7D-4　汎用エキスパートシステムの高速化技術(III)　性能評価と分析

平山　洋一　　島田　優　　黒沢　憲一
日立製作所　日立研究所

1　はじめに

現在、汎用エキスパートシステムの高速化技術として、ＲｅｔｅやＴｒｅａｔアルゴリズム[1][2]に基ずくインタプリタ型やルールコンパイル型の推論処理方式が提案されている。我々はより高速化を目指し、パターンマッチコスト及び中間結果管理コストの両面に着目した、コストバランスアルゴリズムを提案した。[3]

本報告では、従来の処理方式とコストバランスアルゴリズムに基ずくコンパイル型処理方式の性能評価及び分析結果を示す。

2　コンパイル型

本報告のコンパイル型[4]とはルール及びフレームを共にコンパイルし機械語命令にダウンロードする型を言い、第１図はコンパイル型エキスパートシステムの全体構成を示す。コンパイル型は競合解消部、ルール部、フレーム部に大別され、それぞれ機械語命令列及びデータからなる。

競合解消により選択されたルールのＲＨＳはＲＨＳ命令列により実行される。その結果フレームが変更された場合、そのフレームを参照するＬＨＳ命令列が順に実行される。ＬＨＳ命令列は変更されたフレームを含むインスタンシエーションを削除し、照合判定条件をレジスタにセットして、フレーム命令列を順に実行する。フレーム命令列はフレーム内のスロット値をレジスタにセットし、ＬＨＳ命令列へ持ちかえる。ＬＨＳ命令列では判定条件とスロット値とのパターンマッチが行われ、成功した場合インスタンシエーションを新たに追加する。このようにして全てのＬＨＳ命令列の実行が終了すると競合解消に制御が移る。

以上の実行サイクルからわかるように、ＬＨＳ命令列からスロットコードを見つけスロット値を持ち帰る機能、判定条件とスロット値との照合を行う機能、ＲＨＳを実行する機能を、コンパクトな命令列で実現しているため、コンパイル型の高速化に大きく貢献している。

（１）コンパイル型の高速性

コンパイル型により、上記３つの機能がどの程度高速になるか評価を行った。第２図は、（ａ）スロット値アクセス：判定条件を認識しその条件に合ったスロット値を検索、（ｂ）スロット値比較：判定条件とスロット値とのパターンマッチ処理、（ｃ）スロット値変更：変更するスロットを検索し変更の、コンパイル型とインタプリタ型の実行時間比率である。

この結果、コンパイル型はインタプリタ型に比較し、８．５倍（ａ）、２５倍（ｂ）、２２倍（ｃ）高速であることが分かった。

3　性能評価と分析

これより、我々が提案したコストバランスアルゴリズムに基ずくコンパイル型処理方式の性能評価及び分析を行う。

（１）　コストバランスアルゴリズム

従来のアルゴリズム[1][2]ではパターンマッチコストと中間結果管理コストの面から、

（ｉ）条件節毎にのみ中間結果を保持して中間結果管理コストを少なくする。

（ii）条件節毎と条件節間に共に中間結果を保持してパターンマッチコストを少なくする。

の２つが提案されている。しかし、ルール条件部の記述には種々なパターンがあり上記のように無条件に中間結果保存レベルを決定できない。

そこで、コストバランスアルゴリズム[3]はルール条件部をパターンマッチコスト及び中間結果管理コストの両面から評価し(コスト評価)、中間結果保存レベルを決定することで、従来のアルゴリズム[1][2]に比べ、より高速化することを目的とする。

（２）中間結果を保存する利点

第３．１図はフレーム名が定数で与えられている条件節が複数存在するルールの照合時間の内訳及び１認知行動サイクルの性能比較を示す。互いに独立な条件節から成るルール条件部については、ｒｅｔｅアルゴリズムでは無条件に全ての条件節及び条件節間に中間結果を保存する。一方、コストバランスアルゴリズムではコスト評価の結果、全ての条件節に中間結果を保存するが、互いに独立な条件節間には保存しない。

よって、コンパイル型ｒｅｔｅアルゴリズムに対しコンパイル型コストバランスアリゴリズムは中間結果管理コストが８２．４％から６４．８％に減少し、１．３７倍高速化を図ることができた。

