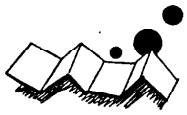
解　説

3. アーキテクチャ



3.2 記号処理マシン†

柴山　潔††

1. はじめに

記号処理分野においても，近年は専用マシンの開発が盛んである．特に最近ではマシンの対象言語やその処理方式・機構になんらかの形で並列性を導入しているものが多い．

本学会でも 1982 年8月に「記号処理と計算機アーキテクチャ」という特集[9]を組んでいる．この特集号では，調査対象の記号処理マシンとして Lisp マシンのみが取り上げられていた．また当時は，リスト処理専用言語として Lisp をとらえ，その専用化を図るというアプローチが主流であった．したがって，言語そのものに並列性を導入したり，記号処理分野そのものを並列処理可能性の点から見直したりするアプローチはあまり盛んではなかった．しかし近年は高機能のマイクロプロセッサ・チップや高集積化された論理チップあるいは大容量メモリ・チップなどが手軽に利用可能となり，人工知能などの応用分野からもトップ・ダウン的に専用ハードウェア，たとえばプロダクション・システム向き並列マシン[60]，の開発を研究目標とするような傾向も見受けられるようになってきた．

また，Lisp マシン以外の記号処理マシンとしても，ICOT で開発している第5世代コンピュータが Prolog 系言語を対象としたことなどに刺激されて，Prolog マシンの開発が盛んになっている．

さらに，最近の記号処理マシン開発では Prolog マシンあるいは Lisp マシンの並列化などが話題となっており，本解説では前の特集以降の記号処理マシンの開発動向について，特に並列処理方式の観点から論述する．

本学会の特集[9]以降にも記号処理マシンに関する調査，解説記事は数多く出版されている．たとえば，論理型言語向きマシンに関しては文献 7)，10)，15)があり，関数型言語向きマシンに関しては文献 12)～14)がある．また，これら記号処理全般に関するマシンの解説を文献 6)，8)が行っている．以上の文献のうち，特に並列マシンに関して言及しているものは，文献 6)，7)，15)などである．

本解説では，最近4～5年間に発表あるいは開発された記号処理マシンを調査対象として取り上げるが，特に重要なマシンを除いて，できるかぎり前述の文献との記事の重複を避けている．また，記号処理は近年新しい計算機応用対象として進展してきた分野であるので，提案されている記号処理マシンも数多い．本文で解説する記号処理マシンは，それらの一部である．調査の際に参考にした文献も膨大なものになったので，本解説の末尾に添えた文献リストもこれらのうちのごく一部しか掲げることができなかったことをあらかじめ断っておく．

本特集号においては，記号処理に関するアルゴリズムや一般的な並列マシンの開発状況の解説も別の章で行われる予定であり，本解説では記号処理マシンの種種のアーキテクチャのみに焦点を絞り，それぞれの並列性を実現するために構築した処理方式上の特徴について述べる．記号処理マシンに関する理論的な側面(たとえば，処理アルゴリズム)や実行モデルの定量的評価のみを示している論文については，本解説の対象とはしていないので，これらについては他の解説記事で補ってほしい．

また，本解説の執筆にあたっては，対象としたマシンの設計思想を浮き彫りにするために，公表論文以外にも，できるかぎり開発者自身のコメントも参考資料とするように努めた．しかし，本特集号の編集上の都合や筆者の考え違いなどにより，調査対象からはずれたり，他の解説と重複したり，開発者の考えとは異なる分類をされたマシンがあるかもしれないことも，あらかじめ断っておく．

† Symbolic Manipulation Machine by Kiyoshi SHIBAYAMA (Department of Information Science, Faculty of Engineering, Kyoto University).

†† 京都大学工学部情報工学教室

第2章では，本解説が対象とする「記号処理」分野について定義し，最近数年間における記号処理マシンの開発動向について，特に並列マシンの出現を軸に概説する．第3章では，記号処理における並列性について述べ，その処理方式について言及する．また第4章では，各種の並列型記号処理マシンに実例をとり，その処理方式の特徴について比較する．

2. 記号処理マシンの開発動向

計算機の応用分野が広がるにつれ，それら全般を汎用マシンのみでまかなうのが困難となり，種々の専用マシンが登場してきた．特に記号処理マシンの応用分野は最近盛んになっている人工知能や知識情報処理との関連も深く，マシン・アーキテクチャとしても既成の概念を打ち破るもの[8]が求められている．

計算機は一般に汎用と専用とに大別され，さらに専用マシンは数値処理（科学技術計算）向きと非数値処理向きとに分けることができる．前者については本特集号の他の章にまとめられている．後者の非数値処理向きマシンには，記号処理マシンを始めとして，データベース・マシンや画像処理マシンなどが含まれる．しかし，マシンの分類方法は人によってまちまちであり，専用マシンを明確に分けることはなかなか難しい．たとえば，画像処理マシンなどは数値処理向きマシンとして分類されている場合もある．

もともと「記号処理」という言葉に明確な定義はないので，その指すところは時とともに微妙に変化している．たとえば最近では Prolog をプログラミング言語として採用した計算機の応用は記号処理の一大分野として認められている．

また，記号処理の意味を非数値処理の代表例として広くとらえるとデータベース処理なども含まれるのであろうが，本特集号ではデータベース・マシンについては，他の章で解説される予定であり，本解説で取り上げる記号処理マシンには含めない．

前回の特集号[9]では，記号処理における基本処理として，(1)リスト処理と，(2)文字列処理とを取り上げている．(2)の文字列処理は，サーチやソートなどのデータベース操作と合わせてデータベース・マシンの基本処理と考えられる．したがって，前述の理由により本解説の対象からは除外する．しかし，文字列処理をテキスト処理，それも直接実行型高級言語マシンの基本処理ととらえることも可能である．高級言語マシンとは，応用ではなく，方式による計算機の分類によるものであるので，本解説で取り上げる記号処理マシンにも高級言語マシンに当てはまるものが多い．しかし，その場合には，Lisp や Prolog などの記号処理用高級言語を対象言語とし，それで書かれたプログラムをコンパイラなどにより一旦なんらかの中間言語プログラムに変換した後実行する間接実行型高級言語マシン（いわゆる Lisp マシンや Prolog マシンなど）に調査対象を限っている．したがって，直接実行型高級言語マシンとして文字列処理を主な応用対象としているものや，記号処理用言語（Lisp や Prolog など）を応用対象に含んでいない高級言語マシンは本解説では取り上げていない．

(1)のリスト処理は，記号処理における本質的な役割を担っているとの認識はほぼ定着しており，本文では主にこのリスト処理向きマシンに対象を絞って解説を加える．リスト処理向きマシンとは，その名前どおりリスト処理を専用に行うマシンであるが，応用対象は非数値処理（ないしは記号処理）分野のほぼ全般にわたり，非数値処理分野における汎用マシンと位置付けてもよい．

また，記号処理マシンにおけるリスト処理を機能としてみると，ハードウェアに近い（低い）レベルの概念であり，記号処理分野における応用指向マシンの大半は，これを基本機能としている．

特に最近は，処理対象がリスト処理そのもののみであるマシンはほとんど見られない．その理由としては，リスト処理そのものが記号処理における基本機能として記号処理マシンに必須のものになっていること，および記号処理分野の広がりとともにもっとマシンの応用対象を絞り込まなければ専用化の特長が活用できなくなっていることなどがあげられる．

したがって，図形処理などのように基本機能のみを抽出すれば専用マシンの基本部分が設計できる場合とは異なり，記号処理マシンの構築ではマシンの処理対象を特定の記号処理応用に向いた言語として明示した高級言語マシンとして設計されているものが多い．たとえば，最近利用されている記号処理用言語としては，論理型言語，関数型言語，オブジェクト指向言語などが主要なものであろう．したがって，(A)Prolog マシンなどの論理型言語向きマシン，(B)Lisp マシンなどの関数型言語向きマシン，(C)Smalltalk-80 マシンなどのオブジェクト指向マシン，などが記号処理マシンに分類される．

これらのマシンは，その明示的な対象を特定の言語

においており，言語のセマンティクスをアーキテクチャとして実現さえすれば，その言語で記述可能な問題分野をカバーする専用マシンを比較的簡単に手に入れることができる．また，これらの言語が対象とする応用は，論理的あるいは物理的になんらかの形でリスト処理機能を必要とし，現時点における記号処理の中心的分野を形成している．本解説が対象とする記号処理マシンとは，主としてこの(A)(B)(C)型のいずれかに当てはまるマシンである．

数年前までは，記号処理マシンといえば Lisp マシン[9]が代表であったが，応用の広がりとともにこの専用マシンも多様化し，Prolog マシンを始めとするさまざまな記号処理マシンが開発されつつある．また，歴史の比較的古い Lisp マシンの技術は，商用化という洗礼を受けて洗練され，安定してきている．Lisp マシンに関する最近の話題は，対象言語である Lisp そのものへの並列性の導入や高機能化などである．

一般的に，これらの記号処理マシンは，(B)，(A)(C)の順でアーキテクチャの抽象度がより高くなっていくものと考えられる．したがって，Lisp マシンを用いて Prolog や Smalltalk-80 のシステムを構築することは普通に行われるが，その逆の例はあまり見られない．

対象言語の高機能化は Lisp のみにとどまらず，他の言語でも盛んであり，前述した(A)(B)(C)型のマシンの対象言語の各機能を融合した言語ないしは応用向きの記号処理マシン（これを本解説では(D)型と呼ぶ）も現れてきている．これは，専用機能の見直しによる汎用マシンへの回帰現象ともみられ，人工知能(AI)分野における専用マシンへの期待がこの現象に拍車をかけ，AI マシン[6]という新しい概念も現実的になりつつある．AI マシンの開発は，応用分野からのトップダウン・アプローチが主であり，記号処理マシンというよりは汎用マシンとしての性格が強い．

また，(A)～(D)型のマシンに共通の最近の話題としては，マシン・アーキテクチャへの並列性の導入である．その方式としては，i)対象言語の機能として並列性を導入する（並列型 Lisp や並列型 Prolog[11]などを対象とするマシンへのトップダウン・アプローチ），ii)処理対象における明示的な並列性の有無にかかわらずマシンの一般的な処理機構として並列性を導入する（ボトムアップ・アプローチ），の2種に大別される．対象言語が明示的な並列性を備えているのに，マシンの処理機能になんら並列性がないというのはオーバヘッドの大きい処理方式であるので得策ではない．しかし，対象言語は並列性を持たないが，マシンそのものに並列処理機能が備わっているものは多々存在する．たとえば，一部の逐次型 Prolog マシンなどであり，これらについては第4.6節で概説する．

次章以降の解説で実例としている記号処理マシンは，対象言語に論理型言語，関数型言語，オブジェクト指向言語のいずれかのセマンティクスを含んでいるものである．

3. 記号処理における並列性

3.1 並列マシンの単位処理機能

一般的に並列マシンにおける並列処理単位の機能の大きさ（これを本解説では「粒度」=granularity と呼ぶ）は，単位処理機能間の通信によるオーバヘッドと，負荷分散の度合いとのトレードオフにより決定される．すなわち，細かい粒度 (fine grain) の単位処理機能は小さいので，適用する問題によっては高い並列性を抽出することが期待される反面，単位処理機能間の通信や負荷の切り替え回数が多くなりオーバヘッドとなる恐れがでてくる．逆に，粗い粒度 (coarse grain) の単位処理機能は大きいので，単位処理機能間の通信によるオーバヘッドは単位処理機能により隠されてしまう反面，高い並列性を抽出することはさほど期待できない．

粒度には，応用や問題そのものが本質的に含んでいる並列性の単位機能（これは「論理的粒度」といえる）と，それをマシン・アーキテクチャ上で実現する場合に示されるハードウェアの処理要素（プロセッサ）としての単位機能（これは「物理的粒度」といえる）とがある．本解説では，誤解の生じる恐れのないかぎり，論理的粒度を単に「粒度」と呼び，記号処理における種々の粒度の実現方法について述べる．

