Vers un langage typé pour la programmation modulaire

Danny Willems

Résumé

La programmation modulaire est un principe de développement consistant à séparer une application en composants plus petits appelés *modules*. Le langage de programmation OCaml contient un langage de modules qui permet aux développeurs d'utiliser la programmation modulaire. Dans ce langage de modules, un module est un ensemble de types et de valeurs, les types des valeurs pouvant dépendre des types définis dans le même module. OCaml étant un langage fortement typé, les modules possèdent également un type, appelé dans ce cas *signature*.

Bien que les modules soient bien intégrés dans OCaml, une distinction est faite entre le langage de base, contenant les types dits « primitifs », comme les entiers, les chaînes de caractères ou les fonctions, et le langage de modules. En particulier, le terme *foncteur* est employé à la place de *fonction* pour parler des fonctions prenant un module en paramètre et en renvoyant un autre.

D'un autre côté, dans les types de base d'OCaml se trouvent les *enregistrements*. Ces derniers peuvent être interprétés comme des listes de couples *(label, valeur)*, et ressemblent aux modules. Cependant, les deux différences majeures sont la possibilité de définir des types dans un module ainsi que d'avoir des champs mutuellement dépendants.

Ce mémoire vise à donner, dans un premier temps et après avoir défini les notions nécessaires, un calcul typé, DOT, dans lequel le langage de modules est confondu avec le langage de base grâce aux enregistrements. Cette unification implique que les modules (et, *in fine*, les foncteurs) sont des citoyens de première classe, c'est-à-dire que nous pouvons les manipuler comme tout autre terme, ce qui n'est pas le cas actuellement en OCaml. Dans un second temps, ce travail propose un algorithme de typage et de sous-typage, une implémentation OCaml de ces derniers ainsi qu'un langage de surface qui nous permet d'écrire des programmes DOT.

Mots-clefs : types, λ-calcul, types dépendants, DOT, programmation modulaire, OCaml, enregistrements.