Capítulo 8: Propiedades de Lenguajes Regulares

8.1. Identificación de lenguajes no regulares

8.1.1. Lema de Bobeo

8.1.2. Aplicaciones del lema de bombeo

8.2. Propiedades de Cierre

8.2.1. Unión, Concatenación, Clausura

8.2.2. Respecto a otras operaciones

8.3. Algoritmos de decisión

8.3.1. ¿Es L finito?

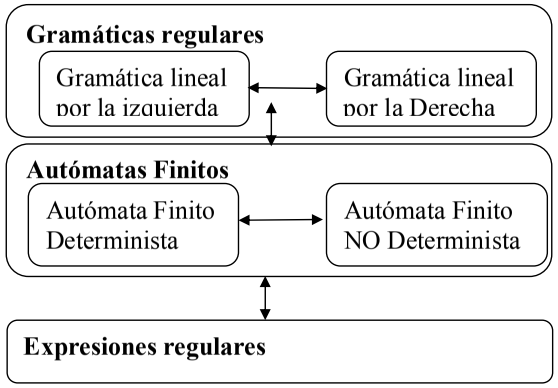
8.3.2. ¿Es L vacío?

8.3.3. Pertenencia de una palabra

8.3.4. ¿L1=L2?

Definición (Lenguaje regular):
Un ***lenguaje*** *L* se denomina ***regular*** si y sólo si existe un autómata finito determinista *A* tal que *L*=L(*A*).

Se pueden dar definiciones equivalentes, utilizando expresiones regulares o gramáticas regulares, ya que estos formalismos definen la misma clase de lenguajes:



Preguntas:

- ¿Son los lenguajes que se obtienen al aplicar determinadas operaciones a lenguajes regulares también regulares?
- ¿Cómo se puede decidir (de forma automática) si un lenguaje regular tiene ciertas propiedades: si es finito, vacío, o infinito?
- ¿Existe algún método para saber si un lenguaje dado es regular o no?

8.1 Identificación de lenguajes no regulares

- No todos los lenguajes son regulares.
- Los autómatas finitos sólo tienen una capacidad limitada en el proceso de identificación de palabras:
  - tienen un número limitado de estados
  - la única información de la que disponen está en la estructura de este número finito de estados

Ejemplo:
El lenguaje $L=\{a^nb^n|n>=0\}$ no es regular.

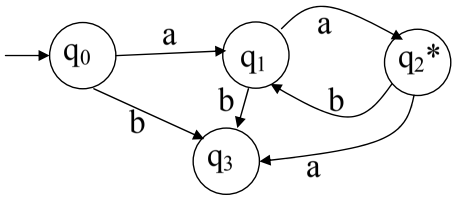
Ilustración intuitiva de este hecho:
- Se requería un autómata reconocedor con infinitos estados para aceptar este lenguaje, por lo que no puede ser regular.
- Construir de manera incremental los AFD para los siguientes lenguajes:
  - $L_0=\{\lambda\}$
  - $L_1=\{\lambda,ab\}$
  - $L_2=\{\lambda,ab,aabb\}$
  - $L_3=\{\lambda,ab,aabb,aaabbb\}$
  - ...
- Se puede observar que el número de estados de los autómatas aumenta para cada uno de estos lenguajes.
- No es posible encontrar autómatas finitos deterministas, tales que el número de estados no aumente de un lenguaje L_i al siguiente L_{i+1}.
- Eso indica que el autómata que reconoce el lenguaje L requiere un número infinito de estados.

8.1.1 Lema de Bombeo

¿Que características tiene un lenguaje para ser regular/no regluar?

Lenguajes finitos:
Todos los lenguajes finitos son regulares.
(Demostración: **Ejercicio)**

Lenguajes infinitos:
No todos los lenguajes infinitos son regulares.
Consideramos el lenguaje regular infinito: $L=\{a(ab)^na|n\geq 0\}$.
- El siguiente AFD A reconoce L:



