数理解析研究所講究録
第461巻 1982年 322-352

外延と内包を取り扱える拡張データベースシステム

大阪大学 工学部 野田通弘 打浪清一 手塚慶一

1．まえがき

電子計算機の処理能力の向上と、外部記憶の大容量化、低価格化に従い、従来の事務計算、技術計算などの単純計算だけではなく、種々のデータを入手時に計算機のファイルに蓄積し、必要に応じてそれを検索して使用するデータベースシステムの技術が発達し、実用化されてきた。

しかしながら、現在実用になっているのは、銀行の預金管理や、在庫管理などの定型事務データベースである。これらの取り扱われる情報は、意味構造が比較的単純な外延情報であり、対象が変われば、対応できる十分な技術、ソフトウェアがまだ十分にできていない。

そこで、このような外延情報だけではなく、内包情報のような、意味構造がより複雑なものまでをも含めた広い範囲の情報を、効率的に記述、処理できるシステムが望まれている。

実際、知識、画像などのデータに対しては、有効なDBMSが無く、各所でその開発研究がなされている。

特に知識に関しては、百科辞典や数学辞典などがデータベース化され、研究や開発段階で仮説の作成や、検定に使用できないかという希望が多い。最近、人工知能における知識表現、論理的演繹の研究と、データベースにおける処理、設計技術の結合である“演繹データベースシステム”の基礎となる、定理証明理論や関係モデルの研究が注目されているが、本研究では、データベース研究の側から取り扱える情報を拡張する方向で考察を行なう。

“知識”と“データ”を明確に区別し、それぞれについて記述、処理形式を定める方法が存在するが、一般に、そのように明確に区別することは困難であり、処理形式も複雑になると考えられるので、本研究では、両者を同一レベルで記述処理する手法について考察する。

以上のような立場で、まず、記述情報について、その記述形式などにより分類を行なう。また、人間の行なう常識を用いた推論について考察を加え、Lindgreenモデルを基に、外延、内包情報を取り扱うことができ、そのような推論を考慮したデータモデルを提案する。そして、LISPによる実際の記述、処理形式を示し、最後に、いくつかの例を挙げて説明を加え

る。

2. 記述情報の分類

まず最初に、本研究で提案するデータベースシステムが取り扱う情報について、それがどのような性質を持つものなのかを分析しなければならない。一般に、知識と呼ばれるものには、画像情報やパターン情報なども含まれるが、ここでは、記述情報についてだけ分析を行なう。

2.1. 外延と内包

『ジョンの好きな飲み物』という例について説明すると、異なった世界に対応して、異なった指示物が対応していることがあり得る。例えば、冬という世界ではウィスキーであり、夏ではビールである。このような2つの異なった指示物（ウィスキー，ビール）を外延と呼び、これらの指示物に共通な性質（ジョンの好きな飲み物）を内包と呼ぶ。つまり、内包とは各世界ごとにその外延をふり分け、もしくは割り当てる関係づけの働きを持った関数であると考えられる。

2.2. 記述形式

記述形式としては、宣言的記述、手続的記述の2種類が存在する。以下に両記述形式の特徴について述べるが、一般に、両記述形式は対称的な特徴を持っており、一方の長所が他方の短所となっている。

（ｉ） 宣言的記述

これは、表形式、辞典形式で記述されたものであり、長所としては、変更、追加が容易で、使用するのが容易であること、また短所としては、ある特定の一連の動作などを表現することが困難であることなどがあげられる。

（ii） 手続的記述

これは、プログラム形式などで記述されたものであり、長所としては、あるアルゴリズムの手順などを表現することが容易であること、階層構造をなす知識の表現が可能であること、また短所としては、変更、追加などにより他へ影響を与えやすいことなどがあげられる。

知識 ∋ 記述情報 { 外延情報 { 宣言的記述 / 手続的記述 } / 内包情報 { 宣言的記述 / 手続的記述 } }

図１　記述情報の分類

2.3．記述例

図１に示された両情報の各記述形式の具体例を以下に示す。

（ｉ）外延情報の宣言的記述

集合などの要素を列挙したものであり、従来のデータベースシステムで取り扱われてきたデータがこれにあたる。

例。ビール、ウィスキー、2、4、6、---

。ジョンの年令は8才である。

(ii) 外延情報の手続的記述

集合などの要素を生成するための手続きがこれにあたる。

例。$I=2\cdot n$　$n=1,2,3$ ---

。N君の年令は、彼が小学校x年生ならば、x+5才ぐらいです。

(iii) 内包情報の宣言的記述

集合などの要素を、その性質で指定したものがこれにあたる。

例。ジョンの好きな飲み物、偶数(である数)

。ジョンの年令は、トムと同じです。

(iv) 内包情報の手続的記述

集合などの性質を手続的に記述したものがこれにあたる。

例。2で割って余りが0ならば偶数である。

一般に、どの記述形式が一番良いとは決定できない。また、必ずしもすべての形式で記述可能ではないので、適宜、相互変換できることが望ましいと言える。しかし、外延情報の手続的記述から外延情報の宣言的記述への変換は、プログラムを実行することに相当するので、比較的容易であるが、その逆の変換は非常に困難である。

2.4．質問による分類

- What-oriented ：情報を検索、または、プログラムなどの手続きを実行して返答する。
- How-oriented ：以下のことを要求するもの。
  - (i) 答を得るまでの演繹過程
  - (ii) アルゴリズム
  - (iii) プログラムなどの手続きの加工

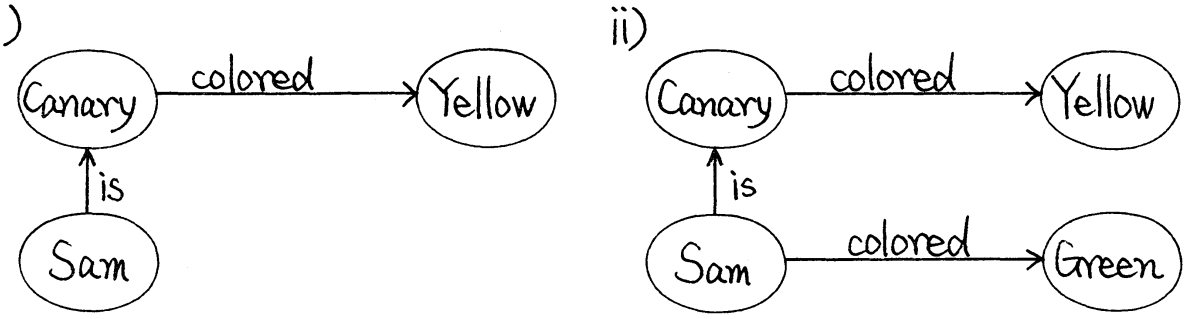
図2　質問の分類

What-oriented な質問には、外延情報の宣言的記述を検索、または、手続的記述情報を実行して答え、How-oriented な質問には、手続的記述情報を用いて答える。このように、手続的記述情報には、答を導き出すための手段、つまり、知識としての面と、加工される対象であるデータとしての面が存在する。従って、知識とデータを同一レベルで記述、処理する必要がある。

3. 人間の行なう常識を用いた推論

サムがカナリアであり、カナリアは一般に黄色であるという情報がある。この場合、我々は、『サムの色は？』という質問には、常識を用いて『黄色だろう』と答えることができる。しかし、これがデータベース内の情報であるとすると、データベースはこのような簡単な推論もすることができない

ので、サムの色に関する検索に対しては、サムの色についての情報がないので答えることができない。（図3－i)参照）



Question. Find Sam's color.
Answer. Yellow.

