Mémoire proposal:
Semantics of Dependent Linear Type Systems

Advisors. Flavien Breuvart and Damiano Mazza

Keywords. functional programming, denotational semantics, dependents types, static analysis, linear logic, quantitative

Potential collaborations. Thomas Seiller

Prerequisite. A strong taste for denotational semantics is required. Additional notion of Linear Logic, advanced type systems, functional programming, category and/or static analysis are also welcomed.

Principle

The so called Dependent Linear type systems[1] are various type systems [5, 2, 6, 3] of similar structures which can be seen as specific but more expressive extensions of what we now call systems with graded comonads [4, 1][2] and can statically specify strong extensional properties at type level.

Your objective will be to dissect those different systems and extract similitudes and differences, with a particular attention given to the denotational semantics.

The ultimate objective is not required for the internship, but may be part of a PhD thesis (already funded, see below). We are aiming at a unification of those article toward two directions : a parameterised semantics that can be plugged on each of these systems and that contains both flavours of linear logic an dependent type semantics; and a parameterised type system in the style of [4, 1], but with the expressiveness of dependency.

References

[1] Aloïs Brunel, Marco Gaboardi, Damiano Mazza, and Steve Zdancewic. A core quantitative coeffect calculus. In Zhong Shao, editor, *23rd European Symposium on Programming, ESOP, Held as Part of ETAPS*, volume 8410 of *Lecture Notes in Computer Science*, pages 351–370. Springer, 2014.

[2] Antonio Bucciarelli and Thomas Ehrhard. On phase semantics and denotational semantics: the exponentials. *Ann. Pure Appl. Logic*, 109(3):205–241, 2001.

[3] Arthur Azevedo de Amorim, Marco Gaboardi, Justin Hsu, Shin-ya Katsumata, and Ikram Cherigui. A semantic account of metric preservation. In Giuseppe Castagna and Andrew D. Gordon, editors, *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages, POPL*, pages 545–556. ACM, 2017.

[4] Dan R. Ghica and Alex I. Smith. Geometry of synthesis III: resource management through type inference. In Thomas Ball and Mooly Sagiv, editors, *Proceedings of the 38th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL*, pages 345–356. ACM, 2011.

[5] Jean-Yves Girard, Andre Scedrov, and Philip J. Scott. Bounded linear logic: A modular approach to polynomial-time computability. *Theor. Comput. Sci.*, 97(1):1–66, April 1992.

[6] Ugo Dal Lago and Marco Gaboardi. Linear dependent types and relative completeness. volume 8, 2011.

[1] The name, from Ugo Dal Lago, may be misleading as the dependency is not the exact dependency of CoC's (i.e. Agda/Coq's) dependent types, but a weaker resource dependency.

[2] In those article they are called BLL-like systems and quantitative coeffect.

CoGITARe Project and fully funded PhD

This *mémoire* can be followed by a PhD fully funded by the ANR CoGITARe on the thematic.

Type systems are used to automatically check security properties of large programs. CoGITARe's goal is to extend typing methodology to a large panel of properties currently unreachable by state-of-the-art techniques, enabling in particular the analysis of quantitative properties of programs.

We will develop a way to keep track of the extensional information inside types in order to perform the whole static analysis at the level of types. For this purpose, we will combine two (re)emerging type systems, namely graded types and intersection type systems, with the well established techniques from the field of abstract interpretation such as widening.

Graded type systems formally embed a first order structure within types, while intersection types will help to circumvent the unconditional non-compositionality of fine grained resource analyses. This is how we plan to tackle the long running problem of applying abstract interpretation result in functional programming.