論理型プログラミング言語向き並列計算機 KPR の並列処理方式†

柴山 潔†† 鹿毛裕史†† 川倉康嗣††
山本雅亮††* 平田博章††**
加納 健†††** 萩原 宏††

本論文では，論理型言語向き並列計算機 KPR の並列処理方式について，対象言語の機能，実行モデル，システム構成方式の各処理機能レベルごとに述べる．「並列リダクション・モデル (PR モデル)」と呼ぶ KPR の実行モデルでは，論理プログラムの実行（証明・推論）過程を AND/OR 推論木における事実の探索過程としてとらえ，この木の各ノードに1つのプロセスを割り当てて得られるプロセス・グラフのリダクション（展開・縮退）操作を並列に行う．プロセスには，プロセス処理における AND 並列性をパイプライン（ストリーム並列処理）方式で処理する Stream プロセス，OR 並列性を実現する Or プロセスなどがある．これらのプロセスの処理は，各々専用のプロセッサで行われる．Stream プロセス処理専用プロセッサ ARP と OR プロセス処理用プロセッサ ORP は対として密結合され，2分木ネットワークの葉ノードに要素プロセッサとして割り付けられる．ARP は，ストリーム並列処理の実行を行う AND リダクション・ユニット (ARU) を中心として構成される．ORP は，4個のユニフィケーション・ユニット (UU) によってユニフィケーションを並列に処理する OR リダクション・ユニット (ORU) を中心として構成される．また，ARP や ORP は，プロセスの実行やプロセス（プロセッサ）間通信管理を専用処理するプロセス制御ユニット (PCU) を装備している．さらに，ネットワークの中間ノードは NNU と呼ばれ，バス・スイッチや状態フラグの伝搬機構などが装備される．

1. はじめに

論理型言語自体に内在する並列性に着目し，これに適合した並列計算機アーキテクチャを構成することによって，記号処理や人工知能処理の飛躍的な高速化を実現しようと，推論マシンの開発が各所で行われている[1]．我々は，数年前より，論理型プログラミング言語向きの並列計算機アーキテクチャの開発研究を行っており，並列論理型言語の高速実行を目指した並列計算機 KPR を開発している[2),3)]．

並列論理型プログラミング言語で記述された論理プログラムはホーン節の集合とみなせる．同一の述語を持つ節集合（これらを「定義体」と呼ぶ）中の節間には論理的な OR 関係があり，本体ゴール列間には論理的な AND 関係がある．これらのセマンティクスをそれぞれ「OR 並列性」，「AND 並列性」と呼んでいる．KPR の対象言語である "KPR-L" は，特に OR 並列性を積極的に活用する並列論理型プログラミング言語である．すなわち，KPR は，OR 並列性を十分に含んだ応用，例えば探索問題，特に全解探索を要する問題，に適合するアーキテクチャを目標として設計されている．

KPR における論理プログラムの実行は，論理プログラムの実行過程を AND/OR 木における事実の探索過程としてとらえる「並列リダクション・モデル (PR モデル)」によって制御される．PR モデルは，プロセス粒度とハードウェア量とのトレードオフにおいて，ハードウェア化できる機能を最大限に引き出すことによって，実行効率の向上を図る方針で設計されている．また，KPR は，PR モデルで表現する論理的粒度ごとに機能の相異なる各種のプロセッサや処理ユニットを用いてシステムを構成する「ヘテロジニアス・マルチプロセッサ方式」という特徴がある．

本論文では，KPR の設計思想，アーキテクチャおよびシステム構成について述べる．さらに，対象言語の機能，実行モデル，システム構成方式の各処理機能レベルごとに，KPR の並列処理方式について，他の

† Parallel Processings of a Logic Programming Language-Oriented Parallel Machine KPR by KIYOSHI SHIBAYAMA, HIROSHI KAGE, YASUSHI KAWAKURA, MASAAKI YAMAMOTO, HIROAKI HIRATA, YASUSHI KANOH and HIROSHI HAGIWARA (Department of Information Science, Faculty of Engineering, Kyoto University).
†† 京都大学工学部情報工学教室
* 現在 通商産業省 Ministry of International Trade and Industry
** 現在 松下電器産業(株) Matsushita Electric Industrial Co., Ltd.
*** 現在 日本電気(株) Nippon Electric Co., Ltd.

並列推論マシンとの定性的な比較・考察を行うことによって，KPR の並列処理方式の特徴を明らかにする．

2. KPR の実行モデル

2.1 並列リダクション・モデル（PR モデル）

KPR における KPR-L プログラムの実行モデルを「並列リダクション・モデル（PR モデル）」と呼ぶ[2]．KPR-L プログラムの実行過程を表現する AND/OR 証明木の各ノードに，処理の単位機能としてプロセスを対応させる．ノードをプロセスに置き換えてできたプロセス・グラフの展開・縮退（プロセスの生成と消滅）操作を「リダクション」と呼ぶ．PR モデルにおけるプロセス・グラフのプロセスと AND/OR 木のノードとの対応は以下のようになる．

（1） Or プロセス（Oプロセス）……「OR ノードから，論理的 OR 関係にある複数の子ノードを展開する」過程を処理する．Oプロセスは，複数の子プロセスを同時に起動し，これらを並行して処理させる方式により，KPR-L の OR 並列性を実現している．

（2） Stream プロセス（Sプロセス）……「AND ノードから，論理的 AND 関係にある子ノードを展開する」過程を処理する．KPR-L の OR 並列性によって，1個のゴールに対しては複数個の解が返される．Sプロセスでは，それらの解を用いて各ゴールごとに，その評価の直前までに生成された1本のストリームを枝分かれさせる．これを「ストリーム並列処理方式」と呼ぶ．Sプロセスはストリーム並列処理によるプロセス・パイプライン方式で本体ゴール列を処理し，KPR-L の AND 並列性を擬似的に実現する．

