Capítulo 3

Otras propiedades de los lenguajes regulares

En los dos capítulos anteriores hemos presentado las propiedades básicas de los lenguajes regulares pero no hemos visto cómo se puede demostrar que un lenguaje no es regular. El llamado "lema de bombeo", expuesto en este capítulo, sirve para tal propósito. También veremos que la regularidad es una propiedad que se preserva por las operaciones booleanas usuales, por homomorfismos y por las imágenes inversas de homomorfismos. Finalmente, analizaremos ciertos problemas de decisión referentes a autómatas y a lenguajes regulares.

3.1. Lema de bombeo

El llamado "lema de bombeo" (*pumping lemma*, en inglés) es una propiedad de los lenguajes regulares que es muy útil para demostrar que ciertos lenguajes no son regulares.

3.1.1. Lema de bombeo. *Para todo lenguaje regular L (sobre un alfabeto dado Σ) existe una constante $n \in \mathbb{N}$, llamada constante de bombeo para L, tal que toda cadena $w \in L$, con $|w| \geq n$, satisface la siguiente propiedad:*

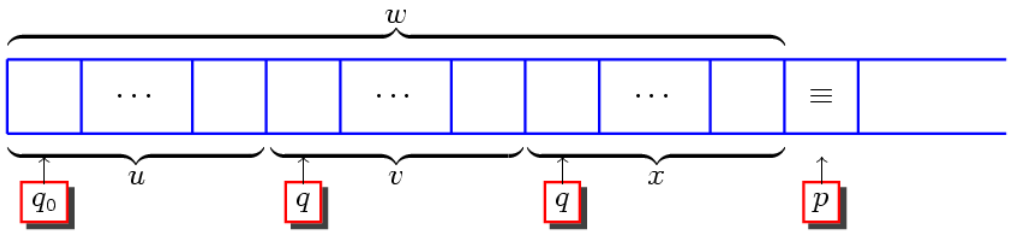
$$(B) \begin{cases} w \text{ se puede descomponer como } w = uvx, \text{ con } |uv| \leq n, \\ v \neq \lambda, \text{ y para todo } i \geq 0 \text{ se tiene } uv^i x \in L. \end{cases}$$

Demostración: Por el Teorema de Kleene y por los teoremas de equivalencia de los modelos AFD, AFN y AFN-λ, existe un AFD M tal que $L(M) = L$. Sea

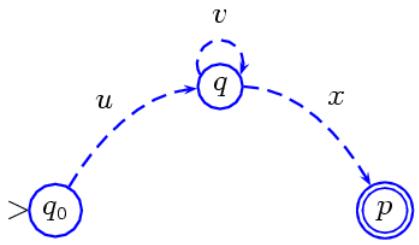
$$n = \#\text{ de estados de } M.$$

Si $w \in L$ y $|w| \geq n$, entonces durante el procesamiento completo de w, hay por lo menos un estado que se repite. Sea q el primer estado que se repite. Tal como se

muestra en la siguiente gráfica, w se puede descomponer como $w = uvx$, donde $|uv| \leq n$, $v \neq \lambda$.



Nótese que tanto u como x pueden ser la cadena vacía λ, pero $v \neq \lambda$. Además, la cadena v se puede "bombear", en el sentido de que uv^ix es aceptada por M para todo $i \geq 0$. En el diagrama de estados, se puede visualizar esta propiedad de bombeo de v:



Uso del lema de bombeo. El lema de bombeo se puede usar para concluir que un cierto lenguaje dado L no es regular, recurriendo a un razonamiento por contradicción (o reducción al absurdo). El razonamiento utilizado tiene la siguiente forma:

1. Si L fuera regular, existiría una constante de bombeo n para L.
2. Se escoge una cadena "adecuada" $w \in L$ y se aplica la propiedad (B) del lema de bombeo, descomponiendo w como
$$w = uvx, \quad v \neq \lambda, \quad |uv| \leq n.$$
3. Se llega a la siguiente contradicción:
   (I) Por el lema de bombeo, $uv^ix \in L$, para todo $i \geq 0$.
   (II) uv^ix no puede estar en L, para algún $i \in I$. Por lo general, basta escoger valores pequeños de i como $i = 0$ ó $i = 2$.

Ejemplo Usar el lema de bombeo para demostrar que el lenguaje $L = \{a^ib^i : i \geq 0\}$ no es regular.

Solución: Si L fuera regular, existiría una constante de bombeo n para L. Sea $w = a^n b^n \in L$. Entonces w se puede descomponer como $w = uvx$, con $|v| \geq 1$ y $|uv| \leq n$. Por lo tanto, u y v constan únicamente de $a$es:

$$\begin{aligned} u &= a^r, \qquad \text{para algún } r \geq 0, \\ v &= a^s, \qquad \text{para algún } s \geq 1. \end{aligned}$$

Entonces,

$$x = a^{n-(r+s)} b^n = a^{n-r-s} b^n.$$

Por el lema de bombeo, $uv^i x \in L$ para todo $i \geq 0$. En particular, si $i = 0$, $ux \in L$. Pero $ux = a^r a^{n-r-s} b^n = a^{n-s} b^n$. Como $n - s \neq n$, la cadena $ux \notin L$ lo cual es una contradicción. Se concluye entonces que L no puede ser regular.

