Towards a Browsable Record of HOL Proofs

Jim Grundy
Thomas Långbacka

Turku Centre for Computer Science

TUCS Technical Reports

No 7, May 1996

Turku Centre for Computer Science
Lemminkäisenkatu 14
FIN-20520 Turku
Finland

http://www.tucs.abo.fi

May 1996

ISBN 951-650-748-4
ISSN 1239-1891

Abstract

This paper presents the motivations for and current stage of development of a tool for recording HOL proofs in a browsable format. The proofs recorded are window inference proofs produced using the TkWinHOL interface to the HOL window Library. The ultimate aim of the work is to help resolve the conflicting requirements for formality and readability in proof.

1 Introduction

It is possible to discern two attitudes toward proofs in the computer science community. Some regard a proof as a formal mathematical object, while the others view proof as a social process — a proof is an argument that convinces its reader. These views have been largely incompatible. Complete and formal proofs, such as the ones recorded by Wai Wong's extension of the HOL system [10], are too long and detailed to be understood by, and therefore convince, a human reader. On the other hand, many researchers place more faith in proofs like these than in less formal, but more readable, proofs because of the number of mistakes found in such proofs in the past.

The eventual goal of the work described here is to develop a method of recording HOL proofs in a readable format. The idea is that such a record would constitute a convincing proof under both definitions by being both readable and machine checked. We have attempted to address the readability problems of large and detailed formal proofs by recording HOL proofs in a browsable format. This format hides unnecessary levels of detail in the proof from the reader, while allowing them to view as much of the proof as they feel is necessary to become convinced of its validity. Presently the recording process requires help from the user who produced the proof to insert notes and comments to assist those who later wish to understand it. However, we are working on providing more machine assistance for this process.

Another possible solution is offered by the MIZAR proof system [8]. MIZAR uses its automated reasoning ability to check proofs in steps that are similar in complexity to those required for a readable presentation. The output of a MIZAR proof is a transcript annotated with automatically generated explanations. John Harrison is currently investigating the implementation of a similar system for HOL.

The HOL system is capable of supporting various styles of proof, including forward proof, goal-directed proof, and window inference. Although goal-directed proof is the style of reasoning most commonly used with HOL, we decided to investigate recording and browsing window inference proofs. Our reasons for this were mostly practical: we wanted to record and browse program derivations, a task for which window inference is more convenient than goal-directed reasoning; we already had access to a tool for browsing window inference proofs, which reduced our task to recording proofs in a format suitable for input to this tool; and, we had already developed a graphical user interface to the HOL library that support window inference, so we merely needed to extend this interface to record the commands it sends to HOL. We also hope the decision to record window inference proofs will turn out to be a good one for aesthetic reasons.

2 Window Inference

The window inference style of reasoning was originally proposed by Robinson and Staples [6], and later generalised and implemented in the HOL window Library by Grundy [2, 3]. Window inference is a transformational style of reasoning, in which a proof begins with a term X, and proceeds by applying transformations that preserve some desired relationship R. The proof ends when X has been transformed into another term Y that has some desired property. The result is a theorem of the form $\vdash X \; R \, Y$.

Program refinement, for example, is an instance of transformational reasoning. Here a proof begins with a specification, and proceed by a series of refinements until an implementation is reached. Goal-directed proof can also be seen as a special case of transformational reasoning. In this case the proof starts from the goal, and proceeds by a series of transformations in which each step implies the last, until the goal has been transformed to true.

The distinguishing feature of window inference is that it allows users to transform an term by restricting attention to a subterm and transforming it. The remainder of the surrounding term is left unchanged. While transforming a subterm, it is possible to make assumptions based on its context. For example, suppose we wish to transform the term $A \wedge B$; this may be done by transforming the subterm A under the assumption that B is true. The assumption of B is valid because if B were false, the entire term would be false regardless of the value of A.

