

# C onstrucciones

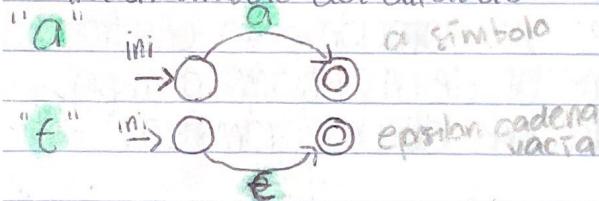
## de Thomsen

Dada una expresión regular obtenemos un AFN (automata finito no determinista) lo hace a través de plantillas.

\* una plantilla no acepta ningún cambio.

### Plantillas

Para un símbolo del alfabeto



dada una expresión regular y se quiere concatenar con otra expresión regular

$r_1 r_2$

1. Tomamos el automata de la primera expresión

2. Fusionamos el estado inicial  $r_2$  con el estado final  $r_1$



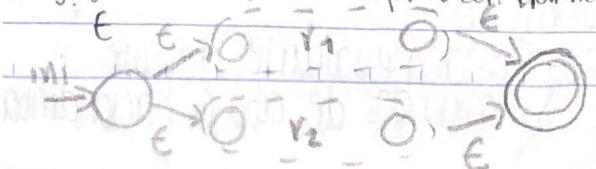
dada una expresión regular  $r_1$

$\cup r_2 \quad r_1 \cup r_2$

1. Se toma el automata de  $r_2$  abajo de  $r_1$

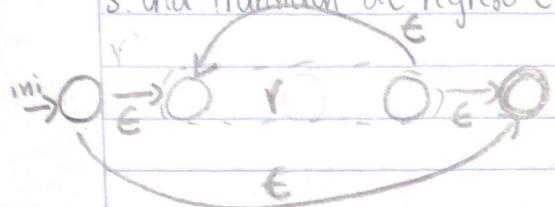
2. Se crean nuevos estados (inicial y final)

3. Se unen a los 2 caminos con transiciones



dada la expresión regular  $r^*$

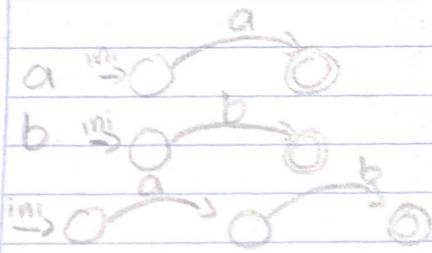
1. Se toma el automata de  $r$
2. Se crea un nuevo (inicial - final)
3. Unimos con transiciones  $\epsilon$  a los nuevos iniciales y final
4. Unimos inicial con final
5. Una transición de regreso  $\epsilon$



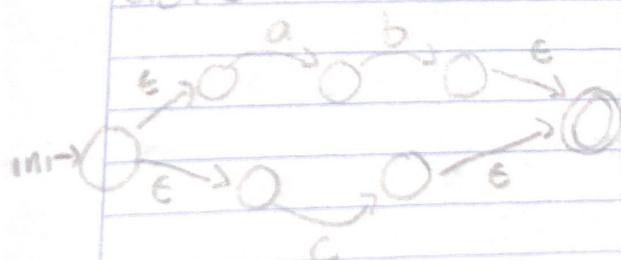
La idea es aplicar las plantillas de forma recursiva

Ejemplo Thompson

ab|c  $\rightarrow$  AFN



ab|c



Ejercicio pasay la sig.  
expresión regular

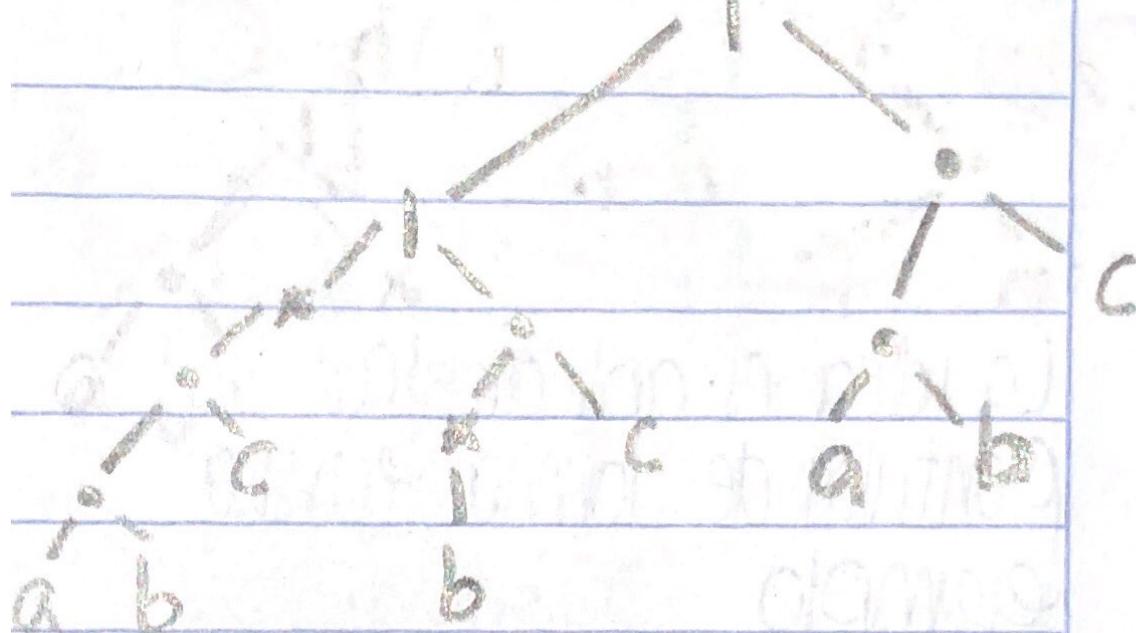
$(abc)^* \mid b^* \mid abc$  a  
un AFN por construccio  
nes de Thompson

