5J-5

Ｒｅｔｅ方式におけるオブジェクト削除トークン処理の高速化

潮田　百合子　山之内　徹　渡辺　正信
日本電気㈱Ｃ＆Ｃシステム研究所

１．　はじめに

競合解消型ルールセットの高速処理に対しては，現在Ｒｅｔｅ方式[1]が有効だとされているが，トークン処理部分に関しては冗長があり，改良の余地が残されている[2]．

本方式は，トークンをネットワークに流すことなくオブジェクト削除トークン処理を行い，ルールの条件節を評価する回数を減らす．

近年，実用化システムが構築されるにつれ，ルールの条件節自体の処理が複雑になってきている．本方式はこのような場合に適した方式であると考える．

２．　基本的な考え方

従来型Ｒｅｔｅ方式では，オブジェクト生成トークン（＋トークン）と削除トークン（－トークン）の２種類をネットワークに流し，ネットワークの更新を行なう．

本方式では，＋トークンが流れる際，テストが成功してトークンが通過したメモリ，および競合集合を対象トークンに含まれる各オブジェクトの属性kept-atに保持させ［ノード通過情報］，オブジェクト削除時はそのノード通過情報を参照して，－トークンを流すことなく，メモリと競合集合から直接にそのオブジェクトが含まれているトークンを削除する（図１，図２参照）．［以降，＋トークンＭを流す処理を＋Ｍ，オブジェクトＮをネットワークから直接削除する処理を－Ｎと略す］

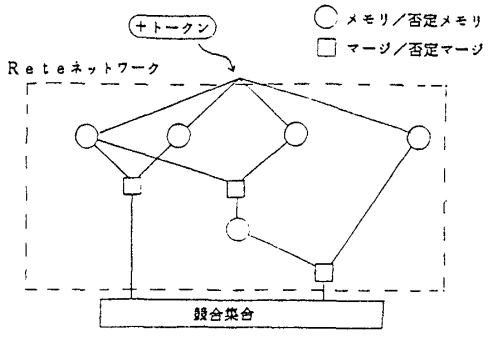


図１．Ｒｅｔｅネットワーク

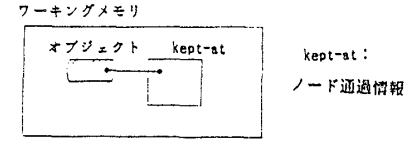


図２．ワーキングメモリ

文献[2]では，オブジェクトごとにそのオブジェクトの情報を保持したノードの位置をすべて記憶する方式は，生成する情報量が多く効率が悪い，必要な記憶容量が膨大との理由で採用していない．しかし，[2]の方式においてオブジェクトに関する情報は，候補，インタ，merge，終端の４種類の各ノードで保持されるのに対して，本方式では，メモリと競合集合（ルールセットに１つ）に限定される．メモリは，１ルール内で複数のオブジェクトが参照される場合に作られる．また，複数のルールで共通に利用できるテストがある場合には対応するメモリも共通に用いられ，メモリ数は増加しない．本方式は，記憶を持つノードを限定することにより，処理量が爆発するのを防いでいる．また，各々のメモリと競合集合に情報をコピーして持つのではなく，ポインタで渡し，必要な記憶容量を低減化している．

３．　否定条件節の高速処理

ここでは，２節で述べた方式を基に，否定条件節 "THERE-DOES-NOT-EXIST ?X"［以下の条件群を満たすオブジェクトＸがワーキングメモリに存在しないならば］の処理を高速化する方式を説明する．否定条件節は通常の条件節とはトークンの流れ方が異なるため，処理に工夫が必要となる．そこで，従来の方式を改良した方式を提案する．

文献[3]で発表した従来の方式は，従来型Ｒｅｔｅ方式を継承していた．

すなわち，否定条件節では，

①もともとその否定条件節を含むルールが成功して，次に流れてきた＋トークンがそのテストに成功した場合，条件節を満足するオブジェクトが付け足されたことがわかる．結果として否定条件節は失敗し，ネットワークからその否定条件節の成功情報を削除するため，－トークンを流す．

②もともと条件節を満足するオブジェクトが存在し，その否定条件節を含むルールは失敗していたが，オブジェクト削除（－トークン）によってテストを満足するオブジェクトがワーキングメモリに存在しなくなった場合，結果として否定条件節は成功し，ネットワークにその否定条件節の成功情報を付け足すため，＋トークンを流す．

このように従来方式では，トークン処理を＋→－，－→＋に変化させ，すべてのトークンをネットワークに流すことにより，実行していた．

A Fast Way to Process Minus Token in Rete Algorythm
Yuriko USHIODA , Toru YAMANOUCHI , Masanobu WATANABE
NEC Corp. C&C Systems Res. Lab.