2.1 Notation and Terminology

Window inference is conducted with a stack of windows. Each window has a *focus*, F, which is the term to be transformed; a set of formulae, $\Gamma_1, \ldots \Gamma_n$, called the *assumptions*, which can be assumed in the context of the window; and a *relation*, R, which is the relation to be preserved between the focus and any term to which it is transformed. Such a window will be written as follows:

$$
\begin{array}{ll}
 & !\ \Gamma_1 \\
 & \vdots \\
 & !\ \Gamma_n \\
R \star & F
\end{array}
$$

Only four kinds of operations are permitted as part of a window inference proof. These operations are as follows:

1. A new proof is begun by creating a new window stack containing a single window with a focus and relation chosen by the user.

2. A transformation can be applied to the focus of the window on top of the stack, providing it preserves the relationship associated with the window. The transformation may be conditional upon the assumptions of the window.

3. A new window can be pushed onto the stack. The focus of the new window must be some subterm of the focus or an assumption of the previous window. The assumptions and relation of the new window are a function of the position of the new focus within the previous window.[1] This operation is called *opening* a window.

4. The top window of a stack can be removed, providing this would not leave the stack empty. The relationship established between the initial and final focus of that window is then used to transform the window revealed underneath. This operation is called *closing* a window.

The objective of a proof of this style is to transform the focus of the bottom window of the stack until it has some desired property. The result of this process is a theorem which states that the initial and final focuses of the proof are related.

2.2 Example

We now present a brief example to illustrate the window inference style of reasoning. The predicate $A \wedge (A \vee B \Rightarrow C \wedge A)$ can be neither proved, nor refuted; for this reason it falls outside the domain of goal-directed proof methods. However, using window inference we can simplify the predicate by transforming it to an equivalent one. We begin with a window that has the predicate as its focus and equivalence as the relation to be preserved.

$$= \star\ A \wedge \boxed{(A \vee B \Rightarrow C \wedge A)}$$

Let us begin the proof by concentrating our attention on the more complex of the two conjuncts by opening a window on it. We indicate our intention of opening a window on some region by enclosing it with corners, $\boxed{\text{like this}}$.

$$\begin{array}{l} !\ A \\ = \star\ A \vee B \Rightarrow C \wedge A \end{array}$$

[1]For a precise description of how the relation and assumptions of the new window are determined, the reader is referred to the original description of window inference in HOL [3].

In the context of the second conjunct, we can assume that the first conjunct, A, is true. We can use this assumption to rewrite all instances of A in the focus to true.

$$
\begin{array}{l}
\quad ! \; A \\
= \star \; \top \vee B \Rightarrow C \wedge \top
\end{array}
$$

By further rewriting with standard facts about boolean connectives, we can simplify the focus to C.

$$
\begin{array}{l}
\quad ! \; A \\
= \star \; C
\end{array}
$$

This is as far as we can go with this focus, so we close the window and return to the enclosing term.

$$
= \star \; A \wedge C
$$

Here the original second conjunct is now replaced by the simpler term C.

This completes our example, which proves that the original term is equivalent to the simpler form $A \wedge C$. This result could also be established using a goal-directed proof, but it requires transformational reasoning to support the discovery of the result as part of the proof.

3 A New Format for Window Inference

The format in which the window inference proof was presented in the previous section closely reflects the format used by the HOL window Library [2]. This is itself only a trivial variation on the format used in Robinson and Staples's original presentation of window inference [6]. While this format is quite convenient for interactive use when constructing a proof, it is less suited for giving a readable presentation of a finished proof.

This is in contrast with the very readable format of calculation proof popularised by Gries and Schneider [1], van Gasteren [9] and others; and supported in HOL by Prasetya's DERIVATION package [4]. However, while short calculational proofs can be easy to read, their lack of structure makes them inconvenient for large problems. Back and von Wright have, in as yet unpublished work, developed an extended notation for calculation proof that is well suited to presenting window inference reasoning. A window inference proof without subproofs can, in fact, be written as a calculational proof. A window inference proof that contains a subproof can be presented as a calculational proof with a second calculational proof nested inside it. The nesting is indicated by indentation, and — for added clarity — the beginning of a subproof is marked with with a dot, like this $\bullet$. In the extended notation of calculational proof, any assumptions local to a subproof are listed in square brackets at its beginning.

