技報

UNIVAC TECHNOLOGY REVIEW

1982年8月　第3号

日本ユニバック

コンピュータおよびコミュニケーション・システムのライフ・サイクル管理は，30年間の試行錯誤を経て成熟期を迎えようとしている．C. Hammer の**ライフ・サイクルの管理**は，ライフ・サイクルの管理の実現によってハードウェアやソフトウェアの企画・開発・管理を総体的にとらえることができ，ハードウェアおよびソフトウェアの供給者と使用者間に共生関係を生み出し，よりよい製品とサービスをもたらす，と述べている．また，精巧で細心の管理によって EDP マネジャーが入手できるオールラウンドでかつ最上の保険となりうる，とその効用を主張している．

分散型データベースへのアクセス管理方式には単一の制御プロセス（DBMS）を移動させるものと複数の DBMS によるものがある．前者はロックによって併行制御を行うもので，一般に，設計は単純であるが，DDB へのアクセスはシングル・スレッドとなる．一方，後者は複雑な設計であり，マルチ・スレッドである．R. J. Greene の**分散データベースの更新に関する新手法 CMT** は，単一の移動型 DBMS とロックを使った同時処理制御を用い，しかも DDB へのマルチ・スレッド・アクセスを可能にする新しい手法（CMT）を紹介する．なお時間駆動型の協調的スケジュール法を基礎とする CMT 手法はコミュニケーション構造の経済性，高い処理能力，柔軟性，効率の予測可能性などの利点をもつ，と解説している．

ディジタル・システムは部品が非常に多く，かつ複雑なため自動設計の道具の開発が必要となった．今日，自動設計の研究は，おもに新しいアルゴリズムや発見的方法の開発，またそれらの効率的な実行や設計・技術の変更に適応させるための修正などを取り上げている．S. Sahni と A. Bhatt の**自動設計問題の計算量**は，これら自動設計分野で発生する問題の計算量分析について述べている．問題の大部分は *NP*-難度であり，どれも答が最適解のある定った相対誤差以内にあることを保証するような，速い近似アルゴリズムが存在する見込みはない．そこで，発見的方法や“普通はうまくいく”アルゴリズムを使って，それらの問題を解決する方法を述べている．

T. J. Ostrand, M. C. Paul, E. J. Weyuker の**有限個の先読みによる正則文法の構文解析**は，一つの文法が与えられたときに，それが有限個の先読みで解析可能か否かを判定し，かつ可能な場合にはその最少先読み回数を求めるアルゴリズムを提示している．さらに，計算ずみの大きなパージング・テーブルを用いてリアルタイムに解析する方法，約 $3n$ 個の記憶域を用いる入力ストリングの長さと比例する時間をかけて解析する方法，の双方を評価している．

新しく発表された 1100/90 シリーズに用いられた LSI パッケージに対する基礎実験の一報告として，T. S. Steele の **LSI パッケージの端子と冷却**がある．一般に，一つの集積回路チップに集積されるディジタル回路量が増えると，チップ・パッケージに必要な入出力端子の数も増す．同論文は，集積量をふやしてゲート密度を高めるために，端子が満たすための条件（パッケージの型，端子ピッチおよび冷却で，トレード・オフの関係にある）を図形による視覚化された設計法を提案する．

流れの問題に対する3次元の解析に有限要素法を適用した例は少ない．これは，モデルの複雑さ，経済性，コンピュータ能力の限界などによると思われる．村上の**基本変数法による粘性流解析**は，このような状況を踏まえて，有限要素法による経済的な手法による3次元非圧縮性流体の数値実験を試み，検討を行っている．

日本文の分かち書きをコンピュータ処理を行う上で，付属語の接続辞書は単語の認定さらに文節の構造解析を行うために欠かすことができない．これまでも多くの構成法が考えられてきた．稲永，小山らの**文節構造解析のための接続辞書の構成と付属語の接続検定**は，一般に採られている接続辞書について問題点を提起し，それを解決した筆者らが用いている接続辞書についてその詳細を報告する．

J. Nelson の**米国における商用暗号化システムの標準化**は，NO. 2 で報告した後の追加情報として紹介した．

☆

論説 ライフ・サイクルの管理

Life Cycle Management

Carl Hammer

要 約 エレクトロニック・コンピュータおよびコミュニケーション・システムのライフ・サイクル管理は，30年の経験といくつかの誤ったスタートを経て成熟の段階を迎えようとしている．

この全体論的概念からエレクトロニック・ハードウェアやソフトウェア・システムの企画，実現，および管理を一貫する改良された枠組が導かれる．また，それはソフトウェアとハードウェアの信頼性や，応用の費用対効果や，急速に変化する技術のインパクトに共に関心をいだく使用者と供給者の共生関係を作り出す．

ハードウェアおよびソフトウェアの供給者と使用者間の協調の改良は，疑いもなくよりよい製品とサービスをもたらしてくれるし，同時に精巧で気を配った管理によってそれらをより知的に使用することができる．

Abstract Life Cycle Management of electronic computer/communications systems is maturing after thirty years of experimentation and some false starts. This holistic concept provides an improved framework for planning, implementation and control of electronic hardware or software systems. It effects of symbiosis between users and suppliers who are both concerned with reliability of hardware and software, cost-effectiveness of applications, and the impacts of a rapidly changing technology. Improved cooperation between hardwar/software vendors and users will undoubtedly result in significantly better product designs and services as well as their more intelligent use by sophisticated and concerned management.

1. はじめに

ライフ・サイクルは，生物学においてすでに定着した言葉であり，発生から死に至る生物の一生を表現するのに用いられる．

高名な歴史家，H.G. Wells (1866-1946)，O. Spengler (1880-1936)，A.J. Toynbee (1889-1975)，P.A. Sorokin (1889-1961) らは，社会を有機体としてとらえ，国家の起源と誕生，栄枯盛衰，崩壊の現象をライフ・サイクルの概念によって解明している．

また，最近では経営科学の分野でも電子システムの選択・設置・運用の経済性の向上を目的として，ライフ・サイクルの概念を採用している．

産業化社会における管理者は潤沢なデータを利用できるが，資源の制約と減少に対処する方法を学ばなければならない．これに反し，200年前の家内工業の経営者は，データは乏しかったが豊富な資源を有しており，経営者個人としては事業に関する十分な知識をもっていた．そして，この家内工業の経営者の後継者ともいえる小企業の自営の経営者は，経営を補佐するプロの人材を雇う方がよいと考えるようになった．やがて，スケール・メリットが追求されるようになり，コングロマリットや国際的な性格をもつ巨大産業が登場した．

極端な専門化が広くゆきわたる一方で，情報の集約化も行われ，EDP システムによって情報を共用する企業記憶 (corporate memory) が実現した．事実，今日の企業は従来推

し進めてきた集中化と逆の動きをみせており，コスト・センタあるいはプロフィット・センタの形態で組織の分散化を図っている．このため，企業活動の総和のみが年次報告に反映されるだけである．企業の構造や目標が変わると新しい部署が設けられ，古い部署は廃止される．これらは，みなライフ・サイクルとしてとらえられる．

さて，工業プロセスの基本は，Wassily Lenotieff によって投入産出マトリックスとして表現されたように，製品の製造・流通・消費にある．現在は消費財で手造りのものはほとんどなく，専用の工場設備をもつ生産ラインで製造される．そして，商品としての魅力が失われると，その生産ラインは閉鎖されるか異なる製品の生産に振り向けられる．いま，ここで，階層的なライフ・サイクルの三つの例として，あるブランドの一つの製品，ブランド，生産ラインを取り上げる．一つの製品のライフ・サイクルは，組立ラインの上で始まり，最終的には食べられたり，摩滅したり，捨てられたりして消費される．また，消費財のなかには，定期的な保守を保証しているものもあれば，不定期に修繕するものもある．

一方，生産ラインのライフ・サイクルは管理計画に始まる．そして生産ラインが建設され稼動する時点までには，実現性の検討と経済的な分析が行われる．生産ラインは，一時的に閉鎖されたり，再開されたりするため（この場合は，おそらく他の商品になることが多い），諺にいう猫（猫は七生をもつ，という）のように数多くの生命（ライフ）をもつかも知れないが，その場合でも生産ラインのライフは一つとみなす．そして，生産ラインの機械がガラクタ同然となるか，工場内から取り除かれるかすると初めて，生産ラインが死んだとみなされる．

ライフ・サイクル管理は，三つの主要な並行的機能，つまり，計画・監視・統制に重点を置いた新しいツールである．

本稿は，ライフ・サイクル管理の概念を電子システムに適用することについて，一般的に論じるものである．このためには今日の大型コンピュータが，明日のミニコンピュータとなり，大型コンピュータとミニコンピュータとの境界が消え，小型低速低価格のシステムから大型高速高価格なシステムまで，ほとんど連続的に商品が揃いつつあるといったコンピュータ産業の爆発的な成長を考慮しなければならない．そして，分散型システム，ネットワーク，データベース管理，さらにはプライバシ規制やセキュリティ要件までも考慮しなければならない．

一方，ハードウェアの能力は，容赦なく進歩するため，われわれはより複雑なアプリケーションを絶えず開発せざるをえない．さらに，ソフトウェア工学[5]と呼ばれるツールの利用によってソフトウェアの生産性を向上させる必要がある．

2. 定義と視点

ライフ・サイクル管理の概念が，プロセス制御と多くの共通点をもつという前提を認めると，ライフ・サイクル管理はプロジェクトや管理対象をトータルとしてとらえるための管理用ツールとなる．このためにライフ・サイクル管理では，管理に関する事柄のすべての局面，つまりその開始から終焉まで，プロジェクトに関連するすべてのコストを含めて取り扱う．たとえば，コストの要素として賃金・建物・設備，ソフトウェアの保守・開発などのはっきりしたコストのほかに，なんらかの理由によるプロジェクトの終了のための費用，間接経費，研究・開発などのあまりはっきりしないコストも含めねばならない．また，終了フェーズが，他のシステムの開始フェーズと同時進行することが多いことも，忘れてはならない事実である．

ライフ・サイクル管理に関する用語のより正確な定義は，いくつかの政府刊行物にみられる．連邦規則41の公共資産の契約と管理 (Public Contracts and Property Management) によると，「システムあるいは物のライフ」の定義として，システムあるいは物の設置に始まり，そのニーズがなくなったときに終わる期間で，それらは予想または予測で決まるとされる．そして，システムあるいは物のライフは，政府によって設定され，通常は提案要求書 (RFP: request for proposal) の中に記載される．システムあるいは物のライフは，設備の現実のライフとは一致しない．

また，この刊行物では設備を選択する場合の要諦として，システムの仕様を満たす能力と，購入・設置・運用に要する全コストをあげている．

このほか，ライフ・サイクルに言及した資料に管理予算局 (OMB: Office of Management and Budget) の告示 (circular) No. A-109 がある．同告示では，ライフ・サイクル・コストの定義が，主要システムの設計・開発・製造・運用・保守・支援などすべての局面で必要となる，あるいは必要となることが予想される諸コスト（直接的・間接的コスト，繰り返し発生するコスト，単発的コスト）をシステムの全予想使用可能期間を通じて合計したものとして述べられている．

なお，同告示では，ライフ・サイクル・コストとして人件費・機器購入費，レンタル費用，保守・設置・プログラミング・訓練・変換に要する費用などを考慮すべきであると指摘している．

ライフ・サイクル管理に関する政府刊行物のもう一つの例として，国防省 (DOD: Department of Diffense) の通達 (directive) 7920.1 を紹介する．同通達では，図1で示されるようにライフ・サイクル管理を6段階に分け，各段階を一連のタスクに分解し説明している．

同通達に盛り込まれたアプローチの論拠となっているのは次の事実である．第1にプロ

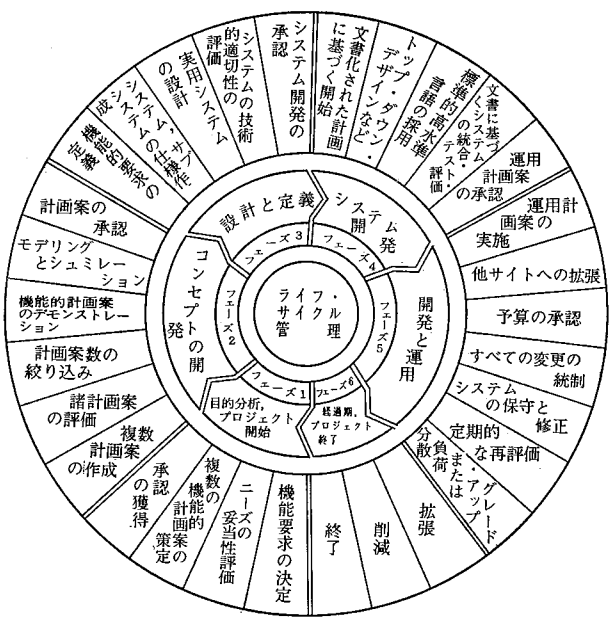


図 1 プロジェクトのライフ・サイクル管理の6段階の30のタスク

Fig. 1 Six phases and thirty tasks detailed for project Life Cycle Management

ジェクト間の同時進行性，つまり一つのライフ・サイクルの終了と他のライフ・サイクルの開始との間にオーバー・ラップがあること．第2にスケール・メリットの名目のもとで増大した階層的な複雑性から生じる問題を予期せねばならないことがあげられる．もっとも，多くの大企業は，すべての経営管理レベルと業務管理レベルを統合するシステムとして設計されたトップ・ダウン的企業経営情報システムの実現を試みる際に手痛い経験をした．実際，そのような統合の試みは，システムの複雑さが設計能力を越えているため，しばしば失敗に終わっている．

さて，ライフ・サイクル管理は，ローカル・レベルにおいて全システムから切り離された単一プロジェクトにも適用しうるし，階層を少し昇ってグローバル・レベルでの一連のプロジェクトにも適用できる．この場合，外生変数の評価が，高位レベルになればなるほど，むずかしくなるため，システムを実現し運用に成功する見込みは薄くなる．たとえば，ソ連の Gosplan は，システムの実現の最高レベルの例であるが，その可能性が明らかにされるのは次の世紀になってからであろう．

ローカルおよびグローバルな視点の差を具体的に示すために，ユーザの問題から採った次のケースをみてみよう．

まず，ローカルな視点で，ホスト・コンピュータから離れた場所に設置されるリモート・ジョブ・エントリ (RJE) 端末の購入プロジェクトを調査してみる．さてこの場合，ライフ・サイクルは計画段階から始め，提案要求の発行，業者の提案書の評価，機種の決定，必要なソフトウェアの準備，機械の設置，数年にわたるアプリケーションの拡充によるシステムの稼働へと進むことになる．ローカルな視点の場合，複雑さはなく単独にホスト・コンピュータの管理計画と無関係に進むことができる．

一方，グローバルな視点では，この RJE 端末の購入は小さな動揺あるいはシステム全体のライフ・サイクル管理計画の一つの修正と考えられる．しかし，図1の段階5でみられるように，RJE 端末の購入によって，コストの増加や，オフィス・スペースや要員の増大が引き起こされ，さらにホスト・マシンにおける負荷の増加によって，ホスト・マシンを予想したより，早く能力一杯にさせ，トータル・システムのライフ・サイクル管理計画の更新を必要とさせるかも知れない．

コストとコンピュータ資源の必要量との関係は，必ずしも線形でなく，非線形の場合にはそのタイプも十分に理解されているわけでないので，マネジメントがローカルかグローバルかのどちらの視点を採るにせよ，その意思決定について正当性をチェックしドキュメント化しなければならない．

3. モデリングとシミュレーション

過去四半世紀のオペレーションズ・リサーチ研究における最大の成果の一つにモデルの概念がある．モデル概念の EDP システムの管理への適用は，内生変数の制御能力を大幅に向上させ，同時に，多生的環境のよりよい評価の方法を学ぶことを可能にした．それゆえ，ライフ・サイクル管理がモデルの研究の発展によって大幅に受益してきたことは当然といえよう[19]．

ライフ・サイクル管理モデルは，プロジェクトの計画・監視・統制のための形式化された管理ツールである．

ライフ・サイクル管理モデルにおいては，現実の金融取引にせよ人件費・材料費，間接費・機会収益などの仮想的な取引にせよ，財務的要素に力点が置かれる[8]．このような

モデルの改良は際限がないため，現実には意思決定プロセスに強い影響を及ぼす変数のみが考慮の対象とされる．

また，必要以上に詳細に立ち入ることを避けるために，このモデルは通常トップ・ダウンで開発される．その結果，モデルは最高水準では大雑把な視点を与え，低水準では，より詳しくみられるような階層的構造をとることになろう．このアプローチは，①組織の構造を反映できる，②ノイズとなる要因を除去できる，という二つの利点をもっている．

図2はあるコンピュータ・メーカのメインフレームのライフ・サイクル・コストのモデルである[14]．このモデルは10万ドルの単一プロセッサで，約千台が生産された．図2(a)をみると，このプロセッサの設計が承認されてから3年後に出荷が開始されたことや，プロセッサの出荷が6年後にピークに達し，10年後に終了したことがわかる．またこの期間に，プロセッサの設置もピークに達している．

図2(b)は，この数100万ドルのプロジェクトで要したコストの内訳を示している．これによると，開発コストは3年後にピークを迎え，5年目にゼロとなってこの開発プロジェクトが終了したことを示している．

このほか，マーケティング・コストは2年後に現われ，5年後にピークに達し，9年目でフェイド・アウトしている．

また，図2(b)によってこのプロセッサの製造が開発の軌道に載った4年目から始まり，6年後にピークとなり，9年目に生産ラインが閉じられたときに終了したことがわか

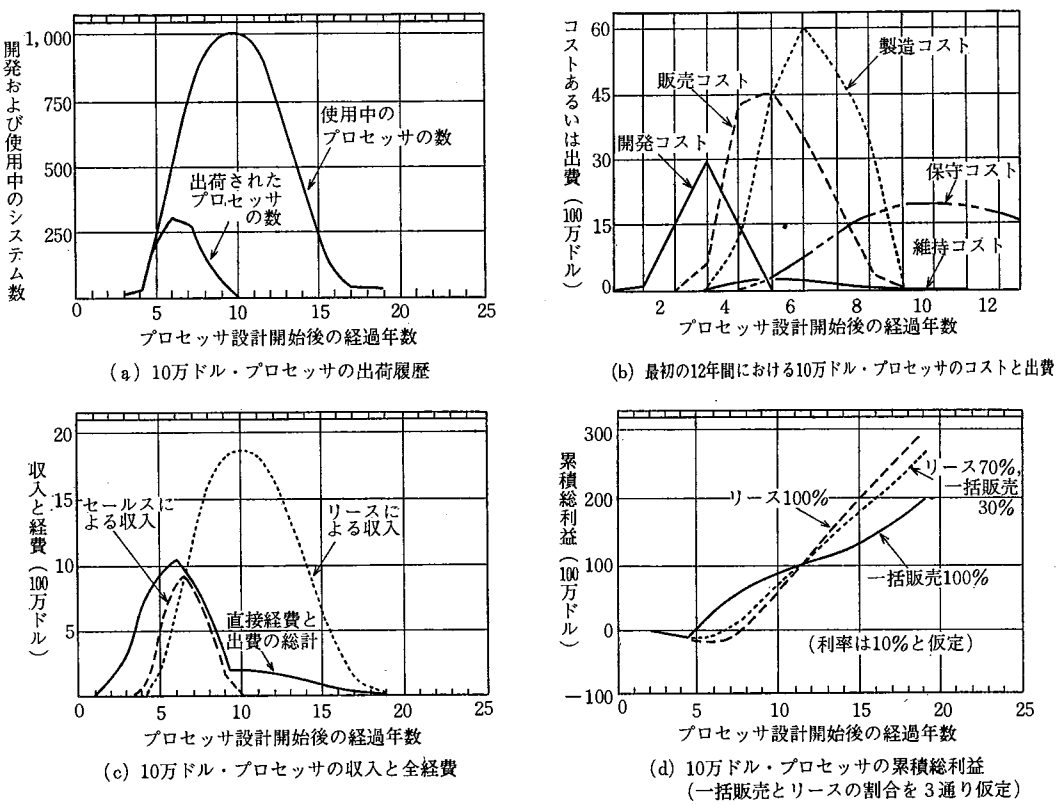


図2 ライフ・サイクル・コスト（メイン・フレーム・ベンダにおける典型的なプロセッサの開発プロジェクト例）

Fig. 2 Life cycle costs—Typical hardware processor: Mainframe vendor's view

る．そして，保守に対する出費は4年目から始まり，12年目まで延びており，ピーク時は，11年後に現れたことが示されている．なお，維持コストは，全コストの1パーセントに満たないことがわかる．また，このモデルは柔軟性を欠くため，これらの諸コストの統制に使えないし，メインフレーム・メーカの一般的な見方とも一致していない．しかし，このモデルは，一括販売あるいは，レンタルおよびリースという二つの収入源の選択を支援するデータを提供する．図2(c)は，リースと一括販売との比率が70パーセントと30パーセントという仮定のもとで，一括販売とリースから生じる全コストと両者の収入の動きを追跡している．つまり，一括販売の収入は最初に増大し，6年目でピークに達し10年目で終了する．また，リース分の収入は10年目でピークとなり，プロダクトの全寿命のつきる20年目に向かって徐々に下降し始める．

さて，支出と収入とを結合すると，最終的には，図2(d)のような累積的なキャッシュ・フロー曲線が得られる．そして，これらの情報は，マネジメントによって一括販売とリースの比率を評価するためにに利用される．資金調達のコストが，利率10パーセントと仮定すると，図2(d)にみられるように12年目に全一括販売方式と全リース方式の累積総利益が交差する点が出現する．19年目には，全リース方式では累積総利益が3億ドルとなる。一方全一括販売方式の場合は2億ドルとなる．前者は最初の7年間はキャッシュ・フローが赤字になるのに対して，後者では5年目で赤字から脱却する．これらについては，M. Phister が明快に論じている[14].

このモデルによって，財務の詳しい状況を正しく評価できる．もちろん，現実の世界では，ここで述べられたように五つのタイプのコストと二つの収入形態（一括販売とリース）だけで表現できるわけではなく，その他のデータが必要であるのはいうまでもない．

4. ユーザの視点

一般にユーザは，メインフレーム・メーカが直面するようなライフ・サイクル管理の問題に関心をもつことは少なく[21]，その関心はメーカの財務状況を知ることによって，取り引きの立場を有利にしたいという欲求が動機となっていることが多い．また，現在ではソフトウェア・コストと運用コストが，予算の大部分を占めるようになってきており，純然たるハードウェア・コストへのユーザの関心は大幅に低下した．さて，ここでは典型的なコンピュータ・システムを総体的にとらえライフ・サイクル管理の観点から考えてみよう．まず，現在円滑に稼動中のコンピュータ・システムを所有しているユーザがあると仮定した場合，ユーザがライフ・サイクル管理モデルを開発するような面倒をあえてするかどうかという疑問が生じる．しかし，次の二つの理由から間違いなく開発されるであろう．

第1に，ライフ・サイクル管理のモデリングの一部を占める現状調査の段階で混乱をほとんど惹き起さないし，さらにすべてのものがうまく管理されている（あるいは管理されていると思われる）ときは，モデルの実現に必要なコストの会計手続きの開発と実現が比較的容易なことがあげられる．第2に，これらの調査努力の結果として開発されたデータベースにマネジメントが徹底的に取り組まざるをえないため，従来，円滑に運用されていると思われてきたものが単にうわべだけであり，現実にマネジメントが注意し行動しなければならない問題のあることなどを学習できる．

おそらく，既存のコンピュータ・システムのライフ・サイクル管理モデルの開発から生じる最も興味をそそられる問題は，うそのないデータの作成と報告の必要性である．ありのままのコスト・データを十分に吟味することによって広く使用されているが，誤ってい

る戦略（たとえば創造的会計など）を看破することができる．

第3に EDP システムをこれから導入するユーザの場合である．このような場合は，むずかしいアプリケーションは他のユーザおよび組織と無関係に実現されたり，新組織や新部門が作られたりする．このようなケースでは，ライフ・サイクル管理が最も有効であり，最初から，「正しく事を行う」ということを実行する非常によいチャンスをマネジャーに与えることになる．マネジャーは，注意すべき日付やイベントを記入したスケジュール表を作成し，方法と手順を開発する時間をもち，詳細なコスト・データの計算とマネジメントの参加・承認を得ることができる．

最後に，処理量が現行のシステムの能力の限界に達しており，レベル・アップやリプレースが検討されている場合，ライフ・サイクル管理の原理は，現在まだ管理下にない要素をマネジメントが摘出するのによい助けとなる．ハードウェア機器を変更する場合，ソフトウェアについては，効率・適用範囲およびスループットの向上を実現するためにコンバージョンの代わりに新たな開発を考える方がよいかも知れない．このような検討は，バッチ方式の運用からリアルタイム方式へ移行する場合にとくに重要な点である．

さて，電子的手段によるマネジメントの革新の連続であったこの30年間に，多くのユーザは，ハードウェア・コストとソフトウェア・コストのはなはだしい較差を経験した．実際，米国情報連合 (AFIPS)[13] の最近の調査によれば，この20年間でハードウェアのコスト・パフォーマンスが，100万倍となったことが示されている．一方それに反し，プログラマの生産性は，この期間でたかだか2倍になったに過ぎない．この驚くほどの差異の一つの理由として，プログラマが従来の100万倍のマシン・タイムを無駄にしても問題にされないという事実がある．

また，ハードウェア・コストの推移をみると，1955年にハードウェアが全システム・ライフ・サイクル・コストの80パーセントを越えていたのが，1965年にはこれが50パーセントに落ち，1975年には20パーセント以下になった[20]．さらに，最近の技術予測ではこの傾向が続くとみられており，1985年頃になるとソフトウェア・コストがシステムのライフ・サイクル・コストの90パーセントを越えると予測されている．このため，われわれのライフ・サイクル管理モデルでは，ハードウェア・コストを重視していない．また，ハードウェアの保守コストも，労働集約的であるが無視できる．

他方，ソフトウェアの設計・開発・テスティングのコストは，ソフトウェア工学[5]やソフトウェア・ツール[10]の採用によって方法面では改良されたが，それでも天文学的比率に達している．

ソフトウェアのスペシァリストは，ソフトウェア・プロダクトのライフ・サイクル・コストは，二つの主要な要素に分かれると主張している[20, 22]．まず，第1の要素は初期投資であり，①問題の分析とプログラム設計，②コーディングとユニット・テスティング，③システム・テストと統合，の三段階で構成され，これらが全開発コストに占める割合はそれぞれ40，20，40パーセントとなる．そして，これらの初期投資と比較すると，さらに大きな投資がソフトウェア・プロダクトの保守のために必要となる．ソフトウェアの保守としては，バグの発見・修正，機能の改良，新規ハードウェア機能を利用するための書き直しなどが含まれる．

ソフトウェア・プロダクトにおいて，保守コストが初期投資の4～10倍に達するという事実を，マネジメントが知ると常にショックを受ける．

ソフトウェア保守によって惹き起こされる大きなコスト要因の一つは，よいドキュメン

テーションの必要性を軽視する嘆かわしい習慣に起因するものである．よいドキュメンテーションをもたずにアセンブラによって目的コードを保守することは，きわめてむずかしいし，また，高水準言語の原始コードの保守をドキュメントなしで行う場合，成功しない可能性が高い[1]．

米国の連邦政府は，全世界で最大のコンピュータ・ユーザであり，米国標準局 (National Bureau of Standards) を通じて，一連の連邦情報処理規格 (FIPS: Federal Information Processing Standards) を発行しているのは驚くに至らない．連邦情報処理規格 38 (FIPS PUB 38)[9]は，ソフトウェア・ライフ・サイクルのドキュメンテーションをとくに取り扱うものである．この規格は，ソフトウェア・プロジェクトのマネジメント・プロセスを三つのフェーズに分けている．その第1フェーズはプロジェクト開始段階であり，特定のプロジェクト計画が作成され承認される．第2フェーズは開発の段階であり，①設計案の定義，②プログラミング，③テスティングなどで構成され，第3フェーズに当たる運用段階の最初にくるプロダクトの受け入れで終わる．そして，第2フェーズでは，まず，第1に機能とデータの要件に関するドキュメンテーション，第2にデータベース，プログラムとシステム（サブシステム）の仕様のドキュメンテーションが重要であり，これらのすべてが終了するまでは一切のコーディングは行われない．第3にプログラミングが少なくともユーザ・操作・保守の三つのマニュアルによって支援されなければならない．第4に全システムおよび，構成プログラムのテスト結果の広範な報告書が用意されねばならない．

この規格は，残念ながらその完璧さにもかかわらず，当該プロジェクトのプログラマとコーダによって行われる友好的なテストを，特別に組織されたチームによる非友好的テストで補完する必要性については言及していない．

5. 例　題

ライフ・サイクル管理の概念を利用している企業は，データやその使用経験を，一般に公開していない．その理由として，そのような情報が企業の財産であり，それをもらすことによって企業競争に不利益を被るためである[7]．しかし，米国政府が，その管轄化のコンピュータ・システムの評価報告書を発刊しており，それによっていくつかの興味ある事例をみつけることができる．

さて，まず典型的なケース・スタディを表1で示す[17]．表1では，耐用年数10年の国税庁 (IRS: Internal Revenue Service) のコンピュータに対し，費用便益分析を詳細に行ったものである．

この国税管理専用システムの貸方は，12億6400万ドルにも及ぶもので，前述したように，システムのライフ・コストの20.6パーセントに当たる2億6050万ドルが基本的なハー

表1　費用便益分析：国税システム[23]

Table 1　Cost benefit analysis: Internal revenue system

種　類	調整ずみのコスト (100万ドル)	種　類	調整ずみの便益 (100万ドル)
開発費	107.0	納税管理	956.4
投資	260.5	高度なデータ処理	195.2
リースなどの諸コスト	84.5	税の戻し金の処理	164.8
ハードウェア保守費	137.9	その他	122.4
ソフトウェア保守費	121.9		
運用要員の人件費	552.5		
総経費	1,264.3	総利益	1,538.8

ドウェア・コストであった．そして，リースなどの諸コストである8450万ドルをハードウェア関連コストとして含めても27パーセントに満たず，しかもこの金額すべてが，ハードウェア・コストというわけではない．

もっとも残念ながら，ソフトウェア保守費（1億2190万ドル）とシステム開発費（1億700万ドル）については，ハードウェアとソフトウェアが一緒に集計されているため，ハードウェアとソフトウェアとの対比という形ではとらえられない．

それにしても，年間1200万ドルもの大金がソフトウェア保守費として使用されている．

結局，10年間で集計すると，人件費がハードウェアの費用よりも5億ドルにも上回ることが発見され，30年後は非常に労働集約的になるというわれわれの予想[15,16]を立証することになった．

一方，借方でみると，コンピュータ・システムは，そのコストよりも高い価値を生むということがみてとれる．納税管理の機械化は，連邦政府の金庫に年額1億ドル近い入金をもたらし，さらに高度なデータ処理によって年間2000万ドルの入金が可能である．また，納税者に対して戻し金を準備することによって，多くの誤ったファイリングを避けられるだけでなく，年間1600万ドルの利益が借方で得られる．このほか，毎年7月から12月まではコンピュータ・システムの負荷が満杯にならないため，国税庁は他の政府機関にコンピュータ資源の空き時間を売って2200万ドルの入金を得ることを予定している．また，1979年に国税庁の全税収1593億3082万9000ドルと比べると見劣りするものの，このシステムのライフ・サイクルの基調収益は年間総収入で2745億の多きにのぼっている．

もう一つの典型的なライフ・サイクル管理のケース・スタディ[17]は表2に示すもので，国防兵站局（Defence Logistic Agency）のシステムの費用便益分析の結果である．この例は，軍事および（あるいは）国防の便益をどのようにして定量化するかという興味ある問題を提示している．この例の一般会計局（GAO：General Accounting Office）の報告書では，定量的便益はゼロであると指摘しているが，ちょっと考えただけでも，かなりの量の便益があるはずである．しかし，一体，その値およびそのことはどういうものなのか不明である．また，この例でも，EDP システムが労働集約的であるという証拠をみることができる．実際に初期投資の12.8パーセントがハードウェアに，全初期投資のほぼ半分が運用コストとして使われている．そして，この初期コストのほとんどがプログラム変換に費やされている．第2世代あるいは初期の第3世代コンピュータから現在のコンピュータに切り替える場合に，変換は不可避であり，この例からそのような努力の大きさを察することができる．

表 2 費用便益分析：国防省システムの開発・実現費用（単位は1000ドル）[24]

Table 2 Cost benefit analysis: Department of defense system —Development and implementation costs (Thousands)

資本コスト		8,455.9
変換コスト		
EDP 関連給与	22,406.5	
その他の給与	23,314.1	
そ の 他	20,156.4	
合 計		65,877.0
全国防省のコスト		74,322.9
年間運用コスト		38,200.0
定量可能な便益		0

6. ソフトウェアの生産性向上のための計画

ソフトウェアが EDP システムのライフ・サイクル・コストの最重要項目であるという主張が立証されたので，つぎには当然，ソフトウェア・コストをより効率よく管理する方法，ソフトウェア・ショップの生産性を向上させる方法，ライフ・サイクル管理によってどのような危険信号が得られるかといったことに関心が向けられる．

これについて F. P. Brooks[6]は，ソフトウェア・プロジェクトの計画が，最も重要であると述べている．彼は全資源の 1/3 を計画に当てることを推奨しており，プロジェクトの資金の 1/6 がコーディング，1/4 がプログラム・モジュールのテスト，1/4 が最終システムのテストへ割り当てられるべきだといっている．そして，スケジュールが遅延しているソフトウェア・プロジェクトに対し，要員を応急的に投入することについて警告を発しており，新チーム・メンバの教育に時間をとられるため，プロジェクトの遅延をさらに増すと述べている．

また，D. S. Alberts[2]は，ソフトウェアのライフ・サイクル・コストに及ぼすプログラミングの誤りの影響を調査した．それによると，一般にプログラミングの誤りは，開発段階の初期に起こるか，ソフトウェアの変更という形で運用段階で発生する．そして，ソフトウェア・ライフ・サイクル・コストの約半分は，これら二つの段階で作られた誤りに帰因すると述べている．さらに，開発段階におけるソフトウェアの誤りレベルを引き下げることができないという事実が，M. Bendick[4] の調査によっても裏付けられている．

Bendick は，3000 から 20 万ラインのいくつかのソフトウェア・プロダクトを分析して，開発段階での誤りを運用段階で修正する場合の平均的コストは，最初に 1 ラインを書くコストの 139 倍だと述べている．

そこで，ついにわれわれは，ソフトウェア工学と総称されるツールの大きな価値を認めるに至るのである．A. L. Kustanowitz[11] は，マネジャーがソフトウェアの生産コストの監視と統制を行う方法についての優れた調査報告を発表している．その中で Kustanowitz は，「すべてのプログラミング・プロジェクトは開始して遅かれ早かれ終了する．つまり，ライフ・サイクルだけは共通である」と述べている．

しかし，「良識による管理」の原理だけを用いても，生産性の向上を支援する環境を作り出すことができる．

Kustanowitz と W. Myers[12] は，構造的プログラミング，トップ・ダウン設計と開発，HIPO，チーフ・プログラマ・チーム，開発支援ライブラリ，ウォークスルーなどの利用を奨めている．どの著者も，はっきりとはドキュメンテーションの必要性について言及していないため，誘惑に負けて，次の二つの重要な法則をあげてこの節を締めくくる．

Hammer の第 1 法則：Hammer の第 2 法則を無視する人は多い．

Hammer の第 2 法則：ドキュメント化されていないものは物の数に入らない．

7. 管理上の配慮

EDP システムのライフ・サイクル管理は，静的なプロセスではなく，一過性の事柄でもない．他の多くの管理用ツールと同様，定期的な計画の訂正と多くの不定期な更新が必要である．最初のライフ・サイクル管理計画は，プロジェクトが承認の対象になったときに立てられる．そして，その時点でライフ・サイクル管理の対象範囲と階層的レベルの限界が設定される．とくに，マネジメントは，ライフ・サイクル管理計画がどの程度ローカルかグローバルなものかを決定しなければならない．計画の基本的構造が後で容易に変更

できないため，この決定は極めて重要である．このような見解のもとで，ライフ・サイクル管理は，直接的および間接的に関連するすべてのコスト・データをとらえなければならない．

一般的なオーバーヘッド・コスト，とくにグローバルなタイプの計画のオーバーヘッド・コストは，コンセンサスのとれた方式によって割り当て，高位のマネジメントによって承認されなければならない．このような手順を踏むことによってグローバルな組織における他の実施中，あるいは計画中のライフ・サイクル管理計画との整合性を保つことができる．すべての関係する財務データを種類や型によって判定し，原価計算するためには，適当な会計の方法と手続の採用が不可欠である． また，EDP 環境における財務データの種類は，ハードウェアとソフトウェアに大別される[3]．まず，ハードウェア・コストとして，ホスト・コンピュータ，周辺装置，端末などの購入費，さらにはファシリティ管理契約によるハードウェアの保守費用も含まれる．また，ソフトウェア・コストとして，システムとアプリケーション・プログラムの開発・変換・保守の費用，さらにはシステム・アナリストの訓練の経費を含める必要がある．このほか，ソフトウェア・コストでは，他部門のグループや個人がコンピュータを使用する場合のユーザ支援コストの推定値も必要となる[18]．

計画段階における基本ルールを端的に述べると，思いつくすべての種類のコストや収入をライフ・サイクル管理と関係付け取り入れることをよく検討することにつきる．そして，これらのデータのいくつかは予測と推測値であり，内部あるいは外部の情報源とこれらの数字との関係をみて補正しなければならないものもあろう．複雑で階層的なライフ・サイクル管理システムでは，初期の推測値は確度の低いものでも必要であり，たとえ，当て推量 (guestimates) によらなければならない場合でも，後のサイクルで改良できるので，それがないよりもはるかによいことである．

8. おわりに

ライフ・サイクル管理は，適切に支援され実現されれば，非常に価値のあるツールとなる．ライフ・サイクル管理の導入は，限られた資源を最適利用するために有用であり，コンピュータ・システムのよりよき計画・監視・統制に必要なデータベースを得られるだけでも引き合うといえるし，EDP マネジャーが入手できるオール・ラウンドでかつ最上の保険であるといっても過言ではない．

（テクニカル・パブリケーション室　高橋　肇 訳）

参考文献

[1] ADP Project Management, "Department of Defense Computer Institute", Washington Navy Yard, Washington DC 20374.

[2] D. S. Alberts, "the Economics of Software Quality Assurance", *Proc. of the 1976 National Computer Conf.*, AFIPS, Vol. 45, 1976, pp. 433-442.

[3] R. Barasia and D. Klang, "The Life Cycle Cost Model", *Telesis*, Vol. 5, No. 12, December 1978, pp. 367-372.

[4] M. Bendick, "Error Rate as a Management Tool", *1976 National Computer Conf.*, Session F-5, 8 June 1976 (unpublished report)

[5] B. W. Boehm, "Software Engineering", *IEEE Trans. on Computers*, Vol. C-25, No. 12, December 1976, pp. 1226-1241.

[6] F. P. Brooks, Jr., "Mythical Man-Month", *Datamation*, Vol. 20, No. 12, December 1974, pp. 45-52.

[7] G. A. Champine, "Univac's Financial Model for Computer Development", *Datamation*,

February 1977, pp. 53-57.
[8] R. Caudill, "Understanding the Developmental Life Cycle", *Proc. of the 1977 National Computer Conf.*, AFIPS, Vol. 46, 1977, pp. 269-275.
[9] FIPS PUB 38, "Documentation of Computer Programs and Automated Data Systems", National Bureau of Standards/U. S. Department of Commerce, 15 February 1976.
[10] W. L. Frank, "The New Software Economics", *Computerworld*, 1979, Part 1 (8 January), Part 2 (15 January), Part 3 (22 January), Part 4 (29 January).
[11] A. L. Kustanowitz, "System Life Cycle Estimation", *IEEE 77 COMPSAC Proc.*, 1977, pp. 226-232.
[12] W. Myers, "The Need for Software Engineering", *Computer* (IEEE), February 1978, pp 12-26.
[13] P. S. Nyborg, P. M. McCarter and W. Erickson (eds.), *Information Processing in the United States: A Quantitative Summary*, American Federation of Information Processing Societies, Inc., 1978.
[14] M. Phister, Jr., *Data Processing Technology and Economics*, Santa Monica Publishing Company, 1976, pp. 234-236.
[15] M. Phister, Jr., "Analyzing Computer Technology Costs—Part I: Development and Manufacturing" *Computer Design*, September 1978, pp. 91-98.
[16] M. Phister, Jr., "Analyzing Computer Technology Costs—Part II: Maintenance", *Computer Design*, October 1978, pp. 109-118.
[17] H. J. Podell, (U. S. General Accounting Office, Washington DC 20548) "Life Cycle Costing of Computer Systems in the Federal Government", (Private Communication), 1978.
[18] L. H. Putnam and A. Fitzsimmons, "Estimating Software Costs", *Datamation*, Vol. 25; No. 10, September 1979, pp. 188-198; No. 11, October 1979, pp. 171-178; No. 12, November 1979, pp. 137-140.
[19] V. Rehg, "A Life Cycle System Simulation", *Proc. Fourth Annual Simulation Symposium*, Gordon and Breach, 1971.
[20] A. E. Sorkowitz (U. S. Department of Housing and Urban Development, Washington DC 20410), "Planning for Life Cycle Management" (Private Communication), April 1978.
[21] H. S. Stone, "Life-Cycle Cost Analysis of Instruction-Set Architecture Standardization for Military Computer Systems", *Computer*, (IEEE-CS), Vol. 12, No. 4, April 1979, pp. 35-47.
[22] R. H. Thayer and J. H. Lehman, "Software Engineering Project Management", Dept. of the Air Force, *Report SM-ALC/ACD TR-77-02*, McClellan Air Force Base, CA 95652, 1 November 1977.
[23] "10 Year Economic Life for the Tax Administration System" GAO Report, (LCD-76-114), 23 November 1976.
[24] "Defense Logistic Agency, Defense Integrated System, Eight Year Econ omic Life" GAO Report, (LCD-77-227), 20 December 1977.

