TR-492

EUODHILOS: A General Approach to Computer Aided Deductive Reasoning

by
T. Minami & H. Sawamura (Fujitsu)

July, 1989

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 Approach to Computer Aided Deductive Reasoning

Toshiro MINAMI and Hajime SAWAMURA

International Institute for Advanced Study of Social Information Science(IIAS-SIS),

FUJITSU LIMITED, 140 Miyamoto, Numazu, Shizuoka 410-03, JAPAN

1. Introduction

This paper presents a new approach to knowledge representation and manipulation expressed in logical forms.

EUODHILOS("*Every universe of discourse has its logical structure.*"[12]) is a general-purpose reasoning assistant system which is used for representing and treating various knowledge represented in logical forms. It is general-purpose, or logic-independent, in the sense that the logics dealt with in the system are defined and given by users. Users can reason, in other words to make proofs for theorems, under the logics which they define.

In these days, a lot of knowledge are written in logical forms in many fields, such as mathematics, computer science, artificial intelligence, and so on. In these fields, various logics such as first-order, higher-order, equational, temporal, modal, intuitionistic, and type theoretic logics are used. The importance of assisting these reasoning activities by computers is increasing day by day.

EUODHILOS takes a new approach to this purpose. It aims at increasing the efficiency and the accuracy of the *human* reasoning process of representing and manipulating knowledge. It regards the reasoning processes as those 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 first phase begins when one becomes aware that some mental images of objects and concepts have some structures and wants to clarify them formally. To clarify the mental images, one has to describe them. A formal framework for the description is called a "logic" in this paper, and a logical description of knowledge about the object is called a "logical model," and its manipulation is called a "reasoning." In the second phase, one makes logical models. At first, one has to determine the syntax of the logical expressions. The logical structure can be described by axioms and derivation rules such as inference and rewriting rules. In the

third phase, one investigates the logical model, and proves its formal properties. 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 it. In this case, one has to modify part 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.

Two major subjects have to be pursued to realize such a system. The first one is the "generality" of the system. As S. K. Langer said, 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 the knowledge and discussing about it. In order to assist human reasoning for various objects, the system must have the ability to assist the users for describing a variety of logical structures and for manipulating the expressions in these logics. The system EUODHILOS is named as an acronym of the phrase by Langer to emphasize the importance of the generality of the system.

The other subject is the investigation of "reasoning-oriented human-computer interface." The fundamental recognition in this subject is that the (mathematical) reasoning proceeds through "Proofs and Refutations" (by Lakatos[11]). In order to make the system helpful for one to conceive ideas in reasoning, it must have a good interface for reasoning so that one can easily reason by trial and error.

2. EUODHILOS

EUODHILOS is a prototype of the general-purpose reasoning assistant system. We intend to clarify the concept of the ideal system image through developing and using it. It is designed by considering the following issues:

(i) Realization methodology of a general-purpose reasoning assistant system, based on the philosophy of Langer[12].

(ii) Provision of environment for experimenting logical model construction based on the philosophy of Lakatos[11].

Figure 2.1 is an illustration of how the reasoning assistant system EUODHILOS is used. In the upper half of the figure which corresponds to the feature (i) above, the user specifies constituents of a logic, such as symbols, syntax of expressions including formulas, deduction

rules. In the lower half, which corresponds to (ii), the user tries to construct proofs of theorems under the logic defined in the previous step. In EUODHILOS, 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, 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 to the library of theorems so that they can be reused as a starting formula in the later proofs for other theorems.

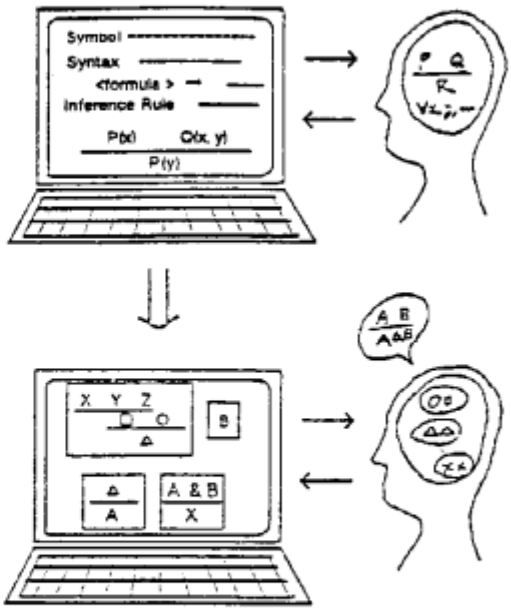

Figure 2.1 Using EUODHILOS

2.1 Language Description in EUODHILOS

In EUODHILOS, as noted above, knowledge about objects is represented as logical forms, and the language system to be used is designed and defined by the user at the beginning. The language system consists of the definitions of the syntax for the logical expressions. The syntax of the expressions is given by using definite clause grammar (DCG)[15] formalism in the current version. DCG is an extension of context free grammar formalism so that context sensitive syntactic constraint can be expressed. Therefore we get expressive power as well as simplicity of expressions. From the description, a bottom-up parser based on BUP[13] is automatically generated and used for parsing the inputs given in string forms.

