データベース・システム　80－6
人　工　知　能　73－6
（1990．11．9）

演繹データベースの実用性に関する一考察

五斗進† 鈴木孝彦† 高木利久‡ 牛島和夫†

†九州大学工学部情報工学科
〒812 福岡市東区箱崎 6-10-1
e-mail:{goto,suzuki,ushijima}@csce.kyushu-u.ac.jp
‡九州大学情報処理教育センター
e-mail:takagi@ec.kyushu-u.ac.jp

概要

演繹データベースシステムのプロトタイプを作成した。このプロトタイプにおいて、演繹データベースのいくつかの質問処理手法を実現した。このプロトタイプを、エンジニアリングデータから機器間の複雑な関係を検索するという実用規模の問題に適用し、効率および表現力を評価した。この評価結果に基づき、エンジニアリングデータの検索に演繹データベースの手法が有効であることを実証した。本論文では、プロトタイプの概要および評価結果を述べ、演繹データベースの実用性を考察する。

A Discussion on the Practicability of Deductive Databases

Susumu GOTO†, Takahiko SUZUKI†, Toshihisa TAKAGI‡ and Kazuo USHIJIMA†

†Department of Computer Science and Communication Engineering, Kyushu University
Hakozaki 6-10-1 Higashiku Fukuoka 812 Japan
e-mail:{goto,suzuki,ushijima}@csce.kyushu-u.ac.jp
‡Educational Center for Information Processing, Kyushu University
e-mail:takagi@ec.kyushu-u.ac.jp

Abstract

We developed a prototype of a deductive database system. In this prototype, we implemented several query processing methods for deductive databases. We applied this prototype to retrieving complex configuration from actual engineering data and evaluated the expressive power and efficiency of the prototype. The result shows that a deductive database system can be put to practical use. In this paper, we describe the overview and evaluation of the prototype to discuss the practicability of deductive database systems.

1. はじめに

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であり、再帰的に定義されたルールや否定を含むルールを用いて、柔軟な質問処理を行なうことができる。演繹データベースでは大量のデータを仮定しているので、再帰質問や否定質問の処理において処理効率が問題となってくる。

そのため、大量データを効率よく扱える質問処理法が研究されてきた。これらの研究成果として、マジックセット法[BeR 87, BaR 88]、NRSU法[NRSU 89a]をはじめとする数多くの質問処理手法が提案されている。また、これらのいろいろな質問処理手法に基づいた演繹データベースシステムを試作した例もいくつか報告されている[MUG 86, BoP 88, BNRST 87, Vie 88, VRK 90]。

しかしながら、それらのシステムを実用規模の問題に適用した例はほとんどない。そのため、演繹データベースや、そこで用いられている質問処理手法が、どの程度実用的であるか実証されていないのが現状である。また、演繹データベースを実用化する上で、いろいろな質問処理手法を具体的にどのように実現すればよいかなどの詳細な実現技法についても未解決な問題が多く残されている。

これらの問題に対する解決の手がかりを得るために、我々は、演繹データベースシステムのプロトタイプDEE(Deducutive Engine for Engineering data)を作成した。今回、このプロトタイプを、石油化学プラントにおける各機器間の接続関係を検索する問題に適用した。石油化学プラントにおける属性データや接続関係データは、一般に複雑、かつ、大規模である。そこで、このようなデータに対する質問を、各種の質問処理手法を用いて行ない、手法の有効性を評価した。

本評価の目的は、以下の問題に解答を与え、実用的な演繹データベースの実現技法を開発することにある。

(1) 演繹データベースシステムの現在の技術レベルは、問題の表現力、質問処理の効率において、どの程度実用性があるか。

(2) 実用的な観点から見た場合、現在提案されている質問処理手法のうち、どの手法が適用可能であるのか、また、適用可能な手法のうち、どの手法が効率がよいのか。

(3) 各種の質問処理法を一つのシステム上でどのように統合、実現すればよいのか。また、それらが行なえたとして、質問に応じて、最適な質問処理法をどのようにして選択すればよいのか。

以下では、今回評価した質問処理法の概要及び、その評価結果について報告する。まず、2節では、現在までに提案されている各種の質問処理法のうち、今回評価したものについて簡単に説明する。3節では、プロトタイプの概要を述べる。4節では、プロトタイプの応用である石油化学プラントの機器配管図面上での接続関係に関する検索問題を紹介する。5節では、質問の例およびその処理法と効率について述べる。6節では、5節での結果を基に演繹データベースの実用性について考察する。

