人工知能学会誌 Vol. 7 No. 2 (March 1992)〔別刷〕

論 文

Technical Papers

論理プログラミング環境におけるEBLの有効性計算

Computing the Utility of EBL in a Logic Programming Environment

山田 誠二*
Seiji Yamada

* 大阪大学産業科学研究所
The Insititute of Scientific and Industrial Research, Osaka University, Osaka 567, Japan.

1991年4月10日 受理

Keywords: explanation-based learning, utility of EBL, logic programming.

Summary

Explanation-Based Learning (EBL) fails to accelerate problem solving in some problem domains. An EBL system therefore needs to evaluate the utility of EBL to a given problem domain for determining whether it adopts EBL. Conventional EBL systems empirically evaluate the utility through test experiments solving a great number of test examples, which results in high evaluation cost. These empirical methods prohibit us from estimating the utility before conducting the test experiment.

This paper presents a formal framework in which the utility of EBL, implemented in a logic programming, is computed analytically without any test experiments. We represent the utility of EBL as a function of two variables, the number and the distribution of test examples, and predict the utility on subsequent problems by analyzing the function. The utility function is determined by analyzing the trace of problem solving on training examples, not test examples.

First, we define a standard EBL procedure including both problem solving and how to add learned rules to a rule base. After only training examples were solved, our method can compute the utility of EBL by evaluating computational cost both of an EBL system and a non-learning system assuming a distribution of test examples. Since our method needs no experimental result on test examples and no execution of learning procedures like EBG, it can efficiently predict the utility of EBL before test experiment. The utility of EBL obtained by our method is a function of the number of test examples, and it approaches to the utility evaluated empirically thus far as the number increases. Finally we show examples of computing the utility of EBL in a Mitchell's SAFE-STACK example. As a result, we have very interesting results that EBL deteriorates problem solving even in simple domain theory such as safe_to_stack (X, Y).

1. はじめに

説明に基づく学習：EBL (Explanation-Based Learning)[1][2] では，問題解決の効率を学習前よりも向上させることに重点が置かれる．EBL により効率が向上する性質を，EBL 有効性と呼ぶ．学習前の効率が，十分に悪いもの(解けない場合も含む)であれば，EBL 有効性は常に保障されるが，そうでない場合は，領域理論や例に依存して，EBL により効率化がなされない場合，つまり EBL が有効でない場合がある．このことが，有効性問題 (utility problem)[3] と呼ばれ，重要課題となっている．

EBL により学習されたルールが，効率化を実現しない場合には，そのルールをシステムが知識として取り込むべきではない．よって，EBL 有効性は学習されたルールを取り込むべきか否かの判断基準となるが，従来この EBL 有効性は，訓練例 (training example) による学習（訓練フェーズ）の後，多数の試験例 (test example) を与えて問題解決の効率向上を実験的に調

べること（試験フェーズ）により求められてきた[3][4]．しかし，このような実験的検証法では，試験フェーズの計算コストが非常に大きいし[4]，さらに大量の問題を実際に解いた後でないと，ルールの取込みの決定ができないという重大な欠点がある．

本研究では，Prologで実装されたEBLを対象として，訓練フェーズだけの結果を分析し，解析的にEBL有効性を計算する手法を提案する[5]．本手法により，EBL有効性は，訓練フェーズで得られる値を変数とする関数で表され，有限および無限個の試験例に対し，学習されたルールを取り込むべきか否かの判断が，試験フェーズなしで可能になる．その結果，前述の実験的検証法の欠点が克服できる．

なお，本研究の目的は，EBL手続きや操作性規準を改良することではなく，固定された単純かつ一般的なEBL手続きが有効となる領域理論および例のクラスを模索することである．本研究の解析の一般性を保つため，解析対象とするEBL手続きとして，領域に依存した最適なものは用いず，単純かつ一般的なものを採用する．そして，その標準的EBL手続きに限定して解析を行うため，結果として得られたEBL有効の条件は，単純なEBL手続きでもEBL有効となる条件を表し，その意味でEBLが有効であるための十分条件となる．

以降では，まず，標準EBL手続き：SEPを設定する．そして，この枠組みに基づいて，EBL有効性を関数としてモデル化し，最後に具体例を示す．

2. EBL実装言語としてのProlog

本研究では，最も一般的な論理プログラミング言語であるPrologを用いて実装されたEBLを対象とする．実装言語として，Prologを選んだ理由を以下に示す．

(1) 領域理論の記述，説明構造（explanation structure）の生成が容易である．

(2) 説明に基づく一般化：EBG（Explanation-Based Generalization）[1]が，Prologの単一化（unification）を用いて簡単に実現できる．

(3) (1)(2)の理由により，PrologがEBL実装のための言語として広く用いられている[6]-[8]．

(4) Prologの定理証明手続きであるSLD導出が，明確でわかりやすく，解析に適している[6]．

3. 標準EBL手続き：SEP

まず，有効性計算の対象となる一般的なEBL手続き：SEPをProlog環境において設定する．EBL有効性は，問題解決手続き（探索戦略など）に依存することが指摘されている[4]ため，SEPは学習および問題解決を統合した手続きとなっている．また，ここでの問題解決とは定理証明を意味し，すべての知識はホーン節で表現される．さらに，PrologはSLD導出による定理証明を行うので，SEPの説明生成および問題解決には，その定理証明手続きを用いる．なお，前述のように本研究の目的は，EBL有効の十分条件を探ることにあり，したがって最適性にはとらわれず，単純で一般的な方法がSEPの手続きとして採用される．

3・1 SEPの入出力

〔1〕 入　力[1]

(1) **目標概念GC**：学習すべき概念を表す述語を頭に持つ確定節集合．その頭をGCHとする．

(2) **領域理論DT**：訓練例が目標概念の肯定例であることを証明するために用いる確定節集合．

(3) **例系列TES**：一般的に学習システムには，まず学習するための訓練例が与えられ，次にパフォーマンス向上を調べるために，新たな学習に寄与しない例，つまり訓練例で学習された概念の肯定例である試験例が与えられる．本研究では，訓練例系列と試験例系列をそれぞれ，$[TRE_1, \cdots, TRE_m]$, $[TEE_1, \cdots, TEE_n]$とし，例全体の系列を$TES=[TRE_1, \cdots, TRE_m, TEE_1, \cdots, TEE_n]$とする．任意の例$E_i$（訓練例または試験例）は，目標概念の肯定例$PE_i$と目標$T_i$からなる集合$PE_i \cup \{T_i\}$である．$PE_i$は事実節集合で，$PE_i \cup GC \cup DT$で目標節$T_i$を証明可能であり，$T_i$はGCHの具象例（ground instance）である事実節とする．

