

技 報

UNIVAC TECHNOLOGY REVIEW

1984年8月 第7号

論 文

高階微分方程式の解析法……………藤野 勉 1

報 告

ボトル設計システム……………佐藤芳雄, 明尾 誠 20

MOS-VLSI欠陥の分類, テスト, および除去……………Y. M. El-ziq 32

シリーズ1100カナ漢字変換システムの開発……………三ッ矢裕一, 吉田正行, 小山憲一 40

プログラム言語の日本語化実験

——日本語 PL/I の作成……………真田正二 57

データ・セキュリティ

——障害に耐える閉じた環境としての1100/90シリーズ……………S. J. Rawlins 77

二相流解析

——その背景とモデリング……………石丸 潤 96

TECHNOLOGY TREND

情報処理教育のカリキュラム……………朝倉文敏 108

言語 Occam の概要……………若鳥陸夫 112

無人化運転システム……………今西秀文 116

BOOKS……………119

MEMORANDUM……………121

CALENDAR……………122

NEW PRODUCTS……………125

EDITORS' NOTE……………表2

4階の微分方程式は板曲げや2次元粘性流解析に実用的に利用されているが、6階以上のものを任意節点配置の差分法で解くとすると、かなり多量型の計算となることが予想される。有限要素法でも高次の要素を用いることが必要となり、その形状関数を求めることも簡単ではない。藤野の**高階微分方程式の解析法**では、高階の微分方程式で Green 積分により導入される境界条件式の数が、境界積分によるそれよりも少ない点に目を付け、これを任意節点配置差分法により数値的に解析する手段を考察している。

佐藤らの**ボトル設計システム**は、ライオン(株)からの依頼で開発したボトル設計を支援する実用的 CAD システム BDAS についての報告である。ボトル設計を対象に開発したシステムではあるが、デザイン図画上の線を入力して面を作成していく BDAS の方法は、各種容器や照明器具、玩具、時計の外装といった、幅広い意匠設計の分野にも応用できるものと思われる。

シリコン集積回路のテストは VLSI 回路生産の急激な拡大によってますます困難になってきている。多くのテスト可能性問題がゲート数とピン類の増加、ならびに速度の向上により引き起こされている。さらに幾何学的微細化による信号線間隔の近接と製造工程の複雑化で、データ依存あるいは近接部の干渉による故障といった新しいタイプの障害を引き起している。Y.M. El-ziq の **MOS-VLSI 欠陥の分類、テスト、および除去**では、縮退故障モデルの適切さを調べ、MOS の物理的欠陥を分類し、これらの障害のあるクラスを検出するテスト生成の方法を提示している。また、これらの障害のいくつかを除去するための設計規則とガイドラインを導いている。

入力の問題は、文字種の多い日本語処理では大きな障害となっている。この障害を取り除く有力な入力方式の一つにカナ漢字変換がある。入力速度、使いやすさ、さらに特殊な入力装置を必要としないという点から他の方式より優れており、使用者のニーズも高い。三ツ矢らの**シリーズ 1100 カナ漢字変換システムの開発**は、シリーズ 1100 のもとで稼動するカナ漢字変換システムである“日本語文章システム/入力”の概要とそのテスト結果の報告である。なお、同システムは稲永氏(九州芸工大)作成の辞

書と、日高氏(九大)、吉村氏(福岡大)達が提唱している文節数最小法を基本としている。

日本語によるプログラミングが脚光を浴びている。アンケート調査によると、既存言語の日本語化はリテラル、注釈、使用者定義名を日本語にできればよいと考える人が多い。真田の**プログラム言語の日本語化実験——日本語 PL/I の作成**は、既存プログラム言語の日本語化の実験報告である。PL/I の文をキーワードの一对一交換だけで、それなりに日本語らしく見せるのに成功している。

信頼できるコンピュータとは必ずしも障害がないものではなく、障害があってもそれに耐えるものであればよい。障害はユーザのプログラムの論理エラーのようにありふれたものもあり、常駐しないページへのアクセスのようにまれなものもある。S.J. Rawlins の**データ・セキュリティ——障害に耐える閉じた環境としての 1100/90 シリーズ**では、1100/90 シリーズに組み込まれた保護機構の概略と、それがどのように障害に耐える閉じた環境を実現しているのかについて述べている。

二相流は、産業や生活の中に数多くその例をみる。原子炉、抽出・蒸留装置、混相輸送、ハイウェイ交通、液晶、雪崩、血液、等々。石丸の**二相流解析——その背景とモデリング**は、相変化を伴う気液二相流に限定し、その解析、とくにモデリングについて紹介する。二相流解析の背景を明らかにし、現象の物理とモデルを結びつけていく方法論を整理することと、モデリング上重要な問題を経験にもとづいて指摘することである。

☆

論文 高階微分方程式の解析法

Analysis Methods of Higher Order Differential Equations

藤野 勉

要約 4階, 6階等の高階の微分方程式では境界条件式の数が複数個である. Green 積分より導入される境界条件式の数はそれぞれ 3, 6, ... 個であるが, 境界積分によってこれらは 2, 3, ... 個に低減することができる. このように, 高階微分方程式では境界条件式の数は複数個となるので, 差分法等により数値解析を行うとき境界には不完全高次節点, または数個の 0 次節点を境界外側に設置することが必要である. これらをダミー節点と呼ぶこととする. 当然, 領域内は 0 次節点のみであるから, ダミー節点を用いる方が統一した取り扱いができるので便利である.

Abstract Higher order differential equations of such as 4th, 6th...has a number of boundary condition which are 3, 6, ... respectively when derived from Green integration whereas they are reduced to 2, 3, ... by boundary integration. Therefore it is necessary to put an imperfect higher order node on boundary or to put some zeroth order node outside of the bounday which are called dummy node for numerical analysis by finite difference method. It is convenient to use dummy node method rather than imperfect higher order node method, because we also have same kind of zeroth order nodes in domain.

1. はじめに

たとえば平板の面外変形の問題で, 板の内部では横変位 w は 4 階の微分方程式で表され, 固定, 支持の境界では二つ, 自由境界では三つの条件式が一般に考えられる. しかるに, その後 Kirchhoff により自由境界でも二つの式で十分であることが指摘されている. このように高階の微分方程式では, 解が一意的に存在するための境界条件のあり方について考察の余地が残されているように思われる. 高階の式としてはさらに 6, 8, ... 階の微分方程式も考えられる. たとえば, 円筒殻における Donnell の式は 8 階の微分方程式である.

高階の微分方程式は, それが導入される過程にも種々あり, それに従って境界条件の形も多種多様であると思われるが, ここでは主として歪関数と作用関数の内積の Green 積分により導かれるものについてのみ考察を進める.

境界条件式も併せて高階の微分方程式が与えられたとき, これを数値的に解析する手段としては有限要素法と任意節点配置差分法の二つが考えられるが, ここでは主として後者の 2 次元問題について述べる.

なお一般に 2, 4, 6, 8, 10, ... 階の微分方程式は, それぞれ 1, 2, 3, 4, 5, ... 個の境界条件式が必要にして十分である.

2. 4 階微分方程式

前に述べたように高階の微分方程式が導入される過程には種々の場合があるが, ここではまず 2 変数, 2 階の微分方程式から, 一つの変数を消去して導く場合について考察する.

いま二つの変数を $u(x, y)$, $v(x, y)$ とし, これらは微分方程式

$$L_{11}u + L_{12}v = f_1 \quad (2-1)$$

$$L_{21}u + L_{22}v = f_2 \quad (2-2)$$

を満たしているものとする。境界上では u, v の値が指定されているものとし、これによる解 u, v は一意的に存在するものとする。ここに L_{mn} は 2 階の微分作用素である。つきに式(2-1)より、

$$v = L_{12}^{-1}(f_1 - L_{11}u) \quad (2-3)$$

をつくり、これを式(2-2)に代入して下記微分方程式をつくる。

$$(L_{11}L_{22} - L_{12}L_{21})u = L_{22}f_1 - L_{12}f_2 \quad (2-4)$$

上式は 4 階の微分方程式である。なお L_{12}^{-1} は L_{12} の逆微分作用素である。すなわち、

$$L_{12}^{-1}L_{12} = 1 \quad (2-5)$$

したがって一般に、 L_{12}^{-1} は無限階の微分作用素で、境界条件式 $v = \bar{v}$ は無限階の微分方程式である。とくに $L_{12} = 1$ ならば $L_{12}^{-1} = 1$ で、上式は 2 階の微分方程式となる。このように 4 階の微分方程式では、境界条件式の数は 2 個であるが、その性質は非常に多様であることを示したものである。

つきに Green 積分により導入される微分方程式、境界条件式について考察する。

2 次元 $x-y$ 平面内における変数を $u(x, y)$ とし、その 2 次までの歪関数を

$$\varepsilon_{xx} = u_{xx}, \varepsilon_{xy} = 2u_{xy}, \varepsilon_{yy} = u_{yy}, \varepsilon_x = u_x, \varepsilon_y = u_y, \varepsilon = u \quad (2-6)$$

とし、広義の応力関数（以後、作用関数と呼ぶ）を

$$M_{xx}, M_{xy}, M_{yy}, M_x, M_y, M \quad (2-7)$$

として、下記の Green 積分を行う。

$$\begin{aligned} \delta U &= \iint_S (M_{xx}\delta\varepsilon_{xx} + M_{xy}\delta\varepsilon_{xy} + M_{yy}\delta\varepsilon_{yy} + M_x\delta\varepsilon_x + M_y\delta\varepsilon_y + M\delta\varepsilon) dx dy \\ &= \int_S (F_x\delta u_x + F_y\delta u_y + F\delta u) ds + \iint_S L\delta u dx dy \end{aligned} \quad (2-8)$$

ただし、

$$F_x = l_x M_{xx} + l_y M_{xy} \quad (2-9)$$

$$F_y = l_x M_{xy} + l_y M_{yy} \quad (2-10)$$

$$F = l_x \left[- \left(\frac{\partial M_{xx}}{\partial x} + \frac{\partial M_{xy}}{\partial y} \right) + M_x \right] + l_y \left[- \left(\frac{\partial M_{xy}}{\partial x} + \frac{\partial M_{yy}}{\partial y} \right) + M_y \right] \quad (2-11)$$

$$L = \frac{\partial^2 M_{xx}}{\partial x^2} + 2 \frac{\partial^2 M_{xy}}{\partial x \partial y} + \frac{\partial^2 M_{yy}}{\partial y^2} - \left(\frac{\partial M_x}{\partial x} + \frac{\partial M_y}{\partial y} \right) + M \quad (2-12)$$

である。なお、 l_x, l_y は境界 s における外方法線 n の方向余弦

$$l_x = dy/ds = \sin \theta, \quad l_y = -dx/ds = -\cos \theta \quad (2-13)$$

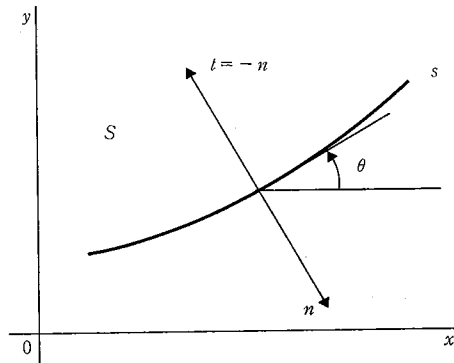


図 1 境界における外方法線 n 内方法線 $t = -n$ 切除角 θ

Fig. 1 Inward and outward normal

を表す. ことに θ は x と s の挟む角である.

