ICOT Technical Report: TR-430

TR-430

EUODHILOS: A General-Purpose Reasoning Assistant System —Concept and Implementation—

by
T. Minami, H. Sawamura,
K. Satoh, and K. Tuchiya
(Fujitsu)

October, 1988

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191 - 5
Telex ICOT J32964

Institute for New Generation Computer Technology

EUODHILOS: A General-Purpose Reasoning Assistant System

— Concept and Implementation —

Toshiro MINAMI*, Hajime SAWAMURA*, Kaoru SATOH**, and Kyoko TUCHIYA**

* International Institute for Advanced Study of Social Information Science (IIAS-SIS),
FUJITSU LIMITED, 140 Miyamoto, Numazu, Shizuoka 410-03, Japan

E-mail: {tos,hajime}%iias.fujitsu.junet@uunet.uu.net

** Software Development Laboratory, *FUJITSU LABORATORIES LTD.*,
1015 Kamikodanaka, Nakahara-ku, Kawasaki, Kanagawa 211, Japan

Abstract

The concept of the general-purpose reasoning assistant system is presented. We are now developing a system named EUODHILOS as a prototype of that kind of system. It assists the user in describing the syntax of the logical expressions, defining a variety of logics which consists of axioms, inference rules, and rewriting rules, and also in constructing proofs for theorems under the logic so defined. Proofs are constructed under the support of the facility named "sheet of thought", which is an interactive and visual proof editing environment. Proofs are expressed and edited in the tree structured forms of the natural deduction like style.

1. Introduction

In these days, logics play important and even essential roles in many fields like mathematics, computer science, artificial intelligence, and so on. In fact, various logics such as first-order, higher-order, equational, temporal, modal, intuitionistic, and type theoretic logics are used in these fields, and human reasoning activities based on these logics appear in daily work.

Here the phrase "human reasoning" is used to denote the process consisting of the following three phases.

(1) Making mental images about the objects and concepts.

(2) Making logical models which describe the mental images.

(3) Examining the models to make sure that they are sufficient.

The process begins with the first phase when one becomes aware of some mental images of objects and concepts having some structures and also wants to clarify what they are. To clarify the mental images, one has to describe them formally. A formal framework for describing objects and concepts is called a "logic" in this paper. A "logical model" is a logical description which specifies some mental image.

The second phase is for making logical models. Since the language is important for describing objects, at first, one defines the syntax of the language. In this phase, one has to determine what objects, concepts, and relations appear in the universe of discourse. The logical structure of the world of objects can be described by giving the axioms and derivation rules. The derivation rules are given in forms of such as inference and rewriting rules.

In the third phase, one derives results from the logical model. One comes to know many formal properties of the model. At the same time, one examines the correctness of the model. The model is insufficient if some properties which are expected to hold by the image of the objects fail to prove in the model. In this case, one has to modify some or all of the logical expressions about the objects. Sometimes one has to modify not only the logical expressions, but also the definition of the language used for the modeling.

Since human beings reason in various fields, it is valuable to help them by a computer system. The purpose of the reasoning assistant system lies in increasing the efficiency of the human reasoning process under the aid of the computer.

Two major subjects are pursued for realizing a reasoning assistant system. The first one is the "generality" of the system. Here the phrase "general-purpose system" indicates that it is logic-free. As S. K. Langer told [Langer 25], we recognize that "Every universe of discourse has its logical structure." That is a thought that for each object which we mention, there must be a logic best suited for expressing about it. In order

to assist human reasoning for the object, the reasoning assistant system must have the ability to allow the user to describe all the existing logical structures and manipulate the expressions under those logics. Our system EUODHILOS (pronounced "you-oh'-dee-los") is named as an acronym of the phrase by Langer. This reflects our intention to emphasize the generality of the system.

The other subject is the "user-friendliness." We investigate the reasoning-oriented human-computer interface that can be established as an aspect of reasoning supporting facilities. A system having good interface so that it can be used easily, is helpful for one to conceive ideas in reasoning. Furthermore, a reasoning methodology, which often reminds us of programming methodology, needs to be investigated.

We aim at building an ideal reasoning assistant system: which can be used for general-purpose and is used easily. EUODHILOS is a prototype of such a system. We intend to clarify the concept of the ideal general-purpose reasoning assistant system through developing and using EUODHILOS.

In Section 2, the reasoning assistant system is characterized through the comparisons to several other types of system which can be used for assisting human reasoning, specifically to theorem provers, proof checkers, and proof constructors. In Section 3, an overview of EUODHILOS is illustrated. In the succeeding two sections, its two significant features are described. In Section 4, the feature of defining logics is explained. In Section 5, the feature of assistance for constructing proofs, called "sheet of thought", is explained. In Section 6, some of the other features of EUODHILOS are illustrated. In Section 7, some concluding remarks and the directions for future research are stated. In Appendix, several proof examples are exhibited.

2. Characterization of the Reasoning Assistant System

We consider the following four as the system-types which can be used for assisting human reasoning:

(i) reasoning assistant system

(ii) automated theorem prover

(iii) proof checker

(iv) proof constructor

In the rest of this section the reasoning assistant system is comparatively characterized with other types of system. In Subsection 2.1 the reasoning assistant system is introduced, and its characteristic features are exhibited. In Subsections 2.2 to 2.4, it is compared with each of the other types of system.

2.1 Reasoning Assistant System

The purpose of the reasoning assistant system is to support the whole phases of human reasoning process. The concept of the system comes from the philosophy of Langer [Langer 25] introduced in Section 1. and the recognition that the (mathematical) reasoning proceeds through "Proofs and Refutations" [Lakatos 76].

The following two major characteristic features of the system reflect these underlying concepts respectively:

(i) Logic free

(ii) Possession of user-friendly proof editing environment

Figure 2.1 is an illustration of how the reasoning assistant system is used. In the upper half of the figure which corresponds to the feature (i), the user specifies each of the components of logic, i.e., symbols, syntax of expressions including (logical) formulas, inference rules, etc. In the lower half of Figure 2.1, corresponding to (ii), the user tries to construct proofs of theorems under the logic defined in the previous step. Proofs are searched by trial and error.

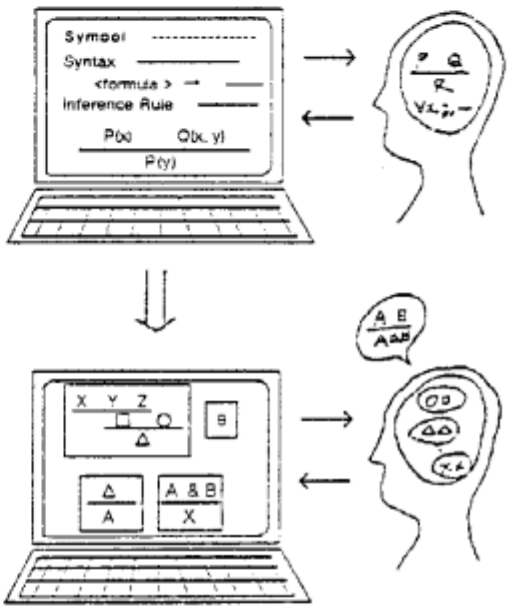


Figure 2.1 Reasoning Assistant System

EUODHILOS is the only one reasoning assistant system. In the system, partially constructed proofs, which are called proof fragments, appear scatteringly on a sheet of thought. The user edits these proof fragments by the editing commands such as create, delete, extend (derive), connect, separate, and so forth. The sheet of thought is the environment for creating theorems and their proofs. The theorems on the sheet can be saved so that they may be reused in the later proofs for other theorems. They can be used just in the same way as axioms.