実際に並列マシンの物理的粒度を決めるのは，処理機能単位の個数や通信ネットワークの性能である．粒度が細かくなるにつれ，単位処理機能を特定の応用向きに専用化することが困難になり，また通信機能も簡単にする必要がある．この場合には，マシン全体としては汎用性を強く帯びた性格のものにならざるを得ない．すなわち，マシン・アーキテクチャに独立したプログラミングや動的な負荷割り付けが可能となり，拡張性や柔軟性という点では優れている．これに対して，粒度が粗くなるにつれ，単位処理機能を特定の応用向けにチューン・アップすることが柔軟にできるよ

うになり，高度に専用化したマシン・アーキテクチャを構成することが可能となる．

細かい粒度を持つマシンを高並列マシンと呼び，最近は AI 分野における汎用マシンとして注目されている．これについては，第**4.4**節で述べる．

また，粗い粒度を持つマシンの例としては，既存のマルチプロセッサ・システムが当てはまる．これについては，第**4.5**節で述べる．

記号処理マシンにおいても，論理型言語向きマシンや関数型言語向きマシンなどのように対象言語や応用を絞りこんだ(A)～(C)型の並列マシンの粒度は比較的粗く，複数言語を対象とする(D)型のマシンなどの粒度は比較的細かい，という傾向がある．

3.2 記号処理における並列性

数値処理に比較して，非数値データを扱う記号処理における特徴とは，(1)取り扱うデータ単位が可変長の整数や固定小数点数であり，それらに対する操作も論理（マスク）演算や算術加減算が主であり，浮動小数点数や算術乗除算を扱うことはあまりない，(2)対象とするデータの構造は不規則で（連続性がなく），複雑である，(3)データ構造は動的可変性が強く，その制御に要する負荷は取り扱うデータそのものに動的に左右される，などである．

もちろん，上述した(1)～(3)が当てはまらない例外もあるが，おおむね(1)～(3)を記号処理における特徴と考えてよい．したがって，記号処理向きのアーキテクチャとして特徴を出すためには，(1)～(3)の各々に配慮してマシンを設計すればよい．たとえば，(1)に対応してビットやフィールド演算機構の強化，(2)に対応してメモリ管理機構の専用化および高機能化（構造化），(3)に対応してタグ・アーキテクチャの採用，などが記号処理マシンの構築における一般的手段である．

これらの記号処理において，明示的な並列性はほとんどない．むしろ，リストのたどりなどは逐次的な処理である．一次元アドレス空間に展開された線形リストを高速にたどる機構としては，リスト・アクセスのパイプライン化などの方式があるが，本質的に高並列化は不可能である．この処理を高速化するには，連想メモリなどを用いてデータ構造のアドレッシングやアクセス機構の構造化を図る方が効果的である．

逐次型の記号処理マシンにおいて核となるものはリスト処理であるが，これを高速に行うためにリスト格納用メモリを構造化する手法も盛んに研究されてきた．特に，ゴミ集め（ガーベジ・コレクション＝GC）はリスト演算と並列に行い得る機能単位として注目され，リアルタイムGC法[72]などの採用が，今までの逐次型記号処理マシンにおける唯一の明示的な並列性であった．

また，言語のセマンティクスに並列性を備えていない（逐次型の）Lisp や Prolog, Smalltalk-80 を処理対象とする記号処理マシンにおいては，単一の物理的粒度（プロセッサ）内での低レベル並列処理方式を採用している場合が多い．この方式の記号処理マシンについては，第**4.6**節で述べる．

むしろ，最近の論理型言語向きマシンや関数型言語向きマシンでは，それが対象とする言語のセマンティクスに並列性を積極的に導入する試みが盛んである．すなわち，並列論理型言語[11](Concurrent Prolog[48], PARLOG[76], DELTA-PROLOG[75], GHC[52] など)や並列関数型言語 (Concurrent Lisp[71], Multilisp[33], CmLisp[96], Queue-based Lisp[110], など)，並列オブジェクト指向言語[2](ABCL/1[88], Orient84/K[85], Dinnerbell[81], Concurrent Smalltalk[79] など)を処理対象として，言語の機能レベルに合わせたマシン・アーキテクチャを構築すれば，それが自然に並列マシンとなっているという仕組みである．これらは並列型言語の処理に向いた高級言語マシンとみなすことができる．これらについては，各言語ごとに第**4.1～4.3**節で述べる．

これらの並列型言語では，問題の持つ並列性を自然に引き出すことが可能となり，逐次型言語に特有の処理（たとえば，Prolog におけるバックトラック処理）は不要となる．その代わり，並列実行のための制御機能が余分に必要となり，粒度の設定いかんによっては，通信オーバヘッドのために並列性が殺されてしまうこともある．むしろ，対象言語が逐次型から並列型に変わっても，粒度にはほとんど影響がないことに留意すべきである．マルチ Prolog マシンやマルチ Lisp マシンは，この性質を全面的に利用したマシン構築法と言える．

リストをたどるという処理はもともと単一のリスト構造用メモリを粒度の粗い（ないしは単一の）プロセッサ（単位処理機能）で操作する発想の上に成り立つものであり，最近ではこれを根本的に覆えすマシン・アーキテクチャが現れてきている．すなわち，粒度をリストの要素に対する演算程度に細かくして，リストのたどりは単位処理機能間の通信としてとらえるマシンである．これらについては，第**4.4**節で概説する．

4. 並列型記号処理マシンの実例とその処理方式

4.1 論理型言語向き並列マシン

4.1.1 論理型言語向きマシンにおける並列性

論理型言語である Prolog は逐次性を主とするセマンティクスを備えており，これを処理対象とするマシンでは単一プロセッサ内にパイプラインや低レベル並列処理方式を導入して高速化を図っている．最近では論理型言語のセマンティクス自体に並列性を取り入れた言語が提案され，これらを処理対象とする並列マシンの開発が盛んになってきた．

Prolog のセマンティクスから逐次性（節の単一化順序，ゴール列の評価順序，およびこれらの逐次性にともなう副作用など）を除いたセマンティクスを持つものを「純粋 Prolog」と言う．純粋 Prolog は全解探索用言語とみなされるが，実用的には並列性を野放しにするのではなく，なんらかの制限を加えることのできるセマンティクスが必要となる．現在提案されている並列論理型言語ではこの並列性をセマンティクスとして制限したり，プログラムに明示したりすることのできる機能を備えているものが多い．純粋 Prolog は次に掲げるような並列性を内包している．

OR並列性……定義体（頭部の述語記号が同じであるような節の集合）中の節間には論理的なOR関係がある．OR並列性とは，ある述語呼び出しに対し，OR関係にある各節の頭部との単一化およびそれに引き続く本体ゴール列の実行を並列に行うことができることを指す．Prolog（逐次型）のセマンティクスではOR並列性は逐次処理機能のもとで完全に制限されている．逆に，純粋 Prolog はOR並列性をまったく制限していないので，探索木が爆発的に増大する恐れがある．Concurrent Prolog (CP)[48] や GHC[52]などでは，OR並列性を単一化の際に利用しているが，それに引き続く本体の実行を1個のゴール列に絞るという制限を加えるセマンティクス（「ガード機構」と呼ぶ）を採用している．また，プログラムでOR並列性の実現を明示させる方式を採っている言語もある．さらに，アーキテクチャそのものにOR並列性を制限する機構を装備したマシンも存在する．

AND 並列性……節本体のゴール列間には論理的な AND 関係がある．AND 並列性とは，本体の各ゴールの実行を並列に行うことができることを指す．AND 並列性の実現においては，ゴール間で共有される変数（共有変数）が存在する場合に，共有変数の値に矛盾がないかを調べる無矛盾性検査 (consistency check) の機能を付加する必要がある．Prolog（逐次型）のように AND 並列性を逐次処理する場合には，この共有変数の存在は問題とならないが，純粋 Prolog や並列論理型言語のようにセマンティクスとして AND 並列性を含んだり，プログラムで明示指定できる場合には，この問題の処理方式が性能を左右する要因となる．したがって，AND 並列性の処理においては，純粋に AND 並列処理を行った後に無矛盾性検査を行う方式と，ゴール列の評価をストリームとしてとらえ，そこに生成された解を順々に流すことによりパイプライン効果を得る（ストリームは逐次的なので無矛盾性検査は不要である）ストリーム並列処理方式とが考えられる．

引数間並列性……複数の引数を持つ節頭部や複合項の単一化を行う際に各引数の単一化を並列に行うことを指す．引数間並列性の実現においても，引数間に共有変数が存在する場合には，AND 並列性の実現と同様に無矛盾性検査の必要性が生じる．

以上の各並列性は論理型言語に特有のものであり，これらを活用した処理機構を実現しようという試みが論理型言語向き並列マシンである．また，単一化において，事実（データベース）を探索する場合などには，一般的な言語処理における並列性と同様な並列性（サーチ並列性）が存在する．

4.1.2 論理型言語向き並列マシンの実例

表-1 で，種々の論理型言語向き並列マシンの特徴を比較する．論理型言語向き並列マシンは命令駆動方式の違いにより次のような3種類に分けられる．

①コントロール駆動方式では，複数の逐次マシンの並列実行により処理を進めるマルチプロセッサ方式が主となる．コントロール駆動方式論理型言語向き並列マシンには，株分け方式[27]，Multi-PSI[47]，Or-parallel token machine[84]，などがある．

②データ駆動方式では，論理型言語プログラムを一旦データフロー・グラフに変換し，それをデータフロー・マシン[1]上で実行する．論理型言語向きデータフロー・マシンの特徴としては，データフローの方向が動的に決まることやデータフローが非決定性を持っていることがある．データ駆動方式論理型言語向き並列マシンには，プロセスグラフ・モデル[18]，PIM-D[20]，ORBIT[59]，千葉大学のデータフロー・マシン[73]，Sains Malaysia 大学のデータ駆動マシン[19]，などが