Question. Find Sam's color.
Answer. Green.

図3　常識を用いた推論例

このような推論は一階述語論理を用いれば可能であるが、サムの色が緑であるという情報が加わった場合、緑と黄色の2つの答が出てくる。しかし、我々は明確に緑と答えることができる。（図3－ii)参照）

このように我々は、質問に関する情報が不足していると、より高いレベルの情報、つまり、常識を使って答えようとする。

4．外延・内包情報を取り扱うためのデータモデル

2節で示した4種類の情報をすべて同一レベルで取り扱い、3節で考察した人間の行なう常識を用いた推論が可能なデータモデルを、Lindgreenモデルを拡張、変形して提案し、その概念について述べる。

〔定義4.1〕 個体 e

個体とは、現実世界の具体的、あるいは抽象的な事物、事象に対応するものであり、何らかの属性を持ち、他の個体と区別できるものである。

個体間の階層性として、異質な構成要素の一般化にあたるアグリゲーション(aggregation)、同質の構成要素の一般化にあたるジェネラリゼーション(generalization)などがあるが、本研究では、個体が集ってその集合が独自の属性を持つと考えられる時は、その集合を1つの個体として他の個体と同等に取り扱う。また、個体間の階層性は、属性の下位伝搬性によってある程度区別する。

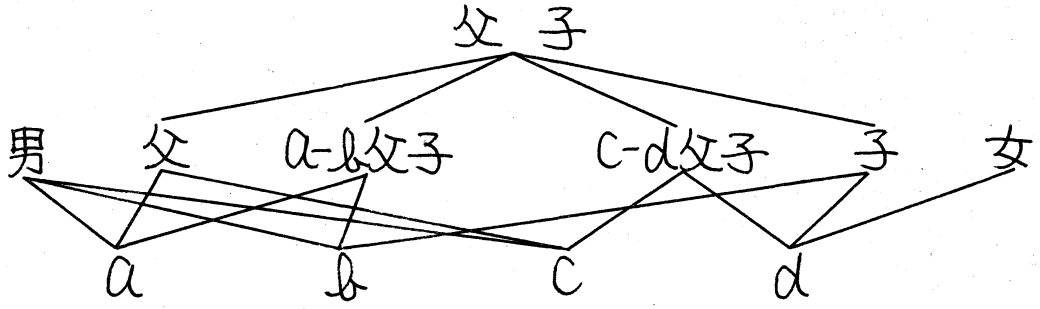


図4 個体の階層性の一例

〔定義4.2〕 属性 p

属性とは個体の有する性質のことであり、個体から属性値への写像である。下位の個体に伝搬する属性と、伝搬しない属性の2種類が存在する。この伝搬属性を下位の個体から上位の個体へたどることにより、3節で述べた推論が可能とな

る。例えば、図4において、個体“父子”の持つ属性は、個体“a-b父子”、“c-d父子”には伝搬するが、個体“父”、“子”には伝搬しない。また、下位の個体の持つ属性の方が、上位の個体から伝搬してくる属性より優先度が高い。

また、属性を写像と考えることにより、以下に示すような、属性の合成、逆属性を定めることができる。ただし、$y=R:x$とは、個体xの属性Rの属性値がyであることを示す。

(i) 属性の合成

$\mathrm{Mother}\circ\mathrm{Father}:x\in G\mathrm{mother}:x$

(ii) 逆属性

$R:x=y$が真である時に、$R':y=x$が真である場合、R'をRの逆属性R^{-1}という。

$\mathrm{Father}\circ\mathrm{Father}^{-1}:x=x$ ただし $\mathrm{Father}^{-1}\circ\mathrm{Father}:x\ni x$

(iii) 応用例 $F=\mathrm{Father}$

$a=F:b$
$b=F:c$
$c=F:d$

→ 個体cに関する情報を削除 →

$a=F:b$
$c=F^{-1}:b$
$c=F:d$

→ 個体bに関する情報を削除 →

$b=F^{-1}:a$
$c=F^{-1}\circ F^{-1}:a$
$c=F:d$

ここにおいて、$c=F^{-1}:b$、$b=F^{-1}:a$、$c=F^{-1}\circ F^{-1}:a$を追加することによって、a、b、c、d間の関係がとぎれることを防ぐことができる。

〔定義4.3〕 属性値 v

属性値とは、属性の程度を、数値、文字列、およびそれらの集合で表現したものである。また、属性値を導出するための手続き、手続きを実行するための関数呼び出しも属性値に含むものとする。このように定めることにより、手続的記述情報や内包情報を属性値として取り扱うことができる。

〔定義4.4〕 基本単位 $(e\ p\ v)$

個体、属性、属性値の3つから構成される組を基本単位として定義する。システムに入力される情報は、表形式などの十分に整理されたものではなく、より断片的なものであるので、属性の追加などに柔軟に対応できるように、基本単位をこのように定義する。

〔定義4.5〕 同型情報集合 I^p

同型情報集合とは、同一の属性を持つ基本単位の集合であり、以下のように定義する。

$$I^p=\{(e_1\ p\ v_1)(e_2\ p\ v_2)\ \cdots\ (e_n\ p\ v_n)\}$$
$$=\{p, W(E,p)\}$$

ただし

$$W(E,p)=\{(e\ v):e\in E\subseteq E(p)\wedge v\in V\subseteq V(p)\}$$

$E(p)$：pを属性として持つすべての個体の集合

$V(p)$：属性pが取り得るすべての属性値の集合

〔定義4.6〕　同質情報集合　I^e

同質情報集合とは、同一の個体に関する基本単位の集合であり、以下のように定義する。

$$I^e=\{(e\ p_1\ v_1)(e\ p_2\ v_2)\ \cdots\ (e\ p_n\ v_n)\}$$
$$=\{e, D(P, e)\}$$

ただし

$$D(P, e)=\{(p\ v) : p\in P\subseteq P(e)\wedge v\in V\subseteq V(p)\}$$

$P(e)$：個体eが持つすべての属性の集合

〔定義4.7〕　システム全体の集合　I^{P_s}, I^{E_s} と
その部分集合　I^P, I^E

同型、同質情報集合を用いて、システム全体の集合とその部分集合は以下のように定義できる。

$I^{P_s}\supseteq I^P=\bigcup_i I^{p_i}$, $p_i\in P\subseteq P_s$, P_s：システムが持つすべての属性の集合

または

$I^{E_s}\supseteq I^E=\bigcup_i I^{e_i}$, $e_i\in E\subseteq E_s$, E_s：システムが持つすべての個体の集合

