論　文　　Technical Paper

完全因果性によるマクロオペレータの選択的学習

Selective Learning of Macro-Operators with Perfect Causality

山田　誠二*　　辻　三郎*
Seiji Yamada　　Saburo Tsuji

*　大阪大学基礎工学部制御工学科
Dept. of Control Eng., Faculty of Engineering Science, Osaka University, Toyonaka 560, Japan.

1988年8月25日　受理

Keywords: macro-operator, machine learning, Explanation-Based Generalization.

Summary

The macro-operator learning means extracting sub-sequences from a worked example and generalizing them for the future problem solving. In general, the candidate of macro-operators extracted from a worked example are so many and they include a lot of useless ones. Therefore, if the learning system generates macro-operators from all the candidates, the amount of macro-operators will get explosively large. Thus, the major problem in the macro-operator learning is how the learning system selects the valid macro-operators out of many candidates. To cope with this problem, we suggest the method of learning macro-operators with Perfect Causality : the heuristics to select only the valid macro-operators. We developed PiL2 system that can acquire useful macro-operators selectively with Perfect Causality and generalize them with EBG (Explanation-Based Generalization) method. We made the experiment in the robot planning by using STRIPS as a problem solver.

1. はじめに

筆者（山田）は，問題解決における戦略知識の学習を目指し，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，個々の基本オペレータに関する戦略知識と複数の基本オペレータを一つに合成したマクロオペレータの二種類を獲得可能である．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可能で効率向上が実現できる．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，ついには学習な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学習することが必須である．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上での各種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た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に適用可能とするために，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ける完全因果性の有効性を検証する[5]．

本論文では，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，マクロオペレータ抽出アルゴリズムとEBGによる一般化手法を説明し，最後に実験方法・結果とその評価および関連研究を示す．

2．PiL2の概要

PiLシステムでは，問題解決モジュールおよ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，PiL2の特徴を以下に示す．

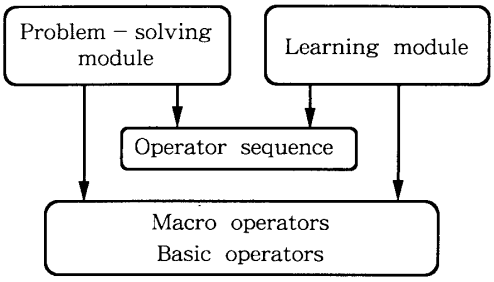


Fig. 1 The structure of PiL 2.

(1) 汎用性を目指し問題状態表現を述語表現にする．
(2) マクロは基本オペレータと同じ構造である．
(3) 拡張された完全因果性を用いる．

以降，各モジュールの働きとマクロオペレータの選択的学習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[6][7]を用いる．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され，その問題状態を変化させるのが基本オペレータである．STRIPSは問題の初期状態，目標状態および基本オペレータを与えられ，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．基本オペレータはFig. 2のようなもので，条件リスト（cond），削除リスト（delete），追加リスト（add），主要効果リスト（main-effect）から構成されている．それぞれの意味は，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．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，これが空リストの場合は，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．主要効果リストは関連したオペレータ（relevant operator）の探索[6]の際に用いられる．

```
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 Basic operators.

なお，STRIPSには二つのバージョン[6][7]があり，一つは学習なしの問題解決システム[6]であり，もう一つは学習可能な問題解決システム[7]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．

重要な点は，PiL2は学習なしの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，まずマクロオペレータだけで探索して，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．また，PiLのSTRIPSはノードの展開のレベルごとに最適なノードを選んでそれだけを展開していく．ノードの順序付けは，①サブゴールの適用により満たされるリテラル数，②サブゴールの適用可能性，③繰り返されるサブゴールなどを評価して行っている．

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すべてのサブシーケンスであるからMC (n) $= \sum_{k=2}^{n}$ 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，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に増大してしまう．よって，学習ありSTRIPS[7]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な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な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．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，数多くあるマクロの候補からどのように役に立つもの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らかであるが，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[1]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考え出したものであり，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}$(1 \leqq i < j \leqq n)$ の任意の要素であるオペレータ OPm（ただし，$m \neq i$）は以下の条件を満たす．

OPm は IS で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．

