Vol. 30 No. 9 情報処理学会論文誌 Sep. 1989

演繹データベースにおける制約付最小不動点†

宮崎収兄†† 伊藤英則†††*

演繹データベースにおける再帰問合せの処理では問合せ中の選択条件を有効に利用して演算量を減少させることが重要である．既にトップダウン処理に基づいた方式ではこのような方法が知られている．従来のデータベースと親和性の良いボトムアップ処理に基づいた方式でこれを可能にすれば，システムの性能の改善をさらに図ることができる．本稿では問合せをもとの最小不動点より小さな最小不動点をもつ問合せに変形することにより効率化を図る方法を提案する．この方法は選択の分配法やマジック集合法などと同様に問合せを変形してからボトムアップ評価を行う方法の一種であり，制約述語と呼ぶ述語を用いて問合せを変形する．制約述語を定める節を求める方法を議論し，この方法が従来の方法では効果のなかった非線形問合せや相互再帰型の問合せにも有効であることを示す．

1. まえがき

演繹データベースにおける再帰問合せの処理は多くの方法が提案されている[5]が，SLD 導出に基づく下降法（トップダウン）と最小不動点を計算しその部分集合として解を求める上昇法（ボトムアップ）に大別される．演繹データベースでは確定節の集合を扱うが，確定節集合においては，その最小エルブラン・モデルと SLD 反駁可能な具象基本式の集合，最小不動点の3者が一致するので[8),13)]，下降法と上昇法のどちらの方法によっても，問合せの解を求めることができる．下降法と上昇法にはそれぞれ特徴があり，どちらが優れているか容易には決められないが，関係データベース処理との親和性の点では上昇法が優れている．どちらの方法も初期の単純な方法には効率などの点で問題があり，各種の改良が提案されている[5]．

下降法の基本としては Prolog のような論理型言語処理系と同様な方法がある．この方法には問合せによっては無限ループになったり，節の本体の述語の評価順序が固定されているなどの問題があり，これらを改良した QSQ (Query/Subquery) 法[11]などが提案されている．

上昇法は最小不動点を計算することに基づいているが，単純な方法には効率的な問題があり，各種の改良が提案されている[5]．改良を行う方法のうちに問合せの目標中の定数（選択条件）を利用して問合せを変形する選択の分配法[1]やマジック集合法[4]などの方法がある．これらの方法は問合せによっては大きな効果があるが，特定の形の問合せにしか効果がないという問題があった．

本稿では制約述語と呼ぶ述語を用いて問合せをより小さな最小不動点をもつ問合せに変換する方法を提案する．本方法により最小不動点の大きさを変換してから計算を行うことにより，効率の良い問合せ処理を行うことができる．本方法にはマジック集合法と同様に節数の増加などの問題があるが，データベースが大きいときはこれらを考慮しても大きな効果が得られる．また本方法は選択の分配法やマジック集合法と比較しより一般的な問合せに効果がある．

2. 最小不動点と制約付最小不動点

2.1 演繹データベースと最小不動点

演繹データベース D は，確定節の有限集合であり，具象単位節からなる外延データベース (EDB) と，それ以外の節からなる内包データベース (IDB) とから構成される．すなわち，

D≡確定節集合
$=\mathrm{IDB}\cup\mathrm{EDB}$.

$A, A_1, \cdots, A_n$ を基本式としたとき確定節を，

$A \leftarrow A_1, \cdots, A_n$

と記述する．確定節の左辺を頭部，右辺を本体と呼ぶ．本体が空の確定節を単位節と呼ぶ．以下では確定節を節と略記する．同じ述語記号を頭部にもつ節により1つの関係が定義されると考える．問合せは一般に論理式で記述されるが，本論文では議論を簡単にするため，(D に現れる述語記号をもつ) 基本式 g で記述されると考える．以下では g を目標と呼ぶ．g' を g

† Restricted Least Fixpoints in Deductive Databases by NOBUYOSHI MIYAZAKI (Systems Laboratory, Oki Electric Industry Co. Ltd.) and HIDENORI ITOH (Research Center, Institute for New Generation Computer Technology).

†† 沖電気工業(株)研究開発本部総合システム研究所
††† (財)新世代コンピュータ技術開発機構研究所
* 現在 名古屋工業大学電気情報工学科

を具象化した基本式とするとき，問合せの解は，⇒ が論理的帰結を表すとして $G=\{g'|g'$ は g の具象例かつ $D\Rightarrow g'\}$ である．ここで与えた定義は論理プログラムの定義と類似しているが，演繹データベースではEDB が大きい場合を問題にするため異なった処理の方法が検討されている[5]．

演繹データベースの問合せと関係データベースの問合せを比較すると，目標と IDB を合わせたものが関係データベースの問合せに対応している．したがって多くの文献で目標と IDB を合わせて問合せと呼んでおり[5]，本稿でも問合せの変形と言うときは IDB の変形を意味する．

確定節集合 D に対し関数 T_D が次のように定義される[8),13)]．基本式の具象例の任意の集合 I に対し，

$T_D(I)=\{A|A\leftarrow A_1,\cdots,A_n$ が D の確定節の具象例で，かつ $\{A_1,\cdots,A_n\}\subset I\}$．

このとき，$I=T_D(I)$ となる I を T_D の不動点という．また任意の D について，T_D の不動点のうち最小のもの，すなわち最小不動点 M_D が一意に定まる．最小不動点は I_0 を空集合とし，$I_{i+1}=T_D(I_i)$ なる I_i の列の極限である．また，最小不動点は最小エルブラン・モデルすなわち D の論理的帰結である具象基本式の集合，と一致する[8),13)]．

