ICOT Technical Report: TR-256

TR-256

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

宮崎収兄，羽生田博美
(沖電気)
伊藤英則

May, 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

Horn Clause Transformation: An Application of Partial Evaluation to Deductive Databases

Nobuyoshi Miyazaki*, Hiromi Haniuda*
and Hidenori Itoh**

* Oki Electric Industry Co., Ltd.
** Institute for New Generation Computer Technology

Abstract

In deductive database systems, the performance of recursive query processing usually depends on the complexity of queries. Therefore, their performance can be improved by reducing the complexity of queries before actual processing.

This paper discusses the Horn clause transformation (HCT) that simplifies queries by eliminating some of the predicates in them. HCT is introduced as an application of partial evaluation of logic programs. Partial evaluation is an algorithm that skips the processing of some predicates. The result of partial evaluation is also expressed by Horn clauses. Three types of HCT are defined as ways to specify which predicates are skipped. HCT can be used to simplify queries that include "not" predicates.

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

宮崎収兄　羽生田博美
（沖電気工業株式会社）

伊藤英則
（ＩＣＯＴ）

梗概

理そて、
処はっ処稿プ評を算付部に一に
ので従ら本型分価計件。とホ化
せ法。か。理部評分条るこの略
合方るてる論。の部がきるら簡
いのすしきを語の果でせれの
問く存化で換す述ム結現さこせ
の多依略が変入のラ終表化。合
型、に簡と節導部グ最で変るい
婦が数しこンき一ロは節をき問
再る語除るーづにプ果ンたでむ
てい述削図ホ基時、結ーか義含
いてのをうに行りのホえ定を
おれ中語化行念実あ価で与を語
にさせ述率を概のので評のの換述
ス案合の効化のムム分す語変
ー提い部り略価ラズ部示述節
ベか問一よ簡評グリ。をるン
タつがらにな分ロゴるとすーｎｏｔ”
ーく能かとう部プルあこ留ホ”
デい性せこよの型アでつ保のは
綜が理合うのム理る種立で種換で
式処い行こラ論す一り価各変用
方の問をはグは留の成評り節適

1　まえがき

演繹データベースの問い合せ処理方式は再帰述語の処理を中心に研究されている。問い合せ処理方式は大きくトップダウン方式とボトムアップ方式に分けられるが大部分の方式では問い合せ中の述語数に性能が依存する[1)]。従って問い合せ中の中間述語を除去し問い合せを簡略化することにより処理性能を向上することができる。

Henschenらの提案した問い合せ処理方式[2)]では基本アルゴリズムの中に中間述語の除去機能が組み込まれているが、適用範囲が簡単な問い合せに限定されている。また問い合せを関係代数に変換してから簡略化する方法も[3)]提案されているが、関係代数に基づいた方法にしか適用できないという限界がある。我々はホーン節問い合せの簡略化を行う方式としてホーン節変換を提案した[4)]。ホーン節変換の出力は入力と同じホーン節集合なので各種の問い合せ処理方式の前処理として適用できる。このアルゴリズムの一部については Prologによる演繹データベース実験システム[5)]と Guarded Horn Clauses（ＧＨＣ）によるシステム[6)]の２種類のインプリメントを行った。本稿ではこのホーン節変換を部分評価によるホーン節集合の変換として一般化し、３つのタイプのホーン節変換とそれらの性質について述べる。

2　基本概念

演繹データベース（ＤＤＢ）はファンクションのないホーン確定節の集合であり、内包データベース（ＩＤＢ：ルール部分）と外延データベース（ＥＤＢ：ファクト部分）から構成される。

ＤＤＢ≡　function free Horn clause set
　　＝　ＩＤＢ∪ＥＤＢ

ＤＤＢに対する問い合せは１つのゴール節を含むホーン節集合である。ゴール節は１つの述語からなると考えても一般性を失わない（本稿では述語という用語は引数をも含む意味で使用する）。問い合せにはファクトを含むこともできるが本稿ではルールのみと仮定

する。問い合せの解は問い合せ中のルールとＩＤＢ中のルールを合せたルール集合を用いて得られるので、本稿ではこの２つのルール集合を区別しない。即ち問い合せはゴール節のみからなり他の節はすべてＩＤＢにあると考える。また逆に問い合せという用語でＧＵＩＤＢを意味することがある。ゴールＧとＤＤＢが与えられた時、問い合せに対する解は次のような節集合である。

