〔原著論文〕

知能ロボットにおける問題解決と学習の統合

三浦 純* 下山 勲** 三浦宏文**

Integration of Problem Solving and Learning in Intelligent Robots

Jun MIURA Isao SHIMOYAMA Hirofumi MIURA

In the real world, there are many kinds of uncertainties in motions of robots. Moreover, robots cannot always get sufficient knowledge about tasks in advance. Therefore, intelligent robots must posses both problem solving ability to decide on proper motions with insufficient knowledge and learning ability to acquire knowledge from experiences. Effective integration of the two abilities is also important.

In this paper, we describe an experimental system named ARPEX-L (Advanced Robot Planning and Execution System with Learning Ability). We applied ARPEX-L to two kinds of robot tasks, pick and place task and pushing block task. The former is an simple example of automatic robot programming. The latter is an attempt to make an actual robot learn by a symbolic method. Current defects of the system and future works are also described.

Key Words : Intelligent Robot, Problem Solving, Learning, Automatic Robot Programming

1. はじめに

知能ロボットが実世界において自律的に行動するためには，計画を生成し実行するための問題解決能力と，経験から有用な知識を獲得するための学習能力の両方が必要である．人工知能の分野では，問題解決，学習について従来から数多くの研究が行われてきている．それらのうち問題解決における学習を扱ったものも多い[1)~5)]．しかしこれらの研究においては問題解決に用いられるオペレータに関する情報が完全であることを仮定している．例えば，STRIPS[1)] 的システムでは，各オペレータの前提条件，追加リスト，削除リストという情報があらかじめ与えられている．したがって，オペレータの実行による状況の変化を完全に予測することができ，問題解決や学習についても計算機内の完全に閉じた世界の中での問題として論ずることができる．しかし，実世界で行動するロボットの場合には，動作に不確かさが存在するなどの理由でオペレータに関する情報が不完全である．このような場合には次のことを考慮しなければならない．まず，オペレータの実行によって生じる状況の変化（これをオペレータの効果と呼ぶ）が未知であれば，それを学習し利用する手段が必要である．また，オペレータの効果は，そのオペレータが実行される状況に依存し，すべての状況に対する効果を与えておくことはできないため，実際にロボットを動作させたときの環境の変化は不確かさを伴う．したがって，そのような経験から獲得された知識は不確かさを伴い，不確かさを表現する手段，不確かな知識から適切な行動を選ぶ手段が必要となる．

本論文では，上述の問題を扱うことのできる問題解決と学習の統合システム ARPEX-L (Automatic Robot Planning and Execution System with Learning Ability) の構築について述べる．ARPEX-L は実世界の問題を記号処理の枠組みによって扱おうとするものであり，その構築にあたっては人工知能の分野における従来の研究成果を十分に利用する．ARPEX-L が単なる問題解決システム，学習システムではなくそれらが統合されたシステムであることは重要である．すなわち，

原稿受付 1989 年 8 月 28 日
* 大阪大学工学部電子制御機械工学科
** 東京大学工学部機械工学科

ARPEX-Lは初期知識を与えられると人手を介さずに自ら行動して経験を蓄積し，そこから学習を行う．このようなシステムの研究は将来のロボット・プログラミングの自動化の基礎となる．

以下，第2節でシステム設計の基本方針について述べる．第3節から第6節でシステム各部の説明を，積木の積みおろし（pick and place）作業を例に取って行う．具体的な作業としては **Fig. 1** に示す3つを想定する．これらの作業では積木の積み方を **Fig. 2** に示す3通りに限定し，またロボットは積木を同時に1つしか持てないとする．第7節，第8節ではARPEX-Lをそれぞれ，積木の積みおろし作業，積木の押し作業へ適用した結果について述べる．前者は自動ロボット・プログラミングの簡単な例であり，後者は記号処理の枠組みで実際のロボットに学習を行わせるという最初の試みである．さらに，第9節で考察を行い，第10節で本論文をまとめる．

2. システム設計の基本方針

2.1 知識の表現，利用，学習

一般に，知能ロボットの用いる知識は，それが成り立つ状況，あるいは適用できる状況が限られている．例えば，**Fig. 3** に示すようにオペレータの効果は状況によって異なり，各状況における効果を知っている必要がある．また，**Fig. 4** のようなある特定の状況に対処するための知識は，その状況においてのみ有効である．したがって，知識は一般に適用可能な状況の記述を含めて表現し，現在の状況を参照しながら利用しなければならない．もちろん，一部の知識，例えば積木を下から積んでゆくという知識は状況に依らず成り立ち，常識としてあらかじめ与えておくこともできる．しかし，状況に左右される知識の中には実際に動作を行うことによってのみ獲得可能なものがあるので，各知識を学習するための手段を与えておく必要がある．

