人工知能　85－2
(1992.11.19)

準同型変換による抽象化を用いたプランニング

馬淵　浩司*　　赤間　清*　　宮本　衛市*

*北海道大学　工学部　情報工学科

*〒060　札幌市北区北13条西8丁目

プランニングとは、初期状態と目標状態を与えることで、目標状態に達するための行動系列を求めることをいう。初期状態を目標状態に変化させるような作用素の列がプランである。本論文では、まず、状態空間上の具体プログラムと抽象プログラムを一般化論理プログラム (GLP) に基づいて論理の枠組で定式化する。次に、具体的プランニングと抽象的プランニングを行ない、プランを求める。また、これら2つの定式化されたプログラムをリンクする準同型変換を定義し、証明する。準同型変換とは、直感的に、写像前の状態間の関係の縮小版が写像後でも保存されるような写像をいう。さらに、具体プログラムと抽象プログラムの関係を示し、準同型変換によるプランニングの有効性を示す。

キーワード：準同型変換, 抽象化, プランニング, 問題解決, 一般化論理プログラム

Planning by using Abstraction of Homomorphism

Hiroshi MABUCHI　Kiyoshi AKAMA　Eiichi MIYAMOTO

Dept. of Information Eng., Faculty of Eng., Hokkaido Univ.

Kita 13, Nishi 8, Kita-ku, Sapporo, 060, JAPAN

Planning means to look for the action sequences to arrive at the goal state by giving the initial state and the goal state. The plan is a sequences of operators which converts the initial state to the goal state. In this paper, we propose the problem abstraction by using homomorphisms and show an example for looking for the plan by using this method. Intuitively homomorphism means a mapping which preserves relations among conditions. From this example we show the availability of abstraction about the problem solving, and we remark the relevancy to the planning.

Key Words : Homomorphism,Abstraction,Planning,Problem Solving,GLP(Generalized Logic Program)

1 まえがき

問題解決 [3][4] や計画立案 (プランニング)[2] は、人工知能の基礎分野の1つであり、特に、計画の自動立案は社会的要請が強い。

プランニングとは、初期状態と目標状態を与えることで、目標状態に達するための行動系列を求めることをいう。初期状態を目標状態に変化させるような作用素の列がプランである。

計画立案は、計画の表現に階層性があるか否かによって、非階層的計画と階層的計画の2つに大きく分けることができる。問題解決やプランニングを階層的に行うことで、問題解決過程の探索の複雑さが大きく減少することはよく知られている。本論文では、まず、プランニングに抽象化を用いるシステムの1つである ABSTRIPS[1] を検討して、その問題点を改善する。

次に、具体的な状態空間と抽象的な状態空間を、一般化論理プログラム (GLP)[5] に基づいて論理の枠組で定式化する。そして、それら2つの状態空間上の具体プログラムと抽象プログラムも論理の枠組で定式化する。そして、具体的プランニングと抽象的プランニングを行ない、プランを求める。また、これら2つの定式化されたプログラムをリンクする準同型変換を定義し、証明する。準同型変換とは、直感的に、写像前の状態間の関係の縮小版が写像後でも保存されるような写像をいう。さらに、具体プログラムと抽象プログラムの関係を示し、準同型変換によるプランニングの有効性を示す。

2 ABSTRIPS からの改善点

ABSTRIPS の問題点に対して、以下のような改善を試みる。

1. ABSTRIPS では、プランニング問題を構成する要素のうち、オペレータしか抽象化されていない。そこで、本論文では、プランニング問題で与えられる全ての要素を抽象化する。すなわち、オペレーターのみではなく、初期状態、目標状態も抽象化する。勿論、抽象化されたオペレータの適用で生成される新しい中間状態も抽象化されている。

2. 述語の削除による抽象化という手法は、すべての抽象化を捕えているわけではない。そこで、本論文では、準同型変換による抽象化という新しい手法を提案する。

3. STRIPS 以来の3つ組によるオペレータ (前提条件、削除リスト、追加リスト) の伝統的な知識表現は、知識表現の形に強く依存しているため記述できる行為が限られている。また、とても手続き的で、それゆえ抽象化や変換がしにくい。そこで、本論文では、オペレータはそのオペレータの適用前と適用後のそれぞれの状態のペアとして、宣言的に表現する。