2.2 Automated Theorem Prover

An (automated) theorem prover is a system which searches a proof of a given formula. Figure 2.2 illustrates how the theorem prover is used. In the upper half of Figure 2.2, a user tries to prove a formula. The user starts by putting an assumption of the formula. But he has no idea how to proceed the proof. In the lower half of Figure 2.2, the user gives the formula to the theorem prover. The system finds out a proof, and displays it on the screen.

The theorem prover is different from the other types in that the proofs of theorems are given by the system. In other types of systems, including the reasoning assistant system, proofs are given by the user.

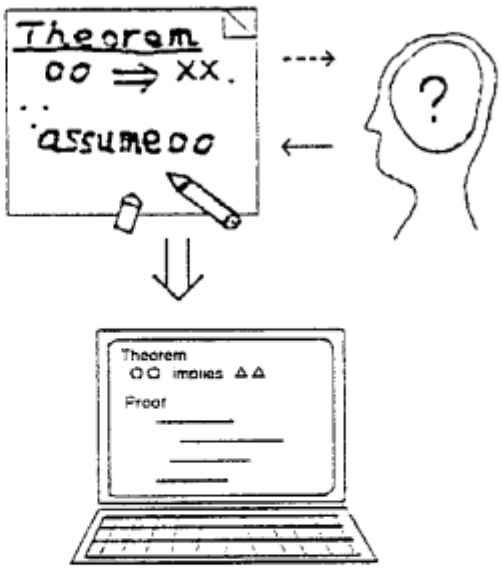


Figure 2.2 Theorem Prover

The situation in Figure 2.2 is, in fact, an ideal one. Considering the current state of the art of the proof-finding power of theorem provers, there is little hope of finding practically useful proofs automatically by computers. From this observation, theorem provers are to be used for finding only those proofs which fill small gaps which appear in a large proof. If the gap in a proof is small, the prover can find the proofs effectively. On one hand human beings are good at making a plan how to find a proof; on the other, they are not good at doing things accurately. Machines are on the contrary to human beings. This is the reason why we pay attention to support the process of human reasoning by the computer. The capacity of computers for finding proofs can be used as a part of a reasoning assistant system.

2.3 Proof Checker

A proof checker is a system which checks the correctness of a proof described by the user. Figure 2.3 is an illustration of how the proof checker is used. In the upper half of the figure, the user makes a proof of a theorem. A human proof may contain some careless mistakes including small gaps in a proof. The checker provides a language for describing human proofs. By using this language, the user describes the proof. In the lower half of the figure, the user gives the description of the proof to the checker. If the checker finds errors in the proof, it shows them on the screen. We can see an error indication in Figure 2.3.

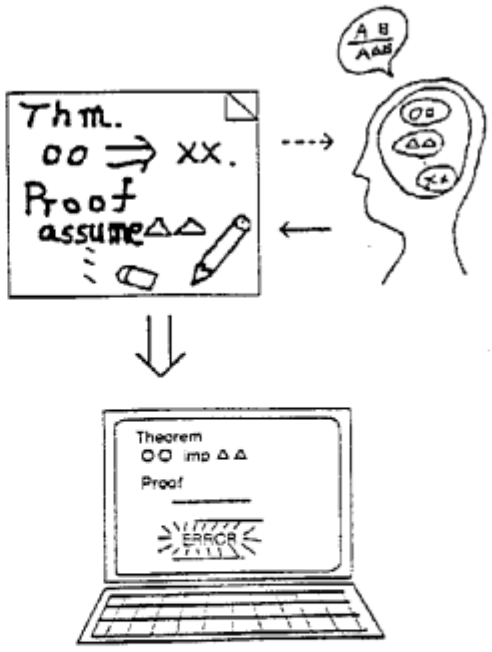


Figure 2.3 Proof Checker

When a user has a proof and wants to verify its correctness, a proof checker is one of the best tools for him. But when one begins to find a proof for some formula, proof constructors described in the next section is more suitable for his purpose of proof finding.

Many proof checkers have been developed up to now. AUTOMATH [de Bruijn 70] is a proof checker in which the user can specify how the proofs are constructed. PL/CV2 [Constable 82] is used for proving the correctness of PL/I like programs. CAP-LA [ICOT 86] deals with the proofs on linear algebra.

2.4 Proof Constructor/Editor

A proof constructor is a system which supports a user to construct proofs as well as theorems through the interaction between the user and the system. The proof construction is, in other words, a "proof editing." Users edit proofs, precisely proof fragments, by inputting, deleting, and combining the proofs. From this point of view, a proof constructor is a proof editor. The function of the proof constructor is included in a reasoning assistant system as an aid to proof constructions.

Figure 2.4 is an illustration of how the proof constructors are used. The user interactively uses a proof constructor by entering some commands and tries to find a proof of some theorem.

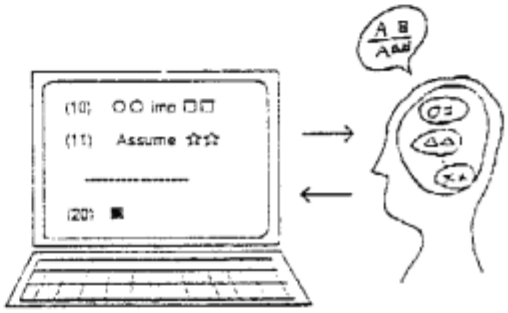


Figure 2.4 Proof Constructor

Many proof constructors have been developed up to now. For example, LCF [Gordon 79], FOL [Weyhrauch 80], EKL [Ketonen 84] and Nuprl [Constable 86] are well known.

The purpose of Nuprl is very similar to ours. It aims at providing the proof construction environment for various logics. But the approach to the realization of it is different from those of the reasoning assistant systems. Nuprl has a fixed underlying logic and other logics must be defined by using the expressions in this logic. In the approach of reasoning assistant system, even the syntax of the logic is specified by the user. It aims at the complete realization of logic free systems which can assist human reasoning in the various fields. Proof construction methodologies are also different. In Nuprl, proofs are constructed only by refinement, while in the reasoning assistant system, proofs are constructed by three types of deductions; i.e. by forward, backward (same as refinement), and interpolating deductions. The interpolation is a deduction which fills a proof gap between two formulas. Another difference is on user-interfaces: that is, Nuprl has been based on character displays, while EUODHILOS uses bit-mapped displays so that symbols can be expressed as its natural shape on a paper.

The significant difference between the proof constructors and the reasoning assistant systems is that in the former, underlying logics are fixed, while in the latter, underlying logics can be defined by the user. There are merits and demerits for fixing the underlying logics. As a merit it is easy to introduce some specific procedures suited to the logic. As a demerit, if the system is applied for general cases of human reasoning, the fixation of logic may restrict the reasoning about some objects under consideration. In such a case, a general framework treating a variety of logics is required.

3. Overview of EUODHILOS

EUODHILOS is a prototype of the general-purpose reasoning assistant system. Its first version is now working on the sequential inference machine PSI on an operating system named SIMPOS.

It is designed by considering the following issues:

(1) Realization of a logic-free reasoning system, based on the philosophy in [Langer 25].

(2) Provision of an environment to be used easily to those who are not necessarily familiar with computers.

(3) Support of logical thought, symbolic or logical manipulations done by human reasoners.

(4) Environment for experimenting logical model construction based on the philosophy in [Lakatos 76].

The issue (1) is reflected to EUODHILOS in its logic definition feature which is shown in Section 4. For (2), it is implemented under the overlapping window system with bit-mapped display. It can be used easily because most of the operations are achieved by selecting the menu item with the mouse. For (3) and (4), the major tools used in EUODHILOS are "editors." The user can modify the language definitions, logic definitions, proof constructions, and so forth directly through the visual editor windows. Considering the importance of the proof construction, the system provides a special proof editing environment called "sheet of thought", which is an interactive, easy to use, and visual proof constructor.

