# プロダクションシステムにおける効率的パタン照合のための連想 Rete ネットワーク表現†

荒屋真二†† 須崎健一†† 百原武敏††

Rete アルゴリズムはプロダクションシステムの高速化に大きく貢献したが，ルールベースの大規模化への対応，高い応答性が要求される実時間システムへの応用にはなお一層の高速化が必要である．本研究の目標は，プロダクションシステム実行時に本質的に必要なパタン照合を明らかにし，無駄な照合を回避する方法を確立することである．本論文は，ルールベースの動作パタン集合と条件パタン集合との関係に関する知識をあらかじめ抽出し，Rete ネットワークをさらに構造化した連想 Rete ネットワークを提案する．これを用いれば，Rete ネットワークのルートノードからではなく，途中のノードからトークンを流し始めても，全く同じ推論結果が得られる．これはトークンを流すべき範囲を局所化できることを意味し，無駄な照合が回避される．本文では連想 Rete ネットワークの定義，その効率的生成法，推論実行時の処理法，および従来方式との比較が述べられている．

## 1. まえがき

プロダクションシステム (PS) は，モジュラリティや可読性の良さなどの点から多くのエキスパートシステムで用いられている．また，PS の欠点である推論効率の悪さを改善した Rete アルゴリズム[1]の出現も応用範囲の拡大に貢献した．しかし，大規模知識ベースを必要とする問題解決や高い応答性の要求される実時間システムでは，なお一層の推論高速化が切実に要望されている．PS の高速化に対するアプローチは，ルールベースの構造化による方法[1)~6)]と，推論計算の並列化による方法[14]に大別できる．前者は推論に真に必要な照合を追及するものであり，後者の基礎を与える．本研究は前者の観点から PS の高速化に接近したものである．

ルールベースの構造化に関する従来研究としては，ルール条件部相互の類似性に関する知識を用いて構造化する Rete ネットワーク[1]が重要な存在である．その改良法として，類似性だけでなく排他性に関する知識も利用したもの[3]，きつい条件が先にテストされるようにしたもの[4]，推論実行のたびに局所構造を動的に最適化する方法[5]などがある．これらはすべてルール条件部の構造化である．これに対して，ルール相互の動作部と条件部の関係を利用して構造化を図るという，注目に値する研究がある[6]．これは，条件パタンや動作パタンを一つのノードで表現した一種のペトリネットで，動作パタンの定数部分と同じ定数部分をもつ条件パタンを連想リンクで結んだものである (以下，ペトリネット方式と呼ぶ)．しかし，変数部テストの共有化や変数部テスト結果の保存による同一照合の回避といった条件パタン間の構造化が行われていない．また，トークンの初期配置を求めるためにペトリネットとは別に弁別ネットも必要である．すなわち，Rete アルゴリズムがもつ効率性やシンプルさが失われている．さらに，開発効率を左右するペトリネット生成法も示されていないし，Rete アルゴリズムとの関連性も考察されていない．

そこで本論文では，第一に，Rete アルゴリズムとペトリネット方式の両者の長所を合わせもつ連想 Rete ネットワークを提案する[8),9)]．連想 Rete ネットワークは，Rete ネットワークの端末ノードに格納されている動作パタンから連想ノードへ至るリンクを付加したものである．これにより，トークンをルートノードからではなく，途中のノードから直接流すことが可能となり，照合範囲が局所化される．作業記憶 (WM) の初期値に対してはルートノードから流せばよく，そのためのネットワークを別途用意する必要はない．また，本方式の推論効率は Rete ネットワークのノード共有度の影響を受けないという利点もあり，逐次構造化法[10]と組み合わせて効果的に利用できる．本方式は推論実行前に部分的推論を行い，その結果を用いて知識の構造化を図ったものとみなすこともできる．第二に，Rete アルゴリズムを応用した，連想 Rete ネットワークの効率的生成法を提案する．これは，個々の動作パタンをトークンとして Rete ネットワークのルートノードから流し，その動作パタンの連想ノードを求

† An Associable Rete Network Representation for Efficient Pattern Matches in Production Systems by SHINJI ARAYA, KENICHI SUZAKI and TAKETOSHI MOMOHARA (Department of Communication and Computer Engineering, Faculty of Engineering, Fukuoka Institute of Technology)

†† 福岡工業大学工学部通信工学科

めるものである．付随的に，ルールベースの維持管理に有用な行き止まり動作パタン[11),12)]も検出される．この構造化処理は推論実行処理と共通点が多いため，理解が容易であると共に処理系がコンパクトになる．

