

技 報

UNIVAC TECHNOLOGY REVIEW

1985年8月 第9号

論 文

流体力学解析における偽境界に関する一考察……………藤野 勉 1

報 告

線描きグラフィック・システムにおける

基本データ型の描象化の試み……………R. Lynn, N. L. Soong 13

狭間隙状態における静電気放電の電磁妨害特性

——ARPがEMIを決定する——……………本田昌實・川村雄克 32

ポスト・ストア・モードにおける

ディスク・キャッシュの飽和現象……………長尾秀実・永田浩一 44

FORTRAN コンバージョンにおける MACRO の適用……………小林五郎 60

ソフトウェア自動生成の実際……………市丸信子 76

TECHNOLOGY TREND

マルチリンク手順の概略と標準化動向……………庭山正幸 93

実施理論の研究について……………佐口 功 100

BOOKS……………104

NEW PRODUCTS……………107

EDITORS' NOTE……………表2

流体力学解析では、たとえば長大な水路の一部の流れの模様を知りたい場合があり、このとき流れの中に仮の仕切りを定めることが必要となる。このような仕切りを偽境界と呼び、条件式が定義された真の境界と区別する。藤野の流体力学解析における偽境界に関する一考察は、この偽境界のもついくつかの問題について、単なる数学的な解釈にとどまらず、流体力学の立場から考察を加えている。

グラフィック・データ型の形式的仕様を与える試みはすでにいくつかある。抽象データ型の概念は1970年代初期からあり、1970年代後期以来グラフィック・データ型にも応用されてきた。R. Lynnらの線描きグラフィック・システムにおける基本データ型の抽象化の試みは、CAD、ビジネス・グラフィック・システム、データ表現システムなどの線描きグラフィック・システムに現れるデータ型の形式的仕様を述べている。点、線分、折れ線、多角形、絵などのデータ型の代数的仕様を与え、絵の生成プロセス図式を扱っている。またグラフィック言語の例を使い、付録にはそのデータ型の形式的仕様を与えている。

最近、電子装置に対する静電気放電 (ESD) の影響が、あらゆる規模のシステムに顕著に見られる。ESD 事象によって周囲へ放射される電磁エネルギーは、導体の表面と空間を通して、衝撃的な電磁妨害 (EMI) となって伝搬する。空間を伝搬する EMI の影響は、電子装置に対する放電電流の直接的な経路がなくても電子装置が誤動作するという事実によってもわかる。本田・川村らの狭間隙状態における静電気放電の電磁妨害特性は、著者らが開発した ESD 検出器を使って、ESD によって生じる EMI の影響を調査した結果を紹介している。すなわち、人が何ら電氣的ショックを感じないような非常に低い帯電電圧 (約 3kV 以下) での ESD によってもコンピュータが誤動作すると報告している。この現象をさらにいくつかの既存する測定装置を使って検証した。その結果、ESD 事象に伴う EMI は放電電圧に比例するとは限らず、ARP (Amplitude Rate of charge of current Product) によって決定されることがわかった。

UNIVAC シリーズ 1100 のディスク・キャッシュは、昭和 52 年以来開発を続け、56 年 12 月に開発評価用サブ・システムを設置し、各種評価テストを開始した。さらにその後、入出力命令効率の測定結果を参考にして、より効率のよいディスク・キャッシュをめざして新たな開発を始めた。長尾・永田らのポスト・ストア・モードにおけるディスク・キャッシュの飽和現象は、この効率改善のための開発に参加した経験から、改善前のディスク・キャッシュにおける問題点を分析し、どのような対策を施し、どのような効果を得たかについて述べている。すなわち、ディスクに書き込む動作を効率的に行い、キャッシュ・メモリの空き領域を確保することによって、従来のディスク装置と比べ 3~5 倍、また当初開発されたディスク・キャッシュと比較して 1.5~3 倍の速さを実現したことを紹介している。

近年 UNIVAC シリーズ 1100 の FORTRAN バージョンの多様化が今後の保守の面から問題化しており、これらを解決すべく ASCII FORTRAN 10 R1A 以降への統一が強く求められている。この移行を推進するために、トランスレータをはじめとする移行ソフトウェアの必要性が高まり、MACRO システムを用いて短期間に保守性の高い移行ソフトウェアを開発し、数社に適用した。小林の FORTRAN コンバージョンにおける MACRO の適用では、開発手段である MACRO システムの概要と、移行ソフトウェアの機能および開発方法としての適用実績とその評価について報告している。

コンピュータ・システムの開発過程における力点は、プログラミング段階から仕様定義段階へと移り、仕様定義あるいは要求定義の段階をより的確に行うための方法論やツールに目が向けられてきた。こうした中で PASE 1100 は、プログラムの仕様書を正確にもれなく記述し、記述された通りのプログラムを作り、また常に仕様書とプログラム (COBOL) とドキュメントとの整合を保全するためにはどうすれば良いかという点に的を絞って開発された。市丸のソフトウェア自動生成の実際は、この PASE 1100 がどのようなものであるか、また自動生成というものをどのようにして実現しているかを紹介している。

論文

流体力学解析における偽境界に関する一考察

A Consideration on Pseudo Boundary in Hydrodynamic Analyses

藤野 勉

要約 流体力学解析では、たとえば長大な水路の一部の流れの模様を知りたい場合があり、このとき流れの中に仮の仕切りを定めることが必要となる。このような仕切りを偽境界と呼び、条件式が定義された真の境界と区別する。ここでは偽境界のもついくつかの問題について考察を行う。

Abstract In hydrodynamic analyses, for instance when we want to know the flow of long water channel at particular part, it is necessary to put some boundary in the channel. We call these boundary as pseudo boundary respectively the real boundary which have boundary conditions. In this paper, the author considered some problems referring to the pseudo boundary.

1. はじめに

偽境界とは解析の都合上、領域の中に仮に設けられる仕切りで、ここでは境界条件式は存在せず、領域内と同じ微分方程式が満たされる。このような条件式が陽に定義されている仕切りを真の境界または単に境界と呼び、偽境界と区別する。このように偽境界の仮定を必要とする連続体解析は流体力学解析に多く、たとえば弾性力学解析などではあまりその例を見ない。この理由は弾性力学では棒や板などの限られた大きさの対象を扱うことが多いのに対し、流体力学では水路内の流れのように長大な形状内の現象の解析を行うことがあるためである。このとき、長大な領域に対して、観察したい領域を中心に適当に分割し解析を行うことは、計算機経済上も必要なことで、ここに偽境界の設定を行う理由がある。

一般に、領域内で微分方程式、境界上で条件式が与えられている問題では、解は一意的に存在する。しかるに境界の一部に偽境界が存在する場合、すなわち開いた領域のときは解の一意性は保証されなくなる。したがって、数値計算によって一つの解が求められてもその存在に疑義がもたれるところである。このことについては、後述するように単なる数学的な解釈にとどまらず、物理的、流体力学的な考察によってその存在の評価を行うこととしたい。

2. 2次元、ポテンシャル流れ

偽境界の概念を理解するため、ここでは最も簡単な2次元、ポテンシャル流れについて考察する。一般に水路、河川などは細長く地上に横たわるので明確な入口・出口の境界をもたないのが普通である。また、仮に境界が定められていても領域全体が長大となり、コンピュータによる経済的な計算は不可能となる。このようなとき解析を必要とする部分を水路全体から切り取り、短い領域を設定することが有効である。

たとえば、図1に示されるように解析領域 S を狭んで偽境界 A-A, B-B を設ける。この場合、偽境界は2個となるが、たとえば A-A の近くは平行壁が続くので一様流が十分

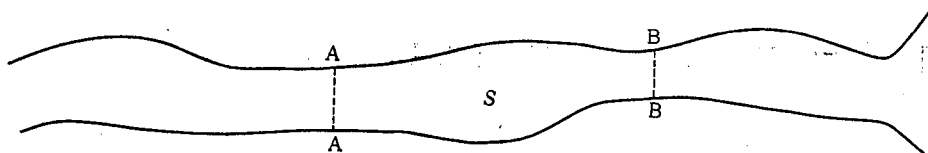


図 1 長大水路中の解析領域 S , 偽境界 A-A, B-B

Fig. 1 Analytical domain S and pseudo boundary A-A, B-B in a long water channel

な精度で形成されているものとして、この断面は真の境界とする。このように偽境界は1個のみとして、図2によって解の一意性について考察を行う。

いま x, y 方向の流速成分を u, v とし、流れ関数 $\phi(x, y)$ を

$$u = \partial\phi/\partial y, \quad v = -\partial\phi/\partial x \quad (2-1)$$

によって定義するとき、渦なしの条件によって

$$\zeta = \frac{\partial u}{\partial y} - \frac{\partial v}{\partial x} = \frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} = L(\phi) = 0 \quad (S \text{中}) \quad (2-2)$$

が満たされる。まず図2(a)により、境界はすべて固定とする

$$\phi = \bar{\phi} \quad (s \text{上}) \quad (2-3)$$

つぎに Green 積分によって次式を用意する。

$$\iint_S \left\{ \left(\frac{\partial \phi}{\partial x} \right)^2 + \left(\frac{\partial \phi}{\partial y} \right)^2 \right\} dx dy = \int_s \phi \phi_n ds - \iint_S L(\phi) \phi dx dy \quad (2-4)$$

いま式(2-2), (2-3)の解が2個存在するものとして、これを $\phi_1(x, y)$, $\phi_2(x, y)$ とし、 $\phi(x, y) = \phi_1(x, y) - \phi_2(x, y)$ とするとき、 $\phi(x, y)$ は式(2-2)ならびに境界条件式

$$\phi = 0 \quad (s \text{上}) \quad (2-5)$$

を満たす。したがって式(2-4)により、次式のようなになる。

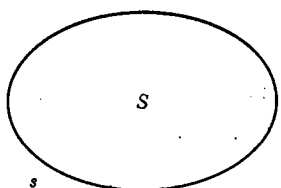
$$\iint_S \left\{ \left(\frac{\partial \phi}{\partial x} \right)^2 + \left(\frac{\partial \phi}{\partial y} \right)^2 \right\} dx dy = 0 \quad (2-6)$$

したがって領域内、境界上いたる所 $\phi = \phi_1 - \phi_2 = 0$ となり、解の一意性が保証される。

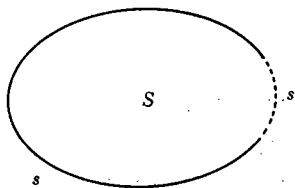
つぎに境界の一部に偽境界 s' をもつ場合は図2(b)より

$$\begin{aligned} \iint_S \left\{ \left(\frac{\partial \phi}{\partial x} \right)^2 + \left(\frac{\partial \phi}{\partial y} \right)^2 \right\} dx dy &= \int_s \phi \phi_n ds + \int_{s'} \phi \phi_n ds - \iint_S L(\phi) \phi dx dy \\ &= \int_{s'} \phi \phi_n ds \neq 0 \end{aligned} \quad (2-7)$$

となる。したがって、解の一意性は存在しない。



(a) 真の境界をもつ領域



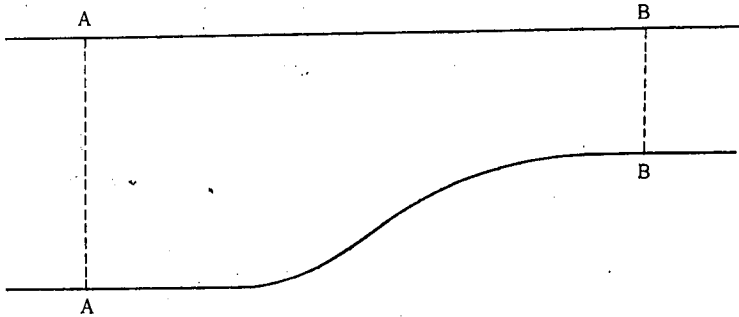
(b) 一部に偽境界をもつ領域

図 2. 境界ならびに偽境界によって囲まれる領域

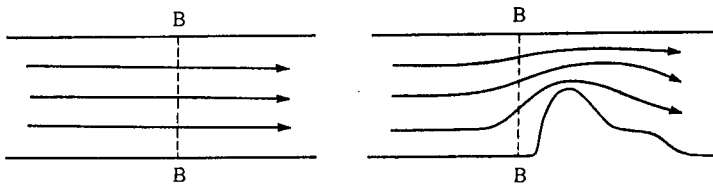
Fig. 2 A domain with boundary and pseudo boundary

このように数学的には、真の境界のみで囲まれた領域については解の一意性が存在し、一部に偽境界を含む場合は否定される、ことが証明された。しかしながら、このことは

界上いたる所 ϕ の分布が与えられたときに主張できる結論で、 ϕ の与え方に唯一性が欠けるときはその限りではない。たとえば図3に示される流路について考察する。同図で断面A-A、B-Bの近くではある程度平行流路が続き、ここでは一様流が仮定できるものとする。したがって、A-A、B-Bは境界で置き換えられる。このときコンピュータにはA-Aの上流、B-Bの下流の情報については何も知らされていない。もしA-Aのすぐ上流、またはB-Bのすぐ下流に急激な突起または凹みがあれば、ここでは一様流は形成されなくなる。このように解の一意性に関しては、偽境界を含むときと大同小異である。実際コンピュータに命令を与える人がB-Bに偽境界を設定するときは上流・下流を含め、その近傍に形状の急激な変化がないことを確認している。真の境界で置き換えるときは平行部がある程度存在するときに限られたが、偽境界のときはこれを緩和して、その近傍の形が緩やかに変化していればよい。これは解析の可能性を大きく広げるものである。このことは、いくつかの数値実験により証明することができる。このように偽境界による解は、下流も含めてその近くに形の急変がないとの期待感によるものである。



(a) 流路の形



(b) 偽境界 B-B 下流の形状

図3 二つの平行部をもつ流路に入口・出口境界の設定

Fig. 3 Putting inlet and outlet boundaries in a channel with two parallel part

つぎに、図3(b)に示されるように一部に偽境界をもつ2次元ポテンシャル流れの数値解析法について考察する。

2.1 有限要素法

一部に偽境界を含む2次元ポテンシャル流れについては、下記の変分原理が成立する。

$$\iint_S L(\phi) \delta \phi dx dy = \delta W - \delta U = 0; \quad \delta' W = \int_S \phi_n \delta \phi ds \quad (2-8)$$

いま領域全体を要素に分解し、その要素番号を e 、節点番号を i とし、その要素構成を

$$1^e = i, \quad 2^e = j, \quad 3^e = k \quad (2-9)$$

とする。ここに1, 2, 3は要素内節点番号で反時計方向に付番する。要素内分布を

$$\psi = \zeta_1^e \psi_1^e + \zeta_2^e \psi_2^e + \zeta_3^e \psi_3^e \quad (2-10)$$

とするとき、要素内部エネルギーは

$$V^e = \frac{1}{2} k_{mn}^e \psi_m^e \psi_n^e, \quad k_{mn}^e = A^e (b_m^e b_n^e + c_m^e c_n^e) \quad (2-11)$$

によって与えられ、偽境界を含む要素では下記の境界積分

$$\begin{aligned} \delta' W^e &= \int_{s, s'} \psi_n^e \delta \psi^e ds \\ &= -A^e \{ (b_1^e b_3^e + c_1^e c_3^e) \psi_1^e + (b_2^e b_3^e + c_2^e c_3^e) \psi_2^e \\ &\quad + (b_3^e b_3^e + c_3^e c_3^e) \psi_3^e \} (\delta \psi_1^e + \delta \psi_2^e) \end{aligned} \quad (2-12)$$

を行う。これらを全系に加算し

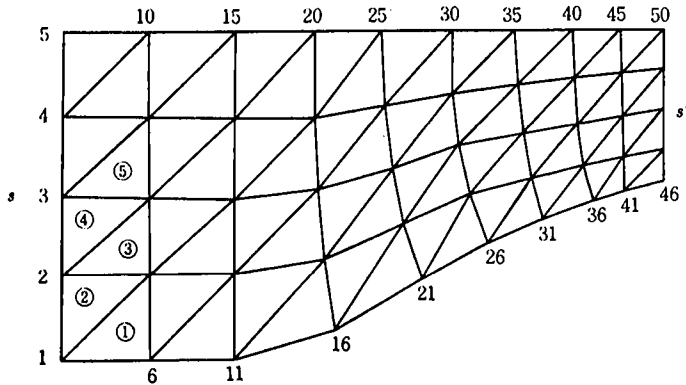
$$V = \sum_e V^e = \frac{1}{2} k_{ij} \psi_i \psi_j \quad (2-13)$$

$$-\int_{s'} \psi_n \delta \psi ds = -\delta' W = -\sum_e \delta' W^e = k'_{ij} \psi_j \delta \psi_i \quad (2-14)$$

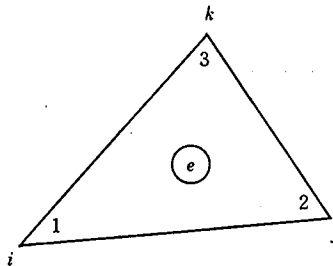
を得る。したがって、全系について下記の連立方程式を得る (図3参照)。

$$(k_{ij} + k'_{ij}) \psi_j = 0 \quad (S \text{ 中および } s' \text{ 上}) \quad (2-15)$$

ここで k_{ij} は対称であるが、 k'_{ij} は一般に非対称である。



(a) 要素分割



(b) 要素構成

図4 有限要素法による要素分割

Fig. 4 Mesh generation by finite element method

2.2 任意節点配置差分法

全領域を有限要素法と同様に曲線格子系によって分割し、その格子点に節点を置き、各節点ごとに2次の差分家族を定める。

差分家族構成 $1^i=i, 2^i=j, 3^i=k, 4^i=l, 5^i=m, 6^i=n$ (2-16)

ここに頭符 i は家族数号, i, j, k, l, m, n は構成家族節点番号, $1, 2, \dots, 6$ は家族内節点番号である。つぎに差分家族ごとに下記の計算を用意しておく。

$$\text{局所座標} \begin{cases} x_1^i=0, & x_2^i=x_j-x_i, & x_3^i=x_k-x_i \\ x_4^i=x_l-x_i, & x_5^i=x_m-x_i, & x_6^i=x_n-x_i \\ y_1^i=0, & y_2^i=y_j-y_i, & y_3^i=y_k-y_i, \\ y_4^i=y_l-y_i, & y_5^i=y_m-y_i, & y_6^i=y_n-y_i \end{cases} \quad (2-17)$$

$$f_{mn}^i=f_n(x_m^i, y_m^i) \quad (f_1=1, f_2=x, f_3=y, f_4=x^2/2, f_5=xy, f_6=y^2/2) \quad (2-18)$$

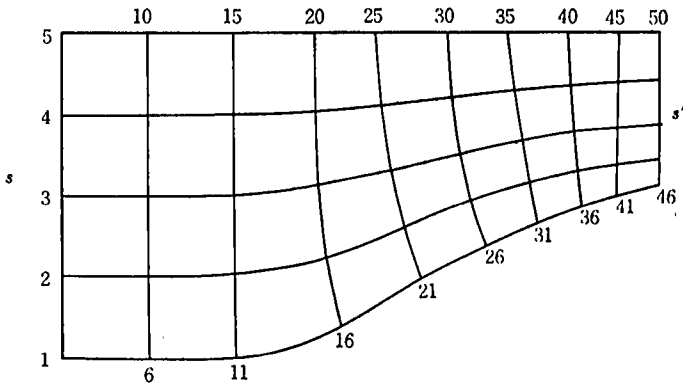
$$g_{mn}^i=f_{mn}^i \text{ の逆行列} \quad (2-19)$$

境界 s ならびに領域 S , 偽境界 s' 上では次の支配方程式が満たされる。

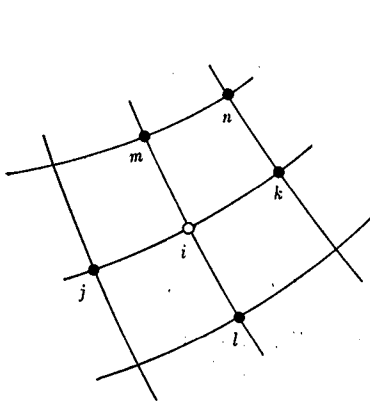
$$\phi_i=\bar{\phi}_i \quad (s \text{ 上}) \quad (2-20)$$

$$\frac{\partial^2 \phi}{\partial x^2} + \frac{\partial^2 \phi}{\partial y^2} \equiv a_4(\phi) + a_6(\phi) = 0 \quad (S \text{ 中および } s' \text{ 上}) \quad (2-21)$$

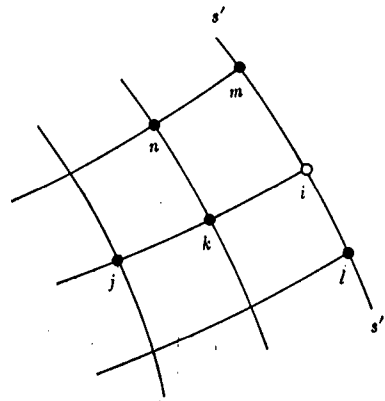
$$k_{ii}=1, \quad f_i=\bar{\phi}_i \quad (s \text{ 上}) \quad (2-22)$$



(a) 曲線格子系ならびに節点配置



(b) 領域内節点配置



(c) 偽境界上節点配置

図 5 任意節点配置差分法による節点配置

Fig. 5 Node distribution by the curvilinear mesh system

これらの式は図 5 に示す差分家族により, 次のように差分表示される。

$$\left. \begin{aligned} k_n^i &= g_{4n}^i + g_{6n}^i, \quad k_1^i = k_{ii}, \quad k_2^i = k_{ij}, \quad k_3^i = k_{ik}, \\ k_4^i &= k_{ii}, \quad k_5^i = k_{im}, \quad k_6^i = k_{in}, \quad f_i = 0 \end{aligned} \right\} \quad (S \text{ 中および } s' \text{ 上}) \quad (2-23)$$

以上により，下記連立方程式が導かれる。

$$k_{ii}\psi_i + k_{ij}\psi_j + k_{ik}\psi_k + k_{il}\psi_l + k_{im}\psi_m + k_{in}\psi_n = f_i \quad (i=1\sim I) \quad (2-24)$$

3. 2次元，粘性流体のクリーブ流れ

2次元 x - y 平面内における粘性流体の応力テンソルは

$$\sigma_{xx} = 2\mu \frac{\partial u}{\partial x} - p, \quad \sigma_{xy} = \mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right), \quad \sigma_{yy} = 2\mu \frac{\partial v}{\partial y} - p \quad (3-1)$$

によって与えられ，その流れは下記の運動方程式，連続方程式によって記述される。

$$\rho \frac{du}{dt} = \frac{\partial \sigma_{xx}}{\partial x} + \frac{\partial \sigma_{xy}}{\partial y}, \quad \rho \frac{dv}{dt} = \frac{\partial \sigma_{xy}}{\partial x} + \frac{\partial \sigma_{yy}}{\partial y}, \quad \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = L^3 = 0 \quad (3-2)$$

ただし u, v は x, y 方向の各流速成分， ρ は満度， p は圧力， μ は粘性係数で

$$\frac{d}{dt} = \frac{\partial}{\partial t} + u \frac{\partial}{\partial x} + v \frac{\partial}{\partial y} \quad (3-3)$$

を表す。ここでは定常流れを取り扱うものとし，かつ流速が小さいとして，慣性項は省略することとする。したがって，運動方程式は

$$\left. \begin{aligned} \frac{\partial}{\partial x} \left(2\mu \frac{\partial u}{\partial x} - p \right) + \frac{\partial}{\partial y} \mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) &\equiv L^1 = 0 \\ \frac{\partial}{\partial x} \mu \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right) + \frac{\partial}{\partial y} \left(2\mu \frac{\partial v}{\partial y} - p \right) &\equiv L^2 = 0 \end{aligned} \right\} \quad (3-4)$$

となる。上式 (3-4) と連続式 (3-2) により，次の構成方程式を仮定する。

$$\left\{ \begin{array}{l} \sigma_0^1 \\ \sigma_1^1 \\ \sigma_2^1 \\ \sigma_0^2 \\ \sigma_1^2 \\ \sigma_2^2 \\ \sigma_0^3 \\ \sigma_1^3 \\ \sigma_2^3 \end{array} \right\} = \left[\begin{array}{ccc} 2\mu & & -1 \\ & \mu & \mu \\ & \mu & \mu \\ & & 2\mu & -1 \\ -1 & & & -1 \end{array} \right] \left\{ \begin{array}{l} \varepsilon_0^1 = u \\ \varepsilon_1^1 = \partial u / \partial x \\ \varepsilon_2^1 = \partial u / \partial y \\ \varepsilon_0^2 = v \\ \varepsilon_1^2 = \partial v / \partial x \\ \varepsilon_2^2 = \partial v / \partial y \\ \varepsilon_0^3 = p \\ \varepsilon_1^3 = \partial p / \partial x \\ \varepsilon_2^3 = \partial p / \partial y \end{array} \right\} \quad (3-5)$$

上の構成方程式により，下記 Green 積分を行う。

$$\begin{aligned} \delta U &= \iint_S \sigma_m^p \delta \varepsilon_m^p dx dy = \iint_S \left(\sigma_1^1 \frac{\partial \delta u}{\partial x} + \sigma_2^1 \frac{\partial \delta u}{\partial y} + \sigma_1^2 \frac{\partial \delta v}{\partial x} + \sigma_2^2 \frac{\partial \delta v}{\partial y} + \sigma_0^3 \delta p \right) dx dy \\ &= \int_S (\sigma^1 \delta u + \sigma^2 \delta v) ds - \iint_S (L^1 \delta u + L^2 \delta v + L^3 \delta p) dx dy \end{aligned} \quad (3-6)$$

$$U = \iint_S \left[\frac{1}{2} \mu \left\{ 2 \left(\frac{\partial u}{\partial x} \right)^2 + \left(\frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} \right)^2 + 2 \left(\frac{\partial v}{\partial y} \right)^2 \right\} - \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) p \right] dx dy \quad (3-7)$$

$$\left. \begin{aligned} \sigma^1 &= l_x \sigma_1^1 + l_y \sigma_2^1, & \sigma^2 &= l_x \sigma_1^2 + l_y \sigma_2^2 \\ l_x &= dy/ds = \sin \theta, & l_y &= -dx/ds = -\cos \theta \end{aligned} \right\} \quad (3-8)$$

式 (3-7) にみられるように，内部エネルギー U は正定値ではない。また，構成方程式に示されるように σ_1^3, σ_2^3 は欠除している。このように，この3元の連立微分方程式は不完全である。このことは数値解析上の困難の原因と思われる。実際，式 (3-6) により境界条

件式として

$$u = \bar{u}, \quad v = \bar{v} \tag{3-9}$$

が与えられるが、 p に関する条件式は存在しない。これは σ_1^3, σ_2^3 が欠けていることによるものである。さらに Gauss の積分定理により

$$\iint_S \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) dx dy = \int_s (l_x u + l_y v) ds = 0 \tag{3-10}$$

が満たされなければならない。したがって、式 (3-9) における u, v はまったく任意に与えることはできず、式 (3-10) の拘束を受けることになる。このとき偽境界の設定は非常に有効である。

つぎに 2 次元粘性流体の解析法として、三つの方法について述べる。

3.1 原始変数法 (Primitive variable method)

ここでは u, v, p を原始変数と呼んでいる。とくに $\mu = \text{一定}$ の場合には微分方程式は

$$\left. \begin{aligned} L^1 &\equiv \mu \left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} \right) - \frac{\partial p}{\partial x} = 0, & L^2 &\equiv \mu \left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2} \right) - \frac{\partial p}{\partial y} = 0, \\ L^3 &\equiv \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0 \end{aligned} \right\} \tag{3-11}$$

とすることができ、境界上では式 (3-9) が満たされるものとする。したがって、領域内では $L^1 = 0, L^2 = 0, L^3 = 0$ 、境界上では $u = \bar{u}, v = \bar{v}$ が満たされる。当然 \bar{u}, \bar{v} は式 (3-10) を満たしていることが必要である。たとえば図 3 のような流路で、境界 A-A, B-B の近くで十分長い平行部が存在するときは、ここで Poiseuille 流れが形成されているものとして流速分布を正しく定めることができる。しかし断面 B-B が完全な平行壁でない場合には、ここで流速分布を定義することは困難で、つぎに述べる偽境界法が有効である。

いま図 3 に似た流路を図 6 に示す。ただし断面 B-B の壁は図 3 では平行であったが図 6 ではいくらかの傾きをもつものとする。なお、この近傍で激しい壁面の凹凸は存在しないものとする。領域 S の内部ならびに境界 s 上では上述と同様の式が満たされるが、偽境界 s' 上では領域 S の内部と同様の式が満たされる。このようにして偽境界上でも流速分布 u, v が定められ、Gauss の積分定理 (3-10) により

$$\iint_S \left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} \right) dx dy = \int_{s+s'} (l_x u + l_y v) ds = 0 \tag{3-12}$$

が成立する。

つぎに任意節点配置差分法による計算手順は、下記のとおりである。図 6 に示すような

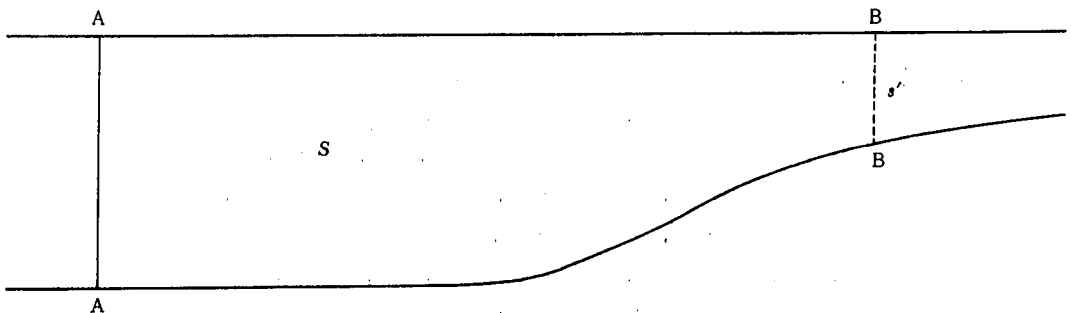


図 6 偽境界をもつ流路

Fig. 6 A channel with a pseudo boundary

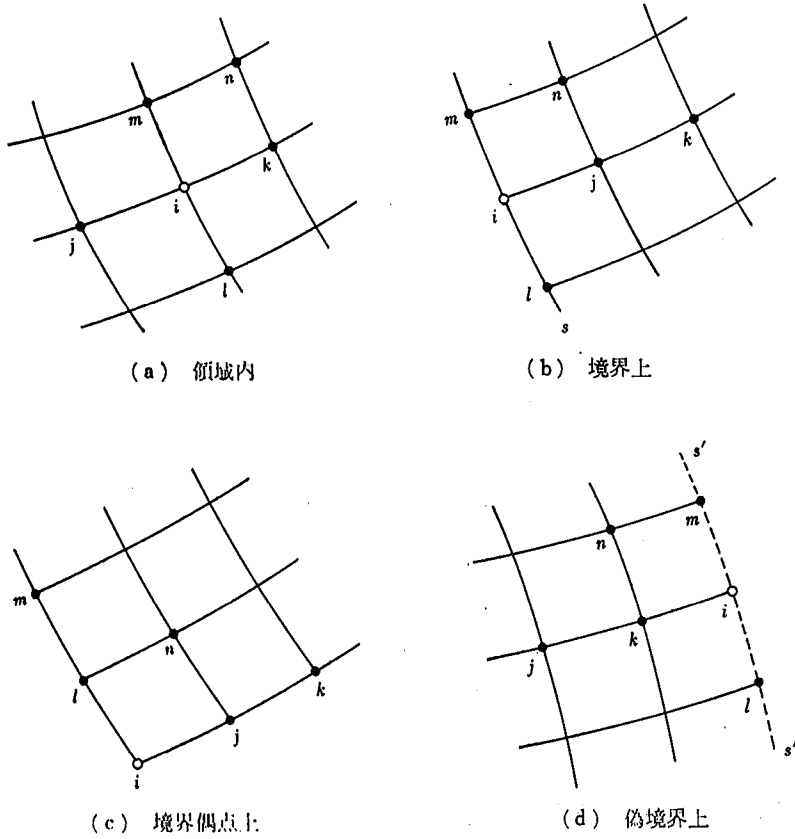


図 7 差分家族節点配置図

Fig. 7 Node distribution of a differential family

流路内に曲線格子系を設定し、その格子点に節点を置く。いま差分家族構成を式 (2-16) と同様定める。なお、局所座標 x^i, y^i 、行列 f_{mn}^i, g_{mn}^i も第 2 章に述べたとおりである。ただし、ここでは標準化を行うために $\mu \neq 0$ として、 $u^3 = p/\mu$ と置く ($u^1 = u, u^2 = v$)、

$$L^1 = \frac{\partial^2 u^1}{\partial x^2} + \frac{\partial^2 u^1}{\partial y^2} - \frac{\partial u^3}{\partial x} = 0, \quad L^2 = \frac{\partial^2 u^2}{\partial x^2} + \frac{\partial^2 u^2}{\partial y^2} - \frac{\partial u^3}{\partial y} = 0, \quad L^3 = \frac{\partial u^1}{\partial x} + \frac{\partial u^2}{\partial y} = 0$$

いま差分化された連立方程式を

$$\sum_{q=1}^3 k_{ij}{}^p{}^q u_j{}^q = f_i{}^p$$

と置くとき、係数行列は次のように与えられる。

$$\left. \begin{array}{l} \text{領域内} \\ \text{偽境界上} \end{array} \right\} \begin{cases} k_{ii}{}^{11} = g_{41}^i + g_{61}^i, & k_{ij}{}^{11} = g_{42}^i + g_{62}^i, & k_{ik}{}^{11} = g_{43}^i + g_{63}^i, \\ k_{ii}{}^{11} = g_{44}^i + g_{64}^i, & k_{im}{}^{11} = g_{45}^i + g_{65}^i, & k_{in}{}^{11} = g_{46}^i + g_{66}^i \\ k_{ii}{}^{13} = -g_{21}^i, & k_{ij}{}^{13} = -g_{22}^i, & k_{ik}{}^{13} = -g_{23}^i, \\ k_{ii}{}^{13} = -g_{24}^i, & k_{im}{}^{13} = -g_{25}^i, & k_{in}{}^{13} = -g_{26}^i, & f_i{}^1 = 0 \\ k_{ii}{}^{22} = g_{41}^i + g_{61}^i, & k_{ij}{}^{22} = g_{42}^i + g_{62}^i, & k_{ik}{}^{22} = g_{43}^i + g_{63}^i, \\ k_{ii}{}^{22} = g_{44}^i + g_{64}^i, & k_{im}{}^{22} = g_{45}^i + g_{65}^i, & k_{in}{}^{22} = g_{46}^i + g_{66}^i \\ k_{ii}{}^{23} = -g_{31}^i, & k_{ij}{}^{23} = -g_{32}^i, & k_{ik}{}^{23} = -g_{33}^i, \\ k_{ii}{}^{23} = -g_{34}^i, & k_{im}{}^{23} = -g_{35}^i, & k_{in}{}^{23} = -g_{36}^i, & f_i{}^2 = 0 \end{cases}$$

$$\begin{cases} k_{ii}^{31}=g_{21}^i, & k_{ij}^{31}=g_{22}^i, & k_{ik}^{31}=g_{23}^i, \\ k_{ii}^{31}=g_{24}^i, & k_{im}^{31}=g_{25}^i, & k_{in}^{31}=g_{26}^i \\ k_{ii}^{32}=g_{31}^i, & k_{ij}^{32}=g_{32}^i, & k_{ik}^{32}=g_{33}^i, \\ k_{ii}^{32}=g_{34}^i, & k_{im}^{32}=g_{35}^i, & k_{in}^{32}=g_{36}^i, & f_i^3=0 \end{cases}$$

境界上 $\begin{cases} k_{ii}^{11}=1 & f_i^1=\bar{u}_i \\ k_{ii}^{22}=1 & f_i^2=\bar{v}_i \end{cases}$

3.2 流れ関数法 (Stream function method)

原始変数法において, u, v が連続式を恒等的に満たすように

$$u = \partial\psi/\partial y, \quad v = -\partial\psi/\partial x \tag{3-13}$$

によって, 流れ関数 $\psi(x, y)$ を導入するとき, ψ は

$$L(\psi) = \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \right)^2 \psi \equiv \left(\frac{\partial^4}{\partial x^4} + 2 \frac{\partial^4}{\partial x^2 \partial y^2} + \frac{\partial^4}{\partial y^4} \right) \psi = 0 \tag{3-14}$$

を満たす. ここで, ψ に関し, 次の構成方程式

$$\begin{pmatrix} \sigma_{xx} \\ \sigma_{xy} \\ \sigma_{yx} \\ \sigma_{yy} \end{pmatrix} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix} \begin{pmatrix} \varepsilon_{xx} = \partial^2\psi/\partial x^2 \\ \varepsilon_{xy} = \partial^2\psi/\partial y\partial x \\ \varepsilon_{yx} = \partial^2\psi/\partial x\partial y \\ \varepsilon_{yy} = \partial^2\psi/\partial y^2 \end{pmatrix} \tag{3-15}$$

を仮定し, Green 積分を行う.

$$\delta U = \iint_S \sigma_{mn} \delta \varepsilon_{mn} dx dy = \int_S (\sigma_0 \delta\psi + \sigma_x \delta\psi_x + \sigma_y \delta\psi_y) ds + \iint L(\psi) \delta\psi dx dy \tag{3-16}$$

$$\left. \begin{aligned} \varepsilon_0 &= \psi, & \sigma_0 &= - \left(l_x \frac{\partial}{\partial x} + l_y \frac{\partial}{\partial y} \right) \left(\frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} \right) \\ \varepsilon_x &= \psi_x, & \sigma_x &= l_x \psi_{xx} + l_y \psi_{yx} \\ \varepsilon_y &= \psi_y, & \sigma_y &= l_x \psi_{xy} + l_y \psi_{yy} \end{aligned} \right\} \tag{3-17}$$

つぎに境界方向を s , 内方法線方向を t として, 式 (3-16) の境界積分を

$$\sigma_x \delta\psi_x + \sigma_y \delta\psi_y = \sigma_s \delta\psi_s + \sigma_t \delta\varepsilon_t \tag{3-18}$$

に変換する.

$$\left. \begin{aligned} \psi_x &= -(l_y \psi_s + l_x \psi_t), & \sigma_x &= -(l_y \sigma_s + l_x \sigma_t) \\ \psi_y &= l_x \psi_s - l_y \psi_t, & \sigma_y &= l_x \sigma_s - l_y \sigma_t \\ \psi_s &= -l_y \psi_x + l_x \psi_y, & \sigma_s &= -l_y \sigma_x + l_x \sigma_y \\ & & &= -l_x l_y \psi_{xx} + (l_x^2 - l_y^2) \psi_{xy} + l_x l_y \psi_{yy} \\ \psi_t &= -(l_x \psi_x + l_y \psi_y), & \sigma_t &= -(l_x \sigma_x + l_y \sigma_y) \\ & & &= -(l_x^2 \psi_{xx} + 2 l_x l_y \psi_{xy} + l_y^2 \psi_{yy}) \end{aligned} \right\} \tag{3-19}$$

さらに境界に沿って一回積分し, 次式のような

$$\int_S \sigma_s \delta\varepsilon_s ds = \int_S \sigma_s \frac{\partial \delta\psi}{\partial s} ds = - \int_S \sigma_{s,s} \delta\psi ds \tag{3-20}$$

変形を行うことにより, Green 積分は次のように変形される.

$$\delta U = \int_S (\sigma^* \delta\psi + \sigma_t \delta\psi_t) ds + \iint_S L(\psi) \delta\psi dx dy, \quad \sigma^* = \sigma_0 - \sigma_{s,s} \tag{3-21}$$

いま境界条件を

$$\left. \begin{aligned} \text{固定} & \quad \psi = \bar{\psi}, & \psi_t &= \bar{\psi}_t, & (s_1 \text{ 上}) \\ \text{自由} & \quad \sigma^* = \bar{\sigma}^*, & \sigma_t &= \bar{\sigma}_t, & (s_2 \text{ 上}) \end{aligned} \right\} \tag{3-22}$$

とし, 外部エネルギーを

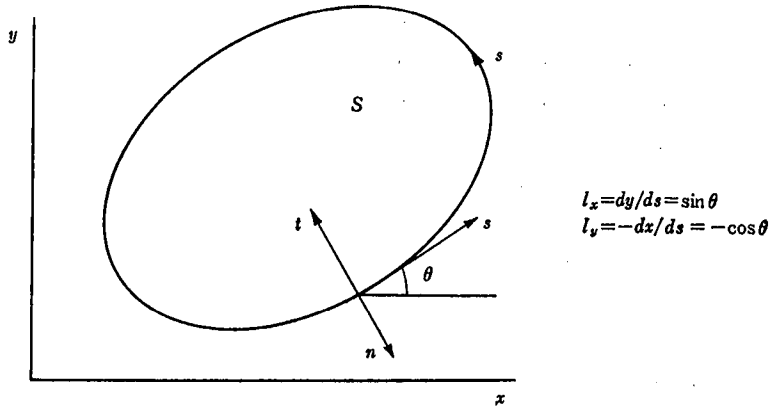


図 8 境界における切線 s ならびに内, 外法線 t, n
 Fig. 8 Tangential line s , inward and outward normal t, n at boundary

$$W = \int_{s_1}^{s_2} (\bar{\sigma}^* \bar{\psi} - \bar{\sigma}_i \psi_i) ds \tag{3-23}$$

によって定義するとき, 下記の変分原理が導かれる.

$$\delta U - \delta W = \int_{s_1}^{s_2} \{(\sigma^* - \bar{\sigma}^*) \delta \psi + (\sigma_i - \bar{\sigma}_i) \delta \psi_i\} ds + \iint_S L(\psi) \delta \psi dx dy = 0 \tag{3-24}$$

実際には境界上で σ^*, σ_i が既知であることは少ないので, ここでは境界はすべて固定と仮定する.

なお, 境界の一部に偽境界 s' を含む場合, 境界 s 上では固定の条件

$$\psi = \bar{\psi}, \quad \psi_i = \bar{\psi}_i \quad (s \text{ 上})$$

が満たされ, 領域 S 内ならびに偽境界 s' 上では

$$L(\psi) = \frac{\partial^4 \psi}{\partial x^4} + 2 \frac{\partial^4 \psi}{\partial x^2 \partial y^2} + \frac{\partial^4 \psi}{\partial y^4} = 0 \tag{3-25}$$

が満たされるものとする. 任意節点配置差分法で解く場合は, 境界の外側に一列のダミー節点を設置することが必要である.