一方，本方式では，ノード通過情報を用いて，－トークンを流さずに処理を行なう．

否定条件節で，その下流のノード通過情報を保持して利用することにより，－→＋，＋→－のトークン処理は，ネットワークからの直接削除処理中にトークンを流す処理に切り替える，あるいはその逆の操作に置き換えられる．

処理の詳細は，単独オブジェクト否定条件節の説明を図３および５で，複数オブジェクト否定条件節の説明を図４および６で行なう．

４．　アクション部のトークン処理の効率化

ワーキングメモリからのオブジェクト削除に関する性質を活かして，下記の問題点の改良を行なう．

①ルールのアクション部では，不必要なトークンを流すことがある．例えば，オブジェクトを生成し，スロットを追加する場合では，＋→－→＋の処理が行なわれる．実際に必要がある処理は，＋が１回のみである．

②オブジェクトａが生成され，ｂが削除される場合，＋ａ→－ｂとなる．ａとｂを組み合わせて成功するようなルールが存在した場合，ａがネットワークに流れる際にはｂはまだネットワークに保持されているので，そのルールは成功し競合集合に取り込まれ（ネットワークへのトークンの保持も行なわれる），ｂをネットワークから削除する時にその情報の削除を行なうことになる．この場合，－ｂ→＋ａの順序で処理を行なう方が効率がよい．

以上を考慮して，＋はアクション部の操作が終わるまでスタックに保持，－は出現した時点で処理する（スタックにおなじオブジェクトの＋情報が存在した場合にはこれを削除）．これにより，処理量を最小限に抑える．

５．　評価

ルールベース(Es1,Es2)を用いて評価実験を行なった．

	Ｅｓ１	Ｅｓ２
ルール発火回数／ルール数	20 /25	8 /12
従来Ｒｅｔｅ方式	3.2 secs	1.2 secs
本方式	1.5 secs	0.7 secs
効果	2.1 times	1.7 times

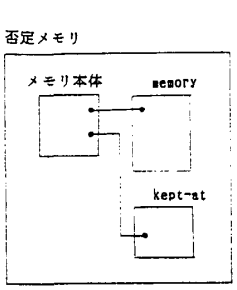
(SUN3/260　FRANZLISP)

Es1：モンキーバナナ問題．否定条件節，論理和条件節，間接領域指定などの拡張ルール言語を利用．

Es2：分類型．拡張ルール言語を使用せず．

表１．ルール実行時間の測定

従来のＲｅｔｅ方式に比べ，－トークンを流す処理，トークンの種類（＋／－）判別処理，競合集合操作処理の一部，アクション部の重複処理が削減され，メモリと競合集合通過情報の格納処理，ネットワーク直接削除処理，実行待ちスタック操作処理が増加している．比較評価実験の結果，本方式は有効と思われる．今後，確証を得るために，実用レベルルールベースを用いた評価を行う予定である。



memory：否定メモリ到達トークン群
kept-at：ノード通過情報

図３．否定メモリ

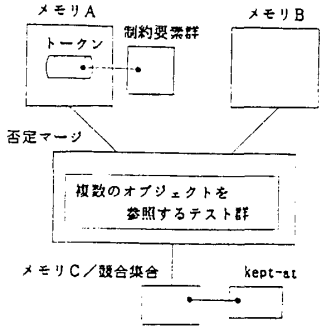


図４．否定マージ

テストを成功したトークンが否定メモリに達した場合	トークンを格納．１つめのトークンの場合，否定メモリのノード通過情報を用いて，下流から成功情報を削除．
否定メモリで保持されているオブジェクトに対する削除要求があった場合	格納トークン群から要素を削除．空になる場合，下流に成功情報を流し，トークンのオブジェクトおよび否定メモリにノード通過情報を追加していく．

参照するオブジェクトが１種類，他メモリとマージを行なわず，メモリ（否定メモリ：図３参照）を用いる．

図５．否定条件節の説明（単独否定条件節）

メモリＡにトークンが来た場合	メモリＢの各々の要素と組んでテストを行い，成功する組合せの場合はノード通過情報をＢ側オブジェクトに格納．全要素と組んだ結果，成功要素リストが空の場合，下流に成功情報を流し，トークンに含まれるオブジェクトとメモリＣに，ノード通過情報を追加していく．それ以外の場合，成功要素リストをＡ側トークンと対にしてメモリＡに格納．
メモリＡに含まれるオブジェクトの削除要求があった場合	通常のネットワーク直接削除処理を行なう．
メモリＢにトークンが来た場合	メモリＡの各々の要素と組んでトークン[A,B]を作り，テストを行う．成功した場合，Ａ側要素の拘束要素群に追加し，ノード通過情報をメモリＢに格納．１つめの拘束要素の場合，Ｃに保持されたトークン[A,B]に対するノード通過情報を用いて下流から成功情報を削除．
メモリＢに含まれるオブジェクトの削除要求があった場合	Ａの要素の拘束要素群から，削除要求のあったオブジェクトを削除．拘束条件がなくなる場合，下流に成功情報を流し，トークンに含まれるオブジェクトとメモリＣに，ノード通過情報を追加していく．

複数のオブジェクトを参照，他メモリとのマージ（否定マージ：図４参照）を用いて処理を行なう．

図６．否定条件節の説明（複数オブジェクト否定条件節）

６．　おわりに

本方式は－トークンをネットワークに流さずに処理するなど，流れるトークン数を減らす．これにより，トークンが流れる間の様々な処理（条件節の評価など）を削減することができる．よって，実用化システムでルール条件部の処理が複雑な場合においても，高速な処理が期待できる．

参考文献

［１］C.L.Forgy：Rete：A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem,Artificial Intelligence 19,1982

［２］田野，増位，坂口，船橋：知識ベースシステム構築用ツールＥＵＲＥＫＡにおける高速処理方式，情報処理学会論文誌，Vo.28,No.12,pp1255-1268(1987)

［３］山之内，潮田，渡辺：統合型ＡＩツールにおけるルールコンパイラ，人工知能学会第２回全大予稿集