Proofs for freshmen with Coqweb

Jérémy Blanc

Laboratoire Jean-Alexandre Dieudonné, Université de Nice-Sophia Antipolis, CNRS

J.P. Giacometti

INRIA Sophia Antipolis

André Hirschowitz

Laboratoire Jean-Alexandre Dieudonné, Université de Nice-Sophia Antipolis, CNRS

Loïc Pottier

INRIA Sophia Antipolis

Abstract

Coqweb is a web interface for Coq, primarily designed for teaching. It allows teachers to propose statements in sufficiently familiar form. Students are invited to prove these statements essentially by clicking. Hints can be given in natural language, in which case the interface checks that the student somehow understands the hints. We present here the main features of Coqweb, and show through examples how it has been used since 2004.

Keywords: Proof Assistant, Interactive Theorem Prover, Education

1 Introduction

Proof assistants are now mature in some sense: they can be used to build formal proofs of highly respectable mathematical results (the fundamental theorem of algebra [FTA], or the four-color theorem [Gonthier Werner 04] for example). Unfortunately, only experts can use them, and learning to use a powerful assistant like Coq needs days of training. How could these proof assistants help teachers to teach what a proof is and how to search for a proof? The Coqweb environment [Coqweb] (a free software developed in ocaml and php by J.P. Giacometti, A. Hirschowitz and L. Pottier) is designed for teaching freshmen. Accordingly its main features are:

- The language of statements is fairly close to the standard mathematical language (thus full of ambiguities).

This paper is electronically published in
Electronic Notes in Theoretical Computer Science
URL: `www.elsevier.nl/locate/entcs`

- The language of proof (tactics) is also fairly close to the standard mathematical practice.
- Proofs are essentially performed by clicking.
- It can be coupled with Wims, so that teachers can include Coqweb exercises into their Wims class sheets.

During the past three years, the regular freshmen (in mathematics and computer sciences) at the University of Nice-Sophia-Antipolis have been proposed a small amount of proving activity with Coqweb, essentially through the WIMS server [Xiao 98]. After several experiments, Coqweb is now a wiki where one can develop mathematics as we do in a course or a book : informal text using Latex, formal definitions of mathematical objects in a language close to the usual mathematical language, statements and proofs of theorems, exercises. As its name suggests, Coqweb heavily relies on the Coq assistant proof, which is used to performing the proofs and verifying them, even if this is quite transparent for the student.

The originality of Coqweb could be that the major concern of its design is to be accepted by typical students and teachers. Accordingly, the traditional approach initiated by the CtCoq pioneering system [Bertot 99] is slightly revised. In Coqweb, we have two separate modes: in the teacher mode, the expert teacher declares definitions and other resources, while in the student mode, the student solves an exercise, or "reads" the proof of a theorem by following the hints of an informal proof. We have focussed our attention onto the student mode, since, just as in the Wims server, non-expert teachers may use resources developed by expert teachers, and do not need to use the teacher mode. On the other hand, the student mode relies on the teacher mode where are generated the statements to be proven as well as the resources to be invoked.

This paper is organized as follows. In section 2, we briefly relate and discuss our experience in teaching proofs with Coqweb. In sections 3 and 4, we describe the student mode and its tactics. In section 5, we describe the teacher mode. In section 6, we show a complete example of a mathematical development in Coqweb for the freshmen at the University of Nice-Sophia-Antipolis. Finally in section 7, we discuss related and future work. The description and the discussion of the implementation of Coqweb will be done elsewhere.

Acknowledgments

We thank Valérie Moreau-Villeger, Joachim Yaméogo and Xiao Gang who took care in various ways of the connection between Coqweb and Wims. We thank Marco Maggesi and Carlos Simpson for stimulating conversations. We thank the organizers of the second Wims conference [WIMS07] who invited us to present our views on proof-teaching with Coqweb. We thank the referees whose criticism led to significant improvements in our presentation. Finally, the third-named author wishes to thank his chairman Michel Merle, his dean Raymond Negrel and his colleague Paul Silici who made possible the experience reported here, at least until fall 06. For what happened then, see [BlogDuB].

2 Teaching proofs

2.1 An impossible task?

Proofs are not really taught at our universities. Teachers just show and explain real proofs and then invite students to understand and reproduce them. From this experience, students are expected to be able to produce new proofs without having ever been explained what a proof is, and while their basic logic needs serious consolidation. Some teachers are satisfied with the way they teach proofs, considering that students who work enough should understand (and the accompanying conclusion is too often that most students do not work enough. For more on this controversial line, see [BlogDuB]). But many others consider teaching proofs as an almost impossible task. Some researchers in Education confirm the latter position. We quote from [Epp 03] (p.8): "A likely consequence for mathematics instruction is that in order to learn a complex process such as proof and disproof, effective integration of new modes of thought with pre-existing contradictory modes is a major undertaking. It is not surprising that easy solutions have not yet been discovered".

2.2 Proof plans versus complete proofs

One of the reasons why proofs are not understood by students is that the traditional informal style for proofs gives just a plan from which the reader is supposed to reconstruct a complete picture. Think of a game of chess as reported in a newspaper: they just give the sequence of moves, in some minimal notation. Grandmasters will understand the game from this minimal information without using a chessboard. But the typical reader of the newspaper will need a chessboard in order to see the corresponding sequence of positions and understand what is going on, if not necessarily why. The sequence of positions enlightens the corresponding sequence of moves.