最小不動点の性質から M_D 中から目標 g を具象化した式の集合を選択すれば問合せの解 G と一致するので，M_D を求めれば G が求まる．このようにして問合せの解を求める方法が上昇法である．一般には M_D が無限集合になる場合があるが，D に関数がなければ M_D は有限集合になり，I_i から，I_{i+1} を計算することを繰り返し，変化しなくなればそのときの I_i の値が最小不動点である．T_D は単調増加なのでこの繰り返しは必ず有限回で終了し解が求まる．このため，本稿では関数はないと仮定しこの繰り返し演算を不動点演算と呼ぶ．また議論を簡単にするため，IDB の節の頭部に現れる述語は EDB には現れない．すなわち IDB と EDB は述語を共有しないと仮定する．

演繹データベースでは上記の定義をそのまま用いるのでなく，比較述語や算術演算述語を導入して拡張することが多い．このような場合，最小不動点の有限性を保証するため上昇評価可能と呼ばれる条件を仮定することが多い[5]．このような仮定のもとで T_D を関係代数式に変換してから計算を行うことにより不動点演算の実現が容易になる[3),5),6),9)]．

[例1 a] 先祖関係についての問合せ

目標：ancestor $(X,$ taro$)$

IDB：

ancestor $(X,Y)\leftarrow$parent (X,Y)

ancestor $(X,Y)\leftarrow$parent (X,Z), ancestor (Z,Y)

ここで親子関係 (parent) は EDB にあるとする．この問合せは関係代数を用いて，

answer $=\sigma_{2=\text{taro}}$ (ancestor)

ancestor $=$ parent $\cup$

$\pi_{1,4}$ (parent $\times_{2=1}$ ancestor)

と表せる．ここで σ_F は選択を，π は射影，$\cup$は和，$\times$は結合を表す．この問合せは，不動点演算により先祖関係 (ancestor) を最小不動点として求めてから選択を行うことにより解が求められる．

2.2 不動点演算の問題点と制約付最小不動点

前節で説明した不動点演算にはいくつかの問題がある．大きな問題点は以下の2つである．

(1) 繰り返しの中で何度も同じ要素が計算され，計算が冗長で効率が悪い．

(2) 目標中に定数があっても最小不動点全体を計算してしまい，効率が悪い．下降法のように目標中の定数を生かして全体を計算することなく一部のみの計算で解が求められれば効率が向上する．

これらの問題点を改良する方法がいくつか提案されている．まず第1の問題については T_D が単調増加であることに着目し，繰り返しごとに各関係の増分を求め，その増分を用いて次の結果を計算すれば効率化が可能である．この方法は差分計算法や，セミナイーブ評価法と呼ばれる[2),3)]．第2の問題については関係データベースにおける問合せ処理の最適化[10)]と同様に，問合せ，すなわち IDB の節を変形してから最小不動点を計算する方法がある．代表的な方法としては選択の分配法[1)]やマジック集合法[4)]がある．第1と第2の問題をそれぞれ改良する方法を組み合わせることにより全体として効率の良い方法が得られる．本稿では第2の問題を改良する方法を議論する．

まず，第2の問題を解決する方法の例として選択の分配法を述べる．議論を簡単にするため，IDB の節の頭部には1種類の述語記号しか現れないとする．このとき，EDB 中の関係は固定されているので不動点式は $r=f(r)$ と書ける．問合せの解は目標中の定数を選択に対応させると $\sigma_F r$ である．σ_F を式の両辺にかけると，$\sigma_F r=\sigma_F f(r)$ になる．もし右辺を変形し，$\sigma_F f(r)=f'(\sigma_F r)$ となるような f' が存在すれば，$\sigma_F r=f'(\sigma_F r)$ となる．したがって r でなく，$\sigma_F r$ の

最小不動点が直接計算できる.

[例1 b] 例1 aの問合せでの選択の分配

answer＝$\sigma_{2=taro}$ (ancestor)

ancestor＝parent ∪

$\pi_{1,4}$ (parent×$_{2=1}$ ancestor)

2番目の式の両辺に $\sigma_{2=taro}$ をかけると,

$\sigma_{2=taro}$ (ancestor)＝$\sigma_{2=taro}$ (parent ∪

$\pi_{1,4}$ (parent×$_{2=1}$ ancestor))

＝$\sigma_{2=taro}$ (parent) ∪

$\pi_{1,4}$ (parent×$_{2=1}$

($\sigma_{2=taro}$ (ancestor)))

となり, $\sigma_{2=taro}$ (ancestor) の不動点式が得られ, 問合せの解を直接最小不動点として求めることができる. もとの最小不動点は先祖関係全部を含むのに対し, 変換後の最小不動点には taro の先祖の部分しか含まれないので後者の計算のほうが効率が良い.

選択の分配法は関係データベースにおける選択の交換[10]と類似しているが, 再帰問合せでは常にこの方法が適用できるとは限らない. たとえば, 例1 aの目標の定数の位置が異なり ancestor (taro, X) の場合はこの方法では効率化ができない.

