

# Grammaires et Analyse Syntaxique - Cours 3

## Grammaires algébriques

Ralf Treinen



Université  
Paris Cité



INSTITUT  
DE RECHERCHE  
EN INFORMATIQUE  
FONDAMENTALE

`treinen@irif.fr`

6 février 2025

## Introduction : Grammaires

- ▶ Vu la semaine dernière : analyse lexicale (ocamllex)
- ▶ Aujourd'hui : formalisme utilisé pour l'étape d'analyse grammaticale qui suit l'analyse lexicale : les *grammaires*.
- ▶ Dans les semaines à venir nous allons étudier la mise en œuvre de l'analyse grammaticale.

## Langages non-reconnaissables

- ▶ On a vu dans le cours AAL3 du L2 deux méthodes pour montrer qu'un langage donné n'est pas reconnaissable (c-à-d ne peut pas être défini par une expression rationnelle) :
  - ▶ Le lemme de l'étoile (ou lemme d'itération, *pumping lemma*)
  - ▶ Le théorème de Myhill-Nerode
- ▶ Intuitivement : les langages qui nécessitent une mémoire non bornée, ne sont pas reconnaissables.
- ▶ Exemple :  $\{a^n b^n \mid n \geq 0\}$  n'est pas reconnaissable (vu dans le cours AAL3).
- ▶ On verra aujourd'hui que ce langage peut être décrit par une grammaire.

## Définition des grammaires algébriques

Une *grammaire algébrique* (ou *grammaire hors contexte*) est un tuple  $G = (\Sigma, N, S, P)$  où

- ▶  $\Sigma$  est un ensemble fini de symboles, appelé les *symboles terminaux* de  $G$  ;
- ▶  $N$  est un ensemble fini et disjoint de  $\Sigma$  de symboles, appelé les *symboles non-terminaux* de  $G$  ;
- ▶  $S \in N$ , appelé l'*axiome* de  $G$  ;
- ▶  $P$  est un ensemble fini de *règles de production* de la forme  $A \rightarrow u$ , où  $A \in N$ , et  $u \in (\Sigma \cup N)^*$ .

En anglais : *context free grammar (CFG)*.

## Premier exemple d'une grammaire algébrique

$G_1 = (\Sigma, N, S, P)$  où

- ▶  $\Sigma = \{a, b\}$
- ▶  $N = \{A\}$
- ▶  $S = A$
- ▶  $P$  consiste en les règles suivantes :

$$A \rightarrow aAb$$

$$A \rightarrow \epsilon$$

- ▶ On verra que cette grammaire définit le langage  $\{a^n b^n \mid n \geq 0\}$ .

## Notation : alternatives

- ▶ Une grammaire peut avoir plusieurs règles avec le même côté gauche :

$$E \rightarrow E+E$$

$$E \rightarrow E * E$$

- ▶ Nous permettons dans la suite dans ce cas d'écrire une seule règle, avec plusieurs *alternatives* sur le côté droit :

$$E \rightarrow E+E \mid E * E$$

## Exemple : Dérivation de $aaabbb$ dans $G_1$

La grammaire  $G_1$  donnée au-dessus a les deux productions :

$$A \rightarrow aAb \mid \epsilon$$

Elle permet par exemple la dérivation suivante :

$$\begin{aligned} A &\rightarrow aAb \\ &\rightarrow aaAbb \\ &\rightarrow aaaAbbb \\ &\rightarrow aaabbb \end{aligned}$$

## Dérivation en une étape

### Définition

Soit une grammaire  $G = (\Sigma, N, S, P)$ , et  $u, v \in (N \cup \Sigma)^*$ .

On dit que  $G$  *permet de dériver  $v$  à partir de  $u$  en une étape*, noté  $u \rightarrow_G v$  (ou abrégé  $u \rightarrow v$ ), si

1.  $u = w_1 A w_2$ , où  $w_1, w_2 \in (N \cup \Sigma)^*$ ,  $A \in N$ ;
2.  $A \rightarrow w$  est une règle de  $P$ ;
3.  $v = w_1 w w_2$ .

## Dérivation en plusieurs étapes

### Définition

Soit une grammaire  $G = (\Sigma, N, S, P)$ , et  $u, v \in (N \cup \Sigma)^*$ .

