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†

荒屋真二†† 百原武敏†† 田町常夫††

純粋なプロダクションシステムではパターン照合が推論コストの大部分を占める．この照合コストを削減するために，Rete アルゴリズムはプロダクション間の類似性に関する知識と，各プロダクションが作業記憶の内容によって現在どの程度満足されているかについての知識をうまく活用している．本論文は上記2種類の知識に加えて，プロダクション間の排他性に関する知識と，照合成功確率に関する知識を利用した，より強力なパターン照合アルゴリズムを提案し，Rete アルゴリズムにはかなり無駄な照合が残されていることを明らかにする．また，OPS5 と同様の文法をもつプロダクションシステム記述言語を 16 ビットパーソナルコンピュータ上に実現し，三つのサンプルプログラムによって提案アルゴリズムの有効性を実験的に示す．

1. まえがき

プロダクションシステムは人工知能システムおよびエキスパートシステムにおける基本アーキテクチャとして多用されている．純粋なプロダクションシステムの推論効率は良くないので，それを改善するための手法がいくつか提案されてきた[1)~7)]．しかし，まだ処理時間の点から実時間制御システムや対話型システムなどへの適用には限界があり，推論の高速化は重要な研究課題となっている[6),7),12),13)]．

プロダクションシステムの推論効率化をねらった従来研究は，その着眼点により次の三つに大別できるだろう．

（1） 無駄な照合の削減による効率化[1),2),5),12),13)]

（2） コンパイラによる効率化[5)]

（3） 並列処理による効率化[3),6),7)]

これらのアプローチは相互に密接な関連をもつとともに，併用も可能である．事実，研究が最も進んでいるプロダクションシステム記述言語 OPS では，上記(1)，(2)のアイデアがインプリメントされており[3),5)]，(3)の研究が現在行われている[6),7)]．

本論文では，知識工学の立場から最も興味深く，かつ本質的な意味での効率化ともいえる上記(1)の問題を考察の対象とする．無駄な照合を減らすために提案されている具体的方法は次の2点に集約できる．

（1） 現在の照合結果を保存しておくことにより，同一の照合の繰返しを避ける．

（2） プロダクション間の類似性に関する知識を利用して，同一の照合をまとめて処理する．

上記(1)は作業記憶の要素数が多く，プロダクション1発火あたりに追加・削除される要素数が少ないほど効果が大きい．(2)は類似した条件部をもつプロダクションの数が多いほど効果的である．ここで，照合結果をどの程度の詳細度で記憶しておくかによって，さらには，知識の類似性をどの程度の詳細度で考えるかによって種々のアルゴリズムが実現できる[2)]．もちろん，詳細度が高いほど無駄な照合の回数は減少するが，必要記憶容量と知識構造化に要する前処理時間は増大する．

現在最も効率的といわれている Rete アルゴリズム[5)]も上記の考えに立脚しているが，まだ無駄な照合が行われている．例えば，ある条件要素が成立したとき，他の条件要素の中には絶対に成立しえないものが存在することがある．この種の関係はプロダクション集合が与えられれば推論実行前にわかるものであり，これを利用すれば明らかに失敗に終わる照合を避けることが可能となる．また，このような知識の排他性を利用した場合には，成功する可能性の高い照合をできる限り早い時点で行うように工夫すれば，枝刈りがより強力なものとなる．筆者らはこれらのアイデアを盛り込んだプロダクションシステム記述言語処理系を 16 ビットパーソナルコンピュータ上に実現し，推論効率が Rete アルゴリズムより向上することを確認した．

本論文では，2章において考察の対象に選んだプロダクションシステムを概説する．3章では提案するパターン照合アルゴリズムにおける基本的考え方，排他性を利用したプロダクションの構造化方法，成功する

† A Fast Pattern Matching Algorithm for Production Systems by SHINJI ARAYA, TAKETOSHI MOMOHARA and TUNEO TAMATI (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

照合をなるべく早期に行う簡便な方法について述べる．また，Rete アルゴリズムの改良法として提案されている EUREKA[12),13)] との比較を行う．次に，4章において試作した処理系の概要を述べたあと，5章において三つのサンプルプログラムを用いて提案アルゴリズムの有効性を実験的に示す．最後の章では本研究のまとめと今後の課題について述べる．

