COMPILATION TD n° 5

DEVAN SOHIER

Exercice 1 : écriture de grammaires

Ecrivez des grammaires dont les langages sont :

(1) $L_0 = \{a^n b^n / n \in \mathbb{N}\}$;
(2) $L_1 = \{a^{n+1} b^n / n \in \mathbb{N}\}$;
(3) $L_2 = \{a^n b c^n / n \in \mathbb{N}\}$;
(4) $L_3 = \{a^n b^m c^m d^n / m, n \in \mathbb{N}\}$;
(5) $L_4 = \{a^n b^n c^n / n \in \mathbb{N}\}$.

Pour chacune de ces grammaires, vous engendrerez un mot (non-trivial) de son langage.

Exercice 2 : instructions "élémentaires"

Les objets ayant potentiellement une valeur dans le langage de programmation que nous considérons sont les constantes, les variables, les appels à des fonctions, et les opérations arithmétiques appliquées à ces objets. Ecrivez une grammaire engendrant tous les objets dotés d'une valeur. Levez son ambiguïté.

Une instruction dans ce langage peut être soit une affectation, soit un appel à une fonction, soit une conditionnelle ou une boucle. Ecrivez une grammaire décrivant ces deux premiers types d'instructions.

Ecrivez une grammaire engendrant des listes d'instructions.

Exercice 3 : conditionnelles et boucles

Ecrivez une grammaire décrivant les conditions d'un langage de programmation : cette grammaire inclura les opérateurs logiques $\vee$, $\wedge$ et $\neg$ et permettra de construire des conditions à partir de tests $=$, $<$, $\geq \ldots$ appliqués à des opérations arithmétiques.

Transformez cette grammaire afin qu'elle ne soit plus ambiguë.

Ecrivez une grammaire engendrant les conditionnelles `si... alors...` et `si... alors... sinon...`. Remarquez que, en l'absence de `finsi`, cette grammaire est ambiguë, et levez son ambiguïté.

Ecrivez une grammaire engendrant les boucles `tantque... faire... ftq`.

Ecrivez une grammaire engendrant les boucles `pour... faire... fpour`. Ecrivez une grammaire pour la boucle `for` du C, et commentez les différences avec la boucle `pour` algorithmique.

Exercice 4 : un langage de liste

On veut écrire un langage de programmation travaillant sur les listes : une liste contient, entre parenthèse, son élément de tête et sa queue. Ses éléments peuvent être des valeurs (analysées au niveau lexical), ou des listes.

Des listes peuvent être concaténées par l'opérateur ., et l'opérateur :: décrit l'ajout en tête de la liste.

Ecrivez une grammaire engendrant de telles listes. Levez son ambiguïté.

Exercice 5 : un langage d'expressions régulières

On veut écrire un langage de programmation travaillant sur les expressions régulières.

Ecrivez une grammaire engendrant les expressions régulières. Levez son ambiguïté.

Exercice 6 : grammaires régulières

Une grammaire est dite régulière si toutes ses règles sont de la forme $A \to \omega$ ou de la forme $A \to \omega B$ avec A et B des non-terminaux (non nécessairement distincts) et $\omega \in \Sigma^*$.

Montrez que le langage défini par une grammaire régulière est régulier. Montrez que pour tout langage régulier, il existe une grammaire régulière qui l'engendre.

Indication : vous utiliserez les automates finis pour démontrer ces résultats.

Exercice 7 : langage engendré par une grammaire

On considère la grammaire : $T \to bTT|TbT|TTb|a$.

Donnez quelques mots engendrés par cette grammaire. Déterminez le langage qu'elle engendre (démontrez-le !)