On dit que  $G$  *permet de dériver  $v$  à partir de  $u$  en plusieurs étapes*, noté  $u \rightarrow_G^* v$  (ou abrégé  $u \rightarrow^* v$ ), s'il existe une suite finie  $w_0, w_1, \dots, w_n$  de mots de  $(N \cup \Sigma)^*$  telle que

$$u = w_0 \rightarrow_G w_1 \rightarrow_G \dots \rightarrow_G w_n = v$$

### Exemple

- ▶  $A \rightarrow_{G_1}^* aaabbb$
- ▶  $A \rightarrow_{G_1}^* aaaaaabbbbbbb$
- ▶  $A \rightarrow_{G_1}^* \epsilon$

## Langage engendré

### Définition

Soit  $G = (\Sigma, N, S, P)$  une grammaire algébrique. Le *langage engendré* par  $G$  est

$$\mathcal{L}(G) = \{w \in \Sigma^* \mid S \rightarrow_G^* w\}$$

Un langage  $L$  est *algébrique* (ou *hors-contexte*, en anglais *context free*) s'il existe une grammaire algébrique  $G$  telle que  $L = \mathcal{L}(G)$ .

### Exemple

$$\mathcal{L}(G_1) = \{a^n b^n \mid n \geq 0\}$$

## Exemple plus intéressant d'une grammaire algébrique

$G_2 = (\Sigma, N, S, P)$  où

- ▶  $\Sigma = \{i, v, +, *, (, )\}$
- ▶  $N = \{E, C\}$
- ▶  $S = E$
- ▶  $P$  consiste en les règles suivantes :

$E \rightarrow E+E$

$E \rightarrow E * E$

$E \rightarrow (E)$

$E \rightarrow C$

$C \rightarrow i$

$C \rightarrow v$

Exemple : Dérivation de  $i+v*i$  dans  $G_2$ 

Productions de  $G_2$  :  $E \rightarrow E+E \mid E * E \mid (E) \mid C$        $C \rightarrow i \mid v$

Ici : en rouge le non-terminal qui est réécrit.

$$\begin{aligned} E &\rightarrow E+E \\ &\rightarrow E+E * E \\ &\rightarrow C+E * E \\ &\rightarrow C+E * C \\ &\rightarrow C+E * i \\ &\rightarrow i+E * i \\ &\rightarrow i+C * i \\ &\rightarrow i+v * i \end{aligned}$$

## Langage engendré

### Définition (rappel)

Soit  $G = (\Sigma, N, S, P)$  une grammaire algébrique. Le *langage engendré* par  $G$  est

$$\mathcal{L}(G) = \{w \in \Sigma^* \mid S \rightarrow^* w\}$$

### Sur l'exemple $G_2$

$\mathcal{L}(G_2)$  est l'ensemble des expressions arithmétiques formées à l'aide des opérateurs  $+$  et  $*$ , des parenthèses, et des constantes  $i$  et  $v$ . Ce langage n'est pas régulier (ça se montre facilement avec le théorème de Myhill-Nerode).

## Choix dans les constructions de dérivations

- ▶ Nous avons vu dans cet exemple (le premier exemple était trop simple) qu'on a en général à chaque étape d'une dérivation deux décisions à prendre.
- ▶ Si on a déjà dérivé un mot de terminaux et non-terminaux  $\alpha$ , alors pour faire l'étape suivante il faut choisir :
  1. une occurrence d'un non-terminal dans  $\alpha$  (en général il y en a plusieurs) ;
  2. puis pour ce non-terminal une règle de la grammaire (en général il y en a plusieurs).

## Dérivation gauche

Soit une grammaire  $G = (\Sigma, N, S, P)$ .

### Réécriture à gauche

Soient  $u, v \in (N \cup \Sigma)^*$ .  $G$  permet de dériver  $u$  à partir de  $v$  en une *étape à gauche*, noté  $u \xrightarrow{g} v$ , si

1.  $u = w_1Aw_2$ , où  $w_1 \in \Sigma^*$ ,  $A \in N$  et  $w_2 \in (N \cup \Sigma)^*$  ;
2.  $A \rightarrow w$  est une règle de  $P$  ;
3.  $v = w_1ww_2$ .