Tomando $i = 2$ también se llega a una contradicción: por un lado, $uv^2 x \in L$, pero

$$uv^2 x = a^r a^s a^s a^{n-r-s} b^n = a^{r+2s+n-r-s} b^n = a^{n+s} b^n.$$

Como $s \geq 1$, $a^{n+s} b^n$ no está en L.

El argumento anterior también sirve para demostrar que el lenguaje $L = \{a^i b^i : i \geq 1\}$ no es regular.

Ejemplo Demostrar que el lenguaje de los palíndromos sobre $\{a, b\}$ no es un lenguaje regular. Recuérdese que un **palíndromo** es una cadena w tal que $w = w^R$.

Solución: Si L fuera regular, existiría una constante de bombeo n para L. Sea $w = a^n b a^n \in L$. Entonces w se puede descomponer como $w = uvx$, con $|v| \geq 1$, $|uv| \leq n$, y para todo $i \geq 0$, $uv^i x \in L$. Por lo tanto, u y v constan únicamente de $a$es:

$$\begin{aligned} u &= a^r, \qquad \text{para algún } r \geq 0, \\ v &= a^s, \qquad \text{para algún } s \geq 1. \end{aligned}$$

Entonces,

$$x = a^{n-(r+s)} b a^n = a^{n-r-s} b a^n.$$

Tomando $i = 0$, se concluye que $ux \in L$, pero

$$ux = a^r a^{n-r-s} b a^n = a^{n-s} b a^n.$$

Como $s \geq 1$, $a^{n-s} b a^n$ no es un palíndromo. Esta contradicción muestra que L no puede ser regular.

Ejemplo Demostrar que el lenguaje $L = \left\{a^{i^2} : i \geq 0\right\}$ no es regular. L está formado por cadenas de $a$es cuya longitud es un cuadrado perfecto.

Solución: Si L fuera regular, existiría una constante de bombeo n para L. Sea $w = a^{n^2} \in L$. Entonces w se puede descomponer como $w = uvx$, con $|v| \geq 1$, $|uv| \leq n$. Por el lema de bombeo, $uv^2 x \in L$, pero por otro lado,

$$n^2 < n^2 + |v| = |uvx| + |v| = |uv^2x| \leq n^2 + |uv| \leq n^2 + n < (n+1)^2.$$

Esto quiere decir que el número de símbolos de la cadena uv^2x no es un cuadrado perfecto y, por consiguiente, $uv^2x \notin L$. En conclusión, L no es regular.

Ejercicios de la sección 3.1

1. Usar el lema de bombeo para demostrar que los siguientes lenguajes no son regulares:
   (i) $L = \{w \in \{a,b\}^* : w$ tiene el mismo número de $a$es que de $b$es$\}$.
   (ii) $L = \{a^i b a^i : i \geq 1\}$, sobre $\Sigma = \{a,b\}$.
   (iii) $L = \{a^i b^j a^i : i,j \geq 0\}$, sobre $\Sigma = \{a,b\}$.
   (iv) $L = \{0^i 1^{2i} : i \geq 0\}$, sobre $\Sigma = \{0,1\}$.
   (v) $L = \{1^i 01^i 0 : i \geq 1\}$, sobre $\Sigma = \{0,1\}$.
   (vi) $L = \{a^i b^j c^{i+j} : i,j \geq 0\}$, sobre $\Sigma = \{a,b,c\}$.
   (vii) $L = \{a^i b^j : j > i \geq 0\}$, sobre $\Sigma = \{a,b\}$.
   (viii) $L = \{ww : w \in \Sigma^*\}$, siendo $\Sigma = \{a,b\}$.
   (ix) $L = \{ww^R : w \in \Sigma^*\}$, siendo $\Sigma = \{a,b\}$.
   (x) $L = \{a^i : i$ es un número primo$\}$, sobre $\Sigma = \{a\}$.
2. ¿Es $L = \{(ab)^i : i \geq 0\}$ un lenguaje regular?
3. Encontrar la falacia en el siguiente argumento: "Según la propiedad (B) del enunciado del lema de bombeo, se tiene que $uv^ix \in L$ para todo $i \geq 0$. Por consiguiente, L posee infinitas palabras y, en conclusión, todo lenguaje regular es infinito."

3.2. Propiedades de clausura

Las propiedades de clausura afirman que a partir de lenguajes regulares se pueden obtener otros lenguajes regulares por medio de ciertas operaciones entre lenguajes. Es decir, la regularidad es preservada por ciertas operaciones entre lenguajes; en tales casos se dice que *los lenguajes regulares son cerrados bajo las operaciones*.

El siguiente teorema establece que la colección $\mathcal{R} \subseteq \wp(\Sigma^*)$ de los lenguajes regulares sobre un alfabeto Σ es cerrada bajo todas las operaciones booleanas.

