ICOT Technical Memorandum: TM-0405

TM-0405

ロジック・データベースと知識ベース

宮崎収兄

October, 1987

ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03) 456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

ロジック・データベースと知識ベース

宮崎 収兄　（沖電気工業株式会社）

1．はじめに

ロジック・データベース，特にその特別な場合であるホーン・データベースの研究が盛んである。ロジック・データベースは，関係データベースの自然な拡張として，演繹推論機能を持ったデータベース・システムや，選言的データやナル値を扱うシステムについての，統一的な枠組みを与える。

一方，知識ベースは，データベースよりも高度な情報を格納するものと考えられる。知識ベースが大規模になり知識共有が必要な場合は，知識ベースの格納や維持，管理は，非常に複雑になると予想される。この問題の検討を行う上で，論理的基礎のしっかりした論理型知識表現が重要な役割を果たす。ロジック・データベースは，論理型で表現された知識ベースと見なすこともできる。

本稿では，ロジック・データベースと知識ベースの関係について論じ，関係データベースで開発された技術を応用して知識ベースを実現する上での諸問題について述べる。

2．ロジック・データベース

本節では，ロジック・データベースの概要と，その特別の場合である，関係データベースおよびホーン・データベース（確定演繹データベース）について述べる。

ロジック・データベースは，一階の述語論理式（well formed formula）の集合であり，データベースへの問い合わせも論理式で表現される。論理式は，定数と変数，関数をもとに構成される項を引数とする述語を，論理記号により結合した式である。論理式の変数には，全称作用素（∀）または存在作用素（∃）を付与することができる。限定作用素の付いた変数を束縛変数，付かない変数を自由変数という。自由変数のない論理式を閉じた論理式と呼ぶ。

ロジック・データベースは，以下のものから構成される。

(1) データベース：閉じた論理式の集合

(2) 問い合わせ言語；$x_1, \cdots, x_n$のみを自由変数とする論理式，$\psi(x_1, \cdots, x_n)$を考え，問い合わせを，$\{<x_1, \cdots, x_n> | \psi(x_1, \cdots, x_n)\}$ と表す。この問い合わせは，c_iを定数としたとき，$\psi(c_1, \cdots, c_n)$がデータベースから証明可能であるような，$c_1, \cdots, c_n$の値の組集合を求めることを意味している。問い合わせを，$\psi(x_1, \cdots, x_n)$と略記することがある。

厳密な定義には，一貫性制約と呼ばれる，データベースが満足すべき条件や，問い合わせの解が実際に値の組になるように保証するための型の概念の導入などの修正が必要であるが，ここではその詳細は議論しない。

上記の定義から分かるように，ロジック・データベースは，ロジック・プログラミングと共通の理論的基盤を持っている。

データベースでは，プログラムやユーザに共有される大容量の対象を扱うことに特徴がある。このため，問い合わせ処理以外に，一貫性制約，障害回復，同時実行制御，データ保護等多くの機能が必要となる。また，ロジック・プログラミングでは，個々の解を問題とするのに対し，データベースでは，解集合を問題とする。

ロジック・データベースの基本的性質としてデータベースと問い合わせの対称性がある。一般に，閉じた論理式QとFがあった時，⊢を証明可能を表わすとして，次式が成立する。

$$DB \cup \{F\} \vdash Q \Leftrightarrow DB \vdash Q \leftarrow F$$

したがって，データベースに論理式を格納することと，その論理式を問い合わせ中に書くこととは等価であり，同じ解を得ることができる。 この関係は，1つの論理式だけではなく，論理式の集合（意味的には連言）でも成り立つ。

ロジック・プログラミングの場合と同様に，データベースでも，一般の論理式を対象とするロジック・データベースに対する効率的処理方法は知られていない。したがって，効率的処理を実現するためには，データベース中の論理式の形を制限する必要がある。

論理式の制限を議論するためにいくつかの記法を導入しよう。 p を述語，t_iを項とした時，$p(t_1, \cdots, t_n)$を基本式，基本式およびその否定をリテラルと呼ぶ。A_iおよびB_iを基本式とした時，

$$\forall x_1, \cdots, \forall x_m (A_1 \vee \cdots \vee A_k \vee \sim B_1 \vee \cdots \vee B_n)$$