Figure 3.1 shows an overview of EUODHILOS. The system consists of two major parts, one for defining a user's logical system and the other for constructing proofs in a sheet of thought. The former is concerned with the second phase of human reasoning discussed in Section 1. It provides the feature for defining the syntax of logical expressions called (well-formed) formulas and the logical structure of the mentioned objects, as well. The latter is concerned with a proof construction which corresponds to the third phase of human reasoning. It is a proof editing environment, where the user constructs the proofs by trial and error. On the sheet of thought, proofs are expressed in the form reflecting

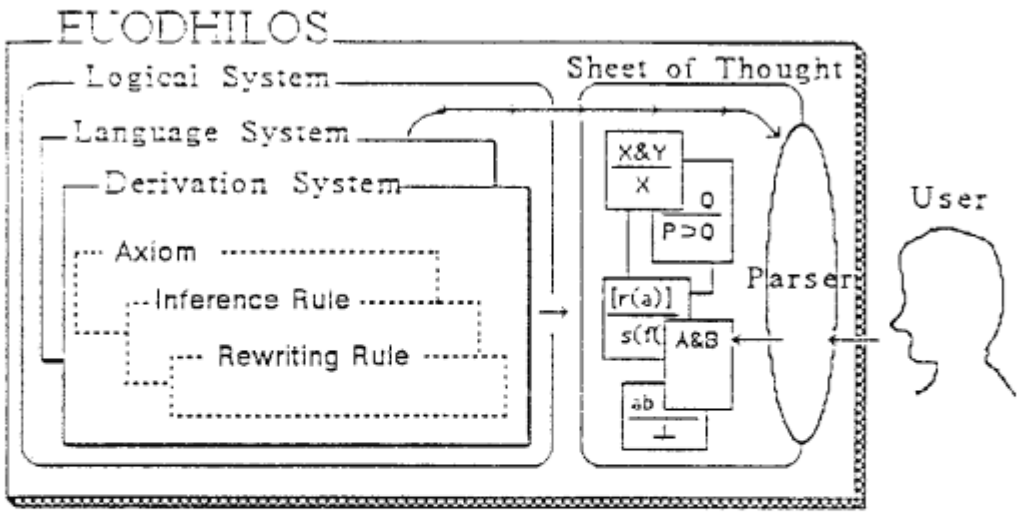


Figure 3.1 An Overview of EUODHILOS

their derivation structures. In EUODHILOS proofs are represented in natural deduction style, i.e. a tree structure, they are shown in their familiar representation.

Among the various features of EUODHILOS, we put stresses on the following topics:

(i) Defining Logics

(ii) Constructing Proofs (Sheet of Thought)

In the following Sections 4 and 5, the features of defining logical expressions and supporting to construct proofs on the sheet of thought are explained respectively mainly based on the specification for the current (the first) version of EUODHILOS.

4. Defining Logics

4.1 Language Description Feature of EUODHILOS

In EUODHILOS, a language system to be used is designed and defined by the user at the first stage. The language system consists of the specifications of the syntax of the logical expressions (i.e. formulas). The syntax of the expression is given by using definite clause grammar (DCG)[Pereira 80] in the first version. We intend to modify DCG by adding the operator-declarations in the second version in order to decrease the amount of descriptions. From the description in DCG, a bottom-up parser called BUP[Matsumoto 83] can be automatically generated. We modify the BUP algorithm by adding the function so that the augmented operators can be treated.

The system generates not only a parser, but also an unparser for the defined language. The unparser translates from the internal expressions into external ones which can be understood by the user. The parser and unparser are used in all the following phases of symbol manipulations. When an expression is entered, the parser is invoked for

checking the validation of it. At the same time the internal structures of the expression of the language are constructed as well. When derivation commands are given by the user, the internal expressions of the formulas are manipulated and new internal expressions are generated. These expressions are presented to the user after translated into the external ones by the unparser.

Figure 4.1 is an example description of the first-order logic in DCG with operator and symbol declarations.

Syntax description:

```
formula('⇒'(F1,F2))—formula(F1), "⇒", formula(F2)
formula('∧'(F1,F2))—formula(F1), "∧", formula(F2)
formula('∨'(F1,F2))—formula(F1), "∨", formula(F2)
formula('¬'(F))—"¬",formula(F)
formula(F)—atomic_formula(F)
atomic_formula(F)—"(",formula(F),")"
atomic_formula(F)—predicate_symbol(P),"(",term_list(TL),")",
                  {F=..[P|TL]}
term_list([T])—term(T)
term_list([T|TL])—term(T),",",term_list(TL)
term(T)—function_symbol(F),"(",term_list(TL),")",{T=..[F|TL]}
term(T)—variable_symbol(T)
term(T)—constant_symbol(T)
```

Operator declaration:

```
"⇒":  xfx, 100
"∧":  yfx, 50
"∨":  yfx, 50
"¬":  fy, 30
```

Symbol declaration:

```
predicate_symbol:  "p"-"r"
function_symbol:  "f"-"h"
variable_symbol:  "x"-"z"
constant_symbol:  "a"-"e"
```

Meta symbol declaration:

```
formula:  "P"-"T"
term:  "A"-"D"
```

Figure 4.1 A description of the first-order logic

4.2 Derivation Description Feature of EUODHILOS

A derivation system in EUODHILOS consists of axioms and derivation rules which are given as inference and rewriting rules. A finite set of formulas is given as the axiom system. Inference rules are given in a natural deduction like style presentation by the

user. An inference rule consists of three parts, where the first is the premises of a rule, each of which may have an assumption, the second is the conclusion of a rule, and finally the third is for the restriction that is imposed on the derivations of the premises, such as variable occurrence conditions (eigenvariable). Well-known typical styles of logic presentations such as Hilbert's style, Gentzen's style, equational style can be treated within this framework.

Schematically, inference rules are given in the natural deduction style format as follows:

$$\frac{\begin{array}{cccc}[\text{Assumption}_1] & [\text{Assumption}_2] & \cdots & [\text{Assumption}_n] \\ \vdots & \vdots & & \vdots \\ \text{Premise}_1 & \text{Premise}_2 & \cdots & \text{Premise}_n\end{array}}{\text{Conclusion}}$$

In this format, each of the assumption parts is optional. If a premise has the assumption, it indicates that the premise is obtained under the assumption, and otherwise it is obtained unconditionally. An inference rule may have a condition on the eigenvariable. An inference rule is applied if all the premises are obtained in this manner, and the restrictive condition is satisfied. Then, the conclusion is obtained by the application of the rule.

Rewriting rules are presented in the following format:

$$\frac{\text{Pre\_Expression}}{\text{Post\_Expression}}$$

A rewriting rule indicates that it is applied to an expression when the formula has a subexpression which matches to the pre_expression part of the rule. The resultant expression is obtained by replacing the subexpression with the expression corresponding to the post_expression part of the rule. Rewriting rules have no condition of application in the first version.

Iterating the applications of the derivation rules described above, one can get a derivation (tree). The following Figure 4.2 is an example of the derivation using both inference and rewriting rules.

$$\cfrac{\cfrac{\cfrac{\cfrac{\cfrac{\forall xyz.(x+y)z = xz+yz}{(a+0)b = ab+0b}\,(\forall \mathrm{E})}{ab = ab+0b}\,(x+0 \multimap x)}{0 = 0b}\,(x = y \multimap x+z = y+z)}{0b = 0}\,(x = y \multimap y = x)}{\forall x.0x = 0}\,(\forall \mathrm{I})$$

Figure 4.2 A derivation in EUODHILOS

5. Constructing Proofs

In EUODHILOS an environment called the "sheet of thought" provides the assistance to find proofs of theorems by trial and error. This originates from a metaphor of work or calculation sheet and is apparently analogous to the concept of sheet of assertion due to C. S. Peirce [Peirce 74]. It allows one to draft a proof, to compose proof fragments, detach a proof, to reason by using lemmas, and so on.