2. 各種の質問処理手法

2.1 再帰質問処理手法

再帰質問処理を効率的に行なうために、現在まで、さまざまな手法が提案されている。その代表的なものとして、マジックセット法[BeR 87, BaR 88]があげられる。マジックセット法はルール変換を行なうことによって、質問処理の効率化を図る手法である。ルール変換を用いて効率化を図る手法としては、この他に、ファクタリング法[NRSU 89b]、NRSU法[NRSU 89a]、コンテキスト法[KRS 90]、カウンティング法[SZ86]などがある。また、Naughtonはone-sided recursion[Nau 87]やseparable recursion[Nau 88]という再帰のクラスを定義し、それらのクラスについて効率のよい手法を提案している。

現在、DEEは上にあげた再帰質問処理法のうち、三つのルール変換手法、すなわち、マジックセット法、NRSU法、コンテキスト法を備えている。その他の手法は、以下の理由により現在採用していない。(1)ファクタリング法はNRSU法を拡張したものと考えられ、適用範囲がNRSU法よりも広い。しかし、多くの実用的な問題においては、NRSU法を適用することができる。(2)one-sided recursionやseparable recursionに対する手法は、マジックセット法と親和性がない。

2.1.1 マジックセット法

直接的な接続関係*joint*から間接的な接続関係*connect*を求めるルール*r1*、*r2*を考える。

r1: connect(X, Y) :-joint(X, Y).
r2: connect(X, Y) :-joint(X, Z), connect(Z, Y).
joint(a, b). joint(b, c). joint(c, d).
joint(d, e). joint(e, f). joint(f, g).

質問が:-*connect(d, Y).*のとき、単純にボトムアップ評価することによって生成されるファクトは

connect(a, b). connect(b, c). connect(c, d).
connect(d, e). connect(e, f). connect(f, g).
connect(a, c). connect(b, d). connect(c, e).
connect(d, f). connect(e, g). connect(a, d).
connect(b, e). connect(c, f). connect(d, g).
connect(a, e). connect(b, f). connect(c, g).
connect(a, f). connect(b, g). connect(a, f).

である。その中から質問 *:-connect(d, Y).* を満足するような解を探し出す。これは、すべての接続関係を求めてから解を探しているので、質問と関係のない解まで導出してしまう。例えば、ここでは *connect(a, b)* や *connect(a, c)* という質問にはまったく関係のない解が導出されている。

マジックセット法は、ボトムアップ処理の際に、質問と関係ない解の導出を避けるために、変数の束縛情報を伝播するようにルールを変換する。上のルールと質問に対して、マジックセット法を適用した結果を以下に示す。

m_connect(d).
m_connect(Z) :-m_connect(X), joint(X, Z).
connect(X, Y) :-m_connect(X), joint(X, Y).
connect(X, Y) :-m_connect(X), joint(X, Z),
connect(Z, Y).

このとき、生成されるファクトは

connect(d, e). connect(e, f). connect(f, g).
connect(d, f). connect(e, g). connect(d, g).

となり、*a*、*b*、*c*を第1引数にとるようなファクトは生成されない。

マジックセット法を用いると、質問処理において生成されるファクト数は、解の個数をnとすると、$O(n^2)$でおさえられるので、無駄な中間ファクトを大幅に削減することができる。

2.1.2 NRSU法

Naughtonらは、再帰ルールのうち、ある特性をもったルールのクラス(右線形ルール)に対してマジックセット法よりも効率のよいルールを生成する変換手法を提案した[NRSU 89a]。

2.1.1の例を考えてみよう。このような接続関係が一方向で枝別れのない例では、解の個数をnとすると生成されるファクト数は$\Omega(n^2)$となってしまう。そこで、次のようにルールを変換する。

m_connect(d).
m_connect(Z) :-m_connect(X), joint(X, Z).
answers(Y) :-m_connect(X), joint(X, Y).
connect(d, Y) :-answers(Y).

このルールで生成される中間ファクトの数は、$O(n)$となり、マジックセット法よりも効率よく処理できる。本稿ではこの手法をNRSU法と呼ぶ。

2.1.3 コンテキスト法

前述の接続関係の例において、質問が

:-t(X), connect(X, Y).

の場合、*t(X)*によって、*X*に複数の束縛が与えられるためNRSU法は適用できない。Kempらは次のような変換を行なうことによりNRSU法と同様の効率を得ることができることを示した[KRS 90]。Kempらは、*connect*に与えられる複数の束縛のうち、どの束縛に対する解を求めているか、という情報(コンテキストと呼ぶ)をルール中に埋め込むことにより、NRSU法の問題点を解決している。

mc_connect(C, C):-t(C).
mc_connect(C, Z) :-mc_connect(C, X), joint(X, Z).
connect(C, Y):-mc_connect(C, X), joint(X, Y).

