TR－A－0078

ノイマンを超えて

－視聴覚認知機構研究における並列計算構成理論について－
＝視聴覚認知機構研究に必要とされるであろう計算機について＝

Joe　Kazuki
城　　和貴

1990. 3. 30

ATR視聴覚機構研究所

〒619-02 京都府相楽郡精華町乾谷　☎07749-5-1411

ATR Auditory and Visual Perception Research Laboratories

Inuidani, Seika-cho, Soraku-gun, Kyoto 619-02 Japan

Telephone: +81-7749-5-1411
Facsimile: +81-7749-5-1408
Telex: 5452-516 ATR J

ノイマンを越えて

－視聴覚認知機構研究における並列計算理論の必要性について－
－視聴覚認知機構研究に必要とされるであろう計算機について－

城　和貴
ＡＴＲ視聴覚機構研究所

1．　概要

人間の視聴覚認知機構の基礎研究は本研究所をはじめとして、工学、心理、生理の様々な分野の研究者の間で研究が行われている。このような研究で構築されたモデルは、脳の並列分散メカニズムを考えると、並列計算機にインプリメントしてこそ、以後の応用分野への発展が期待できる。一方、並列計算機は近年、ノイマン型計算機からの脱出が期待されている疎結合型並列計算機が開発され、既存の逐次処理アルゴリズムの見直しが注目されている。本研究所は視聴覚認知機構の研究において、研究員、研究設備、研究実績とも日本の最先端であることを誇っているが、研究により得られたモデルを並列計算機上に実現する手法については、米国 MIT, CMU, Caltec, JPL 等の研究所に及ばない。それは、視聴覚認知機構の研究によって得られたモデルを、効率よく並列計算機上に実現するための並列計算構成理論についての研究がおこなわれていないからである。そこで、本稿は、現在のノイマン型計算機の問題点及び、並列計算機についての現状について概説し、視聴覚認知機構の研究における並列計算構成理論の必要性について説明し、それらを満足させるための計算機アーキテクチャの提案を行う。

2．　これまでの（現在の）計算機アーキテクチャ

この章では現在のノイマン型計算機の問題点、並列分散処理の可能性、近い将来の計算機科学の動向、日本の現状についての概説を行う。

2．1　ノイマン型計算機の限界

今日の科学技術の進歩は計算機なしでは考えられないことは周知のことであるが、その計算機の大半は今だにフォン・ノイマンの束縛(*)から脱出していないと言えよう。そして、その事実を十分に認識している研究者が少ないのも事実であろう。例えば、ある視覚情報処理アルゴリズムが提案され、既存のワークステーション上にインプリメントした結果、処理に1時間かかったとする。従って、1万倍速いスーパーコンピュータ(*)にインプリメントすれば、人間の処理速度と同じであると言ってよいのだろうか？

現在の計算機のスピード・アップは目ざましく、まるでゴールのないレースのようにも見えるが、実は純粋な意味でのスピード・アップはそれほどなされていない。あるノイマン型計算機の１０倍の性能の計算機を作るためには、理想的にはチップの集積度、ＣＰＵのクロック、メモリ・アクセス、バス転送速度のすべてが１０倍にならなければならない。ところが、こういった純粋なスピードアップはコストが高くつくので、（集積度に関しては明かな限界がある。つまり、チップ内の電子の移動速度は光速を越えられない。）計算機アーキテクター達は、ＣＰＵのクロックを上げる以外に、パイプライン手法(*)を、メモリ・アクセスにはキャッシュ・メモリ(*)を、バス転送速度の問題には複数のバスによる転送パイプライン(*)を、それぞれ開発し、純粋なスピード・アップに見せかけているのが現状である。例えば、１MFLOPS(*)の処理能力を持つワーク・ステーションと１０００MFLOPSの処理能力を持つスーパー・コンピュータを比較した場合、ＣＰＵの速さが１０００倍違うと思われがちであるが、実際の速さの違いは数十倍程度である。（ワークステーションのＣＰＵのクロックが数十ｎｓであるのに対し、スーパーコンピュータのそれは数ｎｓである。）現在のスーパーコンピュータの処理速度の実態は、実は処理を行うタスクの大きさに依存しているのである。しかも、各タスクが十分な大きさのベクトル演算を含んでいなければならず、この意味で、チューリング・マシン(*)の性質を最大限に利用したものと言えよう。図２．１はスーパー・コンピュータのベクトル計算方法を示している。

このようなことを考えると、現在のノイマン型計算機の数万倍以上も速い計算機は純粋な並列処理を組み込む以外に不可能であると言えよう。（スーパーコンピュータのベクトル演算処理は、並列処理と見なせる。）

2．2　並列処理

人間の視聴覚情報処理が並列処理によって実現されていることは明らかなことだが、計算機の並列化も古くから研究されている。（１９世紀の歯車を利用した自動計算機には、複数の計算ユニットがあった。また、並列計算機の元祖と言われているイリアック以前にも、様々なところで並列計算機モデルが提案されている。）イリアック等の並列計算機に代表されるアーキテクチャは、現在のそれの基本となっているが、当時のコスト・パフォーマンスから考えると、１００個の

プロセッサを並列構成させるより、１００倍速いチップを作る方が効果的であったため、提案された様々なモデルは、必ずしも製品化はされなかったのである。

１９７０年代に入ると、クレイ社を初めとするベクトル・プロセッサ(*)が商業的に成功し、現在に至っている。スーパーコンピュータにおけるベクトル・プロセッサは演算パイプライン(*)と呼ばれる手法により大きなベクトルの各要素を並列に計算させるものであり、並列処理のカテゴリーに属する。スーパーコンピュータはノイマン型計算機を念頭に置いたベクトル化がコンパイラにより容易にできたため、高価なハードウェアにもかかわらず、数値計算の権化として現在尚君臨している。

ベクトル・プロセッサ開発が研究されていた頃、一方ではダイクストラ等の数学者により[Dijkstra,1971]、相互排除問題（複数の計算機によるデータ競合の問題）に対する研究が行われ、セマフォ(*)やモニタ(*)と呼ばれるエレガントな解答が提案された。これは、コンピュータ・サイエンスの分野における、チューリング・マシン以後最大の快挙と言えよう。ところが、計算機アーキテクト達はこの成果をデータ・フロー・マシン等に利用するよりは、チューリング・マシンに組み込むことを考えたのである。そしてそれらが実際の計算機アーキテクチャに組み込まれた結果、今日の Alliant や Convex を初めとする密結合型並列計算機(*)が確立した。これら密結合型並列計算機は、イリノイ大学で開発された自動並列化コンパイラ（イリノイ大学は、一番外側のループを並列化するという単純なコロンブスの卵を見つけ、コンパイラ技術の革命的な飛躍を成し遂げた。）を利用することにより、現在の数値計算分野における代表的な計算機になりつつある。

しかしながら、これらの計算機は当然のことながらフォン・ノイマンの束縛から逃れていないのである。密結合型並列計算機は、ただ単に並列に計算を行わせるマシンであり、前提となるアルゴリズムはチューリング・マシンのレベルを上回っていないのである。すなわち、巨大な行列計算のように逐次計算でかつ大部分の計算は他から独立しているようなタスクのためのアルゴリズムを前提としているのである。これは当然ハードウェア的な制約をもたらす。例えば、スーパーコンピュータ開発時に提案されたパイプラインの手法は非常に強力で、ノイマン型計算機の計算能力を数百倍にも引き上げたが、パイプラインの段数、ベクトル・レジスタの長さなど、物理的な制約が存在する。それとまったく同様に、密結合型並列計算機においてはプロセッサの数が最大でも１６台程度という物理的な制約がある。（メモリを共有するという制約のため）図２．２は密結合型並列計算機の問題点を示している。すなわち、共有メモリを任意のプロセッサがアクセスする場合、自分が共有メモリのアクセスをしているということを他の全てのプロセッサに知らせてやらなければならないため、多数のプロセッサの場合、その通信に時間がかかりすぎることになり、十分なパフォーマンスが得られない。

並列計算機には、密結合以外に疎結合(*)とよばれるものがあるが、これはある意味ではフォン・ノイマンからの脱出を期待されている。密結合型やスーパーコンピュータは、前述したように解くべき問題のサイズが大きくなることを利用した並列化であるが、疎結合型の場合は問題のサイズに依存しないからである。（そのかわり、自動並列化等はむずかしい）もちろん、疎結合型並列計算機の各ＣＰＵはノイマン型であるが、効果的なデータ通信のためのトポロジー、例えばハイパーキューブ構造（図２．３参照）、を利用することにより、ノイマン型計算機では計算量の爆発的増大をもたらすようなアルゴリズムがＬｏｇＮの計算量で処理できることが知られている。[NCUBE Handbook] ＮＣＵＢＥやＣＭ（Conne

ction Machine）に代表される疎結合型並列計算機は、この意味で並列分散処理に適したアーキテクチャを持っていると言えよう。図２．４はハイパーキューブ構造を用いた並列計算の簡単な例を示している。

このように、並列計算機として我々は現在、スーパーコンピュータ、密結合型並列計算機、疎結合型並列計算機の３種類を使うことができるが、大ざっぱな特徴で考えると、図２．５で示されているように、スーパーコンピュータは最も高価で、ノイマン型計算機に近く、従ってプログラムも容易である。逆に疎結合型並列計算機は最も安価で、ノイマン型計算機のカテゴリーには入りきらず、従ってプログラムは出来あいのアルゴリズムを利用しにくいためむずかしい。密結合型並列計算機は両者の中間的な性格を持っているといえよう。

２．３　計算コストの問題（チューリング・マシンの問題点）

コンピュータ・サイエンスにおける最初の偉業はチューリング・マシンの確立である。〔Turing,1936〕図２．６はチューリング・マシンの定義を示している。この原始的なモデルによって、我々は処理させるべきタスクの処理時間の概算を行うことができるのである。例えば、オーダＮの計算(*)とは、データが２倍の大きさになったときに、計算量も２倍になるものであり、オーダN^2の計算とは、データが２倍になったときに、2^2倍になるものである。（画像前処理アルゴリズムの多くはオーダN^2である）従って、アルゴリズムを考えるときに、我々は、いかにして計算オーダを少なくするか、というアプローチを無意識のうちに取っている。

このようなアプローチは、スーパーコンピュータや密結合型並列計算機までは異論の余地は無い。ところが、疎結合型並列計算機の場合は条件が変わってくる。なぜならば、チューリング・マシンの定義には、データ格納領域は無限に大きく、データ・アクセスに要する時間が一様であるという仮定が存在するからである。スーパーコンピュータや密結合並列計算機の場合は、仮想記憶システムの導入によりデータ格納領域の問題はなくなり、効率のよいページング・システムの開発によりデータ・アクセスの一様性の問題も少なくなっている。ところが、疎結合並列計算機の場合は、各プロセッサが直接アクセスできる局所的なデータ格納領域には物理的な制限があり、（例えばＮＣＵＢＥの場合は５１２KB）他のプロセッサのデータ格納領域をアクセスするには、目的とする演算時間よりもデータ転送に関する時間が長くかかる可能性があるわけである。（池田・川人の画像復元参照。彼らの最初のプログラムは疎結合型計算機のトポロジーをまったく考慮に入れてなかったため、並列化しないものより遅くなってしまった。その後、リング構造のトポロジーを入れることにより本来の性能を引き出すことができた。池田氏のプログラミング能力がなければ日の目を見なかった研究になっていたかもしれない。）この事実はチューリング・マシンを前提として開発されたアルゴリズムを根底から覆しうる。

