マッチングアルゴリズムＲＥＴＥとＴＲＥＡＴの比較

6J-1

稲葉　浩人　　中村　明

（株）東芝

１．初めに

プロダクションシステムにおいてのマッチングアルゴリズムの重要性は既に多くの人々によって指摘されてきている．これまでもっとも多く使われ，またもっとも高速であるとされてきたアルゴリズムはForgyによって示されたＲＥＴＥアルゴリズムである[1]．しかし近年ＲＥＴＥがプロダクションシステムにとって最適であるということは必ずしも言えないという主張が指摘され始めている．その様な主張の中でもっとも有望なＲＥＴＥの代案としてMirankerによるＴＲＥＡＴアルゴリズムがある[2]．この報告ではＲＥＴＥアルゴリズムとの比較において、ＴＲＥＡＴアルゴリズムの性能，問題点等について検討する．

２．プロダクションシステム

プロダクションシステムは人間の思考過程の１モデルとして提案された，認知行動サイクルを繰り返すことをその根幹としている．

認知行動サイクルは，

（１）プロダクションメモリ（ＰＭ）と呼ばれるif-then型のルールとワーキングメモリ（ＷＭ）と呼ばれるデータとのマッチングを行う条件照合フェーズ

（２）条件が成立したルールのうちどのルールを実行するかを決定する競合解消フェーズ

（３）そのルールの実行を行う行動フェーズ

の３つのフェーズに分けられる．この内条件照合フェーズを高速に行うために考えられたのがＲＥＴＥアルゴリズムである．まずこのＲＥＴＥアルゴリズムについて簡単に説明する．

３．ＲＥＴＥアルゴリズム

ＲＥＴＥアルゴリズムではルール間で同じ照合を行っている部分をマージしてＲＥＴＥ木と呼ばれるネットワークを作る。そこにｗｍの変化をその追加、削除にともない＋，－のトークンとして流すことにより条件照合を行うアルゴリズムである。トークンに対し、まずそのルール中の１条件要素内だけで判定できる条件要素内照合(intra-condition test)が行われる。これに成功すると、そのトークンはαメモリと呼ばれるネットワーク内メモリに蓄えられる。次にそのトークンに対し同じルール中の他の条件要素との間で条件要素間照合(inter-condition test)が行われる。この照合に成功したトークンの組はβメモリと呼ばれるネットワーク内メモリに蓄えられる。各照合はこのαメモリ、βメモリを活用して一度行った照合を極力無駄にしないように行う。

４．ＴＲＥＡＴアルゴリズムの概要

ＴＲＥＡＴアルゴリズムは単純に言えばＲＥＴＥからβメモリとルール間のマージを取り去ったものと言える。これらの二つのテクニックには次のような問題点があるからである。

まずβメモリを用いた場合の問題点としてそのメンテナンスにかなりのオーバーヘッドが必要なことがあげられる。βメモリの役目は部分的なマッチングの結果を取っておくことに有るのだが、実際にはβメモリによるマッチングの保存は余り使われず、かえって無駄な情報を保存していることの方が多い。特にWMの削除の場合、いちいちマイナストークンとしてβメモリをメンテナンスしなければならないのが大きな無駄となっている。

また、もう一つの問題点として動的に必要なメモリが非常に多量となる場合があることがあげられる。βメモリはαメモリの組合せを記憶するものであるから、αメモリの平均的な量がｎで抑えられるときでも条件要素間の照合条件が厳しくない場合にはルール中の条件要素数をｍとしてｎ^{m}に比例する量のβメモリが必要となる可能性がある。

次にルールのマージについてであるがまず、この手法は手間が掛かる割に余り効果をあげないことが報告されている。実際ある種のエキスパートシステムではマージを行った場合と行わない場合とでほとんど差がでない。このルールのマージはルールに冗長性があるときに効果があるものだが，ＯＰＳ５のような比較的記述力のないプロダクションシステムの場合に比べ、ＯＰＳ８３や我々のＡＲＣＨ２のように手続き型言語を使用できる場合、ルールの冗長度は減るものと思われる。また条件要素間照合の順序がコンパイル時に静的に決定されてしまうことも欠点の一つである。

ＴＲＥＡＴアルゴリズムではこの二つをＲＥＴＥアルゴリズムから取り去り、代わりに

（１）ルールのアクティブ性

（２）照合順序の動的な変更

の二つを取り入れている。ルールがアクティブであるとはそのルールの全ての肯定条件要素のαメモリが空

Comparison of Two Maching Algotithms,
RETE and TREAT
Hiroto INABA, Akira NAKAMURA
TOSHIBA CORPORATION

でない時をいう。もしルールがアクティブでなかったらそれ以後の照合は行わずに済ますことが出来る。ＲＥＴＥアルゴリズムではたとえばルールの最後の条件要素にマッチするワーキングメモリがない時でも、途中のβメモリを更新するため照合を行う必要がある。照合順序の動的な変更とは条件要素間照合の順序を例えばαメモリの少ない順にする等、実行中に最適化することをいう。これはＴＲＥＡＴアルゴリズムにおいてはルールのマージを行わないため可能なことで、ＲＥＴＥアルゴリズムでは不可能である。

