Programmation Fonctionnelle I, Printemps 2017 – TD11

http://deptinfo.unice.fr/~roy

Exercice 11.1 Complétez le code la fonction `(exec-HP codeHP)` du cours page 6. Cette fonction prend une liste `codeHP` contenant une séquence de touches pour une calculette HP, produite par le traducteur `(scheme->hp A)` du cours 10 page 14. Elle exécute **itérativement** ces touches avec une pile `P`. A la fin de l'exécution, le résultat est au sommet de la pile. Exemple :

`(exec-HP '(5 enter 2 enter - 3 enter *)) → 9`

Exercice 11.2 Programmez une fonction itérative `(nb-op A op)` prenant un arbre binaire `A` et un symbole `op`, et retournant le nombre d'apparitions de l'opération `op` dans l'arbre `A`. Par exemple `(nb-op '(+ (* x (+ x 1)) (+ y x)) '+) → 3` car il y a trois additions.

Exercice 11.3 a) Programmez la fonction `(postfixe->arbre L)` prenant une liste `L` représentant le parcours postfixe plat d'un arbre `A`, et retournant l'arbre `A` [cours page 9]. Vous procèderez à une analyse itérative avec pile…
b) Utilisez cette fonction `postfixe->arbre` pour obtenir une autre solution à l'exercice 1.

Programmation Fonctionnelle I, Printemps 2017 – TP11

http://deptinfo.unice.fr/~roy

Simplification et Dérivation Formelles

Une erreur courante consiste à simplifier (- 0 y) en… je ne sais pas quoi ? Il faut bien voir que (- 0 y) n'est pas simplifiable, au contraire de (- y 0). Tous nos arbres sont en effet binaires !

Exercice 11.1 Complétez le code du **simplificateur** [cours pages 13-14] en implémentant chaque spécialiste. Faites votre possible pour procéder au maximum de simplifications, vous n'aurez pas un simplificateur complet de toutes façons !…

`(simplif '(+ (* x (- 5 (+ 3 2))) (/ (- y 0) (/ z z)))) → y`

La Dérivation Formelle

Exercice 11.2 Complétez le code du **dérivateur** [cours 11 pages 15-16] en implémentant chaque spécialiste. Le résultat doit être simplifié puisque `diff` appelle `simplif`. Vous ne traiterez que les quatre opérations binaires de base :

```
(diff '(+ (* x 3) (- (/ x y) 1)) 'x) → (+ 3 (/ 1 y))
(diff '(/ y x) 'y) → (/ 1 x)
(diff '(/ y x) 'x) → (/ (- 0 y) (* x x))
```

Exercice 11.3 Afin que vous n'ayez quand même pas trop d'illusions sur la puissance de votre simplificateur (mais enfin, qui sait ?...), demandez à contempler la dérivée 5ème de `x/(x+1)`. Et remerciez les logiciels comme ***Mathematica*** de n'avoir aucun état d'âme à ce sujet :

```
In[1]:= Simplify[D[x / (x + 1), {x, 5}]]

Out[1]= 120 / (1 + x)^6
```

Exercice complémentaire

Exercice 11.4 Augmentez la puissance du logiciel en autorisant des arbres de racine `expt`. Il s'agira de noeuds binaires `(expt A n)` où `A` est un arbre quelconque et `n` une constante entière ≥ 0 [erreur sinon]. Augmentez en conséquence les codes du simplificateur et du dérivateur. *Pourquoi poser cette contrainte sur n ?...*