concatenación es \*

## Árbol sintáctico

$((abc)^* \mid b^*c \mid abc)$

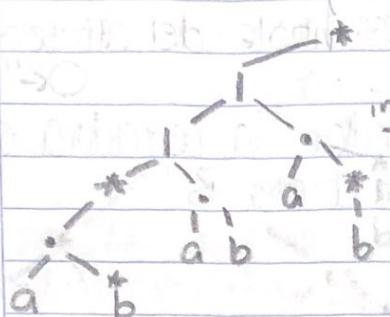
$(a \cdot \text{ or } b) \alpha c$



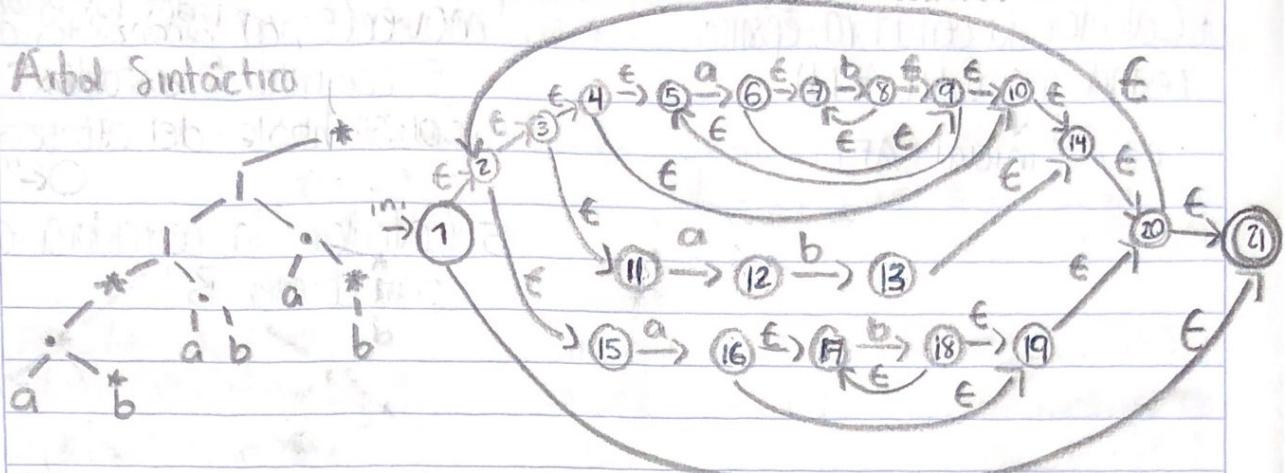
## Ejercicio Construcción de Thompson

$((ab^*)^* | ab | ab^*)^*$

### Árbol Sintáctico



### Autómata

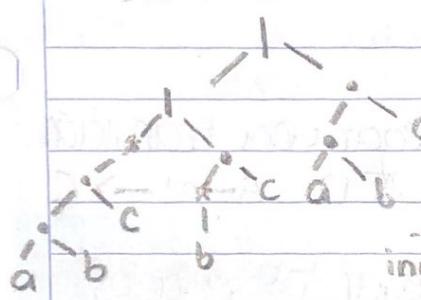


Tiempo aproximado de construcción 10 min 35 seg

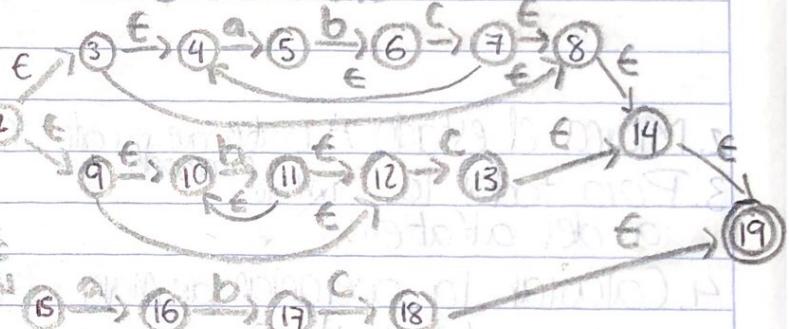
## Ejercicio Construcción de Thompson

$(abc)^* | b^* c | abc$

### Árbol Sintáctico



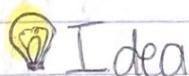
### Autómata



Tiempo aproximado de construcción 8 min 07 seg.