The system automatically generates not only the parser, but also the 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. Since the parser and the unparser are generated automatically from the single descriptions of syntax, the soundness of relationship between these functions is guaranteed.

When an expression is entered, the parser is invoked to check its validation. At the same time the internal structures of the expression in the language are constructed as well. Owing

to this function, one can omit the internal structures of expressions in the syntax definitions in EUODHILOS. 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 translating into the external ones by the unparser.

Although EUODHILOS is developed for proving practical and sophisticated theorems, we present here a very simple example in Figure 2.2 for brevity. The example is a description of the logic for a puzzle of mocking bird by Smullyan[17], which is an interpretation for combinatory logic. From the definition, expressions such as "M•x=x•x" and "(A*B)•x=A•(B•x)" are formulas of this logic. Meta symbols are used in the definitions of axioms, inference rules, and rewriting rules and also in a schematic proof on a sheet of thought.

Syntax description:

```
formula→term, "=", term
term→b_term
term→b_term, "•", b_term
b_term→variable_symbol
b_term→constant_symbol
b_term→"(", b_term, "*", b_term, ")"
b_term→"(", term, ")"
```

Symbol declaration:

```
variable_symbol:  "A"-"E"
variable_symbol:  "s"-"z"
constant_symbol:  "I"-"N"
```

Meta symbol declaration:

```
formula:  "F"-"H"
term:  "Y"-"Z"
elementary_term:  "X"
```

Figure 2.2 A description of a logic

2.2 Axiom and Derivation Rule Description in EUODHILOS

A derivation system in EUODHILOS consists of axioms and derivation rules. Derivation rules consist of inference and rewriting rules. A finite set of formulas is given as the axioms. Inference rules are given in a natural deduction like style presentation by the user. An inference rule consists of three parts; the first one 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 its assumption, it indicates that the premise is obtained under the assumption, and otherwise it is obtained without condition. 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.

Figure 2.3 is the definitions of axioms and inference rules for the logic of mocking bird. In the definitions of inference rules, the expressions '`[X]`' and '`[Y]`' indicate the occurrences of the expressions '`X`' and '`Y`' respectively.

Axioms:

`M•x=x•x` *Existence of the mocking bird.*
`(A*B)•x=A•(B•x)` *Composition.*

Inference rules:

$\frac{\texttt{F[X]}}{\texttt{F[Y]}}$ (substitution) $\frac{\texttt{F[Y]} \quad \texttt{Y=Z}}{\texttt{F[Z]}}$ (equality)

Figure 2.3 Axioms and inference rules for the logic of mocking bird

Considering the fact that mathematicians use rewriting rules so much as inference rules, we decided to add rewriting rules as a kind of derivations. Rewriting rules are presented in the following scheme:

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

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

Iterating the applications of the derivation rules, one can obtain a derivation tree.

2.3 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 analogous to the concept of sheet of assertion due to C. S. Peirce[14]. It allows one to draft a proof, to compose proof fragments, detach a proof, to reason by using lemmas, and so on.

On a sheet of thought, proof fragments (or partially constructed proofs) are the elementary units for manipulation. Proof fragments are newly created as assumptions, axioms, or theorems of the theory. An assumption is embraced by square brackets. Proof fragments composed, separated, and deleted according to the operations given by the user. Applications of inference and rewriting rules are expressed visually as those represented on the paper. This naturally induces that the appearance of a derivation tree on the sheet is also the same as that on the paper. In this way, proofs are expressed visually and recognized naturally.

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[11], Kitagawa's relativistic logic of mutual specification[10], etc.).

