CtCoq: a system presentation

Janet Bertot and Yves Bertot

INRIA Sophia-Antipolis
B.P. 93, F-06902 Sophia Antipolis Cedex, France
{Yves.Bertot,Janet.Bertot}@sophia.inria.fr
tel: (+33) 93 65 77 39

Abstract. The CtCoq system is a graphical user-interface using a distributed architecture adapted to the Coq proof system [3]. Basic features provided by this graphical interface are direct manipulation of formulas and commands using the mouse, mathematical notations with an extended character set and colors, menus for guiding users in their manipulations of commands and formulas. More advanced features also include textual explanation of proofs, proof-by-pointing, and script management.
keywords: system presentation, Application

1 Introduction

It is our belief that proof systems should become an important component of future software and hardware system developments. So far, theorem proving has been limited to experts who are ready to make the effort to learn the behavior of proof systems and their arcane notations. Integrating proofs systems in user-friendly environments is a crucial issue if software engineers are to use these tools on a daily basis. We have concentrated on the user-interface part of proof engine development, by developing this user-interface as a different process, communicating with the proof engine using a simple protocol. The user-interface itself has been developed using a program environment generator, the Centaur system [1, 2].

We believe this paradigm of independent development of user-interface and proof engines as programs integrated in a multi-process, possibly distributed, system has an impact on the future directions of research in the domain of interactive proof systems, as it makes possible the fast integration of research progress in both the automated deduction and man-machine interface domains.

2 Basic features

2.1 Structured editing

The basic elements of the graphical interface are editing windows where the commands and formulas are displayed and edited in a structured way. This means that the data is almost always considered as trees, where the structure reflects the syntax of the language. By comparison with plain textual editing, this makes the syntactic structure more apparent and makes it possible to have more contextual help during the editing process. Another advantage is the possibility to associate special notations to some tree patterns, so that notations can be tuned to usual mathematical practice, with a wide variety of fonts and colors.

in: *Computer Aided Deduction (CADE),* New Brunswick, USA, July, 1996.

Instead of a notion of current position in a document represented by a single character cursor, the editing windows make it possible to have a whole sub-structure as a *current selection*. Editing operations are then related to this current selection. For instance, the position of the current selection is used to provide menu-guided context help. For this tool, a menu is provided to the user, whose content is updated every time the current selection moves, to propose syntactic constructs that are adapted to the syntactic context of this current selection.

For efficient data input, the CtCoq user-interface also accommodates more traditional text editing. At any time, the user can select a structure and edit it as a text fragment, meanwhile forgetting the structure information. Coming back to a structured representation is then necessary, and can be done through a parse operation. This integration of textual editing also makes it possible to communicate with other X-Windows applications through the usual textual selection mechanism. Textual fragments can be received from other applications and parsed for insertion into the editing windows.

2.2 Pretty printing and notations

We use the Centaur displaying mechanism to provide multiple fonts and colors display, with extensibility and incrementality. This display mechanism is parameterized by specifications composed of sets of rules, where each rule associates a pattern with a layout description.

By comparison with other systems, the CtCoq system has two important features. First, the notations are not chosen through a regular expression filtering mechanism, which makes the mechanism more powerful than what can be done with text editors as in TkHOL or Emacs environments. Second, the layout is computed incrementally and it is still sensitive to the mouse, in the sense that it is possible to select sub-expressions of a formula, to copy, paste, or trigger commands with. This makes it more practical to use than layout approaches that use the LaTeX-xdvi chain, where the output is inert.

2.3 Multiple window organization

The opportunity to have several windows makes it possible to arrange the input and the output of the proof engine in elaborated ways. The CtCoq user-interface takes advantage of these possibilities and organizes the data related to a session with a proof engine in a logical way. The main interactor with the proof engine is a window composed of three main sub-parts: a script window where the commands produced by the user are stored, a goal window where the goals addressed in proofs are displayed, a context window where the result of searches in the theorem database are displayed. The goal window is central in this organization, the script window is on top and the context window is on the bottom. Thus, all the important data is concentrated in the middle of the screen: the most important command in the script, the latest, appears at the bottom of the script window, next to the goal window. Also, goals contain a conclusion and a local context. Our organization makes that the global context appears as a simple extension of the local context of each goal.

