# An Overview of the WinKE System

Ulrich Endriss

April 15, 1998

**Abstract:** WinKE is a new interactive theorem proving assistant based on the logical refutation system KE. It has been developed to support teaching logic and reasoning to students of philosophy, computer science, or other disciplines involving logic. The software is supportive of an introductory textbook on classical logic. Constructing a proof tree using WinKE is as close as possible to the common pen-and-paper procedure and the system additionally offers a wide variety of useful features.

This paper starts with a brief introduction to the KE calculus. Next a step-by-step tutorial to the basic usage of WinKE (including its installation) and an overview of the main features of the software are presented. Finally, some aspects of design are addressed.

## Contents

# 1 Preface

In this paper WinKE, a new interactive theorem proving assistant based on the KE calculus (see [D'Agostino and Mondadori, 1994]) is presented. It has been developed to support teaching logic and reasoning to students of philosophy, computer science, or other subjects involving logic. Using WinKE you can work on problems in either propositional or first order classical logic. Running under Windows 95 it is easy to learn and visually satisfying. Important features include a comfortable graphical user interface, on- and off-line proof checking, automated deduction, hints, bookkeeping facilities, editing problem files, and many more. The software is intended to support an introductory textbook on classical logic [Mondadori and D'Agostino, 1997], but may also be used independently.

The WinKE-project started as a student project for a Master's thesis [Endriss, 1996] at Imperial College, London. It has been strongly inspired by the work described in [Pitt, 1995]. The further development of the software is currently funded by the University of Ferrara. It has been implemented entirerly in PROLOG (see [Bratko, 1990] or others) using LPA's programming environment [LPA, 1995].

**Paper Outline:** Section 2 provides a short introduction to the logical refutation system KE. Starting with a description of how to install WinKE on your computer Section 3 illustrates the very basic usage of the program. A more comprehensive overview of WinKE's functionality is presented in Section 4. Finally, Section 5 addresses some aspects of the design of the tool's interface, its internal architecture, and the prove procedure deployed.

# 2 The KE Calculus

The KE calculus is a logical refutation system for proving theorems. This section provides a brief introduction to KE for first order classical logic. Generally speaking, one way to prove that a formula $C$ (the so-called conclusion) logically folllows from a set of formulas $\{P_1, \ldots, P_n\}$ (the so-called premises) is to show that the set $\{P_1, \ldots, P_n, \neg C\}$, i.e. the set of premises together with the negated conclusion, forms an inconsistent set of formlas. Inconsistency can be shown by expanding the formulas into simpler subformulas until a contradiction is found. This expansion is represented as a tree where nodes are labeled with formulas. The premises and the negated conclusion form the root of such a proof tree. Then the formulas are expanded

according to the KE rules given in Table 1. Throughout a proof any particular branch represents the conjunction of the formulas appearing on it. The tree itself represents the disjunction of the conjunctions associated with its branches.

**Alpha Rules (with $\neg\neg$-Elimination)**

$$\frac{A \wedge B}{\begin{array}{l}A\\B\end{array}} \qquad \frac{\neg(A \vee B)}{\begin{array}{l}\neg A\\\neg B\end{array}} \qquad \frac{\neg(A \rightarrow B)}{\begin{array}{l}A\\\neg B\end{array}} \qquad \frac{\neg\neg A}{A}$$

**Beta Rules**

$$\frac{\begin{array}{l}A \vee B\\\neg A\end{array}}{B} \qquad \frac{\neg(A \wedge B)}{\begin{array}{l}A\\\neg B\end{array}} \qquad \frac{A \rightarrow B}{\begin{array}{l}A\\B\end{array}} \qquad \frac{\begin{array}{l}A \rightarrow B\\\neg B\end{array}}{\neg A}$$

**Eta Rules**

$$\frac{\begin{array}{l}A \leftrightarrow B\\A\end{array}}{B} \qquad \frac{\begin{array}{l}A \leftrightarrow B\\\neg A\end{array}}{\neg B} \qquad \frac{\neg(A \leftrightarrow B)}{\begin{array}{l}A\\\neg B\end{array}} \qquad \frac{\neg(A \leftrightarrow B)}{\begin{array}{l}\neg A\\B\end{array}}$$

**Gamma Rules**

$$\frac{\forall x : P}{P[x/t]} \qquad \frac{\neg\exists x : P}{\neg P[x/t]} \qquad \text{for any closed term } t$$

**Delta Rules**

$$\frac{\exists x : P}{P[x/p]} \qquad \frac{\neg\forall x : P}{\neg P[x/p]} \qquad \text{for a new parameter } p$$

**Principle of Bivalence (*PB*)**

$$\frac{}{A \mid \neg A}$$

Table 1: The KE Rules for First Order Logic

The KE rules are to be read as follows: whenever a formula (or two formulas) matching the schema above the line is (are) found on a branch, the formula(s) below the line can be added to that branch. The formula(s) above the line is (are) usually called the premise(s) of that rule, the one(s) below the line are called its conclusion(s). If there are two premises to a rule the first one is usually referred to as the major premise and the second

one as the minor premise. Conjunctive formulas are expanded by applying alpha rules, disjunctive formulas by beta rules, and formulas involving the $\leftrightarrow$ operator are analysed by eta rules. Universally quantified formulas are analysed using gamma rules and existantially quantified ones using delta rules. Note that $P[x/t]$ denotes the formula obtained by substituting all occurences of $x$ (as a free variable) in $P$ with $t$. Finally, as any formula $A$ is either true or false we can always split a branch and add $A$ to the left branch and $\neg A$ the right one. This rule is called $PB$ (for principle of bivalence). Nevertheless, in practice the application of $PB$ should be restricted to subformulas of formulas of either the beta or the eta type which cannot be analysed at the moment. This is called *analytic* application of $PB$, see [D'Agostino and Mondadori, 1994]. A branch is said to be closed, if there is a pair of complementary formulas on it. If all branches of a tree are closed, the formulas on its root form an inconsistent set, i.e. the associated theorem is proven to hold. Proofs for soundness and completeness of KE are given in [Mondadori and D'Agostino, 1997] and [D'Agostino and Mondadori, 1994].