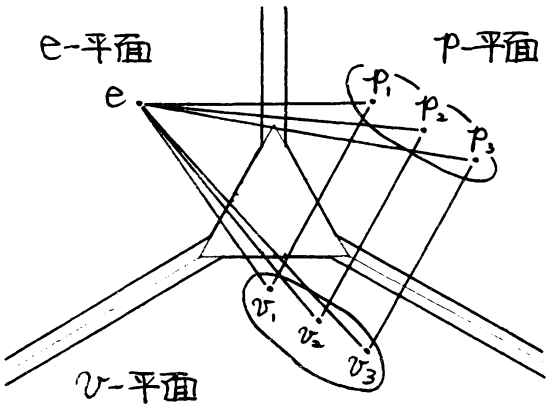
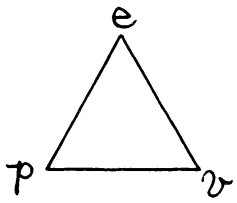


図5　概念図

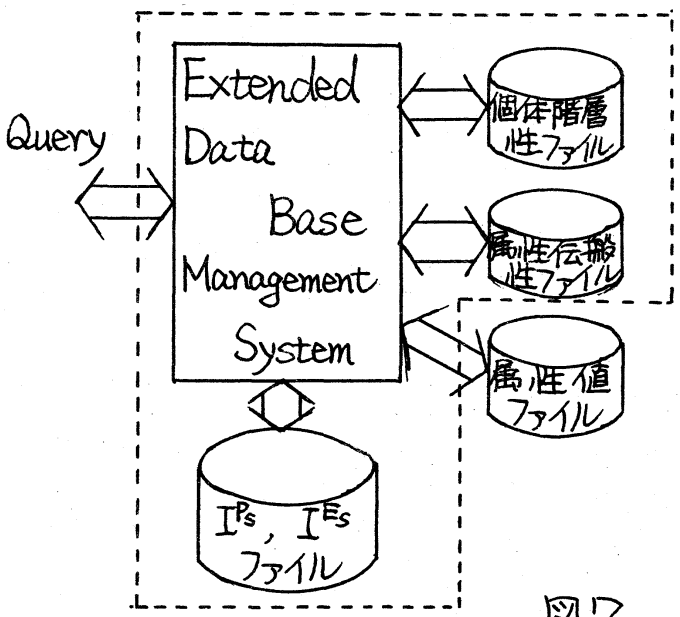


e：個体
p：属性
v：属性値

図6　基本単位

個体を1つ定めると、その個体が持つすべての属性の集合、属性に1対1に対応した属性値の集合が決まる。このことを概念図として図5に示す。

5. システム構成



- 個体階層性ファイル
  個体間の階層性に関する情報
- 属性伝搬性ファイル
  属性の伝搬性に関する情報
- 属性値ファイル
  属性値の単位などに関する情報

図7　システム構成

4節で述べたデータモデルに基づいたシステム構成を図7に示す。今回は、点線内の部分をインプリメントした。

6. LISPによる具体的記述形式

4節で述べたデータモデルを具体的に記述するには、S-M変換可能なLISP系の言語が適している。以下に、5節でのシステム構成に基づいた、LISPによる具体的記述形式を示す。

〔表記法6.1〕 個体 e

個体は、個体名を表現するアトムによって記述する。

例 ∘ HUMAN, ARES, HERA

〔表記法6.2〕 属性 p

属性は、属性名を表現するアトムによって記述する。

例 ∘ SEX, FATHER, MORTAL

〔表記法6.3〕 属性値 v

属性値は、属性値を表現するS式によって記述する。

例 ∘ FEMALE, PROG, T, (HERA ZEUS)

∘ (SEARCH "DIONIS" "FATHER" "E?")

```
∘ ( ((SEX (LAMBDA (E)
        (COND ((NULL (NULL (SEARCH "A?" "FATHER" E)))
               (QUOTE MALE) )
              ((NULL (NULL (SEARCH "A?" "MOTHER" E)))
               (QUOTE FEMALE) )
              (T (QUOTE WAKARIMASEN)) ) )))
```

〔表記法6.4〕 基本単位 (e p v)

基本単位は、個体、属性、属性値の3つの要素から構成されるS式で記述する。

例 ∘ (DIONIS SEX MALE)

∘ (ARES FATHER (SEARCH "DIONIS" "FATHER" "E?"))

```
∘ (PROG SEX
    ( ((SEX (LAMBDA (E)
        (COND ((NULL (NULL (SEARCH "A?" "FATHER" E)))
               (QUOTE MALE) )
              ((NULL (NULL (SEARCH "A?" "MOTHER" E)))
               (QUOTE FEMALE) )
              (T (QUOTE WAKARIMASEN)) ) )))
```

〔表記法6.5〕 同型情報集合 I^{p} :

$$(p\ (e_1\ e_2 \cdots e_n)(v_1\ v_2 \cdots v_n))$$

同型情報集合は、属性名、個体名リスト、個体名リストと要素が1対1に対応した属性値リストの3つの要素から構成されるS式で記述する。

例 ∘ (SEX (ARES HARMONIA DIONIS)
(PROG FEMALE MALE))

〔表記法6.6〕 同質情報集合 I^{e} :

$$(e\ (p_1\ p_2 \cdots p_n)(v_1\ v_2 \cdots v_n))$$

同質情報集合は、個体名、属性名リスト、属性名リストと要素が1対1に対応した属性値リストの3つの要素から構成されるS式で記述する。

例 ∘ (DIONIS (FATHER MOTHER SEX)
(ZEUS SEMELE MALE))
∘ (HUMAN (MORTAL FLY)
(T F))

〔表記法6.7〕 システム全体の情報: I^{P_s} , I^{E_s}
およびその部分集合: I^{P} , I^{E}

システム全体の情報およびその部分集合は、同型または同質情報集合を用いて以下のように記述される。

$I^{P_s} \supseteq I^{P}$: I^{p_i} を要素とするS式 $p_i \in P \subseteq P_s$

または、

$I^{E_s} \supseteq I^{E}$: I^{e_i}を要素とするS式　　$e_i \in E \subseteq E_s$

ただし、E_s：システムが持つすべての個体の集合

P_s：システムが持つすべての属性の集合

例。I^{E} :((ARES (FATHER MOTHER SEX)
(ZEUS HERA PROG))
(HERA (SEX PARENT)
(PROG PROG)))

。I^{P} :((SEX (ARES DIONIS) (PROG MALE))
(MOTHER (ARES DIONIS) (HERA SEMELE)))

I^{E}, I^{P}は質問に対する答の出力形式として用いられるが、システム全体の情報は、実際には、次に述べるI^{P_s}ファイルとI^{E_s}ファイルの記述形式にそって表現される。