3.3 流れ関数, 渦度関数法 (Stream and vorticity function method)

流れ関数による支配方程式 (3-25) は渦度関数

$$\zeta = \frac{\partial u}{\partial y} - \frac{\partial v}{\partial x} \tag{3-26}$$

を導入することによって

$$L^1 \equiv \frac{\partial^2 \zeta}{\partial x^2} + \frac{\partial^2 \zeta}{\partial y^2} = 0, \quad L^2 \equiv \frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} - \zeta = 0 \tag{3-27}$$

に分解される. 上式は下記の構成方程式

$$\begin{pmatrix} \sigma_0^1 \\ \sigma_x^1 \\ \sigma_y^1 \\ \sigma_0^2 \\ \sigma_x^2 \\ \sigma_y^2 \end{pmatrix} = \begin{pmatrix} & & & & & \\ & & & & & \\ & & & 1 & & \\ & & & & & \\ & & & & & 1 \\ & & 1 & & & \\ & & & & & \\ & & & & & 1 \end{pmatrix} \begin{pmatrix} \varepsilon_0^1 = \psi \\ \varepsilon_x^1 = \partial \psi / \partial x \\ \varepsilon_y^1 = \partial \psi / \partial y \\ \varepsilon_0^2 = \zeta \\ \varepsilon_x^2 = \partial \zeta / \partial x \\ \varepsilon_y^2 = \partial \zeta / \partial y \end{pmatrix} \tag{3-28}$$

と Green 積分

$$\delta U = \iint_S \sigma_m^p \delta \varepsilon_m^p dx dy = \int_S (\sigma^1 \delta \psi + \sigma^2 \delta \zeta) ds - \iint_S (L^1 \delta \psi + L^2 \delta \zeta) dx dy \quad (3-29)$$

に変形される。ただし、

$$\sigma^1 = l_x \zeta_x + l_y \zeta_y \equiv \zeta_n, \quad \sigma^2 = l_x \psi_x + l_y \psi_y \equiv \psi_n \quad (3-30)$$

$$L^1 = \frac{\partial^2 \zeta}{\partial x^2} + \frac{\partial^2 \zeta}{\partial y^2}, \quad L^2 = \frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} - \zeta \quad (3-31)$$

境界では \$\zeta\$ に関する情報を知ることが困難な場合が多いので、ここでは条件式

$$\psi = \bar{\psi}, \quad \psi_n = \bar{\psi}_n \quad (3-32)$$

が採用されるものとする。いま外部エネルギーを

$$W = \int_S \bar{\psi}_n \zeta ds \quad (3-33)$$

として、変分原理

$$\delta U - \delta W = \int_S \{ \zeta_n \delta \psi + (\psi_n - \bar{\psi}_n) \delta \zeta \} ds - \iint_S (L^1 \delta \psi + L^2 \delta \zeta) dx dy = 0 \quad (3-34)$$

を導く。なお内部エネルギーは次式で表される。

$$U = \iint_S \left(\frac{\partial \psi}{\partial x} \frac{\partial \zeta}{\partial x} + \frac{\partial \psi}{\partial y} \frac{\partial \zeta}{\partial y} + \frac{1}{2} \zeta^2 \right) dx dy \quad (3-35)$$

境界の一部に偽境界を含む場合は、境界 \$s\$ 上では条件式 (3-32)、領域 \$S\$ の内部ならびに偽境界 \$s'\$ の上では支配方程式 \$L^1=0, L^2=0\$ が満たされる。

■任意節点配置差分法による解析

領域の内部に境界に適合するように曲線格子系をつくり、その格子点に節点を置く。これらの節点はすべて \$\zeta, \psi\$ の 0 次節点である。差分化された連立方程式を

$$k_{ij}{}^{11} \psi_j + k_{ij}{}^{12} \zeta_j = f_i{}^1, \quad k_{ij}{}^{21} \psi_j + k_{ij}{}^{22} \zeta_j = f_i{}^2 \quad (3-36)$$

とするとき、領域内、偽境界上では

$$\left. \begin{aligned} k_{ii}{}^{12} = k_{ii}{}^{21} = g_{41}{}^i + g_{61}{}^i, \quad k_{ij}{}^{12} = k_{ij}{}^{21} = g_{42}{}^i + g_{62}{}^i, \quad k_{ik}{}^{12} = k_{ik}{}^{21} = g_{43}{}^i + g_{63}{}^i, \\ k_{ii}{}^{12} = k_{ii}{}^{21} = g_{44}{}^i + g_{64}{}^i, \quad k_{im}{}^{12} = k_{im}{}^{21} = g_{45}{}^i + g_{65}{}^i, \quad k_{in}{}^{12} = k_{in}{}^{21} = g_{46}{}^i + g_{66}{}^i \\ k_{ii}{}^{22} = -1, \quad f_i{}^1 = 0, \quad f_i{}^2 = 0 \end{aligned} \right\} \quad (3-37)$$

境界 \$s\$ 上では

$$k_{ii}{}^{11} = k_{ii}{}^{22} = 1, \quad f_i{}^1 = \bar{\psi}_i, \quad f_i{}^2 = (\bar{\psi}_n)_i \quad (3-38)$$

によって係数行列、定数項が与えられる。

つぎに、他の構成方程式による変分原理について考察する。

$$\left\{ \begin{array}{l} \sigma_0^1 \\ \sigma_x^1 \\ \sigma_y^1 \\ \sigma_0^2 \\ \sigma_x^2 \\ \sigma_y^2 \end{array} \right\} = \left[\begin{array}{ccc} & & 1 \\ & 1 & \\ & & 1 \\ & & & & 1 \\ & & & & & 1 \end{array} \right] \left\{ \begin{array}{l} \varepsilon_0^1 = \psi \\ \varepsilon_x^1 = \partial \psi / \partial x \\ \varepsilon_y^1 = \partial \psi / \partial y \\ \varepsilon_0^2 = \zeta \\ \varepsilon_x^2 = \partial \zeta / \partial x \\ \varepsilon_y^2 = \partial \zeta / \partial y \end{array} \right\} \quad (3-39)$$

$$\delta U + \iint_S \zeta \delta \psi dx dy = \iint_S \sigma_m^p \delta \varepsilon_m^p dx dy = \int_S (\psi_n \delta \psi + \zeta_n \delta \zeta) ds - \iint_S (L^1 \delta \psi + L^2 \delta \zeta) dx dy \quad (3-40)$$

$$L^1 = \frac{\partial^2 \psi}{\partial x^2} + \frac{\partial^2 \psi}{\partial y^2} - \zeta, \quad L^2 = \frac{\partial^2 \zeta}{\partial x^2} + \frac{\partial^2 \zeta}{\partial y^2} \quad (3-41)$$

$$U = \frac{1}{2} \iint_S \left\{ \left(\frac{\partial \psi}{\partial x} \right)^2 + \left(\frac{\partial \psi}{\partial y} \right)^2 + \left(\frac{\partial \zeta}{\partial x} \right)^2 + \left(\frac{\partial \zeta}{\partial y} \right)^2 \right\} dx dy \quad (3-42)$$

すべての境界が固定境界で

$$\phi = \bar{\phi}, \quad \zeta = \bar{\zeta} \quad s \text{ 上} \quad (3-43)$$

が与えられていれば、領域内で

$$L^1=0, \quad L^2=0 \quad (3-44)$$

が満されるものとして解を求めることができる。しかし、たとえば壁面では $\phi = \bar{\phi}$ であるが ζ に関する情報は欠除している。このような場合には ζ に関し、この境界も偽境界と考えるべきである。すなわち、ここでは

$$\phi = \bar{\phi}, \quad L^2=0 \quad (3-45)$$

が満されるものとする。なお領域内ならびに偽境界上では、式(3-44)が満される。

4. お わ り に

以上、偽境界の意義ならびにその有効性について述べてきたが、この方法は純数学的な論拠に多少の疑義をいだく人が多いと思われるが、筆者は流体力学的立場による考察をしていただくよう読者に望むものである。なお本稿では、実際の数値計算による裏付けを行っていないが、次の機会にこの責任を果たしたいと思っている。また、本稿では粘性流体のクリープ流れについてのみ考察したが、実際はさらに慣性項の導入による非線形性により、数値計算上別の困難な問題が存在するが、ここでは取り扱っていない。

なお、原始変数法、流れ関数-渦度関数法では、差分化による連立方程式の性質が不良で、数値計算上の困難の原因となっている。したがって、これらについては基本的に式の改善を行うことが必要と思われる。

- 参考文献 [1] M. D. Olson, "Composition of various finite element solution method for the Navier-Stokes equations", *International Conference on finite elements in water resources Part II* July 1976.

執筆紹介 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・流体力学等の解析法(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年特任教授を歴任。51年より日本ユニパック(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)等の著書がある。



報告 線描きグラフィック・システムにおける 基本データ型の抽象化の試み

An Abstraction of the Fundamental Data Types of Line Drawing Graphics System

R. Lynn, N. L. Soong

要約 本稿では CAD, CADD, ビジネス・グラフィック・システム, データ表現システムなどの線描きグラフィック・システムに現れるデータ型の形式的仕様を述べる。点, 線分, 折れ線, 多角形, 絵などのデータ型の代数的仕様を与え, 絵の生成プロセス図式を扱う。またグラフィック言語の例を使い, この技法を例示する。

Abstract This article gives a formal specification of the data types that are popular with line drawing graphics systems such as CAD systems, CADD systems, business graphics systems, data representation systems, and others. An algebraic technique is used to specify the data types, such as points, lines, polylines, polygons, pictures, etc. Also treated is the schema of a picture generating process. A sample graphics language is used to illustrate this technique.

1. はじめに

本稿では線描きシステムの基礎的諸概念の厳密な意味論的記述を提示する。この記述はグラフィック・データ型を理解し, 実際に形式的仕様を与えるのに役立つ。付録Aにホストグラフィック言語の例を使い, 付録Bにそのデータ型の形式的仕様を与える。また, われわれの抽象化の体験も述べる。

すでにグラフィック・データ型の形式的仕様を与える試みはいくつかある。抽象データ型の概念は1970年代初期から存在し^[21], 1970年代後期以来, グラフィック・データ型にも応用されてきた^[7, 25, 29]。しかし, そのほとんどは商用グラフィック・システムをあまり考慮しない非現実的な設定で論じられている^[1, 3, 19]。一方, 写植のグラフィックのデータ型は1982年に van Wyk によって研究されている^[30]。

本稿では線描きグラフィック・システムの完全な形式的仕様を与える。われわれの体験については, 2章で要約するが, とくに線描きシステムの階層的性質を使い, 形式的仕様を単純化できた。例に使うグラフィック言語の概要を示し, 言語に含める特徴についてはそれを含める理由を説明する。この選択は, それが一般的か否か, 主なデータ型か否か, あるいは将来重要となるか否か, を基準としている。また付録Bに示す形式的仕様の例を与えるために仕様言語を用いる。結論の節では, われわれの経験から次の諸点を要約して述べる。すなわち,

- 1) 付録Bの折れ線と多角形のデータ型の公理の設定
- 2) システムの融通性を保存しながら仕様を単純化するような階層的グラフィック・システムの構成
- 3) ホスト言語とそのグラフィック用拡張との間の束縛 (binding) 機構の例示
- 4) 様相的 (modal) グラフィック・システムの有限状態機械 (Moore 機械) によるモ

デル化

- 5) モジュール化した仕様言語の使用
- 6) 例に示す形式的仕様が定めるのに十分な意味論的知識ベースの設定と将来の研究のための基礎
- 7) 数学的な関数記法と引数をもつ手続き宣言における引数の束縛の相違点

2. 展 望

絵の表現法としては CORE^[1], GKS^[10] や多数のその変形版^[5, 27] のグラフィック言語に反映されている現実の方法に従う。絵は線分および面分の集合と従来の工学的な方法から導かれる諸概念からなる。また、ホスト言語にグラフィック機能を含めて拡張する問題を考えるが、この問題は、FORTRAN に単純なグラフィック言語を含める拡張で例示する。点、線分、折れ線などの点を基本とする対象を初等幾何の対象として扱い、グラフィック・データ型とする。

一方、点はグラフィック・モニタ上の一つの画素として、あるいはプロッタ・ベッド上の物理的点として実際の装置へ写像されるが、この実際の写像にはふれない。

数学的には、線の両端は無限に伸び、線分には二つの有限の端点がある。しかし線描きグラフィック・システムの場合には、線は普通線分を意味し、また基本対象はそれに伴う属性、たとえば点には色、線には線のスタイルのような属性がある。線を点集合と考えれば、線の抽象化ではすべての線が共有する同じ特徴をとらえる。この場合、線の特徴は“まっすぐである”ということであるから、適当な公理を選択してこの事実をとらえる。

言語の束縛はデータ型宣言の重要な部分である。束縛はデータ型を参照する外部の表現にデータ型の意味を与えるが、グラフィック言語の場合には、この束縛はホスト言語への拡張の形をとる。それぞれのホスト言語は互いに異なる構文をもつから、一般に同じデータ型の同じ演算に対してもそれぞれ異なった束縛を定めなければならない。面倒でも束縛がなければ、データ型の宣言は実際のプログラム言語と結び付かないから、永久に抽象と実現は分離したままにとどまる。例に示すグラフィック言語では FORTRAN の束縛を考える。従来、データ型はプログラム言語の一部で、処理系は整数、浮動小数点実数、固定小数点実数などのデータ型を暗黙の内に支援し、データ型の意味は言語に組み入れられている。またデータ型の構文上の束縛も異なる場合がある。たとえば、数のデータ型上の加算は $X+Y$ の形の式や、ADD X to Y の形や ADD(X, Y) のような関数の形で表現される。もしホスト言語の束縛と無関係にデータ型の意味を定義できれば、ホスト言語ごとにデータ型の抽象を行うといった労力は最小にできるから、このような定義は非常に望ましい。

整数のデータ型の場合には厳密に整数を定義する公理を束縛と独立に与えられるので、言語ごとに加算の構文上の束縛をそれぞれ設ける必要はない。以上より：

- 1) 束縛と無関係なスタイルで公理を構成するのが望ましい。例に示す形式的仕様では、公理を、束縛に関係する公理と関係しない公理との二つのクラスに分割する。
- 2) 座標系は一般的に扱うべきで、宣言を公理の形に書き換えなくても、次のシステムのいずれかで実現できるようにする。
 - a) R^2 または R^3 空間[†]
 - b) R 上の複素数

† (訳注) R は実数体を表す。

- 3) (円柱座標のような) 非線形の座標空間は、興味ある研究材料であるが、すべての考察から除外する。
- 4) プログラム言語で使えるすべての演算は各データ型に制限する(束縛する)。
- 5) 抽象機械の状態とデータ型の属性との間の概念上の区別はしばしば漠然としている。グラフィック・システムはグラフィック開始状態やグラフィック終了状態などの状態集合の下で動作するから、有限状態機械でモデル化できる。属性は、点の色、線の幅などのデータ型に伴う一つ以上の性質であるから、属性と状態は形式的に同一視できる。そこで Moore 機械を使い、データ型に伴う属性とグラフィック・システムの状態とをモデル化する。
- 6) 外部演算子を除けば、抽象化で言語の束縛に関係する演算はない。束縛と関係する演算子に関する公理は、それ自体束縛と関係する。

仕様システムでは基本的に数学的記法を用いる。それ自身の仕様が必要なソース仕様言語でオブジェクトとなるプログラム言語の仕様を定めるといった曖昧な循環に陥らないようにするため、擬似(pseudo)プログラム言語の類いは使用しない。また宣言のスタイルは代数的である。

データ型の抽象化で基本となる方法は集合論であり、公理は述語論理と関数計算^[4]の形式で記述する。

汎用線描きグラフィック・システムの意味は複雑であり、それぞれスタイルの点でも構成規則の点でも互いに異なっている。たとえば、従来通り頂点集合を反時計回りでたどる方法に対して、ランダムな順序で線分を与えても、折れ線は描ける。しかし、すべての折れ線が共有する同じ基本構造には点や線分という対象が暗に含まれていることを、これは示している。これらの性質は公理に反映すべきである。

3. 階 層 構 造

抽象グラフィック・システムには、点、線、折れ線、多角形などのデータ型の間に階層構造がある。この階層は、ホスト言語とは独立した自然な意味論的階層であり、例に使う単純なグラフィック言語にも反映させるが、CORE, GKS, NAPLAPS^[9], IGES^[20]などの複雑なシステムにも存在している。点はベクトル空間の一つのベクトルで表され、座標値をもつ。線分は点の線形集合で、STYLE (LINE) などの属性をもつことができる。しかし、線分は COLOR (LINE) のような属性はもたず、色を線分に伝えるのは属性 COLOR (POINT (LINE)) である。

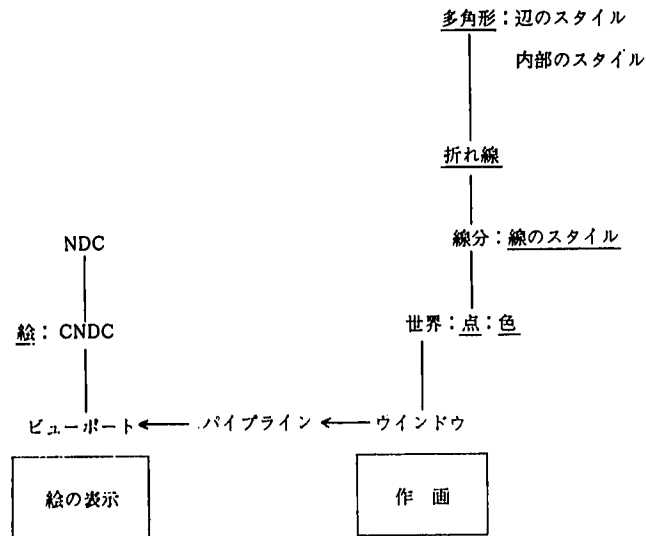
折れ線は(閉じた、または開いた)ディジーチェーン[†]をなす線分の集合である。線分や折れ線は基本的には点の線形集合であるが、多角形は空間内の点集合と考えられる。多角形の点集合は2次元または3次元の集合であり、COREやGKSのスタイルでは、2次元点集合である。したがって、多角形は線分より1次元だけ次元が高い。まとめると、次のように結論できる。

- 1) データ型は属性をもつ。たとえば次のごとくである。
 - 点は色や輝度をもつ。
 - 線はスタイルや幅をもつ。
 - 多角形は辺のスタイルや内部のスタイルをもつ。
- 2) 絵の表示関数は絵のモデルの作画関数とは独立に扱うことができる。また、これら

† (訳注) 線分の端点で連結した線分の集合。

の関数はパイプライン変換(イラスト)で互いに関係付けられる。モデルの構成には作画関数を使い、表示の操作には表示関数を使うが、これらの関数の独立性によって、グラフィック・システムの意味構造を単純化できる。データ型はモデルの作画か表示のいずれかに関係する。

- 3) 実際には、グラフィック・システムを装置と独立した部分と装置に関係する部分とに分割できる場合が多い。これらの部分は別々に扱えるが、システムとしては一緒に働く。本稿では装置と独立した部分を考える。
- 4) データ型の自然な階層と上述の分割とによって、抽象の内部を単純化できた。次の図は例に使うグラフィック言語で示した二つの独立した階層を表している。下線を引いた項はホスト言語からアクセスできるデータ型であり、矢印は射影の方向を示している。またコロンは対象と対象あるいは属性の関係を示している。(例に使うグラフィック言語では、線分は外部からアクセスできないデータ型としている点を注意せよ)。



4. グラフィック言語の例

付録Aに示すグラフィック言語は2次元の骨格のモデル化と表示操作を支援する。これを使ってデータ型の抽象を例示する。この言語は既存のグラフィックの言語の設計に見られる主要な要素を含んでおり、COREやGKSのスタイルを踏襲していて、シンプル・グラフィック・パッケージ(SGP)^[6,27]とも似ている。以下にこの言語の基本的な特徴をいくつか述べる。

- 1) 絶対アドレス・モードを使う。
- 2) 座標系の階層を支援する。
 - a) すべてのモデル化は世界座標(World Coordinate(WC))空間の中で行う。
 - b) すべての表示は正規化した装置座標(Normalized Device Coordinate(NDC))空間、またはその部分空間である現在のNDC(CNDC)空間で行う。
 - c) すべての装置のアドレッシングは離散的な装置座標(Device Coordinate)空間で行う。
- 3) パイプライン(PIPELINE)はWC空間内の絵をNDC空間内の絵に写像する絵の射影関数である。

- 4) すべての属性は様相的 (modal) であり, リセットするまで属性値はセットされ続ける。

モデルとその表示は同一の対象ではなく, 互いにパイプラインで結ばれる対象である。エンド・ユーザは対象をモデルの環境で作画するが, 連続的 WC 空間にあるモデルは有限的な表示装置には適さない形をしている。ウィンドウ (window) と呼ぶ (枠線で) 区切った参照空間を使って絵のモデルの一部分を表示するが, 2次元ではウィンドウは座標軸に平行な辺で囲まれた長方形の空間である。一つのウィンドウ内に含まれるグラフィック情報はビューポート (viewport) 上に写像されるが, このビューポートはまた NDC 空間の中の区切られた長方形である。NDC 空間の全部または一部が物理的表示装置で表示できる。また NDC 空間では複数の互いに交わらないビューポートを定義できる。この写像の列をパイプライン変換あるいは単にパイプライン^[6, 27]ともいう。

エンド・ユーザの作画は, すべて WC 空間の絶対アドレスを使って2個の作画関数 DRAW-POLYLINE と DRAW-POLYGON によって行う。折れ線と多角形が例に使うグラフィック言語で支援している基本対象であり, ポリマーカ (polymarker) やテキスト等の他の対象は使わない。CORE と GKS はこのように作画を折れ線と多角形のデータ型だけで行う商用グラフィック言語の典型的な例である。

基本対象にはいくつかの属性が付く。本稿ではいくつかの属性を調べて, 様相的属性を形式的に扱う方法を示す。折れ線のデータ型は線分のスタイル, 色, 幅を属性とする。また多角形のデータ型は埋め色 (fillcolor), 辺のスタイル, および埋め方のスタイル (fillstyle) を属性とする。現存のグラフィック言語の設計では様相的システムを, すなわち, もしある属性値がセットされているならば, それがりセットされるまで, セットが続くようなシステムを好んで採用している。色の属性は, 色あい, 輝度, 濃度で与えられ, 線分のスタイル値は, ソリッド, ドット, などで与えられる。ウィンドウ定義, ビューポート定義, あるいは使用する NDC 空間の定義などの制御パラメタは様相的状态変数である。別の理由で, このグラフィック言語には問合せ関数[†]が含まれてある。これらの問合せ関数はグラフィック言語の作画能力やモデル化能力には直接関係せず, 望ましいスタイルのプログラム開発法を支援する。これらの関数によって, プログラム言語での概念と数学での概念が必ずしも合致しないことがわかる。一例をあげると, 手続き宣言の場合, 引数には入力変数と出力変数が混じっていてもよいが, 数学での関数記法には出力変数という考え方はない。

たとえば, 三角関数 $\text{SIN}(X)$ は, プログラムでの関数の参照 `INQUIRE-LINESTYLE(L)` とは異なった意味をもっている[†]。

したがって, 返す値の束縛の違いから, 代数的スタイルの構文が担われることになる。実際, $1+\text{SIN}(X)$ は数学的に正しい演算を表すが, $1+\text{INQUIRE-LINESTYLE}(L)$ は誤った演算の定義である。データ型の形式的抽象化の基礎を代数に置くならば, 問合せ関数の束縛は数学での関数記法の純粹性の弱点をあらわにする。このため, 手続き宣言を支援するには有向グラフの概念を使う必要がある。

5. データ型仕様言語

抽象データ型の仕様言語には決定的な設計基準はない。しかし一つの傾向がみられ

† (訳注) 数学的な関数 SIN はタイプ $\{\text{integer}\} \rightarrow \{\text{integer}\}$ の関数であり, 整数値を返すが, 関数 `INQUIRE-LINESTYLE` はタイプ `state-space` \rightarrow `output-space` の関数であり, 返すものは `LINESTYLE` を表示する副作用である。

る^[14, 25, 31]。仕様言語に束縛を導入するとき、新しい問題がいくつか発生する。例題の形式的仕様を作り出すのに使う構文法には、その内のいくつかの必要条件を採り入れている。すなわち、仕様言語には次の概念が組み込まれている。

- 1) データ型の宣言
- 2) 集合をなす対象の宣言
- 3) 内部演算子の宣言
- 4) 外部演算子の宣言
- 5) プログラム言語のルーチンの形式的宣言
- 6) 公理の指定
- 7) 定理の集合

また、仕様言語には次の知識のクラスが組み込まれている。

- 1) 記法……二つの新しい記号を導入する。記号“:=:”は二つの対象が記号表現として等しいことを表す。“:=:”は記号“=”とは異なる。二つの対象が互いに等しいためには、それらは同じクラスが両立するクラスに入っていないなければならない。“:=:”は代入規則によって二つの対象が記号列として等しいことを表す。この記号を使って対象に名前を付けることもできる。関数は一般に $F: X \rightarrow Y$ あるいは $Y = F(X)$ と表すが、問合せ関数の場合には、FORTRAN の関数 INQUIRE-LINestyle (X) のように、 $F(X)$ の引数は出力パラメタを表す。記法 $F; X \leftarrow Y$ および $F^{-1}(X)$ で値域と定義域を入れ換えた関数を表す。一般にプログラム言語の引数束縛は代数的記法に従わない。
- 2) 集合論の応用……仕様言語では、集合、部分集合、空間などの概念を扱う。また集合の間の関係も扱う。集合の操作はメタ言語の形で次のように定義される。

UNION : $\langle \text{SET} \rangle \dots \rightarrow \langle \text{SET} \rangle$;

INTERSECT : $\langle \text{SET} \rangle \dots \rightarrow \langle \text{SET} \rangle$;

COMPLEMENT : $\langle \text{SET} \rangle \rightarrow \langle \text{SET} \rangle$;

記号“...”は複数個の項を示し、角カッコ“ $\langle \rangle$ ”は対象のあるクラスを表す。たとえば $\langle \text{SET} \rangle$ はすべての集合のクラスを表す。

- 3) 基本データ型……仕様言語には実数、整数、論理値などのデータ型がある。丸カッコを優先順位の曖昧さを除くために使う。次の表現でデータ型上の演算を表す。
 実数、整数に対して、 $+$, $-$, $*$, $/$, $<$, $=$, $>$, \leq , \geq
 論理値に対して、AND, OR, NOT
- 4) 線形ベクトル空間……仕様言語では n 次元線形ベクトル空間を n 次元ベクトルの集合として扱い、 n 次元ベクトルは基本データ型の n 個の対象の順序付配列として扱う。次のベクトルの加算、スカラーの乗算、次元、関数 Min を線形ベクトル空間上の基本演算とする：

V-ADD : $\langle \text{Vector-Space} \rangle \times \langle \text{Vector-Space} \rangle \rightarrow \langle \text{Vector-Space} \rangle$;

S-MULT : $\langle \text{Real} \rangle \times \langle \text{Vector-Space} \rangle \rightarrow \langle \text{Vector-Space} \rangle$;

: $\langle \text{Integer} \rangle \times \langle \text{Vector-Space} \rangle \rightarrow \langle \text{Vector-Space} \rangle$;

DIM : $\langle \text{Vector} \rangle \rightarrow \langle \text{Integer} \rangle$;

Min : $\langle \text{Vector} \rangle \rightarrow \langle \text{Integer} \rangle \mid \langle \text{Real} \rangle$;

内積と距離空間の概念は使わない。スカラーは、関係 $\text{DIM}(\text{Scalar}) = 1$ を満たす特別のベクトルである。行列はベクトル上の関数、 $M: \text{Vector} \rightarrow \text{Vector}$ である。行列の乗算と加

算はそれぞれ $M1(M2(M3(V)))$ と $V-ADD (M1(V), M2(V))$ のように表現できる。

キーワード 'Let' で始まる文を使って、対象の具体例をシステムに宣言する。たとえば

Let $P \in WC-Space$.

この階層的スタイルの宣言によって、各データ型が一つのモジュールとなるように宣言の有効範囲が定まる。

データ型のレベル（レベル1）での対象の宣言は、同じ仕様の内部にある他のデータ型からアクセスできる。内部の宣言は公理や演算子の宣言にも書くことができるが、これらの宣言はそれ自身のデータ型の有効範囲の中だけでアクセスできる。したがって、レベル1の宣言は一つのデータ型の仕様全体にわたる基本作業集合を与え、一方、内部の宣言は仕様を読んだり、書いたり、またそれに手を加えるときに表記を簡潔にする。また、注釈はすべて五つの星印*****の対で囲うことにする。

内部と外部の演算子宣言は宣言したデータ型についての意味情報を示す記号を指定する。内部演算子はすべて、他のデータ型や外部のプログラムからは見えないが、外部演算子は他のデータ型から見える。このスコープ・ルールは ALGOL のスコープ・ルールに似ている。

実際には、グラフィック言語はあるホスト言語の拡張である場合が多いが、外部ルーチン宣言によってホスト言語をデータ型に束縛することができる。データ型の一つの集合に対して、いろいろなホスト言語用の外部ルーチンを多数宣言できる。たとえば、折れ線を描く関数 POLYLINE (N, X-Array, Y-Array) の FORTRAN の関数参照が折れ線データ型に外部ルーチンとして与えられると、この FORTRAN の束縛機構を通して関数 POLYLINE は折れ線データ型の仕様に束縛される。複数の束縛機構を通して、PASCAL などの他のホスト言語を同一のデータ型仕様の集合に束縛することもできる。スコープ・ルールが満たされるならば、一つのデータ型仕様の中で内部データ型を作ることもできる。たとえば、例に使うグラフィック言語では線分に関するプログラムを支援していないから、例題の仕様に線分のデータ型は要らないのだが、折れ線のデータ型の中に線分の概念があれば便利である。この場合、外部関数 DRAW-POLYLINE は複数個の内部操作 DR-LINE からなる。次の例は仕様言語の表し方を例示する。例では type A と type D の二つのデータ型が宣言されている。

Data Type: A

Let A, B;

内部演算子宣言

IOP1: $A \rightarrow B$

外部演算子宣言

EOP1: $A \times B \rightarrow B$

公理:

(EOP2(A) AND IOP1(A) を公理化する.)

定理

(もしあれば書く)

End DataType: A;

Data Type: D

Let C, D;

内部演算子宣言

$$\text{IOP1: } A \times D \rightarrow B$$

外部演算子宣言

$$\text{EOP2: } A \rightarrow D$$

公理:

(EOP1(A, B) AND IOP1(A, D) を公理化する)

定理

(もしあれば書く)

End Data Type: D;

例で、IOP1 が重複して現れるが、スコープ・ルールによってそれらは異なり、仕様に曖昧さは生じない。またこの例は、相互参照を含む。すなわち、データ型 A で EOP2(A) を参照し、かつデータ型 D で EOP1(A, B) を参照するデータ型宣言の対を示している。階層的スタイルのデータ型宣言はデータ型仕様を単純化する。データ型宣言の変更や修正が不可欠となったとき、階層的データ型宣言を使えばこれら将来の宣言の変更による波及効果をも低くすることができる。

6. おわりに

グラフィック・データ型の形式的仕様を与えるいくつかの試みがある^[15, 24, 25, 29]。一方では(言語, プロトコルなどの)グラフィック・システムの標準化努力がある。本稿ではグラフィック言語の例を使って形式的仕様の例を示し、これら両方の努力の橋渡しを試みた。まず GKS, CORE, NAPLAPS, IGES^[1, 3, 19, 26] などの多数の現存するグラフィック・システムの基礎となる一般的なグラフィック対象の形式的データ型に関する多くの基本概念を示した。代数的な記法で、線描きデータ型、すなわち、点、線、折れ線および多角形の意味を定める一組の公理の存在を示した。

第2に、作画機構の抽象化の例を示した。世界座標空間でのモデル化からビューポートの中での表示への写像の例を抽象的設定で明らかにした。3次元モデルから2次元表示への変換関数 PIPELINE によって絵を空間的に縮約する方法で多くの数学的空間の関係を明らかにした。また、この縮約変換は色を塗る関数とは独立であった。

データ型の抽象化を装置と独立した環境で扱った。スタックのデータ型^[9]などのデータ型に比べ、グラフィック・データ型は装置に依存するビューの仕様と分離できない。

絵を連続空間から離散空間へ写像するデジタル化が絵の構成での最後の変換となる。装置の分解能に応じてサンプリングを行う装置に依存した変換については本稿ではふれなかった。この表示装置への依存性は、普通グラフィック・データ型の形式的仕様に関する現在の研究でも考えられていない。

仕様はプログラムに束縛されなければならない。グラフィック言語はたいてい複数のホスト言語への拡張となるから、ここでは複数の外部演算子仕様を指定することによって、複数のホスト言語への束縛を支援するデータ型仕様技法を設計した。ただ一つの言語 FORTRAN の束縛を示しただけだが、他の言語への束縛の拡張も追加できる。

仕様全体と代数的に与えようというもとの意図は、ある局面では非現実的であることがわかった。これは代数に問題があるのではなく記法自体の問題である。代数的記法は入力引数をもつ関数のみを表すが、一般にプログラム言語では関数と手続きを表し、その両方で引数の役割は動的に宣言される。問合せ関数のクラスは明らかに(代数的な)抽象化機構とプログラミングの実体との間の相違点を示している。手続き宣言を束縛するに

は、代数的記法とは異なった、完全にグラフ理論的引数束縛が必要である。本稿で提示した FORTRAN の問合せ関数を束縛する解決法は、 $f: A \rightarrow B$ の形式の従来の代数的記法に新しい記法 $f: A \leftarrow B$ を含めるだけの最小の変更にとどめるように設計した一時的な解決法である。一般の引数束縛については他のアプローチも考えられよう。

グラフィック・システムの多くは様相的である。したがって、属性の現在値をシステムの現在の状態と同様に扱うことができる。様相的属性では、代入文の左側値のように、再設定されるまでその値が保たれる。実際、集合属性関数はシステムの状態推移を引き起こす。この様相的属性の性質は、一つの属性に対してその現在値、あるいは状態を記憶しておく1個の記憶セルが必要であることを示している。この属性値あるいは状態の参照は、Moore 機械の状態[†]の参照で表現できる。それには、Moore 機械を連続の状態へ修正した機械 $M(K, E, D, d, L, Q^*, F)$ が適している。形式的仕様の例では、Moore 機械を使って離散空間および連続空間の両方におけるすべての様相的システムをモデル化している。

付 録 A

現実的には、FORTRAN は依然としてグラフィックの応用で最もよく使われている言語である。最初の GKS 言語の束縛は FORTRAN であり、ほとんどすべての CORE システムは FORTRAN を使うシステムである。そこで、グラフィックの言語の束縛の例として FORTRAN を選ぶ。この束縛では、この現実の要請と引き替えにプログラムのエレガンスが失われている。

SWINDO (LEFT, RIGHT, BOTTOM, TOP),

IWINDO (LEFT, RIGHT, BOTTOM, TOP):

WC 空間内の空間 WINDOW の値のセットと問合せ。

SNDCS 2 (WIDTH, HEIGHT), INDCS 2 (WIDTH, HEIGHT):

2次元の NDC 空間のサイズのセットと問合せ。

SVPRT 2 (XMIN, XMAX, YMIN, YMAX),

IVPRT 2 (XMIN, XMAX, YMIN, YMAX):

NDC 空間の中のビューポートのサイズのセットと問合せ。

POLYL 2 (N, X-Array, Y-Array):

配列 X-Array と Y-Array に与えられている N 個の頂点を通る折れ線を描く。

POLYG 2 (N, X-Array, Y-Array):

配列 X-Array と Y-Array に与えられている N 個の頂点を通る多角形を描く。

SLCOLR (H, L, S), ILCOLR (H, L, S):

線属性の色の現在値のセットと問合せ。

SFCOLR (H, L, S), IFCOLR (H, L, S):

多角形の埋め色の現在値のセットと問合せ。

SPESTY (K), IPESTY (K):

多角形の辺のスタイルの現在値のセットと問合せ。

SLSTYL (K), ILSTYL (K):

これはパラメタの移送を行う。線のスタイルの現在値のセットと問合せ。

† (訳注) 原文では memory である。

付 録 B

Data Type: Point

Let CW-Space := World-Coordinate-Space $\in \{R^{**2}, C, R^{**3}\}$,
 Point \in WC-Space,
 Window \subset WC-Space,
 Color-Space := $\{(H, L, S)\} \in R^{**3}$,
 Picture-Model := WC-Space \times Color-Space,
 W-ARG-Space := $\{(LEFT, RIGHT, BOTTOM, TOP)\} \in R^{**4}$;

内部演算子宣言

INQUIRE-WINDOW : {Window} \rightarrow WC-Space \times WC-Space;

外部演算子宣言

LINE-OP : {Real} \times WC-Space \times WC-Space \rightarrow WC-Space;
 COLOR : WC-Space \times Picture-Model \rightarrow Color-Space;
 SET-WINDOW : WC-Space \times WC-Space \rightarrow {Window};

ルーチン宣言 (FORTRAN):

SWINDO : W-ARG-Space \rightarrow {Window};
 IWINDO : W-ARG-Space \leftarrow {Window};

ルーチン宣言 (PASCAL):

***** (If any) *****

公 理

Let $a \in \{Real, 0. \leq a \leq 1.\}$, $b \in \{Real\}$,
 PM \in Picture-Model,
 P, P1, P2 \in WC-Space, (H, L, S) \in Color-Space
 (LEFT, RIGHT, BOTTOM, TOP) \in W-ARG-Space;

*****Line Generator*****
 LINE-OP(a, P1, P2) = V-ADD (S-MULT(a, P1), S-MULT((1. - a), P2));

*****Symmetric LINE-OP*****
 LINE-OP(a, P1, P2) = LINE-OP (a, P2, P1);

*****Window Definition*****
 P \in SET-WINDOW (P1, P2) := Window*
 \rightarrow (LINE-OP(a, P, P1) \subset Window*) (LINE-OP(a, P, P2) \subset Window*);
 *****Convex Window*****
 SET-WINDOW (P1, P2) = SET-WINDOW (P2, P1);
 *****Symmetry*****
 INQUIRE-WINDOW(SET-WINDOW (P1, P2)) \in $\{(P1, P2), (P2, P1)\}$;
 SET-WINDOW(INQUIRE-WINDOW(Window)) = Window;

束縛 (FORTRAN) *****Two dimensional*****

Let MIN (P1(1), P2(1)) := P4(1) := LEFT,
 MAX (P1(1), P2(1)) := P3(1) := RIGHT,
 MIN (P1(2), P2(2)) := P4(2) := BOTTOM,
 MAX (P1(2), P2(2)) := P3(2) := TOP;
 *****For three dimensions, one may choose to include*****

MIN (P3(3), P2(3)) := : P4(3) := : FRONT,
 MAX (P1(3), P2(3)) := : P3(3) := : BACK,
 For a two dimensional space, FRONT := : BACK.*****
 ($1 \leq I \leq \text{DIM}(P)$) ($P4(I) \leq P(I) \leq P3(I)$) \rightarrow ($P \in \text{SET-WINDOW}(P1, P2)$);
 SET-WINDOW (P1, P2) = SWINDO (LEFT, RIGHT, BOTTOM, TOP);
 SWINDO (W-ARG) = IWINDO⁻¹(W-ARG);

束縛 (PASCAL) ***** (If any) *****

定 理

LINE-OP (a, P1, P2) = LINE-OP (1. -a, P2, P1);

End Data Type: Point;

Data Type: Picture

Let NDC-Space \in {R**2, C, R**3},
 CNDC-Space \subseteq NDC-Space,
 Viewport \subseteq CNDC-Space,
 Picture := : {Viewport} \times Color-Space,
 NDC-ARG-Space := : {W, H} \in R**2,
 VP-ARG-Space := : {(XMIN, XMAX, YMIN, YMAX)} \in R**4;

内部演算子宣言

SET-NDC-SPACE : NDC-Space \rightarrow {CNDC-Space};
 INQUIRE-NDC-SPACE : {CNDC-Space} \rightarrow NDC-Space;
 SET-VP : CNDC-Space \times CNDC-Space \rightarrow {Viewport};
 INQUIRE-VP : {Viewport} \rightarrow CNDC-Space \times CNDC-Space;
 COLOR' : Viewport \times Picture \rightarrow {Color: (H, L, S)};
 PIPELINE : Picture-Model \rightarrow Picture;
 P-MAP : WC-Space \rightarrow Viewport;

外部演算子宣言

ルーチン宣言 (FORTRAN):

SNDCS 2 : NDC-ARG-Space \rightarrow {CNDC-Space};
 SVPR2 2 : VP-ARG-Space \rightarrow {Viewport};
 INDCS 2 : NDC-ARG-Space \leftarrow {CNDC-Space};
 IVPRT 2 : VP-ARG-Space \leftarrow {Viewport};

公 理

Let PM \in Picture-Model, Pic \in Picture,
 N, N' \in NDC-Space, a \in {Real, 0. \leq a \leq 1.},
 C, C1, C2 \in CNDC-Space,
 V, V1, V2 \in Viewport, VP \in {Viewport},
 W, W1, W2 \in Window, P, P1, P2 \in WC-Space,
 (H, L, S) \in Color-Space;

***** Relationship of Space Dimensions *****

DIM(P) \geq DIM(N) \geq DIM(C) = DIM(V);

***** NDC-Space Definition *****

$(\forall N) (1 \leq I \leq \text{DIM}(N)) \rightarrow (0. \leq N(I) \leq 1.) ;$
 $(1 \leq I \leq \text{DIM}(N)) (0. \leq N(I) \leq 1.) \rightarrow (N \in \text{NDC-Space}) ;$
*******CNDC_Space Definition*******
 $(1 \leq I \leq \text{DIM}(N)) (0. \leq N(I) \leq N'(I)) \rightarrow (N \in \text{SET-NDC-SPACE}(N')) ;$
 $(C \in \text{SET-NDC-SPACE}(N)) \rightarrow (1 \leq I \leq \text{DIM}(C)) (0. \leq C(I) \leq N(I)) ;$
 $\text{INQUIRE-NDC-SPACE}(\text{SET-NDC-SPACE}(N)) = N ;$
*******Viewport Definition*******
 $(\forall V \subseteq \text{SET-VP}(C1, C2))$
 $(\text{LINE-OP}(a, V, C1) \subset \text{SET-VP}(C1, C2))$
 $(\text{LINE-OP}(a, V, C2) \subset \text{SET-VP}(C1, C2)) ;$
*******Convex VP*******
 $\text{SET-VP}(C1, C2) = \text{SET-VP}(C2, C1) ;$ *******Symmetry*******
 $\text{INQUIRE-VP}(\text{SET-VP}(C1, C2)) \in \{(C1, C2), (C2, C1)\} ;$
*******A PIPELINE Definition*******
 $(\exists B \in \text{VP}) \rightarrow (V := :V\text{-ADD}(B, P\text{-MAP}(W)))$
 $((V, \text{COLOR}'(V, \text{Pic})) = \text{PIPELINE}(W, \text{COLOR}(W, \text{PM})))$
*******Relation between VP and Window*******
束縛 (FORTRAN) ***Two dimensional*******
Lct $N := : (W, H) \in \text{NDC-ARG-Space},$
 $\text{MIN}(C1(1), C2(1)) := : C4(1) := : XMIN,$
 $\text{MAX}(C1(1), C2(1)) := : C3(1) := : XMAX,$
 $\text{MIN}(C1(2), C2(2)) := : C4(2) := : YMIN,$
 $\text{MAX}(C1(2), C2(2)) := : C3(2) := : YMAX,$
 $\text{VP-ARG-Space} := : \{(XMIN, XMAX, YMIN, YMAX)\} ;$
 $\text{NDC-Space} \in R^{**2} ;$
 $\text{DIM}(P) = \text{DIM}(N) = \text{DIM}(C) = \text{DIM}(V) = 2 ;$
 $\text{SNDCS2}(W, H) = \text{SET-NDC-SPACE}(N) ;$
 $\text{SVPRT2}(XMIN, XMAX, YMIN, YMAX) = \text{SET-VP}(C1, C2) ;$
 $(1 \leq I \leq \text{DIM}(C)) (C4(I) \leq C(I) \leq C3(I)) \rightarrow (C \in \text{SET-VP}(C1, C2)) ;$
 $\text{SNDCS2}(W, H) = \text{INDCS2}^{-1}(W, H) ;$
 $\text{GSVP2}(XMIN, XMAX, YMIN, YMAX)$
 $= \text{GRVP2}^{-1}(XMIN, XMAX, YMIN, YMAX) ;$
*******Implementation defined P-MAP could be a matrix multiplication**
 $P\text{-MAP}(W) = \text{TRANSLATE}(\text{ROTATE}(\text{SCALE}(\text{CLIP}(W))))$
where $W := : (W(1), W(2), 1.)$ and $(W(1), W(2)) \in \text{WC-Space}*****$

End Data Type: Picture;

Data Type: Polyline

Let $\text{Vertices} \subset \text{WC-Space},$
 $\text{ARG-Space-2} := : \{\text{Integer}\} \times \{\text{Vector}\} \times \{\text{Vector}\},$
 $\text{Line-Segment} \subset \text{WC-Space},$
 $\text{Vector} \in \{\text{Real}^{**\text{DIM}}(\text{Vector})\} ;$

内部演算子宣言

DR-LINE : WC-Space \times WC-Space \times Picture-Model \rightarrow Picture-Model ;
 DR-PL : {Vertices} \times Picture-Model \rightarrow Picture-Model ;

外部演算子宣言

DR-CLOSED-PL : {Vertices} \times Picture-Model \rightarrow Picture-Model ;
 LINE : WC-Space \times WC-Space \rightarrow {Line-Segment} ;

ルーチン宣言 (FORTRAN):

POLYL2 : ARG-Space-2 \rightarrow Picture-Model ;

公理

Let $a \in \{\text{Real}, 0. \leq a \leq 1.\}$, $PM \in \text{Picture-Model}$,
 $P, P_i, P_j, P_k \in \text{WC-Space}$, $\text{Vertices} := \{P(I) \mid 1 \leq I \leq N\}$,
 $\text{ARG-2} := (N, X\text{-Array}, Y\text{-Array}) \in \text{ARG-Space-2}$,
 $LS \in \{\text{Line-Style. Integer}\}$;

*****Line segment composition*****

$(\forall a) (\text{LINE-OP}(a, P_i, P_j) \in \text{LINE}(P_i, P_j))$;
 $(P \in \text{LINE}(P_i, P_j)) \rightarrow (\exists a) (P := \text{LINE-OP}(a, P_i, P_j))$;

*****Symmetric line drawing*****

$(\forall PM) (\text{DR-LINE}(P_j, P_k, PM) = \text{DR-LINE}(P_k, P_j, PM))$;

*****Line drawing*****

$(\forall PM) (\forall a) (P := \text{LINE-OP}(a, P_j, P_k)) (\text{VISIBLE}(LS, P_j, P_k, a)$
 $\rightarrow (\text{COLOR}(P, \text{DR-LINE}(P_j, P_k, PM)) = \text{C.COLOR}(\text{Line}))$;
 $(\forall PM) ((P \in \text{COMPLEMENT}(\text{LINE}(P_j, P_k))) \text{ OR}$
 $((P := \text{LINE-OP}(a, P_j, P_k) \in \text{Line}(P_j, P_k)) \text{ AND}$
 $(\text{NOT}(\text{VISIBLE}(LS, P_j, P_k, a))))$
 $\rightarrow (\text{COLOR}(P, PM) = \text{COLOR}(P, \text{DR-LINE}(P_j, P_k, PM)))$;

*****Polyline drawing*****

$(\forall P_i, P_{i+1} \in \text{Vertices})$
 $(\text{DR-PL}(\text{Vertices}, PM) = \text{DR-LINE}(P_i, P_{i+1}, \text{DR-PL}(\text{Vertices}, PM)))$;
 $\text{DR-CLOSED-PL}(\text{Vertices}, PM) = \text{DR-PL}(\text{Vertices}, \text{DR-LINE}(P_1, P_n, PM))$;

