汎用論証支援システム EUODHILOS の応用と評価

沢村 一[†] 南 俊朗[†] 大谷 武[†]

問題領域に適した論理系の定義ができ，定義された論理系での論証が行える汎用の論証支援システム EUODHILOS が開発された．このシステムは次の三つの特徴をもつ：(1)記述性が高く，扱いやすい論理記述のための枠組み，(2)強力で柔軟性のある証明構築機能，(3)使いやすく，論証に適したインタフェース．本論文では，まず EUODHILOS の特徴的な機能を簡単に紹介する．次に，計算機科学，人工知能などの分野で考案され使われてきたさまざまな論理系を用いて，EUODHILOS の論理定義法，および定義された論理系での証明例を示し，EUODHILOS の使用経験に基づく評価を与える．

Application and Evaluation of General-Purpose Reasoning Assistant System EUODHILOS

Hajime Sawamura,[†] Toshiro Minami[†] and Takeshi Ohtani[†]

We have developed a general-purpose reasoning assistant system EUODHILOS that allows a user to define his or her own logical system relevant for the intended problem domain and to reason about it. This has three basic features: (1) an expressive and tractable framework for representing a logic, (2) a powerful and flexible proof construction facility, and (3) a visual reasoning-oriented human-computer interface for ease of use. In the paper, some unique functions of EUODHILOS are briefly described. Then, an evaluation of EUODHILOS based on experiences of its use is provided, showing the logic representation methods and proof methods in the defined logics. In doing so, we take up various kinds of logics which have been devised or used in computer science, artificial intelligence and so on.

1. はじめに

対象を論理学的に認識，表現し，対象の論理操作によって問題解決を計るといった論理学的方法論は，今日，計算機科学や人工知能の研究における一つの有力なパラダイムとなっている[37]．これまでこのような論理学的方法論を支援するために，論理系が固定されている特定目的のための証明支援システムについて多くの研究がなされてきた[11),38),3),15),6)]．

このような研究動向とは対照的に，我々はそれぞれの問題領域ごとにより適した論理系を定義することを許し，更に定義された論理系の下で論証による問題解決を可能にする汎用論証支援システム EUODHILOS を提案してきた[28),30),31)]．別の言い方をするならば，これは論理に依存しない証明エディタであるといえる．

EUODHILOS は，すべての対象領域はそれ固有の論理系をもつという認識に基づき，論理系が固定されている証明支援システムの場合に起こる次の問題を克服することを目的としてきた．

(1)論理系を固定しているシステムは，他の論理系を必要とする対象領域には一般には使えない．使えたとしても，対象世界をそのシステムが扱える表現に変換しなければならない．そのさい，その表現は元の表現より不自然であったり長く複雑になることが多い．

(2)各対象領域ごとにその都度論証システムを構築することになれば，そのために多大の労力を必要とするのみならず，類似の作業を繰り返し行うことになり無駄も労力も多くなる．

実際これまで，EUODHILOS の汎用性によってさまざまな論理系を扱うことができた[32)]．そして汎用性をもつ証明支援システムの方が，特定論理用のシステムより，論理による問題解決への利用可能性，および応用の範囲という点ではるかに有利であることが分かった（論証システムがもつべき汎用性に関するその他の議論については文献28),30),31)を見られたい）．

最近，世界的にもこのような汎用の論証支援システ

† (株)富士通研究所 国際情報社会科学研究所
International Institute for Advanced Study of Social Information Science (IIAS), Fujitsu Laboratories Ltd.

ム，あるいは証明構築支援環境の必要性・意義が筆者らとは独立に他の研究者らによっても認識されるようになってきた[12),24),8),7),16),2)]．しかしながら，本論文ではEUODHILOSとこれら他のシステムとを詳細に比較検討することはせず，本文の関連する所でのみ言及することにする．

以下，第2章でEUODHILOSのアプローチと特徴的な機能，特に論理定義法，証明構成法の概観を与える．第3章でEUODHILOSをさまざまな論理系へ適用し，第4章でそれを踏まえた我々のアプローチの評価を与える．

2. EUODHILOSの特徴

この章では，我々が取った汎用論証支援システムへのアプローチを簡単に述べ，EUODHILOSの代表的な諸機能を概観する．したがって，ここではEUODHILOSの機能を網羅的に述べることは意図していない．より詳しくはEUODHILOSマニュアルを参照されたい[21)]．

2.1 EUODHILOSのアプローチ

汎用性をもつ論証支援システムを構築するには，さまざまな論理に共通するもの，および各論理がもつ固有の特異性という相反する側面を捉えるための枠組みが必要になる．しかしながら，これに正面から答えるには，「そもそも論理とは何か？」という大変基本的な問題に行き当たってしまうことになる．そこで，EUODHILOSでは，これまでよく知られた論理に共通する基本的な特徴を集め論理記述の枠組みを考えるという実用的なアプローチを取ることにした．この意味で，EUODHILOSのアプローチは経験的なアプローチであるとも言え，LF[12)]やIsabelle[24)]のように論理を他の形式的体系にコーディングする方法とは全く異なっている[34)]．

2.2 EUODHILOSの諸機能

EUODHILOSを設計するに当たっては，次の三つのポイントが重視された：

(1)記述性が高く，扱いやすい論理記述のための枠組み，

(2)強力で柔軟性のある証明構築支援機能，

(3)使いやすく，論証に適したインタフェース．

(1)と(2)は汎用システムにとっては本質的な部分である．(3)はこれまでの定理証明機や論証支援システムでは軽視されてきたが，対話型の論証支援システムの証明構成機能とも絡む重要な部分として設計の初期段階から考慮に入れられてきた（最近，文献2),7),16)でも，汎用性に加えて，論証支援システムにおけるインタフェースの重要性が論じられている）．