## Algoritmo de los subconjuntos

Sirve para pasar de un AFN (autómata finito no determinista) a AFD (autómata finito determinista)



### Idea

Operaciones auxiliares

- a) Visitar varios estados al mismo tiempo
- b) Todos los estados que se están visitando

Cerradura-épsilon (ε)

forman un estado del AFD

Todos los estados alcanzables con celo o mas transiciones épsilon.

$\Sigma$ : conjunto de estados

# Algoritmo

1. Calcular la cerradura-épsilon (estado inicial del AFN)

↓  
estado inicial AFD

retomando el autómata del ejercicio 1

$$\text{cerradura } \bar{\epsilon}(1) = \{1, 2, 3, 4, 5, 10, 11,$$

$$0 \text{ transiciones - 1} \quad 14, 15, 20, 21\} = A$$

1 transiciones - 2, 21

2 transiciones - 3, 8, 5

3 transiciones - 4, 11

4 transiciones - 5, 10

5 transiciones - 14

6 transiciones - 20

7 transiciones - 21

2. Marca el estado A (saber que ya está)

3. Para todos los símbolos <sup>procesados</sup> de  $\Sigma$

Operación auxiliar

mover(E, a) todos los estados alcanzables con una

E: conjunto de estados

transición 'a'

a: símbolo del alfabeto

5. Calcular la cerradura-é(B)

$$\text{C-}\bar{\epsilon}(B) = \{16, 12, 161, 7, 9, 5, 10, \\ 14, 20, 21, 2, 3, 4, 15, 11, 17, 19\}$$

> C

0 transiciones - 16, 12, 161

1 transición - 7, 9, 17, 19

2 transiciones - 10, 20, 5

3 transiciones - 14, 21, 2

4 transiciones - 20, 3, 15

5 transiciones - 21, 4, 11

6 transiciones - 5

4. Calcular la operación de mover

$$\text{Mover}(A, 'a') = B - \text{estado}$$

$$\text{Mover}(A, 'a') = \{6, 12, 16\} = B$$

7. Agregar los estados

nuevos al AFD e ir

al paso 2, por un estado no marcado

2 con a - no

8. Termina cuando no hay

3 con a - no

mas estados a procesar

4 con a - no

9. Marcar los estados finales

son aquellos que contienen los estados finales de AFN

5 con a - si - 6

estado A será final

10 con a - no

estado C será final

11 con a - 12

14 con a - no

15 con a - si - 16

20 con a - no

21 con a - no

## Tabla de referencia

'a': 5 → 6, 11 → 12, 15 → 16  
 'b': 7 → 8, 12 → 13, 17 → 18

Terminando el ejercicio

Paso 1 c-é(1) = {1, 2, 3, 4, 5, 10, 11, 14, 15, 20, 21} = A

Paso 2 (A, a) = {16, 12, 161, 79, 5, 10, 14, 20, 21, 2, 3, 4, 17, 11, 15, 19} = B

Paso 3 (A, b) = {} = C

Paso 4 (B, a) = {16, 12, 16, 7, 9, 5, 10, 14, 20, 21, 2, 3, 4, 17, 11, 15, 19} = B

Paso 5 (B, b) = {18, 13, 181, 7, 9, 5, 10, 14, 20, 21, 2, 3, 15, 4, 11, 17, 19} = D

Paso 6 (C, a) = {} = C <sup>núcleo o kernel</sup>

Paso 7 (C, b) = {} = C → estado de pozo

Paso 8 (D, a) = {16, 12, 161, ...} = B

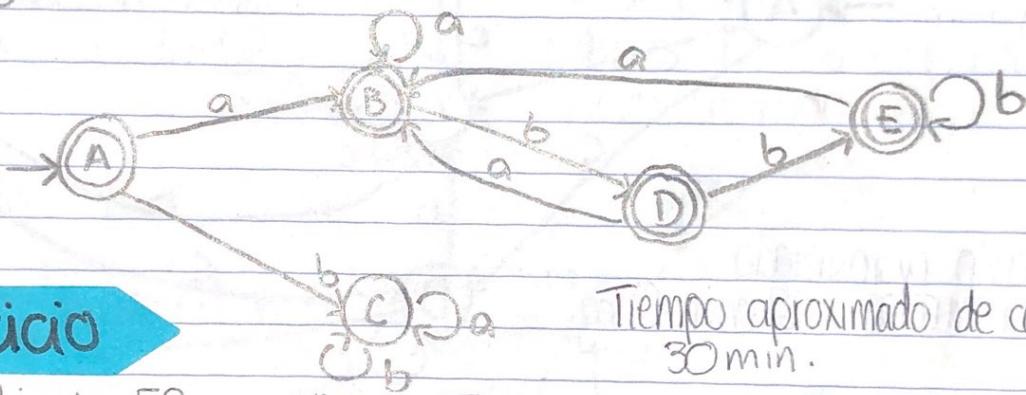
Paso 9 (D, b) = {B, 181, 7, 9, 5, 10, 14, 20, 21, 2, 3, 15, 4, 11, 17, 19} = E