本稿では選択の分配が可能でない場合にも最小不動点を小さくし, 効率化が可能になるような問合せの変形を行い, 上記の第2の問題を解決する方法を提案する. 選択の分配では σ_F を直接用いて式を変形したが, 本稿では σ_F の代わりに制約子と呼ぶ選択演算子 σ^* を導入する. この方法では, $r=f(r)$ を $r=\sigma^*f(r)$ に変形する. この式の最小不動点を制約付最小不動点と呼ぶ. 問合せの解は制約付最小不動点 r を求めれば $\sigma_F r$ として求まる. この変形が有効であると考えられる理由は以下のものである.

(1) 制約付最小不動点は元の最小不動点の部分集合になり, これよりも小さい.

(2) 選択の分配が可能な場合は, $\sigma^*=\sigma_F$, すなわち $r=\sigma_F f(r)$ とすれば制約付最小不動点より求めた解はもとの最小不動点から求めた解と一致する.

(3) 下降法の一種である QSQ 法[11]では, 解の集合とともに副目標の集合を求めている. σ^* を副目標の集合に対応するものと考えると, この変形は以下に示すように QSQ 法で節の本体を評価するとき副目標の集合により条件づけていることに対応する.

(4) マジック集合法での変形は, σ^* をマジック集合に対応させると, 後述のようにこの変形と類似している.

ここで, 選択の分配は特別の場合しか適用できず, またマジック集合法は線形問合せと呼ばれる簡単な問合せにしか効果がない. これに対し QSQ 法は相互再帰を含むような複雑な問合せにも適用できる.

制約子とその構成方法を QSQ 法を参考にして説明する. このため, まず QSQ 法を述べる. SLD 導出に基づいた Prolog の処理では目標(節)の反駁を行う過程で, 順次副目標の反駁が必要になり, これらの副目標の反駁の列により解が求まる. Prolog ではこれらの副目標に対する反駁結果を1つ1つ計算している. QSQ 法では副目標とその反駁結果の集合を扱う. すなわち, おのおのの述語記号に対応する副目標の集合を保持し, またこれらに対する反駁結果の集合を述語記号ごとに保持しつつ SLD 導出を行う. 1つ1つの要素だけでなくその集合も扱うことにより, Prologでおこる無限ループを避けて任意の問合せに対し正しい解を求めることができる. QSQ 法ではさらに, Prolog のように本体中の述語の評価順序を固定せず, 副目標のどの引数が束縛されているかによって評価順序を選択することにより処理の効率化を図っている.

QSQ 法と同様な効果を上昇法で得るには以下のような方法が考えられる. 例として例1 aのうちの以下の節の処理を考える.

ancestor(X, Y)←parent(X, Z), ancestor(Z, Y)

目標の反駁を行う過程で副目標を対象とする導出の結果, 節の本体が反駁の対象式の一部になる. たとえば, ←r, A を反駁するため, 副目標 r と同じ述語記号を頭部にもつ節 r'←R を用いて導出を行うと, ←R', A' を得る. ここで R' と A' は r と r' との単一化により R と A をそれぞれ条件づけた式である. したがって導出の対象となった副目標は新しい反駁対象に対する制約条件に転化したと考えることができる. この例では ancestor を頭部にもつ節を導出に用いるときは必ずこのように副目標により条件づけられる. したがってこれらの節の本体をあらかじめ副目標の集合で条件づけておいても解は変化しない. すなわち ancestor に関する副目標の集合に対応する述語 ancestor* を本体に加えて以下の節に変形しても解は変化しない.

ancestor (X, Y)←ancestor* (X, Y),

parent (X, Z), ancestor (Z, Y)

ここで, 本体の ancestor* の引数は頭部の引数と同じとする, この変形は $r=f(r)$ の $r=\sigma^*f(r)$ への変形を節形式で表現したものに対応している.

次に副目標集合をどのように節で表現するかを検討する．まず最初の目標自身は副目標集合の要素である．これは目標の述語記号を制約述語で置き換えた単位節で表現できる．次に反駁の過程で副目標集合がどのように変化するかを考察する．ancestor を副目標として導出を行ったときの導出に用いた節の本体が次の反駁の対象になる．したがって反駁の過程では，たとえば ancestor* (X, Y), parent (X, Z), ancestor (Z, Y) が反駁の対象になる．ただしここで ancestor* (X, Y) は反駁対象でなく制約条件と考える．もしこれを左から評価して parent (X, Z) を評価したあと ancestor (Z, Y) を評価するとすれば，parent (X, Z) を評価した結果が次に ancestor (Z, Y) を反駁するときの制約条件になる．したがって次の ancestor に関する副目標の制約条件は ancestor* (X, Y), parent (X, Z) である．不動点意味論では節の頭部が本体の関数であると考えるので，このことは以下のように新しい制約条件を本体にし頭部に制約述語をもつ節により表現できる．

ancestor* (Z, Y)←ancestor* (X, Y),
parent (X, Z)

なお，ここで副目標に対応する述語 ancestor* の具象例はあらかじめわかっているわけではなく，変形後の問合せに対する不動点演算の過程で EDB の内容に依存して動的に定まるものである．以上の考察により問合せ全体を変形した例を以下に示す．

［例２］ 例１ a の先祖と同じ IDB で目標を ancestor (taro, X) とする．

まず ancestor の制約述語を本体に追加すると以下を得る．

ancestor (X, Y)←ancestor* (X, Y),
parent (X, Y)

ancestor (X, Y)←ancestor* (X, Y),
parent (X, Z), ancestor (Z, Y)

制約述語を頭部にもつ節として以下のような節が得られる．

まず目標に対応する節は単位節 ancestor* (taro, X) である．次に ancestor を本体にもつ節は１つなのでこの節から上記のように次の節が生成される．

ancestor* (Z, Y)←ancestor* (X, Y),
parent (X, Z)

