Université Mohammed V - Agdal

Faculté des sciences

Département d'Informatique

Cours de Compilation

SMI - S5

Prof. M.D. RAHMANI

mrahmani@fsr.ac.ma

Chapitre II: Notions de la Grammaire

I- Définition de la syntaxe

II- Traduction dirigée par la syntaxe

Notion de la grammaire

Un langage de programmation peut être défini par:

1- la syntaxe du langage (la description de ses programmes).

2- La sémantique du langage (leur signification)

Pour spécifier la syntaxe d'un langage, nous utilisons *la grammaire hors contexte* (notation **BNF**)

I- Définition de la syntaxe:

1- La grammaire:

La grammaire décrit d'une manière hiérarchique les constructions du langage.

Elle est constituée d'une suite de règles.

Exemple 1: soit l'instruction conditionnelle en PASCAL:

« *if (exp) then instr else instr* »

Pour spécifier cette instruction, nous utilisons une règle structurelle appelée: **production**

`instr` ⟶ `if (exp) then instr else instr`

I- Définition de la syntaxe:

1/ `instr` ⟶ `if ( exp ) then instr else instr`
| `if (exp ) then instr`
| `autre`

2/ `exp` ⟶ `terme operel terme`
| `terme`

3/ `terme` ⟶ `id` | `nb`

Les terminaux sont: if , (,) , then, else , operel , id , nb

Les non-terminaux sont: instr, exp , terme

L'axiome est le non-terminal: instr

⟶ veut dire *produit* ou il *peut avoir la forme*

I- Définition de la syntaxe:

- Les éléments en *rouge* sont des **unités lexicales** ou **terminaux**.
- Les éléments définis à gauche des flèches sont des **suites d'*unités lexicales*** appelées des **non-terminaux**.
- Nous avons 3 règles de grammaire
- Chaque alternative dans **une règle** est appelée: **production**
- Le premier symbole non-terminal de la grammaire est appelé **axiome** .

Nous disons que nous avons défini un langage par une grammaire

I- Définition de la syntaxe:

Une grammaire hors contexte est constituée de:

- *un ensemble de* ***terminaux***,
- *un ensemble de* ***non-terminaux***,
- *un ensemble de* ***productions***,
- *un symbole de départ appelé* ***axiome***

I- Définition de la syntaxe:

Exemple 2:

Soit les expressions composées de chiffres et des signes ‘+’ et ‘-’.

La grammaire qui décrit la syntaxe de ce langage est donnée par:

1ère forme:

liste → liste + liste | liste – liste | chiffre

chiffre → 0 | 1 | 2 | …… | 9

2ème forme:

liste → chiffre + liste | chiffre – liste | chiffre

chiffre → 0 | 1 | 2 | …… | 9

3ème forme:

liste → liste + chiffre | liste – chiffre | chiffre

chiffre → 0 | 1 | 2 | …… | 9

Les terminaux sont: **+ , - , 0 , 1, …. , 9**

Les non-terminaux: **liste** et **chiffre**

L’axiome est **liste**

I- Définition de la syntaxe:

2- **Arbre syntaxique:**

Il décrit d'une manière hiérarchique les constructions d'un langage et possède 4 propriétés:

- la **racine** de l'arbre est l'**axiome** de la grammaire,
- chaque **feuille** est un **terminal** de la grammaire ,
- chaque **nœud** est un **non-terminal**,
- si A est le non-terminal d'un nœud intérieur et si les fils de ce nœud sont de gauche à droite X_1, X_2, …., X_n alors :

A $\longrightarrow$ X_1…. X_n est une **production** de la grammaire.

Les X_i sont des **terminaux** et des **non-terminaux**.

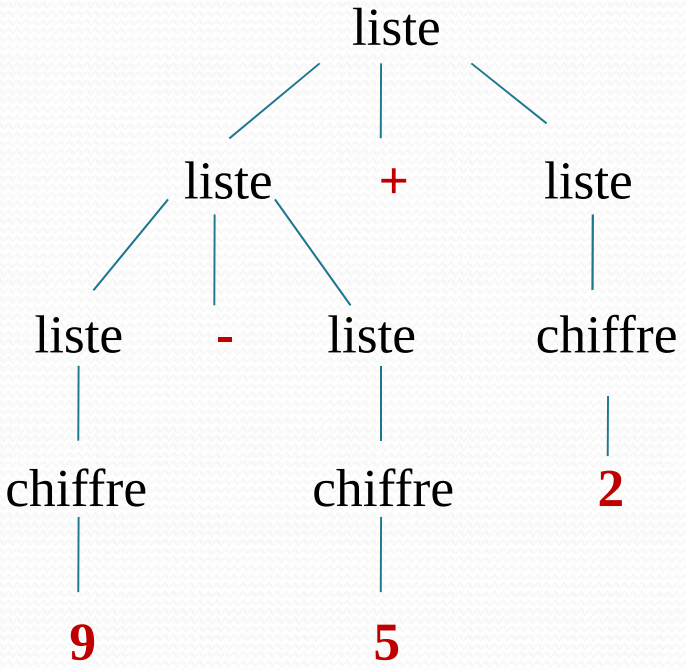
I- Définition de la syntaxe:

