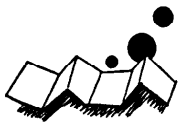
解　説

プロダクションシステムと並列処理†



石　田　　亨††

1. はじめに

プロダクションシステムはエキスパートシステム構築用の基本メカニズムとして幅広く用いられている．化学 (DENDRAL[5])，医療 (MYCIN[6])，数学 (AM[7])，音声認識 (Hearsay[8]) 等へのプロダクションシステムの応用の成功は，応用独立 (Domain Independent) なプロダクションシステム開発用ツール[9)~12)]を生み出す刺激となった．本解説では，プロダクションシステム全般に関する説明は文献1)～4)に譲り，並列処理に関する研究が比較的進んでいるプロダクションシステム記述言語 OPS[12)~17)]を取り上げ，言語仕様，応用システム，及び並列計算機アーキテクチャを概説する．

OPS は CMU (カーネギメロン大学) で開発された応用独立なプロダクションシステム記述言語である．今日に至るまで数々の実験システムと，AT & T (アメリカ電信電話会社) の ACE[18)]，DEC (Digital Equipment Corporation) の R1/XCON[19)~21)] 等の実用システムを生み出している．応用が広がるに伴って，OPS の実行速度が問題とされるようになった．効率の良いマッチングアルゴリズム (RETE[14)]) の採用，BLISS や C 等の手続き型システム記述言語による OPS 処理系の記述，コンパイラの開発等のソフトウェア上の努力の結果，現在に至るまでに100倍以上の効率向上が達成されている[38)]．しかし，今後のエキスパートシステムの規模の増大や応用領域の多様化 (例えばリアルタイム処理[23)]) を考えると，なお効率向上の努力が必要であり，これがプロダクションシステム用並列計算機の研究動機となっている．

プロダクションシステム用の並列計算機としては，コロンビア大学の DADO プロジェクト[25)~34)] と CMU の PSM プロジェクト[35)~39)] の研究が注目されている．DADO プロジェクトでは，多数のプロセッサを2進木状に結合した木構造計算機を用いてプロダクションシステムを並列実行させる研究を進めている．PSM プロジェクトではプロダクションシステムの特性評価を行い，その結果に基づいてアーキテクチャの提案を始めている．いずれのプロジェクトも，それぞれ ACE，R1/XCON 開発の経験を持つ研究者達が中心となっていることが注目を集めているゆえんである．

本解説では，まず2章でプロダクションシステム記述言語 OPS と，OPS を用いて記述されたエキスパートシステムを概説する．次に3章でプロダクションシステムに含まれる並列性と CMU で行われたプロダクションシステムの特性測定結果について述べる．最後に4章で木構造計算機 DADO を中心にプロダクションシステム用並列計算機のアーキテクチャを紹介する．

2. プロダクションシステム

2.1 プロダクションシステム記述言語 OPS

OPS は数多くの問題領域の記述を目的として CMU で開発されたプロダクションシステム記述言語であり，同じく CMU で開発された PSG, PSNLST[11)] から多くの特徴を継承している．ここではまず最も多くのユーザに使用されてきた OPS5[16)] を取り上げ，構文，実行規則を説明し，その後に OPS の他のバージョンの特徴に簡単にふれる．

(1) OPS5 の構文

OPS5 では，プロダクションシステムは手続き型言語の IF 文に近いプロダクションと呼ばれる条件文の集合である．それぞれのプロダクションは，left-hand-side (LHS) と呼ばれる条件要素 (condition element) の連接 (conjunction) と，right-hand-side (RHS) と呼ばれる動作 (action) の集合から構成される．プロダクションの格納場所はプロダクションメモリ (production memory) と呼ばれる．一方，プロダ

† Parallel Execution of Production Systems by Toru ISHIDA (Yokosuka Electrical Communication Laboratory, NTT).
†† 横須賀電気通信研究所知識ベース研究室

クションシステムの実行に必要なデータは，ワーキングメモリ (working memory: WM) と呼ばれるシステム共通のデータベースに格納される．WM 中の各データはワーキングメモリエレメント (working memory element: WME) と呼ばれている．各プロダクションはその実行に際しこの WM を参照し，変更を加える．即ち，LHS に書かれた条件がすべてその時点の WM の内容によって満足されると，RHS が起動され WME の追加削除が行われる．

言語の詳細を例を用いて説明する．まず以下の WME が現在の WM 中にあるとする．

(city ↑name Buffalo ↑state New-York)

OPS5 の WME は属性/値の組(attribute value pair)で表わされる．属性と値を区別するために，属性には“↑”が付加されている．この例では“city”は WME のクラス(class)名を表現している．“name”,“state”は属性名，“Buffalo”“New-York”は各属性の値である．

次にプロダクションの例を示す．

```
(P make-possible-trip
    (city ↑name ⟨x⟩ ↑state New-York)
  −(weather-forcast ↑place ⟨x⟩
      ↑date tomorrow ↑weather rainy)
  −−⟩
    (make possible-trip ↑place ⟨x⟩
      ↑date tomorrow))
```

先頭の“P”はプロダクションの記述開始を表わす．プロダクション名は“make-possible-trip”である．OPS5 では2種の条件要素が許されている．正の条件要素 (positive condition element) は，もし条件要素にマッチする WME が存在すれば満足される．一方，負の条件要素 (negative condition element) はマッチする WME が WM 中に存在しない時に満足される．負の条件要素は“−”を前に付けることによって，正の条件要素と区別する．上記の例では，“(city...)”が正の条件要素，“−(weather-forcast ...)”が負の条件要素である．条件要素の中には変数を記述できる．変数は正の条件要素のマッチを通して値に束縛される．上記の例では，“⟨x⟩”が変数“x”を表わしている．したがって，上記のプロダクションは以下のように読める．

IF

New York 州の市を表わす WME が存在し，かつ

明日その市の天気が雨であることを示す WME が存在しなければ，

THEN

明日その市へ旅行することが可能であることを示す新しい WME を WM 中に追加しなさい．

この場合，このプロダクションの LHS は満足され，結果として以下の新しい WME が WM に追加される．

(possible-trip ↑place Buffalo ↑data tomorrow)