2. 対象とするプロダクションシステム

本論文で考察の対象として取り上げたプロダクションシステムは，OPS 5[4)] とほぼ同様の文法をもつ．すなわち，各プロダクションはプロダクション名，条件部，動作部からなり，さらに条件部は条件要素から，動作部は動作項から構成される．条件要素はクラス名のあとに属性名と属性値のペアが0個以上続く．属性名は記号“^”が前置されることで識別され，属性値としては定数と変数がある．変数は“<”と“>”で囲まれることで識別される．属性値の前では“>”(より大きい)や“<=”(以下)などの範囲を指定する6種類の述語と，データ型が等しいことを表す述語“<=>”を使用できる．さらに選言命題や連言命題もある制約のもとで使用できる．動作項では，make, remove, modify などを含む計15種類の動作を記述できる．これら文法に関する詳細かつ厳密な内容は文献4)を参照されたい．

作業記憶は作業記憶要素の集合である．作業記憶要素の形式は条件要素とほぼ同様である．ただし，属性値として変数，述語，選言および連言命題を使用できないこと，クラス名の前にタイムタグと生成プロダクション名が付けられることが相異している．

条件部を構成するすべての条件要素と作業記憶内の作業記憶要素のどれかとの照合が成功するとき，そのプロダクションは適用可能となる．適用可能なプロダクションの集合を競合集合と呼ぶ．ある競合解消戦略にもとづいて競合集合の中から一つのプロダクションが選択され，その動作部が実際に実行される．その結果として作業記憶の内容が更新される．プロダクションシステムはこのようなプロダクションの選択・実行を繰り返しながら（認知・行動サイクル）推論を進める．本論文で提案するパターン照合アルゴリズムは，それぞれの認知・行動サイクルにおける競合集合を効率的に求める方法である．

3. 照合アルゴリズム

3.1 基本的考え方

完全な競合集合を求める最も単純な方法は，それぞれの認知・行動サイクルにおいて，すべての条件要素とすべての作業記憶要素との間でパターン照合を行うことである．この方法は照合処理だけのために全実行時間の約 90～98% を費し[2)]，推論効率は極めて悪い．この問題を解決するために，Rete アルゴリズム[5)]は次の二つの基本的考え方を生かして競合集合を効率的に求めている．

（1） 現在の作業記憶に対する照合結果を条件要素単位以上で保存し，次の認知・行動サイクルでの照合に活用する．これにより，変化しなかった作業記憶要素と条件要素（条件要素はすべて変化しない）との照合を再度やり直すという無駄を排除できる．

（2） 条件要素間の部分的共通性をあらかじめ抽出しておき，同一の照合をまとめて行うようにする．これにより，同一の照合を何度も繰り返すという無駄を回避できる．

上記(1)において，「照合結果を条件要素単位以上で保存し」とあるが，この意味は条件要素を構成するクラス名や属性名と属性値のペアのすべての照合が成功したときのみ保存されるということであり，一部分が成功してもその結果は保存されない．また，同一プロダクションの二つ以上の条件要素が満足しているときには，個々の条件要素が満足しているという情報のほかに，それら複数個が同時に満足しているという情報も別途独立して保存される．同一プロダクションを構成するすべての条件要素が満足している状態はこの特殊な場合であり，この情報も前述の競合集合という形で別途保存される．競合集合を除いた，条件要素単位以上の照合結果は後述する2入力ノードのノードメモリの中に記憶される．

さて，本論文で提案する照合アルゴリズムでは，上記二つに加えて，次のような考え方が導入されている．

（3） ある作業記憶要素に対して，ある条件要素のクラス名の照合が成功したならば，異なるクラス名をもつ他の条件要素との照合は必ず失敗に終わる．また，ある属性値の照合が成功したときには，クラス名が同じでも対応する属性値が異なっている条件要素との照合は必

ず失敗する．このような知識の排他性をあらかじめ抽出しておき，実行時に利用すれば明らかに失敗する照合を回避することができる．

(4) 前述の排他関係にあるクラス名や属性値が複数個存在する場合，なるべく早い時点で照合が成功するように照合順序を工夫してやれば，より多くの照合失敗を避けられる．

3.2 条件部のネットワーク化と照合処理

条件要素とプロダクションとの対応づけはネットワークとして表現可能である．次の二つのプロダクションを例として考えよう．

```
(P P1 (C1 ^A11 V11 ^A12 V12)
      (C2 ^A21 V21 ^A22 V22)
   --->
      (…) …)
(P P2 (C2 ^A21 V21 ^A22 V'22 ^A23 <X>)
      (C3 ^A31 V31 ^A32 <X>)
   --->
      (…) …)
```

