推論高速化のための弁別ネットワークの動的変形法

7M-4

田野　俊一　　増位　庄一　　船橋　誠壽
（株）日立製作所システム開発研究所

1．はじめに

知識を断片的なIF～THEN～ルールで表すＡＩツールにおいては、ルール条件部の真偽判定に多くの処理量が必要であり、推論速度低下の大きな要因となっている。これを解決するために、ルール条件部を弁別ネットに展開する方式が提案されている[1],[2]。

本論文では、弁別ネットを構成するノードにおける処理を解析し、処理効率が高い弁別ネットに動的に変形する方法について述べる。

2．ルール条件部の弁別ネット表現とその問題点

対象を表現したフレームの状態を、ルールの条件として記述する推論システムを考える。

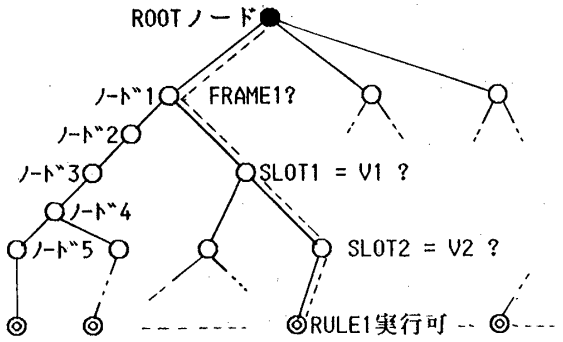
下図に示したフレームは、名称がFRAME1であり、項目としてSLOT1,SLOT2,…を持ち、それぞれの値がVALUE1,VALUE2,…である。一方、ルール条件部では、名称がFRAME1であるフレームがあり、SLOT1,SLOT2,…の値がそれぞれV1,V2,…と等しいという条件が記述されている。

```
フレーム      (FRAME1   SLOT1 VALUE1
                          :     :
                        SLOTn VALUEn )
ルール        IF  (FRAME1  SLOT1 = V1
                           :     :
                         SLOTn = Vn )
              THEN .....
```

図１　フレーム、ルールの記述例

上記ルールの真偽判定を高速に行うために、ルール条件部を図２に示す弁別ネットに変換する。弁別ネットの各ノードは、１つの条件、例えば「SLOT1の値がV1と等しい」、を表している。

弁別ネットの頂点ノードであるROOTノードより、フレームを各枝に沿って流し、条件を満たすと次の枝へと流す操作を行う。終端ノードに到達した場合、どのルールの条件が満たされているかを知ることができる。例えば、図１のルールの条件部が真であると判定するには、点線で示した経路でフレームが流れる必要がある。



●ROOTノード，○条件を表すノード，◎終端ノード

図２　弁別ネットの例

一般に、このような弁別ネットを用いた条件成立判定方法では、ノードに記されている条件を満たすフレームが少ない条件に関するノードが弁別ネットの上部に位置したほうが効率よい真偽判定を行える。しかしこれを実現しようとする時には、条件がどのように弁別ネットに変換されるかをルールの作成者が把握し、意識的にルール条件部の記述位置を変える必要がある。しかし現実にはルール作成者が、どの条件がどのくらいの割合で満足されるかをすべて知ることは困難であるし、また対象とする現象の性質が変化するとノードの満足される割合も動的に変化するため、ルール作成者による条件記述の順序付けによる弁別ネットの効率化には限界がある。

3．弁別ネットの動的変形法

上記問題点を解決するための基本的な考え方は、「弁別ネットの処理過程において、ノードの処理量やどのくらいの割合でノードに記されている各条件が満足されたか等に関するノード処理情報を収集し、これを用いて弁別ネットを変形する」である。

これを、一定時間間隔で行い、動的に弁別ネットを変形すれば、常に効率的な弁別ネットを得ることができる。

このような、弁別ネットの変形には、次に示す解決すべき問題点がある。

（１）ノードの満足される割合は、ノードの上下関

Dynamic Reconfiguration of Discriminant Net for Fast Pattern Match
Shun'ichi TANO, Syouichi Masui, Motohisa FUNABASHI
HITACHI,Ltd.