Paso 10 (E, a) = {16, 12, 161, ...} = B

Paso 11 (E, b) = {18, 181, ...} = E

Paso 12 terminales: A, B, D, E ya no tengo más estados a procesar

Dibujo



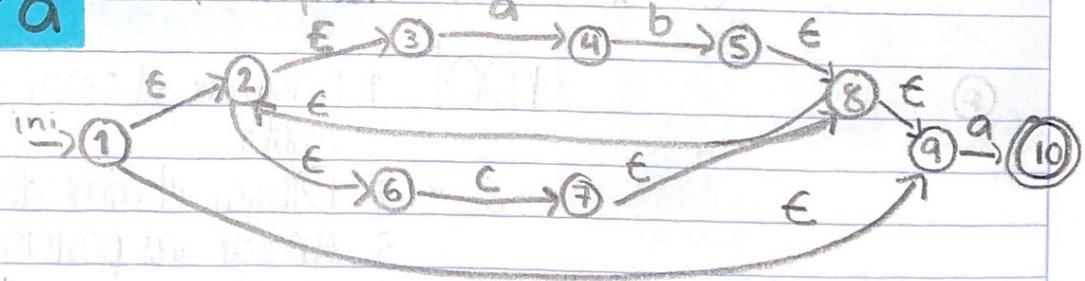
ejercicio

Tiempo aproximado de construcción  
30 min.

Convertir la ER en AFN (Thompson) AFN a AFD (subconjunto)

(ab|c)\* a

Thompson



Subconjuntos

Tabla de ref.

a: 3 → 4, 9 → 10

b: 4 → 5

c: 6 → 7

Paso 1 c-é(1) = {1, 2, 3, 6, 9} = A

2 (A, a) = {14, 10} = B

3 (A, b) = {} = C

4 (A, c) = {17, 8, 2, 9, 3, 6} = D

5 (B, a) = {} = C

$$6(B, b) = \{151, 8, 9, 2, 3, 6\} = E$$

$$7(B, C) = \{f = C\}$$

$$8(C, a) = \{f = C\}$$

$$9(C, b) = \{f = C\}$$

$$10(C, c) = \{f = C\}$$

$$11(D, a) = \{14, 10\} = B$$

$$12(D, b) = \{f = C\}$$

$$13(D, c) = \{171, \dots\} = D$$

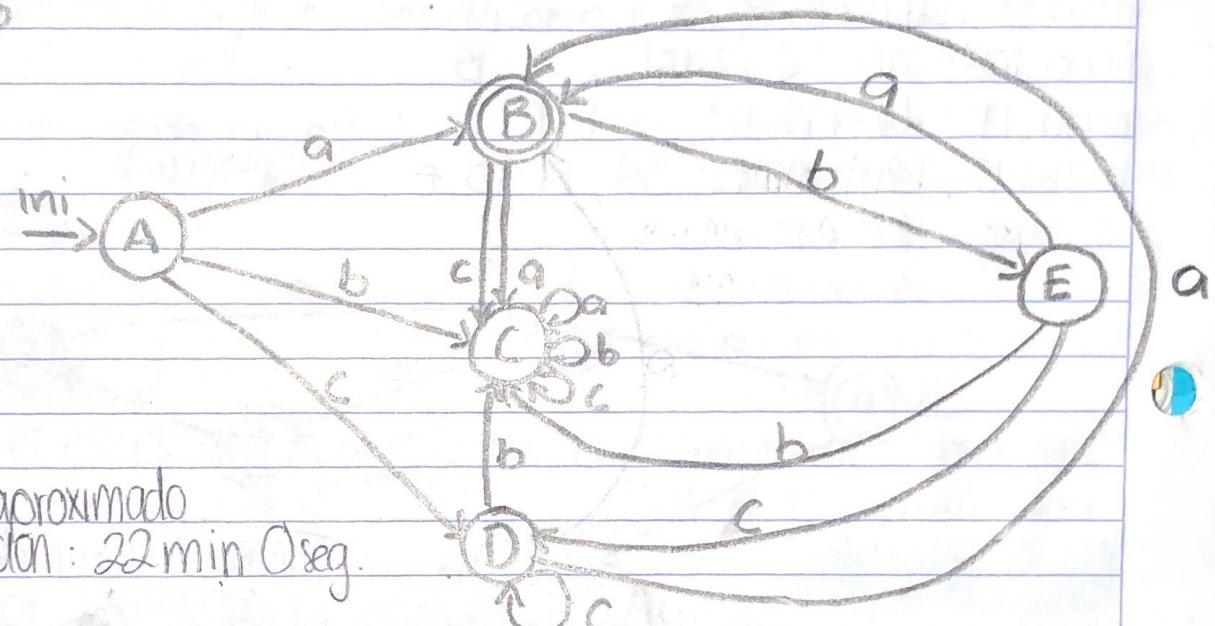
$$14(E, a) = \{14, 101\} = B$$

$$15(E, b) = \{f = C\}$$

$$16(E, c) = \{171, \dots\} = D$$

$$17 \text{ terminales} = B$$

Dibujo



Tiempo aproximado  
de construcción: 22 min 0 seg.

