

RESUMEN LENGUAJES FORMALES

Un LENGUAJE Formal es un CONJUNTO DE PALABRAS

Una palabra a su vez es una concatenación de SÍMBOLOS (Finitos)

"W" (Símbolo MUY AMPLIO)

Ej: 0110, 0000, Pepe, A

PALABRA VACIA

Para definir la cantidad de símbolos que existen en un LF se utiliza un alfabeto.

"Σ"

Conjunto finito no vacío de Σ Ej: $\Sigma_1 = \{0, 1\}$, $\Sigma_2 = \{0, 1, 2\}$

Un lenguaje es Formal porque solo nos importa su sintaxis, nada más. Estas están sujetas a REGLAS GRAMATICALES PREESTABLECIDAS, por lo tanto no predicciones. Tienen que ser discretas / pudiendo ser abstractas.

- EXTENSION: Agrupando sus palabras

- COMPRENSION NATURAL: Por palabras separadas

- COMPRENSION SIMBOLICA: Con símbolos de manejo matemático.

~~Comprendiendo~~ Como los lenguajes formales son conjuntos, dentro de ellos existen subconjuntos llamados sublenguajes.

OPERACIONES ENTRE LENGUAJES:

|L1|: CARDINALIDAD (C. PALABRAS DENTRO DEL LENGUAJE)
define si es finito o infinito.

Σ^* : LENGUAJE UNIVERSAL, contiene todas las palabras que se pueden formar con el alfabeto más lo palabra vacía.

L1 ∪ L2: UNION, juntas todas las palabras de los 2. (sin repetir ni duplicado)

L1 ∩ L2: INTERSECCION, dejo solo las que se repiten en ambos

~L1 o L1: COMPLEMENTO, todas las palabras que no pertenecen a L1

$L_1 - L_2 = L_1 \cap \sim L_2$: DIFERENCIA, quedan las palabras de L_1 que no pertenecen a L_2 .

$L_1 \cdot L_2$: PRODUCTO, todas las w_1 y w_2 .

L^n : POTENCIA

L^{-1} + L^R : REFLEXION, se espejan las palabras.

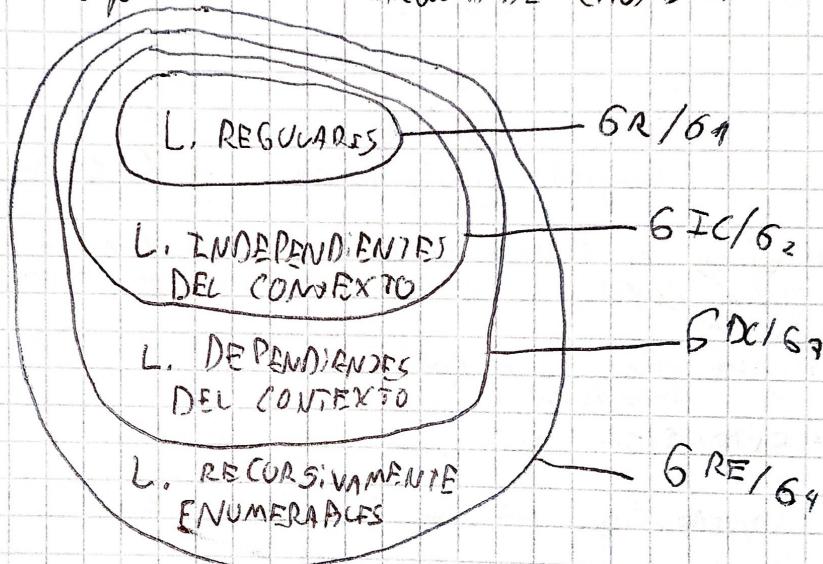
L^* : CIERRE ESTRELLA

PROPIEDADES L.F

• La CONCATENACION es ASOCIATIVA pero NO es COMUTATIVA

• La UNION es distributiva pero la INTERSECCION NO

Ser clasifican en 4 Niveles la JERARQUIA DE CHOMSKY



Empiezan con el primer tipo de L.F:

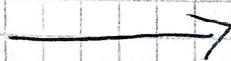
LENGUAJES REGULARES

¿Cuando es un L.R.? Si este es FINITO o género una!