（3） Database プロセス（Dプロセス）……「データベースとして用いられる節（事実）の集合に対するノード」を処理する．Dプロセスは，子プロセスを生成しないので，プロセス・グラフの縮退操作の発火点となる．

PR モデルに基づく KPR の論理プログラムの並列処理機能は，**図1**と以下に示すように，プロセス個々の処理とプロセス間通信処理とから成る．

1) プロセスは，Sプロセスと，OプロセスまたはDプロセスが原則として交互に起動される．

2) プロセス間通信は原則的に親子関係にあるプロセス間のみで行われ，通信メッセージは次の3種である．

（a） デマンド……親から子への要求．プロセス・グラフの展開（プロセスの生成）．

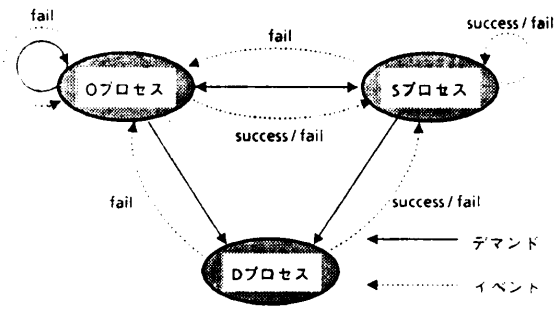

図1 プロセス間通信メッセージ
Fig. 1 Intercommunication messages between processes.

● プロセスの起動を要求する *invoke* メッセージ．

（b） イベント……子から親への応答．プロセス・グラフの縮退（プロセスの消滅）．

● 1個の解を返す *success* メッセージ．

● 解が（もうこれ以上）ないことを知らせる *fail* メッセージ．

さらに，この割り当て戦略を次のように拡張，最適化する．

（i） 冗長なプロセスの除去……単一の子しか持たず，デマンドの単なる通過点でしかないプロセスは生成しない．

（ii） イベントの飛び越し……解を返すプロセスとその解を実際に必要としているプロセスとの間の通信に多くのプロセスが介在する場合は，中間のプロセスを飛び越して，直接，祖先のプロセスにイベントを返す．

2.2 ストリーム並列処理方式

KPR においては，1個の節本体の処理が1個のSプロセスに対応する．しかし，並列実行下では，各 OR 分岐で同じ変数が，異なる値を取り得るので，各分岐ごとに異なる環境（多環境）を管理することが必要となる．そこで，次のように，Sプロセスをさらに細かいプロセスに分解する．

まず，**図2**に示すように，節本体のゴール列をユーザ定義述語の呼び出しを単位として分解する．これを「ゴール・ブロック」と呼ぶ．1個のゴール・ブロックには，0個以上の組み込み述語と高々1個のユーザ定義述語しか含まれない．また，このゴール・ブロックの処理とは，親のOプロセスからの *invoke* メッセージあるいは子の O/S/D プロセスからの *success* メッセージを受信してから，新しい子の O/D プロセスを起動するまでの処理である．この処理を行う実体を「サブプロセス」と呼ぶ．

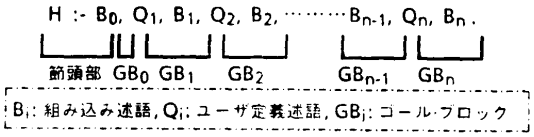

図 2 ゴール・ブロック
Fig. 2 Goal blocks.

KPR のストリーム並列処理では，複数の枝分かれしたストリームが並列に処理される．各ゴール・ブロックを評価する際のストリームの状態を表すものが「サブプロセス情報」であり，これが並列実行下での各ゴールの環境となる．

2.3 プロセス間通信メッセージ

プロセス間で授受されるメッセージは，主として，「プロセス管理情報」と「環境情報」とから構成される．

さらに環境情報は，(a) 節内の各変数の束縛状態を示す「変数束縛情報」; (b) PR モデルでは，述語呼び出しによって節外部における未束縛変数が節内に持ち込まれることがあり，このような変数に関する束縛情報である「外部変数情報」; (c) イベント飛び越しによる最適化実現のため，解として実際にどの外部変数を返してほしいのかを示す「解指定情報」; (d) 解指定情報に従って抽出された変数の束縛情報である「解情報」; (e) 述語呼び出しにおける引数を与える「引数情報」; とから構成されている．

図 3 に論理プログラムが実行される際に生成されるプロセスとその間のメッセージの例を示す．まず，ユーザからの質問（ゴール）がOプロセスへのデマンドの形に直されて，Oプロセスが起動される．次に，このOプロセスからの *invoke* メッセージによって，Sプロセスが生成される．さらに，そのSプロセスは子のOプロセスの *invoke* メッセージを作成し，送信するとイベント待ち状態となる．その後，起動した子のOプロセスから *success* メッセージを受信するごとに，解を返してきた子のOプロセスを起動する直前のサブプロセス情報をコピーして解を取り込み，次のデマンドやイベントを送信する．

PR モデルでは，述語呼び出し（ゴールの実行）の際に必要となる構造データは，すべてメッセージとして授受する「構造コピー方式」を採用している．また，OR 並列処理を行うので，逐次型言語（例えば，Prolog）で記述した論理プログラムの実行時に必要となるバックトラック処理機能（例えば，グローバル情報用スタック機構など）は不要である．