ここで，P_i はプロダクション名，C_i はクラス名，$^A_{ij}$ は属性名，$V^{(')}_{ij}$ は定数の属性値，〈X〉は変数の属性値である．動作部は後の議論に直接関係しないので省略してある．

これら二つのプロダクションに対する最も単純で自然なネットワーク表現は**図1**のようになるだろう．根ノードは条件要素の数だけ分岐し，その先のノードで分岐することはない．根ノードと端末ノード（図中の P_1 と P_2）以外のノードは，一つのリンクが入り込む1入力ノードと二つのリンクが入り込む2入力ノードに分けられる．1入力ノードではクラス名や属性値の照合が行われ，2入力ノードでは対応する二つの条件要素が同時に満足しているかがチェックされる．もし二つの条件要素間で同一の変数が使われているときには，それらが同一の値に束縛されているかがチェックされる．ある作業記憶要素がどの条件要素との照合に成功するかは，その作業記憶要素をトークンとして図1のネットワークに流してやればよい．1入力ノードでの照合に成功したトークンは次のノードへ送り出され，失敗したトークンはバックトラックして別の条件要素に対応するリンクを流れ出す．もしあるトークンが2入力ノードに到達したときには，対応する条件要素との照合が成功したことを意味する．

さて，前記(1)の考え方によれば，2入力ノードに

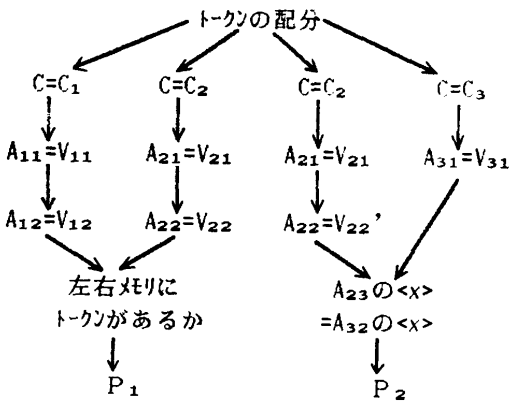

図1 単純ネットワーク
Fig. 1 Simple network.

ノードメモリを設け，そこへ到達したトークンを保存する．左側のリンクから入ってきたトークンは左メモリに，右側のリンクからのトークンは右メモリに格納することによって，そのトークンがどの条件要素と照合に成功したのかを識別できるようにする．このようにして現在の作業記憶との照合結果のすべてを2入力ノードに保持しておくならば，次の認知・行動サイクルにおいては，作業記憶の変化分，すなわち新たに追加あるいは削除された作業記憶要素だけをトークンとして流せばよいことになる．この場合，追加された作業記憶要素には“+”を付け，削除されたものには“−”を付けたトークンを考えることにより，両者の違いを識別できるようにする．マイナストークンがノードメモリに到達したときには，それと同じ内容をもつプラストークンがノードメモリの中から消去される．

次に前節(2)の考え方を導入すると，図1のネットワークは**図2**のようになる．これが Rete ネットワークと呼ばれているものであり[5]，同じ照合を行う1入力ノードがまとめられている．その結果として，Rete ネットワークのノード数は単純ネットワークのそれより減少する．トークンの流れ方やノードメモリの使用方法は単純ネットワークの場合と全く同じである．

さらに，前節(3)の考え方を導入すると，図2のネットワークは**図3**のようになる．図3では，排他関係にある1入力ノードが排他リンク（図中の破線）で連結され，他のリンクと区別されている．排他リンクが出ているノードでは，そこでの照合が成功したときには排他リンクはないものとみなされ，その先でバックトラックによりトークンが排他リンクの方へ流れることはない．照合が失敗したときにのみ排他リンクへ

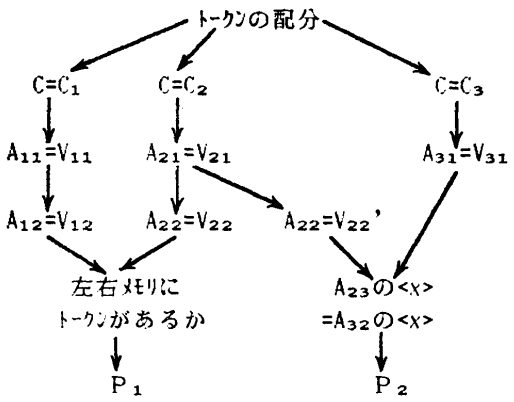

図 2 Rete ネットワーク
Fig. 2 Rete network.

