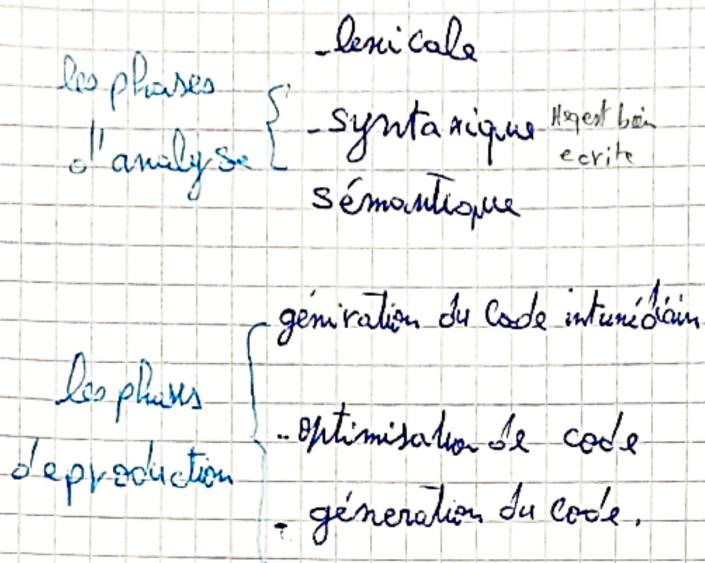


## Chapitre 01: Analyse lexical



### 1.1 Des expressions Régulières

def: on appelle Expression Régulière (E.R) sur un Alphabet A les expressions qui sont construites à partir des règles suivantes

- pour chaque  $a$  dans A,  $a$  est l'expression régulière décrivant le langage  $\{a\}$

-  $\epsilon$  est l'E.R décrivant  $\{\epsilon\}$

- si R et S sont 2 E.R décrivant les langages LR et LS alors

$(R) \cup (S)$  est l'E.R décrivant  $L_R \cup L_S$

$- (R)^*$  est l'E.R décrivant  $L_R^*$

$- R^k$  est l'E.R décrivant  $L_R^k$

Expl:  $a^* = a \cdot a^*$

2) Automates d'état finis (A.F)

Déf: un automate d'état fini M est un quintuplet  $(\$, A, f, s_0, H)$

\$ : ensemble fini d'état

A : l'alphabet

$s_0 \in \$$  : l'état initial

H :  $H \subseteq \$$  : ensemble d'état finis

f: fonction de transition  $\$ \times \$ \rightarrow H$

Propriété: le langage

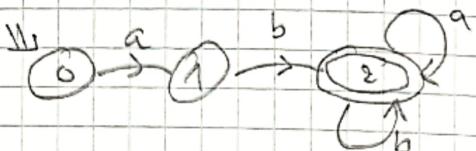
$((M))$  Reconnu par l'automate

M est l'ensemble  $\{w \mid f(s_0, w) \in H\}$

Opérateur permettant d'atteindre un état final à partir d'un état initial

son rôle est de tester l'appartenance en nom d'une chaîne au langage

Expl:  $ab(a/b)^*$



	a	b	table de transition
0	1	-	
1	-	2	
2	2	2	

2.1/ Automate d'état finis Non déterministe (AFN): un automate est Non déterministe si:

- les arcs sortants peuvent

- être étiquetés par  $\epsilon$
- un caractère peut être étiqueté plusieurs fois. Soit d'un état plusieurs AFDs.
- un automate d'état finis et déterministe (AFD) un automate est déterministe si il n'existe pas de  $\epsilon$  transition
- un caractère a de l'ame pas de étiquettes plusieurs AFDs sortant d'un même état.
- simulation d'un AFD:
- soit  $x$  une chaîne à analyser ( $x = t_1 \# t_2 \# \dots \# t_n$ )

- un AFD avec un état initial ou dispose des fonctions

transfert ( $e.c$ ): donne l'état (si  $J$ ) vers lequel  $J$  une transition de plus sur le caractère  $c$ .

- car Suiv(): Donne le prochain caractère à analyser de  $x$

$\rightarrow$  Algo

$e = e_0$

$c = \text{car Suiv}()$

langue ( $C \neq \#\#$ ) et ( $c \neq \phi$ )

$e = \text{transf}(e.c)$

$c = \text{car Suiv}()$

state

chaîne acceptée

sinon chaîne non acceptée

2) transformation d'un AFN en AFD

on dispose d'un AFN dont

$e_0$  = état initial

$T$ : ensemble d'état de l'AFN

$e$ : un état de l'AFN

on dispose des fonctions suivantes

$E$ : fermeture( $e$ ): les états de l'AFN accessible depuis  $e$  par des  $\epsilon$ -transitions

$\epsilon$ -fermeture ( $T$ ): les états de l'AFN accessible depuis tout état de  $T$  par des  $\epsilon$ -transitions

transition ( $T, a$ ): les états de l'AFN vers les quels entrée sera à partir de  $e$  ( $e \in T$ )

$\rightarrow$  Algo

D-états: ensemble d'état de l'AFD

D-trans: la table de transition de l'AFD

initiallement:  $\epsilon$ -fermeture ( $e_0$ ) unique état  $\in D$ -états

et T Marqué

$\Rightarrow Q(T)$  (T fut état à 1 marque de D'état) faire marquer

pour chaque symbole d'entrée  
 $\left\{ \begin{array}{l} \text{V} \leftarrow \epsilon \text{-fermeture}(\text{trans}(T, a)) \\ \text{si } V \in D\text{-état} \end{array} \right.$

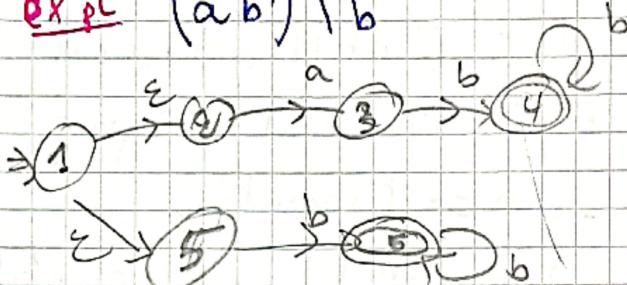
| Ajoute V à D-état et  
 U T marqué  
 fsi

$D\text{-trans}(T, a) \subset V$

FP

futq

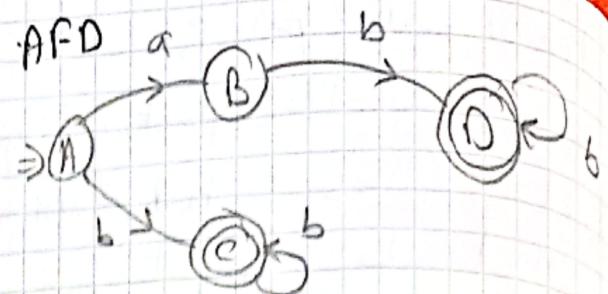
expl  $(ab^+) \setminus b^+$



$\epsilon\text{-fermeture}(1) = \{1, 2, 5\}$

AFD

	a	b
$\{1, 2, 5\}$	$\epsilon\text{-fermeture}(\text{trans}(1, 2, \{1, 2, 5\}, a))$	$\{6\}$
A	$= \epsilon\text{-fermeture}(3) = \{3\}$	
$\{3\}$	-	$\{4\}$
$\{6\}$	-	$\{6\}$
$\{4\}$	-	$\{4\}$



3° Transformation d'une ER en

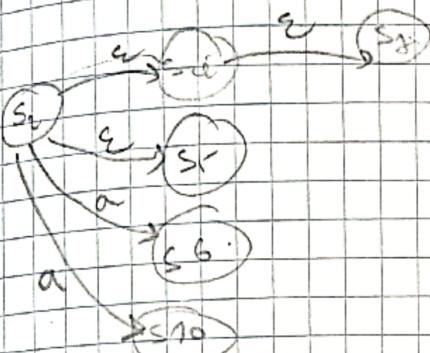
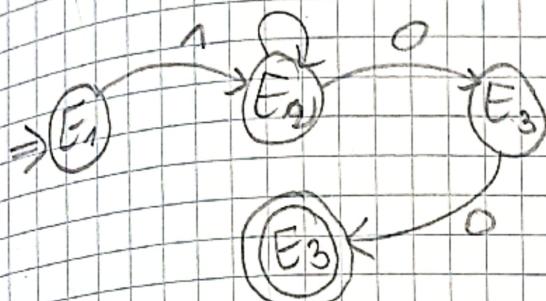
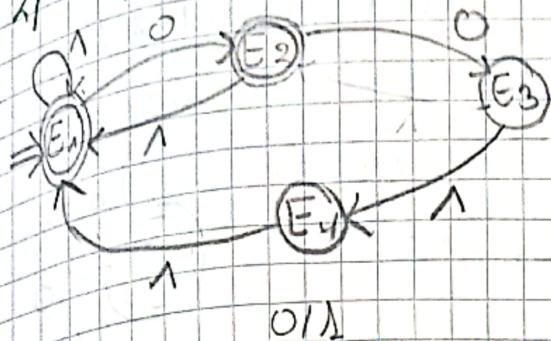
parser Generat lex  
 New project \_ save in g.c.l  
 lex g.c.l  $\rightarrow$  gcc

