Focus Windows: A New Technique for Proof Presentation*

Florina Piroi and Bruno Buchberger

Research Institute For Symbolic Computation,
4232 Hagenberg, Austria
{fpiroi,buchberg}@risc.uni-linz.ac.at

Abstract. Whether they are hand written or generated by an automated prover, long proofs may be difficult to understand and follow. The main reason for this is that at some point in the proof formulae that occur lines, paragraphs or even pages before, are used. The proof presentation method proposed here tries to overcome this by showing, in each proof step, exactly the formulae that are relevant for the particular proof step. We describe the implementation of this method in the frame of the *Theorema* system.

1 Introduction

Proofs in mathematical publications are linear texts. We view them as sequences of proof steps, i.e. at each step a new formula is derived from formulae appearing in earlier steps, by some proof technique. In long proofs, the used formulae may occur a couple of lines, paragraphs, or even pages distant from the place where they are used. These formulae are usually referred by labels and the reader has to jump back and forth between the referenced formulae and the proof step in which they are needed. This is unpleasant and makes understanding proofs quite difficult even if the proofs are nicely structured and well presented.

Most automated theorem provers do not put emphasis on producing proofs that are easy to read and understand. (A telling illustration of this is given in [7].) Even those which provide tools for studying proofs (as, for example, the Omega system [5]) have the problem described above.

From the outset, in *Theorema* [2] we tried to emphasize on attractive proof presentation. *Theorema* proofs are designed to resemble proofs generated by humans, i.e. they contain formulae and explanatory text in english. In addition, *Theorema* provides various tools that help the reader in browsing proofs: references to formulae are realized as hyper–links that display the referenced formula in a small auxiliary window; nested brackets at the right–hand margin make contracting entire sub–proofs to just one line possible; various color codes distinguish the (temporary) proof goals from the (temporary) knowledge base

* partially supported by the RISC PhD scholarship program of the government of Upper Austria and by the FWF (Austrian Science Foundation) SFB project P1302

formulae; etc. Still, reading and understanding long linear proofs is difficult even for proofs generated by the typical *Theorema* provers.

Focus windows provide means to overcome this difficulty. The idea of focus windows as a technique for proof presentation was introduced in [1] and is as follows: Starting from the root of a proof object, in each proof step, one analyzes which formulae are used and which ones are produced. Then, a window containing exactly these formulae for the proof step that is being analyzed is composed. The window also contains buttons for moving to, and analyzing the next proof step in the proof. For proof steps that branch into two or more sub–proofs the subsequent windows are displayed in contracted form the user being allowed to decide which one to open next.

In the following we give some comments on the Focus Windows from the user's point of view. In Section 3, we briefly describe the implementation of the Focus Windows technique in *Theorema* and then present the final conclusions.

2 Using Focus Windows

In this section we try to shortly present the Focus Windows from the user's side.

A typical call for starting a *Theorema* prover to work on a proof problem looks like this:

Prove[Lemma["Lm"], using → KnowledgeBase, by → SomeProver,
ProverOptions → {options for the Prover}, **showBy → SomeDisplayer**];

The user of *Theorema* can control both the work of SomeProver by setting the ProverOptions and the way the proof is presented by setting the showBy option. By default, *Theorema* will present the proof in a new *Mathematica* notebook as a linear proof text. By setting *showBy → FocusWindows* the Focus Windows display method will be invoked. (For a complete description of the options of the Prove command and other details about *Theorema* see [8], [6]).

As mentioned before, the Focus Windows method presents proofs in a step–wise manner. Each step of the proof will be shown to the user in two phases: the attention phase and the transformation phase with the corresponding Attention Window (the formulae inferred at the inspected proof step are not yet shown to the user) and the Transformation Window. Each of these windows has