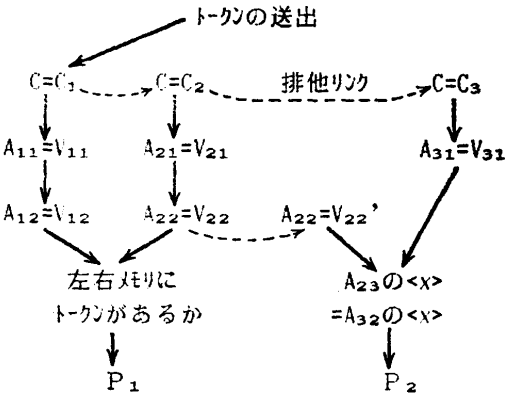

図 3 排他ネットワーク
Fig. 3 Exclusive network.

トークンが送出される．これ以外の点は Rete ネットワークの場合と同じである．

最後に，前節(4)の考え方も反映したネットワークを考える．排他関係にあるノードは排他リンクにより連結され，上流にあるノード（トークンが先に流れてくるノード）ほど早い時点で照合が行われる．ゆえに照合成功の確率が高いノードを上流にもってくれば照合失敗回数を減らすことができる．この照合成功確率は，各ノードでの照合に成功するトークンの発生頻度に比例する．この発生頻度は問題状況によって変化するので，あらかじめ厳密に算出することはできない．そこで，ここでは次のような簡便な方法をとる．つまり，条件要素の中に出現する回数の多い排他ノード（排他関係にあるノード）ほど照合成功の確率が高いと仮定し，出現回数の多い順に排他ノードを並べ換える．これはあくまでヒューリスティックな方法であるが，次節においてその効果が確認される．このような排他ノードの並べ換えを行うと図3は図4のようにな

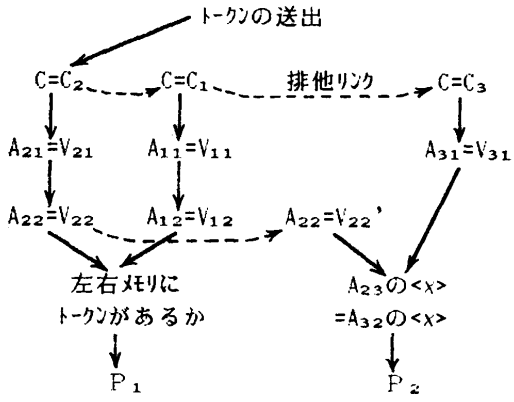

図 4 提案ネットワーク
Fig. 4 Proposed network.

り，これが本論文で提案するネットワークである．

3.3 動 作 例

本節では前節の提案ネットワークによる照合処理方法を前述の例を用いて説明するとともに，照合回数が単純ネットワークや Rete ネットワークの場合よりいかに減少するかを示す．

今，ノードメモリが空の状態で，次の六つのトークンが連続して生成された場合を考える．

$T_1 = +(C_1\ \hat{}A_{11}\ V_{11}\ \hat{}A_{12}\ V_{12})$

$T_2 = +(C_2\ \hat{}A_{21}\ V_{21}\ \hat{}A_{22}\ V_{22})$

$T_3 = +(C_2\ \hat{}A_{21}\ V_{21}\ \hat{}A_{22}\ V'_{22}\ \hat{}A_{23}\ V_x)$

$T_4 = +(C_3\ \hat{}A_{31}\ V_{31}\ \hat{}A_{32}\ V'_x)$

$T_5 = +(C_3\ \hat{}A_{31}\ V_{31}\ \hat{}A_{32}\ V_x)$

$T_6 = -(C_2\ \hat{}A_{21}\ V_{21}\ \hat{}A_{22}\ V_{22})$

まず，トークン T_1（以下では単に T_1 と書く）は図4の根ノードから流れ出し，最初の1入力ノードでクラス名の照合が行われる．クラス名は C_2 でないので照合に失敗し，T_1 は排他リンクへ送出される．再びクラス名の照合が行われるが，今度はそれに成功するので $C=C_3$ につながる排他リンクはないものとみなされ，もう一方のリンクへ T_1 が送出される．次のノードでは属性 A_{11} の値が V_{11} であるかの照合に成功し，T_1 は次のノードへ引き渡される．そこでは属性 A_{12} の値が V_{12} であるかの照合がなされ，それに成功する．次に T_1 は2入力ノードに右側のリンクを通って到着し，右のノードメモリに格納される．この2入力ノードでは単に左のノードメモリの中にトークンがないかどうかがチェックされ，何もないので照合処理は完了する．