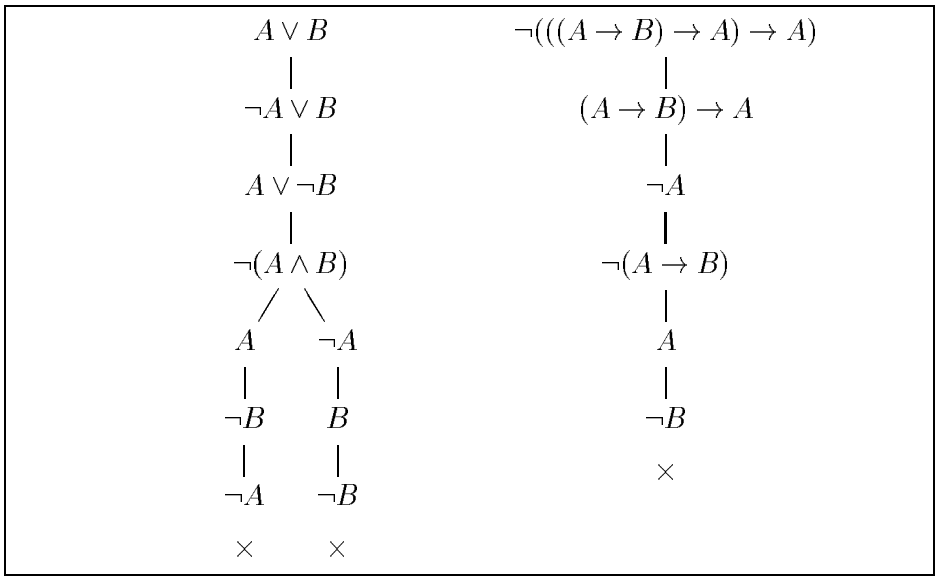


Table 2: Two Example KE Refutations

To illustrate what has just been said we give two very simple examples. The first tree in Table 2 shows that $A \wedge B$ logically follows from the three premises $A \vee B$, $\neg A \vee B$, and $A \vee \neg B$. At the beginning no rule but $PB$

can be applied. The branch is split in two writing $A$ on the left-hand side and $\neg A$ on the right-hand side. Next on the left branch applying beta to $\neg(A \wedge B)$ and $A$ yields $\neg B$. Applying beta to $\neg A \vee B$ and $\neg B$ again yields $\neg A$ on the same branch. As $\neg A$ is the complement of $A$ that branch can be closed. Analysing and closing the right branch works similarly. The other theorem to be shown is $((A \rightarrow B) \rightarrow A) \rightarrow A)$ (withot any premises). The first step is an application of an alpha rule. Then beta can be applied to $(A \rightarrow B) \rightarrow A$ and $\neg A$ to get $\neg(A \rightarrow B)$, which again can be analysed using an alpha rule. The tree is closed with $\neg A$ and $A$.

The KE calculus is close to the well known method of semantic tableaux (see [Fitting, 1990]). The alpha rules and the quantifier rules of both systems are exactly the same. The cruicial difference is, that the tableaux system is cut-free, which means, it does not have a rule that correponds to *PB*. In [D'Agostino and Mondadori, 1994] it has been shown that this lack of a rule expressing the bivalent character of classical logic makes tableaux particularly worse than KE as far as the complexity of proofs is concerned. In [D'Agostino *et al.*, 1997] and [Broda *et al.*, 1995] it has also been argued that KE seems to be better suited for the teaching of basic classical logic than both tableaux and natural deduction (see [Fitting, 1990]).

# 3 First Steps Using WinKE

## 3.1 Installation

To install WinKE on your PC (running Windows 95) the following three steps have to be performed:

1. Install the WinKE-font by copying `WinKE.ttf` to `C:\Windows\Fonts`.
2. Copy the folder named `WinKE` and its entire contents to `C:\`.
3. Create a shortcut to `C:\WinKE\WinKE.exe` using the right mouse button. Enter `C:\WinKE\WinKE.exe /V1 /T512 /H512` as command line.

**Note:** The value after the command line switch `/H` denotes the heap space available to WinKE. You might want to try to increase this value when you are working on rather large problems.

## 3.2 Interface and File System

After having started WinKE four windows will pop up on the screen (see Figure 1). The window captioned `WinKE` is called the *main window*. It contains

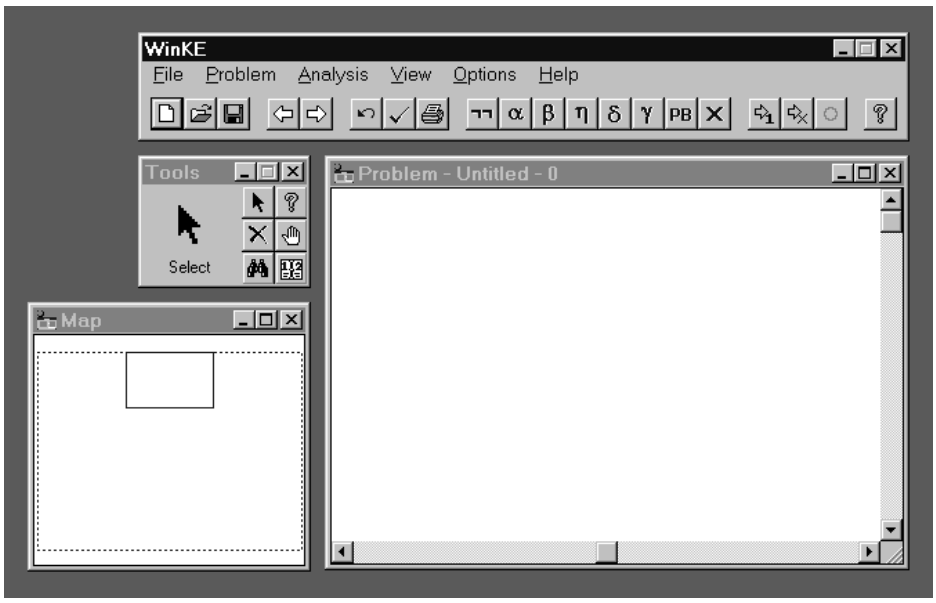


Figure 1: The WinKE Interface

all the menus which are used to initiate most user actions. The buttons on the main window provide shortcuts to menu options likely to be used frequently. The button on the very right will startup the WinKE Help system. The small window on the left captioned `Tools` is the *graphic tool box*. A particular graphic tool is chosen by clicking on it. The default graphic tool is the select tool. Pressing the help button (the questionmark) will directly enter WinKE Help on the page related to graphic tools. Proof trees will be displayed in the large window, which is called the *graphic window*. Finally, the small rectangle in the window captioned `Map` corresponds to the drawing area visible in the graphic window. Using the mouse it can be moved around when working on large trees.

Via the File menu you can open a problem file. Problem files have the extension "`.ke`". A problem can be a set of formulas (i.e. normally some premises and a negated conclusion), an entire proof tree, or a partially finished proof. You can scroll through the problems of a file by selecting Next and Previous respectively from the Problem menu.

Proof trees are displayed in the obvious way. To save space on the screen there are no lines connecting consecutive nodes on a non-splitting branch.

Every branch is associated with a so-called *branch marker*. A little circle denotes an open branch, a cross denotes a closed one. Using the **select tool** you can select or deselect formulas and branch markers. Selected objects will be highlighted in red.

### 3.3 An Example System Session

This section describes the very basic usage of WinKE. It will be shown how the program can be used to prove $((A \to B) \to A) \to A$ (see also Table 2). WinKE provides three different levels of supervision and assistance respectively, the teaching modes. They are called **supervisor**, **pedagogue**, and **assistant**. The teaching mode can be changed via the **Options** menu at any time. Using the **pedagogue mode** is recommended for beginners. In that mode every proof step you perform is checked on-line. In **supervisor mode** on the other hand anything goes, but you can initiate an off-line proof checking whenever you want. Using the **assistant mode** minimizes the user input and therefor is very comfortable for expert users but not advisable to beginners.

