Les lieurs en programmation
Concepts et implémentations

Pierre-Évariste Dagand
Équipe Whisper

Frédéric Peschanski
Équipe APR

Le concept de lieur est fondamental en théorie des langages de programmation. En programmation procédurale, il est incarné par la notion de procédure qui *lie* des variables sur toute la portée de la procédure. Appeler une procédure consiste à *substituer* ces variables par des arguments concrets. En programmation fonctionnelle, les lieurs sont des citoyens de première classe : c'est la notion de lambda abstraction.

Pour les concepteurs de langages de programmation, la correction et l'efficacité de l'implémentation des lieurs est donc cruciale. D'un point de vue théorique, on dispose d'une bonne compréhension conceptuelle de ces objets (sous la forme de monade [Ahrens, 2015], par exemple). Cependant, d'un point de vue pratique, la plupart des langages de programmation dépendent d'une implémentation ad-hoc des lieurs [Bernardy and Pouillard, 2013, Westbrook et al., 2011, Cave and Pientka, 2012].

On propose ici un exercice de zoologie comparée. Il s'agira dans un premier temps d'extraire l'implémentation des lieurs de divers langages de programmation (OCaml, Haskell, Coq, etc.). Il conviendra ensuite de concevoir un banc d'essai permettant de comparer ces diverses implémentations. On souhaiterait finalement abstraire cette diversité dans un cadre conceptuel unifié offert, par exemple, par la présentation monadique.

Profil recherché : Les étudiants devront être intéressé par les aspects pratiques de l'implémentation des langages, de naviguer à travers des programmes complexes et d'en extraire les concepts principaux. Ils seront en mesure de s'adapter à des langages de programmation variés (OCaml, Haskell, C, *etc.*). Ils seront également en mesure d'effectuer et d'interpréter des tests de performance.

Encadrement : Ce stage aura lieu au LIP6, au sein des équipes Whisper et APR (Upmc). Il sera encadré par Pierre-Évariste Dagand et Frédéric Peschanski.

Références

B. Ahrens. Modules over relative monads for syntax and semantics. *Mathematical Structures in Computer Science*, pages 1–35, 1 2015. doi :10.1017/S0960129514000103.

J.-P. Bernardy and N. Pouillard. Names for free : Polymorphic views of names and binders. In *Symposium on Haskell*, pages 13–24, 2013. doi :10.1145/2503778.2503780.

A. Cave and B. Pientka. Programming with binders and indexed data-types. In *Symposium on Principles of Programming Languages*, pages 413–424, 2012. doi :10.1145/2103656.2103705.

E. Westbrook, N. Frisby, and P. Brauner. Hobbits for haskell : A library for higher-order encodings in functional programming languages. In *Symposium on Haskell*, pages 35–46, 2011. doi :10.1145/2034675.2034681.