人　工　知　能　74－5
（１９９１．　１．　１６）

状態記述系列からのオペレータの生成

下畑　光夫　　山田　誠二　　安部　憲広　　辻　三郎

大阪大学・基礎工学部

本研究では，対象領域に依存しないアルゴリズムで，世界の状態記述系列からオペレータを獲得する方法について報告する．システムに与えられる例は，状態記述系列だけであるため，正の例からの学習に限定されている．そのため，個別の例について，一般化を抑えたオペレータを生成し，後に統合を行っている．また，与える例の表現について考察を行い，対象の表現に規定を設けた．その結果，一つの表現述語が表わすことを明確にでき，対象領域に依存しない一般化が可能となった．あわせて，いくつかの対象領域で実験を行い，その結果について検討を行った．

Generating operators from sequences of state discription

Mitsuo Shimohata　Seiji Yamada　Norihiro Abe　Saburo Tsuji

Faculty of Engineering Science , Osaka University

In this paper, we propose a method to acquire primitive operators with an algorithm independent of any domain. Examples given to system are only sequences of state discriptions as positive examples. To avoid the over_generalization, it generates the most special operators from each examples, and then integrates them. Furthermore, to generalize operators without domain knowledge, we propose criterion for predicates describing the world state. We made experiments in some domains, and discuss on the results.

1　はじめに

「例からの学習」の研究は，いろいろとなされているが，対象領域に関する詳細な知識をあらかじめ持っているものや，学習システムが環境に働きかけて知識を修正することができるもの[1][2]など現実問題としては難しいことを仮定していることが多い．本研究では，対象領域固有の手法をできるだけ排除して，対象領域の変化の観察だけからオペレータを獲得する手法について報告する．また環境の表現についても考察を行い，オペレータの獲得に適した表現方法について提案する．

2章では研究の前提とそのための処理について簡単に述べ，3，4章では環境の表現方法とオペレータの獲得方法について説明する．5章では実際に行った具体例を示し，6章では本研究の問題点について述べる．

2　研究の概要

機械学習は，例の与え方と学習する知識によって多くの種類に分類することができる[3]．本研究は，環境のプリミティブな変化を表現したスナップショットから，環境に適合可能な動作（オペレータ）を，獲得することを目的としている．この前提のもとでは，様々なオペレータを表わす例が，区別なく学習者に与えられる（観察からの学習）．つまり，特定のオペレータについての例がまとめて学習者に与えられるわけではない．また，情報源である環境は学習者の内部状態を考慮しているわけではなく，その例の生成はランダムである．こういった条件のもとで機械学習を行うには多くの対象領域の知識を必要とする．しかし，本研究では対象領域固有の手法に依存せずにオペレータの学習を行う．与えられた一つの例から，正確なオペレータを求めるのは困難であるため，まず個々の例それぞれについて，特殊なオペレータを生成し，その後にオペレータを統合し，より一般的なオペレータに近づけていくという方法をとる．

また，環境の変化とは，オペレータが作用した後の結果であり，その作用した結果しか示されないということは，学習者は正の例しか利用することができないことを意味する．この場合には，個々の特殊な例から一般的なオペレータを導き出すことができるが，過度に一般化されてしまったオペレータに対する対処手段はない．これを防ぐためには，一般化を必要最小限に抑えるか，環境に対する知識を用いるかが考えられる．本研究では，環境に対する知識は可能な限り抑えるという方針により，前者の方法を採用している．ただし，このままでは，オペレータが特殊であるため，獲得されたオペレータ群を統合し，一般化を行う．図1にシステムの構成を示す．