For example, here is the window inference proof from the previous section presented in this structured calculational format.

$$
\begin{array}{ll}
 & A \wedge \boxed{(A \vee B \Rightarrow C \wedge A)} \\
= & \{\text{By simplifying the right conjunct.}\} \\
 & \quad \begin{array}{ll}
 \bullet & \langle A \rangle \\
 & A \vee B \Rightarrow C \wedge A \\
 = & \{\text{From the assumption that } A \text{ is true.}\} \\
 & \top \vee B \Rightarrow C \wedge \top \\
 = & \{\text{By boolean algebra.}\} \\
 & C
 \end{array} \\
 & A \wedge C
\end{array}
$$

The notation allows us to combine the appealing presentation of calculational proofs with the ability to decompose a derivation that characterises window inference.

4 ProofViews: A Browser for Proofs

The structured proof format described in the previous section admits the possibility of structured browsing to increase readability. A large proof containing many subproofs can appear daunting to a reader. Rather than presenting the whole proof at once, we can initially present the proof with all its subproofs hidden. If the reader is interested in a subproof, they can select the comment that describes it. The first layer of that subproof will then be revealed. In this way the reader can see not only the individual steps of a proof, but also the structure of the proof as a whole. Furthermore, the reader need only reveal as much of the proof as they feel necessary to be convinced of its validity.

The example derivation in the previous section is not nearly large enough to require browsing; nevertheless, we can use it to illustrate the concept. Figure 1 presents two views of this proofs. A proof browser would initially present a reader with the left-hand view. If the reader were interested in seeing more details, they could select the underlined comment. The browser would then reveal the hidden subproof, thus showing the reader the right-hand view.

The ProofViews package[2] consists of support for some extra macros for HTML, that can be used for formatting structured proofs. Figure 2 shows a ProofViews document that describes the proof from the previous section. Processing such a document creates a family of standard HTML documents. These documents give all the views of the proofs contained in the original document, and contain hyperlinks to navigate between them. Navigating between the documents using an ordinary web browser, like Netscape,[3] allows the reader to browse the proof without any special software.

The two screen shots in Figure 3 are taken from Netscape, and show the documents produced by the mark-up in Figure 2. The reader begins with the collapsed view on the left, and by clicking on the hyper-linked comment, moves to the expanded view on the right. Clicking on the dot in the right-hand view returns the reader to view on the left.

We believe we have demonstrated the utility and readability of this notation by using it to solve a significant and unbiased set of problems, namely the 1995 Finnish High-School Matriculation General Mathematics Exam. The solutions to these problems are available on the web for interested readers to browse and assess.[4]

5 TkWinHOL: A Graphical Interface

TkWinHOL [5] is a graphical user interface to the HOL window Library. TkWinHOL is designed to simplify the work a user has to do while transforming an initial focus towards the desired goal. This is achieved mainly by minimizing the amount of typing the user has to do. For example, one of the main features of TkWinHOL is that it allows the user to point out subterms with the mouse. Thus, no explicit path information has to be typed, which can be tedious if operating on a large focus. Most of the work is done by selecting some part of the current window (an assumption or a subterm of the focus etc.) and choosing an operation from a button or a menu. Possible additional information is supplied through dialog boxes. The long term goal is to make it possible to use the system with very little knowledge of the HOL system. Another goal is to make it possible to generate proofs that can be understood other than by HOL experts.

When TkWinHOL is started three windows appear on the screen. First

[2] The ProofViews package is available at the following URL:
http://www.abo.fi/~jgrundy/proof_views/man.html.

[3] All that is required is a browser that supports tables and can control their alignment.

[4] The problem set is available at the following URL:
http://www.abo.fi/~jgrundy/schoolmath/schoolmath.html.