(4) **操作性規準OC**：説明構造上で葉の部分，つまり事実節との導出を切断し，一般化を行う[1]．この操作性規準は，Prolog環境で一般的と考えられる．一つの目標概念の説明構造が複数個ある場合には，より有効な操作性規準が提案されている[9][10]が，§1で述べた方針により本研究では採用しない．

以下に，学習および問題解決手続きの入力を示す．

(5) **説明構造生成手続きGE(E_i)**：目標T_iを例E_i，領域理論DTを用いて証明することにより，説明構造ES_iを生成する手続き．本研究では，Prologで実装された標準的なEBGであるPROLOG-EBG[6]を用いる．PROLOG-EBGは，説明構造の生成と同時に一般化も行うので，

$GE(E_i)$ と次に示す一般化手続き $G(ES_i)$ の双方を含んでいる．

(6) **一般化手続き $G(ES_i)$**：説明構造 ES_i を EBG で一般化し，操作性規準を満たす概念記述 OCD_i を生成する手続き．

(7) **知識管理手続き $M(OCD_i)$**：得られた OCD_i を現在の知識ベースに追加し，更新・組織化を行う手続き．本研究では，新たに得られた OCD_i を OCD の集合の最後に追加する方法を採用する．この方法は，経験的に有効であることが報告されており[11][12]，単純かつ標準的と考える．

(8) **問題解決手続き PS**

- $PS(E_i, DT)$：学習なしの問題解決手続き．具体的には，$GC \cup DT \cup E_i$ で，$\leftarrow T_i$ を目標節とした SLD 導出手続きである．
- $PS(E_i, OCD_{j\cdots k})$：学習されたルールを用いた問題解決手続き．具体的には，$\{OCD_j, \cdots, OCD_k\} \cup FCS \cup E_i$ で，$\leftarrow T_i$ を目標節とし，$\{OCD_j, \cdots, OCD_k\}$ の要素による導出から始まる SLD 導出手続きである．ただし，FCS は DT 中の事実節のみからなる集合とする．

〔2〕 出　力

操作可能な概念記述の集合：上記の手続きで得られた GCH を頭として持つ確定節 OCD_i の集合．

なお，将来的には，操作性基準や知識管理手続きも変数とした有効性解析を考えているが，その解析は非常に複雑である．よって，現時点では，それらを上記の単純なものに限定して解析を行っている．

3・2 SEP および SNP アルゴリズム

SEP と比較するための標準学習なし手続き SNP は，すべての例を DT だけで解く手続きとして設定される．SEP と SNP のアルゴリズムをそれぞれ Table 1 と Table 2 に示す．表中の m，n はそれぞれ訓練例数，試験例数である．SNP の入力を以下に示す．これらの入力は SEP のものと全く同じものが与えられる．

〔1〕 **SNP の入力**

(1) 目標概念 GC

(2) 領域理論 DT

(3) 例系列 $TES=[TRE_1, \cdots, TRE_m, TEE_1, \cdots, TEE_n]$

(4) 学習なし問題解決手続き $PS(E_i, DT)$

Table 1　SEP algorithm.

```
procedure SEP
begin
    i ← 0 ;
    while not(i=m+n) do
        begin
            i ← i+1;
            if PS(Ei , OCD1,...,i-1) fails then
                begin
                    GE(Ei );
                    G(ESi );
                    M(OCDi );
                end
        end
end
```

Table 2　SNP algorithm.

```
procedure SNP
begin
    i ← 0 ;
    while (i=m+n) do
        begin
            i ← i+1;
            PS(Ei , DT) ;
        end
end
```

4. 基 本 概 念

本論文で用いる基本概念の定義を述べる．なお，SLD 木，節点，成功点，失敗点，路というような一般的概念については，文献(13)などを参考にされたい．ただし，SLD 木の節点と枝は，それぞれ選択基本式 (selected atom) と導出に使われたホーン節のラベルで特徴づけられるとする．また，選択基本式とは，目標節のうちで導出の対象として選ばれる基本式である．Prolog では，目標節を $\leftarrow A_1, \cdots, A_m$ とすると，まず一番左側の A_1 が選ばれる．

【定義 1】 EBL 有効性 (utility of EBL)

SEP と SNP の計算コストを返す関数をそれぞれ LC, NLC とし，EBL の有効性関数 U を，$U=(NLC-LC)/NLC$ と定義する．U の値域は $(-\infty, 1)$ で，その値が 1 に近いほど，学習による効率向上が大きく，0 で学習効果なし，負であれば学習により効率が低下することになる．よって，$U>0$ が，EBL が有効である条件となる． □

【定義 2】 操作可能節点および操作不可能節点 (operational/non-operational node)

ある節点で選択された選択基本式が事実節である場合，その節点を操作可能節点，それ以外の節点を操作不可能節点とする． □

【定義 3】 SLD 説明構造 (SLD explanation structure)

PS(E_i, DT) の SLD 木において，最初に見つかる成功路． □

【定義 4】 SLD 説明経路（SLD explanation history）

PS(E_i, DT) の SLD 木において，SLD 説明構造を見つけるまでにたどった部分木．証明に成功するまでの探索空間を表す． □

【定義 5】 SLD 操作可能経路（SLD operational history）

SLD 操作可能経路は，SLD 説明経路上の節点のうち，以下のものから構成される木である．

(1) 根

(2) SLD 説明構造上の操作可能節点．

(3) (2)の子孫で SLD 説明構造上にない節点のうち，以下の節点．

- (2)の子節点．
- 操作可能節点だけを通って(2)のいずれかの節点にたどり着ける節点． □

なお，SLD 説明構造上，SLD 説明経路および SLD 操作可能経路は，節点の系列で表現されるとする．以上の概念について，Fig. 1 に具体例を示す．DT，TRE_1 から得られる SLD 説明経路，SLD 操作可能経路および SLD 説明構造が示されており，各節点のラベルの下線部が選択基本式である．また，(A)，(B)は，DT と TRE_1 から得られる SLD 説明経路，OCD_1 と TRE_1 から得られる SLD 説明経路をそれぞれ木で表したものである．ここで，DT，TRE_1 から得られた SLD 操作可能経路が，OCD_1 の SLD 説明経路である(B)と等しいことに注目してほしい．この性質は一般に成り立ち，後に証明を示す．

