記　号　処　理　56－4
計算機アーキテクチャ　84－4
（１９９０．　９．　２８）

機能メモリに基づく推論機械のアーキテクチャ

小原清弘　坂井雄介　五十嵐智　小谷善行　阿刀田央一　斎藤延男
東京農工大学工学部数理情報工学科

通常のRAMに記憶内容の回復機能を付加した機能メモリ“REM”と、単一化候補節の連想検索を行う機能メモリ“PAM”について述べる。REM･PAMを用いた推論機械の構成も述べる。REMは書換えにより失われた内容の回復が可能なため、Prologの変数をREM上へ置くことにより、バックトラック時の変数の未代入化がメモリ側で自律的に行われる。そのため処理系は、バックトラックやそれに備える処理を省略できる。PAMは内部にPrologの節を保持する。ゴール節を与えられると、頭部の各引数がパターンマッチする候補節を連想検索しCPUに返す。全ての引数に対してクローズインデキシングを行ったのと同等な節を出力するため、失敗単一化を減少させることができる。

An architecture of inference machine based on ASIC memories

Kiyohiro Obara　Yusuke Sakai　Satoshi Igarashi　Yoshiyuki Kotani　Oichi Atoda　Nobuo Saito
Faculty of Technology, Tokyo University of Agriculture and Technology
2-24-16 Naka-chou, Koganei-shi, Tokyo 184, Japan

This paper discribes two Application Specific Integrated Circuit (ASIC) memories. One is REversible Memory (REM) which is random-access memory embedded function to recover contents. In backtracking, unbinding of Prolog variable on REM is automatically performed, because REM can recover contents lost by overwrite. Prolog processor is excluded backtrack task and trailing using REM. Another is Pattern Associative Memory (PAM), which retains database (Prolog programs). PAM recieves goal clause from CPU, and output matching alternative clause. The clause is equal to clause indexed one for all arities. Consequently, number of failure unifications is decreased. Finaly, inference machine based on REM and PAM is discribed.

1．はじめに

論理型言語Prologは、単一化によるパターンマッチ、バックトラック機能、宣言的な記述能力など他の著名な言語にはみられない機能を持ち、人工知能研究の分野では欠かすことのできないプログラミング言語である。このPrologを効率よく実行するアーキテクチャの研究は数多く行われてきた。特にWarrenによる、抽象命令セットを用いたコンパイラ技術[WAR 83]は、汎用計算機上でのProlog実現の有力な手段となっている。

コンパイルされたPrologプログラムは、トイプログラムや実行が決定的なプログラムに対しては効率よく動作する。しかし、知識処理など大規模な応用問題を記述した場合、適切な単一化候補節の検索やバックトラック処理などのコストが著しく増大してくる。これらはいずれもプロセッサとメモリ間の通信量の増加に起因するものである。WAMを直接実行するようなPrologプロセッサは、プロセッサの高機能化によりこの問題を解決する方法である。

我々はこの問題に対し、機能メモリ指向のPrologアーキテクチャを提案する。このアーキテクチャでは、Prologの処理の中で候補節検索やバックトラック処理などボトルネックとなっている処理を、機能メモリによって実現する。これによりプロセッサとメモリの機能格差を縮小し、いわゆるノイマンボトルネックの解消と、機能メモリ内部及び相互間の並列処理により、系全体としての処理効率向上を図る。我々はインタプリタの高速化を支援するアーキテクチャを意図している。これはインタプリタの高速化が、Prologの持つ会話性や柔軟性を失うことなく大規模プログラムを高速実行させる一義的な手段だからである。

我々は、上記のアーキテクチャ実現のため2種類の機能メモリを提案している。一つは、書換えによって失われた記憶内容の回復が可能な機能メモリであり、可逆メモリ（REversible Memory:REM「れむ」）と呼ばれる。このメモリは、Prologのバックトラック処理の高速化を意図している。もう一つは、単一化候補節検索を行う機能メモリであり、パターン連想メモリ（Pattern Associative Memory:PAM「ぱむ」）と呼ばれる。ここで述べられている機能メモリは、個々のメモリセルへ簡単な回路を付加したようなCAMのような形でなく、大容量のメモリセルとある程度複雑な制御回路を持ったものである。それゆえ現在は、TTL-MSIなどを用いて実現されているが、将来的には超々LSIやWSIなどの技術によりIC化可能である。

