知識工学と人工知能 60 － 9
（１９８８．９．２１）

完全因果性によるマクロオペレータの選択的学習

山田誠二　　辻三郎
大阪大学・基礎工学部

与えられた基本オペレータで問題を解いた結果であるオペレータ・シーケンスの部分シーケンスを一つのマクロオペレータに合成・一般化し、後の問題解決を効率的に行なうのがマクロオペレータ学習である。一つのオペレータ・シーケンスから考えられるマクロオペレータの候補は非常にたくさんあるので、どのように選択的にマクロを抽出するかが問題となる。本報告では「完全因果性」というヒューリスティックによって、役に立つマクロだけを選択的に抽出し、それを「説明に基づく一般化：EBG」により一般化して一つのマクロに合成する手法を示す。また、本手法の有効性をSTRIPSを用いたロボットの行動計画の分野で検証する。

Selective Learning of Macro－operators with Perfect Causality

Seiji YAMADA, Sabro TSUJI
Osaka University
Machikaneyama 1－1, Toyonaka, Osaka, 560, JAPAN

The main problem in macro－operator learning is how the learning system can select the valid macro operators. To cope with this problem, we suggest the method of learning macro operators with Perfect Causality : the heuristics to select valid macro operatos only. We developed PiL2 system that can aquire useful macro－operaotrs selectively with the Perfect Causality and generalize them with EBG (Explanation－Based Generalization) method. And we made the experiment in the robot planning.

1, はじめに

筆者（山田）はこれまで問題解決における戦略知識を学習により獲得する方法を模索しそのフレームワークとして，問題解決における戦略知識学習システム：PiL [1] を構築してきた．PiLは問題状態を変化させる基本オペレータと問題解決の履歴である解法例を入力とし，「説明に基づく一般化：EBG」（Explanation−Based Generalization）[2] により解法例を一般化することで，基本オペレータをどのように用いれば良いのかという戦略知識を獲得することが可能である．PiLで獲得される戦略知識には個々の基本オペレータに関するものと複数の基本オペレータを合成したマクロオペレータの二種類があった．このマクロオペレータは複数ステップを一度に実行することができ効率の向上が可能である．しかし，通常一つの解法例からは多数のマクロの候補が考えられるので，それらをすべて蓄えていたのでは爆発的に増大してしまい，その結果学習無しシステムよりもパフォーマンスが低下してしまうことが報告されている [3]．このような問題に対処するには，数多くのマクロの候補のうちから役に立つと思われるものだけを選択的に取り出して蓄積していくことが必須である．筆者はこのマクロオペレータの抽出基準として，「完全因果性」というヒューリスティックを提案し，PiLシステム上での1次・2次・分数・指数・対数方程式の解法学習においてその有効性を確認した [4]．また，筆者はマクロオペレータによる問題解決の効率向上を前提として操作可能（operational）な解決可能概念（SOLVABLE concept）を定義し，それを用いた「直接解決可能性に基づく一般化：DSBG」を提案した [4]．

しかし，数式処理の分野で有効であった完全因果性が果して他の分野にも適用できるのかという疑問が残った．そこで今回，より広範囲の問題領域で適用可能にするために，過去の方程式解法での実験結果を保証する範囲で完全因果性の定義を拡張し，さらにその普遍性を検証するために汎用マクロオペレータ学習システム：PiL2を構築して，ロボットの行動計画の分野においても完全因果性によるマクロオペレータの選択的学習が可能であることを検証する．本報告ではまず最初にPiL2の概要とその構成モジュールの働きについて述べ，そして完全因果性の定義，それを用いたマクロオペレータ抽出アルゴリズムと抽出されたオペレータシーケンスをEBGにより一般化する手法を説明し，最後に実験方法・結果とその評価及び関連研究を示す．

2, PiL2の概要

PiLシステムは問題解決モジュール及びマクロオペレータを獲得・一般化する学習モジュールが数式のリスト表現に依存したものであった．そこでより一般的な問題状態表現を扱えるようにと考えられたのが汎用マクロオペレータ学習システム：PiL2である．PiL2のシステム構成をFig. 1に示し，PiLとの差異であるところのPiL2の特徴を以下に述べる．

1）問題状態表現が，述語表現（リテラル，ルール）である．

汎用学習システムを目指すために，現在問題状態表現として最も一般的と考えられる述語表現を用いる．

2）マクロオペレータ指向である．

個々のオペレータについての戦略知識を獲得することをやめて，マクロオペレータのみを獲得する．

3）拡張された完全因果性を用いる．

残念ながらPiLで用いていた完全因果性の定義は不十分で拘束が強すぎ，そのまま他の分野に適用することは難しい．よって過去の実験結果 [1] [4] を保証する範囲で完全因果性の定義を拡張する．