表-1 論理型言語向き並列マシン

マシン名	株分け方式	Multi-PSI	Or-parallel token machine	プロセスグラフ・モデル	PIM-D	ORBIT	(未定)	PIE-II	PIM-R	KPR
開発機関	富士通 および ICOT	ICOT	Swedish Institute of Computer Science(SICS)	電子技術総合研究所	ICOT および 沖電気工業	沖電気工業	千葉大学	東京大学	ICOT および 日立製作所	京都大学
現況	・16台PE構成のハードウェア・シミュレータ完成	・PE (PSI) 稼動中 ・1986.7-6 台PE構成稼動 (最大64台 PE 想定)	・1983 発表 ・メモリ構成のシミュレーション中 ・マルチプロセッサ・システム上にインプリメント予定	・実行モデルの提案 ・Dialog. H 上にインプリメント予定	・ハードウェア・シミュレータ (4台PE, 4台以上のPE想定) 稼動中 ・16台PE, 15台メモリまで拡張予定	・1983実行モデルの提案(100台分集合) 方式に	・方式の提案	・1台の推論ユニット (単一化プロセッサ, 3個のMC68000)稼動中 ・構造メモリ試作中	・ハードウェア・シミュレータ (MC68000・2MBメモリ付き16台) 稼動中 ・各メモリ増設 (8MB) 中 ・100 台PE構成想定	・プロトタイプ (5台PE)設計中 ・1000台PE構成を想定
対象記号処理用言語	Prolog (純粋 Prolog+cut 機能)	KL1(GHC)	Prolog, 論理型言語	論理型言語	KL1 (GHC), OR並列型言語	逐次型 Prolog	純粋 Prolog	論理型言語	Prolog, CP, GHC	並列論理型言語 (純粋 Prolog+並列型Prolog の基本機能)
応用分野		並列推論ソフトウェア (言語, OS, 応用) の開発	記号処理, 数値処理 (ベクトル/行列演算)						自然言語処理, レイアウト CAD, 論理DA	論理プログラミング
論理的粒度	インタプリタ (各PEがなるべく大きなプロセス部分木を担当)	ゴール	項リスト間の単一化よりも大きい機能, トークン (プロセスの状態を示しマシン命令列にリンクする)	データフロー・グラフ上を流れるリテラル	データフロー・グラフ (単一化や非決定性を制御する基本ノード) のインタプリタ	可変長トークン (データフロー・グラフ), 単一化	ゴール (リテラルの引数や結果)	推論 (ゴール・フレームの書き換え)	リテラル(ゴール)	ゴール(プロセス,) デマンドやイベント, PE内の単一化
物理的粒度	68010 (10 MHz, 2MBメモリ)	PSI (逐次型 Prolog マシン)	Balance 8000 (汎用マルチプロセッサ・システム) 上の NS32032	MC68000 や MC68010(単一化/コピー/マージ・ユニット)	PE (基本演算ユニット2個, パケット・キュー, 命令制御, パイプライン構造), 構造メモリ	データフロー型 PE, ファイル・サーバ	PE (単一化, 共有変数の置換, トークンの色付け, 全解探索/GC用ユニット)	推論ユニット (単一化プロセッサ, プログラム/データ・メモリ, アクティビティ・コントローラ), 構造メモリ (GC, 並列シーケンサ)	推論モジュール (プロセス・プール, 単一化ユニット), 共有構造メモリ (ground instance 格納用)	ヘテロジニアス PE (ストリーム並列処理用, OR並列処理用, データベース処理用)
ネットワーク	リング (12ビット並列要求/ステータス用), 多段網(8ビット並列DMA, 仕事のブロードキャスト機能)	格子 (10ビット並列)		光バス	階層型バス	多重バス		階層型バス (多重専用ネットワーク)	格子	2重2進木 (40ビット並列)
構造体の保持方式	共有			共有	共有, 遅延コピー(要求駆動)	共有		共有	コピー, 一部共有	共有, 遅延コピー
並列処理方式	OR	AND	OR	OR	OR, AND, 引数間	OR, 引数間	OR, ストリーム, 引数間	OR	OR (Prolog) AND (CP)	OR, ストリーム
そのほかの特徴	・分散メモリ型マルチ Prolog マシン・システム ・負荷分配: ひま (idle) となったPEが負荷の割り付け要求を出し, 忙しい (busy) PEが仕事を分割	・分散メモリ型マルチ Prolog マシン・システム ・KL1-Bにコンパイル後実行 ・可変長パケット転送 ・負荷分散制御やプログラムで明示	・専用ハードウェア開発プロジェクトもある ・ノイマン型マシン命令列にコンパイル後実行 ・AND 並列性の逐次処理 ・OR 並列性を abortion 機構により制御可能	・Dialog. H: 光共有バスを用いたマルチプロセッサ・システム ・プロセス・グラフにコンパイル後実行 ・ヘテロジニアス PE 構成	・データフロー・グラフにコンパイル後実行 ・最小クラスタ: 4台PE, 4台構造メモリ	・プロセス・バンドル (バックトラック点の部分集合) 方式によりOR並列性を要求駆動型で制御 ・逐次型 Prolog の並列処理 ・可変長トークンにコンパイル後実行	・色付きトークン (手続き共有) 方式による非決定性実現 ・共有変数の置換による引数間並列処理 ・固定長パケット転送 ・手続き: Prolog の節	・16個の推論ユニットごとに1個の構造メモリを共有し, ground instance を格納 ・アクティビティ・コントローラ: 探索戦略やプログラムの制御構造の実現 ・共有構造体データの遅延フェッチ機能を専用パスで実現	・探索戦略の変更可 ・ネットワーク・ノードが負荷分散を制御可 ・逆順コンパクションと呼ぶリテラル格納方式 ・プロセス・プール内にメッセージ・ボード (並行プロセス間チャネル用分散化共有メモリ)	・PE内の単一化では最大4個の単一化を並列実行可 ・PE対は共有メモリ (レジスタ) 型密結合 ・イベント飛び越し: 処理機能の小さいプロセスを省略

ある.

③要求駆動方式では，論理型言語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り，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．要求駆動方式論理型言語向き並列マシンには，PIE-II[29]，PIM-R[22]，KPR[30]，などがある．

4.2 関数型言語向き並列マシン

4.2.1 関数型言語向きマシンにおける並列性

関数型言語で最も普及しているのは Lisp である．Lisp は記号処理における汎用言語とも言われて適用範囲も広く，たとえば論理型言語の処理システムは Lisp で記述されている場合が多い．また，Lisp の言語仕様の統一化も進められており，さらにオブジェクト指向機能を導入しようという動き[53]もある．

並列論理型言語の実行のために並列マシン・アーキテクチャの開発が要請される（あるいは，言語とアーキテクチャの両方からアプローチされる）という傾向に対して，関数型言語への並列性の導入は，むしろ汎用並列マシンの開発が引き金となっているように見受けられる．

もともとデータ依存性がない関数は独立性が高く，並列処理単位として適している．しかしその場合，粒度は関数機能の大きさによって左右される．たとえば，基本的なリスト処理関数（car, cdr, cons など）と，関数引数を動的に束縛する FUNARG 機能などとは，機能レベルに大きな差がある．FUNARG 機能の実現では，引数の環境保存などを考慮しなければならず，マシン・アーキテクチャとしても環境保存用メモリを単位処理機能に分散して置くのか共有するのか，という選択を迫られる．また，リストの各要素に対する基本演算などは単一プロセッサ内でパイプライン処理した方が効果的な場合が多い．

データ依存性がある関数の引数を並列に評価する場合には，逐次評価を前提とした副作用は利用できない場合がある．並列性の検出を完全に動的に行う方式では，対象言語が従来の逐次型 Lisp のスーパ・セットとなっている場合に，データ依存性の検査がオーバヘッドとなる可能性が高い．また，コンパイラのみによって並列性の検出を行う方式では，静的にデータフロー解析を行い，関数間のデータ依存性を検査する必要がある．したがって，関数型言語向き並列マシンにおいてはプログラム（並列型プログラミング言語）で並列処理対象を明示指定させる方式を採用しているものが多い．

関数型言語の処理における並列性には，次のようなものが考えられる．（1）独立した関数そのものの並列適用，（2）関数適用時の各引数の並列評価，（3）関数呼び出し側の処理と関数本体の処理の並列実行，（4）条件式の並列評価，（5）構造体データの並列処理，（6）関数適用と並列にリアルタイムＧＣを実行する，などである．もちろん，実行時にはこれらの並列性は複雑に組み合わされるので，並列マシンにおける物理的な粒度をあまり細かくすると，（1）などの論理的粒度が大きい場合との機能差が激しくなり，単位処理機能間の通信がオーバヘッドとなる可能性が高い．したがって，Lisp などの関数の機能差が激しい言語に対しては，粒度を比較的大きくとる（たとえば，要素プロセッサに１個の Lisp インタプリタを割り当てる）場合が多い．

並列マシンのメモリ構成法には分散型と共有型とがあるが，関数型言語の実行においても，環境の保持方式や変数の束縛方式などに関連して，このメモリ・アーキテクチャが重要な問題となってくる．粗い粒度のマシンでは，共有型メモリ構成を採用しているものが多い．しかし，高並列性を狙った細かい粒度の関数型言語向き並列マシンにおいては共有型には限界があり，関数型言語の並列実行モデルとしても分散型メモリ構成を前提としたものの出現が待望されている．

4.2.2 関数型言語向き並列マシンの実例

表-2 に，主な関数型言語向き並列マシンの特徴を比較したものを掲げる．関数型言語向き並列マシンも論理型言語向き並列マシンと同様に，命令駆動方式の違いにより次のような３種類に分類できる．

①コントロール駆動方式の関数型言語向きマシンのほとんどが，対象言語を Lisp としている．言語として既に一定の評価がある Lisp のセマンティクスに並列性を導入した言語を対象とするマシンにおいて，なお実用的な性能を失わないためには，汎用（マイクロ）プロセッサや逐次型 Lisp マシンのマルチ化が最も手軽で妥当な方法である．コントロール駆動方式関数型言語向き並列マシンには，東北大学の並列処理システム[50]，SYNAPSE[58]，EVLIS マシン[63]，Concurrent Lisp システム[71]，などがある．

②データ駆動方式は，関数の先行（eager）評価との親和性が高く，並列性も抽出しやすい．また，データフロー・グラフのノードに対応する機能さえ種々用意

表-2 関数型言語向並列マシン

マシン名	(未定)	SYNAPSE	EVLIS マシン	Concurrent Lisp システム	EM-3	DFM-2	MUSE	ALICE	FFP machine	Parallel Graph Reduction Machine	COBWEB
開発機関	東北大学	慶応大学	大阪大学	京都大学および図書館情報大学	電子技術総合研究所	NTT	イギリス・Nottingham 大学	イギリス・London 大学・Imperial 校	North Carolina 大学・Chapel Hill 校	MIT	イギリス・London 大学・Imperial 校ほか
現況	・稼動中（4台PE） ・実行モデルについて定量的評価	・プロトタイプ（逐次型 Lisp インタプリタとGC）稼動 ・拡張中（マルチ化）	・1979 発表（Lisp 言語への並列性導入の先駆け） ・並列インタプリタ／コンパイラ版が稼動中（2台PE，最大構成は7台PE）	・1981発表 ・稼動中（9台PE）	・1983発表 ・1984プロトタイプ（8台PE）稼動	・1985稼動（2台PE，2台構造メモリ） ・8台PE，8台構造メモリ構成・試作中 ・Valid コンパイラ・試作中	・1984.6プロトタイプ(Mark0)稼動（4ストリーム，16環境） ・シミュレーション中	・1981発表 ・1985.11 プロトタイプ稼動 ・1986.3・4台PE・2台パケット・メモリ構成稼動(16台PE・24台メモリに拡張予定)	・1979発表 ・プロトタイプ開発中（数千台PE想定） ・1983 シミュレータ稼動	・実行モデルの提案 ・シミュレーション中	・方式の提案 ・シミュレーション中
対象記号処理用言語	並列型 Lisp	SYNAPSE-LISP	LISP 1.5+並列処理機能	Concurrent Lisp (Lisp1.5+並列処理機能)	EMLisp (Lisp+並列実行命令+単一代入規則)	Valid（関数型言語），並列型 Prolog（純粋 Prolog, CP, OPROG）	データフロー・グラフ	HOPE（適用型言語），Portable Standard Lisp（関数型言語），Parlog	FFP (Formal Functional Programming) 言語	関数型言語	関数型言語（コンピネータ）
応用分野	AI	記号処理，エキスパート・システム，AI	記号処理，ソフトウェア工学		記号処理		汎用（木構造プログラム）		記号処理，数値処理		
論理的粒度	関数の引数のインタプリタ，リストの各要素への関数の適用（マッピング関数），条件式の評価	Lisp インタプリタ，GC	関数の引数のインタプリタ，コンパイラが生成したオブジェクト（各々マイクロプログラム）	インタプリタ，システム・モニタ，GC	関数の評価（パイプライン方式，凝結果による先行評価方式）	関数の引数の評価，関数の部分的実行，先行(lenient cons)/遅延評価	木グラフ（ストリーム）	関数書き換え（βリダクション）	シンボル（ビット・シリアル）	グラフ・リダクション	トークン
物理的粒度	MC 68000 (12.5 MHz, 512KB ローカル・メモリ)	MC 68000	リスト処理専用マシン：EVAL II（パイプライン処理，メモリの並列アクセス機能）	MC 68000 (8 MHz，ローカル・メモリ)，構造メモリ（リスト・セル用，制御スタック用，非リスト・データ用）	MC68000 (6.3 MHz，パケット・メモリ制御回路)	PE（パイプライン構造，インタプリタの核），構造メモリ（リストの基本処理とGC）	PE（プログラム・メモリとシーケンサ付き）	PE（プロセッサ・エージェント，5個の Transputer），構造メモリ（パケット・プール・セグメント，2個の Transputer と 2 MB の RAM）	葉セル（縮小機能マイクロプロセッサ），木セル (ALU)	PE（プロセッサ：reducer/計算メッセージ用／記憶メッセージ用，2メモリ：グラフ用／プロセス記述子用）	ALU，1トークン用メモリ
ネットワーク	クロスバ・スイッチ（PE-メモリ間，PE 側 51 ビット並列，メモリ側39ビット並列）	バス	バス	多重バス	Delta（専用ルータ・チップによる2段網）	多重バス	リング	Delta (XS 1 と呼ぶ 4×4 スイッチ使用，ビット・シリアル)	2進木		六角形状線形（くもの巣状）アレイ
メモリ構成	共有（4 MB，バンク分け可変）	共有（8 MB）	共有	共有（8 MB）	分散（リスト）	共有（論理的） 分散（物理的）	共有（マッチング用）	共有	分散	分散	分散
GC	全PEが並列実行	複数の専用PEによる並列実行	全PEが並列実行	全PEが並列実行		構造メモリが実行		関数書き換えと並列実行			
そのほかの特徴	・割り込み機能を用いたPE間通信 ・マルチプロセッサ・システム ・1台のシステム管理用PE(8MHz・MC68000, CP/M-68K搭載）が入出力処理	・deep-binding 法	・引数の並列評価部分はプログラムで明示 ・コンパイラ版は並列インタプリタ版の10倍以上の性能 ・Prolog インタプリタの評価も有	・並列処理機能：並列プロセス生成，プロセス間通信 ・マルチプロセッサ・システム	・EMIL（データ駆動図式）に変換後実行 ・負荷分散機構は2方式(命令/関数単位）から選択可 ・分散リスト：リスト構造は cons 命令を実行したPE内に生成	・色付きトークン方式 ・Valid で記述した OPLOG インタプリタ：遅延評価による Or 並列性の制御や，先行評価によるストリーム並列処理を実現	・複数データストリームにコンパイル後，各ストリームをパイプライン処理	・制限された先行／遅延評価 ・1989 までに VLSI 版を製作予定 ・グラフ・リダクション・マシン	・直接実行型高級言語マシン（ストリング・リダクション・マシン） ・セル状マシン	・グラフ・リダクション・マシン ・1グラフのリダクションを複数PEで実行	・WSI (Wafer Scale Integration) ・数百台 PE/1 wafer