```
INPUT (IS,OPS)   {IS is initial problem state,
                  OPS = [OP1 ··· OPn]}
Let IOP be OPS's operaotrs applicable 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 DO BEGIN
    IF OPj ∉ IOP
      THEN IF OPj is applicable 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
```

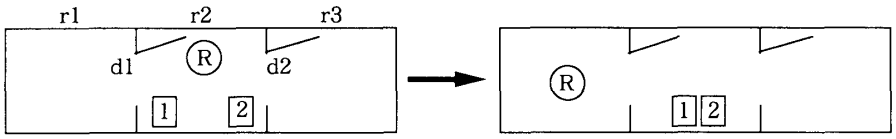
Fig. 3 The algorithm of extracting macro-operators.

この条件は {OPi…OPm−1} 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}$(1 \leqq k \leqq (n-1))$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，そのうち最大のシーケンス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り，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中の各リテラルはユニークなラベルづけ (fn) がされている．図中で f17 と f19 は同じリテラルであるが，それらを追加したオペレータが違うので区別している．

まず最初に IOP= [OP1, OP3] が求まる．次に OP1 が無条件に IS に適用され，その結果が RESULT=



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) {Go to box1 in room2}
↓ cond : [f1,f10,f11] delete : [ ] add : [f17 : nexttc(robot,box1)]
OP2 : pushb(box1,box2,room2) {Push box1 to box2 in room2}
↓ [f1,f2,f8,f11,f12,f17] [f17] [f18 : nexttc(box1,box2),f19 : nextto(robot,box1)]
OP3 : gotod(door1,room2) {Go to door1 in room2}
↓ [f3,f10,f15] [f19] [f20 : nexttc(robot,d1)]
OP4 : gothrudr(door1,room1,room2) {Go through door1 into room1 from room2 }
↓ [f3,f5,f10,f13,f15,f20] [f10,f20] [f21 : inroom(robot,r1)]
GOAL : [f18,f21]

Fig. 4 A given problem and the basic operator sequence.

[f0～f17] となり，MOP= [OP1] とセットされる．次の OP2 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，OP2 を RESULT に適用した結果で RESULT を更新し，MOP= [OP1, OP2] 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≠[OP1] より MOP= [OP1, OP2] 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．MOP= [OP2] とセットされ，OP2 が無条件に IS に適用される．この際，削除リスト中の f17 は IS に存在しないので削除されず，追加リスト中の f18, f19 が追加され RESULT= [f0～f16, f18, f19] となる．次の OP3 は IOP に含まれるので，とばして OP4 を調べる．しかし，OP4 の条件リスト中の f20 は OP3 により追加されるものなので，OP4 は適用できず，結局 MOP= [OP2] のままでこのループは終了する．そして，MOP≠[OPi] の条件が満足されず，このループではマクロは得られない．以下，同様の処理が行われこの例から得られる最終的な出力は，MOPS= [[OP1 : gotob, OP2 : pushb], [OP3 : gotod, OP4 : gothrudr]] となる．この二つのシーケンスはそれぞれ独立に行えるので，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 個のマクロの候補が考えられるが，そのうち二つだけが選択的に抽出されている．また，このアルゴリズムを用いると連続なオペレータシーケンスから，非連続なオペレータシーケンスで構成されるマクロオペレータが得ることも可能である．

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．それは複数のマクロによって一つのマクロが生成されることである．例えば，二つ隣りの部屋へ行くときに macro (gotod, gothroudr), macro (gotod, gothrudr) 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を蓄えていくとどうなるだろう．マクロ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の候補であるオペレータシーケンスは，インスタンスなのでこれを一般

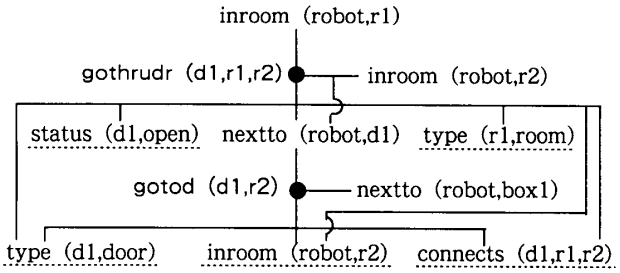


Fig. 5 Explanation tree.

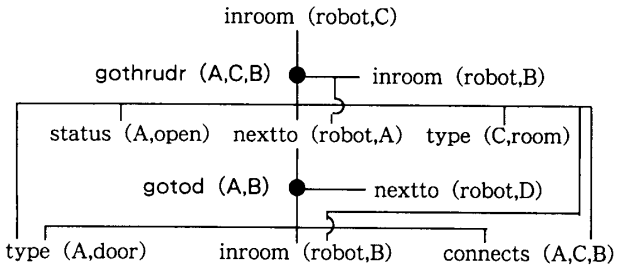


Fig. 6 Generalized explanation tree.