本稿では、この手法をコンテキスト法と呼ぶ。

2.2 否定および閉質問の処理手法

否定が扱えるように(ルールのボディに否定リテラルの記述を許す)、マジックセット法を拡張した質問処理手法がいくつか提案されている[BMPR 87, KeP 88, Bry 89, STU 89]。プロトタイプDEEでは、これらの手法の中で、否定処理手法として、Kerisitらの提案している手法[KeP 88]およびマジックセット法にインデックスを付加したVIMS法[STU 89]を採用している。

2.2.1 層状化データベースでのボトムアップ処理

DEEでは、データベースが層状化可能(stratifiable)[ABW 88]かつ許容された(allowed)[Cla 78]、という条件のもとで、ルールのボディに否定リテラルを記述可能である。マジックセット法やNRSU法で「ルールの書き換え」を行なうと、元のデータベースが層状化可能である場合にも、変換後のデータベースが層状化不可能になる場合がある。

DEEでは、変換後のデータベースにおける否定処理を正しく行なうために、Kerisitらの提案した否定処理法を採用している[KeP 88]。Kerisitらの処理法の概要を以下に示す。

(a) 変換後のデータベースを弱層状化(weakly-stratified)する。すなわち、否定リテラルをボディに持つルールは、その否定リテラルの述語をヘッドに持つルールよりも上の層に属するようにルールを分類する。

(b) まず最下層(層0)において導出を行なう。新しいファクトが導出されなくなるまで繰り返す。次に層1において導出を行なう。層1で新しいファクトが導出されたならば、再び層0に戻って導出を行なう。層1で新しいファクトが導出されないならば、層2における導出を行なう。同様に、層kにおいて、新しいファクトが導出されたならば、層0に戻る。層kで新しいファクトが導出されないならば、層$k+1$に移る。

(c) すべての層において新しいファクトが導出されなくなったならば終了する。

2.2.2 VIMS法

ボトムアップ処理では、閉質問に対してもすべての解を求めてしまう。VIMS法は、マジックセット法において、閉質問に対する解を一つだけ導出することによって、効率化を図る手法である[STU 89, STU 90]。層状化可能かつ許容されたという条件のもとでは、否定は閉質問とみなすことができる。そのため、VIMS法は否定を含む質問を効率的に処理する。

VIMS法では、まず、個々の閉質問にインデックスを付加し、複数の導出を区別する。そして各インデックスに対して、不必要な導出が発生しないように監視することで、複数の閉質問を同時に、しかも、効率よく処理する。インデックスの考え方は、コンテキスト法におけるコンテキスト引数と類似している。すなわち、どの閉質問に対する解を求めているかを、インデックスとして保持しておき、解が一つ導出されたらその閉質問に対する導出を終了する。

VIMS法とKerisitらの処理法は併用可能である。

3. 演繹データベースシステムのプロトタイプ

我々は、演繹データベースシステムのプロタイプDEE(Deducutive Engine for Engineering data)をSUN-3/80ワークステーション(記憶領域12Mbyte)上に作成し、各種の質問処理手法を実現、評価した。このプロトタイプの基本部分は、ルール変換部とボトムアップ評価部とからなる。ルール変換部は約3,000行のPrologプログラムで、ボトムアップ評価部は約6,000行のCプログラムで記述している。

3.1 ルール変換部

ルール変換部は、ホーン節の形式(ボディに否定リテラルの記述を許す)で記述された検索ルールを、質問に応じて、検索効率のよい検索ルールに書き換えるものである。現在は、2節で説明したルール変換手法をこのルール変換部で実現している。ルール変換部での処理概要を以下に示す。

(1) 右線形ルールかどうかを判定し、そうであれば、そのルールにNRSU法またはコンテキスト法を適用する。

(2) それ以外のルールにはマジックセット法を適用する。

(3) ルール中に否定があれば、データベースを弱層状化する。

(4) 閉質問に対してVIMS法を適用する。(この処理は利用者が指定したときのみ行なう。)

3.2 ボトムアップ評価部

ボトムアップ評価部は、ルール変換部で書き換えられたデータベースの不動点を計算することにより、質問処理を行なう。ボトムアップ評価は基本的にはセミナイーブ法[Ull 89]を用いている。否定を含むルールに対しては、2.2.1節で述べたKerisitらの手法を用いている。

また、DEEのボトムアップ評価部は関数記号を含むルールを処理することができる。しかしながら、関数を許すと解を無限に生成してしまう場合がある。現在は、利用者(質問者)が判断して、解を無限に生成しないようなルールを書かなければならない。処理の停止性を保証するルールのクラスを自動的に判定することは今後の課題である。

ボトムアップ評価部では、不動点演算を高速に行なうため、ファクトをハッシュテーブルで管理している。ハッシュの計算は、定数項の場合は定数そのものを、複合項の場合は関数記号と関数の第1引数の値をもとにしている。

4. 機器配管図面上の接続関係検索

演繹データベースの実用性を評価するために、我々は、石油化学プラントの機器配管図面上の検索問題を用いた。一つの石油化学プラントは、約100枚の機器配管図面からなり、一つの図面は、1,000個以上の機器や配管を含む。しかも、これらの機器や配管は、複合的な属性をもち、かつ、複雑に接続されている。