以下，次章で連想 Rete ネットワークを定義し，それが推論時にどのように使用されるかを説明する．第3章では連想 Rete ネットワークの効率的生成法について述べる．第4章では，本方式の有効性を示すために，従来法と比較考察する．第5章では本論文をまとめると共に，今後の課題について述べる．なお，本稿はプロダクションシステムとして OPS5[7)] を対象とし，その文法に関する知識があることを前提に書かれている．

## 2. 連想 Rete ネットワーク

### 2.1 Rete ネットワークの概要

Rete アルゴリズムの基本的考え方は作業記憶(WM)の変化から競合集合の変化を求めるということであり[1)]，ルールベースを Rete ネットワークの形で構造化することにより無駄な照合を回避する．Rete ネットワークは次の2種類の知識（メタ知識）を利用して知識の構造化を図る[1),3)]．

1) 全ルールの条件部を構成する条件パタン間の類似性に関する知識
2) 現在の WM 要素によってルール条件部のどの範囲が支持されているかに関する知識

上記 1) はノードの共有化を図るために，2) は2入力ノードの左右に前置されるメモリに，ルートからその2入力ノードに至るパスの条件を支持する WM 要素を対応付けるために使用される．WM に追加（削除）された事実はトークンとして Rete ネットワークのルートノードから流される．各ノードでの照合に成功したトークンは次ノードに送出される．2入力ノードに到達したトークンは対応するノードメモリに追加(削除) され，条件パタン間にまたがるテストが行われる．そのテストに成功すると合成トークンが次ノードに送出される．合成トークンが端末ノードに到達すると対応するルールが合成トークンと共に競合集合に追加(削除)される．ノードでのテストに失敗したり，端末ノードに到達したときには未探索部分がなくなるまでバックトラックしてトークンが流される．

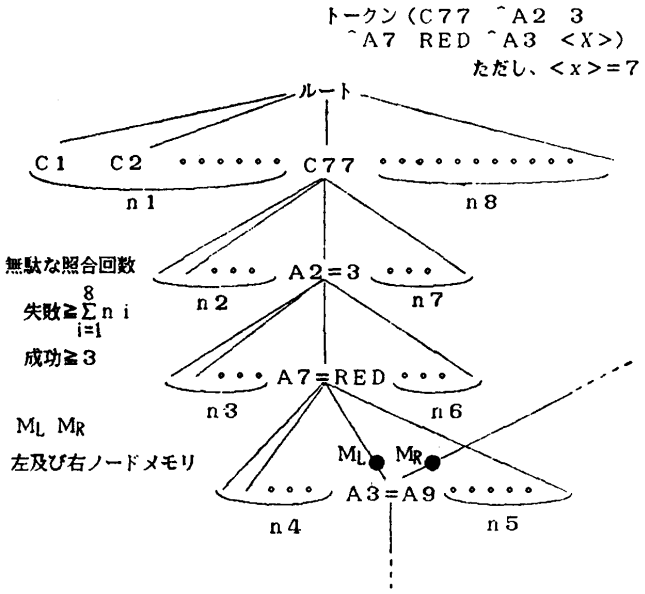


図1 Rete アルゴリズムにおける無駄な照合
Fig. 1 Ineffective pattern matches in Rete algorithm.

### 2.2 連想 Rete ネットワーク

図1は Rete ネットワークの一部を略記したものである．今，あるルールが発火し，それに含まれる動作項 (make C77 ^A2 3 ^A7 RED ^A3 〈X〉) が実行された場合を考える．また，変数〈x〉に7が束縛されているものとする．このとき，動作パタン（動作項から動作関数 make を除去したもの）のインスタンシエィション (C77 ^A2 3 ^A7 RED ^A3 7) がトークンとしてルートノードから流される．このトークンは $n1+n2+\cdots+n4$ 回以上の失敗照合と3回以上の成功照合の後，2入力ノード A3=A9 の左ノードメモリ $M_L$ に格納される．右ノードメモリ $M_R$ が空のときにはバックトラックしながら $n5+n6+\cdots+n8$ 回以上の失敗照合が行われ，トークン処理が完了する．

以上のような照合を推論実行時に行うのは明らかに無駄である．なぜならば，変数〈$x$〉の束縛値が解らなくても，つまりルールベースが与えられた段階で既に，この動作パタンのすべてのインスタンシエィションがノードメモリ $M_L$ に到達可能であることは解るからである．このような知識をあらかじめ抽出しておけば，トークンをわざわざルートノードから流さずに，直接ノードメモリに入れてやることから処理を開始できる．そうすれば，図1の場合には $n1+n2+\cdots+n8+3$ 回以上の無駄な照合をせずに済ませられる．本論文で提案する連想 Rete ネットワークは，各動作項から，トークンを流し始めるノードへのリンクを Rete