2.2 問題解決の基本戦略

知識が完全であれば，ロボットの動作による状況の変化が完全にシミュレートできるので，計算機内でのプランニングにより動作決定は事前に行える．しかし，上に

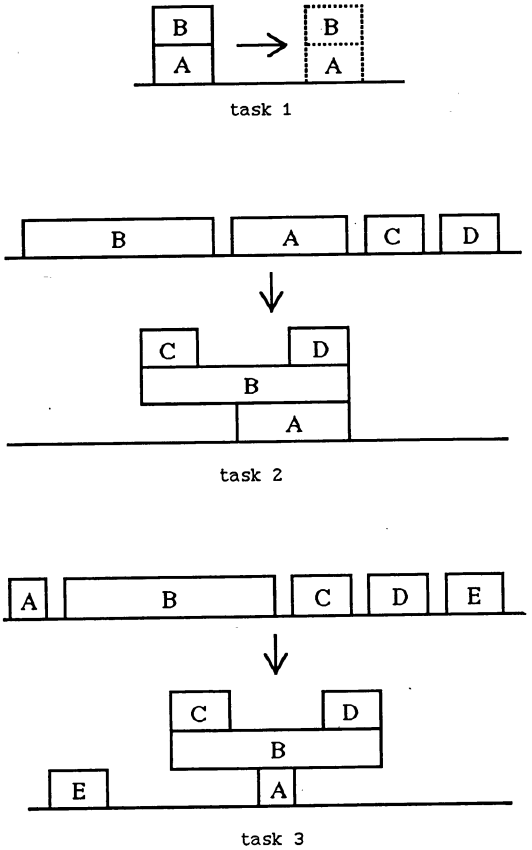

Fig. 1 Three pick and place tasks

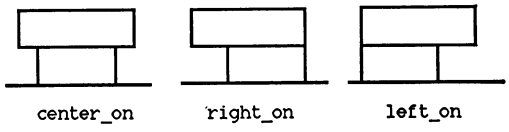

Fig. 2 Three ways to stack blocks

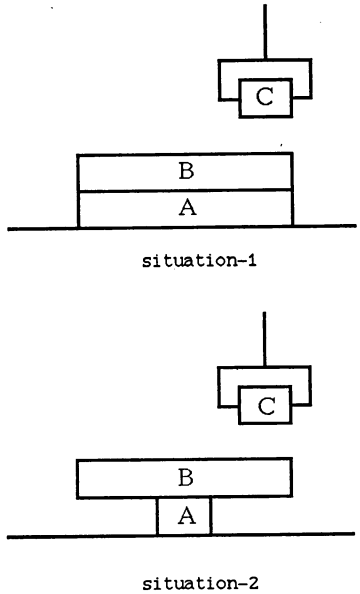

Fig. 3 Effects of an operator and situations

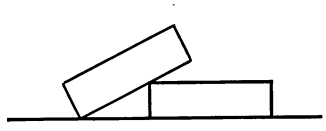

Fig. 4 An erroneous situation

ACT-1	
operator	*mo-1*
situation	*relation(#A, #Table, on_table(#CurrPosA))* *relation(#B, #A, center_on)*
effects	*relation(#A, #Table, on_table(#PosA))* *relation(#B, #A, center_on)*

Fig. 5 An example of extra-ACT

SR-1	
type	*positive*
situation	*relation(Block1, #Table, on_table(Pos))* *relation(Block2, Block1, touching(Trans))*
operator	*carry_on_table(Block2, #PosOnTable)*
constraints	*30 <= Block1 · width <= 40* *30 <= Block2 · width <= 40*
certainty factor	*0.795*

Fig. 6 An example of operator selection rule

も述べたように，知識の欠落，あるいは不確かさのために，動作決定のための知識が完全でない場合には，その場その場の状況を参照しなければ動作決定は行えない．したがって，ARPEX-Lでは一動作ごとに次の動作を決定してゆく．これは知識が不十分な場合には，各々の状況に応じたオペレータを選んでゆけば結果的に効率がよくなるであろうという考えに基づいている．しかし，知識の中には，“積木は下から積んでゆく”というような確かなもの，状況に依らず成り立つものもある．したがって，より効率的な問題解決を行うために，作業開始前に状況に依らず成り立つ知識を用いて大まかな計画を立て，作業遂行中に残りの知識を用いて最終判断を行う，という戦略を採用する．例えば，task 1 の場合には，作業開始前にA→Bの順に積むという方針通を立てる．作業遂行中には状況を見ながら，方針通りの動作を行える場合にはそれを行い，エラーなどが生じて方針りゆかない場合や，作業開始前に方針が立っていなかったところでは，状況に応じて適切な動作を選ぶ．この2段階の動作決定は，いわゆるリアクティブ・プランニング[6),7)]と同様の戦略である．

3. 知識の表現

3.1 状況の記述

状況は述語 relation (Object 1, Object 2, Relation) によって記述する．引数 Relation の値は対象とする問題によって異なる．積木の積みおろし作業では Fig. 2 に示した3つの関係 (center_on, right_on, left_on) および on_table(Pos) (物体がテーブル上の位置 Pos にある)，touching (Trans) (物体同士が変換行列 Trans で接触している)，grasped (ロボットが物体をつかんでいる) の6種を用いた．

3.2 オペレータと ACT

オペレータはその述語記述，前提条件，効果という情報を持つ．さらに，プリミティブオペレータの場合はロボットの動作手順，マクロオペレータの場合には展開されるオペレータ系列に関する情報を持つ．

オペレータの効果は ACT と呼ばれ，そのオペレータの実行によって達成される状況を記述する．ACT は状況に依らない部分とそうでない部分とに分けて表現し，前者を basic-ACT (b-ACT)，後者を extra-ACT (e-ACT) と呼ぶ．b-ACT はオペレータが最低限達成しなければならないと見なした状況を記述し，各オペレータの属性としてあらかじめ与えておく．一方，e-ACT はオペレータ，オペレータが実行される状況，達成される状況の記述という3つ組で表され，あらかじめ与えられるか，学習するかのいずれかである．**Fig. 5** に e-ACT の例を示す．これは task 1 の初期状態において，あるマクロオペレータ mo-1 を実行したところ目標が達成された場合に得られたものである．以下，単に ACT という場合には2種の ACT を合わせたものであるとする．

積木の積みおろし作業では，**Table 1** に示す9個のオペレータを与えておいた．表の左側がオペレータ名，右側が ACT である．pick_up と put_on_xxx はプリミティブオペレータであり，carry_on_xxx はそれらを組み合わせたマクロオペレータである．また，ACT はすべて b-ACT である．

