Computer Assisted Teaching in Mathematics

C. Raffalli and R. David
Laboratoire de Maths, Université de Savoie
{david,raffalli}@univ-savoie.fr
http://www.lama.univ-savoie.fr

1 Introduction.

The master of mathematics (third and fourth year at the university) of the Université de Savoie (Chambéry) has optional courses in logic [3]. Some years ago, these courses were an introduction to the bases of mathematical logic, but due to the difficulties of the students with reasoning, the goal of the third year course changed gradually to become a training in mathematical reasoning.

The introduction of PhoX [13], the proof assistant of C. Raffalli, allowed to do tutorial sessions (TS) with computers. With the machine, students cannot give incorrect proofs. Errors are replaced by situations where they get stuck and the teacher can help them to progress. Our experience shows that the benefit for the student is greater than to show him the erroneous sentences of its reasoning because he does not always understand the explanations of the teacher.

After four years of experience, this tool seems very useful to us. It shows clearly that, up to the third year (and may be even the fourth) at the university, the main difficulties of the students do not come from the mathematical concepts they discover but rather from their very strong incomprehension of the nature of mathematical reasoning.

Before using PhoX, we give a small course of formal logic to the third year students. But the difference with an informal presentation of mathematical reasoning is very small and it should be possible to use PhoX even with first year students: we will try this experiment during the spring 2002 semester.

In this paper, we describe the main points of our experience. Section 2 quickly gives the context of proofs on machine. Our experiment itself is described in section 3. We give a detailed example in section 4. We conclude by some prospects.

The interested reader will find on the web page of the authors a version of this paper with an appendix giving the main commands of PhoX: this will help him to have a deeper understanding of the example.

2 Proof and computers.

The *proof assistants* like ACL2 [9], COQ [7], HOL [6, 8], Isabelle [11], LEGO [12], PVS [10] ... allow to do mathematical proofs with a computer and their correction are thus guaranteed. This is not only automated deduction: the user can (and generally must) guide the machine in the reasoning. The software just checks that each step of the proof is correct.

Most of the softwares above provide more or less sophisticated algorithms of automatic deduction allowing to finish the proof without the participation of the user. But this aspect does not have much teaching interest because the interesting point is the correctness of each step of a proof. These systems were developed by specialists primarily aiming for applications in formal verification for programs, circuits, communications protocols, etc. Learning to use these systems is not simple and clearly requires too much time for a student in mathematics at the university. Thus, none of these system is suitable for teaching purposes.

As far as the authors know, PhoX is the only proof assistant used in France for teaching mathematics. The ideal tool is a system which is simultaneously sufficiently powerful to allow to do proofs at the first or second year level, and as simple and intuitive as possible. These goals are difficult to reconcile because the power is obtained, in general, by a great number of commands (one for each typical situation) and the training is thus necessarily long.

3 Our teaching experience.

3.1 A few word about logic.

Our third year course [3] presents *first order logic* with its connectives and its quantifiers. We also introduce the rules of *natural deduction*.

This formalism is probably not necessary before using PhoX, but a minimal presentation describing what is a mathematical statement and what is a proof is essential. Indeed:

- The software does not handle sentences in human languages but mathematical formulas. It is important for the student to be able to read them. This symbolic notation does not add difficulties because the students do not see differently the symbol $\forall$ and the expression "for all" once this symbol has been defined (they are still young and used to change notations with different teachers).
- Each *command* of the system will correspond to a precise step of a mathematical reasoning and it is necessary to introduce a minimal vocabulary to explain these commands.

Moreover, and even if this is done (at least in France) most of the time, is it reasonable to teach mathematics without defining first, even informally, what is a proposition and a proof, since they are the basis of all the mathematics? When we do not define these concepts, we enforce a division between the students who discover their meaning by themselves (this is what happened to those who became teachers!) and the others.

In the section below, we give the informal answers to these questions that, in our opinion, are necessary (and sufficient) for a student before he uses PhoX.

3.2 What to tell to students before using PhoX.

What is a mathematical statement ? Statements (or formulas) are built starting from *atomic* facts which depend on the type of mathematics we are doing: they can be an equality, an inequality, the membership of an object in a set, the parallelism of two lines, etc. Generally, these atomic facts are the properties that, at this level, we do not wish to break up into more elementary facts. For example, we can consider that "n is prime" is an atomic fact or we can consider that it is defined using only divisibility as atomic facts.