En anglais : *leftmost derivation*.

### Dérivation gauche

Une *dérivation gauche* est une suite finie  $w_0, w_1, \dots, w_n$  de mots de  $V^*$  telle que  $w_i \xrightarrow{g} w_{i+1}$  pour tout  $i \in [0, n - 1]$ .

Exemple : Dérivation gauche de  $i+v*i$  dans  $G_2$ Productions de  $G_2$  :  $E \rightarrow E+E \mid E * E \mid (E) \mid C$      $C \rightarrow i \mid v$ 

$$\begin{aligned}
 E &\xrightarrow{\sigma} E * E \\
 &\xrightarrow{\sigma} E + E * E \\
 &\xrightarrow{\sigma} C + E * E \\
 &\xrightarrow{\sigma} i + E * E \\
 &\xrightarrow{\sigma} i + C * E \\
 &\xrightarrow{\sigma} i + v * E \\
 &\xrightarrow{\sigma} i + v * C \\
 &\xrightarrow{\sigma} i + v * i
 \end{aligned}$$

## Pareil : Dérivation droite

Soit une grammaire  $G = (\Sigma, N, S, P)$ .

### Réécriture à droite

Soient  $u, v \in (N \cup \Sigma)^*$ .  $G$  permet de dériver  $u$  à partir de  $v$  en une *étape à droite*, noté  $u \xrightarrow{d} v$ , si

1.  $u = w_1 A w_2$ , où  $w_1 \in (N \cup \Sigma)^*$ ,  $A \in N$  et  $w_2 \in \Sigma^*$  ;
2.  $A \rightarrow w$  est une règle de  $P$  ;
3.  $v = w_1 w w_2$ .

### Dérivation droite

Une *dérivation droite* est une suite finie  $w_0, w_1, \dots, w_n$  de mots de  $V^*$  telle que  $w_i \xrightarrow{d} w_{i+1}$  pour tout  $i \in [0, n - 1]$ .

Exemple : Dérivation droite de  $i+v*i$  dans  $G_2$ Productions de  $G_2$  :  $E \rightarrow E+E \mid E * E \mid (E) \mid C \quad C \rightarrow i \mid v$ 

$$\begin{aligned}
 E &\xrightarrow{d} E+E \\
 &\xrightarrow{d} E+E * E \\
 &\xrightarrow{d} E+E * C \\
 &\xrightarrow{d} E+E * i \\
 &\xrightarrow{d} E+C * i \\
 &\xrightarrow{d} E+v * i \\
 &\xrightarrow{d} C+v * i \\
 &\xrightarrow{d} i+v * i
 \end{aligned}$$

## Donc faut-il une dérivation gauche ou droite ou quoi ?

- ▶ Dans une dérivation d'une grammaire algébrique, on remplace toujours un non-terminal sans se soucier de ce qu'il y a autour.
- ▶ C'est la raison pour laquelle ces grammaires sont appelées *hors contexte*.
- ▶ Donc, si on a

$$S \rightarrow^* x_1 N_1 x_2 N_2 x_3 \rightarrow x_1 u_1 x_2 N_2 x_3 \rightarrow x_1 u_1 x_2 u_2 x_3$$

- ▶ alors on peut réarranger l'ordre des réécritures en

$$S \rightarrow^* x_1 N_1 x_2 N_2 x_3 \rightarrow x_1 N_1 x_2 u_2 x_3 \rightarrow x_1 u_1 x_2 u_2 x_3$$

## Donc faut-il une dérivation gauche ou droite ou quoi ?

- ▶ Pour cette raison, quand il y a une dérivation  $w$  à partir de  $S$  alors on peut la réarranger en une dérivation gauche, et aussi en une dérivation droite.
- ▶ Preuve exacte omise.
- ▶ Conséquence : pour tout mot  $w$ ,

$$S \rightarrow^* w \quad \text{ssi} \quad S \xrightarrow{g}^* w \quad \text{ssi} \quad S \xrightarrow{d}^* w$$

- ▶ Donc : nous sommes libres de fixer une stratégie (gauche ou droite, par exemple) qui nous convient.