3.3 オペレータ選択規則

ある特定の状況において適切なオペレータを選ぶため

Table 1 Initial operators in pick and place tasks

Primitive Operators	
pick_up(Obj, Pos)	*relation(Obj, #Robot, grasped)*
put_on_table(Obj, Pos)	*relation(Obj, #Table, on_table(Pos))*
put_on_center(Upper, Lower)	*relation(Upper, Lower, center_on)*
put_on_right(Upper, Lower)	*relation(Upper, Lower, right_on)*
put_on_left(Upper, Lower)	*relation(Upper, Lower, left_on)*
Macro Operators	
carry_on_table(Obj, Pos)	*relation(Obj, #Table, on_table(Pos))*
carry_on_center(Upper, Lower)	*relation(Upper, Lower, center_on)*
carry_on_right(Upper, Lower)	*relation(Upper, Lower, right_on)*
carry_on_left(Upper, Lower)	*relation(Upper, Lower, left_on)*

の知識をオペレータ選択規則(operator selection rule, 以下 SR と略す) と呼ぶ. SR には正負の2種があり, 前者は“ある状況ではあるオペレータを選んだ方がよい”という知識を, 後者は“ある状況ではあるオペレータを選ばない方がよい”という知識をそれぞれ表す. このような形式の知識はロボットが経験から容易に獲得可能である. SR は適用可能な状況の記述, オペレータの記述, 記述中に変数があった場合の拘束条件, 確からしさを表すための確信度によって表現される.

SRの例を **Fig. 6** に示す. この SR は Fig. 4 に示す状況では, 上の積み木をテーブルの空いた場所に置いた方がよい, という知識を表している. 変数の拘束条件は個々の経験を帰納的一般化することにより設定される. 対象物の属性は Obj^attr という形式で表すが, どの属性を拘束条件として用いるかは問題によってあらかじめ決めておく.

3.4 計画生成規則

達成すべき副目標間の順序づけを行うための知識を計画生成規則 (planning rule, 以下 PR と略す) と呼ぶ. PR による順序づけは副目標間の干渉の処理であり, 実行するオペレータの選択の際にこの順序づけの情報を考慮すれば, 現在の状況という局所的な視点のみからオペレータを選択する場合に陥りやすい無限ループを避けることができる. PR にも正負の2種があり, 前者は“ある状況ではあるオペレータ Op1 を別のオペレータ Op2 より先に実行した方がよい”という知識を, 後者は“ある状況では Op1 を Op2 より先に実行しない方がよい”という知識をそれぞれ表す. 後者は,“Op2 を Op1 より先に実行した方がよい”という知識を表すものではないことに注意されたい. PR は適用可能な状況の記述, 2つのオペレータの記述, 変数の拘束条件, 確信度によって表現される.

PR-1	
type	*positive*
Op1	*carry_on_center(Object, _)*
Op2	*carry_on_center(_, Object)*
situation	*nil*
certainty factor	*1.000*

Fig. 7 An example of planning rule

PR-2	
type	*positive*
Op1	*carry_on_right(#D, Block1)*
Op2	*carry_on_left(Block2, Block1)*
situation	*relation(#A, #Table, on_table(#PosA))* *relation(Block1, #A, right_on)* *relation(Block2, #Table, on_table(Pos))* *relation(#D, #Table, on_table(#Pos1))*
constraints	*140 <= Block1`width <= 150* *50 <= Block2`width <= 60*
certainty factor	*1.000*

Fig. 8 Another example of planing rule

PR の説明のため task 2 を取り上げる. task 2 を達成するためには, 積木は下から積んでゆく, 積木Dは積木Cより先に積む, という2つの PR が必要である. このような PR は例えば, **Fig. 7**, **Fig. 8** の2つで表される. 前者は状況の記述が任意 (nil と表す) であることから状況に依らないことを表す. Op1, Op2 中の下線 (_) は任意の項目を表す変数であり,“ある積木 Object を別の積木の真上 (center_on) に積むオペレータは, また別の積木を Object の上に積むオペレータより先に実行した方がよい”という意味を持つ. このような PR を適当なオペレータの可能な組み合わせすべてに対して用意しておけば, 積木は下から積んでゆくという知識が表現できる. また, 後者は上の2つの積木は右側から積んでゆくという知識を表す. Fig. 6 に示した SR と同様に, 変数とその拘束条件が設定されている.

4. 知識の利用

4.1 目標の記述と副目標への展開

作業目標は目標状態の記述で与えられる. 例えば task 2 の作業目標は **Fig. 9** で与えられる. 計画生成部はあるオペレータの ACT が目標状態記述の一部と一致するとき, その一致する部分をそのオペレータを実行せよという副目標で置き換える. この操作は可能な置き換えがなくなるまで行われ, 残った状態記述はそのままにしておく. 置き換えられた副目標を目標ノード (goal node, 以下 GN と略す) と呼ぶ. この置き換えは作業開始前に一度だけ行われ, 実行中に変更することはない. 複数のオペレータが置き換えの候補となるときには, できるだけ多くの記述を置き換えられるもの, 初期状態において前提条件が成り立っているものを優先するが, 他の副目標がどのようなオペレータで置き換えられたかには依存しない. task 2 の場合には

```
relation(#A, #Table, on_table(#PosA))
relation(#B, #A, right_on)
relation(#C, #B, left_on)
relation(#D, #B, right_on)
```

Fig. 9 Description of the goal state of task 2

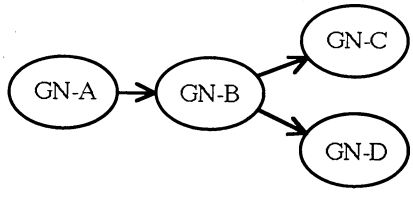

Fig. 10 Goal nodes and orders in task 2

・carry_on_table(#A, #PosA),
・carry_on_right(#B, #A),
・carry_on_left(#C, #B),
・carry_on_right(#D, #B)

の4つのオペレータを実行せよ，という副目標ですべての記述が置き換えられる．

4.2 副目標間の順序づけ

複数のGNが生成されたら，状況に依らないPRを用いてそれらを順序づける．これは作業開始前の大まかな計画生成である．順序づけの情報は順序関係リンクと呼ばれるものに蓄えられる．例えば，task 2においては**Fig. 10** に示す順序づけがなされる．図では，積木Xを運ぶための副目標をGN-Xとし，順序関係リンクを矢印で表している．もちろん，知識が不十分な場合には，完全な順序付けは行われない．