の形の論理式を節と呼ぶ。ここで，$x_1, \cdots, x_m$は式中の全ての変数である。節は，

$$\forall x_1, \cdots, \forall x_m (A_1 \vee \cdots \vee A_k \leftarrow B_1 \wedge \cdots \wedge B_n)$$

と書くこともでき，記法上，全称作用素を省略することが多い。"←"は右辺が真ならば左辺も真であることを意味している。また，←の左辺をヘッド，右辺をボディと呼ぶ。 データベース中の論理式を節に限定したものを，演繹データベースと呼ぶ。演繹データベースは，ロジック・データベースの特別の場合であるが，任意の閉じた論理式は，1つまたは複数の節に変換できることが知られているので，記述能力は実質的に同等である。一般の演繹データベースの研究も行われているが，実現方法の検討は，さらに制限を加えて行われることが多い。これを説明するため，まず節の分類を述べる。

(1) ボディが空の時，正の節と呼ぶ。正の節のヘッドが1つの基本式から構成される時，単位節，変数が含まれない時，グランド節と呼ぶ。

(2) ヘッドの基本式が1つの時，確定節，1つより多い時，不確定節と呼ぶ。また，0の時，ゴール節と呼ぶ。確定節およびゴール節をホーン節と呼ぶ。

既に実用化されている関係データベースはロジック・データベースの特別な場合である。また，関係データベースの拡張であり，知識ベースとの関連からも実用化が期待されるのがホーン・データベースである。演繹データベースの実現方式の研究は，ホーン・データベースを中心に行われているので，多くの文献で，演繹データベースは，ホーン・データベースを意味している。これらのデータベースを節の概念により定義しよう。

2．1．関係データベース

関係データベースは，次のようなロジック・データベースである。

(1) 論理式中に関数は現れない。
(2) データベース中の論理式は，グランド単位節のみである。
(3) 問い合わせは，関係論理式を用いて表される。

ここで，関係論理式は，論理式に以下のような制限を加えたものである。

(1) 定数は有限個しかない。
(2) 述語はデータベース中の単位節と同じか，算術比較述語である。

関係論理式には，∃や∀の限定作用素を用いてよい。また，節のように標準的な形でなくてよい。

ロジック・データベースの一種としての関係データベースは，上記のように定義されるが，通常の関係データベースの用語では，グランド単位節を組（タプル）と呼び，同じ述語名で引数も等しいグランド単位節の集合を関係（リレーション）と呼ぶ。

上記の関係論理式の定義は，定義域関係論理と呼ばれるものとほぼ同じである．問い合わせの表現としては，この他に，良く知られた組関係論理と関係代数等があり，この３者が同等の記述力を持つことが知られている[Ullman80]．

２．２．ホーン・データベース

ホーン・データベース（または，確定演繹データベース）は，データベース中の論理式を確定節に限定したものである．これに対し，不確定節（ヘッドの基本式が２つ以上ある節）を許容したものを不確定演繹データベースと呼ぶ[ＧＭＮ84]．演繹データベースと呼ばれるのは，データベースからその論理的帰結である問い合わせの解を求めることが，演繹推論に対応しているからである．これらのデータベースの概念的関係を図１に示す．

図１　ロジック・データベースの種類と関係

ホーン・データベースでは，関係データベースで用いられている関係演算を用いて，問い合わせ処理を行うことにより効率化を図っている．関係演算では，関数の処理が困難なので，ホーン・データベースでは，論理式を関数を含まないものに制限することが多い．

ホーン・データベースでは，データベース中のグランド単位節集合を外延データベース（Extensional Database：ＥＤＢ），それ以外の節集合を内包データベース（Intensional Database：ＩＤＢ）と呼ぶ．ＥＤＢは，関係データベースに対応している．ロジック・プログラムの用語に従って，ＥＤＢの節をファクト，ＩＤＢの節をルールと呼ぶことがある．