(2) OPS5 の実行規則

OPS5 は前向きのやりなおしなしのプロダクションシステム (irrevocable forward chainning production system) である[2]．プロダクションシステムのインタプリタは以下のサイクルを繰り返し実行する．

Match: すべてのプロダクションについて LHS とその時点の WM とのマッチングを行う．

Conflict Resolution: マッチングの成功したプロダクションの中から1つを定められた戦略に従って選び出す．

Act: 選択されたプロダクションを実行し，WME の追加削除を行う．

Match フェーズは，マッチングの成功したプロダクションと，成功に寄与した WME を組合わせて出力する．1つのプロダクションとマッチする WME が多数ある場合には，1つのプロダクションについて多数の組合わせが出力される．この集合はコンフリクトセット (conflict set) と呼ばれる．Conflict Resolution フェーズでは，この組合わせの集合から1つの組合わせが選択される．(これは発火させるプロダクションを1個選ぶという意味の他に，そのプロダクションの発火原因となる WME をも選択することを意味する．)

OPS5 におけるコンフリクトレゾリューション (conflict resolution) の戦略は以下のとおりである．

① 以前選択したプロダクションと WME の組合わせは再び選択しない．これは同一処理を永久に繰り返さないための手段である．

② 最も最近作られた WME を含む組合わせを選択する．これは深さ優先探索を意味する．

③ LHS の条件要素数が多いプロダクションを含む組合わせを選択する．これは厳しい条件を満足したプロダクションの優先を意味する．

(3) RETE アルゴリズム[14]

最も単純な Match アルゴリズムは，各プロダクションサイクルごとに全プロダクションを全 WME に対してマッチさせる方法であろう．しかしこれでは，1,000個のプロダクションと1,000個の WME があれば，1サイクルごとに1,000×1,000のマッチング操作が必要となる．OPS5では，プロダクションシステムからRETE ネットワークと呼ばれるデータフローグラスを作成し，ネットワーク内に Match フェーズの中間結果をすべて保存することによって，各サイクルごとに新たに行うべきマッチング処理を最小限におさえる方法を採用している．

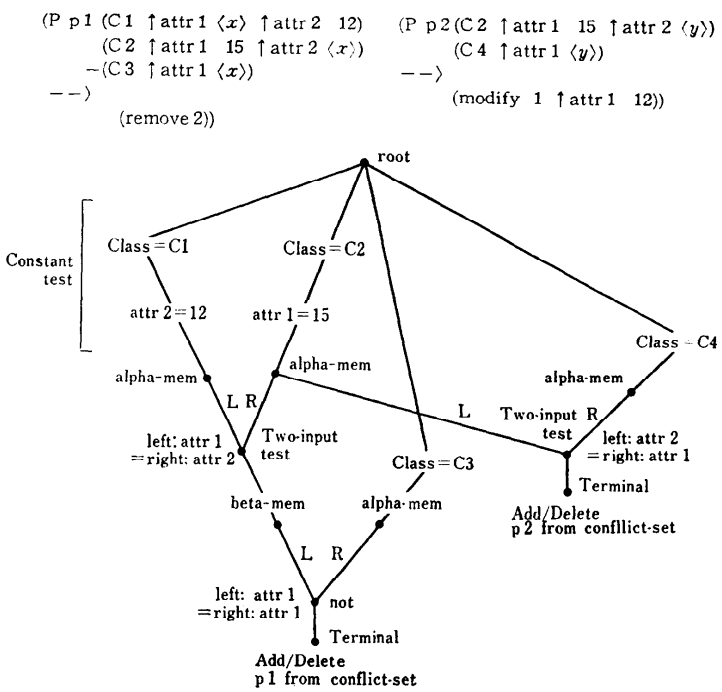


図-1 RETE ネットワークの例[17]

図-1 に RETE ネットワークの例を示す．例えば，Act フェーズで新たに WME が追加されると，その WME が RETE ネットワークの中に流し込まれ，変化に伴うネットワークの更新が行われる．まず属性値の簡単なテスト (constant test と呼ばれる) を通過した WME は，いったん α メモリ (alpha-memory) と呼ばれる WME のプールに蓄えられる．α メモリ上の WME は，必要に応じて他の α メモリ上の WME との間で Join (two-input test と呼ばれる) がとられる．Join の結果である WME の組は β メモリ (beta-memory) に出力される．RETE ネットワークの終端 (terminal) にたどりついた WME の組は終端に対応するプロダクションと組合わされてコンフリクトセットに追加される．

このように RETE アルゴリズムでは，前サイクルで行われた WME の追加/削除に伴う Match 処理だけが行われる．このため，サイクルごとの WM の変化が少ないプロダクションシステムを特に効率良く実行することができる．また，図-1 に示すように，RETE ネットワークの各ノードは複数のプロダクションで共用されるので，LHS がよく似たプロダクションが多数あるシステムでは一層の効率向上が期待できる．この RETE アルゴリズムは OPS の全バージョンに用いられている．

(4) OPS の各バージョン

OPS が多くのユーザに用いられるようになったのは，OPS4[15] からである．AT & T とコロンビア大学により開発された ACE は OPS4 で記述されている．その後，R1/XCON の開発 (DEC と CMU) に並行して，OPS は OPS4，Lisp バージョン OPS5，Bliss バージョン OPS5 と発展をとげ，実行速度も大幅に向上する．この間の事情は文献 20) に詳しい．OPS4 と OPS5 の主な相違はデータの表記法にある．OPS4 がリスト形式の表記であったのに対し，OPS5 は属性/値の組での表記を許している．

最新の OPS は OPS83[17] と呼ばれ，Bliss バージョン OPS5 に比べ，4倍以上速いと報告されている．OPS83 ではプロダクションシステム記述言語と手続き型言語の融合が図られており，①データ型が導入され，②コンフリクトレゾリューションがユーザに開放される等，大幅な仕様変更が行われている．

2.2 OPS で記述されたエキスパートシステム

OPS で記述されたエキスパートシステムは研究用，実用を含め，数多く報告されている．現段階でのプロダクションシステムの応用のイメージ (用途，規模，実行時間) を明確にするために，以下に各システムの

特徴を簡単にまとめる.