次に，訓練フェーズのみから，EBL 有効性が計算可能なことに関する定理を以下に示す．

《定理 1》 OCD の本体

OCD_i の本体のリテラル数は，E_i の SLD 説明構造中の操作可能点の数に等しい．

〈証 明〉

操作性規準により，OCD_i は E_i の SLD 説明構造を一般化したものにおいて，その根の述語を頭とし，それ以外の操作可能節点の述語の連言を本体とする規則節である．よって，定理 1 は明らかである． □

《定理 2》 SLD 操作可能経路

PS(E_i, OCD_i) の SLD 説明経路の要素数は，PS(E_i, DT) の SLD 操作可能経路の要素数に等しい．ただし，PS(E_i, OCD_i) は，$\{OCD_i\} \cup FCS \cup E_i$ で，$\leftarrow T_i$ を目標節とし，OCD_i による導出から始まる SLD 導出手続きとする．FCS は，3•1 節の入力(8)参照．

〈証 明〉

OCD_i の本体は，E_i の SLD 説明構造から，操作可能節のみを取り出したものだから，明らかに PS(E_i, OCD_i) の SLD 説明経路は，PS(E_i, DT) の SLD 説明経路の部分系列である．よって，PS(E_i, DT) の SLD 説明経路から，どの部分を取り除けば，PS(E_i, OCD_i) の SLD 説明経路になるのかを考える．

まず，OCD_i は，操作可能節のみを取り出したものだから，SLD 説明構造中の根以外の操作不可能節（Fig. 1(A)の b, e)とそれらを祖先とする部分木(Fig. 1(A)の c, d)が取り除かれる．次に，PS(E_i, OCD_i) で用いられる規則節は，最初の導出に用いられる OCD_i ただ一つなので，根以外の操作不可能節での導出は必ず失敗する．よって，その操作不可能節の子孫節からなる部分木 (Fig. 1(A)の h, i) が取り除かれ，残った木が PS(E_i, OCD_i) の SLD 説明経路である．定義 5 より，残った木は，枝のラベル以外は，PS(E_i, DT) の SLD 操作可能経路の木と同じであり，よってそれらの節点系列の要素数は等しい． □

定理 1 は，E_i の説明構造から OCD_i の本体のリテラル数が算出可能なことを意味し，定理 2 は，PS(E_i, OCD_i) の探索空間の大きさが，PS(E_i, DT) の SLD 操

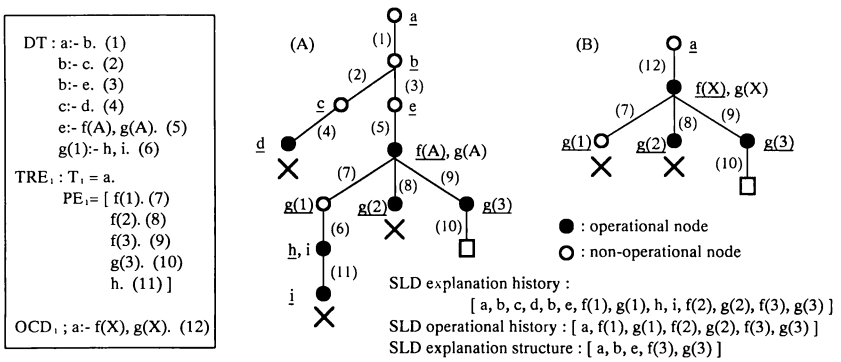


Fig. 1 SLD history and SLD operational history.

作可能経路から求まることを意味する．

5．いくつかの仮定と計算コストの設定

本章では，解析の扱いやすさのための仮定とPrologの計算コストの設定を行う．

5・1 例に関する仮定

一般に，EBL有効性の実験的検証では，訓練例と試験例があらかじめ分類されており，さらに訓練例は学習効率が良くなるように選択されている[3]．よって，訓練フェーズでは冗長な訓練例は与えられないとし，以下の仮定を設ける．

（仮定1） 訓練例系列の非冗長性

訓練例 $[TRE_1, \cdots, TRE_m]$ からは，m 個の異なった概念記述が得られる．

次に，実際に試験例を与えず，EBL有効性を計算するためには，試験例に対する何らかの分布を仮定する必要がある．従来は，パフォーマンスの評価の際，試験例をランダムに与えていた[3]が，ここでは種々の分布に対応できるように，次の（仮定2）を設定する．

（仮定2） 試験例の分布

試験例系列中のある試験例が，$OCD_i\ (1 \leq i \leq m)$ の肯定例である確率は，離散型確率分布 P_i に従う．

（ただし，$\sum_{i=1}^{m} P_i = 1$）

5・2 Prologの計算コストの設定

Prologの計算コストの評価法には，確立されたものはないが[14][15]，計算速度の評価基準としては，LIPS (Logical Inferences Per Second)[16]が，広く用いられている．そこで本研究では，Prologの計算コストとして，LIPSのLI (Logical Inferences) を用いることにする．LIは，推論の回数，正確にはSLD導出の回数のことであり，実際の処理系においては，述語の呼出し回数に対応する[14]．よって，あるSLD説明構造を見つけるまでのLIは，SLD説明経路の要素数に等しい．

ただし，LIでは，バックトラックや単一化のコストが無視されている[14][15]ため，インデキシングしにくい節や引数の数が爆発的に増加する場合には誤差が大きくなるが，EBLではそのような状態は避けられるので，LIの採用は妥当と考える．

6．有限試験例系列の有効性計算

本章では，有限個の試験例の系列に対するEBL有効性の計算法を示す．この方法では，訓練フェーズで得られるSLD説明経路から，試験フェーズなしにEBL有効性が算出可能である．ただし，最悪ケース解析を用いているため，実際に求められるのは有効性関数Uの下限を表す関数LUである．