3 論理の枠組によるプランニング問題の記述

3.1 問題設定

本論文では、例として2ディスクのハノイの塔の問題を取り上げる。

Fig.1 で示すように、ペグは p1,p2,p3 の3本あり、ディスクは大と小の2枚ある。この時に、小さいディスク、大きいディスクともにペグ1にある初期状態 A から、小さいディスク、大きいディスクともにペグ3にある目標状態 B に移す問題を考える。

移動する場合の条件として、次のようなことが決まっている。

- 一度に1つのディスクしか移動できない。
- どのディスクも、それより小さいディスクの上に置いてはならない。

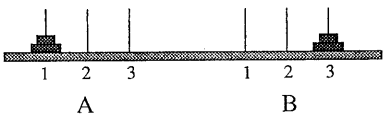

Fig.1 Tower of Hanoi

3.2 アトムの集合を用いた問題の状態表現

ここでは、3.1節で述べた問題の状態をアトム(原子論理式)の集合を用いて表す。

今、大きいディスクがペグxにあるというアトムを(onbig x)とし、小さいディスクがペグxにあるというアトムを(onsmall x)とする。また、大きいディスクがペグxにないというアトムを(Nonbig x)とし、小さいディスクがペグxにないというアトムを(Nonsmall x)とする。

ハノイの塔の問題(Fig.1)では、初期状態A、目標状態Bをアトムの集合の形で表すと次のようになる。

A = {(onbig p1),(Nonbig p2),
(Nonbig p3),(onsmall p1),
(Nonsmall p2),(Nonsmall p3)}

B = {(Nonbig p1),(Nonbig p2),
(onbig p3),(Nonsmall p1),
(Nonsmall p2),(onsmall p3)}

3.3 プランニング問題の記述

本研究では、GLPの理論に基盤を置き、その具現であるプログラミング言語UL/α[6]を用いる。UL/αは、集合を扱うことができるなど、論理プログラムの扱う対象をGLPの理論に基づいて拡張されている。また、具体プログラムも抽象プログラムも同じように、論理の枠組で定式化することができる。

まず、プランニング問題を記述しよう。ここでプランニング問題を記述する述語として次のpathとnextを導入する。

(path *i:初期状態 *g:目標状態
*S:オペレータ列)

(next *s1:状態 *s2:状態 *T:オペレータ)

pathは問題で与えられる初期状態*i, 目標状態*g,*iを*gまで遷移させるオペレータの列である*Sからなる述語、**next**は状態*s1, 状態*s2, 状態*s1を*s2に遷移させるオペレータ*Tからなる。

次にプログラムを示す。

```
(as (path *g *g ()))
(as (path *i *g (*o . *answer))
    (next *s1 *s2 *o)
    (path *s2 *g *answer))
```

プランニングとは、この述語pathの*iと*gにそれぞれ具体的な初期状態と目標状態を与えてpathを展開し、初期状態から目標状態へのオペレータ列*Sを求めることである。

3.4 GLPによる変換ルールの表現

3.2節の集合による問題の状態表現を採用して、変換ルールをGLPのプログラムを用いて表現すると次のようになる。これら2つのプログラムは、変換ルールnextのプログラムで、nextは3つの引数を持っている。第1引数は、最初の中括弧({})で囲まれた部分で、大小のディスクがどのペグにあるかという状態が集合の形で表現されている。第2引数は、次の中括弧で囲まれた部分で、第3引数は、それぞれ最後の(MoveBig *x *y)と(MoveSmall *x *y)である。この変換ルールnextは、第3引数によって第1引数の状態から第2引数の状態へ変換することを意味する。ここで、C_1,C_2は節の番号を示す。

```
C1: (as (next {(onbig *x)(Nonbig *y)
           (Nonbig *z)(Nonsmall *x)
           (onsmall *z)(Nonsmall *y)}
             {(onbig *y)(Nonbig *x)
           (Nonbig *z)(Nonsmall *x)
           (onsmall *z)(Nonsmall *y)}
        (MoveBig *x *y)))

C2: (as (next {(onsmall *x)(Nonsmall *y)
                (Nonsmall *z)(onbig *p)
                (Nonbig *q)(Nonbig *r)}
               {(onsmall *y)(Nonsmall *x)
                (Nonsmall *z)(onbig *p)
                (Nonbig *q)(Nonbig *r)}
        (MoveSmall *x *y)))
```