ホーン・データベースの問い合わせを，関係論理式や一般の論理式で表せば，ホーン・データベースが関係データベースの拡張になっていることは明らかである．理論的には，このような扱いが多いが，問い合わせ処理等の実現方式の検討では，問い合わせの形式をさらに制限することが多い．また，否定を扱うために，ホーン節の拡張を行うことも多い．

この問題を論じる前に，簡単な例でロジック・プログラミングとデータベースの関係を考察しよう．

例１．祖父母－孫の関係

一階述語論理で，親子関係をparentで記述する．

```
parent(太郎, 二郎)←
parent(花子, 二郎)←
parent(二郎, 三郎)←
```

祖父母と孫の関係は，

```
grand_parent(X, Y)
    ←parent(X, Z), parent(Z, Y).
```

ここで，grand_parent(太郎, Y)を真とするＹの値を求めたいとする．ロジック・プログラミングでは，この問題を証明問題として扱う．

証明は，まず証明したい論理式を否定し，その論理式と与えられた論理式集合を合わせた集合が矛盾することを示すことにより行う．すなわち～(∀Ｙ(grand_parent(太郎, Ｙ))をゴールとし，導出原理を用いてこのゴールの反駁を行う．この結果Ｙ=三郎が求まる．Ｙの値を全て求めたければ，この過程を継続すれば良い．

この問題を関係データベースで扱う場合，グランド単位節はデータベースに格納される．問い合わせは

{<Y>| ∃Z(parent(太郎, Z) ∧parent(Z, Y)}

と書かれる。

ホーン・データベースでは，この問題を幾つかの方法で扱うことができる。1つは，ＩＤＢに上記のgrand_parentの定義節を格納し，問い合わせを{<Y>|grand_parent(太郎, Y)}とすることである。

ＩＤＢにgrand_parentの定義がない場合は，問い合わせを２つの方法で表現できる。1つは関係データベースと同じ表現である。もう1つは問い合わせとデータベースの対称性を用いて，問い合わせ中にgrand_parentの定義を書き，

{<Y>|grand_parent(太郎, Y)←
(∀V∀W∀Z(grand_parent(V, W)←
parent(V, Z), parent(Z, W)))}

と表すことである。なお，関係データベースでの表現はこの表現から導出原理と同様にして述語grand_parentを消去することによりえられる。また，関係データベースにおけるビューの導入は再帰的または相互依存的なビューがないという制限のもとでＩＤＢを導入することに相当している。

この例では，関係データベースでも記述できたが，後述の先祖関係のように再帰的な場合は記述できない。ホーン・データベースでは，ＩＤＢまたは問い合わせに，先祖関係が定義できる。　このような性質から，問い合わせに用いる論理式を上記のような形に制限すると，ホーン・データベースのＩＤＢの記述力と問い合わせの記述力がバランスし，しかも，ホーン節を用いたロジック・プログラミングの記法とうまく対応していることが分かる。このため，多くの文献で，ホーン・データベースの問い合わせを基本式で記述し問い合わせの内容はホーン節で記述している。

ホーン・データベースにおいて，問い合わせの記述力をホーン節で記述できるものと同等なものに限定すると関係データベースで記述できる否定の記述ができない。これは，Prolog 等のロジック・プログラミング言語の場合と同様である。したがって，否定は，ロジック・プログラミングの場合と同様に，データベースに記述の無い情報は偽であると考える閉世界仮説のもとで，肯定が証明できない時は否定が証明されたと考えるnegation as failureの概念によって扱う。このような拡張を行い，ホーン節のボデイに，not述語または負リテラルを許すことにより，ホーン・データベースを関係データベースの拡張とすることができる。

以上のようなホーン・データベースの概念は，ホーン節によるロジック・プログラムの概念と密接な関係がある。事実，上記のような概略的定義では両者に大きな差は無い。ホーン・データベースが注目されるのは，そのデータベースとしての側面である。すなわち，ホーン・データベースが関係データベースの拡張であることから，

