A Declarative proof language for the Coq proof assistant.

Pierre CORBINEAU

Foundations group, ICIS
Radboud Universiteit Nijmegen
The Netherlands

TYPES meeting, Nottingham, 2006

Motivations

- Coq is a proof assistant with a powerful formalism.
- Its formalism is quite far from usual set theory.
- Its tactics language does not help. . .
- Solution : borrow ideas from existing declarative proof assistants (e.g. Mizar).

Previous work

- Mizar (A. Trybulec, 1973?)
- Isabelle : ISAR (M. Wenzel, 1999)
- Mizar mode for HOL Light (F. Wiedijk & D. Synek, 2001)
- MMode for Coq (M. Giero, 2003)

LCF-style vs. declarative proofs

- Tactics emphasize proof terms rather than intermediate logical statements.
- Imperative style proofs lack structure.
- Tactics favour backwards proofs.
- Automation does not help enough.

Declarative proofs make automation more tractable

- Automation is mostly use to close a subgoal
- Other uses are very limited
  mostly normalisation in equational theories
- Needs to be strengthened to do bigger steps

Declarative proofs make automation more tractable

- Automation is mostly use to close a subgoal
- Other uses are very limited
  mostly normalisation in equational theories
- Needs to be strengthened to do bigger steps

Instead :

- Make most steps terminal (heavy use of cuts)
- Specify the right hypotheses to use
- Give more intermediate steps

Design choices

A mathematical proof language on top of what exists:

- Keep the same CIC terms.
- Allow switching to/from both modes.
- Enforce strong structure.
- Keep instruction by instruction execution.
- Replace multiple goals by one goal with mutliple conclusions.

Basic structure

Theorem T:ϕ.
dem.
instructions

done.

Basic structure

Theorem T:ϕ.
dem.
instructions

escape.
tactics
done.

Basic structure

Theorem T:ϕ.
dem.
instructions
claim T_1:ψ.
instructions
done.
escape.
tactics
done.

Simple steps

- introduction steps:
  assume/let/given hyps.
- cut steps:
  have/then statement by justification.
  $(\sim = \mid = \sim)$ object by justification.
  justification := objects/tactic tactic
- elimination steps:
  consider hyps from object.
  per cases/induction (on object/of statement by justification)
  suppose [it is pattern and] hyps.
  end cases/induction.
- conclusion steps:
  thus/hence statement by justification.

A small example

```
Lemma double_div2: forall n, div2 (double n) = n.
dem.
  assume n:nat.
  per induction on n.
    suppose it is 0.
      thus (0=0).
    done.
    suppose it is (S m) and Hrec:thesis for m.
      have (div2 (double (S m))
            = div2 (S (S (double m)))).
           ~= (S (div2 (double m))).
      thus ~= (S m) by Hrec.
    done.
  end induction.
done.
Qed.
```

Further work and availability

- arbitrary relation composition
- improve default automation
- automated proof skeleton generation

☞ http://www.cs.ru.nl/~corbinea/mmode.html