TOm 1 08/10/2023

Exo 1

$$11(00M/01/1)^* \cdot 0? \cdot (10)^*$$

$$21^* 1(1/0)^* \cdot 00$$



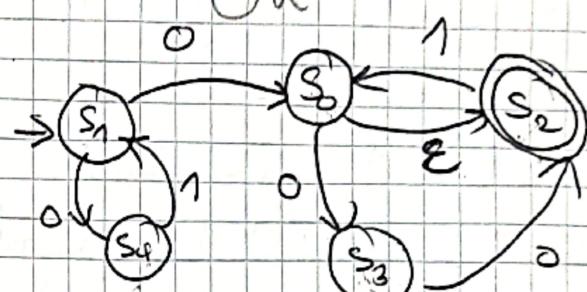
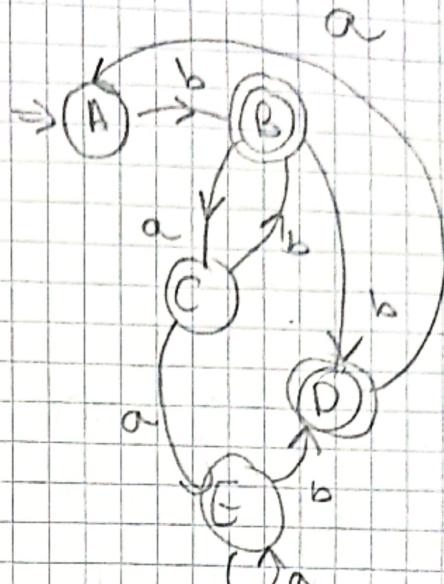
$\Sigma$ -fermeteur ( $S_2$ )  $\{S_2, S_1, S_3, S_4\}$

transitif ( $S_2, a$ )  $\{S_5, S_6\}$

Exo 2

	a	b
$S_0$	$\Sigma f(\text{tra}(S_0, a))$ —	$\Sigma f(\text{tra}(S_0, b))$ $= \Sigma f(S_1, S_2)$ $= \{S_1, S_2\}$
A		
$\{S_1, S_2\}$	$\{S_1, S_0\}$	$\{S_2\}$
$\{S_1, S_0\}$	$\{S_1\}$	$\{S_1, S_2\}$
$S_4$	$\{S_0\}$	—
$S_5$	$\{S_1\}$	$\{S_2\}$

AFD



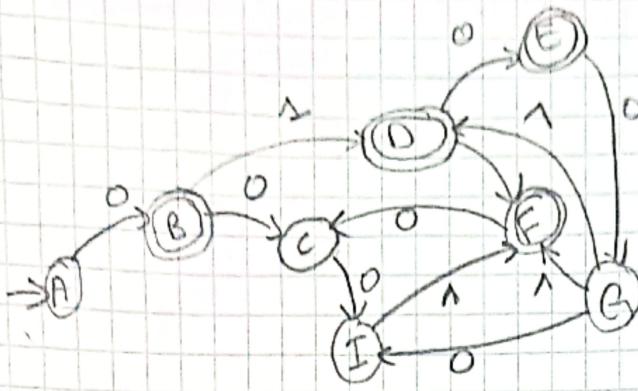
$\Sigma f_{\text{er}}(\text{transit}(S_1, 0)) =$

$\Sigma f_{\text{er}}(S_0, S_4) = \{S_0, S_1, S_2\}$

$\Sigma f_{\text{er}}(\text{tra}(S_1, 1)) =$

$\Sigma f_e(d)$

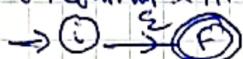
	0	1
$S_1$	A	$S_0, S_4, S_2$
$S_0$	$S_0, S_1, S_2$	B
$S_3$	C	$S_2, S_1, S_0$
$S_3$	$S_2$	—
$S_2$	$S_0, S_1, S_0$	$S_0, S_4, S_3, S_2$
$S_0, S_1, S_3, S_2$	E	$S_3, S_2$
$S_3, S_2$	F	G
$S_3$	$S_3$	—
$S_3, S_2$	$S_2$	$S_0, S_2$
$S_2$	H	$S_0, S_4, S_3, S_2$
$S_2$	I	—
$S_0, S_2$		$S_0, S_2$



3) transformation d'une E.R en AFN

→ Construction de Thompson:

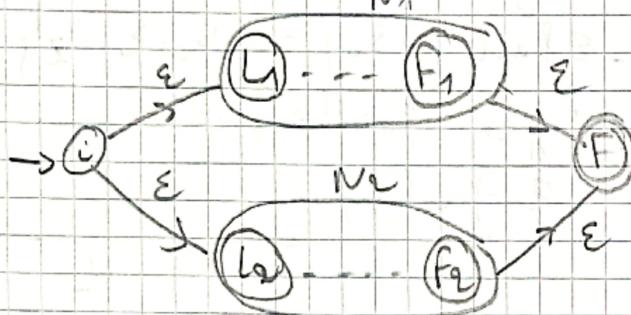
• pour  $\epsilon$  on construit l'AFN suivant:



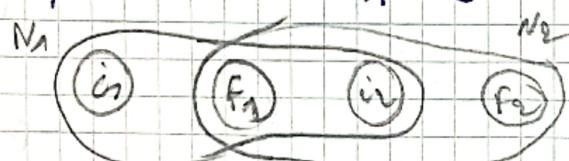
• pour  $a \in A$ :  $i \xrightarrow{a} F$

•  $N_1$  et  $N_2$  les AFN correspondant aux E.R  $R_1$  et  $R_2$ .

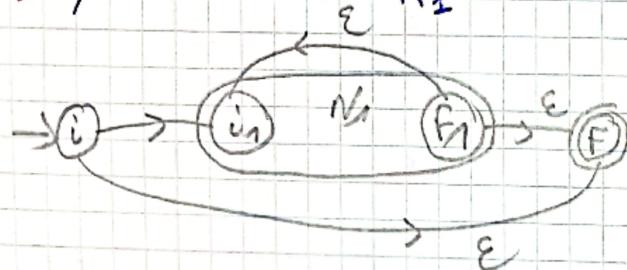
→ pour l'ER  $R_1 / R_2$



→ pour l'ER  $R_1 \cdot R_2$



→ pour l'E.R  $R_1^*$



→ propriété de la construction de Thompson:

1) # d'arcs entrant sur l'état initial de l'AFN

2) un seul état final et # d'arcs sortant de l'état final

4) LEX, générateur d'analyse lexicale

fichier.l

↓  
compilateur lex Unité lex fichier

↓  
lex.y.y.c

lex.y.y.c

↓  
compilateur c Unité gcc  
Visual Studio  
gcc  
édition  
correction  
↓  
a.out

4.1°/ Structure d'un programme  
LEX Declarations

Les délimiteurs de partie

Règles de traduction

procéduraux individuels

et déclaration

Bloc littéral `#include <stdlib.h>`

`#include <stdio.h>`

`int i, a;`  
%

• définition :

ch [0-9]

lettre [A-Z a-z]

; identifiant {lettres} {lettres} {ch})\*;

• Règles d'traduction:

- partie gauche : Spécification des

E.R à R.eConnaitre

(pour éviter certaines sources

d'ambiguité commencer par les

expressions régulières commencer

par les expressions régulières

spécifiques et écrire en suite

les expressions générales)

- partie droite : Action (écrite en e)

exécutée lorsque unités lexicales

reconnues si l'action est absente

LEX copiera les caractères

entress tels quels sur la sortie

standard.

expl printf ("un a suit par  
une suite de b");

\* procédure auxiliaires

Définir toutes les fonctions utilisées

dans les actions associées aux

expressions reconnues.

• L'analyse lexicale effectuée

- la fonction yy.lex()
- yy.text Variante de contenut de la chaîne de caractère rentrée

expl

% int mb;

% }

chaîne (aa bb)\*

% %

{ chaîne } printf ("%s :  
une suite de aa et de bb\n", yy.text); }

printf ("%s : les a  
d'abord et des b en suite  
\n", yy.text); }

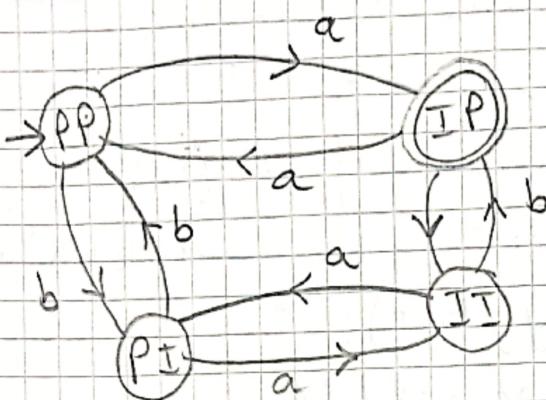
% %

main ()

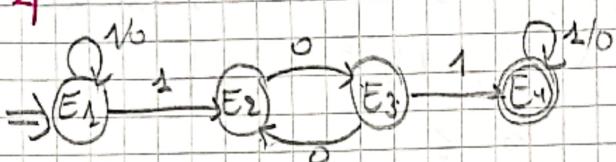
{ yy.lex (); }

TD

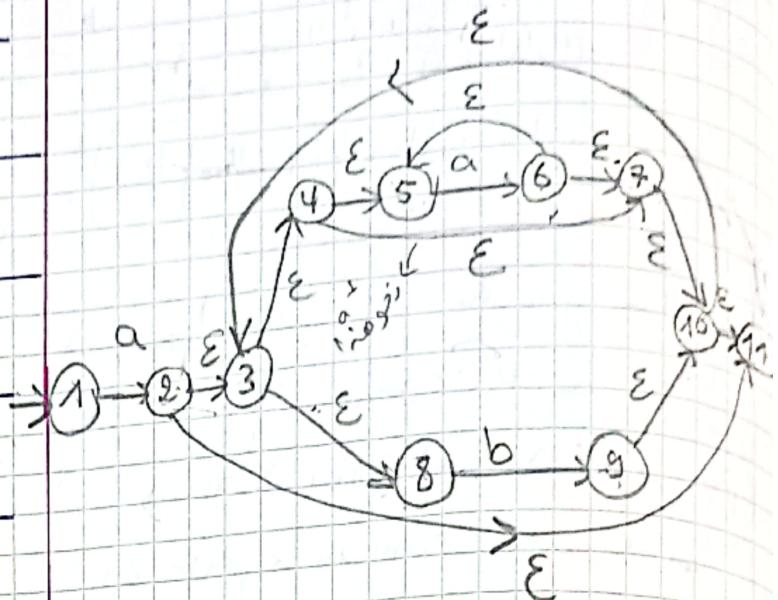
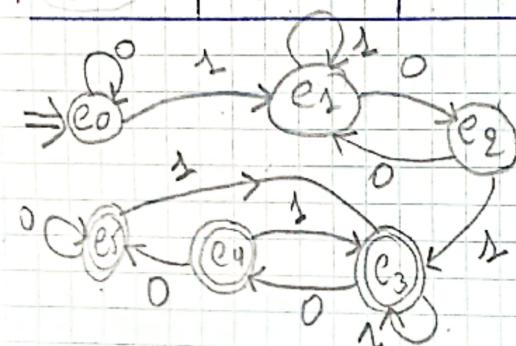
<u>EX03</u>	a	b
<u>PP</u>	I P	P I
<u>PI</u>	I I	P P
<u>II</u>	P I	I P
<u>IP</u>	PP	II