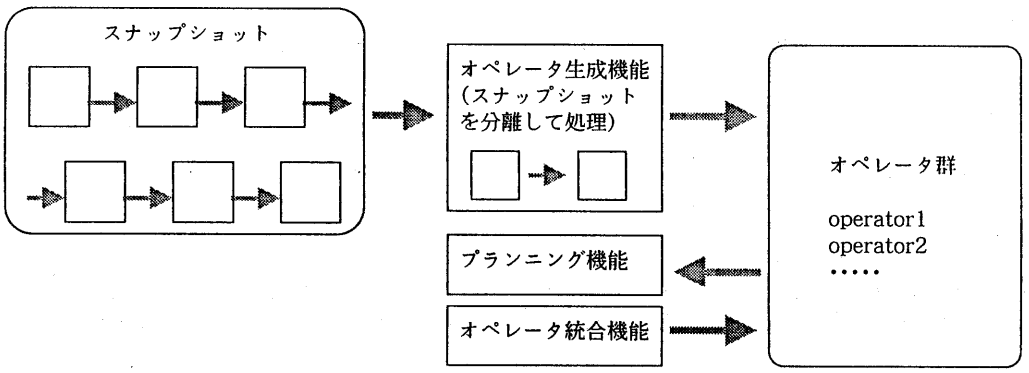


図1　システムの構成

3　述語の表現形式

述語による表現は，STRIPS[4]などのプランナーにおいて多く用いられているが，その表現の形式化の基準には，不明瞭な点が多い．図2に積木の世界で用いられている環境の表現述語を示す[5]．

ontable(X)	*on(A,B)*
holding(X)	*handempty*
clear(X)	

図2　一般的な積木の表現述語

図2から明らかなように，物体の1つの属性に対して2種類の述語が用いられている．例えばハンドの中の状態の表現のためにhandemptyとholding（X）という2つの述語があるが，handemptyをholding(empty)として表現することもできる．同様にontable（X）も，on（X,table）と表現できる．このように，emptyとtableについて別の述語を使う点で，この述語体系は一貫性に欠け，形式化の基準が曖昧である．

対象によって述語を使い分けるということに関しては，プランナー関係の研究ではあまり触れられていない[6]．プランニングとは，オペレータを用いてプランを生成するものであり，表現述語体系が不明瞭であってもオペレータが正確に設定されていれば，そこから生成されるプランは妥当なものが多い．しかし，本研究のように対象領域に依存せずに，一つの例からオペレータを獲得しようとする場合，固定した一般化を行うために，述語表現を規定しなければならない．これにより，対象領域にかかわらず同一のアルゴリズムで，一つの例からの一般化が可能になる．

3.1　述語の設定

本研究においては，一般化を行うことを考えて以下のような設定を設ける．まず環境内において，重要な存在である対象を基本として表現をする．

表現述語は大きく分けて2種類の述語がある．1つは対象の属性について表現した属性述語であり，もう1つは複数の対象間に重要な関係が成り立つときに，それを表現する関係述語である．この設定を用いれば，積木の世界においてもemptyやtableといった対象を特別視することなく，一つの属性について一つの述語が割り当てられる．

3.1.1　属性述語

環境中に重要な意味を持つ対象が存在するとき，各対象について重要な属性がいくつか存在する．一つの属性述語は，ある一つの対象についての，一つの属性の状態について表現している．形式としては，述語の引数の数は2と決まっており，その引数の位置によって意味することが決まっている．述語記号は，対象の属性を表わしており，述語の第一引数，第二引数はそれぞれ対象名とその属性値を表わしている．

例えば，block1という対象が，topという属性（blockの上の状態）において，clear(物体がない）という値をとるならば，top(block1, clear)と表現される．なお，属性の状態は一般的な状態を指しており，具体的対象名が入ることは許されない．したがってinside_hand(h, block1)という表現は，この設定では用いることができない．

3.1.2　関係述語

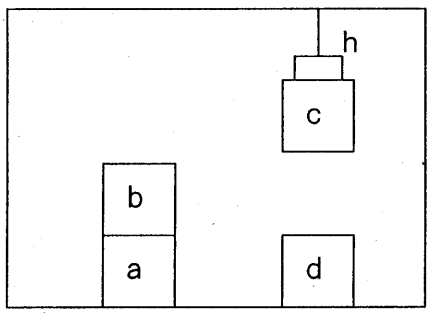
前述の属性述語では，環境中の各対象の状態について表現することができた．属性述語では属性の値に対象名を用いることができないため，対象間の関係については表現できない．そこで，複数の対象間に重要な関係が存在する場合に，関係述語を用いてその関係を表現する．述語記号はすべてrelationとする．引数の数は3以上で，n個の対象がRという関係を持つ場合は引数の数はn＋1であり，最後の引数には関係Rが入る．例えば，block1がblock2の上に積まれているという状態を表現するには，relation(block1, block2, on)と表現する．