この変換結果はもとの問合せと等価である．

3. 制約子による変換と制約節の決定

前章では制約子の概念とその構成方法を例で議論した．本章では制約子による変換を定義しその性質を論じるとともに制約子の構成方法を示す．

3.1 変換の基本アルゴリズム

まず，演繹データベースの等価性を定義する．演繹データベース D と D' の目標 g に対する解をそれぞれ G, G' とする．$G=G'$ のとき，D と D' は g に関して等価であるという．

$D=$IDB∪EDB が与えられたとする．IDB の各節は，

$$r_k \leftarrow h_k\ (r_1, \cdots, r_n) \qquad k=1, \cdots, n \qquad (1)$$

と表せる．ここで，r_i は IDB の節の頭部に現れる述語であり，h_k は基本式の連言である．h_k には r_i 以外に EDB の述語も含んでいる．このとき制約子による変換を以下のように定義する．

［制約子による変換］

1. IDB の節，式(1)をすべて次の形に変形する．

$$r_k \leftarrow r_k^*,\ h_k\ (r_1, \cdots, r_n) \qquad k=1, \cdots, n \qquad (2)$$

ここで r_k^* はもとの D には現れない述語記号の基本式で，r_k の述語記号に対応して r_k^* の述語記号が定まるものとする．また r_k^* の引数は頭部の r_k の引数とすべて同一である．r_k^* を制約述語，式(2)を被制約節と呼ぶ．

2. 制約節，すなわち制約述語を頭部にもつ節を IDB に追加する．
3. ステップ１と２の結果を IDB′ とし，
$D'=$IDB′∪EDB とする．

制約子による変換は制約節の与え方によって定まる．与えられた目標 g に関してこの変換の結果 D' が D と等価なとき，すなわち両者の解が同一であるとき，この変換は g に関する等価変換であるという．等価変換になる制約節は各種考えられる．たとえば，引数にすべて異なった変数をもつ単位節を制約節とすれば，最小不動点を小さくする効果はないが等価変換が得られる．また選択の分配が可能なときは選択に対応する単位節，たとえば例１ a では ancestor* (X, taro) を制約節とすれば等価変換が得られる．

制約子による変換の定義から次の定理が得られる．

［定理１］ 演繹データベース D の制約子による変換結果を D' とする．任意に与えられた目標 g に関する D と D' の解をそれぞれ G と G' とする．この

とき $G \supset G'$ である．

（証明） IDB の節を $r \leftarrow R$，それを変換した被制約節を $r \leftarrow r^*$, R とする．$\{r \leftarrow R\} \Rightarrow r \leftarrow r^*$, R なので，D' の最小エルブラン・モデルのうちもともと D にあった述語記号に対応する部分は D の最小エルブラン・モデルの部分集合である．したがって $G \supset G'$.

この定理より次の系は明らかである．

［系］ D の制約子による変換結果 D' が目標 g に関し D と等価であったとする．D' の制約節 R に対し R の頭部と同じ述語を頭部にもつ節 Q を考える．もし $\{Q\} \Rightarrow R$ ならば D' の R を Q で置き換えても D と等価である．

なお，この系は1つの節でなく複数の節で置き換える場合にも成立する．系は等価変換の制約力を弱めても等価変換であることを意味している．たとえば制約節の本体の基本式を削除すると制約力は弱まるがやはり等価である．できるだけ制約力の強い制約節を用いたほうが最小不動点が小さくなり効率化の効果が大きい場合が多いが，制約節の本体の基本式の数が少ないほうが結合演算が減少し処理が効率的な場合もある．このような場合，系により制約節の本体の基本式を削除しても良いことが保証される．

等価変換となる制約節の与え方については前節で例を示したが，これを一般化したものを以下に示す．

［制約節決定アルゴリズム］

目標 g と演繹データベース D が与えられているとする．以下の手順により制約節を生成する．

1. 目標 g の述語記号を制約述語記号で置き換え引数を同じとして得られる基本式からなる単位節を制約節とする．これを初期節と呼ぶ．
2. D における IDB から被制約節（式(2)）の集合が一意に定まる．各被制約節から以下のように制約節を生成する．
   a． 被制約節は
   $$r_k \leftarrow r_k^*,\ h_k\ (r_1, \cdots, r_n)$$
   と表せる．各被制約節の本体の h_k 中の基本式の順序を入れ換えて得られる節を各被制約節に対し1つずつ選び，その集合を IDB^+ とする．
   b． IDB^+ の要素は
   $$r_k \leftarrow r_k^*,\ h_k^+\ (r_1, \cdots, r_n)$$
   と表せる．h_k^+ には IDB の節の頭部にある述語記号をもつ基本式が合計 m 個あるとする．この節から m 個の制約節を次のように生成する．

   h_k^+ の中の1つの基本式 r_i に着目するとこの節は，
   $$r_k \leftarrow r_k^*,\ h_k^{+}{}_{\text{left}}\ (r_1, \cdots, r_n),\ r_i, h_k^{+}{}_{\text{right}}\ (r_1, \cdots, r_n)$$
   と表せる．r_i^* を述語記号が r_i の述語記号に対応し引数が r_i と同一の基本式とする．この節から次の制約節を作る．
   $$r_i^* \leftarrow r_k^*,\ h_k^{+}{}_{\text{left}}\ (r_1, \cdots, r_n)$$
   このようにして，IDB^+ の1つの節から本体の m 個の r_i に対応して m 個の制約節が生成される．

