



LENGUAJE = Conjunto = ÁLGEBRA.

- Clasificación de Lenguajes y Autómatas.

LENGUAJES	AUTÓMATAS
regulares	finitos
independientes del contexto	de pila
dependientes del contexto	con dos pilas
estructura de frase	maquinas de Turing

→ la mayoría de las máquinas de caja pertenecen ahí.

SECCIÓN 1.5, página 84 ⇒ Capítulo 1

página 143 ⇒ Capítulo 5

def: Un alfabeto es un conjunto finito → Si tenemos un alfabeto podemos definir una palabra.

Ejemplo: $I = \{a, b, \dots, z\}$ $I = \{\text{pape}, \text{juan}\}$

$I = \{\emptyset, \beta, \dots, \Omega\}$ ↳ si ese es mi alfabeto pape no es una palabra.

def: Una palabra o cadena de un alfabeto I , es cualquier $w \in I^* = I^+ \cup \{\epsilon\}$

def: Supongamos la operación interna sobre I^*

$x, y \in I^* \rightsquigarrow xy \dots$ la concatenación.

Palabra vacía

(I^*, \cdot) es un monóide (intern + asociatividad)

+ Elemento neutro.

def: Sean $X, Y \subseteq I^*$, podemos definir $X \cdot Y = \{x \cdot y / x \in X, y \in Y\}$

" " también $X^\emptyset := \{\epsilon\}$

" " también $X^{i+1} := X^i \cdot X \forall i > 0$

" " también $X^* = UX^i$ (cierre de flecha)

" " también $X^+ = UX^i$ (cierre transitivo)

Proposición: Sea $X \subseteq I^*$, con I alfabeto

entonces: a) (X^+, \cdot) es un semigrupo (operación interna + asociatividad)

b) (X^*, \cdot) es un monóide

asociativa + elemento neutro

def: Una gramática es una tupla $\mathcal{G} = (N, I, P, S)$ donde:

N es un conjunto finito variable (o categorías sintácticas o no terminales).

I es un alfabeto cuyos elementos se denominan símbolos terminales.

S es un elemento de N , se denomina axioma o raíz o símbolo principal.

P es un conjunto finito, regla o producción o cláusula.

De hecho, elementos de $(N \cup I)^*$ $N(N \cup I)^* \times (N \cup I)^*$

denotados como pares $(\alpha, \beta) \wedge \alpha \rightarrow \beta \wedge \alpha ::= \beta \wedge \alpha :- \beta$

La gramática está formada por las reglas.

Ejemplo:

$$\left. \begin{array}{l} S \rightarrow S + S \\ S \rightarrow S * S \\ S \rightarrow (S) \\ S \rightarrow \text{número} \end{array} \right\} = \mathcal{G}$$

$N = \{S\} \Rightarrow$ simplemente porque S es mayorizada
 S es el axioma \Rightarrow solo puede ser S porque es el único elemento
 $\Sigma = \{+, (*), (,), \text{número}\} \Rightarrow$ lo que no es mayorizada

dicimos que es una variable porque se descompone
 Se lee que S se puede descomponer en $S + S$ (lectura descendente)

Se lee que si tanto $S + S$, podemos generar S (lectura ascendente). descomponiendo
 → decidido como se combinan estos elementos que son terminales

y/o 2

def: Decimos que $\alpha \beta \gamma$ deriva indirectamente $\alpha' \beta' \gamma'$ si:

- a) $\beta \Rightarrow S_1 \Rightarrow S_2 \dots \Rightarrow S_n = \gamma$ (noción: $\alpha \beta \gamma \xrightarrow{*} \alpha' \beta' \gamma'$) (1 o más derivaciones directas concatenadas).
- b) $\beta \Rightarrow S \wedge \beta' \Rightarrow S$ (noción: $\alpha \beta \gamma \xrightarrow{*} \alpha' \beta' \gamma'$) (cero o más derivaciones directas concatenadas).

En particular, una derivación directa es también indirecta.

def: sea $\mathcal{G} = (N, \Sigma, P, S)$ gramática \Rightarrow Definimos el lenguaje generado por \mathcal{G} .

$$L(\mathcal{G}) := \{ x \in \Sigma^* / S^* \Rightarrow x \} \quad \text{Ejemplo: } 1+2 \in L(\mathcal{G}) \text{ porque } S \Rightarrow S + S \Rightarrow 1 + S \Rightarrow 1 + 2 \text{ esto va concatenando.}$$

Por definición

JERARQUÍA DE COMPLEJIDAD

$$\Rightarrow \begin{array}{l} \alpha \rightarrow \beta \\ S \rightarrow S + S \\ S + S \Rightarrow S * S \end{array}$$

$$\begin{array}{l} S \rightarrow S + S \\ aS \rightarrow S + S \\ bS \rightarrow S * S \end{array}$$

$$\begin{array}{ll} S & S \\ \Delta & \Delta \\ S + S & S * S \end{array}$$

Depende del contexto

GRAMÁTICAS REGULARES.

def: Sea $\mathcal{G} = (N, \Sigma, P, S)$ una gramática.

Decimos que es regular por la derecha (por la izquierda).

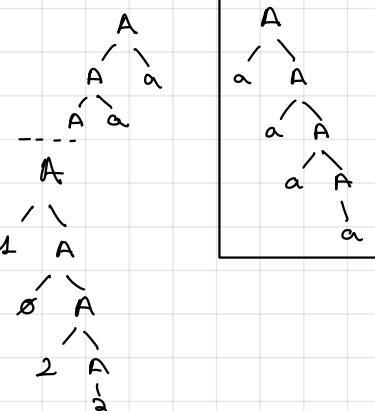
se toma regla en P es de la forma:

- a) $A \rightarrow a$ $A \rightarrow a$
- b) $A \rightarrow a\beta$ $A \rightarrow Ba$

\Rightarrow Una gramática es regular si es regular izquierda ó derecha.

Dado que en este caso las propiedades de las reg. izquierda y derecha son las mismas transformaciones por defecto para las gramáticas reg. derecha.

Ejemplo: $A \rightarrow \text{dígito}$
 $A \rightarrow \text{dígito } A$



Ejemplo: en general el lexico de los lenguajes de programación es regular

$$\begin{aligned} G &= (N, \Sigma, P, S) \\ G &= (V, \Sigma, P, S) \end{aligned}$$

$$\left\{ \begin{array}{l} \text{número, } +, \times \\ \text{simbolo de paréntesis} \end{array} \right. \quad \left. \begin{array}{l} S \rightarrow S+S \\ S \rightarrow S \times S \\ S \rightarrow \text{número} \end{array} \right\} \begin{array}{l} \text{gramática independiente del} \\ \text{contexto} \end{array}$$

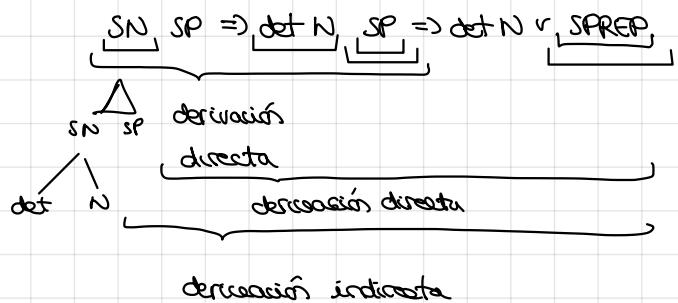
Lo que aquí es sintaxis en G_1 es

Estos "números" que en S_1 son categorías léxicas se generan a partir de una gramática más simple (de tipo regular) en la que $\Sigma = \{0, 1, 2, \dots\}$

$$\begin{aligned} a, b, c, \dots \in \Sigma &= E_j : a = +, a = 18 \cdot 5 \\ \dots x, y, z \in \Sigma^* &= E_j : 1 \cdot 2 + 3 + 18 \cdot 5 \\ A, B, C, \dots \in N &= E_j : A = S \\ \dots X, Y, Z \in N \cup \Sigma &= E_j : X = S, X = +, X = 18 \cdot 5 \\ \alpha, \beta, \gamma, \dots \in (N \cup \Sigma)^* &= E_j : S + J \approx 18 \cdot 5 \\ x, \dots n \in \Sigma^* \text{ y tiene longitud } n &= E_j : x, y = 18 \cdot 5, x_{2,3} = 8 \\ x_{i..j} \in \Sigma^* &\text{ el trozo de la cadena } x \in \Sigma^* \text{ entre las posibilidades "i" y "j".} \end{aligned}$$

$$\left. \begin{array}{l} S \rightarrow S+S \\ S \rightarrow S \times S \\ S \rightarrow \text{número} \end{array} \right\} = G_0$$

$$E_j : S + J \approx 9 + 5 \Rightarrow S + 3 \approx 9 + 5$$



Derivaciones

def: Sea $g = (N, \Sigma, P, S)$ una gramática, decimos que:

α, β, γ derivan directamente de S_j si $P \Rightarrow j \in P$.

(notación: $\alpha \beta \gamma \xrightarrow{P \Rightarrow j} \alpha \gamma \beta$)

JERARQUÍA DE CONSTRUCCIÓN.

14/02

gram. regulares (por defecto construimos reglas por la derecha).

$$A \rightarrow aA / a \in \Sigma, a \in N$$

$$A \rightarrow a$$

def: Un lenguaje L / Γ g. gramática regular con $L = L(\Gamma)$ se dice un lenguaje regular.

def: Sea $g = (N, \Sigma, P, S)$ una gramática, decimos que es independiente del contexto si las reglas son de la forma $A \rightarrow \alpha / A \in N$
 $\alpha \in (N \cup \Sigma)^+$

Ejem: cualquier gram. reg e independiente del contexto

Ejem: $S \rightarrow S+S$

$S \rightarrow S \times S$

$S \rightarrow \text{número}$

NOTA: Contexto \Rightarrow gramática gráfica \subseteq gram. indep. contexto
no es? \Rightarrow lenguaje \subseteq lenguaje " "

Que un lang. puede generarse mediante una gramática indep. del contexto es regular, no implica que el lenguaje no pueda ser regular.

Sea $G_1 = A \rightarrow aA$
 $A \rightarrow a$

G_1 es regular, luego el lenguaje regular $L(G_1) = \{a, aa, aaaa, \dots\} = a^+$

Sea $G_2 = A \rightarrow aA$, entonces $L(G_2) = L(G_1)$

$A \rightarrow a$

$A \rightarrow aAa$

G_2 no es una gramática reg. (o indep. contexto) pero

$L(G_2)$ sigue siendo regular.

def: Una gramática $G = \{N, I, P, S\}$ se dice dependiente del contexto si sus reglas son de la forma:

$$\alpha \rightarrow \beta / \alpha, \beta \in (N \cup I)^+ \\ |\alpha| \leq |\beta|$$

o sensible al contexto

Ejemplo: $S \rightarrow ACaB$
 $Ca \rightarrow aAC$
 $CB \rightarrow DB$
 $CB \rightarrow E$
 $aD \rightarrow Da$
 $aE \rightarrow Ea$
 $AE \rightarrow E$

} Dependiente del contexto.

Ejemplo: 1) $S \rightarrow aSBC$

2) $|abc$

3) $B \rightarrow BC$

4) $bB \rightarrow bb$

5) $bc \rightarrow bc$

6) $CC \rightarrow cc$

Ejemplo (derivación dependiente del contexto sobre G_3).

$$\begin{array}{c} (1) \\ S \Rightarrow aSBC \stackrel{(2)}{\Rightarrow} aab CBC \stackrel{(3)}{\Rightarrow} aabbCBC \stackrel{(4)}{\Rightarrow} aabbBCC \stackrel{(4)}{\Rightarrow} aabb bCC \stackrel{(4)}{\Rightarrow} \\ \Downarrow \stackrel{(5)}{\Rightarrow} aabbC \stackrel{(6)}{\Rightarrow} aabbcc \end{array}$$

De hecho $L(G_3) = \{a^n b^n c^n, n > 1\}$

def: Una $G = \{N, I, P, S\}$ se dice sin restricciones (o con estructura de frase) si sus reglas son de la forma: $\alpha \rightarrow \beta$

Observación: a^n es un lang. reg.

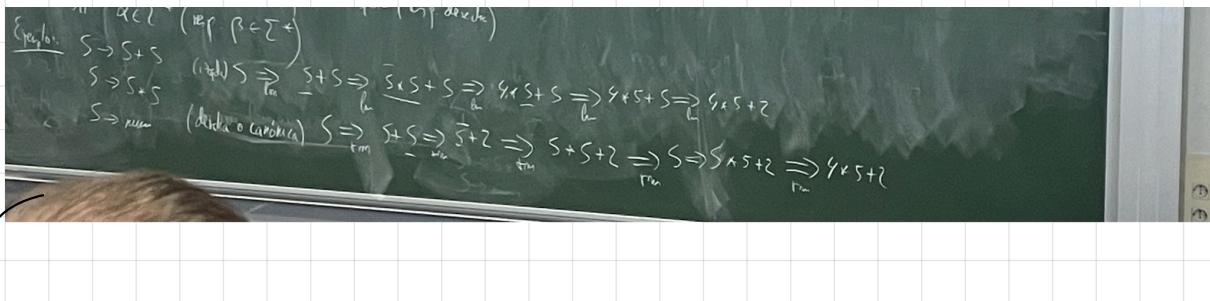
$\{a^n b^n, n > 1\}$ es un lang. independ. del contexto $\left\{ \begin{array}{l} S \rightarrow aab \\ |ab \end{array} \right.$

$\{a^n b^n c^n, n > 1\}$ dep. del _____

$\{a / \exists n > 0, i=2^n\}$ sin restricciones

def: Sea $\mathcal{G} = \langle N, \Sigma, P, S \rangle$ una gramática y sea una derivación $\alpha A \beta \Rightarrow \alpha \gamma \beta$ decimos que es una derivación por la izquierda (derecha) si $\alpha \in \Sigma^*$ ($\beta \rightarrow \Sigma^*$)

Ejemplo:

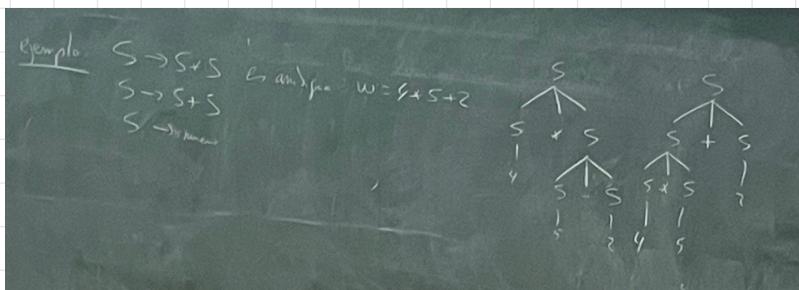


Algoritmo de construcción de árboles sintácticos de expresiones.

