Grammaires et Analyse Syntaxique - Cours 3 Grammaires algébriques

Ralf Treinen

treinen@irif.fr

8 février 2024

Petit rappel du cours 2

- Exemple d'une grammaire algébrique :

$$(\{\mathtt{a}, \mathtt{b}\}, \{\mathrm{A}\}, \mathrm{A}, \mathrm{A} \to \mathtt{a}\mathrm{A}\mathtt{b} \mid \epsilon)$$

- Exemple d'une dérivation dans cette grammaire

$$\mathrm{A} \to \mathtt{a}\mathrm{A}\mathtt{b} \to \mathtt{aa}\mathrm{A}\mathtt{bb} \to \mathtt{aaa}\mathrm{A}\mathtt{bbb} \to \mathtt{aaa}\mathrm{A}\mathtt{bbb} \to \mathtt{aaabbb}$$

- Langage engendré par cette grammaire :

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

Exemple plus intéressant d'une grammaire algébrique

$G_2 = (\Sigma, N, S, P)$ où

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

$$
\begin{aligned}
\mathrm{E} &\rightarrow \mathrm{E}\texttt{+}\mathrm{E} \\
\mathrm{E} &\rightarrow \mathrm{E}\texttt{*}\mathrm{E} \\
\mathrm{E} &\rightarrow \texttt{(}\mathrm{E}\texttt{)} \\
\mathrm{E} &\rightarrow \mathrm{C} \\
\mathrm{C} &\rightarrow \texttt{i} \\
\mathrm{C} &\rightarrow \texttt{v}
\end{aligned}
$$

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

Productions de G_2 : E → E+E | E*E | (E) | C C → `i` | `v`
Ici : en rouge le non-terminal qui est réécrit.

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

Langage engendré

Définition (rappel)

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

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

Sur l'exemple G_2

$\mathcal{L}(G_2)$ est l'ensemble des expressions arithmétiques formées à l'aide des opérateurs `+` et `*`, des paranthèses, 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 (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 α, alors pour faire l'étape suivante il faut choisir :
  1. une occurrence d'un non-terminal dans α (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. $u = w_1 A w_2$, où $w_1 \in \Sigma^*$, $A \in N$ et $w_2 \in (N \cup \Sigma)^*$;
2. $A \to w$ est une règle de P ;
3. $v = w_1 w w_2$.

En anglais : *leftmost derivation*.

Dérivation gauche

Une *dérivation gauche* est une suite finie $w_0, w_1, \ldots, w_n$ 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 | E*E | (E) | C C $\rightarrow$ `i` | `v`

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

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. $A \to w$ est une règle de P ;
3. $v = w_1 w w_2$.

Dérivation droite

Une *dérivation droite* est une suite finie $w_0, w_1, \ldots, w_n$ 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 : $\mathrm{E} \rightarrow \mathrm{E}\texttt{+}\mathrm{E} \mid \mathrm{E}\texttt{*}\mathrm{E} \mid \texttt{(}\mathrm{E}\texttt{)} \mid \mathrm{C} \qquad \mathrm{C} \rightarrow \texttt{i} \mid \texttt{v}$

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

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

- Dans une dérivation d'une grammaire algébrique, on remplace toujours un non-terminal sans de 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

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

- alors on peut réarranger l'ordre des réécritures en

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

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,

$$S \to^* w \qquad \text{ssi} \qquad S \overset{g}{\to}^* w \qquad \text{ssi} \qquad S \overset{d}{\to}^* w$$