Ans＝　｛Ｓ｜Ｇ＞Ｓ　∧　ＤＤＢ⇒Ｓ｝

ここでＧ＞Ｓは（ＧとＳを述語と見なした時）ＧがＳよりジェネラルである（即ちＧの変数を定数または他の変数に置換すればＳが得られる）ことを示し、ＤＤＢ⇒ＳはＳがＤＤＢの論理的帰結であることを意味する。この問い合せを処理するためＧとＤＤＢを与えた時Ansを出力する演繹データベース処理系evalを考える。

Ans　＝　eval（Ｇ，ＤＤＢ）

evalの実現方式は大きくトップダウンとボトムアップの２つに分類される[1)]。トップダウン方式はPrologのような論理型言語処理系と同様にゴールから出発し、ゴールを得るのに必要なサブゴールを順次展開していく。これにたいしボトムアップ方式ではファクト集合から出発して順次ＩＤＢ中の述語に対応する中間解を計算していき、最後にゴールに対応する集合を得る。

演繹データベースが与えられた時、以下のような性質を持つホーン節変換（hct）を考えよう。

ＤＤＢ'　＝　hct（ＤＤＢ）　かつ

eval（Ｇ，ＤＤＢ'）＝　eval（Ｇ，ＤＤＢ）

for any　Ｇ

即ち、hctはＩＤＢ部分を変換し問い合せに対して元のＤＤＢによるものと同じ解を与えるＤＤＢ'を作る。ホーン節変換を行うのは問い合せを簡単化することにより、その後の問い合せ処理の効率化を図るためである。これは通常のプログラムで部分計算を行うことによりプログラムを特殊化し、実行時の高速化を図る[7)]のと同様である。Prologにおける部分評価とその応用も提案されている[8)]。ホーン節変換はＧが前もって与えられていれば一般の場合より単純であり、結果

もより特殊化できる。本稿ではGが与えられた変換について議論する。この時、hctを用いて問い合せ処理を次のように2段階で行うことができる。

DDB' = hcqt(G, DDB)

Ans = eval(G, DDB)

本論に入る前にさらにいくつかの基本概念を定義する。

DDB中の同じ名前の述語を1つのノードで表した時、節のヘッドの述語がボデイの述語に依存すると考えて依存関係の有向グラフを考えることができる。これをDDBの依存グラフと呼ぶ。ある述語から到達可能な述語とは依存グラフ中でその述語からのパスが存在する述語を意味する。到達可能な節も同様に考えることができる。さらにDDB中の再帰述語とは自分自身に到達可能な述語を言う。自己再帰述語は再帰述語のうち長さ1のパスが存在するもの、即ち自分自身へのアークが存在するものである。また中間述語とはゴールまたはEDBに対応する述語以外を指す。議論を簡単化するためEDBの述語と同じ名前の述語を存在ノードとするIDBにはないと仮定する。もし存在でもこの仮定を導入しても一般性を失わない。

DDB中の成分とは、対応する依存グラフ中の強連結成分に対応する述語集合を言う。依存グラフの強連結成分は成分に属するような任意のノード集合を意味する[3]。同じ成分に分解できる。

問い合せの成分が線形であるとは成分中の述語の定義節の全てについて、ボデイに同じ成分の述語が高々1つしかないことを言う。再帰成分は再帰述語を含む成分を意味する。また問い合せが線形であるとは成分中の全成分が線形であることを意味する。

問い合せの処理はGUDDB上で行うが、これはG∪{DDB中でGから到達可能な節}を対象とした処理でも結果は同じである。ホーン節変換は概念的には簡単のみを含むものを考える。また問い合せの処理でもDDBから議論の様な集合を取り出すためDDBとしてGから到達可能な節