(1) 一貫性制約の概念の導入
(2) データベース分野で蓄積された，大容量データに対する効率的処理方式の適用
(3) 共有データベース管理のために開発された，障害回復，同時実行制御，データ保護等の技法の適用
(4) データベースへのルール型知識の格納による演繹推論機能の実現等により，大容量の共有型知識ベース管理技術の基盤になることが期待できる，

等によりデータベース技術をＡＩ分野に適用し新しい応用を可能にすると期待されているためである。

３．データベースから知識ベースへ

知識ベースの概念は，エキスパート・システムにおける知識の集合として導入されたが，ここではもっと広く考え，知識を処理/利用するシステムの基盤となる知識集合と考える。

現在のエキスパート・システム等では，知識の量は少ないが，将来の知識情報システムでは，大量の共有知識の管理が必要になると考えられる。このため，知識ベース管理システム，すなわち，

"知識情報処理システムのための大きな
共有知識ベースを効率良く管理するシステム"

の研究が重要になっている。

知識ベースにおける知識の表現方法としては，

(1) 述語論理
(2) 意味ネットワーク
(3) フレーム
(4) プロダクション・システム

等が提案されている。

知識には，事物の性質や相互の関連や事象を表す宣言的知識と，手続き的知識とがある。上記の知識表現のうち，(1)～(3)は宣言的知識の表現に適していると考えられる。Prologのようなロジック・プログラミング言語では，その処理を手続きと考えることができるので，宣言的知識だけでなく，手続き的知識をも表すとみなすことができる。

知識ベースの管理においては，知識の獲得と利用（検索）の２つの機能が必要である。知識の獲得にあたっては，知識ベース全体の整合性が問題となる。知識表現の内，知識ベース全体の整合性を検討するための理論的基礎がはっきりしているのは述語論理である。このため，述語論理を基礎とした知識表現の検討がされるとともに他の知識表現との関係も論じられている。

例えば，意味ネットワークでの表現例としてis-aリンクが良く用いられる（図２）。

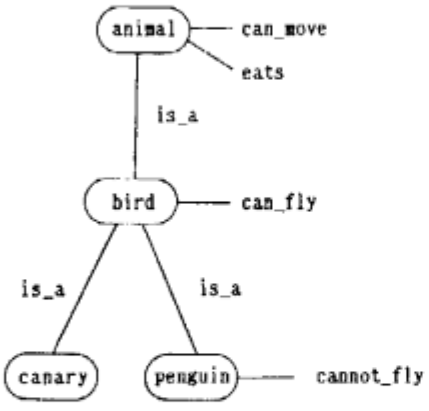


図２　is_aリンクによる意味ネットワーク

図２で，is_aは動物間の概念的関連を表しており，can_moveやeatsはその性質を表している。これをフレームで表すと，animalやbirdが各々フレームになり，それらの性質はフレーム内のスロットで記述される。フレーム表現でis_aリンクに相当するのは，フレーム間の関係を示す上位概念，下位概念の記述である。論理型表現の方法は幾つか考えられるが，ロジック・プログラミングのテキスト等に良く現れる方法でその一部を表すと以下のようになる。

```
can_move(X) ← animal(X).
animal(X) ← bird(X).
bird(X) ← canary(X).
```

ここで，canaryは個体ではなく集合とみなされている。ある個体aがcanaryであることはcanary(a)と表す。

知識型システムは，知識ベースと推論機構を主要な構成要素としている。推論は人口知能技術の基礎の１つである。図２の例で，canaryはis_aリンクによりbirdの性質やanimal の性質を継承するが，is_aリンクをたどって上位の性質を捜すのは論理表現からも分かるように推論に対応している。

論理型知識表現はかなりの記述力を持つが，それだけでは不十分であると言われている。このため，高階論理の導入やメタレベル，非単調論理への拡張等が論じられている。

知識ベース管理システムは強力な知識表現とともに，その検索（または，それを用いた推論）が効率的であることが必要である。さらに，知識ベース管理機能としては，知識ベースの整合性の管理や障害対策等が必要である。このような知識ベース管理システムを実現する上で，これら機能を限定された形だが備えているロジック・データベースが重要な役割を果たすと期待される。すなわち，データベース管理システムは，