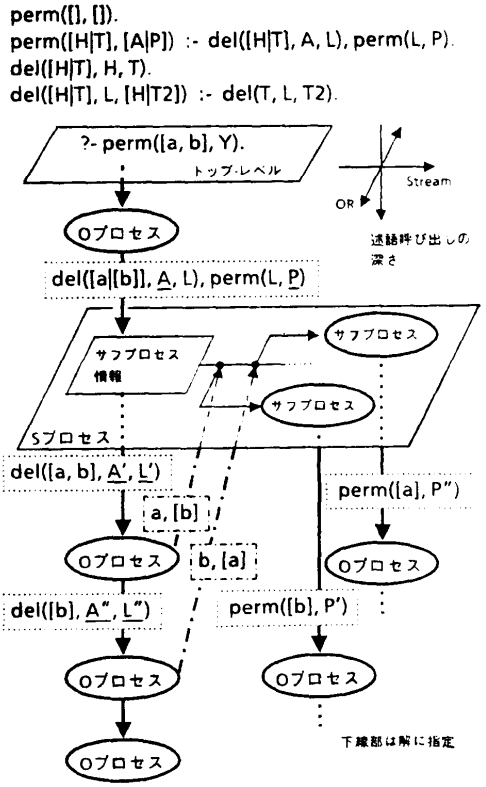

図 3 プロセスと通信メッセージの例
Fig. 3 Example of processes and messages.

3. システム構成方式

3.1 ハードウェア構成の概要

KPR システムでは，**図 4** に示すようなヘテロジニアス・マルチプロセッサ構成方式を採用している．KPR は，PR モデルの各種プロセスそれぞれの機能に対応した専用プロセッサから構成される機能分散処理システムである．KPR では，2分木状のネットワークによって要素プロセッサを結合しており，要素

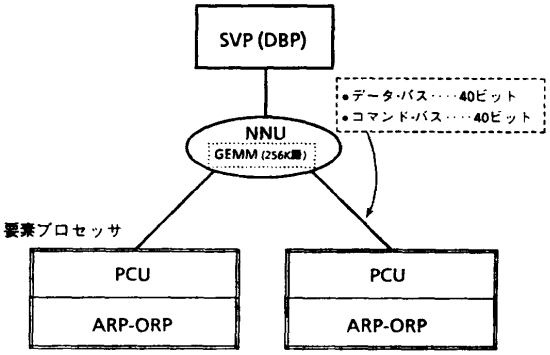

図 4 KPR のプロトタイプ・システム構成
Fig. 4 System organization of KPR prototype.

プロセッサ (PE) はこのネットワークの葉ノードの部分に結合されている.

要素プロセッサは,**図5**に示すように,Sプロセスを処理するANDリダクション・プロセッサ (ARP) と,Oプロセスを処理するORリダクション・プロセッサ (ORP) を対として密結合 (容量4K語の通信メモリCMや容量256K語のローカル環境格納用メモリ・モジュールLEMMを共有) する方式で構成されている. PE内の各ファシリティは,プロセス制御ユニット (PCU) が管理する. また,ORPは4個のユニフィケーション・ユニット (UU) を装備している.

ネットワークの中間ノードはネットワーク・ノード・ユニット (NNU) と呼ばれ,完全にハードウェア化された多重バスのスイッチ機構を持ち,容量256K語のグローバル環境格納用メモリ・モジュール (GEMM) の管理を行っている. GEMMは,論理的な機能はLEMMと同一である. ネットワークの根からあるPEに至るノード上の各GEMMは,そのPE内のLEMMも含めて,そのPEのメモリ空間を構成する. したがって,各GEMMはいくつかのPEが共有しているものとみなせる.

KPRのプロセッサ間ネットワークの設計方針には,(i) PE台数が数百となっても,ハードウェア量が極端に多くならないこと;(ii) 頻度の多い通信に対しても衝突が余り激しくならないこと;(iii) イベントの飛び越しがあるので,任意のPE間での通信が行えること;などがある. これらの条件を満たす方式として,多段スイッチ結合方式,木状結合方式,直接結合方式がある. シミュレーションも交えた定量的な検討および定性的な考察[16]によって,これらのうち,ハードウェア量の少ない割には拡張性に富む2分木状ネットワーク方式を採用することにした. さらにKPRでは,(a) 多段スイッチ結合方式に比べて平均通信路長が長い;(b) 根に近いノードの負荷が大きい;といった2分木状ネットワークの持つ欠点を,PCUとNNUの負荷管理専用機構や,ARP-ORP対というPEのハードウェア構成方式を活用した動的負荷分散機能[3],などによって解決している.

PCUやGEMM管理プロセッサは,ページ分割方式によってLEMMやGEMMの管理を行う. すなわち,PRモデルでは,環境データに対するメモリ領域の割り付け/解放タイミングが各プロセスによって制御されているので,実行時,(L/G) EMMに対するアクセス要求が発生する時点ごとに領域を管理 (割り付けや解放) する機能のみで十分である. 例えば,サブプロセス情報を割り付けたメモリ領域は,「もうこれ以上解がない」というイベント (*fail* メッセージ) が発生した時点で解放することができる. したがって,KPRには,ガーベジ・コレクション機能は不要であり,これを装備していない.

2分木状のプロセッサ間結合ネットワークは,主としてプロセス管理情報を通信するコマンド・バスと環境情報を通信するデータ・バスから成る. 2重系のバスにより性質の異なる2種類の情報をそれぞれ独立に通信することができる.