(1) ACE[18]

ACE (Automated Cable Expertise) は, AT & T Bell 研究所がコロンビア大学の協力の下に開発した電話回線保守用のシステムである. ACE の入力となるデータは CRAS と呼ばれる既存の回線保守用データベースから供給される. CRAS は顧客と保守要員からの故障に関する膨大な量のレポートを蓄えている. ACE はバッチ形態で日々 CRAS の情報から電話回線の故障を分析し, レポートを作成し, 保守に有効な情報をまとめる.

ACE は数100の OPS4 ルールと約50の Lisp 関数で記述され, 40万回線分の CRAS データを VAX 11/780 の CPU 時間約1時間で処理すると報告されている.

(2) R1/XCON[19)-21)]

R1/XCON は DEC と CMU により共同開発された計算機システムの構成配置を行うシステムである. ユーザからの要求を満たす計算機システムの構成要素名と個数を入力し, 付随して必要となる構成要素を検出追加した後, システムの空間的配置を出力する. 現在では VAX 11/780 をはじめ DEC の10機種にもおよぶ計算機システムの構成配置に用いられており, 1983年末までに8,000件以上の要求が処理されたと報告されている. R1/XCON は Bliss バージョン OPS5 で3,300ルールと, 5,500の計算機システム構成要素の記述からなる.

(3) XSEL[22)]

XSEL は計算機システムのセールスを支援するシステムで, DEC と CMU により共同開発中である. XSEL はユーザからの計算機システムに対する要求を入力とし, 必要とされる主要な構成要素名と個数を出力する. XSEL の出力はそのまま R1/XCON の入力となる. XSEL は OPS5 により記述が進められている.

(4) YES/MVS[23)]

YES/MVS (Yorktown Expert System for MVS Operation) は, IBM T. J. Watson 研究所で開発された MVS (Multiple Virtual Storage) オペレーティングシステムのオペレーション支援システムである. YES/MVS は MVS からのコンソール表示メッセージを要約解釈し, オペレータが行うべき適切な処置を指示する. プロダクション数は約500と報告されている. YES/MVS では連続的なリアルタイム処理に適するよう OPS5 の言語仕様, 処理系に変更を加えている.

(5) DAA[24)]

DAA は CMU で開発された VLSI-DA 支援を目的としたプロトタイプシステムである. DAA は VLSI システムのアルゴリズミックなデータフロー記述を入力し, レジスタ, オペレータ, データパス, 制御情報を出力する. DAA は OPS5 約130ルールで記述されており, VAX 11/750 の CPU 時間約4時間で MCS 6502 マイクロコンピュータをエキスパートが満足するレベルで設計したと報告されている.

3. プロダクションシステムの並列実行可能性

さて OPS 型のプロダクションシステムには, どのような並列性がどの程度含まれているだろうか. この章ではまずプロダクションシステムに含まれる並列性を分類し, CMU で行われたプロダクションシステムの特性測定結果を基に並列処理の効果について議論する.

3.1 プロダクションシステムに含まれる並列性

OPS 型のプロダクションシステムの並列実行可能性は以下のように分類される[30),38)].

(1) プロダクションレベルの並列性 (Production Level Parallelism)

プロダクションシステムを複数のグループに分割し, 複数のプロセッサ上で並列に処理するものである.

① 並列マッチ (Parallel Matching)[30),31),36),39)]

複数のプロダクションのMatchフェーズを並列に実行するものである. Match フェーズには副作用(WM の変更, 入出力処理等) がないため, この並列マッチ実行時にプロセス間通信の必要はない. プロダクションサイクルの90% 以上が Match フェーズに費されている[13)]との報告もあり, プロダクションサイクルの短縮に最も効果を発揮するところと注目されている.

② 並列発火 (Parallel Firing)[32)]

コンフリクトレゾリューションを行わずに, 選択された複数のプロダクションの Act フェーズを並列に実行させるもので, プロダクションシステムの実行サイクル数そのものを削減することをねらいとしている. プロダクション間の通信同期処理が必要となる.

(2) コンディションレベルの並列性 (Condition Level Parallelism)

1プロダクション内にある複数の条件要素 (condition element) を複数プロセスとして並列に処理するものである．条件要素内に共通の変数が含まれている場合にはプロセス間の通信同期が必要となる．

(3) アクションレベルの並列性 (Action Level Parallelism)

プロダクション発火時に，そのプロダクションに指定された複数の動作 (Action) を複数プロセスとして並列に実行するものである．動作の系列にもともと順序性がある場合には，プロセス間の通信同期が必要となる．

この他，複数の独立性の高いプロダクションモジュールを複数プロセスとして並列に実行するモジュールレベルの並列性や，WME の集合に対する操作を並列に処理するエレメントレベルの並列性が考えられるが，現在の OPS にはこれらのレベルの並列性を記述する言語機能はない．

3.2 プロダクションシステムの特性測定

プロダクションシステムの並列実行可能性は，そのプロダクションシステムがどのような静的，動的特性を持つかによって大きく左右される．これらの特性は，まだ十分とは言い難いが，少しずつ分析され始めている．図-2 に CMU の PSM (Production System Machine) プロジェクトで行われた測定結果の一部を示す[37),38)]．この測定は，プロダクション数が100～2000程度の6システムを対象に，RETE アルゴリズムを前提にして行われている．

図-2(a)は各システムの静的特性を示している．システムの大小によらずプロダクションはシンプルで，1プロダクション当りの条件要素数は3～6，動作数は2～4となっている．この結果からコンディションレベル，アクションレベルの並列化はそれほど効果がないと報告されている．

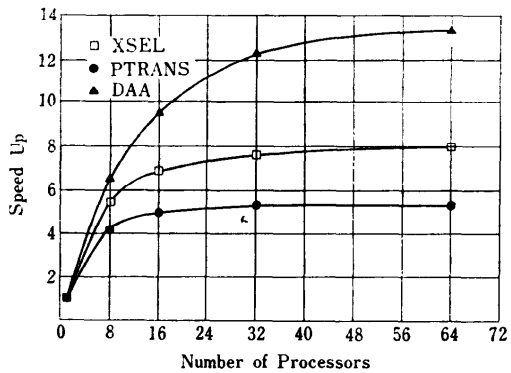
図-2(b)は各システムの動的特性を示している．各欄には1個の WME の追加削除によって，Match フェーズで実行されるテストの回数が記入されている．Constant Test は WME の属性の値を調べるもので，回数は多いが実行に要する時間は Match フェーズの

