知識工学と人工知能 38－2
（１９８５．１．１８）

帰納推論に基づく知的ＣＡＩシステム
――ＰＲＯＬＯＧ教育の場合――

顧化真志† 河合和久† 溝口理一郎†† 喜納久行††† 角所 収†† 豊田順一††
†大阪大学基礎工学部 ††大阪大学産業科学研究所 †††大阪大学工学部

[1] はじめに

最近、マイクロコンピュータが、価格の低廉化に伴い、学校、会社、家庭にと広く普及するようになった。また、これらのコンピュータを教育、あるいはシミュレーション訓練等に利用し、個別教育を行おうとする動きが、各所で見られるようになった。しかしそうしたシステムは簡易なものが多く、その教育効果はあまり期待できない。このような背景から、教育効果が十分期待でき、扱いやすいシステムの開発を切望する声が高まっている。

コンピュータの教育への適用に関する研究は1960年代にその端を発する。初期のシステムは、あらかじめシステムに用意された答や、診断コメントを用いて、生徒の解答に応答するドリル練習のモニタの働きをするものであった。これらはＣＡＩ（Computer-Assisted Instruction）システムと呼ばれる。1970年代に入り、教材と教育手続きとが別個に独立して表現され、個々の学習者に対して、異なった問題の提示、診断コメントが出来るシステムが作成されるようになった。これらは初期のＣＡＩシステムと区別する上で、知的ＣＡＩシステム[1]と呼ばれる。

知的ＣＡＩシステムには、(1) 専門知識モジュール、(2) 学生モデルモジュール、(3) 個人指導モジュールの三つの基本モジュールが必要であると考えられる。このうち、学生モデルモジュールは個人指導モジュールによって学生に提示された教材を学生がどの程度理解したかを示すモジュールである。従って、この学生モデルが実際の学生の理解度をどの程度正確に把握しているかによって、知的ＣＡＩシステム全体の教育効果が左右されることになる。

しかし、既存の多くの知的ＣＡＩシステムでは、システムがあらかじめ用意している専門知識の部分集合として学生モデルを表現していた。この方法では'学生の誤り'の原因が

１）その知識の欠如のよる誤り

２）その知識について誤った知識を持っているのどちらであるか区別出来ないという欠点を持つ。

一方我々は、学生に与えた質問などから得られた学生の応答をもとに、帰納推論を用いて学生モデルを生成する新しい知的ＣＡＩシステムを提案している[5)6)]。この方法では学生モデルを'無'から生成することにより、学生の犯す誤りすべてに対処しようというもので、これまでのシステムにない強力で効果的な教育が期待できる。

本論文では、学生モデルモジュールに帰納推論を導入した知的ＣＡＩシステムの枠組みを説明し、その応用事例として現在開発中のPROLOGプログラミング教育システムについて述べる。なお、本システムはPROLOGでインプリメントされている。

[2] 知的ＣＡＩシステムの構成

一般に、知的ＣＡＩシステムはその構成上、次の三つのモジュールから成る。

１）専門知識モジュール
学生に教授する分野に関する完全な知識を持ち、学生の応答の正当性の評価等を行うモジュールである。いわゆる教科書に相当する。

２）学生モデルモジュール
各学生について、何をどこまで理解したか、またどの部分がわかっていないかを表すモデルを構成し、そのモデルを蓄積するモジュールである。

３）個人指導モジュール
学生に、次にどういった教材や質問を与えれば適切かを判断するモジュールである。

この三つのモジュールの関係は図１のようになる。

以上述べたように、一般的な知的ＣＡＩシステムは三つの基本モジュールから成る。我々はこのうち特に学生モデルモジュールが重要であると考える。なぜなら学生モデルモジュールが学生の理解度を十分把握することによってはじめて個人指導モジュールがうまく働き、システム全体の教育効果が上がると考えられるからである。