５．ＴＲＥＡＴアルゴリズムの問題点

ＴＲＥＡＴは多くのプロダクションシステムにおいて有効であるがいくつかの問題点も存在する。

まず全くルール間のマージを行わないことによる冗長なαメモリの存在がある。βメモリの節約分で補えるとはいえ、やはり動的なメモリを多く使うことになるし、メンテナンスの手間もそれだけかかることになる。

もう一つの問題点として否定条件要素が存在するときに、高速化を図ることが困難であるということがあげられる。ForgyによるオリジナルのＲＥＴＥアルゴリズムと比較すれば、それほどの差はつかない。しかしＲＥＴＥアルゴリズムの場合βメモリ中に情報を取っておくことにより否定条件要素の処理に高速化が図れたのが、ＴＲＥＡＴアルゴリズムではβメモリが存在しないのでこの手法は使えない。

６．改良ＴＲＥＡＴアルゴリズム

ＴＲＥＡＴアルゴリズムとＲＥＴＥアルゴリズムを各要素に分解すると次のようになる。

ＲＥＴＥ＝αメモリ＋βメモリ＋ルールのマージ

ＴＲＥＡＴ＝αメモリ＋ルールのアクティブ性
＋動的照合順序

我々は上に述べたようなＴＲＥＡＴアルゴリズムの問題点に対する改良として、ＲＥＴＥアルゴリズムを組み合わせた次のようなアルゴリズムを現在我々の開発したプロダクションシステムＡＲＣＨ－２に実装中である。

αメモリ＋ルールのアクティブ性＋ルールのマージルールのマージを行うことにより冗長なαメモリを減らすことが可能となり、ＴＲＥＡＴアルゴリズムの問題点の一つは解決される。ただしルールのアクティブ性を用いるためには2input-nodeのマージは行えない。また動的に照合順序を変更することはＡＲＣＨ－２のようにルールを直接手続きにコンパイルして実行する方式では実現が困難であり、取り入れられない。

ＴＲＥＡＴアルゴリズムのもう一つの弱点である否定条件要素の処理の高速化については最良の方式と言えるものはまだ見つかっていない。ｗｍｅの追加に対してはＴＲＥＡＴアルゴリズムの方針であるconflict set support、即ち直接に競合集合をメンテナンスすることが可能であるが、削除に関してはオリジナルのＴＲＥＡＴと同様に、インスタンシエーションを再構成する方法を取るしかないようである。従って、ＴＲＥＡＴアルゴリズムを採用した場合でも、否定条件要素にマッチするｗｍｅが大きく変化するような場合、余り速度の改善がなされない。しかし一般にその様なプロダクションシステムは多くないと思われ、ほとんどの場合、ＴＲＥＡＴはＲＥＴＥを速度、メモリ効率の両者において上回ることが予想される。

７．終わりに

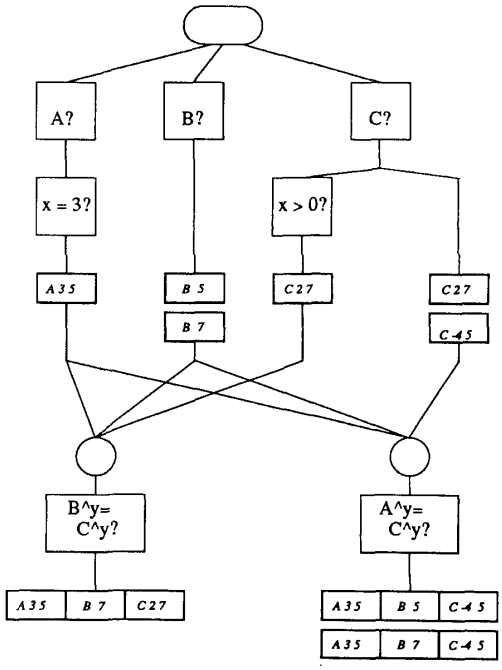
プロダクションシステムのための新しいマッチングアルゴリズムＴＲＥＡＴに対し、そのＲＥＴＥアルゴリズムに対する利点とそれ自身の持つ問題点、及びその一部の解決案を示した。今後，ＡＲＣＨ－２においてこの改良アルゴリズムについてその速度、動的メモリ使用量、コードサイズ等の性能を順次評価する予定である。

参考文献

[1]Forgy, C. L. : RETE:A Fast Algorithm for Many Pattern/Many Object Pattern Match Probrem, Artificial Inteligence, Vol.19, pp.17-37

[2]Miranker, D. P. : TREAT:A Better Match Algorithm for AI Production System, AAAI'87 pp.42-47

[3]中村　明　：　プロダクションシステムＡＲＣＨ－２の処理系概要と性能評価，第一回人工知能学会全国大会,pp.193-196



改良ＴＲＥＡＴのマッチングフロー