このように、チューリング・マシンを原点とし、より速く、より大量なデータを処理することを目指してきたコンピュータ・サイエンスは、疎結合型並列計算機の出現により大きな軌道変更を余技なくされるだろう。すなわち、大規模で複雑な計算を行わせる際に、データのアクセスにも、そのデータがどこにあるかによってアクセス・タイムが変わってくるのである。そして、その通信時間は計算量の一部として組み込まれるべきである。このような提案はすでに G.C.Fox らに

よって提案され、以下のように定式化されている。〔Fox,1984,1986,1986〕

$$S = N / (1 + const * T_{comm} / (n^{1/d} * T_{calc}))$$

（上式は、疎結合型並列計算機にインプリメントされたアルゴリズムの、ノード数Ｎの増加にともなう処理速度増加の割合Ｓを示す。通信にかかる時間T_{comm}がまったく無ければノード数の増加がスピードアップにそのままつながる。T_{calc}は各ノードの行う計算に要する時間、ｎは各ノードに分散して計算されるデータ配列の１辺の長さ、ｄはデコンポジション・ディメンジョンと呼ばれるアルゴリズムに応じた並列度係数、ｃｏｎｓｔは定数を示す。）

２．４　超並列計算機＝将来の汎用計算機

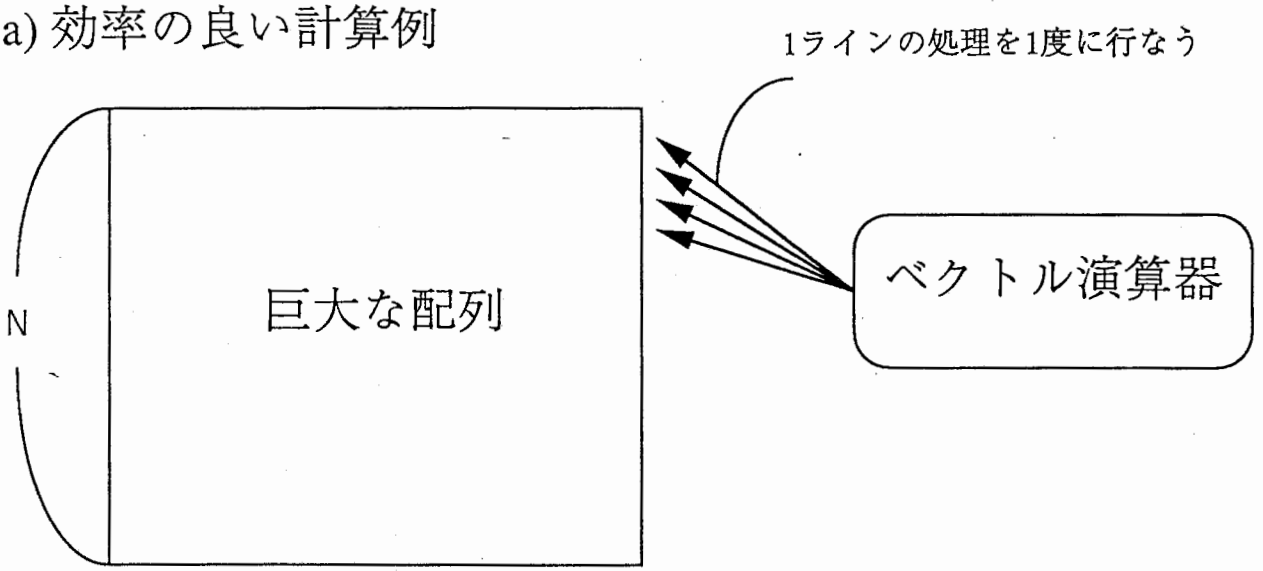
汎用計算機は、この３０年の間に飛躍的な進歩を遂げた。処理速度、データ量ともに数万倍から数十万倍になっている。そして、この進歩はノイマン型のスタイルのまま続くのであろうか？ジョセフソン素子は現在のシリコン素子より数万倍も速くなるのだろうか？１秒間に現在の磁気ディスク数万個分のデータを転送できるような超高速バスは物理的に可能なのだろうか？

このような問いかけに対する一つの解答がＣＭやＮＣＵＢＥをはじめとするハイパーキューブ型超並列計算機であろう。現在の技術で数千ＧＦＬＯＰＳのマシンは不可能だが、ハイパーキューブ型超並列計算機では比較的容易であろう。実際、従業員数わずか５０人のＮＣＵＢＥ社は２７ＧＦＬＯＰＳのマシンをすでに発表している。ＣＭの次世代機もかなり大型のものになると予想される。ただ、これらを使いこなすためには、現在まで蓄えられてきた数値計算アルゴリズムの多くをつくり直さなければならない。そして、そのような研究はすでにＭＩＴ，Ｃａｌｔｅｃｈ，ＪＰＬ，Ｉｎｔｅｌ等で精力的に行われつつある。

このように、コンピュータ・サイエンスの分野においては、ハイパーキューブ型超並列計算機＝将来の汎用機という予想の元に、既存の数値計算アルゴリズムの見直しが注目されているにもかかわらず、日本におけるそのような動向はきわめて少ない。当然、ＮＣＵＢＥやＣＭ、ｉＰＳＣ等の計算機を所有しているところが少ないことも原因ではあるが、それよりはむしろ日本の研究者の意識がフォン・ノイマンの束縛から逃れられないのが最大の要因ではないだろうか。あるいは、米国においてハイパーキューブ型アーキテクチャのための数値計算アルゴリズムの完成をじっと待っているのだろうか。