The central position of the goal window is even more emphasized by the *proof-by-pointing* feature. Using this capability, the user can perform complete proofs by simply clicking at important sub-expressions in the goals.

This multiple window organization also makes it possible to perform several proofs at the same time, with one script window and one goal window associated to each proof. This is especially useful to prove auxiliary lemmas related to a main proof, which is left unfinished while the user concentrates on the lemma. Figure 1 shows an example of the proof window associated to a window where proof text is displayed.

3 Advanced features

The CtCoq user-interface also provides elaborate solutions to problems often encountered in the usage of interactive proof systems. In this section, we present three of these features. *Proof-by-pointing* is concerned with inputting the basic commands. It shows that in the domain of proof manipulation, the mouse can be used to guide a symbolic system in a much more clever way than a simple push-button user-interface. *Script management* is concerned with the support that can be given to a user who tries different solutions to a problem and wants to record his successful attempts. *Textual explanation of proof* provides support to a user who wants to produce documents from his proofs that any mathematician can read, even a mathematician who does not know the Coq system.

3.1 Proof-by-pointing

In the Coq system, most proofs are done interactively in a goal directed style. In this mode of operation, the user first states a goal to prove and then applies commands that reduce this goal to a number of simpler subgoals. Some of the commands have a very elaborate behavior, while others perform very simple logical book-keeping. *Proof-by-pointing* is a method to ensure that the book-keeping commands can be very easily triggered and composed, to make their use less tedious.

The idea of proof-by-pointing is that selecting a position in a goal formula can be interpreted as a command to bring the selected sub-formula to the surface of the goal. In practice, the behavior of proof-by-pointing relies heavily on the ability of the graphical interface to construct graphical representations of logical formulas that are sensitive to the mouse. With this algorithm, the user of the CtCoq environment ends up performing a lot of proofs using mostly the mouse to select relevant parts of goals, and sometimes choosing a more elaborate command from a menu.

3.2 Script management

The CtCoq proof environment supports the activity of looking for a proof. In this mode, the user may try several possibilities before finding the right formalization and the right sequence of inference steps. His work is then a succession of steps forward and backward. Script management is a tool to keep only the successful attempts, discarding the failed ones.

Script management works by separating the window containing the commands sent to the proof system into several areas. One area (called the final area) contains the commands that have already been sent to the proof system, another (the buffer area) contains the command being executed, and a third (the scratch area) contains just the commands being edited by the user. When a command is undone, it is removed from the final area and returned to the scratch area.

The various areas are made visible by different colors. This way the user can see at a glance the current state of his work. This is especially useful when modifying the formalization of a problem and trying to recuperate large amounts of work that were valid in the formalization and that need to be checked again the new formalization.

3.3 Textual Explanation of Proofs

The proof scripts written to check a proof under Coq actually represent the proof but they are unreadable to people who don't have a good knowledge of the proof system's behavior. Fortunately, the Coq system also constructs a typed λ-term that represents the basic inferences used to prove a mathematical result. We are working on pretty-printing facilities that produce formalized English text from these proof terms. With a simple keystroke, it is thus possible to open a window containing the English text describing the proof of a theorem, or even the incomplete proof currently being worked on. Figure 1 contains a window with an example of the text produced to explain a proof.

4 Availability

The CtCoq system is available on Sun and Dec Alpha workstations. Information on how to retrieve this system is available at the following WWW address:

`http://www.inria.fr/croap/ctcoq/ctcoq-eng.html`

References

1. P. Borras, D. Clément, T. Despeyroux, J. Incerpi, G. Kahn, B. Lang, and V. Pascual. Centaur: the system. In *Third Symposium on Software Development Environments*, 1988. (Also appears as INRIA Report no. 777).
2. D. Clément. A distributed architecture for programming environments. *Software Engineering Notes*, 15(5), 1990. Proceedings of the 4th Symposium on Software Development Environments.
3. C. Cornes, J. Courant, J.-C. Filliâtre, G. Huet, P. Manoury, C. Muñoz, C. Murthy, C. Parent, C. Paulin-Mohring, A. Saïbi, and B. Werner. *The Coq Proof Assistant User's Guide*. INRIA, May 1995. Version 5.10.