論理系は一般に，言語系（記号，式，論理式など）と導出系（公理，推論規則・書き換え規則・派生規則など）からなる．以下これらを記述する方法について簡単に説明する．

2.2.1 言語系の記述

演算子優先順位付き確定節文法

EUODHILOSでは，ユーザの論理の言語は，論理に固有の概念（変数の束縛・スコープ，代入，スキーマ変数）の記述を可能にするために拡張された確定節文法(DCG)[26),22),23)]で記述される（具体的な定義例は次節で例示される）．

このさい，特殊な論理記号などは，フォントエディタで設計し，各論理ごとに準備された仮想キーボードに割り付けることができる．

パーザ・アンパーザの自動生成

このような文法に対して，式の内部構造を自動的に生成する機構，パーザ・アンパーザの自動生成アルゴリズムが考案され，EUODHILOSの言語定義部で有効に使われている[19),22),23)]．

構文チェッカ・論理式エディタ

またユーザの意図した言語を効率よく定義することを可能にするため，定義した構文をテストできる構文チェッカや，演算子の強さなどが正しく定義されたかなどを視覚的にチェックできる論理式エディタも準備されている[22)]．これらは言語定義のさいのユーザ負担を軽減するのに役立っている．

2.2.2 導出系の記述

推論規則の記述に対しては次の二つを考慮に入れることが必要となる：規則の適用条件の自動チェックと結論の仮定に対する依存性の管理．LF, Isabelleでは自然演繹法以外の依存性の管理などは未だ考慮されていない[12),24)]．

公　理

推論の出発点となる公理は，論理式のリストとして（名前付きで）与えられる．

推論規則

推論規則は次のように，仮定，前提，結論，適用条件の四つから自然演繹法[27)]スタイルで記述される．

$$\frac{\begin{array}{cccc}[\text{仮定}1]\ [\text{仮定}2]\ \cdots\ [\text{仮定}n] & & & \\ \vdots & \vdots & \cdots & \vdots \\ \text{前提}1 & \text{前提}2 & \cdots & \text{前提}n\end{array}}{\text{結　論}}\ \text{適用条件}$$

適用条件とその自動チェック

EUODHILOS では，適用条件は多くの論理にしばしば現れる次の三つの形式の基本条件とその組み合わせによって与えられる：

(1)t is free for x in P (代入条件)，

(2)x is not free in P (変数自由出現条件)，

(3)a is an eigenvariable (固有変数条件)，

そして，これらの条件は証明の過程で自動的にチェックされることになる．この方法で扱うことのできない他の条件に対しては，ユーザが書いた適用条件チェッカを組み込むためのインタフェースが用意されている．

依存性 (dependency)

EUODHILOS では，通常の自然演繹法[27]における結論の仮定に対する依存性は，仮定に対して与えられた自然数をもとに，集合計算で自動的に管理されている．集合計算でなく他の管理方法が必要とされる論理に対しては，EUODHILOS ではタグ（あるいはラベル）付き論理式と，タグを操作するための書き換え規則を定義することによって扱うことができる[34]．このアイディアは元々 Meyer[20]，Gabbay[9]らによって適切論理などの非標準論理を形式化するさいに導入されたものであるが，汎用システムにおいて依存性を一般的に定義するのにも大変有用である．例えば，次の適切論理の ∧-I 規則

$$\frac{P^{\alpha}\ Q^{\alpha}}{(P \wedge Q)^{\alpha}}$$

は，P，Qが依存する仮定が同一の時にのみ（直観的には，同じ根拠の下でP，Qが得られたとき）推論が可能となるものであり，このような依存性は通常の自然演繹法では取り扱うことができない．EUODHILOS では，論理式をタグ付きのメタ論理式として表し，さらにαの部分の計算を書き換え規則で与えるものとして，次のように表現して取り扱い可能とした．

$$\frac{\alpha => P \quad \alpha => Q}{\alpha => P \wedge Q}$$

書き換え規則

書き換え規則は，書き換え前と後の式で定義される．EUODHILOS では書き換え規則を自動的に何度も適用でき，その回数の上限をユーザが指定できるようになっている．

派生規則

派生規則はそれが以下で述べる思考シート上でその証明が与えられたとき定義可能となる．

EUODHILOS の以上の論理記述枠組みの下では，Hilbert 型，Gentzen 型 (ND および LK，LJ)，等式型の公理系の記述が可能であることは明らかである．また，いくらか不自然になることを除けば，Tableau 型の定義も可能であることがわかる．その他のよく知られている公理系の中で，Fitch 型の公理系については，我々の枠組みでも，また LF，Isabelle でも定義不可能である．

まとめると，これまで述べてきた EUODHILOS の論理記述法は次のように特徴づけられる：

(i)対象論理は，その証明論的性格を直接反映するような形で記述される．

(ii)対象論理は，通常の論理の教科書に現れるような形で記述される．

その結果，EUODHILOS の方法は他の方法[12),24)]に比べて広範囲のユーザにも自然で取り扱いやすいのが大きな特徴であると言える．それのみならず，LF や Isabelle ではユーザの論理系を他の体系にコーディングするため，証明が長くなったり，また証明が直観的にわかりにくくなる傾向にあるが，EUODHILOS のアプローチではこのような欠点は生じない[16]．

2.2.3 証明構築支援環境と機能

これまでの汎用論証システムでは，定義された論理の下での証明を支援する証明構築方法には未だほとんど関心が払われておらず，トップダウンあるいは後ろ向きに証明を構成していく方法が趨勢であった．しかしながら，EUODHILOS では論理の形式的証明を自然にかつ柔軟に，さらに効率よく支援する環境と方法が用意されている[30),31),33)]．EUODHILOS の証明は以下の証明構築機能/環境を用いて対話的に進行する．もちろん，自動定理証明機とは異なり証明の主体は，ユーザである．