• a "goal area" in which the current goals are shown,

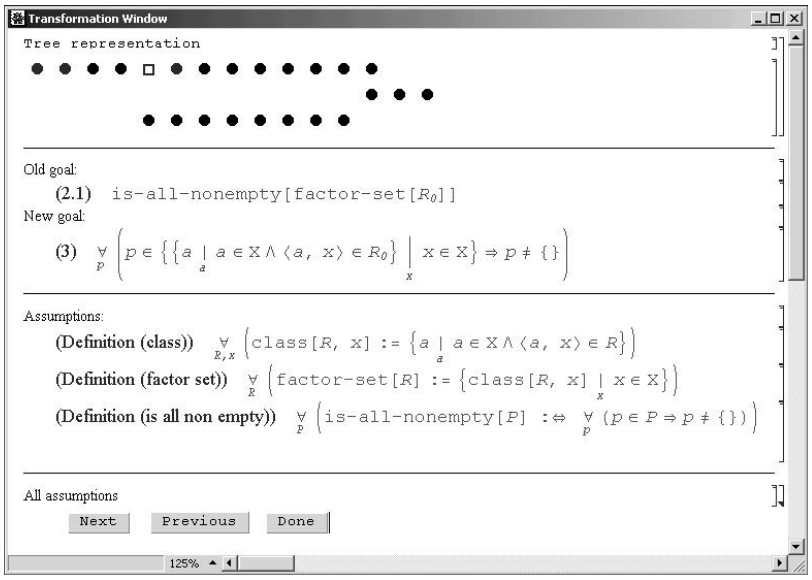
• an "assumptions area" in which the "relevant" assumptions are shown,

• a "proof tree area" in which the proof tree is displayed in a schematic form,

• an area that presents all the assumptions that are available (the "all assumptions area"),

• and a "navigation area" that allows the user to step forward or backward in the proof, in the order suggested by the prover that generated it.

As a concrete example, the goal area of the Transformation Window in the picture below contains the formulae (2.1) and (3). The latter is the formula that was inferred in the presented proof step, therefore its 'New Goal' heading. The assumptions area contains the definitions of the functions 'class' and 'factor–set' and the predicate 'is–all–nonempty'. If new assumptions would have been derived in this proof step, the corresponding formulae would have been shown under the heading 'New Assumptions'. The area containing all the assumptions that are currently available is shown in a closed cell, following the basic philosophy of the Focus Windows technique, that the user will normally *not* want to see all the assumptions that are available in the proof at that point. If the user is interested to see the contents of it (s)he has to double–click on the respective cell bracket. (The organization of notebooks using cells is a standard *Mathematica* feature, see [9]).



Old goal:

(2.1) is-all-nonempty[factor-set[R_0]]

New goal:

(3) $\underset{p}{\forall}\left(p \in \left\{\left\{\underset{a}{a \mid} a \in X \wedge \langle a, x\rangle \in R_0\right\} \underset{x}{\Big|} x \in X\right\} \Rightarrow p \neq \{\}\right)$

Assumptions:

(Definition (class)) $\underset{R,x}{\forall}\left(\text{class}[R, x] := \left\{\underset{a}{a \mid} a \in X \wedge \langle a, x\rangle \in R\right\}\right)$

(Definition (factor set)) $\underset{R}{\forall}\left(\text{factor-set}[R] := \left\{\underset{x}{\text{class}[R, x] \mid} x \in X\right\}\right)$

(Definition (is all non empty)) $\underset{P}{\forall}\left(\text{is-all-nonempty}[P] :\Leftrightarrow \underset{p}{\forall}(p \in P \Rightarrow p \neq \{\})\right)$

All assumptions

The simplified proof representation in the proof tree area, at the top of the above focus window, is not only a graphical representation but it also has some functionality. The nodes of the simplified tree representation are in one–to–one correspondence with the proof steps of the proof object, the current one being high–lighted (□). Clicking any of these nodes will cause the window to shift its focus to the proof step linked to the clicked node. Thus, the user is allowed to read the proof in the order (s)he prefers.

3 Implementation issues

The proof presentation technique explained in the section above should not be difficult to implement in any existing automated prover, even for systems that do not actually generate proofs automatically but restrict automation to checking proofs generated by humans (like HOL [3], Mizar [4]). The main pre–requisite is that the results of the provers in the system must be formal proof objects that contain sufficient information for extracting the used and inferred formulae, in any particular step.

We implemented the Focus Windows method in *Mathematica* [9], which is also the language we chose for the implementation of *Theorema*. In fact, the implementation was straightway because of two reasons:

• From the outset, the data structure of *Theorema* proof objects was carefully designed in order to give easy access to the relevant formulae in each proof step.

• The front end of *Mathematica* provides convenient programming tools for active objects that, basically, allow to apply the usual *Mathematica* programming style also for programming man–machine interfaces. We use this facility for attaching certain information to the buttons of the navigation area and of the schematic proof tree representation, reducing drastically the time needed for searching information in the proof tree. We give some more details below.

