非ホーン節を含む演繹データベースの問合せ処理の効率化

荒山正志[†,☆] 井上克巳[†]

近年，論理プログラミングの表現能力の拡張に関して多くの研究が行われ，その拡張のひとつの方向として非ホーン節の導入が議論されるようになった．非ホーン論理プログラムは，ホーン節の場合と比べて非常に高い知識表現能力を持つ．しかしながらこのプログラムに対して，効率的な問合せ処理を行うことは非常に難しく，マジックセット法などの従来の効率化手法はこれまで考案されていなかった．一方，定理証明の分野ではノンホーン・マジックセット法が提案されており，その成果を十分にあげている．この手法は，定理証明に関して非ホーン節を効率的に処理することができるが，データベースの問合せ処理に関しては不十分である．そこで本研究では，定理証明用のノンホーン・マジックセット法を拡張することにより，非ホーンデータベースに関する問合せ処理の手法を提案する．ここで，問合せ処理の解は，極小モデル意味論に依存するものとする．この意味論に基づいて，深さ優先と幅優先の各々について，確定解を求めるための変形と，可能解を求めるための変形を提案し，それぞれを実現した．さらに実験結果から，ノンホーンデータベースの問合せ処理に関して，提案した手法の実用性を確認することができた．

Optimizations of Query Processing for Disjunctive Deductive Databeses

MASASHI ARAYAMA[†,☆] and KATSUMI INOUE[†]

As one direction to extend the power of logic programs, the introduction of non-Horn clauses has been discussed. Non-Horn logic programs have more expressive power compared with Horn cases, but it is very difficult to have an efficient query processing method. Non-Horn clauses cannot be handled by traditional optimization methods for query processing such as Magic-sets. On the other hand, in the field of theorem proving, NHM (Non-Horn Magic-sets) is proposed. NHM can handle non-Horn clauses efficiently for theorem proving, but it is not sufficient for the query processing for non-Horn databases. Then, we propose a query processing method for non-Horn clauses by extending NHM. For the query processing, we assume the minimal model semantics. Along this semantics, we propose two query processing methods for both "definite answers" and "possible answers". Experimental results show that these query processing methods have a good performance and can be used practically.

1. はじめに

演繹データベース (deductive databases) は，一階述語論理を基本的な知識表現の枠組みとし，関係データベースの概念との統合により，知識ベースの基本的な機能を表現するものである．これまでに，演繹データベースの問合せ処理の高速化を実現する要素技術として，多くの問合せ処理アルゴリズムが研究されてきた[18]．演繹データベースシステムは，大量データが扱える論理プログラミングシステムとして，再帰的に定義された節などを用いることなどにより，柔軟な質問処理を行うことができる．Prolog などの論理プログラミングの手続きでは，計算の重複により処理効率が低下したり再帰質問において無限ループがおこり処理できないという問題があるが，ホーン節に限った問合せ処理に関しては，これらの問題を解決する効率化手法が数多く提案されてきた[2]．なかでも，マジックセット法[3]は，プログラムがトップダウン処理を模倣するように変形を行い，それをボトムアップ処理で実行するようなプログラム変形の手法であり，トップダウン処理とボトムアップ処理のそれぞれの問題を解決し，なおかつ両方の処理の長所が生かせるようになっている．

しかし近年，論理プログラミングの表現能力の拡張に関して，多くの研究が行われ，その拡張のひとつ

† 豊橋技術科学大学工学部情報工学系
Department of Information and Computer Sciences, Toyohashi University of Technology
☆ 現在，株式会社富士通コンピュータテクノロジ
Presently with Fujitsu Computer Technology Limited

の方向として**選言的論理プログラミング**（disjunctive logic programming）の導入が議論されるようになった[7]．選言的論理プログラムでは，節の頭部に複数のリテラルの選言（disjunction）を記述することができる．その結果確定的な情報のみならず，不確定な情報が非ホーン節として表現できる．このような高い知識表現能力を持つことから，アブダクションなどの高次推論[14]にも応用されおり，選言的論理プログラムを用いたデータベースに対する問合せ処理の効率化は重要な問題となってきた[15]．

選言的論理プログラムの意味論に関しては，Minkerによる極小モデル意味論（minimal model semantics）[9]が代表的なものとして知られている．この意味論に対する手続きはいくつか提案されているものの，ホーン節プログラムの場合と比較して問合せ処理が複雑になる．たとえば，SLI導出[7]は，線形導出に基づいたトップダウン型の証明手続きであるが，SLD導出ほど単純ではない．一方，ボトムアップ型の証明手続きとしては，選言的プログラムのモデルを計算する方法がある．この方法は，MGTP[4]やSATCHMO[8]などの定理証明器で簡単に計算できる利点があるものの，与えられたゴール（問合せ）に関係のない計算をしてしまうという欠点を持つ．また，従来のマジックセット法等の効率化手法も，選言を扱っていないためそのままでは使用できない．

一方，定理証明の分野で最近，**ノンホーン・マジックセット**（*non-Horn magic set*: **NHM**）**法**[5),12]が提案され，その成果を十分にあげており，様相論理定理証明にも応用されている[1]．この手法は，与えられたプログラムに負節を中心としたトップダウン処理を模倣するように変形を加え，そのプログラムをMGTPを用いてボトムアップ処理で実行させる．これにより，トップダウン処理を行うときのような複雑な制御を必要とせず，なおかつボトムアップ処理のような冗長な計算を避けるという利点がある．しかし，これは定理証明に限ってのことであり，データベースの問合せ処理に関しては従来考えられていなかった．

そこで，本研究では，上記で述べた定理証明に関するノンホーン・マジックセット法を実現し，それを問合せ処理に拡張することを目的とする．問合せ処理に関するノンホーン・マジックセット法では，定理証明のような負節を中心としたトップダウン処理の模倣に加えて，導出の中心に問合せ式が追加されるように変形を施す必要がある．また拡張に関しては，従来から提案されていた「深さ優先」と「幅優先」の2つの推論方式について行う．さらに，極小モデル意味論では，「確定解」と「可能解」という2つの概念があるため，変形プログラムでも各々の解を求めることができるようにする．最後に，いくつかの例題を用いて実験を行い，その実用性を検証する．