執筆者紹介 Carl Hammer

1955年に Sperry Univac 社に入社, 1959年から1963年にかけて RCA でミニッツマン通信システムの初期設計と弾道ミサイル早期警戒システムのソフトウェア開発に従事. 1963年にコンピュータ・サイエンス担当ディレクタとして Sperry Univac 社に戻る. 1938年に Munich 大学から数理統計学で Ph. D を取得. IEEE 会員, Association for the Advancement of Science Fellow. 1970年より National Defensive Executive Reserve のメンバ. 1973年に DPMA よりコンピュータ・サイエンスの Man-of-the Year Award を受ける. 1979年に ACM より Distinguished Service Award を受ける.

報告 分散データベースの更新に関する新手法 CMT

An Alternative Approach to Distributed Database Updating

Richard J. Greene

要 約 本稿では，トランザクション処理環境で稼働する複写型の分散データベース (DDB)に対する新しい更新手法を述べる．既存のアプローチは，単一の制御プロセス (DBMS) を移動させるか複数の制御プロセスを設けて DDB へのアクセスを制御する．前者の場合では，設計は単純であり，ロックによる同時併行制御が特徴であるが，残念ながら DDB に対するアクセスは一つずつしか処理できない．後者の場合，設計および同時処理制御は一般に複雑であるが，DDB に対するアクセスは同時に複数処理される．これをマルチ・スレッド・アクセス (multi-thread access) と呼ぶ．

本稿で紹介する手法は "Cooperative Multi-Thread" (CMT) と呼ばれ，一つの移動型 DBMS とロックとを使った同時処理制御を用いている．しかも，DDB に対してはマルチ・スレッド・アクセスを許す手法である．CMT 手法の基礎は制御を移動させる時間駆動型の協調的スケジュール法であり，現存する方法に用いられている要求駆動型の競合的スケジュール法と対照的である．この手法の利点は，コミュニケーション構造の経済性，高い処理能力，柔軟性および効率の予測可能性が得られる点にある．

なお，本稿では，CMT 手法の技術面を述べると共に，量的および質的評価も行っている．

Abstract This paper presents a new updating approach for a fully redundant DDB operating in a transaction-based environment. Existing approaches utilize either a single, migrating controlling process (DBMS) or multiple controlling processes to access the DDB. Updating approaches based on a single, migrating DBMS generally exhibit simplicity of design and lock-based concurrency control. Unfortunately, access to the DDB is usually single-thread. Approaches based on multiple controlling DBMSs generally exhibit complex design and concurrency control but do permit multi-thread access to the DDB. The approach proposed in this paper, called "Cooperative Multi-Thread" (CMT), is based on a single, migrating DBMS and lock-based concurrency control and yet permits multi-thread access of the DDB. The basis of this approach is the time-driven, cooperative scheduling of control migration. Contrast this with the request-driven competitive scheduling of control migration of the existing approaches in the genre. The advantages of the proposed approach are an economical communications structure, high throughput, flexibility, and predictable performance. This paper presents the technical aspects of CMT approach as well as a quantitative and qualitative appraisal of it.

1. はじめに

分散データベースの導入は差し迫ったものになってきているが，それには二つの大きな要因が考えられる．

1) DDB システムが技術的に実現可能であること

2) 多くのエンド・ユーザの作業環境の特性と DDB システムの特性とが合致すること

コンピュータ産業において，二つの要素が DDB システムの技術的実現可能性を高めている．その第1の要素はコミュニケーション手段の処理効率が上がっていることと，コンピュータなどのハードウェアの値段が下がっていることである．この傾向は，オンライン

This article is reprinted, with permission, from AFIPS Conference Proceedings (National Computer Conference, 1981) pp. 481-485.

で集中データベースをリモート端末からアクセスすることが増加したことに表れている．手短にいえば DDB システムのハードウェア的手段は，すでに存在している．第2の要素は産業界および行政府の情報処理への要求に技術的に対処するうちに，コンピュータ業界は，これらの分野の要求への理解を広げ，深めてきた点である．すなわち，エンド・ユーザの要求がますます技術的解決策を引き出してきている．

多くの作業環境の特性と DDB システムの特性は類似しており，自然に作業環境における情報処理の問題に合致した解決技法を DDB は与える．たとえば，時を得た的確な情報が入手可能であるということは，組織の目標にとって重要なことである．

この目的に向かって，オンライン集中データベース・システムは重要な役割を演じてきた．しかしながら，データベースに対するアクセスのパターンは地理的に，そして機能的に分散されたユーザの作業環境の性質を反映する．いい換えれば，集中化では使用者の論理的情報の要求とデータ使用の物理的場所の両者を，たとえそれが実態に合致しないときでも集中する．そこで，コミュニケーションに要する費用の増加，コミュニケーションが一個所に集中することによる応答時間の遅れ，およびハードウェアの故障による悲劇的な影響が表われる．したがって，データベースが物理的に分散し組織されているのであればデータベース利用者は，よりよいサービスを受けるであろう．ここでの分散手段とは，必要であれば複写部分をもち，そのデータの利用地区により近い場所にデータを配置するということである．このアイデアから分散データベースが発展する．

DDB システムは集中データベース・システムに比べて，いくつかの重要な利点がある．すなわち，論理的データベースを分割することはデータベースの機密保護を高め，あわせてデータベースの成長を容易にする．一方，利用地区により近い場所に対象データを配置することによって，コミュニケーションの費用を少なく押えられ応答時間を短くすることができる．さらに，故障による影響も最小に押えることができるであろう．つまり，DDB システムによって，ソフトウェアの可用性，信頼性および保守性を達成するためのもう一つの技術を得ることができる．

しかし，分散データベースの利益を得ることは簡単ではない．DDP (data distributed processor) システムの抽象的モデルとそのモデルの実現との間の微妙なバランスを保たなければならないし，システム要素間の複雑な絡みが設計を混乱させるからである．

更新手法の選択は，DDB システムを構築するときの基礎となる．なぜならば，すべてのサブ・システムはこの更新手法に基づいて構築すべきであり，また，処理効率もこの更新手法に依存するからである．実際に更新方法がシステム全体に与える影響については一般にまだ経験が不足しているが，DDB システムのひな型の出現によって解消されつつある．このような専門的技術と結び付いた経験は，DDB システムの将来にしっかりした基盤を与える．

現在，多くの研究機関が新しい更新手法の開発に専念しているが，まだ手法とユーザの作業環境との関係が曖昧である．

本稿の目的は二つある．それは，発展しつつある DDB 更新手法に対する一般的骨組みを提示することと，トランザクション処理環境に対して新しく設計された手法を紹介することである．

本稿では，第1に関係する概念と専門用語について紹介し，第2に新しい手法 CMT について述べ，それを解析・評価する．最後に本稿の主眼点を検討し，将来の研究領域を提案する．

2. 基礎概念と用語

DDB システムを述べるための用語を導入する．分散処理の概念，および専門用語のより完全な説明は LeLann[2] にみられる．

DDB システムを構築している基本概念はプロセス（タスク）である． 非形式的にはプロセスは，他のプロセスと同時に実行できる計算として定義される．ユーザ・プロセス（プログラム）とシステム・プロセス（ソフトウェア）は共に記憶領域，プロセッサ時間，周辺装置および情報を消費する．各プロセスはメッセージだけで他と連絡をとる．また，プロセスはデータベース管理のような特定の機能を形成するためにグループ化することもできる．具体的には，データベース・アクセスを要求するユーザ・プロセスは間接的にデータベース管理機能に連絡し，データベース管理機能は複数のプロセスを順々に結合し要求された機能を実行することになる．

分散制御の考えによって，集中データベース・システムと DDB システムとが区別される．集中データベース・システムは唯一の全知全能な制御プロセスをデータベース管理機能の中にもち，この制御プロセスによって，論理的データベースをアクセスしているすべてのプロセスが，広域データベースを一貫した同一の視野でアクセスできるように保証している．本来の DDB システムはたとえ一つの論理的データベースであっても，複数の制御プロセスをもつ．

図 1 は，設計されつつある分散データベース更新手法の一般的な骨組みを描いたものである．

図 1 の木構造をすべて説明することは本稿の目的を超えており，それは Greene の論文[3]で述べられている． しかし，どのような更新手法でもかならず行われる，設計段階

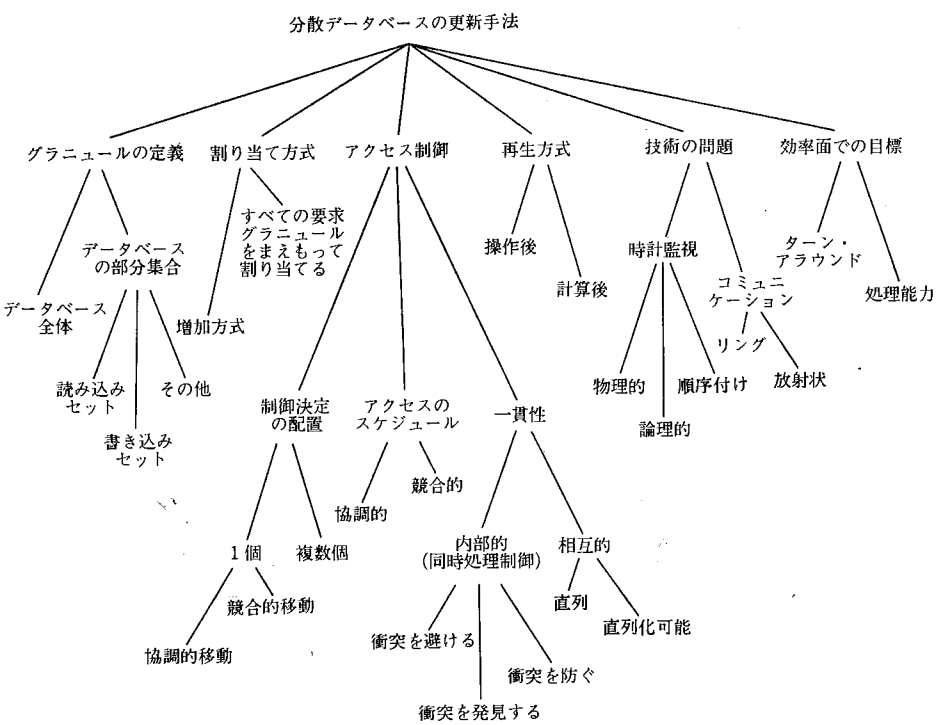


図 1 更新手法の諸要素

Fig. 1 The components of an updating approach

での主要な決定項目を要約して明らかにしておくべきだろう．一般的な骨組みの目的は，DDB 更新手法を設計するための概念設計を提供することである．一度導き出されると，その手法は DDB の共有を実現するアルゴリズムにまで詳細化される．

本質的に，データベース利用は資源の共有とみることができる．木構造のレベル 1 は，更新手法の設計を決定するキーを示している．その下位のレベルは，各々の決定の鍵となる要素を示しており，末端のノードは各決定に対する実際の選択を表す．要約すれば，設計における主要な決定は次の四つの問題を提起する．

1) 手法の効率に関する主な目標とは何か．
2) 資源のグラニュールはどのように定義されるのか．
3) 何が資源の割り当て時に制約となるのか．
4) 資源はいつ割り当てられ，いつ再生されるのか．

論理的データベースを分散させるために使われる二つの基本的方法——複写と分割によって DDB システムは区別される．複写は論理的データベースが物理的冗長さをもって分散されるのに対し，分割は逆である．実際には両者はその両極端である．しかしながら，複写の方はほとんどの分散データベースにいろいろな型で用いられており，より一般的な方法である．したがって，本稿は完全な複写の場合を仮定して話を進める．

3. CMT (Cooperative Multi-Thread)

3.1 概　略

CMT アルゴリズムの主な設計目標は単純な構造，高い処理能力および予測可能な効率である．これらを達成するために，分散データベースの制御は時分割方式で DBMS から DBMS へと協調的に渡される．この時間によって駆動する協調的な制御の移動は，CMT の特徴であり，このジャンルで他と区別される．

CMT アルゴリズムは基本的に次のような操作を行う．単一の DBMS が決められたある時間単位 1 つの DDB へのアクセスを制御する．この時間中，DBMS はマルチ・スレッド・モードで要求を処理し，グラニュールの割り当て／更新といった DDB の状況変更を反映する“更新リスト”を管理する．そして時間を使いつくしたとき，DBMS は制御指示と更新リストの両方をあらかじめ決められた後続のプロセスに渡す．

CMT アルゴリズムの割り当て方針は，あらかじめ，ユーザ・プロセスに要求されたグラニュールのすべてを割り当てることであり，これは同時処理とデッドロック防止を容易にする．

CMT アルゴリズムは支援する環境に対し，いくつかの前提を設けている．その第 1 は各 DBMS は待たされている要求のプールによって常に忙しい状態に保たれていること，第 2 は DBMS 間のコミュニケーションは論理的環状ネットワーク構造を基盤に行われていることである．

すなわち，時間駆動方式の協調的制御移動が CMT アルゴリズムの基本である．また，設計の単純さとマルチ・スレッドによる高処理能力および効率の予測可能性を CMT は提供する．その効率の予測可能性とは n 個の DBMS の各々に与えられる時間量が q 秒であるならば，各 DBMS は $(n-1)q$ 秒ごとに DDB へのアクセスを許される．

3.2 解　説

CMT アルゴリズムは二つの関連した機能を実行しなければならない．

1) 協調的制御の移動機能

2) DDB 状態の移動機能

まず，アルゴリズムを作り上げる二つの支援テーブルと四つの処理段階について説明する．

二つの支援テーブルとは制御テーブルと更新テーブルであり，上記の二つの主要機能を実行するためにアルゴリズムで必要な情報（キー）を提供する．制御テーブルとは，制御の移行順（制御サイクル）を示すもので各 DBMS に内在している．これは (i, p, s) の形の整数の組を含んでおり，i はある特定の DBMS を示す一意名であり，p と s は制御の移行順での DBMS i の前任者 DBMS p と後任者 DBMS s を示す．つまり，DBMS i は DBMS p から制御を受け DBMS s に制御を渡す．制御テーブルはシステム生成時に初期化され，不慮の事故が生じたときコミュニケーション・サブシステムによって変更されるかもしれない．第2のテーブルである更新テーブルは分散データベースの最新状態を示しており，この状態はグラニュールのロックと変更の情報から作られる．したがって，この状態を表現するために更新テーブルは二つの部分，ロック・サブリストと変更サブリストからなり，制御指示と共に DBMS に渡される．

CMT アルゴリズムは四つの段階，すなわち待ちの段階，料金計算の段階，利用の段階および終了の段階を進む．これらの段階は制御の出入を実現すると共に最新の一貫した DDB を維持する．待ちの段階は受身であり，この段階では DDB に対するユーザ要求は待たされる．たとえば，読み込み操作は利用の段階まで待たされ，書き込み操作は料金計算の段階まで待たされる．

利用の段階のために DDB を準備する料金計算の段階は制御指示と更新テーブルの到来と共に開始する．DDB を制御している DBMS は待たされていた書き込み操作を変更サブリストに付加し，それで複写を変更する．この操作により，DDB のグラニュール値は最新状態となる．つぎに DBMS はロック・サブリストを複写に適用する．この操作により，グラニュール割り当ては最新状態となる．これら二つの操作が完結したとき，DDB を制御している DBMS は排他的に最新の論理的 DDB を所有したことになる．最後に更新テーブルそれ自身が更新される．更新テーブル内の各エントリにはサブ・サイクル番号が付いている．DDB を制御している DBMS がエントリを適用したとき，エントリのサブ・サイクル番号が1加算される．そして番号が DDB システム内の DBMS の数と一致すれば，そのエントリは全 DBMS に処理されたことになる．したがって，更新テーブルから取り除くことができる．各 DBMS は制御サイクルにまたがったグラニュール・ロックをもつ未処理のユーザ・プロセスのリストを保守しなければならない．制御が DBMS に渡ったとき，これらのロックは掛け直される．

利用の段階は一番活動的な段階であり，DDB を制御している DBMS はマルチ・スレッドで操作し，すべての新しい割り当てと変更は更新テーブルに反映される．制御の移動順序が固定されているので，実現の際には割り当て方針の改善が必要となるかもしれない．基本的には必要グラニュールが前任者によってロックされ続ければ，利用過程は，止ってしまう可能性がある．これを直すためにロックにより待たされている DBMS をロック・サブリストと関係付けておくことができる．そうしておくと DBMS はグラニュールのロックが解かれていて，かつ，その DBMS が要求している場合にかぎって，そのグラニュールを割り当てるであろう．この割り当て方針の改善は，グラニュールの単位が小さければ必要ないのかもしれない．

終了の段階は割り当てられた時間を使い果たしたとき開始する．このとき DBMS は後

任者をコミュニケーション・サブ・システムに知らせ，制御と DDB 状態の移動を始動させる．

要するに，CMT アルゴリズムの記述は，より一層の詳細化のための手引きとなるように意図されている．アルゴリズムの柔軟性は結論で示しているようにいくつかの変種を許すであろう．

CMT アルゴリズムの分析は，記憶装置とコミュニケーションに対する要求に基づいて行う．この二つの測定基準は，質的に異なる手法の比較を行うためにも共通となるものである．

制御テーブルと更新テーブルは，記憶装置に対するキーとなる要求から組み立てられる．n 個の DBMS をもつ DDB システム内の各 DBMS は，$(3\times n)$ の長さの制御テーブルを必要とする．この長さは DDB システムが DBMS を新たに組み込むが，削除するときにわずかに変化する．これに対し，更新テーブルの長さはより一層変わりやすい．その長さはある与えられた時間で変化するかもしれないが，固定された要求の到着率と時間に対しては安定するものと期待される．

協調的制御移動によって，CMT アルゴリズムは要求される DBMS 間のメッセージの数においてはっきりした経済性を示している．すなわち，各々のロックと変更は $(n-1)$ 回まわるので 1 更新につき最小の $(2n-2)$ 個のメッセージとなる．処理を完結するために二つ以上の制御サイクルを必要としているユーザ・プロセスは，それらのロックを再びまわさなければならないので，その分だけ最小値にいくらか加えることになる．

3.3 CMT アルゴリズムの評価

分析が量的であるが，評価は質的である．評価の目的は CMT アルゴリズムの効率を促進するサブ・システム・アーキテクチャを示唆することと，アルゴリズムのいくつかの効率に重大な影響を及ぼす部分をはっきりさせることである．これによって，支援操作環境が示唆される．

ノード，データベースおよびコミュニケーション・アーキテクチャは CMT アルゴリズムの効率に強く影響する．ノードのアーキテクチャはコミュニケーション，データ処理およびデータ管理機能からなるローカル機能の組織を意味している．ノードの処理能力はそのノード・レベルにおいて，機能的および地理的分散による同時平行および重複処理によって強化できる．事実，ノード・アーキテクチャはコミュニケーションのためのフロント・エンド・プロセッサ，業務処理のためのマルチ・プロセッサおよびデータ管理処理のためのバック・エンド・プロセッサから構成される。マルチ・プロセッサはそのプロセッサ・レベルにおいて処理能力を強化できるであろうし，データベース・アーキテクチャも早い応答を支援することにより処理能力を強化できる。

データ・モデルとアクセス方法の賢明な選択と共に頻繁に行う調整は，これら効率を上げようとする努力を支援する。そして，コミュニケーションのアーキテクチャが，もし媒体の速さと帯域幅が予測される負荷に見合うように選ばれるならば，処理能力は強化できる．そして，これはささいなことではなく，適した帯域幅の選択は更新リスト長の平均に依存しており，それは，また逆に順々に到着する要求の頻度と割り当て時間量に依存している．この選択には十分な情報が必要である．分析的方法とシュミレーションも，しっかりした決定を下すうえでの判断基準を与える．幸いなことに，時間単位の長さを調整することによって，効率を高め，判断の誤りを補正できる．技術的観点からみて，マイクロウェーブ・ラジオ・リンクを使った高速で，非常に高い帯域幅の遠くて長いコミュニケーシ

ョン・リンクや，サテライト・リンクによって大容量データ・ブロック移送が効果的に行える．これらは CMT アルゴリズムに対し効率改善の可能性を広げている[4]．

効率に重大な影響を与えるいくつかの部分について簡単に述べて定性的分析を終える．CMT アルゴリズムは相互の一貫性，デッドロック，頑丈さ，アクセス機会の均等化および処理能力／ターンアラウンドに関して以下のように評価されている．

相互の一貫性——$c(t)$ で示される時間 t における複写間の“一致”は，分散データベース・システムの質をはかるとき通常に用いられる目安である[5]．単に一つの論理的データベースがあるだけなので，すべての物理的複写は理論上同一であるべきである．

コミュニケーション媒体は瞬時に稼働するわけでもないし，メッセージの受け渡し方法も異なるために，複写間の完全な“一致”は稀である．相互の一貫性の測定はすべての複写内で一致しないグラニュールの数を測ることで行われ，CMT アルゴリズムに対しては $c(t)\geq(N+1)/2$ となる．ここで $N=\max\{n\}$ であり，n は DDB の最新の複写内で一致していないグラニュールの数である．もし，すべての更新処理が一つの制御サイクル一杯かかって終了するのであれば，$c(t)\to 0$ となるので，CMT アルゴリズムの相互の一貫性は比較的弱い，しかし，このことはたいして問題にならない．なぜなら，ユーザは最新の DDB だけをアクセスすればよいからである．

CMT アルゴリズムでは，実行に先立ってユーザにすべての要求するグラニュールを要求させることにより，デッドロックを防止している．完全なグラニュール目録はユーザの宣言，あるいはこの目標のために設計されたシステム・ソフトウェアによって，プログラム・インフォメーション・ブロックの一部として格納することができる．そして，DBMS は要求されたすべてのグラニュールが利用可能なときのみ割り当てを行う．

CMT アルゴリズムはいくつかの方法を用いて，システムの頑丈さとリカバリを単純化している．故障したホスト・コンピュータは制御テーブルを変更することによって迂回させることができるし，前任者の更新テーブルは最新の正しい複写を備えることによってリカバリを助ける．CMT アルゴリズムの頑丈さとリカバリ手続きのより詳しい説明は Greene[3] の論文で述べられている．

CMT アルゴリズムはユーザ・プロセス・レベルより，むしろ DBMS レベルにおいてアクセス機会を均等化している．DBMS レベルで DDB を時分割することによって，1 時間単位の中での時間の配分はその DBMS のローカルな制御の下にあり，ローカルな要求にも応えることができる．これによってアルゴリズム全体の融通性が高められる．

処理能力の話題はすでに述べたが，容認されるターン・アラウンドの可能性も問題になる．ロック・リストが広域レベルでのアクセスの相互的排他を保証しているため，ホスト・コンピュータは DDB 制御の現在位置にかかわらずユーザ・プロセスを実行できる．

そして，DDB システムの発展は時間駆動式の協調的制御移動の柔軟性を明らかにしている．DDB システムが作業負荷を変更したりノードを追加／削除するに伴い，資源利用と効率を安定させるために時間単位の長さを適切に調整できる．

ここで CMT アルゴリズムが適する作業環境の特性を示唆できる．その第 1 は，協調的制御移動を正当化するのに十分なほど頻繁に生じる一貫した要求が各ノードごとに存在しなければならないということである．これは，各 DBMS が待ちのユーザをかかえているという協調的制御移動で仮定した前提を支援している．また，更新テーブル長を安定させることも支援している．第 2 はグラニュール・ロックを掛け直すことを防ぐためにユーザ・プロセスの実行時間は一つの制御サイクル内で終了させるべきということである．第

3は作業環境は機能的，かつ地理的に分散すべきということである．いい換えれば，ユーザは物理的に異なった地点にいながら，同じデータを要求して同じ機能を実行しているということである．この特性は DDB の時分割の概念と適合している．また，トランザクション処理環境における DDB システムは容易にこれらの特性を例示してくれる．

4. おわりに

市場への分散データベース・システムの出現は，差し迫ったものである．目標とする多数のユーザに効率的なサービスを行うには更新の方法に特別な注意を向ける必要がある．この更新方法は，ユーザの作業環境での効率上の期待と合致しなければならない．現行のすべての更新の手法は稼働するが，一般的な骨組みのところで述べたように，それぞれ異なる特性を基盤にしており，ユーザに質的に異なる選択を提供している．ユーザ作業環境と更新手法の類似性が，DDB システムの全体的な成功の重要な要因になりうる．更新手法は，その基本的な様相を作業環境に近づけるかぎりにおいてのみ，作業環境と合致する．

CMT アルゴリズムの設計はトランザクション処理作業環境と高処理能力の目標としている．これら二つの性格は方法の設計がどうあるべきかを導くが，時間駆動式の協調的制御構造のアルゴリズムを用いるともっとも効果がある．

将来DDB システムにおける研究に対し，いくつかの挑戦が出てくるであろう．CMT手法内では，時間単位とノード・スケジュールの決定が将来の研究の生産的な領域であると思われる．おそらく発見的方法で行われると思うが，もし時間量が動的に調整されるならば，何が効率面での効果なのかが問われるであろう．同様にラウンド・ロビン以外のどんなノード・スケジューリング方式が，CMTの手法に適用されるのかが問われるであろう．制御の移動が協調的なので，制御テーブルが人工知能の技術を使って作られるのではないかと思う．システムに関して，一般的には特定の作業環境を設定して更新手法をそれに合わせるように設計されるシュミレーションの研究は，知的な DDB システム設計に必要な基礎を提供するであろう．

（プロダクト・サポート統括二部　城代　優二 訳）

参考文献
[1] P. A. Bernstein, et al., "Analysis of Serializability in SDD-1 : A System for Distributed Databases," *IEEE Transactions on Software Engineering*, IEEE 1978.
[2] G. LeLann, "An Analysis of Different Approaches to Distributed Computing," *Proc. of the First Inter. Conf. on Distributed Computing Systems*, IEEE 1979.
[3] R. J. Greene, "Updating Approaches for Distributed Databases," Thesis, University of Alabama in Huntsville, 1980.
[4] M. V. Wilkes, "The Impact of Wide Band Local Area Communication Systems on Distributed Computing," 1979.
[5] W. C. Wesley and P. P. Chen (ed.), *Centralized and Distributed Data Base Systems*, 1979.

執筆者紹介　Richard J. Greene
Huntsville の Alabama 大学より計算機科学で M. S. を取得．Blue Bell の Sperry Univac 社の Information Management Systems グループでシステム・プログラミングに従事．

論説 自動設計問題の計算量

The Complexity of Design Automation Problems

Sartaj Sahni, Atul Bhatt

要 約 本稿では，自動設計分野で発生する問題を検討する．これらの問題の大部分は NP-難度であることが示される．しかも，これらの問題は，どれも答が最適解のある定まった相対誤差以内にあることを保証するような速い近似アルゴリズムが存在する見込みがない．したがって，発見的アルゴリズムやその他の道具を使って，問題の要点をうまくつかまえるアルゴリズムを作ることが重要であろう．

Abstract This paper reviews several problems that arise in the area of design automation. Most of these problems are shown to be NP-hard. Further, it is unlikely that any of these problems can be solved by fast approximation algorithms that guarantee solutions that are always within some fixed relative error of the optimal solution value. This points out the importance of heuristics and other tools to obtain algorithms that perform well on the problem instances of "interest".

1. はじめに

ディジタル・システムは部品が非常に多く，かつ複雑なため自動設計 (DA) の道具の使用と開発が必要となった．自動設計化への努力は，LSI の出現と急増でかなり推進されたが，VLSI と VHSIC の出現によって，その需要と重要さは将来も増加するだろう．今日，自動設計の研究は，おもに新しいアルゴリズムや発見的方法の開発，またそれらの効率的な実行や設計・技術の変更に適応させるための修正などを取り扱っている．

本稿では，自動設計でよく出会う問題の計算量分析について述べる．採りあげる問題は，主として設計過程の各段階，つまり設計の実施，配置，配線（相互接続と径路決定），故障箇所発見のそれぞれで発生するものである．シミュレーションに関するものや自動設計データ・ベースの管理については取り扱わない．

自動設計問題の概要は 2.1 節で述べる．2.2 節では 2.1 節で述べた問題の数学的表現を与える．計算量と非決定性アルゴリズムの概念は 3 章で述べる．この章では，他の関係資料や自動設計と関係のある NP-難度および NP-完全問題についても述べている．

2.2 節で挙げた問題の計算量は，4 章で分析する．これらの問題のほとんどは NP-難度であることが示される．最後に，5 章では発見的方法や "普通はうまくいく" アルゴリズムを使って，これらの問題を解決する方法を述べる．

M. A. Breuer によって編集された本[2]と調査論文[3]は，自動設計の問題と解くための工夫，およびディジタル・システムへのその応用について，よい解説となっている．V. M. van Cleemput[6] には自動設計に関連する分野の詳しい文献表を載せている．

This article is reprinted, with permission from ACM Conference Proceedings (17th Design Automation Conference 1980).
S. Sahni の研究は，その費用の一部を米国の国家科学基金 (NSF, MCS 78-15455) から援助を受けている．
The updated version of this paper is available as "The Complexity of Design Automation Problems" by Sartaj Sahni, Atul Bhatt and Raghunath Raghavan, Technical Report 80-23, COMPUTER SCIENCE DEPARTMENT UNIVERSITY OF MINNESOTA, December 1980 (revised February 1981).

2. 自動設計問題

自動設計問題の大部分は互いに深く関連しあっているが，全体問題として解こうとするとその計算量は膨大なものとなるので，それらを分離して取り扱う．実際には，設計の（パッケージの多重階層の中の）レベル，採用されている特殊技術，回路の電気的制約それに問題を解くために使える道具などが，問題の一般的クラスを決めるのに強く影響するし，具体的問題は，問題の大きさや制約と最適化のためのパラメタの選択，解法を設計する方法論などに影響を与えている．

2.1 総　説

2.1.1 回路設計問題

回路設計問題は以下のもので成り立っている．

1) 合成……ディジタル・システムの論理表現を同じ機能をもつ他の表現に変換する問題であり，その費用や使われている要素の個数を最適化する．自動合成の分野は，まだ多くの研究が必要である．

2) モジュール標準ライブラリの作成……モジュールの種類を適当な数におさえ，費用もそう高くないような標準ライブラリをどのように設計するかが問題である．W. A. Notz らは，論文[24]でモジュール標準ライブラリの周期的更新に役立つものさしを提案している．

3) 分割……分割問題は，システム階層のいろいろなレベルで出会う．一般的にいえば，与えられたレベルで詳細設計の決まっている全体システムを，与えられた標準ユニットのライブラリから組み立てることができるいくつかのサブ・システムに分割することである．しかも，事前に定義されている基準で最適化しなければならない．

論理回路レベルで，一般的に分割問題と関係するパラメタは以下のようなものである．

- 回路によって占められる空間
- 回路が必要とする外部接続の個数
- 回路の費用
- 回路で起こる遅れ
- 回路の信頼性と試験可能性

上記の最初の二つは非常に重要であると考えられていて，分割問題のどの定式化にも含まれている．最適化のために普通選ばれるパラメタは以下のとおりである．

- 分割要素の個数[20]
- 部品間の電気的接続の個数[21],[11]
- 回路全体の最大遅れ[22]

4) モジュールの選択……選択問題は，モジュール標準ライブラリと回路の分割が与えられて，分割された回路を実現するための費用，またはモジュール数を最小にするモジュールの集合を求めることである．

2.1.2 配置問題

配置問題は互いに関連する物を与えられた領域の中で，決まっている場所(セル)に最適に割り当てる組み合わせ問題とみることができる．互いに関連する物と場所の正確な定義は，問題としている（パッケージの多重階層中の）レベルと採用されている技術によって決まる．たとえば，チップ上の論理回路の配置，基板上のチップの配置，バック・プレーンへの基板の配置などであるが，簡便のために基板上にモジュールを配置することを配置問題であると考えることにしよう．ただし，モジュールと基板という言葉は，論理回路と

チップ，基板とバック・プレーンなどと同じである．そして，一般的にはコンピュータのどのレベルにでも適用できるとはいっても，考えているレベルによって問題の大きさ，制約と最適化のためのパラメタの選択，解法の設計などに与える影響が異なる．

配置問題の最適化の基準は，一般的には互いの接続に関して定義される基準である．実際は，以下に述べるいくつかの目標を満足しなければならない．

1) 基板上の配線領域に集中する線を許容限界内に保つ．
2) 絶縁箇所をなるべく少なくして混信をなくす．
3) 信号の反射をなくす．
4) 発熱を制御する．
5) 配線をすっきりと簡潔にする．
6) 基板の信頼性，保守性，試験可能性
7) 混雑の最小化[6]
8) 交差線の除去[19]
9) 枝分かれ（信号線の折れ曲がり）の総数を最小化する[25]．

上記の目標のすべてを満足することは事実上不可能である．実際に使われている基準は，配線の長さの重みつき総和を最小化することである．

2.1.3 配線問題

これはまた，相互接続と径路決定問題とも呼ばれ，システムの要素を互いに接続する導通路の正確な形を定義することである．問題は以下のいくつかを制約とする．

1) 配線層の数
2) スルー・ホール（貫通ピン）または配線層間の径路の存在
3) 交差
4) 雑音
5) 混信
6) 信号の反射
7) 線幅（密度）
8) 径路の方向（横または縦）
9) 配線長
10) 配線に必要な広さと容量

配線問題は，実際には以下のような問題に分けられる．

1) 配線表の決定……配線すべき線を表にまとめることである．
2) 配線層の割り当て……それぞれの線をどこに位置付けるかを決定する．また，線が交差しないような配線層への割り当てのうち，配線層の数を最小にすることである．
3) 配線順序の決定……配線層へ割り当てられた線をいつ配線するかを決める．この順序は総配線長に影響を及ぼすだけではなく，配線処理全体にとっても重要である．
4) 線の配置……二つの要素間の接続径路を決める．

2.1.4 故障箇所の検出

故障箇所検出問題の大部分は，与えられた論理回路のあるクラスの誤りを検出するための試験信号系列を作ることである．存在する技法のほとんどは組み合わせ回路だけを考えていて，以下に述べるもののいくつかを仮定している．

1) 誤りは恒久的であり，縮退型である．
2) 回路に誤りは一つしかない．

3) 回路は初期状態に戻すことができる.

4) 検査は静定状態で行われ，レーシングは考えない.

確かに，上記の仮定がすべて成立つわけではない．この分野での他の興味ある問題としては，①誤りの検出，場所決め，マスキングを手助けできるような回路の性質を決めること，②自動診断問題，たとえば誤り検査（発見と矯正）手順の設計，などがある.

2.2 数学的定式化

2.2.1 回路設計問題 (Implementation Problems)

IP 1 関数の実現

入力：ブール関数Bとブール関数 F_i を実現している基本要素 c_i, $i=1,\cdots,k$

出力：基本要素 c_i, $i=1,\cdots,k$ で作られるBを実現する回路のうち，基本要素の個数が最小のもの

IP 2 回路の検証

入力：ブール関数Bと回路C

出力：CがBを実現していれば，"Yes"，他の場合は "No"

IP 3 回路の割り付け

入力：回路の要求 $(b_1, b_2, \cdots, b_n$；回路を実現するためには タイプ i のゲートが b_i 個必要である.)

モジュール i, $i=1,\cdots,r$. モジュール i 1個の費用は c_i であり，タイプ j のゲートを m_{ij} 個もっている.

出力：非負整数 $x_1, x_2, \cdots, x_n$

ただし，$\sum_i m_{ij}x_i \geq b_j$, $1\leq j\leq n$

$\sum_i c_i x_i \rightarrow$最小

IP 4 最小費用モジュール標準ライブラリの作成

入力：タイプ j, $1\leq j\leq r$ の回路を y_{ij} 個必要とする論理回路 $c_i, 1\leq i\leq n$ と1個のモジュールで使える回路の個数制限 p

出力：モジュール・タイプの集合 $M=\{m_1, m_2, \cdots, m_k\}$

ただし，a_{ij} をモジュール m_i がもつタイプ j の回路の個数とすると，

$\sum_j a_{ij}\leq p$, $1\leq i\leq k$

x_{ij}, $1\leq i\leq n$, $1\leq j\leq k$, は自然数である.

$\sum_j x_{ij}a_{jm}\geq y_{im}$, $1\leq m\leq r$, $1\leq i\leq n$

以上の制約を考えて $\sum_{i,j} x_{ij}$ が最小となるように M を選ぶ.

IP 5 モジュール・タイプの数が最小である標準ライブラリの作成

入力：IP 4 と同じものに，費用制限 c の指定を加える.

出力：$M=\{m_1, m_2, \cdots, m_k\}$

ただし，x_{ij} は自然数である.

$\sum_j x_{ij}a_{jm}\geq y_{im}$, $1\leq m\leq r$, $1\leq i\leq n$

$\sum_{i,j} x_{ij}\leq c$

これらを満たす a_{ij} (IP 4 と同じ)，$1\leq i\leq k$, $1\leq j\leq r$ で要素の数が最小となる M をみつける.

IP 6 分割数の最小化

入力：モジュールの集合 $V=\{1, 2, \cdots, n\}$

モジュール i, j 間の接続数である対称な重み $w(i, j)$

モジュール i が必要とする空間量 $s(i)$

接続数の制限Eとモジュールごとに必要とする空間の制限S

出力：Vの最小分割 $P=\{p_1, p_2, \cdots, p_k\}$

ただし，$\sum_{i\in p_j} s(i)\leq S,\ 1\leq j\leq k$

$$\sum_{i\in p_j, q\notin p_i} w(i, q)\leq E,\ i\leq j\leq k$$

IP 7 最小接続分割 I

入力：IP 6 と同じ V, w, s, S

出力：Vの分割 $P=\{p_1, p_2, \cdots, p_k\}$

ただし，$\sum_{i\in p_j} s(i)\leq S,\ 1\leq j\leq k$

$$\sum_j \{\sum_{i\in p_j, q\notin p_j} w(i, q)\}\rightarrow 最小$$

後者の和は，分割間の接続数の 2 倍となっている．

IP 8 最小接続分割 II

入力：IP 6 と同じ V, w, s, S と定数 r

出力：Vの分割 $P=\{p_1, p_2, \cdots, p_k\}$

ただし，$k\leq r$

$$\sum_{i\in p_j} s(i)\leq S,\ 1\leq j\leq k$$

$$\max_j \{\sum_{i\in p_j, q\notin p_j} w(i, q)\}\rightarrow 最小$$

IP 9 最小空間量分割

入力：IP 6 と同じ V, w, s, E と定数 r

出力：Vの分割 $P=\{p_1, p_2, \cdots, p_r\}$

ただし，$k\leq r$

$$\sum_{i\in p_j, q\notin p_j} w(i, q)\leq E, 1\leq j\leq k$$

$$\max_j \{\sum_{i\in p_j} s(i)\}\rightarrow 最小$$

IP 10 モジュール選択問題

入力：タイプ i, $1\leq i\leq r$ の回路を y_i 個必要とする（IP 6 の出力である）分割要素 A'

モジュール・タイプの集合 $M=\{m_j | 1\leq j\leq n\}$．タイプ m_j のモジュールが z_j 個ある．

各 m_j の費用は h_j であり，タイプ i の回路を a_{ij} 個，$1\leq i\leq r$, $1\leq j\leq n$ もっている．

出力：非負整数 $x_1, x_2, \cdots, x_n$

ただし，$0\leq x_j\leq z_j$

$$\sum_{j=1}^{n} a_{ij}x_j\geq y_i,\ 1\leq i\leq r$$

$$\sum_{j=1}^{n} x_j h_j\rightarrow 最小$$

2.2.2 配置問題 (Placement Problems)

PP 1 モジュール配置問題

入力：m　モジュールの個数

p　利用できる場所（スロット，番地）の数

s　信号の個数

$N=\{n_1, n_2, \cdots, n_s\}$，$n_i \subseteq \{1, 2, \cdots, m\}$

n_i，$1 \leq i \leq m$ は信号線である．

$D(p \times p)=[d_{ij}]$

d_{ij} はスロット i とスロット j の距離

$W(1:s)=[w_i]$

w_i は信号線 n_i，$1 \leq i \leq s$ の重み．

出力：$X(m \times p)=[x_{ij}]$

ただし，$x_{ij} \in \{0, 1\}$

$$\sum_{j=1}^{p} x_{ij}=1 \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots ①$$

$$\sum_{i=1}^{m} x_{ij} \leq 1 \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots ②$$

$$\sum_{i=1}^{s} w_i f(i, X) \rightarrow \text{最小} \cdots\cdots\cdots\cdots\cdots ③$$

モジュール i がスロット j に割り当てられるときに，$x_{ij}=1$ となる．制約①，②はそれぞれ，モジュールはどこかのスロットに割り当てられること，またスロットにはせいぜい 1 個しかモジュールを割り当てないことを保証している．$f(i, X)$はこの割り当ての下での信号線 n_i の費用を与える．たとえば，最小スパニング木の費用，信号線の中のすべてのモジュールを接続する最短 Hamilton 径路の長さ，最小スタイナー木の費用などである．一般的に費用は d_{ij} 関数である．

2.2.3 配線問題 (Wiring Ploblems)

WP 1 Manhattan 距離による信号線の配線

入力：ピンの集合 $P=\{(x_i, y_i) \mid 1 \leq i \leq n\}$

貫通ピンの集合 $F=\{(a_i, b_i) \mid 1 \leq i \leq m\}$

等価クラスの集合 $E=\{E_i \mid 1 \leq i \leq r\}$

等価クラスは，電気的に等価であるピンの集合である．

出力：線の集合 $W_i=\{[(t_j{}^i, u_j{}^i), (v_j{}^i, w_j{}^i)] \mid 1 \leq i \leq q_i\}$

ただし，E_i のピンはすべて電気的に等価にされる．

$(t_j{}^i, u_j{}^i)$ と $(v_j{}^i, w_j{}^i)$ は E_i の中のピンか貫通ピンである．

貫通ピンは，二つ以上の w_i の終点にはならない．

$W=\{w_1, w_2, \cdots, w_r\}$ は，$\sum_{i,j}(|t_j{}^i-v_j{}^i|+|u_j{}^i-w_j{}^i|)$ を最小化したものである．

WP 2 Euclid 距離による信号線の配線

入力：WP 1 と同じ P, F, E.