この図面上で、機器間の複雑な接続関係を検索するには、少なくとも以下の機能が必要である[TSGU 90]。

(a) 再帰: 検索作業には、機器や配管の間接的な接続関係

をたどる必要がある。そのためには、接続関係を再帰的に記述、検索する機能が不可欠である。

(b) 否定: 機器や配管が接続していないことや、ある属性をもたないことを表現するために、否定条件の記述および処理機能が必要である。

このような図面の保全段階における検索作業を行なうには、従来のデータベースシステムの機能だけでは不十分であり、演繹データベースの手法を使う必要がある[TSGU 90]。

5. 質問の例およびその処理法と効率

演繹データベース DEE における質問処理手法の効率測定を、1枚の機器配管図面(機器データ約1,000個、属性・接続関係データ約6,000個; 約300Kbyte)について、Sun-3/80 (記憶容量12Mbyte)上で行った。測定の対象は、ボトムアップ評価部に要するCPU時間と作業記憶領域のサイズである。作業記憶領域のサイズは、質問処理における中間ファクト数で見積もった。

なお、機器データや属性・接続関係データは、既存のDBMSで管理されている。それらはボトムアップ評価を行なう前にホーン節のファクトの形式に変換され、ボトムアップ評価部に渡されるものとする。そのため、ディスクアクセス時間は今回の測定では考慮していない。

また、今回の測定ではルール変換に要する時間も考慮していない。これは、以下の理由により、ルール変換に要する時間は、ボトムアップ処理に比べ一般に無視してもよいと考えられるからである。

- ルール変換はルール数に対し線形時間で行なえる。
- どの処理法を選択するかを決定する手間も、ルール数に対し線形時間で行なえる。
- 演繹データベースでは、一般にファクト数に比べ、ルール数は非常に少ない。

以下に、いくつかの例について測定結果を報告する。測定は、再帰を含まないルールの処理、再帰処理、否定処理、それぞれについて行なった。この際、一部の質問についてはProlog(SICStus Prolog Version 0.7)[Car 90]を用いて処理を行ない、問題の表現力、処理効率について、DEEで採用している各種手法と比較した。

なお、マジックセット法の処理効率はSIP(sideways information passing strategy)[Ull 89] (ボディのリテラルの評価順序) をどのように与えるかによって変わってくる。今回の評価では、すべてのリテラルが、なるべく一つ以上の束縛を持つようにSIPを与えて効率を測定した。

5.1 再帰を含まない質問の処理

例題1：入れ子になったルール

最初の例は、再帰は含まないが、入れ子になっているルールである。この問題は、部品の接続関係からコントロールバルブユニットをすべて検索する問題である。コントロールバルブ *cvvalve* の関係はデータベース中に格納されている。

ルール

```
cvunit(Cv, Node, Type, Name) :-
                    cvunitsub(Cv, Node),
                    node(Node, Code),
                    code(Code, Type),
                    keyx(Node, Name).
keyx(X, Y) :-key(X, Y).
keyx(X, X) :-node(X, Y).

cvunitsub(CvNode, Node) :-nextcv(CvNode, Node).
cvunitsub(CvNode, Node) :-next1cv(CvNode, Node).
cvunitsub(CvNode, Node) :-next2cv(CvNode, Node).
cvunitsub(CvNode, Node) :-next3cv(CvNode, Node).
cvunitsub(CvNode, Node) :-next4cv(CvNode, Node).
cvunitsub(CvNode, Node) :-next5cv(CvNode, Node).
cvunitsub(CvNode, Node) :-next6cv(CvNode, Node).

nextcv(Cv, N) :-cvvalve(Cv),
        jointx(Cv, N), pipe(N).
next1cv(Cv, N) :-nextcv(Cv, N1),
        jointx(N1, N), reducer(N).
next2cv(Cv, N) :-next1cv(Cv, N1),
        jointx(N1, N), pipe(N).
next3cv(Cv, N) :-next2cv(Cv, N1),
        jointx(N1, N), valve(N).
next4cv(Cv, N) :-next3cv(Cv, N1),
        jointx(N1, N), pipe(N),
        jointx(N, N2), kubunten(N2).
next5cv(Cv, N) :-next4cv(Cv, N1),
        jointx(N1, N2), kubunten(N2),
        jointx(N2, N), pipe(N),
        jointx(N, N3), valve(N3).
next6cv(Cv, N) :-next5cv(Cv, N1),
        jointx(N1, N), valve(N).
```

pipe(X) :-node(X, c3000).
valve(X) :-node(X, c1000).
reducer(X) :-node(X, c1001).
kubunten(X) :-node(X, c9003).
cvvalve(X) :-node(X, c1700).

質問

:-cvunit(C, Node, Type, Name).