本論文の構成は次のとおりである．2章で定理証明におけるノンホーン・マジックセット法について説明し，3章でそれをデータベースの問合せ処理に拡張し，実験および評価を行う．4章で考察を述べ，5章でまとめと今後の課題を述べる．

2. 定理証明におけるNHM法

従来のマジックセット法はホーン節集合にのみ適用できる．ここでは，ホーン節のプログラムに対するマジックセット法を拡張した，定理証明用のノンホーン・マジックセット法を紹介する．

最初に節に関する基本的な定義を述べる．式

$$B_1,\ldots,B_n \rightarrow A_1;\ldots;A_m \quad (m \geq 0, n \geq 0)$$

を節といい，$A_1 \vee \ldots \vee A_m \vee \neg B_1 \vee \ldots \vee \neg B_n$ を意味し，以下のようにも表される．

$$A_1;\ldots;A_m \leftarrow B_1,\ldots,B_n$$

ここで $A_1,\ldots,A_m$ と $B_1,\ldots,B_n$ はいずれもアトムであり，$A_1;\ldots;A_m$ を節の**頭部**（または**後件**）といい，$B_1,\ldots,B_n$ を節の**本体**（または**前件**）という．**選言的論理プログラム**は上の形式の節の集合である．

上の形式の節において $m=0$ のとき，**負節**といい，頭部を *false* で表す．また $m=1$ のときは**確定節**である．$m \leq 1$ のときは**ホーン節**であり，確定節または負節である．**非ホーン節**とは $m \geq 2$ の節であり，頭部に複数のリテラルの選言を持つ節である．また $n=0$ の節を**正節**といい，本体を *true* で表す．正節でも負節でもない節を**混合節**と呼ぶ．

2.1 定理証明と関連性テスト

一階述語論理の定理証明器として，Prolog上で容易に実現できるSATCHMO[8]が提案されいる．これは**モデル生成**（model generation）に基づく定理証明器であり，与えられたプログラムのモデルを生成することを試みて充足可能性をテストする．**モデル生成動作**の基本は，モデル候補を設定し，充足されない節を検出し，そのモデル候補を修正していくことである．

しかしながら，あるモデル候補のもとで，充足されない節が複数存在する場合，節の評価順序によって計算コストがかなり異なる．特に，複数の充足されない節の中に非ホーン節が含まれている場合には，節の評価順序によってモデル候補の爆発が生じてしまう場合がある．たとえば以下のプログラム（TPTP[17]の問題SYN009-1）を考える．

$true \rightarrow s(c)$
$true \rightarrow s(b)$
$true \rightarrow s(a)$
$s(X), s(Y), s(Z)$
$\quad \rightarrow p(X,Y,Z); q(X,Y,Z); r(X,Y,Z)$
$p(c,X,Y) \rightarrow false$
$q(X,c,Y) \rightarrow false$
$r(X,Y,c) \rightarrow false$

これは評価順序に何ら制御がない場合，モデル候補を最大で26,873,856個生成する．$s(X)$, $s(Y)$, $s(Z)$ に c が照合されるとただちに $false$ が導かれる．しかし，$s(c)$ よりも $s(a)$, $s(b)$ が先に処理されると組み合わせ的爆発を起こしてしまう．このプログラム中には非ホーン節は1つしかないが，多くの定理証明問題においては非ホーン節を多数含むため，この種の問題が顕在化すると考えられる．この節の評価順序に関して，プログラム中の負節を第一に優先しなければならないのは明らかである．また，非ホーン節を適用する順序も，なるべく負節に依存した要素を追加するような非ホーン節を優先的に選ぶ方がよい．

この点に関して，SATCHMOに対する**関連性テスト**（Relevancy Testing）[11]が提案されている．プログラムの確定節の集合と現時点のあるモデル候補との和集合から，$false$ を導くことに失敗した場合，その原因となったサブゴールを関連リテラルと呼ぶ．関連性テストにおいては，充足されない非ホーン節が複数ある場合に，後件のすべてのアトムが関連リテラルと単一化可能であるような非ホーン節を優先する．

一方，マジックセット法やAlexander法が，演繹データベースの分野で提案されている．これらの方法は，確定節の集合を対象として，ゴールに関連する基礎アトムのみを外延データベース（EDB）として生成するように内包データベース（IDB）を変換する．これらは，重複したサブゴールの証明は行わないというボトムアップ型証明法の利点を保存して，ゴールに関連するサブゴールの証明のみを行うというトップダウン型証明法の長所を取り入れている．

ノンホーン・マジックセット（NHM）法[5),12)]は，ホーン節集合に適用できるマジックセット法を非ホーン節にも適用できるよう拡張したものである．この方法は，関連性テストと同様に，非ホーン節を含めたプログラム節に対して，負節と関連する充足されない節にモデル候補の拡張を限定でき，充足可能性テストの計算時間を削減することができる．特に，負節と関連しない非ホーン節の適用によるモデル候補の数の爆発を避けることができる．また，与えられるプログラムは静的に変換され，証明器としてはモデル生成型定理証明器のみを用いるため，関連性テストにともなうオーバヘッドもかなり削減することができる．

本研究においては，非ホーン節に関する処理系として**MGTP**（Model Generation Theorem Prover）[4]を用いた．次にMGTPについて紹介する．

2.2 MGTP

MGTPはモデル生成に基づくボトムアップ型の定理証明器である．MGTPの入力節は，含意形式

$$A_1, \ldots, A_n \rightarrow C_{1,1}, \ldots, C_{1,l_1}; \ldots; C_{m,1}, \ldots, C_{m,l_m}$$

で与えられる．ここで，$n, m \geq 0$, $l_i \geq 1$ $(i = 1, \ldots, m)$ であり，A_i $(i = 1, \ldots, n)$ および $C_{j,k}$ $(j = 1, \ldots, m;\ k = 1, \ldots, l_i)$ はアトムである．一般の節との形式上の違いは，後件にアトムの連言の選言が記述できることだけである．

