DEUG MIAS 2e année
Projet de programmation fonctionnelle
Le compte est bon
Indication pour la fonction nouvelles-expressions

Pierre Boulet Jean-Marc Nicod Frédéric Vivien

Rappel de la question :

Question. *On veut pouvoir tester toutes les solutions. Pour cela, à chaque étape, il faut construire la liste de toutes les listes d'expressions déductibles de la liste d'expressions courante. Construisez une fonction nouvelles-expressions qui prend une liste d'expressions et retourne la liste de toutes les listes d'expressions construites par remplacement de deux expressions de la liste originale par l'application d'une opération arthmétique à ces deux expressions.*

Note. Toutes les opérations sont symétriques : en effet, l'addition et la multiplication commutent, on soustrait toujours le plus petit nombre du plus grand et on divise le plus grand nombre par le plus petit et seulement si la division tombe juste.

Une première chose à faire est de découper le travail en deux étapes :

1° construire une liste de toutes les paires d'expressions possibles à partir de la liste initiale ;

2° pour chacune de ces paires, former les expressions résultant des quatres opérations appliquées.

La deuxième étape ne pose pas de problèmes insurmontables, par contre, la première est plus compliquée. Nous allons maintenant détailler son fonctionnement.

Une première chose à voir est qu'on doit non seulement générer toutes les paires possibles, mais aussi conserver pour chaque paire les expressions qui n'ont pas été sélectionnées. En effet, on en a besoin pour pouvoir continuer le calcul (appel récursif de *resout*).

Une solution consiste à décomposer cette fonction en étapes plus simples qui seront illustrées à chaque fois sur un exemple. La fonction qu'on cherche à écrire sera nommée *explose* et son fonctionnement est le suivant :

(*explose* '(5 (+ 2 6) (∗ 3 10) 8))

$\Longrightarrow$

```
((5 8 (+ 2 6) (* 3 10))
 (5 (* 3 10) (+ 2 6) 8)
 (5 (+ 2 6) (* 3 10) 8)
 ((+ 2 6) 8 (* 3 10) 5)
 ((+ 2 6) (* 3 10) 8 5)
 ((* 3 10) 8 5 (+ 2 6)))
```

Si on observe l'exemple précédent, on remarque que les deux premiers éléments de chaque liste forment toutes les paires possibles d'éléments de la liste de départ. Leur ordre d'apparition n'est pas quelconque : on a d'abord toutes les paires commençant par la première expression (ici, 5), puis celles commençant par la deuxième (ici, (+ 2 6)) et ainsi de suite. C'est-à-dire que la méthode utilisée énumère les listes commençant par une expression suivie par chacune des expressions suivantes.

Pour réaliser celà, on peut utiliser une fonction *suivants* qui prend comme argument la liste (de longueur l) d'expressions privée de ses premiers éléments (1 ou plus) et qui retourne une liste de l listes de l expressions commençant toutes par une expression différente. Sur l'exemple :

(*suivants* '((+ 2 6) (∗ 3 10) 8))

⟹

((8 (+ 2 6) (∗ 3 10))
((∗ 3 10) (+ 2 6) 8)
((+ 2 6) (∗ 3 10) 8))

(*suivants* '((∗ 3 10) 8))

⟹

((8 (∗ 3 10))
((∗ 3 10) 8))

(*suivants* '(8))

⟹

((8))

Une deuxième fonction ajoutera le premier élément en tête des listes retournées par *suivants*. C'est le rôle de la fonction *debut* (qui appelle *suivants*). Sur l'exemple :

(*debut* '(5 (+ 2 6) (∗ 3 10) 8))

⟹

((5 8 (+ 2 6) (∗ 3 10))
(5 (∗ 3 10) (+ 2 6) 8)
(5 (+ 2 6) (∗ 3 10) 8))

(*debut* '((+ 2 6) (∗ 3 10) 8))

⟹

(((+ 2 6) 8 (∗ 3 10))
((+ 2 6) (∗ 3 10) 8))

(*debut* '((∗ 3 10) 8))

⟹

(((∗ 3 10) 8))

Ces listes ne répondent pas encore complètement à la question : il faut en effet leur ajouter à la fin la liste des premiers éléments qui ont été supprimés. C'est la dernière étape qui est faite dans la fonction *explose* (qui appelle *debut*). Le résultat sur l'exemple a déjà été présenté.

Le schéma de calcul est donc trois fonctions imbriquées : *explose* qui appelle *debut* qui appelle *suivants*. On peut représenter le résultat comme suit :

((*5* 8 (+ 2 6) (∗ 3 10))
(*5* (∗ 3 10) (+ 2 6) 8)
(*5* (+ 2 6) (∗ 3 10) 8)
(*(+ 2 6)* 8 (∗ 3 10) **5**)
(*(+ 2 6)* (∗ 3 10) 8 **5**)
(*(∗ 3 10)* 8 **5 (+ 2 6)**))

avec en gras ce qui est ajouté par *explose*, en italique ce qui est ajouté par *debut* et en caractères normaux ce qui est généré par *suivants*.