- A reconoce las palabras aa, aaba, aababa, ...
- Se puede observar lo siguiente:
  - La palabra x=aaba se puede descomponer en tres cadenas: x=u.v.w=a.ab.a (con $v\neq\lambda$) y todas las palabras $u.v^i.w$, es decir, $a.(ab)^i.a$, (con $i\geq 0$) pertenecen a L.
  - Este hecho se repite para todas $x\in L$ con $|x|\geq 4$.
    Ejemplo: aababaaba=a.ab.ababa $\Rightarrow$ $a.(ab)^i.ababa\in L$ ($i\geq 0$)
  - Las palabras $x\in L$ con $|x|<4$ no tienen esta propiedad.
    Ejemplo: aa
    Intento 1: $aa=a.a.\lambda \Rightarrow a.a^i.\lambda\notin L$
    Intento 2: $aa=\lambda.a.a \Rightarrow \lambda.a^i.a\notin L$
    Intento 3: $aa=\lambda.aa.\lambda \Rightarrow \lambda.(aa)^i.\lambda\notin L$

La causa de este comportamiento es la siguiente:

- Al aceptar una palabra x "suficientemente grande" (como aababa), el autómata necesariamente tiene que entrar en un bucle $q_0 \rightarrow ... \rightarrow \mathbf{q_i \rightarrow ... \rightarrow q_i} \rightarrow ... \rightarrow q_f$.

- Por tanto, se puede descomponer x en tres partes u.v.w, donde:
  - u la subcadena que se acepta antes del bucle (de q_0 a q_i)
  - v la subcadena que se acepta en el bucle (de q_i a q_i)
  - w la subcadena que se acepta después del bucle (de q_i a q_f)

- El autómata, al aceptar palabras similar a x, podría recorrer el bucle 0,1,2,3,... veces.

- Por tanto, el autómata también acepta las cadenas que se obtienen a partir de x=u.v.w eliminando la subcadena v o "bombeándola" varias veces.

La propiedad de "bombeo" es común a todos los lenguajes regulares infinitos y se describe de forma general en el "Lema de Bombeo".

Teorema 10 (Lema de Bombeo):
Sea L un lenguaje regular infinito.
Entonces **existe** una constante n tal que **para toda** palabra $x \in L$ con $|x| \geq n$ **existe** una descomposición de x en tres subcadenas, $x=u.v.w$ verificando las siguientes propiedades:

a) $|u.v| \leq n$
b) $|v|>0$
c) para todo $i \geq 0$: $u.v^i.w \in L$.

Demostración:

Como L es regular, existe un AFD mínimo $A=(Q, \Sigma, q_0, f, F)$ con $L(A)=L$. Definimos $n=|Q|$.
Sea $x=a_1a_2a_3...a_m \in L$ una palabra arbitraria del lenguaje, con $m \geq n$.

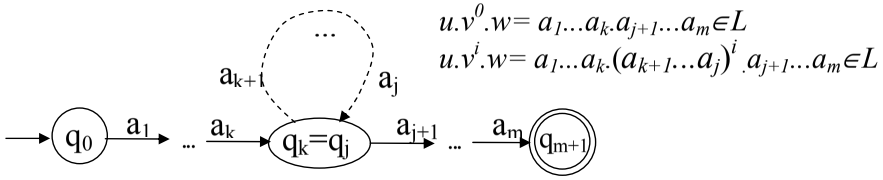
- Dado que $x \in L$, A acepta x. Sea la secuencia de transiciones que recorre A para aceptar x la siguiente:
  $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} ... \xrightarrow{a_m} q_m$ (con $q_i \in Q$, $q_m \in F$)
- Esta cadena tiene $m+1$ estados (pues x tiene m letras).
- Como A sólo tiene n estados distintos y $m+1>n$, se sigue que en la secuencia existe por lo menos un estado repetido:
  $q_0 \xrightarrow{a_1} ... \xrightarrow{a_k} q_k \xrightarrow{a_{k+1}} ... \xrightarrow{a_i} q_j \xrightarrow{a_{i+1}} ... \xrightarrow{a_m} q_m$ con $q_k=q_j$.
  (sean q_k /q_j el primer estado de la secuencia que se repite)

- x puede descomponerse en tres subcadenas:
  $x=u\,.\,v\,.\,w = a_1...a_k\,.\,a_{k+1}...a_j\,.\,a_{j+1}...a_m$
  que cumplen lo siguiente:

a) $|u.v|\leq n$:
   - La primera repetición de un estado ocurre obviamente en los n+1 primeros estados.
   - Por tanto, $u.v=a_1...a_k\,.\,a_{k+1}...a_j$ tiene como mucho n símbolos.

