東京大学工学部総合試験所
年報　　第42巻　(1983)

原稿受付　58年6月20日

推論マシンシステムに関する研究**

元岡　達、田中　英彦、後藤　厚宏、相田　仁*

RESEARCH ON THE INFERENCE MACHINE SYSTEMS

Tohru MOTO-OKA, Hidehiko TANAKA, Atsuhiro GOTO and Hitoshi AIDA

Proof diagram representation of logic programs shows the concept of goalframes, which are intermediate results of an initial goal and fully include their own unification environments. Computational model based on inter-goalframe parallelism of logic programs are considered and basic architecture of inference machines is proposed. The main components of this machine are numerous unify processors with definition memories and numerous memory modules with activity controllers. Formers can execute logic programs in a highly parallel manner and latters, can manage and schedule parallel activities efficiently, using search tree information. These modules are connected through switching networks.

1．はじめに

現在、計算機システムの応用分野は、与えられた問題の解決手段があらかじめ明確なものが主であり、計算機システムの開発目標は、与えられた解決手段に従って最小時間で解くことである。

一方、比較的近い将来の計算機システムは知識情報処理の分野に適応する必要があることが指摘されている。知識情報処理では、与えられた問題と知識から問題解決の手段自体を生成しながら問題解決を行なう推論の機能が中心となる。これによって、問題解決の手順が不明確な分野まで計算機システムの応用分野を拡大することが可能となる。

Prolog に代表される論理型プログラム言語を実行するシステムは、述語論理に基づいているため言語上の性質のよく、また宣言的、非決定的、非数値処理向きといった特徴を持ち、知識システムの初期モデルとして有望である。しかし、上記の特徴は、従来のマシンアーキテクチャになじまないため、処理速度／機能の両面において高性能なマシンアーキテクチャが要求されている。

本報告では、まず論理型プログラムの実行モデルについて基本的な概念を整理し、その並列処理性について議論した後、論理型プログラムの高度な並列処理と直接実行を目指した高並列推論エンジンＰＩＥ（Parallel Inference Engine）のアーキテクチャおよびハードウェア構造の概要について述べる。

2．論理型プログラムの並列処理

2．1　証明図によるプログラムの実行表現

Prolog プログラムは、基本的にHorn 節の集合であり、定義節と初期ゴール節から成る。プログラムの実行は、初期ゴール節内のゴールリテラルに定義節の頭部リテラルを適用する（単一化操作）ことによって、次々と新たなゴール節を導出することによって進められ、空節に至った時に成功となる。このようなProlog プログラムの実行過程は、証明図（Proof Diagram）を用いて表現することにより、内在する並列性等を理解することが容易となる。

（1）節のテンプレート表現

証明図では、例1における各定義節Ｄ1－Ｄ3を図1（a）のような定義節テンプレートによって表わし、初期ゴールＧを図1（b）のようなゴールテンプレートによって表わす。ここで、節中の各リテラルは半円によって示され、頭部リテ

＊　東京大学工学部電気工学科
＊＊（財）新世代コンピュータ技術開発機構の委託研究「推論処理用基本アーキテクチャに関する調査」の報告

ラルは下半円に、ゴールリテラルは上半円に対応する。両半円は、述語記号とその引数に対応した引数ポート（図中の小円）を持つ。変数はリンクで示され、関数式とその引数、定数および引数ポート間を結ぶ。

（２）ゴールフレーム

Prolog プログラムの実行のある時点における計算の中間結果は中間ゴール節およびそれまでに行なわれた単一化によって生じた変数に対する置換により表される。この中間結果をゴールフレーム（ goal frame ）と呼ぶことにする。

```
                < EX. 1 >
[D1] append(nil, *x, *x) ←.
[D2] append((*u・*x), *y, (*u・*z))
                ←append(*x, *y, *z).
[D3] sublist(*x, *y)
                ←append(*u, *x, *v),
                  append(*v, *w, *y).
[G ] ←sublist((a・*x), (*y・nil)).
```

