ICOT Technical Memorandum: TM-0810

TM-0810

EUODHILOS: Interactive Reasoning System for a Variety of Logics

by
T. Minami & H. Sawamura (Fujitsu)

October, 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: Interactive Reasoning System for a Variety of Logics

Toshiro Minami and Hajime Sawamura

International Institute for Advanced Study of Social Information Science(IIAS-SIS),
FUJITSU Ltd., 140 Miyamoto, Numazu 410-03, JAPAN

E-mail: minami%iias.fujitsu.co.jp@uunet.uu.net

Abstract

EUODHILOS is the reasoning assistant system with two characteristic features. One is its "naturality," which indicates that users can see and manipulate proof fragments interactively in a natural style as is used on paper with pencil when they reason on sheets of thought which is a proof supporting environment in EUODHILOS. The other one is its "generality," which means that EUODHILOS is logic-independent in the sense the language and logic to be used are defined by its user. We will see the overview of the system and comparisons to related works.

1 Introduction

Logical reasoning plays important roles in many fields like mathematics, computer science, artificial intelligence. Various logics such as first-order, higher-order, equational, temporal, modal, intuitionistic, and type theoretic logics are used there[9, 18]. Just as S. K. Langer states, we recognize that "Every universe of discourse has its logical structure"[11]. So reasonings in a variety logics will increase and become more and more important in future. As computers become popular in these days, it is natural to think that computer assisted reasoning will be the standard style of formal reasoning in the future. Under these considerations, we have been making research of general-purpose reasoning assistant sytem named EUODHILOS.

We put two major subjects to be pursued for approaching to this theme. The first subject is the investigation of "reasoning-oriented human-computer interface." The fundamental recognition in this subject is that the reasoning essentially proceeds via trial and error. A system is helpful for one to conceive ideas in reasoning if it has a good interface so that one can reason in a natural style.

The other one is the "generality" in reasoning, which corresponds to the requirement that the system must be independent from any specific logics. We think that for each object we mention, there must be a logic suited for expressing and discussion about it. So it is preferable that reasoning on it must be done by using the most suitable logic. The system's name EUODHILOS is an acronym of the Langer's statement. It is taken in order to emphasize the generality of the system.

To summarize, we may say that the characteristic feature of EUODHILOS is on its "naturality." We can use the logic natural to the object domain so that objects in the domain can be described by using natural expressions. The reasoning style on the sheets of thought is natural in the sense the system does not limit the style of reasoning. User can put assumptions, derive upwards or downward, connect two proof fragments, and do other operations in any order he likes. Throughout the system it is the user who makes reasoning, and the system is used for assisting the user in order to boost the reasoning ability of the user.

In order to realize such a system EUODHILOS must have the following functions:

(i) It has to assist the user to define the underlying logic for reasoning.

(ii) Since it is used interactively, it has to assist proof constructions in various styles of reasoning.

2 Overview of EUODHILOS

The current version of EUODHILOS is developed as an experimental system of general-purpose reasoning assistant system. We intend to clarify the image of the ideal reasoning system by using it. Figure 1 is an illustration of how the system is organized.

It includes a function assisting for defining the logic to be used in reasoning. A logic in EUODHILOS consists of language and derivation systems. We will explain what they are like in the following two subsections.

In EUODHILOS, partially constructed proofs, which are called proof fragments, may appear on one or more windows called sheets of thought. A sheet of thought is an environment for creating theorems and their proofs. Manipulations on proof fragments such

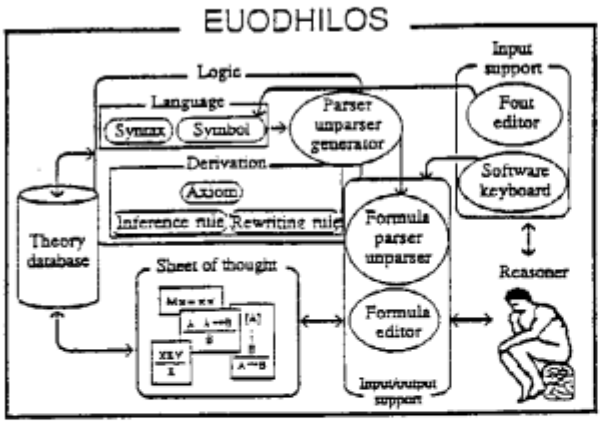

Figure 1: Configuration of EUODHILOS

as creation, deletion, derivation, connection, separation, and so forth are possible by actions to the system mainly by using mouse. The theorems constructed 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. In this way, users can cumulatively construct many theorems in the theory. Software-keyboard and formula editor are provided to facilitate input/output.