MGTPは，与えられた節集合を充足するアトムの集合（モデル）を求めるために，以下の2つの操作をアトムの集合（**モデル候補**と呼ぶ）に繰り返し適用する．モデル候補の初期値は空集合である．

- モデル拡張操作：モデル候補 M に対して，混合節あるいは正節
  $$A_1, \ldots, A_n \rightarrow C_{1,1}, \ldots, C_{1,l_1}; \ldots; C_{m,1}, \ldots, C_{m,l_m}$$
  と置換 σ が存在して，アトム $A_1\sigma, \ldots, A_n\sigma$ がすべて M によって充足され，かつ，すべての $j = 1, \ldots, m$ に対し，M によって充足されないようなアトム $C_{j,k}\sigma$ $(1 \leq k \leq l_j)$ が存在するならば，各 $j = 1, \ldots, m$ に対し，$\{C_{j,1}\sigma, \ldots, C_{j,l_j}\sigma\}$ を M に付け加えた m 個のモデル候補を生成する．
- モデル棄却操作：モデル候補 M に対して，負節
  $$A_1, \ldots, A_n \rightarrow false$$
  と置換 σ が存在して，アトム $A_1\sigma, \ldots, A_n\sigma$ がすべて M によって充足されるならば，モデル候補 M を棄却する．

上記の規則が適用できなくなった時点で，節集合のすべての極小モデルを含むモデル集合が得られる．もしすべてのモデル候補が棄却されれば，入力節集合は充足不可能であることが分かる．

節に現れるすべての変数がその前件に現れるとき，その節は**値域限定**（range-restricted）されているという．値域限定された節集合に対して，生成されるモデル候補は，変数を含まないアトムのみから構成される．したがって，上述の規則を適用する際および節の充足性を判定する際に，完全な単一化は必要なく，照合だけで十分である．

しかし前述のとおり，モデル生成法においては，あるモデル候補で充足されない非ホーン節が複数存在する場合，生成されるモデル候補数の爆発が生じるという問題点がある．NHM 法はこうした無駄なモデル候補の生成を防ぐために考案された．

2.3 NHM 法の変形方式

以下に，NHM 法における変形方式を示す．これらの変形の正当性は文献 5)，12) を参照のこと．

与えられたプログラム P の任意の節

$C:\ A_1,\ldots,A_n \rightarrow B_1;\ldots;B_m$

に対して，次の 2 種類の NHM が存在する．いずれの変形も P のすべての節 C に対して行う．

2.3.1 深さ優先 NHM (Depth-First NHM)

C で $n=0$ ならば，次の単一の節に変換する．

$$magic(B_1),\ldots,magic(B_m) \rightarrow B_1;\ldots;B_m$$

さもなければ，次の $n+1$ 個の節に変換する．

$$magic(B_1),\ldots,magic(B_m) \rightarrow magic(A_1), supmagic_{r,1}(\mathbf{Vr})$$

$$supmagic_{r,k}(\mathbf{Vr}), A_1 \rightarrow magic(A_{k+1}), supmagic_{r,k+1}(\mathbf{Vr}) \quad (k=1,\ldots,n-1)$$

$$supmagic_{r,n}(\mathbf{Vr}), A_n \rightarrow B_1;\ldots;B_m$$

ここで，r は元の節の識別子，$\mathbf{Vr}$ は節 r に含まれるすべての変数の組，$magic$ と $supmagic_{r,i}$ $(r \in P, 1 \leq i \leq n)$ は新しく導入された述語記号である．また，$m=0$ のとき，$magic(false)=true$ とする．

2.3.2 幅優先 NHM (Breadth-First NHM)

まず，$n>0$ であるすべての節に対して，以下の節を生成する．

$$magic(B_1),\ldots,magic(B_m) \rightarrow magic(A_1),\ldots,magic(A_n)$$

次に，すべての節に対して，以下の節を生成する．

$$magic(B_1),\ldots,magic(B_m), A_1,\ldots,A_n \rightarrow B_1;\ldots;B_m$$

ここで，「幅優先」という名称は，サブゴールを並列に展開するところに由来する．これに対して，「深さ優先」の方は，サブゴールを左から右へ順に展開していくものとして名付けられている[12)]☆．

2.4 修飾子の付加

上の NHM 変形によって得られる節集合は，一般にそのままでは値域限定性を満たさない．この問題への対処法として，述語の引数の束縛情報に基づいた修飾子 (adornment) 付き述語の導入が提案されている．ただし n 個の引数を持つ述語の修飾子は 2^n 通りあるため，すべての修飾子について NHM 変形節を用意するのは現実的でない．修飾子が前もって過不足なく分かれば，NHM 変形節の数を抑制することができ，実行効率を上げることができる．証明に寄与する修飾子を静的に解析するには，ゴールに現れる定数（束縛情報）が変形後の節にどのように伝搬していくかを解析する必要がある．これを変数のモード解析という．

解析には Koshimura, et al.[6)] による方法が提案されているが，これは従来の確定節集合のためのマジックセット法における束縛パターンの計算アルゴリズム[10),18)]とは大幅に異なるものである．そこで本研究では，ホーン節（関数なし）のための装飾パターンの計算アルゴリズムを非ホーン節（関数あり）用に拡張して用いた．以下にその手順を文献 10) の書式に従って示す．

束縛パターン（修飾子）を計算するアルゴリズム

入力：プログラム P と負節の集合 F $(F \subseteq P)$

出力：束縛パターンを付加したプログラム P'

ステップ 1：F の束縛パターンを求める．つまり F の本体のアトムの各引数が定数なら b，変数なら f となる束縛パターン α☆☆ を作る．F の述語記号の集合を FP とするとき，束縛パターン α を付加した述語記号の集合を FP^{α} とする．

ステップ 2：本ステップで処理しようとする束縛パターンを付加した述語記号を p^{β} とする（初めはステップ 1 で作られた FP^{α} の各要素が対象となる）．p を頭部の述語記号として持つ P の各節の複製を作る．複製の頭部の p を p^{β} に置き換える．複製の本体の各アトム（述語記号を p_i とする）に対して，左のアトムから順に以下の要領で束縛パターン γ を求め，p_i を p_i^{γ} で置き換え，束縛パターンを付加した後に複製を P' に追加する．