$A \wedge \boxed{(A \vee B \Rightarrow C \wedge A)}$

$=$ {By simplifying the right conjunct.}

$A \wedge C$

$A \wedge \boxed{(A \vee B \Rightarrow C \wedge A)}$

$=$ {By simplifying the right conjunct.}

- $\langle A \rangle$

  $A \vee B \Rightarrow C \wedge A$

  $=$ {From the assumption that A is true.}

  $\top \vee B \Rightarrow C \wedge \top$

  $=$ {By boolean algebra.}

  C

$A \wedge C$

Figure 1: Two Views of the Example Proof

```
<HTML>
  <HEAD><TITLE>A Document with a Browsable Proof</TITLE></HEAD>
  <BODY>
    The example derivation in the previous section is not nearly
    large enough to require browsing; nevertheless, we can use it
    to illustrate the concept:
    _begin(derivation)
      _focus(A /\ _open(A \/ B ==> C /\ A))
      _begin(win)(=,By simplifying the right conjunct.)
        _assum(A)
        _focus(A \/ B ==> C /\ A)
        _trans(=,From the assumption that A is true.)
        _focus(T \/ B ==> C /\ T)
        _trans(=,By boolean algebra.)
        _focus(C)
      _end(win)
      _focus(A /\ C)
    _end(derivation)
  </BODY>
</HTML>
```

Figure 2: ProofViews Source for the Example

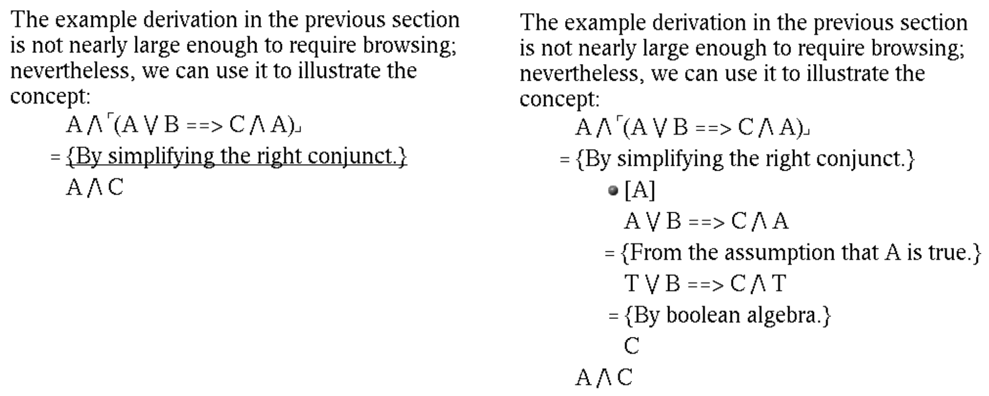

Figure 3: Screen Shots from the Browser

there is a simple text editor window which offers a small set of menu commands for file and editing operations. The text editor window also offers a subset of the ordinary Emacs key sequences for moving the insertion point etc. The second window, called the HOL window, is similar to a terminal window, like an Xterm. Commands can be typed directly into the HOL window and executed. Alternatively, users can execute any HOL command they want by selecting, using the mouse, a text string containing the command and pressing the execute button associated with the HOL window.

The third window is the heart of the application. This window, called the focus window, displays the status of the current window stack. The window is divided into three parts which display the focus and contextual assumptions of the window on top of the stack, and the theorem proved by the transformations applied to this window so far. Commands to transform the current window are bound to various menus and buttons.

In TkWinHOL, double-clicking the right mouse button on a position in the focus will select the smallest enclosing subterm of that position. After this, the user can choose the desired operation to perform; no path information is required. A typical operation is opening a window. The user selects, by clicking, the subterm to open a window on and then simply presses a button.

There is also a method for handling associative operators by means of selecting (by dragging, not clicking) a segment of the string representing the focus, and then opening a window on that selection even though it may not be accessible using a path in the ordinary sense. This is made possible by a routine that automatically reassociates the focus according to what the user needs, and then opens a window on the subterm the user has selected.

Operations that use contextual assumptions are supported by TkWinHOL. Users can click on the assumption that they want to use, this term is then assumed and passed as an argument to any command chosen from a menu. This saves the user from the trouble of typing the term, and additional type information.

In the operations described above, TkWinHOL constructs HOL commands that use only information present in the focus window. Sometimes, however, the user has to supply additional parameters to some menu command chosen. Such parameters are passed via dialogue boxes or by selecting a string, for example in the text editor window.

6 Proof Management in TkWinHOL

Proofs constructed within TkWinHOL must be saved somehow. TkWinHOL contains a scripting mechanism that stores the HOL proofs in a script. The same mechanism can, with small additions, be used to generate a script that the ProofViews tool can turn into a browsable proof.

