プロダクションシステムにおける2個以下の変数をもつ条件要素の高速条件照合

木村 春彦*・阿部 武彦**・南保 英孝*・大藪 多可志***

High Speed Match for Each Condition Element of the Left-Hand Side Having Less Than or Equal to Two Variables in Production Systems

Haruhiko KIMURA*, Takehiko ABE**,
Hidetaka NANBO* and Takashi OYABU***

In order to decrease the number of combinations in matching, we propose a new data structure of α -memory such that, as a rule is given, it can immediately point out working memory elements satisfying the rule's condition element.

Key Words: high speed match, α -memory, production system

1. はじめに

推論の高速化をめざした研究の1つに文献1)がある。これは、従来、意味ネットワークを利用して推論を行ってきたエキスパートシステムを、プロダクションシステム記述言語で書き改め、その高速条件照合の実現を目的としている。それゆえ、対象となるワーキングメモリエlement (WME)は、おもに2項関係の知識であり、属性数が2のことが多い。また、文献1)の論文では、属性数が2以下の条件要素から、その条件を満足させるすべてのWMEをじかに導けるワーキングメモリ(WM)のデータ構造を提案し、これを用いて高速条件照合を可能にするアルゴリズムが示されている。しかし、①属性数が2以下の条件要素の照合を高速化させることにより全体の推論速度を高速化させようとするものであり、一般の知識ベースのように条件要素の属性数がほとんど3以上となる場合には効果が薄い。②WMEの重複を許していないが、現実の知識ベースでは重複するWMEが見えけられる。③高速化の対象である属性数が2以下の条件要素の属性値には、変数と定数しか許していない。つまり、選言命題や連言命題などを許していない。④メモリ使用量が多い。ただし、照合の計算コストが大きくなるにしたがい、Reteアルゴリズム²⁾のメモリ使用量よりも少なくなる。などの欠点があった。

本論文では、上記①～③の欠点を克服するための α メモリのデータ構造を提案する。具体的には、文献1)で提

案されたWMのデータ構造を α メモリのデータ構造に拡張することにより、2個以下の変数をもつ条件要素とWMEとの高速条件照合を実現する。これにより、属性数が3以上の条件要素とWMEとの条件照合に対しても高速化が期待できる。また、 α メモリ内の同じWMEをポインタによって結ぶことにより、②の欠点を除去する。③の欠点は、WMのデータ構造を α メモリのデータ構造に拡張する際に、WMEの属性値で、条件要素の変数に対応しないものをすべて同一とみなすことにより解決する。④のメモリ使用量の問題については、改善できない。本論文の方法では、結果的に、文献1)で提案されたWMのメモリ使用量が α メモリに移っただけであり、全体としてのメモリ使用量はWMEの重複した分のメモリが増えただけである。以下に、提案する α メモリのデータ構造について解説する。

2. 提案する α メモリのデータ構造

提案する α メモリのデータ構造は、その α メモリに対応する条件要素に含まれる変数の数によって異なる。また、イントラコンディションテスト²⁾を通過したWMEは、条件要素の変数に対応しない属性値をすべて同一とみなす。以下に変数の数別のデータ構造を示す。ただし、 k は α メモリの番数であり、 wme はイントラコンディションテストを通過したWMEである。

(1) 変数がないケース

変数をもたない条件要素を満足させた wme はスタック F_k に格納する。つぎに、このタイプの条件要素に対応する α メモリへの wme の追加と削除の処理を示す。ここで、 PR_k : ポインタ, t : wme のタイムタグ, t' : 削除したい wme のタイムタグ

(a) 追加処理

$$PR_k \leftarrow PR_k + 1;$$

$$F_k(PR_k) \leftarrow t;$$

(b) 削除処理

$$PR \leftarrow PR_k;$$

$$\text{while } F_k(PR) \neq t' \text{ do } PR \leftarrow PR - 1;$$

$$F_k(PR) \leftarrow F_k(PR_k);$$

* 金沢大学工学部 金沢市小立野2-40-20

** 石川職業能力開発短期大学校 石川県鳳至郡穴水町由比ヶ丘いの45-1

*** 富山国際大学人文学部 富山市大山町東黒牧65-1

* Faculty of Technology, Kanazawa University, Kanazawa

** Polytechnic College Ishikawa, Fugeshi-gun, Ishikawa

*** Faculty of Humanities, Toyama University of International Studies, Toyama

(Received June 5, 1995)

(Revised November 10, 1995)

$PR_k \leftarrow PR_k - 1$;