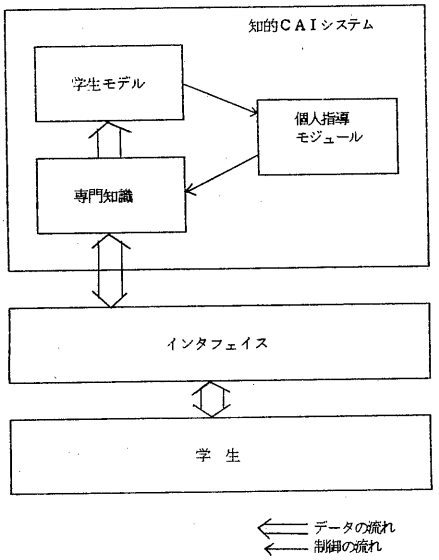

図 1　知的CAIシステムの構成

次に、既存の知的ＣＡＩシステムが学生モデルをいかに表現しているかについて述べる。これらは、大きく分けて次の二つの表現法を取っている。

1）専門知識モジュールの部分集合として表現する。

2）専門知識モジュールの混乱状態、または脱線状態として表現する。

これらの手法では、’学生の誤り’の原因が

a）その知識の欠如による誤り

b）その知識について誤った知識を持つ

のどちらであるかを決定できない。

[3] システム設計

上記のように、従来の知的ＣＡＩシステムの学生モデルモジュールは、教育効果の観点からすると、十分であるとは言えない。その原因は、学生モデルを’有’（システム側があらかじめ用意している専門知識）の状態から生成しようとする点にある。

我々は学生モデルを’無’から作り上げることにし、学生モデル生成過程に帰納推論を導入する。

[3.1] 帰納推論

帰納推論に関する研究としては、Vere[2]，Michalski [3]，Shapiro[4] 等がある。本システムでは Shapiroの提案したＭＩＳ（Model Inference System）を学生モデル生成過程に、ＰＤＳ（Program Diagnosis System）を学生の誤りの検出過程に導入する。この節では、ＭＩＳ，ＰＤＳについて簡単に説明する。

ＰＤＳはPROLOGプログラムの虫探しを行う対話型アルゴリズムである。図２はＰＤＳの働きを示す模式図である。

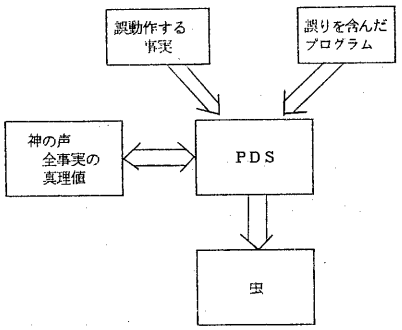

図 2 PDS模式図

虫を含むプログラムと、そのプログラムで実際に誤った答が返されるような事実が与えられると、ＰＤＳはその事実をプログラムを用いて１ステップずつ実行しながら、その実行過程が正しいかどうかを神の声に尋ねる。神の声とは、全事実の真理値を知っているもので、一般にそのプログラムのプログラマである。従って神の声への質問は、プログラマとの対話によって実現される。この神の声への質問を繰り返し、誤って実行される根本的な原因である箇所を発見するのがＰＤＳの働きである。

ＭＩＳは与えられた事実列の集合をすべて実行するようなプログラムを生成するものである。図３はＭＩＳの働きを示す模式図である。ＭＩＳはプログラム生成過程でＰＤＳと精密化オペレータを用いる。

ＭＩＳは、まず一番目の事実を読み込む。この事実を実行するためには、適当な節が必要になる。これを生成するのが精密化オペレータである。精密化オペレータでは、節の生成に際して、1)節の両辺に用いられる述語の組、2)節中の各ゴールの各引数が入力モードか、出力モードか、及びその構造（リストか、定数、変数か）、3)その節の右辺の各サブゴール（事実）の真理値、を神の声から得る。こうして一番目の事実を実行する節が生成され、その節を目的プログラムとする。次に二番目の事実を読み

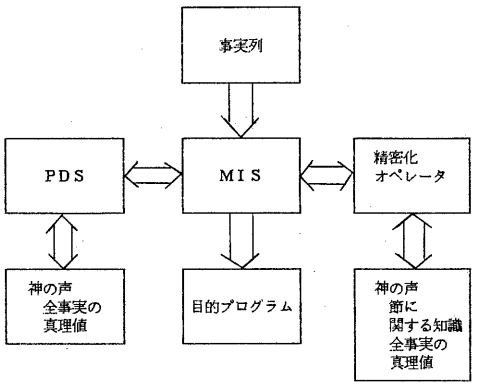

図 3 MIS模式図

込むと、この事実をプログラムで実行してみる。正しく実行されれば、三番目の事実に進むが、正しく実行されない場合は、その事実とプログラムに対してＰＤＳを用いてプログラムの虫を探す。そしてその虫である節を削除し、新しい節を精密化オペレータが生成し、それを加えて新しいプログラムとして、一番目の事実からもう一度繰り返す。これによって、すべての事実が読み込まれた時、そのすべての事実を正しく実行するプログラムが出来上がる。

