第 1 章

序　論

この章では，この論文における研究の背景と動機，研究の目的，論文の構成について順にのべる．

1.1 研究の背景と動機

この研究を開始した背景と動機はつぎの 6 点にまとめられる．

(1) 記号処理の高速化への要求のたかまり．
(2) ベクトル計算機の機能の “汎用機” 化．
(3) ベクトル計算機をふくむ “汎用” 計算機の専用計算機に対する優位性．
(4) 並列計算機より早期に並列記号処理を実用化できるという見通し．
(5) ベクトル計算機による記号処理技術の SIMD 型並列計算機への応用の可能性．
(6) 記号処理プログラムのプログラム変換による並列化の可能性．

これらの点について，それぞれ以下でくわしくのべる．

1.1.1 記号処理の高速化への要求のたかまり

数値計算においては，最初の商用パイプライン型ベクトル計算機である Cray-1 の登場以来，多少のプログラミング環境のわるさには目をつぶってでも高速性がもとめられてきた．ベクトル計算機がもつ桁違いの高性能とそれをもとめる数値シミュレーションの要求のために，数値計算用ベクトル計算機は着実に普及してきた．一方，記号処理においては，従来，数値計算ほど高速処理の要求がつよくなかったとかんがえられるが，記号処理においても高速性をもとめる応用がひらけてきた．膨大な計算時間が必要とされ，かつリアルタイム性が要求される機械翻訳はその代表といえるだろう．また，Prolog のような論理型言語や Lisp の普及により，数値計算ほどではないにせよ，これらの記号処理用言語が高速性をもとめられる応用にしだいにつかわれるようになり，その高速実行がもとめられるようになってきた．

ベクトル計算機は現在のところ，ワークステーションにくらべればもちろんのこと，汎用大型計算機にくらべてもプログラミング環境上の不便がおおい．なぜなら，ベクト

ル計算機のプログラミングには Fortran などの汎用のプログラミング言語をつかうことができるとはいっても，そのプログラミングのためには，しばしば機械指向の特殊なプログラミング技法がもとめられる．また，ベクトル計算機のコンパイル・コードは原始プログラムからはかけはなれたものになるためデバグも困難である．さらに，わりこみ処理がきめこまかくない，記憶が仮想化されていないなどの理由のために TSS での使用はできないかまたは不向きである．

このようなベクトル計算機の欠点にもかかわらず，上記のような理由によって，今後，ワークステーションや汎用機より画期的に高速なら記号処理においてもベクトル計算機の需要が生じるとかんがえられる．また，当初は実記憶でしかつかえなかったベクトル計算機にもその後仮想記憶が導入されるなど，上記の欠点の一部はすでに克服されつつあるという点もベクトル計算機の記号処理への応用にとって有利である．

1.1.2 ベクトル計算機とその機能の汎用機化

パイプライン型ベクトル計算機はもともと数値計算専用機として開発された．数値計算においては配列計算すなわちベクトル演算の高速化が非常に重要であり，それを演算器の高度なパイプライン化，ベクトル・レジスタの導入などによって実現したのが Cray-1 や CDC Cyber 205 を祖とするパイプライン型ベクトル計算機である．また，その成功に触発されて，HITAC M-180 IAP など，汎用計算機の付加機構としての内蔵型配列処理機構 (Integrated Array Processors) も開発された．これらの計算機は数値計算によくあらわれる配列計算に特化された演算装置をもっていた．

しかしパイプライン型ベクトル計算機は，数値計算むきというその基本的な特性をたもちながらも，現在ではスーパ汎用機とよんでもよいほどの多様なベクトル演算が可能な計算機に発展した．ベクトル計算機の発展過程をいわゆるスーパコンピュータと汎用機内蔵型配列処理機構とにわけて**表** 1.1 にまとめる．第 2 世代以降のベクトル計算機の演算命令のなかには，記号処理の基本演算もふくまれている．たとえば，HITAC S-800 シリーズ，富士通 VP シリーズ，日電 SX シリーズなどのベクトル計算機はいわゆるリスト・ベクトル (間接指標ベクトル) のロード命令，ストア命令をもっている．これらの命令を使用することによって，ある種の記号処理をおこなうことが可能だとかんがえられる (第 3 章参照).

