TEMA 1

Un <u>alfabeto</u> es un conjunto finito cuyos elementos se llaman <u>símbolos</u> o letras.

Una <u>palabra</u> sobre el alfabeto A es una sucesión finita de elementos de A.

El conjunto de todas las palabras sobre un alfabeto A se nota como A^*

<u>Longitud</u> de una palabra, $|u|$ es el nº símbolos que tiene.

Palabra vacía ε.

El conjunto de cadenas sobre un alfabeto A (excluyendo la palabra vacía) se denota como A^+.

Concatenación

$u, v \in A^*$, $u = a_1 \dots a_n$, $v = b_1 \dots b_m$, $\quad u.v = a_1 \dots a_n b_1 \dots b_m$

$$\left.\begin{array}{ll} 1) \ |u.v| = |u| + |v| & \forall u, v \in A^* \\ 2) \ u.(v.w) = (u.v)w & \forall u, v, w \in A^* \\ 3) \ u.\varepsilon = \varepsilon.u = u & \forall u \in A^* \end{array}\right\} \text{Monoide}$$

$u \in A^*$, v <u>prefijo</u> de u si $\exists z \in A^* : vz = u$.

$u \in A^*$, v <u>sufijo</u> de u si $\exists z \in A^* : zv = u$

$u^n = u^{n-1}.u$ (concatenación n veces consigo misma)

sea $u \in A^*$: $u = a_1 \dots a_n$, la cadena inversa es $u^{-1} = a_n \dots a_1$

Un <u>lenguaje</u> sobre el alfabeto A es un subconjunto del conjunto de las cadenas sobre A. $L \subset A^*$

A^* siempre es numerable.

El conjunto de lenguajes sobre A^* nunca es numerable.

Concatenación de lenguajes

$L_1 L_2 = \{u_1 u_2 : u_1 \in L_1 , u_2 \in L_2\}$

·) $\emptyset L = L \emptyset = \emptyset$

··) $\{\varepsilon\} L = L\{\varepsilon\} = L$

·) $L_1(L_2 L_3) = (L_1 L_2) L_3$

La clausura de Kleene de L es

$L^* = \bigcup_{i\geq 0} L^i$ $\qquad$ $L^+ = \bigcup_{i\geq 1} L^i$

$L^* = L^+$ si $\varepsilon \in L$

$L^* = L^+ - \{\varepsilon\}$ si $\varepsilon \notin L$

Lenguaje inverso

$L^{-1} = \{u \mid u^{-1} \in L\}$

(prefijos) $\in A^*$
Cosas que se completan con otras cosas y se quedan en el lenguaje

La cabecera es $CAB(L) = \{u \mid u \in A^* \text{ y } \exists v \in A^* \text{ tq } uv \in L\}$

$h : A_1^* \longrightarrow A_2^*$ homomorfismo $h(\varepsilon) = \varepsilon$

$h(uv) = h(u)\,h(v)$ $\qquad$ $h(a_1 \ldots a_n) = h(a_1) \ldots h(a_1)$

Gramática General (V, T, P, S)

V es alfabeto de variables.

T es alfabeto de símbolos terminales.

P conjunto finito de pares, reglas de producción.

S símbolo de partida.

Lenguaje generado por una gramática al conjunto de cadenas formadas por símbolos terminales y derivables a partir del símbolo de partida

$L(G) = \{u \in T^* : S \stackrel{*}{\Rightarrow} u\}$

Una gramática sirve para determinar un lenguage, pero un lenguage está determinado por muchas gramáticas. Esto es, si encontramos una gramática regular, entonces el lenguage es regular, pero podemos encontrar una gramática NO regular que genere un lenguage regular, porque haya otra gramática que lo genere que SI sea regular

Jerarquía de Chomsky

- Tipo 0: lenguages recursivamente numerables.
  No tiene restricciones
- Tipo 1: lenguages dependientes del contexto.
  Si las producciones son de la forma $\alpha_1 A \alpha_2 \to \alpha_1 \beta \alpha_2$
  con $\alpha_1, \alpha_2, \beta \in (V \cup T)^*$, $A \in V$
- Tipo 2: lenguages Independientes del contexto
  Si toda regla de producción es $A \to \alpha$, $A \in V$, $\alpha \in (V \cup T)^*$
- Tipo 3: languages Regulares
  De la forma $A \to uB \mid u$, $u \in T^*$, $A, B \in V$.

→ Unión de L_1 con L_2 es $S \to S_1 \mid S_2$, S_1 y S_2 símbolos iniciales

Tema 2

Un automata finito es la quíntupla $M = (Q, A, \delta, q_0, F)$

- Q es un conj. finito llamado conj. de estados.
- A es el alfabeto de entrada
- δ es la función de transición, $\delta: Q \times A \longrightarrow Q$
- q_0 elemento de Q, estado inicial.
- F un subconj de Q, que son los estados finales.

Diagrama de transición:

- 1 nodo por estado.
- Por cada transición $\delta(q,a) = p$, arco de q a p con etiqueta a
- Estado iniciales $>$ y estados finales con ◎.

Un autómata finito No determinista:

- Puede haber estados con 2 transiciones para una misma entrada o que No tengan ~~tran~~sición.
- Se puede hacer determinista añadiendo un nodo error

El lenguaje aceptado por un autómata finito no-determinista se puede expresar como

$$L(M) = \{u \in A^* : \delta^*(q_0, u) \cap F \neq \emptyset\}$$

⊙ Un lenguaje puede ser aceptado por un autómata finito determinista $\Leftrightarrow$ puede ser aceptado por un finito no determinista

Un autómata no-det y su aut. det asociado aceptan el mismo lenguaje.

La clausura de un estado es el conjunto de estados a los que puedes ir con transiciones nulas.

un aut. finito No det. con transiciones nulas es un aut. fin. No det donde algunas transiciones son la palabra vacía.

- L aceptado por un aut. finito det $\Leftrightarrow$ aceptado por aut. fin. No-det con transiciones nulas.

Clausura: Sea $M = (Q, A, \delta, q_0, F)$

$$Cl(q) = \{p : \exists p_1, \dots, p_n \quad p_1 = q, \; p_n = p \quad, p_i \in \delta(p_{i-1}, \varepsilon) \; i = 2, \dots, n\}$$

la clausura de un conjunto de estados es:

$$Cl(P) = \bigcup_{q \in P} Cl(q)$$

- El aut. fin determinista generado por un aut. fin. No det con transiciones nulas genera el mismo lenguaje.