プロセッサ間通信は,各々のバス上を流れる情報の性質の違いを反映して,2つのバスでは,異なった方式で行われる. (a) コマンド・バスを流れるプロセス管理情報や環境情報の一部である引数情報は,受信側のプロセッサによって即時的に (割り込み) 処理されるべきものであり,送信側のPCUによって通信路を確保した後,PCU間で直接的に情報の授受を行う. (b) データ・バスを流れる環境情報は,プロセッサ内でその環境を処理するプロセスが実行されて初めて必要になるものである. したがって,まず送信側でプロセッサ群 (クラスタ) の根に当たるノードにある共有メモリGEMMに対しデータを転送し,受信側は,

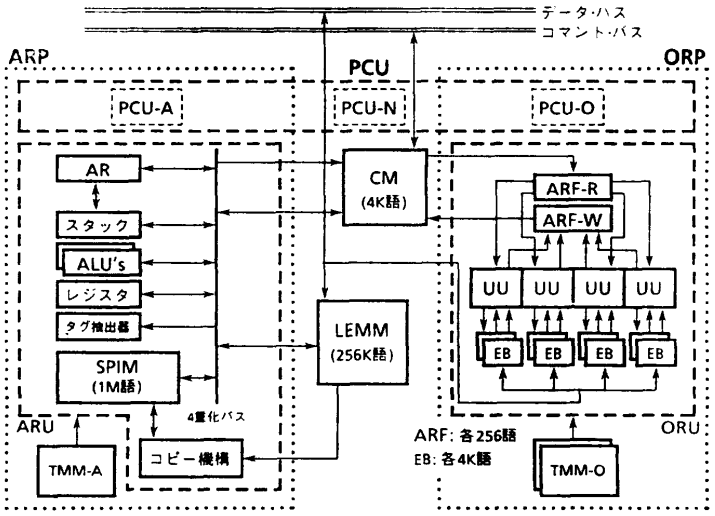

図5 要素プロセッサの構成
Fig. 5 Hardware organization of a processor element.

自分がこのデータを必要とする時点でこの GEMM に対してアクセスを行う．GEMM アクセスのためのアドレス情報などは，コマンド・バスを用いて PCU が NNU 内の GEMM 管理プロセッサに転送する．

2分木状ネットワークの根に当たるノードにはシステム管理プロセッサ SVP が結合され，KPR のシステム全体の管理を行う．その主な仕事は，KPR-L プログラムのコンパイル，入力された質問に対する初動プロセスの起動，入出力などに関する組み込み述語の処理，などである．これらの処理はいずれも論理プログラムの実行とはほぼ独立したものであり，また処理の柔軟性が要求され，専用ハードウェア化も困難であるので，既存のワークステーションを利用している．

KPR では，データを 32 ビット長とし，これに8ビットのタグを付加して，40 ビットを1語長とするタグ・アーキテクチャを採用している．

なお，KPR は，大規模な応用に対して実用上十分な速度が得られることを目標としており，要素プロセッサ 512 台から成るマルチプロセッサ・システムを最大構成規模として設計されているが，現在開発しているプロトタイプでは，図4に示すように，PE 2 対と NNU 1 個，SVP 1 台を実装しており，DBP は SVP が兼任する．

3.2 論理プログラムの処理方式

KPR における論理プログラムの処理・実行は，**図 6** に示すような処理フローに従って行われる．

まず，SVP は，KPR-L プログラム（主に規則や事実から成る）を「定義体テンプレート」と呼ばれる命令オブジェクト表現にコンパイルする．定義体テンプレート表現の機能レベルが，各プロセッサの命令セット・アーキテクチャに対応する．この定義体テンプレートは，節に関する種々の情報を表現している．

節の頭部に対応する ORP の定義体テンプレート表現は，ORP のマシン命令としてそのままテンプレート・メモリ・モジュール TMM (Template Memory Module)-O に格納される．これに対して，節の本体部に対応する ARP の定義体テンプレート表現は，さらに ARP 用マイクロプログラムに展開された後，TMM-A に格納される．

そして，SVP を介して質問が入力されるたびに，各プロセッサはその定義体テンプレートを解釈あるいは実行し，推論を進める．ユーザは，SVP を介して質問の答を得る．

すなわち，KPR では ARP と ORP の機能が異なるので，各命令格納用メモリである TMM に格納される情報の性質や処理方式に次のような違いを設けた．

（1） Oプロセスで必要な情報は ORP のマシン命令として TMM-O へ格納される．ORP では，4個の UU が各々独立に TMM-O から読み出したこれらの命令を各 UU のマイクロ命令で解釈し実行する．この2レベル命令制御方式の利点によって，TMM-O の容量を小さくできる．UU の制御記憶は各 UU ごとに装備されている．

（2） Sプロセスで必要な情報は ARP 用マシン命令列からさらに展開されたマイクロ命令列の形式で TMM-A へ格納され，それらが直接 ARP の実行を制御する．したがって，ARP 用命令とは SVP における論理プログラムのコンパイル過程で一時的に現れる中間コード表現であり，それが ARP のファシリティ内に格納されることはない．ARP では，イベント処理など個々の処理間の連続性が低いので，2レベル命令制御方式を採用すると，マシン命令フェッチに要する時間がマイクロ命令実行時間に比べて無視できず効率が低下する．そのため，TMM-A にはマシン命令ではなくマイクロ命令列を格納し，ARP がこれを

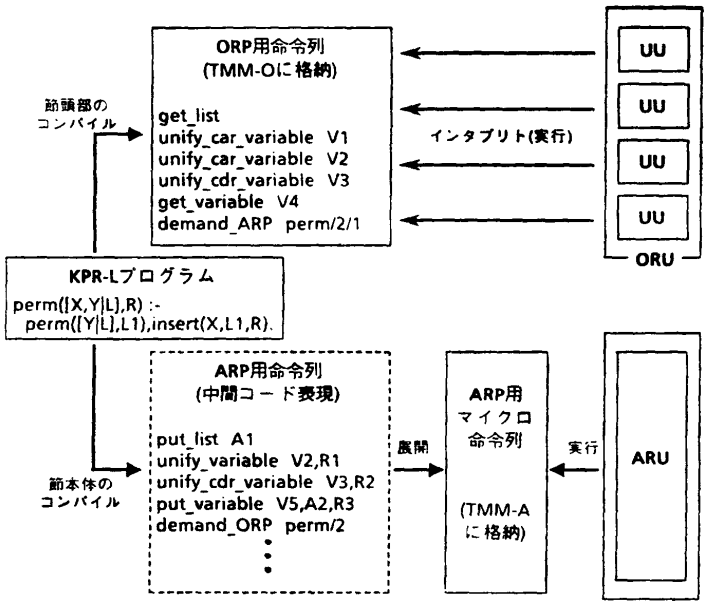

図 6 論理プログラムの処理フロー例
Fig. 6 Example of flow chart on processing a logic program.

