A Declarative Proof Language for the Coq Proof Assistant

What is Coq?
Coq is an interactive proof assistant developed by the LogiCal team at the French INRIA institute. It allows to formally define mathematical objects and helps the user prove properties of those objects. Coq is based on a variant of Type Theory called Calculus of Inductive Constructions. The architecture of the Coq proof assistant ensures the correctness of the proofs written by the user. Coq is mostly used for two kinds of applications:
- the certification of computer systems and software (like textbook proofs)
- the formalisation of mathematics:
  - Four color theorem, C-CoRN...

Basic commands

A proof language consists of commands that modify the proof state, which keeps track of what needs proving. A procedural style language emphasises the use of specific methods to proceed, while a declarative style language emphasises the target proof state.

On the one hand, the declarative style is more readable because the script explicitly contains the successive proof states, more accessible because it relies on a few basic operations, more natural because commands use English words and standard logical formulae, and much maintainable because modifications affect the behaviour of the script only locally. The declarative style on the other hand, is much more verbose than the procedural style and relies heavily on automation to verify proof steps.

Previous uses of the declarative style include the Mizar Proof assistant and the ISAR language for the Isabelle proof assistant.

Procedural vs. Declarative style

A declarative, or top-down, style is more readable, easier to maintain, and much better suited to express the logical steps of a proof. The formalism of mathematics is completely different from the one used in the informal, or proof script.

Formalising Mathematics

The Foundations Team in Nijmegen uses Coq to develop a library of formalised mathematics known as C-CoRN. It contains constructive mathematics. The process of formalising mathematics is quite cumbersome since textbook mathematics are not written in a formal language. The difficulties arise from the formal system (Type Theory), but also to a much greater extend from the proof language.

Those difficulties are the impossibility to read the proof scripts without running them in Coq, the steep learning curve to for new proof developers, the exotic look of the proof script and the trouble to keep older proofs running for newer Coq versions.

To solve these problems, our approach consists in providing a new proof language for the Coq proof assistant, which needs to be:
- readable (by casual mathematicians)
- accessible (easy to learn)
- natural (like textbook proofs)
- maintainable (resistant to software updates)

A theorem from the book

Theorem 1 (Int. Math. Olympiads 1972, B2)
Let \( f \) and \( g \) be real-valued functions defined on the real line such that for all \( x \) and \( y \), \( f(x+y) + f(x+y) = 2f(x)g(y) \). If \( f \) is not identicallly zero and \( |f(x)| \leq 1 \) for all \( x \), prove that \( |g(y)| \leq 1 \) for all \( y \).

Proof: Let \( k \) be the least upper bound for \( |f(x)| \). Suppose \( |g(y)| > 1 \). Take any \( x \) with \( |f(x)| > k \), then

\[
2k = |f(x+y) + f(x+y)| \geq 2|f(x)||g(y)| = 2k|g(y)|
\]

so \( |f(x)| < k/|g(y)| \). In other words, \( k/|g(y)| \) is an upper bound for \( |f(x)| \) which is less than \( k \). Contradiction. □

Links

- http://coq.inria.fr/
  The Coq proof assistant
- http://www.cn.ru.nl/~corbineau/mode.html
  The Declarative Proof Language for Coq.
- http://c-corn.cn.ru.nl/
  The C-CoRN library of formalised mathematics.
- http://paulillac.inria.fr/xleroy/compcert-backend/
  A certified compiler backend in Coq
- http://mizar.org/
  Mizar: the first declarative proof assistant
- http://isabelle.in.tum.de/ISAR/
  ISAR, a declarative language for the Isabelle proof assistant