TP7 et TP8 Programmation fonctionnelle: évaluateur miniML

L3 Informatique Nancy

22 mars et 5 avril 2022

Pour ces deux séances de TP, on vous propose, comme annoncé lors du CM précédent, de travailler sur un évaluateur pour un langage fonctionnel. Ce langage est juste une variante un peu simplifiée de OCaml qu'on appellera *miniML*, et l'évaluateur sera codé en OCaml.

Pour que vous ne partiez pas de rien et pour vous éviter de vous poser des problèmes sur des questions un peu éloignées du sujet (*lexing/parsing*...), les bases de l'évaluateur ont déjà été écrites par M. Jeandel, et sont à télécharger sur Arche.

Une fois l'archive `miniML.zip` extraite, vous pouvez lancer un environnement qui contient toutes les fonctions grâce à la recette *dune* déjà écrite (*dune* est un système de build pour des projets OCaml, il est possible qu'il ne soit pas installé chez vous, dans ce cas-là, exécutez `opam install dune`). Pour ce faire, ouvrez un terminal dans le répertoire extrait et tapez :

```
dune utop .
```

Si tout va bien, vous êtes maintenant dans un environnement `utop`. Commencez par ouvrir les modules nécessaires :

```
open Miniml.Eval;;
open Miniml.Expr;;
```

Exemple d'évaluation d'une expression : "**let** x = 1 **in** x + 1" |> expr_from_string |> eval_ocaml;;, ce qui devrait donner : − : expr = Num 2.

Le fichier eval.ml contient le squelette de l'évaluateur, il vous faut remplir les trous (todo) et implémenter les différentes fonctionnalités (en modifiant subst/eval et string_from_expression).

1. Familiarisez vous avec eval_ocaml et expr_from_string.
2. Implémentez le ifthenelse (If, Eq, Bool).
3. Écrivez une expression incluant une fonction récursive "times" (qui prend deux entiers et qui calcule leur multiplication, en utilisant l'opération primitive + ainsi que la conditionnelle), que vous passerez à l'évaluateur. Cela vous permettra de vérifier le bon fonctionnement de la conditionnelle que vous venez d'implémenter.
4. Implémentez les tuples (Pair, Fst, Snd).
5. Implémentez les listes (Cons, Nil et Match).
6. Implémentez la fonction Tee, qui prend en argument une expression x, qui l'affiche avec un print_string et string_from_expr, puis qui renvoie x (ne pas oublier de modifier le match/with de App(x,y) pour gérer le nouveau cas où x c'est Tee).
7. Implémentez le call by name (App) et tester en utilisant la fonction Tee la différence de comportement avec le call by value.
8. Implémentez les exceptions (Try et Raise).