## Arbre de dérivation

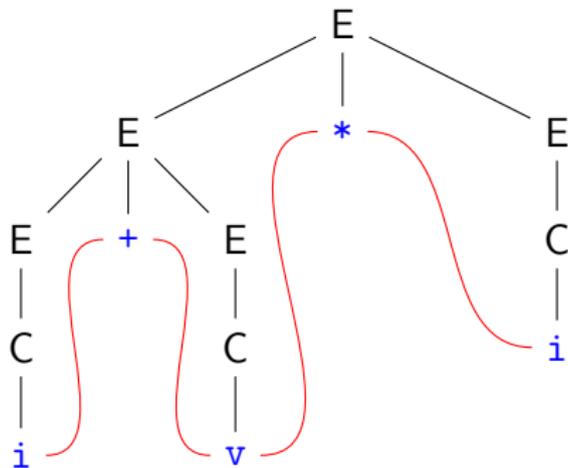
### Définition

Soit  $G = (\Sigma, N, S, P)$  une grammaire. Un *arbre de dérivation* de  $G$  est un arbre tel que

- ▶ la racine est étiquetée par l'axiome  $S$  ;
- ▶ tout nœud interne est étiqueté par un symbole de  $N$  ;
- ▶ toute feuille est étiquetée par un symbole de  $\Sigma \cup \{\epsilon\}$  ;
- ▶ si les fils pris de gauche à droite d'un nœud interne étiqueté par le non-terminal  $A$  sont étiquetés par les symboles respectifs  $\alpha_1, \dots, \alpha_n$ , alors  $(A \rightarrow \alpha_1 \dots \alpha_n) \in P$ .

### Définition

Un arbre de dérivation dont le mot des feuilles est  $u$  est appelé *arbre de dérivation de  $u$* .

Exemple : Un arbre de dérivation de  $i+v*i$  dans  $G_2$ Productions de  $G_2$  :  $E \rightarrow E+E \mid E * E \mid (E) \mid C$      $C \rightarrow i \mid v$ 

## Dérivation vers arbre de dérivation

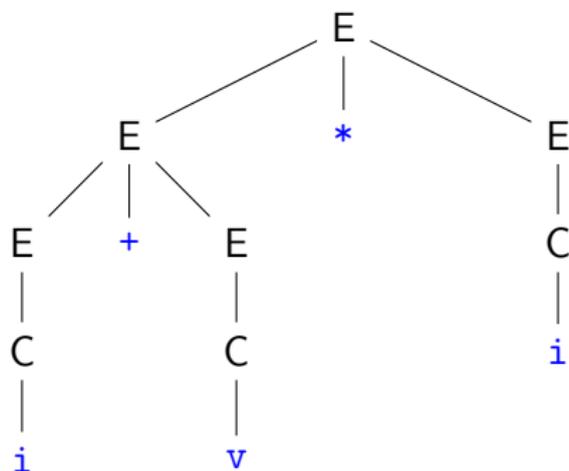
- ▶ On définit, pour une dérivation  $S \rightarrow \alpha_1 \rightarrow \dots \rightarrow \alpha_n$ , où  $\forall i : \alpha_i \in (\Sigma \cup N)^*$ , un arbre de dérivation pour  $\alpha_n$ .
- ▶ Par induction sur  $n$  :
  - ▶ Si  $n = 0$  : l'arbre consiste seulement dans le nœud  $S$ .
  - ▶ De  $n$  à  $n + 1$  : Soient

$$\begin{aligned}\alpha_n &= X_1 \dots X_{k-1} X_k X_{k+1} \dots X_m \\ \alpha_{n+1} &= X_1 \dots X_{k-1} Y_1 \dots Y_l X_{k+1} \dots X_m\end{aligned}$$

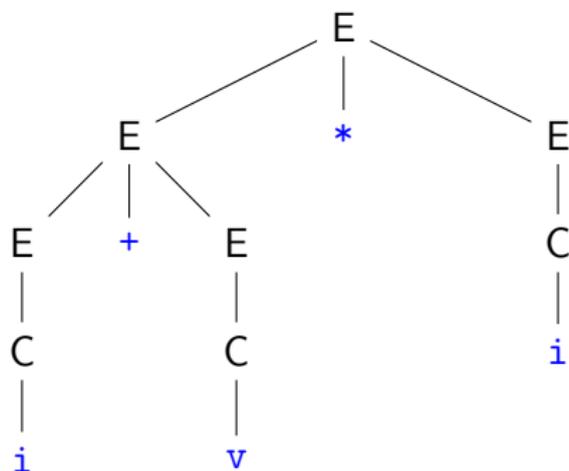
Par hypothèse d'induction, il existe un arbre de dérivation  $t$  pour  $S \rightarrow^* \alpha_n$ .