以降，各モジュールの働きとマクロオペレータの選択的抽出の方法についてより詳細に述べていく．

3, 問題解決モジュール：STRIPS

PiL2は問題解決モジュールとしてロボットの行動計画で有名なSTRIPS [5] [6] を用いる．ロボットが活動する世界の記述である問題状態は述語表現であるリテラルとルールで表わされ，その問題状態を変化させるのがオペレータである．STRIPSは問題の初期状態，目標状態及びオペレータ（以後，この入力であるオペレータを基本オペレータと呼ぶ）を与えられ，その基本オペレータを使って問題状態を変化させ，目標状態を満足するような問題状態を導くオペレータシーケンス（これがロボットの行動計画）を自動生成するものである．基本オペレータは

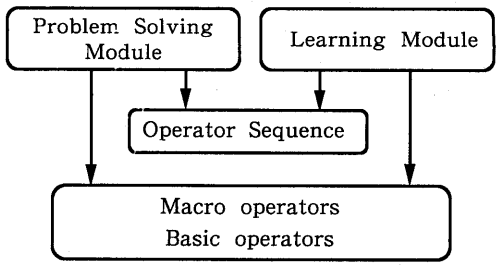


Fig.1 PiL2の構成図

```
gotob (BX,RX) {Go to BX in RX}
   cond : [type (BX,object),inroom (BX,RX),
           inroom (robot,RX]
   delete : [nextto (robot,__)]
   add : [nextto (robot,BX)]
   main−effect : [ ]
pushb (BX,BY,RX) {Push BX to BY in RX}
        [type (BX,object),type (BY,object),
         pushable (BX),nextto (robot,BX),
         inroom (BX,RX),inroom (BY,RX)]
        [nextto (robot,__),nextto (BX,__),]
        [nextto (BX,BY),nextto (robot,BX)]
        [nextto (BX,BY)]
```

Fig.2 基本オペレータ

Fig. 2のようなもので，条件リスト，削除リスト，追加リスト，主要効果リストから構成されている．それぞれのリストの意味は条件リストが満足されると現在の問題状態から削除リストと単一化可能なリテラルを取り除き，追加リストのリテラルをつけ加える．条件リストが現在の問題状態で満足されるか否かは，導出法により求める．主要効果リストはそのオペレータにより得られる効果の主要なものであり，空リストの場合は主要効果リストが追加リストと同じであることを意味する．これは関連したオペレータ（relevant operator）の探索［5］の際に用いられる．

なお，STRIPSには2つのバージョン［5］［6］があり，一つは学習無しの問題解決システム［5］であり，もう一つは学習可能な問題解決システム［6］でMACROPSという一般化されたマクロオペレータを獲得でき，それを三角表という表現で蓄え後の問題解決に役立てることができる．

以下にPiL2のSTRIPSの特徴を述べる．まず，重要な点はPiL2は学習無しのSTRIPSを単なる問題解決モジュールとして用いていることである．また，PiL2のSTRIPSは基本オペレータとマクロオペレータの2種類のオペレータを別の階層に蓄え，マクロオペレータを優先的に適用していく．まずマクロオペレータだけで探索していき行き詰まった場合に，基本オペレータの階層で探索を始める．縦型にはサブゴールの順序付けが必要であるがPiL2のSTRIPSは，・サブゴールの適用により満たされるリテラル数，・サブゴールの適用可能性，・繰り返されるサブゴールなどを評価して順序付け行なっている．さらに，領域固有の枝刈り用の知識も若干与えている．

4, マクロオペレータの選択的抽出とEBGによる一般化

nステップの解法シーケンスについてのマクロオペレータの候補数：MC（n）は順序関係を崩さない範囲でのすべての組合せであるからMC（n）$= \Sigma_{k=2}^{n} nCk$ となり，例えば10ステップだとマクロの候補は1013個にも昇る．つまり，解法シーケンスから考えられるマクロの候補は非常にたくさんあり，それらをすべて蓄えていたのではマクロが爆発的増大してしまう．よって，学習有りSTRIPS［6］のように重複しない限りマクロの候補をすべて蓄えるシステムでは，比較的少数の問題を解かせたときでさえマクロの増大により，問題解決の効率が学習無し問題解決システムよりも低下してしまうという大変興味深い現象がMinton［3］により報告されている．このようにマクロオペレータ学習において，マクロの増大をどのようにおさえ学習無しシステムよりも高い効率を維持するかが最重要課題である．マクロの候補の中には無意味なものが数多く含まれているのでそれらを取り除けば大幅にマクロを減少させることができる．しかし，数多くあるマクロオペレータの候補からどのように役に立つマクロオペレータを選択するかが問題となる．これに対する解決案としてはMinton［3］などの方法があるが，ここで筆者は「完全因果性によるマクロオペレータの選択」という別のアプローチを提案する．