[3.2] 帰納推論の導入

本システムでは、学生モデル生成過程に帰納推論を導入する。その生成過程は次の二つのフェーズから成る。

1）モデルの生成
学生の応答や質問から、学生の理解のモデルを生成する

2）モデル中の誤りの検出
1）で作られた学生モデルと専門知識とを比較し、学生モデルに含まれる学生の誤りを検出する

なお、この生成過程において、一つの前提を仮定しなければならない。それは、フェーズ1）で学生モデルを生成している間は、学生の頭の中の状態は変化しないということである。この仮定がなければ、フェーズ1）によって学生モデルが生成されても、その生成されたモデルが実際の学生の頭の中の状態と一致せず、フェーズ2）によって検出される学生の誤りも、実際に学生が持つ誤りであるとは言えなくなる。

フェーズ1）にＭＩＳを、フェーズ2）にＰＤＳを用いたシステムの構成図を図4に示す。

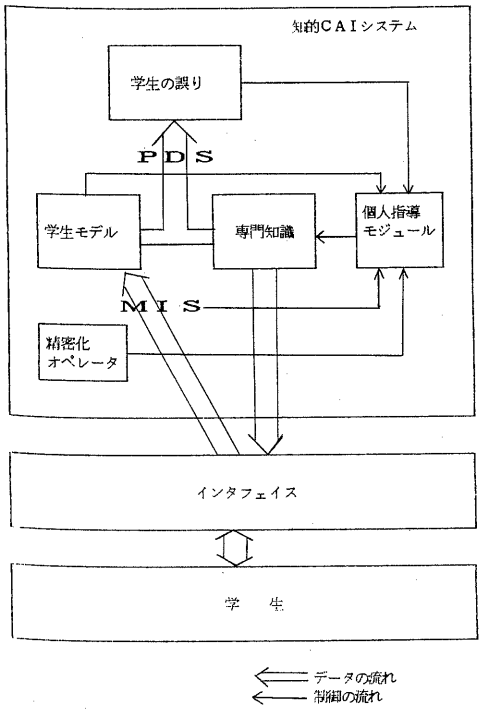

図 4 MIS,PDSを用いた知的CAIシステム

[3.3] モデル記述言語PROLOG

学生モデルの記述言語には、PROLOGを用いる。PROLOGを選んだ理由は、1)帰納推論と述語論理との整合性がよい、2)PROLOGの記述力は、学生モデルの記述に十分である、3)メタレベルの取り扱いが可能、の三点である。

モデルをPROLOGで記述することによって上記の二つのフェーズは、それぞれ次のように言い換えられる。

1）学生の応答や質問を表すいくつかの事実例等から、それらを実行（証明）するPROLOGプログラムを生成する。さらにメタレベルの知識を表すPROLOGプログラムを生成する。

2）学生モデルと同様、PROLOGプログラムで記述された専門知識を正しいPROLOGプログラムとみた場合、学生モデルは誤りを含むんだPROLOGプログラムとみることができる。従って、学生モデルの誤りの検出は、誤りを含んだPROLOGプログラムの虫探しに相当する。

[3.4] メタ知識の導入

人間が用いている知識を分析すると、次の２種類の知識に分けられる。

１）　基本的な事柄を解く知識

２）　１）の知識をいかにうまく用いて、より大きな事柄を解くかという知識

ここでは１）を下位レベルの知識、２）をメタレベルの知識と呼ぶ。

従って、これら二つの知識をモデル化できないシステムでは、十分な教育効果は期待できない。我々は、これら二つの知識をモデル化するため、これらの知識をPROLOGで以下のように定義する。