直接実行することで効率の低下を防ぐ．TMM-A はARP の制御記憶とみなすことができる．

（3） データベース定義体に対応する命令列は，DBP 内に格納される．

KPR の基本的なマシン命令は，大別すると WAM (Warren's Abstract Machine) の命令セット[4]と同じ機能（ユニフィケーション処理）レベルのものと，並列性を制御（並列リダクション）するための命令との2種類に分けることができる．図6に，マシン命令列へのコンパイル例を示す．KPR-L 以外の，例えばAND 並列型言語への対応，実行モデルの修正，各プロセスの処理やプロセッサの割り付け戦略の変更などに対して，KPR では各種プロセッサのマイクロプログラム（ファームウェア）の書き換えによって対処する．

3.3 AND リダクション・プロセッサ (ARP) の処理機能

ARP は，ストリーム並列処理方式によって AND 並列性を実現する．節本体の最左と最右以外の場所にあるゴール・ブロック1個（サブプロセス）の処理は，以下のとおりである．(1)直前の環境のコピー．(2)解の取り込み．(3)組み込み述語の実行による環境の更新．(4)ユーザ定義述語の呼び出しのための引数設定．(5)ユーザ定義述語の呼び出し．

これらのうち(1)を「コピー機構」と呼ぶ専用ハードウェアで，(2)～(5)を ARU のファームウェアで，処理する．これらの処理は，オーバラップして実行される．

ARU のハードウェア構成は，ARU で行われる処理を基本的な操作に分解し，その各々ができるだけ効率的に実行できるよう設計した．具体的には，図5に示すように，容量 1M 語のサブプロセス情報格納用メモリ (SPIM)，汎用レジスタ・ファイル，高速コピー機構，多方向分岐用タグ抽出器，構造データのたどり用ハードウェア・スタック，CM や LEMM へのアクセス機構となる AR (Address Registers) などのファシリティを，4重化したバスによって相互に接続し，それらをマイクロプログラムによって制御することによって，ファシリティ・レベルの並列性（低レベル並列性）を抽出する方式を採用した．また，環境のコピーという，プログラム制御ではオーバヘッドの大きい処理については，専用ハードウェアを用意し，処理の負荷分散を図った．さらに，ARU の実行環境（ALU など）を2重系として構成することで，PCU-A による実行環境の設定と，実際の ARU での処理とをオーバラップさせ，ARU の遊び時間を極力小さくするようにしている．

3.4 OR リダクション・プロセッサ (ORP) の処理機能

ORP は，PR モデルにおけるOプロセスを処理する．その仕事のうち，定義体の複数の候補節に対し，①1個の本体ゴールと候補節の頭部とのユニフィケーションを，②また節頭部のユニフィケーションに成功した候補節の本体中で，どのユーザ定義述語よりも左方にある組み込み述語の処理を，それぞれ ORU が行う．

1個のOプロセスが本体ゴール列に対応する子の O/S/D プロセスを複数個起動することにより OR 並列性の実現を図っている．また，ORU は4個の UU を装備しており，4個の候補節に対する①②の処理を並列に実行できる．

ORP では，OR 関係にある候補節を4個の UU に割り付けて並列に実行し，ORP のマシン命令は，引数によるインデキシングの機能を持たない．インデキシングを採用するとマシン命令レベルでの順序制御機能が必要となるが，ORP では，この順序制御機能を不要とすることでマシン命令フェッチ機構のハードウェア化を実現している．インデキシング機能を利用しないことによって生じるオーバヘッド，例えば，OR 並列処理開始時の(すぐに fail する節に対する)環境データの EB (Environment Buffer) への書き込みなどは，4個の EB への環境コピーを並列実行するなどの方法で防いでいる．さらに，UU でのユニフィケーション処理の失敗や処理の終了は，UU のマイクロプログラムから直接 ORU へ割り込みを発生して伝える．以上のハードウェア化によって，ORP のマシン命令が条件コードによる順序制御機能を備える必要がなくなり，ORP のマシン命令を先読みすることが可能となった．

4個の UU は TMM-O に格納されている命令を独立に解釈実行し，並列にユニフィケーション処理を行う．CM, LEMM と UU とのインタフェースとしては，図5に示すように，それぞれ ARF (Argument Register File) と EB がある．

4個の UU は非同期で動作しており，CM と UU とのインタフェースとして ARF を設けて，CM へのアクセス競合を時分割制御方式（ARU, PCU および ARF 用に分割）によって解決している．ARF は，

CM からOプロセス1個分の引数情報を読み出すための ARF-R と，引数情報と変数束縛情報を CM に書き込むための ARF-W から成る．各容量は，256 語である．

OR 並列処理を行うために，OR 関係にある候補節の環境は個々に用意する必要がある．また，UU では送られてきた環境を変更し，新しい情報を追加することでユニフィケーションを行う．このため，送られてきた環境情報を候補節の数だけ EB にコピーしなければならない．環境情報は引数情報に比べると大量であり，複数回コピーを行うことはオーバヘッドとなる．ORP では，4個の候補節分の環境情報は一度に4個の UU の EB に書き込むことで，コピーの回数を減らしている．また，EB への環境情報の転送や EB からの環境情報送出には時間がかかるので，EB は2重構成にして，UU での処理と環境情報の転送をオーバラップ実行させている．各 EB の容量は，4K 語である．