The user actions are taken in via the buttons 'Next', 'Previous' and 'Done' in the navigation area and the schematic proof tree presentation whose nodes are, in fact, buttons. The schematic proof tree representation is a static object in the sense that the data attached to its node buttons does not change during the presentation of the proof by the focus window viewer. In contrast, the buttons 'Next' and 'Previous' are dynamic objects, whose information is used in the following way:

• Suppose that the focus window is presenting the Attention Window of some node n of the proof tree. Then the data attached to the 'Previous' button is a link to the parent node of n. The data attached to the 'Next' button is a link to the node n because when pressing it we want to bring up the Transformation Window of the same node n.

• Suppose that the focus window is presenting the Transformation Window of some node n of the proof tree. Note that such a window may have several branches. Then the data attached to the 'Previous' button in each of the branches is a link to the node n because when pressing it we want to bring up the Attention Window of the same node n. The data attached to the 'Next' button in each of the branches is a link to the corresponding child node of n.

4 Conclusions

The essence of the method we presented is that we show, in each proof step, exactly the formulae that are *relevant* for the particular proof step and we put these formulae into our *focus*.

In the context of automated theorem proving, when proofs are naturally available as processable data objects (the "proof objects") this focusing operation can be described by an algorithm and can be made available for the users of automated theorem proving systems.

Note that the Focus Windows tool is *not* a prove method! The Focus Windows technique does not assert that each of the proof steps should be "easily" verifiable but, rather, it just gives a method to keep track of the relevant information used in each proof step a particular prover generates.

When comparing the linear proof presentation and the focus windows proof presentation of one and the same proof one may make the following observations:

• In short proofs, the focus windows presentation may generate presentation overhead that will distract the reader rather than help him.

• In proofs that are more than one or two pages long, the focus windows presentation may increase the possibility of verifying proofs drastically.

• Linear presentations are helpful for obtaining a quick overview on the overall flow of the proof whereas the focus windows presentation may drastically increase the process of thoroughly understanding proofs.

• Most probably, browsing a proof in linear representation and, then, studying the details of the proof by focus windows presentation style is the most reasonable and efficient way of understanding proofs.

After having implemented the Focus Windows technique in *Theorema*, we also made another, interesting and unexpected, experience: The tool can of course be applied to wrong proofs. In particular it can be used to check the proofs generated by theorem provers that are under construction and not yet fully tested. Here we noticed that checking the proofs by the Focus Windows technique makes it much easier to detect errors in the provers. Thus, the Focus Window tool may also be a useful research instrument for people working in the design and implementation of automated theorem provers.

References

1. B.Buchberger. *Focus Windows Presentation: A New Approach to Presenting Mathematical Proofs (in Automated Theorem Proving Systems). Theorema* Technical Report, 2000–01–30, RISC, http://www.risc.uni–linz.ac.at/people/buchberg/downloads.html
2. B. Buchberger, C. Dupré, T. Jebelean, F. Kriftner, K. Nakagawa, D. Vasaru, W. Windsteiger. *The Theorema Project: A Progress Report.* In: Symbolic Computation and Automated Reasoning (Proceedings of CALCULEMUS 2000, Symposium on the Integration of Symbolic Computation and Mechanized Reasoning, August 6–7, 2000, St. Andrews, Scotland, M. Kerber and M. Kohlhase eds.), A.K. Peters, Natick, Massachusetts, pp. 98–113. ISBN 1–56881–145–4.
3. *The HOL System.* Developed at the University of Cambridge, directed by R. Milner. http://www.cl.cam.ac.uk/Research/HVG/HOL/.
4. *Mizar System.* Developed at the University of Warsaw, directed by A. Trybulec. http://mizar.uwb.edu.pl/system/.
5. *Omega System.* Developed at the University of Saarbrücken, directed by J. Siekmann. http://www.ags.uni–sb.de/ omega/intro.html.
6. D. Vasaru–Dupré, *Automated Theorem Proving by Integrating Proving, Solving and Computing.* RISC Institute, May 2000, RISC report 00–19. PhD Thesis.
7. F. Wiedijk, *The Fourteen Provers of the World.* 2001, http://www.cs.kun.nl/ freek/notes/index.html
8. W. Windsteiger, *A Set Theory Prover in Theorema: Implementation and Practical Applications*, RISC Institute, May 2001, RISC report 01–03. PhD Thesis.
9. S.Wolfram. *The Mathematica Book*, Wolfram Media and Cambridge University Press, 1996.