この質問の評価結果は以下のとおりである。なお、解は98個得られた。

処理法	処理時間	中間ファクト数
無変換	11.60sec	2,017個
マジックセット法	21.36sec	2,066個
Prolog	0.70sec	

例題2:構造をもつデータ

エンジニアリングデータベースでは、構造をもつデータが存在する。例えば、以下のルールにおける、2つの部品の接続を表す関係 *relate* がそれである。我々の処理系DEEでは、3.2節で述べたように、関数表記を使って、構造をもつデータを記述することが可能である。

ルール

dconn(Node1, Name1, Node2, Name2) :-
key(Node1, Name1),
joint(Node1, Node2),
key(Node2, Name2).
joint(X, Y) :-relate(conn(X, X1), conn(Y, Y1)).

質問

:-dconn(Node1, Name1, Node2, Name2).

この質問の評価結果は以下のとおりである。解は8個得られた。

処理法	処理時間	中間ファクト数
無変換	2.30sec	838個
マジックセット法	15.00sec	592個
Prolog	1.16sec	

この結果は、関係データベースで使われる最適化手法『選択操作をなるべく早く行なう』と矛盾している。これには2つの理由がある。まず、現在のDEEでは、データベース中の関係をファクトに変換する際に、述語の引数単位でインデックスを作成する。そのため、上の例で、*relate(conn(X, X1), conn(Y, Y1))* の変数 *X* に与えられる束縛からは、効率のよいアクセスが現在まだ実現できていないからである。第2に *X* に与えられた束縛と、個々のファクトとのユニフィケーションに要する計算の手間が多くなるためである。そのため、*relate* を正規化した次のような *flatrelate* を用い、マジックセット法で処理すると、1.8秒で処理は終了する。構造をもつデータを効率よく扱う手法の開発は今後の課題である。

ルール

conn(Node1, Name1, Node2, Name2) :-
key(Node1, Name1),
joint(Node1, Node2),
key(Node2, Name2).
joint(X, Y) :-flatrelate(X, X1, Y, Y1).

5.2 再帰質問処理

例題3:右線形ルール(1)

ルール

ctovalve(X, Y) :-jointx(X, Y), notvalve(X, Xx).
ctovalve(X, Y) :-jointx(X, X1),
notvalve(X, Xx), ctovalve(X1, Y).

質問

:-ctovalve(n100001, X).

この例は、右線形[NRSU 89a]再帰なので、NRSU法を使うと高速に解ける。これは、ある機器から、バルブを介さずに接続しているすべての部品を求める問題である。

この質問の処理結果は以下のとおりである。解は55個得られた。

処理法	処理時間	中間ファクト数
無変換	242.97sec	16,380個
マジックセット法	16.58sec	1,815個
NRSU変換	0.98sec	55個
Prolog	2.84sec	

この例をそのままPrologで処理すると無限ループにはいってしまう。これを制御するためには、ルールを書き換える必要がある。ここでは、途中の解をリストに保持し、ループチェックを行なうように書き換えて、効率を測定した。このような制御を利用者が記述するのは、必ずしも容易ではない。

例題4:左線形ルール

例題3のルールは次のように変換することができる。このルールは、左線形[NRSU 89a]であるので、NRSU法でもマジックセット法でも効率よく処理できる。

ルール

```
ctovalve(X, Y) :-jointx(X, Y), notvalve(X, Xx).
ctovalve(X, Y) :-ctovalve(X, X1),
        notvalve(X1, Xx), joint(X1, Y).
```

質問

```
:-ctovalve(n100001, X).
```

この質問の処理結果は以下のとおりである。解は55個得られた。

処理法	処理時間	中間ファクト数
無変換	165.80sec	16,380個
マジックセット法	0.80sec	164個

例題5：右線形ルール（2）

次の例は、*connect*を定義した再帰ルールが右線形になるが、*connect*の第1引数に対する束縛が定数ではないのでNRSU法は使えない。そこで、この質問にはコンテキスト法を適用することにする。これは、ある機器('D-801')に接続している配管以外の部品すべてを求める問題である。

ルール

```
conny(Name, Node, Code, Type) :-
        key(N1, Name),
        connect(N1, Node),
        \+ pipe(Node),
        node(Node, Code),
        code(Code, Type).

connect(X, Y) :-jointx(X, Y).
connect(X, Y) :-jointx(X, X1), connect(X1, Y).
```

質問

```
:-conny('D-801', Node, Code, Type).
```

この例題をマジックセット法で処理したところ、10分待っても処理は終了しなかった。コンテキスト法とPrologとによる処理結果は以下のとおりである。なお、解は561個得られた。

