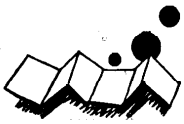
解　説



演繹データベースにおける再帰的な問い合わせの評価法†

西尾章治郎††　楠見雄規†††

1. はじめに

演繹データベース (deductive database) とは，事実 (fact) の集合を格納する**外延データベース** (extensional database : EDB) と，知識の集合（物事の性質を記述した**ルール**の集合として扱う）を格納する**内包データベース** (intensional database : IDB) からなるシステムであり，特に最近，人工知能 (AI) の分野における知識ベースの核としての機能を果たしている．演繹データベースの研究は，1970年代後半から盛んに行われており，現在も新たな成果が次々と報告されている．構文論的な研究については，1970年代後半から1980年代初頭に Minker, Gallaire らによって多くの成果[7),8)]が報告された．近年になって，計算機の容量，速度が著しく増大したのにともない，内外の第五世代コンピュータプロジェクトを中心に実際の演繹データベースシステムの実現のために，演繹データベースにおける問い合わせ評価アルゴリズム，特に，本稿で対象とする**再帰的な問い合わせ** (recursive query) の問題に多くの関心が払われるようになった．

関係データベースのように外延データベースだけからなるシステムでは，たとえば，データの集合として「親子関係」のデータが記憶されている場合，すべての「先祖関係」を求める問い合わせの陽な表現は困難である．なぜなら，関係データベースの数学的基礎である**関係代数** (relational algebra)[17)]を用いて，「先祖関係」を表現する式を記述するとその長さが無限になってしまうからである．ただし，既存の関係データベースソフトウェアのなかには，たとえば ORACLE の問い合わせ言語のように，一つの関係上に“親”と“子”の関係をもつ二つの属性がある場合には，ある値の“先祖”または“子孫”にあたるデータを求める手続きをサポートしているものもある．一方，演繹データベースでは，外延データベースに格納される「親子関係」のデータ以外に，「ある人の親は，先祖である」，また，「ある人の親の先祖は，先祖である」という一般的なルールが内包データベースに格納される．これら外延，内包データベースをもとに，「ある人の先祖を求めよ」という問い合わせに答えることができる．単に先祖関係に留まらず，演繹データベースでは，より複雑な再帰構造で定義される関係についてのデータの検索が可能である．ところが，先祖を規定するのに先祖が用いられるように，ルールが再帰的になっている場合，一般に，問い合わせに答えるために必要な中間結果として得られる関係（以後，**中間関係**という）が爆発的に大きくなる場合がある．そこで最近，与えられた問い合わせに含まれている明示的な情報（たとえば，「太郎の先祖を求めよ」という問い合わせでは，「太郎」という具体的な人名）をもとに，ルールの再帰構造，事実のデータの構造を生かして，いかに効率良く問い合わせに答えるかという研究が盛んである．現在までに，Shapiro, Henschen, Naqvi, Lozinskii, Ullman, Vieille, Zaniolo らが多くの成果を発表しており，彼らの研究の要約として文献 4) がある（日本の ICOT を中心にした研究として，たとえば文献 13) がある）．

本稿では，演繹データベースにおける再帰的な問い合わせの問題，および，その問い合わせを評価する種種のアルゴリズムについて解説する．解説は分かりやすさを旨としたので，詳細については各アルゴリズムについてあげた参考文献を参照されたい．

2. 演繹データベースにおける再帰問い合わせ

2.1 演繹データベース

演繹データベースは，上記のように EDB と IDB

† Evaluation Methods for Recursive Queries in Deductive Databases by Shojiro NISHIO (Department of Applied Mathematics and Physics, Faculty of Engineering, Kyoto University) and Yuki KUSUMI (Division of Applied Systems Science, Graduate School of Engineering, Kyoto University).
†† 京都大学工学部数理工学教室
††† 京都大学工学部応用システム科学教室

から構成される．EDB または IDB に格納された事実または知識の集合は，**一階述語論理式** (first order predicate logic formula) の集合，すなわち，**論理プログラム** (logic program)[12] で表現される．本稿では，一階述語論理式のクラスを特に**ホーン節** (Horn clause)[12] に限定した演繹データベースである，**ホーンデータベース**について述べる．

ホーンデータベースでは，次の形の述語論理式がデータベース格納される．

$$p(X_1, \cdots, X_k) \leftarrow q_1(Y_1^1, \cdots, Y_{n_1}^1), \cdots, q_m(Y_1^m, \cdots, Y_{n_m}^m). \quad (1)$$

(1)式の両辺に現れる $r(Z_1, \cdots, Z_m)$ という形の式は，**正リテラル**と呼ばれ，述語 r と引数 Z_i から構成される．各引数は，変数（個体変項），定数（個体定項），関数のいずれかであるが，本稿では関数引数のない場合について議論する．特に，(1)式で表される一階述語論理式はホーン節と呼ばれ，高々1個の正リテラルからなる頭部（左辺）と，正リテラルの連言からなる本体（右辺）から構成される．本体と頭部の間の記号"←"は論理式の含意を意味し，(1)式は全体として，「本体の述語がすべて真ならば，頭部の述語が真」という意味をもつ．以下本稿では，変数は，大文字で始まる英数文字列で，定数は，小文字で始まる英数文字列または日本語を用いて表す．

ホーン節のうち，本体の述語がなく，引数が定数のみの節を，特に**正の基底節** (positive ground clause) という．EDB は，正の基底節の集合であり，従来の関係データベースの関係と本質的に同じものである．たとえば，「Y が X の親である」という事実を表す述語を par (X, Y) とすると，「勘吉は仁の親である」という事実は，頭部のみからなる節で

par (仁, 勘吉) ←. (2)

と表現され，EDB に格納される．式(2)の par のように EDB に格納される述語を，特に **EDB 述語**と呼ぶ．

一方，この EDB 述語 par を用いて，「X の親が Z であり，その親が Y ならば，Y が X の祖父母である」という家系に関する性質を，二項述語 grand_par を導入して

grand_par $(X, Y) \leftarrow$ par(X, Z), par (Z, Y). (3)

と表現できる．IDB に格納される(3)式のような節をルールと呼び，grand_par のように IDB 中のルールを定義するために導入された述語を **IDB 述語**と呼ぶ．

［例1］ IDB には(3)式のルールが格納され，EDB

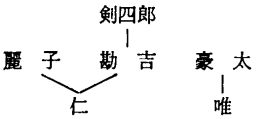


図-1 (4)式のデータ構造

には事実の集合として，たとえば，

par (勘吉, 剣四郎) ←.
par (仁, 麗子)←.
par (仁, 勘吉)←.
par (唯, 豪太)←. (4)

の4つの**組** (tuple) が格納されている演繹データベースを考える(**図-1** 参照，ただし，名前の字数が多いほど古い世代を表している)．このとき，(3)式で定義される述語 grand_par を用いて，「仁の祖父母 X を求めよ」という問い合わせは次式のように書くことができ，答えは，query (X) の X に代入される．

query(X)←grand_par (仁, X).

この問い合わせに答えるためには，リテラル grand_par (仁, X) を評価せねばならない．このような評価すべきリテラルを**ゴール** (goal) と呼ぶ．grand_par (仁, X) のように問い合わせに現れるゴールを，特に，**問い合わせゴール**と呼ぶ．問い合わせゴール grand_par (仁, X) を評価するためには，(3)式の本体の二つの述語 par (X, Z) と par (Z, Y) を評価せねばならない．このようにルールの本体に現れるゴールを，特に，**サブゴール** (subgoal) と呼ぶ．

ゴール中の引数に，定数が割り当てられているとき，その引数は**束縛** (bind) されているといい，そうでない場合，引数が**自由** (free) であるという．多くの場合，引数にどのような値が割り当てられているかよりも，どの引数が束縛されているかが問い合わせ評価の上で問題となる．

ところで，上記の問い合わせは次のように関係代数式を用いても簡単に表現できる．

$$\sigma_{1=仁}\{\pi_{1,4}(\mathrm{par} \underset{2=3}{\bowtie} \mathrm{par})\}$$

ここで，σ は選択演算，π は射影演算，$\bowtie$ は結合演算を表す．この関係代数式の評価は，「EDB から par の組をすべて取り出し，それらを等結合し，その結果を第1属性と第4属性からなる二項関係に射影したものの第1属性が"仁"であるものを選択する」ということである．このように考えると，関係データベースの問い合わせ評価の原則である「結合演算の前に選択を行う」ことは，サブゴール中の引数をできるだけ束縛された状態で，ルール本体の評価を行うことと同じであることが分かる．■

