演繹データベースの問合せ処理へのReteアルゴリズムの適用

1R−1

進藤静一
三菱電機（株）中央研究所

1. はじめに

演繹データベースの問合せ処理アルゴリズム [1] のひとつであるAlexander Templates[2](以下ATと略記)は演繹データベースに蓄えられた任意の確定ホーン節のルールに対して、ボトムアップ計算を用いて、問合せに対する全解を返す。その過程は前向き推論と良く似ている。そこで、前向き推論の高速化技法として知られるReteアルゴリズム [3] をATの実現方法として適用することにより、その高速化を試みた。本稿では、元のルールのReteネットワークへの変換方式、Reteネットワークの動作を中心に報告する。

2. 問合せ処理アルゴリズム AT

ATでは、ルールのトップダウン解釈をボトムアップ実行で模擬できるようにルールを変換し、それをボトムアップに実行することより、問合せに答える。ルールR“a :- $b_1, b_2, \ldots, b_m$”のトップダウン解釈は、

(0) ゴールaが呼ばれるとゴールb_1を呼ぶ。

(i) ゴールaが呼ばれ、且つ、ゴール$b_1, \ldots, b_i$が解かれると、ゴールb_{i+1}を呼ぶ$(1 \leq i \leq m-1)$。

(m) ゴールaが呼ばれ、且つ、ゴール$b_1, \ldots, b_m$が解かれると、ゴールaが解かれた。

である。この解釈を反映して、ATではルールRを以下の$m+1$個のサブルールに変換する。

(sr_0) $call\_a \rightarrow call\_b_1$.

(sr_i) $call\_a, sol\_b_1, \ldots, sol\_b_i \rightarrow call\_b_{i+1}$. (但し、$1 \leq i \leq m-1$)

(sr_m) $call\_a, sol\_b_1, \ldots, sol\_b_m \rightarrow sol\_a$.

ここで、$call_$はゴールが呼ばれることを、$sol_$はゴールが解かれたことを表す。問合せのゴールをpとすると、ATは、以下の手順で計算を行なう。

1. ワーキングメモリ(以下WMと略記)を$call_p$に初期化する。
2. サブルールの左辺要素とWMがマッチするルールの右辺要素の内、WMの既存要素の同型でないもののみをすべてWMに加える。
3. WMがこれ以上変化しなくなるまでステップ2を繰り返す(WMの不動点を求める)。

Alexander Templates based on Rete Network
Seiichi SHINDO
Mitsubishi Electric Corp.

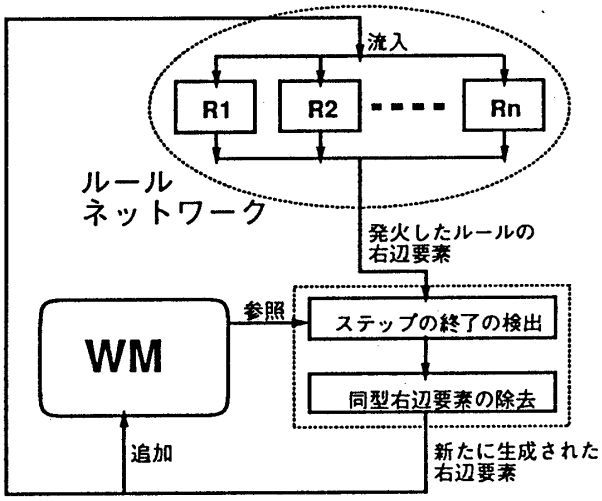

図1: 全体構成

3. 前向き推論としてのAT

前節で示したATのルール、及び、計算手順は以下の2点に於いて、前向き推論の特殊な形態である。

- ルールの形に関して(ファクタリング)： sr_iの左辺要素は全てサブルールsr_{i+1}に含まれる。従って、サブルール間で左辺要素の共通項をくくり出すことができる。
- WMの変化に関して(差分重視のマッチング)：WMの内容は単調増加するので、新たに加わったWM要素とマッチするサブルールのみの発火可能性を調べれば良い。

これらの特徴は共に、ルールの左辺要素とWM間のマッチングの回数の軽減に貢献する。従って、この2点を活かした単純なルール実行系を用いてもそれなりの効果が得られそうである。しかし、ファクタリングと差分重視のマッチングはReteアルゴリズムと相性が良く、Reteアルゴリズムを単純に適用するだけで、ATの処理系が構築できると期待される。

4. Reteアルゴリズムの適用

Reteアルゴリズムを適用した場合のATの処理系の全体構成を図1に示す。前ステップで新たにWMに加えられたWM要素がトークンとしてルールネットワーク(以下NWと略記)に流される。NMは、今までの

マッチング結果を保持しており、それを参照することによりトークンをろ過する。NMを通過し切ったトークンが発火したルールの右辺要素に対応する。

NMは、変換前のルールと1対1に対応するサブネットワーク(以下、SNWと略記)の集合からなる。SNWはinput node、merge node、output nodeの3種類のノードで構成される。ルールRのSNWへの変換は、図2に示すように、左辺要素の共通項をmerge nodeに、非共通項をinput nodeに、右辺要素をoutput nodeに1対1対応させることにより単純に行なわれる。

