Grammaires et Analyse Syntaxique - Cours 3
Grammaires algébriques

Ralf Treinen

treinen@irif.fr

10 février 2022

Introduction : Grammaires

- Vu jusqu'à maintenant : analyse lexicale (ocamllex)
- Aujourd'hui nous verrons le 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 un compteur non borné, ne sont pas reconnaissables.
- Exemple : $\{a^n b^n \mid n \geq 0\}$ n'est pas reconnaissable (vu dans le cours AAL3).

Définition des grammaires algébriques

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

- Σ est un ensemble fini de symboles, appelé les *symboles terminaux* de G ;
- N est un ensemble fini et disjoint de Σ 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 $A \to u$, où $A \in N$, et $u \in (\Sigma \cup N)^*$.

Premier exemple d'une grammaire algébrique

$G_1 = (\Sigma, N, S, P)$ où

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

$$\begin{aligned} \mathsf{S} &\rightarrow \mathtt{a}\mathsf{S}\mathtt{b} \\ \mathsf{S} &\rightarrow \epsilon \end{aligned}$$

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

Exemple plus intéressant d'une grammaire algébrique

$G_2 = (\Sigma, N, S, P)$ où

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

$$\begin{array}{lcl}
\mathrm{E} & \to & \mathrm{E}\mathtt{+}\mathrm{E} \\
\mathrm{E} & \to & \mathrm{E}\mathtt{*}\mathrm{E} \\
\mathrm{E} & \to & \mathtt{(}\mathrm{E}\mathtt{)} \\
\mathrm{E} & \to & \mathrm{C} \\
\mathrm{C} & \to & \mathtt{i} \\
\mathrm{C} & \to & \mathtt{v}
\end{array}$$

Notation : alternatives

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

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

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

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

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 plusieures étapes*, noté $u \to_G^* v$ (ou abrégé $u \to^* v$), s'il existe une suite finie $w_0, w_1, \ldots, w_n$ de mots de $(N \cup \Sigma)^*$ telle que $w_0 = u$, $w_i \to_G w_{i+1}$ pour tout $i \in [0, n-1]$, et $w_n = v$.

Remarques

- $n = 0$ est autorisé dans la définition de $u \to^* v$ (dans ce cas on a que $u = v$)
- $u \to_G^* v$ s'il existe une séquence (éventuellement réduite à un seul élément)

$$u \to_G \cdot \to_G \cdots \to_G v$$

- On appelle n la *longueur* de la dérivation.

Exemple : Dérivation de aaabbb dans G_1

Pour la grammaire G_1 donnée au-dessus on a les étapes de dérivation suivantes :

$$\begin{aligned} S &\rightarrow \texttt{a}S\texttt{b} \\ &\rightarrow \texttt{aa}S\texttt{bb} \\ &\rightarrow \texttt{aaa}S\texttt{bbb} \\ &\rightarrow \texttt{aaabbb} \end{aligned}$$

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

Pour la grammaire G_2 donnée au-dessus on a les étapes de dérivation suivantes :
(Nous mettons en rouge le non-terminal qui est réécrit.)

$$
\begin{array}{lcl}
E & \rightarrow & E+E \\
 & \rightarrow & E+E*E \\
 & \rightarrow & C+E*E \\
 & \rightarrow & C+E*C \\
 & \rightarrow & C+E*\texttt{i} \\
 & \rightarrow & \texttt{i}+E*\texttt{i} \\
 & \rightarrow & \texttt{i}+C*\texttt{i} \\
 & \rightarrow & \texttt{i+v*i}
\end{array}
$$

Langage engendré

Définition

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

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

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

Sur les exemples :

- $\mathcal{L}(G_1) = \{a^n b^n \mid n \geq 0\}$, un langage non-reconnaissable (ça se montre avec le lemme de l'étoile, par exemple) !
- $\mathcal{L}(G_2)$ est l'ensemble des expressions arithmétiques formées à l'aide des opérateurs `+` et `*`, et des constantes `i` et `v`. Ce langage n'est pas régulier non plus (ça se montre mieux avec le théorème de Myhill-Nerode).