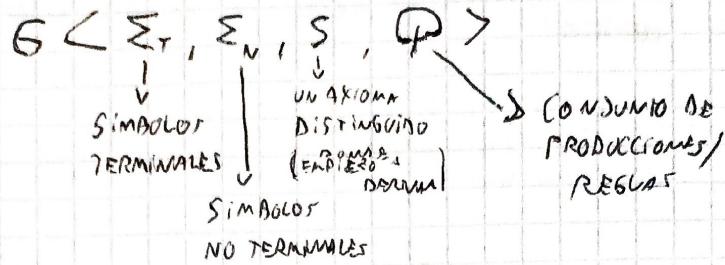
- GR: Gramatica Regular
- ER: Expresion Regular
- AF: Automata Finito

GRAMATICA REGULAR (GR)

San las reglas (llamadas PRODUCCIONES del tipo $\alpha \rightarrow \beta$) para formar palabras en un lenguaje.



Una gramática está formada de este modo; 4 elementos:



P : Para que una gramática sea regular, en el conjunto de produc. debe suceder lo siguiente:

- Si existe λ entonces esto debe siempre dervarse del axioma

$$(S \rightarrow \lambda)$$

- Y cualquier No terminal debe dervarse en un TERMINAL o en un NO TERMINAL ACOMPAÑADO DE UN TERMINAL.

$$(A \rightarrow v) \mid (A \rightarrow vB)$$

$$\{A; B\} \in \Sigma_N \quad ^\wedge \quad \{v\} \in \Sigma_T$$

Ej:
 $L = \{a^m b \mid m \geq 0\}$

$$L = \{b, ab, aab, aaab, \dots\}$$

GR: $S \rightarrow b \mid aS \Rightarrow$ Esto GR tiene 2 reglas $\begin{pmatrix} NT \uparrow \quad T \uparrow \\ S \rightarrow b \\ S \rightarrow aS \end{pmatrix}$

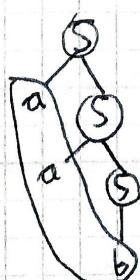
$$P = \{S \rightarrow b, S \rightarrow aS\}$$

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

$$\Sigma_N = \{S\}$$

ARBOL DE DERIVACION (CAMINO DE COMO SE DA FORMA)

Ej: aab



Las hojas forman la palabra.

Cabe aclarar que dentro de los GR existen unas subgramáticas:

- GRAMÁTICAS EQUIVALENTES: Cuando 2 GR distintas generan el mismo L.
- GRAMÁTICAS QUASI REGULARES: Generan un LR más simplificado para facilitar el cálculo de la gramática
- GRAMÁTICAS AMBIGUAS: ~~GRAMÁTICAS~~

↓
No ACEPTABLES: Cuando una palabra no puede derivar de más de una forma.

EXPRESIONES REGULARES

Metalingüaje que permite representar a un lenguaje regular con el objetivo de describir un patrón.

COMPONENTES:

$$\bullet \delta \in \Sigma$$

$$\bullet \lambda \quad \text{si representa como } \rightarrow \lambda$$

• OPERADORES: $\{^*, ^+, |, ()\}$ PERMITEN ALTERAR A LOS DIFERENTES

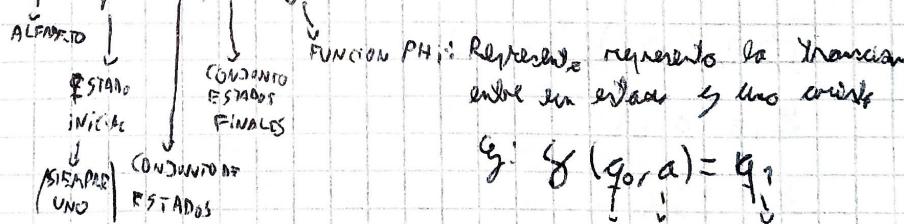
la ER que representa todos los palabras de un lenguaje se llaman ERU (ER. UNIVERSAL)
A su vez existe el concepto de ER equivalentes.

AUTOMATOS FINITOS

Máquinas que basadas en las reglas leen las palabras de un lenguaje regular.