2.2 再帰問い合わせ

2.1 までにみてきた論理プログラムは，従来の関係代数式で記述できる（つまり，関係データベースのビューで扱える）範囲内のものであった．しかし，述語論理式がホーン節形式であるという条件のもとで，EDB 述語 par を用いたより複雑なルールが考えられる．次の二つのルールは，「Y が X の親であるか，または，X の親 Z の先祖であれば Y は X の先祖である」という性質によって二項述語 ancestor (X, Y) を定義する．

$$\begin{aligned}&\text{ancestor}(X, Y)\leftarrow \text{par}(X, Y).\\&\text{ancestor}(X, Y)\leftarrow \text{par}(X, Z),\ \text{ancestor}(Z, Y).\end{aligned}\qquad(5)$$

2番目のルールでは，述語 ancestor を定義するのに述語 ancestor 自身を用いていることから述語 ancestor は再帰的に定義されているといい，このようなルールを**再帰ルール** (recursive rule) と呼ぶ．また，(5)式の論理プログラムにおいて，1番目のルールは停止条件にあたる．これらのことから，二つのルールからなる論理プログラムは，再帰構造を許さないと

$$\begin{aligned}&\text{ancestor}(X, Y)\leftarrow \text{par}(X, Y).\\&\text{ancestor}(X, Y)\leftarrow \text{par}(X, Z),\ \text{par}(Z, Y).\\&\text{ancestor}(X, Y)\leftarrow \text{par}(X, Z_1),\ \text{par}(Z_1, Z_2),\\&\qquad\qquad\qquad \text{par}(Z_2, Y).\\&\text{ancestor}(X, Y)\leftarrow \text{par}(X, Z_1),\ \text{par}(Z_1, Z_2),\\&\qquad\qquad\qquad \text{par}(Z_2, Z_3),\ \text{par}(Z_3, Y).\\&\cdots\cdots\end{aligned}\qquad(6)$$

という無限個の非再帰ルールに展開したのと同じ意味になり，非再帰ルールでは表現が不可能である．

［例2］ 例1の EDB と(5)式の論理プログラムからなる IDB に対して，「仁の先祖 X を求めよ」という問い合わせ，

$$\text{query}(X)\leftarrow \text{ancestor}(仁, X).\qquad(7)$$

を考えると，答えは明らかに，

$X=$ {勘吉，麗子，剣四郎}

である．一方，この問い合わせを関係代数式に展開すると，

$$\begin{aligned}&\sigma_{1=仁}\,\text{par}\\&\cup\sigma_{1=仁}\{\pi_{1,4}(\text{par}\underset{2=3}{\bowtie}\text{par})\}\\&\cup\sigma_{1=仁}\{\pi_{1,6}(\text{par}\underset{2=3}{\times}\text{par}\underset{4=5}{\bowtie}\text{par})\}\\&\cup\sigma_{1=仁}\{\pi_{1,8}(\text{par}\underset{2=3}{\bowtie}\text{par}\underset{4=5}{\times}\text{par}\underset{6=7}{\times}\text{par})\}\\&\cup\cdots\cdots\end{aligned}$$

のように，(6)式同様，無限個の式の列となってしまい意味をもたない． ■

ancestor のルールは，一目して再帰ルールと分かるものであるが，たとえば，

$$\begin{aligned}&p\leftarrow q, r.\\&q\leftarrow p, s, t.\end{aligned}$$

という二つのルール（ただし，r, t, s は EDB 述語）からなる論理プログラムは，おのおののルールは一見再帰的ではないが，全体としては明らかに再帰的である．ここで，再帰性に関する厳密な定義を与えることにしよう．

［定義1］[4] (a)述語 q を頭部にもつルールの本体に述語 p が出現するとき，p から q が導出されるといい，$p\rightarrow q$ で表す．記号$\rightarrow$*を→の推移的閉包とするとき，$p\rightarrow^* q$，かつ，$q\rightarrow^* p$ ならば，述語 p と q は**相互再帰的** (mutual recursive) という．

(b) $p\leftarrow q_1, q_2, q_3, \cdots, q_n$.

で与えられるルールが再帰的である必要十分条件は，$q_1, \cdots, q_n$ の中に p と相互再帰的なものが存在することである． ■

2.3 問い合わせ評価の停止性と安全性

再帰的な論理プログラムは，(6)式や例2でみたように非再帰的な無限個のルールからなる論理プログラムと等価である．したがって，再帰的な論理プログラムを用いた問い合わせを評価する際に，評価が有限時間で停止し（**停止性** (termination)），しかも，評価の途中の計算結果として生じる中間関係の大きさが有限となること（**安全性** (safety)）が EDB の有限性からのみ保証されることは，評価アルゴリズムに課せられる最も重要な条件である．

停止性は，各アルゴリズムの制御と，EDB 中のデータ構造に依存する場合もある．たとえば，評価アルゴリズムの一つである Prolog において，すべての答えを得ようとすれば停止しないプログラムは容易に書ける．

［例3］ Prolog 処理系において，論理プログラム

$$\begin{aligned}&\text{friend}(a, b)\leftarrow.\\&\text{friend}(b, a)\leftarrow.\\&\text{circle\_of\_friends}(X, Y)\leftarrow\text{friend}(X, Y).\\&\text{circle\_of\_friends}(X, Y)\leftarrow\text{circle\_of\_friends}(X, Z),\\&\qquad\qquad\qquad\qquad\text{friend}(Z, Y).\end{aligned}$$

に対して，次の問い合わせを考える．

$$\text{query}(X)\leftarrow\text{circle\_of\_friends}(a, X).$$

Prolog では，一つの答えを得た時点で“;”の入力に

よって強制的にバックトラックを起こすことで，すべての答えを得るための処理手順は用意されているが，このプログラムの場合，答えとして $X=a$ と $X=b$ が交互に現れるだけで，"no" を得ることはできない．この例のように，グラフの推移的閉包を求めることに対応づけ可能な問題に対して，対象となるグラフにサイクルがある場合，Prolog は，「これで答えがすべてである」ことを検出できない．つまり，停止性が保証されない． ■

この例から，データベースのように「すべての答えを求める」ことを要求される場合には，Prolog が必ずしも強力ではないことが分かる．演繹データベースにおける問い合わせ評価アルゴリズムを停止性の観点から議論する際には，アルゴリズムが停止する EDB のデータ構造，および，IDB の論理プログラムのクラス，すなわち，アルゴリズムの停止性が保証できる適用範囲を各アルゴリズムごとに明らかにせねばならない．

これに対し，安全性は構文論的に（すなわち，論理プログラムの形の上から）議論できる．安全性が損なわれる二つの原因を例を用いて解説する．

［例 4］（1）**算術述語**（算術式を用いて記述した論理式．ただし，ホーン節に限る）を用いた論理プログラム

million_seller $(X)\leftarrow X\geqq 1000000$.

に対する問い合わせ，query $(X)\leftarrow$million_seller (X). は安全ではない．なぜなら，集合 $\{X|X\geqq 1000000\}$ は無限集合であり，述語 million_seller に対応する関係の大きさは無限になるからである．この例のように，算術述語のように，EDB 述語でも IDB 述語でもない述語の存在によって，安全性が損なわれる場合がある．

（2） 次の例は，ルール本体に現れない変数が頭部に出現することで，安全性が損なわれる場合である．

million_seller $(X)\leftarrow X\geqq 1000000$.
attractive (madonna)$\leftarrow$.
loves $(X, Y)\leftarrow$attractive (Y).

に対する問い合わせ，query$(X)\leftarrow$loves$(X,$ madonna). は安全ではない．なぜなら，この系の中では loves を定義するルール中の変数には数も代入できることから，loves $(X,$ madonna$)$ は，任意の整数 X について真となるからである．ただし，X の値として整数も考え得ることは，1番目のルールより明らかである．このルールは，たとえば，「人間である」ことを表す EDB 述語 human (X) を用いて，

loves $(X, Y)\leftarrow$human (X), attractive (Y).

