知識工学と人工知能　45 — 8
（１９８６・３・１０）

Prologと関係データベースとの結合システム DB-Prolog

今中　武　　上原　邦昭　　河合 和久　　豊田　順一

（大阪大学産業科学研究所）

１．まえがき

現在，Prolog と関係データベースとの結合システムが多く提案されている．このような結合システムを知識情報処理に用いる場合には，次の二つの機能を持つことが必要である．

1) 大量のファクトとゴールとのユニフィケーションを効率よくおこなう機能．
2) 節形式で表現されていない数値や文字列データを，Prolog プログラムでアクセスする機能．

たとえば，自然言語理解で用いる辞書をPrologで記述する場合に 1) の機能は有用である．この機能を用いると，従来どおりの Prolog プログラムによって，単語の意味，品詞等などの大量のデータを高速に検索することができる．また，画像処理などで用いる地図データを関係データベースに蓄積し，Prolog を使ってそのデータに集合演算を施す場合には 2) の機能が有用である．

本報告で紹介する DB-Prolog は，以上の二つの機能を備えたものである．従来より，引数に構造を持つファクトを，関係データベースに対応づけることは困難とされてきた．この問題は，ファクトの構造を展開して線状化することで解決している．このため，引数に構造を持つファクトと持たないファクトは区別なく関係データベースに登録できる．さらに，登録されたファクトは，関係データベースの高速検索機能を用いて検索される．以上の２点から，1) の効率的なユニフィケーションが実現されている．また，DB-Prolog から C，FORTRAN によって記述されたデータベース操作プログラムを呼び出すことが可能なために，2) のような非定形データに対しても，効率よくアクセスできるようになっている．

本報告では，DB-Prolog について説明する．以下，２章で Prolog と関係データベースとの結合について議論する．３章ではDB-Prolog の機能を具体的に示し，４章，５章で DB-Prolog の実現手法について詳説する．

２．Prolog と関係データベース

ここでは，本稿で用いる基本的な用語について説明する．さらに，Prolog と関係データベースとの結合手法について述べる．

2.1 Prolog

ユニット節のうち，引数に変数の含まれないものを，ファクトと呼ぶ．なお，特に断らないかぎりファクトは引数に構造を持つものとする．

2.2 関係データベース

関係データベースでは，情報を表の形で表現する．この表を関係といい，表の行，列をそれぞれタップル，属性と呼ぶ．さらに，タップルの各属性に対応する値を属性値と呼ぶ[2]．また，関係データベース中の関係はすべて正規化されている．正規化された関係では，表の中の行と列で指定されるあらゆる位置には常にただ一つの値が存在し，値の集合が存在することはない．

2.3 結合方式

Prolog と関係データベースとの結合には，二通りの方式がある．一つは，評価方式と呼ばれている方式で，Prolog を関係データベースの問い合わせ言語とみなすものである[3][12]．評価方式では，Prolog で記述されたプログラムを関係データベース操作言語に変換し，実行する．また，質問の処理をまとまった検索命令でおこなうために，高速集合演算機能が使用される．したがって，検索命令群を最適化するなどして，検索の高速化をすることができる．もう一つは，非評価方式と呼ばれている方式で，関係データベース中のデータは Prolog プログラムと同じ扱いをうける．非評価方式では，Prolog の処理系は関係データベース中のデータに対しても，ユニフィケーション

を試みる．すなわち，関係データベースをPrologの持つデータベースの一部分とみなしており，Prolog の持つ強力な推論機能をそのまま関係データベース中のデータにも利用できる．

DB-Prolog は，同時に二つ以上のデータベースをアクセスすることができ，データベースごとに評価方式，あるいは非評価方式で結合することが可能である．したがって，二つの結合方式を使い分ければ，両方の結合方式の長所を引き出すことができる．知識情報処理の側面から考えると評価方式による結合においては，データベース中の知識と外部データベース・内部データベース中の知識はまったく別のものとみなすことができる．したがって，データベースごとに閉じた世界を仮定することができる(Closed World Assumption)[9]．この仮定をおこなえば，各データベースは，ある世界に固有の知識を保持していることになる．DB-Prolog を用いて分野ごとの知識を別々のデータベースに登録し，それぞれのデータベースを評価方式によって結合すれば，特定の世界における知識を持つデータベースを個別にアクセスすることができる．このように，評価方式によって結合される各データベースをそれぞれの世界に割り当てて，知識の多世界化を実現することができる．さらに，推論時に必要となる一般的事実を非評価方式で結合されているデータベースに登録すれば，多世界の知識を一般的な知識を用いて操作することができる．