*****Programming Language Binding (2D)*****

$2 \leq \text{DIM}(X\text{-Array}) = \text{DIM}(Y\text{-Array}) = N$;
 $(\forall PM) (\text{POLYL2}(\text{ARG-2}) = \text{DR-PL}(\text{Vertices}, PM))$;
 $(\forall P(I) \in \text{Vertices}) (P(I) := (X\text{-Array}(I), Y\text{-Array}(I)))$;
 $(\forall P \in \text{Vertices}) (\text{DIM}(P)=2)$;

定理

$\text{DR-CLOSE-PL}(\text{Vertices}, PM) = \text{DR-LINE}(P_1, P_n, \text{DR-PL}(\text{Vertices}, PM))$;

End Data Type: Polyline;

Data Type: Polygon

Let $\text{Vertices} \subset \text{WC-Space}$, $PG \subset \text{WC-Space}$,
 $\text{ARG-Space-2} := \{\text{Integer}\} \times \{\text{Vector}\} \times \{\text{Vector}\}$,
 *****More Machine=Mls(K, E, D, d, L, Q*, F)*****

```

PE-State-Space := :{Q(ON), Q(OFF)}  ****K=F*****
PE-Output-Space := : Logical-Space := :{TRUE, FALSE},
                                     ****D*****
Q* := : Q(OFF),                      ****Q*****
PL := : {LINE(Pi, Pj)} ;

```

内部演算子宣言

```

CONNECT : WC-Space × WC-Space → {PL} ;
BOUNDARY : {Vertices} → {PL} ;
POLYGON : {Vertices} → {PG} ;
DR-PG : {Vertices} × Picture-Model → Picture-Model ;
DR-BOUNDARY : {Vertices} × Picture-Model → Picture-Model ;
****d****
SET-PG-EDGE : PE-Input-Space × PE-State-Space → PE-State-Space ;
****L****
INQUIRE-PG-EDGE : PE-State-Space → PE-Output-Space ;

```

外部演算子宣言

ルーチン宣言 (FORTRAN):

```

POLYG 2 : ARG-Space-2 → Picture-Model ;
SPESTY : PE-Input-Space → PE-State-Space ;
IPESTY : PE-Output-Space ← PE-State-Space ;

```

公理

```

Let a ∈ {Real, 0. ≤ a ≤ 1.}, PM ∈ Picture-Model,
P, Pi, Pj, Pm, Pn ∈ WC-Space, Vertices := :{P(I) | 1 ≤ I ≤ N},
ARG-2 := : (N, X-Array, Y-Array) ∈ ARG-Space-2,
PE ∈ PE-Input-Space ;

****CONNECT two points via a polyline****
(∃ Vertice = {P(I) | M ≤ I ≤ N} ∈ WC-Space)
(CONNECT(P(M), P(N)) = {LINE(P(I), P(I+1)) | M ≤ I, I+1 ≤ N}) ;

****BOUNDARY Function****
BOUNDARY (Vertices) = {CONNECT(P1, Pn), LINE(Pn, P1)} ;

****Draw Polygon****
(∇ PM) (∇ P ∈ POLYGON(Vertices))
(COLOR(P, DR-PG(Vertices, PM)) = C-COLOR(Fill)) ;
(∇ PM) (P ∈ COMPLEMENT(POLYGON(Vertices)))
→ (COLOR(P, DR-PG(Vertices, PM)) = COLOR(P, PM)) ;

****Draw Boundary****
(∇ PM) (DR-BOUNDARY(Vertices, PM) = DR-CLOSED-PL(Vertices, PM)) ;

****Polygon Edge Style****
SET-PG-EDGE(PE, Q(PE')) = Q(PE) ;
INQUIRE-PG-EDGE(SET-PG-EDGE(PE, Q(PE'))) ∈ Logic-Space ;
INQUIRE-PG-EDGE(Q(ON)) = TRUE ;
INQUIRE-PG-EDGE(Q(OFF)) = FALSE ;

```

(INQUIRE-PG-EDGE(Q(PE))) (DR-PG(Vertices, PM))
 → (DR-PG(Vertices, DR-BOUNDARY(Vertices, PM))
 = DR-BOUNDARY(Vertices, DR-PG(Vertices, PM)));

*****Polygon Connectivity*****

(Qk, Ql ∈ POLYGON(Vertices))
 → (∃ CONNECT(Qk, Ql) ⊆ POLYGON(Vertices));

*****Simply connected Polygon*****

(Qm, Qn ∈ COMPLEMENT(POLYGON(Vertices)))
 → (∃ CONNECT(Qm, Qn) ⊆ COMPLEMENT(POLYGON(Vertices)));

*****Polygon Interior definition*****

(∀ Q ∈ BOUNDARY(Vertices)) (∀ Qk ∈ POLYGON(Vertices))
 (INTERSECT (∃ CONNECT(Q, Qk),
 COMPLEMENT(POLYGON(Vertices))) = {Q});

*****Coplanar Polygon Surfaces*****

(b, c, d ∈ {Real}) (P1, P2, P3, P4 ∈ POLYGON(Vertices))
 Q2 := : V-ADD(P2, S-MULT(-1., P1))
 Q3 := : V-ADD(P3, S-MULT(-1., P1))
 Q4 := : V-ADD(P4, S-MULT(-1., P1))
 → (Q2=S-MULT(b, Q3))
 (Q4=V-ADD(S-MULT(c, Q2), S-MULT(d, Q3)))

束縛 (FORTRAN):

2 ≤ DIM(X-Array) = DIM (Y-Array) = N ;
 (∀ PM) (POLYG 2(ARG-2) = DR-PG(Vertices, PM)) ;
 (∀ P(I) ∈ Vertices) (P(I) := : (X-Array(I), Y-Array(I))) ;
 (∀ P ∈ Vertices) (DIM(P) = 2) ;
 (∀ PE') (SPESTY(PE) = SET-PG-EDGE(PE, Q(PE')) ;
 SPESTY(PE) = IPESTY⁻¹(PE) ;

End Data Type: Polygon;

Data Type: C Color

*****Moore Machine := : Mcc (K, E, D, d, L, Q*, F)*****

Let T ∈ Type := : {Line, Fill},

State-Space = {Q := : Q(T, Color)}, *****K=F*****

Input-Space := : Type × Color-Space, *****E*****

Output-Space := : Color-Space := : {Color := : (H, L, S)} ∈ R**3,

*****D*****

Q* := : {Line, Color*}, (Fill, Color**) ⊂ State-Space ;

*****Q*****

内部演算子宣言

SET-COLOR : Input-Space × State-Space → State-Space ;

*****d*****

INQUIRE-COLOR : Type × State-Space → Output-Space ;

*****L*****

外部演算子宣言

C-COLOR: Type \rightarrow Color-Space;

ルーチン宣言 (FORTRAN):

SLCOLR: Color-Space \rightarrow State-Space;SFCOLR: Color-Space \rightarrow State-Space;ILCOLR: Color-Space \leftarrow State-Space;IFCOLR: Color-Space \leftarrow State-Space;

公理

Let Color := (H, L, S), Color' := (H', L', S'), \in Color-Space;

*****Hue, Lightness, Saturation*****

 $(\forall (H, L, S)) (0. \leq H \leq 360.) (0. \leq L \leq 1.) (0. \leq S \leq 1.);$

*****Modal attributes presented as a Moore machine*****

 $(\forall T) (\forall Q) (\text{INQUIRE_COLOR}(T, Q(T, \text{Color})) = \text{Color});$ $(\forall T) (\forall Q) (\text{INQUIRE_COLOR}(T, \text{SET_COLOR}((T, \text{Color}), Q(T, \text{Color}))) = \text{Color};$ $(\forall Q) (\text{C_COLOR}(T) = \text{INQUIRE_COLOR}(T, Q));$

束縛 (FORTRAN):

Let Color* := (360., 1., 1.);

Color** := (360., 1., 1.);

 $(\forall Q) (\text{SLCOLR}(H, L, S) = \text{SET_COLOR}((\text{Line}, (H, L, S)), Q))$ $(\text{SFCOLR}(H, L, S) = \text{SET_COLOR}((\text{Fill}, (H, L, S)), Q));$ $\text{SLCOLR}(H, L, S) = \text{ILCOLR}^{-1}(H, L, S);$ $\text{SFCOLR}(H, L, S) = \text{IFCOLR}^{-1}(H, L, S);$

End Data Type: C Color;

Data Type: Linestyle

*****Moore Machine = Mls (K, E, D, d, L, Q*, F)*****

Let $a \in \{\text{Real}, 0. \leq a \leq 1.\},$ LS \in Type := {SOLID, DOT, DASH, DOT.DASH*****etc.*****},

State-Space := {Q := Q(LS)}, *****K=F*****

Input-Space := Type, *****E*****

Output-Space := Type, *****D*****

Q* := Q(SOLID); *****Q*****

内部演算子宣言

*****d*****

SET-LINESTYLE: Input-Space \times State-Space \rightarrow State-Space;

*****L*****

INQUIRE-LINESTYLE: State-Space \rightarrow Output-Space;

外部演算子宣言

VISIBLE: Type \times WC-Space \times WC-Space \times {a} \rightarrow {Logical};

ルーチン宣言 (FORTRAN):

SLSTYL : {Integer} → State-Space ;

ILSTYL : {Integer} ← State-Space ;

公 理

Let $I \in \{\text{Integer}\}$, $P_i, P_j \in \text{WC-Space}$;

*****General Properties*****

SET-LINESTYLE(LS, Q(LS'))=Q(LS) ;

INQUIRE-LINESTYLE(Q(LS))=LS ;

INQUIRE-LINESTYLE (SET-LINESTYLE(LS, Q(LS')))=LS ;

VISIBLE(LS, P_i, P_j, a) = VISIBLE (LS, $P_i, P_j, (1. - a)$) ;

*****Device independent linestyle*****

($\forall a$) VISIBLE(SOLID, P_i, P_j, a) = True ;

($a \in \{b*.001\}$) ($b \in \{\text{Real}, 0. \leq b \leq 1000.\}$)

→ (VISIBLE(DOT, P_i, P_j, a)=True) ;

($a \notin \{b*.001\}$) ($b \in \{\text{Real}, 0. \leq b \leq 1000.\}$)

→ (VISIBLE(DOT, P_i, P_j, a)=False) ;

束縛 (FORTRAN) :

($\forall Q$) SLSTYL(1)=SET-LINESTYLE(SOLID, Q) ;

($\forall Q$) SLSTYL(2)=SET-LINESTYLE(DOT, Q) ;

($\forall Q$) SLSTYL(3)=SET-LINESTYLE(DASH, Q) ;

($\forall Q$) SLSTYL(4)=SET-LINESTYLE(DOT.DASH, Q) ;

SLSTYL(Integer)=ILSTYL⁻¹ (Integer) ;

Theorem

VISIBLE(LS, P_i, P_j, a) = VISIBLE(LS, P_j, P_i, a) ;

End Data Type : Linestyle ;

(技術研究部 山田真市 訳)

- 参考文献 [1] "Status Report of the Graphics Standards Planning Committee", *ACM SIGGRAPH Quarterly Report*, Vol. 13, No. 3, 1979.
- [2] G.S. Carson, "The Specification of Computer Graphics Systems", *IEEE Computer Graphics and Applications*, Vol. 3, No. 6, 1983.
- [3] CBEMA, "Videotex/Teletext Presentation Level Protocol System (North American PLPS)", *Draft Standard ANSI X3L2*, 1983.
- [4] A.Church, *Introduction to Mathematical Logic*, Vol. 1, Princeton University Press, 1956.
- [5] J.D. Foley, A. van Dam, *Fundamentals of Interactive Computer Graphics*, Addison Wesley Pub. Co., 1982.
- [6] A. Goguen, J.W. Thatcher, E.G. Wagner, "An Initial Algebraic Approach to the Specification, Correctness, and Implementation of Abstract Data Types", *Current Trends in Programming Methodology* Vol. IV, *Data Structuring* (ed. R. T. Yeh.), Prentice Hall Pub. Co., 1978.
- [7] J.V. Guttag, "The Specification and Application to Programming of Abstract Data Types", Ph.D. Dissertation, University of Toronto, 1975, (および *Tech. Rep. CSRG-59*, Computer Systems Research Group, University of Toronto, 1975)
- [8] J.V. Guttag, "Abstract Data Types and the Development of Data Structures", *Proc. of Conference on Data: Abstraction, Definition and Structure*, 1976. (および *ACM SIGPLAN Notices*, Vol. 11 No. 11, 1976.), (および *ACM FDT: Bull. SIGMOD*, Vol. 8 No. 2, 1976.)
- [9] J.V. Guttag, "Abstract Data Types and the Development of Data Structures", *CACM*,

Vol. 20 No. 6, 1977.

- [10] J. V. Guttag, E. Horowitz, D. R. Musser, "Some Extensions to Algebraic Specifications", *Proc. of ACM Conference on Language Design for Reliable Software*, 1977. (および *ACM SIGPLAN Notices*, Vol. 12 No. 3, 1977.), (および *Operating System Review*, Vol. 11 No. 2, 1977.), (および *Software Engineering Notes*, Vol. 2 No. 2, 1977.)
- [11] J. V. Guttag, J. J. Horning, "The Algebraic Specification of Abstract Data Types", *Acta Informatica*, Vol. 10 No. 1, 1978.
- [12] J. V. Guttag, E. Horowitz, D. R. Musser, "The Design of of Data Type Specifications", *Current Trends in Programming Methodology Vol. IV, Data Structuring* (ed. R. T. Yeh), Prentice Hall Pub. Co., 1978.
- [13] J. V. Guttag, "Notes on Type Abstractions", *Proc. Specifications of Reliable Software Conference*, IEEE Cat. No. 79, CH 1401-9 C, 1979.
- [14] J. V. Guttag, J. J. Horning, "Preliminary Report on the Larch Shared Language", *Xerox PARC Technical Report CSL-83-6*, 1983.
- [15] J. V. Guttag, J. J. Horning, "Formal Specification as a Design Tool", *7th. Annual ACM Symposium on Principles of Programming Languages*, 1980.
- [16] C. A. R. Hoare, "An Axiomatic Basis for Computer Programming", *CACM*, Vol. 12 No. 10, 1969.
- [17] E. Horowitz, S. Sahni, *Fundamentals of Data Structures*, Computer Science Press, 1976.
- [18] J. E. Hopcroft, J. D. Ullman, *Introduction to Automata Theory, Language and Computation*, Addison-Wesley Pub. Co., 1979.
- [19] ISO, "Graphics Kernel System (GKS)", *Draft International Standard ISO/DIS 7942*, 1983.
- [20] B. H. Liskov, S. N. Zilles, "Programming with Abstract Data Types", *ACM SIGPLAN Notices*, Vol. 9 No. 4, 1974.
- [21] B. H. Liskov, S. N. Zilles, "Specification Techniques for Data Abstractions", *IEEE Trans. On Software Engineering*, SE-1: 1, 1975.
- [22] B. H. Liskov, S. N. Zilles, "An Introduction to Formal Specifications of Data Abstractions", *Current Trends in Programming Methodology*, Vol. 1, (ed. R. T. Yeh), Prentice Hall, 1977.
- [23] M. E. Majster, "Limits of the Algebraic Specification of Data Types", *ACM SIGPLAN Notices*, Vol. 12 No. 10, 1977. (および *ACM SIGPLAN Notices*, Vol 13 No. 1, 1978.)
- [24] W. R. Mallgren, "Formal Specification of Interactive Graphics Programming Languages", *Tech. Rep. 81-09-01*, (および Ph. D. Dissertation), University of Washington, 1981.
- [25] W. R. Mallgren, "Formal Specification of Graphics Data Types", *ACM Trans. on Programming Languages and Systems*, Vol. 4 No. 4, 1982.
- [26] National Bureau of Standards, *Initial Graphics Exchange Specification (IGES)*, Version 2.0, 1983. (および *NTIS PB83-137448*, 1983.)
- [27] R. F. Sproull, W. M. Newman, *Principles of Interactive Computer Graphics*, 2nd. ed., McGraw-Hill, 1979.
- [28] J. W. Thatcher, E. G. Wagner, J. B. Wright, "Data Type Specification: Parameterization and the Power of Specification Techniques", *Conference Proc. of the 10th. Annual ACM Symposium on Theory of Computing*, 1978.
- [29] D. Thalmann, N. Thalmann, "Design and Implementation of Abstract Graphical Data Types", *Proc. of the 3rd. IEEE Computer Software and Applications Conference*, 1979.
- [30] C. J. van Wyk, "A High Level Language for Specifying Pictures", *ACM Transaction on Graphics*, Vol. 1 No. 2, 1982.
- [31] R. T. Yeh (ed.), *Current Trends in Programming Methodology Vol. IV, Data Structuring*, Prentice Hall Pub. Co., 1978.

執筆者紹介 Roger Lynn

1964年に Brown 大学より M.S.を、1968年に New York 大学 Courant Institute より Ph. D. を取得。1968年から1971年は Indiana 大学数学科講師および助教授、1971年から Villanova 大学数理科学助教授および准教授。研究分野は応用数学およびコンピュータ・グラフィックス。



Norman L. Soong

1964年に Florida 大学から M.S., 1975年に Pennsylvania 大学から M.S., 1968年に Florida 大学から Ph.D. を取得。1968年から1969年に Wright-Patterson 空軍基地の空軍研究所に勤務、1969年から1973年に NCR のソフトウェア・システム・アナリストを勤めたのち、現在 Sperry 社の Product division の Research Department の Staff Scientist であり、CAD システムでの人工知能を研究。



報告 狭間隙状態における静電気放電の電磁妨害特性 ——ARP が EMI を決定する——

EMI Characteristics of ESD in a Small Air Gap ——ARP Governs the EMI——

本田 昌實, 川村 雄克

要約 静電気放電 (Electrostatic Discharge: 以下 ESD と略す) 事象に伴う電磁妨害 (Electromagnetic Interference: 以下 EMI と略す) は放電電圧に比例するとは限らず, ARP (Amplitude Rate of charge of current Product) によって決定される。本稿では新しく開発した ESD 検知器と既存の測定装置を用いて, この現象をさまざまな火花間隙状態で実験し解明する。

Abstract The EMI (Electromagnetic Interference) with an ESD (Electrostatic Discharge) event is not always proportional to the discharge voltage, but is governed by the ARP (Amplitude-Rate of change of current Product). The present paper clarifies this phenomenon experimentally at various spark gaps with use of a newly developed ESD detector and existing measuring instruments.

1. はじめに

最近, 電子装置に対する ESD の影響が, あらゆる規模のシステムに顕著に見られる。非常に簡単で小さなシステムですら, 外部の機器 (インタフェースとかセンサといったもの) から信号のやりとりをしている。これらのシステムは, 当然のことながら, 一つの LSI が単独で存在している場合に比べ, EMI にさらされる作用断面積が圧倒的に大きく, その結果たやすく影響を受けることになる。ESD 事象によって周囲へ放射される電磁エネルギーは, 二つの経路すなわち, 導体の表面と空間を通過して, 衝撃的な EMI となって伝搬する。空間を伝搬する EMI の影響は, 超広帯域のスペクトラムを含み電子装置に対する放電電流の直接的な経路がなくても, 電子装置が誤動作するという事実によってもわかる。

偶然の一回限りの ESD 事象によってコンピュータが誤動作するのは何故かを明らかにしなければならない。とくに, 放射された EMI をとらえるため, われわれは最近になって ESD 検出器を開発した。

この検出器を使って, ESD によって生じる EMI の影響を調査した。人が何ら電氣的ショックを感じないような非常に低い帯電電圧 (約 3 kV 以下) での ESD によってもコンピュータが誤動作するということがわかった。

この現象をさらに検証するため, さまざまな火花間隙状態での ESD による EMI について研究した。さらに, いくつかの既存の測定装置を使ってこのことを確認した結果, ESD に伴う EMI は必ずしも放電電圧に比例しないということがわかった。

2. ESD 検出器

2.1 ESD 検出器の概要

コンピュータが誤動作した際, コンピュータ近辺に生ずる ESD 事象をチェックしたり

This article is reprinted, with permission, from proceedings of 6th annual EOS/ESD symposium, Oct. 2-4, 1984.

ESD 強度を調べる場合、多くの問題がある。それはほとんどの場合、既存の測定装置がまったく役に立たないことである。その理由の一つは、ESD 事象が起こるとコンピュータの近くは強い電磁場に支配され、当然のことながら測定システムはそれに干渉される。とりわけ、AC 電源で作動し、測定対象物との間で直接プローブあるいはそれに似た物によって信号のやりとりを行う測定システムでは、その干渉が不確実な結果を引き起こす恐れがある。

以上の経験から、ESD 検出器は電池で動作するものとし、伝導性の結合やアース・ループにより生ずる問題をなくすようにし、その結果 ESD から放射される EMI をアンテナのみから検知できるように工夫した。空中に放射されたマイクロ波帯のスペクトルを含む電磁場のうち、ESD の発生源の近くを支配している誘導電界は、2.5 cm (約1インチ) の長さの1本のアンテナにより受信される。たとえ ESD 事象が一時的であり、PS (ピコ秒、 $10^{-10} \sim 10^{-12}$ 秒) オーダしか持続しないものであっても、この検出器は EMI の強さに応じてそれがリセットされるまで、HIGH または LOW の検知レベルで ESD 事象を表示し、保持する機能をもつ。HIGH の検知レベルは、一般のコンピュータにおけるパルス性 EMI のしきい値に相当する 134dB にセットされている。LOW の検知レベルは ESD 事象またはそのアクティビティを検知するのに適した感度である 102dB にセットされている。ESD 検出器のブロック図を図1に示す。

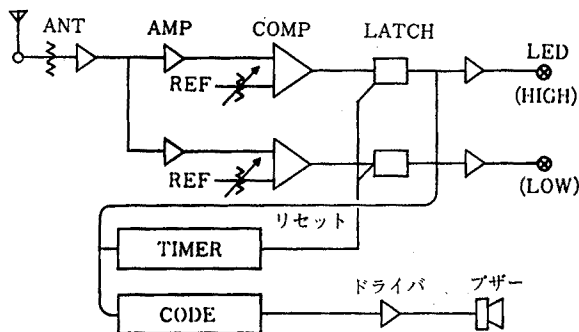


図1 ESD 検出器のブロック図
Fig. 1 Block diagram of ESD detector

2.2 検出器の寸法と重量

寸法：9.6cm(高さ)×6.1cm(幅)×2.3cm(奥行き)

重量：約 90g

図2にコンピュータ内部に設置した ESD 検出器を示す。

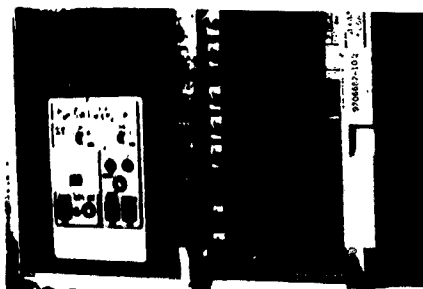


図2 コンピュータ内部に設置した ESD 検出器
Fig. 2 ESD detector installed in a computer

2.3 ESD 検出器を使ってとらえた EMI の一般的な特徴

ESD シミュレータを使って ESD を起こし、それによって周囲の空間に放射された EMI 強度を ESD 検出器で調べる。

この実験においては、ESD シミュレータの放電プローブを手で握り、プローブの先にある放電電極を、アース線の先につけたワニ口クリップと接触させ、その結果生じたスパークによって放射される EMI の到達範囲 (LOW レベル検出) を測定する。その結果、約 6 kV の放電電圧までは、EMI の到達範囲が放電電圧に比例して大きくなるが、放電電圧が約 10 kV を超えると EMI の到達範囲は逆に減少してゆく。約 15 kV における EMI の到達範囲は、2~3 kV の放電電圧におけるそれと同じであることがわかった。

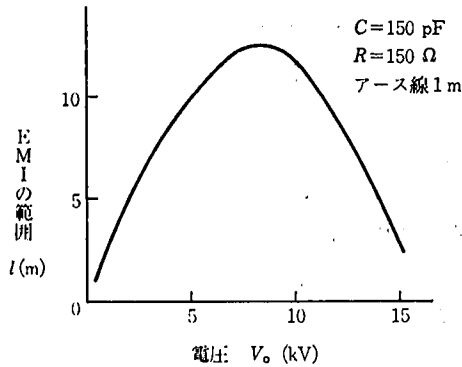


図 3 EMI の範囲と放電電圧

Fig. 3 EMI range vs. discharge voltage

3. 実験の概要

さまざまな放電間隙状態における ESD を次の 2 点から分析する。

- 1) 電流ライズタイム/周波数スペクトル
- 2) 放射された EMI の強度

この分析は、とくに、より小さな放電間隙で詳細に行う必要がある。

第 1 に ESD 発生源のモデルを作り、そこでの放電電流および放電間隙の近くにおける電界と放電間隙から 2 m の距離における電界を、時間領域と周波数領域において測定する。その間隙長 (または放電電圧) をパラメタとして使用する。

図 4 に、ESD 発生源として最も単純化した回路を示す。この回路は放電電流の全経路に沿って測定中に誤りを生じさせる恐れのあるインダクタンスを最小にするように注意して作っている。インダクタンスの近似値は 20 nH (ナノ・ヘンリー) である。スパーク電極は直径 1/2 インチの真ちゅうの球であり、放電間隙長 (d) は、マイクロメータで微調整できる構造をもつ。

ここで、 d は $0 \sim 5000 \mu\text{m}$

最小調整可能距離は $10 \mu\text{m}$

また、実験に使用した主要測定器は以下のとおりである。

- 1) オシロスコープ

タイプ 7104 メイン・フレーム DC~1GHz

7A29 垂直電圧増幅器 ライズタイム 0.4 ns

7B15 タイム・ベース

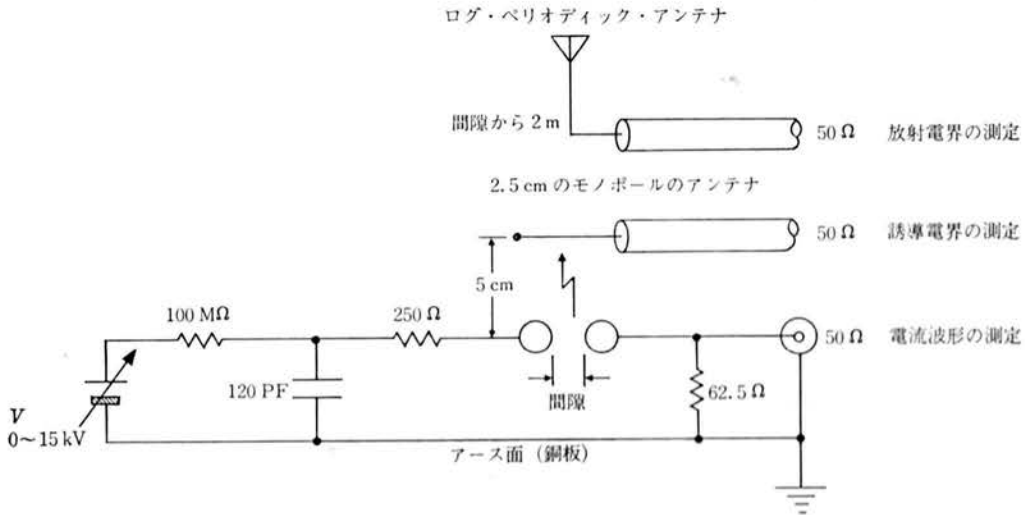


図 4 ESD 実験回路
Fig. 4 ESD experiment circuit

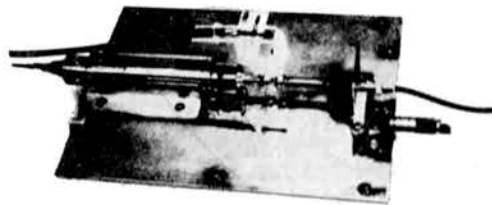


図 5 ESD 発生源
Fig. 5 ESD source

- 2) スペクトラムアナライザ
タイプ 7704A メイン・フレーム
7L13 スペクトラム f_{max} 1.8GHz
- 3) 対数周期アンテナ
タイプ MP 636 A 300 MHz~1.7 GHz

4. 測定結果

4.1 放電電圧と放電間隙長との比較

図 6 は、放電電圧と放電間隙長との対応を先の実験回路を使って測定した結果を示している。放電間隙長における放電寸前の電界の計算値（電圧グラジェント）も示している。間隙が狭くなればなるほど、電界は大きくなるのがわかる。そして当然のことながらこれらは Paschen の法則に従っている。

4.2 電流ライズタイム

放電電流の上昇時間は、明らかに間隙の増加とともに増加する傾向がある。放電電圧と間隙長との対応は、すでに示したとおりである。図 7 に電流の上昇時間 (t_r) と放電電圧 (V_0) との関係について測定した結果を示す。

この測定で使用したオシロスコープには、ライズタイムが 0.4 ns の電圧増幅器が内蔵されている。したがって、測定値が 0.4 ns に近い場合、実際の値はずっと小さいと考え

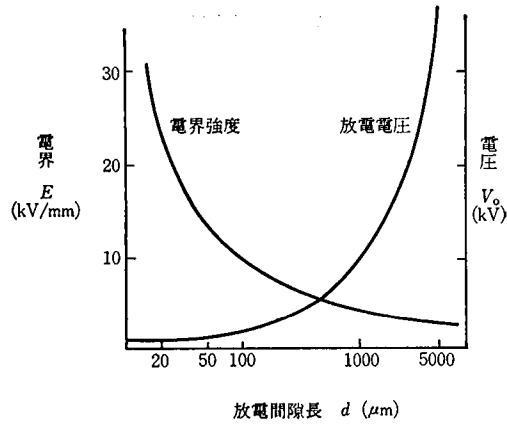


図 6 放電電圧に対応する間隙長と電界
Fig. 6 Discharge voltage vs. spark gap with field in the spark gap

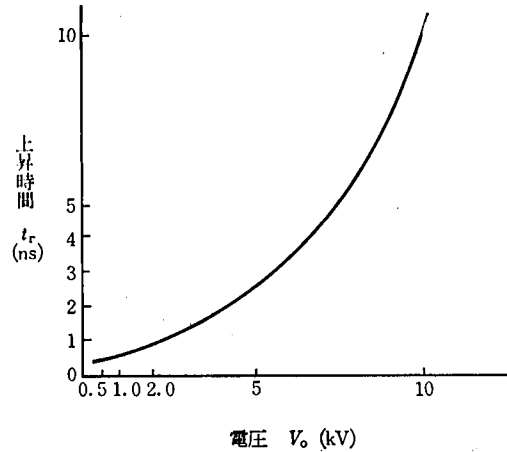
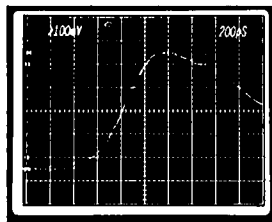
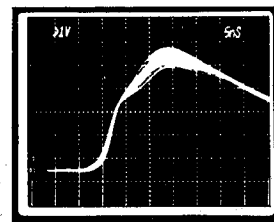


図 7 電流上昇時間と放電電圧の関係
Fig. 7 Current rise time vs. discharge voltage



(a) 放電電圧 0.58 kV での波形



(b) 放電電圧 10 kV での波形

図 8 電流上昇時間の測定した波形
Fig. 8 Measured waveforms of current rise time

られる。

図 8 は、測定した波形を示している。

4.3 ピーク電流と放電電圧の比較

間隙（放電電圧）が変化するにつれ、放電電流は放電電圧に対応して変化するが、直接の比例関係はない。低い放電電圧においては、ピーク電流は V/R 比率（放電電圧/放電

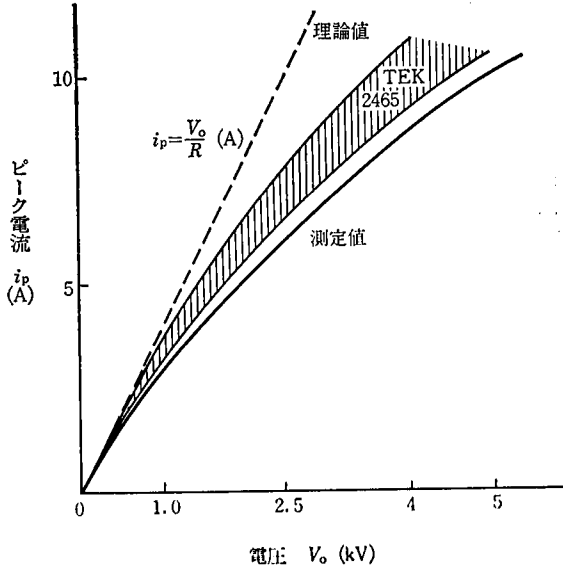
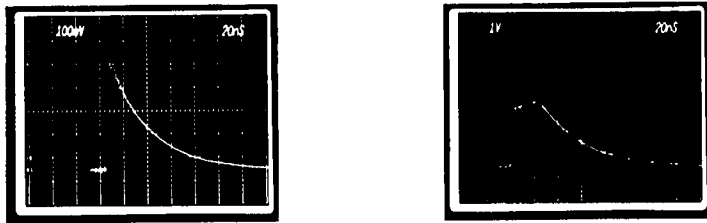


図 9 ピーク電流と放電電圧の比較
Fig. 9 Peak current vs. discharge voltage



(a) 0.58 kV の放電電圧時 (b) 5 kV の放電電圧時

図 10 異なった放電電圧による電流波形の相異
Fig. 10 Different current waveforms with different discharge voltages

回路抵抗) と良く一致するが、放電電圧が増加するにつれ、ピーク電流は比例関係から離れる。たとえば、5 kV ではわずかに V/R 比率の 54 パーセントとなる。

放電電流の測定には一つの問題がある。すなわち、放電が同じ放電電圧で繰り返されるとピーク電流は一定の範囲内で変化する。その範囲は 15~20 パーセントである。図 9 は、TEK 2465 ($f_{max}=300$ MHz) を使って得られたデータである。

図 10 は異なった放電電圧における電流波形の相異を示す。

4.4 放電間隙付近におけるアンテナ誘起電圧

放電間隙から 5 cm の空間にモノポール・アンテナを固定し、そのアンテナに誘起される電圧をオシロスコープで測定する。図 11 はその測定結果である。図 12 は測定した波形のうちの一つを示す。間隙が広くなると、アンテナに誘起される電圧は減少することがわかる。また間隙が大きくなると、アンテナ誘起電圧の波形はゆるやかになることを示している。

4.5 間隙から 2 m の距離におけるアンテナ誘起電圧

間隙から 2 m のところに、対数周期アンテナ (300 MHz~1.7 GHz) を設置し、そのアンテナに誘起された電圧を 1.8 GHz の帯域をもつスペクトラムアナライザで測定する。

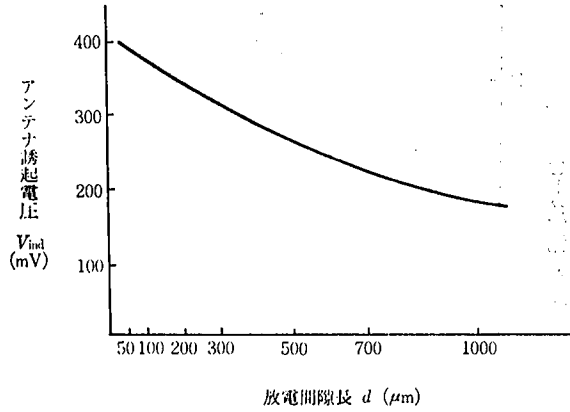
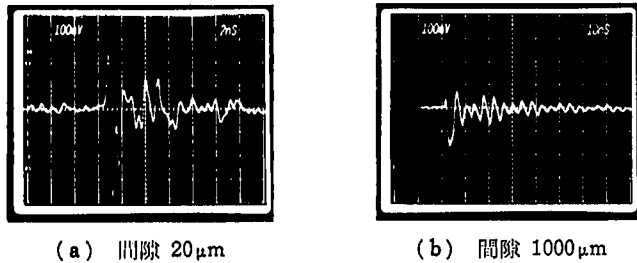


図 11 放電間隙付近のアンテナ誘起電圧
 Fig. 11 Antenna induced voltage in the vicinity of the spark gap



(a) 間隙 $20\mu\text{m}$ (b) 間隙 $1000\mu\text{m}$

図 12 アンテナ誘起電圧の波形と振幅の相異
 Fig. 12 Antenna induced voltage, difference in the waveform and amplitude

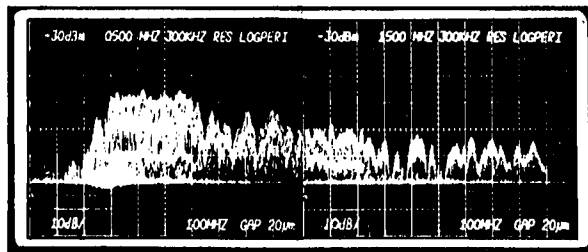


図 13 間隙が $20\mu\text{m}$ のときの周波数スペクトラム
 Fig. 13 Received power frequency spectrum at spark gap $20\mu\text{m}$

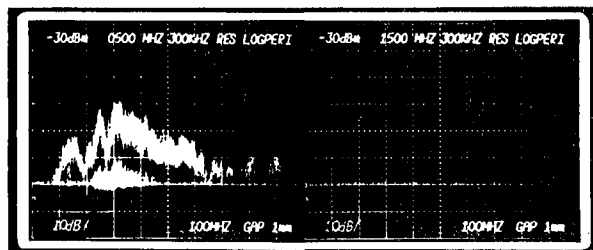
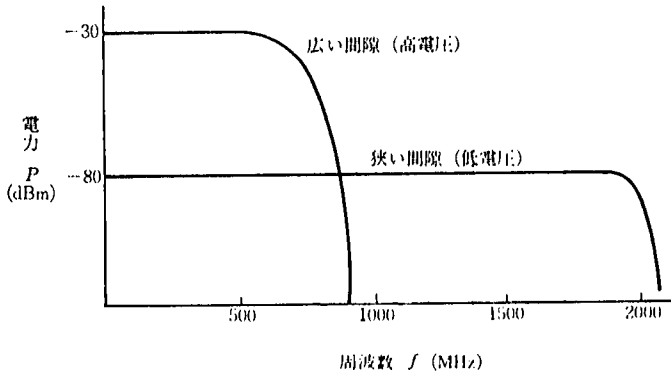


図 14 間隙が $1000\mu\text{m}$ のときの周波数スペクトラム
 Fig. 14 Received power frequency spectrum at spark gap $1000\mu\text{m}$

間隙が広くなり放電電圧が増加すると、周波数帯域は明らかに狭くなる。図 13 と図 14 に間隙が $20\mu\text{m}$ と $1000\mu\text{m}$ のとき得られた結果を示す。使用した測定器の限界を考慮すると、実際の帯域は間隙が $20\mu\text{m}$ のとき、 2GHz を超えることは確かであろう。

図 13, 図 14 とも、 300MHz より下の低周波領域においては電力レベルは落ちている。その理由は、受信アンテナの有効周波数帯域に限界があるためであり、実際のレベルはもっと高い。とくに、広い間隙においては、電力が低周波領域に集中することは明らかである。この傾向を概念的に図 15 に示す。



スペクトラム・アナライザのしきい値 1.8GHz (TEK 71.13)

図 15 放射された周波数スペクトラムと間隙の関係
Fig. 15 Radiated frequency spectrum vs. spark gap

5. 実験結果の要約

- 1) 放電間隙における“電界”(電圧グラジェント)は間隙の減少に伴い急速に増加する
- 2) 放電電流の“上昇時間”は間隙が広くなるにつれ増加する
- 3) ピーク放電電流は、間隙が広くなるにつれ増加するが、その増加率は減少する
- 4) 放電間隙の近くに置いたアンテナに誘起される電圧は、間隙が大きくなるにつれ減少する
- 5) 放電間隙から一定の距離をおいて観察した周波数スペクトルは、間隙が広くなるにつれ、低周波数領域に集中する

6. 検討事項

6.1 放電電流の上昇時間

測定の結果から放電電流の上昇時間は放電電圧の増加に伴い、増加することが確かめられた。

この上昇時間は、ESD による EMI の特徴として電子装置にとって非常に重要である。したがって、なぜ放電電圧の増加に伴って上昇時間が増加するかを検討する。

図 16 において、 C にかかる電圧がエア・ギャップの長さによって決まる空気の絶縁破壊電圧に到達すると、このギャップ内の空気はただちに導電性をおびる。

さて、一定のギャップで生ずるこの絶縁破壊という瞬間的に起こる事象を、空気の導電率 (ρ) の変化という点から検討しよう。

まず、放電電流の上昇時間が導電率の増加率によって決まることを考えてみよう。瞬間

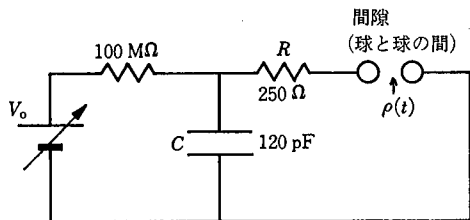


図 16 エア・ギャップ間の CR 放電回路

Fig. 16 CR circuit for discharge through an air gap

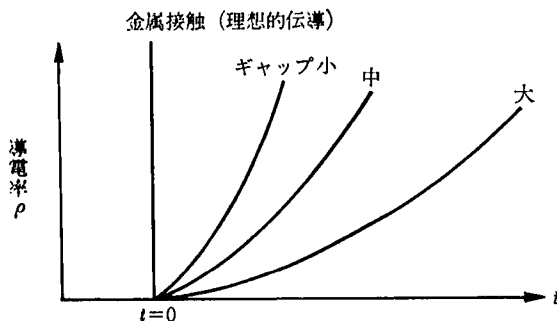


図 17 導電率の増加率とギャップの長さの比較を示す概念図

Fig. 17 Conceptual diagram showing the rate of increase of electric conductivity vs. gap length

における空気の導電率は時間とともに変わる。したがって、異なったギャップの長さにおいては時間とともに導電率 $\rho(t)$ が異なった増加率を示すことがわかる。

この関係を、概念図として図 17 に示す。放電のためのギャップが小さければ小さいほど、導電率はより早く増加する。H. Hyatt 他^[1]によると、電流上昇期間の抵抗フェーズ (τ_R) とそれに続くピークに至るまでの電流の増加期間の形成フェーズ (τ_i) は、放電電流の絶縁から導通に至る過程に関連して定義される。これは形成された τ_i が放電電圧の増加に伴い増加することを示し、われわれが行った実験の結果とよく一致していることを示す。

6.2 電流のピーク制限とシフト

放電中に、間隙の空気の導電率はいつも一定とは限らず、時間とともに変わる。また導電率は、ギャップの長さにも右左される。これらの事実を考慮すると、ピーク電流の比率は放電電圧の増加に比例せずまたピークの移動も生ずることがわかる。

図 16 において、コンデンサ C に電圧 V_0 がかけられたと仮定する。間隙の導電率 $\rho(t)$ が無限だと仮定すると、すなわちこの部分の抵抗成分がゼロとすると、ピーク電流 I_p は電流のみに比例する。しかし実際は、導電率 $\rho(t)$ は有限であり、時間とともに変化する。ピーク電流 I_p は、次の方程式で求められる。

$$I_p(t) = \frac{V_0}{R + \frac{1}{\rho(t)}}$$

さらに、導電率はタイム・ディレイ (遅延) に伴い増加する。この遅延の間、コンデンサ C に蓄積された初期電圧は減少し、ピーク電流はさらに減少する。その結果、実際に測定されるピーク電流 I_p は次の方程式で決定され、ピークの限界とピークシフト(ピーク値

に到達までの時間遅れ)を示す:

$$I_p(t) = \frac{V_0 - v(t)}{R + \frac{1}{\rho(t)}}$$

ここで、 $-v(t)$ はコンデンサから失われる電荷 (電圧) を示す。

理論的には、5 kV の放電電圧におけるピーク電流は 18 A であるが、実験で得られた値はわずか 9.7 A である。このことは、放電回路に実際に存在するインピーダンスに匹敵する電流制限作用が瞬間的に生じることを示唆している。このような高電圧の放電中に、3 ns のピーク移動が生じたことは注目すべきことである (図 7, 図 9)。

6.3 ARP (Amplitude Rate of change of current Product) 電圧振幅と電流変化率の積

実験から明らかなように、ESD による妨害電磁波が電子装置に影響する度合、すなわち EMI 作用は必ずしも放電電圧に比例しないことがわかった。なぜそのような現象が生ずるのかを調べるには、次の要因に注目するのがよい。

- 1) ギャップが大きくなると増加するもの
 - 放電電圧 (V)
 - 放電電流 (I)
 - コロナ損
- 2) ギャップが減少すると増加するもの
 - 電界 (E)
 - 電流の変化率または上昇率 (di/dt)
 - 周波数スペクトラム
 - 空気の瞬間的な導電率 ($\rho(t)$)

さらに、ESD によって生じた EMI 放射の強度は変位電流によって左右され、放電エネルギーに影響されないということも確かめられた (図 18, 図 19)。

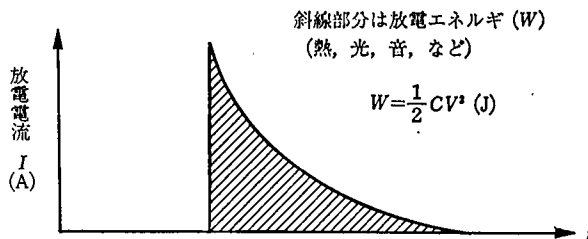


図 18 放電エネルギーの時間的変化
Fig. 18 Time dependency of discharge energy

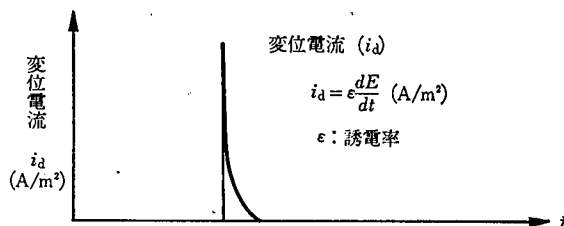


図 19 変位電流の時間的変化
Fig. 19 Time dependency of displacement current

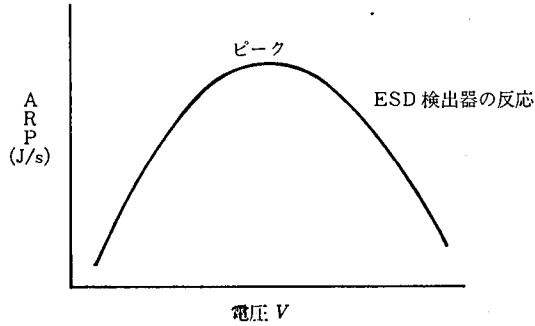


図 20 EMI と放電電圧

Fig. 20 EMI vs. discharge voltage

以上の現象と、放射された EMI が一定の放電電圧になるとピークに達することを示すデータ (ESD 検出器を使って得られた) とをうまく結び付けると、まったく新しい概念を提起することができる。

最終的に、われわれは ESD によって瞬間的に放射された EMI の強度を ARP (Amplitude Rate of change Product) によって表示し、その値によって EMI を判定することが適当であるという結論を引き出した (図20)。

$$\text{ARP} = V \times \frac{di}{dt} \quad (\text{J/s})$$

ここで V は放電電圧、 di/dt は放電電流の変化率である。

実際の電子装置に対する ESD の影響は、妨害波に対する作用断面積 (電子装置の物理的の大きさによって決まる) によっても左右される。したがって、有効な ARP は次のように書き直すべきであろう。

$$\text{WARP} = V \times A \times \frac{di}{dt} \quad (\text{J/m}^2/\text{s})$$

ここで、WARP はワーキング ARP であり、 A は有効断面係数である。

この WARP は、ESD によって生ずる過渡的な電磁エネルギーの流れを示し Poynting のベクトルと同じ次元をもつ。この値が大きくなればなるほど、EMI 作用は大きくなる。

7. おわりに

新しく開発した ESD 検出器によって ESD による EMI の影響を調査した結果、次のような結論を得た。

- 1) ESD による EMI 作用は ARP によって決定される。
- 2) ESD による妨害電磁波の放射は必ずしも放電電圧に比例しない。
- 3) このことはまず ESD 検出器を使って発見され、また既存の測定装置を使っても検証された。
- 4) EMI が電圧と必ずしも比例しない理由は、放電電流の上昇時間が放電間隙の増大につれて長くなることにある。
- 5) 低電圧での ESD であっても無視すべきではない。また、静電気は低電圧に帯電する機会が多いことも知っておく必要がある。

執筆者紹介 本田 昌 實 (Masamitsu Honda)

昭和18年生，40年北海道大学 工教 電気科卒業。同年4月日本ユニパック(株)入社，大型計算機の保守に従事。その後，稼動環境，とくに計算機に与える電磁妨害，静電気障害対策の調査，研究に従事。現在，プロダクト本部ハードウェアプロダクト4部サポートエンジニアリング室，兼カスタマーサービス本部カスタマーサービス企画部業務推進室。

静電気研究懇談会会員，静電気学会会員，電子通信学会会員，米国電子戦学会(AOC)会員，米国 EOS/ESD 協会会員。



川 村 雄 克 (Takeyoshi Kawamura)

昭和8年生，34年横浜国立大学工学部卒業。昭和37年三基電子工業(株)入社。以来，主としてEMCテストおよびシミュレータ開発に従事し，現在に至る。電気学会(事業維持員)会員，静電気学会(賛助)会員。



報告 ポスト・ストア・モードにおける ディスク・キャッシュの飽和現象

Saturation in Disk Cache-Post Store Mode

長尾 秀実, 永田 浩一

要約 他社コンピュータ・メーカーのほとんどが、キャッシュ・メモリの電源故障時のファイル・リカバリのむずかしさを理由にストア・スルー・モードを採用しているのに対し、UNIVAC シリーズ 1100 のディスク・キャッシュはリカバリ手法を研究し、より実効アクセス速度を短縮できるポスト・ストア・モードを採用することにした。しかし、このポスト・ストア・モードにも「ディスクに書き込みが済んでいないデータ・ブロックがキャッシュ・メモリにたまってしまい、実効アクセス速度が遅くなる」という弱点があった。これは、ディスク・キャッシュに対するファイル・アクセス方法の考慮不足や置換アルゴリズムの不備によるものであるが、ディスクに書き込む動作を効果的に行い、キャッシュ・メモリの空き領域を確保することで解決した。

現在、ディスク・キャッシュは、従来のディスク装置と比べ 3～5 倍、また当初開発されたディスク・キャッシュと比べ 1.5～3 倍の速さを実現している。

Abstract Most of the competitors adopt the store-thru mode because of the difficulty in file recovery from cache memory due to power supply failures. After researching into the file recovery procedures, Univac adopted the post store mode for its disk cache.