## Tabla de referencia

a: 4 → 5, 15 → 16

b: 5 → 6, 10 → 11, 16 → 17

c: 17 → 18, 6 → 7, 12 → 13

### Subconjuntos

$$1 \subsetneq \{1, 2, 3, 4, 8, 14, 19, 9, 10, 12, 15\} = A$$

$$2 (A, a) = \{15, 16\} = B$$

$$3 (A, b) = \{11, 12, 10\} = C$$

$$4 (A, c) = \{13, 14, 19\} = D$$

$$5 (B, a) = \{\} = E$$

$$6 (B, b) = \{6, 12\} = F$$

$$7 (B, c) = \{\} = E$$

$$8 (C, a) = \{\} = E$$

$$9 (C, b) = \{11, \dots\} = C$$

$$10 (C, c) = \{13, \dots\} = D$$

$$11 (D, a) = \{\} = E$$

$$12 (D, b) = \{\} = E$$

$$13 (D, c) = \{\} = E$$

$$14 (E, a) = E$$

$$15 (E, b) = E$$

$$16 (E, c) = E$$

$$17 (F, a) = E$$

$$18 (F, b) = E$$

$$19 (F, c) = \{7, 18, 8, 14, 19\} = G$$

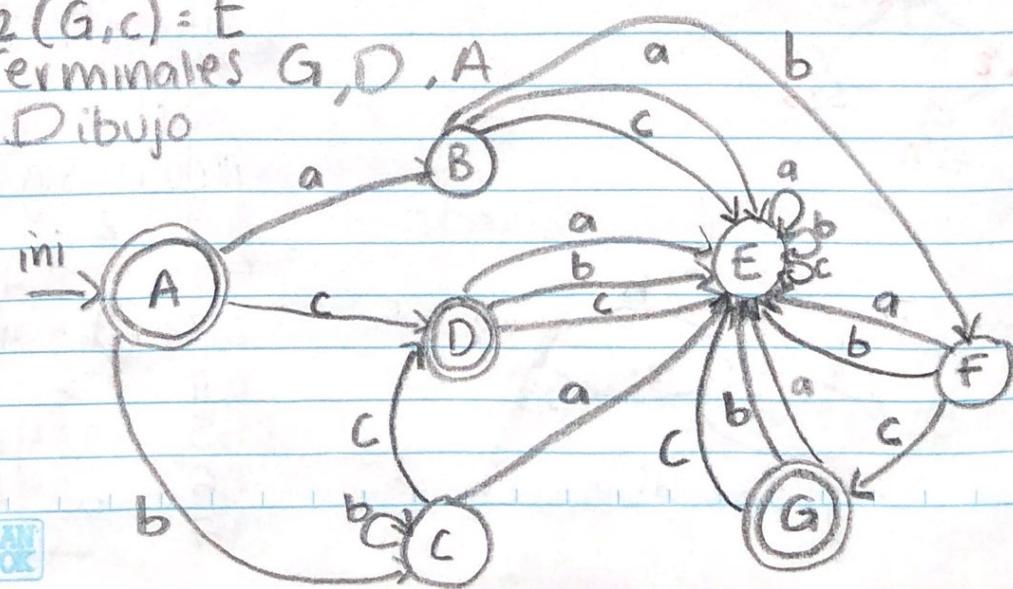
$$20 (G, a) = E$$

$$21 (G, b) = E$$

$$22 (G, c) = E$$

Terminales G, D, A

Dibujo



## Reglas para calcular siguiente

Se aplican estas reglas hasta

- que no pueda agregarse nada a cualquier conjunto siguiente

Las reglas son:

a. Colocar \$ en siguiente(s) en donde S es el símbolo inicial y \$ el delimitador derecho de Id entrada.

b. Si hay una producción  $A \rightarrow \alpha B \beta$ , entonces todo lo que hay en primero( $\beta$ ) excepto e está en siguiente( $\beta$ )

c. Si hay una producción  $A \rightarrow \alpha B \beta$ , una producción  $A \rightarrow \alpha B$  en donde primero( $\beta$ ) contiene a e entonces todo lo que hay en siguiente( $\alpha$ ) está en siguiente( $\beta$ ).

gar a  $M[A, a]$ .

2) Si e está en primero( $\alpha$ ), entonces para cada terminal b en siguiente( $\alpha$ ), se agrega  $A \rightarrow \alpha$  a  $M[A, b]$ . Si e está en primero( $\alpha$ ) y \$ se encuentra en siguiente( $\alpha$ ), se agregan  $A \rightarrow \alpha$  a  $M[A, \$]$ .

Si después de realizar lo anterior, no hay producción  $M[A, a]$ , entonces se establece  $M[A, a]$  a error (representado como entrada vacía en la tabla)

$G(S, N, T, P)$

Ejemplo construir tabla LL(1)  
Dada la gramática

$$E \rightarrow TE' \quad (1)$$

$$E' \rightarrow + TE' \quad (2)$$

$$E' \rightarrow \epsilon \quad (3)$$

$$T \rightarrow FT' \quad (4)$$

$$T' \rightarrow * FT' \quad (5)$$

$$T' \rightarrow \epsilon \quad (6)$$

$$F \rightarrow (E) \quad (7)$$

$$F \rightarrow id \quad (8)$$

lenguaje que genera la gramática

$$L(G) = \{ id + id^* id + (id^* id)^* id \mid id \neq id^*, id \neq id^* id \}$$

\$ → fin de la entrada

## Construcción de Tablas LL(1)

Son concentraciones de información de una G.L.C.

Si la tabla tiene entradas múltiples (a G.L.C. no es gramática LL(1))

Si la tabla NO contiene entradas múltiples, es LL(1).