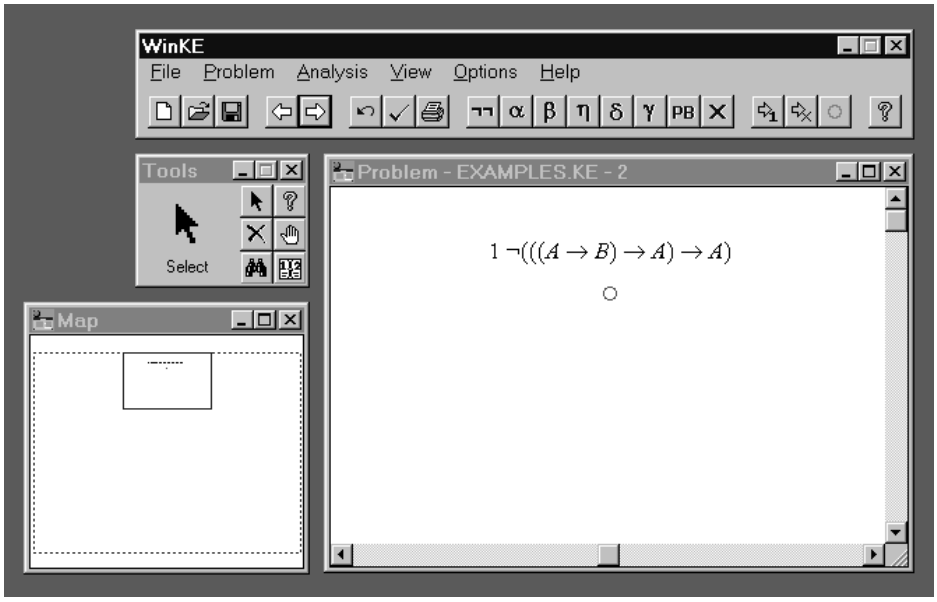


Figure 2: WinKE after having opened a Problem File

Now you should open a file and select a problem in it. Figure 2 shows WinKE after the file `Examples.ke` from the `WinKE\Examples`-folder has been

opened and the second problem in it has been selected. You may choose Reset from the Problem menu to delete any proof steps that might have been performed earlier. The problem consists of just one formula, the negated theorem to be proven. It has to be analysed using an alpha rule. This is done by selecting the formula you want to analyse and the branch it is on (i.e. its branch marker) using the **select tool**. Make sure it is activated by clicking on the arrow-symbol in the graphic tool box and then click on the objects in the graphic window. After that a KE rule, in this case alpha, can be chosen via the Analysis menu or alternatively by pressing the respective button on the main window. This will cause a so-called *rule application dialogue* to pop up on your screen. This situation is shown in Figure 3. Type in the conclusion formulas (which are $(A \to B) \to A$ and $\neg A$) and press OK. For typing the logic operators you can use the virtual keypoard within the dialogue. To find out how to type them in directly, start WinKE Help and search for "typing logic symbols". Another possibility would be to cut and paste from the displayed premise formula. If your input has been correct, you will see the two formulas added to the tree in the graphic window.

The next proof step is to apply beta to formulas number 2 and 3. Select them, make sure the branch marker is still selected, and then press the $\beta$ button. Again a rule application dialogue will pop up and you are asked to fill in the rule's conclusion, which in this case is $\neg(A \to B)$. Press OK to see the resulting proof tree. Now applying the alpha rule to the last formula $\neg(A \to B)$ as described before will add $A$ and $\neg B$ to the (only branch of this) tree. It can now be closed, as there are two contradictive formulas on it. Select $A$ and $\neg A$ and either press the cross-button on th main window or choose Close Branch from the Analysis menu. This will cause the open branch marker to turn into a closed branch marker, i.e. a cross. As all branches (there's just one) of the tree are closed, this is a valid proof for the theorem whose negation labels the tree's root.

Figure 4 shows the entire proof tree for $((A \to B) \to A) \to A$. It also illustrates a useful feature, the **bookkeeping tool**. You can activate it by clicking on the book-symbol in the graphic tool box. Then clicking on a graphic object (i.e. a formula or a branch) in the large window will cause a message containig the *bookkeeping information* associated with that object to be displayed. It tells you, what KE rule and what other formula(s) have been used to derive a formula. For formulas derived by applying an alpha rule or *PB* also its sibling formula will be displayed. You will also be informed, whether the clicked formula itself has already been analysed on all branches it is on.

In case the system hasn't been in pedagogue mode throughout the entire

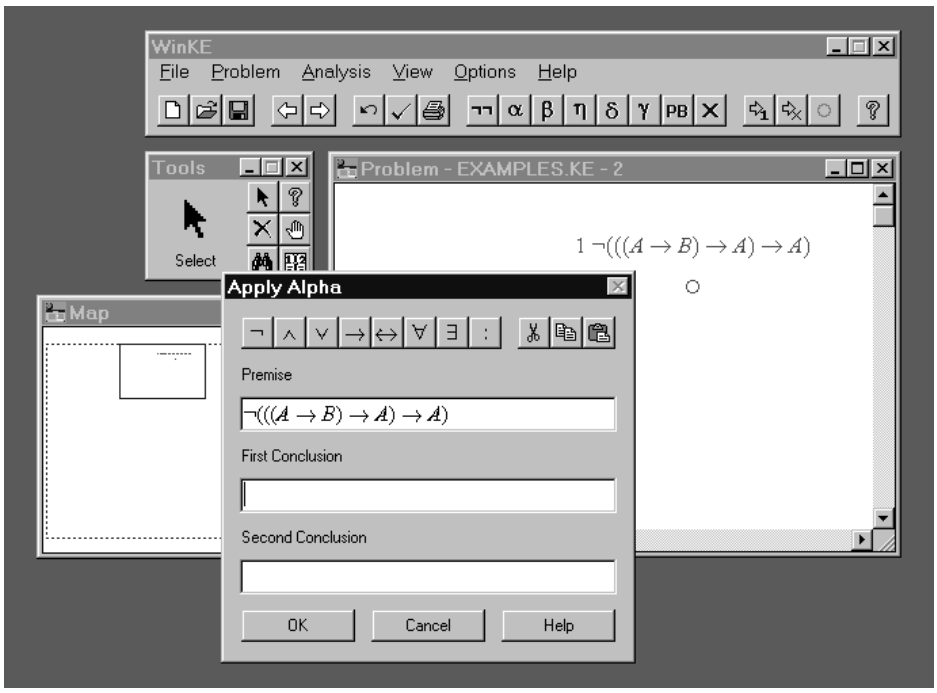


Figure 3: Applying the Alpha Rule

duration of the proving process you may want to check the proof's correctness again. To do so select Check from the Problem menu or press the button showing a little hook. For this example you should simply get a message telling you that all proof steps are correct. In case you made a mistake a dialogue displaying a description of the error will pop up. You have the possibility to retract the wrong formula(s) directly from within this dialogue.