Just the same happens for informal proofs as well as for proof scripts: they consist basically in an enumeration of actions (often called tactics) developing the proof, and this enumeration will be enlightened if we simultaneously show the corresponding enumeration of states of the proof.

Four or five years ago, one of us tried to teach proofs by showing and maintaining the state of the proof on a blackboard. This ended with the conclusion that he was not able to do it efficiently, even for freshmen proofs. On the contrary, to show the complete proof (including actions and states) on the screen through the Coqweb interface is pretty satisfactory. Indeed, we did experiment with this at the University of Nice during the first course in calculus (fall 06). Of course for this to be possible, it was necessary that the pretty-print of statements proposed by Coqweb be sufficiently close to what the typical freshman could accept and understand. In particular Coqweb had to manage ambiguous formulas.

2.3 Procedural versus declarative proofs

One of the reasons why proofs are not taught is that most teachers have not been taught themselves what a proof is. Some of them have been given a course in logic, but the notion of proof given there has essentially nothing to do with a real proof. As a matter of fact, when you ask a logician, like J.Y. Girard for instance [Girard 07], what logic is doing in order to fill the gap between the current notion of logical proof and the current practice of mathematicians, he confirms that this task is taken into account mainly (only?) by the theorem provers community. Indeed each theorem prover, in particular our favorite one, Coq, has its operational notion of proof, which is sufficiently precise, at least for freshmen, and reflects a corresponding practice. We believe that a precise notion of proof is a crucial ingredient when trying to teach proofs for freshmen.

The Curry-Howard paradigm teaches us that a proof is a λ-term, but this point of view, while totally formalized and extremely appealing for researchers, is definitely too abstract for our students. Researchers of the theorem provers community have considered procedural (as in Coq [Coq 06]) as well as declarative (as in Mizar [MIZAR]) notions of proof scripts (see e. g. [Autexier Sacerdoti Coen 06,Corbineau,Sacerdoti 06]), and more recently have started mixing the two styles [Wiedijk 04]. Procedural as well as declarative scripts are again some form of incomplete proofs in the sense that either the successive states or the successive proof steps have to be inferred. The choice between the procedural and declarative styles is not too meaningful for Coqweb because students are not supposed to write proofs but only to generate them by clicking. Thus an informal notion of proof mentioning altogether states (lists of goals) and transitions (performed by tactics) is suitable. We have observed that such a notion of proof is well accepted by students.

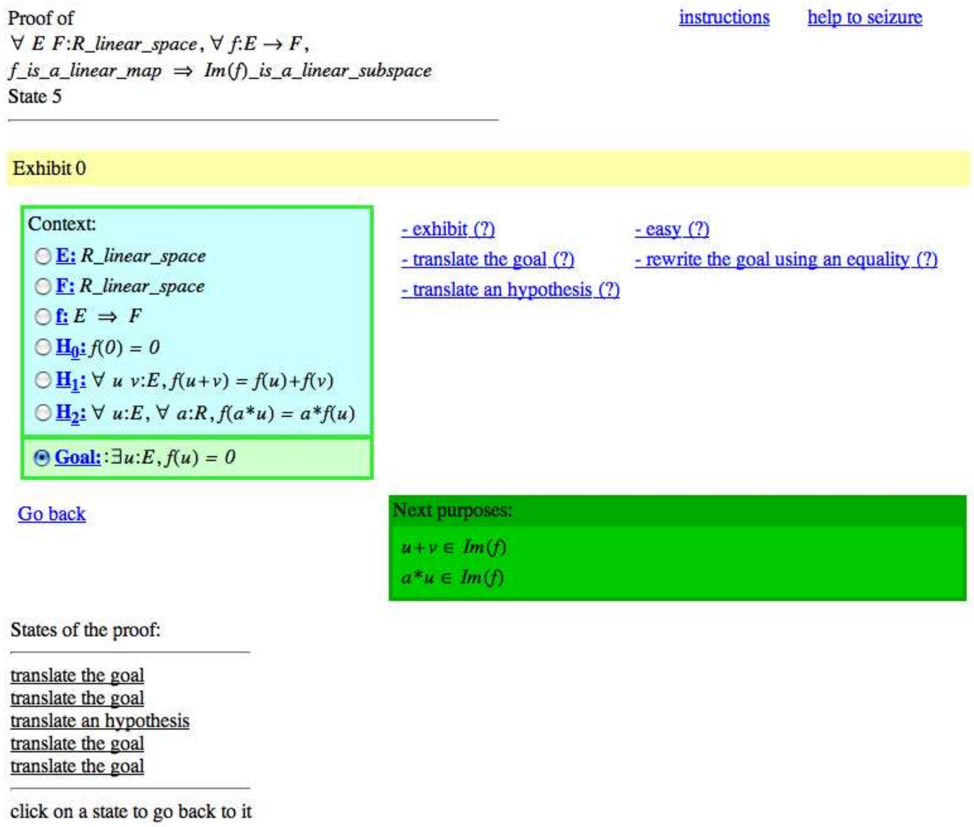
2.4 Formal versus informal proofs

Teachers not acquainted with proof assistants insist that teaching formal proofs is relevant only if it is a way to teach informal proofs. For the moment, the only echo in Coqweb to this legitimate requirement is the guided mode. In this mode, the student is offered a hint which suggests more or less explicitly which tactic should be performed. In particular, the hint may take the form of the relevant sentence in a corresponding informal proof. In this case, the student has to translate the informal proof into a formal one. Of course, it is possible to propose exercises where the student has to perform the translation in the other way, but we did not try this yet.