はＧＵＤＤＢの節と同一視する。

３　部分評価と基本ホーン節変換

演繹データベースではＤＤＢをＩＤＢとＥＤＢに分離し、ＩＤＢ処理とＥＤＢ処理に別々のアルゴリズムを用いることが多い。ホーン節変換はＩＤＢを変形する等価変換である。この変換は問い合せ中の一部の述語を削除することにより行うが、この処理をPrologのような論理型言語処理系を利用して行うことができる。即ちＩＤＢの一部の述語は評価してしまい、他は評価しないような部分評価処理系を用いてホーン節変換が実現できる。論理型言語処理系eval_1が与えられた時、以下のような部分評価処理系p-evalを考えることができる。

eval_1は一般に導出原理にもとづいた処理を行う。この処理では対象となっている述語と単一化できる述語がＤＤＢの節のヘッドにある時、対象述語を節のボデイ部で置換える。この時eval_1をもとに以下のような処理系を考える。部分評価時に評価したくないターミナル述語の集合をTermとする。Ｔ∈Termは述語を表わすとともにＴをヘッドとする単位節をも表わすと考える。またＥを、ＥＤＢの同一名のファクト集合に対応したそれらと同じ名前でよりジェネラルな述語を考え、そのような述語の集合とする。Termとして任意の集合を考えても以下の議論を一部修正すれば同様なことが言えるが、ここではＩＤＢの変換を目的としているのでＥ⊂Termと仮定する。処理系eval_1とゴールＧ、演繹データベースＤＤＢ、ターミナル集合Termが与えられた時、次のようなp-evalを考えることができる。

部分評価処理系 p-eval：

（１）トップレベルのゴール自身の処理時はeval_1と同様に動作する。

（２）ゴール以外のサブゴール（ゴールと同じ述語であることもある）の処理はeval_1を変更し以下の(a)(b)のように行う。

(a)ＤＤＢそのものでなくＤＤＢから次のよう

に構成される節集合ＤＤＢ２を対象に処理する。

ＤＤＢ１＝｛Ｐ｜Ｐ∈ＤＤＢ∧（Ｐのヘッド〈Ｔ∈Ｔｅｒｍ）｝

ＤＤＢ２　＝　（ＤＤＢ－ＤＤＢ１）∪Ｔｅｒｍ

(b)ＤＤＢ２の処理中、サブゴールとＴ∈Ｔｅｒｍとの単一化によりリゾルベントを求めた時は、サブゴールとなった述語を記録し置換の履歴をとる。

Ｇ、ＤＤＢ、Ｔｅｒｍを与えた時p-evalの出力は、条件付である結果が成り立つことを示しており、

Ａｎｓ　＝　p-eval（Ｇ，ＤＤＢ，Ｔｅｒｍ）

〈Ｓ，Ｐ-ｓｅｑ〉∈Ａｎｓ

の形をしている。Ｓは導出の結果の１つであり、Ｐ-ｓｅｑにはｓを求めるのに用いた節との置き換えの履歴が入る。

部分評価処理系の定義で本質的なのは（２）の部分であり（１）はゴールがＴｅｒｍの元でもある時に、何もしないで停止するのを防ぐため加えた修正である。この時、〈Ｓ，Ｐ-ｓｅｑ〉は以下のように解釈できる。［もしＰ-ｓｅｑ中に現われる述語が（全て）ＤＤＢより演繹できるならばＳがＤＤＢから演繹できる。］従ってp-evalはＤＤＢについてのＴｅｒｍに関する部分評価処理系でありＡｎｓは［Ｓ：－Ｐ-ｓｅｑ.］という形の節の集合であると考えることができる。ここでＰ-ｓｅｑが空の時は単位節と考える。部分評価の結果について以下の性質がある。

［部分評価の性質］

ＲＤＤＢを（Ｐ∈ＤＤＢ）∧（Ｐのヘッド〈Ｔ∈Ｔｅｒｍ）∧（Ｔはゴールと単一化可能でない））であるようなすべてのＰの集合であるとする。このＲＤＤＢをもとにＲＤＤＢ*を以下のように再帰的に定義する．(a)ＲＤＤＢの元はＲＤＤＢ*の元である。(b)ＲＤＤＢ*の元のボディの述語と単一化可能なヘッドを持つＤＤＢの元はＲＤＤＢ*の元である。(c)(a),(b)以外はＲＤＤＢ*の元でない。またＴＤＤＢをp-evalの出力を節集合と見なしたものであるとする。この時、