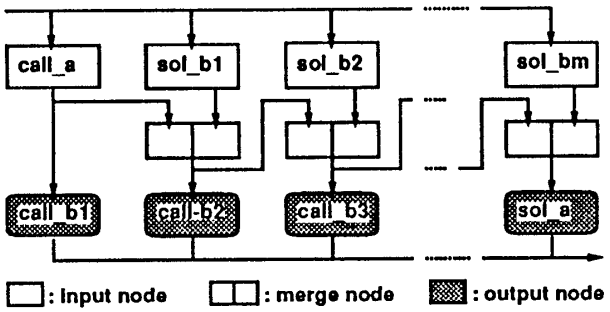

図2: SNWの構成

SNWに出入りするトークンの内容は述語である。ノード間のトークンの内容は、ルールに現れる全変数の値のリストである。リスト内の変数の出現順序はSNW内で共通である。各ノードの機能を以下に示す。

- **input node(= フィルター):** 述語の形で流れてくるトークンと、ラベルがユニファイ可能か調べ、可能な場合のみそのユニフィケーションによるルールの全変数の値をトークンとして流す。
- **merge node(= 保存&フィルター):** 左右2つの入力とそれに対応した2つのメモリを持つ。トークンTが入ってくると、まず、それをメモリに保存する。次に、他方のメモリに蓄えられているトークンの各々とTがユニファイ可能か調べ、可能な場合のみ、そのユニフィケーションによるルールの全変数の値をトークンとして流す。
- **output node(= 整形):** 流れてくるトークンに書かれている変数の値を基に、ラベル内の変数を具体化し、それをトークンとして流す。

"$reach(X,Y)$:- $reach(X,Z), reach(Z,Y)$." なるルールに対応するSNWを図3に示す。この場合、ノード間のトークンは$[X,Y,Z]$となる。I3に流れた$sol_edge(a,b)$と$sol_edge(a,c)$はI3を通過することにより、各々、トークン$[_,b,a]$と$[_,c,a]$となり、M2の右メモリに保存される。その後、各々、M2の左メモリのトークンとユニフィケーションがとられる。その結果、M2から、トークン$[a,b,a]$と$[a,c,a]$が流れ、O3を通過することにより、トークン$sol_reach(a,b)$と$sol_reach(a,c)$が最終的にこのSNWから流れ出る。

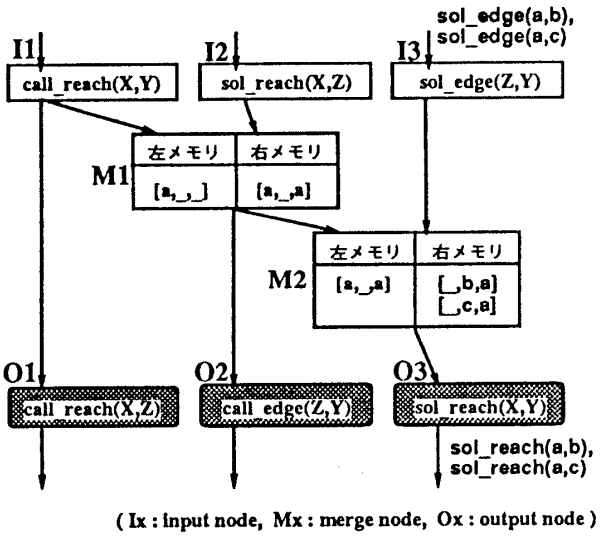

図3: SNWの例

5. 実装

使用マシンはMultiPSI、使用言語はKL1である。KL1は、ストリーム通信を行なうことにより同期をとりつつ並行動作するプロセス群の記述に向く。NWのノードをプロセスに、トークンの流れをストリーム通信に写像することにより簡潔にNWの機能と構造をKL1で記述できた。KL1の提供するユニフィケーションはプロセス間の同期機構に用いられるので、ルールの変数間のユニフィケーションには直接使えない。従って、ルールの変数を基底項表現し、変数間のユニフィケーションを自作した。

6. おわりに

ATのルールの特徴がReteアルゴリズムを導入するのに適したものであるという点に着目して、ATの実装法としてReteアルゴリズムを適用する方法について報告した。KL1での実装の課題として負荷分散がある。トークンの分布やノードでの計算量の見通しが立っておらず、現在、実験を元に方式を検討中である。

謝辞

ATに関して色々教示頂く三菱電機中央研究所の世木博久氏に深謝します。また、日頃御指導頂くICOTの方々に感謝致します。

参考文献

[1] 宮崎, 世木, 「演繹データベースの問合せ処理」, 情報処理, Vol.31, No.2, 1990年2月, pp.216-224.

[2] Seki,H., *"On the Power of Alexander Templates"*, Proc. of 8th ACM Symposium on Principle of Database Systems, 1989, pp.150-159.

[3] Forgy,C.L., *"Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem"*, Artificial Intelligence Vol.19, 1982, pp.17-37.