【表記法6.8】　I^{P_s}ファイルとI^{E_s}ファイル

5節のシステム構成に示した I^{P_s}ファイルと I^{E_s}ファイルは、以下のように記述される。

(i)　I^{P_s}ファイル : IPS

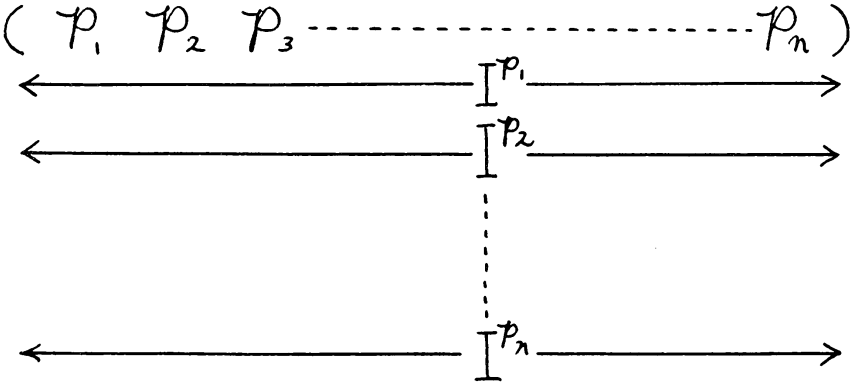
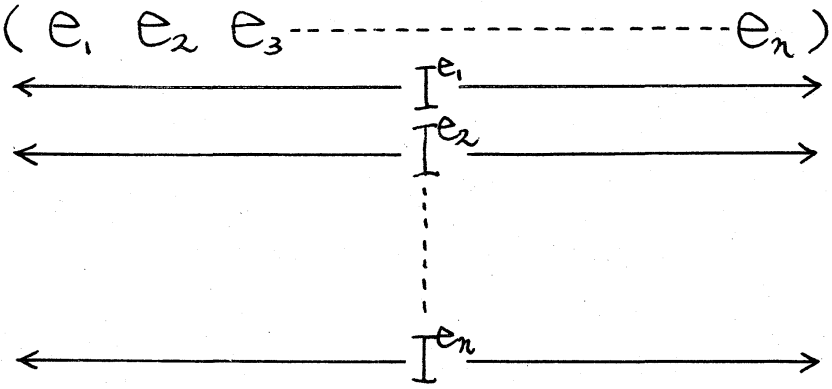


図8　I^{P_s}ファイルの記述形式

I^{P_s}ファイルは、システムが持つすべての属性名のリスト

と、その属性名の順番と同順に記述された同型情報集合がシーケンシャルに表現されている。

(ii) I^{Es}ファイル : IES



図? I^{Es}ファイルの記述形式

I^{Es}ファイルも同様に、システムが持つすべての個体名のリストと、その個体名の順番と同順に記述された同質情報集合がシーケンシャルに表現されている。

〔表記法6.?〕 個体階層性ファイル : EFILE

(e_1 e_2 e_3 ---------- e_n)
(e_1 (e_1の上位の個体)(e_1の下位の個体))
⋮ ⋮
(e_n (e_nの上位の個体)(e_nの下位の個体))

図? 個体階層性ファイルの記述形式

個体階層性ファイルは、個体名リストと、その個体名の順番と同順に記述された、図?に示すような個体間の階層性に関する情報がシーケンシャルに表現されている。

〔表記法6.10〕 属性伝搬性ファイル

属性伝搬性ファイルは、ファイルPFILE1とPFILE2によって構成される。

(i) PFILE1ファイル

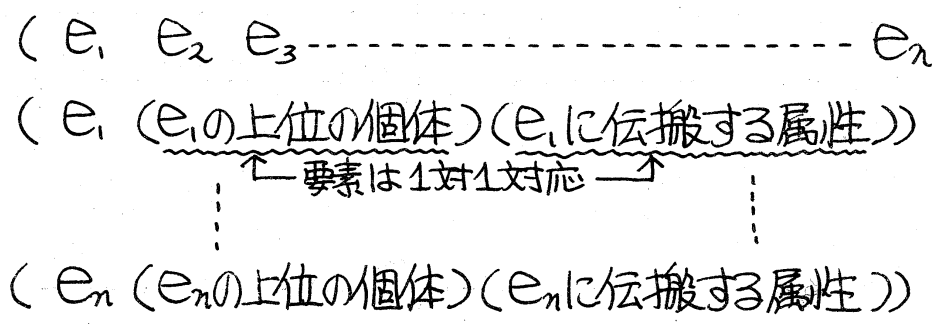


図10 PFILE1ファイルの記述形式

このファイルは、属性の伝搬性を個体をキーにして引くものであり、個体名リストとその個体名の順番と同順に記述された、図10に示すような属性の伝搬性に関する情報がシーケンシャルに表現されている。個体に伝搬する属性の部分で、"ALL"は、その個体に上位の個体が持つ属性がすべて伝搬すること、"NIL"は、その個体には上位の個体から伝搬する属性がないことを表わしている。それ以外は、伝搬する属性を列挙してS式で記述する。

(ii) PFILE2ファイル

このファイルは、属性の伝搬性を属性をキーとして引くものであり、属性名リストとその属性名の順番と同順に記述された、図11に示すような属性の伝搬性に関する情報が

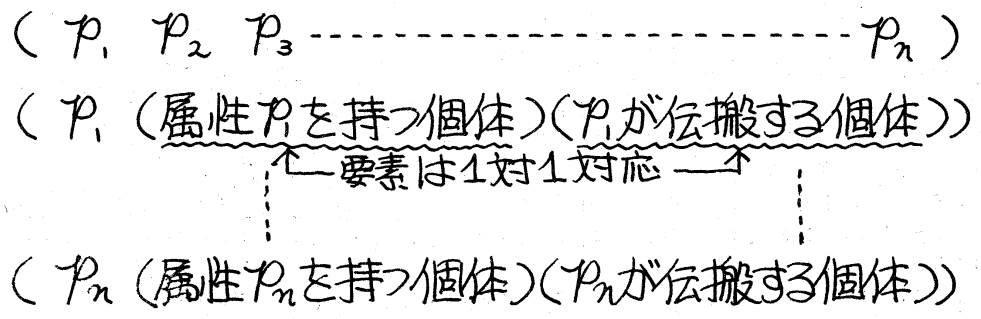


図11 PFILE2ファイルの記述形式

シーケンシャルに表現されている。属性が伝搬する個体の部分で、"ALL"は、属性がその属性を持つ個体の下位の個体すべてに伝搬すること、"NIL"は、その属性が下位の個体には伝搬しないことを表わしている。それ以外は、その属性が伝搬していく個体を列挙してS式で記述する。

なお、LISPには大阪大学大型計算機センターのLISP1.5を用いた。これは他言語とのリンクができず、ファイルからS式を1つだけシーケンシャルに読む機能しかないため、このようなファイル編成をとった。I^{P_S}ファイルとI^{P_S}ファイル、PFILE1ファイルとPFILE2ファイルは、それぞれ互いに一方が他方の転置形になっていると考えられるので、実際にはどちらか一方でこと足りるものであるが、効率面を考慮して、情報検索には両方のファイルを使用している。

