論文 / 著書情報
Article / Book Information

題目(和文)	拡張EBLの理論と問題解決への応用
Title(English)	A Theory of an Augmented EBL and its Application to Problem Solving
著者(和文)	山村雅幸
Author(English)	MASAYUKI YAMAMURA
出典(和文)	学位:工学博士, 学位授与機関:東京工業大学, 報告番号:甲第2225号, 授与年月日:1990年3月26日, 学位の種別:課程博士, 審査員:小林重信
Citation(English)	Degree:Doctor of Engineering, Conferring organization: , Report number:甲第2225号, Conferred date:1990/3/26, Degree Type:Course doctor, Examiner:
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

拡張EBLの理論と問題解決への応用

A Theory of an Augmented EBL and its Application to Problem Solving

平成2年3月

東京工業大学 大学院 総合理工学研究科

システム科学専攻

山村 雅幸

指導教官　小林 重信 助教授

目 次

図目次

Chapter 1

序 論

本章では，問題解決システムにおける学習を探索問題としてとらえ，従来のSBLとEBLの基本的枠組みを考察して，それらの統合化の必要性について述べる．

1.1 問題解決システムにおける学習

エキスパートシステムのように，領域に関する知識と演繹的推論に基づいて問題を解決するシステムを考える．このような知識型問題解決システムにおける学習とは，領域に関する知識を蓄積し，問題解決に必要な手続きや技能を習熟してゆく過程である．その前半を知識獲得，後半を知識洗練化と呼ぶ[6]．

本研究では例題からの汎化に基づく逐次的な知識洗練化を取り扱う．特に合成型問題のように初期の教科書的知識だけでは組合せ的に手に負えない領域では，問題解決の経験に応じて“やればやるほどうまくなる”ような逐次的な知識洗練化への強い期待がある．

一般に学習問題は，現在の知識と例題からの汎化に基づいて生成される仮説の候補から，問題解決の能力を向上させる適切な仮説を

探索する探索問題としてとらえることができる[11]．生成される仮説の候補は一般に広大な探索空間を形成する．効果的な探索方法を発見するためには，生成される仮説の空間の構造と仮説の適切さを検査する規範との関係を調べて，探索の制限に役立つ情報を特徴づけることが必要である．

仮説の生成の方法としては，次節に述べる類似性に基づくデータ強調的方法や説明に基づく知識強調的方法など数多くの方法が提案されている．

仮説の適切さを検査する規範には，解の信頼性(正解か否か)，解の最適性(設定された評価規範に対して最適か否か)，問題解決過程の効率性などが考えられる．現在の知識においてどの規範が満足され，獲得される知識においてどの規範が重要視されるかは，学習の進行状況と問題領域に依存する．

知識洗練化においては図1.1に示すような関係が観察される．診断，分類など解析型の問題領域では，問題解決過程の効率性は分類木に代表される知識構造によってすでに達成されていることが多い．ゆえに，学習においては解の信頼性向上が重要視される．一方，計画，設計など合成型の問題領域では，初期の教科書的知識だけでは，信頼できる解を達成するのには十分であるが，最適性を満足する解を求める問題解決過程が組合せ的に手に負えなくなることが多い．ゆえに，学習においては問題解決過程の効率性の向上や解の最適性が重要視される．効率性向上と最適性向上は相反する要請であることが多い．このことは，合成型問題における知識洗練化の困難の1つである．

問題解決システムにおいて効果的な学習を行なわせるためには，

	現在の性能	要請される性能向上
解析型問題 (診断，分類)	? 信頼性 ○ 最適性 ○ 効率性	信頼性の向上
合成型問題 (計画，設計)	○ 信頼性 ? 最適性 ? 効率性	最適性の向上 効率性の向上

図 1.1: 知識洗練化における問題領域と要請される性能の関係

このように多様な仮説の生成方法と多様な規範を同時に考慮できる枠組みが必要である．このような観点からすれば，従来の接近法は知識洗練化の枠組みとしては不適切な面をもつ．そのことを 1.2 および 1.3 節で述べる．

1.2 類似性に基づく学習 (SBL)

類似性に基づく学習 (Similarity-Based Learning : 以下 SBL とよぶ) は多数のデータによって探索をガイドするデータ強調的方法である．SBL の枠組みを図 1.2 に示す [9]．

SBL における仮説の生成方法は，多数のデータに共通する性質を一般化記述言語にしたがって抽出する帰納的方法である．SBL における仮説の適切さを検査する規範は，正例を含み負例を含まないという信頼性のみである．SBL では本質的に現在の知識では説明できない知識が生成される．前節で述べたように，問題解決システムにおける学習にとって仮説の信頼性は必要ではあるが十分ではないことに注意されたい．このような問題点を回避するために従来の SBL 研究の多くは，適切な仮説を生成するための規範を問題固有の経験

⟨ *given* ⟩

1. 具体例記述言語 (I)：具体例を記述する言語.
2. 一般化記述言語 (G)：一般化仮説を記述する言語.
3. 照合述語 ($\in$)：具体例と仮説の照合を調べる述語.
4. 練習例の集合 (T)：具体例とそれが概念の例であるか否かの判定の対の集合.

⟨ *determine* ⟩
次の信頼性条件を満たす一般化仮説 $g \in G$

$$\forall i \in T \quad (positive(i) \rightarrow i \in g) \wedge (negative(i) \rightarrow i \notin g).$$

図 1.2: SBL の枠組み

則として利用している.

仮説の信頼性には，汎化に対する単調性という探索に都合の良い性質がある．信頼性規範を満足する仮説の空間は一般化関係のもとで範囲をなす．これをバージョン空間という [9]．バージョン空間の維持によって探索は経済化され，その収束状況は学習の進行状況の目安となる．バージョン空間の収束は，一般化関係の設定と練習例の量に関係する．一般化関係における帰納的飛躍の程度をバイアスという．バージョン空間は，強いバイアスの下では少数の練習例で素早く収束し，弱いバイアスの下では収束までに大量の練習例を必要とする．練習例のあらゆる部分集合が記述可能な最も弱いバイアスの下では，バージョン空間の上，下限は常に暗記にとどまる.

最近の研究によれば，バージョン空間法は必ずしも経済的な探索方法ではないことが明らかとされている [3]．しかし，生成される仮説の空間と適切さを検査する規範の関係を利用して効果的な探索方

法を発見するという意味において，バージョン空間法は“巧妙な”方法といえる．

期待される帰納的飛躍を引起こす適切なバイアスは各問題に固有である．問題解決システムにおける学習へのバージョン空間法の適用上の困難は，各問題に専用のバイアスを用意しなければならないという非柔軟性にある．また，“オッカムの剃刀”のような経験則によって仮説の生成をガイドする方法にも同様の困難がある．経験則は問題固有の浅い知識であって，適用範囲や意味が不透明だからである．したがって，SBL を問題解決システムにおける学習研究の枠組みとして採用するためには，仮説の適切さを検査する規範を明示的に扱えるよう拡張して，非柔軟性に対処できるようにする必要がある．

1.3 説明に基づく学習 (EBL)

説明に基づく学習 (Explanation-Based Learning : 以下 EBL とよぶ) は単一の練習例と領域知識によって探索をガイドする知識強調的方法である．EBL の枠組みを図 1.3 に示す [11]．

EBL における仮説の生成方法は，領域理論に基づいて与えられた単一の練習例が目標概念の例であることを説明し，その説明構造を一般化してマクロルールとする演繹的方法である．EBL で生成される仮説は本質的に領域理論から演繹される範囲にある．すなわち EBL では，信頼性が保証されている領域理論を用いるとすれば，単一の例から論理的に正しい仮説を生成することができる．この特徴は領域理論の信頼性が保証されているような合成型問題における知

⟨ *given* ⟩

1. 目標概念：操作性規範を満たさない初期記述.
2. 練習例：目標概念の単一の正例.
3. 領域理論：説明を構成するためのルールと事実の集合.
4. 操作性規範：学習される概念記述を記述すべき形態を特定する述語.

⟨ *determine* ⟩
操作性規範を満たす練習例の説明に基づく汎化.

図 1.3: EBL の枠組み

識洗練化にとっては有望な性質である．ただし，当然のことながら，領域理論が信頼性に欠ける場合には誤った仮説を生成する可能性がある (imperfect domain theory problem) .

EBL で特徴的なことは，SBL においてバイアスや経験則という形で陰に組込まれていた仮説の適切さを評価する規範を操作性規範 (operationality critera) としてように設定しうることである．Keller は Meta-LEX の研究において，図 1.4に示すように操作性規範を問題解決システムとの関連において定義した [5]．Keller による操作性規範は，本研究で扱う“仮説の適切さを評価する規範”に対応するものである．以下，仮説の適切さを評価する規範を操作性規範と呼び，操作性規範の定義および操作性と汎化との関係を明確化する問題を操作性問題と呼ぶことにする．

従来の EBL 研究の多くは，より“高度”な汎化の生成法を追及している．この方向の研究としては，BAGGER における n への一般化や [18]，再帰的規則の獲得 [19]，あるいは失敗からの学習 [13]

⟨ *given* ⟩

1. 概念記述：EBL で生成される説明に基づく汎化.
2. 実行システム：性能向上のために概念記述を用いるシステム.
3. 実行目的：要請されるシステムの性能向上の型と範囲.

⟨ *then* ⟩
次の 2 つの条件を満たすならば，概念記述は操作的である.

1. 利用可能性：実行システムで利用可能でなければならない.
2. 効用：実行システムで利用した時に，実行目的に沿った性能向上がなければならない.

図 1.4: Keller の操作性規範

があげられる．しかし，操作性問題の解決がなければ，EBL は前節で指摘した SBL の問題点を共有することになる．なぜなら，操作性規範を考慮しない汎化の充実は探索空間を徒らに拡大することになり，適切な仮説を得るためには不透明な経験則の導入が必要となるからである．

操作性問題に関して，従来さまざまな問題点が指摘されている．通常，操作性規範は“操作的な述語”として与えられるが [11]，これは不適切な表現であることが指摘されている [16]．また，例題から学習によって生成されるマクロが蓄積されるにつれて問題解決の効率が悪化することが指摘されている [8]．このことは，本来要請される操作性規範を実装上の都合で操作的な述語に“翻訳”した結果である．Segre[17] は問題解決システムにおける操作性と汎化のトレードオフの存在を指摘している．Keller は，生成-検査法による解決策を提案しているが，cpu time を測度とするために，操作性の検査に

	SBL	EBL
仮説と 現在の知識	演繹されない	演繹される (正当性の保証)
仮説の複雑さと 例題の量	収束までに 大量のデータ	少ない例題から 複雑な仮説
適切さの 規範	implicit (信頼性のみ)	explicit (操作性規範)
適切さの 保証	バージョン空間法	一般には困難

図 1.5: SBL と EBL の相補的な特徴

は新しい例題毎に過去の例をすべて解決しなおすことが必要である．このように，操作性問題には十分な解決策が与えられていないのが現状である．

操作性問題の解決は一般に困難な課題である．Segre の指摘したように操作性規範を構成する評価測度の間には複雑なトレードオフ関係が存在する．また個々の規範がどのような性質を持つかが明らかにされていない．さらに，従来の EBL の枠組みでは，学習の 1 セッションが単一の例題ごとに区切られているために操作性の一貫した検査を考えにくい．したがって，EBL を問題解決システムにおける学習研究の枠組みとして採用するためには，まず複数の例題を同時に考慮できるように拡張して，操作性問題に柔軟に対処できるようにする必要がある．

1.4 SBL と EBL の統合化の必要性

以上の考察は図 1.5に示すようにまとめることができる．

それぞれの枠組みは相補的な特徴を持ち，単独では問題解決システムにおける学習研究の枠組みとして不適切なことは明らかである．これらを統合化する方向で拡張することによって，より適切な枠組みが得られることが期待される．本研究では，逐次的な知識洗練化の観点からより有望な特徴を持つ方法として，EBL を複数例題上に拡張した拡張 EBL の枠組みを提案する．すなわち，EBL の持つ有望な性質を保存しながら，EBL の課題である操作性問題に対して SBL の相補的な利点を統合化することを試みる．したがって，陽に説明のための知識を持たない操作性問題に対して，バージョン空間法のように生成される仮説の空間と操作性規範の関係を利用したうまい探索方法の発見が期待される．

本論文は全 6 章から構成される．

第 1 章は序論である．

第 2 章では拡張 EBL の概念的枠組を導入し，それを論理プログラム上に形式化して，理論的考察を行なう．

第 3 章では理論的考察に基づく素朴な学習器を構成して，その性質を例示する．

第 4 章ではより高度な応用のためにメタ領域知識と教師への質問を利用する学習器について議論する．

第 5 章ではメタ領域知識と質問を利用した学習器による問題解決マクロテーブルの学習について述べる．

第 6 章は結論である．

Chapter 2

拡張 EBL の理論

本章では，まず前章で述べた方針に基づいて，EBL を複数例題上に拡張した拡張 EBL の概念的枠組を提案する．次に，拡張 EBL を論理プログラム上で形式化して，複数の例題から生成される汎化空間と操作性の関係について理論的に考察する．最後に，操作性の高い汎化を探索するために有用な概念として最小 EBG の概念を提案する．

2.1 EBL の拡張

拡張 EBL の概念的枠組を提案し，理論的考察のためのいくつかの仮定を行なう．

2.1.1 拡張の概念的枠組み

本研究では，前章で述べたように逐次的な知識洗練化を対象とする．そこでは，領域理論のままでは冗長な探索を必要とするが，説明からの汎化に基づくマクロを蓄積して利用することにより手続き的な解決が促進され，探索ができるだけ回避されることが望まれる．

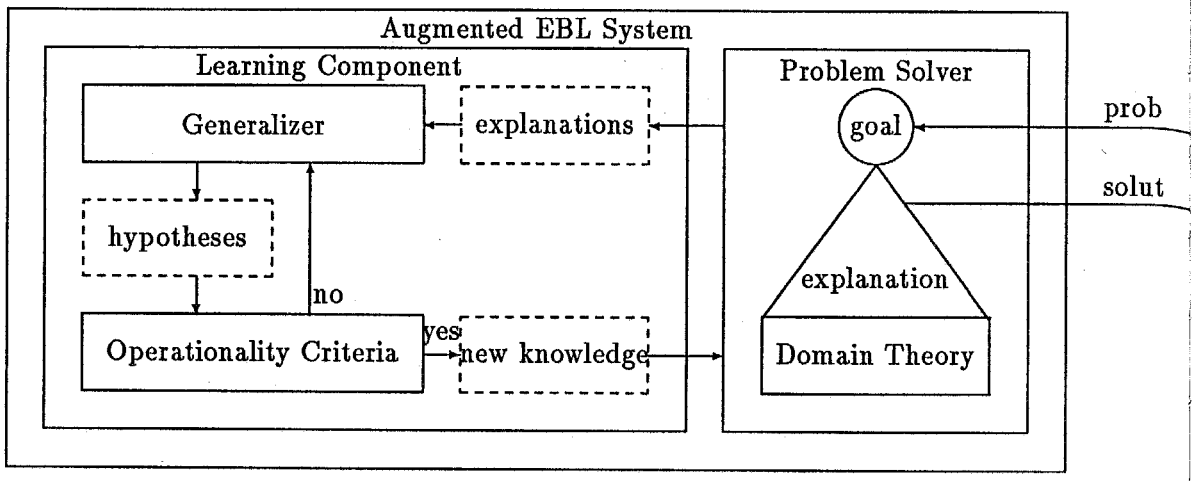


図 2.1: 拡張 EBL の概念的枠組み

このような問題領域において，領域理論のもとで複数の例題を順次与えて，操作性規範を満足する汎化を逐次的に求めることができるように EBL を拡張することを考える．

拡張 EBL の枠組みは生成－検査パラダイムに基づく．その概念図を図 2.1に示す．問題解決器は領域理論と事実から記号的推論によって与えられた目標（問題）を解決し，説明（証明）を解答する．汎化生成器は問題解決器によって生成された複数の説明から汎化を生成する．操作性規範は生成された汎化の適切さを検査する．

2.1.2 仮定

理論的考察に必要ないくつかの仮定を行なう．

まず，記述言語として論理プログラムを採用する．領域理論は確定節の有限集合からなる．具体例の観察事実は単位節の有限集合からなる．問題は目標節として与えられる．プログラムは pure-Prolog

によって実行される.

つぎに，操作性規範を仮定する．問題解決システムにおける知識洗練化にとって有用なマクロテーブル [7] を獲得するという観点から，つぎの 2 つの尺度を設定する.

1. 利用度最大化：マクロの数はできるだけ少なくかつできるだけ多く利用されることが望ましい.

2. 後戻り最小化：獲得されたマクロテーブル上の探索コストができるだけ少なく過去の問題を解決できることが望ましい.

2.2 説明構造とその汎化

論理プログラム上に説明構造とその汎化を定義する.

定義 2.1 (説明構造)
A と A'を同じ述語記号をもつ正リテラル，$A' \leftarrow B_1 \wedge \cdots \wedge B_n$を領域理論の確定節とするとき，次のように再帰的に定義される式を説明構造という.