"情報処理システムのための大きな
共有データベースを効率良く管理するシステム"

であり知識とデータの違いを除けば知識ベース管理システムに対する要求条件を満たしている。このため論理型知識表現のもとでロジック・データベース特にホーン・

データベースが知識ベースの核になると期待される。

ホーン・データベースを知識ベースの核として用いる場合に，ＥＤＢが大量な場合には，関係データベース技術の応用によって効率的な処理が可能である。これに対して，ＩＤＢが大量な場合の効率的処理は，並列処理等のアーキテクチャ的な工夫が無いと困難である。このため，ホーン・データベース・システムは，ＩＤＢの量はそれほど大でないが，ＥＤＢは大であるようなデータベースの処理に適していると考えられる。これは，Prologプログラムで，通常ルール数がファクト数よりかなり大であることと大きく異なっている。

ホーン・データベースを知識ベースの核として用いる場合，知識ベース内でＥＤＢが大でＩＤＢが小であるようなものでないと効率的ではない。このことから，ホーン・データベースにおける知識表現は大きく制約される。図２のような知識を，ロジック・プログラミングのテキストにあるような方法で，述語論理を用いて表現すると，ＩＤＢの量は大でＥＤＢが小となり効率的な処理ができない。述語論理を基礎とした表現が幾つか提案されているが，多くの場合ＩＤＢの量が大となっている。一般に，論理型知識表現でルール型知識が多いのだとすれば，アーキテクチャ的な工夫を行わないと，ホーン・データベースは知識ベースとしてはあまり有効ではなく，データベースの高機能化としての意味しか持たないことになる。

この問題は，知識表現のシンタックスを変更しなくとも，表現のスタイルまたはパラダイムを変更することにより解決できる。先の例では，ルールがis_aリンクと事物の性質とを表していた。この表現が有効なのは，ルールの背景に述語論理の推論規則があるからである。したがって，推論規則を陽に表現すればリンクをファクトで表すことができる。図２の例では，

```
is_a(X, Y) :-
    is_a_fact(X, Y).
is_a(X, Y) :-
    is_a_fact(X, Z), is_a(Z, Y).
```

のように推論規則を陽に表現すれば，is_aリンクは，

```
is_a_fact(bird, animal).
is_a_fact(canary, bird).
```

のようにファクトで表現できる。性質に関する知識に関しても同様である。我々はこのような表現を，ＫＲＢＦ（Knowledge Representation By Fact）と呼んでいる[宮崎87a]。

ＫＲＢＦを用いて知識の表現を行えば，個別的知識はファクトで，一般的知識をルールで表現することになる。したがって，大部分の知識はファクトで表現でき，ロジック・データベースで効率的処理ができる。ＫＲＢＦがどのような知識に適用できるか十分には分かっていないが，例外の記述等，非単調論理に準じる表現も，negation as failureを意味するnot述語を用いて行うことができる。

４．実現技術

論理型知識表現に基づく，知識ベースとロジック・データベースとは，密接な関係にある。ロジック・データベースは，知識ベースの実現方法の１つと言っても良い。ロジック・データベース・システムの技術は，関係データベース・システムやロジック・プログラム言語処理系を参考としながら研究開発が行われている。

ロジック・データベース・システムの研究は，当初は関係データベース処理を，定理証明系で実現したり，Prologと関係データベース・システムを結合したりする構成のものが多かった。しかし，このように従来の方法やアルゴリズムをそのまま適用するのでは効率的な処理を実現するのは困難であるため，新しいアルゴリズムの開発に重点が移行している。