At any stage of a proof you can activate either the **delete tool** (cross-symbol) or the **retract tool** (hand-symbol) in the graphic tool box. Clicking on a formula using the former the formula and the entire branch below it will be deleted. If the clicked formula has got a sibling formula, it will be deleted as well. The difference between the **delete tool** and the **retract tool** is, that the latter only deletes the clicked formula and all formulas which logically depend on it, i.e. which could not have been derived without the clicked formula beeing on the branch. The simplest way to take back the last (wrong) proof step is to choose Undo from the Problem menu.

If you followed the instructions of this section, you now should have at

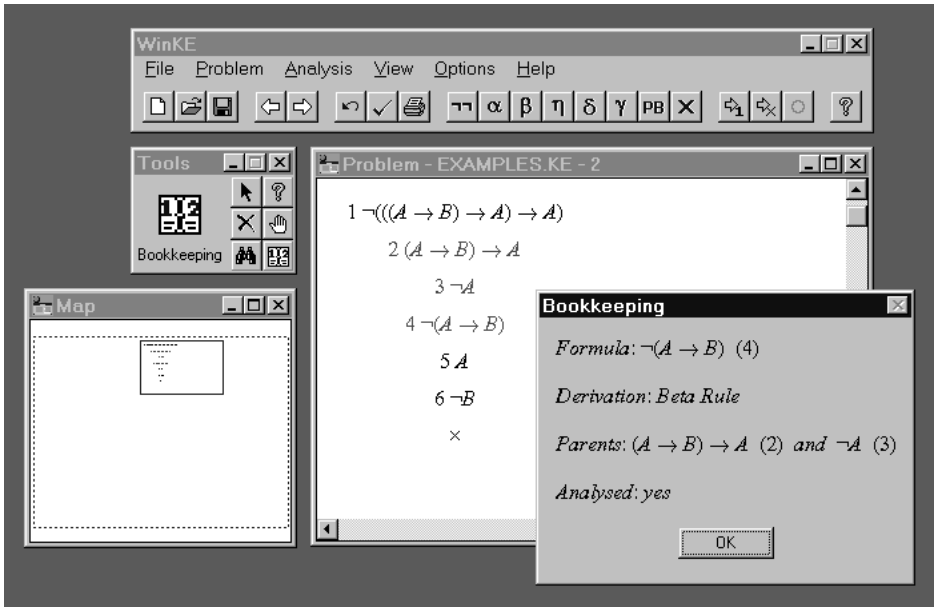


Figure 4: Using the Bookkeeping Tool

least a rough idea how to work with WinKE. The next section provides a systematical overview of the program's functionality.

# 4 Main Features of WinKE

## 4.1 Menus and Graphic Tools

All user action is either initiated via menus or one of the graphic tools. A particular graphic tool is selected by clicking on the associated symbol in the graphic tool box. Using a particular graphic tool means clicking on graphic objects (i.e. either formulas or branch markers in the graphic window) with that tool being activated. Clicking on the questionmark-symbol in the tool box window will enter the WinKE Help system at the page covering the graphic tools. There are five tools available:

- The select tool is the default graphic tool. It is associated with the arrow-symbol. The select tool is used to select the graphic objects you want to apply a KE rule to.

- The **delete tool** is associated with the cross-symbol. If you click on a formula that formula and the entire branch below it will be deleted. If the clicked formula has got a sibling formula, it and the branch below it will be deleted as well.

- The **retract tool**, which can be activated by clicking on the hand-symbol, is also used to delete (wrong) proof steps. The difference to the **delete tool** is, that in this case only the formula itself and those formulas that logically depend on it are deleted from the tree.

- The **hint tool** is associated with the binoculars-symbol. Clicking on a formula will cause all open branch markers associated with branches that formula has not yet been analysed on to be highlighted. The other way round, when using it on open branch markers, the formulas which not have been analysed on the respective branch are highlighted.

- The **bookkeeping tool**, which is associated with the book-symbol, is used to reveal the bookkeeping information available for a graphic object. For formulas this means, how they have been derived and whether they have been analysed on all open branches. For formulas derived by applying either an alpha rule or *PB* also its sibling formula is part of the bookkeeping information. For a closed branch marker the two formulas used for closure are given.

The WinKE main window contains six menus. Everything that has to do with file handling, i.e. opening and saving files, is done via the **File** menu. To quit WinKE you may also user the **File** menu. Actions that concern an entire problem or proof are invoked via the **Problem** menu. This includes jumping to other problems, editing problems and information directly related to them, checking proofs, printing proof trees and others. Via the **Analysis** menu the KE rules are applied and automated deduction can be started. The **View** menu is used to zoom in and out of the graphic window. The **Options** menu is used to determine a teaching mode and a couple of other settings (see Section 4.6). To start WinKE Help use the **Help** menu.

## 4.2 The WinkE Help System

The WinKE Help system is entered either via the **Contents** option of the **Help** menu or by pressing the button showing a questionmark on the main window. WinKE Help is a standard Windows 95 help file. That's why menu information etc. will appear in the language used in your Windows version. Information on all aspects of WinKE is provided. In particular every

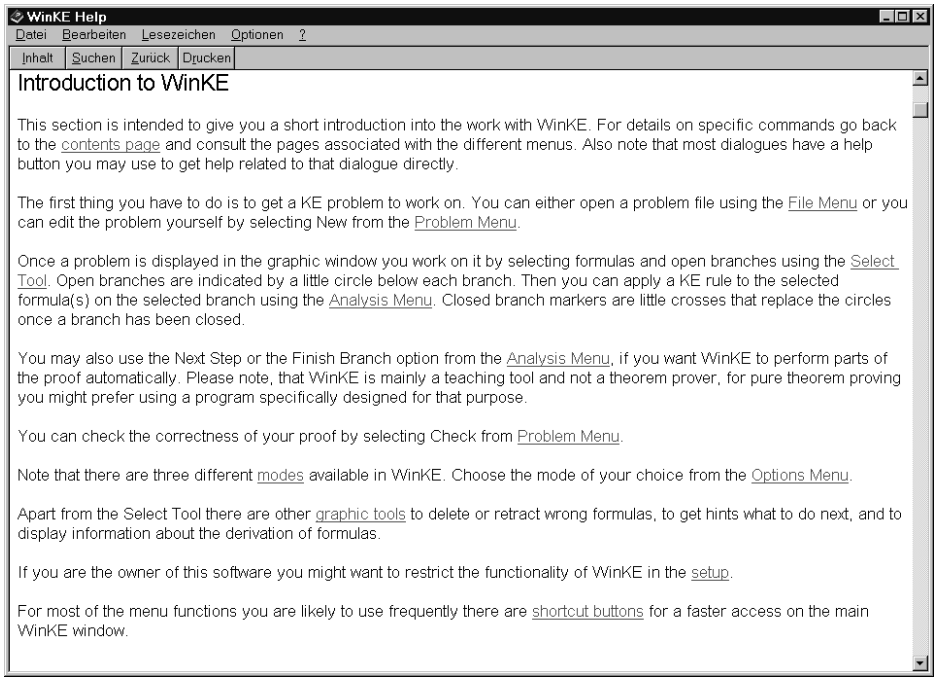


Figure 5: The WinKE Help System

(non-trivial) dialogue provides a Help button to obtain information on that particular dialogue. Via the contents page you can access pages for every single menu on the main window. In addition to that some general pages give a brief introduction to KE and WinKE (like for example the page shown in Figure 5).