ひき続いて T_2 が根ノードから送出され，クラス名の照合，属性 A_{21} の値の照合，属性 A_{22} の値の照合

のすべてに成功して，2入力ノードに左側のリンクを通って到達する．ここでは，T_2 が左のノードメモリに格納されてから，右のノードメモリにトークンが存在するかが調べられる．今度は T_1 が存在しているので，この2入力ノードは次の合成トークンを生成して端末ノードへ送出する．

$$T_{12}=+[(C_2\ \hat{}A_{21}\ V_{21}\ \hat{}A_{22}\ V_{22})\ (C_1\ \hat{}A_{11}\ V_{11}\ \hat{}A_{12}\ V_{12})]$$

ここで注意を要することは，2入力ノードから合成トークンが送出されても，左右のノードメモリには合成前のトークン（T_1 および T_2）を残しておくということである．これによって条件要素単位での照合状態を保持するわけである．合成トークン T_{12} が到着した端末ノードは，T_{12} の符号が＋であるのでプロダクション名 P_1 と T_{12} を競合集合に追加する．T_2 はこれまでに二つの分岐ノードを通過してきたが，そこから出るもう一方のリンクは両者とも排他リンクであった．ゆえに，T_2 はバックトラックする必要がないので照合完了となる．

次のトークン T_3, T_4, T_5 は条件部に同一変数〈X〉が二つ存在するプロダクション P_2 が競合集合に加えられる過程を示すためのものである．まず，T_3 は前と同じような照合処理を行われて図4の右側の2入力ノードの左メモリに格納される．同様にして T_4 は同じ2入力ノードの右メモリに格納される．属性 A_{23} の変数〈X〉には V_x が束縛され，属性 A_{32} の変数〈X〉には V'_x が束縛されている．ゆえに，同一変数〈X〉に異なる値が束縛されているので，この2入力ノードは合成トークンを生成せず照合を完了する．次の T_5 は同様にして同じ2入力ノードに右側のリンクを通って到達し，その右メモリに格納される．今度は，T_5 の属性 A_{32} の値と左メモリにある T_3 の属性 A_{23} の値が同一であるので照合が成功し，次の合成トークンが生成される．

$$T_{35}=+[(C_2\ \hat{}A_{21}\ V_{21}\ \hat{}A_{22}\ V'_{22}\ \hat{}A_{23}\ V_x)\ (C_3\ \hat{}A_{31}\ V_{31}\ \hat{}A_{32}\ V_x)]$$

T_{35} を受けた端末ノードはプロダクション P_2 と T_{35} のペアを競合集合に追加する．

最後のトークン T_6 は，マイナストークンに対する処理を説明するためのものである．マイナストークンの場合にも2入力ノードに達するまではプラストークンと同様の処理が行われる．ゆえに T_6 は T_2 と同じ処理を受けて図4の左側の2入力ノードに左側のリンクを通って到達する．2入力ノードは T_6 がマイナストークンなので，同一の内容をもつプラストークンを左メモリの中からさがし，T_2 を見つけ出す．そして，T_2 を消去するとともに，次の合成トークンを生成し送出する．

$$T_{62}=-[(C_2\ \hat{}A_{21}\ V_{21}\ \hat{}A_{22}\ V_{22})\ (C_1\ \hat{}A_{11}\ V_{11}\ \hat{}A_{12}\ V_{12})]$$

この T_{62} は以前にこの2入力ノードから送出された T_{12} と符号を除いて全く同一である．ゆえに端末ノードは競合集合に以前に加えた P_1 と T_{12} のペアを競合集合の中から削除する．以上で T_1〜T_6 に対する照合処理は終了する．新たに発生するトークンに対しても同様の処理が実行される．

図3のネットワークを用いた場合の照合処理も同じ考え方で実行される．排他リンクを含まない図1および図2のネットワークを用いた場合には，途中で分岐したときの処理方法だけが異なる．すなわち，分岐ノードでは，トークンのコピーが分岐数だけ作られて送出される．並列処理機能のない通常の計算機では，コピーを作らずにとりあえず一つのリンクにトークンを流し，途中で照合が失敗するか，あるいは端末ノードに到達したときに，バックトラックして分岐点にもどり，別のリンクに流してやるようにする．

図1〜図4のネットワークを用いたときのそれぞれの照合過程を**表1**にまとめた．ノードメモリの初期状態は空であり，前述の T_1〜T_6 のトークンがこの順序で発生した場合を想定している．照合に成功した場合を S，失敗した場合を F で表している．ただし，根ノードおよび端末ノードでは照合を行わないので何も書いていない．また，2入力ノードのどちらかのノードメモリにトークンが到達した場合，他方のノードメモリにトークンが存在しないときにはFとしている．ここでの例ではプロダクション数がわずか二つであるが，知識の構造化を進めるにともなって照合回数が次

表1 各アルゴリズムにおける照合過程
Table 1 Pattern match process in each algorithm.