- 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

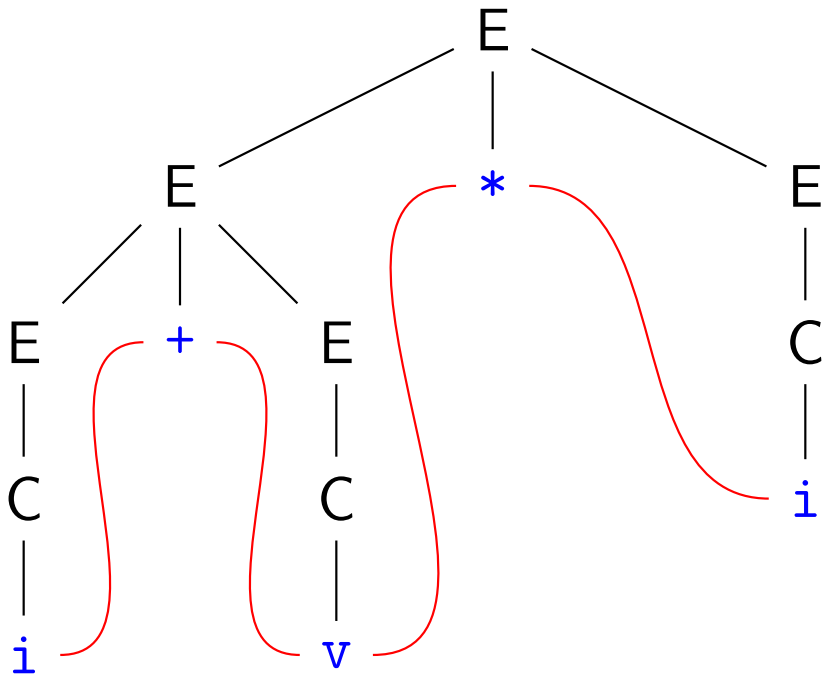
- la racine est étiquetée par l'axiome S ;
- tout nœud interne est étiqueté par un symbole de N ;
- toute feuille est étiquetée par un symbole de $\Sigma \cup \{\epsilon\}$;
- si les fils pris de gauche à droite d'un nœud interne étiqueté par le non-terminal A sont étiquetés par les symboles respectifs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

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 → E`+`E | E`*`E | `(`E`)` | C C → `i` | `v`



Dérivation vers arbre de dérivation

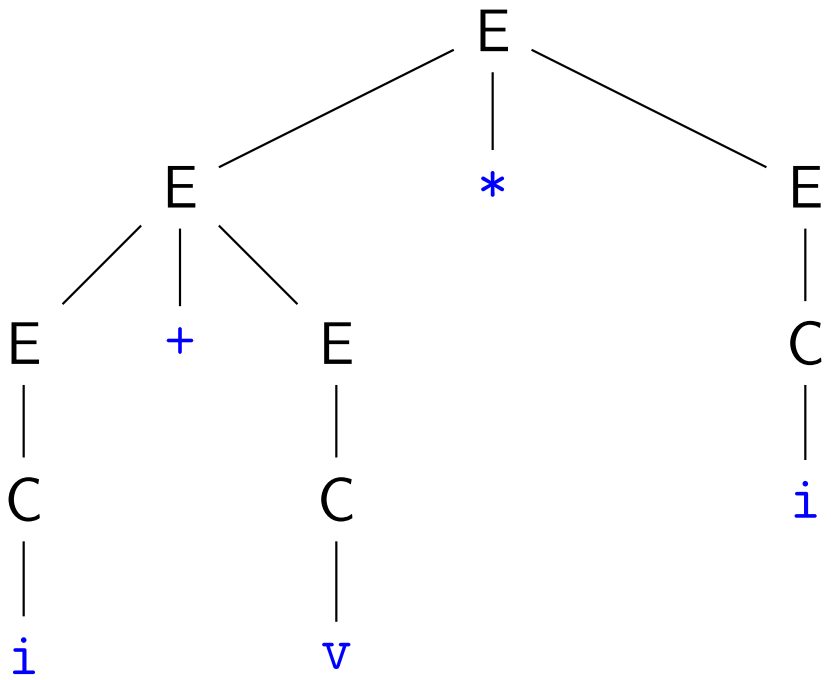
- On définit, pour une dérivation $S \rightarrow \alpha_1 \rightarrow \ldots \rightarrow \alpha_n$, où $\forall i : \alpha_i \in (\Sigma \cup N)^*$, un arbre de dérivation pour α_n.
- Par induction sur n :
  - Si $n = 0$: l'arbre consiste seulement dans le nœud S.
  - De n à $n+1$: Soient

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

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 $Y_1 \ldots Y_l$.

