Implementation of hierarchies of algebraic structures in type theory
Cyril Cohen & Assia Mahboubi

= In short

Laboratory, institution: Inria, Université de Nantes (LS2N).

Location: LS2N, UFR Sciences et Techniques, 2, rue de la Houssinière, Nantes.

Host team: Équipe-projet Inria Gallinette.

Contact: Assia Mahboubi et Cyril Cohen: Mail.

Supervision: The internship will take place within the framework of the ERC FRESCO project, in the Inria Gallinette team. It will be co-supervised by Assia Mahboubi (in Nantes) and by Cyril Cohen (STAMP team, Inria Sophia Antipolis). A trip to visit Cyril Cohen during the course is envisaged. Travel and local expenses for this trip will be fully covered.

Keywords: Type theory, algebraic structures, hierarchies, logic programming, parametricity, Coq.

= Description

Context
Hierarchies of abstract mathematical structures today are the cornerstone of modern libraries of formalized mathematics [10, 2]. These structures provide interfaces for domains equipped with an algebraic structure and hierarchies describe the inheritance and sharing relations between these different abstractions. In type theory, it is possible to give an internal, first-class representation to these abstractions using telescopes (also called dependent tuples, or dependent records). Interactive provers based on type theory such as Coq [15] or Lean [4] implement powerful inference mechanisms to calculate inheritance links between instances of these structures [1, 9]. These mechanisms can be seen as variations on Haskell’s type classes, for a dependent type system, whose instance search is performed by executing a logical program.

As of today, the definition of these hierarchies in an interactive prover nevertheless remain a delicate task. The choice of the definition of the interfaces is of course crucial, but also the declaration and the parameter setting of meta-programs, which carry out the inference. Specific tools are actually required to diagnose [12, 6] errors.

The Hierarchy Builder tool [3] offers a simplified specific language for the definition of complex hierarchies from their elementary building blocks. Thanks to metadata provided by users in this language, it is possible to infer more information than that contained in the definition of structures alone, as a term of type theory. In particular, well-formedness verifications can be carried out a priori and not a posteriori, by diagnostic tools.
The use of this tool has in particular greatly simplified the description of the structure hierarchy underlying the Mathematical Components library, a state-of-the-art Coq library of formalized mathematics.  

Objectives

In this internship we propose to further extend this tool, in order to generate new information from user declarations. In particular, we propose to study the generation of the relevant category associated with a structure, and to automatically set up the associated morphism hierarchy. In practice, such a generation would indeed make it possible to increase the range of automation tools that can be derived generically from a structure definition. For example, one could declare or generate a left adjoint to the forgetting functor: in the case of a structure without axioms, one would thus obtain the syntax tree corresponding to the signature of this structure, the fact that it is an instance of the given structure, and the morphisms performing the addition.

The implementation of such a generation will rely on parametricity techniques, and in particular parametricity translation in the computation of inductive constructions. A simple example of this approach allows for example to generate principles of induction relevant to inductive types.

This internship includes a theoretical part, so as to understand and implement parametricity translation techniques into type theory. Then we will consider different classes of problems, of increasing difficulty. The internship also includes an implementation side, in the Coq proof assistant. The implementation will use Coq and Elpi, a logic (meta)programming language that allows to generate Coq code. The relative part of these two components can be adapted according to the taste and profile of the trainee.

= Bibliography