なお，以降では DB-Prolog と非評価方式で結合されている関係データベースを外部データベースと呼び，Prolog の持つデータベースを内部データベースと呼ぶ．また，DB-Prolog と評価方式によって結合されたデータベースをユーザデータベースと呼ぶ．

3． DB-Prolog のシステム構成

DB-Prolog は，Prolog インタプリタMV-Prologと関係データベースシステム DG/SQL とを結合したもので，日本データゼネラル社のスーパーミニコン MV-8000 II 上に構築されている[1][4]．本章では，DB-Prolog の持つ内部データベース，外部データベース，ユーザデータベースについて述べる．また，DB-Prolog がどのようにしてこれらの三種類のデータベースを操作するかを説明する．システムの構成図を図1 に示す．

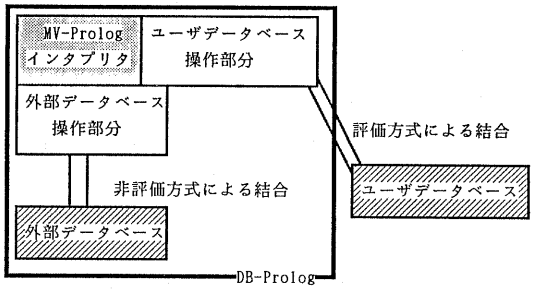


図1 システム構成図

3.1 内部データベース

内部データベースは MV-Prolog インタプリタの持つデータベースであり，Prolog プログラムはすべて内部形式に変換されてこの内部データベースに格納される．

3.2 外部データベース

外部データベースは DB-Prolog があらかじめ準備した関係データベースのことで，外部データベースへのアクセスは，すべてゴールの実行を通じて行われる[8]．ゴールと外部データベースに格納されたファクトとは，関係データベースの検索機能を用いてユニフィケーションされる．検索機能を用いたユニフィケーションを高速におこなうために，外部データベース中のファクトは引数によってインデックシングされたうえに，ユニフィケーションに必要な情報を付加されて関係表に納められている．

また，外部データベースは，非評価方式によって結合されており，Prolog プログラムは，内部データベース，外部データベースのどちらに登録されていても区別なく実行される．しかしながら，DB-Prolog は，ルールを外部データベース中で実行することはできないので，ルールは必ず内部データベースに取り込み，内部形式に変換してから実行する．

3.3 ユーザデータベース

ユーザデータベースは，利用者が自由に定義できるデータベースであり，利用目的に合わせて関係表の形を指定することができる．また，利用者は C，FORTRAN 等の手続き型言語を用いてユーザデータベースを操作するプログラムを記述することもできる．手続き型言語によって記述したデータベース操作プログラムはコンパイルすることができる．コンパイルしたプログラムの実行は DB-Prolog のユーザデータベースに対する操作命令の実行に比べて高速であるために，このようなプログラムの呼び出し機能を DB-Prolog に持たせている．

3.4 DB-Prolog の動作

DB-Prolog は，ゴールが実行されると，ゴールがユーザデータベースに対する操作命令かどうかを判断する．ユーザデータベースに対する操作の場合は，そのゴールからデータベース操作プログラムを合成し，合成されたプログラムを直ちに実行する．また，ゴールがユーザデータベースに対する操作命令でない場合は，内部データベースを検索し，ゴールとのユニフィケーションに成功する節を探し出す．ユニフィケーションに成功する節がなければ，外部データベースを検索する．外部データベースにもゴールとユニフィケーションできる節がなければ，このゴールは失敗する．