Post store is a better caching mode to achieve the high speed access of disk storage devices than the store-thru. However, the saturation due to the accumulation of non-destaged segments in the cache memory caused to slow down the mean access time on the first developed disk cache. This paper focuses the saturation problem and presents the results of the study.

The problem was because the file access method of disk cache had not been adequately examined and the mode was not fully provided with a replacement algorithm. To solve this problem, the usable space in the cache memory was reserved by improving the destaging efficiency.

The current disk cache is about 1.5 to 3 times faster than the first developed disk cache.

1. はじめに

UNIVAC シリーズ 1100 のディスク・キャッシュは、昭和 52 年以来開発が続けられていたが、56 年 12 月に開発評価用サブシステムを日本ユニバックに設置し、各種評価テストを開始した。その後、入出力命令効率の測定結果を参考にして、より効率のよいディスク・キャッシュをめざした新たな開発を開始した。

本稿では、この効率改善のための開発に参加した経験から、改善前のディスク・キャッシュにおける問題点を分析し、どのような対策を施し、どのような効果を得たかについて述べる。

なお、改善前のディスク・キャッシュをミディアム・パフォーマンス・キャッシュ・ディスク (Medium Performance Cache Disk: 以下 MPCD と略す)、改善後のディスク・キャッシュをハイ・パフォーマンス・キャッシュ・ディスク (High Performance Cache Disk: 以下 HPCP と略す) と呼ぶ。

2. UNIVAC シリーズ 1100 のディスク・キャッシュ

ディスク・キャッシュの種々のアルゴリズムについて理解してもらうために、まず、当社のディスク・キャッシュについて概要を述べる。

UNIVAC シリーズ 1100 のディスク・キャッシュは、8470 型ディスク装置を制御する 5057 型キャッシュ・ディスク・システム・プロセッサ (以下 CDS プロセッサと略す) に、7053 型のキャッシュ・メモリを仮想記憶用に使用する。ディスクは 1792 語 (36 ビット) を 1 ブロックとして区画され、キャッシュ・メモリの基本ブロック (セグメントと呼ぶ) もこれに合せ、1792 語としている。接続できるディスク装置の数は最大 16 台であり、キャッシュ・メモリの構成は 512 セグメント (約 1 メガ語) から最高 8192 セグメント (約 14 メガ語) まで拡張できる。

命令形態としてはポスト・ストア^[1] (ストア・インとも呼ぶ) を基本とし、ストア・スルー^[1]も選択できるよう考えられている。キャッシュ・メモリとのマッピング方式^(注1)にはフル・アソシティブ方式^[1]を採用し、限られた容量のキャッシュ・メモリを最大限に利用するよう工夫されている。またキャッシュ・セグメントの置換 (リプレースメント) アルゴリズムは、最も高いヒット率^(注2)を得ることができるといわれている^[2]。LRU (Least Recently Used) 方式^[1]を基本とし、ポスト・ストア特有のファイル・リカバリ方法を考慮したアルゴリズムを追加している。

- 1) ポスト・ストア……ポスト・ストア・モードの出力(書込み)命令は、キャッシュ・メモリに書込みが終了した時点で完了し、ディスク装置への書込みは、CDS プロセッサの空き時間を利用して行う。ディスク装置への書込み動作が見かけ上省略されるので、実効アクセス時間は短い。読み出されるデータは、ミスの場合、ディスクから中央処理装置 (以下 CPU と略す) とキャッシュ・メモリへ同時に転送される。このデータはキャッシュ・メモリに存続し、次の入出力命令がヒットするのを期待する。ヒットした場合には、キャッシュ・メモリとのデータ転送を行うので、ディスク装置からの読出し動作は必要としない。また、ヒットしたセグメントは LRU アルゴリズムにもとづき、最も置換されにくいように管理される。
- 2) ストア・スルー……ストア・スルー・モードでの出力命令は、キャッシュ・メモリに書き込むだけでなく、ディスク装置に対しても書き込んだ時点で完了する。データはキャッシュ・メモリにも存在し、ディスク上も同じデータを保存しているため、ファイル・リカバリの面からはポスト・ストアより容易であるが、常にディスク装置へのアクセスを伴うため実効アクセス時間は延びる。
- 3) トリickle……ポスト・ストア・モードでキャッシュ・メモリに書き込まれたデータをディスク装置へ書き込む動作をトリickle (Trickle) と呼ぶ。トリickleは CPU 側の制御範囲から外れ、CDS プロセッサが忙しくない時間に少しずつ行う^(注3)ように考えられている。トリickleの対象となるセグメントには、WT^(注4) (Written To) 表示があり、古さを示すタイム・スタンプ^(注5)によって処理順番が決められる。
- 4) バイパス・キャッシュ・モード……入出力命令は 1 語から 65 キロ語まで、任意のデータ長を実行できるが、あまり大きいデータ長^(注6)はその命令だけでキャッシュ・メモリを占有してしまうため、キャッシュにはのせないように考えられている。また、データ長とは別に、キャッシュ・メモリの空き領域がなくなった場合にもキャッシュはバイパスされ、ディスク・モードにて入出力命令を実行する。
- 5) 命令待ち行列 (コマンド・キューイング)……CDS プロセッサは、ディスク装置当

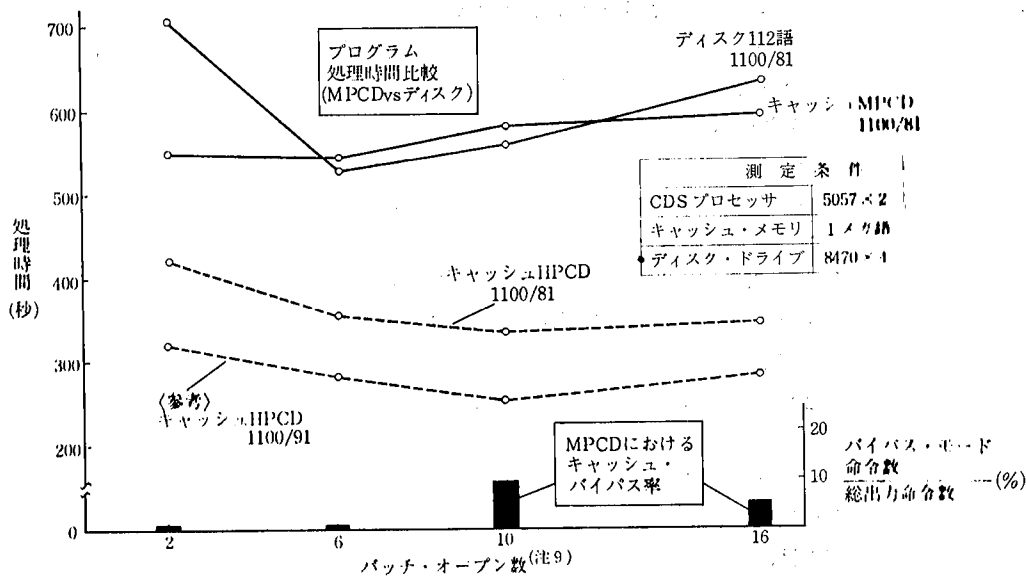


図 1 効率評価プログラム処理時間

Fig. 1 Elapsed time on CDSTAR

たり 8 個の命令を待ち行列に並べて制御できる。8 個のうち 1 個はトリックル専用
に、他の 7 個は CPU からの命令用に割り当てている。したがって、CPU は命令完了
を待たずに 7 個まで異なる命令を出すことができるため、制御用オーバ・ヘッドは従
来のディスク制御装置より軽減されている。

3. 初期ディスク・キャッシュの問題点

入出力命令の実効アクセス時間を短縮することができるはずのディスク・キャッシュだ
が、初期に開発された MPCD において効率評価用プログラム^(注7)を使用し、1 秒当たり
に処理できる入出力命令回数と処理時間^(注8)について測定してみたところ、次のような結
果であった。

- 1) 効率評価用プログラムの処理時間が、112 語レコードで区画された従来のディスク
装置での処理時間と比較して、短縮されていない (図 1)。
- 2) 1 秒当たりの入出力命令回数が低い値を示す領域では、ディスクよりも実行時間が
短く、ディスク・キャッシュの効果が見られるが、入出力命令回数が増すと、ディス
クよりも遅くなる傾向がある (図 2)。

この現象は、入出力命令のトレース・データおよび CDS プロセッサの制御状態を分析
することによってその原因を解明することができた。以下に、この分析結果を述べる。

3.1 キャッシュ・メモリの飽和

1 秒当たりの入出力命令回数が高い値を示している時間帯の CDS プロセッサの制御状
態を、マイクロプログラムを工夫して調べたところ、キャッシュ・メモリ中に WT 表示
のある (まだトリックルできていない) セグメントがたまっていることに気付いた。この
現象は、書き込み命令の多いプログラムを使用すると顕著になる。

UNIVAC シリーズ 1100 のディスク・キャッシュでは、他社のディスク・キャッシ

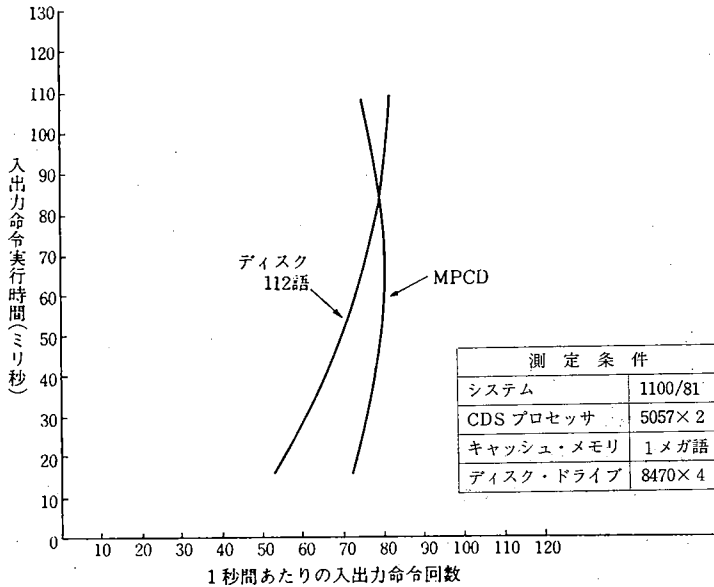


図 2 MPCD とディスク 112 語の効率評価
 Fig. 2 Mean service time and I/O counts

ユ^[3,4,5], (注¹⁰) とは異なり, ポスト・ストア・モードを基本としていることは前にも述べたが, この長所^[2]を生かすため, WT 表示のあるセグメントは CDS プロセッサの空き時間に, ディスクに書き込む方法をとっている. しかし, 入出力命令の応答時間が短くなると CDS プロセッサの空き時間が減少し, さらに CDS プロセッサの特徴ともいえる命令待ち行列機能が, この現象を起こしやすくしているとも考えられる. キャッシュ・メモリが飽和状態になる原因をまとめると, “CDS プロセッサの空き時間が減少するため, トリックル処理の回数が減り, WT 表示のあるセグメントの数を適正に制御することができなくなる” ということである.

キャッシュ・メモリが飽和すると, ヒット率が低下し, 最悪の場合, キャッシュ・セグメントの置換ができなくなり (バイパス・キャッシュ・モード), ディスク装置へのアクセスが増え, 実効アクセス時間は延びる.

3.1.1 ヒット率の低下

キャッシュ・メモリ WT 表示のあるセグメントがたまり, 使用可能領域が少なくなると, それ以後の入出力命令でアクセスされるデータは, キャッシュ上に一たんのせられたとしても, すぐ別の入出力命令によってそのセグメントが置換されてしまう. LRU アルゴリズムでは, 最も使用頻度の少ないものを置換の対象と考えるが, WT 表示のあるセグメントは置換できない. やむをえず, WT 表示のないセグメントを探して置換するため LRU のアルゴリズムは無視されてまう. 当然のことながらヒット率は低下し, 期待する入出力命令効率を得られなくなる.

3.1.2 ディスク・アクセスの増加

3.1.1 項で説明した現象がより悪化し, キャッシュ・メモリが飽和しきってしまう (キャッシュ・メモリが WT 表示のあるセグメントで埋まってしまう) と, 入出力命令はキャッシュを使用できずキャッシュに空き領域ができるまでの間, 直接ディスクにアクセス

する。図1に示すとおり、効率評価用プログラムでの測定の際、確認できたこの現象は最高9パーセントにも達し、ある時間帯では無視できないほどキャッシュはバイパスされている。キャッシュがバイパスされ、ディスク・アクセスが増えると、“入出力命令単位が、ディスクのレコード長と合っていないため、出力命令ではディスク1回転でその動作を完了できず、最低2回転、場合によっては3回転かかる”という問題が派生する。

- 1) 112語レコードと1792語レコード……ディスク・キャッシュでは、基本管理ブロックを1792語とし、ディスク上も1792語レコードに区画している。しかし、レコード単位は、入出力処理の効率と、適度なデータ容量を得るという観点から、112語あるいは448語とし、アクセス方法も112語/448語に合わせてきた。効率評価用プログラムのアクセス形態を分析してみても、112語レコードで割り切れるディスクへのアクセスは75パーセントあるが、1792語レコードで割り切れるアクセスは1パーセントに満たない状況である。

図3に、112語レコードに区画したディスク・モード、1792語に区画したディスク・モードおよびバイパス・キャッシュ・モードでの入出力命令回数と実行時間の関係を示す。1792語で区画したディスクをディスク・モードあるいは、キャッシュをバイパスして使用すると入出力命令効率が低下することがわかる。

- 2) オンバウンダリ・アクセスとオフバウンダリ・アクセス……ディスク装置はレコード内のデータを1部分だけ読み書きすることができない。この特性は、入出力命令のうち出力命令において、効率上不利となる。なぜなら、更新しない部分のデータを、更新する部分のデータとマージするために、あらかじめ読み込まなければならないからである。図4(a),(b)にこのメカニズムを示すが、ファイル・アクセスの単位がディスク・レコード長で割り切れない(オフバウンダリ)出力命令では、割り切れる(オンバウンダリ)の場合より、命令実行時間が2~3倍になってしまう。

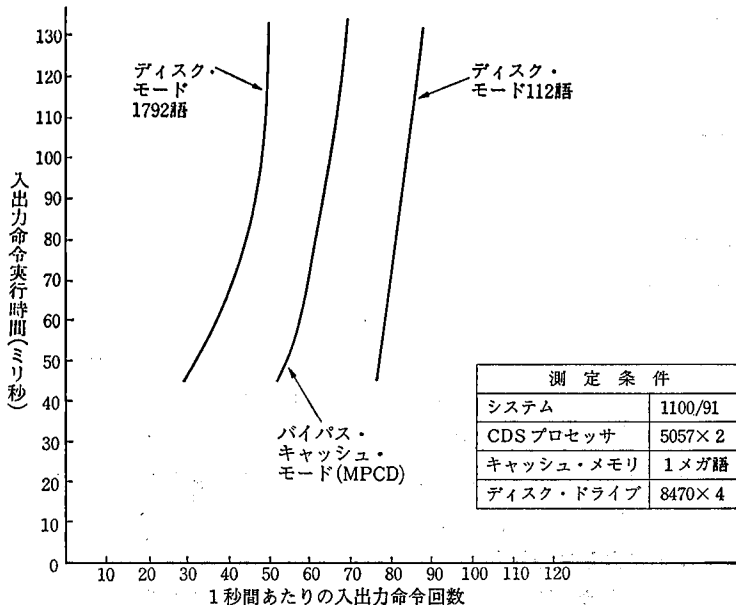


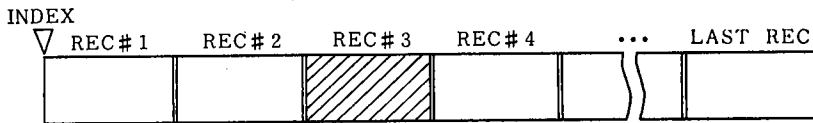
図3 ディスク・レコード長、およびバイパス・キャッシュによる効率比較
 Fig. 3 Disk access and bypass cache

3.2 その他の問題

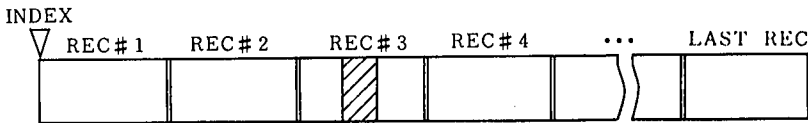
キャッシュ・メモリの飽和が入出力命令効率に及ぼす影響のうち、①ヒット率の低下によるもの、②キャッシュが使用できなくなることによるもの、③ファイル・アクセス方法とディスク・キャッシュのアルゴリズムの不一致によるもの、については3.1節で述べたが、この飽和現象は、LRU 置換アルゴリズムにも問題を提起した。


3.2.1 置換セグメントの検索時間

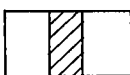
LRU 置換方式は、もっとも昔に参照されたものを置換の対象とする考え方であり、ヒット率を向上させるという面では優れているといわれている。しかし、ポスト・ストア・モードを採用しているディスク・キャッシュでは、“まだディスクに書き込みが終わって



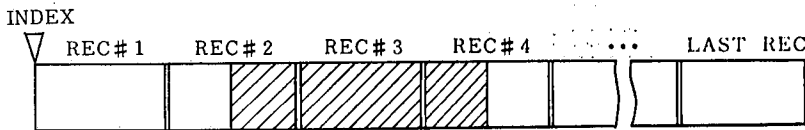
ディスク・レコード・オンバウンダリ・アクセス(1回転にて完了)



1回転目  REC#3を読み込む

2回転目  更新データをマージ後ディスクに書く

ディスク・レコード・オフバウンダリ・アクセス-1(2回転にて完了)



1回転目  REC#2を読み込む

2回転目  REC#2に更新データをマージ後
REC#2, および3をディスクに書き
REC#4を読み込む

3回転目(注11)  REC#4に残りの更新データを
マージ後ディスクに書く

ディスク・レコード・オフバウンダリ・アクセス-2(3回転にて完了)

 更新箇所

図 4(a) ディスク・レコード・アクセスの形態
Fig. 4(a) Method of off-boundary disk access

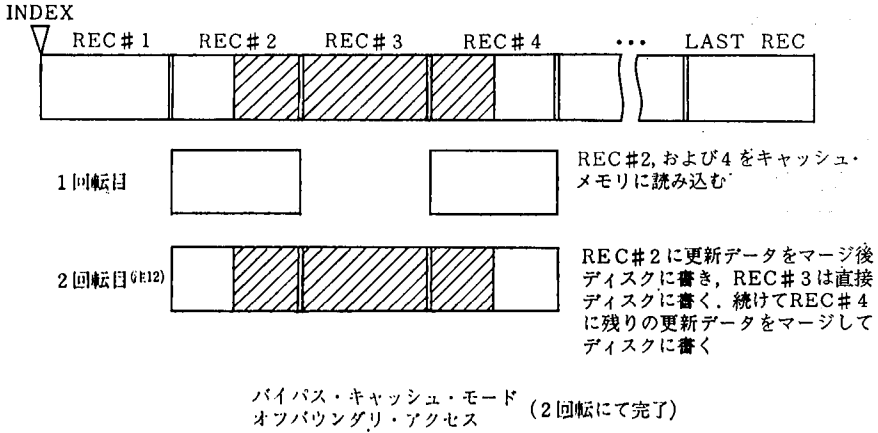
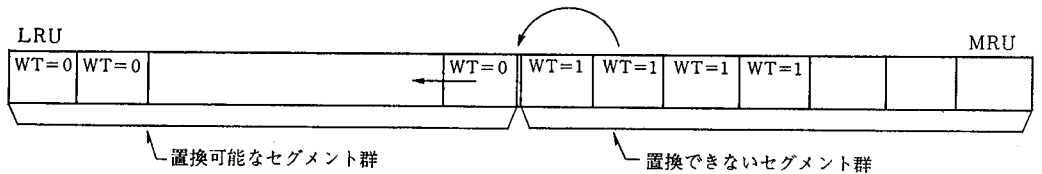
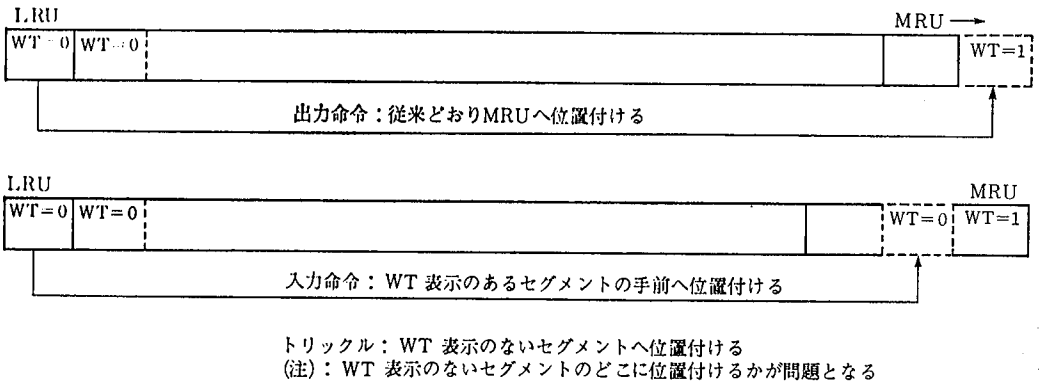
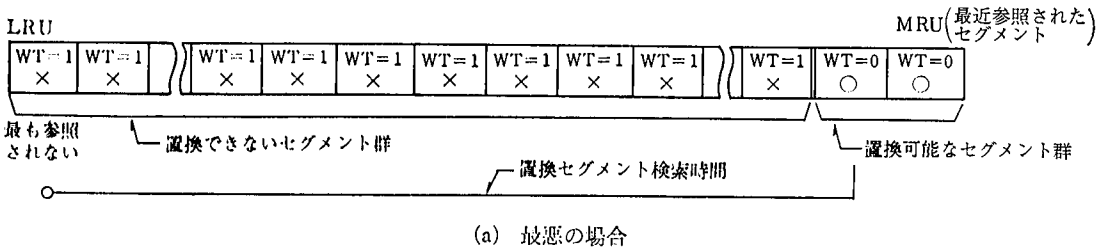


図 4(b) キャッシュ・バイパス時のディスク・アクセス形態
Fig. 4(b) Method of disk access in case of bypass cache



(b) モディファイド LRU 置換方式
図 5 セグメント置換方法
Fig. 5 Modification of LRU algorithm

ないセグメントは、それが最もアクセスされないセグメントであったとしても、置換することができない”という問題が生じる。

これは、フル・アソシティブ方式によりマッピングしているディスク・キャッシュにおいて、より重大な問題となりうる。図5(a)にこの現象の最悪の場合を想定したものを示すが、置換のできる(WT表示のない)セグメントを探す作業に時間がかかりすぎる場合がある。MPCDにおける効率プログラムで確認したこの状態を表1に示す。

表 1 置換セグメント検索状態
Table 1 Search for the replaceable segments

	MPCD	理想
総入出力命令回数	46252	—
置換要求回数	23217	—
セグメント検索回数 (LRUにて見つからなかった)	426549	0
平均検索時間(ミリ秒)	2.6	0.028

測定条件: 5057×2, 8470×4, 1メガ語,
バッチ・オープン=10

以上、3.1.1項から3.2.1項で述べたディスク・キャッシュの飽和の原因と影響を図6にまとめて示す。

4. 原因と改善策についての考慮

MPCD ディスク・キャッシュ・メモリでは、キャッシュ・メモリの飽和という弱点があった。使用可能なキャッシュ領域を確保するために、強制的にWT表示のあるセグメントをディスクに書き込む方法も考えられるが、ポスト・ストア・モードの長所である“CDSプロセッサの空き時間に、トリックルによってキャッシュ・メモリの使用可能領域を作り、CPU側に与える影響を最小にする”という考え方は生かしておきたい。そこで何らかの方法によって、ディスク・アクセス速度をあげることができないかと考えてみた。

4.1 効率よいトリックル処理

ディスク・データ・ブロックをアクセスする時間 T_D は、次式で表すことができる。

$$T_D = T_S + T_R + T_B + \alpha$$

ただし、 T_S : 位置決め時間, T_R : 回転待ち時間, T_B : ブロック転送時間, α : 制御用オーバーヘッド, である。

従来のトリックルの方法は、 $T_{D1} \rightarrow T_{D2} \rightarrow T_{D3} \rightarrow \dots$ と順番に処理していくので、次のように展開できる。

$$\begin{aligned} T_{D-Total} &= T_{D1} + T_{D2} + T_{D3} + \dots \\ &= (T_{S1} + T_{R1} + T_{B1} + \alpha_1) + (T_{S2} + T_{R2} + T_{B2} + \alpha_2) + T_{S3} + T_{R3} + T_{B3} + \alpha_3 + \dots \end{aligned}$$

ここで、ディスクの特性とファイルの連続性を考慮すると T_S と T_R は工夫することにより省略できる。つまり、一度位置決めし、回転待ちしたレコードを処理した後、それに引き続きレコードを連続してアクセスできれば、上式のうち、 $T_{S2,3}$ と $T_{R2,3}$ は無視することができる。この方法は、先読み^(注13)(プリ・フェッチとも呼ぶ)機能に応用されているもので、もしトリックル処理にも応用できれば、見かけ上、ディスク・アクセスを速く

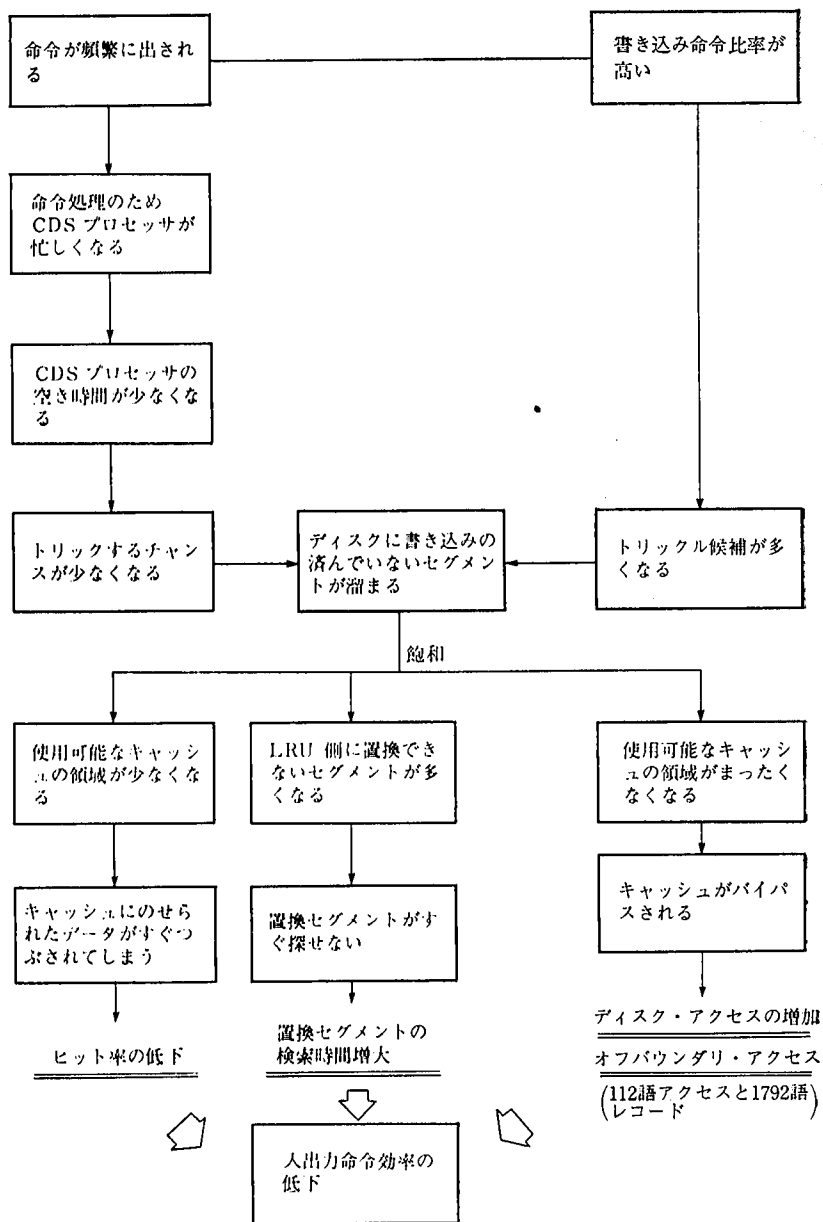


図 6 ディスク・キャッシュ飽和の原因と影響

Fig. 6 Saturation of disk cache system

することができるし、キャッシュ・メモリの空き領域を作ることに効果がある。

4.2 モディファイド LRU アルゴリズム

3.2.1 項で問題となった置換セグメントの検索時間に関しては、図 5 (b) に示す方法を考え、試行してみた。この方法では、必ず 1 回の検索動作で置換すべきセグメントが見つかるが、トリックルを行ったセグメントをどこへ位置付けるかについては最善の方法が見つからなかった。また、この検索動作の無駄による時間は、ディスク・アクセス時間に比較して、はるかに小さく、効率評価プログラムでの効果測定では、効果が目立たなかった。後に述べる“効果的なトリックル”に対しての改善策が、この変更による効果を小さ

くする働きをもつためとも考えられる。したがって、この新しい LRU アルゴリズムは採用せず、検索開始位置をダイナミックに動かさずという代替案を採用することにした。

5. 適用した改善策

MPCD の弱点とされた問題点については、次の改善策を適用した。

5.1 キャッシュ・メモリ飽和に対する改善策

キャッシュ・メモリ飽和を防ぐ方法としては、トリックル処理を効果的に行う方法と優先順位を上げる方法を対策とした。

5.1.1 複数セグメントのトリックル

トリックル処理は CDS プロセッサの空き時間に、WT 表示のある最も古いセグメントを探して命令を待ち行列に並べ、待ち行列の命令として実行する方法を採っている。しかし、一度に処理できるのは1セグメントであったため、トリックルが命令待ち行列からとりあげられ実行されても、使用可能セグメントは1個しか増えず、つぎに他の命令でそのセグメントが割り当てられると元の状態に戻ってしまい、十分な使用可能領域は確保できない。また、連続したディスク・レコードに相当するセグメントに WT 表示があった場合を考えても、一度に1セグメントを書き込む方式ではディスク装置の使用効率もよくない。なぜなら、ディスク装置に対する命令はトリックルだけではなく、別のディスク・アドレスを指定する入出力命令があるから、ディスク装置のヘッドは異なる位置へ動かされてしまい、次のトリックル処理の際に、もう一度元のところへ位置決めしなおさなければならぬからである。ヘッド位置決め時間は平均 23 ミリ秒 (8470 の場合) であり、キャッシュのアクセス時間^(注14)に比べてはるかに大きい。もし連続した複数セグメントを一度に処理できたなら、4.1 節に示した T_D を見かけ上小さくすることができることになる。

複数セグメントのセグメント数を決定するに当たっては、アプリケーションによる影響を受けやすく苦労したが、CDS プロセッサを占有する時間をできる限り小さくし、ディスクの使用効率も向上させる、ディスク1回転の時間を選んだ。この時間はセグメント数に換算すると4セグメントとなる。図7に HPCD で測定した1セグメント、4セグメントおよび32セグメント・トリックルの入出力命令効率の違いを示す。

5.1.2 トリックル処理の優先順位

待ち行列に並んだ命令は、古いものから処理されるよう考えられているが、トリックルについては次の条件で優先順位が上げられる。

- 1) キャッシュ・メモリ中の WT 表示のあるセグメントが増加したとき
 - 2) WT 表示のあるセグメントがトリックルされずに長時間経過したとき
- どちらかの状態になったとき、トリックルの待ち行列中の優先順位が古さによる順番よりも重視される。

したがって、キャッシュ・メモリが WT 表示のあるセグメントでいっぱいになった場合のトリックルは、待ち行列の中では一番優先されるよう考慮されている。しかし、トリックルを命令待ち行列へ並べる作業は、CDS プロセッサの空き時間に行われるよう考えられていた (図8参照) ので、入出力命令が頻繁に出される環境下では CPU の命令処理、ディスク装置からの位置決め完了インタラプト処理および、すでに待ち行列に並んでいる命令処理などで CDS プロセッサの空き時間が少なく、トリックル候補を命令待ち行列に並べる作業が容易でなくなる。このため、待ち行列中の優先度は高いが待ち行列に並ぶことができず、トリックルが処理されないという悪循環を繰り返していた。

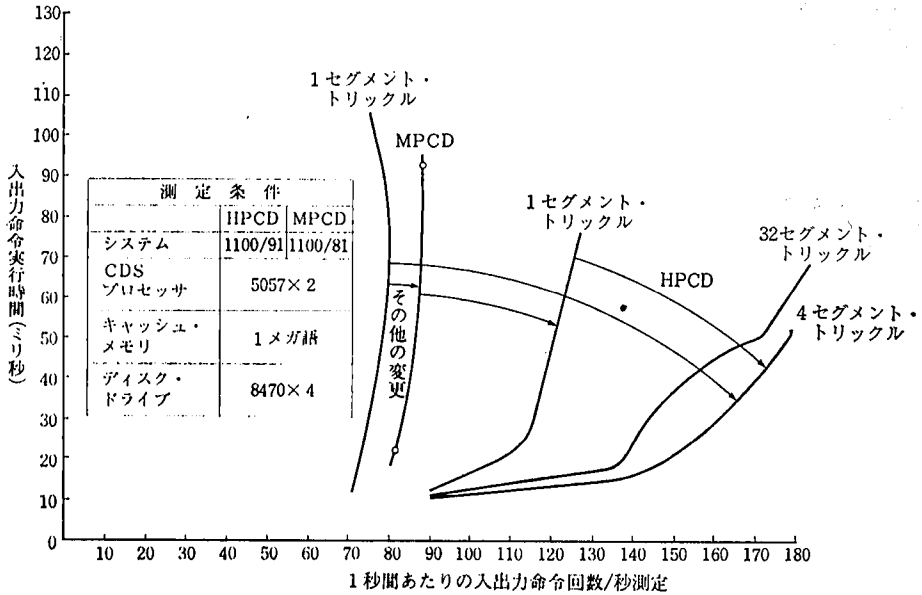


図 7 各改善項目による効果

Fig. 7 Several improvement results

この状態を解決するために、すでに待ち行列に並んでいたトリックル以外の命令を処理した直後、強制的につぎにトリックルすべき候補を待ち行列に並べるロジックを追加し(図8の(注)参照)、従来から考えられていた待ち行列中の優先順位のアルゴリズムを有効にした。

5.1.3 トリックル候補の検索方法

WT表示のあるセグメントは、その命令がいつ出されたかを示すタイム・スタンプの古い順に並べられている(これをタイム・スタンプ・リンクと呼ぶ)。MPCDでは、全体で一つのリンクを構成していたため、ディスク装置が異なっても、リンクの先頭から順番に該当するセグメントかどうか探していかなければならず、検索動作にむだがあった(図9)。

この不都合は、ディスク装置ごとにタイム・スタンプ・リンクをもつことにより解決した(図9(b))。

5.2 置換セグメント検索開始位置の改善

図10に示すとおり、改善方針は一度検索に成功したセグメントの位置を覚えておき、それに引き続く検索にはその次の位置を開始位置として用いるという考えである。トリックルが終了した時点で、この開始位置を振り出しへ戻すという制御を行うことによってLRUアルゴリズムが踏襲される。この方法では完全に検索動作の無駄を解決することはできないが、5.1節での改善方法と組み合わせて使用することによって入出力命令効率に与える影響は小さいため、これ以上の変更は行わなかった。

6. 改善効果

5章で述べた改善策はHPCDに適用され、図11に示すような効果をあげた。CPUの処理能力によっても異なるが、UNIVAC 1100/81システムでの効率評価用プログラムに

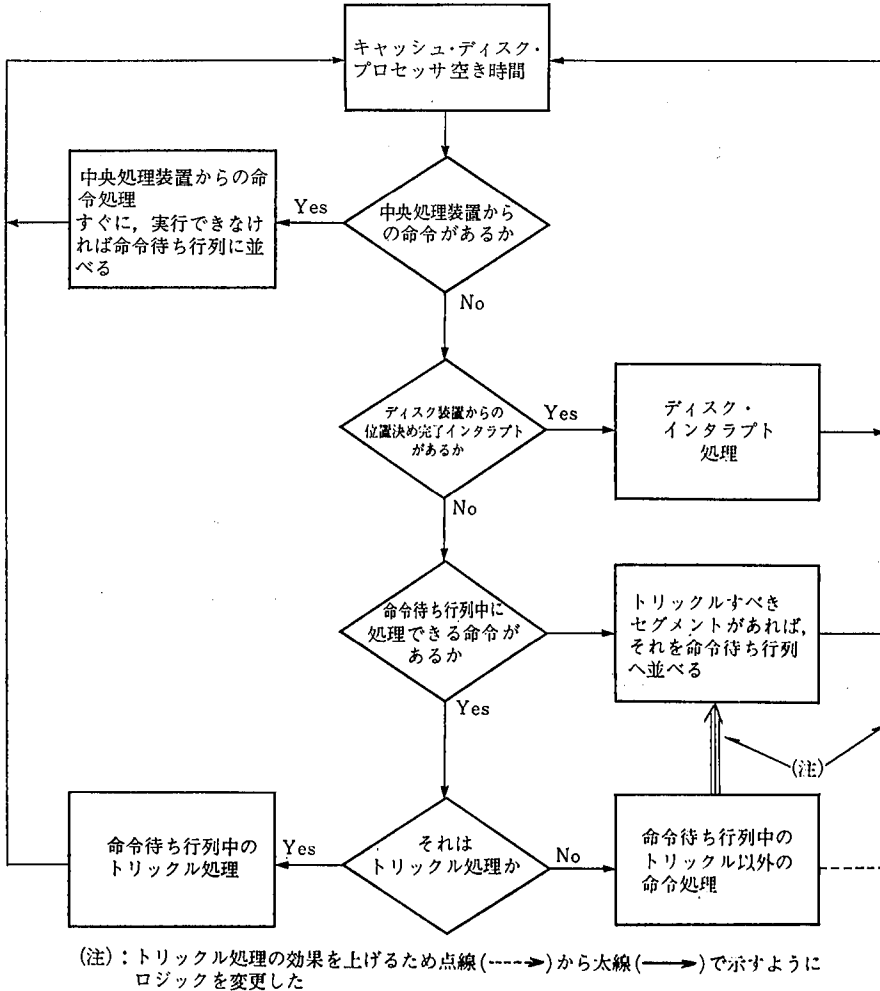


図 8 トリックル効果の改善

Fig. 8 Modified trickle command queuing

よる測定からは、次の結果を得た。

- 1) HPCD でのプログラム処理時間が MPCD の処理時間の約 1/2 に短縮できた (図 1)。
- 2) 入出力命令実行時間と入出力命令回数の関係では HPCD の方が MPCD に比べ 1.68~3.33 倍優れている (表 2, 図 11)。

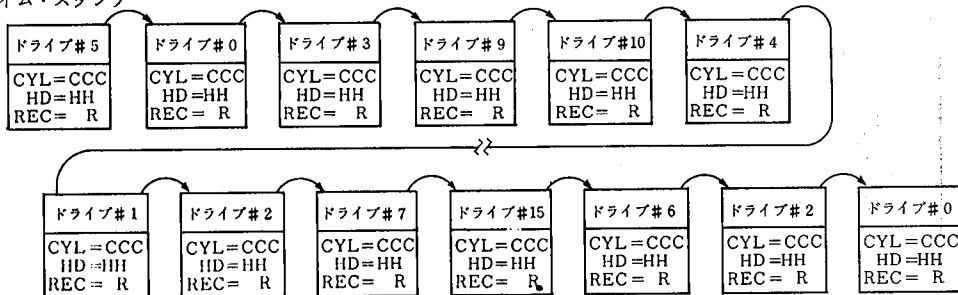
7. おわりに

HPCD 開発作業では、すでに開発されていた MPCD の弱点を補い、より高速なディスク・キャッシュを短期間で実現しなければならなかったことから、基本的なアルゴリズムを変更しないで、キャッシュ・メモリの飽和に留意し、ディスク・アクセスを少なくすることに重点を置いて検討してきた。