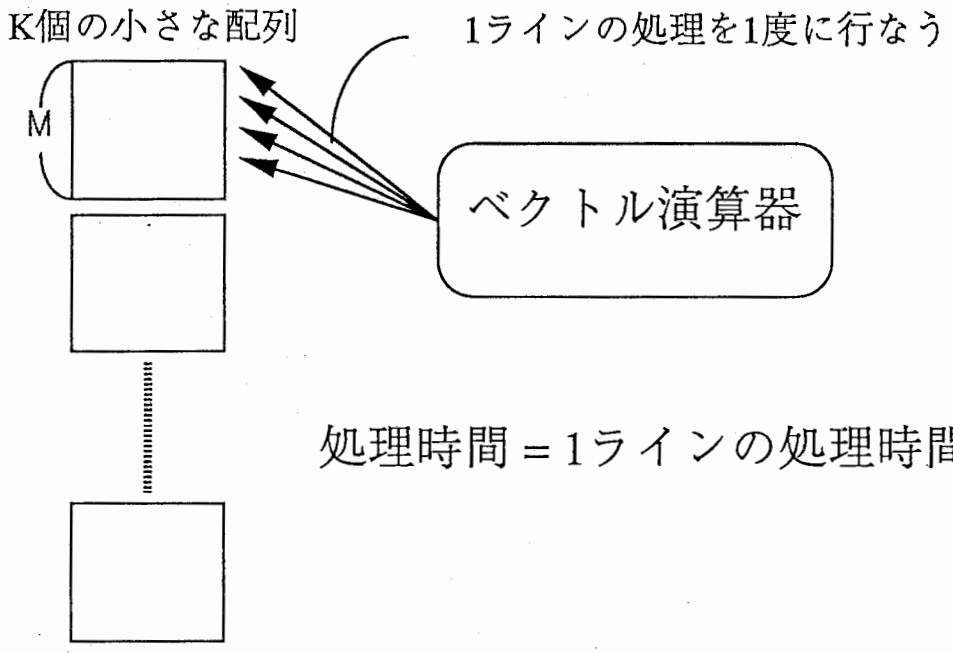
図2.1 スーパーコンピュータのベクトル演算

a) 効率の良い計算例



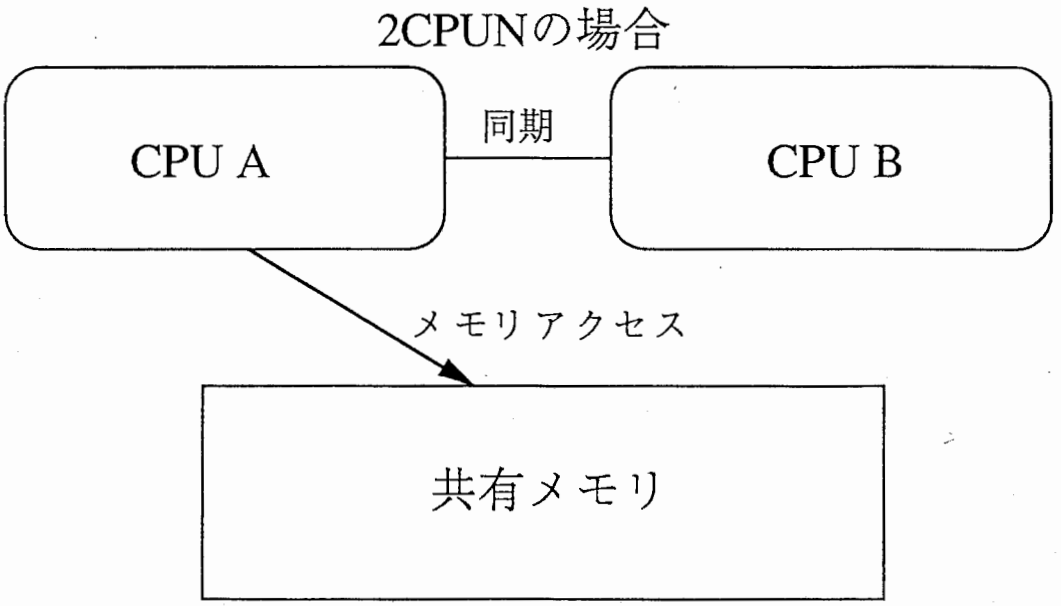
処理時間＝1ラインの処理時間＊N

b)効率の悪い計算例

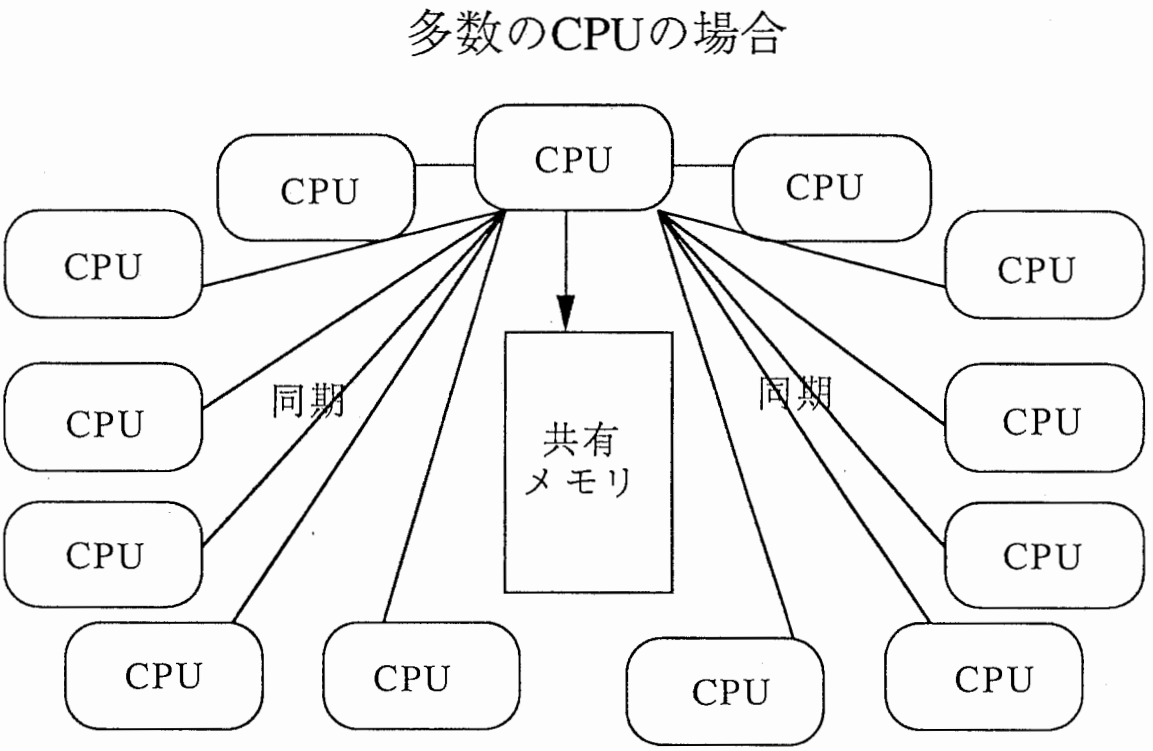


処理時間＝1ラインの処理時間＊M＊K

図2.2 密結合型並列計算機の問題点



1度のメモリアクセスに対して1度の同期



1度のメモリアクセスに対して多数の同期

図2.4 ハイパーキューブ使用例

$$\int f(x)\,dx = \sum_i \int f(x_i)\,dx$$

計算部分をN個に分割し並列計算させた後
LogNの通信で結果を得る

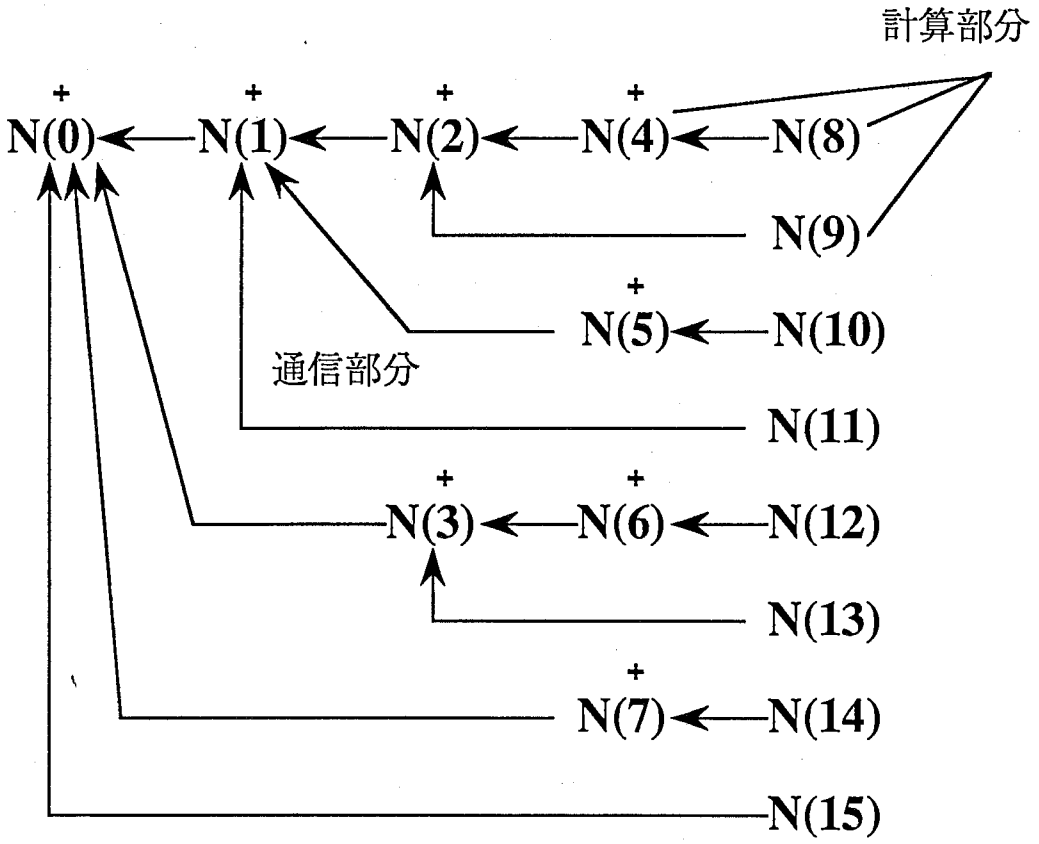


図2.5 現在の並列計算器の位置付け

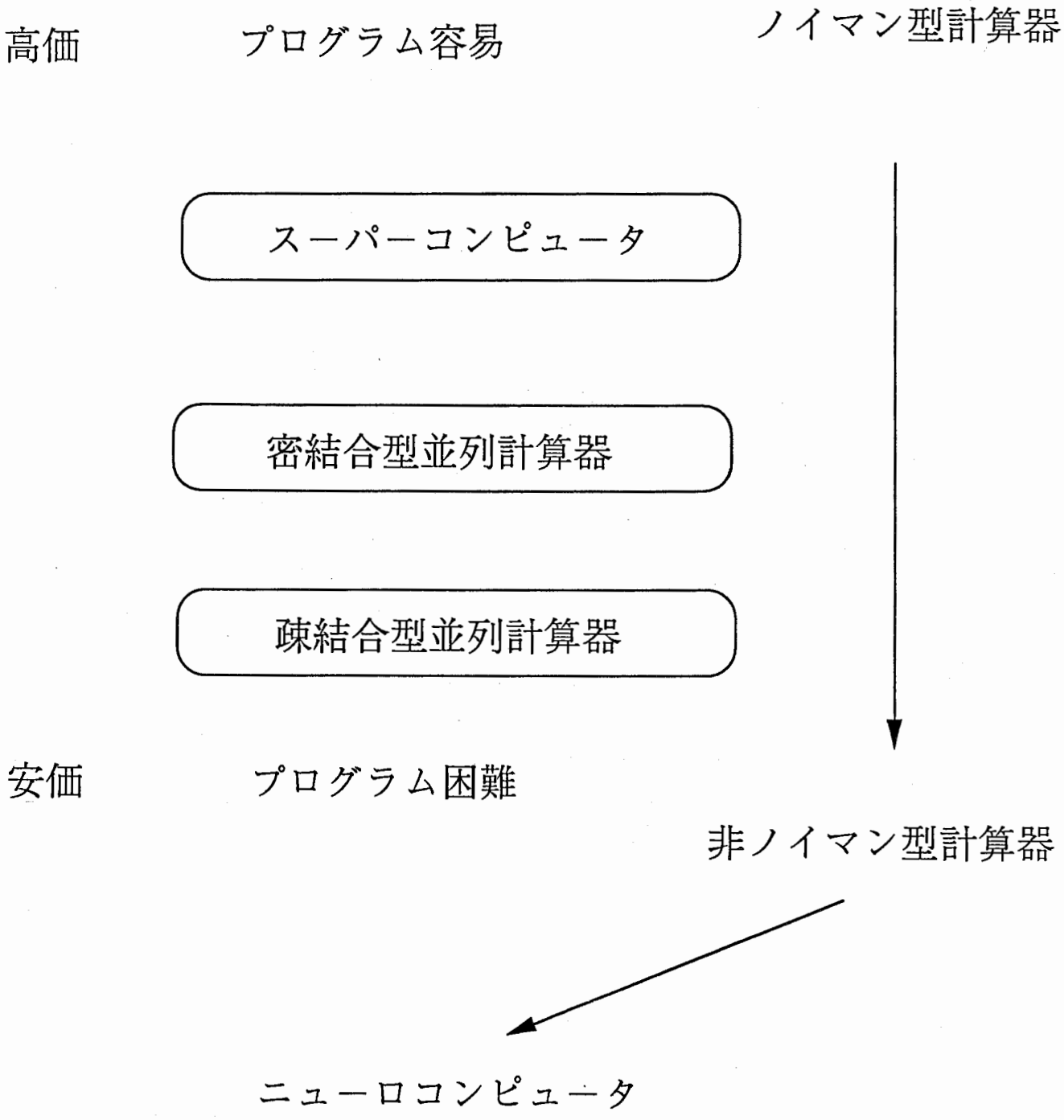
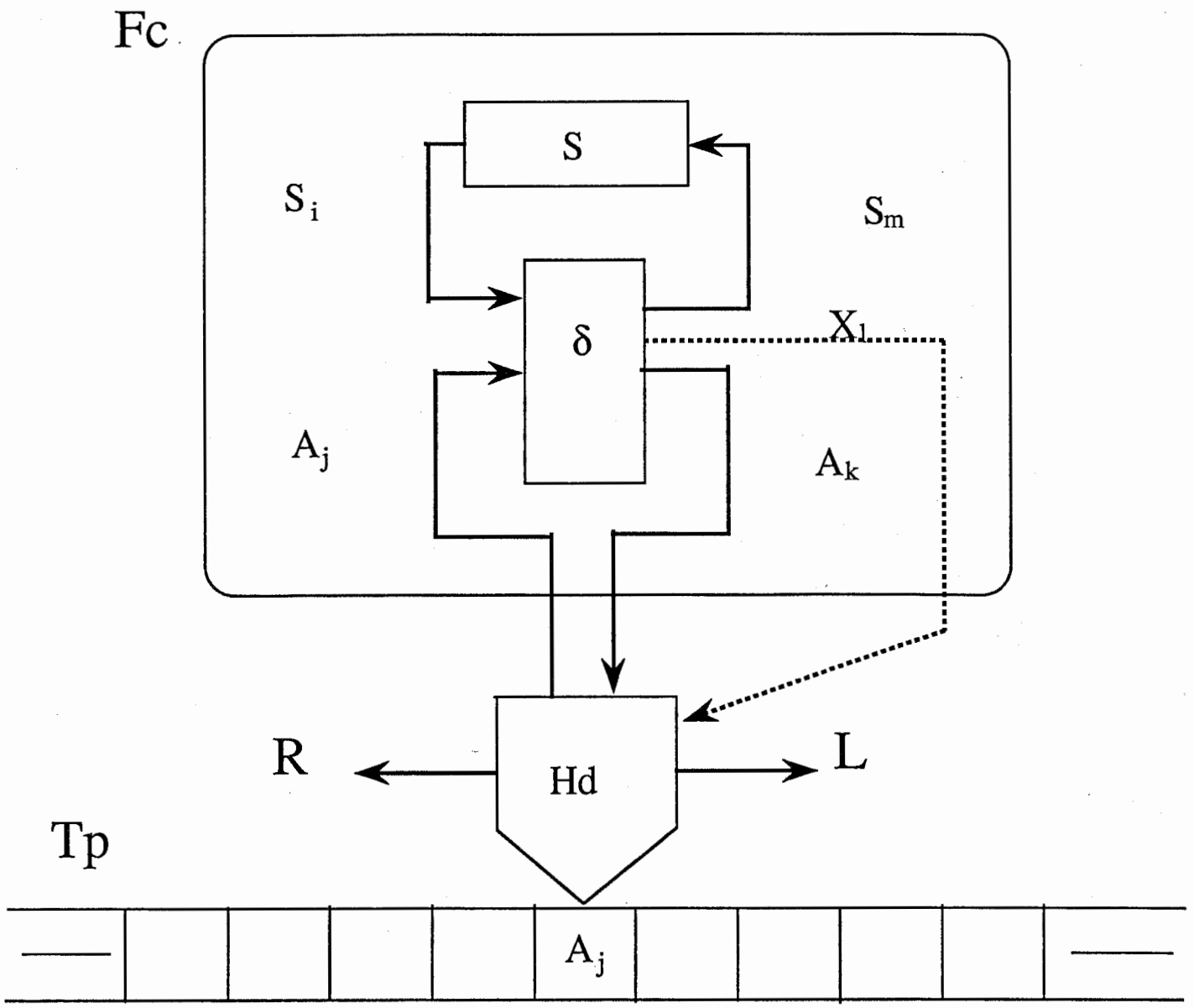