思考シート

思考シートとは，証明断片からのパターンマッチング／ユニフィケーションによる証明の結合，分離などの操作や，下記の証明方法での証明が行える場，あるいは環境である．

証明方法

EUODHILOS では次のようなさまざまなスタイルでの証明方法が可能である：(i)前向き/後ろ向き証明，(ii)証明の結合によるこれらの組み合わせ，(iii)すでに得られたレンマ，派生規則を利用する証明，(iv)メタ変数を用いる図式証明．

証明木の形式

証明は思考シートと呼ばれるウインドウの上で木形式の証明木を保持しながら行われる．参考のために，付録1に一般論理における派生規則 “X|－～∀y. A(y) => X⊢∃y.～A(y)” の証明木を掲げる．

自動証明機とのインタフェース

既存の定理証明機や項書き換えシステムの結果を有効利用し，さらに証明の効率化を計るためにEUODHILOSには，外部自動証明機を付加するためのインタフェースが用意されている．

データベース

証明された定理，派生規則はそれぞれのデータベースにストアでき，他の証明で自由に使われる．また証明の途中結果や，思考シート上の証明部品などはワークと呼ばれる領域に（必要ならば名前を付けて）ストアしておき，後で証明を再開することができる．

EUODHILOSで構築された論理／理論は論理データベースに蓄えられ，論理メニューで見ることができる．ユーザは，新しく論理／理論を考えるとき，これらすでに存在する論理／理論をコピー，修正するか，あるいは全く新しく定義を始めることになる．

またEUODHILOSの使いやすさと論証に適したインタフェースにも配慮がなされている．しかしながら，これについては別の所で論じることにし，本論文ではこれ以上触れない．

2.2.4 EUODHILOSの実現

これまで述べてきた機能がどのように実現されたかについて述べることは本論文の目的ではないが，実現の概要のみを述べておくことにする．EUODHILOSはPSI上のオブジェクト指向PROLOGであるESP[5]で実現された．PROLOGが，EUODHILOSのような記号処理システムの実現に適していることは言うまでもないが，ESPのオブジェクト機構（特に，クラス，インスタンス，インヘリタンス）はEUODHILOSのようなメタシステムの実現には特に有効であった．例えば，各ユーザ定義の論理は論理クラスのインスタンスとして生成，管理できるからである[35]．実際，EUODHILOSは二つの基本的なクラス：assistant_systemとtheory，をもつ．assistant_systemは各論理に共通の機能：定義機能，証明編集機能など，の支援機能を司るクラスであり，theoryは論理データ：記号，言語，推論規則，定理など，を保持し管理することを司るクラスである．

3. EUODHILOSのさまざまな論理への応用

本章では，EUODHILOSで定義され証明実験が行われた論理のいくつかを取り上げ，各論理ごとにEUODHILOSの論理記述法，証明構築機能の有効性を調べる．

EUODHILOSの論理記述法，証明構築機能でこれまでうまく取り扱うことができた論理系，理論，証明例には次のものが含まれる[32]．

(1)一階述語論理：さまざまな妥当論理式，Smullyanの論理パズル，停止問題の非可解性，帰納法による証明，初等カテゴリー理論，ハードウェア検証．
(2)二階述語論理：数学的帰納法と完全帰納法の同値性．
(3)命題様相論理：プログラムについての論証．
(4)内包論理[10]：メタ定理のリフレクティブ証明[38]，Montagueの自然言語の意味論．
(5)Martin-Löfの直観主義的型の理論[17]：数々の直観主義的命題，選択公理など．
(6)Hoare論理[14]，動的論理[13]：プログラムの性質の証明．
(7)一般論理[36]：述語論理，適切論理とその弱い体系の論理式．
(8)適切論理[20),29)]：いくつかの典型的な適切論理式．
(9)知識の論理：論理パズル．

一階述語論理

(1) 停止問題の非可解性

拡張されたDCGで一階言語を定義することは容易である[32]．例えば，限量化された論理式は次のように定義することができる．

```
formula --> bind_op, variable, formula ;
bind_op --> "∀"|"∃".
```

ここで，“bind_op” は変数束縛の概念を扱うための特別な構成要素であり，このすぐ右には束縛変数が現れ，変数の右隣には変数束縛のスコープとなる式が続くことを表す．

停止問題の非可解性は次のいくつかの述語：

A(x)：xはアルゴリズムである，
C(x)：xはあるプログラミング言語で書かれたプログラムである，
D(x, y, z)：xはyが与えられた入力zで停止するか否かを決定できる，

を用いると，

$$\vdash \neg\exists x(A(x)\wedge\forall y(C(y)\supset\forall z D(x,y,z)))$$

と表現され[4]，定義された自然演繹法を基礎論理系とし Church の提唱などを公理とする論理式から証明された[32]．証明のステップは推論・派生規則の適用回数で約70ステップ程度であった．しかしながら，論理式が長いことによる証明木の横への拡がり過ぎによって，画面のスクロールを頻繁に行わなければならなかったという欠点が見られた．このような大きな証明木に対する表示上の欠点は，他のシステムで主流の線形に並べられた証明[11),15),38)]とかある種の証明の短縮・簡約手法との併用によって克服できるものと考えている．