   なお，本体に同じ述語記号の基本式が複数あったときはそれぞれに対応する制約節を生成する．

ここで IDB^+ の選び方は任意であるが，どのように選ぶと効率が良いかは後述する．前述のように本稿では演繹データベースには関数がなく，また IDB と EDB は述語を共有しないと仮定している．この条件のもとで次の定理が成り立つ．

［定理2］ 演繹データベース D と目標 g が与えられたとする．制約節決定アルゴリズムによって定まる制約節の集合を用いた制約子による D の変換の結果を D' とする．D と D' は g に関して等価である．

証明は付録1に示す．

3.2 制約子による変換の改良

本節では制約述語の引数の扱いの改良と制約節決定アルゴリズムのステップ2aでの被制約節の本体中の式の順序の選び方について述べる．

まず，引数の扱いに関しては以下の問題がある．不動点意味論では節の頭部が節の本体の関数であると考える．このとき例2の初期節のように節の本体にない変数が頭部にある場合，すなわち領域限定でない場合は，その変数は任意の値をとりえる．したがってこのような変数が制約述語を頭部とする節にある場合は，最小不動点のその述語に対応する部分が大きくなる．またこのような場合には関数 T_D を関係代数によっては表せない．この点を改良するには，不動点演算でこのような変数を特別な扱いをするか，制約述語の引数の数を変更してこのような変数を除去するか2つの方法が考えられる．ここでは後者の方法を述べる．

一般に制約述語の引数の数を減らすことは該当する変数を自由変数とみなし無視することであり，制約力を弱めることに対応する．したがって定理1の系によ

り引数を減らしても等価変換としての性質は失われないことを示すことができる．このため制約述語の任意の引数を削除できる．

次に前節のアルゴリズムでのステップ 2a での被制約節の本体の基本式の順序の選び方について述べる．この順序をどう与えるかによって最小不動点の大きさが変化するので不動点演算を適用するときの効率が変化する．

たとえば，例2では被制約節の本体は左から評価すると考えて以下の制約節を得た．

ancestor* (taro, X)
ancestor* (Z, Y)←ancestor* (X, Y),
parent (X, Z).

ここで，最初の制約節には本体がなく第2引数は領域限定でないので削除すると，以下を得る．

ancestor* (taro)
ancestor*(Z)←ancestor*(X), parent (X, Z).

これに対し，順序づけにおいて右から評価するとし，第2引数を削除すると次のようになる．

ancestor* (taro)
ancestor*(Z)←ancestor*(X).

第2の組を用いても変換は等価であるが，第2節の Z が領域限定でない．したがってこの問合せの変換に用いる制約節としては第1の組のほうが第2の組よりも優れている．

逆に例1のように目標の第2引数が定数の場合は，左からの評価では制約述語の第1引数を削除すると，

ancestor*(Y)←ancestor*(Y), parent (X, Z)

となる．右からの評価では，

ancestor*(Y)←ancestor*(Y)

となる．この2つの節を比較すると両方とも領域限定である．しかし最初の節の本体の parent はその真偽にのみ意味があり，頭部の引数を決定するのに貢献していない．またこの式は X も Z も条件がついていないので評価する処理量が大である．したがってこの場合は第2の節のほうが優れている．

上記の議論から，制約子の効果を大きくするには，各制約節の頭部の変数が本体に多くあるように，また貢献しない式は本体に現れないようにステップ2aでの順序づけを行うのが良い．これらの点を考慮して制約節決定アルゴリズムを改良すると以下のようになる．

[改良型制約節決定アルゴリズム]

1. a．目標から初期節を定める．初期節に変数があるときは引数の数を減少させ変数を除去する．
   b．各節にある対応する制約述語に変更マークをつけ必要ならその引数の数を変更する．
2. 変更マークのついた制約述語を本体にもつ被制約節から以下のように制約節を生成する．
   a．被制約節の本体を次のように並べ換える．
   (i) 制約述語を一番左とする．
   (ii) 各基本式の引数の少なくとも1つの引数の変数がその式より左の式の引数となる．ただしどのような順序でもこの条件を満たせない式がある場合はその式は右に位置づける．
   b．元のアルゴリズムのステップ 2b と同じ．
   c．生成された制約節の頭部にその節の本体にない変数があるときは，制約述語の引数の数を変更し変数を除去する．
   d．ステップ 1b と同じ．
   e．未処理の被制約節があれば 2a から繰り返す．

なお，このアルゴリズムでは変更マークのつかない被制約節からは制約節は生成されないが，このような場合が生じるのはもともと問合せの解に影響しない節があった場合であり，これらの節は無視できる．

[例1c] 例1aの問合せを改良型アルゴリズムによって変換すると，以下のようになる．

(1) 初期節は目標の第2引数が定数なので第1引数を削除して ancestor* (taro) になる．

(2) 被制約節は，

ancestor (X, Y)←ancestor*(Y), parent (X, Y)
ancestor (X, Y)←ancestor*(Y), parent (X, Z),
ancestor (Z, Y)

になる．

(3) 被制約節から生成される制約節は，ステップ2aの順序づけから，

ancestor*(Y)←ancestor*(Y)

になる．この結果は上記の議論によるものと同一である．

この変換で得られた最後の節は同義反復なので削除しても良い．この結果を用いて最小不動点を計算すると，例1bの制約の分配と同様に taro の先祖の部分しか含まないので効率が向上する．この変換結果は制約述語をマジック述語と考えるとマジック集合法の結果と同等である．また例2についても同様になる．