3.2　述語の設定手続き

この述語体系は,以下のような手続きで求めていく．また，括弧内に図3の環境における具体例を示す．

①環境内に存在し，環境に対し，重要な意味をもつ対象を選び出す．（積木の世界では，a,b,c,d,h）
②各対象について等しい性質を有するものを対象グループに分類し，述語'type' を用いて対象の属するグループを表現する．（block と hand）
③各対象について，環境における位置付けを決定するために必要な属性を選び出す．（blockの場合は上の状態 (top)，ハンドの場合はハンド内の状態 (inside_hand)）
④各属性について，取りうる値を選び出す．（topは objectかclear,in_handはemptyかobject）
⑤対象間において重要な関係を選び出す．（blockが積み重なっている関係とblockがハンド内にある関係）

この設定にしたがって積木の世界を表現した例が図3である．



type(a,block), type(b,block)
type(c,block), type(d,block)
type(h,hand), top(a,object)
top(b,clear), top(c,object)
top(d,clear), inside_hand(h,object)
relation(b,a,on), relation(h,c,holding)

図3 設定を用いた積木の世界の表現

3.3　述語によるネットワーク表現

このように，述語は対象を単位として表現される．このような設定はちょうど対象をノードとし，関係述語をアークとしたネットワークを表わすことになる．図3の状態は，図4のように表わすことができる．

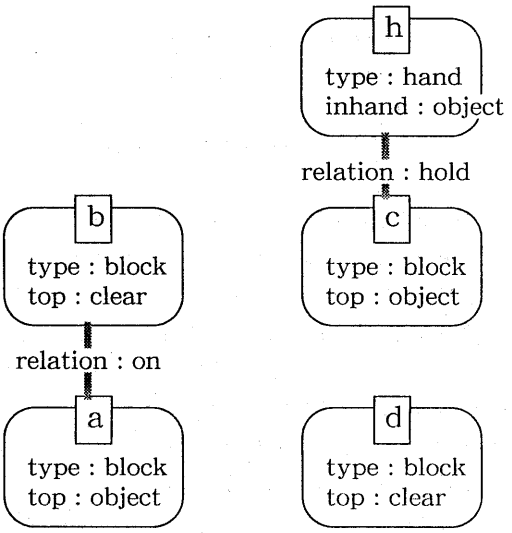
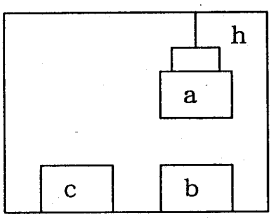


図4 積木の世界のネットワーク表現

4　オペレータの生成

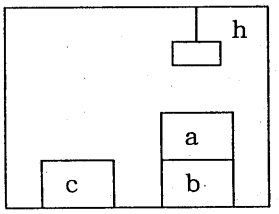
システムには，オペレータの作用前の状態と作用後の状態が与えられる．そして，与えられた作用前後の状態から，そこに成り立つオペレータを生成する．本研究では，オペレータはSTRIPSの形式で生成する．STRIPSのオペレータは3つの重要な部分がある．条件リスト，削除リスト，追加リストである．条件リストはオペレータを適用させる際に，初期状態に必要となる条件を指している．そして，削除リストと追加リストはオペレータを適用させたときに，状態から削除，追加する述語を意味する．この3つのリストが得られればオペレータは生成できる．以下の節で，図5のスナップショットを用いて具体的な生成方法を述べる．

作用前の状態



type(a,block), type(b,block), type(c,block)
type(h,hand), top(a,object), top(b,clear),
top(c,clear), inside_side(h,object),
relation(h,a,hold)

