# 推論機能と関係データベースの融合
## —実行方式の評価—

5L-8

吉田　敦[†]　土井　晃一[†]　大森　匡[†]　嶺野　和夫[††]　田中　英彦[†]

[†]東京大学　工学部　　[††]富士通株式会社

### 1．序論

エキスパート・システムやＣＡＤなどの応用に知識ベースを用いる場合，大量の知識を効率良く扱う機能が必要になる。この機能を実現するため，我々は，関係データベースマシンを基本にした知識ベースマシンの構築方法を検討している。知識ベースの構築方法として関係データベースを用いる方法には，互いに独立した関係データベースと，推論機構を，通信チャネルを経由して接続する結合方式と，関係データベースを拡張して，その中で推論機能を実現する融合方式がある。通信の問題，各要素の稼働率の問題を考えると，融合方式の方が，優れているが，実際のシステムはまだ存在していない。

### 2．推論実行方式

我々は，機能を拡張された関係データベースの中で推論機能を実現する融合方式を，知識ベースの構築方式として採用する。実行環境として，hashとsortを用いて関係演算を高速に実行するようなデータベースマシン〔1〕を，拡張関係演算を実行できるように機能拡張したものを想定する。推論処理は，このマシンで実行される拡張関係演算（ＥＸＴＲＡ）の系列，または，ＥＸＴＲＡを用いた処理として実現される。

このような環境でProlog と同じ推論機能を実現する方法として次のような方法が考えられる。

拡張された関係演算（単一化結合，単一化選択等）を用いて，reduction を実現する。これにはインタプリタ方式とコンパイラ方式の２通りが考えられる。

推論操作を，拡張関係演算の系列に展開し，その系列を実行する。

質問に対する処理は，質問を受ける前に，質問の前処理により，実行方法の選択・組合せ，リテラル評価戦略の決定を行う。質問を受けてからの処理は，この決定に従って実行される。

#### 2.1 reduction の実行

インタプリタ方式

質問（初期ゴール）のリテラルを，予め決定した計算規則に従って処理し，factに対しては，変数の束縛の集合を表す束縛情報を生成し，単一化結合により，それまでの束縛情報との間のconsistency check を行う。ruleに対しては，その時点での束縛情報から，各束縛値の組に対するゴールのインスタンスを生成し，このインスタンスの集合とruleの定義の集合の間で，単一化結合を実行する。得られた結果から，新しいゴールを生成する。

コンパイラ（遅延評価）方式

質問（初期ゴール）を，ruleの定義を用いて，factの系列に展開し，得られた系列から，関係演算の系列を生成して，関係データベースで実行する。

前者は後者と比較すると，束縛情報により，探索空間の縮小を行う為，拡張関係演算の実行にかかる手間が少なくなる。従って，後者より処理時間が短くなる。但し，独立した推論機構と関係データベースをつないだシステムでは，関係データベースと推論機構の間の通信量が多くなるため，実行効率は極めて悪い。融合方式ではこのような問題を考えなくてすむので，推論処理方式の柔軟な選択が可能になる。

#### 2.2 関係演算系列への展開

Prolog における推論操作は，基本的には定義に対する探索であり，再帰的定義のない場合，または，再帰的定義があっても，単純な再帰の場合には，質問に対する一連の処理手順は，拡張関係演算のみからなる系列で表現可能である。そこで，ruleやfactの定義に関する情報（rule・factの間の呼出し関係，定義の数，単一化が成功する確率）を質問を受ける前に調べて，実行方法を決める時に，拡張関係演算の系列にできる部分は，これに相当する関係演算系列を生成し，質問に対する処理を実行する時は，この系列を実行する。関係演算の系列に展開できない部分については，reduction を実行する。この方法では，関係演算の系列に展開できた部分については関係演算系列の最適化等により，更に効率を上げることが可能である。

### 3．方式の評価方法

ここでは，拡張関係演算の実行に要する手間がシステム全体の性能を左右すると仮定し，拡張関係演算の実行にかかる手間で，各実行方式の比較と評価を行う。

Integration of Inference Facility and RDB — Evaluation of Strategy —
A. Yoshida[†], K. Doi[†], T. Ohmori[†], K. Yamane[††], H. Tanaka[†]
[†]University of Tokyo, [††] FUJITSU LIMITED

