計算機アーキテクチャ 46－4
（1982. 9. 17）

PROLOGマシン・アーキテクチャの一考察

梅村 護　　横田 実*
（日本電気 C&Cシステム研究所）

1. はじめに

計算機を人間の知的活動のサポートの為に利用しようとするいわゆる知識情報処理研究の分野は，近年富に関心が持たれ，各方面で研究が進められている。述語論理型言語PROLOGは，英国のエジンバラ大学のWarren等によってDEC-10上にインプリメントされ[1]，にわかに実用性が高まった。PROLOGは，例えば自然言語における文法規則を基礎とした処理系を構築するルールベーストシステムに非常に良くマッチした言語であり，将来の知識情報処理分野で大いに利用されることが期待されている。[2]

筆者等は，PROLOGを実際問題に適用することを目指し，SNOBOL[3]流の文字列パターンマッチ機能を導入した新しい言語ShapeUpを作成した[4]。ShapeUpの基本機能はPrologと同様であり，現在までに言語Cによってそのインタープリタを作成した。当所では，このインタープリタを用いた機械翻訳システムTRAPを構築し[5]，その有用性を確認している。

PROLOGおよびShapeUpの実行メカニズムは，従来型の汎用計算機に必ずしもマッチしていない。このため，汎用機上では充分な処理性能が得られず，実用システムを構築するためには専用のマシンアーキテクチャが是非とも必要となる。本論では，そのマシンアーキテクチャの一方式について提案する。

本報告では，先ずPROLOGとShapeUpの言語仕様について概観し，次にShapeUpを専用に処理するマシンアーキテクチャDAWNについて述べる。

2. PROLOGとShapeUp

PROLOGでは，"述語"を用いて事物を表現し，述語間の関係を記述するクローズ（節）の集まりによって論理的な"世界"を創り上げる。この集まりをデータベースと呼び，ユーザがデータベースに対して問合わせを発生すると，自動的に"世界"の中の論理を展開して何らかの応答を返すシステムである。ShapeUpはDEC-10版のPROLOGを基本とし，それに文字列パターンマッチ機能と，図形処理機能を導入して拡張した言語である。以下DEC-10版PROLOGのノーテーションに則してPROLOGおよびShapeUpの言語仕様を簡単に述べる。

2.1 言語形式

述語は次の表現形式を持つ。

述語名(引数1, 引数2, ---)

そしてクローズの一般形は，この述語を用いて次のように書かれる。

述語1 :- 述語2, 述語3, --- .

クローズでは，記号:-の右側に書かれる述語が真であれば，その左側が真であることを表わす。:-の左側を頭部（ヘッド部），右側を本体部（ボディ部）と称する。PROLOGおよびShapeUpでは，ヘッド部には高々1つの述語しか書けないという制限（Horn クロー

---

* 現在（財）新世代コンピュータ技術開発機構(ICOT)

ず）がある。　ボディ部の"，"は、述語2，述語3，---が論理和で結ばれることを意味し、述語の数に制限はない。

この一般形に対し、ヘッド部のみから成るクローズを宣言節，ボディ部のみから成るクローズを内合わせ節，ヘッド部，ボディ部を共に持つクローズを規則節と呼ぶ。プログラムは宣言節と規則節を用いてPROLOGの"世界"を構築することから始まる。こうして創られた"世界"に内合わせ節が与えられると、システムは述語間の関係をたどり、結論を導く。

2.2　ShapeUp の特徴

DEC-10 版PROLOG およびShape-Up ではリストを［a,b,c］の形式で表わす。　PROLOGでは、例えば"ies"で終わる文字列を表現するには、リストを用いて下記の2つのクローズで表現する。

```
語尾（［*X，i，e，s］）.
語尾（［*X｜*Y］）:- 語尾(*Y).
```

ここで＊付の引数は変数を表わし、｜は二進木を表わす。　一方、ShapeUpには文字列パターンマッチ機能が備えられており、上の例は次のように単一のクローズで表現することができる。

```
語尾（*X^ies）.
```