4. 制約子による変換の適用例

制約子による変換は前章で示したような簡単な問合せに対しては通常マジック集合[4]と同等になる．本章では制約の分配やマジック集合では効果のない複雑な問合せ，すなわち節の本体に同じ再帰述語が2度現れたり，相互再帰型の問合せに対して本方法が効果のあることを例によって示す．

［例3］ 非線形の先祖問合せ

目標：ancestor (taro, X)

IDB：

ancestor (X, Y)←parent (X, Y)

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

この問合せを改良型制約節決定アルゴリズムによって変換すると，

ancestor (X, Y)←ancestor*(X), parent (X, Y)

ancestor (X, Y)←ancestor*(X), ancestor (X, Z), ancestor (Z, Y)

ancestor* (taro)

ancestor*(X)←ancestor*(X)

ancestor*(Z)←ancestor*(X), ancestor (X, Z)

を得る．ここで制約節のうち1つは同義反復なので不要である．この変換結果の最小不動点には taro の子孫しか含まないので効率の良い計算ができる．

次にもっと複雑な問合せの例を示す．

［例4］ 非線形相互再帰型問合せ

目標：$p(j, X)$

IDB：

$p(X, Y)$←$a(X, Y)$

$p(X, Y)$←$p(X, Z1)$, $a(Z1, Z2)$, $q(Z2, Y)$

$q(X, Y)$←$b(X, Y)$

$q(X, Y)$←$p(X, Z1)$, $c(Z1, Z2)$, $q(Z2, Y)$

で a, b, c は EDB で定義されているとする．ここでは，p と q が相互に依存しあっている．これを変換し同義反復を除去すると以下の節集合を得る．

$p(X, Y)$←$p^*(X)$, $a(X, Y)$

$p(X, Y)$←$p^*(X)$, $p(X, Z1)$, $a(Z1, Z2)$, $q(Z2, Y)$

$q(X, Y)$←$q^*(X)$, $b(X, Y)$

$q(X, Y)$←$q^*(X)$, $p(X, Z1)$, $c(Z1, Z2)$, $q(Z2, Y)$

$p^*(j)$

$p^*(X)$←$q^*(X)$

$q^*(Z2)$←$p^*(X)$, $p(X, Z1)$, $a(Z1, Z2)$

$q^*(Z2)$←$q^*(X)$, $p(X, Z1)$, $c(Z1, Z2)$.

この問合せの実行例を以下に示す．なお，実行は上昇法で問合せを評価する演繹データベース実験システム PHI/K[2][12] に上記の2種類の問合せを与えて行った．

EDB には以下のデータがあるものとする．

$a(i, j)$. $a(j, k)$. $a(h, k)$. $a(t, s)$. $a(k, m)$.

$b(h, i)$. $b(k, t)$. $b(j, h)$. $b(s, o)$.

$c(j, k)$. $c(t, h)$. $c(o, j)$.

$p(j, X)$ を目標としてもとの問合せの不動点演算を行うと5回目の繰り返しで収束し，下記の結果を得る．

$p(i, j)$. $p(j, h)$. $p(h, k)$. $p(t, s)$. $p(k, m)$.

$p(i, i)$. $p(j, t)$. $p(i, h)$. $p(j, o)$. $p(i, t)$.

$p(i, o)$.

$q(h, i)$. $q(k, t)$. $q(j, h)$. $q(s, o)$. $q(i, t)$.

$q(j, i)$. $q(i, i)$. $q(i, h)$.

この結果，この問合せの解は $X=\{h, t, o\}$ である．制約子による変換の結果を用いて不動点演算を行うと以下の結果を得る．なお各ループで新しく生成される結果のみ示し，"—" は変化のないことを意味する．

	p^*	p	q^*	q
第1ループ	$p^*(j)$	{ }	{ }	{ }
第2ループ	—	$p(j, h)$	{ }	{ }
第3ループ	—	—	$q^*(k)$	{ }
第4ループ	$p^*(k)$	—	—	$q(k, t)$
第5ループ	—	$p(j, t)$ $p(k, m)$	—	—
第6ループ	—	—	$q^*(s)$	—
第7ループ	$p^*(s)$	—	—	$q(s, o)$
第8ループ	—	$p(j, o)$	—	—
第9ループ	—	—	—	—

この結果をまとめると，

$p^*(j)$. $p^*(k)$. $p^*(s)$.

$p(j, h)$. $p(k, m)$. $p(j, t)$. $p(j, o)$.

$q^*(k)$. $q^*(s)$.

$q(k, t)$. $q(s, o)$.