ここで、(MoveBig x y)は、大きいディスクをペグxからペグyへ動かすことを意味し、(MoveSmall x y)は、小さいディスクをペグxからペグyへ動かすことを意味する。

3.5 GLP による具体プログラムの定式化

具体プログラムを GLP の理論の枠組で定式化するために、$\mathcal{A}_1$,$\mathcal{G}_1$,$\mathcal{S}_1$,μ_1を考える [5] 。$\mathcal{A}_1$ はアトム (onsmall *x) や (onbig *x) などを元に持つ集合で、$\mathcal{G}_1$は $\mathcal{A}_1$の元で変数を含まないアトム全体の集合である。また、$\mathcal{S}_1$は代入の集合で、μ_1は $\mathcal{S}_1$の代入によって、アトムが $\mathcal{A}_1$の元から別の $\mathcal{A}_1$の元 (同じこともある) へ変化するという意味の 3 つ組の元の集合である。例えば、({*/1},(onsmall *x),(onsmall 1)) はμ_1の元であり、代入 {*/1} によって (onsmall *x) が (onsmall 1) に変化することを意味する。$\mathcal{A}_1$,$\mathcal{G}_1$,$\mathcal{S}_1$,μ_1をこれらの元の例によって示す。

```
A1 = {(onsmall *x),(Nonsmall *y),
        (onbig *x),(Nonbig *z),...}
G1 = {(onsmall 1),(Nonsmall 2),
        (onbig *z),(Nonbig 3),...}
S1 = {{*x/1},{*y/2},{*x/*z},{*z/3},...}
μ 1 =
{({*x/1},(onsmall *x),(onsmall 1)),
({*y/2},(Nonsmall *y),(Nonsmall 2)),
({*x/*z},(onbig *x),(onbig *z)),
({*z/3},(Nonbig *z),(Nonbig 3)),...}
```

これら 4 つ組$<\mathcal{A}_1,\mathcal{G}_1,\mathcal{S}_1,\mu_1>$の上に次に示す具体プログラム P_1を考える。

$$P_1 = \{C_1, C_2\}$$

C_1,C_2は 3.4 節に示してある節である。

3.6 具体的なプランニング

本節では、具体的なプランニングについて考える。プランニングとは、初期状態と目標状態を与えて、初期状態から目標状態に達するための行動系列を求めることである。例えば、初期状態 ⟨init1⟩ と目標状態 ⟨final1⟩ は次のようなものである。

```
< init1 > =
  {(onsmall 1)(Nonsmall 2)(Nonsmall 3)
   (onbig 1)(Nonbig 2)(Nonbig 3)}
< final1 > =
  {(onsmall 3)(Nonsmall 2)(Nonsmall 1)
   (onbig 3)(Nonbig 1)(Nonbig 2)}
```

プランを求めるために次に示すQ_1を起動する。

Q1 = (path ⟨init1⟩ ⟨final1⟩ *line1)

⟨init1⟩ と ⟨final1⟩ にはそれぞれ具体例が入る。*line1 は変数で、Q_1の実行結果である初期状態から目標状態に達するための行動系列が代入される。例えば、この例では、θ_1 = {*line1/{(MoveSmall 1 2),(MoveBig 1 3),(MoveSmall 2 3)}} が適用され、行動系列が求まる。

ここで、Q_1,θ_1を正確に表現するために、次に示すプログラム P_1の意味 (= $\mathcal{M}(P_1)$) が必要になる。

$\mathcal{M}(P_1)$ =

```
{path({(onbig 3),(Nonbig 1),(Nonbig 2),
  (Nonsmall 1),(onsmall 2),(Nonsmall 3)},
 {(onbig 3),(Nonbig 1),(Nonbig 2),
  (onsmall 3),(Nonsmall 1),(Nonsmall 2)},
(MoveSmall 2 3)),

 next
({(onbig 1),(Nonbig 2),(Nonbig 3),
  (Nonsmall 1),(onsmall 2),(Nonsmall 3)},
({(onbig 3),(Nonbig 1),(Nonbig 2),
  (Nonsmall 1),(onsmall 2),(Nonsmall 3)},
(MoveBig 1 3)),...}
```

