コンピュータと教育 15－3
(1991. 4. 26)

Prologプログラミングの学習を支援するシステム

府川 和生　伊丹 誠　伊藤 紘二

東京理科大学

あらまし　本稿では，水準の異なる学習者が，Prologによるプログラミングを独習するための事例ベースの学習支援システムの構築について報告する．

今回実現したシステムは，その基本となる部分である．このシステムでは，Prologについての一応の知識は得ていることを前提として，プログラミング技術を独習させるためにプログラム例を用いる．各プログラム例についてデータ型・プログラム構造・プログラム名（述語名）という3つの視点を設け，手掛かり表現を用いて自由に検索し，実行・比較させることで学習者のプログラミング技術を高めるのが目的である．

A Computer Assistant for Learning Prolog Programming

KAZUO FUKAWA　MAKOTO ITAMI　KOHJI ITOH

Department of Applied Electronics, Science University of Tokyo

Yamazaki 2641, Noda city, CHIBA, 278 JAPAN

Abstract　The report conserns with a project of developing a case-based computer assistant for helping students with different levels to learn programming in Prolog by themselves.

It describes realization of a basic part of the project. The system, presupposing on the part of the student exposure to Prolog primer, makes use of program samples.

1.はじめに

Prologの一応の知識は得ていることを前提とした，水準の異なる学習者がPrologプログラミングを独習するためのシステムの構築について報告する．

プログラミングの学習支援を目的に開発されたシステムとして，PascalではBonarのBRIDGEやPROUST[1]，Prologでは日立で開発されたシステム[2]などがある．BRIDGEは初心者プログラマが”自然言語での問題記述から詳細な実行可能なプログラムまでの段階的な詳細化過程”を辿るのを誘導するシステムである．我々は，これらの段階に相当するものとしてデータ型・プログラム構造・プログラムの三つの視点を考え，これを用いたPrologプログラミング学習を支援するシステムを提案する．

このシステムでは，Prologによるプログラミング技術を独習させるために，プログラム例を用いる．

機械学習の研究によれば，問題解決の方略を学習するためには，複数の事例を知識によって説明し，その説明の構造の共通点を方略として抽出するという方法が，極めて有効であることが知られている．我々の目標は，このような事例ベースの学習支援システムを構築することにある．本稿は，そのための基礎となる一部分を実現した結果の報告である．

2.構想

学習の方法は，プログラム例をデータ型・プログラム構造・述語名から自由に検索・実行・比較させるものである．

学習者はまず，データ型による検索を行いプログラム例を選択する．プログラム例が選択されると，プログラム例の学習に移る．さらに，検索されたプログラム例について，同じデータ型・プログラム構造・述語名のプログラム例を検索・学習することができる（図1参照）．

システムに検索における分岐点でのユーザの選択を記録したヒストリを持たせ，ユーザが，いつでもどの分岐点にも戻って，別の選択を試みることができるようにする．

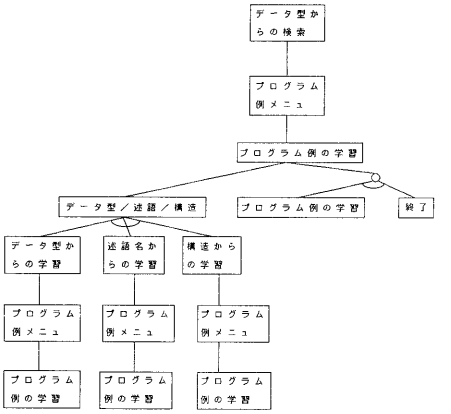

図1　システム構想図

2.1　データ型による検索・学習

システムは，各プログラム例について｛与えるデータ型，結果のデータ型，与えるものと結果の関係｝を知識として所有する．与えるデータ型としては｛列，集合，アトム，数，etc.｝，結果のデータ型としては｛列，数，論理解Yes/No，etc.｝，与えるものと結果の関係としては｛順序づけ，置換，反転，etc.｝があげられる．

学習者が，与えるデータ型，結果のデータ型，与えるものと結果の関係，ないし，それらの一部を入力すると，システムは，条件に該当するプログラム例を検索しプログラム例メニュを表示する．学習者はプログラム例メニュから学習するプログラム例を選択し，プログラム例の学習を行う．

例えば，リストとリストの連結を行うappendでは，

与えるデータ型：　　リスト2つ，
結果のデータ型：　　リスト，
与えるものと結果の関係：結合

ということに，また，
要素がある集合に含まれるかを判断するmemberでは，

与えるデータ型：　　リストあるいは集合と
　　要素になりうるデータ型，
結果のデータ型：　　論理解Yes/No，
与えるものと結果の関係：判定

となる．

2.2 プログラム構造による検索・学習

各プログラム例の構造を{直接型，前向き再帰，後ろ向き再帰，条件分岐，etc.}などに分類しこれらをキーワードとして，あるいはさらに細かい構造の部分構造をキーワードとして，プログラム例を検索する．