$\rightarrow \neq \text{de leng. ambigüo}$

def: Una gramática se dice ambigua si $\exists x \in L(\mathcal{G}) / \exists$ al menos dos derivaciones canónicas $S \Rightarrow x$.

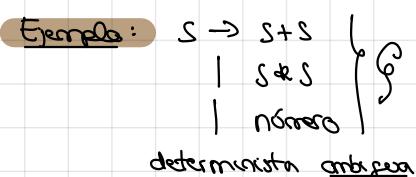
Ejemplo:



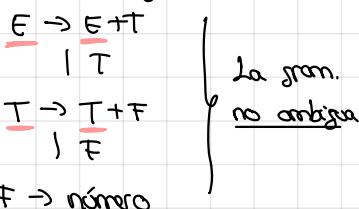
def: Un lenguaje L se dice no ambiguo o no determinista si y solo si existe una gramática no ambigua que lo genera.

28/02

Un lenguaje L tiene ser ambiguo y ser generado por una gramática ambigua (basta con que también eso sea una gramática no ambigua que lo genere).

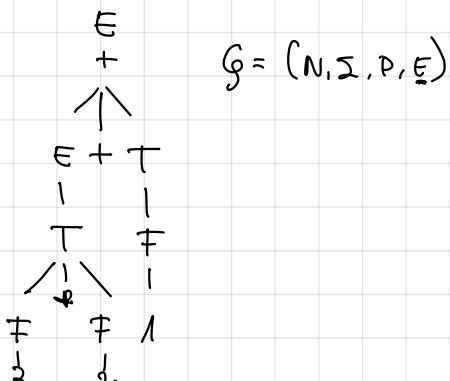


sin embargo, $L(\mathcal{G})$ es no ambiguo (determinista). En particular, la siguiente gram. es determinista y genera a $L(\mathcal{G})$



* Cuanto más abajo esté un operador en un árbol, antes se evalúa.

Entonces metiendo la prioridad y la asociatividad



Enta es la idea para describir esta gramática



Para decir que un \ast tiene más prioridad, tengo que ver si está más al fondo. En este caso, hay que poner por otra forma.

Un lenguaje (L) se dice inherenteamente ambiguo si $\exists g$ determinista ge do genere.

Lo que vamos a ver \Rightarrow **(EXPRESIONES REGULARES)** \Rightarrow las cuales ge vamos a ver. - autómatas finitos.

* Una máquina de Turing $M = (Q, \Sigma, \Gamma, \delta, q_0, F, B)$

$$\text{Lo } Q \times \Gamma \rightarrow Q \times \Gamma \times \{I, D\}$$

son los mismos

- Lenguajes regulares
(ya sea como d-
finido: $A \rightarrow aA$
||| 1 a)
- expresiones regulares.

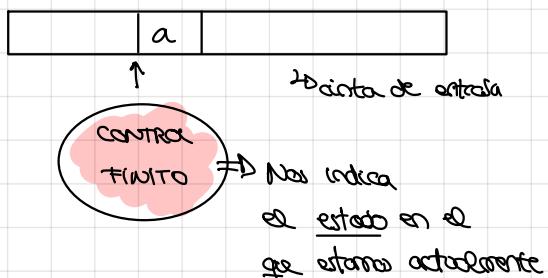
II. Autómatas finitos y gramáticas regulares.

Un **FFF** es una S-pla.

$A = (Q, \Sigma, \delta, q_0, F)$ donde:
 $\underline{\text{desta}}$ Q es un conjunto finito de estados.
 Σ es un alfabeto.
 $q_0 \in Q$ es un estado inicial.
 $F \subseteq Q$ es un conjunto de estados finales.
 $\delta : Q \times \Sigma \rightarrow Q$

Con esto veremos que es una simplificación de la máq. de Turing.

NOTA:



En función del estado actual " q " y de la entrada "aw" me voy al estado " p " si $\delta(q, a) = p$ y aviento una paréntesis (verde) a la derecha en la entrada (diseño que lleva analizado "a").

def: Sea $A = (Q, \Sigma, \delta, q_0, F)$ un AF, definimos una configuración en A como por $(q, w) \in Q \times \Sigma^*$ donde " q " es el estado actual del control finito, y " w " es la parte de la entrada que aún queda por analizar.

NOTA: Hay dos tipos de configuraciones "sencillos".

Configuración inicial: (q_0, w) por tanto " w " se entiende el conj. de instrucciones completa a seguir.

Configuración final: $(q, \epsilon) / q \in F$

def: Sea $A = (\dots)$ un AF un movimiento en A es el paso de una configuración a otra
 \uparrow lo mismo de arriba / por aplicación (una o más veces) de la función de transición δ .

NOTA: $(p, aw) \xrightarrow{} (q, w)$ si $\delta(p, a) = q$

NOTA: Obviamente podemos encadenar movimientos. | $\xrightarrow{+}$ cero ó más movimientos
 $\xrightarrow{*}$ 1 ó más movimientos

def: Sea $A = (\dots)$ un AF, $x \in \Sigma^*$, decimos que A acepta a "x" si $(q_0, x) \xrightarrow{*} (q, \epsilon) / q \in F$.

def: Sea $A = (\dots)$ un AF, definimos el lenguaje aceptado por A como $T(A) = \{ x \in \Sigma^* / A \text{ acepta } x \}$

III

$$\{ x \in \Sigma^* / (q_0, x) \xrightarrow{*} (q, \epsilon) / q \in F \}$$

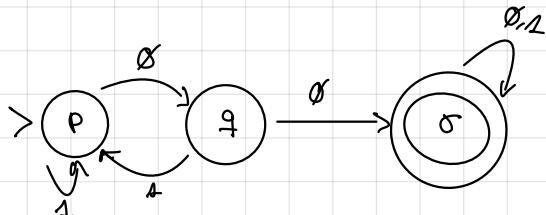
decimos que $L = T(A)$ es un conjunto regular

Demostremos que conjuntos regulares y lenguajes regulares son lo mismo. Esto es, demostremos que un lenguaje es regular si se reconoce por un AF.

Representación de AFs: por tabla de transiciones o mediante grafos de estado.

Ejemplo: Sea $A = (\{p, q, r\}, \{f, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset\})$ donde S viene dado por:

S	\emptyset	f
p	$\{q, r\}$	$\{p\}$
q	$\{p, f\}$	$\{p\}$
r	$\{p, f\}$	$\{p, f\}$



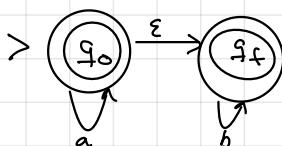
* Habitualmente solo se facilita el grafo o la tabla.

Lenguajes reconocidos por AF (conj. regulares).

tiras de a seguidas de tiras de b.

Ejemplo: $L = \{a^i b^j / i, j > 0\}$ es un conjunto regular

En efecto, L se acepta por el AF.

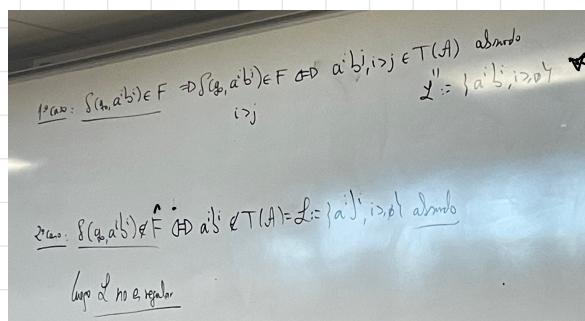


th: El conj $L = \{a^i b^i, i > 0\}$ no es regular.

dem: Supongamos que L es un conjunto regular. $\Leftrightarrow \exists A = (Q, \Sigma, \delta, q_0, F)$ AF tal que $T(A) = L$. Consideremos entonces el conjunto $\{S(q_0, \alpha^i), i > 0\} \subseteq Q$ $\Rightarrow \exists i > j / S(q_0, \alpha^i) = S(q_0, \alpha^j)$

Q es finito

$i > j$ es conj infinito



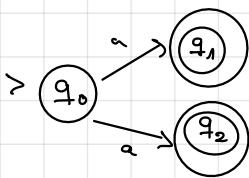
$\frac{1}{2} S(q_0, \alpha^i) = S(q_0, \alpha^j)$
entre los distintos casos:
caso 1:
 $S(q_0, \alpha^i) \in F \Rightarrow \delta(q_0, \alpha^i) \in F \text{ y } \alpha^i \in T(A) \text{ alrededor}$



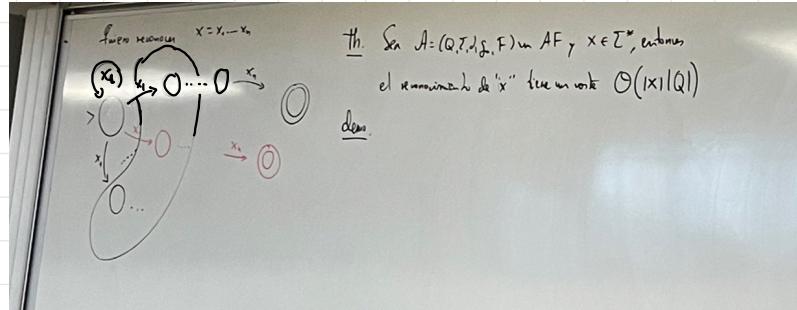
$$f: Q \times \Sigma \rightarrow \wp(Q)$$

as in ? cosa.

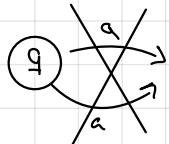
$(q_1, a) \rightsquigarrow \{q_1, q_2, \dots, q_n\}$ esto es los movimientos pueden ser no deterministas (desde un estado con una misma entrada podrían llegar a más de un estado).



7/03



def: Decimos que A es determinista o no ambiguo si $|f(a, q)| \leq 1 \forall a \in \Sigma \forall q \in Q$

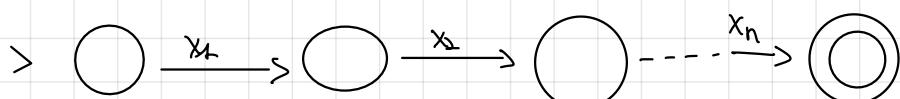


En general, los AF son no deterministas (ambiguos).

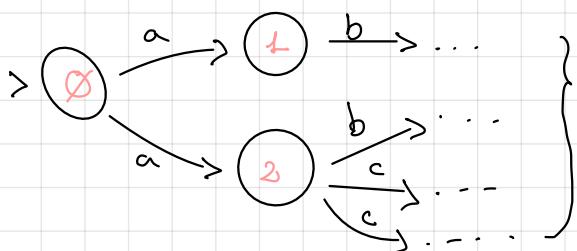
th: Sea $A = (Q, \Sigma, \delta, q_0, F)$ un AF (en general NFA), entonces $\exists A' = (Q', \Sigma, \delta', q_0', F')$ un DFA / $T(A) = T(A')$.

¿Por qué son interesantes los autómatas finitos deterministas?

DFA

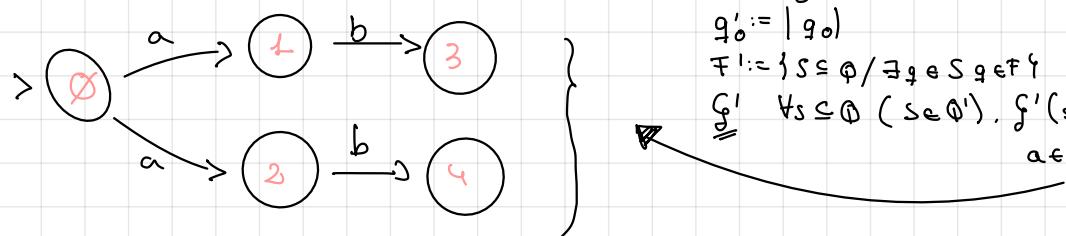


Recorrer $x \in \Sigma^*$ en la AFD, tardaría en tiempo $O(|x|)$

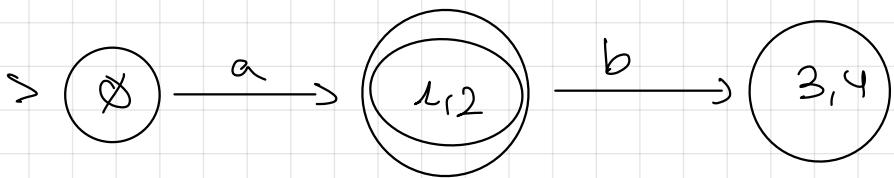


Si quisieras ir así a ese 2 \Rightarrow vamos a hacer la situación, para se funde con la demonstración. se fusionan.

