Stage ou T.E.R. printemps/été 2019 :

Modélisation informatique des espèces combinatoires

Nicolas Magaud (magaud@unistra.fr)

La théorie des espèces combinatoires, proposée initialement par André Joyal [5], fournit un cadre homogène original pour décrire des structures discrètes (comme les graphes finis, les permutations, etc.) en utilisant des *fonctions génératrices*. L'objectif est de pouvoir décrire facilement des structures discrètes complexes par combinaisons et transformations de structures plus simples. Toutes ces opérations peuvent être décrites comme des opérations sur les fonctions génératrices.

Cette théorie permet un niveau d'abstraction élévé, les espèces étant une description purement algébrique, et donc indépendante des structures de données sous-jacentes. La théorie des catégories fournit un langage adapté à la manipulation des concepts de la théorie des espèces, sans qu'il ne soit nécessaire d'en maîtriser les fondements.

Brent A. Yorgey [6] propose un package écrit dans le langage fonctionnel Haskell permettant de modéliser des structures de données informatiques en utilisant la théorie des espèces. L'objectif de son travail était de faire connaître cette théorie dans la communauté de recherche sur la programmation fonctionnelle. L'idée de ce stage est de reprendre cette formalisation et d'étudier comment la transposer en Coq [1] et ainsi être en mesure de prouver formellement certaines des propriétés de ces opérations.

La formalisation en Coq pourra reprendre des éléments des travaux de John Dougherty [2] qui formalise les espèces combinatoires dans le cadre de la théorie des types homotopiques. On pourra également étudier comment réutiliser la bibliothèque sur la théorie des catégories en Coq (basée sur la théorie des types homotopiques) développée par Jason Gross [3] ainsi que celle d'Amokrane Saïbi [4]. Ces bibliothèques Coq pourraient servir de base pour décrire le langage de manipulation des espèces.

Références

[1] Yves Bertot and Pierre Castéran. *Interactive Theorem Proving and Program Development, Coq'Art : The Calculus of Inductive Constructions*. Springer, 2004.

[2] John Dougherty. Species in HoTT. Available from `https://github.com/jdoughertyii/hott-species`, 2015.

[3] Jason Gross, Adam Chlipala, and David I. Spivak. Experience implementing a performant category-theory library in coq. In Gerwin Klein and Ruben Gamboa, editors, *Interactive Theorem Proving - 5th International Conference, ITP 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings*, volume 8558 of *Lecture Notes in Computer Science*, pages 275–291. Springer, 2014.

[4] Gérard P. Huet and Amokrane Saïbi. Constructive category theory. In Gordon D. Plotkin, Colin Stirling, and Mads Tofte, editors, *Proof, Language, and Interaction, Essays in Honour of Robin Milner*, pages 239–276. The MIT Press, 2000.

[5] André Joyal. Une théorie combinatoire des séries formelles. *Advances in Mathematics*, 42(1) :1 – 82, 1981.

[6] Brent A. Yorgey. Species and functors and types, oh my ! In Jeremy Gibbons, editor, *Proceedings of the 3rd ACM SIGPLAN Symposium on Haskell, Haskell 2010, Baltimore, MD, USA, 30 September 2010*, pages 147–158. ACM, 2010.

1. Le système Coq [2] est un assistant de preuves dédié à la fois aux mathématiques et à l'informatique [1]. Il permet notamment de décrire formellement des théories mathématiques et de constuire des démonstrations de théorèmes s'appuyant sur ces théories. Il fonctionne de manière interactive. L'utilisateur construit *interactivement* ce qu'il croit être une preuve du théorème et le système vérifie *automatiquement* que la preuve construite démontre effectivement le thèorème considéré.