第 10 章
結　論

この章では，研究成果をまとめて結論をのべるとともに，今後の課題をまとめる．

10.1 研究成果

この節ではまずこの研究によってえられた成果の概要をしめし，その後，各項目について説明する．

10.1.1 研究成果の概要

この研究の目的は，Cray-1，HITAC S-810，HITAC M-680H IAP/IDP など，数値計算専用機として発展してきたパイプライン型ベクトル計算機を記号処理あるいは非数値処理に適用して応用範囲の拡大をはかるとともに，記号処理プログラムとくに論理型言語プログラムの実行の高速化をはかることであった．またこの研究においては，とくに Prolog で代表される論理型言語で記述された解探索などの AI プログラムの，パイプライン型ベクトル計算機による高速処理を可能にすることをおもな目標としてきた．この面では，Prolog にかぎらず Lisp をはじめとする他の言語で記述された AI プログラムに対しても適用することができる一般性のあるベクトル化技法すなわちベクトル処理可能なプログラムへのプログラム変換技法をめざして研究してきた．

この研究でえられた主要な成果はつぎのようにまとめることができる．

(1) ベクトル計算機の記号処理への適用可能性の実証
(2) 可変データ構造の変換にもとづくベクトル処理法開発
(3) 複雑なくりかえし制御構造のベクトル処理法開発
(4) 共有部分があるデータのベクトル処理法開発
(5) 論理型言語プログラムのベクトル化法開発

これらの各項目について，次節以降で説明する．

10.1.2 ベクトル計算機の記号処理への適用可能性の実証

現在のベクトル計算機は数値計算における汎用性を追求してきた．その結果，数値計算だけではなく，わずかなハードウェア拡張によってあるいは現在のままでも記号処理

に適したアーキテクチャをもっているとかんがえられ，ソフトウェアしだいで広範な記号処理に適用できるとかんがえられる．この事情は Connection Machine のような汎用の計算機構をそなえた SIMD 型並列計算機においてもほぼおなじだとかんがえられる．このように，ソフトウェアしだいで汎用数値計算ベクトル計算機は記号処理に適用でき，極端にいえば「ベクトル計算機は記号処理も実行できる汎用計算機になる」という命題を実証することがこの研究のひとつの目的だった．これに対してこの研究は，従来は数値計算以外にほとんどつかわれることがなかったパイプライン型ベクトル計算機が，解探索などの記号処理においても有効であることをしめした．固定長の配列と DO ループによって特徴づけられる数値計算のために開発されたベクトル計算機が，より複雑な可変データ構造とより複雑な制御構造をもつ記号処理にも使用することができ高速に実行することができることを，N クウィーン問題などにおける実測をつうじて実証した．この研究でベクトル化可能になった記号処理プログラムとしては，つぎのようなものがある．

❑ 解探索問題

N クウィーン問題のような解探索プログラム．ただし，バックトラックにもとづくかぎられた解法だけが対象となる．

❑ ある種の反復的なリスト処理

くりかえし構造の変換によってベクトル化できるリスト処理のプログラム．たとえば，リストを基本データ構造とする N クウィーン問題のプログラムなど．

❑ 複数データの登録・検索処理

複数データのハッシュ表への登録と検索，2 分木への登録と検索など．

❑ 共有部分がある複数の構造データの処理

番地計算ソート，頻度ソートなどの番地計算をともなう配列のソート・アルゴリズム．リスト，2 分木，グラフのかきかえなど．

❑ フィルタ処理

素数生成など．とくに，素数の無限列の生成のような，並列動作するフィルタ処理．

ベクトル処理が可能になった応用はいまのところおおいとはいえず，また十分なくりかえし処理がおこなわれるばあいにかぎって高速処理が可能になっただけである．この事実をかんがえれば，この研究がめざしていたように「汎用計算機」ということばをベクトル計算機に対してあたえるのは，現在の時点では過大評価であって適切でない．しかしながら，この研究によってベクトル計算機の記号処理への適用可能性をしめすという