Sea entonces $A = (Q', \Sigma, \delta', q_0', F')$ donde $Q' := \wp^*(Q)$ (muchas no las usaremos).



$q_0' := \{q_0\}$
 $F' := \{S \subseteq Q \mid \exists q \in S \text{ } g \in F\}$
 $\delta' \quad \forall s \subseteq Q \text{ } (\text{se } Q'), \delta'(s, a) = S' / S' =$
 $a \in \Sigma = \{p \in Q, \exists q \in S, p \in \delta(q, a)\}$



Falta por demostrar que $T(A) = T(A')$

Para ello demostraremos primero que los caminos en A y A' son los mismos, esto es, si, $(s, \omega s) + \frac{i}{\lambda} (s', \omega) \Leftrightarrow s' = \{ p \in Q / \exists q \in S, (q, \omega s) + \frac{f(q, v)}{\lambda} (p, \omega) \}$

Lo haremos por inducción en "i"

i=1 trivial, ya se en ese caso.

$$(s, \omega s) + \frac{1}{\lambda} (s', \omega) \Leftrightarrow s' = \{ p \in Q / \exists q \in S, (q, \omega s) + \frac{f(q, v)}{\lambda} (p, \omega) \}$$

(por def de
movimientos y
de transiciones de
camino)

cíerto

$$\underset{\text{Aplicaremos hipótesis inducción}}{\Leftrightarrow} (s, \omega s) + \frac{n}{\lambda} (s', \omega) \Leftrightarrow (s, \omega s) + \frac{n-1}{\lambda} (s'', \omega) + \frac{1}{\lambda} (s', \omega) \Leftrightarrow$$

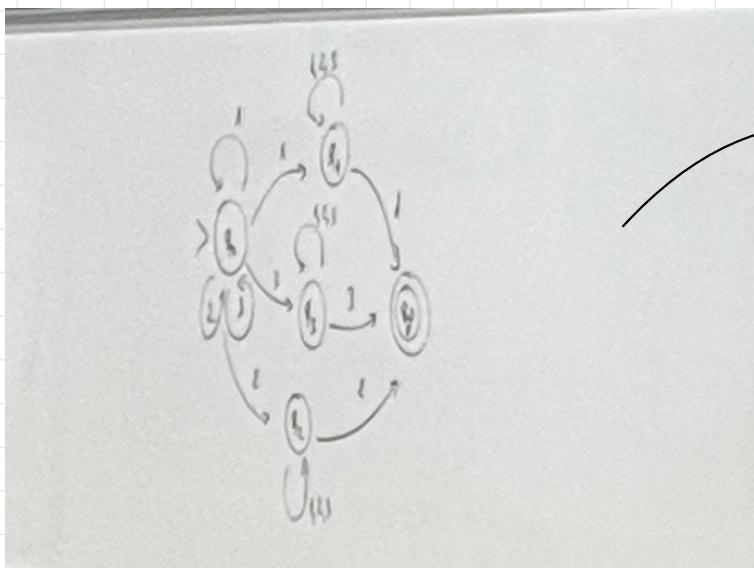
$$\Leftrightarrow \left\{ \begin{array}{l} S'' = \{ p \in Q / \exists q \in S, (q, \omega s) + \frac{n-1}{\lambda} (p, \omega) \} \rightarrow \text{hipótesis.} \\ S' = \{ p \in Q / \exists q \in S'', (q, \omega s) + \frac{1}{\lambda} (p, \omega) \} \end{array} \right\} \Leftrightarrow$$

Aplicaremos la definición

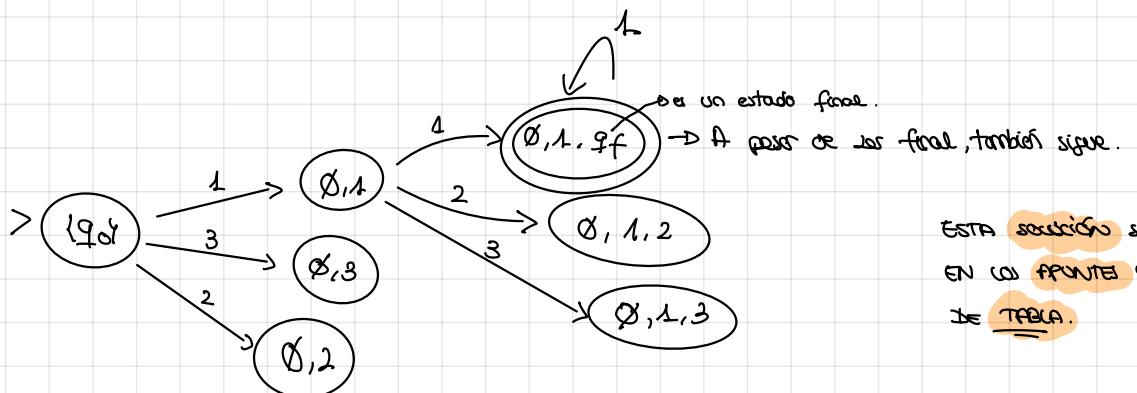
$$\Leftrightarrow S' = \{ p \in Q / \exists q \in S, (q, \omega s) + \frac{n}{\lambda} (p, \omega) \} \quad \text{que es lo que queremos probar.}$$

Entonces yo puedo probar que $T(A) = T(A')$, ya que $x \in T(A') \Leftrightarrow (\exists g_0 \in F, x) + \frac{i}{\lambda} (g'_0, \epsilon), g'_0 \in F' \Leftrightarrow$
 $\Leftrightarrow \exists p \in F / (g_0, x) + \frac{i}{\lambda} (p, \epsilon) \Leftrightarrow x \in T(A)$.

Ejemplo (Vamos a ver los primeros pasos).



construiremos el
equivalente

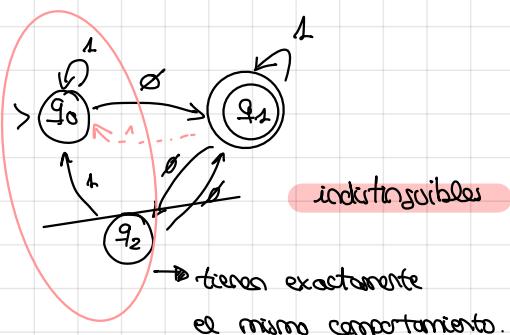
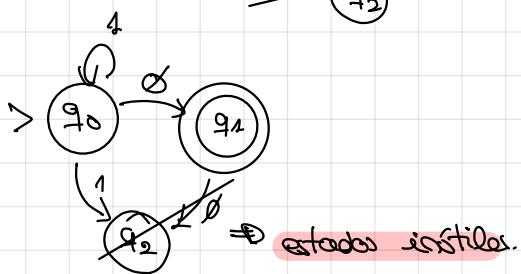
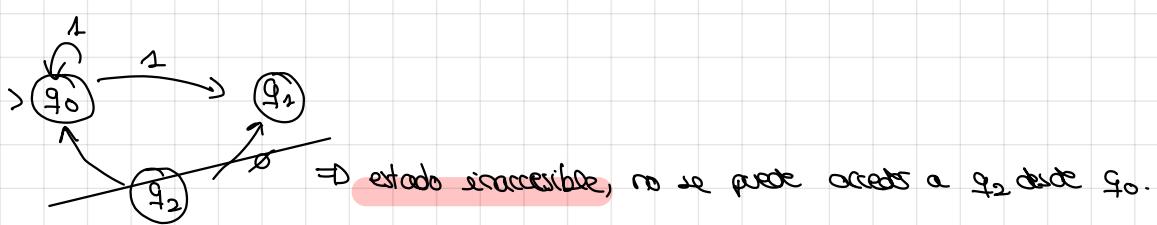


ESTA SECCIÓN SIRVE
EN LOS APUNTES EN FORMA
DE TABLA.

Al determinizar un autómata, conseguimos que vaya más rápido porque elegimos un camino. Aunque el autómata no estuviese determinizado funcionaría igual pero tardaría más.

A pesar de que el autómata determinista vaya más rápido, no garantiza que sea el más eficiente.

Ejemplos:

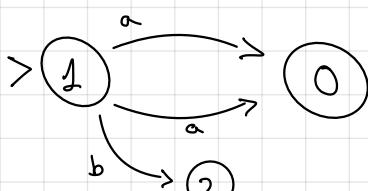


AF

Menos optimizado el "funcionamiento" de un autómata finito (determinización).

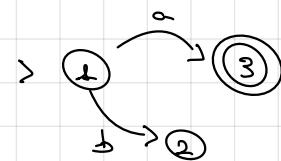
Vamos a optimizar la tabla de los AF deterministas (minimización o reducción).

14/03



Autómata no determinista

si lo determinizamos



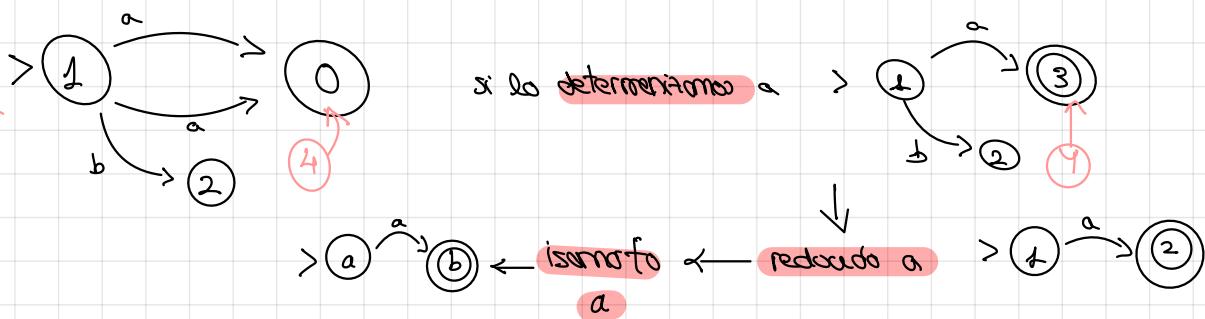
$\exists \quad A' = (\dots)$ AFD minimal / $T(A) = T(A')$
verifica (valores isomorfismo)

La demostración no la haremos.

Ilustraremos mediante ejemplos como reducir AFD. En concreto, veremos cómo eliminar estados



- **Instilos** \rightarrow Un estado $q \in Q$ es instil si $\nexists \omega / S(q, \omega) \in F$
- **Inaccesibles** \rightarrow Un estado $q \in Q$ es inaccesible si $\nexists \omega / f(q_0, \omega) = q$
- **Indistinguibles** \rightarrow A continuación se muestra explicado más detallado.



¿Cómo **eliminar estados inaccesibles**?

Basta recorrer todos los caminos a partir del estado inicial marcando los visitados. Terminado el proceso, eliminamos los estados no marcados.

("Como encontramos el puente que tiene en Valladolid una entrada \Rightarrow empezamos en la puerta del sol y recorremos todos los corredores de España y ponemos una bandera, el que no tiene bandera, se elimina porque es instil")

¿Cómo **eliminar estados instiles**? (**visitamos un estado y no tiene salida \Rightarrow instil**)
 \hookrightarrow que no haga camino hasta el estado final

Los instiles e inaccesibles se ven a ojo, los que puede poner ejemplo en el examen \rightarrow **indistinguibles**

Vamos a ver ejemplo.

EESTADOS INDISTINGUIBLES.

Def. Sea $A = (\dots)$ un AF y sean $q_1, q_2, q \in Q$ tales que $q_1 \neq q_2$. Entonces:

a) $x \in \Sigma^*$ **distingue** a q_1 de q_2 $\xrightarrow{\text{si una cadena de instrucciones}}$

$$S(q_1, x) \in F \circ x \notin$$

AND

$$S(q_2, x) \notin F \circ x \in F$$

Uno llega al final y otro no

"Comportamiento máquina café \rightarrow 1€ café largo, 1€ marchado"

ejemplo de
clase

El comportamiento viene dado por el camino que llega al final, el conjunto de visitados.

es el comportamiento que las transiciones ge "yo para llegar al final"

Para eso no importa que sea instil, que no sea igual es un camino.

\downarrow
cómo saber si son indistinguibles? si en un caso me salta el café y en otro no.

- b) g_1 y g_2 son indistinguibles si $\nexists x \in \Sigma^*, |x| \leq k / g_1$ y g_2 son distinguibles por "x".
- c) g_1 y g_2 son indistinguibles si $\nexists k \in \mathbb{N} / g_1$ y g_2 son indistinguibles.
- "2 botones ≠ pero va a salir la mano en uno que en otro".

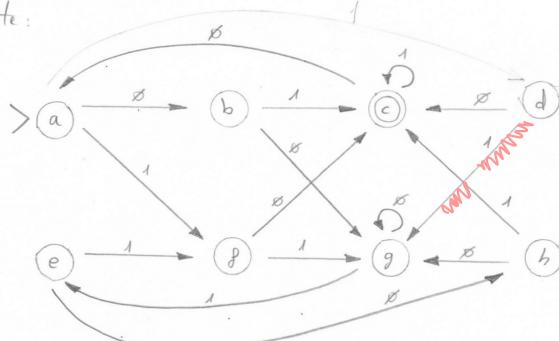
def: Sea $A = (\dots)$ un AFM se dice que A está reducido si no tiene estados innecesarios, ni indistinguibles.

th: Sea $A = (\dots)$ un AFM entonces el AFM reducido eliminando estados innecesarios e indistinguibles es el autómata finito minimal para el lenguaje $T(A)$.

con el menor nro.
de estados posibles

La dem no la vamos a hacer.

