

## Grammaires et Analyse Syntaxique - Cours 3

### Grammaires algébriques

Ralf Treinen



treinen@irif.fr

8 février 2024

© Ralf Treinen 2020–2024

### 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 :

$E \rightarrow E+E$

$E \rightarrow E * E$

$E \rightarrow (E)$

$E \rightarrow C$

$C \rightarrow i$

$C \rightarrow v$

### Petit rappel du cours 2

- ▶ Exemple d'une grammaire algébrique :

$(\{a, b\}, \{A\}, A, A \rightarrow aAb \mid \epsilon)$

- ▶ Exemple d'une dérivation dans cette grammaire

$A \rightarrow aAb \rightarrow aaAbb \rightarrow aaaAbbb \rightarrow aaaAbbb \rightarrow aaabbb$

- ▶ Langage engendré par cette grammaire :

$\{a^n b^n \mid n \geq 0\}$

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

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

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

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

## 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).

## 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  $i \in [0, n-1]$ .

## Choix dans les constructions de dérivations

- ▶ Nous avons vu dans cet exemple (l'exemple du cours 2 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).

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

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

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

## 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

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

### 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

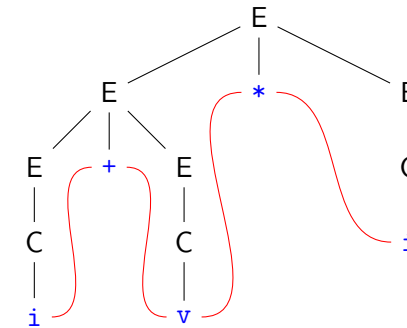
- ▶ la racine est étiquetée par l'axiome  $S$  ;
- ▶ tout nœud interne est étiqueté par un symbole de  $N$  ;
- ▶ toute feuille est étiquetée par un symbole de  $\Sigma \cup \{\epsilon\}$  ;
- ▶ si les fils pris de gauche à droite d'un nœud interne étiqueté par le non-terminal  $A$  sont étiquetés par 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  $G_2$  :  $E \rightarrow E+E \mid E * E \mid (E) \mid C \quad C \rightarrow i \mid v$



## 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

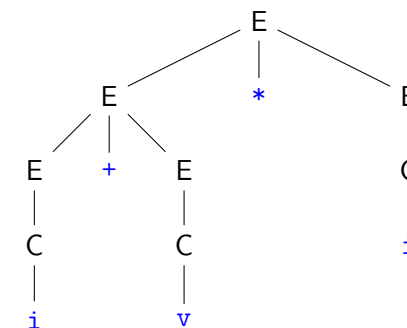
$$\begin{aligned} \alpha_n &= X_1 \dots X_{k-1} X_k X_{k+1} \dots X_m \\ \alpha_{n+1} &= X_1 \dots X_{k-1} Y_1 \dots Y_l X_{k+1} \dots X_m \end{aligned}$$

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

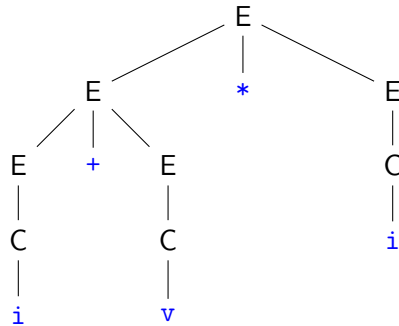
On ajoute dans  $t$  à la feuille  $X_k$  les enfants  $Y_1 \dots Y_l$ .

## Exemple, à partir d'une dérivation gauche

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

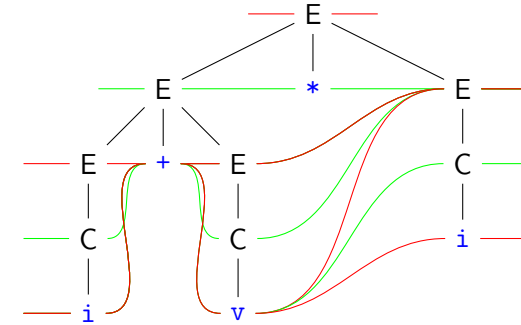


## Exemple, à partir d'une dérivation droite

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


## 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.

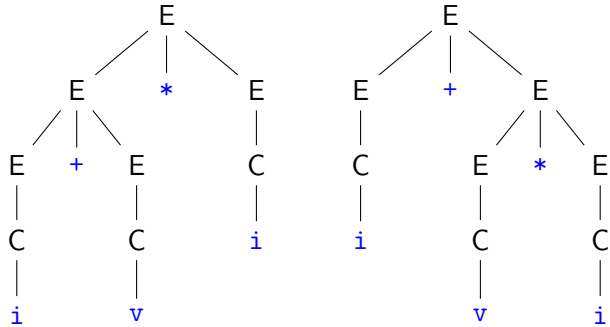
### Sur l'exemple $G_2$

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



## 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,  $w \in \Sigma^*$ .

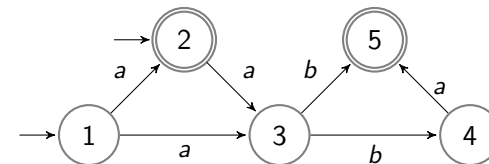
## Un autre exemple

- 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)

## Exemple Automate vers Grammaire



$$\begin{aligned} S_0 &\rightarrow N_1 & S_0 &\rightarrow N_2 \\ N_1 &\rightarrow a N_2 & N_1 &\rightarrow a N_3 \\ N_2 &\rightarrow a N_3 & N_3 &\rightarrow b N_4 \\ N_3 &\rightarrow b N_5 & N_4 &\rightarrow a N_5 \\ N_2 &\rightarrow \epsilon & N_5 &\rightarrow \epsilon \end{aligned}$$

