論　文

プロダクションシステムにおけるジョイン演算の順序に関する一考察

南保　英孝[†]　　木村　春彦[†]　　広瀬　貞樹[††]

A Consideration on Ordering of Join Operations for Production Systems

Hidetaka NAMBO[†], Haruhiko KIMURA[†], and Sadaki HIROSE[††]

あらまし　エキスパートシステムはルール化した専門家の知識とデータを照合することによって，推論を行うシステムである．このエキスパートシステム構築用のツールとしてよく用いられるものにプロダクションシステムがある．現状では，ルールとデータの照合に時間がかかるため，条件照合の高速化がプロダクションシステムの大きな課題となっている．本論文では，ルールの条件部の条件要素の照合順序を変えると，条件照合にかかる時間が変化することに着目し，知識ベースより得られる静的情報のみから照合回数を確率的に予測し，条件照合の効率改善を図る手法を提案する．また，提案手法が効果を発揮するルールのクラスを明らかにし，そのクラスに入るルールに対して提案手法を用い，その他のルールに対しては従来の Rete アルゴリズムを用いる条件照合アルゴリズムを示す．更に，そのアルゴリズムが実際にどの程度有効であるかを実験によって明らかにする．

キーワード　ジョイン演算，照合回数，プロダクションシステム

1. まえがき

専門家の代行を務めるシステムにエキスパートシステムがある．このエキスパートシステム構築用のツールとして，プロダクションシステムが広く用いられている．プロダクションシステムは，知識を“IF～THEN～”の形のルールで表現し，ルールとデータの照合によって推論を行っていくルールベースシステムである．このように，プロダクションシステムは人間の知識を表現するのになじみがよいが，条件照合速度が遅いという問題がある．

これまで，条件照合の高速化に関するさまざまな研究がなされており，多くの高速化の手法が提案されている[1]～[11]．中でも，Rete アルゴリズム[2]と呼ばれる条件照合アルゴリズムは有名であり，現在最も多く使用されている．Rete アルゴリズムは，ルールの条件部を Rete ネットワークと呼ばれるデータフローグラフに変換し，ネットワークにデータを入力することで条件照合を行っている．ルールをネットワークに変換するとき，ルールの条件要素の順序が変化するとネットワークも変化し，条件照合にかかる時間が変化する．本論文ではこの性質を利用して，効率の良いジョイン演算の順序を確率的に予測し，条件照合の速度改善を図る．ここで，ジョイン演算とは条件要素を満足するデータ間の無矛盾性のチェックのことである．

これまでに，知識ベースを実行して得られた動的情報を基にして条件照合にかかる時間を改善する手法が提案されている[12]．それに対し，本提案手法ではルールを実行せずに，静的情報のみから条件照合にかかる時間を改善する．動的情報を用いる必要がないため，より速やかな処理を行うことが可能となる．また，提案手法で必要となる処理は自動的に行われるので，知識ベース記述に関して経験の少ないユーザが効率の悪いルールを記述した場合でも，実行時間の増加を未然に防ぐことができる．以下に，**2.** で Rete アルゴリズムについて説明を行い，**3.** でジョイン演算の順序決めの方法を示し，**4.** でこの方法が効果を発揮するルールのクラスを示す．**5.** では，これらを考慮した条件照合アルゴリズムを提案し，**6.** でこのアルゴリズムの有効性を実験により示す．

2. Rete アルゴリズム

プロダクションシステム OPS5 [3] では，条件照合アルゴリズムに Rete アルゴリズムが用いられている．本章では，Rete アルゴリズムと関連する用語について

[†] 金沢大学工学部電気・情報工学科，金沢市
Faculty of Engineering, Kanazawa University, Kanazawa-shi, 920 Japan
[††] 富山大学工学部知能情報工学科，富山市
Faculty of Engineering, Toyama University, Toyama-shi, 930 Japan

説明する．

次式はOPS5の一般的なルール形式である．

$$(r(C_1)\cdots(C_n) \rightarrow (\text{actions}))$$

ここで，r はルール名，C_i は条件要素である．また，C_1〜C_n をまとめて条件部と呼ぶ．Reteアルゴリズムでは，ルールの条件部をReteネットワークと呼ばれるデータフローグラフに変換し，データをトークンとしてネットワークに入力することで条件照合を行っている．Reteネットワークには α メモリと β メモリの2種類の中間メモリがある．α メモリには，一つの条件要素内で行われるイントラコンディションテスト（条件要素を満足するかどうかのテスト）を満足したトークンが記録される．また，β メモリには条件要素間で共通の変数の値の無矛盾性を調べるジョイン演算をパスしたトークンの組が記録される．最終的には，ルールの条件部のすべての条件を満たしたトークンの組（インスタンシエーションと呼ぶ）が，CSメモリと呼ばれるメモリに記録される．

3. 提案するジョイン演算順序

前述したように，OPS5では条件照合アルゴリズムにReteアルゴリズムが用いられている．Reteアルゴリズムはルールの条件要素の順にジョイン演算を行うが，条件要素の順序を変化させるとジョイン演算の実行時間も変化することが知られている．本論文ではこの点に着目し，知識ベースの静的情報をもとに，ジョイン演算の順序を変えることによりReteアルゴリズムを改善する手法を提案する．

提案するジョイン演算順序は次の手順で決定される．

（1）知識ベースを解析し，静的情報を取得する．
（2）各ルールのReteネットワーク上の照合回数を予測する．
（3）各ルールについて，ジョイン演算順序を決定する．

