ER},} \qquad \frac{}{\boldsymbol{\epsilon} \in \mathrm{ER},} \qquad \frac{}{\boldsymbol{\emptyset} \in \mathrm{ER},}$$

$$\frac{\alpha, \beta \in \mathrm{ER}}{\alpha + \beta \in \mathrm{ER},} \qquad \frac{\alpha, \beta \in \mathrm{ER}}{\alpha \cdot \beta \in \mathrm{ER},} \qquad \frac{\alpha \in \mathrm{ER}}{\alpha^* \in \mathrm{ER},} \qquad \frac{\alpha \in \mathrm{ER}}{(\alpha) \in \mathrm{ER},}$$

Expresiones regulares

Observaciones

(i) Note que el símbolo a es diferente de la expresión regular $\boldsymbol{a}$ (Kozen (2012) no usa está convención).

(ii) Note que la palabra vacía ϵ es diferente de la expresión regular $\boldsymbol{\epsilon}$.

(iii) Note que el lenguaje vacío $\emptyset$ es diferente de la expresión regular $\boldsymbol{\emptyset}$.

Expresiones regulares

Convenciones

- Las expresiones regulares se denotarán por letras griegas minúsculas $\alpha, \beta, \gamma, \ldots$.
- El operador «$\cdot$» se puede omitir, es decir, $\alpha\beta \stackrel{\text{def}}{=} \alpha \cdot \beta$.

Ejemplos

En el tablero.

Expresiones regulares

Precedencia y asociatividad de los operadores

(i) El operador «*» tiene mayor precedencia que el operador «$\cdot$» que a su vez tiene mayor precedencia que el operador «$+$».

(ii) Los operadores «$\cdot$» y «$+$» son asociativos.

Ejemplos

En el tablero.

Expresiones regulares

Definición

El **lenguaje denotado** por una expresión regular α, denotado $\mathrm{L}(\alpha)$, es definido inductivamente por:

Expresiones regulares

Definición

El **lenguaje denotado** por una expresión regular α, denotado $\mathrm{L}(\alpha)$, es definido inductivamente por:

(i) Paso base

$$\mathrm{L}(\boldsymbol{a}) \stackrel{\text{def}}{=} \{a\},$$
$$\mathrm{L}(\boldsymbol{\epsilon}) \stackrel{\text{def}}{=} \{\epsilon\},$$
$$\mathrm{L}(\boldsymbol{\emptyset}) \stackrel{\text{def}}{=} \emptyset.$$

Expresiones regulares

Definición

El **lenguaje denotado** por una expresión regular α, denotado $\mathrm{L}(\alpha)$, es definido inductivamente por:

(i) Paso base

$$\mathrm{L}(\boldsymbol{a}) \stackrel{\text{def}}{=} \{a\},$$
$$\mathrm{L}(\boldsymbol{\epsilon}) \stackrel{\text{def}}{=} \{\epsilon\},$$
$$\mathrm{L}(\boldsymbol{\emptyset}) \stackrel{\text{def}}{=} \emptyset.$$

(ii) Paso inductivo

Sean $\mathrm{L}(\alpha)$ y $\mathrm{L}(\beta)$ los lenguajes denotados por las expresiones regulares α y β, entonces

$$\mathrm{L}(\alpha + \beta) \stackrel{\text{def}}{=} \mathrm{L}(\alpha) \cup \mathrm{L}(\beta),$$
$$\mathrm{L}(\alpha \cdot \beta) \stackrel{\text{def}}{=} \mathrm{L}(\alpha)\mathrm{L}(\beta),$$
$$\mathrm{L}(\alpha^*) \stackrel{\text{def}}{=} (\mathrm{L}(\alpha))^*,$$
$$\mathrm{L}((\alpha)) \stackrel{\text{def}}{=} \mathrm{L}(\alpha).$$

Expresiones regulares

Ejemplo

α	$\mathrm{L}(\alpha)$
$\boldsymbol{a} + \boldsymbol{b}$	$\mathrm{L}(\boldsymbol{a}) \cup \mathrm{L}(\boldsymbol{b}) = \{a\} \cup \{b\} = \{a, b\}$

