Proof General: A Generic Tool for Proof Development

David Aspinall

October 1999

LFCS, University of Edinburgh, U.K.
`http://www.dcs.ed.ac.uk/home/da`

Abstract. This note describes *Proof General*, a tool for developing machine proofs with an interactive proof assistant. Interaction is based around a *proof script*, which is the target of a proof development. Proof General provides a powerful user-interface with relatively little effort, alleviating the need for a proof assistant to provide its own GUI, and providing a uniform appearance for diverse proof assistants.

Proof General has a growing user base and is currently used for several interactive proof systems, including Coq, LEGO, and Isabelle. Support for others is on the way. Here we give a brief overview of what Proof General does and the philosophy behind it; technical details are available elsewhere. The program and user documentation are available on the web at `http://www.dcs.ed.ac.uk/home/proofgen`.

1 Background

Proof General is a generic interface for interactive proof assistants.

A *proof assistant* is a computerized helper for developing machine proofs. There are many uses for machine proofs, both during the specification, development, and verification of software and hardware systems, and in the development and teaching of mathematical proof and formal logic. Proof General helps with developing proof scripts.

A *proof script* is a sequence of commands sent to a proof assistant to construct a machine proof. A script is usually stored in a file. Proof General uses a technique called *script management* to help the user write a proof script without using cut-and-paste or repeatedly typing "load file" commands. Proof General has a sophisticated implementation of script management which covers large developments spread across multiple files.

A guiding philosophy behind Proof General is to provide an interface which is useful to novices and expert-users alike. Some interfaces for theorem provers are aimed at novices and become infeasible for large developments; others are aimed at experts but have steep learning curves or require changes in work methods, discouraging their take-up. With this in mind, Proof General builds on the programmable text editor *Emacs*, the powerful everyday editor of many Unix users (and available on several other operating systems).

Another important aspect of Proof General is that it is *generic*. It provides a uniform interface and interaction mechanism for different back-end proof assistants. This generic aspect is no empty claim or untested design goal; Proof General is already in use for three different proof assistants: Coq, LEGO, and Isabelle. The present implementation is oriented towards proof assistants based on an interactive command interpreter (or *shell*), where interaction consists of a dialogue between the user and the system.

To summarize, Proof General provides a fairly elaborate yet unobtrusive interface, which gives the user many useful features.

2 Features of Proof General

Simplified communication. The proof assistant's shell is hidden from the user. Communication takes place via two or three buffers (Emacs text widgets). The *script buffer* holds input, the commands to construct a proof. The *goals buffer* displays the current list of subgoals to be solved. The *response buffer* displays other output from the proof assistant. The user sees only the output from the latest proof step, rather than a screen full of output. Nonetheless, the user can still access the shell to examine it or run commands.

Script management. Proof script editing is connected to the proof process, maintaining consistency between the edit window and the state of the proof assistant. Visual feedback on the state of the assistant is given by colouring the background of the text in the editing windows. Parts of a proof script that have been processed are displayed in blue and moreover can be *locked* to prevent accidental editing. Parts of the script currently being processed by the proof assistant are shown in red. The screenshot in Figure 1 shows script managament in action.

Multiple file handling. Script management works across multiple files. When a script is loaded in the editor, it is coloured to reflect whether the proof assistant has processed it in this session. Proof General communicates with the assistant to discover dependencies between script files. If I want to edit a file which has been processed already, Proof General will *retract* the file and all the files which depend on it, unlocking them. Thus the editor is connected to the theory dependency or make system of the proof assistant.

Proof by Pointing. Clicking on a subterm of a goal can apply an appropriate rule or tactic automatically, or display a menu of choices. (Proof General relies on support in the assistant for this feature, since it is quite prover-specific).

Syntax highlighting and symbol fonts. Proof scripts are decorated: commands are highlighted and different fonts can be used for definitions and assumptions, for example. Symbol fonts can be used to display proper glyphs for logical operators, Greek letters, etc, which occur throughout mathematical proofs.

Toolbar and menus. A toolbar includes buttons for examining the proof state, starting a proof, manoeuvring in the proof script, saving a proof, searching for a theorem, interrupting the assistant, and getting help. A menu gives access to further commands, and a useful collection of user preferences. Using the toolbar, you can replay proofs without knowing any low-level commands of the proof assistant or any Emacs short-cuts.

Tags and definitions menu. Using a TAGS file, one can quickly locate the definition and uses of an identifier, automatically searching many files. Using a definitions menu, one can quickly navigate within a proof script to find particular definitions, declarations and proofs.

Remote proof assistant. A proof assistant can be run remotely, perhaps across the internet, while Proof General and the proof script reside locally.