このとき、次の関係が成り立ち、θ_1を求めることが、プランを求めることになる。

$$Q_1\theta_1 \in \mathcal{M}(P_1)$$

3.7 具体的なプラン

本節では、具体的なプランを求める。プランを求めることは、前節で述べた行動系列*line1 に θ_1の代入を作用させることである。ここで、S_1 ~ S_4は、7 章の Fig.2 に対応している。(Q_1,P_1) のプランは次のようになる。

```
{(MoveSmall 1 2),(MoveBig 1 3),
                 (MoveSmall 2 3)}
```

状態推移を $path_1$とすると次のようになる。

```
path1 =
[S1:
{(onsmall 1),(Nonsmall 2),(Nonsmall 3),
 (onbig 1),(Nonbig 2),(Nonbig 3)}
                  ↓...(MoveSmall 1 2)
 S2:
{(onsmall 2),(Nonsmall 1),(Nonsmall 3),
 (onbig 1),(Nonbig 2),(Nonbig 3)}
                  ↓...(MoveBig 1 3)
 S3:
{(onbig 3),(Nonbig 1),(Nonbig 2),
 (Nonsmall 1),(onsmall 2),(Nonsmall 3)}
                  ↓...(MoveSmall 2 3)
 S4:
{(onsmall 3),(Nonsmall 2),(Nonsmall 1),
 (onbig 3),(Nonbig 1),(Nonbig 2)}]
```

4 GLPによる抽象的なプランニング問題

4.1 GLPによる抽象プログラムの定式化

抽象プログラムをGLPの理論の枠組で定式化するために、次に示すような4つ組$\mathcal{A}_2$,$\mathcal{G}_2$,$\mathcal{S}_2$,μ_2を考える[5]。$\mathcal{A}_2$はアトム☆, ★,(onbig *x)などを元に持つ集合で、$\mathcal{G}_2$は$\mathcal{A}_2$の元で変数を含まないアトム全体の集合である。また、$\mathcal{S}_2$は代入の集合で、μ_2は$\mathcal{S}_2$の代入によって、アトムが$\mathcal{A}_2$の元から別の$\mathcal{A}_2$の元(同じこともある)へ変化するという意味の3つ組の元の集合である。$\mathcal{A}_2$,$\mathcal{G}_2$,$\mathcal{S}_2$,μ_2をこれらの元の例によって示す。

$\mathcal{A}_2$ =
```
{☆, ★,(onbig *x),(Nonbig *z),...}
```
$\mathcal{G}_2$ =
```
{☆, ★,(onbig 1),(Nonbig 3),...}
```
$\mathcal{S}_2 = \mathcal{S}_1$

μ 2 =
```
{({*x/1},  ☆,  ☆),({*y/2},  ★,  ★),
({*x/*z},(onbig *x),(onbig *z)),
({*z/3},(Nonbig *z),(Nonbig 3)),...}
```

これら4つ組$<\mathcal{A}_2,\mathcal{G}_2,\mathcal{S}_2,\mu_2>$の上に次に示す抽象プログラム$P_2$を考える。

$$P_2 = \{C_1', C_2'\}$$

ここで、C_1',C_2'は次のような節である。

```
C'1:(as (next {(onbig *x)(Nonbig *y)
                (Nonbig *z) ★ ☆}
              {(onbig *y)(Nonbig *x)
                (Nonbig *z) ★ ☆}
        (MoveBig *x *y)))

C'2:(as (next {☆ ★ (onbig *p)
              (Nonbig *q)(Nonbig *r)}
              {☆ ★ (onbig *p)
              (Nonbig *q)(Nonbig *r)}
        (MoveSmall *x *y)))
```

4.2 抽象的なプランニング

本節では、抽象的なプランニングについて考える。3.6節と同様に、初期状態と目標状態を与えて、初期状態から目標状態に達するための行動系列を求める。例えば、初期状態⟨init2⟩と目標状態⟨final2⟩は次のようなものとする。

```
< init2 > =
{☆, ★,
    (onbig 1),(Nonbig 2),(Nonbig 3)}
< final2 > =
{☆, ★,
    (onbig 3),(Nonbig 1),(Nonbig 2)}
```

プランニングを求めるために次に示すQ_2を起動する。

Q2 = (path ⟨init2⟩ ⟨final2⟩ *line2)