出力：WP 1 と同じ線の集合

ただし，$\sum_{i,j}[(t_j{}^i-v_j{}^i)^2+(u_j{}^i-w_j{}^i)^2]^{1/2}$ を最小化する

WP 3 Manhattan 距離による接続数制約付き配線

入力：WP 1 と同じ P, E, F と定数 d

出力：WP 1 と同じ線の集合

ただし，任意のピンまたは貫通ピンに d 本までの線しか結ばない．$d=2$ のとき貫通ピンは何の役にも立たない．

WP 4 Euclid 距離による接続数制約付き配線

入力：WP 1 と同じ P, F, E と定数 d

出力：WP 2 と同じ線の集合

ただし，任意のピンに d 本の線しか結ばない．

WP 5 Manhattan 距離による長さ制約付き配線

入力：WP 1 と同じ P, F, E と定数 L

出力：WP 1 と同じ線の集合

ただし，$|t_j^i - v_j^i| + |u_j^i - w_j^i| \leq L$

WP 6 Euclid 距離による長さ制約付き配線

入力：WP 1 と同じ P, E, F と定数 L

出力：WP 2 と同じ線の集合

ただし，$[(t_j^i - v_j^i)^2 + (u_j^i - w_j^i)^2]^{1/2} \leq L$

WP 7 Euclid 距離による配線層割り当て問題 I

入力：線の集合 $W = \{(u_i, v_i), (x_i, y_i) | 1 \leq i \leq n\}$

(u_i, v_i) と (x_i, y_i) は線 i の端点の座標

出力：W の分割 $W = \{w_1, w_2, \cdots, w_k\}$

ただし，$w_i \cap w_j = \phi$，$i \neq j$

$U w_i = W$

w_i の中の任意の 2 線は交差しない．

線は（Euclid 的に）まっすぐな線で結ぶ．

$k \rightarrow$最小

WP 8 Manhatlan 距離による配線層割り当て問題 I

入力：WP 7 と同じ

出力：WP 7 と同じ

ただし，線は Manhattan 的に結ぶ（すなわち，x 軸に沿った直線と y 軸に沿った直線で接続される）．

WP 9 Euclid 距離による配線層割り当て問題 II

入力：WP 7 と同じ W と定数 r

出力：W を $w_1, w_2, \cdots, w_r$ と X へ分割する．

ただし，端点が直線で結ばれている．任意の W_i の中の 2 線は交差しない．

$|X| \rightarrow$最小

WP 10 Manhattan 距離による配線層割り当て問題 II

入力：WP 9 と同じ

出力：WP 9 と同じ

ただし，線は Manhattan 的に結ばれている．

WP 11 線の径路決定

入力：WP 7 と同じ線の集合 W と $m \times n$ の格子

線の端点は格子点に対応している．W は，端点の組 $[(u_i, v_i), (x_i, y_i)]$ が直角

に曲げる線によってのみ接続でき，どの2線も交差しないように接続できる．

出力：各線の径路

ただし，配線長は最小である．

直角にしか曲げない．

どの2線も交差しない．

2.2.4 故障箇所検出問題 (Fault Detection Problems)

Cを n-入力 1-出力の組み合わせ回路とする．Zは0縮退 ($s-a-0$) と1縮退 ($s-a-1$) が起こりうる場所の集合とする．$(i_1, i_2, \cdots, i_n, j, F(2), F(1))$ の組は，以下のものを満足するときの，Cの故障検出試験である．

① $i_k \in \{0, 1\}$，$1 \leq k \leq n$

$j \in \{0, 1\}$

② $F(0) \subseteq Z$，$F(1) \subseteq Z$，$F(0) \cap F(1) = \phi$，$F(0) \cup F(1) \neq \phi$

③ Cが入力 $i_1, i_2, \cdots, i_n$ に対し，出力 j をもつときのみ，$F(0)$ の中に $s-a-0$ 誤りがあるか，$F(1)$ の中に $s-a-1$ 誤りがある．

試験信号系列Tは，故障箇所検出試験信号の集合であるとする．①Tの試験信号に対するすべての $F(0)$ の和集合がLであり，②Tの試験信号に対するすべての $F(1)$ の和集合がLであるときに，Tは $L \subseteq Z$ のための試験信号系列であるという．$L=Z$ であれば，TはCのための試験信号系列であり，回路Cは試験信号系列をもつときに冗長でないという．

以下のような故障箇所検出問題を定義しよう．

FDP 1 組み合わせ回路は冗長ではないか．(すべての $s-a-0$ と $s-a-1$ 誤りは，入出力試験で検出できるか)

FDP 2 個々の入力線の誤りは入出力試験で検出できるか．

FDP 3 すべての単一入力誤りは入出力試験で検出できるか．

FDP 4 出力線の誤りは入出力試験で検出できるか．

FDP 5 Cが冗長でないときに，Cの最小試験信号系列を求める．

3. 計算量と非決定性アルゴリズム

3.1 計 算 量

アルゴリズムを実行するための所要計算時間と所要記憶容量をそのアルゴリズムの計算量という．これら二つの量は時間（計算）量，空間（計算）量ともいう．プロシージャ MADD（図1）は二つの $m \times n$ 行列の和を作るアルゴリズムである．

```
line  procedure MADD (A, B, C, m, n)
        //compute C=A+B//
  1   declare A(m, n), B(m, n), C(m, n)
  2     for i←1 to m do
  3       for j←1 to n do
  4         C(i, j)←A(i, j)+B(i, j)
  5       end
  6     end
  7   end MADD
```

図 1 行列の和

Fig. 1 Matrix addition

このアルゴリズムをコンピュータで実行させるのに必要なものは，①アルゴリズムをコンパイルするための時間と，②それを実行する時間である．①は使用するコンピュータとコンパイラによるが，mとnの実際の値とは独立である．実行時間②は，使用するコンピュータとコンパイラに加えて，mとnの値に依存する．より大きな行列を加えれば余計に時間がかかる．

アルゴリズムが実際に必要とする時間は，コンピュータによるからアルゴリズムの時間量の解析的分析は，アルゴリズムが必要とするステップ数を決めることしかできない．ステップ数は，入力と出力をパラメタとする関数として得られる．よく使われるパラメタの例としては，入力の個数，出力の個数，入力と出力の値の大きさなどである．

行列の和の例では，行数mと列数nが使うのに適正なパラメタであろう．プロシージャMADDの4番目の命令を実行当たりの1ステップとして数えるとすると，この命令はmn回実行されるから，アルゴリズムの総ステップ数はmnである．S. Sahni[27]の6章にはもっと詳しいステップ数分析が載っている．ステップの考えがいくらか厳密でないので，アルゴリズムの正確なステップ数を得ようとする気にならない．むしろ，ステップ数の漸近的限界を得ようとする．漸近的分析は記号としてO, Ω, θ, oを使う，これらを次のように定義しよう．

定義 ［漸近的記法］

$f(n)=O(g(n)) \Leftrightarrow \forall n \geq n_0$ に対して $f(n) \leq cg(n)$ となる正定数 c と n_0 が存在する．

$f(n)=\Omega(g(n)) \Leftrightarrow \forall n \geq n_0$ に対して $f(n) \geq cg(n)$ となる正定数 c と n_0 が存在する．

$f(n)=\theta(g(n)) \Leftrightarrow \forall n \geq n_0$ に対して $c_1 g(n) \leq f(n) \leq c_2 g(n)$ となる正定数 c_1, c_2, n_0 が存在する．

$f(n)=o(g(n)) \Leftrightarrow \lim_{n\to\infty} f(n)/g(n)=1$

O, Ω, θ, oは簡単に多変数の場合に拡張することができる．たとえば$f(n, m)=O(g(n, m)) \Leftrightarrow \forall n \geq n_0$ と $\forall m \geq m_0$ に対して $f(n, m) \geq cg(n, m)$ となる正定数 c, n_0, m_0 が存在する．

例 1：$3n+2=O(n)$ （$\forall n \geq 2$ に対して $3n+2 \leq 4n$）

$3n+2=\Omega(n)$

$3n+2=\theta(n)$

$6\times 2^n+n^2=O(2^n)$

$3n=O(n^2)$

$3n=o(3n)$

$3n=O(n^3)$

上記の例のように，$f(n)=O(g(n))$ であることは，$\forall n \geq n_0$ に対して $g(n)$ は $f(n)$ の上限であるといっているだけで，この限界がどのくらいよいのかについては何も言及していない．$n=O(n), n=O(n^2), n=O(n^{2.5})$ などに注意．$g(n)$ は $f(n)=O(g(n))$ であるもののうち，できるだけ小さな関数である方が有効である．したがって $3n+3=O(n^2)$ とは使わず，$3n+3=O(n)$ を使う．

記号Oのように，$f(n)=\Omega(g(n))$ にもたくさんの関数 $g(n)$ が存在する．$g(n)$ は $f(n)$ の下限でしかない．記号θは記号OやΩよりもっと精密である．次の定理では，$f(n)$ が nの多項式であるときの $f(n)$ の大きさについて非常に有益な結果が得られている．

定理 1: $f(n)=a_m n^m+a_{m-1}n^{m-1}+\cdots+a_0$, $a_m \neq 0$ とする.

① $f(n)=O(n^m)$

② $f(n)=\Omega(n^m)$

③ $f(n)=\theta(n^m)$

④ $f(n)=o(a_m n^m)$

漸近的分析は，空間量のためにも使われる．漸近的分析は，アルゴリズムが何秒間実行されるか，または記憶領域をどのくらい必要とするかを教えてはくれないけれども，計算量の増加率を性格付けてくれる．そこで，もし $m=100$ と $n=10$ の問題に対してプロシージャ MADD が2ミリ秒かかったとすると，(MADD の計算量は $\theta(mn)$ であるから) $mn=16000$ であれば，16ミリ秒くらいかかるであろう．n が十分大きいときに $\theta(n^2)$ のアルゴリズムは，$\theta(n^3)$ のアルゴリズムよりは早いと思われる．

注意すべきもう1点は，アルゴリズムの計算量が，つねに入力と出力の個数や値の大きさによって決まるわけではないことである．時間のかかり方はデータによることが多い．例として図2を考えてみよう．これは，$\{W(1), W(2), \cdots, W(n)\}$ の中に要素の和が M となる部分集合があるかどうかを決めるための基本的なバックトラック・アルゴリズムである．これは部分集合の和の問題と呼ばれている．プロシージャ SS の実行は，$X \leftarrow SS(1, M)$ で始まる．

```
procedure SS(i, P)
    //determine if W(i : n) has a//
    //subset that sums to P//
    global W(1 : n), n
    if i>n then return (false)
    case
      : W(i)=P : return (true)
      : W(i)<P : if SS(i+1, P−W(i))
                   then return (true)
                   else return SS(i+1, P))
                 end if
      : else :   return (SS(i+1, P)
    endcase
end SS
```

図 2 部分集合の和

Fig. 2 Sum of subset

プロシージャ SS は $W(1:n)$ の和がMとなる部分集合があったときに真を帰す．もし，$\sum_{i=1}^{n} W(i)=n$ ならば，SS は $\theta(n)$ 時間かかる，和がMになるような $W(1:n)$ の部分集合がなければ，SS は終了するまでに $\theta(2^n)$ 時間を必要とする．他の場合の必要な時間は $\theta(n)$ と $\theta(2^n)$ の間にある．計算量がまさにデータ依存であるような場合には，最良の場合の計算量，最悪の場合の計算量および平均（期待）計算量について考える．これらの言葉の厳密な定義は S. Sahni の論文[27]にある．ここでは，これらの言葉に対するわれわれの直観的理解に頼っておこう．

今日得られている計算量研究のほとんどが，最悪の場合の計算量を考察している．このような分析の価値には議論があるだろう． プロシージャ SS は最悪の場合 $\theta(2^n)$ の時間量をもっている．これは，これだけの時間を使う入力が存在することを教えている．けれども，実際の問題に対してアルゴリズムは，もっとずっと効果的（たとえば，$O(n^2)$ くらい）であるかも知れない．平均計算量の考えは，アルゴリズムを実際に使うときに得られる計算量を，よりうまくつかまえるように思える．けれども，平均計算量の分析は最悪の場合の分析よりずっとむずかしいので，うまくいっているアルゴリズムはあまりない．結果として，本稿は主に自動設計問題の最悪の場合の計算量について考えることにする．たくさんの自動設計問題が最悪の場合の計算量で考えると，たぶん手に負えないことが示されるけれども，この結果はアルゴリズムの平均的な働きとしては非常に効率的である可能性を除外するものではない．

3.2 非決定性アルゴリズム

一般的に使われるアルゴリズムの考えは，各ステップの結果が一意に定義される性質をもっている．この性質をもつアルゴリズムは決定性アルゴリズムと呼ばれている．理論的枠組みでは演算結果に対するこの制約を取り除いて，結果が一意に定義されていなくて，ある決まった集合から要素を一つ任意に取り出して，それを結果とするような演算を許すアルゴリズムを考える．このような演算を実行する機械は，これらの結果のどれを取ってもよいようになっている．これは非決定性アルゴリズムの考えを導くものである．このようなアルゴリズムを記述するために，三つの新しい関数を導入しよう．

1) choice (S) ……集合 S の要素を任意に一つ選択
2) failure……失敗終了の信号
3) success……成功終了の信号

このようにして，代入文 X←choice $(1:n)$ は X に $[1,n]$ の整数の一つを代入することになる．ここには，この選択がどのように行われるかを指示する規則は存在しない．失敗終了と成功終了の信号はアルゴリズムの計算結果を定義するのに使われる．この計算は，成功終了に導く選択の列があるときにはいつでも，そのような選択列が作られてアルゴリズムは成功終了する．非決定性アルゴリズムは成功終了に導く選択列が存在しないときに失敗終了する．このような方法で非決定性アルゴリズムを実行できる機械を非決定性機械という．

例 2：与えられた集合の $A(1)$ から $A(n), n\geq 1$ の中で，要素 X を捜す問題を考える．X が集合の中にあればその番号を，なければ 0 を j に入れる．この問題に対する非決性アルゴリズムは次のようになる．

```
j←choice (1 : n)
if A(j)=X then print (j); success endif
print ("0"); failure.
```

非決定性計算の定義から，数字 "0" は $A(j)=X$ であるような j が存在しないときに出力される． choice, success および failore のための計算時間は $O(1)$ とすべきである．したがって，上記のアルゴリズムの非決定性計算量は $O(1)$ である． A は順序づけられていないので，決定性探索アルゴリズムは少なくとも $O(n)$ の計算量であることに注意する．

非決定性アルゴリズムが成功終了するたくさんの選択列があるので，与えられたデータ・セット上で動くこのようなようなアルゴリズムの出力は一意に定義されていないにち

がいない．この困難を克服するためには，ふつう判定問題だけを考える．すなわち，0または1（真または偽）を出力する問題である．失敗終了は0，成功終了は1を出力する．

非決定性アルゴリズムの計算量をはかるために，choice (S) 関数に当てる費用は $O(\log k)$ (k は S の大きさ) である．与えられた入力に対する非決定性アルゴリズムが必要とする時間は，成功終了へ導く選択列が存在するかどうかによる．もし，そのような列が存在すれば，そのような終了を導く最小のステップ数が必要な時間量である．成功終了を導く選択列がないときは，失敗終了をするために $O(1)$ だけ必要である．

非決定性アルゴリズムは強力な道具のようにみえる．図3は，部分集合の和のための非決定性アルゴリズムである．その計算量は $O(n)$ であり，この問題に対する最良の決定性アルゴリズムの計算量は $O(2^{n/2})$ である[12]．

```
procedure NSS(W, n, M)
    declare X(1 : n), W(1 : n), n, M
    for i←1 to n do
        X(i)←choice ({0, 1})
    end
    if Σ_{i=1}^{n} W(i)X(i)=M then success
                          else failure
    endif
end NSS
```

図 3　非決性アルゴリズムによる部分集合の和

Fig. 3　Nodeteministic sum of subsets algorithm

3.3　*NP*-難度と *NP*-完全問題

問題の大きさは，その問題を記述するのに必要な桁の数とする．部合集合の和の問題は，$(W(1), W(2), \cdots, W(n), M)$ で与えられる．これらの数がすべて非負の整数であれば，この問題の大きさは2進数を使うと，$\left[\sum_{i=1}^{n} \log_2 W(i)\right] + [\log_2 M]$ である．入力の大きさ n と多項式 $p(\ )$ に対して，その計算時間が $O(p(n))$ であるときに，多項式時間の計算量をもつアルゴリズムであるという．

決定性多項式時間アルゴリズムで解くことのできる判定問題のすべての集合を P とする．NP は，非決定性アルゴリズムによって多項式時間のうちに解くことのできる判定問題すべての集合であるとする．あきらかに $P \subseteq NP$ であるが，$P=NP$ であるか $P \neq NP$ であるかどうかは，わかっていない．$P=NP$ 問題は，たくさんの興味ある問題（ある種の自動設計問題を含む）の計算量に関係するので重要である．$P=NP$ でなければ，多項式時間のうちに解けない問題がたくさん存在することになる．直感的には，$P \neq NP$ であるから，これらの問題はすべて多項式時間のうちに解けないだろう．$P=NP$ 問題にかかわることを示された最初の問題は，命題式を充足できるかどうかを判定する問題であった．この問題は充足可能性問題と呼ばれている．

定理 2：充足可能性問題が P に含まれるためには，$P=NP$ が必要十分である．
（証明は E. Horowitz らの著作[13] や M. R. Garey らの著作[10]を参照）

A と B を二つの問題とする．B に対する決定性多項式時間アルゴリズムの存在が，A に

対する決定性多項式時間アルゴリズムの存在を意味するときに，問題Aは多項式的に問題Bに変換できるという．また，AがBに変換できるともいい，式$A\propto B$とも書く．このようにして，もし$A\propto B$で，Bが多項式的に解けるならば，Aも多項的に解ける．問題Aは，充足可能性問題$\propto A$であるときに NP-難度である．NP-難度問題Aは，$A\in NP$のとき NP-完全である．

関係$\propto$は推移的（すなわち，$A\propto B$で$B\propto C$ならば$A\propto C$）であることに注意しよう．したがって，$A\propto B$で充足可能性問題$\propto A$であれば，Bは NP-難度である．このようにして，問題Bが NP-難度であることを示すためには，NP-難度であることがわかっているAに対して$A\propto B$であることを示せばよい．よく知られている NP-難度問題を以下に記す．

NP 1 Euclid 距離によるスタイナー木[9]

入力：点の集合 $X=\{(x_i, y_i)|1\leq i\leq n\}$

出力：有限個の点の集合 $Y=\{(a_i, b_i)|1\leq i\leq m\}$

ただし，$X\cup Y$ に対するスパニング木の全長を最小にするようにYを選ぶ．

2点間，(t, u) と (v, w) の距離は $[(t-v)^2+(u-w)^2]^{1/2}$ とする．

NP 2 Manhattan 距離によるスタイナー木[9]

入力：NP 1 と同じ

出力：NP 1 と同じ

ただし，2点間の距離として $|t-v|+|u-w|$ を使う．

NP 3 Euclid 距離による巡回セールスマン[8]

入力：NP 1 と同じ

出力：Xのすべての点を通る最短巡回

ただし，ユークリッド距離を使う．

NP 4 マンハッタン距離による巡回セールスマン[8]

入力：NP 3 と同じ

出力：NP 3 と同じ

ただし，Manhattan 距離を使う．

NP 5 彩色数 I[7]

入力：平面上の線分の交差グラフG

出力：Gを彩色するのに必要な最小の色数

NP 6 彩色数 II[7]

入力：NP 5 と同じ

出力：Gが3色で彩色可能であれば，“Yes”，他は “No”

NP 7 分割[15]

入力：自然数の重複を許す集合 $A=\{a_i|1\leq i\leq n\}$

出力：$\sum_{i\in B} a_i=\sum_{i\notin B} a_i$ であるような部分集合 $B\subseteq\{1, 2, \cdots, n\}$ があれば “Yes”

NP 8a ナップサック問題（最大化）[15]

入力：自然数の重複を許す集合 $P=\{p_i|1\leq i\leq n\}$, $W=\{w_i|1\leq i\leq n\}$ と自然数 M

出力：$x_i\in\{0, 1\}$

ただし，$\sum_i p_i x_i\rightarrow$最大

$\sum_i w_i x_i\leq M$

NP 8b ナップサック問題（最小化）

入力：NP 8 a と同じで，集合 P を $K=\{k_i|1\leq i\leq n\}$ に置き換える．

出力：$x_i\in\{0,1\}$

ただし，$\sum_i k_i x_i \rightarrow$最小

$$\sum_i w_i x_i \geq M$$

NP 9 整数ナップサック問題[23]

入力：非負整数の重複を許す集合 $W=\{w_i|1\leq i\leq n\}$ と二つの非負整数 M と K

出力：$K\leq\sum w_i x_i\leq M$ を満たす非負整数 $x_i, 1\leq i\leq n$ があれば "Yes"，なければ "No"

NP 10 2次の割り当て問題[26]

入力：c_{ij}，$1\leq i\leq n, 1\leq j\leq n$

d_{kq}，$1\leq k\leq m, 1\leq q\leq m$

出力：$x_{ik}\in\{0,1\}$，$1\leq i\leq n, 1\leq k\leq m$

ただし，① $\sum_{k=1}^{m} x_{ik}\leq 1$，$1\leq i\leq n$

② $\sum_{i=1}^{n} x_{ik}=1$，$1\leq k\leq m$

$$\sum_{\substack{i,j=1\\ i\neq j}}^{n}\left\{\sum_{\substack{k,q=1\\ k\neq q}}^{m} c_{ij}d_{kq}x_{ik}x_{jq}\right\}\rightarrow\text{最小}$$

NP-難度であることがわかっている問題が M. R. Garey らの著作[10]に多数掲載されている．問題 A が NP-難度であることを示すことの重要さは，$P=NP$ 問題があるからである．$P=NP$ ではないだろうから，NP-難度問題には，最悪の場合の計算量がその大きさの多項式時間となるようなアルゴリズムはないだろう．A が NP-完全で，しかも $N=NP$ であることにでもなれば，A は多項式時間で解けるだろう．

4. 自動設計問題の計算量

平面上の n 点の最小スパニング木は $O(n\log n)$ の時間でみつけることができる[28]．これは最初に Voronoi ダイアグラムを作り，つぎに，このダイアグラムの双対グラフを作る．この双対グラフは，$O(n)$ の辺と n 個の点をもっている．双対グラフの最小スパニング木は，もとの n 点の最小スパニング木と同じである．双対グラフは $O(n)$ の辺しかもたないから，その最小スパニング木は Kruskal のアルゴリズム (E. Horowitz らの著作[13]の4章) を使うと，$O(n\log n)$，または $O(n\log\log n)$[29] の時間でみつけることができる．アルゴリズム全体の計算量は，Voronoi ダイアグラムをみつけるのに必要な時間によって事実上決まる．この時間は $O(n\log n)$ である．結果としてできるスパニング木アルゴリズムの計算量を，自動設計の文献によく出てくるアルゴリズムと比較してみよう．一般に引用されている最小スパニング木のアルゴリズムは Kruskal のものであるが，その最悪の場合の計算量は $O(n^2\log n)$ である．M. I. Shamos とも D. Hoey[28] は $O(n\log n)$ が，平面上の n 点の最小スパニング木をみつける計算量の下限であることを示している．

4.1 回路設計問題

最小スパニング木問題は能率的に解けるけれども，他の自動設計問題は決してそうはいかないだろう．実際2章で取り上げた問題のほとんどは NP-難度である．問題 IP 1 と IP 2 は O. H. Ibarra と S. Sahni[14] によって NP-難度であることが示されている．整数

ナップサック問題 (NP 9) は回路の割り付け問題 (IP 3) に変換できることを簡単に示すことができる．W, M, K によって整数ナップサック問題を定義して，次のような IP 3 の問題を考えてみよう．

回路の要求　　：$(b_1)=K$

モジュールの構成：$m_{i1}=w_i, 1\leq i\leq n$

モジュールの費用：$c_i=w_i, 1\leq i\leq n$

あきらかに，考えている整数ナップサック問題の解が "Yes" のとき，回路割り付けの最小費用はたかだか M である．だから IP 3 が多項式的に解ければ，NP 9 も解ける．しかし NP 9 は NP-難度であり，IP 3 もまた NP-難度である．

問題 IP 6 は NP-難度であることを簡単に示すことができる．$a_1, a_2, \cdots, a_n$ を分割問題 (NP 7) として，次のような IP 6 を考えよう．

$$S(i)=a_i, 1\leq i\leq n$$
$$S=\sum a_i/2$$
$$w(i, j)=0, 1\leq i\leq n$$
$$E=0$$

分割された要素の和が S となる部分集合があれば，二つに分割される．だから，NP 7 $\propto$ IP 6 であり，IP 6 は NP-難度である．同様にして IP 8 と IP 9 もまた NP-難度である．

モジュール選択問題 (IP 10) は，特別な場合として 0/1 ナップサック問題を含むことを実証することによって，NP-難度であることを示すことができる．0/1 ナップサック問題 (NP 8 b) の W, M, K によって，次のような IP 10 を考える．

$$r=1$$
$$z_j=1$$
$$a_{ij}=w_j$$
$$h_j=k_j$$
$$y_1=M$$
$$i=1, 1\leq j\leq M$$

IP 10 が多項式時間アルゴリズムをもてば，NP 8 b は多項式時間で解くことができることがわかる．しかし，NP 8 b は NP-難度であるから，IP 10 もまた NP-難度である．

選択問題には他の定式化も存在する．Kodres は整数計画法によるものと制約付き最小費用流れ問題によるものを提案している[2]．これらの定式化は，ともに解法を考えるかぎり非常にむずかしい．整数計画 (ILP) によって定式化された問題は大きすぎて，現在のアルゴリズムによっては解くことができない．ディジタル・システムを表現したグラフがチェインをもつ特別な場合について，簡単化した ILP の定式化を M. A. Breuer[1] が提案している．主なねらいは，それぞれが完全な数え上げを許すのに十分な大きさの ILP 問題の並びに，選択問題を分解することである．この接近はもう一つの興味ある副問題（グラフのチェインへの分解，またはより小さいグラフへの分解）を発生させる．

4.2 配置問題

2 次の割り当て問題 (NP 10) は，配置問題 (PP 1) の特別な場合であることが容易にわかる．それには，$|N_i|=2, f(i, x)$ は単に N_i の中の二つのモジュール距離とすれば，NP 10 は PP 1 と等価な問題に変換することとことできる．だから，PP 1 は NP-難度である．

4.3 配線問題

NP 1 と NP 2 は，それぞれ WP 1 と WP 2 の特別な場合（すなわち，$E=\{P\}$ かつ

$F=\{(a_i,b_i)|(a_i,b_i)\notin P\}=\bar{P}$) であることが容易にわかる．だから，WP 1 と WP 2 は NP-難度である．

問題 WP 3 と WP 4 は，$d=2$ のとき貫通ピンを使っても配線長を短くすることはできないので，$F=\phi$ であると仮定する．$E=\{P\}$ である特別な場合には P の点をすべて通る最短 Hamilton 径路をみつけなければならない．径路の長さは，Euclid (WP4) または Manhatan (WP 3) 距離を使う．終点が指定されていると，問題は同じ距離を使った巡回セールス問題と関係がある．最短巡回セールスマンの巡回路は，i と j の間の距離に i から j への最短 Hamilton 径路の長さを足したものが最小になるような点 i,j の対を決定することによって得ることができる．考えるべき (i,j) の対は $n(n-1)$ 個であるから，Euclid または Manhatan 距離による最短 Hamilton 径路の多項式時間アルゴリズムがあれば，この巡回セールスマン問題についての多項式時間アルゴリズムがある．だから，終点を指定された最短 Hamilton 径路問題は，どちらの距離を使っても NP-難度である．終点が指定されていない問題もやさしくはならない．

問題 WP 1 と WP 2 は，WP 5 と WP 6 の特別な場合 ($L=\infty$) である．WP 1 と WP 2 は NP-難度だから，WP 5 と WP 6 もまた NP-難度である．問題 NP 5 (彩色数 I) は WP 7 に変換できる．このために，配線の集合 W についての交差グラフ $G=(V,E)$ を考えてみよう．V の点は W の辺に対応している．(i,j) は，W の辺 i,j が Euclid 空間で交差しているときに E の辺とする．G の彩色は G のとなりあった点が同じ色をもたないから，G の彩色は W に含まれる線の配線層への割り当てに対応する．同じ色をもつ G の点に対応する W の線は，同じ配線層に割り当てができるし，逆もまた同様である．だから W の線は G が K 色で彩色可能であれば，そのときにのみ K 個に層別できる．交差グラフの彩色問題と配線層割り当て問題が，等価であることは Alkers によって指摘されている[2],[3]．

WP 9 は $r=3$ のとき NP-難度である．対応する交差グラフは $X=\phi$ のときにのみ 3 色で彩色可能である．交差グラフの彩色可能性を 3 に決めることは NP-難度であるので，WP 9 もまた NP-難度である．WP 8, WP 10 それに WP 11 が NP-難度であるかどうか今のところわかっていない．しかし，これらの問題に対する多項式時間のアルゴリズムも知られていない．

4.4 故障箇所検出問題

O. H. Ibarra と S. Sahni[14] は，問題 FDP 1 から FDP 5 は NP-難度であることを示している．T. Ibaraki, T. Kameda と S. Toida[30] は，有向グラフ上のある種の故障箇所検出問題が NP-難度であることを示している．

5. 発見的方法と“普通はうまくいく”アルゴリズム

自動設計分野で生起する興味ある問題の多くは計算的に（多項式時間では，たぶん解けないだろうという意味で）むずかしいことがわかっているので，これらの問題を解くための替りの方法を考えることになる．通常考えられるのは，次の二つである．

1) 実用上計算可能であり，役立つ解を得る発見的アルゴリズムをみつける．後者の性格をもつアルゴリズムは近似アルゴリズムと呼ばれている．われわれは適正で早い（すなわち低次の多項式 $O(n), O(n\log n), O(n^2)$ などの）近似アルゴリズムを求めている．

2) 常に最適解をみつけるアルゴリズムであり，解こうと思っている問題のほとんどに対して実用上，計算可能であるような計算量のアルゴリズムである．このようなアル

ゴリズムは，“普通はうまくいく”アルゴリズムと呼ばれている．線形計画法に対する単体法は，“普通はうまくいく”アルゴリズムの好例である．その最悪の場合の計算量は指数的である．けれども，与えられたほとんどの問題を手ごろな（最悪の場合よりもずいぶん少ない）時間で解くことができる．

5.1 近似アルゴリズム

近似アルゴリズムを評価するときに，二つのものさしを考える．

1) アルゴリズムの計算量

2) 解の質（どのくらい最適解に近いか）

計算量の場合と同様に，ものさし 2) には平均の場合と最悪の場合とが考えられる．近似アルゴリズムには，いくつかの種類がある．アルゴリズムAを問題のクラスPのすべての問題Ⅰの実行可能解を作るアルゴリズムであるとする．$F^*(I)$ を最適解の値とし，$F'(I)$ をAによる解とする．

> **定義** A は，すべてのIに対して $|F^*(I)-F'(I)|\leq K$ であるときに P の絶対近似アルゴリズムであるという．Kは定数．Aは，すべてのIについて $|F^*(I)-F'(I)|/F^*(I)\leq f(n)$ であるときに $f(n)$-近似アルゴリズムであるという．nはIの大きさであり，$|F^*(I)|>0$ であるとする．

5.2 “普通はうまくいく”アルゴリズム

“普通はうまくいく”というアルゴリズムを分類するのはむずかしい，現実的には，そのアルゴリズムの実験を十分に行ってから得られるものである．単体法は，いろんな問題を長年にわたってうまく解いてきたから，そのよさが認められている．このような分類を得るための解析的アプローチは，確率論的分析によらなければならない．R. Karp の論文[16],[17]は，いくつかの NP-難度問題に対してこのような分析を行っている．この分析は最適解を保障するアルゴリズムに限定されるものではない．R. Karp の論文[18]では，Euclid 距離による巡回セールスマン問題に対する近似アルゴリズムを分析して，最適に近い巡回路を作ることが期待できる早いアルゴリズムを得ている．

E. Horowitz らの著作[13]の12章は，NP-難度問題の近似アルゴリズム研究のよい解説である．

6. おわりに

最悪の場合の計算量で考えると，ほとんどの自動設計問題は非常にむずかしい．最適解に近いことが保証された解を得るだけにしても，むずかしいことに変わりはない．これらむずかしい問題に対するアルゴリズムの価値を保証するための最も役立ちそうなアプローチは，確率的分析と実験である．もう一つの役に立つと思われる研究テーマは，計算量的にもっとむずかしい問題に対する並列アルゴリズム（それと関連するハードウェア）を設計することである．

（応用ソフトウェア部　森本　晧夫 訳）

参考文献 [1] M. A. Breuer, “The application of Integer Programming in Design Automation,” *Proc. SHARE Design Automation Workshop*, 1966.

[2] M. A. Breuer (ed.), *Design Automation of Digital Systems, Vol. 1, Theory and Techniques*, Prentice-Hall, 1972.

[3] M. A. Breuer, “Recent Developments in the Automated Design and Analysis of Digital

Systems," *Proc. of the IEEE*, Vol. 60. No. 1, January 1972, pp. 12-27.
[4] N. Christofedes, "Worst-case Analysis of a New Hauristic for a Travelling Salesman Problem," *Mgmt. Science Research Report*, Carnegie Mellon University, 1976.
[5] R. L. Clark, "A Technique for Jmproving Wilability in Automated Circuit Card Placement," *Rand Corp. Report R-4049*, August 1969.
[6] W. M. vanCleemput, *Computer Aided Design of Digital Systems*, 3 volumes, Digital Systems Lab., Stanford University, Computer Science Press, 1976.
[7] G. S. Ehrlich, S. Even and R. E. Tarjan, "Intersection graphs of Curves in the Plane," *Jr. Combin. Theo.*, Ser. B. 21, 1975. pp. 8-20.
[8] M. R. Garey, R. L. Graham and D. S. Johnson, "Some NP-Complete Geometric Problems," *Proc. 8th Annual ACM Symposium on Theory of Computing* ACM., 1976, pp. 10-22.
[9] M. R. Garey, R L Graham and D. S. Johnson, "The Complexity of Computing Sceiner Minimal Trees," *SIAM Jr. Appl. Math.*, 32, 1977, pp. 835-859.
[10] M. R. Garey and D. S. Johnson, *Computers and Intractability-A guide to the Theory of NP-Completeness*, W. H. Freeman and Co., 1979.
[11] A. R. Habayeb, "System Decomposition, Partitioning, and Integration for Microelectronics," *IEEE Trans. on System Science and Cybernetics*, Vol. SSC-i, No. 2, July 1968, pp. 164-172.
[12] E. Horowitz and S. Sahni, "Computing Partitions with Applications to the Knapsack Problem," *JACM.* 21. 1974, pp. 277-292.
[13] E. Horowitz and S. Sahni, *Fundamentals of Computer Algorithms*, Computer Science Press, 1978.
[14] O. H. Ibarra and S. Sahni, "Polynomially Complete Fault Detection Problems," *IEEE Trans. on Computers*, Vol. C-24, March 1975, pp. 242-249.
[15] R. Karp, "On the Reducibility of Combinatorial Problems," in R. E. Miller and J. W. Thatcher (eds.), *Complexity of Computer Computations*, Plenum Press, 1972, pp. 85-103.
[16] R. Karp, "The Fast Approximate Solution of Hard Combinatorial Problems," *Proc. 6th Southeastern Conf. on Combinatorics, Graph Theory, and Computing*, Winniipeg, 1975.
[17] R. Karp, "The Probabilistic Analysis of Some Combinatorial Search Algorithms," *Memo No. ERL-M581*, University of California, Berkeley, April 1976.
[18] R. Karp, "Probabilistic Analysis of Partitioning Algorithms for the Traveling Salesman Problem in the Plane," *Math of Oper. Res.*, 2(3), 1977, pp. 209-224.
[19] U. R. Kodres, "Formulation and Solution of Circuit Card Design Problems Through Use of Graph Methods," In G. A. Walker (ed.), *Advances in Electronic Circuit Packaging*, Vol. 2, Plenum Press, 1962, pp. 121-142.
[20] U. R. Kodres, "Logic Circuit Layout," *The Digest Record of the* 1969 *Joint Conference of Mathematical and Computer Aids to Design*, October 1969.
[21] E. L. Lawler, "Electrical Assemblies with a Minimum Number of Interconnections," *IEEE Trans. on Electronic Computers* (Correspondence). Vol. EC-11, February 1962. pp. 85-88.
[22] E. L. Lawler, K. N. Lavitt and J. Turner, "Mojule Clustering to Minimize Delay in Digital Networks," *IEEE Trans. on Computers*, Vol. C-18, January 1969, pp. 47-57.
[23] G. S. Luaker, "Two NP-Complete Problems in Nonnegative Integer Programming," *Report No. 178, Computer Science Lab.*, Princeton University, 1975.
[24] W. A. Notz, E. Schischa, J. L. Smith and M. G. Smith, "Large Scale Integration; Benefitting the Systems Designer," *Electronics*, February 20, 1967, pp. 130-141.
[25] T. Pomentale, "An Algorithm for Minimizing Backboard Wiring Functions," *Comm. ACM*, Vol. 8, No. 11, November 1965, pp. 699-703.
[26] S. Sahni and T. Gonzalez, "*P*-Complete Apploximation Problems," *JACM*, 23, 1975. pp. 555-565.
[27] S. Sahni, *Mathematical Concepts in Sclence and Engineering*, University of Minnesota, 1980.
[28] M. I. Shamos and D. Hoey, "Closes: Point Problems," *16th Annual IEEE Symposium on Founi of Comp. Sc.*, 1975, pp. 151-163.
[29] A. Yao, "An $O(|E|\log\log|V|)$ Algorithm for Minimum Spanning Trees," *Info. Proc. Letters*, 4(1), 1976, pp. 21-23.
[30] T. Ibaraki, T. Kameda and S. Toida, "NP-Complete Diagnosis Problems on System Graphs" Unpublished Manuscript, 1977.

執筆者紹介 Sartaj Sahni

Kanpur の India 工科大学より電子工学で B. Tech., Cornell 大学より計算機科学で M. S. と ph. D を修得. JACM, JCSS, SIAM Jour. on Computing, IEEE Transactions on Computers などに多数の論文を発表. 興味をもつ分野はアルゴリズムの設計と解析, 並列処理, ネットワーク, および自動設計などである. また「データ構造入門」,「コンピュータ・アルゴリズム入門」の共著者であり, 著書には「離散数字の概念」がある. 現在, Minnesota 大学, 計算機科学科の教授.

Atul Bhatt

Kharagpur の India 工科大学より 1974 年に電子工学で B. Tech. (Hons.) を, 1976年と1979年に Minnesota 大学より電子工学で M. S. と ph. D を取得.
1975年に Roseville の Sperry Uniuac 社に入社, ハードウェア・システム・デザイン部でシステム設計に従事. 新しいコンピュータ, とくにデータベース・コンピュータのアーキテクチュア定義とパフォーマンス分析を行う. 現在は高速メッセージ交換システムのローカル・ネットワークのプロトコルを研究中. 興味をもつ研究分野はコンピュータ・アーキテクチュア, 耐故障計算, コンピュータ・ネットワーク, 自動設計などである.

論文 有限個の先読みによる正則文法の構文解析

Parsing Regular Grammars with Finite Lookahead

Thomas J. Ostrand, Marvin C. Paul, Elaine J. Weyuker

要 約 曖昧でない正則文法のすべてが有限状態機械（たとえそれが先読み機構を備えていたとしても）によって必らずしも解析できるとは限らない．（有限な）k 個の先読みで解析できる正則文法は $FL(k)$ であるという．有限個の先読みで解析できる文法が n 個の非終端記号をもつ場合，先読みは $(n(n-1)/2)+1$ を越えることはない．しかし，これよりもっと多い先読みを必要とする文法が存在する．文法が有限個の先読みで解析可能かどうか判定し，もしそうであれば必要最小限の先読み個数をみつけるアルゴリズムを示す．そのアルゴリズムは $\overline{O}(n^2)$ 個の連立方程式を $\overline{O}(n^4)$ ステップで解く，さらに $FL(k)$ 文法を解析するための2つの方法を示す．一つはあらかじめ作成ずみの大きなテーブルを用いて行う方法であり，実時間で動作する．もう一つの方法は，入力列をその長さに比例した時間で，かつ $3n$ の記憶容量で解析する．

Abstract Not every unambiguous regular grammar can be parsed by a finite state machine, even if a lookahead facility is added to the machine's capabilities. Those which can be parsed with a fixed lookahead of k are said to be $FL(k)$. If such a grammar has n non-terminals, it never needs more than $(n(n-1)/2)+1$ lookahead, and there exist grammars which do require this much. An algorithm is presented for determining whether a grammar is fixed lookahead parsable, and if so, for finding the minimum lookahead needed. The algorithm sets up and solves a set of $O(n^2)$ equations using $O(n^4)$ steps. Two parsing methods for $FL(k)$ grammars are discussed. One uses a large precomputed parsing table, and operates in real time. The second parses an input string in time proportional to its length, while using approximately $3n$ storage locations.