3.2.1 Teorema. *Si L, L_1 y L_2 son lenguajes regulares sobre un alfabeto Σ, también lo son:*

(1)	$L_1 \cup L_2$	(unión)
(2)	$L_1 L_2$	(concatenación)
(3)	L^*	(estrella de Kleene)
(4)	L^+	(clausura positiva)
(5)	$\overline{L} = \Sigma^* - L$	(complemento)
(6)	$L_1 \cap L_2$	(intersección)
(7)	$L_1 - L_2$	(diferencia)
(8)	$L_1 \lhd L_2$	(diferencia simétrica)

Demostración:

(1), (2) y (3) se siguen de la definición de los lenguajes regulares.

(4) Por (2), (3) y $L^+ = L^* L$.

(5) Por el Teorema de Kleene y por los teoremas de equivalencia de los modelos AFD, AFN y AFN-λ, existe un AFD $M = (\Sigma, Q, q_0, F, \delta)$ tal que $L(M) = L$. Para construir un AFD que acepte el complemento de L basta intercambiar los estados finales con los no finales. Si M' es el autómata $(\Sigma, Q, q_0, Q - F, \delta)$, entonces $L(M') = \overline{L}$.

(6) Se sigue de (1) y (5) teniendo en cuenta que $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$.

(7) Se sigue de (5) y (6) teniendo en cuenta que $L_1 - L_2 = L_1 \cap \overline{L_2}$.

(8) Puesto que

$$L_1 \lhd L_2 = (L_1 \cup L_2) - (L_1 \cap L_2) = (L_1 - L_2) \cup (L_2 - L_1)$$

el resultado se sigue de (1), (6), (7). □

Recuérdese que un **álgebra booleana de conjuntos** es una familia $\mathcal{A} \subseteq \wp(X)$ cerrada bajo las operaciones de unión, intersección y complemento, tal que $\varnothing \in \mathcal{A}$, $X \in \mathcal{A}$.

3.2.2 Corolario. *La colección $\mathcal{R} \subseteq \wp(\Sigma^*)$ de todos los lenguajes regulares sobre un alfabeto Σ es un álgebra booleana de conjuntos.*

Demostración: Se sigue del teorema 3.2.1 y del hecho de que $\varnothing$ y Σ^* son lenguajes regulares sobre Σ. □

- Hemos visto que un lenguaje finito es regular y que la unión finita de lenguajes regulares es regular. Pero una unión infinita de lenguajes regulares no necesariamente es regular; considérese, por ejemplo,

$$L = \{a^n b^n : n \geq 1\} = \bigcup_{i \geq 1} \{a^i b^i\}.$$

  Cada conjunto $\{a^i b^i\}$ es regular (porque posee sólo una cadena) pero L no lo es.

- Un sublenguaje (subconjunto) de un lenguaje regular no es necesariamente regular, es decir, la familia de los lenguajes regulares no es cerrada para subconjuntos. Dicho de otra forma, un lenguaje regular puede contener sublenguajes no-regulares. Por ejemplo, $L = \{a^n b^n : n \geq 1\}$ es un sublenguaje del lenguaje regular a^*b^*, pero L mismo no es regular.

Las propiedades de clausura permiten concluir, razonando por contradicción, que ciertos lenguajes no son regulares. Esto se ilustra en los siguientes ejemplos en los que se usa el hecho de que los lenguajes $L = \{a^i b^i : i \geq 0\}$ y $L = \{a^i b^i : i \geq 1\}$ no son regulares.

Ejemplo $L = \{a^i b^j : i, j \geq 0,\ i \neq j\}$ no es regular. Si lo fuera, $a^*b^* - L$ también lo sería, pero $a^*b^* - L = \{a^i b^i : i \geq 0\}$.

Ejemplo El lenguaje $L = \{wb^n : w \in \Sigma^*,\ |w| = n,\ n \geq 1\}$ sobre $\Sigma = \{a, b\}$ no es regular. Si L fuera regular, también lo sería $L \cap a^*b^*$ pero $L \cap a^*b^* = \{a^n b^n : n \geq 1\}$.

Ejercicios de la sección 3.2

1. Demostrar que a^*b^* es la unión de dos lenguajes disyuntos no-regulares.

2. Sea L un lenguaje no-regular y N un subconjunto finito de L. Demostrar que $L - N$ tampoco es regular.

3. Demostrar o refutar las siguientes afirmaciones:

   (i) Un lenguaje no-regular debe ser infinito.

   (ii) Si el lenguaje $L_1 \cup L_2$ es regular, también lo son L_1 y L_2.

   (iii) Si los lenguajes L_1 y L_2 no son regulares, el lenguaje $L_1 \cap L_2$ tampoco puede ser regular.

   (iv) Si el lenguaje L^* es regular, también lo es L.

   (v) Si L es regular y N es un subconjunto finito de L, entonces $L - N$ es regular.

(vi) Un lenguaje regular L es infinito si y sólo si en cualquier expresión regular de L aparece por lo menos una $*$.

4. Utilizar las propiedades de clausura para concluir que los siguientes lenguajes no son regulares:

   (i) $L = \{1^i 01^j 0 : i, j \geq 1,\ i \neq j\}$, sobre $\Sigma = \{0, 1\}$. Ayuda: utilizar el ejercicio 1(v) de la sección 3.1.