これらの「汎用機」的機能は，Fortran プログラムにおいてベクトル演算が適用できる範囲を拡大するためにもうけられた．Fortran プログラムにおいては四則演算それもとくに配列要素に対する四則演算がもっとも重要だが，単純な DO ループにおけるそのベクトル化の技術が確立されると，つぎには条件文のもとにある演算や，内積・一次巡回演算 (first-order iteration) などの巡回性 (recurrence) がある変数や配列の計算，不規則に

変化する添字をもつ配列すなわち**リスト・ベクトル**のベクトル化などが高速化の目標とされた．それは，ベクトル化率を 90 % 以上の水準までたかめなければパイプライン型ベクトル計算機のもつ高速性を満足にひきだすことができないためである．これらの演算は数値計算においては最重要とはいえない演算だが，記号処理においてはもっとも重要である．すなわち，数値計算専用機が数値計算のより一層の高速化をもとめた結果として記号処理にも適用できる汎用性を獲得するにいたったということができる．

表 1.1　ベクトル計算機の分類とその汎用機化のあゆみ

大分類	小分類	例	機能[0]			
			算･論[1]	条件[2]	関係[3]	ソート
スーパコンピュータ	第 0 世代スーパコンピュータ	STAR-100 (CDC) ASC (TI)	○	△	×	×
	第 1 世代スーパコンピュータ	Cray-1 (Cray, 1976) Cyber 205 (CDC, 1980)	○	△	×	×
	第 2 世代スーパコンピュータ	S-810 (日立 [Odaka 83]) VP-200 (富士通 [Hirakuri 83]) SX-2 (日電 [Furumasa 84])	○	○	×	×
	…					
汎用計算機付加機構	第 1 世代 内蔵配列演算機構 (IAP)	M-180 IAP (日立, 1978 [Horikoshi 83]) ACOS 1000 IAP (日電 [Osaka 82])	△	△	×	×
	…					
	現世代 内蔵配列演算機構	M-680H IAP (日立, 1986) 3090 Vector Facility (IBM, 1986)	○	○	×	×
	内蔵データベース処理機構 (IDP)	M-680H IDP (日立, 1986)	○[4]	○[4]	○	○

[0] 記法について：○ 機能あり，△ 限定的な機能あり，× 機能なし．

[1] 算術・論理演算．

[2] 条件制御つき演算 (マスクつき演算)．

[3] 関係演算 (関係データベースに関する集合演算)．

[4] IAP とあわせて使用することにより，算術・論理演算，マスク演算も実行できる．

ベクトル計算機が記号処理に適用できるようになったのは，もうひとつの「汎用機化」のためでもある．もうひとつの「汎用機化」とは，主記憶スループットの増大である．すなわち，大量の配列データを高速に主記憶からロードし主記憶にストアするため，現在のパイプライン型ベクトル計算機は他にほとんど例をみないほど主記憶と処理装置間

のスループットがたかい．このようなたかい主記憶スループットは数値計算の応用においてもとめられたために実現されたものだが，記号処理は数値計算以上にその恩恵をうけることができる．なぜなら，数値計算の応用においては全計算時間のなかで当然のことながら四則演算がしめる割合がたかいが，記号処理においては主記憶間でのデータ転送がしめる割合がたかいからである．

ただし，ここでひとこと注意しておかなければならないが，汎用機化したとはいっても，くりかえし計算が存在することがベクトル・パイプラインを有効に利用するために必須であることにはかわりがない．この点は数値計算においても記号処理においてもかわらない．いいかえれば，ベクトル計算機で実行できる処理はベクトル処理にかぎられるということである．