$r^* r = r^+$

Props de las exp. regulares

- $r_1 + r_2 = r_2 + r_1$,
- $r_1 + (r_2 + r_3) = (r_1 + r_2) + r_3$, $r_1(r_2 r_3) = (r_1 r_2) r_3$
- $r\varepsilon = r$, $r\emptyset = \emptyset$
- $r + \emptyset = r$, $\varepsilon^* = \varepsilon$
- $r_1(r_2 + r_3) = r_1 r_2 + r_1 r_3$, $(r_1 + r_2) r_3 = r_1 r_3 + r_2 r_3$
- $r^+ + \varepsilon = r^*$, $r^* + \varepsilon = r^*$, $(r + \varepsilon)^* = r^*$, $(r + \varepsilon)^+ = r^*$
- $(r_1^* + r_2^*)^* = (r_1 + r_2)^*$

- El lenguaje es aceptado por un AFD $\Leftrightarrow$ Puede representarse por una exp. regular.

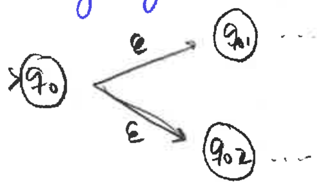
lenguage regular $\Leftrightarrow$ AFND $\Leftrightarrow$ AFD $\Leftrightarrow$ Expresión regular

Para la dem de la anterior proposición vamos a ver:

•) Dada una exp regular podemos obtener el AF que acepta el lenguage asociado.

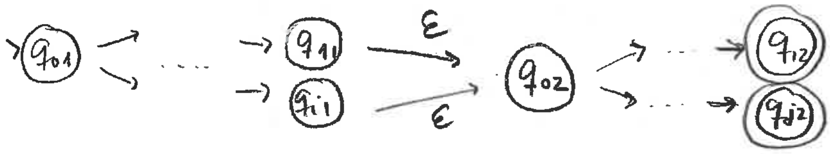
UNION

$(r_1 + r_2)$



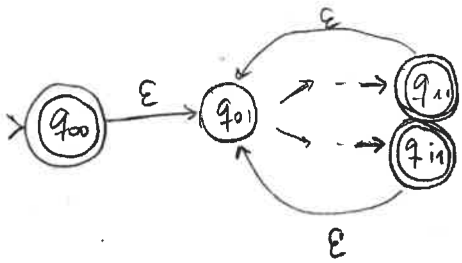
CONCATENACIÓN

$(r_1 r_2)$



CLAUSURA

r_1^*



•) Dado un AF podemos obtener la exp. regular que acepta el lenguage asociado.

Proceso de la hoja en sucio.

•) Existe un mét. alternativo que se encarga de eliminar estados
la idea es construir un AF para cada estado final que solo contenga el inicial y ese final, obtener de cada uno de esos autómatas las exp. regulares asociadas, y sumarlas todas.

$$r_{ij} \leftarrow r_{ij} + r_{ik} r_{kk}^* r_{kj}$$

Ejemplo en la hoja en sucio. (Esto NO solo vale con AFND)

Otra forma de este método es añadir un nuevo estado final conectado con todos los anteriores por transiciones nulas.

Gramáticas Regulares o tipo 3

- lineales por la derecha.

$$A \to uB \mid u$$

- Lineales por la izquierda

$$A \to Bu \mid u$$

•) Dada una gramática podemos obtener un AF de la siguiente forma (solo gram. lin. der o ~~Izq~~)
Ejemplo en la hoja en sucio.

•) Además podemos obtener una gramática lineal a partir de un autómata:
Ejemplos en la hoja a sucio

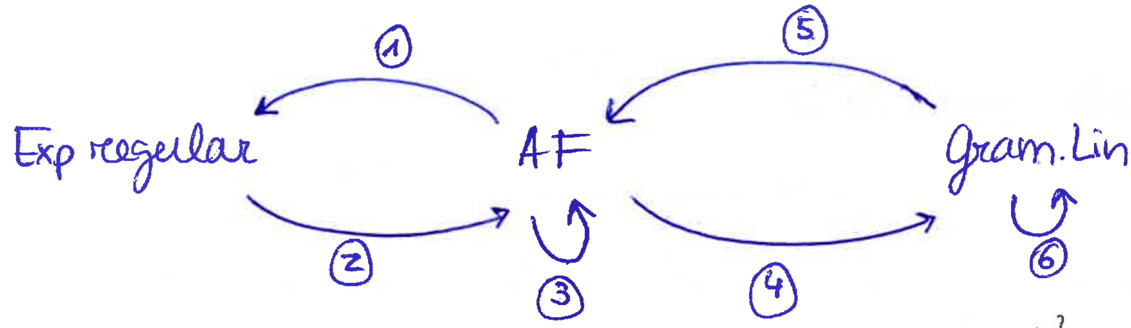
Finalmente, debemos mencionar la equivalencia que existe entre las gram. lin a la derecha y a la izquierda, esto es, dada una, existe otra que genera el mismo lenguage.

IMP No confundir complementario de un AF con el inverso de un AF.

El complementario: Finales $\rightleftarrows$ No finales
Debe ser **AFD.**

El inverso : $\begin{cases} \text{Cambiamos el sentido de las flechas.} \\ \text{Finales} \rightleftarrows \text{Iniciales} \end{cases}$

Si hay más de un estado final, creamos otro que agrupe a todos.