1. $A : A'$.

2. $A : (A' \leftarrow \langle B_1 \rangle \wedge \cdots \wedge \langle B_n \rangle)$.

ここで，$\langle B_i \rangle$ はリテラル B_iを根とする説明構造，":" の両辺は変数を共有しないとする．説明構造の根とは，説明構造を木構造としてみたときの根で，上式における A をさす．また，木構造の末端のうち *1* におけるリテラル A'を葉という. ■

$trav(true).$
$trav(A : A).$
$trav(A : (A \leftarrow B)) \ :- \ trav(B).$
$trav(A \wedge B) \ :- \ trav(A), \ trav(B).$

図 2.2: 説明構造の最左優先訪問プログラム

論理プログラムにおける入力導出は，各入力節が導出されるリテラルで結合された木構造をなす．定義 1 の a) は最後の導出形のリテラルの具体化に，b) は入力導出に対応する．任意の入力導出から説明構造を構成することができる．1 つの説明構造には，導出されるリテラルの選択順序によって複数の入力導出が対応する．図 2.2は説明構造を再左優先訪問して 1 つの入力導出を得るプログラムである．簡単のために，事実 $F \leftarrow$ は $F \leftarrow$ true の形式で記述している．訪問の結果得られる具体化を最終具体化という．

図 2.3の領域理論 D，例題 E_1，目標 G_1が与えられたとき，G_1に対する説明構造 t_1を図 2.4に示す [8]．t_1には葉がないが，これは t_1 が証明であって，最終的な導出形が□であることに対応する．

定義 2.2 (説明構造の汎化)

1. 説明構造 t の根を A，葉を $B_1, \cdots, B_n$ とするとき，これらを次の A'，$B'_1, \cdots, B'_n$ で置き換える操作を変数汎化 *(uninstantiation)* という．

$$(A', B'_1, \cdots, B'_n)\sigma = (A, B_1, \cdots, B_n) .$$

$$
\begin{aligned}
D &= \{kill(X,Y) \leftarrow hate(X,Y) \wedge possess(X,Z) \wedge weapon(Z),\\
&\quad hate(W,W) \leftarrow depressed(W),\\
&\quad possess(U,V) \leftarrow buy(U,V),\\
&\quad weapon(gun),\\
&\quad weapon(knife)\}.
\end{aligned}
$$

$$
\begin{aligned}
E_1 &= \{depressed(john),\ buy(john,gun)\},\\
G_1 &= \leftarrow kill(john,john).
\end{aligned}
$$

$$
\begin{aligned}
E_2 &= \{depressed(tom),\ buy(tom,knife)\},\\
G_2 &= \leftarrow kill(tom,tom).
\end{aligned}
$$

図 2.3: 領域理論と 2 つの例題

ここで，σはある代入である．

2. 説明構造 t の部分構造 $R:X$の最終具体化をθとするとき，Xを $R\theta$に置き換える操作を構造汎化 *(unresolution)* という．構造汎化された説明構造の最終具体化をλとするとき，$A\lambda:X$を汎化の剰余 *(remainder)* という．

   説明構造t'がtから構造汎化と変数汎化によって得られるとき，t'をtの汎化といい,$t' \preceq t$と書く． ■

変数汎化は具体化の逆操作に，構造汎化は入力導出の逆操作に対応する．Mitchell の目標回帰アルゴリズム [2] では，目標に現われる定数を変数に置き換えて“結論”とし，事実との導出を切断して“前提”とすることによりマクロを生成している．マクロの前提と結論に現われる変数の関係は説明構造を回帰して求めている．定義 2 は

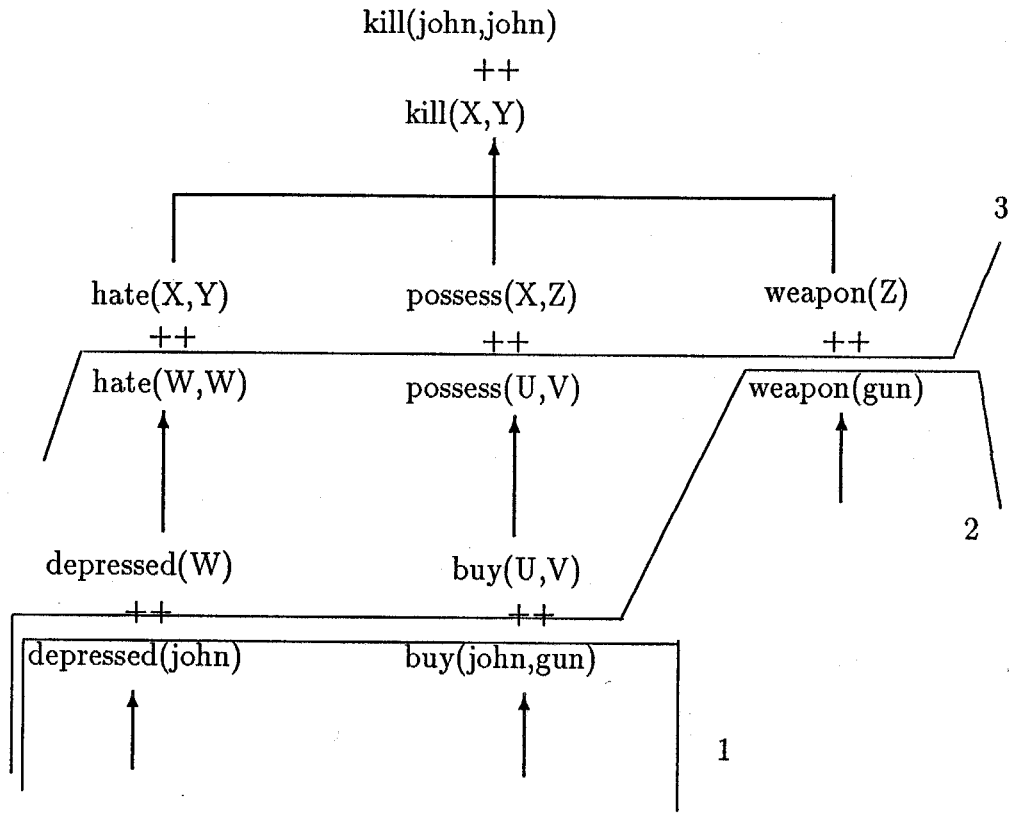


図 2.4: 例題 E_1の説明構造

これらの操作を一般化したものである．説明構造の汎化は一般に複数存在し，それらは半順序関係をなす．

定義 2.3 (マクロ)

説明構造 t の根を A，葉を $B_1,\cdots,B_n$，最終具体化をθとする．このとき，

1. 次の確定節を t に基づくマクロという．

$$(A \leftarrow B_1 \wedge \cdots \wedge B_n)\theta.$$

2. n-項述語 p に対して，$\pm p(X_1,\cdots,X_n)$ (±は符号，X_iはことなる変数)を具体化されていないリテラルという．A と $B_1,\cdots,B_n$ を具体化されていないリテラルで置き換える変数汎化に基づくマクロを原始マクロという． ■

説明構造 t の根 A と葉 $B_1,\cdots,B_n$の最終具体化は，t の原始マクロと式 $A \leftarrow B_1 \wedge \cdots \wedge B_n$の最汎単一化として求められる．

定義 2.4 (基本汎化と EBG)

領域理論 Dと例題 E上の説明構造 t に対して，Eの事実との導出からなるすべての部分構造 $F:(F' \leftarrow)$ を $F:F'$に置き換える構造汎化を t の Dに対する基本汎化といい，t/Dと書く．$t' \preceq t/D$なる t'を t の EBG という． ■

基本汎化とは，各例題に固有の事実との導出を構造汎化することで，マクロが領域理論の定理となるための必要条件である．

図2.5は，図2.4のt_1の汎化に基づくマクロの例である．m_1は基本汎化より特殊である．E_1の事実 $(depressed(john), buy(john, gun))$

$$
\begin{aligned}
m_1 &= kill(john, john). \\
m_2 &= kill(john, john) \leftarrow depressed(john) \wedge buy(john, gun). \\
m_3 &= kill(W, W) \leftarrow depressed(W) \wedge buy(W, gun). \\
m_4 &= kill(john, john) \leftarrow depressed(john) \wedge buy(john, gun) \wedge weapon(gun). \\
m_5 &= kill(W, W) \leftarrow depressed(W) \wedge buy(W, Z) \wedge weapon(Z). \\
m_6 &= kill(X, Y) \leftarrow hate(X, Y) \wedge possess(X, Z) \wedge weapon(Z).
\end{aligned}
$$

図 2.5: t_1から得られるマクロの例

との導出を含んでいるため，m_1を他の例題に適用することは妥当でない．m_2は基本汎化 (図 2.4実線 1)，m_3は m_2の原始マクロである．m_4は基本汎化をさらに構造汎化したもので (同 2)，m_5は m_4の原始マクロである．m_6は領域理論の節にまで汎化が及んだものである (同 3)．このように 1 つの説明構造に対してさまざまなレベルの汎化が存在することに注意されたい．

拡張 EBL では複数の例題の説明構造に対する汎化を取扱う必要があるので，これを次に定義する．

定義 2.5 (説明構造集合上の汎化，マクロテーブル)

1. 説明構造集合Tの *1* つの説明構造tの汎化をt_0，剰余を$t_1, \cdots, t_n$とするとき，次のT'をTの直接の汎化という．

$$T' = (T - \{t\}) \cup \{t_0, \cdots, t_n\}.$$

ここで，集合の和において最終具体化が同一の説明構造と，葉と根のみからなる説明構造は取除かれるものとする．Tから直

$$
\begin{aligned}
M_1 &= [kill(john, john) \leftarrow depressed(john) \wedge buy(john, gun),\\
&\quad kill(tom, tom) \leftarrow depressed(tom) \wedge buy(tom, knife)].\\
M_2 &= [kill(W1, W1) \leftarrow depressed(W1) \wedge buy(W1, gun),\\
&\quad kill(W2, W2) \leftarrow depressed(W2) \wedge buy(W2, knife)].\\
M_3 &= [kill(W, W) \leftarrow depressed(W) \wedge buy(W, Z) \wedge weapon(Z),\\
&\quad weapon(gun),\\
&\quad weapon(knife)].
\end{aligned}
$$

図 2.6: 説明構造集合の汎化に基づくマクロテーブル

接の汎化を繰返してT''が得られるとき，T''をTの汎化といい，$T'' \preceq T$と書く．TからT''に至る直接の汎化の列を汎化経路という．Tの汎化の全体をTの汎化空間といい$\Gamma(T)$で表す．

2. 説明構造集合Tの各説明構造に基づくマクロを適当に順序づけた列を，Tに基づくマクロテーブルという．Tのマクロテーブルの全体を$M(T)$で表す． ■

図2.3の例題E_2は武器がナイフである点がE_1と異なる．E_2の目標G_2に対する説明構造をt_2として，説明構造集合$\{t_1, t_2\}$の汎化に基づくマクロテーブルの例を図2.6に示す．M_1は基本汎化のみからなる．M_2はM_1から変数汎化したもの，M_3はM_2から構造汎化した結果，同一の説明構造を生じてマクロがまとまったものである．

2.3 汎化空間の構造

操作性規範をpure-Prolog上に形式化して，汎化空間と操作性規範の関係について考察する．

定義 2.6 (操作性規範)

1. 説明構造集合 T_0から T_nへの汎化経路を$\pi_n = [T_0, \cdots, T_n]$, T_0から T_iへの部分経路をπ_iとする. このとき, 次のδを汎化による説明構造の重複度という.

   (a) $\delta(t, \pi_0) = 1$ $(t \in T_0)$.

   (b) T_iから T_{i+1}への直接の汎化において, T_iの t の汎化を t_0, 剰余を $t_1, \cdots, t_n$とする.

   i. t_jと同一の t'が T_iに存在するとき,

   $$\delta(t_j, \pi_{i+1}) = \delta(t, \pi_i) + \delta(t', \pi_i) .$$

   ii. それ以外の t_jについて,

   $$\delta(t_j, \pi_{i+1}) = \delta(t, \pi_i) .$$

   iii. 汎化と無関係な t''について,

   $$\delta(t'', \pi_{i+1}) = \delta(t'', \pi_i) .$$

   また, 次の Uを T'の利用度 *(usage degree)* という.

   $$U(T_n, \pi_n) = \sum_{t \in T_n} |t| \cdot (\delta(t, \pi_n) - 1) .$$

   ここで, $|t|$ は t に含まれるルールの数である.

2. Tの汎化 T'に対して, T'に基づくマクロテーブル Mを領域理論の前に挿入して, *pure-Prolog* によって Tのすべての説明構造

を再構成するのに必要な後戻りの総数を$\beta(M,T)$とする．このとき，次のBをT'の後戻り回数 *(backtracking number)* という．

$$B(T',T) = \min_{M \in M(T')} \beta(M,T) .$$

Tの汎化T'とT''に対して次の条件が成立つとき，T'はT''よりも操作的であるという．

$$U(T',[T,\cdots,T']) \geq U(T'',[T,\cdots,T'']) ,$$

$$B(T',T) \geq B(T'',T) .$$

■

一般にTからT'には複数の汎化経路が可能であるが，重複度および利用度は汎化経路によらない．以下，$U(T',[T,\cdots,T'])$を$U(T',T)$と略記する．

利用度は汎化するほど増加するので，利用度最大化のためにはより汎化することが好ましい．説明構造を単に分解することだけで利用度が増加することのないように，2回以上利用された回数をマクロの大きさによって重み付けしている．一方，後戻り回数も汎化するほど増加するので，後戻り最小化のためには過剰に汎化しないことが好ましい．このように，操作性規範を構成する尺度はトレードオフの関係にある．

図2.6において，$M_1 \sim M_3$の利用度は，0，0，3，後戻り回数は，0，1，0である．したがって，M_3は後戻りがなくかつ利用度が最も高い“良い”マクロテーブルといえる．

説明構造集合の汎化と利用度，後戻り回数には次の関係が成立する．

定理 2.1 (汎化空間の構造)

3 つの説明構造集合 $T' \preceq T'' \preceq T$ において,

1. $U(T',T) \geq U(T'',T)$.
2. $U(T',T) = U(T'',T)$ ならば $B(T',T) \geq B(T'',T)$.

証明 利用度については明らか. T'に基づくマクロテーブルから, 後戻り回数が同一かまたはより少ない T''に基づくマクロテーブルが構成できることを示す. 直接の汎化について示せば十分である.