2.1 構成方程式

弾性力学に従って歪関数と作用関数の関係を表す式を構成方程式 (Constitutive Equation) と呼ぶ. ここでは簡単にするため, 下記正値対称な剛性行列を仮定する.

$$M_m = k_{mn}\varepsilon_n, \quad k_{mn} = k_{nm}, \quad k_{mn}\varepsilon_m\varepsilon_n \geq 0 \quad (2-14)$$

たとえば,

$$\begin{pmatrix} M_1 = M_{xx} \\ M_2 = M_{xy} \\ M_3 = M_{yy} \\ M_4 = M_x \\ M_5 = M_y \\ M_6 = M \end{pmatrix} = \begin{pmatrix} k_{11} & k_{12} & k_{13} & & & \\ k_{12} & k_{22} & k_{23} & & & \\ k_{13} & k_{23} & k_{33} & & & \\ & & & k_{44} & k_{45} & \\ & & & k_{45} & k_{55} & \\ & & & & & k_{66} \end{pmatrix} \begin{pmatrix} \varepsilon_1 = u_{xx} \\ \varepsilon_2 = 2u_{xy} \\ \varepsilon_3 = u_{yy} \\ \varepsilon_4 = u_x \\ \varepsilon_5 = u_y \\ \varepsilon_6 = u \end{pmatrix} \quad (2-15)$$

内部エネルギーを,

$$U = \frac{1}{2} k_{mn}\varepsilon_m\varepsilon_n \geq 0 \quad (2-16)$$

とすると, Green 積分式 (2-8) にみられるように, 領域 S の内部で変分の対象となる変数は u のみであるのに対して, 境界 s の上では u, u_x, u_y の 3 個である. したがって, 領域内では次の 1 個の方程式となる.

$$u = \bar{u} \quad \text{または} \quad L(u) = f \quad (2-17)$$

境界上では 3 個の条件式

$$\left. \begin{aligned} u_x = \bar{u}_x \quad \text{または} \quad F_x = \bar{F}_x \\ u_y = \bar{u}_y \quad \text{または} \quad F_y = \bar{F}_y \\ u = \bar{u} \quad \text{または} \quad F = \bar{F} \end{aligned} \right\} \quad (2-18)$$

が可能な支配方程式として採用される. この中で, 領域内で u が拘束されている場合は, 考慮外であるから, 式 (2-17) の第 1 式は除かれる. いま外部エネルギーを

$$W = \int_S (\bar{F}_x u_x + \bar{F}_y u_y + \bar{F}_u) ds + \iint_S f u dx dy \quad (2-19)$$

によって定義すれば, 次の変分原理が満たされる.

$$\begin{aligned} \delta(U - W) = \int_S \{ (F_x - \bar{F}_x) \delta u_x + (F_y - \bar{F}_y) \delta u_y + (F - \bar{F}) \delta u \} ds \\ + \iint_S (L - f) \delta u dx dy = 0 \end{aligned} \quad (2-20)$$

いま s - t 方向へ座標変換を

$$dx = -(l_y ds + l_x dt), \quad dy = l_x ds - l_y dt \quad (2-21)$$

$$ds = -l_y dx + l_x dy, \quad dt = -(l_x dx + l_y dy) \quad (2-22)$$

とすると, これによる微分の変換は次式に示される.

$$\frac{\partial}{\partial x} = -\left(l_y \frac{\partial}{\partial s} + l_x \frac{\partial}{\partial t} \right), \quad \frac{\partial}{\partial y} = l_x \frac{\partial}{\partial s} - l_y \frac{\partial}{\partial t}, \quad (2-23)$$

$$\frac{\partial}{\partial s} = -l_y \frac{\partial}{\partial x} + l_x \frac{\partial}{\partial y}, \quad \frac{\partial}{\partial t} = -\left(l_x \frac{\partial}{\partial x} + l_y \frac{\partial}{\partial y} \right) \quad (2-24)$$

なお, 境界線の曲率半径を ρ とするとき,

$$\rho d\theta = ds \quad (2-25)$$

により,

$$\frac{\partial l_x}{\partial s} = -\frac{l_y}{\rho}, \quad \frac{\partial l_y}{\partial s} = \frac{l_x}{\rho}, \quad \frac{\partial l_x}{\partial t} = 0, \quad \frac{\partial l_y}{\partial t} = 0 \quad (2-26)$$

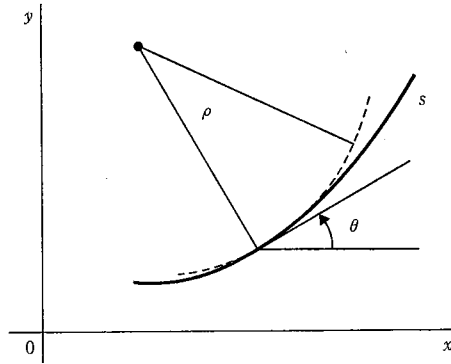


図 2 境界線の曲率半径

Fig. 2 Radius of curvature of boundary curve

の関係が導かれる。さらに曲率の変化まで考慮するときは、

$$\frac{\partial^2 l_x}{\partial s^2} = \frac{1}{\rho^2} \left(-l_x + l_y \frac{d\rho}{ds} \right), \quad \frac{\partial^2 l_y}{\partial s^2} = -\frac{1}{\rho^2} \left(l_y + l_x \frac{d\rho}{ds} \right) \quad (2-27)$$

を用いればよい。

以上の座標変換式により、次式を得る。

$$u_x = -(l_y u_s + l_x u_t), \quad u_y = l_x u_s - l_y u_t, \quad (2-28)$$

$$u_s = -l_y u_x + l_x u_y, \quad u_t = -(l_x u_x + l_y u_y), \quad (2-29)$$

$$F_x = -(l_y F_s + l_x F_t), \quad F_y = l_x F_s - l_y F_t, \quad (2-30)$$

$$F_s = -l_y F_x + l_x F_y = -l_x l_y M_{xx} + (l_x^2 - l_y^2) M_{xy} + l_x l_y M_{yy}, \quad (2-31)$$

$$F_t = -(l_x F_x + l_y F_y) = -(l_x^2 M_{xx} + 2l_x l_y M_{xy} + l_y^2 M_{yy}) \quad (2-32)$$

上の変換により、

$$F_x \delta u_x + F_y \delta u_y = F_s \delta u_s + F_t \delta u_t \quad (2-33)$$

が成り立つことは仮想仕事の原理からも当然期待されることである。したがって、式(2-8)の境界積分は、

$$\int_s (F_s \delta u_s + F_t \delta u_t + F \delta u) ds \quad (2-34)$$

に置き換えられる。ここで境界 s に沿う部分積分

$$\int_s F_s \delta u_s ds = - \int_s F_{s,s} \delta u ds \quad (2-35)$$

を行うことにより、式(2-34)は、

$$\int_s (F_t \delta u_t + F^* \delta u) ds, \quad F^* = F - F_{s,s} \quad (2-36)$$

に変形される。このように境界で自由に選べる独立なパラメータは u, u_t の2個となり、境界条件式も

$$\begin{aligned} u_t &= \bar{u}_t \quad \text{または} \quad F_t = \bar{F}_t \\ u &= \bar{u} \quad \text{または} \quad F^* = \bar{F}^* \end{aligned} \quad (2-37)$$

の2個に低減される。なお、弾性力学では境界条件を通常

$$\begin{aligned} \text{固定} & \quad u = \bar{u}, \quad u_t = \bar{u}_t \\ \text{支持} & \quad u = \bar{u}, \quad F_t = \bar{F}_t \\ \text{自由} & \quad F_t = \bar{F}_t, \quad F^* = \bar{F}^* \end{aligned} \quad (2-38)$$

の三つに分類している。

つぎに $F_{s,s}$ は、式(2-23)以下の曲率も考慮した変換式によって、

$$\begin{aligned}
 F_{s,s} = & l_x l_y (l_y M_{xx,x} - l_x M_{xx,y}) + (l_x^2 - l_y^2) (-l_y M_{xy,x} + l_x M_{xy,y}) \\
 & + l_x l_y (-l_y M_{yy,x} + l_x M_{yy,y}) \\
 & + \{(l_y^2 - l_x^2) M_{xx} - 4l_x l_y M_{xy} + (l_x^2 - l_y^2) M_{yy}\} / \rho
 \end{aligned}
 \tag{2-39}$$

と導かれる。このように $F_{s,s}$ が曲率に影響されることは注意を要することである。いま、 M_{xx}, \dots 等の曲線 s に沿う標準変形波長を λ とし、標準微分長を $l = \lambda/2\pi$ とするとき、 l と ρ の大きさの関係により種々の場合がありうる。曲線が直線に近く ρ が非常に大きいところでは曲率の影響は省略でき、反対に曲率が非常に大きいところ、すなわち曲線が急激に方向を変えているところでは、当然曲率の影響は大きくなる。この極限的な場合として折れ曲がりがある。ここでは曲率無限大、曲率半径ゼロとなり一種の特異点となる。したがって、ここでは数値計算上近似的な取り扱いをせざるをえない。考え方として下記の案が考えられる。

- 1) θ として θ_1 か θ_2 いずれかを優先させる。
- 2) $\theta = (\theta_1 + \theta_2)/2$ として平均傾斜角を用いる。
- 3) 折れ曲り点は無視して、ここには節点を置かない。
- 4) 隅点に s_1, s_2 上の点 c_1, c_2 を併置する。 $c_1, c_2 \rightarrow c$ 。

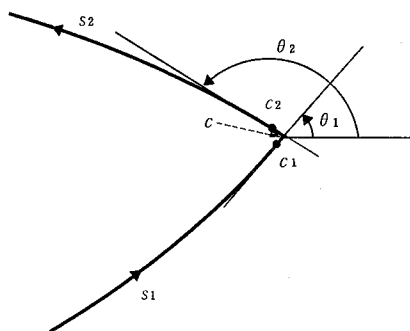


図 3 曲線の折れ曲がり

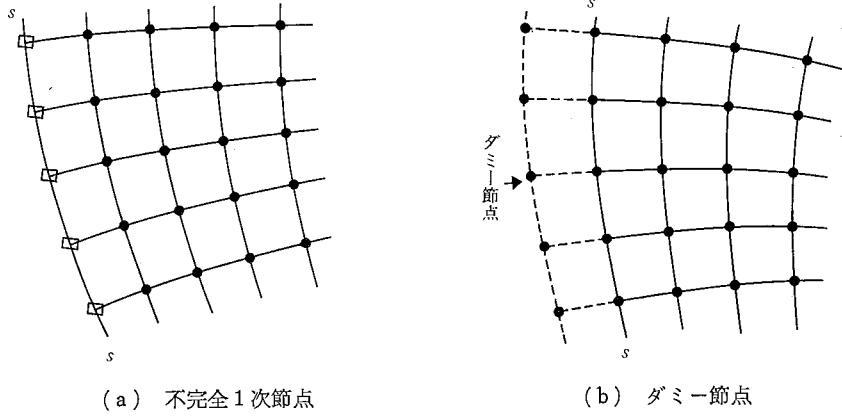
Fig. 3 Discontinuity of boundary slope

2.2 微分方程式の差分化

微分方程式の差分化を行うとき、境界に適合した曲線格子系を作り、その格子点に節点を置く。とくに境界上では、一つのパラメタ u, u_i をもつ不完全 1 次節点を置くことが必要である。さらにこの不完全 1 次節点は二つの 0 次節点の組合せによって代行させることができる。このように境界外に新たに設けられる 0 点節点をダミー節点と呼ぶこととする。差分化に当たりこのいずれの節点を用いてもよいが、後者の方が 0 次節点のみで統一的な取り扱いができるので便利である。

なお、節点の定義およびその記号は次のとおりである。

次数	パラメタ	記号
0	u	●
1	u, u_x, u_y	○
2	$u, u_x, u_y, u_{xx}, u_{xy}, u_{yy}$	◎
3	$u, u_x, u_y, u_{xx}, u_{xy}, u_{yy}, u_{xxx}, u_{xxy}, u_{xyy}, u_{yyy}$	◎◎
不完全 1	u, u_i	□
不完全 2	u, u_i, u_{ii}	◻
不完全 3	u, u_i, u_{iii}	◻◻