COMPONENTES:

$$AF \subset \langle \Sigma, q_0, Q, F, \delta \rangle$$

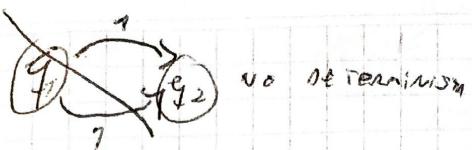


3 TIPOS DE REPRESENTACIÓN:

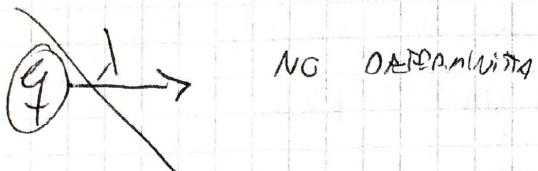
- GRAFICA $\rightarrow \circlearrowright$
- TEXTO (DEF. FORMA $\langle \Sigma, q_0, Q, F, \delta \rangle$ como una)
- MATRICIAL (TABLA DE TRANSICIONES)

EXISTEN 2 TIPOS AF:

- AFD (AUTOMATA FINITO DETERMINISTA): Para cada q en el que me encuentre cada símbolo tiene 1 sola salida SIN AMBIGÜEDAD.
- AFN (AUTOMATA FINITO NO DETERMINISTA)



no se termina



NO DETERMINISTICO

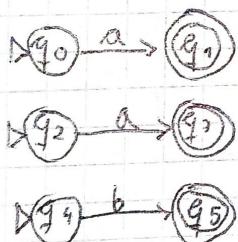
Luego estos AF pueden no:

- COMPLETOS: Para cada "q" deben salir tantas transiciones como símbolos en el alfabeto.
Para lograr esto se añade un estado falso,
 q_0 y arroj.
- INCOMPLETOS

También definen los AF equivalentes y el concepto de AFm (Automata Finito) minimo

ALGORITMOS DE TRANSFORMACIÓN

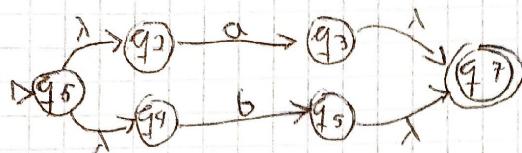
THOMPSON: ER → AFN

Ej: $a \cdot (a \mid b)^*$ 1^o Hay que desmembrar la E.R en sus componentes básicas (AF por cada símbolo)2^o Comienza a unirlos teniendo en cuenta los operadores. Cada operador tiene una manera de unir y sigui la siguiente tabla de precedencia.

PRECEDENCIA: (SE VE ALTERADA POR PARENTESIS)

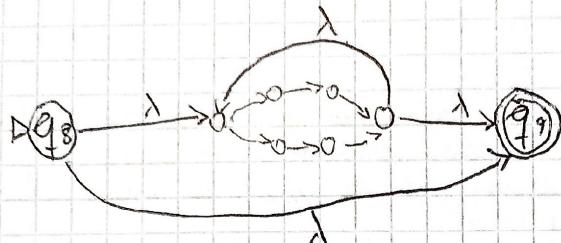
1^o *2^o ·3^o |

○ Para " l " (unir) 2 automatas



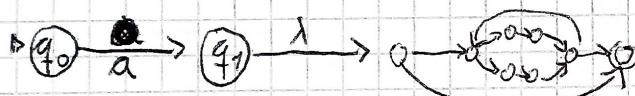
• Agregas 2 estados nuevos, uno inicial y otro final

○ Para "*" añadir cerradura de Kleen.



• Agregas 2 estados nuevos final e inicial + se agregan dos transiciones λ entre ellos para lo nraea y una entre el definial y el inicial para la iteracion

○ Y para ":" concatenar



• Si agrego una transicion λ entre ellos

ALG. CLAUSURA -1: AFN \rightarrow AFD

1^{ro} SACO el $C-1(q)$ de cada estado = $\{q\} \cup \{q\}$ todos los q que llega usando λ

2^{do} ARMO una tabla con el q inicial y las simbolos / entrada en cada cuadro el conjunto hacia del estado, si aparecen q nuevos repito hasta un q fixo
(no surgen q nuevos)

$$h(C-1(q_i), a) = \{q_{i+1}\}$$

$C-1$
del estado

q a los que se llega con ese simbolo en cada uno de los q del $C-1$

REPUCION C-1: AFN \rightarrow AFD

1^{ro} TABLA DE TRANSICIONES

2^{do} En los cuadrados se crea un nuevo estado llamado como una concatenacion de los estados a los que llega del de lo ~~que~~ hijo, y se agregan a la tabla

3^{ro} Los estados finales se heredan y si un estado se repite se da una vez

4^{to} Se eliminan los estados no accesibles desde q inicial.