4．検索機能を用いたユニフィケーション

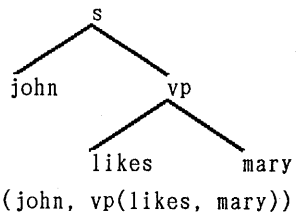
すでに述べたように，DB-Prolog が外部データベースに登録できる節はファクトのみである．引数に構造を持たないファクトは，各引数を属性値に割り当てることで関係に対応づけることができ，関係との相性がよいことが従来より指摘されている[7]．しかしながら，引数に構造を持つ場合には，関係に対応づけることは困難である．これは関係が正規化されているために，属性値に構造を持つことができないからである．この問題を解決するために，ファクトの持つ構造を展開した後に，関係に対応づけるという手法を用いる．以下 4.1 節でファクトをどのように関係に対応づけているかを示す．4.2 節ではファクトとゴールとのユニフィケーションについて述べ，4.3 節で検索機能を用いたユニフィケーションの具体的な手法を説明する．

4.1 ファクトと関係との対応

図2 で示すように，ファクトは 木構造で表わすことができる．さらに木構造の各ノードに座標を割り当てると，ファクトの構造を保存したままで線状化することができる．外部データベースには，この線状化された木構造が格納される．

DB-Prolog では，ファクトの線状化はシステム組み込み述語 assertout によっておこなわれる．述語 assertout は，ファクトを受け取ると，まず木構造に展開する．さらに木構造の各ノードに，ノードの位置をあらわす二次元座標と，ノードから下にのびる枝の数を割り当てる．2次元座標は，図3 に示すように木の右側から順に割り当てられる．これはリストを引数に持つファクトを展開すると，木構造が右方枝分かれ構造になるからである．

また図中の M は外部データベースに登録するファクトの引数の数の上限であり，現在では M=9 としている．この手法に基づいて図2 のファクトを関係に対応づけたものを図4 に示す．図4 において，arg はノードから下にのびる枝の数を表わす．また，枝のない場合は 0 とする．



s(john, vp(likes, mary))

図2 木構造表現

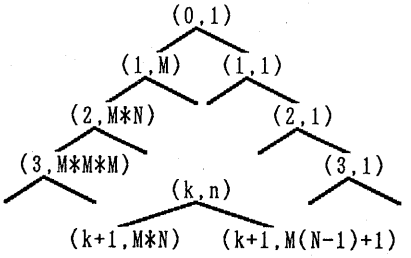


図3 二次元座標

x	y	node	arg
0	1	s	2
1	8	john	0
1	9	vp	2
2	72	likes	0
2	71	mary	0

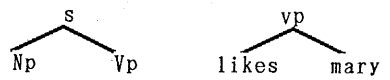
図4 関係表

4.2 ゴールとファクトとのユニフィケーション

木構造によって表現されたファクトと Prolog のゴールをユニフィケーションするために，ゴールもファクトと同様に木構造に展開される．木構造どうしのユニフィケーションは以下のようにおこなわれる．

まず，互いのルートノードが持つ属性値（アトムあるいはファンクタ）を比較する．もし，等しければ，木構造を下にたどって，さらにノードごとの比較をおこなう．下へのたどりは，座標が小さいノードからおこなわれるので，右からの下降縦型探索と同様になる．もし，ゴールのノードが変数である場合は，対応するファクトのノードに付加された属性が代入される．このようにして，最終的にゴールとファクトが同一の木構造になるとき，両者はユニフィケーション可能であるという．

たとえば，図5 に示すようなゴールを実行する場合を考える．このゴールは同図のような木構造に展開される．展開された木構造は，変数 Np に john を，変数 Vp に同図の木構造を代入すると，図2 の木構造と同一になる．したがって，ユニフィケーションは成功する．