目的は達成することができたとかんがえられる．

10.1.3 複雑なくりかえし構造の変換によるベクトル処理法

この論文においては，**くりかえし構造の変換**すなわち Fortran におけるループ構造の変換にもとづくベクトル化技法を拡張した**くりかえし構造の交換**，**くりかえし構造の 1 重化**というプログラム変換戦略をしめし，多重のくりかえしの最内側のくりかえしがベクトル処理可能でない記号処理プログラムをベクトル化するためのくりかえし構造の交換法として最大回数反復法と残存要素検出法とを提案した．また，全解探索および単解探索のプログラムをベクトル計算機で高速実行するための並列バックトラック技法を提案した．これらについて順にのべる．

❑ くりかえし構造の変換

数値計算においては大半の処理は DO ループで記述される．すなわち，くりかえし回数がループの実行開始前に確定する．また，ループ内に条件文が存在しないか，または比較的単純な条件制御がつかわれることがおおい．しかし，記号処理においてはさまざまな制御構造がつかわれる．くりかえし回数がループ実行中にきまるループ長可変ループ (**while** 型ループ)，各種の再帰よびだし，バックトラックなどがつかわれる．これらのくりかえし構造がうまくベクトル計算機であつかえなかったことが，これまでベクトル計算機の記号処理への応用，あるいは SIMD 型計算機の応用拡大をさまたげていたとかんがえられる．

これらの複雑なくりかえし制御構造をベクトル化するうえでとくに重要な技術はくりかえし構造変換法である．数値計算プログラムに関するもっとも重要なループ構造変換法は，ベクトル化の基本であるループ分散を除外すればループ交換とループ 1 重化である．これらの技法は，再帰よびだしなどのループ以外のくりかえし構造にも拡張することができ，もとのままの構造ではベクトル化できないもしくはもとのままの構造では十分な加速率がえられない記号処理プログラムをベクトル化可能にしたり加速率をたかめたりするためのくりかえし構造変換戦略として重要である．それらの戦略とは，**くりかえし構造の交換**と**くりかえし構造の 1 重化**である．

最内側くりかえしが可変長でしかもその最大値がわからないばあいは，くりかえし回数を決定する部分を分離してスカラ処理する以外に適当なベクトル化方法は存在しないが，2 重のくりかえし構造の外側が固定長のばあいあるいは最内側くりかえしの最大値がわかっているばあいは，**最大回数反復法**または**残存要素検出法**によってベクトル化することができる．これらの方法は，**while** 型のループにかぎらず，再帰よびだしやバックトラックで形成されたくりかえし構造にも適用することができる．

❑ 解探索のベクトル処理法

解探索をベクトル処理するために，**完全 OR ベクトル計算法**および**並列バックトラック技法**を提案した．完全 OR ベクトル計算法は，逐次処理による解探索でバックトラックをくりかえしながらおこなわれる計算を，すべての解候補をベクトルの要素とすることによって，解探索のベクトル処理を可能にする方法である．この方法は論理型言語における一種の OR 並列計算をおこなうところから，OR ベクトル計算法という名称をつけている．また，並列バックトラック技法は，完全 OR ベクトル計算法において問題になる記憶消費量の爆発すなわちベクトルの要素数がおおきくなりすぎるという問題を，ベクトルの分割と一種のバックトラック処理によって解決するための技法である．並列処理とバックトラック処理とをくみあわせたことから，この技法を並列バックトラック技法とよんでいる．これらの方法によって，N クウィーン問題をはじめとする解探索問題をベクトル計算機をつかってとくことが可能になった．

10.1.4 可変データ構造の変換にもとづくベクトル処理法

この論文においては，ポインタを使用したリストなどの可変構造データのデータ構造変換によるベクトル化法すなわちプログラムの変換方法を提案し，そのベクトル化法とベクトル計算機のリスト・ベクトル処理機能などにもとづく記号処理プログラムのベクトル処理法を提案した．そして，これらのデータ構造のベクトル処理のために，3 種の条件処理機能とともに，第 2 世代以降のベクトル計算機がもっているリスト・ベクトル処理機能がとくに重要であることを指摘した．また，変換後のプログラムにおけるデータ構造として，とくにマルチ・ベクトルが重要であることを指摘した．