(2) 1変数のケース

1変数をもつ条件要素に対応する α メモリのデータ構造は、スタック R_k と1次元配列 A_k で構成する。 R_k は k 番目の α メモリに対応する条件要素を満足させるWMEのタイムタグの記憶場所を格納するスタックであり、 A_k はその該当するWMEのタイムタグの記憶場所が R_k のどこにあるかを指す1次元配列である。なお、同一のWMEのタイムタグはポインタによってつながれており、タイムタグの追加時には先頭につけられる。また、先頭のタイムタグが記憶されている番地が R_k に格納されている。つぎに、このタイプの条件要素に対応する α メモリへのwmeの追加と削除の処理を示す。ここで、 i :条件要素の変数に対応するwmeの属性値No., ただし、属性値No.はコンパイル時に属性値を番数化したもの、 $t1$:wmeのタイムタグの記憶場所(番地)、 $Pro_LT(a, b, c)$: a 番地から線形リスト(セル)(b, c)を格納する手続き、ただし、 b はタイムタグ、 c はポインタが指す番地(ポインタ値)、 $TT_cell(a)$: a 番地から格納されているセルのタイムタグ、 $PT_cell(a)$: a 番地から格納されているセルのポインタ値、 $AVN(a)$: a 番地に格納されているタイムタグに対応するWMEに対して、条件要素の変数に対応する属性値No.

(a) 追加処理

if $A_k(i)=0$ then begin

$PR_k \leftarrow PR_k + 1$;

$R_k(PR_k) \leftarrow t1$;

$Pro_LT(t1, t, 0)$;

$A_k(i) \leftarrow PR_k$

end

else begin

$Pro_LT(t1, t, R_k(A_k(i)))$;

$R_k(A_k(i)) \leftarrow t1$

end

(b) 削除処理

$a \leftarrow R_k(A_k(i))$;

if $TT_cell(a) = t'$ then begin

$i' \leftarrow AVN(R_k(PR_k))$;

$A_k(i') \leftarrow A_k(i)$;

$R_k(A_k(i')) \leftarrow R_k(PR_k)$;

$PR_k \leftarrow PR_k - 1$;

$A_k(i) \leftarrow 0$

end

else DELETE-CELL(a);

ここで、

Procedure DELETE-CELL(a):

begin

$c \leftarrow PT_cell(a)$;

while $TT_cell(c) \neq t'$ do begin

$a \leftarrow c$;

$c \leftarrow PT_cell(a)$

end;

$PT_cell(a) \leftarrow PT_cell(c)$

end

データ構造の概念図を Fig. 1 に示す。

(3) 2変数のケース

2変数をもつ条件要素に対応する α メモリのデータ構造は、つぎの3種のスタック $S_{k,j}$, $T_{k,i}$, U_k と3つの2次元配列 B_k , C_k , D_k から構成する。条件要素で変数が割り当てられている属性を左側からそれぞれ属性 x , 属性 y と呼ぶことにする。また、対応するWMEの属性についても同様に呼ぶことにする。 k 番目の α メモリに対応する、このタイプの条件要素を満足させるWMEに対して、 $S_{k,j}$ は属性 y の属性値No.が j であるWMEのタイムタグの記憶場所を格納するスタックであり、 $T_{k,i}$ は属性 x の属性値No.が i であるWMEのタイムタグの記憶場所を格納するスタックである。そして、 U_k は k 番目の α メモリに対応する条件要素を満足させるWMEのタイムタグの記憶場所を格納するスタックである。 B_k , C_k , D_k は該当するWMEのタイムタグの記憶場所がそれぞれ $S_{k,j}$, $T_{k,i}$, U_k のどこにあるかを指す2次元配列である。 B_k と $S_{k,j}$, C_k と $T_{k,i}$, D_k と U_k はそれぞれペアとなるが、各 k に対して常にこの3組のペアが必要となる訳ではない。たとえば、ルール内の他の条件要素にこれらの変数と同じものが代入されていないときは、 D_k と U_k のペアだけで済む。このように各 k に対して、どのようなペアが必要かをあらかじめコンパイル時段階でチェックしておく。つぎに、このタイプの条件要素に対応する α メモリへのwmeの追加と削除の処理を示す。ただし、スペースの関係で、 B_k と $S_{k,j}$ のペアについてのみ示す。ここで、 i :wmeの属性 x の属性値No., j :wmeの属性 y の属性値No., $PS_{k,j}$:ポインタ, $AVN(a)$: a 番地に格納されているタイムタグに対応するWMEの属性 x の属性値No.