(a) 不完全1次節点

(b) ダミー節点

図4 節点配置

Fig. 4 Distribution of nodes

ここで、ダミー節点法による微分方程式の差分化を考える。
領域内、境界上で満たされる微分方程式を一般に、

$$\sum_1^{15} A_m a_m = F \tag{2-40}$$

とする。ただし、

$$\left. \begin{aligned} a_1 &= u, a_2 = \partial u / \partial x, a_3 = \partial u / \partial y, a_4 = \partial^2 u / \partial x^2, a_5 = \partial^2 u / \partial x \partial y, a_6 = \partial^2 u / \partial y^2, \\ a_7 &= \partial^3 u / \partial x^3, a_8 = \partial^3 u / \partial x^2 \partial y, a_9 = \partial^3 u / \partial x \partial y^2, a_{10} = \partial^3 u / \partial y^3, a_{11} = \partial^4 u / \partial x^4, \\ a_{12} &= \partial^4 u / \partial x^3 \partial y, a_{13} = \partial^4 u / \partial x^2 \partial y^2, a_{14} = \partial^4 u / \partial x \partial y^3, a_{15} = \partial^4 u / \partial y^4 \end{aligned} \right\} \tag{2-41}$$

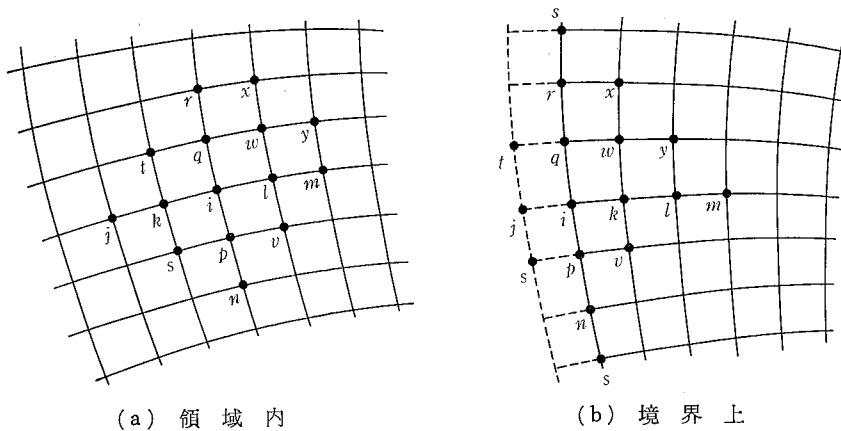
で、 A_m はその係数である。式の形は場所により異なるので、式番号 h の微分方程式を

$$\sum_1^{15} A_{mh} a_m = f_h \tag{2-42}$$

とする。ここで展開関数列を次のように定める。

$$\left. \begin{aligned} f_1 &= 1, f_2 = x, f_3 = y, f_4 = x^2/2, f_5 = xy, f_6 = y^2/2, f_7 = x^3/6, f_8 = x^2y/2, \\ f_9 &= xy^2/2, f_{10} = y^3/6, f_{11} = x^4/24, f_{12} = x^3y/6, f_{13} = x^2y^2/4, f_{14} = xy^3/6, \\ f_{15} &= y^4/24 \end{aligned} \right\} \tag{2-43}$$

差分家族構成は式(2-44)のとおりとする。



(a) 領域内

(b) 境界上

図5 差分家族内節点配置

Fig. 5 Node distribution of difference family in domain and on boundary.

$$1^i=i, 2^i=j, 3^i=k, 4^i=l, 5^i=m, 6^i=n, 7^i=p, 8^i=q, 9^i=r, \quad (2-44)$$

$$10^i=s, 11^i=t, 12^i=v, 13^i=w, 14^i=x, 15^i=y$$

ただし、頭符 i は差分家族番号, $1, 2, 3, \dots$ は家族内節点番号, i, j, k, \dots は全系節点番号を表す. 差分家族内節点配置は正則な図 5 に示すような配置とする.

差分家族局所座標:

$$\left. \begin{aligned} x_1^i &= 0, x_2^i = x_j - x_i, x_3^i = x_k - x_i, x_4^i = x_l - x_i, x_5^i = x_m - x_i, \\ x_6^i &= x_n - x_i, x_7^i = x_p - x_i, x_8^i = x_q - x_i, x_9^i = x_r - x_i, x_{10}^i = x_s - x_i \\ x_{11}^i &= x_t - x_i, x_{12}^i = x_v - x_i, x_{13}^i = x_w - x_i, x_{14}^i = x_x - x_i, x_{15}^i = x_y - x_i \end{aligned} \right\} \quad (2-45)$$

y_m^i についても同様.

差分家族内変数分布:

$$u^i = a_n^i f_n(x^i, y^i). \quad (2-46)$$

$$f_{mn}^i a_n^i = u_m^i, f_{mn}^i = f_n(x_m^i, y_m^i), u_m^i = u^i(x_m^i, y_m^i). \quad (2-47)$$

$$a_m^i = g_{mn}^i u_n^i, (g_{mn}^i \text{ は } f_{mn}^i \text{ の逆行列}). \quad (2-48)$$

上式を式 (2-34) に代入すると,

$$A_{mh} g_{mn}^i u_n^i = k_n^{hi} u_n^i = f_h \quad (2-49)$$

となる. ただし,

$$k_n^h = A_{mh} g_{mn}^i \quad (2-50)$$

差分家族構成 (2-44) により,

$$\left. \begin{aligned} k_1^h &= k_{hi}, k_2^h = k_{hj}, k_3^h = k_{hk}, k_4^h = k_{hl}, k_5^h = k_{hm}, \\ k_6^h &= k_{hn}, k_7^h = k_{hp}, k_8^h = k_{hq}, k_9^h = k_{hr}, k_{10}^h = k_{hs}, \\ k_{11}^h &= k_{ht}, k_{12}^h = k_{hv}, k_{13}^h = k_{hw}, k_{14}^h = k_{hx}, k_{15}^h = k_{hy} \end{aligned} \right\} \quad (2-51)$$

となる. ここで, h は式番号, i は差分家族番号を表す. 領域内節点では $h=i$ であるが, 境界上節点で h は i, j の二つの番号をとる. これは境界条件式は 2 個であることによるものである.

以上により, 全系連立方程式

$$k_{ij} u_j = f_i \quad (2-52)$$

を導入する.

なお, ここでは便宜上境界上節点 i に対応するダミー節点の節点番号を $j=i+1$ と約束する. したがって, 境界節点では k_n^i と $k_n^i = k_n^{i+1}$ が共存することとなる.

計算手順の概略を表すフロー・チャートを図 6 に示す.

2.3 平板の面外変形

平板の面外変位を $w(x, y)$ とするとき, 構成方程式は

$$M_{xx} = D(w_{xx} + \nu w_{yy}), M_{xy} = (1-\nu)Dw_{xy}, M_{yy} = D(\nu w_{xx} + w_{yy}) \quad (2-53)$$

によって表される. ここに $D = Eh^3/12(1-\nu^2)$ である.

つぎに Green 積分 (2-8) 以下により次式が導かれる.

$$F_x = D \{ l_x(w_{xx} + \nu w_{yy}) + l_y(1-\nu)w_{xy} \} \quad (2-54)$$

$$F_y = D \{ l_x(1-\nu)w_{xy} + l_y(\nu w_{xx} + w_{yy}) \} \quad (2-55)$$

$$F = -D \{ l_x(w_{xxx} + w_{xyy}) + l_y(w_{xxy} + w_{yyy}) \} \quad (2-56)$$

$$L = D(w_{xxxx} + 2w_{xxyy} + w_{yyyy}) \quad (2-57)$$

さらに, $s-t$ 座標系に変換する.

$$F_s = (1-\nu)D \{ -l_x l_y w_{xx} + (l_x^2 - l_y^2)w_{xy} + l_x l_y w_{yy} \} \quad (2-58)$$

$$F_t = -D \{ (l_x^2 + \nu l_y^2)w_{xx} + 2(1-\nu)l_x l_y w_{xy} + (\nu l_x^2 + l_y^2)w_{yy} \} \quad (2-59)$$

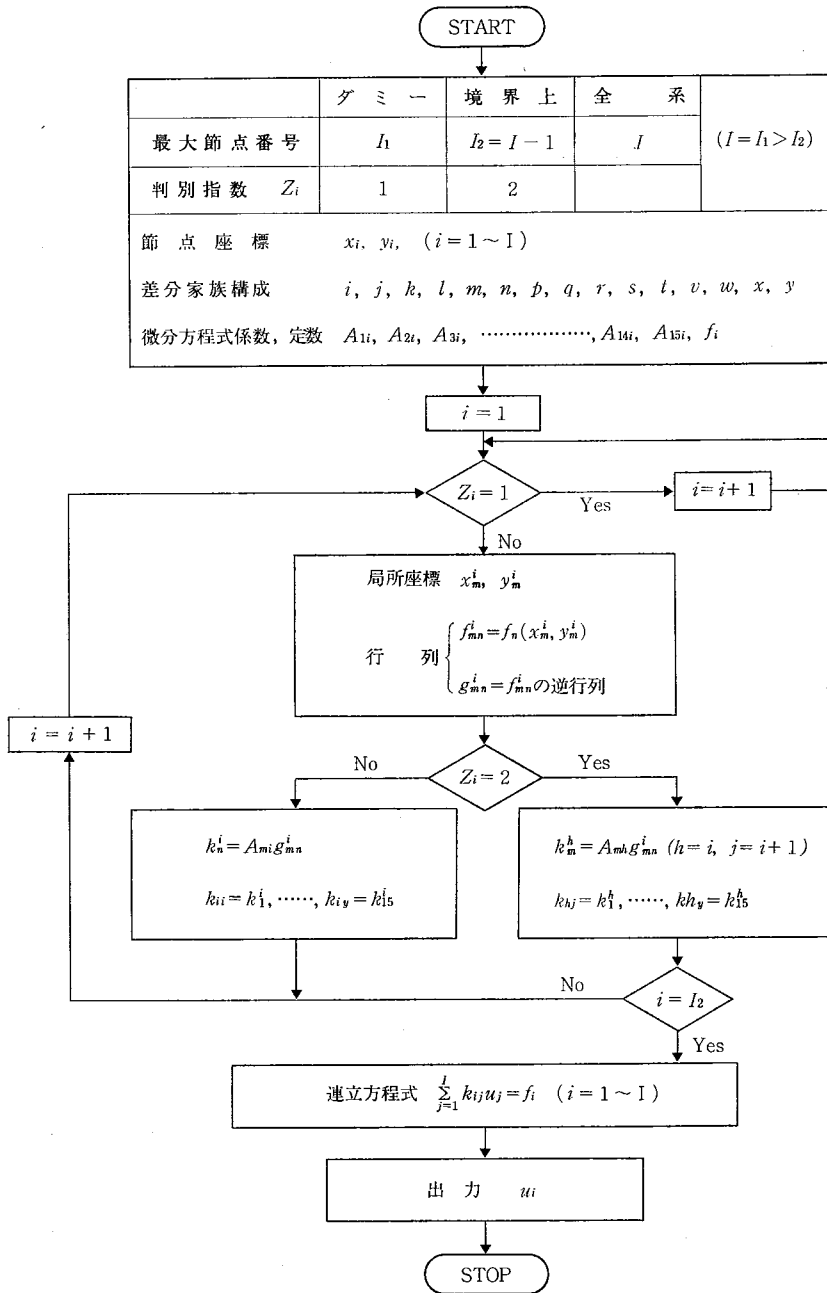


図 6 4階微分方程式計算手順のフロー・チャート
 Fig. 6 Calculation procedure of fourth order differential equation

$$F_{s,s} = (1-\nu)D[l_x l_y^2 w_{xxx} + (l_y^3 - 2l_x^2 l_y)w_{xxy} + (l_x^3 - 2l_x l_y^2)w_{xyy} + l_x^2 l_y w_{yyy} + \{(l_y^2 - l_x^2)w_{xx} - 4l_x l_y w_{xy} + (l_x^2 - l_y^2)w_{yy}\} / \rho] \quad (2-60)$$

$$F^* = F - F_{s,s} = -D[l_x \{1 + (1-\nu)l_y^2\} w_{xxx} + l_y \{1 + (1-\nu)l_y^2 - 2l_x^2\} w_{xxy} + l_x \{1 + (1-\nu)(l_x^2 - 2l_y^2)\} w_{xyy} + l_y \{1 + (1-\nu)l_x^2\} w_{yyy} + \{(1-\nu)/\rho\} \{(l_y^2 - l_x^2)w_{xx} - 4l_x l_y w_{xy} + (l_x^2 - l_y^2)w_{yy}\}] \quad (2-61)$$

$$w_s = -l_y w_x + l_x w_y \quad (2-62)$$

$$w_t = -(l_x w_x + l_y w_y) \quad (2-63)$$

たとえば、図7に示す矩形板については次のようになる。

	w_t	F_t	F^*
s_1	w_x	$-D(w_{xx} + \nu w_{yy})$	$D\{w_{xxx} + (2-\nu)w_{xyy}\}$
s_2	w_y	$-D(\nu w_{xx} + w_{yy})$	$D\{(2-\nu)w_{xxy} + w_{yyy}\}$
s_3	$-w_x$	$-D(w_{xx} + \nu w_{yy})$	$-D\{w_{xxx} + (2-\nu)w_{xyy}\}$
s_4	$-w_y$	$-D(\nu w_{xx} + w_{yy})$	$-D\{(2-\nu)w_{xxy} + w_{yyy}\}$