|?- s(Np, Vp).　代入される木構造

図5 ゴールと代入される部分木

4.3 関係を用いたユニフィケーション

図4 に示したように，一つのファクトは複数のタップルに展開される．各ファクトは，一意となるような識別子が与えられる．これは，一つのファクトを表わすタップル集合と他のファクトを表わすタップル集合とを区別するために用いられる．したがって，実際には，図4 の関係に識別子，さらに検索の高速化のためにいくつかの属性が加えられている．ここでは簡単のために，ノード，X 座標，Y座標，そのノードからのびている枝の数の四つの値の組みを４項組と呼び，タップルは４項組と識別子（id）を属性値として持つものと仮定する．４項組は，ファクトの木構造の各ノードに対応している．したがって，4.2 節で示したユニフィケーションは，ゴールより得られる４項組のうち，変数に関するものを除いた集合を包含するような４項組の集合を外部データベースより検索する操作に対応する．例として，図6 のように二つのファクトが外部データベースに登録されているときに，図5 のゴールを実行する場合を考える．

id	x	y	node	arg
1	0	1	s	2
1	1	9	vp	2
1	1	8	john	0
1	2	81	likes	0
1	2	80	mary	0
2	0	1	s	2
2	1	9	runs	0
2	1	8	tom	0

```
s(john, vp(likes, mary)).
s(tom,runs).
```

図6 外部データベース

ゴールより得られる４項組を図7 に示す．

```
(x, y, node, arg)
(0, 1,    s,   2)
(1, 9,   Vp,    )   > 変数に関する
(1, 8,   Np,    )     ４項組
```

図7 ゴールから得られる４項組

id=1 によって識別される４項組の集合は，変数に関する４項組を除いて，ゴールの４項組集合を含んでいる．したがって，id=1 で表わされるファクトはゴールとのユニフィケーションに成功する．変数への代入は，ゴールより得られる変数に関する４項組と同じ座標を持つ４項組を，外部データベースから検索することによっておこなわれる．検索された４項組の arg が 0 でない時には，

その４項組に対応するノードを根とする部分木が代入される．

5．Prolog プログラムによるユーザデータベースの操作

DB-Prolog は，ユーザデータベースと評価方式によって結合されている．したがって，ユーザデータベースに対して，集合演算による操作をおこなうことができる．さらに，利用者が任意にデータベースを作成することができる．データベースの作成，および検索は，DB-Prolog のシステム述語の呼び出しによって行われる．本章では，この機能の実現手法とユーザデータベース操作用のシステム述語について説明する．

5.1 実現手法

データベースの作成，および検索をおこなう機能は，HLI(Host Language Interface) を用いて実現されている[11]．HLI は，日本データゼネラル社のデータベース管理システム DG/SQL が提供するデータベース操作言語と C 言語とのインタフェースであり，C 言語のライブラリ関数を持っている．このライブラリ関数は，DML（データベース操作言語），あるいは DDL（データベース定義言語）によって記述された命令を引数として受け取り，即時実行する関数である．このライブラリ関数は，ユーザデータベース操作用の組み込み述語が呼び出されたときに起動される．たとえば，データベース作成用組込み述語が呼び出されると，DB-Prolog は DDL によるデータベース定義プログラムを合成して，各命令をライブラリ関数の引数にわたす[10]．また，データベース検索用組込み述語が呼び出された場合は，DML によるデータベース操作プログラムを合成して，各命令をライブラリ関数の引数にわたす．DML を用いたデータベース操作プログラム中では，操作したい属性を指定するために属性名が必要である．この属性名は，データベース定義時に属性名を管理する関係表としてデータベース中に定義される．これによって，データベース操作用組み込み述語の呼び出し時に属性を指定する手間を省くことができる．また，DB-Prolog を用いずに作成したデータベースには属性名を管理する関係表は定義されていない．このようなデータベースにアクセスする場合は，利用者が属性名に関する情報を DB-Prolog に与えてやると，この情報をもとにして，属性名を管理する関係表をデータベース中に定義するようになっている．

5.2 ユーザデータベース操作述語

ここでは，ユーザデータベース操作述語のうち，データベースの定義，データベースへのデータの入力，データベースの検索をおこなう述語について，例を用いて説明する．なお，DB-Prolog のプロンプト，'#-' であり，C-Prolog の '|?-' に相当する．

5.2.1 データベースの定義

例として，ユーザデータベースを定義した後，図8 の関係表をつくる DB-Prolog のプログラムと，そのプログラムを実行する時に合成される DDL のプログラムを図9 に示す．

sname	status	city
smith	20	london
jones	10	paris
blake	30	paris
clark	20	london
adams	30	athens