On the sheets of thought, proof fragments (in other words, partially constructed proofs) are the elementary units for manipulation. Proof fragments are newly created as assumptions, axioms, or theorems of the theory, which are composed, and deleted according to the operations given by the user.

In a sheet of thought, applications of inference and rewriting rules are possible in the same style as those users use on the paper. This naturally induces that the appearance of a proof structure on the sheet is also the same as that on the paper. This way of treating is considered as an example of the proof visualization.

It is desirable that reasoning during proof construction can be done along the natural way of thinking of human reasoners. Therefore EUODHILOS supports the typical method for reasoning, that is, forward (or top-down) reasoning, backward (or bottom-up) reasoning, interpolation (i.e. filling the gap between proof fragments) and reasoning in a mixture of them. They are accomplished interactively by manipulating the fragments on a sheet of thought. It is planned to incorporate not only such a proving methodology but also methodology of science (e.g., Lakatos' mathematical philosophy of science [Lakatos 76], Kitagawa's relativistic logic of mutual specification [Kitagawa 63], etc.).

As an example of deduction process on a sheet, we will illustrate how one can proceed the forward deduction. In order to deduce forward by applying an inference rule, one has to start by selecting the formulas used as premises of the rule. The selection is achieved with the mouse by clicking the desired formula. Then the user may select the inference rule by calling the operation menu of the sheet, or he may enter a formula as the resultant formula. If the user selects a rule, then by the "do it" command (action) the system applies the rule to the premises and deduces the resultant. If the user indicates the resultant formula, then the system searches the list of deduction rules and tries to find one which is applicable to this deduction.

Among the editing functions on a sheet of thought such as delete, copy, move, undo, separate, etc., we will illustrate two of them; connection and separation.

(1) Connection

The user can connect two proof fragments if they have equal formulas, where one of them must be a resultant of a fragment and the other a hypothesis of another fragment. The connection operation begins with selecting the two formulas by clicking them with

the mouse. One of them is a resultant, while the other a hypothesis which must be equal to the resultant. And then, he invokes the operation menu and selects the "connect" item. As the result, the proof fragments are connected into the one fragment. Figure 5.1 is an illustration of this operation. The two fragments in the left part of the figure are connected at the formula "B", and the fragment in the right part is obtained.

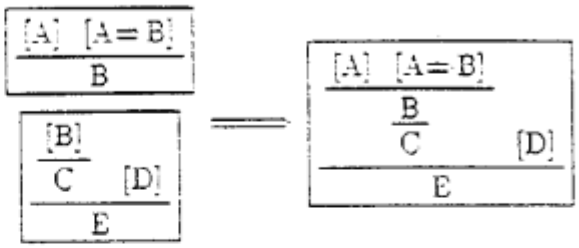


Figure 5.1 Connection at B

(2) Separation

This operation is just the inverse of the connection. When the user indicates a formula in a proof fragment and selects the command "separate" in the operation menu, then the fragment is separated into two parts at the position of the selected formula. We will omit the figure for this command because it gets the two proof fragments from a selected one in just the inverse way of Figure 5.1.

Figure 5.2 is an example of the screen image of sheet of thought in EUODHILOS.

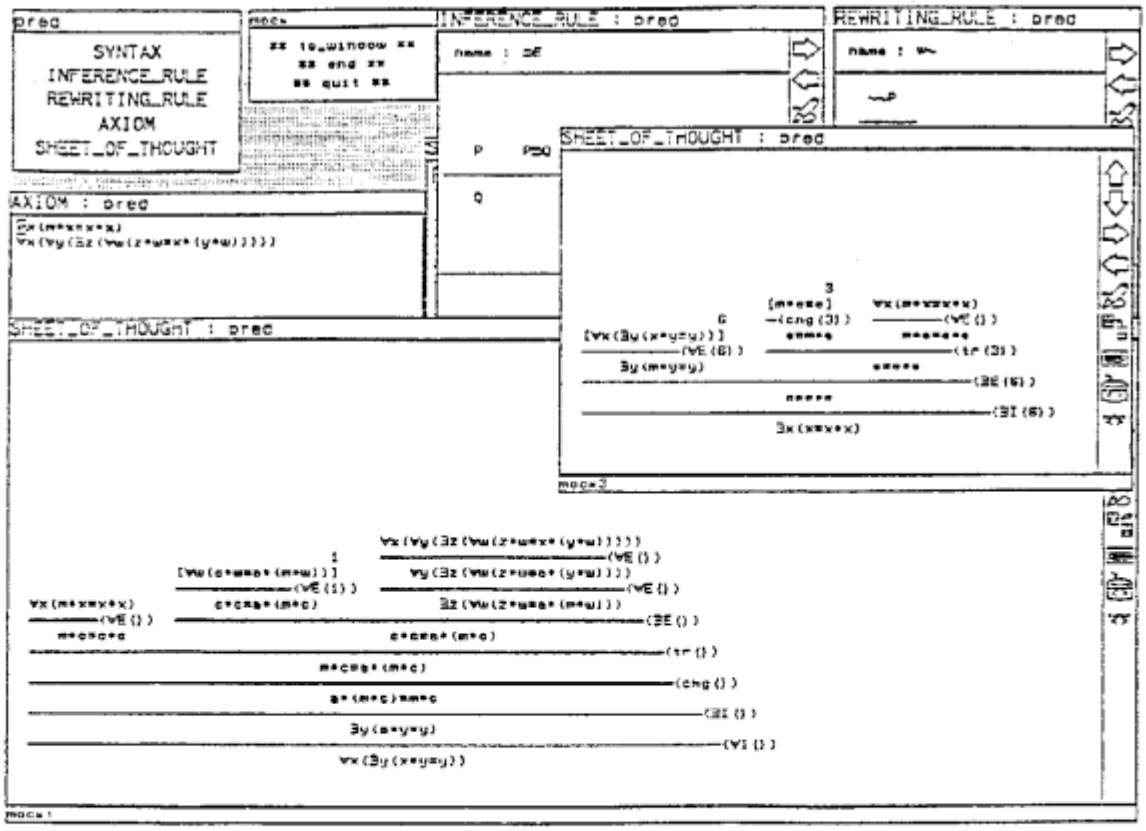


Figure 5.2 Sheet of Thought

6. Other Facilities for Reasoning

In order to make the system user-friendly and easy to use, we have to pay much attention to the visualization of things. For this purpose, the bit-mapped display with multi-window environment, mouse, icon, pop-up menu, etc. are exploited in the implementation of EUODHILOS. The followings are available as the user-interface facilities for inputting formulas, visualizing formulas, and assisting reasoning.

6.1 Software Keyboard and Font Editor

The software keyboard and the font editor are used to design and input special symbols often appearing in various formal systems. It is a matter of course that provision of special symbols which reasoners are accustomed to use makes it possible to reason in a natural way of thinking on a computer. The user designs the fonts of the symbols in the dot matrix pattern of them by using the built-in font editor. These fonts are assigned on the software keyboard and are used with the corresponding keys. In Figure 6.1, one can see a software keyboard on which user-defined symbols such as $\vee$, $\wedge$, $\Rightarrow$, $\exists$, etc. are attached.