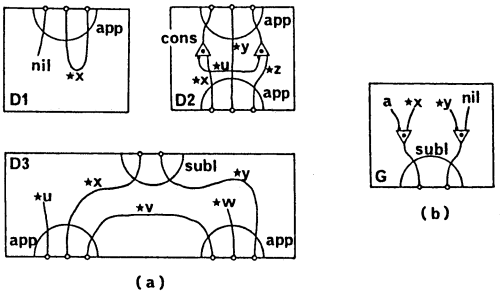


Fig.1 Templates Representation of Logic Programs
(a) Definition Templates
(b) A Goal Template

証明図において、ゴールフレームはテンプレートの組み合わせとして図式的に表現できる。例１のプログラムの実行におけるにゴールフレームの例を図２（ａ）に示す。ここで、成功した単一化は円によって示され、まだ単一化されていない活性ゴールリテラルは上半円として表される。

２．２ 論理プログラムの並列処理方法

論理プログラムの実行を証明図によるゴールフレームと定義節テンプレートの組み合わせとしてとらえた場合、その実行過程において、次の４種類の並列処理を考えることができる。（図３）

（１）ゴールフレーム間並列処理

複数のゴールフレームが処理系内に存在するときに、それらを並列に処理する。

４つの並列処理方法のうちで最も処理レベルの高いものであり、並列処理間の独立性は高い。あるゴールフレームの状態によって他のゴールフレームを消滅／一時不活性化／活性化する等の影響を及し合い、全体として様々な機能を実現することが可能である。

（２）ゴールリテラル間並列処理

あるゴールフレームにおいて活性ゴールリテラルが複数ある場合に、それらに対する処理を並列に行なう。

並列に処理するリテラル間で共有される変数がある場合には、それらのリテラルを処理した結果が共有変数に関して矛盾をおこしていないかどうかを確認しなければならない。

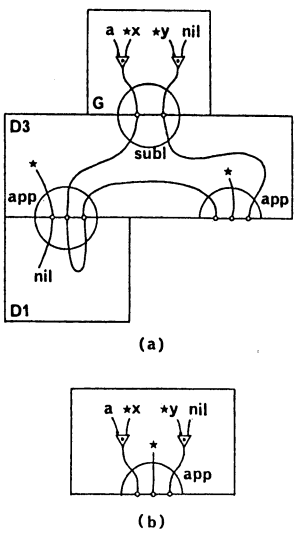


Fig.2 An Example of Goal Frames and Reduced Form

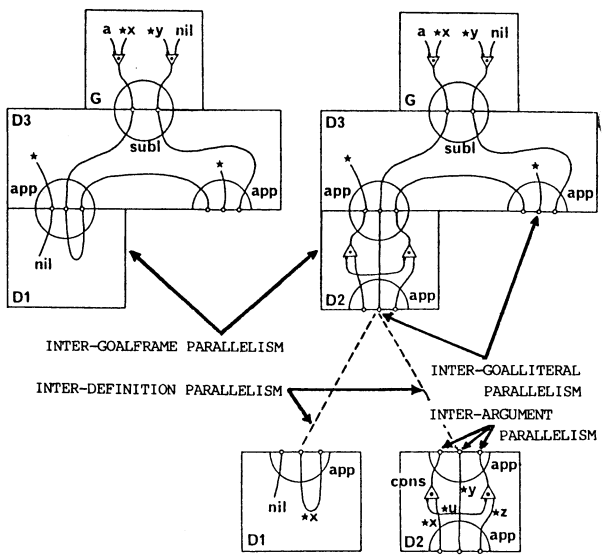


Fig.3 Parallelism of Logic Programs

本並列処理では、この無矛盾性検査（consistency check）をよほど要領よく行なわないと、速度の向上が期待できないばかりか、有限時間内に解が得られない可能性もある。

（３）定義節テンプレート間並列処理

あるゴールフレーム内の特定の活性ゴールリテラルと同じ述語名を持つ定義節テンプレートが複数あるときに、それらとの照合操作を並列に行なう。

この並列処理の下では、並列処理したそれぞれから重複した解が得られることがある。このような解の重複を検出し、とり除くためには、ほぼ、ゴールリテラル間並列処理における無矛盾性検査と同程度の手間を要する。