### 4.3 Teaching Modes and Proof Checking

Via the Options menu a teaching mode can be selected. They are called supervisor, pedagogue, and assistant. In supervisor mode within the rule application dialogues any (syntactically correct) input is accepted, whereas in pedagogue mode the correctness of a rule application is checked on-line. The same is true for the assistant, but here the user input is reduced to a minimum. That means, for the simple rules (the propositional ones apart from $PB$), no input of the conclusion(s) is required as their derivation is straightforward given the premise(s). For the other rules the system gives a list of

possible inputs to choose from (alternatively the user can of course also type in a formula).

In case the supervisor mode has been used WinKE also provides off-line checking. This will display all errors on a tree in turn and offer the possibility to retract the associated formulas directly. You can initiate the checking of the current proof tree by choosing Check from the Problem menu. For the on- as well as for the off-line checking the user may choose the level of error reporting. Only the very basic KE rules are checked in any case, in addition you may or may not add checking for subsumption (beta simplification), analytic application of *PB*, and/or checking of the order of rule applications (like for example: analyse an alpha formula before you split a branch using *PB*, etc.).

### 4.4 Automated Deduction and Countermodels

In particular to make the system a more convenient assistant, but also to be able to demonstrate proofs to novice users, the option to automatically derive (parts of) proofs has been added. You can either ask WinKE to perform the next proof step on a selected branch, to finish a branch, or to complete an entire proof. To automatically perform a single proof step use the select tool to select an open branch marker. Then either choose Next Step from the Analysis menu or press the button showing an arrow and a little "1". The next formula that can be derived will be added to the branch automatically. If there are no further possible proof steps an adequate message comes up. To finish an entire branch select Finish Branch from the Options menu or press the button showing an arrow and a little cross after you selected that branch. To finish a proof (possibly one with several open branches at a time) in one go choose Prove from the Problem menu.

In general every formula should only be analysed once on a branch. Only gamma formulas might have to be instantiated with a new closed term several times before a branch can be closed. This is related to the semi-decidability of first order logic (for a discussion of this point the reader is referred to [Mondadori and D'Agostino, 1997] and [Fitting, 1990]). Via the Proving item of the Optinons menu you can determine the maximal number of times you want to allow an application of a gamma rule on any particular branch. If during automated proving that maximal number has been reached, but WinKE would still be able to find further steps, the system will advice you to consider to increase that number as this might (but not necessarily has to) lead to a closed branch.

In case WinKE takes too much time when calculating the next proof

steps automated dedutcion can always be interrupted by the user. This can be done via the Analysis menu or by pressing the button showing a circle on the main window. That circle appears red whenever it can be used, i.e. whenever automated proving is under way.

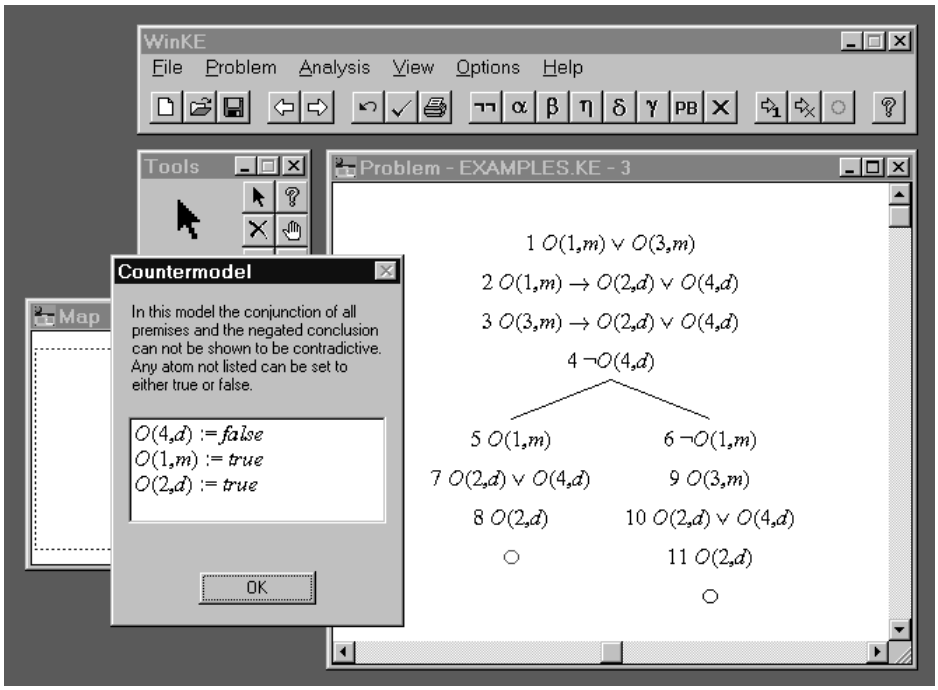


Figure 6: Deriving a Countermodel (for the left branch)

For consistent sets of formulas, i.e. if there are open branches that cannot be closed, WinKE can automatically derive the description of a countermodel. To do so, select the branch marker of a branch which cannot be further expanded and then choose Countermodel from the Analysis menu. The upcoming dialogue contains a list assigning either *true* or *false* to atoms occuring in the premises and the conclusion. Atoms which could either be *true* or *false* are omitted. Figure 6 provides an example taken from [Mondadori and D'Agostino, 1997]. The first three fromulas are the premises of the problem, formula number four is its negated conclusion. Both branches cannot be closed. The dialogue shows a countermodel associated with the left branch. You can see that the conjunction of the premises and the negated conclusion is true independently of the valuation of $O(3, m)$.