6.1 Proof Scripts

The commands issued via the focus window while transforming a given stack are recorded as a script. This is a simple process which records every command sent to HOL, separating them with semicolons. The script can be saved and replayed on a later occasion to recreate the proof.

The recording of proofs is straightforward. Once a command is issued, it passes through a number of procedures that actually carry out the command. After a command has been accepted and executed by HOL and the output of it presented in the appropriate window, the command is passed on to a procedure that performs proof recording.

This processing consists mostly of storing commands as they are. Only a few kinds of commands need to be singled out for special treatment.

- When a new stack is created a new script has to be started.
- When a proof is finished the user has to be prompted for whether the script should be saved or not.
- When the user uses the Undo/Redo commands available in the HOL window Library the script has to be updated accordingly

- Every stack (there can be several at the same time) has its own script, so a command to change from one stack to another has to be handled properly.

6.2 Recording Browsable Proofs

In order to create browsable proofs we need to generate scripts that have a format suitable for the ProofViews tool described earlier. It is mostly the same commands that are of interest when generating this kind of a script as well. Thus, extending the interface to handle such scripts is relatively easy, we needed to add only a few more output actions to the already existing scripting infrastructure. The output is then fed to the ProofViews tool to generate the actual browsable proof.

However, there are a couple of new (as compared to above) operations that need special attention:

- When a new window is opened the specific commands to indicate the start of a subderivation have to be added to the script.

- When a window is closed the subderivation has to be ended properly.

In the HOL window Library there are also commands that allow the user to create useful lemmas in the middle of an ongoing derivation. These can be created by transforming one of the assumptions, or by proving a theorem. In ProofViews there is no built in support for managing lemmas. Our solution is to add the proofs of lemmas at the end of the script as separate derivations.

The above mentioned operations are the only ones which deal explicitly with the different hierarchies of a proof. This has the great advantage that the recording process works well no matter which specialised theories the user of the system might work in.

7 An Example Browsable HOL Proof

To further illustrate the browsable proofs constructed with TkWinHOL we will look at how the term

$$(A \wedge ((A \vee B) \Rightarrow (C \wedge A))) \wedge (((Y \vee Z) \Rightarrow (X \vee Z)) \Rightarrow (X \Rightarrow Y)) \quad (1)$$

can be transformed to the much simpler term

$$(A \wedge C) \wedge (X \Rightarrow Y) \quad (2)$$

while preserving equality, and how this is represented as a browsable proof. While performing the transformations necessary to establish the above fact, a ProofViews script is generated. This script can subsequently be run through the ProofViews tool. The resulting browsable proof has the format shown in Figure 4 at it's highest level of abstraction. Similarly, Figure 6 shows the proof at lowest level of abstraction while Figure 5 shows an intermediate level.

The comments of each step in Figures 4 and 5 (e.g. Open window, Transformation) are not very informative. They are generated on the fly by TkWinHOL. If needed, the user can easily change them manually make them more descriptive. An example of this can be seen in Figure 6 where manual changes have been made to the ProofViews input that is turned into a browsable proof. We are looking at ways to make it possible to enter more instructive comments interactively, and also on how to improve the automatically generated comments.

The proof strategy for transforming (1) into (2) is straightforward. We begin by opening a window on the left conjunct of (1), creating the following focus:

$$A \wedge ((A \vee B) \Rightarrow (C \wedge A))$$

where we open a window on the right conjunct. At this point one of the assumptions is A, which can be used to rewrite the focus $(A \vee B) \Rightarrow (C \wedge A)$ to C. After closing the two windows we opened, the original focus has been reduced to

$$(A \wedge C) \wedge (((Y \vee Z) \Rightarrow (X \vee Z)) \Rightarrow (X \Rightarrow Y)) \tag{3}$$

which corresponds to the first step in the browsable proof shown in Figure 4. Figure 5 shows a view where this derivation is detailed.