To build propositions these atomic facts we use connectives and quantifiers: the conjunction ($\wedge$), the disjunction ($\vee$), the implication ($\rightarrow$), the negation ($\neg$), the universal quantification ($\forall$) and the existential quantification ($\exists$).

It is important for students to be able to translate an informal statement into a statement using the symbols above and vice versa. This is far from being immediate.

In our logic course, we do many exercises of translation and our experience shows that they are not useless. It is for example well-known that the various manners of saying $A \to B$ are not understood by the students: "B is necessary condition for A", "A is a sufficient condition to have B", "if A then B", ... This is a difficulty students have to overcome: they must be able to read a statement and to distinguish what is a statement and what is not.

What is a true statement ? A second year student, who just learned that all symmetric matrices can be diagonalized, did not consider for certain that he would never have the "bad luck" to see a counter-example. Another said that a statement is true if it has no counter-example.

This concept of truth (the logician says the *semantics*) is problematic. But, it is not necessary to define it formally. Indeed, we would say $A \wedge B$ is true if and only if A is true and B is true and we just defined the conjunction from itself using an "if and only if" whose role is not so clear! Such a definition (however essential for the logician) neither is significant nor useful for the student because the intuitive meaning of the conjunction or the universal quantification is enough for him. Nevertheless, the meaning of implication and the absurdity reasoning are specific problems on which we have to spend more time.

What is really important is to say how we establish the truth of a statement: by writing proofs!

What is a proof ? A proof is a sequence of steps: each step is an application of a specific *rule*. At each step of a proof, we have a *knowledge list* (usually logician say hypotheses, but this term can be misleading because it seems to denote something that is fixed).

This knowledge list (KL) changes often. At the beginning of a proof, the list is empty or, more precisely, contains all the results given by the teacher (theorems, lemmas, axioms, ...). Some of the rules extend KL. Moreover, if a proof needs to distinguish several cases, at the beginning of the proof of each case all the KLs are identical, but they will evolve independently during each proof.

For each connective and quantifier, there are two rules:

The *introduction rule*, which is used to *prove* a formula starting by this symbol. For instance to prove $A \to B$, we add A to KL and we prove B.

The *elimination rule*, which allows to *use* an hypothesis (a member of KL) starting with that symbol. For instance if $A \vee B$ belongs to KL and if we want to prove C, it is enough to prove $A \to C$ *and* $B \to C$. The fact we use an "and" where there is an "or" disturbs a lot the students !

It could be surprising that this is enough to prove any mathematical result (this is Godël's completeness theorem). This is not very difficult to explain, but many mathematicians are not aware of it !

We give below an informal presentation of all the rules. We give them a name, between parenthesis, to refer to the rule later (it is the name of the connective indexed by i if it is an introduction rule or e if it is an elimination rule). We use the verb "can" in rules to insist on the fact that we can generally apply more than one rule in a given situation.

(ax) This first rule is not associated to any connective. It says that if a statement belongs to KL, then it is proved!

$(\to_i)$ To prove $(A \to B)$, I can add A to KL and prove B.

$(\to_e)$ If $(A \to B)$ and A belong to KL then, I can add B to KL.

$(\wedge_i)$ To prove (A and B), I can prove A and then prove B.

($\wedge_e$) If (A and B) belongs to KL then, I can add both A and B to KL.

($\vee_i$) To prove (A or B), I can prove A or I can prove B.

($\vee_e$) If (A or B) belongs to KL then, to prove C, I can prove ($A \to C$) and ($B \to C$).

($\forall_i$) To prove $\forall x\, A(x)$, I can prove $A(y)$ with a new variable y (i.e. y not used before).

($\forall_e$) If $\forall x\, A(x)$ belongs to KL then, I can add $A(t)$ to KL for any t such that $A(t)$ is a correct statement.

($\exists_i$) To prove $\exists x\, A(x)$, I can prove $A(t)$ for some t such that $A(t)$ is a correct statement.

($\exists_e$) If $\exists x\, A(x)$ belongs to KL, I can add $A(y)$ to KL where y is a new variable (i.e. y is not used before).

($\neg_i$) To prove (non A), I can add A to KL and get a *contradiction* (see below).

($\neg_e$) If (non A) and A belong to KL then, I can prove anything and we say we have a contradiction.