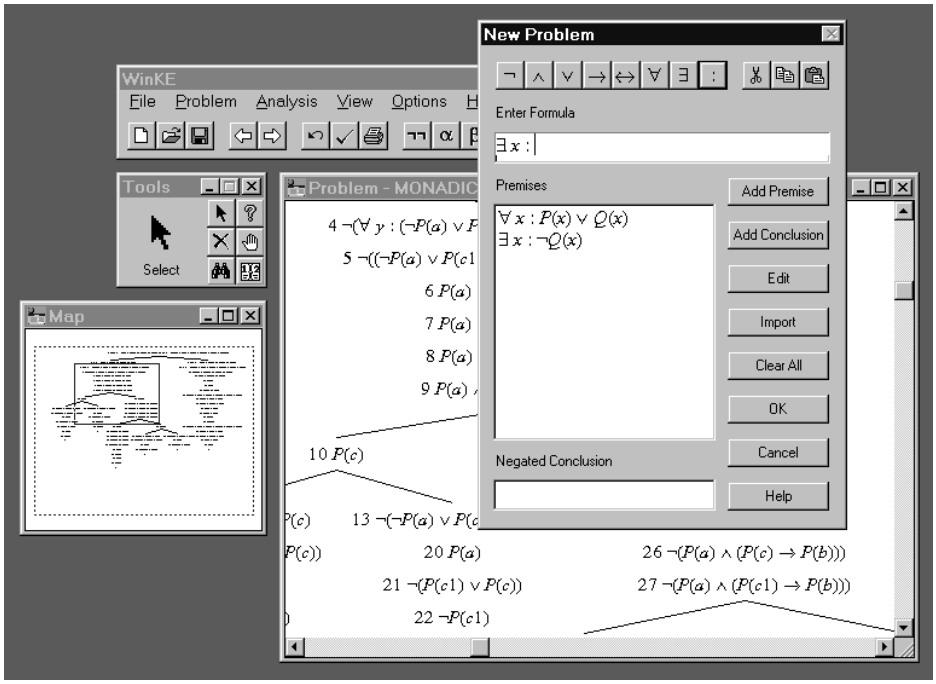


Figure 7: Editing a New Problem

### 4.5 Editing Problems

In WinKE problem files can be edited in the same environment they are worked on. You have the option to cut and paste from existing problems when defining new ones. This offers a comfortable way for teachers to write up and test new exercises. Students could be encouraged to make their own experiments trying different sets of formulas.

To edit a new problem (which will be inserted into the current file) choose New from the Problem menu. This will invoke a dialogue where you can type in premises and the conclusion of a problem. Figure 7 shows a situation where two premises have already been entered and a third one is currently written. Logic symbols can be entered in the same way as for the rule application dialogues. Pressing the Add Premise button will add the formula in the first edit field to the list of premises. If you press Add Conclusion it will be negated and then written to the text field on the bottom of the dialogue. Double-clicking the negated conclusion or a formula in the

list of premises will cause this formula to appear in the edit field, i.e. it then can be changed (pressing the Edit button has the same effect). The Import button will delete all formulas in the various fields and replace them with the premises and the negated conclusion of the current problem.

If you want to edit an existing problem, select Edit from the Problem menu. You will enter the same dialogue with the formulas of the current problem being already in the appropriate places.

Every problem is associated with a text of arbitrary length. Also that text can be edited and read directly within WinKE. In the context of a student exercise it might contain hints for finding a solution or a reference to a page of a textbook. This is done via the Text option from the Problem menu.

### 4.6 Options

Via the Options menu you can change several settings which determine WinKE's behaviour. The three teaching modes have already been discussed. The information that can be obtained using the bookkeeping tool can also be displayed directly on the proof tree. If you select Bookkeeping from the Options menu you can specify what information exactly should be visible. In Figure 8 for example the entire bookkeeping information available is displayed.

The menu items Negation, Checking, PB Strategy, and Proving are related to details of the KE prove procedure used for automated deduction and the checking of manual proofs respectively. Please refer to the appropriate pages in the help system.

By selecting Drawing Board you can change the size of the drawing board, i.e. the area visible in the window captioned Map. This could become necessary when you are working with large trees.

Parts of the functionality of WinKE can be made password protected, for example to disable automated proving, the assistant mode, or the proof checker. Like that teachers have the possibility to set up a version of WinKE which they believe to be appropriate for their students at a particular time. A dialogue to specify such a setup is entered by chooseing Setup from the Options menu. The password is set to "WinKE" (case sensitive) by default. In case you forgot the password search for "password" within WinKE Help.

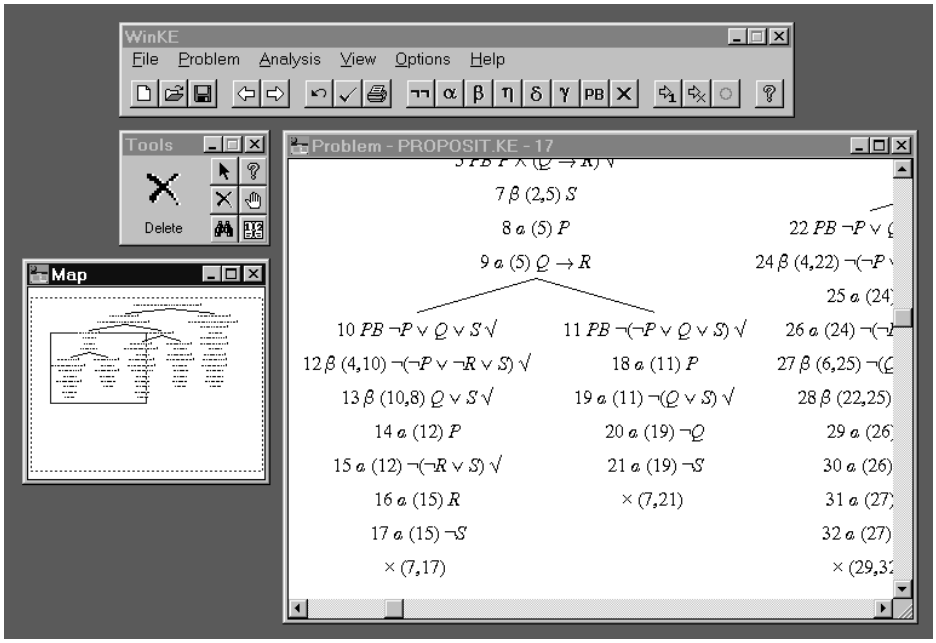


Figure 8: Displaying the entire Bookkeeping Information

### 4.7 Printing and Generating LaTeX Code

Other features of WinKE include printing of proof trees (via the Problem menu) and the option to convert such trees into LaTeX code [Lamport, 1994]. To be able to format the code you require the TreeTeX-package described in [Brüggemann and Wood, 1987] and [Brüggemann and Wood, 1988]). Select LaTeX from the Problem menu to generate the source code. For an example how to insert it into your documents have a look at the file `Winke.tex`. The trees in Table 2 of this document have actually been set up using this feature of WinKE.

## 5 A Look Inside

### 5.1 Choice of Programming Language

Being a logic programming language PROLOG offers serious advantages over imperative or object oriented languages in implementing functions related to logics and theorem proving (see [Bratko, 1990] and [Fitting, 1990]). LPA's

WinProlog [LPA, 1995] combines classical PROLOG syntax with a Windows programming environment that allows the realisation of a comfortable graphical user interface.

## 5.2 Interface Design

The design of WinKE has focussed specifically on teaching logic and reasoning. Therefore a very important task that had to be met by the design of an interface for WinKE was to assure that the process of constructing a proof tree via the program would be as close as possible to the common pen-and-paper procedure. A well designed interface distracts minimally from the principal process of learning how to apply the methods taught during lectures.

The interface of WinKE has been designed in such a way that all functions provided are transparent and easily accessible to the user. Where possible the user should have the possibility to make inputs using the mouse rather than the keyboard. As far as admitted by the special nature of WinKE the system shares common standards with other software products for Windows. This means for example that the File and the Help menus have a familiar structure. The appearance and the usage of the graphic tools is similar to that of standard graphic programs.

When using the program Tableau II [Potter and Watt, 1988], which is a pedagogic tool based on the method of semantic tableaux, the user has to determine the appearance of the proof tree by draggind the nodes to the appropriate position on the screen. This clearly is not very comfortable, takes more time than necessary, and makes it pretty difficult to build up a tidily drawn tree. For WinKE an algorithm has been implemented that always determines the "ideal" tree with respect to space requirements as well as aesthetic considerations. Given this algorithm it was possible to omit the manual positioning of formulas.

