計算機アーキテクチャ　71－15
(1988. 7. 8)

セルラシステムによるＰｒｏｌｏｇの実現について

PROLOG IMPLEMENTED BY CELLULAR SYSTEM

山川　直己　　吉岡　良雄
Yamakawa NAOMI　　Yoshioka YOSHIO
岩手大学・工学部・情報工学科，盛岡市
Department of Computer Science, Faculty of Enginerring,
Iwate University, Morioka-shi 020 Japan

あらまし　PrologはLispと同程度のリスト処理能力を持ち、推論機能が内蔵された言語である。セルラシステムの持つ構造に由来する並列性を利用して、このPrologの持つ並列性を引き出し、高速実行可能なシステムを提案する。本システムは、同一化を行うセルラシステムと、動的なプログラムのマッピングを受け持つセルラシステム、この二つのシステム間のデータの授受を行うユニットからなっている。これらの論理的な構成について述べ、静的な性能評価を行う。

Abstract　Prolog is as capable of processing lists as Lisp and a language with inference engine. In order to obtain the advantage of concurrence that results from cellular system structure, we propose a high-speed Prolog system with concurrence. This system is consisted of three blocks, one is unification system and the second is a mapping system of dinamic programs and the last is a communication unit between two cellular systems. This paper presents the system topologies and the static performance estimate.

1．はじめに

人工知能が実用面で注目され始めたのは、エキスパートシステムが出現してからである。このエキスパートシステムの推論機構の部分は記号処理言語で記述されることが多いが、特に、Lispが採用されることが多い。Lispは強力な言語であるが、色々な関数を定義しなければならない。しかし、Prolog[1][2]はLispと同程度のリスト処理能力を持ち、推論機能が内蔵されている。また、Prologは並列性を持つ言語であり、その並列性を考慮したシステムが提案されている。一方、セルラシステムはその構造自体に並列性を内包している。

本報告では、このようなセルラシステムの並列性を利用したPrologの処理方式を提案し、その論理的な構造について述べ、簡単な性能評価を行う。

2．Ｐｒｏｌｏｇの並列性

Prologのプログラムには、並列性が存在している[3]。この並列性には次の三つが考えられる。

1）　OR並列

一つのゴールが複数の節と同一化可能となる時、この全ての選択可能な節を並列に探索

することをOR並列という。OR並列は、選択肢がOR関係であるから、他の選択肢を意識せずに並列に探索できる。

２）　AND並列

AND関係で結ばれた節の本体のゴール列を、並列に評価（実行）すること。この場合、一般にゴール間には共有変数が存在する。従ってこの部分はパイプライン的に評価を行なわなければならない。例えば、

a(X),b(X,Y),…．

のようなゴール列に対しては次の手順で実行する。

(1)　a(X)を実行し、Xの値を求める。

(2)　(1)で求めたXを用いてbを実行し、同時にaの別解を求める。

(3)　aの解が得られたならば、それを用いてbを実行する。

しかし、この様な方法を採った場合、資源の無駄使い（成功するゴールには辿りつかない無駄なゴールも評価してしまう）になってしまう。従って、本システムでは、OR並列によって得られる選択可能な節を全て得ておき、バックトラックが起きた時、別解により実行を続ける。

３）　同一化並列

複数の引き数を持つ頭部を同一化する際、各引き数の同一化を並列に行なうことをいう。

３．セルラシステムによるＰｒｏｌｏｇの処理方式

３．１　セルラシステムについて

セルラシステムとは、基本的な処理機能を持ったセルを多数連結して、全体として一つの機能を果たすシステムをいう。セルの連結の仕方は4隣接結合、8隣接結合などが一般的であるが、ここでは図3.1.1のようにして結合の柔軟性を高める。