(Abs) To prove A, I can add (non A) to KL and get a *contradiction*. This is known as the absurdity reasoning. Note that this rule is neither an introduction nor an elimination rule.

(other) The axioms for the particular theory we are using: induction on natural numbers, group axioms, ...

3.3 Using PhoX with students.

In our third year course, we start making proofs in the first tutorial session (TS). We begin with formulas without quantifiers. It is, in particular, the occasion to give them some tools: for example, $[(A \wedge B) \to C] \leftrightarrow [A \to (B \to C)]$ which means that to have an assumption in the form of "and" or to have the two assumptions separately is, in fact, the same thing. Then, we introduce formulas using quantifiers. We also begin with simple examples such as $\exists x(A \wedge B) \to \exists xA \wedge \exists xB$. When we ask the students to show (on paper) the converse ... half of them succeeds because they give the same name to the x which satisfies A and the one which satisfies B, even if we had insisted on this problem of name in the $\exists_e$ rule! The fact of being able to show that they badly apply the rule is interesting because they recognize the precise source of their error.

After 4 or 5 TS where we do proofs on paper, we introduce the computer. The students learn rather easily how to use it: two TS are enough.

The use of the computer allows to make more complicated examples that would be impossible otherwise (writing formal proofs, without it, is tedious) but, more importantly, the machine controls the correctness of the proof. Indeed, the student tells the machine which rule he wants to apply: the machine does it... if possible ... and protests otherwise.

The machine does not determine the order of use of the rules, but it helps for using these rules, in particular for the choice of names for variables, which is one of the major difficulties of the students.

3.4 Examples

The main difficulties, for us, are to find examples for which:

- The formalization is very close to the informal reasoning that is given in the course of analysis or algebra.
- The properties of the objects that are considered do not need a complicated axiomatization.
- The reasoning, i.e. the chain of $\forall$ and $\exists$ is, from an educational point of view, interesting.

Note that it is easier to find examples from analysis than from algebra and that these results are often educationally more interesting because the alternation of 3 quantifiers($\forall\varepsilon\exists\alpha\forall x$...) in formulas is one of the main difficulties for the students. Here are the most significant examples they have done during these four years:

1. The intermediate value theorem in $\mathbb{R}$,
2. The uniqueness of the limit,
3. The closure of a union equals the union of the closures (in a metric space),
4. The definition of continuity with strict inequalities (for example $|x - y| < \alpha$) is equivalent to the definition with non strict inequalities (i.e. $|x - y| \leq \alpha$),
5. The image of a connected set by a continuous function is connected (in a metric space),
6. Two permutations of a set E with disjoint supports commute,
7. If the union of two sub-sets of $\mathbb{R}$ is unbounded, then one of them is unbounded,
8. In a ring a prime ideal is irreducible.
9. In a principal ring, an ideal is maximal iff it is prime,

Each of these examples takes a lot of time: 2 hours for the uniqueness of the limit or the closure of a union and more than 4 hours for the intermediate value theorem or the result in a principal ring. It is not the use of the computer which is difficult for them: when we work on these examples they use it easily. It is the details of the proof itself which are not clear for them !

Example (1) was our first experiment four years ago. Even if two groups succeeded in finishing the proof (at home, i.e. without our help), this was clearly too difficult for most of them. Example (9) (treated last fall) has been completely done by most of the students.

The students start their work with a file where the necessary definitions and properties are given (a detailed example is given in section 4.4). In example (1) we also had prepared about 10 lemmas corresponding to the different steps of the proof. In example (9), we had given a short informal proof and stated an intermediate lemma but, in the other examples, the students were left by themselves. Of course we very frequently have to help them when they are stuck. When lemmas are given, they can choose the order in which they are proved, i.e. they can use some of them without proof.

The given examples are such that there is essentially one proof in the sense that various proofs presented in an informal way will probably be considered as identical by any teacher but the length of the proofs made by the students (counted as the number of characters of their file) can change by a factor 2. We usually make a proof whose length is the one of the "best" student also is divided by 2 but this mainly comes from a better use of the system.

There is also variation between proofs where we start from the conclusion, proofs where we start from the hypotheses and mixed proofs. PhoX is quite insensible to this and students really use that freedom.

3.5 The difficulties of the students

The main difficulties, for the students, are the following:

- The fact that, in a proof, the set of hypotheses may change and they must know, at each step of the proof, what can and what cannot be used .