ALG. (LEMA DE ARDEN) AFD → ER

- 1^o Eliminar errores
- 2^o Establecer sistema de ecuaciones
- 3^o Se identifica si hay una iteración y se reduce con el lema de orden.

$$\begin{cases} q = \alpha q + \beta \\ q = \alpha^* \beta \end{cases}$$

- 4^o Sustituir y resolver

ALG. DE CLASES: AFD → AFQmin

- 1^o Se grafica la tabla de transiciones completa
- 2^o Se parte la tabla en 2 conjuntos/clases: Una clase de q finales y Una clase de q no finales
- 3^o Se eliminan estados equivalentes (misma clase, misma transición/condicionamiento) y se obtiene una sola de ellas.
- 4^o Actualizo tabla y repito el 3^o hasta que no haya equivalencias
- 5^o Construyo tabla de transiciones por CLASES (En lugar de ~~estados~~ los q de los que llega y no los que salen) solo la clase (final/no final)
- 6^o Se repiten los pasos 2, 3 y 4 pero en la TTC, solo se crean clases nuevas dependiendo de los estados y se eliminan las equivalencias, se refresca y se repite

LENGUAJES INDEPENDIENTES DEL CONTEXTO



$$Ej: L_{IC} = \{a^m b^n / m \geq 0\}$$

AP
(AUTOMATA DE PILA)

$$G_{IC}(G_2) \subset \Sigma_T, \Sigma_N, S, Q$$

(GRAMATICA
INDEPENDIENTE
DEL CONTEXTO)

$$(S \rightarrow A) (A \rightarrow \text{algo})$$

A su vez una GIC puede ser:

GIC
EFICIENCIA
(2 GIC
BIEN
FORMADA)

DIFERIR COMPLICAR
2 CONDICIONES

Ser GIC LIMPIA

1. ~~REGLAS INNECESSARIAS~~
Ej: $A \rightarrow A$

$\alpha \in \Sigma_N$
(NO TERMINAL)

$S \Sigma, U \Sigma_N^*$
(CANTIDAD
DE DERIVACIONES)
Y NO TERMINALES

$$Ej: A \rightarrow a B B \rightarrow b \dots$$

2. ~~α INACCESIBLES DESDE S~~
Ej: $S \rightarrow A$
 $A \rightarrow a$
~~B~~

NO tener Reglas no Generativas
Ej: $A \rightarrow \bullet A$

NO tener Reglas de Redistribución

Ej: $A \rightarrow B$
 ~~$B \rightarrow D$~~

Y por ultimo, las GIC pueden estar expresadas en:

3 TIPOS de Formas Normales:

1. FNC (CHOMSKY): $(S \rightarrow \lambda) \mid (A \rightarrow BC) \mid (A \rightarrow a)$

"un terminal & 2 no terminales"

2. FNG (GREIBACH): $(S \rightarrow \lambda) \mid (A \rightarrow a X) \quad X \in (\Sigma)^*$

"un terminal o
un terminal y todos los
no terminales que quiera elegir"

3. FNB (BACKUS-NAUR): $\langle \Sigma_N \rangle$ "los NT entre <>"
(BNF)

• ; : = "lo fletita"

* Toda lo demás son terminales

AP (AUTOMATA DE PILA) = AF + PILA → Permite reconocer palabras de un LIC (también L1 y L2)

Si leo todos los SÍMBOLOS del lenguaje y la PILA queda VACÍA → la PALABRA es ACEPTADA

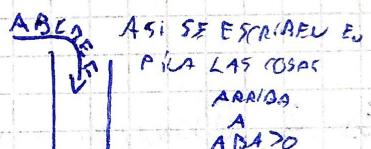
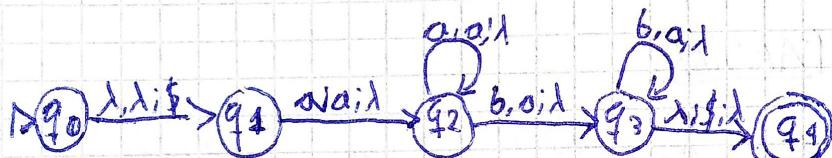
$\langle Q, q_0, F, \Sigma, \Gamma, \$, \delta \rangle$

↓
puede ser:
 \emptyset (ajunto)
vacío
Alfabeto
PILA
↓
Símbolos
Inicial
PILA

↓
: $(q_{\text{origen}}, \text{LEO}, \text{SAIGO}) \rightarrow (q_{\text{destino}}, \text{PAUSA})$

