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†

田野俊一†† 増位庄一††
坂口聖治†† 舩橋誠壽††

知識ベースシステムのリアルタイム性の要求される分野への適用を目的として，知識ベースシステム構築用ツール EUREKA を開発した．本システムは，高記述性・高速性を特徴としており，本論文では，高速処理方式について説明する．まず，従来の高速アルゴリズムである履歴情報を持つ弁別ネットを用いた方式（例えば，Rete Match Algorithm）を，複雑・大規模なルール記述においても高速化するための，3つの課題—ネットワーク表現，ネットワーク最適化，ネットワーク処理—を明らかにした．次に，これらの課題の解決法として，(1)候補ノード，merge ノードを導入した新しいタイプのネットワーク表現法，(2)ノードの特性により，ネットワークを変形・分割・統合する方法，(3)ネットワーク中の情報更新を効率よく行う方法，を示した．さらに，本高速処理方式の性能評価を，(1)実システムにおける有効性の評価，(2)ルール・オブジェクト記述モデルによるルール数と推論速度の関係の予測・実証，に関して行った．この結果，本高速処理方式は，複雑・大規模なルール記述においても高効率であり，また，ある記述モデルにおいて推論速度のルール数非依存性を確認した．

1. はじめに

人工知能の現実的な応用として，応用人工知能あるいは知識工学と呼ばれる手法が注目を浴びている[1]．

ルールベースシステムでは，人間の持つ経験的知識を，断片的な IF～THEN～形式のルールで表現するため，知識の変更，追加が容易であり，システム構築の際，(1)段階的なシステム構築が可能，(2)システムの機能変更に柔軟に対応可能等の特徴を有する．このため，機能が非常に複雑あるいは不明確なシステムの記述に向いており，現在では，知識工学の代表的知識表現手法となっている．しかし，適用分野は，推論速度の遅さが障害となって，リアルタイム性の要求されない分野がほとんどであった．

そこで，われわれはリアルタイム性の要求される分野にも対応できる処理速度を有すると同時に，知識を柔軟に表現できる高記述性もあわせ持つ知識ベースシステム構築用ツール EUREKA[9] (Electronic Understanding and REasoning by Knowledge Activation) を開発した．本システムは，プロダクションシステムとオブジェクト指向システムを融合することにより高記述性を，一方，ルールの条件部をネットワークに変換し，それを用いて，実行可能ルールを捜しだすことにより高速性を達成している[10,11]．

本論文では，ネットワークを用いたプロダクションシステムの高速処理方式を示す．

以下，1.1 節では，EUREKA について概説し，1.2 節では，プロダクションシステムの高速処理に関する他の研究報告を示す．

第2章で，本高速処理方式開発にあたっての基本的な方針および解決すべき問題点を示す．第3章では，本高速処理方式を説明し，第4章で，性能評価を行う．

1.1 EUREKA の概要

EUREKA は，ワーキングメモリに対応するオブジェクトベースに記憶されているオブジェクトの内容と，プロダクションメモリに対応するルールベースに格納されているルールの条件部とを，推論エンジンが比較しルールを順次実行することにより推論を進めるプロダクションシステム型の推論機構を基本とし，メタルールによる推論過程のコントロール機構，オブジェクトベース内におけるオブジェクト指向システムの実現等に関して拡張したシステムである．

図1に示すように，オブジェクトは，オブジェクト名称，データ部，メソッド部より成り，データ部は，フレーム的に，項目と，値の組で表される．一方，ルールは，オブジェクトのデータ部に関する記述より成る条件部と，オブジェクトに対するメッセージ送信より成る実行部で構成される．

条件部は，1つのオブジェクトの状態に関する条件

† A Fast Pattern Match Algorithm for a Knowledge Based Systems Building Tool—EUREKA by SHUN'ICHI TANO, SHOUICHI MASUI, SEIJI SAKAGUCHI and MOTOHISA FUNABASHI (Systems Development Laboratory, Hitachi, Ltd.).

†† (株)日立製作所システム開発研究所

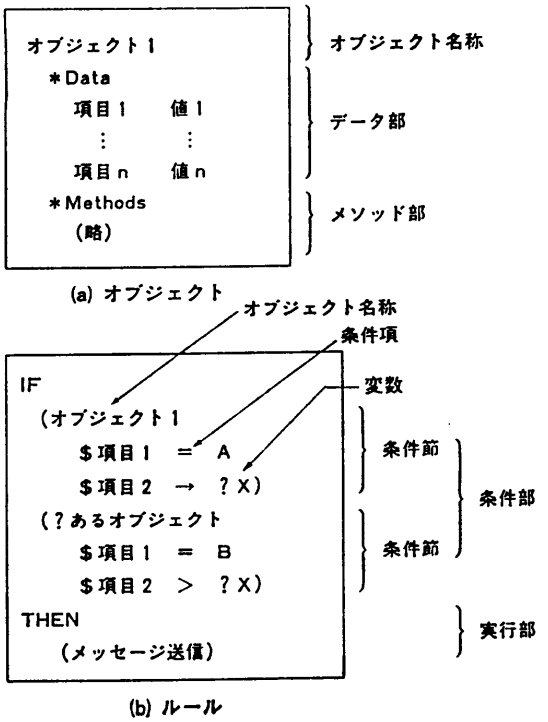


図1 EUREKAにおけるオブジェクト，ルール記述例
Fig. 1 Examples of rule and object description in EUREKA.

を表す条件節の複数の並びで構成され，それぞれの条件節が特定のオブジェクトによって満足された時，このルールは実行可能であると判定する．このルールは(1) オブジェクト1の項目1が，"A" であって，かつ，(2) 項目1が，"B" であり，項目2が，オブジェクト1の項目2より大きいオブジェクトが存在する場合に，この条件を満たすオブジェクトの組で，実行可能となる．"?あるオブジェクト"，"?X" は変数を表し "→" を用いて代入できる．

1.2 高速処理に関する他の研究結果

知識ベースシステムの問題点のひとつである処理速度に関して，数種の手法が提案されている．これら従来手法は，「ルール条件部の成立判定において，各ルールの条件部と対象世界の状態を表すワーキングメモリエレメント (WME) のすべてを比較することの非効率さ」を問題点として捉え，「ルールの条件部の成立判定を，そのルールが関係する WME に関してのみ行う」ことを基本方針としている．そこで，どの WME がどのルールの条件部と関連しているか，あるいは，逆の関連図を作成し，これを用いて，判定処理の高速化を図る方式を，McDermott[2]，Konolige[3]，Mark[4]，鶴田[5] が提案している．

しかし，これらの方式は，ルール条件部と (特定された) WME をルール実行のたびごとに比較する必要がある．この問題点に対し，C. L. Forgy は，WME とルール条件部との関連図内に，条件成立判定の履歴情報を埋め込み，ネットワークとして表現することにより，状態の変化した WME のみの再処理で，条件判定を行うことのできる Rete Match Algorithm を提案している[6]．また，条件の排他性に着目し，RETE アルゴリズムを部分的に改良した方式を荒屋[13]が提案している．

本論文で示す高速処理方式は，従来手法の中で最も処理効率がよい履歴情報を持つネットワークを用いた考え方を基本としつつ，新たにネットワーク表現法，変形法，情報管理法を考え直すことにより，複雑かつ大規模なルール記述においても効率のよい処理を可能としたアルゴリズムである．

2. 高速化の基本方針と解決すべき問題点

2.1 高速化の基本方針

ルールベースシステムでは，実行可能ルールを捜しだす Match 処理および実行すべきルールを1つ選択する Conflict Resolution 処理に多大な処理時間を要する．特に Match 処理は，システム全体の処理時間の大部分を占めており[2]，この処理の効率が，全体の処理速度に大きく影響を与えるため，Match 処理の効率化が最も重要なターゲットとなる．

