技報

UNIVAC TECHNOLOGY REVIEW

1984年2月　第6号

日本ユニバック

データ・メタモデルとは，それぞれが1個の実世界のモデルである多くのデータベースの記述を共通に支える基礎的概念の枠組といえる．柳生の**CADのための data metamodel**は，現行の汎用またはCAD専用的のデータ・システムの内から代表的な例をいくつか選び，メタモデルとしての能力を分析し，評価し，困難を指摘している．さらに，データベースに対するCADに固有な要求の考察，実世界の認識方法についてのパラダイムの選択，選択されたパラダイムにもとづくメタモデルの構成，メタモデルを部分的に実現したと考えられる実用データベース管理システムの開発および適用，について論じている．

最近のいろいろな新しいプログラミング言語の流れは，仕様とプログラムとの距離の大幅な縮少，あるいは仕様記述とプログラミングの一体化の可能性を示している．長谷川の**仕様記述言語P3**は，著者らが開発した事務処理分野での仕様記述級のプログラミング言語の報告である．すなわち，事務処理の特色を分析し，仕様記述の骨組みを定め，これにそって組み立てられたP3の五つの部分（DATA-FLOW部，DATA部，PROCESS-FLOW部，PROCESS部，IMPLEMENT部）の概略を述べている．

パーソナル・コンピュータの利用環境における自然言語利用の実験が進められている．山田の**DCGトランスレータの製作と言語解析への応用**は，その中から有用と思われるDCGトランスレータのやさしい製作法を紹介し，簡単な応用例を述べている．ここでは，DCGトランスレータはパーソナル・コンピュータ用にMicro Prologを使って書いてあるが，PereiraとWarrenに述べられている全機能を含んでいる．また，DCGが組み込まれていないPrologへ容易に移植でき，自然言語あるいは人工言語の解析に応用できる．

垂直磁気記録は，現行の長手磁気記録技術の延長上にあることから，磁気記録技術者の関心の的になっている．とくに我が国では顕著であるが，ハード・ディスクについての実験はまだ少ない．C. S. Chiらの**垂直磁気記録と長手磁気記録の実験による比較**は，既存の記録再生ヘッドを用いハード・ディスクにおける垂直方向と長手方向の記録再生特性についての実験報告である．すなわち，スパッタ法で作成したCo-Cr垂直記録ディスクの作成法と，その磁気特性について述べている．また，記録再生の実験方法について報告し，高性能長手記録用のNi-Coめっきディスクと比較検討している．さらに，25端子付き遅延線で実験したHilbert変換位相等化器による出力波形改善効果についてもふれている．

視点からみて隠されている3次元物体の線や面を消去して表示する技術は，コンピュータによる図形処理の重要なものである．平林の**陰線消去プログラムHLE**は，高精度プロッタ用に開発された陰線消去プログラムHLEの紹介である．HLEでは，陰線消去は正規空間で実行され，処理結果は平面図（平行射影），パース図（透視射影）として出力する．現在までの適用例によると，相当複雑な構造物に対しても，その演算時間はほぼ満足すべき結果が得られている．

一般に任意の形状をもつ曲面板について，その弾性力学的挙動を記述する支配方程式を求めることは困難である．しかし，浅い曲面板についてはその曲率が小さいこと，すなわち，平面板に近いことを利用して近似的に弾性方程式を導入することができる．また，一般に深い曲面板は，多くの浅い曲面板の集まりとしてモデル化することにより，弾性方程式が確立できる．藤野の**浅い曲面板の弾性力学とその解析法**は，この浅い曲面板の弾性力学的挙動を記述する支配方程式を求め，有限要素法ならびに任意節点配置法によるその解析法について研究を行ったもので，シェル構造解析の精度，能率化についての数値実験の基を形成している．

☆

論文 CAD のための data metamodel

Data Metamodels for Computer Aided Design

柳 生 孝 昭

要 約 data metamodel とは，それぞれが1個の実世界の model であるような，多くの種類の異なる database の記述を共通に支え，具体的には概念 schema 言語として定式化される，基礎的，一般的かつ形式的な概念の枠組の謂であり，本稿はその CAD における在り方を論ずる．現行の汎用的または CAD 専用的な data system の内から代表的と目されるいくつかの例を選び，metamodel としての能力を分析し，評価し，困難を指摘する．さらに困難の原因を追求し背後に在る実世界を認識する方法の水準に至り，そこでの反省を経て，拠るべき paradigm として D. Hilbert の公理主義的な数学の方法を見出す．再び data の水準に戻り，選択した paradigm に導かれる metamodel 構成の大筋を示す．最後に実用 system として開発された LMS (Logical Structure Manipulation) をこの metamodel の部分的実現として位置づける．

Abstract This paper discusses data metamodels as they are or should be in cad applications, with the term 'data metamodel' meaning a basic, abstract and formal framework of concepts that commonly supports describing many data systems of diverse kinds, each modelling a specific part of the real world, and is to be formulated in a conceptual schema language. Samples representative of various approaches are taken from the population of currently available data systems, general-purpose or designed ad hoc for cad, analysed and evaluated in the light of modelling capabilities. Difficulties with them are pointed out and their cause is traced back to the underlying method for understanding the real world, the criticism of which results in the selection of a specific, alternate paradigm, that is, the formal axiomatic method of constructing mathematical disciplines by D. Hilbert. An outline for constructing a data metamodel in accordance with this paradigm is shown. Finally, the LMS (Logical Structure Manipulation) that has been developed for practical use is briefly described and regarded as a partial implementation of the said metamodel.

0. 課題と接近法

0.1 課　題

data metamodel とは，それぞれが1個の実世界の model であるような，多くの種類の異なる database の記述を共通に支え，具体的には概念 schema 言語[70]として定式化される，基礎的，一般的かつ形式的な概念の枠組の謂であり，本稿はその CAD における在り方について論ずることを課題とする．

このような課題の設定そのものがただちに，次のような批判を招くかも知れない．'実世界' とは意味不明の言葉である．仮りに意味があるとしても database と何の関りがあるのか，反対に database が実世界の model であるのは当り前のことではないか，また CAD の諸例に照らせば，異種の database にわたる共通の schema 言語のごときは幻想に過ぎない．等々．そこで始めに，これらの予想される批判を考慮に入れて，課題の正当性を確認しておかなければならない．

0.2 課題の正当化

1) CAD の database が実世界の model であること

database, より一般的には計算機内に納められた情報の集まりを model であると見る見方は多くの文献に現れている．例えば次のようである；

> An information system deals with objects and events in the real world.... These real objects and events ... are represented in the system by data[66].
>
> An information system (e.g., database) is a model of a small, finite subset of the real world[68].
>
> Information about the universe of discourse 'describes' or 'models' that universe of discourse.... Universe of discourse: the colletion of all objects (entities) in a selected portion of a real world or postulated world ...[70].

これらの文献はしかし，さらに立ち入って‘実世界 (real world)’とは何であるか，計算機の世界とどう違うのか，database がその‘model’であるとはどのような意味においてであるか，についてほとんど論ずるところがない．言い換えると，‘実世界の model である’のは自明な，ないしはそれほど深い意味のないこととして，つまり軽く言及されているに過ぎない．軽く言われることは，また軽く無視されることにもなる；

> (A database system) is nothing more than a computer-based record-keeping system ... to record and maintain information. The information concerned can be anything that is deemed to be of significance to the organization ...[67].

したがって，これらの例に訴えることによってはわれわれの主張，database 一般ではなく外ならぬ CAD の database がそれなりの理由によって実世界の model であるという主張を正当化することは，二重の意味でできない．

われわれは，database がそれについての一つの記述である世界，universe of discourse[70] が計算機の世界（＝離散化された data と algorithm の世界）に先立って，また独立に存在するとき，それを（計算機の世界との対比において）‘実世界’と呼び，さらに database がこのような世界の一連の挙動の体系を模倣すべく構成されているならば，それが実世界の model であると主張できるであろう．しかし，これらの前提は一般的に成立するわけではない．実際，通常の事務処理はほとんどそのまま計算機の世界であり，そこに model と対立する実物を見る理由は少ない．そうであるか否かは database の意味論にとっては極めて重要である．意味は前者の場合は実世界のそれに依拠するのに対し，後者では納められた data そのものの内に求めなければならないからである．CAD は下に見るようにそうである場合，しかも典型的な場合の一つであり，したがって database の意味論において特権的な地位を占めている；

1. 1) universe of discourse としての設計の世界，すなわち（設計という）人の営為，営為の対象である事物や記号，対象の間に成立する事態，それらの総体の持つ（上に述べた意味での）実世界性は明らかである．特に注目すべきはそれが，感覚的・物理的側面はもちろん，知的・論理的な側面と言えども計算機の上に完全に，つまり意味の欠落なしに model 化されることは期待できないから，modelling の後も自立的に存在し続けるという事実である．

1. 2) database は第1に対象を特定の局面でなく，形，位置，物理的性質，加工法，組立法などを含む全体性において捉えようとする，第2に納めた data から人の視覚に訴える像を再構成し，それに対してあたかも実物に対するのと同様の操作を許す，第3にそのような操作による変化がただちに（新たな対象の創成を含む）実世界の変化を含意するという，実世界との動的で有機的な連関の内に位置づけられている．

2) metamodel が有益であること

metamodel がもたらすであろう利益には理論的なものと実務的なものがある；

2. 1) 理論的利益は，さらに次の3種類に分かれる；

2. 1. 1) 異なる種類の対象の正に設計の対象であることに固有の性質を共通の言葉（＝

概念）による記述へと抽象し，定式化するための道具（＝枠組）という形で，それら対象を model 化する作業に対し一定の方法論的立場を提供する．

2.1.2) その道具によって切り取られる限りではあるが，設計の営為に照明を当てることを通して設計論の発展に寄与する．(寄与は双方向的である．すなわち設計論の成果が形式的な概念として定式化されて，metamodel の改良に役立つであろうことも期待してよい．)

2.1.3) 情報科学一般に顕著な抽象化・形式化の傾向へ同調することにより，他の分野，とりわけ言語，programming 方法論および人工知能の分野の成果を広く吸収することに道を開く．

2.2) 実務的利益は，metamodel が適切に構成され，効率の良い program として実現されるならば，汎用 dbms を手に入れることになるので，明らかである．'適切に' や '効率の良い' に対してはもちろん無限の要求がある．それにどれだけ応えられるかを問うことが metamodel の可能性を問うことに外ならない．

3) metamodel が可能であること

metamodel の可能性を十全に示すには，第1に有効性が実証されている(実)世界認識の方法の選択とその理由の説明，第2に選択された方法に基づく metamodel を実際に構成して見せること，第3にその modelling 能力が十分広くかつ高いことの証明，第4に metamodel の含む諸概念が効率良く実現される見通しを与えること，が必要である．このすべてを尽すのは本稿全体の範囲をさえ越えている．したがって，今は次の3点を指摘するに留めるが，それで課題を正当化するには十分であろう；

3.1) 事務処理における汎用 dbms の成功．

3.2) CAD においても少なくとも一部は汎用の，ないしは汎用化され得る dbms を使用している事実．

3.3) 汎用 dbms に対する批判はもっぱら実務的な，それもほとんどが上記の第4の点についての疑い，つまり (ad hoc な方法に比べて) 効率が悪いという評価に集約されるが，効率の改善，しかも大幅な改善はしばしば望めても，一度失われた一般性を回復するのは不可能に近い．したがって長期的には一般性をより重視するのが得策であること．

0.3 接 近 法

次の順序で議論を進める．

1) 現行の modelling における，われわれの課題の観点から認められる困難の指摘．
2) database に対する CAD に固有な要求の考察．
3) 実世界認識の方法についての反省と確認，すなわち paradigm の選択．
4) 選択された paradigm に基づく metamodel の構成．
5) metamodel の部分的実現としての実用 dbms の開発および適用の報告．

1. 現行の modelling における困難

3種の主要な困難が認められる．第1は対象の特定種類への限定による 'meta' 性の欠如，第2は狭い，十分に根拠づけられていない枠組の設定による記述能力の制約，第3は実世界の意味把捉の失敗である．

1.1 対象の限定

CAD の databaseの多くが modelling の対象を特定の種類に限定している．例えば多面

体[8]-[10]，多面体・部品・assembly[14]-[20]，形状[19]，図面[11]，回路・基盤[21],[22]，巨大行列[23]など．これには，おそらく二つの原因がある．第1に多くの dbms は特定の application の一環として，つまり ad hoc に開発されるから，（当面の関心の範囲を越える）設計の世界一般の考察というような厄介な作業を行う必要がない．第2に（主として事務処理を意識して作られた）汎用 dbms に適切なものがない．そのような限定の結果の第1は取り扱う対象に関する限り必要で十分な機能を備えた，効率の良い dbms，第2は応用 program と data の相互依存から来る困難である．後者について詳しく言えば；

1) 形式的操作の困難

model の構造の当の特定の application に依存する側面とそうでない側面が分離されない，またその多くの部分の記述が（procedure による）implicit な仕方に委ねられる，言い換えれば model についての知識が，両側面を区別する知識でさえも，その application の知識を前提とする結果，application から独立な側面のみに関る操作――これを‘形式的’操作と呼んだ．その一つの典型例は後述（2章の6）および 5.5 節の4））の複製の操作である．――の実現は不必要な重荷を負わされることになる．

2) model の拡張性の欠如

例えば，winged edge structure[8] は多面体の model としては経済的にできているが，曲面や穴を許すような拡張は構造の根本的な変更を要求する．（そして上述の，このような application の変更によっては無傷であるべき形式的操作にも少なからぬ影響を及ぼす．）

3) 方法論的貧困

model を記述する概念の枠組の metamodel 的有効性，すなわち対象が変わっても等しく発揮される記述能力は当然のことながら著しく限定されたものとなる．例えば回路のいわゆる topology の集合論的 model[22] は数値的属性を考慮していないので，結線の形状を問題にする場合[51]に対してはほとんど無力であろう．

1.2 狭い枠組による記述能力の制約

枠組の型によって類別すれば，ad hoc であると汎用的であるとを問わず，過去の model の大半が世界を有向 graph と見て来た．注目すべき例外は，連想三つ組による方法と集合論的構造である．近年これらに関係 model が加わった．関係 model については意味論的検討が重要なので次節に回すこととし，ここでは前三者を取り上げる．

1) 有向 graph

抽象的には2項述語と同値であるが，2項であるという構造の貧弱に加えて，両端の節の一方を主，他方を従とする非対称的な取り扱いや，弧をも節に従属させることによる制約が生じている場合が多い．その辺の事情を最も代表的な model の一つである ASP について調べ，次いで CODASYL model において制約がどう緩和ないし増大しているかを見ることにする．

1.1) ASP[5]

付加的な制約は，応用 program がその本来の関心事とは無関係な諸規定，すなわち3種の ring，それらの線形構造，associator の介在などへの依存（data dependency）という形に現れる．具体例をあげると4面体（図 1, 2）において，与えられた稜（$e1$）を境にして与えられた面（$f1$）と相対する面（f）を求める問題は下のように program される；

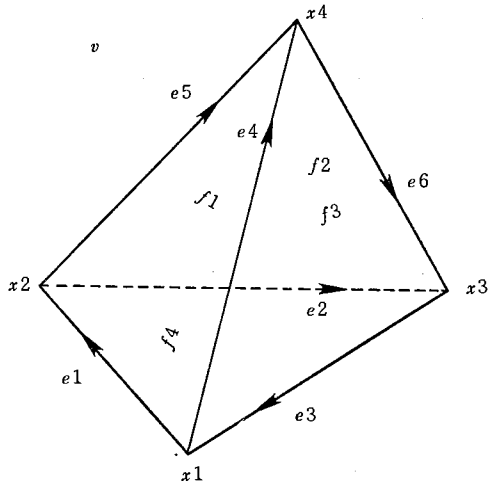


図 1 4面体
Fig. 1 A tetrahedron as a physical object

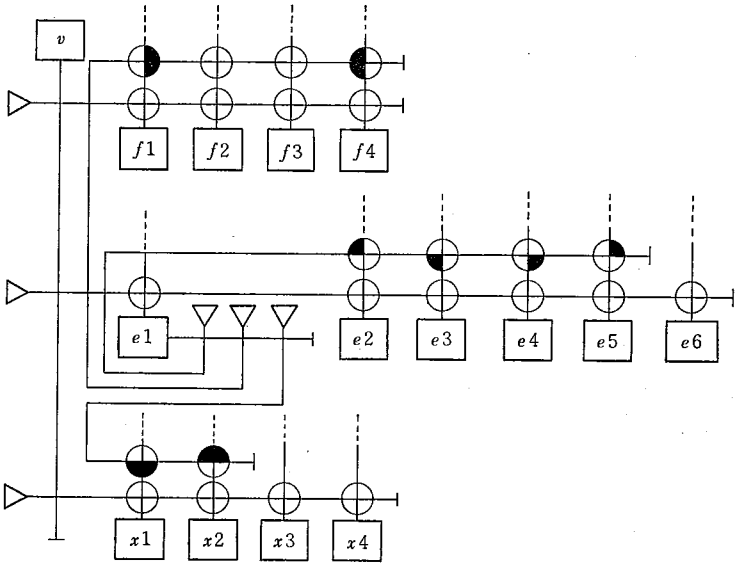


図 2 4面体の ASP model (winged edge 構造)
Fig. 2 An ASP model of a tetrahedron (winged edge structure)

rs:=**lrf**(*e*1); *rsname*:=**snam**(*rs*);
while *rsname*≠*ef* **do begin** *rs*:=**lrf**(*rs*); *rsname*:=**snam**(*rs*) **end**;
as:=**arf**(*rs*); *f*:=**ael**(*as*);
if *f*=*f*1 **then** *f*:=**ael**(**arf**(*as*))

ただちに感じ取られるのは問題の単純さに較べて度を越した煩瑣である．一体なぜこういうことになったのか？ 第1の原因は一つの稜と両側の面の間の関係を記述するには3項述語が最も自然であるのだが，それを欠くため，代りに二つの2項述語（面と稜との関係と同一性（=））の組み合わせによっていること（構造の貧弱），第2はその2項述語による関係づけも直接にではなく associator を介して行われ，しかも一方

から他方に至るには ring を辿らなければならないこと（付加的な制約）である．（実際 3 項述語が許されるならば，program は原子論理式を評価する 1 個の操作 **evaluate** $fef(f1, e1, f)$ で終るであろう．2 項述語のみによっても，もしそれが直接使えるならば，**evaluate** $ef(e1, f)-(f=f1)$ と書ける．）付加的制約を生じたさらに元の原因は上にも述べたように，述語（弧）と述語によって関係づけられる相手（終端節）を属性として項（始端節）に所属させる考えにある．この後の方の因果関係の技術的側面については，例えば古川の文献[2]に詳しいので本稿では省略するが，元にある考えの哲学的側面については後（3.1 節）に取り上げ，分析と批判を加える．

上の例では稜から面を検索したが，逆に面からその境界である稜を検索する問題を考えると，program の形が単に面と稜とを入れ替えたものとは異なることに気がつく．この事情を解消するには面の lower ring に境界関係を表す ringstart を加える，すなわち面と稜を結ぶ弧を二重化し，双方向化すればよい，またそうする外ない．つまり二つの項の対称的な扱い——E. F. Codd[30] が symmetric exploitation of relation と呼んだもの——を許す modelling は ASP の下でも不可能ではないが，それは model の自然さと簡潔さを犠牲にしての話なのである．（ついでながら B. G. Baumgart[8] の構造には今述べた面から見た境界関係を追加することができない．と言うのは関係を一方の項の属性として扱うのが ASP よりさらに徹底している．また要素の型ごとに属性の持ち方が斉一である．したがって，もう一方の項である稜の個数が不定な境界関係に明示的な表現を与えることができないからである．明示されていない関係による検索は，それが含意する他の明示的な関係に頼ることになる．今の場合は一つの面の境界上の稜全体が，どれから出発しても一定方向に閉じた連鎖を形作るという事実を利用する．この事実自体は（多面体という）対象に固有であるが，知識として表現し利用する枠組を対象から独立に考えることはできるだろう．それは metamodel 論議の重要な一部分である．しかし多面体に対象を限る Baumgart の立場からは無意味であり，ASP の場合でも記述能力の低さから，親言語による手続きの内に埋没せざるを得ないのである．）

1. 2) CODASYL model[24]

metamodel の観点からすれば CODASYL model は ASP の element と association ring をそれぞれ record (occurrence) と set (occurrence) によって置き換え，ringstart（したがって lower ring）と associator（したがって upper ring）を廃し，record および set の型の概念を導入したものにほぼ等しい．二つの ring の廃棄による可変長の pointer 領域の問題は厳格な型づけによって回避される．したがって，両者の主な違いは associator と型づけの有無にあると見てよい．そう見るとき CODASYL model において ASP の制約の緩和される点は第 1 に 1 対 n 関係に限れば（associator に相当する）link record を介在させる必要のないこと，第 2 に compile 時の型づけによって実行時の自動的な関係づけが可能であること，第 3 に key あるいは calc key によって検索の自由度を増していること，逆に制約の増大する点は第 1 に homogeneous set (owner と member が同一型に属する set) の禁止，第 2 に静的な型づけによる諸々の制約，第 3 に member による set 選択の許されないこと，そして両刃の剣のごとき currency 制御がある．これらは，しかし副次的な差異であって，制約の本質に変わりはない．実際（CAD でしばしば要求される）m 対 n 関係を実現する構造は ASP によく似たものとなる．

2) LEAP[6]

LEAP は三つ組を成す item を object, attribute, value と呼ぶ独特の言葉使いのために，ASP と同類の有向 graph の software と見なされがちであるが，(例えば古川[2])，事実は ASP よりもはるかに豊かな世界を構築する．と言うのは第1に三つの item はすべて対称的に扱われ，また属性 (datum) を有することから，一つの2項述語と二つの項というよりむしろ (association という) 特別の3項述語で関係づけられた三つの項というに近い，第2に item の集合の定義と通常の演算が許される，第3に三つ組を一つの item として，したがってその集合によって一般の3項述語を，また他の item との association によって4項以上の述語を定義できるからである．この世界は実質的に1階形式理論の model の部分であり，それにかなり近いが同等ではない．例えば x が満たすべき条件 $a1\cdot\bar{o}\equiv x$ または $a2\cdot\bar{o}\equiv x$ を x **in** $(a1\cdot\bar{o})\cup(a2\cdot\bar{o})$ と集合算的に書くことはできるが，$(a1\cdot\bar{o}\equiv x)\vee(a2\cdot\bar{o}\equiv x)$ とは書けない．同様に $a\cdot\bar{o}1\equiv x$ なるどの x についても $a\cdot\bar{o}2\equiv x$ を成立させるために **foreach** $a\cdot\bar{o}1\equiv x$ **do make** $a\cdot\bar{o}2\equiv x$ あるいは **make** $a\cdot\bar{o}2\equiv(a\cdot\bar{o}1)$ はよいが，**make** $\forall x(a\cdot\bar{o}1\equiv x\rightarrow a\cdot\bar{o}2\equiv x)$ は許されない．今度は $a\cdot\bar{o}1\equiv x$ なるどの x についても $a\cdot y\equiv x$ であるような y を検索する心算で **foreach** $a\cdot y\equiv(a\cdot\bar{o}1)$ と書くと，$\exists x(a\cdot\bar{o}1\equiv x\wedge a\cdot y\equiv x)$ の意味になってしまう！ この種の検索条件 (一般的な形は $\forall x(p(x)\rightarrow q(x,y))$ を非手続き的に書く手段はない(図3)．

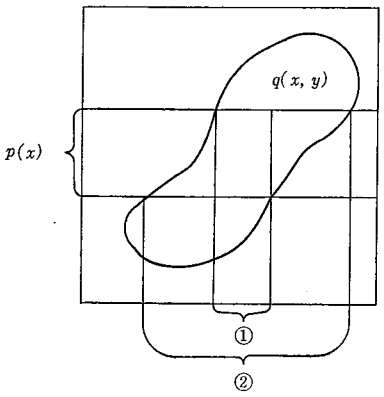


図 3
①$\forall x(p(x)\rightarrow q(x,y))$
②$\exists x(p(x)\wedge q(x,y))$
Fig. 3

LEAP の本質的な困難は今挙げたようなあれこれの機能の欠落そのものよりも，どのような欠落が生じているのか，言い換えれば1階形式理論の model のどのような部分世界を形作っているかを正確に，明瞭に見通すことを許さないところにある．だから ASP の同類では決してないのだが，'自然に' 使えるのがその程度であるのも事実なのである．それは，さかのぼれば概念の枠組が充足性と最小性を欠くことに起因する．形式理論の枠組によればより簡潔に，より豊かな世界を築き得たであろう．もちろんそうしなければいけない理由はない．しかし LEAP は自らを根拠づける有効な paradigm を提示していないのである．

3) 集合論的構造[4]

Childs の理論は LEAP と同じく属性を持つ基礎的な（すなわちそれ以上分解できない）要素の集合から出発し，その上に構造を組み立てる．出来上る構造もほぼ同等，したがって1階形式理論の部分的な model になっている．しかし組み立ての道具には違いがあり，LEAP の場合は三つ組と集合論的概念の二本立てであったのが，後者に一元化されている．このことは重要である．なぜならば

第1に集合論という確かな基盤に拠ることで記述能力を増し，またその限界についての見通しを簡明にした．例えば前項に挙げた $\forall x(p(x)\rightarrow q(x,y))$ はその双対式 $\exists x(p(x)\wedge q(x,y))$ に対応する表現 $Q[P]$ (P,Q はそれぞれ p,q の外延) を用いて $\sim(\sim Q)\ [P]$ と書ける．

第2に集合論的概念の意味は最終的には要素に関する論理式に即して理解されるのであるが，要素への直接的参照や述語論理的語彙の使用を止めることで (Codd[33] が示唆したように) 意味を蔽う結果となる．例えば $p(\boldsymbol{x})\wedge q(\boldsymbol{y})$ は引数の list の重複の仕方に応じて，すなわち $p(x)\wedge q(x)$, $p(x)\wedge q(y)$, $p(x)\wedge q(x,y)$, $p(x,y)\wedge q(y,z)$ に対してそれぞれ $P\cap Q$, $P\times Q$, $Q[P]$, $P[2=1]Q$ (最後のもの (join) は Codd[33] の記法であって Childs のではないが，それは本質的なことではない.) と書かれるが，これらの表現から元の論理的表現の示す，場合の違いを超えた同一の意味，要素 $(\boldsymbol{x},\boldsymbol{y})$ について $p\wedge q$ なる事態が成立するという意味を読み取ることはもはや難しい．

第3に今の例からも分ることであるが，記号体系が乱れる．特に projection や join は関係する引数を何らかの形で示さなければならないから，述語論的記法をまったく捨てるわけにはいかない，というよりも結果は却って錯綜したものになる．例えば $Q[P]$ も正確には $Q[1=1]P$ と書かなければいけない．しかも書き手が残った引数，したがって P, Q の引数の何であるかを知っていることが前提されている．

第4に一つの集合式を書き下す過程を，書き手は，各々の単項または2項操作ごとに一つの集合を作り出す，その動作を繰り返して最終結果を得る手続きとして意識するであろう．同じ手続きは（それが踏むべき唯一の手続きではないが）計算機に対して指示されてもいる．すなわち集合論的構造は手続き的な性格を持つ．この事情は relational algebra が (relational calculus に対して)手続き的であると言われる (Codd[33]) のに似る．

以上を要約すれば，Childs の理論は拠って立つ paradigm の選択を明示的に行っている点で評価されるが，その選択は結果として得られる枠組の意味表現力，記号体系としての洗練および完結性に照らして必ずしも適切であったとは言い難い．

なおこれも LEAP と同様であるが，要素の属性の如何が構造に何ら関与していないことも一つの欠点として見逃すことができない．

1.3 （主として関係 model における）意味把捉の失敗

関係 model に拠る諸例[40]-[51]が共通して挙げるその利点は，単純な形式 (table) による表現の統一，data の物理構造と接近経路からの独立，数学的に厳密（形式的）な取り扱いを許すこと，それに高水準言語，具体的には代数（集合演算）的または述語論理的言語による記述方式である．これらは関係 model が提案された歴史的事情，すなわち従前の model に対する反定立として提出された事情に必然的に結びついており，したがって CAD に適用される場合だけでなく，一般的に主張できることに違いない．従前の model の弱点とされた data dependency はしかし，世界の意味を data 構造の内に捉えようとする企

ての（望ましくない）副産物であり，弱点は企て自体にではなく，企てを実現する方法の不適切にあった．ところが関係 model は委細かまわず，data の世界をただただ flat なものとして眺めようとする．結果は overkill すなわち（物理構造という）盥の水と一緒に（意味を捉える枠組という）赤児まで流すことになった．ここにしばしば指摘される関係 model の意味論的困難が生ずる．われわれはこの事情を Codd の一連の論文[30]-[33]にさかのぼって調べ，次いでいくつかの代表的な意味論的拡張の試みについて検討し，最後に問題が関係 model だけではなく，W. Kent[69] が 'record-based' と呼んだ model 一般に共通するものであることを見るであろう．

1) Codd model における多義性の問題

1.1) 存在 (entity) と表明 (assertion)

database は言葉による model であり，言葉は存在を指し示す，または存在についての事実を表明する．この区別は言葉によって世界を秩序づけ，理解可能なものとするために基本的である．例えば表明はそれが関る存在のいずれか一つでも消滅すれば無意味となるが，逆は真ではない．存在はどのような特定の表明もなしに，同一の存在として存在し続けるからである．

Codd の model について言えば relation は domain の間の関係であるから，domain の元(属性値)が存在，tuple が表明ということになるだろう．ところが tuple があたかも存在であるかのように扱われる．すなわち，まず tuple は element または entity と呼ばれ，実際 database の集合としての操作によって最も基本的な要素として扱われる．第2に属性値の変化を通じて不変の同一性を保持し，domain の元はその同一なる要素にたまたま付随する性質 (attribute) に過ぎない．第3に述語論理的な言語 (ALPHA) において，quantification の対象は tuple であって属性値ではない．ことはしかし単純ではない．と言うのはいわゆる foreign key の概念は tuple を再び表明の地位に引き戻すからである．こうした両義性の疑いに対して Codd は次の言明が示すように，開き直りとも言える態度を取る；

> This distinction (of entity descriptions and relations between entities) is difficult to maintain when one may have foreign keys in any relation. ... there appears to be no advantage to making such a distinction[30].

こうして存在と表明の別は意図的にまったく曖昧にされる．

以上の言わば概念上の混乱は実務にどう持ち込まれるであろうか？ 上に表明の一つの特徴として挙げた存在条件を ALPHA 表現として書くことはできる．しかし，それは一般に application に依存する諸々の整合性条件の一つとして書かれるのであって，その特権的な重みに相応しい扱いを受けるわけではない．key の変更——関係 model にとって常に頭の痛い問題であった——を foreign key に波及させることは静的な条件として記述することさえできない．この危険な仕事は存在条件の事後の確認を除き，応用 program に委ねられる．納得し難いのは application に独立な，しかもこれほどに基本的な条件の維持に対して system が極めて冷淡なことである．

1.2) 存在と相

1個の存在は例えばある形を成し，ある物理的性質を持ち，また他の部分と様々な関係で結ばれるというように，多くの相を呈し，一般にそれらは複数の relation に分散して記述されるであろう．しかし関係 model は異なる tuple をそれぞれ別個の存存として扱うのが基本であって，まとめて1個の存在として認識する仕掛を持たな

い．もちろん relation の分割を nonloss-joinable であるようにする，また join の相手の存在を要求する条件——前項と同じく ALPHA で書ける——を設けることはできる．しかし join を実行して存在のすべての相を検索したり，一つの tuple の更新を相手方に及ぼすことは——これも同じく——応用 program の仕事となる．

相を統べる存在を認識しないどころではない．正規化によって relation は分割され，存在はさらに多くの存在へと増大させられる．Ockham は公然と挑戦されるのである．

1.3) key の両義性

例えば「2点が同じ位置にある」という言い方をする．また「この点が位置を変える」とも言う．これらの例は1個の存在が他から識別されていることと，何らかの不変ないし固有の属性を持つことの違い，しかも素朴な例であるだけに一層，違いの根源的であることを示している．日常の物理的・感覚的世界——設計の世界をその一部として含むような——において，われわれが対象を属性値の様々な変化にも拘わらず同一の存在として認識し続けることはよく見られるのであるが，それはわれわれが安定した世界認識を持ち得るための基礎である．また，異なる存在が同じ属性値を持つことを許す立場でもある．同一性は（例えば点の位置のような）全属性を以てしても決定できない，同じものは同じだから同じなのだと言う外ないほどに基本的な概念なのである．

key とは識別作用と述語作用の両義性の具現化の産物である．それは第1に存在は上述とは正反対に特定のいくつかの属性によって識別できる，第2にその特定の属性はあらかじめ存在の型（種類）ごとに定められているという二つの前提の上に立っている．key に関る困難は第1に(例えば点のように）自然な属性の内に key 候補を見出せない場合がある．これに対する自明な解決は識別子としてのみ働く人工的な属性の追加であるが，応用 program が自らにとって無意味な，通常の属性とは違って変更を許されない値の付与と管理を強制されるのは如何にも馬鹿げている．第2に key の変更は一筋縄ではいかない．と言うのは変更の瞬間に前後の値が同一の存在に属すると判断する手掛りを失うからである．だから ALPHA では key の変更に限って modify ではなく delete 次いで create を行わなければならない．しかし，もちろん両者は本質的に異なる操作である．key と言えども属性である以上変更の可能性を持つはずであるが，事実上それは禁止されている．第3に有効範囲の問題がある．database 全体にわたっての一意性は望ましくも望むべくもないが，union や join で共に扱われる範囲——union compatible または jonable な全範囲の一部——内の（全属性を決定するという意味での）一意性は要求される．この範囲はしかし一般に application に依存するから変更の可能性があり，実際変更しようとすれば大きな問題をひき起こすであろう．また，同一の存在が属性の持ち方を変えるに伴って属する relation を変える場合を考えると（本節の 1.5）項参照），有効範囲を union compatibility や joinability に基づいて決めるのも難しい．

これらの困難を解決しようとする試みは結局，system が与え，管理する，変更が許されない，database 全体を通じて一意的な識別子（system key[48],[50] または surrogate[69]）の考えに導かれるであろう．しかしその考えは関係 model だけでなく，record-based model 一般にとって異質のものである．

1.4) 構造の貧弱

構造とは存在の間の諸関係つまりは表明の総体に外ならないから，先の 1.1) 項で触れた，foreign key を含む tuple が表明と見なせることと合わせて，関係 model は構造を持つと言えなくもない．実際 ALPHA 言語は述語論理の体裁に作られている．しかしこれも 1.1) 項で指摘したように，quantify されるのは tuple であって属性値ではない．foreign key を含む relation に対してもそうなのである．このことは tuple がやはり存在と捉えられており，表明ではないことを示唆する．それが概念上の欠落に留まらず実際 ALPHA の表現力に制約を加えているか否かは，関係 model が真に構造化されていることの試金石になるであろう．そこで具体例として次の問題——Codd の文献[33]による．ただし，ここでは検討に必要な最小限の形に単純化した．——を考える；

次数 1 の relation *supplier*(S, 属性 $s^{\#}$, tuple 変数 s)，*project*(J, 属性 $j^{\#}$, tuple 変数 j)，および次数 2 の relation *supply* (P, 属性 ($s^{\#}$, $j^{\#}$), tuple 変数 p)が与えられているとき，すべての project に供給する supplier の番号を求めること．

まず Codd 自身の答は

$$s[1] : S(s) \wedge \forall j(J(j) \rightarrow \exists p(P(p) \wedge s[1] = p[1] \wedge p[2] = j[1])) \qquad ①$$

があるが，問題の relation P が S と J の間の 2 項述語であるならば，より単純に

$$p[1] : P(p) \wedge \forall j(J(j) \rightarrow p[2] = j[1]) \qquad ②$$

と書けそうである．実際②は文法的には正しい ALPHA 表現であるのだが，意味はどうであろうか？ 意味を定める決め手は relational algebra への還元にあるから，Codd[33] に示される手続きに従って解釈すれば次の通りである；

step 1 (標準形への変形) $p[1] : P(p) \wedge \forall J(j)(p[2] = j[1])$

step 2, 3 (range の決定) $U = P \otimes J$

step 4 ($p[2] = j[1]$ に対応する U の restriction) $T3 = U[(2) = (3)]$

step 5 ($\forall J(j)$ に対応する division) $T2 = T3[(3) \div (1)]J$

$T2 = \phi$ (J が 2 個以上の tuple を含むとき)

P (J が 1 個の tuple より成るとき)

step 6 (target list の計算) $T = \phi$ または $P[1]$

落し穴は明らかに U あるいは $T3$ の定義にある．本来 step 5 で求められるべきは $P[(2) \div (1)]J$ であったのだ．それに対応する論理式は

$$s^{\#} : \exists j^{\#}(P(s^{\#}, j^{\#})) \wedge \forall j^{\#}(J(j^{\#}) \rightarrow P(s^{\#}, j^{\#})) \qquad ③$$

となるはずである．これは，しかし正しい ALPHA 表現ではない．一般に (LEAP についての検討の際も取り上げた) $\forall x(p(x) \rightarrow q(x, y))$ の形の論理式は ALPHA では評価できない．目的を達するには $\{y : \exists x q(x, y)\}$ なる relation をあらかじめ作っておかなければならない．ここで次のような反論が予想される．問題の検索が trivial なものでない限り q の二つの domain は互いに独立であろうから，正規化によって y の属性を記述する relation は始めから存在しているはずだ，それに key だけの検索というのも実務上ありそうにないから，所詮③の形では目的は果たせないのだ，と．それでは次の問題はどうか？

$$\forall y(\forall x(p(x) \rightarrow q(x, y)) \rightarrow r(y, z)) \qquad ④$$

第 1 に $\forall x(p(x) \rightarrow q(x, y))$ なる形の range は許されない．第 2 に仮りに許されたとしても上に述べたような——今の場合理論的にも実務的にもまったく不要の——

relation $\{y : \exists x q(x, y)\}$ との join term を介在させなければならない．無駄の発生源は明らかに属性値を自立的な存在として quantify できないことにある．

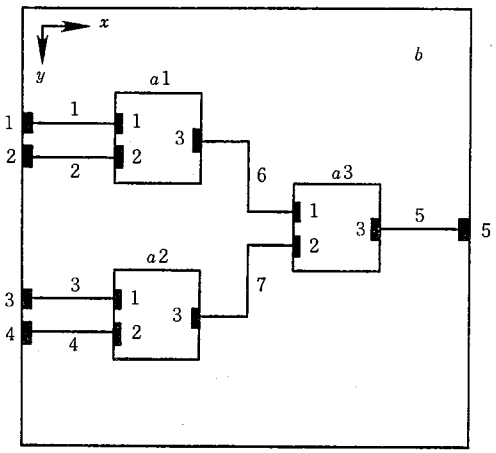
③や④（特に $Q[(1) \div (1)]P$）の評価は relational algebra の内だから，同時に relational calculus の表現力が algebra より弱いことも示された．（後者が前者と同等以上であること，いわゆる completeness は Codd[33] の主題であったのだが，Codd がその逆にまったく触れていないのはいささか奇異の感を抱かせる．）

以上の考察からわれわれは次のように結論する．すなわち関係 model において tuple は存在であり，relation は単項述語である．構造はこの単項述語と，属性を tuple の集合から親言語の data 型への関数と見るとき，＝や＞から逆関数によって導かれる2項述語から成る．relation の n 項述語の形は見掛けに過ぎず，真実は n 個の関数の並びである．

1.5) 存在の型と属性の斉一性

例えば3角形状の部品の一辺を直線分から円弧に変えると，属性の持ち方の変化に応じて辺の属する relation も変わる．しかし先の 1.3) 項でも述べたように，われわれの普通の認識では単なる属性の変化は同一性を失わせるものではない．すなわち部品と辺はそれぞれ同一の存在であり続ける，両者の関係‘一方が他方の境界である’にも変わりはない．同一の存在であれば如何なる種類(型)の存在であるかも変わるはずがないが，関係 model において relation（の union compatiatible な類）は型を決定するものであるから，はなはだ具合が悪い．やはり 1.3) 項で指摘した key の有効範囲の問題も生じる．

次の例は R. L. Haskin と R. A. Lorie[51] による（図4）．



geometry

ent-id	*x*	*y*
a	0	0
a	0	10
a	10	10
a	10	0
b	0	0
b	0	50
b	50	50
b	50	0

図 4　回路の model

Fig. 4　A circuit and its relational model

回路の要素，例えば a の4角形状は relation ‘*geometry*’ の4個の tuple によって表されるが，複数の tuple が互いに独立でない，しかも並ぶ順序に意味があるというのは，関係 model の長所を台無しにする，真に苦しい事情と言わなければならない．1個の tuple で表すのが簡明であるが，形が変われば relation を異にするという上と同じ問題をひき起こす．同様の model は R. Williams[40] にも見られるが，それは関係 model を graphics に応用する最初期（1974年）の試みの一つであった．

Haskin と Lorie の報告は 1982 年であるから，形状の関係 model による表現には 10 年近くほとんど進歩がないことになる．

属性の斉一性による対象の型づけ（分類）は，その原型である伝票や帳表の様式が重要な役割を果たす事務処理に対しては，適切な枠組を提供する．しかし言葉の最も広い意味においては——言葉の意味を広く取るのはわれわれが，‘実世界’を意識することの実践的側面である——二つの存在は類似の挙動を示す，あるいは取り扱いを受けるとき，同じ型に属すると言える．‘類似’とは曖昧な言葉であるから‘型’の概念も流動的とならざるを得ないが，少なくとも属性の斉一性すなわち型とするのが根拠を欠く，硬直した見方であり，設計の世界に対しては限られた有効性しか持ち得ないことは指摘しておこう．

1. 6) 正規化の意味論的役割に対する疑問

第 1 に正規化がいたずらに存在を増すばかりか新たな anomaly を生みかねないことはすでに 1. 2) 項に指摘した．第 2 に一般に一つの文脈の内に置かれた二つの項は一つの意味を成すから，正規形は必ずしも単一の意味を意味しない．正規化が潜在的な意味複合を顕にする場合もある．例えば embedded MVD など．第 3 に属性が切り離される結果，何ほどかの意味喪失は必然である．C. Beeri と P. A. Bernstein[39] が指摘しているように，C. J. Date[67] の例，関数従属 $AB \rightarrow C$, $C \rightarrow B$ を持つ relation $R(A, B, C)$ の $R_1(A, C)$ と $R_2(C, B)$ への分解によって失われる $AB \rightarrow C$ は join $R_1 \times R_2$ によっても回復されない．

そもそも正規化は個々の従属関係を分離して取り出し，維持しやすくするのが目的であった．したがって本質的には整合性を保証する一手段として位置づけられるべきであって，意味論的役割を荷うとただちに認めるわけにはいかないのである．

2) 意味論的拡張の試み

いわゆる semantic model の提案は非常な多数に上るのですべてを尽すのはもちろん，系統的に整理して検討するのも難しい．そこで最も代表的と見られ，しばしば参照される三つの文献と CAD への適用例若干に限ることにした．しかし関係 model を意味論的に拡張しようとするときの障害，拡張後も残る困難について要所を押えるには十分であると考える．

2. 1) H. A. Schmidt と J. R. Swenson の方法[35]

2. 1. 1) model の概要

Schmidt と Swenson は relation を次の 5 種に分類した．type 1 の tuple は 1 個の independent object を表す．type 2 は type 1 の repeating characteristic を，type 3 は non-repeating characteristic を，type 4 は type 3 の repeating characteristic を表す．type 5 は type 1 の間の関係の成立を示す．type 2, 3, 4 は意味と生成・消滅の上で type 1 に従属し，後者の 1 個の tuple から出発し，次々に参照される前者をすべて結合したもの (complex independent object) は実世界における 1 個の対象（存在）を表すとされる．世界は対象と対象間の関係 (type 5) から成る．形式的には type 3 を 1 から区別するのは第 3 正規形への，2, 4 をそれぞれ 1, 3 から区別するのは第 4 正規形への正規化に外ならない．

2. 1. 2) 困難がどう解決されたか？

存在と表明の別は type 1〜4 と 5 の分離によって果たされた．しかし単一の存在の複数の tuple による表現，その背景にある属性の斉一性への拘泥，key の概念，正規

化などに対する疑問は解消しない．可変長の属性を許せば model は type 1 と 5 のみを含むように単純化されるであろう．type 5 が真に構造を実現しているか否かは，1.4) 項で述べたようにむしろ言語——文献[35]では扱われていない——の問題である．

2.2) P.P. Chen による entity—relationship model[36]

2.2.1) model の概要

Chen はCodd と正反対の言明から出発する．曰く

> The entity-relationship model adopts the more natural view that the real world consists of entities and relationships. ..... The method ..... is to separate the information about entities from the information about relationships.