2.1 Language Description

The language system of a logic, which defines what strings of symbols are thought to be valid logical expressions, is designed and defined by the user at first. Characters used in logical expressions are defined by using the standard font editor. The syntax of the expressions is given in definite clause grammar (DCG)[15]-based formalism. From the descriptions, with operator declarations, a BUP[12] parser and an unparser for the defined language are automatically generated.

We have to say here that the string expressions themselves do not denote their "meaning". For example, a string expression "$\forall x.\varphi$" consists of four symbols, i.e. it has four components, while we see it as an universal-quantification expression with two components "x" and "φ". There is no interpretation corresponding to ".". The differences between "$(a+b)$" , "$(a)+(b)$", and "$a+b$" are another examples. They expressions are different but what they denote are the same. Therefore internal expressions should be used for manipulation in the system.

Following to the conventional method of DCG, we can describe the correspondence between external and internal expressions by using an argument for non-terminals. For the examples above, we may describe like:

```
formula('∀'(V,F)) --> "∀", variable(V),
                      ".", formula(F)
e_formula(F) --> "(", formula(F), ")"
```

Since describing the internal expressions is usually a boring task, EUODHILOS automatically generates the DCG expressions from the given descriptions. In order to automate this step, operator declarations are required of the user. These declarations indicate which elements of the syntax descriptions are used as operators in the internal expressions.

As the result, the parser not only checks the validity of given external expressions, it also translates them to the corresponding internal expressions. The unparser translates inversely.

2.2 Axiom and Derivation Rule Description

A derivation system in EUODHILOS consists of axioms and derivation rules. EUODHILOS allows rewriting rules as well as inference rules as derivation rules. This comes from our observation on how human reasoning proceeds. For example, we often use a chains of formulas like $a_1 = a_2 = \cdots = a_n$, each a_{i+1} is obtained by rewriting a_i. Axioms are given as a list of formulas in the axiom definition window. Inference rules are given in a natural deduction style, which meaans that an inference rule consists of three parts; the first one is the premises of the rule, each of which may have an assumption, the second is the conclusion of the rule, and finally the third is the optional restriction that is imposed on the derivations of the premises, such as variable occurrence conditions.

Schematically, an inference rule is given in the following form in which each of the assumption parts is optional:

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

If a premise has its assumption, it indicates that the premise is obtained under the assumption, and otherwise that it is obtained by some way. An inference rule can be applied if all the premises are obtained in this manner, and the restrictive condition, if given, is satisfied. If it is applied, the conclusion is obtained as the result of the derivation.

A rewriting rule is given in the form:

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

A rewriting rule 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.

Well-known typical styles of logics such as Hilbert's, Gentzen's, equational can be treated within this framework. Hilbert's formulation is expressed by using axioms, modus ponens as inference rules, and no rewriting rules are given, while Gentzen's formulation consists only on inference rules, and equational formulation can be given by axioms and rewrting rules.

One can obtain a derivation tree by iterating the applications of the derivation rules. To reduce the similar applications of several rules appearing in many places, one can define the derivation rules. Once defined, a derivation rule can be used just like a primitive one. We can say from our experience that derived rules are much more useful in proving new results than the results expressed by formulas (i.e. theorems).

2.3 Proof Construction

The proof constructing environment provided by EUODHILOS is called "sheet of thought". Maybe we may also call it a proof editor, but we call it other way because we want to regard it more generally as an environment of reasoning with intelligent assisting functions. We can draft a proof, to connect proof fragments (i.e. partially constructed proofs), separate a proof, to reason by using lemmas (or theorems), and so on under the support of sheets of thought.