以下，各処理について説明する．

3.1 静的情報の取得

Reteネットワークにおける条件照合回数の予測に必要な静的情報とそれらの取得方法について述べる．予測の際に必要となる静的情報には，ワーキングメモリエレメント（以下WME）のクラスの分布，属性値の種類の二つがある．

3.1.1 WMEの分布

ルールの実行の際にはデータであるWMEが生成または，削除される．WMEの先頭にはクラス名がついており，クラス名ごとのWMEの生成，削除の割合をWMEのクラスの分布と呼ぶ．本論文では，WMEのクラスの分布を得るためにWMEが生成，削除される頻度を調べている．具体的には，ルールが1度しか発火しないという仮定に基づき，ルールの記述から静的に分布を導き出す．つまり，WMEは各ルールの行動部で`make`，`remove`，`modify`命令により生成，削除されるので，各命令が対象とするWMEのクラス名の統計をとることで生成，削除の頻度を得ている．このようにして得られる分布は，実際の分布と必ずしも一致するとは限らない．本論文では，**4.** で述べる発火木とルールネットワークを使って，以上のようにして求めたWMEの分布を用いても成果が出るルールを判別し，そのようなルールのみを提案方式の対象としている．また，判別されたルールは，WMEの分布の変化に無関係に照合回数を予測できる．WMEのクラスの分布は，α メモリや β メモリに記録されるトークンの量を計算するのに必要となる．正確な分布は必要がなく，Reteネットワークに入力される各条件要素ごとのWMEの比がわかれば十分である．

3.1.2 属性値の種類

属性値の種類とは，あるクラスのある属性の値がどのような値をとり，その値は全部で何種類になるか，ということを意味している．この属性値の種類は，イントラコンディションテストの成功率や，ジョイン演算の成功率の計算に必要となる．属性値の種類の取得は，WMEのクラスの分布を得るときと同じように知識ベースの各ルールの行動部と条件部の条件要素の属性値を参照する．なお，属性のとり得る属性の種類はそのとり得る値を一様にとるものと仮定している．

3.2 Reteネットワーク上の照合回数の予測

WMEのクラスの分布，属性値の種類の二つの静的情報を得た後で，Reteネットワーク上の条件照合の回数を確率的に求める．基本的には，次の四つのフェーズからなる．

（1） ReteネットワークにWMEが入力されたWME（トークンと呼ばれる）が，対応する条件要素を満足する確率，つまり，イントラコンディションテストが成功する確率を求める．
（2） 変数の無矛盾性をチェックするジョイン演算で，ジョイン演算に成功する確率を求める．
（3） 各ノードに記録されているトークンの数を推定する．

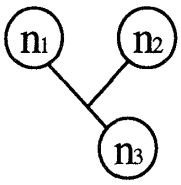


図1 基本的なネットワーク
Fig. 1 A primitive network.

(4) ジョイン演算の回数とジョイン演算をパスして次のノードに至るトークン数を求める.

図1に示すような基本的なネットワークを例に計算方法を説明する.ここで,図中のノード n_1 は Rete ネットワークでの α または β メモリに対応する.また,ノード n_2 は α メモリに対応する.ノード n_3 は β メモリに対応する.一般の Rete ネットワークは,図1のネットワークの組合せからなっている.なお,ノード n_1, n_2 に入力される正の(WME の追加を表す)トークンの数を m_1, m_2, 負の(WME の削除を表す)トークンの数を d_1, d_2 とする.ノードが α メモリに対応している場合,これらの値は WME のクラスの分布より決まる値である.

3.2.1 イントラコンディションテストの成功率

イントラコンディションテストとは,Rete ネットワークに入力されたトークンが,対応する条件要素を満足するかどうかのチェックである.

今,以下のような条件要素を考える.

```
(class ^attribute '属性値の条件')
```

"属性値の条件"には属性値,変数,述語を伴う表現,選言,連言等の場合がある.それぞれの場合について,入力されたトークンがイントラコンディションテストを満足する確率を計算する.具体的には,条件要素の属性に対応するクラスの属性の属性値の種類と属性値の条件が満足する属性値の種類を比較する.例えば,属性の属性値の種類が n 種類,属性値の条件が満足する属性値の種類が k 種類であれば成功率は k/n となる.

3.2.2 変数の一致する確率

変数の一致する確率とは,ジョイン演算で無矛盾チェックの対象となる変数が同じ値をとって一致し,ジョイン演算をパスする確率のことである.図1を用いて説明する.今,n_1, n_2 のジョイン演算で無矛盾チェックの対象となる変数を $V_1, V_2, \cdots, V_n$ とする.また,変数 V_i $(1 \leq i \leq n)$ がノード n_j でとり得る値の集合の位数を $W_{i,j}$ とする $(j = 1, 2)$.ここで,属性の属性値がとり得る値の分布は一様分布であると仮定すると,変数 V_i がノード n_1, n_2 でそれぞれ同じ値 T をとって一致する確率は,

$$\frac{1}{W_{i,1}} \times \frac{1}{W_{i,2}} = \frac{1}{W_{i,1} \cdot W_{i,2}}$$

更に,変数 V_i の値のうち,n_1, n_2 で共通に現れる属性値を調べ,変数が一致する組合せ数を得る.共通の属性値が C_{Vi} 種類となったとき,変数 V_i がジョイン演算において一致する確率は $C_{Vi}/W_{i,1} \cdot W_{i,2}$ となる.他の変数についても同様に考えると,変数 $V_1, V_2, \cdots, V_n$ がすべて同時に一致する確率,つまり,ジョイン演算が成功する確率は,

$$\frac{C_{V1}}{W_{1,1}W_{1,2}} \cdot \frac{C_{V2}}{W_{2,1}W_{2,2}} \cdots \frac{C_{Vn}}{W_{n,1}W_{n,2}} = \prod_{k=1}^{n} \frac{C_{Vk}}{W_{k,1}W_{k,2}} \quad (1)$$

となる.