元来、キャッシュ・ディスク・システムは、シーケンシャル・アクセスを指向して開発されてきた。低速大容量であるディスクとのデータ授受の単位を 1792 語と大きく設定したのは、そのレコード内にホストとの授受されたデータが数多く存在することが期待され

最古の

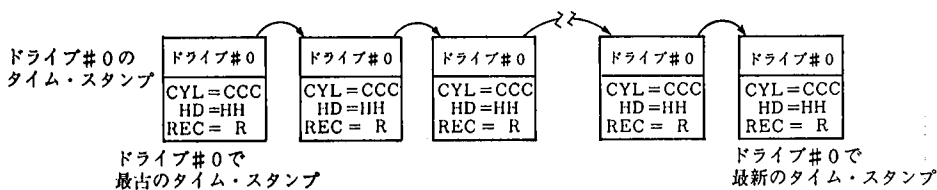
タイム・スタンプ



最新の
タイム・スタンプ

(a) 従来のタイム・スタンプ・リンク (MPCD)

MPCD time stamp link



(b) ドライブごとのタイム・スタンプ・リンク (HPCD)

Per-Disk time stamp link

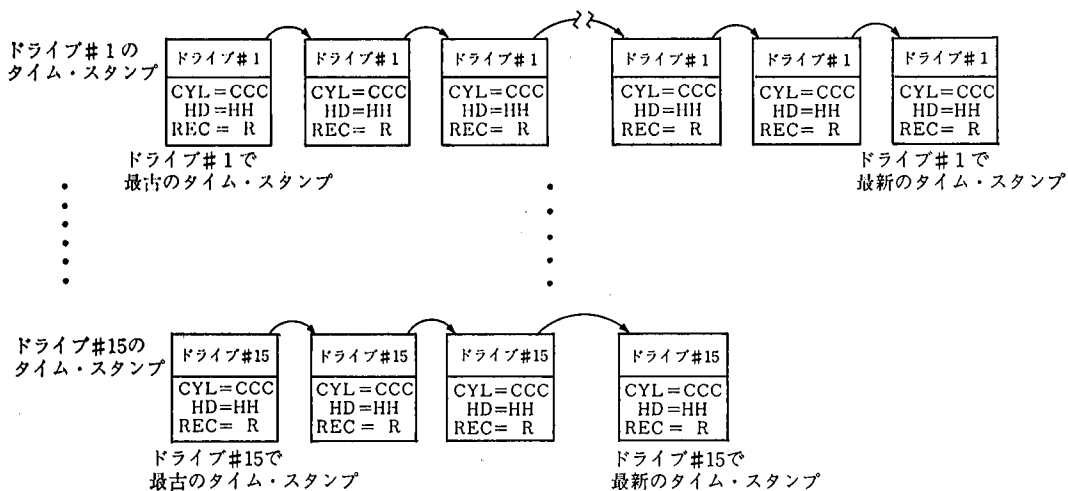


図 9 タイム・スタンプ・リンクの改善

Fig. 9 Improvement of time stamp link

るためであり、ホストとは小出しに高速のデータ授受をすることで入出力命令処理の高速性を生み出す意図があった。また 1792 語レコードにディスクを区画することで、パック当たりのデータ容量は 112 語レコードのそれと比べ 6 割以上増える、というメリットがある。利用者の目的ファイルが 1792 語より小さな単位でシーケンシャルにアクセスされれば、高いヒット率を維持でき、高速アクセスが実現される。

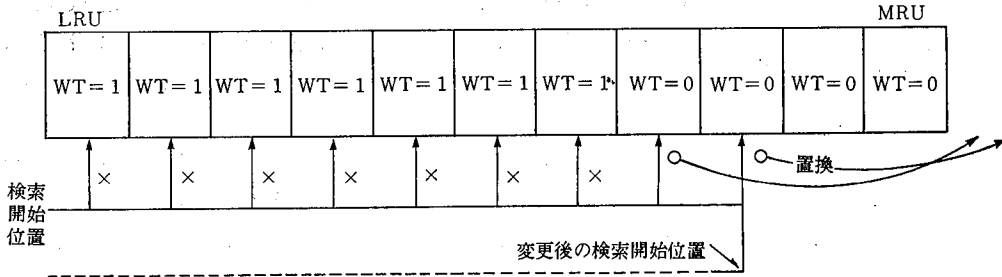


図 10 置換セグメント検索方法の改善
Fig. 10 Modified empty segment search

表 2 命令実行時間と回数の関係
Table 2 HPCD/MPCD figures of merit

バッチ・オープン数	メリット = $\frac{\text{秒当たり入出力命令回数}}{\text{命令実行時間}}$		$\frac{\text{HPCD のメリット}}{\text{MPCD のメリット}}$
	HPCD	MPCD	
2	4.48	2.91	1.68
6	2.84	1.13	2.51
10	2.02	0.63	3.21
16	1.73	0.52	3.33

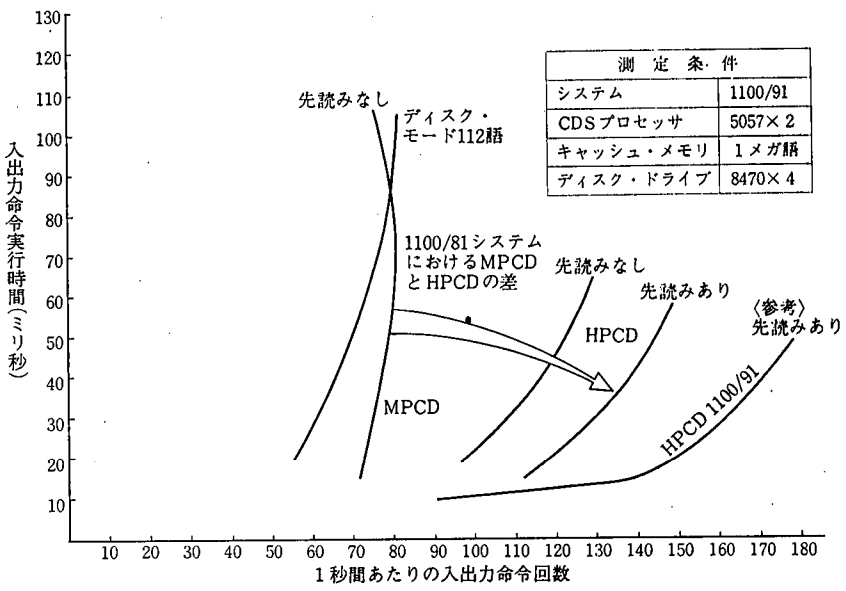


図 11 MPCD と HPCD の差
Fig. 11 HPCD improvement results

しかし、キャッシュ・ディスク・システムへの入出力命令がランダム・アクセスに近い場合、ヒット率が低下し、したがってディスク・アクセスが増大する。この状態はシーケンシャル・アクセスでも、キャッシュ・メモリが飽和状態となれば起きうる。現状のキャッシュ・ディスク・システムの入出力命令の分析では、1回のデータ転送が112語の整数

倍であることが多く（全体の75パーセントに達する）、しかも1792語の整数倍であることは非常に少ない（同1パーセント未満）ことから、1キャッシュ・セグメントを1792語と大きく設定していることが結果的に効率の悪いディスク・アクセスが増えるということになる。

この問題は1キャッシュ・セグメントを112語に設定し、ディスク上も112語レコードとして管理することで改善できると思われる。シーケンシャル・アクセスには先読み機能を有効に生かすことでヒット率に支障は出ない。データ容量が低下することとキャッシュ・メモリの制御情報が増大するデメリットは避けられないが、各アクセス形態、状況での高速化を望めるメリットは大きい。

今後、さらに高性能化されるCPUに合わせ、システム全体の処理能力を向上するため、1秒当たり入出力命令回数が増えても飽和しない（あるいは、飽和したとしても実行時間は延びづらい）高速のディスク・キャッシュを検討していくことが必要と考える。

以上、UNIVAC シリーズ1100のディスク・キャッシュの入出力命令効率を題材にして、現在までに対処してきた問題点、今後考慮しなければならない項目について、ハードウェア、とくにマイクロプログラム側から考察してきた。コンピュータ・メーカ他社をみても、ディスク・キャッシュの最適アルゴリズムの選択を試行錯誤しているのが現状であるが、今後とも、日本ユニパックのホスト・システムに合った方法を追求し、次世代のディスク・キャッシュに反映させていくよう努力したい。

-
- 参考文献** [1] 長島重夫他, “キャッシュ記録”, 情報処理, Vol. 21, No. 4, pp. 332~339, 1980,
 [2] ロイ・チャネイ, “磁気ディスクの転送速度を5~15倍に上げるキャッシュの手法”, 日経エレクトロニクス, 1984. 8. 29, pp. 249~273.
 [3] 平野正信, “アクセス・ギャップを埋めるディスク・キャッシュの機能を見る”. 日経エレクトロニクス, 1982. 3. 22, pp. 71~85.
 [4] 金子 悟他, “ディスク・キャッシュ・システム”, FIJITSU, Vol. 34, No. 2, pp. 253~261, 1983.
 [5] 猪苗代勉他, “総合ディスク・キャッシュ・システム”, 日本電気技報, No. 133, pp. 45~50, 1980.
- 執筆注** (注1) 1ディスク上のデータ・ブロックをキャッシュ・メモリのどの部分にのせるかについての方式。
 (注2) 目的とするデータがキャッシュ・メモリ上にある状態をヒット, ない状態をミスと表現する。ヒット率が高いというのは, データがキャッシュ・メモリ上にある確率が高いことである。
 (注3) 1回の書き込み動作に長い時間をかけないようにするため, MPCD では一度に1セグメントとしていた。
 (注4) キャッシュ・メモリには書き込まれたが, まだディスク装置には書き込んでいないことを示す。
 (注5) 出力命令が出されたときの時刻をコード化したもので, CDS プロセッサが古さの制御に用いる。
 (注6) パラメタによって可変であるが, 現行は5セグメント以上のデータ長の命令はキャッシュをバイパスする。
 (注7) CDSTAR と呼ばれ, バッチ処理主体の17個のプログラムより構成されている。
 (注8) 17個のバッチ・ランのうち, 最初に“スタート”したランの開始時刻から最後に“フィン”したランの終了時刻までの時間。
 (注9) OS の下で併行に処理されるバッチ・ランの数。たとえば, “2”ならば同時に2個までのバッチ・ランの処理が可能。
 (注10) 日立, 富士通, 日電, メモレックス, アンペリフおよび IBM 3880-13 では, ストア・スルー・モードを使用, 唯一 IBM 3880-11 がポスト・ストア・モードを採用しているといわれているが, 詳細は不明である。

- (注11) ディスク・モードはレコード読み用のバッファが1レコード分しかないため、マージが2個所の場合、読みも2回となる。
- (注12) バイパス・キャッシュ・モードでは、マージのためにレコード読み用キャッシュ・スペースがドライブごとによりザーブされているので、あらかじめマージ部分のレコード・読みをすませておくことができる。
- (注13) 先読み機能とは、キャッシュ上に該当セグメントがなく、ディスク・アクセスが必要となった場合に、指定されたセグメントだけではなく、それに続くセグメントもキャッシュ上に読み込むこと。
この機能はシーケンシャル・ファイルに対して、とくに有効となる機能である。
- (注14) 転送速度は秒当たり5メガ・バイトである。

執筆者紹介 長尾 秀実 (Hidemi Nagao)

昭和46年国立釧路工業高専機械工学科卒業。同年日本ユニバック(株)入社。客先担当 CE、フィールドサポート、テクニカル・サポート担当エンジニアを経て、56年よりキャッシュ・ディスク・システムの開発、導入に従事。



永田 浩一 (Koichi Nagata)

昭和50年東京都立小石川工業高校電子科卒業。同年日本ユニバック(株)入社。客先担当 CE を経て56年より磁気ディスク開発、57年よりキャッシュ・ディスク・システムの開発、導入作業に従事。



報告 FORTRAN コンバージョンにおける MACRO の適用

Application of MACRO to FORTRAN Conversion

小林 五郎

要約 プログラム・コンバージョンの分野において、FORTRAN の場合はトランスレータを用いずにコンパイラ自体をトランスレータの代替として位置付ける方法が採られてきた。

COBOL の場合は、すでにトランスレータの開発方法は確立し、多くのトランスレータが開発されている。しかし、FORTRAN の場合、有効な開発手法は確立されていなかった。

近年 UNIVAC シリーズ 1100 の FORTRAN バージョンの多様化が今後の保守の面から問題化しており、これらを解決すべく ASCII FORTRAN 10 R 1 A 以降への統一が強く求められている。この移行を推進・進行するために、トランスレータをはじめとする移行ソフトウェアの必要性が高まり、MACRO システムを用いて短期間で保守性の高い移行ソフトウェアを開発し、T 社および地銀 3 行に適用した。

本稿では開発手段である MACRO システムの概要と、移行ソフトウェアの機能および開発方法、そして適用ユーザにおける実績と評価について記述する。

Abstract Highly maintainable, compact program converters and their related conversion aids are developed for Series 1100 FORTRAN in a very short time period by making effective use of MACRO system. They have also been practically applied to convert a number of FORTRAN codes prepared in the existing multiple versions of FORTRAN at several customers into those in the latest version, ASCII FORTRAN 10 R 1 A, and thus resolved serious problems in maintaining multiple versions of FORTRAN.

This paper outlines the development tool, MACRO, and then describes the program converters. An evaluation of their performance result is also reported.

1. はじめに

UNIVAC シリーズ 1100 プロダクトの ASCII 化の中で FORTRAN も例外ではなく、ASCII FORTRAN 10 R 1 A 以降へのバージョンの統一が強く求められている。

具体的には、

- 1) FORTRAN V からの移行
- 2) ASCII FORTRAN 8 R 1 以前からの移行

である。

しかるにこの移行を考えたとき、非互換項目を自動変換するトランスレータは存在せず、その開発手法さえ確立されていない状態であった。こうした状況の中で、客先の大量の FORTRAN プログラムの移行を推進・遂行するためには膨大なマンパワを要することからトランスレータを中心とした有効な移行ソフトウェアの開発が強く求められた。

従来の 2, 3 の開発例ではいずれも数 10 人月のマンパワを要しており、開発されたもの自体も機能面や保守面でいくつかの問題を抱えていた。

そこで、移行ソフトウェアの開発において MACRO システムを用いて

- 1) 短期間で
- 2) 保守性が高く
- 3) 機能変更が容易で
- 4) コンパクトな

という、4点を目標に ASCII FORTRAN 10 R 1 A への移行ソフトウェア群を開発した。本稿では、開発手段である MACRO システムの概要とともに、移行ソフトウェアの機能、開発手順および適用ユーザと実績・評価について記述している。

なお、本稿では FORTRAN V を FOR-V, ASCII FORTRAN を FTN と略記する。

2. FTN 移行

本章では UNIVAC シリーズ 1100 における FORTRAN プロダクトの FTN 10 R 1 A への移行の概略を説明する。

2.1 移行対象の設定

FTN 10 R 1 A への移行対象となる具体的な FORTRAN プロダクトとユーザを次のように設定した。

- 1) FOR-V からの移行 …………… T 社
- 2) FTN 8 R 1 からの移行 ……… A 行, K 行, H 行

前者は技術計算プログラムであるが、後者は事務計算プログラムで、3行とも同じ特徴をもっている。いずれも移行対象数は多く、数千から1万本である。

2.2 移行上の問題点

FTN 10 R 1 A への移行における問題点は次の3点に分類することができる。

- 1) FOR-V または FTN 8 R 1 固有の機能
- 2) 両 FORTRAN の間で意味上不整合となる機能
- 3) 構文上で互いに相異なる機能

上記3項目をさらに分類してゆくと、両者の相違点はかなりの数にのぼるが、実際の移行で問題となる点は頻度の点でかなり絞り込めることになる。

2.2.1 FOR-V からの移行

FOR-V から FTN 10 R 1 A への移行においては、その問題点の多くが内部コードの違いに起因することがわかっている。

すなわち、COBOL がその性格上バイト・マシンの発想で作られている言語であるのに比べて、FORTRAN はワード・マシンの発想で作られている言語であることから、内部コードの相違が大きな問題となる。

しかし純粋な技術計算の場合、文字データを扱う頻度が少ないので、一般的にこの相違に影響されることが少ないといえる。

FOR-V からの移行は、これまで述べた観点から以下の相違点を問題点として絞り込む

表 1 内部コードと1ワードの文字数
Table 1 Internal code and the number of characters in a word

	内部コード	1ワードの文字数
FORTRAN V	FD コード	6 文字
ASCII FORTRAN	ASCII コード	4 文字

ことができる。

- 補助変数の型の決定規則
- RETURN k/RETURN 0
- ビット処理 (FLD 関数)
- 文字定数/ホラリス定数
- R 型/L 型定数
- 実引き数の型と個数の検査
- STOP 文字列
- PAUSE 文字列, など

2.2.2 FTN 8R1 からの移行

現在の FTN の最新バージョンは 11R1 であるが、FTN 自体そのバージョン・アップの過程を機能面から次の 3 段階に分けて考えることができる。

- a. 8R1 以前
- b. 9R1~10R1A
- c. 11R1

a, b 間の大きな相違は文字変数の割り付けと引き数のチェックの有無にある。b と c との相違は仮想バンクが使用可能になったことである。

8R1 から 10R1A への移行では、文法上と意味上の相違点から頻度を考慮すると、次のように絞り込むことができる。

- 文字エリアの割り付け方法
- 補助変数の型の決定規則
- DO 文の最小繰返し数と仮定される増分値
- リスト型 WRITE 文の出方形式
- 実引き数の型と個数の検査
- 疑似関数 BITS の代入時のタイプ変換
- 文関数の型の決定規則
- 文番号のみ指定された行, など

以上の 8R1 との相違は COMPILER 文のオプション (STD=66) を指定することで解決するが、これでは本当の意味での移行とはいえず対象ユーザにおいても本格的な移行を望んだ。

2.3 移行ソフトウェアの概要

FTN 10R1A への移行ソフトウェアとしては、今まで述べてきた相違点を解決するために変換もしくはチェック情報を出力する必要があり、移行ソフトウェアとして次の三つが必要であると考えた。

- 1) FOR-V からのトランスレータ
- 2) FTN 8R1 からのトランスレータ
- 3) 引き数の型と個数のチェック・ツール

8R1 からのトランスレータは上位非互換項目の包含関係を考えると、実は 8R1 以前の FTN からのトランスレータ、11R1 へのトランスレータとしても十分適用可能である。

3. MACRO

本章では、FOR-V および FTN 8R1 から FTN 10R1A への移行ソフトウェアの開

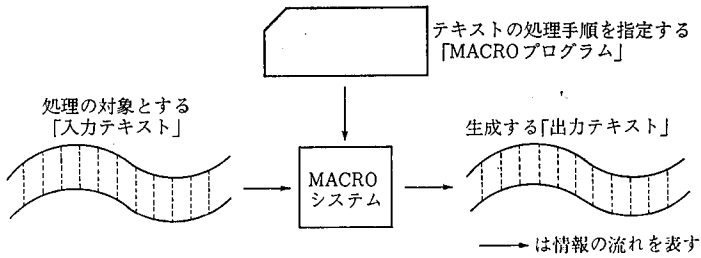


図 1 MACRO の機能概要

Fig. 1 Function outline of MACRO

発において使用した MACRO システムの特徴とその適用例などについて説明する。

3.1 MACRO の特徴

3.1.1 概 要

MACRO は UNIVAC シリーズ 1100 において

- 1) 言語の拡張・変更
- 2) テキストの生成・編集
- 3) テキストの正当性の検証

などのテキスト処理を容易に行うためのシステムとして開発された。したがって、必然的に MACRO の処理対象はシンボリック・エレメントである。

この種のソフトウェアとしては他に GSA (Generalised Syntax Analyser) が存在するが、GSA は専門家向けであり、本格的なコンパイル作成に適する。MACRO はエンド・ユーザー向けであり、効率上の問題はあっても、使い方は一般に容易なシステムであるといえる。

3.1.2. MACRO プログラム

MACRO プログラムの処理は、入力テキストを一連の文字列とみなして左から右へ走査し、プログラム中で定義した特別な文字列（以下、模様という）が出現すると、あらかじめプログラムで定められた変換規則に従って変換し、変換されたテキストを出力する。模様と合致しない入力テキストはそのまま自動的に出力される。

MACRO プログラムを書くということは、認識すべき模様とその変換規則を列挙することであり、通常の汎用言語プログラムで必要な、①入出力操作、②模様の照合、③認識できないテキストの複写操作、などはすべて自動的に行われる。したがって、プログラミングする必要がない。

MACRO プログラムの変換規則は、PL/I 風の高水準言語で記述でき可変長の文字列操作を基本としている。

このため、一般的なコンパイラの説明等で用いられる BNF 記法風で記述される模様の定義を習得し、高水準言語の知識があれば容易にプログラミングできる。

3.2 移行ソフトウェアへの MACRO の適用

本節では、移行ソフトウェアの開発において MACRO を採用した具体的な理由について説明する。

- 1) テキストの入出力・走査・解析ルーチンが不要……MACRO は自動的にテキストの入出力・走査・解析を行うので、認識したい模様の定義と変換規則の作成に集中できる。したがって、少ないステップ数でテキスト走査が確実な保守性に優れたソフトウェアの開発が期待できる。
- 2) テキスト編集用の機能が豊富……MACRO はテキスト編集向きの指示文、組込み関

配列要素名	配列要素の値
ARRAY 1 (^ABCDE^, ^ABCD^, 1)	^_XXXXXX^
ARRAY 1 (^ABCDE^, ^ABCD^, 2)	32
ARRAY 1 (^ABCDE^, ^ABCD^, 2)	27
ARRAY 1 (^QQQQQ^, ^MOT^, 1)	^_ZZZZZZ^

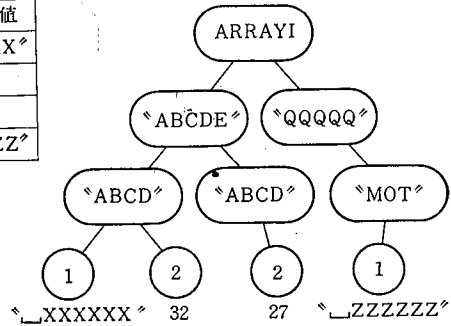


図 2 MACRO のテーブル

Fig. 2 MACRO table

数、演算などの豊富な機能をもっているため、目的とする編集・変換が容易である。

- 3) トランスレータに不可欠な内部テーブルがダイナミックに使用可能……MACROの内部テーブルは添字をキーとした木構造として作成され、COBOLのような次元や繰返し回数に制限がない。また、添字のタイプやテーブルの内容であるデータの型にも制限がなく、テーブルの作成・参照が昇順でもランダムでも可能である。
- 4) MACROプログラムは必然的に構造的……模様単位に模様の定義と変換規則を対にプログラミングし、それぞれがクローズされているため、MACROプログラムは“必然的に”構造化される。また、変換規則の記述においても構造化に適した命令や記述形式をもっているので、ここでも構造化される。
- 5) デバッグが容易……MACROプログラム実行時にエラーが発生すると、①処理中の入力テキストのライン番号、②エラーが発生したMACROプログラムのライン番号、がエラー・メッセージとともに自動的に表示されるので、非常にデバッグが容易である。また、コンパイルし直すことなく、実行時のデバッグ・オプションの指定で豊富なデバッグ情報が得られる。

しかし不都合な点もある。それは、テキストの走査・出力にカラムのチェック機能がないことである。どのカラム範囲を処理対象とするかは指定できるが、現在処理しているテキストのカラム位置を知るためにはプログラミングが必要で、案外これがむずかしい。この欠点は入力テキストをMACROに都合の良いように、あらかじめ加工しておくことで解決できる。

以上が移行ソフトウェアの開発にMACROを採用した理由である。

3.3 MACRO の適用例

前節でMACROがFORTRANの移行ソフトウェア開発手段として有効である理由を述べたが、本節では今までにMACROを適用した例をいくつか紹介する。

- 1) バロース FORTE II から AMDP インタフェースの変換ルーチンの COBOL への適用
- 2) BALCO (BAL TO COBOL トランスレータ) 前処理プログラム……16進表示のバック・エリアをパック10進表示へ変換
- 3) UNIVAC シリーズ 1100 の Meta ASSEMBLER プログラムの命令別/コーディング形式別集計ルーチンの解析部分
- 4) OS/3-J COBOL の日本語シフト・コードの 1100 LETS-J シフト・コードへの変換ルーチン等

(MACRO プログラムの例) BALCO 前処理

```

SET      DIGIT <"0".."9">;
SET      SIGN  <"A".."F">;
//
TOKEN    SPACES <" "...">;
MACRO DC  TRIGGER  PROTECT
//
      <V1:<SPACES> "DC" V2:<SPACES> "X'"
      <V3:<DIGIT...> V4:<SIGN>>... "' " SPACES S"EOL">;
//
      ANSWER V1 & 'DC' & V2;
      I1=1;
      I2=SIZE(V3);
      WHILE I1 LE I2      DO
      BEGIN
      ANSWER 'PL';
      PLEN=(LENGTH(V3(I1))+2)/2;
      ANSWER PLEN & '''';
      IF V4(I1) EQC 'B' OR V4(I1) EQC 'D' THEN
      ANSWER '-';
      ANSWER V3(I1) & '''' & S"EOL";
      IF I1 NE I2 THEN ANSWER V1 & 'DC' & V2;
      I1=I1+1;
      END;
      RETURN;
      END;

```

(MACRO プログラムの実行例)

```

@MDF,S      ASMIN.ASMPROG,MACRO.XTOP,ASMOUT.ASMPROG
MACRO INTERPRETER LEVEL BR1-1A 09/05/84 18:45:51
  1  XCONS   DC    X'1C22D333B4444C55555C'
  2          DC    X'123A234B345C456D567E678F'
  3          DC    X'FFFFFFF'
  4          DC    X'11111111'

```

いずれも実際のコンバージョンやプログラムの調査に有益であった。

例として BALCO の前処理プログラムを示す。汎用言語、たとえば COBOL で作成する場合と比較すれば、開発の容易性と保守性の高さが理解できる。

S オプションは入力テキストの表示を意味する。上記の実行例では、つぎに示すテキストが出力される。

```

XCONS   DC    PL1'1'
         DC    PL2'-22'
         DC    PL2'-333'
         DC    PL3'4444'
         DC    PL3'55555'
         DC    PL2'123'
         DC    PL2'-234'
         DC    PL2'345'
         DC    PL2'-456'
         DC    PL2'567'
         DC    PL2'678'
         DC    X'FFFFFFF'
         DC    X'11111111'

```

4. 移行ソフトウェアの開発

本章では移行ソフトウェアの開発に当たったの目標、種類と機能、開発方法および成果物について言及する。

4.1 開発目標

移行ソフトウェアの開発は、つぎに示す4点を目標に行った。

- 1) 短期間で、少ないマンパワで開発すること……開発要員が一人という現状と、ユーザの求める納入時期の関係で、早急に開発する必要があった。
- 2) 保守性が高いソフトウェアであること……保守マンパワが少なく、保守性にすぐれたものでないと、提供後の移行作業の進捗に大きく支障をきたす。
- 3) 機能変更が容易であること……一部の変換において、ユーザごとに変換方法に対するニーズが異なることが十分予想された。
- 4) 現実的なものであること……移行ソフトウェアはベーシック・ソフトウェアと異なり、すべての非互換項目を対象とせず、ユーザ・プログラムの特徴を考慮した現実的なものであるべきと考えた。

以上の4点を念頭に移行ソフトウェアを開発した。

4.2 移行するソフトウェアの名称と機能

開発する移行ソフトウェアの名称を次のように決定した。

- AFMS (FTN 8 R 1 からトランスレータ)
- FTRAN (FOR-V からのトランスレータ)
- ARGCHK (引数の個数・型のチェック・ツール)

以下に各ソフトウェアの機能概略を述べる。

4.2.1 AFMS

- 1) 変換/チェック項目……FTN 8 R 1 と FTN 10 R 1 A との上位非互換項目の中から頻度を考慮した変換/チェック項目を決定した。おもな項目を表2に示す。

表 2 AFMS 変換/チェック項目
Table 2 AFMS translation and check items

項 目	変換/チェック方法
DO 文	増分値の省略時解釈と最小繰返し回数の相違により変換する。
リスト型 WRITE 文	出力形式が異なる。メッセージを出力する。
文番号だけの行	CONTINUE 文を挿入する。
CHARACTER 文 ER 文	4文字の倍数長に変換する。
文字定数	DATA 文、型宣言文等に現れる文字定数は4文字の倍数長に変換する。
補助変数の型	8R1 と同じになるように型宣言文を挿入する。
Implied DO	増分値の省略時解釈の相違により変換する。

- 2) サマリ・リストの出力……プログラム単位に変換/チェックされた行をリストする。この際、どの種類の変換/チェックなのかを示すメッセージ番号を設定し表示することとした。

4.2.2 FTRAN

- 1) 変換/チェック項目……FOR-V と FTN 10 R 1 A との非互換項目の中からT社の特徴を考慮し、変換/チェック項目を決定した。おもな項目を表3に示す。
- 2) メッセージ番号の出力テキストへの付加……T社では、出力リストの量を削減する方針からサマリ・リストでなく、出力テキストの73カラム以降に変換/チェックの種類を示すメッセージ番号を付加するようにした。

表 3 FTRAN 変換/チェック項目
Table 3 FTRAN translation and check items

項 目	変換/チェック方法
補助変数	補助変数名が実数型のとき、INTEGER 文を挿入する。
RETURN 文	RETURN k の場合、FTN のルールに合致した k' にする。
STOP 文	STOP 文にリテラルが指定されていたら ' ' (引用符) でくくる。
PAUSE 文	STOP 文と同じ。
FLD 関数	BITS 疑似関数へ変換する。
文字定数	4 文字以内は文字以内は 4 文字長に、5 または 6 文字は 8 文字長に変換する。
R型/L型定数	メッセージを表示する。
@コメント	削除する。
副プログラム名	特定のサブルーチン名の変更 (T社 ローカル機能)。

4.2.3 ARGCHK

FTN 9 R 1 以降では副プログラムの参照時に引き数の数と型がチェックされ、一致していないと実行時にメッセージが出力される。ARGCHK はユーザ・プログラムを実行する前に上記のチェックを行うために必要と考えた。

機能は次の通りである。

- 1) 副プログラムの定義情報と参照情報を解析し、照合することによって、定義と参照に矛盾があるか否かをチェックする。
- 2) チェック結果はサマリ・リストとして出力する。
- 3) 処理単位はプログラム・ファイル内の FORTRAN シンボリック・エレメントである。
- 4) 副プログラムの定義情報と参照情報は、それぞれシンボリック・エレメントとしてセーブ可能とし、照合の再スタートを容易にできる。

4.3 開発方法

4.3.1 モジュール構成の決定

MACRO が得意とする部分以外は他言語 (この場合 COBOL と ASSEMBLER) でカバーすることとし、各移行ソフトウェアを次の構成で開始することとした。

- 1) 前処理プログラム……入力テキストを MACRO が処理しやすい形式に加工する。
- 2) 主処理プログラム……加工された FORTRAN テキストを走査・解析し、所定の変換/チェックを行う。
- 3) 後処理プログラム……出力 FORTRAN テキストの作成、サマリ・リストの出力を行う。

AFMS の処理の流れを図 3 に示す。

4.3.2 MACRO プログラムの作成手順

本項では、移行ソフトウェアの主処理プログラムである MACRO プログラムの作成手順を述べる。MACRO プログラムを作成するということは、以下に示す一連の手順を変換/チェック項目単位に追加してゆくことである。

- 1) 文字集合の定義……任意の文字の集合を SET 宣言文を用いて定義する。

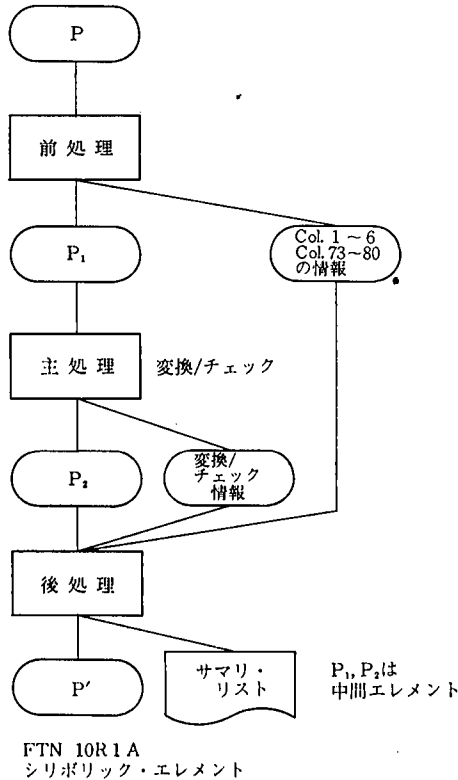


図 3 AFMS の処理の流れ

Fig. 3 Process flow of AFMS

- 2) 語彙の定義……TOKEN 宣言文を用いて語彙を定義する。語彙は入力テキストを走査する手掛りとなる文字列である。また、TOKEN 宣言文で IGNORE オプションを指定することで走査の対象外とする文字列を定義できる。
- 3) 構文マクロの作成……構文マクロは後述の引き金マクロから参照された時のみ起動するマクロであり、模様記述を簡略化できるという意味で非常に有効である。移行ソフトウェアではおもに算術式の定義に構文マクロを活用した。
- 4) 手続き/関数の登録……引き金マクロで記述する変換/チェックの規則を記述する上で共通な処理系を登録できる。
- 5) 引き金マクロの作成……引き金マクロでは変換/チェックの対象である FORTRAN 文の模様記述と変換/チェック方法を記述する。ここで示した模様記述に合致した入力テキストのみが変換/チェックされ出力される。すべての引き金マクロと一致しない入力テキストは、そのまま出力される。

MACRO プログラムの作成の過程は上記五つの繰返しであり、その意味で試行錯誤の開発を可能にしている。すなわち、ある点まで完成している MACRO プログラムに新しい処理を追加する場合、その引き金マクロを追加作成し、必要に応じて語彙や構文マクロを追加するだけでよい。

4.4 成果物：ステップ数と所有マンパワ

開発された移行ソフトウェアのステップ数および所要マンパワを表 3、表 4 に示す。

表 3 移行ソフトウェアのステップ数

Table 3 Steps of conversion aids

	AFMS	FTRAN	ARGCHK
前 処 理	73	75	49
主 処 理	1,707	785	1,558
後 処 理	419	75	1,098
ASSEMBLER ルーチン	223	201	179
実行用スケルトン	23	11	44
合 計	2,445	1,136	2,928

表 4 所要マンパワーと適用ユーザ

Table 4 Man power and applied users

	AFMS	FTRAN	ARGCHK
開 発	1.0 人月	0.7 人月	1.5 人月
保 守	1.0 人月	0.5 人月	0.5 人月
適用ユーザ	地銀 3 行	T 社	T 社および 地銀 3 行

5. 実 績

移行ソフトウェアのうち二つのトランスレータの変換実績は次のとおりである。

- 1) AFMS (地銀 3 行に適用)
 - A 行 10000 本
 - K 行 7500 本
 - H 行 3500 本
- 2) FTRAN (T 社に適用, 33 万ステップ)
地銀では、客先主体で変換作業が行われた。

6. 評価と課題

6.1 評 価

- 1) 開発/保守マンパワーが少ない……変換実績を考慮したとしても開発/保守マンパワーが非常に少なくすみ、移行ソフトウェア開発の当初の目標は十分に達成できた。
- 2) FORTRAN トランスレータ開発手法の確立……MACRO システムの採用により FORTRAN トランスレータの開発手法が確立された。今回の成果物を土台にすることで今後の開発工数は大幅に軽減できる見通しである。
- 3) ユーザ主体の移行作業の遂行促進……移行ソフトウェアの開発により、ユーザ主体の移行を遂行させることができた。やはりマニュアルのみではユーザは移行に踏みきれないと思う。この意味で移行ソフトウェアの果たした役割は非常に大きいといえる。

6.2 課 題

- 1) 効率面……当初懸念していたことであるが、移行ソフトウェアには若干の効率面での問題があった。トランスレータの方はともかく、テーブル・サーチが頻繁な

ARGCHK は入力プログラム数が多いと、とくに遅くなる傾向がみうけられた。効率面でどの程度まで改善できるか今後の課題である。

- 2) 開発/保守要員の育成……移行ソフトウェアの開発/保守のためには MACRO システムの習得と FORTRAN 文法の深い知識が必要であり、両面の知識をもった人が少ない。MACRO は文法書だけでは習得が困難なことから、良いハウ・ツー・マニュアルを整備する必要がある。

7. おわりに

今回の移行ソフトウェアの開発では、今まで 2, 3 の適用を試みたことのある MACRO を採用し、実に有効なものであることがわかった。MACRO の適用分野は、社内のコンバージョンを主体とする部署のみではなく、客先にも多く存在すると思われる。MACRO が CTS や ED と同様に日常の中での身近な「道具」として広く使用されることを望みたい。

今後は、ツール開発面で MACRO だけでなく GSA や PROLOG などの開発手段も適用し、より有効なコンバージョン・ツールの開発を目指したい。

最後になったが、T社、A行、K行およびH行への移行ソフトウェアの適用を推進していただいたシステムの方々へ感謝する。

[附録 A] ARGCHK 実行リストの例

***** * ARGUMENT CHECK LIST * *****										
<SEQ>	<ELEMENT NAME>	<LINE>	<ENTRY>	<SUB/FUNC>	<ARG-SU>	< ARGUMENT CHECK INFORMATION >				
1.	SUBA /TEST	?	12.	SERCH	SUBROUTINE	2 ()	HMT (/ I)	XV (/ R)		
2.	SUBC /TEST	?	26.	FULD	FUNCTION (/ R)	5 ()	PR (/ R) J (/ I)	TR (/ R)	1 (/ I)	I (/ I)
		?	50.	HROOT	SUBROUTINE	6 ()	AA1 (/ R) AA5 (/ R)	AA2 (/ R) HH (/ R)	AA3 (/ R)	AA4 (/ R)
3.	SUBE /TEST	?	28.	FULD	FUNCTION (/ R)	5 ()	PR (/ R) J (/ I)	TR (/ R)	2 (/ I)	I (/ I)
		?	62.	HROOT	SUBROUTINE	6 ()	AA1 (/ R) AA5 (/ R)	AA2 (/ R) HH (/ R)	AA3 (/ R)	AA4 (/ R)
4.	SUBE1 /TEST	?	18.	PSEDO	SUBROUTINE	6 ()	PP (/ R) TCH (/ R)	TT (/ R) HZC (/ I)	PR (/ R)	TR (/ R)
			20.	ENTFUN	SUBROUTINE	6	4 TCH	NZC DH2	PR	TR
			* 22.	ENTFUN	SUBROUTINE	6	3 NCH (R / I)	XZC (I / R) DH1	PR	TR
			28.	ENTFUN	SUBROUTINE	6	4 TCH	NZC DHVL	PR	TR
		?	32.	PSEDO	SUBROUTINE	6 ()	PP (/ R) TCH (/ R)	TY (/ R) HZC (/ I)	PR (/ R)	TR (/ R)
		*	34.	ENTFUN	SUBROUTINE	6 ()	1.0 (I / R) TCH	NZC DH1	PR	NTR (R / I)

* 参照ライン
 * 実引き数の数
 * 実引き数
 * 定義側の型
 * 参照側の型
 * →不一致
 * ? →未定義副プログラムの参照
 * B →一致

* ARGUMENT CHECK LIST *

83/11/28 PAGE. 2

<SEQ>	<ELEMENT NAME>	<LINE>	<ENTRY>	<SUB/FUNC>	<ARG-SU>	< ARGUMENT CHECK INFORMATION >				
		36.	ENTFUN	SUBROUTINE	6	2	NZC	PR	TR	
						TCH	DH2			
		42.	ENTFUN	SUBROUTINE	6	1	NZC	PR	TR	
						TCH	DHVL			
5.	SUBG1 / TEST	18.	EDECHA	SUBROUTINE	1		DHVL			
		20.	EDEYEN	SUBROUTINE	1		DHVL			

* UNMATCH LINE SUMMARY *

DATE 83/11/28 PAGE. 1

< SEQ >	< ELEMENT NAME >	< UNMATCH / UNDEF LINES >			
1.	SUBA/TEST	12 ?			
2.	SUBC/TEST	26 ?	50 ?		
3.	SUBE/TEST	28 ?	62 ?		
4.	SUBE1/TEST	18 ?	22 *	32 ?	34 *

DELTA T.DELTA
ELT BR1-1C 74R1-G 11/28/83 14:30:04 (0)

1.	00	*****			
2.	00	* FORTRAN PROGRAM ENTRY INFORMATION *			
3.	00	*****			
4.	00	*****			
5.	00	*****			
6.	00	*****			
7.	00	1.	SUBA/TEST (SUB)	(SUBROUTINE)	
8.	00	1.	AFLAS (SUB)	ARG-SU = 4	LINE = 4
9.	00		< 1>	R	
10.	00		< 2>	R	
11.	00		< 3>	R	
12.	00		< 4>	R	
13.	00			R	
14.	00			R	
15.	00	2.	SUBB/TEST (SUB)	(SUBROUTINE)	
16.	00	1.	BUSEIA (SUB)	ARG-SU = 3	LINE = 3
17.	00		< 1>	NCON	I
18.	00		< 2>	NHT	I
19.	00		< 3>	IUT	I
20.	00			I	
21.	00	3.	SUBC/TEST (SUB)	(SUBROUTINE)	
22.	00	1.	CHAOK (SUB)	ARG-SU = 1	LINE = 3
23.	00		< 1>	E4K	(20)
24.	00	4.	SUBD/TEST (SUB)	(SUBROUTINE)	
25.	00	1.	COMPOS (SUB)	ARG-SU = 0	LINE = 3
26.	00	5.	SUBE/TEST (SUB)	(SUBROUTINE)	
27.	00	1.	EDECHA (SUB)	ARG-SU = 1	LINE = 6
28.	00		< 1>	DHVL	R
29.	00	6.	SUBE1/TEST (SUB)	(SUBROUTINE)	
30.	00	1.	EDEYEN (SUB)	ARG-SU = 1	LINE = 6
31.	00		< 1>	DHVL	R
32.	00	7.	SUBG/TEST (SUB)	(SUBROUTINE)	
33.	00	1.	ENTFUN (SUB)	ARG-SU = 6	LINE = 6
34.	00		< 1>	NHTYP	I
35.	00		< 2>	NZC	I
36.	00		< 3>	PR	R
37.	00		< 4>	TR	R
38.	00		< 5>	TCM	R
39.	00		< 6>	DH	R
40.	00	8.	SUBG1/TEST (SUB)	(SUBROUTINE)	
41.	00	1.	ENTHA (SUB)	ARG-SU = 1	LINE = 3
42.	00		< 1>	HACT	R
43.	00				
44.	00				
45.	00				
46.	00				

END ELT. ERRORS: NONE. TIME: 0.146 SEC. IMAGE COUNT: 46

BAFMS.POST AFMS\$PF.P2,FTNOUT.CB060/
VER 1R1 74R1-J 09/10/84 10:54:01

* AFMS TRANSLATION SUMMARY *
* (ELEMENT) CB060 *

<LINE>	< PROGRAM IMAGE >	<MSG>
9.	PARAMETER PFZAN1 = 28 @ 1 カタログ	M13
10.	+ ,PFZAN2 = 16 @ カタログ MAX	M13
11.	+ ,PFZAN3 = PFZAN1*PFZAN2 @ 1 カタログ エントリー	M13
12.	+ ,PFZAN4 = PFZAN3+3 @ FCSS IOBUF	M13
13.	+ ,PFZAN5 = PFZAN2-4 @ 537 カタログ	M13
28.	INTEGER UNOPTN(5) / 5 * 0 / @ 36 * 5 PTR	M10
29.	+ ,WANS	M10
30.	+ ,WARNING(8) // , , , , 'STAT', 'US-M', 'AP(, ,)', , '3M' //	M10
34.	DEFINE H1(1) = BITS(1, 1, 18)	M17
35.	DEFINE HZ(1) = BITS(1, 19, 18)	M17
38.	DEFINE FZDATE = BITS(FZAN(1, UKMK), 1, 18)	M17
39.	DEFINE FZKOSU = BITS(FZAN(1, UKMK), 19, 18)	M17
40.	DEFINE FZAN1 = FZAN(2, UKMK)	M17
41.	DEFINE FZAN2 = FZAN(3, UKMK)	M17
42.	DEFINE FZYN1 = FZAN(WYKN*2+2, UKMK)	M17
43.	DEFINE FZYN2 = FZAN(WYKN*2+3, UKMK)	M17
44.	DEFINE FZYKOS = BITS(FZAN(15+WYKN, UKMK), 1, 18)	M17
45.	DEFINE FZMODE = BITS(FZAN(15, UKMK), 19, 18)	M17
47.	DEFINE UNOPTN(A)=BITS(UNOPTN((A+35)/36),	M17
105.	IF (.GT. PMXPTR) THEN	MD4???
106.	1\$D001= 1	MD4???
107.	ELSE	MD4???
108.	1\$D001= PMXPTR	MD4???
109.	END IF	MD4???
110.	DO 10 PTR = 1, 1\$D001	MD4???
111.	C DO 10 PTR = 1, PMXPTR	MD4???