しかし、解の重複の検出は省略してもよいことが多く、また、現在通常のProlog 処理系で行なわれているように、プログラマの責任で解の重複を防止させることも考えられる。そのような場合には各定義節テンプレートとの照合操作を完全に独立して行なうことができる。

（４）リテラル内引数間並列処理

ある活性リテラルと定義節テンプレートとの照合操作において、複数の引数ポートがある場合に、それらのポートの照合操作を並列に行なう。

この並列処理では、並列処理を行なう単位が他の３種の並列処理の場合に比べてかなり小さい。このためオーバーヘッドの問題から、並列処理を複数プロセッサに分配して行なうような処理形態には適さないが、ハードウェアとしてテンプレート照合器の中に実装できれば、数倍程度の速度向上が期待できる。

３．ＰＩＥの基本アーキテクチャ

３．１　ＰＩＥ第一次モデルの並列処理方式

ＰＩＥ第一次モデルにおける、基本処理要素はユニファイプロセッサ（ＵＰ：Unify Processor）である。ＵＰは、ゴールフレームを入力すると、その中のひとつのゴールリテラルについて、対応するすべての定義節テンプレートとの単一化を行ない、新たなゴールフレームを（複数）導出する（図４）。ここで、各ＵＰには定義節テンプレートのメモリが付随しており、その各々がすべての種類の定義節テンプレートを持つものとする。

ＵＰによって導出された複数のゴールフレームはゴールプール（goal pool）に蓄えられ、ＵＰに対する新たな入力となる。ＰＩＥでは、多数台のＵＰによって図５に示すようなゴールフレーム間並列処理が実現される。

ＰＩＥ第一次モデルでは、このようなゴールフレーム間並列処理の実現を第一目標とし、定義節テンプレート間並列処理および、リテラル内引数間並列処理をＵＰの内部機能として考慮する。ゴールリテラル間並列処理については、第二次モデル以降で検討する。

３．２　ゴールフレームの縮退

ＰＩＥでは、ＵＰにおいて、ゴールリテラルに対応するすべての定義節テンプレートを適用するため、後戻り（back-track）がない。これにより、通常の逐次システムにおいて後戻りを考慮してスタック上に残しておく情報の大部分は不要となる。そこでＰＩＥでは、記憶コストおよび転送コストを抑えるために、縮退操作は、導出された新ゴールフレームから不要な情報を取り除き、ゴールフレームの大きさを必要最小限にする。これは、図２（ａ）の証明図を図２（ｂ）のように圧縮することに相当する。

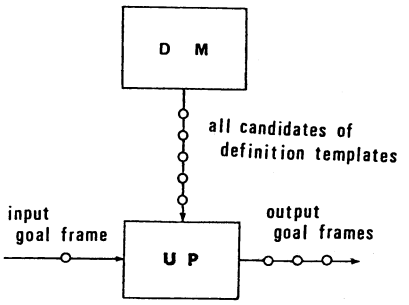


Fig.4　Basic Concept for a Unify Processor

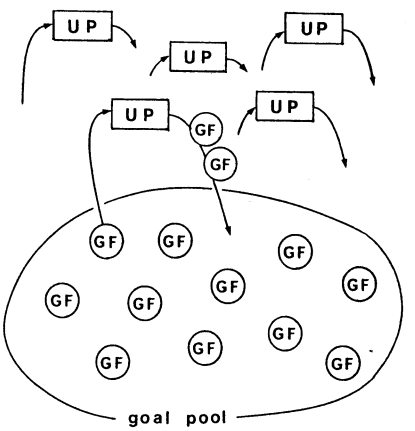


Fig.5　Inter-Goalframe Parallel Processing Feature

3．3　ＰＩＥ第一次モデルのアクティビティ制御方式［３］

（１）ゴールフレームの管理方式

前述のＰＩＥの処理方式におけるプログラムの計算全体は、ＯＲノードからなる探索木として表現できる。そこで、並列処理の単位であるゴールフレームを、計算の途中状態として探索木上に位置付けることによって、各々の関係を明確にすることができる。一方、現実的な問題解決プログラムの記述ではHorn 節を越えた言語上の制御構造の導入が必要とされ、この場合はゴールフレーム間にＯＲ以外の依存関係が生じる。そこでＰＩＥでは、ゴールフレーム間の関係を維持する探索木にＯＲ以外の新たなノード属性を導入することによって、アクティビティコントローラ（ＡＣ）がシステム内の多数のゴールフレームを管理する。さらに以下のようなノード制御コマンドを設定し、これをノード間で送受することによって、基本的なアクティビティ制御を実現する。