2.5 Concepts versus images

One of the reasons why the proving activity is assimilated by so few students is that is is perceived as a kind of artistic activity, where very few helpful concepts and images are proposed to the beginner. Our attempt proposes several key concepts which are sufficiently simple and concrete, and to which Coqweb associates persistent images. The first main attribute of a proof is the corresponding sequence of *states* and the image for a state is the corresponding full student window. A state is itself a sequence of *purposes*, and the current purpose has its reserved place on the student window. A purpose is a pair of a *context* and a *goal* and both have their specific place in the student window. Similarly *variables* and *hypotheses* are the constituents of the context. Finally *tactics* have their own place in the student window.

3 The student window



The student window is the window where the student is supposed to perform a proof. It can be activated either from the starting page of a Wims exercise (module coqweb_new), or through a button attached to an exercise in the teacher window. Note that students have access to the teacher window as everybody do.

The student window displays the original statement (overall goal), the current purpose, consisting of the current context and the current goal, the list of pending goals, the list of available tactics, and, optionally, the current hint. In order to perform a tactic, the student clicks on the corresponding button. In case the current tactic needs arguments, the student window opens a dialog area.

In the first version, Coqweb implemented "proof by pointing", as in CtCoq: to some extent, Coqweb was able to infer the tactic from the pointed area for instance in the goal; but we have discovered that it was too much an invitation to click without relating it to any concept.

3.1 Statements

Mathematical statements appear in the student window as goals or hypotheses. Our main concern has been that they appear in a form as close as possible to the one used by typical teachers. We give below a representative list of exercises which were solved by students during the past academic years, in the form where they appeared in the student window (up to word by word translation from French to English). The reader will observe that it contains "immediate" consequences of definitions, as well as main results of the course. Note also that the disjunction and existential quantifier allow to ask for justified questions and calculations.

(i) Prove: $[2;3] \subset [1;4]$.

(ii) Prove: $(2,7,6)\_is\_a\_linear\_combination\_of\_(1,2,0) and (0,1,2)$.

(iii) Prove: $\exists a : R, \exists b : R, (40,62) = a * (1,14) + b * (6,1)$.

(iv) Prove:$\forall f : R \to R, (\forall x : R, f(x) = x+1) \to f\_is\_injective$.

(v) Prove: $x : R \mapsto 2 * x + 3\_is\_injective$.

(vi) Prove: $\forall f\ g : R \to R, f\_is\_increasing\ and\ g\_is\_increasing$
$\to f \circ g\_is\_increasing$.

(vii) Prove: $\forall f\ g : R \to R, f \circ g\_is\_injective \to f\_is\_injective\ or\ g\_is\_injective$.

(viii) Prove: $\forall f : R \to R, \forall u : sequence(R), (\forall n : N, u_{n+1} = f(u_n))\ and$
$(u_0 < u_1 and\ f\_is\_increasing) \to u\_is\_increasing$.

(ix) Prove: $\forall a\ b : R, \forall f\ g : [a,b] \to R, f\_is\_increasing\ and\ g\_is\_increasing \to f + g\_is\_bounded\_above\_by\_f(b) + g(a)$.

(x) Prove: $\forall u\ v\ w : sequence(R), \forall a : R, limit(u) = a\ and\ (limit(v) = a$
$and\ (u \leq w\ and\ w \leq v)) \to limit(w) = a$.

(xi) Prove: $\forall EF : R\_vector\_space, \forall f : E \to F, f\_is\_linear$
$\to Im(f)\_is\_a\_linear\_subspace$.

3.2 Guided mode

The student window has a guided mode, where the next tactic is suggested by an informal hint. This hint is entered by the teacher directly through the student window (accessed with teacher's rights). In the guided mode, the student has to follow the suggested proof, otherwise an error message is emitted. One of the advantages of this mode is that only the necessary tactics are listed, and often they are very few, typically three or four.

3.3 Pending goals

As in other theorem provers, some tactics generate several goals. This is for instance the case of "apply" when the selected resource has several hypotheses. In this case, the pending goals are simply listed in a specific area on the student window, on the lifo mode. The corresponding contexts are not mentioned.

When using a proof assistant in order to prove a "serious" statement, the number of pending goals may easily grow up to ten or so. If in the same time these goals turn out te be difficult to read, when meeting a goal generated long ago, the user may wish to know "where this damned goal comes from". This problem is note handled at all in Coqweb. Indeed, for the statements proposed to students, pending goals are always simple and their origin is always clear.