トークン	単純	Rete	排他	提案
T1	SSSFFFF	SSSFFF	SSSF	FSSSF
T2	FSSSSSSFF	FSSSSFF	FSSSS	SSSS
T3	FSSFSSSFF	FSSFSFF	FSSFSF	SSFSF
T4	FFFSSF	FFSSF	FFSSF	FFSSF
T5	FFFSSS	FFSSS	FFSSS	FFSSS
T6	FSSSSSSFF	FSSSS	FSSSS	SSSS
照合回数	46 (21)	35 (16)	30 (11)	28 (9)

（Sは照合成功，Fは照合失敗，括弧内はFの数を表す）

第に減少することがわかる．もう少し細かく見ると，知識の共通性を利用することは成功と失敗の両方の照合回数を減少させるが，知識の排他性ならびに利用頻度を用いることは照合失敗の回数だけを減らすことに効果があることがわかる．

3.4 EUREKA との比較[12),13)]

EUREKA では，条件要素間にまたがる変数が多数存在するときの効率改善がねらいの一つである．このために仮想的な候補ノードを導入しているが，その詳細が不明なのでねらいが成功したかどうかは判断できない．ただ言えることは，本論文の提案方式における変数の取扱いは Rete アルゴリズムと全く同じであり，何ら改良を加えていないということである．ゆえに，候補ノードの導入が効率改善につながるのであれば，本方式と組み合わせて利用することが可能と思われる．

また，EUREKA では，条件要素を構成する個々の条件に評価値を与え，それに基づいて条件の並べ替えを行い高速化を図っている．この並び替えはあくまで一つの条件要素内で行われており，いわば縦方向の並び替えである．それに対し，提案方式ではすべての条件要素にわたるグローバルな並べ替えを行っており，横方向の並べ替えと言うことができる．もちろん，提案方式でも Rete アルゴリズムと同様に縦方向の並べ替えも行っている．すなわち，同一の照合をできるだけまとめるために，条件要素間で共通性の高い照合ほどネットワークの上位にくるようにしている．ただし，クラス名の照合は最上位に固定している．また，条件要素間にまたがる変数の照合は2入力ノードで行うので，各条件要素の最後にもってくる．ゆえに，EUREKA 独自の並び替えとしては，OR 条件よりも AND 条件を上位にもってくることと，比較演算子“=”を優先させることの二つと考えられる．OR 条件や比較演算子が多数含まれるようなプロダクション集合に対してはこの方法は効果があるだろう．ただし，この方法と本論文の方法とは対立するものではなく，共存することが可能と思われる．

さらに，EUREKA では，根ノードにポインタテーブルを記憶させることにより，トークンを流すべきリンクを特定するという考え方を提案している．しかし，根ノードにつながる複数のノードがもし排他関係になければ，特定したリンク以外にも照合に成功する可能性が残ることになり，それらを見落としてしまう恐れがある．EUREKA ではポインタテーブル方式の大前提となる排他性に関する議論が全くなされていない．また，ポインタテーブルの作成方法や検索方法についても全く述べられていないので計算量の比較を行うことはできない．

4. 試作システムの概要

前章で述べた照合アルゴリズムを用いたプロダクションシステム記述言語処理系を試作した．そのソフトウェア構成を処理の基本的流れがわかるように書くと図5のようになる．本処理系は大きく分けると(1)ネットワークコンパイラ，(2)推論エンジン，(3)コマンドインタプリタの三つから構成される．

コンパイラは外部記憶装置に格納されているプログラム（プロダクションの集合），あるいはキーボードから直接入力されたプログラムをS式に変換し，条件部をコンディションメモリに，動作部をアクションメモリに格納する．次に，コンディションメモリの情報を，照合アルゴリズムに対応したネットワークの形に変換し，ネットワークメモリに格納する．本処理系では，図2～4に対応したネットワークを生成することができ，OPS5 にはない新しいコマンド optimize に