しておけば，関数型言語向きだけではなく，さらに汎用性を帯びたマシンとすることも可能である．したがって，データ駆動方式関数型言語向き並列マシンには，論理型言語などへの適用を試みている例も多く，データ駆動方式の関数型言語向きマシンと論理型言語向きマシンとの基本的なアーキテクチャの顕著な差は認められないのが普通である．データ駆動方式関数型言語向き並列マシンには，EM-3[68]，DFM-2[16]，MUSE[43]などがある．

③関数型言語の計算モデルはもともとリダクションに基づいているので，要求駆動方式とは親和性がある．数学的基盤（計算モデル）の明確な関数型言語は，要求駆動方式のリダクション・マシンの処理対象とされる場合が多い．要求駆動方式関数型言語向き並列マシンには，ALICE[78]，FFP machine[91]，Parallel Graph Reduction Machine[64]，COBWEB[41]，などがある．

要求駆動方式は，データフロー・グラフ上を要求(demand)がまず伝播し，逆方向のパスでその答え(event)が帰ってくるという仕組みであるので，データ転送コストのみを単純に比較すると，純粋なデータ駆動方式よりは効率が低いと考えられる．しかし，要求駆動方式では遅延(lazy)評価が可能であるという利点もあり，データフロー・マシン[1]においては，データ駆動方式と要求駆動方式との融合なども考えられている．この方式の並列マシンの提案には，前述したDFM-2やMUSEの外にXputer network[28]，などが含まれる．また，イギリスのManchester大学とLondon大学・Imperial校とが共同で(Alvey計画)，FLAGSHIPと呼ぶ関数型言語向き並列マシンやGRIPと呼ぶ並列リダクション・マシンの構想を練っている．

4.3 オブジェクト指向並列マシン

4.3.1 オブジェクト指向マシンにおける並列性

逐次型のオブジェクト指向言語としては，Smalltalk-80が整ったプログラミング環境を提供できる点で代表的である．したがって，現存する逐次型のオブジェクト指向マシンの大半は，Smalltalk-80マシンと言ってもよい．

Smalltalk-80の処理では，高機能命令セットのインタプリタ，動的なデータ型付け，頻繁で煩雑な手続き呼び出し，オブジェクト用メモリ管理，などの機能を必要とするが，Smalltalk-80のセマンティクス自体には並列性は含まれていない．また，Smalltalk-80システムでは，システム移植の便宜を配慮して，「バイト・コード」と呼ぶ仮想マシン命令セットが用意されており，Smalltalk-80マシンの大半は，このバイト・コードのパイプライン処理や低レベル並列処理を行う方式のもの（これらについては，第4.6節で述べる）である．すなわち，Smalltalk-80やそのバイト・コードは逐次マシン（ノイマン型マシン）上で処理されることを前提にして設計された言語である．

しかし元来，オブジェクトとは自己完備で独立性の高い計算実体であり，高度の並列性をセマンティクスとして含んでいると考えるのが自然である．たとえば，Actor理論では並列性を活用できる[2]．実用性（実用的な処理速度）を無視すれば，むしろこのような並列型オブジェクト指向言語が本質的に備えている並列性（オブジェクト生成，メッセージ伝送，返答メッセージ生成，状態の書き換えなどに見られる）を抽出して，それをそのまま物理的な単位処理機能で並列処理する並列マシンが考えられる．この場合の難しさは，個々のオブジェクトの機能（処理，メモリ，通信）がまちまちなので，物理的粒度のそろった並列マシン上で種々のレベルの論理的粒度を実現しなければならないことである．そのほかに，粒度の異なるオブジェクト属性のインヘリタンス（環境）を並列マシン上で，どのように（分散／共有型）実現するのかという難問もある．

また，オブジェクト指向言語は論理型言語や関数型言語に比べて，よりデータ抽象化が進んでおり，データと手続きは一体化されている．このため，オブジェクト指向言語の諸概念は，他の論理型言語や関数型言語の機能を強化するために，そのセマンティクスの一部として取り込まれる傾向が強い．したがって，単一のオブジェクト指向言語を対象とする並列マシンとともに，処理対象を論理型言語や関数型言語とオブジェクト指向言語を融合したセマンティクスを持つ言語に置いたマシン（第2章で(D)型と呼んだ）の開発の方も盛んである．この型のマシンには，FLAGSHIP，FAIM-1[74]，ELIS[55]，Sword 32[36]，PSI[69]，Dorado[82]，などがある．FLAGSHIP（第4.2節で述べた）やFAIM-1（第4.4節で述べる）を除いては，いずれも逐次性の強いマシン・アーキテクチャを持っている（これらについては第4.6節で述べる）．

4.3.2 オブジェクト指向並列マシンの実例

表-3にオブジェクト指向並列マシンの特徴を示す．オブジェクト指向並列マシンの実例はまだ少ないが，

表-3 オブジェクト指向並列マシン

マシン名	Oraga Itself	ZOOM
開発機関	東京大学	慶応大学
現況	・ハードウェア・シミュレータ（3台のMC 68000・1MBメモリ付き）製作中	・1984実行モデルの発表 ・シミュレーション中
対象記号処理用言語	Dinner Bell (Actor 理論，単一代入規則)	オブジェクト指向言語
応用分野	並列型言語のデバッグ支援	
論理的粒度	2レベル（オブジェクト内メソッドとメソッド内基本処理）	オブジェクト
物理的粒度	コンテクストの reducer，メモリ（リダクション・ルール用，コンテクストのスケジューラ用，単一代入変数用，cell Actor 用）	基本命令（オブジェクトと名前の関連付け，名前の連想，オブジェクトの評価）
そのほかの特徴	・PE内ベクトル／パイプライン化 ・ルールはコピー方式 ・トップダウン・アプローチ ・コンテクスト：メソッドの引数	・分散型メモリ（命令用，グローバル名前表，オブジェクト用） ・Orient 84/K（オブジェクト指向システム）の核 ・マルチ・オブジェクト指向逐次マシン・システム

Oraga Itself[44] や ZOOM[70]，などがある．

また，Actor 理論を基にしたオブジェクト指向並列マシンとして，他に MIT の Apiary[93] という提案がある．

4.4 高並列記号処理マシン

4.4.1 高並列記号処理マシンの処理方式

人工知能分野における知識や意味は，複雑なデータ構造をとり，普通これはリスト構造で表現される．このデータ構造をそのままマシン・アーキテクチャ上に写像したものが高並列記号処理マシンである．このマシンでは，細かい粒度の単位処理機能（要素プロセッサ）を多数並べて，汎用性を確保する狙いがある．粒度を細かくすると，単位処理機能間の通信オーバヘッドとなる恐れが出てくるので，完全な MIMD (Multiple Instruction stream Multiple Data stream) 方式よりも SIMD (Single Instruction stream Multiple Data stream) 方式やマルチ SIMD 方式が採用される場合が多い．

たとえば，プロダクション・システム (OPS5)[57] においては，(a)プロダクション・メモリ(PM)とワーキング・メモリ(WM)間とでマッチングをとり，条件を満たすルールを求める，(b)ルール中から選択基準に従って，特定のルールを選択する，(c)選択されたルールのアクションを実行し，WMの内容を更新する，という処理が繰り返される．このサイクルは，もちろん並列あるいはパイプライン処理することができるが，(a)～(c)の各処理にそれぞれ適した粒度やメモリ構成法（共有型／分散型）があり，これらは動的に変化する．この性質に対処するためには，マシン・アーキテクチャとしても SIMD 方式や MIMD 方式を応用に対して使い分けられることが望ましい．

SIMD 方式を実現するためには，システム全体に流れる命令列を管理する機能（プロセッサ）が必要であり，マルチ SIMD 方式のマシンには，コントロール駆動方式に他の駆動方式を組み合わせる方法が適している．したがって，物理的粒度に差がある要素プロセッサを組み合わせたり，複数個の要素プロセッサを別のコントロール駆動方式プロセッサで管理したりする方式によって，階層的なプロセッサ構成とするのが普通である．

4.4.2 高並列記号処理マシンの実例

高並列記号処理マシンには，FAIM-1[74]，IXM[34]，Connection Machine（プロトタイプは CM-1 と呼ぶ）[96]，CAP[95]，PASM[80]，DADO2[77]，NON-VON 3[89]，PSM[94]，PESA-1[3]，SNAP[5]，などがある．表-4 に，これらの高並列記号処理マシンの比較を掲げる．また，これらについては本特集号の他の解説記事で詳述される予定であり，詳細についてはそれによって補ってほしい．

4.5 汎用マルチプロセッサ・システムによる記号処理

4.5.1 汎用マルチプロセッサ・システムの並列処理方式

一般にマルチプロセッサ・システムは MIMD 方式の一種であると考えられ，全プロセッサがメモリを共有する共有メモリ型と，各プロセッサがローカル・メモリを備える分散メモリ型とがある．

共有メモリ型ではメモリ・アクセスの衝突が問題となり，あまり高度な並列化は望めないので，粗い粒度にならざるを得ない．

分散メモリ型では，要素プロセッサ以外に単位処理機能は存在しないので粒度をそろえるのが簡単であり，またその場合に，この型の特徴が生かされる．したがって，分散メモリ型マルチプロセッサ・システムの粒度としては，粗いものから細かいものまで比較的自由に選択できる．たとえば，細かい粒度で高並列性の実現を狙ったマシン（これについては，第4.4節で