・下位レベルの知識

a:-b，c，d．　で

b，c，dはシステム側があらかじめ用意しているプリミティブな知識である。

・メタレベルの知識

a:-b，c，d．　で

b，c，dのうち少なくとも一つは他の知識表現のヘッド部に現れる。つまり、

b:-b1，b2．

のような知識表現がどこかに存在する。

但し、a，b，b1，b2，c，dはPROLOGの述語を表す。

例えば、PROLOGプログラミング教育では、

・下位レベルの知識は、

アトムはどう表されるか。

組み込み述語readはどういう働きをするか。

・メタレベルの知識は、

組み込み述語readを使って、ファイルの内容を読み込むプログラムを作るにはどうしたらよいか。

となる。

[3.5] 各モジュールの設計

この節では、図４で示したシステム構成図の各モジュールについて詳細に説明する。

１）学生の誤り

本システムでは学生の誤りとして、以下の３種類のものを扱う。

(1) 知識の欠如

教材に関する知識がない。例えば、日本の首都を知らない。

(2) 誤った知識

教材に関する知識の記憶違い。例えば、日本の首都は大阪であるという知識を持つ。

(3) 知識の適用の誤り

正しい知識を適用せずに、別の知識を適用してしまった。

前節の議論より、(1)、(2) は下位レベルの知識に関する誤り、(3) はメタレベルの知識に関する誤りとみなすことができる。

図５に誤りの種類をイメージ化したものを示す。

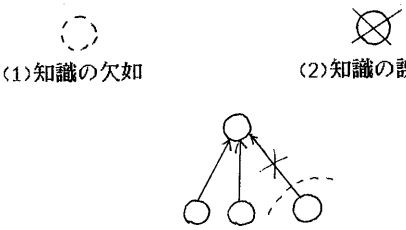

図５学生の誤りの種類

２）個人指導モジュール

システム全体の動きを制御する。システムの動きをその性質上、次の三つのモードに分ける。

・mis モード

ＭＩＳを用いて学生モデルを生成する。以下にその手順を示す。

(1) 専門知識モジュールから適切な教材を取り出し、学生に示し、問題を与える。

(2) 学生の応答からＭＩＳを用いて学生モデルを生成する。

・pds モード

学生モデルと専門知識とから、ＰＤＳを用いて学生の誤りを検出し、それに則した教育を行う。以下にその手順を示す。

(1) 学生モデルと専門知識とから、ＰＤＳを用いて学生モデルの誤った箇所を検出する。

(2) 誤りの種類を同定する。

(3) 誤った箇所を学生に知らせる。

(4) 誤りの種類に応じて適切な教育を行う。

(5) 誤った知識を学生モデルから削除する。

・edu(cation) モード

復習問題を学生に与え、その応答から学生モデ

ルを作り直す。なお、一連のpds モードで学生の誤りが発見されない場合は、ここでは何もしない。

図６に各モード間の状態遷移を示す。まずmis モードで学生モデルを次々に生成していく。適当な時期にpds モードに移り、そこで学生の誤りを検出する。学生の誤りが検出されなければ、edu モードではなにもせず、次のステップに移る。誤りが検出されれば、pds モードで、それに対する教育が行われる。さらにedu モードで、復習問題を学生に解かせ、新たに学生モデルを生成し、再びpds モードに移り、以下同じ処理を繰り返す。すなわち、そのステップでの学生の誤りがなくならない限り、次のステップには移らない。

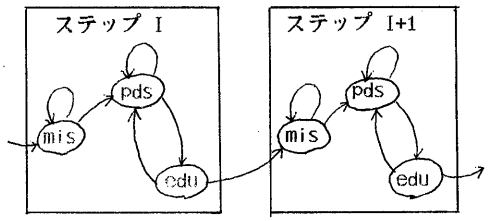

図 6 各モード間の状態遷移

3）専門知識モジュール

PROLOGプログラミングに関する知識はPROLOGで記述されている。図７にシンタクスに関する知識、及びユニフィケーションに関する知識を示す。

PROLOGプログラミングの知識を記述するにあたって、ルールのヘッド部の引数の数を５とした。以下に各引数の機能を述べる。

- 第１、第２引数
  対象を表すd-listである。
- 第３引数
  項目分類用のインデックスである。
- 第４引数
  pds モードで用いられ、pds モード終了後、その知識が正しいと判断された場合は ok の値を取る。
- 第５引数
  知識の階層性を表す。

