

# Projet individuel d'algorithmique-programmation : Travail à rendre

octobre 2007

Le projet est à réaliser en OCaml **individuellement**. Il sera accompagné d'un **dossier** contenant impérativement la description des choix faits, la description des types et des fonctions. Pour chaque fonction, on donnera impérativement l'interface complète (dans le code en commentaire et dans le rapport pour les fonctions présentées). Le dossier fournira également des cas de tests accompagnés des résultats attendus et retournés. Sur le site du cours figure un petit document sur ce que l'on attend dans un rapport. Consultez-le !

Le projet (dossier + copie du listing de code) est à rendre au secrétariat le **7 novembre à 16h30 au plus tard**. Le numéro du groupe, ainsi que le nom du chargé de TD, devront figurer en gros et en rouge sur la page de garde. Le fichier `.ml` contenant votre code devra être déposé électroniquement au plus tard le **7 novembre à minuit** (la procédure sera envoyée par email).

Les soutenances seront organisées une dizaine de jours après le retour des projets. Il vous faudra consulter les panneaux d'affichage et votre courrier électronique pour obtenir l'ordre de passage.

Enfin n'attendez pas pour vous mettre au travail ! Un projet se travaille dès la remise du sujet afin d'avoir le temps de laisser mûrir la solution et de poser des questions au client (dans votre cas, votre chargé de TP).

## 1 Préliminaires

On se propose de coder une version naïve d'un interpréteur de règles de réécriture du premier ordre.

On définit l'ensemble des *termes* construits sur une *signature*. Une signature est une liste de couples `string * int` où la chaîne représente un *symbole* (c'est-à-dire un nom de fonction) et l'entier associé représente son *arité* (c'est-à-dire son nombre d'arguments).

L'ensemble des termes est le plus petit ensemble contenant des *variables* et tel que si  $t_1, \dots, t_n$  sont des termes et  $f$  est un symbole d'arité  $n$  dans la signature, alors  $f(t_1, \dots, t_n)$  est un terme. On utilisera le type somme `term` constitué des deux constructeurs **Var of string** et **Term of (string \* term list)**. Par exemple :

```
Term("+", [Var "x" ; Term("+", [Var "y" ; Term("2", [])])])
```

représente le terme  $x + (y + 0)$  que nous appellerons  $t$  dans la suite. Un terme est dit *linéaire* s'il ne contient qu'une seule occurrence de chacune de ses variables.

**Question 1.** — Définir une fonction qui *teste* si un terme est bien formé, c'est-à-dire si les fonctions ont bien toutes le bon nombre d'argument.

**Question 2.** — Définir une fonction qui *teste* si un terme est linéaire.

Une substitution  $\sigma$  est une application des noms de variables vers les termes. Par exemple si on applique au terme  $t$  la substitution  $\sigma$  qui à la variable de nom  $x$  associe le terme 3 (ce qu'on notera  $t\sigma$ ) on obtient le terme  $3 + (y + 0)$ .

**Question 3.** — Définir une fonction qui, sur la donnée d'une substitution et d'un terme, retourne le résultat de l'application de la substitution au terme.

On dit qu'un terme  $s$  *filtre* un terme  $t$  s'il existe une substitution  $\sigma$  telle que  $s\sigma = t$ .

Nous serons amenés à considérer un symbole particulier, le « trou », noté  $\square$ . Un terme qui contient des trous est un *contexte*.

## 2 Réécriture du premier ordre

La réécriture de termes est un formalisme de calcul sur les termes *bien formés* qui fonctionne par remplacement de termes par des termes égaux mais dans *un seul sens*. On considère ici la relation de réécriture définie par un ensemble (système) fini de couples (les *règles*) dont **le membre gauche est linéaire**. On notera dans la suite de l'énoncé  $l \rightarrow r$  le couple de termes  $(l, r)$ .

Les systèmes seront représentés par des listes de couples de termes.

Le calcul fonctionne de la façon suivante : si un terme  $t$  est un membre gauche de règle, on peut le remplacer par le membre droit correspondant. Jusque là pas de problème. On peut aller plus loin et faire ce qu'on appelle la *clôture par substitution* : si  $t = l\sigma$  pour un couple  $l \rightarrow r$  alors on peut le remplacer par  $r\sigma$ . OK ? Bon alors on va encore plus loin et on fait la *clôture par contexte* en disant que si un terme  $t$  contient  $l\sigma$  alors on le remplace par  $t$  où  $l\sigma$  est devenu  $r\sigma$ . C'est ce qu'on appelle un pas de réduction.

Par exemple pour le système réduit à une seule règle  $\{x+0 \rightarrow x\}$ , notre terme  $t$  se réduit en un pas à  $x+y$  : la substitution convenable est celle qui associe  $y$  à  $x$  et le remplacement s'effectue dans le contexte  $x + \square$ .

Remarquons qu'il est possible que différents remplacements soient possibles à un même endroit et même à plusieurs endroits dans un terme...

**Question 4.** — Définir une fonction qui, sur la donnée d'un système et d'un terme, retourne l'un de ses réduits en un pas par ce système. S'il n'en existe pas la fonction doit lever l'exception **Normal\_form**.

**Question 5.** — Définir une fonction qui, sur la donnée d'un système et d'un terme, retourne l'un de ses réduits *les plus intérieurs* (c'est-à-dire qui ne contiennent pas strictement d'instance de membre gauche de règle) en un pas par ce système. S'il n'en existe pas la fonction doit lever l'exception **Normal\_form**.

**Question 6.** — Définir une fonction qui, sur la donnée d'un système et d'un terme, retourne la liste éventuellement vide de tous ses réduits en un pas.

**Question 7.** — Reprendre les questions 4, 5 et 6 mais cette fois en donnant en plus à vos fonctions un entier qui représente le nombre de pas à effectuer.

On remarque que travailler directement sur le système en tant que liste de couple est peu efficace. On peut penser à définir une structure qui permet de rassembler tous les membres de gauche afin d'en écarter plusieurs en même temps au fur et à mesure du parcours du terme. Cette structure est arborescente, les arêtes sont étiquetées par les noms de symboles et les nœuds contiennent la liste des membres de droite encore envisageables et les substitutions associées.

**Question 8.** — Reprendre toutes les questions précédentes avec cette nouvelle structure à la place du système passé en argument.