表-4　高並列記号処理マシン

マシン名	FAIM-1	IXM	Connection Machine	CAP	PASM	DADO2	NON-VON 3	PSM	PESA-1	SNAP
開発機関	Schlumberger Palo Alto Research	電子技術総合研究所	MIT	ITT	Purdue 大学	Columbia 大学	Columbia 大学	Carnegie-Mellon 大学	Honeywell	Southern California 大学
現況	・1986末・稼動予定(19台PE)	・シミュレーション中(256台PE想定) ・1986連想メモリ・チップ試作予定	・10^4台PEの提案 ・プロトタイプ(CM-1)試作中	・32台(ビット)PE／行・試作中(最大256Kビット／行まで拡張可)	1986夏30台PE試作完了予定(想定は1024台PE)	・1980発表 ・1983. 4-DADO稼動 ・1985. 12-DADO2プロトタイプ(1023台PE)稼動(数千台PE想定) ・OPS5開発完了	・1985. 1 NON-VON 1 (63台PE)稼動 ・NON-VON 3開発中(4000～32000台PE想定，将来は100万台)	・32～64台PEの提案 ・シミュレーション中	・シミュレーション中	・シミュレーション中
対象記号処理用言語	OIL (高級並列型記号処理言語)	IXL (宣言的・手続き的コマンド言語)	Connection Machine Lisp(並列型 Lisp)	OPS5 (RETEアルゴリズム)	並列型アセンブリ言語 (SIMD処理)，CやAdaを基にした並列型手続き向き高級言語 (MIMD処理)	OPS5, PPL/M (並列型システム記述言語)，PPSL (並列型 Lisp)	OPS5	OPS5 (並列型RETEアルゴリズム)	OPS5, ルール指向言語 (分散型RETEアルゴリズム)	(意味ネットワーク)
応用分野	AI (知識表現，論理プログラミング，オブジェクト指向プログラミングプロダクション・ルール)	知識ベース・自然言語処理	イメージ処理，VLSIシミュレーション，意味ネットワーク	AI (意味ネットワーク)，図形処理	イメージ・音声理解，記号処理(エキスパート・システム)	ルール指向プログラミング，論理プログラミング，統計的パターン認識	AI (プロダクション・システム)，データベース管理，イメージ理解，記号処理	プロダクション・システム	プロダクション・システム	サーチ，プロダクション・システム，推論
論理的粒度	タスク	意味ネットワークの複数ノードやリンク	xector (メモリ・セル)		命令 (SIMD処理)，手続き (MIMD処理)	複数ルールと事実(データベース)		RETEネットワーク(データフロー・グラフ)のノード上のトークン (50～100命令)	トークン	意味ネットワークのノード
物理的粒度	Hectagon (ローカル・メモリ，Lispプロセッサ，パイプライン構造の単一化ユニット，構造メモリ，6台の隣接PE)	IXLインタプリタ (2種の連想メモリによる連想サーチと集合演算)，ネットワーク・ノード (マーカ伝搬)	1ビットALU，4Kビット・メモリ，16ビット・フラグ	1ビット・(ビット・スライス)プロセッサ	MC 68000 (プロトタイプ)，マイクロ・コントローラ (4台のPEごと)	Intel 8751 (24 KBローカル・メモリ，4個の8ビット並列ポート)	Small PE (8ビット，32Bメモリ付き) とLarge PE (MC 68020) との階層構成	RISCプロセッサ (ローカル・メモリ，キャッシュ)	PE (入出力バッファ，マッチング・ユニットなど)	PE (連想メモリ，マイクロプログラム制御プロセッサ，通信ユニット)
ネットワーク	2レベル (twisted六角形状格子とその組み合わせ，32ビット並列)	ピラミッド (各ノードに上位：1，下位：4の5ノードへのリンク，最下層ノードの各々に4台PE)	Boolean-cube (16台の隣接PE)	格子	Extra Stage Cube (多段網)	(2重) 2進木	多段網(Large PE間)，2進木(Small PE間)，格子(葉のSmall PE間)	共有バス(複数可)	5本の階層型バス	バス (グローバル)，格子 (ローカル)
並列処理方式	MIMD	マルチSIMD	SIMD	SIMD (行単位)	SIMD MIMD	マルチSIMD	SIMD	MIMD	MIMD	MIMD
そのほかの特徴	・Lispプロセッサ：RISC風，タグ，スタック・キャッシュ ・構造メモリ：命令ストリーム，オブジェクト，連想メモリ ・中間言語命令列にコンパイル後実行 ・負荷割り付け：プログラムで明示／動的 ・並列型Lisp：並列／先行／遅延評価が可能 ・並列型Prolog：OR並列性と制限されたAND並列性を実現	・意味ネットワーク・マシン ・データフロー型マーカ伝搬 ・手続き的知識は節形式で表現 ・最上層ノードに結合されているホスト・マシンからIXL命令をブロードキャスト転送	・パケット・スイッチ・ネットワーク ・CM-1：64K台PE，各PEに4Kビット・メモリ付き (16台PE+1ルータ)／チップ ・PE：複数xectorの各要素同士の並列演算 ・ルータ：隣接PEとのメッセージ授受とxector内の要素間演算	・行中でMIMD処理 (PEの分割) 可 ・共有ローカル・メモリ／行 (最大構成で8MB)	・高信頼性 ・マルチ・ユーザのサポート ・動的にPEをSIMD/MIMD方式の複数クラスタに分割可	・32 PE/1ボードでマルチDADO構成も可 ・SIMD：根から葉へのブロードキャスト機能を使用 ・MIMD：ローカル・メモリ内の命令を使用	・8個のSmall PE/1チップ ・NON-VON 4はマルチSIMD方式	・共有メモリ ・ハードウェア・タスク・スケジューラによる動的スケジューリング ・対 (conjugate) トークンの並列処理	・パイプライン構造 ・動的データフロー	・グローバル・バス：命令のブロードキャスト転送と連想マッチングに利用 ・連想メモリ：セル用，ポインタ用

述べた）も多くなってきた．

現在では，単位処理機能（プロセッサ）としてマイクロプロセッサを手軽に利用できるので，マルチプロセッサ・システムではプロセッサを無理に専用化せずに，粒度をソフトウェアやファームウェアで適当に調整して（あまり細かくせずに），幅広い応用分野での台数効果を狙うものが多い．

4.5.2 汎用マルチプロセッサ・システムによる記号処理例

表-5 に，これらの汎用（多目的）マルチプロセッサ・システムを記号処理へ応用した実例についてまとめてある．表-5 に示した汎用マルチプロセッサ・システムには，Concert Multilisp[33], PARK[24], AQUARIUS[86], CORAL'83[42], DON[56], MANIP-2[25] CDAA[26], Xputer network[28], Dragon[83], などがある．そのほかに，Maryland 大学の PRISM[108] では，ZMOB と呼ぶマルチプロセッサ上に並列推論システムを実現している。

4.6 逐次型記号処理マシンにおける並列性

4.6.1 逐次型記号処理マシンにおける低レベル並列処理

並列演算方式の一種である低レベル並列処理は，比較的長い語長の命令の相異なるフィールドで複数の ALU，メモリ・アクセス機構やバスなどをきめ細かく同時に制御する方式である．したがって，汎用性と高速性の両方を追求する必要のあるユニバーサル・ホスト計算機ではよく採用される方式である．

低レベル並列処理全般に対するハードウェア上のサポート機能としては，命令語が水平型，メモリの分散（バンク）化，データ転送路幅の複数・拡大化，命令やデータ処理に対するパイプライン方式との併用などが必要となる．

また，逐次型言語の処理においても，その処理機能を洗い直してみると，並列処理可能な部分が発見できる．これらの並列性はあまり高くないが，大量データに対してパイプライン効果が利用できたり，対象とする問題に依存した並列／パイプライン処理が可能である場合も多い．たとえば，Prolog マシンや Lisp マシン，Smalltalk-80 マシンなどの専用逐次マシンにおいても，低レベル並列処理方式を採用しているものがある．

4.6.2 低レベル並列処理方式記号処理マシンの実例

表-6 に低レベル並列処理方式を採用している記号処理マシンを掲げる．いずれも処理対象は逐次型言語であり，ユニバーサル・ホスト計算機の場合は記号処理への応用を考慮しているもののみをあげてある．

①Prolog（逐次型）の処理を指向した機能としては，単一化ハードウェア（マッチング機能と高機能順序制御），メモリ（専用高機能スタック，GC 機能付きヒープ），レジスタの専用・大容量化，タグ・アーキテクチャ，などが考えられる．低レベル並列処理方式の論理型言語向き逐次マシン（Prolog マシン）には，PEK[40], CHI[46], Maryland 大学と筑波大学の Prolog マシン[17], AIM[65], ASCA[35], ASSIP-T[39], などがある．

②Lisp（逐次型）の処理では，動的なメモリ管理，リスト格納用に構造化されたメモリ，制御（関数呼び出し管理用）スタックのハードウェア化，関数フレームの切り替え機構，リスト要素に対するパイプライン処理，などの高速化が図られる．低レベル並列処理方式の関数型言語向き逐次マシン（Lisp マシン）には，FLATS[54], 豊橋技術科学大学の関数型言語向きマシン[49], DIALISP[32], などがある．

③ユニバーサル・ホスト計算機の記号処理への応用例には，MUNAP[38] や QA-2[67], などがある．

4.6.3 逐次型記号処理マシンにおけるパイプライン処理

第 4.6.1 節でも述べたが，対象言語が並列型ではなく逐次型である場合には，その記号処理マシン・アーキテクチャには高並列性を導入することは困難である．むしろ，各対象言語のセマンティクスに近い機能レベルのハードウェアやファームウェアを用意する高級言語マシンの設計思想を適用することが有効である．

このような逐次マシンにおいては，前項で述べた低レベル並列処理方式のほかに，その単一処理機能（プロセッサ）内にパイプライン処理構造を備えているものが多い．パイプライン処理は，いわゆるノイマン型である SISD 型や SIMD 型マシンにおける高速化手法の中では最も有効な方法であり，マイクロ命令，マシン命令あるいはもっと高い命令レベルのおのおのにおける適用が試みられている．

たとえば，各機能レベルにおける命令の処理サイクルは，命令フェッチ，命令デコード，オペランド・アクセス，命令オペレーション，命令順序制御の各ステージに分割される．逐次型記号処理マシンにおいては，このうちリストを対象とするオペランド・アク