とすべきである． ■

ルール頭部の変数がすべて本体に現れているとき，ルールが**領域制限されている** (range restricted) という．算術述語を考えなければ，領域制限されたルールは安全に評価することができる．また，ルール中に算術述語が現れる場合，算術述語に現れるすべての変数が EDB 述語の引数に現れれば，ルールを安全に評価することができる．ただし，これは十分条件に過ぎない．

二つのアルゴリズムが停止性と安全性を保証する意味で同じ適用範囲をもつとき，中間関係の大きさが小さく，しかも計算を効率よく行うことのできるアルゴリズムが良いことは明白である．例 1 の最後に述べたように，引数をできるだけ束縛することが，評価の際に生じる中間関係が小さくてすむアルゴリズムの開発上重要なポイントになることも容易に想像できる．

3. ルール/ゴールグラフ

問い合わせに用いられる論理プログラムをグラフ表現することにより，その構造を明らかにすることは，問い合わせ評価を考えるうえで役立つ．多くの研究者がさまざまな記法のグラフを考案してきたが，その多くは AND/OR グラフ，または，その変形である．ここでは，文献18)で提案されたルール/ゴールグラフから引数の束縛/自由の区別を省いた，**縮退ルール/ゴールグラフ**を紹介する．引数の束縛/自由を考えないことから，特に，ゴールの引数を省略して記すことにする．

グラフの各ノードは，ルールまたはゴールに対応する．IDB 中にルール

$p\leftarrow q_1, q_2, q_3, \cdots, q_n$.

が存在するとき，$q_1, q_2, q_3, \cdots, q_n$ に対応するゴールノードから，ルール $p\leftarrow q_1, q_2, q_3, \cdots, q_n$ に対応するルールノードへのアークと，このルールノードからゴール p に対応するゴールノードへのアークが存在する．

ルールノードに入るアークは，対応するルールを評価するのに，その先行ノードに対応するゴールのすべての評価が必要であることを示し（**AND 条件**），ゴールノードに入るアークは，その先行ノードに対応するルールの評価の結果の論理和がゴールの評価となることを示す（**OR 条件**）．

［例 5］ 論理プログラム

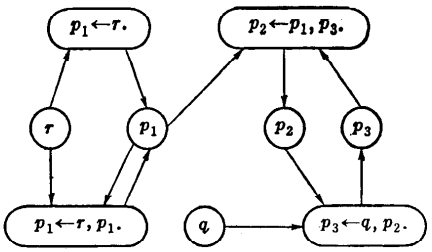


図-2　ルール/ゴールグラフ

$p_1 \leftarrow r.$

$p_1 \leftarrow r, p_1.$

$p_2 \leftarrow p_1, p_3.$

$p_3 \leftarrow q, p_2.$

のルール/ゴールグラフは図-2 のようになる.

4. トップダウン評価とボトムアップ評価

EDB として例1の(4)式, IDB 中のルールとして(5)式で与えられた演繹データベースに対して, 例2と同様, (7)式で記述される問い合わせ

query $(X) \leftarrow$ ancestor (仁, X).

の評価を考える. まず, 例2における考察から, 問い合わせゴール ancestor(仁, X) に対して, (5)式は,

par (仁, X);

par (仁, X_1), par (X_1, X);

par (仁, X_1), par (X_1, X_2), par (X_2, X);

.....

のように展開される. この展開の様子を AND/OR 木で表したのが図-3 のルール/ゴール木である. ただし, 図中, リテラルに記号"←"をつけたものがルールノードを表す. ルール/ゴール木が, ルール/ゴールグラフを展開した形状をとっていることは容易に分かる.

トップダウン評価法は, この木を上から下に向かって生成しながら評価を進める評価法であり(図-4 参照), 逆に, **ボトムアップ評価法**は, 下から上に向かって木を生成しながら評価を進める方法である (図-5 参照).

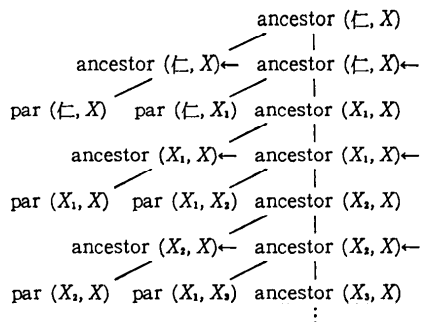


図-3　ルール/ゴール木

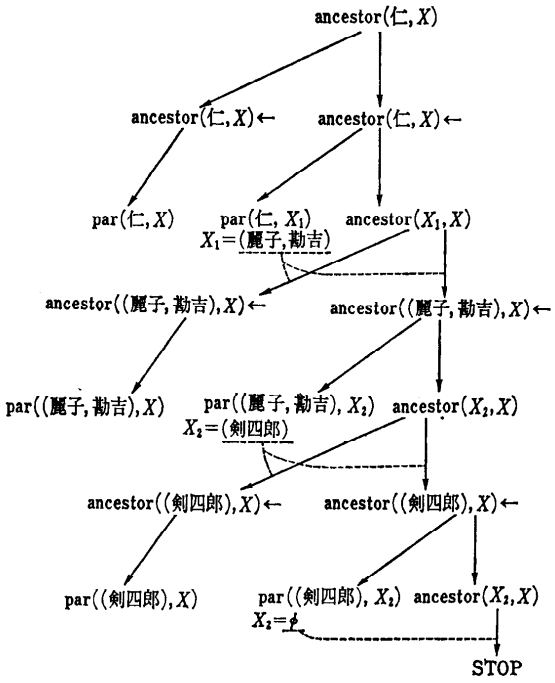
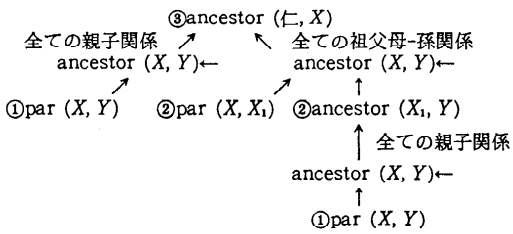


図-4　トップダウン評価



図中のゴールノードの番号は評価の順序である

図-5　ボトムアップ評価

論理プログラムでいえば, トップダウン評価法は問い合わせゴールから始め, IDB 述語にはルールを適用してゴールを展開する. すなわち, ルール頭部からルール本体へと評価を進める. これに対しボトムアップ評価法では, EDB 述語から評価を開始して, IDB 述語に対応する関係を順次生成する.

図-4 を詳しくみると, 同じレベルのノードに対応するゴール間で束縛を伝達することで, 問い合わせに現れた束縛引数, "仁"に関連するものが順次取り出されている. トップダウン評価法における, このような束縛情報の伝達を**情報伝達** (information passing) と

呼ぶ．情報伝達には，図-4のような**ユニフィケーション** (unification) によるものと，同じゴールが現れたときにまとめて評価する（すなわち節点を一つにまとめる）**横方情報伝達** (sideways information passing) がある．情報伝達の仕方は，各評価アルゴリズムによって異なり，それに対応して図-4 のルール/ゴールグラフの形が変化する．

一方，ボトムアップ評価法では，図-5 で明らかなように，ゴールを評価して得られた値が上へ伝えられることで，ルール頭部の IDB 述語に対応する関係が1回生成される．さらに，この上向きの値の伝達による IDB 述語の評価を続けることで，対象となっている EDB の中で考えられるすべての「親-子」，「祖父母-孫」などの関係が順次生成される．

以上の考察から分かるように，トップダウン評価法に比べて，ボトムアップ評価は最終的な答えを導出するのに，多くの不要な中間関係を生成する可能性が大きい．しかし，評価法としてはボトムアップの方が単純であり，例1，2から分かるように，既存の関係データベースとの適合性もよい．

5. 潜在的関連事実

前章の情報伝達のところで触れた，問い合わせの答えに関連する事実とは，**2.** で導入した記号 "→*" を用いて次のように定義できる．

[定義2] query(X) を問い合わせとし，定数 b を答えの集合の要素とすれば，$p(a)$→*query(b) が成立するとき，$p(a)$ は query(X) に**関連している** (relevant) という． ■

ある問い合わせに対し，EDB の部分集合で，問い合わせに関連する事実のすべてからなる集合（関連事実の十分集合）が明らかになれば，EDB をこのより小さな集合で置き換えても問い合わせを正しく評価できる．しかし，このような部分集合を取り出すのは，一般には問い合わせ評価と同等の時間がかかる[11]．したがって，多くの問い合わせ評価アルゴリズムでは，中間関係として関連事実の十分集合を包含し，かつ，できるだけ小さな集合を取り出すように工夫されている．各アルゴリズムについて，その実行中に取り出される事実を**潜在的関連事実**(potentially relevant fact) と呼ぶ．これまでにも述べてきたことから分かるように，問い合わせに現れる束縛を評価中にいかに生かすかが，潜在的関連事実の集合の大きさを左右する重要な問題となる．

6. 問い合わせ評価法

本章以降では，これまでに考案されている主な問い合わせ評価アルゴリズムを解説する．多くのアルゴリズムは，**4.** で述べたトップダウン評価法かボトムアップ評価法のいずれかであるが，それらの両方を取り入れたアルゴリズムもある．