21



	0	1
$E_1 E_0$	$E_1$	$E_1 E_2$
$E_1 E_2$	$E_1 E_3$	$E_1 E_2$
$E_1 E_3$	$E_1 E_2$	$E_1 E_2 E_4$
$E_2 E_1 E_4$	$E_1 E_3 E_4$	$E_1 E_2 E_4$
$E_1 E_3 E_4$	$E_1 E_4$	$E_1 E_2 E_4$
$E_1 E_4$	$E_1 E_4$	$E_1 E_2 E_4$

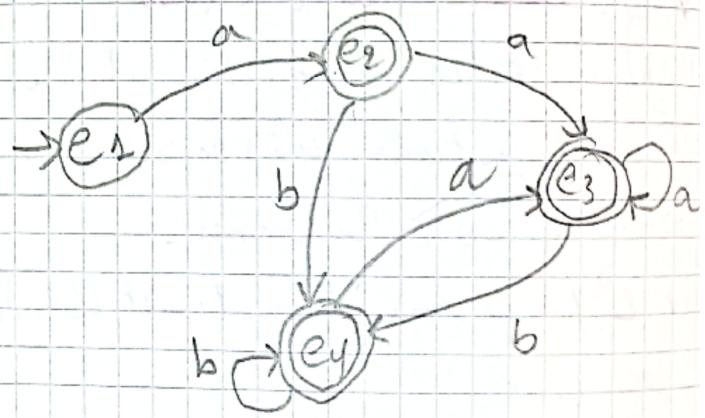


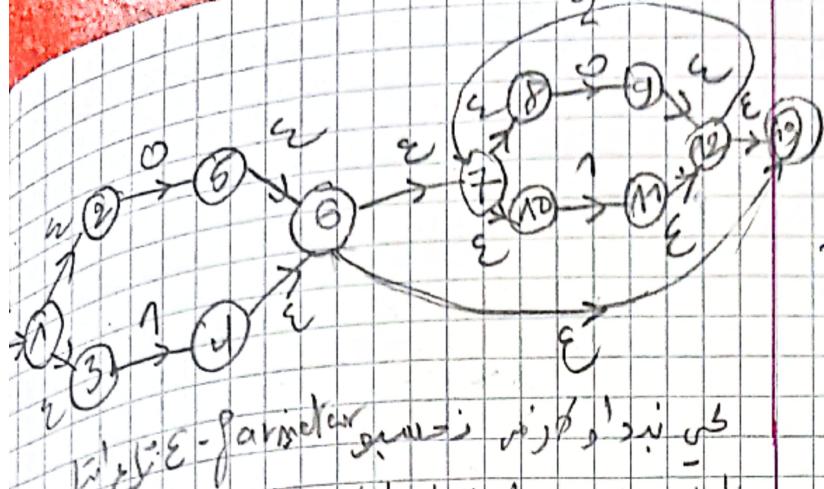
	a	b
P1 → 1	2, 3, 4, 5, 7, 8, 10, 11	—
P2	2, 3, 4, 5, 7, 8, 10, 11 6, 7, 10, 11, 3, 4, 5, 8	9, 10, 11, 3, 4, 5, 7,
P3	6, 7, 10, 11, 3, 4, 5, 8 6, 7, 10, 11, 3, 4, 5, 8	9, 10, 11, 3, 4, 5, 7, 8
P4	9, 10, 11, 3, 4, 5, 7, 8 6, 7, 10, 11, 3, 4, 5, 8	9, 10, 11, 3, 4, 5, 7, 8

$$E_1 \text{ Farmer } \left( \begin{matrix} \text{From } (1, a) \\ / \quad \backslash \end{matrix} \right) =$$

$$\Sigma_{\text{farmer}}(2) = 2, 3, 4, 5, 6, 7, 8,$$

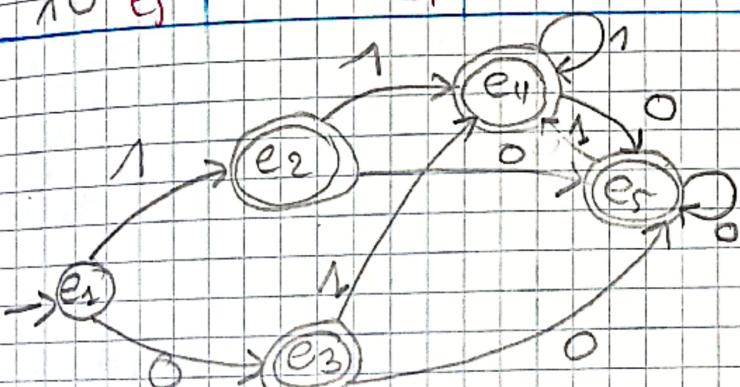
$$\Sigma \cdot \text{far}(\text{met}(\text{trans}\{\}) \cdot a) = \Sigma \cdot \text{far}(6, 7, 10, 11, 3, 4, 5, 8)$$





کی بندوں کی تحریک کی لینا ایسے  
ایسا کوئی فارمیٹر نہیں رکھ سکے  
E-farmer (1) = {1, 2, 3}

	1	0	
1, 2, 3 (e1)	13 eq	13 e3	
4, 6, 7, 8, 10	11, 12, 13, 7, 8	9, 12, 13, 7, 8	
13 e2	10 e4	10 er	
5, 6, 7, 8, 10	11, 12, 13, 7,	9, 12, 13, 7, 8	
13 e3	8, 10 e4	10 er	
11, 12, 13, 7, 8	11, 12, 13, 7,	9, 12, 13, 7, 8	dérivations les plus à droite
10 e4	8, 10 e4	10 er	
9, 12, 13, 7, 8	11, 12, 13, 7,	9, 12, 13, 7, 8	
10 er	8, 10 e4	10 er	

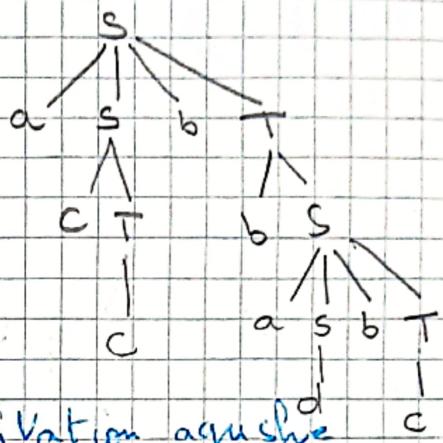


T0N3  
EX01

$$S \rightarrow a S b T | c T | id$$

$$T \rightarrow a T | b S | e$$

$$acc\ bba\ dbbc$$



derivation aggressive

$$S \rightarrow a S b T \rightarrow acc T b T$$

$$\rightarrow acc b T \rightarrow acc b b S$$

$$\rightarrow acc b b a S b T \rightarrow acc b b a t b T$$

$$\rightarrow acc b b a o l b c$$

$$S \rightarrow a S b T \rightarrow a S b b S \rightarrow$$

$$a S b b a S b T \rightarrow a S b b a t b c$$

$$\rightarrow a S b b a d b c \rightarrow ac$$

$$acc T b b a o l b c \rightarrow$$

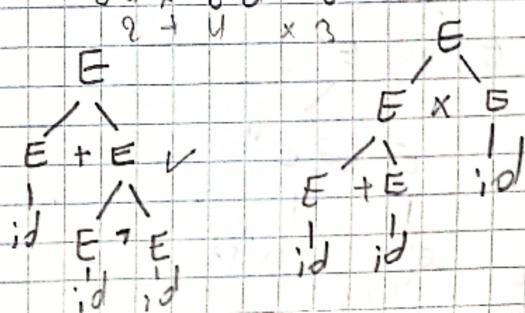
$$acc b b a o l b c$$

EX02

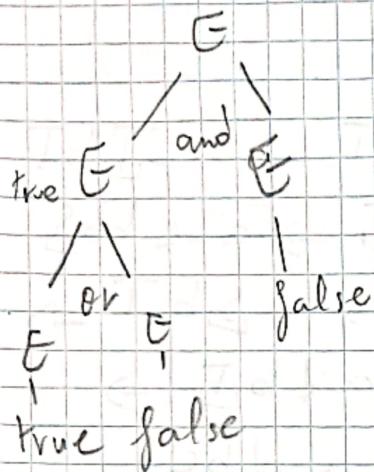
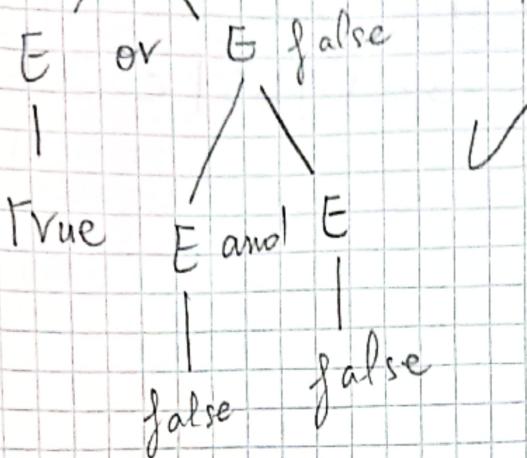
$$E \rightarrow E + E | E * E | (E) | id$$