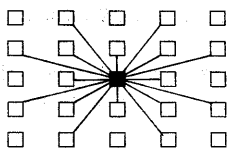


図3.1.1　セルの結合

この場合、一つのセルに着目すると近傍の16個のセルと結合していることになる。

Prologをセルラシステムで実行するために、三つのブロックからなるシステムを考える。

１）　記憶セルブロック

２）　実行セルブロック

３）　実行コントロールブロック

記憶セルブロックは、Prologプログラムのの事実節と規則節の頭部の格納と、同一化を行なう。実行セルブロックは、Prologの実行のための動的なプログラムグラフのマッピングと、同一化のための引数情報を格納する。実行コントロールブロックは、記憶セルブロックと実行セルブロック間のデータの受け渡しを行ないPrologの実行の制御を行なう。

また、実行コントロールブロックはホスト計算機に接続しており、ホスト計算機はプログラムのコンパイル、コンパイルオブジェクトの引き渡し、入出力操作を行なう。以下にその詳細を述べる。

３．２　記憶セルブロック

記憶セルブロックは、Prologの節の格納と、同一化を行なう機能メモリの役割を果たす。全体的なブロック図を図3.2.1に示す。

レジスタは外部との入出力に用いる。セル・インストラクション・ジェネレータはセルが実行すべき命令を保持している。リード／ライト・スイッチはセル・ユニットとの入出力用のバッファである。セル・ユニットは同一化操作を行う機能メモリである。セル・コンディション・ビットマップはセル・ユニット中の各セルの状態を表している。

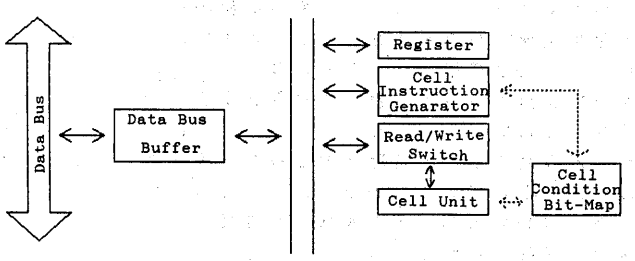
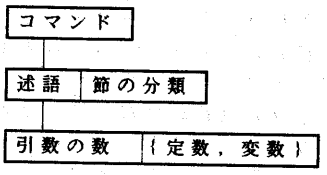


図3.2.1　記憶セルブロック

記憶セルブロックは、実行コントロールブロックから与えられたコマンドを解釈して実行する。コマンドは以下の通りである。

1） データ格納
2） 同一化
3） 初期化

このコマンドは、実行コントロールブロックから送られてくる。そのコマンドのフォーマットは、以下の通りである。



このコマンドはパケットの形で与えられる。このコマンドからセルに対しての命令がセル・インストラクション・ジェネレータで生成され、各処理が行なわれる。次に、図3.2.2にセルのブロック図を示す。

記憶セルブロック内のセルは、全て同一の命令を実行する。その命令は、コントロールバスを通して各セルに与えられ、その命令を保持するのがインストラクション・レジスタである。

命令は、通常励起しているセルに対してのみ有効であり、この制御を行なうのが、状態レジスタである。このレジスタには、そのセルが励起／非励起、使用中／未使用であるかの情報が保持されている。

各セルには番号があり、その値を保持しているのがセルナンバーレジスタである。これは、特定のセルを励起する場合に必要な情報である。セレクタはデータ・バスからのデータをデータ・レジスタ、または、コンパレータに送ったり、データ・レジスタの内容をデータ・バスに出力する。データ・レジスタは、同一化のためのデータが保持されている。このデータに関しては後で述べる。コンパレータでは同一化を行ない、その結果が状態レジスタに反映される。また、連結状態レジスタは、隣接するセルとの関係の情報を持っており、命令の内容によっては、接続しているセルを励起／非励起する。

以下に、セルに与える命令を示す。

1） EXITE ALL …… この命令は例外で、自分自身を励起する命令である。
2） EXITE N …… この命令も例外で、ナンバーNのセルを励起する。
3） WRITE …… 励起しているセルにデータを書き込む。
4） READ …… 励起しているセルからデータを読みだす。
5） FIX …… 励起状態を固定する。
6） NEXT …… 連結しているセルを励起状態にし、自分は非励起状態にする。