SEP/SNPの入出力，アルゴリズム，基本概念などから，SEPとSNPそれぞれの計算コストLCとNLC，および関数LUは，以下の式で表される．式の導出の詳細は，**付録**を参照（LC，NLCの単位は，LI）．

$$LC \leq \sum_{i=1}^{m} \{(m-i)ON_i + LI(P\text{-}EBG(TRE_i)) + 1\} + n\sum_{i=1}^{m} P_i \sum_{j=1}^{i} ON_j \quad (1)$$

$$NLC = \sum_{i=1}^{m} (1 + nP_i) N_i \quad (2)$$

$$LU = 1 - \frac{\sum_{i=1}^{m} \{(m-i)ON_i + LI(P\text{-}EBG(TRE_i)) + 1\} + n\sum_{i=1}^{m} P_i \sum_{j=1}^{i} ON_j}{\sum_{i=1}^{m} (1 + nP_i) N_i} \quad (3)$$

ただし，

- $LI(P)$：手続きPにかかるLIを返す関数で，分配結合法則が成り立つ．
- ON_i：$PS(TRE_i, DT)$ におけるSLD操作可能経路の要素数．定理2より，ON_i は，$PS(TRE_i, OCD_i)$ のSLD説明経路の要素数に等しい．
- N_i：$PS(TRE_i, DT)$ におけるSLD説明経路の要素数．

LCの式中に不等号があるのは，**付録**を参照するとわかるように，最悪ケース解析を行っているためである．

また，LUはUの下限であるため，$LU > 0$ がEBL有効の条件となる．

上記のLUは，m，n，P_i，N_i，ON_i，$LI(P\text{-}EBG(TRE_i))$ を変数とする関数となっているが，このうち m，N_i，ON_i はEBLの入力として与えられるGC，DT，TRE_i，OCを用いた $PS(TRE_i, DT)$ の実行結果をトレースすることにより求められる．また，ここでは P_i も与えられるとし，$LI(P\text{-}EBG(TRE_i))$ は実験的に求めるので，結局LUは試験例数 n を変数とする関数になる．ここで，EBL有効性計算の入力/出力および処理をまとめると，以下のようになる．

〔1〕 入 力

GC：目標概念

DT：領域理論

$[TRE_1, \cdots, TRE_m]$：訓練例系列

P_i：試験例の確率分布

〔2〕 出　力

試験例数 n を変数とする関数 LU(n)

〔3〕 処　理

(1) 訓練例系列 [$TRE_1, \cdots, TRE_m$] と DT∪GC を用いて，目標概念 GC を Prolog で証明する．この処理は，PS(TRE_i, DT) に対応．

(2) (1)で生成される SLD 説明経路から，その要素数である N_i と SLD 操作可能経路の要素数である ON_i を得る．

(3) LI カウントプログラムで，LI(P-EBG(TRE_i)) を求める．

(4) 得られた LI(P-EBG(E_i)), ON_i, N_i を式(3)に代入して，EBL 有効性の下限の関数 LU(n) が出力として得られる．

ただし，上記(1)〜(4)において，$1 \leqq i \leqq m$.

7. 無限試験例系列の有効性計算

従来，EBL 有効性は，試験フェーズだけの CPU 時間等から求められるのが一般的である[3]．これに対し，学習フェーズの計算コストも考慮すべきであるというのが本論文の主張の一つであるが，本章ではこの従来の有効性が，関数 U の特殊な場合として，より少ない変数で計算できることを示す．

その特殊な場合とは，試験例が無限個与えられる場合であり，試験例数 n を無限大にすることに対応する．よって，関数 U において $n \to \infty$ とした極限値から，無限試験例系列の EBL 有効性関数：IU を求めることができる．試験例数に関する情報がない状況では，この IU が EBL 有効性の基準になると考えられる．関数 U の場合と同様に，IU についても，下限の関数 LIU が下式のように導出される（導出過程は付録参照）．

$$LIU = 1 - \frac{\sum_{i=1}^{m} P_i \sum_{j=1}^{i} ON_j}{\sum_{i=1}^{m} P_i N_i} \qquad (4)$$

この LIU は，U の訓練フェーズのコストを無視したもので，従来の EBL 有効性の下限を表している．さらに，LIU は GE(TRE_i) と G(ES_i) を変数として持たないという好ましい性質により，LU のように PROLOG-EBG の計算コストを実験的に求める必要がなくなる．LIU における n 以外の変数である m, ON_i, N_i はすべて領域理論で訓練例を証明するだけで得られる．

ここで注意すべきことは，LU/LIU は，EBL 有効性関数の下限を表していることである．このことは，たとえ LU/LIU の値が負でも，実際には EBL が有効になる可能性があることを意味する．よって，LU/LIU の扱い方としては，それが正の値をとれば，EBL 有効性は必ず保障されるという，十分条件として扱うべきである．

また，GCH や本体の異なる目標概念が複数個与えられても，LU/LIU は対応可能である．さらに，領域理論が再帰規則を含んでいてもかまわない．再帰規則を含む場合の具体例は，次節で示される．

8. 具　体　例

8・1 safe_to_stack (X, Y) の例

Mitchell らの論文[1] の safe_to_stack (X, Y) の例で，LU と LIU の計算を行った．Fig. 2 に safe_to_stack (X, Y) の例における EBL の入出力，SLD 説明構造，SLD 説明経路，SLD 操作可能経路を示す．ここでは，TRE_1, TRE_2 の二つの訓練例を与えたが，それぞれの SLD 説明経路が図中の(A)，(B)である．これらから，$N_1=11$, $ON_1=8$, $N_2=11$, $ON_2=6$ が求まる．