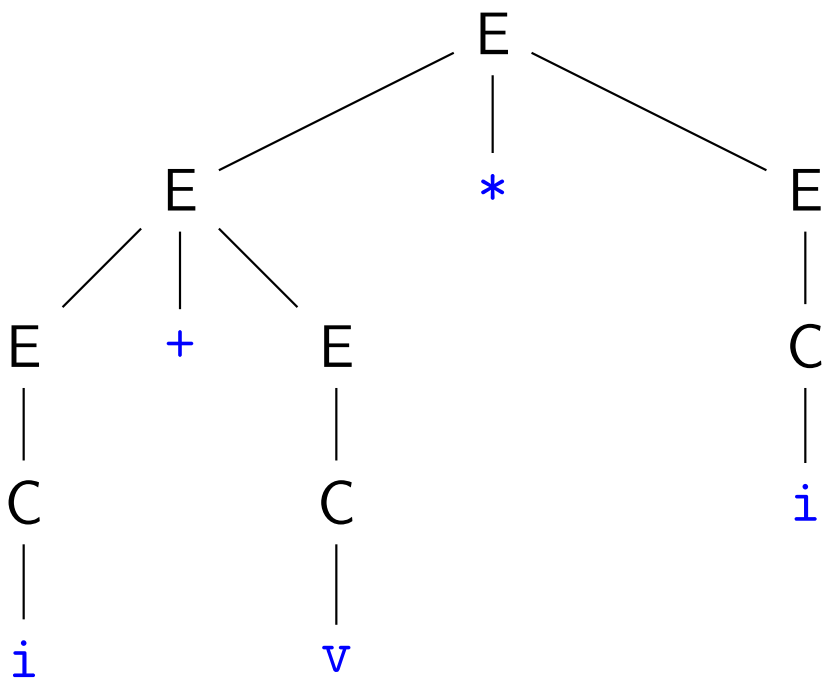
Exemple, à partir d'une dérivation gauche

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



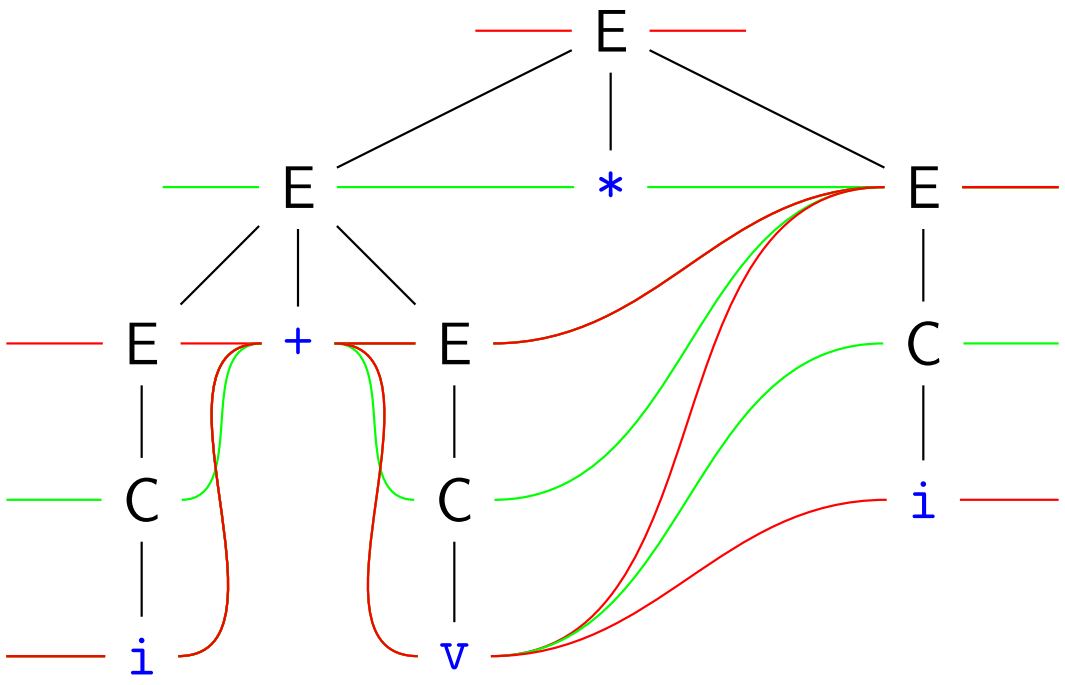
Exemple, à partir d'une dérivation droite