記号^は文字列の結合（Concatenation）を示す。　ShapeUpのパターンマッチオペレータには結合の他に以下のものがある。

｜	選択肢
＿	任意の1文字
～	任意長の任意文字
n{}	n回のくり返し
*変数{パターン}	変数への代入

これらのオペレータを用いて、文字列データ自体の分解，結合を自由に扱うことができ、テキスト処理や自然言語処理に有効である。　また、ShapeUpの文字列処理向機能の拡張のため、要素数可変の配列データを導入した。これを用いることによって可変長のテキストを一次元の配列に保存して扱うことが可能となる。　ShapeUpではこの配列にパターンマッチ機能を施すことができ、テキスト処理等を非常に直接的に記述できる。

ShapeUpは文字列と共に図形データを導入していることが特長的である。図形データの表現法は、ビットマップイメージで持つ方法，グラフィック端末等が備えるインタフェースを利用して定形的図形を表現する方法があり、必要に応じて使いわける。

Prologにおいては、変数のスコープは各クローズ内で閉じており、クローズ間では独立な変数として扱われる。このため、変数に割当てられる値の履歴をとる直接的手段が無い。　しかしながら、実用上はクローズ間で共有できる変数が必要となることがある。ShapeUp にはこの機能を満たすグローバル変数を導入した。　これは、従来のプログラミングスタイルへの逆戻りとも考えられ、扱いには注意が必要であるが、実用上の便利さを考慮して導入したものである。　この種の拡張として、プログラムのモジュール化の為の機構もとり入れている。

3．言語処理手順

ShapeUpの言語処理手順は、基本的にはDEC-10版PROLOG と同様である。[1]
プログラムは、述語間の関連性を記述する規則節と宣言節から成っており、内合わせ節の入力によって実行処理が惹起される。　実行手順は、ユニフィケーションと呼ぶ述語同士の照合と、

照合失敗時の後戻り（バックトラック）処理の2つであり、これらがプログラムで記述される述語間の関係をたどるべく適用されることで成就される。

処理手順は以下のステップで示すとおりである。

(1) 呼び出し側（ボディ部）の述語と同じ述語をヘッド部に持つクローズを探し、見付かれば(2)へ，見付からなければ(3)へ。

(2) 引数同士の関連付けを行い、呼び出されたクローズが宣言節であれば処理を終了し、(4)へ。そうでなければそのボディ部の述語を呼出し側として(1)へ。

(3) ボディ部の対象述語の直前に述語があれば、それを他の選択肢によってユニフィケーションを行う（バックトラック）ため(1)へ。もしなければ「失敗」として呼出し元へ制御を戻す。

(4) 対象述語の直後に述語があればそれを呼出し側として(1)へ。もしなければ動作を終了して呼出し元へ制御を戻す。

ステップ(2)の引数同士の関連付けでは、呼出し側 呼出された側の引数をその並び順に照合する。引数同士が共に定数のときには値が一致していなければユニフィケーションはその時点で失敗に終る。もし、一方が変数であれば対応する定数値がセットされる。もし双方共変数であれば、それらは同値である事を示す為、リンク付けされる。

以上のステップを、図1の例を用いて具体的に説明する。

図1の①，②は規則節，③～⑤は宣言節，⑥は問合わせ節である。②が入力されることにより、その述語である P(B,*Y) が呼び出し元となって

```
① P(A,*X) :- Q(*X).
② P(B,*X) :- Q(*X), R(*X).
③ Q(C).
④ Q(D).
⑤ R(D).
⑥ :- P(B, *Y).
```

図1. プログラム例

ステップ(1)が実行される。即ち、先ず⑥と①の照合が試みられるが、引1の引数BとAが不一致の為、次のクローズ②との照合が試みられる。ユニフィケーションは成功し、ステップ(2)へ進む。(2)では *Y と *X がリンク付けされた後、②のボディ部の最初の述語 Q(*X) を呼出し側としてステップ(1)へ戻る。(1)では③とのユニフィケーションが成功し、(2)で *X には値Cがバインドされる。③は宣言節の為ステップ(4)へ進む。いま、対象述語は②のQ(*X)であったので、次にはその直後の述語 R(*X) を呼出し側として(1)へ進む。但しここで *X はQの *X と共通の為、呼出しは R(C) で行われる。これに一致するクローズは存在せず、ステップ(3)が実行されて、Q(C) がバックトラックされる。バックトラック時には先立ってバインドされていた値はキャンセルされ、他の選択の余地のある探索点まで処理が戻される。本例では、Q(C) がキャンセルされ、Q(*X) で再びユニフィケーションが試みられ、④が見付けられる。ステップ(2)で *X にはDがバインドされ(4)へ進む。次に R(D) が呼出し側となって(1)，(2)へと処理が進み、⑤と一致し、それが宣言節の為、④へ進む。②で R(*X) の次には述語が無いので、これを呼出した⑥の、