ユニフィケーション処理では，ゴールの引数のデータ・タイプに基づく多方向分岐を行って，処理内容を動的に決定する操作が非常に多く現れる．したがって，サイクル・タイムを短くし，命令パイプライン化を行っても，条件分岐によるパイプの乱れが頻繁に起こり，パイプライン化による高速性が発揮されないと考えられる．これらのことから，UU では多方向分岐機能を持った水平型マイクロ命令による制御方式を採用した．

組み込み述語以外のマシン命令は，平均2～3マイクロ命令で実行される．中には，1マシン命令が1マイクロ命令で実行できるマシン命令もあるが，TMM-O からのマシン命令の取り出しをハードウェアで行っていること，マシン命令に順序制御機能がなく先読みできること，マシン命令のオペレーション部をそのまま制御記憶のマイクロプログラム・アドレスとしておりマイクロプログラムへのディスパッチが簡単に行えること，などからこのことはオーバヘッドにならない．

3.5 プロセス制御ユニット (PCU) とネットワーク・ノード・ユニット (NNU) の処理機能

システム規模が大きくなると，1台の特別なユニットがシステム全体の状況を集中管理しながら負荷分散を行うのは現実的ではない．そこで，KPR では各 PE 内のプロセス制御ユニット (PCU) と各ネットワーク・ノード・ユニット (NNU) とが分担して，動的負荷分散を行う．

KPR の各要素プロセッサ (PE) には，高速なプロセス・スイッチやプロセッサ間通信制御などのために PCU が設けられている．PCU は3つのサブ・ユニットから成り，PCU-A が S プロセスおよび ARU，PCU-O がOプロセスおよび ORU の制御を行う．また，PE 内では，ARP と ORP とがネットワークに対する通信ポートを共有する．したがって，PE のネットワークへのインタフェース・ユニットとして PCU-N が設けられている．PCU では，3つのサブ・ユニットにそれぞれの機能を分散させることにより，外部からの割り込みに対する応答の高速化を図っている．プロセス管理に必要な静的な情報は，ユニット間通信用のマシン命令のオペランドとして TMM-A/O 内に格納され，ARU/ORU がそれらの命令を実行することにより，その時点で必要な情報のみが PCU に伝えられる．PCU は，(i) 通信やプロセスの効率が良くなるようにスケジューリングする，(ii) PE におけるプロセス・スイッチを高速に行う，(iii) 動的負荷分散を実現する，などの目標を持つ．これらを実現するために，①優先度に基づくプロセス・スケジューリング，②タスク・キューの高速処理，③高速割り込み処理，の各機能をハードウェア化している．

NNU は，2分木状のプロセッサ間結合ネットワークの葉 (PE) 以外のノードを構成し，各 PE 間の通信を管理する．PE はデマンドやイベントをネットワークに対して送信し，それらは1つ以上の NNU を経由して受信側の PE に受け取られる．NNU は，負荷伝播線による負荷管理と連動する高速バス・スイッチ機構を用いて各 NNU 間のバスを順次切り換えていく方式により，PCU が発したデマンドの送出先を，負荷量が最も少なく，かつ通信パスのできるだけ短い PE へと動的に決定する．ネットワーク・トポロジは異なるが，このような自動負荷分散機能をネットワークに付加したアーキテクチャは，PIE[5] でも採用されている．

KPR が採用している2分木状ネットワークの欠点の1つは，ランダムに通信を行った場合，木の根に近いノードが隘路になりやすいことである[16]．したがって，このオーバヘッドの存在を負荷分散戦略に反映させる必要がある．すなわち，KPR における通信には，局所性を持たせる必要があり，現在の負荷分散戦略では，できる限りデマンドを発する PE (根ではなく葉) に近い GEMM を使用するような負荷分散戦略

を採用している．また，並列解探索では，「並列性の爆発」が問題となる．この問題には，PCU が担当しているプロセス管理機能としてのプロセス・スケジューリング方式で対処している．具体的には，深さ優先方式と幅優先方式とを融合したスケジューリング方式を用いている．これらの動的負荷分散方式およびシミュレーション結果については，文献3）で詳しく述べている．

4. 他の推論マシンの並列処理方式との比較

4.1 対象プログラミング言語の機能

「OR 並列型言語」と呼ばれる論理型言語は，論理プログラムの AND/OR 並列性を制限するセマンティクスを持たない，いわゆる全解探索を指向する言語である．一方，GHC[6] などの「AND 並列型言語」と呼ばれる論理型言語では，「トラスト」あるいは「コミット選択」と呼ぶセマンティクスを入れて，OR 並列性を制限している．

この並列論理型言語の分類法によると，KPR-L は「OR 並列・AND 逐次型言語」と位置付けることができる．すなわち，定義体の節頭部を並列に実行（OR 並列実行）し，1個の節本体のゴール列を左から右に逐次に実行（AND 逐行）するというセマンティクスを持つ．したがって，プログラマは AND 関係にある本体ゴールの実行順序も意識してプログラミングする必要がある．

KPR-L のような OR 並列型言語と，GHC などの AND 並列型言語は，各々適応する問題分野が異なると考えられる．したがって，KPR の応用分野を，OR 並列処理向き問題だけではなく，論理型言語で記述できる問題分野に拡大することも必要である．KPR では，AND 並列型言語の処理に必要となる機能をファームウェアとソフトウェアで実現する方式を採用して，応用分野の拡大を図っている．すなわち，KPR-L プログラムの AND 関係にあるゴール列は基本的に逐次実行されるため，AND 並列型言語のセマンティクスを満足するためには，（i）トラスト機能，すなわちガード部の実行が成功した節の中から任意に1つをコミットする機能，OR 並列性を制限する機能；（ii）AND 関係にあるゴールの並列実行，すなわち AND 並列性の実現，共有変数間の通信；といった機能を実現するように，KPR-L のセマンティクスを拡張する（例えば，ファームウェア化された組み込み述語としてこれらの機能を付加する，プログラム変換ソフトウェアを利用するなど），あるいは実行モデルを変更する，などの対処が必要である．