図2.6 チューリングマシンの定義



Fc 有限制御部
Tp テープ(無限長)
Hd ヘッド

S 内部状態
δ 状態遷移関数
X ヘッドの移動方向

3．　現在（近い将来）必要とされる計算機アーキテクチャ

この章ではハイパーキューブ型並列計算理論の研究が、コンピュータ・サイエンスの立場からだけではなく、視聴覚認知機構の研究を行う上でも必要なことの説明を行う。

3．1チューニングの問題

あるアルゴリズムを計算機にインプリメントする際に、その計算機特有の環境をうまく利用し、高速化を図ることをチューニングと言う。一般的なチューニングには次のようなものがある。

① 数値計算ライブラリの利用
例えば、行列の固有値を求めるような場合、既存の数値計算ライブラリのルーチンを利用する方が高速な処理が期待できる。
② ファイルＩＯの最適化
ユーザ・プログラムは一般的にブロック単位のファイルＩＯを行っていないことが多いので、ブロック単位のファイルＩＯに変更するだけで、全体的に数倍速くなることもある。
③ システム・ルーチンの利用
特殊なタスクはシステム・コールを利用することにより高速化できる。
④ プログラムの最適化
レジスタの割付等を考慮にいれて、プログラムの構造を最適化する。コンパイラのオプティマイザ(*)が信用できる場合は不必要である。
⑤ 明示的なベクトル化、並列化
アレイ・プロセッサ、ベクトル・プロセッサ（スーパーコンピュータを含む）、密結合型並列計算機等の場合、システム構成に応じたベクトル化、並列化が効果的である。
⑥ アルゴリズム自体の変更
ここまでくると、もはやチューニングではない。

一般的なチューニングは、ユーザのレベルで①、外注やコンサルタントで②③程度と言えよう。④ではプログラムの内容を完全に把握する必要があり、⑤ではさらに計算機アーキテクチャについての知識が必要となる。

さて、あるアルゴリズムを疎結合型並列計算機にインプリメントするとき、その計算機の性能を出し切るようなインプリメンテーションを行うにはどうすればよいのだろうか。まず逐次処理計算機用の数値計算ライブラリは利用できない。従って、ハイパーキューブ構造をもたせたアルゴリズムでそのルーチンを作成しなければならない。次に、ファイルＩＯは通信経路の問題がからんでくるので、やはりハイパーキューブ構造のデータ・パスを使わなければならない。システム・ルーチンはさほど使われることはないと思われる。そして、プログラムの最適化はやはり、通信経路を考慮したものになる。つまり、⑥になってしまうのである。

ＮＣＵＢＥのようにＭＩＭＤマシンの場合は、高速のものを目指さなければ、ハイパーキューブ構造を使う必要もなく、プログラムは比較的容易であろう。（そのかわり、データ転送量に比例して遅くなる。つまり、疎結合型並列計算機を使って、ノイマン型の計算を行わせるわけである。図３．１はＭＩＭＤ型並列計算器を用いた最も単純な並列化の例を示している。）しかしながら、ＣＭのようにＳＩＭＤマシンの場合は、最悪の場合、パソコンより遅くなりうる。（たとえばＣＭの各ノードが演算を行う際に他のノードのデータが必要な場合、必要なノードとの同期が必要となる。同期をとる間は、ＭＩＭＤマシンとは違って、そのノードは完全な待ち状態に陥るわけであるから、このことはプログラムによってはＣＭの稼働状態にあるノード数が全体の数百分の一、数千分の一になってしまい、実行時間がパソコンよりも遅くなってしまうことを意味する。図３．２はＳＩＭＤ型並列計算機を用いた単純な並列化の例を示している。この例でわかるように、通信に何らかの構造を使わないと、大部分のノードは通信待ち状態ばかりで、ほとんど計算する時間がなくなってしまう。）つまり、疎結合型並列計算機のインプリメントは、とりあえず動くようになってからチューンするという考え方はまったく通用しないのである。そして、特に新しいアルゴリズムのインプリメントについては、そのアルゴリズムが疎結合型並列計算機のアーキテクチャを考慮して作られているものでなければ、とりあえず動くかしてからといった安易な戦略は意味をなさないのである。

このように、新しいアルゴリズムの疎結合型並列計算機へのインプリメントは、密結合型並列計算機のようにチューニング・レベルの考慮はできず、むしろ、それ自身が一つの研究分野であると言えよう。そしてそのようなコンピュータ・サイエンスにおける新しい研究は、高価な疎結合型並列計算機を所有している研究所が行ってしかるべきものではないだろうか。

３．２　視聴覚認知機構の解明を行う際のインプリメントの問題

本研究所の主要テーマは、人間の視聴覚情報処理や認知・行動の仕組みを解明し、工学的に応用しようと言うものである。このような研究においては、実際にニューラルネット等の構築原理を導出し、シミュレーションにより検証を行い、その応用可能性を示さなければならない。この計算機へのインプリメントを行う際に、疎結合型並列計算機が魅力的である理由は以下のとうりである。

①ターン・アラウンド・タイムの短さ

通常、我々は図３．３で示すように、何らかのアイデアをもとにモデルを考え、シミュレーションを繰り返してアルゴリズムを修正し、結論を得る。このシミュレーションが１回につき数週間もかかっていたのでは、十分なシミュレーションはむずかしい。例えば、手書き漢字認識の研究では、数千コネクションに及ぶニューラルネットを様々な条件を変えつつ数百回も実験を行い、その結果学習則の改善を行うことができた。これは、NCUBE や Alliant のような高速の計算機を用いたために可能となったことである。そして、もしスーパーコンピュータ、密結合型並列計算機、疎結合型並列計算機の３種類が同程度のマシン・パフォーマンスを持つとすると、コスト・パフォーマンスの最もすぐれる疎結合型並列計算機が魅力的であるのは言うまでもないだろう。

②実時間処理の可能性

目的とするアルゴリズムを一度ハイパーキューブ構造を使用した並列アルゴリズムにデコンポーズ(*)することにより、近い将来、大幅なスピードアップの期待できる疎結合並列計算機上で、人間の視聴覚情報処理を実時間でシミュレーションすることも可能である。

③チューリング・マシンの束縛から解放されたアルゴリズムの必要性

ニューラルネットや Early Vision の特に応用的側面からの研究開発の最も大きな原動力は並列分散処理メカニズムの魅力である。これまでの工学的情報処理方式がノイマン型計算機アーキテクチャの影響を受け逐次処理に偏っていたため、人間の持つ並列分散情報処理をうまくモデル化できていなかった可能性もいなめない。疎結合型並列計算機へのインプリメントは、その本来の並列分散処理能力により、既存技術からの飛躍的発展が期待できるものと思われる。逆に言えば、我々は視聴覚認知機構の計算理論を暗中模索しているわけだが、ノイマン型アーキテクチャのもとで研究を進める限り、ノイマン型では解決不可能な問題（ＮＰ完全問題）は初めから無視しなければならない。あるいは、現在のノイマン型計算機が数千倍速くなるのを数十年指をくわえて待っていなければならない。（速くなるという保証もないのに）はたしてこのような束縛条件の中で、脳の並列分散メカニズムが解明されるのだろうか？

３．３　ダイナミック・ロード・バランシングの問題

計算アルゴリズムには、大別して、データに依存するものとしないものがある。例えば画像や音声処理の各種フィルタはデータに依存しない場合が多い。あるいは、各種行列計算などもデータに依存しない。これら、データに依存しない問題を以下スタティックな問題と呼ぶ。次に、データに依存する問題であるが、例えば有限要素法、境界値問題などを使って解く構造解析の問題（特に流体など）が有名である。あるいはＡＩの世界でよく用いられる Data Driven 手法(*)などもデータに依存する問題である。最近の話で言うと、ニューラルネットの学習も広い意味でデータに依存する。これらデータに依存する問題を以下ダイナミックな問題と呼ぶ。

並列計算機において各ＣＰＵに効率よくタスクを処理させなければ全体のパフォーマンスが上がらないことは自明のことであるが、タスクの各ＣＰＵへの均等な負荷分散のことを以下ロード・バランシングと呼ぶ。

