

Research in the context of WG 2.11

Christine Paulin-Mohring
LogiCal Project
Universit Paris Sud, Orsay

April 15, 2004

1 Introduction

The LogiCal project at Universit Paris Sud, INRIA Futurs and cole Polytechnique aims at developing proof environments for the generation of certified programs.

It has been developing the Coq proof assistant. Based on a powerful higher-order logic, this environment is suitable for abstract development of mathematical theories (for instance real numbers). Unlike other proof assistants (HOL, PVS), Coq is based on an intuitionistic type theory where there is a strong relationship between proofs and programs. Coq provides automatic translation from a constructive proof of an informative statement (such as a proof of existence or of a disjunction) into a functional program. This *extraction* mechanism performs an elimination of part of the proofs not required for computation as well as program transformation in order to improve the quality of generated programs which should be typable, readable and efficient.

2 Current and future research

A main achievement during the last few years has been the integration of modules into the proof language of Coq. The modules system which has been integrated at the proof level is similar to the CAML modules system. The extraction can consequently transform proof modules and functors into their functional counter-part. This has been successfully applied to the verification of CAML libraries for balanced trees. The fact that extracted modules can be associated to interfaces is important in order to build systems combining extracted modules and others developed in a traditional way and possibly involving non functional constructions.

The code generated from proofs contains extra arguments and inefficiency that need further analysis.

It is sometimes difficult to develop a proof that will give a good program via extraction. An alternative is to take the point of view of programming and use a rich type system that expresses the specification of the program. The environment will help developing the specification and write programs which are correct by construction. Part of the verification of programs could be done automatically, via the type system or automatic theorem proving, and part of it should be justified interactively by the user. To build on top of Coq a specific environment for programming with dependent types is part of our long term goals. This environment could integrate specific tactics for the development of programs implementing development techniques such as refinement or templates.

Currently, we are developing specialised environments for the specification and verification of programs written in Java or C. An originality of the approach is to generate from the source code a functional representation (which is certified modulo the validity of generated proof obligations). It should be possible to use this generated code for testing or simulation of programs.