   (ii) $L = \{uvu^R : u, v \in \{a, b\}^+\}$, sobre $\Sigma = \{a, b\}$. Ayuda: utilizar el ejercicio 1(ii) de la sección 3.1.

   (iii) $L = \{u : |u| \text{ es un cuadrado perfecto}\}$, sobre $\Sigma = \{a, b, c\}$. Ayuda: utilizar el último ejemplo de la sección 3.1.

3.3. Propiedades de clausura para autómatas

Las propiedades de clausura del teorema 3.2.1 se pueden enunciar como procedimientos algorítmicos para la construcción de autómatas finitos.

3.3.1 Teorema. *Sean M, M_1 y M_2 autómatas finitos (ya sean AFD o AFN o AFN-λ) y $L(M) = L$, $L(M_1) = L_1$, $L(M_2) = L_2$. Se pueden construir autómatas finitos que acepten los siguientes lenguajes:*

(1)	$L_1 \cup L_2$.	(5)	$\overline{L} = \Sigma^* - L$.
(2)	$L_1 L_2$.	(6)	$L_1 \cap L_2$.
(3)	L^*.	(7)	$L_1 - L_2$.
(4)	L^+.	(8)	$L_1 \lhd L_2$.

Demostración: La construcción de autómatas para $L_1 \cup L_2$, $L_1 L_2$, L^* y L^+ se presentó en la demostración de la parte I del Teorema de Kleene. En el numeral (5) del teorema 3.2.1 se vio cómo se puede construir un AFD M' que acepte $\overline{L}$ a partir de un AFD M que acepte L.

Los procedimientos de construcción de (1), (2), (3), (4) y (5) se pueden combinar para obtener autómatas que acepten los lenguajes $L_1 \cap L_2$, $L_1 - L_2$ y $L_1 \lhd L_2$. □

Para diseñar un autómata que acepte $L_1 \cap L_2$, según el argumento del teorema 3.3.1, hay que usar la igualdad $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ y los procedimientos para unión, complemento, eliminación de transiciones λ y eliminación del no-determinismo. El siguiente teorema muestra que existe una construcción más sencilla para el caso $L_1 \cap L_2$.

3.3.2 Teorema. *Sean* $M_1 = (\Sigma, Q_1, q_1, F_1, \delta_1)$ *y* $M_2 = (\Sigma, Q_2, q_2, F_2, \delta_2)$ *dos AFD. Entonces el AFD*

$$M = (\Sigma, Q_1 \times Q_2, (q_1, q_2), F_1 \times F_2, \delta)$$

donde

$$\begin{array}{rcl} \delta : (Q_1 \times Q_2) \times \Sigma & \longrightarrow & Q_1 \times Q_2 \\ \delta((q_i, q_j), a) & = & (\delta_1(q_i, a), \delta_2(q_j, a)) \end{array}$$

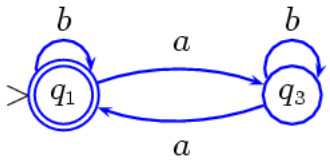
satisface $L(M) = L(M_1) \cap L(M_2)$.

Demostración: Sea $w \in \Sigma^*$. La conclusión del teorema se sigue demostrando primero por inducción sobre w que $\delta((q_1, q_2), w) = (\delta_1(q_1, w), \delta_2(q_2, w))$ para toda cadena $w \in \Sigma^*$ y observando que:

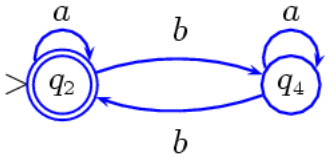
$$\begin{array}{rcl} w \in L(M) & \iff & \delta((q_1, q_2), w) \in F_1 \times F_2 \\ & \iff & (\delta_1(q_1, w), \delta_2(q_2, w)) \in F_1 \times F_2 \\ & \iff & \delta(q_1, w) \in F_1 \quad \& \quad \delta(q_2, w) \in F_2 \\ & \iff & w \in L(M_1) \quad \& \quad w \in L(M_2) \\ & \iff & w \in L(M_1) \cap L(M_2). \qquad \square \end{array}$$

Ejemplo Utilizar el teorema teorema 3.3.2 para construir un AFD que acepte el lenguaje L de todas las cadenas sobre $\Sigma = \{a, b\}$ que tienen un número par de $a$es y un número par de $b$es.

AFD M_1 que acepta las cadenas con un número par de $a$es:



AFD M_2 que acepta las cadenas con un número par de $b$es:

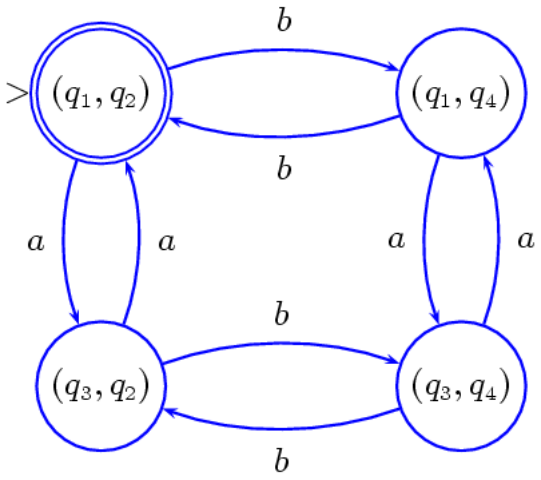


Entonces $L = L(M_1) \cap L(M_2)$. El nuevo autómata tiene 4 estados: (q_1, q_2), (q_1, q_4), (q_3, q_2) y (q_3, q_4); el único estado de aceptación es (q_1, q_2). Su función de transición