3.2.3 各ノードに記録されているトークンの数

イントラコンディションテストが成功する確率と,変数が一致してジョイン演算が成功する確率を計算した後,各ノードに記録されるトークンの数を計算する.再度,図1を用いて説明する.

まず,ノード n_i が α メモリに対応する場合を考える.また,ノード n_i におけるイントラコンディションテストの成功率がそれぞれ I_i $(i = 1, 2)$, ジョイン演算の成功率が r となったとする.ノード n_i に入力されるトークン数を m_i とする.ノード n_i は α メモリであるので,入力されたトークンに対してイントラコンディションテストが行われる.イントラコンディションテストを満足し,ノード n_i に記録されるトークンの数 M_i は,

$$M_i = m_i I_i \quad (i = 1, 2) \quad (2)$$

となる.このとき,ノード n_3 に記録されるトークンの数 M_3 は,n_1, n_2 に記録されているトークンの組合せ中で,ジョイン演算を通過する組の数となる.トークンの組合せ数は $M_1 \times M_2$ なので,

$$M_3 = rM_1M_2 \quad (3)$$

がノード n_3 に記録されるトークン数となる.

ノード n_1 が β メモリに対応する場合には，n_1 に記録されるトークン数 M_1 は n_1 の直前のジョイン演算を通過したトークン数となる．よって，式 (3) と同様にして，

$$M_1 = r' M_0 M_0'$$

となる．ここで，r' は n_1 の直前のジョイン演算の成功率であり，M_0，M_0' は n_1 の直前の二つのノード（n_3 に対する n_1，n_2 の関係にあるノード）に記録されるトークン数である．

3.2.4 ジョイン演算で必要となる照合回数

次に，Reteネットワークを用いて照合作業を行うときに，ジョイン演算時に必要となる照合回数を計算する．この計算は，WMEが追加される場合と削除される場合に分けて考える必要がある．

（1） WMEが追加される場合

WMEが追加されたときには，そのWMEを表すトークンがReteネットワークに入力される．今，ノード n_1 にトークンが一つ入力された場合を考える．このトークンがイントラコンディションテストを満足し n_1 に記録される確率は I_1 である．そして，ノード n_2 との間のジョイン演算が行われる．そのときの照合回数は，n_1 に記録されているトークンの変化分（$1 \times I_1$）と n_2 に記録されているトークンの数（M_2）との積となる．よって，

$$1 \times I_1 \times M_2 = I_1 M_2 \tag{4}$$

となる．同様に，n_2 に対応するトークンが一つ追加されたときの照合回数は，

$$1 \times I_2 \times M_1 = I_2 M_1$$

となる．ノード n_1，n_2 に入力されるトークンの割合は，それぞれ M_1，M_2 であるため，追加時に図 1 のネットワークで必要となる照合回数 MT_a は，

$$\begin{aligned} MT_a &= M_1 \times I_1 M_2 + M_2 \times I_2 M_1 \\ &= M_1 M_2 (I_1 + I_2) \end{aligned} \tag{5}$$

となる．

また，ジョイン演算をパスして次のノードに入力されるトークンの組の数 M_{next_a} は，ジョイン演算で行った照合のうち，変数が一致したトークンの組の数となる．よって，

$$M_{next_a} = r \times MT_a \tag{6}$$

となる．

（2） WMEが削除される場合

WMEが削除されるときには，マイナストークンがReteネットワークに入力される．マイナストークンが入力された場合には，それぞれのノードで削除の対象となるWMEを探すための照合が行われる．その照合は，ノード内のすべてのトークンを調べる作業であり，照合回数はノード内のトークン数となる．

n_i にマイナストークンが入力される割合を d_i とする $(i = 1, 2)$．n_1 に入力された場合には，n_1 と n_3 で削除の照合が行われ，n_2 に入力された場合には n_2 と n_3 で照合が行われる．よって，WMEの削除時に必要となる照合回数 MT_r は，式 (2)，式 (3) より，

$$\begin{aligned} MT_r = {} & d_1 (M_1 + r M_1 M_2) \\ & + d_2 (M_2 + r M_1 M_2) \end{aligned}$$

となる．

また，次のノードに入力されるマイナストークンの数 M_{next_r} は，

$$M_{next_r} = d_1 + d_2$$

となる．

よって，最終的に図 1 で必要となる照合回数 MT は，

$$MT = MT_a + MT_r$$

で表される．

[例 1] 次のルール **rule** を用いて，具体的な計算方法を説明する．なお，このルールをReteネットワークに変換したものを図 2 に示す．簡単のため，すべての属性の属性値のとり得る値は 1〜10 までの整数 10 種類とする．またWMEの追加に関するクラスの分布は **cl1** : **cl2** : **cl3** = 10 : 20 : 30 とし，削除はないものとする．