図12以降に、各ファイルの具体的内容例を示す。

```
(ARES HARMONIA DIONIS ROBINSON HERA ZEUS SEMELE HUMAN PROG)
(ARES (FATHER MOTHER SEX PARENT GPARENT ANCESTOR DESCEND1 DESCEND2)
      ((SEARCH (QUOTE DIONIS) (QUOTE FATHER) (QUOTE E*)) HERA
       PROG PROG PROG PROG PROG PROG) )        内包情報の宣言的記述
(HARMONIA (FATHER MOTHER SEX PARENT GPARENT ANCESTOR DESCEND1 DESCEND2)
          (ARES ROBINSON FEMALE PROG PROG PROG PROG PROG) )
(DIONIS (FATHER MOTHER SEX PARENT GPARENT ANCESTOR DESCEND1 DESCEND2)
        (ZEUS SEMELE MALE PROG PROG PROG PROG PROG) )   手続きを実行させるためのトリガー
(ROBINSON (SEX PARENT GPARENT ANCESTOR DESCEND1 DESCEND2)
          (PROG PROG PROG PROG PROG PROG) )
(HERA (SEX PARENT GPARENT ANCESTOR DESCEND1 DESCEND2)
      (PROG PROG PROG PROG PROG PROG) )
(ZEUS (SEX PARENT GPARENT ANCESTOR DESCEND1 DESCEND2)
      (PROG PROG PROG PROG PROG PROG) )
(SEMELE (SEX PARENT GPARENT ANCESTOR DESCEND1 DESCEND2)
        (PROG PROG PROG PROG PROG PROG) )
(HUMAN (MORTAL FLY) (T F) )
(PROG (SEX PARENT GPARENT ANCESTOR DESCEND1 DESCEND2)
   ((SEX (LAMBDA (E)
           (COND ((NULL (NULL (SEARCH (QUOTE A?) (QUOTE FATHER) E)))
                  (QUOTE MALE) )
                 ((NULL (NULL (SEARCH (QUOTE A?) (QUOTE MOTHER) E)))
                  (QUOTE FEMALE) )
                 (T (QUOTE WAKARIMASEN)) ) )))
    ((PARENT (LAMBDA (E)
           (PROG (X ANS)                                  手続的記述情報
             (SETQ X (SEARCH E (QUOTE FATHER) (QUOTE E?)))
             (SETQ ANS (APPEND ANS X))
             (SETQ X (SEARCH E (QUOTE MOTHER) (QUOTE E?)))
             (COND ((NULL X) (RETURN ANS)))
             (RETURN (APPEND X ANS)) ))))
    ((GPARENT (LAMBDA (E)
        (PROG (X ANS)
          (SETQ X (SEARCH E (QUOTE PARENT) (QUOTE A?)))
       L1 (COND ((NULL X) (RETURN ANS)))
          (SETQ ANS (APPEND ANS (SEARCH (CAR X) (QUOTE PARENT)
                                        (QUOTE A?)) ))
          (SETQ X (CDR X))
          (GO L1) ))))
    ((ANCESTOR (LAMBDA (E)
        (PROG (X ANS)
          (SETQ X (SEARCH E (QUOTE PARENT) (QUOTE A?) ))
          (SETQ ANS X)
       L1 (COND ((NULL X) (RETURN ANS)))
          (SETQ ANS (APPEND ANS (SEARCH
                    (CAR X) (QUOTE ANCESTOR) (QUOTE A?))))
          (SETQ X (CDR X))
          (GO L1) ))))
    ((DESCEND1 (LAMBDA (E)
        :
```

図12 I^{ES} ファイルの内容例：IES

```
(FATHER MOTHER SEX PARENT GPARENT ANCESTOR DESCEND1 DESCEND2 MORTAL FLY)
(FATHER (ARES HARMONIA DIONIS)
    ((SEARCH (QUOTE DIONIS) (QUOTE FATHER) (QUOTE E?)) ARES ZEUS) )
(MOTHER (ARES HARMONIA DIONIS) (HERA ROBINSON SEMELE) )
(SEX (ARES HARMONIA DIONIS ROBINSON HERA ZEUS SEMELE)
     (PROG FEMALE MALE PROG PROG PROG PROG) )
(PARENT (ARES HARMONIA DIONIS ROBINSON HERA ZEUS SEMELE)
        (PROG PROG PROG PROG PROG PROG PROG) )
(GPARENT (ARES HARMONIA DIONIS ROBINSON HERA ZEUS SEMELE)
         (PROG PROG PROG PROG PROG PROG PROG) )
(ANCESTOR (ARES HARMONIA DIONIS ROBINSON HERA ZEUS SEMELE)
          (PROG PROG PROG PROG PROG PROG PROG) )
(DESCEND1 (ARES HARMONIA DIONIS ROBINSON HERA ZEUS SEMELE)
          (PROG PROG PROG PROG PROG PROG PROG) )
(DESCEND2 (ARES HARMONIA DIONIS ROBINSON HERA ZEUS SEMELE)
          (PROG PROG PROG PROG PROG PROG PROG) )
(MORTAL (HUMAN) (T) )
(FLY (HUMAN) (F) )
```

図13　I^{PS}ファイルの内容例：IPS

```
(ARES HARMONIA DIONIS ROBINSON HERA ZEUS SEMELE HUMAN)
(ARES (HUMAN) () )
(HARMONIA (HUMAN) () )
(DIONIS (HUMAN) () )
(ROBINSON (HUMAN) () )
(HERA (HUMAN) () )
(ZEUS (HUMAN) () )
(SEMELE (HUMAN) () )
(HUMAN () (ARES HARMONIA DIONIS ROBINSON HERA ZEUS SEMELE) )
```

図14　個体階層性ファイルの内容例：EFILE

```
(ARES HARMONIA DIONIS ROBINSON HERA ZEUS SEMELE)
(ARES (HUMAN) (ALL) )
(HARMONIA (HUMAN) (ALL) )
(DIONIS (HUMAN) (ALL) )
(ROBINSON (HUMAN) (ALL) )
(HERA (HUMAN) (ALL) )
(ZEUS (HUMAN) (ALL) )
(SEMELE (HUMAN) (ALL) )
```

図15　属性伝搬性ファイルの内容例：PFILE1

```
(MORTAL FLY)
(MORTAL (HUMAN) (ALL) )
(FLY (HUMAN) (ALL) )
```

図16　属性伝搬性ファイルの内容例：PFILE2

7. LISPによる処理形式

本節では、アクセス関数を中心として、外延・内包情報の処理形式について述べる。

7.1 アクセス関数

アクセス関数として、図17に示すようなSEARCH, および、SEARCHL関数が存在する。

	e	p	v	Answer
SEARCH	?	?	?	システム全体の情報　非現実的
	?	?	○	vで規定される(e p)の集合
	?	○	?	pで規定される(e v)の集合
	?	○	○	p,vで規定されるeの集合
	○	?	?	eで規定される(p v)の集合
	○	?	○	e,vで規定されるpの集合
*	○	○	?	e,pで規定されるv
	○	○	○	T or F

図17　アクセス関数

	E	P	V	Answer
SEARCHL	?	?	?	システム全体の情報　非現実的
	?	?	○	集合Vの要素で規定される I^E
	?	○	?	集合Pの要素で規定される I^P
	?	○	○	集合P,V(1対1対応)で規定される I^P
	○	?	?	集合Eの要素で規定される I^E
	○	?	○	集合E,Vの要素で規定される I^E
	○	○	?	集合E,Pの要素で規定される I^E
	◎	○	○	TorFを要素とする集合