係に強く依存する。例えば、ノードAがノードBより上に位置し、それぞれの満足される割合が、0.5、0.2であっても、0.2は条件付き確率であるのでノードA、B、の順位を入れ換えることが、必ずしも弁別ネットの処理効率を上げるとは言えない。

（2）ノードの上下関係の並べ換えは、組み合わせ問題となり、計算量が多くなる。

そこで、弁別ネットの処理モードを2つに分け、通常モードでは、ノード処理情報は収集せず、従来の弁別ネット処理情報を行い、ログモードでは、ノード処理情報を、弁別ネットの全ノードに関して収集する。つまり、ログモードでは、あるノードで、チェックが失敗しても、ノード処理情報を得るために次の枝へフレームを流し処理を続ける。

このように、弁別ネットの全ノードに対してノード処理情報を収集するため、上記問題（1）に示した上下関係に関するデータもノード処理情報より抽出することが可能となり、また、問題点（2）も、ノード間の上下関係に関する好ましさを用いることにより解消できる。

以下、ノードの位置関係の変更アルゴリズムを図2のﾉｰﾄﾞ1、2、3、4、5の並べ換えを例にとり説明する。

まず、次のように定義する。

N＝｛n1,…,nm｝　Nはノードの集合

F＝｛f1, ,fm｝　Fはログモードの時に弁別ネットを流れたフレームの集合

I（ni, fk）　フレームfkを処理した時のノードniの処理情報

P（ni, nj）　ノードniがノードnjの上位に位置する好ましさ

I（ni,fk）の例を図3に示す。I（ni,fk）の絶対値は、ノードniにおいて、フレームfkを処理するために要した時間であり、正数の場合は条件成立、負数の場合は条件不成立であることを示す。例えば、I（ノード5、FRAME4）が－3とはノード5に記されている条件の成立判定に時間3を要し、条件不成立と判定されたことを意味する。

ni＼fk	FRAME1	FRAME2	FRAME3	FRAME4	FRAME5
ノード1	5	1	6	2	3
ノード2	2	-1	-3	5	-4
ノード3	-5	-2	6	-1	3
ノード4	5	4	-3	2	-1
ノード5	-5	-2	1	-3	-4

図3　I（ni,fk）の例

次に、I（ni,fk）より、P（ni,nj）を求める方法を説明する。

ステップ1　上下の位置関係を効率化したいNを設定する。例えば、N＝｛ﾉｰﾄﾞ1、ﾉｰﾄﾞ2、…、ﾉｰﾄﾞ5｝

ステップ2　このNの要素ni,njの組で、i<jを満たすものすべてに対してステップAを繰り返す。

ステップA　evを0とする。Fの全ての要素fkについて①②③を行い、得られたevを用いて、Pを、P（ni,nj）＝ev,P（nj,ni）＝ -1*ev,で算出する。

①ev_ni=I(ni,fk),ev_nj=I(nj,fk)

②ev'=
- 0　(ev_ni>0,ev_nj>0)
- -1*ev_ni　(ev_ni>0,ev_nj<0)
- ev_nj　(ev_ni<0,ev_nj>0)
- ev_ni-ev_nj　（上記以外）

③ev=ev+ev'

この操作によりP（ni,nj）は下図となり、これを用いてノードの効率的な並びを導く方法を示す。

ni＼nj	ﾉｰﾄﾞ1	ﾉｰﾄﾞ2	ﾉｰﾄﾞ3	ﾉｰﾄﾞ4	ﾉｰﾄﾞ5
ﾉｰﾄﾞ1		-10	-8	-9	-11
ﾉｰﾄﾞ2	10		3	1	-5
ﾉｰﾄﾞ3	8	-3		2	-1
ﾉｰﾄﾞ4	9	-1	-2		-7
ﾉｰﾄﾞ5	11	5	1	7	

図4　P（ni,nj）の例

各ノードの行データを加えた値を、そのノードが一番上位に位置する好ましさとし、それをすべてのノードについて計算する。図4では、ノード1　-10-8-9-11= -38,ノード2　10+3+1-5= +9,ノード3　8-3+2-1= +6,ノード4　9-1-2-7= -1,ノード5　11+5+1+7=24となり、ノード5が最上位に位置すべきであると判定する。次に、図4よりノード5に関する行データ、列データを取り除き同様の操作を行う。これを繰り返し、ノードの順序を決定する。

以上より、ノードの並びは、ノード5、2、3、4、1、の順に変更され、同じFの要素を弁別ネットに流す時間が38から18に減少し、約2．1倍の効率化となる。

［参考文献］

［1］Forgy,C.L.:"Rete:A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem",A.I.,vol.19,No.1

［2］田野：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回情処全国大会、P993