① Hay tres procesos:

- Megalargo
- Eliminar estados — que estados se eliminan?
- Ley Arden

$$\left.\begin{aligned} r_0 &= r_0 1 + r_1 1 + \varepsilon \\ r_1 &= r_0 0 \\ r_2 &= r_1 0 + r_2 0 \end{aligned}\right\}$$

Los que entran

Ley: $X = X r_1 + r_2 \Rightarrow$

$X = r_2 r_1^*$

Solución: suma de los finales

$$\left.\begin{aligned} r_0 &= 0 r_1 + 1 r_0 + \varepsilon \\ r_1 &= 1 r_0 + 0 r_2 \\ r_2 &= 0 r_2 \end{aligned}\right\}$$

Los que salen

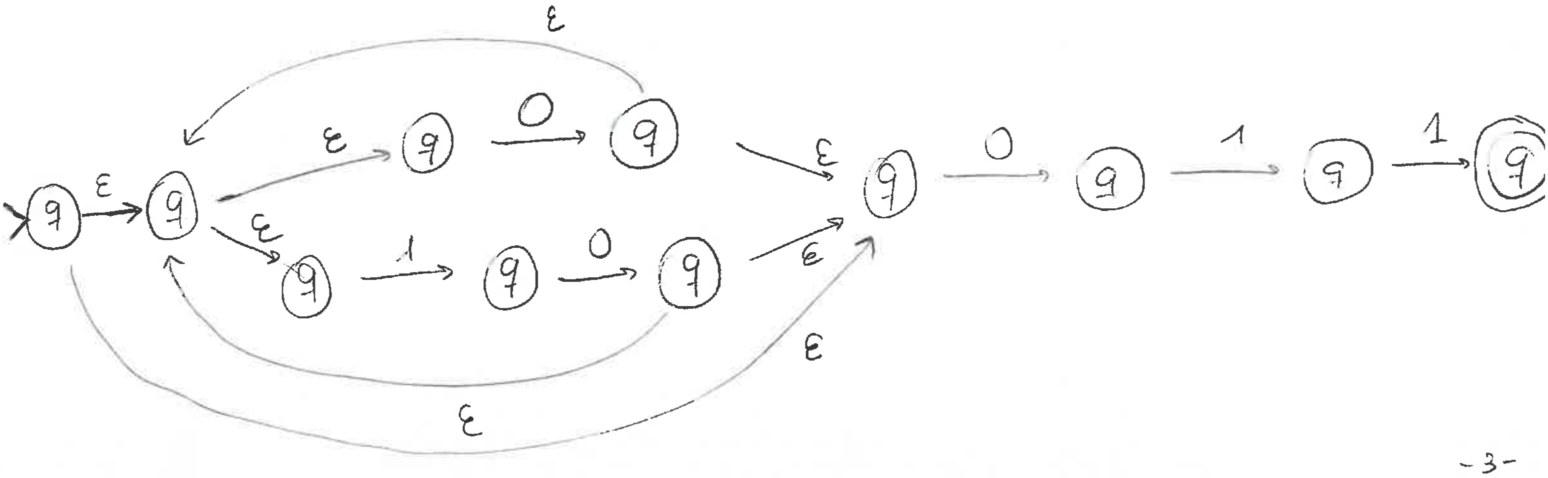
ley: $X = r_1 X + r_2 \Rightarrow$

$X = r_1^* r_2$

Solución. estado inicial

② Hay dos procesos: $(0+10)^* 011$

- Intentar sacarlo a ojo.
- Sustituir cada simbolo por el autómata correspondiente

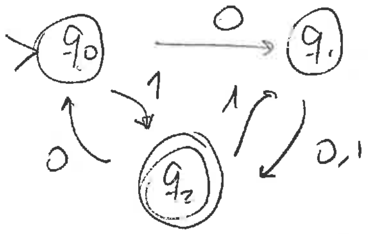


③ AFND ⟶ AFD

Partir de la clausura del estado inicial del AFND y ver todas las posibilidades.

④ AF ⟶ gram. Lin Der.
gram. Lin Izq.

AF ⟶ GLD



$q_0 \to 0q_1 \mid 1q_2$

$q_1 \to 1q_2 \mid 0q_2$

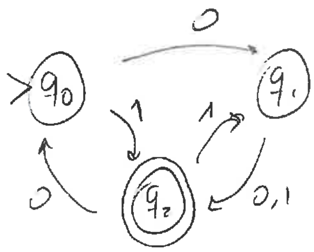
$q_2 \to 0q_0 \mid 1q_1 \mid \varepsilon$

¿Si tomamos los que entran obtenemos la GLI?

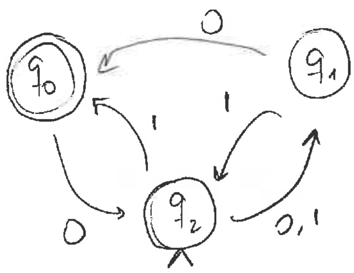
Tener en cuenta el camino de inicial a final

AF ⟶ GLI

- Invertir AF
- obtener GLD
- Invertir GLD



Invertimos el AF



Obtenemos la GLD

$q_0 \to 0q_2 \mid \varepsilon$

$q_1 \to 1q_2 \mid 0q_0$

$q_2 \to 1q_0 \mid 0q_1 \mid 1q_1$

Invertimos la GLD

$q_0 \to q_2 0 \mid \varepsilon$

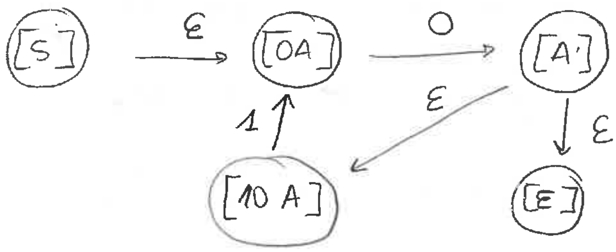
$q_1 \to q_0 0 \mid q_2 1$

$q_2 \to q_1 0 \mid q_0 1 \mid q_1 1$

5 GLD $\longrightarrow$ AF

$S \to 0A$

$A \to 10A \mid \varepsilon$

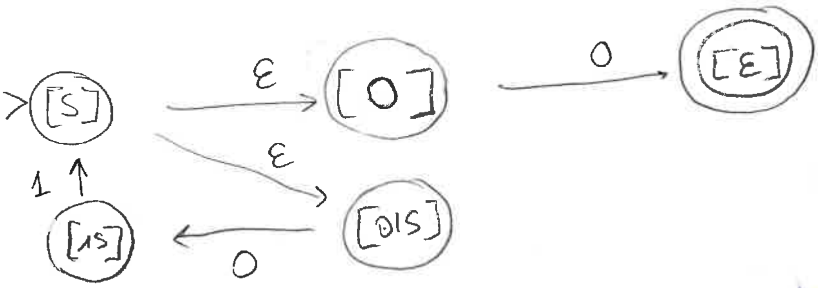


GLI $\longrightarrow$ AF

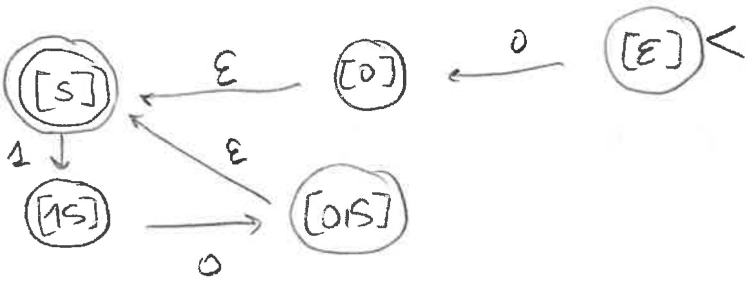
- Invertir GLI
- Construir el AF
- Invertir el AF

GLD(L')
↓
AF(L') → AF(L)

$S \to S10 \mid 0 \implies S \to 01S \mid 0$



Invertimos



⑥ GLD ⟶ GLI.