4.3 状況に応じたオペレータの選択

オペレータ選択は1つのオペレータの実行が終了する度に，以下の手順で行われる．このとき，状況によって有効な知識が適宜利用される．

まず，達成されたGNがあるかどうかを調べ，そこから発している順序関係リンクを削除する．このとき，まだ達成されていないGN中で，自分に向かう順序関係リンクのないものを達成可能なGNと呼ぶ．これは，他のGNの達成を待つ必要がないということから，現時点で達成を試みると達成できると判断されるGNである．Fig. 10 の場合には，GN-Aがこれにあたる．達成可能なGNを次々に達成してゆく過程が，作業開始前に立てた計画通りに目標を達成する過程である．

次にPRを用いて，達成可能なGNの集合から現在達成を試みるのが望ましくないものを取り除く．例として，task 2においてFig. 10のような順序付けが行われた後，GN-A，GN-Bが達成された状況（**Fig. 11**）を考える．このとき順序関係リンクはすべて削除されているので，GN-CとGN-Dが達成可能である．ここでFig. 8で示されるPRを適用すると，GN-Dを先に達成した方がよいという判断が下されるので，GN-Cを取り除くことにより，その達成が試みられないようにする．一般に，PRによって先に達成を試みない方がよいと判断されたGNを取り除くが，GNの1つのペアに対し，複数のPRが競合して両方向の順序関係が認められた場合には次のように対処する．正負のPRが競合する場合には正のPRを優先する．正のPR同士が競合する場合には，どちらを先にやってもよいということから何も操作を加えない．負のPR同士が競合する場合には，どちらを先にやってもいけないということから両方のGNを取り除く．

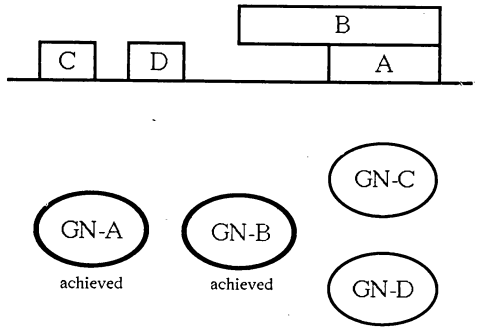

Fig. 11 An intermediate situation in task 2

次に各オペレータの前提条件を調べて，現在の状況で実行可能なオペレータを選び出し，その中から以下のアルゴリズム（step A～D）にしたがってオペレータを決定する．

step A：実行可能なオペレータのうち達成可能なGNと適合するものがあるとき，そのうちのどれかをランダムに選ぶ．例えば，Fig. 11 の状況で，GN-CとGN-Dを順序づける知識がなければともに達成可能であり，一方積木C，Dに対して pick_up，carry_on_xxx が実行可能である．したがって，適合するものとして carry_on_left(#C, #B) と carry_on_right(#D, #B) があり，例えば前者が選ばれる．適当なオペレータがなければ step B 以降でオペレータを決定する．

step B：SRを用いてオペレータを選ぶ．まず，各オペレータに対して，評価値（EV）の初期値0を与えておく．次に現在の状況において適用可能であり，確信度がある閾値を超えるSRを順に取り出し，各オペレータに適用を試みる．あるSR（確信度がCF）があるオペレータに適用されたとすると，そのEVを次式にしたがい修正する．

・正のSRが適用された場合：$EV=EV+CF$
・負のSRが適用された場合：$EV=EV+CF$

可能なSRがすべて適用されたら，最もEVの高いオペレータを選ぶ．もし適用されたSRがまったくなければ，step C 以降でオペレータを決定する．

step C：ACT を用いてオペレータを選ぶ．目標状態の記述のうちGNへの置き換えがなされずに残っている部分をできるだけ達成するようなオペレータを選択する．場合によっては，適当な評価関数を与えておきそれによって目標達成度を測ることもある．積木の積みおろし作業では，GNへの置き換えが完全に行われるのでこの方法は用いられないが，GNへの置き換えはこの手法を用いた前処理であると見なすこともできる．ここで適切なオペレータが見つからなければ step D でオペレータを決定する．

step D：実行可能なオペレータの中からランダムに選ぶ．選択に際し知識は用いられない．

4.4 オペレータの実行

プリミティブオペレータが実行されるとロボットが動作する．マクロオペレータが実行された場合には，展開される下位のオペレータ系列中のすべてのオペレータについて，そのオペレータを実行せよというGNを生成する．さらにそのGN群に対し，1つのGNと次のGNとのすべての間に順序関係リンクを設定しておくことにより，副目標の達成順序を規定する．

プリミティブオペレータはACTに記述された状態が達成された場合に成功，そうでない場合に失敗と見なす．マクロオペレータは下位のGN群がすべて達成されたら成功したと見なす．下位のGN群の達成過程で失敗が生じたらその時点でマクロオペレータは失敗したと見なし，生成したGN群をすべて削除する．

オペレータを実行することによってすでに達成されたGNを破壊することがある．この場合には，破壊されたGNを達成可能にし，さらにそのGNから発していた順序関係リンクを再び設定し直す．例えば，Fig. 11の状態において積木Cを運ぶオペレータを選んで実行すると，すでに達成されていたGN-Bが破壊される．この場合には，GN-B→GN-CとGN-B→GN-Dの2本のリンクを設定し，GN-Bが再び達成可能となる．