Chen のmodel は Schmidt-Swensonのそれに似る．すなわち regular entity (*re*), weak entity (*we*), regular relationship (*rr*) はそれぞれ type 1, 2, 5 に相当する．ただし，type 3, 4 はそれぞれ 1, 2 に昇格し，これに伴って type 1, 3 は *re* と *rr* に分割される．また Schmidt-Swenson にはなかった characteristic object 間の関係が weak relationship (*wr*) として許される．

2.2.2) 困難がどう解決されたか？

上記の対応に沿う言い替えをすれば 2.1.2) 項で述べたことは Chen の model についてもあてはまる．加えて次の点にも注意しておこう．第1は *we* および *wr* というのは中途半端な概念であって，存在条件を設定しておけば前者は *re* として扱い得るし，後者はそもそも *rr* と区別する必要があるとは思われない．第2に relationship relation とか relationship の属性とは如何にも奇怪な用語である．その奇怪さは取りも直さず，属性の斉一性を柱とする relation なる枠組への拘泥が存在と表明の別を蔽っていることの現れである．

2.3) J.M. および D.C.P. Smith による aggregation および generalization[38]

2.3.1) model の概要

relation の集合に2種類の抽象階層が導入される．R が aggregation の意味で S の上位にあるとは R の属性が S の key を含むこと，generalization の意味で $\{S_1, \cdots, S_p\}$ の上位にあるとは各 S_i が R のある属性 A について同一の値を取る tuple に S_i 固有の属性を追加した形になっていることである．$\{S_1, \cdots, S_p\}$ を cluster と言う．一つの R がいくつもの cluster を持ち得る．model を導くのは特殊な意味論的立場である．すなわち第1に各 relation を自然言語の一つの名詞に対応する対象 (generic object) であると捉える，第2に対象間の関係をより抽象的な対象として位置づける（表明の事物化），第3に類としての対象を特定の属性を種差として再分類する．形式的には aggregation 軸での分割は第4正規形への正規化，generalization は foreign key による関係づけの特別な場合である．関係づけは関係づけられる tuple の一方の内で為される，すなわち Schmidt-Swenson や Chen の relationship の概念はここにはない．

2.3.2) 困難がどう解決されたか？

存在と表明の別は再び曖昧にされた．generalization の軸に沿って関係づけられる複数の tuple が複数の存在なのか，同一存在の異なる相であるのかも判然としない．例えば削除を関係する tuple に及ぼすべきか否かは 'higher level of modelling' に委ねられる．Smith の model の構造は抽象階層という言葉が示す通り本質的に樹構造であり，複雑な外観の割合には貧弱であると言わざるを得ない．

2.4) CAD への適用例

Codd の model に忠実な実現はまったくと言ってよいほどなく，いづれも以下のように何らかの改変を含んでいる．

2.4.1) 順序づけられた複数の tuple による対象の記述[40],[42],[44],[50]

これについてはすでに 1.5) 項に述べた．

2.4.2) 属性としての関係[40],[42],[44],[46]

第1正規形を否定することになるが，単なる形式的制約の緩和ではなく，階層の導入を意図している点が重要である．

2.4.3) 属性としての手続き[41],[42] あるいは数式[45]

値は手続き呼び出しや数式の文字列ではなくそれらを評価した結果であるから，このような拡張は関係 model の定義を大きく逸脱する．

2.4.4) system key の設定[48],[50]

これも先の 1.3) 項で述べた通りである．

2.4.5) entity-relationship model の採用[43],[47]-[49]

e-r model は第1に個物と個物間の関係を entity と relationship に対応させるという仕方で，設計対象を‘自然に’捉えることを許す，第2に (relationship が一般の多項述語であるので) 複雑な関係を記述できる，第3に概念構成が比較的簡明であるという理由で，CAD に適用される関係 model の最も有力な形の一つである．例えば Grabowski-Eigner[48],[49] は他の entity についての述語を characterizing entity，主・述の結合を characterizing association と呼ぶが，それぞれ Chen の weak entity, weak relationship に当たる．ただし char entity は複数の entity と結びつき，もし属性値を変えるならばすべての相手方に影響を与える，すなわち一定の自立性を持つが，他方それが現れる association がすべて消滅すれば自動的に消滅させられる点で，存在論的性格を曖昧にしている．

2.4.6) 抽象階層の導入[45],[48],[49]

Grabowski-Eigner[48],[49] はまた異なる entity 型をまとめて一つの型 (generic object) と定義することを許すが，これは Smith の abstraction に由来する．R. Phillips[45] の階層を構成する class header, tuple, basic datarecord はそれぞれ Grabowski-Eigner[48] の generic object, entity, characterizing entity に相当する．

2.4.7) 巨視的対象 (complex object) の認識[46],[50]

Schmidt-Swenson 的なものだけでなく，例えば Haskin-Lorie の例 (図4) では要素 b, b が使用する a, b における結線をまとめ，a がさらに細かい要素を用いているならば同様のことを繰り返して得られる data のすべてを1個の complex object として認識する．

2.4.8) CODASYL model との hybrid 化[46]

model の改変の行き着くところは CODASYL model との hybrid 化である．database 全体は set 的な階層関係によって構造化されるが，1個の owner に連なる member 全体は1個の relation を形作る．このような接近法の動機は第1に関係 model の利点を生かしつつ，第2に設計対象に見られる自然な network 構造を捉える．第3に冗長性の除去と効率の向上を図ることにある．

3) 未解決の困難

第1に上に取り上げた拡張の試み——無数の試みの一小部分でしかない——のすべ

てを包含する体系は無く，おそらくは可能でもないであろうし，仮りに可能であったとしても関係 model と呼ぶには Codd の定式化から隔たること余りに遠い．第2に存在と属性，存在と表明，識別作用と述語作用は依然として混同されたままである．例えば Grabowski-Eigner[49] によれば entity e が属性値 $a_1, \cdots, a_n$ を持つとき，各 a_i を entity 化し，2項述語 $p(e, a_i)$ により関係づけるのが（a_i の変更を関係するすべての e に及ぼすには）望ましい，だが e をそのように $\wedge_i p(e, a_i)$ として捉えるか，属性値の集まり $(a_1, \cdots, a_n)$ として捉えるかは使用者や応用 program にとってどうでもよいことである．混同の背後には存在とは属性の集まりである，他の存在との関係も当の存在の属性の一種である，同一の型(種)に属する存在は同一の属性を持つとする世界観がある．混同が避けるべきことであるならば——それは確かに避けるべきことであるが——このような世界観に固執するのはなぜか？ それはどのように正当化されるのか？（正当化され得ないであろうことをわれわれはいくつかの例に見た．）関係 model に対するこの疑問は様々な改変によっても遂に解消されることがない．と言うのも問題の世界観がより広い範囲の model, record-based information model[69] の根底にあるからである．

2. CAD に固有な要求

以下は諸文献[7]-[23],[26]-[29],[40]-[51],[56]-[65] の指摘と筆者自身の経験を整理したものであって，理論的に導いた結果ではない．したがって言わば必要条件ではあっても十分条件とは言えない．十分条件の導出には設計論の発展を待たなければならないであろう．とは言え，これらの項目だけでも（database 一般に共通する基本的な要求に加えて）洗練された体系の内に，しかも効率よく機能するように実現するのは，今日の技術水準では至難のことであるに違いない．

1) 複雑な構造

少なくとも m 対 n 関係，おそらくは一般の多項関係が要求される．加えて属性領域から導かれる implicit な構造の取り扱いも可能でなければならない．

2) 長大な，しかも可変長の data

例えば有限要素法による解析のためには必須である．Haskin-Lorie[51] の cursor を用いた区分的読み取りの方法は注目される提案の一つである．

3) schema の動的定義および変更

属性の持ち方や関係の種類をあらかじめ静的に決めておくのは CAD においては硬直的に過ぎる．動的な方法が効率と整合性の維持の上で難しい問題をひき起すのはもちろんであるが．

4) 整合性条件の設定および維持

多相的な認識，したがって情報の冗長は実世界においては不可避である．しかし実物が現にそこにあるということがあらゆる論理を超えて，整合性を決定的に保証する．これに対して database の整合性は数学的な model としての無矛盾性であり，動的な環境の下では常に脅かされているものと見なければならない．整合性条件は，また公理として外延的に存在しないが論理的に導出される情報の源でもある．

5) 接近および旋錠の単位としての巨視的対象

前述 1.3 節の 2.1.1) 項および 2.4.7) 項の通りである．

6) data の複製機能

database の一部を複製して利用するのは，ほとんど CAD の常識の内と言える．例えば既設計対象の一部変更，類似形状の繰り返し，代替案の検討など．その際しばしば見られることの一つは，対象の属性値は変えるが対象の間の関係は不変に保つ複製である．属性値の変更は一般に application に依存するが，関係の維持は維持される関係の如何に依らないという意味で形式的であり，system に期待される機能である．

7) 私的作業領域 (private work area)

複数の設計者が共通の対象を相手にする場合，database の一部について特定の1人がある期間排他的に更新しながら，他の設計者には元の data の読み取りを許す必要が出て来る．つまり public と private 両 version の共存である．この期間の前後では前項の複製が行われることになる．

8) 代替設計

代替設計は興味深いが困難な課題であり，実現は少なくとも上記の 5)〜7) 項が実現されていることを前提とする．

9) 接近法に対する多様な要求

例えば名前による読み取りを含む random access, 接近範囲の局所化（操作の目的に応じて動的に為される必要がある），concurrency 制御（事務処理と違って更新の過程で database の整合性が崩れている期間が極めて長いという事情を踏まえた）等々．

3. 認識の方法——反省と paradigm の選択

CAD の database が実世界の model であり，意味が実世界の意味に依拠するものとすれば，意味を捉える枠組は世界を認識する方法によって導かれる．特定の metamodel に拠る立場はしたがって，認識の特定の方法に commit する立場でもある．metamodel の構成に先立ち，われわれはこれまでに見た他の例について背後にある，多くの場合明示されてはいない立場を推量し，反省し，われわれ自身の方法を選び取らなければならない．

3.1 事物主義の方法

有向 graph の方法は**もの**(事物) としての節の間に成立する関係（**こと**，事実) としての弧や関係づけられる相手方（終端節）を一つの**もの**（始端節）に従属させる，言わば**もの**中心的かつ主語-述語構造的定式化に傾きがちであった．LEAP の三つ組構造も関係づけられる項を attribute, object, value を呼び，関係の成立を $a\cdot o\equiv v$ と書くことにより**もの**へと指向する．実は a や v も o とまったく同等に扱われるという事情は，一方ではそれらが**もの**化する，他方では（独立な存在であるはずの）**もの**が他の**もの**との関係において後者の属性あるいは属性値として位置づけられるという二重の意味で，この指向を支える．LEAP の構造を有向 graph と見る見方があり，実際その程度に抑えて使うのが使いやすいことを先に指摘したが，それは単なる名称や syntax でなく根底にある世界観の然らしめるものである．

様々な変種を通じて変わらない関係 model 的世界の特徴は第1に存在 (entity) は属性の集まりである．第2に存在の型は属性の斉一性によって決定される，第3に属性の内でもある特別な一組，当然最も重要な一組が存在を識別し，他はこれに従属し付随するということであった．Codd はさらに存在と関係の別を，表現形式の同一を理由に否定した．それらは，ただちに属性と他の存在の別の否定に結びつく．両者を区別することから出発する Chen ですらまったく同じ理由で，その区別を同一概念の内なる種差としか見ない．

ここに見られるのは明らかにすべてを単一の概念，ものとしての tuple あるいは record 概念に収斂させるもの主義的世界観である．

世界にはまずものがある，ものは属性あるいは他のものを述語として持つ，ある特別な属性(本質)はそのものであることを決定する，世界はこのような主語-述語関係によって構造化される，もの(事物)主義をこう要約すればそれは，おそらくアリストテレスにさかのぼる一つの強固な思想的伝統であると了解される．だが了解を確かなものとするためにわれわれはまず鍵を握る諸概念を語義的に押え，次いで主として「形而上学」に依ってアリストテレス自身が何を語っているかを見ておきたい．

1) 語　義

鍵となる言葉は database の議論に必ず現れる，本稿でもすでに幾度も出た entity, attribute, relation の三つである．それに「形而上学」において最も中心的な位置を占める ousia (‘実体’) と to ti ēn einai (‘何であるか’, ‘本質’) の英訳 substance と essence を加えた五つについて，以下は現代の代表的な辞書の一つ (Webster's 3rd International Dictionary) による定義の，われわれの当面の関心の限りでの要約である．

entity	1.	independent, separate or self-contained existence
	2.	the existence of something as contrasted with its attributes or properties
	3.	the essence, fundamental nature of real being of something
attribute	1.	a characteristic either essential and intrinsic or accidental and concomitant
	2.	an object closely associated with and thought of as belonging to a specific person, thing or office
	3.	(logic) any quality or characteristic that may be predicated of some subject
	4.	(philos) a necessary or essential quality or characteristic of substance
relation	1.	an aspect or quality that can be predicated of two or more things taken together
	2.	the mode in which one thing or entity stands to another, itself or others
substance	1.	essential nature: essence
	2.	[trans of Gk ousia] something that underlies all outward manifestations: the abiding part of existence or an existing thing as distinguished from what is accidental to it
essence	1.	basic underlying or constituting entity, substance or form
	2.	the permanent as contrasted with the accidental and variable phases or foundation of being
	3.	condition or fact of being or existing: existence considered as a property of a thing
	4.	entity especially an abstract entity
	5.	the/a most significant element, attribute, quality, property or aspect of a being

果たして，これらの語の互いに深く関連し合っていることは一目瞭然である．それぞれの語義がこの関連から析出して来るとさえ言える．だが関連の仕方，したがって語義は複雑，曖昧であり，われわれは取りあえず関連があるという事実を知るだけで満足しておこう．

2) アリストテレス「形而上学」における実体と本質

「形而上学」A 巻 3 章は実体と本質を物事の 4 原因の一に挙げることで始まる；

> (物事の第 1 の原因の) 一つは物事の実体 (ousia) であり本質 (to ti ēn einai) である．蓋しそのものが何のゆえにそうあるかは結局それの説明方式 (logos) に帰せられ……当の何は窮極においてはそれの原因であり原理である．

実体とは何か？

> 二つの意味がある．一つはもはや他の如何なる基体 (主語) の述語ともなり得ない窮極の基体であり，他はこれと指し示され得る，かつ離れて存し得るものである．(Δ 8)

ある意味では質料がそうした基体と言われ，他の意味では型式 (morphē) がまたこれら両者から成るものがそれである．(Z 3)

すべての属性の抽き去られた後には明らかに質料以外には何も残らない．しかし質料がすなわち実体であるという結論は不可能である，なぜなら離れて存することと，これと指し示し得ることが最も主として実体に属すると認められているから (Z 3)．一方普遍は常にある主語の述語となり (Z 13)，また共通的なものとして同時に多くの所に属し，それゆえ諸個物から離れて存在し得ない (Z 16)，それゆえ決して実体ではないとされる．もちろん形相 (morphē, eidos) は普遍と同じではない．しかし個物のそれぞれがそれぞれの形相をもつのでない限り，区別がつけられるのであろうか？ 実際普遍は類，形相，型式と密接に関連し，同じ論拠から実体と考えられている (H 1)．アリストテレスは決して断定はしないが，質料と形相を備えた個物を最も実体の名に値すると考えているようである；

実体はその最も明瞭な形では物体に属すると思われている．(Z 2)
一般に感覚的な事物のあるものが実体であるということに見解が一致している．(Z 3)
個々の事物の実体はその事物に特有であり他の何物にも属さない．(Z 13)
実体というのはその最も優れて第1のまた最も主要な意味では如何なる主語の述語ともならず，如何なる主語の内にもあらぬもの，すなわち個物，例えばこの人，この馬 (「範疇論」5章 2911)

もちろん個物の付帯的 (symbēkos) な属性は実体とは何の関り合いも持たない．付帯的でないとはつまり本質的なということであろうが，それでは本質とは何か？ それは定義であり，そのものの意味であり，自体的な属性である；

ものはある一つのことを意味するはずであった．これがそのものの実体なのであった，そして実体であるというのはこれがそのものの正にそれであるとの意であった．(Γ 4)
そのものがそれ自体において (kath' hauto) その本質である．……ある基体が離れて単独にそれ自体でそのようにあるとき，ただこの基体にのみ属するそのような属性をそれの自体的な属性と言う．(Δ 18)
本質なるものはただそれの説明方式がそれの定義であるものにのみ存する．(Z 4)

より具体的には何を意味するのか？

実体と付帯性はこの点で区別される．すなわち白さは人間に付帯的であると言われるが，それはただ人間がたまたま白くあるというのであって，正に白であるということが人間の人間であるというのではないから．(Γ 4)
人間が……動物であるとは付帯的にではない．(E 2)
(分割を重ねていって出て来る) 最下位の種差一つだけがその当の事物の形相であり実体である．(Z 12)
定義とは……最下位の種差を含む説明方式である．(同)

これまでのところ引用は専ら理論の骨格を浮かび上らせるためのものであった．しかし「形而上学」は示唆に富む多くの例示を含んでおり，それらに目を向けるのは理解に言わば血を通わせるために有益である．例えば

毀損の場合には実体はそのまま存続していなくてはならない．例えば盃は毀損されてもなお盃であるが底に孔が開いたのではもはや毀損されたとは言われない．(Δ 27)
ある人は (その不可能なことに気づかず，太陽を定義しようとして) それが取り除かれても太陽は依然として残り存するであろうような (余分の非本質的な) 属性，例えば「地球の周囲を運行するような」とか「夜分に隠れる所の」とかいう属性を定義に付加するような誤りを犯しているだけでなく (蓋しもし運行しないで静止しており，または隠れないで姿をみせていればもはや太陽ではないということになる．しかし，それは不都合である．なぜならば「太陽」は実在するある実体を意味しているのだから)．さらに彼等はまた他のものにも属し得るものを付加している．ために，例えばある他のものがそのような属性を持つものになれば，明らかにこの他のものもある一つの太陽であるということになろう．(Z 15)

さて以上をわれわれは次のように要約できるであろう：個物を他から分かち正にその

ものたらしめているのは本質と呼ばれる特別な属性である．それは複数の個物を共通に記述するのではなく，このもののみに自体的に属する．それはこのものが属する類に始まってこのものに至る何階層かの種差から成る，このものの定義であり意味である．質料とこのような本質の結合が個物の実体であり，他の一切の属性は付帯的に過ぎず，後者は有ろうが無かろうが，個物の存在，実体，本質には何の影響もない．

これは事物主義的立場の宣明以外の何物でもないではないか？ 'key' 概念の '本質' 概念に何と正確に照応していることか！ もちろんアリストテレス自身はこのような言い方をしてはいない．要約する，しかも決定的な形に要約すること自体がアリストテレスの思索からほど遠いと批判されるかも知れない．本質を属性の一種と見るのは見当違いもはなはだしいと言われるかも知れない．しかし，下に引用する一人の碩学の読み（出隆訳[75] 解説）をどう見るか？

> （アリストテレス）はこれに内在する，これの本質的要素，本質的諸属性，これの内在的原理原因を探究しようとする．……プラトンのしたようにその普遍的な或るものを感覚的な個物の外に求めるのではなしに……それぞれの個物の内に求め……それぞれ最も真に実在するこれなる具体的実体たらしめているところの或る普遍的なもの（個物に内在する普遍的本質，形相的実体）であると答えようとする……．

立場を鮮明にすることを求められるならば，上の要約以外にはあるまい．断定的でないのは事物主義が世界の意味を把捉するに到底足るものでないことに起因する，アリストテレスの動揺でしかないと筆者は考える．そう考える今一つの理由は次項に述べる関係概念の影の薄さである．

3) '関係' 概念の欠落

「形而上学」の '関係' の考察は第1にこの概念への言及そのものが極めて乏しいこと，第2に言及している場合でも関係する一方のものの属性として位置づけていることに特徴がある．曰く

> 数において関係的なものも能力において関係的なものも，すべてこれらが関係的と言われるのは，これらがこれら自らの本質において他のものへの関係を持っているからであって……．(*A* 15)
>
> 限定，運動，関係，配置や割合などは何物かの実体を表示するものとは考えられない……諸性質はそのものの限定であって実体ではない．(*B* 5)
>
> ……ある論では物事の関係にも eidos を措定することになる，しかもわれわれの主張では関係には自体的に存在する類は無いはずなのに．(*A* 9)

関係の eidos を措定するのは言うまでもなくプラトンに見られるのであるが，一方が実体化によって言わば格上げを計り，他方が実体性を否定し，それゆえに自体性を否定するのは，関係が関係である限りものに従属せざるを得ない，言い換えると窮極の認識対象はものでしかあり得ないとする点において共通する．その共通性に限ってわれわれはプラトンを一瞥しておかなければならない．

4) idea＝属性および関係の存在化，あるいは述語の名詞化

idea のものとして現れは，例えば下の引用に顕著である；

> 純粋な美そのもの，善そのもの，大そのもの……がある．……もし美そのもの以外に何か美しいものがあるとすれば，それはかの美に与るからであって外の如何なる原因によるのでもない．(「パイドン」100 b, c)
>
> すべて大きいものは正に「大」によって大きいのであり……小さいものは「小」によって小さい．……(シミアスは)パイドンとソクラテスの中間にあって，一方の「大」に対しては凌駕されるようにと自分の「小」を提出し，もう一方へは自分の「大」を相手の「小」に勝るようにと提出する．(同 101a-102d)

形式的な体裁で言い直せば，最初の例は普通単項述語的に *beautiful* (*A*) と書くとこ

ろを2項述語を用いて *partake* $(A, BEAUTY)$ と表される．2番目は *greaterthan* (A, B) であるがこれを書き替える術を筆者は知らない．一般に個物 A についての表明 $a(A)$ の真実は述語 a の名詞化である idea $I(a)$ との関係 *partake* $(A, I(a))$ の形を取るとされる．さらに，ある場合には二つの idea の間に（*partake* とほとんど同義の）関係 *blend* (I_1, I_2) が成立する（「ソフィスト」251d-259b）．

述語 *blend* はプラトンの世界観において第1級の重要性をを持つと思われる．と言うのは真なる認識は感覚的対象や似像に拠らず idea のみに拠り，idea を織り合わせ，idea から idea へと辿ることによって得られるからである（「ソフィスト」259e，「国家」VI 510b, 511b）．その *blend* が成立する，しないとはどういうことか？ 例えば「存在」は「動」とも「静」とも *blend* する．なぜならばすべて動くものも静なるものも存在するものであるから．自明な理由で「動」と「静」は *blend* しない．不思議なことに「同」と「異」は「存在」と同じくすべての idea と *blend* し，とくに *blend*（「同」，「異」）さえも成立する．蓋し2項述語 a, b については *blend* $(I(a), I(b)) \equiv \exists X \exists Y \exists Z\,(a(X, Y) \wedge b(X, Z))$ なのであろう．（ついでに $a' = \lambda X(\exists Y a(X, Y))$, $I(a) = I(a')$, *blend* $(I(a'), I(b')) \equiv \exists X(a'(X) \wedge b'(X))$ と考えることができる．）*blend* の成立，不成立に基づいて idea の世界が構造化される．その具体的な方法は Cornford[73] によれば類の種差による分割の最下位の種に至るまでの積み重ねである．とすれば（Cornford の懸命の否定にもかかわらず）アリストテレスの方法との類似は明白ではないか．もちろんプラトンが知の対象としては退けた感覚的・物理的個体がアリストラレスに在っては窮極の実体であることは決定的な違いである．しかし idea であろうが個体であろうが自体的に意味を荷う存在を認め，しかもそれが世界の窮極的実在であると考える点に違いはない．その共通性がつまりは事物主義なのであった．

アリストテレスと同じくプラトンは興味深い例示を豊富に含む．その一つは（プラトン自身の説ではないが）いわゆる「夢理論」（「テアイテトス」201e-206b）であって次のように語られる：最も基本的な要素（例えば字母）はただ名付けられるだけでそれ以上何の説明をすることも，したがって知ることもできない．それに対して複合体（例えば音節）は要素から成る成り立ちを記述できる，すなわち定義できるがゆえに可知的である．単純に言えばこの理論は，知り得ないものの集まりから成るものをどうして知り得るのかという理由で退けられた．複合体が単なる集合ではなく構造すなわち要素間の関係を持つこと，その構造こそが対象を可知的ならしめる所以であることをプラトンは見逃した．（Cornford はプラトンが複合体が集合以上のものであることを知らなかったわけではないと述べている．今井[74] によれば大胆にもプラトンが「夢理論」を肯定しているとする説もあるという．またアリストテレスは「形而上学」Z 巻17章で同じ例について議論し，音節 BA は集合 $\{A, B\}$ と違う何物かを持つと言う．その何物かが BA の実体だとしているのはもちろんのことである．）

当然のことながら「夢理論」のその後の発展はない．もしあったとしたら事物主義の伝統はまったく違う歴史を持っていたかも知れない．

3.2 事実主義の paradigm

個体を個体たらしめる，すなわち他の個体から識別する根拠を，質料にせよ普遍にせよ，その個体が関る何らかの事態の内に求めようとするのは，すでにアリストテレスにおいて手に負えない代物であったのだが，「個体化の原理 (principle of individuation)」としてスコラ哲学の重要問題の一であり，B. Russel[77] によればまず Leibnitz により本質と付帯的

な属性の区別が取り除かれ，次いで（例えば Berkeley や Hume を経て）実体概念が放棄されるまで様々な形で居座り続けた．その後には問題はどう述べられるのか？ 再び Russel によれば第1に一つの個体が関るすべての命題が「本質」に取って代る，第2に‘実体’とは当の個体を巡って生起する事態を束ねる言語上の約束に過ぎない．これは事物主義の破産宣告に等しいが，L. Wittgenstein[78] に至って曖昧さをほとんど残さぬ明瞭さを以て述べられることになる；

> 世界は事物の寄せ集めであって，物の寄せ集めではない．(1.1)
> 事実とは幾つかの事態の成立に外ならない．(2)
> 事態は対象(事物)の結合である．(2.01)
> 事態の構成要素となり得るということは物にとって本質的である．(2.011)
> 言うなれば対象は無色である．(2.0232)
> 同一の論理的形式を持つ二つの対象は，外的な性質を無視すれば，ただそれらが異なっているということによってのみ，互いに区別される．(2.0233)

個体化原理はついに識別されてあることという極限の一点に行き着いた．個体は文(命題)の名詞の位置に開けられた孔のようなもので，それ自体は何の意味も持たない．意味は同じ孔が現れるすべての文から言わば析出して来る．ここに主張されているのは事物主義に対して事実（こと）主義と呼ぶべき，ほとんどコペルニクス的に転換された立場である．

今日われわれは事実主義的世界観の現れを様々な知の分野に見出すことができるが，その一例は次に示されるごとくである；

> 精神は表示体系の中というよりはむしろ，一方には種 x と種 y との間，他方には氏族 a と b との間に存在する示差的格差の間に相同性を要求する．……類似しているのは類似点ではなくて相違点なのだ[79]．
> 事物自体よりはむしろ事物間の関係に注意を向けてこそ何らかの成果を期待し得る[80]．

こうして見るとソクラテスの語った「夢理論」は，実は事実主義に近いものであったことが分る．非常に興味深いのは3世紀余り後に成立した仏典「ミリンダ王の問い」が，さらに急進的な実体不在説を説いていることである．例えば，曰く

> 「ナーガセーナ」と申す者は……様態・感受・知覚・表象・認識に依存し（た相対関係の下に）て始めて，呼称・標徴・記号表出・言語的通念・名のみのものとして成立する一方，絶対的次元においては……人格的実体が存在するものとは認められない．

事実主義は思潮として支配的でも主要でもなかったが，事物主義と同じくらい長い歴史を持っているのである．しかし決定的な現れは，数学の基礎を巡って前世紀以来の大いなる希望と深刻な危機の交錯する問題的状況，すなわち非ユークリッド幾何学の発見，K. Weierstrass による解析学の合理化，集合論の瞠目すべき成果と数々の paradox との遭遇，G. Frege や B. Russel による論理学の現代化，等々のもたらした状況の下に成立した D. Hilbert の公理主義ないし形式主義的数学であろう．「点，直線，平面と言う代わりに卓，椅子，酒盃と言って幾何学が構成できなくてはならない．」という言明に接するとき，われわれは「点とは部分を持たないものである．」と定義するユークリッドの世界とはまったく異質な世界に移り住んだことを知る．

さて，われわれの選択すべき paradigm は，今や明らかとなった．公理主義に随うことは metamodel に形式の洗練と意味の明晰をもたらすが，加えて一定の数学的正統性を与える．この最後の点は，database が数学的 model であることと合わせて，metamodel 論を T. S. Kuhn[82] の言う‘normal science’に定位するものと期待されるのである．

4. metamodel の構成

4.1 基礎概念

database は以下のように構成された1階形式理論 (S) の model (M) である：

1) 対象領域＝識別子の集合(I) ∪ 既知領域(A) ∪ 名前の集合(N)
A＝親言語の許す data 型全体の和

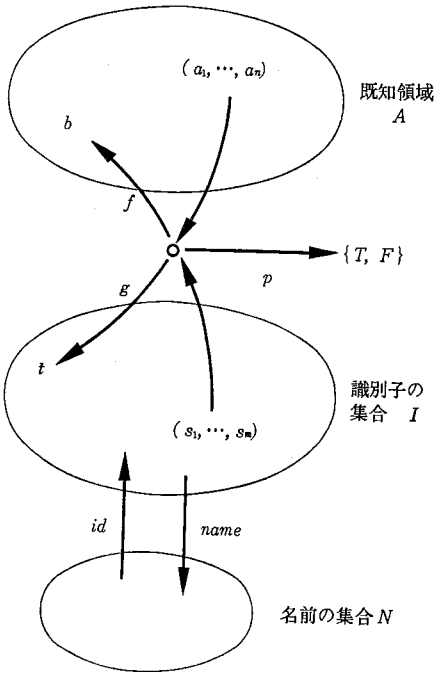


図 5 M の構成
Fig. 5 The organization of the model (M)

1.1) 実世界の事物に対応する対象(存在, entity) は自体的には識別される働きだけを持つ記号的存在であるから，われわれはこれらを端的に識別子として定義する．

1.2) あらゆる数学的科学と同じく，S や M に関する理論は一定の既知領域 (A) を前提し，その上に建てられる．いわゆる属性は A の内に値を持つ．

1.3) 自己完結的な検索言語は CAD の要求する複雑で多様な data 処理に到底応えられないから，何らかの親言語を想定することになる．領域 A はしたがって親言語の data 型全体の和とするのが自然である．

1.4) 名前は文字列としては A に属し，したがって N は A の一部であるが，下に述べる特異な機能を要求されるところから，便宜的に独自の領域として取り出しておく．

2) (部分) 関数

I または A の上で定義された I または A の内に値を持ついくつかの関数，すなわち $f: I^m \times A^n \to I$, $g: I^m \times A^n \to A$ など．特に $name: I \to N$, $id = name^{-1}$ を組み込み関数として設定する．また存在のいわゆる属性は $I \to A$ なる関数として(望むならば) 定義することができる．

3) 述　語

I または A の上で定義されたいくつかの述語，すなわち $p: I^m \times A^n \to \{T, F\}$ た

だし値 T の場合のみを明示し，それ以外は自動的に F となるといういわゆる closed world assumption を要請する．

4) 公　　理

(恒真でない) いくつかの論理式の真なることを要求する．これらは整合性条件に外ならないが，注意しなければならないのは，冒頭に書いたように database を形式理論でなく model すなわち interpretation と捉えているから，公理に矛盾しない論理式が必ずしも M において真ではないことである．例えば *intersect* (l_1, l_2, p) を「平面直線 l_1, l_2 が点 p で交わる」こととするとき，平行でない l_1, l_2 について $\exists p$(*intersect* (l_1, l_2, p)) を真ならしめるには，実際そのような p を M に創成しなければならない．

4.2 幾何学との analogy

D. Hilbert[82] には，幾何学を構成する二つの方法が示されている．第1は本文の連続公理に先立つ部分で，そこではすべての定義と公理は未定義の幾何学固有の概念 (‘点’, ‘線’, ‘結合する’, ‘の上にある’ 等々) のみを用いて述べられ，したがって形式理論的な記述に直接翻訳できる．連続公理の記述はしかし quantify された自然数を含んでいるので，そうはいかない．第2は付録 IV の方法で，第1のとは逆に連続性を始めから要請するために，直交座標系を持つ数空間の体系を前提し，その体系内の諸概念と諸定理を既知として記述の素材に加える．われわれの場合は上の注意からも明らかなように，後者の方法に沿うのが適切である．すなわち形式理論 (S) の記述に既知領域 (A) の要素とそれについての知識，および集合論的概念を自由に用いることを許す．

4.3 schema 記述言語

schema とは形式理論 (S) を指す．以下は，schema 記述言語を精密に定式化するための素描である．

1) 基礎概念 (記述の素材)

1.1) 対象を指示する記号，すなわち定数記号 $(x, y, \cdots)$，および変数記号 $(?x, ?y\cdots)$.

1.2) 値が (data 操作の段階で) 応用 program により直接与えられる関数および述語の記号 (名前)，引数および値の型．領域 I の要素の型とは I で定義された単項述語の外延である．

1.3) 関数記号 *name*, *id* および $=$ (等号).

1.4) 領域 A で許される記号.

1.5) 論理記号 $\wedge$, $\vee$, $\neg$, —, $\rightarrow$, $\exists$, $\forall$.

1.6) 集合定義の記号 **set**{　} および集合関数記号，特に基数を表す *card*.

2) 関数，述語および集合の定義

2.1) 許される論理式は，外延が当の論理式に現われる原子論理式の外延によって決定される (すなわち union compatible なものの和を取り，次に natural join を取り，さらに自由変数への射影によって得られる) 集合に含まれるものに限る．例えば

$p(x)\wedge q(y)$ や $\forall x(p(x)\rightarrow q(x, y))$ は許されるが，$p(x)\vee q(y)$ や $p(x)\rightarrow q(x, y)$ は許されない．

2.2) 集合の定義 **set**$\{x1, \cdots xn : r(x1, \cdots xn)\}$ ここで r は任意の (許される) 論理式.

2.3) 関数および述語は 1.2)—1.6) を用いて一般に再帰的に定義される.

2.4) (許される) 論理式のいくつかを公理として掲げる.

3) 例

3.1) 有向 graph において弧 p によって (直接または間接に) 連結されている (x, y) を

特徴づける述語 *conn-by-p* の定義.

predicate *p*(*type* 1, *type* 2) ;

formula *conn-by-p*(?*x* : *type* 1, ?*y* : *type* 2)=*p*(?*x*, ?*y*)∨
∃?*z*(*conn-by-p*(?*x*, ?*z*)∧*p*(?*z*, ?*y*))

3. 2) 多角形 *pg* と点 *pt* の間の距離を与える関数の定義. *pg* と辺 *e* は 2 項関係 *pge*(*pg*, *e*) によって, *e* と端点 *v*1, *v*2は 3 項関係 *lspt*(*e*, *v*1, *v*2) によって関係づけられ, *v* および *pt* は関数 (属性) *coord* によって値 (*x* : *real*, *y* : *real*) を持つとする.

table *coord*(*point*)=**record** *x*, *y* : *real* ;

function *coord-ls*(?*ls* : *line-seg*)
=**record** *orig* : *coord*(ε?*pt*1(∃?*pt*2(*lspt*(?*ls*, ?*pt*1, ?*pt*2)))),
dest : *coord*(ε?*pt*2(∃?*pt*1(*lspt*(?*ls*, ?*pt* 1, ?*pt*2)))) ;

function *dist-lspt*(?*ls* : *line-seg*, ?*pt* : *point*) : *real* ;

type *vector*=**record** *x*, *y* : *real* ;

var *v*10, *v*20, *v*12 : *vector*, *a*1, *a*2 : *real* ;

begin *v*10:=*vector*(*coord*(?*pt*). *x*−*coord-ls*(?(*ls*). *orig*. *x*,
coord(?*pt*). *y*−*coord-ls*(?*ls*). *orig*. *y*) ;
*v*20:=…… ; *v*12:=…… ;
*a*1:=*inner-prod*(*v*10, *v*12) ; *a*2:=*inner-prod*(*v*20, *v*12) ;
if *a*1≦0∧*a*≦0 **then** *dist-lspt*:=*length*(*v*10)
else if *a*1≧0∧*aa*2≧0 **then** *dist-lspt*:=*length*(*v*20)
else *dist-lspt*:=*length*(*v*10)∗
sqrt(1−(*a*1/(*length*(*v*10)∗*length*(*v*12)))∗∗2)

end

function *dist-pgpt*(?*pg* : *polygon*, ?*pt* : *point*)
=*min*(**set** {?*d* : ∃?*e*(?*d*=*dist-lspt*(?*e*, ?*pt*)∧*pge*(?*pg*, ?*e*))}) ;

3. 3) 曲面 patch (*s*) と頂点 (*v*) が関係 *sv*(*v*, *s*) で結ばれているとき, 各 *s* について全頂点を覆う, なるべく半径の小さい球を値とする関数の定義.

function *sph-clos-sp*(?*s* : *surf-patch*)=*sph-clos-pts*(**set**{(?*x*, ?*y*, ?*z*) :
∃?*v*((?*x*, ?*y*, ?*z*)=*coord*(?*v*)∧*sv*(?*s*, ?*v*))}) ;

type *sphere*=**record** *r* :*real*, *center* : *coord-pt* ;
coord-pt=**record** *x*, *y*, *z* : *real* ; *point-set*=**set of** *coord-pt* ;

function *sph-clos-pts*(?*pts* : *point-set*) : *sphere* ;
{*function body to calculate the spherical closure containing* ?*pts*}

3. 4) 与えられた平面 (*pl*) と上で定義した球が交わるような曲面 patch を特徴づける論理式の定義.

formula *intersect* (?*pl* : *plane*, ?*s* : *surf-patch*)
=(*dist-plpt*(?*pl*, *sph-clos-sp*(?*s*). *center*)≦*sph-clos-sp*(?*s*). *r*)

4.4 data 操作言語

data 操作とは形式理論である schema の model としての database を構築, あるいはそこから情報を抽出する行為の謂である. 以下も前節と同じく formal な言語を設計するための素描である.

1) 基本機能

基本的な data 操作は対象，特に識別子の創成，削除，領域の構造の変更および論理式の評価（真理値ないし外延の決定）である:

1.1) 識別子の創成

create $x_1, \cdots, x_n$ **making-true** $formula(x_1, \cdots x_n)$

[意味] 1 $I_1 := I \cup \{s_1, \cdots, s_n\}$

2 $x_1^1 := s_1, \cdots, x_n^1 := s_n$

3 **make-true** $formula(x_1, \cdots, x_n)$

1.2) 識別子の削除

delete $x_1, \cdots, x_n, y_1, \cdots, y_m$ **evaluating** $formula(?x_1, \cdots, ?x_n)$

[意味] 1 **evaluate** $formula(?x_1, \cdots, ?x_n)$

2 $I_1 := I - \{x_1, \cdots, x_n, y_1, \cdots, y_m\}$

1.3) 構造の変更

make-true $formula(x_1, \cdots, x_n)$

[意味] 1 $formula(x_1, \cdots, x_n) := T$

1.4) 論理式の評価

evaluate $formula(?x_1, \cdots, ?x_n)$

[意味] 1 ($n \geqq 1$ の場合) $formula(d_1^{(j)}, \cdots, d_n^{(j)}) = T$ なる $d_i^{(j)} \in A \cup I \cup N$ について $x_1^{(j)} := d_1^{(j)}, \cdots, x_n^{(j)} := d_n^{(j)}$

2 ($n=0$ の場合) $formula$ の値 (T または F) を戻す.

2) 例

2.1) 多角形 pg と点 pt の間の距離 d および d を与える辺を求める.

evaluate $(?d = dist\text{-}pgpt(pg, pt)) \wedge (?d = dist\text{-}lspt(?e, pt)) \wedge pge(pg, ?e)$

2.2) 与えられた頂点 $(v(1), \cdots, v(n))$ と方程式の係数の配列 a を持つ曲面 patch s を創成し（1），それを含む球を計算し，改めて属性 *neighb* として付与し（2），そのような patch がいくつかあるとき，与えられた平面と球が交わるものを検索する（3）.