ところで，ベクトル計算機の応用範囲拡大は，単に数値計算用ベクトル計算機の汎用化だけではなく，一方では記号処理専用のベクトル計算機の開発という形でもすすんだ．すなわち，関係データベース処理用のベクトル計算機 HITAC M-680H IDP (Integrated Database Processor) [Kojima 87, Kojima 90] などが開発された．M-680H IDP はデータベース処理のために機能が最適化された汎用計算機内蔵型のベクトル計算機である．すなわち，M-680H IDP においては関係の和，差，積などの関係演算やマージ・ソートなどをベクトル処理することができる．データベース処理用に最適化されてはいるが，他の各種の記号処理にも使用することができる機能をそなえている．たとえば，ベクトルの要素のなかである条件をみたすものを高速にフィルタする，汎用的な条件処理機能をもっている．

しかし，数値計算用ベクトル計算機の記号処理機能は，もともと数値計算を補足するために付加された機能であり，また記号処理分野ではまだ数値計算分野ほどは高速処理がもとめられていない．そのため，数値計算用ベクトル計算機は記号処理分野ではまだその能力は十分には注目されているとはいえない．また，記号処理用ベクトル計算機の応用もまだデータベース処理やソート・マージなどの一部の記号処理にかぎられている．すなわち，これらの汎用的な機能が十分にはいかされていないのが現状である．

1.1.3 “汎用” 計算機の専用計算機に対する優位性

MIT や ICOT (新世代コンピュータ技術開発機構) をはじめ，おおくの研究機関で記号処理専用のスカラ計算機の研究がさかんにおこなわれ，製品化されたものも Symbolics, Lambda，富士通の Alpha などの Lisp マシン，三菱の PSI などの論理型言語マシンなどがある．しかし，これらの高級言語マシンすなわち専用機にくらべて，RISC (Reduced Instruction Set Computer) にせよ CISC (Complex Instruction Set Computer) にせよ，またメインフレームにせよワークステーションにせよ，汎用機は専用機にくらべて現在のところ

優位にたっているといってよいだろう．すなわち，汎用の機能をもった計算機あるいはそれに付加機構をつけるかまたは小規模の改造をくわえて記号処理の機能をもたせた M-680H IDP のような計算機は，専用機に適している一部の応用をのぞいてはむしろコスト・パフォーマンスがたかいといえるだろう．そのおもな理由は，汎用機のほうが生産台数がおおく，設計にも工数がかけられるからである．また，既存のソフトウェアが使用できるという点でも有利である．上記の記号処理専用機のなかには，商用機として一時的に世界を席捲し，たかい売上を記録したものもあるが，IBM System 360 / 370，DEC VAX，SUN Workstations などの特定のアーキテクチャが何世代にもわたって市場を支配している汎用機のばあいとはちがって，2 世代にわたってその記録を持続できたものは皆無である．

おなじ理由によって，まだ応用範囲のせまい記号処理用計算機よりは，すでにはばひろい応用がひらけている (また，すでに周辺のソフトウェアが作成ずみの) 数値計算用計算機のほうが優位にたっているということがいえる．

以上のような現実をみれば，記号処理専用の高速計算機を設計してそのソフトウェアをつくるよりは，既存の数値計算用高速計算機をそのまま，あるいは改良し，そのうえに記号処理用のソフトウェアを開発するほうが魅力的なアプローチだとかんがえられる．このことは，逐次計算機だけではなくベクトル計算機や並列計算機においてもおなじだとかんがえられる．すなわち，数値計算用ベクトル計算機といまだ存在しない記号処理専用ベクトル計算機，あるいは数値計算用並列計算機と記号処理専用並列計算機を比較するかぎりは，いずれのばあいも前者のほうが優位にたつものと予想することができる．