Ejemplo: Dado el DFA representado por el grafo de transiciones (pag 131, libro 4) siguiente:



inaccesible
(es imposible llegar a d)
se eliminará ese estado
y todo lo que sale de él.

En cualquier caso podemos detectar indistinguibles sin eliminar innecesarios e innecesibles.

Si ya es un AFM, luego queda determinar sólo innecesarios e indistinguibles.

↑
no hay (con uno llega a final
y con otro no).

Vamos a calcular esa matriz de distinguidos:

b	X	← (a,b)
c	X	X
d		X
e		X
f		X
g		X
h		X

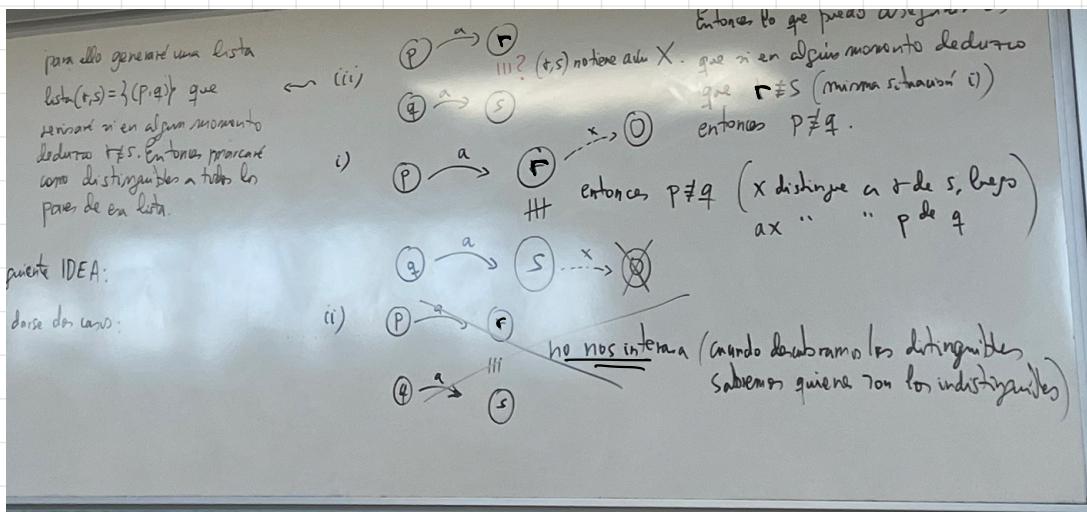
Nota: si $g \in F$ y $g \notin F$, $g_1 \neq g_2$ (\in los distinguibles de forma trivial)

Aplicaremos un proceso induutivo, basado en la siguiente idea:

Sea $r, s \in Q$ y $a \in \Sigma / d(r,a) = r$ y $d(s,a) = s$

darse tres casos:

- i) $r \neq s$
- ii) $r = s$
- iii) no se aplica vñ $r = s$.



(a, b)

$$\begin{cases} f(a, b) = b \\ f(b, a) = g \end{cases} \quad \left\{ \text{dado modo sabemos del par } (b, g) \right.$$

situación iii, entonces

genero lista $L(b, g) = \{(a, b)\}$

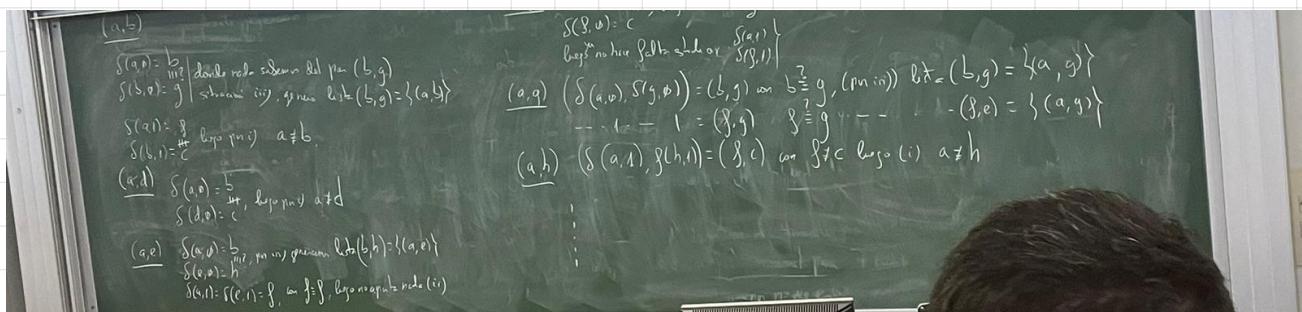
$$\begin{cases} f(a, 1) = f \\ f(b, 1) = c \end{cases} \quad \left\{ \text{luego por i) } a \neq b \Rightarrow \text{marcamos } X \text{ en la casilla } b \boxed{X} \right.$$

a

(a, d)

$$\begin{cases} f(a, \emptyset) = b \\ f(d, \emptyset) = c \end{cases} \quad \left\{ \text{luego por i) } a \neq d. \right.$$

(a, 0)

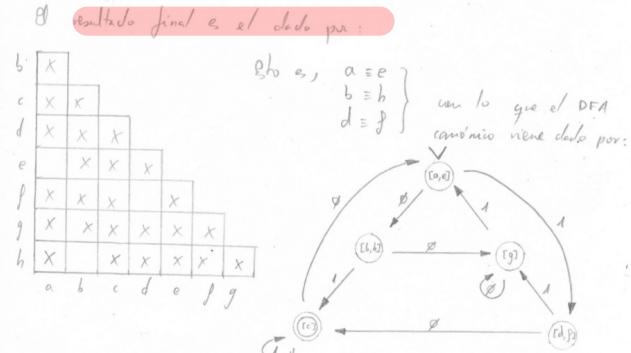


↑
 (finalización del ejercicio).

21/03

(b, g) son distinguibles \Rightarrow todos los estados de la lista b y g también son distinguibles.

$$(f(b, 1), f(g, 1)) = (c, e)$$



- Lenguaje regular (generado por gram regular)

||| estos 2 cosas son iguales

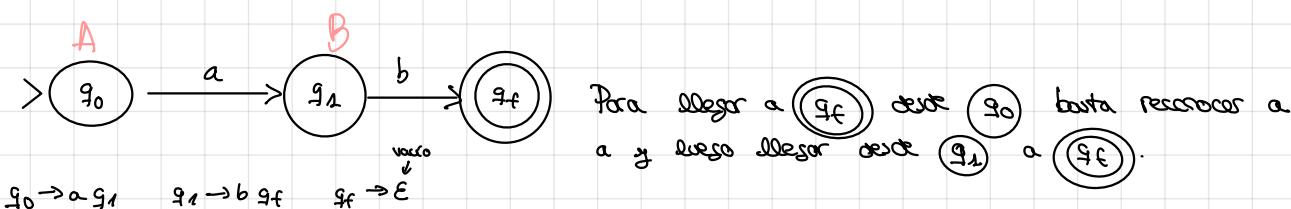
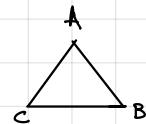
- Conjuntos regulares (los reconocidos por aut. finito)

} Veremos que un lenguaje regular es el reconocido por un aut. finito.

CONJUNTOS REGULARES. (generadores y reconocedores)

Lema: Sea L un conjunto regular, entonces existe \mathcal{G} gramática regular / $L = \mathcal{L}(\mathcal{G})$.

$A \rightarrow aB$ Para reconocer A basta reconocer a a luego B .
 $B \rightarrow b$



El papel que en uno hacen los terminales y en otro los estados. La lectura es la misma, simplemente se cambia la notación.

Demo: L conj reg \Leftrightarrow $\exists \mathcal{A} = (\mathcal{Q}, \Sigma, \delta, q_0, F)$ AFD / $L = T(\mathcal{A})$.

Construimos $\mathcal{G} = (N, \Sigma, P, S)$ en la forma:

$$N := \mathcal{Q}$$

$$S := q_0$$

$$P := \{ q \rightarrow a \delta(q, a), q \in \mathcal{Q} \cup \{ q \rightarrow \epsilon / q \in F \} \}$$

(II)

\mathcal{G} es regular: Trivial (sus reglas tienen la forma adecuada)

$L = \mathcal{L}(\mathcal{G})$ demostraremos ante:

$$\boxed{\text{I)} \quad \forall y \in T(\mathcal{A}), \exists q_0 \xrightarrow{*} y \quad / \quad \begin{matrix} q \in \mathcal{Q} \\ \delta(q_0, q) = y \end{matrix}}$$

→ los símbolos que construimos con la gramática se corresponden con los caminos que construimos con el autómata.

Lo haremos por inducción a $|y|$ (longitud de $y \in \Sigma^*$).

$|y| = 0 \Leftrightarrow y = \epsilon$, luego la conclusión es trivial. Basta tener $q = q_0$.

$|y| = n \Leftrightarrow$ Suponemos el resultado cierto.

$|y| = n+1 \Leftrightarrow \exists w \in \Sigma^* / w = \text{término } n, \text{ duplica por hipótesis de inducción}$
que el resultado se verifica.

$$\therefore q \Rightarrow aq \quad / \quad \begin{matrix} q \in \mathcal{Q} \\ \delta(q, a) = w \end{matrix} \quad \{ \Rightarrow \}$$

⇒ Por construcción de \mathcal{G}

$$\hookrightarrow \delta(q, a) \rightarrow a \delta(q, a) \in P$$

"
ESTO ES UN EJ.
PARA REPETIR
ANOTAR!"

De nuevo, esto implica $\Rightarrow \omega \in \mathcal{L}(\mathcal{S}, \omega) \Rightarrow \omega \in \mathcal{L}(\mathcal{S}, \omega) \Rightarrow g \in \mathcal{L}(g, j)$ y hemos demostrado.

Visto (I), es trivial demostrar que $L = L(\mathcal{S})$. En efecto:

$$\left[g_0 \Rightarrow g \in \mathcal{L}(g, j) \Rightarrow g \right] \stackrel{(II)}{\Leftrightarrow} \mathcal{L}(g_0, j) \in F.$$

\Updownarrow
 \Updownarrow

$j \in T(A)$

$j \in L(f)$

demostado

Corolario: Si L es un conj regular entonces L es un lenguaje regular no ambiguo.

Demo: L conj reg $\Rightarrow \exists S$ gram reg / $L = L(S)$
de derivación en S son "nulas" para una cadena dada

Lema: Sea L lenguaje regular, entonces existe $A = (\quad) AF / T(A) = L$

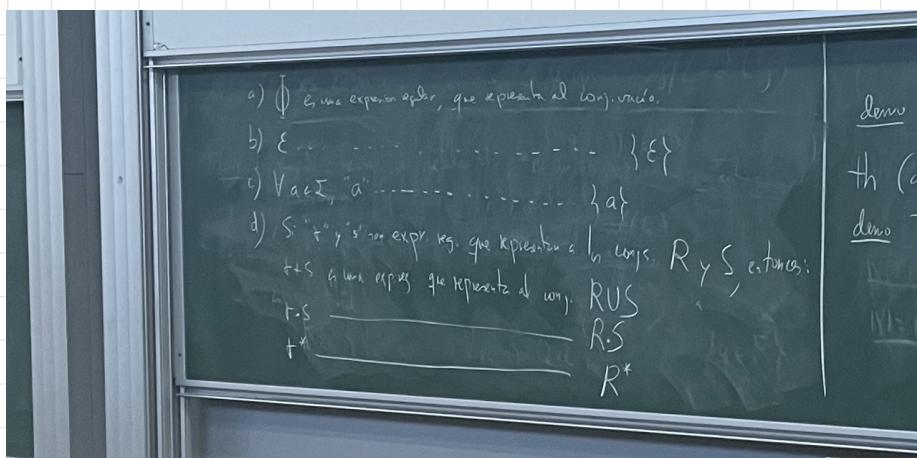
La demo no la vamos a ver.

Th (de Chausk-Tüller): L es un lenguaje regular $\Leftrightarrow L$ es un conj. regular

Demo: Trivial para los dos lemas anteriores.

Expresiones regulares.

Def: Sea Σ un alfabeto, definimos las **expresiones regulares** sobre ese alfabeto, como:



Ejemplo: \emptyset es una expr. reg que representa al conjunto $\{\emptyset\}$

NOTA: Prioridad de operadores (de menor a mayor) $\ast, ., +$

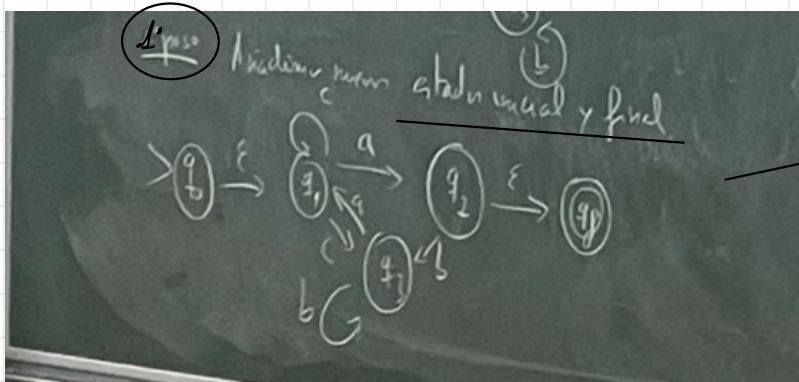
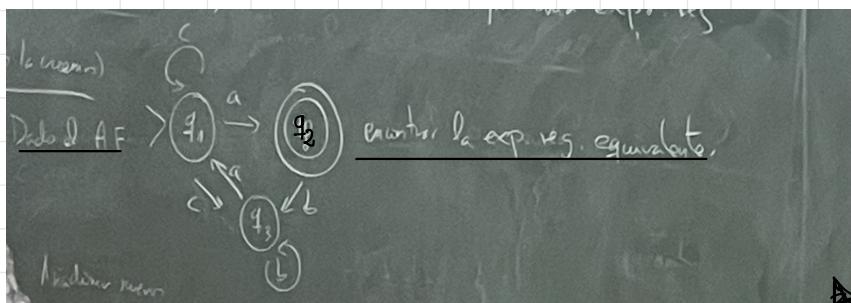
$(\emptyset + \emptyset)^*$ → tira de 0's y 1's

Ejemplo (tonto) de $\rightarrow \ast^+ := \ast^* \ast$
macro

th: Si L es un conj. regular, entonces L se deriva por una exp. reg.