・探索木のノード情報とコマンド

実行途中の探索木は、根ノード・中継ノード・末端ノード（成功／失敗ノード）・ゴールノード（ゴールフレームに対応するノード）によって構成される。

ＰＩＥでは各ノード操作を並列に実行できるように図６のようなノード情報によって探索木情報を保持する。

・ゴールノードにおける探索木の伸長　（図７）

ゴールフレームに対応したゴールノードはＵＰによって実行が進められ、新たなゴールフレームが生成される。この際にＵＰとＡＣ間でメッセージを交換し、探索木を更新する。

・末端ノードにおける探索木の刈込み

末端ノード（特に失敗ノード）の多くは探索木情報として不要となり、資源の節約のために刈込まれる。

・中継ノードにおける不要ノードの削除

子ノードがただひとつになった中継ノードも多くの場合不要である。図８の手順により非同期的に不要中継ノードを削除することが可能である。

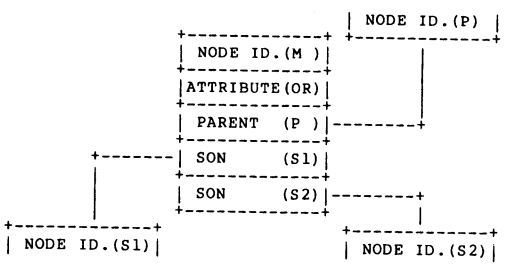


Fig.6　Node Information for Search Tree

（２）ゴールフレーム間並列処理の効率化

ＰＩＥの処理方式における並列度は非常に大きく、そのままでは“数の爆発”の問題を引き起こす可能性がある。また、処理途中におけるすべてのゴールフレームが成功に到るわけではなく、多くのものが失敗の確認に終る。このため、種々の解の探索ストラテジの導入、および資源の枯渇を防止する対策が必要である。

そこでＰＩＥでは次のような処理の効率化手法を考えている。

ａ．多数のゴールフレームの中からＵＰにおいて実行するものを選択する。

ｂ．選択されたゴールフレームをシステム内の負荷が均一になるようにそれぞれのＵＰへ割り付ける。

ｃ．割り付けられたゴールフレームの中から単一化を実行するリテラルを選択する。

ｄ．現実的な問題解決プログラムの記述に必要な言語上の拡張機能を実行する。

ｅ．単一化の失敗等によって不要になったデータを消去し資源を回収する。

・優先度スコアによる実行制御

上述の制御には多くの因子があるばかりでなく、その制御ストラテジにも、プログラマによって直接に指定される場合と、システムが自動的に判断する場合とがある。これらを統一して扱うために、ＰＩＥでは、それぞれのリテラルやゴールフレームに対して優先度スコアを設ける。

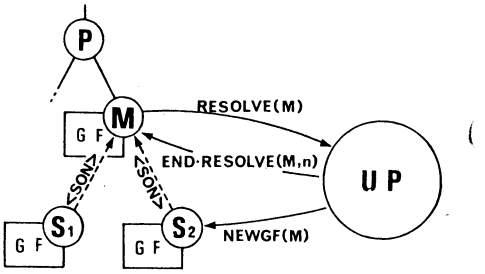


Fig.7　Expansion Mechanism for Leaf Nodes

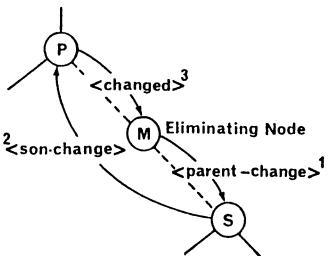


Fig.8　Eliminating Mechanism for Unnecessary Relay Nodes

変数の数または定数引数や構造体との割合の因子からリテラルの選択を自動的に最適化する手法について、これまでに、aに関しては根ノードからの深さ、cに関しては手続き呼び出しの決定性およびリテラルの引数中の変数の数や割合から優先度スコアを設定する手法についてシミュレーション評価を行なった結果、多くのプログラムを効率的に実行することができた［７］。しかし、リテラルやゴールフレームの選択等をすべて単一の優先度スコアにより制御可能であるかどうか、更に検討が必要である。