2.2 従来方式の背景

認知心理学の分野では，「鳥であり，羽を使って泳ぐものは，ペンギンである」などの知識の記憶構造・利用に関する問題を，人間の認知過程のモデリングの問題として捉え，研究している．

そこでのモデリングの一方法として，「鳥である」，「羽を使って泳ぐ」等をそれぞれ1つのノードで表し，そのノードの条件を満たした場合にチェックすべきノードへの枝と，満たさない場合にチェックすべきノードへの枝を有するツリー (弁別ネットと呼ぶ) により，上記知識を表現し，このツリーを用いて対象世界で観測された状態から結論を導く方式がある．これを具体化した方式である EPAM[7] (Elementary Perceiver And Memorizer) では，パターンのネガティブな性質を積極的に用いて弁別ネット (EPAM ネット) を展開し，対象のすべての状態のパターンが既知であり，かつ，同時に2つのパターンが成立しない場合の高速 MATCH 処理を実現している[8]．ルール

ベースシステムの対象は，多くの場合，上記の制約（対象の任意の状態において必ず1つのルール条件部が成立）を満足するように知識（ルール）を収集することは困難であり，EPAM ネットの直接適用は難しい．

しかし，弁別ネットの構造的特質は，処理の観点から見て以下のような好ましい性質を持っている．

（1）同一条件を表すノードを共通化できるため，同じ条件の重複チェックを回避できる．

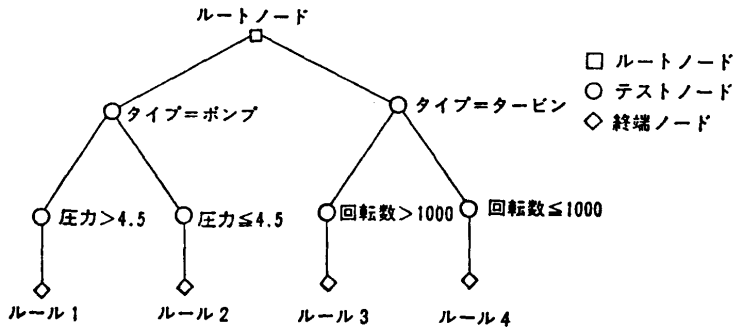
（2）ノードの処理制御を，ネットワークとして明示的に表現でき，あるノードでの条件チェックが失敗した場合，このノードに続くノードの真偽判定は無意味であることが表現できる．

そこで，**図2**のような弁別ネットを考え，ルールベースシステムへの適用を試みてみる．まず，同図(a)の条件部の「タイプ＝ポンプ」等を，パターンの構成要素と捉え，テストノードとして表す．それに，頂点を表す ROOT ノード，条件成立を表す終端ノードを加えて弁別ネットを構成する（同図（b））．対象の状態（オブジェクト）は，ROOT ノードより順にネットワーク内を流れ下り，各テストノードでは，流れてきた情報に対し，そこに記憶されているチェック項目の真偽を判定し，成功の場合のみ，流れてきた情報を，テストノードに続くノードすべてに対し流す．このようにすれば，オブジェクトが終端ノードに到達した場合，その条件が満たされたと判定できる．

このような表現・処理法では，「満足されるパターンは唯一でなければならない」という上記の問題点を解決し，かつ，弁別ネットの好ましい性質も保存している．しかし，図2の弁別ネットでは，1ルール実行するごとにすべてのオブジェクト

```
ルール1  IF（?機械  $タイプ＝ポンプ   $圧  力＞4.5 ）THEN……
ルール2  IF（?機械  $タイプ＝ポンプ   $圧  力≦4.5 ）THEN……
ルール3  IF（?機械  $タイプ＝タービン $回転数＞1000）THEN……
ルール4  IF（?機械  $タイプ＝タービン $回転数≦1000）THEN……
```

(a) ルールの例

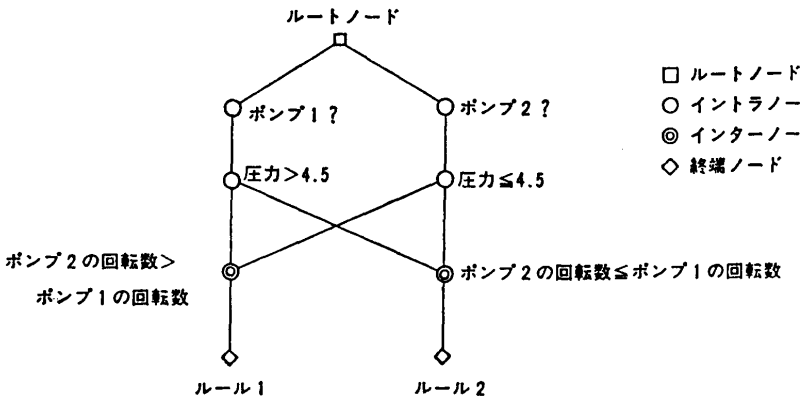


(b) 弁別ネット

図 2 弁別ネットによるルール条件の表現
Fig. 2 Examples of rules and its discrimination net representation.

```
ルール1  IF（ポンプ1  $圧力＞4.5  $回転数→？X）
           （ポンプ2  $圧力≦4.5  $回転数＞？X）
         THEN……
ルール2  IF（ポンプ1  $圧力＞4.5  $回転数→？X）
           （ポンプ2  $ 圧力≦4.5 $ 回転数≦？X）
         THEN……
```

(a) ルールの例



(b) ルール条件部のネットワーク表現

図 3 ルール条件部のネットワーク表現
Fig. 3 Network representation of rule condition.

をROOTノードより流す処理を行わなければならない.

この問題点は，弁別ネット内にパターンマッチの履歴を記憶しておき，ネットワークには変化した状態量(オブジェクト)のみをネットワークに流すようにすることで解決できる．図2(b)の場合は，終端ノードへの枝に到達したオブジェクトのみを記憶すればよいが，**図3**(a)のように，変数を用いて異なるオブジェクトの項目間の比較をする場合には，これだけでは不十分である．そこで，テストノードを，項目と定値との比較を表すイントラノード，異なるオブジェクト間の項目の比較を表すインタノードに分け，履歴情報をインタノード，終端ノードに至る枝に記憶するネットワークを考える必要がある(同図(b)).

例えば，ポンプ1の圧力が5，回転数が2000であり，ポンプ2の圧力が3，回転数が2500である状態では，ネットワーク中のインタノードに至る枝には，それぞれ，ポンプ1，ポンプ2が記憶され，ルール1の終端ノードには，ポンプ1，ポンプ2の組が記憶される．この状態では，ルール1が実行可能であることが示されている．さて，この状況で，ポンプ2の回転数が1500になった場合，ルートノードから，ポンプ2オブジェクトのみを流せばよい.

このように，履歴情報を用いる方法では，変化したオブジェクトに関する処理のみで，ルールの条件成立判定を行える.

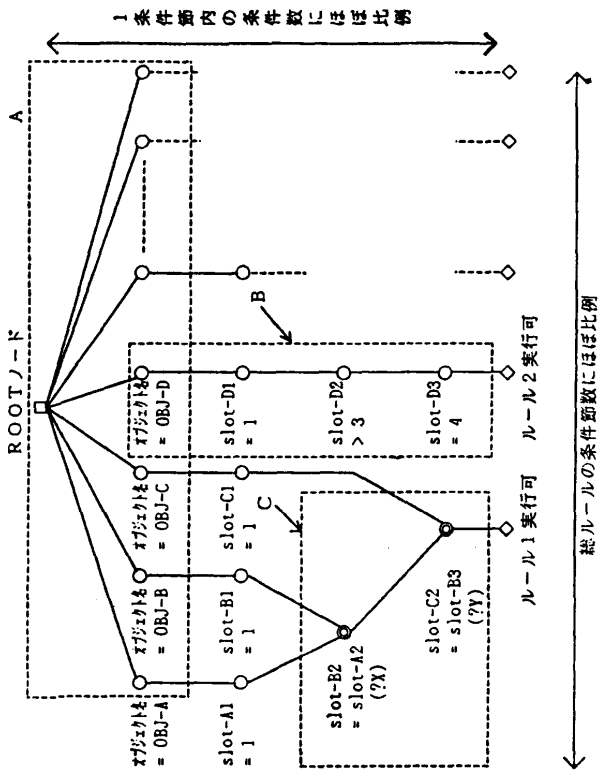
これに分類される方法として Rete Match Algorithm が有名であり，実用システムに適用されており，高速性も確認されているが以下に述べるいくつかの点で改良すべき問題を含んでいる.