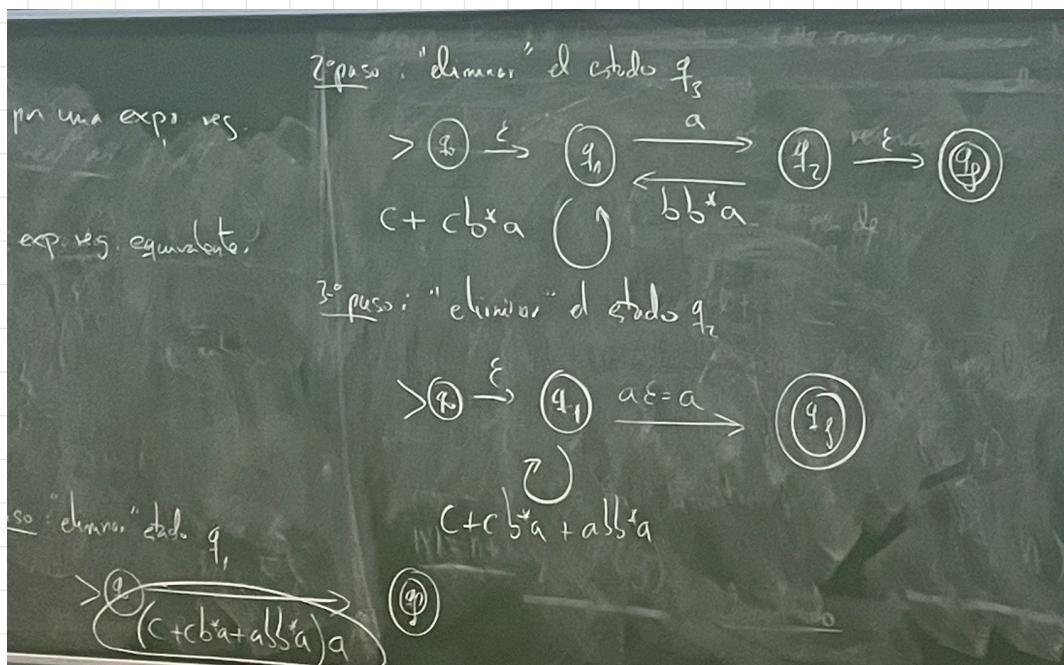
demos (no se saltamos)

Ejemplo (Puede caer en similar)



1^o paso: es igual

2^o paso



th. Los lenguajes regulares son un subconjunto propio de los lenguajes indep. del contexto.
(esto es, $L_R \subseteq L(C)$)

demo: "C" trivial

" \emptyset " Basta considerar $L = \{a^i b^i \mid i > 0\}$

La lenguaje visto que \emptyset no es un C.R.

Sin embargo, \emptyset es un C.R. Basta considerar que está generado por la gram. dep. del contexto.

Leng. reg \equiv Clangs. reg.

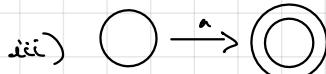
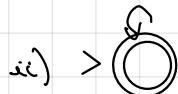
¿Expresión regular \equiv lenguaje regular?

th. Sea R una expresión regular $\exists A = (Q, \Sigma, \delta, q_0, F)$ un AFN. con transiciones $/L(r) = T(A)$

demo. Sea r una expre. regular hacemos la demo por inducción en el número $|r|$ de operadores en " r ".

$$|r| = 0, \text{ entonces } r = \begin{cases} \emptyset & \\ \epsilon & \\ a \in \Sigma & \end{cases} \quad |r| \leq n \text{ Sustituyendo esto:}$$

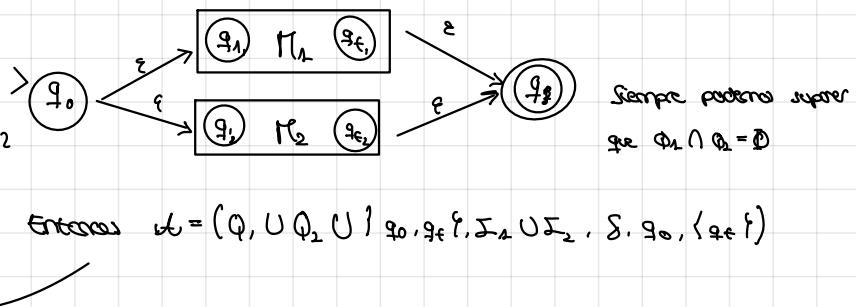
i) $> \bigcirc$: recorre ruta.
el conj vacío



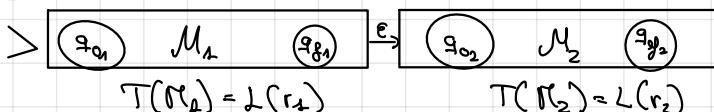
$|r| = n+1$. Podemos considerar 3 situaciones.

caso 1: $r = r_1 + r_2 / |r_i| \leq n, i=1,2$, luego
por hipótesis de inducción \exists
 $M_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, F_1) / L(r_1) = T(M_1)$
 $M_2 = (Q_2, \Sigma_2, \delta_2, q_{02}, F_2) / L(r_2) = T(M_2)$.

- i) Se dice que $f(q, \epsilon) = \{q_{01}, q_{02}\}$
- ii) $f(q, a) := f(q, a), \forall q \in Q, \{q\}, \forall a \in \Sigma, i=1,2$
- iii) $f(q_{f1}, \epsilon) := \{q_{f2}\} i=1,2$



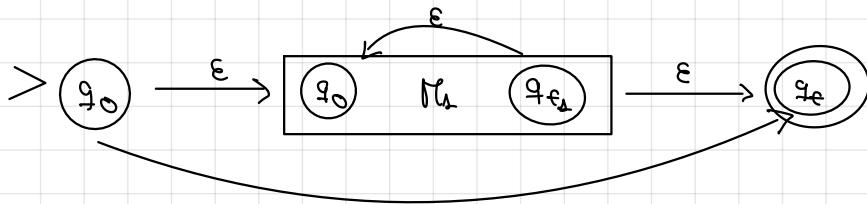
caso 2: $r = r_1 \cdot r_2$



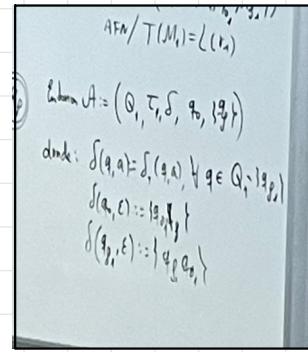
Ento es, $L = (\underline{Q_1 \cup Q_2}, \Sigma_1 \cup \Sigma_2, \delta, q_{01}, \{q_{f2}\})$
siempre podemos suponer $Q_1 \cap Q_2 = \emptyset$

caso 3: $r = r_1^*$ / $|r_1| \leq n$, luego por hipótesis de inducción $\exists M_1 = (Q_1, \Sigma_1, \delta_1, q_{01}, F_1)$ AFN / $T(M_1) = L(r_1)$

- i) $f(q, a) := \delta_1(q, a), \forall q \in Q_1 \setminus \{q_{f1}\}, \forall a \in \Sigma$
- ii) $f(q, a) := \delta_1(q, a), \forall q \in Q_1, \forall a \in \Sigma$
- iii) $f(q_{f1}, \epsilon) := q_{01}$



Ejemplo \Rightarrow está en los apéndices



Propiedades de los lenguajes REG.

EXERCICIO

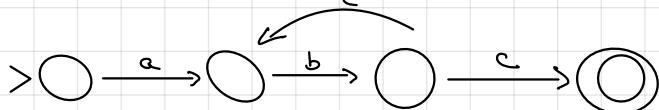
th. de iteración para lenguajes reg. (Lema del barbero).

Que un conj sea reg. no tiene pq cumplir
eso para para esto si tiene que ser reg.

Es una condición necesaria para que un conj. sea regular

sea $t = (\dots)$ un AFD / $|t|_0 = n$.

Entonces $\forall w \in T(t)$, $|w| = m > n$, tenemos que $\exists x, y, z \in \Sigma^*$ / $w = xyz$
 $y \notin \epsilon$
 $xy^kz \in T(A) \quad \forall k > 0$



$$w = a^x b^y c^z$$

Decmo: Sea $w = a_1 \dots a_m / m > n$

$\tilde{Q} := \{\delta(q_{i_1}, a_1 \dots a_{i_k}), i=1 \dots m\}$
 indigo elementos repetidos, no está así en los
 apéndices porque se olvidó.

$\Rightarrow |\tilde{Q}| = m+1 > n := |Q| \Rightarrow$ (al menos un estado
 en \tilde{Q} está repetido)
 $\tilde{Q} \subseteq Q$
 $\Rightarrow \exists q \in \tilde{Q} \subset Q /$
 $q = q_r, \text{ con } q := \delta(q_0, a_1 \dots a_n)$

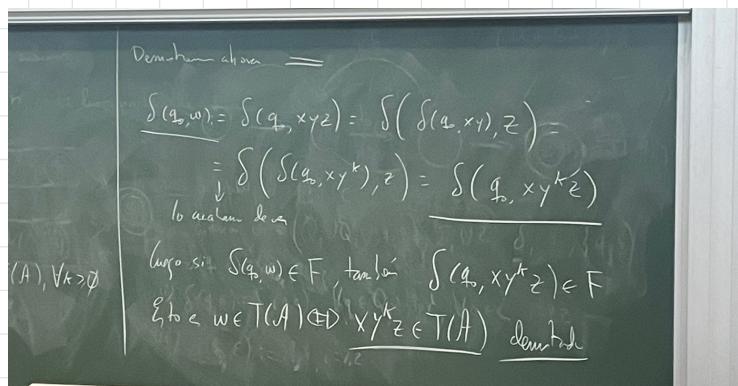
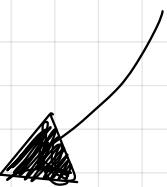
$$A = DFA$$

Consideremos entonces: $w = xyz /$
 $x = a_1 \dots a_r$
 $y = a_{r+1} \dots a_r$
 $z = a_{r+1} \dots a_m$

Falta por ver que $xz^k \in T(A), \forall k > 0$. Para
 ello primero veremos que: $(\exists) \delta(q_0, x) = \delta(q_0, x, y^k),$
 $\forall k > 0$.

Lo haremos por inducción en k :

$$\begin{aligned} \underline{k=0} \quad \delta(q_0, x) &= \delta(q_0, x, y^0) \text{ trivialmente} \\ \underline{k=n+1} \quad \delta(q_0, x, y^{n+1}) &= \delta(\delta(q_0, x, y^n), y) = \\ &= \delta(\delta(q_0, x)_y) - \delta(q_0, x)_y = \delta(q_0, x) \end{aligned}$$



Ejemplo. ¿ $L = \{0^n 1^n, n \geq 1\}$ es regular? Si es necesario contar, pues entonces no es regular como L - imprecision.

(Lo vamos a comprobar usando el teorema de iteración)

Supongamos que lo es. Entonces $\exists A = (Q, \Sigma, \delta, q_0, F) \text{ DFA } L = L(A)$. En particular, se verificaría el th. de iteración (long. reg.). Esto es $\forall w \in L(A) / |w| > |Q| = n$. Entonces $\exists x, y, z \in \Sigma^* / \begin{cases} i) \\ ii) \\ iii) \end{cases} \dots$

Tenemos ese valor de $n = |Q|$, entonces $|0^n 1^n| = 2n > |Q|$

Por tanto, por el th. de iteración $\exists x, y, z / \begin{cases} i) w = xyz \\ ii) y \neq \epsilon \\ iii) xy^k z \in L(A), \forall k > 0 \end{cases}$

$\frac{\cancel{x} \cancel{y} \cancel{z} M}{x \cancel{y^k} z} \rightarrow \cancel{x} \cancel{y} \cancel{z} M$

ya no hay el mismo núm. de os que de L .

Podemos distinguir 3 casos:

- $y \in \emptyset^+ \Rightarrow xy^k z \notin L(A) \forall k > 0$ ya que se intercalaría el nro de 0s y de 1s.
- $y \in 1^+ \Rightarrow xy^k z \notin L(A) \forall k > 0 \dots$
- $y \in \emptyset^+ 1^+ \Rightarrow xy^k z \notin L(A), \forall k > 0 \dots$ se intercalarían los 0s y 1s.

Luego, no hay ninguna posibilidad de que se verifique el th. de iteración.

Luego L no puede ser regular.

¡No poner EJEMPLO CONCRETO! \Rightarrow Es un 0 en el examen!!