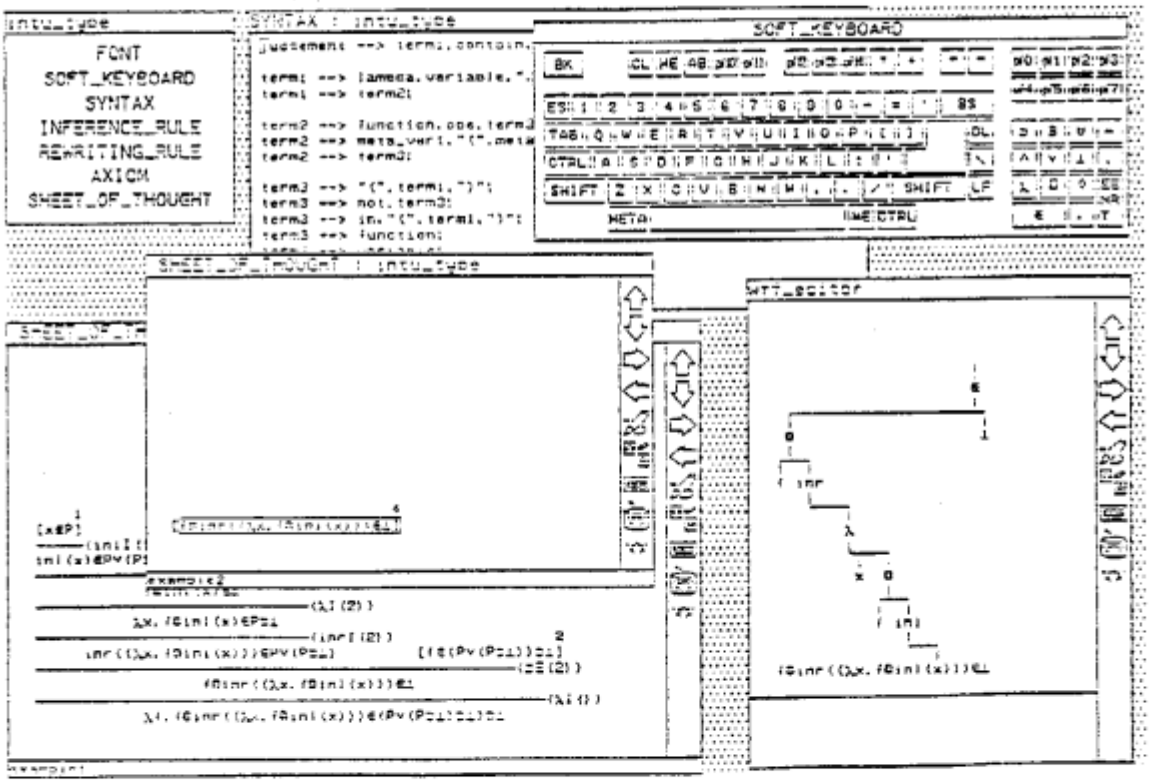


Figure 6.1 Using the user-defined fonts in the system.

6.2 Formula Editor

The formula editor is a structure editor for logical formulas. It aims to simplify

inputting formulas by displaying complicated formulas. In addition to ordinary editing functions, it provides some formula rewriting functions. It can be used not only for entering formulas, but also for just displaying the structure of formulas clearly. This is a visualization of formulas. You can see the formula editor in the bottom-right corner of Figure 6.1.

6.3 Stationery for Reasoning

Independently of a logic under consideration, various reasoning tools such as decision procedures become helpful and useful in reasoning processes. In a sense it may also play a role of a model which makes up for a semantical aspect of reasoning. You can display and erase the stationery at any time within the system. Currently, a logic calculator for Boolean logic is realized as a desk accessary. Figure 6.2 indicates what it looks like on the screen.

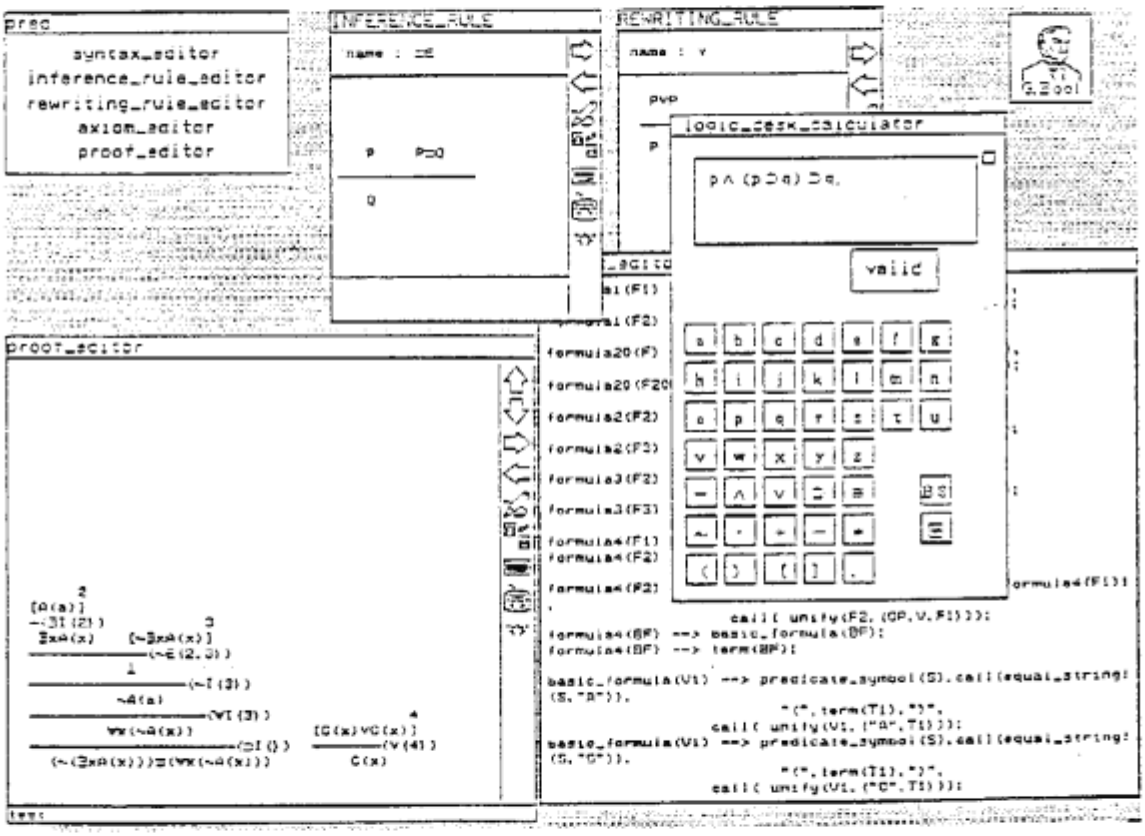


Figure 6.2 Boolean logic calculator

7. Concluding Remarks and Directions for Future Research

The first version of EUODHILOS is now available and the second version is under development. The system is being improved by reflecting the experience of using it.

So far, we have dealt with logics, such as first-order logic (NK), propositional modal logic (T), intensional logic (IL), combinatory logic, and Martin-Löf's type theory. Many

logics can be treated in the current version. Though, some logics such as tableau method can not be treated in the current framework for logic description. We intend to extend the framework so that these "irregular" logics can be treated in the system.

From the experiments so far in EUODHILOS, the followings become aware.

(i) Describing the syntax of logical expressions is difficult at first. But, by making good use of the experience, it becomes easier. If the system keeps descriptions for typical logics as a library, the description of a new logic is an easy task even for beginners.

(ii) On a sheet of thought, users are free from deduction errors. On the paper, they may make mistakes in deriving a new formula when deduction rules are applied. The difference is important, because the users have to pay attentions only to the decision how to proceed the proof on the sheet of thought.

(iii) The reasoning assistant system can be used as a tool for CAI. In the system, users can deal with a variety of logics.