*** END AFMS SUMMARY LIST ***

BAFMS.POST AFMS\$PF.P2,FTNOUT.HB741/
VER 1R1 74R1-J 09/10/84 10:46:33

* AFMS TRANSLATION SUMMARY *
* (ELEMENT) HB741 *

<LINE>	< PROGRAM IMAGE >	<MSG>
110.	DEFINE OYDATE(A) = BITS(A(1), 1, 18) @ エントリー	M17
111.	DEFINE DYERRC(A) = BITS(A(1), 19, 6) @ エラー (CIF)	M17
112.	DEFINE DYERRY(A) = BITS(A(1), 25, 12) @ エラー (3桁)	M17
113.	DEFINE DYPTRS(A) = BITS(A(2), 1, 9) @ エントリー (746ビット)	M17
114.	DEFINE OKKMG (A) = BITS(A(2), 10, 6) @ カタログ	M17
115.	DEFINE DYCIF(A) = BITS(A(2), 16, 21) @ CIF (746ビット)	M17
116.	DEFINE DYKMK (A) = BITS(A(3), 1, 6) @ カタ	M17
117.	DEFINE DYSHU (A) = BITS(A(3), 7, 6) @ エントリー	M17
118.	DEFINE DYKKN (A) = BITS(A(3), 13, 6) @ カタ	M17
119.	DEFINE DYKEY (A) = BITS(A(3), 19, 18) @ エントリー	M17
120.	DEFINE DYEDA (A) = BITS(A(4), 1, 7) @ エントリー	M17
121.	DEFINE DYPTRY(A) = BITS(A(4), 8, 8) @ エントリー (3桁)	M17
122.	DEFINE DYCIFY(A) = BITS(A(4), 16, 21) @ CIF (746ビット)	M17
123.	DEFINE OYRATI(A) = BITS(A(5), 1, 18) @ エントリー (7桁)	M17
124.	DEFINE DYRATZ(A) = BITS(A(5), 19, 18) @ エントリー (3桁)	M17
125.	DEFINE OYMKK (A) = BITS(A(6), 1, 18) @ エントリー	M17
126.	DEFINE OYVRAT(A) = BITS(A(6), 19, 18) @ エントリー	M17
127.	DEFINE OYSTB (A) = BITS(A(7), 1, 18) @ エントリー	M17
128.	DEFINE DYKAI (A) = BITS(A(7), 19, 18) @ エントリー	M17
129.	DEFINE DYZANI(A) = A(8) @ エントリー	M17
130.	DEFINE OYZANZ(A) = A(9) @ エントリー	M17
131.	DEFINE OYONU(A) = BITS (A(11), 1, 9) @ 56.10 TT	M17
132.	DEFINE OYTYP (A) = BITS (A(11), 13, 6) @ 56.10 TT	M17
133.	DEFINE OYERR (A) = BITS(A(11), 19, 18) @ エントリー	M17
136.	DEFINE DKDATE (B) = BITS(B(1), 1, 18) @ エントリー	M17
137.	DEFINE DKERRC (B) = BITS(B(1), 19, 6) @ エラー (CIF)	M17
138.	DEFINE DKERRK (B) = BITS(B(1), 25, 12) @ エラー (3桁)	M17
139.	DEFINE OKPTR (B) = BITS(B(2), 1, 9) @ エントリー	M17
140.	DEFINE OKKMG (B) = BITS(B(2), 10, 6) @ カタログ	M17
141.	DEFINE OKCIF (B) = BITS(B(2), 16, 21) @ CIF	M17
142.	DEFINE OKKBN (B) = BITS(B(3), 1, 4) @ エントリー	M17
143.	DEFINE DKKEY (B) = BITS(B(3), 5, 11) @ エントリー	M17
144.	DEFINE DKSDA (B) = BITS(B(3), 16, 3) @ エントリー	M17
145.	DEFINE OKSNO (B) = BITS(B(3), 19, 18) @ エントリー	M17
147.	DEFINE OKERR (B) = BITS(B(11), 19, 18) @ エントリー	M17
148.	DEFINE OKMKK (B) = BITS(B(4), 1, 6) @ カタ	M17
149.	DEFINE DKSTAN(B) = BITS(B(4), 7, 6) @ エントリー	M17
150.	DEFINE DKTAN (B) = BITS(B(7), 1, 9) @ エントリー	M17
151.	DEFINE DKSTP (B) = BITS(B(7), 10, 11) @ エントリー	M17
152.	DEFINE DKSKMK (B) = BITS(B(4), 13, 6) @ エントリー	M17
153.	DEFINE OKSRAT (B) = BITS(B(4), 19, 18) @ エントリー	M17
154.	DEFINE OKRATI (B) = BITS(B(5), 1, 18) @ エントリー	M17

```

<LINE>          < PROGRAM IMAGE >                                     <MSG>
155.    DEFINE   DKRAT2(B) = BITS(B(5),19,18) @ 'ウツリ (キレヒン)'      M17
156.    DEFINE   DKZAN1(B) = B(8) @ 'トウシツツキカ'                    M17
157.    DEFINE   DKZAN2(B) = B(9) @ 'トウシツツキカ'                    M17
158.    DEFINE   DKJKB (B) = BITS(B(4), 1,18) @ 'トウシツツキカ'        M17
159.    DEFINE   DKRSK (B) = BITS(B(4),19,18) @ 'ウツリツキカ'        M17
165.    DEFINE   DMKMT1(1) = MKAS1 (1) @ 'ウツリツキカ'                M17
166.    DEFINE   DMHST1(1) = MYOKIN(1+1) @ 'ウツリツキカ'                M17
168.    DATA     MFLBL1 / 'HBYK','MSTM','TQ' / @ 'ウツリツキカ'          M10
169.    +         + 3 # ' / / @ 'ウツリツキカ'                          M10
171.    DATA     MFLBL2 / 'HBYK','MSTM','TQ' / @ 'ウツリツキカ'          M10
172.    +         + 3 # ' / / @ 'ウツリツキカ'                          M10
174.    DATA     MASG1 / 'ASG','AF' / 3# ' / / @ 'ウツリツキカ'          M10
175.    DATA     MASG2 / 'ASG','AF' / 3# ' / / @ 'ウツリツキカ'          M10
176.    DATA     MFREE1 / 'FRE','E' / 3# ' / / @ 'ウツリツキカ'          M10
177.    DATA     MFREE2 / 'FRE','E' / 3# ' / / @ 'ウツリツキカ'          M10
491.    DO 2400  WUHMDD=HBPSDY(MYOKIN) @ 'UU DATE' @ 'ウツリツキカ'      M20
492.    +         HBPEDY(MYOKIN),1 @ 'ウツリツキカ'                      M20
498.    IF(1.GT.HBPTRX(MYOKIN)) THEN @ 'ウツリツキカ'                    M02
499.    I#0001=1 @ 'ウツリツキカ'                                          M02
500.    ELSE @ 'ウツリツキカ'                                              M02
501.    I#0001=HBPTRX(MYOKIN) @ 'ウツリツキカ'                            M02
502.    END IF @ 'ウツリツキカ'                                             M02
503.    DO 2150  I=1,I#0001,1 @ 'ウツリツキカ'                            M02
504.    C         DO 2150  I=1,HBPTRX(MYOKIN),1 @ 'ウツリツキカ'          M02
537.    IF(1.GT.HBKPN(MYOKIN)) THEN @ 'ウツリツキカ'                    M02
538.    I#0002=1 @ 'ウツリツキカ'                                          M02
539.    ELSE @ 'ウツリツキカ'                                              M02
540.    I#0002=HBKPN(MYOKIN) @ 'ウツリツキカ'                            M02
541.    END IF @ 'ウツリツキカ'                                             M02
542.    DO 2250  I=1,I#0002,1 @ 'ウツリツキカ'                            M02
543.    C         DO 2250  I=1,HBKPN(MYOKIN),1 @ 'ウツリツキカ'          M02
457.    DO 3300  WUHMDD=HBQSDY(MKAS1) @ 'ウツリツキカ'                  M20
458.    +         HBQEDY(MKAS1),1 @ 'ウツリツキカ'                      M20
462.    IF(1.GT.HBQEN(MKAS1)) THEN @ 'ウツリツキカ'                    M02
463.    I#0003=1 @ 'ウツリツキカ'                                          M02
464.    ELSE @ 'ウツリツキカ'                                              M02
465.    I#0003=HBQEN(MKAS1) @ 'ウツリツキカ'                            M02
466.    END IF @ 'ウツリツキカ'                                             M02
467.    DO 3200  I=1,I#0003,1 @ 'ウツリツキカ'                            M02
468.    C         DO 3200  I=1,HBQEN(MKAS1),1 @ 'ウツリツキカ'          M02
742.    IF(WFLNO.GT.PFLNO) THEN @ 'ウツリツキカ'                        M02
743.    I#0004=WFLNO @ 'ウツリツキカ'                                      M02
744.    ELSE @ 'ウツリツキカ'                                              M02
745.    I#0004=PFLNO @ 'ウツリツキカ'                                      M02
746.    END IF @ 'ウツリツキカ'                                             M02
747.    DO 4100  I=WFLNO,I#0004,1 @ 'ウツリツキカ'                      M02
748.    C         DO 4100  I=WFLNO,PFLNO,1 @ 'ウツリツキカ'              M02

```

*** END AFMS SUMMARY LIST ***

QAFMS.POST AFMS%PF.P2,FTNOUT.GT220/
VER IR1 74R1-J 09/10/84 10:41:00

* AFMS TRANSLATION SUMMARY *
* <ELEMENT> GT220 *

```

<LINE>          < PROGRAM IMAGE >                                     <MSG>
121.    DEFINE   UDTEN = BITS(WTRRD(1), 1,12) @ 'ウツリツキカ'          M17
122.    DEFINE   WDKBN = BITS(WTRRD(1),28, 9) @ 'ウツリツキカ'          M17
123.    DEFINE   WDKMK = BITS(WTRRD(2), 1, 6) @ 'ウツリツキカ'          M17
124.    DEFINE   WDKO22= BITS(WTRRD(2),16,21) @ 'ウツリツキカ (CO???)' M17
125.    DEFINE   WDKO21= BITS(WTRRD(3),19,18) @ 'ウツリツキカ (CO???)' M17
126.    DEFINE   UDTL = WTRRD(4) @ 'TEL-NO'                               M17
127.    DEFINE   WDKIN = WTRRD(5) @ 'ウツリツキカ'                      M17
128.    DEFINE   WDFUR2= BITS(WTRRD(4),19,18) @ 'ウツリツキカ'          M17
129.    DEFINE   WDEKBN= WTRRD(20) @ 'ウツリツキカ'                      M17
130.    DEFINE   WDERR(1) = WTRRD(20+1) @ 'ウツリツキカ'                M17
136.    DATA     + UTRFLN / 'GTR','ANS1','SA-1' / / @ TRANS            M10
137.    +         + UTRLBL / 'GTR','ANS1','SA' / 3# ' / / @ TRANS        M10
138.    +         + WMSFLN / 'GTE','LMST','RA-1' / / @ MASTR            M10
139.    +         + WMSLBL / 'GTE','LMST','RA' / 3# ' / / @ MASTR        M10
140.    +         + WUOFLN / 'GTR','ANS2','SA-0' / / @ 'ウツリツキカ'    M10
141.    +         + WUOLBL / 'GTR','ANS2','SA' / 3# ' / / @ 'ウツリツキカ' M10
142.    IF( 1.GT.WI) THEN @ 'ウツリツキカ'                                M04???
395.    I#0001= 1 @ 'ウツリツキカ'                                        M04???
396.    ELSE @ 'ウツリツキカ'                                              M04???
397.    I#0001=WI @ 'ウツリツキカ'                                          M04???
398.    END IF @ 'ウツリツキカ'                                             M04???
399.    DO 10     WJ = 1,I#0001 @ 'ウツリツキカ'                            M04???
400.    DO 10     WJ = 1,WI @ 'ウツリツキカ'                                M04???
401.    C         IF( 1.GT.UX) THEN @ 'ウツリツキカ'                        M04???
423.    I#0002= 1 @ 'ウツリツキカ'                                          M04???
424.    ELSE @ 'ウツリツキカ'                                              M04???
425.    I#0002=UX @ 'ウツリツキカ'                                          M04???
426.    END IF @ 'ウツリツキカ'                                             M04???
427.    DO 20     WL = 1,I#0002 @ 'ウツリツキカ'                            M04???
428.    DO 20     WL = 1,UX @ 'ウツリツキカ'                                M04???
429.    C         DO 20     WL = 1,UX @ 'ウツリツキカ'                        M04???

```

*** END AFMS SUMMARY LIST ***

SAFMS.POST AFMS&PF.P2.FTNOUT.TD400/
VER 1R1 74R1-J 09/10/84 10:48:26

* AFMS TRANSLATION SUMMARY *
* (ELEMENT) TD400 *

<LINE>	< PROGRAM IMAGE >		<MSG>
93.	DEFINE DUW(A) = BITS(A,21, 7)	◎ 21	M17
94.	DEFINE DMH(A) = BITS(A,28, 4)	◎ 28	M17
95.	DEFINE Q1 (A) = BITS(A, 1, 9)		M17
96.	DEFINE Q2 (A) = BITS(A,10, 9)		M17
97.	DEFINE Q3 (A) = BITS(A,19, 9)		M17
98.	DEFINE Q4 (A) = BITS(A,28, 9)		M17
99.	DEFINE H1 (A) = BITS(A, 1,18)		M17
100.	DEFINE H2 (A) = BITS(A,19,18)		M17
102.	DEFINE DSKEN(A) = A(1)	◎ タイプ フォーマット	M17
103.	DEFINE DSRK(A) = A(2)	◎ =	M17
104.	DEFINE DSZEI(A) = A(3)	◎ =	M17
105.	DEFINE OPKEN(A) = A(4)	◎ PRINT フォーマット	M17
106.	DEFINE DPRK(A) = A(5)	◎ =	M17
107.	DEFINE DPZEI(A) = A(6)	◎ =	M17
526.	INTEGER PGESU		M12
529.	IF (1.GT. PGESU) THEN		M04???
530.	I#0001= 1		M04???
531.	ELSE		M04???
532.	I#0001= PGESU		M04???
533.	END IF		M04???
534.	DO 10 I = 1,I#0001		M04???
535.	C DO 10 I = 1, PGESU		M04???

*** END AFMS SUMMARY LIST ***

- 参考文献 [1] UNIVAC シリーズ 1100, MACRO 解説書.
 [2] S.R. Greenwood, "MACRO: プログラム言語", 技報, 日本ユニバック(株), No.1, 1981.
 [3] UNIVAC シリーズ 1100, ASCII FORTRAN 解説書, 移行ガイド編.
 [4] UNIVAC シリーズ 1100, FORTRAN-V 解説書応用編.
 [5] T.N. Turba, "汎用構文解析プログラム=GSA", 技報, 日本ユニバック(株), No.1, 1981.

執筆者紹介 小林 五郎 (Goro Kobayashi)
 昭和 23 年生, 47 年日本大学理工学部卒業. 同年日本ユニバック(株)入社, 以来コンバージョン業務, ツール開発およびサポートに従事. 現在に至る.



報告 ソフトウェア自動生成の実際

Software Automatic Generation Tool PASE 1100

市丸 信子

要約 本稿で紹介する PASE 1100 は、事務処理プログラムの仕様を簡潔に定義する言語であり、その仕様記述から COBOL プログラムを自動生成するツールである。

PASE 1100 によってプログラムを設計するときは、プログラムを単機能のプロセスに分解し、そのプロセスごとに定義を進めてゆく。これは、わかりやすく簡潔にその仕様を定義するため、単機能のプロセスをファイルでつないで順次処理をするというデータ・フローの考えによる。そして、このように単機能単位に定義されたプロセスの組合せによるプログラムを自動生成する際には、各プロセスが順次処理ではなく、併行処理されるようにスケジュールし生成する。また、各プロセスは、Jackson の構造化プログラム設計技法を使って、入力データ構造から導き出された処理構造をもつものとなっている。

このように仕様記述からプログラムを自動生成することによって、膨大なプログラミング作業を取り除くと同時に仕様とプログラムの一元化を図ることができる。これは、プログラムの開発や保守における生産性と信頼性の向上に大きく貢献するものと信じる。

Abstract PASE 1100 is an automated business-oriented program derivation system for Series 1100 which consists of PASE 1100 specification language and its specification processor.

PASE 1100 specification language is designed on the basis of both data flow design method of T. de Marco and structured program design method of M. Jackson to realize comprehensible, compact and correct specifications. On the other hand, PASE 1100 specification processor automatically generates the corresponding COBOL codes as well as their program documents from the given specifications in PASE 1100 specification language, hence reducing programming efforts.

PASE 1100 therefore contributes to the promoted software productivity and reliability in business-oriented program development and maintenance.

This paper delineates automated program derivation tool PASE 1100.

1. はじめに

コンピュータ・システムの開発において、担当者が望んでいることは、ソフトウェア作りという知的作業に楽しく従事し、その結果のできばえによって満足を得たいということである。また、現在多量のバック・ログをかかえている企業は、より低コストで、正確・簡潔なシステムを迅速にかつ効率よく作る方法を模索している。

今までは、これらを解決する鍵はプログラミング工程にあるとして、それを改善すべくいろいろなプログラミング方法が生まれ、実用化されている。しかし、本当の原因はシステム開発過程のもっと初期の段階にあるということが認識され、最近では仕様定義あるいは要求定義の段階をよりの確に行うための方法論やツールに目が向けられてきた。

こうした中で、プログラムの仕様書を正確にもれなく記述し、記述された通りのプログラムを作り、また常に仕様書とプログラムとドキュメントとの整合を保全するためには、どうすれば良いかという点に的を絞って PASE 1100 を開発した。

本稿はこの PASE 1100 がどのようなものであるか、また自動生成というものをどのよ

うにして実現しているかを紹介するものである。

すなわち、PASE 1100 は、事務処理システムのプログラム仕様を人間が理解しやすい表現で、しかも正確に記述できる言語を提供するもので、仕様書をそのまま入力したら、その仕様を実現するプログラムに自動的に翻訳し、同時にそのプログラムのドキュメントを生成する実験的なツールである。

2. 仕様記述言語 PASE 1100

2.1 PASE 1100 の特徴

PASE 1100 が対象としている事務処理プログラムは、de Marco 流に書くと図 1 のようになる。これは、入力ファイルを出力ファイルに変換する処理がプログラムであるということを示している。

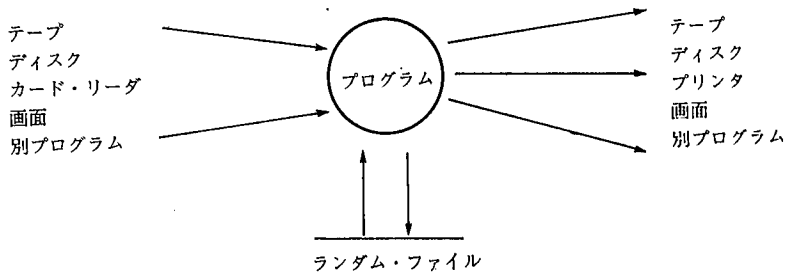


図 1 事務処理プログラム

Fig. 1 Business application program

一般的に、ファイルはレコードの有限集合体であり、プログラムはある入力データから出力データへの変換を順次に繰り返す行いを前提としている。

PASE 1100 は、このデータ変換を簡潔に定義したいというところから出発している。

データ操作、すなわち入力データから出力データへの変換は、入力データおよび出力データ間の関係を示明することによって定義する。

この関係を定義するとき PASE 1100 では、トップ・ダウンの思考法によって、概要から詳細へと段階的に円滑に進むように配慮している。データ・フロー設計と構造化プログラム設計の二つの考え方を融合して、概要から詳細化してゆく考え方が PASE 1100 の最大の特徴である。

また PASE 1100 でのプログラムの仕様書は、データを定義する部分、データ操作を定義する部分、プログラムの実現方法を指定する部分の各々を分離して記述している。

2.2 データ・フロー設計による概要定義

プログラムの仕様を定義するに当たり、まずはデータ・フローを基にした機能分解によって、入力データを出力データに変換するブラック・ボックスを連続的に適用する。このブラック・ボックスを PASE 1100 ではプロセスと呼んでいる。プログラムの働きをどのようなプロセスの組合せで実現するかを決定するこの段階が、PASE 1100 でプログラムの仕様を定義するときの重要なポイントとなる。

プログラムを設計するときを考えるべきことは、プログラムが複雑な処理を一度に行うものではなく、簡単なデータ変換を一つ一つ積み重ねることによって実現するものという

点にある。すなわち、プログラムをできるだけ単機能のプロセスに分解し、分解したプロセスはデータ・ファイルでつながれて、順次処理されるものと考えてゆけばよい。

分解されたプロセスの流れを PASE 1100 で表現するには、各プロセスごとにその出力ファイルと入力ファイルとの関係付けで行う。

入力と出力の関係、すなわち、出力ファイルから見たときの入力ファイルの関係によって、PASE 1100 では四つのプロセス型に分類している (図 2)。

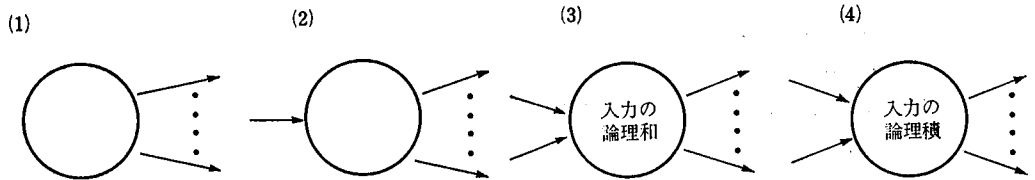


図 2 プロセスの型
Fig. 2 Process type

この段階でプロセスにおける出力ファイルと入力ファイルとのファイル間の関係が定義付けられる。すなわち、プログラムを構成するプロセスの組合せが定義される。

PASE プロセッサは、プログラムをデータの流れとして考え、各プロセスをできるだけ併行処理できるように自動的にスケジュールし、実行データの流れに再編して一つのプログラムに組み上げる。このため設計時に順次処理のデータの流れとして考えられたものが、実稼働のときは、最も効率の良い流れとして動くものとなっている。

2.3.7 構造化プログラム設計による詳細定義

プログラムの働きが入力ファイルから出力ファイルへの変換であるならば、プロセスの働きも、やはりそのプロセスにおける入力から出力への変換である。したがって、プロセスの働きを定義することは、各プロセスごとにそのプロセスの出力ファイルが入力ファイルとどのような対応関係をもっているかを記述することによってなされる。そして、プロセスの中の入力と出力との対応関係を定義するときも、段階的に詳細化してゆくことになる。

ファイル間の対応は概要定義のところですでに定義済みなので、ここではレコード間の対応関係、さらに項目間の対応関係を定義すればよい。そこで、まず出力レコードに対する入力レコードの単位を考える。

出力レコードに対する入力レコードの単位とは、基本的には次の 3 種類である。出力フ

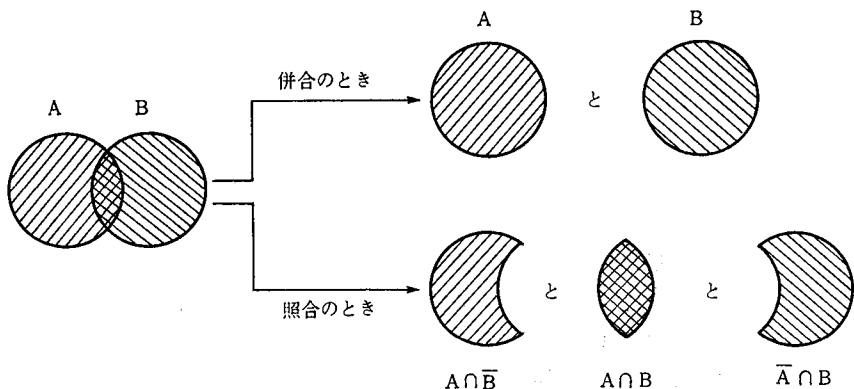


図 3 二つのファイルのレコードの組
Fig. 3 Record set in a pair of file

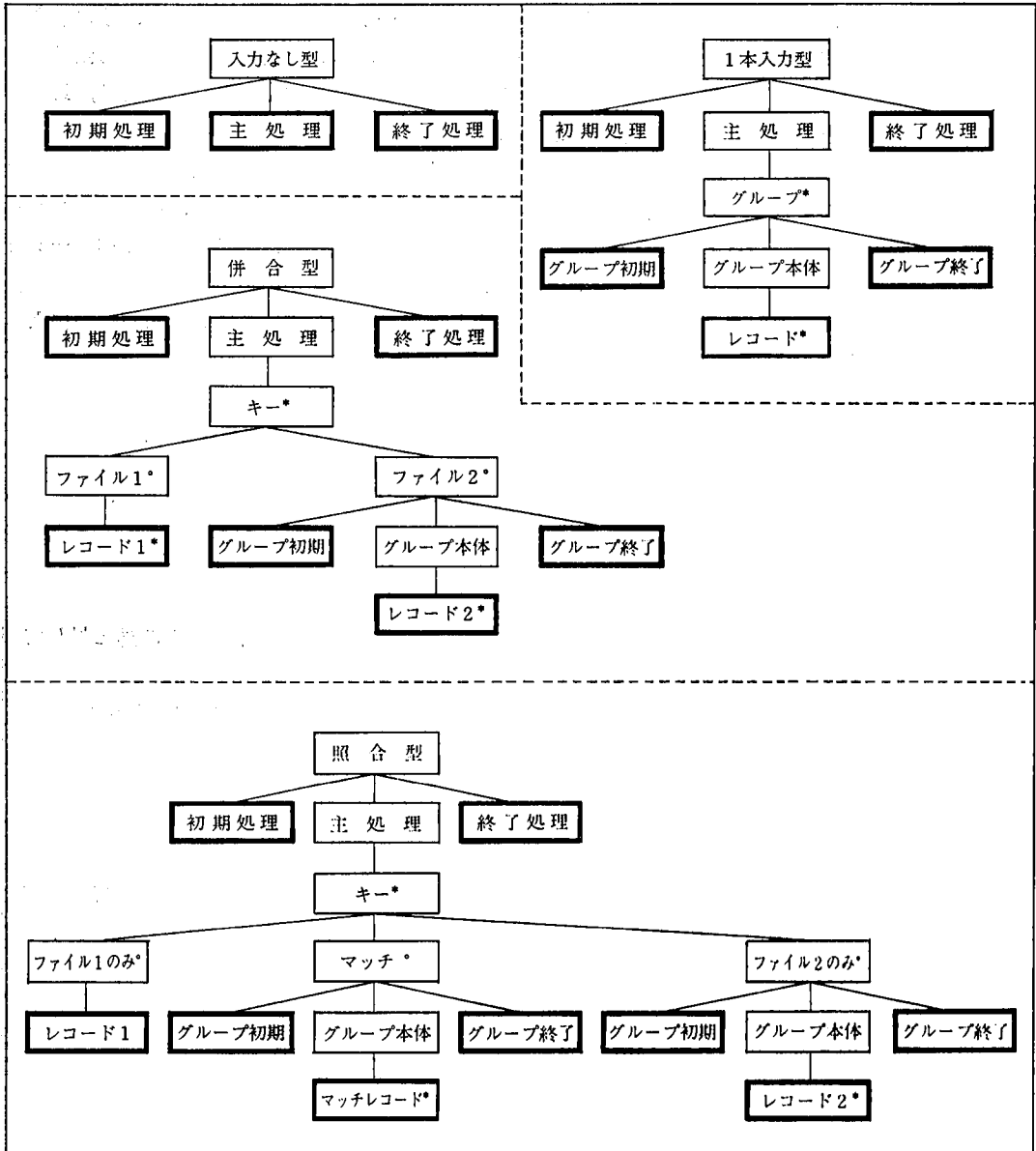


図 4 四つの型のプロセス構造
Fig. 4 Four process structures

ファイルはこれらの対応関係をもつ複数のレコード種類をもつものと考えられる。

- 1) 一つのファイルの単独レコード
- 2) 一つのファイルのレコードの集合
- 3) 二つのファイルのレコードの組

ここで、組には2種類あり、それらは二つのファイルのレコードの和を採るか、あるいは積を採るかの違いがある。さらに、組を構成する要素が単独レコードであるか、レコードの集合であるかによって類別される。

レコード間の関係を定義することで、この関係を満たす「場」が存在し、その中において出力レコードを作り出すことを宣言する。

最後に、それぞれの場の中でどのような出力レコードを作るかを定義する。すなわち出力レコードの項目が、入力項目（あるいは内部データ項目）をどのように変換した結果をもつものであるかを定義する。ここでの変換を定義するために PASE 1100 では、10種類の指示文と 15 種類の関数を用意している。

このようにして、PASE 1100 は構造化プログラム設計技法に基づき、段階的に定義された情報によって、データ構造からプログラム構造を自動的に導き出す。そして、レコード間の対応関係からプログラム構造上の妥当な位置を判定して、そこに基本オペレーションを割り振るものである。さらに、指定したオペレーションが構造上のどの位置に割り振られたかは、PASE 1100 がドキュメントとして自動作成しているプロセス構造図によって確認される。

2.4 仕様書の構成

PASE 1100 で記述する定義書は、以下に示すように七つの部で構成する。

- 1) 見出し部……プログラムの見出しであり、プログラム名称を定義する。

データの定義

- 2) データ・フロー部……プログラムの外から入力するファイルおよび外に対して出力するファイルを定義する。
- 3) データ部……プログラムが扱うすべてのデータの構成と、その属性を定義する。

データ操作の定義

- 4) プロセス・フロー部……プログラム全体を機能分解することによってできたプロセスの流れを定義するとともに、各プロセスにおける入力と出力とのファイル間の対応関係を定義する。
- 5) プロセス部……各プロセスごとに、レコード間の対応関係とそれに基づく場での基本オペレーションを定義する。

実現方法の指示

- 6) 実現指示部……おもにファイルの物理情報など、プログラムの実現方法を指示する。また、目的プログラムがサブプログラムである場合は、その指示とともに受け渡しをするパラメタ等もここで指定する。さらに、入力あるいは出力するファイルについても、COBOL の標準的な入出力命令を使わないときには、ここで当該ファイルに対する入出力命令を指定できる。

- 7) デバッグ部…… PASE 1100 は目的プログラム実行時に、入出力レコードのオクタルダンプや作業域のモニタ・リストを取ることができるよう、デバッグ用のコードを生成する機能を備えている。この機能を使用したいとき、ここにその指示を行う。

つぎに、PASE 1100 で記述したプログラム仕様書の一例を示す。

このプログラムは、MERGE 2 という名前をもち、その処理概要は次のとおりである。

「入力ファイルは TRX と OLD-M1 と OLD-M2 であり、出力ファイルは NEW-M

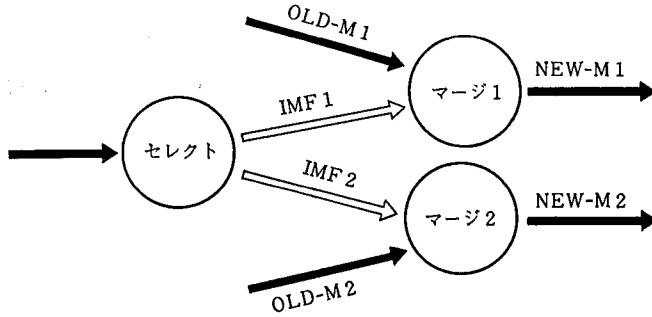


図 5 例題のプロセス・フロー図

Fig. 5 Process flow diagram of example

[PASE 1100 によるプログラム仕様書]

PASE INPUT-CARD & DIAGNOSTICS LIST DATE 850509 PAGE 1

1	/*
2	/* TRXの日付けが '09' のレコード' と OLD-M1のレコード' とを
3	/* 商品コード' をキーにしてマージし NEW-M1のレコード' を作成する。
4	/* TRXの日付けが '10' のレコード' と NEW-M2のレコード' とを
5	/* 商品コード' をキーにしてマージし NEW-M2のレコード' を作成する。
6	/* OLD-M1, OLD-M2は共に商品コード' でソート
7	/* 済みである。
8	/*
(1) 9	\$ID MERGE2
10	\$DATA-FLOW
(2) 11	INPUT OLD-M1, OLD-M2, TRX
12	OUTPUT NEW-M1, NEW-M2
13	\$DATA
14	\$FILE OLD-M1 \$RECORD OLD-M1-REC
15	02 ショウヒンコード" PIC 9(3).
16	02 ショウヒンシ"ヨウホウ PIC X(21).
17	\$FILE OLD-M2 \$RECORD OLD-M2-REC SAME-AS OLD-M1-REC
(3) 18	\$FILE NEW-M1 \$RECORD NEW-M1-REC SAME-AS OLD-M1-REC
19	\$FILE NEW-M2 \$RECORD NEW-M2-REC SAME-AS OLD-M1-REC
20	\$FILE TRX \$RECORD TRX-REC
21	02 ショウヒンコード" PIC 9(3).
22	02 ツキ PIC 99.
23	02 ショウヒンシ"ヨウホウ PIC X(21).
24	\$FILE IMF1 \$RECORD IMF1-REC SAME-AS TRX-REC
25	\$FILE IMF2 \$RECORD IMF2-REC SAME-AS TRX-REC
26	\$PROCESS-FLOW
27	
28	セレクト DEFINE IMF1 AND IMF2 ON TRX SORTED ショウヒンコード" ;
29	
(4) 30	マージ"1 DEFINE NEW-M1 ON OLD-M1 AND IMF1
31	MERGE-KEY ショウヒンコード" / ショウヒンコード" ;
32	
33	マージ"2 DEFINE NEW-M2 ON OLD-M2 AND IMF2
34	MERGE-KEY ショウヒンコード" / ショウヒンコード" ;
35	\$PROCESS セレクト /***** TRXのレコードを選択する
36	\$OUTPUT IMF1
37	\$WHERE RECORD-OF TRX
(5) 38	IF ツキ OF TRX-REC = 9 THEN EDIT IMF1-REC ENDIF
39	\$OUTPUT IMF2
40	\$WHERE RECORD-OF TRX
41	IF ツキ OF TRX-REC = 10 THEN EDIT IMF2-REC ENDIF
42	\$IMPLEMENT
43	\$FILE OLD-M1 DISC, ST, 25
(6) 44	\$FILE OLD-M2 DISC, ST, 25
45	\$FILE NEW-M1 DISC, ST, 25
46	\$FILE NEW-M2 DISC, ST, 25
47	\$FILE TRX DISC, OM, 100

1 と NEW-M2 である。

TRX の日付けが、“09” のレコードと、OLD-M1 のレコードとを商品コードをキーにしてマージし、NEW-M1 レコードを作成する。TRX の日付けが“10” のレコードと、OLD-M2 のレコードとを商品コードをキーにしてマージし、NEW-M2 のレコードを作成する。」

これをプロセスに分解し、プロセス・フロー図で表わすと図-5 のようになる。

この例のドキュメント類と生成されたプログラム・リストを次ページ以降に示す。

3. 自動生成の実現法

3.1 プログラム自動生成システムの動作

プログラム自動生成システムは、動作の面から見ると図6のように四つのフェーズから成っている。

フェーズ1では、2章で述べた方法論と記述法に従って目的プログラムに何を期待するのか、その情報化の要求を定義文で記述し PASE プロセッサに入力する。

フェーズ2と3を自動化したものが PASE プロセッサであり、フェーズ2では入力された定義文を解析し、フェーズ3ではその解析結果の情報から COBOL プログラムとドキュメント類を作成する。

フェーズ4は、目的プログラムを実行させるフェーズで、現在では手動である。

3.2 プロセスの実現

プログラムの基本となる単位はどのようなものであるか。これを明確にし、設計できたとき、はじめて自動化が可能になる。

PASE 1100 において、この基本単位はプロセスである。類似のプロセスには、同一の構造が見出せる。この類似性からプロセスを四つの型に識別し、その構造をスケルトンとして、あらかじめ用意している。

各プロセスは、構造部とオペレーション部とに分けて生成する。構造部は該当するスケルトンを枠組として拾い、それを定義情報によって部分変更して生成する。オペレーション部は、レコード間の対応関係の指示に従って、構造図上の葉の妥当な位置に指定された基本オペレーションを COBOL 言語に変換して割り振ってゆく。このとき、実行可能プログラムとするための物理情報を同時に加味しながらファイルの入出力命令等を選択する。

このように事務処理プログラムに特有なレコード処理の繰返しや終了のアルゴリズム、レコード群を判定するキーの取扱い方法、あるいはファイル間のキーの大小比較による処理シーケンス決定等は、すべて PASE 1100 が自動生成し、仕様書で定義した WHAT をプログラムの HOW への変換を実現させている。

3.3 論理ファイルの実現

プログラムを構成する基本単位をみきわめることと同様に重要なことは、この基本単位を結合する技術である。このため PASE 1100 では論理ファイルという概念を導入し、プロセスの同時併行処理（擬似コールーチン）を実現している。

プログラムを分解することによって生まれてきたファイルを論理ファイルとして実現させることは、プログラムの実行効率を維持するためには是非とも必要な事柄である。

一つのプロセス群の中のプロセスとプロセスを結ぶためのファイルは、通常のディスクやテープに作られる物理ファイルとは異なるもので、これを論理ファイルと呼んでいる。

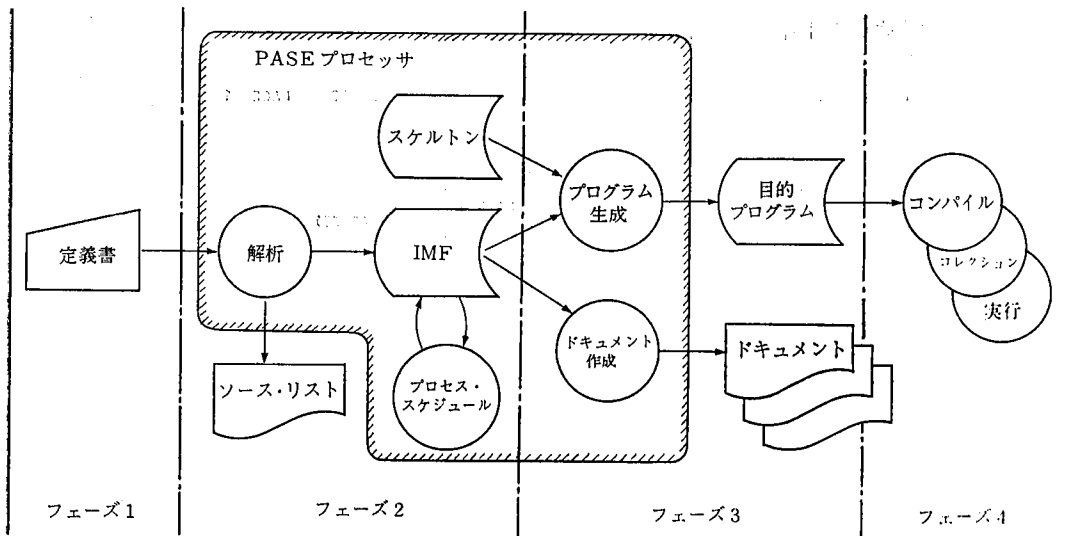


図 6 プログラム自動生成システム
Fig. 6 Program automatic generation system

〔日本語仕様書〕

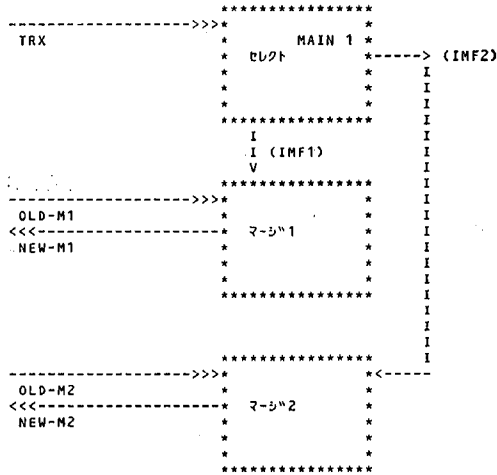
PASE SOURCE IMAGE IN JAPANESE PAGE 1 DATE 850510

```

1  /*****
2  /*  TRXの日付けが '09' のレコード' とOLD-M1のレコード' とを
3  /*  商品コード' をキーにしてマージしNEW-M1のレコード' を作成する。
4  /*  TRXの日付けが '10' のレコード' とNEW-M2のレコード' とを
5  /*  商品コード' をキーにしてマージしNEW-M2のレコード' を作成する。
6  /*  OLD-M1, OLD-M2は共に商品コード' でソート
7  /*  済みである。
8  /*****
9  プログラム名      MERGE2
10 データフロー
11     入力      旧商品マスタ1, 旧商品マスタ2, 売上ファイル
12     出力      新商品マスタ1, 新商品マスタ2
13
14 データ定義
15     ファイル定義 旧商品マスタ1 レコード定義 旧商品マスタ1のレコード"
16     02 商品コード"      PIC 9(3)。
17     02 商品情報      PIC X(21)。
18     ファイル定義 旧商品マスタ2 レコード定義 旧商品マスタ2のレコード" は 旧商品マスタ1のレコード" と同じ
19     ファイル定義 新商品マスタ1 レコード定義 新商品マスタ1のレコード" は 旧商品マスタ1のレコード" と同じ
20     ファイル定義 新商品マスタ2 レコード定義 新商品マスタ2のレコード" は 旧商品マスタ1のレコード" と同じ
21     ファイル定義 売上ファイル   レコード定義 売上ファイルのレコード"
22     02 商品コード"      PIC 9(3)。
23     02 月              PIC 99。
24     02 商品情報      PIC X(21)。
25     ファイル定義 9月マスタファイル   レコード定義 9月マスタレコード" は 売上ファイルのレコード" と同じ
26     ファイル定義 10月マスタファイル   レコード定義 10月マスタレコード" は 売上ファイルのレコード" と同じ
27
28 データフロー
29     選択      の出力は 9月マスタファイル および 10月マスタファイル  入力は 売上ファイル ソートキー
30
31     併合1     の出力は 新商品マスタ1  入力は 旧商品マスタ1 および 9月マスタファイル
32     併合キーは 商品コード" / 商品コード" ;
33
34     併合2     の出力は 新商品マスタ2  入力は 旧商品マスタ2 および 10月マスタファイル
35     併合キーは 商品コード" / 商品コード" ;
36
37 プロセス定義  選択      /***** TRXのレコードを選択する
38     出力定義   9月マスタファイル
39     【場: 売上ファイル のレコードで】
40     もし 月 OF 売上ファイルのレコード" = 9   ならば 9月マスタレコード" を作成する 』
41     出力定義   10月マスタファイル
42     【場: 売上ファイル のレコードで】
43     もし 月 OF 売上ファイルのレコード" = 10  ならば 10月マスタレコード" を作成する 』
44
45 ファイル物理定義
46     ファイル定義 旧商品マスタ1   DISC, ST, 25
47     ファイル定義 旧商品マスタ2   DISC, ST, 25
48     ファイル定義 新商品マスタ1   DISC, ST, 25
49     ファイル定義 新商品マスタ2   DISC, ST, 25
50     ファイル定義 売上ファイル   DISC, OM, 100

```

[生成されたドキュメント]



ファイル アサイン 5"303030

タイプ	ファイル	メイ	ディスク	ファイル	メイ
OLD-M1			DISC		
OLD-M2			DISC		
SD-TRX			DISC		TRX
TRX			DISC		SD-TRX
XA-TRX			DISC		XA
NEW-M1			DISC		
NEW-M2			DISC		

```

-----
I I
I トレクト I
I I
I I
-----

```

レコード
REC-OF: RECORD-OF
REC-G-OF: RECORD-GROUP-OF
REC-GI-OF: RECORD-GROUP-INIT-OF
ALL-REC-OF: ALL-RECORDS-OF

```

I
I
-----

```

二重のファイル
F: TRX

```

I I
I 7°ob入 I
I 5039イ I
I I
-----

```

```

I
I
-----

```

```

I I * I
I SWHERE I
I REC-OF F I
I I
-----

```

[生成された COBOL プログラム]

1100 ASCII COBOL SOURCE LISTING

```

1 CONTROL DIVISION.
2 ALPHABET SECTION.
3 SOURCE-ALPHABET 'A' THRU 'Z'.
4 IDENTIFICATION DIVISION.
5 PROGRAM-ID. MERGE2.
6 ENVIRONMENT DIVISION.
7 CONFIGURATION SECTION.
8 SOURCE-COMPUTER. UNIVAC-1100.
9 OBJECT-COMPUTER. UNIVAC-1100.
10 INPUT-OUTPUT SECTION.
11 FILE-CONTROL.
12 SELECT OLD-M1 ASSIGN DISC.
13 SELECT OLD-M2 ASSIGN DISC.
14 SELECT TRX ASSIGN DISC SD-TRX.
15 SELECT SD-TRX ASSIGN DISC TRX.
16 SELECT XA-TRX ASSIGN DISC XA.
17 SELECT NEW-M1 ASSIGN DISC.
18 SELECT NEW-M2 ASSIGN DISC.
19 DATA DIVISION.
20 FILE SECTION.
21 FD OLD-M1
22 LABEL RECORD STANDARD
23 BLOCK 25 RECORDS.
24 01 OLD-M1-REC.
25 02 ショウベンコート PIC 9(3).
26 02 ショウベンシヨウホウ PIC X(21).
27 FD OLD-M2
28 LABEL RECORD STANDARD
29 BLOCK 25 RECORDS.
30 01 OLD-M2-REC.
31 02 ショウベンコート PIC 9(3).
32 02 ショウベンシヨウホウ PIC X(21).
33 FD SD-TRX
34 LABEL RECORD OMITTED
35 BLOCK 100 RECORDS.
36 01 SD-TRX-REC.
37 02 SD-ショウベンコート PIC 9(3).
38 02 SD-ツキ PIC 99.
39 02 SD-ショウベンシヨウホウ PIC X(21).
40 FD TRX
41 LABEL RECORD STANDARD
42 BLOCK CONTAINS 7168 CHARACTERS.
43 01 TRX-REC.
44 02 ショウベンコート PIC 9(3).
45 02 ツキ PIC 99.
46 02 ショウベンシヨウホウ PIC X(21).
47 SD XA-TRX.
48 01 XA-TRX-REC.
49 02 XA-ショウベンコート PIC 9(3).
50 02 XA-ツキ PIC 99.
51 02 XA-ショウベンシヨウホウ PIC X(21).
52 FD NEW-M1
53 LABEL RECORD STANDARD
54 BLOCK 25 RECORDS.
55
169 88 MERGE-2 VALUE 2.
170 02 JUMP-Q PIC 9(10) COMP VALUE 1.
171 PROCEDURE DIVISION.
172 MERGE2.
173 PERFORM エレクト THRU EXIT-エレクト.
174 EXIT-MERGE2.
175 STOP RUN.
176 エレクト.
177 SORT XA-TRX
178 ON ASCENDING KEY
179 XA-ショウベンコート OF XA-TRX-REC
180 USING SD-TRX
181 GIVING TRX.
182 IF OPEN-TRX = 0
183 MOVE ' ' TO EOF-TRX
184 MOVE 0 TO IN-REC-CNT OF WK-TRX
185 OPEN INPUT TRX.
186 ADD 1 TO OPEN-TRX.
187 PERFORM マ-シ"1 THRU EXIT-マ-シ"1.
188 PERFORM マ-シ"2 THRU EXIT-マ-シ"2.
189 PERFORM RD01-TRX THRU EXIT-RD01-TRX.