2.2.1) **create** s, $(i=1, n)v(i)$ **making-true** $((i=1, n) \wedge sv(s, v(i))) \wedge ((i=1, n) \wedge point(v(i))) \wedge surf\text{-}patch(s) \wedge (coeff(s) = a)$;

2.2.2) **make-true** $neighb(s) = sph\text{-}clos\text{-}sp(s)$

2.2.3) **evaluate** $dist(pl, neighb(?s).\ center) \leqq neighb(?s).\ r$

2.3) 部品 $(?p)$ と供給者 $(?v)$ の間に関係 $supply(?v, ?p)$ が成立しているとき，唯一の供給者しか存在しないような組 $(?v, ?p)$ を検索する.

evaluate $supply(?x1, ?x2) \wedge \forall ?v(supply(?v, ?x2) \rightarrow ?v = ?x)$ または

evaluate $supply(?x1, ?x2) \wedge (card(\mathbf{set}\{?v : supply(?v, ?x2)\} = 1)$

4.5 現行 modelling との比較

まず1章に指摘した3種の困難は，以下のように解決ないし緩和されている．第1に metamodel 性は，定式化の抽象性からして十分であると言える．第2に1階述語論理と帰納的述語による枠組は，理論的（超数学的）にも経験的（数学的）にも確かな根拠を持っている．また，考えられる最も広汎なとは言えないが，少なくとも現行のどの model よりもはるかに大きな記述能力と見通しの良い構造を提供する．しかも現状は，その可能性を十分開拓したと言うにはなお程遠い．このことに関連して，われわれの metamodel が特別な場合として現行の model を含むことに注意しておこう．例えば，Codd の model は I の互に素な集合 (relation) への分割，それぞれの集合から特定の record 型への関数と親言

語の data 間の等号および不等号による構造化である．第3に主として関係 model に見られた意味論的困難は，二つの方向で解決が計られている．その一は困難の主要な原因であった多義的な概念，すなわち属性の集まりとしての存在 (record または tuple)，属性の斉一性による存在の型 (record 型または relation) および key の廃棄である．これによって存在，属性，関係の別を明確にした．その二は存在を徹底的に記号化し——完璧を期するために記号（識別子）は dbms の管理下に置かれる——意味はそれが与る事態，すなわち関数および述語関係の総体であるとすることによって‘意味’の意味を確立した．

2章に挙げた要求の内 1)，2)，4)，5) は概ね，3) は部分的に満たされている．3) が部分的にと言うのは例えば述語や関数を動的に定義したり，それらの名を変数として扱うことができないからである．9) の内名前による接近は可能となった．6)〜9) は基礎概念の水準を超える複合的機能の問題であり，また privacy や security など database の (model としてでない) 管理的側面にも関係するので単純に論じられない．今はわれわれの metamodel がこれらの実現を妨げる如何なる要因も含まないことを指摘するに留める．ただし 6) の複製機能については 5.5 節に一つの実現方法を例示する．

5. 実用 system (LMS) の開発

前章に述べた metamodel の完成には，多くの技術的課題を克服しなければならない．就中正確な言語仕様の決定，論理式の評価手続きの最適化および application の特性に適合する内部 schema の記述方式の設計は困難が予想される．一方（大規模で多目的的な環境での使用を含む）実用に供するために，LMS (Logical Structure Manipulation) と呼ぶ dbms が開発された．LMS は次の意味で前章の metamodel の部分的実現として位置づけられる（図 6)；

1) 根底において世界観を共有する．したがって対象領域の構成，述語論理による構造化は基本的に変わりない．ただし以下のような制約を受けている．
2) 関数は *name*, *id* それに I から A への1変数関数が各識別子に対して（同時には）1種類のみ許される．その値を‘要素’または‘値’と称する．事実上それは属

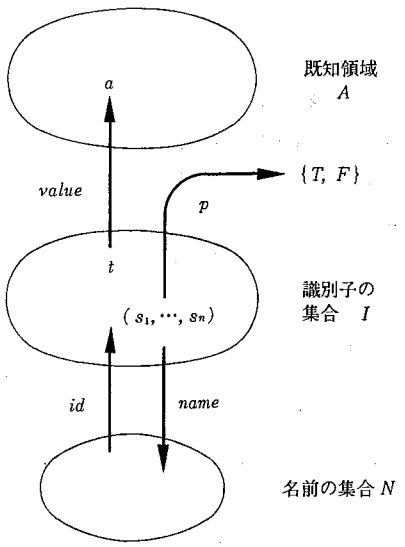


図 6 LMS の model

Fig. 6 The organization of the LSM model

性値の集まりとしての record であり，関数の種類は record 型であるので，この制約は LMS の従来の record 指向型 dbms に対する妥協を示している．

3) 述語は I の上の任意項数のものを許す．

4) 論理式の構成において term として使えるのは識別子を指示する記号である．‘値’は許されない．これは LMS が対象の‘属性’を‘見ない’すなわち述語による I の明示的・形式的な構造の扱いに専念する，‘属性’の扱いは応用 program に委ねることを意味する．‘logical structure manipulation’と呼ぶ所以である．

5.1 dbms としての LMS

管理的側面は本稿の主題ではないが，modelling に直接 関係する限りで述べると；

1) (識別子，要素)の組の集合としての database は互いに素な幾つかの workunit(wu)に，1個の wu はこれも互いに素な section に分割される．1個の応用 program は1個の wu 内の section への書き込みと任意個数の wu 内の section からの読み取りを許される．読み，書きいずれにしても，open された section のみが操作の対象になる．(接近範囲の局所化)

2) database はまた area にも分割される．1個の section は物理的には1個の area 内の連続した page の集合として定義される．

3) 応用 program と database の間で受け渡しされる情報の主たるものは識別子，要素，要素の型，wu 名，名前で，受け渡しは dda(data delivery area) を介して行われる．加えて制御のための情報が ca(communication area) に与えられる．

4) dbms は次の subsystem と物理的管理を荷う CODASYL DMS とから成る．

4.1) LDL (data description language) : schema を記述する．

4.2) LML (manipulation language) : FORTRAN を親言語とする data 操作言語．

4.3) LMR (manipulation routine) : data 操作を実行する program 群．

4.4) LMU (manipulation utility) : 応用program の水準の utility program 群．

4.5) LRU (reorganization utility) : 物理的管理のための utility program 群．

この内で modelling に関る LDL, LML および LMU について以下に概要を述べる．

5.2 schema 記述言語 (LDL)

1) **area** および **section** の定義

area 〈area 名〉(〈area code〉, 〈頁数〉), 〈頁当たり語数〉, 〈頁当たり record 数〉

section 〈section 名〉(〈始頁〉, 〈終頁〉), …, 〈section 名〉(〈始頁〉, 〈終頁〉)

2) 述語の登録

predicate 〈述語名〉(〈項数〉), …, 〈述語名〉(〈項数〉)

5.3 data 操作言語 (LML)

1) 論理式の定義

formula 〈論理式名〉(〈変数名〉, …, 〈変数名〉)=〈論理式〉

[説明] ① 変数名は必ず ？ で始まることによって定数項と区別される．

② ここで定義した論理式名は LML 文中で述語名と同様に使える．

③ 次のような形の再帰的な定義が許される；

$$r(?)=p(?)\wedge\exists ?1\cdots\exists ?n(r(?1)\wedge\cdots\wedge r(?n)\wedge q(?1,\cdots,?n,?))$$

2) 許される論理式は (〈項〉=〈項〉 を含む) 原子論理式または既定義の論理式から $\wedge$, $\vee$, —, $\rightarrow$, $\neg$, $\exists$, $\forall$ を用いて組み立てられる．ただし $\forall$ と $\rightarrow$ は次の形のみが

許される；

$$\forall ?1 \cdots \forall ?n(p(?1, \cdots, ?n) \rightarrow q(?1, \cdots, ?n, ?x1, \cdots, ?xm)$$

また¬は構造の変更において，原子論理式を偽ならしめるためにのみ使える．定数項は識別子を値に持つ program 変数型 (*genid*) として宣言しておかねばならない．

3) 識別子および要素の創成

create 〈dda 名〉[:〈構造変更式〉][:〈名前の付与〉]

[説明] 1 応用 program が所定の形式で用意した dda の値を用いて要素が創成され，識別子が同じ dda に戻される．

2 optional な構造の変更と名前の付与の部分で，この指令によって創られつつある要素の識別子を，dda に並べられた順に従って ＊(1),, ＊(2),… と参照することができる．

4) 識別子および要素の創成および埋め込み

imbed : 〈series 名〉(**initial**) : 〈range 決定式〉

imbed 〈dda 名〉:〈series 名〉: ＊(1). **to.** 〈項〉, …, ＊(*n*). **to.** 〈項〉[/〈要素選択式〉][:〈構造変更式〉][:〈名前の付与〉]

imbed : 〈series 名〉(**final**)

[説明] 1 創成した要素を既存の集合 (range) の選択された要素(項)に対応させることにより，述語構造的に埋め込む．中央の **imbed** は一般に複数個あり，一連の series を形成する．

2 series の最初に range を決定する 1 自由変数の論理式を与える．

5) 識別子および要素の削除

delete 〈dda 名〉: 〈要素選択式〉

[説明] 1 1 自由変数の要素選択式を満たす要素を削除する．

6) 要素の変更

modify 〈dda 名〉: 〈要素選択式〉[:〈構造変更式〉][:〈名前の付与〉]

[説明] 1 dda に用意した値によって，選択された要素を変更する．

7) 名前の付与

name 〈項〉=〈名前〉, …, 〈項〉=〈名前〉[/〈要素選択式〉]

[説明] 1 選択された要素(項)に名前を与える．〈名前〉は *genname* 型として宣言されている program 変数名．

2 〈名前〉を **denamed** とすれば付いている名前を削除する．

8) 要素の検索

get 〈dda 名〉:: 〈要素選択式〉 または **get** 〈series 名〉(**initial**) : 〈要素選択式〉

get 〈dda 名〉: 〈series 名〉

get 〈series 名〉(**final**)

9) 構造の変更

modify str 〈構造変更式〉

[説明] 1 構造変更式と真ならしめるように，原子論理式の値を変更する．

10) 論理式の評価

get tval 〈dda 名〉: 〈論理式〉 または

get pred 〈dda 名〉: (〈n 項組〉); …; (〈n 項組〉)

[説明] 1 **tval** の option では，論理式の真理値を

② **pred** では，すべての n 項組が満たす述語名のすべてを dda に戻す．

5.4 utility program 群 (LMU)

LMU の主な機能はwuの登録と削除，section の初期化，wu への割り付けとそこからの解除，wu 間の data の複製，compaction，archival storage への格納とそこからの取り出し，統計的情報の抽出などである．複製について付言すると，複製された data はまったく別の識別子を持つが，要素としての値，述語構造は同じものが作られ，名前は qualifier のみを指定されたものに付け替える．複製は 1 個の wu から他の 1 個の wu に，一般に複数個の section を 1 個の section の中へと移す．望むならば応用 program を途中に介在させて不要な複製を抑えたり，要素の値を変更することができる．

5.5 例（読みやすくするために，FORTRAN の代りに PASCAL 的表現を用いた．）

1) 1.3 節の 1.4) 項で取り上げた例題．

① ないし ③ に対しては **get** *dda*:: $\forall ?x(p(?x) \rightarrow q(?x, ?y))$

④ は **get** *dda*:: $\forall ?y(\forall ?x(p(?x) \rightarrow q(?x, ?y)) \rightarrow r(?y, ?z))$ と書ける．

2) 曲面 patch が格子状に張られているとき，1 点 *pt* とそこを通る平面 *pl* を与えて断面線を求める．

```
var c, s: genid; ssed, ssing: set of genid;
{determine the curve c on which pt lies.}
get dda:: cbdrys(c, ?s) ; ssed:=φ ;
{move id of surf-patches containing c to ssing.}
foreach s in ssing do
begin if inters(s, pl)=T then
    begin calint(s, pl) ; ssed:==ssed∪[s] ;
        get dda:: ∃?c(sbdryc(s, ?c)∧cbdrys(?c, ?s))
                  −ssed ;  {move id of ?s to ssing}
    end ;
      ssing:=ssing−[s]
end
```

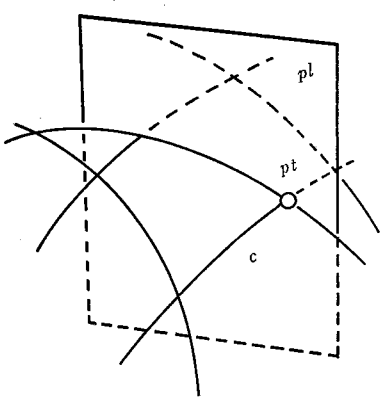


図 7 断面線の計算

Fig. 7 The intersection of a surface and plane

3) 部品 x が部品 y を使っている，その際変換 t を施すという関係を $use(x, y, t)$ とする．今部品 a から出発し *use* しているものすべてを検索し，*use* しているものがさ

れているものに先行するように整列する.

formula $f(?)=(?=id(a))\vee\exists?x(f(?x)\wedge\exists?t(use(?x, ?, ?t)))$;

get *dda*:: $f(?1)\wedge\exists?t(use(?1, ?2, ?t))$; {*sort* $\{(?1(i), ?2(i)): 1\leqslant i\leqslant n\}$ *so that the tuple of the form* (y, z) *never precedes those of the form* $(x, y.)$}

4) 上の例で *a* が直接または間接的に使っている部品全体 *assy* を *aprime* に使われる部品の集合として複製する（図 8）.

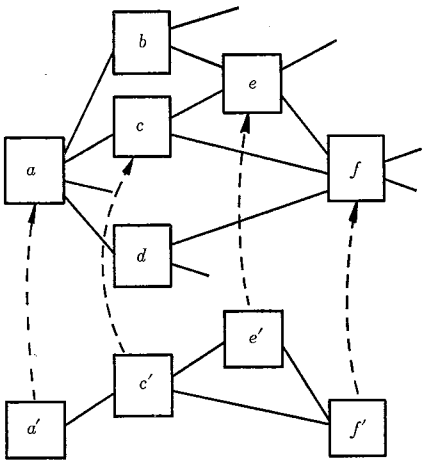


図 8 埋め込み写像

Fig. 8 An imbedding function

formula $assy(?)=\exists?1(((?1=id(a))\vee assy(?1))\wedge$
$\exists?2(use(?1, ?, ?2)\vee use(?1, ?2, ?)))$;

get *in*(**initial**): $(?=id(a))$; *assy* {*initiates reading 'assy'.*} ;

imbed *out*(**initial**): **set**(*in*) {*initiates creating a set of elements and imbedding it in* **set**(*in*), *the range of series 'in', that is, 'assy'.*} ;

get *w*0: *in* {*reads 'a' into w*0 *placing its id in oldid*(0).} ;

imbed *w*0: *out*: (∗. **to.** *oldid*(0)): **name** ∗=*aprime* {*creates 'aprime'.*} ;
{*following are made available to this program with each* **get** *statement* ;
n=the number of elements being delivered with the curreent **get**
nt=the total number of elements in the range 'assy'
np=the number of elements already delivered.}

while $np\leqslant nt$ **do**

begen get *w*: *in* {*reads 'assy' into w placing id's in array oldid.*} ;
imbed *w*: *out*: $(i=1, n)$ (∗(*i*). **to.** *oldid*(*i*)) {*maps the i-th created element to oldid*(*i*) *with the imbedding function.*}

end ;

imbed *out*(**final**) ;

get *in*(**final**) ;

5.6 実務への適用

初版（79年10月）以来今日までの適用は金型（構造部および die face），船舶，自動車，電気器具の形状，機械部品，建築物など多様な対象の設計ないし製図を含む．metamodel 性はしたがって実証された．また多数の設計者による同時的な，しかし多目的的な使用という負荷の大きい環境に耐え得ることも実証された．最大の関心事の一つである効率は処理単位をできるだけ大きく取り，処理に必要な情報を少ない操作で一挙に検索するときに好結果が得られる．そうすることは世界を巨視的な対象の水準において見通し，かつ対象を関係する対象の構造づけられた総体として見ることに外ならない．

参考文献

1960年代

[1] J.C. Gray, "Compound Data Structure for Computer Aided Design: A Survey" *Proc. ACM National Meeting*, 1967.

[2] 吉川康一, "コンピュータ・グラフィックスにおけるデータ構造の問題", 情報処理, Vol. 11, No. 9, 1970.

[3] R. Williams, "A Survey of Data Structures for Computer Graphic Systems", *Computing Surveys*. Vol. 3, No. 1, March, 1971.

[4] D.L. Childs, "Description of a Set-theoretic Data Stracture", *Proc. FJCC*, 1968.

[5] C.A. Lang, J.C. Gray, "ASP-A Ring Implemented Associative Structure Package," *CACM*, Vol. 11, No. 8, Aug., 1968.

[6] J.A. Feldman, P.D. Rovner, "An ALGOL-Based Associative Language", *CACM*, Vol. 12, No. 8, Aug., 1969.

1970年代以降の CAC data model

[7] D.R. Warn, "VDAM-A Virtual Data Access Manager for Computer Aided Design", *Proc. Workshop on Databases for Interactive Design ACM*, 1975.

[8] B.G. Baumgart, "A Polyhedron Representation for Computer Vision," *Proc. NCC*, 1975.

[9] C.M. Eastman, et al, "A Database for Designing Large Physical Systems" *Proc. NCC*, 1975.

[10] C.M. Eastman, "The Concise Structuring of Geometric Data for Computer Aided Design", *Data Structures, Computer Graphics and Pattern Recognition*, Academic Press, 1977.

[11] C. Cavagna, U. Cugini, "Data-Structure for the Description and Handling of Engineering Drawing", *Computer Aided Design*, Vol. 9, No. 1, Jan., 1977.

[12] M.E. Newell, D.C. Evans, "Modelling by Computer", *CAD Systems*, North-Holland, 1977.

[13] C.A. Linden, "Grammars Which Describe Large Bodies of Data", *Computer Aided Design*, Vol. 10, No. 1, Jan., 1978.

[14] I.C. Braid, "On Storing and Changing Shape Information", *SIGGRAPH Computer Graphics*, Aug., 1978.

[15] R.J. Athay, "Object Models for Computer Aided Design", *SIGGRAPH Computer Graphics*, Aug. 1978.

[16] C.M. Eastman, "Systems Facilities for CAD Databases", *Proc. 17th Design Automation Conf.* 1980.

[17] J. Foisseau, et al, "On CAD Data Modelling Through Mechanical Engineering Design", *Proc. Int'l Conf. Interactive Techniques in CAD*, Bologna '78.

[18] J. Foisseau, F.R. Vallette, "A Computer Aided Design Data Model: FLOREAL", *Preprint IFIP Working Conf. on CAD Data Bases*, Sep., 1981.

[19] N. Baron, et al, "An Approach to the Integration of Geometrical Capabilities into a Data Base for CA DApplications", *Preprint IFIP Working Conf. on CAD Data Bases*, Sep. 1981.

[20] R. Daßler, et al, "Databases for Geometric Modelling and Their Application", *Preprint IPIP Working Conf. on CAD Data Bases*, Sep., 1981.

[21] K.A. Roberts, et al, "A Vertically Organized Computer-Aided Design Data Base", *Proc., 18th Design Automation Conf.*, 1981.

[22] M. Lacroix, A. Pirotte, "Data Structures for CAD Object Descriptions," *Proc. 18th Design Automation* Cont., 1981.

[23] O. A. Daini, "Numeical Database Management System: A Model; *Proc. Int'l Conf. on Management of Data*, June, 1982.

CODASYL Model とその適用

[24] CODASYL Data Base Task Group (ACM), April, 1971 Report.

[25] R. W. Taylor, R. L. Frank, "CODASYL Data-Base Management Systems," *Computing Surveys*, Vol. 8, No. 1, March, 1976.

[26] A. J. Korenjak, A. H. Teger, "An Integrated CAD Data Base System", *Proc. 12th Design Automation Conf.*, 1975.

[27] A. E. Bandurski, D. K. Jefferson, "Enhancements to the DBTG Model for Computer-Aided Ship Design" *Proc. Workshop on Databases for Interactive Design ACM*, 1975.

[28] G. Zintl, "A CODASYL CAD Data Base System", *Proc. 18th Design Automation Conf.*, 1981.

[29] S. Ulfsby et al, "TORNADO: A DBMS for CAD/CAM Systems", *Computer Aided Design*, Vol. 13, No. 4, July, 1981.

Relational Model とその適用

[30] E. F. Codd, "A Relational Model of Data for Large Shared Data Banks," *CACM* 13, No. 6, June, 1970.

[31] E. F. Codd, "A Data Base Sublanguage Founded on the Relational Calculus," *Proc. 1971 ACM-SIGFIDET Workshop on Data Description, Access and Control.*

[32] E. F. Codd, "Further Normalization of the Data Base Relational Model", *Courant Computer Science Symposia 6 Data Base Systems*, Prentica-Hall, 1971.

[33] E. F. Codd, "Relational Completeness of Data Base Sublanguages", *Courant Computer Science Symposiab Data Bass Systems* Prentice-Hall, 1971.

[34] D. D. Chamberlin, "Relational Data-Base Management Systems", *Computing Survevs* Vol. 8, No. 1, March, 1976.

[35] H. A. Schmidt, J. R. Swenson, "On the Semantics of the Relational Data Model", *Proc. ACM SIGMOD*, 1975.

[36] P. P. Chen, "The Entity-Relationship Model-Toward a Unified View of Data", *ACM TODS*, vol. 1, No. 1, March, 1976.

[37] J. M. Smith, D. C. P Smith, "Database Abstractions: Aggregation", *CACM*, Vol. 20, No. 6, June, 1977.

[38] J. M. Smith, D. C. P Smith "Database Abstractions: Aggregation and Generalization". *ACM TODS* Vol. 2, No. 2, June, 1977.

[39] C. Beeri, P. A. Bernstein, "Computational Problems Related to the Design of Normal Form Relational Schemas", *ACM TODS* Vol. 4, No. 1, March, 1979.

[40] R. Williams, "On the Application of Relational Data Structures in Computer Graphics", *Information Processing 74*, North-Holland.

[41] R. Williams, et al, "Data Structures in Computer Graphics", *Proc. Workshop on Database for Interactve Design ACM*, 1975.

[42] D. Weller, R. Williams, "Graphics and Relational Data Base Support for Problem Solving", *SIGGRAPH Computer Graphic*, Oct, 1976.

[43] 宇野 栄, 宇土正信, "対話型アプリケーションにおけるリレーショナル・モデルの応用例", 情報処理, Vol. 17, No. 10, Oct., 1976.

[44] G. Valle, "Relational Data Handling Techniques in Computer Aided Design Procedures", *CAD System*, North-Holland, 1977.

[45] R. Phillips, at al, "AESOP: An Architectual Relational Database", *Computer Aided Design*, Vol. 11, No. 4, July, 1979.

[46] M. N. Haynie, "The Relational/Network Hybrid Data Model for Design Automation. Databases", *Proc., 18th Design Automation Workshop*, 1981.

[47] H. Grabowski, M. Eigner, "Employing a Relational Data Structure in a CAD System", *Proc. International Conf. on Interactive Techniques in Computer Aided Design*, Bologna 1978.

[48] H. Grabowski, M. Eigner, "Semantic Datamodel Requirements and Realization with a Relational Data Structure", *Computer Aided Design* Vol. 11, No. 3, Mar., 1979.

[49] H. Grabowski, M. Eigner, "A Data Model for a Design Data Base", *Preprint IFIP Working Conf. on CAD Data Bases*, Spe., 1981.

[50] R. A. Lorie, "Issues in Databases for Design Applications", *Preprint IFIP Working Conf. on CAD Data Bases*, Sep., 1981.

[51] R. Haskin, R. L./Lorie, R. A. "On Extending the Functions of a Relational Database System", *Proc. Int'l Conf. on Management of Data*, June, 1982.

知識工学への接近

[52] Ed. H. Gallaire, J., Minker, *Logic and Data Bases*, Plenum Press, 1976.

[53] 大須賀節雄, "アドバンスト・データベース・システムへの期待", 情報処理学会アドバンスト・データベース・システム シンポジウム論文集, 1981.
[54] Y. Utagawa, S. Ohsuga, "Design and Implementation of a Database System Based on the Multi-Layer Logic", 情報処理学会アドバンスト・データベース・システムシンポジウム論文集, 1981.
[55] 長沢 勲, 他, "プロローグ系言語 ADL による機械設計プログラム", *Proc. The Logic Programming Con., Tokyo* '83, 1983.

CAD database に対する要求

[56] A.E. Bandurski, D.K. Jefferson, "Data Description for Computer Aided Design", *Proc. ACM SIGMOD Workshop*, 1975.
[57] P.A.V. Hall, "Requirements of Databases for Design", *Proc. Conf. on Database Technology* 1976.
[58] G.M.E. Lafue, "Integrating Language and Database for CAD Applications", *Computer Aided Design*, Vol. 11, No. 3, May, 1979.
[59] T.W. Sidle, "Weakness of Commercial Data Base Management Systems in Engineering *Proc. 17th Design Automation Workshop*, 1980.
[60] C.M. Eastman, "Recent Developments in Representation in the Science of Design", *Proc. 18th Design Automation Conf.*, 1981.
[61] J. Encarnacao, T. Neumann, "A Survey of DB Requirements for Graphical Applications in Engineering", *Lecture Notes in Computer Science 81*, Springer-Verlag, 1980.
[62] K. Bo, "Data Base Design", *Lecture Notes in Computer Science 89*, Springer-Verlag, 1980.
[63] T. Neumann, "CAD Data Base Requirements and Architectures", *Lecture Notes in Computer Science 89*, Springer-Verlag, 1980.
[64] M.F. Challis, "Typing in Data Base Models", *Preprint IFIP Working Conf. on CAD Data Bases*, Sep., 1981.
[65] J. Bell, "Data Modelling of Scientific Simulation Programs", *Proc. Int'l Conf. on Management of. Data*, June, 1982.

Database 一般

[66] CODASYL Development Committee, "An Information Algebra", *CACM*, Vol. 5, No. 4, April, 1962.
[67] C.J. Date, *An Introduction to Database Systems*, 3rd ed., Addison-Wesley, 1981.
[68] W. Kent, *Data and Reality*, North-Holland, 1978.
[69] W. Kent, "Limitations of Record-Based Information Models", *ACM TODS*, Vol. 4, No. 1, March, 1979.
[70] ISO TC97/SC5/WG3, *Concepts and Terminology for the Conceptual Schema and the Information Base*, April, 1982.

哲学的背景

[71] 田中美知太郎編, プラトン「パイドン」(世界の名著「プラトン」), 中央公論社, 1978.
[72] 藤沢令夫訳, プラトン「国家」(岩波文庫), 岩波書店, 1979.
[73] F.M. Cornford, *Plato's Theory of Knowledge*, Routledge & Kegan Paul, 1935.
[74] 今井知正, 「テアイテトス」研究覚書, 「理想」11月号, 理想社 1980.
[75] 出隆訳, アリストテレス「形而上学」(岩波文庫), 岩波書店, 1959.
[76] 長尾雄人編, 「ミリンダ王の問い」(世界の名著「バラモン教典・原始仏典」), 中央公論社, 1979.
[77] B. Russel, *History of Western Philosophy*, 2nd ed., George Allen & Unwin, 1961.
[78] 藤本隆志, 坂井秀寿訳, L. ヴィトゲンシュタイン「論理哲学論考」, 法政大学出版局, 1968.
[79] 仲沢紀雄訳, C. レヴィ=ストロース「今日のトーテミスム」, みすず書房, 1970.
[80] 大橋保夫編, C. レヴィ=ストロース「構造・神話・労働」, みすず書房, 1979.
[81] T.S. Kuhn, *The Structure of Scientific Revolutions*, 2nd., ed., The University of Chicago Press, 1970.
[82] 寺坂英孝/大西正男訳, D. ヒルベルト「幾何学の基礎」(現代数学の系譜 7), 共立出版, 1970.

執筆者紹介 柳生 孝昭 (Takaaki Yagiu)
1957年東京大学理学部数学科卒業. 1958年7月日本ユニバック(株)入社. 現在, 応用ソフトウェア事業部長.

報告 垂直磁気記録と長手磁気記録の実験による比較

Experimental Studies on Longitudinal and Perpendicular High Density Recording

C.S. Chi, K.A. Fray,
R.A. Johnson, W.T. Maloney

要 約 スパッタ法により，固体基板（ガラス）上に，保磁力の異なる1層および2層の垂直記録用のコバルト・クローム (Co-Cr) 磁気記録ディスクを数種類試作し，その磁気特性および記録再生特性を測定した．その結果をニッケル・コバルト (Ni-Co) めっきの高性能長手記録用磁気ディスクと比較した．記録再生特性の測定には，フェライト製および薄膜製の2種のリング・ヘッドを用いた．長手記録用に設計されたリング・ヘッドで測定したにもかかわらず，試作した垂直記録磁気ディスクは，高性能長手記録磁気ディスクと比べて性能上遜色がなかった．2層の垂直ディスクは，記録電流が少なくてすむという利点もある．また予備実験ではあるが，矩形および Hammig の窓を用いた Hilbert 変換信号処理の効果についても述べる．

Abstract Single layer and double layer Co-Cr disks of various coercivities were sputter-deposited on rigid substrates and magnetic parameters measured. Record and playback properties were studied using both ferrite and thin film heads under identical system environments. A well optimized Ni-Co plated longitudinal disk was used as a benchmark throughout this investigation for direct comparison. With the objective of using "off the shelf" ring heads to bring up the perpendicular recording technology on rigid substrates, it was found that the performance of both our preliminary single and double layer Co-Cr perpendicular disks were at least as good as the well optimized longitudinal disk. The double layer disks have an added advantage of lower write current. Signal processing via Hilbert transform using both rectangular and Hamming windows was also studied and applied to the output waveforms.

1. はじめに

垂直磁気記録は，現行の長手磁気記録技術の延長上にあることから，磁気記録技術者の関心の的となっている．記録再生両方にリング・ヘッドを使った研究[1],[2]，両方に単磁極ヘッドを使用，あるいは記録に単磁極ヘッドを，再生にリング・ヘッドを用いた研究[3],[4]など，これまでに多くの研究報告が出ている．単磁極ヘッドは垂直記録に適した磁界分布が得られるとされているが[5]，垂直記録システムの初期研究には，すでに長手記録で実績のあるリングヘッドを使用する方がよいと判断した．本稿では，スパッタ法で作成したコバルト・クローム (Co-Cr) 垂直記録ディスクの作成法と，その磁気特性について述べる．また，記録再生の実験方法について報告し，その実験結果を高性能長手記録用のニッケル・コバルト (Ni-Co) めっきディスクと比較検討する．さらに，25 端子付き遅延線で実現した Hilbert 変換位相等化器による，出力波形改善効果についてふれる．

2. 垂直および長手記録ディスクについて

表1で実験に使用した5種のディスクの磁気特性を示す．4種の垂直記録ディスク（#7～#10）は，アルゴン (Ar) ガス雰囲気中で厚さ 5.6mm の固体基板上に合金ターゲット

を高周波スパッタし，Co-Cr 層を付着させて作成した．基板としては，表面粗さが平均 2.5 nm (0.0025 μm) の焼きなまし耐熱ガラス（パイレックス-7740）を用いた．#7 から #9 のディスクは，Co-Cr 層の1層だけを付着させてある．#10 のディスクは下地にパーマロイを，上層に Co-Cr を，それぞれスパッタした2層構造となっている[6]．保磁力向上のため #9 のディスクだけ，120℃ の予備加熱を行った．なお4枚とも，磁性層生成中の加熱・冷却は一切行っていない．また垂直記録ディスクは，Co-Cr 層の保護とヘッドの潤滑剤を兼ねて表面に厚さ 0,05μm の炭素層をスパッタしてある．

一方，長手記録ディスク (#L) は，アルミニウム (Al) 基板に下地として非磁性の Ni 層を，磁性層として Co-Ni 層を，ともに無電界めっきしたディスクである．保護層には，酸化コバルトを用いている．

表 1　使用ディスク（試験片）の磁気特性

Table 1　Disk (coupon) and system parameter

ディスク	層の厚さ (μm)			保磁力 (エルステッド)		角形比		飽和磁束密度 $4\pi M_s$ (ガウス)	備考
	Fe-Ni	Co-Cr	C	垂直	長手	垂直	長手		
#7	—	0.50	0.05	250	100	0.09	0.16	3000	1層
#8	—	0.56	0.05	300	230	0.11	0.11	3500	1層
#9	—	0.55	0.05	840	380	0.24	0.14	4000	1層
#10	0.55	0.58	0.05	900	400	0.30	0.10	3400	2層
#L	Co-Ni 0.075		Co-Ox 0.12		650		0.80	15000	めっき

基　板：パイレックス-7440，直径 10 cm のディスク
記録再生ヘッド：ギャップ長 1μm，Mn-Zn フェライト・ヘッド (6×6 ターン，
ギャップ幅 17.78μm，浮上量 0.25μm)
ギャップ長 1 μm，3370 型薄膜ヘッド（7 ターン，ギャップ幅 32.02 μm，
浮上量 0.35μm)
実験条件：3560 回転/分，測定トラックは半径 4.6 cm (周速約 1700 cm/s，100 ns が約 14750 磁束反転/インチ相当)

3. 実 験 方 法

実験開始の前に，使用するディスクとヘッドを化学成分を含まない布で清掃しておく．実験はすべてコンタクト・スタート・ストップ方式で行ったが，ディスクやヘッドが傷つくようなことはなかった．ヘッドをディスクの半径 4.6 cm (1.8″) の位置に設定して，毎分 3560 回転の速度で回転させる．この場合ヘッドとディスクの相対速度は，約 1700 cm/s (670 インチ/s) になる．記録面を交流消去後，広帯域 A 級増幅器を通して NRZ 信号を記録する．記録再生には，低浮上量 (0.25 μm) の Mn-Zn フェライト・ヘッドと，3370 型薄膜ヘッドを用いて測定した．薄膜ヘッドの場合，発熱量が大きく最大記録電流が制約された．この制約により高保磁力垂直記録ディスク (#9) は，飽和まで記録電流が流せなかった．ディスクごとに飽和電流をもとに定めた記録電流 (図1)で，孤立パルス列と，3ビット・データ・パターンのパルス列を書き込む．再生特性は，Hilbert 位相等化器のない場合と，ある場合とについて測定した[2]．記録密度測定には掃引正弦波を記録した後，スペクトラム・アナライザで出力波形を表示し，グラフ化して比較図を作成した．

4. 測定結果と検討

図1は，実験で求めた飽和電流曲線である．飽和電流は，5種のディスク（表1）に対して Mn-Zn フェライト・ヘッドおよび薄膜ヘッドで測定した．保磁力の小さい1層ディ

スク（#7, #8）は，高保磁力ディスク（#9）に比べて低電流で飽和するが，再生出力信号が明らかに小さい．2層垂直ディスク（#10）のパーマロイ層は，記録再生特性を大幅に改善（低飽和電流，高出力電圧）している．フェライト・ヘッドと薄膜ヘッドの出力電圧の差は，主としてヘッド浮上量の差（0.25 μm と 0.35 μm）によるものである．

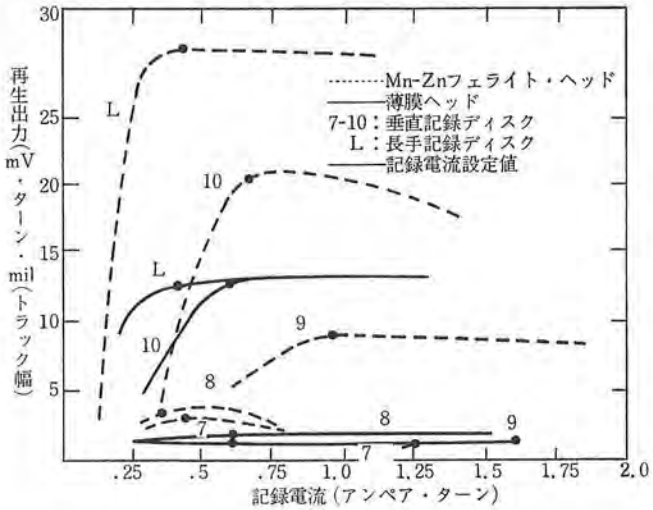
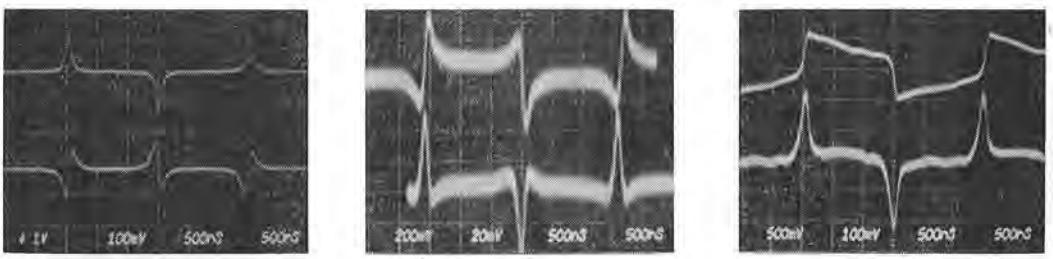


図 1 飽和電流曲線

Fig. 1 Saturation curves

写真1から写真4までの(a),(b),(c)は，それぞれ #L, #9, #10 のディスクに対応している．写真1は，フェライト・ヘッドを用いたときの孤立パルス列の再生信号である．上段は前置増幅器の出力波形であり，下段は Hilbert 位相等化器を通した後の波形である．この等化器は，開発段階のもので改良すべき点がまだ多数残っている．このような不完全な等化器であるが，垂直記録ディスクの出力に適用すると，その出力波形は高性能めっき型長手記録ディスクのそれよりも特性が改善される．写真2は，フェライト・ヘッドに対する周波数特性である．

写真3はフェライト・ヘッドを用いて，600-200-200 ns の間隔の3個のパルスの組み合せを連続して記録再生したときの出力波形である[7]．垂直記録ディスクに位相等化を適用

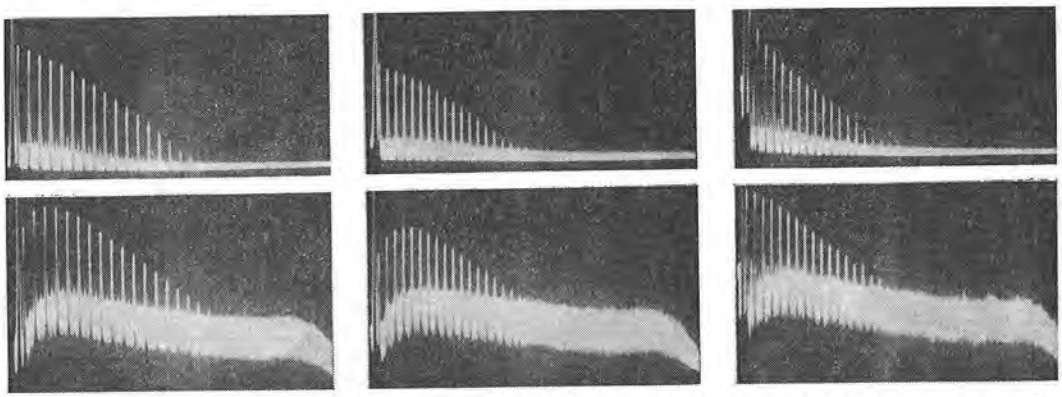


（a）長手記録ディスク（#L） （b）1層垂直記録ディスク（# 9） （c）2層垂直記録ディスク（#10）

写真 1 再生出力波形

（上段：前置増幅器出力，下段：Hilbelt 位相等化後出力．Mn-Zn フェライト・ヘッド使用，トラック半径 4.6 cm, 3560回転/分）

Photo 1 Characteristic read pulses from a longitudinal and perpendicular disks



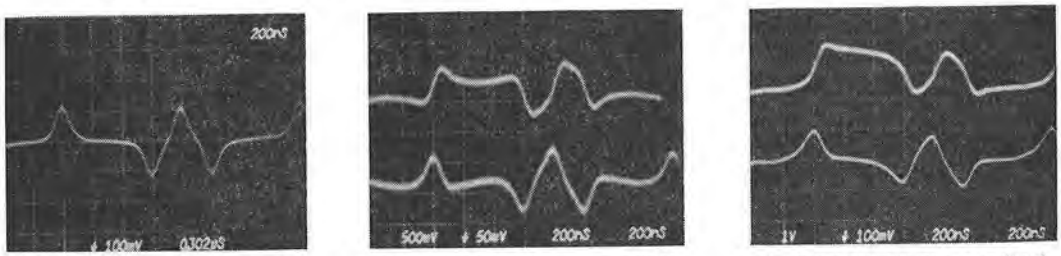
(a) 長手記録ディスク (#L)　(b) 1層垂直記録ディスク (#9)　(c) 2層垂直記録ディスク (#10)

写真 2　孤立波のスペクトル分布

(上段：前置増幅器出力，下段：Hilbelt 位相等化後出力．Mn-Z_nn フェライト・ヘッド使用（1目盛当たり）10 dB および 2 MHz)

Photo 2　Unequalized and Hilbert transformed spectral distributions for isolated read pulses from a longitudinal and perpendicular disks

したときの波形のピーク・シフトは，長手記録よりやや大きくなっている．これは位相等化器が不完全であることと，記録時にヘッド・ギャップが飽和することによるものと思われる．同じ実験を薄膜ヘッドで行った結果が写真4である．写真4(b)がぼけているのは高保磁力1層ディスク (#9) の場合，発熱の制約から飽和まで記録電流が流せなかったためである（3節および図1参照)．

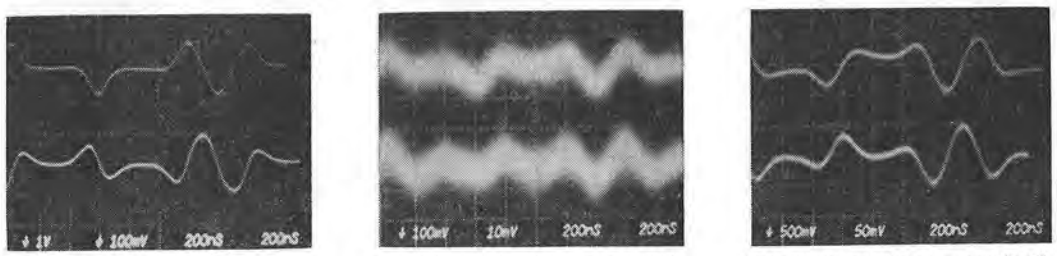


(a) 長手記録ディスク (#L)　(b) 1層垂直記録ディスク (#9)　(c) 2層垂直記録ディスク (#10)

写真 3　600-200-200 ns パルス列の再生出力波形（フェライト・ヘッドの場合）

((b), (c)の上段：前置増幅器出力，下段：Hilbelt 位相等化後出力．Mn-Zn フェライト・ヘッド使用)

Photo 3　Readback triplet data pattern (600-200-200 ns) using a Mn-Zn ferrite head from a longitudinal and perpendicular disks



(a) 長手記録ディスク (#L)　(b) 1層垂直記録ディスク (#9)　(c) 2層垂直記録ディスク (#10)

写真 4　600-200-200 ns パルス列の再生出力波形（薄膜ヘッドの場合）

(上段：前置増幅器出力，下段：Hilbelt 位相等化後出力．3370 型薄膜ヘッド使用)

Photo 4　Readback triplet data pattern (600-200-200 ns) using a 3370 type thin film head from a longitudinal and perpendicular disks

図2は5種のディスクにフェライト・ヘッドで同一条件で記録再生したときの記録密度特性である．1層垂直記録ディスクの場合，再生出力電圧は保磁力に比例すると考えられる（Co-Cr 層の厚さは，ほぼ等しい）．2層垂直記録ディスクは，低密度で再生出力電圧が一番高い．既存の長手記録用リング・ヘッドに若干の改造を加えただけの記録再生ヘッドを用いたにもかかわらず，高密度での応答特性は高性能長手記録ディスク（#L）をやや上回っている．薄膜ヘッドを用いたときもほぼ同様の結果が得られている．

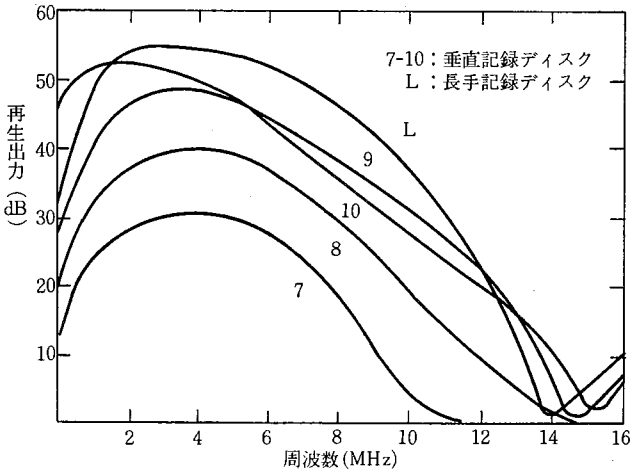


図 2　記録密度と再生出力電圧の関係

（Mn-Zn フェライト・ヘッド使用，トラック半径 4.6 cm, 3560 回転/分：10 MHz の点は 1160 磁束反転/mm に相当）

Fig. 2　Readback amplitude spectra for recording density assessment

5. おわりに

既存の記録再生ヘッドを用いて同一条件下で，ハード・ディスク上における長手記録および1層と2層の垂直記録の記録再生特性を測定した．その結果，位相等化を行った垂直記録ディスクの再生出力は，高性能の Ni-Co めっき型長手記録ディスクのそれと比べて，パルス幅に改善がみられる．1層の垂直記録ディスクの場合，保磁力が増えるに従って再生出力と記録密度が向上するが，反面保磁力が高くなると通常の薄膜ヘッドでは発熱のため記録電流が制約され，飽和させることができなくなる．1層と2層の垂直記録ディスクでは再生出力波形に差がある．前者はパルス幅で優れ，後者は記録密度と低密度の応答性の点で有利である．垂直記録ディスクは，長手記録用に設計された既存のリング・ヘッドを用いても，記録密度，再生信号のパルス幅および重ね書き特性の面で，今日の高性能めっき型長手記録ディスクと同等以上の性能があると認められる．ハード・ディスクによる大容量垂直磁気記録装置が市場に導入されるまでには，磁性媒体の特性，垂直記録再生ヘッドの新しい設計法および信号処理技術に関して大幅の改善・改良が必要であるが，その実現は可能であると思われる．

なお，本研究に当たり指導をいただいた T. Bonn, H. von de Vert, G. Jacoby および C. Lustig の各氏に感謝します．また Hilbelt 位相等化器試作について協力いただいた R. Schissler 氏にも感謝します．

（技術企画部　森　宗正　訳）

参考文献 [1] T. Okuwaki, M. Koizumi, Y. Uesaka and H. Fujiwara, *J. Appl. Physics*, 53, March 1982, p. 2588.
[2] B. J. Langland, *IEEE Trans. Magn.*, MAG-18, November 1982, p. 1247.
[3] S. Iwasaki, Y. Nakamura and H. Muraoka, *IEEE Trans. Magn.*, MAG-17, November 1981, p. 2535.
[4] K. Yamamori, R. Nishikawa, T. Asano and T. Fujiwara, *IEEE Trans. Magn.*, MAG-17, November 1981, p. 2538.
[5] T. J. Szczech, M. Steinback and M. Jodeit ,Jr., *IEEE Trans. Magn.*, MAG-18, January 1982, p. 229.
[6] K. Ouchi and S. Iwasaki, *IEEE Trans. Magn.*, MAG-18, November 1982, p. 1110.
[7] C. S. Chi, *IEEE Trans. Magn.*, MAG-18, November 1982, p. 1197.