Exemple: L'arbre syntaxique de l'expression « **9 – 5 + 2** » d'après la grammaire précédente.

1ère forme:

liste ⟶ liste + liste | liste – liste | chiffre

chiffre ⟶ 0 | 1 | 2 …… | 9



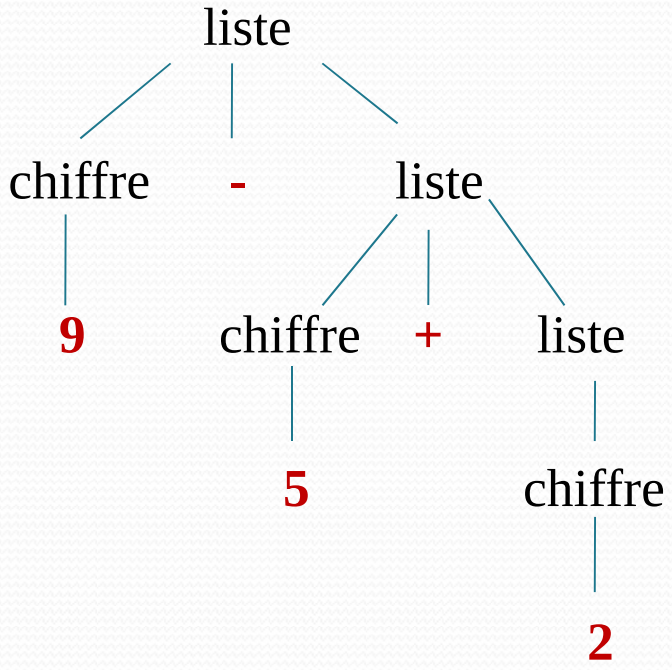
I- Définition de la syntaxe:

Exemple: L'arbre syntaxique de l'expression « **9 – 5 + 2** » d'après la grammaire précédente.

2 ème forme:

liste ⟶ chiffre + liste | chiffre – liste | chiffre

chiffre ⟶ 0 | 1 | 2 …… | 9



Cette grammaire est **récursive à droite**

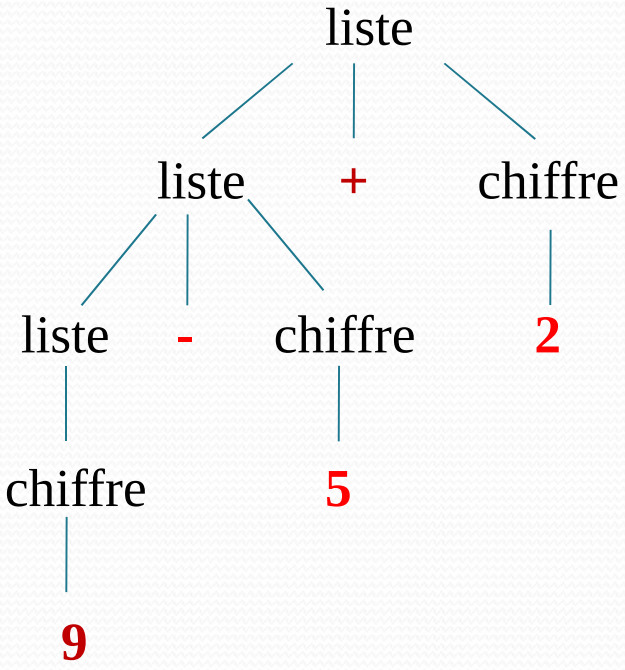
I- Définition de la syntaxe:

Exemple: L'arbre syntaxique de l'expression « **9 – 5 + 2** » d'après la grammaire précédente.