3.4 Dialogs

Some tactics need arguments. This is the case for instance of the tactic "exists" which we call "exhibit". When this tactic is invoked, Coqweb opens a (one-line) area where the student is expected to enter the witness. The more important case is the case of the tactic "apply a resource" where Coqweb lists all relevant resources. The order in which they appear is programmed in such a way that, for our range of goals, the desired resource appears almost always among the very first ones. There is no alternative mechanism for browsing resources.

4 Tactics

In Coq as in many other theorem provers, the *tactics*, are the actions/transitions/moves which are available for jumping from one state of a proof to a new one, presumably easier. The list of Coq tactics contains around a hundred entries (see [Coq 06]). We cannot teach a hundred tactics at once, so we have to identify basic ones, and present them in such a way that the typical freshman will recognize an activity of proof which he already met earlier. We review below some of the most useful tactics and their relationship with freshmen.

4.1 The introduction tactic

The first Coq tactic, in the sense that almost every proof starts with it, is introduction. This is bad luck because, while introducing is certainly the right word (you definitely *introduce* a variable by telling its name), this cannot be understood by the typical freshman as a serious mathematical activity. If you try to teach introduction, students consider you just as mathematicians consider logicians: people concerned with meaningless mathematics. Thus introduction cannot be one of the basic freshman tactics. We will merge it with the "unfold" tactic, see below.

4.2 The apply tactics

While the typical freshman is not interested in introducing, he is perfectly interested in *applying*. Even too much. Indeed, teachers do not distinguish between three kinds of application.

The first one is what we call "apply a resource" or "apply a hypothesis". It is the major tool for a backward proof. The resource is a statement and the current goal should be the conclusion of an instance of this statement. Then the current goal is replaced by as many new goals as the instance has hypotheses. Of course, the resource should be available in the library.

The second one is what we call "deduce": again we apply a resource or a hypothesis, this time in order to prove not the current goal, but some other statement which we want to add to the current context. This tactic is what allows to proceed *forwards* (from the assumptions to the goal) while in a theorem prover like Coq, the default direction is *backwards* (from the goal to the assumptions). Note that teachers often prefer to think forwards and may reject the whole approach described here as "too backwards". While if you observe proofs from this point of view, you will remark that the forward approach opens new room for the opacity of proofs since you may gather intermediate statements without saying why. Of course something similar happens in the backward approach. The difference is that in general the assumptions of a resource are considered easier to prove than the conclusion.

The third one is of a very different nature: we apply a resource which claims an equality in order to rewrite something in the current goal or in the current context. This is the "rewrite" tactic, which the typical freshman does not accept so easily, because he is so much used to apply and not at all to rewrite. More generally, the distinction between these three "apply" tactics is difficult to teach. At least we can stress their differences.

4.3 The unfold and simplify tactics

Just as "introduce", "unfold" is a word that mathematicians do not use. Indeed, since unfolding a definition is totally transparent for "Grandmasters", it is also completely neglected by teachers. Nevertheless, what emerges from our experience is that "unfolding" is a crucial proof activity for freshmen. This tactic shares with "introduction" the property that it has stricty no mathematical content. Thus we have merged these two tactics in a single one. For its name we have chosen the French verb "expliciter" (in the present paper, we will use "translate") which we can indeed use in the right places in our informal proofs. Also we have extended this tactic as much as possible in order to include all kinds of similar replacements. For instance, when applied to the goal $x \in A \cap B$, this tactic generates the two goals $x \in A$ and $x \in B$, although this is certainly not obtained by unfolding the definition neither of $\in$ nor of $\cap$. Similarly, from the goal $AandB$, it generates the two goals A and B.

In the same vein, we have a "simplify" tactic which we extend as far as possible (see 6.2 for an explicit use of this tactic).

4.4 Other tactics

First of all, we certainly need a tactic "easy", which handles "too" easy goals. Indeed, the typical freshman hates being asked to prove something too easy. But what looks easy for a freshman, is not necessarily so easy to prove in Coq, so that the "easy" tactic is not so easy to implement.

Next, the freshman uses the "exists" tactic, which we call "exhibit" since we consider as mandatory to find for each tactic a name which is a verb and reflects the activity. Of course this tactic is very important for understanding existence, which is a very delicate notion (not only) for freshmen. See 6.6 for an explicit use of this tactic.

Finally we have a tactic for contraposition and a tactic for induction.

5 The teacher mode

The teacher mode allows the expert teacher to declare definitions, axioms and exercises. The exercises may be activated, yielding a student window, from where resources (definitions and axioms) will be invoked. The teacher mode runs on Wiki-Coqweb [WCW], a wiki based on Spikini, which is a variant of Wikini for Spip. For the moment, it is designed for expert teachers, more precisely for highly motivated teachers. Let us say briefly that the main task of Coqweb is to convert adequately the teacher's input into the corresponding Coq declarations, and to acknowledge it on a course page available on the wiki. More precisely, this task splits in two steps. First Coqweb checks syntactically the input before eventually writing a course page. While when a button is activated on a course page, Coqweb performs a semantic check before definitely acknowledging the definition, or before opening a student window. . We briefly describe the contents of course pages and the way they are edited.