```

```

190       C1-ルクト.
191       IF NOT TRX-OK GO TO C1E-ルクト.
192       PERFORM PR01-TRX THRU EXIT-PR01-TRX.
193       PERFORM WT5-ルクト THRU EXIT-WT5-ルクト.
194       PERFORM RD01-TRX THRU EXIT-RD01-TRX.
195       GO TO C1-ルクト.
196
197       C1E-ルクト.
198       SUBTRACT 1 FROM OPEN-TRX.
199       IF OPEN-TRX = 0
200           CLOSE TRX.
201       MOVE 'E' TO EOF-IMF1.
202       PERFORM 2-5"1 THRU EXIT-2-5"1.
203       MOVE 'E' TO EOF-IMF2.
204       PERFORM 2-5"2 THRU EXIT-2-5"2.
205
206       EXIT-ルクト.
207       EXIT.
208
209       RD01-TRX.
210       READ TRX
211           AT END MOVE 'E' TO EOF-TRX.
212       EXIT-RD01-TRX.
213       EXIT.
214
215       PR01-TRX.
216       ADD 1 TO IN-REC-CNT OF WK-TRX.
217       EXIT-PR01-TRX.
218       EXIT.
219
220       WT5-ルクト.
221       IF NOT (ツキ OF TRX-REC = 9)
222           GO TO W101-ルクト.
223       MOVE CORR TRX-REC TO IMF1-REC.
224       ADD 1 TO OUT-REC-CNT OF WK-IMF1.
225       PERFORM 2-5"1 THRU EXIT-2-5"1.
226
227       W101-ルクト.
228       IF NOT (ツキ OF TRX-REC = 10)
229           GO TO W102-ルクト.
230       MOVE CORR TRX-REC TO IMF2-REC.
231       ADD 1 TO OUT-REC-CNT OF WK-IMF2.
232
233
234       GO TO KE-2-5"2.
235
236       KE-2-5"2.
237       IF MFLG OF WK-2-5"2 = 1
238           MOVE KEY1-2-5"2 TO KEY-2-5"2
239       ELSE
240           MOVE KEY2-2-5"2 TO KEY-2-5"2.
241       MOVE 0 TO REC-NO OF WK-2-5"2 REC-CNT OF WK-2-5"2.
242       E-KEYSET-2-5"2.
243       EXIT.
244
245       RD03-OLD-M2.
246       READ OLD-M2
247           AT END MOVE 'E' TO EOF-OLD-M2.
248       MOVE EOF-OLD-M2 TO EOFMARK OF KEY1-2-5"2.
249       MOVE 537777777 OF OLD-M2-REC TO KEY01 OF KEY1-2-5"2.
250       EXIT-RD03-OLD-M2.
251       EXIT.
252
253       RD03-IMF2.
254       MOVE 02 TO JUMP-Q OF WK-2-5"2.
255       GO TO EXIT-2-5"2.
256
257       Q02-2-5"2.
258       MOVE EOF-IMF2 TO EOFMARK OF KEY2-2-5"2.
259       MOVE 537777777 OF IMF2-REC TO KEY01 OF KEY2-2-5"2.
260       EXIT-RD03-IMF2.
261       EXIT.
262
263       PRO3-OLD-M2.
264       ADD 1 TO IN-REC-CNT OF WK-OLD-M2.
265       EXIT-PRO3-OLD-M2.
266       EXIT.
267
268       PRO3-IMF2.
269       ADD 1 TO IN-REC-CNT OF WK-IMF2.
270       EXIT-PRO3-IMF2.
271       EXIT.
272
273       WT-2-5"2.
274       IF MFLG OF WK-2-5"2 = 1
275           MOVE CORR OLD-M2-REC TO NEW-M2-REC
276       ELSE
277           MOVE CORR IMF2-REC TO NEW-M2-REC.
278       ADD 1 TO OUT-REC-CNT OF WK-NEW-M2.
279       WRITE NEW-M2-REC.
280       EXIT-WT-2-5"2.
281       EXIT.

```

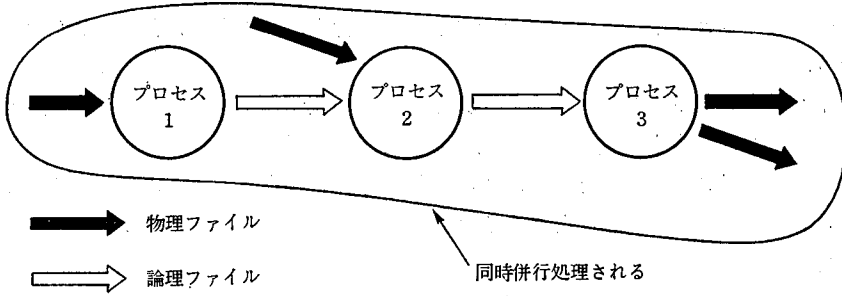



図 7 プログラムのプロセス・フロー
Fig. 7 Process flow

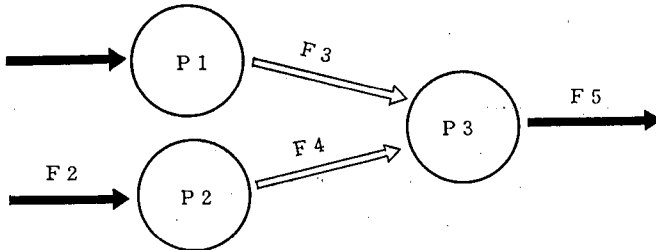


図 8 同時併行処理される三つのプロセス
Fig. 8 Parallel run process

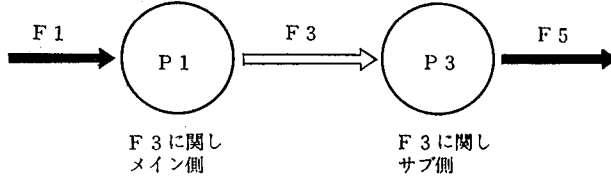


図 9 論理ファイルでつながれる二つのプロセス
Fig. 9 Two process combined with logical file F3

この論理ファイルはメモリ上にあるバッファであり、プロセスからプロセスに1レコードずつ受け渡され、その受渡しの際に入出力を伴うものではない。

論理ファイルで結ばれたプロセスは、片方はメイン側と呼び、他方をサブ側と呼ぶ。一つのプロセスは複数の論理ファイルに関してメイン側となることができるが、サブ側には1回しかなることができないという制約を考えて、ファイルとプロセスの関係を判定する。

図8の例では、次のように判定する。

- 1) F3 に関して P1 はメイン側、P3 はサブ側とする。
- 2) P3 は、F3 に関してすでにサブ側となっているため、F4 に関してはメイン側となる。その結果、P2 はサブ側となる。
- 3) 1) と 2) によって P1, P2, P3 はつながっているので、同時併行的に実行可能である。

同時併行的に実行するという事は、図8で見ると P1 が F3 を出力するとき、P3 を呼ぶ。呼ばれた P3 は、F3 を入力したと見なし F3 の1レコードを処理する。P3 と P2 の関係は、逆に P3 が F4 のレコードを入力したいとき P2 を呼ぶ。呼ばれた P2

は、F4 を1レコード作成する処理だけを行い、メイン側の P3 にコントロールを戻す。このようにレコード1件ずつプロセスが順次実行されることを、同時併行的に実行するという。

インプリメント方法は、出力側がメインで入力側がサブの場合と、出力側がサブで入力側がメインの場合とで異なるが、前者の例を図9に示す。

F3 が物理ファイルのときは、P1 の実行後 P3 が実行される。この場合と比較して、F3 が論理ファイルであるときのインプリメント方法を図10に示す。

3.4 ドキュメント作成

ドキュメントは、プログラムを保守する際に、プログラムが何をどのようにして行うものであるかを理解する上で手助けとなるものである。また、PASE 1100 使用者の要求がどのように受け取られ、どのようなプログラムに作り上げられたかを確認するためにも有効なものでなければならない。

PASE 1100 では、次のドキュメント類を作成している。

- 1) ソース・リスト……使用者が PASE 1100 に入力した情報と、それを解析したときに発見されたエラーに対するメッセージを出力したものである。
- 2) 日本語リスト……上記のソース・リストをかな漢字交じりの日本語に変換したものである。これによって仕様書はさらに読みやすくなり、仕様を理解する上で効力を発揮するものである。
- 3) プロセス・フロー図……プロセスの実行順に、プロセスが入出力ファイルによってどのようにつながっているかを図に表したものであり、これによってプロセスのスケジュール結果を確認することができる。
- 4) プロセス構造図……各プロセスごとに、そのプロセスの構造を連順・選択・繰返しの基本3構造で表したものである。これによって、使用者はプロセスのアルゴリズムを一目で読み取ることができ、指定したレコード間の対応に基づく基本オペレーションが、どこに割り振られたかを知ることができる。
- 5) 実稼動に必要なファイル情報……プログラムの実行に必要なファイルの一覧表である。基本的に物理ファイルについては使用者が認識をしているものであるが、たとえばファイルをソートするような場合、中間ファイル的な物理ファイルが必要となる。これらを含めてすべての物理ファイルを明示することで、実稼動のための必要条件を確認するためのものである。

4. 期待できる効果

これまでに述べてきた機能をもつ PASE 1100 を使用することにより、ソフトウェアの開発および保守の段階において、次のような効果が期待できる。

4.1 品質の向上

一度に複雑な仕事を考えようとすれば、当然そこに発生する誤りの率は高くなる。PASE 1100 ではプログラムを簡単な機能に分解して定義するため、一時点ではある限られた範囲内だけに注目して定義すればよい。このことは、定義書内に含まれる誤りを最小限に食い止めるのに役立つ。

また仕様の誤りを発見するためのレビューの際にも、PASE 1100 はレビュー者にわかりやすく平易な言葉で、しかも誤解を与えない標準化された言語を提供する。これにより、従来の自由言語で記述された仕様書の言葉の解釈の違いから生じる誤りを防止できる。

F 3 が物理ファイルである場合のプログラム・テキスト

```

P 1.
  OPEN INPUT F 1.
  OPEN OUTPUT F 3.
  READ F 1.
A ITR UNTIL EOF-F 1.
  WRITE F 3-RECORD.
  READ F 1.
A END
  CLOSE F 1.
  CLOSE F 3.
P 1-EXIT.
EXIT.
    
```

```

P 3
  OPEN INPUT F 3.
  OPEN OUTPUT F 5.
  READ F 3.
B ITR UNTIL EOF-F 3.
  WRITE F 5 RECORD.
  READ F 3.
B END
  CLOSE F 3.
  CLOSE F 5.
P 3-EXIT.
EXIT.
    
```

F 3. を論理ファイルとするときのプログラム・テキスト

```

P 1.
  OPEN INPUT F 1.
  PERFORM P 3 THRU P 3-EXIT.
  READ F 1.
A ITR UNTIL EOF-F 1.
  PERFORM P 3 THRU P 3-EXIT.
  READ F 1.
A END
  CLOSE F 1.
  EOF-F 3 を ON にする
  PERFORM P 3. THRU P 3-EXIT.
P 1-EXIT.
EXIT.
    
```

```

P 3.
  GO TO Q 1-P 3, Q 2-P 3, Q 3-P 3
  DEPENDING ON JUMP-Q.
  Q 1-P 3.
  OPEN OUTPUT F 5.
  MOVE 2 TO JUMP-Q.
  GO TO P 3-EXIT.
  Q 2-P 3.
B ITR UNTIL EOF-F 3.
  WRITE F 5-RECORD.
  MOVE 3 TO JUMP-Q.
  GO TO P 3-EXIT.
  Q 3-P 3.
B END
  CLOS F 5.
P 3-EXIT.
EXIT.
    
```

図 10 コーディング例 (JSP オリエンテーション・コース・テキスト^[7]の p. 21 を参考)

Fig. 10 Logical file implementation

仕様書からプログラムへの変換は PASE 1100 が自動的に行うため、これまでの仕様書 (WHAT) からプログラム (HOW) への変換の際に発生する誤解による誤りを廃除できる。

4.2 生産性の向上

プログラムの目的が似ているものでも、プログラマが違えば、設計法・構造・技術的明快さなどの点に大きな差が現れることは、多くの例で見うけられる。その結果、変更しようとするれば、そのつど異なるスタイルやアプローチを強いられることになる。これが生産性の妨げとなっている。

記述の容易さ、記述の量、思考の流れが円滑に行われるか否かなどが、仕様書を作成するときの生産性を決める要因と考えられる。

PASE 1100 では、やさしい言語設計で、英語風かな交じりで入力し、入力する量も COBOL の 1/3 から 1/5 ですむ。仕様書を作成するアプローチも、2章で述べたようにデータ・フローに基づく機能分解をしたあとは、分解された機能ごとに入出力データの対応関係を段階的に定義してゆけば、それがそのまま仕様書となる。ここで工夫の余地が残されているところは、いかなるデータ・フローにするかという点である。プログラム設計者には、このところに集中し、その経験とセンスを十分に発揮されることを望みたい。

PASE 1100 では仕様書からプログラムを自動生成するため、プログラミングの段階は、まったく不要となる。そして、その生成されたプログラムに手を加えることなく実行できるということも利点の一つである。

つぎに、プログラムのテストについて考えてみると PASE 1100 は次のような形で支援している。プログラムを生成すると同時にドキュメントを出力しているため、仕様を入力した時点でどのようなプログラムが作り出されたかが確認できる。デバッグに際しては入出力ファイルのダンプやプロセスをトレースする機能を備えており、またデータ構造から導き出されたプロセスの構造部分についてはテスト済みであるとみることができる。このためテストに要する期間を短縮するのに役立つ。

しかし、テスト・ケースの洗出し、テスト・データの作成、実行結果の検証という面で、PASE 1100 が直接支援するものはなく、この点は今後の課題として残されているところである。

4.3 保守性

システムがひとたび運用段階に入れば、それを維持することは避けられない連続的な仕事である。保守とは、ソフトウェアが正しく動かないから修正するという仕事ではない。以前には想像もつかなかった新しい要求事項が発生したために、それを満たすようにソフトウェアを変更する仕事である。

ソフトウェアの保守を容易に、正確に、かつ効率よく行うためには、仕様書とプログラムとドキュメントとの整合性が保たれていることが必要である。また、保守の対象物を標準化し変更の個所を見つけやすいものとし、その変更が他に与える影響をいかに少なくするかを考えてゆかなければならない。

PASE 1100 は仕様書からプログラムとドキュメントを生成しているため、この三者は完全に整合を保証されたものとなっている。また、仕様書を保守することが、そのままプログラムとドキュメントを保守することにつながっている。

そして、保守の対象である仕様書は、標準的な記述法によって類似性を高めている。この生産性の向上に寄与している書きやすさ、読みやすさはそのまま保守性にもつながっている。また、機能分解によってできたプロセスごとに記述しているため、機能変更に関係

するプロセスだけに注目して保守に当たればよい。また、一つのプロセスをできるだけ単機能ごとに定義し、さらに出力ファイルごとに定義しておけば、変更個所を見つけやすく他に与える影響も少なくてすむ。

4.4 開発費の削減

間接費を削減する一つの方法は、多数のデータ処理要員をかかえるかわりに、少数の質の高いプログラマやアナリストを置くことである。この点において自動生成ツールは、多くのプログラミング要員の雇用を要するような機械的なプログラミング作業にとって代わるものであり、直接費はもとより間接費を含めて開発保守費の削減に効果的である。

5. 自動生成システムの動向

ソフトウェアの開発ツールは、生産性と品質を向上させるものでなくてはならない。これを実現させるためのソフトウェア生産の自動化について、いくつかの興味あるテーマにつきに記す。

- 1) 既存のソフトウェアの再利用……データ定義の再利用は、これまでもある程度なされているが、これをデータ操作の部分にまで広げようとするものである。すなわち、プログラムあるいはそれを構成するモジュールをいくつかの型に分類し、標準パターンとして登録しておく。使用者は、その標準パターンと部品を合成してできたものに、独自の細部操作を付加してプログラムに仕上げる方法である。標準パターンの選別、部品の種類や合成の容易さが生産性を左右するものになる。既存の実践的なツールにはこの型のものが多い。
- 2) 超高級言語……これは、COBOL や FORTRAN のような高級言語よりも、より人間側に立った言語と言うことができる。PASE 1100 はこの分野に入る。
- 3) ソフトウェア CAD でコンピュータによるシステム設計支援……前者の二つはシステム設計がなされた後のプログラム設計、あるいはプログラム製作の段階に適用されるものであることに比べ、これはシステムの要求を設計する段階に適用するシステム設計自動化ツールである。すなわち、システム設計方法論とそれを実現する支援ツール群からなるもので、これらは対話型のマン・マシン・インタフェイスをもち、情報をフィードバックしながら曖昧なものを形あるものに仕上げてゆく過程において支援するものでなくてはならない。ソフトウェアの生産性や信頼性をより向上させるようにシステム開発過程のより初期の段階での自動化を考えていかななくてはならない。そして、初期であればあるほどその効果は大きいものとなる。
- 4) ワークベンチ……以上のもろもろのツール群やシステムに必要な資源を統合的に蓄積し、管理する環境支援ツールである。コンピュータ・システムの開発および維持するに当たっては共有資源の管理やプロジェクトの進捗管理、品質管理などの作業が伴うものである。これら管理情報を含めて、システム開発の全工程を一貫した思想の下で各種ツールを連動させて、はじめて本当の自動プログラミングということができようであろう。

6. おわりに

PASE 1100 を使ってソフトウェア自動生成を推進してゆくためには、統合的な管理支援ソフトウェアの下で、他のツールとの連動を実現させてゆかなければならない。また PASE 1100 言語の仕様書を作るために、ソフトウェア再利用の技術を取り入れて、仕様書の再利用を図ってゆけば、案外簡単に効果を倍増できるかもしれない。

- 参考文献 [1] 原田 実, “ソフトウェア生産性向上ツール” 日経コンピュータ 3.5 号, 1984.
[2] G.D. パークランド, “プログラム設計の考え方” 日経コンピュータ 5.17 号, 1982.
[3] “激変するソフトウェア開発スタイル” 日経コンピュータ, 9.20 号, 1982.
[4] 葉木洋一也, “シテム開発支援ソフトウェア EAGLE” 日立評論, Vol. 66, 3 月, 1984.
[5] M. A. Jackson, “構造的プログラム設計の原理”, 日本コンピュータ協会.
[6] “JSP Design Technique Description” Michael Jackson System Ltd.
[7] “JSP オリエンテーション・コース・テキスト”, JBA 刊.

執筆者紹介 市丸 信子 (Nobuko Ichimaru)

昭和 21 年生, 44 年東京理科大学工学部卒業, 45 年日本ユニバック (株) 入社 UNIVAC 418 のフィールド・サポートに従事, 50 年 (株) 日本ユニバック総合研究所入社. 52 年日本ユニバック (株) に移籍. 主に大規模なアプリケーション・システム開発に従事, 57 年より生産性向上ツールの開発を担当.



執筆注 本稿の図 10 は, 日本ビジネス・オートメーション (株) 発行, 「JSP オリエンテーション・コース・テキスト p. 21」の掲載図を, 発行者の承諾を得て一部書直したものです.

マルチリンク手順の概略と
標準化動向
An Introduction of Multilink Procedures

庭山正幸

1. はじめに

データリンク層における伝送制御手順の一つとして、マルチリンク手順の国内外での標準化作業がほぼ終了した。

マルチリンク手順は、従来のデータリンクを利用して通信回線を効率良く使用し、信頼性の高いデータ伝送を可能にする有効な手順である。

次章以降では、このマルチリンク手順の標準化作業に参加した経験をもとに、その概略と利用方法と実装の検討および標準化作業の現状について記述する。

なお、階層構造の考え方については ISO (International Organization for Standardization: 国際標準化機構) の OSI (Open Systems Interconnection: 開放型システム間相互接続) 基本参照モデルに準じている。

2. マルチリンク手順の概略

2.1 シングルリンク手順とマルチリンク手順の関係

マルチリンク手順が出現するまでの従来データリンク手順を、シングルリンク手順と呼ぶことにする。シングルリンク手順は、一つのデータリンク手順が一つの通信回線を制御し、上位層であるネットワーク層に一つの論理経路として提供される。一

方、マルチリンク手順は、一つのデータリンクが複数の通信回線を制御し、しかし上位層のネットワーク層に対しては一つの論理経路として提供できるものである。このため、マルチリンク手順ではデータリンク層を二つの副層に分け、マルチリンク手順層はその上位の副層として位置し、シングルリンク手順層はその下位の副層として位置する。マルチリンク手順層と直接インタフェースをとるネットワーク層およびシングルリンク手順層は、従来のものを何ら変更することなく利用できる。

シングルリンク手順とマルチリンク手順との関係を図1に示す。

マルチリンク手順とともに使用されるシングルリンク手順は、HDLC (Highlevel Data Link Control: ハイレベル・データリンク制御)-ABM (Asynchronous Balanced Mode: 非同期平衡モード) 手順を用いる。なお、シングルリンク手順は標準規格では符号に依存しない手順となっているが、内容からみて他の手順では不都合が生じることがある。

2.2 マルチリンク・フレームの形式

シングルリンク手順層、マルチリンク手順層およびネットワーク層で用いられるデータの形式の関係を図2に示す。

マルチリンク制御フィールドは、データ・ユニットの順序制御を行うため使用し、2オクテットからなり、図3に示す構成になっている。

マルチリンク送信順序番号 MN(S) は、12ビットで構成されたモジュール 4096 である。MN(S) は、図3に示すように MNH(S) と MNL(S) の二つのフィールドに分かれる。

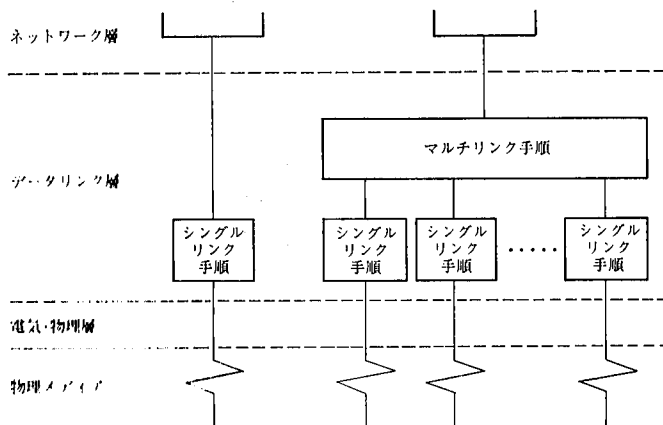
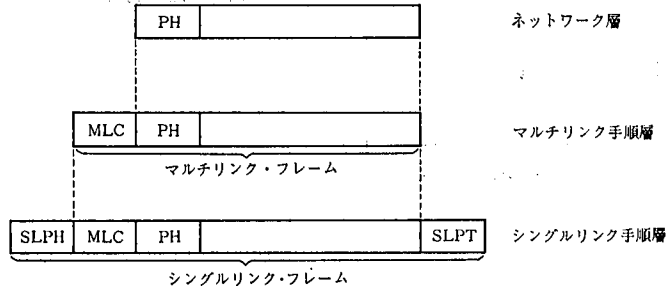


図1 シングルリンク手順とマルチリンク手順の関係
Fig. 1 Relationship to singlelink procedures



PH : パケット・ヘッダ
 MLC : マルチリンク制御フィールド
 SLPH : シングルリンク手順ヘッダ
 SLPT : シングルリンク手順トレイラ

図 2 各階層でのデータの形式
 Fig. 2 Data format relationship

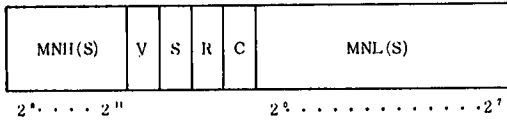


図 3 マルチリンク制御フィールドの構成
 Fig. 3 Multilink control field format

順序制御無効ビット V は、再順序制御を行うか否かを示す。再順序制御とは、相手ネットワーク層より受け取ったデータ・ユニットを、マルチリンク手順層でその送信順序番号を揃えてから自ネットワーク層へ渡すことをいう。一般的には再順序制御を行う方がよい。

順序検査オプション・ビット S は、再順序制御を行わない場合に、受信マルチリンク・フレームの重複や正当性の検査を行うか否かを示す。

MLP リセット要求ビット R は、マルチリンク・リセット手順を開始するときに使用し、MLP リセット確認ビット C は、マルチリンク・リセット手順が完了したときに使用する。

2.3 送信側と受信側の動作

2.3.1 基本動作

マルチリンク・フレームの送信と受信の基本的な動作について図 4 に記述する。

図 4 中の番号に合わせて基本的な動作を以下に説明する。

- ① ネットワーク層は、転送のためにデータ・ユニットをマルチリンク手順層に渡す。
- ② マルチリンク手順層では、ネットワーク層から受け取ったデータ・ユニットにマルチリンク手順で必要な情報を付加して (マルチリンク制御フィールドを作成して) マルチリンク・フレームを作成する。そしてマルチリンク・フレームをそれぞれのシングルリンク手順層のシング

ルデータリンク・コネクションに渡す。マルチリンク・フレームをそれぞれのシングルデータリンク・コネクションに分配する方法は、システム的设计による。

- ③ シングルリンク手順層では、マルチリンク手順層から受け取ったマルチリンク・フレームにシングルリンク手順層で必要な情報を付加してシングルリンク・フレームを作成して、回線上に送出する。回線におけるデータ転送の失敗に対する回線処理は、それぞれのシングルデータリンク・コネクションで行う。
- ④ 回線より受け取ったシングルリンク・フレームに誤りがなければ、マルチリンク手順層にマルチリンク・フレームの形式として渡す。
- ⑤ シングルリンク手順層より受け取ったマルチリンク・フレームに誤りがなければ、相手局のマルチリンク手順層で付加した送信順序番号によって順序制御を行い、ネットワーク層へシーケンシャルなデータ・ユニットとして渡す (再順序制御がある場合)。

2.3.2 伝送制御手順の特色

マルチリンク手順層は、データリンク層の副層としてシングルリンク手順層の上位に位置し、またネットワーク層の下位に位置している。このためマルチリンク手順層での伝送制御手順は、極力簡単なものを目指している。マルチリンク手順では、以下に記述するような他の伝送制御手順とは異なった特色がある。

- 1) マルチリンク・フレームの種類……2.2 節で述べたようにマルチリンク・フレームは 1 種類しかない。機能の面からそれを分割すると、データ・ユニット転送用のマルチリンク・フレームとリセット手順用のマルチリンク・フレーム

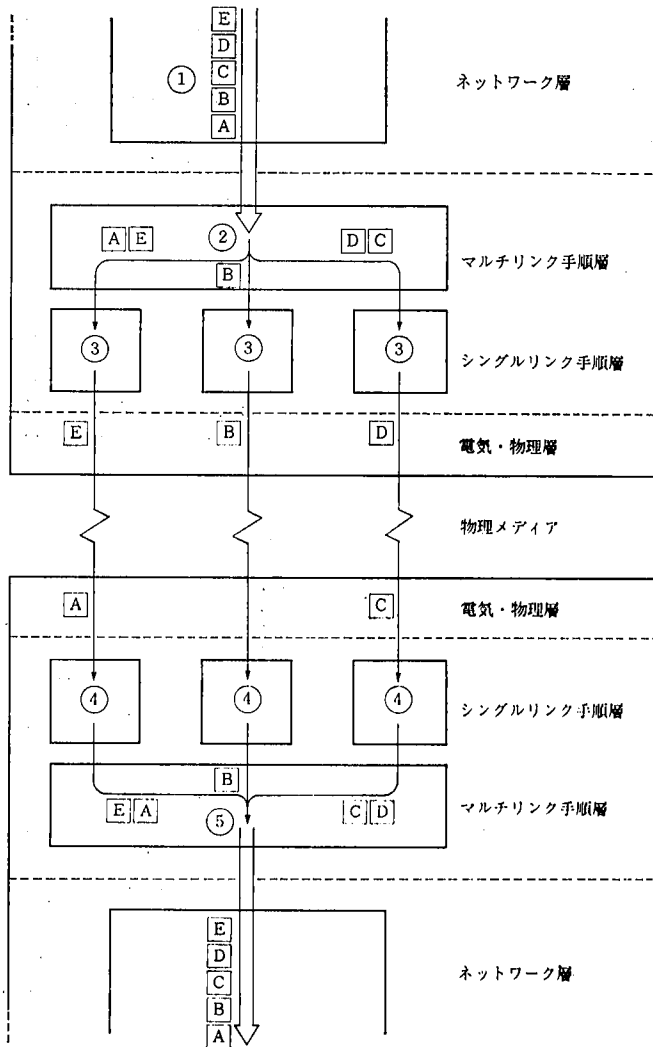


図 4 マルチリンク手順を使用したデータ伝送の基本動作
Fig. 4 Basic operations of multilink procedures

になる。

- 2) マルチリンク・フレームの送達確認……2.2 節で示したようにマルチリンク・フレームには、受信順序番号が付与されない。このため、マルチリンク・フレームの送達確認は、シングルデータリンク・コネクションでの送達確認をもって代行させる。また、転送成功率を上げるために、同じマルチリンク・フレームを複数のシングルデータリンク・コネクションに割り当ててもよい。この場合は、最初のシングルデータリンク・コネクションの送達確認か、あるいは最終のシングルデータリンク・コネクションの送達確認のどちらかをマルチリンク・フレームの送達確認として扱う。
- 3) マルチリンク・フレームの再送……マルチリ

ンク手順では複雑な手順を避けるために、マルチリンク・フレームの再送の機能は備えていない。このことは、シングルリンク手順層およびネットワーク層に再送の機能を委ねているためである。マルチリンク・フレームの再送の機能を備えていない点を補うために、つぎに述べるウィンドウ制御を行っている。

- 4) ウィンドウ制御……マルチリンク手順におけるウィンドウ制御では、送信側に加えて受信側でもウィンドウを管理している。これは、マルチリンク・フレームの紛失の可能性を考慮しているためである。一つの伝送方向において、送信側と受信側で同一のウィンドウ・サイズを用いる。
送信側では、最旧未確認の送信順序番号から

ウインドウ・サイズの範囲でのマルチリンク・フレームを送信できる。シングルデータリンク・コネクションからの送達確認によって最期末確認の送信順序番号が進めば、またさらに送信できる。

受信側で再順序制御を行っている場合、順序制御が完了したマルチリンク・フレームのデータ・ユニットはネットワーク層へ渡される。順序制御が完了していないマルチリンク・フレームは、完了するまでマルチリンク手順層で保持される。保持されているマルチリンク・フレームは、受信側での受信状態変数からウインドウ・サイズの範囲に入っている。何らかの原因でマルチリンク・フレームが紛失した場合、受信側でウインドウ・サイズを越えたマルチリンク・フレームを受信することがある。このとき、受信側ではマルチリンク・フレームの紛失が発生したことをネットワーク層に通知し、ウインドウを進めて受信動作を続行する。ネットワーク層では、必要に応じてリセット手順を行う方がよい。

- 5) リセット手順……マルチリンク手順層の相互の局の間でマルチリンク送信順序番号の同期をとるために、マルチリンク・リセット手順が提供される。同期をとる方法は、2.2 節で示した MLP リセット要求ビット R と MLP リセット確認ビット C をセットしたマルチリンク・フレームを相互に交換する。

3. マルチリンク手順の長所と適用例

マルチリンク手順では、ネットワーク層からみてデータリンクは従来と同じく 1 対 1 に対応しているが、通信回線とシングルデータリンク・コネクションは複数使用できるため、次のようなサービスが提供できる。

- 1) 低い伝送容量の通信回線を複数使用すること

により、あたかも高い伝送容量の通信回線を使用するのと同様の効果が得られる。

- 2) トラフィック量の増減に応じて、必要なだけのシングルデータリンク・コネクション数の追加または削除を行えばよい。また各シングルデータリンク・コネクションへの負荷も分散でき、過負荷のないデータ伝送が実現できる。
- 3) 障害により、シングルデータリンク・コネクションが切断されたとき、他のシングルデータリンク・コネクションを使用してデータ伝送が行えるので、障害による影響を小さく抑えられ、データ伝送の信頼性が向上する。

3.1 データ量の増減に対応した適切な回線使用方法

マルチリンク手順は、ユーザのデータ量の増減に対応した通信回線の効果的な使用方法を提供している。

以下その使用例を記述する。

- 1) 補助回線を使用する場合……これは、データ量の増加に従って補助回線をデータ伝送路に加えてゆく考え方である。補助回線としては交換回線（電話回線または DDX 回線交換）を用いると、データ量が少ない時には他への利用手段が考えやすいであろう。

図 5 の点線は、データ量の少ない時の利用法について示している。電話回線の場合には通常の通話に、DDX 回線交換の場合には他のコンピュータまたは端末との通信に使用すればよい。マルチリンク手順を用いることにより、補助回線を使用する際に、コンピュータ内のネットワークおよびアプリケーションは何ら変更する必要がなく、単に回線接続の切換えの操作だけでよい。

- 2) 適切な伝送容量を考える場合……シングルリンク手順では適用できる通信回線は、唯一のため、データ量が増加すればそれに見合った伝送

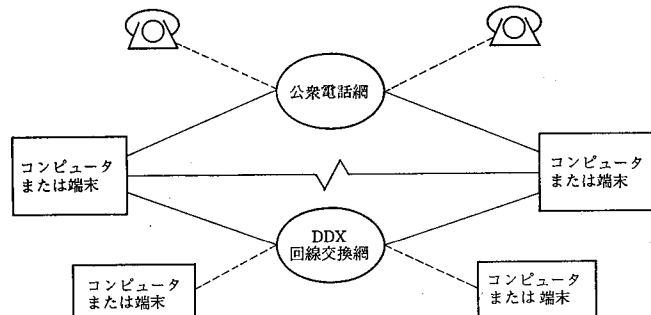


図 5 補助回線を適用した例

Fig. 5 Application of public data network

容量の通信回線を使用しなければならない。たとえば、9600 bps の通信回線では伝送容量が不足してきた場合には、48000 bps の通信回線を使用して満足させなければならない。しかし、9600 bps の通信回線では不足するが、48000 bps では十分に余裕が出てしまうことがある。この場合（コンピュータ内のオーバーヘッド等を考えないで、単に数式計算だけで）マルチリンク手順を用いて 9600 bps に 2400 bps の通信回線を追加して使用すれば、適切な伝送容量を得られることになる。

また、TDM（時分割多重方式）を用いてデータ量の最大値に見合った伝送容量の通信回線を使用し、データ量の少ないときに、電話、ファクシミリ等を併用する方法もある。

図 6 の実線は、データ量の多い時の使用方法で、点線は通信回線の伝達容量に余裕ができたときの使用方法である。この場合も、マルチリンク手順を用いればネットワークおよびアプリケーションに何ら変更することなく回線を他の目的にも利用できる。

3.2 回線障害時に対応した適切な回線使用法

シングルリンク手順では唯一の通信回線を使用しているため、その通信回線に障害が発生した場合、データ伝送がまったく不可能になってしまう。現在のところ、通信回線の障害に対処する方法としては、通常使用している通信回線の他にバックアップ用の回線を用意するしかない。このバックアップ用の回線を用いる方法では、回線コストが高くなり、またネットワークおよびアプリケーションにもその対応が必要になる。

マルチリンク手順を用いて通常時に複数の通信回線で分散してデータ伝送を行って行けば、一部の通信回線の障害ではシステムにその影響を与えずにデータ伝送が可能である。またバックアップ用の通信回線に交換回線を流用すれば、ネットワークおよびアプリケーションにはまったく関与せず、そして回線コストもわずかですむ。

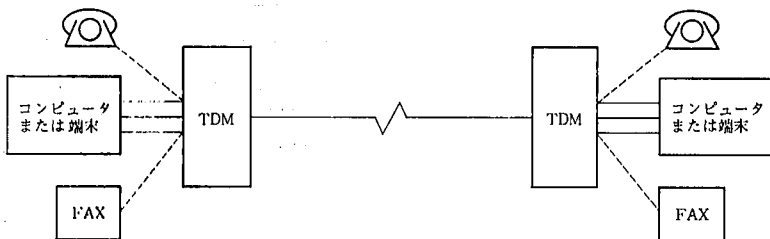


図 6 TDM を使用した例
Fig. 6 Application of TDM

4. CCR におけるマルチリンク手順実装の検討

4.1 方針

現在、NUKCMS において他接標準 CCR の一つとして ABM 手順で特定通信回線を用いて接続する DP-M1 ABM-CCR というのがある。この DP-M1 ABM-CCR をベースとしてマルチリンク手順を実装する方法について検討してみる。

先に述べたように、マルチリンク手順ではデータリンク層に副層という概念が必要であるが、今回の検討に際しては、既存のネットワーク構成およびデータリンク層 CCR について根本的に考え直すのではなく、極力既存のものを流用して最小工程数で実装する方法を試みる。また現行の DP-M1 ABM-CCR のユーザ・インタフェースは、なるべく変更しない方向で検討する。

4.2 ネットワーク・テーブル構成と CCR の対応

- 1) 現行の DP-M1 ABM-CCR のネットワーク・テーブル構成、各階層 CCR および関連するコモンルーチンの関係を図 7 に示す。
- 2) マルチリンク手順実装の DP-M1 ABM-CCR のネットワーク・テーブル構成、各階層 CCR および関連するコモンルーチンの関係を図 8 に示す。

図 8 では、三つのシングルデータリンク・コネクションを一つのマルチリンク手順で構成した例について記述している。MLP TABLE は、マルチリンク手順層で使用されるネットワーク・テーブルに該当するもので、一つのマルチリンク手順に一つ対応し、DLC 層 CCR 内に位置する。MLP TABLE の内容としては、以下の項目が含まれる。

- シングルデータリンク・コネクション(LTT)のグループ構成
- 各シングルデータリンク・コネクションのステータス、出力キュー情報
- マルチリンク手順の各種状態変数
- 順序制御が完了していない受信マルチリンク・フレームのキュー

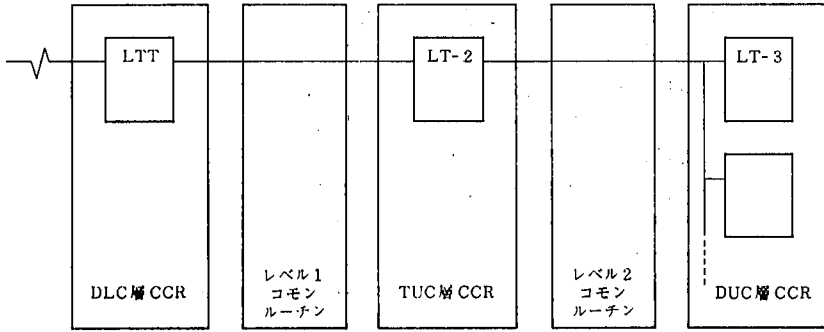


図 7 現行の DP-M1 ABM-CCR の構成
Fig. 7 Constitution of DP-M1 ABM-CCR

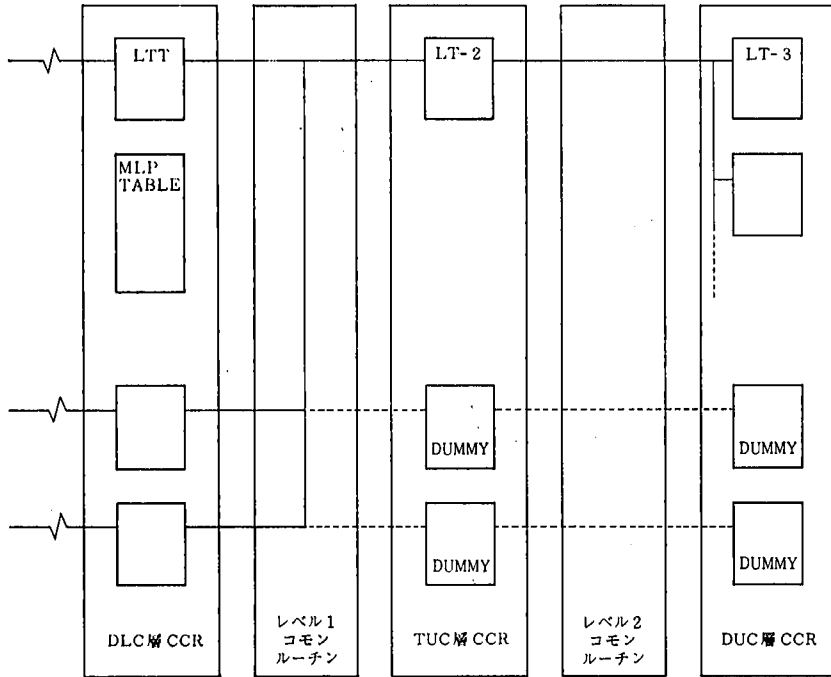


図 8 マルチリンク手順実装の DP-M1 ABM-CCR の構成
Fig. 8 Constitution of DP-M1 ABM-CCR implemented multilink procedures

● 送達確認待ちの送信マルチリンク・フレームのキュー

図8でネットワーク・テーブルを結ぶ線のうち、実線はマルチリンク手順で関係付けられるものであり、点線はコンフィグ上では関係付けられているが、マルチリンク手順運用の場合には関係がなくなること示す。現行のNUKCMSのネットワーク構成上で必要となるだけなのでDUMMYと示したテーブルが存在する。

4.3 修正箇所とその内容

図8で示したように修正を必要とする箇所は、DLC層 CCR とレベル1コモンルーチンだけですむ。

TUC 層以上の階層については、マルチリンク手順の長所で述べたように、まったく現行のものとは変わらない。このため、DDX パケット網と接続する DP-M1 PKT-CCR に対しても同様な修正でマルチリンク手順の実装が可能となる。以下に修正箇所と内容を記述する。

- 1) DLC 層 CCR……DLC 層 CCR に対する修正はほとんど不要である。必要なのは MLP TABLE の領域を確保すること、その MLP TABLE に含まれるシングルデータリンク・コネクションのグループ化および各シングルデータリンク・コネクションのステータスの処理だけである。
- 2) レベル1コモンルーチン……レベル1コモン

ルーチンに対する修正は、マルチリンク手順対応とネットワーク・テーブルの関係付けと大きく二つに分かれる。

マルチリンク手順対応としては、マルチリンク・フレームの作成、マルチリンク状態変数の更新、出力フレームのキューイング処理、送達確認処理、受信マルチリンク・フレームの順序制御、リセット手順処理がある。

ネットワーク・テーブルの関係付けとしては、本来のネットワーク・テーブル構成とは異なった関係付け(図8中の実線と点線の関係)の処理が必要である。

4.4 交換回線対応

前章で記述したようにマルチリンク手順を適用する場合、交換回線を使用することは回線コスト等の面から見て必要なことである。現行の DP-M1 ABM-CCR は、この交換回線の対応がなされていない。現在、BASIC 手順の CCR (UTS4000-CCR, U-BSC D-CCR) において交換回線対応(電話網, DDX 交換網)がなされているが、それと同等の機能を現実するためには、コモンルーチンの作成を含めてかなりの作業量が必要となる。

しかし、電話網または DDX 回線交換の V インタフェースを用いて手動の発着信操作を行えば、上記の交換回線対応の機能がなくてもよい。また切断/復旧についても手動になるが、運用面でカバーできると思える。確かに、DDX 回線交換の X インタフェースが使用できれば伝送容量の大きい回線が使用でき自動発着信操作ができるため有効ではあるが、交換回線を補助回線またはバックアップ用の回線として使用する場合に手動操作の電話網か DDX 回線交換の V インタフェースで十分使用に耐えられると思う。