3 ème forme:

liste ⟶ liste + chiffre | liste – chiffre | chiffre

chiffre ⟶ 0 | 1 | 2 …… | 9



Cette grammaire est **récursive à gauche**

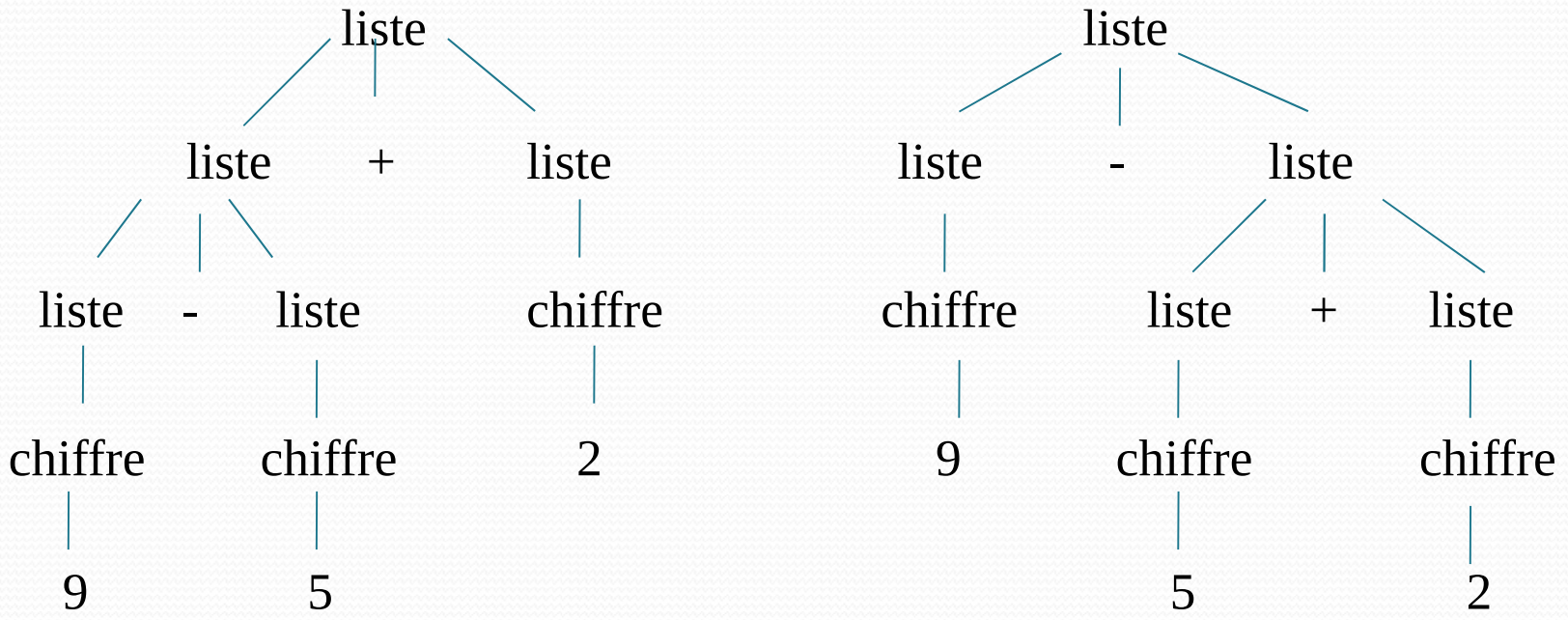
I- Définition de la syntaxe:

3- Ambiguïté d'une grammaire:

*Une grammaire est dite **ambiguë** si elle peut donner **plus** d'un arbre syntaxique pour la **même** phrase.*

Exemple: « `9-5+2` »

1ère forme:



Nous devons éviter l'utilisation d'une grammaire ambigüe, donc la 1ère forme.

Question: quel est le choix, de la grammaire, le plus approprié (la 2ème forme ou la 3ème forme)?

I- Définition de la syntaxe:

4- Associativité des opérateurs:

La 2ème forme favorise le + par rapport au - : ***9-5+2*** sera interprétée comme ***9-(5+2)***= 9-7=2

La 3ème forme favorise le – par rapport au + : ***9-5+2*** sera interprétée comme ***(9-5)+2***=4+2=6

Par convention: nous considérons l'interprétation de la 3ème forme.

Définition:

*Soit un opérande entouré de 2 opérateurs, s'il est traité en priorité par celui qui est à **sa gauche** (à **sa droite**), on dit que cet **opérateur est associatif à gauche** (à **droite**).*

Convention:

Dans la plus part des langages de programmation, **les 4 opérateurs arithmétiques « +,-,*,/ » sont associatifs à gauche**.