REM及びPAMの原案は、既報論文[川藤 87][安田 88]に見られる。現在のPAM及びREMは、Prologへのより効果的な対応のため、いくつかの改良がなされている。基本的な機能に変更はない。以降の章では、このアーキテクチャの中核をなす機能メモリである、改良されたREMとPAMについて述べる。最後に主題である、これらの機能メモリを用いたPrologマシンのアーキテクチャについて述べる。

2．可逆メモリ

可逆メモリ(REM)は、記憶内容の回復機能を組み込んだ機能メモリである。すなわちREMは、従来ソフトウェアにより行われてきたアンドゥの処理をメモリ自身に持たせたものである。REMの利用により、アンドゥの高速化のみならずアンドゥに備える処理をも含めて省略でき、システム全体の高速化と簡素化が達成できる。

2．1 外から見たＲＥＭ

REMは、通常のリード/ライトに関しては、アドレスを与えてデータをアクセスする通常のRAMとなんら変わるところはない。しかし、ライトされた内容を元に戻す動作をソフトウェアの指令で適切に実行するには、REMの内部と外部で論理的な時間、又は世代の概念を共有する必要がある。ここではこの世代の事を「エポック」と呼ぶ。エポックは、0から始まる整数値で表現される。REMは、通常のリード/ライトの他に、インクリメントエポック(INCEP)、デクリメントエポック(DECEP)及びその他の制御コマンドを受け付ける。これらのコマンドに対し、REMは次のように応答する。

過去に書き込まれた内容の読出しについては、通常のメモリと変わるところはない。すなわち、現在をエポックnとすると、エポック0からエポックnまでのいずれの世代に書かれた内容であっても、その後同じ番地にオーバーライトされていない限り、現在そのまま読み出せる。

ここで、デクリメントエポックを実行すると、メモリのすべての内容がエポックn-1の最後の時点、すなわ

ちn-1からnへエポックがインクリメントされる直前の内容へ戻る。すなわち、エポックnで書かれた内容は失われ、エポックnでのオーバーライトによって消失した記憶内容が回復する。ここでさらにデクリメントエポックを行うと、エポックn-2の最後の時点まで記憶内容が回復する。したがって、同じエポックで同じアドレスへ2回以上書き込んだ場合は、そのエポック内で最後に書いた内容だけが回復される。

インクリメントエポックは、その時点のREMの内容をそのエポックの最終状態とするコマンドである。これにより、エポックの値が一つ増え、以降の書換えは次のエポック内での操作となる。

REMはこの他に、イニシャライズ、マージエポックの各コマンドを持つ。イニシャライズは、エポックを0へ初期化するとともに、現在の記憶内容をエポック0の記憶内容とする。マージエポックは、指定したエポックから現在のエポックまでを一つのエポックへまとめる。現在のエポックの値は、指定したエポックと同じ値になる。REMの動作の概念を、図1に示す。

電気的には、REMは通常のマイクロプロセッサのバスに接続されるように設計されており、一般のRAMとなんら変わりはない。したがって、アドレッシングやリード/ライトは通常のメモリサイクルで行われる。プロセッサの立場からみれば、REMはメモリ空間とコントロール空間の二つの領域を持つペリフェラルとして見える。それぞれの領域は、独立したセレクト信号を持つためそれぞれ任意のプロセッサアドレスへ配置可能である。メモリ空間は、記憶内容の回復が可能な空間である。コントロール空間は、コマンドを与えるレジスタや、現在のエポックの値を保持するレジスタなどをまとめた領域である。コントロール空間を重ねることにより、複数のREMを同期して動作させることが可能である。