4. 1 完全因果性によるマクロオペレータの選択

人間は問題解決においてマクロオペレータを用いていることは明かであるが，ではどのような基準でマクロオペレータを抽出しているのだろうか？筆者はここで完全因果性というマクロオペレータの選択基準であるヒューリスティックを提案する．完全因果性は一次方程式の解法のオペレータシーケンス（文献［1］参照）において人間が自然に使っているマクロオペレータを構成しているオペレータ間にはどのような関係があるのかを分析することによって筆者が帰納的に考えだしたものであり，以下のような仮説に基づいている．

1）因果関係のないオペレータシーケンスはマクロオペレータにならない．

2）シーケンス中のあるオペレータシーケンスの適用結果が他のオペレータの適用を保証しているとき，それらのオペレータ間には強い因果関係があるとし，強い因果関係があるものだけがマクロオペレータになる．

2）の強い因果関係が筆者が「完全因果性」と呼んでいるものである．完全因果性は以下のように再帰的に定義される．

＜完全因果性＞

いま対象となる例示化されたオペレータシーケンスをOPS＝{OP1・・OPn}，問題の初期状態をISとする．ISはリテラルの集合であり，またオペレータはOP1・・OPnの順で適用されたとする．このときOPS中の完全因果性の成り立つオペレータシーケンス：PCOPS＝{OPi・・OPj}（1≦i<j≦n）の任意の要素であるオペレータOPm（i≦m≦j）は以下の条件を満たす．

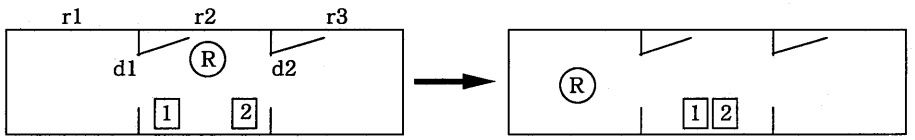
ISではOPmは適用不可能であり，かつISに{OPi・・OPm−1}を順に適用した結果の問題状態ではOPmが適用可能である．このとき{OPi・・OPm−1}とOPmには完全因果性があるという．

この条件は{OPi・・OPj}の適用がOPmの適用を保証しており，またPCOPS中の任意のオペレータOPmと{OPi・・OPm−1}間には必ず完全因果性が成り立つことを表わしている．実際のマクロの抽出は{OP1・・OPn}の部分シーケンス：{OPk・・OPn}（1≦k≦（n−1））それぞれについてOPkを含む完全因果性の成り立つ集合を調べ，そのうち最大のものだけをマクロオペレータとして抽出するということを行なう．その具体的なアルゴリズムをFig．3に示す．また，この完全因果性の定義はPiL［1］におけるものをより広範囲に対処できるように拡張したものであるが，PiLにおける各種方程式の実験結果［1］［4］を保証している．以上のマクロオペレータの抽出を以下に具体例で説明する．

例として，Fig．4のような問題とそれを解いたオペ

```
INPUT (IS,OPS)   {IS is initial problem state,
                  OPS= {OP1 ••• OPn}}
Let IOP be a set of OPS's appliable operaotrs
in IS
MOPS← [ ]
i←1
WHILE i≠n DO BEGIN
   Let RESULT be the problem state after OPi
   was applied to IS without checking condition
   MOP← [OPi]
   j←i+1
   WHILE j<n
      IF OPj ∉ IOP
        THEN IF OPj is appliable in RESULT
                THEN • RESULT←RESULT OPj
                                  applied
                     • assert OPj into MOP
      j←j+1
   END
   IF MOP≠ [OPi] THEN assert MOP into MOPS
   i←i+1
END
OUTPUT : MOPS is macro−operator sequences
```

Fig.3 マクロ抽出アルゴリズム



GOAL STATE : [nextto (box1,box2),inroom (robot,r1)]

IS＝ {f0 : type (robot,robot), f1 : type (box1,object), f2 : type (box2,object), f3 : type (d1,door),
f4 : type (d2,door), f5 : type (r1,room), f6 : type (r2,room), f7 : type (r3,room),
f8 : pushable (box1), f9 : pushable (box2), f10 : inroom (robot,r2), f11 : inroom (box1,r2),
f12 : inroom (box2,r2), f13 : status (d1,open), f14 : status (d2,open),
f15 : connects (d1,r1,r2), f16 : connects (d2,r2,r3)}

OP1 : gotob (box1,room2)
↓ [f1,f10,f11] [ ] [f17 : nextto (robot,box1)]
OP2 : pushb (box1,box2,room2)
↓ [f1,f2,f8,f11,f12,f17] [f17] [f18 : nextto (box1,box2),f19 : nextto (robot,box1)]
OP3 : gotod (door1,room2)
↓ [f3,f10,f15] [f19] [f20 : nextto (robot,d1)]
OP4 : gothrudr (door1,room1,room2)
↓ [f3,f5,f10,f13,f15,f20] [f10,f20] [f21 : inroom (robot,r1)]
GOAL : [f18,f21]

