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)
Can you solve this equation?

OK, let's 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, let's 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 \textbf{solve} this equation?
- OK, let's learn how to \textbf{program} a solution!

- The entry cost is a lot \textbf{higher}
- The possibility are much \textbf{larger}
- It is hard (but you are \textbf{good}, and it is also \textbf{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 provide automated strategies (internal or external)

Do we have to choose?
Why3 platform

(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. Grâl'goire, 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)
    ```coq
cmatch goal with
        id:?A \ / \ ?B |- ?A => destruct id; trivial
        | _ => idtac
    end

cmatch 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)

\[
\frac{\Gamma \vdash U : s \quad \Gamma \vdash t : T \quad T \equiv U}{\Gamma \vdash t : U}
\]

- 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

\[
\text{refl}_\text{eq} : t = t \quad t \equiv u
\]

\[
\frac{}{\text{refl}_\text{eq} : t = u}
\]

- **Principle**
  - function \(r2b : A \rightarrow \text{bool},\)
  - function \(r2P : A \rightarrow \text{Prop},\)
  - proof \(\text{rcor} : \forall a : A, r2b a = \text{true} \rightarrow r2P a\)
    \[
    \text{rcor } a(\text{eq}_\text{refl } \text{true}) : r2P a
    \]
  - is well-typed when \(r2b a \equiv \text{true}\)

- **Problems**
  - \(a : A\) should be closed (reification), and preferably **small**.
  - \(r2b\) should be **proved** and reduce **efficiently**

- **Applications**
  - Ring, (R)Omega, Setoid Rewrite. . .
  - Interfaces between Coq and other systems using traces.
Example of reflection

Coq? \textbf{Inductive} form : Set :=
Coq? \hspace{1em} T | F | \text{Var} : \text{nat} \rightarrow form
Coq? \hspace{1em} | \text{Conj} : \text{form} \rightarrow \text{form} \rightarrow \text{form}.

Coq? \textbf{Fixpoint} \ f2P \ e \ (f:\text{form}) \ \{\text{struct} \ f\} : \text{Prop} :=
Coq? \hspace{1em} \text{match} \ f \ \text{with}
Coq? \hspace{2em} T \Rightarrow \text{True} \ | \ F \Rightarrow \text{False}
Coq? \hspace{2em} | \text{Conj} \ p \ q \Rightarrow \ f2P \ e \ p \ \text{/\} \ f2P \ e \ q
Coq? \hspace{2em} \text{Var} \ n \Rightarrow e \ n
Coq? \hspace{1em} \text{end}.

Coq? \textbf{Definition} e \ n := \text{match} \ n \ \text{with}
Coq? \hspace{2em} 0 \Rightarrow (0=0) \ | \ 1 \Rightarrow (1=2) \ | \ _ \Rightarrow \text{True} \ \text{end}.
e \text{is defined}

Coq? \textbf{Eval compute in}
Coq? \hspace{1em} (f2P \ e \ (\text{Conj} \ (\text{Var} \ 0) \ (\text{Conj} \ (\text{Var} \ 1) \ (\text{Var} \ 1))))).
\hspace{1em} = 0 = 0 \ \text{/\} \ 1 = 2 \ \text{/\} \ 1 = 2
\hspace{1em} : \text{Prop}
Small reflexion: deciding $\forall b : \text{board}, P(b)$

See board_tac.v
Want to learn more about Coq?

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