以後，多くの場合，例題を用いてアルゴリズムを解説するが，もちろん，各アルゴリズムは一つの例題を解くためのものではない．各アルゴリズムで問題になるのは，問い合わせゴールのどの引数が束縛されていて，どの引数が自由かということである．たとえば，(7)式の問い合わせゴール ancestor (仁, X) についていえば，アルゴリズムで問題になるのは，第1引数が束縛されているということだけであり，その値が"仁"であることはアルゴリズムの実行を左右するものではない．

問い合わせゴールの各引数の束縛/自由に関する情報だけを取り出して，問い合わせゴールをより抽象化したものを**問い合わせ形式** (query form) と呼ぶ．問い合わせ形式の束縛/自由の情報は，**修飾子** (adornment) と呼ばれる上付きの添字で表す．修飾子は，述語の項数と同じ数の文字からなり，第 i 番目の文字は第 i 引数の束縛/自由に関する情報であり，おのおの b/f で表す．たとえば，問い合わせゴール ancestor (仁, X) に対応する問い合わせ形式は，ancestor^{bf} である．

問い合わせ形式を用いることの効用の一例として，次のようなことが考えられる．効率の良い問い合わせ評価を行うために，論理プログラムと問い合わせをあらかじめ別の形に**翻訳** (compile) する手法が考えられるが，翻訳を問い合わせ形式について行えば，実行時には修飾子の b に対応する引数に定数をあてはめればよい．すなわち，問い合わせ形式に対する翻訳アルゴリズムを考えることで，同じ問い合わせ形式をもつ複数の問い合わせに関しては，一度だけ翻訳を実行すればよい．

以下で例題として用いる論理プログラムは，(5)式で定義される「先祖問題」，または，

$$\text{sg}(X, X)\leftarrow.$$
$$\text{sg}(X, Y)\leftarrow\text{par}(X, X_1), \text{par}(Y, Y_1), \text{sg}(X_1, Y_1). \quad (8)$$

のように定義される「同世代問題」のいずれかに限ることにしよう．「同世代問題」は，(8)式によって定義

される関係が「同世代の親戚」であることから，このように呼ばれる．

「先祖問題」の再帰ルールは，

ancestor (X, Y)←ancestor (X, Z), par (Z, Y). (9)

あるいは，

ancestor (X, Y)←ancestor(X, Z), ancestor(Z, Y). (10)

と定義することもできる．(5)式の再帰ルールは再帰が右方向に展開されることから，**右再帰的**であるといい，逆に，(9)式は**左再帰的**であるという．また，(5)，(9)式のように，本体に再帰述語が一個しか現れないルールを**線形の再帰ルール**という．(10)式は，(5)，(9)式と異なりルール本体に再帰述語が二つ現れており，「先祖問題」の非線形な定義になっている．

6.1 単純なボトムアップ評価法

単純なボトムアップ評価法 (naive evaluation)[2),5)] では，まず，再帰的に定義された論理プログラムを繰り返し的なプログラムに変換する．変換後の繰り返し的なプログラムを「先祖問題」の(5)式に対する問い合わせ形式

ancestor^{bf}

の評価について考えると，アルゴリズムは次のようになる．

Algorithm 1. 単純なボトムアップ評価法

1) 停止条件である

ancestor (X, Y)←par (X, Y).

に従い ancestor の中間関係として，関係 par の組を代入する．

2) **while** ancestor に新たな組が生じた **do**

これまでの ancestor と par を用いて，

ancestor (X, Y)←par (X, Z), ancestor(Z, Y).

の本体を評価し，新たに ancestor に加える．

endwhile

3) 求められた ancestor を，問い合わせにおいて第1引数を束縛している値で選択する． ■

このアルゴリズムは，問い合わせゴールに現れるIDB 述語に対応する関係の組のすべてをボトムアップに導出し，その後で，問い合わせゴールと適合するものを取り出すという簡単なものである．

ボトムアップ評価が進む過程はルール/ゴールグラフと対応づけると分かりやすい．ステップ1は，ルール/ゴールグラフの EDB 述語に対応するゴールの評価に対応する．ステップ2は，グラフ中のサイクル，すなわち，相互再帰的な各ルールの評価を新たな関係が生じなくなるまで行うことと対応している．したがって，上記のアルゴリズムの場合，ステップ1で親-子関係が生成され，ステップ2の1回目の繰り返しで祖父母-孫関係が，2回目の繰り返しで曽祖父母-曽孫関係がそれぞれ導出される．ただし，2回目の繰り返しで，祖父母-孫関係も再度導出される．

単純なボトムアップ評価法では，ステップ2の評価の繰り返しの各段階で，それまでに求まった組のすべてを用いて評価を行うため，上記のように同じ関係が重複して導出される場合がある．この冗長性を避けるために，繰り返しの各段階で求まる関係の組の差分に着目する方法も種々考案されている．これらの評価法は，**セミナイーブ評価法** (semi-naive evaluation)[2),5)] と呼ばれ，繰り返し部分の制御が，単純なボトムアップ評価法より洗練されている．

ここで，ボトムアップ評価法の安全性について述べておく．まず，領域制限的でないルールが望ましくないことは明らかであろう．しかし，ルールが領域制限されていても，たとえば，次の論理プログラム

$p_1(1, X, Y)$←$X=Y$.

$p_2(2, X, Y)$←$X=Y$.

$r(X, Y)$←$p_1(X, Z, Z)$, $p_2(Y, Z, Z)$.

に対して，問い合わせ query(X, Y)←$r(X, Y)$. の答えは，$X=1$, $Y=2$ であるが，この問い合わせを安全に評価することはできない．なぜなら，第1式と第2式の本体の算術述語に現れる変数を束縛する要素がないからである．算術述語を含めたボトムアップ評価の安全性に関する十分条件として，たとえば，次の条件が考えられている[20)]．

まず，ルール本体に現れる変数 X と Y について，X が束縛されたときに，Y のとり得る値の集合が有限になることが確定できれば，$X \to Y$ と書き，**安全従属性** (safety dependency) をもつという．たとえば，算術述語 $X=Y$ の場合，安全従属性 $X \Rightarrow Y$ と $Y \Rightarrow X$ が共に成立する．この安全従属性を用いて，変数の安全性を次のように定義する．

[定義3] (1) 算術述語でない EDB 述語および IDB 述語に現れる変数は安全である．

(2) 算術述語に現れる変数 Y は，安全な変数 X からの安全従属性 $X \Rightarrow Y$ が存在するとき，安全である． ■

このように定義された変数の安全性を用いると，ボトムアップ評価が安全に行われるための十分条件は，

次のように表現することができる[20].

[ボトムアップ評価可能性条件] (a)ルールが領域制限されていて，かつ，(b)ルール本体のすべての変数が安全である. ■

ボトムアップ評価可能なルールは，各述語の評価順序を適当に選べば，単純なボトムアップ評価アルゴリズムで安全に評価できる.

6.2 トップダウン評価法

トップダウン評価法の中で最もよく知られているのは，Prolog である. Prolog は，ある手順で展開されたルール/ゴール木を単純に**深さ優先探索** (depth first search) を行う評価法である. 例3で論じたように，Prolog プログラムの停止性は，利用者が責任を負わねばならない. ここでは，Prolog についての詳述は避け，特に，演繹データベースの問い合わせ評価法として考案された **Query/Subquery 法** (QSQ)[19]，および，Henschen-Naqvi のアルゴリズム[9),14)]について述べる.

QSQ には，繰り返し型アルゴリズムと再帰型アルゴリズムがあるが，以下では，より効率的とされている再帰型アルゴリズム (**QSQR**) について紹介する. QSQR が Prolog と大きく異なるのは，Prolog が一段階の評価で一つの組を生成するのに対し，QSQR は集合を生成する点である. この意味では，ボトムアップ評価法と共通点がある. QSQR では，ルールを評価するときに現れるサブゴールを新たな問い合わせ(**問い合わせ実体** (query instance) と呼ぶ) と見なし，新たな問い合わせ実体が現れた時点で，問い合わせ実体を評価する手続きが再帰的に呼び出される.

QSQR では，アルゴリズムの実行中二つの状態 R と Q を保持する. 状態Rは，各 IDB 述語に対し，それまでに求まっている関係であり，次の IDB 述語の評価の際に用いられる. 状態 Q は，問い合わせ実体の集合であり，それまでに発せられた問い合わせ実体が保持され，問い合わせ実体の生成を制御するために用いられる. 状態Qの中では，同じ問い合わせ形式に対する複数の問い合わせ実体を，一つの問い合わせ実体と見なしたものを中間状態として保持する (つまり，二つの問い合わせ実体，$q(a, X)$ と $q(b, X)$ は，一つの $q(\{a, b\}, X)$ という問い合わせ実体と見なされる) ことにより効率の良いループ制御を実現している. アルゴリズムは，どの IDB 述語を最初に評価するかを選ぶ選択関数を用いて，次のように記述できる.