As the design principles of sheet of thought, we take:

1. Theorems and proofs are found and constructed under the user's initiative through the process which would proceed basically via trial and error.

2. Reasoning during proof constructions can be done along with the natural way of thinking of human reasoners.

When one wants to reason, one may make a candidate of theorem or derive some trivial results to see what the logic is like. Both styles of reasoning are allowed on the sheets. At first, the user opens a window of sheet of thought, and then he may enter a goal to be proved or may enter several assumptions which seem to be used to derive some results. Entering one or some of the axioms is also possible. He can apply the derivation rules to these formulas and obtain the resultants. He can also combine some resultants to make more complicated results. In this way the partially constructed proofs, which we call proof fragments, are obtained and become bigger and bigger, and finally a complete proof fragment, which is a proof of a theorem, is obtained. We can store the theorem to the theory database so that it can be used in the later reasoning.

We can define derived rules so that we do not have to apply the same sequence of rules at several places. Inference, rewriting and derived rules are applied and expressed schematically just like those which we write on the paper.

EUODHILOS supports the typical styles for reasoning, that is, forward (or top-down) reasoning, backward (or bottom-up) reasoning, and reasoning in a mixture of them. Most of other systems allows only one style of reasoning. For example, the underlying logic of Nuprl[3] is version "nu" of Proof Refinement Logics. It is a logic for top-down reasoning. Proofs in EKL[10] is constructed bottom-up. From our observation of human reasoning, both reasoning styles is used in reasoning. So for our purpose, both types of reasoning must be provided.

3 Example

In order to get the image of reasoning on EUODHILOS more intuitively, we show an example in this section. We take a mocking bird puzzle by R. Smullyan[17] in which combinatory logic is treated with the interpretation into a forest of birds.

```
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, ")";

variable_symbol→ "A"|"B"|"x";
constant_symbol→ "M";

meta_formula→ "F";
meta_formula→ meta_pred, "[", term, "]";
meta_pred → meta_formula;
meta_variable → "X";
meta_term→ "Y"|"Z";
b_term→ meta_term.

operator
  "*";   "•";   "=";
predicate
  meta_pred.
```

Figure 2: A description of the language for the puzzle of mocking birds

Figure 2 is an example description of the language. From the definition, we can see that expressions such as "`M•x=x•x`" and "`(A*B)•x=A•(B•x)`" are formulas of this logic.

Figure 3 is the definitions of axioms and inference rules. In the definition of inference rule named substitution, the expressions '`[X]`' and '`[Y]`' indicate the occurrences of the expressions of variable '`X`' and term '`Y`' in a formula '`F`' respectively.

As an example of derivation process on a sheet, we will illustrate in Figure 4 how one can proceed derivations in the example of the mocking bird. The problem is to prove the statement: "Any bird is fond of some