### 3.1 シミュレーションの方針

各推論方式を，処理の手間で比較するため，C言語によるシミュレータを作成し，単一化結合（ゴールインスタンスの集合と定義の集合の間のマッチング，consistency checkの実行時），ゴールインスタンスの生成，束縛の生成，新しいゴールの生成にかかる手間，推論処理のサイクル数，ゴールの数の最大値を調べてみた。単一化結合にかかる手間は，hashとsortによる関係代数マシンの拡張版を想定しているので，2つの関係の濃度の和に比例すると考える。ゴールインスタンスの生成，束縛の生成，ゴールの導出は，それぞれ，束縛情報の濃度，factとゴールとの単一化結合の結果の濃度，ruleとゴールとの単一化結合の結果の濃度に比例すると考える。各処理に対する手間は，質問に対する全解探索にかかる手間とする。実験に用いた例題は，いづれも，簡単な例題である。

### 3.2 単一化成功率の変化

factに対する束縛情報によりruleの候補が残る確率を単一化成功率と定義する。この値を変化させると，次のようなことがわかる。

インタプリタ方式においてゴールインスタンス生成の手間は，単一化成功率にほぼ比例する。単一化成功率が高いと，コンパイル方式よりこの手間が多くなる。

ゴールと定義の間の単一化結合は，多くの場合，インタプリタ方式の方が手間が少ない。但し，再帰的な定義のある場合，インタプリタ方式における手間が，コンパイル方式より多くなる場合がある。

consistency check にかかる手間は，インタプリタ方式では，コンパイル方式より1～2桁少ない手間で済む。

### 3.3 評価順序の変更

2つのfactからruleに対する絞り込みがある場合，2つのfactの間で順序を換えても，処理の手間に差は見られない。ruleをどちらかのfactより先に評価した場合，処理の手間は，最後のfactにより落される筈のrule定義に対する分だけ多くなる。ruleを2つのfactより先に評価すると，処理の手間は遅延評価方式の場合とほぼ同じになる。

1つのfactから2つのruleに対して絞り込みのある場合にも，ruleをfactより先に評価すると，そのruleの候補で，factにより落される筈の定義に対する処理の分だけ手間が余計にかかる。

## 4．考察と今後の課題

以上の実験結果より，ゴールと定義の間でマッチングを行う場合，常に単一化結合を実行するのは，あまり効率が良くないことがわかる。これは，ruleが定義の場合や，束縛情報のない場合，1対多の単一化結合になる場合がおこるからである。このような場合，単一化選択を用いる（単一化可能な定義のみをフィルタを通してとりだし，単一化・縮退操作を施す）方が良い。

評価順序に関しては，共有変数の存在等により，絞り込み効果の高いfactとruleがある時は，factを先に評価する戦略は，関係演算にかかる手間の削減効果が大きい。従って，これを実行制御の戦略（リテラルの選択等）に取り入れることで処理の効率を上げることが可能である。

今後の課題としては，関係演算の系列に置き換えて実行した場合の評価，評価可能述語がある場合の推論処理方式（データフローの概念を導入），asseert,retract 等の，知識に対する更新操作のある場合の推論処理制御（知識に対してlockをかけ，更新を実行），アーキテクチュアの検討等があげられる。

例題1　単一化成功率の変化

束縛情報による絞り込み方式

| 単一化成功率 | 20% | 40% | 60% | 80% | 100% |
|---|---|---|---|---|---|
| インスタンス生成 | 29 | 39 | 61 | 83 | 105 |
| 要求定義間単一化結合 | 91 | 95 | 157 | 219 | 281 |
| 束縛情報の生成 | 16 | 21 | 32 | 43 | 54 |
| 新しいゴールの生成 | 12 | 18 | 29 | 40 | 51 |
| consistency check | 10 | 24 | 42 | 60 | 78 |

遅延評価方式

| 単一化成功率 | 20% | 40% | 60% | 80% | 100% |
|---|---|---|---|---|---|
| インスタンス生成 | 57 | 57 | 57 | 57 | 57 |
| 要求定義間単一化結合 | 550 | 562 | 574 | 586 | 598 |
| 束縛情報の生成 | 250 | 265 | 275 | 280 | 285 |
| 新しいゴールの生成 | 56 | 56 | 56 | 56 | 56 |
| consistency check | 275 | 308 | 382 | 426 | 440 |

例題2　評価順序の変更

f1->r1，f1->r2の単一化成功率　20%

① f1，r1，r2の順に評価

② r1，f1，r2の順に評価

③ r1，r2，f1の順に評価

| 評価順序 | ① | ② | ③ |
|---|---|---|---|
| インスタンス生成 | 29 | 57 | 232 |
| 要求定義間単一化結合 | 91 | 170 | 1080 |
| 束縛情報の生成 | 12 | 40 | 75 |
| 新しいゴールの生成 | 16 | 16 | 156 |
| consistency check | 10 | 30 | 100 |

参考文献

〔1〕M.Kitsuregawa ：Relational Algebra Based on Hash and Sort：GRACE　東京大学学位論文