となり，目標 $p(j, X)$ の解は $X=\{h, t, o\}$ でありもとの解と一致する．この例では EDB が小さいので変換による効率向上の効果は小さいが，EDB が大きいほどその効果は大きくなる．

5. 処理性能

本章では制約子による変換と他の方法の性能を比較する．従来の方法の代表的なものについては既に比較

されている[5]ので，ここではそれを基礎として本方法の性能を検討する．以下の議論では性能の尺度は計算中で生成される中間結果のタプル数であり，各方法の比較数値は 10 万タプルの親子関係が EDB にあるものとした場合のものである[5]．

代表的な問合せ処理方法としては HN[7]，不動点演算（ナイーブ評価[3]），差分計算またはセミナイーブ評価[2),3]，QSQ[11]（これには再帰型の QSQR と繰り返し型の QSQI がある），マジック集合[4]（MS と略す），カウンティング[4](C)などがある．HN と QSQ は下降法であるが，他は上昇法のアルゴリズムである．

例 1 や例 2 のような線形な問合せの場合は，EDB の内容によって性能の良否の関係が変化するが大略以下のような関係になる[5]．

{C, HN} ≪ {QSQR, MS}
≪QSQI ≪セミナイーブ≪ナイーブ

ここで {A, B} はAとBがほぼ同等なことを示す．また≪は 1 桁またはそれ以上の差を表す．なお Prolog を演繹データベース処理系とみなした場合，問合せ中の定数の位置や EDB の内容により性能が変化するが，最良で QSQR と同等で最悪では不動点演算よりも効率が悪い．これは Prolog では節の本体の評価順序が固定されていることと，冗長な計算が多いことがあるためである．また，例 1 のように選択の分配が可能な場合，選択の分配を行ってから最小不動点を計算すれば，QSQR や MS とほぼ同等になるが，例 2 のように分配できない場合はセミナイーブ評価と同等である．

性能の向上の効果は EDB の大きさや内容によるが，上記の条件で先祖関係の問合せでは，MS はセミナイーブ評価に比較し 1 万倍程度[5]の性能向上がある．MS での不動点演算はセミナイーブ評価を用いているので，この向上はマジック集合による計算領域縮小の効果である．MS や制約子による変換では節数の増加や結合演算の増加という問題があるが，この結果は EDB が大きければこれらの点を考慮しても大きな効率化が図れることを示している．

例 3 や例 4 のような非線形な問合せではカウンティングや HN 等の効率の良い方法は適用できない．また Prolog も無限ループになってしまい適用できない．適用可能な方法の比較は大略以下のようになる．

QSQR≪QSQI≪{セミナイーブ，NS}
≪ナイーブ

ここで MS は効率化の効果はなくセミナイーブ評価と同等である．例 3 の非線形先祖関係の問合せの場合は QSQI で MS と比較し 1 万倍程度，QSQR ではさらに 10 倍程度の向上がある[5]．

制約子による変換の効果をこれらの方法と比較すると以下のようになる．例 1 や例 2 のような簡単な問合せでは制約子による変換結果は MS と同等なので，差分計算と組み合わせればその性能は MS および QSQR と同等になる．

例 3 や例 4 のような非線形問合せの場合には次のようになる．制約子による変換では 2 章で述べた制約節の構成方法から，変換結果の最小不動点のうち制約述語に対応する部分は QSQ の副目標集合に，もとの述語に対応する部分は QSQ の解集合とそれぞれ同一である．QSQI ではナイーブ評価と同様に冗長な計算が多いという問題がある．本方法では変換結果の不動点演算を行う際の冗長性については考慮していないが，セミナイーブ評価や差分計算はナイーブ評価よりも冗長な計算が少ない．このため本方法と差分計算等を併用すると，その性能は例 3 や例 4 では QSQR と QSQI の中間になると推定される．したがって本方法は制約の分配や MS よりも広い範囲の問合せに対して有効であり，非線形先祖関係の問合せではこれらと比較し 1 万倍程度の性能向上が期待できる．

制約子による変換は問合せのコンパイルを行ったあと上昇法で評価するので，関係演算は QSQR のインタプリティブな方法より高速化できる．このため，本方法のほうが QSQR より効率を良くできる可能性がある．

6. むすび

最小不動点の大きさを変えることにより上昇法の効率化を図る制約子による変換の基本原理と制約子の構成方法を提案しその例を示した．また提案した方法の性能について考察を行った．本方法により非線形な問合せや相互再帰を含む問合せも効率良く処理できる．しかし 3.2 節で述べた引数の扱いや順序決定の問題については節中の引数間の関連が複雑な場合には改善の余地があると考えられる．また簡単な問合せに対しては本方法よりも効率の良いカウンティングなどの方法が存在する．文献 5）でも述べられているように 1 つの方法で常に最も効率の良いことを保証するのは困難である．このため，問合せによって用いるアルゴリズムを選択する方法が良いと考えられる．上昇法で問合せの変換を用いる方法では，問合せによって変換を

選択したり，ある方法による変換の結果に別の変換を行うことが容易である．今後は本方法の改良と実現評価を行うとともに他の方法，たとえば選択の分配[1]や導出による変換[9),14)]，との組み合わせによる問合せ処理の最適化の検討を行う予定である．

謝辞 本研究は第5世代コンピュータ・プロジェクトにおける KBMS PHI 研究の一環である．研究に際しご支援ご指導をいただいた ICOT 研究所の渕一博所長，沖電気工業(株)総合システム研究所知識情報処理研究所の羽下雄之輔部長，および本稿の作成に際しご意見ご助言をいただいた PHI 研究の関係各位に感謝します．