As an example of deduction process on a sheet, we will take a very simple one to illustrate how one can proceed deduction. The example proof is done under the logic of mocking bird cited before. The problem to be solved is the statement: 'Any bird is fond of some bird.' That is, for any bird '`A`' there exists a bird '`X`' such that "`A•X=X`" holds. At first, one may enter two axioms "`M•x=x•x`" and "`(A*B)•x=A•(B•x)`" on a sheet. To deduce some formula, he may deduce "`M•A=A•A`" from the axiom "`M•x=x•x`" by substituting '`A`' to the variable '`x`.' He cannot proceed any more in this case, so he tries other substitution. Next, he may substitute '`A*B`' to '`x`'. In this case, he gets "`M•(A*B)=(A*B)•(A*B)`" and "`(A*B)•(A*B)=A•(B•(A*B))`." After

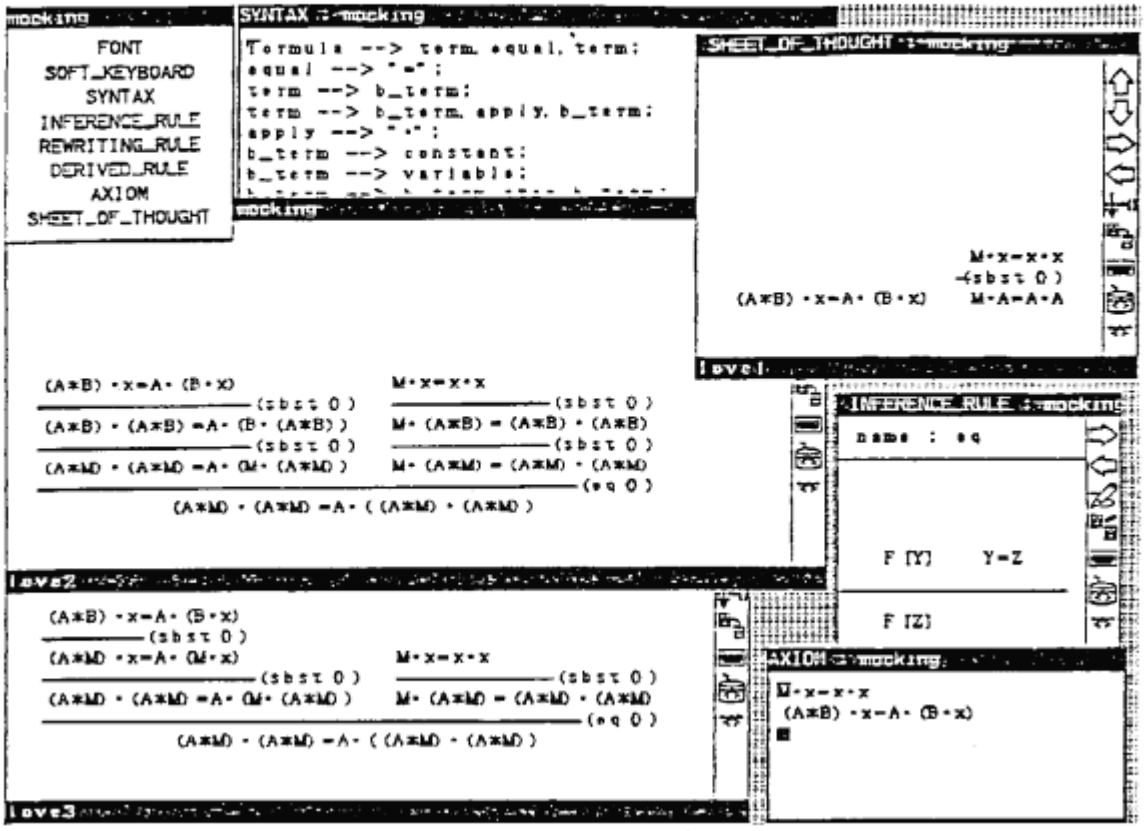

Figure 2.4 An Example of Proof Construction on the Sheet of Thought

looking these, he makes aware that by substituting 'M' to 'B' he can get the desired formula "(A*M)•(A*M)=A•((A*M)•(A*M))." This indicates that a bird 'A' is fond of the bird denoted by the expression '(A*M)•(A*M)." If he re-reads his proof carefully, he may become aware that the proof is redundant, and he can get the final proof of the theorem; By substituting 'A*M' to 'x' and 'M' to 'B', and by the inference rule of equality, one can get the desired formula. Proofs on the sheet of thought proceed like this. Figure 2.4 is the actual screen image of sheet of thoughts for this example. More practical and sophisticated example will be presented in the full paper.

3. Related Systems

Aside from the reasoning assistant system (RAS for short), we consider the following three types of the systems which can be used for assisting human reasoning:

(1) automated theorem prover,

(2) proof checker, and

(3) proof constructor.

