3G-7

EWS4800シリーズ －EXCOREにおける ルール推論処理方式－

新田 徹
(日本電気株式会社)

1．はじめに

EXCORE(EXpert system CORE)[1] は、オブジェクト指向とプロダクションシステムをベースにしたエキスパートシステム構築支援ツールであり、EWS4800が提供している基本ウィンドウをベースにした知識エディタ、視覚に訴えるビジュアルデバッガ等を備え、エキスパートシステム構築の生産性の向上を目指したものである。

本稿では、EXCOREにおいて生じたルール推論に関する問題点、および、それを解決するために取った方策についての一考察を示す。 その主要な内容は(1) ルール記述に陽に現れない知識、情報をルール推論中に更新した場合の更新把握問題に対して、更新通知方式を取ったこと、(2) 無限ループ回避のためのトークン単位の細分化などにより生じたオーバヘッドに対し、Reteﾂﾘｰｺﾝﾊﾟｲﾙ方式を取ったことである。

2．ルール記述性に関する問題点

EXCOREでは高速化のために、Reteｱﾙｺﾞﾘｽﾞﾑ[2] を使用したﾙｰﾙｺﾝﾊﾟｲﾙ方式を取っているが、次に示すようなルール記述性に関する問題が生じた。

(1) 更新把握問題

Reteﾂﾘｰ に流すトークンの生成は、ﾙｰﾙｺﾝﾊﾟｲﾙ時に収集した情報に基づいて行われる。そのため、ルール記述に陽に現れない知識、情報をルール推論中に更新した場合の知識、情報に関するトークンは生成されない。つまり、ルール推論実行時に行われた更新のうち、ルール推論に反映されない更新が存在する可能性があるわけである。そのため、利用者はルール推論に反映させたい知識、情報は必ず陽に記述しなければならないという制限をうけることになる。

(2) 無限ループ問題

トークンの単位がフレーム、レコード単位である為、無限ループに陥るケースが生じる。 そのため、利用者は無限ループを回避するためにフラグを立てたり、そのような記述を避ける等の負担を強いられることになる。

3．ルール記述性の向上

2節で示した問題点を解決するためにEXCOREでは次のような方法を取った。

(1) 更新通知方式

更新把握問題を解決するために、ﾙｰﾙｺﾝﾊﾟｲﾙ時に、ルールソースに記述された更新に関する記述 ((SEND-MESSAGE 田中 PUT-VALUE 持病 頭痛) など) から更新情報を生成し、その更新情報に基づいてトークンを生成する方式の代わりに、次に示すような更新通知方式を取った。すなわち、ルール推論時に行われる更新を管理する更新情報テーブルと更新情報テーブルをメンテナンスする更新通知モジュールを設けた。 その一連の処理の流れを以下に示す（図1参照）。推論エンジンは、更新機能を持つLISP関数を発行することにより、フレーム型知識、述語型知識、データベース情報の更新を行う。 フレーム管理、述語管理、データベース管理などの各管理モジュールは更新の行われたことを更新通知モジュールに伝える。更新通知モジュールは受け取った更新情報 ((UPD (FRAME 田中 持病 頭痛) など) に基づいて更新情報テーブルを更新する。 推論エンジンは条件照合処理を行う直前に、更新情報テーブルを参照し、更新情報を得、その更新情