The current state is the first step toward the realization of a practical reasoning assistant system. To put the step forward, we have to investigate various subjects including the followings:

- Treatment of relationships between meta and object theories
- Maintaining dependency relations among various theories
- Opening up various new application fields of reasoning
- Improvement and refinement of human-computer interface for the reasoning system

The experiments with different logical systems have shown the potential and usefulness of EUODHILOS in the realm of logics appearing in various fields.

Acknowledgements

The authors are grateful to Dr. T. Kitagawa, the president of IIAS-SIS, and Dr. H. Enomoto, the director of IIAS-SIS, for their ceaseless encouragements. We wish to thank also to Mr. T. Hai for his contribution to our project and to Dr. K. Kobayashi and Mr. Y. Takada for their many valuable comments on the draft of the paper. This work is a part of the major research and development of FGCS project conducted under the program set up by MITI.

References

[Constable 82] R. L. Constable, S. D. Johnson and C. D. Eichenlaub : An Introduction to the PL/CV2 Programming Logics, *LNCS* **135**, Springer–Verlag, 1982.

[Constable 86] R. L. Constable et al. : Implementing Mathematics with the Nuprl Proof Development System, *Prentice–Hall*, 1986.

[de Bruijn 70] N. G. de Bruijn : The Mathematical Language AUTOMATH, its Usage, and some of its Extensions. In M. Laudet, D. Lacombe, L. Nolin and M. Schutzenberger (eds.), *Symposium on Automated Demonstration*. Springer-Verlag, December 1970.

[Gordon 79] M. J. Gordon, A. J. Milner and C. P. Wadsworth : Edinburgh LCF, *LNCS* 78. Springer-Verlag, 1979.

[Goguen 83] J. A. Goguen and R. M. Burstall : Introducing Institutions, *LNCS* 164, Springer-Verlag, 1983.

[Griffin 87] T. G. Griffin : An Environment for Formal Systems, *ECS-LFCS-87-34*, University of Edinburgh, 1987.

[Harper 87] R. Harper, F. Honsell and G. Plotkin : A Framework for Defining Logics. *ECS-LFCS-87-23*. University of Edinburgh, 1987.

[Ketonen 84] J. Ketonen and J. S. Weening : EKL — An Interactive Proof Checker, User's Reference Manual. *Dept. of Computer Science*, Stanford University, 1984.

[Kitagawa 63] T. Kitagawa : The Relativistic Logic of Mutual Specification in Statistics, *Mem. Fac. Sci. Kyushu University*, Ser. **A**, 17.1, 1963.

[Lakatos 76] I. Lakatos : Proofs and Refutations — The Logic of Mathematical Discovery —, J. Worrall and E. Zabar (eds.), Cambridge University Press 1976.

[Langer 25] S. K. Langer : A Set of Postulates for the Logical Structure of Music, *Monist* **39**, 1925.

[Matsumoto 83] Y. Matsumoto, H. Tanaka, H. Hirakawa, H. Miyoshi and H. Yasukawa : BUP : A Bottom-Up Parser Embedded in Prolog, *New Generation Computing* **1**, 1983.

[Peirce 74] C. S. Peirce : Collected Papers of C. S. Peirce, Ch. Hartshorne and P. Weiss (eds.), *Harvard Univ. Press*, 1974.

[Pereira 80] F. C. N. Pereira and D. H. D. Warren : Definite Clause Grammars for Language Analysis — A Survey of the Formalism and a Comparison with Augmented Transition Networks, *Artificial Intelligence* **13**, 1980.

[Smullyan 85] R. Smullyan : To Mock a Mockingbird, and Other Logical Puzzles Including an Amazing Adventure in Combinatory Logic, *Alfred A. Knopf Inc.*, 1985.

[Weyhrauch 80] R. W. Weyhrauch : Prolegomena to a Theory of Mechanized Formal Reasoning, *AI Journal* **13**, 1980.

[Mizoguchi 85] F. Mizoguchi et al. : Prolog and Its Applications 2, *Souken Syuppan*, 1985. (in Japanese)

[ICOT 86] ICOT CAP-WG : The CAP Project (1)-(6), *Proc. 32nd Annual Convention IPS Japan*, 1986. (in Japanese)

[Satoh 86] K. Satoh et al. : Well-Formed Formulas Editor for the Reasoning Assistant System. *Proc. 33rd Annual Convention IPS Japan*. 1986. (in Japanese)

[Minami 86a] T. Minami and H. Sawamura : Proof Constructors for the Reasoning Assistance. *Proc. 33rd Annual Convention IPS Japan*. 1986. (in Japanese)

[Minami 86b] T. Minami and H. Sawamura : A Construction of Computer-Assisted-Reasoning System. *Proc. 3rd Conf. JSSST*, 1986. (in Japanese)

[Tsuchiya 87] K. Tsuchiya et al. : Well-Formed Formulas Editor for the Reasoning Assistant System, *Proc. 35th Annual Convention IPS Japan*. 1987. (in Japanese)

[Minami 87] T. Minami and H. Sawamura : Proof Construction with Working Sheets — A Consideration on the Methodology for Computer Assisted Reasoning —, *Proc. 35th Annual Convention IPS Japan*, 1987. (in Japanese)

[Sawamura 87] H. Sawamura and T. Minami : Conception of General-Purpose Reasoning Assistant System and Its Realization Method, *Proc. WG of Software Foundation IPS Japan (87-SF-22)*, 1987. (in Japanese)

Appendix

In this Appendix several proof examples are exhibited.

First-Order Logic with NK

(A) Smullyan's logical puzzles (originates in combinatory logic) [Smullyan 85]

Axioms:

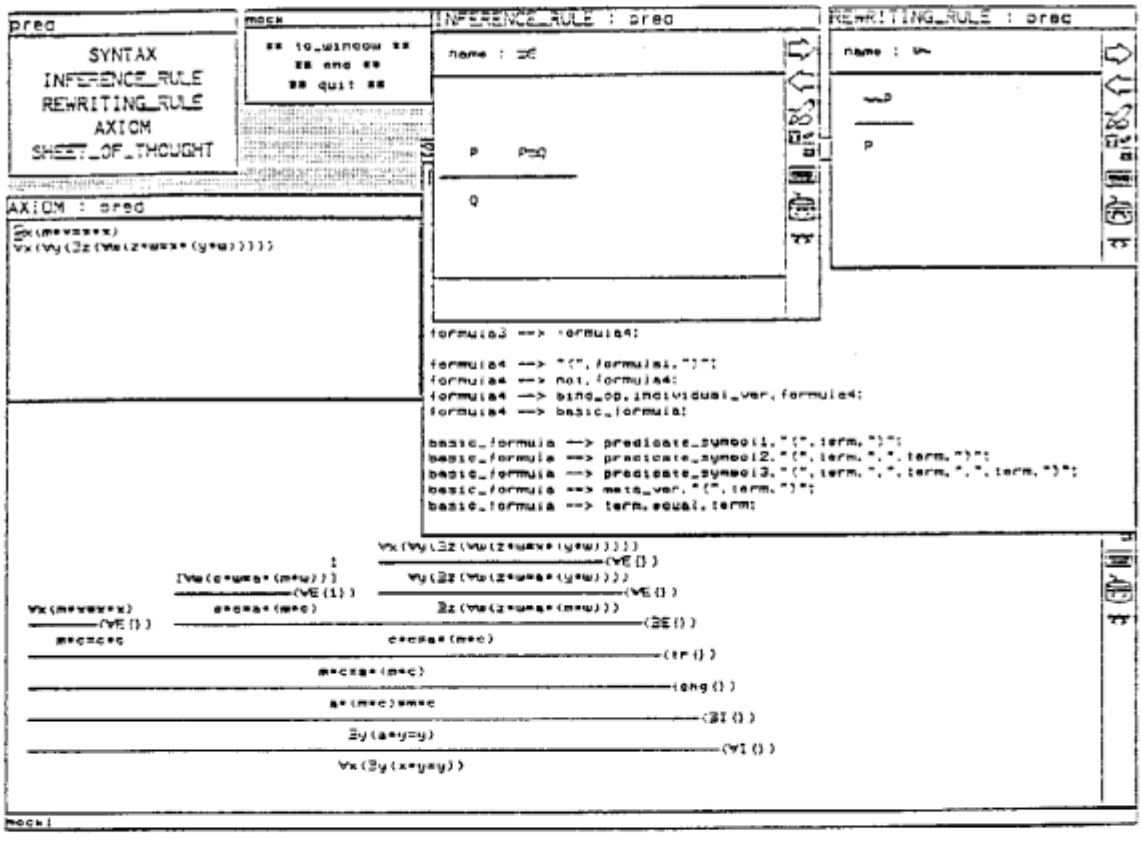
(1) $\forall x\ m \bullet x = x \bullet x$ *(Mockingbird Condition; m is the mocking bird)*