1. T''からT'への直接の汎化が変数汎化の場合, 利用度が同一であるから, 同一の説明構造は生じない. したがって, 同じ配列のマクロテーブルは同一かまたはより少ない後戻り回数をもつ.
2. 構造汎化の場合, T'における構造汎化と剰余を T''の対応する説明構造と置き換えればよい. ■

操作性の比較の観点から模式的に見た汎化空間の構造を図 2.7に示す. 定理 2.1は, 利用度は汎化に対して単調増加するが, 後戻り回数は汎化空間のうち利用度同一の部分空間においてのみ単調増加することを意味する. 後戻り回数が汎化空間全体に対して単調増加しないことは, 次のように説明される. 汎化によって頭部は同一だが本体が少しずつ異なる複数のマクロが生成され, それが原因で後戻りが起こったとする. このとき, さらに汎化を重ねることによってこれらが同一となり利用度が増加すると, 後戻りの原因が失われて後戻り回数は減少する. 一方, 操作性の高い汎化を獲得するという観点からは, 前者は“中途半端な”汎化である. したがって, “利用

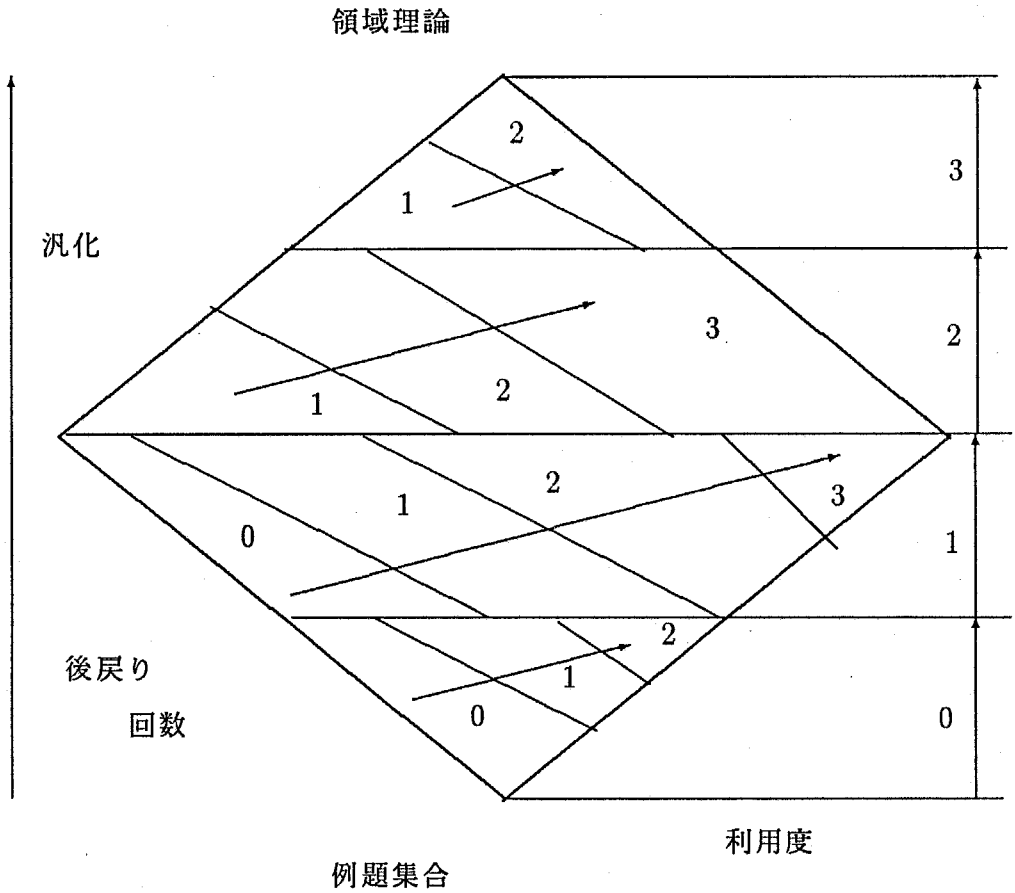


図 2.7: 汎化空間の構造

度が増加する直後" すなわち利用度同一の部分空間における極小元が有用であると考えられる．以下，Tの汎化空間における利用度 n の部分空間を n-利用度部分空間 (n-usage subspace) といい，$\Gamma(T,n)$ で表わす．

2.4 最小 EBG に基づく汎化の生成

前節で得られた汎化空間の構造を考慮して，操作性の高い汎化の探索に有用な概念として最小 EBG の概念を導入する．

2.4.1 最小 EBG

操作性の高い汎化を逐次的に生成するためには，ある利用度の部分空間において極小元の逐次的更新が容易に行えることが必要である．そのような汎化として，複数の説明構造の最小 EBG の概念を導入する．

定義 2.7 (最小 EBG)

$t_1,\cdots,t_n$を根の述語記号を共有する *EBG* とするとき，$t \preceq t_1,\cdots,t \preceq t_n$なる t を共通 *EBG* という．$t_1,\cdots,t_n$のすべての共通 *EBG* t'に対して$t' \preceq t$ となる共通 *EBG* t を $t_1,\cdots,t_n$の最小 *EBG* という． ■

最小 EBG の存在を示すには次の定理が必要である．

定理 2.2 (最汎単一化と最小汎化の可換性)

ある式の 3 つの具体化 p_1, p_2, qに対して，p_iと qの最汎単一化を p_i'，p_1と p_2の最小汎化 [9] を p，p_1'と p_2'の最小汎化を p'，p と qの最汎単一化を p^*とするとき，p'と p^*は変数の書き換えを除いて同一である．

証明 p'がp^*の具体化であることを示し，さらにそれが変数の書き換えであることを示す．以下，xがyの具体化であることをyはxの汎化であるともいう．

1. pとqはともにp_1'とp_2'に共通の汎化である．p'はp_1'とp_2'の最小汎化であるから，p'はpとqに共通の具体化(単一化)である．一方，p^*はpとqの最汎単一化であるから，p'はp^*の具体化でなければならない．

2. p^*からp'への具体化が，τ/Xなる代入を含むとする．Xはpとqの同じ位置に現われる変数に，τはp_i'のその位置に現われる項τ_iの最小汎化に対応する．このとき，pではその位置に変数が現われるので，p_iのその位置にτ_iが現われる．pはp_iの最小汎化であるからτは変数でなければならない． ■

図2.8に定理の意味を概念的に示す．

定理 2.3 (2 つの EBG に対する最小 EBG の存在)
根の述語記号を共有する2つのEBG t_1とt_2に対して，最小EBG tが存在する．t_1, t_2を与えたとき，図2.9のプログラムはtを出力する．

証明 プログラムの出力をt，t_1とt_2のある共通 EBG をt'とする．説明構造の最終具体化が原始マクロと根および葉から最汎単一化によって得られることと補題を利用して，$t' \preceq t$を示す．

1. t_1, t_2, tとt'を同時に訪問するようにプログラムをトレースする．t'に現われるルールはt_1とt_2の同じ位置にも現われる．こ

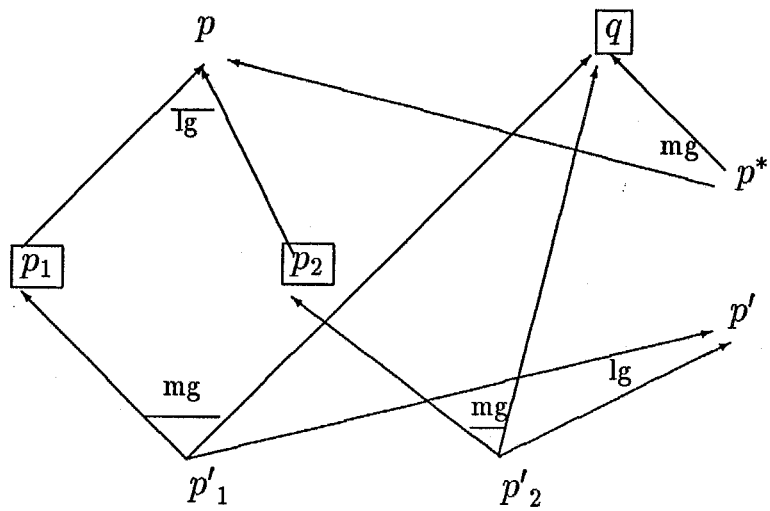


図 2.8: 最汎単一化と最小汎化の可換性

$lebg(true, true, true).$
$lebg(R:L1, R:L2, R:L) :-$
　　$lg(L1, L2, L).$
$lebg(R:(A \leftarrow B1), R:(A \leftarrow B2), R:(A \leftarrow B)) :-$
　　$lebg(B1, B2, B).$
$lebg(A1 \wedge B1, A2 \wedge B2, A \wedge B) :-$
　　$lebg(A1, A2, A),$
　　$lebg(B1, B2, B).$
$lebg(R:T1, R:T2, R:L) :-$
　　$copy(R, L1), trav(L1, T1),$
　　$copy(R, L2), trav(L2, T2),$
　　$lg(L1, L2, L).$

図 2.9: 2 つの説明構造の最小 EBG の計算

のとき，プログラムの第 3 節が照合するので，このルールは t の同じ位置にも現われなければならない．ゆえに，t'の木構造は t と同じかまたは t より小さい．

2. 木構造が同一である場合，t の変数汎化は t_iの最小変数汎化であるので t'は t の変数汎化である．t'の木構造が t より小さい場合，t をそこで構造汎化した説明構造を t^*として，t'が t^*の変数汎化であることを示す．

　構造汎化は最終具体化に影響しないので，t_1と t_2は t と同じ構造としてよい．1 回の構造汎化について示せば十分である．t の部分構造 $A:X$の構造汎化によってt^*が得られるとして，$A:X$の葉を $B_1, \cdots, B_n$とすると，t^*の対応する部分の葉は $A:X$の原始マクロ qと $p = A \leftarrow B_1 \wedge \cdots \wedge B_n$との最汎単一化 p^*から求め

られる．同様に，t_iの同じ位置からはじまる部分構造 $A:X_i$の構造汎化によってt_i'が得られるとして，$A:X_i$の葉を $B_{i1},\cdots,B_{in}$とすると，t_i'の対応する部分の葉はqと $p_i = A \leftarrow B_{i1}\wedge\cdots\wedge B_{in}$との最汎単一化$p_i'$から求められる．補題により，$p^*$は p_i'の最小汎化である．t'の対応する部分の葉はp_i'の共通変数汎化から求められるから，t'はt^*の変数汎化である． ■

複数の EBG からの最小 EBG の逐次的計算に関して次の性質が成立する．

定理 2.4 (最小 EBG の逐次的計算)
n 個の EBG $t_1,\cdots,t_n$に対して，$t_1,\cdots,t_k$の最小 EBG を t'，$t_{k+1},\cdots,t_n$の最小 EBG をt''とすると，t'とt''の最小 EBG t は，$t_1,\cdots,t_n$の最小 EBG である．

証明 $t_1,\cdots,t_n$のある共通汎化をt^*とすると，t^*は$t_1,\cdots,t_k$の共通汎化でもあるから，仮定により$t^* \preceq t'$．同様に$t^* \preceq t''$．ゆえに，定理 2.3 により，$t^* \preceq t$． ■

定理 2.3 と 2.4 は，複数の例題の EBG には最小 EBG が存在し，それはいちいち計算しなおすことなく逐次的に求めることができることを意味する．このように最小 EBG は操作性の高い汎化の候補として有用であると考えられる．

図 2.6の M_3は最小 EBG を含むマクロテーブルである．

2.4.2 最小 EBG 空間

最小 EBG が操作性の高い汎化の候補として有用であることを示すために，最小 EBG を汎化空間に位置付け，それが利用度同一の部

分空間の極小元となることを示す.

定義 2.8 (最小 EBG 空間)

1. Sが説明構造集合 Tの部分集合のとき, Sを Sの最小 *EBG* とその剰余とに置き換えて得られる Tの汎化を最小 *EBG* 汎化 *(genelalization by least EBG)* という.

2. Tから最小 *EBG* 汎化を繰返して得られる汎化の空間を最小 *EBG* 空間といい, $\Lambda(T)$ で表わす. ■

定理 2.5 (最小 EBG の極小性)

$\Lambda(T)$ のある要素 T'の利用度を n とすると, T'は$\Gamma(T,n)$ の極小元である.

証明 $T' \preceq T'' \preceq T$なる T''から T'への汎化は利用度を増加させることを示す. 直接の汎化について示せば十分である. T''から T'への直接の汎化において, T''の t''の汎化を t'_0, 剰余を $t'_1, \cdots, t'_n$とする.

1. t''の重複度が 1 のとき, T'における重複度 1 の説明構造は, 最小 EBG 汎化の定義により, Tの説明構造かまたは最小 EBG の剰余である. したがって, t'_0は最小 EBG であり, 利用度は増加しなければならない.

2. t''の重複度が $k(\geq 2)$ のとき, T'における重複度 kの説明構造は, k個の説明構造の最小 EBG である. t'_0の重複度が kとすると, t''は同じ k個の説明構造の共通汎化であるから t'_0が最小

EBG であることに矛盾する．したがって，利用度は増加しなければならない． ■

最小 EBG 空間は，汎化空間の一部を最小 EBG 汎化によって“格子状に”取り出した部分空間である．定理 4 は，これが n-利用度部分空間における極小元によって構成されていて，中途半端な汎化は排除されていることを意味する．

最小 EBG 汎化以外にも n-利用度部分空間の極小元となる汎化が存在することに注意されたい．最小 EBG 汎化だけを利用する汎化生成器は一般には不完全である．これは，最小 EBG は木構造の根からの共通部分に注目するために，木構造の途中が共通であるような汎化を生成できないことによる．また，最小 EBG 空間だけを考えても，後戻り回数には部分的単調性が生じることがわかっている．最小 EBG 汎化が他の極小元より操作的であることが保証されるような問題領域のクラスを明らかにすることは，今後の課題である．

2.5 むすび

本章では，拡張 EBL の概念的枠組を示し，説明構造の汎化と操作性規範を論理プログラム上に形式化して，理論的考察を行なった．その結果，操作性規範の部分的単調性が明らかとなった．また，操作的な汎化の探索に有用な概念として最小 EBG の概念を導入した．

しかしながら，後戻り回数の部分的単調性は SBL における信頼性規範の単調性と比較して操作性の高い汎化の探索にとっては都合の悪い性質である．また，最小 EBG の概念だけでは操作性の高い汎化の探索に十分ではないことも考えられる．拡張 EBL を実際の問

題に応用するためには最小 EBG を利用する学習器をさまざまな領域に適用して，その限界を明らかにする必要がある．

3 章では，最小 EBG の概念に基づく学習器を構成し，いくつかの例題への適用を通じて，理論的結果を確認するとともにその限界を論じる．

Chapter 3

最小EBGに基づく学習器

本章では，前章で導入した最小EBGの概念に基づく素朴な学習器を構成して，いくつかの簡単な問題に適用して，理論的考察の結果を例示し，その限界について考察する．

3.1 学習器の設計

前章で示した拡張EBLの枠組みを学習器としてストレートに実装することを考える．学習器のトップレベルの動作を図3.1に示す．問題解決器はpure-Prologによって問題解決を行い，説明を出力する．図3.2に汎化生成器の骨格を示す．汎化生成器は現在までに与えられた例題の最小EBG汎化に新しい説明構造を付加することによって最小EBG汎化を逐次的に生成する．操作性規範は，生成された最小EBG汎化に基づくマクロを適当に並べ替えて得られるマクロテーブルについて，現在のマクロテーブルより利用度が高く，後戻りが少ないことを検証する．

学習器は次の2つの動作モードをもつ．第1は例題と目標が与えられて，説明構造とマクロテーブルを生成する自主学習モードであ

$learner(Domain, Example, Goal, Explanation) :-$
$\quad problem\_solver(Domain, Example, Goal, Explanation),$
$\quad lebg\_generator(Domain, Explanation, Generalization),$
$\quad operationality\_criterion(Domain, Generalization, MacroTable),$
$\quad knowledge\_manager(modify, Domain, MacroTable).$

図 3.1: 拡張 EBL 学習器のトップレベル

$lebg\_generator(Domain, Explanation, Generalization) :-$
$\quad knowledge\_manager(retrieve, Domain, OldGeneralization),$
$\quad add\_lebg(OldGeneralization, [Explanation], Generalization).$
$add\_lebg(G, [\,], G).$
$add\_lebg(G, A, H) :-$
$\quad one\_of(E, A, A'), one\_of(F, G, G'),$
$\quad lebg(E, F, L, RE, RF),$
$\quad union([G', [L], RE, RF], G''), union([A', RE, RF], A''),$
$\quad add\_lebg(G'', A'', H).$

図 3.2: 最小 EBG 汎化生成器

$$member(A,[A|\_]) \leftarrow .$$
$$member(A,[\_|X]) \leftarrow member(A,X).$$

$$append([\,],Y,Y) \leftarrow .$$
$$append([A|X],Y,[A|Z]) \leftarrow append(X,Y,Z).$$

図 3.3: リスト処理の領域理論

る．第 2 は例題と目標と説明構造が与えられて，マクロテーブルを生成する教示学習モードである．これらのモードは，同一のプログラムに対して変数を具体化するかどうかによって切り替わる．

このような学習器を MS-DOS 上の Arity Prolog(V5.0) 上に実装し，いくつかの実験を行った．

3.2 適用例

いくつかの簡単な例題に前節で構成した学習器を適用して，その動作を評価する．

3.2.1 リスト処理

操作性問題のうち，記憶効率の悪化問題が最小 EBG によって解決される例を示す．簡単なリスト処理に適用する．領域理論として，図 3.3に示すような良く知られた $member/2$ と $append/3$ を例とする．

数多くの例題が入力されるにしたがって，仮説は元の領域理論に収束する．元の 2 つの節からなる領域理論が最も利用度が高く後戻り階数の少ない (0) 仮説である．その過程を図 3.4に示す．

同様の例題を扱う PROLEARN [15] では，例題の数だけマクロ

例題	獲得されるマクロテーブル	利用度	後戻り
$\leftarrow member(e,[a,b,c,d,e,f,g])$.	$member(e,[a,b,c,d,e,f,g]) \leftarrow$.	0	0
$\leftarrow member(3,[1,2,3,4,5])$.	$member(A,[A\|X]) \leftarrow$. $member(C,[A,B\|X]) \leftarrow$ $member(C,X)$.	4	0
$\leftarrow member(t,[s,t,u,v,w])$.	$member(A,[A\|X]) \leftarrow$. $member(B,[A\|X]) \leftarrow$ $member(B,X)$.	8	0

図 3.4: リスト処理への適用例

ルールが生成されて，かえって記憶効率が悪化することに注意されたい．このように簡単なリスト処理のような問題では，操作性問題が解決される．

3.2.2 不定積分

従来の SBL システムと比較する．学習システムを不定積分の領域に適用する．LEX[10] は単一のオペレータの前提条件を，それが問題解決に役立つか否かで分類して帰納する SBL（ Simirality-Based Learning ）システムである．LEX は木構造のパターン階層からなるバイアス（汎化の記述言語）を持ち，バージョン空間法によって汎化の生成を管理する．LEX のように単一のオペレータのみに着目するシステムでは，オペレータの系列を必要とする問題に対して，過剰汎化のおそれがある．一方，単にマクロを蓄積する手法では過剰特殊化，すなわち照合効率の悪化のおそれがある．

