Implémenter la Théorie des Types Dépendants

Pierre-Évariste Dagand
Équipe Whisper

Frédéric Peschanski
Équipe APR

Les langages de programmation sont divisés en deux grandes familles : les langages dynamiquement typés, comme Python ou Ruby, et les langages simplement typés, comme OCaml ou Java. Un langage simplement typé est équipé d'un vérificateur de type qui garantie, *durant la compilation*, l'absence de certaines classes d'erreur à l'éxecution. Cependant, l'analyse de type étant statique, celle-ci est nécessairement restreinte : des programmes correctes peuvent être rejetés par le vérificateur de type. Les types dynamiques offrent une plus grande flexibilité : les types sont manipulés *durant l'exécution* afin d'éviter des erreurs grossières, tel qu'additionner un nombre avec une chaîne de caractère. Ainsi, des erreurs qu'un système de type aurait pu détecter risquent à tout moment d'interrompre l'exécution d'un programme.

La *théorie des types dépendants* offre un compromis intéressant entre ces deux extrêmes. Dans ce formalisme, les types sont eux-mêmes des programmes : il est alors possible, *durant le développement d'un programme*, d'écrire des types charactérisant très précisement son évolution au cours de l'exécution. Ces types sont ensuite vérifiés *durant la compilation*, la validité du typage garantissant l'absence d'erreur à l'exécution. La flexibilité du système de type nous permet donc d'accepter plus de programmes sans pour autant ralentir ou déstabiliser l'exécution par des vérifications dynamiques de type.

En pratique, la théorie des types dépendants forme le noyau de l'assistant de preuve Coq, dans lequel sont développés des logiciels certifiés ainsi que des résultats mathématiques méchaniquement vérifiés. Cette architecture permet à Coq de satisfaire le *critère de De Bruijn* : une preuve développée dans Coq se réduit en un témoin de preuve dans la théorie des types dépendant, témoin qui peut être vérifié indépendamment de Coq. À l'heure actuelle, cette fonctionnalité est partiellement fournie par l'outil `coqchk`. Sur le long terme, on souhaiterait effectuer le *bootstrap* de ce vérificateur de type et en vérifier la correction.

Afin de progresser dans cette direction, ce stage propose d'implémenter un vérificateur de type pour la théorie des types dépendants. Ayant en vue sa vérification formelle, on s'attachera tout particulièrement à développer des abstractions logicielles proches des concepts mathématiques en jeu. L'objectif de ce stage sera, *a minima*, de définir les entiers naturels et de vérifier que "1 + 1 = 2" dans notre théorie des types. Ce résultat d'apparence anodine est en réalité l'un des résultats fondamentaux obtenu par Whitehead et Russell dans les *Principia Mathematica* en 1927, après 17 ans de travail.

Profil recherché : Le candidat devra être intéressé par les aspects théoriques et pratiques liés à la conception de langages de programmation. Une certaine aisance dans un langage fonctionnel (Haskell, OCaml, Scheme, ...) est attendue.

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