Expresiones regulares

Ejemplo

α	$\mathrm{L}(\alpha)$
$\boldsymbol{a} + \boldsymbol{b}$	$\mathrm{L}(\boldsymbol{a}) \cup \mathrm{L}(\boldsymbol{b}) = \{a\} \cup \{b\} = \{a, b\}$
$\boldsymbol{a}^*$	$\{\epsilon, a, aa, aaa, \ldots\}$

Expresiones regulares

Ejemplo

α	$\mathrm{L}(\alpha)$
$\boldsymbol{a} + \boldsymbol{b}$	$\mathrm{L}(\boldsymbol{a}) \cup \mathrm{L}(\boldsymbol{b}) = \{a\} \cup \{b\} = \{a, b\}$
$\boldsymbol{a}^*$	$\{\epsilon, a, aa, aaa, \ldots\}$
$(\boldsymbol{a} + \boldsymbol{b})(\boldsymbol{a} + \boldsymbol{b})$	$\mathrm{L}(\boldsymbol{a} + \boldsymbol{b})\mathrm{L}(\boldsymbol{a} + \boldsymbol{b}) = \{a, b\}\{a, b\} = \{aa, ab, ba, bb\}$

Expresiones regulares

Ejemplo

α	$\mathrm{L}(\alpha)$
$\boldsymbol{a} + \boldsymbol{b}$	$\mathrm{L}(\boldsymbol{a}) \cup \mathrm{L}(\boldsymbol{b}) = \{a\} \cup \{b\} = \{a, b\}$
$\boldsymbol{a}^*$	$\{\epsilon, a, aa, aaa, \ldots\}$
$(\boldsymbol{a} + \boldsymbol{b})(\boldsymbol{a} + \boldsymbol{b})$	$\mathrm{L}(\boldsymbol{a} + \boldsymbol{b})\mathrm{L}(\boldsymbol{a} + \boldsymbol{b}) = \{a, b\}\{a, b\} = \{aa, ab, ba, bb\}$
$\boldsymbol{a} + (\boldsymbol{ab})^*$	$\{a, \epsilon, ab, abab, ababab, \ldots\}$

Expresiones regulares

Ejemplo

α	$\mathrm{L}(\alpha)$
$\boldsymbol{a} + \boldsymbol{b}$	$\mathrm{L}(\boldsymbol{a}) \cup \mathrm{L}(\boldsymbol{b}) = \{a\} \cup \{b\} = \{a, b\}$
$\boldsymbol{a}^*$	$\{\epsilon, a, aa, aaa, \ldots\}$
$(\boldsymbol{a} + \boldsymbol{b})(\boldsymbol{a} + \boldsymbol{b})$	$\mathrm{L}(\boldsymbol{a} + \boldsymbol{b})\mathrm{L}(\boldsymbol{a} + \boldsymbol{b}) = \{a, b\}\{a, b\} = \{aa, ab, ba, bb\}$
$\boldsymbol{a} + (\boldsymbol{ab})^*$	$\{a, \epsilon, ab, abab, ababab, \ldots\}$
$(\boldsymbol{0} + \boldsymbol{1})^*\boldsymbol{01}(\boldsymbol{0} + \boldsymbol{1})^*$	$\{\, x01y \mid x, y \in \{0, 1\}^* \,\}$

Expresiones regulares

Ejemplo

α	$\mathrm{L}(\alpha)$
$\boldsymbol{a} + \boldsymbol{b}$	$\mathrm{L}(\boldsymbol{a}) \cup \mathrm{L}(\boldsymbol{b}) = \{a\} \cup \{b\} = \{a, b\}$
$\boldsymbol{a}^*$	$\{\epsilon, a, aa, aaa, \ldots\}$
$(\boldsymbol{a} + \boldsymbol{b})(\boldsymbol{a} + \boldsymbol{b})$	$\mathrm{L}(\boldsymbol{a} + \boldsymbol{b})\mathrm{L}(\boldsymbol{a} + \boldsymbol{b}) = \{a, b\}\{a, b\} = \{aa, ab, ba, bb\}$
$\boldsymbol{a} + (\boldsymbol{ab})^*$	$\{a, \epsilon, ab, abab, ababab, \ldots\}$
$(\mathbf{0} + \mathbf{1})^*\mathbf{01}(\mathbf{0} + \mathbf{1})^*$	$\{\, x01y \mid x, y \in \{0, 1\}^* \,\}$
$\boldsymbol{a_i}(\boldsymbol{a_1} + \boldsymbol{a_2} + \cdots + \boldsymbol{a_n})^*$	$\{\, x \in \Sigma^* \mid x$ comienza por $a_i \,\}$