EWS4800 Series - Inference Mechanism in EXCORE -
Tohru Nitta
NEC Corporation

報からトークンを生成し、Reteﾂﾘｰ に流すことになる。 これにより、ルール推論時の更新はすべてルール推論に反映されることになった。

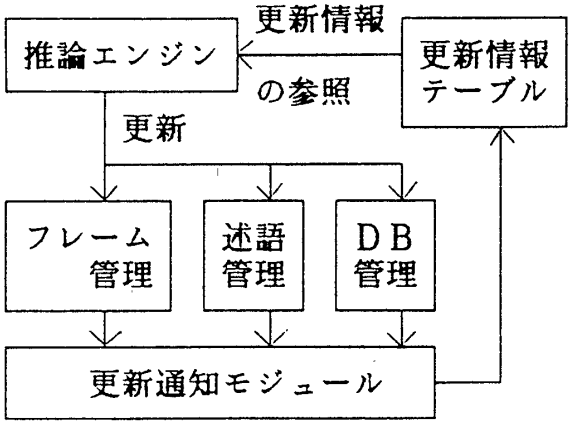

図１

(2) トークン単位の細分化

無限ループ問題を解決するために、トークンの単位をフレーム、レコード単位からスロット、データ項目単位へと細かくした。 これにより、例えば、あるスロットが更新された場合、そのスロットに関するトークンがReteﾂﾘｰ を流れることになり、無限ループを回避することが可能となった。

上述した方法はルール記述性を向上させるが、若干のオーバヘッドをもたらす。

４．ルール推論の高速化

３節で示した解決策の適用によるオーバヘッドを解消するために、そして、ルール推論速度をより高速化するために次のような方法を取った。

(1) Ｒｅｔｅツリーコンパイラ

従来は、Reteﾂﾘｰ をテーブル（その実体はフラグを含んだLISPのＳ式）で実現し、推論エンジンは、そのReteﾂﾘｰ 管理テーブル中のフラグを参照しながら、LISP関数を評価することにより、トークンをReteﾂﾘｰ に流すという方式を取っていた。そこで、ルール推論の高速化を図るために、Reteﾂﾘｰ 管理テーブルをLISP関数に変換するReteﾂﾘｰｺﾝﾊﾟｲﾗを作成した。推論エンジンは、Reteﾂﾘｰｺﾝﾊﾟｲﾗが出力するReteﾂﾘｰ 関数を単に評価（eval）することでReteﾂﾘｰ にトークンを流すことになる。 図２にReteﾂﾘｰ 管理テーブルとReteﾂﾘｰ 関数のイメージを示す。 図２において、ROOT,SLCT はフラグであり、IS-FRAMEはトークンがフレームに関するものであるか否かを調べるLISP関数である。

Reteﾂﾘｰ 管理ﾃｰﾌﾞﾙ:
（ROOT（SLCT（(IS-FRAME ﾄｰｸﾝ) ～）
Reteﾂﾘｰ関数:
（LAMBDA（ﾄｰｸﾝ).Ｓ式列）

図２

(2) グループノード

Reteﾂﾘｰ にグループノードを設けることにより、イントラノードをグループ化した。 これにより、トークンを流すべきイントラノードの範囲が絞られ、その分、高速化が図れる。 グループ化の基準としては、フレーム型知識、述語型知識、LISP変数、データベース情報などが考えられる（図３参照）。 グループノードの効果はイントラノードの数が多いほど高い。

上記(1),(2) の適用の結果、ルール記述性の向上を行う前と比較して、ベンチマークテストで、２～３倍のルール推論速度の向上が図られた。

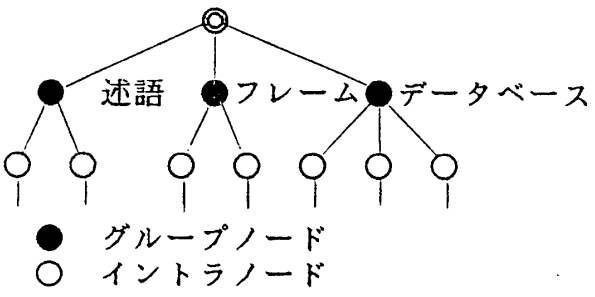

図３

５．おわりに

本稿ではルール記述性およびルール推論速度を高める方式について報告した。ルール記述性とルール推論速度はトレードオフの関係にあるが、可能な範囲で双方とも向上させるべく検討を続ける予定である。

６．参考文献

[1] EXCOREﾌﾟﾛｸﾞﾗﾐﾝｸﾞ 説明書, EWQ51-3, 日本電気株式会社, (1988)

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