	R1	XSEL	PTRANS	HAUNT	DAA	SOAR
1. プロダクション数	1,932	1,443	1,016	834	131	103
2. 条件要素数／プロダクション	5.6	3.8	3.1	2.4	3.9	5.8
3. 動作数／プロダクション	2.9	2.4	3.6	2.5	2.9	1.8

(a) プロダクションシステムの静的特性

	R1	XSEL	PTRANS	HAUNT	DAA	SOAR
1. Constant Test/動作	136.3	105.3	122.1	88.5	35.9	26.5
2. Two-input Test/動作	47.1	32.4	35.0	36.8	22.2	39.5
3. Terminal/動作	1.0	1.7	1.7	1.5	2.0	4.0

(b) プロダクションシステムの動的特性



(c) 並列処理の効果

注) PTRANS (factory management), HAUNT (adventure game), SOAR (problem solving) はいずれも CMU で開発されたプロダクションシステム．R1 は DEC に移行する以前のもの．

図-2 CMU でのプロダクションシステム特性計測結果[38)]

10～30%を占めるにすぎない．Two-input Test は追加削除された WME と他の WME の Join を行う操作で，1WME の追加削除に伴って20～50回程度行われている．したがって，この部分を並列化する，即ちプロダクションレベルの並列マッチを行う効果は大きい．また，1サイクルごとにコンフリクトセットに追加/削除されるプロダクション数は3～7回程度である．

図-2(c)に並列マッチとコンディションレベル，アクションレベルの並列処理を総合した効果（シミュレーション結果）を示す．32程度のプロセッサ数で5～12倍の性能向上が期待できると報告されている．

なおこの特性評価では，並列発火，およびモジュールレベル，エレメントレベルの並列処理の効果については，報告されていない．

4. プロダクションシステム用並列計算機

本章では，プロダクションシステムの並列実行を目的として研究が進められている，コロンビア大学の木構造計算機 DADO と CMU の PSM プロジェクトの提案を紹介する*．

4.1 アーキテクチャ選択の要因

マイクロプロセッサを数十～数千用いた並列計算機の研究にはさまざまなアプローチがある[40),41)]．同様にプロダクションシステムの応用領域にもさまざまなものがあり，どのような応用に対してどのようなアーキテクチャが適するかという整理は十分には行われていない．アーキテクチャの選択に影響するプロダクションシステムの要因には次のものが考えられる[30)]．

① PM，WM の大きさ．

② 各プロダクションの WM 参照の局所性．

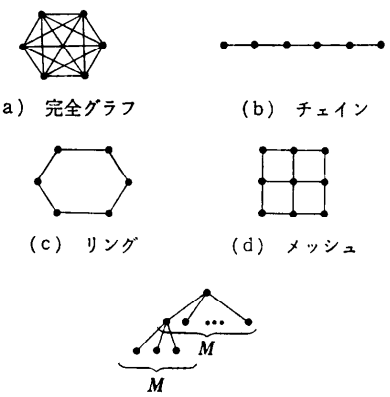
③ 各プロダクションが追加削除する WME の量．

④ 各 WME の追加削除によって次サイクルでマッチング処理が必要となるプロダクション数．

⑤ 各 WME の追加削除によって次サイクルで発火可能となるプロダクション数．

例えば，②が小さければ，各プロダクションは WM の限られた範囲をアクセスできればよい．③が大きければ，RETE アルゴリズムのように Match フェーズの途中経過を保存することは意味を失う．④⑤はそれぞれ並列処理の効果を左右し，プロダクションを割振るプロセッサ数を規定する．

* DADO については論理型言語への適用[33),34)]も検討されているが，本解説では説明を割愛する．



(a) 完全グラフ　(b) チェイン　(c) リング　(d) メッシュ　(e) 木構造，fanout=M

結合方式	ワイヤ本数	アクセス時間
(a) 完全グラフ	$n(n-1)/2$	1
(b) チェイン	$n-1$	$n-1$
(c) リング	n	$n/2$
(d) メッシュ	$2(n-\sqrt{n})$	$2(\sqrt{n}-1)$
(e) 木構造，fanout=M	$n-1$	$2\log_M n$

図-3 マルチプロセッサシステムの結合方式の比較[42)]

4.2 木構造計算機 DADO

4.2.1 アーキテクチャ

DADO はプロダクションレベルの並列化，及びエレメントレベルの並列化をねらいとして以下のアーキテクチャを採用している．

(1) 木構造アーキテクチャ

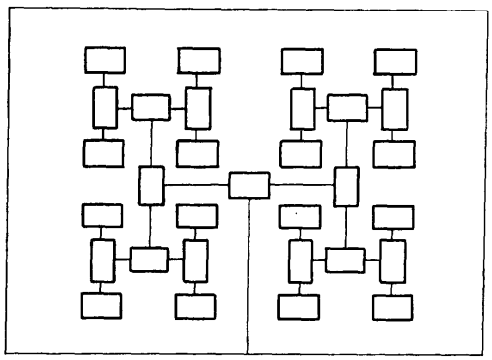
木構造アーキテクチャがプロダクションシステム用計算機アーキテクチャとして選択された理由は以下のとおりである[25)]．

① 木構造計算機は通信能力が優れている．

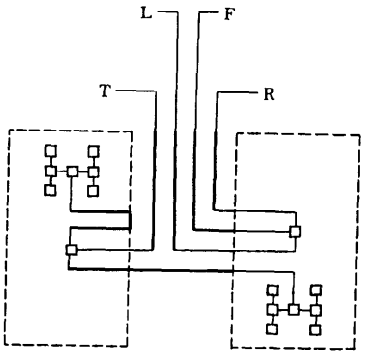
図-3 にさまざまなプロセッサの結合方式を示す[42)]．仮に1万個のプロセッサを結合させるとすると，木構造が約1万本のワイヤを必要とするのに対し，完全グラフは約5,000万本のワイヤを必要とする．ワイヤ数が比較的近いいくつかの方法の中では木構造計算機のアクセス時間が最も短い．