- obtenemos el AF
- Invertimos el AF
- obtenemos la GLD
- Invertimos la GLD.

$$\begin{tikzcd}
GLD \arrow[d] & & GLI(L) \\
AF \arrow[r, "\text{Invertimos}"] & AF(L^{-1}) \arrow[r, "GLD"] & GLD(L^{-1}) \arrow[u, "\text{Inv.}"']
\end{tikzcd}$$

GLI ⟶ GLD

- Invertimos la GLI
- obtenemos el AF
- Invertimos el AF
- obtenemos la GLD

$$\begin{tikzcd}
GLI(L) \arrow[d] & & GLD(L) \\
GLD(L^{-1}) \arrow[r] & AF(L^{-1}) \arrow[r] & AF(L) \arrow[u]
\end{tikzcd}$$

TEMA 3

$$z = \overbrace{\underbrace{(\qquad\quad}_{u}\underbrace{\quad)}_{v}}^{n}\underbrace{\quad)}_{w}$$

Lema de Bombeo

sea L conjunto regular $\Rightarrow \exists n \in \mathbb{N}$ tq $\forall z \in L$, si $|z| \geq n$, entonces $z = uvw$ tq

- ·) $|uv| \leq n \Rightarrow |w| \geq 1$
- ·) $|v| \geq 1$
- ·) $uv^i w \in L$

Además n es el nº estados de cualquier autómata que acepte el lenguage. (Esto se usa para probar que un lenguage NO es regular, ya que es cond. necesaria para que sea regular)

si L No es regular $\Leftarrow \forall n \in \mathbb{N}\ \exists z \in L, |z| \geq n$ tq dado $z = uvw$ si se verifica

- ·) $|uv| \leq n$
- ·) $|v| \geq 1$

Entonces $\exists i \in \mathbb{N} : uv^i w \notin L$

Ej: $L = \{0^j 1^j \; j \geq 0\}$

$z = 0^n 1^n$

$u = 0^K, v = 0^I, w = 0^{n-K-I} 1^n, I \geq$

$i = 2, \; uv^2w = 0^K 0^{2I} 0^{n-K-I} 1^n$
$= 0^{n+I} 1^n \notin L$

Hay más ejemplos en las diapos

Contraejemplos ~~(Lema bombeo)~~ (Lema bombeo)

$L = \{a^l b^j c^K : (l=0) \vee (j=K)\}$ No es regular pero satisface la condición esto es, satisface el lema de bombeo.

- La unión ($L_1 \cup L_2$), concatenación ($L_1 L_2$) y clausura (L_1^*) de dos lenguages regulares es regular.
- Si L es un leng. regular $\Rightarrow \bar{L} = A^* \setminus L$ (complementario) es regular.

Esta propiedad No se verifica en AFND.

- La intersección de lenguages regulares es regular. $L_1 \cap L_2$

Así, unión, concatenación, clausura, intersección y complementario es regular

•) Proceso de unión e intersección de AFD.
Ejemplo en hoja sucio. (Autómata Producto)

• Sean A, B alfabetos, $f: A^* \to B^*$ homomorfismo entre alfabetos, entonces, si $L \subset A^*$ regular,

$$f(L) = \{u \in B^* : \exists v \in A^* : f(v) = u\}$$ es regular

La dem se basa en conseguir la exp regular de L, y para obtener la de $f(L)$ sustituir cada símbolo v por su $f(v)$.

• Si A, B alfabetos $f: A^* \to B^*$ homomorfismo, entonces si $L \subset B^*$ regular, $f^{-1}(L) = \{u \in A^* : f(u) \in L\}$ también lo es

Podemos usar esto para ver que un lenguage No es regular.

$A = B = \{0,1\}$, $f: A^* \to B^*$, $f(0) = 00$, $f(1) = 11$ entonces $L = \{0^{2k} 1^{2k} ; k \geq 0\}$ no es regular, porque si lo fuese $f^{-1}(L) = \{0^k 1^k \; k \geq 0\}$ lo sería, y hemos visto que No lo es.

Cociente

Si R regular, L un lenguage.

$$R/L = \{u \in A^* : \exists v \in L \text{ tq } uv \in R\}$$

es un conjunto regular.

Si $M(Q, A, \delta, q_0, F)$ es un AFD que acepta a R, entonces R/L es aceptado por $M'(Q, A, \delta, q_0, F')$

$$F' = \{q \in Q ; \exists y \in L \text{ tq } \delta^*(q, y) \in F\}$$

• Existe un algoritmo para determinar si el lenguage aceptado por un AF es vacío.
Basta eliminar estados inaccesibles y comprobar si quedan estados finales.

Lenguage finito o infinito?
• Eliminamos los estados inaccesibles, de error y desde los que No podemos llegar a estados finales ¿Quedan ciclos?

- Existe un algoritmo para comprobar si dos AF aceptan el mismo lenguaje.

se construye $(L(M_1) \setminus L(M_2)) \cup (L(M_2) \setminus L(M_1)) = (L(M_1) \cap \overline{L(M_2)}) \cup (L(M_2) \cap \overline{L(M_1)})$

y se comprueba si es vacío

Un **AFD** se dice <u>minimal</u> si $\nexists$ otro con menos estados y que acepte el mismo lenguaje

Dados dos estados de un AFD, p y q son <u>indistinguibles</u>

$\iff \forall u \in A^*, (\delta^*(p,u) \in F \iff \delta^*(q,u) \in F)$

Para toda transformación uno es final $\iff$ el otro lo es

En caso contrario son <u>distinguibles</u>.