## 5.3 System Architecture

In Table 3 a schematic representation of the architecture underlying WinKE is given. The system can be seen as divided into three layers. The topmost of them is the *interface layer*. It deals with user inputs and routes them to the next layer. It also generates user outputs like messages, dialogues, or displayed graphics. The interface layer consists of four modules: the *Graphic Window Manager*, the *Tool Manager*, *the Dialogue Manager*, and the *Menu Manager*.

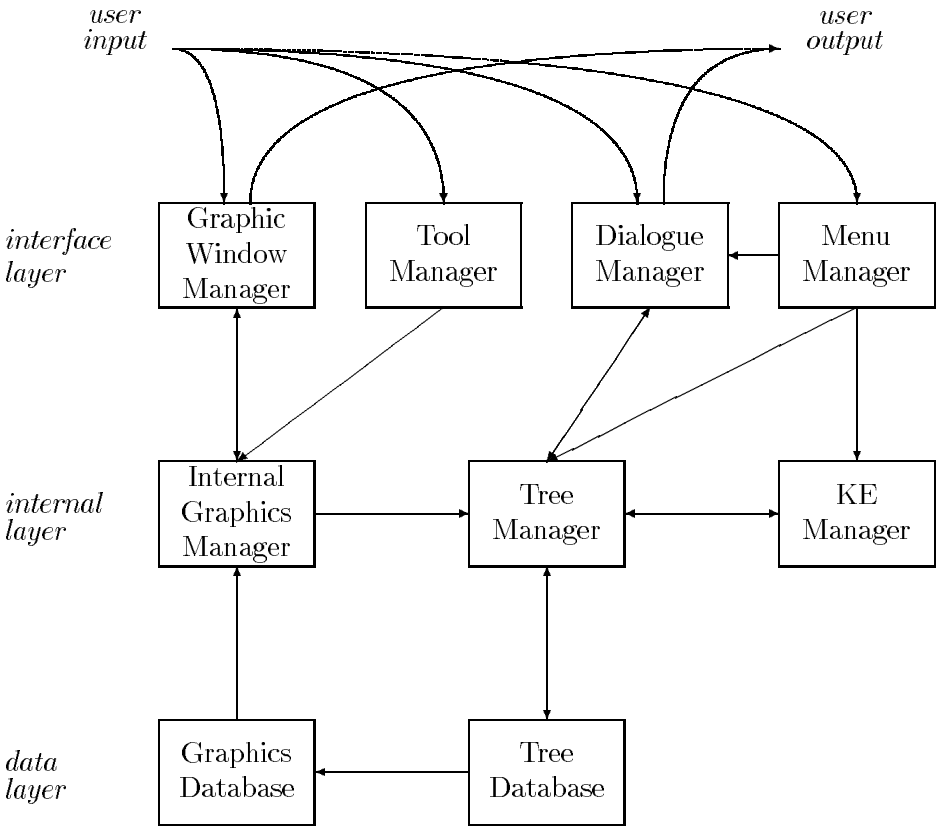


Table 3: Schematic Architecture of WinKE

In the *data layer* information about the tree structure is stored in the *Tree Database*. Parallely the *Graphics Database* contains a graphical description of the currently displayed proof tree. That description can be calculated directly given the information from the *Tree Database* by applying the tree drawing algorithm mentioned before.

The middle layer is the so-called *internal layer*. This is where the actual computations take place. The *Internal Graphics Manager* takes the information from the *Graphics Database* and passes it on to the *Graphic Window Manager*, which finally displays it on the screen. User inputs that directly affect the graphics come either through the *Graphic Window Manager* (e.g. mouse clicks) or the *Tool Manager* to the *Internal Graphics Manager*. The *Internal Graphics Manager* informs the *Tree Manager* about such events.

The *Tree Manager*, which works on the data provided by the *Tree Database*, is the most important module. Here all manipulations of the proof tree take place. Such a manipulation can be triggered by either the *Internal Graphics Manager*, the *Dialogue Manager* or the *Menu Manager*. If the desired change is due to a rule application, then the *Tree Manager* has to consult the *KE Manager* to find out whether that particular manipulation is approved by the rule checker provided by that module. The information the *KE Manager* needs for such a decision is actually provided by the *Tree Database* and the *Dialogue Manager* (which maintains the user input), but from a more systematic point of view this information can also be seen as first being collected by the *Tree Manager* and then being passed on to the *KE Manager* as a whole. Automated deduction is initiated through the *Menu Manager* and is performed within the *KE Manager*. The results are forwarded to the *Tree Manager*, which in turn invokes the necessary updates of the databases and possibly arranges for the *Dialogue Manager* to put out a message to the user. Also the settings made via the Options menu directly determine the behaviour of the *KE Manager*.

## 5.4 WinKE's Prove Procedure

WinKE's automated deduction feature can be used to perform the next proof step on a given branch, to complete an open branch, or to finish an entire proof. The *basic prove procedure* determines what is the next proof step on a given branch. To obtain a KE procedure for several consecutive steps that basic procedure has to be combined with a *branch selection strategy*. In WinKE a depth-first strategy is applied, i.e. a branch has to be completed before moving on to the next one.

### 5.4.1 The Basic Procedure

In the sequel we describe the basic prove procedure. For a given open branch it computes the next proof step to be applied.

1. If there are two complementary formulas on the branch, close it.
2. If there is a formula which is not yet analysed nor subsumed on the branch and which can be analysed by any propositional elimination rule or a delta rule, analyse the first such formula on the branch.

3. If there is a gamma formula with a useful instantiation[1] and there have been less than $N$ instantiations[2] yet, analyse the first such formula on the branch.

4. If there are valid *PB* formulas on the branch (i.e. if an *analytic* application of *PB* is possible), choose one according to the *PB selection strategy* and apply *PB* to it.

5. Otherwise no rule can be applied. But if there are still gamma formulas with useful instantiations left, which cannot be analysed, because there have been $N$ instantiations already, inform the user that s/he should increase $N$ and try again.

### 5.4.2 Restricting Instantiation

WinKE is not applying a free-variable calculus. Such an approach though usually being preferred in automated theorem proving would have caused several difficulties with the graphical representation. First of all there would have been a problem in graphically distinguishing between free variables and constants. One way would have been to reserve $x$, $y$, ... for variables and $a$, $b$, ... for constants, but on the other hand that would have been too restrictive. Another problem would have been to make unificaction really "visible" to the user.

Therefore, in WinKE when analysing a delta or a gamma formula instantiations have to be made immediately. For the delta rule this is very simple, we just have to find a new constant. For the gamma rule the simplest approach would be to consider the entire set of ground terms that can be formed using the terms appearing on the proof tree.