Pasos

Para cada construcción  $A \rightarrow \alpha$  de la gramática

hacer:

- Para cada terminal a en primero( $\alpha$ ), agre-

gar a

terminales

\* de la  
producción

No  $\Rightarrow$

Terminales + \* ( ) - id \$

E (1) (1)

E' (2) (3) (3)

T (4) (4)

T' (6) (5) (6) (6)

F (7) (8)

Para (1)

P(TE') = {}

P(T) = {}

P(FT')

P(F) = {} - empiezan

con terminal  $\therefore$  lo

apunto

Por lo tanto

P(F) = {'(', id, ...}

P(FT') = {'(', id}

P(T) = {'(', id}

P(TE') = {'(', id}

Primeros ( $\lambda$ ) de todas las producciones

$E \rightarrow T E'$

Primeros (TE') = {'(', id}

Regla 4

$x_1 = T$

$x_2 = E' - ya no necesito$

(2)

$\Rightarrow$  Primeros (T) = {'(', id, ...} / P(TE') = {+?}

Regla 2

$T \rightarrow S T' - x_1 x_2$

Regla 4

$\Rightarrow$  Primeros (F) = {'(', id}

Regla 3 (E')

$\Rightarrow$  P(TE')  $\times x_2$  Regla 2

$\Rightarrow$  P('C') = Regla 1

= {} ({} )

$\Rightarrow$  P(id)  $\times x_1$   
= {id}

(3) - Siguiente

P(E) = vacío

Siguiente (E') = S(E') = {+, '}'}

¿ Cuantas veces  $\lambda$ ?

120  $E'$  de lado derecho?

- No es inicializar R1

- 2 veces.

Aplicar 2 veces la

regla 3.1

no agrego más porque  
no hay epsilon

no agrego

$E \rightarrow T E' (1)$

forma A  $\rightarrow$   $\lambda E'$  {+, '}'

Siguiente (E)

S(E) = {+, '}' regla 1

¿ Cuantas veces  $\lambda$ ?

del lado der? 1

$F \rightarrow (E) (1)$

regla 2

agregar P(')'') = {''}'

P(e)

hago el cruce con quién  
los produce

$$E' \rightarrow +TE' \\ S(E') - \text{opsi (?)}$$

tengo ciclo, para que  
también ej tener otra forma  
de calcularlo como ya  
lo hicimos

$$(7) F \rightarrow (E) \\ P((E)) = \{ '()' \}$$

$$(8) F \rightarrow \text{id} \\ P(\text{id}) = \{ \text{id} \}$$

Fin de construcción de  
tabla

$$(4) T = FT' \\ P(FT') = \{ '(', id \} \\ P(F) = \{ '(', id \}$$

$$(5) T' = *FT' \\ P(T') = \{ * \} \\ P(*) = \{ * \}$$

$$(6) T' \rightarrow E$$

$$S(T') = C \# \text{ veces q aparece?} \\ 2 \text{ veces (4)(5)}$$

Tomo (4)

$$T = FT' \quad A \rightarrow a \in E \\ S(T) = \{ +, \$, () \} \\ \# \text{ veces q aparece?} \\ 2 \text{ veces (1)(2)}$$

$$\textcircled{1} \quad S(E) \rightarrow TE' \text{ regla 3}$$

$$P(E') = \{ +, E \}$$

$$P(E) = \{ \$, () \}$$

$$\textcircled{2} \quad E' \rightarrow +TE'$$

$$\text{Tomo (5)} \quad P(E') = \{ +, E \} \\ T' \rightarrow *FT' \quad S(E') = \{ \$, () \} \\ \rightarrow S(T')$$

se cicla

## Pertenencia de L(G)

a) id + id \* id

b) (id)

c) id + \* id

Pertenece o no pertenece  
usando la tabla



Algoritmo de análisis  
LL(1)

algoritmo de análisis  
sintáctico predictivo.

Establecer ip para que apunte al  
primer símbolo de w;  
establecer X con el símbolo de la  
parte superior de la pila;  
mientras (X ≠ \$) { // la pila no estuvo vacía

si (X es a) sacar de pila y avanza ip;  
else si (X es un terminal) error();  
else si (M[X,a] es una entrada de  
error) error();

else si (M[X,a] = X → Y<sub>1</sub>, Y<sub>2</sub>, Y<sub>3</sub>, ..., Y<sub>k</sub>) {

enviar de salida la producción

X → Y<sub>1</sub>, Y<sub>2</sub>, ..., Y<sub>k</sub>;

sacar de la pila;  
meter Y<sub>k</sub>, Y<sub>k-1</sub>, ..., Y<sub>1</sub> en la pila  
con Y<sub>1</sub> en la parte superior.