- Ser indistinguible es una relación equivalencia.
- Un estado final y otro no final son siempre distinguibles entre ellos, porque la palabra vacía les lleva a ellos mismos
- Si p, q son indistinguibles, $\forall a \in A$, $\delta(p,a)$ y $\delta(q,a)$ son indistinguibles y $\forall u \in A^*$, $\delta^*(p,u)$ y $\delta^*(q,u)$ son también indistinguibles.

Minimalización autómatas.

Si $M = (Q, A, \delta, q_0, F)$ AFD q_1, q_2 una pareja de estados indistinguibles entonces $M' = (Q', A, \delta', q_0', F')$ AFD acepta el mismo lenguaje.

- $Q' = Q \setminus \{q_1\}$
- $\delta'(p,a) = \begin{cases} \delta(p,a) & \text{si } \delta(p,a) \neq q_1 \\ q_2 & \text{si } \delta(p,a) = q_1 \end{cases}$
- $q_0' = \begin{cases} q_0 & \text{si } q_0 \neq q_1 \\ q_2 & \text{si } q_0 = q_1 \end{cases}$
- $F' = F \setminus \{q_1\}$

Dem en pg 33 diapos

- Si $M=(Q,A,\delta,q_0,F)$ y $M'=(Q',A,\delta',q_0',F')$ acepta el mismo lenguaje y $u,v\in A^*$ tq $\delta^*(q_0,u)=\delta^*(q_0,v)$ $\Rightarrow \delta'^*(q_0',u)$, $\delta'^*(q_0',v)$ son indistinguibles.

  M, M' AFD, u,v palabras que podemos formar, pues si del primer AFD podemos obtener u y v, en el segundo son indistinguibles

- Un autómata No es minimal $\Leftrightarrow$ tiene estados indistinguibles.
- Un autómata es minimal $\Leftrightarrow$ No tiene estados indistinguibles
  (sin estados inaccesibles) (todos distinguibles)

- Si M y M' son dos AFD minimales que aceptan el mismo lenguaje $\Rightarrow$ son isomorfos, esto es,

  $\exists f: Q \to Q'$ tq

  $f(q_0)=q_0'$

  $f(\delta(q,a))=\delta'(f(q),a) \quad \forall a\in A,\ q\in Q$

  $f(F)=F'$

- Dos estados son <u>distinguibles de nivel n</u> $\Leftrightarrow$ $\exists u\in A^*$, $|u|\leq n$ tq $\{\delta^*(p,u),\delta^*(q,u)\}$ hay un estado final y otro no final.
- distinguible $\Leftrightarrow$ distinguible de nivel n para algún n.

  distinguibles a nivel 0 es que uno es final y otro no final.

  Las parejas distinguibles a nivel n+1 son las que son distinguibles a nivel n + aquellas tales que existe $a\in A$ tq $\{\delta(q,a),\delta(p,a)\}$ distinguible a nivel n

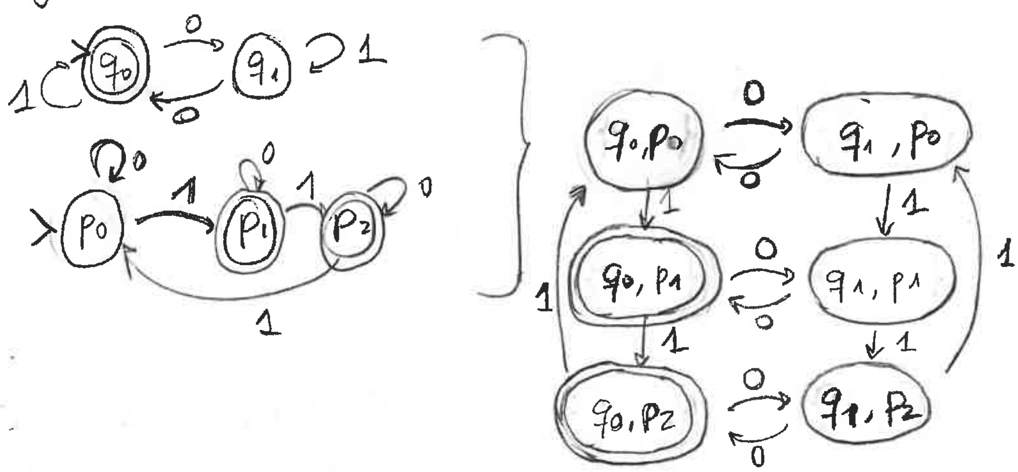
•) Procedimiento de cálculo de parejas de estados indistinguibles.

Hoja en sucio.

•) Proceso de construcción del AFD minimal.

Hoja en sucio.

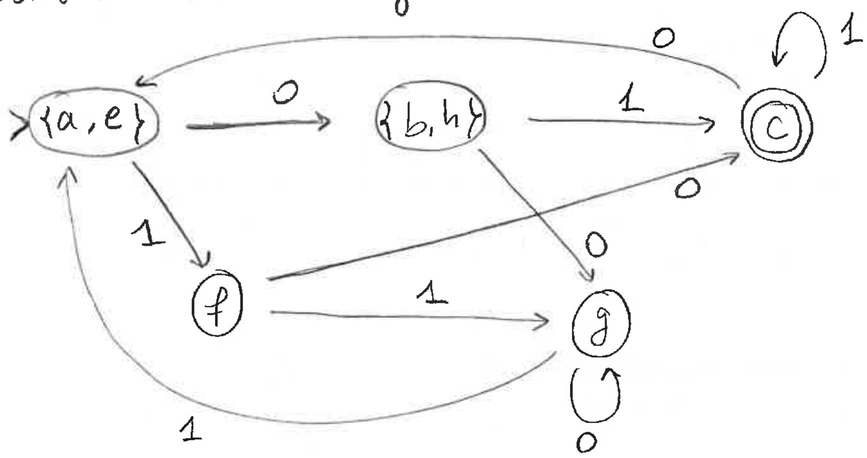
→ Ej de intersección de AFD



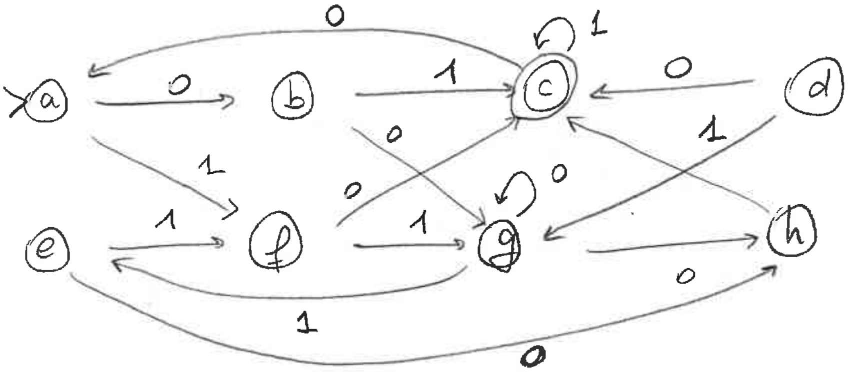
Para ser terminal debe ser términal en ambos AF.
Además los autómatas de la intersección deben ser AFD.
Importante! No son conjuntos, son pares de nodos!