$$id + id + id$$

$$q + u \times 3$$



$E = \text{true}$



or  $\rightarrow 1$

and  $\rightarrow 2$

( )  $\rightarrow 3$

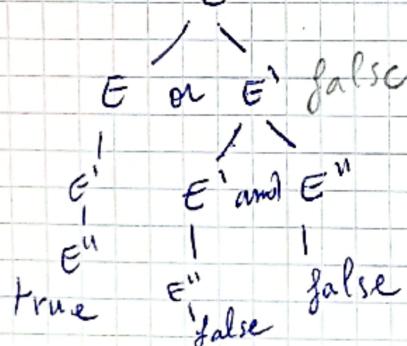
2) la grammaire

$E \rightarrow E \text{ or } E' \mid E'$

$E' \rightarrow E' \text{ and } E'' \mid E''$

$E'' \rightarrow \text{false} \mid \text{true} \mid (E)$

$E = \text{true}$



parser  $\rightarrow$  Salle  $\rightarrow$  project  $\rightarrow$

compile file  $\rightarrow$  server .output

.h file  $\rightarrow$  ... .h et ... .c

$\rightarrow$  Visual C++ Project

+ Fichiers d'en-tête

... .h

et

+ fichier source

... .c

$\rightarrow$  Générer  $\rightarrow$  Générer la Solution  $\rightarrow$  build

% yytext  $\rightarrow$  Afficher l'entrée

% { # B bibliothèque (encluse)  
Variable

% }

% % Begin

a\* b\* { printf ("%s\n"); }

a et b \n " | yytext |

couplage

% % }

main()

{ yytext(); }

exp  $s \rightarrow aTb/c$

$T \rightarrow CSSIS$

le mot acc acc bb  
+ derivation a gauche

$s \rightarrow a\overline{T}b$

$\rightarrow a\overline{SS}b$

$\rightarrow a\overline{CSS}b$

$\rightarrow a\overline{CS}\overline{S}b$

$\rightarrow acc\overline{aT}bb$

$\rightarrow accaccbb$

+ derivation a droit

$s \rightarrow a\overline{T}b$

$\rightarrow ac\overline{S}b$

$\rightarrow ac\overline{S}a\overline{T}bb$

$\rightarrow ac\overline{S}accbb$

$\rightarrow accaccbb$

l'ambiguité:

un grammaire hors contexte

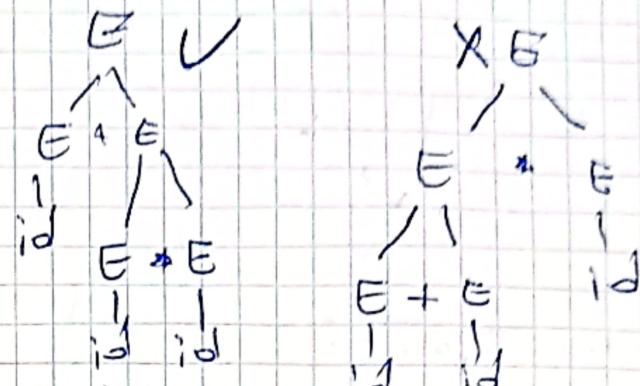
il y a ambiguïté si un mot  $w(E^*T^*)$  possède plusieurs

arbres syntaxique.

Exp

$E \rightarrow E + E / E^* E / E / Id$

Le mot ID + ID . ID



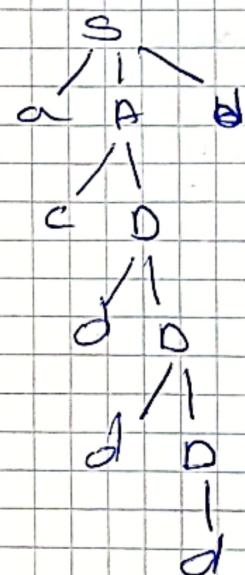
L'analyse déterm. d'at.

$s \rightarrow a\overline{Ab}$

$A \rightarrow cD/e$

$D \rightarrow dD1d$

$w = accaccbb$



\* Calcul descendante "début"

début( $x$ ) =  $\{ a \mid x \Rightarrow$

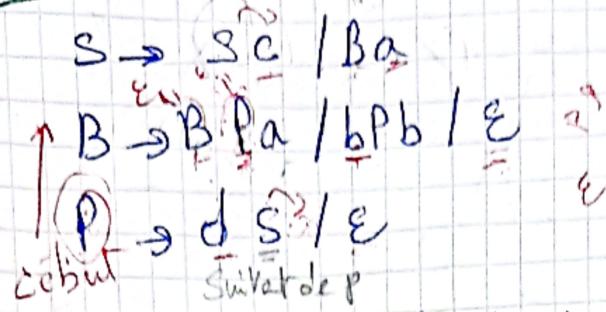
$a \alpha, \alpha \in (TUN)^*, a \in T \}$

• Si  $X$  terminal .début( $x$ ) =  $X$

• Si  $X \rightarrow E$  . ajoute  $E$  à début( $x$ )

• Si  $X \rightarrow P \frac{1}{1} \frac{1}{2} \dots \frac{1}{m}$

Exp



	debout	sivant
S	b, d, a	a, b, c, #
B	(b, b, b) a	d, a
P	d, e	b, a

الكلام هو ما هي نوسيونتاج

الفعل سار منه

رسيدونتاج

الكلام هو امر مستعار لمسنون

رسيدونتاج هناك

رسيدونتاج

رسيدونتاج

رسيدونتاج

رسيدونتاج

رسيدونتاج

رسيدونتاج

رسيدونتاج

رسيدونتاج

رسيدونتاج

$S \rightarrow S^C / Ba$

$B \rightarrow B^P a / b Pb / \epsilon$

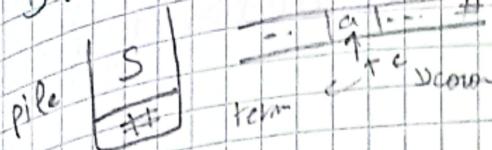
$P \rightarrow d S / \epsilon$

رسيدونتاج

رسيدونتاج

# L'analyseur L L (1)

Données.



M. Table d'analyse  $\text{L L}(1)$

$$\text{si } X = \# \text{ et } t_c = \#$$

alors chaîne correct

Syntaxiquement.

$$\text{Si } X = tc \text{ et } X \in T$$

Dépiler

$$tc \leftarrow ts$$

Si  $X$  est  $T$  terminal

$$\text{Si } M[x, tc] \text{ est } T \text{ vide}$$

Dépiler

Empiler  $MDP[M[x, t]]$  de droite à gauche ( $x \rightarrow y z w$  empiler  $w, z, y$ )

sinon

ERR eur

fin si

fin si

Analysse de la chaîne

$\text{id} + \text{id} * \text{id}$

pile	chaîne	Action
# E'	id + id * id #	$E \rightarrow T, E'$
# E' T'	id + id * id #	$T \rightarrow F, T'$
# E' T' id	id + id * id #	$T \rightarrow id$
# E' T' id	id + id * id #	dépiler avec
# E' T'	+ id + id #	$T \rightarrow \epsilon$
# E'	+ id * id #	$E \rightarrow + T E'$
# E' T +	+ id * id #	dépiler avec
# E' T'	id * id #	$T \rightarrow FT'$
# E' T'	id * id #	$F \rightarrow id$
# E' T' id	id * id #	dépiler avec
# E' T'	* id #	$T \rightarrow * FT'$
# E' T' F *	* id #	dépiler avec
# E' T' F *	id #	$F \rightarrow id$
# E' T' id	id #	dépiler avec
# E'	#	$E \rightarrow \epsilon$
#	#	chaîne

syntaxique  
correct

+	*	(	)	id	#	#
$E \rightarrow E'$	$E \rightarrow E'$	$E \rightarrow E'$				
$E \rightarrow id$	$E \rightarrow id$	$E \rightarrow \epsilon$	$E \rightarrow \epsilon$	$E \rightarrow \epsilon$		
$T \rightarrow FT'$		$T \rightarrow FT'$		$T \rightarrow FT'$		
$T \rightarrow \epsilon$						
$F \rightarrow (E)$		$F \rightarrow id$				

## \* grammaire LL(1)

Une grammaire est dite LL(1) si  $\forall A \rightarrow \alpha / \beta (\alpha \beta) \in T^* V(N)^*$

Début (α . suivant(A))  $\cap$

Début (β . suivant(A)) =  $\emptyset$

\* PPTés :  $\Rightarrow$  si la table d'analyse

Contient au plus une production par case alors G est LL(1)

$\rightarrow$  si G est LL(1)  $\Rightarrow$  G est

LL(K) ;  $\forall K \geq 1$

### Ex 1

$$G: S \rightarrow Sc | Ba$$

$$B \rightarrow BPa | bpb | P | \epsilon$$

$$P \rightarrow d | s$$

	Début	Suivant
S	b, d, a	a, b, c, $\epsilon$
B	$\epsilon, b, d, a$	d, a
P	d, $\epsilon$	a, b

$$S \rightarrow Sc | Ba$$

$$\text{Déb}(Sc, \text{suiv}(S)) \cap \text{Déb}$$

$$(Ba, \text{suiv}(S)) \stackrel{?}{=} \emptyset$$

$$\{b, d, a\} \cap \{b, d, a\} \neq \emptyset$$

$$\emptyset \Rightarrow G \text{ n'est pas LL(1)}$$

$$B \rightarrow BPa | \epsilon$$

$$\begin{aligned} & \text{Déb}(BPa, \text{suiv}(B)) \cap \text{Déb}(\text{suiv}, \\ & = \{b, d, a, d\} \cap \{d, a\} \neq \emptyset \end{aligned}$$