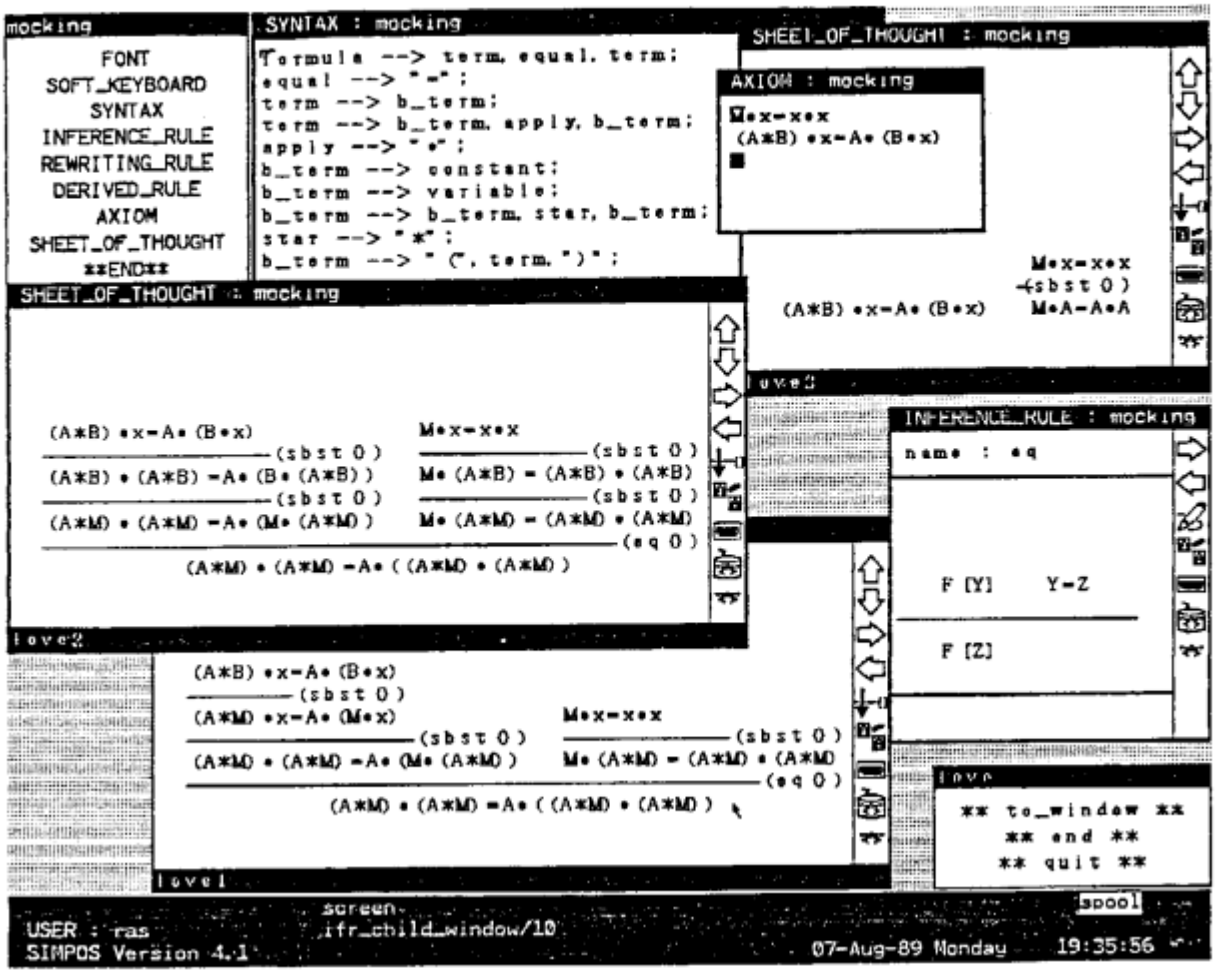

Figure 4: Proof Construction on the Sheet of Thought

Axioms:

`M•x=x•x` *Existence of the mocking bird.*

`(A*B)•x=A•(B•x)` *Composition.*

Inference rules:

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

Figure 3: Axioms and inference rules for the logic of mocking birds

bird." That is, for any bird 'A' there exists a bird 'X' such that "A•X=X" holds.

At first, in a sheet of thought at the top-right corner of the figure, one enters 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, at the middle-left sheet, 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 looking these, he makes aware that by substituting 'M' to 'B' he gets 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. (It is shown on the bottom-left sheet.) Proofs on the sheet of thought proceed like this. To see the more practical examples, refer to [13] and other papers in it.

4 Related Works

When we think about what the automated reasoning is like, probably most popular idea would be fully automatic theorem prover(ATP). An ATP(such as BMTP[1]) is a system which searches a proof of a formula given by the user. In a RAS(Rasoning Assistant System), on the other hand, proofs are searched and found through the interactions between the system and the user. The initiative is taken by the user.

This is the major difference between the features of ATP and RAS, which may come from the difference of the purposes of the research of them. The purpose of the research of ATP may be originated to the interest how the intelligent activities of human can be simulated by machines. The purpose of RAS, and probably of other types of interactive reasoning systems, is to find what is the ideal way of reasoning under the cooperation between human and machine.

With all the difference, they do not confront each other. At least from the view of interactive system, the outcome of ATP may be used to make the system more sophisticated and intelligent. For example, ATP can be used for filling the small gaps between formulas in proofs.