1.1.4 並列計算機とくらべての実用化時期のはやさ

記号処理の高速化をかんがえるばあい，並列計算機の利用が重要な選択枝であることはうたがいがない．しかし，並列計算機はハードウェア，ソフトウェアともに未解決の課題がおおい．数値計算分野では，すでに出荷され，実用に供されている並列計算機もすくなくない．だが，とくに記号処理分野では，研究はさかんにおこなわれているが，商用の並列計算機が安定して供給され，かつ実用に供されるようになるにはまだ時間がかかるとかんがえられる．

一方，数値計算用のベクトル計算機は，この研究を開始した 1984 年の時点ですでに実用段階にあり，ソフトウェアはまだ改良の余地がおおきいものの，ハードウェアは成熟している．そして，上記のように記号処理用の機能ももっている．したがって，ベクトル計算機のためのソフトウェア研究はいますぐ実機をつかって実験することができ，成功すればただちに実用化につなげることができるというおおきな利点がある．

1.1.5 SIMD 型並列計算機への応用可能性

ベクトル計算機は，Flynn の分類にしたがえば一種の SIMD (Single Instruction stream Multiple Data stream) 型計算機であり，マルチ・プロセッサによる SIMD 型並列計算機との共通点もおおい．したがって，そこで開発したソフトウェア技術のおおくは，将来，SIMD 型並列計算機が実用に供されるようになったときに応用できるとかんがえられる．現在研究され製品化されている並列計算機の大半は Flynn の分類にしたがえば MIMD (Multiple Instruction stream Multiple Data stream) 型であるが，研究レベルではイリノイ大学の Illiac IV，ICL の DAP (Distributed Array Processor)，IBM の GF11 などの SIMD 並列計算機が開発されており，さらには MIT / Thinking Machines 社の Connection Machine CM-1 / CM-2 [Hillis 85, Hillis 90] のように商業的にも成功したということができる SIMD 型並列計算機もあり，その将来は有望だとかんがえられる．

SIMD 計算機が有望だとかんがえられる理由はつぎのとおりである．MIMD 型並列計算機においては，プロセッサ間の同期・通信が基本的にソフトウェアによっておこなわれれる．そのため MIMD 型並列計算機においては同期・通信のオーバヘッドがおおきく，したがって，データベース処理やエキスパート・システムなどにおける記号処理におおいとかんがえられる，粒度のこまかい並列処理には適さない．これに対して，SIMD 型並列計算機においてはプロセッサ間の同期・通信が基本的にハードウェアでおこなわれる．したがって，同期・通信を高速に，かつ再現性がたかく (比較的デバグしやすく) 信頼性がたかい方法でおこなうことができるという利点がある．

1.1.6 プログラム変換による並列化の可能性

“日本製スーパコンピュータ” を中心とするパイプライン型ベクトル計算機ハードウェアのこの研究への影響については 1.1.2 節においてすでにのべたが，これらのためのコンパイラ技術もまたこの研究の強力な背景となっている．これらのベクトル計算機に対してはほとんど唯一の言語処理系として Fortran コンパイラがサポートされているが，これらのコンパイラは単純なコンパイルをおこなうだけではなく，逐次処理のかたちで記述されたプログラムをベクトル処理可能なかたちにベクトル化する，すなわちプログラム変換する [Yasumura 87]．すなわち，標準の Fortran に並列処理記述用の特殊な構文を追加した Illiac IV の Fortran などとはちがって，これらのコンパイラにおいては，プログラマは標準 Fortran によって，したがって逐次処理のかたちでベクトル処理のためのプログラムを記述することができる．この「プログラム変換による並列化」を，この研究においても基本的な戦略としている．

Prolog などの論理型言語や Lisp などで記述された記号処理プログラムにも，Fortran によって記述された数値計算プログラムと同様に，パイプライン化あるいは並列化可能