2.3 従来方式の問題点と高速化のポイント

ここでは，RETE アルゴリズムにおける問題点を明示し，新しいネットワーク表現・処理形式を考察する.

2.3.1 ネットワーク表現上の問題点

図4(a)のルールは，Rete の場合，同図(b)のネットワークに変換される．このネットワークは，通常横に広く縦に浅い形となる．これは，横幅が総ルールの条件節数にほぼ比例し，深さが1つの条件節内の個々の条件数にほぼ比例するので，一般的な特徴であると考えてよい．さて，1つの



(b) (a)のルールネットワーク表現

図4 ルールネットワークの特徴
Fig. 4 Features of rule network.

```
ルール1
  IF  (OBJ-A $slot-A1 = 1  $slot-A2 → ?X)
      (OBJ-B $slot-B1 = 1 $slot-B2 = ?X  $slot-B3 → ?Y)
      (OBJ-C $slot-C1 = 1  $slot-C2 = ?Y )
  THEN ...
ルール2
  IF  (OBJ-D $slot-D1 = 1 $slot-D2 > 3  $slot-D3 = 4)
  THEN ...
ルール3
  IF  (OBJ-E  $slot-E3 ...
       :      :
```

(a) ルールの例

オブジェクトの変化は，少数のルールにしか影響しないのであるが（逆に，この性質がない場合は，履歴情報を用いた方式は，不適である），このような形状のネットワークの場合は，点線部Aで示した ROOT ノード付近の処理が多くなる．この処理量は，ネットワークの横幅，つまり，ルール数に比例してしまう．これを，ROOT ノードの問題点と呼ぶ．

つぎに，ネットワークの内部構造について考える．ネットワークは，イントラノードが直線的に並ぶ部分と，その直線部分を結合するインタノードの部分より構成される．

イントラノードが直線的に並ぶ部分，例えば，点線部Bは，項目の値を順にチェックする部分である．このチェックの順序は，処理の効率化に大きく影響する．例えば，「SLOT-D2>3」の条件が「SLOT-D1=1」の条件より，オブジェクトの通過を阻止する可能性の高い条件であった場合，前者を，上位に位置させた（すなわち先にチェックする）方が処理効率がよくなる．このように，個々のノードは，ノードに記憶されている制約条件のきつさや，必要となる処理量によって特徴づけられる．これらの特性を生かし，ノードの位置関係を決めることにより，処理の効率化が図れる．これを，ノード位置の問題点と呼ぶ．

点線部Cは，直線部分をインタノードで結合する部分である．この結合では OBJ-A に関するネットワーク処理を行う場合にも，インタノード「SLOT-C2=SLOT-B3」の処理が必要となる．つまり，変化したオブジェクトである OBJ-A と全く関係のないオブジェクト同士「OBJ-B と OBJ-C」の再比較が必要となってしまい，「変化したオブジェクトに関する条件チェックだけをやりなおす」という高速化に対する基本的な考え方から逸脱してしまう．これは，以下の場合に発生する．

COMP (I, J) をオブジェクト I と J の比較を表す条件とする．例えば，COMP (OBJ-B, OBJ-C) は，オブジェクト OBJ-B, OBJ-C の項目の比較を表すインタノードである．ここで，1つのルールの条件部に含まれる COMP の集合の中で，

{COMP (I, J), COMP (I, K)}，ただし $J \neq K$ を満たす COMP の組が存在する場合，オブジェクト J, K のいずれかのネットワーク処理において，必ず上記の問題点が発生する．

さらに，変数による値比較が論理和で結ばれる条件記述の場合，上記に加え，図5に示すようなネット

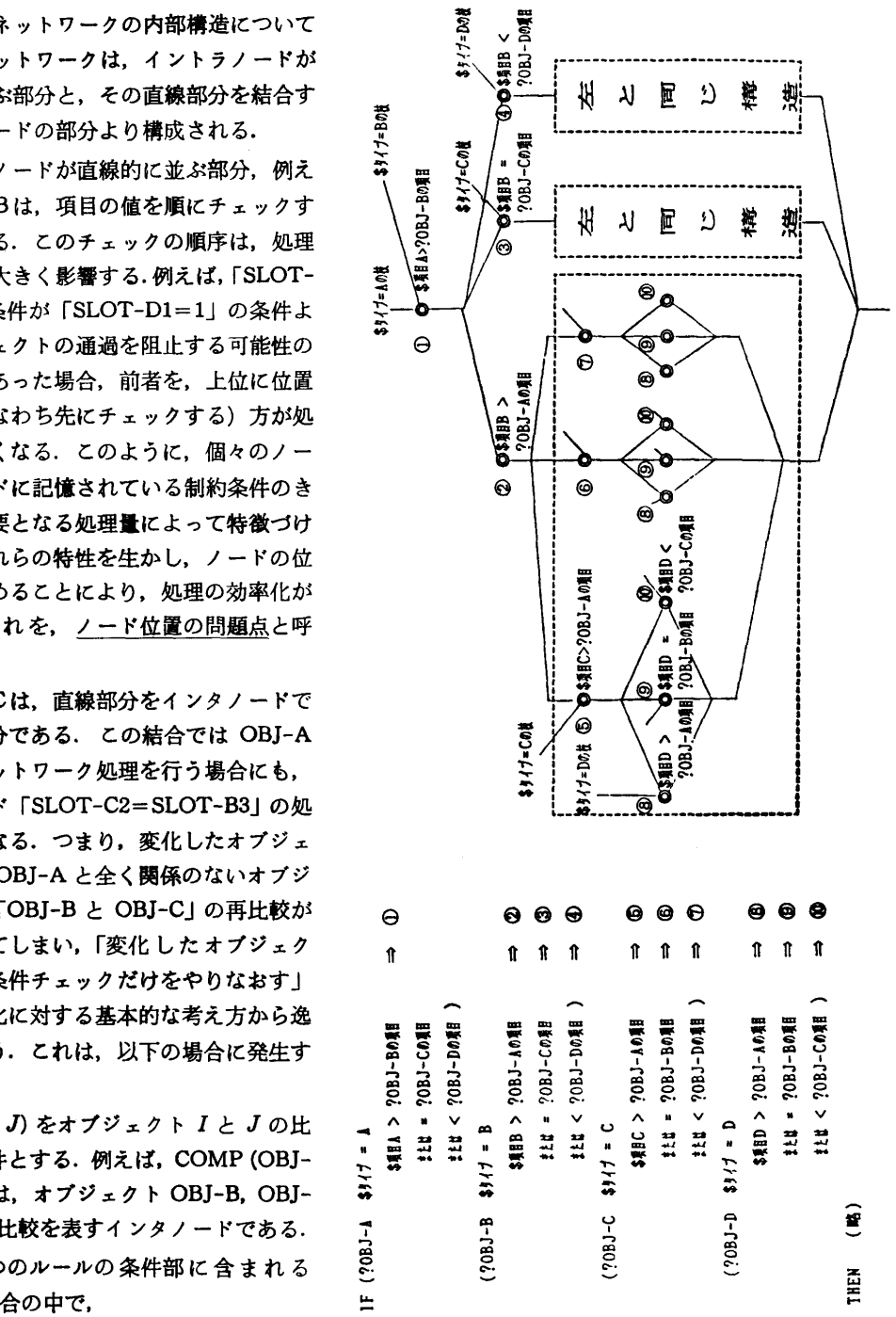


図5 複雑なルールのネットワーク表現の例
Fig. 5 Examples of a complex rule and its network representation.