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



Arbre de dérivation vers dérivation

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



$E \to E*E \to E+E*E \to C+E*E \to \texttt{i}+E*E \to \texttt{i}+C*E \to \texttt{i}+\texttt{v}*E \to \texttt{i}+\texttt{v}*C \to \texttt{i}+\texttt{v}*\texttt{i}$

Arbres de dérivation et Dérivation gauche

- Pour chaque arbre de dérivation il y une unique dérivation gauche.
- Pour chaque arbre de dérivation il y une unique dérivation droite.
- Pour chaque arbre de dérivation il y 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.

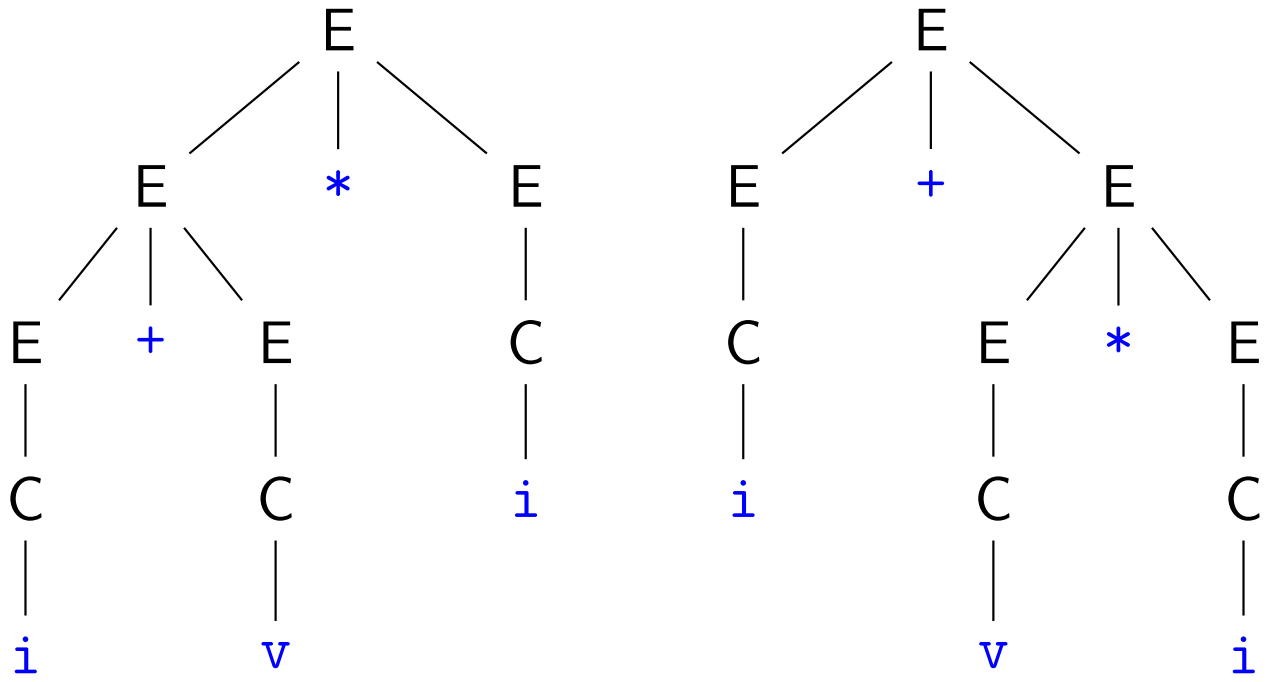
Sur l'exemple G_2

Rappel : E $\rightarrow$ E`+`E | E`*`E | `(`E`)` | C C $\rightarrow$ `i` | `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 → E`+`E | E`*`E | `(`E`)` | C C → `i` | `v`



Un autre exemple

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

$$\mathrm{E} \quad \rightarrow \quad \texttt{i} \mid \texttt{v} \mid \texttt{(}\mathrm{E}\texttt{+}\mathrm{E}\texttt{)} \mid \texttt{(}\mathrm{E}\texttt{*}\mathrm{E}\texttt{)}$$