5.1 Course pages

A course page is a page where the teacher can write a text including latex formulas, and insert Coqweb definitions and exercises. Here is for instance how appears the definition for vector space

```
Definition  cons_ev : qs E :Ens, (E->E->E) -> E->( R->E-> E)-> PREV
         (en coq: cons_ev)
Properties:

    * qs E :Ens, qs a : E->E->E, qs z :E, qs m : R->E ->E,
      E=( cons_ev a z m)
    * qs E :Ens, qs a : E->E->E, qs z :E, qs m : R->E ->E,
     ( (plus_ev (E := ( cons_ev a z m) ) ) = a )
    * qs E :Ens, qs a : E->E->E, qs z :E, qs m : R->E ->E,
     ( (zero_ev ( cons_ev
a z m) ) = z )
    * qs E :Ens, qs a : E->E->E, qs z :E, qs m : R->E ->E,
      ( (mult_ev (E := ( cons_ev a z m) ) ) = m )
```

Note that this is for the teacher, not for the student, even if it is freely available.

For each definition, there is a button for converting altogether the definition into a Coq declaration and the properties into axioms. For this conversion to work, the relevant other pages should have been loaded, and the previous definitions in the current page should have been already converted. If Coqweb does not succeed in this conversion, a (rough) error message appears. For each exercise, there is a button which is supposed (if the formulation is correct with respect to previously loaded material) to open a student window.If Coqweb does not succeed in converting the statement into a Coq statement, a (rough) error message appears.

Course pages are linked to one another in the standard hypertext way.

5.2 Editing course pages

Course pages may be edited within the wiki. The conception and edition of course pages is the main teacher's task, along with the optional task of writing hints. When trying to validate a definition together with its accompanying properties, the teacher often requests the disambiguation ability of Coqweb. However we do not expect Coqweb to choose among various possible interpretations according to the context. Our idea is rather that teachers (and students) should write statements in a way offering only one reasonable interpretation. As an illustration of this position, our quantifiers are always explicitly typed.

Coqweb offers the possibility to declare two names for each definition: one name is for Coq and supports no overloading, while the second one is for Coqweb only and supports overloading. Of course the two names may coincide, which is the case in the above exemple..

6 Algebraic numbers in Coqweb

We present here some pages of Coqweb which have been used at the University of Nice-Sophia-Antipolis in 2007 along within a mini-course for motivated students on polynomial equations and their solutions. to present Coqweb to teachers.

6.1 Fields

Firstly, we will work with fields. It is convenient to code this in one page, called "fields", and then to specialize to some special fields like $\mathbb{Q}$, $\mathbb{R}$, or $\mathbb{C}$. Here is what the page on the web looks like (we show only the compiled page, instead of the code page, as it is more readable, but the syntax is quite the same):

Definition *field: Set*
Definition *coercion:field→Set*
Definition *+: ∀K: field, K→K→K*
Properties:
- *∀K:field, ∀x y z:K, (x+y)+z=x+(y+z)*
- *∀K:field, ∀a b:K, a+b=b+a*

Definition **: ∀K: field, K→K→K*
Properties:
- *∀K:field, ∀x y z:K, (x*y)*z=x*(y*z)*
- *∀K:field, ∀a b:K, a*b=b*a*
- *∀K:field, ∀x y z t:K, (x+y)*(z+t)=x*z+x*t+y*z+y*t*

Definition *1: ∀K: field, K*
Properties:
- *∀K:field, ∀x:K, x*1=x*

Definition *0: ∀K: field, K*
Properties:
- *∀K:field, ∀x:K, x*0=0*
- *∀K:field, ∀x:K, x+0=x*
- *not(0=1)*

Definition */: ∀K: field, K→K→K*
Properties:
- *∀K:field, ∀x y z:K, (not (y=0)→(x/y)*y=x*

Definition *-: ∀K: field, K→K*
Simplifications:
- *∀K:field, ∀x:K, x+(-x)=0*
- *∀K:field, ∀x:K, (-x)+(x)=0*

Exercise: ∀K: field, ∀x y: K, x+y=0→y=-x

Definition *-: ∀K: field, K→K→K*
Properties:
- *∀K:field, ∀x y:K, (x-y)=x+(-y)*

Exercise: ∀K: field, ∀x: K, (-(-x)=x

Exercise: ∀K: field, ∀x y: K, -(x+y)=(-x)+(-y)

Definition *power: ∀K: field, K→N→K*
Properties:
- *∀K:field, ∀x:K, x^0=1*
- *∀K:field, ∀x:K, ∀a:N, x^(a+1)=x^a*x*

Each element of the set *field* corresponds to a field, and may be also viewed – using the coercion above – as a set. The symbols 0 and 1 represent the traditional constants. Observe the standard overloading for "-" .

Note that the function *power* may be used writing the symbol ^, and that the set $N = \mathbb{N}$ was introduced earlier, in a similar way.

The three exercises above are simple examples suited for the student who begins to use Coqweb. The student will see that although the assertions seem very simple, they are not so trivial to prove. He will need less than ten tactics in general , and will see the so many steps that we usually skip in a classical proof.

6.2 A first proof in detail

We illustrate now in detail a proof of the first exercise of Section 6.1, describing what the student sees after each of his uses of tactics:

Goal: *∀K: field, ∀x y: K, x+y=0→y=-x*

The student clicks on the tactic *"Translate the goal"*. This corresponds to the usual mathematical sentence "Let K be some field, and let x, y be two elements in K".