② 木構造計算機は VLSI 技術によって経済的に開発できる．

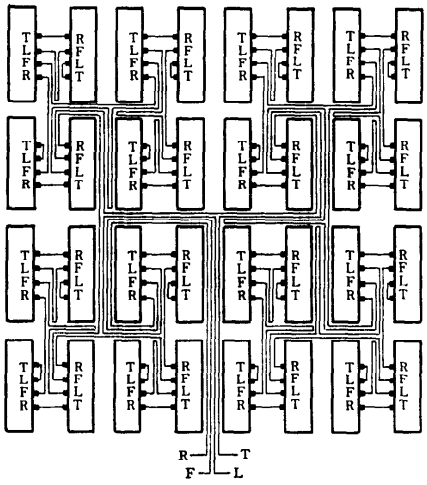
Hyper-H (図-4(a))状のプロセッサ配置を採用するとチップ面積はプロセッサ数に比例する．したがって，集積技術が進めば，それに比例して多くのプロセッサを1チップに埋め込むことができる．また Leiserson[43)] は，任意の大きさの2進木と1個の付加プロセッサを組合わせて1チップとする方法を提案し



(a) Hyper-H



(b) チップの結合



(c) プリント基板

図-4 木構造計算機におけるプロセッサの配置[43]

ている．この方法を採用すると，チップのピン数がチップ内のプロセッサ数に無関係な定数でおさえられるのに加えて，任意の大きさの木構造計算機をこのチップを組合わせてスペース効率良く実現できる（図-4 (b)(c)）．

③ 木構造計算機はプロダクションシステムの階層の表現に適する．

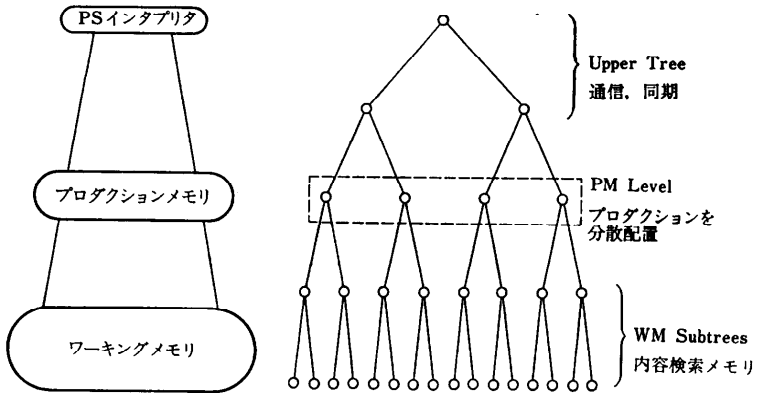
図-5 にプロダクションシステムの構成と DADO の代表的な機能構成を示す．プロダクションシステムにおけるインタプリタ，PM, WM の階層を木構造計算機の階層にマッピングしている．この図から DADO は WM 参照に局所性のある応用に適することが分かる．

（2） MSIMD アーキテクチャ

DADO はプロダクションシステムの並列実行を実現するために，独特の MSIMD (Multiple-SIMD) アーキテクチャを有している．以下にいくつかの木構造計算機を紹介し，DADO の MSIMD アーキテクチャの特徴を説明する．

① Browning の木構造計算機[42]

Browning は MIMD (Multiple-Instruction stream and Multiple-Data stream) アーキテクチャの木構造計算機と，その上での各種アルゴリズムを提案している．各プロセッサは独自のプログラムを内蔵し，隣接するプロセッサとの間で非同期にメッセージの交換を行うことができる．Browning はこの計算機を幅広い用途に適用できる汎用計算機として設計しており，アルゴリズムの記述には CSP[44] をベースにした言語を



(a) プロダクションシステム (b) DADO

図-5 プロダクションシステムの構成と DADO の構成[35]

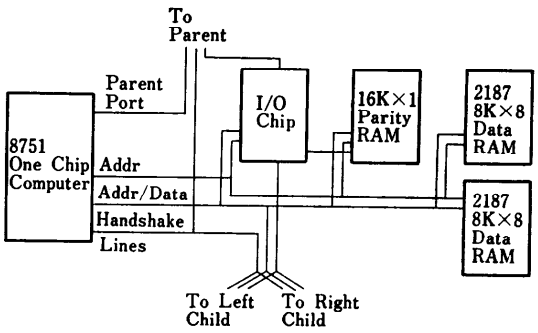


図-6 DADO プロトタイプ[27]

採用している*.

② NONVON[46]

NONVON は多数の小規模プロセッサからなる SIMD (Single-Instruction stream and Multiple-Data stream)アーキテクチャを採用している．NONVON ではコントロールプロセッサ上のプログラムが木全体を制御する，即ちコントロールプロセッサが1サイクルごとに命令をブロードキャストし，各プロセッサがその命令を実行する**.

各プロセッサは 64 Byte の RAM を有し，その機能はデータの比較や通信を中心としたごく少数の命令を実行するだけにおさえられている．NONVON は当初データベースへの応用を主眼として考案されたが，最近では画像認識への応用も検討されている．

* 文献 45) の中には SIMULA をベースにした言語で記述された例を見ることができる．

** 最近の文献 47) では多数の SIMD 木間にネットワークをはり，MSIMD アーキテクチャを実現するアイデアが発表されている．

③ DADO

DADO のアーキテクチャは，Browning の木構造計算機と NONVON の両方の特徴を兼ね備えた MSIMD アーキテクチャである．木全体の初期状態は SIMD 状態である．SIMD 状態では DADO のルートノードは以下の2通りの方法で木を制御する．

ⓐ ルートノードがプログラムをブロードキャストし，全プロセッサがそのプログラムを実行する．

ⓑ ルートノードが木構造計算機制御用の特殊命令を実行することによって，木全体が同期して同一の処理を行う．

また，DADO は必要が生じると，任意の複数の部分木を全体の木から切り離してプログラムを実行させる (MIMD 状態) ことができる．切り離された部分木のルートノードは，上記の2通りの方法で部分木を制御する．

4.2.2 プロトタイプシステム

15 個のプロセッサからなる DADO-1 は 1983 年4月以来稼働しており，実験，評価に用いられている．プロセッサには INTEL 8751 (8 bit プロセッサ，4 kByte EPROM 内蔵) を用い，1プロセッサ当り 8 kByte の RAM を有する．各プロセッサは 3.5 MHz で動作し，プロセッサの総計は 4 MIPS 程度 (計算値) となる．物理的な大きさはパーソナルコンピュータ程度である．DADO-1 では隣接するプロセッサ間の通信しかできないため，データや命令のブロードキャストには大きなオーバヘッドが伴う．

1,023 個のプロセッサからなる DADO-2 は，**図-6** に示すようにいくつかの性能改善が図られている．専用の I/O チップの導入により，8 bit データのブロードキャストは1命令サイクル以内に完了する．1プロセッサ当りのメモリは，論理型言語への応用を考えて 16 kByte に拡張されている．プロセッサはメモリチップを取りかえたために 12 MHz での実行が可能となり，単純に総計すると 570 MIPS となる．しかし，その複雑さは VAX 11/750 と同程度であると報告されている[27].