- Cette grammaire décrit les expressions arithmétiques complètement paranthésées.
- 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 :

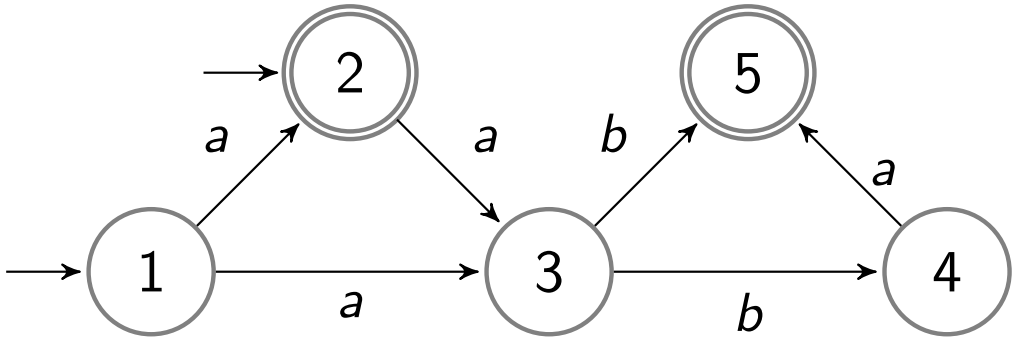
$$\begin{array}{lcll} S_0 & \rightarrow & N_q & q \in I \\ N_q & \rightarrow & a\ N_p & p \in \delta(q, a) \\ N_q & \rightarrow & \epsilon & q \in F \end{array}$$

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

$$\begin{array}{lcll} \mathrm{N} & \rightarrow & w & \mathrm{M} \\ \mathrm{N} & \rightarrow & w & \end{array}$$

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

Exemple Automate vers Grammaire



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

Traduction d'une expression rationnelle en grammaire

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

Exemple Expression Rationnelle vers Grammaire

- Expression Rationnelle :

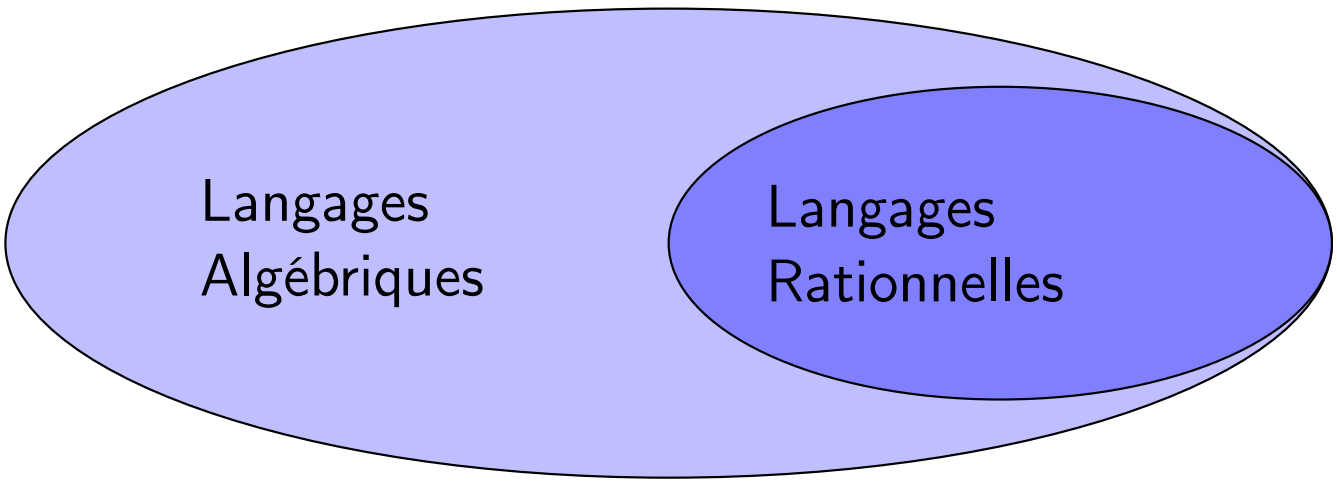
$$e = \underbrace{(a \mid b)^*}_{1}\underbrace{(aa \mid \epsilon)}_{2}$$

- Grammaire :

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

Relation entre Langages Rationnels et Langages Algébriques

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



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 : `<explist> ::= "(" <exp> {"," <exp>} ")"`
- C'est encore équivalent aux grammaires algébriques.