ネットワークに付加したものである．このトークンを流し始めるノードを動作パタンの連想ノードと呼ぶことにする．

連想ノードをより厳密に定義しよう．動作パタンPの連想ノードがNであるための必要十分条件は次の三つである．

1) ルートノードからノードNまでのパスに含まれるすべてのノード（ただしNは除く）でのテストにPは必ず成功する．

2) ノードNでのテストにはPは成功する可能性がある（必ず成功する場合や必ず失敗する場合を除く）．ただし，Nが端末ノードのときにはこの条件は不要である．

3) ノードNからある端末ノードに至るパスに含まれるすべてのノードでのテストにPが成功する可能性があるような端末ノードが少なくとも一つ存在する．

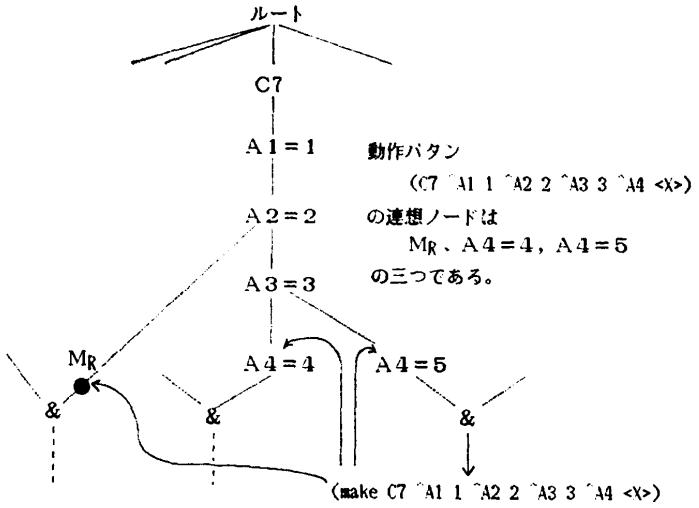


**図 2** 複数個の連想ノードをもつ例
Fig. 2 An example that has more than one association node.

ここで注意しなければならない点は，一つの動作パタンに対して連想ノードは一般に複数個存在するということである．**図2**にその1例を示す．連想 Rete ネットワークとは，すべての動作パタンに対するすべての連想ノードを求め，それらをリンク（連想リンク）で関連付けたものである．連想ノードをすべて抽出しなければ，動作パタンが支持する条件パタンを見逃してしまう可能性があるため，誤った結論に到達する可能性を残してしまう．連想ノードの効率的探索アルゴリズムは3.2節で詳述する．

Rete アルゴリズムはすべての動作パタンの連想ノードがルートノードになっているとみなすこともできる．ゆえに，連想 Rete ネットワークは推論効率化に有効な連想ができるように Rete ネットワークをさらに組織化したものであるといえる．

### 2.3 推論実行時処理

連想 Rete ネットワークを用いた推論実行時処理の概略フローを**図3**に示す．Rete アルゴリズムとの相違点は，WM に追加/削除された事実（動作パタンのインスタンシエィション）がルートノードからではなく，対応する連想ノードから直接流されるということである．また，動作パタンが連想ノードをもたない場合には，トークンを全然流す必要がない．これは行き止まり動作パタンと呼ばれており[11]，後の推論の流れに全く影響を与えないので実行しても意味がない．連想ノードが端末ノードとなる場合もあり（3.2節），このときはトークンを流すことなく競合集合の変化を直