→ Ej de unión de AFD
Funciona de igual modo que la intersección, solo que para que un par de nodos sean terminales basta con que uno de los dos sea terminal.

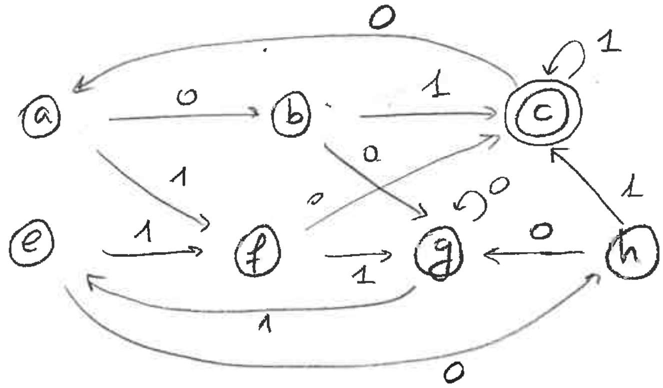
Una vez sabemos como es el AFD minimal, para construirlo debemos hacer lo siguiente:



Cálculo parejas de estados indistinguibles



• Eliminamos inaccesibles



b	X					
c	X	X				
e		X	X			
f	X	X	X	X		
g	X	X	X	X	X	
h	X		X	X	X	X
	a	b	c	e	f	g

	0	1
h	g	c
a	b	f

	0	1
h	g	c
b	g	c

	0	1
h	g	c
e	h	f

	0	1
g	g	e
b	g	c

	0	1
h	g	c
f	c	g

	0	1
h	g	c
g	g	e

	0	1
g	g	e
a	b	f

	0	1
g	g	e
e	h	f

	0	1
g	g	e
f	c	g

	0	1
f	c	g
a	b	f

	0	1
e	h	f
a	b	f

	0	1
e	h	f
b	g	c

	0	1
b	g	c
a	b	f

	0	1
f	c	g
b	g	c

	0	1
f	c	g
e	h	f

Resultado $e \equiv a$ y $b \equiv h$

En un homomorfismo podemos hacer $f(algo) = \varepsilon$? Mas de 1 vez?

TEMA 4

Una gramática es independiente del contexto <=> todas las producciones son de la forma

$$A \to \alpha \quad , \quad A \in V,\ \alpha \in (V \cup T)^*$$

Generan los denominados lenguajes independientes del contexto.

Vamos a usar un árbol de derivación, que es un conjunto de nodos con símbolos del alfabeto, donde se efectua una ramificación por cada producción que se aplique.

Un árbol puede proceder de dos cadenas distintas, esto es, derivación por la izq, consiste en ir leyendo las variables de la izq de la palabra. Respectivamente la derivación por la derecha.

Una gramática es ambigua <=> $\exists$ una palabra con dos arboles derivación distintos.

Un lenguage es inherentemente ambiguo <=> toda gramática es ambigua

Para probar que un leng es inh. ambiguo No basta con demostrar que 1 gramática que lo genera es ambigua, sino que hay que hacerlo con todas las gramáticas que lo generan

Un símbolo $X \in (V \cup T)$ es útil <=> $\exists$ cadena de derivaciones tq $S \overset{*}{\Rightarrow} \alpha X \beta \overset{*}{\Rightarrow} w \in T^*$

Una producción es útil <=> todos sus símbolos son útiles <=> se usa para producir alguna palabra.

•) Algoritmo para eliminar símbolos y producciones inútiles

1) Eliminar las variables desde las que no se puede llegar a una palabra de T^* y sus producciones

2) Eliminar los símbolos No alcanzables desde el inicial y sus producciones.

IMPORTA EL ORDEN. Explicado en la hoja a sucio

Algoritmo bueno: Explicado en la hoja en sucio.

• Si el lenguaje generado es vacío, la variable S resulta como inútil. Eliminamos todo menos el símbolo S.

• Es posible construir un algoritmo que dada una gramática G, construye G_n sin producciones nulas tal que $L(G_n) = L(G) \setminus \{\varepsilon\}$

las variables tales que en algún momento van a la palabra vacía se les llama <u>anulables</u>

•) Algoritmo para eliminar las producciones nulas.
(Si queremos añadir ε solo debemos hacer $S_v \to S \mid \varepsilon$)
Almacenamos en un conjunto las variables que llevan al vacío y sustituye en las variables del conjunto todas las posibilidades. (Si S es anulable, genera la ε)

↳ Me apunto las variables anulables y hago todas las combinaciones.

•) Algoritmo para eliminar producciones unitarias.
Creamos un conjunto con $\{(A,S)\}$ indica que existe la producción unitaria $A \to S$. Una vez creado, A debe poder ir a todo lo que vaya S.

Si una palabra no es generada, para comprobarlo debemos llegar a la profundidad $2n-1$, donde $|u| = n$. (Sin producciones nulas ni unitarias).

Forma Normal Chomsky $(2n-1)$

Forma $A \to BC$, $A \to a$ $A, B, C \in V$, $a \in T$.

·) Algoritmo para obtener Chomsky a partir de una gramática sin producciones nulas ni unitarias.

- Eliminar ~~los~~ terminales unitarios creando una variable para cada uno.
- Producciones con más de dos símbolos separarlas.

$A \to BEFG \longrightarrow A \to BD_1 \quad D_1 \to EFG$

y eliminamos $A \to BEFG$.

Forma Normal Greibach (n)

Forma $A \to a\alpha$, $a \in T$, $\alpha \in V^*$

$A \to \alpha$ $\alpha \in V^*$, $|\alpha| \geq 2$

Esto se aplica después de Chomsky.

No tiene que estar en Chomsky!

Procedimiento...