De EBNF aux grammaires : [ ]

- Si la forme EBNF contient

$$\texttt{<A>} \quad ::= \quad \ldots \; [e] \; \ldots$$

- alors on peut le remplacer par

$$\texttt{<A>} \quad ::= \quad \ldots \; \texttt{<N>} \; \ldots$$
$$\texttt{<N>} \quad ::= \quad \epsilon \mid e$$

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

De EBNF aux grammaires : { }

- Si la forme EBNF contient

$$\texttt{<A>} \quad ::= \quad \ldots \ \{e\} \ \ldots$$

- alors on peut le remplacer par

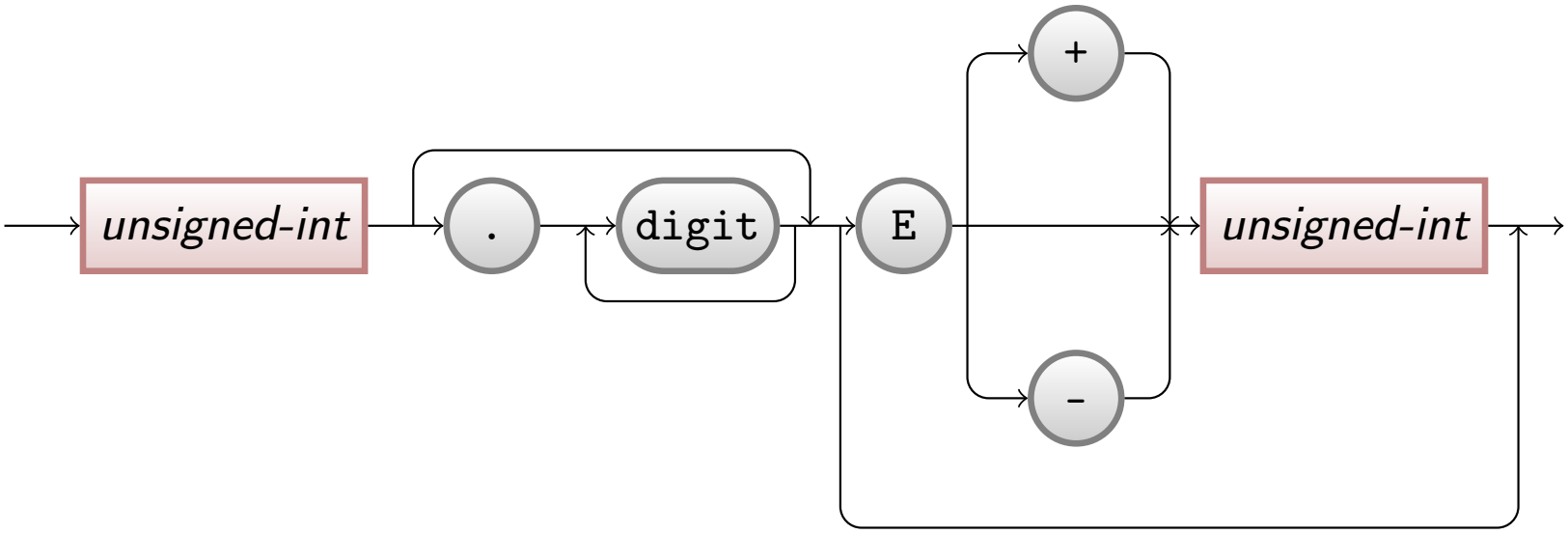
$$\texttt{<A>} \quad ::= \quad \ldots \ \texttt{<N>} \ \ldots$$
$$\texttt{<N>} \quad ::= \quad \epsilon \mid e \ \texttt{<N>}$$

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

Diagramme de syntaxe non récursif

unsigned-int :

unsigned-number :



Correspond à une séquence d'expressions rationnelles.