On ajoute dans  $t$  à la feuille  $X_k$  les enfants  $Y_1 \dots Y_l$ .

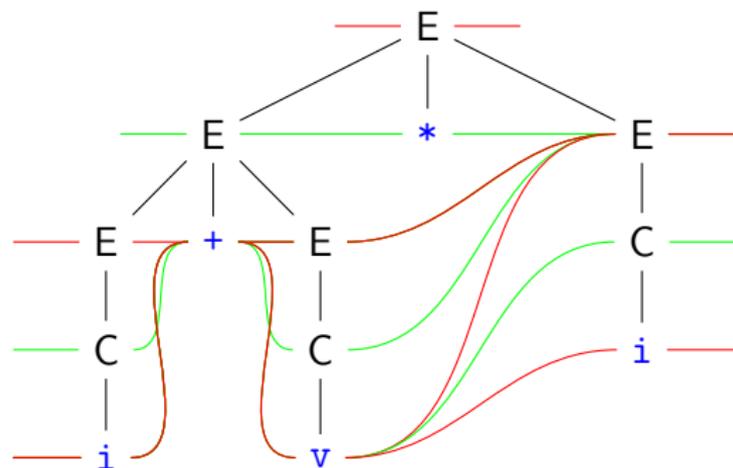
## Exemple, à partir d'une dérivation gauche

$$E \rightarrow E * E \rightarrow E + E * E \rightarrow C + E * E \rightarrow i + E * E \rightarrow i + C * E \rightarrow i + v * E \rightarrow i + v * C \rightarrow i + v * i$$


## Exemple, à partir d'une dérivation droite

$$E \rightarrow E * E \rightarrow E * C \rightarrow E * i \rightarrow E + E * i \rightarrow E + C * i \rightarrow E + v * i \rightarrow C + v * i \rightarrow i + v * i$$


## Arbre de dérivation vers dérivation

Exemple : Un arbre de dérivation de  $i+v*i$  dans  $G_2$ 

$$E \rightarrow E * E \rightarrow E + E * E \rightarrow C + E * E \rightarrow i + E * E \rightarrow i + C * E \rightarrow i + v * E \rightarrow i + v * C \rightarrow i + v * i$$

## Arbres de dérivation et Dérivation gauche

- ▶ Pour chaque arbre de dérivation il y a une unique dérivation gauche.
- ▶ Pour chaque arbre de dérivation il y a une unique dérivation droite.
- ▶ Pour chaque arbre de dérivation il y a une unique dérivation pour n'importe quelle stratégie qu'on peut imaginer.
- ▶ Pour chaque dérivation (gauche, droite, n'importe) il y a un unique arbre de dérivation.
- ▶ Une dérivation contient l'information sur la structure de l'arbre, plus l'information dans quel ordre il est construit.

## Grammaires ambiguës

### Définition

Une grammaire  $G$  est non-ambiguë quand tout  $w \in \mathcal{L}(G)$  a un seul arbre de dérivation.