処理法	処理時間	中間ファクト数
コンテキスト法	23.78sec	3,798個
Prolog	193.28sec	

ここでも、例題3と同様にPrologで処理する際、ルールを書き換えなければならない。

5.3 否定質問及び閉質問処理

否定の処理には、2.2節で述べたKerisitらの否定処理手法を用いる。また、閉質問の処理にはVIMS法を用いることができる。

例題6：否定質問

次の例題は、再帰的に定義された述語(*connecttocheck*)が否定リテラルとして現れる例である。これは、吐出ライン(コード番号：*c502*)にチェック弁(コード番号：*c1130*)が付いていない遠心ポンプをすべて検索する問題である。この例題は、マジックセット法による変換後Kerisitの評価法を用いることによって、0.45秒の処理時間で解ける（中間ファクト数48個、解4個）。

ルール

```
badcfpump(X) :-centrifugalpump(X),
        \+ connecttocheck(X).
connecttocheck(X) :-jointx(X, Y), check(Y).
connecttocheck(X) :-joint(X, Y),
        \+ valve(X), connecttocheck(Y).
centrifugalpump(X) :-node(X, c501).
centrifugalpump(X) :-node(X, c502).
centrifugalpump(X) :-node(X, c503).
check(X) :-node(X, c1130).
```

質問

```
:-badcfpump(X).
```

例題7：閉質問

以下のルールと質問には、再帰的に定義された述語(*connecttosafety*)が閉質問として現れる。これは、安全弁が付いている機器をすべて検索する例題である。

ルール

```
instwithsafetyvalve(X) :-instrument(X),
        connecttosafety(X).
connecttosafety(X) :-jointx(X, Y),
        safetyvalve(Y).
connecttosafety(X) :-jointx(X, Y),
        \+ blockdevice(Y),
        connecttosafety(Y).
blockdevice(X) :-instrument(X).
safetyvalve(X) :-valve(X),
        jointx(X, Y), pipe(Y),
        onlyjointto(X, Y).
onlyjointto(X, Y) :-\+ jointtoother(X, Y).
jointtoother(X, Y) :-jointx(X, Y1),
        \+ (Y = Y1).
```

質問

:-instwithsafetyvalve(X).

この例題の処理結果を以下に示す。解は11個得られた。

処理法	処理時間	中間ファクト数
マジックセット法 +Kerisitらの手法	252sec	16,765個
VIMS法 +Kerisitらの手法	37sec	5,666個

6. 評価および考察

実用的な問題に対しプロトタイプを適用した結果、演繹データベースの表現力と効率とに関して以下のことが明らかになった。

6.1 表現力

(1) 本稿で取り上げた、石油化学プラントの機器配管図面の間接的な接続関係に基づく検索問題の多くは、再帰と否定とを組み合せることで記述できた。逆に言えば、この種の問題に対しては再帰と否定とは重要で、これらを使わなければ多くの問題が記述できない。

(2) 他の分野のCADやエンジニアリングデータにおいても、部品の属性や間接的な接続関係に基づく複雑な検索を要する問題には、再帰や否定を扱う機能が必須である。したがって、他の分野の検索問題にも演繹データベースの手法が有効であろう。

6.2 効率

(1) 本稿で扱った石油化学プラントの保守作業上の検索例題について、全般的に、DEEは実用的な時間で処理を行なった。これは、マジックセット法を基本とした「ルール変換+ボトムアップ評価」という、DEEにおける質問処理の枠組が、実用的な問題に対し有効であることを示している。

(2) 再帰質問の場合、マジックセット法とNRSU法（またはコンテキスト法）を組み合わせることで、実用的な効率で処理を行なえた。マジックセット法は中間ファクトの数を抑えるのに有効である。しかし、マジックセット法単独では、実用上解けない問題もあり、この場合はNRSU法（またはコンテキスト法）を用いる必要がある。

NRSU法およびコンテキスト法は、マジックセット法に比べて格段に効率がよいが、適用できるルールの範囲が狭い。しかし、実用上はこの範囲内で記述できる問題が多いことが分かった。また、そのままの形では、NRSU法やコンテキスト法が適用できないルールでも、簡単な変換を行なうと適用可能になるようなものがある。そのような変換を自動的に行なうツールを今後開発する必要がある。

(3) 否定を含む質問は、マジックセット法とKerisitらの手法とを組み合わせることにより、実用的な処理が行なえた。

(4) 閉質問に対しては、VIMS法が有効であった（例題7）。この手法をNRSU法やコンテキスト法と組み合わせて使うことを現在検討している。

(5) 再帰を含まない質問に対しても、マジックセット法を適用し、効率化を図ることができるという結果が報告されている[MFP 90]。DEEにおいても、マジックセット変換によって、一部の非再帰的な質問を効率的に処理することができた。しかし、変換によって冗長な補助述語が生成される場合（例題1）や、関数の一部だけに束縛が与えられる場合（例題2）には、マジックセット法によって効率が悪化することがある。前者の場合は、冗長な補助述語[Ull 89]の削除を行なう最適化が可能である。後者に関しては今後の課題である。