SEARCH関数は、○印のところに、個体、属性、属性値の具体値、またはアクセス関数をinputして、?印のところに全称作用素にあたる"A?"、または存在作用素にあたる"E?"をinputして、?印部分の情報をoutputするものである。

SEARCHL関数は、○印のところに、個体、属性、属性値の具体値の集合、またはアクセス関数をinput、◎印のところに個体の具体値をinputして、?印のところに"A?"または"E?"をinputして、?印部分の情報をoutputするものである。

図18にアクセス関数の実行例、図19に*印のSEARCH関数のフローチャート、図20、21にLISPによるSEARCH関数のプログラムの一例を示す。

```
=SEARCH(DIONIS FATHER A?)
END OF EVALQUOTE, VALUE IS..
(ZEUS)

=SEARCH(ARES FATHER A?)
END OF EVALQUOTE, VALUE IS..
(ZEUS)

=SEARCH(ARES PARENT A?)
END OF EVALQUOTE, VALUE IS..
(HERA ZEUS)

=SEARCH(ARES PARENT E?)
END OF EVALQUOTE, VALUE IS..
(HERA)

=SEARCH(ARES ANCESTOR A?)
** G.C. HAS BEEN CALLED **
END OF EVALQUOTE, VALUE IS..
(HERA ZEUS)

=SEARCH(HARMONIA ANCESTOR A?)
** G.C. HAS BEEN CALLED **
END OF EVALQUOTE, VALUE IS..
(ROBINSON ARES HERA ZEUS)

=SEARCH(ARES MORTAL A?)
END OF EVALQUOTE, VALUE IS..
(T)

=SEARCH(PROG PARENT A?)
END OF EVALQUOTE, VALUE IS..
((PARENT (LAMBDA (E) (PROG (X ANS) (SETQ X (SEARCH6 E (QUOTE FATHER) (
  QUOTE E?))) (SETQ ANS (APPEND ANS X)) (SETQ X (SEARCH6 E (QUOTE MOTHER
  ) (QUOTE E?))) (COND ((NULL X) (RETURN ANS))) (RETURN (APPEND X ANS)))
  )))
```

図18　アクセス関数の実行例

```
>SEARCHL((ARES DIONIS) (SEX PARENT) A?)
 END OF EVALQUOTE, VALUE IS..
 ((ARES (SEX PARENT) ((MALE) (HERA ZEUS))) (DIONIS (SEX PARENT) ((MALE) (
   SEMELE ZEUS))))
```

```
>SEARCHL(A? (SEX FATHER) A?)
 END OF EVALQUOTE, VALUE IS..
 ((SEX (ARES HARMONIA DIONIS ROBINSON HERA ZEUS SEMELE) ((MALE) (FEMALE)
   (MALE) (FEMALE) (FEMALE) (MALE) (FEMALE))) (FATHER (ARES HARMONIA
   DIONIS) ((ZEUS) (ARES) (ZEUS))))
```