作用後の状態



type(a,block), type(b,block), type(c,block)
type(h,hand), top(a,clear), top(b,object),
top(c,clear), inside_hand(h,empty),
relation(a,b,on)

図5 作用前後の状態

4.1 削除，追加リストの生成

作用前後の状態が与えられた場合に削除，追加リストを求めることは容易である．削除リストとは，作用前の状態にあって作用後の状態にない述語であり，追加リストはその逆である．したがって，作用前後の状態の差を求めることでこの2つのリストを求めることができる．作用前の述語集合をA，作用後の述語集合をBとすると，削除リスト集合Delと追加リスト集合Addには次のような論理式が成り立つ．

$Del = A \cap \overline{B}$

$Add = B \cap \overline{A}$

図6に，図5から求めた削除リストと追加リストを示す．

削除リスト

[top(b,clear),top(a,object),inside_hand(h,object),
relation(h,a,hold)]

追加リスト

[top(b,object),top(a,clear),inside_hand(h,empty),
relation(a,b,on)]

図6 削除，追加リストの生成

4.2 条件リストの生成

条件リストは，そのオペレータを行うために必要となる前提条件である．そして，作用前の状態はこの条件リストを満たしていることが分かっているため，条件リストは，作用前の状態の部分集合となる．したがって，作用前の状態から条件であると思われる述語を取り出せばよいわけだが，その選択は一意には定まらない．さきに述べたように，本研究では与えられる例は正の例だけであり，一般化し過ぎた場合にはこれを修正する手段を持たない．そのことを考慮すれば条件と考えられる述語をすべて取り出した方が安全である．そこでここでは，次のようなヒューリスティックを用いて条件リストを抽出することとする．

「作用前後で変化した対象，及びその対象に直接関係する対象はそのオペレータに関係がある．」（ここで，2つの対象が直接関係するとはその対象間に成り立つ関係述語があるということを意味する．）このことをネットワークで表わすと，図7のようになる．このヒューリスティックに該当する物体を取り出し，次に引数の対象名が，すべてその対象で満たされているような述語を取り出す．図8に，図5から得られた条件リストを示す．条件リストの抽出は本来，領域に関する知識が必要とされるが，前述の述語設定にしたがって関

係述語を設定すれば，この一般的な方法で必要な述語を取り込むことができる．

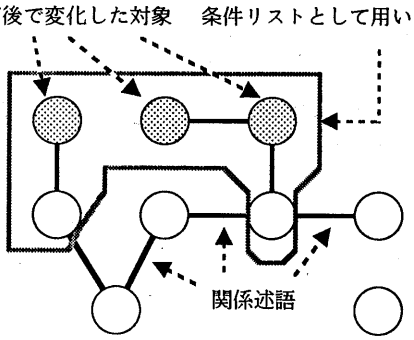


図7　条件リストの抽出範囲

[type(a,block),type(b,block),type(h,hand),
top(a,object),top(b,clear),
inside_hand(h,object),relation(h,a,hold)]

図8　条件リストの生成

4.3　オペレータの一般化

4.1，4.2節で，得られたオペレータの各リストは，例に依存した具体的な値をもったものであり，これを一般化しなければならない．一般化の方法は，機械学習の分野で，様々なものが提案されているが，ここでは属性，関係述語ともに最後の引数以外の引数の値を変数化することにより，一般化を行う．スナップショット中の述語は，特定の対象が特定の属性において，特定の状態にあることを表わしており，このうち対象に関する部分を変数化することで不特定の対象が特定の属性において特定の状態にあることを表わすようになる．図9に図6と図8の各リストを一般化した結果を示す．

operator(stack(_85,_81,_83),

条件リスト

[top(_81,clear), inside_hand(_85,object),
relation(_85,_83,hold), type(_81,block),
type(_83,block), type(_85,hand),
top(_83,object)],

削除リスト

[top(_81,clear),top(_83,object),
inside_hand(_85,object), relation(_85,_83,hold)],

追加リスト

[top(_81,object), top(_83,clear),
inside_hand(_85,empty), relation(_83,_81,on)]).