b) $|v|>0$:
   - Entre los dos estados repetidos hay por lo menos una transición: $...q_k \xrightarrow{a_{k+1}} ... \xrightarrow{a_j} q_j....$
   - Por tanto, $v= a_{k+1}...a_j$ tiene por lo menos un símbolo.

c) $u.v^i.w \in L$ para $i\geq 0$: se observa del grafo de transición



$u.v^0.w= a_1...a_k.a_{j+1}...a_m \in L$
$u.v^i.w= a_1...a_k.(a_{k+1}...a_j)^i\,.a_{j+1}...a_m \in L$

Lo que demuestra el teorema.

8.1.2 Aplicación del Lema de Bombeo

¿Para que sirve el lema de bombeo?

$\Rightarrow$ Para demostrar que ciertos lenguajes no son regulares.

Observación:
El lema no sirve para demostrar que un lenguaje es regular. (Proporciona una condición necesaria pero no suficiente.)

Para demostrar que un lenguaje **no** es regular basta probar que **no** cumple el lema de bombeo. Estas demostraciones son siempre por contradicción.

Los pasos son los siguientes:

1. Se supone que L es regular y, por tanto, debe cumplir el lema de bombeo.
2. Se elige un valor genérico n (**no especificado**) para la constante.
3. Se selecciona una palabra $x\in L$ (en función de n) con $|x|\geq n$
4. Se descompone x en $u.v.w$, tal que $|u.v|\leq n$ y $|v|>0$, de **todas** las formas posibles.
5. Se demuestra que para todas estas descomposiciones existe siempre i tal que $u.v^i.w \notin L$.
6. Si se consigue demostrar que para todas estas descomposiciones existe siempre i, tal que $u.v^i.w \notin L$, L no cumple el lema y, por tanto, no es regular.

Ejemplo:
$L=\{a^m b^{2m} | \ m \in N, m \geq 0\}$

1. Supongamos que L sea regular.
2. *Sea* $n=N$ la constante del lema
3. Seleccionamos la palabra x con $x=a^N b^{2N} \in L$ $(|x|=3N>n)$
4. Descomponemos $x=u.v.w$ de todas las formas posibles, tales que $|u.v| \leq n$ y $|v|>0$:
   - Todas las descomposiciones tienen la forma:
     $a^k \cdot a^j \cdot a^{N-k-j} b^{2N}$ con $k \geq 0$; $j \geq 1$
5. consideramos la palabra $u.v^0.w$ de estas composiciones:
   $u.v^0.w = u.w = a^k.a^{N-k-j} b^{2N} = a^{N-j} b^{2N}$ con $k \geq 0$ y $j \geq 1$
   Por tanto, $u.v^0.w \notin L$ (ya que $2(N-j) \neq 2N$).
6. Por tanto, L no cumple el lema y no puede ser regular.

Ejercicios:
1. ¿Es el lenguaje $L=\{yy^{-1} | \ y \in \{a,b\}^*\}$ regular?
2. Sea el lenguaje $L=\{a^n b^m c^m | \ m,n \geq 1\}$. Demuestra que L no cumple el lema de bombeo.
3. Sea el lenguaje $L=\{a^n b^m c^m | \ m,n \geq 1\} \cup \{b^m c^n | m,n \geq 0\}$. Cumple L el lema de bombeo. ¿Es L un lenguaje regular?
4. ¿Es el lenguaje $L=\{a^m | \ m=\Sigma^n_{i=0}(i) \ , n \geq 0\}$ regular?

8.2 Propiedades de Cierre

8.2.1 Unión, concatenación y cierre

Teorema 1:
Dados dos lenguajes regulares L_1 y L_2, los lenguajes $L_1 \cup L_2$, $L_1 L_2$ y L_1^* también son regulares.

Demostración:
Si L_1 y L_2 son regulares, existen dos expresiones regulares r_1 y r_2 tales que:

$$L_1=L(r_1) \text{ y } L_2=L(r_2)$$

Por definición de las operaciones de cierre, concatenación y suma de expresiones regulares:

$$L_1 \cup L_2 = L(r_1) \cup L(r_2) = L(r_1+r_2)$$
$$L_1 L_2 = L(r_1)L(r_2) = L(r_1 r_2)$$
$$L_1^* = L(r_1)^* = L(r_1^*)$$

Es decir, existen expresiones regulares que describen los lenguajes $L_1 \cup L_2$, $L_1 L_2$ y L_1^* y, por tanto, estos lenguajes son regulares.