δ es

$$\begin{aligned}
\delta((q_1,q_2),a) &= (\delta_1(q_1,a),\delta_2(q_2,a)) = (q_3,q_2)\\
\delta((q_1,q_2),b) &= (\delta_1(q_1,b),\delta_2(q_2,b)) = (q_1,q_4)\\
\delta((q_1,q_4),a) &= (\delta_1(q_1,a),\delta_2(q_4,a)) = (q_3,q_4)\\
\delta((q_1,q_4),b) &= (\delta_1(q_1,b),\delta_2(q_4,b)) = (q_1,q_2)\\
\delta((q_3,q_2),a) &= (\delta_1(q_3,a),\delta_2(q_2,a)) = (q_1,q_2)\\
\delta((q_3,q_2),b) &= (\delta_1(q_3,b),\delta_2(q_2,b)) = (q_3,q_4)\\
\delta((q_3,q_4),a) &= (\delta_1(q_3,a),\delta_2(q_4,a)) = (q_1,q_4)\\
\delta((q_3,q_4),b) &= (\delta_1(q_3,b),\delta_2(q_4,b)) = (q_3,q_2)
\end{aligned}$$

El diagrama de estados del autómata así obtenido es:



Ejercicios de la sección 3.3

1. Utilizar el teorema 3.3.2 para construir AFD que acepten los siguientes lenguajes sobre el alfabeto $\{a,b,c\}$:

   (i) El lenguaje L de todas las cadenas que tienen longitud par y terminan en a.

   (ii) El lenguaje L de todas las cadenas de longitud par que tengan un número impar de $b$es.

   (iii) El lenguaje L de todas las cadenas de longitud impar que tengan un número par de $c$es.

   (iv) El lenguaje L de todas las cadenas de longitud impar que tengan exactamente dos $a$es.

2. Utilizar el procedimiento del teorema 3.3.1 para construir un autómata que acepte el lenguaje L de todas las cadenas sobre $\Sigma = \{a,b\}$ que tienen un número par de $a$es y un número par de $b$es. Compárese con el AFD construido en el último ejemplo de esta sección.

3.4. Homomorfismos

Un homomorfismo es una sustitución h que reemplaza cada símbolo a de un alfabeto de Σ por una cadena $h(a) \in \Gamma^*$, donde Γ es otro alfabeto (por supuesto, Σ y Γ pueden ser el mismo alfabeto). Más precisamente, un homomorfismo es una función $h : \Sigma^* \to \Gamma^*$ tal que $h(\lambda) = \lambda$ y para toda cadena $u = a_1 a_2 \cdots a_n$, con $a_i \in \Sigma$, se tiene

$$h(a_1 a_2 \cdots a_n) = h(a_1)h(a_2) \cdots h(a_n).$$

Un homomorfismo h está completamente determinado por sus imágenes en los símbolos de Σ, es decir, por los valores $h(a)$, con $a \in \Sigma$.

Ejemplo Sean $\Sigma = \{a, b, c\}$, $\Gamma = \{0, 1\}$ y $h : \Sigma^* \to \Gamma^*$ el homomorfismo definido por $h(a) = 0$, $h(b) = 1$ y $h(c) = 010$. El homomorfismo h convierte cadenas de Σ^* en cadenas de Γ^* siguiendo las siguientes reglas: cada a se reemplaza por 0, cada b por 1 y cada c se reemplaza por la cadena 010. Así,

$$h(a^2b^2) = h(aabb) = h(a)h(a)h(b)h(b) = 0011.$$
$$h(acb^2c) = h(a)h(c)h(b)^2h(b) = 001011010.$$

También se deduce fácilmente que $h(a^*b^*c^*) = 0^*1^*(010)^*$.

El siguiente teorema afirma que los homomorfismos preservan lenguajes regulares; dicho de otra manera, la imagen homomorfa de un lenguaje regular es un lenguaje regular.

3.4.1 Teorema. *Sea $h : \Sigma^* \to \Gamma^*$ un homomorfismo.*

(1) *Para cadenas u, $v \in \Sigma^*$ y lenguajes A, $B \subseteq \Sigma^*$ se tiene*

$$h(uv) = h(u)h(v),$$
$$h(A \cup B) = h(A) \cup h(B),$$
$$h(AB) = h(A)h(B),$$
$$h(A^*) = [h(A)]^*.$$

(2) *Si L es un lenguaje regular sobre Σ, entonces $h(L)$ es un lenguaje regular sobre Γ.*

Demostración:

(1) Se sigue directamente de la definición de homomorfismo; los detalles se dejan al lector.