I- Définition de la syntaxe:

4- Associativité des opérateurs:

- L'opérateur d'affectation est *associatif à droite*:

a = b =c est interprétée comme a=(b=c)

- Les opérateurs de relation *ne sont pas associatifs*:

en effet, les langages de programmation n'autorisent pas:

if (a < b < c)

Pour le cas du langage C, par exemple il faut écrire:

if (a<b && b<c)

I- Définition de la syntaxe:

5- Priorité des opérateurs:

Soit l'expression: ***9 + 5 * 2***

L'***associativité*** ne résout pas le problème !

L'interprétation sera :

(9 + 5) * 2

ou 9 + (5 * 2) !

- Par convention, on favorise la 2ème forme.

Nous considérons par convention que les opérateurs ***** *et* / *sont prioritaires par rapport à* + *et* -

Remarques:

- Pour deux opérateurs arithmétiques qui ont le **même niveau de priorité**, on favorise l'**associativité gauche**.

- S'ils **n'ont pas la même priorité**, on favorise la **priorité**.

I- Définition de la syntaxe:

Question:

Construire une grammaire pour les expressions arithmétiques constituées des nombres, des expressions parenthesées et des 4 opérateurs arithmétiques +, -, * et / en tenant compte de l'**associativité gauche** des **opérateurs** et de la **priorité**.

Réponse:

Nous définissons 3 non terminaux pour les 3 niveaux de priorités: E, T et F

La grammaire est:

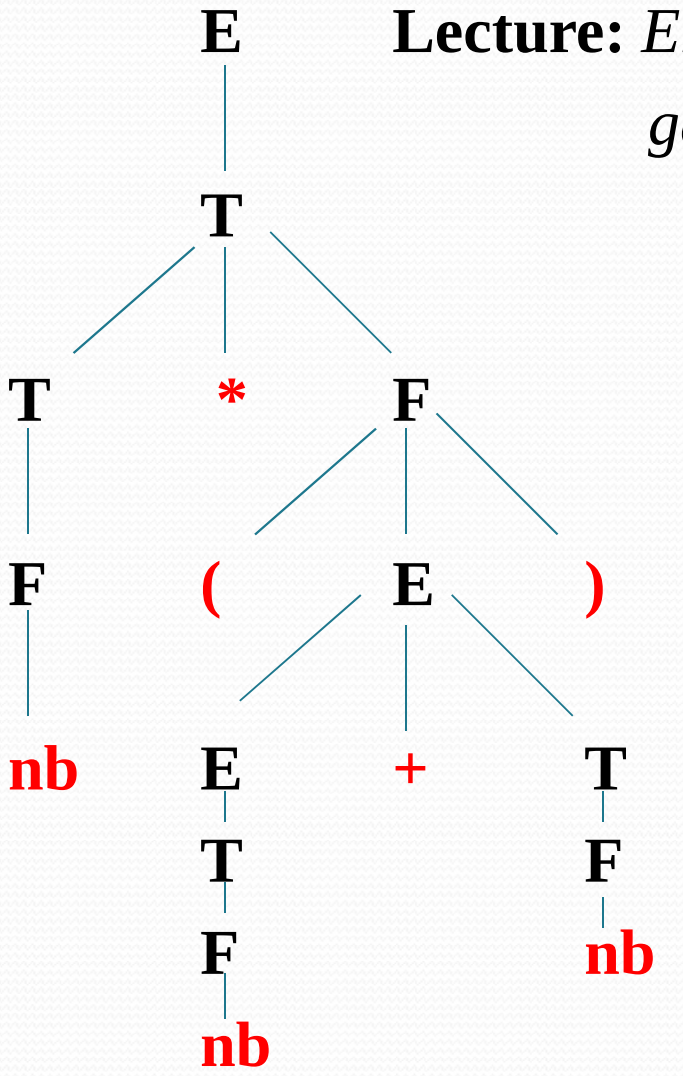
E ⟶ E + T | E - T | T

T ⟶ T * F | T / F | F

F ⟶ nb | (E)

I- Définition de la syntaxe:

Exemple: arbre syntaxique de **nb * (nb + nb)**



Lecture: *En profondeur d'abord, gauche-droite*

II- La traduction dirigée par la syntaxe (TDS):

1- **Définition:**

C'est la traduction d'un langage guidée par une grammaire hors contexte à laquelle on associe des règles sémantiques.