↓ o si se llegó
a estado final
(depende del tipo)
del automata P

Ej: $L = \{ a^n b^m \mid n \geq 0, m \geq 0 \}$ {RECORDAMOS: Es un LIC porque tenemos 2 símbolos que se están BOMBAEANDO en forma cíclica}



Ahoro... Existen 2 tipos de AP:

- APv (Automata de pila 100% VACÍA DE PILA)

- * Debe estar normalizado a formato normal de GREIBACH (FNG)

- * Existen tantas transiciones como producciones ~~existen~~ en un único estado inicial siguiendo el siguiente formato:

terminal, No terminal ; Cantidad 'x' de
del que deriva ; No terminal que
vuelva al terminal

Ej: $S \rightarrow r_1 \mid r_2 \mid g \text{ ASC} \mid g \text{ ASC}' \text{ SC}$

$A \rightarrow ($

$E \rightarrow)$

$C \rightarrow ", -$

En este
caso q
sería q
\$

$r_1, S; \lambda$
 $r_2, S; \lambda$
 $g, S; \text{ASC}$
 $g, S; \text{ASC}' \text{ SC}$
 $(, A; \lambda$
 $), C; \lambda$
 $C \rightarrow ", -$

↓
 q_0

* La pila
esta implícita
mente marcada
por el número
de

• APE (Automato de pilas para ~~ESTADO FINAL~~ ESTADO FINAL)

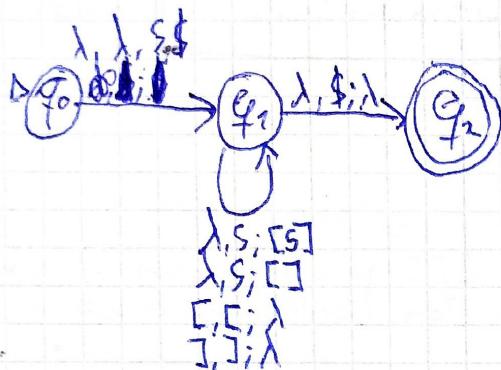
* Acepta cualquier pila.

* Cada transicion tiene una transicion donde no se lee nada, se saca del que dentro, y es la derecha de la flecha en la pila (tao).

* Y luego por cada simbolo terminal crea una transicion donde lee el terminal, saca el terminal y no saca nada.

* Tiene un estado final al cual llega vaciando la pila.

Ej: $L = S \rightarrow [S] \mid []$ * El ultimo ingresa al marcado de la pila.



PARSING

Es la descripción instantánea de los movimientos que van sucediendo al reconocer una palabra en un AP.

Tiene 3 maneras de representar:

• FORMAL

Palabras que nos quedan

$(q_0, [[]], \lambda) \vdash (q_1, [[]], \$) \dots$

dónde estoy
en la pila
y qué tengo
dónde quedo

Representa que
van al otro movimiento

entre los movimientos

y así hasta

al final

dónde quedo

en un "q"

final es q

por leer

en la pila

MATRICIAL

Genera una tabla con 3 columnas

PILA	ENTRADA	TRANSICION
λ	$[[]]$	$(q_0, \lambda, \lambda) =$ $(q_1, \$)$
$\$$	$[[]]$	Siguiente transicón...

GRAFICO

* Usar
pilas

la pila

y leer
mientras

$(q_0, \lambda, \lambda) = (q_1, \$)$

PARSERS

Son analizadores sintácticos, los hay de 2 TIPOS:

- AS descendente (TOP-DOWN) = LL BACKTRACKING,

Este es igual al APF

- AS ascendente (BOTTOM-UP) = LR BACKTRACKING,

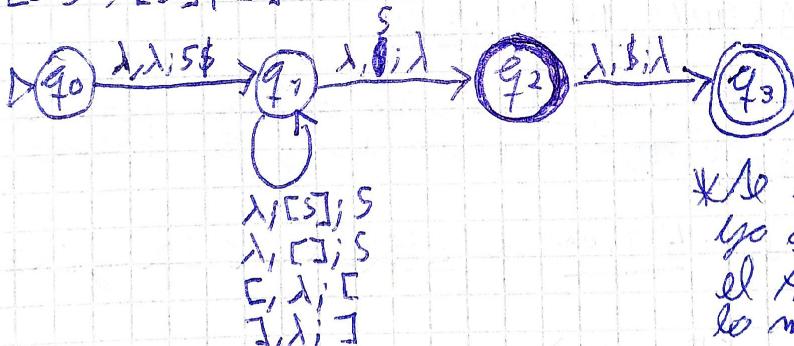
Este es diferente al APF, lo que utiliza una técnica llama SHIFT, REDUCE