- 引数が定数のときにはその引数には b を印付ける．
- 引数が変数のとき，その変数が頭部に現れ β により b が印付けられているか，その変数が p_i の左側（一度本体の左に現れた変数は，装飾が f だったものでも束縛変数として記憶される）の本体中のアトムに現れている場合，その引数に b を印付ける．引数が変数でこれら 2 条件のいずれも満たさない場合，f を印付ける．
- 引数が関数を含むとき，関数中の引数のすべてが

☆ これらの用語の使い方は，SLD 導出などのトップダウン証明における探索戦略で使われる幅優先探索（横型探索）や深さ優先探索（縦型探索）とは異なるものである．

☆☆ 引数が関数を含む場合，関数の中に 1 つでも変数があれば f で，それ以外なら b とする．本体にリテラルが複数あるときは，ステップ 2 の処理に準ずる．

定数のときは b，また，変数がある場合，そのすべてが頭部に現れ β により b が印付けられているか，p_i の左側の本体中のアトムに現れていれば b を印付け，それ以外のときは f を印付ける．

$p_i^?$ がまだ処理していない新しい束縛パターンをもつ述語記号ならば本ステップの最初から処理する．

ステップ3：束縛パターンを付加した述語記号がすべてステップ2で処理されれば終了．束縛パターンは各述語記号に対して有限なので，本手続きは必ず停止する．すなわちアルゴリズムである．

2.5 修飾子付き NHM 変形

修飾子を導入した NHM 変形を示す．変形の際には，前節で述べた方法によってすでに必要な修飾子が分かっているものとする．

修飾子付き NHM では，与えられた節集合 P の各節

$$A_1,\ldots,A_n \rightarrow B_1;\ldots;B_m$$

を次のように変換する．以下では，得られている B_i の修飾子付き述語の集合を $Adorn_{B_i}$ で表すことにする．

2.5.1 修飾子付き深さ優先 NHM

まず，各 i $(1 \leq i \leq m)$ の装飾子付き述語 B_i^* $(\in Adorn_{B_i})$ を1つずつ取り出し，次のような $\prod_{i=1}^{m} |Adorn_{B_i}|$ 個の混合節を生成する☆．

$$magic(B_1^*),\ldots,magic(B_m^*) \rightarrow \begin{cases} magic(A_1^*), supmagic_{r,1}(V_{r,1}) & \text{if n} > 0 \\ B_1;\ldots;B_m & \text{if n} = 0 \end{cases}$$

ここで，$V_{r,1}$ は $B_1^*,\ldots,B_m^*$ に現れる変数の列であり，A_1^* は A_1 と $V_{r,1}$ から得られる修飾子付きアトムである．

次に各 j $(1 \leq j \leq n)$ について，以下のような節を1つ生成する．

$$supmagic_{r,j}(V_{r,j}), A_j \rightarrow \begin{cases} magic(A_{j+1}^*), supmagic_{r,j+1}(V_{r,j+1}) & \text{if } j < n \\ B_1;\ldots;B_m & \text{if } j = n \end{cases}$$

ここで，$V_{r,j+1}$ は $V_{r,j}$ と A_j に現れる変数の列であり，A_j^* は A_j と $V_{r,j}$ から得られる修飾子付きアトムである．

2.5.2 修飾子付き幅優先 NHM

まず，$n > 0$ であるすべての節に対して，以下の節を生成する．

$$magic(B_1^*),\ldots,magic(B_m^*) \rightarrow magic(A_1^*),\ldots,magic(A_n^*)$$

次に，すべての節に関して，以下の節を生成する．

$$magic(B_1^*),\ldots,magic(B_m^*), A_1,\ldots,A_n \rightarrow B_1;\ldots;B_m$$

2.6 値域限定性

前述のとおり MGTP の使用においては，入力節に対する値域限定の制約がある．しかし元の節集合が値域限定性を満たしていなければ，たとえ修飾子付き NHM 変形を用いても，変形後の値域限定性は保証されない．そのような場合，以下の手順により，変数の領域を表す述語 dom[8] をプログラムに付加する．

dom の計算アルゴリズム

- $A \rightarrow B, p(X)$ のように，前件に含まれない変数 X が後件に存在するような節があるとき，前件に $dom(X)$ を加え，$A, dom(X) \rightarrow B, p(X)$ とする．
- 節に現れるすべての定数 c について，$true \rightarrow dom(c)$ を加える．もしプログラム中に定数が出現しない場合，任意定数を a として $true \rightarrow dom(a)$ を加える．
- 節中に現れるすべての n 項関数 f について，$dom(X_1),\ldots,dom(X_n) \rightarrow dom(f(X,\ldots,X_n))$ を加える．

すなわち，前件に現れない後件部の各変数に対して，代入可能な基礎項（エルブラン領域）を節集合中から取り出して dom で表す．

3. 問合せ処理における NHM 法

本章では，定理証明用のノンホーン・マジックセット法を問合せ処理用に拡張する．

3.1 極小モデル意味論

選言的論理プログラム P の意味は，**極小モデル意味論**[9] が代表的である．以下，$\mathcal{MM}_P$ を P の極小モデルの集合とする．ここで，問合せ Q は，

$$query:\ Q(\mathbf{X})$$

により与えられるとする．この式を**質問節**といい，$\mathbf{X}$ は変数の組である．極小モデル意味論では，基礎式 Q が与えられたとき，Q に対する解を次のように，“**true**”（Q は**確定解**），“**false**”（$\neg Q$ は確定解），もしくは“**unknown**”（Q は**可能解**）で定義する．

Q が **true** $\leftrightarrow \forall M_i \in \mathcal{MM}_P\ (M_i \models Q)$

Q が **false** $\leftrightarrow \forall M_i \in \mathcal{MM}_P\ (M_i \not\models Q)$