Il y'a 2 notations pour introduire les règles sémantiques qui donnent un sens aux constructions d'un langage:

- La **définition dirigée par la syntaxe** (DDS), où les règles sémantiques calculent les valeurs des attributs associés aux symboles de la grammaire.

- Le **schéma de traduction**, où les règles sémantiques prennent la forme d'actions insérées dans la partie droite des productions de la grammaire.

II- La traduction dirigée par la syntaxe (TDS):

2- **Notation post fixée:**

Soit ***E*** une expression:

- si ***E*** est une variable ou une constante, la notation post fixée de **E** est elle-même.

- si ***E*** est une expression de la forme infixée ***E1 op E2***, sa notation post fixée est ***Ep1 Ep2 op***, avec ***Ep1*** et ***Ep2*** les notations post fixées de ***E1*** et ***E2***.

- si ***E*** est une expression de la forme ***(E1)*** , sa notation post fixée est celle de ***E1***.

Exemple: notation infixée → notation post fixée

9 – 2	*9 2 –*
(9 – 5) + 2	*9 5 – 2 +*
9 – (5 + 2)	*9 5 2 + –*

II- La traduction dirigée par la syntaxe (TDS):

3- **La définition dirigée par la syntaxe (DDS):**

Une DDS d'un langage de programmation est constitué par:

- la grammaire qui spécifie la syntaxe du texte d'entrée.

*- les règles sémantiques qui **calculent les valeurs des attributs** associés aux symboles d'une construction du langage.*

Exemples d'attributs:

type, adresse mémoire, portée d'une variable....

II- La traduction dirigée par la syntaxe (TDS):

Exemple 1: traduction d'une expression arithmétique d'une ***notation infixée*** en ***notation post fixée***.

Soit la grammaire:

E ⟶ E + T | E – T | T

T ⟶ F

F ⟶ c

c ⟶ 0

c ⟶ 1

. ⟶ .

. .

c ⟶ 9

II- La traduction dirigée par la syntaxe (TDS):

À la production: E ⟶ E1 + T , on associe une règle sémantique:

E.t = E1.t || T.t || '+' || *veut dire concaténé.*

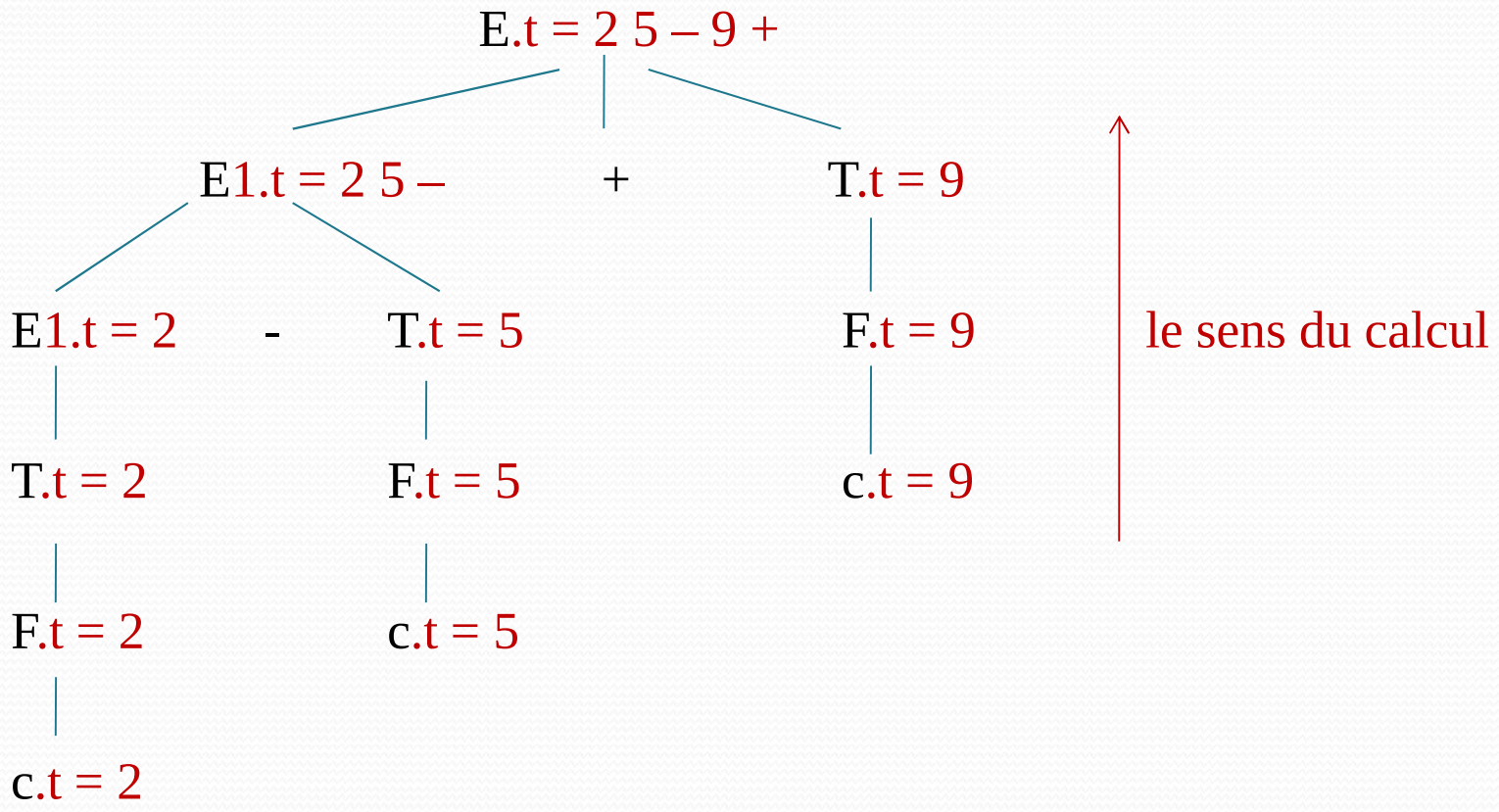
La DDS: la grammaire et les règles sémantiques