(2) Por inducción sobre el número de operandos (uniones, concatenaciones y estrellas) en la expresión regular que representa a L. La base de la inducción es inmediata ya que $h(\lambda) = \lambda$, para cada $a \in \Sigma$, $h(a)$ es una cadena determinada en Γ^* y $h(\varnothing) = \varnothing$.

La hipótesis de inducción afirma que si L es regular también lo es $h(L)$. Para el paso inductivo se supone, en tres casos, que $L = A \cup B$, $L = AB$ o $L = A^*$. Utilizando la hipótesis de inducción y la parte (1) del presente teorema se llega a la conclusión deseada. □

La parte (2) del teorema 3.4.1 es muy útil para demostrar que ciertos lenguajes no son regulares, razonando por contradicción. En los siguientes ejemplos ilustramos la técnica que se usa en estas situaciones.

Ejemplo Sabiendo que $\{a^i b^i : i \geq 1\}$ no es regular (sección 3.1), podemos concluir que $L = \{0^i 1^i : i \geq 1\}$ tampoco lo es. Razonamos de la siguiente manera: si L fuera regular, lo sería también $h(L)$ donde h es el homomorfismo $h(0) = a$, $h(1) = b$. Pero $h(L) = \{h(0)^i h(1)^i : i \geq 1\} = \{a^i b^i : i \geq 1\}$. Por consiguiente, L no es regular.

Ejemplo $L = \{0^n 2 1^n : n \geq 1\}$ no es regular; si lo fuera, $h(L)$ también lo sería, donde h es el homomorfismo $h(0) = 0$, $h(1) = 1$, $h(2) = \lambda$. Pero $h(L) = \{0^n 1^n : n \geq 1\}$ no es regular, como se dedujo en el ejemplo anterior.

Ejercicios de la sección 3.4

1. Llenar los detalles de la demostración de la parte (1) del teorema 3.4.1.

2. Utilizar homomorfismos y el hecho de que $\{a^i b^i : i \geq 1\}$ y $\{0^i 1^i : i \geq 1\}$ no son regulares para concluir que los siguientes lenguajes tampoco son regulares:

   (i) $L = \{a^i c b^j : i, j \geq 1,\ i \neq j\}$, sobre $\Sigma = \{a, b, c\}$.
   (ii) $L = \{a^i b^i c^i : i \geq 1\}$, sobre $\Sigma = \{a, b, c\}$.
   (iii) $L = \{a^i b^j c^i : i, j \geq 1\}$, sobre $\Sigma = \{a, b, c\}$.
   (iv) $L = \{0^i 1^j 2^k : i, j, k \geq 0,\ i + j = k\}$, sobre $\Sigma = \{a, b, c\}$.

3.5. Imagen inversa de un homomorfismo

Dado un homomorfismo $h : \Sigma^* \to \Gamma^*$ y un lenguaje $B \subseteq \Gamma^*$, la **imagen inversa de** A **por** h es

$$h^{-1}(B) := \{u \in \Sigma^* : h(u) \in B\}.$$

La regularidad es también preservada por imágenes inversas de homomorfismos, tal como lo afirma el siguiente teorema.

3.5.1 Teorema. *Sea $h : \Sigma^* \to \Gamma^*$ un homomorfismo y $B \subseteq \Gamma^*$ un lenguaje regular sobre Γ. La imagen inversa $h^{-1}(B)$ es un lenguaje regular sobre Σ.*

Demostración: Comenzando con un AFD $M = (\Gamma, Q, q_0, F, \delta)$ que acepte a B podemos construir un AFD $M' = (\Sigma, Q, q_0, F, \delta')$ que acepte $h^{-1}(B)$. La función de transición δ' se define mediante $\delta'(q, a) = \delta(q, h(a))$ (aquí se usa la función extendida δ, según la definición 2.5.2. No es difícil demostrar por inducción sobre w que $\delta'(q_0, w) = \delta(q_0, h(w))$, para toda cadena $w \in \Sigma^*$. Como los estados de aceptación de M y M' coinciden, M' acepta a w si y sólo si M acepta a $h(w)$. Por lo tanto,

$$w \in L(M') \iff h(w) \in L(M) = B \iff w \in h^{-1}(B)$$

Esto quiere decir que $L(M') = h^{-1}(B)$. □

Con la ayuda del teorema 3.5.1 y de las demás propiedades de clausura también podemos concluir que ciertos lenguajes no son regulares.

Ejemplo El lenguaje $L = \{0^n 1 0^n : n \geq 1\}$ no es regular ya que si lo fuera, también lo sería $h^{-1}(L)$ donde h es el homomorfismo $h(0) = h(1) = 0$, $h(2) = 1$. Pero

$$h^{-1}(L) = \Big\{ \{0,1\}^n 2 \{0,1\}^n : n \geq 1 \Big\}.$$

Entonces $h^{-1}(L) \cap 0^*21^*$ sería regular; este último lenguaje es igual a $\{0^n 21^n : n \geq 1\}$. Finalmente, por medio del homomorfismo $g(0) = 0$, $g(1) = 1$, $g(2) = \lambda$ se concluiría que $g\big(\{0^n 21^n : n \geq 1\}\big) = \{0^n 1^n : n \geq 1\}$ es regular, lo cual sabemos que no es cierto.