執筆者紹介 Chao S. Chi

電気工学の分野で，台湾の Cheng Kung 大学の B. S. および Worcester の Polytechnique Institute の M. S. と Rh. D. を取得．IEEE の上級会員で，Boston の IEEE magnetic Society の元議長．1967年から1971年まで RCA の PTS，1971年から5年間は DEC の主席プロジェクト・エンジニア，1977年から1983年まで Sperry Research Center の Magnetic Recording 部のマネジャーであった．現在 MPI 社の Advanced Systems 部のグループ・マネジャー．

Robert A. Johnson

1964年，Rensselear Polytechnique Institute より物理学で B. S. を，1971年に，Brown 大学より Ph. D. 号を取得．1971年から1974年まで，MIT で超伝導性についての研究を行い，1974年から1979年まで Bucknell 大学で物理学の講義を行った．1979年から1983年まで Sperry Research Center で磁気記録について研究．現在，Polaraid 社に在職．

William T. Maloney

1957年と1958年に Case 工科大学より電気工学で B. S. と M. S. をそれぞれ取得．また1959年と1964年に Harvard 大学より応用物理学の A. M. と Ph. D. をそれぞれ取得．1963年から1965年まで応用物理学の講師として，同大学に残り，1964年に特別研究員となった．1965年には Sperry Research Center のテクニカル・スタッフとなり，マイクロウェーブとプラズマの相互作用，固体の超音波／光学的研究，光信号処理および光メモリについて研究．最近6年間は，ディジタル磁気記録方式，および磁気記録媒体の研究に従事．現在，Polaroid 社に在職．Tau Beta Pi, Eta Kappa Nu, Sigma Xi および米国物理学会の会員．

Karl A. Frey

1961年に，Northeastern 大学より，電気工学の B.S.E.E. を取得し，コンピュータのプロダクト開発と磁気記録の研究・開発に従事してきた．その間，Honeywell, Sperry Univac を経て，DEC などのコンピュータおよび周辺機器の会社に在職し，現在，DEC の Advanced Development グループに在職．

報告 仕様記述言語 P3

The High-level Language P 3 in Bussiness Applications

長谷川 邦夫

要 約 本稿では，われわれが開発したプログラミング言語 P3 を紹介する．最近のいろいろな新しいプログラミング言語の流れは，仕様記述とプログラミングとの一体化の可能性を示している．P3 は事務処理分野での仕様記述級のプログラミング言語であり，P3 のプロセッサは COBOL プログラムを生成する．すなわち P3 の狙いは，容易に，わかりやすく仕様記述できることと，その仕様記述からプログラムを生成することにある．

本稿では，P3 の考え方と言語の主な構成について述べる．

Abstract This paper describes a high-level programming language named P 3, which we have developped. P 3 can be used for business applications. P 3 may be a language at the level of specification. The specification in P 3 is translated into an executable COBOL program automaticaly by P 3 processor.

Concepts and outlines of P 3 are described in this paper.

1. はじめに

プログラミングでの新しい概念と言語（たとえば，論理型プログラミング，関数型プログラミング，オブジェクト指向プログラミング，言語では PROLOG や SMALLTALK など），あるいはシステム開発過程でのモデリングやプロトタイピングなどのアプローチの方法は，プログラミング・スタイルに変化をもたらしそうである．それは，仕様とプログラムの間の距離が大幅に縮まる，あるいは，仕様記述とプログラミングが一体化する方向にある，ということである．

たとえば，COBOL やFORTRAN でプログラムを作らねばならず，しかも設計者とプログラマという制度が存在している開発環境の中では，仕様の作成はプログラムの作成に先立つ分離した作業であるという前提がある．そこで，仕様記述の方法について，かなり論じられてきてはいるのであるが，まだ実用的ではなく，現実には，プログラムがひとり歩きする仕様離れの現象が続いている．

論理型プログラミングや関数型プログラミングおよびオブジェクト指向プログラミングは，仕様とプログラムの一体化について一つの方向を示している．また，モデリングもプロトタイピングも，前者は静的，後者は動的という違いはあるが，やはり仕様記述とプログラミングの一体化の方向を示している．

高級プログラミング言語の“高級”という内容には，記述の量を減らす意味もあれば，正しさを検証できるということも含まれるが，高級の行きつく先は“仕様記述級”というものであろう．そのときに必要なのは完全な形式的仕様の形式ではなくて，人間が容易に，かつわかりやすく記述できる方法である．そのためには，対象分野に即した記述方法を取り入れることが必要である．そのための一つの方法は，対象分野における特有の概念や処理を言語の要素として，言語の中に実現しておくことである．

本稿で紹介するP3は，事務処理の分野における「仕様記述級のプログラミング」を目指して開発した．すなわち，P3とは事務処理でみられる問題の大部分の仕様を記述するための道具であり，そのコンパイラは仕様記述からCOBOLプログラムを作り出す．

以下，次の項目について説明する．

1) 事務処理プログラムの特徴
2) 事務処理のためのプログラミング言語要素
3) 仕様記述の方法
4) P3言語の概要
5) P3の処理系

2. 事務処理プログラムの特徴

P3が対象とする事務処理は，次の特徴をもっている．

1) 入力はトランザクション（伝票）とマスタ・ファイル（帳簿）であり，出力は更新されたマスタ・ファイルと報告書である．
2) データ量は多いが，一つ一つのデータに対する処理は同じで，単純である．
3) 基本となるのは，次の定形処理である．すなわち，トランザクションを一つずつ順次入力したとき，その形式の検査とマスタ・ファイルとの照合の結果，誤りのないデータから，新しいマスタ・ファイルのデータと報告書を作る．
4) 上の基本的な処理に加えて，利用者の求めに応じて報告書の編集，作成がある．
5) 上の3)と4)の目的遂行のために，分類・併合・集計・転記などの処理がある．

要するに，トランザクションとマスタ・ファイルと報告書とが，入力あるいは出力として存在し，その処理はいくつかの定形の処理パターンに分類できる．現実の事務処理システムでは，データの実現の方法にバラエティがあるので（たとえば，端末装置から入力するとか，マスタ・ファイルは索引ファイルであるとか，より汎用のデータベースであるとか），一見複雑なものにみえることがあるが，本質は単純である．

3. 事務処理のためのプログラミング言語要素

前に述べた事務処理自体の定形性と単純性に着目して，事務処理のための簡易プログラミング言語や仕様記述言語が，すでにいろいろ開発されてきている．それらの大部分に共通していえることは，事務処理はいくつかの処理パターンに分類できる，という特徴に着目して，言語の中にそれらのパターンを用意していることである．たとえば，照合，集計，転記，選択のような機能別要素ができ上がっている．しかし，この処理パターンの分類というのは，その処理の意味や目的に沿ったものではなく，処理のロジックで分けたものであることに注意しなければならない．また，ある時期の取引データの中から，製品別に最も注文量の多い取引データを抜き出すことも，取引相手別に最近の取引データのリストを作ることも，言語によっては集計のロジックで記述せざるを得ないものがある．しかし，どちらも集計という意味合いは薄い．

処理のロジックで分類した処理パターンでも記述できるはずであるが，かえってパターンの名前や形が記述された内容の理解を妨げることがありそうだ．

4. 仕様記述の方法

3章で，事務処理の処理パターンを言語要素にするだけでは，「仕様記述級のプログラミング」には向かないことを述べた．P3 の説明に入る前に，仕様の記述方法について述べる．

一般に仕様記述方法というと，厳密性，形式性を重要視するあまり，論理的記述に注目しがちだが，1章で述べたように事務処理分野で求めているのは，人間が容易に，わかりやすく記述できる方法である．

そこで，再び事務処理の特徴を考えてみると，データ量は多くとも，個々のデータ，あるいはキー項目で区別されるデータの集まりごとに同じ処理が施される，という特徴がある．その処理では，入力されたデータの内容を使って，目的の出力データが作られるわけである．

したがって，仕様記述の骨組は次の項目からなるであろう[1]．

1) 何が入力で，何が出力であるか．
2) 処理を施す入力データの単位は何か．
3) 入力データの単位から，出力データの内容をどう作るか．

この3項目で，入力，出力の媒体は別として，順次に入力されるデータから，順次にデータを出力する処理の仕様が記述できる．すなわち，事務処理をシーケンシャル・プロセスとして記述しようというのである（図1）．この論理レベルの記述によって仕様は完結するが，P3 はプログラム生成を目的としているので，物理レベルの記述を加えることにする．すなわち，次の項目を加えた4項目が仕様記述の骨組である．

4) 入力および出力はどのように実現するか．

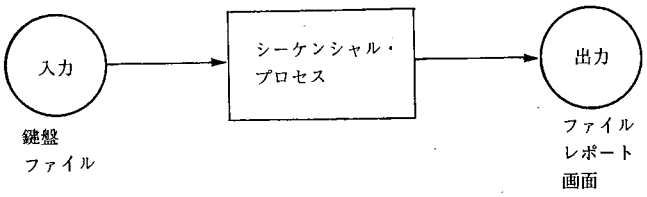


図 1　順ファイルを入出するシーケンシャル・プロセス

Fig. 1　A sequential process with input files and output files in bussiness application

5. P3 言語の概要

P3 言語で書く「仕様記述級プログラム」は，五つの部分からなる．これらは4章で示した仕様記述の骨組に沿っている．

1) DATA-FLOW 部
2) DATA 部
3) PROCESS-FLOW 部
4) PROCESS 部
5) IMPLEMENT 部

図2は，P3 による完全な記述の一例である．このプログラム TTL は，入力ファイル中のタイプ (R-TYPE) が1のレコードを対象として，それに含まれる量 (QTY) の，部品

```
 1      $ID TTL
 2      $DATA-FLOW INPUT IN-F
 3                 OUTPUT OUT-F
 4
 5      $DATA  $FILE IN-F
 6             $RECORD IN-F-R1
 7                 02 R-TYPE  PIC 9.
 8                 02 PART-NO PIC 9(4).
 9                 02 QTY     PIC 9(5).
10
11             $FILE OUT-F
12             $RECORD OUT-F-R SAME-AS IN-F-R1
13
14             $FILE IM-F
15             $RECORD IM-F-R SAME-AS IN-F-R1
16
17     $PROCESS-FLOW
18     TTL1 DEFINE RECORD-OF IM-F SORTED BY PART-NO
19              ON RECORD-OF IN-F;
20     TTL2 DEFINE RECORD-OF OUT-F
21              ON RECORD-GROUP-OF IM-F GROUP-KEY PART-NO
22
23        $PROCESS TTL1
24             $OUTPUT IM-F
25                 EDIT IM-F-R ON (R-TYPE OF IN-F-R1 = 1)
26
27        $PROCESS TTL2
28             $OUTPUT OUT-F
29                 EDIT OUT-F-R BY ED-OUT
30             $EDIT ED-OUT
31                 R-TYPE OF OUT-F-R = 1;
32                 PART-NO OF OUT-F-R = VALUE-OF(PART-NO OF IN-F-R1, FIRST);
33                 QTY OF OUT-F-R = SUM-OF(QTY OF IM-F-R);
34
35
36     $IMPLEMENT
37             $FILE IN-F  DISC,ST,10
38             $FILE OUT-F DISC,ST,10
```

図 2　P 3 による記述例

Fig. 2　A sample of the text in P 3

番号 (PART-NO) ごとの集計値を求めるものである．行 18 の SORTED 句は，中間ファイル (IM-F) を部品番号の昇順にソートすることを示す．

つぎに，各部について概略を述べる．

5.1 DATA-FLOW 部

プログラムの入力ファイル，出力ファイルを明示することによって，プログラム点でのデータ・フローを示す．

```
$DATA-FLOW
    INPUT   ファイル名…
    OUTPUT  ファイル名…
    I-O   ファイル名…
```

I-O は，プログラム中で入力および出力の両方に使うファイルを示す．

5.2 DATA 部

プログラムが扱うファイルのデータ記述をする．ほとんど COBOL のデータ記述の方法によるが，P 3 特有の簡便記法が用意されている．

```
$DATA
  $FILE  file-1
  $RECORD  rec-1
    ⋮ rec-1 のレコード記述 (COBOL の記法)
$FILE  file-2
```

```
$RECORD  rec-2 SAME-AS rec-1
```

SAME-AS は，rec-2 が rec-1 と同じレコード記述であることを示すのに使う．

報告書の場合には，次の書き方を使う．

```
$FILE  file-3
$REPORT  report-1
        ⋮ レポート記述 (COBOL のレポート記述による.)
```

5.3 PROCESS-FLOW 部

シーケンシャル・プロセスにおける入力の処理単位を示す．つまり下の形式の DEFINE 文で，入力のどんな単位で出力を作るかを記述する．

```
$PROCESS-FLOW
    プロセス名  DEFINE  出力  ON  入力
```

たとえば，入力ファイル inf-1 の各レコードから，出力ファイル outf-1 の各レコードを作るプロセス prl は例1のようになる．

例 1

```
prl  DEFINE
    RECORD-OF  outf-1
    ON  RECORD-OF  inf-1
```

RECORD-OF の語は常に省略できる．したがって，

```
prl  DEFINE  outf-1  ON  inf-1
```

と書いてもよい．

例 2　inf-1 中の，キー項目 key-1 が同じ値をもつレコード群の各々から出力レコードを作る場合であれば，次のように書く．

```
prl  DEFINE  outf-1
    ON  RECORD-GROUP-OF  inf-1
        GROUP-KEY  key-1
```

ON の後の句が，出力を作るための入力単位，すなわち key-1 で類をなすレコード群を示している．

例 3　一つのプロセスが，二つ以上の出力ファイルを作ることもある．その場合は，DEFINE の後に各々の出力ファイルを示せばよい．

```
prl  DEFINE  optf-1, outf-2, …  ON  inf-1
```

例 4　事務処理での照合を考えよう．2章で示したように，トランザクション・データに対してマスタ・ファイルとの照合が必要である．マスタ・ファイルを mas-1 とし，照合キーを，トランザクションもマスタ・ファイルも key-1 ということにすれば，そのためのプロセスは，次のように記述する．REFER 句が照合を表すことになる．

```
prl  DEFINE  outf-1  ON  inf-1
    REFER  mas-1  BY  Key-1
```

例 5　たとえばトランザクション inf-1 とマスタ・ファイル mas-1 とから，キー項目 key-1 の昇順に順次1件ずつレコードを入力して，key-1 の並びの順に出力データを作る場合には，次のように DEFINE 文を用いて記述する．

```
prl  DEFINE  outf-1
    ON  mas-1  AND  inf-1
    MATCHING-KEY  key-1/key-1
```

例 6　事務処理システムはいくつかのプログラムで構成され，それらはファイルで連結されている．また，一つのプログラムも種々の事情から大きな複雑なものになり得る．その場合，プログラムの中ではデータが順次加工されて最終的な出力に変換されることが多い．そのために，P3 のプロセス・フロー部には二つ以上のプロセスを記述することができるようになっている．プロセスはシーケンシャル・ファイルを介して結ばれる．すなわち，図3のようなシステムであれば次のように記述する．

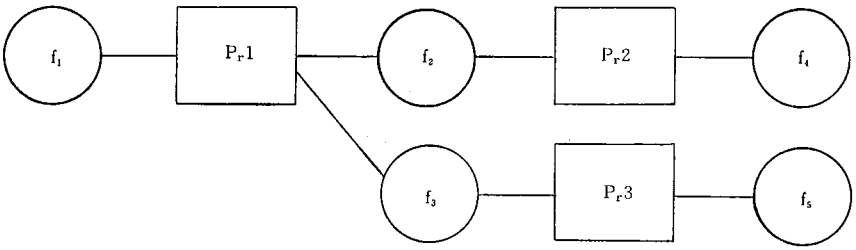


図 3　プロセス・フロー
Fig. 3　Process flow

```
$ PROCESS-FLOW
    pr1  DEFINE  f2, f3  ON  f1;
    pr2  DEFINE  f4  ON  f2;
    pr3  DEFINE  f5  ON  f3
```

以上の他に DEFINE 文では，

1)　入力あるいは出力ファイルの分類

2)　二つの入力ファイルの併合

を記述することができる．また出力を作るための入力レコード群の指定のしかたには，キー項目による以外の方法も用意されている．

5.4 PROCESS 部

DEFINE 文では，出力を作るための入力の単位を明示したが，PROCESS 部では

1)　出力を作る条件

2)　出力データの内容

について記述する．

5.4.1 出力を作る条件

出力を作るための入力の単位は，入力ファイルの中に繰り返し現れる．常に出力を作るのであれば，出力を作る条件を記述することは必要ないが，そうでないときは陽に示さなければならない．

たとえば，5.3 節の例1で，出力レコード outr-1 は，入力レコード inr-1 の中にある項目 item の値がAであるものから作るのであれば，次の記述をする．

```
$ PROCESS  pr1
  $ OUTPUT  outf-1
      EDIT  outr-1  ON  item=▽A▽
```

条件を示す ON の代わりに IF 文で，

```
IF  item=▽A▽
    EDIT  outr-1
```

```
ENDIF
```

とすることもできる．EDIT 文が出力の作成を示す．

5.4.2 出力データの内容

入力データと出力データの関係を式で示す．式の左辺は出力レコードの項目名，右辺は入力レコードの項目名である．

```
EDIT  outr-1  BY  edit-1
                  ⋮
$EDIT  edit-1
    oitem-1=item-1
    oitem-2=item-2
                  ⋮
```

出力を示す EDIT 文中の BY の後に，入力項目と出力項目の関係記述に付けた名前を示している．

5.4.3 関　数

出力を作る入力の単位が RECORD-GROUP-OF で示されるレコード群の場合（5.3 節の例2》には，出力の条件を記述するため，あるいは，出力データの内容を記述するために P3 の組込み関数を使うことになる．そのような関数のいくつかの例として，次のものがある．

- SUM-OF (item)

  入力レコード群中での item の集計値

- VALUE-OF (item, n)

  入力レコード群中での n 番目のレコードの item の値．整数 n の代わりに LAST とすれば，最後のレコードの item の値．

関数には他に，入力ファイル間の関係を示すものがある．たとえば，REFER（5.3 節の例4》で指定するマスタ・ファイルとの照合結果を示す FIND 述語や，MATCHING（5.3 節の例5》を指定したときに使うことのできる MATCH, UNMATCH 述語などがある．

5.5 IMPLEMENT 部

ここでは，主にファイルの実現の方法を記述する．プロセスの入力，および出力ファイルの物理的情報を次の形で指定する．

```
$IMPLEMENT
    $FILE  inf-1  CARD-READER
    $FILE  outf-1  PRINTER
```

ここで，以下のことは重要である．

IMPLEMENT で実現方法の与えられていないファイルは，概念的なファイルである．すなわち，プロセスとプロセスを結ぶ概念ファイルとは，1メッセージ分のメッセージ・バッファである．

6. P3 の処理系

P3 のコンパイラは，COBOL プログラムを作るプリプロセッサである（図4）．

IMPLEMENT 部で，実現方法の記述をしない概念ファイルはメッセージ・バッファであると述べたが，メッセージ・バッファによるプロセスの結合は，Jackson によるプログラミング法[2]のプログラミング・インバージョンの技法で実現している．

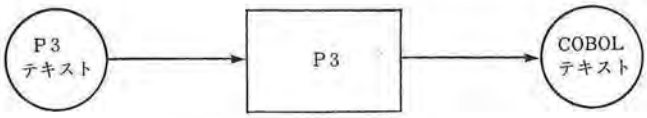


図 4　P3 プロセッサ
Fig. 4　P3-Processor

7. おわりに

本稿では，P3 による仕様記述級のプログラミングの概略とその特徴を述べた．P3 はその応用分野に即した要素をもつが，このことは，逆に言語としての汎用性がなくなることでもある．P3 は事務処理向きであり，かつ COBOL をベースにしているとはいっても，COBOL で記述できるのと同じように，何でも P3 でプログラミングできる訳ではない．

P3 には，改良すべき点がいくつかある．なかでも，

1)　COBOL 色をできるだけなくす．

2)　IMPLEMENT 部を強力にして，オンラインやデータ・ベースとのインタフェースを可能にする．

3)　日本語による記述を取り入れる．

などが，今後の有用性を増すことになると考える．

また，2) については，物理レベルの記述であってプログラム仕様とは区別して考えるべきことであるが，プログラム自動生成という P3 の一面にとっては必要なことである．

なお，本稿で紹介した仕様記述言語 P3 は商品化を計画しており，本年 7 月に発表の予定である．

最後に，P3 の構想，開発に大きな力を尽された市丸信子さん，森本理君，および P3 に到るまでに適切な助言と支援を寄せられた方々に感謝します．

参考文献　[1] S. Itakura, K. Hasegawa, "A very High-Level Language for Bussiness Data Processing", *Proc. of The 21 st Annual Spring Technical Symposium*, Vol. II, Sperry, 1983.
[2] M. Jackson, *Principles of Program Design*, Academic Press, 1975.

執筆者紹介　長谷川　邦夫　(Kunio Hasegawa)
昭和 23 年生．47 年東北大学理学部卒業，同年(株)日本ユニバック総合研究所入社，51 年日本ユニバック(株)に移籍，入社以来事務処理分野でのユーザ・システム開発に従事，近年はシステム開発およびプログラム開発技法の調査・研究を主とする．現在，システム開発事業部に所属．

報告 DCG トランスレータの製作と言語解析への応用

An Easy Implementation of Definite Clause Grammar Translator and its Application to Language Analysis

山 田 真 市

要 約 現存の UP 10 Q のようなパーソナル・コンピュータの利用環境における自然言語利用の実験を進めてきている．本稿では，その中から，利用価値の高い DCG トランスレータのやさしい製作法を紹介し，簡単な応用例を述べる．ここに述べる DCG トランスレータはパーソナル・コンピュータ用に Micro Prolog[1] を使って書いたものであるが，Pereira と Warren[2] に述べられている全機能を含んでいる．また DCG が組み込まれていない Prolog へ容易に移植でき，自然言語あるいは人工言語の解析に応用できる．

DCG (Definite Clause Grammar) は A. Colmerauer[3] と R. Kowalski[4] によって提案された自然言語，あるいは人工言語を簡潔に記述するための形式系であって，文法のコンパクトな記述を可能にする．DCG トランスレータは DCG で記述した文法を対応する Prolog の下降形パーサ (topdown parser) へ変換する単純な Prolog のプログラム・モジュールである．DCG トランスレータで生成される Prolog プログラムは Prolog の計算機構によって下降形パーサとして動作するが，DCG を用いて上昇形パーサ (bottom up parser) を書くこともできる．松本や田中ら[5] の BUP の原理と同様の方法で，上昇形パーサを生成する簡単な応用例を述べる．

Abstract We have been making experimental studies on natural language processing in personal computing environment. This report describes an easy implmentation of a version of DCG (Definite Clause Grammar) translator in Prolog, which has been proved to be an useful tool for language analysis in our linguistic experimentation. Whereas the DCG translator is written in Micro Prolog[1] for small personal computers, all functions of DCG stated in Pereira and Warren[2] are incorporated into it. The DCG translator can also be easily transplanted onto any Prolog in which DCG is not available, and thus facilitates language analysis for both natural and artificial languages. DCG is a clear and powerful descriptive formalism for both natural and artificial languages due to A. Colmerauer[3] and R. Kowalski[4] and makes it possible to give a compact description of grammatical information. The DCG translator is a simple Prolog program module which convert the grammar file in DCG to the corresponding topdown parser in Prolog.

While the Prolog program produced by the DCG translator operates as a topdown parser according to the computation architecture of Prolog, a bottom up parser can also be written in DCG. A simple example of bottom-up parser in DCG is described on the same lines with BUP of Matsumoto and Tanaka[5].

1. はじめに

DCG は，論理式を Horn 節に限定した一階の論理を用いて対象とする言語を記述する一つの形式系である．したがって，文法としては DCG は A, Colmerauer のメタモルフォシス文法[3]を文脈自由規則の形へ制限した，いわばパラメタをもつ文脈自由文法である．しかし，表現の能力の点では DCG は Chomsky のタイプ 0 と同等であり，この制限は理論的には本質的ではない[6,7]．一方，Pereira と Warren[2] が述べるように，DCG は Horn 節の論理で表現されるから，文法のコンパクトな記述を与えると同時に，元来 Prolog と自然に対応している．それゆえ，簡単なシンタクティカルな変換によって DCG で記述した文法を Prolog の構文解析プログラムに変換することができる．実際，DEC シ

ステム 10 Prolog のように，あらかじめ DCG が組み込まれている Prolog の処理系もある[8]．しかし，普通，実際に用いたようなパーソナル・コンピュータ用 Prolog には DCG が組み込まれていない．そこで，Pereira と Warren[2] の述べる DCG と同一機能をもつパーソナル・コンピュータ用の DCG を定め，DCG の文法ファイルを対応する Prolog の下降形パーサに変換するトランスレータを Prolog のプログラム・モジュールとして用意し，実験で使用した．この DCG トランスレータは構文変換プログラムの演習問題という程度のプログラムであるが，その有用性を考えてここに紹介する．なお，この DCG トランスレータは容易に任意の Prolog へ移植できる．とくに，使用した Micro Prolog はリスト記法を用いているから，同様の Prolog あるいは PiL (Prolog in Lisp)[9] にはそのまま移植できる．

なお，簡単な例を Winograd[10]，および Pereira と Warren[2] からとって，この DCG トランスレータの使用例を述べる．また Winograd[10] からとった例について上昇形パーサの生成の事例を述べる．

以下において，Prolog と DCG の知識を仮定する．DCG の詳細については Pereira と Warren[2] を合わせて参照されたい．

2. DCG の記法

Pereira と Warren[2] の DCG と，ここに述べる DCG とは同一の機能をもつが，使用した Micro Prolog の記法に従って若干の構文上の相異がある．まず，使用した Micro Prolog においては項 (compound term) と アトム (atomic formula) はリスト記法を用いて表現する．関数記号および述語記号は前置記法 (prefix notation) によりリストの第一要素として書く．たとえば，図1に示すように，項 (compound term)：

s(x, y)

は，

(s x y)

と表記し，アトム：

sentence (s(x, y))

は，

(sentence (s x y))

と表記する．また節 (clause) はアトムのリストで表す．その第一要素が節の頭 (head) であり，その cdr 部分が体 (body) である．すなわち，Kowalski の記法：

$A \leftarrow B_1, B_2, \cdots, B_n$

は，

$(A\ B_2\ B_2\ \cdots\ B_n)$

と表記する．この記法を DCG でも採用する．

図1は，典型的な DCG の文法則の対話型入力と変換された Prolog プログラムのリストとの対応を示している．PROLOG で Prolog のセッションに入り，LOAD DCG で DCG トランスレータをロードしている．プロンプト & に続く行で，

(sentence (s x y))→(noun-phrase x), (verb-phrase y)

は，「文 (sentence) は名詞句 (noun-phrase) と動詞句 (verb-phrase) からなる．」という DCG の文法則である．それぞれの引数 (s x y), x および y は構文木のリスト表現をうるためのパラメタである．この文法則を対話型コマンド add(・) でDCG に入力している．DCG は

```
LOAD DCG
&.add((sentence (s x y))->(noun-phrase x),(verb-phrase y))
&.LIST(sentence)
((sentence (s
       X
       Y) Z x)
  (noun-phrase X Z y)
  (verb-phrase Y y x))
&.add((noun x)->(ter (x)),{(rootform x y),(is-noun y)})
&.LIST(noun)
((noun X Y Z)
  (ter (X) Y Z)
  (rootform X x)
  (is-noun x))
&.add((category arguments)->(ter (x)),{(cat x arguments)})
&.add({(cat word arguments)})
&.LIST(category)
((category arguments X Y)
  (ter (Z) X Y)
  (cat Z arguments))
&.LIST(cat)
((cat word arguments))
&.add((date x y)->(month y),(ter (x)),{(NUM x),(LESS 0 x),(LESS x 32)})
&.LIST(date)
((date X Y Z x)
  (month Y Z y)
  (ter (X) y x)
  (NUM X)
  (LESS 0 X)
  (LESS X 32))
```

図 1　DCG の記法

Fig. 1　DCG notation

この文法則を Prolog のプログラムへ変換する．コマンド LIST (sentence) で変換後の Prolog プログラムをリストすると，Prolog プログラム：

((sentence (s

⋮

(verb-phrase Y y x))

へ変換されている．

次の例は，Pereira と Warren[2] の 3.3.1 項に対応するカツコ {,} の使用法と辞書表現の用例である．すなわち，

(noun x)→(ter (x)), {(rootform x y), (is-noun y)}

において，(ter (x)) はリスト (x) の要素 x が終端記号 (terminals) であるという性質を表しており，カツコ {,} は Pereira と Warren の記法と同じ機能をもつ．すなわち，A_1, A_2, …, A_n をアトムとするとき，DCG の表現：

$\{A_1, A_2, \cdots, A_n\}$

は，アトムのリスト：

$(A_1\ A_2\ \cdots\ A_n)$

へ変換される．したがって，カツコ {,} の中には任意の Prolog プログラムを記入できる．

図 1 で，LIST (noun) で変換後の (noun) のプログラムをリストすると，Prolog プログラム：

((noun X Y Z)

⋮

(is-noun x))

へ変換されている．

つぎに辞書表現の一般的な形：

(category arguments)→(ter (x)), {(cat arguments)}

を add (') で入力すると，コマンド LIST (category) でリストされた Prolog プログラム：

((category arguments X Y)
(ter (Z) X Y)
(cat Z arguments))

が得られる．カツコ {,} によって，Pereira と Warren[2] の 3.3.2 項と同様に条件 (extra conditions) を Prolog で記述できる．図 1 の最後の例は Pereira と Warren の文献[2]. 3.3.2 項の同一の例の DCG 表現と対応する Prolog プログラムを出力したものである．

Backus Naur Form と同様に，文法則を簡約形で表現することもできる．たとえば，

S→NP, VP
S→Q, NP, VP
S→neg, NP, VP
S→Q, neg, NP, VP

を BNF で表記すると，

S→(Q| ( )), (neg|( )), NP, VP

と簡約でき，Pereira と Warren の記法では，

S→(Q ; [ ]), (neg ; [ ]), NP, VP

と表現できる．ここに述べる DCG では前置記法の Micro Prolog の組み込み述語 OR を用いて，これを

(S)→(OR ((Q)) ((ter( )))), (OR ((neg)) ((ter( ))), (NP), (VP)

と表す．図 2 にこの簡約形の用例を示す．

```
&.LOAD DCG
&.add((s)->(OR ((Q))((ter ())) ),(OR ((neg))((ter ())) ),(NP),(VP))
&.LIST(s)
((s X Y)
  (OR (((Q
          X
          Z))) (((ter
          ()
          X
          Z))))
  (OR (((neg
          Z
          x))) (((ter
          ()
          Z
          x))))
  (NP x y)
  (VP y Y))
```

図 2　OR による簡約形の用例

Fig. 2　Use of OR in DCG grammar

LIST (s) で変換された Prolog プログラムも示してある．この他に，命令形の表記用に，文法則の左側に，

nonterminal, terminal

の形式を書くこともできる[cf. [8], [11]].

なお，以下に DCG の構文法 (Syntax) を BNF 流の記法でまとめる：

DCG の構文則

(Syntax of Definite Clause Grammars)

1) ⟨rule⟩ : : = ⟨head⟩ − > ⟨body⟩
2) ⟨head⟩ : : = ⟨nonterminal⟩
   | ⟨nonterminal⟩, ⟨terminal⟩

3) 〈body〉::=〈body〉, 〈body〉
|〈OR-item〉|/|〈nonterminal〉
|〈terminal〉|〈condition〉

4) 〈nonterminal〉::=〈Prolog atom〉

5) 〈terminal〉::=(ter 〈list〉)

6) 〈OR-item〉::=(OR 〈list of atom〉〈list of atom〉)

7) 〈condition〉::={〈atoms〉}

8) 〈atoms〉::=〈Prolog atom〉|〈atoms〉, 〈Prolog atom〉

9) 〈list of atom〉::=(〈atoms〉)

ここで,

「→」「/」「ter」「{」「}」「(」「)」「,」は DCG の記号である.

なお, コマンドとしては add, convert, load save, delete, kill が用意してある.

3. 応用例 1——GRAMMAR 1

まず簡単な例から述べる. 図 3 は Winograd[10] からとった文脈自由文法の例を DCG で表した文法ファイルである. 終端記号は, Pereira と Warren の記法を用いている.

```
 1: {{{sample of context-free grammer from T.Winograd(1972) }}}
 2: { (sentence (S x y)) -> (noun-phrase x), (verb-phrase y) }
 3: { (noun-phrase (NP x y)) -> (determiner x), (noun y) }
 4: { (verb-phrase (VP x y)) -> (trans-verb x), (noun-phrase y) }
 5: { (verb-phrase (VP x)) -> (intrans-verb x) }
 6: { (determiner (DET x)) -> (ter (x)),{(is-det x)} }
 7: { (noun (N x)) -> (ter (x)), {(is-noun x)} }
 8: { (intrans-verb (IV x)) -> (ter (x)), {(is-intrans x)} }
 9: { (trans-verb (TV x)) -> (ter (x)), {(is-trans x)} }
10: {{{is-det the)}}
11: {{{is-noun giraffe)}}
12: {{{is-noun apple)}}
13: {{{is-intrans dreams)}}
14: {{{is-trans eats)}}
15:
16: {{{ter () X X)}}
17: {{{ter (x) (x|X) X)}}
```

図 3 応用例 1 の DCG 文法ファイル GRAMMAR 1

Fig. 3 DCG file GRAMMAR 1 of example 1

```
LOAD DCG
&.convert(GRAMMER1 PARSER1)
Converting DCG file GRAMMER1 to Prolog file PARSER1
* * * Conversion completed * * *


&.LOAD PARSER1
&.?((sentence x (the giraffe dreams) ())(P x)PP)
(S (NP (DET the) (N giraffe)) (VP (IV dreams)))


&.?((sentence x (the giraffe eats the apple) ())(P x)PP)
(S (NP (DET the) (N giraffe)) (VP (TV eats) (NP (DET the) (N apple))))


&.?((sentence x (the apple dreams) ())(P x)PP)
(S (NP (DET the) (N apple)) (VP (IV dreams)))


&.?((sentence x (the apple eats the giraffe) ())(P x)PP)
(S (NP (DET the) (N apple)) (VP (TV eats) (NP (DET the) (N giraffe))))
&.QT.
```

図 4 応用例 1 の実行例

Fig. 4 Sample results of example 1

16, 17行目は終端記号の connect 用のプログラムであり，補足する必要がある．この文法ファイルはエディタでディスケットに編成されており，GRAMMAR 1 というファイル名が付けられている．

図4はこのファイルの変換と変換後の Prolog のパーサによる構文解析の結果を示している．コマンド：

convert (GRAMMAR 1 PARSAR 1)

によって，DCG の文法ファイル GRAMMAR 1 は Prolog のプログラム・ファイル PARSER 1 へバッチ・モードで変換され，ディスケット上に書き出される．図4は LOAD PARSER 1 で，それをロードして，入力文に対してその構文木を出力する実行例を示している．

また，図5は出力された Prolog ファイル PARSER 1 のリストである．内容は自明である．

```
 1: ((is-trans eats))
 2: ((is-intrans dreams))
 3: ((is-noun giraffe))
 4: ((is-noun apple))
 5: ((is-det the))
 6: ((ter () X X))
 7: ((ter (X) (X|Y) Y))
 8: ((intrans-verb (IV
 9:        X) Y Z)
10:   (ter (X) Y Z)
11:   (is-intrans X))
12: ((trans-verb (TV
13:        X) Y Z)
14:   (ter (X) Y Z)
15:   (is-trans X))
16: ((noun (N
17:        X) Y Z)
18:   (ter (X) Y Z)
19:   (is-noun X))
20: ((determiner (DET
21:        X) Y Z)
22:   (ter (X) Y Z)
23:   (is-det X))
24: ((verb-phrase (VP
25:        X
26:        Y) Z x)
27:   (trans-verb X Z y)
28:   (noun-phrase Y y x))
29: ((verb-phrase (VP
30:        X) Y Z)
31:   (intrans-verb X Y Z))
32: ((noun-phrase (NP
33:        X
34:        Y) Z x)
35:   (determiner X Z y)
36:   (noun Y y x))
37: ((sentence (S
38:        X
39:        Y) Z x)
40:   (noun-phrase X Z y)
41:   (verb-phrase Y y x))
42: ((sample of context-free grammer from T . Winograd (1972)))
43: ((dict sample))
44: ((dict sentence))
45: ((dict noun-phrase))
46: ((dict verb-phrase))
47: ((dict determiner))
48: ((dict noun))
49: ((dict intrans-verb))
50: ((dict trans-verb))
51: ((dict is-det))
52: ((dict is-noun))
53: ((dict is-intrans))
54: ((dict is-trans))
55: ((dict ter))
```

図 5 出力された Prolog ファイル PARSER 1

Fig. 5 Prolog file PARSER 1 of example 1

4. 応用例 2——GRAMMAR 2

Pereira と Warren[2] からとった例で，主語と述語の数の一致に関する文脈依存関係を含んだ事例が GRAMMAR 2 である．引数 z は singular. あるいは plural を値とするパラメタである．

図6に DCG ファイル GRAMMAR 2 を示す．

```
 1: ({{Context dependency rules in DCG, in Pereira & Warren(1980)}})
 2: ( (sentence (s x y))->(noun-phrase z x),(verb-phrase z y) )
 3: ((noun-phrase z (np x1 x2 x3))->(determiner z x1),(noun z x2),
 4:                                  (rel-clause z x3))
 5: ((noun-phrase singular (np x))->(name x))
 6: ((verb-phrase z (vp x y))->
 7:    (trans-verb z x),(noun-phrase z1 y))
 8: ((verb-phrase z (vp x))->
 9:    (intrans-verb z x))
10: ((rel-clause z (rel (that x)))->(ter(that)),(verb-phrase z x))
11: ((rel-clause z (rel (()) ))->(ter(()) )
12: ((determiner z (det x))->(ter(x)),{{is-determiner x z}} )
13: ((determiner plural (det (()) ))->(ter (()) )
14: ((noun z (n x))->(ter(y)),{{is-noun y z x}})
15: ((name (name y))->(ter(y)),{{is-name y}})
16: ((trans-verb z (tv x))->(ter(y)),{{is-trans y z x}})
17: ((intrans-verb z (iv x))->(ter(y)),{{is-intrans y z x}})
18: ({(ter x y z),(append x z y)})
19: ({{append () x x)})
20: ({{append (x|X) Y (x|Z)),(append X Y Z)}})
21: ({{is-determiner every singular)}})
22: ({{is-determiner all plural)}})
23: ({{is-determiner that singular)}})
24: ({{is-noun men plural man)}})
25: ({{is-noun man singular man)}})
26: ({{is-noun pianos plural piano)}})
27: ({{is-noun tunes plural tune)}})
28: ({{is-name Mary)}})
29: ({{is-trans likes singular like)}})
30: ({{is-intrans lives singular live)}})
31: ({{is-trans like plural like)}})
32: ({{is-trans whistles singular whistle)}})
33: ({{is-trans tunes singular tune)}})
34: ({{is-intrans whistles singular whistle)}})
35: ({{is-intrans live plural live)}})
```

図 6　応用例2の DCG ファイル GRAMMAR 2

Fig. 6　DCG file GRAMMAR 2 of example 2

図7に応用例1と同様に GRAMMAR 2 から PARSER 2 へのバッチ・モードの変換と，PARSER 2 による構文解析の結果を示している．この例では，「tunes」と「whistles」の品詞決定が Prolog のバック・トラッキングで処理されている点に注意されたい．出力された PARSER 2 のファイルは長いので省略する．

```
LOAD DCG
&.convert(GRAMMER2 PARSER2)
Converting DCG file GRAMMER2 to Prolog file PARSER2
* * * Conversion completed * * *
&.QT.


A>LOAD PARSER2
&.?((sentence x (every man that lives likes Mary) ())(PP x)PP)
(s
  (np (det
      every) (n
      man) (rel
      (that (vp
          (iv live)))))
  (vp (tv
      like) (np
      (name Mary))))
```

```
&.?((sentence x (every man likes Mary) ())(P x)PP)
(s (np (det every) (n man) (rel (()))) (vp (tv like) (np (name Mary))))


&.?((sentence x (every man that likes Mary lives) ())(PP x)PP)
(s
  (np (det
      every) (n
      man) (rel
      (that (vp
          (tv like)
          (np (name
              Mary))))))
  (vp (iv
      live)))


&.?((sentence x (that man that whistles tunes pianos)())(PP x)PP)
(s
  (np (det
      that) (n
      man) (rel
      (that (vp
          (iv whistle)))))
  (vp (tv
      tune) (np
      (det (()))
      (n piano)
      (rel (()))))))


&.?((sentence x (that man that whistles tunes likes pianos)())(PP x)PP)
(s
  (np (det
      that) (n
      man) (rel
      (that (vp
          (tv whistle)
          (np (det
              (())) (n
              tune) (rel
              (()))))))))
  (vp (tv
      like) (np
      (det (()))
      (n piano)
      (rel (()))))))


&.?((sentence x (every man that whistles whistles tunes)())(PP x)PP)
(s
  (np (det
      every) (n
      man) (rel
      (that (vp
          (iv whistle)))))
  (vp (tv
      whistle) (np
      (det (()))
      (n tune)
      (rel (()))))))


&.?((sentence x (every man that tunes tunes whistles)())(PP x)PP)
(s
  (np (det
      every) (n
      man) (rel
      (that (vp
          (tv tune)
          (np (det
              (())) (n
              tune) (rel
              (()))))))))
  (vp (iv
      whistle)))
```

図 7　応用例 2 の実行例

Fig. 7　Sample results of example 2

```
 1: {{(The DCG Proper for the ATN example from Woods, ibid(1980))}}
 2: ((sentence x)->
 3:       (ter (y)),{(aux-verb y x1 x2)},
 4:       (noun-phrase x3),
 5:       (rest-sentence q x3 x1 x2 x))
 6: ((sentence x)->
 7:       (noun-phrase x3),
 8:       (ter (y)),{(verb y x1 x2)},
 9:       (rest-sentence dcl x3 x1 x2 x))
10: ((rest-sentence X x3 x1 x2 (s X Y (tns x21) Z))->
11:       (rest-verb x1 x2 x11 x21),
12:       {(verbtype x11 z1)},
13:       (complement z1 x11 x3 Y Z))
14: ((rest-verb have x2 x1 (x2 perfect))->
15:       (ter(y)),{(past-participle y x1)})
16: ((rest-verb x1 x2 x1 x2)->(ter()))
17: ((complement copula be X1 X2 (vp (v X3) X4))->
18:       (ter(y)),{(past-participle y X3),(transitive X3)},
19:       (rest-object X1 X3 X4),
20:       (agent X2))
21: ((complement transitive X1 X2 X2 (vp (v X1) X3))->
22:       (noun-phrase y),
23:       (rest-object y X1 X3))
24: ((complement intransitive X1 X2 X2 (vp (v X1)))->(ter()))
25: ((rest-object x y z)->
26:       {(s-transitive y)},
27:       (ter(to y1)),{(infinitive y1)},
28:       (rest-sentence dcl x y1 present z))
29: ((rest-object x y x)->(ter()))
30: ((agent x)->(ter(by)),(noun-phrase x))
31: ((agent (np (pro someone)))->(ter()))
32: ((noun-phrase (np x1 (adj x2) (n x3)))->
33:       (ter(x1)),{(determiner x1)},
34:       (adjectives x2),
35:       (ter(x3)),{(noun x3)})
36: ((noun-phrase (np (npr x)))->(ter(x)),{(proper-noun x)})
37: ((adjectives (x|X))->
38:       (ter(x)),{(adjective x)},
39:       (adjectives X))
40: ((adjectives ())->(ter()))
41: (((ter x y z),(append x z y)))
42: (((append () x x)))
43: (((append (x|X) Y (x|Z)),(append X Y Z)))
44: (((aux-verb x y z),(verb x y z),(auxiliary y)))
45: (((auxiliary be)))
46: (((verb is be present)))
47: (((verb shot shoot past)))
48: (((verb was be past)))
49: (((verb told tell past)))
50: (((proper-noun John)))
51: (((proper-noun Dave)))
52: (((proper-noun Fred)))
53: (((proper-noun Mary)))
54: (((determiner the)))
55: (((adjective nice)))
56: (((noun book)))
57: (((verbtype be copula)))
58: (((verbtype x transitive),(transitive x)))
59: (((verbtype x intransitive),(intransitive x)))
60: (((transitive shoot)))
61: (((transitive buy)))
62: (((transitive like)))
63: (((transitive tell)))
64: (((transitive believe)))
65: (((intransitive sleep)))
66: (((s-transitive believe)))
67: (((s-transitive tell)))
68: (((infinitive be)))
69: (((infinitive have)))
70: (((infinitive tell)))
71: (((infinitive buy)))
72: (((infinitive shoot)))
73: (((past-participle been be)))
74: (((past-participle liked like)))
75: (((past-participle told tell)))
76: (((past-participle shot shoot)))
77: (((past-participle believed believe)))
```

図 8 応用例 3 DCG ファイル GRAMMAR 3

Fig. 8 DCG file GRAMMAR 3 of example 3

5. 応用例 3——GRAMMAR 3

Woods の ATN[12] と DCG を比較する目的で，Pereira と Warren[2] が記述した例である．

図8にその DCG ファイル GRAMMAR 3 を示す．図9には上の応用例と同様に GRAMMAR 3 から PARSER 3 へのバッチ・モードの変換と，PARSER 3 による構文解析の例が示してある．PARSER 3 への入力文は Pereira と Warren の例文である．

構文解析の第5例は曖昧さが存在する場合であって，述語 FAIL を用いて2通りの構文木を出力している．すなわち，

?((sentence x (John was belived to have been shot by Fred) ( )) (PP x) PP FAIL)

によってすべての構文木を求めているが，二つの構文木が得られている．

第1の構文木は，

Someone believed Fred has shot John.