5. 学　　習

学習は1つのオペレータの実行が終了する度に，可能ならば行われる．

5.1 オペレータ選択規則の学習

SRはオペレータの選択を評価することにより学習される．オペレータ選択をそのオペレータの実行結果によって次の3つの場合に分ける．

case 1：オペレータの実行に失敗した場合

case 2：実行には成功したが，実行結果の評価ヒューリスティックにより，望ましいオペレータではなかったと判断した場合

case 3：実行に成功し，かつ評価ヒューリスティックにより，望ましいオペレータであったと判断した場合

SR-2	
type	*positive*
situation	*relation(#A, #Table, on_table(#PosA))* *relation(#B, #A, center_on)* *relation(#C, #Table, on_table(#Pos1))* *relation(#D, #Table, on_table(#Pos2))* *relation(#E, #Table, on_table(#Pos3))*
operator	*carry_on_center(#E, #A)*
constraints	*nil*
certainty factor	*0.300*

Fig. 12 An operator selection rule acquired in task 3

オペレータ選択の場合分けが得られたら，それに基づき選択にかかわったSRの学習を以下のように行う．case 1, case 2の場合にはオペレータ選択が望ましいものではなかったと判断し，正のSRの確信度を減少させ，負のSRの確信度を増加させる．case 3の場合にはオペレータ選択が望ましいものであったと判断し，正のSRの確信度を増加させ，負のSRの確信度を減少させる．確信度の変更量はオペレータ選択への寄与量に応じて決定される．選択にかかわったn個のSRのうちi番目のSRの寄与量$value_i$は，その確信度をCF_iとすると次式で与えられる．

$$value_i = K * CF_i / \sum_{i=1}^{n} CF_i$$

ただし，Kは0～1の値をとる適当な係数である．この寄与量を用い，確信度を増加させるときには，

$$NewCF_i = OldCF_i + (1 - OldCF_i) * value_i$$

減少させるときには，

$$NewCF_i = OldCF_i - OldCF_i * value_i$$

という式を用いて確信度を変更する．

評価ヒューリスティックは次の2つを用いている．

“未だ達成されておらず，達成可能でもなかったGNを達成可能にするオペレータは望ましい”

“達成可能なGNに適合するオペレータを実行可能にするオペレータは望ましい”

ARPEX-Lは実行開始前に生成されたGN群（4.1項参照）を次々に達成してゆくことによって，最終的な作業目標を達成する．個々のGNはまず待ち状態（他のGNの達成を待っている状態）から達成可能な状態へ遷移し，さらに各GNに記述された実行すべきオペレータが実行可能になって実際に実行される（4.3項のstep A），という2段階で達成される．上に示した2つのヒューリスティックは，それぞれこの2つの段階を進めるオペレータを望ましいとするものである．

オペレータ選択において適用されたSRがなかった場合には，一般化（5.5項参照）に適したSRがあれば一般化を行い，なければ新たなSRを生成する．このときcase 1, case 2の場合には負のSRを，case 3の場合には正のSRを生成する．初期確信度はオペレータ選択の評価の確からしさを考慮して，case 1の場合には高めの値を，case 2, case 3の場合には低めの値を与える．例えばtask 3において，積木Eを積木Bの真上に置くことにより積木C，Dを置くためのカウンタ・ウェイトとするという知識は，実際にその動作を行ってそれが望ましいオペレータであったと判断されることにより，Fig.

12 に示すような正の SR として獲得される．

このように SR に確信度をもたせ，それを修正する手段を与えることにより不確かさを含む知識を表現できる．例えば，ある SR によって選ばれたオペレータが実行に失敗したとすると，その SR の表現する知識を否定するのではなく確信度を下げることによって確からしさを減らす．その失敗が非常にまれなものであった場合には，その後何回か成功を繰り返すことにより SR の確信度は再び上がり知識の信頼性は高まる．一方，そのオペレータがその状況で失敗し易いものであったなら，SR の確信度は低い値に落ち着き，そのオペレータはあまり使用されなくなる．また，互いに競合するような意味を持つ SR が生成された場合にも，互いの確信度の相対的大小関係が経験的に適当な値に落ち着くため問題はない．SR の数が増えてきた場合にはその評価に時間がかかるという問題があるが，評価を並列化することにより対処できる．

5.2 計画生成規則の学習

PR の学習手順は SR の学習手順と類似している．ただし，PR の場合，確信度は常に 1 としてその変更は行わない．オペレータ選択の step A (4.3 項参照) において達成可能な GN と適合する実行可能なオペレータが複数個あった場合，その状況を記録しておく．オペレータの実行後に SR の場合と同じヒューリスティックを用いて同様な場合分けを行い，選ばれた GN に記述されたオペレータを Op1，他のそれぞれの GN に記述されたオペレータを Op2 として PR を生成する．このとき，そのオペレータ選択が望ましいものであると判断された場合には正の PR を，そうでない場合には負の PR を生成する．例えば，Fig. 11 に示す状況において carry_on_left(#C, #B) が選ばれて実行され，結果として失敗したとすると，“その状況において carry_on_left(#C, #B) を carry_on_right(#D, #B) より先に実行しない方がよい”という負の PR を生成する．

5.3 ACT の学習

ACT のうち e-ACT は学習可能である．ある状況であるオペレータを実行したときに起こった状況変化を e-ACT とする．ただし，動作の不確かさを考慮して，ある回数以上同じ状況の変化が観測されたら，それを e-ACT として採用する．

5.4 マクロオペレータの学習

マクロオペレータはオペレータの実行系列を解析することにより学習されるが，このとき無用なマクロオペレータの生成を防ぐために ACT を用いる．

実行を繰り返すうちに，特定の成功したオペレータの系列がある回数以上生じたら，その系列をマクロオペレータの候補とし，その ACT を評価する．マクロオペレータはある程度のまとまった作業を表すものであり，一方 ACT はオペレータの効果を表すものであるから，“まとまり”を ACT で判断するためのヒューリスティックを与えておく．積木の積みおろし作業においては，

“ACT の記述中にロボットが積木をつかんだ状態の記述が含まれないものをマクロオペレータとする”