図9　変数化による一般化

4.4　オペレータの統合

観察からの学習の場合，特定の概念についてまとまった例が与えられないため，与えられた例ごとに概念を生成する必要がある．本研究でも，与えられた例ごとに一つのオペレータを生成している．一方，正の例からの学習であることを考慮し，オペレータの条件リストは非常に特殊なものになっている．その結果として，例を多く入力すると，条件が厳しく一般性に乏しいオペレータが数多く生成されることになる．これではオペレータの利用が難しいため，オペレータの統合を図る．まず，生成されたオペレータ群の中には，本来同一のオペレータが異なる条件リストのため，異なるオペレータとして生成されていることが多い．それらを統合することを考え，次の2通りの方法をとる．

①

削除リストと追加リストが等しく，条件リストが包含関係にある2つのオペレータがある場合には，包含する条件リストをもつオペレータを削除する．

例

オペレータ1の条件リスト：cond1

a(X,on),b(X,off),c(Y,on)

オペレータ2の条件リスト：cond2

a(X,on),b(X,off)

cond1 ⊃ cond2

統合結果：オペレータ1を削除

②

削除リストと追加リストが等しい2つのオペレータの差異が，ある述語とその否定になっている場合，その述語を除いた条件リストを持つオペレータを生成する．これを論理式で表現すると次式のようになる．

オペレータ1の条件リスト：A∩B

オペレータ2の条件リスト：A∩¬B

統合後のオペレータの条件リスト：A

例

オペレータ1の条件リスト

a(X,on),b(X,off),c(Y,on)

オペレータ2の条件リスト

a(X,on),b(X,off),c(Y,off)

述語c(OBJ, STATE)の第二引数は，onかoffしかとらないと分かっている．

⇩

統合結果：オペレータ1，2を削除．新たに条件リストがa(X, on), b(X, off)だけのオペレータを生成する．

どちらの方法も論理的な統合であり，領域に関する知識はいっさい用いていない．もちろん領域知識を用いればより効率的にオペレータの統合ができるが，その方法はこの研究の方針にそぐわないためここでは用いていない．

5　実験結果

以上で述べたような方法に基づき，様々な対象領域について，述語設定を行った．そして，システムにスナップショットを与えて，オペレータを生成させ，その正当性について評価した．次に，生成したオペレータを用いて他のタスクのプランニングを行った．以降の節にその結果を述べる．

5.1　積木の世界

積木の世界について，図2と同程度のレベルの表現について述語の設定を行った．その述語体系は図3と同じである．はじめに，図10で示されるような状態記述系列を与え，4種類のオペレータを生成させた．この4種類のオペレータは，生成した順に，unstack, put_down, pick_up, stackにあたり，それぞれ作用が異なるため統合はできない．図11にその4つのオペレータを示す．

これらのオペレータを用いて，他の問題をSTRIPSで解かせたところ，支障なくプランニングを行うことができた．なお，用いたSTRIPSでは，条件リストの述語を並び変え，競合解消のルールを取り入れている．

このように，積木の世界ではよい結果が得られた．これは積木の世界が関係述語の少ない述語体系で，与えた例もシンプルなものであるため，適切なオペレータを生成することができたと思われる．

