Correction TD de Compilation n°4

Ingénieurs 2000 - IR2

—2005-2006—

Introduction aux grammaires

L'objectif de ce TD est de se familiariser avec les grammaires algébriques, les arbres de dérivation et la désambiguïsassions des grammaires.

▸ **Exercice 1.**

– *Écrire une grammaire G_1 qui reconnaît le langage des palindromes (mots non vides qui peuvent se lire de droite à gauche et de gauche à droite) sur l'alphabet* $\{\texttt{a},\texttt{b}\}$ *;*

✂ -

$$G_1 : \begin{array}{ll} S \rightarrow P\$ & P \rightarrow \texttt{a}\ P\ \texttt{a} \\ P \rightarrow \texttt{b}\ P\ \texttt{b} & P \rightarrow \texttt{aa} \\ P \rightarrow \texttt{bb} & \end{array}$$

- ✂

– *Écrire une grammaire G_2 qui reconnaît les mots du langage décrit par l'expression rationnelle* `a*b*` *et qui contient plus (strictement) de* `a` *que de* `b` *;*

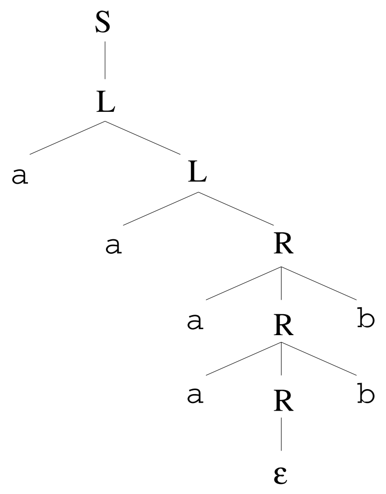
✂ -

$$G_2 : \begin{array}{ll} S \rightarrow L\$ & L \rightarrow \texttt{a}\ L \\ L \rightarrow \texttt{a}\ R & R \rightarrow \texttt{a}\ R\ \texttt{b} \\ R \rightarrow \epsilon & \end{array}$$

- ✂

– *Dessiner l'arbre de dérivation pour la grammaire G_2 du mot* `aaaabb`.

✂ -

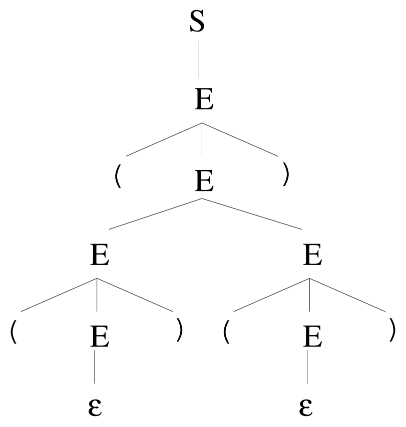


▸ Exercice 2.

Soit la grammaire suivante :

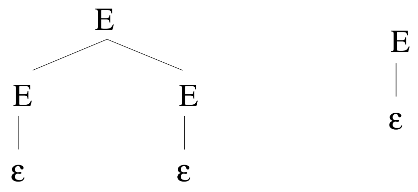
$$G_3 : \quad \begin{array}{ll} S \rightarrow E\$ & E \rightarrow E\ E \\ E \rightarrow \texttt{(}E\texttt{)} & E \rightarrow \epsilon \end{array}$$

– *Donner un arbre de dérivation de* `(()())` *pour cette grammaire.*



– *Cette grammaire est-elle ambiguë ? Expliquer.*

Oui, elle est ambiguë. En particulier, parce qu'elle accepte plusieurs arbres de dérivation pour le mot vide (ϵ).



– *Proposer une autre grammaire équivalente qui ne soit pas ambiguë.*

La première solution consiste à essayer de supprimer les ambiguïtés sur le mot vide en supprimant la règle $E \to \epsilon$ de la grammaire par fermeture des autres règles. On obtient la grammaire suivante :

$$\begin{array}{lll} G'_3 : & S \to E\$ & S \to \epsilon\$ \\ & E \to E\ E & E \to \texttt{()} \\ & E \to \texttt{(}E\texttt{)} & \end{array}$$

Autre solution, qui consiste à forcer l'ordre de la règle $E \to E\ E$. On obtient la grammaire suivante :

$$\begin{array}{lll} G''_3 : & S \to E\$ & E \to \texttt{(}E\texttt{)}E \\ & E \to \epsilon & \end{array}$$

– *Le langage décrit par cette grammaire peut-il être décrit par une expression rationnelle ?*

Si ce langage pouvait être décrit par une expression rationnelle, il devrait y avoir dans l'automate correspondant un état différent par nombre de parenthèses « encore ouvertes ». Ceci n'est pas possible dans un automate fini, si le nombre de parenthèses est quelconque.

▸ Exercice 3.

Soit la grammaire suivante :

$$\begin{array}{lll} G_4 : & S \to T\$ & T \to \texttt{(}T\texttt{)} \\ & T \to T\ T & T \to \texttt{INT} \\ & T \to \epsilon & \end{array}$$

– *Cette grammaire est-elle ambiguë ?*

✂ -

Oui elle est ambiguë pour la même raison que tout à l'heure.

- ✂

– *Proposer une grammaire non ambiguë.*

✂ -

Un première étape dans la transformation de la grammaire consiste à supprimer la production $E \to \epsilon$ par fermeture des productions comme dans l'exercice précédent.

$$\begin{array}{llll} G'_4 : & S \to T & S \to \epsilon & T \to T\ T \\ & T \to \texttt{(}T\texttt{)} & T \to \texttt{()} & T \to \texttt{INT} \end{array}$$

Cette grammaire reste ambiguë, par exemple, sur l'entrée `()()()` qui peut être interprété de deux manières. Pour supprimer cette autre ambiguïté, il suffit de forcer l'associativité de la proposition $T \to T\ T$. On la force de façon classique en ajoutant un non-terminal intermédiaire. Ici, on force l'associativité à droite.

$$\begin{array}{llll} G''_4 : & S \to T & S \to \epsilon & T \to U\ T \\ & T \to U & U \to \texttt{(}T\texttt{)} & U \to \texttt{()} \\ & U \to \texttt{INT} & & \end{array}$$

- ✂

▸ Exercice 4.

Soit la grammaire suivante des expressions arithmétiques :

$$\begin{array}{llll} G_5 : & S \to E & E \to E + E & E \to E - E \\ & E \to E * E & E \to E / E & E \to \texttt{INT} \\ & E \to \texttt{(}\ E\ \texttt{)} & & \end{array}$$

Désambiguïsez cette grammaire.

✂ -

Voir cours.

- ✂