2.2 REMの容量

REMの容量は、次の三つの要素により定義される。

① 通常のメモリサイズと同じ意味の、アドレスバスからみたメモリ空間の大きさ。これを「間口サイズ」と呼ぶ。

② 復帰可能な情報の最大量。この大きさは、トレールスタック[川藤 87]と呼ばれる、記憶内容の履歴を保持する領域の大きさにより規定される。これを「奥行きサイズ」と呼ぶ。

③ エポックの最大値。

それぞれのサイズの大きさと比率は、応用目的に依存する。ここでは、Prolog処理系に適用した場合、すなわちProlog処理系の変数格納領域（例えば、WAMにおけるヒープ）を、考える。

「間口サイズ」は、処理系が解を導くまでに保持する変数を格納するに十分な大きさが必要である。具体的な大きさは、我々が行ったPrologプログラムの動的解析の結果、300KByte以上の大きさが必要とされる。これは、単体のLSIでは実現不可能であるが、複数個のREMを同期して動作させることにより、実質的な間口サイズの拡大が可能であるため、これに対して問題はない。

「奥行きサイズ」の決定は、間口サイズとの比が重要である。すなわち、間口サイズのn倍の奥行きサイズを設定することにより、このREMは、ある内容に対し平均n-1回の回復が可能である。Prologの変数は、単一代入規則により、いったん代入された変数の内容は変わることはない。内容の回復は、バックトラック時に1回必要となるだけである。したがって、Prolog処理系用のREMの奥行きサイズは、間口サイズの2倍で必要十分である。

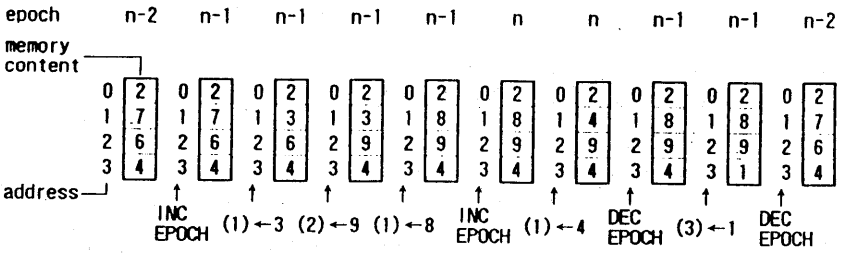

図1 REMの動作の概念。時系列の列（左から右へ）を示す

「エポックの最大値」は、実行するPrologプログラムの探索木の最深節点の深さ以上必要である。実際のプログラムの探索木の深さを測定するのは困難であるが、その大きさの上界の推定は可能である。動的解析における、ボックスモデルでのゴールのCALL回数、すなわち成功節点に到達するまでたどった探索木の大きさをもって上界とするものである。これによれば、Prolog処理系に使用する可逆メモリに必要なエポックの最大値は、約6万以上必要であると思われる。

2.3 REMの改良

REMのエポック操作は、単一化前にINCEPを行い、バックトラック時にDECEPを行うのが基本である。これにより、失敗した単一化による変数への代入がキャンセルされる。これを反映し、[川藤 87]のREMのコマンドは、イニシャライズ、INCEP、DECEPの3種であった。このREMをProlog処理系に適用した場合、カットを含むプログラムに対してうまく整合しないことが判明した。カットへの対応のため、マージエポックコマンドが導入された。

また[川藤 87]では、記憶内容の書換えを行わずにINCEPが行われた場合、REM内部の領域が無駄に使用され、エポックの最大値を制限する要因となっていた。現在のREMは、この無駄を省き、エポックの最大値をキロオーダーからギガオーダーへ拡大し、現実的なすべてのプログラムへ対応可能とした。

現在REMは、主にFタイプのTTL-MSIと高速S-RAMを用いて、400mm×230mmのユニバーサル基板2枚で実現されている。細かい仕様及び処理系への適用方法は、[小原 89]に述べられている。