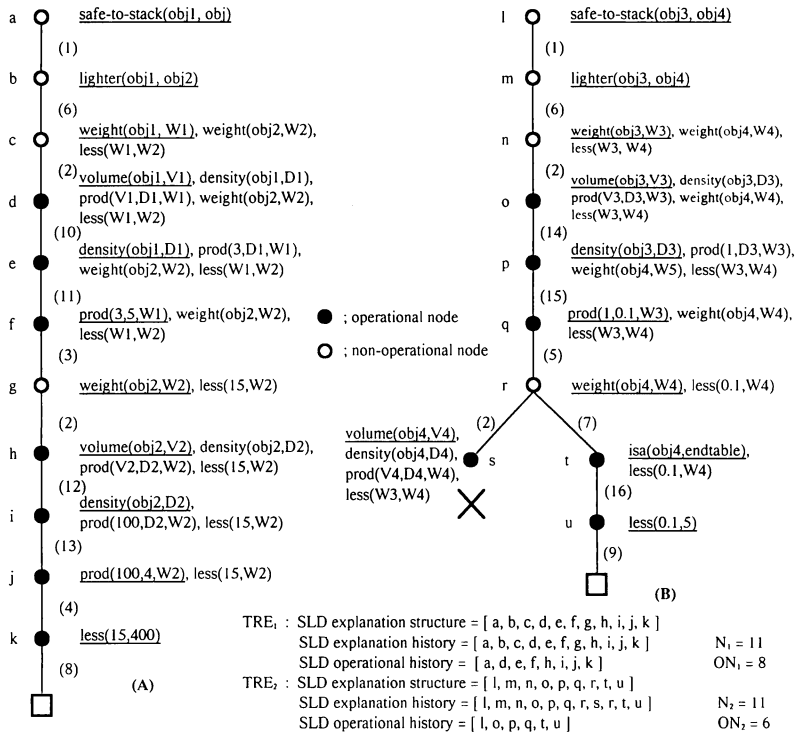
ここでは，訓練例系列が [TRE_1], [TRE_1, TRE_2] の二つ与えられる場合，さらに，[TRE_1, TRE_2] の場合において，確率分布 P_1, P_2 が (P_1, P_2)=(1/2, 1/2), (1/5, 4/5), (4/5, 1/5) の3通りの場合について，LU/LIUを計算した．なお，LI(P-EBG(TRE_1))=74, LI(P-EBG(TRE_2))=93 が，LIカウントプログラムにより実験的に得られた．以上から，算出された LU/LIU を Table 3 に，そのグラフを Fig. 3 に示す．このグラフから，試験例数とともに LU が増加しており，さらに，試験例が 30〜100 個の場合は LI1 だけが，また 100 個以上では LU1 と LU4 が EBL 有効であること等がわかる．また，関数 LU が単調増加している点が興味深い．

8・2 member (X, Y) の例

再帰規則を含む領域における EBL 有効性について，否定的な報告がされている[19]．ここでは，その例として，member (X, Y) のプログラムについて EBL 有効性を計算した．member (X, Y) の領域理論，訓練例，学習された概念記述 OCD，そして算出された LU/LIU を Fig. 4 に示す．ただし，確率分布は，$P_1=P_2=P_3=1/3$ である．また，LU 5 のグラフを Fig. 3 に示す．この結果から，member については，EBL 有効でないことがわかるが，再帰規則を含む領域理論一般については，今後のより厳密な解析が必要であろう．

8・3 具体例の評価

本論文で扱っている EBL の有効性は，単純にいうと，領域理論を用いた問題解決の証明木の葉の数と全



<INPUT>
GC: safe_to_stack(X,Y):- lighter(X,Y). (1)
DT: weight(P1,W):- volume(P1,V1), density(P1,D1), prod(V1,D1,W). (2)
prod(3,5,15). (3) prod(100,4,400). (4) prod(1,0.1,0,.1). (5)
lighter(P1,P2):- weight(P1,W1), weight(P2,W2), less(W1,W2). (6)
weight(P1,5):- isa(P1,endtable). (7) less(15,400). (8) less(0.1,5). (9)
TES: TRE_1: T_1=safe_to_stack(obj1,obj2), PE_1=[volume(obj1,3)(10), density(obj1,5)(11), volume(obj2,100)(12), density(obj2,4)(13)]
TRE_2: T_2=safe_to_stack(obj3,obj4), PE_2=[volume(obj3,1)(14), density(obj3,0.1)(15), isa(obj4,endtable)(16)],
<OUTPUT>
OCD_1 : safe_to_stack(X,Y):- volume(X,V1), density(X,D1), prod(V1,D1,W1), volume(X,V2), density(X,D2), prod(V2,D2,W2), less(W1,W2).
OCD_2 : safe_to_stack(X,Y):- volume(X,V1), density(X,D1), prod(V1,D1,W1), isa(Y,endtable), less(W1,5).

Fig. 2 safe_to_stack (X, Y).

Table 3 LU/LIU.

TES = [TRE_1]	LU1 = $\frac{3n-64}{11n+11}$, LIU1 = 0. 27		
	(P_1,P_2) = (1/2, 1/2)	(P_1,P_2) = (1/5, 4/5)	(P_1,P_2) = (4/5, 1/5)
TES= [TRE_1, TRE_2]	LU2 = $\frac{-155}{11n+22}$, LIU2= 0	LU3 = $\frac{-9n-775}{55n+110}$, LIU3 = −0. 16	LU4 = $\frac{9n-775}{55n+110}$, LIU4= 0. 16

体の節の数で決まる．よって，safe_to_stack，member のような単純な領域理論では，領域理論を用いた問題解決自体の探索空間が広くないため，直観的に EBL が有効でない場合があると考えられるが，8•1 節および 8•2 節の結果はその予測に合致したものとなっている．

ここで取り上げた具体例は単純で，EBL が有効か否かの判別が直観的に可能なものであり，我々の手法が複雑な領域理論でも適用可能なことを示すには，不十分であろう．しかし，確認しておきたいのは，本研究の目的は，単純な EBL 手続きでも十分有効な問題領域を探ることであり，ここでの具体例が EBL 有効で

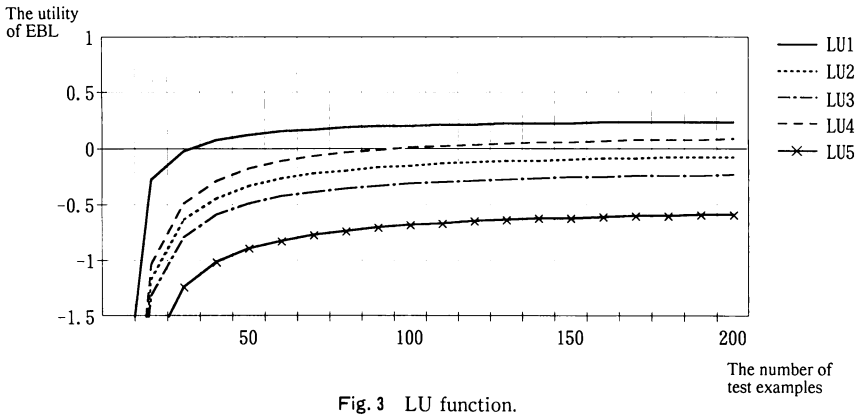


Fig. 3 LU function.