両プロトタイプは共に，VAX 11/750 にバックエン

ドプロセッサとして接続され，ユーザは VAX を通じて DADO を起動する．VAX 11/750 は DADO の頂点のプロセッサと共に仮想的に DADO のルートノードを構成する．

4.2.3 プログラミング

表-1 に DADO の各プロセッサが持つ木構造計算機制御用の特殊命令の一覧を示す．これらの命令は，木全体のルートノードあるいは切り離された部分木 (MIMD 状態) のルートノードからのみ発行され，木全体あるいは部分木中の全プロセッサで同期して実行される．これらの命令を用いてプログラミングが容易に行えるよう，DADO は以下の2種の木構造計算機用システム記述言語を有している．

(1) PPLM[28]

マイクロプロセッサ用システム記述言語 PL/M 中に DADO の特殊命令を記述可能としたもので，すでに稼働している．言語の設計は ILLIAC-IV の並列処理用言語 GLYPNIR[48] から影響を受けている．

(2) PPSL[29]

ユタ大学で開発された移植容易な Lisp 言語，PSL (Portable Standard Lisp)[49] 中に DADO の特殊命令を記述可能としたもので現在開発中である．

これらの言語で記述されたプログラムは，制御計算機 VAX と DADO の各プロセッサ上にロードされ実行される．8 bit プロセッサ上で Lisp の全仕様をサポートすることは困難であるため，PPSL ではごく基本的な関数と特殊命令だけを DADO プロセッサ用にサポートする予定である．

DADO 上でのプログラミングのイメージを明確にするために，**図-7** に PPLM で書かれた文字列マッチのプログラムを示す．このプログラムの外枠は DADO のルートノード (物理的には VAX 11/750 と DADO の頂点プロセッサ) 上で実行されると考えてよい．プログラム中の "DO SIMD; …END;" で囲まれた部分だけが DADO の全プロセッサにロードされ実行される．SLICE 属性付きで宣言された変数や配列は全プロセッサ上に割付けられる．また，特殊レジスタは宣言なしで用いることができる．このプログラムは以下の手順で実行される．

① まず DADO の全プロセッサを動作可能 (Enable) とする．

② ルートノードから1文字ずつ全プロセッサにブロードキャストし，各プロセッサ上の文字列と比較し，一致しないことが明らかとなったプロセッサを動作不能 (Disable) とする．

③ 動作可能なプロセッサから1個を選択し (Resolve)，選択できれば文字列マッチ成功とする．

PPLM ではこのように木の制御を陽に記述するが，PPSL では特殊レジスタを隠蔽するなど開発効率の良いプログラミング環境の実現を目指している．

表-1 DADO の特殊命令[30]

命　令	機　能
Enable/Disable	配下のプロセッサをすべて動作可能／不能とする．(各プロセッサの特殊レジスタ EN1 を ON/OFF する．) Broadcast 等この表の命令はすべて，Enable 状態のプロセッサのみに影響する．
Broadcast	配下のプロセッサすべてに同一データを送出する．(データは各プロセッサの特殊レジスタ A8 に設定される．)
Resolve	配下のプロセッサのうち，1個のプロセッサを選択する．(各プロセッサを inorder で調べ，特殊レジスタ A1 が ON であるような最初のプロセッサを選択する．他のプロセッサの A1 レジスタはすべてリセットされる．選択に成功した場合には，ルートノードの特殊レジスタ CPRR が ON となる．)
Report	配下のプロセッサからデータを受信する．配下に Enable 状態のプロセッサが2個以上ある場合には，その動作は保証されない．Resolve を実行し1個のプロセッサを選択した後，この機能を用いる．
Send/Receive	隣接するプロセッサ間で，データを送出／受信する．この動作も，配下の全プロセッサで一斉に行われる．
MIMD	配下のプロセッサを切り離し，それぞれの処理を行わせる．(Enable 状態にあるプロセッサをルートとする部分木が切り離される．) 切り離されたプロセッサは，その処理が終了すると再び，以前の結合状態に戻る．

注) これらの命令は，すべて MIMD プロセッサ (木全体のルートノード，あるいは切り離された部分木のルートノード) で発行され，配下の全プロセッサ上で同期して実行される．

4.2.4 プロダクションシステム実行モデル

DADO 上でのプロダクションシステム実行モデルについては文献 30) に概念がまとめられている．代表的なモデルは図-5 に示すように DADO を以下の3階層に分けるものである．

Upper Tree: PM Level のプロセッサ間の通信，同期を行う．

PM Level: プロダクションを分散配置する2進木の特定のレベルである．

WM Subtrees: WM を配置する．内容検索メモリ (content addressable memory) として働く．

```
ASSOCIATIVE-PROBE: PROCEDURE (Search);

  DECLARE Intelligent-Record (64) BYTE SLICE; /*An instance*/
  DECLARE Index BYTE SLICE;                   /*of each appears in every PE.*/
/*We assume each of the instances of Intelligent-Record
  have been previously loaded within the DADO tree.*/

  DECLARE i BYTE;            /* i is local to this routine.*/
  DECLARE Search (64) BYTE; /*The search string is provided by some external source.*/

  DO SIMD;
    Call ENABLE; /*All descendent PE's enter SIMD enabled state.*/
    A1=0;        /*All A1 flags within the tree below are cleared.*/
  END;
  DO i=1 to length (Search); /*Repeat the following for
                               each character of the search string.*/
    DO SIMD;
      Call BROADCAST (i); /*The value of the index variable i is broadcast and loaded in each
                            SLICEd A8 variable within the tree,      */
      Index=A8;           /*and transfered to a local SLICEd variable.*/
      Call BROADCAST (Search (i)); /*The i th character of the search string is then broadcast
                                     and loaded in each A8 register.*/
      EN1=A8=Intelligent-Record (Index);
                          /*After comparing the search character, currently
                            stored in A8, with the locally stored data, disable
                            those PE's which do not match (by assignment of
                            logical 0 to the enable variable EN1).      */
    END;
  END;
  DO SIMD;
    A1=1;          /*Only those PE's that remain enabled, that is only those which match the
                     search string, will set their A1 variables high.*/
    Call RESOLVE; /*Lastly, we test for whether or not any A1 flags in the tree are high.
                    CPRR is set accordingly.*/
  END;
  IF CPRR THEN /*we have responders!*/;
END ASSOCIATIVE-PROBE;
```