（３）中間結果を保存しない利点

第３．２図は、フレーム名がfaの条件節から?yの条件節へスロットs1の値を渡しスロットs2の値とのパターンマッチを行い、フレーム名がf1のスロットs3の値と100とのパターンマッチを行うルールについて、スロットs1の値を変更した時の照合時間の内訳及び１認知行動サイクルの性能比較を示す。この場合、ｒｅｔｅアルゴリズムは無条件に全ての条件節及び条件節間に中間結果を保存する。一方、コストバランスアルゴリズムではコスト評価の結果、フレーム名がfaとf1の条件節は中間結果管理コストが少ないため中間結果を保存し、フレーム名が?yの条件節は多くのフレームが成立し再照合の可能性が高くなり、中間結果管理コストが多くなるため中間結果を保存しない。

しかし、第３．２図の場合、スロット値の変更によって再照合となるフレームはfaであり、Ｒｅｔｅアルゴリズムに基ずき、フレーム名が?yの条件節に保存した中間結果は有効に利用でき、パターンマッチが高速になる。

よって、この場合、コンパイル型ｒｅｔｅアルゴリズムに対しコンパイル型コストバランスアリゴリズムはパターンマッチコストが３１．８％から６８．３％に増加し、性能は０．８３倍となる。

第３．３図は、第３．２図と同じルールについて、スロットs2の値を変更した時の照合時間の内訳及び１認知行動サイクルの性能比較を示す。この場合、ｒｅｔｅアルゴリズム及びコストバランスアルゴリズムの中間結果保存レベルは上記第３．２図の場合と同様である。

第３．３図の場合、スロット値の変更によって再照合となるフレームは、フレーム名が?yの条件節において成立した多くのフレームの1つであり、Ｒｅｔｅアルゴリズムに基ずいて中間結果を保存すると中間結果の中から変更されたフレームを探す手間がかかる。一方、コストバランスアルゴリズムに基ずいて中間結果を保存しなければ変更されたフレームのパターンマッチだけでよく、高速化が図れる。

A Performance Improvement Technology in Expert Systems (III)
Evaluation and Analysis of Performance Enhancement

Hirokazu HIRAYAMA , Masaru SHIMADA , Ken'ichi KUROSAWA
Hitachi Research Laboratory , Hitachi Ltd.

よって、この場合、コンパイル型ｒｅｔｅアルゴリズムに対しコンパイル型コストバランスアルゴリズムは中間結果管理コストが８９．６％から７８．１％に減少し、１．６１倍高速化を図ることができた。

（4）分析

第３．２図、第３．３図の性能比較からわかるように、従来のアルゴリズムに比べコストバランスアルゴリズムは常に高速なわけではない。しかし、ルール及びフレームから成る知識ベースが大規模なったとき、フレーム名が定数の条件節とフレーム名が変数の条件節について再照合の対象となる確率の面から見ると、後者の確率が圧倒的に高くなる。

つまり、第３．３図のようなパターンが多くなることから、従来のアルゴリズム[1][2]に比べ総合的にはコンパイル型コストバランスアルゴリズムが高速と言える。

4　おわりに

本報告では、パターンマッチコスト及び中間結果管理コストの両面に着目したコスト評価によって中間結果保存レベルを決定する新しいアルゴリズムの性能評価及び分析を行った。これによってコストバランスアルゴリズムは実際のアプリケーションでも有効であると考えられ、今後、それを実証することが必要である。

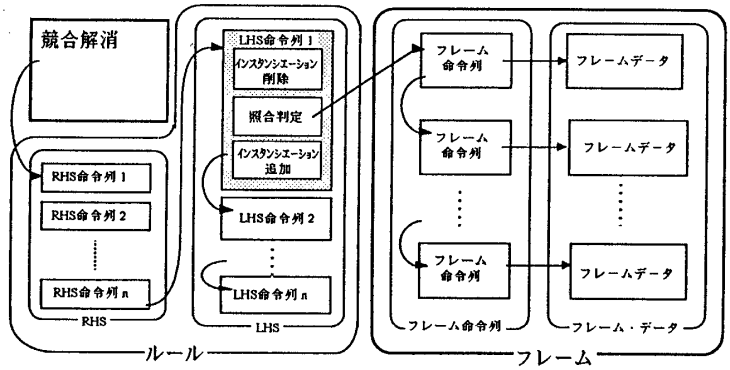
参考文献

[1] Forgy,C.L : "RETE : A Fast Algorithm For Many Pattern/Many Object Pattern Match Problem", Artificial Intelligence,vol.19 pp17-37,1982