図７(b) で示したユニフィケーションの知識の階層性を示したのが、図８である。

```
small_letter([X|Xs],Xs,[a,1,1],_,_):-
   member(X,[a,b,c,d,e,f,g,h,i,j,k,l,m,n,
         o,p,q,r,s,t,u,v,w,x,y,z]).
capital_letter([X|Xs],Xs,[a,1,2],_,_):-
   member(X,['A','B','C','D','E','F','G','H','I','J','K','L',
    'M','N','O','P','Q','R','S','T','U','V','W','X','Y','Z']).
digit([X|Xs],Xs,[a,1,3],_,_):-
   member(X,[0,1,2,3,4,5,6,7,8,9]).
built_in([X|Xs],Xs,[a,1,4],_,_):-
        builtin(X).
symbol([X|Xs],Xs,[a,1,5],_,_):-
   member(X,['+','-','*','/','<','>','¥','@','#',
        '$','^','&','~',':',',','?','.','''']).
number([X|Xs],Xs,[a,1,6],_,_):-
        integer(X).
char(X,Y,[a,2,1],_,_):-
        small_letter(X,Y,[a,1,1],_,_).
char(X,Y,[a,2,2],_,_):-
        digit(X,Y,[a,1,3],_,_).
char_list(X,Y,[a,2,3],_,_):-
        char(X,Y,[a,2,_],_,_).
char_list(X,Z,[a,2,4],_,_):-
        char(X,['_'|Y],[a,2,_],_,_),
        char_list(Y,Z,[a,2,_],_,_).
char_list(X,Z,[a,2,5],_,_):-
        char(X,Y,[a,2,_],_,_),
        char_list(Y,Z,[a,2,_],_,_).
symbol_list(X,Y,[a,2,6],_,_):-
        symbol(X,Y,[a,1,5],_,_).
symbol_list(X,Z,[a,2,7],_,_):-
        symbol(X,Y,[a,1,5],_,_),
        symbol_list(Y,Z,[a,2,_],_,_).
                .
                .
                .
```

(a) シンタクスに関する知識（一部）

```
single_assignment(X,Z,[c,1,1],_,unification):-
        assign(X,Y,variable,value),
        not_rewrite(Y,Z,value).
substitute_list(X,U,[c,1,2],_,unification):-
        variable(X,Y,v),value(Y,Z,u),
        form(Z,U,'[v/u]').
unify(X,W,[c,1,3],_,unification):-
        obtain(X,Y,value_1),from(Y,Z,substitute_list),
        obtain(Z,U,value_2),from(U,V,substitute_list),
        unify_exe(V,W,[c,1,_],_,unify).
unify_exe(X,V,[c,1,4],_,unify):-
        variable(X,Y,value_1),
        variable(Y,Z,value_2),
                .
                .
                .
```

(b) ユニフィケーションに関する知識(一部)

図 7 PROLOGによる専門知識の記述

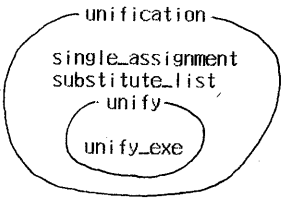

図 8 知識の階層性(ユニフィケーション)

4）学生モデル

学生モデルもPROLOGで記述されている。学生モデルの内容は、学生がシステム起動時に入力した名前（次章参照）のファイルに格納されている。

図９に学生モデルの内容を示す。これは次章の図10で示す教育が行われた後の学生モデルの内容を示したものである。図中の番号に従い、以下に

説明を加える。

(1) 学生の進度を示すもので、専門知識モジュールへのポインタである。

(2) 図10における教育で実際生成された学生モデルである。

(3) 帰納推論用の制御情報である。

```
pointer(5).                                          (1)
char(X,Y,Z,yet,U):-
   small_letter(X,Y,Y1,Z1,U1).
char(X,Y,Z,yet,U):-
   digit(X,Y,Y1,Z1,U1).
symbol_list(X,Y,Z,yet,U):-
   symbol(X,Y,Y1,Z1,U1).
symbol_list(X,Y,Z,yet,U):-
   symbol(X,X1,Y1,Z1,U1),
   symbol_list(X1,Y,Y2,Z2,U2).
char_list(X,Y,Z,yet,U):-
   char(X,Y,Y1,Z1,U1).
char_list(X,Y,Z,yet,U):-
   char(X,['_'|X1],Y1,Z1,U1),
   char_list(X1,Y,Y2,Z2,U2).                         (2)
char_list(X,Y,Z,yet,U):-
   char(X,X1,Y1,Z1,U1),
   char_list(X1,Y,Y2,Z2,U2).
variable(X,Y,Z,ok,U):-
   capital_letter(X,X1,Y1,Z1,U1),
   char_list(X1,Y,Y2,Z2,U2).
variable(['_'|X],Y,Z,ok,U):-
   char_list(X,Y,Y1,Z1,U1).
variable([X|Y],Y,U,ok,V):-
   member(X,['_']).
solutions(variable([V,a,r],[],X,ok,Y),
[variable([V,a,r],[],X,ok,Y)]).
solutions(variable(['_',a,b],[],X,ok,Y),
[variable(['_',a,b],[],X,ok,Y)]).                    (3)
solutions(variable(['_'],[],X,ok,Y),
[variable(['_'],[],X,ok,Y)]).
```