この論文で提案したベクトル記号処理法はつぎのようにまとめることができる．

❑ リスト・ベクトル処理にもとづくポインタ・データ処理

くりかえし構造の交換をおこなったばあいには，線形リスト，木，グラフのようなポインタをつかったデータ構造は，インデクスまたはポインタを要素とするベクトルすなわちインデクス・ベクトルに変換される．そして，プログラム変換後のプログラムにおいては，ポインタをたぐる操作はベクトル計算機がもつリスト・ベクトル処理機能を使用することによって高速に実行される．ベクトル計算機における記号処理の性能をきめるもっとも重要な機能がこのリスト・ベクトル処理機能である[注1]．

❑ 3 つの条件制御法にもとづく可変長データ処理

可変長データ処理のためには，条件制御が必要である．この論文においては，可変長データ処理法としてマスク演算方式，インデクス方式および圧縮方式を提案した．これらは，“日本製スーパコンピュータ” において実現されている条件制御の各方式

[注1] したがって，リスト・ベクトル処理機能をもたないベクトル計算機においては，ベクトル記号処理の可能性はいちじるしくひくいといわざるをえない．

[Kamiya 83] を可変長データ処理に応用したものである．これらのうちのいずれがより適しているかは，ばあいによる．たとえば，OR ベクトル計算法による *N* クウィーン問題のプログラムの S-810 における実行に関するかぎりは，これらの可変長データ処理方式の間に有意な性能差はみられなかった．一方，配列線形検索においてはマスク演算方式にもとづく方式がよりよい性能をしめした．M-680H IDP においてはインデクス方式をサポートするハードウェアがあたえられているため，それが有利である．

❑ **ベクトル処理による動的記憶管理**

記号処理においては，リスト処理における cons 操作 (リスト・セルの生成) で代表される動的記憶わりあてが重要である．この論文においては，ベクトル処理によって複数の記憶要素の動的記憶わりあてを実行する方法をしめした．

データ構造をベクトルに変換しても，結果としてえられるベクトルは数値計算におけるようにベクトル長が一定ではなく，ひとつの処理をへるたびにベクトル長が変化するばあいがおおい．ベクトル長が変化するたびにベクトル全体をつくりかえることによるベクトル再生成オーバヘッドをさけるには，可変長ベクトルを連結したデータ構造である**マルチ・ベクトル**を使用すればよいことをしめした．マルチ・ベクトルはまた，バックトラックや並列処理における処理単位をきめるという点でも重要である．

エイト・クウィーン問題のような解探索のプログラムや並列論理型言語による素数生成プログラムのようなばあいには，計算がすすむにつれて部分ベクトル長が短縮する[注2]．この部分ベクトル長短縮による性能低下をさける方法として**部分ベクトル併合法**を提案し，素数生成においてその効果をたしかめた．

10.1.5 共有部分があるデータのベクトル処理法

この論文においては，共有部分がある複数の (または 1 つの) データの更新に関するくりかえし処理 (すなわちループや再帰よびだし) をベクトル化するための**上書きラベル・フィルタ法**を提案して，そのアルゴリズムをしめすとともに，そのいくつかの重要な性質をしめした．また，複数データのハッシング，番地計算ソートなどの応用において上書きラベル・フィルタ法の有効性を実証した．上書きラベル・フィルタ法によってベクトル処理すると，衝突検出という，ベクトル処理にともなうオーバヘッドが生じるが，衝突が十分すくなければスカラ処理にくらべて処理全体で 5 〜 11 倍程度加速されうることがしめされた．また上書きラベル・フィルタ法は，各プロセス (単位処理) が 1 個ずつの共有データ (正確には共有されている可能性があるデータ) を処理するばあいだけではなく，各プロセスが複数個の共有データを処理するばあいにも拡張できることをしめし