P(B,*Y) へ制御が戻る。 そしてこの述語もその節の最終述語であるので処理はすべて終了した。 ここで変数 *Y には、リソク付けによって値Dがバインドされて戻される。

4. マシンアーキテクチャDAWN

前節の説明から明らかなように、処理系の動きを要約すれば、クローズ内の述語名とその引数を記述するエレメントを備え、エレメント間の関連性をポインタチェーンで結んでおいて、これを次々にたどっていくことになる。各エレメントはプログラムを反映するものであってメモリ内に置かれる為、処理装置は、メモリをアクセスしてエレメントを読出し、エレメント内のポインタを番地として再びメモリをアクセスするという単純な動作の繰返しに終始する。

従来型の汎用計算機では、通常メモリアクセスは負荷が大きく、上記のような処理系をその上で実行しても効率は上がらない。 Shape Upの高速実行のためには、述語間の関係記述をできる限りコンパクトにし、そのアクセスが高速に行えるような専用のアーキテクチャが必要である。

マシンアーキテクチャDAWNはこの目的を達成するものであり、クローズを表形の内部形式で記述し、スタックサポート機構と、表を高速にアクセスできるようなメモリ機構を用いてShape Up を直接実行するアーキテクチャである。

Shape Upのクローズはそれ自身で完結しており、変数のスコープも1クローズ内で閉じている。 したがって1クローズの入力毎に直ちに構文解析を行い、内部形式に変換することができる。 本アーキテクチャは、このセミコンパイル方式を採用し、インタラクティブなインタフェースをユーザに提供すると同時に、内部実行はコンパイルされたオブジェクトを用いて高速な処理を実現している。

4.1 内部形式

DAWNの内部形式は、図2のような5種類の表から成っている。

- プロシージャテーブル(PT)

  述語間の関係をたどるとき、キーとなるのは述語名である。 同一の述語をヘッド部に持つクローズの集合をプロシージャと呼び、ボディ部に現われる述語によってプロシージャがコールされることで実行が進む。PT エントリはプロシージャ毎に1つ作られ、他PT エントリへのリンク、述語名(又はポインタ),プロシージャの先頭クローズの内部形式CT エントリへのポインタの各フィールドから成る。Tagは表idを示す。

- クローズテーブル(CT)

  CT エントリは、プログラムのクローズ毎に1つ作られる。 フィールド構成は、他のCT エントリへのリンク、ヘッド部の引数を記述する内部形式HT へのポインタ、ボディ部を記述する内部形式BT へのポインタである。

PT	Tag	LINK	P-NAME	CTP
CT	Tag	LINK	HTP	BTP
HT	Tag	LINK	Arg.	
BT	Tag	LINK	P-NAME	ATP
AT	Tag	LINK	Arg.	

図2. DAWN内部形式

・ヘッド引数テーブル（HT）

HTエントリは、ヘッド部の引数毎に1つ作られる。同一述語内の他の引数に対応するHTエントリへのリンクフィールドと、引数表現フィールドから成る。

・ボディテーブル（BT）

BTはクローズのボディ部の述語を表現する内部形式である。同一ボディ内の他のBTエントリへのポインタ、述語名、引数テーブルエントリへのポインタから構成される。

・引数テーブル（AT）

ボディ部述語の引数毎に1つのエントリが作られる。同一述語の他の引数を表わすATエントリへのリンクと、引数表現又はポインタのフィールドから成る。

これらの表を用いて各クローズが表現される。図3は、

likes(John, *X) :- isred(*X).