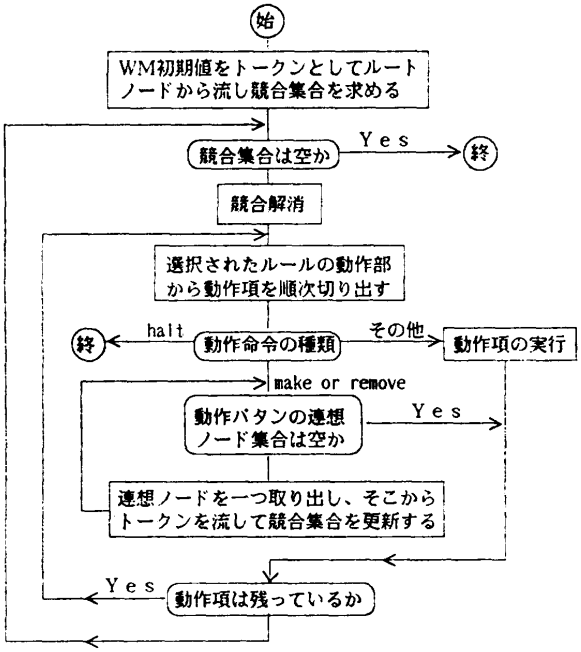


**図 3** 推論実行処理の概略フロー
Fig. 3 Processing flow during inference execution.

接求めることができる．このように，連想ノードの導入は Rete アルゴリズムで行われていた無駄な照合の回避に有効である．

ノード共有度と推論実行効率の関係についての考察は連想ノード導入の別な有用性を浮き彫りにする．**図4**は同一ルールベースを異なる Rete ネットワーク生成法によって構造化したものであり，対応する一部分だけを切り出して示している．(a)は OPS5 の Com-

mon Lisp 版により逐次構造化されたもの，(b)は文献 10) の逐次構造化法によるもの，(c)は一括構造化によりノード共有の最大化を図ったものに相当する．(a)，(b)，(c)は，それぞれテストの順序が異なっているため共有ノードの数が異なっている．図中に示されたトークンが流された場合，Rete アルゴリズムでは，(a)，(b)，(c)とも最終的には2重丸で示された右側のノードに到達するが，それまでに行われる照合回数が違うので推論効率は異なったものとなる．しかし，連想 Rete ネットワーク方式では，いずれの場合も右側の2重丸ノードから直接トークンが流されるので，そこに至るまでの照合は全く不要となる．このことは，Rete アルゴリズムの推論効率がノード共有度の影響を受けるのに対し，提案方式ではその影響を受けないことを意味する．つまり，最適性を若干犠牲にしてコンパイル時間の短縮化を図る逐次構造化法[10]を用いても，推論実行時の効率が悪化しないことになる．ゆえに，逐次構造化法と連想 Rete ネットワーク方式の組み合わせは有効となる．

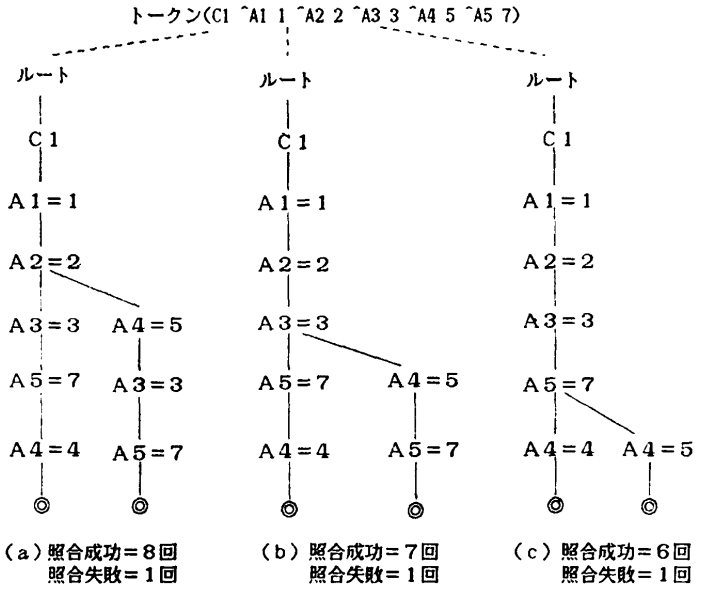


**図 4** Rete ネットワークのノード共有度と照合効率
Fig. 4 Influence of node sharing rate on match efficiency.

提案方式の効率は，動作パタンに含まれる変数の割合が増加するにつれて次第に低下するという性質がある．なぜならば，動作パタンに変数が増えてくると必ず成功あるいは必ず失敗するという判断が推論実行前に下せなくなるからである．そのため連想ノードは Rete ネットワークの上流の方に位置するようになる．これは 3.2 節で述べる連想ノードの求め方を見れば明らかである．ただし，動作パタンに含まれる変数が多い問題に対しても，Rete アルゴリズムよりも無駄な照合が増えることは決してない．

連想 Rete ネットワーク方式は推論時における無駄な照合を完全に除去できるわけではない．その1例を**図 5** (a)に示す．つまり，動作パタンに変数が含まれており，それに対するテストの後に，成功するテストが続くような場合である．これへの対処法は種々考えられる．例えば(b)のようにテストの順序を変えてやればよい．しかし，このような構造変更処理はこのパスを通過する可能性のあるすべての動作パタンを考慮しなければならないので，処理が繁雑にならないように工夫する必要がある．この問題点の解決は今後に残されている．