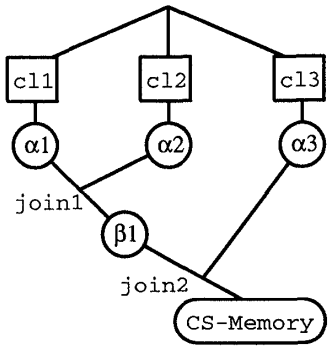


図 2 ルール rule の Rete ネットワーク
Fig. 2 A Rete network of rule.

```
(p rule
  (cl1 ^at11 <A> ^at12 { > 5})
  (cl2 ^at21  8  ^at22 { <A> < 7})
  (cl3 ^at31 <B> ^at32 { <A> << 1 2 >>})
-->
  (actions))
```

まず，イントラコンディションテストの成功率を計算する．クラス cl1 の条件要素では，属性 at11 の属性値は変数であり，必ず条件が満たされる．また，属性 at12 では述語 ">" が使用されており，この条件を満たす属性値は 6，7，8，9，10 の 5 種類となる．よって，クラス cl1 の条件要素についてのイントラコンディションテストが成功する確率は 5/10 となる．クラス cl2 の条件要素では，属性 at21 の条件が満たされる確率は 1/10，属性 at22 の条件が満たされる確率は 6/10 であり，イントラコンディションテストの成功率は 6/100 となる．クラス cl3 では，属性 at31 は変数，属性 at32 は 1，2 のいずれかのときに条件を満足する．よって，イントラコンディションテストの成効率は 2/10 となる．

次にジョイン演算の成功率を計算する．join1 では，変数 ⟨A⟩ についての無矛盾チェックが行われる．cl1 の条件要素では，変数 ⟨A⟩ は 10 種類の値をとる可能性があり，cl2 では 1〜6 の 6 種類の可能性がある．また，共通の属性値は 1〜6 の 6 種類であるので，成功率は式 (1) より 1/10 となる．そして，変数 ⟨A⟩ の値の種類は 1〜6 の 6 種類に制限される．同様にして，join2 でも変数 ⟨A⟩ のチェックが行われ，属性値の種類は cl2 までては 1〜6 の 6 種類，cl3 では 1，2 の 2 種類である．共通の属性値は 1，2 であるので，ジョイン演算の成功率は 1/6 となる．

次に各ノード内のトークン数を予測する．イントラコンディションテストの成功率と式 (2) より，$\alpha_1 : \alpha_2 : \alpha_3 = 5 : 1.2 : 6$ となる．また，β_1 のトークン数はジョイン演算の成功率と式 (3) から，0.6 となり，CS メモリのトークン数は 0.6 となる．

最後に Rete ネットワークにおける照合回数を計算する．式 (4) より，クラス cl1 の WME が追加されたときは，join1 で 1.2 回の照合が必要となる．また，join1 をパスし join2 へ至るトークン数は，式 (6) より 0.12 となる．よって，join2 では 0.72 回の照合が必要となる．同様にして，cl2 に追加されたときは join1 で 5 回，join2 で 3 回の照合が，cl3 に追加されたときは join2 で 0.6 回の照合が必要となる．

各 α メモリに入力されるトークン数の分布を考慮して，最終的な照合回数は，

$$5 \times 1.92 + 1.2 \times 8 + 6 \times 0.6 = 22.8$$

となる．

3.3 ジョイン演算順序の決定

条件部の条件要素の順序を換えることでジョイン演算順序を変化させながら，そのときの Rete ネットワークでの予測照合回数を求めていく．すべての Rete ネットワーク中で最も照合回数が少なくなったジョイン演算順序を最も効率の良い順序として，ルールの条件部をそのときの条件要素に並べ換える．

その際，次のことに注意する．本論文で対象としている OPS5 の競合解消戦略には LEX と MEA があり，LEX ではルール条件部のすべての条件要素を演算順序の変更の対象にできるのに対し，MEA では主に先頭の条件要素を満足させた WME のタイムタグの大小関係で競合解消を行うため，先頭の条件要素の位置を変えられず，第 2 番目以降の条件要素が対象となる．つまり，競合解消戦略を判別することで，演算順序の変更の対象範囲を切り換える．

また，条件要素のすべての順列を考えていたのでは，組合せ数が多く計算コストが大きくなるので，N 番目までの条件要素を並べ換えることを考える．

更に OPS5 では，「条件を満足する WME が存在しない」という条件を表す負の条件要素が使用できるが，提案方式では簡単のため負の条件要素は演算順序の変更の対象から外した．よって，N 番目までの条件要素を決定するが，N の最大値は，最初に現れる負の条件要素の一つ前までとなる．N が大きいほど並べ換えに要する計算量は増えるが，並べ換えた結果はより効率がよくなることになる．本論文では，N を改善レベルと呼ぶことにする．

4. 改善するルールのクラス

本論文で提案するジョイン演算順序の決定方法では静的情報のみを用いているが，この情報は各ルールが 1 度しか発火しないという仮定に基づいて得られた情報である．そのため複数回発火するルールが含まれている場合には，必ずしも効果を発揮するとは限らない．そこで，提案手法が効果を発揮するルールとそうでないルールのクラス分けをする．

