Programmation : TP 9

Objectifs du TP : définition de type, équivalence entre récursivité de type et récursivité de calcul.

[O'Caml, en utilisant le *toplevel*]

1. Définir le type `expr` pour la représentation d'expressions arithmétiques. On pourra considérer qu'une expression arithmétique est soit un nombre (un nombre flottant), soit une variable (un identificateur représenté par une `string`), soit une somme ou un produit de deux expressions, soit l'opposé ou l'inverse d'une expression. Représenter avec ce type l'expression suivante :

$$-\frac{1}{2}(1+3x)$$

2. Écrire une fonction

   `eval : expr -> float`

   permettant d'évaluer une expression ne contenant pas de variables pour lesquelles on déclenchera une erreur avec la fonction `failwith`[1]

3. Modifier (en la recopiant) la fonction précédente pour écrire une fonction

   `eval2 : (string -> float) -> expr -> float`

   prenant en argument une substitution pour les variables et une expression et évaluant cette dernière pour cette substitution :

   ```
   let subst = fun v ->
     match v with
      "x"-> 1.
     | _ -> failwith "variable non correcte";;
   ```

   `eval2 subst` évalue $-\frac{1}{2}(1+3x)$ en -2.0.

4. Écrire une fonction

   `derive : expr -> string -> expr`

   qui dérive une expression par rapport à une variable donnée.

5. Écrire l'itérateur pour le type `expr`.

6. En utilisant l'itérateur, écrire une fonction

   `simplifie: expr -> expr`

   qui simplifie une expression donnée en utilisant les règles suivantes :

$$\forall e\ 0+e=e+0=e,\ \ \forall e\ 1*e=e*1=e,\ \ \forall e\ 0*e=e*0=0,\ \ -0=0$$

[1] La fonction `failwith : string -> 'a` est une fonction qui prend une chaîne de caractères en argument (typiquement un message d'erreur) et qui lève une *exception*. Cette exception stoppe l'exécution du programme en cours et la chaîne de caractères est affichée.