[注2] この論文でのべたベクトル処理方法では，一般に，併合をおこなわないかぎりはマルチ・ベクトルの部分ベクトル長は一定であるかまたは短縮する．

た．

10.1.6 論理型言語プログラムのベクトル化法

この論文においては，OR ベクトル化にもとづいて逐次論理型言語プログラムを自動的にベクトル化するための方法を，非常に限定された範囲のプログラムに対してではあるが，しめした．そして，OR ベクトル化にもとづく自動ベクトル化と実行をおこなう処理系を S-810 上に試作して，*N* クウィーン問題のプログラムなどにおいて高速化がはかられることを実証した．また，並列論理型言語で記述されたプログラムの手動変換による実行結果もしめした．ここでは開発した逐次論理型言語処理系を中心として論理型言語プログラムのベクトル化法および実行方法についてまとめ，最後に並列論理型言語のばあいの補足をする．

ベクトル計算機のための論理型言語処理系の処理手順においてもっとも重要なのはベクトル化であり，その対象言語としてのベクトル中間語の設計も非常に重要である．試作処理系においては中間語として論理型言語を採用した．ベクトル化以外の重要な機能としては，決定性判定，決定化，制御構造変換，データフロー解析などのステップがある．

ベクトル化後のプログラムの実行法に関していえば，完全 OR ベクトル処理方式と並列バックトラック方式とがあるが，これらに関しては，解探索などに関係してすでにのべた．論理型言語プログラムのベクトル処理に関して特記すべきことは，中間語プログラムを変更せずに完全 OR ベクトル処理方式と並列バックトラック方式とをきりかえる方法を開発したことである．この方法は，8.3 節でのべたように，論理型中間語を採用することにより可能になった．一方，開発された試作処理系は，完全 OR ベクトル処理方式にもとづいている．実用的なベクトル処理のためには並列バックトラック処理方式による実行が不可欠だが，現在のところ並列バックトラック処理方式については実行シミュレータを開発するにとどまっている．

並列論理型言語においては，ベクトル処理のための一般的なデータ構造変換法は確立されていないため，いまのところ自動ベクトル化はできない．しかし，並列論理型言語が得意とするデータのフィルタリングをおこなうプログラムにおいては，おおくのばあい，手動でマルチ・ベクトルへの変換をおこなうことによってベクトル処理が可能になるとかんがえられる．そこでわれわれは，フィルタリングのプログラムの例として，エラトステネスのふるいによる素数生成のプログラムを手動でベクトル化し，性能を測定した．性能向上のためマルチ・ベクトルの併合をとりいれたが，加速率は 1.7 倍にとどまった．十分なベクトル長がえられているのに加速率がひくいのは，スカラ処理部分のオーバヘッドがたかいためだとかんがえられる．

10.2 結論

この研究の目的は，ベクトル計算機の応用範囲を拡大することと，記号処理・論理型言語実行のベクトル計算機による高速化をはかることだった．これらに関して順にのべる．

まずベクトル計算機の応用範囲を拡大することに関していえば，いまだ実用化にはいたっていないものの，この論文で提案した可変データ構造のマルチ・ベクトルなどへの変換にもとづくベクトル処理法，並列バックトラック技法・くりかえし構造の交換や 1 重化などの複雑なくりかえし制御構造のベクトル処理法，共有部分があるデータのベクトル処理法などによって，またベクトル計算機がもつマスク演算機能をはじめとする各種の条件処理機能やリスト・ベクトル処理機能を使用することによって，これまでベクトル処理できなかったさまざまなリスト処理をはじめとする各種の記号処理をベクトル処理可能にすることができ，ある程度目的を達成することができたとかんがえられる．

