Proposition de stage M1 printemps 2010

Preuve formelle d'un algorithme de calcul de point fixe

Lieu du stage :
Équipe ProVal / INRIA Saclay – Île-de-France
4 rue Jacques Monod / 91893 Orsay Cedex

Durée du stage :
3 mois (avril–juin 2010)

Encadrants :
Jean-Christophe Filliâtre (`filliatr@lri.fr`)
Johannes Kanig (`kanig@lri.fr`)

Sujet

Le contexte de ce stage est celui de la preuve formelle de programmes fonctionnels et plus spécifiquement celui de l'outil Who développé dans l'équipe ProVal par Johannes Kanig [1, 2].

L'objectif de ce stage est de faire la preuve formelle d'un programme Objective Caml réalisant un calcul de point fixe [4]. Ce programme calcule le point fixe d'un ensemble d'équations $\vec{X} = F(\vec{X})$ et se présente sous la forme d'une fonction

$$\texttt{lfp} : (V \to (V \to D) \to D) \to (V \to D)$$

où V désigne le type des variables et D le type de leurs valeurs. L'algorithme utilisé calcule le point fixe « à la demande » et maintient, de manière invisible, des structures de données impératives. La difficulté de la preuve formelle tient donc en particulier à la combinaison de fonctions d'ordre supérieur (`lfp` est d'ordre 3) et de traits impératifs.

Le système Who est précisément un système permettant la preuve formelle de programmes d'ordre supérieur avec effets de bord. Le premier objectif de ce stage sera donc de spécifier le code de calcul de point fixe, en effectuant notamment la modélisation des traits du langage Caml utilisés dans ce programme. Le second objectif sera de réaliser la preuve proprement dite, en utilisant les outils de démonstration supportés par l'outil Who, dont en particulier l'assistant de preuve Coq.

Références

[1] Johannes Kanig and Jean-Christophe Filliâtre. *Who : A Verifier for Effectful Higher-order Programs*. In ACM SIGPLAN Workshop on ML, Edinburgh, Scotland, UK, August 2009. `http://www.lri.fr/~filliatr/ftp/publis/wml09.pdf`

[2] Johannes Kanig. *Who – A verification condition generator for imperative higher-order programs*. `http://www.lri.fr/~kanig/who.html`

[3] François Pottier. *Lazy Least Fixed Points in ML*. Under consideration for publication in Journal of Functional Programming. `http://pauillac.inria.fr/~fpottier/publis/fpottier-fix.pdf`

[4] François Pottier. *Code source du module de calcul de point fixe [3]*. `http://pauillac.inria.fr/~fpottier/fix/fix.html.en`