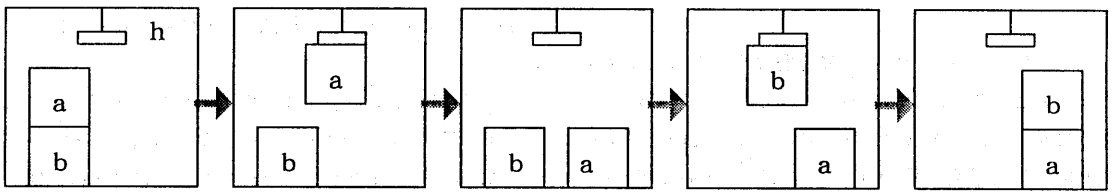


図10　システムに与えたスナップショット

```
operator(unstack(_71, _69, _73),
[relation(_69, _73, on), top(_69, clear), inside_hand(_71, empty), top(_73, object), type(_69, block), type(_73, block), type(_71, hand)],
[top(_69, clear), top(_73, object), inside_hand(_71, empty), relation(_69, _73, on)],
[top(_69, object), top(_73, clear), inside_hand(_71, object), relation(_71, _69, hold)]).

operator(put_down(_73, _75),
[relation(_73, _75, hold), inside_hand(_73, object), type(_75, block), type(_73, hand),
 top(_75, object) ],
[top(_75, object), inside_hand(_73, object), relation(_73, _75, hold)],
[top(_75, clear), inside_hand(_73, empty)]).

operator(pick_up(_79, _77),
[inside_hand(_79, empty), top(_77, clear), type(_77, block), type(_79, hand)],
[top(_77, clear), inside_hand(_79, empty)],
[top(_77, object), inside_hand(_79, object), relation(_79, _77, hold)]).

operator(stack(_85, _81, _83),
[top(_81, clear), inside_hand(_85, object),
 relation(_85, _83, hold), type(_81, block), type(_83, block), type(_85, hand),  top(_83, object)],
[top(_81, clear), top(_83, object), inside_hand(_85, object), relation(_85, _83, hold)],
[top(_81, object), top(_83, clear), inside_hand(_85, empty), relation(_83, _81, on)]).
```

図11　積木の世界で生成されたオペレータ

5.2　ハノイの塔の世界

ハノイの塔はよく知られたパズルである．ハノイの塔ではpegとdiscを対象グループとする．discを移動させるときの条件として，disc間での大小関係が重要であるため，その関係を表わす関係述語が必要となる．用いた述語体系を図12に示す．

type(X,[disc,peg]).
top(X,[clear,fill].
relation(DISC1,DISC2,big).
relation(DISC,PEG,at).
relation(DISC1,DISC2,on)

図12 ハノイの塔の表現述語

discが2，3，4枚の場合の最適解の手順をスナップショットとして与えた．生成されたオペレータ数はそれぞれ3，7，15であった．次にこれらすべてのオペレータについて統合を行わせた．はじめは全部で25あったオペレータが7つまで統合された．統合前のオペレータ数と統合されたオペレータ数の関係を図13に示す．

discの枚数	統合前のopr数	統合後のopr数
2	3	3
3	7	0
4	15	4

図13 統合前後のオペレータの数

この述語体系の場合，全部の動作を表現するために必要な最小オペレータ数は4である．その4つのオペレータはそれぞれ以下の動作を意味する．

Ⅰ.下にdiscがないdiscが，空のpegへ移動
Ⅱ.下にdiscがないdiscが，discのあるpegへ移動
Ⅲ.下にdiscがあるdiscが，空のpegへ移動
Ⅳ.下にdiscがあるdiscが，discのあるpegへ移動

この実験では，Ⅰ，Ⅱ，Ⅲに相当するオペレータはそれぞれ1つずつ生成したが，Ⅳに相当するオペレータについては4つのオペレータが統合されずに，残ってしまった．

この本来一つのはずのオペレータは，discが5枚，6枚の場合の最適解のスナップショットをあたえても，

一つに統合することはできない．このオペレータを表わした，最も単純な例を与えないと生成することができない．しかし，その例はdiscが3枚の場合の冗長な動作にあたるため，最適ステップのみを与えている場合は獲得できない．

5.3 移動ロボットの世界

STRIPSの論文 [4] で用いられていた移動ロボットの世界を，本研究の述語設定に合わせて実験を行った．元々の表現述語が，本研究の述語設定に近い形をしていたため，述語体系の決定はさほど労を要しなかった．図14に述語体系を示す．

type(X,[room,door,box.robot]).
door_condition(DOOR,[close,open]).
robot_condition(ROBOT,[carry,empty]).
relation(DOOR,ROOM1,ROOM2,connect).
relation(X,Y,near).
relation(X,Y,carry).