ワークの肥大化という問題点も発生する．

同図（a）のルールでは，タイプA，B，C，Dのオブジェクトに関して，互いに項目値の比較を行っている．タイプAのオブジェクトの項目Aは，タイプBのオブジェクトのある項目の値より大であるか，または，タイプCのオブジェクトのある項目と等しいか，または，タイプDのオブジェクトのある項目の値より小であることを表しており，これは他のタイプのオブジェクトについても同様である．

このルールをネットワーク化する過程を示す．まず，ROOT ノードより，「タイプ＝A」「タイプ＝B」「タイプ＝C」のノードを結合する．つぎに，「タイプ＝A」のノードの下に，「項目 A＞？ OBJ-B の項目」「項目A＝？ OBJ-C の項目」「項目A＜？ OBJ-D の項目」のインタノードを論理和の関係で配置する．

しかし，「項目A＞？OBJ-B の項目」（①の条件）の比較は，？OBJ-B の条件節の条件を満たすオブジェクトとの比較であり，これを確認するために，このノードの下に，「項目 B＞？OBJ-A の項目」（②の条件）「項目 B＝？OBJ-C の項目」（③の条件）「項目B＜？OBJ-D のある項目」（④の条件）のインタノードを論理和の関係で配置する必要があり，このノードの下に，⑤，⑥，⑦のノードが必要となり，…を繰り返しネットワークを生成する必要がある．

このように，他のオブジェクトとの値比較が論理和で結合されている場合は，論理和で結合された条件を論理和のない形に分解しネットワークに変換せざるを得ない．上記の例では，3^4 通りのルールに分解されることになりノードを共通化しても，同図（b）に示すネットワークを3つ論理和で結合した非常に大規模なネットワークになってしまう．

このように，変数を用いて異なるオブジェクト間で項目値を比較する場合，それぞれの条件節を表すネットワークをインタノードで直接結合しなければならないというネットワーク表現方法に起因して上記2種の問題点が生ずる．これをインタノードの問題点と呼ぶ．

2.3.2 ネットワーク処理上の問題点

オブジェクトの値が変化した場合，変化後のオブジェクトをネットワークに登録する前に，ネットワーク中に記憶されているそのオブジェクトに関する履歴情報を消去する必要がある．

履歴情報は，ネットワーク内に散在しており，ネットワークのすべての枝に対して，特定オブジェクトに関する履歴情報があるかどうかを調べることは非効率である．そこで，変化前のオブジェクトを再度ネットワークに流し，到達した枝よりそのオブジェクトに関する情報を消去する方法がとられている．この方式では，変化前のオブジェクトを登録した時と同じように，ネットワークに流し，到達した枝から履歴情報を消去し，次に，変化後のオブジェクトを流すことによりネットワークの情報更新を行うことになり，いわば，二度手間の処理であり，効率が悪い．これを，履歴情報消去の問題点と呼ぶ．

3. 高速処理方式

本章では，前記の問題点を解決した新しいネットワークを用いた高速処理方式を提案する．

3.1 高速処理方式の概要

ルールのネットワーク表現法に対しては，変数の多く出現する複雑な条件についても高速処理が可能となるように，候補ノード，merge ノードを導入し，各条件中にある条件節を独立なサブネットとして表現する．

そして，上記サブネットを，無駄な処理が少なくなるように等価変形する．その変換の方針は，(1)同じ条件を表すノードが複数個存在する場合1つにまとめる，(2)ノードの条件のきつさ等の特徴量を基にネットワークを変形する，(3)オブジェクトの持つ排他的，かつ，共通の属性を表す特徴的項目に着目し，ネットワークを分割・統合する，の3点である．

ネットワーク中の履歴情報の更新に関しては，変化前のオブジェクトを再度ネットワークに流すことなく，直接に情報消去を行う．

以上述べた3つの方式を次節以下で説明する．

3.2 ネットワーク表現法

2.3 節で指摘したインタノードの問題点は，ルールの条件節間に変数を用いたインタラクションがある場合，それぞれの条件節を表す部分ネットワーク（以降，1つの条件節を表す部分的なネットワークをサブネットと呼ぶ）を，インタノードで直接結合することに起因している．このため，本高速処理方式では，条件節間にインタラクションが存在しても，サブネットをインタノードで直接結合せず，値の参照される（変数に値を代入した）条件節を表すサブネットに仮想的なノードである候補ノードを間接的に対応付け，この候補ノードを介してサブネットを結合する．

本手法におけるネットワーク生成方法を，図4の

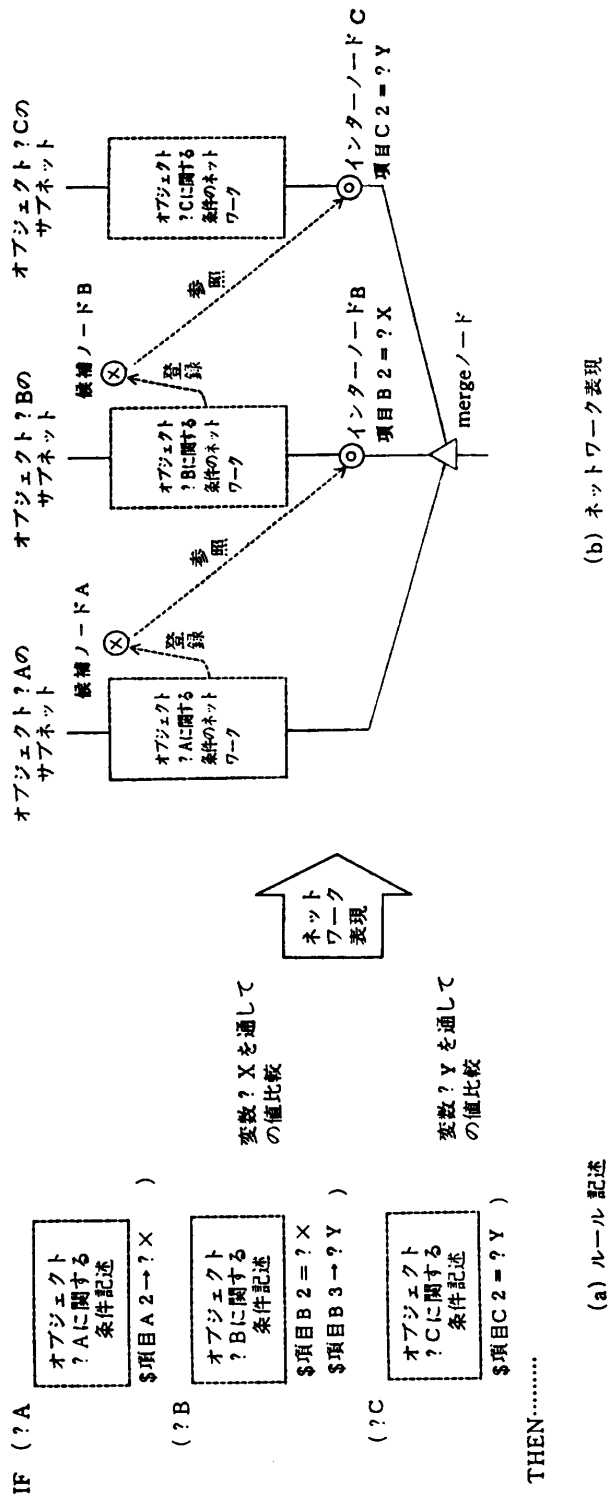


図6 ネットワーク生成法
Fig. 6 Network generation flow.