４．１．システム・モデル

ロジック・データベース，特にホーン・データベースでの検索処理は，大別して２通りの実現方法がある。１つは導出原理に基づいた定理証明系のアルゴリズムを使

う方法であり，もう1つは関係データベース・システムで開発された方法を用いることである．後者の方法は，例えば，関係代数のような演算を用いて実現する．

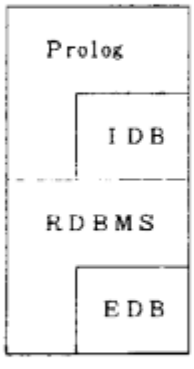
例1のgrand_parent(太郎, Y)の例で考えてみよう．この問い合わせの解は，Prologのような定理証明系では，←grand_parent(太郎, Y)を反駁する方法で得ることができる．この方法では，ゴールから出発しそれを解くために必要なサブゴールを順次解いていく．この問い合わせは，関係代数では，

$$Ans = \pi_4((\sigma_{1=}\text{太郎}(parent)) \bowtie_{2=1} parent)$$

と書くことができる．ここで，πは射影，σは選択，$\bowtie$は結合を表している．したがって，この問い合わせは，まずparentから第1属性が太郎である組集合を選択し，その結果をparentと結合し，その結果から必要な属性を取り出すことによって解を求めることができる．

大容量のデータベースでは，二次記憶への高速アクセスや集合演算としての効率化技法，問い合わせの形態に応じての最適化処理等を利用できる関係演算系の方が，定理証明系よりも効率的な処理ができる．これは，例えば，例1で定数の位置を変えた問い合わせを考えた場合，Prologの処理が非効率的なことからも分かる．

このため，Prologと関係データベース管理システム（ＲＤＢＭＳ）を結合することにより，ホーン・データベースを実現することができる（図3）．



Prologプログラム：
利用者プログラム
＋ＩＤＢ
ＩＤＢ：内包データベース
ＥＤＢ：外延データベース
ＲＤＢＭＳ：
関係データベース
管理システム

図3　ホーン・データベース・システムの構成1
PrologとＲＤＢＭＳの結合

このシステムでは，ＩＤＢの処理はPrologで行い，ＥＤＢの処理はＲＤＢＭＳで行う．この時，ＥＤＢへのアクセスが発生する毎にＲＤＢＭＳをコールしたのでは，全体として定理証明系と同じ動作になり効率が悪いので，遅延評価法等によりＥＤＢへのアクセスをまとめて行い効率化を図ることが試みられた　[Yokota84]．

ＩＤＢの処理をPrologで行う方法では，利用者プログラムとＩＤＢが未分化であることや，効率および停止性の点で問題があるため，新しいシステム構成の検討が行われた．主な構成は次の2つである．

(1) ＩＤＢを管理する部分を分離しＲＤＢＭＳの処理方式はそのまま利用する（図4）
(2) ＲＤＢＭＳのアルゴリズムを修正し，ホーン・データベースの処理を可能とする（図5）

図4　ホーン・データベース・システムの構成2
階層型システム

図4のような階層型の構成の例としては，ＫＢＭＳ ＰＨＩ［羽生田87］がある．また，多くの文献で演繹データベース・システムは図4またはその変形で，知識管理がなく問い合わせの表記のみを再帰問い合わせが可能なように拡張したものとして議論している．図5の例としては，単一化検索（ＲＢＵ）にもとづいたシステム

図５　ホーン・データベース・システムの構成３
統合型システム

がある[Yokota86]．ＲＢＵは，知識ベース・マシンの核部分の処理方式として提案された並列型アーキテクチャ用演算であり，関数の処理も可能なように関係演算を拡張しておりホーン・データベースの処理も実現できる．この方式では，前節の議論とはことなってＩＤＢが大の場合にも効率の良い処理を行えるような工夫がなされている．

４．２．問い合わせ処理

ホーン・データベースにおける問い合わせ処理が，関係データベースの処理と大きく異なるのは，再帰的問い合わせ処理が必要なことである．もう１つは，関数の存在であるが，多くのホーン・データベースでは，関数の無い形に制限している．関数の無いホーン節を用いた言語をDatalogと呼ぶ．本節では，ＩＤＢが比較的小さくＩＤＢ部分を問い合わせの一部とみなしてもよいような場合の問い合わせ処理を議論する．

問い合わせ中の確定節のヘッドの述語pはボディ中の述語qに依存していると考えることができる．この関係をp→qと書く．依存関係は推移的であるので，推移閉包を考えることができる．qがpの推移閉包の要素である時p→*qと記述する．問い合わせが再帰的であるとは，p→*pなる述語が存在することである．