## 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 :

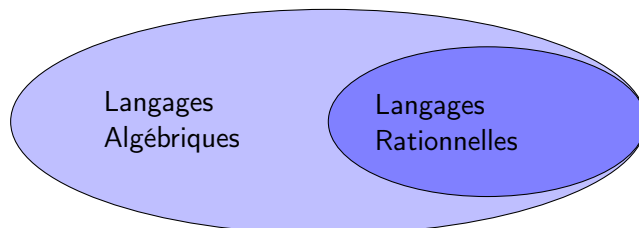
$$e = \underbrace{(a \mid b)^*}_1 \underbrace{(aa \mid \epsilon)}_2$$

- ▶ Grammaire :

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

## Relation entre Langages Rationnels et Langages Algébriques

- ▶ Tout langage rationnel est algébrique.
- ▶ Nous avons vu deux preuves pour ça (une aurait suffi).
- ▶ 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 :  
 $\langle \text{cond} \rangle ::= \text{if } "(" \langle \text{condition} \rangle ") " \{ " \langle \text{code} \rangle " \}$

## 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.

## 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  $[ \text{ et } ]$
  - ▶ des groupes répétés un nombre quelconque de fois entre accolades  $\{ \text{ et } \}$
- ▶ 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

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

où  $N$  est un nouveau symbole non-terminal.

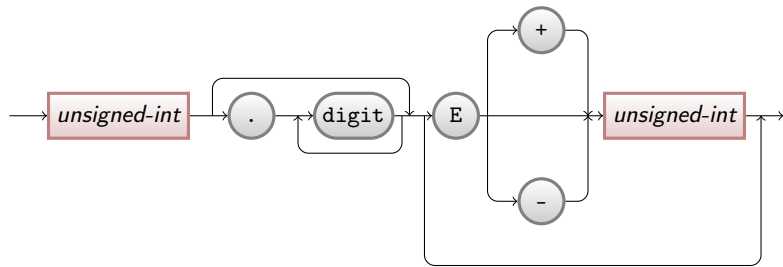


## Diagramme de syntaxe non récursif

unsigned-int :



unsigned-number :



Correspond à une séquence d'expressions rationnelles.

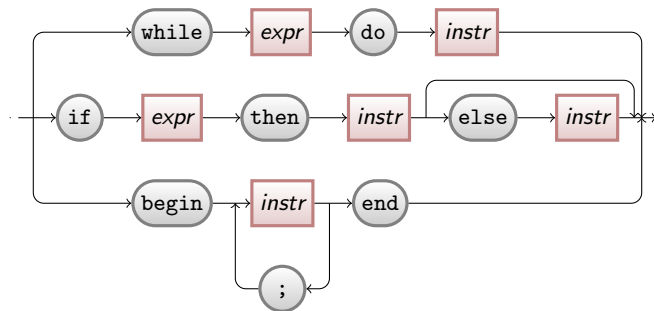
## Expressions rationnelles pour l'exemple

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

$$\begin{aligned} \text{unsigned-int} &= \text{digit} + \\ \text{unsigned-number} &= \text{unsigned-int} (. \text{digit} +)^? (\text{E} (+ | -)^? \text{unsigned-int})? \end{aligned}$$

## 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*).

## Plusieurs techniques pour l'analyse grammaticale

- ▶ **Analyse descendante** : construction de l'arbre de dérivation, à partir de l'axiome jusqu'aux feuilles.  
Ordre de construction : parcours préfixe de l'arbre.  
C'est l'approche que nous étudions à partir de la semaine prochaine.
- ▶ **Analyse ascendante** : construction d'un arbre de dérivation à partir des feuilles jusqu'à l'axiome. Plus complexe à maîtriser, mais aussi plus puissante.  
C'est l'approche que nous commencerons à étudier dans trois semaines.

## Exploration complète de l'espace de recherche ?

- ▶ Une façon de réaliser une analyse grammaticale est maintenant d'essayer simplement toutes les possibilités de choisir des règles.
- ▶ Cela donne lieu à un algorithme **non-déterministe** :
  - ▶ soit par **retour en arrière** (angl. : backtracking)
  - ▶ soit par **programmation dynamique**
- ▶ Approche complète : on est sûr de trouver un arbre de dérivation si le mot est dans le langage ☺
- ▶ Problème : efficacité ☹
- ▶ On cherche des solutions efficaces, éventuellement en imposant des restrictions aux grammaires qu'on peut traiter.

## Construction d'un arbre de dérivation

- ▶ Dans la construction d'un arbre de dérivation (ou, d'une dérivation), il y a à chaque moment deux choix à faire :
  - ▶ le non-terminal qu'on va remplacer à l'aide d'une règle de la grammaire,
  - ▶ une fois le non-terminal choisi, la règle parmi celles qui ont ce non-terminal sur le côté gauche.
- ▶ Nous avons vu la semaine dernière que le premier choix n'est pas essentiel : on peut imposer une stratégie pour choisir le non-terminal à remplacer (par ex., celui qui est le plus à gauche).

## Quelle efficacité cherche-t-on ?

- ▶ Le temps d'exécution d'un analyseur grammaticale est au moins linéaire dans la longueur du texte à analyser (c-à-d la longueur du flot des tokens). Évidemment on ne peut pas faire mieux.
- ▶ Puisqu'on cherche aussi à construire l'arbre de dérivation, on ne peut même pas faire mieux que la taille de l'arbre de dérivation.
- ▶ La taille de l'arbre de dérivation est  $\geq$  la taille de l'entrée (car chaque symbole de l'entrée est une feuille de l'arbre).
- ▶ On veut aussi que l'analyseur fasse un seul passage sur le texte.