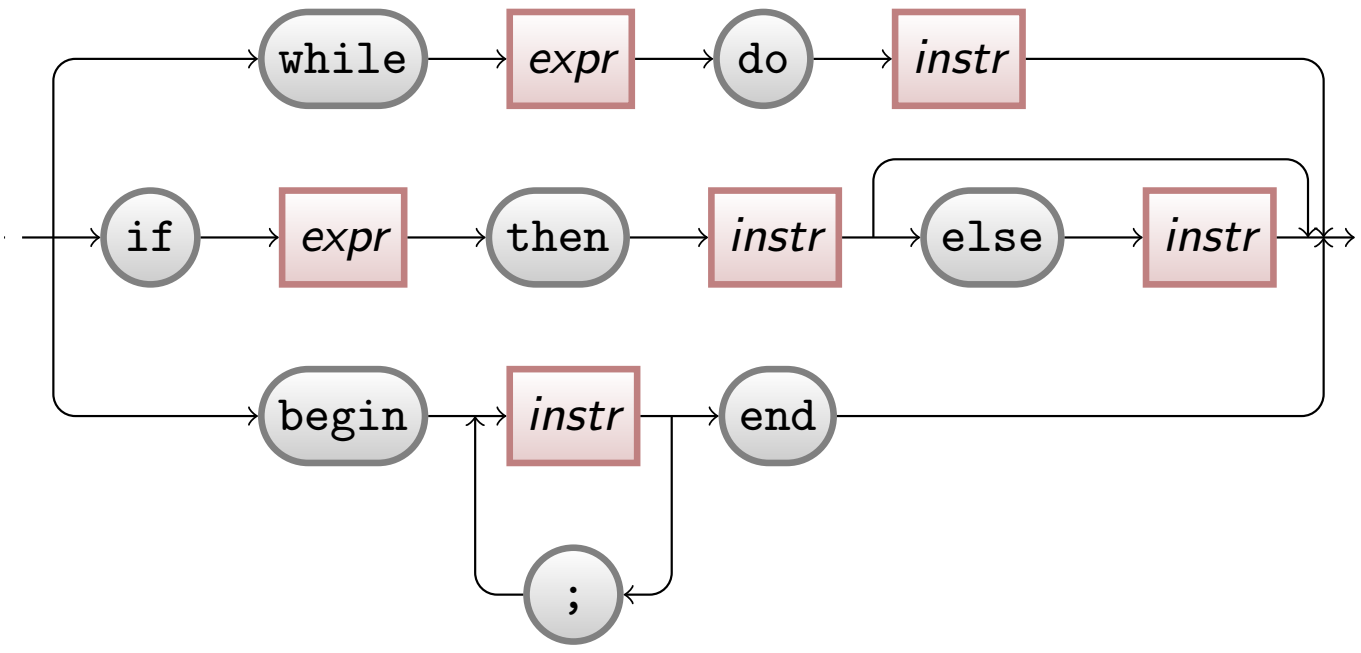
Expressions rationnelles pour l'exemple

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

$$\begin{aligned} \textit{unsigned-int} &= \textit{digit}+ \\ \textit{unsigned-number} &= \textit{unsigned-int}\ (\texttt{.}\,\textit{digit}+)?\ (\texttt{E}(\texttt{+} \mid \texttt{-})?\textit{unsigned-int})? \end{aligned}$$

Diagramme de syntaxe récursif

instr : (fragment seulement)



Correspond à une grammaire algébrique.
Les anglophones parlent de *railroad diagram*.

Grammaire pour l'exemple

- $$\begin{array}{rcl} \text{instr} & \rightarrow & \texttt{while}\ \text{expr}\ \texttt{do}\ \text{instr} \\ & | & \texttt{if}\ \text{expr}\ \texttt{then}\ \text{instr}\ N_1 \\ & | & \texttt{begin}\ N_2\ \texttt{end} \\ N_1 & \rightarrow & \epsilon \\ & | & \texttt{else}\ \text{instr} \\ N_2 & \rightarrow & \text{instr} \\ & | & \text{instr}\ \texttt{;}\ N_2 \end{array}$$

- 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.

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).

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.

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.