DT : member(X, [X|Y]).
member([X|Y]):- member(Y).
TRE_1 : T_1 = member(a, [a, b, c]), PE_1 = []
TRE_2 : T_2 = member(b, [a, b, c]), PE_2 = []
TRE_3 : T_3 = member(c, [a, b, c]), PE_3 = []
TES = [TRE_1, TRE_2, TRE_3]

OCD_1 : member(A, B):- member(A, B).
OCD_2 : member(A, [_, B]):- member(A, B).
OCD_3 : member(A, [_, _, B]):- member(A, B).

$$LU5 = \frac{-n-37}{2n+6}, \quad LIU5 = -\frac{1}{2}$$

Fig. 4 member (X, Y).

ないことは，本研究にとって何ら否定的な意味を持たないことである．

より複雑な領域理論や例の分布を用いて，EBL 有効のクラスを探ること，有効性関数の解析を行うこと等により本手法の有効性を示すことが今後の課題である．

9. 最悪ケース解析の妥当性

付録の式(1)の導出における FC の算出で，最悪ケース解析を行っているが，FC が正確に求められない原因について以下に述べる．FC は，ある OCD が，否定例（ほかの OCD の肯定例）の証明に失敗する計算コストなので，FC を算出するには，その否定例に対して，OCD の本体のリテラルのどれが偽になるかを判定する必要がある．

SLD 説明経路は，SLD 木のように探索空間全体を表しているのではなく，最初の説明構造が見つかるまでの探索のトレースであるから，前述の判定が SLD 説明経路だけから可能である保証はない．このことを，Fig. 2 の例で説明しよう．この例では，OCD_1 と OCD_2 が学習されるが，OCD_1 の肯定例 TRE_1 に対し OCD_2 の本体のどのリテラルが偽になるかは，二つの SLD 説明経路(A)，(B)からは判定不能である．なぜなら，(A)は，obj の質量計算において，[h, i, j, k] という路で成功するので，(B)の [t, u] に対応する節点を含んでいない．よって，TRE_1 に対し OCD_2 が(B)の [o, p, q] で成功することはわかるが，[t, u] のいずれが偽になるのか，あるいは両方とも真または偽なのかは，SLD 説明経路だけからではわからないのである．この判別を行うには，探索空間全体を表す SLD 木を用いるか，学習後にそれぞれの OCD_i ですべての TRE_j $(i \neq j)$ を解き直さなければならない．いずれにしても，計算コストがかかるし，また実際に一般化を行わねばならず，本手法の解析的側面が失われてしまう．

また，仮に，FC を求めたとしても，それは訓練例に対する FC の値である．訓練例の FC から試験例の FC を求めるには，「訓練例と試験例の FC は同じである」等の厳しすぎる仮定を設けねばならず，望ましくない．

以上の理由で，本研究では，最悪ケース解析を行い，EBL 有効性の十分条件を求めるのみにとどめた．また，本研究の初期段階[5]においては，$LI(PS(E_j, OCD_i))(i \neq j)$ の最悪ケースのコストを，定理 1 を用いて，E_i の SLD 説明構造上の操作可能節点（Fig. 1 では f(X)，g(3)）の個数で表していた．しかし，それでは，途中の経路（Fig. 1 では g(1)，g(2)）が含まれていないため，本論文では，SLD 操作可能経路の要素数 ON_i に変更した．これにより，OCD の本体のリテラルのマッチングにおける探索のコストが有効性計算に反映され，そのマッチングコストが NP-完全である領域理論および訓練例[3]にも対応できると考える．

10. 関 連 研 究

Greiner ら[18]も，EBL の形式的枠組みを提案し，計算コストの設定を行っている．彼らは，ある導出の成功確率を設定し，学習されたルールをどの位置に入れるかについて，いくつかの特殊な場合について分析している．しかし，導出の成功確率を求める具体的方法を与えていないし，計算コストの評価法についても，操作可能節とその他の節のコストを区別しており，本手法のほうがやや緻密さでは劣るが，適切な計算コストの設定をしていると考える．また，LI が Greiner の評価の特殊ケースであることがわかっている[20]．

Flann ら[10]や山村ら[9]は，複数の説明構造に対して有効な操作性規準を提案している．複数の説明構造に対する我々の立場は，その処理を最も単純(あるいは，最悪)にしておいて，EBL 有効の十分条件を満たす領域のクラスを探るというもので，彼らとは研究の目的が異なる．また，コストの評価法について山村らは，バックトラック数とルールの利用度で，学習後のコストのみを評価しており，各導出のコストは捨象されている．よって，本手法のコスト評価のほうが，より現実的と考える．

11. まとめと今後の課題

本論文では，これまで実験的にしか検証されていなかった EBL の有効性を，解析的手法で調べる方法を示した．その結果，大量の試験例を与えなくても，訓練フェーズだけで，EBL の有効性を計算できることを示した．最後に今後の課題として，① LIU による適応 EBL の実現，②確率分布 P_i の統計的推測と関数 LU/LIU の解析，③仮定 1 の検討，などがある．

謝　　辞

Prolog の計算コストについて，JUNET 上で有益なコメントをいただいた近山隆氏 (ICOT)，山崎憲一氏 (NTT)，中島秀之氏 (ETL) に感謝致します．また，御意見をいただいた豊田順一，辻三郎両教授に感謝します．さらに，WOL'90&'91, ICOT KSA-WG, 辻研・豊田研 AI グループミーティングにおいても有益な助言をいただきました．最後に，多くの的確な指摘により，本論文の質を格段に向上させて下さった査読者の方に感謝する次第です．