と解析したケースであり，第2の構文木は，

Fred believed someone has shot John

と解析したケースである．

なお，変換行の PARSER 3 ファイルは長いので省略する．

```
LOAD DCG
&.convert(GRAMMER3 PARSER3)
Converting DCG file GRAMMER3 to Prolog file PARSER3
* * * Conversion completed * * *
&.QT.

A>LOAD PARSER3
&.?((sentence x (Fred shot John) ())(P x)PP)
(s dcl (np (npr Fred)) (tns past) (vp (v shoot) (np (npr John))))

&.?((sentence x (Mary was liked by John) ())(P x)PP)
(s dcl (np (npr John)) (tns past) (vp (v like) (np (npr Mary))))

&.?((sentence x (Fred told Mary to shoot John)())(PP x)PP)
(s
  dcl
  (np (npr
      Fred))
  (tns past)
  (vp (v
      tell) (s
      dcl
      (np (npr
          Mary))
      (tns present)
      (vp (v
          shoot) (np
          (npr John))))))


&.?((sentence x (John was believed to have been shot by Fred)())(PP x)PP FAIL)
(s
  dcl
  (np (pro
      someone))
  (tns past)
  (vp (v
      believe) (s
      dcl
      (np (npr
          Fred))
      (tns (present
          perfect))
      (vp (v
          shoot) (np
          (npr John))))))
```

```
(s
  dcl
  (np (npr
      Fred))
  (tns past)
  (vp (v
      believe) (s
      dcl
      (np (pro
          someone))
      (tns (present
          perfect))
      (vp (v
          shoot) (np
          (npr John))))))


&.?((sentence x (was Dave believed to have told Mary to tell Fred to buy
3.                              the book by John)())(PP x)PP)
(s
  q
  (np (npr
      John))
  (tns past)
  (vp (v
      believe) (s
      dcl
      (np (npr
          Dave))
      (tns (present
          perfect))
      (vp (v
          tell) (s
          dcl
          (np (npr
              Mary))
          (tns present)
          (vp (v
              tell) (s
              dcl
              (np (npr
                  Fred))
              (tns present)
              (vp (v
                  buy) (np
                  the
                  (adj ())
                  (n book)))))))))
```

図 9 応用例 3 の実行

Fig. 9 Sample results of example 3

6. 応用例 4——GRAMMAR 4

上述の応用例 1 を用いて，上昇形パーサを記述した例である．上昇形パーサを生成するには，適当な引数によって根 (root) に解析情報を渡すようにして葉 (leaves) を左から右へ resolve すればよい．すなわち，松本と田中ら[5]の BUP (bottom up parsing) の原理と同様に，文法則：

(cat 1 param 1)→(cat 2 param 2), (cat 3 param 3), …, (cat N param N)

を，

(cat 2 Z param 2 X)→(goal cat 3 param 3), …, (goal cat N param N),
(cat 1 Z param 1 X)

へ変え，終端則：

(cati parami)→(ter (a))

を，

({(cat cati parami (a|X) X)})

へ変える．それを同時に，

```
 1: (((bottom-up parser for sample grammer from T. Winograd(1972))))
 2: ((noun-phrase Z x X)->(goal verb-phrase y),(sentence Z (S x y) X))
 3: ((determiner Z x X)->(goal noun y),(noun-phrase Z (NP x y) X))
 4: ((trans-verb Z x X)->(goal noun-phrase y),(verb-phrase Z (VP x y) X))
 5: ((intrans-verb Z x X)->(verb-phrase Z (VP x) X))
 6:
 7: (((cat determiner (DET the) (the|X) X)))
 8: (((cat noun (N giraffe) (giraffe|X) X)))
 9: (((cat noun (N apple) (apple|X) X)))
10: (((cat intrans-verb (IV dream) (dreams|X) X)))
11: (((cat trans-verb (TV eat) (eats|X) X)))
12:
13: (((goal X Y x z),(cat Z Y1 x y),(eval(Z X Y1 Y y z))))
14: (((evsl x),x))
15:
16: (((sentence sentence x x y y)))
17: (((noun-phrase noun-phrase x x y y)))
18: (((verb-phrase verb-phrase x x y y)))
19: (((determiner determiner x x y y)))
20: (((noun noun x x y y)))
21: (((intrans-verb intrans-verb x x y y)))
22: (((trans-verb trans-verb x x y y)))
```

図 10　応用例 4

Fig. 10　GRAMMAR of example 4

```
LOAD DCG
&.convert(GRAMMER4 PARSER4)
Converting DCG file GRAMMER4 to Prolog file PARSER4
* * * Conversion completed * * *
&.QT.


A>LOAD PARSER4
&.?((goal sentence x (the giraffe dreams) ())(P x)PP)
(S (NP (DET the) (N giraffe)) (VP (IV dream)))


&.?((goal sentence x (the giraffe eats the apple) ())(P x)PP)
(S (NP (DET the) (N giraffe)) (VP (TV eat) (NP (DET the) (N apple))))
```

図 11　応用例 4 の実行

Fig. 11　Sample results of example 4

{(goal X Y x z), (cat Z Y1 x y), (eval (Z X Y1 Y y z))

{(eval x), x} (注)

と，すべての cat について，

{(cat cat x x y y)}

を付加すればよい．この場合には，終端記号の connect プログラムはいらない．

GRAMMAR 1 をこのように書き直すと図 10 の GRAMMAR 4 が得られる．

図 11 に GRAMMAR 4 から PARSER 4 への変換と PARSER 4 による構文解析の実行例を示す．

図 12 は DCG によって変換された Prolog のプログラム PARSER 4 のリストである．

また，図 13 は，簡単な文：

the apple dreams

の上昇形構文解析の過程の追跡 (Trace) を示している．実際には，上昇形パーサは DCG で書かないで，それ用のトランスレータを用いるが，それについてはここでは述べない．

(注)　メタ変数 x を利用している．

```
 1: ((eval X)
 2:   X)
 3: ((cat determiner (DET
 4:       the) (the|X) X))
 5: ((cat noun (N
 6:       giraffe) (giraffe|X) X))
 7: ((cat noun (N
 8:       apple) (apple|X) X))
 9: ((cat intrans-verb (IV
10:       dream) (dreams|X) X))
11: ((cat trans-verb (TV
12:       eat) (eats|X) X))
13: ((intrans-verb X Y Z x y)
14:   (verb-phrase X (VP
15:       Y) Z x y))
16: ((intrans-verb intrans-verb X X Y Y))
17: ((trans-verb X Y Z x y)
18:   (goal noun-phrase z x X1)
19:   (verb-phrase X (VP
20:       Y
21:       z) Z X1 y))
22: ((trans-verb trans-verb X X Y Y))
23: ((noun noun X X Y Y))
24: ((determiner X Y Z x y)
25:   (goal noun z x X1)
26:   (noun-phrase X (NP
27:       Y
28:       z) Z X1 y))
29: ((determiner determiner X X Y Y))
30: ((sentence sentence X X Y Y))
31: ((verb-phrase verb-phrase X X Y Y))
32: ((goal X Y Z x)
33:   (cat y z Z X1)
34:   (eval (y
35:       X
36:       z
37:       Y
38:       X1
39:       x)))
40: ((noun-phrase X Y Z x y)
41:   (goal verb-phrase z x X1)
42:   (sentence X (S
43:       Y
44:       z) Z X1 y))
45: ((noun-phrase noun-phrase X X Y Y))
46: ((bottom-up parser for sample grammer from T . Winograd (1972)))
47: ((dict bottom-up))
48: ((dict noun-phrase))
49: ((dict determiner))
50: ((dict trans-verb))
51: ((dict intrans-verb))
52: ((dict cat))
53: ((dict goal))
54: ((dict eval))
55: ((dict sentence))
56: ((dict verb-phrase))
57: ((dict noun))
```

図 12 応用例 4 の Prologプログラム PARSER 4

Fig. 12 Prolog file PARSER 4 of example 4

```
&.LOAD TRACE
&.??((goal sentence x (the apple dreams)() ))
ENTER (goal sentence X (the apple dreams) ()).C
ENTER (cat X Y (the apple dreams) Z).
.C
FINISH (cat determiner (DET the) (the apple dreams) (apple dreams))
ENTER (eval (determiner sentence (DET the) X (apple dreams) ())).C
ENTER (determiner sentence (DET the) X (apple dreams) ()).C
ENTER (goal noun X (apple dreams) Y).C
ENTER (cat X Y (apple dreams) Z).C
FINISH (cat noun (N apple) (apple dreams) (dreams))
ENTER (eval (noun noun (N apple) X (dreams) Y)).C
ENTER (noun noun (N apple) X (dreams) Y).C
FINISH (noun noun (N apple) (N apple) (dreams) (dreams))
FINISH (eval (noun noun (N apple) (N apple) (dreams) (dreams)))
FINISH (goal noun (N apple) (apple dreams) (dreams))
ENTER (noun-phrase sentence (NP (DET the) (N apple)) X (dreams) ()).C
ENTER (goal verb-phrase X (dreams) Y).C
ENTER (cat X Y (dreams) Z).C
FINISH (cat intrans-verb (IV dream) (dreams) ())
```

```
ENTER (eval (intrans-verb verb-phrase (IV dream) X () Y)).C
ENTER (intrans-verb verb-phrase (IV dream) X () Y).C
ENTER (verb-phrase verb-phrase (VP (IV dream)) X () Y).C
FINISH (verb-phrase verb-phrase (VP (IV dream)) (VP (IV dream)) () ())
FINISH (intrans-verb verb-phrase (IV dream) (VP (IV dream)) () ())
FINISH (eval (intrans-verb verb-phrase (IV dream) (VP (IV dream)) () ()))
FINISH (goal verb-phrase (VP (IV dream)) (dreams) ())
ENTER (sentence sentence (S (NP (DET the) (N apple)) (VP (IV dream))) X () ()).C
FINISH (sentence sentence (S (NP (DET the) (N apple)) (VP (IV dream))) (S (NP (D
ET the) (N apple)) (VP (IV dream))) () ())
FINISH (noun-phrase sentence (NP (DET the) (N apple)) (S (NP (DET the) (N apple)
) (VP (IV dream))) (dreams) ())
FINISH (determiner sentence (DET the) (S (NP (DET the) (N apple)) (VP (IV dream)
)) (apple dreams) ())
FINISH (eval (determiner sentence (DET the) (S (NP (DET the) (N apple)) (VP (IV
dream))) (apple dreams) ()))
FINISH (goal sentence (S (NP (DET the) (N apple)) (VP (IV dream))) (the apple dr
eams) ())
&.QT.
```

図 13　PARSER 4 による上昇形解析

Fig. 13　Bottom up parsing example by PARSER 4

7. DCG トランスレータのリスト

移植の便宜のために，DCG トランスレータ・モジュールのリストを図 14 に示す．トランスレータの中心はプログラム (translate) である．28 行～64 行のプログラムは 2 節に述べた DCG の構文則と対応している．1～5 行と 65 行目はモジュールの頭と尾である．6～27 行目はコマンド処理である．

```
 1: DCG-translator
 2: ((add convert load save delete kill)
 3: (dict -> [ ] , ->[ ,[ ],)
 4: ((definite clause grammer translator version 1.02))
 5: ((written by S. Yamada 16-May-1982 ))
 6: ((add x)
 7:     (NUM x)/(R y)(add1 x y))
 8: ((add y)
 9:     (add1 32767 y))
10: ((add1 x y)
11:     (translate y ((y1|y2)|y3))(declare y1)(ADDCL ((y1|y2)|y3) x))
12: ((convert (x y))(P Converting DCG file x to Prolog file y)PP
13:     (OPEN x)(process x)(CLOSE x)(SAVE y))
14: ((process x)
15:     (READ x z)(add z)(process x))
16: ((process x)(P * * * Conversion completed * * *)PP)
17: ((load x)(LOAD x))
18: ((save x)(SAVE x))
19: ((delete (x|y))/
20:     (translate (x|y) z)
21:     (OR ((DELCL z))((PP No such rule)) ))
22: ((delete x)
23:     (CON x)(R y)(IF (DELCL x y)()((PP No such rule)) ))
24: ((kill x)(DELCL x 1)(kill x))
25: ((kill x)(P Rule x deleted)PP)
26: ((declare x)
27:     (OR ((CL ((dict x)) ))((ADDCL ((dict x)) )) ))
28: ((translate (x1 , x2~>|x3) (y1|y2))
29:     (heads x1 x2 X1 X y1)(body x3 X1 X y2))
30: ((translate (x1 ->|x2) (y1|y2) )
31:     (head x1 X1 X y1)(body x2 X1 X y2))
32: ((translate ( {|x} y)
33:     (cond-tail x X1 X y))
34: ((translate (x1 , x2 ->{|x3) (y1|y2) )
35:     (heads x1 x2 X1 X y1)(cond-tail x3 X1 X y2))
36: ((translate (x1 ->{|x2) (y1|y2) )
37:     (head x1 X1 X y1)(cond-tail x2 X1 X y2))
38: ((heads x1 (ter x2) X1 X y)
39:     (append x2 X z)(append x1 (X1 z) y))
40: ((head x1 X1 X y1)(append x1 (X1 X) y1))
41: ((body () X X () ))
42: ((body (x1 ,|x2) X1 X (y1|y2) )
43:     (item  x1 X1 X2 y1)(body x2 X2 X y2))
44: ((body (x1|()) X1 X (y1|()) )
45:     (item x1 X1 X y1))
```

```
46: ((body ((|x) X1 X y)
47:     (cond-tail x X1 X y))
48: ((body (x1 ,(|x2) X1 X (y1|y2))
49:     (item x1 X1 X2 y1)(cond-tail x2 X2 X y2))
50: ((cond-tail (x1 ,|x2) X1 X (x1|y2) )
51:     (cond-tail x2 X1 X y2))
52: ((cond-tail (x1 ) ,|x2) X1 X (x1|y2))
53:     (body x2 X1 X y2))
54: ((cond-tail (x1 ),|x2) X1 X (x1|y2))
55:     (body x2 X1 X y2))
56: ((cond-tail (x1 )) X X (x1) ))
57: ((item (OR x1 x2) X1 X2 y1)
58:     (body x1 X1 X2 z1)(body x2 X1 X2 z2)(append (OR) ((z1)(z2)) y1))
59: ((item (x1|x2) X1 X2 y1)
60:     (CON x1)(append (x1|x2) (X1 X2) y1))
61: ((item / X X /)/)
62: ((append () x x))
63: ((append (x|X) Y (x|Z))
64:     (append X Y Z))
65: CLMOD
```

図 14　DCG トランスレータ・モジュール
Fig. 14　DCG translator module

8. おわりに

パーソナル・コンピュータの利用環境における自然言語利用の実験の中から，有用と認められる DCG を取り上げて紹介した．実験からはその他いくつかの結論を導くこともできるが，ここではそれを述べなかった．ここに述べた DCG 等を使用し，好ましいパーソナル・コンピューティングの環境が構築されることを望みたい．最後に，マルセイユの論理プログラミング国際会議への出席や論理プログラミングの実験に配慮いただいた山崎利治氏に感謝する．

参考文献
[1] K.L. Clark, F. McCabe, *Micro Prolog Reference Manual*, Logic Programming Associates.
[2] F.C.N. Pereira, D.H.D. Warren, "Definite Clause Grammars for Language Analysis ——A Survey of the Formalism and a Comparison with Augmented Transition Networks", *Artificial Intelligence*, No. 13, 1980, pp. 231〜278.
[3] A. Colmerauer, "Metamorphosis Grammars", *Natural Language Communication with Computers*, Lecture Notes in Computer Science, No. 63, Springer, 1978, pp. 133〜189.
[4] R.A. Kowalski, *Logic for Problem Solving*, Artificial Intelligence Series, North-Holland, 1979.
[5] 松本裕治，田中穂積，平川秀樹，三吉秀夫，安川秀樹，向井国昭，横井俊夫，"プロローグに埋め込まれたボトムアップ・パーサ：BUP", *Proc. The Logic Programming Conference '83 March*, ICOT, 1983.
[6] S-Å Tärnlund, "Horn clause computability", *BIT*, Bind 17, 1977, pp. 215〜226.
[7] J. Šebelík, P. Štěpánek, "Horn clause programs for recursive functions", in K.L. Clark and S-Å Tärnlund eds., *LOGIC PROGRAMMING*, Academic Press, 1982.
[8] L.M. Pereira, et al., "User's Guide to DEC system-10 Prolog", Laboratorio Nacional de Engenharia Civil, Lisbon, 1978.
[9] R.S. Wallace, "An Easy Implementation of PiL", *ACM SIGART Newsletter* No. 85, July 1983, pp. 29-32.
[10] T. Winograd, *Understanding Natural Language*, Academic Press, 1972.
(淵一博，田村浩一郎，白井良明訳，「言語理解の構造」，産業図書，1976)
[11] W.F. Clocksin, C.S. Mellish, *Programming in Prolog*, Springer, 1981, p. 204.
[12] W.A. Woods, "Transition network grammars for natural language analysis", *Comm. ACM* 13, October 1970.

執筆者紹介 山 田 真 市 (Shinichi Yamada)
昭和12年生，34年東京大学理学部数学科卒業．同年日本ユニバック(株)入社，45年 Harvard 大学大学院マスタ課程修了．S.M.（応用数学），プログラミング等に従事．現在，技術企画部所属．日本数学会，AMS，ASL，IEEE，ACM 会員．

報告 陰線消去プログラム HLE

Hidden Line Elimination Program

平林 繁

要 約 HLE (Hidden Line Elimination Program) は，高精度プロッタ用に開発された陰線消去プログラムである．

複雑な構造物に対しても演算時間の極端な増加を防ぐため，データ型に階層構造をとり入れ，さらに3次元空間内の構成要素相互の位置関係を記述する PREDICATE データを付加した．

対象物は，多角形，折れ線分の集合として表現されるが，凸多角形以外にも，凹多角形，複数の穴付き多角形，折れ線分による多角形上の模様なども効率的に処理できる．

陰線消去は正規空間で実行され，EXTENT 値，PREDICATE データにより，最終的な線分の可視・不可視決定まで持ち込まれる多角形，折れ線分の組をできるだけ少なくするように努めた．

また入力データの作成を容易にするため，多角形同士の交差により発生する線分の切断，交線の創成は内部で処理している．処理結果は，平面図（平行射影），パース図（透視射影）として出力される．

本稿では HLE のデータ構造，アルゴリズムの概要を述べる．

Abstract HLE is a hidden line elimination program developed for high-precision plotters.

For very complicated structures, HLE adopts hierarchical data type and also introduces PREDICATE data which describe mutual location of components in three dimensional space, so that extreme increase in computation time can be prevented.

Objects are a set of poly-line segments and polygons which include convex polygons, concave polygons, polygons with several holes. Figures on a polygon can be processed efficiently.

Hidden line elimination is performed in canonical space. HLE makes use of EXTENTs and PREDICATE data to reduce number of pairs of polygons and poly-lines whose visibility should be determined.

And this program internally performed cutting of a line segment and generation of intersecting line segments by intersection of a polygon with the other one to simplify pre-processing.

Plane figures (parallel projection) or perspective figures (perspective projection) are created through execution of this program.

This paper shows the outlines of data structure and algorithms which are employed in HLE.

1. はじめに

3次元空間内の物体をグラフィック・ディスプレイ，あるいはプロッタを使用して表示しようとするとき，視点方向から見て他の面によって隠されている線や面を消去する技術は，コンピュータによる図形処理の中でも，最も基礎的なものの一つであり，すでに数多くの研究が報告され処理プログラムも実用化されている．

とくに，近年コンピュータの能力が著しく向上し，同時に CAD/CAM 分野への関心が高まる中で，コンピュータが扱う図形も一段と複雑さを増して，陰線（面）消去されていない図面を判読することは不可能に近い．この点からも陰線（面）消去は，必要欠くべからざる技術といえる．

陰線消去を物体が定義されている座標系（ワールド座標系）で行うのは一般に得策でなく，射影変換によって視点座標系，正規（透視）座標系，あるいは3次元スクリーン座標

系へ変換して処理するのが効率的である．その際，平行射影変換をすれば平面図が得られ，透視射影変換をすれば透視図（パース図）が得られる．視点座標系を使用すると処理が3次元となるため一般には，さらに座標変換を行う．すなわち，視点を無限大に変更して（透視変換の場合）対象物を正規空間（$-1 \leqq x, y \leqq 1, 0 \leqq z \leqq 1$，左手座標系）へ変換する．この空間では，視線が$z$軸と平行となるため，投影面上での交点における$z$値の比較のみで，どちらが視点に近いかを決定できるという利点がある．

正規空間における陰線消去のアルゴリズムは，通常3次元物体を構成する面を一つずつ取り上げ，他の面によって隠されるかどうかを決定していく手法が主となる．

一方，最近グラフィック・ディスプレイを出力装置として使用するケースが増大するにつれて，3次元スクリーン座標系における陰線消去も盛んとなり，一部ではハードウェアに組み込まれているものもある．グラフィック・ディスプレイにおける陰線消去の特徴は，ハードウェアの分解能以上の精度を必要としないため，ピクセルあるいはラスタ単位のアルゴリズムが可能となる点である．たとえば，①すべてのピクセルについて，最小のz値を与える面を決定するデプス・バッファ・アルゴリズム，②可視・不可視の判定ができるまで，ウィンドウを分割していく Warnock のアルゴリズム，③スキャンラインに着目し，可視・不可視の判定が可能となるまでスキャンラインをスパンに分割していく Watkins のアルゴリズム，などの興味深いアルゴリズムがある．

本稿では，当社で新たに開発した陰線消去プログラムについて述べる．本プログラムは，高精度プロッタへの出力を目的とするため，正規空間において陰線消去を実行する．

2. 射影変換

ワールド座標系で記述された対象物を正規空間へ射影変換する．射影変換には，大別して平行射影変換と透視射影変換の2種類があるが，射影変換行列が異なるのみで正規空間への変換後はまったく同等に陰線消去が可能である．

射影変換行列の作成には，ACM SIGGAPH 提案の標準グラフィック・システム CORE に準拠した VIEW VOLUME データを入力して使用する．

VIEW VOLUME は，View-Reference-Point (VRP), Center-Of-Projection (COP), View-

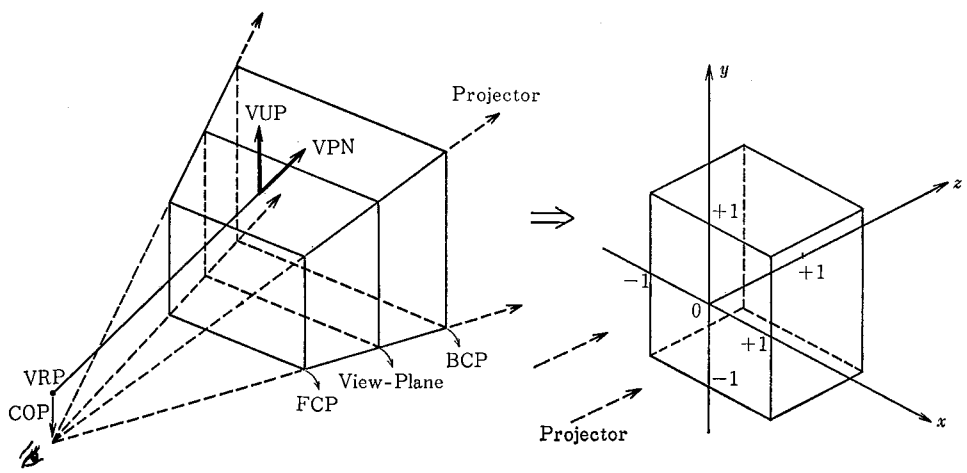


図 1 VIEW VOLUME と正規空間

Fig. 1 View volume and canonical view volume

Plane-Normal (VPN), View-Plane-Distance (VPD), View-Up-Vector (VUP), Window, Front-Clipping-Plane-Distance (FCPD), Back-Clipping-Plane-Distance (BCPD) から構成されるが，これを正規空間へ変換する詳細は成書[2] に譲る（図1）.

VIEW VOLUME データより作成された射影変換行列によって，入力された図形の座標データを変換していく．同時に各面の法線ベクトルも計算して，視点と逆方向を向く面（後方面）は処理から除く．この変換により，視点が $-\infty$ に置かれ，視線は z 軸と平行になるため投影面上の面と面との交点における前後関係は，両者の z 値を比較するだけで判定できる.

3. データ構造

3.1 データ型

本プログラムは，3次元空間内の物体を最終的にポリゴン（多角形）とポリライン（折れ線）の集合として表現している．またデータには階層構造をもたせ，3次元空間内の物体相互の位置関係を規定するデータが付加される.

WORLD=TITLE・COORDINATE・〈CLUSTER〉
CLUSTER=ID・TEXTURE・〈OBJECT〉
SHAPE (CLUSTER)= $\bigcup_{O \in C}$ (SHAPE (OBJECT))
OBJECT=ID・TEXTURE・PLANE-FORM | LINE-FORM
INDEPENDENT OBJECT=PLANE-FORM | LINE-FORM
NOT ON PLANE-FORM
DEPENDENT OBJECT=LINE-FORM ON PLANE-FORM
PLANE-FORM=VISIBILITY・OUTER-BOUNDARY・〈INNER-BOUNDARY〉
SHAPE (PLANE-FORM)=SHAPE (OUTER-BOUNDARY)
OUTER-BOUNDARY=CONCAVITY・LINE-FORM
INNER-BOUNDARY=CONCAVITY・LINE-FORM
SHAPE (LINE-FORM)=POLY-LINE

（・によって左辺の構成要素を，| は選択，〈 〉は複数回繰り返されることを示す）.

制約
- 外部境界・内部境界を構成する多角形は閉じていること.
- PLANE-FORM を構成する境界は，すべて同一平面上に存在すること.
- 内部境界は，外部境界の内部に存在しなければならない.
- 内部境界同士は互いに交差しないこと.

上に述べたデータ型の特徴を以下に列記する.

1) 3次元空間内の対象物 (WORLD) を，機能・構造面から複数個の部分集合 (CLUSTER) に分割する．さらに，各々の CLUSTER を構成する多角形（上では PLANE-FORM），折れ線（上では LINE-FORM）が OBJECT として定義される.

CLUSTER, OBJECT という二つの階層に分けることによって，単一階層のデータ構造より対象物の変更が容易に取り扱える．また，PREDICATE データとの併用で，プログラムの処理効率も向上する.

2) 多角形では，複数の穴付きも許され，多角形の外周を OUTER-BOUNDARY，穴の周囲を INNER-BOUNDARY と呼んでいる．処理対象には，凸・凹の多角形を含

む．凸凹性は CONCAVITY としてデータに付与され，多角形と直線の交点計算などで使用される．

3) OBJECT を構成する折れ線として，INDEPENDENT OBJECT と DEPENDENT OBJECT の2種類がある．DERENDENT OBJECT は多角形上の折れ線であり，面上の模様などを表現できる．

4) 入力時に，多角形の外周に対して，可視・不可視 (VISIBILITY) を指定できる．

5) 作図の際の情報として，線の太さ，種類，色などの属性 (TEXTURE)を指定できる．

3.2 PREDICATE

陰線消去を効率的に処理するには，対象となる物体のうち最終的に可視・不可視の決定へ持ち込む OBJECT の数をできるだけ少なくするのがよい．そのため，3次元空間内における CLUSTER, OBJECT 相互の位置関係をあらかじめ明確にしておくことが重要となる．この位置関係は，対象となる物体を CLUSTER, OBJECT という階層構造に分解した後，以下の PREDICATE によって把握する．

1) NOT HIDDEN (○ and ◎)

① SOP 1 (=SET-OF-POINTS) ○ SOP 2

→ Any points of SOP 1 is not hidden by SOP 2

② C 1 (=CLUSTER) ◎ C 2

→ SHAPE (C 1) ○ SHAPE (C 2)

C 1 は C 2 によって隠されない．

2) HIDDEN (? and ▣)

① A 1 (=OBJECT | CLUSTER) ? A 2

→ .NOT. (EXTENT (SHAPE (A 1)) ○ EXTENT (SHAPE (A 2)))

A 1 は A 2によって隠される可能性がある．

② C 1 (=CLUSTER) ▣ C 2

→ (SHAPE (C 1) ○ SHAPE (C 2)) .AND. (C 1 HIDES C 2 in the same plane)

C 1 は C 2 と同一平面上にあり，C 2 によって隠されない．

3) ORDER (<)

A 1 (=CLUSTER | OBJECT) <A 2

→ A 1<A 2 in the lexicographic order of (min (x), min (y), min (z)) of EXTENTs of shapes

EXTENT の最小値の順序で，A 1<A 2 の関係がある．

4) INTERSECTION (×)

A 1 (=CLUSTER | OBJECT) × A 2

→ (A 1 ? A 2) .AND. (A 2 ? A 1)

A 1 と A 2 は，各々相手によって隠される可能性がある．

5) INCLUSION (IN and ON)

① LF (=LINE-FORM) on PF (=PLANE-FORM)

→ SHAPE (LF) be inside of SHAPE (PF)

② O (=OBJECT) IN C (=CLUSTER)

→ O is an element in C

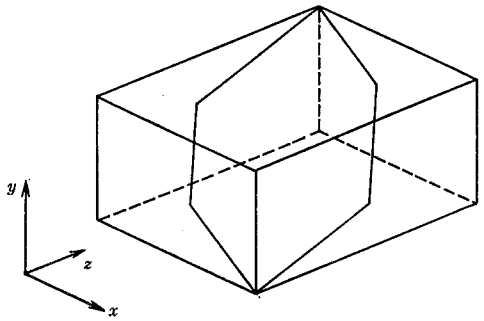


図 2　多角形を囲む EXTENT
Fig. 2　EXTENT for a polygon

以上の PREDICATE のうち (◎, 回, ON) については，PREDICATE データとして座標データに付加され，入力される．

実際に陰線処理を必要とするのは，(?, 回, ×) の PREDICATE で関連付けされた OBJECT の組のみである．

◎ の PREDICATE データにより，あらかじめ CLUSTER 同士の可視・不可視関係を，また回の PREDICATE データにより，同一平面上の CLUSTER に関して可視・不可視の指定を行うことができる．

3.3 EXTENT

3次元空間内で，各 CLUSTER, OBJECT の占有する概略の領域を求めておくことは，効率的に陰線処理をする上で重要である．

EXTENT は，CLUSTER, OBJECT を囲む最小の直六面体として定義される．六面体の各平面は，座標平面に平行である．

EXTENT 値として，六つの頂点のうち，(min (x), min (y), min (z)) と (max (x), max(y), max (z)) のデータが保持され，使用される (図 2)．

視点方向からみて CLUSTER, OBJECT 同士の重なりが生じ，その結果，陰線消去の処理が必要か否かの判断は，各 EXTENT の重なりによって行う．また交差処理，可視・不可視決定の過程で，線分と多角形の交点計算が必要となるが，あらかじめ EXTENT の比較により，交点の生じない場合を排除しておくのがよい．

以下に，PREDICATE, EXTENT を使用した基本的なアルゴリズムを示す．

```
for each C (=CLUSTER) do
    for each C 1 (=CLUSTER) SUCH THAT
        (C≦C 1) .AND.
        (((C 1 ? C) .AND. .NOT. (C 1 ◎ C)) .OR.
          ((C ? C 1) .AND. .NOT. (C ◎ C 1))) .OR.
          (C 回 C 1) .OR. (C 1 回 C) do
          ESTABLISH HIDDEN RELATION BETWEEN
          INDEPENDENT OBJECTS IN C AND THOSE IN C 1
          if ((C 1 ? C) .AND. .NOT. (C 1 ◎ C) .AND.
              (C ? C 1) .AND. .NOT. (C ◎ C 1))
```

```
                then for each PF (=PLANE-FORM) IN C
                    AND PF 1 IN C 1 SUCH THAT
                    (PF ? PF 1) . AND. (PF 1 ? PF) do
                    CUT LINE SEGMENTS OF EACH PLANE-FORM
                    AND GENERATE INTERSECTING LINES
            end
        end
    end
    for each C(=CLUSTER) do
        for each O(=INDEPENDENT OBJECT) do
            for each PF1(=PLANE-FORM) SUCH THAT
                (O ? PF 1) do
                DIVIDE EACH LINE SEGMENT OF SHAPE
                OF O OR EACH LINE-FORM ON O
                INTO SUBSEGMENTS HIDDEN OR NOT
                HIDDEN BY PF 1
            end
            CREATE THE IMAGE-OF-OBJECT FOR O
        end
    end
```

4. プログラムの構成

プログラム全体の構成を以下に示す.

Step 1

入力ファイルより，対象物の座標データ，VIEW VOLUME データ，PREDICATE データを読み出し必要なテーブルを作成する．その過程で VIEW VOLUME データから射影変換行列を作成して，座標データを 3 次元正規空間へ変換する．CLUSTER, OBJECT テーブルは，EXTENT の最小値をキー (key) にして分類される.

Step 2

視点方向からみて重なりが生じ，陰線消去処理が必要な OBJECT の組を選び出す．その際，最初に EXTENT 値と PREDICATE データから CLUSTER 同士の選択が行われ，つぎに所属する OBJECT 同士が EXTENT 値により選ばれて，OVERLAP テーブルに登録される.

また，互いに相手を隠す可能性のある組については交差の処理がされ，必要ならば線分の切断，交線の創成が行われる.

Step 3

前ステップで作成した OVERLAP テーブルをもとに，重なりの生じる可能性のある組のみ陰線消去処理が施され，各 OBJECT を構成する線分の可視・不可視部分が決定される.

処理結果は，投影面上の 2 次元データに可視・不可視フラッグが付与され出力ファイルへ書き込まれる.

5. 交 差 処 理

二つの OBJECT が相互に相手を隠す可能性のある場合の陰線消去処理が一番面倒である.

本プログラムでは，陰線処理に先立ち交差の可能性のある組について，特別に線分の切断，交線の創成を実施している.

交差平面において，線分を切断して独立の 2 本の線分にしておけば，多角形により線分が隠されるかどうかの判定の際，交点 1 点における z 値比較だけで，線分全体が多角形の前方にあるか後方にあるかを決定できる. 交差を許すと多角形の前方と後方とに線分が存在して後の処理が面倒になる.

ただし，新しい頂点の出現により，座標データが増加して入力時の領域に収容できず，それを拡張領域に納めて，ポインタでチェイニングをする必要がある. このため座標データが連続した領域に存在せず，幾分プログラムが繁雑となるのが欠点である.

さらに，多角形同士の交差では交線が発生する点も重要である.

交差による線分の切断と交線の創成を，二つの四辺形の交差により説明する（図 3）.

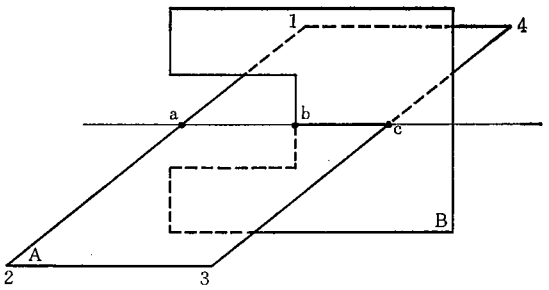


図 3　交差による線分の切断と交線の創成
Fig. 3　Line-cutting and line-generation by intersection

四辺形Aは，線分 (1-2) と (3-4) が，点 a, c で切断される. その結果，頂点四つが六つに増加する. Bを構成する線分も同様に切断される. 一方，線分 (b-c) が，交線として創成される. この線分は以後，一方の四辺形上の線分 (LINE-FORM ON PLANE-FORM) として処理される.

2 個の穴付き四辺形同士をずらして交差させた処理例を示す（図 4）.

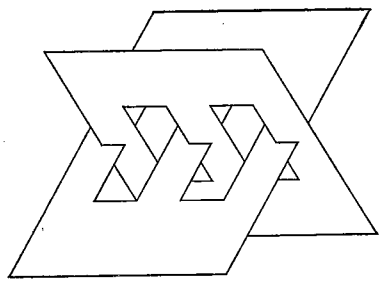


図 4　交差の処理例
Fig. 4　Example of intersection

6. 可視・不可視部分の決定

陰線処理の最も基本となる計算は，1個の多角形により1本の線分が隠され，線分のどの部分が可視となり，どの部分が不可視となるかを決定することである．

穴付き四辺形と線分の例で説明する（図5）．

四辺形Bは，外部境界と内部境界との合わせて8本の線分で構成され，各線分は一定向きに方向付けられる．線分Aと8本の線分との交点計算により，4個の交点が求まる．同時に線分の四辺形への出入が決定される．

$\vec{l}\times\vec{l_1}>0$　交点aで $\vec{l}$ は四辺形に入る．（左手座標系）

$\vec{l}\times\vec{l_3}<0$　交点bで $\vec{l}$ は四辺形から出る．

また交点は，座標データではなく線分Aのパラメタとして保持される(始点0，終点1)．

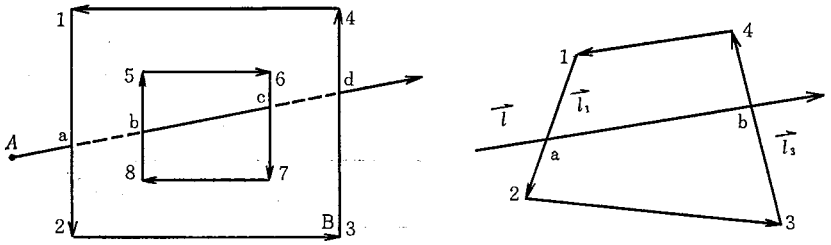


図 5　線分の可視・不可視

Fig. 5　Visibility of line segment

P_a	P_b	P_c	P_d	← パラメタ値
+	−	+	−	← $\vec{l}\times\vec{l_i}$

図 6　パラメタ・テーブル

Fig. 6　Parameter table

PARAMETER テーブルには，分類されたパラメタ値と多角形への出力が記録される(図6)．

交点のうち1点におけるz値を比較して，線分が四辺形の前方にあれば線分全体が可視となり，逆に後方にあれば，上のテーブルより，線分Aのうち (a-b) 区間と (c-d) 区間が不可視と判定する．

7. 陰 線 処 理

1個の OBJECT (Poly-line, Polygon) を構成する全線分と，それを隠す可能性のある OBJECT (Polygon) の間で陰線処理が行われ，各線分に関して PARAMETER テーブルが作成される．それを合わせて，当該 OBJECT の VISIBILITY テーブルが作られる(図7)．

OBJECT を構成する境界数（穴付きの場合は内部境界を含む），境界を構成する線分数，線分の交点数，その後に交点パラメタ値と可視・不可視フラッグが続く．

隠す可能性のある OBJECT を順番に取り上げ，まず PARAMETER テーブルを作成し，その結果を使って VISIBILITY テーブルを更新していく．

境界数
境界1を構成する線分数
線分1の交点数
0.0
0または1
P_1
0または1
P_2
0または1
線分2の交点数
⋮

図 7　可視・不可視テーブル
Fig. 7　Visibility table

関連する全 OBJECT について処理が終了すると，VISIBILITY テーブルのパラメタ値を座標データに変換して，可視・不可視フラッグを付加してファイルに出力する．

8. おわりに

最後に実際の処理例を示す（図8）．

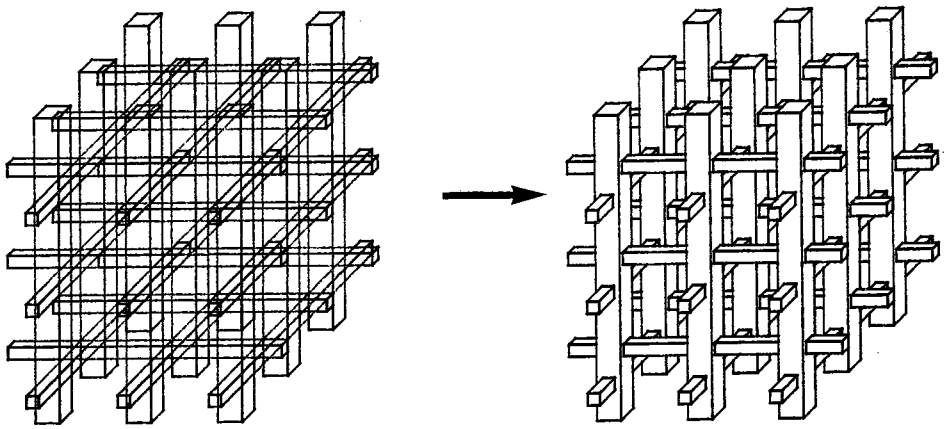


図 8　陰線処理の実際例
Fig. 8　Example of hidden line elimination

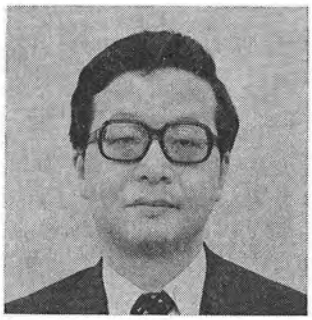
現在までの適用で，相当複雑な構造物に対してもその演算時間は，ほぼ満足すべき結果が得られている．

実際のプログラム開発に当たっては，交点計算における許容誤差，平行の処理，z 値による前後判定の際，許容範囲内でほぼ一致する場合の処理など，プログラミング上繁雑な処理も多かったことを付け加えておく．

本プログラム開発に当たって，仕様段階で参加された柳生孝昭氏をはじめとする関係者各位，開発中も貴重な助言をいただいた稲泉成彦氏に深く感謝致します．

参考文献　[1] 山口富士夫，「図形処理工学」，日刊工業新聞社，1981.
[2] J. D. Foley, A. V. Dam, *Foundamentals of Interactive Computer Graphics*, Addison-Wesley, 1982.

執筆者紹介 平　林　　繁 (Shigeru Hirabayashi)
昭和 23 年生．48 年東京大学理学部物理学科卒業，同年日本ユニバック(株)入社，技術計算関連プログラムの開発保守に従事．現在，応用ソフトウェア事業部技術計算グループに所属．



論文 浅い曲面板の弾性力学とその解析法

Elastostatics of Shallow Curved Plate and Its Analytical Methods

藤野 勉

要 約 本稿は浅い曲面板について，その弾性力学的挙動を記述する支配方程式を求め，その有限要素法ならびに任意節点配置差分法による解析法について研究を行ったものである．

Abstract This paper is concerned with a study of governing equation which describe the elastostatical behaviour of shallow curved plate and its analytical method by the finite element method (FEM) and the finite difference method (FDM) with arbitraly distributed nodes.