問題を拡張 EBL 上に定式化する．被積分式のパターン階層は項によって表現する．例えば，整数 “1” は次式のように表現される．

$$exp(poly(const(int(1)))) .$$

$$
\begin{aligned}
&solve(\textstyle\int Exp\mathrm{d}x, Ans) \leftarrow \\
&\quad call(part\_of(Exp, SubExp), \\
&\quad call(subst\_1(\textstyle\int Exp\mathrm{d}x, t = SubExp, \int Exp'\mathrm{d}t)), \\
&\quad solve(\textstyle\int Exp'\mathrm{d}t, Ans'), \\
&\quad call(subst\_2(Ans', t = SubExp, Ans)).
\end{aligned}
$$

図 3.5: 不定積分の領域理論 (置換積分)

a.	$\int(x^2-1)(x^2+1)\mathrm{d}x$	b.	$\int(x^3+1)(2x^3+x+1)\mathrm{d}x$
c.	$\int(x+2)^7(2x+5)\mathrm{d}x$	d.	$\int(2x+3)^5(3x+2)\mathrm{d}x$
e.	$\int(x+3)^{\frac{1}{2}}(x+2)^2\mathrm{d}x$	f.	$\int(3x+2)^{\frac{2}{3}}(2x+1)\mathrm{d}x$

図 3.6: 例題集合

変数汎化によってパターンの汎化が表現される．例えば，“整数” は次式のように表現される．

$$exp(poly(const(int(X)))) .$$

以下，繁雑さを避けるために通常の表記を用いる．図 3.5に領域理論の一部として置換積分オペレータを示す．図中 $call/1$ は Prolog の手続き呼出しを表わし，理論上は例題の事実として取扱われる．

2 つの多項式の積の展開と置換積分による積分に注目する．これらのオペレータはどんな状況でも用いることができるが，展開にはコストがかかり，置換は無駄に終る可能性がある．図 3.6, 3.7に一連の例題とその解を示す．これらは 3 つのグループに分類される．a) と b) は多項式の積を展開するもの，c) と d) は一次式の整数乗の展

a. $\int(x^2-1)(x^2+1)\mathrm{d}x$	$=$	$\int(x^4-1)\mathrm{d}x$	expand
	$=$	$\frac{1}{5}x^5-x+C$	integrate
c. $\int(x+2)^7(2x+5)\mathrm{d}x$	$=$	$\int t^7(2t+1)\mathrm{d}t$	subst.with $t=x+$
	$=$	$\int(2t^8+t^7)\mathrm{d}t$	expand
	$=$	$\frac{2}{9}t^9+\frac{1}{8}t^8+C$	integrate
	$=$	$\frac{2}{9}(x+2)^9+\frac{1}{8}(x+2)^8+C$	recovcer subst.
e. $\int(x+3)^{\frac{1}{2}}(x+2)^2\mathrm{d}x$	$=$	$\int t^{\frac{1}{2}}(t-1)^2\mathrm{d}t$	subst.with $t=x+$
	$=$	$\int(t^{\frac{5}{2}}-2t^{\frac{3}{2}}+t^{\frac{1}{2}})\mathrm{d}t$	expand
	$=$	$\frac{2}{7}t^{\frac{7}{2}}-\frac{4}{5}t^{\frac{5}{2}}+\frac{2}{3}t^{\frac{3}{2}}+C$	integrate
	$=$	$\frac{2}{7}(x+3)^{\frac{7}{2}}-\frac{4}{5}(x+3)^{\frac{5}{2}}+\frac{2}{3}(x+3)^{\frac{3}{2}}+C$	recover subst.

図 3.7: 例題の解

開を置換積分によって単純化するもの，e) と f) は一次式の非整数乗を置換しなければ解けないものである．

図 3.8に，例題とその解を a~f の順に教示した後に生成されたマクロテーブルを示す．このように 2 つのグループが最小 EBG として獲得された．

単一オペレータの前提を帰納する SBL システムと比較する．6 つの例題はいずれも展開を含むため，展開オペレータの前提条件は最小汎化であっても過剰汎化である．すなわち，a)~f) のどの例題に対しても，展開を使うべきか置換を使うべきかが区別できず，過去の問題を解きなおすとき後戻りを生じる．

一般に，SBL において過剰汎化を避けるためには，複雑な仮説を生成するようにバイアスを調整する必要があり，適切な仮説を獲得するためには膨大なデータが必要であることに注意されたい．少数の例題から複雑な仮説を生成できることは EBL の特徴である．

1	$\int(Ax+B)^N(Cx+D)\mathrm{d}x$	$=$	$\int t^N(Et+F)\mathrm{d}t$	subst.with $t=Ax+B$
		$=$	$\int Poly_1\mathrm{d}t$	expand
		$=$	$Poly_2$	integrate
		$=$	$Poly_3$	recover subst.
2	$\int(Ax+B)^F(Cx+D)^N\mathrm{d}x$	$=$	$\int t^F(Et+F)^N\mathrm{d}t$	sust.with $t=Ax+B$
		$=$	$\int Poly_1\mathrm{d}t$	expand
		$=$	$Poly_2$	integrate
		$=$	$Poly_3$	recover subst.
3	$\int Poly_1\cdot Poly_2\mathrm{d}x$	$=$	$\int Poly_3\mathrm{d}x$	expand
		$=$	$Poly_4$	integrate

図 3.8: 最小 EBG 学習器によるマクロテーブル

また，単にマクロを蓄積する手法と比較すると，マクロテーブルが整理されていることに注意されたい．不必要に複雑（特殊）な仮説を生成しないことは最小 EBG 汎化の特徴である．

ここで実装した学習システムは試作品であって理論に対して不完全である．理論上は最小 EBG 汎化として得られるはずのマクロテーブルが，本システムでは例題の提示順序によっては獲得されないことがある．次の例題を考える．

$g)$ $(x+2)(x+1)\mathrm{d}x$.
$h)$ $(2x+3)(3x-1)\mathrm{d}x$.

これら例題と解を a~f) に続けて教示したときには，1 次式の積を展開するマクロがマクロテーブルの先頭に付加される．一方，c) より前に教示したときには，展開について過剰汎化されたマクロテーブルが生成される．

このように，SBL と比較してより少ない例題から複雑な仮説が生成できる EBL の利点が保存されている反面，素朴な学習器は例題

$$
\begin{aligned}
&hanoi(1, From, With, To) \leftarrow move(1, From, To).\\
&hanoi(N, From, With, To) \leftarrow\\
&\qquad MisN-1,\\
&\qquad hanoi(M, From, To, With),\\
&\qquad hanoi(N, From, To),\\
&\qquad hanoi(M, With, From, To).
\end{aligned}
$$

図 3.9: ハノイの塔の再帰手続き

の順序に敏感であって，偶然の一致によって誤った仮説を生成することがある．

3.2.3 ハノイの塔

より複雑な仮説の生成を試みる．学習器をハノイの塔における再帰的手続きの獲得に適用する．ハノイの搭はよく知られた問題解決の例題である．

定義 3.1 (ハノイの搭) *3* 本の搭 *A,B,C* があり，そのうち *A* には何枚かの大きさの異なる円盤がある．搭から搭へ *1* 度に *1* 枚ずつ円盤を移動して，*A* のすべての円盤を *C* に移動する手順を求めよ．ただし，移動先の搭には円盤がないか，または移動される円盤は，移動先の搭の頂上にある円盤より小さくなければならない． ■

ハノイの搭の問題は，通常状態計算による探索問題として形式化され，図 3.9に示すような手続によって決定的に解決されることが知られている．ハノイの搭を探索問題として形式化した領域理論のもとで，異なる枚数の円盤を持つ例題を与え，最小 EBG によって再

帰的手続きを獲得することを試みる.

領域理論を図 3.10に示す. これらは必要最小限の状態遷移規則と, 再帰的手続きを得るために付加された知識とからなる. 必要最小限の状態遷移規則からでは, 再帰的手続きを得られないことに注意されたい. 再帰的手続きは「搭 From の一番下の円盤以外の円盤を With に移動し, From の一番下の円盤を To に移動した後, With の円盤を To に移動する」ことからなる. 生成される副問題が主問題と同一の手続きによって再帰的に解かれるためには, 少なくとも搭の役割を識別することと, 一番下の円盤の固定を識別することが必要である.

付加された知識は次の内容からなる.

1. 搭の交換規則

   「2つの塔を交換しても問題は変らない.」

2. 円盤の固定規則

   「状態遷移の前後である塔の一番下の円盤が移動しないならば, その円盤を固定して解いても問題は変らない.」

円盤の固定規則は, 一番下の円盤の識別のために特殊化して実装されている. 状態は述語 *hanoi*/3 によって表現される. 第1引数は搭の交換規則で参照される搭の役割を表わす. 第2, 3引数はそれぞれ状態遷移の前後の状態を表わす. 個々の搭の円盤の固定状況は重リストの形式で表わす. 例えば, $[1,2,3]-[3]$ は円盤 1,2,3 がこの順でその搭にあり, 3 が固定されていることを表わす.

領域理論において, *call*/1 は制約の計算を示すメタ述語で, 理論上各例に固有の事実とみなし, 必ず構造汎化されるものとする. 単

/* 円盤移動 */
$hanoi(\_, s([P|A]-AT, B-BT, C-CT), s(A-AT, B-BT, [P|C]-CT)) \leftarrow$
$free(P, [P|A]-AT) \wedge$
$safe\_to\_stack(P, C-CT).$
$\cdots$ (6 通り)

/* 円盤の固定チェック */
$free(P, T-TT) \leftarrow call(T \neq TT).$

/* 円盤の大きさチェック */
$safe\_to\_stack(P, []-[]).$
$safe\_to\_stack(P, [Q|\_]-\_) \leftarrow call(P < Q).$

/* 行動の合成 */
$hanoi(T, S1, S3) \leftarrow hanoi(T, S1, S2) \wedge hanoi(T, S2, S3).$

/* 塔の交換規則 */
$hanoi(t(From, To, With), s(A-AT, B-BT, C-CT), s(NA-AT, NB-BT, NC-CT$

$hanoi(t(From, With, To), s(A-AT, C-CT, B-BT), s(NA-AT, NC-CT, NB-$
$\cdots$ (6 通り)

/* 円盤固定規則 */
$hanoi(T, s(A-AT, BT-BT, CT-CT), s([P|AT]-AT, B-BT, C-CT)) \leftarrow$
$hanoi(T, s(A-[P|AT], BT-BT, CT-CT), s([P|AT]-[P|AT], B-BT, C-CT$.
$\cdots$ (3 通り)
$hanoi(T, s([P|AT]-AT, B-BT, C-Ct), s(A-AT, BT-BT, CT-CT)) \leftarrow$
$hanoi(T, s([P|AT]-[P|AT], B-BT, C-CT), s(A-[P|AT], BT-BT, CT-CT$.
$\cdots$ (3 通り)

図 3.10: ハノイの塔の領域理論

$$hanoi(t(F,W,T),s([1,2|AH]-A,B-B,C-C),s(A-A,B-B,[1,2|CH]-C)) \leftarrow$$
$$hanoi(t(F,T,W),s([1,2|AH]-[P|A],C-C,B-B),$$
$$s([P|A]-[P|A],C-C,[1,2|BH]-B))\ (1)\wedge$$
$$call([P|A] \neq A)\ (2)\wedge$$
$$safe\_to\_stack(P,B-B)\ (3)\wedge$$
$$hanoi(t(W,F,T),s([1,2|BH]-B,A-A,[P|C]-[P|C]),$$
$$s(B-B,A-A,[1,2|CH]-C))\ (4).$$

図 3.11: 獲得された最小 EBG

一化は occur check をともなう.

最小 EBG の生成器を，PC98XA，MS-DOS 上の Arity Prolog Compiler (Ver.4.0) に実装して，2 枚の円盤と 3 枚の円盤からなる 2 つの問題について，それぞれ解答を教示して最小 EBG を生成させた．獲得された最小 EBG とその説明構造を図 3.11 と図 3.12に示す．これは前述の再帰的手続きに対応する．

獲得された再帰的手続きについて考察する．獲得された手続きは，搭 A に 2 枚以上ある円盤を搭 C にそっくり移動するタイプの問題に適用可能である．このタイプの問題のみが与えられる場合にはマクロの個数が増えることはない．このことは，例題の増加に従ってマクロの数が増加し続けるシステムと比較して重要な性質である．

この領域理論では，搭 A の円盤を搭 C にそっくり移動するタイプの問題に限らず，任意の状態から任意の状態へ遷移する問題が記述可能である．このタイプから外れたとき，最小 EBG は，過去の例に対しても後戻りを生じる過剰汎化を引起こす．このとき，例題の集合を分割して，マクロを集合として維持することは今後の課題である．ハノイの搭は，任意の状態遷移に対しても手続化可能であ

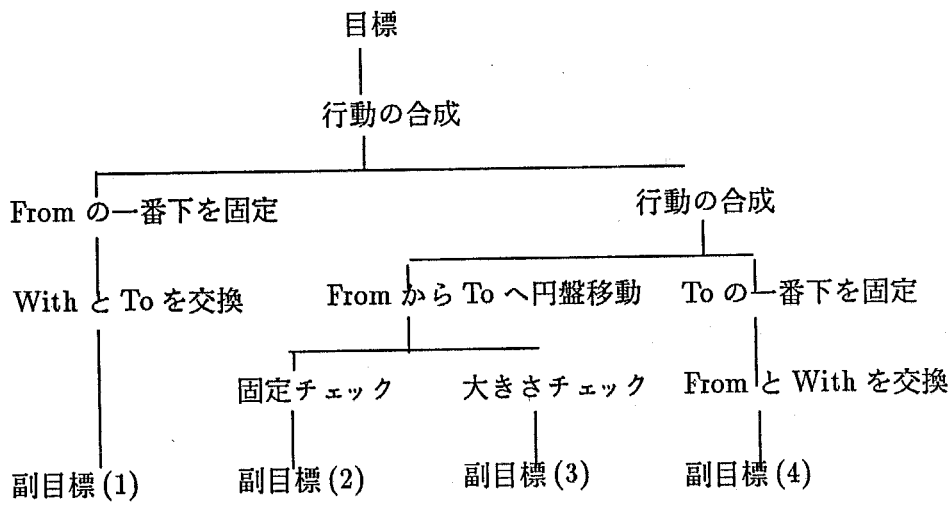


図 3.12: 最小 EBG の説明構造

ることが知られている [7].

獲得されたマクロの第 2 , 3 連言項は, 広い意味で冗長な制約の計算である. 第 2 連言項の単一化失敗の検証は, 現在の領域理論においても部分計算によって除去可能である [15] が, 第 3 連言項の *safe_to_stack* については不可能である.

このように, 複雑な仮説を少ない例題から生成することができるが, 生成される仮説は知識表現に敏感であって, 注意深く知識表現を調整しなければならない.

3.3 むすび

本章では, 最小 EBG を利用する素朴な学習器を構成し, いくつかの例題に適用してその効用と限界について考察した.

その結果, 理論的成果で示したような操作性の保証問題は, 簡単な例 (リスト処理問題) では解決されることが示された. しかしながら, より複雑な例では, 偶然の一致の影響 (不定積分問題) や, 知識表現への敏感性 (ハノイの塔問題) があることがわかった. したがって, 実際の応用のためには領域固有の知識や, 教師への積極的な働きかけによって, より多くの情報を利用することが必要であると考えられる.

4 章では, 本章で提案した最小 EBG に基づく学習器の問題点を解決するためにメタ領域知識と質問の学習における役割を論じ, これらを利用した学習器のイメージを提案する.