Fig.4 問題とその解決オペレータシーケンス

レータシーケンスが問題解決モジュールから与えられたとする．ここでOPS＝｛OP1, OP2, OP3, OP4｝であり，またこのシーケンスは各リテラルがユニークなラベルづけ（fn）をされている．まず最初にIOP＝［gotob, gotod］（引き数省略）が求まる．次にOP1：gotobが無条件にISに適用されその結果がRESULT＝［f0～f17］となり，MOP＝［gotob］とセットされる．次のOP2：pushbはIOPに含まれずかつその条件リスト：［f1, f2, f8, f11, f12, f17］がRESULTに含まれるので適用可能である．よって，pushbをRESULTに適用した結果でRESULTを更新し，MOP＝［gotob, push］になる．しかし，次のOP3はIOPの要素であり，またOP4はRESULTに適用できないので以上でi＝1のループは終了し，MOP≠［gotob］よりMOP＝［gotob, pushb］が一つのマクロになる．そして，i＝2で再びマクロの候補が調べられる．この例から得られる最終的な出力はMOPS＝［［gotob, pushb］, ［gotod, gothrudr］］である．この結果は因果関係のない独立したオペレータシーケンスが完全因果性により切り離されて抽出されることを示している．また，このシーケンスのステップ数は4なのでMC（4）＝11ケのマクロの候補が考えられるがそのうち2つだけが選択的に抽出されている．

4．2 もう一つの拘束条件

完全因果性によるマクロオペレータの選択によって，因果関係の希薄なマクロの生成が抑えられるがそれでもまだあまり有効とは思われないマクロが生成されることがある．それは完全因果性のある複数のマクロオペレータのシーケンスによって一つのマクロが生成されることである．例えば，macro（gotod, gothroudr），macro（gotod, gothroudr）のようなオペレータシーケンスが得られた場合，完全因果性によりこのシーケンス全体が一つの新しいマクロオペレータとして生成される．もしこのような既にあるマクロで構成されるマクロオペレータを蓄えていくとどうなるだろう．マクロオペレータの階層には個々のマクロとそのマクロを複数個組み合わせたマクロが混在することになる．これは基本オペレータとその組合せであるマクロオペレータが混在している状態と等しく，そのような状態では効率は向上しない場合が生じる．よって，《拘束条件R：現存のマクロによって構成できるマクロオペレータは生成しない》ことにする．

4．3 説明に基づくマクロオペレータの一般化とその合成

以上のようにして得られたマクロオペレータの候補であるオペレータシーケンスは，オペレータの引き数が全て定数化されたインスタンスであるのでこれを一般化しなければいけない．ここでは「説明に基づく一般化：EBG」［2］の手法で一般化を行なう．一般にEBGでは次の4つの要素が必要である．

1）目標概念（Goal Concept）
2）訓練例（Training Example）
3）領域知識（Domain Theory）
4）操作可能性の基準（Operationality Criterion）

マクロオペレータの一般化で上述のそれぞれの要素が何に対応するかを考える．まず，前節のようにして抽出された有効なマクロオペレータであるオペレータシーケンスはEBGにおける「説明木」にあたる．例としてFig. 4で抽出されたオペレータシーケンスによって構成される説明木をFig. 5に示す．訓練例，領域知識はそれぞれオペレータシーケンスの葉と枝にあたり．Fig. 5では点線の部分が訓練例に，それから基本オペレータgotod, gothrudrが領域知識に対応する．また，操作可能性の基準は完全因果性に対応すると考えられ（このことについては後に検討する），目標概念はLEX2［2］においてあるオペレータの適用が有効なときの条件を導くUSEFUL－OP（COND）と同様なものでこの例の場合は例えばuseful－macro－gotod－gothrudr（COND）とかける．

一般化過程の詳細は割愛するがMitchellらのEBG［2］によりFig. 5を一般化した結果がFig. 6である．これで一般化が完了したわけだが基本オペレータと同じシンタックスであるマクロオペレータを生成するには，この一