1. はじめに

すべての正則言語（3型言語）は DFA (deterministic finite Automaton：決定性有限機械）によって認識可能であり，また NFA (non-deterministic finite Automaton：非決定性有限機械）をそれと等価な DFA に変換するアルゴリズムが存在する．しかしながら，決定性有限機械で解析できない正則文法が存在する．入力文字列が文法に合致しているかどうかを調べることは，文字列の導出木を決定することと等価である，つまり文法に対応する有限機械に文字列を入力した場合，それがたどる状態系列を決定することと同じである．

正則文法では，すべての生成規則は $A \to bB$ または $A \to b$ のいづれかの形である．ここで A, B は非終端記号，b は終端記号である．われわれの関心は構文解析であるから，この定義は空列を生成する規則を除外している．決定性正則文法とは (A, b) なるすべての組に対して，左辺に A，右辺に b をもつ生成規則がたかだか1個しかないものをいう．決定性正則文法に対する構文解析は入力文字 b に対して常にただ一つの次の状態しかないため，自明である．非決定性正則文法に対する構文解析は，もし文法が曖昧ならば一意に決められないし，曖昧でないとしても入力列すべてを読み込んだ後か，入力列を有限個先読

みすることによって可能である．ここでは，右から左へ有限個先読みすることによって解析できる右線形正則文法のクラスについて論じる．

以下の記述では次の記法を用いる．まず，$G=(N,\Sigma,P,S)$ は非終端記号の集合 N，終端記号の集合 Σ，生成規則の集合 P，および開始記号 S $(S\in N)$ からなる文法を表す．w,x,y,z で終端記号列を，a,b,c で終端記号，非終端記号を A,B,C で表す．ギリシャ小文字 α,β,γ,η は N 上の変数とする．$|x|$ は，文字列 x の長さ（文字数）を示す．$x\in\Sigma^+$ かつ $1\leq i\leq j\leq|x|$ なる x について $x[i:j]$ は x の i 番目から j 番目までの部分列を表す．$x[i]$ は $x[i:i]$ を意味する．$\mathrm{FIRST}_k(x)$ は $|x|\geq k$ ならば $x[1:k]$，$|x|<k$ ならば x である．ここで扱う文法は無駄な非終端記号をもたないものとする．つまり，すべての非終端記号は開始記号から生成可能であり，かつ空列を生成しない．同様にすべての有限機械は無駄な状態がないものとする．つまり，すべての状態は初期状態から到達可能であり，またそこから最終状態へ到達可能であるとする．

正則文法の先読み構文解析の適切なモデルは，最後の k 個の入力文字を蓄える先読みレジスタを備えた有限状態機械である．入力文字列 w の最初の $t-k-1$ 文字を読んだとき，機械は状態 $q(t)$ になり，先読みレジスタには $w[t:t+k-1]$ が蓄えられる．そして次の入力文字 $w[t+k]$ を読むと状態 $q(t+1)$ に移る．有限状態機械のそれぞれの状態は，正則文法の非終端記号に対応している．

ここでは先読み構文解析を有限機械ではなく，文法をもとに定義する．なぜならば，構文解析は文法に関することがらであり，かつ固定長の先読みで解析可能な文脈自由文法の一つである $LL(k)$ 文法[1,3]との対比を強調するためである．

正則文法は，現在の文形式 (sentential form) 中に現れる非終端記号と，入力列の次の k 個の文字になり唯一の生成規則が決まるとき，k 個の先読みによって解析可能である．$G=(N,\Sigma,P,S)$ を正則文法とする．二つの導出

$$S\overset{*}{\Rightarrow}wT\Rightarrow waA\overset{*}{\Rightarrow}wx \quad \text{および} \quad S\overset{*}{\Rightarrow}wT\Rightarrow waB\overset{*}{\Rightarrow}wy$$

ここで，$\mathrm{FIRST}_k(x)=\mathrm{FIRST}_k(y)$

があったとき，常に $A=B$ が成り立つならば，G は k 個 $(k>0)$ の固定長先読み機能をもった有限状態機械によって解析可能である，すなわち G は $FL(k)$ であるという．ある k に存在に対して G が $FL(k)$ ならば G は FL 文法である．

G が FL 文法ならば（G があるに対し $FL(k)$），G に対して必要な先読みの個数は最小の k である．

ここでは，しばしば文法 G をそれと等価な非決定性有限状態機械で表現する．図1に正則文法 G_1 とそれと等価な機械 M_1 を示す．G_1 に対しては（入力列 $abba$ で示されるよ

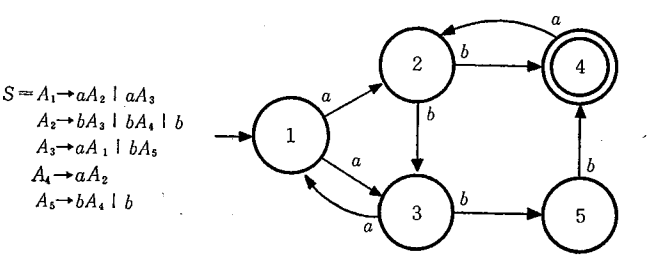


図1 正則文法 G_1 と等価な有限状態機械 M_1

Fig. 1 $FL(4)$ grammar G_1 and equivalent finite automaton M_1

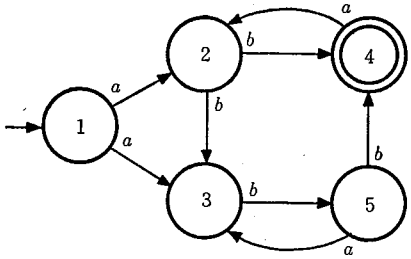


図 2　曖昧な有限状態機械 M_2
Fig. 2　Ambiguous automaton M_2

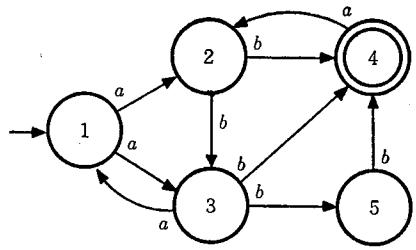


図 3　先読みの数を限定できない有限状態機械 M_3
Fig. 3　Unbounded lookahead automaton M_3

うに）4個の先読みが必要であり，かつそれだけですべての入力列が解析できる．

文法が曖昧であるとき，あるいは先読みの個数が有限におさえられないとき，その文法は固定長先読みによって解析できない．図2の有限状態機械は M_1 を変えたものであるが曖昧である．ab と abb はともに二つ以上の径路を通って受理される．

さらに M_1 を変更して得られる M_3 を図3に示す．M_3 は曖昧でないが，先読みの数を限定できない．M_3 は集合 $a(ba)^*b$ と $a(ba)^*bb$ の和を受理する．受理可能な入力列の先頭の部分列は2つの集合に共通であり，それぞれの集合はまったく別の径路で受理されるため先読みの個数を限定できない．

第2章で別な観点から $FL(k)$ 文法の特徴を示し，文法が FL ならば必要な先読みは非終端記号の単純な関数でおさえられることを示し，この条件に合う文法が存在することを示す．先読みの数が限定されることから任意の正則文法 G がある k に対して $FL(k)$ かどうか決定可能である．第3章では G が FL かどうか決定し，FL の場合に最小の k を求める効率のよいアルゴリズムを示す．

第4章では，$FL(k)$ 文法の構文解析のいくつかの設計について論じる．そのうちの一つは，すでに述べた先読みレジスタを備えた有限状態機械である．有限状態機械は，非終端記号 A と長さ k の入力部分列 z の組からなるエントリをもつ構文解析表 (parsing table) に基づいて動く．エントリは，z の最初の文字を生成するための正しい生成規則を定めている．機械の状態の集合は G の非終端記号の集合に対応し，開始記号 S は初期状態に対応している．機械の動作は構文解析表から次の状態と生成規則を決定し，先読みレジスタを左にシフトし，次の入力文字を右端に読み込む．この機械は言語 $L(G)$ を実時間で認識し，解析できるが構文解析表の大きさは先読み数のべき乗になる．

入力文字列の長さに比例した時間で動作し，表の大きさがさらに少なくてすむ別の構文解析アルゴリズムを示す．このアルゴリズムは，先読み数の上限がない文法の構文解析にも使える．もっとも，この場合の効率はずっと悪くなる．

2.　先読み数の上限

任意の文脈自由文法 G が固定長の先読みで解析可能かどうか，つまり G が $LL(k)$ であるような k が存在するかどうかは決定不能である．しかし，任意の正則文法に対する同様の問題は決定可能である．以下の定理は，FL 文法の決定可能性を示す必要な先読み数の上限を与えるものである．

この定理は曖昧でない文法が相異なる非終端記号から出発して，二つの同じ導出径路(ループ)をもつ場合にのみ，任意の数の先読みが必要なことを述べている．

定理 1

n 個の非終端記号をもつ曖昧でない文法 G は，次の条件が成り立つ場合のみ，固定長の先読みによる解析は不可能である．

非終端記号 A, B と終端記号列 $w_1 \in \Sigma^*$, $w_2 \in \Sigma^+$, $|w_2| \leq n(n-1)/2$ が存在し，

$$S \overset{*}{\Rightarrow} w_1 A \overset{+}{\Rightarrow} w_1 w_2 A' \quad \text{かつ} \quad S \overset{*}{\Rightarrow} w_1 B \overset{+}{\Rightarrow} w_1 w_2 B'$$

ここで，$A' = A$ から $B' = B$，または $A' = B$ かつ $B' = A$ である．

[証明]

導出のループは任意回数繰り返すことができるため，十分条件については明らかである．

必要条件については G を曖昧でなく，かつ FL でない文法とし，n 個の非終端記号をもつとする．とくに，G は $r = [n(n-1)/2]+1$ 個の先読みで解析できない．したがって，

$$S \overset{*}{\Rightarrow} wT \Rightarrow wa\alpha \overset{*}{\Rightarrow} wx, \tag{2-1}$$

$$S \overset{*}{\Rightarrow} wT \Rightarrow wa\beta \overset{*}{\Rightarrow} wy \tag{2-2}$$

なる導出が存在して，$\mathrm{FIRST}_r(x) = \mathrm{FIRST}_r(y)$ かつ $\alpha \neq \beta$ である．G は曖昧でないから $|x| > r$ かつ $|y| > r$ である．$wa\alpha, wa\beta$ から始めて，wT 以降の導出を二つ並行に行ったときの最初の r ステップを考える．それぞれのステップで二つの導出に一つずつ一組の非終端記号が表れる．G には相異なる非終端記号の組は $n(n-1)/2$ しかないため，あるステップではすでに表れた組(それを A, B とする)が再び表れなければならない．

したがって，二つの導出は

$$S \overset{*}{\Rightarrow} wT \overset{+}{\Rightarrow} ww_1 A \overset{+}{\Rightarrow} ww_1 w_2 A' \overset{+}{\Rightarrow} wx \tag{2-1}'$$

$$S \overset{*}{\Rightarrow} wT \overset{+}{\Rightarrow} wu_1 B \overset{+}{\Rightarrow} wu_1 u_2 B' \overset{+}{\Rightarrow} wy \tag{2-2}'$$

のようになる．繰り返しは $r-1$ ステップ以内で起き，かつ $\mathrm{FIRST}_r(x) = \mathrm{FIRST}_r(y)$ であるため $w_1 = u_1$, $w_2 = u_2$ かつ $|w_2| \leq r-1$ となる． □

系

n 個の非終端記号をもつ曖昧でない正則文法 G が，k 個の先読みで解析可能ならば，$k \leq [n(n-1)/2]+1$ である．

[証明]

必要な先読みが $[n(n-1)/2]+1$ より大きいと仮定すると，定理 1 で述べたような非終端記号の組が存在することになり，G は解析不可能になってしまう． □

次の定理は，すべての $n>2$ に対し，n 個の非終端記号をもち，かつ必要な先読みが $[n(n-1)/2]+1$ である文法が存在することを示し，先読みの上限ができる限り小さくできることを示す．

定理 2

すべての $n>2$ に対し正則文法 $G=(N, \Sigma, P, A_1)$, $|N| = n$ が存在し，G は $[n(n-1)/2]+1$ 個の先読みで構文解析できる．

[証明]

$N = \{A_1, \cdots, A_n\}$, $\Sigma = \{a_0, \cdots, a_k\}$, $k = [n/2]$ とする．P は次の規則からなるとする．

$A_1 \rightarrow a_0 A_1 \mid a_0 A_2$,

$A_{n-1} \rightarrow a_1$,

$A_2 \rightarrow a_2$.

$i = 1, 2, \cdots, n-1$ に対し $A_i \rightarrow a_1 A_{i+1}$.

$i=2,3,\cdots,k$ に対し　　$A_n \to a_i A_1$.

$i=1,2,\cdots,k-1$ に対し　$A_{n-i} \to a_{1+i} A_{2+i}$.

$i=1,2,\cdots,k-1$ に対し　$A_{1+i} \to a_{1+i} A_{n+1-i}$.

この文法では開始記号 A_1 からのときのみ，非決定性であることに注意する．右辺に同じものが表れる規則は2個以上存在しないから，この文法は曖昧でない（右辺が同じ生成規則が2個以上存在しないことは，文法が曖昧でないための必要条件であって，十分条件ではないことに注意する）．

はじめに，Gは固定長の先読みで解析できることを示す．まず，固定長の先読みで解析できないとする．

$$A_1 \overset{*}{\Rightarrow} wA \overset{*}{\Rightarrow} ww_1A',$$

$$A_1 \overset{*}{\Rightarrow} wB \overset{*}{\Rightarrow} ww_1B'$$

を定理 1 で述べたような導出とする．ここで，$\{A', B'\}=\{A, B\}$ は導出の過程で繰り返し表れる非終端記号の組のうち最初に表れるものとする．Gの導出規則から

1) 非終端記号 A_1, A_2 は二つの導出の一番最初のステップに表れる．

2) それ以外のすべての組 $\{\alpha, \beta\}$ に対し，その直前に並行に表れる唯一の組 $\{\gamma, \eta\}$ が存在する．$\{A_i, A_j\}$ なる組に対し，その直前に表れる組は，

$i, j \neq 1$ ならば $\{A_{i-1}, A_{j-1}\}$，および $i=1$ ならば $\{A_n, A_{n-i+2}\}$

である．

したがって，$\{A', B'\}$ の直前にどんな組が表れようと，それは，$\{A, B\}$ の直前に表れなければならない．このことは $\{A', B'\}$ が最初の繰り返しであるという仮定に反し，ループする導出が存在する．

Gに対し，$[n(n-1)/2]+1$ 個の先読みが必要であることを示すために，

$$y=a_0a_1^{n-2}a_2a_1^{n-3}a_3\cdots a_1^{n-k}a_k,$$

$$z=a_1^{k-2}a_{k-1}a_1^{k-3}a_{k-2}\cdots a_1a_2,$$

とする．nが偶数ならば，$w_1=ya_1^{k-1}a_kz$, n が奇数ならば $w_2=yz$ を用いると，Σ^* の中では w_1 または w_2 で始まるどの文字列も少なくとも $[n(n-1)/2+1]$ 個の先読みを必要とする．なぜならば，最後の a_2 は $A_n \to a_2A_1$ か $A_2 \to a_2A_n$ のいづれかによって導出され，最初の a_0 は $A_1 \to a_0A_1$ または $A_1 \to a_0A_2$ によって導出されうるからである．□

図4に，$n=8$ の場合のGに相当する有限状態機械を示す．

定理2の文法に対する大きい先読み個数は終端記号の数が比較的多いためであって，非終端記号に対する終端記号の数が少ない割合の文法の場合，そんなに多い先読みを必要としないのではないかと思われるかも知れない．しかし，次の定理は $n+1$ 個の非終端記号

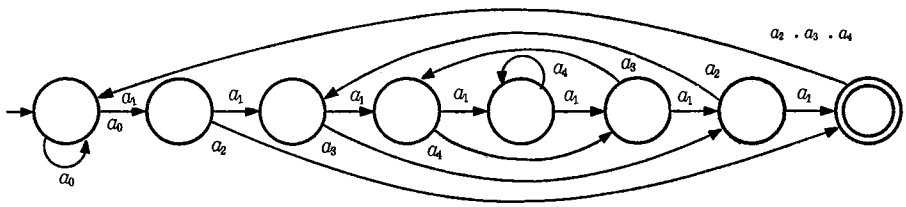


図 4　*FL*(29) 有限状態機械(入力 $a_0a_1^6a_2a_1^5a_3a_1^4a_4a_1^3a_4a_1^2a_3a_1a_2$ 29 文字の先読みを要する)

Fig. 4　$FL(29)$ automaton. Input $a_0a_1^6a_2a_1^5a_3a_1^4a_4a_1^3a_4a_1^2a_3a_1a_2$ requires lookahead 29

および，2個だけしか終端記号だけをもたない曖昧でない文法でさえも n^2 に比例した先読みが必要なことを示している．

定理 3

すべての $n>2$ に対し，正則文法 $G=(N, \Sigma, P, S)$, $|N|=n+1$, $\Sigma=\{a, b\}$ が存在し，Gの構文解析には少なくとも $(n^2+3)/4$ 個の先読みが必要である．

[証明]

$N=\{S, A_1, \cdots, A_n\}$, $\Sigma=\{a, b\}$ とする．Pは次の生成規則からなっているとする：

$$
\begin{aligned}
S &\to aA_1 \mid aA_2,\\
A_{n-1} &\to a,\\
A_3 &\to b \text{ (もし } n \text{ が偶数ならば)}\\
A_i &\to aA_{i+1}, \quad i=1, \cdots, n-1,\\
A_n &\to bA_1,\\
A_{n-k} &\to bA_{k+3}, \quad k=1, 3, 5, \cdots, 2\lfloor n/2 \rfloor - 3.
\end{aligned}
$$

規則 $S\to aA_2, A_1\to aA_2$ があるにもかかわらず，開始記号は，導出の最初にだけしか表れえないため，曖昧でない．定理2と同様の議論でGは固定長の先読みで解析可能であることが示される．

Gに対する最大長の先読みは，nが偶数か奇数かによって次の文字列に対して必要な先読みの長さと同じである．

$$w_1=a^{n-1}ba^{n-4}ba^{n-6}b\cdots a^4ba^2b,$$

$$w_2=a^{n-1}ba^{n-4}ba^{n-6}b\cdots a^3ba.$$

nが偶数の場合，文形式 w_1A_1 と w_1A_n は導出可能であり，もう一つ終端記号を付け加えることによって一意に解析できるようになる．nが奇数の場合，導出可能な形式は w_2A_2 と w_2A_n である．

いづれの場合も終端記号を一つ追加することによって，構文解析の曖昧さを除くことができる．$|w_1|=n^2/4$, $|w_2|=(n^2-1)/4$ であるから必要な先読みの数はそれぞれ $(n^2+4)/4$, $(n^2+3)/4$ となる． □

図5に $\{a, b\}$ に関する11個の状態からなる有限状態機械を表す．

標準のアルゴリズムによって，NFA から DFA を作ると，状態の数は指数関数的に増大する．Meyer と Fischer[4] によればn状態の NFA に対し，それと等価な DFA はどれでも 2^n 個の状態をもつような NFA が存在する．最近，Kintala と Wotschke[2] は，非決定性をきわめて少ししか有していない NFA でさえも DFA に変換されたとき，状態の数は指数関数的に増大することを示した．

$FL(k)$ 有限状態機械はその動作上，ほとんど決定性であるため，これらと等価な決定性機械に比べて指数関数的に簡単であることは興味深い．Kintala らの論文[2]に示された有

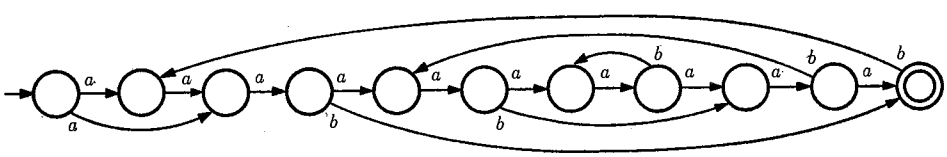


図 5 *FL*26 有限状態機械（入力 $a^9ba^6ba^4ba^2b$ 26 文字の先読みを要する）

Fig. 5 $FL(26)$ automaton. Input $a^9ba^6ba^4ba^2b$ requires lookahead 26

限の言語 $R_k'=\{x\,1\,y|x,\ y\in\{0,1\}^*;\ |x|\leq k-1,\ |y|=k\}$ を受理する NFA は $FL(2k-1)$ であり，$(2k+2)$ の状態を有する．一方，それと等価な DFA は少なくとも 2^{k+1} の状態をもつ．無限の言語 $(R_k'c)^+$ は，基本的に同じ NFA で受理される（最終状態から初期状態への遷移を追加することによって），これらの有限機械もまた $FL(2k-1)$ であり，それらと等価な DFA は少なくとも 2^{k+1} の状態をもつ．

3. 正則文法の先読みの決定

この章では与えられた正則文法Gに対し，それと等価な次のような非決定性有限状態機械Mを用いる．Mの各状態はGの非終端記号に対応し，Mの初期状態はGの開始記号Sに対応している．Gの非終端記号 A_i に対応するMの状態を状態 A_i または単に状態 i と呼ぶ．これらの状態に加えて特別の最終状態が導入される．Mへの入力はGの終端記号列である．Gのすべての生成規則 $A\to aB$ に対し，Mは状態Aで入力 a があると状態Bに移る．$A\to a$ に対して，M は状態Aで入力 a があると，新しく導入された最終状態に移る．このようにして構成される有限状態機械を標準有限状態機械と呼ぶ．（図1の有限状態機械においては最終状態は状態4とマージされている）

導出 $S\overset{+}{\Rightarrow}w\in\Sigma^*$ は初期状態から出発して，それぞれの導出過程で表れる非終端記号に対応した状態を通り，最終状態に到達するまでの径路に対応する．径路に付けられた記号を合わせるとwに一致する．文法は不必要な非終端記号をもたないと仮定したが，それゆえ標準有限状態機械は不必要な状態をもっていない．さらに，空列を導出する生成規則を除外したため，初期状態と最終状態が一致することはない．

言語 $L(G)$ の列wの構文解析は，wを入力したときのMの径路を決定する問題に帰着する．Gが k 個の先読みで解析可能であるとは，Mのある与えられた状態と k 個の入力文字に対して，次の M の状態が定まることに等しい．文法に対する定義と同様に有限状態機械Mは $FL(k)$ である，つまり k 個の先読みで解析可能であるという．Mに対して必要な先読みの個数はMが $FL(k)$ であるような最小の k である．この章で標準有限状態機械が固定長の先読みで解析可能か判断し，もしそうであれば必要な先読み個数を決定するアルゴリズムを呈示する．

このアルゴリズムを構成するには，次の二つの問題に答えなければならない．第1に，入力列の解析時に次に取りうる状態が複数個あるのはどの状態でか．2番目に，次に取りうる状態が複数個あったとき，その中から次の状態を決定するためには何個の入力文字を読まなければならないか．3番目の問題は，このアルゴリズムのキー・ポイントである状態集合の分解能 (resolvability) の概念によって解決される．

定義

状態の組 (p_1, p_2) は，次の条件が成り立つとき，かつそのときのみ，入力 b の下で (q_1, q_2) に**含まれる**．

$$p_1\in\delta(q_1, b),\ \text{かつ}\ p_2\in\delta(q_2, b),$$

または

$$p_1\in\delta(q_2, b),\ \text{かつ}\ p_2\in\delta(q_1, b).$$

(q_1, q_2) は，(p_1, p_2) がある入力に対して (q_1, q_2) に含まれるとき，かつそのときのみ，(p_1, p_2) を**含む**という．

定義

入力列wは，$\delta(q, w)\neq\phi$ のとき，かつそのときのみ，状態qに**適用可能**であるという．

定義

標準有限状態機械の状態集合Qは，相異なる状態 $q_i, q_i \in Q$ と入力列wに対し $\delta(q_i, w) \cap \delta(q_j, w) \neq \phi$ であるとき，**曖昧**であるという．つまり，同じ入力列wに対し二つの異なる状態は次に同じ状態を取りうる．

定義

標準有限状態機械の状態集合Qは，Qがただ一つだけの状態を含むとき，かつそのときのみ，**0-分解可能** (0-resolvable) という．$k>0$ に対し，(空でない集合)Qは次の条件が成り立つとき，かつそのときのみ，***k*-分解可能**という．

1) すべての長さkの文字列はQの**たかだか1個**の状態に適用可能である，

2) $\delta(q, w) \neq \phi$ なるような $q \in Q$, $|w| = k$ が存在する，

3) Qは曖昧でない．

集合Qの分解能は，Qが k-分解可能であるときの最小のkである．もし，そのようなkが存在しないとき，Qは分解不能である．もし，Qが分解能をもつならば長さ $(k-1)$ のある列が存在して，それはQの二つ以上の状態に適用可能であることに注意する．

最初の補題は状態集合Qの分解能はQの要素を1組ずつ調べることによって決定されることを示す．

補題1

状態集合Qは，Qの要素のすべての組の分解能がk以下であり，分解能kをもつある組が存在するとき，かつそのときのみ，分解能 $k>\phi$ をもつ． □

さらに状態の組の分解能は，その組（を構成する状態）の次の状態の分解能および，その組自身の非曖昧さによって決まる．

補題2

状態の組 (q_1, q_2) は次のことが成り立つとき，かつそのときのみ，分解能 $k<\phi$ をもつ．

1) (q_1, q_2) に含まれるすべての状態の組の分解能 $\leq k-1$,

2) (q_1, q_2) に含まれる組のうち，少なくとも一つは $k-1$ の分解能を持つ，

3) (q_1, q_2) は曖昧でない．

さて，入力列wの構文解析，つまりwを入力したとき有限状態機械のたどる状態列の決定問題について考えよう．初期状態はただ一つしかなく，かつすでにわかっている．入力列wのはじめのいくつかを読んだところで有限状態機械は状態qにあり，次の入力文字がbであるとする．次に取りうる状態は $\delta(q, b)$ である．もし $\delta(q, b)$ の分解能が $k-1$ ならば，bに続く長さ $(k-1)$ の文字列が $\delta(q, b)$ のただ一つの状態にのみ適用可能であり，次の状態はk個の先読みによって決まる．

換言すれば，もしk個の先読みで十分に次の状態が決められるならばbに続く長さ$(k-1)$の文字列xは，その状態にのみ適用可能である．もし，次の状態を決定するためにk個の先読みが必要ならば，xの初めの $(k-2)$ 個の文字列は少なくとも二つの状態に適用可能であるはずである．したがって $\delta(q, b)$ の分解能はたかだか $(k-1)$ であり，$\delta(q, b)$ 中には分解能 $(k-1)$ をもつある組が存在する．これらのことから，次のことがいえる．

定理 4

標準有限状態機械 M は，$q \in Q$, $b \in \Sigma$ に対しすべての空でない集合 $\delta(q, b)$ が最大 $k-1$ の分解能をもつとき，かつそのときのみ，k 個の先読みで解析可能である．

Mが決定性の場合に起こる定理の特別なケース：すべての集合 $\delta(q, b)$ はただ一つの要素からなり，したがって，分解能はϕとなる．決定性有限状態機械は一個の先読みで解析

可能である.

アルゴリズム 1: 先読みの長さの決定

入力: それぞれ一個の初期状態，最終状態をもった n 状態からなる非決定性有限状態機械 M の遷移表

出力: M が FL かどうかの識別と FL の場合，必要な先読み長さ

1) (a) M の直接含まれる組をすべてみつける．つまり，一つの状態から一個の文字入力で取りうる次の状態の組 $(q_i, q_j), q_i \neq q_j$ をすべてみつける．リスト L にこれらの組を登録する．

(b) 直接含まれる組がない場合，M は決定性で，$FL(1)$ である．(アルゴリズム終了)

2) リスト L のそれぞれの組 (q_i, q_j) に対して，

(a) (q_i, q_j) によって含まれるすべての組をみつけ，すでに L に存在していないもの L のみに追加する．新しく L に追加された組についても同様に，それによって含まれる組をみつけ L に追加する．これを新規追加がなくなるまで続ける．

(b) ある入力 b に対し，$\delta(q_j, b)$ と $\delta(q_i, b)$ が共通の状態をもつならば，M は曖昧である．(アルゴリズム終了)

3) M が曖昧でないとわかったとき，L のすべての組 (q_i, q_j) の分解能 R_{ij} について次のような連立方程式を立てる．何も含まない組 (q_i, q_j) については，

$$R_{ij}=1$$

$(q_i, q_j)^1, \cdots, (q_i, q_j)^t$ を含む組 (q_i, q_j) に対しては，

$$R_{ij}=1+\max\{R_{ij}^1, \cdots, R_{ij}^t\}.$$

4) 後退代入法によって R_{ij} について解く．もし，両辺に同じ R_{ij} を含む式に出会った場合は，M には二つの並行ループがあることを示し，定理 1 によって固定長の先読みで解析不能である．(このような場合，式の形は $R=1+\max\{R, \cdots\}$ となる．どんな有限の値も $R>R$ を満たさない.)

5) すべての R_{ij} について有限の解が存在すると，M は先読み長さ $k=\max\{R_{ij}\}+1$ で解析可能である. □

アルゴリズム 1 は，一つの最終状態をもつ有限状態機械のみに適用可能である．なぜならば，このアルゴリズムでは異なる最終状態に到達して停止するような曖昧な径路がみつからない．この章の始めに述べたように標準有限状態機械は一つの最終状態のみをもつ.

Appendix 1 にこのアルゴリズムを図 1 の M_1 に適用したときの動作例を示す.

このアルゴリズムは n 状態，m 個の入力文字をもつ FA に対し，たかだか mn^4 に比例する計算時間が必要である．もっとも計算時間がかかるステップは 2) と 4) である．ステップ 2) では，状態の組のリストを作るために mn^4 に比例した計算時間が要る．ステップ 3) ではたかだか $n(n-1)/2$ 個の方程式が立てられる．FA が FL であれば，FA が決定性のときは直接含まれる状態の組は存在しないし，非決定性のときは含まれる組のうち少なくとも一つは分解能 1 をもつ．後者の場合，どの組もそれ自身によって含まれることはないから連立方程式は上三角形になる．したがって，後退代入法は方程式を解くのに十分であり，ステップ 4) での代入と比較の回数はたかだか n^4 に比例する；n^2 の線形方程式を解くのに $\bar{O}(n^{2\log 7})$ を要する Strassen のアルゴリズム[6]と対照的である.

4. 固定長先読みによる構文解析の構成方法

構文解析手法には，$FL(k)$ 文法の特徴を考慮したいろいろなものがある．もっとも速い

構文解析手法は，非終端記号と長さ k の入力列の組み合わせをエントリしてもつ構文解析表に基づくものである．このような表は有限状態構文解析手法における次の状態を示す表と基本的に同じである．非終端記号 A および入力列 x に対するエントリは正しい次の非終端記号（状態）と，入力列の導出に使われる生成規則を含む．この方法の主な欠点は，解析表が大きくなることである． n 個の非終端記号と r 個の終端記号をもつ $FL(k)$ 文法では，エントリの数は nr^k にまでなりうる．実際よくあるように，もし最大長の先読みが必要になるのが文法全体のうちごくわずかであれば，構文解析表は nr^k に比べてかなり小さくなる．必要な先読みが決定されれば構文解析表の構成は簡単であるから，ここではこれ以上議論しない．

2番目の構文解析アルゴリズムは構文解析表を用いる手法よりも時間がかかるが，ずっと少ない記憶容量ですむ．このアルゴリズムの説明では，前と同様に，文法 G に対応する有限状態機械を用いる．構文解析とは，ある入力列に対して機械がたどる状態列を決定することである．このアルゴリズムは，時間 t，$(t+1)$ 番目の入力文字での状態から次に取りうる状態の集合 $N=\delta(q(t), w[t+1])$ を決定する．集合 N のうち，次の $(k-1)$ 個の入力文字が適用可能な一つの状態を，実際の次の状態として識別する．このアルゴリズムでは，N から一つの状態を取り出すために必要なだけしか先読みしない．ほとんどの場合，先読の長さは最大 $(k-1)$ より短くてすむ．

アルゴリズム2では N の各状態から入力列 $\{w[t+2:t+j]\}, j=2, \cdots, k$ に対し到達可能な状態を把握している．つまり，すべての $q \in N$ に対し状態集合 $\delta(q, w[t+2:t+j])$, $j=2, \cdots, k$ が計算される．$FL(k)$ の性質から，ある $j \leq k$ に対し一つの状態集合を除く他のすべての状態集合は，空でなければならない．空でない状態集合の要素 q が次の状態である．

FA は曖昧でないと仮定したため，到達可能な状態の集合を記憶する効率的な方法を使うことができる．$q, q' \in N$ に対し $\delta(q, w[t+2:t+j])$ と $\delta(q', w[t+2:t+j])$ は共通要素を含んではいけない．なぜなら，もし共通要素を含むとすると，入力列 $w[t+1:t+j]$ に対し，FA は状態 $q(t)$ から出発し，二つの相異なる径路を通って共通状態に到達できることになる． アルゴリズム2は時間 $(t+j)$ での到達可能な状態を記憶するためにベクトル $V[1:n]$ を使う．ここで，$q_i \in \delta(q_r, w[t+2:t+j])$ ならば，かつそのときのみ $V(i)=r$ である．

アルゴリズム 2: n 個の状態をもつ $FL(k)$ 有限状態機械 M の固定長先読みによる構文解析

入力: 文字列 $w[1:d]$

出力: M が w を受理した場合，M がたどった状態列 $[q(0), q(1), \cdots, q(d)]$，あるいは w が受理できなかったことの指示

1) {Initialize}

$S \leftarrow q_0$ {q_0 is start state $q(0)$}

$t \leftarrow 0$ {t points to current input}

2) Parseloop

while $(t<d)$

do:

(2.1) **for** $i=1$ **to** n $V(i) \leftarrow 0$

$N \leftarrow \delta(S, w[t+1])$ {N is the set of states accessible

from present state by next input}
for each $q_i \in N$ $V(i) \leftarrow i$
(2.2) {Lookahead to resolve V to one state}
$j \leftarrow 2$
while (V contains at least 2 unequal nonzero entries)
do: for $i=1$ **to** n $Y(i) \leftarrow 0$
for $i=1$ **to** n
if $V(i) \neq 0$ **then** $N \leftarrow \delta(q_i, w[t+j])$
for each $q_r \in N$
$Y(r) \leftarrow V(i)$
fi
$V \leftarrow Y$
$j \leftarrow j+1$
od
(2.3) **if** $V(i)=0$ **for every** $i=1, \cdots, n$
then print 'w not accepted by M'
exit from Algorithm 2
fi
(2.4) {Next state $q(t+1)$ is identified as the only
nonzero number in V}
$S \leftarrow q_r$, where r is nonzero number in V
$t \leftarrow t+1$
print $S, w[t]$
(2.5) **od** {End of Parseloop}
3) {End of input; check whether w is accepted}
if S is not accepting state
then print 'w not accepted by M'

このアルゴリズムは長さnのベクトル V, Y，長さがたかだかnのリストLといくつかの記憶場所を使う．計算時間のほとんどは 2) の Parseloop で使われる．このループは長さdの入力に対し，d回繰り返される．ループの中での注意すべき部分は (2.2) である．(2.2) は各入力文字に対し，最大 $(q-1)$ 回実行される．2番目の **for** ループの内の代入文 $Y(r) \leftarrow V(i)$ はjのそれぞれの値に対し最大n回実行される．なぜならば有限状態機械は1文字読む度に最大すべてのn個の状態を取りうるであろうから，アルゴリズム2は，それゆえnに比例した記憶容量と knd に比例した時間で動作する（dは入力列の長さ）．

状態 q_0 からはじまって入力 w に対する非決定性有限状態機械 FA の動作は，節が状態で，節から出ている枝にはwの中の文字が付けられている木構造で表せる．根は初期状態 q_0 で，根から距離 t にあるすべての節はwの始めの t 個の文字を入力したとき，q_0 から到達可能な状態である．レベル t の節 q の次の節は $\delta(q, w[t+1])$ に属する状態である．すなわち，レベル t から $t+1$ への枝に $w[t+1]$ が付けられている．

曖昧でない FA に対しては，受理列wに対して作られる木は長さ $|w|$ の道を一つだけもつ．$FL(k)$ 有限状態機械に対しては，受理列の木はさらに，次の特性をもつすべての節はたかだか1個の子孫をもち，その子孫は長さ k 以上の径路を含む部分木の根になっている．

アルゴリズム2では，現在の状態の子孫部分木の根と先端の節追跡管理を行い，子孫部分木の先端の節が空となったとき，根を削除する．次の状態はただ一つの部分木の根が残っているとき一意に定まる．先読み処理の間は，選ばれた部分木の内部は保存されていない．先読み処理の間，部分木を完全な形で保持するという方法もあるだろう．この場合，新しい入力文字に部分木を計算する必要がなくなる．

$k\times n$ の配列あるいはリンクされた木構造（どのレベルにおいても広がり n はより大きくならず，また深さはたかだか k である）を用いて実現できる．この方法はアルゴリズム2より速いように思えるが，依然として各入力文字に対し kn に比例した時間がかかる．

アルゴリズム2の使う記憶容量は固定ゆえ，限界のない先読みを必要とする曖昧でない文法の構文解析にも，そのまま使うことができる．そのような文法に対しては，長さ d の入力列中の文字 $w[i]$ を処理するために最大 $d-i+1$ の先読みを必要とするかも知れない．したがって，最大計算時間は nd^2 に比例する．

最後に，Paul Abraham の有意義な示唆に感謝の意を表する．

Appendix 1：アルゴリズム1の動作例

図1の M_1 について動作例を示す．

（1） 状態 1,2 によって直接含まれる組はそれぞれ(2, 3), (3, 4)である．

（2） (2, 3)は(3, 5)と(4, 5)を含む．二つはリストに加えられる．
(3, 4)は(1, 2)を含む．
(3, 5)は(4, 5)を含む．
(4, 5)，(1, 2)は何も含まない．

（3） 方程式は次のようになる．

$$R_{23}=1+\max(R_{35}, R_{45})$$
$$R_{34}=1+R_{12}$$
$$R_{35}=1+R_{45}$$
$$R_{45}=1$$
$$R_{12}=1$$

（4） 解は，$R_{34}=R_{35}=2$，$R_{23}=3$ である．

（5） M_1 への入力列を解析するために必要な最大の先読みは4個となる．

図2の M_2 にアルゴリズムを適用すると，組(2, 3)は入力 a で状態 a に到達するため（ステップ 2)）曖昧であることがみつかる．もとの有限状態機械 M_1 も(1, 4)のように一つの状態に到達する状態の組をもつが，これらの組は自分自身によって含まれないため曖昧でないことに注意せよ。

M_3 にアルゴリズムを適用するとステップ 4）で必要な先読みに限界がないことがわかる．方程式は

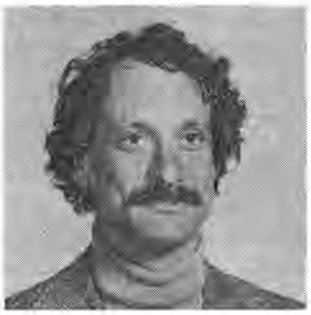
$$R_{23}=1+R_{34}$$
$$R_{34}=1+R_{23}$$

の二つしかなく，R_{23} または R_{34} の代入によって必要な先読みに上限が存在しないことが明らかである．

（応用ソフトウェア部　熊倉 啓介 訳）

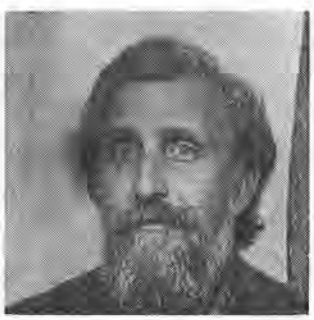
参考文献 [1] A. V. Aho, J. D. Ullman, *The theory of parsing, translation and compiling*, Vol. 1, Prentice-Hall. 1972.
[2] C. M. R. Kintala, D. Wotschke, "Amounts of non-determinism in finite automata", *Acta Informat.* Vol. 13, 1980, pp. 199-204.
[3] P. M. Lewis, R. E. Stearns, "Syntax-directed transduction," *JACM*, Vol. 15, 1968, pp. 464-488.
[4] A. R. Meyer, M. J. Fischer,"Economics of description by automata, grammars, and formal systems", *Proc. of 12th Annual Symp. on Switching and Automata Theory*, IEEE, 1971, pp. 188-191.
[5] M. O. Rabin, D. Scott, "Finite automata and their decision problems", *IBM J. Res. Develop*, Vol. 3, 1959, pp. 114-125.
[6] V. Strassen, "Gaussian Elimination is not Optimal", *Numer. Math.*, Vol. 13, 1969, pp. 354-356.

執筆者紹介 Thomas J. Ostrand



MIT より数学で S. B. を，Pennsylvania 大学から計算機科学で M. S. E. と Ph. D. を取得．1970 年から 1971 年に RCA 社でプログラミングに従事，1971 年から 1978 年に Rutgers 大学の計算機学部で教鞭をとる．1978 年 9 月より，Sperry Univac 社のソフトウェア研究グループの Senior Computer Scientist となる．
研究分野は，セル・オートマトン，プログラムの検証，プログラムの同値変換，有限状態オートマトンを用いた構文解析等．
現在，Sperry Univac 社のソフトウェア工学研究グループの Research Manager で，ソフトウェアの検証とテスト，ソフトウェアの設計法，アルゴリズムの解析等に関心をもっている．

Marvin C. Paul



1952 年に Clarkson 工科大学より BSEE を取得．Bell 研究所および RCA の研究所勤務を経て，Rutgers 大学の計算機科学学科の教授となる．
これまで，コンパイラや有限状態機械の設計を含む多くの分野での研究開発に従事してきた．現在は，アルゴリズム設計の研究に従事．

Elaine J. Weyuker

New York 州立大学 Binghamton 校 (Harpur College) より数学で A. B. を，Pennsylvania 大学より計算機科学で M. S. E. を，Rutgers 大学より計算機科学で Ph. D. をそれぞれ取得．1968 年から 1969 年まで IBM でシステム・エンジニアとして勤務．1969 年から 1975 年まで New York 州立大学 Richmond 校で計算機科学の講師を務める．1977 年より New York 大学の Courant Institute of Mathematical Sciences の助教授．
研究分野はプログラム・テストと検証，プログラム図式，アルゴリズムの設計と解析等．

報告 LSIパッケージの端子と冷却

Terminal and Cooling Requirements for LSI Packages

Thomas S. Steele

要　約　一つの集積回路チップに集積されるディジタル回路量が増すと，チップ・パッケージに必要な入出力端子数も増加する．従来の DIP 型のチップ・パッケージを使って端子数をふやそうとすると形が大きくなってしまい，印刷回路基盤としてのゲート密度を高められない．端子数がチップ・パッケージの面積の関数として決められるか，端子ピッチを一段と狭められるなら解決する．

一方，ゲート密度は熱によっても制限される．端子の満たすべき条件を予測するため Rent の式の係数と指数の値を経験をもとに定めた。パッケージの型，端子ピッチおよび冷却の間の二律背反の関係を図型を用いて視覚化する設計支援法を提案し，実例について述べる．

Abstract　As the level of digital circuit integration on integrated circuit (IC) chips increases, more input-output (I/O) terminals are required on the chip package. More terminals mean a larger conventional package. This can frustrate achievement of higher overall gate density on the printed circuit (PC) board assembly. Chip packages providing terminals as a function of package area or at reduced terminal pitch offer some relief. Gate density may also be thermally limited. Coefficient and exponent values are empirically derived for the Rent equation to predict terminal requirements. A graphical means is developed which can help the designer visualize the trade-offs between package type, terminal pitch, and cooling requirements. Some practical examples are described.