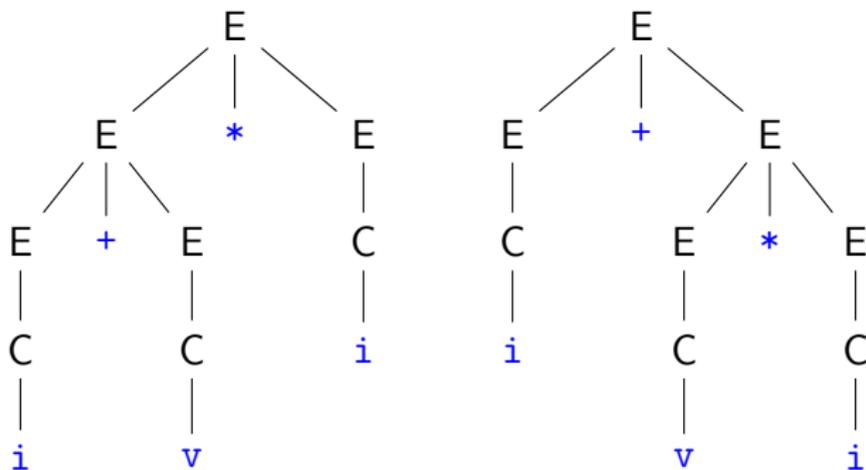
### Définition équivalente

Une grammaire  $G$  est non-ambiguë quand tout  $w \in \mathcal{L}(G)$  a une seule dérivation gauche.

### Sur l'exemple $G_2$

Rappel :  $E \rightarrow E+E \mid E * E \mid (E) \mid C \quad C \rightarrow i \mid v$

La grammaire  $G_2$  est ambiguë : le mot  $i+v*i$  a deux arbres de dérivation différents !

Deux arbres de dérivation de  $i+v*i$  dans  $G_2$ Productions de  $G_2$  :  $E \rightarrow E+E \mid E * E \mid (E) \mid C$      $C \rightarrow i \mid v$ 

## Un autre exemple

- ▶ La grammaire  $G_3 = (\{E\}, \{i, v, +, *, (, )\}, E, P)$ , où  $P$  est

$$E \rightarrow i \mid v \mid (E+E) \mid (E * E)$$

- ▶ Cette grammaire décrit les expressions arithmétiques complètement parenthésées.
- ▶ Ici il y a un seul terminal  $i$  pour les entiers, et un seul terminal  $v$  pour les noms des variables car on imagine qu'il s'agit des jetons issus d'une analyse lexicale.
- ▶ Cette grammaire est non-ambiguë (on verra dans quelques semaines pourquoi)

## Traduction d'un automate en grammaire

- ▶ Soit  $(\Sigma, Q, F, I, \delta)$  un automate non-déterministe.
- ▶ Grammaire :  $(\Sigma, \{N_q \mid q \in Q\} \cup \{S_0\}, S_0, R)$  avec  $R$  comme suit :

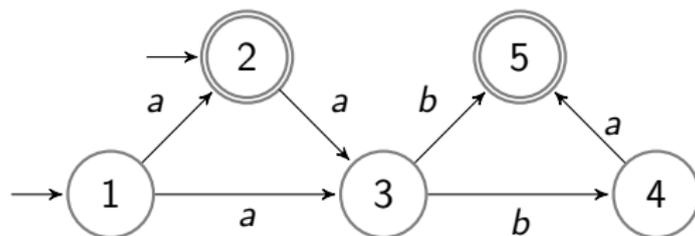
$$\begin{aligned} S_0 &\rightarrow N_q & q \in I \\ N_q &\rightarrow a N_p & p \in \delta(q, a) \\ N_q &\rightarrow \epsilon & q \in F \end{aligned}$$

- ▶ C'est une grammaire *linéaire droite* : toutes les productions sont d'une des deux formes :

$$\begin{aligned} N &\rightarrow w M \\ N &\rightarrow w \end{aligned}$$

avec  $N, M$  non terminaux,  $w \in \Sigma^*$ .

## Exemple Automate vers Grammaire



$$S_0 \rightarrow N_1$$

$$N_1 \rightarrow a N_2$$

$$N_2 \rightarrow a N_3$$

$$N_3 \rightarrow b N_5$$

$$N_2 \rightarrow \epsilon$$

$$S_0 \rightarrow N_2$$

$$N_1 \rightarrow a N_3$$

$$N_3 \rightarrow b N_4$$

$$N_4 \rightarrow a N_5$$

$$N_5 \rightarrow \epsilon$$

## Traduction d'une expression rationnelle en grammaire

- ▶ Étant donnée une expression rationnelle  $r$  sur alphabet  $\Sigma$ .
- ▶ Grammaire :  $(\Sigma, \{N_e \mid e \text{ sous-expression de } r\}, N_r, R)$  avec  $R$  comme suit :
  - ▶ Si  $e = a$  :  $N_e \rightarrow a$
  - ▶ Si  $e = \epsilon$  :  $N_e \rightarrow \epsilon$
  - ▶ Si  $e = \emptyset$  : aucune règle pour  $N_e$
  - ▶ Si  $e = rs$  :  $N_e \rightarrow N_r N_s$
  - ▶ Si  $e = r \mid s$  :  $N_e \rightarrow N_r \mid N_s$
  - ▶ Si  $e = r^*$  :  $N_e \rightarrow N_r N_e \mid \epsilon$