* Aca por cada producción, nos saca la de la derecha de la flechita y pongo del que derro. (REDUCE)

* Y por cada terminal, nos saca mada y pongo el terminal (SHIFT)

(Cans APF pero al revés)

Ej: $L = S \rightarrow [S] \mid []$



* Se anota en estado mas lo que hay que sacar el ultimo antes que lo marca de la vila.

LEMA DEL BOMBAZO

Hay 2 tipos de LEMAS DE BOMBAZO:

- Para LR: Cuando un $|L| = \infty$ podemos utilizar el lema para afirmar que no es LR , lo que para que es lo se debe cumplir la siguiente

1º Se debe tomar un " w " $\rightarrow w \in L$
y un " n " $\rightarrow |w| \geq n$

2º Luego dividir " w " en 3 secciones: x, y, z cumpliendo: $|y| \geq 1$
 $|xyz| \leq n$

3º Entonces debe existir \bullet "K" que para todos $k \geq 0$ lo palabra:

$$\boxed{x y^k z} \bullet$$

Debe pertenecer al lenguaje.

• Para LIC:

1º Se debe tomar un " w " $\in L$
y un " n " $\rightarrow |w| \geq n$

2º Luego dividir lo palabra en 5 secciones: u, v, t tal que: $|v| > 0$

x

y

z

$$|vxy| \leq n$$

* Para cumplir las condiciones puedes rellinar secciones con " λ "

3º Entonces debe existir \bullet "K" que para todos $k \geq 0$ lo palabra:

$$\boxed{uv^k x y^k z}$$

Debe pertenecer al lenguaje

(Con LEMA DE BOMBEO siempre demuestras que NO, no puedes que v)

MAQUINA DE TURING

Una maquina de Turing es una abstraccion de una persona escribiendo en uno trozo de papel con un lapiz. Fue creada por Alan Turing en 1936.

Es capaz de reconocer los 4 TIPOS de LF. (LR, LIC, LDC y LRE)

$$MT \langle \Sigma, \Gamma, b, Q, q_0, F, \gamma \rangle$$

↓
ALF.
ENTRADA

↓
ALP.
CIMA

↓
SIMBOLO
BLANCO

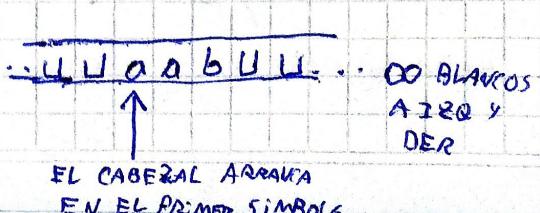
$$\Gamma = [\Sigma \cup \{b\} \cup \{\text{SÍMBOLOS}\}]$$



Movimiento Cabeza

$$\gamma (q_{actual}, b que lee) = (q destino, b que escribo, \{L, R, S\})$$

$$\text{Ej: } L = \{a^n b / n \geq 1\} \quad b = \text{U}$$



NOTA

MODIFICACIONES M.T.

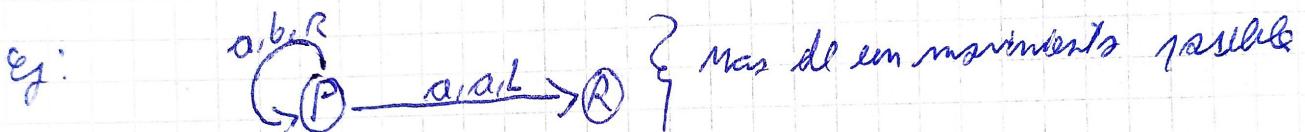
- MULTICINTA: Tiene mas de una cinta, cada cinta tiene su propio cabezal.

* Se debe especificar, b que se escribe, b que se lee y movimiento de CADA UNA DE LAS CINTAS en las transiciones.

Ej: $\delta(q_0, (a, b)) = (q_1, (b, a), \{R, L\})$

CINTA 1 CINTA 2

- NO DETERMINISTA: La función de transición pasa a ser RELACION



$$\Delta(P, a) = \{(P, b, R), (R, a, R)\}$$