隅点 c_1, c_2, c_3, c_4 では、 w_t, F_t は隅点を挟む両辺の値を用い、 F^* は $F_{s,s}$ を省略し、境界線の傾きは両辺の平均を用いて、

$$\begin{aligned}
 c_1 & (D/2)(w_{xxx} + w_{xxy} + w_{xyy} + w_{yyy}) \\
 c_2 & (D/2)(-w_{xxx} + w_{xxy} - w_{xyy} + w_{yyy}) \\
 c_3 & -(D/2)(w_{xxx} + w_{xxy} + w_{xyy} + w_{yyy}) \\
 c_4 & (D/2)(w_{xxx} - w_{xxy} + w_{xyy} - w_{yyy})
 \end{aligned}$$

とする。

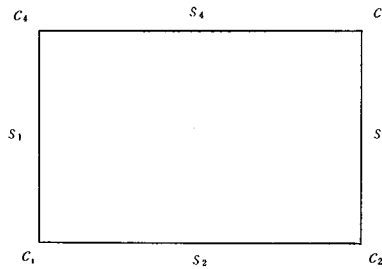


図7 矩形平板

Fig. 7 Rectangular plate

一般に任意形状、境界条件の平板について領域内、境界上では下記支配方程式が満たされる。

面外力を $p(\text{kg/cm}^2)$ として、平衡方程式は、

$$D\left(\frac{\partial^4 w}{\partial x^4} + 2\frac{\partial^4 w}{\partial x^2 \partial y^2} + \frac{\partial^4 w}{\partial y^4}\right) = p \tag{2-64}$$

となる。ここで、境界条件式は

$$\left\{ \begin{array}{l} \text{固定} \quad w = \bar{w}, \quad w_t = \bar{w}_t \end{array} \right. \tag{2-65}$$

$$\left\{ \begin{array}{l} \text{支持} \quad w = \bar{w}, \quad F_t = \bar{F}_t \end{array} \right. \tag{2-66}$$

$$\left\{ \begin{array}{l} \text{自由} \quad F_t = \bar{F}_t, \quad F^* = \bar{F}^* \end{array} \right. \tag{2-67}$$

である。

これらの微分方程式は任意節点配置差分法により差分化するとき、その係数および定数項は次のように与えられる。

平衡方程式：

$$A_{11} = 1, A_{13} = 2, A_{15} = 1, f = p/D \tag{2-68}$$

固定境界条件式：

$$A_1 = 1, f = \bar{w} \tag{2-69}$$

$$A_2 = l_x, A_3 = l_y, f = -\bar{w}_i \quad (2-70)$$

支持境界条件式:

$$A_1 = 1, f = \bar{w} \quad (2-71)$$

$$A_4 = l_x^2 + \nu l_y^2, A_5 = 2(1-\nu)l_x l_y, A_6 = \nu l_x^2 + l_y^2, f = -\bar{F}_i/D \quad (2-72)$$

自由境界条件式:

$$A_4 = l_x^2 + \nu l_y^2, A_5 = 2(1-\nu)l_x l_y, A_6 = \nu l_x^2 + l_y^2, f = -\bar{F}_i/D \quad (2-73)$$

$$\left. \begin{aligned} A_7 &= l_x \{1 + (1-\nu)l_y^2\}, A_8 = l_y \{1 + (1-\nu)(l_y^2 - 2l_x^2)\}, \\ A_9 &= l_x \{1 + (1-\nu)(l_x^2 - 2l_y^2)\}, A_{10} = l_y \{1 + (1-\nu)l_x^2\}, \\ A_4 &= (1-\nu)(l_y^2 - l_x^2)/\rho, A_5 = -4(1-\nu)l_x l_y/\rho, A_6 = (1-\nu)(l_x^2 - l_y^2)/\rho \end{aligned} \right\} \quad (2-74)$$

3. 2次元, 6階微分方程式

ここでは, Green 積分により導入される形の微分方程式のみを取り扱うこととする. 変数を $u(x, y)$ として, その歪関数を

$$\left. \begin{aligned} \varepsilon_{xxx} &= u_{xxx}, \varepsilon_{xxy} = 3u_{xxy}, \varepsilon_{xyy} = 3u_{xyy}, \varepsilon_{yyy} = u_{yyy}, \\ \varepsilon_{xx} &= u_{xx}, \varepsilon_{xy} = 2u_{xy}, \varepsilon_{yy} = u_{yy}, \varepsilon_x = u_x, \varepsilon_y = u_y, \varepsilon = u \end{aligned} \right\} \quad (3-1)$$

とし, これに対応する作用関数を

$$M_{xxx}, M_{xxy}, M_{xyy}, M_{yyy}, M_{xx}, M_{xy}, M_{yy}, M_x, M_y, M \quad (3-2)$$

とし, 次の Green 積分を行う.

$$\begin{aligned} \delta U &= \iint_S (M_{xxx} \delta \varepsilon_{xxx} + M_{xxy} \delta \varepsilon_{xxy} + M_{xyy} \delta \varepsilon_{xyy} + M_{yyy} \delta \varepsilon_{yyy} + M_{xx} \delta \varepsilon_{xx} \\ &\quad + M_{xy} \delta \varepsilon_{xy} + M_{yy} \delta \varepsilon_{yy} + M_x \delta \varepsilon_x + M_y \delta \varepsilon_y + M \delta \varepsilon) dx dy \\ &= \int_S (F_{xx} \delta u_{xx} + 2F_{xy} \delta u_{xy} + F_{yy} \delta u_{yy} + F_x \delta u_x + F_y \delta u_y + F \delta u) ds \\ &\quad - \iint_S L \delta u dx dy \end{aligned} \quad (3-3)$$

ただし,

$$F_{xx} = l_x M_{xxx} + l_y M_{xxy} \quad (3-4)$$

$$F_{xy} = l_x M_{xxy} + l_y M_{xyy} \quad (3-5)$$

$$F_{yy} = l_x M_{xyy} + l_y M_{yyy} \quad (3-6)$$

$$F_x = l_x P_{xx} + l_y P_{xy} \quad (3-7)$$

$$F_y = l_x P_{xy} + l_y P_{yy} \quad (3-8)$$

$$F = l_x Q_x + l_y Q_y \quad (3-9)$$

である. なお P_{xx}, \dots, Q_y は次の定義による.

$$P_{xx} = - \left(\frac{\partial M_{xxx}}{\partial x} + \frac{\partial M_{xxy}}{\partial y} \right) + M_{xx} \quad (3-10)$$

$$P_{xy} = - \left(\frac{\partial M_{xxy}}{\partial x} + \frac{\partial M_{xyy}}{\partial y} \right) + M_{xy} \quad (3-11)$$

$$P_{yy} = - \left(\frac{\partial M_{xyy}}{\partial x} + \frac{\partial M_{yyy}}{\partial y} \right) + M_{yy} \quad (3-12)$$

$$Q_x = \frac{\partial^2 M_{xxx}}{\partial x^2} + 2 \frac{\partial^2 M_{xxy}}{\partial x \partial y} + \frac{\partial^2 M_{xyy}}{\partial y^2} - \left(\frac{\partial M_{xx}}{\partial x} + \frac{\partial M_{xy}}{\partial y} \right) + M_x \quad (3-13)$$

$$Q_y = \frac{\partial^2 M_{xxy}}{\partial x^2} + 2 \frac{\partial^2 M_{xyy}}{\partial x \partial y} + \frac{\partial^2 M_{yyy}}{\partial y^2} - \left(\frac{\partial M_{xy}}{\partial x} + \frac{\partial M_{yy}}{\partial y} \right) + M_y \quad (3-14)$$

$$L = \frac{\partial^3 M_{xxx}}{\partial x^3} + 3 \frac{\partial^3 M_{xxy}}{\partial x^2 \partial y} + 3 \frac{\partial^3 M_{xyy}}{\partial x \partial y^2} + \frac{\partial^3 M_{yyy}}{\partial y^3} - \left(\frac{\partial^2 M_{xx}}{\partial x^2} + 2 \frac{\partial^2 M_{xy}}{\partial x \partial y} + \frac{\partial^2 M_{yy}}{\partial y^2} \right) + \frac{\partial M_x}{\partial x} + \frac{\partial M_y}{\partial y} - M \quad (3-15)$$

構成方程式

$$M_m = k_{mn} \varepsilon_n \quad (3-16)$$

の係数行列は、正値対称

$$k_{mn} = k_{nm}, \quad k_{mn} \varepsilon_m \varepsilon_n > 0 \quad (3-17)$$

とする。

内部エネルギーは次式となる。

$$U = \frac{1}{2} k_{mn} \varepsilon_m \varepsilon_n > 0 \quad (3-18)$$

式(3-3)にみられるように、変分の対象となるパラメタは境界上で $u_{xx}, u_{xy}, u_{yy}, u_x, u_y, n$ の6個、領域内では u のみの1個である。したがって、境界上では6個の条件式

$$\left. \begin{aligned} u_{xx} &= \bar{u}_{xx} \text{ または } F_{xx} = \bar{F}_{xx} \\ u_{xy} &= \bar{u}_{xy} \text{ または } F_{xy} = \bar{F}_{xy} \\ u_{yy} &= \bar{u}_{yy} \text{ または } F_{yy} = \bar{F}_{yy} \\ u_x &= \bar{u}_x \text{ または } F_x = \bar{F}_x \\ u_y &= \bar{u}_y \text{ または } F_y = \bar{F}_y \\ u &= \bar{u} \text{ または } F = \bar{F} \end{aligned} \right\} \quad (3-19)$$

である。領域内では1個の微分方程式

$$L + f = 0$$

が存在する。このとき外部エネルギーは、

$$W = \int_S (\bar{F}_{xx} u_{xx} + 2\bar{F}_{xy} u_{xy} + \bar{F}_{yy} u_{yy} + \bar{F}_x u_x + \bar{F}_y u_y + \bar{F} u) ds + \iint_S f u dx dy \quad (3-20)$$

によって定義され、変分原理

$$\begin{aligned} \delta(U - W) = \int_S \{ (F_{xx} - \bar{F}_{xx}) \delta u_{xx} + 2(F_{xy} - \bar{F}_{xy}) \delta u_{xy} + (F_{yy} - \bar{F}_{yy}) \delta u_{yy} + (F_x - \bar{F}_x) \delta u_x \\ + (F_y - \bar{F}_y) \delta u_y + (F - \bar{F}) \delta u \} ds - \iint_S (L + f) \delta u dx dy = 0 \end{aligned} \quad (3-21)$$

が満たされる。

つぎに前節と同様に $s-t$ 座標に変換を行い、境界積分を

$$\int_S (F_{ss} \delta u_{ss} + 2F_{st} \delta u_{st} + F_{tt} \delta u_{tt} + F_s \delta u_s + F_t \delta u_t + F \delta u) ds \quad (3-22)$$