（2） 初等的圏論

圏論の言語は，射と対象を基本オブジェクトとする一階言語であるので，我々の拡張された確定節文法で容易に定めることができる．圏論の推論規則は数十にのぼるため，ここですべてを述べることはしない．それらのいくつかを示す．射Fの定義域を dom(F)，値域を cod(F) と表すと，射FとGが合成可能なのは，cod(F) と dom(G) が一致するときであり，逆にその時は合成可能である．これらのことは次のように二つの推論規則によって表すことができる．

$$\frac{\mathrm{cod}(F)=\mathrm{dom}(G)}{F\cdot G}(\text{dot-I})$$

$$\frac{F\cdot G}{\mathrm{cod}(F)=\mathrm{dom}(G)}(\text{dot-E})$$

射Fの定義域がXで値域がYであることは，F : X->Y と表されるが，このような定義は次のような三つの推論規則で表される．

$$\frac{\mathrm{dom}(F)=X\ \ \mathrm{cod}(F)=Y}{F:X->Y}(->\mathrm{I})$$

$$\frac{F:X->Y}{\mathrm{dom}(F)=X}(\text{dom-I})\qquad\frac{F:X->Y}{\mathrm{cod}(F)=Y}(\text{cod-I})$$

このような例からもわかるように，これらの推論規則は EUODHILOS の規則記述法で直接的に記述することができる[32]．

証明実験は，これらの理論の定義を加えた自然演繹法の下で，初等的圏論の教科書に現れるいくつかの定理を証明することによって行われた．その結果，この理論の証明では類似の証明パターンが適用できるケースが多く見られ，これらはすでに証明された派生規則を用いて証明を簡単化することができた．また含意形の定理を定理データベースに蓄えて他の証明で用いるより，それらを仮定からの派生規則の形式にして保存し利用することの方が便利であることが多かった．さらに，このような理論では規則数が膨大になるため，導出のさい規則メニューから規則を選び出すより，EUODHILOS の適用可能な規則を捜し出す機能が特に有効であった．

内包論理

内包論理[10]は，単純型の理論に基づく高階の様相論理である．この論理の言語は，型に依存した形成規則，すなわち文脈依存性をもつ形成規則によって定義されている．しかしながらこのような文脈依存性は拡張された DCG で容易にかつ自然に記述できる．例えば，内包論理の形成規則の一つ，「Tが型 (A, B) の項でSが型Aの項であれば，T・Sは型Bの項である」は，我々の内部構造自動生成機構付き DCG 記法[22]では，

term(B)-->term((A, B)), "・", term(A).

と表現することができる．他方，本来の DCG[26]では処理の対象となる項の内部構造を含めて，

term(T・S, B)-->term(T,(A, B)), "・", term(S, A).

と書かなければならない．この例からもわかるように我々の記法の方が，内部構造の組立というユーザにとって本質的でない作業を避けることができ，さらに誤りの混入を防ぐことができるという点で，明らかに優れていると言える．

内包論理の Gallin による公理系[10]は Hilbert 型であるから，前節で述べた枠組みで容易に記述される[32]．

（1） メタ定理の証明

証明実験の一つとして，内包論理のメタ定理である一般化規則，

$\vdash P:t \Rightarrow \vdash \forall x:a.P:t$

の証明を試みた．この定理を証明するためには，まずメタ表現である $\vdash P:t$ を仮定し，次に "P : t" をオブジェクト論理としての内包論理の世界に持ち込み，そこで証明を進め結論 "$\forall x:a.P:t$" を得た後でそれを再びメタの世界に引き戻し $\vdash \forall x:a.P:t$ を得るということを行わなければならない．そのためには必要な限りのメタ言語を定義し，さらに次のメタ，オブジェクト間の相互行き来を許すリフレクションの規則[38]，

$$\frac{\mathrm{provable}(A)}{A}(\text{reflection})$$

$$\frac{A}{\mathrm{provable}(A)}(\text{reflection})$$

を定義する必要があった．ただし，このような場合，メタとオブジェクト論理の区別は現バージョンの EUODHILOS によって自動的にサポートされているわけではないので，どの部分がメタでどの部分がオブ

ジェクトであるかといった点には注意を払わなければならない.

(2) Montague の意味論

Montague の言語理論では，自然言語文は初めに内包論理の式に変換され，次いでそれが多世界意味論の下で分析されることになる[10),32)]．ここではこの理論の前半の変換の部分に対する EUODHILOS の応用を考える．例えば，次の自然言語文，

John believes that a fish walks.

は，自然言語から内包論理への変換規則を EUODHILOS の書き換え規則で表現し，それらを適用すると，

$$(\lambda p : (s,(e,t)).\ \exists x : e.\ (fish : (e,t) \bullet x : e \wedge p : (s,(e,t))\{x : e\})) \bullet \hat{}\lambda y : e.\ (believe : ((s,t),(e,t)) \bullet \hat{}(walk : (e,t) \bullet y : e) \bullet j : e)$$

のようなかなり複雑な論理式が得られる．これに対して，内包論理の推論規則を適用し同値変形していくと，最終的に次のより簡単な論理式を得ることができる，

$$\exists x : e.\ (fish : (e,t) \bullet x : e \wedge believe : ((s,t),(e,t)) \bullet \hat{}(walk : (e,t) \bullet x : e) \bullet j : e).$$

このように，自然言語分析における複雑な論理式操作を容易にかつ正確に行うためのツールとして EUODHILOS を有効に用いることができる．

Martin-Löf の直観主義的型の理論

この論理の基本論理式 (judgement) は "a∈p" という形式である．ここで a はラムダ式，p は型として解釈された論理式を表す[17)]．したがってこれらを EUODHILOS で定義することは容易である．また推論規則も通常の自然演繹型であるので，EUODHILOS の記述法に最も適合する．例えば，λ-導入規則は次のように Martin-Löf の記法そのままに表現される．

$$\frac{\begin{array}{c}X \in A \\ \vdots \\ F(X) \in B\end{array}}{\lambda X.\ F(X) \in A \supset B}$$

ただし，ここで適用条件 "X is not free in B" が必要になるが，これは前節で述べた EUODHILOS の基本適用条件の一つであるから問題なく記述でき，証明の過程でこの条件は自動的にチェックされることになる．