図18のつづき　アクセス関数の実行例

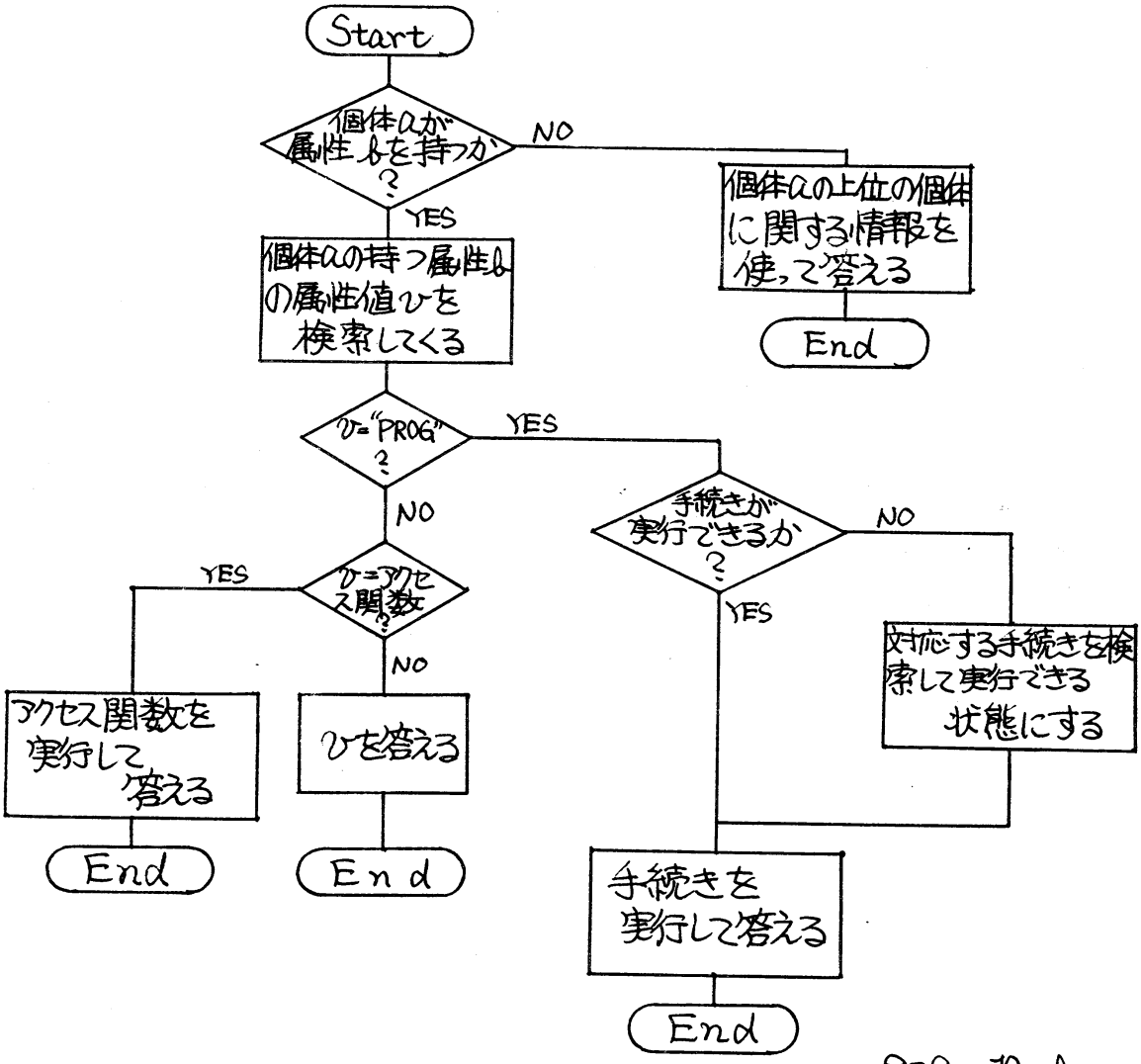


図19　*印のSEARCH関数のフローチャート　e=α, p=β とする

SEARCH(α β A?)

```
4450DEFINE(( (SEARCH (LAMBDA (E P V)
4460          (PROG (M2 M4 X)
4470          (COND ((NULL (ATOM E)) (SETQ E (EVAL E NIL))))
4480          (COND ((NULL (ATOM P)) (SETQ P (EVAL P NIL))))
4490          (COND ((ATOM E) (GO L1)))
4500          (COND ((ATOM P) (SETQ P (LIST P))))
4510          (COND ((ATOM V) (SETQ V (LIST V))))
4520          (RETURN (SEARCHL E P V))
4530      L1  (COND ((ATOM P) (GO L2)))
4540          (COND ((ATOM E) (SETQ E (LIST E))))
4550          (COND ((ATOM V) (SETQ V (LIST V))))
4560          (RETURN (SEARCHL E P V))
4570      L2  (SETQ M2 (QUOTE 2))
4580           (SETQ M4 (QUOTE 4))
4590          (SETQ X (QUOTE 0))
4600          (COND ((NULL (OR (EQUAL V (QUOTE A?))
4610                           (EQUAL V (QUOTE E?))))
4620                 (SETQ X (ADD1 X))))
4630          (COND ((NULL (OR (EQUAL P (QUOTE A?))
4640                           (EQUAL P (QUOTE E?))))
4650                 (SETQ X (PLUS X M2))))
4660          (COND ((NULL (OR (EQUAL E (QUOTE A?))
4670                           (EQUAL E (QUOTE E?))))
4680                 (SETQ X (PLUS X M4))))
4690          (COND ((EQUAL X (QUOTE 0))
4700                 (RETURN (QUOTE SYSTEM7ENTAI)))
4710                ((EQUAL X (QUOTE 1)) (RETURN (SEARCH1 E P V)))
4720                ((EQUAL X (QUOTE 2)) (RETURN (SEARCH2 E P V)))
4730                ((EQUAL X (QUOTE 3)) (RETURN (SEARCH3 E P V)))
4740                ((EQUAL X (QUOTE 4)) (RETURN (SEARCH4 E P V)))
4750                ((EQUAL X (QUOTE 5)) (RETURN (SEARCH5 E P V)))
4760                ((EQUAL X (QUOTE 6)) (RETURN (SEARCH6 E P V)))
4770                ((EQUAL X (QUOTE 7)) (RETURN (SEARCH7 E P V))) )
4780)))))
```

図20 SEARCH関数のメイン部分

```
530DEFINE(( (SEARCH6 (LAMBDA (E P V)
540          (PROG (X Y Z)
550           (SETQ X (RFILE E (QUOTE IES)))
560           (COND ((NULL (CDR X)) (RETURN NIL)))
570           (SETQ Y (CADR X))
580           (SETQ X (CADDR X))
590           (COND ((NULL (MEMBER P Y)) (GO L2)))
600           (SETQ X (PICKAT (ANUM P Y) X))
610           (COND ((EQUAL X (QUOTE PROG)) (GO L1)))
620           (COND ((EQUAL (CAR X) (QUOTE SEARCH)) (GO L6)))
630           (COND ((NULL X) (RETURN NIL))
640                 ((ATOM X) (SETQ X (LIST X)))
650                 ((EQ V (QUOTE A?)) (RETURN X)))
660           (RETURN (LIST (CAR X)))
670       L1(COND ((NULL (NULL (GET P (QUOTE EXPR))))
680                (GO L3)))
690           (SETQ X (CDR (RFILE (QUOTE PROG) (QUOTE IES))))
700           (SETQ Y (CAR X))
710           (COND ((NULL (MEMBER P Y)) (RETURN NIL)))
720           (SETQ X (CADR X))
730             (DEFINE (PICKAT (ANUM P Y) X ))
740         L3(SETQ X (EVAL (LIST P (LIST (QUOTE QUOTE) E)) NIL))
750           (COND ((NULL X) (RETURN NIL))
760                 ((ATOM X) (SETQ X (LIST X)))
770                 ((EQ V (QUOTE A?)) (RETURN X)))
780           (RETURN (LIST (CAR X)))
790       L2(SETQ X (CDR (RFILE E (QUOTE PFILE1))))
800           (COND ((NULL X) (RETURN NIL)))
810           (SETQ Y (CAR X))
820           (SETQ X (CADR X))
830       L5(COND ((NULL Y) (RETURN Z))
840               ((NULL (CAR X)) (GO L4))
850               ((EQ (CAR X) (QUOTE ALL))
860                (SETQ Z (SEARCH6 (CAR Y) P V)))
870               ((MEMBER P (CAR X))
880                (SETQ Z (SEARCH6 (CAR Y) P V))))
890           (COND ((NULL (NULL Z)) (RETURN Z)))
900       L4(SETQ Y (CDR Y))
910           (SETQ X (CDR X))
920           (GO L5)
930       L6(SETQ Y (EVAL X NIL))
940           (COND ((NULL Y) (RETURN X)))
950           (COND ((EQ V (QUOTE A?)) (RETURN Y)))
960           (RETURN (LIST (CAR Y)))
970 )))))
```

図21　*印部分のプログラム

7.2 エンドユーザランゲージの設計

現在、アクセス関数は、SEARCH、SEARCHLだけであるので、複雑な質問に答えることができない。従って、より複雑な質問に答えることができるようなエンドユーザランゲージを、以下のように設計した。これは、SEARCH、SEARCHL関数をサブルーチンとする、LISPによって記述された関数である。LISPの関数としてエンドユーザランゲージを設計すると、カッコが多くなって解りにくくなるが、構文解析などの複雑な処理を考慮しなくてよいという利点がある。

代表的なものを以下に説明し、それを用いた質問の記述例を示す。

(1) FIND関数　FIND(ae x f_n)

ae: "A" または "E" (全称、存在作用素)

x: 変数名

f_n: 条件関数

- f_nなる条件を満たす、すべての、または、あるxを求める。

(2) f_n関数

(i) WHERE(e p v)

e: 個体名またはFIND関数または変数名

p: 属性名またはFIND関数または変数名

y：属性値またはFIND関数または変数名

◦変数名にあたる部分の情報を求める。

(ii) WHEN(g_n)

g_n：条件関数

◦g_nなる条件を満たす情報を求める。

(3) g_n関数

(i) ALL(x y f_n)

x：変数名

y：個体名またはFIND関数

f_n：条件関数

◦yに含まれるすべてのxに対して、条件f_nを満たすならT、それ以外はNIL。

(ii) AV(x)　◦xの平均を求める。

(iii) GREATERP(x y)　◦$x>y$ならT、それ以外はNIL

(iv) EQUAL(x y)　◦$x=y$ならT、それ以外はNIL

etc.

