Internship proposal L3 ENS – summer 2011

Topic Design of a proof-producing decision procedure for combined equivalence relations

Supervisor Pierre Corbineau (Maître de Conférences – Université Joseph Fourier)

Place & Duration VERIMAG (DCS team), Grenoble, 3 months

Contact `Pierre.Corbineau@imag.fr`

Scientific context : the need for setoids Equality reasoning is a very common paradigm for proofs both in the field of automated theorem proving and when using interactive proof assistants such as Coq[1]. The Coq proof assistant comes with a natural notion of equality which encompasses the notion of *computation* within the language (typed $\lambda-$calculus) that is used to express logical statements. However, it is often difficult to find a canonical representation for complex objects while maintaining simple implementations of functions and proofs of their properties. This can be illustrated by two examples :

- When adding polynomials represented as lists of integer coefficients, one has to remove leading zeros from the result.
- When adding or multiplying rationals (as pairs of integers), one has to reduce fractions.

There are cases where canonicalisation isn't even possible, especially in the case of functions. When reasoning over functions, the Coq equality captures equality of the *programming code* ($\lambda-$term) of the function rather than *pointwise equality* (for all inputs). However, most properties about functions are *extensional* (i.e. they are in fact properties of the images of the function).

A common approach to circumvent the problem is to work in a *setoid* : rather than using canonical representations, one can actually replace equality with an ad-hoc (partial) equivalence relation. The main consequence of this change is that replacement of equivalent objects can only occur in a suitable context — i.e under proper *relation morphisms* — in order to preserve logical equivalence.

Proposed Work In practice reasoning *modulo* the equivalence relation is often mixed with standard equality reasoning. The Coq proof assistant is equiped with proper procedures for replacing equivalent objects under morphisms and with an efficient decision procedure for quantifier-free equality reasoning [2] based on the *Congruence-closure algorithm*[3]. However, it lacks a procedure which would combine those two aspects and provide a generic equivalence reasoning procedure. We believe such a goal can be reached by extending the *union-find* data structure used for the Congruence-closure algorithm into a *hierarchical union-find* data structure.

Here are the tasks proposed for the internship :

- Study and specify the proposed algorithm, especially, identify sources of incompleteness and of incorrect or non-terminating proof generation.
- Implement and test a standalone prototype in Objective Caml.
- If there is time, some options may be studied :
  - Coq proof script generation
  - Pattern-matching modulo relations to handle quantified hypotheses
  - Extension to pre-orders and monotonic morphisms

Requirements The main goal of the internship is to study automated reasoning rather than Coq itself, so no prior knowledge of Coq is required and what is needed can be learned on the fly. However, we expect the candidate to be familiar with basic first-order logic and a little typed $\lambda-$calculus and to be fluent in Objective Caml.

[1] The Coq Development Team. The Coq Proof Assistant Reference Manual – Version 8.3. `http://coq.inria.fr`, 2010.

[2] P. Corbineau. Deciding equality in the constructor theory. In *TYPES 2006 : Types for Proofs and Programs*, volume 4502 of *Lecture Notes in Computer Science*, pages 78–92. Springer-Verlag, 2007.

[3] Downey, Peter J. and Sethi, Ravi and Tarjan, Robert Endre. Variations on the Common Subexpression Problem. In *Journal of the ACM*, volume 27 of issue 4, October 1980.