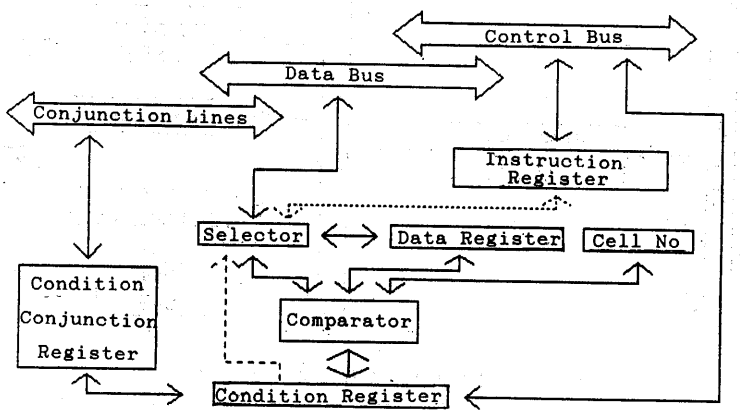


図3.2.2 セルのブロック図

7） COMPARE …… 入力されたデータと比べて等しい時、連結しているセルを励起し、自分自身を非励起状態にする。

8） READ STATUS N …… セルナンバーNのセルの状態を読む。

Prologの実行においてはプログラムの格納の順番が重要なので、TRUEというデータにはラインナンバーをタグとして付加する（規則節の本体は格納しない）。データは図3.2.3のようにマッピングされて格納される。

```
head.
head(arg1,arg2,arg3).
head(arg1,[lst1,lst2,…,lstn],arg2).
head1(X,Y):-head2(X,Z),head3(Z,Y).
```

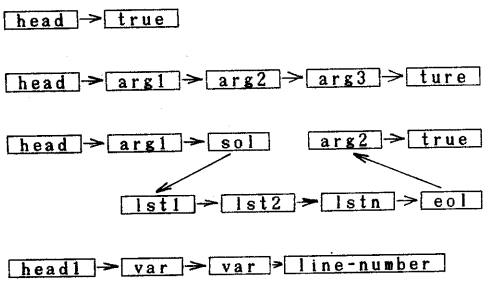


図3.2.3 セルへのデータのマッピング

3．3 実行セルブロック

実行セルブロックは、Prologの動的なプログラムグラフのマッピングのために用いる。

実行セルブロックの各セルは、処理要素でもあるがメモリでもある。図3.3.1に実行セルブロックのブロック図を示す。

実行セルブロックには、実行モードと入出力モードの二つがある。実行モードは、マッピングされた命令、データをもとにプログラムの実行を行うモードであり、入出力モードは、データの入出力（表示や、キー入力など）、プログラム実行に必要な同一化の情報の授受を行なうモードである。このモードを切り替えるのがモード・コントローラである。

リストデータを実際に処理するのが、リスト・プッロセサであり、リストは、セル・ユニット中ではポインタとして扱われる。実際のリストを記憶するためと、規則節の本体を記憶するために（規則節の本体を記憶セルブロックから読みだすと、オーバーヘッドが生じるため）メモリがある。

