Rapport projet 2, où l'on parle de Fouine

Monsieur Chekroun et Monsieur Hohnadel

1 Présentation

Nous avons programmé un interpréteur, un compilateur et une machine à pile pour le langage Fouine, pas l'animal [2].

2 Organisation du code

Le code est structuré de la manière suivante :
— `main.ml` qui est le "chef d'orchestre" de notre chef-d'œuvre et gère les options passées à l'exécutable,
— `type.ml` dans lequel nous définissions les types,
— `eval.ml` qui interprète le code Fouine,
— `machine.ml` qui compile et fait tourner la machine à pile,
— `affiche.ml` qui affiche les programmes,
— `purity.ml` qui s'occupe de la ségrégation des branches en maintenant les "pures" à part,
— `lexer.mll` qui définit les tokens,
— `parser.mly` qui classe les tokens et traduit l'entrée en arbre.

3 Description du fonctionnement de Fouine

3.1 Arguments

— `-debug` qui affiche le code interprété par Fouine,
— `-machine` qui permet d'exécuter les branches "pures" sur la machine à pile,
— `-stackcode` qui affiche le code compilé pour la machine à pile,
— `programme` qui se fait traiter.

3.2 Objets traités par Fouine

— Entiers
— Fonctions
— Fonctions récursives
— Tests booléens
— Couples

3.3 Les *guys* (les types, mais à l'américaine)

On annote les variables avec un type à l'américaine (un *guy*), défini par le type :

— `WE` le type *whatever* par défaut,
— `INT` le type entier,
— `FUN of guy*guy`, où *guy* est un type (mais à l'américaine), le type fonction,
— `TUP of guy*guy`, où *guy* est un type (mais à l'américaine), le type couple.

3.4 Arguments de fonctions

— `Var of bool*string*guy` une variable caractérisée par un booléen d'appartenance à un arbre pur, un nom et un *guy* qui est son type (mais à l'américaine),
— `Couple of expr*expr` qui désigne un tuple ; le parser s'assure que ce sont d'autres arguments de fonctions en arguments, ie que les `expr` sont des arguments de fonctions.

3.5 Code de la machine à pile

— `C of int` Constantes entières,
— `A, S, M` les opérateurs respectifs d'additions, soustractions et multiplications,
— `LET of string` affectation du `let`,
— `ACC of string` lecture dans l'environnement,
— `ENDLET` fin de portée du `let`,
— `PRINT` affichage de la tête de pile, sans la modifier.

4 Critique

Tout d'abord, la mémoire de notre tableau de référence est finie : nous sommes limités à 99 références, soit moins que le nombre de langages mentionnés dans le titre de ce livre [1].

Les *guys* qui sont des types (mais à l'américaine), étant les derniers arrivés, peuvent encore être sujets à quelques bugs, même si tous nos tests se sont avérés concluants.

Il nous semble ne rien avoir à ajouter.

Références

[1] Peter J. Landin. The next 700 programming languages. *Commun. ACM*, 9(3) :157–166, 1966.

[2] Wikipedia. Fouine — wikipedia, l'encyclopedie libre, 2018. [Enligne].