４．ＰＩＥ第一次モデルのハードウェア構造

４．１　ハードウェア全体像

高並列推論エンジンＰＩＥ第一次モデルのハードウェア構成の全体像を図９に示す。［Got83］

（１）Unify Processor（ＵＰ）
および　Definition Memory（ＤＭ）

ＵＰは単一化をハードウェアによって直接サポートする要素プロセッサであり、ＤＭは定義節を保持するメモリである。ＵＰは、ひとつのゴールフレームを入力し、その中のひとつリテラルについて、対応する定義節のすべての選択肢をＤＭから取り出す。この時、失敗が明白なものについてはＤＭ側でチェックして選択肢を絞り込む。ＵＰは、単一化に成功し新たに生成された複数のゴールフレームを結果として出力する。

（２）Memory　Module（ＭＭ）

ゴールフレームの実体はＭＭ内に置かれ、全体でゴールプールを構成する。各ＭＭには、ハードウエア・ガーベジ・コレクタを設ける。

（３）Activity Controller（ＡＣ）

ＡＣは与えられた問題における探索木の情報を分割して保持することによってＭＭ内の各ゴールフレームの関係を把握する。また、ＡＣ間の相互通信とその探索木情報によって、前章で述べたようなアクティビティ制御のうち、ＵＰに対するゴールフレームの割り付け／効率化のための制御／言語上の拡張機能／資源の管理等を行なう。

（４）Acivity　Manager

Ａctivity Managerは、全ＡＣ及びＭＭを監視し、システム内の負荷分散を均一にする。

（５）System Manager

System Managerは入出力、データベースマシン等との外部インタフェースを司る。

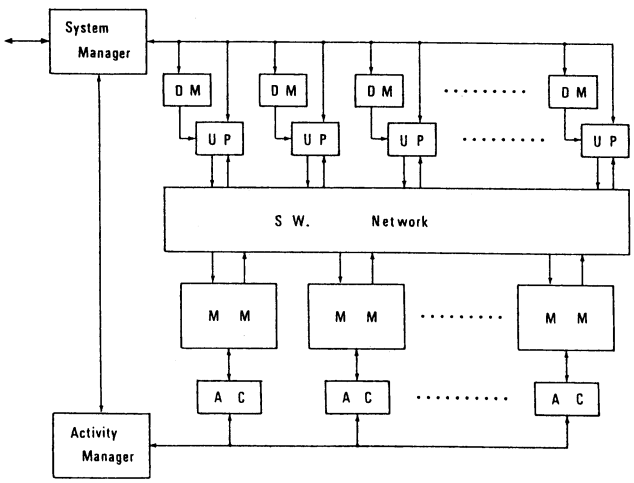
（６）Network

現在、結合網は任意のＵＰ－ＭＭ間でゴールフレームが転送できるものを考えている。

４．２　ゴールフレームと定義節の表現

ゴールフレームや定義節は、タグ付の固定長セル（４０ビット程度）の連続形とする。図10にその例を示す。

データ型は次の３種類に大別される。



UP : Unify Processor
DM : Definition Memory
MM : Memory Module
AC : Activity Controller

Fig.9　System Organization of Proposed Inference Machine

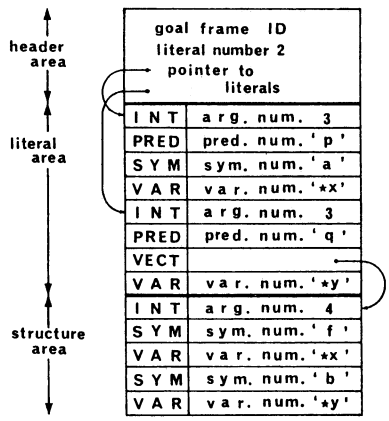


Fig.10　Internal Representation of Goalframes

① アトム

シンボル、整数値などのアトムは、セル１個で表される。シンボルは、シンボル番号で示され、シンボルの印字名は、System Managerが管理する。

② 構造体

ベクタ等１セルで表現できないものは、実体を構造部内におき、そこへのポインタにより表わす。

③ 変数

変数は、変数番号で区別する。縮退により、すべての変数は、未定義であることが保証されるので、結合網上やMM（Memory Module）中でのゴールフレームには、変数の実体を収納する領域は存在しない。