At this point we concentrate on the right conjunct of (3). After opening a window on that subterm we continue by opening a window on the antecedent arriving at the following focus.

$$(Y \vee Z) \Rightarrow (X \vee Z) \tag{4}$$

Now one of the assumptions is X, which can be used to reduce (4) to true (written as $\top$). When closing that window the focus becomes $\top \Rightarrow (X \Rightarrow Y)$ which can be rewritten, using only built-in rewrite rules, to $X \Rightarrow Y$. After closing one more time our focus becomes (2) and we are done.[5] In Figure 4

[5]It is worth noting that no typing is required for the entire proof, except for the creation of the original stack. The proof process progresses through selecting information on the screen and choosing commands that use the information – in accordance to what was described in section 5.

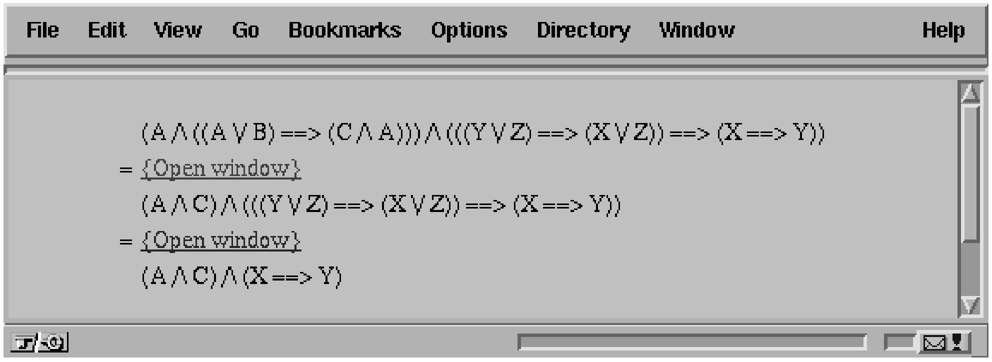

Figure 4: Top level of browsable proof

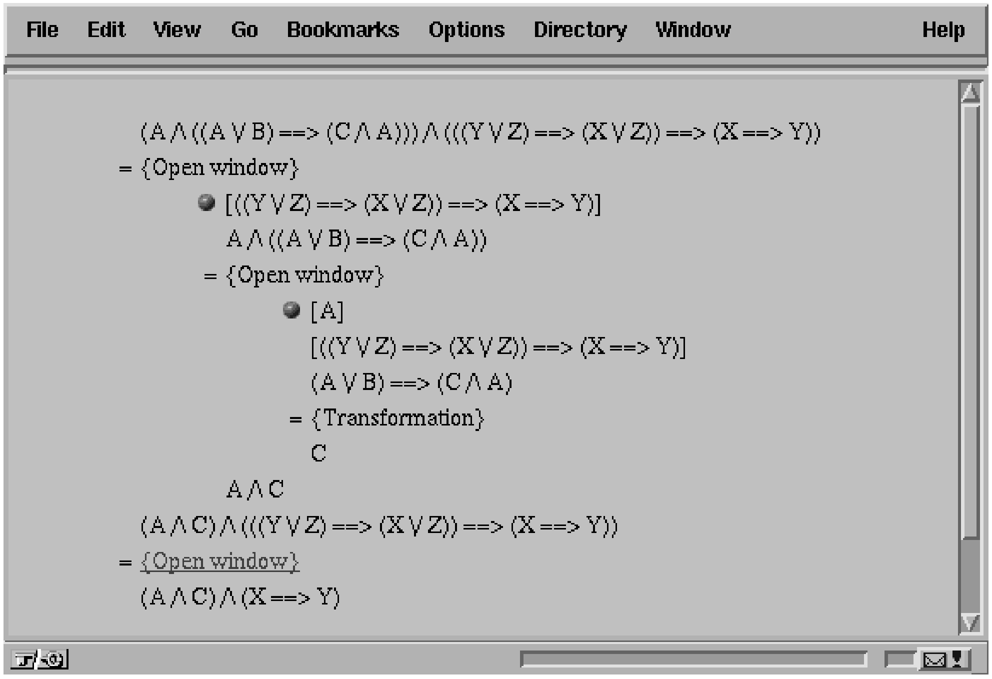

Figure 5: Transformations of left conjunct in browsable proof

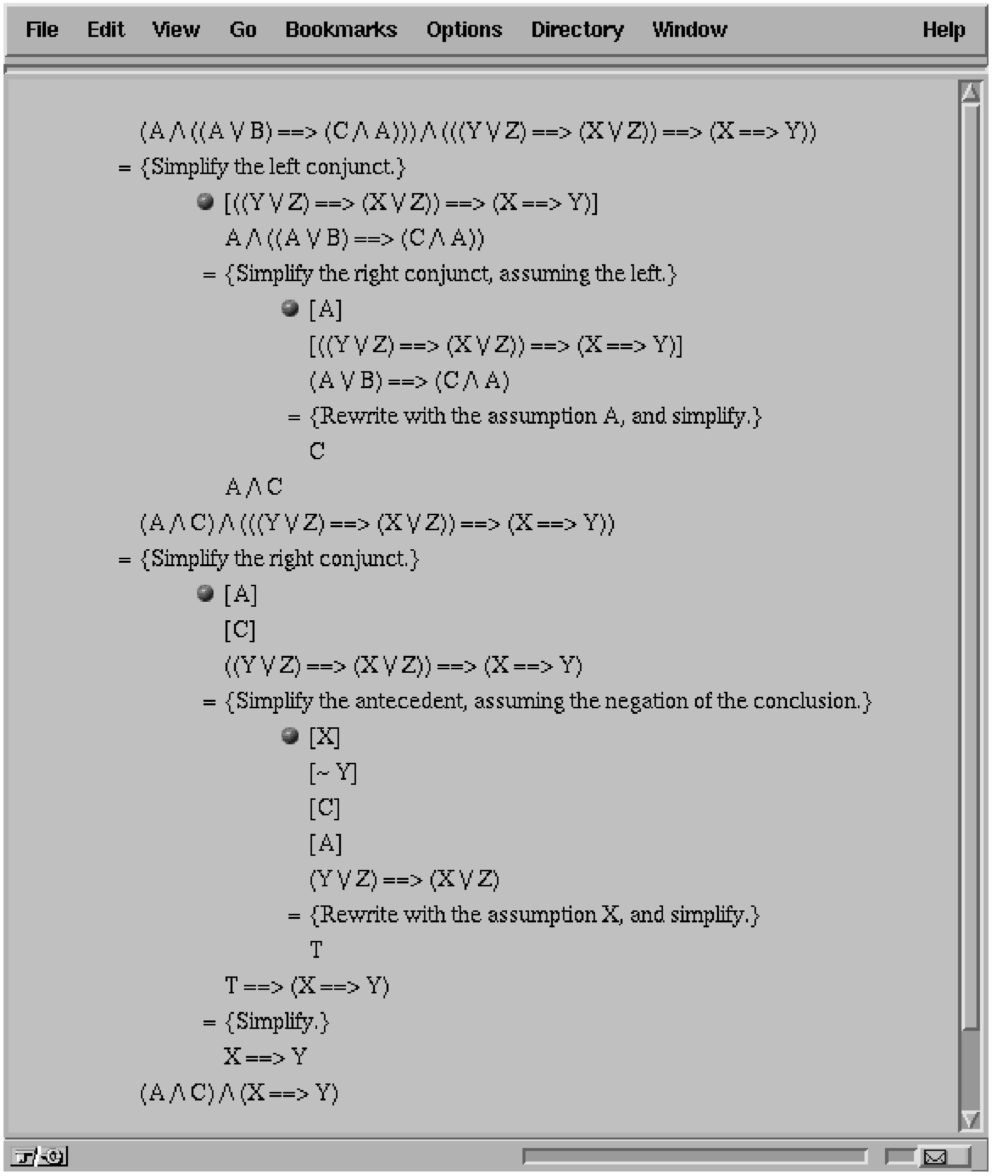

Figure 6: Browsable proof showing all details

the transformation of the right conjunct from (3) to (2) is shown as just one step. The detailed derivation is shown in Figure 6.