⟨init2⟩と⟨final2⟩にはそれぞれ具体例が入る。*line2は変数で、Q_2の実行結果である初期状態から目標状態に達するための行動系列が代入される。

ここで、Q_2,θ_2を正確に表現するために、次に示すプログラムP_2の意味(= $\mathcal{M}(P_2)$)が必要になる。

プログラムP_2の意味(= $\mathcal{M}(P_2)$)は次のようになる。

$\mathcal{M}(P_2)$ =
```
{path
({(onbig 3),(Nonbig 1),(Nonbig 2),
                          ☆, ★},
```

```
{(onbig 3),(Nonbig 1),(Nonbig 2),
                              ☆, ★},
(MoveSmall 2 3)),

 next
({(onbig 1),(Nonbig 2),(Nonbig 3),
                              ★, ☆},
{(onbig 3),(Nonbig 1),(Nonbig 2),
                              ★, ☆},
(MoveBig 1 3)),...}
```

このとき、次のような関係が成り立ち、θ_2を求めることがプランを求めることになる。

$$Q_2\theta_2 \in \mathcal{M}(P_2)$$

ここで、θ_2 = {*line2/((MoveSmall 1 2),(MoveBig 1 3),(MoveSmall 2 3))}である。

4.3 抽象的なプラン

本節では、抽象的なプランを求める。プランを求めることは、前節で述べた行動系列*line2にθ_2を作用させることである。ここで、$S_1' \sim S_4'$は、7章のFig.2に対応している。(Q_2,P_2)のプランは次のようになる。

```
{(MoveSmall 1 2),(MoveBig 1 3),
                    (MoveSmall 2 3)}
```

状態推移を$path_2$とすると次のようになる。

```
path2 =
[S'1: {☆, ★,
    (onbig 1),(Nonbig 2),(Nonbig 3)}
                  ↓...(MoveSmall 1 2)
 S'2: {☆, ★,
    (onbig 1),(Nonbig 2),(Nonbig 3)}
                  ↓...(MoveBig 1 3)
 S'3: {☆, ★,
    (onbig 3),(Nonbig 1),(Nonbig 2)}
                  ↓...(MoveSmall 2 3)
 S'4: {☆, ★,
    (onbig 3),(Nonbig 1),(Nonbig 2)}]
```

5 準同型変換 (Homomorphism)

5.1 準同型変換によるリンク

準同型変換とは、直感的に、ある事物間の写像前の関係の縮小版が写像後においても、その関係を保持しているような写像を言う。本節では、3.5節の4つ組$<\mathcal{A}_1,\mathcal{G}_1,\mathcal{S}_1,\mu_1>$と4.1節の4つ組$<\mathcal{A}_2,\mathcal{G}_2,\mathcal{S}_2,\mu_2>$の準同型変換によるリンクを考える。

$\mathcal{A}_1$から$\mathcal{A}_2$の変換について見てみる。まず、$h_{\mathcal{A}}$:$\mathcal{A}_1 \rightarrow \mathcal{A}_2$という写像を考える。これは、(onsmall *x),…は☆に、(Nonsmall *y),…は★に置き換え、(onbig *x)などは変化しない写像である。また、$h_{\mathcal{S}}$:$\mathcal{S}_1 \rightarrow \mathcal{S}_2$を恒等写像とする。このとき、次のようにhを定義する。

$$\mathrm{h} = \langle h_{\mathcal{A}}, h_{\mathcal{S}} \rangle$$

5.2 準同型変換の証明

本節では、前節のhが準同型変換となることを示す。その定義は、hが次の2つの条件を満たすことである。

1. $\mathrm{h}(\mu_1) \subset \mu_2$
2. $\mathrm{h}(\mathcal{G}_1) \subset \mathcal{G}_2$

ここで、$\mathrm{h}(\mu_1)$, $\mathrm{h}(\mathcal{G}_1)$とは次のような意味である。

$\mathrm{h}(\mu_1) = \{(h_{\mathcal{S}}(\mathrm{s}),h_{\mathcal{A}}(\mathrm{a}),h_{\mathcal{A}}(\mathrm{b})) \mid (\mathrm{s,a,b}) \in \mu_1\}$
$\mathrm{h}(\mathcal{G}_1) = \{h_{\mathcal{A}}(\mathrm{g}) \mid (\mathrm{g}) \in \mathcal{G}_1\}$

