Université Gustave Eiffel

Travaux Dirigés d'analyse syntaxique n°4

Licence d'informatique

———

Traçage des numéros de lignes. Grammaires algébriques (suite). Récursivité à gauche

D'une part, on trace les numéros de lignes depuis `flex` jusqu'à `bison`, et on les inclut dans les messages d'erreur. D'autre part, on manipule des grammaires algébriques, on détecte l'ambiguïté et on supprime la récursivité à gauche.

▸ **Exercice 1.** *Récupérez le fichier du projet* `tpc-2020-2021.y`. *Écrivez un programme* `flex` *transmettant les lexèmes attendus par* `tpc-2020-2021.y`. *Complétez ce programme* `flex` *ainsi que* `tpc-2020-2021.y` *pour que les messages d'erreur de syntaxe donnent le numéro de ligne de l'erreur dans la donnée. Testez avec le fichier* `trinome.tpc` *puis en introduisant une erreur dedans.*

▸ **Exercice 2.** *Proposer des grammaires pour les langages suivants :*

1. *Les mots de* $\{a, b\}^*$ *de longueur paire*
2. $\{w \in X^* \,|\, |w| \text{ est pair}\}$, *étant donné un alphabet fini* X. *Combien de règles a la grammaire ?*
3. $\{a^n b^m \,|\, n \leq m \leq 2n\}$
4. *Les appels d'une fonction* `f` *sachant que :*
   — `f` *renvoie un* `int` *et prend trois* `int` *en argument,*
   — *on peut appeler une fonction* `g` *qui renvoie un* `int` *et prend deux* `int` *en argument,*
   — *on peut appeler une fonction* `h` *qui renvoie un* `int` *et ne prend pas d'argument,*
   — *on peut utiliser les opérateurs arithmétiques habituels.*
5. *Les appels d'une fonction* `f` *sachant que :*
   — `f` *renvoie un* `int` *et le nombre d'arguments est indéfini,*
   — *on peut appeler une fonction* `g` *qui renvoie un* `int` *et prend un nombre non défini d'arguments,*
   — *on peut utiliser les opérateurs arithmétiques habituels.*

▸ **Exercice 3.** *Adapter l'analyseur du TD 3, exercice 1, à la grammaire de l'exercice 2.3. Compiler et tester. L'analyseur syntaxique accepte-t-il tous les mots du langage ?*

▸ **Exercice 4.** *Prouver que chacune des grammaires suivantes est ambiguë :*

1) $\begin{cases} P & \to & SbS \\ S & \to & \varepsilon \mid Sa \mid Sb \end{cases}$ *2)* $\begin{cases} S & \to & \textit{if } C \textit{ then } S \textit{ else } S \mid \textit{if } C \textit{ then } S \mid \textit{instructions} \\ C & \to & \textit{conditions} \end{cases}$

3) $B \to (\ B\ ) \mid B \textit{ et } B \mid B \textit{ ou } B \mid \textit{non } B \mid vrai \mid faux$

▸ **Exercice 5.** *Dans la grammaire de l'exercice 4.3,*

1. *quelles règles créent une récursivité à gauche ?*
2. *éliminer la récursivité à gauche de la grammaire.*

▸ **Exercice 6.** *Supprimer la récursivité à gauche (directe et indirecte) de la grammaire suivante :*

$$\begin{cases} S & \to & Ba \mid Sa \mid SSa \mid aa \\ A & \to & Ab \mid Sb \mid b \mid bb \mid Aaa \mid Abba \\ B & \to & Ac \mid Sc \mid Bc \mid \varepsilon \end{cases}$$

Indication : on peut traiter les non-terminaux dans l'ordre S, A puis B.

▸ **Exercice 7.** *Montrer qu'il existe une grammaire qui engendre le complémentaire dans $\{a,b\}^*$ de $\{ww | w \in \{a,b\}^*\}$.*

Indications :

— *le complémentaire est obtenu par l'union de l'ensemble des mots de longueur impaire avec celui des mots obtenus par la concaténation d'un mot avec un a quelque part et d'un mot de même longueur avec un b au même endroit, ou l'inverse ;*

— *ce dernier ensemble est aussi l'ensemble des mots obtenus par la concaténation d'un mot avec un a au milieu et d'un mot avec un b au milieu, ou l'inverse.*