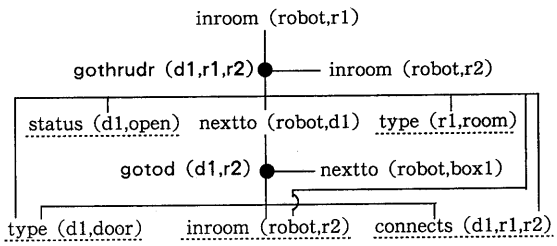


Fig.5 説明木

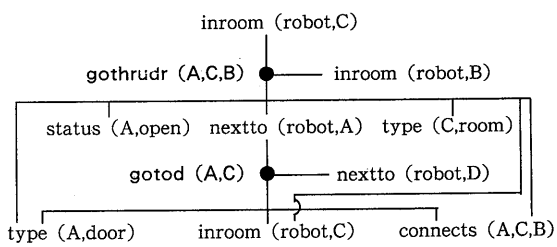


Fig.6 一般化された説明木

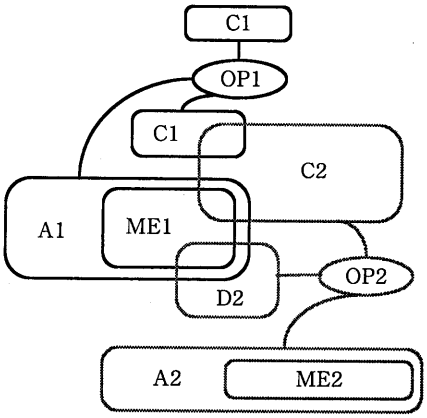


Fig.7 オペレータの合成

般化された説明木から条件リスト，削除リスト，追加リスト，主要効果リストを抽出しなければならない．それをFig. 7を用いて説明する．この図は二つのオペレータの一般化された説明木を表わしている．Cn, Dn, An, MEnはそれぞれ条件リスト，削除リスト，追加リスト，主要効果リストを表わす．そして，各オペレータの各リストは述語の集合と考え，以下のような集合演算を行なうことにより新しいマクロオペレータの各リストを生成する．MC, MD, MA, MMEはそれぞれ新しくつくられるマクロオペレータの条件リスト，削除リスト，追加リスト，主要効果リストである．

$MC = C1 \cup C2 \cap \overline{D1} \cap \overline{A1}$

$MD = D1 \cup D2 \cap A1$

$MA = A2 \cup A1 \cap \overline{D2}$

$MME = ME2 \cup ME1 \cap \overline{D2} \cap \overline{C2}$

この手順をマクロオペレータの候補であるオペレータシーケンスに再帰的に適用することにより基本オペレータが合成されマクロオペレータが生成される．例えばFig. 6からは

macro−name (gotod (A, B), gothrudr (A, C, B))

MC : [type (A,door), status (A,open),
type (C,room), inroom (robot,B),
connects (A,B,C)]

MD : [nextto (robot, __), inroom (robot, B)]

MA : [inroom (robot, C)]

MME : [inroom (robot, C)]

が得られる．

5, 実験方法

今まで述べてきたマクロオペレータの選択的学習がロボットの行動計画において有効なことを示すためには，どのような実験を行なえばよいのだろうか．ここでは，客観的に妥当なものと考えられるMinton [3], Fikes [6]の実験方法を用いることにする．それは以下のようなものである．

まず，実験に用いるシステムはつぎの3つである．

(a) 学習無し問題解決システム：STRIPS

(b) 選択的でないマクロオペレータ学習システム：
M−STRIPS

(c) 選択的マクロオペレータ学習システム：PiL2

ここで (a), (b) のシステムについて若干の説明をしておくと，まず (a) はPiL2の問題解決モジュールであるSTRIPSだけをそのまま用いることにより実現できる．また (b) はPiL2で解法シーケンスの順序関係を崩さない全ての組合せをすべて4. 3の手法でマクロオペレータとすることにより容易に実現できる．さらにここで重要なことは，この3つのシステムの問題解決モジュール：STRIPSは全く同じもので問題解決パフォーマンスの際における違いは獲得されるマクロオペレータの質と量だけであるということである．よって，この3つのシステムに同じ問題を与えていき，その問題解決のパフォーマンスの効率を比較することにより，完全因果性を用いた選択的マクロオペレータ学習の有効性が検証されることになる．

つぎに問題となるのは，問題解決のパフォーマンスを何をもって評価するかということとどのような基本オペレータ及び問題をシステムに与えるかということであるが，ここでもFikes [6], Minton [3] などで使われている一般的と考えられる以下のものを使う．

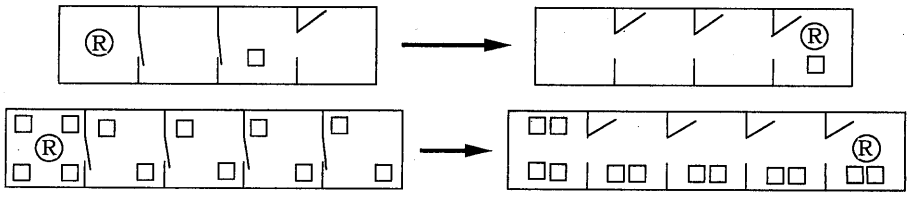


Fig.8 与えた問題の例

1) 評価された枝の数（Branches evaluated）・・・解決に至るまでに評価された枝の合計．展開されてないものを含む．当然，マクロオペレータが多いほどこの数は大きい．