## Exemple Expression Rationnelle vers Grammaire

- ▶ Expression Rationnelle :

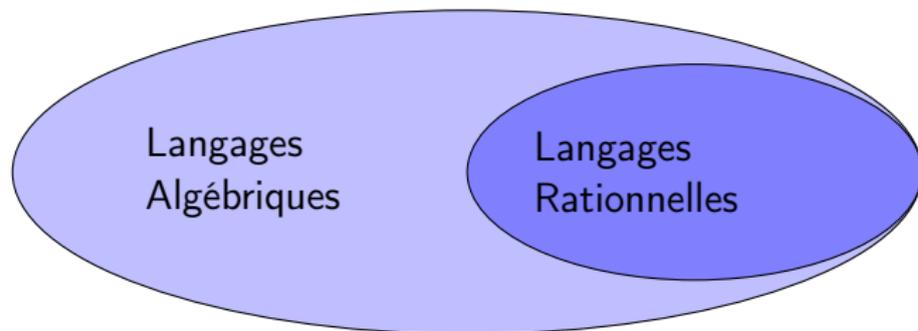
$$e = \underbrace{(a \mid b)^*}_1 \underbrace{(aa \mid \epsilon)}_2$$

- ▶ Grammaire :

$$\begin{array}{ll} S_e & \rightarrow N_1 N_2 \\ N_1 & \rightarrow N_{a|b} N_1 \mid \epsilon \\ N_{a|b} & \rightarrow N_a \mid N_b \\ N_a & \rightarrow a \\ N_b & \rightarrow b \end{array} \qquad \begin{array}{ll} N_2 & \rightarrow N_{aa} \mid N_\epsilon \\ N_{aa} & \rightarrow N_a N_a \\ N_\epsilon & \rightarrow \epsilon \end{array}$$

## Relation entre Langages Rationnels et Langages Algébriques

- ▶ Tout langage rationnel est algébrique.
- ▶ Nous avons vu deux preuves pour ça.
- ▶ Il y a des langages algébriques qui ne sont pas rationnels.
- ▶ Exemple :  $\{a^n b^n \mid n \geq 0\}$



## Formalismes pour des types de langages

Langages ...	Formalisme générateur	Formalisme analyseur
... rationnels	Expressions rationnelles	Automates déterministes
... algébriques	Grammaires algébriques	(dans quelques semaines)

## Forme de Backus-Naur

- ▶ Une notation très utilisée pour la documentation des langages de programmation est la notation *BNF* (Backus-Naur Form) qui est équivalente aux grammaires algébriques.
- ▶ Les non-terminaux sont écrits entre chevrons :  $\langle T \rangle$ .
- ▶ La flèche est remplacée par `::=`
- ▶ Les alternatives pour le même non-terminal sont regroupées, et séparées par `|`
- ▶ Exemple :  
`<cond> ::= if "(" <condition> ")" "{" <code> "}"`

## Forme de Backus-Naur étendue

- ▶ *EBNF* : Extended Backus-Naur Form
- ▶ La forme étendue permet des constructions connues des expressions rationnelles dans les côtés droits des règles :
  - ▶ des choix séparés par |
  - ▶ des groupes optionnels entre crochets [ et ]
  - ▶ des groupes répétés un nombre quelconque de fois entre accolades { et }
- ▶ Exemple :  $\langle \text{explist} \rangle ::= "( \langle \text{exp} \rangle \{ ", \langle \text{exp} \rangle \} ") "$
- ▶ C'est encore équivalent aux grammaires algébriques.

## De EBNF aux grammaires : [ ]

- ▶ Si la forme EBNF contient

$$\langle A \rangle ::= \dots [e] \dots$$

- ▶ alors on peut le remplacer par

$$\langle A \rangle ::= \dots \langle N \rangle \dots$$

$$\langle N \rangle ::= \epsilon \mid e$$

où N est un nouveau symbole non-terminal.