1. はじめに

集積回路技術が進歩するにつれ集積する回路の量が増す．ディジタル回路の場合，これはチップ当たりのゲート数の増加を意味する．また，印刷回路基盤の単位面積当たりのゲート数の増加にもつながる．熱による制限を受けない範囲では，ゲート密度が上がると，動作速度は速まり，原価は下がる．動作速度は接続線長により制限されることが多い．ゲート密度が高まると平均的な接続線長は短くなり，これに対応して遅延時間も小さくなる．また，所与の機能を実現するための印刷回路基盤の数も少なくてすむので原価の削減につながることが多い．

このようにゲート密度が高まると都合のよいことばかりで，MSI から LSI さらに VLSI へ進む技術進歩によって，自動的に実現されるものであるとする考え方がある．しかし，実際は，それほど容易ではない．

チップに集積する回路の量が一定水準をこえると，ゲート密度を現実に制約する次のようなパラメタがある．

1)　製造・修理……集積回路チップのデバイスの大きさはどんどん縮んでいるが，人間の指の大きさはまったく変わらない．チップを装着する機構は製造や修理可能な大きさでなければならない．

2)　接続……チップ当たりゲート数をふやすと，チップ間の相互接続もふえる．この相

互接続に要する空間もゲート密度を制限する.

3) 出力端子……チップのゲート数がふえると相互接続の必要が増し，これを可能とするためチップ・パッケージに必要となる端子もふえる．チップ・パッケージの大きさ，したがって全体のゲート密度は実質上，接続端子の必要数によって決まる.

4) 放熱……高速動作をするチップは熱くなるのが通例である．印刷回路基盤のチップ・パッケージ密度，したがってゲート密度は，放熱系の性能により制限される.

本稿では課題を出力端子と放熱からくるゲート密度の制限にしぼる.

2. 端子制限

パッケージの端子をふやしたい．一方，チップ・パッケージの大きさは小さいままにしておきたい．この二つの相反する要請があるために，ゲート密度は端子によって制限される．端子をどれだけつめて配置できるかは，どんな技術が使えるかによって異なる．SSI や MSI では必要なだけの入出力端子を実現するに当たり経験が先行したが，集積度が高い場合の経験はまだ少ない．端子に対する要求は，個人の経験や観点によって異なる．端子の必要を予測する一つの方法として E. F. Rent の集めたデータから抽出したものがある．Rent の法則は，Landman と Russo が書いたものである．論理回路に必要な信号端子数は，おおむねその論理回路中のゲート数に関連し，次の指数式によって求められる.

$$n=\alpha g^{\beta} \tag{2-1}$$

ただし，n: 信号端子数

g: ゲート数

α: 2.0〜4.0

β: 0.5〜0.7.

この式は，6以上に分割されたコンピュータ型論理の場合にあてはまる[注1]．係数 α と指数 β の値の範囲は彼らの文献[1]に記述されているものである.

係数と指数の値を上述の範囲で選択するのは，この考察の本質に影響しない.

現在の技術では，チップ・パッケージの主流をなす型として，次の2種がある.

1) 周辺端子……端子はパッケージの周辺に沿って配置される．したがって，端子数は端子ピッチとパッケージの周長との関数である.

2) 面端子……端子はパッケージ表面になんらかの方法で分散配置される．したがって端子数は，端子ピッチとパッケージ面積との関数である.

この2種類の型のパッケージにつき，ゲート密度の集積度をふやしたときの効果を Rent

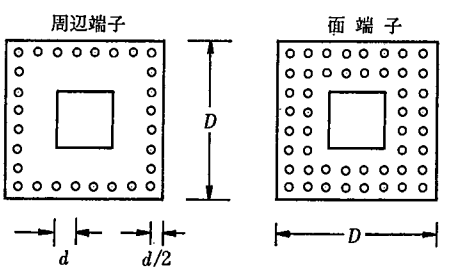


図 1 パッケージ構造

Fig. 1 Package configurations

の法則を援用して調べた．この目的で，物理的構造は図1のように定義し，すべての端子が信号端子であると仮定して，単純化を図った．なぜなら，実際にも大部分の端子は信号端子であり，このように仮定しても原理を表現する上で支障とならないからである．

ここで，

n：端子数

p：外側の行，または列の端子数

m：除外した端子

d：端子ピッチ

D：正方形パッケージの一辺の長さ

A：パッケージの面積

$$D=pd \tag{2-2}$$

$$A=D^2 \tag{2-3}$$

のように定義する．

チップ・パッケージの周辺に端子を配した場合

$$n=4p-4 \tag{2-4}$$

となり，式(2-2)と(2-4)を式(2-3)へ代入すると，

$$A=\frac{d^2(n+4)^2}{16} \tag{2-5}$$

となる．

また，nを Rent の式で置き換えると

$$A=\frac{d^2(\alpha g^\beta+4)^2}{16} \tag{2-6}$$

となる．

したがって，ゲート密度は次のようになる．

$$\frac{g}{A}=\frac{16g}{d^2(\alpha g^\beta+4)^2} \tag{2-7}$$

また，面に端子を配した場合，

$$n=p^2-m \tag{2-8}$$

が得られ，したがって，ゲート密度は

$$\frac{g}{A}=\frac{g}{d^2(\alpha g^\beta+m)} \tag{2-9}$$

となる．

図2は式(2-7)および(2-9)のグラフである．ただし，β の値は 0.5 および 0.7，d は 0.1，m は 4，α は 2.2 とした．プロットしたゲート密度は，集積回路のパッケージだけを考慮したものであり，印刷回路基盤上のパッケージ間隔への配慮はしていない．周辺に端子を配した場合，単位面積当たりのゲート数が増大しないことは明らかである．β の実用的な値，すなわち，$\beta>0.5$ に対しては集積の度合が高まるにつれて，実際のゲート密度は低下している[2]．印刷回路基盤の原価を削減し，相互接続線長を短くしたいという期待に，これは明らかに反する[注2]．かろうじて折り合えるゲート密度は，β の値を非現実的なほど低く，$\beta=0.5$ にしたときに得られる．

これと対照的に面に端子を配置した場合には，すべての実際的なβの値に対し，集積度を高めるにつれてゲート密度が増す．現実的には端子の配置問題からくる制約はあるが，一般的には集積度に比例して実効的なゲート密度を上げたい．この目的から次のような選

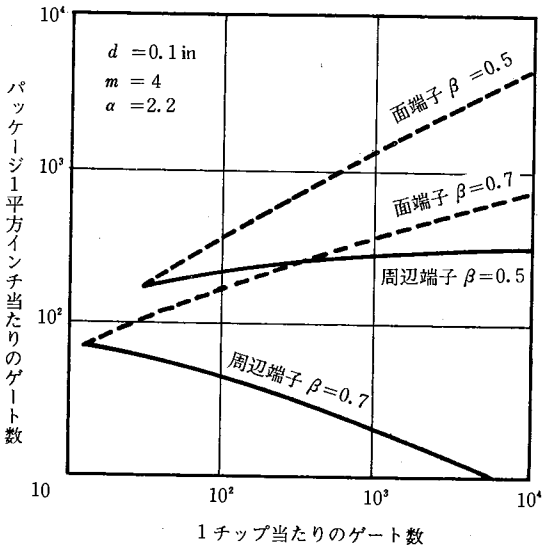


図 2 構造と β の違いによる異なるゲート密度

Fig. 2 Packaged gate density as a function of configuration and β

択項目が考えられる.

1) 論理回路の構成法……論理回路の構成法として, Rent の法則の適用可能なもの以外を採ることもできる. この目的で, バス構造が使われ, ある程度の成功をみた. しかしながら, この種の構造では信号端子を時分割して使う場合もあり, 動作速度犠が牲となる.

2) 端子ピッチ……設計と製造の技術が進歩すると, 端子ピッチを狭くできる. したがって, パッケージの大きさの縮少も可能となる.

3) 面端子……パッケージの面積に比例した数の端子を配置することができる場合には, Rent の法則が適用可能である.

3. 端子の必要条件

実例を示すために Rent の式の α と β に代表的な値を設定する. ゲート・アレイと印刷回路基盤の実例からこの値が抽出できる. 図3はこの目的でデータから作図したもの

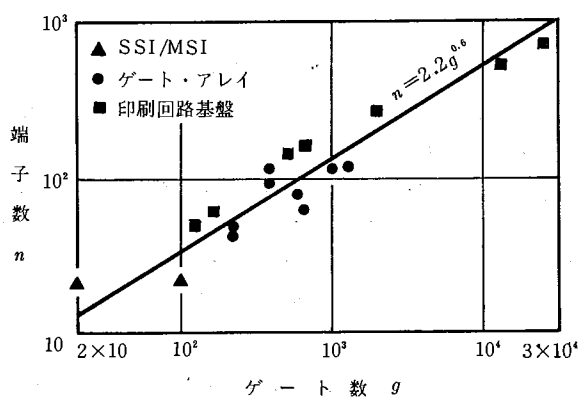


図 3 事例によって抽出した Rent の式の係数と指数

Fig. 3 Empirical derivation of Rent coefficient and exponent

で，目測による傾向線も描き入れた．

この場合 $\alpha=2.2$, $\beta=0.6$ である．一般に同じゲート数に対し，印刷回路基盤よりも，ゲート・アレイの方が端子が少なくてすむ．私見であるがこれは，ゲート・アレイでは実際に使うよりも多くのゲートがあり，ゲートの無駄づかいの傾向が強まっているためと考える．Gordon Moore の考察によると，昔はゲートは高価で，線は安価であったが，今は線が高価でゲートは安価である．その結果 LSI の設計においてゲート数を極小化することの重要性は，相互接続に関する配慮の重要性にくらべて低い．

α と β との値は何が正しいか議論はおそらく果てしがない．しかし，図3から抽き出した値を相当に変えても提示の実例から得られる結論は変わらない．

4. 放熱からくる制限

高速動作をするハードウェアの設計では，放熱は大きな問題となる．一般に高速動作は熱の発生をもたらす．ある特定の半導体をとりあげると，ゲート遅延時間とゲート消費電力とは，ゲート電力遅延積の関係が示すように，二律背反の関係にある．高速動作を実現するには遅延を小さくする必要があり，これを実現するにはゲート電力を大きくする必要がある．この電力の増加は，電力を増加してもゲート遅延時間の増大が生じないところまで行われる．その結果，一定電力のチップでは，遅延の極小となったときのゲート数が最も少なくなる．

実際のハードウェアによる放熱能力例を以下に示す．放熱能力は，印刷回路基盤全体の平均として表現してある．

空冷で 0.5 W/in² 放熱する場合……この放熱能力の例は，10 KECL を用いる現在生産中のコンピュータである．

液冷で 7 W/in² 放熱する場合……この例は，100 KECL と LSI ゲート・アレイを使い，実験に用いたハードウェアによる．

チップ・パッケージの大きさを非常に小さくしても放熱能力以上の放熱を避けるために，チップの間に空間をとる必要があるようでは，チップ・パッケージだけを小さくする価値はほとんどない．高速動作の場合は，素子の消費電力によってゲート数が制限されるので，端子数にゆとりがあっても，必ずしも端子を全部使えるとは限らない．

5. 実　例

端子および放熱によるゲート密度の制限の様子は仮説をたてたうえ，実際のハードウェア上で典型的な例を作ると多少はっきりする．図3からひき出された Rent のパラメタを使う．チップの特性は，

- 正方形
- 周辺端子，または面端子
- 端子ピッチは 50 mil または 100 mil　(1 mil＝1/1000 in)
- 空冷または液冷

パッケージの構成を図4に示す．Rent の式を用いて周辺端子および面端子の場合のゲート密度の式を導く．

ここで，用語は2章で定義したものを適用する．s は隣接するパッケージの外側端子の中心間距離である．

周辺端子の場合，

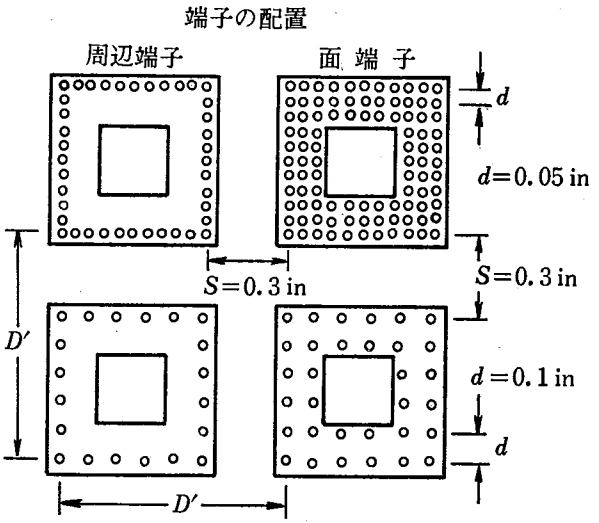


図 4 パッケージの構造と配置

Fig. 4 Package configuration spacing

$$D'=(p-1)d+s \tag{5-1}$$

となる．これに，式(2-4)を代入して，

$$D'=\frac{d}{4}n+s \tag{5-2}$$

となる．式(5-2)を(2-3)へ代入し，nを Rent の式(2-1)で置き換えると，

$$A=\left(\frac{d}{4}\alpha g^{\beta}+s\right)^2 \tag{5-3}$$

となり，ゲート密度は，

$$\frac{g}{A}=\frac{g}{((d/4)\alpha g^{\beta}+s)^2} \tag{5-4}$$

となる．

面端子の場合，式(2-8)を(5-1)へ代入する．

$$D'=\{(n+m)^{1/2}-1\}d+s \tag{5-5}$$

式(5-5)を(2-3)へ代入し，nを Rent の式(2-1)で置き換える．

$$A=[\{(\alpha g^{\beta}+m)^{1/2}-1\}d+s]^2 \tag{5-6}$$

したがって，ゲート密度は，次のようになる．

$$\frac{g}{A}=\frac{g}{[\{(\alpha g^{\beta}+m)^{1/2}-1\}d+s]^2} \tag{5-7}$$

つぎに電力による密度の制限を考えるため，

P：チップの消費電力

w：冷却系による単位面積当たりの冷却能力

とすると，冷却系と素子を決めたときの最小パッケージ面積は，

$$A=\frac{P}{w} \tag{5-8}$$

である．このように電力が密度に課す制限のもとでは，ゲート密度は，

$$\frac{g}{A}=\frac{gw}{P} \tag{5-9}$$

となる．

図5に示したゲート密度のグラフは式(5-4)と(5-7)を2つの端子ピッチについて，プロットしたものである．定数は，次の通りである．

$\alpha=2.2$

$\beta=0.6$

$s=0.3$ in

$d=0.1$ in の場合 $m=4$

$d=0.05$ in の場合 $m=25$

さらに式(5-9)によって，空冷および液冷の温度限界が与えられている．その定数は，

$P=5$ W

$w=0.5$ および 7.0 W/in^2

である．

この曲線は一般に図2の曲線に類似している．しかし，実際に製造するためと受動素子を取り付けることを考えると，チップ・パッケージ間に，0.3in の間隔が必要となる．

面端子では潜在的に，ゲート密度を高くできることは明らかで，とくにゲート数が多いとき著しい．しかし，その性質の利用は放熱能力に左右される．図5の薄墨部は空冷方式，斜線部は液冷方式が利用できる熱領域を定義している．興味をひく例をいくつかみてみよう．1チップ当たり200ゲートの近くで端子ピッチ 0.05in の周辺端子の場合，または端子ピッチ 0.1 in の面端子の場合に，1in^2 当たりほぼ200ゲートの密度が有効となる．1チップの消費電力が5W なので，このゲート密度は液冷によってのみ達成できることは明らかである．0.5 W/in^2 の空冷しか使えないとしたら，端子ピッチ 0.1 in の周辺端子を用いた場合200 ゲートよりもっと複雑なパッケージをめざす意義はほとんどない．なぜなら，温度の制限により，有効なゲート密度は，ほぼ20ゲート/in^2 となるからである．

空冷でチップ電力 5W の場合，端子ピッチ 0.1 in の周辺端子を用いると，73ゲート/in^2 未満のゲート密度で有効である．また，端子ピッチ 0.05in の周辺端子を用いると，230ゲート/in^2 未満のゲート密度で有効である．1チップ当たり1,000ゲートでは，液冷は空

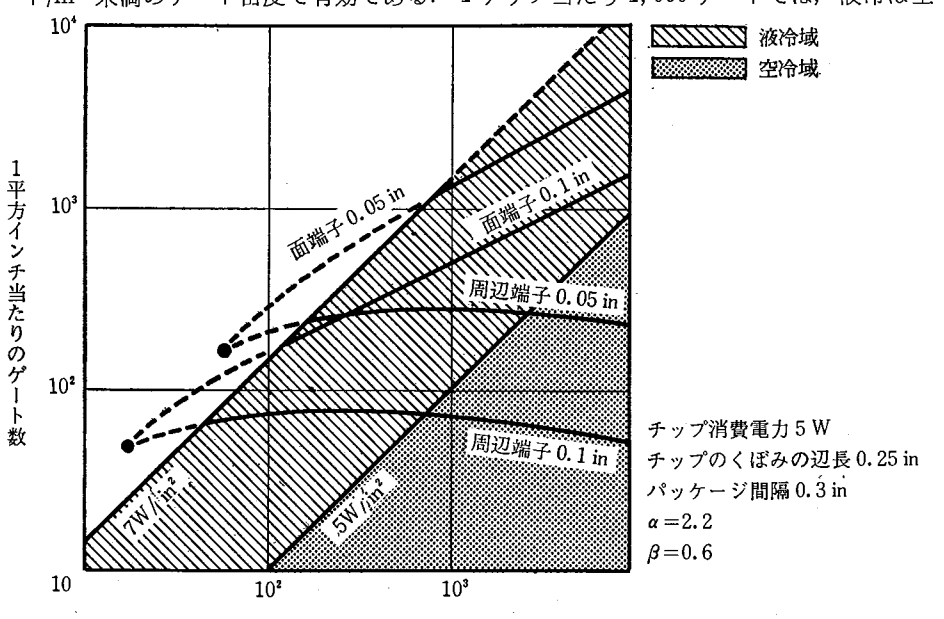


図 5　印刷回路基盤としての実効ゲート密度

Fig. 5　Effective gate density at PC board level

冷に比べてゲート密度を1桁大きくできる．ただし，それには端子ピッチ 0.05 in の面端子とする必要がある．

チップの電力を，低くすると図5の温度による限界線が上方へ動く．チップの電力が360 mW では空冷の上限は図示の液冷のところまで上がる．このようにチップの電力が低い LSI を設計する場合には，ゲート密度を実際に高くするために，ある種の面端子を採用する必要のあることは明らかである．

チップを設計する場合，パッケージ構成や冷却方法の条件に基づき，そのチップに合わせた図をプロットして作り上げると有効である．このようにプロットすると二律背反の様子を視覚化できる．この例として John Balde が示唆した次のパッケージの比較がある．

- 面端子　　端子ピッチ 100 mil
- 周辺端子　端子ピッチ　50 mil
- 周辺端子　端子ピッチ　20 mil

図6は図5を単純化したもので，必要なデータだけを拾ったものに端子ピッチ 20 mil の周辺端子の場合を追加したものである．端子ピッチ 100 mil の面端子パッケージは，1チップ当たり 200 ゲートより上で，端子ピッチ 50 mil の周辺端子パッケージに比べて高い密度が可能である．これらは，いずれも現在使用されているものである．当面の技術としては面端子パッケージを採用し，ピンにより基盤上のスルーホールに直接接続する方式が使用されることと思う．基盤上にスルーホールを位置づけるため，この構造の相互接続の自由度は面実装 (surface mount) の周辺端子パッケージの場合に潜在的に利用可能な相互接続の自由度より少ない．将来，20 mil の端子ピッチの面実装端子が実用化されるようなことになれば，図6に示すように1チップ当たり4500ゲートまでは，周辺端子パッケージの方が，高密度実装が可能となるであろう．面実装の面端子パッケージの端子ピッチがこれに匹敵するようになれば，この利点は失われるであろう．

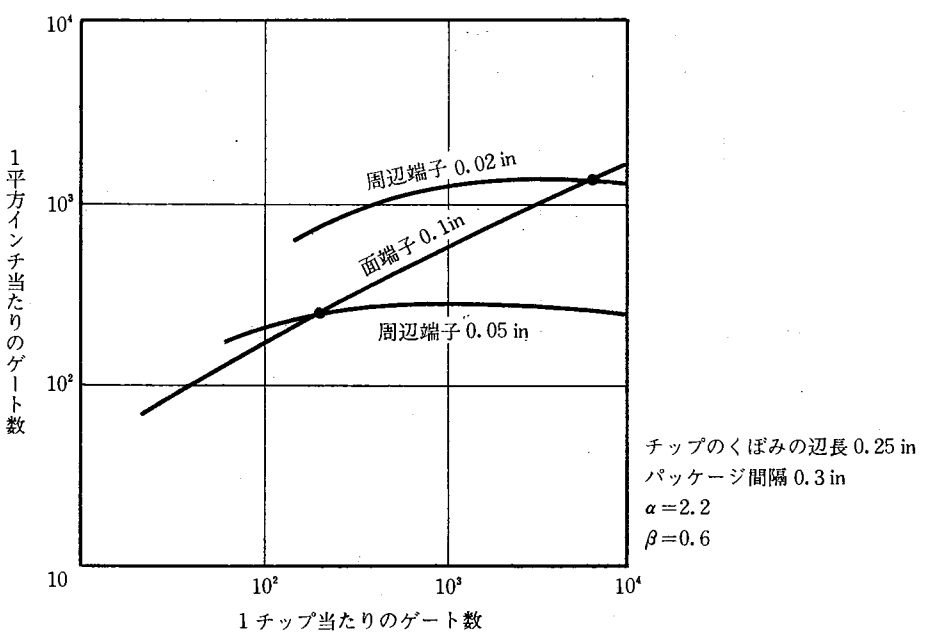


図 6　パッケージの比較例

Fig. 6　Example package comparison

6. ハードウェアの例

図7に示すチップ・パッケージは，1チップ当たりほぼ200ゲートで，図6の交点に該当するものである。大きさの目安として16ピンの DIP パッケージも示した．左側の68端子 JEDEC (Joint Electron Device Engineering Council)] リード線なしA型のパッケージは，端子ピッチ 50 mil の周辺端子の例である．右側の面端子パッケージは同じ大きさで，端子ピッチは100 mil，端子数64のものである．2種のパッケージは空冷することを前提としたもので放熱板によって，約4.5 W まで放熱可能である．チップの電力がこの水準となるとパッケージとしては潜在的に利用可能でもゲート密度を印刷回路基盤全体にわたって利用しつくすのは，空冷の限界があってむずかしい．しかし，チップの電力が2 W 以下なら，おそらくゲート密度は限界まで利用できるであろう．

図8のパッケージは液冷を前提に設計されている．パッケージへの熱伝導接点はチップのくぼみのある側の反対側に設けられている．右側の24端子パッケージは SSI および MSI 用である．左側の54端子パッケージは，ゲート・アレイ LSI 用でチップの消費電力は6 W である．この場合，液冷で潜在的に利用可能となったゲート密度を活用するため，面端子が必要となる．両者とも，端子ピッチは 50 mil である．

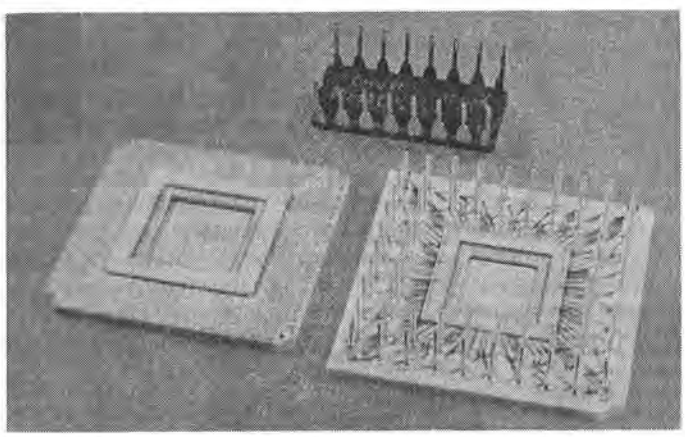


図 7　50 mil 周辺端子パッケージと 100 mil 面端子パッケージ

Fig. 7　50-mil perimeter terminal and 100-mil area terminal packages

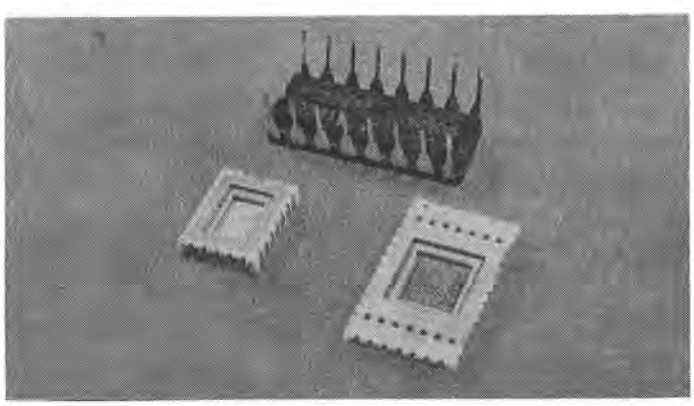


図 8　液冷用パッケージ

Fig. 8　Packages for liquid cooling

7. おわりに

端子ピッチを小さくするには技術的限界がある場合，集積量をふやしてゲート密度を高めようとすると面端子型のチップ・パッケージが必要となる．大電力の高速チップを密に配置する場合，おそらく液体を用いる高密度冷却が必要であろう．ハードウェアの設計者にとっては，図型を使った表示が，パッケージ実装上の二律背反の様子を視覚化する手段となる．

（テクニカル・パブリケーション室　桑野　龍夫 訳）

参考文献 [1] B. S. Landman and R. L. Russo, "On a pin versus block relationship for partition of logic graphs", *IEEE Trans. Comput.*, Vol. C-20, pp. 1470-1471, Dec. 1971.
[2] R. L. Russo, "On the tradeoff between logic performance and circuit-to-pin ratio for LSI," *IEEE Trans. Comput.*, Vol. C-21, p. 149, Feb. 1972.

執筆者紹介 Thomas S. Steele

Wisconsin 大学で電子工学によって1949年 BS，1950年 MS を無線通信の計数的分野の業務をきっかけに，コンピュータの高速計数技術の世界へ入り，プラグ・コンパティブル周辺機器について研究を重ねた．その後，1973年 Sperry Univac 社に入社．現職は Roseville にある Univac Engineering Center のコンピュータ技術グループのマネジャーであり，高速メインフレーム・ハードウェア技術の開発の職責にある．

執筆者追加注 [注1] 5以下の部分に分割されたコンピュータ型論理の場合，一つ一つの部分は機能単位としてまとまりやすい．ゲート数が，同じ場合，機能単位としてまとまりがあるものは，そうでないものより少ない端子数ですむ．この知見は Landman と Russo が Rent のデータに基づいて得たものである[1]．論理の分割はチップ一つのゲート数では足りないときに行う．

[注2] 相互接続線長のチップ全体での平均は論理の構造に左右される．つまり，逐次演算論理のゲートが相互接続される場合は，チップが大きくなっても平均した相互接続線長は短くなる．これは，ふつうチップ内部の相互接続線長が，チップ間の相互接続よりずっと短いためである．

報告 基本変数法による粘性流解析

A Numerical Experiment in Incompressible Viscous Flows by the Primitive Variable Finite Element Method

村 上 紀 佐

要 約 Navier Stokes 方程式によって支配される粘性流体の数値解析は，差分法によって古くから行われており，さまざまな計算手法が開発されてきた．一方，近年構造解析の分野で発達の目覚ましい有限要素法をこの流れ問題に適用し[1],[2],[3],[7]，活発に研究・開発が進められている．流れ問題への有限要素法の適用は，実用性も認識され始め，解析対象も 2 次元問題から徐々に 3 次元問題に移行している．

しかしながら，3 次元の解析となると，モデルの複雑さ（煩雑さ），経済性，コンピュータ能力の限界に阻まれて，順調にはいかない．本稿では，このような状況を踏まえて，有限要素法による経済的な手法によって，3 次元非圧縮性流体の数値実験を試みと検討を行った．

Abstract The Finite Element Method (FEM) is widely used mainly for structural mechanics-applications. The distinguished merits of FEM are in its flexibility for complex geometry and for incorporating "natural" boundary conditions.

For the past ten years, FEM has been applied for flow problems. But the computation by FEM of Navier-Stokes equations in the convection dominant situation has shown some difficulties.

Two main difficulties are; ① The processing of large problems consumes more time and requires more cost than a rumerical approach like the Finite Difference Method (FDM). ② It is difficult to handle incompressibility. The usual finite element technique for continuous equations (div $\boldsymbol{v}=0$) has often added to significant computational complexity.

This report discusses, a new technique proposed at the Symposium on Finite Element Methods in Flow Problems, JUSE and its applications to 3-dimensional viscous incompressible flows. This technique, aimed at implementing an efficient and accurate method for steady viscous flows, is based on the penalty function formulation with incompressibility constraints.

Basic features and the approximation process, with penalty function-like formulations used, are described in Sections 2 and 3. Pseudo transient algorithm adopted to solve time consuming large scale simultaneons equations is described in Section 4. Section 5 represents the obtained numerical results as compared with those obtained by FDM.

The evaluation of this method and suggestions on further numerical experiments are summarized in Section 6.

1. はじめに

実験を試みた手法は，2 次元流れの解析手法として川原[3]らにより提案されたもので，Penalty 関数法[2]に時間進行法を組み合せた方法である．これはまた，非圧縮流体に対しても用いられる質量連続式を，その本質に立ち戻った質量保存式と状態方程式で置き換えた系を解く方法であるともいえる．

基礎方程式は，総和規約を用いて記すると，

$$\frac{D}{Dt}(\rho u_i)=\sigma_{ij,j}+f_i \qquad (i=x,y,z) \tag{1-1}$$

$$\frac{D\rho}{Dt}=0 \tag{1-2}$$

$$p=p(\rho) \tag{1-3}$$

である.

ここで用いた記号は次のとおりである.

ρ: 密度 〔kg·s^2·m^{-4}〕

u_i: i 方向の速度 〔m·s^{-1}〕

i: x, y, z

$\sigma_{ij}=-p\delta_{ij}+\mu(u_{i,j}+u_{j,i})$ 〔kg·m^{-2}〕

p: 圧力 〔kg·m^{-2}〕

μ: 粘度 〔kg·s·m^{-2}〕

$\frac{D}{Dt}$: 物質微分

f_i: i 方向に働く体積力 〔kg·m^{-3}〕

式(1-2), (1-3)から,

$$\frac{Dp}{Dt}=q'(\rho) \tag{1-4}$$

で表される圧力についての時間微分項を導出する. すなわち, 流体挙動を記述する偏微分方程式系は, 以下のようになる.

$$\frac{\partial u_i}{\partial t}=f_i(u_x, u_y, u_z, p) \tag{1-5}$$

$$\frac{\partial p}{\partial t}=f_p(u_x, u_y, u_z, p) \tag{1-6}$$

ただし, f_i, f_p は空間についての 2 次の偏微分式である.

ここで, シミュレーションを行う対象は定常流れであるので, $\frac{\partial u_i}{\partial t}=0, \frac{\partial p}{\partial t}=0$ とおいて, 空間について離散化し連立 1 次方程式を解くというのが常道である. しかし, 3 次元でかつ自由度が $4(u, v, w, p)$ となると, 係数マトリックスの次数が上がり, 効率のよいシミュレーションが行えるとはいいがたい. この困難を回避するため, 従来圧縮性流体の解析によく用いられてきた時間進行法を適用する. この方法は非定常計算の漸近解を定常解とみなすもので, 時間についての積分を工夫すると, 計算量を飛躍的に少なくすることができる.

空間における離散化は, 4 面体 1 次要素[6]を用いた Galerkin 法に拠った. また, 境界条件については, 固定境界は速度の各成分の値または圧力値を指定し, 自由境界は流体表面に垂直に働く力を与えるよう考慮した. さらに, 対称面 (xz 平面に平行な面にかぎるが), 流出口 (yz 平面に平行) の条件を考慮した.

今回はこのような手続きによる解析計算と簡易データ・ジェネレータのプログラムを作成した. ただし, 各物理量は無次元化されたものとして取り扱う. 手法の有効性とともに手法上現れてくる以下の経験的パラメタ (経験的と判断するのは誤解かもしれないが) についてのまとめも試み, プログラムを 2, 3 の例題に適用した.

R_C: 非圧縮性の性質の保存のために使用 $\left(\sim\sqrt{\frac{p}{\rho}}\right)$

Δt: 積分計算をする際の時間刻み幅

α: 質量行列の集中化の補償として使う人工粘性のようなもの

モデルは 3 次元強制くぼみ流と, 3 次元平行平板間流れである.

この手法は, 技巧的・経験的パラメタが存在するという欠点はあるが, 経済的で手軽な

実用プログラムが作成できる手法だといえる．

2. 基礎方程式

運動方程式は次のとおりである．ただし，$\theta=\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}+\frac{\partial w}{\partial z}$ とする．

$$\rho\left(\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}\right)+\frac{\partial p}{\partial x}=\mu\left(\nabla^2 u+\frac{\partial \theta}{\partial x}\right) \tag{2-1 a}$$

$$\rho\left(\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z}\right)+\frac{\partial p}{\partial y}=\mu\left(\nabla^2 v+\frac{\partial \theta}{\partial y}\right) \tag{2-1 b}$$

$$\rho\left(\frac{\partial w}{\partial t}+u\frac{\partial w}{\partial x}+v\frac{\partial w}{\partial y}+w\frac{\partial w}{\partial z}\right)+\frac{\partial p}{\partial z}=\mu\left(\nabla^2 w+\frac{\partial \theta}{\partial z}\right) \tag{2-1 c}$$

さらに，連続式は式(2-2)，状態方程式は式(2-3)である．

$$\frac{\partial \rho}{\partial t}+\frac{\partial \rho u}{\partial x}+\frac{\partial \rho v}{\partial y}+\frac{\partial \rho w}{\partial z}=0 \tag{2-2}$$

$$p=p_0+c^2(\rho-\rho_0) \tag{2-3}$$

ここで，未知数 ρ は密度 [kg·s²·m⁻⁴]，u, v, w は速度ベクトルの成分 [m·s⁻¹]，p は圧力 [kg·m⁻²] である．

物性値（一定とする）としては，μ が粘度 [kg·s·m⁻²]，c が圧力波の伝搬速度 [m·s⁻¹] を表す．

座標系としては，(x, y, z) 直交座標系 [m] を用いた．

流体は非圧縮に十分近いものとして（$\rho\fallingdotseq\rho_0$ 式(2-3)より），式(2-2)を変形する．

$$\frac{\partial p}{\partial t}+u\frac{\partial p}{\partial x}+v\frac{\partial p}{\partial y}+w\frac{\partial p}{\partial z}+c^2\rho\theta=0 \tag{2-4}$$

代表的な速度を U [m·s⁻¹]，長さを L [m]，基準となる密度を ρ_0 [kg·s²·m⁻⁴]，圧力を p_0 [kg·m⁻²] として，これらの量を単位に各式を無次元化する．

まず，無次元量として，

$$x'=\frac{x}{L},\quad y'=\frac{y}{L},\quad z'=\frac{z}{L},\quad R=\frac{\rho_0 UL}{\mu},\quad t'=\frac{t}{L/U}$$

$$u'=\frac{u}{U},\quad v'=\frac{v}{U},\quad w'=\frac{w}{U},\quad p'=\frac{p}{p_0}$$

を用い，各式を変形する．その際，t' をさらに次のように置き換える．

$$t''=\frac{t'}{R}$$

一方，p_0 に関して次の式が成り立っている．

$$p_0=\frac{\mu U}{L}\quad [\mathrm{kg\cdot m^{-2}}]$$

これらの式から求める無次元式は次のように変形できる．

$$\frac{\partial u'}{\partial t''}+R\left(u'\frac{\partial u'}{\partial x'}+v'\frac{\partial u'}{\partial y'}+w'\frac{\partial u'}{\partial z'}\right)+\frac{\partial p'}{\partial x'}=\nabla'^2 u'+\frac{\partial \theta'}{\partial x'} \tag{2-5}$$

$$\frac{\partial p'}{\partial t''}+R\left(u'\frac{\partial p'}{\partial x'}+v'\frac{\partial p'}{\partial y'}+w'\frac{\partial p'}{\partial z'}\right)+R^2\left(\frac{C}{U}\right)^2\theta'=0 \tag{2-6}$$

再び $R_C=R\frac{C}{U}$ とし $P=\frac{p'}{R_C}$ として，式(2-5)，(2-6)を変形する．ただし，みやすくするため，無次元量の ′（ダッシュ）は略す．

$$\frac{\partial u}{\partial t}=-R\left(u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}+w\frac{\partial u}{\partial z}\right)-R_c\frac{\partial P}{\partial x}+\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}+\frac{\partial^2 u}{\partial z^2}+\frac{\partial \theta}{\partial x} \tag{2-7 a}$$

$$\frac{\partial v}{\partial t}=-R\left(u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}+w\frac{\partial v}{\partial z}\right)-R_c\frac{\partial P}{\partial y}+\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 v}{\partial z^2}+\frac{\partial \theta}{\partial y} \tag{2-7 b}$$

$$\frac{\partial w}{\partial t}=-R\left(u\frac{\partial w}{\partial x}+v\frac{\partial w}{\partial y}+w\frac{\partial w}{\partial z}\right)-R_c\frac{\partial P}{\partial z}+\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}+\frac{\partial^2 w}{\partial z^2}+\frac{\partial \theta}{\partial z} \tag{2-7 c}$$

$$\frac{\partial P}{\partial t}=-R\left(u\frac{\partial P}{\partial x}+v\frac{\partial P}{\partial y}+w\frac{\partial P}{\partial z}\right)-R_c\theta \tag{2-8}$$

上式は，任意の初期境界条件の下に解を求めることができる．

3. 方程式の離散化

式(2-7)と(2-8)を次の境界条件を考慮しつつ，有限要素定式化[3]を行う．ここで，境界条件の表記は無次元化後のものである．

$$(u, v, w)=(\bar{u}, \bar{v}, \bar{w}) \qquad (S1_v \text{ で}) \tag{3-1}$$

$$P=\bar{P} \qquad (S1_p \text{ で}) \tag{3-2}$$

$$\sigma_x=l_x\left(-P+2\frac{\partial u}{\partial x}+\left(\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}\right)+\left(\frac{\partial w}{\partial x}+\frac{\partial u}{\partial z}\right)\right)=\bar{\sigma}_x \quad (S2 \text{ で}) \tag{3-3 a}$$

$$\sigma_y=l_y\left(-P+2\frac{\partial v}{\partial y}+\left(\frac{\partial w}{\partial y}+\frac{\partial v}{\partial z}\right)+\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)\right)=\bar{\sigma}_y \tag{3-3 b}$$

$$\sigma_z=l_z\left(-P+2\frac{\partial w}{\partial z}+\left(\frac{\partial u}{\partial z}+\frac{\partial w}{\partial z}\right)+\left(\frac{\partial v}{\partial z}+\frac{\partial w}{\partial y}\right)\right)=\bar{\sigma}_z \tag{3-3 c}$$

ただし，(l_x, l_y, l_z) は表面 $S2$ における法線の方向余弦ベクトルである．

(u, v, w) を $u_i\ (i=x, y, z)$ と書き，α, β, γ が各節点番号を代表しているとして，Einsteinの総和規約を用いて，連立方程式を書き下す．

$$[M_{\alpha\beta}]\dot{u}_{i\beta}=R[A_{\alpha\beta\gamma}]u_{j\gamma}u_{i\beta}+R_c[H_{\alpha\beta}]P_\beta+[K_{\alpha\beta}]u_{i\beta}+[G_{\alpha\beta}]u_{j\beta}+\hat{\Omega}_{i\alpha} \tag{3-4}$$

$$[M_{\alpha\beta}]\dot{P}_\beta=R[A_{\alpha\beta\gamma}]u_{j\gamma}P_\beta+R_c[B_{\alpha\beta}]u_{j\beta} \tag{3-5}$$

ここで，各マトリックス要素は以下のとおりである．

$$M_{\alpha\beta}=\int_v f_\alpha \cdot f_\beta dv,\quad A_{\alpha\beta\gamma}=-\int_v f_\alpha \cdot f_{\beta,j} \cdot f_\gamma dv,\quad H_{\alpha\beta}=\int_v f_{\alpha,i} \cdot f_\beta dv$$

$$K_{\alpha\beta}=-\int_v f_{\alpha,j} \cdot f_{\beta,j} dv,\quad G_{\alpha\beta}=-\int_v (f_{\alpha,i} \cdot f_{\beta,j}) dv,\quad B_{\alpha\beta}=-\int_v f_\alpha \cdot f_{\beta,j} dv$$

$$\hat{\Omega}_{i\alpha}=\int_{S_2} f_\alpha \cdot \bar{\sigma}_j dS \quad (\text{表面に働いている力を示す．})$$

積分 $\int_v dv$, $\int_{S_3} dS$ は各要素内での演算を示す．また，u_i, $\dot{P}$ は，$\dot{u}_i=\frac{du_i}{dt}$, $\dot{P}=\frac{dp}{dt}$ である．さらに $f_\alpha, f_\beta, f_\gamma$ は四面体4節点1次の要素内近似関数を示す．