表-5　汎用マルチプロセッサ・システムの記号処理への応用例

マシン名	Concert Multilisp	PARK	AQUARIUS	CORAL '83	DON	MANIP-2	CDAA	Xputer network	Dragon
開発機関	MIT	神戸大学	California 大学・Berkeley 校	徳島大学	豊橋技術科学大学	Purdue 大学	ドイツ・IBM 研究所	Utah 大学	Xerox
現　況	・稼動中(4台PE, 32台PEまで拡張可) ・並列型 Prolog コンパイラ開発中	・稼動中(4台PE) ・並列型 Prolog コンパイラ開発中	・AQUARIUS-I (単一PE) 稼動中 ・AQUARIUS-II (マルチPE)設計中	・1983稼動 (15台PE) ・68台 PE (MC 68000) 版製作中	・稼動中(7台PE) ・評価中	・方式の提案 ・シミュレーション中	・方式の提案	・方式の提案 ・シミュレーション(数百台PE)	・1984発表 (最大10台PE)
対象記号処理用言語	Multilisp (Scheme+並列性)	並列型 Prolog (PARK-Prolog)	Prolog	Prolog	純粋 Prolog	論理型言語	データフロー言語	FEL (関数型言語)	Cedar, Interlisp, Smalltalk-80
応用分野	汎　用	汎用 (逐次／並列処理の共存), 最適化問題 (動的計画法) など	DA, 離散的シミュレーション, エキスパート・システム, 信号処理	汎用 (数値計算, 記号処理, データベース処理)	組み合わせ最適化問題, ソーティング	NP完全問題	AI, トランザクション処理, 汎用マルチプロセッシング	並列型プログラミング	汎　用
論理的粒度	式 (プログラムで明示), タスク (MCODE インタプリタ)	AND/OR プロセス (Prolog インタプリタ)	20～100 命令	Cで書かれたプログラム	節 (ゴール)	AND/OR 木のノード	プロセス	関数 (パケット)	プロセス
物理的粒度	Concert (汎用マルチプロセッサ・システム共有メモリ, リング・バス) 上の MC 68000 (最大 20 MBメモリ)	MC 68000 (8 MHz, 8 KB ローカル・メモリ), 共有メモリ (データ用 16KB, パケット用 16KB)	論理型言語向きプロセッサ (PLM, パイプライン構造), 数値演算向きプロセッサ, 入出力プロセッサ	Intel 8085 A (RAM: 17KB, ROM: 8 KB)	Z 80 A (4 MHz, ROM: 4KB, RAM: 64KB, DMA 制御回路, SIO, PIO)	汎用プロセッサ	汎用プロセッサ (ローカル・メモリ, キャッシュ付き)	汎用プロセッサ (メモリ付き)	RISC プロセッサ (10 MHz)
ネットワーク	各PEと共有メモリ間にバイパスあり	バ　ス	クロスバ・スイッチ (PE-メモリ間)	2進木	2重2進木	リング	バ　ス	格　子	バ　ス
メモリ構成法	共　有	共有 (512KB)	共有 (複数台)	分　散	分　散	分散 (階層的)	共　有	分　散	共有 (4～16MB)
そのほかの特徴	・Multilisp 自身のコンパイラが稼動 ・中間言語命令 (MCODE) 列にコンパイル後実行 ・MCODE インタプリタはCで記述 ・GCは各PEが独立実行 ・先行評価のパイプライン処理 ・unfair 法: 並列性 (の爆発) を制御するタスクのスケジュール機能	・共有メモリにはプロードキャスト機能付き ・OR並列性の実現 (プログラムで明示) ・AND 並列性のストリーム処理による実現 (プログラムで明示) ・1台のホストPE (Z80)が Prolog システムを管理 ・負荷の割り付け: プログラムで明示可	・ヘテロジニアス・マルチプロセッサ・システム ・AND 並列性をマルチ PLM で実現 (検出はコンパイラによる)	・OR並列性の実現	・OR並列性の実現 ・パイプライン処理の併用	・複数PEがローカル・メモリ (コントローラ) に結合 ・ローカル・メモリにはゴール列を格納 ・AND 並列性とOR並列性との使い分けが可 ・サーチにおける発見的手法の適用	・AND 並列性とOR並列性との実現	・"rediflow"により リダクションとデータフローの各計算モデルを融合 ・リアルタイムGCを目標 ・パケット・スイッチ・ネットワーク ・自動負荷分散 ・マルチプロセッサシステム	・PE: バイパス付きパイプライン構造, 2ポート・キャッシュ付き ・Cedar: strong 型付け, 名前呼び, 実行時GC ・32ビット・汎用ワークステーション

表-6 低レベル並列処理方式記号処理マシン

マシン名	PEK	CHI	(未定)	AIM	ASCA	ASSIP-T	FLATS	(未定)	DIALISP	MUNAP	QA-2
開発機関	神戸大学	ICOT および 日本電気	Maryland 大学 および 筑波大学	NTT	NTT	ドイツ・Kaiseral-autern 大学	理化学研究所 および 東京大学	豊橋技術科学大学	ルーマニア・Polytechnical Institute of Bucharest	宇都宮大学	京都大学
現況	・1984稼動 ・1985 Prolog インタプリタ稼動 ・1986 Prolog コンパイラ稼動	・稼動中（テストプログラム・レベル） ・プログラミング環境デバッグ中	・方式の提案，評価	・論理設計完了 ・LSI レイアウト設計中	・Prolog のファームウェア・インタプリタ稼動中 ・マルチ・システム化を予定	・シミュレーション中	・REDUCE 3 稼動中	・1984稼動	・稼動中	・1981稼動	・1983稼動
対象記号処理用言語	Prolog	Prolog	Prolog	Prolog	純粋 Prolog＋cut/fail 機能	第一階述語論理	Lisp	関数型言語	Lisp	純粋 Prolog, L$^{\bullet}$, Smalltalk-80, MSDL（システム記述言語）など	Prolog, Lisp など
応用分野		知識処理			VLSI 用 DA システム	定理証明	数式処理（記号・数値処理）			記号処理，データベース処理，数値計算	図形・画像・信号のリアルタイム処理，高級言語処理
低レベル並列処理機能	・メモリの分散化 ・UNDO 処理の並列化 ・構造体データのパイプライン処理	・引数マッチング，スタック操作，固定小数点演算，浮動小数点演算を命令実行ステージで専用ハードウェアにより並列実行	・連想メモリを利用した大規模知識データベースの並列サーチ ・引数間並列性を並列／パイプライン処理で実現 ・連想メモリを用いた UNDO 処理	・論理合成 CAD システムによって分割された33個の機能（頭部引数やゴールのフェッチ，変数束縛，トレイル処理，変数初期化など）を並列／パイプライン処理方式で実現 ・メモリのバンク化	・節起動と引数の単一化とのパイプライン構造 ・連想メモリにより逐次 UNDO を省き，並列ＧＣを実現	・連想メモリ（推論過程用，単一化グラフ用）を用いて，推論と単一化の並列実行	・スタックの並列読み出し ・スタック上のデータ移動操作やデータ型検査の並列化 ・ハードウェアによる並列ハッシング機能 ・ハードウェアによるＧＣ	・関数の引数関数適用部，条件式評価部，基本演算部などにより並列評価 ・リスト処理の基本関数のハードウェア化 ・ＧＣと関数評価との並列実行	・デュアル・メモリを用いた，ＧＣやスタック・メモリ・アクセス，木構造関数などの並列実行	・２レベル・マイクロプログラム制御方式 ・４個の16ビット・プロセッサ（ALU，ビット／フィールド処理用回路，16ワード・レジスタファイル，１Ｋワードのスクラッチパッドメモリ）	・４個の 32 ビット同一機能 ALU，主記憶，シーケンス制御部などを１マイクロ命令により MIMD 制御 ・６KB の共有大容量レジスタファイル
そのほかの特徴	・構造体共有方式	・命令セット・アーキテクチャは WAM コード・レベル ・パイプライン処理の併用	・直接実行型プログラミング環境	・１チップ LSI マシン ・構造体共有方式 ・最適化(TRO)	・４Ｋビット連想メモリ使用	・単一化：コンストレイント付き ・コンビネータ・コードにコンパイル後実行	・バックエンド型数式処理マシン ・パイプライン処理の併用	・マシン命令列にコンパイル後実行 ・データ駆動方式による関数評価 ・パイプライン処理の併用	・deep-binding 法 ・キャッシュ・ハードウェア・インタプリタ（タグ・アーキテクチャ） ・ミニコン複合体(DIAGRAM)に結合	・Prolog：引数間／ＯＲ並列性の評価 ・L$^{\bullet}$：ソフトウェアのテスト・システムの実験対象 ・Smalltalk-80：メッセージ授受用高機能中間言語命令を設計	・Prolog：高レベル・インタプリタ版と最適化コンパイラ版

セスや命令オペレーションが機能的に負荷が大きいステージであると考えられる．したがって，これらのステージにおいては，連想メモリを用いたサーチ並列性の実現や，専用機能による低レベル並列処理などの併用が必要である．

また，逐次マシン内の処理構造だけではなく，並列マシンの単位処理機能がパイプライン構造を持ち，マシン全体のアーキテクチャによる高並列性を補完する（並列-パイプライン）方式のマシンも数多い．これらについては，低レベル並列処理方式のマシンとの区別が困難であるものが多い．

4.6.4 逐次型記号処理マシンの実例

以上に述べてきた逐次型記号処理マシンの実例としては，他に次のようなものをあげることができる．

論理型言語向き逐次マシン（Prolog マシン）としては，ICOT の PSI[69]や PSI-II，および California 大学・Berkeley 校の PLM[87] がある．これらはマイクロ命令レベルでのパイプライン処理を採用している．また，PSI ではファームウェア化された高レベル・インタプリタにより，きめ細かくハードウェアを制御しているので一種の低レベル並列処理（マイクロ命令レベルでの）方式のマシンとみなすこともできる．PLM は，第4.5節で述べた AQUARIUS の要素プロセッサでもある．

また，Stanford 大学が提案した逐次型 Prolog マシン[90]は，アーキテクチャを簡潔にして高速のマシン・サイクルを実現し，これをパイプライン方式で処理している．このマシン命令セット[92]（「WAM コード」などと呼ばれている）は，現在各所で提案されている Prolog コンパイラやコンパイラ指向逐次型 Prolog マシン（CHI や PSI-II, PLM など）の対象言語の基本となっている．さらに，このマシン命令レベルのパイプライン構造を2本備えるマシンの提案・評価[98]が行われている．

関数型言語向き逐次マシン（Lisp マシン）としては，NTT の ELIS[55]，富士通の FACOMα[51]，Symbolics 社の Symbolics 3600[97]，California 大学 Berkeley 校の SPUR[4]，Oregon Graduate Center の G-machine[45]，などがある．

オブジェクト指向逐次マシン（Smalltalk-80 マシン）には，東京大学の Sword 32[36]や沖電気工業の Hobbes[61]がある．いずれも Smalltalk-80 のバイト・コードをパイプライン処理するアーキテクチャを採る．以上のほかに，オブジェクト指向逐次マシンとしては，日立製作所の AI 32[37]，California 大学 Berkeley 校の SOAR[23]，Caltech の COM[21]，カナダ・Tronto 大学の Swamp[109]がある．特に，SOAR には並列タグ検査や，並列レジスタ初期化機能などが備わっている．

そのほかに，パイプライン処理方式の記号処理マシンとして，プロダクション・システム・マシンである Carnegie-Mellon 大学の RISCF[66] を初めとして，フランス・LSI 研究所の LTR[62]，Xerox 社の Dorado[82] などがある．また，高級言語の直接実行を記号処理として捉えるならば，筑波大学の UDEC[31] もこの範ちゅうに入る最新の稼動例である．

以上に述べた逐次型記号処理マシンのうち，RISC (Reduced Instruction Set Computer) アーキテクチャを採用しているものは，SPUR, SOAR, G-machine RISCF などである．

5. おわりに

本文では，各種の並列型記号処理マシンの開発動向やそれらのアーキテクチャ上の特徴について述べた．なにぶん「記号処理」は多岐にわたる問題分野であるので，おのおののマシンの詳細な説明は紙数の都合で割愛せざるを得なかった．それらについては，末尾に付けた文献リストなどを参考にして，各論文によって補ってほしい．

逐次マシンの開発としては先行した Lisp マシンであるが，並列記号処理マシンとしては並列 Lisp マシンと並列 Prolog マシンとが肩を並べて開発を競い始めたというのが現状である．

応用の面からは，本質的に問題の持つ並列性を記述可能な並列記号処理言語の出現も待たれる．これらの言語としては，Lisp や Prolog, Smalltalk-80 などの各機能を融合したセマンティクスを持つものが考えられようが，いずれにしても大規模な並列性を含んだ応用の掘り起こしが危急の課題である．

細かい粒度のシステムにおいてはもちろん，粗い粒度の並列マシンを構成する要素プロセッサにもチップ化の波が押し寄せてくることが予想され，マシンの高並列化は容易になるとともに，単位処理機能間の通信方式のブレーク・スルーが要求されてくるであろう．

記号処理マシンの並列化（対象言語や処理機能への並列性の導入）にともなう解決すべき課題は，むしろ一般的なものであり，並列マシン全般に関連する問題と言えるものが多い．たとえば，高並列化にともない

システム全体の制御が難しくなるが，マシンの初期化や処理結果の取り出し方法などについての研究は，まだ緒についたばかりである．

ハードウェア資源が有限であるシステム上で細かい論理的粒度を実現する場合には，要素プロセッサ自体のハードウェア／ファームウェア／ソフトウェアのトレードオフが重要な問題となってくる．たとえば，最近では RISC アーキテクチャを採用した逐次型マシン（前節の最後で列挙した）も見られるので，これらを用いて並列マシンを構築する場合などにもシステム全体のトレードオフに対する慎重な考察が必要である．