Para saber si la palabra no es generada, hay que bajar hasta n para asegurarnos, donde $|u|=n$

Algoritmo para eliminar símbolos y producciones inútiles

$S \to AB \mid a$

$A \to a$

1) Eliminamos B. y su producción
2) Eliminamos A y su producción $\Rightarrow S \to a$

Algoritmo bueno:

1) Ir guardando los símbolos que llevan a palabras o a símbolos que ya están en V_T, y guardarlos en V_T. Eliminar las producciones que no estén en V_T.
2) crearte 3 conj. V_S (historial de lo que examinas), J(variables a estudiar), T_S (símbolos terminales usados) Eliminar todas las producciones donde estén los símbolos que no he examinado.

TEMA 5

Un <u>aut. con pila No determinista</u> es una septupla

$(Q, A, B, \delta, q_0, Z_0, F)$

- ·) Q conjunto finito de estados
- ·) A es un alfabeto de entrada
- ·) B es un alfabeto para la pila
- ·) δ es la f. transición

$$\delta : Q \times (A \cup \{\varepsilon\}) \times B \longrightarrow \mathcal{P}(Q \times B^*)$$

- ·) q_0 estado inicial
- ·) Z_0 es el símbolo inicial de la pila.
- ·) F conjunto de estados finales.

Se llama <u>descripción instantanea</u> o <u>configuración.</u> de un aut. con pila a la tripleta

$$(\underbrace{q}_{\text{estado autómata}}, \underbrace{\mu}_{\text{cadena entrada}}, \underbrace{\alpha}_{\text{contenido pila}}) \in Q \times A^* \times B^*$$

$(q, au, Z\alpha) \vdash (p, u, \beta\alpha) \iff (p, \beta) \in \delta(q, a, Z)$

puede llegar mediante un <u>paso de calculo</u>.

$C_1 \overset{*}{\vdash} C_2 \iff \exists T_1, \ldots, T_n$ tq $C_1 = T_1 \vdash T_2 \vdash \cdots \vdash T_n = C_2$

La configuración inicial sería (q_0, u, Z_0), $u \in A$.

Criterios de APND: estados finales / pila vacía

- estados finales: $L(M) = \{w \in A^* : (q_0, w, Z_0) \overset{*}{\vdash} (p, \varepsilon, \gamma)\}$, $p \in F$, $\gamma \in B^*$
- pila vacía: $N(M) = \{w \in A^* : (q_0, w, Z_0) \overset{*}{\vdash} (p, \varepsilon, \varepsilon), p \in Q\}$

L es el leng. aceptado por un aut. con pila, M, por el criterio de pila vacia $\iff$ aceptado por el criterio de estados finales.

•) Pila Vacía $\longrightarrow$ Estados finales

$$\delta(q_0^n, \varepsilon, Z_0^n) = (q_0, Z_0 Z_0^n)$$

... Pila Vacia ...

$$\delta(q, \varepsilon, Z_0^n) = (q_f, Z_0^n)$$

Transformación determinista

•) Estados finales $\longrightarrow$ Pila Vacia.

$$\delta(q_0^n, \varepsilon, Z_0^n) = (q_0, Z_0 Z^n)$$

Transformación es No determinista

... Estados finales...

$$\delta(q, \varepsilon, H) = (q_s, H)$$

$$\delta(q_s, \varepsilon, H) = (q_s, \varepsilon)$$

Leng. Ind. Cont. es determinista $\iff$ aceptado por un ACP determinista por estados finale

ACP determinista $\iff$

•) $\delta(q, a, X)$ tiene máximo 1 elemento.

•) $\delta(q, a, X) \neq \emptyset \implies \delta(q, \varepsilon, X) = \emptyset$ para algún $a \in A$

$\iff \forall C_1 \; \exists$ como maximo C_2 tal que $C_1 \vdash C_2$

• L aceptado por ACPD (crit pila vacia) $\overset{+\text{ prefijo}}{\Longleftarrow}$ / $\Longrightarrow$ / $\not\Longleftarrow$ aceptado por ACPD (crit est. finales)

• L tiene la prop. prefijo $\iff \forall x \in L$, ningún prefijo de x (distinto está en L.

- Si tenemos un lenguage L determinista y No cumple la prop prefijo, podemos hacer que acepte esta propiedad $\Rightarrow$ acepta un autómata determinista por el crit. pila vacía. Solo hay que añadir un nuevo símbolo al final de todas las palabras: $L\{\$\} = \{u\$: u \in L\}$

- Dada una gramática libre contexto, $\exists$ ACP M que acepta el mismo lenguaje, y recíprocamente, dado un autómata M $\exists$ gramática libre contexto que genera el mismo lenguaje. (No tiene por qué ser determinista)

Gram $\rightarrow$ Aut CP

$S \rightarrow aSb \mid cSb \mid a$

$\delta(q, \varepsilon, S) = \{(q, aSb), (q, cSb), (q, a)\}$

$\delta(q, b, b) = \{(q, \varepsilon)\}$

$\delta(q, c, c) = \{(q, \varepsilon)\}$

$\delta(q, a, a) = \{(q, \varepsilon)\}$

Aut. CP $\longrightarrow$ gramática

(No tiene que ser determ)

Sea el siguiente autómata

$\delta(q_0, 0, Z_0) = \{(q_0, XZ_0)\}$

$\delta(q_0, 0, X) = \{(q_0, XX)\}$

$\delta(q_0, 1, X) = \{(q_1, \varepsilon)\}$

$\delta(q_1, 1, X) = \{(q_1, \varepsilon)\}$

$\delta(q_1, \varepsilon, X) = \{(q_1, \varepsilon)\}$

$\delta(q_1, \varepsilon, Z_0) = \{(q_1, \varepsilon)\}$

$S \longrightarrow [q_0, Z_0, q_0] \mid [q_0, Z_0, q_1]$ estado inicial, pila inicial, estados