に変換する。ただし、簡単にするため境界線の曲率は省略してある。上式は部分積分により、さらに次のように変形される。

$$\int_S (F_{tt} \delta u_{tt} + F_t^* \delta u_t + F^* \delta u) ds \quad (3-23)$$

このように、境界上の節点パラメタは u, u_t, u_{tt} の3個で不完全2次節点によって代表され、さらにこの節点は図8に示されるように境界上に1個、境界外に2個の0次節点をもつダミー節点群で置き換えることができる。ただし、

$$F_t^* = F_t - 2F_{st,s} \quad (3-24)$$

$$F^* = F - F_{s,s} + F_{ss,ss} \quad (3-25)$$

である。したがって、境界条件式も次の3個

$$\begin{aligned} u_{ii} &= \bar{u}_{ii} \quad \text{または} \quad F_{ii} = \bar{F}_{ii} \\ u_i &= \bar{u}_i \quad \text{または} \quad F_i^* = \bar{F}_i^* \\ u &= \bar{u} \quad \text{または} \quad F^* = \bar{F}^* \end{aligned} \quad (3-26)$$

で与えられる。

3.1 変分原理

6個の境界条件式に対応する外部エネルギーおよび変分原理を式(3-20), (3-21)に示したが, 3個の境界条件の場合は

$$W = \int_S (\bar{F}_{ii} u_{ii} + \bar{F}_i^* u_i + \bar{F}^* u) ds + \iint_S f u dx dy \quad (3-27)$$

$$\begin{aligned} \delta(U - W) &= \int_S \{ (F_{ii} - \bar{F}_{ii}) \delta u_{ii} + (F_i^* - \bar{F}_i^*) \delta u_i + (F - \bar{F}^*) \delta u \} ds \\ &\quad - \iint_S (L + f) \delta u dx dy = 0 \end{aligned} \quad (3-28)$$

によって与えられる。これら二つの変分原理はいずれを用いてもよいが, 一般に差分法では式(3-28), 有限要素法では式(3-21)を用いる方が扱いやすいものと思われる。

なお, $x-y$ から $s-t$ 座標式への変換は次の式と,

$$u_s = -l_y u_x + l_x u_y \quad (3-29)$$

$$u_t = -(l_x u_x + l_y u_y) \quad (3-30)$$

$$u_{ss} = l_y^2 u_{xx} - 2l_x l_y u_{xy} + l_x^2 u_{yy} \quad (3-31)$$

$$u_{st} = l_x l_y u_{xx} + (-l_x^2 + l_y^2) u_{xy} - l_x l_y u_{yy} \quad (3-32)$$

$$u_{tt} = l_x^2 u_{xx} + 2l_x l_y u_{xy} + l_y^2 u_{yy} \quad (3-33)$$

$$F_s = -l_y F_x + l_x F_y = -l_x l_y P_{xx} + (l_x^2 - l_y^2) P_{xy} + l_x l_y P_{yy} \quad (3-34)$$

$$F_t = -(l_x F_x + l_y F_y) = -(l_x^2 P_{xx} + 2l_x l_y P_{xy} + l_y^2 P_{yy}) \quad (3-35)$$

$$\begin{aligned} F_{ss} &= l_y^2 F_{xx} - 2l_x l_y F_{xy} + l_x^2 F_{yy} \\ &= l_x l_y^2 M_{xxx} + l_y (l_y^2 - 2l_x^2) M_{xxy} + l_x (l_x^2 - 2l_y^2) M_{xyy} + l_x^2 l_y M_{yyy} \end{aligned} \quad (3-36)$$

$$\begin{aligned} F_{st} &= l_x l_y F_{xx} + (l_y^2 - l_x^2) F_{xy} - l_x l_y F_{yy} \\ &= l_x^2 l_y M_{xxx} + l_x (2l_y^2 - l_x^2) M_{xxy} + l_y (l_y^2 - 2l_x^2) M_{xyy} - l_x l_y^2 M_{yyy} \end{aligned} \quad (3-37)$$

$$\begin{aligned} F_{tt} &= l_x^2 F_{xx} + 2l_x l_y F_{xy} + l_y^2 F_{yy} \\ &= l_x^3 M_{xxx} + 3l_x^2 l_y M_{xxy} + 3l_x l_y^2 M_{xyy} + l_y^3 M_{yyy} \end{aligned} \quad (3-38)$$

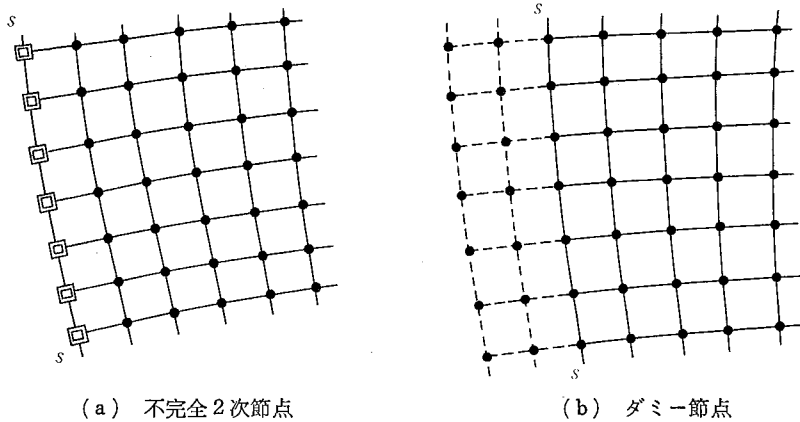


図8 節点配置

Fig. 8 Nodes distribution

次式から求められる。

$$F_{s,s} = \left(-l_y \frac{\partial}{\partial x} + l_x \frac{\partial}{\partial y} \right) \{ -l_x l_y P_{xx} + (l_x^2 - l_y^2) P_{xy} + l_x l_y P_{yy} \} \quad (3-39)$$

$$F_{ss,ss} = \left(l_y^2 \frac{\partial^2}{\partial x^2} - 2l_x l_y \frac{\partial^2}{\partial x \partial y} + l_x^2 \frac{\partial^2}{\partial y^2} \right) \{ l_x l_y^2 M_{xxx} + l_y(l_y^2 - 2l_x^2) M_{xxy} + l_x(l_x^2 - 2l_y^2) M_{xyy} + l_x^2 l_y M_{yyy} \} \quad (3-40)$$

$$F_{s1,s} = \left(-l_y \frac{\partial}{\partial x} + l_x \frac{\partial}{\partial y} \right) \{ l_x^2 l_y M_{xxx} + l_x(2l_y^2 - l_x^2) M_{xxy} + l_y(l_y^2 - 2l_x^2) M_{xyy} - l_x l_y^2 M_{yyy} \} \quad (3-41)$$

3.2 任意節点配置差分法による解析

ここでは前述の理由により、タミー節点法を用いることとする。Teilor 級数展開法を用いるためには、6階の微分方程式を少なくとも6次多項式によることが必要で、その項数は28である。たとえば原点における展開式を

$$u(x, y) = \sum_1^{28} a_n f_n(x, y) \quad (3-42)$$

とする。ここに係数 a_n は、

$$\left. \begin{aligned} a_1 &= u, a_2 = \partial u / \partial x, a_3 = \partial u / \partial y, a_4 = \partial^2 u / \partial x^2, a_5 = \partial^2 u / \partial x \partial y, \\ a_6 &= \partial^2 u / \partial y^2, a_7 = \partial^3 u / \partial x^3, a_8 = \partial^3 u / \partial x^2 \partial y, a_9 = \partial^3 u / \partial x \partial y^2, \\ a_{10} &= \partial^3 u / \partial y^3, a_{11} = \partial^4 u / \partial x^4, a_{12} = \partial^4 u / \partial x^3 \partial y, a_{13} = \partial^4 u / \partial x^2 \partial y^2, \\ a_{14} &= \partial^4 u / \partial x \partial y^3, a_{15} = \partial^4 u / \partial y^4, a_{16} = \partial^5 u / \partial x^5, a_{17} = \partial^5 u / \partial x^4 \partial y, \\ a_{18} &= \partial^5 u / \partial x^3 \partial y^2, a_{19} = \partial^5 u / \partial x^2 \partial y^3, a_{20} = \partial^5 u / \partial x \partial y^4, a_{21} = \partial^5 u / \partial y^5, \\ a_{22} &= \partial^6 u / \partial x^6, a_{23} = \partial^6 u / \partial x^5 \partial y, a_{24} = \partial^6 u / \partial x^4 \partial y^2, a_{25} = \partial^6 u / \partial x^3 \partial y^3, \\ a_{26} &= \partial^6 u / \partial x^2 \partial y^4, a_{27} = \partial^6 u / \partial x \partial y^5, a_{28} = \partial^6 u / \partial y^6 \end{aligned} \right\} \quad (3-43)$$

の原点値で、関数列 f_u は次のように定義される。

$$\left. \begin{aligned} f_1 &= 1, f_2 = x, f_3 = y, f_4 = x^2/2, f_5 = xy, f_6 = y^2/2, f_7 = x^3/6, \\ f_8 &= x^2 y/2, f_9 = x y^2/2, f_{10} = y^3/6, f_{11} = x^4/24, f_{12} = x^3 y/6, \\ f_{13} &= x^2 y^2/4, f_{14} = x y^3/6, f_{15} = y^4/24, f_{16} = x^5/120, f_{17} = x^4 y/24, \\ f_{18} &= x^3 y^2/12, f_{19} = x^2 y^3/12, f_{20} = x y^4/24, f_{21} = y^5/120, f_{22} = x^6/720, \\ f_{23} &= x^5 y/120, f_{24} = x^4 y^2/48, f_{25} = x^3 y^3/36, f_{26} = x^2 y^4/48, \\ f_{27} &= x y^5/120, f_{28} = y^6/720 \end{aligned} \right\} \quad (3-44)$$

3.3 差分家族節点配置

差分家族は差分評価節点 i を含み 28 個の節点によって構成される。当然その配置は正則でなければならない。その家族構成は、

$$1^i = i, 2^i = j, 3^i = k, 4^i = l, \dots, 28^i = \dots \quad (3-45)$$

によって与えられるものとする。図 9 に領域内および境界上における節点配置を示す。

前節と同様に差分家族ごとに、

$$f_{mn}^i = f_n(x_m^i, y_m^i) \quad (3-46)$$

$$y_{nm}^i = f_{mn}^i \text{ の逆行列} \quad (3-47)$$

を求める。つぎに、式番号 h , 差分家族番号 i の式を

$$A_{mh} a_m^i = f_h \quad (3-48)$$

とし、上式に、

$$a_m^i = g_{mn}^i u_n^i \quad (3-49)$$

を代入して

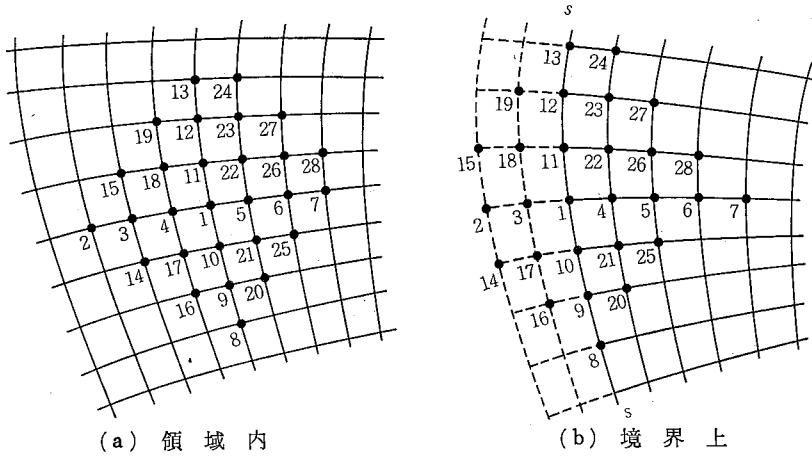


図 9 差分家族節点配置
Fig. 9 Node distribution of difference family

$$A_{mh}a_m^i = A_{mh}g_{mn}^i u_n^i = k_n^h u_n^i = f_h \quad (3-50)$$

をつくる。ここで、

$$k_n^h = A_{mh}g_{mn}^i \quad (3-51)$$

は係数行列を表し、差分家族構成 (3-45) により全系連立方程式

$$k_{i,j} u_j = f_i \quad (3-52)$$

が導かれる。この計算では g_{mn}^i を算出するため、 28×28 行列の逆行列計算を各差分家族ごとに行うことが必要である。また、領域内節点では $h=i$ のみであるが、境界上節点では $h=i, j, k$ の 3 個の番号をとらなければならない。なお $j=i+1, k=i+2$ とする。