ルール1と同じ参照関係を持つルールに関し，図6により説明する．このルールは，オブジェクト？A，？B，？Cに関する3つの条件節より構成されており，一番目の条件節では，変数？Xに項目A2の値を代入し，二番目の条件節の項目B2との値の比較に用いている．また，二番目の条件節では，変数？Yに項目B3の値を代入し，三番目の条件節の項目C2との値の比較に用いている．

本手法では，まず，条件節をサブネットにそれぞれ変換する．この例では，3つのサブネットが生成される．二番目の条件節には，変数？Xを用いて，オブジェクト？Aとの項目値の比較が記述されているので，オブジェクト？Bの条件節を表すサブネットには，インタノードが現れる．従来手法では，このような場合，直接オブジェクト？Aの条件節を表すサブネットをインタノードで結合しているが，本手法では，値の参照される条件節を表すサブネット，この例では，オブジェクト？Aのサブネットに仮想的ノードである候補ノードを間接的に対応付け（登録の矢印で示す），インタノードと候補ノードを結合する（参照の矢印で示す）．同様に，値の参照されるオブジェクト？Bのサブネットにも，候補ノードを対応付け，オブジェクト？Cのサブネット内にあるインタノードと結合する．

候補ノードには，対応付けられたサブネットを通過する見通しのあるオブジェクトが記憶される．通過する見通しのあるオブジェクトとは，

（1）インタノードでチェックが失敗する

（2）すべてのノードのチェックが成功するのいずれかを満たすオブジェクトである．つまり，従来のインタノードでは，入力枝から流れてくる情報は，そのサブネットの条件をすべて満たしたものであるのに対し，本表現では，上記(1)，(2)を満たしたオブジェクトを候補オブジェクトとしてサブネットから拾い上げ，インタノードでは，このオブジェクトと比較する．つまり，インタノードは，候補ノードを介して，相手サブネット全体と間接的に結合されていることになる．以下，例を用いて説明する．

図6の例では，オブジェクト？A自身に関する条件を満たしたオブジェクトが，候補オブジェクトとして候補ノードAに記憶される．一方，候補ノードBには，オブジェクト？B自身に関する条件を満たしたオブジェクトが，つまり，インタノードBでチェックが失敗したとしても，候補オブジェクトとし

て，候補ノードBに記憶される．インタノードCでは，この候補ノードBに記憶されている候補オブジェクトと項目値を比較する．候補ノードに新たに記憶された候補オブジェクトは，「参照」の矢印にそって，他サブネットのインタノードへ流される．

例えば，二番目のサブネットのインタノードBにB1というオブジェクトが到達し，そのインタノードの条件が失敗した場合でも，B1は，候補ノードBを経由して，三番目のサブネットのインタノードCに流れていく．つまり，インタノードBで失敗したとしても，失敗の原因は，オブジェクト？Aの条件節を満たしているオブジェクトにあると考え，オブジェクト？Cとの比較を行っておく．

このように，候補ノードを介したインタノードの比較は，従来方式とは異なり，比較対象は，あくまで，条件を満たす可能性のある候補オブジェクトである．このため，各サブネットより流れて来たオブジェクトに関する情報の整合性チェックを行う必要がある．例えば，上記の例で，オブジェクトB1との比較がサブネット？C内で成功し，インタノードCを通過した場合，本当にオブジェクトB1がインタノードBから流れて来ているかをチェックする必要がある．そこで，これを確認するための merge ノードでサブネットを結合し，1つのルールを表すネットワークを生成する．

3.3 ネットワークの変形・統合方法

3.3.1 サブネットの変形

3.2 節で生成されたサブネットをその条件の意味を変えずに変形する．イントラノード，インタノードの制約条件をA，B，…で表し，図 7（a）のサブネットの意味を，ポーランド記法を用いて，

AND A AND OR OR B C D AND E F　　(1)

と表現する．この論理式を項と呼ぶ．項（1）において，制約条件 X が制約条件 Y より左にあることは，サブネットでは，AND 条件の場合，X が Y の上位，OR 条件の場合，X が Y の左に位置することを表す．このサブネットは，上から下へと処理され，OR 条件の場合は，左から右へと処理される．以下，変形の方法について説明する．

まずトップレベルでは，先頭の論理演算子について，可換な項を選択する．項（1）では，A，(OR OR B C D)，E，F の4つの項が可換である．それぞれの項の評価値を計算し，より小さい評価値を持つ項から順に並べる．すなわち，ネットワークの上位に位置すべきノードは，より小さい評価値を持つ．

例えば，評価関数を EV として，

EV(F)<EV(E)<EV(OR OR B C D)<EV(A)

である場合は，

AND F AND E AND OR OR B C D A

のようになる．可換な項が，上記の (OR OR B C D) のように，さらに分解できる場合は，その項に対して前述と同じ操作を行う．

A，B，C，…の評価値は，対象問題を分析し，例えば，EV(A)=1000，EV(F)=10 のようにあらかじめ決めておく．AND，OR を含む項の評価値の計算は，L，Mを項として，

EV(OR L M)=MAX(EV(L), EV(M))+a

EV(AND L M)=EV(L)+EV(M)

（ただし，a は論理和の処理分を表す正定数）

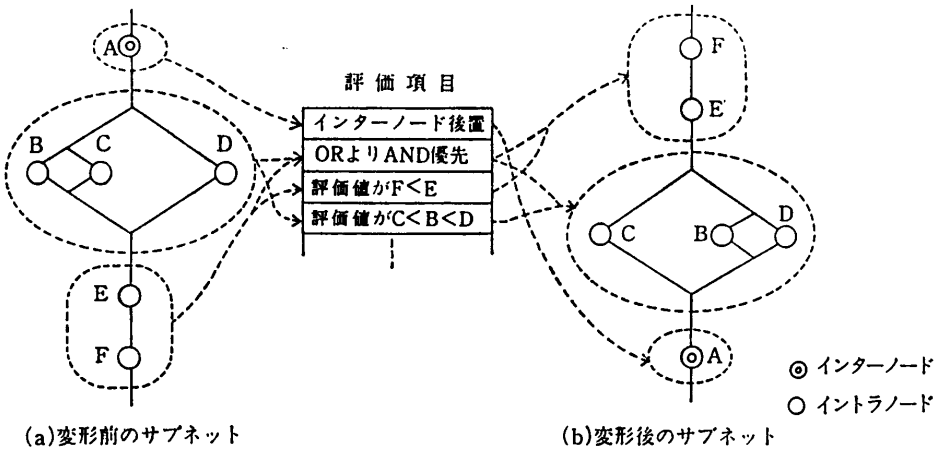


図 7 サブネットの効率化
Fig. 7 Subnet transformation.