図8　関係 Supplier

```
プログラム
 #- creatae_database(db),
    make_table(supplier(sname,status,city),
               key(sname,status)).
プログラムより合成される DDL
 CREATE DATABASE "db" DEFINED BY
 "DATABASE DEFINITION; TABLE attribute;
  table_name : CHAR(40); number : INTEGER;
  attribute_name : CHAR(40);
  KEY(table_name,number);
 END DATABASE DEFINITION;"
 ALTER DATABASE db APPLYING
 "ALTER DATABASE; ADD TABLE supplier;
 sname : CHAR(5); status : INTEGER;
  city : CHAR(10); END TABLE supplier;
 END ALTER DATABASE;"
```

図9 データベース作成

図9 の DDL プログラムは，まずユーザデータベースを db という名前で定義している．同時に，関係表の属性名を管理する関係表 attribute を

データベース中に定義している．関係 attribute は三つの属性を持つ．属性 table_name はデータベース中に存在する関係表の名前を表わす．属性 number は，属性の定義した時の順番を表わす．そして，属性 attribute_name は table_name, number で指定される関係表の属性の属性名を表わす．また DDL プログラムは，関係表 supplier をデータベース中に定義している．実際には，関係表 supplier の属性に関する情報を，データベース中の関係 attribute に登録する必要があるが，ここでは省略する．

5.2.2 データの登録

図8 に示す関係の一番めのタップルを 5.2.1 節でつくった関係表に登録するプログラムと，そのプログラムを実行する時に合成される DML のプログラムを図10 に示す．DML のプログラムは，まず関係 attribute から関係 supplier の属性名を検索して，変数 att_name1, att_name2, att_name3 に代入する．これらの属性名から，関係 supplier にタップルを登録する DML プログラムが合成される．図中において，DML のプログラム中で att_name1 にコロン ':' がついているのは，att_name1 が変数であることを示す．

```
プログラム
 #- insert_db(supplier(s1,smith,20,london)).
プログラムより合成される DML
 SELECT attribute_name INTO :att_name1
 FROM attribute IN db"
 WHERE table_name="supplier" AND number=1
同様の検索によって att_name2,att_name3,
を求める。
 INSERT INTO supplier IN db SET sname="smith",
    status=20, city="london"
```

図10 データベース定義

5.2.3 データの検索

図8 に示す関係から，status が 20 である sname と city を検索するプログラム，およびそのプログラムの実行にともなって合成される DML のプログラムを図11 に示す．DG/SQL では，検索結果が複数の場合に結果を蓄えておく領域として，cursor と呼ばれるスタックが用意されている．この例では，検索結果は，cursor1 に入れられている．cursor1 に蓄えられた結果は，システム組込み述語 fetch_db を用いて取り出すことができる．fetch_db を呼び出すたびに，status が20 であるタップルの属性 sname と city の値がそれぞれ変数 arg1, arg2 に代入される．さらに，これらの値は fetch_db の引数となっている項 (term) の第1引数 Sname, 第2引数 City に代入される．

```
プログラム
#- select_db(supplier(Sname,20,City),cursor1),
   fetch_db(cursor1(Sname,City)).
 プログラムより合成される DML

 SELECT attribute_name INTO :att_name1
   FROM attribute IN db"
  WHERE table_name="supplier" AND number=1
 同様の検索によって att_name2,att_name3
 を求める
 SELECT sname, city INTO cursor1
 FROM supplier IN db WHERE status=20
 FETCH FROM cursor1 IN db SET :arg1, :arg2
```

図11 データベース検索

6．応用例

DB-Prolog を知識情報処理に用いる例として，フレームの概念にもとづく簡単な知識ベースシステムを示す．フレーム構造に基づいた知識ベースはユーザデータベースに構築され，一般的な知識は外部データベースに登録される．DB-Prolog の組み込み述語 create_database を用いてユーザデータベースを定義した後，insert_db を用いて知識を蓄積するプログラムを図12 に示す．ユーザデータベースにフレームを登録する組み込み述語 insert_db の引数は，Prolog のファクトの形をしている．したがって，insert_db を用いると，ユーザデータベースの関係を考えることなくデータベースへの登録ができる．この例では，二人の秘書に関する知識が格納される．次に，外部データベースに知識を蓄積するプログラムを図13 に示す．知識の蓄積は，外部データベースにファクトを登録する述語 assertout を用いて行なわれる．この例では，応答をおこなうための質問に対する知識を外部データベースに格納している．た