### Ex 2

$$S \rightarrow aA b$$

$$A \rightarrow \overline{c \quad d} \overline{e \quad \epsilon}$$

	Déb	Suiv
S	a	#
A	c, e	b

$$\text{Déb}(cd, \text{suiv}(A)) \cap$$

$$\text{déb}(e, \text{suiv}(A)) = \emptyset$$

$$\{c\} \cap \{e\} = \emptyset$$

### Ex 3

$$E \rightarrow TE'$$

$$E' \rightarrow + TE' | \epsilon$$

$$\text{déb}(+, \text{suiv}(E')) \cap (\text{suiv} E')$$

$$\{+\} \cap \{\text{suiv}(E')\}$$

$$= \{+\} \cap \{+, \#\} = \emptyset$$

### Ex 4

$$T \rightarrow FT$$

$$T \rightarrow *FT' | \epsilon$$

$$\text{début} \left( * \cdot \text{suiv}(t) \right) \cap (\text{suiv}(t)) = \emptyset$$

$$\{ * \} \cap \{ * \} = \emptyset$$

Ex 1

$$F \rightarrow (E) / \omega$$

$$\{ \} \cap \{ \omega \} = \emptyset$$

$$\Rightarrow \text{G est LL(1)}$$

TD 05/11/2023

Ex 01 / SN

$$E \Rightarrow OEE / A$$

$$O \Rightarrow + / *$$

$$A \Rightarrow m \mid (E)$$

	début	suivant
E	m, (+, *)	(,), m, +, *, #
O	+ / *	m, (, +, *)
A	m, (	(,), m, +, *, #

2) La grammaire est-elle LL(1) ?

$$E \rightarrow OEE / A$$

$$= \text{début}(OEE, \text{suiv}(E)) \wedge \text{début}(A, \text{suiv}(E))$$

$$= \{ +, * \} \cap \{ m, ( \} = \emptyset$$

$$O \rightarrow + / *$$

$$= \text{début}(+, \text{suiv}(O)) \wedge \text{début}(*, \text{suiv}(O))$$

$$= \{ + \} \cap \{ * \} = \emptyset$$

$$A \rightarrow m \mid (E)$$

$$\text{début}(m, \text{suiv}(A)) \wedge \text{début}( (, \text{suiv}(A)) = \{ m \} \cap \{ ( \} = \emptyset$$

donc G est LL(1)

3) La table d'analyse

	+	*	m	(	)	#
E	$E \rightarrow OEE$	$E \rightarrow OEE$	$E \rightarrow A$	$E \rightarrow *$		
O	$O \rightarrow +$	$O \rightarrow *$				
A			$A \rightarrow m$	$A \rightarrow (E)$		

G est non LL(1) car il y a plusieurs dérivations pour la même chaîne.

Table d'analyse Monodéfinie  $\Rightarrow$  G est LL(1)

G est LL(1)

4) Analyse de chaîne (+ \* m m m)

pile	chaîne	Action
# E	(+ * m m m) #	$E \rightarrow A$
# *	(+ * m m m) #	$A \rightarrow (E)$
# ) E	(+ * m m m) #	dépiler, afficher
# ) E E	+ * m m m) #	$E \rightarrow OEE$
# ) E E *	+ * m m m) #	$O \rightarrow +$
# ) E E *	+ * m m m) #	dépiler, afficher
# ) E E E	* m m m) #	$E \rightarrow OEE$
# ) E E E *	* m m m) #	$O \rightarrow *$
# ) E E E *	* m m m) #	dépiler, afficher
# ) E E E A	* m m m) #	$E \rightarrow A$
# ) E E E A	* m m m) #	A $\rightarrow m$
# ) E E m	* m m m) #	dépiler, afficher

#) E	M M ) #	$E \rightarrow A$
#) E A	M m ) #	$A \rightarrow M$
#) E M	M m ) #	d'épile, Alors
#) E	M ) #	$E \rightarrow A$
#) J A	M ) #	$A \rightarrow M$
#) m	M ) #	(épile, Alors) <sup>*</sup> l'
#	#	chaine est Accepter

Factorisation à gauche d'une grammaire

$$A \rightarrow \alpha \beta_1 | \alpha \beta_2$$

$(\alpha, \beta)$  = E est  $\beta$  ne commence pas par  $\alpha$  à Remplacer par

$$A \rightarrow \alpha X | Y$$

$$X \rightarrow \beta_1 | \beta_2 | \dots | \beta_m$$

### Expt

$$S \rightarrow a D I a A I d$$

$$A \rightarrow D y | A y | D ly$$

$$D \rightarrow D d I d$$

à Remplacer par

$$S \rightarrow a X I d$$

Grammaire après factorisation

$$X \rightarrow D I A$$

(Factorisation)

$$A \rightarrow D x | A y | y$$

$$x \rightarrow y | \epsilon$$

$$D \rightarrow D d I d$$

\* La Recursivité gauche une grammaire est Recursive gauche si  $\exists A \in N$  tq:

$$A \Rightarrow^+ A \alpha$$

→ Elimination de la récursivité gauche

$$A \rightarrow A \alpha_1 | A \alpha_2 | \dots | A \alpha_m | B_1 | \dots | B_n$$

(où les  $B_i$  ne commencent pas par A)

à Remplacer par:

$$A \rightarrow B_1 A' | B_2 A' | \dots | B_n A'$$

### Exo

$$S \rightarrow \{ S; A \} | A$$

$$A \rightarrow A (-A) | a | b$$

	début	suiv
S	{, a, b	; #
A	a, b	(, ;, +, )

2) <sup>alg</sup> ~~(S)~~ → ~~(A)~~ col = Def( $A, \text{ suiv}(S)$ )

;	a	b	{	}	(	)	-	#
S	$S \xrightarrow{A}$	$S \xrightarrow{B}$	$S \xrightarrow{a}$	$S \xrightarrow{b}$				
A	$A \xrightarrow{a}$	$A \xrightarrow{b}$						

table multidefinie  $\Rightarrow$  G 7 LL(1)

④ ne peut pas Analyse Pa  
classe car G 7 LL(1)

$$A' \rightarrow \alpha_1 A' | \alpha_2 A' | \dots | \alpha_n A' | \epsilon$$

$\Rightarrow$  Elimination de la Récurivité  
gauche ordonner les

Terminaux:  $A_1, A_2, \dots, A_m$   
pour  $i = 1 \text{ à } m$  faire

pour  $j = 1 \text{ à } i-1$  faire

Remplacer chaque  
production  $A_i \rightarrow A_j \gamma$

par

$$A_i \rightarrow S_1 \gamma | S_2 \gamma | \dots | S_k \gamma$$

ou

$A_j \rightarrow S_1 | \dots | S_k$  sont  
les production

fin pour

Éliminer la Récurivité des  
 $A_i$  production

fin pour

### Expl

$$E \rightarrow E \underbrace{+ T}_{\alpha_1} | T \underbrace{\epsilon}_{\alpha_2}$$

$$T \rightarrow T \underbrace{* F}_{\alpha_1} | F \underbrace{| F}_{\alpha_2}$$

Remplacer par :

$$E \rightarrow T E'$$

$$E' \rightarrow + T E' | \epsilon$$

$$T \rightarrow F T$$

$$T \rightarrow * F T' | \epsilon$$

### Expl

$$S \rightarrow S a | T S c | d \xrightarrow{\text{R. G}} \text{olive parallèle}$$

$$T \xrightarrow{\text{R. Grindley}} \overline{S} b T | \epsilon$$

TD 19/11/2023

### Exercice 3

G est TLL(1) car elle est  
récurative gauche et  
T factorisée

$(xa + xb)$  T factorisé

$x(a+b)$  factorisé

<inst> (1)

if <cond>

The <inst>

if <cond> Then <inst> else <inst>

(2)

<inst>

if <cond> Then <inst> else <inst>

if <cond> Then <inst>

A → α X<sub>1</sub> / α X<sub>2</sub> / α X<sub>n</sub> / β

A → α X / β

X → X<sub>1</sub> / X<sub>2</sub> / X .. X

A → A α<sub>1</sub> / A α<sub>2</sub> ... A α<sub>n</sub> / B<sub>1</sub> / B<sub>n</sub>

A → B<sub>1</sub> A<sup>1</sup> / B<sub>2</sub> A<sup>1</sup> ... B<sub>m</sub> A<sup>1</sup>

A' → α<sub>1</sub> A' / α<sub>2</sub> A' ... α<sub>m</sub> A' / ε

<inst> → if <cond> then <inst> X |

B Begin <list> end | i

X → else <inst> | ε

<cond> → C

<list> → ; i <list>

<list> → ; i <list> | ε

	début	suivant
<inst>	if, Begin, i	else, #
X	else, ε	else; #
<cond>	C	then
<list>	i	end
<list>	; i, ε	end

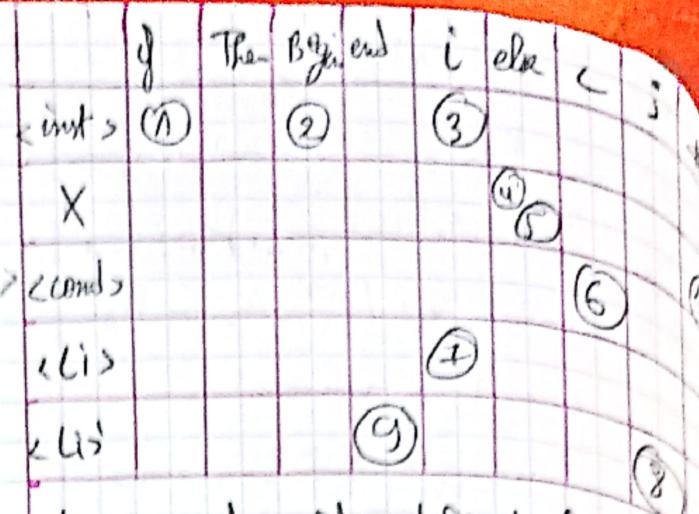


table est multi-définie alors

TLLL(1)

en lever la production (4)

Exo 4

E → E or T | T

T → T and F | F

F → not F | (E) | true | false

E → T E'

E' → OR TE' | ε

T → FT'