Algorithm 2. QSQR

```
procedure Answer (q: 問い合わせ実体) ;
begin
  while 状態が変化 do
    if q が元の問い合わせ実体 then Q=φ
    (*元の問い合わせ実体とは，たとえば，(7)式
    の本体である ancestor(仁, X) が対応する*)
    for q とマッチする全てのルール do
      問い合わせにおける束縛を伝播する ;
      (*このとき EDB 述語と R を用いる*)
      while 本体に未評価の述語がある do
        選択関数によって評価する IDB 述語を選
        び，対応する問い合わせ実体 q' を生成
        する ;
        Q: Q∪{q'}
        Answer (q') ;
      endwhile;
      q を評価する ;
      (*このとき EDB 述語と R を用いる*)
      結果をRに加える ;
    endfor
  endwhile
end Answer;

begin (*of evaluation*)
  Q:= {query(X)} ;
  R:=φ;
  Answer (query(X));
end.
```

■

IDB として(5)式の論理プログラム，EDB として(4)式の「親子関係」のデータをもつ演繹データベースに対して，(7)式の「先祖問題」の問い合わせを発したときの評価の進行の様子を**図-6** に示す. 図中ゴールノードに付した番号①，②は，EDB 述語によって，束縛が①から②へと伝達されることを示すものである. 図-6 の各段階で生成される木を合成すると，図-4 と同様の木が得られる. すなわち，図-4 における横方束縛伝達の仕方は QSQR の場合と同様である.

このように，(繰り返し型アルゴリズムと再帰型アルゴリズムのいずれの場合も) QSQ は，束縛の伝播に関しては典型的なトップダウン評価であり，領域制限的で，しかも，算術述語のないルールに適用できる. QSQ では，アルゴリズムは状態に変化がなくなれば停止するので，Prolog のように EDB のデータ

論理プログラム
ancestor (X, Y)←par (X, Y).
ancestor (X, Y)←par (X, Z), ancestor (Z, Y).
問い合わせ
query(X)←ancestor (仁, X).

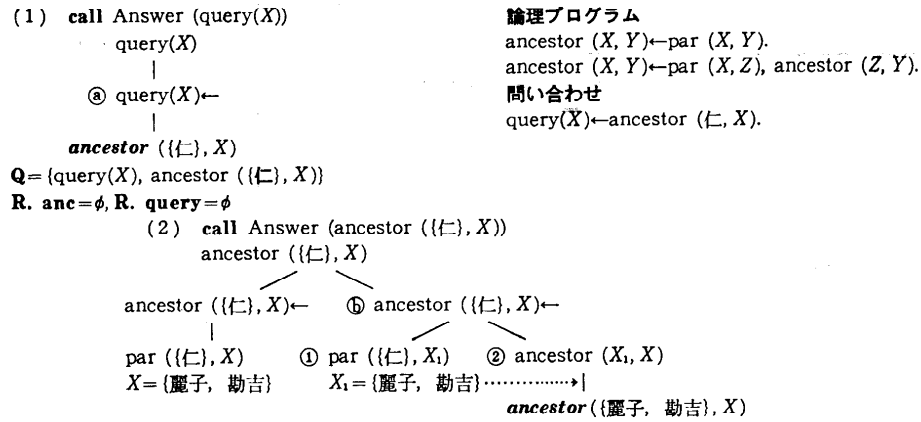


図-6 QSQR による評価
(図中, 斜体で記されたゴールは新たに生成された問い合わせ実体を表す)

構造のサイクルが原因で停止性を損なわれることはない．QSQ は，M. Stonebraker らによって開発されたデータベースシステム INGRES[16]に実装されている．

QSQ がサブゴールの取り扱いに工夫を凝らしていたのに対し，Henschen-Naqvi のアルゴリズムは，基本的には，(6)式のように論理式の展開をトップダウンに行い，各段階で展開された式を，それまでの結果を生かして効率よく評価する方法である．このアルゴリズムは，線形の論理プログラムに対して形式化されているが，本稿では「先祖問題」を例にとって解説する．

まず，Aを集合とし，r を二項関係とすると，

$$A \circ r = \{y \mid r(x, y) \text{ かつ } x \in A\}$$

により記号 "∘" を定義する．これは，集合の写像にほかならず，r を EDB 述語と考えれば，問い合わせ評価では EDB 述語による束縛の伝播を意味する．

(5)式のルールに対して，(7)式の問い合わせを発すると，答えは，記号 "∘" を用いて次のように書くことができる．

$$\{仁\} \circ par \cup \{仁\} \circ par \circ par \cup$$
$$\{仁\} \circ par \circ par \circ par \cup \cdots \cup \{仁\} \circ par^n \cup \cdots$$

そこで，Henschen-Naqvi のアルゴリズムでは，問い合わせを次のプログラムに変換する．ただし，下記のアルゴリズム中 V, answer は単項関係の値の集合である．

Algorithm 3. コンパイル後のプログラム

$V := \{仁\}$;
answer $:= \phi$
while 未出現の新たな要素が V に生じた **do**
  $V := V \circ par$
  answer $:=$ answer $\cup V$
endwhile. ■

このプログラムを実行すると，ループを1回まわるたびに，"仁"の親，祖父母，曽祖父母，…が順に求まることは明白であろう．特に，プログラムの実行時に生じる中間関係が単項関係であることは注目すべきである．つまり，上記のプログラムは潜在的に，「結合演算の前に選択演算と射影演算を実行する」という性質を備えている．

以上で，単純なボトムアップ評価と，主なトップダウン評価アルゴリズムについて解説した．ここで解説したもの以外にも，トップダウン評価とボトムアップ評価を融合した方法もあるが本稿では詳述しない．次章では，最近特に話題となっている，ボトムアップ評価に先立つ論理プログラムの書き換えについて述べる．

7. ボトムアップ評価に対する項書換え手法

単純なボトムアップ評価の欠点は，評価の途中で，問い合わせに対する答えと関連しない組を生成することであった．本章では，そのような組，すなわち，潜在的関連事実をできるだけ少なくするために，ルール（つまり，論理プログラム）を書き換える手法について解説する．

評価がボトムアップに行われることから，答えと関連しない事実を取り除くには，Algorithm 1 中の各式の評価の際に，答えに関連する組を選択すればよい．この選択の仕方によって，大きく二通りの方法に分けることができる．一つは，問い合わせに現れる束縛値をそのまま使うもの．もう一つは，トップダウン評価における横方束縛伝達を模倣するものである．前者は「結合の前に選択演算を施す」という原則を忠実に再現したものであるのに対し，後者は，より，積極的に選択演算を行う方法である．以下，**7.1** で前者の範疇に属するアルゴリズムを，**7.2** で後者の範疇に属するアルゴリズムを解説する．

7.1 最小不動点演算子による方法

「先祖問題」に関するルールが，(9)式のように左再帰のルールで定義され，

ancestor $(X, Y) \leftarrow$ par (X, Y).
ancestor $(X, Y) \leftarrow$ ancestor (X, Z), par (Z, Y).

のように与えられている場合のボトムアップ評価法の進行の過程を考える．評価は，まず第1式から ancestor に対応する関係を取り出し，それを用いて第2式を評価する．以後，それまでに求められた ancestor を用いて，第2式の評価を ancestor に新たな組が生成されなくなるまで続けるというものであった．この間，ancestor は単調に増え続け，必要な解がすべて求まった時点で停止する．つまり，前章で定義した演算子 "∘"を用いて上式を

$$\text{ancestor} = \text{ancestor} \circ \text{par} \cup \text{par} \qquad (11)$$

と書いたとき，方程式(11)の**最小不動点**[12]を求めていることにほかならない．そこで，最小不動点演算子 LFP を用いて(7)式の問い合わせを表現すると，

$$\sigma_{1=仁}(\text{LFP}(\text{ancestor} = \text{ancestor} \circ \text{par} \cup \text{par})) \qquad (12)$$

となる．

(12)式の最初に現れている選択演算子をできるだけ式の内側に移し変えることで，できるだけ早い段階で選択演算を施そうというのが **Aho-Ullman のアルゴ**

リズム[1] である.