４．３ UPの機能および内部構造 （図11）

（１）単一化操作

UP内のローカルメモリは、ゴールフレーム側のセルと、定義節側のセルに同時にアクセスできるように、２つに分れている。単一化は、一般に１つのゴールリテラルと複数の定義節の頭部との間で行なわれる。定義節はDMから定義節側ローカルメモリに取り込まれ、ゴールフレームも入力バッファからゴール側のローカルメモリにコピーされる。

単一化は、ハードウェア化されたUnifierが実行する。論理変数の束縛状況は、ローカルメモリ中の変数メモリに記録される。構造体を単一化を行なう場合には、スタックを用いて再帰的に処理する。

（２）縮退操作

単一化が成功した場合には、Reducerが縮退操作を行ないつつ、新たなゴールフレームを出力バッファ上に再構成する。縮退のアルゴリズムは、圧縮型ガーベジ・コレクション方式を応用したものであり、ポインタをたどりながら、必要なセルのコピーを行なう。

（３）リテラルの選択、及び、優先度スコアの計算

新ゴールフレームから、次回に処理すべきリテラルを選択する。さらに、ゴールフレーム間の優先度を定めるための優先度スコアをゴールフレームの状態から計算する。

（４）新ゴールフレームの送出

新しく作成されたゴールフレームは、結合網を介してMMに分配される。入力ゴールフレームが存在したMMに付随するACには、新たに生成された子ゴールフレームの数が渡される。

５．おわりに

本報告では、まず、論理型プログラムに関して、実行モデルと並列性について述べ、次に、高並列推論エンジンＰＩＥの基本アーキテクチャについて並列処理方式と制御方式の両面からその基本機能を明らかにし、さらに、ＰＩＥの第一次モデルのハードウェア構造を示した。

現在までに、UPおよびACの基本的な設計が終了し、ソフトウェアシミュレータによりその正当性の検証を急ぐとともに、各種の特性データの収集を行なっている。

今後は、各部の詳細設計を進め、部分試作を行なってゆく予定である。

参考文献

１．後藤 他，“推論向き高並列計算機システムの基本アーキテクチャ”，信学技法ＥＣ 82-43（1982）

２．後藤 他，“推論向き高並列計算機システムのアーキテクチャ”，第２６回情処全大，４Ｎ－４（1983）

３．丸山 他，“推論向き高並列計算機システムのアクティビティ制御機構”，ｉｂｉｄ，４Ｎ－５

４．湯原 他，“推論向き高並列計算機システムのユニフィケーション制御機構”，ｉｂｉｄ，４Ｎ－６

５．相田 他，“推論向き高並列計算機システムの基本言語機能”，ｉｂｉｄ，４Ｎ－７

６．後藤 他，“高並列推論エンジンＰＩＥについて”，The Logic Programming Conference,10.2（1983）

７．後藤 他，“高並列推論エンジンＰＩＥにおける並列処理の効率化手法について”，信学技法ＥＣ83-9（1983）

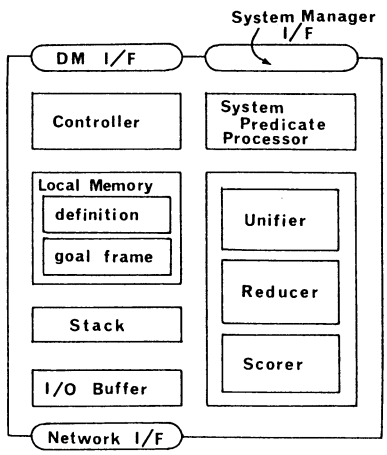


Fig.11 Functional Block Diagram of Unify Processor