1R−3　RETEアルゴリズムの高速化

稲葉 浩人

(株)東芝

1.はじめに

前向きプロダクションシステムの高速化において条件照合が最も重要であることはほとんど常識となっている。条件照合のためのマッチングアルゴリズムとして最も有名なものとしてForgyによるRETEアルゴリズム[F82]があるが、近年このRETEの欠点が指摘されてきている。特にMirankerによりRETEにかわるアルゴリズムとしてとしてTREATアルゴリズム[M87], lazyマッチングアルゴリズム[M90]等が提案されてきた。しかし,これらのアルゴリズムは常にRETEに対し有利であるとはいいがたい。

本稿では条件照合アルゴリズムとしてTREATアルゴリズムやlazyマッチングアルゴリズムのようにRETEに対して根本的な変更はせず、RETEの枠の中で高速化をはかった改良RETEアルゴリズムを提案する。このアルゴリズムは、TREAT等と比較してRETEの性格に近い高速アルゴリズムであり、既存のRETEアルゴリズムを意識して作られた多くのアプリケーションを修正する必要なく高速化することができる。

2.TREATアルゴリズムとlazyマッチングアルゴリズム

まず、両アルゴリズムの特徴にふれ、そこからなぜRETEアルゴリズムを基本とする照合アルゴリズムを提案するかについて説明する。

TREATアルゴリズムはRETEアルゴリズムと比較して、プロダクションシステムの照合するデータの変化が激しいときに有利になるアルゴリズムである。特徴は、ルールのネットワーク化はほとんど行わないことと、αメモリは使用するが,βメモリは使用しないことにある。しかし、コンパイル時のコードサイズがRETEに比較して数倍になること、またβメモリの効果を全く用いていない等の欠点が挙げられる。

Lazyマッチングアルゴリズムは RETE、TREATの両アルゴリズムが共にすべてのルールインスタンスを列挙する点を指摘し、それらを列挙しないアルゴリズムとして提案された。インスタンスの列挙は大規模データに対する場合にコストが大きい為、そのようなデータを扱うプロダクションシステムにおいて有効である。特徴として、競合解消戦略をインスタンスの探索方式に組入れていることと、一回の認知行動サイクルで作成されるルールインスタンスはたかだか一つであり、作成されれば必ず発火されることが挙げられる。しかし、競合解消戦略が固定されてしまうのはあまりにも大きな欠点である。

TREAT/ Lazyマッチングではなく RETEアルゴリズムを基本とする照合アルゴリズムを提案する理由は、両アルゴリズム共に全てのアプリケーションについて効果があるわけではないという点にある。すなわち両アルゴリズム共にかなり重大な欠点を持っているため、その欠点にふれるようなアプリケーションではかえって非常に低速となったり、あるいは修正なしでは実行が不可能になってしまう。

ここで提案する改良RETEアルゴリズムでも、全てのアプリケーションについて必ず速度が向上するわけではない。しかし既存のRETEを用いたプロダクションシステムのために作成されたアプリケーションは修正なしで実行が可能であり、もし低速となったとしても比較的その率は小さい。

3.改良RETEアルゴリズム

ここで提案する改良RETEアルゴリズムは3つの改良点からなる。TREAT、Lazyマッチングそれぞれのアイデアを用いた改良と、独自にRETEの欠点を補う改良である。

TREATから取りいれる改良点は冗長なβメモリを使用しない事である。TREATでは全くβメモリを使用しないが、ここでは一部の冗長なβメモリを不使用にする。

Lazyマッチングから取りいれる改良点は,ルールインスタンスの生成制御である。Lazyマッチングではたかだか一つのインスタンスしか生成しないが、ここでは一部のルールのインスタンスの生成を抑制する。

その他の改良点は、RETEアルゴリズムで最もコストのかかるβメモリのメンテナンスの高速化である。これはβメモリのチェイニングを用いて行う。

冗長βメモリの不使用

RETEの枠の中でどのβメモリを不使用にできるかが問題となる。もしあるβメモリが簡単に再生成できるならば、あえてメモリとして記憶する必要はない。ここでは1つ上にバインドテストが存在しないようなANDノードの左にあるβメモリを不使用にする事とした。このようなβメモリはその上のノードのαメモリとβメモリの単なる組み合せであるからである。

ルールインスタンスの生成制御

ここでもまたどのようなルールインスタンスの生成を抑制するかが問題である。実際、競合解消戦略を全く仮定せずにルールインスタンスの生成を抑制することは不可能である。

競合解消戦略で最初にルールの優先度を考慮することを仮定する。この場合優先度の高いルールのインスタンスがすでに生成されているとき優先度の低いルールのインスタンスを生成しても、その回の認知行動サイクルでは決して発火されない。よってそのようなルールインスタンスの生成は抑制できる。

これを実装するためにトークンバッファと呼ぶものをRETEネットワーク上に設けた(図1)。これはRETEネットワークの1インプットノードからなる部分と2インプットノードからなる部分との間に置かれ、そこを通るトークンを

Acceleration Methods for RETE Match Algorithm
Hiroto Inaba
TOSHIBA Corporation

一時的に溜めておく。これによって優先度の高いルールに対応する2インプットノードから先にトークンを流し、ルールインスタンスが生成された時点でトークンを流す事を打ち切って優先度の低いルールインスタンスの生成を抑制することが可能となった。

βメモリメンテナンスの高速化

RETEアルゴリズムの最大の欠点はワーキングメモリを削除するときに関連する全てのβメモリから不要となった要素をサーチしなくてはならず、βメモリのメンテナンスコストが大きい点にある。これがTREATにおいてβメモリは使用しないよう決定した大きな原因である。