◇ 参 考 文 献 ◇

(1) Mitchell, T. M., Keller, R. M. and Kedar-Cabelli, R. T.: Explanation-Based Generalization: A Unifying View, *Machine Learning*, Vol. 1, No. 1, pp. 47-80 (1986).
(2) DeJong, G. and Mooney, R.: Explanation-Based Learning: An Alternative View, *Machine Learning*, Vol. 1, No. 2, pp. 145-176 (1986).
(3) Minton, S.: Learning Search Control Knowledge—An Explanation-Based Approach—, Kluwer Academic Publishers, Bonton (1988).
(4) Keller, R. M.: Defining Operationality for Explanation-Based Learning, *AAAI-87*, pp. 482-487 (1987).
(5) 山田, 辻：論理プログラミング環境における EBL の有効性計算, 情処学人工知能研資, AI-90-72 (1990).
(6) Kedar-Cabelli, S. T. and McCarty, L. T.: Explanation-based Generalization as resolution theorem proving, *4th Int. Machine Learning Workshop*, pp. 383-389 (1987).
(7) Hirsh, H.: Explanation-Based Generalization in a Logic Programming Environment, *IJCAI-87*, pp. 221-227 (1987).
(8) Prieditis, A. E. and Mostow, J.: PROLEARN: Towards A Prolog Interpreter that Learns, *AAAI-87*, pp. 494-498 (1987).
(9) 山村, 小林：EBL の複数例題下への拡張, 人工知能学会誌, Vol. 4, No. 4, pp. 389-397 (1989).
(10) Flann, N. S. and Dietterich, T. G.: A Study of Explanation-Based Methods for Inductive Learning, *Machine Learning*, Vol. 4, No. 2, pp. 187-226 (1989).
(11) Shavlik, J. W.: Generalizing the Structure of Explanations in Explanation-Based Learning, Ph. D. Thesis, Department of Computer Science, Univerisity of Illinois, Urbana (1988).
(12) Mooney, R. J.: The Effect of Rule Use on the Utility of Explanation-Based Learning, *IJCAI-89*, pp. 725-730 (1989).
(13) 野口, 滝沢：知識工学基礎論, オーム社 (1986).
(14) 梅村, 山崎：記号処理におけるベンチマーク, 情報処理, Vol. 31, No. 3, pp. 321-327 (1990).
(15) 近山, 山崎, 中島, 平田各氏との JUNET fj. lang. prolog での議論, 5/14-6/7 (1990).
(16) Moto-oka, T.: Challenge for Knowledge Information Processing Systems, *5th Generation Computer Systems*, pp. 3-92 (1981).
(17) Aho, A. V., Hopcroft, J. E. and Ullman, J. D.: アルゴリズムの設計と解析 (野崎・野下共訳), サイエンス社 (1976).
(18) Greiner, R. and Likuski, J.: Incorporating Redundant Learned Rules: A Preliminary Formal Analysis of EBL, *IJCAI-89*, pp. 744-749 (1989).
(19) Letovsky, S.: Operationality Criteria for Recursive Predicates, *AAAI-90*, pp. 936-941 (1990).
(20) Yamada, S.: On the Utility of EBL in Prolog, *Technical Report*, AI-TR-91-6 (1991).

〔担当編集委員：田中穂積，査読者：沼尾正行〕

◇ 付　　録 ◇

1. 式(1), (2), (3)の導出

LC は，まず訓練例 TRE_i $(1 \leqq i \leqq m)$ に対し，それまでに学習された $OCD_1 \sim OCD_{i-1}$ による証明 $PS(TRE_i, OCD_{1 \cdots i-1})$ をまず行うが，(仮定 1)により必ず失敗し，TRE_i それぞれについて，$GE(TRE_i)$, $G(ES_i)$, $M(OCD_i)$ により OCD_i を学習する(下の式(i)の最初の Σ)．その後，学習で得られた $OCD_1 \sim OCD_m$ で $TEE_1 \sim TEE_n$ を証明する (式(i)の二つ目の Σ)．よって，LC は下の式(i)のように表される．なお，以降の式で用いられる LI(P), ON_i, N_i については§6 を参照のこと．

$$LC = \sum_{i=1}^{m} LI(PS(TRE_i,\ OCD_{1 \cdots i-1}) + GE(TRE_i) + G(ES_i) + M(OCD_i)) + \sum_{i=1}^{n} LI(PS(TEE_i,\ OCD_{1 \cdots m})) \quad \text{(i)}$$

式(i)の二つ目の Σ を P_i を用いて展開すると，式(ii)が得られる．

$$= \sum_{i=1}^{m} LI(PS(TRE_i,\ OCD_{1 \cdots i-1}) + GE(TRE_i) + G(ES_i) + M(OCD_i)) + n \sum_{i=1}^{m} P_i LI(PS(TRE_i,\ OCD_{1 \cdots m})) \quad \text{(ii)}$$

次に，式(ii)に以下の操作(a), (b), (c)を施す．

(a) **$LI(PS(TRE_i,\ OCD_{1 \cdots i-1}))$の展開**

TRE_i を証明するときの知識ベースの SLD 木を Fig. 5 に示す．証明にはまず $OCD_1 \sim OCD_{i-1}$ が用いられるが，(仮定 1)によりこの証明はすべて失敗する．そのコストが，式(ii)の $LI(PS(TRE_i, OCD_{1 \cdots i-1}))$ であり，これを FC とする．残念ながら，FC を算出するには，試験例に対して新たな仮定を加える必要があり，ここではあえてその方法をとらず，最悪ケース解析 (worst-case analysis)[17] を行う．FC における最悪ケースとは，$PS(TRE_i, OCD_{1 \cdots i-1})$ において，それぞれの $OCD_k (1 \leqq k \leqq i-1)$ の本体のリテラルをすべて探索する場合，つまり全部調べて最後に失敗する場合である．よって，$LI(PS(TRE_i, OCD_{1 \cdots i-1}))$ は，$OCD_1 \sim OCD_{i-1}$ の SLD 説明経路，つまり $TRE_1 \sim TRE_{i-1}$ の SLD 操作可能経路 (定理 2 より) をすべてたどった場合の節点数となり，$\sum_{i=1}^{i-1} ON_j$ と表される．

ここで，訓練例を DT で証明するだけで，FC が算出可能な点に注目してほしい．なお，最悪ケース解析の妥当性については，§9 で触れている．

(b) **$LI(GE(TRE_i) + G(ES_i) + M(OCD_i))$の展開**

PROLOG-EBG では，説明構造の生成とその一般化が同時に行われるので，$GE(TRE_i)$ と $G(ES_i)$ をまとめて $P\text{-}EBG(TME_i)$ とする．この P-EBG の LI は，SLD 木から解析的に求めるのは困難なので，ここでは LI のカウントプログラムを実際に走らせて，実験的に求めることにする．それから，$M(OCD_i)$ は，OCD_i を assert するだけであり，その LI は 1 でよい．よって，$LI(GE(TRE_i) + G(ES_i) + M(OCD_i)) = LI(P\text{-}EBG(TRE_i)) + 1$ となる．