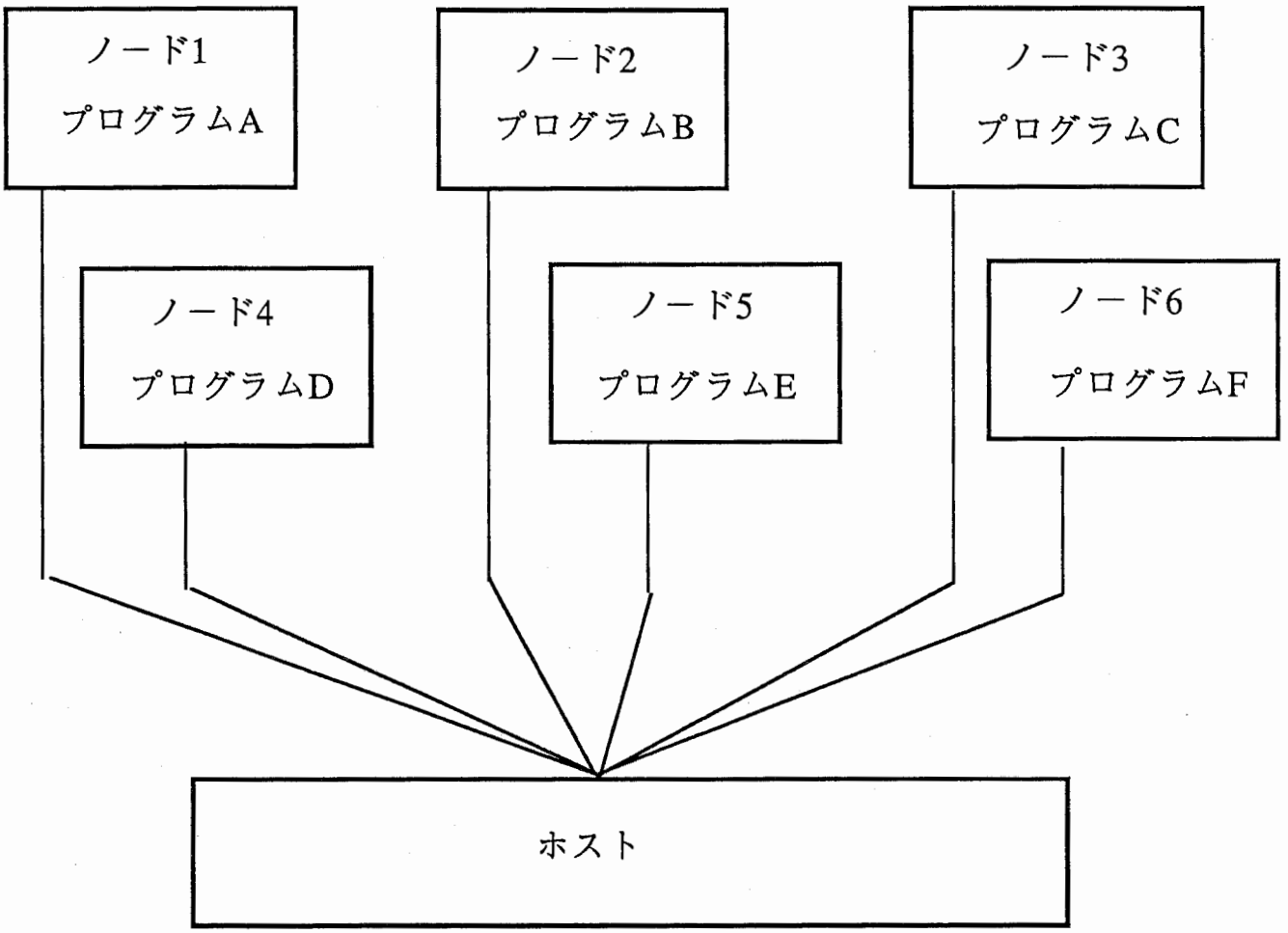
さて、コンピュータ・サイエンスはその歴史を振り返るまでもなく、スタティックな問題からダイナミックな問題へとターゲットが変わってきている。むしろスタティックな問題については研究し尽くされた感もある。これらの研究は当然チューリング・マシン・モデルの上で研究され、インプリメントされてきた。そして今、疎結合型並列計算機ではどうだろうか。現状は、スタティックな問題のデータをいかにしてハイパーキューブ・トポロジーにマップするかという検証が行われつつあるところで、ダイナミックな問題についてはこれからといったところであろう。従って、並列計算機論の研究において今最も必要とされている基礎研究は、ハイパーキューブ・トポロジーを使ったダイナミック・ロード・バランシングの研究であると言えよう。

我々は今 3.2 で述べたように解明すべき問題、モデル（アルゴリズム）を抱え

ている。そして疎結合型並列計算機も所有している。つまり、純粋なコンピュータ・サイエンスの立場からだけでなく、実際の視聴覚・認知機構の研究におけるモデルの具体化においても、ハイパーキューブ・トポロジーを利用したダイナミック・（もちろんスタティックも）・ロード・バランシングの基礎を固めておくことが必要とされているわけである。

このような研究は、視聴覚認識機構の研究を心理・生理・工学的に行う研究員が片手間に出きるものでもなければ、外注で出きるものでもない。すなわち、図4で示すように、方式やアルゴリズムを構築する研究者と、計算機理論の研究者が有機的に共同して推進すべきものである。また、そのような形態からこそ、既存技術からの飛躍的な革新がなされ、次世代計算機の原形が生まれてくるものと思われる。