ｅｖａｌ（Ｇ，ＴＤＤＢ∪ＲＤＤＢ*）

＝ｅｖａｌ（Ｇ，ＤＤＢ）　　である。

この性質からＤＤＢ'＝ＴＤＤＢ∪ＲＤＤＢ*とすればＤＤＢ'はＤＤＢを部分評価によって変換した節集合と考えることができる。このように構成したＤＤＢ'の元のボデイにはTermの元以外の述語が含まれていることがある。これはＲＤＤＢ*にはＤＤＢの節がそのまま残っているためである。従ってこれをホーン節集合の変換と考えた場合まだ不充分であり、ＲＤＤＢ*の部分についてはさらに部分評価を繰り返す必要がある。これはＲＤＤＢ*についてTermの元をゴールとしてp-evalを適用すれば良い。これを依存グラフでＧに近いTermの元から順に行うと結果としてTermの元の定義節はすべてどれかのＴＤＤＢに生成され、ＲＤＤＢ*は空になるかＴＤＤＢの内容と重複したものになる。従って、ＴＤＤＢのユニオンのみ考えればよい。この集合は有限である限りどの元に対応するものから求めても同じである。またＥＤＢに対応する述語の集合Ｅ（⊂Term）の元については、これらの述語をゴールとしてp-evalを実行しても出力は元のＥＤＢの内容と変わらないのでこの部分についてはp-evalを適用しなくても良い。。従ってＤＤＢを変換してもＥＤＢの部分は変化しないし、ＤＤＢからＥＤＢを除いてp-evalを行ってもＩＤＢの部分の変換は行える。この結果、p-evalを用いてＧとＩＤＢ、Termが与えられた時ＩＤＢ'を出力する基本ホーン節変換ＩＤＢ'＝b-hct（Ｇ，ＩＤＢ，Term）が以下のように定義できる。

```
b-hct：基本ホーン節変換
  ＩＤＢ'＝{}；
  ＳＧ＝Ｇ∪Term－Ｅ；
  do for every  Ｐ∈ＳＧ；
    ＴＤＤＢ＝p-eval（Ｐ，ＩＤＢ，Term）；
    ＩＤＢ'＝ＩＤＢ'∪ＴＤＤＢ；
  ｅｎｄ；
```

ＩＤＢ'に現われるゴール以外の述語Ｐはすべて
Termの元ＴとＰ＜Ｔの関係にある。

ここで定義した基本ホーン節変換はTermの与え方

によりその結果が大きく変化する。最初にTerm＝Eとした場合を考えよう。この時、

ＩＤＢ'＝ b-hct（Ｇ，ＩＤＢ，Ｅ）

で求められるＩＤＢ'にはゴールをヘッドとしＥの元をボデイとする節のみがある。これらの節のボデイの述語がＥＤＢのリレーションを表すと考えて、関係演算に変換しそれらのユニオンをとることにより、ゴールに対応する仮想リレーションが実リレーションの式で表される。この式はもしそれが有限ならば再帰処理機能を持たない通常の関係データベース管理システムで処理可能である。しかしＩＤＢに再帰述語がある場合、ＩＤＢ'は無限集合になってしまいこの変換は停止しないという問題がある。例えば、

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

をＩＤＢとしparentはＥＤＢにあるとする。この時、

Term＝｛parent(X,Y)｝

としてＩＤＢ'は、

```
ancestor(X,Y):-parent(X,Y).
ancestor(X,Y):-parent(X,Z),parent(Z,X).
ancestor(X,Y):-parent(X,Z1),parent(Z1,Z2),
                    parent(Z2,Y).
                :
                :
```

のように無限に続く。この例のように簡単な場合はＩＤＢ'の節に対応するancestorの集合を順次求めていけば、どこかでそれ以上続けても結果がでなくなることを判定できるので、ＩＤＢ'の元を１つずつ求めていけばこの方式でも処理可能である。Prologと関係データベースの接続方法として試みられた遅延評価法や部分評価法は本質的にはこの方法と同じである[9)10)]。しかし一般の場合にはこのような停止条件は知られていないのでこの考えに基づいたシステムの適用範囲は限定されている。従って、ＩＤＢ'をＥＤＢに対応する述語だけで構成することには無理があり、中間述語を含むターミナル述語の集合を考える必要がある。