- The confusion between hypothesis and conclusion: we, sometimes, find a student trying to prove some formula which, in fact, is an hypothesis !

- The use of free and bound variables: they do not know clearly which are the objects which, at some point of the proof, "exist".

- The computer does the necessary renaming, but the student may decide, at some point of the proof, to rename x in y even if the name y is already used. This is reasonable, and often done, if we know that y will no more be used. But we had the case of students who found that renaming y in x allowed to use the hypothesis $x = 0$ for y ! They did not understand why the computer refused to prove $x = 0$ even if it was an hypothesis: the machine had not forgotten that there were two distinct x. But it took us some times to understand what they had done.

- The distinction, for the quantifiers, between the introduction and the elimination rules:

  - To show $\forall x\, A$, the only way is to take a *new* x (the machine chooses it, not the student !) and to show A for this x. A problem is that this is not (always) true: we can also use the absurdity rule !
  - To use an hypothesis of the form $\forall x\, A$, it is necessary to give an object t and the student, not the machine, must give it.

- In the proof of a formula of the form $\exists M \forall n$ ... a frequent error is that the students give an M which depends on n. The machine is flexible enough to accept to know M only at the end of the proof (this is often useful in analysis when we "cut the ε") but it protests if the M that is given depends on n. When, for example, the student tells the machine I take $M = 2n + 1$ and the machine does not accept, this helps him to understand his error whereas if the teacher says something as "your M may not depend on n" he does not accept this answer so well.

  On the machine, this happens often when you use the rule $\exists_i$ too soon, i.e. you say that the M we are looking for uses objects that the machine does not "know" yet.

- Even if it is less difficult, the implication also causes some problems. The traditional question on the fact that the formula "false $\to$ true" is true is not a problem here since were are only concerned with proofs not with truth. But it is difficult for the students to imagine that, if we know $\forall x(A[x] \to B[x])$ and we want to prove $B[t]$, it will be enough to prove $A[t]$: a lot of examples are necessary before this becomes "natural" for them.

- To use an hypothesis of a form $\forall x$ twice (a first time with an x introduced somewhere else and a second time with, for example, $f(x)$) is difficult. If the name of x was x_0, it would be simpler !

Generally speaking, it is very hard for the students to prove elementary facts on which, in a course of analysis or algebra, the teacher will spend only a few minutes in the first year and will say "trivial" in the third year. However, the formal aspect of the machine is necessary for the student to fully understand these notions which, we probably have forgotten, are difficult. Here are two specific examples.

- For a proof of the intermediate value theorem we took the standard proof: if $m = Sup\,A$ where $A = \{x \in [a,b] \,/\, \forall y \in [a,x],\ f(y) < c\}$ then $f(m) = c$ and we had divided the proof in about ten lemmas. It took two hours for most of the students to prove the first one: if f is continuous and $f(x) < c$ then there is a neighborhood of x (i.e. for some $\alpha > 0$ ...) on which f is smaller than c. It was also difficult for them to find an upper bound for A. The practical use of the two properties characterizing the least upper bound (it is a bound and there are no smaller bounds) is a difficult exercise, especially when the computer asks for a high-level precision

- In the proof that the formulas C_1 and C_2 below are equivalent

$$C_1 : \forall x \forall e{>}0\, \exists a{>}0\, \forall y(|x-y| < a \to |f(x) - f(y)| < e)$$
$$C_2 : \forall x \forall e{>}0\, \exists a{>}0\, \forall y(|x-y| \leq a \to |f(x) - f(y)| \leq e)$$

  to show $C_1 \to C_2$, it is necessary, after having introduced x and e, to use C_1 with x and e. C_1 gives then an a. This a is not what we are looking for (we should take $a/2$!) but many students take this a and, of course, do not succeed ! We had done this proof in detail (without the computer) during a TS and we gave it, on the computer, for the examination : after an hour, none of the students had finished the proof !

4 A detailed example.

4.1 Introduction to PhoX

To get a system simultaneously powerful and easy to use C. Raffalli has, in PhoX, tempted to limit the commands to a small number, each of them corresponding to easily identifiable situations. These commands are extensible: you do not have to add new commands, you change the behavior of the old ones. This allows to prepare simple exercises because the teacher will have customized the system.