証明実験では，直観主義論理における 50 個程度の代表的な論理式，さらに選択公理などが選ばれ証明された．この中で例えば，背中律の二重否定の証明過程では，前向き証明で得られた結論，

$$\lambda f.\ f \bullet inr(\lambda x.\ f \bullet inl(x)) \in (p \vee (p \supset \bot) \supset \bot) \supset \bot$$

と，別の思考シートに置かれた証明のゴール，$F \in \neg\neg(p \vee \neg p)$ に対して，書き換え規則，$\neg p \Rightarrow (p \supset \bot)$，を逆向きに適用して得られる証明断片，

$$\frac{F \in (p \vee (p \supset \bot) \supset \bot) \supset \bot}{F \in \neg\neg(p \vee \neg p)}$$

の上式がユニファイされ，次の基本論理式，

$$\lambda f.\ f \bullet inr(\lambda x.\ f \bullet inl(x)) \in \neg\neg(p \vee \neg p)$$

が最終的に得られた[32),33)]，ここで F はメタ変数である．この簡単な例は，証明の過程で何らかの情報を求めるような論理計算のとき，それらをいくつかの証明断片から得る方法として前節で述べたユニフィケーションによる証明の結合機能が大変有効であることを示している．

Hoare 論理

Hoare論理は，プログラムの部分正当性を証明するための論理，あるいはプログラムの公理的意味論の一手法としてよく知られている[14)]．Hoare論理の基本論理式は，"P{S}Q" で P と Q は述語論理式，S はあるプログラミング言語で書かれたプログラムである．プログラムの定義も拡張された DCG で容易に表現でき，また公理系も Hilbert 型であるので，Hoare 論理は EUODHILOS の論理記述枠組みで容易に捉えられる．特に，代入文の公理を次のように直接的に表現することが可能である☆：

$$P(T/X)\{X := T\}P.$$

ただし，P(T/X) は P の中の X のすべての自由生起に T を代入して得られる式を表す．

これまで，文献1)にあるいくつかの基本的なプログラムに対してその正当性の証明を行った．プログラムに関する証明木の葉には，トートロジー，算術式，データ構造に関する命題などがいつも現れる．これらの証明に対しては，EUODHILOS の外部定理証明機インタフェースを利用して，すでに証明済みの定理データベースから定理を検索する外部定理検索機を呼び出すことによって証明が効率よく行われた．

Hoare 論理の基本論理式 P{S}Q に現れる S はプログラムであるため，一般に極めて長い式となる．実際，この場合も証明木の横への拡がりは縦より飛躍的に増加していく．それを救う一方法として，プログラムセグメントに名前付けを行う機構が望まれた．

☆ Isabelle の開発者である Paulson によれば，Isabelle では代入の処理が一般的でないため，Hoare論理を扱うことができない[23)]．LF における Hoare 論理のかなり複雑な定義については文献18)を見よ．

一般論理

一般論理とは，広範囲の論理に対して統一的な説明を与えることのできる論理体系である．ここで取り上げた一般論理は，Slaney によって形式化された論理[36]で，Gentzen のシーケント型の形式的体系となっている．しかしながら，この論理のシーケントは通常の LK などとは異なり，二つの区切り記号，“,” と “;” によって区切られた論理式の有限列を扱い，それらの区切り記号に与えられる性質によっていろいろな論理が表示される．例えば，この論理のモーダスポネンスは，次のような形式をもつ．

$$\frac{X \vdash A->B \quad Y \vdash A}{X;\ Y \vdash B}$$

ここで，X，Y は上の意味での論理式の列を表す．このような一般論理の言語および推論規則は EUODHILOS の枠組みで容易に記述される．一般論理の枠組みでは，例えば古典述語論理は，すでに定義された適切論理に次のような “;” に関する Thinning の規則を追加することによって得られる[36]．

$X \vdash A => X;\ Y \vdash A.$

明らかに，このような “;” の性質は EUODHILOS では書き換え規則，$X=>X;\ Y$，として記述できる．

EUODHILOS による一般論理の証明は，区切り記号が2種類あることを除き，基本的には LK と同じように進めることができる．行われた証明は次のような基本的な論理式，派生規則に対してであった．

$p, q \vee r \vdash p \wedge q \vee r$ (分配則)，

$X;\ B \vdash A \wedge \sim A => X \mid - \sim B$

(Reductio ad absurdum)，

$\vdash \exists y.(g(y)-> \forall x.\, g(x))$ (Baffling formula)，

$X \vdash \sim \forall y.\, A(y) => X \mid - \exists y.\, \sim A(y)$ (この派生規則の証明木を付録1に掲げる)．

これらの証明に対しては，前向き，後ろ向き，あるいはそれらの組み合わせによる試行錯誤的証明スタイルを柔軟に適用することができ，EUODHILOS の証明スタイルを強制しない自由度の高い証明方法の便利さ，有利さが確認された[32]．特に，一般論理のようなあまり知られていない論理の証明，あるいは新しい論理を形式化したときなどのように，導かれる定理が予想しにくいときなどでは，EUODHILOS の自由度の高い証明スタイルは効果的であった．

適切論理

適切論理 (Relevant logic)[29]では，含意命題 “A->B” が真となるのは前提Aが帰結を適切に含意するとき，そしてそのときに限るとされる．このような命題の導出を可能にする適切論理の形式化では，推論規則に古典論理とは異なったかなり複雑な条件が付加されることが多い．論理実験では，適切な推論を結合子の計算によって行うように形式化された適切論理の含意部分体系 $R_{\to}$[29]を取り上げた．

$R_{\to}$ の言語は論理記号として “->” のみをもつ命題言語である．ただし，論理式は推論を適切に制御するための Tag をもった “Tag=>Formula” の形式である．Tag は論理式の正当化（根拠）を表す記号とそれらを二項演算子（適用オペレータ）で結合してできる結合項である．したがって，これらを EUODHILOS の文法で記述することは容易である．$R_{\to}$ の推論規則は自然演繹型の，

$$\frac{\alpha => P->Q \quad \beta => P}{\alpha\beta => Q}(->E) \qquad \frac{\begin{array}{c}\alpha => P \\ \vdots \\ \beta\alpha => Q\end{array}}{\beta => P->Q}(->I)$$