But not all the terms of that "universe" will prove to be "useful" in closing a branch. For example when analysing $\forall x : P(x)$ it doesn't make sense to instantiate $x$ with $a$ just because there is $Q(a, b)$ somewhere on the same branch. So already before applying a gamma rule it is possible to exclude some of the terms which won't be useful. The strategy applied in WinKE is to try to unify every atom of the chosen gamma formula with every atom on the branch. From the resulting set of unificators the substitutions for the quantified variable to be instantiated are extracted. They form the set of *useful instantiations*, i.e. they are the only candidates for instantiation. Only if that set is empty and the gamma rule is applied for the first time

[1] see next paragraph on *restricting instantiation*

[2] Taking the semi-decidability of first order logic into account a maximum number of gamma instantiations has to be specified by the user.

to that particular formula on that particular branch, we have to choose an arbitrary instantiation, say $c1$.

### 5.4.3 PB Selection Strategies

In the paragraph on the *basic prove procedure* we mentioned a *PB selection strategy*. Such a strategy governs which *PB* formula to pick when there are several candidates for an analytic *PB* application.

In WinKE the user can choose between two different strategies from the *PB strategy dialogue* accessible via the Options menu. The first one is simply to take the first *PB* formula appearing on the particular branch. In the advanced strategy the *PB* formula appearing *most frequently* on the branch is chosen. If there are several ones appearing equally often the first one of them is used. Applying that strategy you would expect that one application of *PB* should usually allow for more subsequent applications of a beta or an eta rule than with the simple strategy. Note that this is only an heuristic. Currently there is no optimal *PB* selection strategy known.

### 5.4.4 Checking Manual Proving

The checking of proof steps initiated manually by the user works similarily to automated deduction. The differences are that in manual mode you can instantiate gamma formulas as often as you like, there are no restrictions for those instantiations, and you don't have to follow a *PB* selection strategy. Before *PB* can be applied on a particular branch, every gamma formula on that branch has to be instantiated at least once. Also you don't necessarily need to select the first formula on a branch that can be analysed.

By selecting Checking from the Options menu the user may specify to make WinKE's checking more or less restrictive. Via the upcoming dialogue s/he can choose whether to allow the analysis of subsumed beta formulas or not, whether to restrict *PB* to analytic applications or not, and whether to check for an efficient rule application order or not. The latter one means to close branches whenever possible, to apply elimination rules before *PB*, etc.

## 6 Conclusion

We have presented the pedagogic tool WinKE. Its principal task is to support teaching in the context of an introductory course on elementary classical logic. The software is complementary to the logic textbook

[Mondadori and D'Agostino, 1997] which is based on KE. Given the automated deduction feature the system can also be used as a proving assistant. It is easy to learn, comfortable to use, and visually satisfying. Teachers will particularly benefit from WinKE when preparing and testing exercises for their students. The system has successfully been used in introductory courses on logic held at the Department of Human Sciences at the University of Ferrara and at the Department of Electrical and Electronic Engineering at Imperial College, London.

Other logic tutors include for example popular programs like Tarski's World [Barwise and Etchemendy, 1991]. Using Tarski's World students are asked to verify first order formulas stating propositons about simple three-dimensional worlds inhabited by geometric objects. Though it might be useful in teaching the very basics (in particular how to translate 'real world' situations into logic), it does not deploy a systematic prove procedure and therefor cannot be called a proving assistant. The Hyperproof program [Barwise and Etchemendy, 1994] is a derivative of Tarski's World. It is used to construct proofs of sentences on that same geometric world applying a natural deduction like calculus. As it is restricted to examples of that particular domain is is difficult to be compared with WinKE. The highly sophisticated interface of Hyperproof definitely makes it very attractive to students, but at the same time it also makes it more difficult to learn how to use the tool. WinKE has been designed to simulate an existing prove procedure. In that sense it is supportive of the teaching process. For Hyperproof on the contrary teaching has to be centered around the software. Tableau II [Potter and Watt, 1988], which has already been mentioned in Section 5, is based on semantic tableaux. As far as interface and usability are concerned WinKE clearly offers noticeable advantages over Tableau II (which is not that surprising given its age).

Each of the other three tools comes with a comprehensive set of examples. For WinKE so far only a few ones have been included, but users can always set up their own ones. It is planned to edit problem files covering all examples and exercises given in [Mondadori and D'Agostino, 1997] in the near future.

# References

[Barwise and Etchemendy, 1991] Jon Barwise and John Etchemendy. *Tarski's World.* CSLI Publications, Stanford, 1991.

[Barwise and Etchemendy, 1994] Jon Barwise and John Etchemendy. *Hyperproof.* CSLI Publications, Stanford, 1994.

[Bratko, 1990] Ivan Bratko. *Prolog Programming for Artificial Intelligence.* 2nd edition, Addison-Wesley Publishing Company, 1990.

[Broda *et al.*, 1995] Krysia Broda, Marcello D'Agostino, and Marco Mondadori. *A Solution to a Problem of Popper.* In *The Epistemology of Karl Popper*, Kluwer, 1995.

[Brüggemann and Wood, 1987] Anne Brüggemann-Klein and Derick Wood. *TreeTEX: Documentation and User Handbook.* Technical Report, University of Waterloo, 1987.

[Brüggemann and Wood, 1988] Anne Brüggemann-Klein and Derick Wood. Drawing Trees Nicely with TEX. In *Proceedings of the Conference of the Third European TEX Meeting*, Exeter, 1988.

[D'Agostino *et al.*, 1997] Marcello D'Agostino, Ulrich Endriss, Dov Gabbay, Jeremy Pitt, and Marco Mondadori. A New Approach to the Mechanization of Deductive Reasoning (in Italian). In *Proceedings of the Conference of the Italian Philosophical Society on "Man and Machine: thirty years later"*, University of Bari, 24–26 October 1997, to appear.

[D'Agostino and Mondadori, 1994] Marcello D'Agostino and Marco Mondadori. The Taming of the Cut. Classical Refutations with Analytic Cut. *Journal of Logic Computation*, 4(3):285–319, 1994.

[Endriss, 1996] Ulrich Endriss. *A KE Based Theorem Proving Assistant.* Master's thesis, Department of Computing, Imperial College, London, 1996.

[Fitting, 1990] Melvin Fitting. *First-Order Logic and Automated Theorem Proving.* Springer Verlag, 1990.

[Lamport, 1994] Leslie Lamport. *LATEX: A Document Preparation System.* 2nd edition, Addison-Wesley Publishing Company, 1994.

[LPA, 1995] *LPA-Prolog for Windows*, Version 3.0, Logic Programming Associates Ltd., London, 1995.

[Mondadori and D'Agostino, 1997] Marco Mondadori and Marcello D'Agostino. *Logica.* Edizioni Scolastiche Bruno Mondadori, Milan, 1997.

[Pitt, 1995] Jeremy Pitt. MacKE: Yet Another Proof Assistant & Automated Pedagogic Tool. In P. Baumgärtner, R. Hähnle, and J. Possegga,

editors, *Theorem Proving with Analytic Tableaux and Related Methods*, Springer Verlag, 1995.

[Potter and Watt, 1988] Michael Potter and Duncan Watt. *Tableau II: A Logic Teaching Program.* Oxford University Computing Services, Learning and Resource Centre, Oxford, 1988.