8 Summary and Future Work

In this paper we have considered the possibility of resolving the conflict between the differing requirements of formality and readability in proof by recording HOL proofs in a browsable format. This has included the description of a new, structured notation for calculational proof due to Back and von Wright, and an example of how we can use this notation to give a more readable presentation of window inference. We then gave a brief overview of a new tool, called ProofViews, for producing browsable versions of such proofs which can be viewed with an ordinary web browser. Finally, we described how we extended the TkWinHOL interface to the HOL window Library to record its proofs in a format suitable for input to the ProofViews tool, thereby generating a browsable record of a HOL proof.

Much work remains to be done, and there are a number of ways in which the current version of the proof recording tool could be improved:

- The default comments added to the proof record are not very useful. We are working on ways to improve this, e.g. by adding command specific comments for rewriting and simplification, etc. We are also considering adding hyperlinks to the HOL documentation for the commands and theorems used in a proof.

- Any lemmas created during the proof process are currently listed at the end of the record as separate derivations. In later versions of the tool we hope to make hyperlinks between these side proofs and the places where they are used.

- The list of local assumptions at the beginning of a subproof presently includes all those assumptions valid in that context. Ideally, only those assumptions actually used in the proof should be listed.

- At present the system does not record commands other than those issued via the focus window of the TkWinHOL tool. Future versions of the tool must be capable of recording all commands issued to the theorem provers.

Despite these present deficiencies, we feel that the approach of recording HOL proofs in a browsable format will prove to be a useful one. Our experience with the ProofViews package has shown it to be a valuable tool for increasing the readability of informal proofs. We hope that further experience with the tool described in this paper will also demonstrate the value of browsing for increasing the readability of formal proofs.

References

[1] D. Gries and F. B. Schneider. *A Logical Approach to Discrete Math.* Texts and Monographs in Computer Science. Springer-Verlag, New York, 1993.

[2] J. Grundy. The HOL window library. In *The HOL System*, volume Libraries. SRI International, Cambridge Research Center, Millers Yard, Mill Lane, Cambridge CB2 1RQ, England, 2.01 edition, July 1991.

[3] J. Grundy. Window inference in the HOL system. In M. Archer, J. J. Joyce, K. N. Levitt, and P. J. Windley, editors, *The HOL Theorem Proving System and its Applications: Proceedings of the 1991 International Workshop*, pages 177–189, University of California at Davis, Aug. 1991. IEEE Computer Society Press.

[4] ISWB Prasetya. On the style of mechanical proving. In J. J. Joyce and C.-J. H. Seger, editors, *Higher Order Logic Theorem Proving and its Applications: Proceedings of the 6th International Workshop, HUG'93*, volume 780 of *Lecture Notes in Computer Science*, pages 475–488, Vancouver, Canada, Aug. 1993. Springer-Verlag.

[5] T. Långbacka, R. Rukšėnas, and J. von Wright. TkWinHOL: A tool for doing window inference in HOL. In Schubert et al. [7], pages 245–260.

[6] P. J. Robinson and J. Staples. Formalizing a hierarchical structure of practical mathematical reasoning. *Journal of Logic and Computation*, 3(1):47–61, Feb. 1993.

[7] E. T. Schubert, P. J. Windley, and J. Alves-Foss, editors. *Higher Order Logic Theorem Proving and Its Applications: Proceedings of the 8th International Workshop*, volume 971 of *Lecture Notes in Computer Science*, Aspen Grove, Utah, Sept. 1995. Springer-Verlag.

[8] A. Trybulec and H. A. Blair. Computer aided reasoning. In R. Parikh, editor, *Logic of Programs: Proceedings*, volume 193 of *Lecture Notes in Computer Science*, pages 406–412, Brooklyn, New York, June 1985. Springer-Verlag.

[9] A. J. M. van Gasteren. *On The Shape of Mathematical Arguments*, volume 445 of *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 1990.

[10] W. Wong. Recording and checking HOL proofs. In Schubert et al. [7], pages 353–368.

Turku Centre for Computer Science
Lemminkäisenkatu 14
FIN-20520 Turku
Finland

http://www.tucs.abo.fi

University of Turku
- **Department of Mathematical Sciences**

Åbo Akademi University
- **Department of Computer Science**
- **Institute for Advanced Management Systems Research**

Turku School of Economics and Business Administration
- **Institute of Information Systems Science**