Aho-Ullman のアルゴリズムは，問い合わせゴール

$\sigma_{1=仁}$(ancestor)

に対して，(12)式と等価で，より内側に選択演算が挿入された式を導く．まず，ancestor を方程式(11)の右辺で置換すると，次式が得られる．

$\sigma_{1=仁}$(ancestor ∘ par ∪ par)

和演算の左右に選択演算を分配することができ，

$\sigma_{1=仁}$(ancestor ∘ par) ∪ $\sigma_{1=仁}$(par)

が得られるが，この式の第1項は，

($\sigma_{1=仁}$(ancestor)) ∘ par

と等価なので，

($\sigma_{1=仁}$(ancestor)) ∘ par ∪ $\sigma_{1=仁}$(par)

を得る．この式は，もとの問い合わせゴールと同じ形の式を含んでおり，したがって，(12)式は，

LFP($\sigma_{1=仁}$(ancestor)) ∘ par ∪ $\sigma_{1=仁}$(par)

と等価であることが分かる．これを論理プログラムの形で表現すると，

ancestor(仁, Y)←par(仁, Y).
ancestor(仁, Y)←ancestor(仁, Z), par(Z, Y). (13)

となる．この論理プログラムを用いて，問い合わせをボトムアップに評価を行うと，各段階で，評価の途中で生じる中間関係のうち，第1属性が“仁”である組のみを選択することになる．

文献1)によれば，このアルゴリズムが適用できるのは，問い合わせゴールの述語を定義する論理プログラムにただ一つ再帰ルールがあり，しかもその再帰ルールが線形である場合（このような論理プログラムのクラスを**強線形**と呼ぶ）に限られ，それ以上複雑な論理プログラムの場合，一般に，Aho-Ullman のアルゴリズムは適用できない．すなわち，不動点方程式を $r = f(r)$ と書いたとき，r が関数 $f(r)$ にただ1回現れるような場合に対してのみ適用できる．最小不動点演算子に基づく方法のより一般的な記述は，文献13) にある．

Aho-Ullman のアルゴリズムによって得られた論理プログラムに対する問い合わせ評価をルール/ゴール木（図-5 参照）で考えると，ゴールノードからルールノードに，上向きに値を伝達するときに，“1 =仁”というフィルタをかけることに対応する．したがって，ボトムアップ評価は，ルール/ゴールグラフ中のアークの上のデータの流れとして捉えることができ，Aho-Ullman の発想は，問い合わせゴールにおける束縛値によるフィルタを，より多くのアークに設けることにほかならない．

以上の Aho-Ullman のアルゴリズムの適用可能範囲を広げるために，ルール/ゴールグラフを用いて定式化することで，領域制限的で算術述語がない論理プログラムに対して適用可能にしたのが，**Kifer-Lozinskii のアルゴリズム**[10]である．ルール/ゴールグラフ(文献10)では，**システムグラフ**と呼ばれている）上でのデータの流れは，アークの向きと一致することに注意すると，Aho-Ullman のアルゴリズムをルール/ゴールグラフ上でシミュレートするためには，あるアーク上に与えられたフィルタは，そのアークが出ているノードに入るアークに，すなわち，データの流れと反対向きに，可能なかぎりあらかじめ伝達しておく必要がある．

Kifer-Lozinskii のアルゴリズムは，ルール／ゴールグラフ上で，次の二つの基本操作をグラフ中のフィルタに変化がなくなるまで続ける．

（a） あるゴールノードから出るすべてのアークにフィルタが与えられているとき，それらのフィルタ条件の選言をそのゴールノードに入るすべてのアークにおく．

（b） あるルールノードから出るすべてのアークにフィルタが与えられているとき，各フィルタ条件として与えられている選言について，対応するリテラルの変数に着目したときに得られる最も強い帰結を，そのルールノードに入るすべてのアークにおく．

(11)式の例について，変換前のルール/ゴールグラフと変換後のルール/ゴールグラフを**図-7** に示す．ただし，図中，アークに付けたフィルタにおいて，“1”は頭部の述語の第1引数を，a_1 はゴール ancestor の第1引数を表す．変換後のグラフは，まさに(13)式に対応している．

Aho-Ullman および Kifer-Lozinskii のアルゴリズムは，最小不動点演算子による方法が基礎になっている．したがって，(5)式のような右再帰ルールのもとでの「先祖問題」，および，上記のような左再帰ルールのもとでの「子孫問題」（すなわち，第2引数が束縛される問い合わせを発する）の場合には，Aho-Ullman および Kifer-Lozinskii のアルゴリズムは効果がない．

7.2 束縛伝達による方法

本節では，横方情報伝達を模倣することで潜在的関連事実の集合を小さくする方法について述べる．

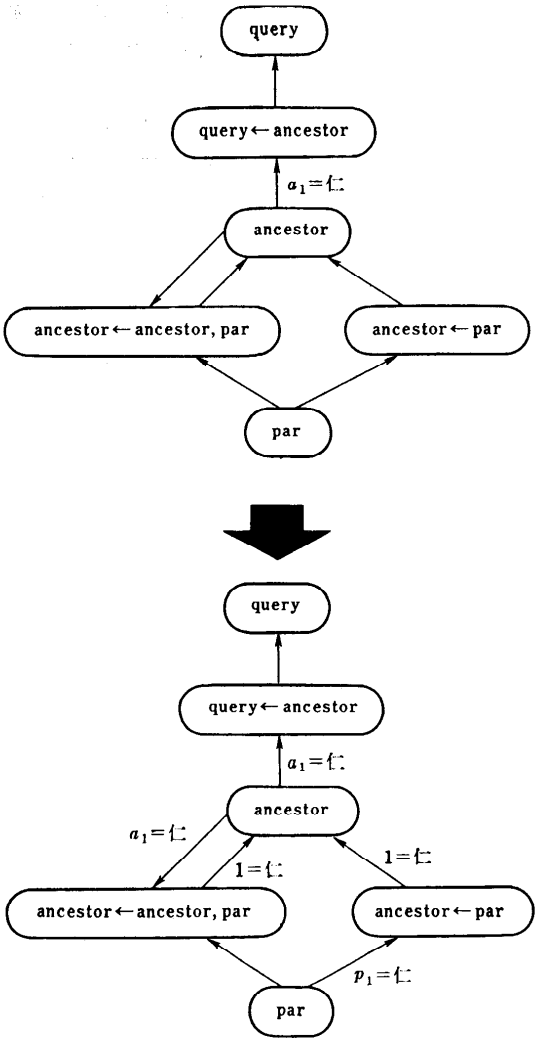


図-7 Kifer-Lozinskii のアルゴリズムによるルール/ゴールグラフの変換

まず，(5)式のルールのもとで，(7)式の問い合わせをする「先祖問題」について考える．図-4 に示した，この問題に対するトップダウン評価のルール/ゴール木を見ると，述語 ancestor のゴールノードには"仁"と，その先祖が順次現れることに気づく．ところで，"仁"とその先祖を求める論理プログラムは簡単に，

anc_jin (仁)←.

anc_jin (*Z*)←par (*X*, *Z*), anc_jin (*X*). (14)

と書ける．(14)式の anc_jin で定義される単項関係は，ancestor の第1引数を束縛し得る要素の集合であり，QSQR の中間状態のなかで，ancestor に対応する問い合わせ実体（図-6 参照）の第 1 引数に現れる値の集合とも同じである．この述語 anc_jin を用いると，(5)式に基づくボトムアップ評価で，QSQR と同様の横方束縛伝達を模倣するように，ルールを次のように書き換えることができる．

ancestor (*X*, *Y*)←anc_jin (*X*), par (*X*, *Y*).

ancestor (*X*, *Y*)←anc_jin (*X*),
par(*X*, *Z*), ancestor(*Z*, *Y*). (15)

(15)式において，anc_jin は，選択演算子として答えに関連のない組を取り除く働きをする．

以上のアイデアを定式化したのが**マジック集合の方法**[3),6)]である．特に，anc_jin のように，ある引数を束縛し得る要素の集合を**マジック集合**と呼ぶ．anc_jin は単項関係であるため，(14)式は，選択演算によって評価できる．したがって，多くの場合，(15)式は(5)式より小さなコストで評価できる．(5)式が右再帰ルールであることを考えると，マジック集合を導入することは，（前節の最後に述べた）右再帰ルールに関する問題の一つの解決策ともなっている．

横方束縛伝達の仕方を与えることで，一般にマジック集合の方法は，領域制限的な論理プログラムに適用できる[6)]．以下では，論理プログラムの書換えの様子を，次の強線形の「同世代問題」の例[3)]を用いて考えてみることにする．

sg (*X*, *X*)←.