Choix dans les constructions de dérivations

- Nous avons vu dans le deuxième exemple (le premier 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. $v = w_1 w w_2$.

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

$$
\begin{array}{lll}
\mathsf{E} & \overset{g}{\rightarrow} & \mathsf{E*E} \\
 & \overset{g}{\rightarrow} & \mathsf{E+E*E} \\
 & \overset{g}{\rightarrow} & \mathsf{C+E*E} \\
 & \overset{g}{\rightarrow} & \mathtt{i+}\mathsf{E*E} \\
 & \overset{g}{\rightarrow} & \mathtt{i+}\mathsf{C*E} \\
 & \overset{g}{\rightarrow} & \mathtt{i+v*}\mathsf{E} \\
 & \overset{g}{\rightarrow} & \mathtt{i+v*}\mathsf{C} \\
 & \overset{g}{\rightarrow} & \mathtt{i+v*i}
\end{array}
$$

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

$$
\begin{array}{lcl}
E & \xrightarrow{d} & E+E \\
  & \xrightarrow{d} & E+E*E \\
  & \xrightarrow{d} & E+E*C \\
  & \xrightarrow{d} & E+E*\mathtt{i} \\
  & \xrightarrow{d} & E+C*\mathtt{i} \\
  & \xrightarrow{d} & E+\mathtt{v}*\mathtt{i} \\
  & \xrightarrow{d} & C+\mathtt{v}*\mathtt{i} \\
  & \xrightarrow{d} & \mathtt{i}+\mathtt{v}*\mathtt{i}
\end{array}
$$

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 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 (il faut quand même faire un peu attention).
- Conséquence : pour tout mot w,

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

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

Arbre de dérivation

Définition

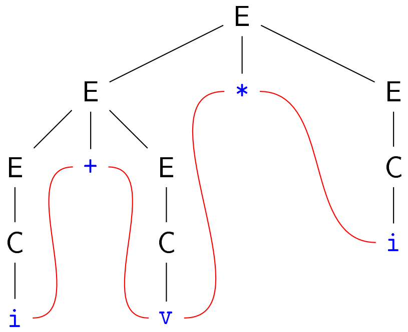
Soit $G = (\Sigma, N, S, P)$ une grammaire. Un *arbre de dérivation* de G est un arbre tel que

- 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 $\alpha_1, \ldots, \alpha_n$, 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



Dérivation vers arbre de dérivation

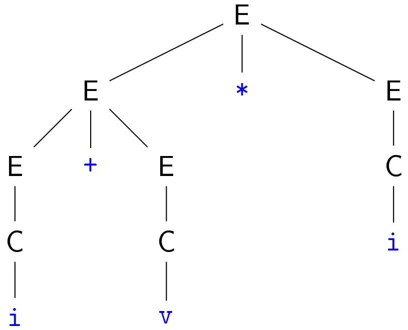
- On définit, pour une dérivation $S \to \alpha_1 \to \ldots \to \alpha_n$, où $\forall i : \alpha_i \in (\Sigma \cup N)^*$, un arbre de dérivation pour α_n.
- Par induction sur n :
  - Si $n = 0$: l'arbre consiste seulement dans le nœud S.
  - De n à $n+1$: Soient

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

    Par hypothèse d'induction, il existe un arbre de dérivation t pour $S \to^* \alpha_n$.
    On ajoute dans t à la feuille X_k les enfants $Y_1 \ldots Y_l$.