であり，Tag の計算は，例えば結合子Cに対応するものは，EUODHILOS では書き換え規則として，

$$\frac{\alpha\beta\gamma}{\alpha\gamma\beta}$$

のように定義することができる．Tag によって推論規則の帰結の仮定に対する依存性を表現するという方法は，他の論理にも通用する有効な方法であり，通常の集合計算を Tag として表せば自然演繹法でさえ表現できることになる[34]．

証明実験は，$R_{\to}$ の Hilbert 型の公理系における次の四つの公理の証明に対して行われた：

(1) $\vdash (P->(Q->R))->(Q->(P->R))$ (Permutation)，

(2) $\vdash (P->Q)->((R->P)->(R->Q))$ (Prefixing)，

(3) $\vdash (P->(P->Q))->(P->Q)$ (Contraction)，

(4) $\vdash P->P$ (Self-implication).

付録2にこれらの証明を示す．この論理では，その言語が単純で，さらに Tag によって推論が簡単化されたことにより，論理の定義および上記の証明が極めて短時間で遂行された．

ここで取り上げた論理は，証明スタイルは自然演繹型であるが，依存性の計算は通常のものとは異なる論理体系であった．EUODHILOS は，新しい論理を形式化したとき，その言語の修正，導かれる定理の確認といった論理構築作業にも都合がいい機能と柔軟性を備えており，論理構築ツールとしての有用性も期待できる．

4. 評価と今後の課題

これまで，さまざまな論理による論理定義，および定義された論理の下での証明を通して，EUODHILOS の使用経験を積み評価を行ってきた．そこでは，EUODHILOS の汎用性の優位さとさまざまな領域における潜在的な有用性が検証された．この章では，三つのポイントについて EUODHILOS の評価を与える．

（1） 論理記述法

3章で列挙したように，EUODHILOS ではこれまで計算機科学や人工知能の研究で現れた論理，および純粋論理学の研究から生まれた論理など，かなり広範囲の論理を扱うことができている．これを LF，Isabelle などと比較した場合，EUODHILOS は記述能力のみならず記述のしやすさの点においても優れていると言える[12),18),24)]．

3章で取り上げた論理に対しては，その定義とそこでの 2，3 の簡単な証明作成をも含めて，平均して 30 分から 2〜3 時間で完成することができた．EUODHILOS を用いずに，各論理ごとにそのための証明支援システムを初めから作成することを考えれば，EUODHILOS のような汎用論証支援システムの存在意義がいかに大きいものであるかが想像できよう．

（2） 証明構築法

EUODHILOS には，ユーザが証明構築を自然にかつ柔軟に行えるようなさまざまな証明方法，環境が用意された．3章で，各論理ごとにその使用経験に基づく利点を中心に述べた．しかしながら，EUODHILOS には Tactics，Tacticals などの証明戦略による証明法が未だ組み込まれていないため，大きな証明構築に対して問題を残していることは否定できない．汎用システムにうまく整合する，戦略による証明の方法，およびこれらの言語の設計については今後の課題として残される．

（3） 応用領域

EUODHILOS の有力な応用領域は，計算機支援プログラミング，研究者の論理構築支援ツール，論理教育ツールなどが有望である．これまで，EUODHILOS では Hoare 論理，動的論理などにおいて簡単なプログラムの検証が試みられた．今後，EUODHILOS 上に VDM，Z などの仕様記述言語を定義し，仕様からの正しいプログラムの導出に EUOD HILOSの証明構築支援機能を適用する研究を考えていきたい．

謝辞 EUODHILOS のプログラム開発に当たっては，大橋恭子，横田かおる，お二人のご尽力に負うところが大きく，ここに感謝の意を表したい．

EUODHILOS の研究の過程で，ケンブリッジ大学の Paulson，およびインペリアルカレッジの Dawson から貴重なコメントをいただき，またさまざまな角度から議論していただいた．オーストラリア国立大学の Meyer からは適切論理の依存性の問題についてご教示いただいた．