Sobre eso, no podemos hacer cualquier tipo de razonamiento, porque es para cualquier cadena mayor que el nro de estados, y cualquier cadena que pongamos no cumple esto porque si no se verifica el th, entonces tampoco va a existir el nro de estados y no lo sabremos. No sabemos la long. de la cadena por lo tanto, NO PONER

Los aut. finitos no saben contar este tipo de cadenas porque no tienen una memoria. Cualquier expresión que implique contar no va a ser regular. Si por ejemplo, la maq. del café tiene que desvelar cambio, ya no es regular. No se puede contar en general, en reloj por ej sí porque siempre cuenta lo mismo pero una calculadora no.

A veces no es evidente que sea regular y a veces no salen bien en el th. de iteración porque. Veremos unos casos.

Ejemplo. ¿ $L = \{a^i b^j c^k, i \geq 1\}$ es regular?

Babila (tendríamos que poner el de antes).

Tenemos ese valor de $n = |Q|$, y sea $i := n$

Entonces $w = a^n b^j c^k, j \geq 1$ debería verificar el th. de iteración (long. reg.). Por tanto, deberían existir $x, y, z \in \Sigma^* / \begin{cases} i) \\ ii) \\ iii) \end{cases} \dots$

Podemos distinguir los siguientes casos:

- a) $y \in a^+ \Rightarrow xy^{k_2}$ desequilibra el n^o de as y de cs. $\notin L(A)$
b) $y \in a^+b^+ \Rightarrow xy^{k_2}$ intercalaria as y bs. $\notin L(A)$
c) $y \in a^+b^+c^+ \Rightarrow xy^{k_2}$ intercalaria bs, cs. $\notin L(A)$
d) $y \in b^+ \Rightarrow xy^{k_2} \in L(A)$.
e) $y \in b^+c^+ \Rightarrow xy^{k_2}$ intercalaria bs y cs. $\notin L(A)$.
f) $y \in c^+ \Rightarrow xy^{k_2}$ desequilibra el n^o de as y cs. $\notin L(A)$

→ no podemos demostrar que no se verifica el th. de alteración. Para que no se verifique, ninguno de los casos debe verificarse. Y por tanto, hay que buscar otra técnica para probar que L no es regular.

si borramos b^+ sacando la j), podríamos demostrar que no verifica el th porque solo se podría poner un b.

corolario: Sea $A = (\emptyset, \Sigma, \delta, q_0, F) / |\Sigma| = n$ ^{nº de estados.}

Entonces, cualquier trayectoria o camino en A de longitud mayor o igual que n tiene formalmente un ciclo.

demo: trivial.

Propiedades de cierre. (de los conj. reg, sólo vale para ellos).

def: Una operación n-únaria Θ sobre un conj C, es cerrada si: $\Theta(x_1, \dots, x_n) \subseteq \bigcup_{x_i \in C} C$

th. Los lenguajes son cerrados a la unión, concatenación y cierre de Kleene.

demo: Trivial por def. de expr. reg y por la equivalencia de expr. reg. y conj. reg.

th. Los conj. regulares son cerrados al complemento. Esto es si L es un conj reg sobre el alfabeto Σ , entonces $\Sigma \setminus L$ también es regular.

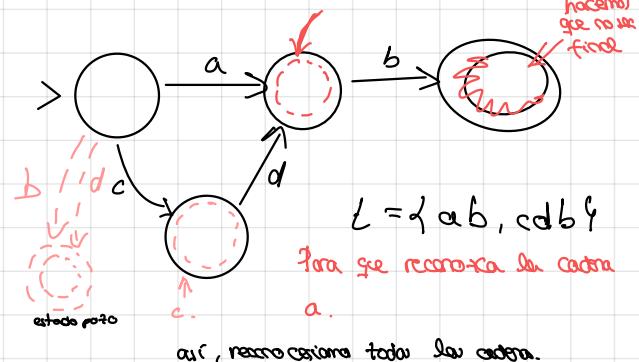
demo. $L \subseteq \Sigma^* / L$ sea regular t.g.d $\Sigma \subseteq$ los reg.

Si L es regular $\Rightarrow \exists A = (\emptyset, \Sigma, \delta, q_0, F) \text{ DFA} / L = L(A)$.

Sea $d \neq \emptyset$ lo consideremos nuevo estado a el autómata que construimos para recorrer $\Sigma^* \setminus L$. En particular definiremos:

$$f(d, a) := d, \forall a \in \Sigma$$

$$f(q, a) := d, \text{if } a \in \Sigma / \exists S(q, a) \text{ en } A.$$



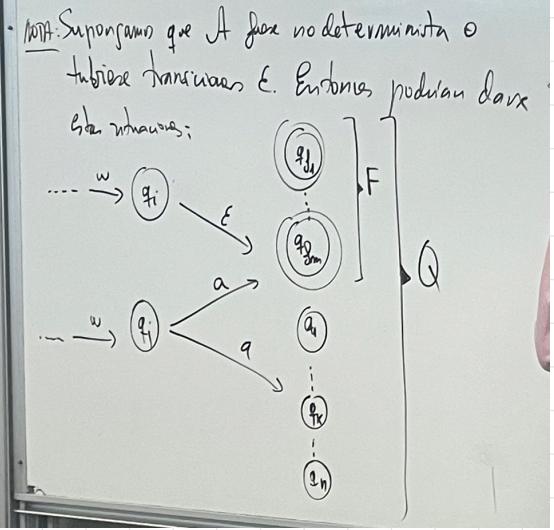
Definiremos entonces: $\tilde{A} = (\emptyset \cup \{d\}, \Sigma, \delta, q_0, \{q_0\} \neq \emptyset \cup \{d\})$.

Trivialmente (por construcción). $L(\tilde{A}) = \Sigma^* \setminus L$

Nota: Para que la demo sea válida, hemos de suponer que el FA original A es determinista, sin tránsiciones.

Nota: Siempre podemos eliminar las transiciones E de un AF.

Supongamos que A sea no det. o tuviese tránsiciones E. Entonces, podrían darse estos situaciones:



Entonces, en nuestra construcción " ω " no debería estar en I^* , sin embargo si estaría.

" ω " no debería estar en I^* , si embargo si estaría.

* PROPIEDADES DEL CIERRE (conj. regulares)

th: Los conj. reg. son cerrados a la intersección.

demo: Sean L_1 y L_2 dos conj. regulares, + q d $L_1 \cap L_2$ es un conj. reg.

$$\overline{L_1 \cup L_2} = L_1 \cap L_2$$

es regular

A = (...)

demonstración alternativa: bastará construir un automata finito / $T(A) = L_1 \cap L_2$
 $L_i, i=1,2$ regular $\Leftrightarrow \exists A_i = (Q_i, \Sigma, \delta_i, q_{0i}, F_i)$ AF / $T(A_i) = L_i, i=1,2$

Bastará tomar: $Q := Q_1 \times Q_2$ trivialmente (por construcción) $T(A) = L_1 \cap L_2$
 $\Sigma = \delta_1 \times \delta_2$
 $q_0 := (q_{01}, q_{02})$
 $F := F_1 \times F_2$

Ejemplo: ¿Es regular $L = \{a^n b^m c^n / n, m \geq 1\}$?

Supongamos que fuera regular:

$a^* b^* c^*$ es una exp. reg. luego define un conj. reg.

$\Rightarrow L \cap a^* b^* c^*$ también será un conj. reg.
||
 $\{abc^n / n \geq 1\}$ que ya vimos que no era regular

CONTRADICIÓN

def: Una sustitución, es una aplicación $s: I_1 \rightarrow \wp(I_2^*)$ con I_1 y I_2 alfabeto.

Luego, L no es regular.

Ejemplo: Sea $I_2 = \{\emptyset, 1\}$, $I_1 = \{a, b\}$ $L = \emptyset^* (\emptyset + 1) 1^*$

Consideramos la sustitución $\begin{cases} s(\emptyset) := a \\ s(1) := b^* \end{cases} \Rightarrow s(L) = a^* (a + b^*) b^* = (a^* a + a^* b^*) b^* = a^* b^* + a^* b^* b^* = a^* b^* + a^* b^*$

th: Los conjuntos reg. son cerrados por sustituciones.

demo: Sean I_1 y I_2 alfabetos, con $s: I_1 \rightarrow \wp(I_2^*)$ una sustitución. Sea r una exp. reg. en I_1^* (ω -sustitución).

to es, un conj. reg), t.g.d $s(r)$ también es un conjunto regular.

Partida vez que $s(r)$ es una expr. reg.

Lo haremos por inducción en el nro de operaciones de " r " que denotaremos por $|r|$

$$|r|=0 \Leftrightarrow \begin{cases} r \in \Sigma_1 \Rightarrow s(r) \in \mathcal{P}(\Sigma_1^*) \\ r = \emptyset \Rightarrow s(r) = \emptyset \\ r = \varepsilon \Rightarrow s(r) = \varepsilon \end{cases}$$

{ En cualquier caso, es regular.

En cualquier caso, tendremos una expr. regular, luego un conj. regular.

$|r| \leq n$ Supuesto correcto

$|r| = n+1$ Hay 3 subcasos:

a) $r = r_1 + r_2 / |r_i| \leq n$ para $i=1,2$. Por hipótesis de inducción $s(r_i)$ es una expr. reg, $i=1,2$

Teniendo en cuenta que $s(r_1 + r_2) = s(r_1) + s(r_2)$

\Rightarrow

$\Rightarrow s(r) = s(r_1) + s(r_2)$ es una expr. regular.

b) $r = r_1 \cdot r_2 / |r_i| \leq n$ para $i=1,2$. Por inducción $s(r_i)$ es reg $\forall i=1,2$

Dado que $s(r_1 \cdot r_2) = s(r_1) \cdot s(r_2)$

\Rightarrow

$\Rightarrow s(r) = s(r_1) \cdot s(r_2)$

c) $r = r_1^*$ / $|r_1| \leq n$, luego por inducción $s(r_1)$ es una expr. reg $\Rightarrow s(r) = s(r_1)^*$ es expr. reg.

Dado que $s(r) = s(r_1^*) = s(r_1)^*$

Corolario: Los conj. regulares son cerrados por homomorfismo.

Demo: Un **homomorfismo** es una sustitución $s: \Sigma_1 \rightarrow \Sigma_2$, esto es una sustitución que cambia un símbolo en Σ_1 por un (y solo uno) Σ_2

Luego se dice es trivial.

Th: Los conj. regulares son cerrados por homomorfismo inversos

Nota: Esto es, que si $h: \Sigma_1 \rightarrow \Sigma_2$ es un homomorfismo y r es una expr. reg. en Σ_2^* , entonces $h^{-1}(r)$ también es reg.

Ejemplo: ¿Es regular $\mathcal{L}_1 = \{a^n b a^n, n \geq 1\}$?

Supongamos que \mathcal{L}_1 es regular entonces también debería serlo cualquier conjunto resultado de aplicar sobre \mathcal{L}_1 operaciones para las que los regulares son cerrados.

Vamos a considerar los homomorfismos h_1 y h_2 siguientes:

$$\begin{array}{ll} h_1(a) = a & h_2(a) = \emptyset \\ h_1(b) = ba & h_2(b) = 1 \\ h_1(c) = a & h_2(c) = 1 \end{array} \quad \left\{ \begin{array}{l} \text{Entonces} \\ \xrightarrow{\hspace{1cm}} \end{array} \right.$$

Si L_1 regular, entonces $h_2(\overbrace{h_2^{-1}(\{\underbrace{a^n b a^n}_{L_1}, n \geq 1\}) \cap \underbrace{a^* b c^*}_{L_2}}^{\text{reg}})$ tambien deberia ser reg.

$\xrightarrow{\text{reg}}$
 \parallel

$$h_2((a+c)^n b (a+c)^{n-1} \cap a^* b c^*)$$

$$h_2(a^n b c^{n-1}) = \emptyset \cap L_2^{n-1} = \emptyset \quad \text{que ya vimos que no es regular}$$

Luego absurdo, luego L_1 no es regular.

Def: Sean L_1 y L_2 dos lenguajes, definiremos el cociente de L_2 por L_1 como $L_1 / L_2 = \{x \in \Sigma^* / \exists y \in L_2 \quad xy \in L_1\}$

Th: Los conjuntos reg son cerrados al cociente por cualquier conj.

Demo: Sea $L_1 = T(A_1) / A_1 = (\emptyset, \Sigma, \delta, q_0, F_1)$ AF. Esto es L_1 , reg } t.g.d L_1 / L_2
Sea L_2 es un lenguaje arbitrario. } es regular

Bastara demostrar que $\exists A_2 = (\emptyset, \Sigma, \delta, q_0, F_2)$ AF / $T(A_2) = L_1 / L_2$.

Definimos $F_2 := \{q \in \emptyset / \exists y \in L_2, S(q, y) \in F_1\}$

Entonces $x \in T(A) \iff \delta(q_0, x) \in F_2 \iff \exists y \in L_2, S(q_0, xy) \in F_1 \iff xy \in L_1$

$$\begin{array}{c} \uparrow \\ x \in L_1 / L_2 \\ \downarrow \end{array}$$

Jerarquía

Vacuidad, finitud, infinitud de un lengu. reg.

POSSIBILIDAD DE EXISTENCIAS

Th. (de Rado) Si ω es una cadena aceptada por un AF $A = (\emptyset, \Sigma, \dots) / |\omega| = n$, entonces:

a) $T(A) \neq \emptyset \iff \exists$ una cadena aceptada de long. $< n$.

b) $T(A)$ es infinito $\iff \exists$ una cadena aceptable de longitud $l / n \leq l < \infty$

continuacion en el

25/04.

a) " \Leftarrow " trivial

" \Rightarrow " $T(A) \neq \emptyset$, t.g.d $\exists \omega \in T(A) / |\omega| < n$ (eso es lo que hay que demostrar).