Context: **K:***field*, **x:***K*, **y:***K*, **H:***x+y=0*, **Goal:***y=-x*

The student clicks on the tactic *"Prove an intermediate result"*. The computer asks him what the result is, and he chooses "`-x+(x+y)=-x`".

Context: **K:***field*, **x:***K*, **y:***K*, **H:***x+y=0*, **Goal:***-x+(x+y)=-x*, **Next purpose:***y=-x*

The student clicks on the tactic *"Rewrite the goal using an equality"*; the computer offers a list of available equalities, among which the student selects the hypothesis **H**. The computer replaces the expression $x+y$ by 0 in the goal.

Context: **K:***field*, **x:***K*, **y:***K*, **H:***x+y=0*, **Goal:***-x+0=-x*, **Next purpose:***y=-x*

The student chooses the tactic *"Simplify the goal"*. (Recall that the property *"∀K:field, ∀x:K, x+0=x"* was declared as a simplification).

Context: **K:***field*, **x:***K*, **y:***K*, **H:***x+y=0*, **Goal:***-x=-x*, **Next purpose:***y=-x*

The student may now click on the tactic *"easy"* to achieve the proof of the intermediate goal, which becomes an additional assumption in the next context.

Context: **K:***field*, **x:***K*, **y:***K*, **H:***x+y=0*, **H_0:***-x+(x+y)=-x*, **Goal:***y=-x*

The student clicks on *"Rewrite the goal using an equality"* and selects the hypothesis **H_0**. The computer replaces the expression $-x$ by $-x+(x+y)$ in the goal.

Context: **K:***field*, **x:***K*, **y:***K*, **H:***x+y=0*, **H_0:***-x+(x+y)=-x*, **Goal:***y=-x+(x+y)*

Three tactics are then needed to conclude. The first one rewrites $-x+(x+y)$ into $-x+x+y$, using the commutativity defined as a property of the addition. The second one simplifies the goal (the computer will replace $-x+x$ by 0 and then $0+y$ by y), to obtain $y=y$ instead of $y=-x+x+y$. The last one is *"easy"*.

6.3 The field $\mathbb{Q}$, $\mathbb{R}$ and $\mathbb{C}$

We define some special elements of the set *field*, which correspond to the standard fields $\mathbb{Q}$, $\mathbb{R}$ and $\mathbb{C}$, and add some specific properties to these. For $\mathbb{R}$ we add some order, and the n-th roots:

Definition *nroot: R→N→R (in coq: nroot)*
Properties:
- *∀x:R, ∀n:N, x>=0→n>0→(nroot x n)^n=x*

In a Coqweb's proof, the expression *(nroot x n)* is displayed as $\sqrt[n]{x}$.

For $\mathbb{C}$, we add the functions *Im* and *Re* which give the real and imaginary parts. We do not present these explicitly here.

6.4 Operations on functions over a field

Since our polynomials will be coded as functions, we need some operations on functions over a field. We define the addition, multiplication and subtraction of two functions and the opposite function of a function. The addition is for example coded like this:

Definition *+: ∀E:Set,∀K: field, (E→K)→(E→K)→E→K*
Properties:
• *∀E:Set, ∀K:field, ∀f g:E→K, ((f+g) x)=(f x)+(g x)*

The other operations are similarly defined.

6.5 Polynomials

In our implementation, a polynomial is a special function. We define the standard polynomial X (which is in fact the identity map), the constant polynomials, and continue to define other polynomials inductively. We work over the complex field $\mathbb{C}$.

Definition *X: C→C*
Properties:
• *∀a: C, (X a)=a*
Definition *coercion: C→(C→C) (coq name: cfcst)*
Properties:
• *∀a x: C, (cfcst a x)=a*
Definition *is_a_polynomial: (C→C)→Prop*
Properties:
• *∀a: C, (is_a_polynomial (a)*
• *is_a_polynomial (X)*
• *∀P Q :C→C, (is_a_polynomial P)→(is_a_polynomial Q)→is_a_polynomial(P+Q)*
• *∀P Q :C→C, (is_a_polynomial P)→(is_a_polynomial Q)→is_a_polynomial(P*Q)*
• *∀P :C→C, ∀a :C, (is_a_polynomial P)→is_a_polynomial(a*P)*
• *∀P :C→C, ∀n :N, (is_a_polynomial P)→is_a_polynomial(P^n)*
Exercise: is_a_polynomial (nroot 2 3)*X^3+1)
Exercise: ∀a b c:C, is_a_polynomial (a*X^2+b*X+c)

The proofs of the two exercises above are similar. Only two tactics are needed, i.e. *"apply a resource"* and *"easy"*. For example, in the first exercise, the student will apply the resource *"∀P Q :C→C, is_a_polynomial P→is_a_polynomial Q→is_a_polynomial (P+Q)"* and then will have to prove that $\sqrt[3]{2} \cdot X^3$ and 1 are polynomials. Applying the other resource *"∀P Q :C→C, is_a_polynomial P→is_a_polynomial Q→is_a_polynomial (P*Q)"*, he will have to prove that $\sqrt[3]{2}$, X^3 and 1 are polynomials. And so on, he will achieve the proof applying several times the properties of the definition *"is_a_polynomial"*.