$[q_0, Z_0, q_0] \longrightarrow 0\,[q_0, X, q_0]\,[q_0, Z_0, q_0]$

$[q_0, Z_0, q_0] \longrightarrow 0\,[q_0, X, q_1]\,[q_1, Z_0, q_0]$

$[q_0, Z_0, q_1] \longrightarrow 0\,[q_0, X, q_0]\,[q_0, Z_0, q_1]$

$[q_0, Z_0, q_1] \longrightarrow 0\,[q_0, X, q_1]\,[q_1, Z_0, q_1]$

$[q_1, X, q_1] \longrightarrow 1$

$[q_0, X, q_0] \rightarrow 0\,[q_0, X, q_0]\,[q_0, X, q_0]$

$[q_0, X, q_0] \rightarrow 0\,[q_0, X, q_1]\,[q_1, X, q_0]$

$[q_0, X, q_1] \rightarrow 0\,[q_0, X, q_0]\,[q_0, X, q_1]$

$[q_0, X, q_1] \rightarrow 0\,[q_0, X, q_1]\,[q_1, X, q_1]$

$[q_1, X, q_1] \rightarrow \varepsilon$ $\qquad$ $[q_0, X, q_1] \rightarrow 1$

$[q_1, Z_0, q_1] \rightarrow \varepsilon$

Ahora hay que eliminar las producciones inútiles

TEMA 6

Lema Bombeo $\not\Leftarrow$ $L=\{a^ib^jc^kd^l \mid i=0 \vee j=k=l\}$

Sea L indep. contexto $\Rightarrow$ $\exists n$ cte que depende de L

tq si $z\in L$, $|z|\geq n \Rightarrow z = \overset{a^n\, b^L\, b^I\, b^{n-L-I}\, c^n}{uvwxy}$

1) $|vx|\geq 1$
2) $|vwx|\leq n$
3) $\forall i\geq 0$, $uv^iwx^iy \in L$

Sup que $\exists n$ $L=\{a^ib^ic^i : i\geq 1\}$

$z = a^n b^n c^n \in L$

$uv^2wx^2y = a^n b^{2L} b^I b^{2(n-L-I)} c^n$

$= a^n b^{2L+I+2n-2L-2I} c^n$

$= a^n b^{2n-I} c^n \notin L$

$\Rightarrow$ L No es indep

Se utiliza $\neg b \Rightarrow \neg a$

Los lenguages independientes del contexto son cerrados para la unión, concatenación y clausura.

¡Para la intersección NO son cerrados! ¡Tampoco para el complementario!

$L=\{a^ib^ic^i : i\geq 1\} = L_1=\{a^ib^ic^j : i\geq 1 \text{ y } j\geq 1\} \cap L_2=\{a^jb^ic^i : i\geq 1, j\geq 1\}$

No es ind.cont — Son Indep Cont

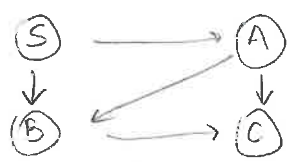
Si L es l. indep. contexto determinista $\Rightarrow$ su complementario es indep. contexto determinista.

Si L es l. indep contexto, R es regular $\Rightarrow$ $L\cap R$ es indep contexto.

- Algoritmo para ver si el lenguage generado por una gramática indep. contexto es infinito:
  - → Eliminamos símbolos y producciones inútiles
  - → Eliminamos producciones nulas.
  - → Construimos un grafo donde hay un arco de A a B $\Leftrightarrow$ $A\rightarrow \alpha B \beta$

lenguage infinito $\Leftrightarrow$ $\exists$ ciclos en el grafo.

Ej. $S\rightarrow AB$ $A\rightarrow BC\mid a$ $B\rightarrow CC\mid b$ $C\rightarrow a$



No tiene ciclos $\Rightarrow$ No infinito = finito

Algoritmo pertenencia < Cocke - Younger - Kasami (CYK)
Early

CYK es de complejidad n^3, con $n = |u|$
Es necesario que esté en forma normal Chomsky.
Ejemplo en hoja sucio

Earley exige que la gramática no tenga producciones nulas ni unitarias. (Basado en prog. dinámica).
Es de complejidad n^3, y n^2 para gramáticas No ambiguas.
Ejemplo en hoja sucio

Algoritmo CYK

Veamos si baaba pertenece a la siguiente gramática.

$S \to AB \mid BC$ $\quad B \to CC \mid b$

$A \to BA \mid a$ $\quad C \to AB \mid a$

b	a	a	b	a
B	A C	A C	B	A C
A S	B	S C	A S	
∅	B	B		
∅	S C A			
S A C				

$\Rightarrow$ baaba si es generada

Algoritmo Early

$S \to AB \mid BC$ $\quad B \to CC \mid b$

$A \to BA \mid a$ $\quad C \to AB \mid a$

Encontrar si baa pertenece.

- Inicialización $(0,0,A,\varepsilon,BA)$
- (j,d) Clausura: mirando la 1ª letra del último campo e inicializarla
- Avance: buscamos la letra que queremos, sumamos 1 al 2º campo y movemos el primer símbolo a la izq
- Terminación: buscar en el registro [i] en el último a la izq y el tercer campo, en el último a la izq y sumar en j y mover el 1er símbolo del último campo al penúltimo.

REGISTRO[0] = $(0,0,S,\varepsilon,AB),(0,0,S,\varepsilon,BC),(0,0,A,\varepsilon,BA)$
$(0,0,A,\varepsilon,a),(0,0,B,\varepsilon,CC),(0,0,B,\varepsilon,b),(0,0,C,\varepsilon,AB),(0,0,C,\varepsilon,a)$

REGISTRO[1] = $(0,1,B,b,\varepsilon)(0,1,S,B,C)(0,1,A,B,A)$
$(1,1,C,\varepsilon,AB),(1,1,C,\varepsilon,a),(1,1,A,\varepsilon,BA),(1,1,A,\varepsilon,a)$
$(1,1,B,\varepsilon,CC),(1,1,B,\varepsilon,b)$

REGISTRO[2] = $(1,2,C,a,\varepsilon),(1,2,A,a,\varepsilon),(0,2,S,BC,\varepsilon)$
$(1,2,B,C,C),(0,2,A,BA,\varepsilon)(1,2,C,A,B),(0,2,S,A,B)$
$(0,2,C,A,B),(2,2,C,\varepsilon,AB),(2,2,C,\varepsilon,a),(2,2,B,\varepsilon,CC)$
$(2,2,B,\varepsilon,b),(2,2,A,\varepsilon,BA),(2,2,A,\varepsilon,a)$

REGISTRO[3] = $(2,3,C,a,\varepsilon),(2,3,A,a,\varepsilon),(1,3,B,CC,\varepsilon)$
$(2,3,B,C,C)(2,3,C,A,B)(1,3,A,B,A)$

NO GENERADA

$(0,3,S,\alpha,\varepsilon) \Longrightarrow$