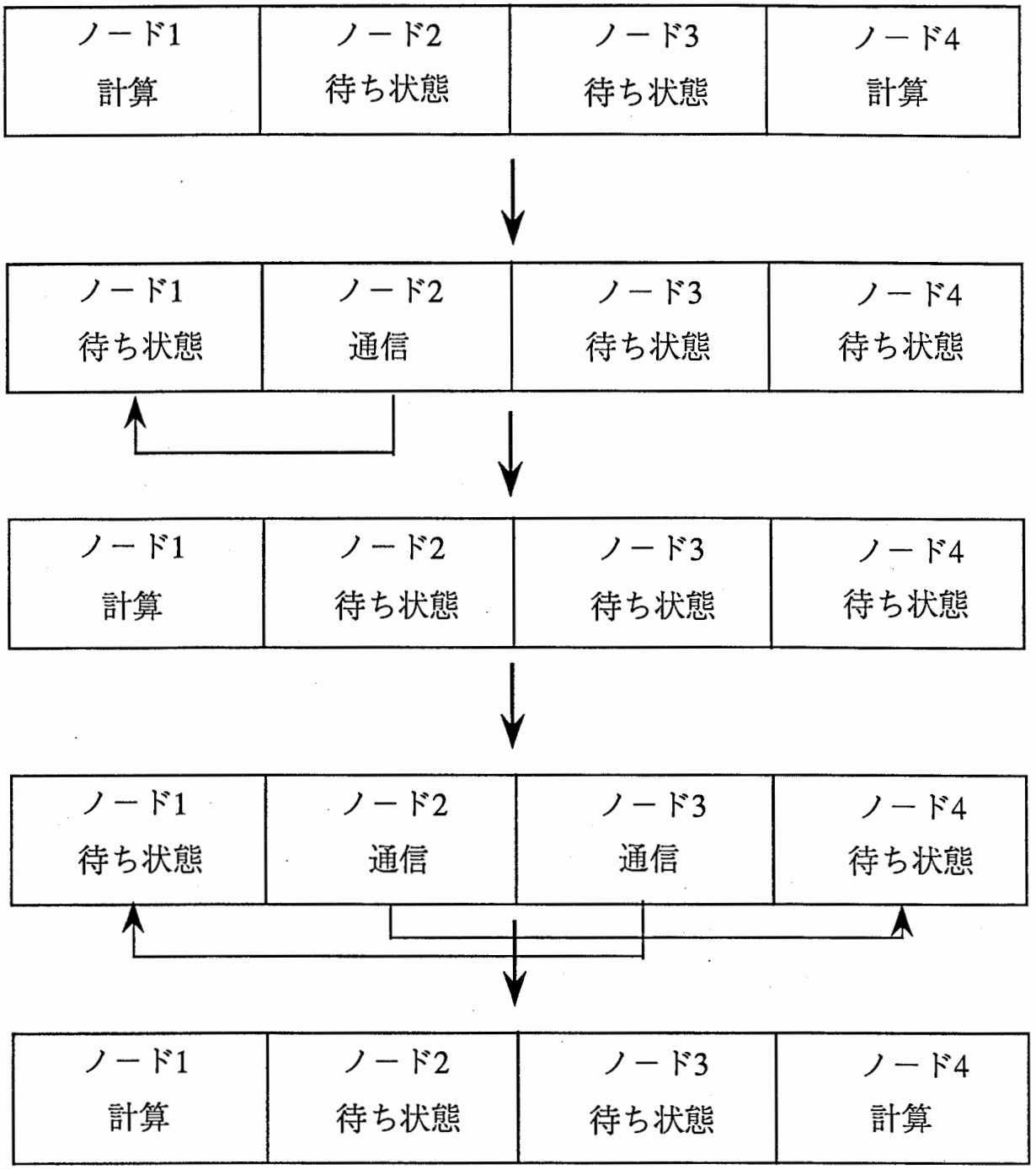
図3.1 単純な並列化の例



ノード間での通信が少ないほど有効

プログラムが簡単

図3.2 SIMDマシンでの単純な並列化



通信回数が増えるにつれて全体のパフォーマンスが低下する

すべてのノードが常に計算状態にあるときに最高性能が発揮される

図3.3 シミュレーションのターンアラウンド

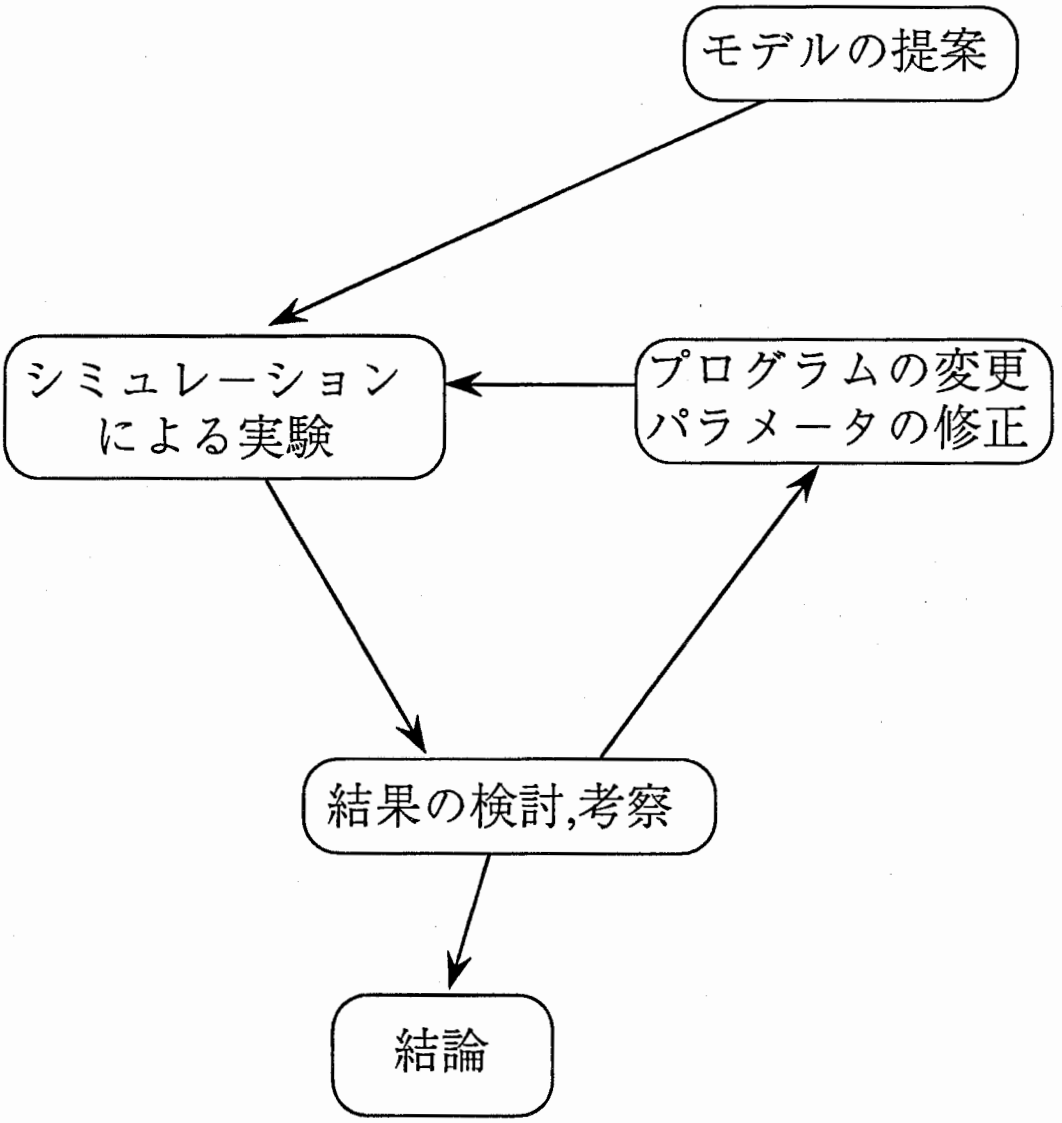
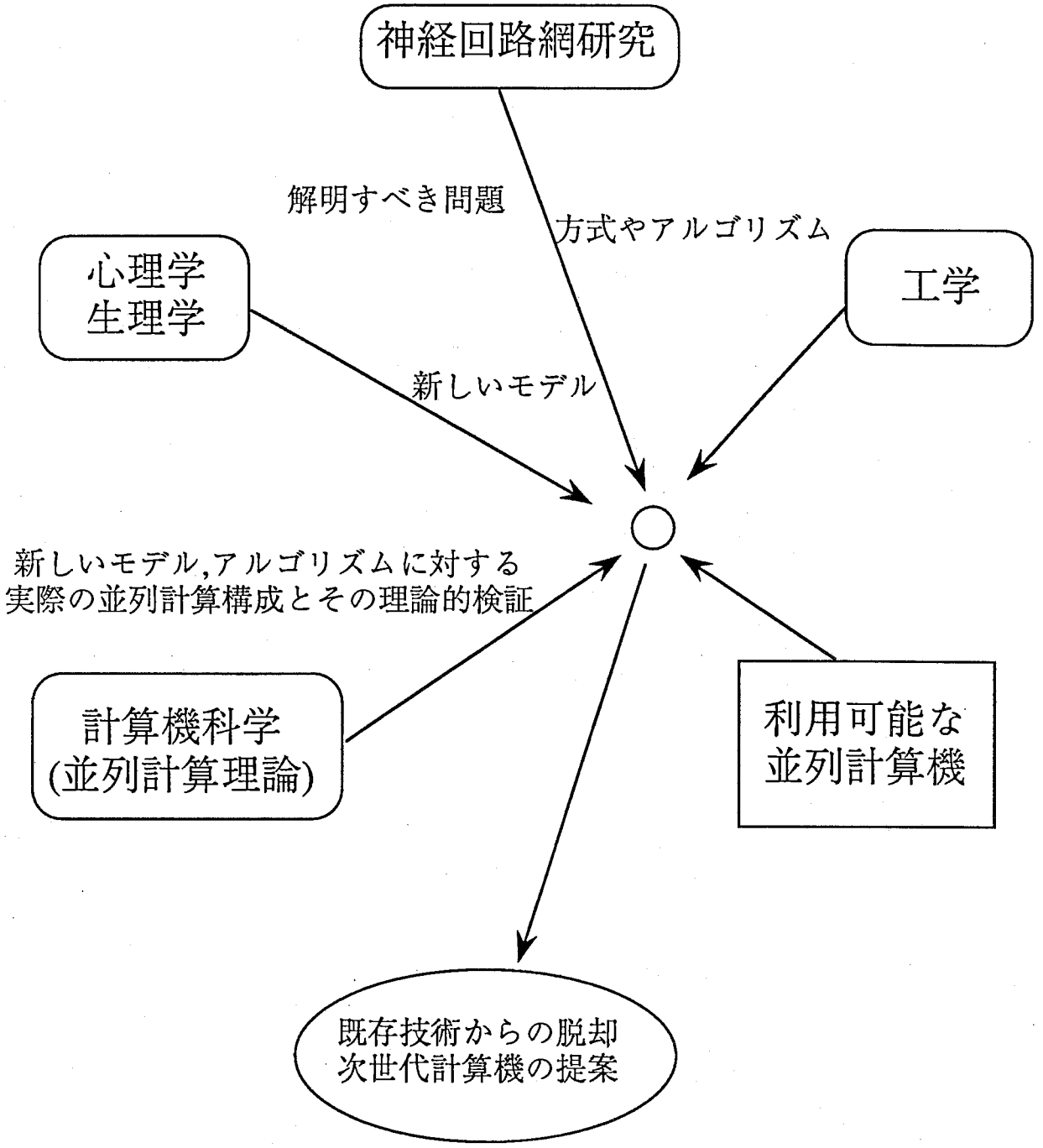


図3.4 視聴覚認知機構の研究における並列計算構成理論の役割



4．　将来進化すべき計算機アーキテクチャ

現在のノイマン型計算機信奉者は、現在の計算機が計算機のあるべき姿（すなわち人間の脳）とかけ離れているにもかかわらず、それを計算機の定義と見なし、その上だけでのシミュレーションを行い、速度・記憶容量の問題は時間が解決してくれるという立場にいるようである。これでは、我々の本来の目的である人間の脳機能の解明はむずかしいのではないだろうか。心理・生理・工学の基礎研究は常に進歩している。それに伴って、計算機に対する考え方、計算機の使い方、計算機アーキテクチャの改善されるべき方向、といったものも同時に変えて行かなければならないのではなかろうか。

この章では、とりあえず現在の技術を考えずに、脳機能の解明を行いやすいような観点から必要とされる計算機アーキテクチャについての提案を行う。

4．1　ハイパーキューブ型通信網の限界

これまでに述べてきたように、計算機はチューリング・マシンという机上のモデルから現在の並列計算機にまで進化してきている。ここで、計算機のモデルをチューリングの言うように、計算部分とデータ格納領域と見なせば、現在の密結合型並列計算機は、チューリング・マシンの純粋な実現例であり、最も進化した形のものの一つと言えるであろう。ところが、疎結合型並列計算機の場合、特に超並列の場合、データ通信部分がクローズアップされてくる。例えば、チューリング・マシンに最適なアルゴリズムが、データ通信時間のため、最適にならない場合もあるわけである。従って、前章で述べたように既存のアルゴリズムの見直しが必要とされているわけであるが、それでは、アルゴリズムの判っていない問題についてはどうだろうか。

ハイパーキューブ型通信網をもつ超並列計算機は、既存のアルゴリズムにデータ通信コストを加えることにより、新しい計算量を算出することができ、そのため、ＮＰ困難な問題も十分に多くのＣＰＵを疎結合させることにより解くことができるものと期待されている。ところが、このような形での疎結合超並列計算機の利用は、解くべき問題のアルゴリズムが決定されていて、それに応じた動的な、あるいは静的なデータ構造が判っていなければならない。つまり、非決定的な問題については、その通信網を有効利用できないわけである。

4．2　脳機能は決定的か？

我々の目的は脳機能、特に認知・理解・認識等の解明であるが、はたしてこれらは決定的な問題なのであろうか？つまり、何らかのアルゴリズムがあって、その手順を踏めば、誰の脳を使っても、反応が正確に予測できるのであろうか？

現在のところ脳機能が決定的である、すなわち、アルゴリズム記述ができるという根拠はないし、できないという根拠もない。それならば、決定的でない場合にも対応できるような計算機アーキテクチャを念頭において脳機能の解明をすべきではないだろうか。

あるいは、人間の認知過程の状態遷移は有限オートマトンである根拠がどこにあるのだろうか。現在、それらの一部に着目して有限オートマトンと見なしていても、将来、全体的なシステムとして見た場合の予測は、全く立てようがないのでは無かろうか。それらがもし、非決定性オートマトンで記述出来るなら少なく

とも現在の有限オートマトンで考えるよりも現実的なものが出てくるだろう。

このように、脳機能は非決定的であり、認知・知覚・理解・認識・学習・推論・．．．が、各個人はバラバラであるようで、全体としてカオスティックな類似性があり、それぞれの状態遷移が決定的であるかどうかわからないとすると、各々のアルゴリズムを解明するよりは、その統計的に判っている範囲での特徴を学習することにより、あたかも各個人のアルゴリズムを知っているかのような振舞いをするシステムを作る方が、魅力的であろう。もちろん、チューリング・マシン・モデルでは、すなわちノイマン型計算機では、事実上実現不可能である。

4．3　カオス・コンピュータ

前提条件：カオスは全てを含んでいる。

これまでの計算機進化の歴史は計算実行部分とデータ格納領域の改善が大部分であった。計算実行部分の高速化は先に述べたように限界があり、いずれ超並列化せざるを得なくなるであろう。そしてそれぞれの計算ユニットにデータを分散して格納することにより、計算速度に応じたデータ格納領域は確保される。残る問題はデータ通信であり、これは前章で述べたように現在のところハイパーキューブ型が有望である。このように計算機の基本構成要素を計算部・データ格納部・データ通信部の3つと考えると、超並列化により計算部、データ格納部は必要なだけ大きなシステムを構成でき、データ通信部がどのような通信（静的にも動的にも）をも可能とするような構成であれば、有望な新しい計算機モデルになると思われる。

図4．1で示すカオス・コンピュータはその新しいモデルの具体例である。カオス・ユニットは既存の計算機のローカル・メモリ及び計算ユニットに相当するもので、入力の数、場所、強さ、方向に応じて（さらに、必要な場合は内蔵されたプログラムに従った計算を行い）、出力を行う。カオス・バスはカオス・ユニットの入出力を結ぶデータ通信網であるが、既存の通信網と全く異なり、任意の2点間の通信を任意の軌道で結ぶことが出来る。

カオス・コンピュータの動作原理は次のようになる。

まず、カオス・コンピュータは原則としてプログラムは存在せず、（カオス・ユニット内のみ明示的に存在出来る。）目的とする動作ができるようになるまで学習を繰り返す。学習の目標は、カオス・バスに無限に存在する軌道の中から、目的とするタスクに最も適合するものを選ぶことである。

カオス・コンピュータ実現のためには、カオス・バスの素子、学習則等、問題点が多いが、不確定性をもつ現象の内部状態を解析することなく、情報処理として利用できるため、パターン認識をはじめとして数多くの応用分野が考えられる。

4．4　ニューロ・コンピュータ　Ｖｅｒ　Ｊ

先に述べたカオス・コンピュータは現在の技術からの隔たりが大きいと思われるので、既存技術で可能な範囲で次のようなアーキテクチャをもつ計算機を提案する。

図4．2はニューロ・コンピュータ　Ver J の概要を示している。各セル・ユニットは通常のローカル・メモリ及び計算ユニットから成り、プログラム可能である。セル間のコネクションはハイパーキューブ構造をもつ通信網である。

このニューロ・コンピュータの動作原理は次の通りである。まず、適当に各セルのポテンシャルを決め、作動を開始する。各セルは得られた入力に対して出力

を計算し、他のセルに以下のようにブロードキャストする。

（Ａ）　Inp(a,b) = k * pot(a) / dist(a,b)

〔 Inp(a,b) はセルｂに対するセルａからの入力。
pot(a) はセルａの内部ポテンシャル。
dist(a,b) はセルａとセルｂの間の距離で、ハイパーキューブ通信におけるルーティングの回数。
ｋは係数。〕

もしくは

（Ｂ）　Inp(a,b,t) = k * pot(a,t-dist(a,b))

〔 Inp(a,b,t) はセルｂに対するセルａからの時刻ｔにおける入力。
pot(a,t) はセルａの時刻ｔにおける内部ポテンシャル。
dist(a,b) はセルａとセルｂの間の距離で、ハイパーキューブ通信におけるルーティングの回数。
ｋは係数。〕