ここでは，前向き再帰と後ろ向き再帰について具体的な例を紹介する．

```
プログラム構造：後ろ向き再帰
ゴール（Given,Desired,Predicate)
→縮小（Given,Given1,Reduce），
  ゴール（Given1,Desired1,Predicate），
  更新（Given,Desired1,Desired,Step）．
```

Givenは入力で与えるもの，Desiredは結果として得られるもの，Predicateは述語名を表す．縮小でGivenをもとにGiven1が作られ，続いてゴールでGiven1を入力としてPredicateを再帰呼び出しする．更新ではGiven,Desired1をもとにStep（手続き）を行いDesiredを得る．

これをappendを用いたreverseに適用すると次のようになる．

```
1)% reverse(Xs,Ys).
2)  reverse([],[]).
3)  reverse([X|Xs],Zs):-
        reverse(Xs,Ys),
        append(Ys,[X],Zs).

3)ゴール（[X|Xs],Zs,reverse)
  →縮小（[X|Xs],Xs,tailの取り出し）
    ゴール（Xs,Ys,reverse)
    更新（[X],Ys,Zs,リストの結合）
1)ゴール（Xs,Ys,reverse)
  →条件分岐（Xs,Yes/No,Xs=[])
  Yes:2)
  No: 3)
2)ゴール（[],[],reverse)

プログラム構造：前向き再帰
ゴール（Given,Desired,Relation)
→ゴール（[Given,[]],Desired,Relation1）．
ゴール（[Given,Interium],Desired,Relation1)
→与件の簡単化（Given,Given1,Reduce），
  中間結果の更新（[Given,Interium],
                  Interium1,Step），
  ゴール（[Given1,Interium1],
          Desired,Relation1）．
ゴール（[a,Interium],Interium,Relation1）．
```

まずは，累算器を用いたゴールを作り，与件の簡単化では[Given,Interium]をもとにGiven1を作る．中間結果の更新は，[Given,Interium]を用いてInterium1を作り，続くゴールで[Given1,Interium1]をもとに再帰呼び出しを行う．最後のゴールでは，[a,Interium]のaが終了条件を示し，InteriumをDesiredにコピーしている．

累算器を用いた前向き再帰型のreverseに適用すると以下のようになる．

```
1) reverse(Xs,Ys):-
       reverse(Xs,[],Ys).
2) %reverse(Xs,Acc,Ys).
3) reverse([X|Xs],Acc,Ys):-
       reverse(Xs,[X|Acc],Ys).
4) reverse([],Ys,Ys).

1)ゴール（Xs,Ys,reverse2)
  →ゴール（[Xs,[]],Ys,reverse3)
2)ゴール（Xs,Ys,reverse3)
  →条件分岐（Xs,Yes/No,Xs=[])
    Yes: 4)
    No:  3)
4)ゴール（[[],Ys],Ysreverse3)
3)ゴール（[[X|Xs],Acc],Ys,reverse3)
  →与件の縮小（[X|Xs],Xs,tailの取り出し）
    中間結果の更新（[X,Acc],X|Acc,
                    リストの生成）
    ゴール（[Xs,X|Acc],Ys,reverse3)
```

2.3 述語名による検索・学習

プログラム例の中には，appendを定義するのに単なる後ろ向き再帰を用いる方法と累算器を用いた前向き再帰で定義する方法があるように，同じ述語名を違ったアルゴリズム・プログラム構造で定義しているものがある．また，プログラム例の中で別のプログラムが呼び出されていることも多い．これらのプログラム例を比較・検討することはプログラミング技術を学習するうえで大切なことと考えられる．このシステムでは，選択されたプログラム例と同じ述語名を定義しているプログラム例を検索し，それらを比較することで学習を進める．

2.4 実行による学習

学習対象となったプログラム例を実行させてみることが大切である．また単に実行するだけでなく，Prologインタプリタを作成して，ユーザフレンドリなトレースなどの学習支援機能を実現する必要がある．

2.5 検索のための手掛かり表現について

ユーザの検索要求に見合ったプログラム例を検索するためには，データ型あるいはプログラム構造による検索要求を，その組み合わせによって表現することができるような基本的概念表現のセットを用意し，プログラム例の各々について，これら基本概念表現の組み合わせを用いた手掛かり表現を置く．一方，ユーザによる検索要求も，同じ基本概念表現の組み合わせによって記述させる．このようにすれば，検索要求に少なくとも部分的にマッチした手掛かり表現を検索することにより，要求に見合ったプログラム例の候補を見い出すことができる．