This article was processed using the LaTeX macro package with LLNCS style

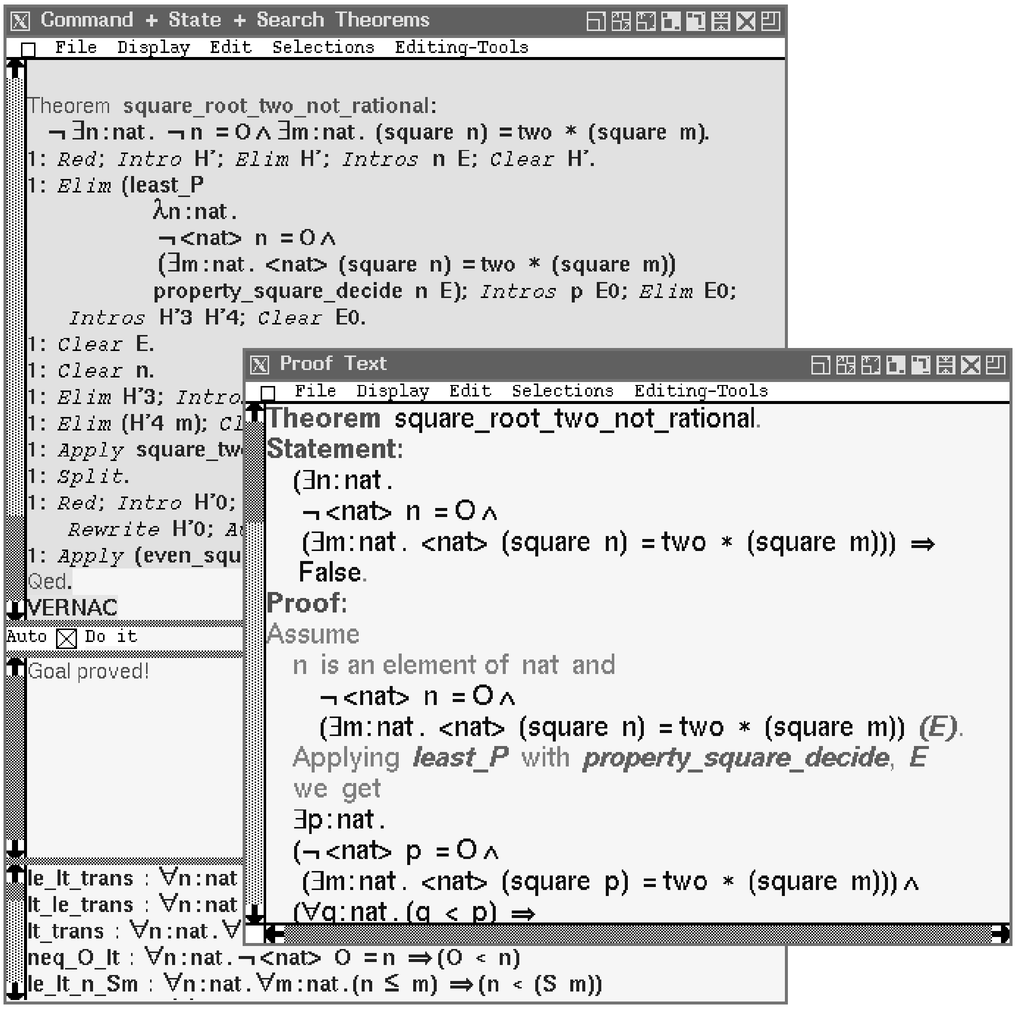

Fig. 1. Windows of the CtCoq graphical interface