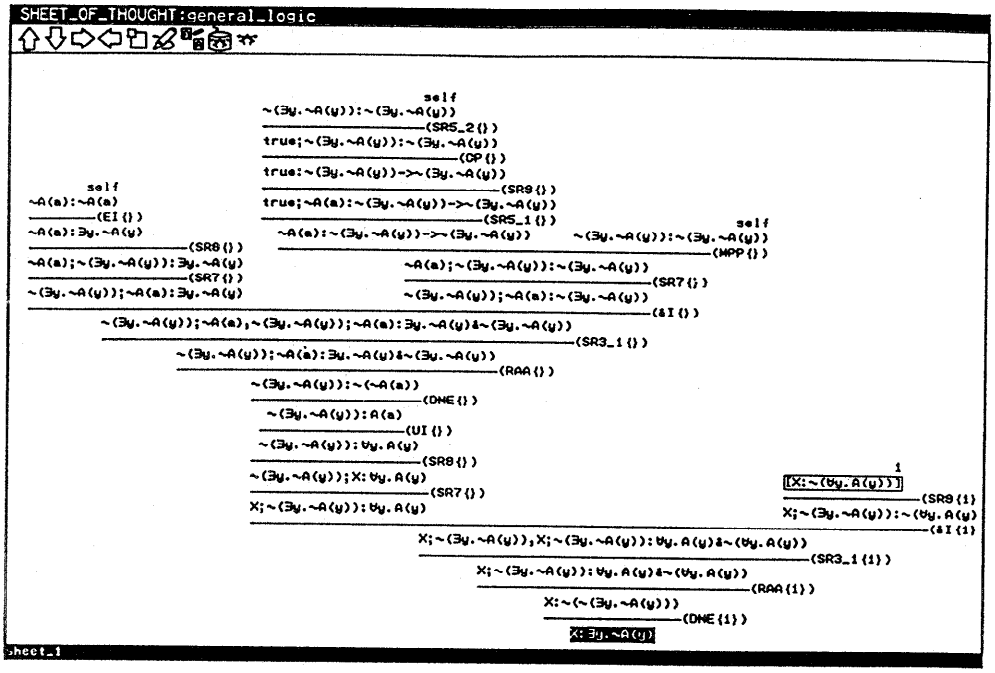
参考文献

1) Alagic, S. and Arbib, M. A.: *The Design of Well-structured and Correct Programs*, Springer (1978).
2) Bedau, M. and Moor, J.: Proof Designer: A Programmable Prover's Workbench, in Burkholder., L. (ed.): *Philosophy and the Computer*, Westview, pp. 1-11 (1991).
3) de Bruijn, N. G.: A Survey of the Project Automath, in: Seldin and Hindley (eds.), *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, Academic Press, pp. 576-606 (1980).
4) Burkholder, L.: The Halting Problem, *SIGACT NEWS*, Vol. 18, No. 3, pp. 48-60 (1987).
5) Chikayama, T.: ESP—Extended Self-Contained Prolog—as a Preliminary Kernel of Fifth Generation Computers, *New Generation Computing*, Vol. 1, No. 1, pp. 11-24 (1983).
6) Constable, R. L., et al.: *Implementing Mathematics with the Nuprl Proof Development System*, Prentice-Hall (1986).
7) Dawson, M.: A Generic Logic Environment, Ph. D. thesis, Dept. of Computing, Imperial College (1991).
8) Felty, A. and Miller, D.: Specifying Theorem Provers in a Higher-order Logic Programming Language, *LNCS 310*, Springer, pp. 61-80 (1988).
9) Gabbay, D.: Labelled Deductive Systems, CIS-Bericht-90-22, University of Munchen (1990).
10) Gallin, D.: *Intensional and Higher-order Modal Logic, with Applications to Montague Semantics*, North-Holland (1975).
11) Gordon, M. J., Milner, A. J. and Wadsworth, C. P.: Edinburgh LCF, *LNCS 78*, Springer (1979).
12) Harper, R., Honsell, F. and Plotkin, G.: A Framework for Defining Logics, *Proc. of Symposium on Logic in Computer Science*, pp. 194-204 (1987).

13) Harel, D.: Dynamic Logic, in Gabbay, D. and Guenthner, F. (eds.): *Handbook of Philosophical Logic, Volume II: Extensions of Classical Logic*, D. Reidel, pp. 497-604 (1984).
14) Hoare, C. A. R.: An Axiomatic Baisis for Computer Programming, *CACM*, Vol. 12, No. 10, pp. 576-580, p. 583 (1969).
15) Ketonen, J. and Weening, J. S.: EKL—An Interactive Proof Checker, User's Reference Manual, Dept. of Computer Science, Stanford Univ. (1984).
16) Lindsay, P. A.: The μral Proof Assistant, *Proc. ACSC-13, Australian Computer Science Communication 12*, pp. 236-245 (1990).
17) Martin-Löf, P.: Intuitionistic Type Theory, Bibliopoplis (1984).
18) Mason, I. A.: Hoare's Logic in the LF, ECS-LFCS-87-32, Dept. of Comp. Sci., Univ. of Edinburgh (1987).
19) Matsumoto, Y., Tanaka, H., Hirakawa, H., Miyoshi, H. and Yasukawa, H.: BUP: A Bottom-up Parser Embedded in Prolog, *New Generation Computing*, Vol. 1, pp. 145-158 (1983).
20) Meyer, R. K.: A General Gentzen System for Implicational Calculi, *Relevance Logic Newsletter*, Vol. 1, No. 3, Australian National University, pp. 189-201 (1976).
21) 南, 大橋, 沢村, 大谷: 汎用論証支援システム EUODHILOS 図解マニュアル, IIAS-RR-92-18 J, (株) 富士通研究所 (1992).
22) 大橋, 横田, 南, 沢村, 大谷: 確定節文法のための内部構造変換機構付きパーザとアンパーザの自動生成方式, 情報処理学会論文誌, Vol. 31, No. 11, pp. 1616-1626 (1990).
23) 大橋, 南, 沢村, 大谷: 論証支援システム EUODHILOSのための構文記述法の改良とパーザの実現, 第44回情報処理学会全国大会論文集 (1992).
24) Paulson, L. C.: The Foundation of a Generic Theorem Prover, *J. of Automated Reasoning*, Vol. 5, pp. 363-397 (1989).
25) Paulson, L. C.: Private communication (1992)
26) Pereira, F. C. N. and Warren, D. H. D.: Definite Clause Grammars for Language Analysis—A Survey of the Formalism and a Comparison with Augumented Transition Networks, *Artif. Intell.*, Vol. 13, pp. 231-278 (1980).
27) Prawitz, D.: *Natural Deduction*, Almqvist & Wiksell (1965).
28) 沢村, 南: 汎用の論理支援システムの構想とその実現法, 情報処理学会ソフトウェア基礎論研究会, 87-SF-22 (1987).
29) 沢村: 適切さの論理, 情報処理学会論文誌, Vol. 30, No. 6, pp. 665-673 (1989).
30) Sawamura, H., Minami, T., Yokota, K. and Ohashi, K.: A Logic Programming Approach to Specifying Logics and Construcing Proofs, *Proc. of the Seventh International Conference on Logic Programming*, Warren, D. H. D. and Szeredi, P. (ed.), The MIT Press, pp. 405-424 (1990).
31) Sawamura, H., Minami, T., Yokota, K. and Ohashi, K.: Potential of a General-Purpose Reasoning Assistant System EUODHILOS, in Nakata, I. and Hagiya, M. (eds.): *Software Science and Engineering, Selected Papers from the Kyoto Symposia*, World Scientific Pub., pp. 164-188 (1991).
32) Sawamura, H., Minami, T., Ohtani, T., Yokota, K. and Ohashi, K: A Collection of Logical Systems and Proofs Implemented in EUODHILOS I, IIAS-RR-91-13E, Fujitsu Lab. (1991).
33) Sawamura, H., Minami, T. and K. Ohashi: Proof Methods Based on Sheet of Thought in EUODHILOS, IIAS-RR-92-6E, Fujitsu Lab. (1992).
34) Sawamura, H., Minami, T. and Meyer, R. K.: Representing a Logic in EUODHILOS, IIAS-RR-93, Fujitsu Lab. (1993).
35) Sawamura, H., Minami, T. and Ohashi, K.: EUODHILOS: A General Reasoning System for a Variety of Logics, *Proc. of Logic Programming and Automated Reasoning, Lecture Notes in Artificial Intelligence 624*, Springer, pp. 501-503 (1992).
36) Slaney, J.: A General Logic, *Australasian J. of Philosophy*, Vol. 68, No. 1, pp. 74-88 (1990).
37) Turner, A.: *Logics for Artificial Intelligence*, Ellis Horwood Limited (1984).
38) Weyhrauch, R. W.: Prolegomena to a Theory of Mechanized Formal Reasoning, *Artif. Intell.*, Vol. 13, pp. 133-179 (1980).