以下に，ルールネットワークと発火ネットワークを

用いたルールのクラス分け手法を提案する．

4.1 ルールネットワーク

ルールネットワークとは，ルールが発火したときにどのルールの Rete ネットワークにトークンが入力されるかを表したものである．ルールネットワークを生成するには，まず各ルールに対応するノードを設ける．次にノード間に弧を設けるが，ノード r_1 からノード r_2 に向かう弧を設けるのは，ノード r_2 に対応するルール r_2 の条件部とノード r_1 に対応するルール r_1 の行動部に共通のクラスが含まれるとき，かつそのときに限る．例を用いて説明する．

[例 2] 以下の A～E の五つのルールを考える．

```
(p A(cl1)(cl2)(cl3)-->(make cl2)(make cl4))
(p B(cl2)(cl3)-->(make cl1))
(p C(cl4)-->(make cl3))
(p D(cl3)-->(make cl5))
(p E(cl5)-->(make cl3)(make cl5))
```

まず各ルールに対応して A～E の五つのノードを設ける．次に，例えばルール A の条件部にクラス cl1 の条件要素があり，ルール B の行動部にクラス cl1 の WME を生成する命令があるので，ノード B からノード A に向かう弧を設ける．すべてのルールの条件部にあるクラスについて同様のことを行うと，結果として図 3 のようなルールネットワークが生成されることになる．

4.2 発火ネットワーク

発火ネットワークは，あるルールが発火したとき次にどのルールが発火するかを表したネットワークである．このネットワークを用いて複数回発火するルールを判別する．発火ネットワークを生成するには，知識ベースを実際に実行してルールの発火順序を利用するのが一番望ましいが，提案手法ではルールを擬似的に発火させることで知識ベースの実行をシミュレートし，そのときに得られたルールの発火木を有向グラフに変換することにより発火ネットワークを生成する．発火木は以下の手続きで生成される．

[発火木の生成手続き]

（1） 初期化ルールを発火または初期 WME を読み込んで初期 WME をワーキングメモリ（以下 WM）に追加する．また，初期化ルール（初期化 WME）に対応する根ノードを設ける．

（2） 擬似的な条件照合（属性値が定数の属性についてのみの照合）を行い，発火可能となるルールを調べる．実際の条件照合では変数の値のチェックを行うが，ここでは変数は無視する．

（3） 発火可能となったルールを一つ発火させ，そのときの発火順序を記録する．また，一つ前に発火したルールに対応するノードを親ノードとし，今発火したルールに対応するノードを子ノードとして設け，親ノードから子ノードに辺を設ける．但し，そのときの発火順序が既存の発火順序に含まれている場合は発火させない．発火可能なルールがなければ（5）へ飛ぶ．

（4） 発火したルールの行動部を実行する．実行時には変数の値等は考えずそのまま実行する．実行後は再び条件照合を行うため（2）へ戻る．

（5） 一つ前に発火したルールが初期化ルール（または最初に発火したルール）であれば（7）へ飛ぶ．そうでなければ（6）へ飛ぶ．

（6） 一つ前に発火したルールに戻り，そのルールの発火直後の状態に戻す（バックトラックを行う）．但し，生成したノードは削除しない．（3）へ飛ぶ．

（7） 終了．

[発火ネットワークの生成手続き]

上記の手続きにより生成された発火木に対し，親ノード v_i から子ノード v_j への辺を弧 (v_i, v_j) に変換することにより有向グラフを生成する．これが発火ネットワークである．

[例 3] 例 2 の A～E のルールに対して，上記の手続きを実行した結果，図 4 のような発火木が得られたとする．これを有向グラフに変換することにより図 5 のような発火ネットワークが得られる．

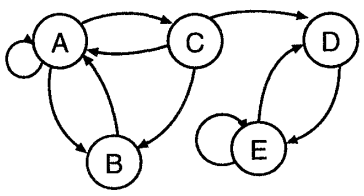


図 3 ルールネットワークの例
Fig. 3 An example of rule network.

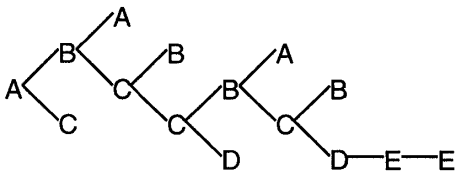


図 4 発火木の例
Fig. 4 An example of fire tree.

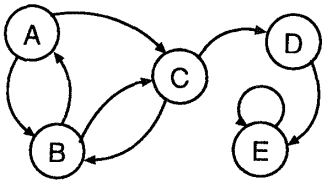


図5 発火ネットワークの例
Fig. 5 An example of fire network.

4.3 ルールのクラス分け

生成されたルールネットワークと発火ネットワークを用いることにより，提案手法が効果を発揮するルールかどうかの判別ができる．以下にその判別方法を示す．

[ルールのクラス分け手続き]

(1) 発火ネットワーク中の各ノードに対して閉路をもつかどうかをチェックする．閉路をもっているノードに対応するルールは複数回発火するものと考える．

(2) 複数回発火すると判断されたルールに対応するルールネットワーク中のノードに印を付ける．そして，印を付けたノードから距離1で到達できるノードに対応するルールは提案手法の対象外とする．このルールを対象外とするのは，複数回発火するノードより得られた静的情報（WMEのクラスの分布）を使用して条件照合回数の予測計算を行うため，予測照合回数の結果の正当性が保証されないためである．但し，上記の条件から提案手法の対象外と判断されたルールであっても，そのルールに対応するノードの入り次数が1である場合には提案方式の対象のルールとする．なぜならば，そのルールの照合回数の予測で用いられる静的情報はある一つのルールの発火によってしか生成されず，複数回発火した場合であってもWMEのクラスの分布の比は正確である．よって，条件要素の順序を変化させながら照合回数を計算した場合，求められた照合回数の大小関係は正しいからである．

