推論機能と関係データベースの融合

1M-5

－ 関係データベースを用いたProlog インタプリタの試作と評価 －

吉田 敦　田中 克己　土井 晃一　大森 匡　田中 英彦

（東京大学 工学部）

1．はじめに

現在研究が行なわれている知識ベースマシンは、知識表現形式として、一階述語論理、意味ネットワーク、プロダクションルール等を採用している。本研究では、一階述語論理のサブセットであるホーン節を扱うための知識ベースマシンの構築法を検討している。

2．実験

本研究では、膨大な量の知識を扱うことのできる知識ベースマシンとして、関係データベースとProlog マシンを融合したものを想定している。このようなマシンのシステム構成を考えるための材料として、Prolog インタプリタと関係データベース管理システムからなる簡単な実験システム（図1）を試作し、性能の測定と評価を行ってみた。

2.1 実験システム

実験システムでは、推論機構側にCProlog ver. 1.5 を、関係データベース側にＩＮＧＲＥＳ ver. 7.10を操作するCのプログラム（clause server ）をもちいた。推論機構側は、demo述語の形式で記述されている。質問に対する処理方式には、評価型と非評価型がある。前者は、質問をfactの系列に展開してから、関係データベースに検索コマンドを一度に送る方式である。この方式は、通信回数がすくなくてすむ、関係データベースでの処理の最適化がしやすい等の長所を持つ。後者は、factが導出されるたびに、関係データベースに問い合わせを発生する方式で、変数に対する束縛情報（これが格納される関係をbinding-relationと呼ぶことにする。）が早期に得られるので、探索木の枝刈りの早期実行により関係データベースへの無駄な質問を回避できる。本研究では、これまであまり評価されていなかった、後者の方式を中心に実験と評価を行なってみた。

2.2 測定

実験は、何種類かの例題に対して、factに対する定義の数を変えながら、推論機構側での処理のステップ数、処理に要した時間、関係データベース側での各関係演算の回数、関係演算に要した時間を測定した。

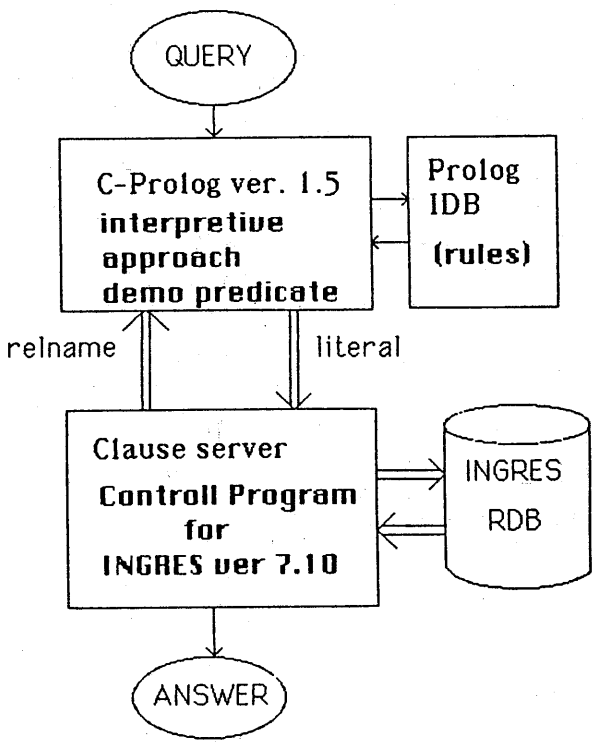

図1　実験システムの構成

なお、構造体である引数は、ここでは扱っていない。また、規則は全て推論機構側にあるものとした。

さらに、ボトルネックになる部分を明らかにするため、推論機構側の処理を、探索木の展開、通信の前処理、通信の実行、通信の後処理にわけ、各グループ毎の、処理ステップ数、実行時間を測定した。関係データベース側に対しても同様に、各関係演算（join, selection , projection, その他）に対する実行時間を個別に測定してみた。

2.3 測定結果

同一の問題でfactの定義数だけを変えた場合には、推論機構側の処理ステップ数は変わらない。問題を変えることにより、factが導出される回数が増えると、推論機構側の処理ステップが大きく増えた。これは、非評価型の方式をとっているため、factを一回導出するごとに通信を行なうためで、しかも、通信に関連する処理のステップが全体の処理に対して