2) 展開されたノード（Nodes expanded）・・・解決に至るまでに展開されたノードの合計．

3) 解決ステップ数（Solution length）・・・解決経路のステップ数．

4) マクロオペレータの数・・・その問題を解いているときのマクロオペレータの数．

5) CPUタイム・・・解決に至るまでのCPUタイム（秒）

次にどのような問題と基本オペレータで実験を行なったかであるが，基本オペレータはSTRIPSの論文［6］に載っている6つのオペレータをそのまま抜粋して用いた．これはPiL2にとって都合のよい基本オペレータで実験したのではないことを強調するためである．また与えた問題は基本オペレータを用いて解決可能な範囲でやさしいもの（解決ステップ数が少ない）から徐々に難しいものへという順序で50問を与えた．Fig. 8にその抜粋を示す．なおこの問題の最高ステップ数は28である．

6, 実験結果とその評価

Table1に紙面の制約上50問中の5問の結果を抜粋した実験結果を示す．P20以降はM−STRIPSの結果がなく，STRIPSとPiL2の結果だけであるが，これは残念なことにP13においてM−STRIPSがマクロオペレータを生成するところでスタックがオーバーフローしてしまいそれ以上実験を続けられなかったからである．なお，P13におけるM−STRIPSのマクロオペレータ数は202個，P13のオペレータシーケンスからのマクロの候補は502個であった．また，M−STRIPSはこの202個のマクロを一通り探索するのにCPUタイムで約200（sec）かかるのでP14以降は必ず200sec以上時間がかかるはずである．以下，実験結果について検討していく．

＜P10＞ すでにこの時点でM−STRIPSはSTRIPSよりも効率が低下している．評価された枝はSTRIPSよりも少ないのだが，一般にマクロオペレータの条件リストは基本オペレータよりも長く，従って同じ個数のオペレータを探索するとマクロの方が時間がかかる［3］ので，その影響が出ていると考えられる．またすでにこのときM−STRIPSは70ものマクロを持ってしまっている．対照的にPiL2は評価された枝もマクロも非常に少なく当然効率も最も良い．

＜P20〜50＞ P20以降はPiL2とSTRIPSとの比較になる．まず最も特徴的なことはPiL2のマクロオペレータがP50のときでさえたった5個しかないことである．これをもし選択的に学習していなければ少なくとも1000個以上にはなると考えられるので完全因果性により著しくマクロオペレータの生成が抑えられていることがわかる．またこの少数のマクロオペレータが役に立つものであることの証明としてPiL2の解決ステップ数がSTRIPSの半分以下に減少しており，当然それにともなって評価された枝数，展開されたノード数そして全体的な評価であるCPUタイムなどがすべて減少している．またTable2にP13までとP50までのCPUタイムの平均値を示す．これからわかるようにM−STRIPSはすでにP13までSTRIPSよりも圧倒的に効率が低下している．PiL2はP50までの平均でSTRIPSの約7．8倍の効率を示しており少数のマクロオペレータを抽出することによりかなりの問題の範囲で学習無しシステムよりも効率的なパフォーマンスが実現できることがわかる．

	P13	P50
PiL2	7	28
STRIPS	16	218
M−STRIPS	104	?

Table2 CPUタイムの平均（sec）

problemID	P10	P20	P30	P40	P50
Branches evaluated	180 : 140 : 9	72 : 10	315 : 20	486 : 20	756 : 60
Nodes expanded	31 : 9 : 5	16 : 5	51 : 7	77 : 9	155 : 54
Solution length	8 : 2 : 3	5 : 2	10 : 4	12 : 4	28 : 12
Macro−operators	0 : 70 : 3	0 : 5	0 : 5	0 : 5	0 : 5
CPU time (sec)	47 : 104 : 10	20 : 12	114 : 19	244 : 27	1819 : 197

STRIPS : M−STRIPS : PiL2

Table1 実験結果

ただし，この50問中でPiL2がSTRIPSよりも遅い場合もあった．それはいままで生成してきたマクロがまったく使えない新しいタイプの問題を解くばあいで，そのときはPiL2は最初マクロだけで解こうとして解けずに基本オペレータで解決するわけで，はなから基本オペレータで解いていくSTRIPSの方が速く解けるわけである．

以上のようにPiL2は少数のマクロオペレータだけで学習無し問題解決システムよりも効率の良い問題解決を行うことが可能であることが実験的に証明された．このことはPiL2が完全因果性に基づいて少数の役に立つマクロオペレータのみを選択的に学習可能であり，またその学習方法が問題解決の効率化に有効であることを示している．

7, 関連研究

1) STRIPSにおけるMACROPS

学習有りSTRIPS [6] において学習されたMACROPSというマクロオペレータは三角表という表現形式で蓄えられる．それに対してPiL2ではマクロオペレータも基本オペレータと同様の構造をしているのでM−STRIPSイコール学習有りSTRIPSとはならずMACROPSとの直接的な比較はできない．しかし，Mintonの指摘 [3] のとおりMACROPSは重複するもの以外は全く選択なしに過去の解決過程の履歴をすべて蓄積していくのでかなり少数の問題（Minton [3] によると最高16ステップの問題で20問程度）ですでにマクロの数が増加してしまい，学習無しシステムよりもパフォーマンスが低下する現象が起こりはじめる．これに対してPiL2は最高ステップ数：28の問題50問でも学習無しSTRIPSよりもほとんどの場合において効率がよいことから，MACROPSを学習するSTRIPSよりも優れていると考えられる．