(a) 追加処理

if $B_k(i, j)=0$ then begin

$PS_{k,j} \leftarrow PS_{k,j} + 1$;

$S_{k,j}(PS_{k,j}) \leftarrow t1$;

$Pro_LT(t1, t, 0)$;

$B_k(i, j) \leftarrow PS_{k,j}$

end

else begin

$Pro_LT(t1, t, S_{k,j}(B_k(i, j)))$;

$S_{k,j}(B_k(i, j)) \leftarrow t1$

end

(b) 削除処理

Fig. 1 Data structure of α -memory for a condition element with one variable

Fig. 2 Data structure of α -memory for a condition element with two variables

```

(p north-pick-one-4
 (context ^name north)
 (scheduling ^night <n>)
 { <THE-FOURSOME> (foursome ^night <n> ^north nil) }
 { <THE-PLAYER> (player ^number <p> ^nights-scheduled <n>) }
 ->
 (modify <THE-PLAYER> ^nights-scheduled <n>)
 (modify <THE-FOURSOME> ^north <p>))

(p north-done
 { <THE-CONTEXT> (context ^name north) }
 ->
 (modify <THE-CONTEXT> ^name make-candidates))

(p make-candidates-make-candidate
 (context ^name make-candidates)
 (scheduling ^night { <n> <3> })
 (foursome ^night <n> ^group <g> ^north <yankee>)
 (player ^number <redneck> ^nights-scheduled <n>)
 (player ^number { <oriental> <redneck> } ^nights-scheduled <n>)
 (player ^number { <cowboy> <oriental> } ^nights-scheduled <n>)
 - (already-played ^player1 <yankee> ^player2 <redneck>)
 - (already-played ^player1 <yankee> ^player2 <oriental>)
 - (already-played ^player1 <yankee> ^player2 <cowboy>)
 - (already-played ^player1 <redneck> ^player2 <oriental>)
 - (already-played ^player1 <redneck> ^player2 <cowboy>)
 - (already-played ^player1 <oriental> ^player2 <cowboy>)
 - (candidate ^group <g> ^chosen no ^south <redneck> ^east <oriental>)
 - (candidate ^group <g> ^chosen no ^south <redneck> ^west <cowboy>)
 - (candidate ^group <g> ^chosen no ^east <oriental> ^west <cowboy>)
 ->
 (make candidate ^group <g> ^chosen no
 ^south <redneck> ^east <oriental> ^west <cowboy>))

```

Fig. 3 A part of knowledge-base called Tourney

```

 $\alpha \leftarrow S_{k,j}(B_k(i, j))$ ;
if TT_cell( $\alpha$ ) =  $t'$  then begin
   $i' \leftarrow A1VN(S_{k,j}(PS_{k,j}))$ ;
   $B_k(i', j) \leftarrow B_k(i, j)$ ;
   $S_{k,j}(B_k(i, j)) \leftarrow S_{k,j}(PS_{k,j})$ ;
   $PS_{k,j} \leftarrow PS_{k,j} - 1$ ;
   $B_k(i, j) \leftarrow 0$ 
end
else DELETE-CELL( $\alpha$ );

```

B_k と $S_{k,j}$ のペアに対する概念図を Fig. 2 に示す。

(4) その他のケース

従来型の α メモリのデータ構造である。

3. 実験

提案した方法はただちに照合ができるため、高コストルールの条件照合に有用であると思われる。高コストルールとは条件照合の計算コストが大きいルールである。そこで、高コストルールの条件照合時間を Rete アルゴリズムと提案方法と比較した。用いたルールを Fig. 3 に示す。これは、tourney と呼ばれるトーナメント試合の

Fig. 4 Accumulative match time

スケジュール表を作成する知識ベースの一部である。3番目のルールが高コストルールであるので、このルールが実行されるまでの累積条件照合時間を求めた。Fig. 4 にその結果を示す。このことから提案方法の有用性が確かめられた。

4. むすび

本論文では、文献1)のWMのデータ構造を α メモリのデータ構造に応用することにより、属性数2以下の条件要素の高速条件照合を2変数以下の条件要素の高速条件照合に拡張する方法を提案した。これにより、文献1)で対象になった条件要素のクラスよりも広いクラスで高速条件照合が可能になった。今後の課題としては、提案した α メモリのデータ構造を採用した条件照合アルゴリズムをプロダクションシステムのコンパイラに組み込むことにより、実際の知識ベースで、高速条件照合が可能となることを実証することである。

謝辞 本研究の一部は大川情報通信基金の援助によるものである。ここに記して謝意を表する。

参考文献

- 1) 木村, 住吉, 小林, 武部: プロダクションシステムの高速条件照合アルゴリズム, 電子情報通信学会論文誌 D II, J77-D-II-1, 143/153(1994)
- 2) C. L. Forgy: RETE: A fast algorithm for the many pattern/many object pattern match problem, Artif. Intell., 19-1, 17/37 (1982)