Coq

LASER 2011 Summerschool
Elba Island, Italy

Christine Paulin-Mohring
http://www.lri.fr/~paulin/LASER

Université Paris Sud & INRIA Saclay - Île-de-France

September 2011

Lecture 5 : Proof automation

- Example : `board_tac.v`

Why/When should I use Coq ?

Coq is not:

- a direct tool to find bugs in C, Java, concurrent programs . . .

Is it really your job ?

- design methods, tools to help others write correct programs!

Coq is helpful for:

- developing complex mathematical proofs with high guaranty
  - theorems in your papers (semantics)
  - back-end for program verification
- develop/prove pure functional programs (algorithms, tools)

Why/When should I use Coq ?

Coq is not:

- a direct tool to find bugs in C, Java, concurrent programs . . .

Is it really your job ?

- design methods, tools to help others write correct programs!

Coq is helpful for:

- developing complex mathematical proofs with high guaranty
  - theorems in your papers (semantics)
  - back-end for program verification
- develop/prove pure functional programs (algorithms, tools)

Why/When should I use Coq ?

Coq is not:

- a direct tool to find bugs in C, Java, concurrent programs ...

Is it really your job ?

- design methods, tools to help others write correct programs!

Coq is helpful for:

- developing complex mathematical proofs with high guaranty
  - theorems in your papers (semantics)
  - back-end for program verification
- develop/prove pure functional programs (algorithms, tools)

Misunderstanding?

- Can you solve this equation?
- OK, lets learn how to program a solution !

- The entry cost is a lot higher
- The possibility are much larger
- It is hard (but you are good, and it is also fun!)
- If you are lucky, somebody has developed a similar application that you can reuse.

Misunderstanding?

- Can you solve this equation?
- OK, lets learn how to program a solution !

- The entry cost is a lot higher
- The possibility are much larger
- It is hard (but you are good, and it is also fun!)
- If you are lucky, somebody has developed a similar application that you can reuse.

Misunderstanding?

- Can you solve this equation?
- OK, lets learn how to program a solution !

- The entry cost is a lot higher
- The possibility are much larger
- It is hard (but you are good, and it is also fun!)
- If you are lucky, somebody has developed a similar application that you can reuse.

Using a theorem prover as a back-end

Program correctness has been reduced to a logical statement.
Is it true ?

- use a first-order theorem prover
  - automatic
  - it possibly fails (undecidable), what can you do ?
  - if it succeeds, can you trust it ?
    (bug, inconsistent or inadequate theories ?)

- use a proof assistant
  - interactive : requires expertise and time
  - not much help to prove or refute a statement, use your brain!
  - reasonably trustable

Using a theorem prover as a back-end

Program correctness has been reduced to a logical statement.
Is it true ?

- use a first-order theorem prover
  - automatic
  - it possibly fails (undecidable), what can you do ?
  - if it succeeds, can you trust it ?
    (bug, inconsistent or inadequate theories ?)

- use a proof assistant
  - interactive : requires expertise and time
  - not much help to prove or refute a statement, use your brain!
  - reasonably trustable

Where automated deduction meets proof assistants

- most first-order theorem provers produce traces
- most proof assistants provides automated strategies (internal or external)

Do we have to choose ?

Why3 platform

`http://why3.lri.fr`
(J-C. Filliâtre, F. Bobot, C. Marché, A. Paskevich and others)
see *Why3: Shepherd Your Herd of Provers (BOOGIE 2011)*

New version of the WHY platform (still under development)

- Description of theories (polymorphic multi-sorted logic)
  - functions, algebraic data-types, axioms, lemma
  - modules
- Translation to multiple provers
  - proof assistants,
  - SMT/TPTP solvers,
  - Specialized solver: Gappa
- A programming language with annotations

Examples from the distribution: `genealogy`, `vstte10_queens`

The Frama-C platform

Static analysis tools for C programs
`http://frama-c.com/`
(B. Monate, L. Correnson, CEA LIST)

- ACSL : specification language for C programs
- Jessie : interpretation of C programs in WHY
- Examples (from P. Cousot course)

Which proof assistant?

Practical reasons:

- What they use in my team/company
- I have an expert next door
- The one I learned at school
- The library I need exists in that proof assistant

Ideological reason

- Classical versus intuitionistic logic
- Trust base

Great achievements by great people in all proof assistants
Coq, HOL, PVS . . .
Biodiversity is healthy!

Which proof assistant?