$T(A) \neq \emptyset \Rightarrow$ podemos escoger $\omega \in T(A)$

$|\omega|$ sea la máxima. Demostremos que $|\omega| < n$

Supongamos que no es así, esto es, que $|\omega| \geq n$

Entonces (th iteración long. reg) $\exists x, y, z \in \Sigma^* / \frac{\omega = xyz}{|y| \geq n}$

En particular, (iii) si $k = \emptyset, xy^kz = xz \in T(A), \forall k \geq 0$

$$|xz| < |xy^kz| = |\omega| \quad \text{Absurdo (I)}$$

longo (ω) $> n$

b) " \Leftarrow " $\exists \omega \in T(A) / n \leq |\omega| < 2n \Rightarrow$ (th. iteración) \Rightarrow (iii) \Rightarrow $xy^kz \in T(A), \forall k \geq 0$

con $\omega = xyz$

Luego hay ω aceptado en $T(A)$

" \Rightarrow " $|T(A)| = \infty$, t.g.d $\exists \omega \in T(A) / n \leq |\omega| < 2n$

(I) Supongamos que no es así, esto es, $\exists w \in T(A) / n \leq |w| < 2n$
Supongamos $w \in T(A)$ con la longitud más pequeña $\geq 2n$

En particular, $|w| \geq n$, luego por el th. de iteración en lenguajes. bla bla i) ii) iii)... teniendo en
iii) $k = \emptyset$, tendremos que $w = xy^z / x, y^{\emptyset} + \in T(A)$

donde 1 caso $|x| \leq 2n$ contradicción con (I)

2 caso $|x| > 2n$ contradicción con (II)

$$T(A) / n \leq |w| < 2n.$$

Th de Moore: Existe un algoritmo para determinar si 2 automóviles finitos reconocen el mismo lenguaje.

demo: Supongamos que $A_i = (Q_i, \Sigma_i, \delta_i, q_{0i}, F_i)$ $i=1,2$ AF

Supongamos $T(A_i)$, $i=1,2$, sabemos que son lenguajes reg.

Entonces:

$$\left[\overline{T(A_1)} \cap \overline{T(A_2)} \right] \quad \left[\overline{\overline{T(A_1)} \cap T(A_2)} \right]$$

también es regular, y se denominaría L donde $L \neq \emptyset$ si $T(A_1) \neq T(A_2)$ y donde además sabemos es decidable que $L \neq \emptyset$

Luego, también es decidable que $T(A_1) \neq T(A_2)$

EXERCICIOS.

1. Sean L_1 conj. reg. ¿ $L_1 \setminus L_2$ es reg?

$$L_1 \setminus L_2 = \underbrace{L_1}_{\text{reg}} \cap \overline{\underbrace{L_2}_{\text{reg}}} \text{, luego reg.}$$

→ Cogemos el conjunto de instrucciones de L_1 y le quitamos las de L_2 .

¹ $a^k b^m \Rightarrow$ Esto si se recorre U
 $a^n b^n \Rightarrow$ Esto no porque habrá que costar.

2. Sea en $\Sigma = \{0, 1\}$, entonces si son regulares:

a) los códigos que terminan en $\phi\phi$.

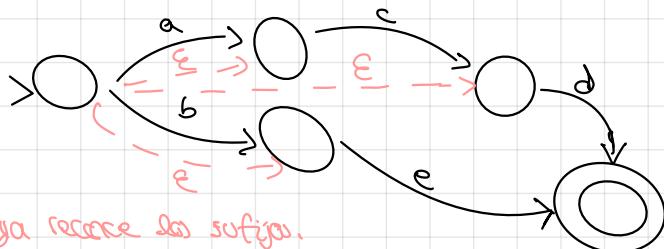
Si, dado que es el conjunto: $(0+1)^* \phi\phi$
códigos de 00 y 11

→ En los apartados de clase estaba mal.

b) los códigos donde cada grupo de 5 símbolos termina en $\phi\phi$.
Sí, $((0+1)(0+1)(0+1)\phi\phi)^*$ luego es regular.

3. Sea L_1 un conj. reg. y $L_2 := \{w \in \Sigma^* / \exists j \in \Sigma^d, y w \in L_1 \text{ f. d } L \text{ es regular?}$

Aquí no hace falta escribir, tiene punto de ser regular.



$$L = \{acd, bcf\}$$

$$Sof = \{cd, d, e\}$$

Sea $A_2 = (\emptyset, \Sigma, \delta_2, q_0, F)$ con $\delta_2 = (\emptyset, \Sigma, \underline{\delta_1}, q_0, F) / T(A_1) = L_1$

dnde $\delta_2 = (q, a) := \delta_1 = (q, a), \forall q \in Q, \forall a \in \Sigma$.

$\delta_2 = (q_0, \emptyset) := \{q \in Q, q \neq q_0\}$ tricadicamente $T(A_2) = L_2$, luego L_2 es regular.

4. Sea $L := \{a^i, i \in \mathbb{N}$: natural primo} d L es regular?

Supongamos qe lo es $\exists A(\emptyset, \Sigma, \delta, q_0, F)$, $|Q| = n$ tal qe $L = T(A)$

Entonces (th. iteración) $\forall \omega \in T(A), |\omega| > n, \exists x, y, z \in \Sigma^*$ / $\begin{cases} w = xyz \\ y \neq \epsilon \\ xy^kz \in T(A), \forall k \geq 0 \end{cases}$

Sea $\geq n$, estmoo en las condiciones del th. de iteración para:

Cualquier $a^m \in T(A)$. Supongamos en qd $k := m+1$, entonces:

$x y^k z \in T(A)$

En particular, $\exists t = m+1 \in \mathbb{N}$ tal qe $\Rightarrow x y^{m+1} z \in L$, pero

a^t ; i primo

$$\left. \begin{array}{l} |x y^{m+1} z| = |x y z| + |y^m| = m + m |y| = m(1 + |y|) \\ x y^{m+1} z = a^i \end{array} \right\} \Rightarrow$$

$\Rightarrow i$ no primo $\Rightarrow a^i \notin L$ absurdo \Rightarrow vde de supos qe es regular, luego L no regular.

5. Sea $L := \{a^i b^j / \underbrace{i, j}_{i \neq j} \geq 0\}$ d L es regular?

se necesita contrar

el nmro i, j.

$$\overline{L} \setminus L = \{a^i b^j\}^*$$

Supongamos L conj. reg. $\Rightarrow \overline{L}$ reg } \Rightarrow
 $a^* b^*$ conj. reg

$$\Rightarrow \overline{L} \cap a^* b^* = \{a^i b^i, i \geq 0\} \text{ reg.}$$

absurdo

$\{a^i b^i, i \geq 0\}$

Vamos a entrar en: lenguajes indep. del contexto (LICs).

(seguiremos un camino similar a lenguajes regulares).

Lema de iteración en lenguajes indep. del contexto.

Sea L un CFL, entonces \exists un $K \geq 1 / \forall z \in L, |z| \geq K$ se clasifica que $\exists u, v, w, x, y \in \Sigma^*$

- i) $ux \notin E, wy \in L$
- ii) $|uvwx| \leq K$
- iii) $\forall i \geq 0, uv^iwx^i \in L$

esta longitud

La demostración no se hace.

Ejemplo: $\{a^n^2 / n \geq 1\}$ es lenguaje indep. del contexto? (no lo vamos a ver).

No vale elegir una cadena concreta !! \Rightarrow IMPORTANTE. \Rightarrow No podemos un valor concreto nosotros no sabemos qué es K , no sabemos a qué DE R .

Lo haremos por reducción al absurdo.

Supongamos que es el contrario... bla bla... (def en arriba).

Sea entonces $n = k$, entonces $|a^{n^2}| = |a^{k^2}| = k^2 \geq K$
yo no sé qué es lo correcto!!

Luego entonces en las hipótesis del teorema, por lo que $\exists u, v, w, x, y \in \Sigma^*$ tales que $a^{n^2} = uvwxy$

donde:

$$\begin{aligned} ux \neq \epsilon & (\text{por i}) \Leftrightarrow |ux| > 0 \\ & \text{por q i paro} \\ k^2 &= |a^{n^2}| = |a^{k^2}| = |uvwxy| \end{aligned} \quad \left\{ \begin{aligned} &\Rightarrow k^2 < |vwx^2y| \\ & \text{ya no sé qué es lo correcto!!} \end{aligned} \right.$$

vamos a buscar en
bambas tal que sea
imposible que no pertenezca a ese lenguaje.
 $\Rightarrow k^2 < |vwx^2y| \leq K$
 $\Leftrightarrow (K+1)^2 \leq K$, esto
es, $\nexists i \in \mathbb{N} /$
 $|vwx^2y| = i^2$, esto
es $|vwx^2y| \notin L$, contradicción con el
punto 3 (iii) luego L no es CFL.

Por otro lado:

$$\begin{aligned} |vwx| \leq K & (\text{por ii}) \\ w \neq \epsilon & \end{aligned} \quad \left\{ \begin{aligned} &\Rightarrow |ux| < K \\ & |vwx| = K \end{aligned} \right.$$

Ejemplo: $\{a^n b^n c^n, n \geq 1\}$ es CFL?

Para ello, aplicaremos el lema de iteración en CFGs.

Supongamos que L es un CFL, entonces $\exists k \geq 1 / \forall z \in L, |z| \geq k$.

Tenemos que:

$$\begin{aligned} z &= uvwxy \\ \text{y} &= \begin{cases} i) ux \neq \epsilon \\ ii) |vwx| \leq k \\ iii) \forall i \geq 0, uv^iwx^i \in L \end{cases} \end{aligned}$$

$|vwx| \leq k \Rightarrow vwx$ no puede contener a, b y c simultáneamente $\Rightarrow uvwy$ contiene k o $> k$ c. (I)

Por otro lado, $|ux| > 0$ (por i) $\Rightarrow |uvwy| < 3R$ (II)

$$\text{con } |uvwy| = |a^K b^K c^K| = 3K$$

* contradicción con iii (teniendo $i=0$, ya que $wx^0y = \underline{\underline{wxy}} \in L$)



$$\exists r / \underline{\underline{wxy = a^r b^r c^r}}$$

GRAMÁTICAS REDUCIDAS.

Sea $G = (N, \Sigma, P, S)$ una CFG y sea $X \in N^*$, decimos que X es inicial si $\exists x \in \Sigma^* / X \stackrel{*}{\Rightarrow} x$

Ejemplo: Sea la gramática $E \rightarrow E + E$
 $| T$
 $| F$
 $F \rightarrow F * E$
 $| (T)$
 $| a$
 $T \rightarrow E - T$

Tu a ser un símbolo inicial, porque nunca se puede derivar de Q una cadena de terminales

Th. (Bar-Hillel, Perles y Shamir):

Teorema: (Bar-Hillel, Perles y Shamir):

Sea $G = (N, \Sigma, P, S)$ una CFG, entonces existe $G' = (N', \Sigma', P', S')$ otra CFG tal que:

- 1) $L(G) = L(G')$
- 2) $L(G) = \emptyset \Rightarrow P' = \emptyset$
- 3) $L(G) \neq \emptyset \Rightarrow \forall A \in N, \exists x \in \Sigma^* / A \stackrel{*}{\Rightarrow} x$

dem: Definimos $W_i := \{A \in N / \exists x \in \Sigma^*, A \stackrel{*}{\Rightarrow} x \}$, variables de G están en un solo paso de derivación. Si yo tengo:

$A \rightarrow \text{hola } P$
 $\triangle_h \quad A \text{ deriva a hola en un sólo paso, es stel.}$

$W_{i+1} := W_2 \cup \{A \in N / \exists x \in (\Sigma \cup W_K)^*, A \stackrel{*}{\Rightarrow} x \}$, $A \rightarrow a \in P \quad | \quad (A \in W_i)$
 esto es, las variables que son stel... .

Entonces tenemos que:

- i) $W_K \subseteq W_{K+1}$
- ii) $\exists i / W_i = W_{i+1} \Rightarrow W_i = W_{i+m}, \forall m > 0$
- iii) $W_m = W_{m+1}, n = |N|$
- iv) $A \in W_i \Leftrightarrow A \stackrel{*}{\Rightarrow} x / x \in \Sigma^* \text{ en un árbol de derivación } T \text{ de altura} \leq i$
- v) $W_n = \{A \in N / \exists x \in \Sigma^*, A \stackrel{*}{\Rightarrow} x\}, n = |N|$

$\triangle_h \quad A$
 $\triangle_h \quad \text{texto.}$
 $\triangle_h \quad \text{hola que tal.}$

Entonces basta tener como $S' = (N', \Sigma', P', S')$

$$N' := N \cap \text{Wm}$$

$$\Sigma' := \Sigma$$

$$S' := S$$

$P' := \{P : A \rightarrow \alpha / \text{tanto } A \text{ como } \alpha \text{ variable nfa, son fdo}\}$

Ejemplo: d'eliminar inútiles a $E \rightarrow E+E$

$$\cancel{TT}$$

$$F \rightarrow F * E$$

$$\cancel{IF}$$

$$\cancel{| T}$$

$$\cancel{T \rightarrow E-T}$$

$$\cancel{| \alpha}$$

$$\underline{\text{sol:}} \quad w_1 = \{F\}$$

$$w_2 = \{F + E, F * E\} \quad \text{y son iguales, entonces (ii), } w_{1,2} = \{F, E\}, \text{ luego la gramática } \underline{\text{elimina}}$$

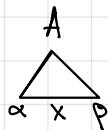
$$w_3 = \{F, E\} \cup \{E\} = \{F, E\}$$

9/05

Símbolos inaccesibles.

def: Sea $S = (\dots)$, una CFG, y sea $A \in N, X \in N \cup I$. Decimos que X es accesible desde A si $\exists \alpha, \beta \in (N \cup I)^*$

$$\overline{A \xrightarrow{\alpha} \beta X \beta}$$



Ejemplo: Sea CFG dado por:

$$S \rightarrow aBa$$

$$B \rightarrow sb$$

i) no es accesible desde S ! No podemos construir este árbol



$$C \rightarrow abb$$

$$\Rightarrow D \rightarrow ac$$

Th. (Bar-Hillel, Perles y Shamir)

Sea $G = (N, \Sigma, P, S)$ un CFG, entonces $\exists S' = (N', \Sigma', P', S')$ CFG /

$$a) L(G) = L(S')$$

b) Todas símbolos $N \cup I$ es accesible a partir de S .