Chapter 4

メタ領域知識と質問の利用

本章では，領域固有の情報を利用して学習器における探索の洗練化を行なう問題について考察する．領域固有の情報の利用のために，メタ領域知識と質問の2つの形式を提案する．

4.1 探索の洗練化

前章の議論で，最小EBGを用いた素朴な学習器は，偶然の一致の影響を受け，知識表現に敏感であって，誤りの可能性を除去できないという限界があることが明らかとなった．これらの限界は，第2章で述べた後戻り回数の部分的単調性と最小EBG汎化の不完全性に起因するもので，一般には克服できないものと考えられる．したがって，何らかの情報を補うことによって探索の洗練化をはかることが必要である．

素朴な学習器は生成-検査法に基づく．生成-検査法における探索の洗練化とは，生成に検査の情報を組み込んで無駄な仮説の生成をなるべく制限し，有用な仮説の生成をなるべく優先することである．

前章までは，問題固有の情報の利用を避けて，仮説の適切さの検

査の規範イコール操作性規範として議論してきた．ここでは，操作性規範の他にメタ領域知識と質問の2種類の問題固有の情報を考え，探索をガイドする上での役割を論じる．

4.2 メタ領域知識の利用

領域固有の第1の情報は，学習すべきマクロテーブルの形式を特定する知識である．前章までに扱った領域理論は，問題解決システムに対して与えられる問題の状態および状態遷移を記述するレベルの知識である．これに対して，探索をガイドするような知識 (例えば，学習すべきマクロテーブルの形式を特定する知識) は，領域理論に関するメタレベルの知識であることから，これらをメタ領域知識と呼ぶ．

4.2.1 メタ領域知識による枝刈り

メタ領域知識は，領域理論の形式やその領域で発生される問題の型などのメタレベルの知識からなり，学習すべきマクロテーブルの形式を特定して，学習器が生成する非合法的な仮説の枝刈りに用いられる．ここでは，いくつかの例を示して，メタ領域知識がどのように利用できるかについて議論する．

まず，前章で取り上げた不定積分問題を考える．不定積分問題において学習されるマクロは，その前提条件と照合するクラスの被積分式を，その結論と照合するより簡単なクラスの式に変換する機能を持つ．学習すべきマクロテーブルは積分記号を含まないクラス，すなわち解を根とする木構造となるとき，任意のクラスから一直線に

解にたどり着くことができるので，最も効率的な問題解決が可能である．したがって，各マクロの前提条件と結論を結んだとき，循環が生じるようなマクロテーブルは誤ったマクロテーブルである．学習すべきマクロテーブルがこのような構造を持つことがわかっていれば，循環を引き起こすような誤ったマクロテーブルを枝刈りすることができる．

次に，次章で詳述する8パズルについて考える．8パズルは直列分解可能な副目標を持つクラスに属することがわかっている．直列分解可能な副目標を持つ問題解決では，副目標の間に適当な解決順序を設定することによって，任意の問題がマクロテーブルによって後戻りや繰り返しなしに解決される．このようなマクロテーブルを問題解決マクロテーブルという．したがって，8パズルで学習されるべきマクロテーブルは問題解決マクロテーブルの形式という制約を満足する形でなければならない．これによって，探索空間を枝刈りすることができる．さらに，もし解決順序がメタ領域知識として明らかに定義されている時は，問題は非常に単純となる．なぜなら，生成されるマクロが解決順序におけるどの副目標と副目標の間にはいるかを確かめて振り分ければ良いからである．

ここに現れた“マクロは循環してはならない，”あるいは“問題解決マクロテーブルの形式を持たなければならない”というメタ知識は経験則ではなく，領域に関するより明らかなメタ知識から導出される“メタ定理”であることに注意されたい．このように，メタ領域知識を探索の洗練化に利用するためには，より単純で明らかなメタ領域知識から，有用なメタ領域知識を導出する必要がある．このことについて次節で議論する．

4.2.2 メタ領域知識の導出

ここでは，8パズルを例にとって，有用なメタ領域知識の導出について述べる．

メタ領域知識の導出のためには，初期に利用可能なより単純で明らかなメタ知識を定義する必要がある．これには次のような種類の知識があると考えられる．

1. 領域理論の形式を特定するメタ領域知識．

   このタイプのメタ領域知識は，領域理論に現れる項，述語，節の形式の制約，およびPrologプログラムとしての節の順序づけの制約からなる．

   8パズルの領域理論の一部を図4.1に示す．8パズルの領域理論は状態を記述する唯一の関数記号，状態遷移を意味する唯一の述語記号によって記述され，唯一の目標状態を表す節と，24個の作用素を表す節からなる．

2. マクロテーブルの形式を特定するメタ領域知識．

   このタイプのメタ領域知識は，マクロテーブルに現れる節の形式の制約，およびPrologプログラムとしての節の順序づけの制約からなる．

   8パズルのマクロテーブルの一部を図4.2に示す．8パズルのマクロテーブルには，“ブランクを中央に合わせる，タイル1を合わせる，タイル2を合わせる，…”という解決順序が設定されている．マクロテーブルを構成する35個のマクロは，前提と結論のみからなる単純な形式の節で，それぞれ対応する副目

標を達成するという機能を持つ．Prologプログラムとしては解決順序と逆順，すなわちより目標に近いものから順に並べられる．

3. 発生しうる問題を特定するメタ領域知識．

   問題解決システムを実際の問題に適用することを考えると，問題は単純化されて形式化されることが多い．すなわち，領域理論を記述する言語で記述可能な問題がすべて問題解決システムに与えられるわけではない．このタイプのメタ領域知識は，そのような実際に与えられうる問題，ないしは実際には与えられない問題の特徴づけからなる．

   8パズルでは，与えられる問題は目標状態の置換であって，同じタイルを2枚以上含むような問題は考えない．

4. 操作性規範に関するメタ領域知識．

   操作性規範に関するすでに知られていることがらも，もちろん一種のメタ領域知識である．第2章で考察した操作性規範と汎化の関係，最小EBGに関する知見などからなる．

この他には，いくつかの常識的知識や経験的知識が考えられる．

以上のようなメタ領域知識に基づいて，探索に有用なメタ領域知識が導出される．8パズルでは定理5.1 (5章で詳述) に代表されるメタ領域知識が導出される．あるマクロテーブルから生成される解を例題として与えた時，元のマクロテーブルを学習するために学習器が獲得すべき情報は，解決順序と各マクロの構造汎化 (作用素列)，変数汎化 (前提と結論) およびマクロテーブル上の位置である．定理

$$solve\left(\begin{array}{|c|c|c|}\hline 1&2&3\\\hline 8&0&4\\\hline 7&6&5\\\hline\end{array},[\,]\right) \quad (s(0,1,2,3,4,5,6,7,8) \text{ in a term}).$$

$$solve\left(\begin{array}{|c|c|c|}\hline T_1&T_2&T_3\\\hline T_8&0&T_4\\\hline T_7&T_6&T_5\\\hline\end{array},[u_0|A]\right) \leftarrow solve\left(\begin{array}{|c|c|c|}\hline T_1&T_2&T_3\\\hline T_8&T_6&T_4\\\hline T_7&0&T_5\\\hline\end{array},A\right).$$

$$solve\left(\begin{array}{|c|c|c|}\hline 0&T_2&T_3\\\hline T_8&T_0&T_4\\\hline T_7&T_6&T_5\\\hline\end{array},[u_1|A]\right) \leftarrow solve\left(\begin{array}{|c|c|c|}\hline T_8&T_2&T_3\\\hline 0&T_0&T_4\\\hline T_7&T_6&T_5\\\hline\end{array},A\right).$$

$$solve\left(\begin{array}{|c|c|c|}\hline T_1&0&T_3\\\hline T_8&T_0&T_4\\\hline T_7&T_6&T_5\\\hline\end{array},[u_2|A]\right) \leftarrow solve\left(\begin{array}{|c|c|c|}\hline T_1&T_0&T_3\\\hline T_8&0&T_4\\\hline T_7&T_6&T_5\\\hline\end{array},A\right).$$

$$solve\left(\begin{array}{|c|c|c|}\hline T_1&T_2&0\\\hline T_8&T_0&T_4\\\hline T_7&T_6&T_5\\\hline\end{array},[u_3|A]\right) \leftarrow solve\left(\begin{array}{|c|c|c|}\hline T_1&T_2&T_4\\\hline T_8&T_0&0\\\hline T_7&T_6&T_5\\\hline\end{array},A\right).$$

$$solve\left(\begin{array}{|c|c|c|}\hline T_1&T_2&T_3\\\hline T_8&T_0&0\\\hline T_7&T_6&T_5\\\hline\end{array},[u_4|A]\right) \leftarrow solve\left(\begin{array}{|c|c|c|}\hline T_1&T_2&T_3\\\hline T_8&T_0&T_5\\\hline T_7&T_6&0\\\hline\end{array},A\right).$$

$$solve\left(\begin{array}{|c|c|c|}\hline T_1&T_2&T_3\\\hline 0&T_0&T_4\\\hline T_7&T_6&T_5\\\hline\end{array},[u_8|A]\right) \leftarrow solve\left(\begin{array}{|c|c|c|}\hline T_1&T_2&T_3\\\hline T_7&T_0&T_4\\\hline 0&T_6&T_5\\\hline\end{array},A\right).$$

...

図 4.1: 8 パズルの領域理論

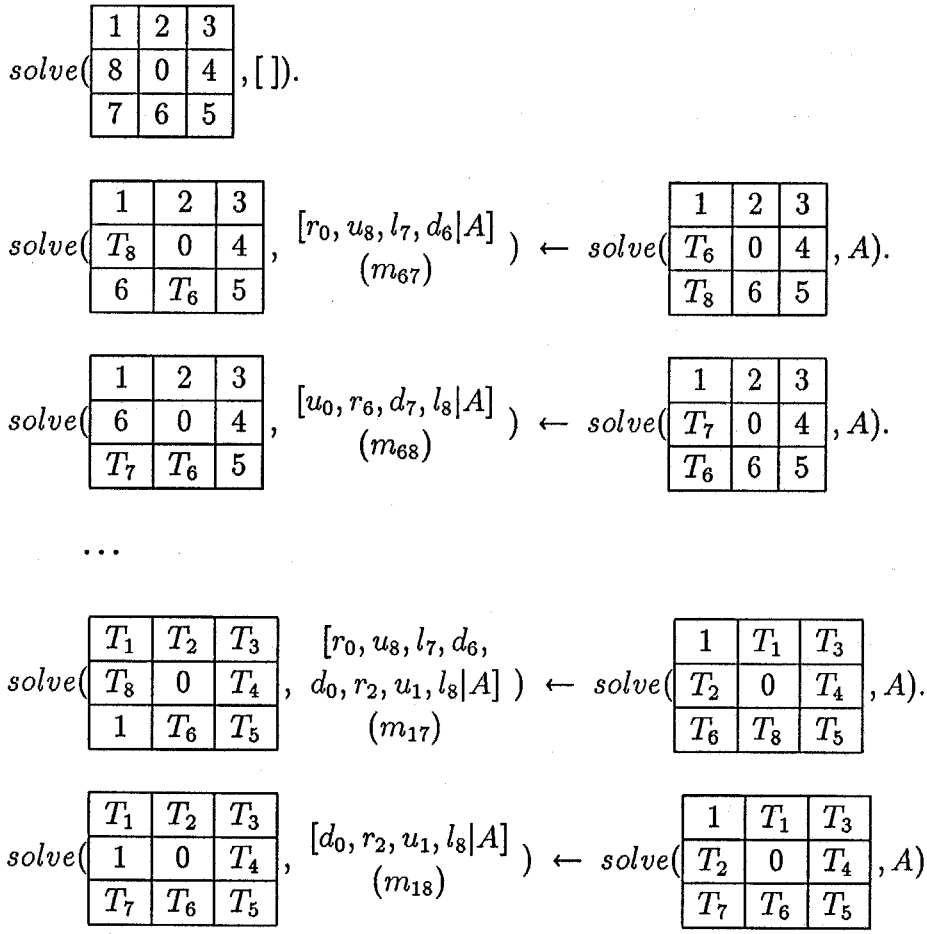


図 4.2: 8 パズルのマクロテーブル

5.1 は，各マクロの構造汎化とマクロテーブル上の位置がわかれば，マクロの変数汎化と解決順序を安価に求めることができることを保証する．このメタ領域知識に基づいて，8 パズルの学習器における探索は，各マクロの構造汎化を求めるフェーズとそれらを順序づけるフェーズに分割される．このようなフェーズの分割は探索の効率化の観点から有用である．なぜなら，それらが独立ならば，探索すべき空間は積から和へと縮小されるからである．

このように，より単純で明らかなメタ領域知識から，探索の制限に有用なメタ領域知識が導出される．

4.3 質問の利用

領域固有の第 2 の情報は，学習器が外界 (教師) に働きかけることで得られる情報である．前章までに扱った外界からの情報は，学習器の意図とは無関係であった．これに対して，学習器が外界 (教師) に働きかける行為を質問と呼ぶ．

学習器に強力過ぎる質問を許すことには気をつけなければならない．なぜなら，広い意味での質問は知識をそっくりコピーする行為も含むからである．ここでは，membership query という質問のみを考える．membership query とは，問題と解の対からなる学習器の問いに対して，教師は *yes* か *no* かのみを答える形式の質問である．

4.3.1 質問による枝刈り

membership query は，学習器にとって未知の例題に対する解の良否を教師に問う形式の質問である．membership query によっても

$$e_1 = \overbrace{r_0 \circ u_8 \circ l_7 \circ d_6}^{m_{68}},$$

$$e_2 = \overbrace{\underbrace{r_0 \circ u_8 \circ l_7 \circ d_6}_{(m_{68})} \circ \underbrace{d_0 \circ r_2 \circ u_1 \circ l_8}_{(m_{18})}}^{m_{17}}.$$

図 4.3: 例題集合

やはり仮説の候補の枝刈りが可能である．操作性規範および前節で導入したメタ領域知識を満足するマクロテーブルの候補が複数ある場合を考える．このとき，あるマクロテーブルと別のマクロテーブルでは生成される解が異なるような未知の例題があったなら，それらの良否を教師に質問することによって片方のマクロテーブルの候補を枝刈りすることができる．

8パズルにおいても，操作性規範およびメタ領域知識を満足するマクロテーブルの候補が複数生じることがありうる．例えば，図4.3のような例題を考えると，これには図4.4と図4.5の2種類のマクロテーブルの候補が生成できる．後者は正しいマクロテーブルの一部であるが，前者は偶然の一致によってたまたまマクロテーブルの形式が満足された誤ったマクロテーブルである．

これらのいずれが正しいかは，可能な未知の例題に対する解の良否を質問することによって判定される．どんな組合せに対しても判定が可能であることは，質問と前節のメタ領域知識を組み合わせて導出されるメタ領域知識によって保証される．

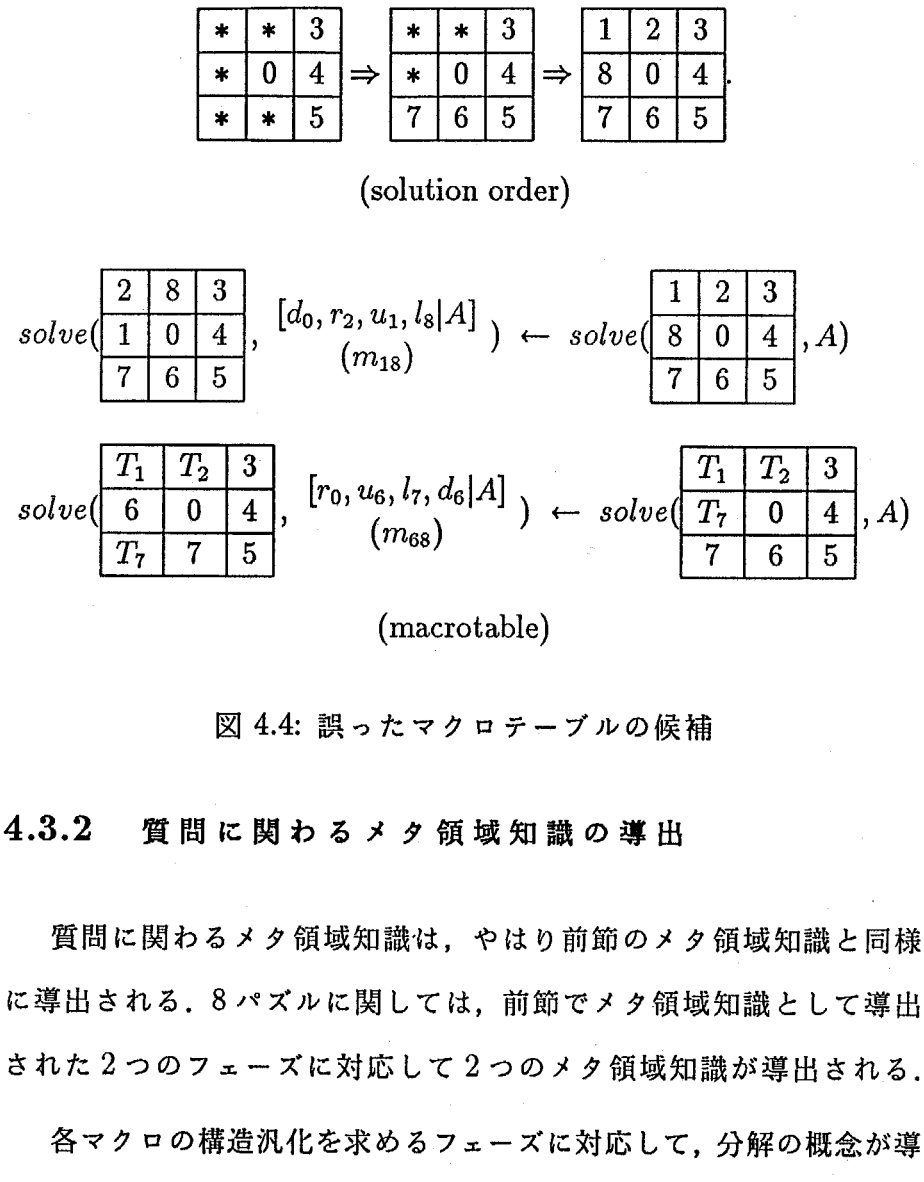


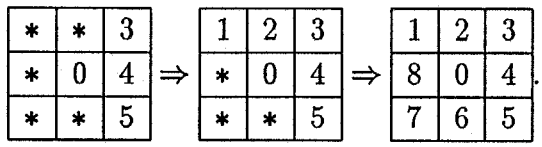
図 4.4: 誤ったマクロテーブルの候補

4.3.2 質問に関わるメタ領域知識の導出

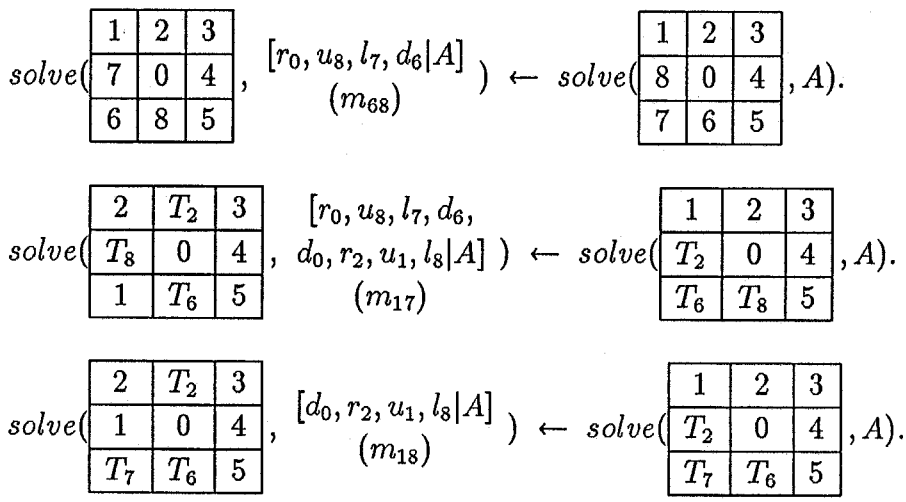
質問に関わるメタ領域知識は，やはり前節のメタ領域知識と同様に導出される．8パズルに関しては，前節でメタ領域知識として導出された2つのフェーズに対応して2つのメタ領域知識が導出される．