Q が **unknown** $\leftrightarrow$ **true** でも **false** でもない

3.2 NHM 変形の拡張 (1)

ここでは，前章で紹介したノンホーン・マジックセット法を，データベースの問合せ用に拡張する．定理証明に関するノンホーン・マジックセット法は，プログ

☆ $magic(B^*)$ は，B の束縛引数だけを残したものである．
例：$magic(ex\_fbf(X,Y,Z)) = magic\_ex\_fbf(Y)$

ラムに変形を施すことにより負節を中心としたトップダウン処理を模倣する．これを問合せ処理に拡張するには，導出の中心に問合せ式が追加されるように変形しなければならない．

このことから，問合せ式に対応する処理は，問合せ式に対する処理以外では定理証明のそれとおおむね一致する．ここで2章で負節に対して行った処理を，問合せ *query*: $Q(\mathbf{X})$ （$\mathbf{X}$ は変数の組）に対しても行い，それと並行して負節に対する処理も行う．さらに，問合せに対する解代入（$\mathbf{X}$ の値）を求めなければならないため，MGTP が求める極小モデルの中から $Q(\mathbf{X})$ の例だけを集め，確定解の場合は，それがすべての極小モデルに含まれているかどうかをテストする必要がある．

拡張アルゴリズムは以下のようになる．

- *query*: $Q(\mathbf{X})$ に対して，$true \rightarrow magic(Q(\mathbf{X}))$ とする．
- 2.4 節の束縛パターンを計算するアルゴリズムにおいて，負節に関する束縛パターンと並行して，質問節に対しても，アルゴリズムの入力である負節を質問節に置き換えて同様に行う．
- 修飾子付き NHM 変形を次のように変更する．
  query: $Q(\mathbf{X})$ に対して，$true \rightarrow magic(Q(\mathbf{X})^*)$ とする．
- 上記以外の NHM 変形については，2.5 節と同様である．
- MGTP において解代入を求める．ここで確定解と可能解の切換えが可能なようにする（後述）．

上のように拡張したノンホーン・マジックセット法を用いて，同世代問題に関する以下の実験を行った．ここで同世代問題の **IDB** は以下で与えられるとする．

$$parent(A,B), sgc(A,D), parent(D,C) \rightarrow sgc(B,C)$$
$$parent(A,B), parent(A,C) \rightarrow brother(B,C); sister(B,C)$$
$$parent(A,B) \rightarrow father(A,B); mother(A,B)$$
$$brother(A,B) \rightarrow sgc(A,B)$$
$$sister(A,B) \rightarrow sgc(A,B)$$
$$parent(A,B) \rightarrow ancestor(A,B)$$
$$parent(A,B), ancestor(B,C) \rightarrow ancestor(A,C)$$

またこの問題における **EDB** は，**図 1** で与えられる親子関係を表しているとする．

$$true \rightarrow parent(1,2)$$
$$true \rightarrow parent(1,3)$$
$$true \rightarrow parent(2,4)$$

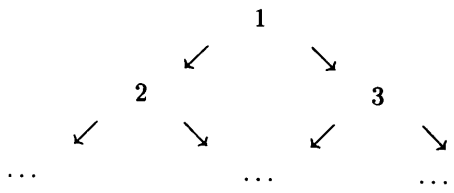

図 1 sgc 問題
Fig. 1 Same-Generation Cousins' problem.

表 1 $sgc(X,4)$ における問合せの実験結果
Table 1 Results on the query $sgc(X,4)$.

世代数	手法	#M	#A	time
5	naive	4096	32	4005.45
	d-f-nhm	16	62	0.54
	b-f-nhm	256	61	23.67
7	naive	–	–	T.O.
	d-f-nhm	64	94	4.30
	b-f-nhm	–	–	T.O.
9	naive	–	–	T.O.
	d-f-nhm	128	134	44.72
	b-f-nhm	–	–	T.O.

$$true \rightarrow parent(2,5)$$
$$\vdots$$

問合せ — 4 と同世代の人をすべて求める．

query : $sgc(X,4)$

このプログラムをそのまま実行すると，すべての *sgc*, *brother*, *sister*, *father*, *mother*, *ancestor*, *parent* を求めてしまい，実行時間が膨大になる．NHM 変形をかけたプログラムでは，トップダウン模擬により問合せ *sgc* に関係する *brother*, *sister*, *parent* のみを求めるように探索を行うため，実行時間を大幅に削減できる．

この実験結果を**表 1** に示す．表中で，naive は MGTP をそのまま用いた場合，d-f-nhm は深さ優先 NHM を MGTP で計算した場合，b-f-nhm は幅優先で NHM を MGTP で計算した場合である．また，#M はモデル候補数，#A は平均アトム数，time はモデル候補を求める時間（単位 sec），T.O.（Time Out）は 2 時間以上とし，以下これらの記号を用いる．測定はすべて SPARCstation20/71 上で行った．

表 1 から，マジックセット変形の問合せ処理に関する有効性が確認でき，*query*: $sgc(X,4)$ に関しては，変形プログラムは十分に効果をあげている．また，問合せを *query*: $sgc(4,X)$ に変えた実験結果を**表 2** に示す．これについても変形プログラムの方が処理が格段に速く，良い結果が出ている．NHM プログラムは，トップダウン処理を模倣するため，トップダウン処理

表2 $sgc(4,X)$ における問合せの実験結果
Table 2 Results on the query $sgc(4,X)$.

世代数	手法	#M	#A	time
5	naive	4096	32	4005.45
	d-f-nhm	16	47	0.33
	b-f-nhm	256	52	18.12
7	naive	–	–	T.O.
	d-f-nhm	16	53	0.34
	b-f-nhm	4096	84	4814.25
9	naive	–	–	T.O.
	d-f-nhm	16	53	0.36
	b-f-nhm	–	–	T.O.

表3 $sgc(4,X)$, $sgc(X,Y)$ における問合せの実験結果
Table 3 Results on the query $sgc(4,X)$, $sgc(X,Y)$.