3.4 有限要素法による解析

ここで取り扱う式は 6 階の微分方程式であるから、クラス C (2 次節点までを含む) 要素を用いることが必要である。たとえば、図 10 に示す 28 個のパラメタの要素を利用すればよい。

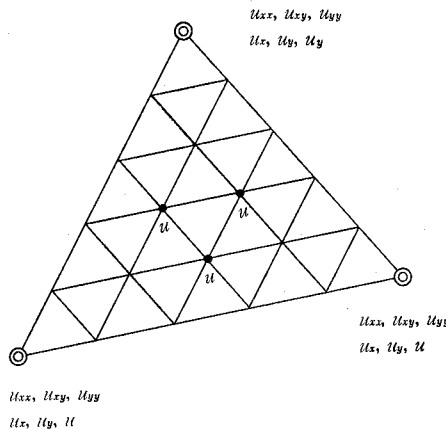


図 10 6 節点 5 次 C 要素
Fig. 10 Six nodes, five order, Class C element

4. 2次元, 8階微分方程式

変数 $u(x, y)$ の歪関数を

$$\epsilon_{xxxx} = u_{xxxx}, \epsilon_{xxxx} = 4u_{xxxx}, \epsilon_{xyxy} = 6u_{xyxy}, \epsilon_{yyyy} = 4u_{yyyy}, \quad |$$

$$\left. \begin{aligned} \epsilon_{yyyy} &= u_{yyyy}, \epsilon_{xxx} = u_{xxx}, \epsilon_{xxy} = 3u_{xxy}, \epsilon_{xyy} = 3u_{xyy}, \\ \epsilon_{yyy} &= u_{yyy}, \epsilon_{xx} = u_{xx}, \epsilon_{xy} = 2u_{xy}, \epsilon_{yy} = u_{yy}, \\ \epsilon_x &= u_x, \epsilon_y = u_y, \epsilon = u \end{aligned} \right\} \quad (4-1)$$

とし、これに対応する応力関数を

$$M_{xxxx}, M_{xxxxy}, M_{xxyyy}, \dots, M_x, M_y, M \quad (4-2)$$

として、次の Green 積分を行う。

$$\begin{aligned} \delta U &= \iint_S (M_{xxxx} \delta \epsilon_{xxxx} + M_{xxxxy} \delta \epsilon_{xxxxy} + \dots + M \delta \epsilon) dx dy \\ &= \int_S (F_{xxx} \delta u_{xxx} + 3F_{xxy} \delta u_{xxy} + 3F_{xyy} \delta u_{xyy} + F_{yyy} \delta u_{yyy} \\ &\quad + F_{xx} \delta u_{xx} + 2F_{xy} \delta u_{xy} + F_{yy} \delta u_{yy} + F_x \delta u_x + F_y \delta u_y + F \delta u) ds \\ &\quad + \iint_S L \delta u dx dy \end{aligned} \quad (4-3)$$

ただし、

$$F_{xxx} = l_x M_{xxxx} + l_y M_{xxxxy} \quad (4-4)$$

$$F_{xxy} = l_x M_{xxxxy} + l_y M_{xxyyy} \quad (4-5)$$

$$F_{xyy} = l_x M_{xxyyy} + l_y M_{xyyyy} \quad (4-6)$$

$$F_{yyy} = l_x M_{xyyyy} + l_y M_{yyyyy} \quad (4-7)$$

$$F_{xx} = l_x P_{xxx} + l_y P_{xxy} \quad (4-8)$$

$$F_{xy} = l_x P_{xxy} + l_y P_{xyy} \quad (4-9)$$

$$F_{yy} = l_x P_{xyy} + l_y P_{yyy} \quad (4-10)$$

$$E_x = l_x Q_{xx} + l_y Q_{xy} \quad (4-11)$$

$$F_y = l_x Q_{xy} + l_y Q_{yy} \quad (4-12)$$

$$F = l_x R_x + l_y R_y \quad (4-13)$$

である。なお、

$$P_{xxx} = -\left(\frac{\partial M_{xxxx}}{\partial x} + \frac{\partial M_{xxxxy}}{\partial y}\right) + M_{xxx} \quad (4-14)$$

$$P_{xxy} = -\left(\frac{\partial M_{xxxxy}}{\partial x} + \frac{\partial M_{xxyyy}}{\partial y}\right) + M_{xxy} \quad (4-15)$$

$$P_{xyy} = -\left(\frac{\partial M_{xxyyy}}{\partial x} + \frac{\partial M_{xyyyy}}{\partial y}\right) + M_{xyy} \quad (4-16)$$

$$P_{yyy} = -\left(\frac{\partial M_{xyyyy}}{\partial x} + \frac{\partial M_{yyyyy}}{\partial y}\right) + M_{yyy} \quad (4-17)$$

$$Q_{xx} = \frac{\partial^2 M_{xxxx}}{\partial x^2} + 2 \frac{\partial^2 M_{xxxxy}}{\partial x \partial y} + \frac{\partial^2 M_{xxyyy}}{\partial y^2} - \left(\frac{\partial M_{xxx}}{\partial x} + \frac{\partial M_{xxy}}{\partial y}\right) + M_{xx}, \quad (4-18)$$

$$Q_{xy} = \frac{\partial^2 M_{xxxxy}}{\partial x^2} + 2 \frac{\partial^2 M_{xxyyy}}{\partial x \partial y} + \frac{\partial^2 M_{xyyyy}}{\partial y^2} - \left(\frac{\partial M_{xxy}}{\partial x} + \frac{\partial M_{xyy}}{\partial y}\right) + M_{xy}, \quad (4-19)$$

$$Q_{yy} = \frac{\partial^2 M_{xxyyy}}{\partial x^2} + 2 \frac{\partial^2 M_{xyyyy}}{\partial x \partial y} + \frac{\partial^2 M_{yyyyy}}{\partial y^2} - \left(\frac{\partial M_{xyy}}{\partial x} + \frac{\partial M_{yyy}}{\partial y}\right) + M_{yy}, \quad (4-20)$$

$$\begin{aligned} R_x &= -\left(\frac{\partial^3 M_{xxxx}}{\partial x^3} + 3 \frac{\partial^3 M_{xxxxy}}{\partial x^2 \partial y} + 3 \frac{\partial^3 M_{xxyyy}}{\partial x \partial y^2} + \frac{\partial^3 M_{xyyyy}}{\partial y^3}\right) \\ &\quad + \frac{\partial^2 M_{xxx}}{\partial x^2} + 2 \frac{\partial^2 M_{xxy}}{\partial x \partial y} + \frac{\partial^2 M_{xyy}}{\partial y^2} - \left(\frac{\partial M_{xx}}{\partial x} + \frac{\partial M_{xy}}{\partial y}\right) + M_x \end{aligned} \quad (4-21)$$

$$R_y = -\left(\frac{\partial^3 M_{xxxxy}}{\partial x^3} + 3 \frac{\partial^3 M_{xxyyy}}{\partial x^2 \partial y} + 3 \frac{\partial^3 M_{xyyyy}}{\partial x \partial y^2} + \frac{\partial^3 M_{yyyyy}}{\partial y^3}\right)$$

$$+\frac{\partial^2 M_{xx}}{\partial x^2}+2\frac{\partial^2 M_{xy}}{\partial x\partial y}+\frac{\partial^2 M_{yy}}{\partial y^2}-\left(\frac{\partial M_{xy}}{\partial x}+\frac{\partial M_{yy}}{\partial y}\right)+M_y \quad (4-22)$$

$$L=\frac{\partial^4 M_{xxxx}}{\partial x^4}+4\frac{\partial^4 M_{xxxy}}{\partial x^3\partial y}+6\frac{\partial^4 M_{xxyy}}{\partial x^2\partial y^2}+4\frac{\partial^4 M_{xyyy}}{\partial x\partial y^3}+\frac{\partial^4 M_{yyyy}}{\partial y^4}$$

$$-\left(\frac{\partial^3 M_{xxx}}{\partial x^3}+3\frac{\partial^3 M_{xxy}}{\partial x^2\partial y}+3\frac{\partial^3 M_{xyy}}{\partial x\partial y^2}+\frac{\partial^3 M_{yyy}}{\partial y^3}\right)$$

$$+\frac{\partial^2 M_{xx}}{\partial x^2}+2\frac{\partial^2 M_{xy}}{\partial x\partial y}+\frac{\partial^2 M_{yy}}{\partial y^2}-\left(\frac{\partial M_x}{\partial x}+\frac{\partial M_y}{\partial y}\right)+M \quad (4-23)$$

である。

式(4-3)にみられるように、境界上における変分の対象となるパラメタは u_{xxx}, u_{xxy}, \dots u の 10 個、領域内では u のみの 1 個である。したがって、境界上では 10 個の式、領域内では 1 個の式が存在する。さらに、式(4-3)における境界積分は次のように変形される。

$$\int_s (F_{sss}\delta u_{sss}+3F_{ss}\delta u_{ssi}+3F_{sii}\delta u_{sii}+F_{iii}\delta u_{iii}+F_{ss}\delta u_{ss}$$

$$+2F_{si}\delta u_{si}+F_{ii}\delta u_{ii}+F_s\delta u_s+F_i\delta u_i+F\delta u)ds \quad (4-24)$$

上式はさらに部分積分により、

$$\int_s (F_{iii}\delta u_{iii}+F_{ii}^*\delta u_{ii}+F_i^*\delta u_i+F^*\delta u)ds \quad (4-25)$$

に変形される。ただし、

$$F_{ii}^*=F_{ii}-3F_{sii,s} \quad (4-26)$$

$$F_i^*=F_i-2F_{si,s}+3F_{sse,ss} \quad (4-27)$$

$$F^*=F-F_{s,s}+F_{ss,ss}-F_{sss,sss} \quad (4-28)$$

で境界条件式は次の四つとなる。

$$u_{iii}=\bar{u}_{iii} \text{ または } F_{iii}=\bar{F}_{iii}$$

$$u_{ii}=\bar{u}_{ii} \text{ または } F_{ii}^*=\bar{F}_{ii}^*$$

$$u_i=\bar{u}_i \text{ または } F_i^*=\bar{F}_i^*$$

$$u=\bar{u} \text{ または } F=F^* \quad (4-29)$$

領域内では次の微分方程式が満たされる。

$$L-f=0 \quad (4-30)$$

1) 変分原理

外部エネルギーを

$$W=\int_s (\bar{F}_{iii}u_{iii}+\bar{F}_{ii}^*u_{ii}+\bar{F}_i^*u_i+\bar{F}^*u)ds+\iint_s fudxdy \quad (4-31)$$