各マクロの構造汎化を求めるフェーズに対応して，分解の概念が導出される．例題の解の分解とは，解の作用素列をそれぞれがmemberであるような部分列の合成に分解することである．正しい分解が容易に得られるときは，2つのフェーズの独立性が高められて，探索効率が改善されることが知られている．

マクロを順序づけるフェーズに対応して，直列化テーブルの概念が導出されている．直列化テーブルは，2つのマクロの2通りの合



(solution order)



(macrotable)

図 4.5: 正しいマクロテーブルの候補

成に対する membership query の結果に基づいて，可能な順序関係を列挙した表である．前節の2つの可能なマクロテーブルの候補は，直列化テーブルの適用によってひとつに絞り込むことができる．

このように，質問の利用によってより強力なメタ領域知識が利用可能となる．

4.4 まとめ

本章では，操作性の高い汎化の探索を洗練化するために，メタ領域知識と質問の2つのタイプの情報の利用について考察した．次章では，問題解決マクロテーブルの獲得にメタ領域知識と質問を利用する学習器を提案し，8パズルにおけるマクロテーブルの獲得に適用する．

Chapter 5

問題解決マクロテーブルの学習

本章では，前章で議論したメタ領域知識と質問の利用の適用例として，8パズルに代表される直列分解可能な副目標を持つクラスを対象として，問題解決マクロテーブルの学習を試みる．まず，メタ領域知識を利用可能なものとするために，問題解決マクロテーブルを論理プログラム上に形式化して，その性質を明らかにする．ついで，membership queryの導入と，これに基づく直列化テーブルを定義し，これらに基づく学習器を構成し，8パズルへの適用を例示する．最後に，学習器の正しさと計算量の問題にも言及する．

5.1 問題解決マクロテーブル

積木の世界など連言目標の充足問題には，ある副目標の達成がすでに達成されている他の副目標を破壊するという困難がある．Korfは，8パズルのような置換パズルのあるクラスでは，連言目標間に適当な解決順序を設定することにより，任意の問題がマクロテーブルによって手続的に解決可能となることを示している[7]．このようなクラスの問題解決は，直列分解可能な副目標を持つという．この

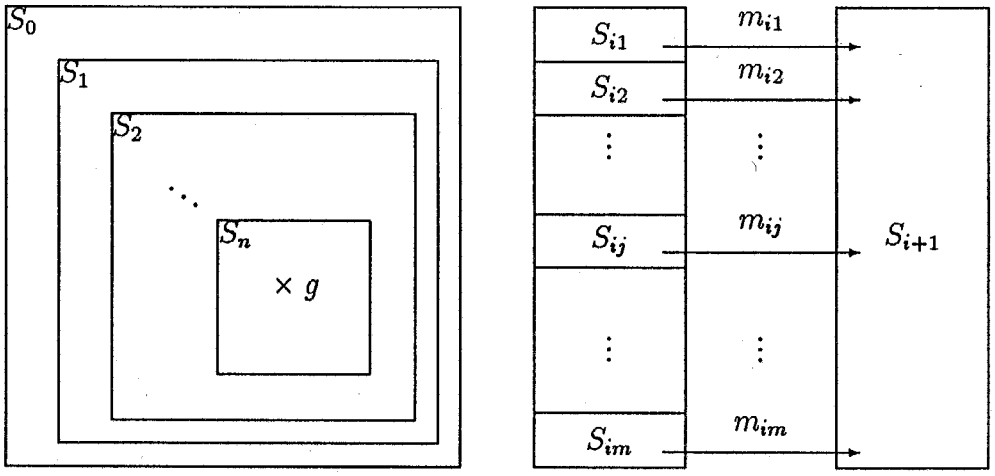


図 5.1: A hierarchy of serially decomposable subgoals and the function of a macrotable.

クラスのマクロテーブルを問題解決マクロテーブル，以下単にマクロテーブルとよぶ．

図 5.1に解決順序による状態空間の階層化とマクロテーブルの機能を模式的に示す．状態空間は $S_i \supset S_{i+1}$ となるように組織化される．S_i は S_{ij} に分類され，それぞれにマクロ m_{ij} が対応する．m_{ij} は S_{ij} の状態を S_{i+1} の状態に遷移させる．直列分解可能な副目標を持つ問題解決では，マクロテーブルを第 0 行からはじめて順次探索することにより，任意の問題を後戻りや繰返しなしに手続き的に解くことができる．

例えば 8 パズルでは，“ブランクを中央に合わせる，タイル 1 を合わせる，タイル 2 を合わせる，…” という解決順序を設定することにより，任意の問題を 35 個のマクロからなるマクロテーブルによって手続き的に解くことができる．

本研究では，あるマクロテーブルから生成される解を拡張 EBL

に基づく学習器に与えて，それらの汎化に基づいて元のマクロテーブルを獲得することを学習問題として設定する．拡張 EBL では，利用度最大化と後戻り最小化の 2 つの操作性規範が考慮される．例題集合に対して，元のマクロテーブルは利用度最大かつ後戻り回数 0 の最も操作的な EBG である．したがって学習器による EBG の探索においては，後戻り回数 0 の最小 EBG 汎化からなる部分空間のみを考えれば良い．

5.2 メタ領域理論の利用

5.2.1 問題解決マクロテーブルの形式化

問題解決マクロテーブルを論理プログラム上に形式化する．

定義 5.1 (領域理論)

領域理論は次の論理プログラム Dで記述される．

$$D = \begin{cases} s(g,[\,]). \\ s(h_1,[op_1|Ans]) \leftarrow s(b_1,Ans). \\ \quad \cdots \\ s(h_n,[op_n|Ans]) \leftarrow s(b_n,Ans). \end{cases}$$

ここで，gは目標状態，h_i, b_iはそれぞれ作用素 op_iの前提，結論を表わす．Dで証明可能な $s/2$ の第 1 引数の集合は状態空間を表わし，第 2 引数はその状態に対する解を表わす． ■

定義 5.1は，Korf による状態ベクター表現に基づく作用素定義 [7] の一般化である．領域理論として，問題解決に必要な最小限の知識である作用素定義のみを考える．作用素の前提条件と結論は単一化によって表現される．図 5.2に 8 パズルの領域理論の一部 (作用素 up)

$$solve\left(\begin{array}{|c|c|c|}\hline 1&2&3\\\hline 8&0&4\\\hline 7&6&5\\\hline\end{array},[\,]\right) \quad (s(0,1,2,3,4,5,6,7,8) \text{ in a term}).$$

$$solve\left(\begin{array}{|c|c|c|}\hline T_1&T_2&T_3\\\hline T_8&0&T_4\\\hline T_7&T_6&T_5\\\hline\end{array},[u_0|A]\right) \leftarrow solve\left(\begin{array}{|c|c|c|}\hline T_1&T_2&T_3\\\hline T_8&T_6&T_4\\\hline T_7&0&T_5\\\hline\end{array},A\right).$$

$$solve\left(\begin{array}{|c|c|c|}\hline 0&T_2&T_3\\\hline T_8&T_0&T_4\\\hline T_7&T_6&T_5\\\hline\end{array},[u_1|A]\right) \leftarrow solve\left(\begin{array}{|c|c|c|}\hline T_8&T_2&T_3\\\hline 0&T_0&T_4\\\hline T_7&T_6&T_5\\\hline\end{array},A\right).$$

$$solve\left(\begin{array}{|c|c|c|}\hline T_1&0&T_3\\\hline T_8&T_0&T_4\\\hline T_7&T_6&T_5\\\hline\end{array},[u_2|A]\right) \leftarrow solve\left(\begin{array}{|c|c|c|}\hline T_1&T_0&T_3\\\hline T_8&0&T_4\\\hline T_7&T_6&T_5\\\hline\end{array},A\right).$$

$$solve\left(\begin{array}{|c|c|c|}\hline T_1&T_2&0\\\hline T_8&T_0&T_4\\\hline T_7&T_6&T_5\\\hline\end{array},[u_3|A]\right) \leftarrow solve\left(\begin{array}{|c|c|c|}\hline T_1&T_2&T_4\\\hline T_8&T_0&0\\\hline T_7&T_6&T_5\\\hline\end{array},A\right).$$

$$solve\left(\begin{array}{|c|c|c|}\hline T_1&T_2&T_3\\\hline T_8&T_0&0\\\hline T_7&T_6&T_5\\\hline\end{array},[u_4|A]\right) \leftarrow solve\left(\begin{array}{|c|c|c|}\hline T_1&T_2&T_3\\\hline T_8&T_0&T_5\\\hline T_7&T_6&0\\\hline\end{array},A\right).$$

$$solve\left(\begin{array}{|c|c|c|}\hline T_1&T_2&T_3\\\hline 0&T_0&T_4\\\hline T_7&T_6&T_5\\\hline\end{array},[u_8|A]\right) \leftarrow solve\left(\begin{array}{|c|c|c|}\hline T_1&T_2&T_3\\\hline T_7&T_0&T_4\\\hline 0&T_6&T_5\\\hline\end{array},A\right).$$

...

図 5.2: The domain theory of 8 puzzle.

を示す．簡単のために状態を表わす項のかわりに盤面によって表示する．このように，通常ひとつの作用素で表現されるものが6通りに表現される．

定義 5.2 (解決順序および問題解決マクロテーブル)

1. 解決順序πは次のような目標gの汎化の列である．

$$\pi \ = \ [S_0, \cdots, S_n, g] \ \ (S_0 \prec \cdots \prec S_n \prec g).$$

ここで，$x \prec y$は，xがyの汎化 (yがxの変数の書き換えを除く具体化) であることを表わす．

2. 解決順序πの下でのマクロテーブルMは次のような論理プログラムとして記述される．

$$M = \left\{ \begin{array}{ll} s(g,[\,]). & \\ s(h_{n1}, m_{n1}) & \leftarrow s(b_{n1}, Ans). \\ \quad \cdots & \\ s(h_{np}, m_{np}) & \leftarrow s(b_{np}, Ans). \\ s(h_{n-11}, m_{n-11}) & \leftarrow s(b_{n-11}, Ans). \\ \quad \vdots & \\ s(h_{01}, m_{01}) & \leftarrow s(b_{01}, Ans). \\ \quad \cdots & \\ s(h_{0q}, m_{0q}) & \leftarrow s(b_{0q}, Ans). \end{array} \right.$$

ここで，各ルールは領域理論の作用素同士の導出形の具体化，すなわちマクロである．m_{ij}は$[op_x, op_y, \cdots | Ans]$の形式を持ち，対応する作用素列を表わす．各マクロの前提および結論は，$h_{ij} \prec S_i$，$b_{ij} \prec S_{i+1}$の条件を満足する． ■

定義 5.2も，Korfの表現の一般化である．解決順序による状態空間の階層化は，項の汎化の関係によって実現される．各マクロは解

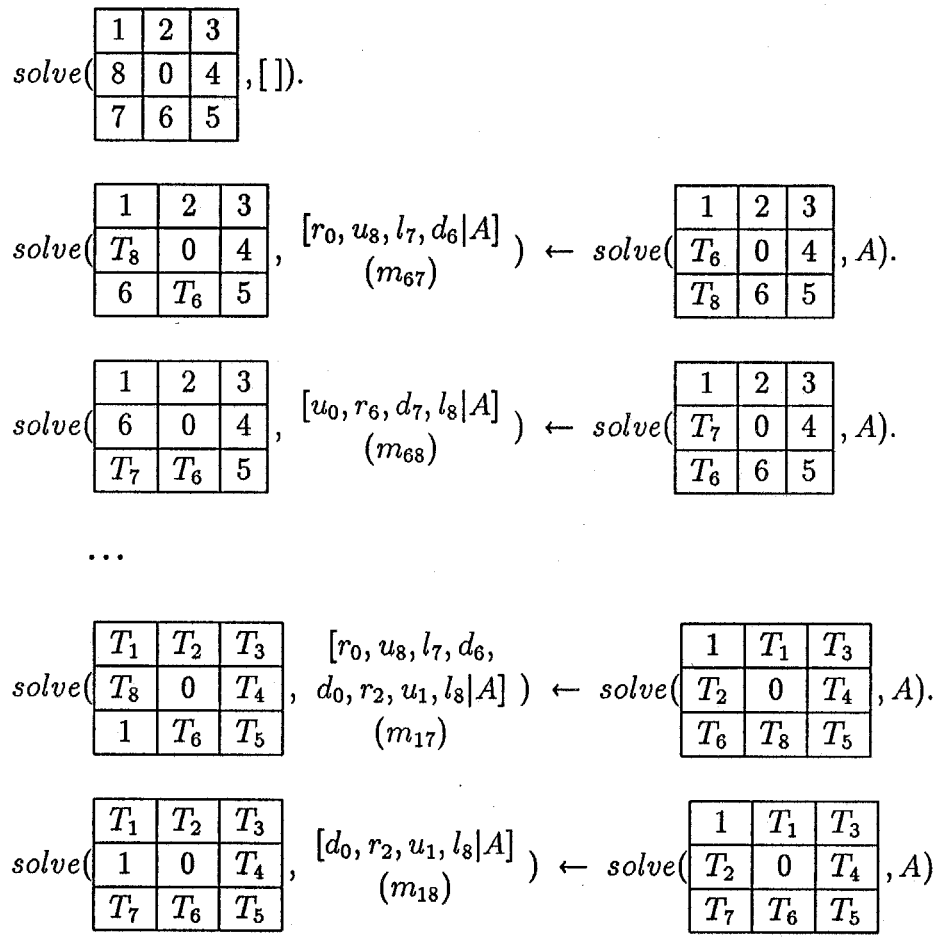


図 5.3: Korf's macro table for 8 puzzle.