謝辞 本解説の執筆にあたり，各種記号処理マシンの開発者の皆さまには，資料やコメントの提供などに関してご協力いただきました，ここに深謝いたします．

末筆ながら，日頃ご指導いただく，京都大学工学部萩原宏先生，九州大学大学院総合理工学研究科富田眞治先生に深甚なる謝意を表します．

参 考 文 献

1) 雨宮真人：データフロー・アーキテクチャについて，コンピュータ・ソフトウェア，Vol. 1, No. 1, pp. 42-63 (1984).
2) 米澤明憲：オブジェクト指向型プログラミングについて，コンピュータ・ソフトウェア，Vol. 1, No. 1, pp. 24-41 (1984).
3) Ramnarayan, R., Zimmermann, G. and Krolikoski, S.: PESA-1: A Parallel Architecture for OPS5 Production System, 19th Annual Hawaii Int. Symp. on System Sciences, pp. 201-205 (1986).
4) Taylor, G. S. et al.: Evaluation of the SPUR Lisp Architecture, 13th Annual Int. Symp. on Comput. Architecture, pp. 444-452 (1986).
5) Moldovan, D. I.: An Associative Array Architecture Intended for Semantic Network Processing, ACM '84 Annual Conf. (The Fifth Generation Challenge), pp. 212-221 (1984).
6) 稲葉則夫：人工知能プログラムを高速に実行するマルチプロセサ方式の並列処理マシン，日経エレクトロニクス，No. 387, pp. 143-168 (1986).
7) 田中英彦：並列推論マシン，情報処理学会研究会資料，Vol. CA-57, No. 3, p. 8 (1985).
8) 田中英彦：非ノイマン型コンピュータ[I],[II] 信学誌，Vol. 68, No. 7～8, pp. 773-778, 869-874 (1985).
9) 井田，山本，竹内(編)：特集：記号処理と計算機アーキテクチャ，情報処理，Vol. 23, No. 8, pp. 705-772 (1982).
10) 金田悠紀夫：Prolog マシン，情報処理，Vol. 25, No. 12, pp. 1345-1352 (1984).
11) 新田克己：並列 Prolog，情報処理，Vol. 25, No. 12, pp. 1353-1359 (1984).
12) 井田哲雄：LISP マシンのアーキテクチャ，情報処理，Vol. 26, No. 7, pp. 732-740 (1985).
13) 小長谷，山本：関数型言語とリダクションマシン，情報処理，Vol. 26, No. 7, pp. 751-764 (1985).
14) 雨宮真人：関数型言語とリスト処理向きデータフロー・マシン，情報処理，Vol. 26, No. 7, pp. 765-779 (1985).
15) ICOT WGI・並列推論マシン・サブグループ：並列推論マシン・アーキテクチャの調査，ICOT 研究速報，No. TM-0098, p. 38 (1985).
16) Amamiya, M. et al.: Implementation and Evaluation of a List-Processing-Oriented Data Flow Machine, 13th Annual Int. Symp. on Comput. Architecture, pp. 10-19 (1986).
17) Chu, Y. and Itano, K.: Architecture for a Parallel Associative Prolog Machine, 19th Annual Hawaii Int. Symp. on System Sciences, pp. 9-23 (1986).
18) Umeyama, S. and Tamura, K.: A Parallel Execution Model of Logic Programs, 10th Annual Int. Symp. on Comput. Architecture, pp. 349-355 (1983).
19) Halim, Z.: A Data-Driven Machine for OR-Parallel Evaluation of Logic Programs, New Generation Computing, Vol. 4, No. 1, pp. 5-33 (1986).
20) Ito, N. et al.: The Architecture and Preliminary Evaluation Results of the Experimental Parallel Inference Machine PIM-D, 13th Annual Int. Symp. on Comput. Architecture, pp. 149-156 (1986).
21) Dally, W. J. and Kajiya, J. T.: An Object Oriented Architecture, 12th Annual Int. Symp. on Comput. Architecture, pp. 154-161 (1985).
22) 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，Logic Programming Conf. '85, No. 2.1, pp. 14-25 (1985).
23) Ungar, D. et al.: Architecture of SOAR: Smalltalk on RISC, 11th Annual Int. Symp. on Comput. Architecture, pp. 188-197 (1984).
24) 松田ほか：並列 Prolog マシン PARK について，Logic Programming Conf. '85, No. 2.4, pp. 39-48 (1985).
25) Li, G. and War, B. W.: MANIP-2: A Multicomputer Architecture for Evaluating Logic Programs, 1985 Int. Conf. on Parallel Processing, pp. 123-130 (1985).
26) Diel, H.: Conurrent Data Access Architecture, Int. Conf. on Fifth Generation Comput.

Systems 1984, pp. 373-382 (1984).
27) Kumon, K. et al.: KABU-WAKE: A New Parallel Inference Method and Its Evaluation, COMPCON Spring 86, pp. 168-172 (1986).
28) Keller, R.M., Lin, F.C.H. and Tanaka, J.: Rediflow Multiprocessing, COMPCON Spring 84, pp. 410-417 (1984).
29) 平田ほか: PIE のメモリ試作ハードウェアの設計について, 信学技報, Vol. EC-85, No. 64, pp. 55-65 (1986).
30) 柴山ほか: 論理プログラミング指向並列リダクション・マシン KPR のアーキテクチャ, 信学技報, Vol. EC-85, No. 70, pp. 43-54 (1986).
31) 板野ほか: UDEC: 汎用直接実行型計算機(1)～(4), 情報処理学会第 30 回全国大会講演論文集, No. 3C-3～6, pp. 223-230 (1985).
32) Stefan, G. et al.: DIALISP—A LISP Machine, 1984 ACM Symp. on LISP and Functional Programming, pp. 123-128 (1984).
33) Halstead, R.M. Jr.: Implementation of Multilisp: Lisp on a Multiprocessor, 1984 ACM Symp. on LISP and Functional Programming, pp. 9-17 (1984).
34) 樋口ほか: 並列連想記憶を用いた意味ネットワークマシン, 信学技報, Vol. EC-85, No. 55, pp. 9-20 (1986).
35) 長沼, 小倉, 山田: 連想メモリを用いた Prolog マシンの構成と処理アルゴリズム, 情報処理学会研究会資料, Vol. SM-32, No. 3, p. 8 (1985).
36) Suzuki, N., Kubota, K. and Aoki T.: Sword 32: A Bytecode Emulating Microprocessor for Object-Oriented Languages, Int. Conf. on Fifth Generation Comput. Systems 1984, pp. 389-397 (1984).
37) Nojiri, T., Sakoda, K. and Kawasaki, S.: Microprogrammable Processor for Object-Oriented Architecture, 13th Annual Int. Symp. on Comput. Architecture, pp. 74-81 (1986).
38) 石川ほか: 多重プロセッサ計算機における Prolog の並列処理, 情報処理学会第 30 回全国大会講演論文集, No. 7C-6, pp. 213-214 (1985).
39) Dilger, W. and Schneider, H-A.: ASSIP-T. A Theorem Proving Machine, Int. Conf. on Fifth Generation Comput. Systems 1984, pp. 497-506 (1984).
40) 田村ほか: シーケンシャル実行型 Prolog マシン PEK—ハードウェア構成—, 情報処理学会論文誌, Vol. 26, No. 5, pp 855-861 (1985).
41) Hankin, C.L., Osmon, P.E. and Shute M.J.: COBWEB—A Combinator Reduction Architecture, Functional Programming Languages and Computer Architecture, Springer-Verlag, Berlin, pp. 99-112 (1985).
42) Takahashi, Y. et al.: Efficiency of Parallel Computation on Binary-Tree Machine CORAL '83, Journal of Information Processing, Vol. 8, No. 4, pp. 288-299 (1986).
43) Brailsford, D.F. and Duckworth, R.J.: The MUSE Machine — An Architecture for Structured Data Flow Computation, New Generation, Computing, Vol. 3, No. 2, pp. 181-195 (1985).
44) 河野ほか: 統合プログラミング環境 ORAGA (IV)並列オブジェクト指向アーキテクチャ Oraga Itself, 情報処理学会第 31 回全国大会講演論文集, No. 1E-9, pp. 303-304 (1985).
45) Kieburtz, R.B.: The G-machine: A Fast, Graph-Reduction Evaluator, Functional Programming Languages and Computer Architecture, Springer-Verlag, Berlin, pp. 400-413 (1985).
46) Nakazaki, R. et al.: Design of a High-Speed Prolog Machine (HPM), 12th Annual Int. Symp. on Comput. Architecture, pp. 191-197 (1985).
47) 瀧ほか: Multi-PSI システムの概要, 情報処理学会第 32 回全国大会講演論文集, No. 5Q-8, pp. 101-102 (1986).
48) 竹内彰一: 論理型並列プログラミング言語—Concurrent Prolog, コンピュータ・ソフトウェア, Vol. 1, No. 2, pp. 25-37 (1984).
49) 飯田ほか: 関数型言語向き計算機アーキテクチャの一方式, 情報処理学会論文誌, Vol. 27, No. 3, pp. 339-349 (1986).
50) 黒川ほか: MC 68000 を用いた並列処理システムの試み, 情報処理学会研究会資料, Vol. CA-59, No. 14, pp. 119-132 (1985).
51) Yuhara, M. et al.: Evaluation of the FACOM ALPHA LISP Machine, 13th Annual Int. Symp. on Comput. Architecture, pp. 184-190 (1986).
52) Ueda, K.: Guarded Horn Clauses, Logic Programming Conf. '85, No. 9.3, pp. 225-236 (1985).
53) 石田ほか: Common Lisp へのオブジェクト指向機能導入の動向, 情報処理学会研究会資料, Vol. SM-36, No. 7, p. 8 (1986).
54) 平木, 後藤, 数式処理計算機 FLATS のアーキテクチャ, 情報処理学会論文誌, Vol. 27, No. 1, pp. 81-89 (1986).
55) 日比野, 渡辺, 大里: Lisp マシン ELIS のアーキテクチャ—メモリレジスタの汎用化とその効果, 情報処理学会研究会資料, Vol. CA-24, No. 3, p. 8 (1983).
56) 佐野ほか: マルチコンピュータシステム DON 上での PROLOG の実現と性能評価, 信学技報, Vol. AL-83, No. 85, pp. 49-62 (1984).
57) 小林重信: プロダクションシステム, 情報処理, Vol. 26, No. 12, pp. 1487-1496 (1985).
58) 加藤ほか: SYNAPSE の機能と動作について,