1. はじめに

一般に任意の形状をもつ曲面板について，その弾性力学的挙動を記述する支配方程式を求めることは困難である．しかし浅い曲面板についてはその曲率が小さいこと，すなわち平面板に近いことを利用して近似的に弾性方程式を導入することができる．このときベクトル的に力，モーメントの平衡条件式をつくることは非常に繁雑で誤りをおかしやすいが，ここに述べる Green 積分によるエネルギ法を用いれば，比較的容易にこの目的を達することができる．工学的には，浅い曲面板も自動車のボンネットなどのように直接利用の例があるが，実際には深い曲面板，すなわち任意形状のセル構造解析が興味の対象となることが多い．しかるに深い曲面板については，その弾性方程式を確立することは非常に困難である．たとえば，軸対称殻のように曲面が直交曲線座標系により創成されている場合はその限りではないが，任意形状の曲面板についてはほとんど不可能に近い作業と思われる．そのため一般に深い曲面板は多くの平面板または浅い曲面板の集まりとしてモデル化され，その応力解析が行われている．このとき平板よりも曲面板の集まりでモデル化する方が置換の精度が当然ながら良好である．このように浅い曲面板についてその弾性力学的特性を理解することは，さらに利用度の高い一般のシェル構造解析のための準備としても重要である．

なお，本稿で用いる符号は以下のとおりである．

E＝ヤング率 (kg cm^{-2})，$E^*=E/(1+\nu)(1-2\nu)$

$G=E/2(1+\nu)$＝剛性率 (kg cm^{-2})

ν＝ポアソン比，$\alpha=(1-\nu)/2$，$\beta=(1+\nu)/2$

$\lambda=(1-2\nu)/2$，標準変形波長

h＝板厚 (cm)

$k^*=E^*h$，$k=Eh/(1-\nu^2)$，$k'=Gh=\alpha k$ (kg cm^{-1})

$D^*=E^*h^3/12$，$D=Eh^3/12(1-\nu^2)$，$D'=Gh^3/12=\alpha D$ (kg cm)

f_{xx}，f_{xy}，f_{yy}＝曲率 (cm^{-1})

$\rho_{xx}=1/f_{xx}$，$\rho_{xy}=1/f_{xy}$，$\rho_{yy}=1/f_{yy}$＝曲率半径 (cm)

s，t，n＝曲線座標

$\boldsymbol{i}$，$\boldsymbol{j}$，$\boldsymbol{k}$＝s，t，n 方向の単位ベクトル

g_1，g_2，g_3＝s，t，n の測度係数

2. 浅い曲面板の微分幾何

3次元空間 x-y-z 座標系において，曲面が

$$z=f(x,y) \tag{2-1}$$

により定義されているものとする．ここで x-y 面に対する曲面の傾き $f_x=\partial f/\partial x, f_y=\partial f/\partial y$ の大きさが1に比べてかなり小さいとき，この曲面を浅い曲面と呼ぶこととする．つぎに浅い曲面板についてその弾性方程式を導入するため，まずその微分幾何学的特性について考察する．

2.1 位置ベクトル

浅い曲面上の点と原点を結ぶ位置ベクトルは，図1により

$$\boldsymbol{r}_0=\boldsymbol{i}_0x+\boldsymbol{j}_0y+\boldsymbol{k}_0z \tag{2-2}$$

と与えられる．ここに $\boldsymbol{i}_0, \boldsymbol{j}_0, \boldsymbol{k}_0$ は x, y, z 方向の単位ベクトルである．位置ベクトルの微分は式 (2-3) により与えられる．

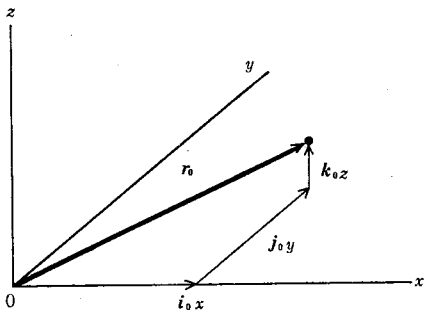


図 1 位置ベクトル
Fig. 1 Position vector

$$d\boldsymbol{r}_0=\boldsymbol{i}_0dx+\boldsymbol{j}_0dy+\boldsymbol{k}_0dz=(\boldsymbol{i}_0+f_x\boldsymbol{k}_0)dx+(\boldsymbol{j}_0+f_y\boldsymbol{k}_0)dy=\boldsymbol{i}ds+\boldsymbol{j}dt \tag{2-3}$$

ただし，s, t は曲面を x, y で切ったときの曲線座標であり，$\boldsymbol{i}, \boldsymbol{j}$ は s, t 方向の単位ベクトルである．

s, t 曲線座標の測度係数　$g_1=\sqrt{1+f_x^2}$, $g_2=\sqrt{1+f^2{}_y}$ (2-4)

s, t 方向単位ベクトル　$\boldsymbol{i}=(\boldsymbol{i}_0+f_x\boldsymbol{k}_0)/g_1$, $\boldsymbol{j}=(\boldsymbol{j}_0+f_y\boldsymbol{k}_0)/g_2$ (2-5)

s, t 方向微小線素　$ds=g_1dx$, $dt=g_2dy$ (2-6)

仮定により g_1, g_2 は1に近い数である．g_1, g_2 を展開して

$$g_1=1+0.5f_x{}^2+\cdots, \quad g_2=1+0.5f_y{}^2+\cdots$$

を得る．したがって，f_x, f_y の大きさが0.1の程度ならば，g_1, g_2 の1からの差は0.005の程度である．

つぎに単位ベクトル $\boldsymbol{i}, \boldsymbol{j}$ に直交して，単位ベクトル $\boldsymbol{k}$ を定める．

$$g_3\boldsymbol{k}=(g_1\boldsymbol{i})\times(g_2\boldsymbol{j})=(\boldsymbol{i}_0+f_x\boldsymbol{k}_0)\times(\boldsymbol{j}_0+f_y\boldsymbol{k}_0)=-f_x\boldsymbol{i}_0-f_y\boldsymbol{j}_0+\boldsymbol{k}_0 \tag{2-7}$$

$$g_3=\sqrt{1+f_x{}^2+f_y{}^2} \tag{2-8}$$

さらに，曲面を定義する式 (2-1) を中性面とし，単位ベクトル方向に曲線座標 n を定義するとき，曲面板上の一つの点の位置ベクトルは

$$\boldsymbol{r}=\boldsymbol{r}_0+n\boldsymbol{k} \tag{2-9}$$

$$d\boldsymbol{r}=\boldsymbol{i}ds+\boldsymbol{j}dt+\boldsymbol{k}dn \tag{2-10}$$

によって定められる．

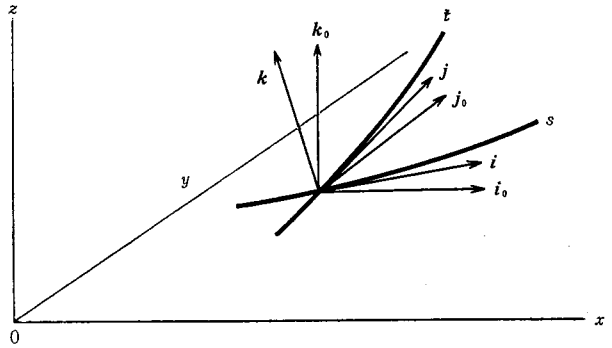


図 2 曲線座標ならびに単位ベクトル
Fig. 2 Curvilinear coordinate and unit vector

2.2 単位ベクトルの変換式

単位ベクトルの変換式は次のようになる.

$$\begin{Bmatrix} g_1\boldsymbol{i} \\ g_2\boldsymbol{j} \\ g_3\boldsymbol{k} \end{Bmatrix}=\begin{bmatrix} 1 & & f_x \\ & 1 & f_y \\ -f_x & -f_y & 1 \end{bmatrix}\begin{Bmatrix} \boldsymbol{i}_0 \\ \boldsymbol{j}_0 \\ \boldsymbol{k}_0 \end{Bmatrix} \quad \text{または} \quad \begin{Bmatrix} g_3^2\boldsymbol{i}_0 \\ g_3^2\boldsymbol{j}_0 \\ g_3^2\boldsymbol{k}_0 \end{Bmatrix}=\begin{bmatrix} g_2^2 & -f_xf_y & -f_x \\ -f_xf_y & g_1^2 & -f_y \\ f_x & f_y & 1 \end{bmatrix}\begin{Bmatrix} g_1\boldsymbol{i} \\ g_2\boldsymbol{j} \\ g_3\boldsymbol{k} \end{Bmatrix} \tag{2-11}$$

なお, 単位ベクトル $\boldsymbol{i},\boldsymbol{j}$ は完全な直交ではない.

$$\boldsymbol{i}\cdot\boldsymbol{j}=(\boldsymbol{i}_0+f_x\boldsymbol{k}_0)\cdot(\boldsymbol{j}_0+f_y\boldsymbol{k}_0)/g_1g_2=f_xf_y/g_1g_2\fallingdotseq f_xf_y\neq 0 \tag{2-12}$$

2.3 測度係数の微分

測度係数の微分は, 式 (2-13) のように表される.

$$\begin{aligned} &dg_1\fallingdotseq f_x(f_{xx}dx+f_{xy}dy),\quad dg_2\fallingdotseq f_y(f_{xy}dx+f_{yy}dy) \\ &dg_3\fallingdotseq(f_xf_{xx}+f_yf_{xy})dx+(f_xf_{xy}+f_yf_{yy})dy \end{aligned} \tag{2-13}$$

2.4 単位ベクトルの微分

単位ベクトルの微分は, 式 (2-14) のように表せる.

$$\begin{aligned} g_3^2g_1d\boldsymbol{i}&=(f_{xx}dx+f_{xy}dy)\{f_x(g_1-1)\boldsymbol{i}+f_yg_2\boldsymbol{j}+g_3\boldsymbol{k}\} \\ g_3^2g_2d\boldsymbol{j}&=(f_{xy}dx+f_{yy}dy)\{f_xg_1\boldsymbol{i}+f_y(g_2-1)\boldsymbol{j}+g_3\boldsymbol{k}\} \\ g_3^3d\boldsymbol{k}&=\{-(f_{xx}dx+f_{xy}dy)g_2^2+(f_{xy}dx+f_{yy}dy)f_xf_y\}g_1\boldsymbol{i} \\ &\quad+\{(f_{xx}dx+f_{xy}dy)f_xf_y-(f_{xy}dx+f_{yy}dy)g_1^2\}g_2\boldsymbol{j} \\ &\quad+[-\{(f_xf_{xx}+f_yf_{xy})dx+(f_xf_{xy}+f_yf_{yy})dy\}g_3 \\ &\quad+(f_{xx}dx+f_{xy}dy)f_x+(f_{xy}dx+f_{yy}dy)f_y]g_3\boldsymbol{k} \end{aligned} \tag{2-14}$$

式 (2-14) は小さな項を省略して, 次のように近似できる.

$$\begin{aligned} d\boldsymbol{i}&=(f_{xx}dx+f_{xy}dy)\boldsymbol{k} \\ d\boldsymbol{j}&=(f_{xy}dx+f_{yy}dy)\boldsymbol{k} \\ d\boldsymbol{k}&=-\{(f_{xx}dx+f_{xy}dy)\boldsymbol{i}+(f_{xy}dx+f_{yy}dy)\boldsymbol{j}\} \end{aligned} \tag{2-15}$$

なお, f_{xx}, f_{xy}, f_{yy} は曲面の曲率, その逆数 $\rho_{xx}=1/f_{xx}$, $\rho_{xy}=1/\rho_{xy}$, $\rho_{yy}=1/f_{yy}$ は曲率半径を表す. さらに, 微分式の大きさを評価するため, f_x, f_y の大きさを $O(f)$, 曲率半径の大きさを $O(\rho)$ とする. なお, x-y 軸が曲面の曲率主軸と一致するときは $f_{xy}=0$ である.

3. 浅い曲面板の変形，歪，応力

浅い曲面板の $s(\boldsymbol{i})$, $t(\boldsymbol{j})$, $n(\boldsymbol{k})$ 方向変位成分をそれぞれ u, v, w とし，これらはともに s, t, n の関数とし，n について次のように展開されるものとする．

$$\begin{aligned} u(s,t,n)&=u(s,t)+nu'(s,t)+n^2u''(s,t)+\cdots \\ v(s,t,n)&=v(s,t)+nv'(s,t)+n^2v''(s,t)+\cdots \\ w(s,t,n)&=w(s,t)+nw'(s,t)+n^2w''(s,t)+\cdots \end{aligned} \tag{3-1}$$

3.1 変位ベクトルならびにその微分

$$\boldsymbol{u}(s,t,n)=\boldsymbol{i}u(s,t,n)+\boldsymbol{j}v(s,t,n)+\boldsymbol{k}w(s,t,n) \tag{3-2}$$

$$\begin{aligned} d\boldsymbol{u}(s,t,n)=&\boldsymbol{i}\{u_sds+u_tdt+u'dn+n(u_s'ds+u_t'dt+2u''dn)+\cdots\} \\ &+\boldsymbol{j}\{v_sds+v_tdt+v'dn+n(v_s'ds+v_t'dt+2v''dn)+\cdots\} \\ &+\boldsymbol{k}\{w_sds+w_tdt+w'dn+n(w_s'ds+w_t'dt+2w''dn)+\cdots\} \\ &+(u+nu'+\cdots)(f_{xx}dx+f_{xy}dy)\boldsymbol{k} \\ &+(v+nv'+\cdots)(f_{xy}dx+f_{yy}dy)\boldsymbol{k} \\ &-(w+nw'+\cdots)\{f_{xx}dx+f_{xy}dy)\boldsymbol{i}+(f_{xy}dx+f_{yy}dy)\boldsymbol{j}\} \end{aligned} \tag{3-3}$$

上式は $O(f^2)$ の項を省略することにより s, t, n を x, y, z で置き換えることができ，次のように近似される．

$$\begin{aligned} d\boldsymbol{u}(s,t,n)=&\boldsymbol{i}[u_xdx+u_ydy+u'dz-w(f_{xx}dx+f_{xy}dy) \\ &+n\{u_x'dx+u_y'dy+2u''dz-w'(f_{xx}dx+f_{xy}dy)\}+\cdots] \\ &+\boldsymbol{j}[v_xdx+v_ydy+v'dz-w(f_{xy}dx+f_{yy}dy) \\ &+n\{v_x'dx+v_y'dy+2v''dz-w'(f_{xy}dx+f_{yy}dy)\}+\cdots] \\ &+\boldsymbol{k}[w_xdx+w_ydy+w'dz+u(f_{xx}dx+f_{xy}dy) \\ &+v(f_{xy}dx+f_{yy}dy)+n\{w_x'dx+w_y'dy+2w''dz \\ &+u'(f_{xx}dx+f_{xy}dy)+v'(f_{xy}dx+f_{yy}dy)\}+\cdots] \end{aligned} \tag{3-4}$$

式 (3-4) と dn を dz に置き換えた式 (2-10) により，下記の歪テンソルを導入することができる．

$$\begin{aligned} \varepsilon_{ss}&=u_x-f_{xx}w \\ \varepsilon_{tt}&=v_y-f_{yy}w \\ \varepsilon_{nn}&=w' \\ \varepsilon_{st}&=u_y+v_x-2f_{xy}w \\ \varepsilon_{sn}&=w_x+f_{xx}u+f_{xy}v+u' \\ \varepsilon_{tn}&=w_y+f_{xy}u+f_{yy}v+v' \end{aligned} \tag{3-5}$$

$$\begin{aligned} \varepsilon_{ss}'&=u_x'-f_{xx}w' \\ \varepsilon_{tt}'&=v'_y-f_{yy}w' \\ \varepsilon_{nn}'&=2w'' \\ \varepsilon_{st}'&=u_y'+v_x'-2f_{xy}w' \\ \varepsilon_{sn}'&=w_x'+f_{xx}u'+f_{xy}v'+2u'' \\ \varepsilon_{tn}'&=w_y'+f_{xy}u'+f_{yy}v'+2v'' \end{aligned} \tag{3-6}$$

s-t-n 座標は微視的には x-y-z 座標系と同様に直交座標系であるから，x-y-z 座標系による構成方程式をそのまま借用することが許される．

$$\begin{Bmatrix} \sigma_{ss} \\ \sigma_{tt} \\ \sigma_{nn} \\ \sigma_{st} \\ \sigma_{sn} \\ \sigma_{tn} \end{Bmatrix} = E^* \begin{bmatrix} 1-\nu & \nu & \nu & & & \\ \nu & 1-\nu & \nu & & & \\ \nu & \nu & 1-\nu & & & \\ & & & \lambda & & \\ & & & & \lambda & \\ & & & & & \lambda \end{bmatrix} \begin{Bmatrix} \varepsilon_{ss} \\ \varepsilon_{tt} \\ \varepsilon_{nn} \\ \varepsilon_{st} \\ \varepsilon_{sn} \\ \varepsilon_{tn} \end{Bmatrix} \tag{3-7}$$

σ'_{mn} についても同様である．

4. 浅い曲面板の変分原理，弾性方程式

上に述べたことにより浅い曲面板について歪，応力が求められ，つぎに述べる Green 積

分により変分原理，境界条件式，弾性方程式を導入することができる．このようなエネルギ法による数式化はベクトル法によるより求めやすく，誤りをおかす危険が少ない．

4.1 面内力，境界力ならびに内部エネルギ

内部エネルギをUとすると，その変分は次のように与えられる．

$$\begin{aligned}\delta U=&\iint_S\int_{-h/2}^{h/2}(\sigma_{mn}+n\sigma'_{mn}+\cdots)(\delta\varepsilon_{mn}+n\delta\varepsilon'_{mn}+\cdots)dsdtdn\\=&\iint_S\Big\{N_{ss}\Big(\frac{\partial\delta u}{\partial x}-f_{xx}\delta w\Big)+N_{tt}\Big(\frac{\partial\delta v}{\partial y}-f_{yy}\delta w\Big)+N_{nn}\delta w'\\&+N_{st}\Big(\frac{\partial\delta u}{\partial y}+\frac{\partial\delta v}{\partial x}-2f_{xy}\delta w\Big)+N_{sn}\Big(\frac{\partial\delta w}{\partial x}+f_{xx}\delta u+f_{xy}\delta v+\delta u'\Big)\\&+N_{tn}\Big(\frac{\partial\delta w}{\partial y}+f_{xy}\delta u+f_{yy}\delta v+\partial v'\Big)\\&+M_{ss}\Big(\frac{\partial\delta u'}{\partial x}-f_{xx}\delta w'\Big)+M_{tt}\Big(\frac{\partial\delta v'}{\partial y}-f_{yy}\delta w'\Big)+2M_n\delta w''\\&+M_{st}\Big(\frac{\partial\delta u'}{\partial y}+\frac{\partial\delta v'}{\partial x}-2f_{xy}\delta w'\Big)+M_{sn}\Big(\frac{\partial w'}{\partial x}+f_{xx}\delta u'+f_{xy}\delta v'+2\delta u''\Big)\\&+M_{tn}\Big(\frac{\partial\delta w'}{\partial y}+f_{xy}\delta u'+f_{yy}\delta v'+2\delta v''\Big)+\cdots\Big\}dxdy\\=&\int_s(N_s\delta u+N_t\delta v+N_n\delta w+M_s\delta u'+M_t\delta v'+M_n\delta w'+\cdots)ds\\&-\iint_S(L_s\delta u+L_t\delta v+L_n\delta w+L_s'\delta u'+L_t'\delta v'+L_n'\delta w'+\cdots\Big)dxdy\end{aligned}\tag{4-1}$$

ただし面内力，モーメントは次のようになる．

$$N_{ss}=\int_{-h/2}^{h/2}\sigma_{ss}dz=k^*\Big\{(1-\nu)\Big(\frac{\partial u}{\partial x}-f_{xx}w\Big)+\nu\Big(\frac{\partial v}{\partial y}-f_{yy}w\Big)+\nu w'\Big\}\tag{4-2}$$

$$N_{tt}=\int_{-h/2}^{h/2}\sigma_{tt}dn=k^*\Big\{\nu\Big(\frac{\partial u}{\partial x}-f_{xx}w\Big)+(1-\nu)\Big(\frac{\partial v}{\partial y}-f_{yy}w\Big)+\nu w'\Big\}\tag{4-3}$$

$$N_{nn}=\int_{-h/2}^{h/2}\sigma_{nn}dn=k^*\Big\{\nu\Big(\frac{\partial u}{\partial x}-f_{xx}w\Big)+\nu\Big(\frac{\partial v}{\partial y}-f_{yy}w\Big)+(1-\nu)w'\Big\}\tag{4-4}$$

$$N_{st}=\int_{-h/2}^{h/2}\sigma_{st}dn=k'\Big(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}-2f_{xy}w\Big)\tag{4-5}$$

$$N_{sn}=\int_{-h/2}^{h/2}\sigma_{sn}dn=k'\Big(\frac{\partial w}{\partial x}+f_{xx}u+f_{xy}v+u'\Big)\tag{4-6}$$

$$N_{tn}=\int_{-h/2}^{h/2}\sigma_{tn}dn=k'\Big(\frac{\partial w}{\partial y}f_{xy}u+f_{yy}v+v'\Big)\tag{4-7}$$

$$M_{ss}=\int_{-h/2}^{h/2}\sigma_{ss}n^2dn=D^*\Big\{(1-\nu)\Big(\frac{\partial u'}{\partial x}-f_{xx}w'\Big)+\nu\Big(\frac{\partial v'}{\partial y}-f_{yy}w'\Big)+2\nu w''\Big\}\tag{4-8}$$

$$M_{tt}=\int_{-h/2}^{h/2}\sigma_{tt}n^2dn=D^*\Big\{\nu\Big(\frac{\partial u'}{\partial x}-f_{xx}w'\Big)+(1-\nu)\Big(\frac{\partial v'}{\partial y}-f_{yy}w'\Big)+2\nu w''\Big\}\tag{4-9}$$

$$M_{nn}=\int_{-h/2}^{h/2}\sigma_{nn}n^2dn=D^*\Big\{\nu\Big(\frac{\partial u'}{\partial x}-f_{xx}w'\Big)+\nu\Big(\frac{\partial v'}{\partial y}-f_{yy}w'\Big)+2(1-\nu)w''\Big\}\tag{4-10}$$

$$M_{st}=\int_{-h/2}^{h/2}\sigma_{st}n^2dn=D'\Big(\frac{\partial u'}{\partial y}+\frac{\partial v'}{\partial x}-2f_{xy}w'\Big)\tag{4-11}$$

$$M_{sn}=\int_{-h/2}^{h/2}\sigma_{sn}n^2dn=D'\Big(\frac{\partial w'}{\partial x}+f_{xx}u'+f_{xy}v'+2u''\Big)\tag{4-12}$$

$$M_{tn}=\int_{-h/2}^{h/2}\sigma_{tn}n^2dn=D'\left(\frac{\partial w'}{\partial y}+f_{xy}u'+f_{yy}v'+2v''\right) \tag{4-13}$$

したがって，境界力およびモーメントは，

$$\begin{aligned}&N_s=l_xN_{ss}+l_yN_{st},\quad N_t=l_xN_{st}+l_yN_{tt},\quad N_n=l_xN_{sn}+l_yN_{tn}\\&M_s=l_xM_{ss}+l_yM_{st},\quad M_t=l_xM_{st}+l_yM_{tt},\quad M_n=l_xM_{sn}+l_yM_{tn}\end{aligned} \tag{4-14}$$

となる．ただし l_x, l_y は境界における外方法線の方向余弦で

$$l_x=\frac{dy}{ds}=\sin\theta,\qquad l_y=-\frac{dx}{ds}=-\cos\theta \tag{4-15}$$

を表す（図 3）．

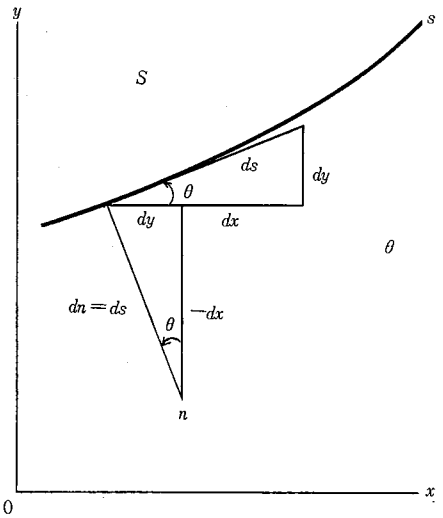


図 3　境界における法線の方向余弦

Fig. 3　Direction cosine of outward normal at boundary

境界力およびモーメントに関する微分式は次のようになる．

$$L_s=\frac{\partial N_{ss}}{\partial x}+\frac{\partial N_{st}}{\partial y}-f_{xx}N_{sn}-f_{xy}N_{tn} \tag{4-16}$$

$$L_t=\frac{\partial N_{st}}{\partial x}+\frac{\partial N_{tt}}{\partial y}-f_{xy}N_{sn}-f_{yy}N_{tn} \tag{4-17}$$

$$L_n=\frac{\partial N_{sn}}{\partial x}+\frac{\partial N_{tn}}{\partial y}+f_{xx}N_{ss}+2f_{xy}N_{st}+f_{yy}N_{tt} \tag{4-18}$$

$$L_s'=\frac{\partial M_{ss}}{\partial x}+\frac{\partial M_{st}}{\partial y}-f_{xx}M_{sn}-f_{xy}M_{tn}-N_{sn} \tag{4-19}$$

$$L_t'=\frac{\partial M_{st}}{\partial x}+\frac{\partial M_{tt}}{\partial y}-f_{xy}M_{sn}-f_{yy}M_{tn}-N_{tn} \tag{4-20}$$

$$L'_n=\frac{\partial M_{sn}}{\partial x}+\frac{\partial M_{tn}}{\partial y}+f_{xx}M_{ss}+2f_{xy}M_{st}+f_{yy}M_{tt}-N_{nn} \tag{4-21}$$

さらに，内部エネルギに関する積分式は次式のとおりである．

$$\begin{aligned}U=\frac{1}{2}\iint_S k^*\Big[&(1-\nu)\left\{\left(\frac{\partial u}{\partial x}-f_{xx}w\right)^2+\left(\frac{\partial v}{\partial y}-f_{yy}w\right)^2+(w')^2\right\}\\&+2\nu\left\{\left(\frac{\partial u}{\partial x}-f_{xx}w\right)\left(\frac{\partial v}{\partial y}-f_{yy}w\right)+\left(\frac{\partial u}{\partial x}-f_{xx}w\right)w'+\left(\frac{\partial v}{\partial y}-f_{yy}w\right)w'\right\}\end{aligned}$$

$$+\frac{1-2\nu}{2}\left\{\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}-2f_{xy}w\right)^2+\left(\frac{\partial w}{\partial x}+f_{xx}u+f_{xy}v+u'\right)^2\right.$$
$$\left.+\left(\frac{\partial w}{\partial y}+f_{xy}u+f_{yy}v+v'\right)^2\right\}+\cdots\Big]dxdy$$
$$+\frac{1}{2}\iint_S D^*\Big[(1-\nu)\left\{\left(\frac{\partial u'}{\partial x}-f_{xx}w'\right)^2+\left(\frac{\partial v'}{\partial y}-f_{xy}w'\right)^2+4(w'')^2\right\}$$
$$+2\nu\left\{\left(\frac{\partial u'}{\partial x}-f_{xx}w'\right)\left(\frac{\partial v'}{\partial y}-f_{xy}w'\right)+2\left(\frac{\partial u'}{\partial x}-f_{xx}w'\right)w''+2\left(\frac{\partial v'}{\partial y}-f_{yy}w'\right)w''\right\}$$
$$+\frac{1-2\nu}{2}\left\{\left(\frac{\partial u'}{\partial y}+\frac{\partial v'}{\partial x}-2f_{xy}w'\right)^2+\left(\frac{\partial w'}{\partial x}+f_{xx}u'+f_{xy}v'+2u''\right)^2\right.$$
$$\left.+\left(\frac{\partial w'}{\partial y}+f_{xy}u'+f_{yy}v'+2v''\right)^2\right\}+\cdots\Big]dxdy \qquad (4\text{-}22)$$

4.2 外力ならびに外部エネルギ

外力，モーメントは，曲面板の境界ならびに実体部に作用する境界はこれを分けて，変位，変角が拘束されている固定（幾何学的）境界 s_1 と，外力，モーメントが作用している自由（力学的）境界とする．さらに一般的な線形または非線形境界もあるが，ここでは取り扱わないこととする．

境界条件式は (4-23) のである．

$$\begin{aligned} &u=\bar{u},\quad v=\bar{v},\quad w=\bar{w},\quad u'=\bar{u}',\quad v'=\bar{v}',\quad w'=\bar{w}'\cdots \quad (s_1\ 上)\\ &N_s=\bar{N}_s,\ N_t=\bar{N}_t,\ N_n=\bar{N}_n,\quad M_s=\bar{M}_s,\ M_t=\bar{M}_t,\ M_n=\bar{M}_n\cdots \quad (s_2\ 上)\end{aligned} \qquad (4\text{-}23)$$

領域 S の内部では s, t, n 方向に面内力 S, T, N 板の上下面に面力 $p_s^u, p_s^l, p_t^u, p_t^l, p_n^u, p_n^l\cdots$ が作用するものとする．

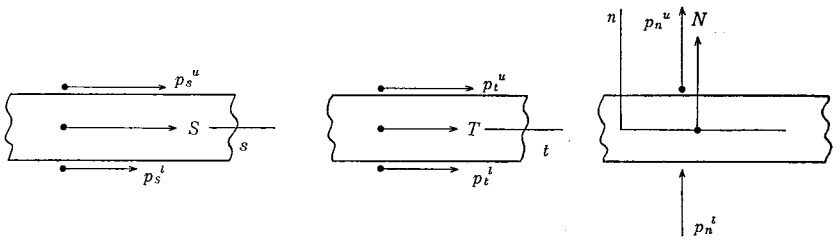


図 4 面力ならびに表面力

Fig. 4 Internal, upper and lower surface force

これら外力による外部エネルギは次のように表される．

$$W=\int_{s_2}(\bar{N}_s u+\bar{N}_t v+\bar{N}_n w+\bar{M}_s u'+\bar{M}_t v'+\bar{M}_n w'+\cdots)ds$$
$$+\iint_S(f_s u+f_t v+f_n w+f_s' u'+f_t' v'+f_n' w'+\cdots)dxdy \qquad (4\text{-}24)$$

ただし，

$$\begin{aligned} &f_s=S+p_s^u+p_s^l, \quad &&f_s'=(h/2)(p_s^u-p_s^l)\\ &f_t=T+p_t^u+p_t^l, \quad &&f_t'=(h/2)(p_t^u-p_t^l)\\ &f_n=N+p_n^u+p_n^l, \quad &&f_n'=(h/2)(p_n^u-p_n^l)\end{aligned} \qquad (4\text{-}25)$$

4.3 変分原理ならびに支配方程式

式 (4-1) および (4-25) によって，変分原理 (4-26) が成立する．

$$\int_s\{(N_s-\bar{N}_s)\delta u+(N_t-\bar{N}_t)\delta v+(N_n-\bar{N}_n)\delta w$$
$$+(M_s-\bar{M}_s)\delta u'+(M_t-\bar{M}_t)\delta v'+(M_n-\bar{M}_n)\delta w'\}ds$$
$$-\iint_S\{(L_s+f_s)\delta u+(L_t+f_t)\delta v+(L_n+f_n)\delta w$$
$$+(L'_s+f_s')\delta u'+(L_t'+f_t')\delta v'+(L_n'+f_n')\delta w'\}dxdy$$
$$=\delta U-\delta W=0 \tag{4-26}$$

上式により，境界条件式（4-23）ならびに平衡方程式（4-27）が導入される．

$$L_s+f_s=0,\ L_t+f_t=0,\ L_n+f_n=0,\ L_s'+f_s'=0,\ L_t'+f_t'=0,\ L_n'+f_n'=0 \tag{4-27}$$

これらの変分原理は有限要素法による数値解析に利用され，境界条件式および平衡方程式は差分法による数値解析に利用される．

とくに板厚 h=一定　のとき，平衡方程式（4-27）は下記のように整理される．

$$(1-\nu)\frac{\partial^2u}{\partial x^2}+\frac{1-2\nu}{2}\frac{\partial^2u}{\partial y^2}-\frac{1-2\nu}{2}(f_{xx}{}^2+f_{xy}{}^2)u+\frac{1}{2}\frac{\partial^2v}{\partial x\partial y}-\frac{1-2\nu}{2}f_{xy}(f_{xx}+f_{yy})v$$
$$-\left(\frac{3-4\nu}{2}f_{xx}+\nu f_{yy}\right)\frac{\partial w}{\partial x}-\frac{3}{2}(1-2\nu)f_{xy}\frac{\partial w}{\partial y}-(1-\nu)(f_{xxx}+f_{xyy})w$$
$$-\frac{1-2\nu}{2}(f_{xx}u'+f_{xy}v')+\nu\frac{\partial w'}{\partial x}+\frac{f_s}{k^*}=0 \tag{4-28}$$

$$\frac{1}{2}\frac{\partial^2u}{\partial x\partial y}-\frac{1-2\nu}{2}f_{xy}(f_{xx}+f_{yy})u+\frac{1-2\nu}{2}\frac{\partial^2v}{\partial x^2}+(1-\nu)\frac{\partial^2v}{\partial y^2}-\frac{1-2\nu}{2}(f_{xy}{}^2+f_{yy}{}^2)v$$
$$-\frac{3}{2}(1-2\nu)f_{xy}\frac{\partial w}{\partial x}-\left(\nu f_{xx}+\frac{3-4\nu}{2}f_{yy}\right)\frac{\partial w}{\partial y}-(1-\nu)(f_{xxy}+f_{yyy})w$$
$$-\frac{1-2\nu}{2}(f_{xy}u'+f_{yy}v')+\nu\frac{\partial w'}{\partial y}+\frac{f_t}{k^*}=0 \tag{4-29}$$

$$\left(\frac{3-4\nu}{2}f_{xx}+\nu f_{yy}\right)\frac{\partial u}{\partial x}+\frac{3}{2}(1-2\nu)f_{xy}\frac{\partial u}{\partial y}+\frac{1-2\nu}{2}(f_{xxx}+f_{xyy})u$$
$$+\frac{3}{2}(1-2\nu)f_{xy}\frac{\partial v}{\partial x}+\left(\nu f_{xx}+\frac{3-4\nu}{2}f_{yy}\right)\frac{\partial v}{\partial y}+\frac{1-2\nu}{2}(f_{xxy}+f_{yyy})v$$
$$+\frac{1-2\nu}{2}\left(\frac{\partial^2w}{\partial x^2}+\frac{\partial^2w}{\partial y^2}\right)-\{(1-\nu)f_{xx}{}^2+2\nu f_{xx}f_{yy}+(1-\nu)f_{yy}{}^2+2(1-2\nu)f_{xy}{}^2\}w$$
$$+\frac{1-2\nu}{2}\left(\frac{\partial u'}{\partial x}+\frac{\partial v'}{\partial y}\right)+\nu(f_{xx}+f_{yy})w'+\frac{f_n}{k^*}=0 \tag{4-30}$$

$$-6\frac{1-2\nu}{h^2}\left(f_{xx}u+f_{xy}v+\frac{\partial w}{\partial x}\right)+(1-\nu)\frac{\partial^2u'}{\partial x^2}+\frac{1-2\nu}{2}\left\{\frac{\partial^2u'}{\partial y^2}-\left(f_{xx}{}^2+f_{xy}{}^2+\frac{12}{h^2}\right)u'\right\}$$
$$+\frac{1}{2}\frac{\partial^2v'}{\partial x\partial y}-\frac{1-2\nu}{2}f_{xy}(f_{xx}+f_{yy})v'-\left(\frac{3-4\nu}{2}f_{xx}+\nu f_{yy}\right)\frac{\partial w'}{\partial x}-\frac{3}{2}(1-2\nu)f_{xy}\frac{\partial w'}{\partial y}$$
$$-(1-\nu)(f_{xxx}+f_{xyy})w'-(1-2\nu)(f_{xx}u''+f_{xy}v'')+2\nu\frac{\partial w''}{\partial x}+\frac{f_s'}{D^*}=0 \tag{4-31}$$

$$-6\frac{1-2\nu}{h^2}\left(f_{xy}u+f_{yy}v+\frac{\partial w}{\partial y}\right)+\frac{1}{2}\frac{\partial^2u'}{\partial x\partial y}-\frac{1-2\nu}{2}f_{xy}(f_{xx}+f_{yy})u'$$
$$+\frac{1-2\nu}{2}\frac{\partial^2v'}{\partial x^2}+(1-\nu)\frac{\partial^2v'}{\partial y^2}-\frac{1-2\nu}{2}\left(f_{xy}{}^2+f_{yy}{}^2+\frac{12}{h^2}\right)v'$$
$$-\frac{3}{2}(1-2\nu)f_{xy}\frac{\partial w'}{\partial x}+\left(\nu f_{xx}+\frac{3-4\nu}{2}f_{yy}\right)\frac{\partial w'}{\partial y}-(1-\nu)(f_{xxy}+f_{yyy})w'$$
$$-(1-2\nu)(f_{xy}u''+f_{yy}v'')+2\nu\frac{\partial w''}{\partial y}+\frac{f_t'}{D^*}=0 \tag{4-32}$$

$$-12\frac{\nu}{h^2}\left\{\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}-(f_{xx}+f_{yy})w\right\}+\left(\frac{3-4\nu}{2}f_{xx}+\nu f_{yy}\right)\frac{\partial u'}{\partial x}+\frac{3}{2}(1-2\nu)f_{xy}\frac{\partial u'}{\partial y}$$
$$+\frac{1-2\nu}{2}(f_{xxx}+f_{xyy})u'+\frac{3}{2}(1-2\nu)f_{xy}\frac{\partial v'}{\partial x}+\left(\nu f_{xx}+\frac{3-4\nu}{2}f_{yy}\right)\frac{\partial v'}{\partial y}$$
$$+\frac{1-2\nu}{2}(f_{xxy}+f_{yyy})u'+\frac{1-2\nu}{2}\left(\frac{\partial^2 w'}{\partial x^2}+\frac{\partial^2 w'}{\partial y^2}\right)$$
$$-\left\{(1-\nu)f_{xx}{}^2+2\nu f_{xx}f_{yy}+(1-\nu)f_{yy}{}^2+2(1-2\nu)f_{xy}{}^2+12\frac{1-\nu}{h^2}\right\}w'$$
$$+(1-2\nu)\left(\frac{\partial u''}{\partial x}+\frac{\partial v''}{\partial y}\right)+2\nu(f_{xx}+f_{yy})w''+\frac{f_n{}'}{D^*}=0 \qquad (4\text{-}33)$$

以上の式において，u'', v'', w'' を落とせば6個の変数 u, v, w, u', v', w' に関する6個の連立微分方程式となり，境界条件式と併わせて，たとえば任意節点配置差分法により数値的に解くことができる．これらの式は $h \ll l \ll \rho$ を仮定して，次のように近似することができる．

式 (4-28) では第3，5項，式 (4-29) では第2，5項，式 (4-30) では第8項を省略し，式 (4-31)～(4-33) は次のように変形する．

$$f_{xx}u+f_{xy}v+\frac{\partial w}{\partial x}-\frac{h^2}{12}\left(2\frac{1-\nu}{1-2\nu}\frac{\partial^2 u'}{\partial x^2}+\frac{\partial^2 u'}{\partial y^2}\right)+u'-\frac{h^2}{12(1-2\nu)}\frac{\partial^2 v'}{\partial x\partial y}=\frac{f_s{}'}{k'} \qquad (4\text{-}34)$$

$$f_{xy}u+f_{yy}v+\frac{\partial w}{\partial y}-\frac{h^2}{12(1-2\nu)}\frac{\partial^2 u'}{\partial x\partial y}-\frac{h^2}{12}\left(\frac{\partial^2 v'}{\partial x^2}+2\frac{1-\nu}{1-2\nu}\frac{\partial^2 v'}{\partial y^2}\right)+v'=\frac{f_t{}'}{k'} \qquad (4\text{-}35)$$

$$2\nu\left\{\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}-(f_{xx}+f_{yy})w\right\}-\frac{1-2\nu}{12}h^2\left(\frac{\partial^2 w'}{\partial x^2}+\frac{\partial^2 w'}{\partial y^2}\right)+2(1-\nu)w'=\frac{f_u{}'}{k'} \qquad (4\text{-}36)$$

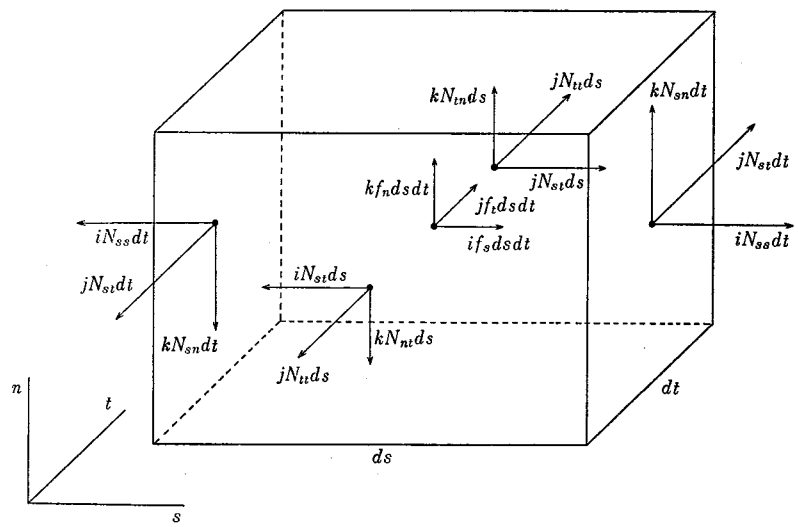


図 5 曲面板の微小面素に作用する面内力，面力

Fig. 5 Internal and external surface force which act to elementary part of curved plate

力の平衡方程式は図5のベクトル図からも導かれる．

$$(iN_{ss}+jN_{st}+kN_{sn})dt\,|_s{}^{s+ds}+(iN_{st}+jN_{tt}+kN_{tn})ds\,|_t{}^{t+dt}$$
$$+(if_s+jf_t+kf_n)ds dt=\{i(L_s+f_s)+j(L_t+f_t)+k(L_n+f_n)\}ds ds=0 \qquad (4\text{-}37)$$

モーメントの平衡についても同様である．

4.4 有限要素法による計算

有限要素法により計算を行う場合は，内部エネルギ (4-22) ならびに外部エネルギ (4-24) を用いる．ここでは前記の仮定により，内部エネルギの第2積分式を次のように変形する．

$$\frac{1}{2}\iint_S D^*\Big[(1-\nu)\Big\{\Big(\frac{\partial u'}{\partial x}\Big)^2+\Big(\frac{\partial v'}{\partial y}\Big)^2\Big\}+2\nu\frac{\partial u'}{\partial x}\frac{\partial v'}{\partial y}$$
$$+\frac{2-2\nu}{2}\Big\{\Big(\frac{\partial u'}{\partial y}+\frac{\partial v'}{\partial x}\Big)^2+\Big(\frac{\partial w'}{\partial x}\Big)^2+\Big(\frac{\partial w'}{\partial y}\Big)^2\Big\}\Big]dxdy \tag{4-38}$$

いま節点における曲面の高さを $z_m=f_m$ とすれば，要素内分布は

$$z=p_m f_m \tag{4-39}$$

によって表される．ここに p_m は節点 m に関する形状関数である．変数 u, v, w, u', v', w' はすべて同じクラスA要素によって展開することができる．

4.5 任意節点配置差分法による計算

微分方程式 (4-28)～(4-33) ならびに境界条件式を標準形

$$A_m{}^{pq}a_m{}^q=f^p \qquad (p,q=1\sim6)$$

によって表し，汎用プログラムにより計算する．ただし

$$u^1=u,\quad u^2=v,\quad u^3=w,\quad u^4=u',\quad u^5=v',\quad u^6=w'$$
$$a_1{}^q=u^q,\ a_2{}^q=u^q{}_x,\ a_3{}^q=u_y{}^q,\ a_4{}^q=u_{xx}{}^q,\ a_5{}^q=u_{xy}{}^q,\ a_6{}^q=u_{yy}{}^q \tag{4-40}$$

で $A_m{}^{pq}$ はその係数である．

計算に当たっては，まず境界に適合するように曲線格子系を設定し，その格子点に節点 i を付番する．節点 i において微分方程式の差分化を行うため差分家族構成を定める．ここでは微分方程式の階数は，すべて2であるから通常6節点2次の構成でよい．

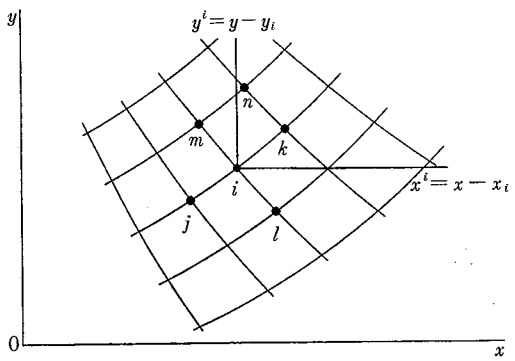


図 6 差分家族節点配置

Fig. 6 Node distribution of differential family

i 差分家族構成

$$1^i=i,\ 2^i=j,\ 3^i=k,\ 4^i=l,\ 5^i=m,\ 6^i=n$$

i 差分家族局所座標

$$x_1{}^i=0,\ x_2{}^i=x_j-x_i,\ x_3{}^i=x_k-x_i,\ x_4{}^i=x_l-x_i,\ x_5{}^i=x_m-x_i,\ x_6{}^i=x_n-x_i$$

$y_m{}^i$ についても同様

展開関数列

$$f_1=1,\ f_2=x,\ f_3=y,\ f_4=x^2/2,\ f_5=xy,\ f_6=y^2/2$$

$u_m{}^i=u(x_m{}^i, y_m{}^i)$ $f_{mn}{}^i=f_n(x_m{}^i, y_m{}^i)$ とすると $\sum_{n=1}^{6} f_{mn}a_n{}^i=u_m{}^i$

$g_{mn}{}^{i}$ は $f_{mn}{}^{i}$ の逆行列とすると，$a_m{}^{i}=\sum_{n=1}^{6} g_{mn}{}^{i}u_n{}^{i}$