4　ホーン節変換

演繹データベースの処理方式として現在までに提案されたものの大部分は問い合せ中の述語を仮想または実リレーションに対応させる方法をとっている。従って問い合せ中の述語の数や種類を減らすことにより処理効率を改善することが期待できる。また問い合せの構造が簡単になるのでどのような処理戦略を選択するかの決定が容易になる。我々は問い合せ中の述語を除去し問い合せを簡略化するホーン節変換を提案したが[4]、このアルゴリズムは部分評価によるホーン節変換の特別な場合である。本章では部分評価によるホーン節変換とその性質について述べる。

前章までにのべたようにp-evalは常に停止するとは限らないが、Termの選び方により常に停止させることができる。まず前章と同じくＥＤＢにあるリレーションを表わす述語はTermの元であるとする。また比較述語（＝，＜，等）を導入する場合、これらはデータベース中の定数の組の上に定義されたリレーションと考えることができるので、Termの元とする。比較述語は関係演算による処理時には制約や結合演算の条件に変換される。演繹処理が無限に続き停止しなくなるのはＤＤＢ中の再帰述語を繰り返し処理するためなので、ＤＤＢに再帰定義がなければp-evalが、従ってb-hctが停止することは明らかである。従って、すべての再帰述語をTermの元とすればb-hctは停止する。ＤＤＢ中の全再帰述語をTermの元とすることにより得られるホーン節変換をタイプ１ホーン節変換と呼ぶ。

タイプ１ホーン節変換（hctl）

Ｅ＝｛実リレーションを表わす述語｝，Ｃ＝｛比較述語｝，Ｒ＝｛再帰述語｝とした時、

ＩＤＢ' ＝ b-hct（Ｇ，ＩＤＢ，Ｅ∪Ｃ∪Ｒ）

＝ hctl（Ｇ，ＩＤＢ）

となるような、ＧとＩＤＢを入力としＩＤＢ'を出力とする変換hctlをタイプ１ホーン節変換という。

hctlの結果は有限で一意に定まり、ＩＤＢ'の元と

なる節のボデイにはＥＵＣＵＲの元の述語のみが現われる。hct1の実現は再帰述語の検出と部分評価実行の２つの部分に分けることにより容易に行える。タイプ１ホーン節変換により非再帰型でボデイに現われる中間述語はすべて除去できる。しかし依存グラフ中に長いサイクルがある場合、このサイクル中の述語は全て再帰述語になるのでhct1によっては簡略化が行えない。従って次の例の様な簡単な線形問い合せでも簡略化ができず単純な問い合せにのみ適用できる効率の良い処理アルゴリズム[3)4)]が適用できない。

```
a(X,Y):-r1(X,Y).
a(X,Y):-r2(X,Z),b(Z,Y).
b(X,Y):-r3(X,Y).
b(X,Y):-r4(X,Z),a(Z,Y).
```

この例を簡略化できるもっと強い変換を考えよう。問い合せを依存グラフで表わした時、１つのサイクルとそこに致るパスについて、サイクルを構成する述語のうちもっともゴールからの距離が短いものをゴール依存再帰述語と呼ぼう。ゴールから１つのサイクルへのパスが複数ある時はそのそれぞれについてゴール依存再帰述語が存在すると考える。ゴール依存再帰述語はゴールを導出原理によって処理する時、それをゴールをルートとして展開するＡＮＤ／ＯＲ木[11)]で表わすと、各パスにおいて各サイクルに属する述語のうち最初に２度同じものが現われる述語でもある。従って、次のホーン節変換を導入できる。

タイプ２ホーン節変換（hct2）：

ＥとＣをhct1と同様にとりＲ＝｛ゴール依存再帰述語｝とする。

ＩＤＢ' ＝ b-hct（Ｇ，ＩＤＢ，ＥＵＣＵＲ）

＝ hct2（Ｇ，ＩＤＢ）

となるＧ、ＩＤＢを入力としＩＤＢ'を出力とする変換hct2をタイプ２ホーン節変換と呼ぶ。ゴール依存再帰述語の定義からhct2は必ず停止する。

ゴール依存再帰述語はeval_1を変更し木展開で最初に同一の述語が現われた時、それを記憶しその述語をそれ以上展開しないようにしたものを用いれば求める

ことができる。先に提案したhct[4]はhct2と同等でありそこにはゴール依存再帰述語を求めるアルゴリズムも記述されている。明らかに｛ゴール依存再帰述語｝⊂｛再帰述語｝なのでhct2はhct1より強い変換である。前記の例にhct2を適用すると、a(X,Y)がゴールの時、ＩＤＢ'として、

```
a(X,Y):-r1(X,Y).
a(X,Y):-r2(X,Z),r3(Z,Y).
a(X,Y):-r2(X,Z),r4(Z,Z1),a(Z1,Y).
```