2.6 ヒューマンインタフェース

実行による学習におけるトレース表示は，プログラム構造と結び付けた形の表示によって行うのが望ましい．

検索要求の入力に対しては，グラフィカルな比喩表現を利用するのがよいだろう．

プログラム例におけるプログラム構造を提示する方法としては2.2に掲げた形が考えられるが，今一つ工夫を要する．

3.実現

プログラム例は「The Art of Prolog」[3]から採用した．システム自体はC言語を用いて開発しているが，検索部分はPrologを用いている．図2は，昨年開発したシステムの流れ図である．

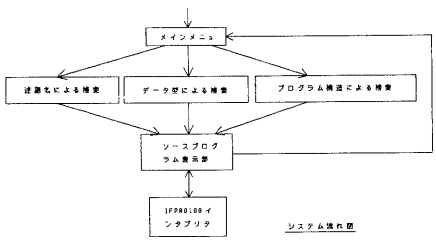

図2　システム構成図

3.1 データ型の手掛かり表現

データ型による検索のための手掛かり表現について，appendを例に挙げて述べてみる．

```
% append(Xs,Ys,XsYs):-
% リストXs,Ysを結合したものがリストXsYs

append([],Ys1,Ys1).
append([X|X1],Ys2,[X|Zs]):-
    append(Xs1,Ys1,Zs).
```

上述したappendのデータ型による検索のための手掛かり表現は以下のようになる．

```
1)  index(p,from,
        [list,Xs],[Xs,'Xs'],append1).
2)  index(p,from,
        [list,Ys],[Ys,'Ys'],append1).
3)  index(p,to,
        [list,XsYs],[XsYs,'XsYs'],append1).
4)  index(p,procedure,
        [ketugo,APPEND,Xs,Ys,XsYs],
        [[APPEND,'append'],[Xs,'Xs'],
          [Ys,'Ys'],[XsYs,'XsYs']],
        append1).
```

この例では，４つの手がかりが挙げられる．第一引き数はデータ型のための手掛かりであることを，第二引き数は｛与えるデータ型，結果のデータ型，与えるものと結果の関係｝のいづれかを示す．第三引き数は手掛かりで，その対象となるものを変数で表している．第四引き数はその対象となるものの対応を示している．第五引き数がプログラム例のIdである．

各行の意味は次のようになる．

1) append1というプログラム例を実行する場合に与えるデータ型はリストXsである．
2) 与えるデータ型はもう一つあってリストYsである．
3) 結果として得られるデータ型はリストXsYsである．
4) 与えるものと結果の関係はappendである．

3.2 プログラム構造の手掛かり表現

同じプログラム例のプログラム構造のための手掛かり表現は以下の通りである．

```
1) index(struc,category,CAT,
     [CAT,
      ushiromukisaiki_hibunnki_syuturyoku],
     append1).
2) index(struc,given,
     GIVEN,[GIVEN,['Xs','Ys']],
     append1).
3) index(struc,procedure,
     PRED,[PRED,append],
     append1).
4) index(struc,given1,
     GIVEN1,[GIVEN1,['Xs1','Ys']],
     append1).
5) index(struc,reduce,
     RED,[RED,list_tail],
       append1).
6) index(struc,desired1,
     DESIRED1,[DESIRED1,'Zs'],
     append1).
7) index(struc,step,
     STEP,[STEP,head_tail_list],
     append1).
```

それぞれの意味は，次のようなものである．

1) プログラム例append1のプログラム構造は後ろ向き再帰非分岐出力型である．
2) 与えるデータはXsとYsである．
3) 結果として得られるのはXsYsである．
4) 再帰する際に与えるGIVEN1はXs1とYsである．
5) 縮小で行われるのはlist_tailと呼ばれる操作である．
6) 再帰を行って得られるデータがZsである．
7) 更新で行われる操作はhead_tail_listと呼ばれるものである．

4. 今後の課題

システムは現在開発中であるので，構想で述べたことと重複するが，今後の課題としては，

・ウィンドウシステムを用いたグラフィカルユーザインタフェースの実現
・トレースやヒストリ機能の開発

などが挙げられる．

また，データ型における与えるものと結果の関係からプログラム構造を決定していくための知識ベースに関する研究が本質的な課題の一つである．

参考文献

[1]Etienne Wenger,"ARTIFICIAL INTELLIGENCE and TUTORING SYSTEMS",Chapter 11, Morgan Kaufmann

[2]Haga, H. and Masui, S.,Program Information Management Method for and Intelligent Programming Support Enviornment,Proceedings of InfoJapan'90

[3]Leon Sterling and Ehud Shapiro:The art of Prolog