AND 並列型言語 KL 1 を対象としているマルチ PSI[7] などでは，ユーザ指定による負荷分散方式を提案しているが，KPR が対象としている KPR-L は OR 並列型の言語であるため，負荷分散プラグマ使用に関する問題はより難しいと考えられる．まず，AND 並列型言語ではプログラマに対してプロセスの概念を陽に意識させるのに対して，OR 並列型言語ではプロセスを特に意識させることはない．したがって，ユーザにプログラムの並列実行下の状況把握を強いるのは無理な場合が多い．また，OR 並列型言語では，述語呼び出しの際の変数に対して入出力のアクセス・モードの制限はなく，述語の呼び出し方によって，プロセス木の構造は全く異なったものとなる．動的な実行過程の中で，同じ述語が何回も呼び出される場合，それがいつも同じ状況で呼び出されるとは限らないし，また，常に同じ呼び出し方をされるとも限らない．このような理由により，現時点では，負荷分散プラグマの使用を考えていないが，今後，本格的に取り組む必要もあろう．

4.2 実行モデルと処理方式

論理型言語処理システムのマルチプロセッサ上での実現に対して，AND/OR 木の各ノードにプロセスを割り付けるという考え方を最初に示したのは，Conery ら[8] である．PR モデルも，基本的には彼らの AND/OR モデルに従ったものであるが，次のような点でそれとは異なる特徴を持つ．（1）PR モデルのプロセス粒度が粗い，すなわちプロセス機能の大きさが，O/S プロセスとともに Conery のモデルのプロセス粒よりも大きい．（2）PR モデルでは，OR 並列処理機能を O プロセスと D プロセスとに分担させている．Conery らは D プロセスに相当する機能をサーチ並列性を持つ機能として暗示したものの，AND プロセスと OR プロセスとについて議論しているだけである．（3）PR モデルでは，プロセス割り付けやプロセス間通信において最適化戦略を適用している．（4）Conery らの AND/OR モデルでは深さ優先的なプロセス・スケジューリングを行うことがモデルに含まれているが，PR モデルではスケジューリングに関する要素を含んでいない．

並列論理型言語の AND 並列性を実現するための処理方式には，ストリーム並列処理方式以外にも，（1）AND 並列型言語の処理系の大半が採用してお

り，AND 関係にあるゴール列の各々を並行プロセスとして実行する「pure-AND 方式」；(2) 変数の共有関係に着目して，限られた AND 関係のゴール列のみの並列実行を明示したプログラムに，論理プログラムを変換する「Restricted-AND 方式[9]」；などがある．AND 関係にあるゴールが完全に並列実行されるようなセマンティクスを持つ AND 並列型言語では，前節で述べた (i) (ii) の機能を実現する必要がある．したがって，ストリーム並列処理方式では，その機能を完全に実現することが困難となるが，全解探索プログラムなどの処理の場合にはこの方式の方が効率が良い．

4.3 アーキテクチャおよびシステム構成方式

KPR で採用した命令駆動方式は「要求駆動方式」であり，論理型言語で記述されたプログラム，あるいはそれから変換された中間言語命令列をリダクション・マシンで実行する方式である．これは，論理型言語のセマンティクス（ゴールの書き換え）をそのまま実行する高級言語マシンの一種とみなすことができる．論理プログラムをいったんデータフロー・グラフに変換し，それをデータフロー・マシン上で実行する「データ駆動方式」を採る ICOT らの PIM-D[10] などとは，この命令駆動方式上の違いがある．

一般的にリダクション・マシンにおいては，Lisp マシンを代表とする関数型言語向きマシンよりも，Prolog マシンを代表とする論理型言語向きマシンの方が，アーキテクチャの抽象度が高いと言える[1]．したがって，London 大学の ALICE[11] のように関数型言語と論理型言語との両方に向いた汎用リダクション・マシンとは異なり，KPR では論理型言語のみに対象を絞ることにより，専用化の度合をより強めていると言える．

KPR は，PR モデルとして提案した実行モデルが論理型言語の自然な論理的粒度を表現しているものと考え，マシン・アーキテクチャの物理的粒度をできる限りそれに合わせる方式に基づいて設計されている．例えば，富士通らの株分け方式[12]や ICOT の PIM/p[13] などは，KPR よりは論理的粒度を粗くとらえているものと考えられる．また逆に，東京大学の PIE[14] などは KPR より細かい粒度を持つアーキテクチャである．KPR と同じ機能レベルの論理的粒度の実現を指向したマシンとしては，ICOT らの PIM-R[15] があるが，KPR ではこの論理的粒度をさらに細かく機能別に分け，各々を別々の物理的粒度として実現するヘテロジニアス・マルチプロセッサ構成方式を採用している点に特徴がある．

5. おわりに

KPR のハードウェアは，TI 社の 74 ACT 88 xx シリーズのビルディング・ブロック（シーケンサや ALU など）を中心に構成しており，市販の汎用マイクロプロセッサ・チップは使用していない．

現在は，ハードウェア，ファームウェア，ソフトウェアの実装設計中であり，並行して，対象言語を AND 並列型言語にした場合の処理方式などについても検討している．すなわち，ストリーム並列処理方式を用いずに，KPR 上に Flat GHC の処理システムを実現する方法については，現在，新しい実行モデルを構築する方向で検討している．