通常、学習は適当なエラー評価を定め、コネクションの重みを変更してゆくわけだが、本提案ではコネクションはデータ通信網自体であり、重みの変更という概念はない。その代わりに、図４．３で示すように任意の２つのセルをそっくりスワップするという基本動作概念を導入する。そして、セルのスワップを繰り返して目的とする構造、つまりハイパーキューブ構造のネットワークを得る。

本提案の特徴は、ネットワークの構造をハイパーキューブ構造に限定し、既存のハードウェア技術を最大限に利用することである。方式（Ａ）は方式（Ｂ）を実現するための土台であり、得られるネットワーク自体には大きな特徴はない。方式（Ｂ）は、各セルの位置による時間遅れをそのままネットワーク構造に利用でき、通信時間の無駄が全くないシステムになる。

ニューロ・コンピュータ Ver.J は、データ通信網の問題をネットワーク構造がハイパーキューブ型であるものという条件をつけることにより緩和することを目的としたアーキテクチャを持ち、時系列を扱うネットワーク等、多くの応用分野が考えられる。

図４．１　　カオスコンピュータ

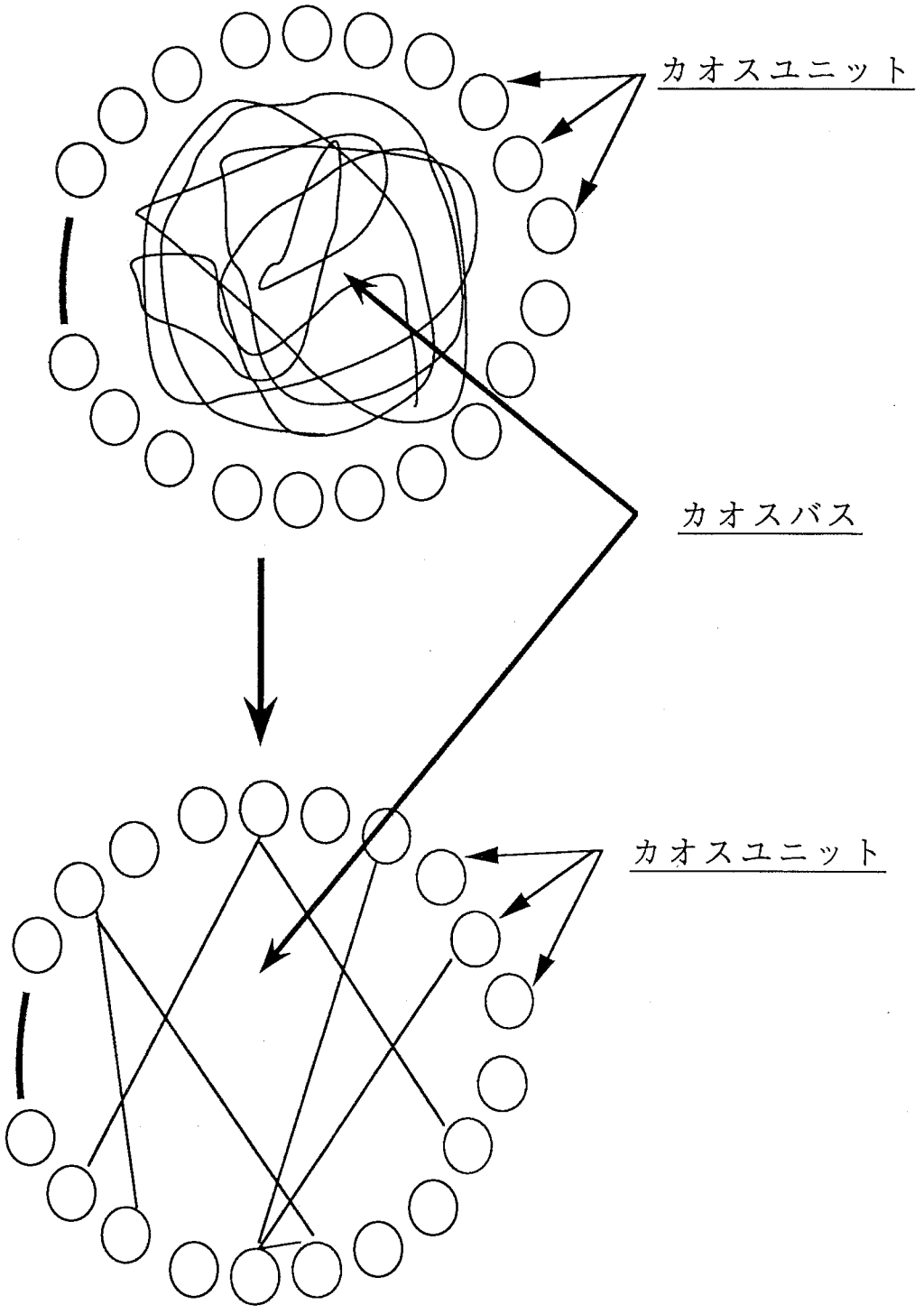


図４．２　　ニューロコンピュータ
Ｖｅｒ　Ｊ

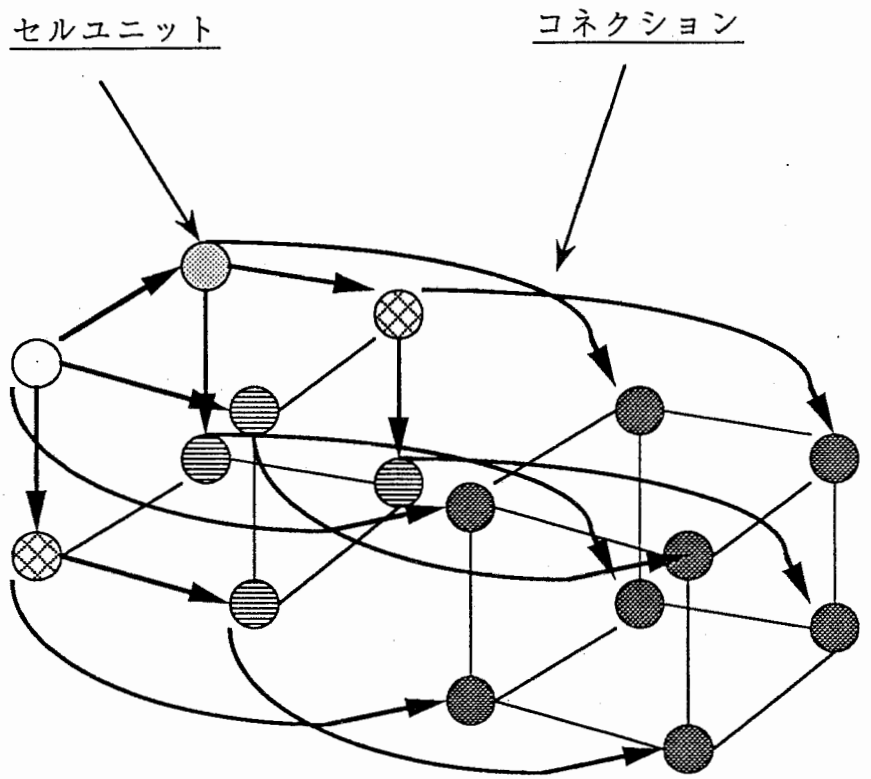
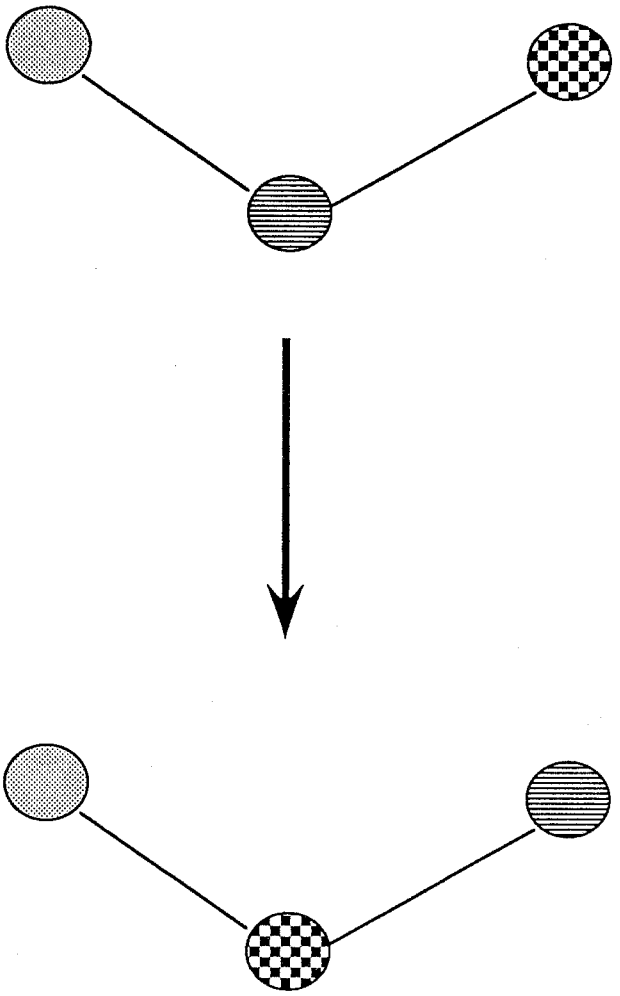


図４．３　　セルの入れ換え



5. おわりに

本稿では現在のノイマン型計算機についてのレビューを行った後、その問題点を述べ、早急に対応すべき問題として、疎結合型超並列計算機の並列計算構成理論の必要性について説明した。さらに、将来進化すべき計算機の姿としてのカオス・コンピュータ・アーキテクチャについての提案を行った。

参考文献

"NCUBE Handbook", NCUBE cop.

Aho,Hopcroft,Ullmann, "The design and analysis of computer algorithms", Addison-Wesley, Reading, Mass. (1974)

Djkstra, "Hierarchical ordering of sequential processes", Acta Information, 1,115-138 (1971)

Fox, "Square Matrix decomposition: Symmetric, Local, Scattered", 1984,C3P-97

Fox, "Load balancing and sparse matrix vector multiplication on the hypercube", 1986, C3P-288

Fox, Otto, "Concurrent computation and the theory of complex systems", 1986, HCCA1

Pineda, "Generalization of Back-Propagation to Recurrent Neural Network" 1987, AMS, 59-19

Turing, "On Computable Numbers, with an Application to the Entscheidungsproblem", Proc. London Math Soc.

