

Projets de programmation

Le projet pourra être réalisé par groupes de deux étudiants. Chaque groupe devra rédiger un bref dossier de projet détaillant les algorithmes utilisés et les structures de données choisies. Ce dossier devra aussi comporter le code des programmes. Les projets donneront lieu à une soutenance avec démonstration qui aura lieu au mois de juin. *Les dossiers de projet doivent être rendus au plus tard le 6 juin.*

Sujet 1 : Algorithmique géométrique

Définir tout d'abord un type Caml permettant de représenter des figures géométriques formées de segments de droites et d'arcs de cercle. Écrire une fonction permettant de visualiser de tels objets à l'aide du module `graphics` de Caml.

Écrire une fonction permettant de tester si un point se trouve à l'intérieur ou à l'extérieur d'une figure géométrique fermée.

Écrire une fonction permettant de calculer l'enveloppe convexe d'une figure géométrique, c'est-à-dire le plus petit polygone convexe la contenant, et de la visualiser.

Écrire une fonction permettant de fusionner deux enveloppes convexes et donc de calculer l'enveloppe convexe de la réunion de plusieurs figures géométriques.

On pourra consulter [2] (chapitre 11) pour les algorithmes à utiliser.

Sujet 2 : Lambda-calcul

Écrire un évaluateur du lambda-calcul qui opère par application de la règle β suivant au moins deux stratégies : la stratégie standard et la stratégie par valeur.

L'évaluateur devra pouvoir fonctionner en pas-à-pas, c'est-à-dire en affichant à chaque étape l'expression obtenue.

On pourra travailler soit sur des λ -termes avec variables symboliques, soit sur des λ -termes avec indices (notation de deBruijn).

Dans les deux cas, on définira des fonctions de conversion permettant des visualiser les expressions dans les deux représentations.

Sujet 3 : Compilation de Caml

On partira du compilateur de Caml décrit dans [3] (chapitre 12). Une copie de ce chapitre sera fournie aux étudiants intéressés ainsi que les sources des programmes correspondants.

Le projet consiste à écrire un simulateur du code compilé incluant un mécanisme d'allocation de mémoire. Le code compilé sera représenté par un tableau d'instructions, les indices du tableau représentant les adresses des instructions. Le tas, dans lequel sont allouées les structures de données, sera représenté par un tableau de doublets.

Par rapport au simulateur décrit dans [3] (chapitre 12), il s'agit ici d'un simulateur de plus bas niveau dans lequel la gestion mémoire est réalisée directement sans utiliser celle de Caml. Si le temps le permet, on pourra aussi réaliser un récupérateur de mémoire mais ça ne fait pas partie du travail demandé.

Sujet 4 : Automates, recherche de motifs dans un texte

Le but de ce projet est la manipulation des automates finis. Il suppose connus les concepts et algorithmes les plus courants concernant les mots, les langages rationnels et les automates. Voir [1] pour des détails.

Définir d'abord des types Caml permettant de représenter les mots et les automates (non déterministes).

Écrire ensuite une fonction prenant en argument un automate, et rendant l'automate déterministe obtenu par l'algorithme de déterminisation usuel. Écrire une fonction testant l'appartenance d'un mot au langage reconnu par un automate. Implémenter enfin les opérations booléennes sur les automates : automate de l'union, de l'intersection et du complément.

Appliquer ce travail à la recherche de motifs dans un texte : écrire une fonction prenant en argument deux mots m et t , qui renvoie la première position de m dans t si m est facteur de t . Une amélioration de l'algorithme sous-jacent est présentée dans [2] (algorithme de Simon).

Sujet 5 : Les expressions rationnelles

Même pré-requis que pour le projet précédent. Ce projet peut aussi être interfacé avec le sujet 4.

Définir des types Caml pour représenter les mots, les automates et les expressions rationnelles.

Écrire une fonction prenant en argument une expression rationnelle et rendant un automate reconnaissant cette expression. Si le temps le permet, écrire aussi la fonction inverse.

Écrire une fonction convertissant une expression rationnelle en chaîne de caractères la représentant. Inversement, en utilisant les flux, écrire une fonction de conversion d'une chaîne de caractères représentant une expression rationnelle en expression rationnelle.

Sujet 6 : Grammaires algébriques

Ce sujet suppose connues le vocabulaire et les notions de base sur les grammaires et langages algébriques. Ici encore, se reporter à [1] si besoin est.

Définir des types Caml pour représenter les mots et les grammaires algébriques.

Écrire une fonction testant si un non terminal engendre le langage vide, puis une fonction testant si un non terminal peut être atteint de l'axiome de la grammaire, enfin une fonction prenant en argument une grammaire et retournant une grammaire réduite qui engendre le même langage.

Écrire de même une fonction testant si le mot vide appartient au langage engendré par un non terminal donné. On pourra écrire de manière optionnelle, une fonction prenant une grammaire en argument et rendant une grammaire propre engendrant le même langage.

Écrire une fonction prenant en argument un mot et une grammaire, supposée réduite et propre, et qui teste si le mot est dans le langage engendré par la grammaire.

References

- [1] J.-M. AUTEBERT, *Théorie des langages et des automates*, Masson, 1994.
- [2] D. BEAUQUIER, J. BERSTEL ET P. CHRÉTIENNE, *Éléments d'algorithmique*, Masson, 1992.
- [3] G. COUSINEAU ET M. MAUNY, *Approche fonctionnelle de la programmation*, Ediscience, 1995.