[例4] 図5において閉路をもつノードはA，B，C，Eである．更に，図3においてこれらのノードより距離1で到達可能なノードはA，B，C，D，Eである．但し，Cについてはそのノードの入り次数が1であるため提案方式の対象ルールであると判断する．

5. 提案する条件照合アルゴリズム

提案する条件照合アルゴリズムの概要は以下のようになる．ルールネットワークと発火ネットワークを用いて提案方式の対象となるルールと対象外のルールを判別する．提案方法の対象と判別されたルールについては，そのルールのReteネットワークのジョイン演算順序を変えることで効率改善を図る．また，対象外と判別されたルールについては，従来のReteネットワークをそのまま用いる．

以上のようにして知識ベース内で提案方法が効果を発揮するルールのジョイン演算順序を変えることで，そのルールのReteネットワークで必要となる予測照合回数が少なくなる．また，条件照合にかかる時間と照合回数がほぼ比例関係にあることから，ジョイン演算順序を変更したルールの条件照合にかかる時間が短縮される．一方，従来のReteネットワークをそのまま用いるルールに関しては，条件照合にかかる時間に変化はない．ルールネットワークと発火ネットワーク生成に要する手間については，発火木生成に最も時間がかかる．発火木生成ではルールを擬似的に発火させるが，実際にジョイン演算をしているわけではなく，イントラコンディションテストも完全に行っているわけではない．プロダクションシステムでは条件照合処理のために全実行時間の約90～98%が費されている[13]という報告にもあるように，イントラコンディションテストや，とりわけジョイン演算に要する計算コストが大きい．それも高コストルールになればなるほど大きくなる．それ故，ルール判別に要する手間はルールが高コスト状態に陥るとき，軽減される照合時間よりも小さくなることが予測される．

6. 実　　験

6.1 実験方法

実際の知識ベースに提案手法を適用し，条件照合にかかる時間がどの程度改善されるかを測定した．OPS5の知識ベースに対して，提案した方法によりルール内の条件要素の順番を変える最適化処理を行った後にコンパイルしたものと，最適化処理を行わずにコンパイルしたものの照合時間を測定し比較した．なお，改善レベル N は1～10まで変化させて実験を行った．

用いた知識ベースを以下に説明する．

(1) 知識ベースA

この知識ベースは次の問題を解く．

『長さと形状が異なるいくつかのパイプ部品がある．また，蛇口が三つある．三つの蛇口を各々180 cm以上離してパイプで接続したい．但し，蛇口間のパイプは部品を四つ組み合わせて作成する．また，部品や蛇口の両端の形状によって接続できる部品の型や限界の長

さが決まっている.』

なお，この知識ベースのルール数は 10 である．1 ルール当りの平均条件要素数は 2.8 であり，最大条件要素数は 7 である．また，提案手法の対象となるルールは二つあり，それぞれの条件要素数は 7 と 3 である．

（2） 知識ベース B

この知識ベースは次の問題を解く．

『20 人の人が 4 人ずつのグループに分かれて円卓に座る．但し，各人にはそれぞれ二つの趣味があり，また同じ卓にはなりたくない嫌いな人が 1 人いる．嫌いな人と同じ卓にならず，かつ，両隣の人と共通の趣味をもつように座るにはどのようにグループ分けをすればよいか.』

なお，この知識ベースのルール数は 16 である．1 ルール当りの平均条件要素数は 3.75 であり，最大条件要素数は 20 である．また，提案手法の対象となるルールは一つあり，その条件要素数は 20 である．

（3） 知識ベース C

この知識ベースは次の問題を解く．

『30 個の荷物と 1 から 9 までの番号がついた容器がある．各荷物には容器番号と荷物の重さが付加されている．また容器には中に入る最大の重さが決まっており，4 個の荷物が入る．九つの容器のうちの三つを使って荷物を移動させるとき，最大の重量を移動させるにはどのような組合せがよいか．但し，4 個の荷物のうちの 3 個は荷物に付加されている容器番号と容器に付加されている番号が一致しなくてはならない．つまり，4 個のうち 1 個だけは容器に関係なく選べる.』

なお，この知識ベースのルール数は 10 である．1 ルール当りの平均条件要素数は 2.6 であり，最大条件要素数は 6 である．また，提案手法の対象となるルールは一つあり，その条件要素数は 6 である．

6.2 実験結果

測定は SUN Sparc Classic 上で行い，最適化にかかった時間と照合時間を測定し，その user 時間を調べた．知識ベースのコンパイルにかかる時間は最適化した場合もしない場合もほとんど同じなので省略した．なお知識ベースのコンパイルには CParaOPS5 を用いた．CParaOPS5 は Carnegie-Mellon University で作成された C 言語による OPS5 のコンパイラである．最適化にかかった時間は time 命令によって，照合時間は CParaOPS5 の機能によって測定した．また，それぞれ 20 回測定しその平均値を求めた．各知識ベースに対して行った実験結果を図 6〜8 に示す．図中には，知識