というものを用いた．これにより，積木を運ぶ途中の状態で止まるようなマクロオペレータは生成されない．一般的には，何らかの中間状態，例えばボルトだけセットされてナットが締められていない状態，収納箱の蓋が開いた状態，などをその ACT に含むマクロオペレータは生成しない，という方針が考えられる．不必要なマクロオペレータの生成を制限するために ACT を用いることは，1 つの有望な方法である．

MO-1	
operator	*mo1(#A, #B, #PosA)*
precondition	*relation(#A, #Table, on_table(#Pos1))* *relation(#B, #A, center_on)*
suboperators	*carry_on_table(#B, #TempPosB)* *carry_on_table(#A, #PosA)* *carry_on_center(#B, #A)*

Fig. 13 A macro operator acquired in task 1

得られたマクロオペレータの前提条件は，そのオペレータ系列の実行直前の状態とする．さらに，マクロオペレータを述語形式で表現するため適当なオペレータ名と引数を与える．引数はマクロオペレータを構成する各オペレータの引数のうち，マクロオペレータの ACT 中に現れるものとする．すなわち，ACT に現れない引数はそのマクロオペレータにとって重要でないものと判断する．

例として，task 1 におけるマクロオペレータ獲得を示す．task 1 の最短手順は

- carry_on_table(#B, #TempPosB)
- carry_on_table(#A, #PosA)
- carry_on_center(#B, #A)

の 3 つのオペレータからなる．ここで #PosA は積木 A の目標位置，#TempPosB は積木 B の待避位置である．このオペレータ系列がマクロオペレータの候補になったとすると，その ACT は Fig. 5 に示す通りであり，積木をつかんだ状態の記述は含まれないので，マクロオペレータとして採用される．また，引数の候補としては #A, #B, #PosA, #TempPosB の 4 つがあるが，ACT を参照して最初の 3 つだけが採用され (#TempPosB この作業において本質的ではない)，**Fig. 13** に示すマクロオペレータが生成される．

5.5 一般化による学習

ARPEX-L では SR，PR，マクロオペレータを一般化

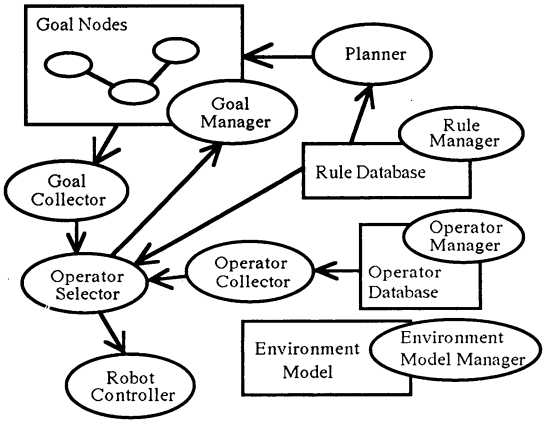

Fig. 14 Constitution of ARPEX-L

することにより，知識の適用範囲の拡大を図る．一般化の方法としては全くの形式のみに基づいた，定数項の変数化および変数の区間の設定，条件節の削除，の2つを用いている．その具体的なアルゴリズムは省略するが，Fig. 6 に示した SR，Fig. 8 に示した PR はこの一般化を行って得られた知識の例である．

6. システムの全体構成とその動作

ARPEX-L は筆者らが開発した知能ロボット用言語 ARP[8] 上に作成され，**Fig. 14** に示すような構成となっている．ARPEX-L に対し作業目標が与えられると，Goal Manager がそれを解釈して GN を生成し，Planner が状況に依らない PR を用いて順序関係を設定する．この初期化が終わると ARPEX-L は通常の実行サイクルに入る．そこでは Goal Collector が達成可能な GN を，Operator Collector が実行可能なオペレータを集めて Operator Selector に送る．Operator Selector は 4. 3 項で述べたアルゴリズムにしたがってオペレータを決定し，プリミティブならば Robot Controller に送って実行させ，マクロならば Goal Manager に情報を送って下位の GN 群を生成させる．Rule Manager，Operator Manager は知識の学習と管理を行う．また，Environment Model Manager が環境モデルを管理する．

7. 積木の積みおろし作業への適用結果

ARPEX-L を用いて Fig. 1 の3つの作業をシミュレーションによって実行した．初期知識としては Table 1 に示した9個のオペレータと積木は下から積んでゆくという知識を表す PR だけを与えておいた．これらの知識は全作業に共通して利用されるものであり，個々の作業に特有の知識は，ARPEX-L を動作させることにより自動的に獲得することができる．

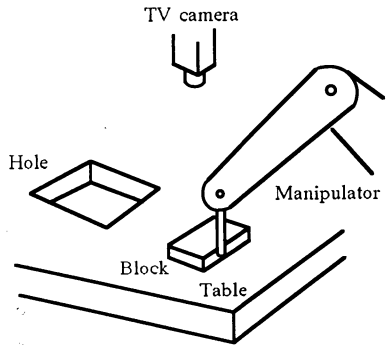

Fig. 15 Overview of pushing block task

task 1 では1回目の試行で最短の6ステップ（ステップ数は実行されたプリミティブオペレータの数で数える）で目標を達成した．その際，SR を1個獲得した．

task 2 においては，1回目の試行では 14 ステップ，2回目の試行では最短の8ステップで目標を達成した．その際，PR を2個，SR を1個獲得した．また，使用する積木の大きさを多少変えた問題を行わせたところ，Fig. 8 に示したものも含めた，一般化された PR，SR を獲得した．

task 3 を行う際，まず積木 E の最終位置をテーブル上のある位置に限定したところ，問題空間が広がりすぎて目標を達成することができなかった．そこで積木 E の最終位置を限定せずに行わせたところ 26 ステップで目標を達成し，積木 E の最終位置は積木 B の真上（center_on）となった．その際，Fig. 12 に示したものも含めて SR を4個獲得した．次にその獲得した知識を用いて，積木 E の最終位置を限定して行わせたところ，1回目の試行は 26 ステップ，2回目の試行は最短の 12 ステップで目標を達成した．この例から与える問題の順序が重要であることが示される．すなわち，易しい問題から徐々に難しい問題を与えてゆくことにより，順調な知識の獲得が可能となる．