関係データベースでの問い合わせは，関係代数で表現できる．関数の無いホーン・データベースでは，関係代数を拡張して左辺の関係が右辺にも現れることを許せば再帰的な問い合わせも表すことができる．例えば，先祖関係はホーン節では，parentがＥＤＢであるとして，

```
ancestor(X, Y) :- parent(X, Y).
ancestor(X, Y) :-
    parent(X, Z), ancestor(Z, Y).
```

である．これを（拡張した）関係代数で表すと，

$$ancestor = parent \cup \pi_{1,4}(parent \bowtie ancestor)$$

となる．一般のDatalogの問い合わせは，１つの関係代数式では表せず，再帰述語の各々に式が必要である．関係代数で表した時，ＥＤＢに現れる以外の述語に対応する関係を仮想関係と呼ぶ．

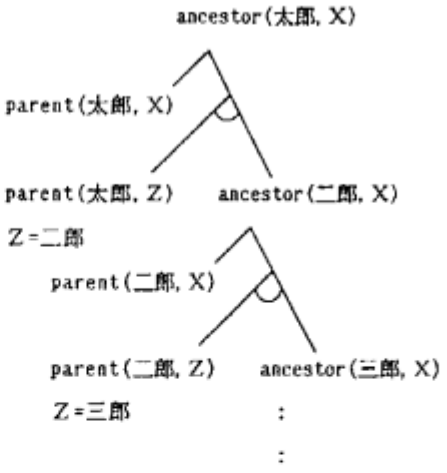


図６　Prologによるancestor(太郎, X)の計算

ホーン・データベースの処理方式は，大別してトップダウンとボトムアップの２種類に分けられる[Bancilhon 86a]．トップダウン方式は，Prologと同様に，問い合わせを解くのに必要な部分問い合わせを順次発行してい

く方式である．これに対し，ボトムアップ方式では，ＥＤＢの側から順次計算していく．ancestor(太郎, X)のトップダウン処理の例を図５に示す．

非再帰型問い合わせを，ボトムアップ処理で，ＥＤＢの述語から順次計算することは容易である．しかし，再帰型問い合わせでは，仮想関係の値が定まっているわけではないので容易ではなく，例えば，次のように繰り返し処理が必要となる．

ancestorのような関係代数式はＥＤＢの関係を定数と考え，仮想関係を変数として，

$$a = f(a)$$

のような形をしている．このような式を満足する関係を，fの不動点と呼ぶ．したがって，問い合わせの解は不動点を求めることに対応している．不動点（正確には最小不動点）は，次のような列の極限（$a=\lim_i a_i$）となっている．

$$a_0 = \{\}$$
$$a_{i+1} = f(a_i)$$

このような列は（否定の無い場合には）必ず有限回で収束するので，解を繰り返し処理によって求めることができる．このような処理による方法はナイーブ評価法や最小不動点（lfp）演算法と呼ばれている．これらの方法は，限定付だが否定のある場合にも適用できる．

ホーン・データベースの研究の初期には，トップダウン系のアルゴリズムの研究が多かったが，最近はボトムアップ系アルゴリズムの研究も盛んである．ボトムアップ方式は，問い合わせ中の定数をうまく伝播して，必要な情報のみをアクセスするようにするという効率化を行うのに適しているのに対し，ボトムアップ方式は関係データベースの処理をうまく使って効率化するのに適している．これをancestorの例で考えてみよう．

ancestor(太郎, Y)の計算は，Prologでは，図６のように行われるが，この計算の部分問い合わせは，常にancestor(定数, 変数)の形をしており，無駄な計算は行わない．また，parentについても，常にparent(定数, 変数)の形をしており効率的処理ができる．ただし，ancestor(X, 二郎) のような場合，parentへのアクセスがparent(変数, 変数)の形になるため効率は落ちる．