6.3 Prologとの比較

再帰を含まない質問では、Prologを用いてそのまま処理した場合、DEEを用いるよりも高速に処理できることがある。この理由は以下のように説明できる。

Prolog処理系は、次のような特徴を持つ。

(p-1) 重複した導出のチェックを行わない。

(p-2) 一般に、コンパイラは述語の第一引数に対するインデックシングを行なうコードを生成する。

(p-3) SICStus Prolog Version 0.7コンパイラは、MC68000シリーズの機械語を生成する。

これに対し、DEEは、

(d-1) すべての導出に対する重複チェックを行なう。

(d-2) ファクトのすべての引数に対してインデックシングを行なう。

(d-3) インタープリタである。

のような特徴を持つ。

(p-3)と(d-3)の比較から、計算コストが同程度である場合、Prologの方がより高速に処理を行なう。また、

非再帰的質問の場合は、重複する導出をチェックする必要がないため、演繹データベースのルールをPrologプログラムと見なして処理できる。この場合には、(p-1)、(d-1)から、Prologの処理コストが少なくなると考えられる。さらに、本稿で述べた例では、ボディリテラルの第1引数が束縛された形が多く、(p-2)と(d-2)との違いは計算コストにあまり影響を与えないと予想される。

上記の考察から、既存のProlog処理系が非再帰的な質問を高速に処理できる理由には、処理系の性能に依存する部分が多いと考えられる。将来、DEEの処理系をより洗練されたものとすることで、非再帰的な質問に対しても、Prologと同程度の処理効率を実現することが可能であろう。

再帰的な質問を処理する場合には、5.2節で述べたように、DEEがPrologよりも問題の表現力、処理効率の点で勝っている。また、Prologでは停止性を保証したり、全解探索を行なったりするためには、利用者が制御を陽に記述する必要がある（5.2節参照）。

6.4 その他のルール変換手法

Naughtonが提案している、検索規則から冗長なリテラルを削除する手法[Nau 89]や制約条件をなるべく早期に検査するように検索ルールを書き換える手法など[TTU 90, MFPR 90, KRBM 89]をDEEのルール変換手法として採り入れる予定である。これらの手法は、実際に有効であることをほぼ確認済みである。この確認は、ルール変換を人間の手で実現することにより行なった。

ルール変換とボトムアップ処理とに基づくDEEの枠組は、これらの効率化手法を組み合わせて用いるのに適している。

6.5 発見的手法

質問処理の効率を評価する際に、今回は発見的手法(例えば、なるべく数の少ない部品から評価を始めること)は、用いていない。発見的手法をSIP[Ull 89]で実現することによって、より高い効率が得られる可能性がある。発見的手法を採り入れたり、マジックセット法で書き換えられたルールをさらに改善したり、質問処理法をいろいろ組み合わせたりするためには、ルール変換を支援する知識ベースが必要になる。これは、今後の大きな課題の一つである。

7. 今後の課題

今後の課題は、石油化学プラントだけでなく、他のCADやエンジニアリングデータの検索問題にも適用してプロトタイプのより精密な評価を行なうこと、ルール変換のための知識ベースを開発すること、関数の効率的な処理法を開発すること、データベースとボトムアップ評価部とのインタフェースを改善すること、などである。

謝辞

石油化学プラントの機器配管図面データを提供していただき、現場で実際に生ずる検索問題を教えていただいた三井石油化学工業の芝尾紘一氏に感謝いたします。なお、本研究の一部は文部省科学研究費補助金重点領域研究(2)（課題番号02249206）の補助を受けている。

参考文献

[ABW 88] Apt, K.R., Blair, H.A.,Walker, A.: Towards A Theory of Declarative Knowledge, in *Foundations of Deductive Databases and Logic Programming*, Minker, J. Ed., pp.89-148, Morgan Kaufmann Los Altos, California (1988).

[BMPR 87] Balbin,I., Meenakshi, K.,Port, G.S., and Ramamohanarao, K.: Efficient Bottom-up Computation For Stratified Databases, *Tech. Report, Dept. Comput. Sci.*, University of Melbourne, Australia (1987).

[BaR 88] Bancilhon, F., and Ramakrishnan, R.: Performance Evaluation of Data Intensive Logic Programs, in *Foundations of Deductive Databases and Logic Programming*, Minker, J. Ed.,pp.519-543, Morgan Kaufmann Los Altos, California (1988).

[BNRST 87] Beeri, C., Naqvi, S., Ramakrishnan, R., Shmueli, O., Tsur, S.: Sets and Negation in a Logic Database Language(LDL1), *Proc. 6th ACM PODS*, pp.21-37, San Diego, California (1987).