図 9 学生モデルの内容

[4] 教育例

図10に教育例を示し、図中の番号に従い、以下に説明を加える。

(1) 開始メッセージ。 ここで入力された名前が新しいものであれば、その名前を持つ学生モデル格納用ファイルが生成される。既に存在すれば、そのファイルの内容がシステム内に取り込まれる。

(2) mis モード。

(3) y:次に進む i:中断 end:教育終了

(4) i により教育を一旦中断し、システムのユーティリティーを用いて、学生モデルの内容を表示させたり、PROLOGのプログラムを実行させたりしている。

(5) mis モード。 ここでは変数についての教育が行われている。

```
| ?- start.
今から、Prologの勉強を始めます。
名前を入力してください。name.                        (1)
Prologのシンタックスについて勉強しましょう。

Prologでは次のものが文字として使われます。
                                                     ┐
          英大文字(A-Z)                              │
          英小文字(a-z)                              (2)
          数字(0-9)                                  │
          記号(+ - * / < > = ? '                     │
               ~ : $ & @ ¥ . # ^)                    ┘
y または 番号で答えなさい --> y.

次に進みます。(y,i,end) --> i.                       (3)

** メニュー **
 call: Prologの実行
 disp: 生徒のモデルの表示
 end : 終了
 ==> disp.

 どのモデルを表示させますか ==> 文字列.

文字列に関する学生のモデルは ==>
char_list(X,Y,Z,yet,U):-
   char(X,Y,Y1,Z1,U1).
char_list(X,Y,Z,yet,U):-
   char(X,[_|X1],Y1,Z1,U1),
   char_list(X1,Y,Y2,Z2,U2).
char_list(X,Y,Z,yet,U):-
   char(X,X1,Y1,Z1,U1),
   char_list(X1,Y,Y2,Z2,U2).
                                                     (4)
  ==> call.

実行するゴールを入力して下さい ==>
 member(A,[1,2,3]),append([A],[o],M),fail.
1    [1,o]
2    [2,o]
3    [3,o]
 ** no **

  ==> end.

次に進みます。(y,i,end) --> y.                       ┐
                                                     │
** 変数について説明します **                         │
                                                     │
変数は名前付けのできない、                           │
   ある対象物を表わす時に用います。                  │
つまり英語の代名詞に相当します。                     │
変数は次の性質を持ちます。                           │
          1) 英大文字で始まる文字列である            (5)
          2) _で始まる文字列である                   │
          3) _(無名変数)である                       ┘
```