により定義するとき、式(4-29)、(4-30)により変分原理が導かれる。

$$\delta(U-W)=\int_s \{(F_{iii}-\bar{F}_{iii})\delta u_{iii}+(F_{ii}^*-\bar{F}_{ii}^*)\delta u_{ii}+(F_i^*-\bar{F}_i^*)\delta u_i$$

$$+(F^*-\bar{F}^*)\delta u\} ds+\iint_s (L-f)\delta udx dy=0 \quad (4-32)$$

2) 座標変換

$u_s, u_i, u_{ss}, u_{si}, u_{ii}$ は式(3-29)~(3-33)と同様に次のように座標変換できる。

$$u_{sss}=-l_y^3u_{xxx}+3l_xl_y^2u_{xxy}-3l_x^2l_yu_{xyy}+l_x^3u_{yyy} \quad (4-33)$$

$$u_{ssi}=-l_xl_y^2u_{xxx}+l_y(2l_x^2-l_y^2)u_{xxy}+l_x(2l_y^2-l_x^2)u_{xyy}-l_x^2l_yu_{yyy} \quad (4-34)$$

$$u_{sii}=-l_x^2l_yu_{xxx}+l_x(l_x^2-2l_y^2)u_{xxy}+l_y(2l_x^2-l_y^2)u_{xyy}+l_xl_y^2u_{yyy} \quad (4-35)$$

$$u_{iii}=-l_x^3u_{xxx}+3l_x^2l_yu_{xxy}+3l_xl_y^2u_{xyy}+l_y^3u_{yyy} \quad (4-36)$$

$$F_s = -l_y F_x + l_x L_y = -l_x l_y Q_{xx} + (l_x^2 - l_y^2) Q_{xy} + l_x l_y Q_{yy} \quad (4-37)$$

$$F_t = -(l_x F_x + l_y F_y) = -(l_x^2 Q_{xx} + 2l_x l_y Q_{xy} + l_y^2 Q_{yy}) \quad (4-38)$$

$$\begin{aligned} F_{ss} &= l_y^2 F_{xx} - 2l_x l_y F_{xy} + l_x^2 F_{yy} \\ &= l_x l_y^2 P_{xxx} + l_y(l_y^2 - 2l_x^2) P_{xxy} + l_x(l_x^2 - 2l_y^2) P_{xyy} + l_x^2 l_y P_{yyy} \end{aligned} \quad (4-39)$$

$$\begin{aligned} F_{st} &= l_x l_y F_{xx} + (l_y^2 - l_x^2) F_{xy} - l_x l_y F_{yy} \\ &= l_x^2 l_y P_{xxx} + l_x(2l_y^2 - l_x^2) P_{xxy} + l_y(l_y^2 - 2l_x^2) P_{xyy} - l_x l_y^2 P_{yyy} \end{aligned} \quad (4-40)$$

$$\begin{aligned} F_{tt} &= l_x^2 F_{xx} + 2l_x l_y F_{xy} + l_y^2 F_{yy} \\ &= l_x^3 P_{xxx} + 3l_x^2 l_y P_{xxy} + 3l_x l_y^2 P_{xyy} + l_y^3 P_{yyy} \end{aligned} \quad (4-41)$$

$$\begin{aligned} F_{sss} &= -l_y^3 F_{xxx} + 3l_x l_y^2 F_{xxy} - 3l_x^2 l_y F_{xyy} + l_x^3 F_{yyy} \\ &= -l_x l_y^3 M_{xxxx} + l_y^2(3l_x^2 - l_y^2) M_{xxxy} + 3l_x l_y(l_y^2 - 3l_x^2) M_{xxyy} \\ &\quad + l_x^2(l_x^2 - 3l_y^2) M_{xyyy} + l_x^3 l_y M_{yyyy} \end{aligned} \quad (4-42)$$

$$\begin{aligned} F_{sst} &= -l_x l_y^2 F_{xxx} + l_y(2l_x^2 - l_y^2) F_{xxy} + l_x(2l_y^2 - l_x^2) F_{xyy} - l_x^2 l_y F_{yyy} \\ &= -l_x^2 l_y^2 M_{xxxx} + 2l_x l_y(l_x^2 - l_y^2) M_{xxxy} + (-l_x^4 + 4l_x^2 l_y^2 - l_y^4) M_{xxyy} \\ &\quad + 2l_x l_y(l_y^2 - l_x^2) M_{xyyy} + l_x^2 l_y^2 M_{yyyy} \end{aligned} \quad (4-43)$$

$$\begin{aligned} F_{stt} &= -l_x^2 l_y F_{xxx} + l_x(l_x^2 - 2l_y^2) F_{xxy} + l_y(2l_x^2 - l_y^2) F_{xyy} + l_x l_y^2 F_{yyy} \\ &= -l_x^3 M_{xxxx} + l_x^2(l_x^2 - 3l_y^2) M_{xxxy} + 3l_x l_y(l_x^2 - l_y^2) M_{xxyy} \\ &\quad + l_y^2(3l_x^2 - l_y^2) M_{xyyy} + l_x l_y^3 M_{yyyy} \end{aligned} \quad (4-44)$$

$$\begin{aligned} F_{ttt} &= -(l_x^3 F_{xxx} + 3l_x^2 l_y F_{xxy} + 3l_x l_y^2 F_{xyy} + l_y^3 F_{yyy}) \\ &= -(l_x^4 M_{xxxx} + 4l_x^3 l_y M_{xxxy} + 6l_x^2 l_y^2 M_{xxyy} \\ &\quad + 4l_x l_y^3 M_{xyyy} + l_y^4 M_{yyyy}) \end{aligned} \quad (4-45)$$

$$F_{s,s} = \left(-l_y \frac{\partial}{\partial x} + l_x \frac{\partial}{\partial y} \right) \{ -l_x l_y Q_{xx} + (l_x^2 - l_y^2) Q_{xy} + l_x l_y Q_{yy} \} \quad (4-46)$$

$$\begin{aligned} F_{st,s} &= \left(-l_y \frac{\partial}{\partial x} + l_x \frac{\partial}{\partial y} \right) \{ l_x^2 l_y P_{xxx} + l_x(2l_y^2 - l_x^2) P_{xxy} \\ &\quad + l_y(l_y^2 - 2l_x^2) P_{xyy} - l_x l_y^2 P_{yyy} \} \end{aligned} \quad (4-47)$$

$$\begin{aligned} F_{stt,s} &= \left(-l_y \frac{\partial}{\partial x} + l_x \frac{\partial}{\partial y} \right) \{ -l_x^3 l_y M_{xxxx} + l_x^2(l_x^2 - 3l_y^2) M_{xxxy} \\ &\quad + 3l_x l_y(l_x^2 - l_y^2) M_{xxyy} + l_y^2(3l_x^2 - l_y^2) M_{xyyy} + l_x l_y^3 M_{yyyy} \} \end{aligned} \quad (4-48)$$

$$\begin{aligned} F_{ss,ss} &= \left(l_y^2 \frac{\partial^2}{\partial x^2} - 2l_x l_y \frac{\partial^2}{\partial x \partial y} + l_x^2 \frac{\partial^2}{\partial y^2} \right) \{ l_x l_y^2 P_{xxx} \\ &\quad + l_y(l_y^2 - 2l_x^2) P_{xxy} + l_x(l_x^2 - 2l_y^2) P_{xyy} + l_x^2 l_y P_{yyy} \} \end{aligned} \quad (4-49)$$

$$\begin{aligned} F_{sst,ss} &= \left(l_y^2 \frac{\partial^2}{\partial x^2} - 2l_x l_y \frac{\partial^2}{\partial x \partial y} + l_x^2 \frac{\partial^2}{\partial y^2} \right) \{ -l_x^2 l_y M_{xxxx} \\ &\quad + 2l_x l_y(l_x^2 - l_y^2) M_{xxxy} + (-l_x^4 + 4l_x^2 l_y^2 - l_y^4) M_{xxyy} \\ &\quad + 2l_x l_y(l_y^2 - l_x^2) M_{xyyy} - l_x^2 l_y^2 M_{yyyy} \} \end{aligned} \quad (4-50)$$

$$\begin{aligned} F_{sss,sss} &= \left(-l_y^3 \frac{\partial^3}{\partial x^3} + 3l_x l_y^2 \frac{\partial^3}{\partial x^2 \partial y} - 3l_x^2 l_y \frac{\partial^3}{\partial x \partial y^2} + l_x^3 \frac{\partial^3}{\partial y^3} \right) \{ -l_x l_y^3 M_{xxxx} \\ &\quad + l_y^2(3l_x^2 - l_y^2) M_{xxxy} + 3l_x l_y(l_y^2 - 3l_x^2) M_{xxyy} \\ &\quad + l_x^2(l_x^2 - 3l_y^2) M_{xyyy} + l_x^3 l_y M_{yyyy} \} \end{aligned} \quad (4-51)$$

4.1 任意節点配置差分法による解析

8階微分方程式の任意節点配置差分法による数値解析法について考察する。上述のように領域内では1個、境界上では4個の自由度をもつので、差分を行うときの節点も領域内では0次、境界上では3次とする必要がある。したがって、境界上ではパラメタ u, u_i, u_{ii} ,

u_{III} の不完全3次節点を設ければよいが、この節点はさらに境界外に3個の0次節点をもつダミー節点群で置き換えることができる。

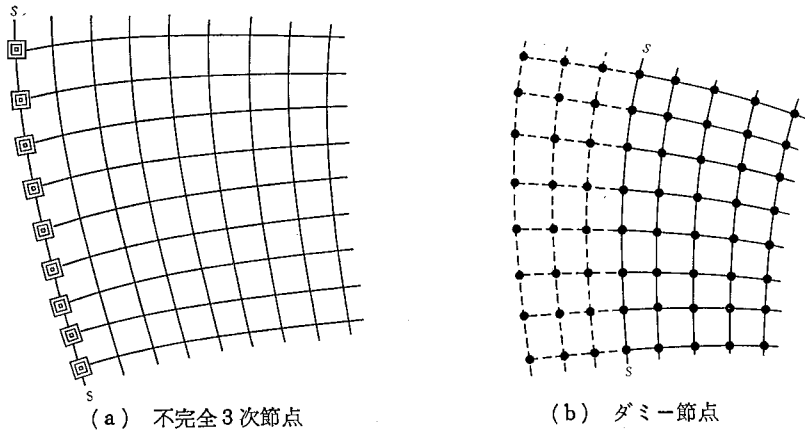


図 11 境界上でのパラメタ
Fig. 11 Parameters on boundary

ここはダミー節点法を用いることとする。原点における展開式は前節と同様に、

$$u(x, y) = \sum_1^{45} a_n f_n(x, y) \tag{4-52}$$

で表される。ただし、展開式の精度は8次で、項数は45である。ここで $n=1\sim 28$ までの a_n は式(3-43)と同様で、それ以上は次のようになる。

$$\left. \begin{aligned} a_{29} &= \partial^7 u / \partial x^7, & a_{30} &= \partial^7 u / \partial x^6 \partial y, & a_{31} &= \partial^7 u / \partial x^5 \partial y^2, & a_{32} &= \partial^7 u / \partial x^4 \partial y^3, \\ a_{33} &= \partial^7 u / \partial x^3 \partial y^4, & a_{34} &= \partial^7 u / \partial x^2 \partial y^5, & a_{35} &= \partial^7 u / \partial x \partial y^6, & a_{36} &= \partial^7 u / \partial y^7, \\ a_{37} &= \partial^8 u / \partial x^8, & a_{38} &= \partial^8 u / \partial x^7 \partial y, & a_{39} &= \partial^8 u / \partial x^6 \partial y^2, & a_{40} &= \partial^8 u / \partial x^5 \partial y^3, \\ a_{41} &= \partial^8 u / \partial x^4 \partial y^4, & a_{42} &= \partial^8 u / \partial x^3 \partial y^5, & a_{43} &= \partial^8 u / \partial x^2 \partial y^6, \\ a_{44} &= \partial^8 u / \partial x \partial y^7, & a_{45} &= \partial^8 u / \partial y^8 \end{aligned} \right\} \tag{4-53}$$