というクローズを内部表現する各表エントリ間のリンク付けを示している。各表エントリは、各クローズが入力される毎に創られる。

4.2 実行メカニズム

PROLOGおよびShapeUpの実行では、変数への動的な値の割当てと、バックトラック時の再割当てが必要であり、スタックを用いる。DAWNでは機能別の4種のスタックを用意している。

- グローバル変数スタック
  構造体の引数を表現する変数の値を格納する。
- ローカル変数スタック
  上記以外の変数の値を格納する。
- トレイリングスタック
  バックトラック発生時に、それ以前に行われた変数への値のバインドをキャンセルするためのスタック。このスタックの各セルはバインドが行われた変数セルのアドレスを格納している。
- コントロールスタック
  クローズの呼出し処理を完了したときのスタックのポップアップ、あるいは、バックトラック発生時のスタック状態の復帰を行うために必要な各種スタック管理情報を格納するスタック。

以下に例を用いてスタックの基本的動作を説明する。

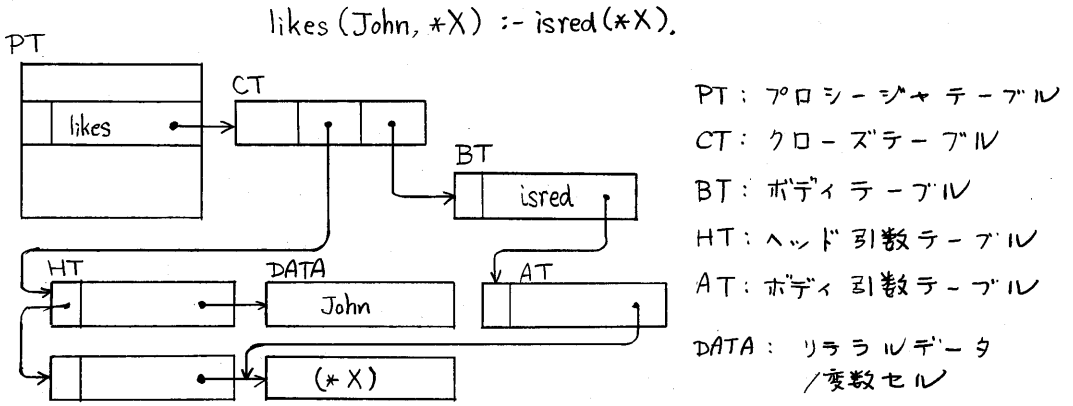


図3. プログラムの内部形式表現

[プログラム例]

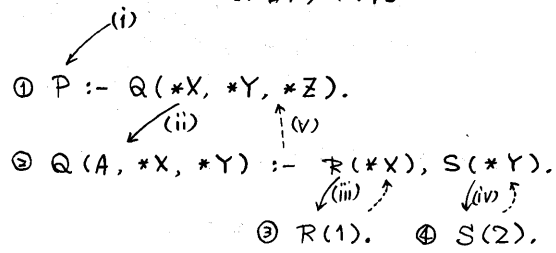
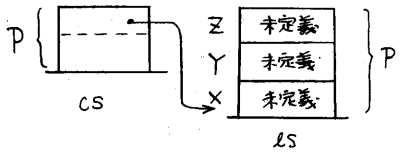


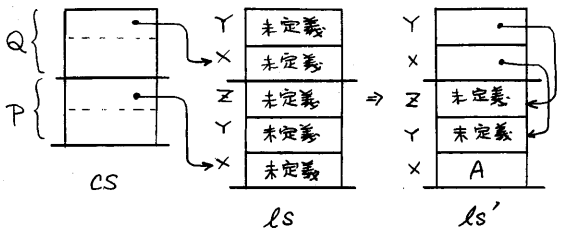
図4. 実行の流れ

図4の(i)で、クローズ①が呼出されると、このクローズに含まれる3つの変数のために、ローカル変数スタック(ls)上に3つのセルがとられる。各



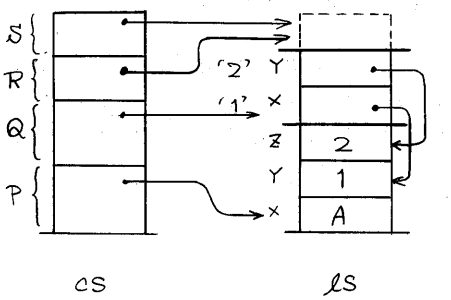
セルの初期値は未定義がセットされる。コントロールスタック(cs)には、制御情報の1つとしてlsのスタックフレームベースのアドレスが格納される。上の図は、この時のcs, lsの状態を示す。

