6F−9

RETEネットワークの動的最適化方法に関する考察*

桑田 喜隆　　牛田 修司†

NTTデータ通信株式会社‡

1　はじめに

エキスパートシステムにおいて、専門家から得たルールを効率良く実行する方法としてRETEマッチアリゴリズムが有望であり[1]、OPS5を初め多くのAIツールで応用されてきている。RETEマッチアルゴリズムでは予めルールの構造を解析し実行時の効率が最高になるような条件木(RETEネットワーク)を生成しておき、RETEネットワークを基にインタプリトしやすい中間形式を生成する。

近年、RETEマッチアルゴリズムに代わるTREAT[2]も提案されているが、否定ノードでの処理が増大してしまうことを考えると実際のエキスパートシステムでは必ずしも有効ではない。

我々は実際に使われるエキスパートシステムでルールを効率良く実行するという観点で、RETEマッチアルゴリズムに注目して最適化の検討を行なっている。

RETEマッチアルゴリズムでは、データ(ワーキングメモリエレメント, WME)の分布が実行時にしか分からないため、静的な解析だけでは最適な条件木を生成することは困難である。本稿では実行時のWMEの分布状況から最適なRETEネットワークを生成する方法について論じる。

2　RETEネットワーク

2.1　RETEの特徴

RETEマッチアルゴリズムでは静的な解析を基にルールをRETEネットワークに展開する。

RETEネットワークは変数単独の条件をチェックするノード(1入力ノード、αノード、イントラノード)と複数の変数間の条件をチェックするノード(2入力ノード、βノード、インターノード)で構成される。2入力ノードには部分的にマッチングした情報を蓄えるメモリ(βメモリ)が用意される。また、否定処理をするための特殊な2入力ノードも用意されている。

2.2　RETEの動作原理

WMEはトークンとしてルートノードから入力され、対応するノードを経て処理される。1入力ノードでは条件の単独チェック、2入力ノードでは変数間のマッチングとマチングしたトークンの蓄積が行なわれ、一連の処理の結果、一番下のノード(ターミナルノード)まで到達したトークンはインスタンシエーションに加えられる。

3　ルールの静的な解析

動的な最適化の前段階として、ルールのコンパイル段階で処理可能な静的最適化が考えられる。静的最適化の基本的な考え方は、冗長なノードを減らしてメモリおよび実行時の処理を減少させることにある。

静的な最適化の方法として、以下のような項目が考えられる。

- **1入力ノードの共有**

  複数のルール間でチェック条件が共通になっている場合、チェックするための1入力ノードを共有する。

- **1入力ノードのマージ**

  分岐のない1入力ノードの列をマージする。マージされたノードの条件は全ての条件のANDとなる。

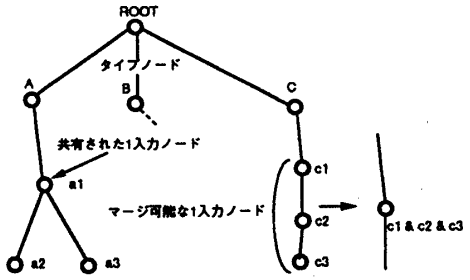

図1：1入力ノードの共有、マージ

- **2入力ノードの共有**

  2入力ノードで複数のルール間で共通のものを共有する。ルールの条件の順序を入れ換えることで2つのルールの条件が共通となるような場合、Rete-network上の2入力ノードも共有することができる。

4　動的最適化方法

実行時には2入力ノードには対応したWMEが蓄積／変更されるが、その量は静的な解析では予想することは

*Dynamic optimization of RETE-network

†Yoshitaka KUWATA, Shuji USHIDA

‡NTT DATA COMMUNICATIONS SYSTEMS CORP.