のように計算する．例えば，

EV(AND OR A B OR C D)=EV(OR A B)
+EV(OR C D)=MAX(EV(A), EV(B))+a
+MAX(EV(C), EV(D))+a

となる．このような再帰的操作により

AND F AND E AND OR C OR B D A (2)

のように変形され，図7（b）のサブネットが得られる．

一般的な評価関数の設定方針は，インタノードよりイントラノードを優先し，それぞれにおいて，比較演算子の“=”が最も優先度を高くし，次に，“<”，“>”，次に，“≠”のように優先する．例えば，

EV（“=”のイントラノード） = 100
EV（“<”か“>”のイントラノード） = 110
EV（“≠”のイントラノード） = 120
EV（“=”のインタノード） =1000
EV（“<”か“>”のインタノード） =1100
EV（“≠”のインタノード） =1200

のように設定する．図6（b）の「項目C 2＝？Y」は“=”の比較を表すインタノードであるので評価値は 1000 となる．

また，（1）オブジェクトに共通に出現する，あるいは，多くのオブジェクトに出現する，（2）ルールの条件部に多く出現する，（3）項目に関する条件記述が“項目＝定値”である，の条件を満たす項目は，対象の共通かつ排他性を持つ項目であり，特徴的項目と呼ぶ．本高速方式では，次項で示すように，この特徴的項目の共通かつ排他性に注目し，ネットワークを分割・統合することにより ROOT ノードの問題点の解

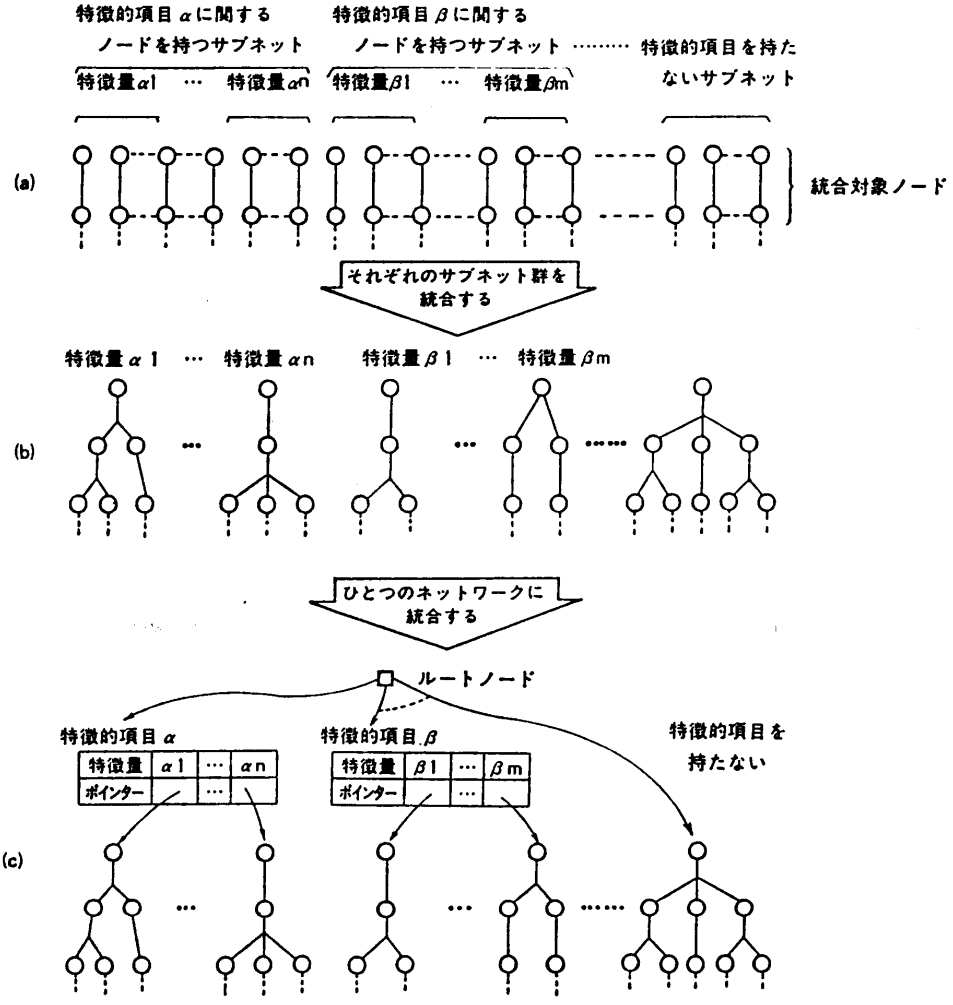


図 8 ネットワークの分割・統合
Fig. 8 Network divide and combine method.

消を図る．このため，特に，特徴的項目に関する条件に対する優先度を高める．

例えば，特徴的項目として，オブジェクト名称，上位オブジェクト名称とした場合，

EV(オブジェクト名称に関する条件) =10

EV(上位オブジェクト名称に関する条件)=20

のように設定する．

3.3.2 サブネットワークの統合

サブネットは，前項の処理により，一般に，その上部にイントラノードが直列に並び，その最上部に特徴的項目に関するイントラノードが位置する形に変形される．このサブネット変形上の性質に基づき，ネットワークの分割・統合を行う．

この処理過程を図8を用いて説明する．まず，サブネットを，特徴的項目 α に関するイントラノードを最上位に持つサブネット群，特徴的項目 β に関するイントラノードを最上位に持つサブネット群，…，特徴的項目に関するイントラノードを持たないサブネット群に分類する．さらに，特徴的項目に関するイントラノードを持つ各サブネット群を，最上位のイントラノードが，"特徴的項目 α=特徴 $\alpha 1$" である群，"特徴的項目 α=特徴 $\alpha 2$" である群，…，"特徴的項目 β=特徴 $\beta 1$" である群，…に細分化する（同図（a））．

つぎに，細分化されたサブネット群ごとに，最上位のイントラノードを除いたサブネットについて同じノードを共通化するように，統合する（同図（b））．

最後に，各特徴的項目ごとにその特徴量と，統合化されたサブネットへのポインタをテーブル化し，このテーブルを特徴量でソーティングし，このテーブルを用いて，各統合化されたサブネット群を図（c）に示すようにルートノードで結合し，ネットワークを得る．

3.4 ネットワーク中の履歴情報の更新法

ネットワーク中の履歴情報更新方式の非効率さを解消する方法としては，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リストとして記憶しておき，それを用いてオブジェクトに関する履歴情報を消去する方法が考えられる．しかし，ネットワークにおいては，新たに入力されたオブジェクトだけが流れるのはまれで，入力されたオブジェクトがトリガになって他のオブジェクトも流れてしまう．つまり，一度に多くのオブジェクトに関するリストを，各オブジェクトごとに生成しながらネットワークを処理する必要がありかえって効率が悪い．また，あるオブジェクトがどのノードに記憶されているのかをすべて記憶するためには，多くの記憶容量を必要とする．

我々の方式において，オブジェクトに関する情報を保持するノードは，候補，インタ，merge，終端の各ノードであり，これらのノードは，ネットワーク変形の過程でネットワーク下部に集中して出現する．また，その出現順序は，

（1）インタ→(インタ→…)merge→終端

（2）merge→終端

（3）終端

の三つのパターンである．そこで，ネットワークに新たに入力されたオブジェクトだけに対して，それぞれの出現パターンの先頭ノードの位置，および，候補ノードに記憶された場合はその候補ノードの位置をリストとして記憶する．

履歴情報を消去するオブジェクトに関する上記リストに記憶されているすべてのノードに対して次の処理を行うことにより履歴情報を消去する．指定されたノードの枝より，そのオブジェクトを含む履歴情報があれば消去し，このノードに結合している次のノードについて同様の処理を行う．オブジェクトを含む履歴情報がなければ，リスト内の別のノードを取り出し，同様の処理を繰り返す．候補ノードについても同様であり，参照の矢印の先のインタノードから上記処理を行う．

4. 推論速度評価

4.1 個別評価

本方式の各々の有効性は，対象問題に依存し，一概に，従来方式と比較して何倍である，と示すことは困難である．そこで，各高速化方式の定性的性質を述べ，実システムにおける測定結果によってそれを確認する．

（1） ROOT ノードを特徴的項目により構造化し，ネットワークを分割することにより，変化オブジェクトを流すべきネットワークを各特徴項目のテーブルのバイナリサーチのみで知ることができる．なお，テーブルに結合しているネットワークは排他性を有するように構成してあるため，各テーブルごとに1つのネットワークだけ処理すればよい．例えば，図8(c)のネットワークに，特徴項目 α の値が，$\alpha 1$ であるオブジェクトを登録する場合，特徴項目 α のテーブルの特徴量が $\alpha 1$ であるポインタに結合しているネットワー