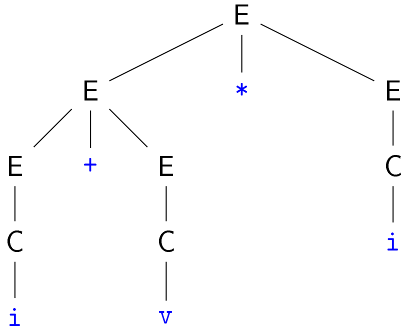
Exemple, à partir d'une dérivation gauche

$\mathsf{E} \rightarrow \mathsf{E{*}E} \rightarrow \mathsf{E{+}E{*}E} \rightarrow \mathsf{C{+}E{*}E} \rightarrow \mathtt{i{+}}\mathsf{E{*}E} \rightarrow \mathtt{i{+}}\mathsf{C{*}E} \rightarrow \mathtt{i{+}v{*}}\mathsf{E} \rightarrow \mathtt{i{+}v{*}}\mathsf{C} \rightarrow \mathtt{i{+}v{*}i}$



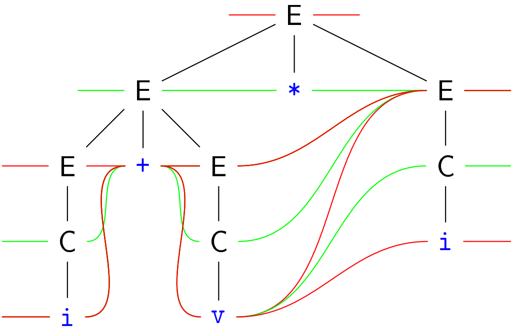
Exemple, à partir d'une dérivation droite

E → E*E → E*C → E*i → E+E*i → E+C*i → E+v*i →
C+v*i → i+v*i



Arbre de dérivation vers dérivation

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



E → E*E → E+E*E → C+E*E → i+E*E → i+C*E → i+v*E → i+v*C → i+v*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.

Grammaires ambiguës

Définition

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

Définition équivalente

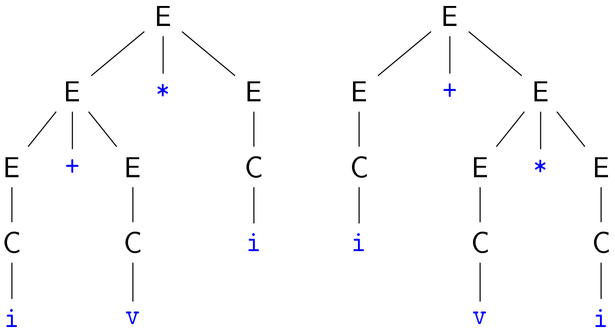
Une grammaire G est non-ambiguë quand tout $w \in \mathcal{L}(G)$ a une seule dérivation gauche.

Sur l'exemple G_2

Rappel : $\mathrm{E} \rightarrow \mathrm{E}\texttt{+}\mathrm{E} \mid \mathrm{E}\texttt{*}\mathrm{E} \mid \texttt{(}\mathrm{E}\texttt{)} \mid \mathrm{C} \qquad \mathrm{C} \rightarrow \texttt{i} \mid \texttt{v}$

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



Un autre exemple

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

$$\mathsf{E} \rightarrow \mathtt{i} \mid \mathtt{v} \mid \mathtt{(}\mathsf{E}\mathtt{+}\mathsf{E}\mathtt{)} \mid \mathtt{(}\mathsf{E}\mathtt{*}\mathsf{E}\mathtt{)}$$

- 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 :

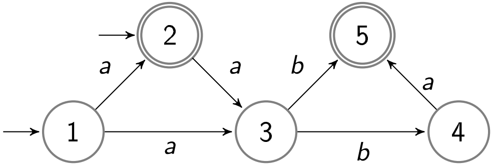
$$\begin{array}{lcll} S_0 & \to & N_q & q \in I \\ N_q & \to & a\ N_p & p \in \delta(q, a) \\ N_q & \to & \epsilon & q \in F \end{array}$$

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

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

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

Exemple Automate vers Grammaire



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

Traduction d'une expression rationnelle en grammaire

- Étant donnée une expression régulière 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 Regulière :