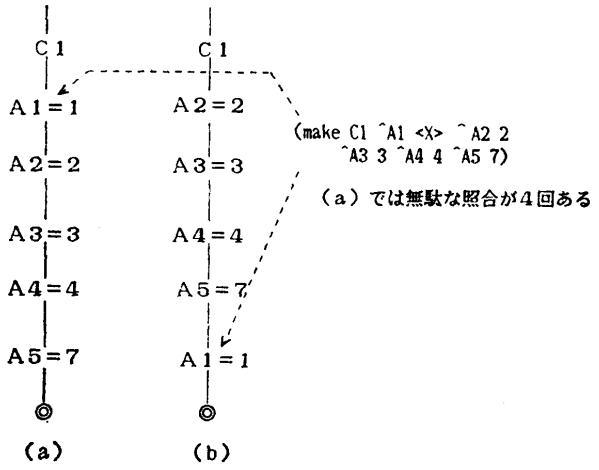


**図 5** 連想 Rete ネットワークにおける無駄な照合の例
Fig. 5 An example of ineffective matches in associable Rete network.

## 3. ネットワーク生成法

### 3.1 基本的考え方

連想 Rete ネットワークを生成するためには，前述の連想ノードを効率よく求めることが重要である．なぜならば，推論実行に先立って行われるこの種の知識の組織化はルールの追加削除にともない頻繁に行われるため，ルールベースの開発効率を大きく左右するからである[8]．そのためには，最も計算負荷の大きい動

作パタン集合と条件パタン集合との照合の効率化が効果的である．そこで，条件パタン集合を Rete ネットワークの形で構造化しておき，動作パタンをトークンとして Rete ネットワークのルートノードから流してやれば，その過程で連想ノードを効率的に求めることができる．ただし，Rete アルゴリズムと異なる点は，流されるトークンの中に，一般には変数が含まれるということである．ゆえに，各ノードでのテスト結果が，成功か失敗かのどちらかに確定されるとは限らず，成功する可能性があるといった新しい状況も考慮しなければならなくなる．

### 3.2 生成アルゴリズム

連想 Rete ネットワーク生成の概略フローを図6に示す．最初に Rete ネットワークを作成するが，この方法は一括構造化法でも逐次構造化法でも何でも構わない[8),10)]．OPS5 の動作項では種々の動作命令が記述できるが，WM に事実の追加/削除を行うもの (make, remove) だけに対して連想ノードを求める．modify は make と remove に分けて考えてやればよい．

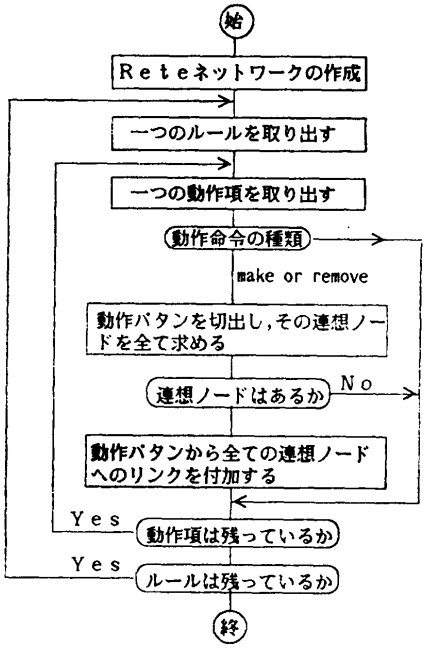


図 6　連想 Rete ネットワーク生成の概略フロー
Fig. 6　Processing flow for generating associable Rete network.

図6における連想ノードを求める部分を詳細化したのが図7である．最初に連想ノード集合と連想ノード候補を Nil にリセットしてから，動作パタンをトークンとして Rete ネットワークのルートノードから流してやる．その過程で連想ノード候補（最終的にはその動作パタンの連想ノードとなる）はセットあるいはリセット（図7の「連想ノード候補を削除」に対応）され，端末ノードに到達したときに連想ノード集合に追加される．1入力ノードおよび2入力ノードにおけるテストの結果は前述のように3種類に分けられ，それらの意味は次のとおりである．