世代数	手法	#M	#A	time
4	naive	256	21	11.3
	d-f-nhm	32	61	1.42
	b-f-nhm	32	38	0.77
5	naive	4096	32	4005.45
	d-f-nhm	256	84	107.77
	b-f-nhm	256	54	46.14
6	naive	–	–	T.O.
	d-f-nhm	512	109	837.16
	b-f-nhm	512	72	336.77

が得意とするような問題 ($sgc(4,X)$) では，非常に良い結果を出すことができる．トップダウン処理が苦手とするような問題 ($sgc(X,4)$) についても良い結果を出しているが，やや性能が落ちる．

さらに，表1と表2において，深さ優先の NHM の方が良い結果を示していが，これは *dom* の付加が関係している．この実験では，深さ優先 NHM は *dom* が加えられておらず，幅優先には加えられている．そこで問合せを，*query*: $sgc(4,X), sgc(X,Y)$ にして，深さ優先でも *dom* が加えられるようにして処理を行った結果を**表3**に示す．この表では深さ優先と幅優先の結果が逆転している．この問題に関する考察は 4 章で行う．

3.3 NHM 変形の拡張 (2)—可能解の計算

ここで，文献 16) による以下の例題 (**Shima's problem** と呼ぶ) を考える．

$rain, fine \rightarrow false$
$true \rightarrow has(a, gym)$
$true \rightarrow has(b, gym)$
$true \rightarrow has(c, gym)$
$true \rightarrow has(d, pool)$
$true \rightarrow has(e, pool)$
$swims(A) \rightarrow wet(A)$
$has(A, pool) \rightarrow swims(A); rain$

表4 Shima's problem の処理
Table 4 Results on Shima's problem.

query	手法	#M	#A	time	可能解
$swims(X)$	naive	24	22	0.39	d,e
	d-f-nhm	2	19	0.03	d,e
	b-f-nhm	2	16	0.01	d,e
$trains(X)$	naive	24	22	0.39	a,b,c
	d-f-nhm	1	11	0.00	–
	b-f-nhm	1	11	0.00	–

$rain \rightarrow wet(A)$
$true \rightarrow fine; cloudy; rain$
$has(A, gym) \rightarrow trains(A); plays(A, pp)$
$has(A, pool) \rightarrow swims(A); rain$

この問題に関して，いくつかの問合せについて解いた結果を**表4**に示す．表4では，$trains(X)$ の可能解を NHM では求めていないが，これは変形した節に原因がある．プログラム中で $trains(A)$ は，1つの節の後件に1回しか出現しないので，この節での分岐が起こらないと解は出ない．ところが，定理証明用に考案された NHM は，非ホーン節に関して，頭部の全部の述語の導出が揃ったら分岐を行うように変形を行っている．これは，問題の充足不可能性を判定するには，分岐により生じたすべての枝を棄却しなければならず，その選言のリテラルはすべて解かれなければならないという特性を表した結果である．

問合せ処理では，前述の拡張だけでも，確定解の判定 (問合せがすべての極小モデルで真かどうかの判定) には使用できるが，可能解の判定には使えない．そこで，非ホーン節の頭部の部分に不確定な要素がある場合は，分岐の要素の片方だけでも導出が起こったら分岐が行われるようにする必要がある．いま Shima's problem について深さ優先 NHM の場合を例にして説明する．これまで述べた NHM を用いると，*query*: $trains(X)$ の場合，変形後に以下のような式ができる．

$magic\_plays, magic\_trains\_f$
$\rightarrow magic\_has\_fb(gym), supmagic\_12\_1$

このときプログラム中で，$plays(A, pp)$ が不確定な分岐要素なので，上式を以下のように変更する．

$magic\_plays$
$\rightarrow magic\_has\_fb(gym), supmagic\_12\_1$
$magic\_trains\_f$
$\rightarrow magic\_has\_fb(gym), supmagic\_12\_1$

すなわち，まとめられていた条件部を分離し，$trains(X)$ に関する分岐を生じさせる．

以上のように，変形した節中に不確定な分岐要素があった場合，条件を分離することにより分岐を生じさせ，片方の枝のモデルを調べる．これにより，問合せ

の可能解を求めることができる．

次に，この拡張アルゴリズムを深さ優先と幅優先のそれぞれについて示す．以下いずれも，節

$C: A_1,\ldots,A_n \rightarrow B_1;\ldots;B_m$

を変換するものとする．

3.3.1 修飾子付き深さ優先 NHM の拡張

まず，節 C を次式で変換する．

$$magic(B_1^*),\ldots,magic(B_m^*) \rightarrow \begin{cases} magic(A_1^*), supmagic_{r,1}(V_{r,1}) & \text{if n} > 0 \\ B_1;\ldots;B_m & \text{if n} = 0 \end{cases}$$

ここで，B_i $(1 \leq i \leq m)$ が不確定な要素だったときは

$$magic(B_1^*),\ldots,magic(B_{i-1}^*), magic(B_{i+1}^*),\ldots,magic(B_m^*) \rightarrow \begin{cases} magic(A_1^*), supmagic_{r,1}(V_{r,1}) & \text{if n} > 0 \\ B_1;\ldots;B_m & \text{if n} = 0 \end{cases}$$

$$magic(B_i^*) \rightarrow \begin{cases} magic(A_1^*), supmagic_{r,1}(V_{r,1}) & \text{if n} > 0 \\ B_1;\ldots;B_m & \text{if n} = 0 \end{cases}$$

という変換を代わりに用いる．次に，各 j $(1 \leq j \leq n)$ について，$supmagic_{r,j}(V_{r,j})$ を前件に含む式を 2.5.1 項と同様の変換により生成する．

3.3.2 修飾子付き幅優先 NHM の拡張

まず，$n > 0$ であるすべての節 C に対して，以下の式を生成する．

$$magic(B_1^*),\ldots,magic(B_m^*) \rightarrow magic(A_1^*),\ldots,magic(A_n^*)$$