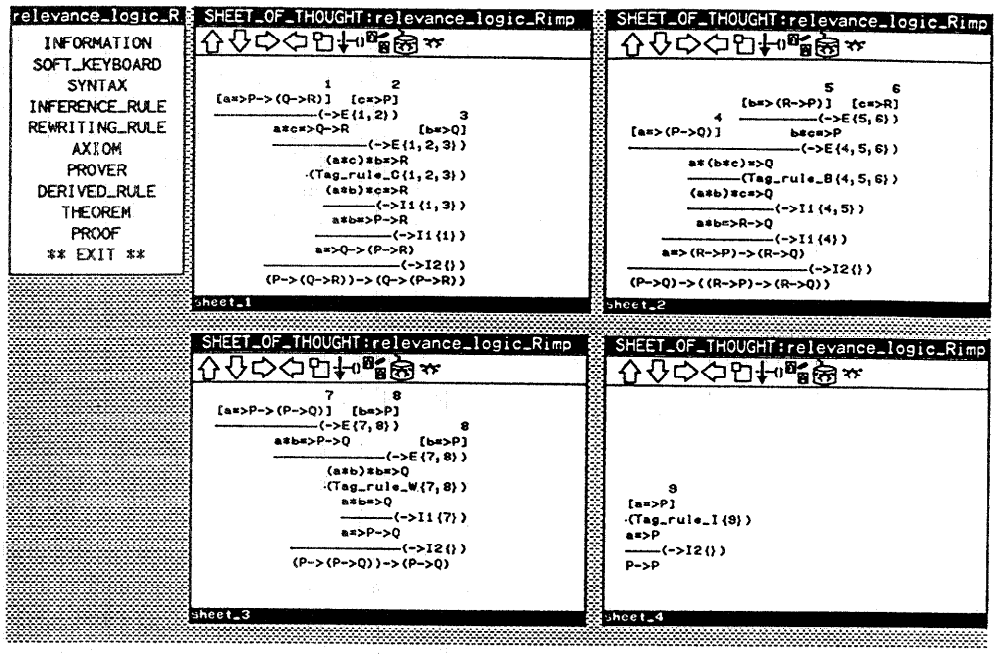
付 録 1

スクリーン中の証明木は，一般論理[36]における派生規則，"X⊢～∀y. A(y)=>X⊢∃y. ～A(y)" の証明を表している，ただし証明木の中では，記号 "⊢" の代わりに ":" が用いられている．証明木の葉にはself と名付けられた一般論理の公理が現れている．白枠で囲まれ，番号1をもちカッコで括られた論理式 "[X⊢～∀y. A(y)]1" はこの証明の仮定を表す．黒塗の枠で囲まれた式はこの証明の結論である．



付 録 2

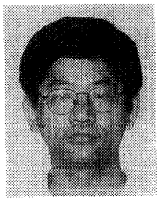
スクリーン中では，$R_{\rightarrow}$ の Hilbert 型の公理系[29]の四つの公理：Permutation, Prefixing, Contraction, Self-implication の証明が見られる．これらの証明ではそれぞれ結合子，C, B, W, I に対応する Tag の計算規則がそれぞれ一度使われている．



(平成 4 年10月 8 日受付)
(平成 5 年 1 月18日採録)

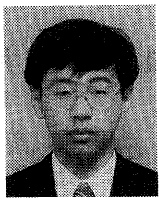
沢村 一（正会員）

1949年生．1978年北海道大学工学部情報工学専攻修了．富士通(株)国際情報社会科学研究所に勤務．論理学とその応用に興味をもつ．日本ソフトウェア科学会，日本科学哲学会などの会員．



南 俊朗（正会員）

1951年生．九州大学理学部数学専攻修了．富士通(株)国際情報社会科学研究所に勤務．圏論，計算機科学，論理学に興味を持つ．日本数学会，LA，EATCS 各会員．



大谷 武

1964年生．1987年東北大学理学部数学科卒業．1989年東北大学大学院工学研究科（情報工学専攻）修士課程修了．同年，富士通(株)国際情報社会科学研究所入所．論証支援システムの研究に従事．ソフトウェア科学会会員．