國藤、"知識情報処理とニューロコンピューティング"、

補足説明

・フォン・ノイマン、チューリング・マシン、オーダー、ＮＰ完全

1936年に A.M.Turing が提案したチューリング・マシンは以後のコンピュータ・サイエンスの基盤として現在に至っている。チューリング・マシンは計算部と記憶部からなる単純な理論的モデルで、処理すべき問題の計算手順（アルゴリズム）が存在すれば、有限時間で計算を終らせることができる。（たとえ１億年かかっても有限である。）このときの計算時間はアルゴリズムによって異なり、データのサイズをＮとしたときにＮの関数によってその上限を求めることができる。例えば、Ｎ、Ｎｌｏｇ Ｎ、N^2、N^3、...、2^N、などである。そして、このＮの式のことをオーダーと呼んでいる。あるアルゴリズムのオーダーがＮの他項式（$k_nN^n + k_{n-1}N^{n-1} + ... + k_0$）であらわせられるとき、そのアルゴリズムは計算量的にみて実現可能であるが、2^N 以上のオーダーの場合は計算量が爆発的に増えるため、事実上計算不可能である。解くべき問題の最良のアルゴリズムのオーダーが2^N以上である場合、その問題はＮＰ完全であると言い、チューリング・マシンで解くことは事実上不可能である。例えばトラベラーズ・セールスマン問題はＮＰ完全の有名な例である。

1940年代に入ると J.von Neumann はチューリング・マシンの実際例としての、プログラム記憶方式の逐次処理型計算機を提案し、以後の計算機の土台を作った。そして、それ以後の科学技術の発展は、ノイマン型の計算機を用いてなされてきたため、解くべき問題がＮＰ完全かどうかということが一つの研究の目安となっている。もしＮＰ完全であれば不可能であり、そうでなければチューリング・マシンを前提とした最も効率のよい計算手順（アルゴリズム）を考えればよいわけである。つまり、新しいことを研究しているはずの研究者が、自分の取り組む問題が解決可能かどうかを半世紀も前の理論にいちいち照らし合わせていることになる。人間の視聴覚認知機構の研究のような計算理論そのものを研究する際に、このような束縛があってよいものだろうか？

・ＭＩＰＳ、ＦＬＯＰＳ

ＭＩＰＳ（Mega Instruction Per Second）は１秒間に何百万回の命令を実行できるかという計算機の評価値の一つである。ところが計算機の命令には、整数演算の他に、データのロード・セーブ、分岐命令、比較命令、論理演算命令等があり、一つの命令実行にかかる時間は一定でない。これはチップを作成する時点で決まることであり、一般的な相対時間すら予測できない。そこで通常のＭＩＰＳ値は整数和算に要する時間を用いることが多い。これに対してＦＬＯＰＳ（FLoating Operation Per Second）は１秒間に何回の浮動小数点演算が実行できるかという評価値である。ＭＩＰＳとＦＬＯＰＳの間の関係はほとんどないと言ってよい。すなわち、計算機によっては１回の浮動小数点演算に要する時間で数十回の整数演算ができたり、逆に１回の整数演算に要する時間で数十回の浮動小数点演算ができたりする。スーパーコンピュータ等の性能にはＦＬＯＰＳを使うことが多いが、これは理想的な状態での理論値であり、プログラムによってはパフォー

マンスが数百分の１に低下することすらある。また、同じ浮動小数演算でも積・和・差・商によって処理時間が異なる。通常、割り算はかけ算の数十倍の処理時間を要する。

・スーパーコンピュータ、ベクトル・プロセッサ

スーパーコンピュータの定義には適切なものはなく、現在のところ数百ＭＦＬＯＰＳの数値計算能力を持つマシンをスーパーコンピュータと呼ぶことが多い。この意味ではＮＣＵＢＥやＣＭもスーパーコンピュータとなってしまうので、本提案書におけるスーパーコンピュータの定義を以下のように定める。

「ベクトル・プロセッサを利用することにより数百 MFLOPS 以上の数値計算　能力を有する逐次処理型計算機」

ベクトル・プロセッサとは、演算パイプライン(*)方式により大規模な行列計算を高速に行わせるプロセッサである。行列計算は１次元のベクトルの演算に分解され、ベクトルとベクトルもしくはベクトルとスカラーの演算に帰着される。ベクトル・プロセッサは１回に処理できるベクトルのサイズが決まっており、処理すべきベクトルのサイズが、ベクトル・プロセッサの処理できるサイズの整数倍であるときに最も効率がよくなる。例えば、ベクトル・プロセッサの処理できるサイズが２５６であるとき、(512,512) + (512,512) の行列の和計算は２＊５１２回のベクトル演算で実行できる。ところが (2,512) + (2,512) の行列の和計算は、サイズが２であるベクトルの演算を５１２回行わなければならず、前例に比べデータ量が２５６分の１になったにもかかわらず計算量は半分にしかなっていない。逆に言えば、ベクトル・プロセッサとは大規模なベクトル演算でしか威力を発揮できないと言える。

ベクトル・プロセッサ自体は並列動作するが、ベクトル演算を１つの基本命令と見なせばベクトル・プロセッサも逐次処理型計算機である。スーパーコンピュータには和計算用のベクトル・プロセッサ、積計算用のベクトル・プロセッサ、商計算用のベクトル・プロセッサ等、複数のベクトル・プロセッサが装備されている。そして、A(i) = B(i) * C(i) + D(i) のようなベクトル演算を行わせる場合、連結（チェイニング）と呼ばれる手法により、積計算と和計算が並列に行われる。その結果、積及び和計算用のベクトル・プロセッサがそれぞれ 100MFLOPS の場合、ピーク性能は 200MFLOPS ということになる。スーパーコンピュータ上でユーザ・プログラムを動かした場合、ピーク性能の１０分の１程度しか出ない場合が多いのはこういった理由による。

・パイプライン

一連の逐次処理を各ステージごとに分割して、ステージごとに別々のユニットで処理を行わせることにより、全体として並列処理を行わせる手法をパイプラインという。自動車生産ラインにおける流れ作業は典型的なパイプライン処理である。計算機におけるパイプラインには命令パイプライン、転送パイプライン、演

算パイプラインの３種類がある。このうち命令パイプラインは比較的安価かつ容易に実現できるため、現在のワークステーション・レベルの計算機でも使用されている。転送パイプラインはデータ転送の高速化のために使われる。命令パイプラインよりは高価なため、大量のデータを扱う汎用大型計算機やＣＧ専用のワークステーションで使われる。演算パイプラインは非常に高価であり、ミニ・スーパーコンピュータ以上で使われる。

・セマフォ、モニタ

複数のプロセッサがタスクを分割して１つの処理を行う場合、共有データの扱いが問題になる。例えば、プロセッサＡとＢとが変数ｘを共有していたとする。ＡとＢとがｘの値をかってに変更してしまえば不都合であるのは明白であろう。従って、ｘの値を変更するばあいには何らかの形でロックをかけなければならない。そこで次のような手順を考えてみよう。

Ａ	Ｂ
ｘにロックをかける	
	ｘのロック解除を待つ
ｘの値を変更する	
	ｘのロック解除を待つ
ｘのロックを解除する	
	Ａによって変更された ｘの値を読み込む

実は、この手順ではすぐに不都合が生じる。ＡとＢとは並列に作動しているためＡがロックをかけにいくのとまさに同時にＢがｘを読み込みにいく可能性があるからだ。セマフォやモニタは、このような場合でも数学的に不都合が生じないような証明のなされたロックの手順である。

なお、一般のデータ・ベースにおいてもこのような問題があるが、この場合、ＡとＢはプロセッサではなく、逐次処理計算機上の１つのタスクであるので、ＡとＢとの処理がまったく同時に起こることはない。

・キャッシュ・メモリ、ページ・ファイル

今日の計算機の多くは仮想記憶方式をとっている。プログラムのデータ領域は常に物理メモリ上にある必要はなく、データがアクセスされるときにページ・ファイルと呼ばれるディスク上の領域から物理メモリ上にコピーされる。物理メモリ上のデータ・アクセスとページ・ファイル上のデータ・アクセスの速さは数百倍の違いがあり、効率の悪いプログラムほどページ・ファイル上のデータ・アクセスを引き起こし、パフォーマンスが大幅に低下する。また、物理メモリ上のデータも、頻繁に使われるものはキャッシュ・メモリと呼ばれる高速メモリにコピーされ、そこでアクセスされる。このときのアクセスの速さは数十倍違う。パイプライン手法は、このキャッシュ・メモリ上にデータがあることを前提の上に設

計されているので、必要とされるデータがキャッシュ・メモリになかったり、最悪の場合、ページ・ファイル上にあったりするばあい、実行速度が大幅に低下する。ちなみに、このようなことを考慮にいれたコンパイラ・オプティマイザは現在のところ開発されていない。

・オプティマイザ

コンパイラがプログラムをコンパイルする際に、ユーザ・プログラムの不必要な命令を最適化してくれる場合がある。この最適化をしてくれるタスクをオプティマイザと呼ぶ。VAX-Fortran や VM-Fortran(IBM) などはユーザが多いためオプティマイザが十分に研究されているが、UNIX の Fortran などはオプティマイザがない。Fortran が数値計算用の言語であると言われる由縁の多くは、このオプティマイザの良さによる。

・密結合型並列計算機

複数のプロセッサによってメモリを共有する方式の並列計算機。メモリを共有するためには、各プロセッサ間に通信経路を持ち、共有メモリのアクセスのたびに他のすべてのプロセッサと通信をしなければならない。このときプロセッサの数と通信経路の総数はN^2のオーダーで増えるため、通信にかかるオーバー・ヘッドが問題となる。従って、プロセッサを数百も数千もつなげるような密結合型並列計算機は事実上不可能である。（１６台程度が限度であると言われている）現在主流となっている密結合型並列計算機の多くは、各プロセッサにベクトル・プロセッサを付加し、２重ループの外側を並列に、内側をベクトル化することにより、高いパフォーマンスを実現している。そのため、処理方式はチューリング・マシンのレベルを上回れず、単にスーパーコンピュータの廉価版でしかありえない。

・疎結合型並列計算機

複数のプロセッサがメモリを共有せずに独立に動作する方式の並列計算機。メモリの共有がないため、プロセッサ間の通信は、プログラムで明示してやる以外には必要ない。ＣＭやＮＣＵＢＥはさらにハイパーキューブ構造と呼ばれるオーダーＬｏｇＮの通信経路を持つため、数千、数万、あるいはそれ以上のプロセッサの結合が容易に行われる。このため、スーパーコンピュータで不可能なＮＰ完全の問題を、これら疎結合型並列計算機で将来解くことができるだろうと期待されている。ただ、その場合、チューリング・マシンのレベルを上回るため、既存のアルゴリズムを捨て去り、新しく開発しなければならないという問題点がある。

・デコンポーズ

１つのタスクを複数のタスクに分割する方法。密結合の場合には共有メモリのアクセスのタイミングが、疎結合の場合には通信経路が問題となる。ただし、現在の密結合型並列計算機では、共有メモリを複数のプロセッサが同時変更するようなプログラムは、コンパイラが自動並列化の対象からはずしてしまう。いずれにしても、デコンポーズはアルゴリズムを考えるのと同じくらい重要なことである。

・Data Driven

Ｌｉｓｐのような記号処理言語では、プログラムそのものもデータとなりうる。そして、あるデータに属性を定義しておいて、その属性値に応じてデータの一部であるプログラムを自動的に起動させることができる。このような手法を Data Driven と言う。

ところで、ＡＩの言語はＬｉｓｐと言われるが、Ｌｉｓｐではこのような処理が簡単にできるに過ぎないのである。例えば、Ｃ言語でも同様なことはできるし、Fortran でさえもＯＳのルーチンを利用することによりできる。つまり、ＬｉｓｐはＡＩ研究をするのに便利なのである。一方、本提案書で何回も言及している、「疎結合型並列計算機でチューリング・マシンを上回るレベルアルゴリズムが利用できる」、と言うことは、「スーパー・コンピュータや密結合型並列計算機でもできるけれど、疎結合の方が便利である」、と言うのではない。逐次処理型計算機では不可能なのである。