ただし，ここで B_i $(1 \leq i \leq m)$ が不確定な要素であれば，上式の代わりに次の式を生成する．

$$magic(B_1^*),\ldots,magic(B_{i-1}^*), magic(B_{i+1}^*),\ldots,magic(B_m^*) \rightarrow magic(A_1^*),\ldots,magic(A_n^*)$$

$$magic(B_i^*) \rightarrow magic(A_1^*),\ldots,magic(A_n^*)$$

次に，すべての節 C に関して，以下の式を生成する．

$$magic(B_1^*),\ldots,magic(B_m^*), A_1,\ldots,A_n \rightarrow B_1;\ldots;B_m$$

ただし，ここで B_i $(1 \leq i \leq m)$ が不確定な要素であれば，上式の代わりに次の式を生成する．

$$magic(B_1^*),\ldots,magic(B_{i-1}^*), magic(B_{i+1}^*),\ldots,magic(B_m^*) \rightarrow B_1;\ldots;B_m$$

$$magic(B_i^*) \rightarrow B_1;\ldots;B_m$$

3.4 可能解を求める拡張に関する評価

3.3.1 項と 3.3.2 項の処理を施したプログラムの実験結果を**表 5** に示す．表が示すように，不確定要素の導出に関する条件を分離することによって，分岐を起こし，問合せの解を正しく求めることができる．

表 5 Shima's problem の処理 (2)
Table 5 Results on Shima's problem (2).

query	手法	#M	#A	time	可能解
$trains(X)$	naive	24	22	0.39	a,b,c
	d-f-nhm	8	19	0.05	a,b,c
	b-f-nhm	8	18	0.04	a,b,c

表 6 Shima's problem の処理 (3)
Table 6 Results on Shima's problem (3).

query	Answer	runtime[sec] 文献 16)	d-f-nhm	b-f-nhm
$wet(X)$	true	0.02	0.03	0.01
$swims(X)$	unknown	0.02	0.03	0.01
$swims(a)$	false	0.03	0.03	0.01

なお，この制御を行うと，分岐が増えるためモデルの数が多くなってしまう．しかしながら，表 5 では，単純にボトムアップ処理をした場合に 24 のモデル候補を生成したのに対して，拡張した NHM 変形を施したものはモデル候補を 8 個しか求めておらず，その処理時間はやはり非常に高速である．

ちなみに，文献 16) における問合せの結果は，**表 6** のようになっている．文献 16) では，SATCHMORE[11] における関連性テストの手法を取り入れている．この問合せ処理に関しては，多少条件が異なるものの，ほぼ同等の結果が出ており，このことからも NHM 変形の効果が分かる．

3.5 1000 の問題に関する問合せ処理

評価の最後に，定理証明 1000 の問題[17]について，負節の一部を問合せに変えて，問合せ処理を行い，この拡張した NHM の特性を調べた．

ここで，各々の問題の負節の一部を問合せに変えるために，ある負節

$$A_1,\ldots,A_n \rightarrow false$$

を

$$A_1,\ldots,A_n \rightarrow answer(\mathbf{X})$$

なる節に変換し，$answer$ を問合せ述語として処理を行った．ここで $\mathbf{X}$ は負節中に出現する変数の組である．

表 7 に結果を示す．前章でも述べたが，定理証明問題は関数を含んだものも多く，変形を施すことにより処理が遅くなってしまう場合もある．しかしながら，一般に演繹データベースにおける問合せ処理においては，関数は出現しないという仮定を置くことが多いので，拡張した NHM は問合せ処理に関して非常に有効であると考えられる．

4. 問合せ処理に関する考察

前章における実験の比較結果を**表 8** にまとめる．こ

表7 1000の問題に対する問合せ処理
Table 7 Results on TPTP.

問題	手法	#M	#A	time
SYN009-1	naive	–	–	T.O.
	d-f-nhm	1	12	0.01
	b-f-nhm	1	7	0.00
SYN101-1.002:002	naive	1	32	0.03
	d-f-nhm	1	252	0.89
	b-f-nhm	1	110	0.35
SYN102-1.007:007	naive	–	–	T.O.
	d-f-nhm	1	1087	15.17
	b-f-nhm	1	295	7.62
PUZ012-1	naive	729	50	204.40
	d-f-nhm	27	65	1.21
	b-f-nhm	729	70	232.10

表8 NHMの効率化
Table 8 Efficiency of NHM.

手法	モデル候補数の削減	
	Shima's problem	sgc
naive	1	1
d-f-nhm	$\frac{1}{12}\sim\frac{1}{3}$	$\frac{1}{256}$
b-f-nhm	$\frac{1}{12}\sim\frac{1}{3}$	$\frac{1}{16}$

表9 深さ優先NHMと幅優先NHMの比較
Table 9 Comparison between depth-first and breadth-first NHMs.

手法	節数増加	導出	負節評価	値域限定
深さ優先	大	少	遅	崩れにくい
幅優先	小	多	速	崩れやすい

れより，そのまま処理した場合よりもNHM変形を加えた方が，一般にモデル候補数・処理時間ともにかなり削減できることが分かる．ただし，NHMはトップダウン処理を模倣するため，トップダウン処理が苦手とする問合せに関しては性能が少し落ちる．

次に，深さ優先と幅優先のNHMの各々の特徴を表9に示す．深さ優先NHMでは，1つの節につき複数個からなる変形節を作る．よって，節の本体が大きいと変形節の数が非常に多くなり，処理を低下させる原因となる．しかし，左から順に1つずつリテラルのマッチングを行うため，導出される事実を絞り込みかなり減らすことができる．ただし負節については，段階的にマッチングを行うので適用が遅れる．さらに，本体に現れるすべての引数を述語 *supmagic* の引数に用いる形で変形を進めるため，基本的には元のプログラムでの値域限定性が満たされていれば，変形プログラムの値域限定性も崩れない．

これに対して，幅優先NHMでは，1つの節につきたかだか2個からなる変形節しか作らないため，変形節数の増加は抑えられる．しかし，本体のリテラルのマッチングをまとめてとるので，比較的多くの事実が導出され，深さ優先NHMよりも効率が悪いことが多い．ただし，本体のリテラルが2個以下の節や命題節では，*supmagic* リテラルのマッチングが不要な分だけ，導出される事実の数が少なくて済む場合があり，深さ優先NHMより速く解けることもある．さらに，負節に関しては速い段階で適用するができ，現在処理しているモデル候補を棄却し，すぐに次のモデル候補を探索することができる．値域限定性については，変形による制約節を作るときに崩れる場合が多い．値域限定性が崩れると，節の本体に *dom* を加えなければならなくなり，これが問題となる場合がある．