困難である。そこで、実行時にどれだけのWMEが蓄積または変更されたかを記録しておき、最適化のための情報として使う方法を考える。

- **方法1: 動的な解析とRETEネットワークの再構成**

  あるタイミングでWMEの変化状況を基に最適なRETEネットワークを再構築し、WMEを新しいネットワークに移す。(RE-RETE) この方法では実行時のデータに応じた最適化が可能であるが、RE-RETEの間マッチング処理が行なえず、LISPのガベージコレクションの様に処理が止まってしまうという欠点を持つ。インタラクティブに使うエキスパートシステムでは空いた時間を見つけてRE-RETEを行なうことも可能であるが、リアルタイムエキスパートシステムへの応用等を考えた場合には適応が難しいと考える。

- **方法2: 動的な解析とルールの再コンパイル**

  WMEの変化状況の情報をもとに、ルールをRETEネットワークをコンパイルし直す。再コンパイルの最適化情報として以前の実行結果を使うという方法である。この方法では推論処理中にRE-RETEを行なう必要がないという点で優れている。次の実行で最適化したデータと同じようなデータが入力される保証はないが、実際に使われているエキスパートシステムでは、以前の実行と同じような過程をとる場合が多いため、妥当な方法と考える。

5 インプリメント

我々は社内用にES構築支援ツール「JinK's」を開発している。JinK'sでは予めルールを解析してC言語のルーチンに展開する方式を採用しているため、方法1のような動的な最適化を行なうことが難しい。そこで、方法2による最適化を行なうべく現在、検討、実験をおこなっている段階である。

インプリメントは次のような方針で行なっている。

1. 1入力ノードでは、なるべく早い時期にトークンが少なくなるようにする
2. 2入力ノードではマッチングの処理を減少させるため、βメモリに蓄積されるトークンがなるべく少なくなるようにRETEネットワークを構成する

6 おわりに

RETEネットワークの動的な解析による最適化の方法について述べたが、その有効性についての検証はまだ済んでいない。今後、我々の扱っている実際のエキスパートシステムを例題にとり、動的な最適化の有効性を検証する予定である。

参考文献

[1] Charles L.Forgy: RETE: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, Artificial Intelligence No.19(1982), pp.17-37

[2] Miranker, D. P. : TREAT: A Better Match Algorithm for AI Production System, AAAI'87 pp.42-27

[3] Kazuhiro Kuwabara : Implementation of a Production System : An Object-Oriented Approach, 昭和63年度人工知能学会全国大会 (第2回), pp. 199-202

[4] 稲葉浩人: マッチングアルゴリズムRETEとTREATの比較, 情報処理学会第37回全国大会, pp. 1435-1436

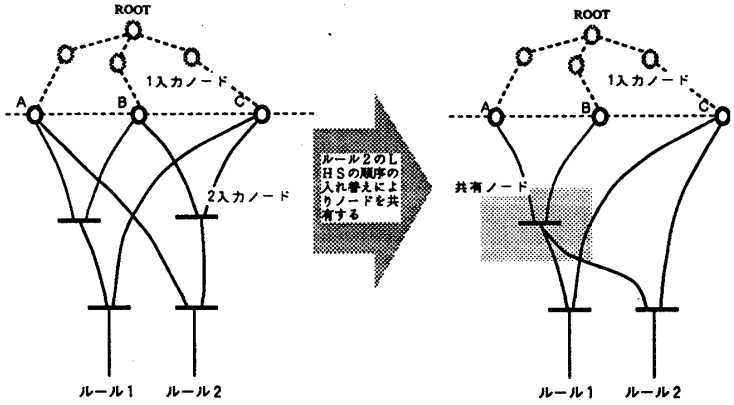

図2：2入力ノードの共有

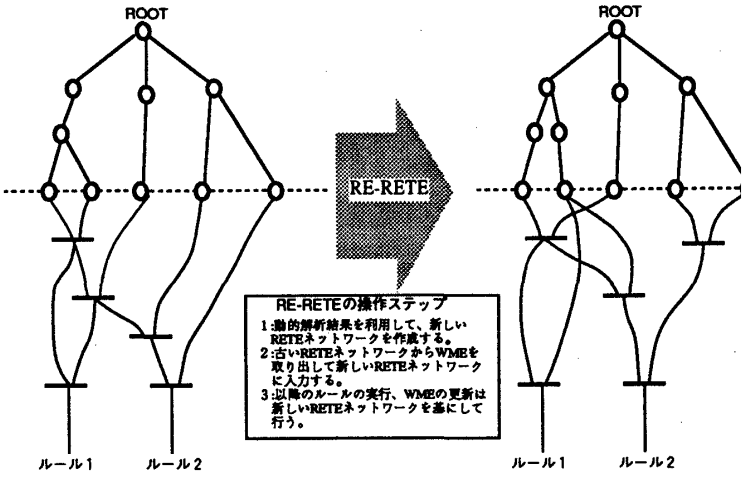

図3：RE-RETE