4.5 検討結果

既存の DP-M1 ABM-CCR を用いて最小工程数でマルチリンク手順の実装方法について検討してみた。その結果、階層構造ではかなり無理があるが、マルチリンク手順の長所を生かせる実装方法が予想していたより簡単な方法で可能なことが判った。また、今回検討した方法では TUC 層以上の階層は何ら変更がなく、その部分のユーザ・インタフェースはまったく変わらない。そのために、DP-M1 DKT-CCR にも同様の実装方法で対応できることが判った。マルチリンク手順の副層は、レベル1 コモンルーチンの中にほとんど吸収したため、ユーザには見えない部分となっている。DLC 層 CCR についても既存のインタフェースはまったく変更なく、ユー

ザはマルチリンク手順に含まれるシングルデータリンク・コネクションのグループ構成だけを把握していればよいだけである。交換回線の対応については、マルチリンク手順の長所を生かせるだけならば、電話網か DDX 回線交換の V インタフェースを用いて手動操作を行えば条件は満たされると思う。

しかし、今回の検討は既存のソフトウェア構造をそのまま流用してマルチリンク手順を追加する方法であったため、マルチリンク手順の長所を生かせると思うが、本格的に使用した場合どのような影響が発生するか不明である。交換回線対応にしても、自動発着信機能、さらには DDX 回線交換の X インタフェースが可能になって本格的な使用が可能になると思う。

5. マルチリンク手順の標準化作業と勧告

ISO では、1982年に DIS 7478 (Draft International Standard: 国際規格案) が作成され、またそのうちリセット手順については、その後審議が進められ、1984年3月のストックホルム会議で合意が得られて DAD (Draft Addendum: 国際規格修正案) が作成された。現段階で内容的には終結したものと見られる。なお、IS (International Standard: 国際規格) の出版は1985年の予定である。

CCITT (International Telegraph and Telephone Consultative Committee: 国際電信電話諮問委員会) では、1980年版 X.75 にマルチリンク手順がはじめて盛り込まれ、1984年版の X.25 および X.75 では ISO 標準と同等であると思われる。

JIS 規格では、1983年10月から原案作成の作業に入り、1984年3月に終了した。内容は ISO 標準と同等である。なお、JIS 規格の出版は1985年度の予定である。

日本電信電話(株)の DCNA (Data Communication Network Architecture: データ通信網アーキテクチャ) では、第2版(1983年版)のデータリンク・プロトコルにマルチリンク手順が初めて盛り込まれた。内容的には、ISO の DIS と同等であるが、リセット手順に関する ISO の DAD の内容は入っていない。

6. おわりに

データの通信が、これほどまでに世の中に広く深く浸透している現在、データ伝送の信頼性の向上が当然ながら強く要求されている。この意味でマルチリンク手順は非常に有効な手順であり、とくに基幹

となるデータ通信においては必須のものとなるのであろう。

すでにいくつかの国内のメーカーにおいて、実験段階であるマルチリンク手順の試行を行っているという話を聞いている。また、DDX パケット交換においても 84 年版 X. 25 対応で、マルチリンク手順の導入を検討するということである。

マルチリンク手順を実現するためには、現行のデータリンク層に副層という考え方を採り入れなければならないし、また交換回線を利用できるシングルリンク手順も用意しなければならない。そのため、実際にマルチリンク手順を動作させるのは容易ではないが、早急に行う必要があると思う。

参考文献

- [1] JIS C 6372 マルチリンク手順。(未刊)
- [2] JIS C 6371 開放型システム間相互接続の基本参照モデル。(未刊)
- [3] JIS C 6363 ハイレベルデータリンク制御手順のフレーム構成。
- [4] JIS C 6364 ハイレベルデータリンク制御手順の手順要素。
- [5] JIS C 6365 ハイレベルデータリンク制御手順の手順クラス。
- [6] ISO/DIS 7478.2 Data Communication. Multilink Procedures.
- [7] ISO/DIS 7478/DAD 1 Revised Multilink Resetting Procedures.
(ネットワーク・ソフトウェア部)

実施理論の研究について

Implementation Theory

佐 口 功

1. はじめに

筆者が実施問題というものを考えるようになったのは、次のようなきっかけからである。

1979 年から 1980 年にかけて健診検査システムの研究を財団法人医療情報システム開発センター (MEDICAL Information System-Development Center: MEDIS-DC) から依頼された。テーマは「健康医学へのコンピュータ適用の可能性についての調査研究」であった。

成人病検診などでは、コンピュータ化はかなり進んでいるようであったが、健康情報全国ネットワークを作る上でいくつかの問題が挙げられた。その中で最も基本的な次のような問題があった。それは、

いくら素晴らしいシステムを構築しても人々が健診を受けに来なければ何の意味もない、ということである。これは、システム屋の考えることではないほど基本的な問題とも思えた。しかし、使われないシステムをいくら設計しても意味がない。そこで、どうすればこのシステムが浸透定着するのかを、システム設計の中で考える必要があり、これを実施問題の一例として実施理論を研究することにした。また、OR 学会の実施理論研究部会にも参加し、今日に至っている。

2. 実施理論

実施理論 (Implementation Theory) というのは、事務改善を行うに当たり、実施化サイクル (問題設定→システム解析→システム設計→システム実現→システム革新) において (図 1 参照)、システムを組織へ導入し定着させるために行うべきことを、行動科学、組織科学、社会科学などの立場から解明しようとする理論である。

この理論の研究は OR (Operations Research) 活動を中心に 1950 年代から散発的に問題提起され始め、その後 20 年程して 1975 年には OR/MS (Management Science) Implementation Conference の第 1 回目が米国で開催されて、MIS (Management Information System) 活動や DSS (Decision Support System) 活動にも広がっていった。

当初、実施問題は OR/MS を担当しているグループがそのキー・マンの移籍と共に消滅していく傾向に注目して問題提起がなされた。OR/MS モデルが経営システムの意思決定にさほど役立っていなかったかといえれば必ずしもそのようなことはなく、消滅の原因はもっと別な観点から研究していく必要があった。一方 MIS を中心とする情報システムにもよく似た現象が現われており、この方も含めて研究していくことになる。

3. 研究のアプローチ

われわれが経験しているシステムのニーズ把握法やシステム設計法は、参加的設計法 (Joint Approach) による相互理解 (Mutual Understanding) を狙ったものが最近の傾向であるが、さらに実施の研究では、個人の態度、パーソナリティ、プロジェクトの要因抽出、EDP (Electronic Data Processing) グループの成長過程、分権化・集権化などを考えるべきということが意識されてきている。その中からいくつかの研究事例を紹介してみたい。

一つは、Lewin-Schein Model といわれ、別名

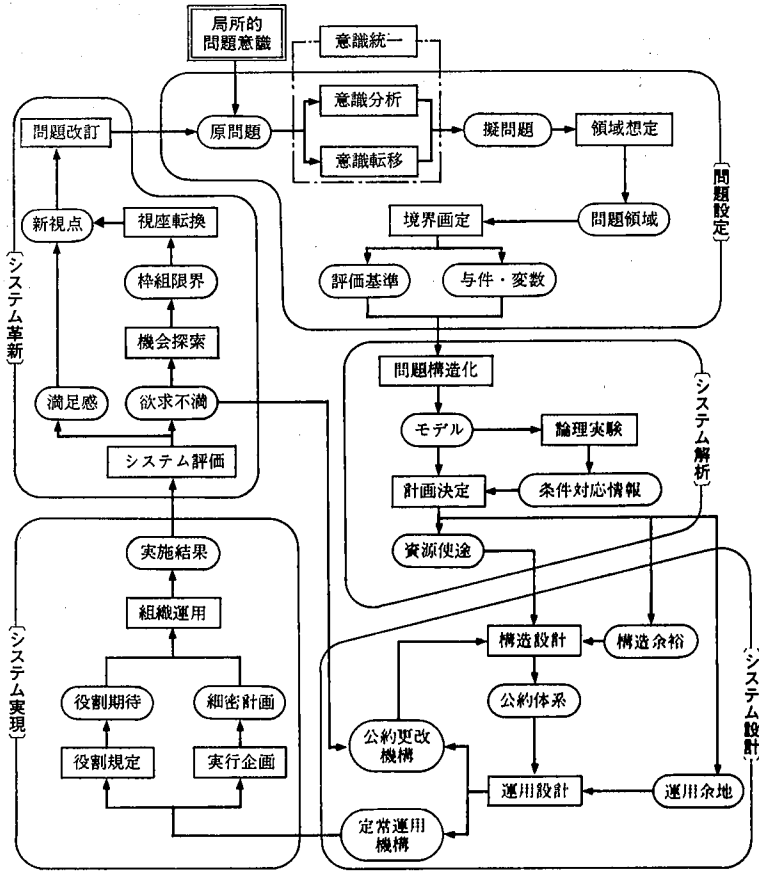


図1 実施化サイクル
Fig. 1 Implementation process cycle

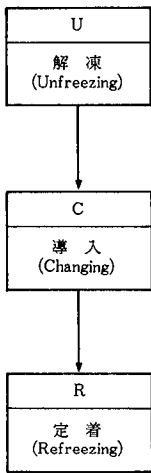


図2 UCR モデル
Fig. 2 UCR model

UCRモデルと呼ばれている実施過程である(図2)。Uは Unfreezing の Uで、変革準備といわれる段階である。たとえば、今まである一つの考えとか態度とかを持っていた人々に対して、まずその意識や

態度をやわらげるというようなことをする。つまり冷凍されていたものを解冻するというので、今まである価値体系下の意識なり態度にもとづいて行われていた仕事のやり方を解冻させて、新しい変化を受け入れられるように準備するということである。この準備が行われていないと変革導入に失敗する可能性があるので大切な段階である。次のCは Changing の Cで変革導入の段階である。この導入については、いわゆる動機付け (motivating) が大切で、別の考えに乗ろうという気にさせることが大事である。Rは Refreezing の Rで変革定着の段階で、一旦解冻して変えた後、元に戻らないようにもう一度冷凍して固めるということである。この定着がしっかりしていないと効果があがらず、思いがけない事態になったりするので、人々の意識や態度をもう一度キッチリ固める方策が必要になるというわけである。

これは Ford 社で車をどのようにして売るかを考えたときに出てきた方法といわれている。西部開拓時代に、人または物の移動には馬を使うことが常識

であったが、馬以外にも移動の手段があることをわからせ、自動車の存在を認めさせる。つぎに自動車は馬車に比べて速いし、疲れないし、などいろいろと利点を並べて自動車の優位性を認めさせ、自動車を使う気にさせる。最後に、やっぱり馬の方が馴れていて良いなどといって逆戻りしないようにする。

業務改善をすすめるときも、このようにして進めれば、浸透・定着するのではないかという発想である。この考え方は実施理論を理解する上でわかりやすく、概念をつかみやすいのでよく使われている。

システムを成功に導くためにどのようなことをすれば良いかを研究するのが実施研究であることは先に述べたが、ではシステムの成功とは一体何かという議論がわれわれの研究部会であった。成功の要因を探ってみると、①意思決定の支援ができること、②経営トップが満足すること、③採算がとれること、④業務改善の効果が顕われること、⑤利用されること、⑥浸透・定着すること、⑦組織革新ができること、等が考えられた。どれも決定的ではなかったが、とくに重要なものというも指摘できず、結局どれも大切な成功要因であるということになった。これらの要因に対して、どのようなことが必要かということによってユーザの関与、好意的な態度、能力などが挙げられた。もちろん誰がこのようなものを持つ必要があるのかといえ、スポンサ（経営トップ）、システム・アナリスト、ユーザ管理者、組織メンバ等であることが同時に考えられた。

これらの人々がシステムの実施サイクルのどの段階でどのような役割を果たすべきか、またどのような関係にあるかを一般化して相互の役割交換モデル (role-exchange model) としてとらえ、これをさらに精緻化して三幅対モデル (triad model) が提唱された (図3)。一般化されたものは、資源供与者

(sponsor), 解析担当者 (analyst), 情報利用者 (user) = 意思決定者), 効果摂受者 (receptor) であり, 相互の位置関係には図3のようになると考えられた。

これは資源供与者と効果摂受者の間に直接の関係がある場合で、四つの三幅対が考えられるケースである。相互の関係がないものとか関与者の欠ける場合は双三幅対 (bi-triad model) や、 Δ (デルタ) 型または ∇ (ナブラ) 型、さらに単純には対モデル (dyad model) などに変わる。このような役割の変化をどのようにとらえていくかについては、最近社会学とか社会心理学における役割理論 (role theory) においていわれている、「組織の中の公式の役割だけでなく、むしろ非公式の役割、つまり実際に仕事をする上での人と人とのつながりに、どういう役割関係が生れてくるのか」ということの研究結果の助けを借りて、実施化サイクルの過程と関与者の相互作用とを、どのように組み合わせていくかを考えていくことが今後の研究課題となる。

変革・革新の行われる組織では、たとえば情報処理の技術を変化させるとき、その技術を受け入れられるように、組織や仕事の構造を変え、同時にその組織に属する人々の態度を変容させ、意識を変革させなければ技術の変革を組織には導入できない。

この構造・技術・人の三つの要素は組織をコントロールする際の鍵になっており、それらは相互に深いつながりを持っている。このように考えると、コンピュータを使った OR/MS/IS (Information System) 等は、情報処理に関する技術を大きく変えることになり、それに伴う構造・人の変容を必要とする。

また事務改善において、仕事の仕組みを変えようとする、それに関係する人々の行動様式を仕事の仕組みに合うように変えなければならないし、そこで用いられる種々の技術も変えていかなければ、効

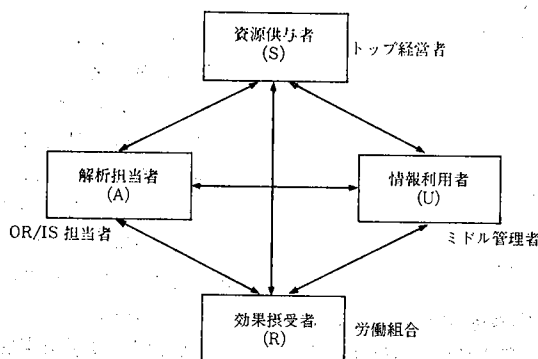


図3 一般四三幅対モデル^[2]

Fig. 3 General tetra-triad model

果をより大きく発揮させることができない。これも事務改善実施の過程における組織コントロールの革新という見方ができる。

4. おわりに

今まで述べてきたように、実施理論は実際に行われている組織変革や業務改善に関する研究成果を追求するものであるため、学際チーム(interdisciplinary team)によって行われていくべきものと考えている。OR学会の一分科会では、学者だけでなく、一般の企業人を含めた研究チームがあり、合理性の追求だけでなく、人間の情動性(emotionarity)をも含めた研究をしている。この種の研究が今後の情報システムの成功に、是非必要なことであろう。

参考文献

- [1] R.L.Schultz, D.P. Slevin, eds., *Implementing Operations Research/Management Science*, American Elsevier, 1975.
- [2] 松田武彦, "OR 実施システム・モデルと日本的組織風土", オペレーションズ・リサーチ, Vol. 23, No. 11, 1978.
- [3] 松田武彦, "80年代の経営と OR", オペレーションズ・リサーチ, Vol. 25, No. 8, 1980.
- [4] 根本忠明, "OR/MS 実施化研究の動向", オペレーションズ・リサーチ, Vol. 25, No. 11, 1980.

(教育部)

Raymond Turner 著

“Logics for Artificial
Intelligence”

Ellis Horwood, 1984, pp. 121.

本書は Ellis Horwood series in Artificial Intelligence の 16 冊目に当たる小冊子であり、タイトルが示すように、わずか 120 ページ程の中で、情報科学を含んだ広義の人工知能の分野で応用される各種の論理、とくに非古典論理の諸体系とその応用事例を紹介し、また多くの参考文献をあげている。論理と人工知能の両方に興味のある読者には、すでに論文等でなじみの深い内容ではあるが、本書のように人工知能と非古典論理の関係を述べた入門書は皆無であろうから一読を勧めたい。とくに論理の知識があまりない情報科学や人工知能の関係者には興味深いと思われる。その場合には、8.2 節の Prospects をまず最初を読むとよい。そうすれば、人工知能と論理についての眺望が得られるし、著者の問題意識もよく理解できよう。逆にまた人工知能にうとい論理の専門家にも一読を勧めたい。

広い意味での人工知能の目標はヒトの“知的な”情報処理の機能を明らかにし、それを機械によって実現し拡大するところにある。ヒトの知的活動は実に広範で多岐にわたるが、これを *lógos* すなわち、言語を使う知的活動に限定するならば、この *lógos* を研究の対象とする学問は Aristoteles に始まる論理学 (Logic) であることは言を俟たない。周知のように 19 世紀後半から論理学は主として数学を研究の対象とする数学基礎論によって著しく発展してきたが、その反面、こうした *μάθησις* を除く *lógos* の研究は立ち遅れていると言わざるをえない。

ところでこの両者、*lógos* と *μάθησις* の論理の間には明らかに大きな相違がある。たとえば、数学的理論は静的 (static) であり単調 (monotonic) であるのに対して、ヒトの *lógos* は時に矛盾を含み、動的 (dynamic) であり、非単調な論理を許容する。Popper^[1] や Lakatos^[2] によれば、実際、科学的理論においても矛盾の発生や反証によって Hegel 流の弁証法的展開がみられる。また数学的概念の厳密性の要請によって数学的理論では一様で外延的な概念の定義が行われるのに対して、*lógos* では概念は一般に話者によって個別に述べられ、局所的で一様で

なく、また内包的な性格をもち、形式化しようとする、たとえば Scott^[3] の indexicals のように多数の局所パラメータを伴う。したがって McCarthy^[4] が指摘するように、*lógos* を機械によって実現する人工知能においては、こうした *lógos* の特徴が改めて重要な問題となる。人工知能の条件である「機械によって実現する」という要請は、現存の工学を仮定すれば、計算論が示すように「実現しようとする知的機能は形式的に表現できなければならない」という要請と同等である。したがって、“哲学的”論理学とは異なり、人工知能では対象とする *lógos* を形式系として定式化する必要がある。L. E. J. Brouwer^[5] や E. Bishop^[6] にとっては、直観主義数学や構成主義数学は本来形式化すべきものではなかったのに対して、人工知能の世界はいわば formalisms と constructivism の交りの中にあり、それぞれの formalisms が不可欠である。同様にして、データベースやエキスパート・システム等の情報システムは動的に変化するから、Lakatos のいう counterexamples などのいわゆる矛盾を機械によって処理する必要があり、矛盾等、理論について語るメタ概念の形式化や概念の内包を局所性によって外延的に定義することが必要となる。1967 年の E. Bishop^[6] の構成主義数学の成功以来、P. Martin-Löf や S. Feferman から基礎論の研究者側から情報科学および人工知能への接近がみられる。また、とくにプログラムの合成ないしは導出では、日本でも高須達^[7]、佐藤雅彦^[8]、林晋^[9]らによる研究がみられる。しかし何分にも多様な *lógos* の諸局面での論理の研究はまだ試行の段階である。たとえば情報科学では、Y. N. Moschovakis^[10] らの generalized recursion theory による定式化の試みはあるものの、たとえばデータ型など現在のプログラム言語を反映したアルゴリズムの一般論さえ存在していない。こうした意味では、ヒトの *lógos* の諸局面の論理を形式化する努力は息の長いこれからの仕事であろう。

本書は小冊子であり、各論理系を円満にカバーしているとは言えないが、本書を入口として人工知能とその論理とを考えてみるのは非常に意義深いと思われる。

章立てを述べると、以下の 8 章から成っており、1 章は 2 章以降の紹介と一階述語論理を紹介し、2～4 章は情報科学、とくにプログラムの仕様、導出、検証に应用される諸論理を、3, 5, 6, 7 章

はいわゆる人工知能に応用される諸論理を紹介している。8章では本書でふれなかった論理として many-sorted logic, weak second-order logic, および infinitary logic をあげ、最後に将来の展望を述べている。

- 1章 introduction
- 2章 Modality and dynamic Logic
- 3章 3-valued Logics and their computational interpretations
- 4章 intuitionistic logic : Martin-Löf's theory of types
- 5章 Towards a semantic theory of non-monotonic inference
- 6章 Temporal logic in artificial intelligence
- 7章 Fuzzy Logic and expert systems
- 8章 Other Logics and further prospects

参考文献

- [1] K. R. Popper, *Conjectures and Refutations: The Growth of Scientific Knowledge*, Routledge & Kegan, London (1963).
- [2] I. Lakatos, "History of Science and its Rational Reconstructions", Y. Elkana, ed., *The Interaction between Science and Philosophy* (1974).
- [3] D. Scott, "Advice on model Logic", K. Lambert, ed., *Philosophical problems in Logic: some recent development* (1970).
- [4] J. McCarthy and P. Hayes, "Some Philosophical Problems from the Standpoint of Artificial Intelligence", *Machine Intelligence 4*, Edinburgh University Press (1969).
- [5] D. van Dalen, ed., *Brouwer's Cambridge lectures on intuitionism*, Cambridge University Press (1981).
- [6] E. Bishop, *Foundations of Constructive Analysis*, McGraw-Hill (1967).
- [7] S. Takasu, "Proofs and Programs", 3rd IBM Symp. on Math. Foundations of Computer Science, (1978).
- [8] M. Sato, "Toward a mathematical theory of program synthesis", 6th IJCAI, (1979).
- [9] S. Hayashi, "Constructive mathematics and program synthesis", 数学基礎論予稿集, 京都大学数理解析研究所, 1983.
- [10] Y. N. Moschovakis, *Abstract Recursion as a Foundation for the Theory of Algorithms*, LNM 1104, Springer (1983).
- とくに, Constructivism について以下を補足しておく。
- [11] M. J. Beeson, *Foundations of Constructive Mathematics*, Springer (1984).
- [12] F. Richman, ed., *Constructive Mathematics*, LNM 873, Springer (1980).

野崎昭弘／はやし・はじめ／柳瀬尚紀 共訳

ゲーデル, エッシャー, バッハ

—あるいは不思議の環—

白揚社, 菊判, 744+xxii pp. 1985年, 4800円

ゲーデル, エッシャー, バッハ不思議の環の著者ハ | 5歳の頃に自分が興味のある事柄に関心を持つデキの良い年代者に読んで欲しいといったガンバルな著者ごい本が出たな。これが御歳 1 0 0 1 0 | エ、年とった私が6年前に原著を眺めた時の感想ヤツパリいい本だな。これ今回の読後感想で文句無し

☆

ゲーデル(G)の不完全性定理の自己参照性(自己言及性)に画家エッシャー(E)や大音楽家バッハ(B)の作品に見られる“繰返し”の技法の面白さからまかせて、書き上げられたのが本書である。著者にとって「ゲーデルとエッシャーとバッハは、何か中心をなす堅固な本質によって異なる方向に投げられた影にすぎない」のであった。すなわち、著者はこの中心をなすものを再構成しようと努め、そして本書を仕上げた。

著者が本書のアイデアを得たのは、物理学の大学院生の時であるという。その後、アイデアはどんどん成長し、膨大な量の原稿となり、1979年に単行本として日の目を見た。本は、翌年度のピューリッター賞を獲得、一躍ベストセラーとなった。その頃、日本でも話題になったものである。

☆

エッシャーに「上昇と下降」という作品がある。修道士たちが環になって階段を永久に昇りあるいは降っている絵である。この絵に、環に加わらない修道士が2人描かれている。1人はベランダの手摺に肘をかけて、環の彼らを見ている。彼は、画家エッシャー?ゲーデル?著者ホフスタッター?あるいは読者?そしてあと1人はそれらすべてに背を向けて座り込んでいる修道士。彼は???

☆

さて内容であるが、本書の「GEBの概要」から紹介してみよう。

「序論 音楽＝論理学の捧げもの」 バッハにおける自己言及と相異なるレベルの間で交わされるインタープレイを論じ、エッシャーの絵やゲーデルの定理に見られる同様のアイデアに言及する。定理のためバックグラウンドとして、論理とパラドックスに関する歴史を簡単に紹介する。そして、機械による推論とコンピュータ、人工知能は可能かとの討議に進む。

「第1章 MUパズル」 パズルを通して文字列、定理、公理、推論規則、生成過程、形式システム、決定手続き、システムの内／外での仕事、のような数々の基本概念を紹介する。

「第2章 数学における意味と形」 意味と同型対応との密接なつながりを考え、その他、真理、証明、記号操作、「形」というとらえどころのない概念など、意味に関連した問題に論及する。

「第3章 図と地」 芸術における図と地の区別を、形式システムにおける定理と非定理の関係とを対比させ、さらに再帰的に可算な集合と再帰的集合の区別に言及している。

「第4章 無矛盾性、完全性、および幾何学」 幾何学の歴史を紹介し、これから、幾何学どうしの無矛盾性の考えを導き出す。さらに、無定義術語の一般概念、知覚および思考のプロセスと無定義術語の関係を明らかにする。

「第5章 再帰的構造と再帰的過程」 再帰性という概念を、音楽の関数、物理学の理論、コンピュータ・プログラムなどを例にして示す。

「第6章 意味の所在」 コード化されたメッセージ、解き手、受け手の間でどのように意味が分裂するかを論じる。

「第7章 命題計算」 “そして”、“もし……ならば”、“または”、“でない”といった言葉、形式規則でどこまで統御できるかを示す。そして、こうしたシステムにおける同型対応と意味の自動的獲得という考え方を提示する。

「第8章 字形的数論」 TNTと名づけた命題計算の拡張である。TNTでは、数論の証明は明確な記号操作によってなされる。形式的な証明と人間の思考の違いを考察する。

「第9章 無門とゲーデル」 ある意味で禅の思想は今日の数学をめぐる哲学に似ていると考え論じ、さらにゲーデル数をめぐるゲーデルの根本的な考えとゲーデルの定理が生まれた経緯を紹介する。

「第10章 記述のレベルとコンピュータ・システム」 絵画、チェス盤、コンピュータ・システムを見るさいの、さまざまなレベルについて論じる。さらにシステムの合成の仕方へと方向を転じていく。

「第11章 脳と思考」 脳の構造について、大小、スケールを変えて示し、概念と神経の働きの関係にある程度詳しく憶測的に論じる。

「第12章 心と思考」 コミュニケーションがどうして可能なのか、人間が共通して脳にもっているものとは何か、その答を求めて、地理学的アナロジを試みる。

「第13章 ブーとフォーとグー」 数論における原始再帰関数と一般再帰関数について直観的理解を得るため、予測できる有限の探究や、予測がつかず無限であるかもしれない探究を取り扱う。

「第14章 形式的に決定不可能な TNT と関連するシステムの命題」 ゲーデルの証明の二つの主要な部分をなぞって、数学をめぐる哲学が暗に秘めているものを調べる。

「第15章 システムからの脱出」 TNTは本質的に不完全であることを暗示し、ゲーデル問題が繰返し可能であることを明らかにする。

「第16章 自己言及と自己増殖」 コンピュータ・プログラムやDNA分子のような自己増殖するものと、コンピュータやタンパク質のようにそれらを外から助けるメカニズムとの関係や両者の境界の曖昧さを論じる。

「第17章 チャーチ、チューリング、タルスキ、その他」 人間の思考を機械的にシミュレートしたり、美を感じる、あるいは生み出す能力を機械プログラミングするといった点を分析する。脳の活動と計算の結びつきから、チューリングの停止問題とかタルスキの真理定理といった他の話題にも及んでいく。

「第18章 人工知能=回顧」 チューリング・テストから人工知能の歴史の要約へと進む。ある程度までゲームができるもの、定理を証明できるもの、…、自然言語が使えるものといったプログラムにも言及する。

「第19章 人工知能=展望」 何層にもなった文脈に知識がいかに表されるかの議論から、フレームの考えへと導く。そして、概念一般の相互作用に関する奥深い問題を論じ、創造性に関する考察へとつながっていく。

「第20章 不思議の環、あるいはもつれた階層」 システムがそれ自体にかかわってくるときに生じるもつれに関するものを論じ、ゲーデル、エッシャー、バッハをもう一度一緒に論じる。

これらの各章の間には、バッハの作品をもじった対話劇が挿入されている。

☆

以上のように羅列しても、本章の雰囲気伝わらないかと思うが、本書が学際的な時代感覚と彼一流のユーモアによって書かれた良書で、考えられる最良の訳者たちによって発行されたことは確かである。現代科学を知るうえでも一読を薦める。

(Douglas R. Hofstadter, “GÖDEL, ESCHER, BACH”, Basic Books, 1979)

今年発表の製品の中から主なものを選んで紹介します。各製品の詳細についてはマニュアル等をご参照ください。

●シリーズ 1100 IPF 1100/JED

IPF 1100 (Interactive Processing Facility for Series 1100)/JED は、対話型処理支援プログラムの中核となるソフトウェアであり、利用者の生産性向上に寄与する機能を備えた汎用システムである。

IPF 1100/JED の特長は、①指定言語はキーワードを用いたパラメタ指定方式を採用し学習しやすく、かつ覚えやすい考慮がされていること、②全面編集モードと行編集モード、日本語データ編集機能とカナ漢字変換機能、パターン照合機能、編集指令の取り消し機能など豊富で使いやすい編集機能、③日本語文書システム/作成・出力プロセッサ (JDOC) と連動し、JDOC の出力するゲラ刷りファイル (校正用ファイル) と原稿行イメージを画面分割により同時表示し、原稿行イメージの編集が可能なこと、④指令を組み合わせて、利用者独自の指令 (手続き) を作成し指令と同様に呼び出すことができ、制御の流れを変える指令やループ指令を用いて柔軟性のある手続きを作成することが可能なこと、⑤ IPF 1100/JED の指令、パラメタ、システム変数/関数および誤りメッセージについて、利用者の要求に応じて即座に説明する援助機能などがある。

(資料コード: 481205829)

●シリーズ 1100 PASE 1100

PASE 1100 (Program Automatic Specification Environment for 1100) は、プログラミング言語であると同時に、プログラムの仕様をそのまま記述する言語である。プログラムの仕様は、一般に入力データについて成立する述語 (入力表明) と入力と出力間の関係 (出力表明) によって構成されるが、PASE 1100 ではこの表明を比較的自然な文章で容易に記述できる。また、課題をデータ・フローに基づきプロセス分解し、プロセスの結合として表現することができる。また、PASE 1100 は COBOL との親和性があり、仕様記述から COBOL プログラムとドキュメントを自動生成することもできる。ドキュメントとしては、仕様書である PASE ソース・リスト、プロセスの分解・結合を表現したプロセス・フロー図、各プロセス構造図など図形で出力する。さらに、PASE ソースを日本語に変換した日本語リストを出力することもできる。

(本誌 p. 76 参照, 資料コード: 481205337)

●新シリーズ 8 システム 100~400



従来のシリーズ 8 モデル 15 II ~ 45, 50, 60 を整理し、ワイドレンジをカバーする 4 モデルとし、従来はモデル 15 II ~ 45 まで同一 CPU 処理能力をモデル別に階層化した新システムである。

システム 100 では、5 インチ固定ディスク (20/40/65 MB) 採用によるコンパクト化、自動運転機能の標準装備が図られている。

システム 200 では、高速ターボ・プロセッサ搭載が可能で、処理業務の形態によっても異なるが 10 パーセントから 30 パーセントもの性能向上が図られる。

システム 300, 400 では主記憶容量がシステム 300 で 4 から 12 MB, システム 400 で 4 から 16 MB と拡張でき、ワークステーション台数についても、システム 300 で最大 32 台、システム 400 で最大 64 台接続することができる。

オペレーティング・システムについても機能が強化され、システム 100, 200 の OS である DPS IV では、ファイル自動拡張機能、分散データベース自動ファイル・アクセス機能、マイクロメインフレ

ム・コネクション機能、カナ漢字変換機能の改良、ディスク・キャッシュ機能、64 KB 超プログラムのサポートなど多くの機能強化がなされている。

システム 300, 400 の OS である DPS 10 についても、処理能力の向上、マイクロインフレーム・コネクション機能、大規模高度ネットワーク・システム対応など多くの機能強化がされている。

(資料コード：S-8-100 472755320)

(資料コード：S-8-200 472755321)

●UNIVAC DS7



多機能デスクステーション DS7 は、CPU にインテルの 16ビット MPU である i80186 を、オペレーティング・システムに本格的なマルチ・タスクを可能にした Concurrent CP/M™ を標準搭載。最大四つの独立した仕事（タスク）を同時併行処理するマルチ・タスク機能、一つのディスプレイ画面を最大四つに分割、複数のタスクを一つの画面で同時表示できるマルチ・ウィンドウ機能を備えている。

Concurrent CP/M™ の下に統合ソフトウェア U-MENU（ポップアップ・メニュー）を配置することにより、多彩なユーティリティ・プログラム群を効率的に統合・管理する統一的操作環境の提供を可能としている。プログラミング言語としては LEVEL II COBOL, J BASIC を、通信ソフトウェアとしては、通信プロトコルとして UTS シリーズ, SDLC, BSC (JCA, 全銀手順など), TTY, 通信回線は、特定・公衆回線, DDX (回線交換網, パケット交換網) をサポートする豊富なライブラリが用意されている。

その他、表計算/データベース、グラフ/イメージ、日本語文書作成などを支援するソフトウェア群も用意されている。

ハードウェアの特徴としては、最大 1 MB の主記憶装置と 5.25 インチ・ディスク (0.64 MB/1 MB) の内蔵補助記憶装置を標準搭載され、オプションとして 5.25 インチ・ディスクまたは 5.25

インチ・ディスク (10 MB/20 MB) のいずれか 1 台が内蔵できる。

また、キーボード、CRTディスプレイ（高解像度ビット・マップ・ディスプレイ、1120×750ドット、カラー7色）には、使いやすく、長時間使用しても疲れない人間工学設計が施されている。

* Concurrent CP/M™ は米国デジタル・リサーチ社の登録商標です。

(資料コード：481843005)

●UNIVAC DS3



デスクステーション DS3 は、UTS 4000 ファミリーとの互換性を備えており、UTS 4000 ファミリーで開発された各種プログラムをそのまま利用できるので過去の投資が有効に活かされる。また、DS3 は日本語ターミナルとして、15×16ドットの画面表示、24×24ドットの高品質の日本語出力機能を備えている。オプションとしてグラフ機能が用意されており、663×414ドットの解像度で鮮明なグラフを描くことができる。ディスプレイは白い画面上に情報を表示するペーパー・ホワイトタイプで、文字・図形のチラツキがないブラウン管、反射をなくす防眩スクリーンを採用、ディスプレイの傾斜角度の変更、回転が自由に行えるチルト・ローテート・ベースが標準装備されており、オペレータの疲労を軽減する人間工学的な配慮がされている。

(資料コード：481843006)

●SCHIP

SCHIP (SCHematic Input Progroam) は、シリーズ1100の下で稼動する図形入力システムであり、各種ネットワーク・モデルの構築・変更・検索・検証・作画などが容易に行える汎用システムである。

SCHIP の特徴として、①AGS 2000シリーズ・カラー・グラフィック・ディスプレイにネットワークの図を表示し、目視により正当性を確認しながらネットワーク・モデルの構築、変更、検索、検証が可能なこと、②ネットワークを構築するシンボルの形状を任意に定義でき、接続関係を扱う広汎な分野に適

項目	UNIVAC DS7 の仕様	UNIVAC DS3 の仕様	
プロセッサ	16ビット・マイクロプロセッサ (180186) (8 MHz)	8ビット・マイクロプロセッサ (Z80B) (4 MHz)	
表示部	画面寸法	12型	
	表示色	カラー7色 (緑, 赤, 青, 白, 桃色, 淡青) またはモノクローム (ペーパー・ホワイト)	
	グラフィック表示	1120(横)×720(縦) ドット	
	表示モード	文字表示, グラフ表示および文字/グラフ重ね表示	
編集機能	表示文字数	漢字 960 字 (40字/行×24行), 英数字仮名 1920 字 (80字/行×24行)	
	表示方式	ドット・マトリックス: 漢字 (24×24ドット), 英数字カナ (11×17ドット)	
	表示文字種	約 12,000 種	
	編集機能	けい線表示, 反転表示, 点滅表示, データ保護, 非表示, 行単位の複写/挿入/消去, 文字単位の挿入/消去など	
記憶部	主記録 (RAM) 漢字フォント・メモリ 外字用メモリ	120 KB 基本: 非漢字, JIS 第1水準 236 文字	
	通信機能	2400~9600 BPS 独立同期方式 ユニバック標準ポーリング方式 (ポーリング/セレクション) 特定通信回線, 公衆通信回線, DDXパケット交換網	
補助記憶装置 (内蔵形)	最初の1台: 5.25インチ・ディスク装置 (660 KB/1 MB 対応)* 2 台目: 5.25インチ・ディスク装置または固定ディスク装置 (10または20 MB)	3.5インチ マイクロ・ディスク装置 (500 KB) 1台	
	入力キーボード / 付属機器	キーボード (磁気ストライプ読取装置およびマウス付加可能)	キーボード (標準) (磁気ストライプ読取装置付加可能)
周辺装置	補助記憶装置	5.25インチ固定ディスク装置 (10または20 MB) 8インチ・ディスク装置	—
	印書装置	0473型, 0469-II型, 0474型 (これらのうちから, 基本構成として1台, 増設1台)	0474型 (I型またはII型のうち1台選択可)
その他	イメージ・スキャナ	接続可能	—
	その他	3台目以降のディスクおよびディスク装置, 増設用印書装置, イメージ・スキャナ, 増設用の通信アダプタはオプション・スロットに接続する。	グラフ機構を接続可

備考 * 5.25インチ・ディスク装置は, 1台の駆動機構で660 KB, 1 MB の形式を扱う。

用できること、③マクロ機能を用いることにより、大規模なネットワーク・モデルを容易に構築できること、④コンピュータ内部に構築したネットワークのデータは、いろいろなアプリケーション・プログラムに渡すことが可能であり、それぞれの目的に応じた処理を容易に指示することができる。

(資料コード：483205414)

●UNIVAC 1100 70/II シリーズ

UNIVAC 1100 70/II シリーズは、UNIVAC 1100/70 シリーズのハードウェア技術を継承・発展させたもので価格/性能比は一段と向上している。

システムの中核である中央系処理装置は、中央処理装置 (CPU)、記憶制御装置 (SIU)、主記憶装置 (MSU)、入出力処理装置 (IOU)、システム・サポート・プロセッサ (SSP)、システム・コンソールなどから構成されており、各プロセッサは機能的に独立し、機能分散化されている。

UNIVAC 1100 70/II シリーズでは、CPU、SIU、IOU を一つのキャビネットに実装し、さらに CPU あたり浮動小数点演算速度の改善が図られた 2 個の IP (Instruction Processor) を内蔵している。これにより中央処理装置を著しくコンパクト化し、省スペース、省エネルギー化を実現している。

UNIVAC 1100 70/II シリーズは、これらの機能分散プロセッサの組み合わせにより、単一プロセッサと多重プロセッサの二つのシステム構成をとることができる。

(資料コード：08128003)

●FSP 1100 グラフ・オプション

FSP 1100 グラフ・オプションは、FSP 1100 のもとでグラフ編集処理およびシンビオント・ファイル・イメージをファクシミリ端末へ送信する使用者支援ソフトウェアである。

主な機能は、グラフ用ライブラリを使用し、DSP ファイル内の作画情報からファクシミリ端末用のイメージに編集すること、パラメタ指示により印書装置出力用シンビオント・イメージをファクシミリ端

末へ送信することであり、ASCII コード、ユニバック漢字コードの JIS 第 1 水準および第 2 水準相当コードを対象としている。

(資料コード：481205739)

●MAPPER 1100 ユーティリティ

MAPPER 1100 を取り巻く汎用性のあるユーティリティとして、13個のランと 2 個の MASM プログラムで構成されている。

MAPPER ランの内訳は、モード/ごとのレポート、ライン数集計ラン (ACTRID)、レポートの追加・削除ラン (ADDDEL)、レポートのリザルト化ラン (CRES)、文字列検索ラン (CROSS)、RID0 構築ラン (FMTGEN)、必要項目収集レポート合成ラン (GREPO)、けい線/カラー画面作成支援ラン (KC-DSP)、レポートけい線印書サポート・ラン (KCOP)、行挿入ラン (LINS)、132桁レポート会話型処理ラン (EXxxx)、複数レポートの FCC ラン (NFCC)、MAPPER レポート/SDF ファイル間レポート転送ラン (SSETUP)、モード・タイプ・コピー・ラン (TCOPY) である。

MASM プログラムとしては、MAPPER レポート作成サブルーチン (MAPPER-EXT)、ファイル分割アサイン・プログラム (MPA) がある。

(資料コード：481205502)

●シリーズ 1100 VIDEOTEX 1100/DF

VIDEOTEX 1100/DF は、UNIVAC シリーズ 1100 を NTT の提供するビデオテックス通信網に接続して、直接型情報センタ (DF) を容易に構築することができるソフトウェア・パッケージである。

主な機能は、ビデオテックス通信網に接続する際に必要となる通信プロトコル制御機能、情報入力端末を接続するための通信プロトコル制御機能、センタ業務の運用、画像情報の管理などを強力に支援する機能、容易に画像情報を作成・更新することのできる使用者業務支援機能などである。

(資料コード：481205744)

▶テクニカル・コーディネータ

《Sperry, Computer Systems》

Dr. A. J. Schneider (Vice President, Advanced Technology Research), P. J. Lazar (Director, Strategic and Business Planning)

《日本ユニバック》

加藤嗣弘 (商品企画二部長), 佐藤政俊 (CAD/CAM システム一部 CAD/CAM システム統括一課長), 菅 克夫 (ネットワークソフトウェア部長), 関谷忠一 (開発四部長), 高橋 肇 (知識システム開発部 業務課長), 高山龍雄 (システム本部 副本部長), 外山晴夫 (アドバンスドプロダクト企画部長), 中村 脩 (営業推進本部 副本部長), 横 元治 (カスタマーサービス企画部長), 村上信夫 (ソフトウェア一部長), 山口拓 (生産技術二部長), 山田真市 (技術研究部), 渡部義維 (応用ソフトウェア二部長)

▶エディトリアル・スタッフ

《日本ユニバック》

(テクニカル・パブリケーション室)

村井啓一 (室長), 下田宏一 (主任研究員), 桑野龍夫 (主任研究員), 小山憲一, 青柳幸久, 丹野敬子

●Technical Coordinators

P. J. Lazar, M. Maki, N. Murakami, O. Nakamura, M. Satou, A. J. Schneider, T. Sekiya, K. Suga, H. Takahashi, T. Takayama, H. Toyama, Y. Watanabe, S. Yamada, T. Yamaguchi

●Editorial Staff

K. Murai, K. Shimoda, T. Kuwano, K. Oyama, Y. Aoyagi, K. Tanno

ISSN 0289-6257

技 報

UNIVAC TECHNOLOGY REVIEW

No. 9

発行日 昭和60年8月31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

禁無断複製転載

●最新刊●

MS-FORTRANによる 有限/境界要素解析 プログラミング

三好俊郎著 A5・2300円
16ビットパソコンとFORTRANの組合せによる
実用的な有限/境界要素解析を詳述する。
→ソフト別売【PC-9801; MS-FORTRAN,
PC-FORTRAN用】. 詳細は小社営業部まで。

例解 有限要素法への アプローチ

—有限要素法による偏微分方程式の数値解法—
A.J. デービス著/藤川洋一郎訳 A5・2800円
有限要素法を用いて偏微分方程式の実際的解法
とその理論が修得できるよう配慮された書。予
備知識を必要とせず初学者用テキストとしても
最適。

有限要素法による 熱応力・クリープ・熱伝導解析

矢川元基・宮崎則幸共著 A5・4800円

FORTRAN77による 境界要素法の基礎

水本久夫・原 平八郎共著 A5・予2500円

明解COMP-X&CAP-X

—第1種・第2種情報処理技術者試験対策—
玉井 浩著 A5・1300円

Personal Computing

—IBM5550/5540科学技術計算への活用—
棚町芳弘・田村栄悦共著 A5・2800円

PROLOG入門

後藤滋樹著 A5・1800円

計算機科学入門

M.A. アービブ他著/甘利俊一他訳 A5・2300円

オートマトン 言語理論 計算論 I

J. ホップクロフト他著/野崎昭弘他訳 A5・2800円

パーソナルコンピュータによる 量子力学

J.P. キリンベック著/有馬朗人他訳 A5・2400円

隔月刊誌

Computer Today

定価880円
偶数月18日発売

本誌の特色

本誌は、人間とコンピュータという視点で
計算機科学固有の問題、
周辺諸科学との関係、
人間社会との関わりあい、
をトータルにとらえてゆこうとする雑誌です。
専攻の学生・研究者/隣接領域の専門家/コ
ンピュータに関心のある方々のために、広く
関心のもたれているテーマをタイムリーに取
り上げてゆきます。

■お得な年間購読のおすすめ
年間予約購読料 5000円(別冊を除く)

既刊特集

- 84/ 5 第5世代コンピュータとProlog
- 7 スーパーコンピュータ
- 9 UNIX & C
- 11 Smalltalk-80
- 85/ 1 コンピュータと認知科学
- 3 コンピュータネットワーク
- 5 リレーショナルデータベースシステム
- 7 高機能ワークステーションのすべて

別冊 **PAD** —構造化プログラム開発技法—
1200円