Design and Evaluation of Strategies for Automated Proofs using Reasoning Modulo Equivalence.

Supervised by Pierre Corbineau, Lionel Rieg, Karine Altisen
`Pierre.Corbineau@univ-grenoble-alpes.fr`
`Lionel.Rieg@univ-grenoble-alpes.fr`
`Karine.Altisen@univ-grenoble-alpes.fr`

Research Group Formal Proofs

Research Lab Verimag Lab
`http://www-verimag.imag.fr`

Lab Director David Monniaux
`david.monniaux@univ-grenoble-alpes.fr`

Research Institution Université Grenoble Alpes, Grenoble, France

Keywords Logic & Verification, Automated reasoning, Proof assistants

Scientific Context Automated reasoning is a widely-used technique for system and program verification, where problems are translated into logical formulas. These formulas can be checked by automated theorem provers. However, an automated prover may fail to answer a specific problem because of undecidability.

By contrast, interactive proof assistants allow the user to manually guide the proof process. This allows for more expressive proof techniques to be used at the cost of more user time and expertise. In order to reduce the burden on the expert user, it is desirable to provide suitable automation, especially for the seemingly trivial parts of interactive proofs.

Congruence closure[3] and equivalence graphs (E-graphs) are common tools used in Satisfiability Modulo Theory (SMT) automated provers. They have also been successfully applied to small-scale automation in the Coq interactive proof assistant [1]: indeed the `congruence` tactic is commonly used by Coq users to deal with proof problems involving equality and difference[2].

The final goal of the proposed research is to provide a decision procedure for the Coq proof assistant [1] that would extend equality-based reasoning to heterogeneous problems where equalities are expressed using multiple equivalence relations.

Scientific Problem 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 (mostly typed λ-calculus). When reasoning over functions, the Coq equality captures equality of the *programming code* (as a λ-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* (that is, a set equipped with an equivalence relation): instead of using Coq equality, one can define an *ad hoc* user-defined equality. In our current research prototype, we have designed aan extended E-graph data structure capable of handling multiple equivalence and partial equivalence relations and we have implemented the corresponding (heterogeneous) congruence-closure algorithm.

In order to give our solver additional deductive power, an E-matching algorithm allows to find instances of patterns within the E-graph, and to perform actions such as adding new terms and equivalences to the E-graph. In particular, this allows to perform deductions and simplifications of propositions modulo logical equivalence. Preliminary tests look promising.

The next step is to evaluate different strategies for expanding the E-graph during the proof-search process. In order to evaluate such strategies, we plan to instrument Coq in order to massively test our proof-search procedure on a properly identified proof corpus.

(see next page)

Proposed Work In this internship, we propose to work on the following tasks :

- Study & implement the addition of user-specified deduction rules to the existing procedure.
- Conduct tests and identify a suitable proof corpus where the procedure can be tested.
- Implement a repeatable testing infrastructure.
- Study aspects of proof-search strategies such as:
  - Criteria for instantiating quantified equivalence lemmas
  - Priorities and bounding strategies for the search space
  - Structural rules for reasoning with logical propositions
  - Structural rules for dealing with tuples / constructors

Required Skills The main goal of the internship is to study automated reasoning rather than Coq itself, so no prior knowledge of Coq is required. We expect the candidate to be familiar with basic first-order logic. Typed $\lambda-$calculus is a plus. Development skills in Objective Caml are mandatory.

Bibliography

[1] The Coq Development Team. The Coq Proof Assistant Reference Manual. `http://coq.inria.fr`.

[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.

Possible extension into a PhD thesis.