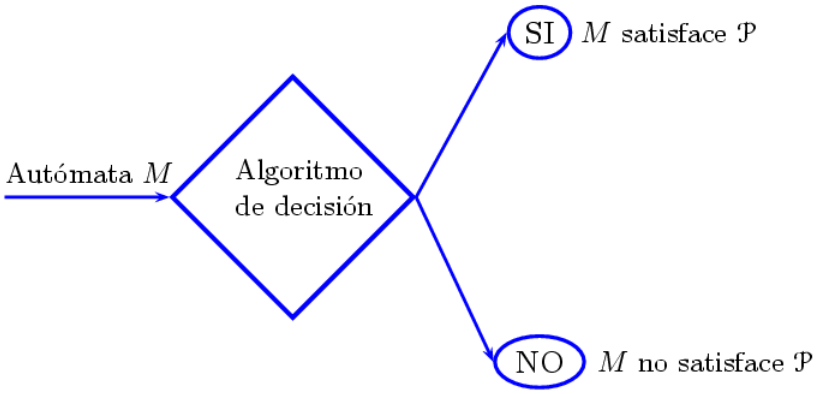
Ejercicios de la sección 3.5

Por medio de un razonamiento similar al del ejemplo de esta sección, demostrar que los siguientes lenguajes sobre $\Sigma = \{0, 1\}$ no son regulares:

1. $L = \{ww : w \in \Sigma^*\}$. Ayuda: intersectar primero L con 0^*10^*1.
2. $L = \{ww^R : w \in \Sigma^*\}$. Ayuda: intersectar primero L con 0^*110^*.

3.6. Algoritmos de decisión

Dada una propiedad $\mathcal{P}$, referente a autómatas sobre un alfabeto Σ, un **problema de decisión para $\mathcal{P}$** consiste en buscar un algoritmo[1], aplicable a un autómata arbitrario M, que responda SI o NO a la pregunta: ¿satisface M la propiedad $\mathcal{P}$? El siguiente diagrama ilustra la situación.



Si existe un algoritmo de decisión, se dice que el problema $\mathcal{P}$ es **decidible** o **resoluble**; en caso contrario, el problema $\mathcal{P}$ es **indecidible** o **irresoluble**.

Es importante tener presente que, para que un problema $\mathcal{P}$ sea decidible, no basta responder SI o NO a la pregunta "¿satisface M la propiedad $\mathcal{P}$?" para uno o varios autómatas aislados M; es necesario exhibir un algoritmo aplicable a *todos* los autómatas. Es posible que en algunos casos concretos se pueda decidir afirmativa o negativamente sobre la satisfabilidad de una propiedad $\mathcal{P}$ y, sin embargo, el problema $\mathcal{P}$ sea indecidible.

El primer problema de decisión que consideraremos es el problema de decidir si un autómata acepta o no *alguna* cadena.

3.6.1 Teorema. *Sea Σ un alfabeto dado. Existe un algoritmo para el siguiente problema de decisión referente a autómatas sobre Σ:*

> *Dado un autómata (AFD o AFN o AFN-λ) M, ¿Es $L(M) \neq \varnothing$? (es decir, ¿acepta M alguna cadena?).*

Demostración: Podemos diseñar un algoritmo sencillo para encontrar los estados que son accesibles (o alcanzables) desde el estado inicial de M, es decir, los estados para los cuales existen trayectorias desde el estado inicial. Si algún estado de aceptación es alcanzable, se tendrá $L(M) \neq \varnothing$; en caso contrario $L(M) = \varnothing$. En el siguiente recuadro aparece el algoritmo para encontrar el conjunto **ALC** de estados alcanzables:

[1] La noción de algoritmo aquí considerada es la corriente: un algoritmo es un procedimiento claro y preciso para efectuar una determinada operación secuencialmente, en uno o más pasos.

INICIALIZAR:

ALC := $\{q_0\}$, donde q_0 es el estado inicial de M.

REPETIR:

Para cada $q \in$ **ALC** buscar los arcos que salen de q y añadir a **ALC** los estados p para los cuales haya un arco de q a p con cualquier etiqueta (puede ser λ).

HASTA:

No se añaden nuevos estados a **ALC**.

Claramente, el autómata M acepta alguna cadena (o sea, $L(M) \neq \varnothing$) si y solo si **ALC** contiene algún estado de aceptación. □

3.6.2 Corolario. *Sea Σ un alfabeto dado. Existen algoritmos para los siguientes problemas de decisión referentes a autómatas sobre Σ:*

(1) *Dados dos autómatas M_1 y M_2 (AFD o AFN o AFN-λ), ¿$L(M_1) \subseteq L(M_2)$?*

(2) *Dados dos autómatas M_1 y M_2 (AFD o AFN o AFN-λ), ¿$L(M_1) = L(M_2)$?*

Demostración:

(1) Sea $L_1 = L(M_1)$ y $L_2 = L(M_2)$. Se tiene

$$L_1 \subseteq L_2 \iff L_1 - L_2 = \varnothing.$$

Algoritmo: construir un AFD M' que acepte el lenguaje $L_1 - L_2$ (esto se puede hacer en razón de la parte (7) del teorema 3.3.1). Utilizar luego el procedimiento de la parte (1) para decidir si $L_1 - L_2$ es o no vacío.

