制約論理型言語を用いたプランニングの記述

福澤俊幸[1]　中島俊介[2]　長谷川晴朗[1]　長谷川隆三[3]
[1]沖電気工業(株)　[2]沖通信システム(株)　[3](財)新世代コンピュータ技術開発機構

1　はじめに

論理プログラミングの基本に,「アルゴリズム＝論理＋制御」として,論理だけをプログラマが記述すればよいという考え方がある.制約パラダイムは,この思想を強化するため,単一化の拡張という形で論理型言語に導入された[1].その特質を利用して,種々の問題の記述に応用されている.本稿では,制約論理型言語によるプランニングの記述実験について報告する.簡単な例題に対する Prolog と制約論理型言語のプログラムを示し,両者の比較から,制約論理型言語による記述の利点について述べる.

2　例題

次に示す簡単な例題を用いる.

> 3 つの部屋 a, b, c があり(図 1),1 台のロボットが置かれている.ロボットが実行できる動作は,隣の部屋へ移動することである.ロボットが部屋 A から部屋 C へ移動する時の動作系列を求めよ.

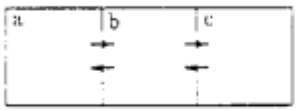

図 1: ロボットが置かれる部屋

この例題の探索木は,図 2 のような無限木になる.

項 at(R) はロボットが部屋 R にいる状態を表し,項 move(R1,R2) はロボットが部屋 R1 から部屋 R2 へ移動する動作を表している.

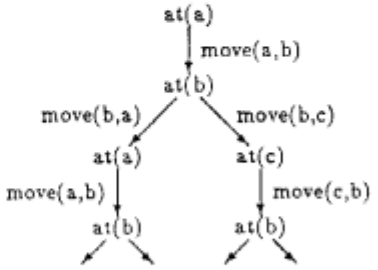

図 2. 例題の探索木

3　Prolog による記述

例題を解くプログラムは, Prolog を用いて以下のように記述される.

プログラム 1: Prolog による記述

```
robot([],S,S).
robot([A|As],S1,Sn) :-
    can(A,S1,S2),
    robot(As,S2,Sn).

can(move(a,b),at(a),at(b)).
can(move(b,a),at(b),at(a)).
can(move(b,c),at(b),at(c)).
can(move(c,b),at(c),at(b)).
```

述語 can/3 は,ロボットが実行可能な動作(第 1 引数)とその実行前後の状態(第 2, 3 引数)の関係を記述したものである.述語 robot/3 は, can/3 の関係を,ロボットが実行可能な動作系列(第 1 引数)とその実行前後の状態(第 2, 3 引数)の関係に拡張したものである.

ゴールは robot(As,at(a),at(c)) となるが,その実行は無限ループに陥ってしまう.この原因は,図 2 の木を縦型戦略で探索し,最も左の枝だけを無限にたどってしまうためである.このように,プログラム 1 では,解が存在するにもかかわらずその解を発見できない可能性があり,また,最初に発見された解が最短の動作系列である保証もない.これらの問題を解決するには,横型探索を行なえば良い.横型探索を行なうプログラムは, robot/3 の第 2 節において,ボディ部の述語の順序を逆にするだけで得られる.

以上から,次のことが考察される.節のボディ部の順序は論理的に無意味なものであるが, Prolog の実行結

A Description of Planning in Constraint Logic Language
Toshiyuki FUKUZAWA[1]　Shunsuke NAKAJIMA[2]
Haruo HASEGAWA[1]　Ryuzo HASEGAWA[3]
[1]Oki Electric Industry Co., Ltd.
[2]Oki Telecommunication Systems Co., Ltd.
[3]Institute for New Generation Computer Technology

果に影響を及ぼすことがある．このことはプランニングの記述の場合にもあてはまり，robot/3 の節中の述語の順序によって求まる解が異なる．従って，Prolog によるプランニングのプログラムは，制御に関する情報を意識して書く (読む) 必要がある．