また，いずれの作業においても繰り返し試行することにより，作業目標を達成するためのマクロオペレータを獲得した．

8. 積木の押し作業への適用結果

ARPEX-L で実際のマニピュレータを制御し，積木の押し作業（**Fig. 15**）を行わせた．平らなテーブル上に積木と，積木より大きい穴がある．マニピュレータの先に取り付けた棒で積木を押してゆき，穴の中に入れることが目標である．状況（積木の位置・姿勢）は TV カメラを用いた視覚システムによって検出する．ARPEX-L は

積木を押し，その効果を視覚で認識することによって各オペレータの ACT を獲得し，それを用いて目標を達成する．このようにオペレータに関する情報が不完全な問題は従来の問題解決システムは扱えなかったものである．

扱う積木は直方体のものであり，押す位置を **Fig. 16** に示す6か所に限定する．したがって，作業開始前には6個のオペレータが与えられているが，積木の重心位置の偏り，摩擦やすべりなどの影響により積木の挙動をあらかじめ与えておくのは困難であるので，オペレータの ACT は未知である．

重心が左に偏った積木を用いた実験の結果を **Fig. 17** に示す．図中には積木の外形と図心の軌跡が描かれている．オペレータ選択の際には，ACT の分かっているオペレータ中で目標位置との距離を縮めると判断されるものがあればそれを選び（4.3 項の step C），なければ ACT が未知であるオペレータの中から適当に選ぶ（4.3 項の step D）．このとき状況は積木の位置・姿勢で，ACT はそれらの変化量で表現し，目標との距離を評価する関数をあらかじめ与えておいた．この積木の場合，真ん中を押してもまっすぐ進まないため，前進させるためには重心に近いところを押す必要があるが，試行を重ねるに連れ，より的確なオペレータを選ぶようになっていることが図から分かる．

9. 考　　察

9.1 積木の積みおろし作業について

積木の積みおろし作業において，ARPEX-L は与えられた初期知識を基に行動し，成功，あるいは失敗した経験から SR，PR，マクロオペレータを獲得し，より効率的な問題解決を行った．作業全体に共通の知識を与えておくだけで，各作業ごとに手順をプログラミングすることなく，作業を行わせることができた．結果として得られたマクロオペレータはそれぞれの作業を行うためのプログラムであり，簡単な自動ロボット・プログラミングの例であるといえる．ただし，本作業はシミュレーションで行ったため現実世界の不確かさは存在せず，不確かさに対する確信度の有効性も検証されてはいないが，競合する SR に対しては確信度による制御が有効であった．実際の作業における，各知識を用いた問題解決と学習の有効性の検証が今後の課題である．

9.2 積木の押し作業について

積木の押し作業は非常に簡単なものであり，適応制御のような数値的方法を用いてもっと確実に行わせることが可能である．しかし，[本論文のように ARPEX-L という記号処理の枠組みで，現実のロボットに学習を行わせた例は見あたらず，そこに本実験の意義がある．ただし，本論文では積木の押し場所を6か所に限定し，目標までの距離の評価関数をあらかじめ与えておくことにより，記号処理の枠組みで扱いやすくしている．また，本作業において学習し，利用する知識は ACT のみであ

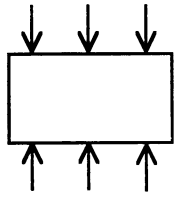

Fig. 16 Six pushing positions

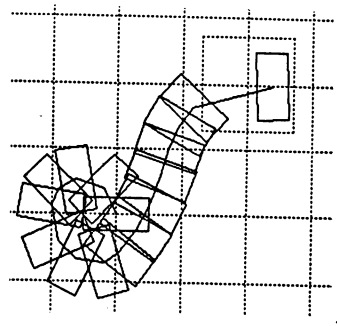

Fig. 17 Experimental results of pushing block task

り，ARPEX-Lの機能の一部を用いているに過ぎない．さらに複雑な作業へ適用してARPEX-Lの問題点を検討することが今後の課題である．より高度な知能の実現には記号処理的アプローチが不可欠であるので，現実の問題に含まれる非記号情報を記号処理の枠組みで扱うための技術の開発も今後の課題である．

9.3 従来の研究との関連，比較

知能ロボットにおける問題解決と学習の統合を扱った研究は見あたらないが，ここでは人工知能，認知科学の分野の研究との関連，比較について述べる．

Anzai[9),10)]は，“行動による学習，理解（learning, understanding by doing)”という考え方を提案し，人間が船の操縦技術を習得する過程を認知心理的見地から考察し，プロダクション・システムを用いてシミュレートした．この研究では，オペレータの効果が与えられていない場合には，まずそれを獲得することにより問題空間を規定する行為が行われ，さらに行動することによって知識を獲得し，それに伴い問題解決戦略がより効率的なものへ変化してゆくことを示した．ARPEX-Lはこのような考え方を知能ロボットを対象に，問題から独立した枠組ねとして具現化した例である．

Mintonらによる PRODIGY[2)]は，正負のSRや正のPRにあたる探索制御規則を獲得する．説明に基づく特殊化（EBS）を行って意味のある知識を獲得している点が特徴である．また，Carbonell[3)]は類推推論を基本にした問題解決と学習の統合システムを提案した．しかしいずれのシステムも，実際のロボット制御の枠組みとしては，前にも述べたオペレータに関する情報の不十分さへの対処や不確かな知識の表現という点で不十分である．