情報処理学会研究会資料, Vol. SM-30, No. 3, p. 7 (1984).
59) Yasuhara, H. and Nitadori K.: ORBIT: A Parallel Computing Model of Prolog, New Generation Computing, Vol. 2, No. 3, pp. 277-288 (1984).
60) 石田 亨: プロダクションシステムと並列処理, 情報処理, Vol. 26, No. 3, pp. 213-225 (1985).
61) 飯塚, 山田: Smalltalk-80 専用マシンのハードウェア構成, 情報処理学会第 32 回全国大会講演論文集, No. 5S-1, pp. 253-254 (1986).
62) Lecussan, B.: A High Level Language Computer System [HLLCS] in a Multitasking Environment, Int. Workshop on High-Level Comput. Architecture 84, pp. 8.16-29 (1984).
63) 前川ほか: 試作 EVLIS マシンの EVALII と開発支援機能, 情報処理学会研究会資料, Vol. SM-17, No. 1, p. 9(1982).
64) Traub, K.R.: An Abstract Parallel Graph Reduction Machine, 12th Annual Int. Symp. on Comput. Architecture, pp. 333-341 (1985).
65) 小栗, 中村: SFL システムによる推論マシンの設計, 信学会昭和 61 年度総合全大, No. 1463 pp. 6.105 (1986).
66) Lehr, T.F.: The Implementation of a Production System Machine, 19th Annual Hawaii Int. Symp. on System Sciences, pp. 177-186 (1986).
67) Tomita, S. et al.: A Computer with Low-Level Parallelism QA-2—Its Applications to 3-D Graphics and Prolog/Lisp Machines—, 13th Annual Int. Symp. on Comput. Architecture, pp. 280-289 (1986).
68) 戸田, 山口, 弓場: 記号処理用データ駆動計算機EM-3 の予備的評価, 信学技報, Vol. EC-85, No. 13, pp. 59-66 (1985).
69) Nakajima, K. et al.: Evaluation of PSI Micro-Interpreter, COMPCON Spring 86, pp. 173-177 (1986).
70) Ishikawa, Y. and Tokoro, M.: The Design of an Object Oriented Architecture, 11th Annual Int. Symp. on Comput. Architecture, pp. 178-187 (1984).
71) Sugimoto, S. et al.: A Multi-Microprocessor System for Concurrent LISP, 1983 Int. Conf. on Parallel Processing, pp. 135-143 (1983).
72) 日比野靖: 並列形ガーベジコレクション, 情報処理, Vol. 24, No. 4, pp. 423-430 (1983).
73) 塩原ほか: データフロー方式による Prolog 並列処理, 信学技報, Vol. EC-85, No. 63, pp. 43-54 (1986).
74) Davis, A.L. and Robison S.V.: The Architecture of the FAIM-1 Symbolic Multiprocessing System, 9th Int. Joint Conf. on Artificial Intelligence, pp. 32-38 (1985).
75) Pereira, L.M. and Nasr R.: DELTA-PROLOG: A Distributed Logic Programming Language, Int. Conf. on Fifth Generation Comput. Systems 1984, pp. 283-291 (1984).
76) Clark, K. and Gregory, S.: PARLOG Parallel Programming in Logic, Research Report DOC 84/4, Imperial College of Science and Technology, Univ. of London, p. 37 (1984).
77) Stolfo, S.J. and Miranker D.P.: The DADO Production System Machine, Journal of Parallel and Distributed Computing, to appear, p. 33 (1986).
78) 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 Comput. Architecture, pp. 65-74 (1981).
79) 横手, 所: Concurrent Smalltalk, 日本ソフトウェア科学会第1回大会・論文集, No. 2E-2, pp. 179-182 (1984).
80) Meyer, D.G. et al.: The PASM Parallel System Prototype, COMPCON Spring 85, pp. 429-434 (1985).
81) 神田ほか: 統合プログラミング環境 ORAGA (I)並列オブジェクト指向言語 Dinner Bell, 情報処理学会第 31 回全国大会講演論文集, No. 1E-6, pp. 297-298 (1985).
82) Pier, K.A.: A Retrospective on the Dorado, A High-Performance Personal Computer, 10th Annual Int. Symp. on Comput. Architecture, pp. 252-269 (1983).
83) Monier, L. and Sindhu, P.: The Architecture of the Dr agon, COMPCON Spring 85, pp. 118-121 (1985).
84) Ciepielewski, A., Hausman, B. and Haridi, S.: Initial Evaluation of a Virtual Machine for Or-Parallel Execution of Logic Programs, IFIP TC-10 Working Conf. on Fifth Generation Comput. Architectures, p. 19 (1985).
85) 石川, 所: 並行オブジェクト指向知識表現言語 Orient 84/K, コンピュータソフトウェア, Vol. 3, No. 3, pp. 24-42 (1986).
86) Despain, A.M. and Patt Y.N.: Aquarius—A High Performance Computing System for Symbolic/Numeric Applications, COMPCON Spring 85, pp. 376-382 (1985).
87) Dobry, T.P., Patt Y.N. and Despain A.M.: Design Decisions Influencing the Microarchitecture for a Prolog Machine, 17th Annual Workshop on Microprogramming, pp. 217-231 (1984).
88) 米澤ほか: オブジェクト指向に基づく並列情報処理モデル ABCM/1 とその記述言語 ABCL/1, コンピュータソフトウェア, Vol. 3, No. 3, pp.

9-23 (1986).
89) Shaw, D. E.: NON-VON's Applicability to Three AI Task Areas, 9th Int. Joint Conf. on Artificial Intelligence, pp. 61-72 (1985).
90) Tick, E.: Sequential Prolog Machine: Image and Host Architectures, 17th Annual Workshop on Microprogramming, pp. 204-216 (1984).
91) Magó, G.: Making Parallel Computation Simple: The FFP Machine, COMPCON Spring 85, pp. 424-428 (1985).
92) Warren, D. H. D.: An Abstract Prolog Instruction Set, SRI Technical Note, No. 309, p. 30 (1986).
93) Hewitt, C. and Lieberman, H.: Design Issues in Parallel Architectures for Artificial Intelligence, COMPCON Spring 84, pp. 418-423 (1984).
94) Forgy, C. and Gupta A.: Preliminary Architecture of the CMU Production System Machine, 19th Annual Hawaii Int. Symp. on System Sciences, pp. 194-200 (1986).
95) Brooks, R. and Lum, R.: Yes, An SIMD Machine can be Used for AI, 9th Int. Joint Conf. on Artificial Intelligence, pp. 73-79 (1985).
96) Hillis, W. D.: The Connection Machine, The MIT Press, p. 145 (1985).
97) Moon, D. A.: Architecture of the Symbolics 3600, 12th Annual Int. Symp. on Comput. Architecture, pp. 76-83 (1985).
98) Tick, E.: Towards a Multiple Pipelined Prolog Processor, Int. Workshop on High-Level Comput. Architecture 84, pp. 4.7-17 (1984).
99) 阿部, 加久間, 中川: 通信系主導型並列 Prolog 処理システムの検討, 情報処理学会研究会資料, Vol. CPSY-86, No. 21, p. 13-20 (1986).
100) 片岡ほか: グラフ操作に基づく Prolog 並列実行マシンのアーキテクチャ, 情報処理学会研究会資料, Vol. CA-62, No. 1, pp. 8 (1986).
101) Knight, T.: An Architecture for Mostly Functional Languages, 1986, ACM Symp. on LISP and Functional Programming, pp. 105-112 (1986).
102) Castan, M. et al.: MARS: A Multiprocessor Machine for Parallel Graph Reduction, 19th Annual Hawaii Int. Symp. on System Sciences, pp. 152-159 (1986).
103) Steinberg, S. A. et al. The Butterfly Lisp System, Proc. of the National Conf. on Artificial Intelligence (AAAI), pp. 730-734 (1986).
104) Kacsuk, P.: Some Approaches to Parallel Implementation of Prolog, Information Processing 86 (IFIP 86), pp. 803-809 (1986).
105) Ramsdell, J. D.: The CURRY Chip, 1986 ACM Symp. on LISP and Functional Programming, pp. 122-131 (1986).
106) Scheevel, M.: NORMA: A Graph Reduction Processor, 1986 ACM Symp. on LISP and Functional Programming, pp. 212-219 (1986).
107) Kawasaki, S., Nojiri, T. and Sakoda, K.: A User-Adaptable VLSI Engine for Artificial Intelligence, Information Processing 86 (IFIP 86), pp. 367-372 (1986).
108) Kasif, S., Kohli, M. and Minker, J.: PRISM: A Parallel Inference System for Problem Solving, 8th Int. Joint Conf. on Artificial Intelligence, pp. 544-546 (1983).
109) Lewis, D. et al.: Swamp: A Fast Processor for Smalltalk-80, 1986 ACM lst Annual Conf. on Object-Oriented Programming Systems, Languages and Applications (OOPSLA '86) (1986).
110) Gabriel, R. P. and McCarthy, J.: Queued-based Multi-processing Lisp, 1984 ACM Symp. on LISP and Functional Programming, pp. 25-44 (1984).

付 録

本解説の初稿を投稿後にもいくつかの記号処理マシンが発表された．付録として，これらのうち本解説の範囲にあると思われるものについて，簡単に紹介する．

金沢工業大学が，複数の単一化プロセッサやゴール生成用プロセッサを，負荷分散機能付き階層型バスとそれらバス間の通信メモリとで結合したマルチプロセッサ構成方式により，純粋 Prolog の OR 並列性を実現するシステムを検討している[99]．

千葉大学は，表-1 に掲げたデータ駆動方式のマシン[73]とは別に，ホーン節をグラフに変換後，連想メモリを用いてリダクションを行い，純粋 Prolog の OR 並列処理とストリーム並列処理を実現する並列グラフ・リダクション・マシンを検討している[100]．

Symbolics 社と MIT は，共有メモリ型マルチプロセッサ上で Lisp 関数の引数の完全な並列評価を許す方式を提案している．プログラムは一旦トランザクション・ブロックと呼ぶ部分の集合にコンパイルされ，そのおのおのが独立に要素プロセッサにより並列実行される．Multilisp[33] のように言語の構文を並列実行向きに拡張するのではなく，引数の並列評価を実行すると正しくない場合を発見し，それを修正する操作を，実行時に，各要素プロセッサが装備している2個の連想メモリを用いて行う方式である[101]．

フランス・ONERA-CERT は，MaRS と呼ぶ並列グラフ・リダクション・マシンを提案し，シミュレーションにより，その構成方式について評価している．MaRS は，複数のリダクション・プロセッサ，メモリ・プロセッサ，入出力プロセッサが複数個の通信プロセッサを多段構成したスイッチング・ネットワークを介して結合されたマルチプロセッサである．対象は関数型言語であり，そのコンビネータ・コードをマシン語とする[102]．

BBN 社は，自社が開発した Butterfly と呼ぶマルチプロセッサ上に，"future" 構文という並列評価機構を付加することにより Common Lisp システムを拡張した並列記号処理環境を構築した．現在，Butterfly Lisp はインタプリトされているが，将来はコンパイラも開発する予定である[103]．

ハンガリー・SZKI は，バス結合のマルチプロセッサ（要素プロセッサは IBMPC コンパチブル・マシン）上に MPROLOG システムを開発し，制限された OR 並列性を実現している．将来は，トランスピュータのようなプロセッサを多数用いたホモジニアス・システム上にデータフロー型計算モデルに基づいた MPROLOG システムを構築し完全な OR 並列性を実現する予定である[104]．

MITRE 社は，シリコン・コンパイラを用いて，CURRY と呼ぶ逐次型リダクション・マシンの VLSI チップを作った．ラムダ式で表現されたプログラムをコンビネータにコンパイル後，CURRY がこれをリダクションする．GC 専用チップも同様にして合成され，これらにコントローラとメモリを付加したものを 24 ビット・バスで結合してシステムを構成する[105]．

Burroughs 社が開発した NORMA は，コンビネータ・グラフ用のリダクション・マシンである．200 ns/サイクルで逐次リダクションを行うグラフ・プロセッサ，グラフ・ノード（2 セルで 64 ビット長）を格納するメモリと GC 用マーク・ビットを格納するメモリとからなるグラフ・メモリ，GC などのメモリ管理をリダクションと並行して行うアロケータ，などがバス結合され，これらが 370 ビット長のマイクロ命令で制御される低レベル並列処理方式を採用している[106]．

日立製作所の AI 32 は，当初発表されたオブジェクト指向逐次マシン[97]としてよりも，ユーザ定義可能なマイクロプログラム記憶を備えて，記号処理分野におけるユニバーサル・ホスト計算機アーキテクチャとしての性格を強めている[107]．

その後，逐次型の Lisp マシンや Prolog マシンの商品化は，ますます盛んになっている．また，RISC 上に汎用の記号処理システムを構築する試みも多い．逐次型 Prolog マシンの構成方式では，中間言語レベルの仮想マシンとして WAM コード[92]を採用する場合が大半である．

複数の並列論理型言語に共通の仮想マシン命令セットを設計する試みも各所で行われている．本解説では除外したが，単一化処理とデータベース処理の融合を図るために，データベース・マシンに論理型言語の諸機能を取り込もうとする動向もある．

（昭和 61 年 5 月 16 日受付）