The choice of the logical bases of the system is important. Some systems, like COQ with the calculus of constructions or ACL2 with a logic without quantifier, are too far away from the usual mathematical practice. To remain close to this practice while preserving a sufficient expressive capacity, there are, to date, only two known formalisms : the set theory (ZF) and the higher order logic (HOL). For PhoX, we chose the second solution (it is also the choice of Isabelle) because the *typed* aspect of this system allows to distinguish between the type errors (for example, $\lambda + x$ where λ is a scalar and x a vector) and the errors of reasoning.

Quite complex proofs have already been carried out with PhoX: the proof of the completeness theorem of first order logic, Zorn's lemma, the infinite version of Ramsey's theorem etc. Several tutorials, (among which the one below), are provided with the software.

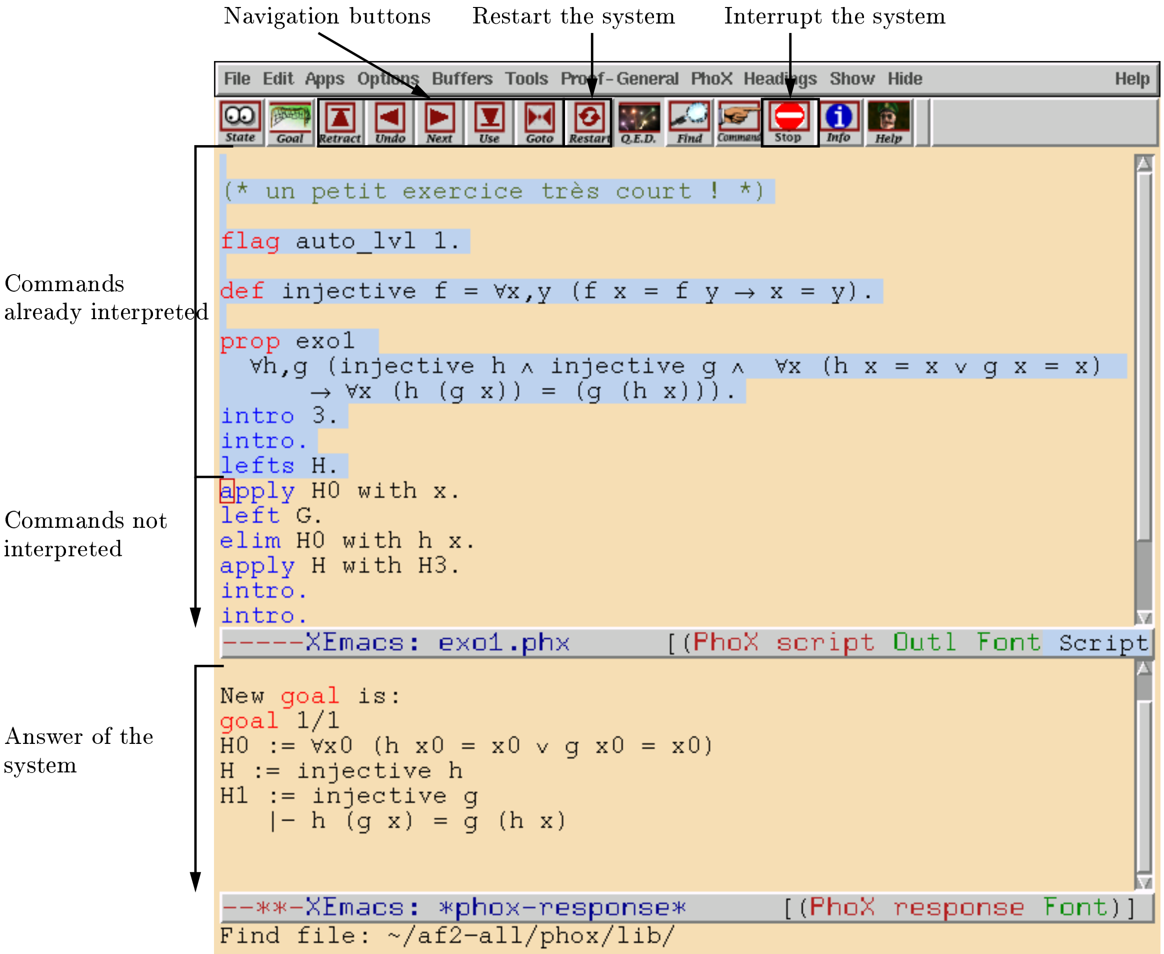

Figure 1: A typical PhoX screen

4.2 The interface.

To make a proof in PhoX, you give *commands* to the system: the current interface is sufficiently user-friendly for third year students but it is not enough for first year ones. Basically, there are two kinds of commands: those (called *global commands*) which allow to make definitions or to customize the system and those which are used to do proofs.