8.2.2 Otras operaciones

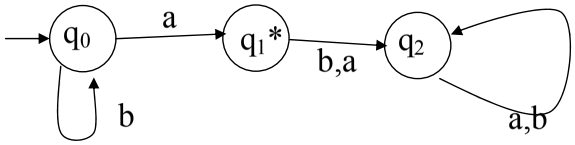
Teorema 2:
Dado un lenguaje regular L definido, su complemento $\overline{L}$ también es regular.

Demostración:
Dado que L es regular, existe un AFD $A=(Q, \Sigma, q_0, f, F)$, tal que $L(A) = L$.

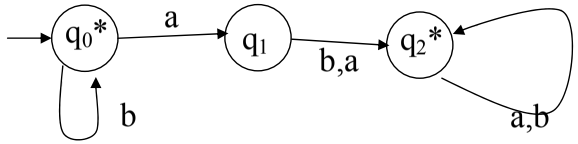
Consideremos el autómata $A'=(Q, \Sigma, q_0, f, Q-F)$.
Este autómata A' acepta exactamente todas las palabras del lenguaje universal Σ^* que el autómata A rechaza. Por tanto:
$L(A') = \Sigma^* - L(A) = \Sigma^* - L = \overline{L}$.

Ejemplo: $\Sigma=\{a,b\}$, $L=L(b^*a)=\{b^n a | n\geq 0\}$

A:



A':



$L(A')=\{b^n | n\geq 0\} \cup \{b^n ax | n\geq 0, x\in \{a,b\}^*, |x|>0\}$
$= \overline{L}$

Teorema 3:
Dados dos lenguajes regulares L_1 y L_2 definidos, el lenguaje $L_1 \cap L_2$ también es regular.

Demostración:
a) No constructiva:

Por las leyes de Morgan:
$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$ (La operación de intersección se puede reducir a las operaciones complemento y unión.)

b) Constructiva:
Dado que L_1 y L_2 son regulares, existen dos AFDs:
$A_1=(Q_1, \Sigma, q_{01}, f_1, F_1)$ y
$A_2=(Q_2, \Sigma, q_{02}, f_2, F_2)$
tales que $L(A_1)=L_1$ y $L(A_2)=L_2$.

Considérese el autómata:
$A=(Q_1 \times Q_2, \Sigma, (q_{01},q_{02}), f, F_1 \times F_2)$
donde: $f((p,q),a)=(f_1(p,a), f_2(q,a))$

Evidentemente, $L(A)= L_1 \cap L_2$, por lo que $L_1 \cap L_2$ es un lenguaje regular.

Ejercicio:
Sean
$L_1=\{ax | x\in \{a,b\}^*$ y x tiene por lo menos una subcadena "ab"$\}$ y
$L_2=\{a(ab)^n | n\geq 0\}$
Aplicando la demostración del último teorema, construye un AFD para el lenguaje $L_1 \cap L_2$.

Teorema 4:
Dado un lenguaje regular L, su lenguaje inverso L^{-1} también es regular.

Demostración:

Dado que L es regular, existe una expresión regular r tal que $L(r)=L$. Para demostrar que L^{-1}es regular construiremos una expresión regular r^{-1} tal que $L(r^{-1})=L^{-1}$ a partir de r:

Si $r = \varnothing$, o $r=a$ o $r=\lambda$, entonces $r^{-1}=r$
Si $r=r_1+r_2$, entonces $r^{-1}= r_1^{-1} +r_2^{-1}$
Si $r=r_1r_2$, entonces $r^{-1}= r_2^{-1}r_1^{-1}$
Si $r=r_1^*$, entonces $r^{-1}=(r_1^{-1})^*$

Por construcción, $L(r^{-1})=L^{-1}$.

Teorema 5:
Dados dos lenguajes regular L_1 y L_2, el lenguaje diferencia L_1-L_2 también es regular.