4. 数値計算手法

ここでは，非定常流体解析の計算手法としては広範に利用されている Lax-Wendroff 法を用いた[7]．空間の離散化が終了した式は，

$$[M_{\alpha\beta}]\dot{u}_i=f_i(u_x, u_y, u_z, P) \tag{4-1}$$

$$[M_{\alpha\beta}]\dot{P}=f_p(u_x, u_y, u_z, P) \tag{4-2}$$

である．式(4-1), (4-2)を変形して，

$$\dot{u}_i=[M_{\alpha\beta}{}^{-1}]f_i \tag{4-3}$$

$$\dot{P}=[M_{\alpha\beta}{}^{-1}]f_p \tag{4-4}$$

を Lax-Wendroff 時間積分の式に代入する.

$$\begin{cases} M_{\alpha\beta}u_i{}^{n+1}=M_{\alpha\beta}u_i{}^n+\Delta t\cdot f_i(u_x{}^{n+(1/2)},\ u_y{}^{n+(1/2)},\ u_z{}^{n+(1/2)},\ P^{n+(1/2)}) & (4\text{-}5)\\ M_{\alpha\beta}P^{n+1}=M_{\alpha\beta}P^n+\Delta t\cdot f_p(u_x{}^{n+(1/2)},\ u_y{}^{n+(1/2)},\ u_z{}^{n+(1/2)},\ P^{n+(1/2)}) & (4\text{-}6)\end{cases}$$

$$\begin{cases} M_{\alpha\beta}u_i{}^{n+(1/2)}=M_{\alpha\beta}u_i{}^n+\dfrac{\Delta t}{2}\cdot f_i(u_x{}^n,\ u_y{}^n,\ u_z{}^n,\ P^n) & (4\text{-}7)\\ M_{\alpha\beta}P^{n+(1/2)}=M_{\alpha\beta}P^n+\dfrac{\Delta t}{2}\cdot f_p(u_x{}^n,\ u_y{}^n,\ u_z{}^n,\ P^n) & (4\text{-}8)\end{cases}$$

式(4-5)から(4-8)までを使用して数値計算に移るわけであるが，この際，連立1次方程式と解くことを省略したいため，左辺の質量行列を集中化する．左辺の質量行列を集中化するということは，回りの格子点に波及すべき運動量を，ある注目している格子点が一身に背負うことになるわけであるから，粘性係数の減少を意味する．その救済のために右辺の質量行列を $\tilde{M}_{\alpha\beta}=(1-\alpha)\overline{M}_{\alpha\beta}+\alpha M_{\alpha\beta}$ の形にする．

α は流速に関しては α_v とし，圧力に対しては α_p として異なる値を与える．

5. 解 析 例

近似法数・値計算手法の検証のため，3次元強制くぼみ流および3次元 Poiseuille 流れの数値実験を行った．

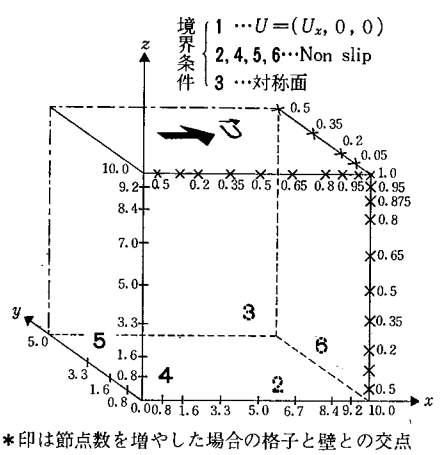
モデルのサイズは，

- 3次元強制くぼみ流（図1） ① 節点数＝ 405 要素数＝1280
- 3次元強制くぼみ流（図1） ② 節点数＝1315 要素数＝5020
- 3次元 Poiseuille 流れ（図2） ③ 節点数＝1315 要素数＝5020

とした．

くぼみ流に対する結果（図3(a)，図4(a)）は，差分による数値実験[4],[5]の数値解（図3(b)，図4(b)）と比較する．これは，傾向やプロファイルを数値実験からつかむという意味で有意義だと考える．

Poiseuille 流れについては，次の章で $R=0$ の場合（図5）を解析解と比較してみる．



＊印は節点数を増やした場合の格子と壁との交点

図 1 3 次元強制くぼみ流の解析領域分割

Fig. 1 Grid pattern for computation and boundary condition (3-dimensional driven cavity flow)

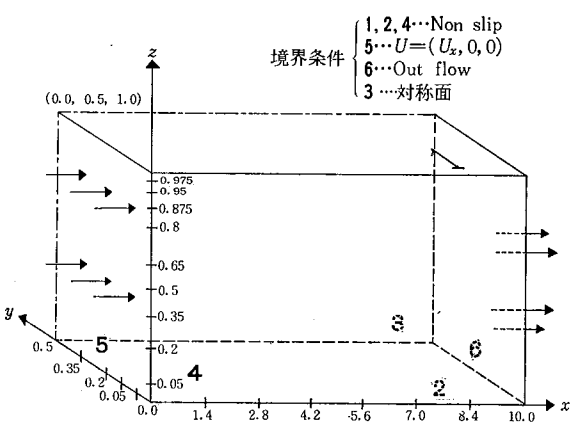
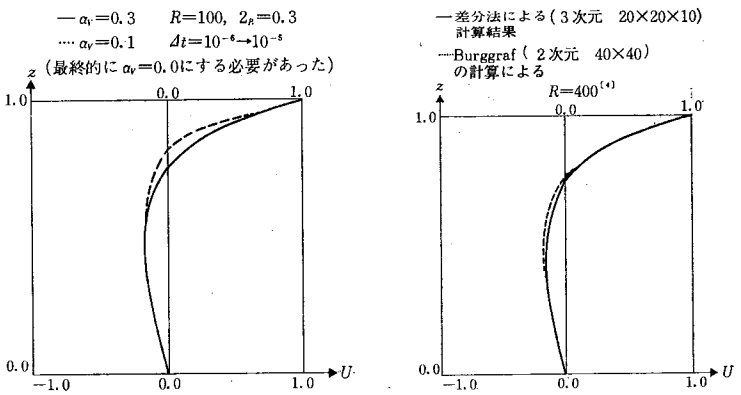


図 2 3 次元 Poiseuille 流れの解析領域分割

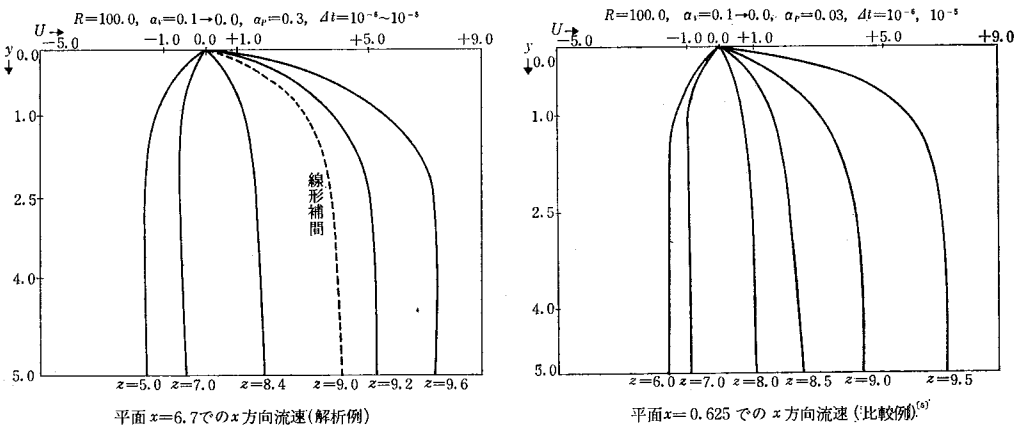
Fig. 2 Grid pattern for computation and boundary condition (3-dinensional Poiseuille flow)



(a)　作成したプログラムでの結果　　(b)　差分による計算結果

図 3　3 次元くぼみ流の計算結果の比較 (z 軸に平行な軸に沿った流速分布)

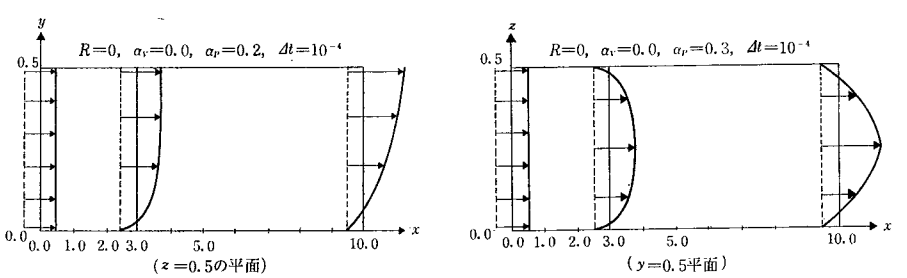
Fig. 3　Numerical results for 3-dimensional driven carvity flow by new algorithm



(a)　作成したプログラムの結果　　(b)　差分による計算結果

図 4　3 次元くぼみ流の計算結果の比較 (yz 面に平行な面における x 方向流速)

Fig. 4　Numerical results for 3-dimensional driven carvity flow by new algorithm



(a)　z 軸に沿った x 方向流速分布　　(b)　y 軸に沿った x 方向流速分布

図 5　3 次元 Poiseuille の流れの数値計算結果

Fig. 5　Numerical results for 3-dimensional Poiseuille flow

実行時間，プログラム・サイズは以下のとおりである（解析フェーズについて）．

モデル	サ イ ズ	SUP 時間 (UNIVAC シリーズ 1100/82 による)	
①	65 kWD	23分	$R=100.0$
②	150 kWD	35分	$R=100.0$
③	150 kWD	5分	$R=0.0$

なお，$R_c=1500$ である．

6. 考察と問題点

6.1 計算結果から

$R=0.0$ の Poiseuille 流れは予想される結果となっている．すなわち，x方向に進むにつれて，y軸（もしくはz軸）に沿った流速が，$x=0$ で一様であったものが次第に変化が生じてきて壁と，壁の2等分点の所で流速が最大となり，壁の所で流速が0となるような放物流になっていく様子が計算で追跡されている．また，出口付近ではきれいな放物線が描けている（図5(a)．流量も各断面で保存されているとみうけられる．構成方程式系が物理的に意味のあるものであること，また，その方程式系の離散化手法や流速を求めるための計算手法が，解析的にも正しい方向で行われているということが確認できたと考える．圧力も流入口から流出口に向けてほぼ線形に減少していっており，実際との対応ができている．しかし，計算で求まった圧力の値は R_c とか P_0 を導入して正規化したものであり，物理的に意味のある数値を得ようとするのはむずかしいと思われる．

同じ，Poiseuille 流れのモデルに対し，Rをつぎつぎと大きくしていったときの流速分布の変化を追跡し，Rの効果が十分現れるかどうか検討すべきであろう．

$R=100.0$ のくぼみ流（図3(a)）は差分法で計算された結果（図3(b)）と比較したわけであるが，流速分布プロファイルの傾向はよくあっている．差分の結果についての詳細な資料がないため，比較は一般的な傾向と，大体の数値で行うわけであるが，実用的なコンピュータの能力の制限等を考えると，広い範囲における流れの様子を一目瞭然につかめるということでは，これで十分ともいえよう．$R\fallingdotseq 0.0$ でも（非線形の項を入れても），この手法で首尾よく計算がなされることがわかった．

いまだ計算結果を整理するツールがないためまとめていないが，渦の発生という観点で，$y=y_1$ なる平面の流れの様子を観察するという課題が残っている．2次元くぼみ流にどのように似るか，興味ある点である．

6.2 R_c, Δt, α について

圧力の時間依存項を陽に出すために $R_c=R\cdot U^{-1}\sqrt{\partial p/\partial\rho}$ (R: Reynolds 数，U: 代表的速度，P: 圧力，ρ: 密度）を書き替えて，$R_c=\dfrac{L}{\nu}\sqrt{\partial p/\partial\rho}$ (L: 代表的長さ，ν: 動粘性係数）を与える必要がある．$\sqrt{\partial p/\partial\rho}$ は，圧力波の速度であり，現在このオーダ程度の値を与えているわけであるが，非圧縮ということを考えれば，じつは $(\partial p/\partial\rho)\to\infty$ であるとが要請される．数値計算上の桁合わせということを考えると $\sqrt{\partial p/\partial\rho}$ を圧力波の速度（水の場合 $10^3\,\mathrm{m\cdot s^{-1}}$ のオーダ）とすると，計算上は $(\partial p/\partial\rho)\fallingdotseq\infty$ なることと同程度の効果になるかと思われるが，注意すべきパラメタである．

Δt（積分の時間ステップ幅）については，(最大速度)$\times\Delta t\times R_c>1$ となると発散する可

能性が大きくなるため注意する必要がある．とくに，積分が陽積分であるため Δt はかなり小さい値（紹介した計算例では，10^{-4}〜10^{-6} で計算すべきである．流速の成長は時刻の進行（繰り返し回数）に関して，対数関数的であるため，とくに解析の初期では，より微小な Δt で急速な変化を緩和し，解がおかしな方向へ加速されないよう防ぐことが必要である．各ステップでの計算量は連立1次方程式を解いたりする手法に比して（節点数）$^{-1}$から（節点数）$^{-2}$程度少ないし，手法自体シンプルであるため時間積分を細かく行うことで，精度を高めようということになる．

α については，図3(a)をみることで大体の意味がつかめると思う．α を小にした結果，動壁からかなり離れた所でも正の流速をもち，壁付近での流速カーブが緩かになってしまっている．つまり，動壁の効果がより遠方まで作用したということである．α の値を加減することで，環境の影響つまり運動量の拡散の度合を制御することができる．すなわ

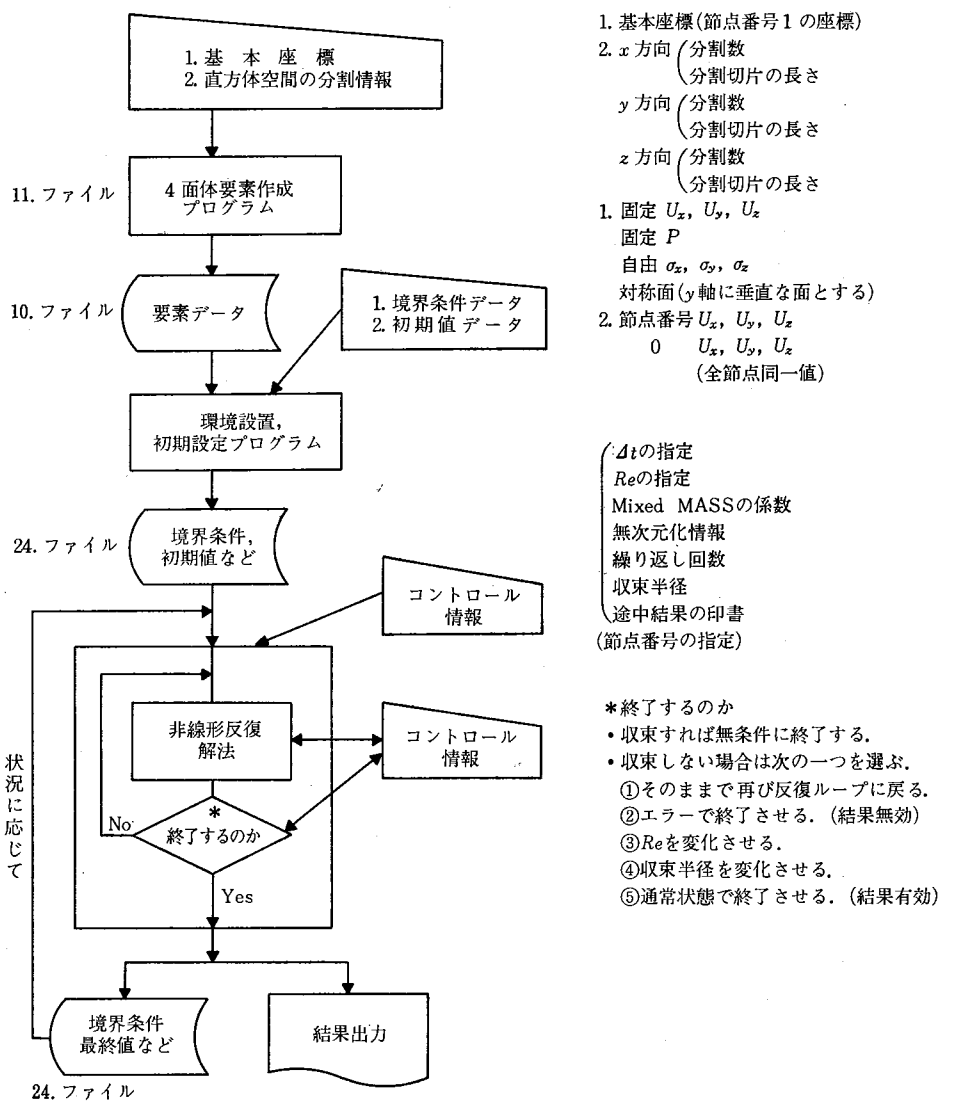


図 6　処理のための流れ図

Fig. 6　Flow chart of the process

ち，場に応じて適切な α を選択しなければならないということを示している．

流速成長（場の動揺の様子）が対数関数的であるということから，解析の初期では $\alpha > 0.0$ とし，定常状態に近くなったときに $\alpha = 0.0$ として，収束への準備に入るという方法が提案されうると考える．

6.3 プログラムについて

計算のための処理の流れ図を図6に示す．

ここで，

1) 反復積分計算において，流速，圧力値の更新，および積分を行う時機
2) 計算結果をみやすく表示する方法
3) メッシュをより細分化するための考慮

などが，問題点として上げられる．

7. おわりに

有限要素法の非構造問題への応用という面で興味深い，流体解析の数値計算を試行してみた．

数値計算法は，通常の変分原理の成立する系に対する方法とは異なり，いくつかのパラメタを導入した簡易計算手法である．

今回行ったようなモデルで，その計算法の有効性は確認できるが，モデルがより複雑になると，現象自体が諸条件に微妙に左右されるものであるため，未解決のことが多いと思われる．

今後のコンピュータの進歩に期待するとともに，利用技術の一担としての流体数値解析が，その発達に追随するよう努力したい．

参考文献

[1] J. J. Connor and C. A. Breddia, (奥村敏恵訳), "流体解析の有限要素法の応用", サイエンス社, 1978.

[2] 川原，平野，"ペナルティ関数法による3次元定常流れの有限要素解析"，第1回流れの有限要素法解析報文集，pp. 93-100, 1979.

[3] 川原，平野，"構造物周辺の風の流れの有限要素解析"，第2回流れの有限要素法解析報文集，1980, pp. 17-24.

[4] K. Goda, "Finite Difference Method for Calculating Two or Three Dimensional Flow in a Cavity", マトリックス解析法研究会論文集，1972, pp. 99-103.

[5] H. Takami : K Kuwahara, "Numerical Study of Three-Dimensional Flow within a Cubic Cavity", *J. of the Physical Society of Japan*, Vol. 37, No. 6, 1974.

[6] 藤野勉，"熱伝導と熱応力"，培風館，1972.

[7] 松田安弘，"環境汚染問題への有限要素法の応用"，情報処理，Vol. 20, No. 7, 1980, pp. 590-600.

執筆者紹介 村上紀佐 (Kisa Murakami)

昭和30年生，52年東京大学理学部卒業，同年日本ユニバック(株)入社．プログラム開発部配属，54年応用ソフトウェア部に転属．ディジタル回路シミュレーション・プログラム担当．

報告 文節構造解析のための接続辞書の構成と付属語の接続検定

Construction of the Postpositional Words Dictionary for Analyzing BUNSETSU and Connection Test of the Postpositionals

稲永 紘之, 小山 憲一

要 約 日本文の構造解析をコンピュータで処理するうえで，付属語の接続辞書は，単語の認定さらに文節の構造解析を行うための辞書として欠かすことはできない．いままでにも多くの構成法が考えられてきた．

本稿では，一般に採用されている接続辞書について問題点を提起し，それを解決し，われわれが用いている接続辞書について，その詳細を報告する．今回編集した接続辞書では，

1) 辞書の容量を小さくすること
2) 解析時間の短縮が図れる構造とすること
3) 辞書保守に対して容易な構造であること
4) 処理プログラムは COBOL を使用すること

などを考慮した．

さらに，該辞書を使用した実験により，期待した結果が得られたことも併せて報告する．

Abstract The dictionary of Japanese postpositional words has grown to be an inevitable tool for computer analyses of Japanese sentences. There have been various methods developed to construct such dictionaries, which help computers recognize words and analyze *BUNSETSU* in Japanese sentences. This paper presents not only the details of the Japanese postpositional word dictionary compiled and now being used by the author, but many problems that were encountered in the process of its designing.

The dictionary that has been compiled this time has the following features:

1) smaller-sized capacity
2) data structures designed to require less time to analyze
3) easier maintenance
4) use of COBOL for computer programs

This paper also discusses the results obtained from actual experiments by the use of the new dictionary; the results were as good as had been expected.

1. はじめに

日本語文を文としてとらえた機械処理を行う場合，入力文が仮名文字文であろうと，漢字仮名交じり文であろうと，分かち書きしてあろうとなかろうと，その形態を問わず，①単語の認定，②文節の構造解析，③文節相互の係り受け関係の解析，の順に処理を進めるのは，一般に採られる方法であろう．通常，文節あるいは文節らしきものが入力文字列に現れたとき，これの解析を行うことにより，単語の認定を行うとともに，文節の構造解析を行うための辞書として，付属語の接続辞書があり，多くの構成法が発表されてきた．われわれは文節単位で入力される仮名文字文を漢字仮名交じり文に変換する研究を進めてき

九州大学工学集報 (1981年8月), 第54巻, 第4号, pp. 399—405 より転載.

たが[1,3,5], 形式名詞や補助用言などは付属語に続けて入力するのが望ましいとの方式を採っている．しかし，個々人の語意識のゆれから切り離されて入力される場合もありうるので，その場合にも対処できるよう付属語の接続辞書の構成を考える必要がある．また，次のステップの文節相互の係り受け関係の解析のためには，各付属語の意味，活用形，助詞の種類などの抽出を行うことも必要である．

本稿では，われわれが用いている接続辞書について，その詳細を述べ，併せて実験結果を報告する．われわれの方法は，基本的には稲永，小西らの文献[1]で報告した方法と同様であるが，処理プログラムをアセンブリ語である FASP から COBOL に変更したことと，今回の実験では接続辞書の妥当性を見るのが主目的であったために，接続辞書のみを用いて対話形式で行ったことが主な変更点であり，細部はいろいろな点で変更したので，改めて報告する．

今回編集した接続辞書では，とくに次の項目を考慮した．

1) 辞書は通常主メモリに常駐させるのでコンパクトにすること
2) 解析（接続検定）時間が短く，能率的処理ができる構成とすること
3) 解析は文節の先頭から末尾に向けて進める方法を採るので，それに合った構成とすること
4) 処理プログラムはすべて COBOL で書くので，それに対応して辞書内容はビット単位の表現は採らないこと
5) 辞書編集のための入力データは，内容の確認が容易でかつ，内容の不備が認められた時には，修正などが容易に行えること

2. 付属語列の特性と接続辞書の構成

2.1 付属語の接続の一般的性質

1文節は次のような形をしている．

1個の自立語＋n 個の付属語　$n=0,1,2,\cdots$

ここで付属語とは，助動詞と助詞であり，助動詞と助詞の接続では，一般に助動詞が助詞に先行する性質がある．しかし，準体助詞と副助詞は助動詞に先行することもある．また，助動詞同士の接続では，ある助動詞が他の助動詞に先行する性質があるが，助動詞間で必ずしも順序付けは行えない．より細かに調べると後続する助動詞は，先行する助動詞の活用形別に異なっているが，一方その後続する助動詞の活用形は，一部の例外を除いては特別の活用形に限定されることはない．したがって，助動詞に先行する可能性のある助詞（準体助詞，副助詞）と，助動詞の接続において，先行する要素と後続する要素の間を有向グラフで表現すれば，複雑な網目状を呈した図ができる．一方，助詞同士の接続は上と同様に図示すれば一応網目状とはなるが，助詞は無活用語であるため，形状は助動詞などのそれに比べると簡単なものとなる．

2.2 文節構造解析と接続辞書

ここでいう文節構造の解析とは，自立語と付属語間，付属語相互間の接続の可否，あるいは文節末に現れた単位が，実際に文節末尾になりうる活用形であるか否かを文法的に解析し，文節をその構成単位である自立語と付属語列に分解することによって，文節の構造を明らかにし，あわせて，各構成単位の品詞，活用形，意味などを抽出することをいう．この構造解析の方法は付属語相互間の接続関係（2.1 節で述べた有向グラフ）を何らかの方法で表した接続辞書の構成法に依存することは明らかであろう．

係り受け辞書を用いる次のステップの処理を考慮して，われわれは接続辞書の有しておくべき項目を

1) 付属語の活用形別の見出し
2) 自立語と付属語間，付属語相互間の接続の可否と接続の条件
3) 助動詞の意味，活用形，助詞の種類，助動詞の活用の型など
4) 用言の活用語尾（必ずしも接続辞書に入れる必要はないが，これも付属語に含める方法が便利である.）
5) 切り離して入力される可能性のある位置の提示

と定めた.

2.3 接続辞書の構成法

文節構造の解析を前から後に向かって進めていくか，逆に後から前に向かって進めていくかに対応して，接続辞書の構成は，見出しの書き方，接続条件の書き方などで全く異なったものとなる．後者の方が接頭語の処理も含めて，文節全体の統一的な扱いができるのだが，われわれは接尾語辞書を引かずに，マッチングのとれる自立語を優先する方針で自立語辞書の見出し語を選択していることと，自立語辞書でマッチングがとれた状態を仮定して実験を行う性質上，本稿では文節構造の解析は前から後に向かって進めていく方法を採用している．後から前に向かって解析を進めていくことにより，自立語辞書を引く回数を少なくする手法を用いたものもあるが[14]，ほとんどの文字（1文字）は接尾語となりうるので，接尾語辞書を完備した状態で接尾語処理も行う場合には，解決すべき問題が残されている.

接続辞書は，接続関係をどういう視点でとらえるかにより，そのデータ構造は異なり，次のような方法が考えられてきた.

2.3.1 付属語のグルーピングによる方法

この方法は，付属語を活用形のあるものについては活用形を一つの単位としてコード化し，文法的に接続機能の類似するものを一つにまとめてグループ化し，コード化された各単位に対して接続可能性のある単位の入っているグループ・コードがポインタとして付いている．これは，九州大学，沖電気，NHK などの初期のシステムが採用したが，辞書内容の一部の修正が他の部分に影響を及ぼし増補改訂が困難なことや，単位によってはグループに重複して存在し，また不要な接続が登録され辞書の増大を招き，さらに接続検定の際，不要なマッチングが生ずるなどのことから，現在この方法を採用しているシステムは見当たらない.

2.3.2 付属語の見出しテーブルと接続行列による方法

この方法は一般に広く用いられている方法であり，見出しテーブルには各用言すべての活用語尾と付属語の活用形ごとの見出しなどを設け，各見出しには，それに接続する付属語の接続行列へのエントリ（たとえば行番号）が入っている．見出しの配列法は，固定長形式，可変長形式，トリー形式などがある．また，接続行列は接続関係を行列（たとえば，行に接続先行単位，列に接続後続単位）に配した2次元テーブルである．ここでは，先行単位と後続単位の見出しのマッチングがとれた時点で改めて接続行列をみるか，あるいは先行単位に対して接続可能な後続単位の候補を接続行列を探してみつけて，改めて実際の入力文字列とその候補の後続単位とのマッチングをとるか，いずれにしても接続可能な後続単位候補だけとのマッチングがとられるわけではないことと，辞書容量の増大を防ぐため2次元の接続行列はビット・テーブルとなっていることなどが特徴である.

2.3.3 その他の方法

以上のほかに，解析を文節末から頭に向かって進める方針の下にビット処理，論理演算を用いた接続ベクトルによる方法[14]，拡張された意味での文節の解析を対象として，語（語列）を分類したカテゴリ間の接続ルール表と，後続の語（語列）に対して先行語（語列）の活用の型，活用形を規定した接続条件表とを併せ用いる方法[15]などがある．

3. 実験システム

3.1 接続辞書の構成とデータ構造

2章でも述べたとおり，文節の構造解析は接続辞書のデータ構造に依存する．したがって，付属語の文法情報をいかに効果的に辞書に反映するか，また解析時間がいかに短縮でき，辞書の増補改訂がいかに容易であるかが重要な問題である．

これら接続辞書としての条件に適合したデータ構造をもつ接続辞書として，2.1 節で述べた付属語の接続関係の複雑さに対応して，次の2種類を用意した．

1) 名詞，用言，助動詞，一部の助詞（助動詞に先行する可能性のある助詞）と，これらに接続する助動詞と一部の助詞との接続関係を表にした辞書

2) 各助詞に対し，その助詞に先行する語の性質（品詞，活用の型，活用形など）を表にした辞書

種類 1) を助動詞辞書と呼び，2) を助詞辞書と呼ぶ．助動詞辞書ではその語に続く可能性のある語を各活用形ごとに直接登録し，助詞辞書では先行する語の性質を規定する情報を登録する．図1に助動詞辞書のデータ構造を，図2に助詞辞書のデータ構造を示す．

3.1.1 助動詞辞書

助動詞辞書は，用言活用語尾テーブル，助動詞・一部の助詞テーブル，接続テーブル，用言エントリ・テーブルからなっている．

用言活用語尾テーブルは，5段活用動詞の各行の活用語尾を除き，各活用語尾と接続テーブルのエントリの対よりなる．5段活用動詞の各行の活用語尾は，別に2次元のテーブ

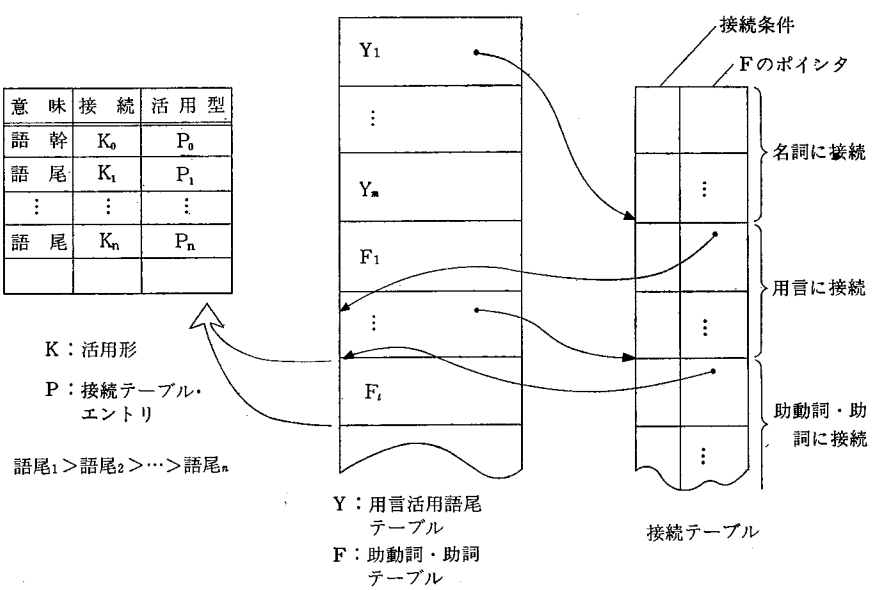


図 1 助動詞辞書データ構造

Fig. 1 Data structure for auxiliary verb dictionary

ルを作って格納している．

助動詞・一部の助詞テーブルでは，各助動詞・助詞ごとに，まず意味，接続の型，活用の型が示され，ついで，活用形が語幹（不変化部分），語尾$_1$，語尾$_2$，…語尾$_n$（無活用語は語幹のみ）のように見出し順に並び，各活用形は，接続テーブルのエントリをもっている．語尾の出現順位は，マッチングの便のため降順（語尾$_1$＞語尾$_2$＞…＞語尾$_n$）となっている．

接続テーブルは，先行の語幹や語尾に対して，後続可能な単位の助動詞・一部の助詞テーブルの位置を示すポインタと，接続条件（活用形の指定）の対を並べたもので，その出現順位は，後続単位の語幹の見出しについて降順となっている．

用言エントリ・テーブルには，各用言の語幹（見出し部は空白）の位置を示すポインタが入っている．

3.2.1 助詞辞書

助詞辞書は，助詞テーブルと先行語(単位)の性質を格納したテーブルからなっている．

助詞テーブルは，一つまたは二つ以上の助詞が連接した見出しと，助詞の種類，見出しの切れ目，先行語性質テーブルのエントリを格納したテーブルである．二つ以上の助詞のつながりでは，特定のつながり方をしたものが多いので助詞のつながりを一つの見出し語として登録している．

先行語性質テーブルを設けた理由は，助動詞から助詞へのつながりは，同じ助詞につながる助動詞の活用形の数が多いため，各助詞に対して先行する語の性質（品詞・活用の型・活用形）を規定することによって，メモリの節約につながるためである．また，助詞テーブルの助詞見出しは，降順とすることにより解析時間の短縮を図っている．

以上の接続辞書の構成をまとめると，図３のとおりである．

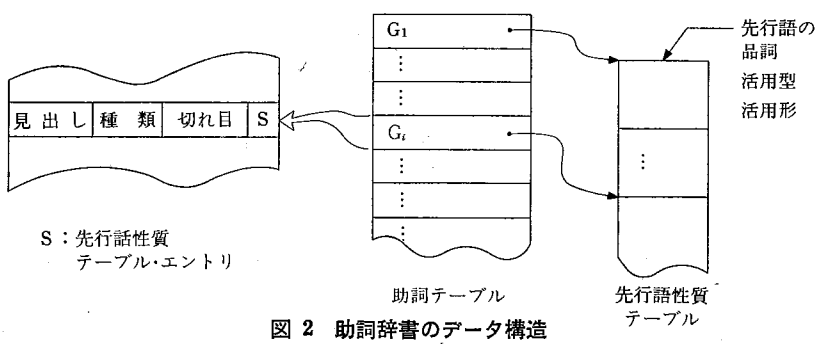


図 2 助詞辞書のデータ構造

Fig. 2 Data structure for particle dictionary

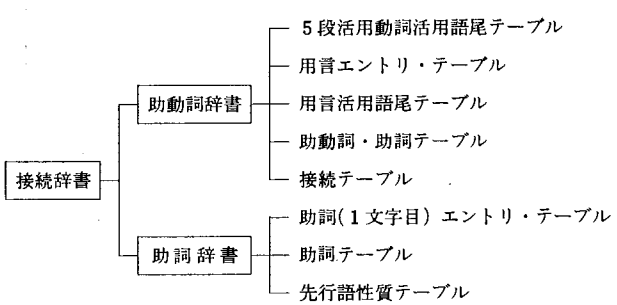


図 3 接続辞書の構成

Fig. 3 Classification of the postpositional words dictionary

タイプ1：助動詞の意味・接続・活用型

名称（注1）		意味（注2）	接続（注2）	活用型（注2）	…

タイプ2：用言，助動詞などの活用形，語幹・語尾，接続する助動詞などの名称

名称（注1）	活用	語幹または語尾	接続助動詞名称 $_1$	接続助動詞名称 $_2$	…
	↑（注3）				

（注1）：品詞または助動詞などの名称
（注2）：表 1 参照
（注3）：表 2 参照

図 4　助動詞などのデータ形式
Fig. 4　Record format of an auxiliary verb

表 1　助動詞の意味・接続・活用型
Table 1　Meaning, connection and conjugation for an auxiliary verb.

コード	意味	接続	活用の型
1	受身	未然形に	動詞活用型
2	可能	連用形に	形容詞活用型
3	自然	終止形に	形容動詞活用型
4	尊敬	連体形に	特別活用型
5	使役	体言，助詞に	無変化型
6	丁寧	語幹に	
7	過去		
8	完了		
9	推量		
A	打消		
B	様態		
C	伝聞		
D	希望		
E	断定		
F	比況		

表 2　活用形
Table 2　Conjugation for an auxiliary verb

コード	動詞型	形容詞型	形容動詞型	特別活用型 無変化型
0（注）				
1	未	未	未	
2	志＊1	志＊2	志＊3	志
3				
4	用	用＊4	用＊5	用
5	音＊6	音＊7	音＊8	
6	音＊9	音＊A		音
7	終	終	終	終
8				
9	体	体	体	体
A	仮		仮	
B	仮＋ば	仮＋ば		仮＋ば
C	命			
F				体＋を

（注）：語幹，あるいは他の活用形に共通な先頭文字，無活用語も含む．
未：未然形　志：志向形　用：連用形　音：音便形
終：終止形　体：連体形　仮：仮定形　命：命令形
＊1：5段動詞　＊2：カロ　＊3：ダロ　＊4：ク
＊5：デ，ニ，ト　＊6：イ音便，ウ音便，促音便
＊7：カッ　＊8：ダッ　＊9：ン　＊A：ウ，ユウ，シウ

3.2　入力データの形式

入力データの作成に当たり，考慮した点は以下のとおりである．

1)　解析は文節の先頭から末尾に向けて進める方法を採用
2)　接続可能性のある語（名称）が容易に確認できること（見出しを数値などでコード化しないこと）
3)　データの増補改訂が容易であること
4)　文節単位認定，後続文節判定への配慮をしたこと
5)　接続に関連した情報を豊富に，かつ単純化すること

助動詞辞書を編集するための入力データの形式を図 4 に示す．ここで，接続助動詞名称 $_i(0\leq i\leq n)$ には，以下の項目が任意に含まれている．

・助動詞，助動詞に先行する助詞の名称
・文節末尾となりうるか否かの情報

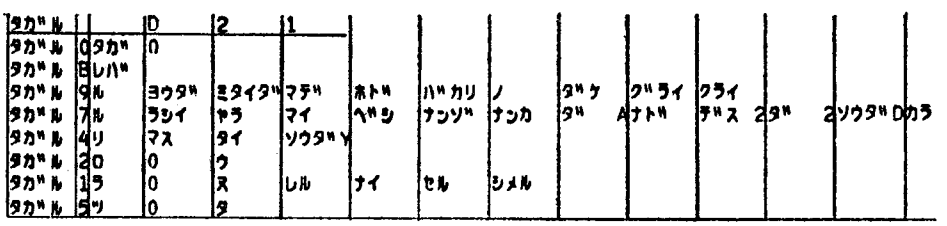
助詞見出し	種類（注1）	先行語の性質1（注2）	先行語の性質2（注2）		見出しの切れ目

（注1）：助詞の種類（表3）
（注2）：先行語の品詞，活用形，優先度（表3）

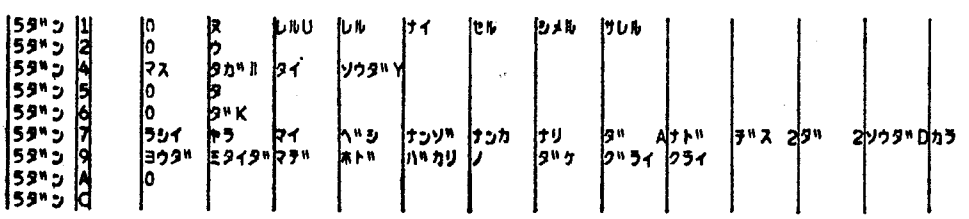
図5 助詞データの形式
Fig. 5 Record format of a particle

表3 助詞の種類と先行語の性質
Table 3 Kinds of particleand type of forward term

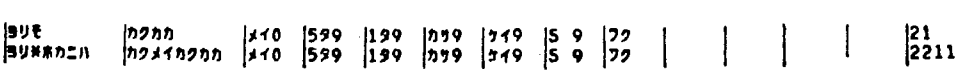
コード	動詞の種類	コード	先行語の品詞，活用の型
カク	格助詞	メイ	名詞
セツ	接続助詞	5タ	5段活用動詞，5段活用動詞型助動詞
フク	副助詞	1タ	1段活用動詞，1段活用動詞型助動詞
カカ	係り助詞	カサ	カ変動詞，サ変動詞，ザ変動詞
カン	間投助詞	ケイ	形容詞，形容詞活用型助動詞
シュ	終助詞	ダ	形容動詞，形容動詞活用型助動詞
（メイ）	（形式名詞）	S	特別活用型助動詞
		F	無変化型助動詞
		カク	格動詞，準体助詞
		セツ	接続助詞
		フク	副助詞



（a） 5段活用動詞用データ



（b） 希望の助動詞「たがる」用データ



（c） 格助詞「より」，係り助詞「も」が連接した「よりも」用データと，格助詞「より」，形式名詞「ほか」，格助詞「に」，係り助詞「は」が連接した「よりほかには」用データ