The interface is made with XEmacs [1] and ProofGeneral [14] of D. Aspinall. Figure 1 shows an example of PhoX screen. The screen is divided into two parts: the upper part contains the *script* of the commands given to the system and the lower part, PhoX's response to the last command.

The commands already interpreted are given with a different color. Buttons of navigation allows to go (back and forth) at the desired point. Going back (by using the navigation buttons) is the only way to modify a command already interpreted: this allows an intuitive use of the system while guaranteeing a coherence between the script and the state of the system.

The prompt `>PhoX>` indicates that the machine waits for a command. It is transformed into `%PhoX%` when the system is inside a proof.

4.3 The syntax.

PhoX uses a *functional notation* (also used in languages like LISP or CAML) : for example $\mathrm{d}\,x\,y$ for $\mathrm{d}(x,y)$ or $\mathrm{Im}\,f\,y$ for $y \in \mathrm{Im}(f)$ or open $U = \forall x(U\,x \rightarrow \exists e{>}0\,\forall y(d\,x\,y < e \rightarrow U\,y))$.

This notation is not a problem for the students because it is very uniform. Unfortunately, it is not the case for some of our colleagues (not logicians)! It would be very easy to define an infix symbol for the membership relation (this is the main missing symbol) to be able to write $y \in \mathrm{Im}(f)$ instead of $\mathrm{Im}\,f\,y$ but this is really unnecessary with students.

The notations for the formulas do not introduce particular problems because they use traditional mathematical symbols (except conjunction and disjunction). PhoX suggests useful abbreviations such as $\forall x, y\,A$ for $\forall x \forall y\,A$, $\forall x{\in}A\,B$ for $\forall x(A\,x \rightarrow B)$ or $\exists x{<}y\,B$ for $\exists x(x < y \wedge B)$. PhoX also uses traditional priorities to avoid numerous parentheses, with a little exception which is less common: $A \rightarrow B \rightarrow C$ should be read $A \rightarrow (B \rightarrow C)$.

4.4 An example of proof.

The example given below is typical of those which can be done with the students. It has been treated by them.

We prove that two definitions of the continuity of a function are equivalent: this equivalence, obvious for the teacher, is not at all immediate for the students. We give only one of the directions, the other is similar. We have written it in a rather elaborate way in order to show the possibilities of the system. In practice, the students make longer proofs by breaking up some commands with more elementary ones (cf. the remark at the end of the example). Note that the first part of the example is prepared by the teacher: the work of the student begins only after the command `goal`.

- We define the sort of reals.
  `>PhoX> Sort real.`

- We give predicate for inequalities.
  ```
  >PhoX> Cst Infix[5] x "<=" y : real -> real -> prop.
  >PhoX> Cst Infix[5] x "<" y : real -> real -> prop.
  >PhoX> def Infix[5] x ">" y = y < x.
  >PhoX> def Infix[5] x ">=" y = y <= x.
  ```

- as well as a symbol for the distance and the real 0 (denoted by $R0$).
  ```
  >PhoX> Cst d : real -> real -> real.
  >PhoX> Cst R0 : real.
  ```

- Here are the two definitions of the continuity :
  `>PhoX> def continuous1 f x =`
  $\forall e{>}R0\,\exists a{>}R0\,\forall y(\mathrm{d}\,x\,y < a \rightarrow \mathrm{d}(fx)(fy) < e)$.
  `>PhoX> def continuous2 f x =`
  $\forall e{>}R0\,\exists a{>}R0\,\forall y(\mathrm{d}\,x\,y \leq a \rightarrow \mathrm{d}(fx)(fy) \leq e)$.

- and the lemmas needed for the proof.
  `>PhoX> claim lemme1` $\forall x, y(x < y \rightarrow x \leq y)$.
  `>PhoX> claim lemme2` $\forall x{>}R0\,\exists y{>}R0 \forall z(z \leq y \rightarrow z < x)$.