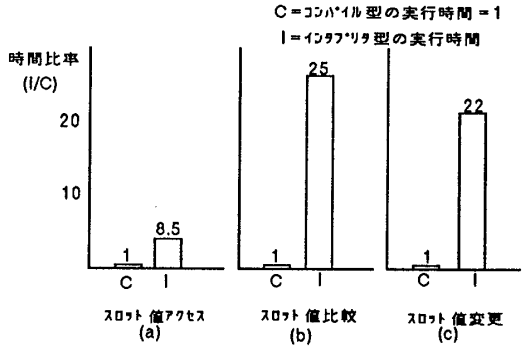
[2] Miranker,D.P : "TREAT : A Better Match Algorithm For AI Production System, AAAI`87 pp42-47,1987

[3] 黒沢憲一、島田優、平山洋一："汎用エキスパートシステムの高速化技術"、情報処理学会第39回全国大会(I)-(III)

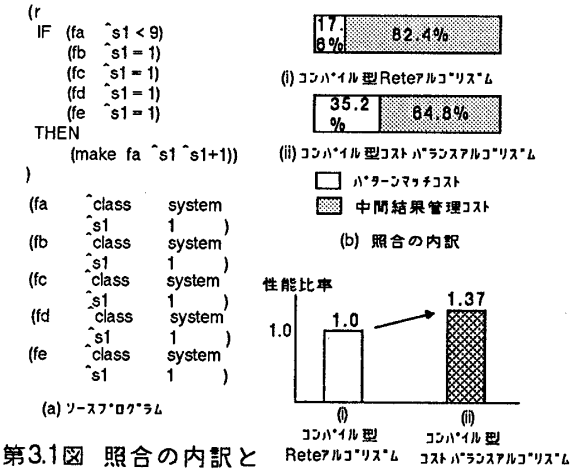
[4] 島田優、黒沢憲一、平山洋一："IPPによる汎用エキスパートシステムの高速化技術"、情報処理学会研究会 計算機アーキテクチャ-75-5



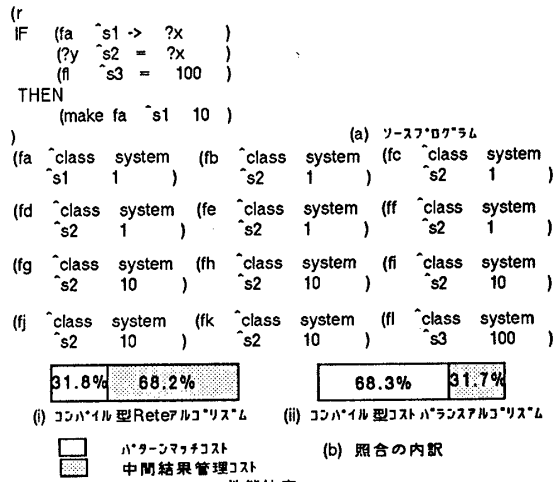
第１図　全体構成



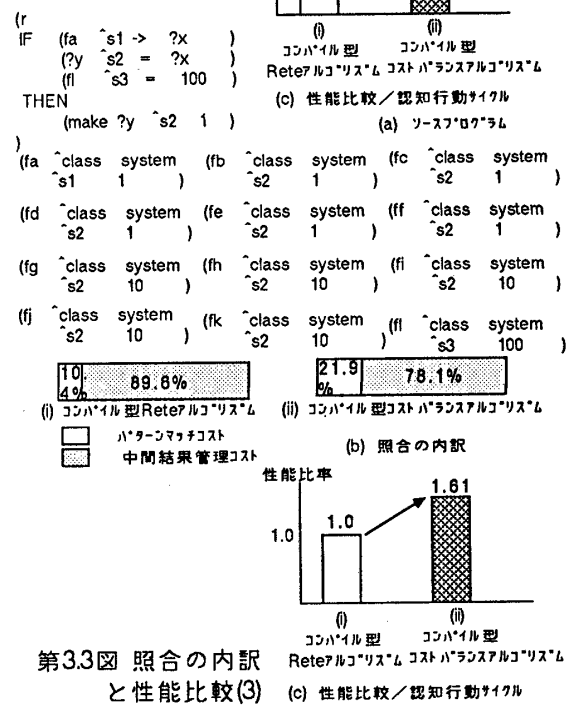
第2図　コンパイル型の高速性



第3.1図　照合の内訳と性能比較(1)



第3.2図 照合の内訳と性能比較(2)



第3.3図 照合の内訳と性能比較(3)