図6 データ例
Fig. 6 Examples of data

・形式名詞
・後続文節の先頭になりうる可能性
・後続文節の動詞判定情報
・つぎに接続する助動詞の活用形

助詞辞書を編集するための入力データの形式を図5に示す．ここで，先行語の性質i（$\leq i \leq n$）には，以下の項目が任意に含まれている．

・先行する語の品詞
・先行する語の活用の型

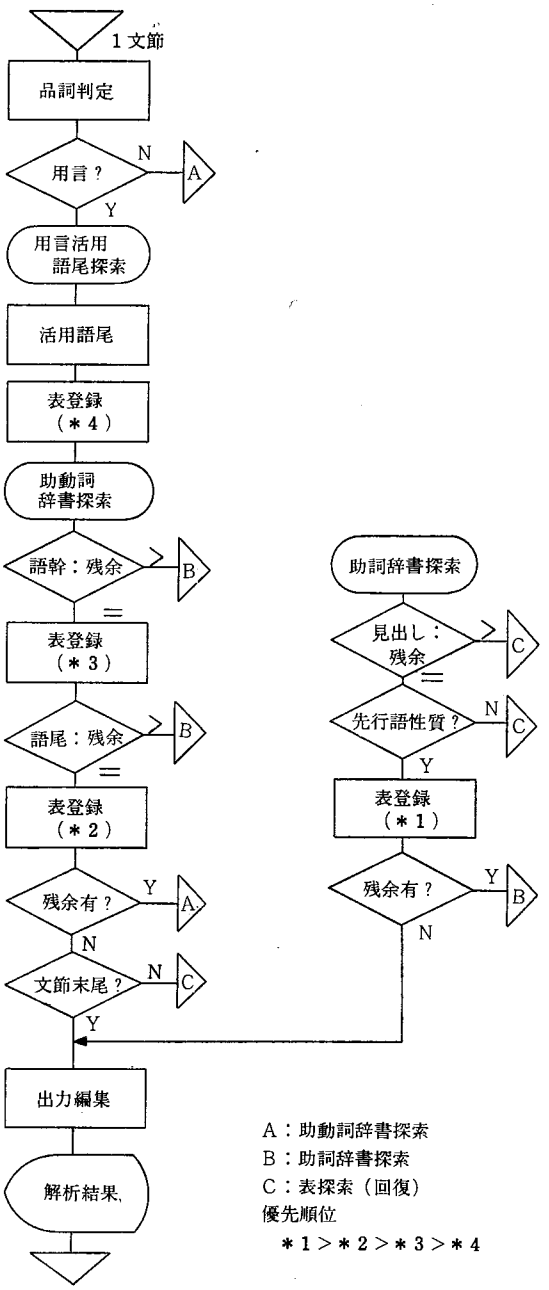


図 7 解析の流れ図
Fig. 7 The flow-chart of analyzing *BONSETSU*

・先行する語の活用形

・2種類以上のマッチングの可能のあるときの優先度

また，助詞見出しは，格助詞あるいは，二つ以上の助詞が連接する可能性を考慮した見出しをもっている．

4. 付属語接続検定の方法

解析の流れ図を図7をに示す．解析での補足説明として，

1) 解析は，①用言活用語尾探索，②助動詞辞書探索，③助詞辞書探索の順で行う．
2) 助動詞辞書探索において，語幹または語尾と付属語残余部とのマッチングは語幹，語尾ともに降順であることから，語幹または語尾＞付属語残余部 の時点で打ち切る．
3) 助詞辞書探索において，助詞見出しと付属語残余部とのマッチングは助詞見出し（二つ以上の助詞が連接した見出しも含む）が文節末条件を満たし，かつ降順であることから，助詞見出し＞付属語残余部 の時点で打ち切る．
4) 解析不可能となった時点で，表へ登録した最新の順位でリカバリする（＊1＞＊2＞＊3＞＊4）．
5) ワースト・ケースとして，用言の活用語尾（同形語がある場合）まで戻り，再度解析を開始する．（助動詞の同形活用語尾に対しても，助詞の 同形語に 対しても 同様である．） すべてを調べ（総当たり方法），文節条件を満たさないとき，そこで解析を中止する（本来であれば自立語探索へバックトラック）．
6) 出力編集では，助動詞辞書探索で登録された表から語幹＋語尾により単語に合成し，助詞辞書探索で登録された表から，二つ以上の助詞が連接した見出しがあるため，切れ目情報により各単語に分離する．

5. 実　験

本章では，3章で述べたデータ構造ならびにテーブル構成をもつ接続辞書の編集と，4章で述べた接続検定方法により接続辞書のみを使用して行った文節構造解析の実験とその結果について述べる．

実験システムの構成を図8に示す．辞書編集プログラム，文節構造解析プログラムの使用言語はともにCOBOLであり，九州芸術工科大学情報処理センターに設置してある MELCOM 700 II の TSS のもとで実行した．入出力は仮名鍵盤付きの端末装置（M 2311 A キャラクタ・ディスプレイ装置）によって行った．

辞書の編集では，あらかじめ3.2節で述べた形式にカードせん孔されたデータ441枚（助動詞データ 306枚，助詞データ 135枚）をディスク・ファイルに登録し，エディタにより

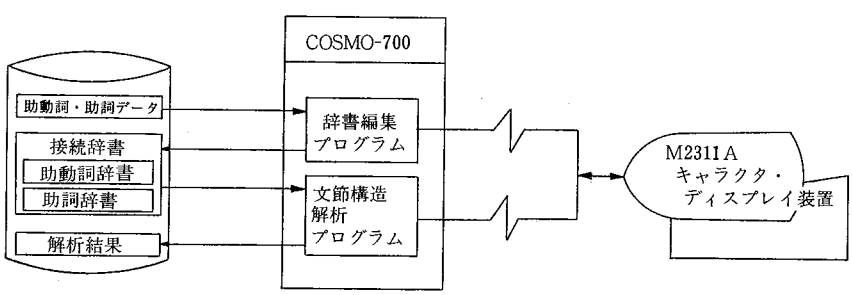


図 8　実験システム
Fig. 8　Experimented system

EXAMPLE 03 :ﾍﾝｷﾖｳｾﾈﾊﾞﾅﾗｽﾞ

SEGMENTATION			NO.
ﾍﾝｷﾖｳｾ	ｻﾍﾝﾄﾞｳｼ	(ﾐｾﾞﾝ ｹｲ)	0050
ﾈﾊﾞ	(ｼﾞﾖﾄﾞｳｼ)	(ｶﾃｲ ｹｲ)	0072
ﾅﾗｽﾞ	(ｼﾞﾖﾄﾞｳｼ)	(ﾚﾝﾖｳ ｹｲ)	0203

EXAMPLE 30 :ｶﾝｶﾞｴﾚﾊﾞｺｿ

SEGMENTATION			NO.
ｶﾝｶﾞｴﾚ	1ﾀﾞﾝﾄﾞｳｼ	(ｶﾃｲ ｹｲ)	0037
ﾊﾞ	(ｾﾂｿﾞｸ ｼﾞﾖｼ)		0028
ｺｿ	(ｶｶﾘ ｼﾞﾖｼ)		0028

EXAMPLE 04 :ｲｶｻﾚﾀｶﾞﾙﾎﾄﾞﾃﾞｼﾀｿｳﾃﾞｽ

SEGMENTATION			NO.
ｲｶ	5ﾀﾞﾝﾄﾞｳｼ ｶ ｷﾞﾖｳ	(ﾐｾﾞﾝ ｹｲ)	0024
ｻﾚ	(ｼﾞﾖﾄﾞｳｼ)	(ﾚﾝﾖｳ ｹｲ)	0322
ﾀｶﾞﾙ	(ｼﾞﾖﾄﾞｳｼ)	(ﾚﾝﾀｲ ｹｲ)	0262
ﾎﾄﾞ	(ﾌｸ ｼﾞﾖｼ)		0182
ﾃﾞｼ	(ｼﾞﾖﾄﾞｳｼ)	(ﾚﾝﾖｳ ｹｲ)	0247
ﾀ	(ｼﾞﾖﾄﾞｳｼ)	(ｼｭｳｼ ｹｲ)	0281
ｿｳ	(ｼﾞﾖﾄﾞｳｼ)	(ｺﾞｶﾝ)	0296
ﾃﾞｽ	(ｼﾞﾖﾄﾞｳｼ)	(ｼｭｳｼ ｹｲ)	0245

図9 解析結果例
Fig. 9 Result of experiment

若干のデータ（補助用言など）の追加を行った．その後，編集プログラムにより，図3に示した構成の接続辞書を作成した．

文節構造解析では，今回は，接続辞書のみを使用しての実験ということから，付属語接続検定の範囲にとどめている．したがって，実験は自立語辞書検索後，自立語の一致がとれた状態を仮定している．すなわち，1文節入力後，自立語部の品詞と自立語部の文字数，さらに5段活用動詞の場合には，活用の行の別を入力情報として与えた．図9に解析結果例を示す．図中の NO. は助動詞，助詞テーブルの行番号を示している．

1番目の例では，動詞「なる」の未然形に打消の助動詞「ぬ」のついた複合助動詞「ならぬ」（連用形「ならず」）が，助動詞「ぬ」の仮定形「ねば」の後続可能なものとして選ばれていることを示している．

3番目の例では，「そう」は，伝聞の助動詞「そうだ」と，様態の助動詞「そうだ」が

あるが，先行の過去・完了の助動詞「た」の終止形に後続可能なものとして伝聞の助動詞が選ばれている．また，連体形と終止形は，同形となるものが多いが，後続する単位によって，いずれであるか定まっている．

6. おわりに

付属語の接続辞書は，その他の文節解析用の機械辞書（接尾語辞書，自立語（基本語）辞書，係り受けの句辞書）と無縁ではありえず，それらの辞書内容やそれらの辞書を用いる処理方式によって規定される面が少なくない．われわれは現在までに，8万4千件の基本語辞書，12万5千件の係り受けの句辞書を蓄積し，接尾語辞書については，チェックの終わった基本語辞書をデータとして，単語内での係り受けをも考慮に入れた形式にするため改訂作業を行っている．今回は，これらの辞書との関わりを勘案し，接続辞書自体に対する要請も考慮して辞書の編集を行い，接続辞書のみを利用した文節構造解析の実験もあわせて行い，所期の目的を一応達成することができた．しかし，辞書容量の圧縮，解析結果の打ち出し形式の改良（助動詞の意味をコードでなく仮名で出すなど）については，十分といえない面もあるので，プログラムの改良を検討している．また，ビット処理を行わないということでは，本稿で述べた方法は有効な方法と思われるが，辞書の容量の面からは，ビット処理を行う方法と比べると若干劣っていることは否めない．本稿では，文節単位入力に対する付属語の接続検定実験について報告したが，ここで用いた接続辞書と他の辞書との有機的なつながりを考慮した総合的な性能・評価は稿を改めて報告したい．

日頃ご指導ご鞭撻頂く吉田将九州大学教授に深甚の謝意を表する．

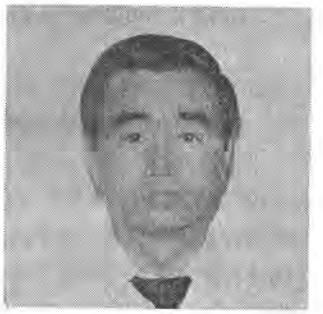
参考文献 [1] 稲永，小西，“機械による文節構造の解析について”，電気四学会九州支部連合大会論文集，1973.
[2] 浜田，和智，“機械による文節構造の解析”，九大・工卒論，1978.
[3] 栗原，稲永，“カナ漢字変換〔I〕”，九大工学集報，Vol. 42, No. 6, 1970.
[4] 相沢，他，“計算機によるカナ漢字変換”，NHK 技研，Vol. 25, No. 5, 1973.
[5] 稲永，小西，“カナ文字文の機械処理のための辞書について”，電子通信学会技報 EC76-45, 1976.
[6] 木村，他，“日本語入力用カナ漢字変換システムの試作”，情報処理，Vol. 17, No. 11, 1976.
[7] 河田，他，“ミニコンピュータを用いたカナ漢字変換システム”，電子通信学会技報 PRL 76-47, 1976.
[8] 勝部，他，“カナ漢字変換の一方法”，情報処理，Vol. 18, No. 7, 1977.
[9] 長尾，他，“国語辞書の記憶と日本語文の自動分割”，情報処理，Vol. 19, No. 6, 1978.
[10] 牧野，他，“べた書き文の分かち書きと仮名漢字変換”，情報処理，Vol. 20, No. 4, 1979.
[11] 森，他，“かな漢字変換”，情報処理，Vol. 20, Nol. 10, 1979.
[12] 田中，吉田，“かな漢字変換による日本語入力システムに関する調査”，九大工学集報，Vol. 53, No. 4, 1980.
[13] 吉村，日高，吉田，“日本語の構文分析—文節数最小法による文節分析アルゴリズムとその能率”，九大工学集報，Vol. 54, No. 2, 1981.
[14] 大河内，他，“仮名漢字変換のための文法解析”，情報処理学会技報計算言語学 25-4, 1981.
[15] 首藤，楢原，吉田，“日本語の機械処理のための文節構造モデル”，電子通信学会論文誌(D)，J 62-D, 12, 1979.
[16] 松村，“日本文法大辞典”，明治書院，1971
[17] 松村，“助詞・助動詞詳説”，学燈社，1969

執筆者紹介　稲永　紘之 (Hiroyuki Inanaga)

1965年九州大学工学部電子工学科卒業．1970年同大学工学研究科博士課程退学．同年九州大学芸術工科大学講師となる．1967年より仮名漢字変換を中心とする日本語文の機械処理に関する研究に従事．1975年電子通信学会米沢賞受賞．電子通信学会および情報処理学会会員．

小山　憲一 (Kenichi Oyama)

1969年東京農工大学工学部卒業．1970年日本ユニバック(株)入社，現在，プロダクト・サポート統括二部開発1部日本語ソフトウェア開発グループに所属．



追 補 米国における商用暗号化システムの標準化

Jim Nelson
(Communication Systems, Roseville)

前号（第2号）の掲題の論説（The Development of Commercial Cryptosystem Standards）につき，多数のご照会をいただきました．その多くは，論説の発表時期1980年10月（CRYPOLOGIA 誌）以降の標準化団体の活動状況を問うものでした．編集部では，著者 Jim Nelson から同論説への追加となる情報を得ましたので紹介します．（編集子）

以下は，1982年3月15日現在の暗号化システムの標準の状況である．

■米国政府情報処理標準化（ANSI/X3)…

米国規格協会の コンピュータと情報処理委員会X３では，以前データ・リンクの暗号化規格グループX3S38により用意された規格を受けた X3T1（暗号化）のグループによって，専ら標準が作られている．X3T1 の結成以来，ANSI のデータ暗号化アルゴリズム規格 X3.92が米国連邦規格として手直しされている．

X3T1.1 として完成したリンク暗号化規格は案としてまとめられ，X3T1 グループの無記名投票にかけられ，すべてのコメントは解決された．規格原案は勧告案として，X３の母体とSPARC (Standards Planning and Requirement Committee) へ送られた．

暗号化のモード規格も X3T1 で案出され，投票にかけられた．現在「編集に当たって」のコメント付けで懸案となっている．

トランスポート・レベル（ISO のレイヤー４）の暗号化規格の作業原案が X3T1.1 に述べられ，プレゼンテーション・レベル（ISO のレイヤー６）の作業原案も X3T1 でまとめられた．双方とも，初期の定義段階にある．

現在まで，ファイルの暗号化規格について，ANSI ではこれといった作業は行われていない．しかしながら，米国政府標準局では，ANSI の金融システム向けメッセージ認証規格X 9.9 に用いられている認証コードに基づいて，データの完全性に関する規格を準備しているといわれている．

■米国政府金融サービス標準（ANSI/X9)…

ANSI の金融サービスと銀行業の委員会（X9）で，個人識別番号（PIN）の暗号化の管理（これには米国のデータ暗号化規格 X3.92 が使える）規格が ANSI の標準化評議会で承認された．間もなく ANS X9.8として米国銀行協会から公表される．この規格は，ISO に，国際規格化の案件として手渡されている．

X9E8委員会は，電子資金転送の大口銀行向けメッセージ安全保護規格 X9.9を無記名投票にかけ，一般に公開した．すべての否定的コメントのうち，解決できるものは解決され，規格として一般に公表する前に ANSI の標準化評議会で検討を加えている．これはまた ISO に案件として提出されており，TC（専門委員会）68/WG（作業グループ）2 によって国際規格となる見通しである．

X9A3委員会は，米国の小口銀行業界に対し，X9.9 に類似した特殊銀行向けのメッセージ安全保護規格を初めて作成している．

米国の金融業界として，選択すべきアルゴリズムとして DEA 規格（データ暗号アルゴリズム規格：X3.92）を承認するかどうかを問う無記名投票が，ANSI X9 の委員の間で進められている．

X9 は最近，米国の金融業界に対するキー管理の規格を案出させる目的で新しい作業グループ X9E9 を結成した．X9E9 は，公開キー・データ暗号化の方法との合体化を含めて，すべてのキー管理の方式を調べあげようとしている．

■国際暗号化システム標準…

TC 97/WG 1 が，ISO における情報処理の暗号化標準原案を提言する権限をもっている．

ここに到るまでに，ANSI X3.92 とよく似た，データの暗号列を作るアルゴリズムの一番目の規格について作業原案の提案書が作られ TC 97 へ回布されている．間もなく規格原案として提示される見通しである．

同様に ANSI の，オペレーション・モードについての規格原案に似た，オペレーション・モード規格の作業原案が作られており，次の会議にかけられる手はずとなっている．

ひとたび同期がとられ，初期化されると，物理的な接続が続いている間ずっと，転送され，受け取られるデータ・ビットを暗号列化する，物理的（レイヤー1）暗号化規格の最初の案が作られたが，原案の様式を作る作業はまだ始まっていない．

リンク（レイヤー2），トランスポート（レイヤー4），およびプレゼンテーション（レイヤー6）の暗号化システム規格についての作業原案を作成するための研究が，TC 97/WG 1 で計画されている．

ISO の TC 68/WG 2 は，ISO における金融業界に対して，メッセージの認証とキー管理の提言を行う権限をもっている．

TC 68/WG 1 では，ANSI 規格 X9.9 に似たテスト・キーを用いるメッセージの認証方式に対して規格原案を提示していた．TC 68/WG 2 では認証に当たって使われる許可手順を案出した．さらに二つのアルゴリズムが，金融におけるメッセージ認証のプロセスに使われるべく提示された．一つは ANSI X9.9 のデータ暗号化アルゴリズムに基づいており，もう一つはドイツ銀行協会によって提案されている．

■米国連邦政府の標準…

連邦標準 1027 のテレコミュニケーション安全保障規格は，政府の事務官レベルの承認は採りつけたが，執行局長レベルの承認はまだ得ていない．また，連邦標準 1026 のテレコミュニケーション暫定規格の方は，まだ政府機関の事務官レベル相互の調整の段階にとどまっている．

（システム統括一部 システム推進部 河上 一郎 訳）

☆

Pásca1 標準化の動向

山田　勲

プログラミング言語 Pascal は，Niklaus Wirth が1968年に開発し公表したもので，その後数回にわたり改訂が重ねられ，1973年にその最終版ともいえる言語仕様に至った[1]．これが標準 Pascal といわれるものである．その後，この言語を国際標準規格とするための作業が英国を中心に行われている．

一方，いくつかの大学やメーカでその処理系が作成され，実際に使用されている．これらの処理系は，前述の標準 Pascal の機能を包含しているが，独自の拡張もなされている．本稿は，現在使用されている代表的な Pascal（表1）を取り上げ，どのような拡張がなされているかを述べたものである．

名　称	システム名	作 成 者
Pascal/VS	IBM 370	IBM
TI Pascal	TI 990	Texas インスツルメンツ
UCSD Pascal	8/16 ビット・マイコン	UCSD およびソフテック
Pascal 6000	CDC 6000, Cyber 70, Cyber 170	ETH Zurich と Minnesota 大学
NOSC Pascal	UNIVAC シリーズ 1100	Naval Ocean System Center
UW-Pascal	UNIVAC シリーズ 1100	Wisconsin 大学

表 1　本稿で扱った Pascal

名前 (identifier)：これには，下線文字（_）を許す．ドル記号（$）を許しているシステムもある．

注釈 (comment)：いくつかの Pascal システムでは埋め込み注釈を認めている．UCSD Pascal は，“{…}” および “(*…*)” を注釈と認め，その結果，“(*…{…}…*)” は，一つの正しい注釈となる．Pascal/VS では “/*…*/” を埋め込み注釈として導入している．

数値表現 (Number)：多くの Pascal では数の底を10以外のもの通常2進数，8進数，16進数を認めている．たとえば16進数 F65A は，TI Pascal では #F65A で，Pascal/VS では 'F65A'X で表す．

宣言の順序：いくつかの Pascal では宣言の順序の規則が緩められている．これはコンパイラ指示文 COPY または INCLUD で挿入される別々の原文ファイルの独立を保証するためである．宣言の順序は，また **CONST** 宣言の前に **TYPE** 宣言を置けるようにも緩められる．これは，構造体の定数を宣言できるようにするためである．

名札 (label)：名札は，二つの方法で拡張される．一つは，標準の範囲 0…9999 以上に長くする方法であり，もう一つは，名札として数値以外の文字を認めるやり方である．

定数 (constant)：いくつかの Pascal では **CONST** 宣言に式を使うことができる．たとえば，

```
CONST  radius=39.75;
       pi=3.14159;
       area=pi*radius*radius;
```

文字列型 (string type)：いくつかの Pascal では明白に宣言できる文字列型を提供している．UCSD Pascal での例を示す．

```
VAR  title: STRING;  {とくに長さを指定しなければ80文字である.}
     name: STRING [20];  {name の最大長は20文字でである．Pascal/VSでは STRING(20) と書く.}
```

拡張された精度の型：これは，とくに16ビット・マシンにとって重要である．それは値の範囲を16ビット整数から広げたい場合である．TI Pascal では **LONG INT** 型を用意しており．これで20億を超える値がもてる．UCSD Pascal では，使用者が整数宣言するときに必要な桁数を指定する（たとえば，**VAR** x: INTEGER [8]）．TI Pascal も，また実数変数を宣言するとき有効桁を使用者が指定する（たとえば，**VAR** x: REAL (12)）．

外部変数 (external variable)：外部変数は，別にコンパイルされたモジュールを仮定してのものである．この機能はいろいろな方法で実現されている．Pascal/VS は，**VAR** 宣言と同じような文法 **DEF** 宣言で変数を外部化している．これらの外部変数は，別のモジュールから **REF** 宣言で参照される．TI Pascal には **COMMON** 宣言がある．コモン・ブロックの中の変数を参照するのに **ACCESS** 宣言を使う．

静変数 (static variable)：静変数とは，手続きや関数を呼ぶたびにブロック中に一時的に割り当てられるスペースではなく，プログラムの実行中，常に同

じ記憶域を占める変数である．ある Pascal においては **VAR** 宣言と同じような文法 **STATIC** により宣言される．別の Pascal では変数を大域的にすることによってのみ静的となる．TI Pascal では静変数は変数を **COMMON** 宣言の中に置くことによりなされる．

関数と手続きパラメタの引き渡し：Pascal/VS では，パラメタを読み取りのみの参照で渡す方法を提供している．この方法は，**VAR** 宣言と同じ文法でルーチンの仮パラメタ並び中に **CONST** 宣言を用いて行われる（たとえば，**PROCEDURE** *myproc* (**CONST** *s*: INTEGER, …)）．

パラメタの引き渡しに関する重要な拡張は，動的な大きさをもった配列を引き渡す機能である．この機能を実現するために論争のもとでもある一致配列形式 (conformant array schema) も含めていろいろな実現方法がある．たとえば TI Pascal では，次元の上限を仮パラメタ並び中に疑問符で書く（たとえば **FUNCTION** *max* (*Nector*: **ARRAY**[1, ?] OF INTEGER): INTEGER)．実行中に次元の実際の大きさを知るために組み込み関数 *UB* も用意されている．Pascal 6000 では，たとえば **TYPE** *data* =**ARRAY** [INTEGER] **OF** REAL, のように書き，実行時動的配列の上限および下限を知るために組込み関数 *LOW* と *HIGH* が用意されている．

他の言語で書かれたルーチンの参照：多くの Pascal では，他の言語で書かれたルーチンを呼ぶ機能を用意している．FORTRAN は，そのうちでもっとも多用される言語である．FORTRAN ルーチンは通常，手続きまたは関数の本体を指示語 FORTRAN で置き換えて宣言することにより参照される．

外部ルーチン：外部ルーチンとは別々の Pascal コンパイレーションから呼ばれる手続き，または関数である．NOSC Pascal では，呼ばれるルーチンは手続き本体の前に "**PROCEDURE** ENTRY *myproc*(…)" のように書く．このルーチンを外部から参照するには **PROCEDURE** *myproc* (…), EXTERNAL のように書く．

コンパイル単位：コンパイル単位 (UNIT) とは同時にコンパイルされたコードの機能単位のことである．UCSD Pascal では外部コンパイル単位を参照するのに **USES** 宣言を使う．プログラムが，"ある単位を使う (uses)" と宣言すれば，その単位の参照可能部分 (interface) 中の宣言を無条件に参照できる．外部コンパイル単位は，二つの部分 (**INTERFACE** 部分と **IMPLEMENT** 部分) よりなる．**INTERFACE** 部分は，その単位を使うプログラムで直接参照される宣言とルーチン頭書きをもつ．**IMPLEMENTATION** 部分は，その単位に個有な宣言およびルーチン本体を含み，この部分はその単位を使用するプログラムからは隠されている．単位の配置は次のようになっている：

```
UNIT unitname;
  INTERFACE
    {宣言およびルーチン頭書き}
  IMPLEMENTATION
    {宣言およびルーチン本体}
  END;
```

代替記号：ある Pascal では演算子を表すのに特殊記号を用いる．Pascal/VS では，**OR** の代りに "|", **NOT** の代りに "¬", **AND** の代りに "&" そして "〈〉" の代りに "¬=" を認めている．

構造体変数の等価テスト：UCSD Pascal では，演算子 "=" および "〈〉" は配列型変数およびレコード型変数の比較を行うまでに拡張されている．

論理式：Pascal/VS では整数についてビットごとの論理演算ができる．

文字列演算：**STRING** 型をもつ Pascal では，たいてい文字列の連結の手段を用意している．Pascal/VS では "||" を使う．別の Pascal では **CONCAT** のような関数を用意している．

型識別子の関数としての使用：Pascal/VS ではスカラ型識別子を式の中で変換のための関数として使用できる．たとえば，color が (red, green, blue) 型として定義されている場合 color(1) は green を返す．これは，すべてのスカラ型に対して **ORD** 関数の逆を用意していることになる．(標準 Pascal ではこのような逆作用を型 **CHAR** についてのみ許している)．

代入文：UW Pascal には変数が "≔" の両側に現れた場合，別の形の代入文がある．たとえば，a≔a+1 は a+≔1 のように書ける．これは演算子 +, −, *, /, **DIV, MOD, AND** および **OR** について許される．

CASE 文：ほとんどの Pascal では **CASE** 文に関して **OTHERWISE** 句を用意している．たとえば，

```
CASE year OF
  jan, sep: X≔5;
  mar, jan: X≔6;
  OTHERWISE X≔0 END;
```

FOR 文：TI Pascal は **FOR** 文を **IN** という演算子を用いて拡張した．たとえば，集合 *setostuff*

に対し **FOR** *j* **IN** *setofstuff* **DO**…は，**FOR** *j*:= *firststuff* **TO** *laststuff* **DO IF** *j* **IN** *setosftuff* **THEN**… に等しい．

LOOP 制御文: いくつかの Pascal ではループからの脱出および **FOR**, **WHILE** および **REPEAT** ループの拡張がなされている．Pascal/VS では **LEAVE** 文でもっとも内側のループを抜け出す．UW Pascal では，**EXIT** ラベルでラベルの付けられたループを脱出する（たとえば 340: **FOR** *j*:=1 **TO** *j* **DO IF** *a*<*j* **THEN EXIT** 340 **ELSE**…）．

手続きおよび関数を脱出する文: いくつかの Pascal では，手続きまたは関数からの脱出するための文を用意している．Pascal/VS では **RETURN** 文でルーチンから抜け出す．UW Pascal は関数から値を返すため，語 **RETURN** に続けて式が書ける．UCSD Pascal では，**EXIT** (*g*) があり，*g* は脱出すべきルーチンの名前である．

参考文献

[1] K. Jensen, N. Wirth, *PASCAL USER MANUAL AND REPORT* (2nd Ed.), Springer-Verlag.

（プロダクトサポート統括二部）

MRPS について

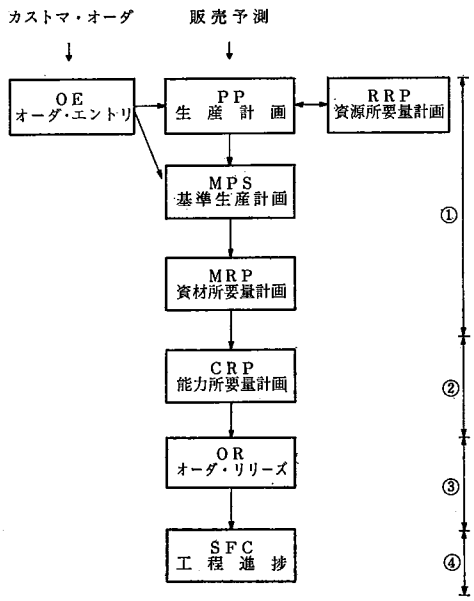
金子　功

1. はじめに

近年，生産管理に携わる人の間に MRPS (Material Requirements Planning System) の考え方が急速に広まりつつあり，一部の企業の導入報告がなされている．

2. MRPS の論理

MRPS（「資材所要量計画」と訳されることが多い）とは生産・在庫管理の一方式であるが，これまでの手法と違って計画主導型であるところが特徴である．利益計画や販売計画をもとに独立需要品目（需要が他の品目とは無関係に発生する品目のことで，製品やサービス・パーツが代表）の生産計画さえあれば，独立需要品目の生産に必要な従属需要品目（需要が他の品目の需要によって誘発される品目のことで，中間品・部品・原材料が代表）の生産・在庫計画は一意に決定できるという考え方をとっている．また，計算を独立需要から従属需要へと多段階に行うとともに，計画対象としての時間軸をタイム・バケットという小間隔（通常1週間程度）に区切り，このタイム・バケット単位に所要量を求めている．こうして得られた所要量計画をもとに，実際の運用に当たっては，近い将来のバケットに入った需要を製造や購買の指示として現場に発令し，それ以降のバケットの所要量は今後の予定として将来の準備活動に役立てる．さらに，バケットの最終日には，当バケット内で発生した全ての活動実績（入出庫報告や納入日変更報告）と営業部門からきた独立需要の変更を取り込んで，再び次のバケット以降の MRP 計算を行うことになる．この MRP の計算プロセスを示すと次のとおりである（図1）．



①プライオリティ計画　③キャパシティ管理
②キャパシティ計画　④プライオリティ管理

図1 MRPS のフロー

① 生産計画（独立需要品目の所要量計画）から従属需要としてどの品目が，いつ，どれだけ必要となるかをタイム・バケットごとに計算する．（プライオリティ計画）

② バケットごとの所要量をさまざまなリソース（生産設備の能力やマンパワー）と照らし合わせて，生産可能かどうか検証する．（キャパシティ計画）

③ 定められたバケット内に求められた品目を産出するようにリソースの投入具合を管理する．（キャパシティ管理）

④ バケット内の個々の作業の着手順序を管理する．（プライオリティ管理）

3. MRPS の前提条件

MRPS を成立させるには品目中心主義ともいうべき最大の前提がある．原材料から製品に至る物の

流れを管理の対象として特定のプロセス点として分断し，この点ごとに異なる品目としてとらえようとする考えである．この**品目単位**に所要量を計算し，オーダーとして手配し，実績も収集する．製番管理のようにある特定受注に対して原材料，中間部品から最終製品に至るまで一貫した超オーダーとしてひも付けするものでない．MRPS では，ある品目が在庫としてとらえる時点で別の上位品目に引き当てられ上位品目のオーダーが活性化する．こうしたMRPS を成立させる条件に以下のものがある．

① 部品表が確立していること
② 品目ごとのリード・タイム，在庫，発注状況が正しく把握されていること
③ 独立需要が与えられていること
④ 実際の製造・購買活動が MRPS によって出力・維持されている計画を中心に遂行するという制度

4. MRPS のねらい

MRPS の論理は何ら目新しいものでなく，一見利害が矛盾するマーケティグ部門と製造部門の両者に簡潔で，納得のいく統一的な計画中心の調整機能を果たすところにある．その意味で製造業の計画管理を担当する部門の強力なツールである．MRPS がなぜ今日注目を浴びるかは，MRP 処理そのものが膨大な量のデータ処理を前提にしており，今日のコンピュータのコスト・パフォーマンスでほぼ実行できるに至ったことが一つの理由である．また企業環境からみると，過剰生産や納期遅れが許される時代ではなく，製品の多様化・納期の短縮化・低コスト化が厳しく要求される今日，フレキシブルで合理的な管理システムとしての MRPS に期待が寄せられているといってよい．

また，日本の土壌で独自の展開をしているトヨタ生産方式（かんばんシステム）が，物の流れに添った管理を中心にしているが，かんばんの中・長期的な計画系をカバーするバック・グラウンド・プランニング・システムとして MRPS との隔合を指向しようとする企業が現れている．同様にして，FMS (Flexible Manufacturing System) は NC 機械やロボットに代表されるハードウェア面が脚光を浴びているが，FMS を支えるソフトウェアのうち，上位の計画系を支えるものが MRPS であるといってよい．

5. MRPS の動向

MRPS は製造業の中でも電機・機械といったアセンブリの業種から導入されているが，今日では食品や化学のプロセス型の業種へも浸透しはじめた．また，物流分野への応用も期待されている．工場から出荷された製品が需要家に届くまでには物流倉庫，地域配送センター，営業所といった流通経路が存在するが，これらの経路上の出庫（需要）計画・入庫計画・在庫計画を統合化し，保有能力・輸送能力・流通加工能力を合理的に統制することにより効率的な物流を実現しようとするものである．一般消費材を中心にこのアプローチが期待されており，DRP (Distribution Requirements Planning) と呼ばれている．

さらに，MRPS が生産物の数量と製造能力を主たる対象としているのに対し，財務管理・経営計画との結合をめざし，部門別・製品別の損益計画，投資・回収計画などを含んだ BRP (Business Resource Planning) への発展も検討されはじめた．

また，日本では，同一企業内でも製品系列によって管理範囲を分離させて独立した事業部制を採用したり，組立てや部品加工といった生産機能を垂直的に分離させる傾向が大企業を中心に広がっている．こうしたところでは一つの MRPS ですべての事業部や工場をカバーするには無理があり，それぞれの製品や機能の範囲を別個にとらえる分散型のMRPS を指向しているところもある．

6. MRPS のためのソフトウェア

MRPS は前述のように処理するデータ量からコンピュータ処理が必須であり，このためのソフトウェアは大規模なものとなる．しかし，部品表の維持・在庫把握・MRP 計算のロジックは明確であり，企業ごとの特別仕様は少ないため，汎用ソフトウェアとしてコンピュータ・メーカが提供している．ソフトウェアの自社開発とその後のメンテナンス工数を

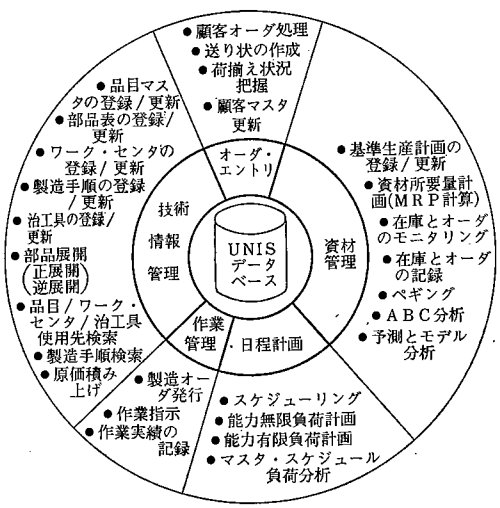


図 2 UNIS のシステム構成

考えると，汎用ソフトウェアの利用に踏み切るのが多い．

7. おわりに

日本ユニバック(株)でも1979年にMRPSを実現するソフトウェアとして生産管理パッケージUNIS (Univac Industrial System) を発表し，導入中を含めると60社の企業で採用されている．UNISは部品表の維持・検索を中心とする技術情報管理，MRP計算を核に生産計画，在庫管理を行う資材管理，能力計画や工程管理を行う日程管理，作業管理，および受注処理分野のオーダ・エントリ・システムなどのサブシステムで構成されている（図2）．

（システム推進部）

☆

UNIVAC 1100/90 シリーズの新機能一覧

編　集　子

Sperry Univac社のUNIVACシリーズは1100, 1960年代の初めに発表され，アーキテクチェア上からみれば，次の五つの段階を経ている．すなわち，1107, 1108 (1106, 1100/10, 1100/20), 1100/40, 1100/80, 1100/60である．括弧の中のモデルは，各各元のモデルに対して，ハードウェアの性能を向上させたモデルである．さて，UNIVACシリーズ1100は，発表以来，いくつかの点でそのアーキテクチェアは改善され，時代の要請に応えてきた[1],[2]．UNIVACシリーズ1100は，モデル間の差が利用者のアプリケーション・プログラムのレベルからはみえないという点で，一つのファミリーである．このことは，UNIVACシリーズ1100のすべてに対して，同じオペレーティング・システムが使われることによっても明らかである．また，1108発表以降，非特権命令の変更は上位互換性を保ち，特権命令の変更はオペレーティング・システムで吸収されている．完全な利用者レベルでの上位互換性の基礎は，1100オペレーティング・システムによって与えられており，これによって1100/80や今回の1100/90のような上位機種が世に出たといえよう．

UNIVACシリーズ1100の主な構成要素は，CPU (Central Processor Unit, 中央処理装置)，IOP (Input/Output Prosessor, 入出力処理装置)，MSU (Main Storage Unit, 主記憶装置) である．このほか，通信処理機能，オンライン保守機能，システム監視機能などの機能を分散させ，機能分散型コンピュータ・システムとして構成される．

つぎに，利用者側からみた1100/90シリーズの特長をシステム全体の生産性・拡張性・信頼性を支える技術について述べる．その要約をまとめると，次ページの図のようになる．なお，詳細は参考文献[3]を参照されたい．

参考文献

[1] B. R. Borgerson, M. L. Hanson, P. A. Hartley, "The Evolution of the Sperry UNIVAC 1100 Series", *Communication of the ACM*, Vol. 21, No. 1, pp 25-43, Jan. 1978.

[2] B. R. Borgerson. M. D. Godfrey, P. E. Hagaty, T. R. Rykken, "The Architecture of the Sperry UNIVAC 1100 Series Systems", *Conf. Proc. of the 6th Annual Symposium on Computer Architecture*, IEEE, pp. 137-146, Apr. 1979.

[3] "UNIVAC 1100/90 シリーズ，システム概説書"，日本ユニバック(株)，資料コード481293001-0, 1982.