- We begin the proof.
  `>PhoX> goal` $\forall x, f(\mathrm{continuous1}\,f\,x \rightarrow \mathrm{continuous2}\,f\,x)$.
  `goal 1/1`
  `|-` $\forall x, f(\mathrm{continuous1}\,f\,x \rightarrow \mathrm{continuous2}fx)$

- We start with some introductions.
  `%PhoX% intro 5.`
  `goal 1/1`
  `H :=` continuous1 $f\,x$
  `H0 :=` $e > R0$
  `|-` $\exists a{>}R0\,\forall y(\mathrm{d}\,x\,y \leq a \to \mathrm{d}(fx)(fy) \leq e)$

- We use the continuity of f with e, and we remove the hypotheses H and H0 which will not be used anymore.
  `%PhoX% apply H with H0. rmh H H0.`
  `goal 1/1`
  `G :=` $\exists a{>}R0\,\forall y(\mathrm{d}\,x\,y < a \to \mathrm{d}(fx)(fy) < e)$
  `|-` $\exists a{>}R0\,\forall y(\mathrm{d}\,x\,y \leq a \to \mathrm{d}(fx)(fy) \leq e)$

- We *de-structure* hypothesis G by indicating that we want to consider all the $\exists$ and all the conjunctions (the students use `lefts G` twice with no more indication).
  `%PhoX% lefts G $∃ $∧.`
  `goal 1/1`
  `H :=` $a > R0$
  `H0 :=` $\forall y(\mathrm{d}\,x\,y < a \to \mathrm{d}(fx)(fy) < e)$
  `|-` $\exists a_0{>}R0\,\forall y(\mathrm{d}\,x\,y \leq a_0 \to \mathrm{d}(fx)(fy) \leq e)$

- We use the second lemma with H and we remove it.
  `%PhoX% apply lemme2 with H. rmh H.`
  `goal 1/1`
  `H0 :=` $\forall y(\mathrm{d}\,x\,y < a \to \mathrm{d}(fx)(fy) < e)$
  `G :=` $\exists y{>}R0\,\forall z{\leq}y\ z < a$
  `|-` $\exists a_0{>}R0\,\forall y(\mathrm{d}\,x\,y \leq a_0 \to \mathrm{d}(fx)(fy) \leq e)$

- We de-structure again G and we rename the variable y created.
  `%PhoX% lefts G $∃ $∧. rename y a'.`
  `goal 1/1`
  `H0 :=` $\forall y(\mathrm{d}\,x\,y < a \to \mathrm{d}(fx)(fy) < e)$
  `H1 :=` $a' > R0$
  `H2 :=` $\forall z{\leq}a'\ z < a$
  `|-` $\exists a_0{>}R0\,\forall y(\mathrm{d}\,x\,y \leq a_0 \to \mathrm{d}(fx)(fy) \leq e)$

- Now we know what is the a_0 we are looking for. We do the necessary introductions for $\forall$, $\exists$, conjunctions and implications (again, the students will use `intros` several times with no more indication). Two goals are created, as well as an existential variable (denoted by `?1`) for which we have to find a value.

  `%PhoX% intros $∀ $∃ $∧ $→.`
  `goal 1/2`
  `H0 :=` $\forall y(\mathrm{d}\,x\,y < a \to \mathrm{d}(fx)(fy) < e)$
  `H1 :=` $a' > R0$
  `H2 :=` $\exists z{\leq}a'\ z < a$
  `|-` $?1 > R0$
  `goal 2/2`
  `H0 :=` $\forall y(\mathrm{d}\,x\,y < a \to \mathrm{d}(fx)(fy) < e)$
  `H1 :=` $a' > R0$
  `H2 :=` $\forall z{\leq}a'\ z < a$
  `H3 :=` $\mathrm{d}\,x\,y \leq ?1$
  `|-` $\mathrm{d}(fx)(fy) \leq e$

- The first goal is solved with the hypothesis H1 indicating this way that `?1` is a'. The second is automatically solved by PhoX by using lemma1, and this finishes the proof.
  `%PhoX% axiom H1. auto +lemme1.`

Remark. Instead of the command `auto +lemme1` one could also say `elim lemme1. elim H0. axiom H3.` or `apply H0 with H3. elim lemme1 with G.` where `G` is an hypothesis produced by the first command.
A good exercise for the reader consists in understanding what these commands do. The appendix, available on the long version of this article on our web page, should help you !