2) Mintonのマクロオペレータ [3]

Mintonの提案したマクロオペレータには次の二つがある．1) S−MACRO：過去の解決履歴を残しておき，その共通部分シーケンスをマクロオペレータとする．また，このマクロオペレータの数は予め上限値が設定されており，それを越えると使用頻度の少ないものから削られる．2) T−MACRO：サブゴールに相互作用がある場合に有効なマクロ．探索木が枝分かれする点の近傍で数ステップのマクロが生成される．

完全因果性で得られるマクロオペレータはS−MACROに対応する．ただし，かなりの量の過去の履歴を比較して共通部分を抽出するという相当のコストがかかるであろうMintonの手法に比べ，完全因果性の手法は過去の履歴を必要としないので1問でも問題を解けば比較的容易にマクロを生成可能できる点がMintonの手法よりも優れていると思われる．

3) Operationality Criterion [2] としての完全因果性

マクロオペレータの生成も効率面からみると操作可能化（operationalization）と考えられる．EBGによる一般化のところでも述べたように，オペレータの条件部はすべて問題解決モジュールにとって従来の意味で操作可能なものなので，得られた説明木は全てのオペレータの区切れが操作可能であり，よってどの区切れで切って一般化してもかまわない．しかし，実際は全ての区切れでマクロオペレータを生成していたのではその中に役に立たないものが含まれてくる．このことはどのように考えればいいのだろうか．その答えは最近のKellerの操作可能性（operationality）についての研究 [7] が与えてくれる．Kellerは従来の操作可能性を以下のようにより厳密に定義した．（前提となる要素は省略してある）

[従来の操作可能性の定義]：概念記述がある概念の外延であるインスタンスを認識するために効率的に使用できればその記述は操作可能である．

[Kellerの操作可能性の定義]：概念記述が以下の2つの条件を満たすとき，その記述は操作可能である．

・使用可能性（usability）：その概念記述がパフォーマンスシステムにとって使用できる．

・有効性（utility）：その概念記述がパフォーマンスシステムによって使われたとき，明記された目的についてそのシステムのパフォーマンスが改善される．

この二つの定義は重なり合う部分も多いが，従来の定義は「パフォーマンスシステムにとって使用可能な概念記述は必ずパフォーマンスにおいても有効な概念記述である」という暗黙の前提のもとに使われるのに対し，Kellerの定義は使用可能性と有効性を独立したものとして考えそれぞれを評価するところと，有効性を「目的」に依存するものとして捉えているところが決定的に異なる．

ここでは前者の相違点が意味を持つ．つまり，考えられる全てのマクロオペレータが従来の定義により操作可能であるということはあくまでマクロの条件リストが使用可能であるということのみからいえることで，決して有効性を保証するものではないからである．Kellerの定義に基づきマクロオペレータを考えた場合，多くの使用可能なマクロオペレータのうち実際にパフォーマンスを改善するものだけが操作可能になるのである．KellerのMETA−LEXシステムではこの有効性の検証を実際に

ベンチマークテストを解かせて行なっているがそれでは非常に効率が悪い．それに対して完全因果性は実際に問題解決することなしに有効なマクロオペレータの記述を選択できるヒューリスティックであり，非常に効率的な評価が可能な操作可能性の基準であるといえる．

8, 問題点

1）サブゴールの相互作用

STRIPSは縦型探索を行なうので複数個のサブゴールがある場合順序づけをして最も優先度の高いものから展開していく．この際サブゴール間に相互作用があるとき，つまりサブゴール間に不可能な順序づけがあるとき，システムが誤って不可能な順序づけを行なってしまいその結果デッドエンドに陥り，バックトラックで正しい順序づけに戻すという無駄なことを往々にしてやる．完全因果性によって得られるマクロオペレータではこのような問題に対処できないと考えられる．この問題についてはMinton［3］のT−MACROが解決案としてある．

2）マクロオペレータの増加

PiL2はマクロの選択的抽出により6つのオペレータと一定の難易度の問題において学習無しシステムよりも効率がよいことが検証された．しかし，基本オペレータ及び難しい問題が非常に多く与えられた場合でも効率の良さを維持できるのだろうか．この問題は今後の課題であるが筆者の希望的な考えでは完全因果性で得られるマクロオペレータの個数の上限は決まっており，いくら多くの基本オペレータと多くの難しい問題を与えようともマクロオペレータは一定量以上増えないのではないかと思われる．この現象はPiLでの1次方程式の解法学習［1］では確認されている．もしこの仮定が正しいとするとあとはこの定常状態で学習無しシステムよりも効率がよいかどうかであるが，それは今のところなんとも言えない．