As the most significant features of RAS, we can state (i) that it is logic free, and (ii) that it supports the interactive and visual proof constructions.

An (automated) theorem prover is a system which searches a proof of a formula given by the user. In a RAS, proofs are searched and found through the interactions between the system and the user. This is the major difference between a theorem prover and a RAS.

A proof checker is a system which checks the correctness of a proof described by the user. In a proof checker, the user has a putative proof of a theorem from the beginning. 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 and gives it to the checker. The system begins to check the correctness of the proof. If the checker finds errors in the proof, it shows them to the user. 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, the system such as RAS which assists to construct a proof is more suited than the proof checker.

Many proof checkers have been developed up to now. AUTOMATH[1] is a proof checker in which the user can specify how the proofs are constructed. PL/CV2[2] is used for proving the correctness of PL/I like programs. CAP-LA[8] deals with the proofs on linear algebra.

A proof constructor(e.g. LCF[5], FOL[18],IPE[16], EKL[9] and Nuprl[3]) 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.

EUODHILOS is a kind of proof constructors. The most significant difference of it from other proof constructors is that in EUODHILOS underlying logic can be defined in the system, while in others, eventhough some of them have the function of extending its syntax, the logic is basically fixed. There are merits and demerits for fixing the underlying logic. 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, and EUODHILOS is the best choice.

4. Concluding Remarks

The first version of EUODHILOS is now available and the next version is under design by reflecting the experience of using the current one. 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. We also had much proof experiments and experience on these logics. Some logic such as tableau method seems impossible to be treated in the current framework. We plan to extend the framework so that logics given in other formulations can be treated in the system.

The current state is the first step towards the realization of an ideal 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, and
- Improvement and refinement of human-computer interface for the reasoning system.

From the experiments so far in EUODHILOS, we are convinced of the followings:

(i) Describing the syntax of logical expressions is difficult at first. But, after defining several logics, we can define a new logic in a few hours. If the system keeps descriptions for typical logics as a library, the description of a new logic would be 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.

By using the general-purpose reasoning assistant system EUODHILOS, we can treat various kinds of knowledge represented in logical forms in a uniform way, and we can investigate the relationships between them. This will develop a new research field of reasoning assistance as well as knowledge manipulation by computers.

References

[1] N.G.de Bruijn: The Mathematical Language AUTOMATH, its Usage, and some of its Extensions, In M. Laudet et al. (eds.), *Sympoosium on Automated Demonstration*, Springer-Verlag, pp.29–61, 1970.

[2] R.L.Constable et al.: An Introduction to the PL/CV2 Programming Logics, *LNCS* 135, Springer-Verlag, 1982.

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

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

[5] M.J.Gordon et al.: Edinburgh LCF,*LNCS* 78, Springer-Verlag, pp.221–270, 1979.

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

[7] R.Harper et al.: A Framework for Defining Logics, *ECS-LFCS-87-23*, Univ. of Edinburgh, 1987.

[8] ICOT: The CAP Project (1)–(6), *Proc. 32nd Annual Conv. IPS Japan*, 1986. (in Japanese)

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

[10] T.Kitagawa: The Relativistic Logic of Mutual Specification in Statistics, *Mem. Fac. Sci. Kyushu Univ.*, Ser. A, 17, 1, 1963.

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

[12] S.K.Langer: A Set of Postulates for the Logical Structure of Music, *Monist* 39, pp.561–570, 1925.

[13] Y.Matsumoto et al.: BUP:A Bottom-Up Parser Embedded in Prolog, *New Generation Computing* 1, pp.145–158, 1983.

[14] C.S.Peirce: Collected Papers of C.S.Peirce, Ch.Hartshorne et al. (eds.), *Harvard Univ. Press*, 1974.

[15] F.C.N.Pereira et al.: Definite Clause Grammars for Language Analysis—A Survey of the Formalism and a Comparison with Augmented Transition Networks, *AI Journal* 13, pp.231–278, 1980.

[16] B. Ritchie & P. Taylor: The Interactive Proof Editor An Experiment in Interactive Theorem, *ECS-LFCS-88-61*, Univ. of Edinburgh, 1988.

[17] R.Smullyan: To Mock a Mockingbird, *Alfred A. Knopf Inc.*, 1985.

[18] R.W.Weyhrauch: Prolegomena to a Theory of Mechanized Formal Reasoning, *AI Journal* 13, pp.133-179, 1980.