ここで、sは代入、aは代入前のアトム、bは代入後のアトム、gはグラウンドアトムを示す。注意すべき点は、sは$h_{\mathcal{S}}$で、a,b,gは$h_{\mathcal{A}}$で別々に変換されることである。

μ_1の各3つ組の要素に準同型変換(h)を行なうと、次のようになる。

h({*x/1},(onsmall *x),(onsmall 1))
=({*x/1}, ☆, ☆) $\in \mu_2$

h({*y/2},(Nonsmall *y),(Nonsmall 2))
=({*y/2}, ★, ★) $\in \mu_2$

h({*x/*z},(onbig *x),(onbig *z))
=({*x/*z},(onbig *x),(onbig *z) $\in \mu_2$

h({*z/3},(Nonbig *z),(Nonbig 3))
=({*z/3},(Nonbig *z),(Nonbig 3) $\in \mu_2$
⋮

これらより、次の関係が言える。

$$\mathrm{h}(\mu_1) \subset \mu_2$$

次に、$\mathcal{G}_1$について見てみる。
$\mathrm{h}(\mathcal{G}_1) = h_{\mathcal{A}}(\mathcal{G}_1)$
$= h_{\mathcal{A}}(\{$(onsmall 1),(Nonsmall 2),
(onbig 1),(Nonbig 3),$\cdots\})$
$= \{$ ☆, ★,(onbig 1),(Nonbig 3),$\cdots\}$
これらより、次の関係が言える。

$$\mathrm{h}(\mathcal{G}_1) \subset \mathcal{G}_2$$

6 具体プログラムと抽象プログラム

6.1 具体プログラムと抽象プログラムの関係

本節では、本論文で述べた具体プログラム P_1 と抽象プログラム P_2の関係について述べる。P_1 は次のようなプログラムである。

$P_1 = \{C_1, C_2\} =$

```
{(as (next {(onbig *x)(Nonbig *y)
           (Nonbig *z)(Nonsmall *x)
           (onsmall *z)(Nonsmall *y)}
             {(onbig *y)(Nonbig *x)
           (Nonbig *z)(Nonsmall *x)
           (onsmall *z)(Nonsmall *y)}
      (MoveBig *x *y)))

 (as (next {(onsmall *x)(Nonsmall *y)
            (Nonsmall *z)(onbig *p)
            (Nonbig *q)(Nonbig *r)}
           {(onsmall *y)(Nonsmall *x)
            (Nonsmall *z)(onbig *p)
            (Nonbig *q)(Nonbig *r)}
      (MoveSmall *x *y)))}
```

プログラム P_1に対して、準同型変換を行なうと次のようになる。

$\mathrm{h}(P_1) =$

```
{(as (next {(onbig *x)(Nonbig *y)
                (Nonbig *z) ☆ ★}
           {(onbig *y)(Nonbig *x)
                (Nonbig *z) ☆ ★}
      (MoveBig *x *y)))

 (as (next {☆ ★ (onbig *p)
           (Nonbig *q)(Nonbig *r)}
           {☆ ★ (onbig *p)
           (Nonbig *q)(Nonbig *r)}
      (MoveSmall *x *y)))}
```

これは、P_2に等しい。つまり、次の式が成り立つ。

$$P_2 = \mathrm{h}(P_1)$$

6.2 準同型変換の意味

$\mathcal{M}(P_1)$,$\mathcal{M}(P_2)$ は、3.6 節と 4.2 節で述べた通りである。

h は準同型変換だから、次の関係が成り立つことが証明されている [7]。

$$\mathcal{M}(\mathrm{h}(P_1)) \supset \mathrm{h}(\mathcal{M}(P_1))$$

この関係式が、例について成り立つことは明らかである。

さらに、$path_1$と $path_2$の関係を述べる。それぞれ $path_1$,$path_2$は既に、3.7 節と 4.3 節で述べてある。$path_1$の $S_1 \sim S_4$とプランの 3 つのオペレータの計 7 つに関して準同型変換を行なう。