を得る。

hct1もhct2も変換アルゴリズムは必ず停止し、しかも結果が一意に定まる。しかしhct2はこれ以上部分評価によっては述語を消去できないという意味での最も強い変換ではない。hct2の結果残った中間述語は大部分が自己再帰になるが、例外的に自己再帰でないものが残ることがある。一般に自己再帰述語は部分評価の方法では消去できないが、そうでない述語がＩＤＢに含まれる場合は簡略化可能である。従って、結果に現われる中間述語がすべて自己再帰になるような変換を考えることができる。これをタイプ３ホーン節変換と呼ぶ。

タイプ３ホーン節変換（hct3）：

```
R ← ｛ＩＤＢ中の再帰述語｝
repeat;
   R = R − R中でＩＤＢの対応する節が自己再帰
       でない述語（１つ）
   ＩＤＢ' = b-hct（Ｇ，ＩＤＢ，
                   Ｒ∪Ｅ∪Ｃ）
   ＩＤＢ = ＩＤＢ';
until no self-recursive predicate in R;
```

でＩＤＢ'を定義した時、与えられたＧとＩＤＢに対しＩＤＢ' = hct3（Ｇ，ＩＤＢ）となる変換hct3をタイプ３ホーン節変換と呼ぶ。

hct3の結果，ＩＤＢ中の中間述語はすべて自己再帰になるので、部分評価によってはこれ以上の簡略化は行えない。この意味でhct3は最も強い変換である。hct2によっては簡略化できないが、hct3では簡略化可能

な例を図1に示す。ルール集合は省略し依存グラフを示した。

ホーン節変換では一般に問い合せの性質から変換後の形を推定することは困難である。しかし非再帰な中間述語が消去される、即ちこのような述語からなる成分が消去される、という性質とともに次の性質がある。

[性質1] 問い合せの連結成分が線形で、成分中の各述語について再帰述語が現われる定義節が高々1つならばこの成分はhct2（従ってhct3）によって1つの述語からなる成分に縮退する。

この性質が成り立つのは上の条件を充たす問い合せの成分にはサイクルが1つなのでどの順序で述語を消去しても再帰述語のサイクルの数は変化せず最後には1つの自己再帰述語のサイクルが残るからである。線形問い合せでも1つの述語をボディとする複数の再帰的な節があるときは縮退できない場合がある。

またタイプ3ホーン節変換について関係代数の縮退アルゴリズム[3]と同様に次の性質がある。

[性質2] 問い合せの連結成分が1つの述語に縮退可能な必要十分条件はその依存グラフから1つの述語を削除したとき、その成分にサイクルがなくなるような述語が存在することである。

hct3は最も強い変換であるが、結果が消去する述語というの選び方に依存し必ずしも一意には定まらないとまら述語数も一意には定まらない戦略がい。このためhct3では消去する述語を選択する戦略を重要となる。通常は多くのサイクルに含まれる述語をできるだけ残すようにするのが良いと考えられる。ま多くの場合、1つずつ述語を消去することが数項たの述語を同時に消去するのでなく複数あるのを考慮したアルゴリズムの改良は今後の検討課題である。

5 否定の扱い

一階述語論理をホーン節に限定したPrologのような処理系では厳密には”否定”の概念は拡張を行わないと扱えない。Prologではこの問題を閉世界仮説のもとでのnegation as failureとして"not"述語を用いて扱っている[12)13)]。即ち、Pが証明できない時Pの否定が証明されたと見なして扱っている。このような”否定”をホーン節に導入した時、関係データベースにおける関係代数で記述できるものは全てホーン節で記述できる。従って”否定”を加えれば演繹データベースは関係データベースの拡張になる。このためホーン節変換でも”否定”をどのように扱うかが重要である。

”否定”(not述語)を含んだ問い合せは今までのホーン節変換を以下のように変更すれば扱うことができる。この変更は前章のどのタイプの変換にも共通である。

(1) DDB中に現われるnot(P)の形の述語を全てターミナル述語とする。従ってnotの引数は部分評価時には評価されない。

(2) h-hct中でターミナル述語をゴールとした部分評価p-evalを適用する時、そのターミナル述語がnot(P)の形ならnot(P)でなくPをゴールとしてp-evalを適用する。