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う．一般に EBG では次の四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4・1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．Fig. 5 では黒丸がオペレータを表し，上，右，下についているノードがそれぞれ追加，削除，条件リストを表す．また，点線のノードが訓練例に，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（詳細は関連研究で言及）と考えられ，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) となる．

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．次にこの一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て一つのマクロを生成する方法を Fig. 7 で説明する．この図は一般化された二つのオペレータのシーケンスで，Cn, Dn, An, MEn はそれぞれ条件リスト，削除

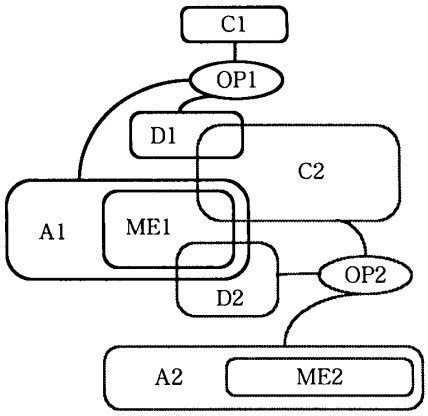


Fig. 7 Generation of macro-operators.

リスト，追加リスト，主要効果リストを表す．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うことにより新しいマクロの各リストが生成される．MC, MD, MA, MME はそれぞれ新しくつくられるマクロの条件リスト，削除リスト，追加リスト，主要効果リストであり，上付きの線は補集合を表す．

$$MC=C1\cup(C2\cap\overline{D1}\cap\overline{A1})$$
$$MD=(D1\cup D2)\cap\overline{A1}$$
$$MA=A2\cup(A1\cap\overline{D2})$$
$$MME=ME2\cup(ME1\cap\overline{D2}\cap\overline{C2})$$

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が生成される．例えば，Fig. 6 からは

macro (gotod (A, B), gothrudr (A, C, B))
MC : [type (A, door), status (A, open),
type (C, room), inroom (robot, B),
connects (A, C, B)]
MD : [nextto(robot,_), inroom(robot, B)]
MA : [inroom (robot, C)]
MME : [inroom (robot, C)]

が得られる．

5. 実 験 方 法

以上のマクロ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[7] 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次の三つである．

(a) 学習な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

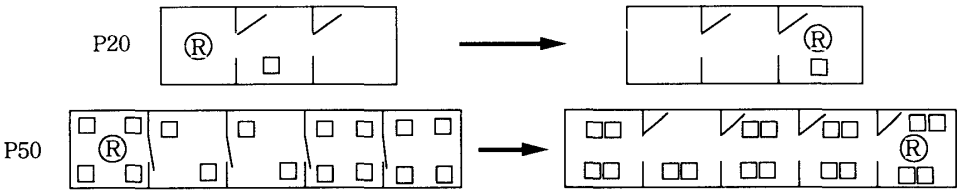


Fig. 8 The examples of training problems.

(c) 選択的マクロオペレータ学習システム：PiL2

(a)は PiL2 の問題解決モジュールである STRIPS をそのまま用いることにより実現できる．また，(b)は PiL2 において全く選択せずにすべての候補からマクロを生成することにより実現できる．ここで重要なことは，この三つのシステムの問題解決モジュールは全く同じもので，その違いは獲得されるマクロの質と量だけということである．

次に問題解決のパフォーマンスを何をもって評価するかであるが，ここでも Fikes[7]，Minton[3] などで使われており一般的と考えられる以下のものを採用する．

(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．

(2) 展開されたノード数 (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うかであるが，基本オペレータは STRIPS の論文[7]に載っている六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，与えた問題は全部で 50 問で，基本オペレータを用いて解決可能なものである．そして，それらをやさしいものから徐々に難しいものへという順序で与えた．Fig. 8 にその抜粋を示す．なお，与えた問題を STRIPS で解いた場合の解決経路の最大ステップ数は 28 である．

6．実験結果とその評価

紙面の制約上，Table 1 に 50 問中の 5 問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 M-STRIPS が P13 を解いて，そのオペレータシーケンスからマクロ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，P1～12 で M-STRIPS が生成したマクロ数は 202 個，P13 を解いたあとでそのシーケンスから生成しようとしたマクロの候補は 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 秒かかるので，P14 以降は必ず 200 秒以上かかることになる．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，一般にマクロの条件リストは基本オペレータよりも長く，したがって，同じ個数のオペレータを探索するとマクロのほうが時間がかかる[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

Table 1 Experimental results.

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