なくりかえし計算がふくまれているばあいがあるとかんがえられる．それらの計算は，Fortran におけるように比較的容易にベクトル計算機で計算できるかたちにプログラム変換できるわけではない．その理由としてはたとえば Fortran における DO ループのかわりにより複雑なくりかえし構造または再帰よびだしで表現されていること，Fortran にはないポインタをつかったデータ構造をつかっていること，記号処理プログラムは数値計算プログラムにくらべて不均質性がたかいことなどがあげられる[注1]．しかし，そのようなばあいでも Fortran プログラムのベクトル化をさらに発展させたプログラム変換をほどこすことによって，ベクトル計算機で計算できるようにすることができるばあいがあるとかんがえられる．そして，プログラム変換の適用範囲を，あらたな技術開発によって Fortran プログラムのベクトル化を応用できる範囲からさらに拡大できれば，ベクトル計算機による記号処理を実用化することができるとかんがえられる．このプログラム変換においては，とくに，不均質な構造を均質な構造に変換することが重要だとかんがえられる．

これらの処理は，変換をおこなうまえは不均質であるため，従来はパイプライン型ベクトル計算機や SIMD 型並列計算機による SIMD 型並列処理に不適であり，したがって SIMD 型並列処理の適用範囲もせまいとかんがえられてきた．現在の並列処理の研究の大半が MIMD 型並列処理にそそがれている理由もここにあるとかんがえられる．しかし，不均質な処理をそのまま並列処理しようとする MIMD 型のアプローチは，プログラミング，デバッグ，通信オーバヘッドなどのさまざまな面で困難にぶつかっている．上記のような不均質な構造をプログラム変換によって均質にすることができれば SIMD 型並列処理に適するようになり，したがってパイプライン型ベクトル計算機や SIMD 型並列計算機の応用範囲を拡大することができ，並列処理の限界を打破することができるとかんがえられる．

[注1] この「不均質性」に関しては 1.3 節でよりくわしくのべる．

1.2 研究の目的

この研究の目的は，ベクトル計算機の応用範囲拡大および記号処理・論理型言語実行のベクトル化による高速化という 2 点にまとめることができる．

1.2.1 ベクトル計算機の応用範囲拡大

この研究により，汎用機化したベクトル計算機を，数値計算だけでなく，解探索やリスト処理をはじめとする各種の記号処理の応用にも適用することを可能にし，計算の飛躍的な高速化をはかるソフトウェア技術を開発する．この開発により，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・ベクトル処理機能が，リスト処理をはじめとする各種の記号処理において使用することができ，かつそれによって高性能がえられることを実証することを目的とする．そしてさらには，パイプライン型ベクトル計算機，内蔵型ベクトル計算機だけではなく SIMD 型並列計算機までふくめたベクトル計算機の応用範囲を拡大することを目的とする．いいかえれば，“汎用機化” したベクトル計算機が実際に記号処理にも応用できるという命題，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を目的とする．そして，この研究からのアーキテクチャへのフィードバックと応用範囲の拡大による計算需要の拡大とをつうじて，これらの計算機アーキテクチャを発展させることをめざす．当面は大規模な応用への適用はかんがえず，基礎技術の確立をめざす．

1.2.2 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

1.2.1 節でのべたことのうらがえしになるが，この研究は，高速処理への要求がたかまってきた記号処理，とくに論理型言語によって記述された記号処理プログラムの実行の高速化をベクトル計算機によって実現することを目的とする．とくに，高速化のためのユーザの負担を最小限にするために，自動ベクトル化をおこなう処理系を開発することを最終的な目的とする．すなわち，ベクトル計算機のアーキテクチャを意識せずにかかれたプログラムをベクトル計算機むきのプログラムに自動的にプログラム変換する処理系を開発する．

このような処理系においては，ベクトル計算機のための Fortran の処理系においてそうであるように，どのようなプログラムでもひとしく高速処理可能なプログラムに変換するというわけにはいかないとかんがえられる．すなわち，まずベクトル処理に適したある種のくりかえし処理があることが不可欠な条件であり[注2]，また，プログラミング・