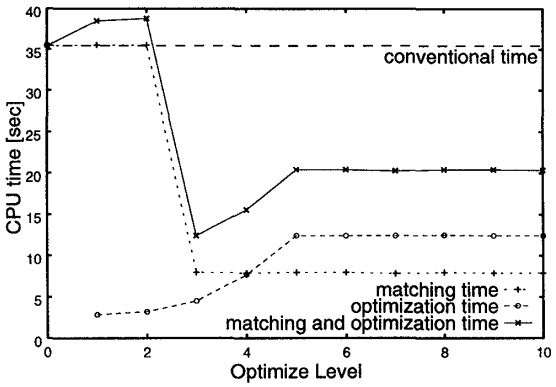


図 6 知識ベース A に対する実験結果
Fig. 6 Experimental results for knowledge-base A.

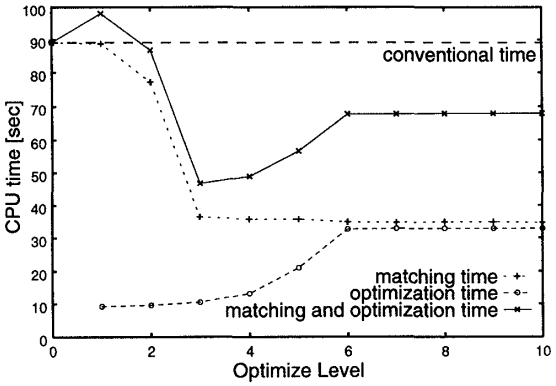


図 7 知識ベース B に対する実験結果
Fig. 7 Experimental results for knowledge-base B.

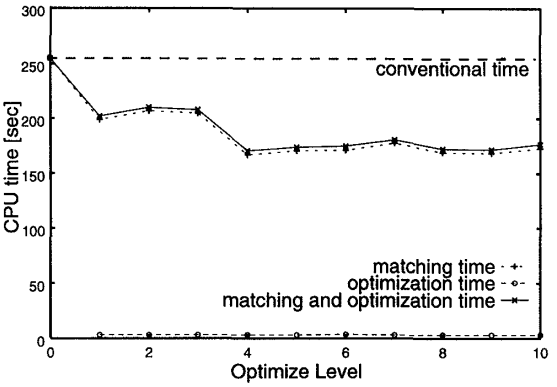


図 8 知識ベース C に対する実験結果
Fig. 8 Experimental results for knowledge-base C.

ベースの実行時間（matching time），最適化にかかった時間（optimization time），実行時間 + 最適化にかかった時間（matching and optimization time）を示した．また，図中の横軸に平行な点線は，最適化なしのときの知識ベースの実行時間（conventional time）を表している．つまり，評価の基準となる値である．なお，縦軸は CPU 時間，横軸は改善レベルとなって

表1 実験結果
Table 1 Experimantal results at $N = 4$.

	知識ベースA	知識ベースB	知識ベースC
照合時間（最適化なし）	35.52	89.23	254.49
照合時間（最適化あり）	7.91	35.80	167.32
最適化に要した時間	7.67	13.15	3.42
照合時間 + 最適化時間	15.58	48.95	170.74

（単位：秒）

いる．

実験結果から以下のことがわかる．最適化にかかる時間は，改善レベルが上がるに従って指数関数的に増えていく．但し，提案方式の対象となるルールの条件要素数を超えた改善レベルを指定しても，条件要素の順序の組合せ数は変わらない．そのため，知識ベースA，Bでは最適化にかかる時間は，改善レベルがある値以上になると一定になった．知識ベースCでは，提案方式の対象となるルールに不等式などによる条件要素の順序の制約が数多くあったため，改善レベルが増加しても条件要素の順序の組合せ数にほとんど変化がなかった．例えば，『x 以上』という条件に対しては，変数 x の値が先に確定するような条件要素の順序でないと実行できないため，実行可能な条件要素の順序の組合せ数が少なかった．それ故，最適化にかかる時間は小さく，ほぼ一定となった．

一方，照合時間は改善レベルが大きくなるにつれて短縮された．しかし，改善レベルが3〜4を超えるとほぼ一定になった．これは，ルール条件部の全条件要素を最適に並べ換えなくとも，前方に置く条件要素だけ最適に並べればよいことを意味している．言い換えると，後方の条件要素を入れ替えても照合時間はほとんど変わらなかった．

改善レベルが4のときの結果を表1に示す．照合時間については，最適化ありの場合には最適化なしの場合と比較して約40〜80%の改善が見られる．また，知識ベースの初期WMEを変化させながら何度も実行を繰り返す場合には，最適化時間も含めた評価が必要である．照合時間と最適化時間を含めた時間に注目すると，約30〜60%の改善が見られる．

知識ベースによって大きな差が見られるのは，最適化されたルールが知識ベースの実行時の照合時間に与える影響の差が知識ベースによって異なっているためと考えられる．

今回実験した知識ベースでは，属性値の分布が極端に偏っていなかったために，一様分布で仮定してもある程度対応できるという結果を得ることができた．しかし，実験結果の一部で一様分布では対応できず，予測がうまくいかないケースもあった．例えば，知識ベースCの $N = 1$ と2や $N = 6$ と7のように N の値が大きい方が結果が若干悪くなった．属性値の分布が極端に偏るケースには，今回提案した手法は対応できない．

