Proposition de TER M1

Un compilateur certifié pour le λ-calcul

Sujet

Historiquement, il y a eu de nombreuses approches pour l'implémentation des langages de programmation fonctionnels. Dans ce TER, nous allons en particulier nous intéresser à la CAM [1] (« Categorical Abstract Machine »), qui est une machine abstraite pour l'implémentation des langages de programmation fonctionnels avec liaison statique des variables, c'est-à-dire des langages basés sur le λ-calcul.

La CAM est une machine intéressante car elle peut être vue comme la synthèse de 3 approches pour l'implémentation des langages de programmation fonctionnels :

- Le formalisme de De Bruijn pour éliminer les problèmes dus à l'α-conversion [2] ;
- La SK-reduction machine de Turner [5] ;
- La SECD machine de Landin [3].

L'objectif de ce TER est d'écrire un compilateur pour le λ-calcul pur (sans types) vers la CAM, puis de démontrer que ce compilateur est correct. Le compilateur est correct si une (β)-réduction dans le λ-calcul correspond à une réduction de la CAM avec le même résultat. On peut simplifier cette définition lorsque le λ-terme à réduire est terminant en disant qu'après normalisation, le λ-terme et le programme (CAM) compilé donnent le même résultat.

Pour faire la preuve de correction, l'idée est de mécaniser la preuve en utilisant l'outil d'aide à la preuve Coq [4]. Il y aura plusieurs étapes de formalisation à réaliser dans Coq :

1. Formaliser le λ-calcul et sa règle de réduction ;
2. Formaliser la CAM et ses règles de réduction ;
3. Écrire le compilateur du λ-calcul vers la CAM ;
4. Démontrer la correction de ce compilateur.

Travail à réaliser

- Formaliser en Coq le λ-calcul (le langage source) et la CAM (le langage cible), ainsi que leurs règles de réduction ;
- Écrire en Coq le compilateur du λ-calcul vers la CAM (fonction) et démontrer sa correction (preuve mécanisée).

Prérequis

- Aucun prérequis en programmation fonctionnelle n'est nécessaire, mais il faudra avoir un goût prononcé pour le paradigme fonctionnel de manière générale ;
- Aucun prérequis en Coq n'est nécessaire. Une petite formation Coq sera faite pendant le TER. Être en train de suivre l'UE HMIN229 pourra être un plus.

Remarques additionnelles

L'encadrement du TER sera réalisé par :

- David Delahaye (Université de Montpellier, LIRMM, David.Delahaye@lirmm.fr).

Références

[1] G. Cousineau, P.-L. Curien, and M. Mauny. The Categorical Abstract Machine. *Science of Computer Programming*, 8(2) :173–202, Apr. 1987.

[2] N. G. de Bruijn. Lambda Calculus Notation with Nameless Dummies, a Tool for Automatic Formula Manipulation, with Application to the Church-Rosser Theorem. *Indagationes Mathematicae (Proceedings)*, 75(5) :381 – 392, June 1972.

[3] P. J. Landin. The Mechanical Evaluation of Expressions. *The Computer Journal*, 6(4) :308–320, Jan. 1964.

[4] The Coq Development Team. *Coq, version 8.10.1.* Inria, Oct. 2019. http://coq.inria.fr/.

[5] D. A. Turner. A New Implementation Technique for Applicative Languages. *Software – Practice and Experience (SPE)*, 9(1) :31–49, Jan. 1979.