(c) **$LI(PS(TRE_i,\ OCD_{1 \cdots m}))$の展開**

これは，学習済みの OCD で試験例を証明するコストである．OCD_i の肯定例の証明では，$OCD_1 \sim OCD_{i-1}$ による証明を失敗した後に，OCD_i で成功する．この $OCD_1 \sim OCD_{i-1}$ での失敗分のコストも，(a)の FC と同様に，決

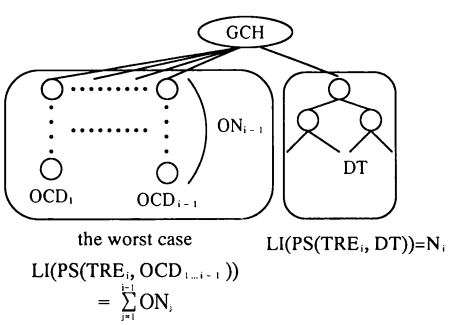


Fig. 5　SLD tree as proving TRE_i.

定できない．よって，ここでも，最悪ケース解析を行う．その結果が，$\sum_{j=1}^{i}\mathrm{ON}_j$ である．

以上の結果，下の式(iii)が得られる．

$$\mathrm{LC}\leq\sum_{i=1}^{m}\{\sum_{j=1}^{i-1}\mathrm{ON}_j+\mathrm{LI}(\mathrm{P\text{-}EBG}(\mathrm{TRE}_i))+1\}+n\sum_{i=1}^{m}\mathrm{P}_i\sum_{j=1}^{i}\mathrm{ON}_j \tag{iii}$$

式(iii)の二つ目の $\sum$ を外して，式(1)が得られる．

$$\mathrm{LC}\leq\sum_{i=1}^{m}\{(m-i)\mathrm{ON}_i+\mathrm{LI}(\mathrm{P\text{-}EBG}(\mathrm{TRE}_i))+1\}+n\sum_{i=1}^{m}\mathrm{P}_i\sum_{j=1}^{i}\mathrm{ON}_j \tag{1}$$

以下 NLC，U の説明は，割愛する．

$$\begin{aligned}\mathrm{NLC}&=\sum_{i=1}^{m}\mathrm{LI}(\mathrm{PS}(\mathrm{TRE}_i,\mathrm{DT}))+\sum_{i=1}^{n}\mathrm{LI}(\mathrm{PS}(\mathrm{TRE}_i,\mathrm{DT}))\\&=\sum_{i=1}^{m}(1+n\mathrm{P}_i)\mathrm{LI}(\mathrm{PS}(\mathrm{TRE}_i,\mathrm{DT}))\\&=\sum_{i=1}^{m}(1+n\mathrm{P}_i)\mathrm{N}_i\end{aligned} \tag{2}$$

$$\begin{aligned}\mathrm{U}&=1-\frac{\mathrm{LC}}{\mathrm{NLC}}\\&=1-\frac{\sum_{i=1}^{m}\mathrm{LI}(\mathrm{PS}(\mathrm{TRE}_i,\mathrm{OCD}_{1\cdots i-1})+\mathrm{GE}(\mathrm{TRE}_i)+\mathrm{G}(\mathrm{ES}_i)+\mathrm{M}(\mathrm{OCD}_i))+n\sum_{i=1}^{m}\mathrm{P}_i\mathrm{LI}(\mathrm{PS}(\mathrm{TRE}_i,\mathrm{OCD}_{1\cdots m}))}{\sum_{i=1}^{m}(1+n\mathrm{P}_i)\mathrm{LI}(\mathrm{PS}(\mathrm{TRE}_i,\mathrm{DT}))}\\&\geq 1-\frac{\sum_{i=1}^{m}\{(m-i)\mathrm{ON}_i+\mathrm{LI}(\mathrm{P\text{-}EBG}(\mathrm{TRE}_i))+1\}+n\sum_{i=1}^{m}\mathrm{P}_i\sum_{j=1}^{i}\mathrm{ON}_j}{\sum_{i=1}^{m}(1+n\mathrm{P}_i)\mathrm{N}_i}\end{aligned}$$

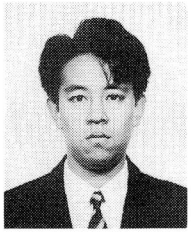
$$\therefore\quad \mathrm{LU}=1-\frac{\sum_{i=1}^{m}\{(m-i)\mathrm{ON}_i+\mathrm{LI}(\mathrm{P\text{-}EBG}(\mathrm{TRE}_i))+1\}+n\sum_{i=1}^{m}\mathrm{P}_i\sum_{j=1}^{i}\mathrm{ON}_j}{\sum_{i=1}^{m}(1+n\mathrm{P}_i)\mathrm{N}_i} \tag{3}$$

2. LIU の導出

$$\begin{aligned}\mathrm{IU}&=\lim_{n\to\infty}\mathrm{U}\\&=1-\frac{\sum_{i=1}^{m}\mathrm{P}_i\mathrm{LI}(\mathrm{PS}(\mathrm{TRE}_i,\mathrm{OCD}_{1\cdots m}))}{\sum_{i=1}^{m}\mathrm{P}_i\mathrm{LI}(\mathrm{PS}(\mathrm{TRE}_i,\mathrm{DT}))}\end{aligned}$$

$$\therefore\quad \mathrm{LIU}=1-\frac{\sum_{i=1}^{m}\mathrm{P}_i\sum_{j=1}^{i}\mathrm{ON}_j}{\sum_{i=1}^{m}\mathrm{P}_i\mathrm{N}_i}\qquad((\mathrm{c})\text{より}) \tag{4}$$

著 者 紹 介



山田 誠二（正会員）

1984 年大阪大学基礎工学部卒業．1989 年同大学院博士課程修了．同年，基礎工学部制御工学科助手．1991 年より大阪大学産業科学研究所講師，現在に至る．工学博士．人工知能，特に，説明に基づく学習，リアクティブプラニング，制約指向プログラミングに関する研究に従事．情報処理学会，日本認知科学会，AAAI 各会員．