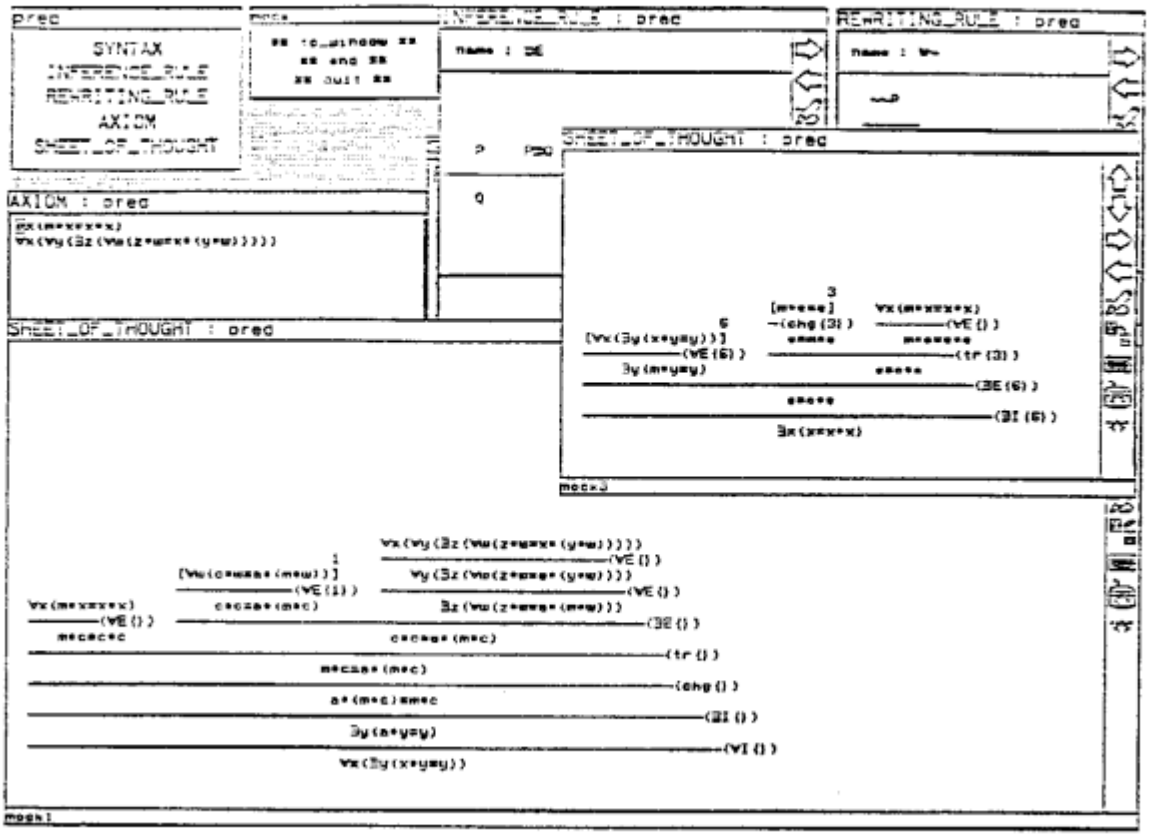
(2) $\forall x \forall y \exists z \forall w\ z \bullet w = x \bullet (y \bullet w)$ *(Composition; z is the composition of x and y.)*

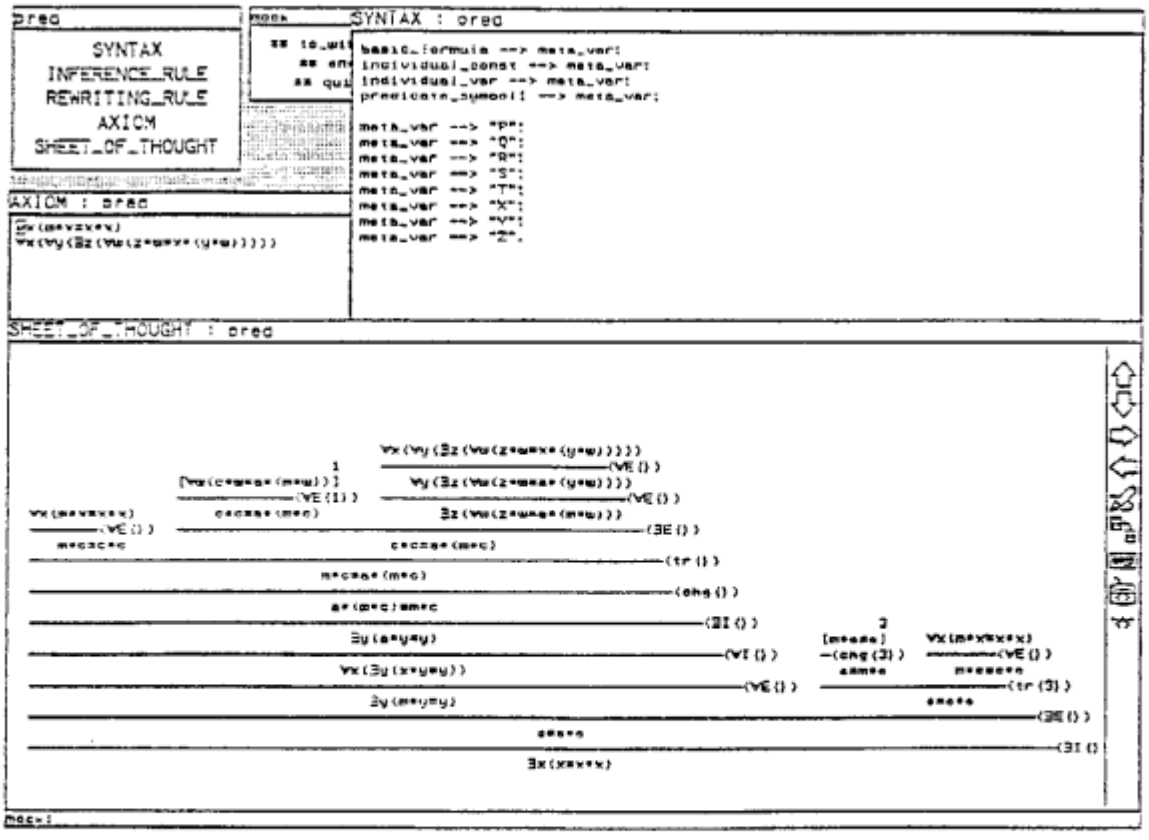
Theorems:

(1) $\forall x \exists y\ (x \bullet y = y)$ *(Every bird of the forest is fond of at least one bird.)*