sg (*X*, *Y*)←par (*X*, X_1), par (*Y*, Y_1), sg (Y_1, X_1).

query (*X*)←sg (仁, *X*). (16)

(16)式の再帰ルール本体の述語 sg の引数の現れ方が(8)式と異なるが，(16)式で「同世代の親戚」が正しく求まることは確かである．(16)式では，頭部の sg の第1引数の束縛は，1 番目の par を通じて本体の sg の第2引数に伝達され，頭部の sg の第2引数の束縛は，2番目の par を通じて本体の sg の第1引数に伝達される．これは再帰ルールを評価する際，偶数回目の評価と奇数回目の評価で，束縛が伝達されるべき引数が異なることを意味する．そこで，トップダウン評価において束縛が伝わる様子を模倣するために，sg を定義する再帰ルールを問い合わせ形式に応じて二つのルールに分割する．この操作は，各引数の束縛/自由を考慮して構成した修飾子付きのルールに関するルール/ゴールグラフを用いて行う．

sg^{bf} (*X*, *X*)←.

sg^{fb} (*X*, *X*)←.

sg^{bf} (*X*, *Y*)←par (*X*, X_1), par (*Y*, Y_1), sg^{fb} (Y_1, X_1).

sg^{fb} (*X*, *Y*)←par (*X*, X_1), par (*Y*, Y_1), sg^{bf} (X_1, Y_1).

query (X)←sg^{bf}(仁, X). (17)

次に，この論理プログラムに現れる各 IDB 述語に対して，マジック集合を求めるルール(**マジックルール**)を導く．(17)式で束縛伝播の様子を考えると，sg^{bf} の第1引数は“仁”から数えて偶数世代目の先祖で，sg^{fb} の第2引数は“仁”から数えて奇数世代目の先祖で束縛される．これらをおのおの magic. sg^{bf}, magic. sg^{fb} で表すと，二つのマジック集合 magic. sg^{bf}, magic. sg^{fb} を求めるルールは次の式で与えられる．

magic. sg^{bf}(仁)←.

magic. $sg^{fb}(X_1)$←par (X, X_1), magic. $sg^{bf}(X)$.

magic. $sg^{bf}(Y_1)$←par(Y, Y_1), magic. $sg^{fb}(Y)$.

以上の手順を強線形の論理プログラムについて，一般的に記述すると次のようになる．

Algorithm 4. 線形ルールに対するマジックルールの求め方

for すべての再帰ルール **do**

1) 一つの再帰ルールに着目し，本体の IDB 述語を一つ選ぶ．本体のほかの IDB 述語は消去する．
2) IDB 述語の名前 (p とする) を magic. p で置き換える．
(magic. p で書かれる述語を**マジック述語**と呼ぶ)
3) IDB 述語で修飾子 f に対応する変数を消去する．
4) EDB 述語で，頭部と本体の両方の IDB 述語の修飾子 b に対応する変数から束縛が伝達しないものを消去する．
5) 二つのマジック述語を入れ換える．

endfor

6) 問い合わせゴールから，マジックルールの停止条件を書く． ■

ステップ1から4によって，たとえば，

$sg^{bf}(X, Y)$←par (X, X_1), par (Y, Y_1), $sg^{fb}(Y_1, X_1)$.

は，

magic. $sg^{bf}(X)$←par(X, X_1), magic. $sg^{fb}(X_1)$.

と書き換えられ，ステップ5によって，

magic. $sg^{fb}(X_1)$←par(X, X_1), magic. $sg^{bf}(X)$.

を得る．ステップ5はトップダウンの束縛伝達をボトムアップの実行系で求めるための操作であると考えればよい．

最後に，(17)式の再帰ルールを書き換える．再帰ルール頭部の述語が p であれば，magic. p(X) を本体に挿入すればよい．ここで，X は頭部の述語の修飾子の b に対応する変数である．結局，(16)式は，

magic. sg^{bf}(仁)←.

magic. $sg^{fb}(X_1)$←par(X, X_1), magic. $sg^{bf}(X)$.

magic. $sg^{bf}(Y_1)$←par(Y, Y_1), magic. $sg^{fb}(Y)$. (18)

というマジックルールと，

$sg^{bf}(X, X)$←$magic^{bf}(X)$.

$sg^{fb}(X, X)$←$magic^{fb}(X)$.

$sg^{bf}(X, Y)$←$magic^{bf}(X)$,
par(X, X_1), par(Y, Y_1), $sg^{fb}(Y_1, X_1)$.

$sg^{fb}(X, Y)$←$magic^{fb}(X)$,
par(X, X_1), par(Y, Y_1), $sg^{bf}(Y_1, X_1)$.

query(X)←sg^{bf}(仁, X). (19)

に変換され，(18)式と(19)式は，2式あわせてボトムアップに評価される．(18)式のマジックルールで取り出されるマジック集合を図-8 に示す．

マジック集合の方法で問題となるのは，特に，強線形より複雑な論理プログラムの場合，どの横方情報伝達を抽出して模倣するかということである．与えられた横方情報伝達に対して論理プログラムを書き換える手法は，文献6)に詳しい．

マジック集合の方法において，各ルールにおけるマジック集合によるフィルタは，最も強い条件ではない．なぜなら，マジック集合は各ルールの何回目の適用で，どのような値で束縛されるかを考慮していないからである．EDB 中のデータにサイクルが存在しない場合，マジック集合の各要素が，何回目のルールの適用に現れるものかをボトムアップに算出することができる．この情報を利用した手法が，**数え上げ法**[3),6),15)]，**逆数え上げ法**[3)] である．これらの手法について，(8)式のルールに問い合わせを加えた次式の「同世代問題」を例にとって考察する．

sg(X, X)←.

sg(X, Y)←par(X, X_1), par(Y, Y_1), sg(X_1, Y_1).

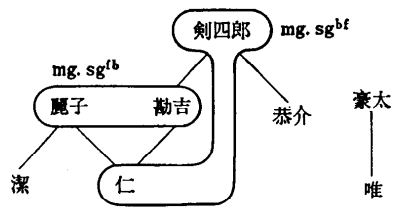


図-8 (18)式のマジック集合
(理解を助けるために，図-1 の EDB の集合に二つの組を加えた)

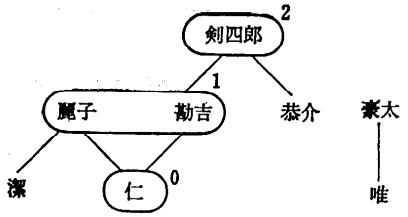


図-9 数え上げ集合
(図-8 と同じデータ構造を用いた)

query(X)←sg(仁, X). (20)

に対するマジックルールは,

magic^{bf}(仁)←.

magic^{bf}(X_1)←par(X, X_1), magic^{bf}(X).

となる.

数え上げ法では,マジック集合の各要素に対し,問い合わせに現れる要素(この場合"仁")に0,以下,マジックルールを n 回適用して得られる要素(この場合 n 代前の祖先)に n という番号を次のように与える.

anc_count(仁, 0)←.

anc_count(X_1, I)←par(X, X_1), anc_count(X, J), $I=J+1$.

この述語 anc_count で定義される集合を**数え上げ集合**という(**図-9** 参照).この述語を用いると,マジック集合の場合と同様,(20)式は,

sg′(X, X, I)←anc_count(X, I).

sg′(X, Y, I)←anc_count(X, I), par(X, X_1), par(Y, Y_1), sg′(X_1, Y_1, J), $I=J-1$.

query(X)←sg′(仁, X, 0). (21)

と書き換えることができる.このルールを用いれば評価の各段階で,横方情報伝達を正確に模倣できる.さらに(21)式において,変数 I の内容は,評価の各段階における sg′ の第1引数の"仁"から数えた世代数と正確に対応しているので,第1引数は省略することができる.よって,

anc_count(仁, 0)←.

anc_count(X_1, I)←par(X, X_1), anc_count(X, J), $I=J+1$.

sg′(X, I)←anc_count(X, I).

sg′(Y, I)←par(Y, Y_1), sg′(Y_1, J), $I=J-1$, $J>0$.

query(X)←sg′(X, 0). (22)

を得ることができる.(22)式は Henschen-Naqvi のアルゴリズムと本質的な類似性があることは興味深い.ただし,(22)式は,Henschen-Naqvi のアルゴリズムと異なり,ボトムアップに評価される.