クだけ処理すればよく，他の $\alpha 2, \cdots, \alpha n$ に結合しているネットワークを処理する必要はない．この方式により，図4に示した点線部分Aの処理が大幅に削減される．

（2） ノードの評価値による並べ換えにより，対象問題の性質に合ったネットワークが生成される．例えば，文献[12]に示しているように，ネットワーク処理過程のログデータから，ノードの処理量，条件成立の確率を得，これを評価値として用いれば，イントラノードのみの並べ換えであっても，約2倍の効率化が期待できる．ノードに関して詳細な特徴データが得られない場合でも，前章で例示した一般的な評価関数を用いることにより処理の効率化が図れる．これについては，後述する．

（3） 候補ノード，マージノードの導入により，変数を用いた異なる条件節間の比較においても「変化オブジェクトに関する条件チェックのみをやりなおせばよい」というネットワーク処理の特徴を保持できる．例えば，図4（b）のネットワークにおいて，OBJ-Aが変化した場合，OBJ-A と OBJ-B および OBJ-B と OBJ-C どうしの比較が必要になるのに対し，図6（b）のネットワークでは，OBJ-B と OBJ-C の比較は不要である．

また，図5（a）のルールは，同図（b）に示すように，非常に大規模なネットワークになるが，本論文で示したネットワーク表現法を用いれば，図9に示すようなシンプルなネットワークで表すことができる．この例では，インタノード数が 120 から 12 へと 10分の1に減少する．この減少は比較回数の大幅な減少につながり，処理速度も比例して向上できる．これは，異なる条件節間の比較が論理和の中で現れた場合に発生し，論理和が多用されるほど効果が大きい．なおこのような論理和を含まない場合でもノード数が増加することはあり得ないので効率が低下することはない*．

（4） 変化前のオブジェクトを再度ネットワークに流すことなく，直接，履歴情報を保持しているノードからの消去処理が可能であり，ROOT ノードから履歴情報を保持したノードまでのネットワーク処理を削減できる．

上記各項目が実システムにおいてどのくらい有効で

* 候補ノード，マージノード導入により，「変化オブジェクトのみの再マッチ処理を行えばよい」という性質を生むが，これは，逆に，チェックできるものは，できるだけ事前に処理することであり，事前に処理した結果が無効になる場合は，処理性が悪くなる．

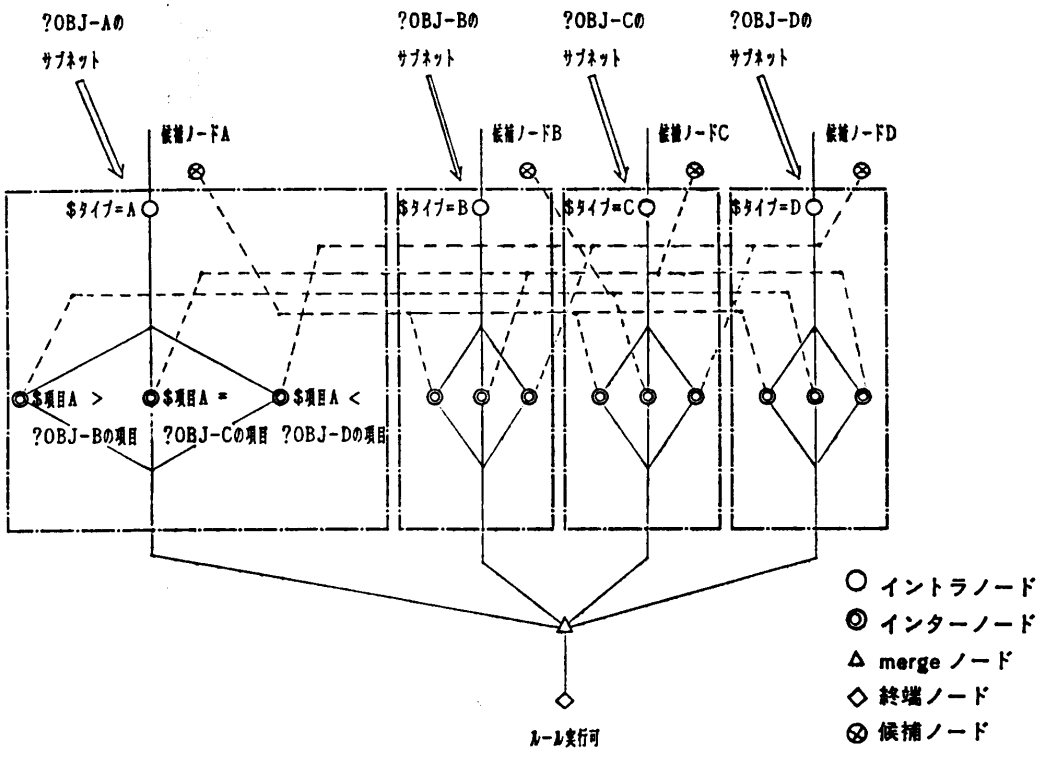


図 9 複雑なルールのネットワーク表現
Fig. 9 Network representation of a complex rule.

あるかを確認するために，約200ルール，100オブジェクトの規模のプロセス診断システムを用いた測定結果を示す．(1)，(2)の有効性は，特徴項目，評価関数に依存するが，測定には，前章で例示した標準的なものを用いた．上記項目(1)および(2)を適用すると，ネットワークへの登録処理時間は，約2分の1以下になった．また，上記項目(4)を適用することにより，ネットワークからの情報消去処理時間は，約3分の1以下になった．(4)を適用した場合，ネットワークの登録処理は，情報消去のためのリスト生成が必要であるため実行スピードが遅くなることが予想されたが，第4章で示したように，記憶するノードを限定したことにより，実行速度の低下は，測定値として観測できないほど小さかった．

上記（3）が有効となる場合は，このプロセス診断システムでは現れていない．そのため，図5（a）のルールおよび40オブジェクトを用いて，RETEアルゴリズムとの比較を行った．RETEアルゴリズムを，EUREKAの記述言語であるC言語で記述し，HIDIC V90/50 (約2 mips) を用いて測定した．この場合，RETEアルゴリズムに比べて，約十倍の推論速度が得られた．

4.2 総合評価

本方式は，履歴情報を用いる手法の本質的目標である「変化したオブジェクトに関する条件の再チェックのみで実行可能ルールを捜し出せる」の達成を目的として，従来手法を拡張している．

そこで，本高速処理方式の上記性質を示すために，まず，ルール・オブジェクトをモデル化し，このモデルに基づきルール数と推論速度の関係を導き，本高速処理方式がこの関係を満たすことを実証してみる．

ルール数と推論速度の関係を解析可能とするために，ルール記述，オブジェクト記述に関して，以下に示す条件を仮定した．

（1）ルール条件部，実行部には，オブジェクトが均等に出現する．

（2）オブジェクト数とルール数の関係は線形の関係を満たす

ルール総数を R 個，オブジェクト総数を N 個，条件部の平均条件節数を C 個，実行部における平均オブジェクト更新数を E 個とする．N と R には，$N=a*R+b$ (a, b は定数) の関係がある．