Expresiones regulares

Ejemplo

Escribir una expresión regular para el lenguaje L definido por

$$L = \{\, x \in \{a, b\}^* \mid a \text{ y } b \text{ están alternadas en } x \,\}.$$

Expresiones regulares

Ejemplo

Escribir una expresión regular para el lenguaje L definido por

$$L = \{\, x \in \{a, b\}^* \mid a \text{ y } b \text{ están alternadas en } x \,\}.$$

Solución.

$$(\boldsymbol{ab})^* + (\boldsymbol{ba})^* + \boldsymbol{a}(\boldsymbol{ba})^* + \boldsymbol{b}(\boldsymbol{ab})^*$$

Expresiones regulares

Ejemplo

Escribir una expresión regular para el lenguaje L definido por

$$L = \{\, x \in \{a, b\}^* \mid a \text{ y } b \text{ están alternadas en } x \,\}.$$

Solución.

$$(\boldsymbol{ab})^* + (\boldsymbol{ba})^* + \boldsymbol{a}(\boldsymbol{ba})^* + \boldsymbol{b}(\boldsymbol{ab})^*$$

Otra solución.

$$(\boldsymbol{\epsilon} + \boldsymbol{b})(\boldsymbol{ab})^*(\boldsymbol{\epsilon} + \boldsymbol{a})$$

Expresiones regulares

Ejemplo

La expresión regular

$$(\mathbf{10} + \mathbf{0})^*(\boldsymbol{\epsilon} + \mathbf{1})$$

denota el conjunto de palabras de 0s y 1s que no tienen dos 1s adyacentes.

Expresiones regulares

Equivalencia autómatas finitos y expresiones regulares

Expresiones regulares

Bibliotecas

Varios lenguajes de programación tienen bibliotecas o soportan las expresiones regulares, entre ellos están: .NET, C, Haskell, Java, Mathematica, MATLAB y Perl.

Expresiones regulares

Bibliotecas

Varios lenguajes de programación tienen bibliotecas o soportan las expresiones regulares, entre ellos están: .NET, C, Haskell, Java, Mathematica, MATLAB y Perl.

Observación

Expresiones regulares «teóricas» $\neq$ Expresiones regulares «implementadas».

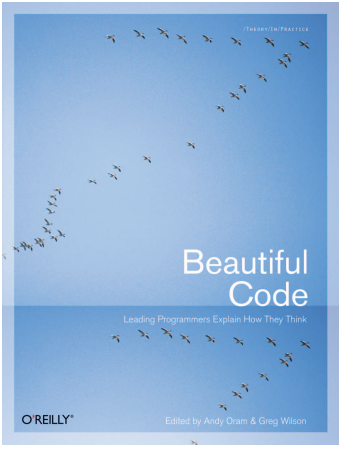
Expresiones regulares

Algunos programas que usan expresiones regulares

Grep:	Busca patrones (cadenas de texto) en un archivo
Awk:	Procesa patrones (cadenas de texto) en líneas de texto
Flex y Lex:	Generadores de analizadores lexicográficos
Emacs y Vim:	Editores de texto
MySQL y Oracle:	Bases de datos

Expresiones regulares

Correspondencia con expresiones regulares (*a regular expression matcher*)



«Rob's implementation itself is a superb example of beautiful code: compact, elegant, efficient, and useful. It's one of the best examples of recursion that I have ever seen.»

Brian Kernighan, pág. 3.