とえば，「Where ではじまる疑問文は場所を尋ねている」といったような知識が格納される．また，外部データベースに格納された知識を用いて応答するプログラムを図14 に，その応答例を図15 に示す．例に示したように，DB-Prolog は，知識を一般的な知識と，分野に固有の知識とに分離して用いることが可能である．一般的な知識は，推論時に用いられることから，Prolog の非決定動作に対して，いつでも利用可能でなければならない．DB-Prolog は，非評価方式による結合により，外

```
define :- create_database(':udd:imanaka:db'),
  open_db(':udd:imanaka:db'),                          データベース db をオープンする．
  make_table(frame(frame_name,slot,facet,value),       関係 frame を定義する．
  key(frame_name,slot,facet)),
  insert_db(frame(secretary,weekday,value,at_office)), frame_name, slot, facet, value は属
  insert_db(frame(green,inheritance,upper,secretary)), 性名，frame_name,slot,facet はキーで
  insert_db(frame(white,inheritance,upper,secretary)), ある．
  insert_db(frame(green,place,if_needed,search)),
  insert_db(frame(white,place,if_needed,search)),
  insert_db(frame(green,holiday,value,at_home)),
  insert_db(frame(white,holiday,value,at_tennis_court)),
  insert_db(frame(green,age,value,twenty_four)),
  insert_db(frame(white,age,value,twenty)),
  insert_db(frame(secretary,job,value,secretary)).
```

関係 frame にタップルを登録する．

図12 ユーザデータベースの準備

```
data :- assertout(rule(search,'search:-message(day,Mess),put_out_q(Mess),
  nl,read(D),date(D,Day),get_f(Frame),f_get(Frame,Day,Ans),asserta(ans(Ans))')),
  assertout(day(monday,weekday)),
  assertout(day(tuesday,weekday)),
  assertout(day(wednesday,weekday)),
  assertout(day(thursday,weekday)),                    ファクトを外部データベースに登録する．
  assertout(day(friday,weekday)),
  assertout(day(saturday,weekday)),
  assertout(day(sunday,holiday)),
  assertout(week([monday,tuesday,wednesday,thursday,friday,saturday,sunday])),
  assertout(question([where,is,_,_],place)),
  assertout(question([how,old,is,_,_],age)),
  assertout(question([what,is,_,_],job)),
  assertout(question([how,many,years,old,is,_,_],age)),
  assertout(message(day,what_day_of_the_week_is_it_today)).
```

図13 外部データベースの準備

```
f_get(Frame_name,Slot,Ans) :- select_db(frame(Frame_name,Slot,value,Ans)).
f_get(Frame_name,Slot,Ans) :- select_db(frame(Frame_name,Slot,
     if_needed,Attached)),                             ユーザデータベースの関係 frame から
     asserta(current_frame(Frame_name)),               タップルを検索する．
     procedure(Attached),get_ans(Ans),
     retract(ans(_)),retract(current_frame(_)).
f_get(Frame_name,Slot,Ans) :- select_db(frame(Frame_name,inheritance,upper,Upper)),
     f_get(Upper,Slot,Ans).
procedure(Attached) :- rule(Attached,Rule),assert(Rule),call(Attached).
get_ans(Ans) :- ans(Ans).
get_f(Frame_name) :- current_frame(Frame_name).
date(D,Day) :- week(List),member(D,List),day(D,Day).
date(D,Day) :- write('It''s an uncorrect answer.'),nl,
     write('Please type a correct answer.'),nl,
     read(H),date(H,Day).
f_print(Frame_name) :- select_db(frame(Frame_name,Slot,Facet,Value),cursor1),
     fetch_name(Frame_name).
fetch_name(Frame_name) :- fetch_db(cursor1(Slot,Facet,Value)),  スタック cursor1 からデータを取り出
     write('Frame name -- '),write(Frame_name),                 す．
     nl,tab(5),write('Slot -- '),write(Slot),nl,
     tab(7),write('Facet -- '),write(Facet),nl,
     tab(10),write('Value -- '),write(Value),nl,
     fetch_name(Frame_name).
fetch_name(_).
start :- open_db(':udd:imanaka:db'),look,
     write('Please input your question '),nl,read_sentence(Input),go(Input).
start :- look,write('Please input your question '),nl,
     read_sentence(Input),go(Input).
go(List) :- question(List,Topic),get_object(List,Object),
     f_get(Object,Topic,Ans),write('She is '),put_out(Ans).
get_object([H],H).
get_object([Head|Rest],R) :- get_object(Rest,R).
```