2 establecer X con el símbolo de la  
parte superior de la pila;

## Ejemplo

Para a

$id + id * id \$$

entrada(w)

en tabla

tope

símbolos  
iniciales  
XXXX

fondo

$id + id * id \uparrow TE' \$$

$(T, id) = T \rightarrow FT'$

$E' \$$

push  $T' F$

$FT'E' \$$

$(F, id) = F \rightarrow id$

$id T'E' \$$

$(id, id) = pop - avanza$

$T'E' \$ \uparrow$

$(T, id) = T \rightarrow FT'$

aviso

$pop() E' \$$

push  $T' F$

$FT'E' \$ \rightarrow$  pila

$(T', *) = T' \rightarrow * FT'$

aviso  
pop  $E' \$$

push  $T' F *$

$*FT'E' \$$

$(F, id) = id$

$pop() T'E' \$ \rightarrow$  pila

$push() id T'E' \$ \rightarrow$  pila

$(*, *)$  pop avanza

$FTE' \$$

$\uparrow$

$id == id$  - coincido

$pop(); T'E' \$ \rightarrow$  pila

avanza  $+ id * id \$$

$(E, id) = F \rightarrow id$

pop()  $T'E' \$$

push  $id T'E' \$$

$(id, id) pop avanza$

$T'E' \$$

$\uparrow$

$(T', +) = T' \rightarrow E$

$pop(); E' \$ \uparrow$

$push() de 0 símbolos$

$(T', \$) = T' \rightarrow E$

$pop() E' \$$

$push() de 0 símbolos$

$E' \$$

$(E', +) = E' \rightarrow +TE'$

aviso

$pop() \$$

$push() E' T +$

$+ TE' \$$

$(+, +) pop() y avanza$

$(E', \$) = E' \rightarrow E$

$pop() \$$

$push() E$

$\uparrow$

JEAN  
BOOK

$(\$, \$) \uparrow$  termino  $\therefore$  pertenece

b) Entrada Pila Consulta

(id) \$ E \$

(id) \$ T E' \$ (E, ( ) ) = E → TE'

(id) \$ F T' E' \$ (T, ( ) ) = T → FT'

(id) \$ (E) T' E' \$ (F, ( ) ) = F → (E)

(id) \$ E ) T' E' \$ (( , ) ) empate

(id) \$ T E' ) T' E' \$ (E, id) = E → TE'

(id) \$ F T' E' ) T' E' \$ (T, id) = T → FT'

(id) \$ id T' E' ) T' E' \$ (F, id) = F → id

) \$ T' E' ) T' E' \$ (id, id) empate

) \$ E' ) T' E' \$ (T', ) ) = T → E

) \$ ) T' E' \$ (E', ) ) = E' → E

\$ T' E' \$ ( , ) ) empate

\$ E' \$ (T', ) ) = T → E

\$ \$ (E', ) ) = E' → E

\$ ( \$ ) fin

Notas!!!

Siempre se tiene que construir la tabla LL(1) aunque tenga múltiples entradas al final se hace la conclusión "hay múltiples entradas, la gramática no es LL(1)"

D (3)(7) - múltiples entradas

Puedes preguntar que sucede con:

A → A B S D

B → a s d f

C → S A S D A

D → E

∴ si pertenece  
derivación, se ponen las reglas

E → (1) → (4) → (7) → (7) → (4) → (8)  
→ (6) → (3) → (6) → (3) → (id)

X → s d j ... A S D a b c  
S(A)

X → A A A A A

S(A)

Para C Pila Consulta

id + \* id \$ E \$

id + \* id \$ T E' \$ (E, id) = E → TE'

id + \* id \$ F T' E' \$ (T, id) = T → FT'

id + \* id \$ id T' E' \$ (F, id) = F → id

+ \* id \$ T' E' \$ (T', +) = T → E

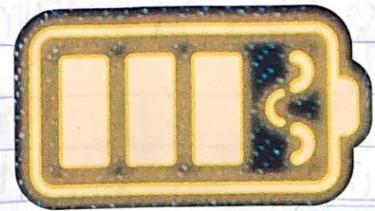
+ \* id \$ E' \$ (E', +) = E' → + TE'

\* id \$ + T E' \$ ( +, + )

\*) formado por T E' \$ (T, \*) = (error!)

∴ la cadena no pertenece

al L(G)



JEAN  
BOOK

# Analizador Sintáctico Ascendente

Parte de la familia LR (0). Estos analizadores empiezan en las hojas y buscan llegar a la raíz.

L - left to right (lectura)

R - rightmost derivation  
busca la derivación por la derecha.

O - no necesita token de anticipación

Sigue siendo método tabular

① Se construye tabla

② Se usa la tabla

Bison usa un algoritmo de fam LR

Estos métodos utilizan el algoritmo de los subconjuntos. — cerradura-e AFN — cerradura-LR mover

Los elementos LR(0)

son producciones de la gramática con un punto indicador

## algoritmo LR

Para construir la tabla se calculan los "subconjuntos LR(0)"

Los subconjuntos LR(0) son agrupaciones de "elementos"

"LR(0)" Algoritmo de los subconjuntos:  
→ subconjuntos de estados

{1, 3, 4} = A subconjunto A, contiene estados

→ funciones auxiliares:

- cerradura-ε (εclosure)

- mover()

Dado un subconjunto inicial aplicar cerradura/mover para encontrar otro subconjunto hasta ya no poder agregar más.