$\mathrm{h}(S_1) = S_1'$
$\mathrm{h}(S_2) = S_2'$
$\mathrm{h}(S_3) = S_3'$
$\mathrm{h}(S_4) = S_4'$
h((MoveSmall 1 2)) = (MoveSmall 1 2)
h((MoveBig 1 3)) = (MoveBig 1 3)
h((MoveSmall 2 3)) = (MoveSmall 2 3)
これらから、$\mathrm{h}(path_1) = path_2$がいえる。

7 プランニング

論理の枠組で定式化された具体プログラム P_1 と抽象プログラム P_2があり、それぞれのプラン

ニングを行なうときに起動すべき節を Q_1,Q_2 とする。Fig.2 で示すように、(Q_1,P_1) のプランは、段階 1 の $S_1 \sim S_4$(3.7 節) で表され、(Q_2,P_2) のプランは、段階 2 の $S'_1 \sim S'_4$(4.3 節) で表される。S'_1,S'_2,S'_3,S'_4は、それぞれ S_1,S_2,S_3,S_4を準同型変換したものである (6.2 節)。段階 2 の $S'_1 = S'_2$,$S'_3 = S'_4$ の関係より、段階 3 では、$S'_2 \rightarrow S'_3$ の 1 ステップのみにプランが減少している。つまり、(MoveSmall 1 2) と (MoveSmall 2 3) は意味を持たないことが分かる。これは、抽象プログラム P_2の C'_2 は恒等写像の節なので省略できるということである。よって、段階 3 の抽象プログラムは $\{C'_1\}$ となる。この省略の分と☆と★に置き換えた分、プランを求めるための処理が減り、簡単に問題を解くことができる。

準同型変換後の $S'_1 \sim S'_4$(段階 2) は、$S_1 \sim S_4$(段階 1) の関係を保持した縮小版であり、お互いの関係が分かっている。これより、プランニングを行なう場合、最も求め易い段階 3 のレベルで行動系列を求め、次に、段階 2 で段階 1 のステップの数に合わせる。そして、段階 2 の 1 つ 1 つの状態から、広い目標領域を作り、その大きな的に到達する行動系列を求める。

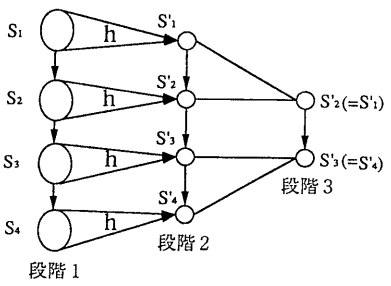

Fig.2 Homomorphism

8 むすび

本研究では、GLP に基づいて、具体プログラムと抽象プログラムをそれぞれ論理の枠組で定式化し、これら 2 つの定式化されたプログラムをリンクする準同型変換を定義し、証明した。また、具体プログラムと抽象プログラムの関係を示した。準同型変換によるプランニングによって、アトムを抽象化したことによって、プランを求めるための処理が減り簡単に問題を解くことができた。さらに、準同型変換後の段階で恒等写像の節が生じることもあり、この節の省略によって、アトムの抽象化に加えて、準同型変換前よりもプランを求めるための処理が減り簡単に問題を解くことができることを説明した。

今後の課題としては、不完全な情報しか得られない状況の中でのプランニングや抽象階層の自動生成などが挙げられる。

参考文献

[1] Earl D.Sacerdoti:Planning in a Hierarchy of Abstraction Spaces, Artificial Intelligence 5 pp.115-135 (1974)

[2] Craig A. Knoblock, Josh D. Tenenberg, Qiang Yang: Characterizing Abstraction Hierarchies for Planning, Hierarchy in Planning, pp. 692-697

[3] Amy Unruth, Paul S. Rosenbloom: Abstractiom in Problem Solving and Learnig, Proceedings of the eleventh International Joint Conference on Artificial Intelligence, pp.681-687 (1989)

[4] Craig A. Knoblock: Learning Abstraction Hierarchies for Problem Solving, Proceedings of the Eighth National Conference on Artificial Intelligence, pp.1004-1009 (1990)

[5] 赤間 清: 媒介表現系上の論理プログラムの宣言的意味論, Proc. of LPC '90 (1990)

[6] 赤間 清: 意味計算 2 制約付き変数とユーザー定義オブジェクト 情報処理学会 自然言語研究会資料, 74-1, pp.1-8 (1989)

[7] Akama K.: Homomorphism Theorem of Generalized Logic Programs, HIER-LI-9213 (1992)