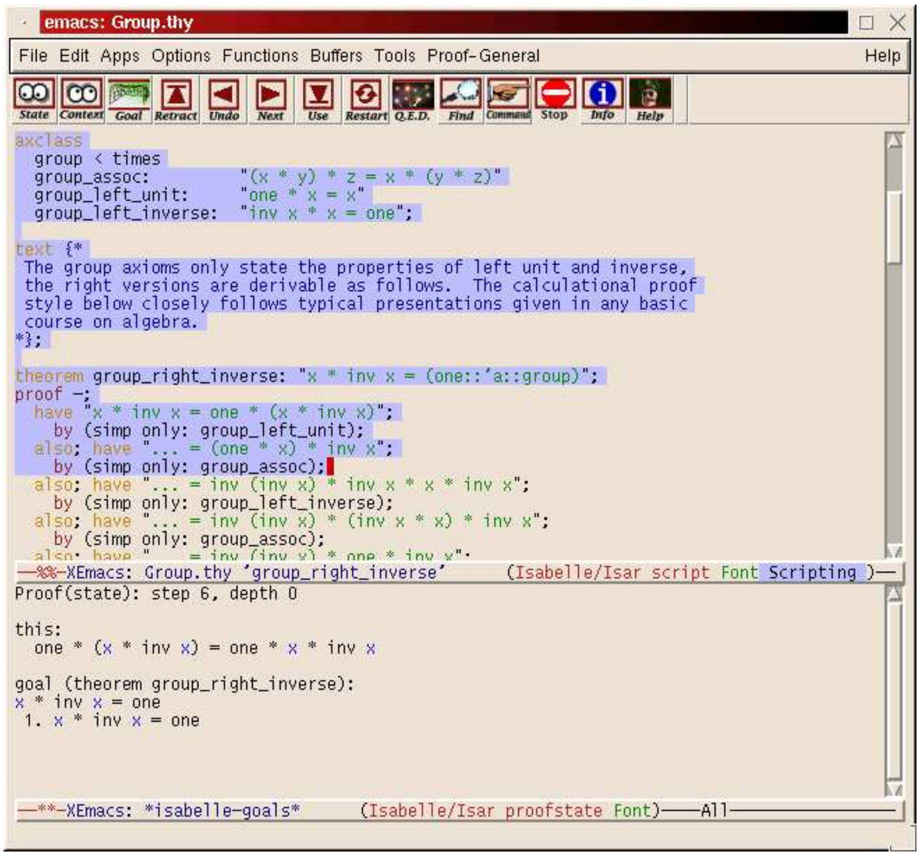

Fig. 1. Using Proof General

3 Proof General in Use

Figure 1 shows a screenshot of Proof General running in a single window on the screen. The window is split into two parts. The upper half displays the proof script Group.thy which is being processed. This is a script written for Isabelle using the new Isar proof language [4]. The lower half displays the current list of subgoals which are to be solved to complete the proof. Instead of this split window, it is perfectly possible to have separate windows on the screen, as the user likes. Proof General is even happy to run on a plain console, although graphical facilities will be reduced (e.g. no toolbar).

In the script file, the cursor appears at the end of the *locked* region, which has a blue background to indicate it has already been processed. The arrow buttons on the toolbar are used to manipulate the locked region, by sending commands to the proof assistant, or by issuing undo steps. In this manner, a user can *replay* a proof interactively, without needing to know any low-level commands needed to start the proof assistant, or issue proof and undo steps. And without the extreme tedium of cut-and-paste.

4 Further Details

Technical references. Proof General has a detailed user manual [1] which also contains instructions for instantiating it to new proof assistants. The ideas of script management and proof by pointing were adapted from the CtCoq system [3]; proof by pointing in Proof General is described in an LFCS technical report [2]. (Proof General goes beyond CtCoq in some ways, but is less sophisticated in others; the biggest difference is that CtCoq provides its own GUI based on structure editing.)

Implementation. Proof General is implemented in Emacs Lisp. There is a generic core (about 7000 lines) which implements the toolbar, menus, script management, and process handling features. Each supported proof assistant has some additional Emacs Lisp (30 – 500 lines) for prover-specific configuration: setting regular expressions and command strings, and perhaps providing extra features. For robust operation and features like proof by pointing, the proof assistant may need modification to output special messages for Proof General.

Availability and System requirements. Proof General is easy to install, and is available free of charge (with sources and documentation) for research and educational use. The current release is Proof General 3.0. For best results, it requires a recent version of XEmacs (21.1 or later), alongside recent versions of one or more proof assistants: Coq (6.3 or later), Isabelle (version 99 or later), or Lego (version 1.3.1). Details of where to obtain these components can be found on the web page mentioned below. Success is guaranteed with a Unix (or Linux) environment, although XEmacs and some proof assistants are available for other operating systems, and there is nothing operating system specific in Proof General itself.

Acknowledgements. Many people have contributed to the design and code, both in the generic basis and for the prover-specific instances. For each instance of Proof General, we try to encourage somebody familiar with the proof assistant to develop and maintain the prover-specific code, perhaps also enhancing to the generic basis. In order of appearance, the main workers on Proof General have been: T. Kleymann, Y. Bertot, D. Sequeira, H. Goguen, D. Aspinall, P. Loiseleur, M. Wenzel, P. Callaghan. Many other people provided useful feedback, including: P. Brisset, R. Burstall, M. Hofmann, J. McKinna, and D. von Oheimb. Thomas Kleymann originated the Proof General project. David Aspinall is the present manager.

For more information about Proof General, please visit the home page at `http://www.dcs.ed.ac.uk/home/proofgen`.

References

1. D. Aspinall, H. Goguen, T. Kleymann, and D. Sequira. Proof General. System documentation, see `http://www.dcs.ed.ac.uk/home/proofgen`, 1999.
2. Yves Bertot, Thomas Kleymann, and Dilip Sequeira. Implementing proof by pointing without a structure editor. Technical Report ECS-LFCS-97-368, University of Edinburgh, 1997. Also published as Rapport de recherche de l'INRIA Sophia Antipolis RR-3286.
3. Yves Bertot and Laurent Théry. A generic approach to building user interfaces for theorem provers. *Journal of Symbolic Computation*, 25(7):161–194, February 1998.
4. Markus Wenzel. Isar — a generic interpretative approach to readable formal proof documents. In Y. Bertot, G. Dowek, A. Hirschowitz, C. Paulin, and L. Thery, editors, *Theorem Proving in Higher Order Logics, 12th International Conference, TPHOLs'99*, Lecture Notes in Computer Science 1690. Springer-Verlag, 1999.