3. パターン連想メモリ

パターン連想メモリ(PAM)は、論理型言語向きの候補節検索機能を持った連想メモリである。現在、その対象としてはPrologを考えている。PAMを用いたPrologシステムでは、プログラム（いわゆるデータベース）はPAM内に格納される。PAMは外部から検索キーを与えられ、それと単一化可能な節を自身の中から検索し出力する。この意味では、PAMは一種のデータベースマシンのようにも見ることができる。このことにより、通常の検索で生じるプロセッサとメモリ間で大量のデータが行き交うボトルネックを回避し、PAMとプロセッサが並列に処理を行うことにより系全体の効率化高速化を図る。

3.1 節の表現

PAMは、Prolog向きの候補節検索機能を持った連想メモリである。したがってPAMにおいて、検索のキーはPrologにおけるゴール節である。被検索データ、すなわちPAM内に記憶されるデータはホーン節であり、これはPrologのプログラムに他ならない。節は、各項を項目とし、アリティを構造長とする構造体として表現される。このとき、定数項及び変数項はそれぞれ定数項目、変数項目として表現し、複合項はその構造体本体へのポインタ項目として表現する。PAMは、構造体表現された節を基本データとして処理する。データ型は前述のように、定数、変数及びポインタ型の3種である。節の構造体表現の例を、図2に示す。

PAMは、ハードウェア的にはCPUに対してI/Oとして結合し、CPUからはアドレスではなくゴール節が与えられる。PAMは、これとマッチする候補節を内部で検索しCPUに返す。この動作の概念図を図3に示す。

候補節検索に必要なのは、節の頭部だけである。構造体中で、頭部リテラルを連想項目と呼び、その長さを連想項目長と呼ぶ。本体は、PAM内に記憶されるが、パターンマッチから除外される。さらに、構造体には、

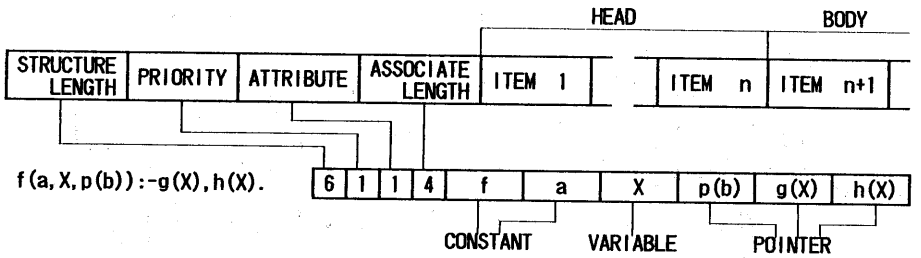

図2 PAMの記憶単位（構造体表現された節）

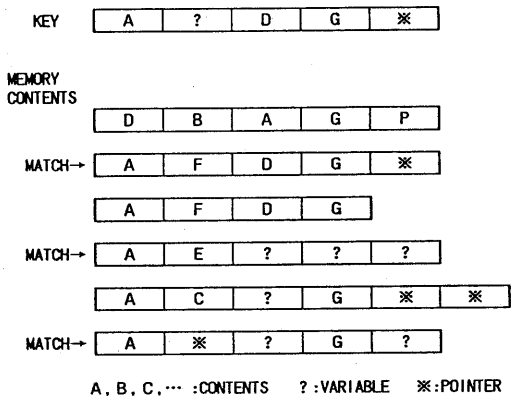

図３ パターンマッチングによる構造体検索の概念

プライオリティとアトリビュートを付加する。プライオリティは、候補節検索時の検索順序を指定する。同じプライオリティを持つものは先に入力されたものから出力される。アトリビュートは、節をグループ分けする場合に用いられ、述語名の有効範囲の指定、いわゆるモジュール宣言の実現に利用できる。

３．２ ＰＡＭのオペレーション

PAMへの書込みによって節がアサートされる。書込みや読出しなどの基本オペレーションは、WRITE、READ及びDELETEの3種である。

WRITEオペレーション時には、PAM内で内部データベースの一貫性管理が行われる。すなわち、外部から与えられた節を同一節の二重定義のチェックをした後に書き込む。DELETEオペレーションは、与えられた節と同じものを内部データベースから消去する。