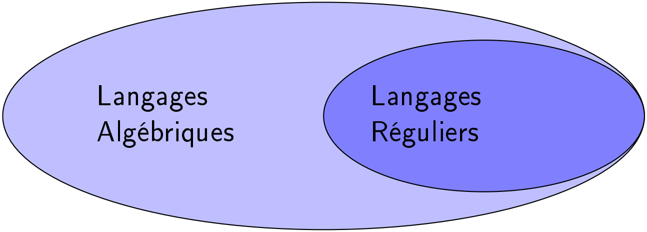
$$e = \underbrace{(a \mid b)^*}_{1}\underbrace{(aa \mid \epsilon)}_{2}$$

- 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 Réguliers et Langages Algébriques

- Tout langage régulier est algébrique.
- Nous avons vu deux preuves pour ça (une aurait suffit).
- Il y a des langages algébriques qui ne sont pas réguliers.
- Exemple : le langage des expressions parenthésées.



Formalismes pour des types de langages

Langages $\cdots$	Formalisme générateur	Formalisme analyseur
$\cdots$ réguliers	Expressions rationnelles	Automates déterministes
$\cdots$ algébriques	Grammaires algébriques	(voir 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

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

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

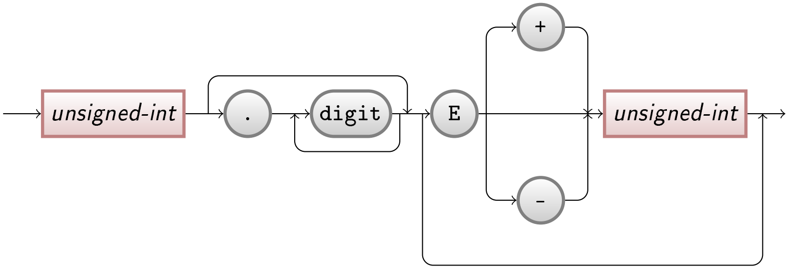
$$\begin{array}{lcl} \texttt{<A>} & ::= & \ldots \ \texttt{<N>} \ \ldots \\ \texttt{<N>} & ::= & \epsilon \mid e \ \texttt{<N>} \end{array}$$

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

Diagramme de syntaxe non récursif

unsigned-int :

unsigned-number :



Correspond à une séquence d'expressions rationnelles.

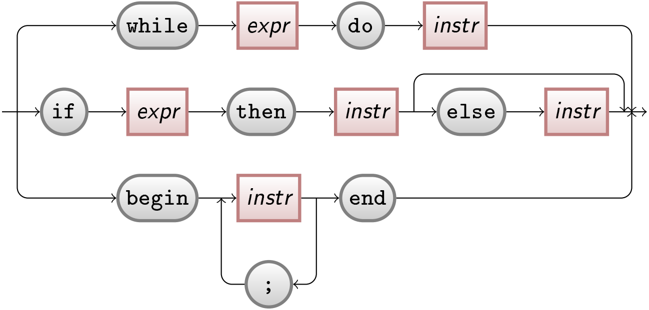
Expressions rationnelles pour l'exemple

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.

Grammaire pour l'exemple

- $$\begin{array}{rcl} \text{instr} & \rightarrow & \texttt{while}\ \text{expr}\ \texttt{do}\ \text{instr} \\ & | & \texttt{if}\ \text{expr}\ \texttt{then}\ \text{instr}\ \mathsf{N}_1 \\ & | & \texttt{begin}\ \mathsf{N}_2\ \texttt{end} \\ \mathsf{N}_1 & \rightarrow & \epsilon \\ & | & \texttt{else}\ \text{instr} \\ \mathsf{N}_2 & \rightarrow & \text{instr} \\ & | & \text{instr}\ \texttt{;}\ \mathsf{N}_2 \end{array}$$
- Attention cette grammaire est ambiguë (problème du *dangling else*).