3）ゴミのようなマクロがもし生成された場合への対処

完全因果性はあくまでもヒューリスティックなので常に役に立つマクロオペレータだけが得られる保証はない．もし全く無益なマクロオペレータが得られた場合への対処が今のところまったくない．今後の課題である．

9, まとめと今後

完全因果性というヒューリスティックに基づき少数の役に立つマクロオペレータのみを選択的に学習する方法を示し，PiL2を含む3つのシステムを用いた実験によってその有効性を検証した．本報告のマクロオペレータ学習手法は方程式解法，ロボットの移動計画での学習において十分有効であることが証明された．また，問題状態を述語表現で記述しており汎用性・拡張性に富むと考えられる．なお，PiL2はsun3上でK−PROLOG（インタプリタ）を用いてインプリメントされた．

今後はさらに多くの基本オペレータと解決ステップを必要とするより複雑な問題領域におけるPiL2の能力評価を行ないたいと考えている。

＜謝辞＞

毎週土曜日，研究方針及び進行状況について有益な議論をしていただいた大阪大学産業科学研究所の安部憲広助教授，失業中の石川智浩氏及び辻研AIグループの皆さんに感謝します．

＜参考文献＞

［1］山田，安部，辻：問題解決における戦略知識学習システム：PiL −1次方程式・不等式でのケーススタディー，人工知能学会誌，Vol. 3, No. 2 （1988）

［2］Mitchell, T. M., Keller & Kadar−Cabelli : Explanation−Based Generalization : A Unifying View, pp. 47−80, Machine Learnig, 1−1 (1986)

［3］Minton, S. : Selectively Generalizing Plans for Problem−Solving, Proc. of IJCAI−85, pp. 596−599 (1985)

［4］山田，辻，安部：直接解決可能性に基づく一般化：DSBG −操作可能なSOLVABLE概念の定義付け−，人工知能学会誌，Vol. 3, No. 6 （1988）［掲載予定］

［5］Fikes, R. E. and Nilsson, N. J. : STRIPS : A New Approach to the Application of Theorem Proving to Problem Solving, Artif. Intell., Vol. 2, No. 3／4, pp. 189−208 (1971)

［6］Fikes, R. E., Hart, P. E. and Nilsson, N. J. : Learning and Executing Generalized Robot Plans, Artif. Intell., Vol. 3, No. 4, pp. 251−288 (1972)

［7］Keller, R. M. : Defining Operationality for Explanation−Based Learning, pp. 482−487, AAAI−87 (1987)

express my thanks to Dr. Kazuhiro Fuchi, Director of the ICOT Research Center, who provided me with the opportunity of doing this research in the Fifth Generation Computer Systems Project.

References

[Boose 84] Boose, J., Personal Construct Theory and the Transfer of Human Expertise, *Proceedings of the National Conference on Artificial Intelligence, Austin, Texas*, 1984

[Boose 87] Boose, J., Expertise transfer and complex problems: using AQUINAS as a knowledge-acquisition workbench for knowledge-based systems, *Int. J. Man-Machine Studies 26*, pp3-28, 1987

[de Kleer 86] de Kleer, J., An Assumption-based TMS, *Artificial Intelligence 28*, pp127-162, 1986

[Doyle 79] Doyle, J., A Truth Maintenance System, *Artificial Intelligence 12*, pp231-272, 1979

[Fujita 87] Fujita, H. and Furukawa, K., A Self-Applicable Partial Evaluator and Its Usein Incremental Compilation, *New Generation Computing*, Vol.6, No.2, 1988

[Kahn 85] Kahn, G., Nowlan, S., and McDermott, J., Strategies for Knowledge Acquiqsition, *IEEE Transactions on Pattern Analysis and Machine Intelligence 7(5)*, 1985

[Mitchell 85] Mitchell, T., Mahadevan, S., and Steinberg, L., LEAP: A Learning Apprentice for VLSI Design, *Proceedings of the 9th IJCAI, Los Angeles*, 1985

[Mitchell 86] Mitchell, T., Keller, R. and Kedar-Cabelli, S., Explanation-Based Generalization: A Unifying View, *Machine Learning 1*, January 1986

[Taki 85] Taki, H. and Sakaue, Y., A Programming System for Intelligent Robots with Force Feedback, *ROBOTS 9 Conference Proceedings (SME)*, Vol. 1, 1985

[Taki 87] Taki, H., Tsubaki, K., and Iwashita, Y., EXPERT MODEL for Knowledge Acquisition, *Proceedings of IEEE 3rd Annual Expert Systems in Government Conference, Washington, D.C.*, 1987

[Taki 88] Taki, H., Interpretation Based Knowledge Acquisition, *Report of Information Processing Society of Japan, 88-AI-56-5*, 1988 (in Japanese)