各セルへの入出力データや同一化のためのデータは、データマッパーを通して行なわれる。

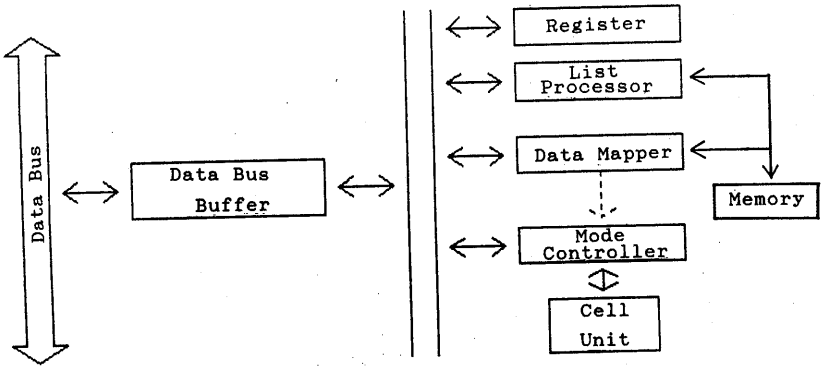


図3.3.1 実行セルブロック

次にセルのブロック図を示す。

セルは入力にデータが揃っていない時は、スリープ状態であり、入力にデータが揃うまで待つ。セルは入力バッファにデータが揃った時点で演算を実行し、結果を出すが、その結果が確定するのは、実行条件に入力があった時である。そして、データを受け渡すべき隣接するセルの入力バッファが空の時、コンジャンクション・ラインを通じてデータを送り、スリープ状態になる。スリープ状態からアクティヴ状態に変わる場合はデータが揃った時か、実行条件に入力があった時である。実行条件に入力があったことを示すのが、ステイタス・レジスタである。演算はインストラクション・レジスタによって示される演算を行なう。

セルには、個々に命令が割り当てられる。セルの命令には、大きく分けると、データの流れを制御する制御命令とPrologに組込まれている組込み述語を実行する組込み述語命令とがある。

各命令を示す。

1） 制御命令

(1) HOLD …… 定数データを保持するのに用いる。

(2) INPUT …… これは、入力のための命令で、必要なデータを得る。

(3) OUTPUT…… これは、出力の時使う。

(4) PASS …… 入力データの通り出力する。

(5) CALL …… 実行コントロールブロックに実行すべき節の本体を要求し、その節の本体をを実行させる。そして、自分は、実行させた節の終了を待つ。

2） 組込み述語命令

(1) 比較 …… ＝，≠，≧，＞

これらは、演算の結果が正しい時TRUE、誤っている時FALSEの値を持つ。

(2) 四則演算 …… ＋，－，×，÷

実行時の変数は多くても一つである。

(3) NOT …… 成功と失敗を反転する。

(4) MEMBER(A,B) …… 要素Ａがリスト Ｂに含まれている時成功する

(5) SELECT(A,B,C) …… 要素Ａをリスト Ｂからとる。ＣはＢからＡを除いた残り。

3） その他

(1) FALSE …… （必ず）失敗する。

(2) ! …… カットオペレータ（バックトラックの時、これを含む節の頭部を呼びだした項が失敗する）。

(3) PRINT …… 出力装置への出力。

(4) INPUT …… 入力装置からの入力（引数は変数である）。

組込み述語命令は最低限のものしか用意していない。

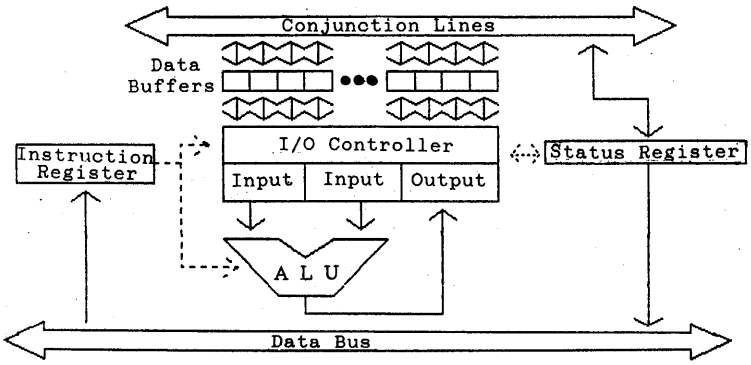


図3.3.2 セルの構造

また、命令のフォーマットは以下の通りであり、これがインストラクション・レジスタに格納される。

OPCODE	I/O FLAG	1st ARG	2nd ARG

2ビット×16セル　　データの来るセル
00－OUTPUT　　の方向を示す
01－INPUT
01－NOT CONECT
10－CONTROL DATA実行条件）

セルの内部では、比較演算や四則演算のとき、

〔OUTPUT〕＝〔1st ARG〕〔命令〕〔2nd ARG〕

というように演算している。

3．4　実行コントロールブロック

実行コントロールブロックは、実行セルブロックと記憶セルブロックとの間にある。実行セルブロックからの入出力要求を処理したり、同一化の要求により記憶セルブロックに同一化のためのコマンドを送る。ブロック図を図3.4.1に示す。