To define the algebraic numbers, we have to highlight polynomials which have integer coefficients:

Definition *has_int_coeffs : (C→C)→Prop*
Properties:
- $\forall a$ *:Z, has_int_coeffs a*
- *has_int_coeffs X*
- $\forall P\ Q$ *: C→C, (has_int_coeffs P)→(has_int_coeffs Q)→has_int_coeffs(P+Q)*
- $\forall P\ Q$ *: C→C, (has_int_coeffs P)→(has_int_coeffs Q)→has_int_coeffs(P-Q)*
- $\forall P\ Q$ *: C→C, (has_int_coeffs P)→(has_int_coeffs Q)→has_int_coeffs(P*Q)*
- $\forall P$ *: C→C,* $\forall a$ *:Z, (has_int_coeffs P)→has_int_coeffs (a*P)*
- $\forall P$ *: C→C,* $\forall n$ *:N, (has_int_coeffs P)→has_int_coeffs (P^n)*
- $\forall P$ *: C→C, (has_int_coeffs P)→(is_a_polynomial P)*

Exercise: has_int_coeffs (5*X+1)
Exercise: has_int_coeffs (3*X^3+1)
Exercise: ∀n :N, (has_int_coeffs (X^n))
Exercise: ∀n :N, ∀a b :N, has_int_coeffs (a*X^n+b)

The exercises above are similar to those concerning *"is_a_polynomial"*.

6.6 Algebraic numbers

We may now define what an algebraic number is:

Definition *is_algebraic : C→Prop*
Properties:
- $\forall a$ *:C, (is_algebraic a)= (ex P : C→C, (has_int_coeffs P) and (ex b :C, not (P(b) =0)) and (P(a)=0))*

Exercise: ∀a : Z, (is_algebraic a)
Exercise: is_algebraic (nroot 2 2)
Exercise: is_algebraic (i)

We describe a proof of the first exercise. The student applies the tactic *"Translate the goal"* and obtains the following:

Context: **a:***Z*, **Goal:**$\exists P{:}C{\Rightarrow}C$, (*has_int_coeffs* P) ***and*** ($\exists b{:}C$, $P(b){\neq}0$, ***and*** $P(a){=}0$)

Clicking on the "Exhibit" tactic, the student may choose the value of P to be $X - a$.

Context: **a:***Z*, **Goal:**(*has_int_coeffs X-a*) ***and*** ($\exists b{:}C$, $X\text{-}a(b){\neq}0$, ***and*** $X\text{-}a(a){=}0$)

The tactic *"Translate the goal"* may be used again, and the student has to prove the three subexpressions of the previous goal. The first one, *has_int_coeffs X-a*, is proved as above, using the properties of *"has_int_coeffs"*. The second one, $\exists b{:}C$, $X\text{-}a(b)$), is proved choosing $b = 1 + a$ and simplifying the expression obtained, thanks to the properties of the functions over a field and the operations on a field. The third one, $X\text{-}a(a){=}0$, is proved similarly.

For first-year students, proving the following theorems

Theorem: ∀a b : C, (is_algebraic a)→(is_algebraic b)→(is_algebraic (a+b)
Theorem: ∀a b : C, (is_algebraic a)→(is_algebraic b)→(is_algebraic (a*b)

would be far too hard. However, the exercises below treat examples which may be handled directly, and could give a feeling on the theorem.

Exercise: is_algebraic (i+1)
Exercise: ∀a b : C, (a^2+a+1)=0→(b^2-b-1)=0→(is_algebraic (a+b))
Exercise: ∀a : C, a^5-a-1=0→(is_algebraic (a+1))

Exercise: ∀a : C, aˆ5-a-1=0→(is_algebraic (aˆ2+ (nroot 2 2)*a-2))
Exercise: is_algebraic (2/(1+ (nroot 2 2)
Exercise: is_algebraic (2 + 3*(nroot 3 2)

The student has to find the good polynomials, and replace these in the proofs. These are examples of proof exercises that test in fact calculation.

We now define which elements and subsets of $\mathbb{C}$ are "solvable".

Definition *is_solvable : (partie C)→Prop*
Properties:
• *∀X : (partie C), (is_solvable X)= ((is_subfield X) and (∀a : C, ∀b : X, ∀n : N, (n>0)→(b=aˆn)→(a _appartient_a_ X)))*
Exercise: ∀X Y : partie(C), (is_solvable X)→(is_solvable Y)→(is_solvable (inter X Y))
Definition *is_solvable : C→Prop*
Properties:
• *∀a :C , (is_solvable a)= (∀X : partie(C),(is_solvable X)→(a _appartient_a_ X))*

The above exercise is quite long, but not difficult and may be interesting to do for the student. He has to use every particularity of the two definitions of *"is_subfield"* and *"is_solvable"*, and see that these are propagated to the intersection.

The set of all solvable elements was called "RESOL", and the last exercise of the file was to prove that this set is solvable:

Definition *RESOL : partie(C)*
Properties:
• *∀a :C , (a _appartient_a_ RESOL)=(is_solvable a)*
Exercise: is_solvable (RESOL)

This exercise is also quite long but not too difficult; the student has to show that RESOL satisfies each of the condition of *is_solvable*.