数え上げ法は,マジック集合の方法と同様の適用範囲をもつが,EDB 中のデータ構造にサイクルのない場合に限定される.データ構造のサイクルの問題に対する一つの解決として,マジック集合の方法と数え上げ法を融合した**マジック数え上げ法**が提案されている.この方法は,データ構造の中で数え上げ法が適用できないサイクル部分を検出して,その部分に関してマジック集合の方法を適用するというものである.

ところで,数える操作を問い合わせゴール中の値ではなく,マジック集合の各要素から始めることもできる.つまり,マジック集合のある要素から数えて,"仁"が n 代目の子孫であるとき,その要素から数えて n 代目の子孫のすべてが"仁"の同世代の親戚であるので,それらを答えとすればよいのである.つまり,マジック集合の要素から世代を算出するルールを b_descendant とすると,(20)式は,

magic(仁)←.

magic(X_1)←par(X, X_1), magic(X).

b_descendant(X, 0)←magic(X).

b_descendant(X, I)←b_descendant(Y, J), par(X, Y), $I=J+1$.

query(X)←b_descendant(仁, I), b_descendant(X, I).

と書き換えることができる.これが逆数え上げ法である.この方法も EDB にサイクルのある場合には停止しない.

8. アルゴリズムの性能評価

本章では,各アルゴリズムを性能面から考察する.これまでの議論から予想できるとおり,各アルゴリズムを実行させた場合,実行時間は問い合わせ形式や,EDB 中のデータ構造に大きく左右される.また,多くのデータベースシステムや,コンパイラなどがそうであるように,実システムの性能はその実現の仕方にもかなり左右される.もちろん,効率的に評価が進められるアルゴリズムでも,場合によっては適用範囲が問題となるだろう.したがって,どのアルゴリズムが最適かということは一概に断言できない.

Bancilhon ら[4]は,以上のことを踏まえた上で,中間関係の大きさのみに着目し,各アルゴリズムを比較している.評価の方法は,「先祖問題」「同世代問題」に対する問い合わせを用意し,EDB のデータ構造を5つのパラメータで表現し,中間関係の大きさを算出するというものである.彼らは得られた結果を総合し

て，中間関係の小さくなるものから順に次のような評価を下している．

1. Henschen-Naqvi のアルゴリズム，数え上げ法
2. マジック集合の方法，QSQR
3. セミナイーブ法
4. 単純なボトムアップ評価

Henschen-Naqvi のアルゴリズムと数え上げ法の類似性については，前章で述べた．また，QSQR とマジック集合の方法の類似性は，マジック集合の方法の横方束縛伝達の模倣の仕方から予想できる．上記の評価の対象として，Kifer-Lozinskii のアルゴリズムがあがっていないが，Kifer-Lozinskii のアルゴリズムは，問い合わせ形式によって，マジック集合の方法と同等になる場合と，セミナイーブ法と等価になる場合がある．さらに，Prolog については，前述のように EDB 中のデータ構造に左右されやすい．データ構造によっては，QSQR と同等の結果を得るが，単純なボトムアップ評価より悪くなる場合もある．

9. 今後の課題

演繹データベースにおける問い合わせ評価に関する研究は，近頃，ようやく再帰の問題についての一般的な解法を確立する見通しが得られてきた段階である．今後，より広いクラスの論理プログラムについて適用するために，各手法のより抽象的な定式化がなされるとともに，各手法の本質が捉えられることによって，それらを融合する試みも行われることと思われる．また，最近，主記憶と二次記憶というデバイス環境を意識したうえでの，再帰的な問い合わせの効率的な評価法に関する研究も始まっている．

昨年，著者も出席する機会を得た第13回 Very Large Data Bases 国際会議においても，「再帰問い合わせの評価」に関するセッションが設けられ，さらに「論理とデータベース」などのセッションにおいても再帰問い合わせの評価に関する論文が発表され，この分野の研究の盛況ぶりが伺われた．日本では，ICOT を中心に再帰問い合わせ法の開発が知識ベースマシーン PHI への実装も含めて行われているが，今後この分野の研究が国内でも盛んになることが期待される．

本稿は，直観的かつ非形式的な記述に留まってしまったが，演繹データベースの再帰的な問い合わせ評価手法を理解する上での一助となれば幸いである．

謝辞 末筆ながら，筆者らが，日頃，ご指導と温かい励ましをいただいている本学長谷川利治教授に衷心より感謝の意を表す．本稿を草稿の段階で詳細に読み，貴重なコメントを寄せてくださった沖電気(株)宮崎収兄氏，日本電信電話(株)勝野裕文氏，本学大学院生中畑雅嗣氏には，深謝の意を表す．最後に，本稿に関して，構成を考える段階から始終有益なご意見をいただいた横浜国立大学有澤博助教授ならびに東京女子大学守屋悦朗助教授には，心より感謝の意を表す．

参 考 文 献

1) Aho, A. V. and Ullman, J. D.: Universaliy of Data Retrieval Languages, Proc. ACM Sixth Symposium on Principles of Programming Languages, pp. 110-120 (1979).
2) Bancilhon, F.: Naive Evaluation of Recursively Defined Relations, On Knowledge Base Management Systems-Integrating Database and AI Systems, Brodie and Mylopoulos, Eds., Springer-Verlag, Berlin, pp. 165-178 (1985).
3) Bancilhon, F., Maiyer, D., Sagiv, Y. and Ullman, J. D.: Magic Sets and Other Strange Ways to Implement Logic Programs, Proc. Fifth ACM SIGMOD-SIGACT Symposium on Principles of Database Systems, pp. 1-15 (1986).
4) Bancilhon, F. and Ramakrishnan, R.: An Amateur's Introduction to Recursive Query Processing Strategies, Proc. ACM SIGMOD '86, pp. 16 52 (1986).
5) Bayer, R.: Query Evaluation and Recursion in Deductive Database Systems, Tech. Rep. TUM-I 8503, Institut für Informatik, Technische Universität München (1985).
6) Beeri, C. and Ramakrishnan, R.: On the Power of Magic, Proc. Sixth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Data-base Systems, pp. 269-283 (1987).
7) Gallaire, H. and Minker, J.: Logic and Data Bases, Plenum Press, New York (1978).
8) Gallaire, H., Minker, J. and Nicolas, J.-M.: Logic and Data Bases: A Deductive Approach, ACM Comput. Surv., Vol. 16, No. 2, pp. 153-185 (1984).
9) Henschen, L. J. and Naqvi, S. A.: On Compiling Queries in Recursive First-Order Data Bases, J. ACM, Vol. 31, No. 1, pp. 47-85 (1984).
10) Kifer, L. and Lozinskii, L.: Filtering Data Flow in Deductive Databases, Proc. International Conference on Data Theory '86, pp. 186-202, Lecture Notes in Computer Science 243, Springer-Verlag, Berlin (1986).
11) 楠見，西尾，長谷川：ホーンデータベースにおける強線形ルールによる問合せの一評価法，電子

情報通信学会データ工学研究会資料, DE 86-23, pp. 7-12 (1987).
12) Lloyd, J. W.: Foundations of Logic Programming, Springer-Verlag, Berlin (1984).
13) 宮崎, 羽生田, 伊藤: ホーン節変換: 演繹データベースにおける部分評価の応用, 情報処理学会論文誌, Vol. 29, No. 1, pp. 47-53 (1987).
14) Naqvi, S. A. and Henschen, L. J.: Synthesizing Least Fixed Point Queries into Non-Recursive Iterative Programs, Proc. Ninth International Joint Conference on Artificial Intelligence, pp. 25-28, Karlsruhe, W. Germany (1983).
15) Sacca, D. and Zaniolo, C.: The Generalized Counting Method for Recursive Logic Queries for Databases, Proc. International Conference on Data Theory '86, pp. 186-202, Lecture Notes in Computer Science 243, Springer-Verlag, Berlin (1986).
16) Stonebraker, M., Wong, E., Kreps, P. and Held, G.: The Design and Implementation of INGRES, ACM Trans. Database Syst., Vol. 1, No. 3, pp. 189-222 (1976).
17) Ullman, J. D.: Principles of Database Systems (Second Edition), Computer Science Press, Rockville, MD (1982).
18) Ullman, J. D.: Implementation of Logical Query Languages for Databases, ACM Trans. Database Syst., Vol. 10, No. 3, pp. 289-321 (1985).
19) Vieille, L.: Recursive Axioms in Deductive Databases: The Query/Subquery Approach, Proc. First International Conference on Expert Database Systems, pp. 179-193 (1986).
20) Zaniolo, C.: Safety and Compilation of Non-Recursive Horn Clauses, First International Conference on Expert Database Systems, pp. 167-178 (1986).

(昭和63年1月6日受付)