①成功：動作パタンのすべてのインスタンシエィションに対して必ず成功する．

②失敗：動作パタンのすべてのインスタンシエィションに対して必ず失敗する．

③成功可能：動作パタンのインスタンシエィションの中に成功する可能性のあるものが存在する．

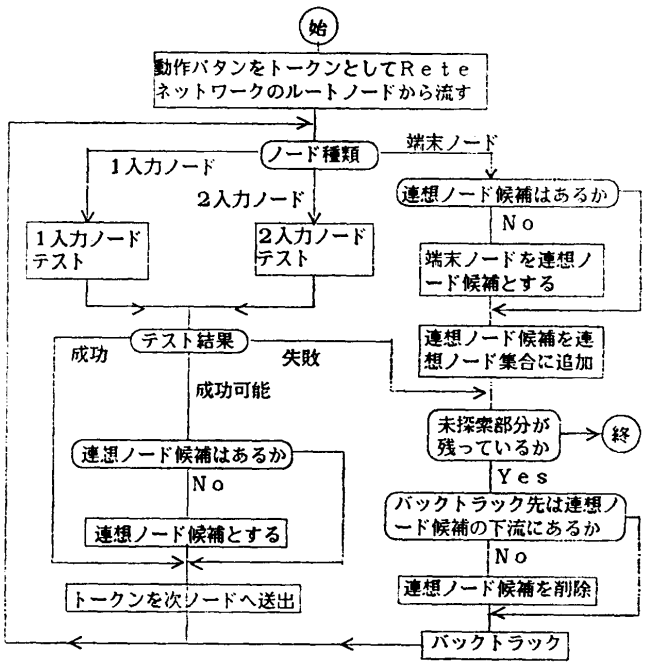


図 7　連想ノード探索のための処理フロー
Fig. 7　Processing flow for searching association nodes.

Rete ネットワークにおける各ノードでのテスト方法は次のようになる．

［1入力ノードでのテスト方法］

1)　トークン側に対応する属性がある場合

i）条件パタン内同一変数テスト（=, ≦, ≧, ≠）の場合

①対応する二つの属性値が共に変数の場合——変数名が同じならば=, ≦, ≧は成功, ≠は失敗

とし，それ以外は成功可能とする

②一方が変数で他方が定数の場合——数値定数ならば成功可能，それ以外ならば＝，≠は成功可能で≦，≧は失敗とする

③対応する二つの属性が共に定数の場合——テスト条件を満足すれば成功，それ以外は失敗とする（もちろんデータ型が異なれば失敗）

ⅱ）条件パタン内同一変数テスト（>，<）の場合

①対応する二つの属性値が共に変数の場合——変数名が同じならば失敗，それ以外は成功可能とする

②一方が変数で他方が定数の場合——数値定数ならば成功可能，それ以外は失敗とする

③対応する二つの属性が共に定数の場合——テスト条件を満足すれば成功，それ以外は失敗とする（もちろんデータ型が異なれば失敗）

ⅲ）定数との比較テストの場合

①対応する属性値が変数の場合——成功可能とする

②対応する属性値が定数の場合——テスト条件を満足すれば成功，それ以外は失敗とする（もちろんデータ型が異なれば失敗）

2) トークン側に対応する属性がない場合——失敗とする

[2入力ノードでのテスト方法]

1) 条件パタン間の変数テストがある場合

①対応する属性がある場合——成功可能とする

②対応する属性がない場合——失敗とする

2) 条件パタン間の変数テストがない場合——成功可能とする

1入力ノードのテスト結果は，動作パタン側の対応する属性値が定数のときに確定するのは明らかだろう．注意すべき点は，動作パタン側の対応する属性値が変数のときでも確定する場合がかなりあるということである．これはペトリネット方式[6]との大きな相違点の一つである．2入力ノードでのテスト方法を見れば解るように，Rete ネットワークのノードメモリは使用されず，成功可能と判定されたトークンは合成トークンの形ではなく，そのまま次のノードに送出される．トークンが端末ノードに到達した時点で連想ノード候補がないということは，その端末ノードに対応するルールの条件パタンが一つだけで，かつそれがトークンによって必ず支持されることを意味する．この場合には端末ノード自体が連想ノードとなり，推論時に照合は不要となる．

### 3.3 付随して得られる情報

ある動作パタンは連想ノードを一つも持たない場合がある．これは，その動作パタンがどの条件パタンをも支持する可能性がないことを意味する．このような動作パタンが WM に追加，削除されても推論の流れに全く影響を及ぼさないので無意味であり，行き止まり動作パタン[11]と呼ばれている．逆に，この存在は1入力ノードでの無駄な照合を引き起こすだけでなく，2入力ノードにおいてメモリを無駄使いし，最終的には失敗に終る無駄な照合を増加させる可能性がある．ゆえに，行き止まり動作パタンの検出はルールベース管理にとって重要である．提案方式はこの行き止まり動作パタンを推論実行前に検出，警告することが可能である．また，もし推論実行時にそれがあったとしてもトークンとして流されないので，前述のような無駄な照合やノードメモリの無駄使いを一切行わない．