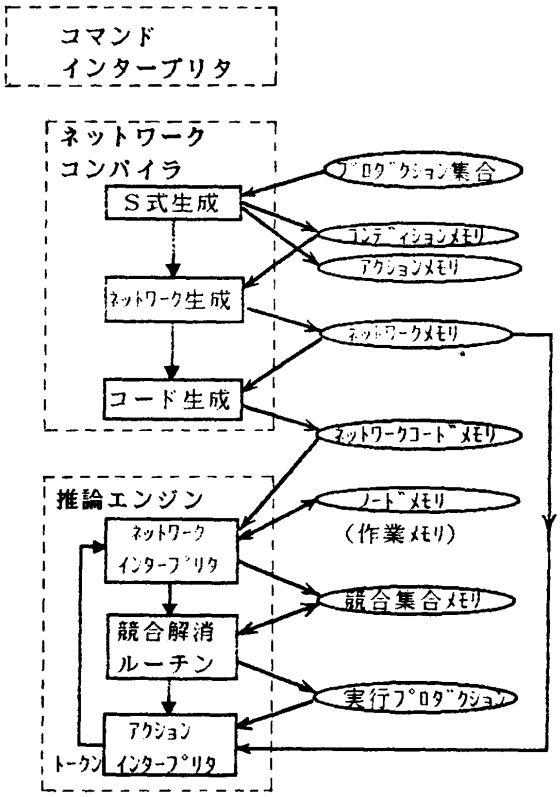

図5　システム構成と処理フロー
Fig. 5　System structure and processing flow.

より構造化レベルを選択できるようにした．最後に，このネットワークは線形化[5]されてネットワークコードとなる．このコードは OPS5 とかなり異なっており，図2～4の三つの構造化レベルに対応できる内部表現になっている[11]．このネットワークコンパイラは前処理として知識ベースの構造化を行う部分であり，一度実行すれば知識ベースに変更が加えられない限り再実行の必要はない．

推論エンジンは実際に推論計算を行う部分で，知識の構造化が済んだあとでのみ実行可能である．まず，ネットワークインタプリタは，生成されたトークンに対してネットワークコードを順次解釈しながらパターン照合処理を行う．そのトークンがある条件要素を満足したとき，すなわち2入力ノードに到達したときにはノードメモリの内容を更新する．さらに，そのトークンによって新たなプロダクションが成立したり，これまで成立していたプロダクションが不成立になったときには競合集合メモリの内容も更新される．競合集合は競合解消戦略にもとづいてソートされた形で保存される．ゆえに，競合集合メモリの更新は競合解消ルーチンが行う．実行プロダクションは常に競合集合メモリの先頭に存在している．

コマンドインタプリタはユーザの入力したコマンドを解釈して対応する処理ルーチンに制御を渡す．未サポートのコマンドも若干あるが，逆に OPS5 にないコマンドも準備されている[11]．

なお，本処理系は MS-DOS 版 Turbo Pascal で記述された．

5. 実験的性能評価

5.1 実験内容

Rete アルゴリズムと提案アルゴリズムの性能を比較するためにサンプルプログラムを用いて処理時間を計測した．サンプルプログラムは(1)列車機器故障診断[10]，(2)ハノイの塔[8]，(3)モンキーとバナナ[9]の三つである．これらを用いた理由は，単に OPS5 の文法で書かれたプログラムのリストが入手できたからにすぎない．それぞれのアルゴリズムおよびプログラムに対して，処理時間をコンパイル時間と実行時間の二つに分けて計測した．ここでいうコンパイル時間とは，プロダクションを構造化してネットワークコードを生成するのに要する時間である．実行時間とは，実際の推論計算に要する時間である．使用計算機は PC-9801 である．

表 2 計算時間の比較
Table 2 Comparison of computing time.

プログラム名	列車機器故障診断	ハノイの塔 ディスク=3	モンキーとバナナ
プロダクション数	85	8	27
Rete アルゴリズム	11.42	2.53	7.00
(A)	57.54	7.76	7.38
提案アルゴリズム	33.42	2.55	7.63
(B)	40.41	6.73	6.84
増減率 (%)	292.64	100.79	109.00
($B/A*100$)	70.23	86.73	92.68

(上段：コイパイル時間，下段：実行時間，単位：sec)