UNIVAC 1100/90 シリーズの統合的拡張アーキテクチャ（利用者の視点から）

生産性向上を目指して（Integrated Highly Productive System）

高処理能力実現のために（High Performance System）

命令語処理能力 ページ 13

●マシン・サイクル時間の短縮　HPP（High Performance Packaging，2万ゲート/BPA）と分散 COS（Control Storage）を採用し，マシン・サイクル時間 30 ナノ秒．

●パイプライン制御方式　5 段階のパイプライン制御．パイプライン制御乱れの削減．

方式技術 ページ 14

●中央処理装置の方式技術　ID 分割された大容量高速緩衝記憶，ミニバッファ（ライト・ファイル），割り込み処理機構のハードウェア化．

●豊富な命令語群　ベーシック・モード 227 種，エクステンデッド・モード 235 種，アクティビティ制御，スタック制御，UPI 制御，仮想計算機機能．

●同時並行処理（機能分散プロセッサ方式）

IOP（入出力処理装置，ブロック多重チャネルの状況テーブリング，デバイス・パス・セレクション，オートセンス・リトリーブ，ビジー C/U ハンドリング），MSU（主記憶装置，リード・アクセス時間 360 ナノ秒/ 2 語，8 語 4 ウェイ・インタリーブ/ユニット，ブロック・リード・ライト（ブロック・リード・アクセス時間 540 ナノ秒/ 8 語，ブロック・リード・サイクル時間 600 ナノ秒/ 8 語，ブロック・ライト・サイクル時間 660 ナノ秒/ 8 語））．

ソフトウェア技術 ページ 15

●システム制御プログラムの負荷軽減　ダイナミック・アロケータのステップ数の削減，コモン・ファイル機能によるファイル割り当て負荷軽減，TIP デュプレックス・ファイル・ロード・バランシングの精度向上による人出力待ち時間の短縮．

● UDS（Universal Data System）の主記憶ファイル方式．多数のコモン・データ・バンクをもつことができる．

入出力データ処理能力 ページ 16

● UPI（Universal Processor Interface）　任意の 1 台の CPU（または IOP）から見て 4 台までの IOP（または CPU）に対して同一レベルの通信手段を提供．1100/91 システムから 1100/94 システムまでの自由度の大きいシステム構成．

●入出力チャネル速度の向上　ブロック多重チャネル・モジュール（4 個の BMC，転送速度：各チャネル 5.5 メガバイト/秒，モジュール単位 18 メガバイト/秒）とワード・チャネル・モジュール（8 個のワード・チャネル，転送速度：各チャネル 0.83 メガ語/秒（ISI）+0.25 メガ字/秒（ESI），モジュール単位 4 メガ語/秒）を任意の組み合わせで合計 6 個まで入出力処理装置に格納できる．

使いやすさの向上（Highly Useful System）

利用者への配慮 ページ 17

●統一化・単純化されたシステム・インタフェース　言語処理系（UCS）・データベース処理系（UDS）などが統合され，包括的なコマンド言語体系をもつ IPF 1100 と連絡，容易に対話型処理が行える．DPS 1100 による端末の仮想化．

●利用者援助機能　IPF 1100 などの対話型の各種インタフェース，メニュー・モードとコマンド・モードによるデータ操作．

●日本語処理　英数字仮名システムからの拡張と共存，高水準言語による日本語処理，システム・サポート機能，標準漢字コード体系，グラフ出力と文書処理，漢字入力の容易性，日本語による計算機システムとの対話，漢字カラー・コンソール，アプリケーション・プログラムの日本語表示．

最終利用者機能 ページ 19

● ADVISE1100　UCS，UDS と連結，IPF1100 と共通のコマンド言語体系，利用者援助機能，機密保護・障害時の回復機能．

システムの設計・開発・保守のしやすさ P. 20

●システム設計　UCS，UDS，IPF 1100 により設計負荷を軽減．

●プログラミングとデバッグ　IPF 1100 で EXEC の大部分の機能の使用可能．EDIT 1100 の編集機能，DDP 1100 による分散処理を利用したファイル転送とジョブ転送，DPS 1100 による画面制御，IPF 1100 により 1 台の端末装置から複数のジョブを起動させ，前景ジョブと背景ジョブを任意にスイッチ，UCS（Universal Compiling System）は，COBOL，FORTRAN，PL/I，Pascal，RPG などのコンパイラを包括した処理系，システム開発時の言語選択の自由度が高い．あらゆる処理形態でプログラムの開発・保守可能．リンキング・システムはスタティック・リンキングとダイナミック・リンキング，PADS 1100 による簡単なデバッギング，69 ギガ語のアドレス空間（エクステンデッド・モード），

●テスト・保守　仮想計算機機能（現システムを稼働させながら，新しい OS とこれを利用するプログラムを実行・テストできる），コモン・バンクの変更・置換が簡単．

●開発・保守の支援　SDASS（GCOB，デバッグを支援する TRTR-C，IDSS，ADMS）．

システムの運用・管理の簡便さ ページ 22

●無人化・省力化　SMCP と SSP による自動化，総合運用管理システム COSTAR．

●運用・管理機能　コース・スケジューラ，OS 1100 下のすべてのファイルの管理を支援する FAS 1100，各種管理情報の統合整備．

●導入・移行　対話型処理によるシステム生成．ベーシック・モードとエクステンデッド・モードの混在を許すミックスド・モード．

拡張性向上を目指して（Integrated Highly Flexible System）

ソフトウェアの拡張（Expandability to All Aspects） P. 24

●ソフトウェア・パーティションに沿って，モジュール化・ブロック化を進め，それらの処理を統一化する．UCS，UDS，IPF 1100 のもとで自由かつ容易な拡張が行える．

アドレス空間（Extended Mode） ページ 24

● 69 ギガ語の仮想アドレス空間　データ・バンク 16777 キロ語×基底レジスタ 16 個与アドレス空間 268 メガ語，BDT 4 個×BDT 中のエントリ 65500×命令実行のバンク 262 キロ語与仮想アドレス空間 69 ギガ語．

システム構成 ページ 25

● 最大 16 メガ語の主記憶容量　MSU の容量は，装置当たり 2～4 メガ語，1 メガ語単位で拡張，最多 4 台まで接続可能．

●多重プロセッサ・システム　CPU 4 台までの多重プロセッサ・システムが構成可能．

● IOP の拡張性　IOP 当たり最多 6 チャネル・モジュールの搭載が可能．

UPI（Universal Processor Interface） ページ 26

● CPU，IOP，専用付加プロセッサが基本的に対等の関係で能率よい連結が行える．

システム間の接続 ページ 26

● Optical ICC　UNIVAC 1100/90 シリーズ間，UNIVAC 1100/90 シリーズと UNIVAC 1100/60 シリーズなどを，転送速度 45～83 キロ語/秒の光ファイバ・ケーブルによるチャネル間結合ができる．

● SAU（System Availability Unit）　SAU により周辺装置群を 2 システム間で共有できる．

分散処理への対応 ページ 26

●分散処理　DCP/TELCON システムによる通信処理と，分散型通信網のアーキテクチャ DCA（Distributed Communications Architecture）に基づき容易に分散処理が実現．

高信頼性実現を目指して（Integrated Highly Reliable System）ARMIS

可用性（Availability） ページ 28

●システム監視　システムが正常に稼働しているかを常に監視し，異常事態に適切な対応を図る．CPU，MSU，IOP の自動回復機能，SSP による回復，EXEC による監視と障害の回復，SMCP による監視．

●閾（しきい）値検査

●データ転送路の多重化

●システムの動的再編成機能

●自動システム切り替え

信頼性（Reliability） ページ 29

●障害予防　HPP の採用などハードウェア技術の向上，EXEC を高水準システム記述用言語 PLUS で書き直し，再構成して信頼性を高めている．

●障害検出　データ・スルー・チェッキング，二重比較検査，クロック・タイミング・チェック，環境・電源の異常検出．

●障害回復　フォールト・トレラント技術を駆使し，システム内で統一的に障害回復．CPU のリダンダント・パワー・サプライ，縦断記憶の自動縮退，ハードウェア再試行機能，PCU の自動切り替えスイッチ，SSP のバックアップ・コンソール，COS の誤り訂正機能，MSU によるインタフェース監視，MSU のリカバリ・タイマ，システム・パネル，MSU，SSP の 3 個所から IPL（初期プログラム・ロード）ができること，CPU，IOP，MSU の動的再構成機能，装置監視機能の装置ごと分散．

保守性（Maintainability） ページ 29

● ソフトウェアの保守支援　PRIMUS，COMUS．

●遠隔保守支援　SSP の遠隔保守インタフェース，ROLAF．

●システムの動的再編成機能

●保守支援機能　論理回路テスタの組み込み，入出力チャネル・テスタ，MSU のエクササイザ，限界テスト，ディスクからディスクへの複写機能．

●誤り報告機能　CPU，IOP，SSP の障害記録の拡張，バンク状態レジスタからの誤り報告．

●診断機能強化　通電後の機能確認テスト機能，CPU，IOP，MSU の単体テスト機能，IOP のマイクロプログラム診断．

完全性と安全保障（Integrity and Security） P. 30

●ソフトウェア・パーティション　ソフトウェアの保護（protection）だけでなく，他から独立しているので導入（Install），置換（Replacement）が容易になり，さらに各構成要素の連結（Linkage）を誤りなく行わせることによって完全性と安全保障が確保できる．

●領域保護　利用者プログラムを実行する場合，プログラムの制御は EXEC，システム・プロセッサ，コモン・バンクなどに移り，それぞれのバンクに対するセキュリティ環境も遷移する．遷移先プログラムにはロックを，遷移元プログラムにはキーをハードウェアで設定，キーとロックが合致するときに限りバンクおよび対応セキュリティ環境の遷移が許可される．ロックとキー機構で守られた領域はリングとドメインから構成され，一般に外側のリングに属するバンクから内側のリングのバンクを直接アクセスできない．セキュリティの取り扱い単位は，ロックを備えている領域（ドメイン）であり，ロックには，同じドメイン内のアクセスのみを許す，登録されているドメインからのアクセスを許す，内側のリングに属するドメインからのアクセスだけを許す，すべてにアクセスを許すの 4 種類がある．ドメインからドメインへのプログラム制御の遷移を制御するのがゲート・プロセッシングで，ドメインの安全性を確保している．また，プログラム環境の復元処理をハードウェアが行う．

● 統合回復システム（Integrated Recovery）データの高度の完全性を実現するための誤り回復法を統合化した統合回復システム．

●大容量記憶装置の仮想化を進め，共通な管理を可能にする．ファイル管理のソフトウェア FAS 1100．

●システム・パネル　アプリケーションと SSP との対応を指定，各アプリケーションの資源分割・実行操作を各 SSP に分離でき，誤操作などの発生を一層確実に防止．

● SAU のプログラム・コントロール　SAU が操作員の操作誤りを監視し，システムの運転が妨げられないように保護．

●障害検出機能の強化　誤り情報収集機能，誤り状態再現機能，クロック信号の誤り検出機能．

●環境異常検知センサ

この表は，UNIVAC 1100/90 シリーズの新機能および新製品を中心に構成しています．一つの新機能や新製品は表中の特定の項目に従属するわけではありませんが，表中では最も関連の深い項目に位置付けています．

Seymour Papert 著

"Mindstorms: Children, Computers and Powerful Ideas"

Basic Books Inc., 15.8×23.4, viii+230pp., 1980.

本書は S. Papert 教授の率いる Massachusetts 工科大学 (MIT) の人工知能研究所の Logo プロジェクトの12年間に及ぶ研究から得られた「子供の才能を伸ばすための教育論」とそれに基づき開発されたプログラミング言語 Logo を紹介するものである．そして，また，Papert は，コンピュータに溢れたコンピュータ文化の時代（パーソナル・コンピュータが身の回りに常に存在して，子供が容易かつ自然にコンピュータから知識を得られる時代）の到来によって惹き起こされる精神的危機状況（Mind Storm, 心の嵐）を予告し，それを克服する方法を提唱している．

この本の著者の Papert は，数学・心理学・神経生理学・人工知能などの多方面に通じた学際的学者であり，現在は MIT の数学科に籍を置いている．また，同教授の経歴は多彩であり，Cambridge 大学で数学，英国の国立物理学研究所でサイバネティクス，Geneva で児童心理学者 J. Piaget の下での5年間，子供の思考能力の発達についてのサイバネティカルなモデルの研究を行っている．そして，1964年からは MIT の人工知能研究所に移り，神経回路網モデル（パーセプトロン）の研究，人工知能を利用した子供のためのプログラミング言語 Logo の開発，人間の認識モデルなどを研究している．

また，Papert は MIT の人工知能研究所では，最近まで M. Minsky と共にそのリーダを務めていた．

さて，本書の主題である Logo のルーツは，Piaget の発生的認識論と人工知能研究にある．まず，Papert は①子供は偉大な学習者であること，②子供は大人とは異なる独自の論理をもち，自己の経験に対する考察によって学習すること，を Piaget から学んでいる．一方，Papert は人工知能研究を通じ，"人間の思考" について思考すること，つまり認知科学の重要性を強調している．このため，Logo では子供によき体験をさせるための学習環境の整備と，子供が自分の思考を考察することが強調される．

なお Logo は 1967 年以来，現在に至るまで，研究が続けられているが，その間にあって MIT の人工知能研究者たちが，Logo に与えた影響は多大である．たとえば，Papert の親友 Minsky を初め，D. Bobrow, C. Hewitt, G. Sussman, T. Winograd, R. Greenblatt らの人工知能研究で知られるそうそうたるメンバーが若かりし頃に Logo と係わっている．Logo は，まさしく MIT の人工知能コミュニティの文化の所産である．

本書の内容を目次を追って紹介しよう．

まず，序文「私の歯車」では，Papert を学習理論の研究に進ませる動機となった幼児期の体験が述べられる．Papert は幼児期に玩具の自動車の差動歯車の魅力に取りつかれ，夢中で遊んでいるうちに，その機構を修得してしまった．そして，この差動歯車が九九や2変数の1次方程式を理解するための強力なツールとして役立ったという．実は Papert の Logo プロジェクトでの研究の目的は，コンピュータの変幻自在性を利用し，子供自身の手で Papert における歯車に相当する自分用のモデルを作らせることにあったと述べている．

導入の「子供のためのコンピュータ」では，この本の基本的主張が要約されている．従来の CAI の発想から Copernicus 的転回を行い，コンピュータが子供をプログラムするのではなく，子供がコンピュータをプログラムするという視点に立つことが示される．そして，このことによって，子供が主体性を確立し，コンピュータという強力なツールを使えるという自信を得るだけでなく，コンピュータを通じて深遠な科学知識を手に入れることや，自分の思考のプロセスを分析することができることが述べられる．

このほか，本書の主題が，知的活動や文化を成長させていく方法論にあり，Papert がコンピュータに文化の種子の担体としての機能を果たさせようと考えていることが述べられる．

第1章では，コンピュータが人文科学と自然科学という二つのサブカルチュアに引き裂かれた現代文化の病状を癒すこと，つまりコンピュータが両者間の障壁を取り去り，再度結び付ける媒体として働き，コンピュータによって新しい文化が生みだされることが述べられる．そして，タイプライタの鍵盤の配列が現在では無意味なものとなっていることをあげ，BASIC の繁栄も同様な現象として説明され，このため Logo の普及には抵抗があるだろうという

ことが述べられる．

第2章では現在の数学嫌い（Mathophobia）が子供の知的・肉体的活動と学習とを切り離したモデルなしの学習によって生まれていること，および学校での数学教育の問題点を指摘する．そして，数学の理解には，数学の体験が必要であり，そのためには数学国（Math Land）をコンピュータ上で実現すればよいことが述べられ，Logo システムで数学嫌いの子供が数学を好きになった事例が紹介される．

第3章では数学国の指導原理としてのタートル幾何学が述べられる．タートル幾何学は一種の微分幾何学である．このほか，本章では Logo によってタートル（亀を形どった作画用ロボット）を動かし，図形を描くプログラムの作成を通じて，自己発見的に幾何学を修得する例が紹介される．タートルによって描かれる図形に関する知識は，子供が自分の体をタートルと同一視することによって深まる．

第4章では通常は知的活動とみなされていない肉体的技術，たとえば竹馬やボールの曲投げ（ジャッグリング）などの学習にも，プログラミングにおけるプロシージャの概念が使えることが示される．

第5章では，人工知能研究でよく用いられる小世界（Micro World）のアプローチが紹介される．ここでは，数学国に続き，Newton 力学の支配する小世界が述べられる．この小世界では，子供は直接 Newton 力学を体験できる．

第6章では直観によるアイデアの重要性，およびコンピュータがそのアイデアの具体化やモデル化に非常に有効であることが述べられる．そして，最後に手順（プロシージャ）を物（entity）として名前を与え，操作・デバック・変更することの重要性や，日常生活においてプロシージャを援用することによって形式的モデルの理解を深めることができることなどが述べられる．

第7章では Logo のルーツである Piaget の発生的認識論と人工知能研究の最近の成果によって，学習の理論的枠組みが明らかにされる．

第8章では将来の学校のイメージとして，Rio de Janeiro のサンバ・スクールが紹介される．サンバ・スクールはカーニバルで踊るサンバの練習を行う社交クラブであり，みんなが各自の技量に応じてサンバを楽しんでいる．Papert は，ここでコンピュータを中心にした，このような文化的コミュニティの建設の草の根運動を提唱している．また，映画の登場が新しいサブカルチュアおよび職業を生みだしたと同様に，パーソナル・コンピューティングの世界も新しい技術・感性・人生哲学をもった人々の出現が待たれていることが述べられる．

終章では，フランスの有名な数学者 H. Poincaré の数学的思考の分析結果としての仮説を紹介している．これは数学者に必要なものは論理よりも審美的能力であり，数学者は複数の代案を無意識の内に評価し，美的な解のみを抽出し意識下へもってくる能力を有している，というものである．そして，Papert はこの審美的な能力がきわめて個性的であることを指摘するとともに，自己同一化（ego-syntonic）あるいは肉体同一化（body-sgntonic）な数学および審美的な能力の重要性を述べている．

あと書きおよび謝辞では，1967年に始まる Logo プロジェクトの歴史が紹介され，その中で Logo システムに寄与した人々の名前があげられている．

パーソナル・コンピュータが子供の身の回りに存在し，鉛筆のように使える（computer as pencil）環境の実現に，あと何年かかるのかはわからないが，その到来は不可避であり，Papert 教授の主張に耳を傾けることは有意義なことであろう．

本書は，コンピュータの社会に及ぼす影響に関心をもつ人々にだけでなく，すべての人に一読をお奨めしたい．

なお，Logo は従来は PDP-10 用しかなかったが，最近は Apple Ⅱ や TI 99/4 などを初めとするパーソナル・コンピュータの版が登場しており，新しい言語として注目されている．

また，本書の邦訳が，未来社から本年の9月に「マインド・ストームズ――子供，コンピュータ，そして強力な概念」（奥村貴世子訳）として発刊される予定である．

高橋秀俊，森口繁一　他 編

岩波講座　情報科学　全24巻

岩波書店，A 5 判，平均 240 pp.. 1981 年刊行開始

20世紀後半，とりわけ残り四半世紀を特徴づけるのは，いうまでもなく飛躍的な発展をとげつづける情報科学，情報工学であろう．エレクトロニクス技術の急速な進歩とあいまって驚異的発展をとげたこの分野の理論や技術は，他の科学・工学分野に影響を与えただけでなく，社会環境や経済活動にまで大きな変化をもたらしつつある．コンピュータにたずさわる人間も増加しつづけ，その一方で慢性的人手不足をつづけている．これからの時代は，だれもが

情報処理に関する何らかの知識をもつことを要請している．そして，これら新しい時代の新しい学問領域となった情報科学は，これまでに集積された成果を整理し，今日の重要な技術的課題を明らかにし，将来への確かな展望をもたねばならない時期にさしかかった．

本講座は，こうした状況下での理想的カリキュラムの編成という，一つの試みを行ったものである．全 24 巻は，入門，表現（情報の表現法や構造を説く），処理（人間の理知を蓄積・制御して活用する方法を述べる），機構（コンピュータがどのように構成されているかを説明する），数理（抽象的な情報を扱うための基礎としての数学，数理的手法を記述する），特論（これからのコンピュータで特に重要なテーマについて解説する）の六つのジャンルからなっている．このうち，これまでに入門が 1 冊，表現が 3 冊，処理および機構が各 1 冊，数理が 3 冊，特論が 2 冊の計 11 冊が発行されており，その内容は順に以下のとおりである．

3 巻「プログラムの読み方」プログラムを通じてコンピュータを理解するため，BASIC, PASCAL, FORTRAN, COBOL, PL/I で書かれたプログラムを読んで基本的アルゴリズムや制御構造，データ構造が学べるよう記述

4 巻「情報と符号の理論」符号の立場から Shannon の情報量，エントロピー，通信路について述べ，さらに符号の理論を解説

5 巻「情報ネットワーク理論」情報伝送を行う通信システムから情報ネットワークに至る各システムの理論的な階層関係を明らかにし，ついで，情報の伝送理論，通信トラヒックの理論，通信ネットワークの理論，コンピュータ・ネットワークの理論を解説

6 巻「オートマトン・形式言語理論と計算論」書名の通り，情報の科学・工学の基礎理論を計算の複雑さの話題までを説明

7 巻「算法表現理論」再帰的関数，λ アルゴリズムなどの作用的表現の基礎から出発し，手続きおよびデータの関数的表現，命令的表現，それらを抽象化する表現形式を説明し，非決定性，並列性のアルゴリズムを論述

14巻「計算機の機能と構造」コンピュータ，とくにマイクロコンピュータをハードウェアとソフトウェアを統一的にまとめて解説，さらに仮想コンピュータについても言及

17巻「離散数学」情報理論の基礎数学として，有限体，有限環，グラフ，マトロイド，組合せ論を講述

18巻「数値計算」コンピュータ内部での数値の取り扱いとその特徴，関数計算，連立 1 次方程式と固有値問題，数値微積分，常微分方程式，Fourier 変換を解説

20巻「信号処理とシステム制御」信号処理理論を工学の立場から現代的に導入し，それを現代制御理論に結びつけるよう，信号と Fourier 変換，不規則信号とスペクトル解析，線形システムのモデル化，スペクトル推定，システム同定，制御動作の決定法を説明

22巻「人工知能」問題解決，問題の表現，探索の方法，人工知能用言語，コンピュータによる知識表現とその利用法を説明

23巻「数と式と文の処理」初等整数論を中心に数学とコンピュータの関係と現状，データベース・システムを設計するための組合せ論，化学物質の構造式の同定法，代数学を基礎にした因数分解，不定積分の扱い方，自然言語の取り扱い，コンピュータによる設計などの話題を紹介

これらは，基礎的な事項をていねいに解説したわかりやすい入門書もあれば，通読するのが容易でないものもある．“講座”として全巻予約制のため，単独に購入できないのが残念である．

●第23回プログラミング・シンポジウム開催——箱根，1982年1月12日～14日

招待講演は，淵一博氏（電総研）によって「第5世代コンピュータへのアプローチ」と題して行われた．また，一般講演の主なテーマは，Prologによるプログラミング，SNOBOL 3を利用した文字列の高速処理手法，Pascalの実行時支援システム，FIFOキューを同期手段とする待ちなし並列プログラム，画像解釈言語PLIS，アドレスの誤動作に起因する虫の検出法，事務処理ソフトウェアの保守支援システムMSF，英文と和文のタッチタイプ練習システム（ソフトウェア作成法のケース・スタディ），TSS端末の使用状況と学業成績との相関，計算による美の扱い，などである．

●第12回画像工学コンファレンス開催——東京，1982年12月10日～11日

招待講演は，沓沢淳之助氏（NHK技研）の「アニメーションとコンピュータ」，長谷川敬氏（NHK基礎研）の「画像と色情報の心理的性質との対比」，坂根厳夫氏（朝日新聞）の「錯視と芸術」，河村孝夫氏（大阪府大）の「アモーファス・シリコン薄膜の電子写真への応用」，坂井利之氏（京大）の「オフィス・オートメーションにおける文字・図形の体系的扱い」，武内敏氏（大日本印刷中研）の「印刷における新しい情報メディア」などがあった．

●COMPCON '82 Spring 開催——San Francisco，1982年2月22日～25日

会議の前日に「VLSIの設計および利用」（R. Rice，コンサルタント），「1980年代におけるVLSI技術の動向」（D. J. McGreivy，Gnostic Concepts），「コンテクスト指示型パターン認識と人工知能技術」（Y. Pao，Case Western Reserve Univ.）の三つのチュートリアル・セミナーが開かれた．Pao教授のセミナーでは，属性文法を利用した構造的パターン認識法，人工知能における知識準拠システム（Knowledge Based System）などのエキスパート・システム技法，パターン指示推論システム（Pattern Directed Inference System）などが取り上げられた．

また，本年の基調講演は，L. W. Branscomb氏によって「コンピュータを人々の手へ——普及への挑戦」と題して行われた．このほか，ネットワーク・ファイル・サーバー（Network File Server），Ethernetの評価，ソフトウェア工学の現状，80年代のVLSI設計技術，創造的企業家の特性などをテーマとしてパネル討論が開かれた．なお，F. Brooks氏（Univ. of North Carolina）を議長とした特別パネル・セッションも行われた．

また，今回の一般講演の主なテーマは，VLSI分野では，MIPSマシン，高水準プログラム・ユニットのVLSIへの組み込み，プログラマブル・シストリック・アレー，バス結合多重プロセッサのためのキャッシュ技術，シリコン製造システムのアーキテクチャ，VLSI鋳造サービス，環状型データフロー多重プロセッサ，LISPマシンのCADへの応用，シリコン・コンパイラなど．

ネットワーク分野では，分散型データベース・コンピュータDDBC，ローカル・ネットワーク用データベース・サーバーのアーキテクチャ，EXPRESS-NET，OMNI-NET，統合音声・データPABXの応用，データ・ネットワークと音声郵便との統合，実用会話型CATVと家庭情報システム（INDAX），などである．

ソフトウェアおよびアプリケーションでは，暗号化キーのサイズの問題，ローカル・ネットワークの暗号化システムのアーキテクチャ，CP/MとUNIXの比較，米国海軍の電子ドキュメント・ネットワーク，高品質コンピュータ出力，マイクロプロセッサ用組版システム，パーソナル・コンピュータの教育への影響，音声認識装置の性能評価の標準化，隠曲面消法，ソフトウェアの保護（著作権・特許・所有権）などであった．

●情報処理学会第24回全国大会（昭和57年前期）——東京，1982年3月22日～24日

特別講演および招待講演は，唐津一氏（松下通信工業）による「世界における日本技術の役割」と，渡辺誠氏（電電公社）による「LSI/CADの現状と課題」である．このほか，「ソフトウェアの使いやすさ」，「知識工学とその応用」と題する二つのパネル討論がなされた．また，一般講演では616件の論文が発表された．なお，本年の一般講演は人工知能・日本語処理・ソフトウェア工学などついて行われた．

●電子通信学会総合全国大会——東京，1982年3月26日～28日

特別講演は，「パリ大学日本館の思い出—日仏学術交流について」（中央大学　戸張智雄氏），および「高度情報化社会へ向けての課題」（電電公社　北原安定氏）であった．このほか奥村善久氏（金沢工大）を座長として，パネル討論「移動通信の発展にアンテナ伝搬研究者は何をなすべきか」が開かれた．ま

た，特定のテーマをめぐる一連の報告がシンポジウム講演と銘打って行われ，GaAs IC，テキスト処理，データ・音声パケット複合通信，音声認識，薄膜磁気ヘッド，移動体とディジタル・データ通信，画像におけるカラー処理・多階調処理，並列処理システム，教育情報のデータベースなどが取りあげられた．

なお，一般講演の主なテーマは，非決定性プログラムの全面的正当性，代数的仕様記述，最大カット・セットを用いたプログラム・テスト網羅性，データベースにおける外延・内包情報の取り扱い，言語理解における語順，日本語処理と Prolog，フレーム知識表現システムの支援ソフトウェア，電子ノート・システム，三面図認識のためのアルゴリズム，手書教育漢字認識，日本語音声認識研究用音声データベース，印刷・出版システム，1チップ単語音声認識 LSI，記号処理向きデータフロー・マシンのアーキテクチャなど．

●CAD 82 (5 th International Conference on Exhibition on Computers in Design Engineering) **——Metropole, 1982年 3 月31日～4 月 1 日**

招待講演は，英国の F. R. A. Hopgood 氏(SERC Rutherford and Appleton Laboratory) による「グラフィック標準化への道」，および K. G. Nichols 氏 (Univ. of Southampton) による「大規模非線形回路のシミュレーション」であった．Hopgood 氏は最近活発化しているコンピュータ・グラフィックスの標準化を取り上げ，GKS (Graphics Kernel System) を標準案として ISO に提出するまでの経緯を紹介した．また，今回の一般講演のテーマでは，Tektronix 4054 などのマイクロコンピュータの CAD/CAM への適用が数多く発表された．

なお，Sperry Univac 社からは J. Z. Gingerich 氏が「機械工業におけるハイブリッド CAD/CAM システム (A Hybrid CAD/CAM System for Mechanical Applications)」と題し，1980 年代の CAD/CAM の課題を解決するアプローチとして同システムを紹介した．

●ACM-SIGOA Conference on Office Information Systems——Philadelphia, 1982年 6 月21日～22日

この会議では，ウォートン・スクールの H. L. Morgan 教授をホストとして，オフィス・オートメーションの理論的課題が討論された．各セッションのテーマは，オフィス・システムのモデル，ユーザ・インタフェース，人工知能，ユーザ・ニーズ，音声とテキスト処理，オフィス・フォーム，コミュニケーションなどであった．このほか，オフィスの評価，オフィス・データのビュー，オフィスに必要な通信容量などと題し，パネル討論が開かれた．

●NCC '82 (National Computer Conference) ——Huston, 1982年 6 月 7 日～10日

ハードウェアおよびコンピュータ・アーキテクチャ，ソフトウェア工学，パーソナル・コンピューティング，社会および組織体関連，オフィス・システム，マネジメントの問題および意思決定支援システム，言語およびデータベース処理，計算応用の八つの分野に分かれて講演が行われ，約 6 万人が参加した．また，パイオニア・デイの催しでは，FORTRAN が取り上げられ，その生みの親である J. Backus 氏らが FORTRAN の誕生前後の状況などを紹介した．

さて，各分野におけるパネル討論および一般講演の主なテーマは次のとおり．

ハードウェアおよびコンピュータ・アーキテクチャ分野では，ファームウェアの品質管理，CMOS を使ったマイクロプロセッサ，16ビット・マイクロプロセッサによる分散処理など．

また，ソフトウェア工学分野では，1990年代のソフトウェア工学，コンピュータ援用ドキュメンテーション，ソフトウェア開発環境，アプリケーション・ジェネレータ，ソフトウェア品質保証，ソフトウェアのライフ・サイクルなど．

パーソナル・コンピューティングでは，コンピュータ娯楽ソフトウェア業における諸問題，オフィス・オートメーション用高水準言語 QBE/OBE など．

オフィス・システム分野では，オフィスにおける画像処理とデータ処理の統合，マネジメントと専門職用のワーク・ステーション，暗号準拠高セキュリティ・オフィス・システム，電子郵便など．

言語およびデータベース処理分野では，人工知能技術の社会に及ぼす影響，マイクロコンピュータにおけるデータベース，Ada の特徴および効用など．

☆

MEMORANDUM

〈UNIVAC 関係の論文・講演等の要約〉

●UNIVAC シリーズ V 77-800 を利用した回路の CAD (Computer Aided Design)——システム構成は，Ramtek 6214 カラー・グラフィックス・システム，Hewlett-Packard 4 色プロッタ，Summagraphics ディジタイザ，TI 810 プリンタ，V 77-800 ミニコンピュータであり，入出力機器はデータ通信リンク経由で V 77-800 に接続されている．

本 CAD システムの主な機能は，ディジタイザからの IC レイアウトの入力，レイアウトの編集，多角形 (polygon) のディジタイズ，回路の寄生電流 (パラスティクス，parastics) の計算などである．(P. Groner et al., "Computer Aided Design of VLSI Saves Man Hours, Reduces Errors", Control Engineering, April 1981)

●IEEE のローカル・エリア・ネットワークの標準化動向——ローカルネットワーク普及の鍵は，リンク・レベル・コントロールを行うインテリジェント LSI および，コミュニケーション媒体変換装置 (Communication Media Transducer Device) の大量生産による低価格化であり，その前提となるのがインタフェースとプロトコルの標準化である．IEEE では 1979 年の 8 月にローカル・エリア・ネットワーク標準化プログラムを発足させ，以来標準案の作成を行っている．現在，この標準案の作成を行っているのが，IEEE 802 ローカル・エリア・ネットワーク標準化委員会であり，次の 3 分科会によって構成される．まず，媒体分科会は物理的媒体とその媒体アクセス・ユニット (MAV) インタフェースを対象としており，基本帯域・広帯域・光通信の三つの媒体についての標準案の作成および，エンド・ユーザ・デバイス (DTE, Data Terminating Equipment) と媒体アクセス・ユニット間の共通インタフェースの標準化を行う．第 2 のロジカル・リンク・コントロール分科会は，リンク・レベル・プロトコルを対象とし，上位層の媒体独立なデータ・リンク (共通データ・リンク) プロトコルと下位層の二つの媒体依存プロトコルの標準案を作成する．なお，後者は衝突検出付きキャリア・センス多重アクセス (CSMA/CD) とトークン・パスの 2 種の媒体アクセス・プロトコルに分けられる．第 3 の高水準プロトコル分科会は，標準案は作成しないが，複数ローカル・ネットワーク間の接続などの高水準機能の調査やローカル・ネットワークの参照モデルの開発などを行っている．

このほか，ローカル・ネットワークに関しては，米国規格協会の入出力の標準化分科会 (X3T9) と通信分科会 (X3S3)，米国標準局 (NBS)，国際電気標準会議 (IEC) の Proway 委員会 (IEC65C/WG 6) で標準化が検討されている．(J. Nelson, et al., "Development of the IEEE 802 Local Area Network Standards", NEC '81)

●音声合成技術の現状——音声合成技術は，音声信号の表現方法の相違によってアナログ方式とディジタル方式に分けられる．アナログ方式の音声合成は通常の (アナログ) 磁気テープ装置と増幅器，スピーカで行われる．一方，ディジタル方式は，波形表現 (waveform representation) とパラメタ表現 (parametric representation) に大別される．波形表現はアナログ波形を一定の時間間隔でサンプリングし離散化したもので，再生は復号装置と DA 変換装置などによって行われる．パラメタ表現方式は，音声符号化 (vocoding : Voice encoding) 方式とも呼ばれ，離散化された波形から特性パラメタを推定し，励振パラメタと声道応答パラメタを求め，これらによって励振モデルと声道モデルから構成される音声合成モデルを駆動して音声を再生するものである．

これらはいずれも，実際の音声信号から抽出したパラメタを用いるものであった．しかし，最近では第 3 の表現として入力文字列から，音声信号パラメタをプログラムによって生成する「規則による合成 (synthesis by rule)」あるいは「テキスト-音声合成 (text-to-speech synthesis)」ともいうべき方式が急速に発展してきた．この方式は実際の音声信号の代わりに，言語学や音声発生機構に関する生理学の知識を用いるもので，そのプログラムはテキストから音素列 (phoneme string)，音素列から音声モデルの制御パラメタといった二つの変換ステップによって構成される．そして，テキスト変換規則・発音辞書・音声変換規則・音素移行規則などがその処理の過程で用いられる．

波形表現方式は最も単純なアルゴリズムによって最上の音声品質が得られる．しかし，語彙の追加などに関し柔軟性が乏しく，しかも必要記憶容量が大きい．一方，テキスト－音声合成方式は最も柔軟性に富むが，音声品質は最も劣り，しかもアルゴリズムは複雑である．そして，パラメタ表現方式は，波形表現とテキスト－音声表現の両方式の中間的性質

をもつものといえる．現在製造されている音声合成装置は，波形表現（線形予測符号化以外），線形予測符号化，フォルマント（formant，パラメタ方式の1種）などの方式があり，なかでも線形予測符号化方式がよく用いられている．(T. L. Bates, "Speech Synthesis", Utah Chapter Meeting of the ACM, 1980)

●科学技術計算用付加プロセッサのアーキテクチャの考察——科学技術計算用コンピュータの性能の分析には，演算命令1個当たりの記憶参照回数と演算装置の性能との関係の解明が必要となる．このため，高速の科学技術計算システムの実現には，ホストプロセッサ，付加プロセッサ，入出力プロセッサの三つからアクセス可能な高速記憶装置を用いることが必要である．また，付加プロセッサのハードウェア機能としては，高速ベクトル計算，ホスト・プロセッサより速いスカラー計算，4倍精度浮動小数点計算などが求められる．このほか，ソフトウェアの面では FORTRAN のベクトル演算処理機能の拡張，効率の良いプロセッサ間コミュニケーションが可能にするキュー駆動型スケジューリング（queue-driven scheduling）の採用などが必要となろう．(L. E. Sheets, "More Performance through Attached Processors", Presented to Edison Electric Institute)

●MAPPER を利用した CAD/CAM システム TACTICS (Total Assurance and Control by Testing and Interactive Collection System)——TACTICS は，16の独立したコンピュータ・システムで構成される中央集中型の分散処理ネットワークで，その中核は MRCC (Midwest Regional Computer Center) の UNIVAC シリーズ1100/82であり，Rosevill周辺ばかりでなく，電話回線や通信衛星によって北米各地の製造工場から利用されている．TACTICS には，約1,700台のユニスコープとそれ以外の端末1,000台，試験・検査装置140台以上が接続されている．MRCC での中心的なソフトウェアは，MAPPER で，製造・検査データベースの蓄積・検索を行えるだけでなく，製造装置・検査装置や検査中のコンピュータとのインタフェスをとりデータを得る．

なお，TACTICS のサブシステムとしては，回路設計の CAD システム UCADS，プリント基板検査システム QUIET，コンピュータ検査システム MIDAS，資源管理システム，（静電）環境監視システムなどがある．(L. F. Rogney, "TACTICS—Automated Manufacturing with MAPPER", Presented to the American Society for Quality Control)

●ディスクの動的スペース管理——動的ディスク・スペース管理はファイルをディスク上に最適配置し，シーク・タイムの短縮，ディスク装置間の競合と記憶の細分化などを回避する一連の手法としてとらえられる．同氏らは OS/3 を対象としてシミュレーション・モデルを作成し，①大域的再編成 (global reorganization)，②最多頻度参照パーティションの再配置 (reference spike relocation) などを評価し，それらがシステムのスループットおよびレスポンス・タイムに与える影響を予測する．その結果，中央コンピュータで 20～30パーセント のスループットが向上することが明らかになった．(J. O. Dyal, et al.. "Performance Aspects of Disk Space Management", IEEE 1981 Winter Simulation Conference)

●データベースの併行性制御のためのアルゴリズム——レコード（組，tupple）・レベルのロック法には，①物理的ロック・システム（組・ページ・エリア・ファイルなどの物理的単位にマークを付けたり，ロックのためのリストに名前を登録する方法），②論理的ロック・システム（個々のトランザクションごとに述語ロック（predicate lock）を付与しロックする方法など）がある．しかし，両システムとも必要以上にロックしすぎるため並行性のよいロック・システムが求められていた．本報告の手法は，精密ロック（precision lock）と呼ばれ，データベースの整合性を保証するために必要な最小限の組の集合を正確にロックするものである．精密ロックは，述語ロックの実現法の1種であり，起こりうる衝突（potential conflict）と実際の衝突（real conflict）を区別し，後者のみを防止することができ，さらに幽霊組問題（phantom tupple problem，トランザクションが，存在しない組を要求した後に，もう一つのトランザクションによって，その組が生成されること）を効率良く解決できる．本報告ではロッキング・システムの比較の基本的考え方，現在システムにおける誤り，許容される述語およびトランザクションの条件，実施上のコストなどを紹介している．(J. R. Jordan et al., "Precision Locks", 1981 ACM-SIGMOD International Conference on Management of Data)

●データ管理システムの動向——今後重要となるデータベース技術の一つに，即時，情報にアクセスできる使いやすい問い合わせプロセッサ（query processor）がある．なかでも，関係モデルによる見方（relational view）が可能な問い合わせプロセッサ

の開発が求められている．最近の関係演算の操作系列に関する最適化アルゴリズムの研究からは，明るい見通しが得られており，将来パフォーマンスのよいプロダクトが開発される可能性が高い．また，企業で使用されるデータベースのタイプは1種類に限定されないため，(ネットワーク・モデルに対し関係モデルによる見方をするケースのように)，あるデータモデルの見方によってアクセスする場合には，モデル間でのデータの写像 (mapping) が必要となる．この面での研究の最大の課題は，関係的モデルにより，効率よくネットワーク・モデルをアクセス技術するの開発である．

データ辞書システムは，当初データベースを統制するものとして出現したが，最近では受動的辞書(データおよびプログラムの情報をユーザ自身が入力する) としても用いられるようになった．将来はこの二つの辞書機能が統合され，データとプログラムに関する情報が自動的に入力・保守される能動的辞書システムへと移行しよう．また，データベース・システムの機能が整理されるにつれて，データベース・プロセッサの実現性が増大している．関係モデルによるデータベースの場合，パフォーマンスが隘路となっているため，データベース・プロセッサの登場が期待される．

現時点でデータベース技術の将来を予測するのはむずかしいが，データとそれを処理するタスクをうまく結合するためには，ジョブ制御とオペレータの指示による現在の方式は限界があり，今後はデータフローの記述による制御の採用へと移行しよう．最近活発なデータフロー分析によるソフトウェア設計は，この傾向を示唆するものである．(M. D. Holt, "Trends in Data Management Systems", UUA/E in Monte Carlo)

●会話型システムへの人間工学的手法の採用——アプリケーション・システムを成功させるには，機能だけでなくユーザ・イスタフェースにおける人間工学的要因を考慮しなければならない．一般にシステム設計では，機能を果たすことにのみ注意を払い，使いやすさとか使用の楽しさなどといったことは軽く考えられやすい．ところが，現実のシステムでは，使いやすさを配慮しなかったために使用されなかったものも多い．システム設計では，技術的要素と審美的要素とのバランスが重要である．なお，使いやすさとか望ましさ (desirability) は後者に属する特性である．

人間工学は，使いやすさを解明する学問であり，オペレータの疲労や退屈を最小化する技術として第2次大戦中に生まれた．最近は人間工学のコンピュータ・システムへの適用が活発化しており，ユーザの不安や緊張と，パフォーマンスとの関係などが分析されている．そして，不安とパフォーマンスとの関係が逆U字型になることがわかっている．

さて，Sperry Univac 社のソフトウェア・リサーチ・グループでは，①コマンド数の制限とその最良省略型，②行番号の付与方式 (階層型も分数型)，③コマンド構文の区切り方法などの実験・研究，を行っている．このほか同グループでは，ユーザの誤り，疲労・退屈を軽減する技術を研究しており，使いやすさの要因を分析している．

また，人間工学の大きなテーマとして，人間とコンピュータが，どういう言語で話し合えばよいかという問題があるが，ジョブ制御言語として英語を用いるのは時期向早である．現状ではコンピュータに対する質問に英語を用いるのが選択として優れているであろう．

さて，コンピュータ・システムの設計者としては，最も有用なのは懐疑的態度であり，すべての設計の決定を疑い，その正当性を確めるべきである．システムの設計に当たっては，設計者がユーザになるのではなく，ユーザが設計者の上司であると思って設計すべきであろう．(R. L. Wexelblat, "Design of Systems for Interaction between Humans and Computers", 1981 Meeting of British Computer Society)

●プログラミング言語の動向——代表的プログラミング言語として，BASIC, FORTRAN, LISP, Pascal, Ada, 関数的プログラミング言語，自然言語の八つをとり上げ紹介する．まず，BASIC は，制御構造の不十分さ，変数の制約の存在，可読性・保守性の悪さなどの欠点がある．そして，Sperry Univac 社の R. Wexelblat の「プログラマはその最初に覚えたプログラミング言語のスタイルや構造に規制される」という仮説からいっても，BASIC の使用は奨められない．FORTRAN については，FORTRAN '77 に対する拡張が米国エネルギー庁の提案に基づき，ANSI で検討されている．その内容として，非同期入出力，ビット操作，NAMELIST，動的アレー，Pascal と同様なデータ型，マクロ処理，アレーの各要素に対する並行操作などが含まれている．

LISP は，計算機科学，とくに人工知能の研究用ツールとしてよく用いられ，Sperry Univac 社でも後述の自然言語の研究プロジェクトで使われている．

PLUS は，ALGOL 系言語 JOVIAL を改良したシステム・プログラム記述用言語で，UNIVAC シリーズ 1100 の EXEC や UCS の記述に用いられている．

Pascal は，プログラミングの教育用に開発された小規模で学習しやすい言語である．Pascalの特徴は，スカラー型・サブレインジ型・レコード型・ファイル型などのデータ型にある．初期の Pascal には，各種の欠点があったが現在は改良されつつある．

Ada は，国防省の支援によって開発されたプログラミング言語で，その命令の多くは Pascal と類似している．Adaで拡張された機能は，タスク（並行処理されるコード）間のコミュニケーション，エラー条件の処理，新しいデータ型などである．なお，データ型では，あるデータ型に属する対象のカプセル化 (encapsulation) を実現するプライベート型と，データ型をパラメタの例 (instance) として与えられるジェネリック型が追加されている．

関数的プログラミング言語は，FORTRAN の開発者として知られる J. Backus の提唱で有名になったもので，そのプログラムは入力を，目的とする出力に変換する関数として表現される．Backus は，関数的プログラミング言語が小さくかつ強力であり，効率のよい表現への変換や，正当性の証明が容易である，と述べている．

また，英語は，①データベースの問い合わせ言語，②データベースの操作言語，③汎用プログラミング言語としての利用が考えられるが，現在，Sperry Univac 社では，データベースに対する問い合わせ言語としての利用を研究している．(N. H. Cohen, "Programming Languages and Systems: Past, Present and Future", AUUA 1980 Fall Meeting)

技　　報

UNIVAC TECHNOLOGY REVIEW

No. 3

発　行　日	昭和 57 年 8 月 30 日
発行人兼編集人	富　田　和　夫
発　行　所	日本ユニバック株式会社 東京都港区赤坂 2-17-51 〒107 TEL (03) 585-4111 (代表)
頒 布 価 格	1,500 円
印　刷　所	三美印刷株式会社