Demostración:

Por las leyes de Morgan: $L_1-L_2=L_1\cap\overline{L_2}$. Ya se ha demostrado que tanto el complemento de un lenguaje regular como la unión de dos lenguajes regulares son regulares.

Definición (homomorfismo):
Sean Σ y Γ alfabetos. Se llama ***homomorfismo*** a una función $h:\Sigma\rightarrow\Gamma^*$, tal que asigna una palabra de Γ^* a cada símbolo de Σ.

Esta definición se puede extender para definir la función h para palabras de Σ:
$w=a_1a_2...a_n\in\Sigma^* \quad \Rightarrow \quad h(w)=h(a_1)h(a_2)...h(a_n)\in\Gamma^*$

Definición (imagen homomórfica):
Dado un lenguaje $L\subseteq\Sigma^*$ y dado un homomorfismo $h:\Sigma\rightarrow\Gamma^*$, se llama ***imagen homomórfica*** de L respecto a h al lenguaje definido de la siguiente manera:

$$h(L)=\{h(w) \mid w\in L\}.$$

Teorema 6:
Sean Σ y Γ dos alfabetos y sea L un lenguaje regular definido sobre Σ. *Sea* $h:\Sigma\rightarrow\Gamma^*$ un homomorfismo. Entonces la imagen homomórfica de L respecto a h también es regular.

Demostración:
Dado que L es regular, existe una expresión regular r tal que $L(r)=L$. Para demostrar que $h(L)$ es regular construiremos recursivamente una expresión regular r' tal que $L(r')=h(L)$ a partir de r:

Si $r = \varnothing$ o $r=\lambda$, entonces $r'=r$
Si $r = a$, $a\in\Sigma$, entonces $r'=(h(a))$
Si $r=r_1+r_2$, entonces $r'= r_1'+r_2'$
Si $r=r_1r_2$, entonces $r'= r_1'r_2'$
Si $r=r_1^*$, entonces $r'=(r_1')^*$

Es decir, se sustituye cada símbolo $a\in\Sigma$ en r por la cadena $h(a)$.
r' es una expresión regular que representa $h(L)$, por lo que $h(L)$ es regular.

Ejemplo:
$\Sigma=\{a,b\}$, $L=L(b^*ac^*)$ y $h(a)=abc$, $h(b)=bac$, $h(c)=cab$
$\Rightarrow h(L)=L((bac)^*(abc)(cab)^*)$

8.3 Algoritmos de Decisión

8.3.1 ¿Es L vacío?

Teorema 7:
Dado un lenguaje regular L, existe un algoritmo para decidir si dicho lenguaje es vacío.

Demostración:
Considérese el autómata finito determinista mínimo que acepta L.
L es vacío si y sólo si el AFD mínimo no tiene estados finales.

8.3.2 ¿Es L infinito?

Teorema 8:
Dado un lenguaje regular L, existe un algoritmo para decidir si dicho lenguaje es infinito.

Demostración:
Considérese el diagrama de transiciones del AFD A mínimo tal que $L(A)=L$. L es infinito si y sólo si existe un ciclo en un nodo de este autómata.

8.3.3 ¿w pertenece a L?

Teorema 9:
Dado un lenguaje regular L, existe un algoritmo para decidir si una palabra w pertenece al lenguaje.

Demostración:
Considérese el diagrama de transiciones del AFD A tal que $L(A)=L$. Simulamos el funcionamiento del autómata tomando como entrada la palabra w. Si el estado en el que termina la simulación es un estado final, entonces $w \in L$; en otro caso, w no pertenece a L.

8.3.4 ¿ $L_1=L_2$?

Teorema 10:
Dados dos lenguajes regulares L_1 y L_2, existe un algoritmo para decidir si dichos lenguajes son equivalentes, es decir, si $L_1= L_2$.

Demostración:

a) Dado que L_1 y L_2 son regulares, existen dos AFD mínimos A_1 y A_2 tales que $L(A_1)=L_1$ y $L(A_2)=L_2$. Los dos lenguajes son equivalentes si A_1 y A_2 son isomorfos (iguales salvo renombramiento de estados).

b) Construye un AFD A para $L_3=(L_1 \cap \overline{L_2}) \cup (\overline{L_1} \cap L_2)$.
$L_1=L_2$, si y solo si L_3 es vacío.