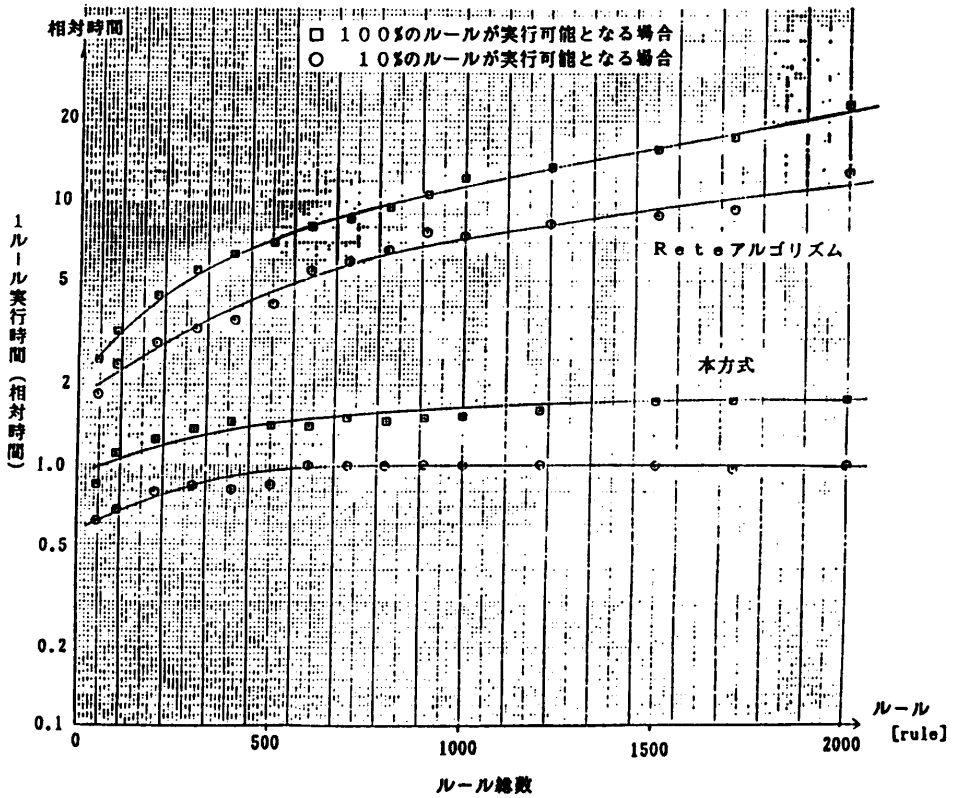


図10 ルール数とルール実行時間の関係
Fig. 10 Relationship between the numbers of rule and rule execution time.

この記述モデルでは，1ルール実行すると，平均E個のオブジェクトを更新することになる．1ルールの平均条件節数がC個であるため，全ルールに含まれる条件節数は，$R*C$個となり，仮定(1)から，このうちのE/N個の条件節の再チェックが必要となる．つまり，1ルール実行すると，平均，$R*C*E/N$個の条件節の再チェックが必要となる．また，オブジェクトとルール数の関係は，線形を仮定しているため，平均$R*C*E/(a*R+b)$個の条件節の再チェックとなる．一般にルール実行に要する時間のほとんどは，実行可能ルールを捜す処理に占められているから，この値は，1ルール実行に要する時間と考えてよい．ルール総数Rを大きくしていくと，再チェック必要な条件節の数は，一定値$C*E/a$に収束し，その結果ルール実行速度も解析上は一定値になると予測できる．

前述のプロセス診断システムにおける上記係数は，$C=2$, $E=1$, $a=0.3$, $b=20$である．そこで，この値を持つルール，オブジェクトを自動生成するプログラムを作成し，得られた50～2000ルールの各ケースについて，1ルール実行に要するCPU時間を測定した．

従来方式との比較を行うために，前述のC言語で記述したRETEアルゴリズムも同様に測定した．測定に用いた機器は，HIDIC V90/50 (約2 mips)である．測定結果を**図10**に示す．

図からわかるように，本方式は，ルール数が増加しても，ルール実行速度は，一定になり，解析結果と一致している．一方，RETEアルゴリズムは，ルール数の増加に従い，急速に推論速度が低下する（グラフは片対数で表してある）．

これから，本方式は，履歴情報を用いる手法の本質を大規模なルール記述においても実現していることが証明できる．また，上記モデルにおける推論速度は，100～140 rule/sec程度であり，十分高速である．また，上記のプロセス診断システムにおいても，同等の推論速度が得られることを確認している．

なお，実システムの適用においては，モデルの各定数の違い，ルール記述の複雑さにより，推論速度は大きく変化する．

5. おわりに

本論文では，推論の高速処理方式を提案した．履歴情報を持つ弁別ネットを用いることを基本とし，候補ノード，mergeノードを導入した新しいネットワーク表現を提案した．加えて，ネットワーク変形法，ネットワーク中の履歴情報更新法を用いることにより，さらに効率化した．

また，ルール，オブジェクト記述をモデル化し，このモデルにおいて，ルール数と推論速度の関係を明らかにし，本高速処理方式の基本的性質である「変化オブジェクトに関する条件の再チェックのみで実行可能ルールを捜し出せる」を確認した．

なお，本方式は，HIDIC V90/25および，エンジニアリングワークステーション ES-330 上の EUREKA-Ⅱに適用されている．

謝辞 本研究の場，およびその方向付けを与えて下さった，(株) 日立製作所システム開発研究所 川崎淳所長，同春名公一部長，ならびに，有益な御意見をいただいた日立製作所大みか工場 林利弘部長，篠本学主任技師に深く感謝いたします．

参考文献

1) Feigenbaum, E. A.: The Art of Artificial Intelligence: Themes and Case Studies of Knowledge Engineering, *IJCAI 77*, pp. 1014-1029 (1977).
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) Konolige, K.: An Inference Net Compiler for the Prospector Rule-Based Consultation System, *IJCAI 79*, pp. 487-489 (1979).
4) Mark, W.: Rule-Based Inference in Large Knowledge Bases, *AAAI Aug. 1980*, pp. 190-194 (1980).
5) 鶴田，能見，宮本：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).
6) Forgy, C. L.: Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, *Artif. Intell.*, Vol. 19, pp. 17-37 (1982).
7) Anderson, J. R.: *Human Associative Memory*, pp. 69-77, Hemisphere Publishing Corporation, Washington, D. C. (1974).
8) Barsalou, L. W.: Discrimination Nets as Psycological Models, *Cognitive Science*, Vol. 8, pp. 1-26 (1984).
9) 増位，田野，舩橋：知識制御核ソフトウエア

EUREKA の記述体系，第 29 回情報処理学会全国大会論文集，pp. 1373-1374 (1984).
10) 田野，増位，舩橋：知識処理型ソフトウエア 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論文集，pp. 993-994 (1985).
11) 田野，増位，坂口，舩橋：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).
12) 田野，増位，舩橋：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).
13) 荒屋，百原，田町：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).

(昭和 62 年 4 月 1 日受付)
(昭和 62 年 11 月 11 日採録)

田野 俊一 (正会員)

昭和 33 年生．昭和 56 年東京工業大学工学部制御工学科卒業．昭和 58 年同大学院総合理工学研究科システム科学専攻修士課程修了．同年，(株)日立製作所に勤務．同システム開発研究所所属．主として，人工知能，知識工学の研究に従事．人工知能学会，計測自動制御学会各会員．

増位 庄一 (正会員)

昭和 25 年生．昭和 47 年京都大学工学部電子工学科卒業．昭和 49 年同大学工学研究科電気工学第 2 専攻修士課程修了．同年(株)日立製作所入社．システム開発研究所にて制御関連および知識工学の研究に従事，現在同所主任研究員．昭和 56 年 8 月より一年間，カーネギメロン大学客員研究員．計測自動制御学会，人工知能学会，電気学会，IEEE，AAAI 各会員．

坂口 聖治

昭和 37 年生．昭和 55 年山口県立下関工業高等学校電子科卒業．同年，(株)日立製作所に入社．同システム開発研究所所属．制御関連および知識工学の研究に従事．言語変換系の自動生成系に興味をもつ．

舩橋 誠壽 (正会員)

昭和 19 年生．昭和 42 年京都大学工学部数理工学科卒業．昭和 44 年同大学院修士課程修了．同年(株)日立製作所入社，中央研究所を経て 48 年よりシステム開発研究所にてシステム制御の研究に従事．50，51 年 MIT，スタンフォード大学客員研究員．電気学会，AAAI など各会員．