Practical reasons:

- What they use in my team/company
- I have an expert next door
- The one I learned at school
- The library I need exists in that proof assistant

Ideological reason

- Classical versus intuitionistic logic
- Trust base

Great achievements by great people in all proof assistants
Coq, HOL, PVS ...
Biodiversity is healthy!

Basic automation in Coq

- Database search: `auto`, `trivial`
- Decision procedure: `tauto`, `firstorder`, `omega`
- Propositional simplification: `intuition`
- Algebraic manipulations: `ring`, `field`

Specialized automation

- `gappa, interval` (G. Melquiond) (see PVS example)
- see also manual :
  - Psatz
    (F. Besson and E. Makarov, arithmetics over ordered rings)
  - Nsatz (L. Pottier, equalities in integral domains)
- external tools
  - resolution (M. Bezem, D. Hendriks and H. de Nivelle)
  - rewriting : color (F. Blanqui), coccinelle (E. Contejean)
  - ongoing work on integrating SAT/SMT solvers :
    MiniSat, VeriT (B. GrÃľgoire, C. Keller and al.)
    alt-ergo (S. Lescuyer)

A language for writing tactics

Ltac designed by D. Delahaye

- Write complex tactics without writing ML code.
- Specific language
  - specific patterns for matching goals (non-linear)

```
match goal with
   id:?A /\ ?B |- ?A => destruct id; trivial
   | _ => idtac
end
match goal with |- context[?a+0] => rewrite ...
```

  - specific backtracking
    - patterns are tried until the right-hand side succeeds
  - specific constructions : **fresh** *name*, **type of** *term* . . .
  - Coq data structures and terms

Example on the board

Computation

- Computation is part of type-checking (verification of convertibility)

$$\frac{\Gamma \vdash U : s \qquad \Gamma \vdash t : T \qquad T \equiv U}{\Gamma \vdash t : U}$$

- Internal programming language (functional kernel of CAML)
- Write complex programs and use them in proofs
  - Four colors theorem
  - Reflexive tactics
- Efficient reduction technics:
  byte-code compiling, machine integers (B. Grégoire)

Reflexive tactics

$$\frac{\texttt{refl\_eq} : t = t \qquad t \equiv u}{\texttt{refl\_eq} : t = u}$$

- Principle
  - function $\texttt{r2b} : A \to \texttt{bool}$,
  - function $\texttt{r2P} : A \to \mathsf{Prop}$,
  - proof $\texttt{rcor} : \forall a : A, \texttt{r2b}\, a = \mathit{true} \to \texttt{r2P}\, a$

    $$\texttt{rcor}\, a\, (\texttt{eq\_refl}\, \texttt{true}) : \texttt{r2P}\, a$$

    is well-typed when $\texttt{r2b}\, a \equiv \texttt{true}$
- Problems
  - $a : A$ should be closed (reification), and preferably small.
  - $\texttt{r2b}$ should be proved and reduce $\texttt{efficiently}$
- Applications
  - Ring, (R)Omega, Setoid Rewrite...
  - Interfaces between Coq and other systems using traces.

Example of reflection

```
Coq? Inductive form : Set :=
Coq?     T | F | Var : nat -> form
Coq?   | Conj : form -> form -> form.

Coq? Fixpoint f2P e (f:form) {struct f} : Prop :=
Coq?  match f with
Coq?     T => True | F => False
Coq?   | Conj p q => f2P e p /\ f2P e q
Coq?   | Var n => e n
Coq? end.

Coq? Definition e n := match n with
Coq?      0 => (0=0) | 1 => (1=2) | _ => True end.
e is defined

Coq? Eval compute in
Coq?   (f2P e (Conj (Var 0) (Conj (Var 1) (Var 1)))).
     = 0 = 0 /\ 1 = 2 /\ 1 = 2
     : Prop
```

Example on the board

Small reflexion: deciding $\forall b : \texttt{board}, P(b)$

See `board_tac.v`

Want to learn more about Coq ?

- CEA-EDF-INRIA summer school on: Modelling and verifying algorithms in Coq: an introduction, 14-18 November 2011 - INRIA Paris, France.
  `http://moscova.inria.fr/~zappa/teaching/coq/ecole11/` (register before sept 15)
- International Spring School on FORMALIZATION OF MATHEMATICS
  March 12-16, 2012 - INRIA, Sophia Antipolis, France
  `http://www-sop.inria.fr/marelle/Map-Spring-School.html`