T' → and FT' | ε

F → not F | (E) | true | false

Exple La récursivité indirecte  
 $S \rightarrow S a T S e l d$  <sup>Récurseur direct</sup>  
 $T \rightarrow S b T \mid e$  récursivité indirecte

1. Règle

$$A \rightarrow A \alpha_1 \mid A \alpha_2 \mid A \alpha_M \dots \mid B_1 \dots B_n$$

$$A \rightarrow B_1 A^1 \mid B_2 A^1 \mid \dots \mid B_n A^1$$

$$A^1 \rightarrow \alpha_1 A^1 \mid \alpha_2 A^1 \mid \dots \mid \alpha_M A^1 \mid \epsilon$$

ordre des règles plus générales

$$S \rightarrow \textcircled{S} \rightarrow \textcircled{T} \rightarrow \dots$$

$$S \rightarrow S a \mid T S e l d \mid \dots$$

Remplacer par

$$S \rightarrow T S C S \mid D S$$

$$S' \rightarrow a S' \mid \epsilon$$

Remplacer par  $S$  dans  $T$

$$T \rightarrow S b T \mid e$$

$$T \rightarrow S C S \mid D S$$

$$T \rightarrow T S C S \mid b T \mid d S \mid b T \mid e$$

$$T \rightarrow D S b T \mid T \mid e T$$

$$T \rightarrow S C S b T \mid T \mid e$$

Mauvaise grammaire

$$S \rightarrow T S C S \mid D S$$

$$S' \rightarrow a S' \mid \epsilon$$

$$T \rightarrow D S b T \mid T \mid e T$$

$$T \rightarrow S C S b T \mid T \mid \epsilon$$

## Chapitre 5

$$S \rightarrow a T b \mid a$$

$$T \rightarrow c S \cdot b \mid d$$

$$a \quad c \quad d \quad b \quad b$$

### Sémantique LR

L : Left to right parsing  
constructing "a"

R : Right Most Derivation  
in reverse using Mg.

1. Tolent OTake an Analysis decision.

### Opération Goto

I : un état

X : un symbole de la grammaire

Goto(I, X) = fermeture  $(A \rightarrow \alpha X \beta)$

$$= I \bar{J}$$

tg:  $[A \rightarrow \alpha, B]$  est dans I.

Expl construction de la collection d'items LR(0)

$$E \rightarrow E + T \mid T \quad \text{SLR}$$

$$T \rightarrow T * F \mid F \quad \text{method}$$

$$F \rightarrow (E) \mid \dots$$

$$I_0 = \{ [E' \rightarrow \cdot E], [E \rightarrow \cdot E + T], [E \rightarrow \cdot T] \}$$

$$[T \rightarrow \cdot T * F], [T \rightarrow \cdot F], [F \rightarrow \cdot (E)],$$

$$[F \rightarrow \cdot i] \}$$

$$I_1 - \text{Goto}(I_0, E) = \{ [E' \rightarrow E_0],$$

$$[E \rightarrow E_0 + T], \dots \}$$

$$I_2 = \text{Goto}(I_0, T) = \{ [T \rightarrow T_0 \cdot F] \},$$

eigne  
[T → T<sub>0</sub>] { SWV(E) }

$$I_3 = \text{Goto}(I_0, F) = \{ [T \rightarrow F \cdot T_0] \}$$

$$I_4 = \text{Goto}(I_{01}, \cdot) = \{ [F \rightarrow (\cdot E)]_1, [E \rightarrow \cdot E + T], [E \rightarrow \cdot T] \mid [T \rightarrow T \cdot F], [F \rightarrow \cdot (E)]_2 \mid [F \rightarrow \cdot T] \}$$

$$I_5 = \text{Goto}(I_0, i) = \{ [F \rightarrow i \cdot T] \}$$

$$I_6 = \text{Goto}(I_{11}, +) = \{ [E \rightarrow E + \cdot T], [T \rightarrow T \cdot F], [F \rightarrow \cdot F], [F \rightarrow \cdot (E)]_1, [F \rightarrow \cdot i] \}$$

$$I_7 = \text{Goto}(I_{11}, *) = \{ [T \rightarrow T \cdot \cdot F], [F \rightarrow \cdot (E)]_2, [F \rightarrow \cdot i] \}$$

$$I_8 = \text{Goto}(I_{11}, \in) = \{ [F \rightarrow (E \cdot)]_1, [E \rightarrow E \cdot + T] \}$$

$$= \text{Goto}(I_{11}, \tau) = \{ [E \rightarrow T_0] \}$$

$$[T \rightarrow T \cdot \cdot f] = I_2$$

$$\text{Goto}(I_{11}, f) = \{ [T \rightarrow F \cdot] \} = I_3$$

$$\text{Goto}(I_{41}, \cdot) = I_{41} \quad \dots$$

$$\text{Goto}(I_0, i') = I_5$$

$$I_9 = \text{Goto}(I_6, T) = \{ [E \rightarrow E \cdot + T_0], [T \rightarrow T \cdot \cdot F] \}$$

$$\text{Goto}(I_6, F) = I_3$$

$$\text{Goto}(I_6, \cdot) = I_4$$

$$\text{Goto}(I_6, i) = I_5$$

$$I_{10} = \text{Goto}(I_7, f) = \{ [T \rightarrow T \cdot f] \}$$

$$\text{Goto}(I_{71}, \cdot) = I_4$$

$$\text{Goto}(I_7, i) = I_5$$

$$I_{11} = \text{Goto}(I_8, \cdot) = \{ [F \rightarrow (E \cdot)]_3 \}$$

$$\text{Goto}(I_8, +) = I_6$$

$$\text{Goto}(I_9, \cdot) = I_7$$

SWV

E	/	i	+	1	#			
T	*							
F			*	,	1	+	1	#

	+	*	(	)	?	#	E	T	F
0					D <sub>4</sub>	D <sub>5</sub>	①	②	③
1	D <sub>6</sub>								
2	R <sub>2</sub>	D <sub>7</sub>			R <sub>2</sub>	R <sub>2</sub>			
3	R <sub>4</sub>	R <sub>4</sub>			R <sub>4</sub>	R <sub>4</sub>			

4			D <sub>4</sub>	D <sub>5</sub>	⑧	②	③	
5	R <sub>6</sub>	R <sub>6</sub>		R <sub>6</sub>	R <sub>6</sub>			
6			D <sub>4</sub>	D <sub>5</sub>	⑨	③		
7			D <sub>4</sub>	D <sub>5</sub>			⑩	

8	D <sub>6</sub>			D <sub>6</sub>				
9	R <sub>1</sub>	D <sub>7</sub>		R <sub>1</sub>	R <sub>1</sub>			
10	R <sub>3</sub>	R <sub>3</sub>		R <sub>3</sub>	R <sub>3</sub>			
11	R <sub>5</sub>	R <sub>5</sub>		R <sub>5</sub>	R <sub>5</sub>			

8	D <sub>6</sub>			D <sub>6</sub>				
9	R <sub>1</sub>	D <sub>7</sub>		R <sub>1</sub>	R <sub>1</sub>			
10	R <sub>3</sub>	R <sub>3</sub>		R <sub>3</sub>	R <sub>3</sub>			
11	R <sub>5</sub>	R <sub>5</sub>		R <sub>5</sub>	R <sub>5</sub>			

8	D <sub>6</sub>			D <sub>6</sub>				
9	R <sub>1</sub>	D <sub>7</sub>		R <sub>1</sub>	R <sub>1</sub>			
10	R <sub>3</sub>	R <sub>3</sub>		R <sub>3</sub>	R <sub>3</sub>			
11	R <sub>5</sub>	R <sub>5</sub>		R <sub>5</sub>	R <sub>5</sub>			

8	D <sub>6</sub>			D <sub>6</sub>				
9	R <sub>1</sub>	D <sub>7</sub>		R <sub>1</sub>	R <sub>1</sub>			
10	R <sub>3</sub>	R <sub>3</sub>		R <sub>3</sub>	R <sub>3</sub>			
11	R <sub>5</sub>	R <sub>5</sub>		R <sub>5</sub>	R <sub>5</sub>			

8	D <sub>6</sub>			D <sub>6</sub>				
9	R <sub>1</sub>	D <sub>7</sub>		R <sub>1</sub>	R <sub>1</sub>			
10	R <sub>3</sub>	R <sub>3</sub>		R <sub>3</sub>	R <sub>3</sub>			
11	R <sub>5</sub>	R <sub>5</sub>		R <sub>5</sub>	R <sub>5</sub>			

8	D <sub>6</sub>			D <sub>6</sub>				
9	R <sub>1</sub>	D <sub>7</sub>		R <sub>1</sub>	R <sub>1</sub>			
10	R <sub>3</sub>	R <sub>3</sub>		R <sub>3</sub>	R <sub>3</sub>			
11	R <sub>5</sub>	R <sub>5</sub>		R <sub>5</sub>	R <sub>5</sub>			

In Table SLR(1) ist Monodrama  
 $\Rightarrow$  Gest SLR(1)

Algorithm d'analyse pour  
SLR, LR, LALR

la pile contient initialement  
l'état I initial

- l'asymbole courant de  $\Sigma^{\#}$

-  $E_K$  état en sommet de pile

- si  $A[E_i, a] = \text{Décler} E_j$

Empiler  $a$ , Empiler  $E_j$ ,  
et varcer sous la chaîne

sinon

si  $A[E_i, a] = \text{Réduire par } B \rightarrow \alpha$

Dépiler ( $\alpha$ )

sous  $E_K$  état en sommet de pile

Empiler ( $B$ )

Empiler ( $A[E_K, B]$ )

sinon

si  $A[E_i, a] = \text{Accepter } (\alpha = \#)$   
[chaîne syntaxiquement corrigée]