Limitaciones de los autómatas finitos

Introducción

- ¿Es $L_1 = \{ 0^m 1^n \mid m, n \geq 0 \}$ un lenguaje regular?

Limitaciones de los autómatas finitos

Introducción

- ¿Es $L_1 = \{\, 0^m 1^n \mid m, n \geq 0 \,\}$ un lenguaje regular? Sí, $L_1 = \mathrm{L}(\mathbf{0^*1^*})$.

Limitaciones de los autómatas finitos

Introducción

- ¿Es $L_1 = \{\, 0^m 1^n \mid m, n \geq 0 \,\}$ un lenguaje regular? Sí, $L_1 = \mathrm{L}(\mathbf{0}^*\mathbf{1}^*)$.
- ¿Es $L_2 = \{\, 0^m 1^n \mid m, n \geq 1 \,\}$ un lenguaje regular?

Limitaciones de los autómatas finitos

Introducción

- ¿Es $L_1 = \{\, 0^m 1^n \mid m, n \geq 0 \,\}$ un lenguaje regular? Sí, $L_1 = \mathrm{L}(\mathbf{0^*1^*})$.
- ¿Es $L_2 = \{\, 0^m 1^n \mid m, n \geq 1 \,\}$ un lenguaje regular? Sí, $L_2 = \mathrm{L}(\mathbf{00^*11^*})$.

Limitaciones de los autómatas finitos

Introducción

- ¿Es $L_1 = \{\, 0^m 1^n \mid m, n \geq 0 \,\}$ un lenguaje regular? Sí, $L_1 = \mathrm{L}(\mathbf{0^*1^*})$.
- ¿Es $L_2 = \{\, 0^m 1^n \mid m, n \geq 1 \,\}$ un lenguaje regular? Sí, $L_2 = \mathrm{L}(\mathbf{00^*11^*})$.
- ¿Es $L_3 = \{\, 0^m 1^n \mid m \geq 2, n \geq 4 \,\}$ un lenguaje regular?

Limitaciones de los autómatas finitos

Introducción

- ¿Es $L_1 = \{\, 0^m 1^n \mid m, n \geq 0 \,\}$ un lenguaje regular? Sí, $L_1 = \mathrm{L}(\mathbf{0^*1^*})$.
- ¿Es $L_2 = \{\, 0^m 1^n \mid m, n \geq 1 \,\}$ un lenguaje regular? Sí, $L_2 = \mathrm{L}(\mathbf{00^*11^*})$.
- ¿Es $L_3 = \{\, 0^m 1^n \mid m \geq 2, n \geq 4 \,\}$ un lenguaje regular? Sí, $L_3 = \mathrm{L}(\mathbf{000^*11111^*})$.

Limitaciones de los autómatas finitos

Introducción

- ¿Es $L_1 = \{\, 0^m 1^n \mid m, n \geq 0 \,\}$ un lenguaje regular? Sí, $L_1 = \mathrm{L}(\mathbf{0^*1^*})$.
- ¿Es $L_2 = \{\, 0^m 1^n \mid m, n \geq 1 \,\}$ un lenguaje regular? Sí, $L_2 = \mathrm{L}(\mathbf{00^*11^*})$.
- ¿Es $L_3 = \{\, 0^m 1^n \mid m \geq 2, n \geq 4 \,\}$ un lenguaje regular? Sí, $L_3 = \mathrm{L}(\mathbf{000^*11111^*})$.
- ¿Es $L_4 = \{\, 0^n 1^n \mid n \geq 1 \,\}$ un lenguaje regular?

Limitaciones de los autómatas finitos

Introducción

- ¿Es $L_1 = \{\, 0^m 1^n \mid m, n \geq 0 \,\}$ un lenguaje regular? Sí, $L_1 = \mathrm{L}(\mathbf{0^*1^*})$.
- ¿Es $L_2 = \{\, 0^m 1^n \mid m, n \geq 1 \,\}$ un lenguaje regular? Sí, $L_2 = \mathrm{L}(\mathbf{00^*11^*})$.
- ¿Es $L_3 = \{\, 0^m 1^n \mid m \geq 2, n \geq 4 \,\}$ un lenguaje regular? Sí, $L_3 = \mathrm{L}(\mathbf{000^*11111^*})$.
- ¿Es $L_4 = \{\, 0^n 1^n \mid n \geq 1 \,\}$ un lenguaje regular?