5.2 結果と考察

実験結果は表2のようになった．提案アルゴリズムを用いた場合のコンパイル時間は，Rete アルゴリズムの場合より大きいが，これはコンパイル時に知識の排他性の検出と出現頻度によるノードの並べ換えが行われているためである．しかし，実行時間は逆に提案アルゴリズムの方が小さくなっており，コンパイルに時間をかけた効果がここに現れている．このように，コンパイル時間と実行時間との間にはトレードオフ関係が存在する．実行時間の減少率がサンプルによってばらついているが，これは知識の排他性を利用することの効果が，知識ベースに内在する排他性の量に依存しているからである．サンプルの中では列車機器故障診断プログラムが知識の排他性を一番多く含んでいるといえる．

本処理系は現在のところ Turbo Pascal で記述されているため，プログラムおよびデータのアドレス空間が 64kB と小さい．ゆえに，プロダクション数が 100 を越えると，個々のプロダクションが大きい場合には処理が不可能になるのが現状である．しかし，これは本質的な問題ではなく，より大きなデータ領域を扱える Lattice C などで書き換えれば解決される．

6. あとがき

本論文では，知識の排他性ならびに知識の利用頻度を活用したパターン照合アルゴリズムを提案し，Rete アルゴリズムにおける無駄な照合を削減できることを示した．

今回試作した処理系は Turbo Pascal で記述したため取り扱えるプロダクション数をこれ以上増やす余裕はあまりない．ゆえに，未サポートのコマンドの整備も含めて現在C言語で書き換えを進めている．今後の

課題には，知識ベースの開発効率に大きな影響を及ぼすコンパイル時間の短縮化がある．プロダクションがわずか一つ追加されたときでも始めから知識の構造化をやり直すのは明らかに無駄だからである．実用面からは，問題の特性や開発段階に応じて構造化レベルを切り換えることにより開発効率を上げることが有効であり，この方向からのアプローチも重要な研究課題と思われる．

謝辞　貴重な資料を提供して頂いた三菱電機(株)の匹田志郎氏，プログラムを作成して頂いた本学卒業生の安富伸浩氏に感謝する．なお，本研究の一部は福岡工業大学エレクトロニクス研究所の助成によるものであり，あわせて謝意を表したい．

参考文献

1) Hayes-Roth, F. and Mostow, D. J.: An Automatically Compilable Recognition Network for Structured Patterns, *Proc. of 4th Int. Joint Conf. of Artif. Intell.*, pp. 246-251 (1975).
2) McDermott, J., Newell, A. and Moore, J.: The Efficiency of Certain Production System Implementations, in Waterman, D. A. and Hayes-Roth, F. (Eds.): *Pattern Directed Inference Systems*, pp. 155-176, Academic Press, New York (1978).
3) Forgy, C. L.: On the Efficient Implementation of Production Systems, Ph. D. Thesis, Carnegie-Mellon University (1979).
4) Forgy, C. L.: OPS 5 User's Manual, Department of Comput. Sci., Carnegie-Mellon University (1981).
5) Forgy, C. L.: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, *Artif. Intell.*, Vol. 19, pp. 17-37 (1982).
6) Forgy, C. L. et al.: Initial Assessment of Architectures for Production Systems, *AAAI-84* (1984).
7) 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).
8) 小林：プロダクションシステム，情報処理，Vol. 26, No. 12, pp. 1487-1496 (1985).
9) 吉村：ルールベース・エキスパート・システム構築ツールの基本形 OPS 5，日経コンピュータ別冊，pp. 73-88 (1985).
10) 匹田，西内，荒屋：列車機器故障診断システムの OPS 5 によるプロトタイピング，第 5 回シミュレーション・テクノロジー・コンファレンス発表論文集，pp. 23-26 (1985).
11) 百原，安富，荒屋：OPS 5 のパーソナルコンピュータ上での実現，福岡工業大学研究論集，No. 19, pp. 111-124 (1986).
12) 田野，増位，舩橋：知識処理ソフトウェア EUREKA における推論機構の高速化，第 31 回情報処理学会全国大会講演論文集，5 M-7, pp. 993-994 (1985).
13) 田野，増位，坂口，舩橋：知識処理ソフトウェア 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講演論文集，5 P-4, pp. 1517-1518 (1986).

(昭和 61 年 8 月 6 日受付)
(昭和 62 年 4 月 15 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，IEEE 各会員．

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．

田町　常夫（正会員）

昭和 20 年九州大学工学部電気卒業．昭和 24 年九州大学工学部通信工学科助教授．昭和 37 年同教授．その後情報工学科，大学院総合理工学研究科（情報システム学専攻）教授を経て，昭和 61 年福岡工業大学工学部通信工学科教授，現在に至る．工学博士．自然言語の機械処理，機械翻訳，言語理解，画像理解の研究に従事．電子情報通信学会会員．