図14 移動ロボットの表現述語

はじめに，多くの対象があるシーンでのスナップショットを与えたところ，生成されたオペレータの条件リストには，必要な述語が入ってはいるが，不必要な述語も多く含まれており，オペレータの一般性が乏しくなっている．必然的に，他の問題をプランニングさせてもオペレータが適用できない．オペレータの統合法の②を適用させて，より一般化するためには，一般化に適した例が必要となる．ランダムな例を与えただけでオペレータを収束させるのは難しい．

単純な例を与えると，統合法の①が適用され，一般的なオペレータが得られる．一般的なオペレータが得られると他の問題を解くことが可能になる．

5.4 実験の検討

実験した限りにおいては，条件リストに必要な述語が入っていないということはなかったが，その代わりとして不必要な述語が多く含まれてしまった．不必要な述語は統合機能によって排除できると考えていたが，実験を通してみるとさほど有効でないことがわかった．特に，条件リストが交差関係にあるオペレータの統合は，一般化に適切な例を多く教示しないと用いられない．実際，この実験ではこの統合法はほとんど使われなかった．

6 検討

以上のように述べてきた表現述語の設定と,オペレータの生成について検討する．

6.1 高度な帰納学習

本研究で行っている一般化とは，特定の物体が特定の状態においてある動作が行えるときに，この特定の物体にあたる部分を変数化してオペレータとしているに過ぎない．例えば，1加算するというオペレータを考えた場合，1が入力されれば2を出力し，2ならば3というオペレータは獲得できるが，さらに，これらのオペレータを一般化した，XならばX＋1というオペレータは獲得できない．こういった，複数のオペレータからの帰納学習による高度な一般化は今後の課題である．

6.2 スナップショットに要求される条件

学習システムに与えるスナップショットには，プリミティブでなくてはならないということと単一エージェントの動作しか含まないという条件が必要である．プリミティブなスナップショットが与えられなかった場合，マクロなオペレータを生成する．また，複数のエージェントが含まれるスナップショットが与えられた場合，各エージェントの複合作用を表わすオペレータが生成される．

対象領域の知識を持たない本システムではこの問題には対処できないが，領域固有の知識を用いれば，ある程度対処できると思われる．

6.3 条件リストに用いるヒューリスティック

今回の実験では，条件リストの抽出には4.2で述べたヒューリスティックを用いた．しかし，より一般的なヒューリスティックとして「作用前後で変化した物体は，そのオペレータに関係がある」ということも考えられる．このヒューリスティックを用いると，ハノイの塔では，本実験と比較して，一般化と統合が進み，ほとんど最適に近い状態のオペレータ群が得られた．しかし，移動ロボットの世界でこのヒューリスティックを用いると，条件リストに必要な述語が抽出できないオペレータがある．

理論的には，特殊なオペレータは例がたくさん与えられれば，統合されて一般的になるが，現実問題として統合は難しい．この2種類のヒューリスティックは，対象領域に応じて使い分けるほうが良いと思われる．

7 まとめ

状態変化系列から，対象領域固有の知識に依存せずに，オペレータを獲得する方法について述べた．また，述語表現に規定を設け，一つの述語が表現することを明確にした．

参考文献

[1]Wei-Min Shen and Herbert A. Simon :Rule Creation and Rule Learning through Environmental Exploration , *IJCAI89*, pp675-680

[2]Carl M. Kadie: "Diffy-S:Learning Robot Operator Schemata from Examples" , *Machine Learning*, pp430-436

[3]R. S. Michalski他編 :知識獲得入門, 共立出版

[4]Richard E. Fikes and Nils J. Nilson "STRIPS:A New Approach to the Application of Theorem proving to Problem Solving", Artificial Intelligence, Vol2, No. 3, 1971, pp189-208

[5]辻井潤一 ：知識の表現と利用，昭晃堂

[6] 安部憲広，「プラニング」，人工知能学会誌，Vol5, No.6,pp737 - 747,(1990)