$k_n{}^{pqi}=A_{mi}{}^{pq}g_{mn}{}^{i}$

差分家族構成により $k_1{}^{pqi}=k_{ii}, k_2{}^{pqi}=k_{ij}, k_3{}^{pqi}=k_{ik}, k_4{}^{pqi}=k_{il}, k_5{}^{pqi}=k_{im}, k_6{}^{pqi}=k_{in}$

したがって，次の連立方程式が求まる．

$$k_{ij}{}^{pq}u_j{}^{q}=f_i{}^{p}$$

5. 直接，剪断応力拘束による支配方程式

4章において述べた微分方程式は，板厚が薄くなるとともに数値計算が困難となることが予想される．ここでは，これを避けるために薄い平板の応力解析などに用いられている下記の変形を行う．その基準として，変形の標準波長 $\lambda=2\pi l$ を仮定する．ここで $l=\lambda/2\pi$ は標準微分長と呼ぶこととする．まず u, v, w を標準として，u', v', w' はその $1/l$ 程度と仮定する．したがって，式 (4-2)～(4-13) により了解されるように $N_{ss}, \cdots, N_{tn}$ の大きさは $o(h/l)$ 程度に，$M_{ss}, \cdots, M_{tn}$ は $o(h^3/l^2)$ 程度に評価される．曲率半径は $\rho>l$ と考える．したがって，式 (4-19) において第1ないし第4項の大きさは $o(h^3/l^3)$ 程度，第5項の N_{sn} は $o(h/l)$ の程度である．このように N_{sn} に対し，それ以外の項は $(h/l)^2$ 程度の大きさとなり，板厚が薄くなるとともに無視できる大きさとなる．式 (4-20), (4-21) についても同様である．このようにして平衡方程式 (4-27) の第1近似式は

$$N_{sn}=f_s', N_{tn}=f_t', N_{nn}=f_n' \tag{5-1}$$

とすることができる．上式より u', v', w' の近似式

$$u'=\frac{f_s'}{k'}-\left(f_{xx}u+f_{xy}v+\frac{\partial w}{\partial x}\right) \tag{5-2}$$

$$v'=\frac{f_t'}{k'}-\left(f_{xy}u+f_{yy}v+\frac{\partial w}{\partial y}\right) \tag{5-3}$$

$$w'=\frac{f'_n}{k^*}-\frac{\nu}{1-\nu}\left\{\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}-(f_{xx}+f_{yy})w\right\} \tag{5-4}$$

が導れる．さらに，ここでは次の仮定を行う．

$$M_{sn}=0,\ M_{tn}=0,\ M_{nn}=0 \tag{5-5}$$

したがって，面内力およびモーメントは次のように変形される．

$$N_{ss}=k\left\{\frac{\partial u}{\partial x}+\nu\frac{\partial v}{\partial y}-(f_{xx}+\nu f_{yy})w\right\}+\nu f_n' \tag{5-6}$$

$$N_{tt}=k\left\{\nu\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}-(\nu f_{xx}+f_{yy})w\right\}+\nu f_n' \tag{5-7}$$

$$N_{nn}=f_{xx}M_{ss}+2f_{xy}M_{st}+f_{yy}M_{yy}+f_n' \tag{5-8}$$

$$N_{st}=k'\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}-2f_{xy}w\right) \tag{5-9}$$

$$N_{sn}=\frac{\partial M_{ss}}{\partial x}+\frac{\partial M_{st}}{\partial y}+f_s' \tag{5-10}$$

$$N_{tn}=\frac{\partial M_{st}}{\partial x}+\frac{\partial M_{tt}}{\partial y}+f'_t \tag{5-11}$$

$$M_{ss}=D\left\{\frac{\partial u'}{\partial x}+\nu\frac{\partial v'}{\partial y}-(f_{xx}+\nu f_{yy})w'\right\} \tag{5-12}$$

$$M_{tt}=D\left\{\nu\frac{\partial u'}{\partial x}+\frac{\partial v'}{\partial y}-(\nu f_{xx}+f_{yy})w'\right\} \tag{5-13}$$

$$M_{st}=D'\left(\frac{\partial u'}{\partial y}+\frac{\partial v'}{\partial x}-2f_{xy}w'\right) \tag{5-14}$$

上式の u', v', w' には (5-2), (5-3), (5-4) を用いる．すなわち，

$$\begin{aligned}M_{ss}=-D\Bigg[&\frac{1}{1-\nu}\{(1-2\nu)f_{xx}-\nu^2 f_{yy}\}\frac{\partial u}{\partial x}+\nu f_{xy}\frac{\partial u}{\partial y}+(f_{xxx}+\nu f_{xyy})u\\&+f_{xy}\frac{\partial v}{\partial x}+\frac{\nu}{1-\nu}\{-f_{xx}+(1-2\nu)f_{yy}\}\frac{\partial v}{\partial y}+(f_{xxy}+\nu f_{yyy})v\\&+\frac{\partial^2 w}{\partial x^2}+\nu\frac{\partial^2 w}{\partial y^2}+\frac{\nu}{1-\nu}(f_{xx}+\nu f_{yy})(f_{xx}+f_{yy})w\\&-\frac{\partial}{\partial x}\left(\frac{f_s'}{k'}\right)-\nu\frac{\partial}{\partial y}\left(\frac{f_t'}{k'}\right)+(f_{xx}+\nu f_{yy})\frac{f_n'}{k^*}\Bigg]\end{aligned} \tag{5-15}$$

$$\begin{aligned}M_{tt}=-D\Bigg[&\frac{\nu}{1-\nu}\{(1-2\nu)f_{xx}-f_{yy}\}\frac{\partial u}{\partial x}+f_{xy}\frac{\partial u}{\partial y}+(\nu f_{xxx}-f_{xyy})u\\&+\nu f_{xy}\frac{\partial v}{\partial x}+\frac{1}{1-\nu}\{-\nu^2 f_{xx}+(1-2\nu)f_{yy}\}\frac{\partial v}{\partial y}+(\nu f_{xxy}+f_{yyy})v\\&+\nu\frac{\partial^2 w}{\partial x^2}+\frac{\partial^2 w}{\partial y^2}+\frac{\nu}{1-\nu}(\nu f_{xx}+f_{yy})(f_{xx}+f_{yy})w\\&-\nu\frac{\partial}{\partial x}\left(\frac{f_s'}{k'}\right)\frac{\partial}{\partial y}\left(\frac{f_t'}{k'}\right)+(\nu f_{xx}+f_{yy})\frac{f_n'}{k^*}\Bigg]\end{aligned} \tag{5-16}$$

$$\begin{aligned}M_{st}=-D'\Bigg\{&\frac{1-3\nu}{1-\nu}f_{xy}\frac{\partial u}{\partial x}+f_{xx}\frac{\partial u}{\partial y}+2f_{xxy}u+f_{yy}\frac{\partial v}{\partial x}\\&+\frac{1-3\nu}{1-\nu}f_{xy}\frac{\partial v}{\partial y}+2f_{xyy}v+2\frac{\partial^2 w}{\partial x\partial y}+\frac{2\nu}{1-\nu}f_{xy}(f_{xx}+f_{yy})w\\&-\frac{\partial}{\partial y}\left(\frac{f_s'}{k'}\right)-\frac{\partial}{\partial x}\left(\frac{f_t'}{k'}\right)+2f_{xy}\frac{f_n'}{k^*}\Bigg\}\end{aligned} \tag{5-17}$$

N_{sn}, N_{tn} の第2近似式は上記の M_{ss}, M_{tt}, M_{st} を式 (5-10), (5-11) に代入することにより得られる．ここでは，簡単のため曲率は3次式分布で近似されるものとして f の4次以上の微係数を省略し，さらに曲率半径は変形の標準波長よりもかなり大きいものとして f に関する非線形項は省略することとする．また板厚が一定の場合，N_{sn}, N_{tn} は次のように与えられる．

$$\begin{aligned}N_{sn}=-D\Bigg[&\frac{1}{1-\nu}\left\{(1-2\nu)f_{xx}-\nu^2 f_{yy}\right\}\frac{\partial^2 u}{\partial x^2}+\frac{1}{2}(1-\nu)f_{xy}\frac{\partial^2 u}{\partial x\partial y}+\frac{1}{2}(1-\nu)f_{xx}\frac{\partial^2 u}{\partial y^2}\\&+\frac{1}{1-\nu}\left\{(2-3\nu)f_{xxx}+\frac{1}{2}(1-2\nu-\nu^2)f_{xyy}\right\}\frac{\partial u}{\partial x}+\frac{1}{2}(3-\nu)f_{xxy}\frac{\partial u}{\partial y}\\&+(f_{xxxx}+f_{xxyy})u+f_{xy}\frac{\partial^2 v}{\partial x^2}+\frac{1}{1-\nu}\left\{-\nu f_{xx}+\frac{1}{2}(1-3\nu^2)f_{yy}\right\}\frac{\partial^2 v}{\partial x\partial y}\\&+\frac{1-3\nu}{2}f_{xy}\frac{\partial^2 v}{\partial y^2}+\left(2f_{xxy}+\frac{1+\nu}{2}f_{yyy}\right)\frac{\partial v}{\partial x}+\frac{1}{1-\nu}\Bigg\{-\nu f_{xxx}\\&+\frac{1}{2}(3-6\nu+\nu^2)f_{xyy}\Bigg\}\frac{\partial v}{\partial y}+(f_{xxxy}+f_{xyyy})v+\frac{\partial^3 w}{\partial x^3}+\frac{\partial^3 w}{\partial x\partial y^2}\Bigg]+F_s'\end{aligned} \tag{5-18}$$

$$\begin{aligned}N_{tn}=-D\Bigg[&\frac{1-3\nu}{2}f_{xy}\frac{\partial^2 u}{\partial x^2}+\frac{1}{1-\nu}\left\{\frac{1}{2}(1-3\nu^2)f_{xx}-\nu f_{yy}\right\}\frac{\partial^2 u}{\partial x\partial y}+f_{xy}\frac{\partial^2 u}{\partial y^2}\\&+\frac{1}{1-\nu}\left\{\frac{1}{2}(3-6\nu+\nu^2)f_{xxy}-\nu f_{yyy}\right\}\frac{\partial u}{\partial x}+\left(\frac{1+\nu}{2}f_{xxx}+2f_{xyy}\right)\frac{\partial u}{\partial y}+(f_{xxxy}\\&+f_{xyyy})u+\frac{1-\nu}{2}f_{yy}\frac{\partial^2 v}{\partial x^2}+\frac{1-\nu}{2}f_{xy}\frac{\partial^2 v}{\partial x\partial y}+\frac{1}{1-\nu}\{-\nu^2 f_{xx}+(1-2\nu)f_{yy}\}\frac{\partial^2 v}{\partial y^2}\end{aligned}$$

$$+\frac{3-\nu}{2}f_{xyy}\frac{\partial v}{\partial x}+\frac{1}{1-\nu}\left\{\frac{1}{2}(1-2\nu-\nu^2)f_{xxy}+(2-3\nu)f_{yyy}\right\}\frac{\partial v}{\partial y}$$

$$+(f_{xxyy}+f_{yyyy})v+\frac{\partial^3 w}{\partial x^2\partial y}+\frac{\partial^3 w}{\partial y^3}\Big]+F_t' \tag{5-19}$$

これらの式を (4-16), (4-17), (4-18) ならびに (4-27) に代入し，u, v, w に関する平衡方程式を導入することができる．

$$k\left\{\frac{\partial^2 u}{\partial x^2}+\frac{1-\nu}{2}\frac{\partial^2 u}{\partial y^2}+\frac{1+\nu}{2}\frac{\partial^2 v}{\partial x\partial y}-(f_{xx}+\nu f_{yy})\frac{\partial w}{\partial x}-(1-\nu)f_{xy}\frac{\partial w}{\partial y}-(f_{xxx}+f_{xyy})w\right\}$$

$$-(f_{xx}N_{sn}+f_{xy}N_{tn})+\nu\frac{\partial f_n'}{\partial x}-(f_{xx}f_s'+f_{xy}f_t')+f_s=0 \tag{5-20}$$

$$k\left\{\frac{1+\nu}{2}\frac{\partial^2 u}{\partial x\partial y}+\frac{1-\nu}{2}\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}-(1-\nu)f_{xy}\frac{\partial w}{\partial x}-(\nu f_{xx}+f_{yy})\frac{\partial w}{\partial y}-(f_{xxy}+f_{yyy})w\right\}$$

$$-(f_{xy}N_{sn}+f_{yy}N_{tn})+\nu\frac{\partial f_n'}{\partial y}-(f_{xy}f_s'+f_{yy}f_t')+f_t=0 \tag{5-21}$$

$$D\Big[\frac{1}{1-\nu}\{(1-2\nu)f_{xx}-\nu^2 f_{yy}\}\frac{\partial^3 u}{\partial x^3}+(1-2\nu)f_{xy}\frac{\partial^3 u}{\partial x^2\partial y}+\frac{1}{1-\nu}\{(1-\nu-\nu^2)f_{xx}-\nu f_{yy}\}\frac{\partial^3 u}{\partial x\partial y^2}$$

$$+f_{xy}\frac{\partial^3 u}{\partial y^3}+\frac{1}{1-\nu}\{(3-5\nu)f_{xxx}+(1-3\nu)f_{xyy}\}\frac{\partial^2 u}{\partial x^2}+\frac{1}{1-\nu}\{(4-6\nu)f_{xxy}$$

$$-2\nu f_{yyy}\}\frac{\partial^2 u}{\partial x\partial y}+(f_{xxx}+3f_{xyy})\frac{\partial^2 u}{\partial y^2}+\frac{1}{1-\nu}\{(3-4\nu)f_{xxxx}+(3-5\nu)f_{xxyy}$$

$$-\nu f_{yyyy}\}\frac{\partial u}{\partial x}+3(f_{xxxy}+f_{xyyy})\frac{\partial u}{\partial y}+(f_{xxxxx}+2f_{xxxyy}+f_{xyyyy})u$$

$$+f_{xy}\frac{\partial^3 v}{\partial x^3}+\frac{1}{1-\nu}\{-\nu f_{xx}+(1-\nu-\nu^2)f_{yy}\}\frac{\partial^3 v}{\partial x^2\partial y}+(1-2\nu)f_{xy}\frac{\partial^3 v}{\partial x\partial y^2}$$

$$+\frac{1}{1-\nu}\{-\nu^2 f_{xx}+(1-2\nu)f_{yy}\}\frac{\partial^3 v}{\partial y^3}+(3f_{xxy}+f_{yyy})\frac{\partial^2 v}{\partial x^2}$$

$$+\frac{1}{1-\nu}\Big\{-2\nu f_{xxx}+(4-6\nu)f_{xyy}\Big\}\frac{\partial^2 v}{\partial x\partial y}+\frac{1}{1-\nu}\{(1-3\nu)f_{xxy}+(3-5\nu)f_{yyy}\}\frac{\partial^2 v}{\partial y^2}$$

$$+3(f_{xxxy}+f_{xyyy})\frac{\partial v}{\partial x}+\frac{1}{1-\nu}\{-\nu f_{xxxx}+(3-5\nu)f_{xxyy}+(3-4\nu)f_{yyyy}\}\frac{\partial v}{\partial y}$$

$$+(f_{xxxxy}+2f_{xxyyy}+f_{yyyyy})v+\frac{\partial^4 w}{\partial x^4}+2\frac{\partial^4 w}{\partial x^2\partial y^2}+\frac{\partial^4 w}{\partial y^4}\Big]$$

$$-(f_{xx}N_{ss}+2f_{xy}N_{st}+f_{yy}N_{tt})+F_n=0 \tag{5-22}$$

ただし，

$$F_s'=\frac{h^2}{6(1-\nu)}\left(\frac{\partial^2 f_s'}{\partial x^2}+\frac{1-\nu}{2}\frac{\partial^2 f_s'}{\partial y^2}+\frac{1+\nu}{2}\frac{\partial^2 f_t'}{\partial x\partial y}\right)-\frac{h^2}{12}\left\{(f_{xx}+\nu f_{yy})\frac{\partial f_n'}{\partial x}+f_{xy}\frac{\partial f_n'}{\partial y}\right\}$$

$$+f_s' \tag{5-23}$$

$$F_t'=\frac{h^2}{6(1-\nu)}\left(\frac{1+\nu}{2}\frac{\partial^2 f_s'}{\partial x\partial y}+\frac{1-\nu}{2}\frac{\partial^2 f_t'}{\partial x^2}+\frac{\partial^2 f_t'}{\partial y^2}\right)-\frac{h^2}{12}\left\{f_{xy}\frac{\partial f_n'}{\partial x}+(\nu f_{xx}+f_{yy})\frac{\partial f_n'}{\partial y}\right\}$$

$$+f_t' \tag{5-24}$$

$$F_n=\frac{h^2}{6(1-\nu)}\left(\frac{\partial^3 f_s'}{\partial x^3}+\frac{\partial^3 f_s'}{\partial x\partial y^2}+\frac{\partial^3 f_t'}{\partial x^2\partial y}+\frac{\partial^3 f_t'}{\partial y^3}\right)+\left(\frac{\partial f_s'}{\partial x}+\frac{\partial f_t'}{\partial y}\right)$$

$$-\frac{h^2}{12}\left\{(f_{xx}+\nu f_{yy})\frac{\partial^2 f_n'}{\partial x^2}+2f_{xy}\frac{\partial^2 f_n'}{\partial x\partial y}+(\nu f_{xx}+f_{yy})\frac{\partial^2 f_n'}{\partial y^2}\right\}+f_n \tag{5-25}$$

いま表面力 $p_s{}^u\cdots p_n{}^l$ の大きさを p の程度とすれば，式(5-20)における $\nu f_{n,}{}'_x$, $(f_{xx}f_s'+f_{xy}f_t')$ の大きさは f_s を標準として，それぞれ $o(h/l)$, $o(h/\rho)$ の程度である．したが

って，浅く薄い曲面板では f_s に対して他の項を省略することができる．式 (5-21) についても同様である．式 (5-25) についても第1，2，3項の大きさは f_n を標準として，それぞれ $o(h^3/l^3), o(h/l), o(h^3/\rho l^2)$ の程度である．したがって第1，3項は第4項に比べて非常に小さく，無視できる．また第2項も特別に高い精度の計算を行う場合を除いて省略してもよい．式 (5-20) における $f_{xx}N_{su}+f_{xy}N_{tn}$ は前の項に対して $o(h^2/\rho l)$ の大きさであるから省略できる．式 (5-21) についても同様である．(5-22) において $D(w_{xxxx}+2w_{xxyy}+w_{yyyy})$ を第2項，それ以前を第1項，$f_{xx}N_{ss}+2f_{xy}N_{st}+f_{yy}N_{tt}$ を第3項として，これらの大きさはそれぞれ $o(h^3/l^3\rho), o(h^3/l^4), o(h/l\rho)$ である．したがって第2項を基準として，第1項は $o(l/\rho)$，第3項は $o(l^3/\rho h^2)$ である．したがって非常に浅い曲面板では，第1項は省略できる．なお第1項は非常に詳細な形で与えられているが，たとえば $f(x,y)$ を2次分布で近似すれば，その3次以上の微係数は落とすことができる．第3項の大きさは $(l/\rho)\times(l/h)^2$ で極端に浅い曲面板では省略できるが，一般には残すべき項である．なお厚板または薄板でも非常に高調波の波形をもつ変形に対しては h/l が1と同程度の大きさとなりうるので，この場合は第1項も無視することはできなくなる．このように支配方程式は不変のものでなく，変形の有様により形を変えるものである．

以上の考察により，実用的な近似式として次の式を推薦する．

$$\frac{\partial^2 u}{\partial x^2}+\alpha\frac{\partial^2 u}{\partial y^2}+\beta\frac{\partial^2 v}{\partial x\partial y}-\left\{(t_{xx}+\nu f_{yy})\frac{\partial w}{\partial x}+(1-\nu)f_{xy}\frac{\partial w}{\partial y}+(f_{xxx}+f_{xyy})w\right\}+\frac{f_s}{k}=0 \tag{5-26}$$

$$\beta\frac{\partial^2 u}{\partial x\partial y}+\alpha\frac{\partial^2 v}{\partial x^2}+\frac{\partial^2 v}{\partial y^2}-\left\{(1-\nu)f_{xy}\frac{\partial w}{\partial x}+(\nu f_{xx}+f_{yy})\frac{\partial w}{\partial y}+(f_{xxy}+f_{yyy})w\right\}+\frac{f_t}{k}=0 \tag{5-27}$$

$$-\left\{(f_{xx}+\nu f_{yy})\frac{\partial u}{\partial x}+(1-\nu)f_{xy}\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)+(\nu f_{xx}+f_{yy})\frac{\partial v}{\partial y}\right\}$$
$$+\frac{h^2}{12}\left(\frac{\partial^4 w}{\partial x^4}+2\frac{\partial^4 w}{\partial x^2\partial y^2}+\frac{\partial^4 w}{\partial y^4}\right)+\{f_{xx}{}^2+2\nu f_{xx}f_{yy}+f_{yy}{}^2+2(1-\nu)f_{xy}{}^2\}w=\frac{f_n}{k} \tag{5-28}$$

なお，式 (5-28) 右辺の第3項は非常に浅い曲面板については省略することができる．当然曲率を落とせば，上式は平板の弾性方程式と一致する．また $h=0$ と置けば膜の方程式となる．

5.1 内部エネルギ

簡単のため曲率の微分値は省略し，さらに式 (4-28) における $(w_{x'})^2+(w_{y'})^2$ の項を落として次式を導く．

$$U=\frac{1}{2}\iint_k\left[\left(\frac{\partial u}{\partial x}-f_{xx}w\right)^2+2\nu\left(\frac{\partial u}{\partial x}-f_{xx}w\right)\left(\frac{\partial v}{\partial y}-f_{yy}w\right)+\left(\frac{\partial v}{\partial y}-f_{yy}w\right)^2\right.$$
$$+\frac{1-\nu}{2}\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}-2f_{xy}w\right)^2+\frac{h^2}{12}\left\{\left(f_{xx}\frac{\partial u}{\partial x}+f_{xy}\frac{\partial v}{\partial x}+\frac{\partial^2 w}{\partial x^2}\right)^2\right.$$
$$+2\nu\left(f_{xx}\frac{\partial u}{\partial x}+f_{xy}\frac{\partial v}{\partial x}+\frac{\partial^2 w}{\partial x^2}\right)\left(f_{xy}\frac{\partial u}{\partial y}+f_{yy}\frac{\partial v}{\partial y}+\frac{\partial^2 w}{\partial y^2}\right)^2+\left(f_{xy}\frac{\partial u}{\partial y}+f_{yy}\frac{\partial v}{\partial y}+\frac{\partial^2 w}{\partial y^2}\right)^2$$
$$\left.\left.+\frac{1-\nu}{2}\left(f_{xy}\frac{\partial u}{\partial x}+f_{xx}\frac{\partial u}{\partial y}+f_{yy}\frac{\partial v}{\partial x}+f_{xy}\frac{\partial v}{\partial y}+2\frac{\partial^2 w}{\partial x\partial y}\right)^2\right\}\right]dxdy \tag{5-29}$$

5.2 有限要素法による解析

u, v ならびに w の展開を行う要素をそれぞれ面内，面外要素とするとき，そのクラスはそれぞれA，Bとすべきである．

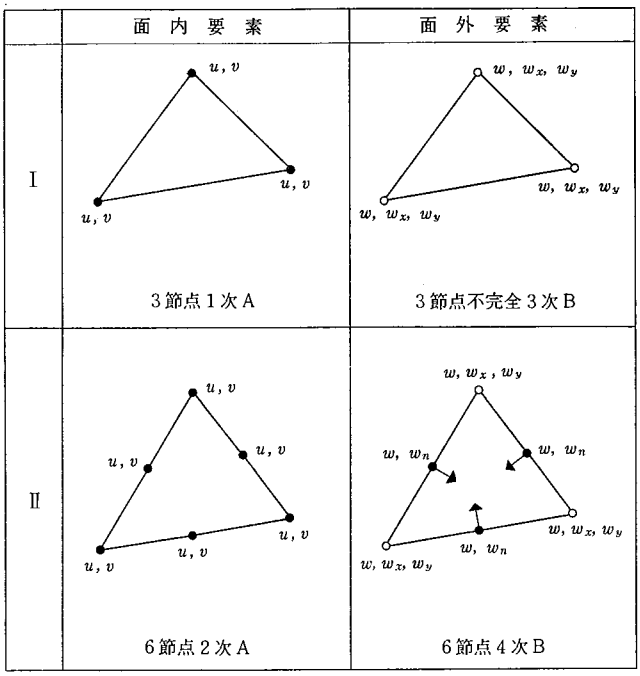


図 7 面内，面外要素の組み合せ

Fig. 7 Combination of internal (class A) and external (class B) element

たとえば図7に示されるような組み合せを用いればよい.

5.3 差分法による解析

式 (5-26), (5-27), (5-28) は任意節点配置差分法により差分表示される. これらの式は u, v に関し2階, w に関し4階の微分方程式である. したがって, 差分家族は u, v に関し6節点2次, w に関し15節点4次を用いる. なお, w の境界には, 1または2個のダミー節点を設けることが必要である.

6. おわりに

以上浅い曲面板について, その弾性力学的挙動を表す支配方程式を導き, その有限要素法ならびに任意節点配置差分法により解析法について, その概要を述べたものである. 実際の数値計算はまだ行っていないので, その精度および速度については未知で, これからの研究課題としたい. 今後さらにこの解析法を基として, 一般のシェル構造解析の精度, 能率化について数値実験研究の実施を予定している.

参考文献 [1] A. E. H Love, *Mathematical Theory of Elasticity*, Cambridge University Press, 1927.
[2] S. Timoshenko and J. N. Goodier, "Theory of Elasticity", *Engineering Societies Monographs*, 1951.
[3] S. Timoshenko and S. Woinowsky-krieger, *Theory of Pate and Shell*, McGraw-Hill, 1959. 長谷川節訳,「板とシェルの理論」上・下, ブレイン図書出版株式会社 1973, 1978.
[4] K. Washizu, *Variational Mechanics in Elasticity and Plasticity*, Pergamon Press, 1968.

執筆者紹介 藤 野　　勉 (Tsutomu Fujino)

明治45年生，昭和11年東京帝国大学理学部物理科卒業，同年三菱重工(株)入社，主に応力・振動流体力学などの解析法（有限要素法を含む）の研究に従事，32年，工学博士号を取得，47年，同社技術本部顧問となる．また，48年より東海大学工学部教授，52年依嘱教授を歴任．51年より日本ユニバック（株）の技術顧問となり，現在に至る．「コンピュータによる構造工学講座Ⅱ-4-B—熱伝導と熱応力」（培風館，1972）などの著書がある．

製品紹介

CHAPARRAL ファミリのハードウェア

間宮健二，牛尾　守

日本ユニバック(株)は，全世界で多くの導入実績をもち好評を博している UNIVAC シリーズ 1100 大型機の設計思想を新たに中型機に実現した「CHAPARRAL（シャパレル）ファミリ」を昨年発表した．

この CHAPARRAL ファミリは，シリーズ 1100 エントリ・システムとして中型機市場への参入を図ると同時に，個々の業務に応じて最適な問題解決が図れるアプリケーション専用機，およびプログラムレスで，利用者主導型情報処理システムができる「MAPPER 専用機」，さらにシリーズ 1100 ユーザに対する分散処理ファシリティを低価格で提供することを目的に開発されたものである．

そのハードウェア上の主な特徴は，次のとおりである．

特　徴

（1） 高性能 LSI の採用

VLSI を用いて，高性能・小型化を実現している．

（2） 1100/90 シリーズと同様に 3 種類の命令語実行モードの採用（ベーシック・モード，エクステンデッド・モード，ミックスド・モード）．

（3） 大容量の主記憶機構

524 キロ語(約 2 メガ・バイト相当)から最大 4192 キロ語（約 16 メガ・バイト相当）まで，524 キロ語単位で拡張が可能である．

（4） 内蔵型の通信制御機構および周辺装置制御機構の採用

通信制御装置の小規模なものおよび周辺装置制御機構は内蔵化され，システム全体が極めてコンパクトになっている．

（5） 日本語ワークステーションを最大 64 台まで接続可能

（6） 自動運転機能の採用

自動電源制御機構とソフトウェアの支援によって，オペレータの負荷が大幅に削減され，省力化が図られている．

（7） コンパクトで使いやすい豊富な入出力装置の採用

- 最小 848 メガ・バイト(1 台，2 スピンドル)から 424 メガ・バイト（1 スピンドル）単位に，最大 13.5 ギガ・バイト（16 台，32 スピンドル）までの磁気ディスク装置を接続できる．
- コンパクトで高性能な磁気テープ装置を 16 台まで接続できる．
- 簡便なストリーミング磁気テープ装置が接続できる．
- 日本語カット紙用プリンタから高速ライン・プリンタまでの豊富な印書装置群が利用できる．

（8） 二階層バス構造の採用

CHAPARRAL ファミリでは，各装置間の接続にシステム・バス（S バス）とライン・バス（L バス）を用いている．システム・バスは中央処理機構(CP)，主記憶機構（MS)，システム・サポート・プロセッサ（SSP)，ブロック多重チャネル（BMC)，低速入出力チャネル（BBC）間を接続する．

ライン・バスには，低速の入出力装置および端末装置群が制御機構を介して接続される．

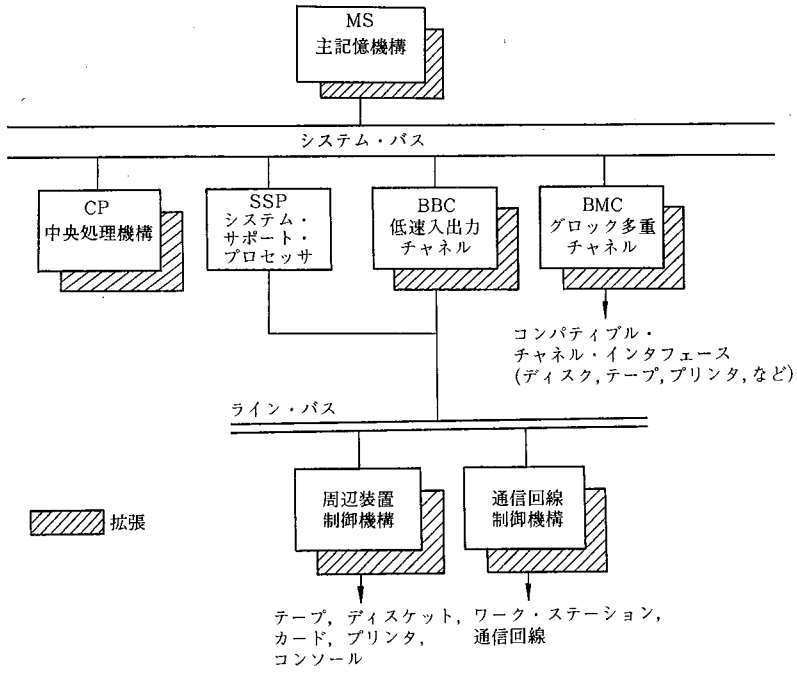
以下，これらの特徴のうち最も CHAPARRAL ファミリを特色づけている二階層バス構造について述べる．

CHAPARRAL ファミリの二階層バス構造

CHAPARRAL ファミリはバス結合方式を採用した．これは，バス結合方式の低コスト性，優れたモジュラリティや柔軟性に注目しての選択である．ただし，二階層バス構造すなわち，高速な装置を低速な周辺装置などと分離することによってシステム性能の低下を防いでいる（図参照）．

この方式における中心はシステム・バス（S バス）と呼ばれる高性能主記憶バスであり，この S バスを通して MS, CP, 各チャンネルおよび SSP との間でのデータや制御を交換する．S バスと並行してライン・バス（L バス）があり，セントラル・コンプレックスと各種周辺装置制御機構および通信制御機構との多重バイト・インタフェースを提供する．システムの入出力は，S バスに接続される次の二つのタイプのチャンネルを通して処理する．

低速入出力チャンネル（BBC)……バイト準拠の複数の内蔵型制御機構によって共有される多重チャンネルを介して接続する低速度の入出力用の装置であり，マイクロプログラム制御により内蔵周辺装置制御機構と内蔵通信制御機構を L バスを通して支援している．なお，このバイト入出力データはチャン



二階層バス構造（概念図）

ネルによって語準拠主記憶形式に変換する.

ブロック多重チャネル (BMC)……UNIVAC 1100/90 シリーズの BMC 機能をエミュレートするマイクロ・プログラム制御の装置で，セントラル・コンプレックスからケーブルで接続された外部フリー・スタンディング・サブシステムに標準入出力バスを提供する.

システムは，システム本体に組み込まれているシステム・サポート・プロセッサ (SSP) をもっている．この SSP のシステムに対する管理・回復・保守などの機能は，従来の UNIVAC シリーズ 1100 のもののサブセットとなっている.

システム・バス

システム・バス（Sバス）は，先に述べたようにセントラル・コンプレックスの基本的なデータ・パスである．また，Sバスは36ビット幅のデータ・インターフェースで，主記憶アクセス，汎用プロセッサ・インタフェース・メッセージ，システム制御メッセージ，エラー・ステータス・トランスファ，初期化，テストなどに用いられる．移動速度は 926 万語/秒（37メガ・バイト/秒相当）である．このように，Sバスは中央処理機構，ブロック多重チャネル，低速入出力チャネル，主記憶機構，SSP 間のインタフェースをとる.

Sバスは個々のバス・サイクルごとにデータを送ることができる．これはバス要求，情報転送，肯定応答をパイプライン処理によってオーバラップさせて実現している.

バスのアドレシングには，直接と間接の2種類の方式があり，直接アドレシングは主記憶機構がデータを要求した装置に読み出しデータを戻す場合などに使用する．すなわち，送り出し側の装置が，その情報の伝送のために単一の相手先装置があることを前もって決める場合に直接アドレシングが用いられる.

一方，間接アドレシングはアドレスとファンクションを特定の単一の相手先の装置へでなく，同一種類のすべての装置に同時送信したいときに使用し，どの装置が正しい唯一の相手先であるかを送り出し側の装置が前もって決められない場合に用いられる.

Sバスは固定的な優先順位評価方法を採用している．各装置は優先順位の制御論理回路が並列に動作するよう各装置のドロップに優先順位制御回路を内蔵しており，各装置は CP の指示を待たずにバスにアクセスできるかを決定できる.

Sバス上のすべての情報およびアドレスは，パリティ・チェックされる．各装置のリクエストあるいは肯定応答の論理回路における障害によって引き起こされるSバスのハング・アップは SSP によって検出される.

ライン・バス

CHAPARRAL は一つのライン・バス（Lバス）を内蔵している．このLバスは中央処理機構とキャビネット内の各種周辺装置制御機構，および通信制御機構との多重バイト・インタフェースを提供するものである．そして，バイト・バス・チャネルとSSP はSバスとLバスとの両方のインタフェースを有している．

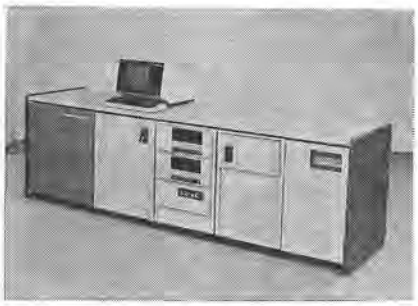
そして，Lバスには周辺装置制御機構と通信制御機構が合わせて16個まで接続できる．移送速度は660キロ・バイト/秒である．また，Lバスは高位，中位，低位の3段階のリクエスト優先順位をもっており，2台の装置が同一リクエスト順位でバスをアクセスする場合，ドロップ番地にもとづく順位が設定される．

スキャン/セット・インタフェース

Sバスは5本の信号線からなるスキャン/セット・インタフェースを有しており，このインタフェースから各Sバス装置内にある LSI ゲート・アレー・チップからなる一つあるいは複数のループが接続されている．そして，各チップは各ラッチ用の逐次スキャン／セット論理回路を内蔵している．

スキャン/セット論理回路とインタフェースは，SSP を経由することによってソフトウェア・メンテナンス・パネルとしてメンテナンスに使用されるとともに，ゲート・アレー・チップのテスト，プリント基板アセンブリの製造時の試験，装置の運転試験，プリント基板アセンブリ試験による欠陥チップの発見なども可能にする．

（ハードウェア・プロダクト統括部 C1 ハードウェア・プロダクト室/企画部　商品企画1室）



CHAPARRAL ファミリ

関係問い合せの最適化

原　　潔

1. はじめに

関係データベースは，自由度は高いが処理効率に難がある．そこで，問い合せ評価の効率を向上させる問い合せ最適化 (Query Optimization) の研究が活発化している．

現在の関係データベース・システム，たとえばIBM の System R, California 大学の INGRES, Computer Corporation of America の ADAPLEX などのオプティマイザは，二つの段階に分けて考えると見通しがよくなる．

第1の段階は，問い合せを一連の論理的な結合操作の系列 (Join Strategy, J 方策) に変換する．そして，第2の段階で結合方策をデータの移送，キャッシング，分類，インデックス生成などの物理的基本操作の系列 (Physical Strategy, P 方策) に変換する．そして，この二つの方策空間の探索を最適化する．本稿では，Sperry Research Center の D. Reiner と A. Rosenthal の文献[1]にもとづき，各関係データベース・システムで用いられているオプティマイザの性格を明らかにし，さらに同氏らのP方策の実行仮想機械，方策空間の生成および探索の効率の問題点などを紹介する．

2. 2段階方策空間

問い合せオプティマイザ (Query Optimizer) は，問い合せ (Query) を入力とし最適なデータ・アクセス方策 (Data Access Strategy) を生成するもので，通常次の2段階のプロセスを踏んで行われる．

まず，第1にオプティマイザは，問い合せをデータの格納されている場所や順序に依存しない論理レベルの結合方策 (Join Strategy) に変換する．これをJ方策 (J-Strategy) と呼んでいる．そして，つぎにこのプロセスと併行し，J方策をデータ移送，キャッシング，分類，索引作成，結合などの基本命令で構成された物理レベルの方策，P方策 (P-Strategy) に変換する．また，上記の基本命令を実行する仮想機械をPマシン (P-Machine) と呼ぶ．

このように方策空間を2レベルに分けることによって探策される方策が統一的に記述でき，同時に各オプティマイザの特徴が明確になる．

D. Reiner らの問い合せオプティマイザは，J方策の各結合に対するP方策を完全に実現するもので，J方策およびP方策のどちらの方策空間の改良も独立に行える．

このため，J方策レベルに索引を含む結合をつけ加えることや，P方策レベルで新しい結合の組み込みや蓄積構造の付加ができる．

3. 結合の拡張使用による機能強化

Reiner らの文献[5],[6]で取り扱われているデータはテーブル，つまりシステムの実行時にアクセスされる第1正規形の同一の構造をもつアクセス・レコードの集合である．このように定義の抽象度を上げることによって関係だけでなく，索引，CODASYL のセット，他の DAS (Direct Access Structure, ランダム・アクセス・ファイル) も結合可能なテーブルとみなしうる．

4. 各オプティマイザのJ方策間の比較

IBM の System R[7] は，左側の部分木 (Sub-tree) がつぎつぎと中間結果を表し，右側の部分木が一度に1回ずつ結合可能な単一の関係で構成されている．各左側部分木が結合の外部ループを，各右側部分木が内部ループを表すので，System R のP方策はその親のJ方策におけるテーブルの位置に依存している．

また，California 大学の INGRES システム[8]は，まず第1に最も効率のよい結合を試み，つぎに残りの結合を調べていく．INGRES の方策は中間結果のサイズがわかるため，System R よりも正確なコスト予想が可能である．一方，System R は最初の結合を行う前に全アクセス方策を考慮している．

INGRES の興味ある変種に Computer Corporation of America の ADAPLEX[2] がある．このオプティマイザは，はじめに最も効率のよい n 個の結合を行い，つぎに n 個の結合のすべての組み合せを試みる．INGRES のオプティマイザは，ADAPLEX の場合での $n=1$ のケースに相当する．

Reiner らのオプティマイザは，すべての結合木を考慮し，かつJ方策中の結合可能エンティティとして DAS やアクセス・リスト・テーブルを扱え，半結合 (Semijoin) および共通部分集合 (Intersection) の生成が可能である．

INGRES の拡張版のオプティマイザは Reiner らのアプローチと同様なものを採用しているが，索引の動的生成や CODASYL 型レコードやセット型の処理を考慮していない．

5. 単純だが強力なPマシン

Pマシンの基本命令は十分な抽象度をもち，データ・アクセスおよび操作の詳細を隠弊することが必要である．一方，基本命令は汎用で，分散システム，データベース・マシン，異なる複数のデータ・モデルをもつシステムのような多種のアクセス方策が支援できる低水準の機械も，併せもつことが求められる．

Pマシンは，テーブルに対する四つの基本命令，走査 (Scan)，結合，分類，DAS 生成 (Create-DAS) をもつ機械である．

結合命令には，併合走査法 (Merge-Scan)，入れ子走査などの結合法 (Nested-Loop Equijoin) のほかに入れ子ブロック法 (Nested Block) を組み込むことも考えられる．

これらの命令で操作するテーブルは，ディスク (d)，主記憶 (m)，ストリーム・ステート (s, 主記憶バッファ中の tuple) 上のいずれかにある．そして，走査命令はテーブルを一つの場所から他の場所に移動するもので，順次走査 (Sequential Scan) と DAS 走査 (DAS-Scan) の2種が考えられる．

前者はテーブルをストリーム・スナートから出し入れするもので (d→s, m→s, s→d, s→m)，後者は索引，ハッシュ・テーブル，CODASYL セット，他の DAS などを使用し，テーブルをストリーム・ステートにするものである (d→s, m→s)．

また，分類命令はテーブルを分類し (s→s)，DAS 生成命令はテーブルをキャッシングし (s→d, s→m)，副作用としてテーブル上にストリーム・ステート DAS を生成する．

このPマシンは，非第1正規形のデータの取り扱いこそできないが，ADAPLEX, INGRES, RDB/V1, P.G. Selinger, E-Worg などのほとんどのオプティマイザで利用できる．

6. 方策空間の生成と探索——効率の問題

P方策は計算開始時のテーブルを入力ノードとし，Pマシンの命令および中間結果のテーブルを内部ノードとする有向非巡回グラフで表現される．そこで，P方策空間は一つの問い合せに対する可能なすべてのグラフを重ね合せて実現できる．なお，重ね合せ時に同一の後続処理をもつ中間テーブルを同一のノードとする必要がある．

そして，この重ね合せたグラフに対しコスト準拠刈り込み (Cost-Based Pharing)，つまり動的計画法によって各中間テーブルに至る最も効率のよい探索経路を計算し，それを残し他の経路を刈り取る．このことによって，すべてのP方策を生成しなくても最もよい方策を発見できる．場合によっては，P方策空間の大きさと複雑性を制約する必要があり，このために比較的大きな中間テーブルを生成する結合に対するP方策の生成を遅らせたり，DAS と他のテーブルとの結合を制限したりするなどの方法が考えられている．

このほか，探索の効率を向上させるためには，入力問い合せストリームの特徴を利用したJ方策とP方策空間の動的チューニング技術や，資源利用状況にもとづくコスト・モデルのパラメタの調整などが有効であろう．

いずれにせよ，最適化に要する時間がP方策が実行されるのに必要な時間を超えないように閾値を設ける必要があろう．

参考文献

[1] D. Reiner, A. Rosenthal, *Strategy Space and Abstract Target Machines for Query Optimization*, Sperry Research Center, SRC-RP-82-34, August 1982.

[2] A. Chan, S. Fox, K. Lin, D. Ries, *The Design of the ADAPLEX DBMS*, Computer Corporation of America Technical Report CCA-09-81, 1981.

[3] R. Kooi, D. Frankforth, "Query Optimization in INGRES", *Database Engineering*, September 1982 (this issue).

[4] A. Makinouchi, et al, "The Optimization Strategy for Query Evaluation in RDB/VI", *Proc. 7th VLDB*, September 1981.

[5] A. Rosenthal, D. Reiner, "An Architecture for Query Optimization", ACM SIGMOD Conf., June 1982.

[6] A. Rosenthal, D. Reiner, "Querying Relational Views of Networks", COMPSAC 82 Conf., November 1982.

[7] P.G. Selinger, et al, "Access Path Selection in a Relational Database Management System", ACM SIMOD Conf., May-June 1979.

[8] E. Wong, K. Youssefi, "Decomposition-A Strategy for Query Processing", *ACM TODS.*, Vol. 1, No. 3, September 1976.