(2) $\exists x\ (x \bullet x = x)$ *(At least one bird is egocentric or narcissistic.)*




preo
SYNTAX
INFERENCE_RULE
REWRITING_RULE
AXIOM
SHEET_OF_THOUGHT
AXIOM : preo
SHEET_OF_THOUGHT : preo
REWRITING_RULE : preo
SHEET_OF_THOUGHT : preo


preo
SYNTAX
INFERENCE_RULE
REWRITING_RULE
AXIOM
SHEET_OF_THOUGHT
SYNTAX : preo
AXIOM : preo
SHEET_OF_THOUGHT : preo

(B) Unsolvability of the halting problem

Axioms:

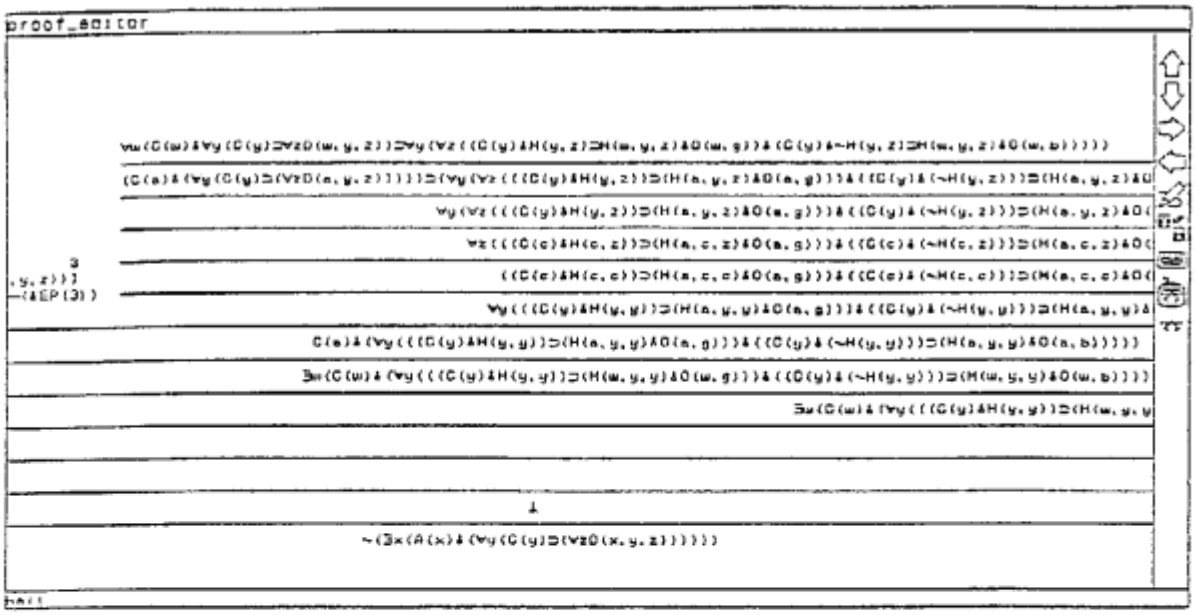
(1) $\exists x\ (A(x)\&\forall y\ (C(y) \supset \forall z\ D(x,y,z))) \supset \exists w\ (C(w)\&\forall y\ (C(y) \supset \forall z\ D(w,y,z)))$
(Church's thesis)

(2) $\forall w\ (C(w)\&\forall y\ (C(y) \supset \forall z\ D(w,y,z)) \supset$
$\forall y\ \forall z\ ((C(y)\&H(y,z) \supset H(w,y,z)\&O(w,g))\&$
$(C(y)\& \sim H(y,z) \supset H(w,y,z)\&O(w,b))))$

(3) $\exists w\ (C(w)\&\forall y\ ((C(y)\&H(y,y) \supset H(w,y,y)\&O(w,g))\&(C(y)\& \sim H(y,y) \supset$
$H(w,y,y)\&O(w,b)))) \supset$
$\exists v\ (C(v)\&\forall y\ ((C(y)\&H(y,y) \supset H(v,y)\&O(v,g))\&$
$(C(y)\& \sim H(y,y) \supset H(v,y)\&O(v,b))))$

(4) $\exists v\ (C(v)\&\forall y\ ((C(y)\&H(y,y) \supset H(v,y)\&O(v,g))\&(C(y)\& \sim H(y,y) \supset$
$H(v,y)\&O(v,b))) \supset$
$\exists u\ (C(u)\&\forall y\ ((C(y)\&H(y,y) \supset \sim H(u,y))\&(C(y)\& \sim H(y,y) \supset H(u,y)\&O(u,b)))$

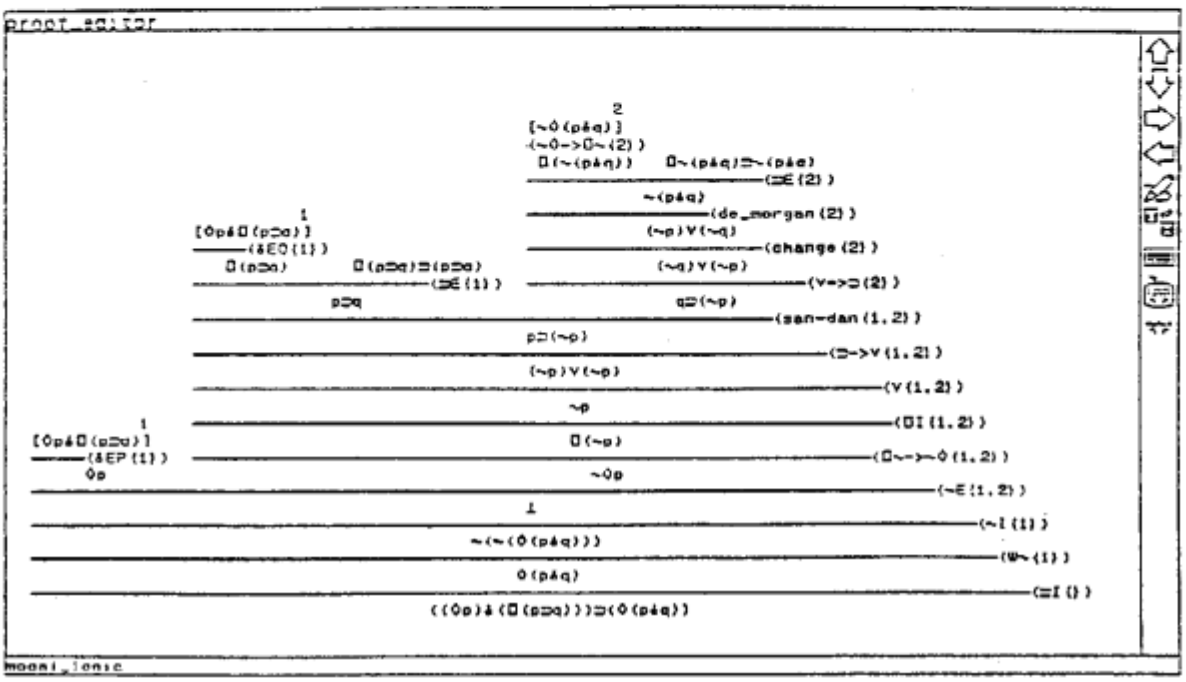
Theorem: *(No algorithm to solve the halting problem exists.)*
$\sim \exists x\ (A(x)\&\forall y\ (C(y) \supset \forall z\ D(x,y,z)))$



Propositional Modal Logic

Theorem: *(A strong correctness assertion is implied from a termination assertion and a weak correctness assertion.)*

$$\Diamond p \& \Box (p \supset q) \supset \Diamond (p \& q)$$



Martin-Löf's Intuitionistic Type Theory

Theorem: *(The law of excluded middle cannot be refuted.)*

$$\sim\sim (P \vee \sim P) \quad (\equiv \ (P \vee (P \supset \bot) \supset \bot) \supset \bot)$$