応用範囲の拡大に関してさらにいえば，この研究によって**汎用数値計算ベクトル計算機は汎用記号処理計算機になる**という命題を部分的に実証することができたとかんがえている．数値計算とくらべて記号処理においては必要な機能としてもとめられるものが基本的にことなるということはなく，単に重点がことなるだけである．記号処理においては数値計算機能は相対的に重要度がひくくなり，リスト・ベクトル機能やマスク演算機能，ベクトル圧縮命令などの重要度がたかくなる．しかし，これらの機能は数値計算においても必要であり，したがって現在のベクトル計算機にはとりいれられている機能である．ただし，IAP のような汎用スカラ計算機の付加機構においては，パイプライン型ベクトル計算機におけるのとはちがって，実験対象とした記号処理においては十分な加速率をえることができなかった (第 2 章).

つぎに記号処理・論理型言語実行のベクトル計算機による高速化をはかることに関していえば，まず，さまざまな記号処理アルゴリズムから手動でベクトル処理アルゴリズムを構成し，ベクトル計算機においてたかい加速率で実行することができたという点では目的を達成することができたとかんがえられる．一方，論理型言語プログラムについては，そのごく一部が自動ベクトル化によってベクトル計算機で処理できるようになったという点では成果があったが，このプログラム変換法の適用範囲のせまさのために，実用にはほどとおいところにあり，その点においては目的はまだ達せられていないということができる．

10.3 今後の課題

各成果項目に対応させて，今後の課題を説明し，最後にそれらをまとめる．

10.3.1 複雑なくりかえし構造の変換によるベクトル処理法

複雑なくりかえし制御構造の変換に関しては，つぎのような課題がある．

まず，くりかえし構造の変換に関する最大の課題は，第 3 〜 4 章でしめした以外の種類のプログラムに関しても，自動ベクトル化が可能になるところまで変換手順をより精密にすることだとかんがえられる．また，これまでの研究では配列線形検索などの非常にかぎられた応用プログラムへの適用をこころみただけであるが，今後はほかのプログラムへも適用をはかり，どのようなばあいにいずれの方法を適用することがより適切かを決定することもひとつの課題である．

つぎに，解探索のベクトル処理に関しては，これまでに並列バックトラック技法の適用対象となったのは単純な全解探索や単解探索のプログラムであるが，この技術を実用化につなげるためには，こまかい枝刈りなどの制御を必要とする分岐限定法や各種の AI 近似最適解探索アルゴリズムのベクトル化をめざすべきだとかんがえられる．このような陽な制御をふくんだバックトラック・アルゴリズムをベクトル化するためには，かぎられた範囲の論理型言語プログラムからの変換だけでなく，C や Lisp などの手続き型言語あるいは関数型言語からの変換を可能にすることが必要だとかんがえられる．また，並列バックトラック計算法において短縮したベクトル長を増大させるための残留バックトラック方式を研究するべきだとかんがえられる．

10.3.2 可変データ構造の変換にもとづくベクトル処理法

可変データ構造の変換に関しては，つぎのような課題がある．まずマルチ・ベクトルに関しては，この報告でしめした応用以外にもマルチ・ベクトルが有効であるかどうかをたしかめること，この報告でしめした以外の形式もふくめてどの形式のマルチ・ベクトルがより有効でありハードウェア支援に値するかをたしかめることなどがあげられる．また，ベクトル記号処理においてマルチ・ベクトル以外にも有用なデータ構造が存在するかどうか，それをどのようにつかうことができるかをしらべるのも今後の課題である．

10.3.3 共有部分があるデータのベクトル処理法

共有部分があるデータのベクトル処理法に関しては，つぎのような課題がある．まずこれまでベクトル化できないとかんがえられていたさまざまなアルゴリズムに上書きラベル・フィルタ法を適用することがあげられる．とくに，木の再バランス (rebalancing) のアルゴリズムのように，ひとつのプロセスが複数のデータ要素をかきかえる処理への

適用をこころみるべきだとかんがえられる．また，第 5 章ではデッドロックをふせぐためのひとつの方法をしめしたが，そのためのよりよい方法をみつけることも今後の課題としてあげることができる．