次にクローズ①のボディにより、述語Qの呼出しが指示され(ii)、クローズ②が呼出されると、②のための変数領域がlsに積まれる。



ここで、呼び出されたクローズ②のヘッド部引数A, *X, *Yと①の呼出し元引数*X, *Y, *Zの間でユニフィケーションが行われる。その結果、クローズ②の内部形式に含まれる定数Aが第1引数値としてPの*Xの位置にセットされ、②の*X, *Yは夫々①の*Y, *Zと対応するので、同一の値を採るべきことを表現するため、ポインタで結合される。すなわち、スタックの内容は上図のlsからls'の状態に進む。

ここまでで①のボディと②のヘッドのユニフィケーションを終了し、②のボディ部処理へ移る。この述語呼出し(iii)および(iv)は、①と同様に行われるが、変数の新たな生成はなく、クローズ③および④の内部形式からリテラル定数'1'と'2'とが引出され、夫々Qの*X, *Yを介しPの*X, *Zにセットされる。変数が存在しなければlsにはフレームは作られないが、csには呼出し毎に制御フレームが積まれる。ここまでのスタックの状態を下図に示す。



述語の呼出しが完了すると、その都度スタックのポップアップが行われ、呼出し前の状態に復帰する。PROLOGの特徴は、このとき完全にスタックをクリアせず、バックトラックに備えて保存されることである。バックトラック時には、再試行は選択肢を含む位置を保持するトレーリングスタックを用い、中途からの再呼出しが行われる。上の図でQ, R, Sに対応する部分がポップアップされても、スタックフレームはバックトラックの可能性を含む限

り消去されない。

4.3 メモリシステム

ShapeUpの実行は上記説明のように、表形式の内部表現で示される述語間の関係を、スタック機構を用いて次々とたどりながら遂行される。DAWNのメモリシステムは内部形式に合致した構成を持ち、高速なユニフィケーション処理とスタック処理を実現する。

内部形式が表型であり、各エレメントが夫々固定長であることを保証することによって以下に述べるようなメモリ構成が可能である。すなわち、図5に示すように、各テーブルにユニークな論理空間を与え、メモリアクセスは〔テーブルid＋テーブル内オフセット〕で行うものとする。内部形式はテーブルid毎に、特定のフィールドがユニークなテーブルへのポインタであることが定まっているため、ポインタフィールドにはオフセットのみを格納すればよい。したがって内部形式のフィールド長も短かくすることができ、1エントリ内に複数のポインタを含み得ることになる。

図5は、論理空間を2^{16}エントリ分としたときのアロケーション法の一例を示している。この構成によれば、各テーブルのエントリ数が65,535を超えない限りは、内部形式のポインタは16ビットのフィールドで保守することができる。したがって内部形式は非常にコンパクトな表現法となり、プロセッサも16ビットを基本とする構成で充分で、システム全体を非常に簡素にまとめ上げることができる。2^{16}エントリを超える空間を必要とするときには、マッピングを一段導入することで実現可能である。