## 4. 従来法との比較

### 4.1 Rete ネットワーク方式との比較

Rete アルゴリズムとの比較に関しては前章で説明したので，ここではそのまとめにとどめる．

1) 提案方式はトークンをルートノードからではなく連想ノードから流すため，必ず成功あるいは失敗するような照合の大部分を回避できる．

2) 提案方式の推論効率はノード共有度の影響を受けず，共有最大化を図った場合と同じ効率が得られる．

3) WM の初期値は両方式ともルートノードから流されるので，それに必要な照合の回数は同じである．この照合回数はノード共有度の影響を受ける．

4) 上記2)より提案方式は Rete ネットワークの作成に時間効率の良い逐次構造化法を利用しても推論効率が低下しない．

5) 提案方式は推論実行前に行き止まり動作パタンを検出できるので，ルールベースの維持管理に有効である．

6) 提案方式は推論実行時に行き止まり動作パタンが存在していても，それに起因する無駄な照合やメモリの使用を避けられる．

7) 提案方式は Rete ネットワークを構成してから連想ノードを求めるので，そのための処理時間並びに格納用メモリが余分に必要となる．

8) 提案方式は動作パタンに含まれる変数の割合が

増加するにつれて推論効率は低下する．ただし，Rete アルゴリズムよりも悪くなることはない．

次に，Rete アルゴリズムを改良した排他方式[3]との比較を行う．排他方式は条件パタン間の類似性だけでなく，排他性もあらかじめ抽出して Rete ネットワークをさらに構造化したものである．これにより，あるノードがいくつかに分岐している場合，分岐後のノードが互いに排他関係にあるならば，そのうちの一つのノードでのテストに成功すれば他のノードでのテストは必ず失敗するので無駄なバックトラッキングを行わなくて済む．ただし，成功ノードが見つかるまでの照合は避けられないため，成功可能性の高いノードがなるべく早くテストされるように構造化される．これに対して提案方式では，必ず失敗するようなノードには始めからトークンが流れないのでより効率的である．また，排他方式では失敗照合回数だけが減少するのに対し，提案方式では成功照合回数も減少するという利点がある．

### 4.2 ペトリネット方式との比較

文献 6) の方式は動作パタンと条件パタンとの関連性を利用してルールベースの構造化を図っているという点において本研究と類似している．ゆえに両方式の比較考察は不可欠であろう．この方式では，条件パタンや動作パタンを一つのノード（プレース）で表現し，条件パタンと動作パタンの定数部分が同じもの同士をリンクで結んだ一種のペトリネットである（ゆえに，ペトリネット方式と呼ぶことにする）．WM 初期値に対するトークンの各ノードへの初期配置は別に作られた弁別ネットが使用される．以下に主な相違点を示す．

1) ペトリネットとは別に弁別ネットも必要なので，メモリ効率が悪くなる．また，WM 要素に対する処理が初期要素と推論時に生成される要素で異なるので処理系が複雑になる．

2) 条件パタン間にまたがる部分は単にトランジションとしてモデル化されているのでほとんど構造化されていないといってよい．

3) Rete ネットワークにおける 2 入力ノードのノードメモリに相当するものがない．そのため条件パタン間にまたがる変数テストの結果は一部きり保存されないので無駄な同一照合が繰り返される．

4) Rete ネットワークにおける 1 入力ノードおよび 2 入力ノードにおけるテストの共有化が図られていないため，無駄な同一照合が繰り返される．

5) 単に定数部分が等しい条件パタンと動作パタンをリンクで結んでいる．条件パタンや動作パタンに変数が含まれる場合でも照合結果があらかじめ確定する場合があることを考慮に入れていない．これは無駄な照合を引き起こす要因となる．

6) ペトリネットの作成法（構造化アルゴリズム）が示されていない．もし，その作成に Rete ネットワークを利用していないとすれば，処理時間は大きくなりルールベースの開発効率の点で問題となる．