[9] S.B. Yao, "Optimization of Query Evaluation Algorithms", *ACM TODS.*, Vol. 4, No. 2, June 1979.

(ソフトウェア・プロダクト統括部 1100/90導入プロジェクト)

コマンド言語の設計における人間工学的配慮

M. L. Schneider

1. はじめに

コマンド言語に対する人間工学的配慮は，その設計の最初の段階から必要であり，人間とソフトウェアとのすべてのインタフェースが対象となる．このためインタフェースの設計上で考慮すべき点としては，これらがシステムに対するユーザの見方，ユーザに対する認知的（知覚・記憶・思考に関するCognitive）側面，システムでユーザが使用できるツール，ソフトウェア・システムに対する経験の蓄積とともに変化していくことなどがあげられる．

そこで，もしこれらの諸点への配慮がなおざりにされると，欠点のあるソフトウェアになる．以上の諸点を踏まえた上でコマンド言語の設計プロセスを考えると，そのプロセスは，

① ユーザによるソフトウェア・システムの概念的モデルの決定
② 技術レベルによるユーザのグループ分け
③ コマンド言語の構文の決定
④ コマンド言語のテストと評価

の各ステップに分けられよう．

そして，最初の二つのステップで，インタフェースの基礎が築かれる．つまり，システムの概念的モデルやユーザの集団を明確に理解しないと，システムの明確な記述は困難になるからである．

なおここで，概念的モデルはシステムの状態を，またユーザ集団の分析はインタフェースの構成諸技術（メニュ，コマンドなど）の決定を支援するのに役立つ．

2. 概念的モデル

システムの概念的モデルは，ユーザの視点からみたシステムの仕様である．概念的モデルの定義技術について詳細に論じた資料がないが，著者は次の三つのガイド・ラインがあるように思う．

① モデルの定義に当たっては多数のユーザの意見を参考にする．
② モデルの記述は簡潔にし，1～2ページで全容がつかめるようにする．
③ モデルは一般的なものを考え，コマンドの詳細までは規定しない．

そして，概念的モデルはエンティティ（対象と操作）を決定するために有用であり，新たにエンティティを追加する場合には，このモデルとの整合性がチェックされる．また，概念的モデルとエンティティとの関係は相補的であり，概念的モデルは定義済みのエンティティの吟味によって修正される．

3. ユーザの技術的能力による分類

ユーザのモデルは，ソフトウェア・システムと独立な一般的なものを定める必要がある．ユーザの技術的能力の発達のモデルを特徴付ける尺度ないし，基準として次のものをあげることができる．

① インタラクションのモード……インタラクションの主体がコンピュータかユーザか
② 対象の一般性……ユーザが考えている対象が，具体的かつ特定のものか，あるいは類似の特性をもつグループか
③ 操作の一般性……ユーザが考えている操作が個々かつ特定のものか，あるいは複数操作の組み合せ，および集合か
④ 予知性……インタラクションの発生前に，ユーザがどの程度のインタラクションを予期し制御できるか
⑤ 単位……ユーザによって操作可能な最大の構文単位はどの程度のものか
⑥ グループ化……小さな構文の項目をより大きな意味をもつ単位にグループ化する能力を，ユーザがどのようにして獲得できるか

そして，これらの基準によってコマンド言語の獲得能力の発達段階を

① 機械的にまねる人
② 初級（入門者）
③ 中級（中堅）
④ 上級（エキスパート）
⑤ 名人（マスター）

に分けることができる．

3.1 ユーザの技術的能力の発達段階

機械的にまねる人は，与えられたコマンドについて考えたり疑問に思ったりせずに，そのまま用い自分でコマンドを作れない段階である．

初級は，コマンドの項目についてのいくつかの個個の概念を理解し，特定のコマンドの語句を選べる段階である．

中級は，コマンド文の全体を取り扱える段階で，知識の単位は項目から文へと拡大し，文がいまや基本的な概念の単位となる．

上級は，システム中のどのような機能をも利用できる能力をもち，コマンド言語に通暁し，グローバルかつ抽象度の高い（メタ言語）レベルでコマンド言語を取り扱える段階である．この段階での対象の単位は，個々の文ではなくコマンド文のプログラム，あるいはプロシージャとなる．

名人は，コマンド言語の適用範囲をわきまえている．システムに組み込まれた機能や対象をそのまま用いたのでは解決できない問題に対しても，システムを拡張し，新しい対象と機能を作り出し，解く能力をもつ段階である．言い換えると，名人は有限のシステムを生成能力のあるシステムに変える能力をもつ段階といえよう．

これらのモデルを用いると，ユーザを特徴付けることやコマンド言語に必要なインタラクション技術のタイプが決められる．たとえば，初級と中級ではメニュ方式とコマンド方式といったように，それぞれに適した構文を選んだ方がよい場合もあるが，その場合でもセマンティクスは同一でなければならない．一方，中級と上級では構文は同じでよいが，上級の場合はセマンティクスはより広いものとなるだろう．また，システムの使用頻度の少ない初級や中級では，コマンド言語の学習・理解の容易性が求められ，一方，使用頻度の多い中級や上級では，よく使う機能を自動化することによるタスク実行の容易性が求められよう．

4. インタフェースの詳細

4.1 コマンドの構文

コマンド構文の決定に当たっては，心理学，および人間工学的配慮としてキーボードを操作する運動能力，コマンドの形式間での規則の整合性，一つのコマンド内における情報のグループ化による記憶のしやすさなどが求められる．そのほか，コマンド自身がドキュメントになる性質（セルフ・ドキュメンティング）をもち，構文単位が顕著に見分けられ，かつ思い出しやすいことも要求される．

キーボードの操作能力では，1キー・ストローク当たりの所要時間などが問題にされ，当然ながら繰り返しの少ない長い入力になるほど誤りが多くなる．

また，規則の不整合性は，あるソフトウェアに対し二つの相反する概念モデルが存在する場合や，コマンドの形式が複数の異なる規則（キーワード方式とポジショナル・パラメタ方式）を併用している場合などにおきる．これらの配慮とユーザ・モデルをもとにして，ポジショナル・パラメタ方式とキーワード方式の比較が可能となる．

4.2 コマンド名と対象名

一般にコマンド名は動詞であり，対象名はその目的格となる名詞である．コマンド名については曖昧さが少なく，自然言語では多用されていないものがよい．また，相反する動作については，構文的な対（ASSIGN と UNASSIGN）や意味的な対（GET, PUT）がよいとされている．

4.3 コマンドの省略型

コマンドの省略型の作成法としては，末尾の文字列を取り去る“丸め法（Truncation）”と，内部のいくつかの文字を除去する“詰め法（Contraction）”が代表的である．省略型の作成では，その一般的手順の学習のしやすさ，特定の語の省略型作成の容易さ，省略型から一般型への復元の容易さを考慮する必要がある．

また，省略型の作成法には，①単純な丸め，②中間母音の除去，③コマンド頭部のいくつかの文字の採用，④強調音の採用，⑤ユーザが任意に設定，がある．これらを評価したところ，単純な丸め法は学習および省略型作成が容易であり，強調音および母音除去法は一般型への復元性がよいことがわかった．

5. おわりに

本稿では，コマンド言語の設計に必要な四つのステップを略述した．

最初の二つのステップは概念的モデルとユーザ・モデルの定義で，これらはコマンド言語のインタフェーズを決定する際のフレームワークを与える．つまり概念的モデルは，システムの機能を，ユーザ・モデルはシステムの構文の大綱を決める基礎となる．

これらの設定が終ると，まず最初に主たる（多数を占める）水準のユーザを対象としたインタフェースが構築され，つぎに他の水準のユーザへと対象を拡大していくことになる．

さて，最後に会話型システムを対象とする人間工学の現状を述べると，コマンド文を対象とした研究は，すでに十分行われており，構文・コマンド名・省略型作成などに関し，ガイド・ラインが得られている．一方，メニュと対話法を対象とした研究は発展中であるが，まだガイド・ラインが設定できるまでに至っていない．

（講演：Ergonomic Considerations in the Design of Command Languages より）

（ソフトウェア・プロダクト統括部
ソフトウェア一部　渡部研一 訳）

機械翻訳

高橋　肇

1. 歴　史

コンピュータを翻訳に利用するというアイデアをはじめて提案したのは Rockefeller 財団の W.

Weaver と London の Birkbeck 大学の A.D. Booth である．彼らは，外国語を一種の暗号としてとらえ，第2次大戦中の暗号解読技術を外国語の解釈にも利用できると考えた．Weaver は，1949年に彼のアイデアをメモとしてまとめて関係者に配っており，これが引き金となって Washington 大学と California 大学の Los Angeles 校で研究グループが組織されている．

そして，1952年の春には，MIT で最初の研究集会が開かれ，秋の London での第7回国際言語会議でも機械翻訳がテーマとして取り上げられている．

さらに，1954年3月には情報交換誌 MT (Mechanical Translation) が発刊された．そして，この年に Georgetown 大学と IBM との共同による最初の露英翻訳システムのデモンストレーションが IBM 701 を使用して行われている．また，1955年にはソ連の科学アカデミーでも同様な試みが BESM コンピュータで行われている．

そして，1956年の MIT における第2回機械翻訳国際会議の頃には，欧米諸国，インド，日本でも機械翻訳の研究がはじめられている．

米国における機械翻訳の研究は，1960年代中頃には最盛期を迎えた．当時米国は，1957年にはじまるスプートニク・ショックによって科学技術の振興に最も力が注がれ，大量のロシア語の技術文献が翻訳された時代であった．このため，2千万ドルを越える予算が機械翻訳の研究に投じられている．

しかし，機械翻訳の壁は厚く，1960年代の中頃になっても，人間による翻訳と同等の質をもつ翻訳が期待できそうにもなかった．そこで，米国科学財団 (NSF) は J.R. Pierce を長とする自動言語処理諮問委員会 (ALPAC, Automatic Language Processing Advisory Committee) を組織し，機械翻訳の技術的可能性を調査させた．

ALPAC の報告は1966年に発表されたが，その内容は計算言語学の重要性は認めたものの，機械翻訳の実用性を否定するものであった．この報告書によって米国における機械翻訳の研究予算は大幅に削減され，1967年以後の機械翻訳の研究の中心は複数言語を利用するカナダと EC に移った．また，この報告書の影響は米国以外の国々の機械翻訳の研究にも影響を与え，この報告書が米国における機械翻訳のニーズが少ないという特殊事情を根拠にしているにもかかわらず，世界中の至るところで機械翻訳プロジェクトの予算が縮小されている．

そして，ALPAC 報告書以後，米国における機械翻訳の関心は，人間が介入する半自動翻訳システムとしての会話型翻訳システム，自動辞書 (Automated Dictionary) などの翻訳者援助システム，対象分野を限定した実用システム，および言語そのものの理解の研究へと向った．なかでも，言語理解の研究は活発に行われており，データベースの自然言語インタフェースなどでは一部実用化されている．

2. 機械翻訳システム

機械翻訳システムは，その設計方法の違いによって三つの世代に分類される．第1世代 (1955年〜1967年) は2国語間の翻訳を目的として原始言語から目的言語へ直接翻訳するもので，第2世代 (1967年〜) は多国語間の翻訳を考慮し，中間言語を導入し間接的に翻訳するものである．

また，第3世代 (1970年〜) は，1970年代から活発化した自然言語理解の成果を取り入れたシステムである．従来のシステムが構文解析を中心としたのに対して，この世代のシステムは意味解析を重視していた．

2.1 第1世代システム

第1世代の翻訳システムは当初，語対語の逐語訳として出発し，試行錯誤的な改良によって使用していたが，1960年代の前半には構文解析にもとづく実用的なシステムも開発されている．

たとえば，1950年代後半の Washington 大学の露英システム Automatic Language Translator Mark I は初期の代表例であり，この改良版である Mark II は1964年に米空軍 (USAF) の FTD (Foreign Technology Division) で採用され，1970年に SYSTRAN に置き換えられるまで実際に用いられた．

また，L. Dostert の率いる当時の米国最大の機械翻訳グループであった Georgetown 大学の Georgetown System (露語-英語) は1960年代前半の最も進んだシステムであり，1964年に米国原子力委員会 (AEC) の Oak-Ridge 国立研究所とイタリアの Ispra にある欧州原子力機構 (EURATOM) の研究所に引き渡され使用されている．

また，1960年代の後半には，LATSEC Inc. の P. Toma は，Georgetown System を大幅に改良し，SYSTRAN (露語-英語) を開発した．SYSTRAN は，1970年から米空軍にリリースされ，1975年にはアポロとソユーズのランデブーの際に活躍している．さらに，SYSTRAN は1976年には欧州原子力機構の Georgetown System を置き換えており，EC の欧州共同体委員会 (CEC) でも英-仏翻訳版が1976年から試用されている．

このほか，第1世代システムとして重要なものにLogos Corp. の LOGOS（英語-ベトナム語）がある．これはベトナム戦争の落し子として生まれたもので，米空軍に1971年にリリースされ，ベトナムパイロットの軍事教練マニュアルの翻訳に用いられた．

2.2 第2世代システム

第2世代の機械翻訳システムは多国語間の翻訳を考慮し，構文構造の変換のための中間プロセスや中間言語を導入して間接的翻訳を行うシステムである．第2世代のシステムの代表例としては，Texas大学の METALS（独語-英語），Montreal 大学の TAUM（英語-仏語），Grenoble 大学の GETA（旧称 CETA）（露語-仏語）などがある．

TAUM プロジェクトでは，TAUM を使って天気予報を対象とする機械翻訳システム TAUM-METEO を開発し，1976年から実際に放送で使用している．このほか同プロジェクトでは，カナダ空軍のジェット機の油圧系保守のための英文マニュアルを仏訳する TAUM—AVIATION を開発中である．

また，GETA の最新版 ARIANE 78 は，文法と解析プログラムを分離した多国語間翻訳システムであり，現在開発中の最も進んだシステムである．そして，GETA は EC 翻訳局の指導により1978年に発足した多国語間翻訳システム開発プロジェクト EUROTRA の枠組を与えた．なお EUROTRA は，英・独・仏・伊・蘭・デンマークの6か国語を相互に翻訳するシステムで，意味解析も一部取り入れ曖昧性の解消を目指している．

2.3 第3世代システム

第3世代のシステムは，1970年代に発展した自然言語理解における人工知能研究の成果を取り入れ，意味理解の問題まで立ち入り，代名詞の参照，省略語，多義語の問題を解決しようとするシステムである．

1973年における英国の Y. Wilks の NLUS（英語-仏語）などはそのような最初の試みである．このほか，Yale 大学の R.C. Schank らも分野を限定した（たとえば，テロリズムに関するニュースなど）機械翻訳システム（英語-スペイン語，など）を開発している．

なお，従来の人工知能研究の関心は自然言語インタフェースをもつ質問応答（Q/A）システムの開発にあり，機械翻訳の問題が直接取り上げられることはなかった．しかし，1980年代に入って1970年代の自然言語理解の研究成果を取り入れた実験的第3世代システムの試みが増えつつある．

2.4 現　状

現在，実用に供されているのは TAUM-METEO を除けばすべて第1世代のシステムであり，開発中の EUROTRA などのシステムでは第2世代システムを目指している．なお，第2世代のシステムでは不十分とはいえ意味解析の過程が組み込まれている．また，第3世代システムは，自然言語理解研究の発展に待つところが多く，実用化は今後の課題である．

さて，現在の機械翻訳の動向をみると，第1に第1世代システムの商用化があげられる．SYSTRAN や1970年代末に登場した Weidner System，AIPS System などの商用システムは，改良された第1世代のシステムである．このため，文法や辞書などのデータがプログラムと分離しにくいなどの欠点をもち保守がむずかしいが，膨大な単語数の辞書や複雑な文法を取り入れ，かつ最近発展の著しいワード・プロセシング機能を採用し，実用レベルにまで至っている．

また，第2に第2世代システムの実用化促進があり，カナダの TAUM プロジェクトや EC の EUROTRA プロジェクトのように大学の研究室レベルのプロジェクトを発展させ大規模実用システムの開発を行う傾向がうかがえる．

第3に人工知能で開発された自然言語理解の技法を取り入れたシステムの開発があり，武蔵野電気通信研究所，電子技術総合研究所，Yale 大学などでそのようなアプローチが行われている．

3. 日本における機械翻訳

日本での機械翻訳研究は1950年代中頃から開始され，1955年に九州大学で実験システムが作られている．なお，本格的な機械翻訳は，1959年の電気試験所（現在の電子技術総合研究所）による英文和訳システムであり，これは専用機「ヤマト」を使用していた．さらに，1960年代前半には京都大学により和文英訳などの実験が行われている．その後10年間は，機械翻訳は低調であったが，1970年代の後半になると再び活発化しはじめた．現在，京都大学，九州大学，大阪府立大学，電電公社武蔵野電気通信研究所，国際電電（KDD）研究所などが研究しており，このほか民間企業の研究所でも研究が行われるようになった．これらの民間の研究所では実用を目指した研究が多いのが特色である．

また，1982年には日本科学技術情報センター（JICST），電子技術総合研究所，京都大学などによって科学技術庁機械翻訳プロジェクトが発足し，現

在科学技術論文の抄録を対象とした日英・英日実用システムを開発中であり，京都大学が中心となって1985年までに実用化を目指してプロジェクトを進めている．なお，このシステムは従来知られている技術を集大成したものとされている．

このほか，最近，日本国内でもSYSTRANおよびWeidner Systemが，それぞれ日本システラン社およびブラビス・インターナショナル社によって商用化されつつある．

参考文献

[1] W. J. Hutchins, "Progress in Documentation, Machine Translation and Machine-aided Translation", *Journal of Documentation*, Vol. 34, No. 2, June 1978.

[2] ALPAC, *Language and Machines, Computers in Translation and Linguistics*, National Academy of Sciences, 1966.

[3] 「機械翻訳システムの調査研究」，日本電子工業振興協会，1982.

[4] 「機械翻訳システムの調査研究」，日本電子工業振興協会，1983.

[5] "特集：自然言語処理技術と言語理論"，電子技術総合研究所調査報告，第205号，1981.

[6] "特集：日本語処理"，日経エレクトロニクス No. 324, 8-29, 1983.

[7] "特集：日本語情報処理"，情報処理，Vol. 20, No. 10, 1979.

[8] V. Lawson, *Practical Experience of Machine Translation*, North-Holland, 1982.

[9] 長尾真，"機械翻訳"，電子通信学会誌，Vol. 65, No. 4, 1982.

[10] 長尾真，"科技庁機械翻訳プロジェクトの概要"，情報処理学会，自然言語処理研究会資料 38-2, 1983.

(技術企画部　テクニカル・パブリケーション室)

Jörg Siekmann and Graham Wrightson 編

Automation of Reasoning, volume 1. Classical Papers on Computational Logic 1957-1966 volume 2. Classical Papers on Computational Logic 1967-1970

Symbolic Computation Series, Springer-Verlag, XII+525 pp./XII+637 pp., 1983.

サブタイトルが示すように，本書は1970年以前に発表された多数の computational logic に関する論文から現在古典と考えられる論文を選択し集録した論文集である．大冊のため発表の年代によって volume 1, 2 の2分冊に分けられている．各分冊ともに発表年代順に論文を並べており，巻頭には該当する年代の状況を見通す通覧 (overview) の論文をおさめ，巻末には約30ページにのぼる Bibliography が付けられている．なお，volume 1 の通覧は；

M. Davis, "The Prehistory and Early History of Automated Deduction"

と，

S. Yu. Maslov, et al, "Mechanical Proof Search and the Theory of Logical Deduction in the USSR"

の二つであり，volume 2 の 通覧は，

L. Wos and L. Henschen, "Automated Theorem Proving 1965-1970"

である．

Computational logic は実際に計算機を利用し，各種の論理系や形式的理論における定理の証明を自動化することを目的として logic の構成的研究を行う分野であり，このいわゆる Leibniz の thesis を実現する上で実際的な意味で効率よく定理を証明する自動定理証明手続きや inference machine の研究開発は，とくに重要なテーマである．言い換えれば，computational logic の基本テーマは自動定理証明 (automated theorem proving, automated deduction) の研究であり，編者が述べるように本書には計算機用の定理証明アルゴリズムに関する論文が集録されている．computational logic は1965年の J. A. Robinson の resolution principle 以降急速に発展した分野であるが，初期には Herbrand の証明手続きの冗長性を除き combinatorial explosion を回避して，証明アルゴリズムの効率化を図る strategies や heuristics のアカデミックな研究が中心であった[1]．しかし1970年代に入るや，ようやく論理的推論を要する応用分野での定理証明システムの実装が進み，最近では automated reasoning assistant を用いてブール代数や有限半群の未解決問題を解いたり，論理回路設計を効率化する事例も報告されている[2]．定理証明システムは，プログラム合成[3,4]情報システムや Q/A[5]，reasoning assistant[2,6]，あるいは plan formation[7] 等のロボット技術など，いわゆる AI の多岐にわたる分野で基本的に重要であり応用も活発化している．また周知のように線形の resolution 手続きからは，Prolog などの論理プログラミングの概念が誕生している．

しかし，半面，1階の述語論理に限定しても，表現や推論則の点で十分な研究が進められているとは言い難い．たとえば，Gentzen の自然推論の体系と resolution inference の関係や，もとの Herbrand 手続きに内在する redundancy については W. Bibel の研究があるが[8]，見通しがよいとはいえず，今後の研究が待たれる．Bibel のように，もう一度 Herbrand の定理から証明手続きの効率を考え直すのも一つの方法であろう．定理証明を研究したり応用する際に，本書のような論文集が手もとにあれば大いに有用である．編者によれば，引き続いて1970年以降の論文集も編集する予定であるという．この続巻のタイムリな刊行を期待したい．

参考文献

[1] D. Loveland, *Automated Theorem Proving: A Logical Basis* North-Holland, 1978.

[2] L. Wos, "Solving open questions with an automated theorem-proving program", Keynote Adddress, *6th Conference on Automated Deduction* 1982, Lecture Notes in Computer Science 138, Springer-Verlag, 1982, pp. 1〜31.

[3] R. L. Constable, et al, *An Introduction to the PL/CV2 Programming Logic*, Lecture Notes in Computer Science 135, Springer-Verlag, 1982.

[4] M. J. Gordon, et al, *Edinburgh LCF*, Lecture Notes in Computer Science 78, Springer-Verlag, 1979.

[5] H. Gallaire and J. Minker, eds., *Logic and Data Bases*, Plenum Press など

[6] R. Weyhrauch, *Prolegomena to a theory of mechanized formal reasoning*, Artificial Intelligence 13, 1980, pp. 133-170.

[7] D. H. D. Warren, *WARPLAN: A system for Generating Plans*, DCL memo 76, University of Edinburgh, 1974.

[8] W. Bibel, *Automated Theorem Proving*, Vieweg, 1982.

コンピュータ関連の辞典と事典

現在刊行されているこの種の文献は，汎用計算機に携わっている専門技術者向けから，パソコンやマイコン関連の『早わかり…』といった一般ビジネス・マン向けにいたる非常に幅広い利用者層を対象にして，かなりの数にのぼる．書店の店頭において筆者の目に触れただけでも，和書が約40点，洋書約20点であった．最近異常なブームとなっているパソコンやマイコン関係については全体としてどれくらいの点数が刊行されているかは完全に把握できないほどである．

本稿では，いわゆる専門家向けの和洋書を紹介するが，あくまでも部分的例示に過ぎないことを，あらかじめお断りしておかなければならない．

辞典とか事典という名前が付いていても，実質的には，計算機の歴史，ハードウェア編，ソフトウェア編のように体系的に展開していって巻末の索引によって知りたい項目を引かせている文献や，英和対訳や和英対訳のように用語の対応日本語あるいは対応外国語だけを調べる目的のものなど形態も多様である．本稿では，説明の便宜上，文献をハンドブック，定義書，対訳集の三つに分けて例示することにする．

1. ハンドブックないし事典

[1a] *Encyclopedia of Computer Science and Engineering 2nd edition*, A. Ralston (ed), Van Nostrand Reinhold.

[1b] *Encyclopedia of Computer Sciences and Technology*, J. Belzer, et. al. (ed) Dekker.

[1c] *Computer Dictionary and Handbook, 3rd edition*, C. J & R. J. Sippl., Howard W. Sams & Co..

[1d]「総合コンピュータ辞典（第2版）」，日本ユニバック総合研究所編，共立出版．

[1e]「情報処理ハンドブック」，情報処理学会編，オーム社．

[1f]「電子通信ハンドブック」，電子通信学会編，オーム社，

[1g]「コンピュータの事典」，元岡編，朝倉書店．

[1h]「マイクロコンピュータの事典」，石井・田丸・山中編，朝倉書店．

[1i]「マイコン小事典」，講談社，

[1a] は，本体ページ数1578，索引を含めると1663ページにもなる．見出し語数は550であるが，たとえば，リスト処理について約15ページを割いているように，各用語に対して系統的に説明文が記述され，文中で出現する用語も巻末の索引で参照できるほか，各見出し語の最後には参考文献が掲げられている．（朝倉書店より翻訳版近刊予定）

[1b] は，全16巻より構成され数年を費やしてAの項目から刊行されたが，最新情報を盛り込むために第15巻を追補に当てている．第16巻は全ページが索引となっており，辞典というより教科書に近い性格をもつ．

[1c] と [1d] は，最初の部分が小項目の見出し語からなる用語編，後半の部分に解説編を配し，相互に参照できるようにはなっているが，用語編は単独の辞書としても利用できる．（[1d] については，本誌4号（1983）で詳しく紹介した）．

[1e] から [1h] は，コンピュータ技術の各分野についてその基礎から応用までを体系的に解説しており，巻末の索引を引くことによって辞書として利用できる．これらのうち [1g] と [1h] は，1983年に刊行されたもので内容的にも新しい．しかし [1e] から [1h] は，執筆者が多人数にわたっているため，内容的に重複する部分があったり，用語の不統一が目立ったりするのが気になる．

価格も手頃でブームに乗って売れているのが，[1i] である．初心者用にしてはやや難解であり，汎用計算機の経験者が辞書代わりに使おうとすると，やや物足りない．

2. 定義書

特定の用語の定義や意味を知りたい場合に，本来的な辞書として使うものである．この種の辞書では，限られた紙面で用語の意味が記述されるため，初心者にとっては他の用語との関連が理解できないという不具合が生じる．

[2a] *IEEE Standard Dictionary of Electrical and Electronics Terms*, IEEE Std 100-1977.

[2b] *Data Communications Dictionary*, C. J. Sippl (ed), Van Nostrand Reinhold.

[2c] *Dictionary of Computers, Chandor*, Penguin, 1977.

[2d] *Microcomputer Dictionary 2nd edition*, C. J. Sippl (ed), Howard W. Sams & Co..

[2e]「コンピュータ用語辞典」，講談社．

[2f]「JIS 用語辞典」，日本規格協会

[2g]「JIS 工業用語大辞典」，日本規格協会

[2h]「JIS 情報処理用語解説」, 情報処理学会編, 朝倉書店,

[2i] *The Devils DP Dictionary*, Stan Kelly-Bootle, McGrow-Hill.

[2a] は, 米国内の国内規格や業界規格の中で記載されている用語の定義をすべて網羅している. ただソフトウェア関連の用語は非常に少なく, 1980 年に追補版が刊行されたが, 1984 年にはマージされ第 2 版が出される予定である.

[2c] は, 定義調ではなく平易に解説しているため初心には有効である. しかし収録用語自身はやや古く, 英国英語の表記が採用されている. [2e] の原典であって日本版では市場性に合わせるために, 人工知能, マイクロコンピュータなどの項目が巻末附録として解説風に追加されている.

[2d] は, 収録語数約 5000, [1c] の用語編を修正加筆したうえ, マイクロコンピュータ関係の用語を追加したもので, 単独の辞書として十分使用に耐えうる内容構成となっている. (共立出版より翻訳近刊予定)

日本工業規格では, 各分野ごとに用語, 読み方, 意味, 対応外国語が規定されており, [2f] は産業界別にまとめたものである. 情報処理用語は, 基本・一般編と電気編の両方に現れる. [2f] は, 全部で 6 編あるが, これらを統合して 1 冊にしたのが [2g] である. JIS 情報処理用語は, ISO 用語集の完全翻訳版である. ISO 規格は各国の利害関係を調整した妥協の産物であり, 用語集も例外ではない.

また用語の意味よりも定義の方を重視したあまり, 法律の条文を読むようで, 初心者には難解である. 掲載されている用語も古いため, これらを初心者に理解させるために, 用語の定義に対して解説をしたのが [2h] である.

[2i] は, いわゆる『悪魔の辞典』である. ブラック・ユーモアあり, ワーグナーのジークフリートからの引用ありで著者の博学ぶりを感じさせられるが, 英語国民でないわれわれには本当の面白さがよくわからない. 時間があれば目を通してみるのも一興であろう.

3. 対訳集

コンピュータ業界ほど話しことばと書きことばが使い分けられている世界は他に見当たらない. コンピュータが誕生したのが欧米であり, ほとんどの用語が外来語であることを考えれば当然のことであろうが, 日常の会話で使っている用語をいざ技術文書に書くとき, どのように対応日本語に表現したらよいか困惑することがある. この種の辞典は, 対応日本語なり対応外国語 (ほとんど英語) なりを知りたい場合, 利用価値は多いと思われる.

[3a]「コンピュータ英和, 和英辞典」, 日本ユニバック編, 共立出版,

[3b]「電子通信, 英和和英辞典改訂版」, 平山編, 共立出版.

[3c]「英露和情報処理用語対訳集」現代工学社.

[3d]「コンピュータエンジニアリング用語 34000」インタプレス社.

[3e]「英和和英　学術用語に基づく科学技術用語辞典 (全 9 巻)」インタプレス社

[3f]「英和和英 JIS に基づく技術用語辞典 (全 5 巻)」, インタプレス社

[3g]「英和和英　情報処理用語辞典」, 日本理工出版会.

[3h]「日中計算技術用語集」, 科学出版社 (北京)

収録語数に関していえば, [3d]～[3f] が他を圧倒している. しかし, この種の辞書では対応日本語は世の中で使用されているものをすべて掲載しており, 中には自然陶汰されて消え去ったものもあろうから, きめ細かい保守作業が必要である.

[3h] の原名は,『日漢計算技術詞彙』である. 収録語数約 34,000 で, 巻末の日本語漢字索引 (画数順) から読みの平仮名を求め, 本体を参照することもできる. この辞典を見て感嘆するのは, わが国では外来語を片仮名書きにして使用するところを, 中国では徹底的に自国語の意味に言い換えて使用していることである.

[3a] は, 本来弊社内において製品マニュアルを執筆する際の標準用語集を発展させたもので, どちらかといえばソフトウェア関連の比重が大きい. 発刊は 1978 年であるが, これをもとにして社内版が 2 年ごとに改訂刊行されている. [3b] は, 情報処理のほかに回路技術, 通信, 数学などの周辺分野の用語も収録したもので, [3a] に比べて語数が多くなっている. [3c] は, 英語-ロシア語, ロシア語-英語の基本ラインに対応日本語を付与したもので, この分野の辞書としては特異なものである. 発刊時期が最も新しいのが [3g] であるが, 対応日本語にすべて出典が付けられているため, 少々うるさい気がしないでもない.

■ ■

●IFIP Congress '83——Paris, 1983年9月19日～23日

今回は40の招待講演，98の一般講演，32のパネル討論が以下のテーマで行われた．

まずハードウェアとアーキテクチャの分野では，新技術の発見で学んだ教訓，高速プロセッサで高性能を得る方法などのパネル討論，高性能汎用コンピュータのアーキテクチャ，第5世代コンピュータのアーキテクチャ，データフロー・マシンなどの講演．

ソフトウェアでは，大規模プログラミング言語対小規模プログラミング言語，プログラミング環境などのパネル討論のほか，Adaの設計，論理プログラミング，並列プログラムの仕様定義と設計，知識ベースにもとづくプログラミング環境，ドメイン・ベースドPROLOGのコンパイル，数学的思考によるプログラミング・システム，分散型制御流れのプログラミング，などの講演があった．

情報処理の理論的基礎では，並列プロセスの数学，理論コンピュータ科学の難問などのパネル討論，推論型プログラミング，データベース・システム用の汎用関係インタフェース，ペトリ・ネットと時制論理の採用，プログラム変換と最適化，帰納的推論によるプログラム合成，並列プログラムの意味論など．

ネットワークとコミュニケーションの分野では，ISO参照モデル，分散システム技術の動向，広域広帯域ネットワークなどのパネル討論と，プロトコル工学，パーソナル・コンピュータのローカル・ネットワーク，分散システムの原理，コミュニケーション・プロトコルの検証ツールなどの講演．

データベースと情報システムでは，データ・モデルと知識表現，データベース・マシン，などのパネル討論のほか，データベース・システムの回復アルゴリズム，分散型データベースの並行制御の性能評価，問い合せ評価，多重プロセッサ・データベース・マシン，演繹的データベースBDGENなど．

アプリケーションでは，要求分析定義技術の有効性，大規模アプリケーション・システムの管理技術と経験などと題し，パネル討論が開かれたほか，機械設計用のエキスパート・システム，など．

OA分野では，人間工学，オフィス情報システムにおける問題解決，などのパネル討論，オフィス作業の形式的および非形式的モデル，オフィス・オートメーションへの人工知能の応用，など．

マイクロプロセッサの応用では，ロボット工学，などのパネル討論とVLSIによる超高速並行処理，などがとりあげられた．

●COMPCON '83 Fall——Arlington, 1983年9月25日～29日

今回の基調テーマは，「エンド・ユーザにコンピュータ・パワーを」で，また，大会チュートリアル・セミナーが，生存可能コンピュータ・システム，1980年代のDBMS，ネットワーク・プロトコルの開発，ローカル・ネットワーク，ロボット工学とコンピュータ・ビジョン，ソフトウェア品質保証などをテーマとして行われた．

パネル討論でのテーマは高度情報データ・サービスISDNの進歩と見込み，次世代のグラフィック・ソフトウェア・ツール，コンピュータ・グラフィック標準，電子スプレッド・シート，大学におけるコンピュータ利用の新動向，オフィス・オートメーション・ツールの統合など．

なお，一般講演の主なテーマは，ローカル・エリア・ネットワークにおけるデータの遅延に及ぼすトポロジーの影響，ローカル・エリア・ネットワークの統合診断サブシステム，階層構造システムにおけるリング通信の実現，パケット交換ネットワークにおける流れと混雑の制御，キューイング・ネットワーク分析プログラムの実現，米国企業におけるビデオ会議，米国規格VIDEOTEX/TELETEXTのプレゼンテーション・レベル・プロトコルの開発，エンド・ユーザにサービスを提供する情報資源センター，分散型プログラミング用のプログラミング環境，多機能マイクロコンピュータ意思決定支援システムALPHA，メニュー選択設計，ウィンドウ・システム，小型コンピュータ・システムにおけるセキュリティ，状況評価用のエキスパート・システム，知識ベース意思決定支援システム，意味論的データ・モデル，自然言語データベース・アクセス，組立ロボット制御用高水準言語，3次元画像の産業応用，音声通信機能をもつパーソナル・コンピュータ・ネットワーク，並行LISP用の多重マイクロプロセッサ・システム，自然言語対話機能をもつCAIシステム，知識ベース・エキスパート・システム開発用のエンド・ユーザ指向言語などであった．

●情報処理学会第27回全国大会（昭和58年後期）——名古屋，1983年10月18日～20日

今回の特別講演と招待講演は，北原安定氏（電電公社）および島津康男氏（名大）によって，「高度

情報化社会へ向けて電気通信」,「自然現象のシミュレーション」と題し行われた．このほか,「コンピュータ・ビジョンと視覚情報処理」および「スーパーコンピュータへの期待」をテーマとしてパネル討論が開かれた．

また，一般講演では，約800件の論文が発表された．なお，その主なテーマは次のとおりである．

ソフトウェア工学分野では，仕様から Prolog プログラムを作成する一手法，実行可能な要求定義モデル，ソースコード管理プログラム，プログラム仕様の曖昧表明法，代数的仕様記述検証支援系，仕様書作成支援ツール，日本語をベースにした仕様記述言語 NBSG，仕様の図式表現用の対話型図形作成支援ツール，データ構造とオペレーションにもとづくプログラム自動生成法，ペトリネットによる並行プログラム合成，言語適応型プログラミング環境の設計，知識工学手法によるソフトウェア作成支援システム，Concurrent Lisp のプログラミング環境，検査網羅度測定システム，要求仕様言語によるテスト・データ・ジェネレータ，ソフトウェア・コスト・モデルの応用，ソフトウェア障害の分析，など．

人工知能分野では，コンピュータによる指将棋システム，Lisp をベースにした Fuzzy 集合処理システム，ユニフィケーションの並列時間計算，組み合せ子を用いた関数型言語，知識埋蔵オブジェクト指向言語 Monju，対象指向言語「菊」の仕様，Prolog への手続き的制御記述の導入，知識獲得とメタ推論，パソコンによる日本語知能言語，知識ベース用フルスクリーン・エディタ，類似ソフトウェアの選択機構を有するソフトウェア再利用，Prolog による知識ベース型シミュレーション・システム，パーソナル逐次型推論マシン Ψ，光学設計エキスパート・システムなど．

日本語分野では，Prolog による日本語文節 DOC の生成，日韓機械翻訳システム，露文理解システム，日英機械翻訳システム，UNIX 下での手書き入力エディタなど．

アーキテクチャ分野では，知識ベースマシン，数式処理コンピュータ FLATS, Lisp マシン ALPHA，マルチ CPU Lisp マシン SYNAPSE，抽象データ型支援計算機，データフロー方式 Prolog マシン，Prolog マシン PEK，高並列推論エンジン PIE，2進木構造にもとづくマルチコンピュータ・システム DON, Josephson 素子用高速除算方式，超立方体構造アレイプロセッサ NEBULA，科学技術計算用データ駆動計算機 (Sigma-1, EM-3)，ベクトル・コンパイラなど．

データベース分野では，総合学術情報システム ISIS，エンジニアリング・データ文書化支援システム，核酸塩基配列データベース・システム GENAS，関数型データ・モデル表現，フィード・バックを用いる文献検索法，など．

ネットワークおよびオフィス・システム分野では，遠隔運転のための運転用交信パス，DCNA 製品試験用記述言語，端末間文書通信処理方式，ローカル・ネットワークにおけるファイル・サーバ，FAX 端末を用いた帳票出力処理，ドキュメント通信システムなど．

グラフィックおよび画像処理分野では，Octree 3次元形状モデル，隠面消去用並列プロセッサ，ゲートアレイ用総合 CAD システム，マスタスライ CAD LAMBDA II，グラフィックスにおけるエイリアス除去，図面自動入力システム，頭蓋骨図形の構造解析，曖昧な顔の再現，フォント・パターンの拡大スムージング，一般ラプラシアンを用いた画像復元，などがとりあげられた．

■ ■

MEMORANDUM

〈Sperry 関係の論文・講演等の要約〉

●トランザクション処理のノンストップ化を実現するコンピュータ・システムのモデルを提案——1980年代中頃のコミュニケーション・プロダクトでは，無人化などの要請によってノンストップ化が最大の課題となる．本稿のモデルは，プロセッサ，ファイル・サーバ，プリント・サーバ，ネットワーク・プロセッサを構成要素として，それらを光ファイバ・分散データ・インタフェース（FDDI, Fiber Distributed Data Interface）で結合したものである．これらの構成要素は，ISO の開放システム接続参照モデルにもとづき開発される．なお，FDDI は ANSI の X３T９ 委員会で標準化を検討中のもので，周囲数キロメートルの圏内に分散しているプロセッサおよび高速ブロック移送機器間を広帯域の伝送路で接続するインタフェースである．媒体としては高速性（50メガビット/秒以上）を満たすため，光ファイバが使用され，障害の発生した各種要素の切り離しには液晶技術が採用される見込みである．

また，各サーバで提供されるコンピュータ資源はローカル・エリア・ネットワークを介してサーバと接続され，プログラマブル・ライン・モジュールは，ローカル・ネットワークを介してネットワーク・プロセッサと接続される．なお，このローカル・ネットワークは，20メガビット/秒の速度でよいため，縒り２線方式あるいは同軸ケーブルが用いられる．VLSI の利用上から後者を採用している．

なお，このシステムで最も重要なのはソフトウェア，とくにネットワーク・プロセッサおよびプログラマブル・ライン・モジュールにおけるソフトウェアである．（J. J. Reymond, “Non-Stop Network Processors for Non-Stop Transaction Processing”, Sperry Executive Center in St. Paul-de-Vence, 1983）

●人工知能の応用に関しサーベイ——人工知能（AI）の研究は25年の歴史をもつが，最近急速に関心を集めつつある．研究テーマとしては，人工知能の探索，知識表現，自動言語理解，音声理解，AI 研究用プログラミング言語，AI 応用研究（科学），AI 応用研究（医療），AI 応用研究（教育），自動プログラミング，自動演繹，視覚，学習・帰納，計画および問題解決などがあげられる．

現在 AI が注目を集めている背景に，AI の応用としての実用的なエキスパート・システムあるいは知識準拠システム（Knowledge-based System）の登場がある．エキスパート・システムのエッセンスは知識ベースで，エキスパート・システムの構築では，知識獲得に最も時間がかかり，かつ適切な知識表現の決定が最も重要な問題となる．エキスパート・システムを成功させる要因として，①限定され，はっきりと定義された領域を対象とすること，②エキスパートが十分な問題解決の知識をもっていること，③プロジェクトの開始から完了まで少なくとも１人のエキスパートが参加すること，などがあげられる．また，実用に耐えるシステムの構築における開発実績では，最低でも５人年要しているケースが多いことがわかる．

AI の将来の応用分野としては，①建築・土木（設計・計画・スケジューリング・管理），②設備（設計・監視・制御・診断・保守修繕・教育），③指揮統制（情報分析・計画・標的決定・通信），④兵器システム（目標同定・適応制御・電子戦争），⑤専門職（医療・法律・会計・経営・不動産・財務・工学，コンサルティング・教育・分析），⑥教育（授業・試験・診断・概念形成・知識獲得），⑦画像（写真解釈・地図作成・地理的問題解決），⑧ソフトウェア（教育・仕様作成・設計・本番・検査・保守），⑨家庭の娯楽および助言（知的ゲーム・家計・購入・買物・知的情報検索），⑩知的エージェント（コンピュータ準拠システムの利用に対する支援），⑪オフィス・オートメーション（知的システム），⑫プロセス制御（工場およびプラントの自動化），⑬探査（宇宙・試掘）などが考えられる．

また，Sperry 社は，AI 技術を暗号解釈，脅威評価，戦略的標的決定，航空管制，回路診断，VLSI 設計，機器故障診断，コンピュータ機器構成の選択，音声理解などへ利用することを考えている．現在すでにコンピュータ・コミュニケーション・システムの機器構成を決定するエキスパート・システム ORDER EDIT を Salt Lake City の工場で開発し利用している，なお，同システムは MAPPER を使って作成されている．（W. D. Mc Bee, “Artificial Intelligence: Application Perspectives”, June 23, 1983）

●コンピュータ・システムの能力計画用モデリング・パッケージ MVAP を開発——MVAP（Mean Value Approximation Package）は Sperry 社の Eagan 工場のマーケティング・テスト・サービス（MTS）で開発されたもので，MTS の提供するパ

フォーマンス・マネジメント・サービスの主要な解析的モデリング・ツールとして使用されている．MVAP は，Sperry Research Center のキューイング・ネットワーク・モデルの研究にもとづくもので，多重プロセッサ，優先付きサービス，制約された記憶容量，複雑な入出力システムをモデル化するシリーズ1100用会話型パッケージである．

MVAP では，積形キューイング・ネットワークの正確に近い解法だけでなく，積形でないキューイング・ネットワークの近似解法によるモデル化も可能にしている．そして，前者では，HIMVA (Heuristic Mean Value Analysis) として知られる方法を採用している．

MVAP を用いると，典型的なコンピュータ・システムのモデリングの場合で，処理時間は2～3秒，所要記憶容量は 60K 語で計算でき，しかも，かなり正確な結果を得られる．

また，このパッケージは，コンピュータ・システムのキューイング・ネットワーク・モデルを会話型で，作成・実行・修正・保存・再使用し，結果を表示できる．(P. M. Broderson, et al, "MVAP: A Modeling Package for Computer System Capacity Planning", 1983)

☆

技　　報
UNIVAC TECHNOLOGY REVIEW
No. 6

発　行　日	昭和 59 年 2 月 29 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 (大代表)
頒布価格	1,500 円
印　刷　所	三美印刷株式会社