la grammaire			les règles sémantiques
E	⟶	E1 + T	{E.t = E1.t || T.t || '+' }
E	⟶	E1 - T	{E.t = E1.t || T.t || '–' }
E	⟶	T	{E.t = T.t }
T	⟶	F	{T.t = F.t}
F	⟶	c	{F.t = c.t}
c	⟶	0	{c.t = '0'}
.....			.....
c	⟶	9	{c.t = '9'}

II- La traduction dirigée par la syntaxe (TDS):

La notation post fixée de « 2-5+9 »

L'arbre syntaxique décoré



II- La traduction dirigée par la syntaxe (TDS):

Exemple 2:

Soit un robot commandé pour se déplacer d'une suite de pas dans l'espace à 2 dimensions.

1- La grammaire qui engendre une séquence de pas:

seq ⟶ seq pas | début

pas ⟶ nord | sud | est | ouest

2- D.D.S qui calcule la position finale du robot à partir d'une position initiale et d'une séquence de pas.

La grammaire			**+ Les règles sémantiques**
seq	⟶	seq1 pas	{seq.x = seq1.x + pas.Δx , seq.y = seq1.y + pas.Δy }
seq	⟶	début	{seq.x = début.x = 0 , seq.y = début.y = 0}
pas	⟶	nord	{pas.Δx = nord.Δx = 0 , pas.Δy = nord.Δy = 1}
pas	⟶	sud	{pas.Δx = sud.Δx = 0 , pas.Δy = sud.Δy = -1}
pas	⟶	est	{pas.Δx = est.Δx = 1 , pas.Δy = est.Δy = 0 }
pas	⟶	ouest	{pas.Δx = ouest.Δx = -1, pas.Δy = ouest.Δy = 0 }

II- La traduction dirigée par la syntaxe (TDS):

3- Application de cette DDS à la séquence de pas: « début ouest sud »

II- La traduction dirigée par la syntaxe (TDS):

4- <u>Le schéma de traduction</u>:

C'est une grammaire dans la quelle on insère les règles sémantiques, appelées actions, dans la partie droite des productions.

<u>Exemple</u>: Un schéma de traduction qui traduit une expression infixée en une notation post fixée.

II- La traduction dirigée par la syntaxe (TDS):

Exemple: Traduction des expressions arithmétiques d'une notation infixée en notation post fixée.

Le schéma de traduction:

E	⟶	E + T	{imprime '+'}
E	⟶	E – T	{imprime'-'}
E	⟶	T	{ }
T	⟶	F	{ }
F	⟶	c	
c	⟶	0	{imprime'0'}
c	⟶	1	{imprime'1'}
c	⟶	.	
.		.	
c	⟶	9	{imprime'9'}

II- La traduction dirigée par la syntaxe (TDS):

L'arbre syntaxique décoré de **« 2-5+9 »**

Lecture des feuilles rouges en profondeur d'abord, gauche-droite.