(2) $L(M_1) = L(M_2) \iff L(M_1) \subseteq L(M_2)$ y $L(M_2) \subseteq L(M_1)$. Por lo tanto, basta aplicar dos veces el algoritmo de la parte (1). También se puede encontrar un algoritmo de decisión teniendo en cuenta que $L_1 = L_2 \iff L_1 \lhd L_2 = \varnothing$ junto con la parte (8) del teorema 3.3.1. □

El argumento utilizado en la demostración del lema de bombeo sirve para establecer un criterio que permite decidir si el lenguaje aceptado por un autómata es o no infinito. El criterio aparece en el siguiente teorema.

3.6.3 Teorema. *Sea M un AFD con n estados y sea $L = L(M)$. L es infinito si y solo si M acepta una cadena w tal que $n \leq |w| < 2n$.*

Demostración: Si $w \in M$ y $n \leq |w| < 2n$, entonces por la demostración del lema de bombeo, w se puede descomponer como $w = uvx$, donde $|uv| \leq n$, $v \neq \lambda$. M acepta infinitas cadenas: $uv^i x$ para todo $i \geq 0$.

Recíprocamente, si L es infinito, existe $w \in L$ con $|w| \geq n$. Por la demostración del lema de bombeo, $w = uvx$ donde $|uv| \leq n$, $v \neq \lambda$. Si $|w| < 2n$ la demostración está terminada. Si $|w| \geq 2n$, puesto que $|v| \leq |uv| \leq n$, se tendrá $|ux| \geq n$ y $ux \in L$. De nuevo, si $|ux| < 2n$, la demostración termina; en caso contrario, se prosigue de esta forma hasta encontrar una cadena en L cuya longitud ℓ satisfaga $n \leq \ell < 2n$. □

3.6.4 Corolario. *Sea Σ un alfabeto dado. Existe un algoritmo para el siguiente problema de decisión referente a autómatas sobre Σ:*

Dado un autómata M (AFD o AFN o AFN-λ), ¿Es $L(M)$ infinito?

Demostración: Algoritmo: construir un AFD M' que acepte el lenguaje $L(M)$ y chequear la aceptación o rechazo de todas las cadenas $w \in \Sigma^*$ tales que $n \leq |w| < 2n$, donde $n = \#$ de estados de M'. □

Hemos trabajado con problemas de decisión referentes a autómatas, pero también podemos considerar problemas sobre lenguajes regulares. Dada una propiedad $\mathcal{P}$, referente a lenguajes regulares sobre un alfabeto Σ, un problema de decisión para $\mathcal{P}$ consiste en buscar un algoritmo, aplicable a todo lenguaje regular L (es decir, a toda expresión regular R), que responda SI o NO a la pregunta: ¿satisface L la propiedad $\mathcal{P}$? Puesto que conocemos algoritmos de conversión entre la representación por expresiones regulares y la representación por autómatas, los problemas decidibles sobre autómatas corresponden a problemas decidibles sobre lenguajes regulares y viceversa. Así por ejemplo, en razón del Corolario 3.6.4, el siguiente problema es decidible: "Dada una expresión regular R, ¿es $L(R)$ infinito?"

Ejercicios de la sección 3.6

1. Sea Σ un alfabeto dado. Encontrar algoritmos para los siguientes problemas de decisión referentes a autómatas sobre Σ:

   (i) Dado un autómata M (AFD o AFN o AFN-λ), ¿es $L(M) = \Sigma^*$?

   (ii) Dado un autómata M (AFD o AFN o AFN-λ), ¿acepta M alguna cadena de longitud 1250?

   (iii) Dado un autómata M (AFD o AFN o AFN-λ), ¿acepta M alguna cadena de longitud mayor que 1250?

   (iv) Dado un autómata M (AFD o AFN o AFN-λ), ¿acepta M *todas* las cadenas de longitud impar?

(v) Dados dos autómatas M_1 y M_2 (AFD o AFN o AFN-λ), ¿existe alguna cadena que no sea aceptada por ninguno de los autómatas M_1 y M_2?

(vi) Dado un autómata M (AFD o AFN o AFN-λ), ¿Es $L(M)$ cofinito? (Un conjunto es **cofinito** si su complemento es finito).

2. Encontrar algoritmos de decisión, que no utilicen autómatas, para resolver los siguientes problemas:

(i) Dada una expresión regular R, ¿es $L(R) \neq \varnothing$? Ayuda: puesto que las expresiones regulares se definen recursivamente, el algoritmo requiere descomponer la expresión R y utilizar criterios de vacuidad para la unión, la concatenación y la estrella de Kleene.

(ii) Dada una expresión regular R, ¿contiene $L(R)$ por lo menos 150 cadenas?

✎ Al considerar problemas de decisión lo importante es la *existencia* o no de algoritmos de decisión, no la *eficiencia* de los mismos.

✎ En el Capítulo 6 se mostrará que existen problemas indecidibles, es decir, problemas para los cuales no hay algoritmos de decisión.