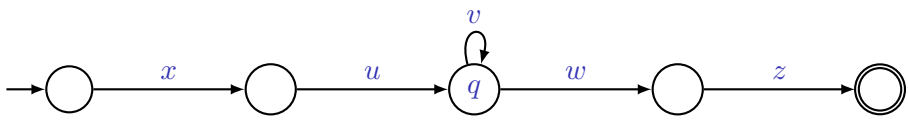
  No, demostración informal (en el tablero).

Limitaciones de los autómatas finitos

Teorema 11.1 (El lema del bombeo (*the pumping lemma*))

Sea L un lenguaje regular, entonces L satisface la siguiente propiedad:

(P) Existe $k \in \mathbb{N}$, tal que para todas las palabras x, y, z con $xyz \in L$ y $|y| \geq k$, existen palabras u, v, w tales que $y = uvw$, $v \neq \epsilon$ y para toda $i \in \mathbb{N}$, $xuv^iwz \in L$.



Limitaciones de los autómatas finitos

Observación

La propiedad (P) es una condición necesaria pero no suficiente, es decir, existen lenguajes que satisfacen la propiedad pero no son regulares.

Limitaciones de los autómatas finitos

Teorema 11.2 (Contrapositivo del lema del bombeo)

Sea L un lenguaje que satisface la siguiente propiedad:

($\neg$P) Para toda $k \in \mathbb{N}$, existen palabras x, y, z con $xyz \in L$ y $|y| \geq k$, y para todas las palabras u, v, w con $y = uvw$ y $v \neq \epsilon$, existe $i \in \mathbb{N}$ tal que $xuv^iwz \notin L$.

Entonces L no es un lenguaje regular.

Limitaciones de los autómatas finitos

Un juego entre adversarios

El contrapositivo del lema del bombeo se puede emplear para demostrar que un lenguaje no es regular. La demostración se puede pensar como un juego entre adversarios, en donde usted desea demostrar que un lenguaje L es no regular y su adversario lo contrario.

Limitaciones de los autómatas finitos

Un juego entre adversarios

El contrapositivo del lema del bombeo se puede emplear para demostrar que un lenguaje no es regular. La demostración se puede pensar como un juego entre adversarios, en donde usted desea demostrar que un lenguaje L es no regular y su adversario lo contrario.

El juego es el siguiente:

1. Su adversario selecciona $k \in \mathbb{N}$.
2. Usted selecciona x, y, z tales que $xyz \in L$ y $|y| \geq k$.
3. Su adversario selecciona u, v, w tales que $y = uvw$ y $v \neq \epsilon$.
4. Usted selecciona $i \in \mathbb{N}$.

Usted gana si $xuv^iwz \notin L$ y su adversario gana de lo contrario.

Limitaciones de los autómatas finitos

Otros métodos para demostrar que un lenguaje no es regular

Frishberg y Gasarch (2018) muestran otros métodos e indican algunos problemas abiertos cuando se demuestra que un lenguaje no es regular. El problema abierto 3.2 está relacionado con el lema del bombeo.

Referencias


Frishberg, Daniel y William Gasarch (2018). «Open Problems Column. Different Ways to Prove a Language is Not Regular». En: *SIGACT News* 49.1, págs. 40-54. DOI: 10.1145/3197406.3197413 (vid. pág. 107).


Hopcroft, John E., Rajeev Motwani y Jefferey D. Ullman [1979] (2007). *Introduction to Automata Theory, Languages, and Computation*. 3.ª ed. Pearson Education (vid. págs. 9, 47).


Hopcroft, John E. y Jefferey D. Ullman (1979). *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley (vid. pág. 4).


Kozen, Dexter C. [1997] (2012). *Automata and Computability*. Third printing. Undergraduate Texts in Computer Science. Springer. DOI: 10.1007/978-1-4612-1844-9 (vid. págs. 2, 4, 15, 16, 28-30, 73).