Sinon  
Erroné

Fini

Fini

Fini

pile	chaine	Action
$\emptyset$	$\text{id} + \text{id} + \text{id} \#$	DR
$\emptyset id \#$	$+ \text{id} \# \text{id} \# R_2(F \rightarrow id)$	
$\emptyset R_3 \#$	$+ \text{id} \# \text{id} \# R_1(T \rightarrow F)$	
$\emptyset T \#$	$+ \text{id} \# \text{id} \# R_2(E \rightarrow T)$	
$\emptyset E_1 \#$	$+ \text{id} \# \text{id} \# D_S$	
$\emptyset E_1 + \text{id}$	$\text{id} \# \text{id} \# DR$	
$\emptyset E_1 + \text{id} \#$	$* \text{id} \# R_6(F \rightarrow id)$	
$\emptyset E_1 + \text{id} F_3$	$* \text{id} \# R_4(T \rightarrow F)$	
$\emptyset E_1 + \text{id} T_9$	$* \text{id} \# D_T$	
$\emptyset E_1 + \text{id} T_9 \#$	$\text{id} \# D_S$	
$\emptyset E_1 + \text{id} T_9 \# \#$	$\# R_6(F \rightarrow id)$	
$\emptyset E_1 + \text{id} T_9 \# \#$	$\# R_3(T \rightarrow F)$	
$\emptyset E_1 \#$	$\# R_1(E \rightarrow E_S)$	
$\emptyset E_1 \#$	$\# \text{ chaîne acceptée}$	

# 2/ l'analyse LR(1)

1) des items LR(1) <sup>unification</sup>

$$[A \rightarrow \alpha, \beta, @] \xrightarrow{\text{unif}} [A \rightarrow \alpha\beta]$$

est une production et a € T<sup>0+1</sup>

premier symbole du contexte droit

2) l'opération fermeture

fermeture ( $I_0$ )

$I_0$ : un ensemble d'items LR(1)

Si  $[A \rightarrow \alpha, \beta, @] \in I_0$  fermeture

( $I_0$ ) alors A gène l'item

$$[\beta \rightarrow \dots, b] \text{ où } \beta \rightarrow \gamma$$

est une production de Début ( $\beta_0$ )

3)  $\text{Goto}(I, x) = \text{fermeture}$

$$([A \rightarrow \alpha X, \beta, @])$$

$$\xrightarrow{\text{tg}}, [A \rightarrow \alpha X B, \beta, @] \in I$$

Expl

$$E \rightarrow E + T \mid T$$

$$T \rightarrow T * F \mid F$$

$$F \rightarrow (E) \mid i \mid \epsilon$$

$$I_0 = \{ [E \rightarrow \dots, E, @], [E \rightarrow E + t, @] \mid t \in T \}$$

TD

$$Ex0.1 \quad R_1 \quad | \quad R_2 \quad | \quad R_3$$

$$G: S \rightarrow i \cdot s e s | i s l a$$

$$I_0 = \{ [S \rightarrow \dots, S] | [S \rightarrow i \cdot s e s], [S \rightarrow \dots, C S] | [S \rightarrow \dots, a] \}$$

$$I_1 = \text{Goto}(I_0, S) = \{ [S \rightarrow S_0] \} \quad \begin{matrix} \text{ne pas} \\ \text{prendre} \end{matrix} \quad \begin{matrix} \text{line} \\ \text{colon} \end{matrix}$$

$$I_2 = \text{Goto}(I_0, i) = \{ [S \rightarrow \dots, S_{0,i}] | [S \rightarrow i \cdot S] | [S \rightarrow \dots, S_{0,i}] | [S \rightarrow \dots, a] \}$$

$$I_3 = \text{Goto}(I_0, a) = \{ [S \rightarrow a_0] \} \quad \begin{matrix} \text{ne pas} \\ \text{prendre} \end{matrix} \quad \begin{matrix} \text{line} \\ \text{colon} \end{matrix}$$

$$col = \text{SUV}(S) = \{ e, \# \}$$

$$I_4 = \text{Goto}(I_2, S) = \{ S \rightarrow i S_0, S \}, [S \rightarrow i S_0] \}$$

$$\text{Goto}(I_2, i) = \{ [S \rightarrow i \cdot S_{0,i}], [S \rightarrow \dots, i S_{0,i}] | [S \rightarrow \dots, i S_{0,i}] | [S \rightarrow \dots, a] \}$$

$$\text{Goto}(I_2, a) = \{ [S \rightarrow a_0] \} = I_3$$

$$\text{Goto}(I_4, e) = \{ [S \rightarrow i S_0, S] | [S \rightarrow \dots, i S_{0,i}] | [S \rightarrow \dots, i S_{0,i}] | [S \rightarrow a] \}$$

$$\text{Goto}[I_4, S] = \{ [S \rightarrow i S_0, S] \} = I_6$$

$$\text{Goto}(I_4, i) = I_2$$

$$\text{Goto}(I_4, a) = I_3$$

	SUV
S	e #

	i	e	a	#	s
0	D <sub>2</sub>		D <sub>3</sub>		
1				Accepte	(1)
2	D <sub>2</sub>		D <sub>3</sub>		(4)
3		R <sub>3</sub>		R <sub>3</sub>	
4		D <sub>5</sub>		R <sub>2</sub>	
5	D <sub>2</sub>		D <sub>3</sub>		(6)
6		R <sub>1</sub>		R <sub>1</sub>	

table multi déf donc

G n'est pas SLR(1)

لما داد جدول اكتمال انتقالات الـ LR(0) ،  
نلاحظ ان هناك انتقالات من نفس المدخل الى اسفل المدخل ،  
مثلاً من a الى a ، b ، c ، d ، e ، # .  
لذلك لا يمكن اكتفاء بـ SLR(1).

### 3/ Construction de la table LR(1).

expl  $S \rightarrow aAd \mid bBd \mid aBe \mid bAe$

A  $\rightarrow C \quad R_1$

B  $\rightarrow C \quad R_6$

I<sub>0</sub> = { [S'  $\rightarrow \cdot S, \#$ ] , [S  $\rightarrow \cdot aAd, \#$ ] ,

[S  $\rightarrow \cdot bBd, \#$ ] , [S  $\rightarrow \cdot aBe, \#$ ] ,

[S  $\rightarrow \cdot bAe, \#$ ] }

I = Goto(I<sub>0</sub>) : { [S'  $\rightarrow \cdot S_0, \#$ ] }

I<sub>1</sub> = Goto(I<sub>0</sub>, a) = { [S  $\rightarrow \cdot aAd, \#$ ] ,

[S  $\rightarrow a \cdot Be, \#$ ] , [A  $\rightarrow \cdot C, d$ ]

[B  $\rightarrow \cdot C, e$ ] }

I<sub>2</sub> = Goto(I<sub>0</sub>, b) = { [S  $\rightarrow \cdot bBd, \#$ ] ,

[S  $\rightarrow b \cdot Ae, \#$ ] , [B  $\rightarrow \cdot C, d$ ] , [A  $\rightarrow \cdot C, e$ ] }

$$I_0 = \text{Goto}(I_0, A) = \{ [S \rightarrow aA \cdot d, \#] \}$$

$$I_1 = \text{Goto}(I_0, B) = \{ [S \rightarrow bB \cdot e, \#] \}$$

$$I_2 = \text{Goto}(I_0, C) = \{ [A \rightarrow C \cdot d, \#], [B \rightarrow C \cdot e] \}$$

$$I_3 = \text{Goto}(I_0, B) = \{ [S \rightarrow bB \cdot d, \#] \}$$

$$I_4 = \text{Goto}(I_3, A) = \{ [S \rightarrow bA \cdot e, \#] \}$$

$$I_5 = \text{Goto}(I_3, C) = \{ [B \rightarrow C \cdot d, \#], [A \rightarrow C \cdot e] \}$$

$$\{ [A \rightarrow C \cdot e] \}$$

$$I_{10} = \text{Goto}(I_4, d) = \{ [S \rightarrow aAd, \#] \}$$

$$I_{11} = \text{Goto}(I_5, e) = \{ [S \rightarrow aBe, \#] \}$$

$$I_{12} = \text{Goto}(I_7, d) = \{ [S \rightarrow bBd, \#] \}$$

$$I_{13} = \text{Goto}(I_8, e) = \{ [S \rightarrow bAe, \#] \}$$

table d'analyse LR(1).

	a	b	c	d	e	#	B	A	B
0	D <sub>2</sub>	D <sub>3</sub>							(1)
1							Accepte		
2				D <sub>6</sub>					(4) (5)
3				D <sub>9</sub>					(8) (7)
4					D <sub>10</sub>				
5						D <sub>11</sub>			
6						R <sub>5</sub>	R <sub>6</sub>		
7						D <sub>12</sub>			
8						D <sub>13</sub>			
9						R <sub>6</sub>	R <sub>5</sub>		
10							R <sub>1</sub>		
11							R <sub>3</sub>		
12							R <sub>2</sub>		
13							R <sub>4</sub>		

table mono défini donc  $G$  est LR(1)

$$I_3 = \text{Goto}(I_5, B) = \{ A \rightarrow B \cdot \cdot \cdot \}$$

$$\text{Goto}(I_4, y) = I_4$$

$$I_5 = \text{Goto}(I_6, +) = \{ A \rightarrow B \cdot \cdot \cdot \}$$

$$\text{Goto}(I_7, +) = I_7$$

table LR(1)

	+	y	#	S	A	R
0	D4			①	②	③
1						
2	D5			R1		
3	D6					
4	R4					
5	D4				④	③
6	D8					
7	D5/R2			R2		
8	R3			R3		

donc  $G$  est TLR(1) car la table multiplié finie

exercice

$$S \rightarrow R$$

$$R \rightarrow R + R^* | R^* | \epsilon$$

1)  $G$  n'est pas LL(1)