図-7 PPLM を用いたプログラム例[30]

PM レベル以下の部分木は必要に応じて全体の木から切離され (MIMD 状態), 並列に Match フェーズや Act フェーズを実行する. PM レベルをどの階層にとるべきかはプロダクションシステムの特性によって変ると考えられている. DADO-2 では当面第6レベル (32プロセッサ) を PM レベルとし, その配下にそれぞれ30プロセッサからなる部分木を配置することが考えられている. このモデル上での並列処理には次のものがある.

(1) プロダクションレベルの並列処理

並列マッチとしては, ①各プロダクションサイクルごとに毎回改めてマッチングを実行するモデル[25], ②RETE ネットワークを各 WM Subtree 上に構築するモデル[30], ③RETE ネットワークを並列処理に適するように改良して各 WM Subtree 上に構築するモデル[31]が考えられている. 並列マッチによる性能向上は1桁程度であると報告されている.

並列発火はシミュレーションにより最大13程度の並列性が確認されており, プロダクションサイクルを1/7 程度に削減することが期待されている. 並列度を上げるためのプロダクションシステムの分割方法は文献32) で議論されている.

(2) エレメントレベルの並列処理

DADO では WM Subtree 上で以下のようなエレメントレベルの並列処理を行うことによって, 大量の WME を有するプロダクションシステムの実行をさらに高速化することを計画している.

① OPS の言語仕様を拡張して LHS で WME の集合を検索し, RHS でその集合に対して変更が加えられるようにする.

② RHS の動作に木構造計算機が威力を発揮する演算を加える. 例えば, WME の特定の属性値の総

和を求める演算等がこれにあたる.

4.3 CMU の PSM プロジェクト

CMU のPSM プロジェクトでは OPS の開発者である Forgy を中心に,プロダクションシステム用計算機の研究を進めている. ILLIAC-IV 上でのプロダクションシステム並列実行の検討[35] や DADO 上での検討[36], 及び 3.2 節で述べたプロダクションシステムの特性測定の結果, DADO とは異なるアーキテクチャの採用を検討している. その概要は,プロダクションシステムの並列処理の効果は5〜10倍であるとし,計算機アーキテクチャとしては比較的少数の単純で高速なプロセッサ (例えば ECL 等の高速素子を用いた RISC プロセッサ[50]) を組合わせたものが適するとしている[38].

PSM プロジェクトでは, この計算機上でプロダクションレベルの並列マッチとコンディションレベル, アクションレベルの並列処理を計画している. 並列マッチに適したプロダクションシステムの分割方式は文献 39) で議論されている. また, 総合的な処理速度の向上は, 並列処理による効果 (5〜10倍), RISC プロセッサを用いることによる効果 (2〜4倍), ECL を用いることによる効果 (4倍) を総計して 40〜160 倍程度が期待できるとしている.

5. むすび

プロダクションシステムはその単純で強力な機能により, 今日のエキスパートシステム開発の主要なツールとなっている. すでに 3,000 ルール以上のものが実用に供されているが, 今後さらにルール数の増大が予想されており, 高速な並列実行環境が必要とされつつある. 本解説で紹介した木構造計算機 DADO 及び PSM プロジェクトの提案は, プロダクションシステムの並列処理の実現を目指しており, そこから得られる経験は今後の関連研究に共通の基礎を与えるものと期待される. しかし並列計算機の構成にはさまざまな提案があり, もとより現時点で勝者が決っているわけではない. 今後とも, プロダクションシステムの応用の広がりと, 並列計算機の動向に注視していくことが必要である.

参考文献

1) Davis, R. and Jonathan, K.: An Overview of Production Systems, Machine Intelligence, Vol. 8, pp. 300-332, J. Wiley and Sons, New York (1977).
2) Nilsson, N. J.: Principles of Artificial Intelligence, Tioga, Palo Alto, Calif. (1980).
3) 辻井潤一: プロダクションシステムとその応用, 情報処理, Vol. 20, No. 8, pp. 735-743 (1978).
4) 斎藤正男, 溝口文雄: 知的情報処理の設計, コロナ社 (1982).
5) Feigenbaum, E. A., Buchanan, B. G. and Lederberg, J.: On Generality and Problem Solving: A Case Study Using DENDRAL Program, Machine Intelligence, Vol. 6 (1971).
6) Shortliffe, E.: Computer-Based Medical Consultations: MYCIN, American Elsevier, New York (1976).
7) Lenat, D. B. and Brown, J. S.: Why AM and EURISKO Appear to Work, Artificial Intelligence, Vol. 23, pp. 269-294 (1984).
8) Lesser, V. R. and Erman, L. D.: Distributed Interpretation: A Model and Experiment, IEEE Trans. on Comput., Vol. C-29, No. 12, pp. 1144-1163 (1980).
9) VanMelle, W., Scott, A. C., Bennett, J. S. and Peairs, M.: The Emycin Manual, Technical Report STAN-CS-81-885, Department of Computer Science, Stanford University (1981).
10) Weiss, S. M., Kern, K.B., Kulikowski, C. A. and Uschold, M.: A Guide to the Use of the EXPERT Consultation System, Technical Report CBM-TR-94, Department of Computer Science, Rutgers University (1984).
11) Rychener, M. and Newell, A.: An Instructable Production System: Basic Design Issues, in Pattern-Directed Inference Systems, Academic Press (1978).
12) Forgy, C. L. and McDermott, J.: OPS, A Domain-Independent Production System Language, IJCAI 5, pp. 933-939 (1977).
13) Forgy, C. L.: On the Efficient Implementation of Production Systems, Technical Report, Department of Computer Science, Carnegie Mellon University, Ph. D. Thesis (1979).
14) Forgy, C. L.: Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem, Artificial Intelligence, Vol. 19, pp. 17-37 (Sep. 1982).
15) Forgy, C. L.: OPS 4 User's Manual, Technical Report CS-79-132, Department of Computer Science, Carnegie Mellon University (1979).
16) Forgy, C. L.: OPS5 User's Manual, Technical Report CS-81-135, Department of Computer Science, Carnegie Mellon University (1981).
17) Forgy, C. L.: OPS 83 Reference Manual,