## 5. あとがき

プロダクションシステム高速実行のための連想 Rete ネットワークと，その効率的生成を提案した．また，従来法[1),3),6)]との比較考察により，提案方式の有効性を示した．本方式では，構造化のための計算量，ならびに連想ノード格納のための記憶容量が増加するが，これは Rete ネットワークと純粋プロダクションシステムとの関係と同じである．本文中で例示したように，本方式にはまだ無駄な照合が残っており，その完全除去は今後の課題である．また，本方式が逐次構造化[10]や逐次メンテナンス[11]となじみやすいことを本文中で示唆したが，その具体的統合方法も検討する必要がある．本研究はルール合成による新ルールの自動生成[13]などの学習機能を，より実用的なレベルで実現する場合の土台になり得ると考えており，その方向へ発展させてゆく所存である．

**謝辞** 本研究の一部は福岡工業大学附属エレクトロニクス研究所の助成によるものであり，ここに心より謝意を表する．

## 参考文献

1) Forgy, C.L.: Rete: A Fast Algorithm for Many Pattern/Many Object Pattern Match Problem, *Artif. Intell.*, Vol. 19, pp. 17-37 (1982).
2) McDermott, J., Newell, A. and Moore, J.: The Efficiency of Certain Production Implementations, in Watermann, D.A. and Hayes-Roth, F. (eds.), *Pattern Directed Inference Systems*, pp. 155-176, Academic Press, New York (1978).
3) 荒屋ほか：プロダクションシステムのための高速パターン照合アルゴリズム，情報処理学会論文誌，Vol. 28, No. 7, pp. 768-775 (1987).
4) 田野ほか：知識処理ソフトウエア EUREKA におけるルールネットワークの効率化方式，第 32 回情報処理学会全国大会論文集，pp. 1517-1518

(1986).
5) 田野ほか：推論高速化のための弁別ネットワークの動的変形法，第 33 回情報処理学会全国大会論文集，pp. 1417-1418 (1986).
6) 鶴田ほか：連想・選別型推論のアナロジーによるプロダクションシステムの高速実行方式，情報処理学会論文誌，Vol. 26, No. 4, pp. 696-705 (1985).
7) Forgy, C. L.: OPS5 User's Manual, Dept. of Compt. Sci., Carnegie-Mellon Univ. (1981).
8) 荒屋：知識の構造化に関する考察，情報処理学会知識工学と人工知能研究会資料，AI-50-7 (1987).
9) 荒屋ほか：連想関係を利用した改良 Rete アルゴリズム，電気関係学会九州支部第 40 回連合大会，No. 1015 (1987).
10) 荒屋ほか：知識ベースの逐次構造化—Rete ネットワークの逐次構築法—，電子情報通信学会論文誌，Vol. J71-D, No. 6, pp. 1100-1108 (1988).
11) 荒屋ほか：ルールベースのインクリメンタルメンテナンス，電気関係学会九州支部第 40 回連合大会，No. 1016 (1987).
12) Nguyen, T. A. et al.: Checking an Expert Systems Knowledge Base for Consistency and Completeness, IJCAI-85, pp. 375-378 (1985).
13) Anderson, J. R.: Knowledge Compilation, in Michalski, R. S., Carbonell, J. G. and Mitchell, T. M. (eds.), *Machine Learning*, Vol. 2, pp. 289-310, Morgan Kaufmann Pub. Inc. (1986).
14) 石田：プロダクションシステムと並列処理，情報処理，Vol. 26, No. 3, pp. 213-225 (1985).

(昭和 62 年 10 月 26 日受付)
(昭和 63 年 6 月 24 日採録)

**荒屋 真二**（正会員）

昭和 24 年生．昭和 47 年東北大学工学部通信工学科卒業．同年三菱電機(株)入社．昭和 60 年福岡工業大学通信工学科助教授．この間，昭和 58 年 4 月～9 月神戸大学工学部非常勤講師．工学博士（東京大学）．人工知能，知識工学，コンピュータミュージックの研究に従事．昭和 57 年電気学会論文賞受賞．電気学会，計測自動制御学会，電子情報通信学会，人工知能学会，IEEE 各会員．

**須崎 健一**（正会員）

昭和 23 年生．昭和 47 年日本大学理工学部電気工学科卒業．昭和 52 年東京電機大学大学院修士課程修了．昭和 47 年防衛大学校電気工学教室助手．昭和 55 年福岡工業大学通信工学科講師，現在に至る．分散処理に関する研究に従事．電子情報通信学会会員．

**百原 武敏**（正会員）

昭和 19 年生．昭和 42 年福岡工業大学電子工学科卒業．同年同大助手．昭和 59 年同大通信工学科講師．人工知能，知識工学の研究に従事．電子情報通信学会会員．