図 10 教育例

(6) 学生モデルが生成された。

(7) pds モード。 この場合、誤りが見つかったので、その箇所を学生に知らせ、誤りに則した教育がなされている。

(8) edu モード。 復習問題が与えられている。

```
下の選択枝の中で変数として正しいものを選びなさい。

 1. var  2. Var  3. _xyz  4. 345  5. _

y または 番号で答えなさい --> 4.

 学生モデルへの追加
    variable(X,Y,Z,yet,U):-
                number(X,Y,Y1,Z1,U1)
 学生モデル variable(X,Y,Z,U,V):          (6)
    variable(X,Y,Z,yet,U):-
                number(X,Y,Y1,Z1,U1).

次に進みます。(y,i,end)  --> y.

あなたは、変数について、誤った知識を持っていますね。
あなたが変数について持っている知識を検証した結果、
変数の下位概念である整数が
   間違っていることが判明しました。

  以下の中で変数として正しいものの記号と、
その根拠を番号で答えなさい。

   a. Variable b. var c. _var d. 3542 e.$var f. _

 1. 変数は記号で始まらなければならない                (7)
 2. 変数は英大文字で始まらなければならない
 3. 変数はアトムである
 4. 変数は _ から始まる文字列である
 5. 変数は整数である
 6. _ は変数を表わす

記号-番号 --> a-2,d-5.
間違いです。正解は a-2  c-4  f-6   です。

変数について復習しましょう。

変数は次の性質を持ちます。
     1) 英大文字(A,B,...,Y,Z)で始まる文字列である
     2) _で始まる文字列である                       (8)
     3) _ (無名変数)である

 次の選択枝の中で、変数として正しいものを選びなさい

 1. Var  2. _ab  3. 35  4. var  5. _  6. var_Var
番号で答えなさい --> 1,2,5.

次に進みます。(y,i,end)  --> end.                  (9)

今日の勉強はこれまで。
あなたの学習記録は、
ファイル name に登録されています。                 (10)
さようなら。

yes
| ?-
```

図 10（つづき）教育例

(9)（8）の最後に学生が入力した応答をpds モードで検証した結果、誤りがなかったので、次の教材の教育へ進む。この場合、end を入力し、システムの実行を終了させている。

(10)終了メッセージ。

[5] むすび

本論文では、帰納推論を用いた知的ＣＡＩシステムの枠組みと、その応用事例として現在開発中のPROLOGプログラミング教育システムについて述べた。

最後に、今後の研究課題を挙げ、解決の方針をまとめる。

- インタフェイス部の充実

  現在、システムと学生とのやりとりは、選択枝、あるいは単語入力により行っているが、将来は、より柔軟な自然言語で行う予定である。

- 学生モデルと学生の頭の中の状態との間に存在する矛盾にどう対処するか

  我々は、[3.2] で述べた仮定に基づいて学生モデルを生成している。しかし、実際には、学生がシステムとの対話中に、自分の犯した誤りに気付き、同じ対象についての問題に対して、以前入力した値(A) とは異なった値(B) を入力する場合がある。この時、出来上がった学生モデルと学生の頭の中の状態との間に矛盾が生じる（この矛盾をここでは正の矛盾と呼ぶ）。また、学生が数日前までは正しく応答していたにもかかわらず、今日、同じ問題をやらせてみると、誤った応答をする場合がある。この時も矛盾が生じる（この矛盾をここでは負の矛盾と呼ぶ）。従って、システムはこうした矛盾に敏感に反応、適切な処置を取る必要がある。正の矛盾に対しては、システムは、学生の応答Ａ、Ｂについてそれぞれの学生モデルを生成する。これらのモデルがpds モード、edu モードを通過する時、何が起きるかを考える。Ａに対して生成されたモデルは、pds モードで'誤り'であると判定され、学生モデルから削除され、この誤りについての教育が行われる。さらにedu モードで復習が行われる。Ｂに対して生成されたモデルは、pds モードで'正しい'と判定され、edu モードでは何もしない。以上のようにpds モード、edu モードの実行終了時点では、学生モデルはＢから生成されたモデルだけとなり、これは学生の頭の中の状態と一致する。つまり、正の矛盾は一時的なもので、学生モデルがpds モード、edu モードを通過することにより、正の矛盾を解消することができる。負の矛盾に対しては、適当な時期に'おさらい'することによりこの矛盾の検出に努めねばならない。

謝辞

日頃、熱心な討論をしていただく産研角所、豊田両研究室の諸氏に感謝します。

[参考文献]

1)田中・淵監訳：人工知能ハンドブック I,II，共立出版，(1983).

2)Vere,S.A.:Inductive Learning of Relational Productions,PATTERN-DIRECTED INFERENCE SYSTEMS,pp 281-295,N.Y.,Academic Press,(1978).

3)Michalski,R.S.:A Theory and Methodology of Inductive Learning,Artificial Intelligence, Vol.20,pp 111-161,(1983).

4)Shapiro,E.Y.:Algorithmic Program Debugging, London,MIT Press,(1982).

5)願化他：帰納推論を用いたPROLOGプログラミング教育システムについて，28回情処全大，1G-1, (1984).

6)河合他：帰納推論を用いたICAIシステムー汎用システムアーキテクチャの設計ー,Proc.of Logic Programming Conference,'84,15.4 (1984).