DAWNのメモリシステムは、内部形式を専用に格納するキャッシュと、実行管理用スタックおよびデータを格納

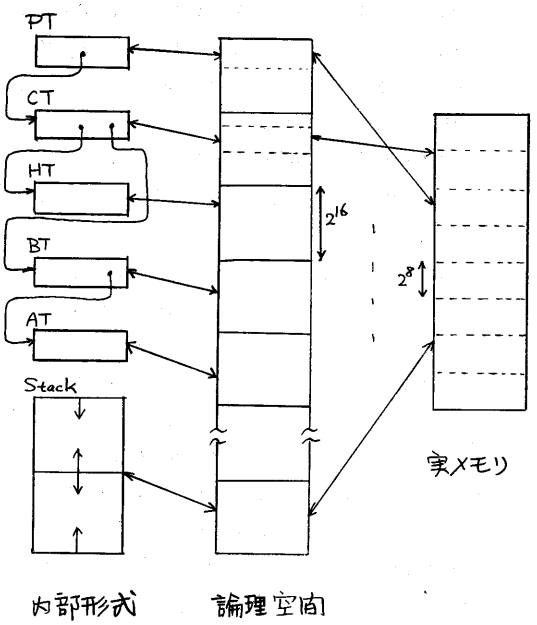


図5. メモリアロケーション

するキャッシュの2種のキャッシュメモリを設ける。内部形式の特定のフィールドは、次にアクセスすべき内部形式エレメントへのポインタであることを利用し、内部形式格納キャッシュの読出しデータのポインタフィールドは、アドレスレジスタに直結され、読出すと同時に次のエントリのアクセスシークエンスが始まるよう構成する。これによって非常に高速なポインタリンク経由のメモリアクセスが実現される。スタック用キャッシュメモリは言うまでもなく実行の基礎となるスタック動作を高速化するためのものである。これらキャッシュメモリは夫々独立にアクセスされ得ることが望ましい。

ShapeUpの実行環境としては、メモリが非常にダイナミックに運用されることを想定している。したがってメモリのアロケーション／リアロケーション法にも工夫が必要であるが、こ

こで再びエレメント長が固定であることが利用できる。すなわち、物理アロケート時に、フリーエレメントリストを単位として扱う。

上記動作をサポートするために、メモリシステム制御部は以下のような命令実行を行う。

- new　フリーリンクを生成し、フリーリストの先頭にリンクする。
- get element　フリーリストから1つのエレメントを取り、フリーリストからはずす。
- put element　エレメントをフリーリストへ返却する。
- reallocation　メモリ再構成。
- read　メモリリード。
- write　メモリライト。
- read + next　読込んだエレメントのリンクフィールドにより、直ちに先読みシークエンスを起動する。
- next-read　先読みデータを返す。

4.4 制御プロセッサ

DAWNの要は上述のメモリシステムにあり、プロセッサの負荷はむしろ軽いと言って良いであろう。プロセッサの役割は、ユニフィケーション処理とクローズ呼出しの制御である。具体的には、内部形式のリンクをたどったメモリアクセスの制御と、ユニフィケーションにおける比較、照合動作およびスタックの制御である。

スタック制御を高速に実行するため、プロセッサは複数個のレジスタスタックファイルとも呼ぶべきレジスタ群を多用する。更に複雑にネストされた構造体データのユニフィケーション等の為に、リカーシブな制御をサポートすることが必要で、この目的にもレジスタスタックを利用する。

また、ShapeUp独自の機能である、文字列パターンマッチを遂行するためのハードウェアと、図形処理の為の特殊なインタフェースも備える。

5. おわりに

PROLOGは、第五世代計算機システムプロジェクトで取上げられて注目され、将来の知識情報処理分野で大いに利用されることが期待されている。しかしながら、PROLOGの上に実用システムを構築するには、言語面、並びに処理系実現面で尚一層の研究が必要であろう。

本報告では、PROLOGの具体的な応用として自然言語処理、テキスト処理を想定し、拡張した言語ShapeUpの概要を述べ、更にその専用処理マシンアーキテクチャDAWNのアウトラインについて述べた。筆者等は、このアーキテクチャDAWNをもとに、ShapeUp処理専用パーソナルマシンを構築すべく検討を進めている。

最後に、日頃有益な御指導、御助言を下さる当研究所、CS研究部箱崎部長、山本課長に深謝します。

参考文献

1) Warren, D.H. "Implementing PROLOG compiling predicate logic programs", Dept. of A.I. R.R (40) Univ. Edinburgh '77.
2) 淵「記号処理専用マシンへの期待」情報処理, Vol.23, No.8 '82.
3) Griswold, R.E. et al. "The SNOBOL4 Programming Language 2nd Edition", Prentice-Hall '71.
4) 横田, 梅村「PROLOGの記号処理への機能拡張」情処学会記号処理研究会資料19-2 '82.
5) 市山, 村木「機械翻訳に関する1つのアプローチ」情処学会自然言語処理研究会資料31-6 '82.