READオペレーションでは、候補節の検索読出しが行われる。PAMは、構造体表現されたゴールリテラルが連想項目長と共に与えられると、連想項目長が一致し、かつ各連想項目がパターンマッチする節を候補節として出力する。項目のデータ型に対するパターンマッチ規則は、

(1) 少なくとも一方が変数項目ならば一致。
(2) 同一の定数項目同士ならば一致。
(3) ポインタ項目同士ならば一致。
(4) 定数項目とポインタ項目は不一致。

であり、Prologの単一化可能性の条件を反映している。したがって、このパターンマッチにより、PAM内部で候補節の検索と共に単一化可能性のチェックの一部を行ってしまう。このため、単一化処理の前段階で候補節が絞り込まれ、不成功単一化の回数を減らすことができる。完全な単一化可能性のチェックには、節内部の共有変数の一貫性チェック及びポインタ項目同士が一致したとき、すなわち複合項同士の単一化可能性チェックが必要である。しかし後者は本質的に逐次的な処理であり、前者に対しては現在連想読出しと共存できる適切なアルゴリズムを得ていない。いずれにせよこれらは逐次処理しなければならず、PAM内部に組み込む利点を得られない。それよりも、これらの処理を除外することにより、PAMの構造を簡単化した方が得策であると考えられる。これらの処理は、PAMから候補節を読出した後、外部で行えばよい。

３．３ ＰＡＭの内部構造

PAMの内部構成を、図４に示す。PAMは節に固有の番号を付ける。これを節番号と呼ぶ。節は節番号を付加され、PAM内のシーケンシャルテーブルに格納される。このほかにPAMは、連想検索に必要なため節の頭部の写しを取り、構造体の各項目（すなわち述語名と引数）ごとに分解し、それぞれ節番号と連想項目長を付加し、項目と連想項目長をキーとして独立したHash Tableへ格納する。これは例えば、述語名はHash Table #1、第一引数はHash Table #2、第二引数はHash Table #3のように格納される。このことにより、項目間の並列ハッシュによって引数間並列による高速な候補節検索が可能となる。

候補節検索は次のように行われる。統括制御プロセッサ(CP)は、与えられたゴール節を引数、すなわち項目ごとに分解する。次にそれぞれの項目に連想項目長を加えてキーを作り、それぞれの項目に対応するテーブルプロセッサ(TP)で独立かつ並列にハッシュを行う。

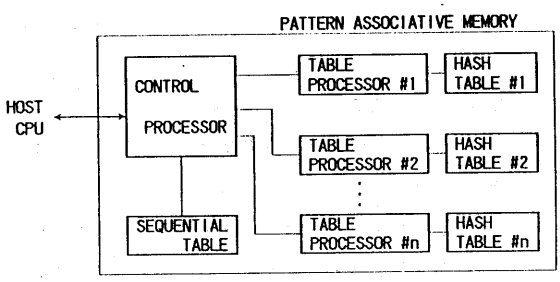

図４ PAMの内部構成

次にCPは、ハッシュ読出しを行った各TP間ですべてに共通して存在する節番号を見つけ出す。一致した節番号は、シーケンシャルテーブルを参照することにより、完全な節の形に変換され、候補節として出力される。これは見かけ上、すべての引数に対してクローズインデキシングを施したように見える。

ここでは、PAM内の制御構造を簡単に述べた。実際は、ハッシュテーブルの構成法や節番号の一致照合などに特筆すべきアルゴリズムが用いられている。それらは[安田 88]に詳述してある。なお[安田 88]でタグと呼ばれているものが、ここでの節番号である。

3.4 PAMの改良

PAMの基本的な機能は、[安田 88]で述べられたとおりであるが、現在のPAMはPrologへのより効果的な対応のためいくつかの改良がなされている。次にこの改良点を述べる。