◦アンダーソン氏のマネージャーのサラリーは？

```
FIND('E' x (WHERE (FIND 'E' y (WHERE 'ANDERSON'
                                'MGR' y ))
                  'SALARY' x))
```

◦平均サラリーが25000より多い部門をすべて求めよ。

```
FIND('A' x
     (WHERE (FIND 'E' y
       (WHEN (ALL y x
         (WHEN (GREATERP (AV (FIND 'A' z
           (WHERE y 'SALARY' z))) '25000')))))
            'DEPT' x))
```

図22　質問の記述例

7.3 知識とデータの複合処理

知識とデータは相対的なものであり、視点を変えると今まで知識として考えていたものをデータとして処理しなければならない状況が必ずある。また、本研究では、知識とデータを同一レベルで記述、処理しているので、知識とデータの複合処理機能が必要不可欠である。

そこで、図23の祖先関係における処理例を用いて、知識とデータの複合処理の一例を段階を追って説明する。

また、[STEP4]でのSCHKL関数とは、引数となる知識が他のどんな知識と接続しているかを調べるものである。ここで、この関数は、今まで知識として処理されてきたANCESTORに関する知識を、ただ単なる文字列として処理している。

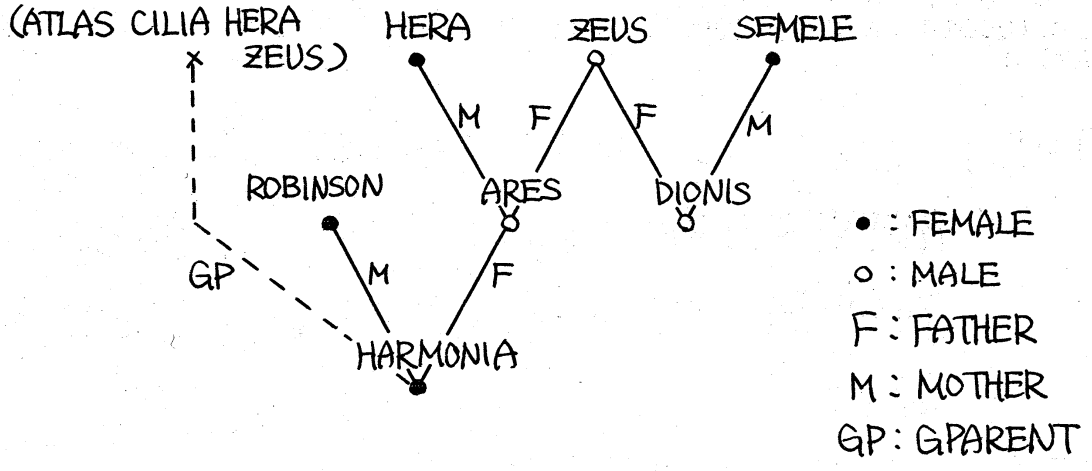


図23　祖先関係

〔STEP1〕 HARMONIAの先祖(ANCESTOR)を求める。

```
>SEARCH(HARMONIA ANCESTOR A?)
END OF EVALQUOTE, VALUE IS..
(ROBINSON ARES HERA ZEUS)
```

〔STEP2〕 そのうちHARMONIAの祖父母(GPARENT)は誰かを求める。

```
>SEARCH(HARMONIA GPARENT A?)
END OF EVALQUOTE, VALUE IS..
(ATLAS CILIA HERA ZEUS)
```

〔STEP3〕 かしこいユーザは、ANCESTORの知識が不十分であることに気付く。

〔STEP4〕 ANCESTORの知識は、他のどんな知識と結びついているのかを調べる。

```
>SCHKL(ANCESTOR)
(SEARCH E (QUOTE PARENT) (QUOTE A?))
(SEARCH (CAR X) (QUOTE ANCESTOR) (QUOTE A?))
END OF EVALQUOTE, VALUE IS..
((SEARCH E (QUOTE PARENT) (QUOTE A?)) (SEARCH (CAR X) (QUOTE ANCESTOR) (
  QUOTE A?)))
```

〔STEP5〕 かしこいユーザは、現在のANCESTORの知識は、PARENTを再帰的に求めていくもので、PARENTとG-PARENT間のパスがない場合は不適当であることに気付く。

〔STEP6〕 ANCESTORの知識を拡張(更新)する。

```
>SEARCH(HARMONIA ANCESTOR A?)
** G.C. HAS BEEN CALLED **
END OF EVALQUOTE, VALUE IS..
(ROBINSON ARES ATLAS CILIA HERA ZEUS)
```

```
((ANCESTOR (LAMBDA (E)
    (PROG (X ANS)
      (SETQ X (SEARCH E (QUOTE PARENT) (QUOTE A?) ))
      (SETQ ANS X)
    L1(COND ((NULL X) (RETURN ANS)))
     (SETQ ANS (APPEND ANS (SEARCH
             (CAR X) (QUOTE ANCESTOR) (QUOTE A?))))
      (SETQ X (CDR X))
      (GO L1) ))))
```

図24　拡張(更新)前のANCESTORの知識

```
((ANCESTOR (LAMBDA (E)
    (PROG (X ANS)
     (SETQ X (UNI
      (SEARCH E (QUOTE PARENT) (QUOTE A?))
      (SEARCH E (QUOTE GPARENT) (QUOTE A?)) ))
     (SETQ ANS X)
   L1(COND ((NULL X) (RETURN ANS)))
     (SETQ ANS (UNI ANS
      (SEARCH (CAR X) (QUOTE ANCESTOR) (QUOTE A?)) ))
     (SETQ X (CDR X))
     (GO L1) ))))
```

図25　拡張(更新)後のANCESTORの知識

以上まとめると、知識データベースシステムには、例えば、あるAという知識がBという知識を使用しているならば、Aという知識は使用せずにCという知識を使用して答を求めよ、といった知識とデータの複合処理機能が必要である。本研究で提案したシステムは、知識とデータを同一レベルで記述処理できるので、このような要求に、柔軟に、しかも十分に対応できるものと考えられる。

8．むすび

最後に、本研究で提案した拡張データベースシステムの特徴をまとめると以下のようになる。

（1）知識は必要時に、必要最小限のものだけが使用される。

（2）個々の知識は、他の知識とアクセス関数のみにより接続しているので、互いに独立している。

（3）システム全体の動作は、データの検索、知識の連鎖的、再帰的行動によって遂行される。

（4）他の知識データベースシステムとは違って、知識とデータの役割は認めるが、区別しないで同一レベルで記述、処理しているので、複雑な知識とデータの複合処理に柔軟に対応できる。

（5）従来のデータベースシステムで取り扱われている情報よりも広範囲の情報、つまり、外延情報の宣言的記述に加えて、内包情報や手続的記述情報を統合的に取り扱うことができる。

今後の課題としては、高速アクセス可能なファイル編成やLISPコンパイラの使用による効率面の改善、より大きな対象を用いた実験結果のデータモデルやシステム構成へのフィードバックなどが残されている。

<参考文献> 1. P. Lindgreen: "Basic Operations on Information as A Basis for Data Base Design" Proc. IFIP-74 (1974) 2. L. Kershberg: "A Taxonomy of Data Models" System for Large Data Bases (1976) 3. 野田: "外延・内包情報を取り扱える知識データベースシステムに関する研究" 阪大大学院通信特別研究報告 (1982)