Sea $A \in N$ definimos $w_k(A) := A \xrightarrow{*} \text{Accesible desde } A \text{ en meno de } k \text{ derivaciones.}$

$$w_{k+1}(A) := \underbrace{w_k(A)}_{\text{accesible desde } A} \cup \{X \in N \cup I / \exists \alpha, \beta \in (N \cup I)^*, \exists \beta \in w_k(A) \text{ tal que } \beta \xrightarrow{*} \alpha \times \beta \in P'\}$$

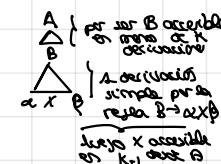
accesible desde A
en meno de k der.
vacíos y por tanto
también accesible en meno
de $k+1$ derivaciones.



i) Son los elementos de F_{k+1} accesibles desde A en meno de $k+1$ derivaciones? Si

Sea $X \in F_{k+1}$ entonces (por definición)

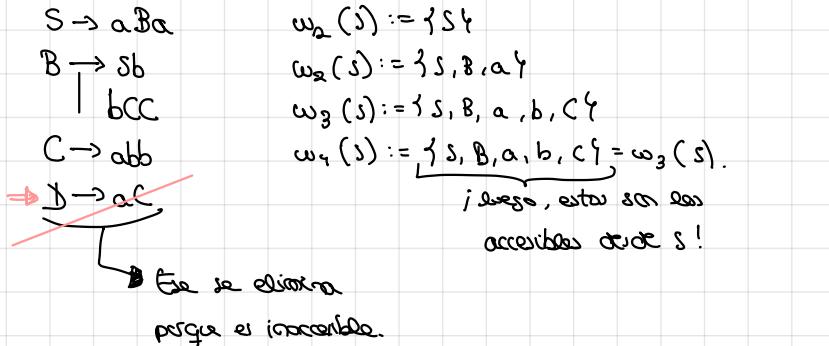
luego $w_{k+1}(A)$ es los símbolos accesibles desde A
en meno de $k+1$ derivaciones.



los $w_k(A)$ tienen cierta propiedad:

- i) $w_k(A) \subseteq w_{k+1}(A)$
- ii) $w_k(A) = w_{k+1}(A) \Rightarrow \forall m > k, w_k(A) = w_{k+m}(A)$
- iii) $w_n(A) = w_{n+1}(A), n = |N|$
- iv) $w_n(A) = \{x \in N\cup\Sigma / \exists \alpha, \beta \in (N\cup\Sigma)^*, A \xrightarrow{*} \alpha x \beta\}$

Ejemplo: Sea la CFB dada por:



def: Sea $G = (N, \Sigma, P, S)$ una CFB, decimos que es **reducida** si verifica una de estas propiedades:

- a) $P = \emptyset$
- b) $\forall x \in N\cup\Sigma, \exists \alpha, \beta \in (N\cup\Sigma)^*, \exists \omega \in \Sigma^* / \underbrace{S \xrightarrow{*} \alpha X \beta}_{\substack{x \text{ es accesible desde} \\ S}} \xrightarrow{*} \omega$

PROPIEDADES DE COMBINACIÓN DE CFGs.

Th: Los CFG son cerrados por unión.

demo: Sean $L_i = L(G_i)$, $G_i = (N_i, \Sigma_i, P_i, S_i)$ CFGs $\forall i \in \{1, 2\}$ $\exists G = (N, \Sigma, P, S)$ CFG: $L(G) = L_1 \cup L_2$
Podemos suponer siempre que $N_1 \cap N_2 = \emptyset$

Basta considerar $N = N_1 \cup N_2$, $\Sigma = \Sigma_1 \cup \Sigma_2$,
 $P := P_1 \cup P_2 \cup \{S \rightarrow S_1, S \rightarrow S_2 \mid S \notin N_1 \cup N_2\}$

Entonces se verifica th.

def: Los CFG son cerrados por sustitución y homomorfismo.

def: Los CFG no son cerrados para la intersección.

demo: lo vamos a ver en un contraejemplo.



demo.

Consideremos los lenguajes $\begin{cases} L_1 = \{a^n b^n c^i / n \geq 1, i \geq 1\} \\ L_2 = \{a^i b^n c^n / i \geq 1, n \geq 1\} \end{cases}$, entonces:

- a) L_1 es CFL, $i=1,2$
- b) $L_1 \cap L_2 = \{a^n b^n c^n / n \geq 1\}$
- c) $L_1 \cap L_2$ no es un CFL.

a) caso L_1 Trivialmente podemos ver que está generado por la CFG

$$G_1 \text{ cuyas reglas son: } \begin{array}{l} S \rightarrow AC \\ A \rightarrow aAb \\ \quad | ab \\ C \rightarrow Cc \\ \quad | c \end{array}$$

caso L_2 Está generado por la CFG G_2 de reglas:

$$\begin{array}{l} S \rightarrow AB \\ B \rightarrow bBc \\ \quad | bc \\ A \rightarrow Aa \\ \quad | a \end{array}$$

sin embargo $C_1 \cap C_2$
que hemos visto no
es independiente del
contenido.

b) Trivial

c) Ya demostrado en un ejemplo anterior.

demonstrado

def) \Rightarrow

En cambio, son cerrados para la intersección con conj. regulares.

(no esté en los apuntes pero eso sí se da).

def: No son cerrados para el complemento. (apuntes)

Ejemplo:

Sea el lenguaje $L_1 = \{ww / w \in \{a, b\}^*\}$, veremos que no es un CFL.

Lo demostraremos por reducción al absurdo. Supongamos que L_1 es un CFL, entonces $L_2 := L_1 \cap a^m b^m a^m b^m = \{a^m b^m a^m b^m / m \geq 1\}$ sería también un CFL. Veremos que L_2 no es un CFL, con lo cual habremos probado nuestra tesis primera.

L_2 no es un CFL

Supongamos L_2 CFL \Rightarrow (Lema de iteración de CFL) $\Rightarrow \exists k \geq 1 / \forall z \in L_2$, $|z| \geq k$, tenemos que:

$$z = uvwxy \quad \left/ \begin{array}{l} i) v \neq \epsilon, w \neq \epsilon \\ ii) |vwx| \leq k \\ iii) \forall i \geq 0, uviwx^i y \in L_2 \end{array} \right.$$

Sea pues $z := a^k b^k a^k b^k = uvwxy \quad \left\{ \begin{array}{l} \Rightarrow vwx \text{ no puede contener} \\ |vwx| \leq k \\ \text{simultáneamente } k \text{ a's y } k \text{ b's} \end{array} \right. \Rightarrow$

$$\Rightarrow uwy \text{ contiene } k \text{ 'a's sucesivas y } k \text{ 'b's sucesivas} \quad \left. \begin{array}{l} |vx| \neq \emptyset \Rightarrow |vwx| > \emptyset \\ |akbkakbk| = |uvwxy| = 4k \end{array} \right\} \Rightarrow |uwy| \leq 4k$$

$\Rightarrow uwy \notin L_2$, abriendo pn iii), tomando $i=0$. deshidado

TEOREMA DE ITERACIÓN DE LENGUAJES REGULARES.

Sea $A = (\Phi, \Sigma, S, q_0, F)$ un AFD | $|Q| = n$. Entonces $\forall z \in T(A)$, $|z| = m > n$ se verifica que $\exists xyz \in \Sigma^*$

- i) $w = xyz$
- ii) $y \neq \epsilon$
- iii) $\forall k \geq 0, xy^kz \in L$

TEOREMA DE ITERACIÓN DE LENGUAJES INDEPENDIENTES.

Sea L un LIC, entonces $\exists k \geq 1 / \forall z \in L, |z| \geq k$ entonces se verifica que $\exists z = uxwy$

- i) $wx \neq \epsilon$ $w \neq \epsilon$
- ii) $|uxw| < k$
- iii) $\forall i \geq 0 \quad ux^iwxy \in L$

2. (1 pto) Razonar la verdad o falsedad de la afirmación siguiente:

"El conjunto $L = \{a^{n^2}, n \geq 1\}$ es un lenguaje independiente del contexto."

Vamos a demostrar que no lo es demostrando que no verifica el teorema de iteración de lenguajes independientes del contexto. Lo haremos por reducción al absurdo. Supongamos que L es regular, entonces $\exists K \geq 1 / \forall z \in T(A), |z| \geq K$, entonces verifica el teorema y por tanto $z = uxwy$

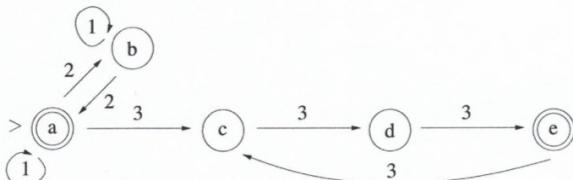
- i) $wx \neq \epsilon$ $w \neq \epsilon$
- ii) $|uxw| < K$
- iii) $\forall i \geq 0 \quad ux^iwxy \in L$

Supongamos que $n = k$, entonces $|z| = |a^{k^2}| = k^2 \geq K$, entonces se verifica el teorema de it. y cumple los condicines anteriores, dnd:

$$\begin{aligned} & \left. \begin{array}{l} wx \neq \epsilon \quad wx > 0 \\ |uxwy| \geq k^2 \end{array} \right\} \quad |ux^2wx^2y| \geq k^2 \\ & \left. \begin{array}{l} w \neq \epsilon \\ |uxw| < K \\ |uxwy| \geq k^2 \end{array} \right\} \Rightarrow |ux^2wx^2y| < k^2 + K \end{aligned} \quad \left. \begin{array}{l} K^2 \leq |ux^2wx^2y| \leq k^2 + K \\ \hline \end{array} \right\}$$

Esto verifica que $\exists l \quad ux^2wx^2y = l^2$
Por tanto, $ux^2wx^2y \in L$ pero Aburdo
por iii) $ux^2wx^2y \notin L$ No cumple el th.

3. (1.5 ptos) Calcular la gramática regular asociada al AF de la figura. Justificar los pasos ejecutados en el cálculo.



$$\begin{array}{lllll} A \rightarrow 1A & B \rightarrow 1B & C \rightarrow 3C & D \rightarrow 3E & E \rightarrow 3C \\ A \rightarrow 2B & B \rightarrow 2B & & & E \rightarrow \emptyset \\ A \rightarrow 3C & & & & \\ A \rightarrow 3 & & & & \end{array}$$

5. (0.5 ptos) Enunciar el Teorema de iteración en lenguajes regulares.

Accesible en la clase del día 30/04/21, en la plataforma Moovi.

6. (1 pto) Razonar la verdad o falsedad de la afirmación siguiente:

"El conjunto $\mathcal{L} = \{a^n b^m c d^m e^{n+2}, \text{ tal que } n \geq 1, m \geq 0\}$ es un lenguaje regular."

Vamos a demostrar que no es regular por el teorema de iteración de lenguajes regulares. Supongamos que \mathcal{L} es regular, entonces se verificaría el teorema para lenguajes regulares.

Supongamos $n=10$, entonces trivialmente $w = a^n b^m c d^m e^{n+2}$, $m > 0$ entonces debemos verificar dicho teorema y verificar que $\exists xyz \in \Sigma^*$

i) $w = xyz$
ii) $y \neq \epsilon$
iii) $\forall k \geq 0, xy^k z \in \mathcal{L}$

donde la condición para "y" sería:

$y \in a^* \Rightarrow xy^k z \in \mathcal{L}$ porque desequilibraría el número de a y e .

$y \in \dots$

$$L = \{a^n b^n c^n d^n, n \geq 0\} \text{ lenguaje indefinido.}$$

Vamos a demostrar que no se verifica que no cumple el teorema de泵送定理 de los lenguajes de contexto. Lo haremos por reducción al absurdo. Supongamos que L es un CFB, entonces $\exists K > 1 / \forall z \in L |z| \geq K$.

Supongamos que $n = K$ entonces $|z| = |a^k b^k c^k d^k| = 4K \Rightarrow 4K$, cumple la hipótesis del teorema y por tanto debemos verificar que $\exists z = uvwxy /$

- i) $u \neq \epsilon$ $w \neq \epsilon$
- ii) $|uvx| < K$
- iii) $Hi > 0$, $uv^i w x^i y \in L$

$$|z| = |uvwxy| = 4K$$

$|uvw| < K \Rightarrow$ no puede contener a, b, c, d consecutivamente.
 $uvw \in L \Rightarrow$ puede contener Ka, Kb, Kc, Kd siquiera.

$$\left. \begin{array}{l} ux \neq \epsilon \quad |ux| > 0 \\ |z| = |uvwxy| > 4K \end{array} \right\} |vwy| < 4K \Rightarrow \text{no puede contener} \\ \text{Ka, Kb, Kc, Kd}$$

$\Rightarrow uvwxy \notin L$ pero
 por iii) $i=0$ $uvw \in L$

Absurdo