なお，最適化に要する時間の主な手間は発火木の生成に要する時間である．

7. むすび

本論文では，知識ベースの各ルールの条件要素の順序に着目し，条件要素の順序を並べ換えることで条件照合時間を短縮させる手法を提案した．条件要素を並べ換える際には，知識ベースより得られる静的情報（WMEのクラスの分布，属性値の種類）をもとに条件照合回数の予測を行い，正確な静的情報を得るためにルールネットワークと発火ネットワークを用い，正確に条件照合回数が予測できるルールとそうではないルールのクラス分けを行う方法を提案した．正確に条件照合回数が予測できるクラスに属するルールについては，提案手法を適用することで条件照合にかかる時間を短縮することが可能となる．また，実験によって実際に提案手法の効果を測定しその有効性を確認した．

文献[10]の手法は属性数が2以下の条件要素とWMEとの高速条件照合を実現させたものであり，文献[11]の手法は，高コストルール対処法であった．本手法は，属性数に関する制限はなく，また高コストルールに限らず，発火木とルールネットワークを用いて判別されたルールをすべて高速化できる．また，条件照合アルゴリズムを変えるものではないので，容易に適用することができる．

目下，発火木の生成にかかる時間が最適化に要する時間の大半を占めている．これは，擬似的な条件照合によって予測される発火順序がかなり冗長なものとなっているためだと考えられる．今後はこの点を改良し，より有効性を高めていくことを考えていく．

謝辞 査読者の方々に有益な御助言を頂きました．

ここに記して謝意を表します．

文　献

[1] E. Wong and K. Youssefi, "Decomposition — A strategy for query processing," ACM Trans. Database Systems, vol.1, no.3, pp.223–241, 1976.

[2] C.L. Forgy, "RETE: A fast algorithm for the many pattern / many object pattern match problem," Artif. Intell., vol.19, no.1, pp.17–37, 1982.

[3] L. Brownston, R. Farrell, E. Kant, and N. Martin, "Programming Expert System in OPS5: An Introduction to Rule–Based Programming," Addison–Wesley, 1985.

[4] 服部文夫，清水信昭，土屋秀幸，桑原和宏，和佐野哲男，"知識ベース管理システム（KBMS)，" 情報処理学会，知識工学と人工知能研究会，41–6, 1985.

[5] 安西祐一郎，"OPS5 と私，" Computer Today, vol.5, no.13, pp.4–9, 1986.

[6] 荒屋真二，百原武敏，田町常夫，"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，" 情処学論，vol.28, no.7, 1987.

[7] 石田　亨，桑原和宏，"プロダクションシステムの高速化技術，" 情報処理，vol.29, no.5, pp.467–477, 1988.

[8] 石田　亨，横尾　真，"プロダクションシステムのトポロジカル・オプティマイザ，" 情報処理学会，知識工学と人工知能研究会，56–7, 1988.

[9] M. Tambe and P. Rosenbloom, "Eliminating expensive chunks by restricting expressiveness," IJCAI–89, pp.731–737, 1989.

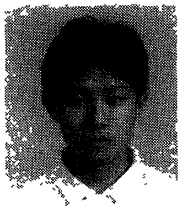
[10] 木村春彦，住吉一之，小林真也，武部　幹，"プロダクションシステムの高速条件照合アルゴリズム，" 信学論（D-II)，vol.J77-D-II, no.1, pp.143–153, Jan. 1994.

[11] 木村春彦，住吉一之，小林真也，武部　幹，"プロダクションシステムの直接条件照合アルゴリズム，" 信学論（D-II)，vol.J77-D-II, no.2, pp.370–379, Feb. 1994.

[12] T. Ishida, "An optimization algotirhm for production systems," IEEE Transactions on Knowledge and Data Engineering, vol.6, no.4, pp.549–558, Aug. 1994.

[13] J. McDermott, A. Newell, and J. Moove, "The Efficiency of Certain Production System Implementations," in Pattern-directed Inference Systems, ed. D. Waterman, et al., pp.155–176, Academic Press, 1978.

（平成 8 年 11 月 11 日受付，9 年 4 月 25 日再受付）



南保　英孝　（学生員）

平 6 金沢大・工・電気・情報卒．平 8 同大大学院電気・情報工修士課程了．同年，同大学院自然科学研究科システム科学専攻博士課程入学．現在に至る．その間，プロダクションシステムに関する研究，特に，高コストルール対処法，高速条件照合アルゴリズムに関する研究に従事．

木村　春彦　（正員）

昭 49 東京電機大・工・応用理化卒．昭 54 東北大大学院情報工学専攻博士課程了．工博．同年，富士通（株）入社．昭 55 金沢女子短大講師，昭 56 同短大助教授，昭 59 金沢大・経助教授，平 4 同大・工・電気・情報工助教授，平 6 同学科教授．現在に至る．その間，最適コード変換，プロダクションシステムの高速化の研究に従事．情報処理学会，人工知能学会各会員．

広瀬　貞樹　（正員）

昭 49 富山大・工・電子卒．昭 51 東北大大学院工学研究科修士課程情報工学専攻了．昭 55 同博士課程了．工博．同年，富士通研究所入社．昭 59 神奈川大・工助教授．平 1 富山大・工助教授．現在に至る．その間，オートマトン・形式言語理論，アルゴリズム解析，計算の複雑さの理論などの研究に従事．情報処理学会会員．