[注2] ただし，Fortran におけるのよりは柔軟なくりかえし構造をあつかえるようにすることが必須だとかんがえられる．

スタイルに制約が課せられたり，多少の特殊なユーザ・オプションが必要とされたりすることはさけられないとかんがえられる．しかし，これらの条件や制約は，それがうまく設定できたばあいには，汎用言語をつかわずにベクトル処理プログラムを記述するばあいに必要になる条件や制約にくらべれば，むしろとるにたりないものだということができる．

また，この研究は論理型言語プログラムのベクトル化の研究において開発された技術を，Lisp やエキスパート・シェルなどをはじめとする他の言語による記号処理プログラムへも適用し，自動ベクトル化をおこなう処理系を開発するための基礎をきずくことをめざす．

1.3 論文の構成

この論文では，まず第 2 章で解探索のベクトル処理法について説明する．この論文における研究の原点は，解探索のベクトル処理法である OR ベクトル計算法と並列バックトラック計算法を開発したことである．N クウィーン問題などの解探索は，従来はバックトラックをつかって逐次計算されたが，これらのいずれかの方法によってベクトル処理することが可能になった．また，解探索のベクトル処理法はベクトル計算機の記号処理への応用のための基礎技術としてもっとも重要なもののひとつだとかんがえられる．解探索のベクトル処理法により，バックトラックによって形成される一種の不均質なくりかえし処理を，ベクトル計算機に適した均質なくりかえし処理に変換できるようになった．ここでバックトラックによって形成されるくりかえし構造は，くりかえし開始前にくりかえし回数がわかるなどの点で単純な構造をもった Fortran の DO ループにくらべればはるかに不均質性がたかい．なぜなら，どこでバックトラックが発生するかがコンパイル時には決定できないからである．

第 3 章と第 4 章とにおいては，解探索のベクトル処理に関する研究成果をもとにしておこなってきた論理型言語プログラムのベクトル化方法についてのべる．論理型言語プログラムであつかわれるもっとも重要なデータ構造はリストであり，そのベクトル処理の実現のために解探索のベクトル処理以外に，リスト処理基本演算のベクトル処理方法の開発，制御構造変換法の開発が必要だった．これらのうち，リスト処理基本演算のベクトル処理方法については第 3 章でのべる．ここでリスト処理基本演算とは，ベクトル化に不可欠なリストの分解 (car, cdr)・合成 (cons) などの基本演算のベクトル処理方法のことである．制御構造変換法については第 3 章においてのべたあと，第 4 章において一部の方法についてさらにくわしくのべる．制御構造変換法はプログラムの制御構造に 1 重化，交換などの変換をおこなうことによってベクトル化可能にする方法であり，第 1 章でのべた解探索の逐次処理法とベクトル処理法とをつなぐ方法でもある．ここでしめす制御構造変換法により，ポインタをつかってつくられた可変長のリストや木などの不均質性があるデータ構造の処理を，ベクトル計算機に適した均質な処理に変換することができるようになった[注3]．第 3 章では制御構造変換にもとづいてリスト処理をベクトル化する方法についてのべる．第 4 章では制御構造変換の一種であるくりかえし構造交換のための 2 つの方法の比較評価をおこなう．

解探索のベクトル処理および論理型言語プログラムのベクトル化においては現在のところ使用されていないが，この研究で開発されたもうひとつの重要な基礎技術として，共有部分があるデータのベクトル処理法である上書きラベル・フィルタ法がある．これ

