Raffinements Temporels pour la Programmation Fonctionnelle Réactive

Guilhem Jaber - LS2N - Université de Nantes
guilhem.jaber@univ-nantes.fr - http://guilhem.jaber.fr

Mots clés : Types Coinductif, Programmation Fonctionnelle, Logique Temporelle, Types Raffinés

1 Contexte

La programmation sur des valeurs infinies comme les streams est cruciale pour représenter les systèmes réactifs. Dans un tel cadre, les programmes ne terminent pas en général, mais calculent en temps finie une partie de leur sortie. Par exemple, si un programme doit produire une stream, il doit toujours être capable de produire le prochain élément de la stream en un temps fini. Cette propriété s'appelle la productivité.

La programmation fonctionnelle offre des abstractions de haut-niveau pour raisonner sur les types de données infinies, grâce aux définitions déclaratives et au raisonnement équationel, que l'on retrouve notamment en programmation fonctionnelle réactive. Dans ce cadre, les types de donnés formés de valeurs infinies sont représentés par des types coinductifs (définis par des plus grands points-fixes, ce sont les duaux au sens catégorique des types inductifs).

Dans un travail en cours avec Colin Riba [JaberRiba20], nous avons défini un système de types raffinés pour un langage manipulant de tels types coinductifs. Ce langage est basé sur les types récursifs gardés [Nakano00, CBBGB16], qui garantissent la productivité des programmes.

Les raffinements sont écrits dans une logique temporelle, le μ-calcul [Kozen83], disposant de définitions par plus petits et plus grands points fixes. On peut ainsi écrire des raffinements correspondant à des propriétés de sûreté, en définissant une modalité $\Box$, et de vivacité, via une modalité $\Diamond$. Nous avons construit un modèle catégorique de ce système en se basant sur le topos des arbres [BMSS12].

2 Sujet

L'objectif de ce stage sera d'explorer une des directions suivantes :

— Formaliser un plongement de ce système de types dans l'assistant de preuve Coq, et concevoir des tactiques permettant d'effectuer les preuves de raffinements temporels directement dans Coq. Cela nécessitera de formaliser les différentes briques du système (langage de programmation, logique temporelle, système de type) dans Coq. On cherchera également à trouver des règles de raisonnement de "haut-niveau" qui offrent le bon niveau d'abstraction pour raisonner sur les modalités $\Box$ et $\Diamond$, pouvant ainsi être implémenté sous forme de tactiques.

— Explorer les techniques issues du model-checking du μ-calcul pour raisonner sur l'automatisation de la vérification de types de notre système. On pourra se consacrer tout d'abord aux propriétés de sûreté.

3 Informations

Le stage se déroulera dans l'équipe Inria Gallinette du LS2N, à l'université de Nantes sur le campus de la Lombarderie. Colin Riba (ENS de Lyon, LIP) participera également à l'encadrement.

Références

[JaberRiba20] G. Jaber, C.Riba - Temporal Refinements for Guarded Recursive Types - `http://guilhem.jaber.fr/refitempo.pdf`

[Nakano00] H. Nakano - A Modality for Recursion - LICS'00

[CBBGB16] R. Clouston, A. Bizjak, H. Bugge Grathwohl, and L. Birkedal - The guarded lambda calculus : Programming and reasoning with guarded recursion for coinductive types - Logical Methods in Computer Science, 2016

[Kozen83] D. Kozen - Results on the propositional μ-calculus. Theoretical Computer Science 27, 3 (1983), 333 – 354.

[BMSS12] L. Birkedal, R. Møgelberg, J. Schwinghammer, and K. Støvring - First steps in synthetic guarded domain theory : step-indexing in the topos of trees - Logical Methods in Computer Science, 8(4), October 2012