4 制約論理型言語による記述

例題を解くプログラムは，制約論理型言語を用いて以下のように記述される．状態の成立や動作の生起をブール変数で表し，その間の関係をブール多項式で記述している．

プログラム 2: 制約論理型言語による記述

```
robot([],s(A,B,C),s(A,B,C)) :-
    state(A,B,C).
robot([a(AB,BA,BC,CB)|As],s(A1,B1,C1),S) :-
    state(A1,B1,C1),
    action(AB,BA,BC,CB),
    move(AB,A1,B2),
    move(BA,B1,A2),
    move(BC,B1,C2),
    move(CB,C1,B2),
    robot(As,s(A2,B2,C2),S).

state(A,B,C) :-
    (A∨B∨C)=1,
    (A∧B)=0,
    (A∧C)=0,
    (B∧C)=0.

action(AB,BA,BC,CB) :-
    (AB∨BA∨BC∨CB)=1,
    (AB∧BA)=0,
    (AB∧BC)=0,
    (AB∧CB)=0,
    (BA∧BC)=0,
    (BA∧CB)=0,
    (BC∧CB)=0.

move(XY,X1,Y2) :-
    (XY⊃X1)=1,
    (XY⊃Y2)=1.
```

項 s(A,B,C) 中の変数は，ロボットがある部屋にいる状態を表すブール変数で，項 a(AB,BA,BC,CB) 中の変数は，ロボットがある部屋から隣の部屋へ移動することを表すブール変数である．ちょうど，s(1,0,0) とプログラム 1 の at(a)，a(1,0,0,0) とプログラム 1 の move(a,b) のような対応関係にある．述語 state/3，action/4 および move/4 は，制約のみから定義され，複合的な制約を表す．さらに，これらの述語を用いて述語 robot/3 が定義される．state/3 は，常にある状態が成立するが，同時に複数の状態が成立しないという制約を表している．同様に，action/4 は，常にある動作を実行するが，同時に複数の動作を実行しないという制約を表している． move/3 は，動作とその実行前後の状態に関する制約を表している．robot/3 の第 2 節に現れる 4 つの move/3 が，プログラム 1 の 4 つの can/3 に対応している．robot/3 は，プログラム 1 と同様に，ロボットが実行可能な動作系列 (第 1 引数) とその実行前後の状態 (第 2, 3 引数) の関係を記述したものである．

次に，ゴール robot(As,s(1,0,0),s(0,0,1)) の実行において，どのように制約が解消されていくかを示す．制約解消系として，与えられたブール多項式からその標準簡約形を求めるアルゴリズム (例えば，文献 [2]) を用いることにする．解が求まるまでに，robot/3 は起動ゴールを含めて 3 回呼び出され，1, 2 回目は第 2 節が，3 回目は第 1 節が使用される．各呼び出しに対して，節実行後の制約解消結果を表 1 に示す．2 回目の実行が終了した時点で残った制約は，図 2 の探索木 の 2 段目において，2 通りの動作が可能であることに対応している．このように，制約解消によって横型探索が実現されている．

表 1: 制約解消のようす

	As の値	残りの制約
1	[a(1,0,0,0)|_]	⑦
2	[a(1,0,0,0), a(0,BA,BC,0)|_]	{(BA∨BC)=1, (BA∧BC)=0}
3	[a(1,0,0,0), a(0,0,1,0)]	⑦

プログラム 2 のすべての節について，ボディ部の述語 (制約) の順序をどのように変えても，実行の結果得られる解は同じものになる．また，制約解消系を実現するアルゴリズムを変えても，そのアルゴリズムが完全である限り，同じ解が求まる．つまり，プログラム 2 は制御に関する情報は含まず，論理のみが記述されていると考えることができる．

5 おわりに

状態と動作の関係をブール多項式で定義することで，制約論理型言語によるプランニングの記述を行なった．制御に関する情報が含まれていないことから，論理プログラミングの考え方に則した記述が得られた．なお，本研究は第 5 世代コンピュータ・プロジェクトにおいて，制約論理型言語による記述実験の一環として行なわれたものである．

参考文献

[1] Jaffar, J., Lassez, J-L.: Constraint Logic Programming, POPL-87, Munich, 1987.

[2] Sakai, K., Sato, Y.: Boolean Gröbner Bases, TM-488, ICOT, 1988.