これに対し，lfp演算法ではancestor全体を計算してからその一部を取り出す．したがって，個々の計算は効率的に行えても，全体として無駄が多く効率が落ちてしまう．この点を改良するため，ボトムアップ方式で定数伝播をうまく行う方法が各種提案されている．例えば，ancestor(X, 二郎)のような問い合わせでは，

```
ancestor(X, 二郎)←
    parent(X, Z), ancestor(Z, 二郎)
```

なので，ancestor(変数, 二郎)をlfp演算法で直接計算できる．しかし，ancestor(太郎, Y)のような場合はこれができない．

トップダウン法とボトムアップ法にはそれぞれ一長一短があり，一方だけでは真の効率化が図れない可能性があるため，最近では両方の要素を取り入れた方式も提案されている．例えば，ボトムアップを基本とした方式としては，a=f(a)を計算するのではなく，新しく述語（仮想関係）a*を導入し，a=a*∩f(a)のようにしa*をうまく求められば効率化を図ることができる．このような考えに基づいた方法にマジック集合法およびその拡張や制約lfpによる方法がある[Bancilhon86b, Beeri87, 宮崎87b]．トップダウンに基づく方法でも同様な考えの導入が行われている[Vieille86]．これらの方式により，一般の問い合わせの効率的処理方法が確立しつつある．

４．３　データベースの更新

ホーン・データベースの内容の更新を行うとき，通常のデータベースの場合と同様に次のような機能が必要になる．

(1) データベースの満たすべき制約条件の保証
(2) 複数ユーザからの同時アクセスに対する整合性の保証

(3) 障害時のリカバリ

これらの処理は関係データベースなどでは以前から研究が行われている。しかしホーン・データベースではその処理方式がまだ十分には確立していない。これらの内(1)については一貫性制約の拡張[GMN84]や無矛盾性の保証，冗長情報の除去[北上85]などの研究が行われているが，大規模のホーン・データベースでこれらの処理を効率良く行うためには解決すべき問題が多い。

更新問題のうち，(2)と(3)についてはルールの更新頻度などの条件が不明なためもあり，関係データベースで開発された技術の直接的適用以外には，まだあまり研究が進んでいない。

5．まとめ

ロジック・データベースの実現上の諸問題について，関係データベースからホーン・データベースへの拡張という点を中心に議論した。また，ロジック・データベースをもとにした知識ベースの実現という視点から，その技術課題について論じた。ここで述べた問題以外にも手続き的知識の扱いや複雑な構造的知識の表現などの課題がある。

本稿で述べたような知識ベース管理システムの研究は，日本では第５世代コンピュータ・プロジェクトなどで行われており，将来の知識情報処理システムの基礎となることが期待されている。

≪参考文献≫

[Bancilhon86a]F. Bancilhon, et. al., An Amateur's Introduction to Recursive Query Processing, ACM SIGMOD, 1986.

[Bancilhon86b]F. Bancilhon, et. al., Magic Sets and Other Strange Ways to Implement Logic Programs, ACM PODS, 1986.

[Beeri87]C. Beeri, et. al., On the Power of Magic, ACM PODS, 1987.

[GMN84]H. Gallaire, et. al, Logic and Data Bases; A Deductive Approach, ACM Computing Surveys, June 1984.

[羽生田87]羽生田 他，分散知識ベース管理システム PHI，情報処理学会データベースシステム研究会，DBS61-2，1987.

[宮崎87a]宮崎 他，KRMS PHI:例外のある性質継承の表現，34回情報処理学会全国大会，1987.

[宮崎87a]宮崎 他，演繹データベースにおける制約付最小不動点，ICOT TR，1987.

[Ullman80]J. D. Ullman, Principles of Databaase Systems, Computer Science Press, 1980.

[Vieille86]L. Vieille, Recursive Axioms in Deductive Databases: The Query/Subquery Approach, Intl. Conf. on EDS, 1986.

[横田84]H. Yokota, et. al., An Enhanced Inference Mechanism for Generating Relational Algebra Queries, ACM PODS, 1984.

[横田84]H. Yokota, et. al., A Model and an Architecture for a Relational Knowledge Base, Symp. on Computer Architecture, 1986.