IFT2035 – Travail pratique #2 – 11 novembre 2014

PROGRAMMATION FONCTIONNELLE

Marc Feeley

Le TP2 a pour but de vous faire pratiquer la programmation fonctionnelle et en particulier les concepts suivants: les fonctions récursives, la forme itérative, et le traitement de liste et d'arbres. Vous devez reproduire fidèlement le comportement du programme du TP1 et comparer l'approche fonctionnelle avec l'approche impérative.

Pour ce travail vous devez utiliser uniquement le sous-ensemble fonctionnel de Scheme (en particulier, vous ne devez pas utiliser les formes-spéciales "`set!`" et "`begin`" dans votre programme, ni les fonctions prédéfinies contenant un point d'exclamation, comme "`set-car!`"). Le but du travail étant de vous faire pratiquer les concepts de programmation fonctionnelle, n'hésitez pas à utiliser des fonctions d'ordre supérieur lorsque c'est approprié. L'élégance de votre codage est un facteur important dans l'évaluation de ce travail.

Vous avez à réaliser un programme en Scheme ainsi que rédiger un rapport contenant une analyse du programme.

---

1 Programmation

Vous devez réaliser en Scheme l'application spécifiée dans le TP1 en exploitant les forces du langage Scheme pour exprimer au mieux votre programme. La gestion du dictionnaire doit se faire avec un arbre "splay" (`http://en.wikipedia.org/wiki/Splay_tree`). Un arbre splay est un arbre binaire de recherche qui se fait balancer au fur et à mesure qu'on accède à son contenu, ce qui lui donne une meilleure complexité algorithmique qu'un arbre non-balancé.

Vous pouvez vous inspirer de l'implantation en Haskell des arbres splay suivante :

`http://hackage.haskell.org/package/TreeStructures-0.0.2/docs/src/Data-Tree-Splay.html`

La traduction de ce code en Scheme aura l'intérêt de vous familiariser avec sa syntaxe.

Tout comme le TP1, vous devez partager la mémoire des définitions existantes. En d'autre mots, la concaténation des chaînes X et Y doit être une structure avec des références vers X et Y, au lieu d'être une chaîne de caractères nouvellement allouée. Celà économise la mémoire mais rend la gestion des définitions plus complexe.

Pour simplifier et faciliter la correction, vous devez vous limiter à l'utilisation de listes pour représenter les noeuds des arbres splay et votre structure qui représente les définitions. Spécifiquement, un arbre splay vide doit être représenté par la liste vide, et un noeud interne d'un arbre splay doit être représenté par la liste `(` *g terme définition d* `)`, où g et d sont les sous-arbres splay gauche et droite, et *terme* est une chaîne de caractère, et *définition* est sa définition. Une définition est soit une chaîne de caractère, ou la paire `(` X `.` Y `)` qui représente la concaténation des définitions X et Y.

Votre programme doit être robuste. Les termes et leur définition peuvent être arbitrairement longs (votre programme ne doit donc pas imposer de limite sur leur longueur).

Votre programme Scheme doit être entièrement contenu dans un fichier dont le nom est "`tp2.scm`". Vous devez utiliser le fichier "`tp2.scm`" de la page Studium du cours comme point de départ et seulement modifier la première section. L'exécution du programme doit se faire au moyen de l'interprète `gsi` avec la commande "`gsi tp2.scm`".

2 Rapport

Vous devez rédiger un rapport qui:

1. Explique brièvement le fonctionnement général du programme (maximum de 1 page au total).
2. Explique comment les problèmes de programmation suivants ont été résolus (en 2 à 3 pages au total):
   (a) comment se fait l'analyse syntaxique d'une requête et la lecture d'une requête de longueur arbitraire
   (b) comment les dictionnaires et définitions sont représentés, et comment se font les opérations sur ces structures
   (c) comment se fait en Scheme l'affichage des réponses aux requêtes
   (d) comment se fait le traitement des erreurs
3. Compare votre expérience de développement avec le TP1 (en 1 à 2 pages au total). Combien de lignes de code ont vos programmes C et Scheme? Sans tenir compte de votre niveau de connaissance des langages C et Scheme, quels sont les traitements qui ont été plus faciles et plus difficiles à exprimer en Scheme? Indépendamment des particularités de Scheme, pour quelles parties du programme l'utilisation du style de programmation fonctionnel a-t'il été bénéfique et pour quelles parties détrimental? Pour quelles parties avez vous utilisé des récursions en forme itérative?

3 Évaluation

- Ce travail compte pour 15 points dans la note finale du cours. Indiquez vos noms clairement au tout début du programme. **Vous devez faire le travail par groupes de 2 personnes. Vous devez confirmer la composition de votre équipe (noms des coéquipiers) au démonstrateur. Si vous ne trouvez pas de partenaire d'ici quelques jours, venez me voir.**
- Le programme sera évalué sur 50% et le rapport sur 50%. Un programme qui plante à l'exécution perdra une partie importante des points, allant jusqu'à 50% si la nature de l'erreur le justifie. Vous devez donc accorder une grande importance à l'exécution et aux cas limites. Assurez-vous de prévoir toutes les situations d'erreur.
- Vous devez remettre un fichier ".tar" qui contient uniquement deux fichiers : votre rapport (qui doit se nommer "rapport.pdf") et le programme (qui doit se nommer "tp2.scm"). La remise doit se faire au plus tard à 23h55 lundi le 8 décembre sur le site Studium du cours. En supposant que vos deux fichiers sont dans le répertoire "tp2", vous pouvez créer le fichier ".tar" avec la commande "`tar cf tp2.tar tp2`".
- L'élégance et la lisibilité du code, l'exactitude et la performance, la lisibilité du rapport, et l'utilisation d'un français sans fautes sont des critères d'évaluation.