[注3] これらのデータ構造は可変長であり，構造も一定でないばあいがあるという点で「不均質」であるといえる．

は，ハッシングのベクトル処理のこころみのなかから開発された方法である．ハッシングのベクトル処理を研究した理由は，解探索のベクトル処理において必要なばあいがある構造データの複写を効率よくおこなうために，ハッシングが必要とかんがえられたからである．構造データの効率的な複写方法はいまだ完成していない．しかしながら上書きラベル・フィルタ法の応用は，ハッシングや解探索にかぎらない．これはグラフ・DAG (Directed Acyclic Graphs) あるいはある種のリストや木などのように複数の要素からささされた共有要素をもつさまざまな不均質な構造データのベクトル処理を可能にする方法であり，ベクトル記号処理の応用範囲を拡大するうえで重要だとかんがえられる[注4]．第 5 章では，この上書きラベル・フィルタ法についてのべる．

第 6 章および第 7 章では，制御構造変換法などの基礎技術の応用としての論理型言語プログラムのベクトル化法をしめす．第 6 章では，この研究の当初の最大の目的であった解探索のベクトル処理プログラムの自動生成をめざした，OR 並列化を基本とする逐次論理型言語のベクトル化法をしめす．第 6 章でのべるベクトル化法はまだ完成されたものとはいえず，限定された範囲にしか適用されない．しかし，その限定された範囲においては形式化されている．すなわち，自動ベクトル化が可能になっている．第 7 章では AND 並列化を基本とする論理型言語のベクトル化法をしめす．この方法はいまのところ形式化されていないため，適用範囲も明確化されていないが，第 7 章では素数生成というストリーム処理の逐次論理型言語プログラムと並列論理型言語プログラムの両方のベクトル化をおこない，評価している．第 7 章では，AND 並列化をおこなうばあいのリスト処理基本演算のベクトル化法についてもふれる．

この研究の最終的な目的は自動ベクトル化をおこなう処理系の開発であるが，第 8 章では，第 7 章の方法にもとづいて試作した自動ベクトル化をおこなう論理型言語の処理系の構造と中間語の仕様についてのべる．この処理系はベクトル化できるプログラムの範囲が非常にかぎられているため実用にはならないが，論理型言語自動ベクトル化のための一歩をふみだしたものということができる．

第 9 章では，論理型言語プログラムのベクトル化の研究のなかからみいだされた，ベクトル記号処理において重要な役割をはたすデータ構造であるマルチ・ベクトルの応用，機能，操作法についてのべる．

最後に，第 10 章で上記のすべての研究成果をまとめる．

なお，この論文の第 1 章は金田 [Kanada 85] を参照してあらたに記述した．第 2 章は金田ら [Kanada 88b] を下敷きにし，鳥居ら[Torii 88a] および鳥居ら [Torii 88b] を参照して記述した．第 3 章は金田ら [Kanada 89b] を下敷きにして記述した．第 4 章はまったくあら

[注4] このばあい，各要素データは，共有されていたりされていなかったりするという点において「不均質」であるといえる．

たに記述した．第 5 章は金田ら [Kanada 90a]，金田 [Kanada 91a] を参照して記述した[注5]．第 5 章の内容をもとにした論文を投稿中 [Kanada 91b] である．第 6 章は金田ら [Kanada 89a] を下敷きにし，金田ら [Kanada 88a]，金田ら [Kanada 89c] および金田 [Kanada 87] を参照して記述した．また，第 6 章をもとにして金田 [Kanada 91c] を記述している．第 7 章は金田ら [Kanada 90b] をもとにして，大幅にかきかえた．第 8 章はあらたに記述した．第 9 章および第 10 章もまったくあらたに記述した．第 9 章に関しては，その内容をもとにして金田ら [Kanada 91b] を執筆した．ただし，既存の論文をもとにして記述した部分でも，ほとんどの図はあらたに記述した．また，あらたに記述した部分もふくめて，各部分を日立製作所中央研究所の研究報告に記載している．

[注5] 第 5 章に記述されたプログラムは金田ら [Kanada 90a] および金田 [Kanada 91a] に記述されているものとほぼおなじである．