実行コントロールブロックへの出力要求は次の三通りである。

1）　出力装置へのデータ
（→ホスト計算機）
2）　同一化のためのデータ
（→記憶セルブロック）
3）　実行の終了（成功／失敗）
（→ホスト計算機）

また、入力は

1）　入力装置からのデータ
（ホスト計算機→）
2）　同一化のためのデータ
（記憶セルブロック→）
3）　実行開始のためのデータ
（ホスト計算機→）

4．　性能評価

Prologの実行は動的な部分が多いので、実際にはプログラムの実行速度で評価をしなければならないが、ソフトウェア・シュミレータが、まだ完成していないので、性能の目安として、事実節の同一化に要する時間を逐次処理と比較を行う。

仮定として、逐次処理は呼び出す側と呼ばれる側がメモリ上にあり、結果がメモリ上に格納されるものとする。一方、本システムにおいては、呼び出す側は、実行セルブロック上にマッピングされており、同一化に必要なデータのみが記憶セルブロックに送られ、最終的な結果だけが実行コントロール・ブロックに返される。

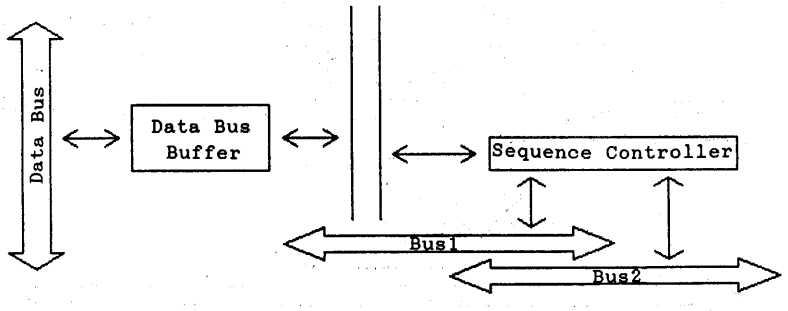


図3.4.1　実行コントロールブロック

記号は次のように定義する。

T_M : メモリ・アクセス時間
T_{UNIF} : 同一化操作時間
N : 引数の数
M : 頭部が等しい節の数

まず逐次処理の場合であるが、同一化が全て成功する場合の処理時間は、

$$T_s = (2T_M + T_{UNIF})N + T_M$$

となる。逆に、同一化が全て失敗する場合（最悪の場合）の処理時間は、

$$T_s = (2T_M + T_{UNIF})M + T_M$$

である。実際には、処理開始のためのスタート・アップ時間があるが、ここでは無視している。また、一つの引数に関して確率Kで同一化が失敗する場合の処理時間の期待値は、

$$E_s = \{(1-K)^N(2T_M + T_{UNIF})N + \sum_{n=1}^{N}(1-K)^{n-1}\cdot K(2T_M + T_{UNIF})n\}M(M+1)/2 + T_M$$

一方、本システムの場合は、成功／失敗に関わらず並列に探索可能であるので、

$$T_c = (T_M + T_{UNIF})N + T_M$$

となる。

逐次処理の場合は、頭部の等しい節が多数あると、同一化が失敗する確率が高ければ高いほど、処理時間が長くなる。一方、本システムの場合は、頭部の等しい節の数や同一化が失敗する確率の高さに関わらず、一定の時間で結果が得られ、逐次処理よりもその時間が短い。

5. むすび

以上、セルラシステムでPrologの並列姓を考慮して実現する方法を述べた。

システムの性能評価は、Prologの実行が動的に行われるので、実際には、個々のプログラムに関してなされなければならない。従って、従来のものと単純に比較できない。しかし、並列姓を考慮しているので、高速な処理が可能であると思われる。しかし、詳細な性能解析はシミュレータを作成し、動的な特性を調べなければ得られない。

今後、詳細な性能の検討を行う予定である。

参考文献

（１）中島：" Prolog"
産業図書（昭和58-8）
（２）安部：" Prologプログラミング入門"
共立出版（1985-3）
（３）" VLSIコンピュータII"
岩波書店（1985-2）