本論文で述べた KPR については，次のような諸点における定量的考察という課題が残されている．(1) PE とネットワークの機能バランスが取れているか．特に，コピー方式の採用により，コピーのオーバヘッドがどれくらい生じるのか，メモリが容量不足とならないか．(2) PE 内で ARP と ORP の処理機能のバランスが取れているか．(3) PE の VLSI 化が可能か．(4) KPR における AND 並列型言語の処理機能の実現．これらについては，プロトタイプ実機上での評価やシミュレーションの結果などもフィードバックして，さらに考察を加える必要がある．

参考文献

1) 柴山：記号処理マシン，情報処理，Vol. 28, Nol. 1, pp. 27-46 (1987).
2) 柴山ほか：論理型言語向き並列計算機 KPR のアーキテクチャ，*Proc. of the Logic Programming Conf. '87*, ICOT, pp. 183-192 (1987).
3) 柴山ほか：論理型プログラミング言語向き並列計算機 KPR の並列処理方式，並列処理シンポジウム JSPP '89 論文集，情報処理学会など，pp. 91-98 (1989).
4) Warren, D. H. D.: An Abstract Prolog Instruction Set, Report of Stanford University Computer Science Department, SRI-309, pp. 1-30 (1984).
5) 山内，田中：並列推論マシンにおける負荷分散方式の評価，情報処理学会研究報告，Vol. 88-ARC, No. 71, pp. 125-132 (1988).
6) Ueda, K.: Guarded Horn Clauses. *Proc. of the Logic Programming Conf. '85*, ICOT, pp. 225-236 (1987).
7) Takeda, Y. et al.: A Load Balancing Mecha-

nism for Large Scale Multiprocessor Systems and Its Implementation, *Proc. of the Int. Conf. of Fifth Generation Computer Systems*, ICOT, pp. 978-986 (1988).
8) Conery, J. S. and Kibler, D. F.: Parallel Interpretation of Logic Programs, *ACM Symp. on Functional Programming Language and Computer Architecture*, pp. 163-170 (1981).
9) DeGroot, D.: Restricted And-Parallelism and Side Effects, *1987 Symp. on Logic Programming*, IEEE and ACM, pp. 80-89 (1987).
10) Ito, N. et al.: The Architecture and Preliminary Evaluation Results of the Experimental Parallel Inference Machine PIM-D, *13th Annual Int. Symp. on Computer Architecture*, pp. 149-156 (1986).
11) Darlington, J. and Reeve, M.: ALICE A Multi-Processor Reduction Machine for the Parallel Evaluation of Applicative Languages, *ACM Conf. on Functional Programming Languages and Computer Architecture*, pp. 65-74 (1981).
12) Kumon, K. et al.: KABU-WAKE: A New Parallel Inference Method and Its Evaluation, *COMPCON Spring 86*, pp. 168-172 (1986).
13) 服部ほか：並列推論マシン PIM/p のアーキテクチャ，並列処理シンポジウム JSPP '89 論文集，情報処理学会など，pp. 107-114 (1989).
14) 平田ほか：PIE のメモリ試作ハードウェアの設計について，電子通信学会技術研究報告，Vol. EC-85, No. 64, pp. 55-65 (1986).
15) 尾内ほか：リダクション方式並列推論マシン PIM-R のアーキテクチャ，*The Logic Programming Conf. '85*, ICOT, pp. 14-25 (1987).
16) 八田ほか：並列リダクション・モデルに基づく Prolog マシンのハードウェア構成，第 30 回情報処理学会全国大会論文集，No. 6 C-4, pp. 191-192 (1985).

(平成元年 5 月 23 日受付)
(平成元年 9 月 12 日採録)

柴山 潔（正会員）

昭和 26 年生．昭和 49 年京都大学工学部情報工学科卒業．昭和 54 年同大学院博士課程単位修得退学．同年同大学工学部情報工学教室助手．昭和 61 年同助教授，現在に至る．工学博士．計算機システム，計算機アーキテクチャなどの教育・研究に従事．電子情報通信学会，人工知能学会，IEEE，ACM 各会員．ICOT・WG 委員．昭和 61 年度本学会論文賞受賞．

鹿毛 裕史

昭和 40 年生．昭和 63 年京都大学工学部情報工学科卒業．現在同大学院修士課程情報工学専攻在学中．

川倉 康嗣（学生会員）

昭和 39 年生．昭和 63 年京都大学工学部情報工学科卒業．現在同大学院修士課程情報工学専攻在学中．

山本 雅亮（正会員）

昭和 39 年生．昭和 62 年京都大学工学部情報工学科卒業．平成元年同大学院修士課程情報工学専攻修了．同年通商産業省に入省．現在同省機械情報産業局電子機器課に勤務．在学中 KPR 開発プロジェクトに参画．

平田 博章（正会員）

昭和 38 年生．昭和 62 年京都大学工学部情報工学科卒業．平成元年同大学院修士課程情報工学専攻修了．同年松下電器産業(株)に入社．現在同社情報通信研究センター情報通信関西研究所に勤務．在学中 KPR 開発プロジェクトに参画．

加納 健（正会員）

昭和 37 年生．昭和 62 年京都大学工学部情報工学科卒業．平成元年同大学院修士課程情報工学専攻修了．同年日本電気(株)に入社．現在同社 C&C システム研究所コンピュータ・システム研究部に勤務．在学中 KPR 開発プロジェクトに参画．

萩原 宏（正会員）

大正15年生．昭和25年京都大学工学部電気工学科卒業．NHKを経て，昭和32年京都大学工学部助教授，昭和36年同教授，現在に至る．工学博士．情報理論，パルス通信，電子計算機などの研究に従事．昭和31年度稲田賞受賞．昭和50，62年本学会論文賞受賞．昭和56～58年度本学会副会長．著書「電子計算機通論 1～3」「マイクロプログラミング」など．電子情報通信学会，ACM，IEEE 各会員．