```
M1 : [gotod (D,R1), open (D),
      gothrudr (D,R2,R1)]
M2 : [gotod (D,R1), gothrudr (D,R2,R1)]
M3 : [gotob (A,R1), pushb (A,B,R1)]
M4 : [gotob (A,R1), pushd (A,D,R1),
      pushthrudr (A,D,R2,R1)]
M5 : [pushd (A,D,R1),
      pushthrudr (A,D,R2,R1)]
```

Fig. 9 Macro-operators in PiL2.

Table 2 The averages of CPU times (sec).

	P1～P13	P1～P50
PiL2	7	28
STRIPS	16	218
M－STRIPS	104	?

少なく当然効率は最も良い.

＜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のマクロがP50のときでさえたった5個しかないことである. その5個のマクロ名をFig. 9に示す. これらのうち, M5はM4に包含されるが, M4とM5の差が一つの基本オペレータ：gotodであり, マクロではないので4・2の拘束条件には違反しない. マクロ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. さらに, Table 2にP1～P13, P1～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倍の効率の良さを示しており, 少数の有効なマクロオペレータを抽出することによって, かなりの問題の範囲で学習な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.

以上のようにPiL2は, 少数のマクロオペレータだけで学習な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. 関連研究と問題点

(1) STRIPSにおけるMACROPS

学習ありSTRIPS[7]において, 学習されたMACROPSというマクロは, 三角表という形式で蓄えられる. それに対してPiL2では, マクロも基本オペレータと同様の構造をしているので, M-STRIPSが学習ありSTRIPSと等価にはならず, 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, 学習な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問でも学習なし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. ①S-MACRO：過去の解決履歴を残しておき, 共通する部分シーケンスをマクロとする, ②T-MACRO：固定された評価関数の極大値間のオペレータシーケンスをマクロにする.

S-MACROは相当量の履歴を比較しなければならないし, T-MACROは固定された評価関数に依存している. これに対し, 完全因果性によるマクロ学習は一つのオペレータシーケンスからも学習可能であり, 評価関数からも独立である.

(3) Korfのマクロオペレータ[8]

Korfは,「今まで達成したサブゴールを覆すことなくさらに別のサブゴールを達成する」というマクロが存在する問題領域の条件：operator decomposabilityとそのようなマクロを自動生成する手法を提案した. このマクロの特性は完全因果性とは大きく異なっており, 比較的狭義なものである. Korfの問題点は, problem solverの問題解決戦略が上記のマクロの特性に強く依存していることと, operator decomposabilityが常に成り立つとは限らないことである. これに対し完全因果性も常に成り立つ保証はないが, problem solverの戦略の依存性はない.

(4) 沼尾らのマクロ学習[10]

沼尾らはオペレータシーケンスを説明グラフで表現

し，その部分グラフを合成してマクロオペレータを学習する手法を提案している．本研究と関連する肯定例(正しい解法シーケンス)のみからの学習におけるマクロの選択は，肯定例から外れていくルールの適用シーケンス：Sを肯定例に加えたものを否定例と考え，その最初に外れだしたルールを含む否定例中の任意の部分グラフに一致しない，肯定例中の最小の部分グラフをマクロの候補として抽出し，合成する．この手法には，Sが非常に多い場合や深さが有限でない場合に有効なマクロが生成できるのかという疑問がある．完全因果性は純粋に肯定例のみからマクロを選択するのに対し，沼尾らの方法はあくまでも否定例に大きく依存しているところが本質的に異なる．文献(10)の論理回路の設計例から我々の手法で学習されるマクロを比較例として以下に示す．

＜**沼尾らのマクロ**＞

(and X Y) → (not (or (not X) (not Y)))

(マクロではないが，(not (or X Y)) → (NOR X Y) も得られる)

＜**完全因果性によるマクロ**＞

(or X Y) → (not (NOR X Y),

(and (not (NOR X Y)) (not (NOR V W)))

→ (NOR (NOR X Y) (NOR V W))

(5) 操作可能性の基準[2]としての完全因果性