SRの確信度のように，オペレータの制御規則に重みをつけ，その値を修正することにより学習を行うシステムとして，Langleyによる SAGE[4)]がある．SAGEは成功した局面で用いられた規則の重みを増加させることにより，有用な規則を強化するが，ARPEX-Lでは失敗した局面からも学習を行う．また，SAGEでは規則適用の際に最大の重みを持つものだけを用いるが，ARPEX-Lではすべての適用可能な知識を用い，その寄与分に応じた修正を行っているため，競合するような知識も同時に考慮される．したがって，より的確なオペレータを選ぶことが可能である．確信度を用いて不確かな知識を表現する方法にはさまざまなものがある[11)]．本論文で示したSRの確信度の利用・修正方法をさまざらな例題に適用し，他の方法との比較を行うことは今後の課題である．

PRのような副目標間の順序関係を利用してプランニングを行うシステムとして，長田ら[12)]によるものがある．このシステムでは，オペレータに関する情報から順序関係を自動的に導き出して利用するが，オペレータに関する情報が不完全な場合にはあまり有効ではない．一方，ARPEX-Lでは順序関係をPRで直接表現し，その学習を可能としているため，不確かさの存在する実世界への応用に適している．

マクロオペレータの獲得に関する研究の最初のものは，STRIPSにおける MACROP[1)]である．STRIPSでは三角表を用いて変数間の依存関係を考慮し，一般化されたマクロオペレータを獲得するが，無用なオペレータの生成を制限することが必要である．Minton[13)]はS-マクロ（頻繁に用いられるオペレータ），T-マクロ（複雑な問題を解くときに用いられるオペレータ）という2つの概念を導入して，効率的なマクロオペレータの管理を行った．また，山田ら[14)]はマクロオペレータ抽出の基準として，完全因果性という概念を用いた．一方，ARPEX-LではACTというオペレータの意味的情報をマクロオペレータ抽出の基準としている点が特徴的であるが，判断のために問題固有の知識を与えておく必要がある．

9.4 その他の問題点，課題

ARPEX-Lは基礎的なものであり，ここまでに述べたもの以外にも，次のような問題点，課題が残されている．

- 目標状態記述のGNへの置き換えを作業開始前に行ってしまわず，必要十分な情報が得られるまで遅らせることが望ましい．また，常に一動作ごとにオペレータを決定するのではなく，ある程度以上の確からしさを持つ知識を用いた多段階の推論を実行時に行うことも考えられる．
- 他の問題解決手法の導入を検討すべきである．特に類推推論に基づく問題解決は有効であろう．ただし，SR，PRによる知識の表現と利用は部分的な類推を行っていると考えることもできる．
- 他の学習手法の導入を検討すべきである．特に説明に基づく学習などの，知識を用いた学習の制御が有効であろう．また，現在用いている一般化手法は知識の形式的情報のみに基づいたものであって，それによって得られる知識の適用範囲は限られている．したがって，例えば物理法則のような深い知識や，知識の階層性を利用して一般化を行い，より汎用的な知識を獲得することが望ましい．

10. おわりに

本論文では，実世界での応用を目指して作成した，知能ロボットのための問題解決と学習の統合システム，ARPEX-Lについて述べた．積木の積みおろし作業と積木の押し作業の2つの作業にARPEX-Lを適用し，その有効性を確認するとともに，問題点，今後の課題を明

らかにした．前者の作業では知能ロボット自動プログラミングの可能性を示し，また後者の作業では実際のロボットに記号処理の枠組みを用いて学習を行わせることができた．

謝辞：論文をまとめるにあたり貴重なご意見を頂きました，大阪大学白井良明教授に感謝致します．

参考文献

1) Fikes, R. E., Hart, P.: and Nilsson, N. J., Learning and Executing Generalized Robot Plans, Artificial Intelligence, Vol. 3, pp. 251-288, 1972.
2) Minton, S. and Carbonell, J. G.: Strategies for Learning Search Control Rules : An Explanation-Based Approach, in Proc. of the 10 th IJCAI, pp. 228-235, 1987.
3) Carbonell, J. G.: Derivational Analogy : A Theory of Reconstructive Problem Solving and Expertize Acquisition, in Machine Learning : An Artificial Intelligence Approach volume II, R. S. Michalski et al. (eds.), pp. 371-392, Morgan Kaufmann, Los Altos, California, 1986.
4) Langley, P.: Learning Effective Search Heuristics, in Proc. of the 8 th IJCAI, pp. 419-421, 1983.
5) Mitchell, T. M.: Learning and Problem Solving, in Proc. of the 8 th IJCAI, pp. 1139-1151, 1983.
6) Firby, R. J.: An Investigation into Reactive Planning in Complex Domains, in Proc. of AAAI-87, Seattle, pp. 202-206, 1987.
7) Goergeff, M. P. and Lansky, A. L.: Reactive Reasoning and Planing, in Proc. of AAAI-87, Seattle, pp. 677-682, 1987.
8) 三浦純，下山勲，三浦宏文：知能ロボット用言語 ARP の開発，日本システム工学会誌，Vol. 11, No. 2, pp. 33-46, 1987.
9) Anzai, Y.: Cognitive Control of Real-Time Event-Driven Systems, Cognitive Science, Vol. 8, pp. 221-254, 1984.
10) Anzai, Y.: Doing, Understanding, and Learning in Problem Solving, in Production System Models of Learning and Development, D. Klahr et al. (eds.), pp. 55-97, The MIT Press, Cambridge, Mass. 1987.
11) 田中幸吉編：知識工学，pp. 187-199，朝倉書店，1984.
12) 長田正，寺本良明：メタ知識を利用したプランニングシステム，人工知能学会誌，Vol. 3, No. 2, pp. 186-195, 1988.
13) Minton, S.: Selectively Generalizing Plans for Problem-Solving, in Proc. of the 9 th IJCAI, pp. 596-599, 1985.
14) 山田誠二，辻三郎：完全因果性によるマクロオペレータの選択的学習，人工知能学会誌，Vol. 4, No. 3, pp. 321-329, 1989.