このような変更を行えば"not(P)"は変換後も保存されるので等価変換の性質は失われない。

6 むすび

演繹データベースの問い合せを簡略化するホーン節変換を部分評価の概念に基づき導入した。問い合せの簡略化をすることにより述語数の削減による効率化や、最適な実行方式選択を容易にする効果がある。変換の出力は入力と同じくホーン節なので関係演算により実行する方式だけでなくトップダウン的な処理方式の前処理としても適用できる。ここで述べた3種の変換の内、タイプ2はPrologとGHCによる2つの演繹データベース実験システム中で実現した。今後はタイプ3の改良の検討を行うとともに全体システムへの組み込みを検討する予定である。

謝辞

本稿の作成にあたり御意見御助言をいただいたＩＣＯＴおよび沖電気工業株式会社のＫＢＭＳ　ＰＨＩ研究の関係者に感謝します。

［参考文献］

1) Bancilhon, F.: An Amateur's Introduction to Recursive Query Processing Strategies, Proc. of ACM SIGMOD, pp16-52 (1986).
2) Henschen, L. and Naqvi, S.: On Compiling Queries in Recursive First-Order Databases, JACM, Vol.31, No.1, pp47-85 (1984).
3) Ceri, S., Gottlob, G. and Lavazza, L.: Translation and Optimization of Logic Queries: The Algebraic Approach, Proc. of 12th VLDB, pp395-402 (1986).
4) Miyazaki, N., Yokota H., and Itoh, H.: Compiling Horn Clause Queries in Deductive Databases: A Horn Clause Transformation Approach, ICOT Technical Report TR-183, ICOT, (1986).
5) 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予稿集、pp1491-1492, (1987).
6) Itoh, H., Takewaki, T., Miyazaki, N., and Mitomo, Y.: Deductive Database System Written in Guarded Horn Clauses, ICOT Technical Report TR-214, ICOT, (1986).
7) 二村良彦：プログラムの部分計算法，電子通信学会誌，第６６巻２号，pp157-165 (1983).
8) 竹内彰一　古川康一：Prologプログラムの部分計算とメタプログラムの特殊化への応用、Proc. of Logic Programming Conf. 85', ICOT, pp155-165, (1985)
9) 田中譲、堀内謙二、田川遼三郎：推論システムとデータベースシステムとの部分評価法による結合、第１回　知識工学シンポジウム予稿集，pp5-10

(1983)
10) Yokota, H., Kunifuji, S., Kakuta, K., Miyazaki, N., and Shibayama, S.: An Enhanced Inference Mechanism for Generating Relational Algebra Queries, Proc. of 3rd ACM PODS, (1984)
11) Kowalski, R.: Logic for Problem Solving, North Holland, p.85 (1979).
12) Clark, K.L.: Negation as Failure, in (eds.) Gallaire, H. and Minker, J., Logic and Data Bases, Plenum Press, pp293-322 (1978).
13) Reiter, R.: On Closed World Data Bases, in (eds.) Gallaire, H. and Minker, J., Logic and Data Bases, Plenum Press, pp55-76 (1978).

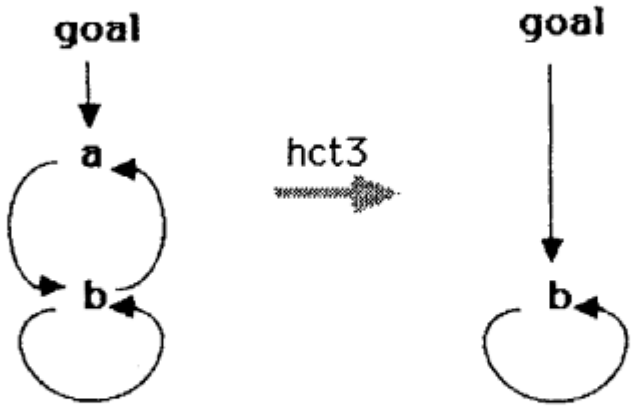

図１　タイプ３ホーン節変換の例

Fig.1 An example of type-3 Horn clause transformation.