Types Summerschool
Coq-lab

J-C Filliâtre and C. Paulin-Mohring

Bertinoro-august 2007

How to start with Coq

The following document (15 pages) is an introduction to the main features of Coq with included exercises.

Beginners with Coq should start with it. Doing all the exercises will take more time than available for Coq-lab sessions: do not hesitate to skip some of them.

Paradox

This exercise show that the excluded middle implies proof-irrelevance. We assume as an axiom that $\forall A, A \vee \neg A$.

- Show $\forall A, \neg\neg A \to A$.
- Introduce `BOOL` of type Prop with two elements T and F.
- Assume $T \neq F$, build f : Prop $\to$ `BOOL` such that $\forall A, A \leftrightarrow f\ A = T$.
- Load the module `Hurkens` in the standard library, look at the type of theorem `paradox`. Deduce a proof of $T = F$.
- Prove the proof-irrelevance property: $\forall (A : \mathsf{Prop})(p\ q : A), p = q$.

Mini-compiler

The purpose of this exercise is to use Coq to verify the correctness of a mini-compiler. The source language is a single expression involving integer constants, variables and additions. The target language is a assembly-like language with a single accumulator and an infinite set of registers. A template file for this exercise is available at

compiler.v

where the occurrences of the comment (* TO BE COMPLETED *) must be replaced.

Description of the exercice

Inductive definitions

This exercice illustrates the use of inductive definitions by analysing a simple cryptographic protocol for paid-TV.

Template file crypto.v

Extraction

1. Define inductively a relation `Fib` which specifies the fibonnaci function ($fib\,0 = 0$ $fib\,1 = 1$ $fib\,(n+2) = fib\,n + fib\,(n+1)$) as a binary relation.
2. Prove the following principle by one simple induction: $\forall P : nat \to \mathsf{Type}, P\,0 \to P\,1 \to (\forall n, P\,n \to P\,(S\,n) \to P\,(S\,(S\,n))) \to \forall n, P\,n$
3. Use this principle in order to prove

$$\forall n, \{m | Fib\,n\,m\}$$

4. Look at the extracted program, what is its complexity ?
5. Propose another proof of the induction principle using the **Fixpoint** construction, compare the programs obtained.

Proof by reflection and tactic language

This exercise illustrates the construction of tactics by reflection and the use of the tactic language for building the reification function.

The file ltac_refl.v contains the example of tactic by reflection demonstrated in the course (simplification of conjunctive formula)

1. Implement the simplification function of your choice
2. Prove that it is correct
3. Implement your own tactic for doing the simplification
4. Improve the reification function in order to assign the same variable for two occurrences of the same formula.
   Hint: it is possible in Ltac to test that 2 terms are equal by the following construction

```
match t=u with ?x=?x ⇒ ... end
```