[BeR 87] Beeri,C., and Ramakrishnan, R.: On The Power of Magic, *Proc. 6th ACM PODS*, pp.269-283, San Diego, California (1987).

[BoP 88] Bocca, J. and Pearson, P.: On Prolog-DBMS connections: a step forward from Educe, in *Prolog and databases*, Gray, P. and Lucas, R., eds., pp.55-66, Ellis Horwood, Chicester, England (1988).

[Bry 89] Bry, F.: Logic Programming as Constructivism: A Formalization and its Application to Databases, *Proc. 8th ACM PODS*, pp.34-50, Philadelphia, Pennsylvania (1989).

[Car 90] Carlsson, M.: *SICStus Prolog User's Manual*, Swedish Institute of Computer Science, Sweden (1990).

[Cla 78] Clark, K.L.: Negation as Failure, in *Logic and Data Bases*, Gallaire, H., and Minker, J., Eds., pp.293-324, Plenum Press, New York (1978).

[KRBM 89] Kemp, D., Ramamohanarao, K., Balbin, I. and Meenakshi, K.: Propagating Constraints in Recursive Deductive Database, *Proc. NACLP*, pp.981-998, Cleveland (1989).

[KRS 90] Kemp, D., Ramamohanarao, K., and Somogyi, Z.: Right-, Left-, and Multi-Linear Rule Transformations that Maintain Context Information, *Proc. VLDB'90*, pp.380-391, Brisbane, Australia (1990).

[KeP 88] Kerisit, J.M. and Pugin, J.M.: Efficient Query Answering on Stratified Databases, *Proc. Int. Conf. on Fifth Generation Computer Systems*, pp.719-726, Tokyo (1988).

[MUG 86] Morris, K., Ullman, J.D., and Gelder, A.V.: Design overview of the NAIL! system, *Proc. 3rd Int. Conf. on Logic Programming*, pp.554-568 (1986).

[MFP 90] Mumick, I., Finkelstein, S., Pirahesh, H.: Magic is Relevant, *Proc. ACM SIGMOD*, pp.247-258 (1990).

[MFPR 90] Mumick, I., Finkelstein, S., Pirahesh, H. and Ramakrishnan, R.: Magic Conditions, *Proc. ACM PODS* (1990).

[Nau 87] Naughton, J.F.: One sided recursions, *Proc. ACM PODS*, pp.340-348, San Diego, California (1987).

[Nau 88] Naughton, J.F.: Compiling separable recursions, *Proc. ACM SIGMOD*, pp.312-319 (1988).

[Nau 89] Naughton, J.F.: Minimizing Function-Free Recursive Inference Rules, *J.ACM*, pp.69-91 (1989).

[NRSU 89a] Naughton, J.F., Ramakrishnan, R., Sagiv, Y., and Ullman, J.D.: Efficient evaluation of right-, left-, multi-linear rules, *Proc. ACM SIGMOD'89*, pp.235-242 (1989).

[NRSU 89b] Naughton, J.F., Ramakrishnan, R., Sagiv, Y., and Ullman, J.D.: Argument Reduction by Factoring, *Proc. VLDB'89*, pp.173-182 (1989).

[STU 89] Suzuki, T., Takagi, T., and Ushijima, K.: Efficient Bottom-up Evaluation of Negative Closed Queries on Stratified Databases, *Proc. Advanced Database System Symposium'89*, pp.143-150, Kyoto (1989).

[STU 90] Suzuki, T., Takagi, T., and Ushijima, K.: The Multiplicity of Answers and its Reduction on Deductive Databases, *Proc. Logic Programming Conf.*, pp.75-83, Tokyo (1990)(in Japanese).

[SZ86] Sacca, D., Zaniolo, C.: The generalized counting methods for recursive logic queries, *Proc. the First International Conference on Data Theory* (1986).

[TSGU 90] 高木利久、鈴木孝彦、五斗進、牛島和夫: エンジニアリングデータ検索における演繹データベースの有効性について、アドバンスト・データベース・システム シンポジウム (1990). (発表予定)

[TTU 90] Tsutsumi, F., Takagi, T., and Ushijima, K.: An Effective Program Transformation of Logical Recursive Queries in Deductive Databases, *Proc. Far-East Workshop on Future Database Systems*, pp.13-25, Melbourne (1990).

[Ull 89] Ullman, J.D.: *Princeples of Database and Knowledgebase Systems*, Vol.II, Computer Science Press (1989).

[VRK 90] Vaghani, J., Ramamohanarao, K., and Kemp, D.: Design overview of Aditi: A deductive database system, *Proc. Far-East Workshop on Future Database Systems*, pp.202-216, Melbourne (1990).

[Vie 88] Vieille, L.: Recursive query processing: fundamental algorithms and the Dedgin system, in *Prolog and databases*, Gray, P. and Lucas, R., eds., pp.135-158, Ellis Horwood, Chicester, England (1988).