The second style of automated reasoning would be that of proof checker. We call a system proof checker if it's main purpose is to verify the correctness of a proof described by the user. Suppose there is a putative proof of some theorem obtained by a human. A human proof may contain some careless mistakes including small gaps in a proof. He may say "trivial" which is in fact not trivial but just tedious. This is a typical situation when a proof checker is used. The checker provides a language for describing human proofs. The user describes his or her proof by this language and gives it to the system. The system checks the correctness of the proof. If the checker finds errors in the proof, it shows them to the user. The situation of proof constructors and RAS are little bit different. In this situation, there is no proof description at the beginning. The purpose of the systems is to assist their user to create validated proofs easily.

In fact, it is not so easy to classify the purpose or the situation of the system clearly, because most of the systems would not be used in the typical styles cited above. AUTOMATH [2] is a proof checker in which the user specifies the construction of proofs. CAP-LA [8] checks the proofs in linear algebra. The systems such as LCF [5, 14], FOL [19], EKL [10], and Nupri [3] are proof constructors.

EUODHILOS may be called a kind of proof constructors in some point of view. The major difference of RAS (such as EUODHILOS) and the ordinary proof constructors lies in generality and the form of proof fragments. Maybe this difference also comes from that of the purpose of the user of the system. Most of the proof constructors seem to have an intention that they are used mainly for verifications. The most important point for this purpose would be to find the specific feature to make reasoning for the underlying logic which is fixed to the system. On the other hand the purpose of RAS is to assist as many cases of human reasoning as possible. So the fixation of the logics to be treated in the system becomes a restrictions to the domain of reasonings.

In order to realize naturality we designed the system so that it can be used just like an intelligent "paper". We think the expressions written on the paper would be the most suitable one for human reasoners. From this thought, we think a feature that facilitate to use a new symbol is necessary, and the we decided to take the natural deduction style of formulation frame for the fundamental reasoning style in the system. It is possible to say that the generality of the system is also an example of naturality. From our thought expressed in the Langer's statement, it is natural to think that the RAS system has to deal with logics in domains which me intend to reason about.

5 Concluding Remarks

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

From the experiments so far in EUODHILOS, though it is still in its ealiest step, 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 may be quite easy even for beginners.

(ii) On sheets 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 decisions how to proceed the proof on the sheet of thought.

(iii) The reasoning assistant system can be used as a tool for CAI(Computer Assisted Instruction) in logics and logical reasonings. The students can deal with a variety of logics in a single system.

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

(1) Extending the framework of logic description so that logics given in other formulations can be treated in the system as well;

(2) Adding the facility of maintaining dependency relations among various theories;

(3) Treatment of relationships between meta and object theories so that reflective proof like FOL[19] is available;

(4) Improvement and refinement of reasoning-oriented human-computer interface so that it can make natural assistance;

(5) Opening up various new application fields of computer assisted reasoning such as verification, programming, education, etc.

We believe the research on reasoning assistant system EUODHILOS will eventually open up a new dimension of automated reasoning for practical use.

Acknowledgements

The authors are indebted to our colleagues Kaoru Yokota and Kyoko Ohashi in system implementation. This is a part of the major research and development of the FGCS project conducted under the program set up by MITI.

References

[1] R.S. Boyer & J S. Moore: A Computational Logic Handbook, Academic Press, 1988.

[2] 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.

[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, F. Honsell & G. Plotkin: 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] P. Jackson et al. (eds): Logic-Based Knowledge Representation, The MIT Press, 1989.

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

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

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

[13] T. Minami et al.: EUODHILOS: A General-Purpose Reasoning Assistant System – Concept and Implementation –, IIAS-SIS Research Report No. 84, *FUJITSU LIMITED*, 1988.

[14] L.C. Paulson: Logic and Computation, Interactive Proof with Cambridge LCF, Cambridge Univ. Press, 1987.

[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, University of Edinburgh, July 1988.

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

[18] R. Turner: Logics for Artificial Intelligence, Ellis Horwood Limited, 1984.

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