図14 応答プログラム

```
Now defining user's database
message from DG/SQL--> ok
Now preparing the process of system database
database successfully opned
DB-Prolog version 1.5a

yes
#-['hyouka.pro',read_in,'print.pro'].
hyouka.pro consulted 3852 bytes 0.750006 sec.
read_in consulted 2388 bytes 0.600003 sec.
print.pro consulted 900 bytes 0.183338 sec.

yes
#- define.

yes
#- data.

yes
#- start.
Please input  your question
#: Where is Miss Green?
What day of the week is it today ?
#: tuesday.
She is at office
yes
#- start.
Please input your question
#: How old is Miss White?
She is twenty.
yes
#- start.
Please input your question
#: How many years old is Miss White?
She is twenty.
yes
#-
```

図１５ 応答例

部データベース中の知識を常時，推論に用いることができる．一方，分野に固有の知識は，各分野ごとに独立していなければならない．DB-Prologでは，分野ごとに，評価方式で結合されたユーザデータベースを割り当てることによって，違った分野の知識を用いた処理が可能になる．このように知識を，分類して利用・管理することは，知識ベースの管理，完全性の保持，検証をおこなう上で重要であり，DB-Prolog を用いれば簡単にできるという利点がある．

7．むすび

Prolog と関係データベースとを評価方式，非評価方式の両方で結合する DB-Prolog を示した．DB-Prolog を用いると，分野に特有の知識と一般的な知識を分けて用いることが簡単にでき，知識ベースシステムにおける知識の混同，複雑化を緩和することができる．ユーザデータベースはそれぞれを閉じた世界と考えることができ，複数のユーザデータベースを別々の世界に割り当てて定義することにより，知識の多世界機能を実現させることができる．また，複数のユーザデータベースを知識の階層ごとに割り当てて定義すると知識の階層性を実現させることになる．今後の課題として，ユーザデータベースに対する検索を高速化するために，検索プログラムの最適化があげられる．

参考文献

[1] AOS, AOS/VS 解説書, Data General Co. (1983).

[2] Date, C.J.: An Introduction to Database Systems, 3rd edition (1981).

[3] Deyi, L.: A PROLOG Database System, John Wiley & Sons Inc. (1984).

[4] DG/SQL User's Manual, Data General Co. (1984).

[5] Gallaire, H. and Minker, J.: Logic and Data Base, Plenum Press (1978).

[6] 林 他: 大規模知識ベースシステム実現のための Prolog の拡張，情報処理学会，第30回全国大会講演論文集，1L-4 (1985).

[7] Kowalski, R.: Logic for Problem Solving, North-Holland (1979).

[8] 今中 他: 大規模知識ベースシステム実現のための Prolog の拡張(2)，情報処理学会，第31回全国大会講演論文集，6M-1 (1985).

[9] Reiter, R.: On Closed World Data Bases, in H. Gallaire and J. Minker (eds.), Logic and Data Bases, Plenum Press (1978).

[10] The C Language Reference and Runtime Manual, Data General Co. (1983).

[11] The Host Language Interface User's Manual, Data General Co. (1985)

[12] 角田 他: 関係代数演算専用エンジンを備えた関係データベースマシン Delta，日経エレクトロニクス，1985.9.23日号，pp.235-280 (1985).

[13] 上野 晴樹: 知識工学入門，オーム社 (1985).