Justification : on a récursivité gauche

2) les items LR(1)

$$I_0 = \{ (S \rightarrow \cdot S, \#), (S \rightarrow \cdot R, \#) \}$$

$$R \rightarrow \cdot R + R^*, \cdot, \cdot, \# \} , (R \rightarrow \cdot R^*, \cdot, \cdot, \#)$$

$$S \rightarrow A \quad R_1$$

$$A \rightarrow A + A \quad | \quad B^* \quad R_2$$

$$B \rightarrow y \quad R_3$$

$$I_0 = \{ [S \rightarrow \cdot S, \#], [S \rightarrow \cdot A, \#] \}$$

début (#)

$$[A \rightarrow \cdot A + A, \cdot \#], [A \rightarrow \cdot B^*, \cdot \#]$$

début (+A, #)

début (+B\*, #)

$$[B \rightarrow \cdot y, \cdot \#] \}$$

début (+y, #)

$$I_1 = \text{Goto}(I_0, S) = \{ [S \rightarrow S \cdot, \#] \}$$

$$I_2 = \text{Goto}(I_0, A) = \{ [S \rightarrow A \cdot, \#] \},$$

$$[A \rightarrow A \cdot + A, \cdot \#] \}$$

$$I_3 = \text{Goto}(I_0, B) = \{ [A \rightarrow B \cdot \cdot \cdot, \cdot \#] \}$$

$$I_4 = \text{Goto}(I_0, y) = \{ [B \rightarrow y \cdot, \cdot \#] \}$$

$$I_5 = \text{Goto}(I_2, +) = \{ [A \rightarrow A \cdot + A, \cdot \#] \},$$

$$[A \rightarrow \cdot A + A, \cdot \#], [A \rightarrow \cdot B^*, \cdot \#]$$

$$[B \rightarrow \cdot y, \cdot \#] \}$$

$$I_6 = \text{Goto}(I_3, +) = \{ [A \rightarrow B \cdot \cdot \cdot, \cdot \#] \}$$

$$I_7 = \text{Goto}(I_5, A) = \{ [A \rightarrow A \cdot + A, \cdot \#] \}$$

$$[A \rightarrow \cdot A + A, \cdot \#] \}$$

$(R \rightarrow \cdot, \#, *, +) \}$

$I_1 = \text{Goto}(I_0, S) = \{ (S \rightarrow S \cdot, \#) \}$

$I_2 = \text{Goto}(I_0, R) = \{ (S \rightarrow R \cdot, \#) \}$

$[R \rightarrow R \cdot + R^*, +, *, \#]$

$(R \rightarrow R \cdot, +, *, \#) \}$

## Chapitre 6: Traduction

Dirigée par la syntaxe

1<sup>e</sup> Langage intermédiaire

\* Notation Post fixée

Si E est une constante ou une variable Notation de E est E<sup>(3info 20)</sup>

Si E est de la forme E<sub>1</sub> OP E<sub>2</sub> (OP: opérateur binair) alors :

Notation de G est E<sub>1</sub>' E<sub>2</sub> OP

(E<sub>1</sub>', est la post fixée de E<sub>1</sub>, E<sub>2</sub>': la Postfixe fixée de E<sub>2</sub>)

Expl a+b → abt

2<sup>e</sup> Quadruplets est une structure à 4 champs :

(OP, S<sub>1</sub>, S<sub>2</sub>, dest)

dest ← S<sub>1</sub> OP S<sub>2</sub>

Exmpl a = b + (-c)

(-, c, 0, t<sub>1</sub>)

(\*, b, t<sub>1</sub>, t<sub>2</sub>)

(=, t<sub>2</sub>, , a)

3<sup>e</sup> triplet et structure à 3 champs (OP, Arg<sub>1</sub>, Arg<sub>2</sub>)

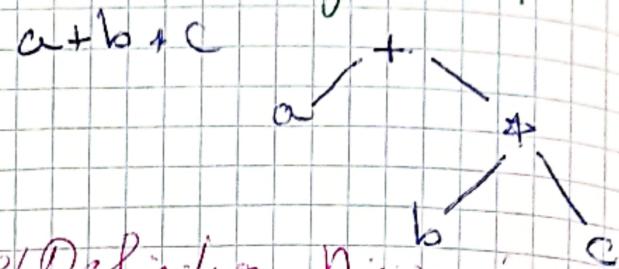
Expl a = b + (-c)

o (-, c, )

1 (+, b, (0))

2 (=, a, (1)')

4<sup>e</sup> Arbre Abstrait : forme complexe de l'arbre syntaxique.



2<sup>e</sup> Définition Dirigée par la syntaxe

à chaque symbole de la grammaire on associe un ensemble d'attribut et à chaque production un ensemble de règle sémantique pour calculer la valeur des attributs introduits en off symbol

\*) Clôture génération des attributs:

→ Attribut synthétisé : valeur calculée à partir des valeurs d'un symbole

→ Attribut hérités : valeur calculée à partir des descendants en frère du symbole

\*) Définition utilisant les attributs synthétisés.

On Notera X.Val l'attribut associé à un symbole X

$$E \rightarrow E + T$$

$$E \rightarrow T$$

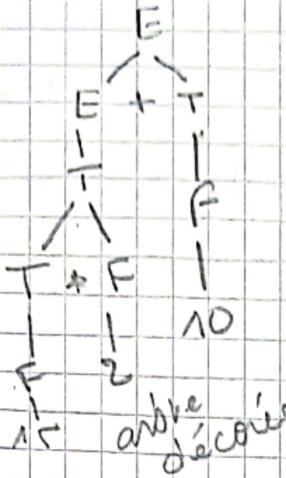
$$T \rightarrow T * F$$

$$T \rightarrow F$$

$$F \rightarrow (E)$$

$$F \rightarrow \text{mb}$$

$$15 * 2 + 10$$



Production

Règle Sémantique

$$E \rightarrow E + T$$

$$E.\text{Val} \leftarrow E.\text{Val} + T.\text{Val}$$

$$E \rightarrow T$$

$$E.\text{Val} \leftarrow T.\text{Val}$$

$$T \rightarrow T * F$$

$$T.\text{Val} \leftarrow T.\text{Val} * F.\text{Val}$$

$$T \rightarrow F$$

$$T.\text{Val} \leftarrow F.\text{Val}$$

$$F \rightarrow (E)$$

$$F.\text{Val} \leftarrow E.\text{Val}$$

$$F \rightarrow \text{mb}$$

$$F.\text{Val} \leftarrow \text{mb}$$

$$(+, E.\text{Val}, +. \text{Val}, A_1)$$

$$(=, A_1, , E.\text{Val})$$

$$(=, T.\text{Val}, , , E.\text{Val})$$

Définition utilisant des attributs  
très tés

Production

Règle Sémantique

$$D \rightarrow T L$$

$$L.\text{Type} \leftarrow T.\text{Type}$$

$$T \rightarrow \text{entier}$$

$$T.\text{Type} \leftarrow \text{entier}$$

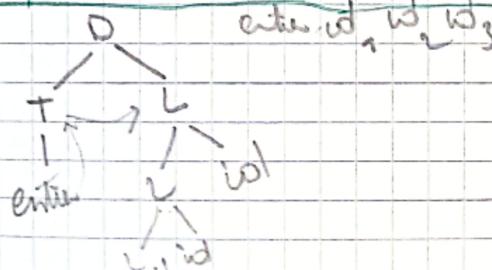
$$T \rightarrow \text{Réel}$$

$$T.\text{Type} \leftarrow \text{Réel}$$

$$L \rightarrow L_1, \text{id}$$

$$L_1.\text{Type} \leftarrow L.\text{Type}$$

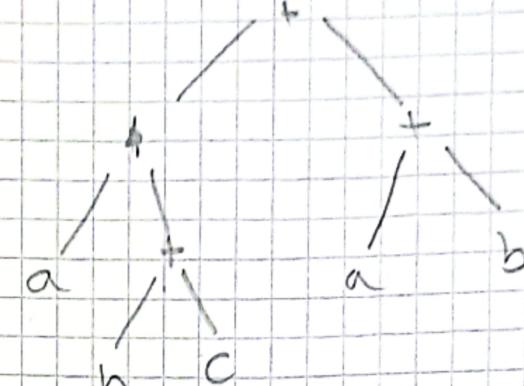
$$L \rightarrow \text{id}$$



TD 17/12/2023

EX01

Arbre abstrait



Notation post-fixée

$$abc + * ab + +$$

لبيان نوع يكون المتأتي

Quadruplet

$$S = a * (b + c) + (a + b)$$

$$(+, b, c, t_1)$$

$$(*, a, t_1, t_2)$$

$$(+, a, b, t_3)$$

$$(+, t_{a1}, t_3, t_u)$$

$$(=, t_u, , S)$$

Ex02

$$a = 3; b = 2j$$

while (a < 25)

$$\{ \text{if } (b < 6)$$

$$\{ a = a^4 2 + b; \}$$

else

$$\{ b = b * 2 + a; \} \}$$

Le code 3 adresses

(1)  $a = 3$

(2)  $b = 2$

(3) if ( $a < 25$ ) Goto (5)

(4) goto (15)

(5) if ( $b < 6$ ) Goto (7)

(6) goto (11)

(7)  $t_1 = a * 2$

(8)  $t_2 = t_1 + b$

(9)  $a = t_2$

(10) goto (14)

(11)  $t_3 = b * 2$

(12)  $t_4 = t_3 + a$

(13)  $b = t_4$

(14) goto (3)

(15)

while (( $a < 25$ ) and ( $b < 15$ ))

  if ( $a < 25$ ) goto (7)

    goto (fin)

    if ( $b < 15$ ) goto (7)

      goto (fin)

Ex 03

$S \rightarrow (S)S / \epsilon$

(( ))

$S \quad S.mV=0$

$S.mV=1$

$S.mV=2$

$S.mV=2$

$(S)S \quad S.mV=1$

$S.mV=1$

$S.mV=2$

$S.mV=2$

$S.mV=2$

$S.mV=2$

$S' \rightarrow S$	$S.mV = 0$
$S \rightarrow (S)S$	$S_1.mV \leftarrow S.mV + 1; S_2.mV \leftarrow S.mV$
$S \rightarrow \epsilon$	errire ( $S.mV$ )