初期のPAMは、節の頭部と本体へのポインタを格納していた。節の本体は外部での存在、すなわちホストとなるプロセッサが管理することを前提としていた。しかし、Prolog実行系全体として考えた場合、PAMには連想検索のみならず、グローバルな意味での節管理機能をもたせるべきだとの結論を得た。すなわち、連想検索、アサート/リトラクト処理及びそれらに伴う一貫性管理機能などである。このため、現在のPAMは節全体を保持している。

さらに、RESUME機能も廃止してある。RESUME機能は、最後のREADオペレーションの一つ前のREADオペレーションにおける次候補を検索出力する機能である。これは、いわゆるディープバックトラック時の候補節検索再開に対応する。この機能の実現のためにPAMは内部にスタックを持ち、成功した候補節の節番号と中間情報を保持していた。しかしこれらの情報は本来処理系が管理すべき情報であり、PAMが持つべきではないと考え、RESUME機能を廃止した。このため検索結果と併せてTP内での検索の中間情報を出力し、ディープバックトラック時にはその情報により迅速な検索再開ができるように改良した。

この他に、現在実装試験中の機能をいくつか述べる。一つは、節番号とゴール位置によるインデックス連想検索である。これは、PAM内に節がすべて格納されていることを利用し、外部からキーとなるゴール節を与える代わりに、そのゴール節を節番号とゴール位置により指定する方法である。これにより、CPU−PAM間のバストラフィックはさらに減少できる。変数の内容も同時に与えるのは言うまでもない。

もう一つは、候補節先読み機能である。これは、与えられたキーに対する候補節を出力した後も、PAM内部で外部と並列に次候補の検索を継続し、シャロウバックトラック時の高速な次候補の提示を実現するものである。

PAMはProlog向きの連想検索、すなわちゴール節を与えて単一化候補節を出力する機能メモリであるが、その機能拡張や改良を考えすぎると、質問を与えて結果を返す、すなわちPrologマシンそのものまで拡張されてしまう。PAMへの機能の切り分けは、系全体の処理効率向上という観点から十分な考察と評価からのフィードバックが必要である。

4 機能メモリに基づく推論機械

4.1 ハードウェア構成

REM及びPAMに基づくProlog実行系の構成図を図5に示す。このシステムは、PAM、単一化部及び実行管理部の三つのモジュールから構成される。モジュール同士は、2ポートのRAMにより一対一接続されており、相互に割り込みも可能である。各モジュールは専用のシーケンサでの構築が理想だが、現在は次のような構成となっている。

PAMは、CP及びTP共にクロック速度8MHzのMC68000をCPUとするシングルボードコンピュータよりなる。各Hash Tableは、0.5MByteの大きさを持つ。

単一化部は、クロック速度8MHzのMC68000と間口8K

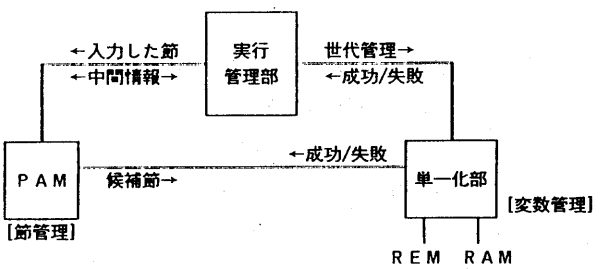

図5 機能メモリに基づくProlog実行系の構成

ByteのREM及び0.5Mbyteの通常のRAMから構成される。

実行管理部も、クロック速度8MHzのMC68000をCPUとするシングルボードコンピュータよりなる。ローカルメモリは、0.5MByteの容量である。

各モジュールのメモリ容量は小さいが、システムの安定にともない順次拡張する予定である。

4.2 作業領域

このマシンは、Prologをインタプリタ方式で実行する。インスタンスの表現方法は、構造複写法を採用する。作業領域は、次のようになる。データベースすなわちPrologプログラムは、PAM中におかれる。アサート/リトラクト等のデータベースの操作もPAMが行う。スタックは、Control、Header、Structure、Variableの4本である。トレールスタックは、REMがその任を行うので必要ない。Controlスタックは、処理系の実行を制御するための情報がおかれる。Headerスタックは、WAMで言うところのスタックに相当する。StructureスタックとVariableスタックは。WAMで言うところのヒープに相当する。