Technical Report, Department of Computer Science, Carnegie Mellon University (1984).
18) Vesonder, G. T., Stolfo S. J., Zielinski, J. E., Miller, F. D. and Copp, D. H.: ACE: An Expert System for Telephone Cable Maintenance, Proc. of IJCAI-83, pp. 116-121 (1983).
19) McDermott, J.: R 1: A Rule Based Configurer of Computer Systems, Artificial Intelligence, Vol. 19, pp. 39-88 (1982).
20) McDermott, J.: R 1: The Formative Years, AI Magazine, Vol. 2 (1981).
21) Bachant, J. and McDermott, J.: R1 Revisited: Four Years in the Trenches, AI Magazine, Vol. 3 (1984).
22) McDermott, J.: XSEL: A Computer Sales Person's Assistant, Machine Intelligence, Vol. 10, pp. 325-337, J. Wiley and Sons, New York (1982).
23) Griesmer, J. H., Hong, S. J., Karnaugh, M., Kastner, J. K., Schor, M. I., Ennis, R. L., Klein, D. A., Milliken, K. R. and VanWoerkom, H. M.: YES/MVS: A Continuous Real Time Expert System, Proc. of the National Conference of Artificial Intelligence (1984).
24) Kowalski, T. and Thomas, D.: The VLSI Design Automation Assistant: Prototype System, Proceedings of the 20th Design Automation Conference (June 1983).
25) Stolfo, S. J. and Shaw, D. E.: DADO: A Tree Structured Machine Architecture for Production Systems, Proc. of the National Conference of Artificial Intelligence (1982).
26) Stolfo, S. J., Miranker, D.P. and Shaw, D. E.: Architecture and Applications of DADO: A Large-Scale Parallel Computer for Artificial Intelligence, Proc. of IJCAI-83 (1983).
27) Stolfo, S. J. and Miranker, D. P.: DADO: A Parallel Processor for Expert Systems, Proc. of International Conference on Parallel Processing (1984).
28) Stolfo, S. J., Miranker, D. P. and Lerner, M.D. :PPL/M : The System Level Language for Programming the DADO Machine, Technical Report, Department of Computer Science, Columbia University (1984).
29) VanBiema, M., Lerner, M. D., Maguire Jr., G. Q. and Stolfo, S. J.: //PSL: A Parallel Lisp for the DADO Machine, Technical Report, Department of Computer Science, Columbia University (1984).
30) Stolfo, S. J.: Five Parallel Algorithms for Production System Execution on the DADO Machine., Proc. of the National Conference of Artificial Intelligence (1984).
31) Miranker, D. P.: Performance Estimates for the DADO Machine: A Comparison of TREAT and RETE, Proc. of FGCS-84 (1984).
32) Ishida, T. and Stolfo, S. J.: Towards the Parallel Execution of Rules in Production System Programs, Technical Report, Department of Computer Science, Columbia University (1984).
33) Taylor, S., Lowry, A., Maguire Jr., G. Q. and Stolfo, S. J.: Logic Programming Using Parallel Associative Operations, Proc. of International Symposium on Logic Programming (1984).
34) Lowry, A., Taylor, S. and Stolfo, S. J.: LPS Algorithms, Proc. of FGCS-84 (1984).
35) Forgy, C. L.: Note on Production Systems and Illiac IV, Technical Report CS-80-130, Carnegie Mellon University (1980).
36) Gupta, A.: Implementing OPS 5 Production System on DADO, Proc. of International Conference on Parallel Processing (1984).
37) Gupta, A. and Forgy, C. L.: A Measurements on Production Systems, Technical Report, Carnegie Mellon University (1984).
38) Forgy, C. L., Gupta, A., Newell, A. and Wedig, R.: Initial Assessment of Architectures for Production Systems, Proc. of the National Conference of Artificial Intelligence (1984).
39) Oflazer, K.: Partitioning in Parallel Processing of Production Systems, Proc. of International Conference on Parallel Processing (1984).
40) Schwartz, J.: A Taxonomic Table of Parallel Computers Based on 55 Designs, Courant Institute, New York University (Nov. 1983).
41) Boley, H.: A Preliminary Survey of Artificial Intelligence Machine, SIGART 72, pp. 21-28.
42) Browning, S.: The Tree Machine: A Highly Concurrent Computing Environment, Ph. D. Thesis and Technical Report #3760, California Institute of Technology (Jan. 1980).
43) Leiserson, C. E.: Area-Efficient VLSI Computation, Ph. D. Thesis, Department of Computer Science, Carnegie Mellon University (Oct. 1981).
44) Hoare, C. A. R.: Communicating Sequential Processes, CACM, Vol. 21, No. 8, pp. 669-677 (1978).
45) Mead, C. and Conway, L.: Introduction to VLSI Systems, Addison-Wesley (1980).
46) Shaw, D. E.: The NON-VON Supercomputer, Technical Report, Department of Computer

Science, Columbia University (Aug. 1982).
47) Shaw, D. E.: SIMD and MSIMD variants of the NON-VON Supercomputer, COMPCON-84 (1984).
48) Lowrie, D. D., Layman, T., Daer, D. and Randal, J. M.: GLYPNIR—A Programming Language for ILLIAC Ⅳ, CACM, Vol. 18, No. 3 (1975).
49) Galway, W., Griss, M. L., Morrison, B. and Othmer, B.: The Portable Standard LISP User's Manual, University of Utah (1983).
50) Patterson, D. A. and Sequin, C. H.: A VLSI RISC, Computer, Vol. 9 (1982).

(昭和60年1月22日受付)