決順序と逆順，すなわち特殊な順に並べられていることに注意されたい．pure Prolog インタープリタのもとで，このように組織化された論理プログラムを実行することによって，マクロテーブル上の探索が行われる．

図 5.3に Korf による 8 パズルのマクロテーブルの一部を示す．i 行 j列のマクロ m_{ij}は，jの位置にあるタイル i を目標の位置に移動させるマクロである．

5.2.2 問題解決マクロテーブルの性質

あるマクロテーブルから生成される解を例題として与えたとき，元のマクロテーブルを学習するために拡張EBL学習器が獲得すべき情報は，解決順序と各マクロの構造汎化(作用素列)，変数汎化(前提と結論)およびマクロテーブル上の位置である．以下に，これらの関係について考察する．

作用素およびマクロについて次の記法を用いる．

$$\begin{array}{lcll} m_{ij}(s) & = & b_{ij}\sigma & (\sigma = h_{ij} \wedge s). \\ m_{ij}^{-1}(s) & = & h_{ij}\sigma & (\sigma = b_{ij} \wedge s). \\ m_1 \circ m_2 & = & (h_1 \leftarrow b_2)\sigma & (\sigma = b_1 \wedge h_2). \end{array}$$

ここで，$x \wedge y$はxとyのmguを表わす．

作用素mが情報保存的(information preserving)であるとは，mが状態sとtに適用可能であるとき，$m(s) = m(t)$ならば$s = t$であることである[7]．情報保存性の意味は，結論に無関係な項を前提に含まないことである．作用素が情報保存的であるとき，それらからなるマクロも情報保存的であって，逆の適用を一意に定めることができる．以下，領域は情報保存的であると仮定する．

定理 5.1 (最小汎化による解決順序の計算)
解決順序は次のように最小汎化を用いて計算できる．

$$\begin{array}{lcl} h_{ij} & = & \widehat{m_{ij}}^{-1}(S_{i+1}), \\ S_i & = & g \vee [\bigvee_j h_{ij}]. \end{array}$$

ここで，$\widehat{m_{ij}}$はm_{ij}と同じ作用素列を持つ最も一般的なマクロで，$x \vee y$はxとyの最小汎化*[14]*を表わす．

証明 最小汎化と最汎単一化の可換性[25]を利用して，式の変形によって証明する．この事実は，本論文の記法では$m(x)\wedge m(y)=m(x\wedge y)$と表記される．各マクロは，そのマクロによって解かれるすべての状態と単一化可能，すなわちそれらの最小汎化である．m_{ij}によって解かれるすべての状態は，m_{ij}のみによって解かれる状態，m_{ij}と$i+1$行のマクロによって解かれる状態，…からなる．同様に解決順序は，その行のマクロによって解かれるすべての状態と目標との最小汎化である．したがって，次のような式の変形によって定理 5.1を得る．

$$
\begin{aligned}
h_{nj} &= \widehat{m_{nj}}^{-1}(g).\\
S_n &= g \vee [\bigvee_j \widehat{m_{nj}}^{-1}(g)]\\
&= g \vee [\bigvee_j h_{nj}].
\end{aligned}
$$

$$
\begin{aligned}
h_{n-1j} &= \widehat{m_{n-1j}}^{-1}(g) \vee [\bigvee_k \widehat{m_{n-1j}}^{-1}(\widehat{m_{nk}}^{-1}(g))]\\
&= \widehat{m_{n-1j}}^{-1}(g) \vee \widehat{m_{n-1j}}^{-1}([\bigvee_k \widehat{m_{nk}}^{-1}(g)])\\
&= \widehat{m_{n-1j}}^{-1}(g \vee [\bigvee_k \widehat{m_{nk}}^{-1}(g)])\\
&= \widehat{m_{n-1j}}^{-1}(S_n).\\
S_{n-1} &= g \vee [\bigvee_j \widehat{m_{n-1j}}^{-1}(g) \vee [\bigvee_k \widehat{m_{n-1j}}^{-1}(\widehat{m_{nk}}^{-1}(g))]]\\
&= g \vee [\bigvee_j h_{n-1j}].
\end{aligned}
$$

…(以下同様)

■

定理 5.1は，各マクロの構造汎化とマクロテーブル上の位置がわかれば，マクロの変数汎化と，解決順序を安価に求めることができることを保証する．必要な単一化と最小汎化の回数はマクロの総数に対して線形である．このように，マクロテーブルの学習においてはマクロの構造汎化とマクロテーブル上の位置を同定することが本質的である．

$$e_1 = \overbrace{r_0 \circ u_8 \circ l_7 \circ d_6}^{m_{68}},$$

$$e_2 = \overbrace{\underbrace{r_0 \circ u_8 \circ l_7 \circ d_6}_{(m_{68})} \circ \underbrace{d_0 \circ r_2 \circ u_1 \circ l_8}_{(m_{18})}}^{m_{17}}.$$

図 5.4: An example set.

5.3 質問の利用

5.3.1 学習器における例題の役割

拡張 EBL をそのまま利用した最も素朴な学習器の手順を考え，学習器における例題の役割について考察する．学習器が利用可能な情報は正例のみとする．教師からは逐次的に例題が与えられる．学習器は，例題とそれまでに作られた仮説の間に最小 EBG 汎化を生成する．同時に汎化とその剰余との間にマクロテーブル上の順序関係を決定する．定理 5.1にしたがって，解決順序を計算して，それがマクロテーブルであることを確かめる．制約を満足するマクロテーブルは複数存在するが，最も利用度の高い仮説を選択する．ここで，最小 EBG の剰余とは，構造汎化された“余り”の説明構造である．例えば，2 つの説明構造を最小 EBG 汎化すると 2 つの剰余が生成され，これらがマクロならば最小 EBG はその剰余よりも前の行にある．

正例のみからの学習では，偶然の一致の影響が大きく，学習の成功は教師側のカリキュラムに大きく依存する．8 パズルについて例を示す．図 5.4の例題集合を考える．m_{17}はマクロ $m_{68} \circ m_{18}$の合成

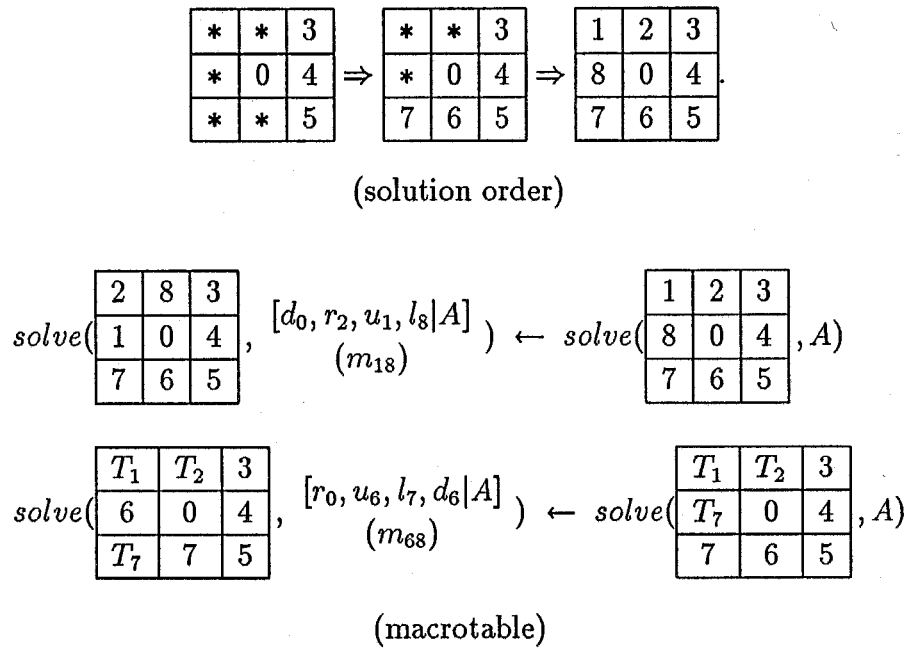


図 5.5: An incorrect macrotable learned from positive examples.

からなる．m_{68}を最小 EBG とする仮説が最も利用度の高い仮説である．m_{68}はm_{18}よりも本来前の行にあるので，これは誤った仮説である．しかしながら，タイル 5,6,7 の偶然の一致により，この仮説は解決順序の制約を満足する．生成された誤ったマクロテーブルを図 5.5 に示す．

このように，与えられた正例のみからマクロテーブルを学習することは仮説の正しさの点から問題がある．したがって，正しいマクロテーブルを学習するためには正例以外に情報を補うことが必要である．

5.3.2 membership query

正例以外に membership query が利用可能な場合を考える．membership query [1] [21] とは，ある例が学習すべき概念の正例であるか負例であるかを学習者が教師に質問する形式の学習プロトコルの一種である．

定義 5.3 (membership query)
マクロテーブル M に対して，初期状態 s とその解 m に関する *membership query* は，s の M による解が m のとき *yes*，そうでないとき *no* を返す述語である． ■

解 m に対応する初期状態 s は，$m^{-1}(s)$ によって求めることができる．以下，初期状態を省略して，m が M の member であることを $m \in M$ と書く．

membership query の利用のアイデアは次の 2 つの事実に基づく．

マクロテーブル M において，

1. m が M のマクロならば，$m \in M$.
2. m_1 が m_2 より前にあるならば，$m_1 \circ m_2 \in M$.

第 1 の事実から，与えられた例題をあらかじめマクロの合成に分解する操作が考えられる．第 2 の事実から，マクロの順序関係を定める操作が考えられる．次にこれらの操作を行うための概念を定義する．

定義 5.4 (分解 (decomposition))
m のマクロテーブル M に関する分解 p とは，条件 $m_1 \circ \cdots \circ m_n = m$, $m_i \in M$ を満たすマクロの列 $[m_1, \cdots, m_n]$ である．m の 2 つの分

membership query		guess about ordering of macros and new composite macros
$m_1 \circ m_2$	$m_2 \circ m_1$	
no	*no*	$m_1 = m_2$ (in the same row).
yes	*no*	1. $m_1 < m_2$ (m_1 is before m_2), or 2. $m_1 = m_2$ and $m_1 \circ m_2$ ($> m_i$) exists.
yes	*yes*	1. $m_1 < m_2$ and $m_2 \circ m_1$ ($> m_2$) exists, or 2. $m_2 < m_1$ and $m_1 \circ m_2$ ($> m_1$) exists, or 3. $m_1 = m_2$ and 3a. $m_1 \circ m_2$ ($\geq m_i$) and $m_2 \circ m_1$ ($> m_i$) exists, or 3b. $m_2 \circ m_1$ ($\geq m_i$) and $m_1 \circ m_2$ ($> m_i$) exists.

図 5.6: The serialization table.

解 p_1 と p_2 に対して，p_1 の各マクロが p_2 のマクロの分解であるとき，p_1 は p_2 より細かいという．最も細かい分解を極大分解という． ■

ある例題の解を生成した正しいマクロの組合せは，明らかに 1 つの分解であるが，それは極大分解であるとは限らないことに注意されたい．前節で述べたように，m_{17} は $m_{68} \circ m_{18}$ に極大分解されるが，これは正しいマクロの組合せではない．m_{17} のようなマクロを大きさ 2 の合成マクロ (composite macro) という．後に議論するように，合成マクロの存在は学習器の計算量に大きく影響する．一般に最大分解が存在するかどうかは知られていないが，8 パズルではすべての問題に最大分解が存在することが知られている．

定義 5.5 (直列化テーブル (serialization table))
m_1 と m_2 が M のマクロであるとき，図 *5.6*に示すテーブルを直列化テーブルという．ここで，$m_1 < m_2$ はマクロ m_1 が m_2 より前の行に

あることを示す. ■

直列化テーブルは，2つのマクロの2通りの合成に対するmembership queryの結果に基づいて，可能な順序関係を列挙した表である．直列化とは，直列化テーブルに基づいて各マクロに順序関係を推測する過程，およびその結果をいう．可能な順序関係がすべて列挙されていることについては後に考察する．

5.4 学習器の構成

前節で導入した分解と直列化テーブルによってmembership queryを利用する学習器について考える．教師から逐次的に与えられる例は，分解によって前処理される．分解とそれまでの仮説を直列化テーブルに基づいて併合する．ただし，最大分解は正しい分解とは限らないので，正しい分解が得られるまで探索を繰返す必要がある．この手順はPrologプログラムとして次のように記述される．

$$
\begin{aligned}
&learn(CurMT) \leftarrow \\
&\quad example(Ex), \\
&\quad decompose(Ex, Max, P), \\
&\quad serialize(CurMT, Max, P, NxtMT), \\
&\quad !,\ learner(NxtMT).
\end{aligned}
$$

$example/1$ は例題を無作為に生成する．$decompose/3$ は例題の分解を生成する．Max は最大分解に現われるマクロのリストである．分解 $P = [m_1, \cdots, m_n]$ には，マクロテーブル上の順序関係 $m_1 < \cdots < m_n$ が自動的に仮定される．直列化が失敗して後戻りが生じると，最大分解からはじめて細かい順に分解を次々と生成する．$serialize/4$ は，直列化テーブルを用いて現在までのマクロテーブル $CurMT$ に最大

incorrect	$m_{18} \circ m_{68}$	m_{68}	m_{18}
$m_{18} \circ m_{68}$	*no*	*no*	*no*
m_{68}	*yes*	*no*	*yes*
m_{18}	*no*	*yes**	*no*

correct	$m_{68} \circ m_{18}$	m_{18}	m_{68}
$m_{68} \circ m_{18}$	*no*	*no*	*yes*
m_{18}	*no*	*no*	*yes*
m_{68}	*no*	*yes**	*no*

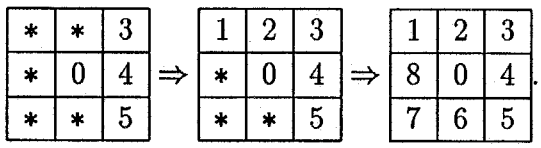
図 5.7: A membership query table.

分解のマクロ M と分解 P を併合して新しいマクロテーブル $NxtMT$ を生成する．新しいマクロが推測されたときは，*serialize*/4 を再帰的に起動して併合する．直列化テーブルは 7 つの節で記述され，結果的に深さ優先探索が行なわれる．直列化が成功すると次の例題に tail-recursion で移行する．以上のプログラムは骨格を示したもので学習の停止条件が無視されているが，実装上は例題の数の上限を与えて停止させる．

5.5 適用例

前節で構成した学習器を Sun4 上の SICStus Prolog 上に実装した．8 パズルおよび 5 パズル (8 パズルの 2×3 版) についてマクロテーブルの獲得実験を行った．

図 5.7に，正例のみからでは失敗した例題 (図 5.4) に対する実行過程の一部を示す．表の ij 要素は $m_i \circ m_j$ に対する membership query の結果である．*yes** は，直列化テーブルによってそれが合成マクロ



(solution order)

$$solve\left(\begin{array}{|c|c|c|}\hline 1&2&3\\\hline 7&0&4\\\hline 6&8&5\\\hline\end{array}\right),\ \underset{(m_{68})}{[r_0,u_8,l_7,d_6|A]}\ )\ \leftarrow\ solve\left(\begin{array}{|c|c|c|}\hline 1&2&3\\\hline 8&0&4\\\hline 7&6&5\\\hline\end{array}\right),A).$$

$$solve\left(\begin{array}{|c|c|c|}\hline 2&T_2&3\\\hline T_8&0&4\\\hline 1&T_6&5\\\hline\end{array}\right),\ \underset{(m_{17})}{[r_0,u_8,l_7,d_6,\ d_0,r_2,u_1,l_8|A]}\ )\ \leftarrow\ solve\left(\begin{array}{|c|c|c|}\hline 1&2&3\\\hline T_2&0&4\\\hline T_6&T_8&5\\\hline\end{array}\right),A).$$

$$solve\left(\begin{array}{|c|c|c|}\hline 2&T_2&3\\\hline 1&0&4\\\hline T_7&T_6&5\\\hline\end{array}\right),\ \underset{(m_{18})}{[d_0,r_2,u_1,l_8|A]}\ )\ \leftarrow\ solve\left(\begin{array}{|c|c|c|}\hline 1&2&3\\\hline T_2&0&4\\\hline T_7&T_6&5\\\hline\end{array}\right),A).$$

(macrotable)

図 5.8: The correct macrotable.