10.3.4 論理型言語プログラムのベクトル化法

自動ベクトル化可能なプログラムの範囲を拡大することは，困難ではあるが重要な課題だとかんがえられる．自動 OR ベクトル化をめざすアプローチとしては 2 とおりがかんがえられる．

ひとつは，この論文で中心的にのべてきたボトム・アップの研究をつづけることである．すなわち，変換後のプログラムの性能を第 1 にかんがえ，つづいて性能をおとさずに適用範囲をひろげる方法をさぐるアプローチである．これはもっとも困難な課題であり，解決のためにはおおきなブレーク・スルーが必要だとかんがえられる．論理型言語のベクトル化にかぎらず，ベクトル記号処理におけるこのようなブレーク・スルーにいたるためのアプローチとして，特定の (小規模または大規模) 応用プログラムのベクトル化法に関する研究をつみかさねる地道なアプローチが重要だとかんがえられる．この研究をはじめるきっかけになったのが N クウィーン問題のベクトル化に成功したことであること，ハッシングのベクトル化法の研究が共有データのベクトル化法につながったことなどをかんがえると，このような研究をさらにべつのプログラムに関してもおこなっていくことがブレーク・スルーにつながる可能性がたかいとかんがえられる．

もうひとつは，6.7 節および第 6 章の付録でしめしたトップ・ダウンの研究である．すなわち，変換できるプログラムの範囲を第 1 にかんがえ，つづいて変換されたプログラムをいかにして最適化するかをかんがえるアプローチである．研究の対象はいささかことなるが，Nilsson のアプローチはこれにちかい．OR ベクトル化あるいは並列バックトラック化に関してはこのトップ・ダウン・アプローチはまだふかめられていないため，今後の研究に期待することができる．しかし，うめられるべきボトム・アップ・アプローチとのあいだの溝は，いまもなお非常にふかい．

自動ベクトル・コンパイラに関していえば，完全 OR ベクトル化方式ではなく並列バックトラック技法にもとづく実行系の開発がひとつの課題である．

また，AND ベクトル化に関する今後の課題としては，まず，手動ベクトル化されたプログラムにおけるオーバヘッドを減少させて，加速率の向上をはかることがあげられる．また，リストの CDR コーディングにもとづくベクトル化をさまざまなプログラムに適用をこころみ，それによって第 7 章でしめした方法の発展をはかることである．容易ではないが，このような研究をつうじて AND ベクトル化に関しても自動ベクトル化をめざすことがより究極の目的である．

10.3.5 他の課題とまとめ

この節では，前節までに分類できなかった課題をしめしたのち，課題をまとめる．

第 1 に，この論文においては実測はすべて日立製ベクトル計算機によっておこなってきた．同様の機能をもつ他社のベクトル計算機において実測をおこなうことが，この論文でしめした各種の方法の適用範囲のひろさを実証するうえで必要であろう．第 2 に，序章で課題としてとりあげた Lisp やエキスパート・シェルなどへの記号処理ベクトル化技術の適用はそのまま課題としてのこされている．第 3 に，この研究の成果を，今後しだいにパイプライン型ベクトル計算機にとってかわるとかんがえられる SIMD 型並列計算機に適用することも重要な課題である．第 4 に，この研究を発展させるなかから，ベクトル計算機や SIMD 型並列計算機に付加すべき機能をあきらかにし，そのアーキテクチャへの提案をおこなっていくことが，重要な課題のひとつであろう．

これまで，比較的こまかい課題を列挙してきたが，いうまでもなく，これらの課題はベクトル計算機の応用範囲を拡大し，記号処理・論理型言語実行のベクトル計算機による高速化というこの研究の目的に，この研究をさらに発展させることによって，ちかづくための課題である．しかし，これらの目的にちかづくためには，この研究の成果をあらたな目でみなおしてあらたなアプローチをさぐることも，また必要だとかんがえられる．