5 Conclusion.

At the end of the semester, we ask the students their opinion on the interest of this training. This small investigation seems to show that the experiment is appreciated ... even if we do not use demagogy because the scores the students obtain at the examination do not have anything exceptional. It is also important to note that this experience is done with students in math (not computer science!) and, at the beginning, many can hardly use a computer.

5.1 Questionnaire

The goal of the logic course was twofold

goal 1: to help you (by formalizing the concept of proof) to make correct proofs (also in the other courses) in particular by correctly handling the concepts of free or bound variables and the rules of introduction and elimination of the implication and the quantifiers.

goal 2: to do a bit of logic as such, i.e.

- to help you to understand the difference between syntax (proofs) and semantics (the concept of structure and truth)
- to show you how the basic concepts in math (integers and sets) can be introduced.

Would you like to answer the following questions. Their purpose is the improvement of this course for the next year. Answer to each question by a note from 0 to 5:

1. Do you think (independently of your final score) that, for you, goal 1 was achieved ? (0 = not at all, 5 = completely)
2. Same question for goal 2 ? (0 = not at all, 5 = completely)

   The purpose of the work on computers was to help you to reach goal 1.
3. Do you think it actually helped you ? (0 = not at all, 5 = completely)
4. Do you think we spent too much time with the computer ? (0 = too much, 5 = not enough)
5. Was it difficult for you to learn how to use PhoX ? (0 = very difficult, 5 = very easy)

5.2 Results

The table below gives the averages obtained to each question. They are of course only averages which hide the fact that some answers are 5... and some others are 0.

Question	1	2	3	4	5
98/99	2.3	2.3	2.5	2.3	1.9
99/00	2.8	2.1	2.9	3.5	3.1
00/01	3.2	2.8	3.3	3.3	3.0
01/02	3.3	3.0	2.8	3.1	3.0

5.3 For the future

Even if the results are not spectacular, this experiment appears extremely positive: it helped us to better understand where the major difficulties of the students are and it made them progress. For better results, it would undoubtedly be necessary to be able to spend much more time and to start this kind of work at the beginning of the undergraduate studies. Our experiment shows that the acquisition of the very simple (for us !) reasonings that are necessary to prove the uniqueness of the limit needs a long time for the students. Since one cannot spend one hour in the calculus course to show this result, this training should be made elsewhere. When the process of imitation of the teacher goes well (this has been the case for us when we were students), that is enough but it seems that, for the majority of the students, this process is not enough any more. Will we succeed to convince our colleges of that ?

The students can use PhoX by their own. They can thus do detailed proofs of results given in the other courses. But we must be realistic: one should say “they could do” because, in fact, they do not. They have to understand heaps of other incomprehensible things! For the immediate future, we will

- Try to improve the interface to make the software user-friendly, in particular for first year students. But it is a very big work... and it is very little recognized as a real work by our community!
- Try the same experiment with first year students.

References

[1] *The XEmacs editor*. `www.xemacs.org`.

[2] L. Damas et R. Milner. Principal type schemes for functional programs. In *Ninth Annual ACM Symposium on the Principles of Programming Languages*, pages 207–212. ACM, 1982.

[3] René David, Karim Nour, et Christophe Raffalli. *Introduction à la logique*. Masson, 2001.

[4] Yannis Delmas-Rigoutsos et René Lalement. *La Logique ou l'Art de raisonner*. Le Pommier, 2000.

[5] Gilles Dowek. *La Logique*. Flammarion, 1995.

[6] John Harrison. *HOL Light*. `www.cl.cam.ac.uk/Research/HVG/HOL`.

[7] INRIA. *The Coq Proof Assistant*. `coq.inria.fr`.

[8] Slind Konrad et al. *HOL98*. `www.cl.cam.ac.uk/Research/HVG/HOL`.

[9] J. Strother Moore et al. ACL2. `www.cs.utexas.edu/users/moore/acl2`.

[10] Sam Owre et Natarajan Shankar. *PVS*. `pvs.csl.sri.com`.

[11] Larry Paulson et al. *Isabelle*. `www.cl.cam.ac.uk/Research/HVG/Isabelle`.

[12] Randy Pollack. *LEGO*. `dcs.ed.ac.uk/home/lego`.

[13] Christophe Raffalli. *PhoX*. `www.lama.univ-savoie.fr/~RAFFALLI/phox.html`.

[14] Kleymann Thomas, David Aspinall, et al. *Proof General*. `zermelo.dcs.ed.ac.uk/~proofgen`.