7 Related and future works

The pionneering interface for Coq has been CtCoq [Bertot 99], based on LeLisp, which is no more maintained. Later on, the main features of CtCoq, including "proof by pointing" have been recollected and extended in PCoq [PCOQ], which is implemented in Java. While CtCoq and PCoq were designed specifically for Coq, ProofGeneral [Proofgeneral] is designed for a generic theorem prover, but does not support proof by pointing. CoqIde is a variant of ProofGeneral dedicated and integrated to Coq. PhoX [PHoX] is a theorem prover aiming at a "minimal learning time". It has been used by third-year students in Chambery for several years. Matita [Asperti and al.07] seems to be the latest theorem prover: it is "document-centric" and its two main concerns are efficient interaction with a (presumably large) library and disambiguation of user input (see in particular [Sacerdoti Zacchiroli 04,Sacerdoti Zacchiroli 07]). Coqweb seems to be the first interface designed for teaching, with separate modes for students and teachers. It needs all kinds of improvements:

(i) the student window is far from perfect. In particular, the display of statements could still be much better.

(ii) at the end of a proof, it should offer a trace, which could be accepted by typical teachers as a correct informal proof.

(iii) there is no mechanism for searching resources, which is a problem when, form some unexpedted reason, Coqwed does not find the desired one.

(iv) the teacher window is far from user-friendly. In particular, the way overloading and coercions may be used is not sufficiently clearly specified and error messages are often frustrating.

(v) the library should be completed and the wiki should be better organized.

On the longer range, we also hope that Coqweb could be upgraded and become useful for more and more advanced mathematicians.

References

[Asperti and al.07] A.Asperti, C.Sacerdoti Coen, E.Tassi, S.Zacchiroli. "User Interaction with the Matita Proof Assistant." To appear in the Journal of Automated Reasoning, Special Issue on User Interfaces for Theorem Proving.

[Autexier Sacerdoti Coen 06] S. Autexier, C.Sacerdoti Coen: A Formal Correspondence between OMDoc with Alternative Proofs and the lambda-bar-mu-mu-tilde-Calculus MKM2006, LNAI, 4108, 67–81, 2006.

[Bertot 99] Yves Bertot, "The CtCoq System: Design and Architecture", Formal aspects of Computing, Vol. 11, pp. 225-243, 1999.

[BlogDuB] `http://pcmath165.unice.fr/annamath/spikini/?wiki=LeBlogDuB`

[Coq 06] The Coq proof assistant `http://coq.inria.fr/coq-eng.html`

[Coqweb] `http://pcmath165.unice.fr/wcw/spikini`

[Corbineau] Pierre Corbineau, Declarative Proof Language for Coq `http://www.cs.ru.nl/ corbinea/mmode.html`

[Epp 03] Suzanna S. Epp, The Role of Logic in Teaching Proof, American Mathematical Monthly (110)10, Dec. 2003, 886-899.

[FTA] Herman Geuvers, Freek Wiedijk, Jan Zwanenburg, Randy Pollack, Henk Barendregt, Fundamental Theorem of Algebra `http://www.cs.ru.nl/ freek/fta/`

[Girard 07] J.Y.Girard, Informal conversation, March 14, 2007

[Gonthier Werner 04] Georges Gonthier and Benjamin Werner, Coq Proof of the Four Color Theorem, `http://research.microsoft.com/ gonthier/`

[Luo 99] Z.Luo "Coercive Subtyping" Journal of Logic and Computation.

[MIZAR] The Mizar Home Page `http://mizar.uwb.edu.pl/`

[PCOQ] `http://www-sop.inria.fr/lemme/pcoq/pcoq-fra.html`

[PHoX] `http://www.lama.univ-savoie.fr/ RAFFALLI/phox.html`

[Proofgeneral] `http://proofgeneral.inf.ed.ac.uk/`

[Sacerdoti 06] C. Sacerdoti Coen: Explanation in Natural language of lambda-bar-mu-mu-tilde-terms In Fourth International Conference on Mathematical Knowledge Management (MKM2005), LNAI, Vol. 3863, 234–249, 2006.

[Sacerdoti Zacchiroli 04] Claudio Sacerdoti Coen, Stephano Zacchiroli: Efficient Ambiguous Parsing of Mathematical Formulae MKM04 , LNCS 3119, 347-362, 2004.

[Sacerdoti Zacchiroli 07] Claudio Sacerdoti Coen and Stefano Zacchiroli: Spurious Disambiguation Error Detection, in Proceedings of MKM 2007: . LNAI, to appear.

[WCW] `http://pcmath165.unice.fr/wcw/spikini/?wiki=AccueilWikiCoqWeb`

[Wiedijk 04] Freek Wiedijk, "Integrating procedural and declarative proof", small TYPES workshop, University of Nijmegen, 2004-11-01,

[WIMS07] Deuxime colloque international WIMS Enseigner, créer des ressources avec WIMS 9 - 10 - 11 Mai 2007 Nice Sophia Antipolis.

[Xiao 98] Gang Xiao, WWW Interactive Multipurpose Server, `http://wims.unice.fr/wims/`