拡張 Prolog による群論電卓の作成*

2P−1

佐藤尚　近藤邦雄　島田静雄†
埼玉大学工学部情報工学科‡

1 はじめに

飯高茂は１９８０年代後半より、Prolog の持つバックトラッキングとパターンマッチの機能を利用して、数学の世界を組み立てることを行ってきた[1]。その中で群論電卓の作成が大きなテーマとして取り上げられている。群論計算システムとして、CAYLE などの大規模なシステムがよく知られている[2]。飯高の群論電卓は、これらのシステムとは異なり、群論電卓の作成を通して、群論を理解することに力点がある。このような小規模なシステムでも新しい定理を発見するための具体例の計算にも使えることができる。数学では、「ある構造をもつ数学的対象の間に、その構造を保つ写像を考えつつ議論を進めるのが基本である」と言われている。飯高の群論電卓では、同時に複数の群を扱うことが出来ないので、この基本に沿った計算を実行することが出来ない。本報告では、Prolog に オブジェクト指向風の拡張を施し、それを利用して複数の群を扱うことの出来る群論電卓の概要を報告する。

2 飯高の群論電卓

飯高の群論電卓は次のような方法で作られている。

1. 各元に番号付けを行い、述語の形で記憶する。
2. それをもとに逆元表を作り、述語の形で記憶する。
3. 各元の番号対応述語と逆元対応述語をもとに計算を行う。

このような方法で作成すると、通常 Prolog では局所的に有効な述語が定義出来ないために、群の計算に利用される各種の対応表を大域的な述語に記録する。このため、複数の群を同時に扱うことが困難となる。飯高の群論電卓では、述語の定義が一部だけ異なる述語が数多く存在する。これは、個々の群の性質から共有可能な部分があるにも関わらず、通常の Prolog では、それを利用した記述が出来ないためである。これらの欠点を補うために Prolog にオブジェクト指向風の拡張を施し、それを利用して飯高の群論電卓を拡張することにした。

*Group Theoretical Calculator Using Extended Prolog
†Hisashi SATO,Kunio KONDO,Shizuo SHIMADA
‡Saitama University

3 拡張 Prolog の概略

ここで示す拡張 Prolog は、従来の Prolog に対して上位互換性を持つようにし、従来の使用者が容易に利用できることが必要である。そのために、なるべく簡単な拡張で済むようにした。

3.1 インスタンス

この拡張 Prolog では述語の集まりを instance と呼ぶ。Smalltalk-80 などとは異なり、クラスは存在せず、インスタンスのみで、すべてのオブジェクトが構成される。インスタンスは、特定のインスタンスをプロトタイプとして持ち、プロトタイプのインスタンスの述語を継承する。インスタンスとプロトタイプインスタンスの間には親子関係があると考える。インスタンスは、自分のインスタンスを調べ、目的の述語を持っているかを調べる。もし見つかればその述語を実行し、そうでなければ、親インスタンスに検索に行く。この拡張 Prolog では、述語は子孫のインスタンスで上書きして変更することが出来る。 先祖インスタンスと子孫インスタンスの述語を組み合わせて、一つの述語として利用することは出来ない。インスタンスの生成には、 make_instance/2 という述語を用いる。

3.2 述語の呼び出し

述語の呼び出しには、次のような構文をとり、インスタンス名を指定して呼び出す。

インスタンス名 :: 述語名

インスタンス名を指定しない暗黙の述語呼び出しの場合は、通常の Prolog の述語が呼び出される。疑似変数として、self,this,super,prolog の四つを用意する。自分自身は this,self、親インスタンスは super を表す。インスタンス名に prolog を指定すると、通常の Prolog の述語が呼び出される。次のような二つのイン

スタンスを作る。インスタンス natural はインスタンス good の親インスタンスとする。

親インスタンス (natural)

```
kind(natural).

character(C) :-
        self::kind(K),
        this::allowed(K,C).

allowed(_,fighter).
allowed(natural,thief).
```

子インスタンス (good)

```
kind(good).

character(lord).
character(C) :-
        super::character(C).
```

このとき、natural::character(A) と good::character(A) は次のようになる。

```
?- natural::character(A).
A = fighter;
A = thief;
no
?- good::character(A).
A = lord;
A = fighter;
no
```

4 拡張 Prolog を用いた群論電卓

この拡張 Prolog を利用して作成した群論電卓は、前章で示した方法を利用して作られている。群を司るインスタンスを作成した。このインスタンスは演算表から群の元同士の計算を行う述語と演算表を作るのに必要となる述語を持っている。この他に群の性質などを調べる述語を定義しておく。

```
X is E*X :- identity(E),!.
X is X*E :- identity(E),!.
Z is X*Y :-
        self::group_op_data(X,Zs),!,
        nthelem(Zs,Y,Z).
```

群の各元を生成する述語と具体的な元同士の演算を行う述語を持つインスタンスを、親インスタンスとして持つことで群論電卓が完成する。次のような述語により、写像 Map が群 G1 から群 G2 への準同型写像かどうかを調べることが出来る。

```
is_homo(G1,G2,Map) :-
        G1::for_each(X1),
        G1::for_each(Y1),
        G1::(Z1 is X1*Y1),
        map(X1,X2,Map),
        map(Y1,Y2,Map),
        map(Z1,Z2,Map),
        (G2::(Z2 is X2*Y2)->fail;true),
        !,
        fail.
is_homo(_,_,_) :- !.
```

5 まとめ

本報告では、拡張 Prolog を用いた群論電卓の概要を示した。この拡張 Prolog は５００行程度の Prolog で作られており、各種の Prolog 上で動作している。これにより複数の群を同時に扱うことの出来る群論電卓を作成することが出来た。今後の予定しては、多くの群を扱えるようにインスタンスを作成すること、群論電卓の機能を拡張すること、拡張 Prolog の実行速度を向上させることを考えている。

謝辞

日頃から熱心に指導して頂いている学習院大学の飯高茂教授に感謝します。

参考文献

[1] 飯高茂 Prolog で作る数学の世界 朝倉書店 (1990)

[2] M.Atokinso ed. Computational Group Theory Academic Press (1984)