参考文献

1) Aho, A. and Ullman, J.: Universality of Data Retrieval Languages, *Proc. of ACM POPL*, pp. 110-120 (1979).
2) Balbin, I. and Ramamohanarao, K.: A Differential Approach to Query Optimization in Recursive Deductive Databases, Univ. of Melbourne, Technical Report 86/7 (1986).
3) Bancilhon, F.: Naive Evaluation of Recursively Defined Relations, in (eds.) Brodie, M. L. and Mylopoulos, J., *On Knowledge Base Management Systems*, pp. 165-178, Springer-Verlag (1986).
4) Bancilhon, F., Maier, D., Sagiv, Y. and Ullman, J.: Magic Sets and Other Strange Ways to Implement Logic Programs, *Proc. of 5th ACM PODS*, pp. 1-15 (1986).
5) Bancilhon, F. and Ramakrishnan, R.: An Amateur's Introduction to Recursive Query Processing Strategies, *Proc. of ACM SIGMOD*, pp. 16-52 (1986).
6) Ceri, S., Gottlob, G. and Lavazza, L.: Translation and Optimization of Logic Queries: The Algebraic Approach, *Proc. of 12th VLDB*, pp. 395-402 (1986).
7) Henschen, L. and Naqvi, S.: On Compiling Queries in Recursive First-Order Databases, *JACM*, Vol. 31, No. 1, pp. 47-85 (1984).
8) Lloyd, J. W.: *Foundations of Logic Programming*, Springer-Verlag (1984).
9) Miyazaki, N., Yokota, H. and Itoh, H.: Compiling Horn Clause Queries in Deductive Databases: A Horn Clause Transformation Approach, ICOT Technical Report, TR-183 (1986).
10) Ullman, J. D.: *Principles of Database Systems*, Second ed., Computer Science Press (1982).
11) Vieille, L.: Recursive Axioms in Deductive Databases: The Query/Subquery Approach, *Proc. of 1st International Conf. on Expert Database Systems*, pp. 179-193 (1986).
12) 阿比留幸展，羽生田博美，宮崎収兄，森田幸伯 KBMS PHI: 演繹データベース実験システム PHI/K[2]，第34回情報処理学会全国大会論文集，No. 3 K-8, pp. 1491-1492 (1987).
13) 野口正一，滝沢 誠：知識工学基礎論，オーム社，東京 (1986).
14) 宮崎収兄，羽生田博美，伊藤英則：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 (1988).

付録 定理2の証明

もとの節集合 のSLD 木の任意の節点に対応する節点が変換後の節集合の SLD 木に存在することを示す．このような節点が存在すれば，もとの SLD 木のすべての成功点が変換後の SLD の成功点となるので，D の解が D' の解になる．逆は定理1より明らかなので D' が g に関して D と等価になる．

SLD 導出における導出の適用順序（計算規則）を D, D' とも Prolog と同様に左から右に評価するものとする．また D, D' とも節の本体を制約節を生成するときに用いた被制約節の本体の基本式の順序と同じになるように並べ換えておく．SLD 導出は計算規則とは独立に完全かつ健全であるのでこのようにしても一般性を失わない．以上の準備のもとに証明を行う．

［1］ 目標の反駁．

（a） D における反駁

$\leftarrow g$ と g を頭部にもつ節 $g \leftarrow A$ から，$\leftarrow A$ に対応する節点を得る．

（b） D' における反駁

g と g を頭部にもつ $g \leftarrow g^*, A$ から $\leftarrow g^*, A$ に対応する節点を得る．g^* の初期条件から g^* の反駁は成功し，$\leftarrow A$ に対応する節点を得る．

［2］ D の SLD 木での節点 N に対応する節点 N^* が D' の SLD 木に存在すると仮定する．

（a） D における反駁

N が $\leftarrow r, A$ に対応するとする．r と節 $r \leftarrow R$ から $\leftarrow R, A$ を得る．

（b） D' における反駁

N^* は $\leftarrow r, A$ に対応する．r と節 $r \leftarrow r^*, R$ から $\leftarrow r^*, R, A$ を得る．ここで r^* を副目標とする

反駁は必ず成功し，$\leftarrow R$, A に対応する節点が得られる．これは次の理由による．

N^* に対応する$\leftarrow r$, A に至るまでの経路で，r を本体にもつ節 $p \leftarrow p^*$, L, r, B が反駁に用いられている．この節による導出の結果が$\leftarrow p^*$, L, r, A だったとする．このとき N^* に到達するのは r より左の p^*, L の反駁が成功した結果である．D' には制約節 $r^* \leftarrow p^*$, L が存在する．したがって，この節を副目標 r^* の反駁に用いれば必ず成功し，$\leftarrow R$, A に対応する節点を得る．

[1]と[2]より節点の目標からの経路長に関する数学的帰納法により D の SLD 木の節点に対応する節点が D' の SLD 木に存在する．

q. e. d.

(昭和62年6月1日受付)
(平成元年7月18日採録)

宮崎 収兄（正会員）

昭和48年京都大学理学部卒業．同年沖電気工業(株)入社．昭和54年イリノイ大学大学院計算機科学科修士課程修了．昭和57年より3年間(財)新世代コンピュータ技術開発機構に勤務．現在，沖電気工業(株)総合システム研究所に勤務．データベースおよび知識情報処理システムの研究開発に従事．電子情報通信学会，人工知能学会，ACM 各会員．

伊藤 英則（正会員）

昭和21年生．昭和44年福井大学工学部卒業．昭和49年名古屋大学大学院工学研究科博士課程修了．工学博士．昭和49年4月，日本電信電話公社横須賀電気通信研究所入社．昭和60年4月，(財)新世代コンピュータ技術開発機構に出向．平成元年2月，日本電信電話(株)情報通信処理研究所に復帰．同年7月より名古屋工業大学電気情報工学科教授．これまでに，オートマトンと数理言語の研究，計算機ネットワークシステムの研究開発，知識ベースシステムの研究開発に従事．電子情報通信学会，人工知能学会各会員．