以上のように，深さ優先NHMと幅優先NHMは各々特徴を持っているので，節（負節，本体のアトム数）の状態や問合せの種類，問題の形式により性能が変わってくる．これらを適切に使い分けたり，両方の長所を組み合わせて新しい変形方式を考えることが今後有望であると考えられる．

5. まとめと今後の課題

本研究ではデータベース問合せ用のノンホーン・マジックセット（NHM）法を実現し，実験を通して評価を行った．NHMの実現に際しては，拡張したモード解析手法を提案し，効果をあげることができた．

変形プログラムの特徴としては，制約が加られており，マッチングによる導出は少なくなり効率が良くなるという長所がある．ただし，NHMの動きがトップダウン処理を模倣するため1つの推論が深くなり，処理が遅くなる場合もある，という短所もある．しかしながら，関数の存在を仮定しないことが多いデータベースの問合せにおいては，推論が深くなる問題は顕著ではなく，拡張したNHMについては非常に良い結果を示すことができた．

最後に今後の課題として，以下の点があげられる．

- 失敗による否定（negation as failure）を節の本体に加えることにより拡張したプログラムに対して，NHMが適用できるか検討すること．これは，否定を含む層状あるいは一般論理プログラムにおけるマジックセット法（たとえば文献10)）を，頭部に選言を含むものに拡張しなければならず，意味論の選定を含めかなり難しい問題を含んでいる．
- 極小モデル意味論とは別の意味論（たとえば可能世界意味論[14]）に対する問合せ手続きをNHMを基礎に検討すること．
- 問合せには関係しない負節が存在するときに，負節のすべてを評価しない方法（たとえば文献13)）

を適用すること.

謝辞 本研究に関して貴重な意見をいただいた九州大学の長谷川隆三教授と越村三幸助手に感謝いたします.

参考文献

1) 赤埴淳一，井上克巳，長谷川隆三：様相節変換に基づくボトムアップ型様相論理証明法，情報処理学会論文誌，Vol.36, No.4, pp.822–831 (1995).
2) Bancilhon, F. and Ramakrishnan, R.: An Amateurs's Introduction to Recursive Query Processing Strategies, *Readings in Artificial Intelligence & Databases*, pp.376–430, Morgan Kaufmann (1989).
3) Beeri, C. and Ramakrishnan, R.: On the Power of Magic, *J. Logic Programming*, Vol.10, pp.255–299 (1991).
4) Fujita, H. and Hasegawa, R.: Model Generation Theorem Prover in KL1 Using a Ramified-Stack Algorithm, *Proc. ICLP '91*, pp.535–548, MIT Press (1991).
5) 長谷川隆三，井上克巳，太田好彦，越村三幸：上昇型定理証明の探索効率を高めるノンホーン・マジックセット，情報処理学会論文誌，投稿中 (1996).
6) Koshimura, M., Yamaga, H. and Hasegawa, R.: Mode Analysis for Generating Non-Horn Magic Sets, *Proc. Workshop on Finite Domain Theorem Proving*, pp.44–51, ICOT (1994).
7) Lobo, J., Minker, J., and Rajasekar, A.: *Foundations of Disjunctive Logic Programming*, MIT Press (1992).
8) Manthey, R. and Bry, F.: SATCHMO: A Theorem Prover Implemented in Prolog, *Proc. 9th International Conference on Automated Deduction*, LNCS, Vol.310, pp.415–434, Springer (1988).
9) Minker, J.: On Indefinite Data Bases and the Closed World Assumption, *Proc. 6th Internatinal Conference on Automated Deduction*, LNCS, Vol.138, pp.292–308, Springer (1982).
10) 森下真一：知識と推論，共立出版 (1994).
11) Loveland, D.W., Reed, D.W. and Wilson, D.S.: SATCHMORE: SATCHMO with RElevancy, *J. Automated Reasoning*, Vol.14, pp.325–351 (1995).
12) 太田好彦，井上克巳，長谷川隆三，中島 誠：ノンホーン・マジックセット，ICOT Technical Memorandum TM-1227, ICOT (1992).
13) Ohta, Y. and Inoue, K.: Incorporating Top-Down Information into Bottom-up Hypothetical Reasoning, *New Generation Computing*, Vol.11, No.3, 4, pp.401–421 (1993).
14) Sakama, C. and Inoue, K.: On the Equivalence between Disjunctive and Abductive Logic Programs, *Proc. ICLP '94*, pp.489–503, MIT Press (1994).
15) 世木博久：選言的論理プログラミングについて，人工知能学会基礎論研究会資料 SIG-FAI-9501-7, pp.37–38 (1995).
16) Shimajiri, Y., Seki, H. and Itoh, H.: Goal-Directed Query Processing in Disjunctive Logic Databases, *Proc. PLILP '95*, LNCS, Vol.982, Springer (1995).
17) Suttner, C.B. and Sutcliff, G.: The TPTP Problem Library (version 1.2.0), Technical report 95/6, Department of Computer Science, James Cook University, Australia (1995).
18) Ullman, J.D.: *Principles of Database and Knowledge-Base Systems, Vol.II: The New Technologies*, Computer Science Press (1989).

(平成8年3月19日受付)
(平成8年9月12日採録)

荒山 正志

1971年生. 1994年豊橋技術科学大学工学部情報工学課程卒業. 1996年同大学院工学研究科修士課程情報工学専攻修了. 同年（株）富士通コンピュータテクノロジ入社，同社豊橋開発センター勤務. 電子CADの開発研究に従事.

井上 克巳（正会員）

1959年生. 1982年京都大学工学部数理工学科卒業. 1984年同大学院工学研究科数理工学専攻修士課程修了. 同年松下電器産業（株）入社，同社技術本部勤務. 1986年（財）新世代コンピュータ技術開発機構に出向. 1993年豊橋技術科学大学情報工学系講師. 1994年同助教授. 工学博士（京都大学）. 人工知能，論理プログラミング，計算機科学に関する教育研究に従事. 人工知能学会，AAAI各会員.