高い割合を占めていたからである。全体の処理に対して、通信の前処理の占める割合だけでもほぼ５０％であるが、実装方法によっては、この割合はかなり減らせるはずである。

関係データベース側での処理は、factの導出回数が決まると、それに必要な関係代数演算の種類と数も決定される。factの導出回数をＮとすると、実験で用いたシステムではselection がＮ回、projectionが（Ｎ＋１）回、joinが（Ｎ－１）回である。各関係演算に要する時間は、リレーションのタプルの数とは関係なしに存在する演算準備のオーバヘッドと、リレーションのタプルの数による時間にわかれる。後者は、joinではタプルの数の二乗に比例し、他の二つの演算ではタプルの数に比例する。従って、タプルの数が少ないうちは、全体の処理時間は大体一定であるが、タプルの数が多くなると、join演算にかかる時間が支配的になる（図２）。実験で得られた結果も、ほぼこの傾向を示しており、タプル数が 100以下で、ほぼ一定の処理時間、タプル数1000個以上でjoin演算の影響が強く出始めていた。

３．改善のための提案

以上の実験を通して、非評価方式を取った時の、Ｐrolog-ＤＢＭ（二次記憶系に関係データベースを用いたＰrolog マシン）の構成の方法について、いくつかの提案を述べてみる。

3.1 資源割り付けの最適化

factの定義数が多い時にfactの処理を関係データベース側で行なうのは、探索木の枝の数を抑えるためであるが、実験結果が示すように、factの数が少ない時には、処理時間はあまり短くないので、むしろ、推論機構側で処理したほうがよい。また、実際の応用例を考えてみても、factの定義数が多いものばかりとは限らない。

従って、factの定義数が少ない時は、factの定義を推論機構側に転送し、推論機構側で処理するような資源割り付けを考えた方がよい。本研究で用いた実験システムでは、ruleの定義も関係データベースに格納し、ruleの定義が必要になったら、hashとsortを用いた検索［１］で必要な定義を推論機構に転送する機構を実現する予定であり、これをfactの定義の転送に用いてやればよい。

3.2 lazy-join

非評価方式の欠点として、consistency-check のためのjoinの順序の最適化ができないことがあった。これは、factに対するbinding-relationをつくるたびにjoinを実行していたからである。そこで、binding-relationを作っても、すぐにjoinを実行しないで、いくつかbinding-relationが揃った時点で中間で生じるrelationのサイズが小さくなるよう、joinの順序を最適化し、この順序でjoinを実行すればよい。これをlazy-join と呼ぶことにする。

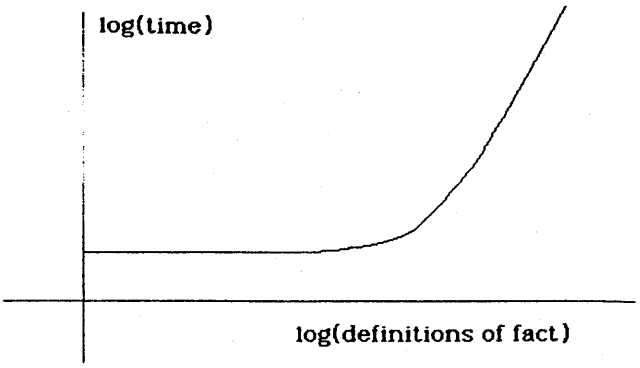

図２　処理時間と定義数の関係

lazy-join は、factの定義数にばらつきが十分にあると効果を発揮する。但し、推論機構で探索木の枝刈りに用いるbinding 情報（変数の束縛に関する情報）の作成が多少難しくなる。

3.3 関係データベースマシンの利用

factの定義数の多い時には、関係データベース側での処理時間は、join演算の実行時間に支配される。joinの実行時間は、通常の関係データベースシステムではＯ（Ｎ＊Ｎ）ないしＯ（Ｎlog Ｎ）になる。これを下げるためには、hashとsortを用いた関係データベースマシン（例えば［２］）の採用によりjoin演算の実行時間のオーダをＯ（Ｎ＋Ｍ）にする必要がある。

４．おわりに

非評価方式は、通信のオーバヘッドの多さや、関係データベース側での演算順序の最適化ができない等の問題があった。その反面、factの処理で得られたbinding 情報を探索木の枝刈りに使用できる、factに対する処理の結果を、次の述語で利用する方法が実現しやすいといった長所を持っている

従って、今後、上記評価を並列推論システムに拡張するとともに、３で述べた改善方法により、効率の高いシステムを構成することが次の課題である。

＜＜参考文献＞＞

［１］大森　他、「推論機能と関係データベースの融合　－ハッシュとソートによる検索－」第32回情報処理学会全国大会１Ｍ－６

［２］喜連川　他、「Hash とSort による関係代数マシン」　信学技報ＥＣ－８１－３５　1981