Controlスタックは、実行管理部へ置かれる。Variableスタックは、REM上に取られ、変数だけを格納する。Structureスタックは、単一化部中の通常RAM上へ取られ、変数以外の固定項を格納する。Header、Structure、Variable各スタックによるインスタンスの表現方法を図6に示す。Structure、Variableの区別をなくし、ヒープとして、すべてREM上へ確保することも可能でありかつ簡単である。しかし、不変データをREM上へおくのは無駄である。REMの効率的な利用のため、このような表現方法を採用する。

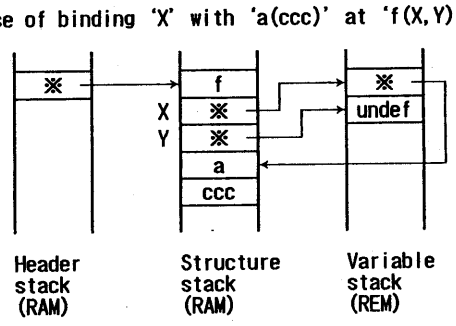

図6 三つのスタックによるインスタンスの表現

4.3 実行方式

このマシンの実行方式は、基本的には、通常のインタプリタと同じである。ただ実行管理部に置かれる処理系は、従来の処理系と比較して簡素なものとなる。これは、節管理はPAMに、変数管理及び単一化はREMに委譲されているからである。

ここの処理系が行う主な仕事は、Controlスタックまわりの仕事である。すなわち、通常の実行制御情報管理の他、REMの世代管理情報の管理、ディープバックトラック時の次候補検索再開のための中間情報の管理などである。また実行管理部は、外部との入出力、数値計算なども行う。

単一化部は、単一化作業に加え、REMの操作、Header、Structure、Variable各スタックの管理を行う。単一化は、PAMから提示された節に対して行われる。単一化失敗時には、REMのDECEPを行い、PAMへ次候補の出力を指令する。シャローバックトラックは、単一化部とPAM間の閉じた世界で行われる。このことは、処理系のシャローバックトラックに対する考慮が不用であるのみならず、単一化と候補節検索の並列動作により、より高速なシャローバックトラックが可能である。単一化成功時には、REMのINCEPを行い、PAMと実行管理部へ報告する。

このマシンは、各モジュールのコアに汎用プロセッサを用いているため、タグジャンプなど、プロセッサ側での特別な仕掛はない。これは、このような機能が必要ないと判断したわけではない。タグの導入のように、性能向上が定量的に推測できる機能よりも、処理の機能メモリ化による分散並列化の研究に大きな興味を持つからである。

5. おわりに

書換えによって失われた記憶内容の回復が可能な機能メモリである可逆メモリ(REM)と単一化候補節検索を行う機能メモリ(PAM)の動作とその改良について述べた。最後にこれらの機能メモリを用いたPrologマシンについて述べた。現在このマシンは、ハードウェアが完成し、ソフトウェアの実装中である。

謝辞

本研究を行うにあたり多大なご協力を頂いた日立超LSIエンジニアリング㈱酒井要部長に深謝する。また可逆メモリの利用法に関して討論ならびに有益なご助言を頂いた元本学大学院横山大先生に深謝する。

参考文献

[川藤 87] 川藤 他:可逆メモリ，信学論(D)，J-70D，12，pp.2793-2795(1987).

[小原 89] 小原 他:可逆メモリのProlog変数管理への応用，信学論(D-I)，J72-D-I，2，pp.136-139(1989).

[坂井 90] 坂井 他:機能メモリに基づくProlog処理系の構成，情報処理学会記号処理研究会資料56-5，1990.

[WAR 83] Warren, D. H. D. (1983):AN ABSTRACT PROLOG INSTRUCTION SET, Technical report 309, Artificial Intelligence Center, SRI International.

[安田 88] 安田 他:「パターン連想メモリ」とその論理型言語処理系への応用，信学論(D)，J-71D，9，pp.1614-1622(1988).