すなわちβメモリのメンテナンスの高速化がRETEを高速化するために最も必要なことである。ここではそのためにβメモリの構造化を行うこととした。

これまでのRETEにおけるβメモリは通常線形な構造で実現され、また2つのβメモリ要素の関係を示す情報は保持されなかった。そのため削除する要素ををサーチする際にそれぞれのβメモリで先頭からリニアサーチを行っていた。

これを図2のようにあるβメモリにおいて、その中の1要素が与えられた時に、そのすぐ下のβメモリ中の対応する要素が容易に検索できるようにチェイニングを行う。こうすることによってワーキングメモリを削除したときに不要となるβメモリの要素を効率的にサーチでき、βメモリのメンテナンスの高速化がはかられる。

4. 終りに

プロダクションシステムの高速化手法を検討した。RETEアルゴリズムを基本とし、TREAT、Lazyマッチングのアイデアを取りいれて冗長βメモリの不使用とルールインスタンスの生成抑制という改良を行い、またβメモリのメンテナンスを効率化するためにβメモリのチェイニングを行うこととした。

今後、このアルゴリズムを実装したプロダクションシステムを作成、評価することによって、これらの高速化の効果を検証していきたい。

参考文献

[F82] Forgy, C. L., RETE: A Fast Algorithm for Many Patern/Many Object Match Problem, Artificial Intelegence Vol.19, pp.17-37, 1982

[M87] Miranker, D., TREAT: A Better Match Algorithm for AI Production Systems, Proceedings of the 1987 National Conference on Artificial Intelligence, pp.42-47, 1987

[M90] Miranker, D. et. al., On the Performance of Lazy Matchng in Production Systems, Proceedings Eighth National Conference on Artificial Intelligence, pp.685-692, 1990

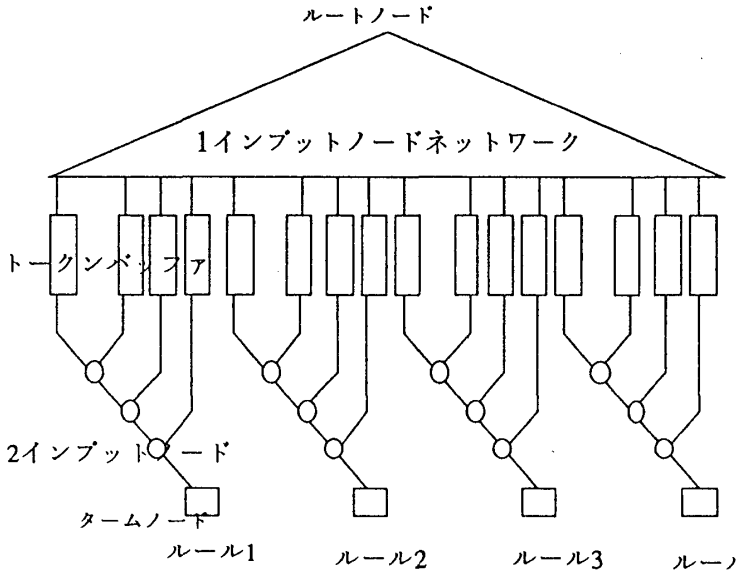

図1.トークンバッファ方式

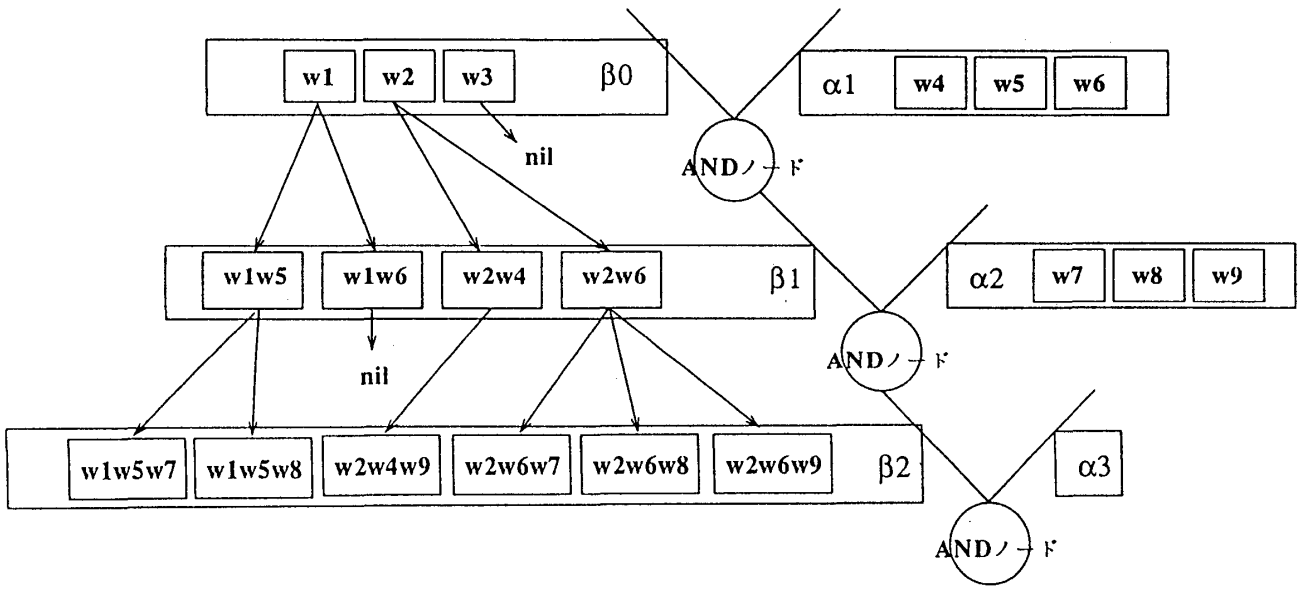

図2.βメモリのチェイニング方式