## elemento LR(0)

son: producción de la gramática + punto indicador

producción:

$A \rightarrow aBC$

elementos: indica x lo leí

$A \rightarrow \cdot aBC$

$A \rightarrow a \cdot BC$

$A \rightarrow aB \cdot C$

$A \rightarrow aBC \cdot$

producción:

$A \rightarrow E$

elemento:  $A \rightarrow \cdot$

## Construcción de subconjuntos

① Extender la gramática:  
agregar la producción  $S' \rightarrow S$

$S' \rightarrow S \leftarrow$

$S \rightarrow aBca$

$S \rightarrow bcA$

$S \rightarrow BC$

el ascenso a la raíz es único.

② Aplicar la cerradura ( $\{S' \rightarrow .S\}$ ) = primer subconjunto  
 ③ Para todos los símbolos

$x$  de NoTerminales

③.1 Cerradura(Mover( $C, x$ )) = nuevo subconjunto.

③.2 Agregar a los subconjuntos

## Familia LR

se les llama

así porque tienen la misma plantilla (tabla, conjuntos, pila)

saber si una cadena pertenece

LR(0) - ya

SLR - su forma sintetizada.

LR(1) → tiene producción punto token

LALR → su forma sintetizada ↑

(ventajas pueden ver mas errores)

GLR → caminos genéricos

PLR → computo paralelo si hay múltiples entradas; no son LR

Para los errores existen

varias técnicas, tokens de error gramática de error, etc.

## ejemplo

construir los subconjuntos

LR(0)

$E \rightarrow E + T$

$E \rightarrow T$

$T \rightarrow T * F$

$T \rightarrow F$

$F \rightarrow (E)$

$F \rightarrow id$

Calcular el subconjunto inicial

cerradura ( $\{E' \rightarrow .E\}$ ) =

$\{E' \rightarrow .E, E \rightarrow .E + T, E \rightarrow .T, T \rightarrow T^* F, T \rightarrow .F, F \rightarrow .(E), F \rightarrow .id\} (1)$

ciclo para todos los  $x$  ( $T$ ,  $y$ )

mover((1), E) =  $\{ | E' \rightarrow E., E \rightarrow E + T | \} (2)$   
 cerradura

$C(M((1), T)) = \{ | E \rightarrow T., T \rightarrow T^* F | \} (3)$

$(1), F = \{ | T \rightarrow F. | \} (4)$

$(1), (1) = \{ | F \rightarrow (E) |, E \rightarrow .E + T,$   
 cerradura

$E \rightarrow T, A, T \rightarrow T^* F, T \rightarrow .F, F \rightarrow (E), F \rightarrow .id \} (5)$  acept

$(1), id = \{ | F \rightarrow id. | \} (6)$

$(2), + = \{ | E \rightarrow E + T |, T \rightarrow T^* F, T \rightarrow .F, F \rightarrow .(E), F \rightarrow .id \} (7)$

$(3), * = \{ | T \rightarrow T * .F |, F \rightarrow .(E), F \rightarrow .id \} (8)$

$(5), E = \{ | F \rightarrow (E) |, E \rightarrow E + T | \} (9)$

$(5), T = (3) (5), (1) = (5) (9)$

$(5), F = (4) (5), id = (6)$  AN BOOK

$$\begin{array}{l}
 S \rightarrow (A) \quad (1) \\
 A \rightarrow CB \quad (2) \\
 B \rightarrow ;A \quad (3) \\
 B \rightarrow \epsilon \quad (4) \\
 C \rightarrow X \quad (5) \\
 C \rightarrow S \quad (6)
 \end{array}$$

$$\begin{array}{ll}
 S \quad (1) & \\
 A \quad (2) & \\
 B \quad (3) & \\
 C \quad (4) & \\
 & \text{Pertenece} \\
 & \text{sí a) } (x; (x)) \quad \text{b) } (x) \\
 & \text{no c) } (; (x)) \quad \text{d) } (; x) \text{ no} \\
 (1) & \\
 P((A)) = \{ \{ \} \} &
 \end{array}$$

$$\begin{array}{l}
 (2) \\
 P(CB) = \{ x, \{ \} \} \\
 P(C) = \{ x, \{ \} \} \\
 P(x) = \{ x \} \\
 P(S(A)) = \{ \{ \} \}
 \end{array}$$

a) entrada	pila	consulta
(x; (x)) \$	S \$	(S, ( )) = (A)
(x; (x)) \$	(A) \$	(( , ), ) = empate
x; (x)) \$	A) \$	(A, x) = CB
x; (x)) \$	(CB) \$	(C, x) = x
x; (x)) \$	xB) \$	(x, x) = empate
;(x)) \$	B) \$	(B, ;) = ;A
;(x)) \$	;A) \$	(;, ;) = empate
(x) \$	A) \$	(A, ( )) = CB
(x) \$	(B) \$	(C, ( )) = S
(x) \$	SB) \$	(S, ( )) = (A)
(x) \$	(A) B) \$	(( , ), ) = empate
x) \$	A) B) \$	(A, x) = CB
x) \$	(B) B) \$	(( , x) = x
x) \$	xB) B) \$	(x, x) = empate
) \$	B) B) \$	(B, ) ) = E
) \$	) B) \$	(( , ) ) = empate
) \$	B) \$	(B, ) ) = E
) \$	) \$	(( , ) ) = empate
) \$	\$	(( , ) ) ok
P((A)) = ( )	a $\rightarrow$ pertenece	