Compte-rendu

Mini Projet - Manipulation de termes

Groupe : Bardiot Maxime
Bovie Pierre-Edouard
Belloto Fabien
Roche Julie

Questions :

(1) *Comment fonctionnerait une fonction servant à tester la validité complète d'un terme ?*

Une telle fonction devrait vérifier que le symbole du terme soit valide pour le terme père, et que son attribut 'père' soit null, puis devrait parcourir chacun de ses arguments, et pour chacun d'eux vérifier la validité du symbol, le bon lien avec le père, ainsi qu'avec les fils et rappeler récursivement la fonction de manière à vérifier tous les niveaux de sous-termes.

Quelle serait sa complexité en fonction de a le nombre de symboles dans le terme et l le nombre moyen de lettres d'un symbole ?

Cette fonction utiliserait symbol_is_valid, dont la complexité est de l'ordre de O(l), puisque l'on parcourt tous les char du symbole, et cela pour chaque terme/sous-terme. Les vérifications des liens père/fils sont de l'ordre de O(1), donc negligeables dans cette fonction. L'appel de symbol_is_valid se ferait donc a fois, d'où a*O(l) = O(a*l).

(2) *Pourquoi l'algorithme s'arrête-il ? Par quoi borner le nombre d'égalités que l'on va devoir traiter ?*

(3) *Quelle peut être l'utilité d'un tel algorithme ?*

(4) *Donner des exemples où les résultats renvoyés par * et ** ne seraient pas des ensembles finis.*

(5) *Est-il possible de prévoir si le nombre de ré-écritures sera fini ou non ?*

Méthode de travail :

Tous les membres du groupe ont utilisé Sublime Text ainsi qu'un terminal Ubuntu pour ce projet. Pour nous partager nos codes respectifs, nous avons utilisé Git, ce qui nous semblait le plus pratique au niveau du versionnage.
Nous nous sommes réparti le travail selon l'ordre de priorité des fichiers, sstring.c ainsi que term.c ont donc été les premiers à être répartis, puis term_io.c et term_variable.c, et enfin les autres fichiers.
Une fois les fonctions répaties, nous avons les avons codé individuellement, puis nous nous sommes réunis régulièrement pour se débloquer si l'un de nous avait un problème, et se répartir les fonctions suivantes.
Pour le debuggage et les différents tests, nous avons travaillé au minimum deux par deux, pour repérer plus efficacement les erreurs et les corriger.

Répartition du travail :

<table>
<tr><th>BARDIOT Maxime</th><th>BELLOTO Fabien</th><th>BOVIE Pierre-Edouard</th><th>ROCHE Julie</th></tr>
<tr><td>Fonctions de valuate.c</td><td>Fonctions de sstring.c</td><td>Fonctions de term.c (partie term_traversal)</td><td>Fonctions de term.c (jusqu'à term_compare)</td></tr>
<tr><td></td><td colspan="2">Fonctions de term_io.c</td><td></td></tr>
<tr><td colspan="2">Fonctions de term_variable.c</td><td></td><td></td></tr>
</table>

Problèmes rencontrés :

Les problèmes majeurs que nous avons rencontré ont été des problèmes de compréhension, au niveau de certaines fonctions. Nous n'arrivions pas à comprendre leur but, ou la manière de les coder, qui aurait pu entrer en contradiction avec la documentation fournie.

Avancement du projet :

Les fichiers sstring, term, term_io, term_variable ont été codés, mais nous obtenons encore des erreurs de compilation ou d'execution selon les fichiers. Les fichiers restant sont en cours de codage à ce jour.
Nous n'avons pas encore codé d'extension.

Bilan du groupe :

Notre planning prévisionnel de codage à été légèrement décalé lors des révisions des différents contrôles continus. Nous avions prévu de terminer la première partie pour le rendu blanc, mais nous sommes en retard à ce niveau.
Nous avons renforcé nos connaissances en matières de listes et d'arbres à coder, notament au niveau des algorithmes de recherche.