として解釈されたことを示す．誤った順序付け (上段) では，この後 $m_{18} \circ m_{18} \circ m_{68}$ (本来は存在しないマクロ) に関する前後関係が矛盾して，直列化に失敗する．正しい順序付け (下段) は矛盾しない．結果的に得られる正しいマクロテーブルを図 5.8に示す．

例題を無作為に発生させる実験を 20 回繰返した結果，5 パズルでは，各例題について約 2 ～ 3 分かかって，5 ～ 6 個目の例題までに元のマクロテーブルを獲得した．同様に 8 パズルでは，約 2 ～ 3 時間かかって，13 ～ 16 個目の例題までに元のマクロテーブルを獲得した．

5.6 考察

5.6.1 学習器の正しさ

分解が完全であることは明らかである．学習器の正しさは，直列化テーブルに依存する．直列化テーブルが完全であるとは可能な順序付けをすべて列挙できることである．直列化テーブルが健全であるとは教師のマクロテーブルとは異なるマクロテーブルを生成しないことである．

まず，完全性について確かめる．直列化テーブルに現われる or 選択肢の数が，十分な組合せを列挙していることは明らかである．推測される合成マクロと元のマクロの順序関係の必要性について，次の定理が成立する．

定理 5.2 (合成マクロと元のマクロの順序関係)
n (≥ 2) 個のマクロを結合したマクロを m，それらのうち最も前の行にあるマクロを m_pとするとき，$m_p \leq m$ である．特に，m_pと同

じ行にあるマクロがm_pのみであるときは，$m_p = m$である．

証明 マクロを写像とみなして，その不変部分と変化部分の関係から証明する．

変数，定数および関数の出現位置とは，次のように再帰的に定義される数のリストである．

1. 変数vにおけるv，定数cにおけるcおよび項$f(t_1, \cdots, t_n)$における関数f/nは，[0]の位置に現われる．

2. xがt_iにおいてpの位置に現われるとき，項$f(t_1, \cdots, t_n)$において$[i|p]$の位置に現われる．

2つの項を同時に訪問して同じ位置に出現する変数，定数および関数を比較する．それが同一であるときその出現位置は不変であるといい，異なるときは変化するという．項xとyに対して，不変の出現位置の集合を${}^i(x, y)$，変化する出現位置の集合を$\overline{{}^i(x, y)}$で表わす．

マクロと解決順序の間には明らかに次の関係が成立つ．ここで，マクロm_{ij}に対しては，${}^i(h_{ij}, b_{ij})$を${}^i m_{ij}$，解決順序S_iに対しては，${}^i(S_i, g)$を${}^i S_i$等と略記する．

$$
\begin{aligned}
&{}^i S_i \subset {}^i S_j \quad (i < j),\\
&{}^i S_i \subset {}^i m_{ij},\\
&{}^i S_{i+1} - {}^i S_i \subset \overline{{}^i m_{ij}}.
\end{aligned}
$$

第3式は，i番目に満たされるべき副目標が必ず第i行のマクロによって変化することを意味する．

2つのマクロm_{ix}とm_{jy} $(i \leq j)$の合成$m = m_{ix} \circ m_{jy}$が第k行のマクロであるとする．位置集合${}^i S_i$は明らかに${}^i m$に含まれる．位

置集合 $p = {}^{i}S_{i+1} - {}^{i}S_{i}$について考える．$i < j$のとき，$p \subset \overline{{}^{i}m_{ix}}$かつ $p \subset {}^{i}m_{jy}$なので，$p \subset \overline{{}^{i}m}$である．このとき，$k = i$である．$i = j$のときは，$p \subset \overline{{}^{i}m_{jy}}$なので，それぞれのマクロの変化部分が打消し合って$p \subset {}^{i}m$となるような例を作ることができる．このとき，$k > i$である．ゆえに，$k \geq i$である．

この結果は，3個以上のマクロの合成にも簡単に拡張することができる．いずれも，最も前の行にあるマクロで充足される副目標に対応する位置が合成では変化することを利用する． ■

定理 5.2は，合成マクロと元のマクロのマクロテーブル上の順序関係が解決順序によって制約されることを意味する．直列化テーブル (図 5.6) の順序関係はこれによって定めたものである．ただし，$m_1 < m_2$に対して$m_1 = m_2 = m_1 \circ m_2$を展開形といい冗長であるので排除してある．

直列化テーブルの健全性について，次の定理が成立する．

定理 5.3 (直列化テーブルの健全性)
目標のどの部分も互いに等しくなく，大きさ 4 以上の合成マクロが存在しないとき，任意の 2つのマクロに対して，それらのマクロテーブル上の順序関係は，直列化テーブルの繰返し適用によって一意に決定できる．

証明 直列化テーブルは推測される順序関係に関して曖昧性を持っている．2つのマクロm_1とm_2に対して直列化テーブルの1回の適用では，次の2種類の矛盾した順序関係を部分として持つマクロテー

ブルを生成し得る.

$$M_1 \supseteq \boxed{m_1 \mid m_2}$$

$$M_2 \supseteq \boxed{\begin{array}{c|c} m_1 & m_1 \circ m_2 \\ m_2 & \end{array}}$$

簡単のために，マクロテーブルは左が前の行になるように示す．これを矛盾する部分テーブルとよぶ.

2つのマクロテーブルが，任意の membership query に同一の答えを発するときそれらは等価であるという．直列化テーブルが完全でないとき，矛盾する部分テーブルを含む等価なマクロテーブルが存在する．まず，2行からなるそのようなマクロテーブルが存在しないことを示す.

命題 5.1 目標のどの部分も互いに等しくなく，大きさ4以上の合成マクロが存在しないとき，矛盾する部分テーブルを含む2行からなる等価なマクロテーブルは存在しない.

証明 M_2の member である $m_1 \circ m_1 \circ m_2$ および $m_2 \circ m_1 \circ m_2$ が M_1の member ではないことに着目して，これらを member とするように M_1に新しいマクロを追加する．追加されるマクロは定理 5.1 を満たさなければならない．更新された M_1の member で，M_2の member でないものを探し，M_2に追加することを試みる．この手順を繰返して，可能な追加方法を列挙するプログラムを実装して，大きさ3の合成マクロまでは等価なマクロテーブルが存在しないことを確認した．プログラムの実行結果については省略する．ただし，目標に等しい部分があるときは，矛盾する部分テーブルを含む等価なマクロテーブルを構成することができる.

次に，この命題をを n 行の場合に拡張する．

命題 5.2 目標のどの部分も互いに等しくなく，大きさ4以上の合成マクロが存在しないとき，矛盾する部分テーブルを含む等価なマクロテーブルは存在しない．

証明 あるマクロテーブルについて，次のように2つの行 L_1 と L_2 から1つの行 $L_1 \times L_2$ を得る操作を行の展開という．

$$L_1 \times L_2 = L_1 \cup L_2 \cup (L_1 \circ L_2).$$

展開された行を持つマクロテーブルは元のマクロテーブルと等価であるが，冗長である．

もし，n 行からなる等価なマクロテーブルが存在すると仮定すると，それらを展開することによって2行の等価なマクロテーブルが存在する．これは命題 5.1 に反する．

定理 5.3 は命題 5.2から直接導かれる． ■

定理 5.3において，第1の前提は，例えば，8パズルでは同じタイルが存在しないという条件である．これが満たされない場合には，マクロテーブルは一意に決定できない．第2の前提は，4つ以上のマクロの合成からなる合成マクロが存在しないことである．8パズルの合成マクロは大きさ3で，これらの条件を満足する．大きさの制限は証明の手法によるが，本論文の目的には十分である．

以上のことから，membership query を利用する学習器は正しいマクロテーブルを獲得できることが示された．

5.6.2 学習器の計算量

学習にかかる計算量について考察する．最大分解が存在するものと仮定する．領域について，マクロテーブルの行数を n，マクロの総数を m，合成マクロの大きさの最大値を kとする．与えられる例題の大きさを lとする．計算量は membership query の回数を測度とする．

まず，合成マクロが存在しない最も効率的な場合について考察する．

例題のすべての部分列について l^2回の membership query を発して，*yes* の部分列にリンクをはる．最大分解はこのグラフ上の最短パスであるから，これを求める計算量は $O((l^2)^2)$ である．

直列化では，分解でえられた順序関係を現在のマクロテーブルと併合する．最大分解の大きさは $O(n)$，現在のマクロテーブルの大きさは $O(m)$ である．*yes-no* 行の or 選択肢によって $O(mn)$ 回の membership query のそれぞれについて同じ数の存在しない合成マクロが導入される．これらの矛盾が検出されるまでに，再帰的に呼出された直列化では，ひとつにつき $O(m)$ 回の membership query を必要とする．ゆえに，直列化には $O(mn \times m)$ 回の membership query が必要である．

最大分解は正しい分解であるから，総計算量はこれらの和となる．ひとつの例題に対して，$O(m^2n + l^4)$ 回の membership query が必要である．

合成マクロが存在するときに必要な計算量は厳密には知られていない．ここでは，前章で構成した単純な後戻りに基づく学習器にし

たがって計算量を推定する.

最大分解の大きさは $O(kn)$ である. 最大分解は正しい分解とは限らないので, 正しい分解が得られるまでには可能な $O(2^{kn})$ 個の分解を最大分解からはじめて細かい順に調べなければならない.

直列化では, 過剰に合成されたマクロが生成されて矛盾が検出されるまで, 分枝度 4(*yes-yes* 行) 深さ kの探索を行わなければならない. 各段階では $O(m)$ の membership query が必要である. ゆえに, 直列化には $O(mkn \times m^{4k})$ 回の membership query が必要である.

総計算量はこれらの積となる. ひとつの例題に対して, $O(2^{kn} \times mkn \times m^{4k} + l^4)$ の membership query が必要である. $k = 1$ のときのオーダーが一致しないのは, 合成マクロが存在しないときは, 分解への後戻りがないことと, 分枝度 4 の *yes-yes* 行が使われないことによる.

合成マクロが存在するときの計算量が膨大である理由は, 現在の手続きでは正しい分解を求めるために生成-検査法によって可能な分解をすべて列挙することにある. 分解の計算量は, 直列化の情報を利用することによってかなり減少できることが期待できる. 直列化の計算量は利用可能な情報が membership query のみであるため, 最悪の場合はやはり kに対して指数的である. このように, kは領域固有の学習の困難さを現わす因子である. しかし, 8 パズル等の実験では, 学習されたマクロテーブルによって直列化の or 選択肢が枝刈りされてゆくことが観察されており, このことを利用した効果的な方法が存在することが予想される. これらの改良は今後の課題である.

5.6.3 より強力なメタ領域理論の利用

前節で考察した学習の困難さについて，その原因を考える．学習器は直観的に必要なもの以上に不自然な membership query を発生することが観察される．これらは誤った順序付けを網羅的に排除するために用いられる．誤った仮説を排除するために膨大なデータを必要とすることは，SBL に特徴的な困難である．

EBL において SBL 的な欠点が現われる原因は，利用可能な領域知識の不足にあると考えられる．現在，領域知識は必要最小限の作用素定義のみである．本研究で得られたようなマクロテーブルに関する知識が“メタ領域知識”として利用可能であれば，より少ない知識から正しいマクロテーブルを獲得することが可能となると予想される．特に，問題領域が直列分解可能な副目標を持つことがわかっており，解決順序が与えられているときには，membership query なしに正しい分解を求めることができる．このことに関しては試験的に好ましい結果が得られている．

5.7 むすび

本章では，直列分解可能な副目標を持つ問題を対象として，メタ領域知識と membership query を利用した学習器の構成と評価を行なった．すなわち，論理プログラムに形式化されたマクロテーブルから導出される基本的な性質のみをメタ領域知識として与え，membership query を繰り返すことにより元のマクロテーブルが獲得されることを示した．

しかし，本章での適用例では，学習器の EBL 的側面よりも SBL

的側面が強調され，学習器の本質的な性能を示すには不適切であったといえる．より一般の合成型問題領域では，利用可能なメタ領域知識は各問題毎に完全ではないにしても，十分な蓄積があると見るのが自然であり，そのような問題領域を取り上げてこそ本学習器の良さが発揮されるものと思われる．

Chapter 6

結 論

本研究では，例題からの汎化に基づく逐次的な知識洗練化のためにEBLを複数例題上に拡張した拡張EBLの枠組みを提案した．まず，EBLの操作性問題の解決に焦点を当て，複数の例題の説明構造から生成される汎化空間と利用度最大化と後戻り最小化からなる操作性規範の関係について理論的に考察した．操作性規範の部分的単調性を明らかにし，操作性の高い汎化の探索に有用な概念として最小EBGの概念を導入した．次に，最小EBGを利用する素朴な学習器を実装して，さまざまな領域に適用し，その有用性と限界を示した．さらに，それらの限界を克服するために，問題固有の情報をメタ領域理論と質問の形式で利用する学習器の洗練化を提案した．この手法を問題解決マクロテーブルの獲得問題に適用して，正確かつ効率的な学習器であることを実証した．

拡張EBLは，SBLとEBLの相補的な特徴の統合化を考慮して提案された枠組みである．したがって，拡張EBLの枠組みは，領域理論をバイアスとして，信頼性以外の操作性規範も考慮するようにSBLを拡張した枠組みとしてとらえることもできる．すなわち，拡

張 EBL は“拡張 SBL”とみなすこともできる．拡張 SBL としての特徴は，通常の SBL におけるバイアスのような不透明な部分を領域理論と操作性規範によって明確に表現できることにある．操作性規範の拡張にともなって，より有用なマクロを生成し得るが，バージョン空間法を成功させていた単調性は失われ，探索は一般には困難な課題となる．この問題を回避するためには，メタ領域知識や質問の導入が必要であることを示した．しかし，メタ領域知識と質問を導入しても解決順序のような決定的な情報が欠けている時には膨大な membership query が必要となることが観察されるが，このことは弱いバイアスの下でバージョン空間の収束に大量のデータが必要であることに対応する．このように，拡張 EBL では SBL と EBL の相補的な特徴が良く統合化されていると考えられる．

今後，大きく分けて次の 3 つの発展的な課題が考えられる．

1. 知識表現の充実

   ハノイの塔にみられる知識表現への敏感性の原因には，論理プログラムの表現力の貧弱性に起因するものがある．例えば，“塔の交換規則”などは，項の等価性の導入によって簡単化される．このような知識表現の拡張によって拡張 EBL の適用範囲の拡大が期待される．

2. 他の操作性規範の考慮

   本研究で考慮した操作性規範は，おもに問題解決過程の効率性に関わるものである．解の信頼性の考慮によって，EBL の不完全領域理論問題の解決が期待される．解の最適性は問題解決過程の効率性と相反する要求であることが多い．このため，解

の最適性の取り扱いには困難が生じることが予想される．しかしながら，拡張 EBL を実際の問題に応用するためには，要請されるすべての操作性規範についての考察が必要であると考えられる．

3. メタ領域知識の充実

本研究で取り扱ったメタ領域知識は問題解決マクロテーブルの獲得に特化されたものである．今後さまざまな領域でメタ領域知識を構成して収集することが必要と考えられる．このことによって，メタ領域知識のライブラリ化，学習器の洗練化の自動コンパイルなどへの発展が期待される．

Bibliography

[1] Angluin,D. *Learning Regular Sets from Queries and Counterexamples*, Information and Computation, Vol.75, pp.87-106 (1987).

[2] DeJong G. and Mooney,R. *Explanation-Based Learning : An Alternative View*, Machine Learning, Vol.1, pp145-176 (1986).

[3] Haussler,D. *Learning Conjunctive Concepts in Structural Domains*, Proc.of AAAI 87, pp.466-470 (1987).

[4] Hirsh,H. *Explanation-Based Generalization in a Logic-Programming Enviroment*, IJCAI 87, pp.221-227 (1987)

[5] Keller,R.M. *Defining Operationality for Explanation-Based Learning*, Artificial Intelligence, Vol.35, pp.227-241 (1988).

[6] 小林重信, 知識獲得と学習, 人工知能技術動向に関する講演会資料 (1988).

[7] Korf,R.E *Macro-Operators : A Weak Method for Learning*, Artificial Intelligence, Vol.26, pp.35-77 (1985).

[8] Laird,J.E. Rosenbloom P.S. and Newell,A. *Chunking in Soar ; the Anatomy of a General Learning Mechanism*, Machine Learning, Vol.1, pp.11-46 (1986).

[9] Mitchell,M.T. *Generalization as Search*, Artificial Intelligence, Vol.18, pp203-226 (1982).

[10] Mitchell,T.M. Utgoff P.E. and Banerji,R. *Learning by Experimentation : Acquiring and Refining Problem-Solving Heuristics*, Machine Learning, Michalski et al.ed., Tioga pub. (1983).

[11] Mitchell,T.M. Keller R.M. and Kedar-Cabelli,S.T. *Explanation-Based Generalization : An Unifying View*, Machine Learning, Vol.1, pp.47-80 (1986).

[12] Mostow,D.J. *Machine Transformation of Advice into a Heuristic Search Procedure*, Machine Learning, Michalski R.S. et.al. ed., Springer-Verlag (1984).

[13] Mostow J. and Bhatnagar,N. *Failsafe - A Floor Planner that Uses EBG to learn from its Failures*, AAAI 87, pp.249-255 (1987)

[14] Plotkin,G.D. *A Note on Inductive Generalization*, Machine Intelligence 5, pp.153-163 (1970).

[15] Prieditis A.E. and Mostow,J. *PROLEARN : Towards A Prolog Interpreter that Learns*, Proc.of AAAI 87, pp.494-498 (1987).

[16] Rosenbloom P.S. and Laird,J.E. *Mapping Explanation-Based Generalization onto Soar*, Proc.of AAAI-86, pp.561-167 (1986).

[17] Segre,A.M. *On the Operationality/Generality Trade-off in Explanation-Based Learning*, Proc.of 9th IJCAI, pp.242-248 (1987).

[18] Shavlik J.W. and DeJong,G.F. *BAGGER : An EBL System that Extends and Generalizes Explanations*, Proc.of AAAI 87, pp.516-520 (1987).

[19] Shavlik,J.W. *Acquiring Recursive Concepts with Explanation-Based Learning*, Proc.of 11th IJCAI, pp.688-693 (1989).

[20] Utgoff,P.E. *Shift of Bias for Inductive Concept Learning*, Dr.thesis, Ritgers Univ., 1984.

[21] Valiant,L.G. *A Theory of the Learnable*, CACM, Vol.27, No.11, pp.1134-1142 (1984).

[22] 山村雅幸, 李相龍, 小林重信, **SBL と EBL の融合による知識獲得**, 第 7 回知識工学シンポジウム資料 (1988).

[23] 山村雅幸, 小林重信, **操作性規範に基づく SBL と EBL の統合化**, 計測自動制御学会第 1 回知識工学部会資料, (1988).

[24] 山村雅幸, 小林重信, **操作性規範を考慮した複数例題下での拡張 EBL**, 第 8 回知識工学シンポジウム資料 (1988).

[25] 山村雅幸, 小林重信, **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).

[26] Yamamura M.and Kobayashi,S. *An Augmented EBL for Problem Solving Macrotable*, Proc.of SICE Annual Conference, pp.1377-1380 (1989).