## De EBNF aux grammaires : { }

- ▶ Si la forme EBNF contient

$$\langle A \rangle ::= \dots \{e\} \dots$$

- ▶ alors on peut le remplacer par

$$\langle A \rangle ::= \dots \langle N \rangle \dots$$

$$\langle N \rangle ::= \epsilon \mid e \langle N \rangle$$

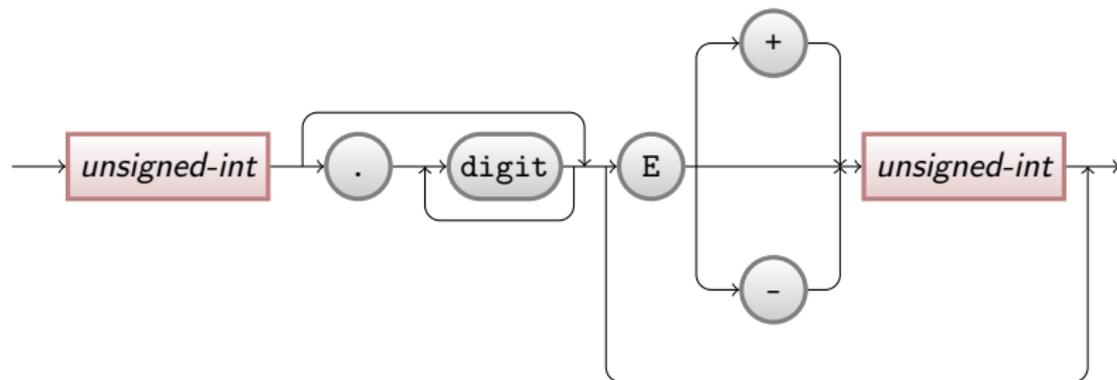
où N est un nouveau symbole non-terminal.

## Diagramme de syntaxe non récursif

unsigned-int :



unsigned-number :



Correspond à une séquence d'expressions rationnelles.

## Expressions rationnelles pour l'exemple

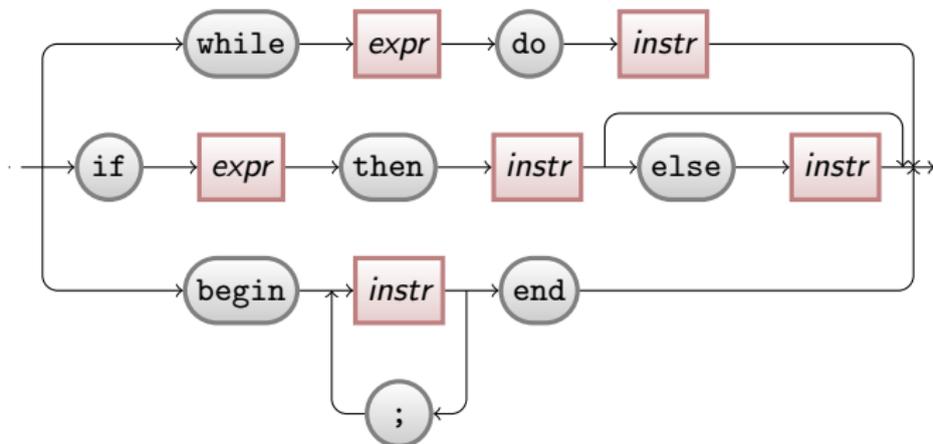
On suppose donnée une définition de l'expression rationnelle *digit*.

$$\textit{unsigned-int} = \textit{digit}^+$$

$$\textit{unsigned-number} = \textit{unsigned-int} (. \textit{digit}^+)? (\mathbf{E}(\mathbf{+} \mid \mathbf{-})? \textit{unsigned-int})?$$

## Diagramme de syntaxe récursif

instr : (fragment seulement)



Correspond à une grammaire algébrique.

Les anglophones parlent de *railroad diagram*.

## Grammaire pour l'exemple



```
instr  →  while expr do instr
        |  if expr then instr N1
        |  begin N2 end
N1   →  ε
        |  else instr
N2   →  instr
        |  instr ; N2
```

- ▶ Attention cette grammaire est ambiguë (problème du *dangling else*).