基本オペレータの条件部は均一レベルの述語で記述されており，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，サブシーケンスから得られた説明木は，すべて操作可能である．よって，マクロ学習において操作可能性基準に対応するものがないことになる．これは，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[9]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．

〔**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**〕：概念記述が以下の二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」という暗黙の前提があるのに対し，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とらえているところが決定的に異なる．

このようなKellerの定義からみた場合，すべてのマクロの候補は使用可能であるというだけでその有効性は保証されていない．多くの使用可能なマクロの候補のうち実際にパフォーマンスを改善するもの，つまり有効なものだけが操作可能になるのである．完全因果性はこの有効性を満たすための基準であるから，完全因果性が操作可能性基準に対応することになる．KellerのMETA−LEX[9]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っているがそれでは非常に効率が悪い．対照的に完全因果性は，実際にテストすることなしに有効なマクロを選択できるヒューリスティックであり，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．

最後に完全因果性によるマクロ学習の問題点として，①マクロ数の上限が保証されない，②無意味なマクロができた場合の対処方法がない，③問題を与える順序への依存などの問題点があり今後の課題である．

8. ま と め

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を含む三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．

その結果，本論文のマクロオペレータ学習の手法は，方程式解法のみならずロボットの行動計画での学習においても，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．

謝　　辞

本研究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，松下電産情報システム研究所の石川智浩氏および辻研FAIグループの皆さんに感謝します．また，最後に本研究の契機を与えてくれたAAAI-88のreviewersに感謝する次第です．

◇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, pp. 206-215 (1988).

(2) Mitchell, T. M., Keller and Kadar-Cabelli : Explanation-Based Generalization : A Unifying View, pp. 47-80, Machine Learning, 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, pp. 783-791 (1988).

(5) 山田, 辻：完全因果性によるマクロオペレータの選択的学習, 情報処理研究会, 88-AI-60 (1988).

(6) 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).

(7) 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).

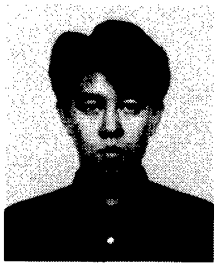
(8) Korf, R. E. : Macro-Operators : A Weak Method for Learning, *Artif. Intell.*, Vol. 26, No. 1, pp. 35-77 (1985).

(9) Keller, R. M. : Defining Operationality for Explanation-Based Learning, AAAI-87, pp. 482-487 (1987).

(10) 沼尾, 志村：説明の部分構造に基づくルール学習法, 電子情報通信学会論文誌, D-II Vol. J72-D-II, No. 2, pp. 263-270 (1989).

〔担当編集委員・査読者：元田　浩〕

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. 同年より基礎工学部制御工学科助手. 工学博士. 人工知能, 特に学習, 説明に基づく一般化およびマクロオペレータに関する研究に従事. 情報処理学会, IEEE各会員.

辻　三郎（正会員）

1953年大阪大学工学部卒業. 1955年同大学大学院修士課程修了. 同年, 電子技術総合研究所入所. 1970年より大阪大学基礎工学部制御工学科教授. 工学博士. 現在, 人工知能, ロボティックス, コンピュータビジョンなどの研究に従事. 情報処理学会, 電子情報通信学会, 計測自動制御学会, IEEE各会員.