1) 展開関数列

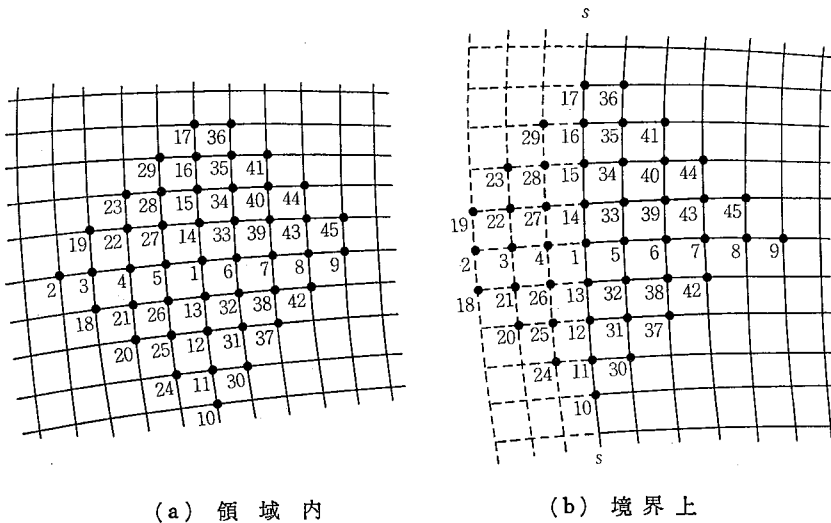


図 12 差分家族内節点配置
Fig. 12 Node distribution of difference family

f_{28} までは式(3-44)と同じで、それ以上は次のように示される。

$$\begin{aligned}
 f_{29} &= x^7/5040, f_{30} = x^6y/720, f_{31} = x^5y^2/240, f_{32} = x^4y^3/144, f_{33} = x^3y^4/144 \\
 f_{34} &= x^2y^5/240, f_{35} = xy^6/720, f_{36} = y^7/5040, \\
 f_{37} &= x^8/40320, f_{38} = x^7y/5040, f_{39} = x^6y^2/1440, f_{40} = x^5y^3/720, f_{41} = x^4y^4/576, \\
 f_{42} &= x^3y^5/720, f_{43} = x^2y^6/1440, f_{44} = xy^7/5040, f_{45} = y^8/40320
 \end{aligned}$$

2) 差分家族構成

差分家族は 45 の節点群により構成される。

4.2 有限要素法による解析

この微分方程式の階数は 8 であるから有限要素法により解析するためには 3 次節点をもつクラス D の要素が必要である。その代表的な要素を図 13 に示す。

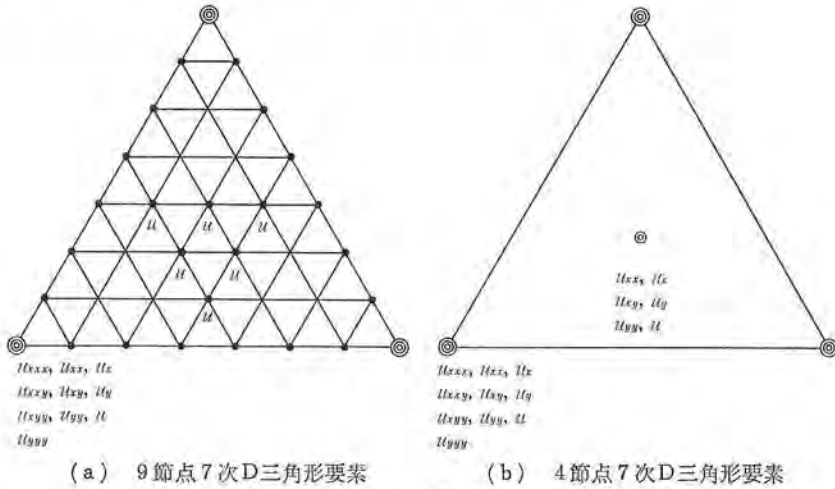


図 13 三角形要素

Fig. 13 Triangular elements

5. おわりに

高階の微分方程式も有限要素法，差分法等により数値的に求解することができる。任意節点配置の差分法を用いるとき 4, 6, 8 階の微分方程式ではそれぞれ 15×15 , 28×28 , 45×45 行列の逆行列をすべての差分家族ごとに求めることが必要である。4 階の微分方程式は板曲げや 2 次元粘性流解析に実用的に利用されているが，6 階以上の場合はかなり多量型の計算となることが予想される。有限要素法でも高次の要素を用いることが必要で，その形状関数を求めることも簡単ではない。現在のところ 6 階以上の微分方程式を取り扱う機会は少ないが，将来その必要性を生じたときにそなえて本稿を執筆した。

執筆者紹介 藤野 勉 (Tutomu 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 年依嘱教授を歴任。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 ボトル設計システム

Bottle Design Art Systems

佐藤 芳雄, 明尾 誠

要 約 ボトル設計を支援する CAD システムとして, BDAS (Bottle Design Art Systems) を開発した. 開発に当たっては, 過去のデザイン図面を参照し, ボトルのデザイン的特徴, 主に曲面や曲線の性質を分析した. その結果, 形状定義, 形状修正, 設計計算, 精密レンダリング, 三面図出力, 強度計算用メッシュ・データ作成, NC プログラムへのデータ出力等が簡単な操作でできるように, 設計業務の大幅な期間短縮が可能となった.

BDAS はボトル設計を対象に開発したシステムであるが, デザイン図面上の線を入力して面を作成していくこの方法は, 化粧品・食料品・医薬品等の各種容器や, 照明器具, 玩具, 時計の外装といった幅広い意匠設計の分野にも応用できるものと思われる.

Abstract BDAS is a CAD system developed for the purposes of bottle design. It was developed with the primary requirement that it be a practical system, and as such its development required a full analysis of bottle characteristics. The results of this analysis were employed to produce a system designed to permit dramatic improvement in the efficiency of design work and require only simple operation for such operations as shape definition, shape modification, design calculation, precision rendering, output of 3-view drawings, output of mesh data, and output of data to NC programs.

1. はじめに

洗剤, シャンプ, 化粧品等の新製品開発競争も, 他の例にもれず非常に激烈であり, 発売の機会損失のないように開発の期間を短縮することが重要な課題である.

一方, ボトルの形状は, 市場での誘目性, 使用感等を考慮してデザイナーが入念に作りあげるもので, むずかしい曲面を結合し, 面の境界にはエッジ・ラインと呼ばれるアクセントをつける場合が多い. また, 各々のボトルの形状は典型的でなく, 1,000 種を上回るほど多様なものである.

以上を考慮したボトルの製品設計を支援する CAD システムを, ライオン(株)からの依頼で開発した.

一般に, ボトルは, CAD システムの対象としては取り組みやすい形状と思われる. すなわち, 自由曲線を定義し, これをある軸を中心に回転させればボトルの形状となり, さらにこの回転面を修正すれば, かなり変化に富むボトル形状が得られるからである. このため, CAD システムが研究され始めた当初から今日に至るまで, ボトル形状をグラフィックス端末装置に表示した例は数多く紹介されている.

しかしながら実用に即した事例は, きわめて少ない. その理由は, 回転面や単一の面として扱えるボトル形状は少なく, ほとんどのボトルが複雑な面が結合したものであり, 容量・成形性・強度等を考慮した上で, デザイナーが意図する複雑な形状をモデリングすることは, 現実にはむずかしいからである.

ボトル設計システム BDAS は, 実用的システムとして利用されることを最大の目標として開発した. 以下, BDAS の機能を中心に述べる.

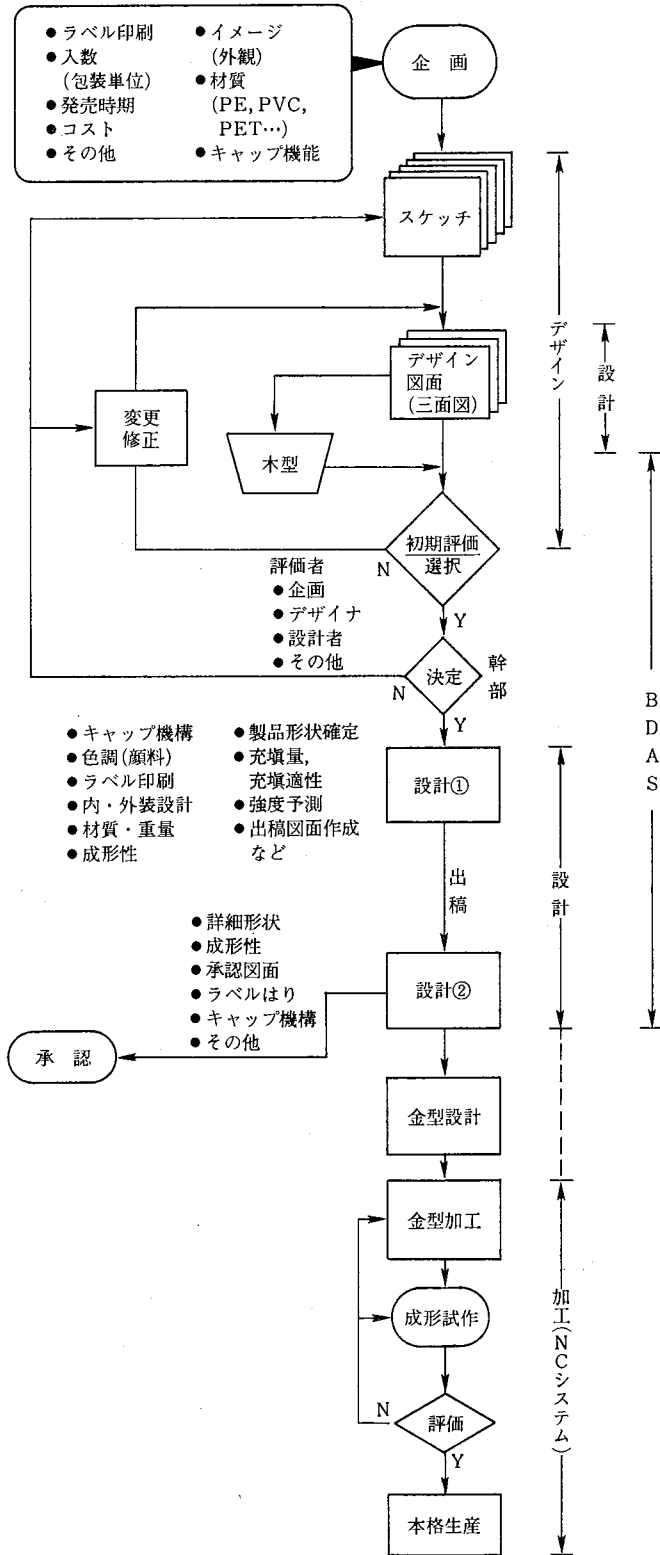


図 1 ボトル製品化の流れ
Fig. 1 Bottle production process


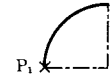



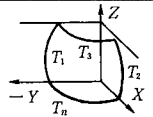

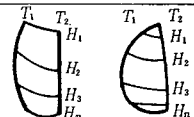
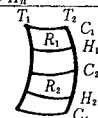
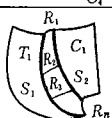
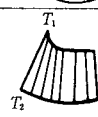

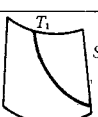
線 定 義	A2P		2点を通過する円
	A1P		通過点(1点)がわかっている円
	APN		直前に定義された図形(円, 直線)に接する円
	ACNR		直前・直後に定義された図形(円, 直線)によって作られる鋭角を丸める円弧
	L2P		2点を結ぶ直線
面 定 義	SBA 1		正面図, 側面図より入力された統合線 (T_1, T_2) と断面図より入力された統合線 ($T_3 \sim T_n$) で作られる面
	SBA 2		$T_1 \sim T_n$ の統合線による面
	SWP 1		T_1 と基準面 (X-Z 平面または Y-Z 平面)
	SWP 2		$T_1 \sim T_2$ と $C_1 \sim C_n$ (キャラクタ・ライン) または $R_1 \sim R_n$ (高さ指定: $H_1 \sim H_n$) による面
	FLT 1		二つの面 (S_1, S_2) をつなぎ一方の面 (S_2) に接する面
	SRUL		T_1, T_2 間を直線で結ぶことにより作られる面
	SREV		T_1 を回転 (Z 軸) させることにより作られる面
	SSUB		おおまかに作られた面の必要な部分を取り出す

図 3 コマンド例

Fig. 3 Example of commands and parameters

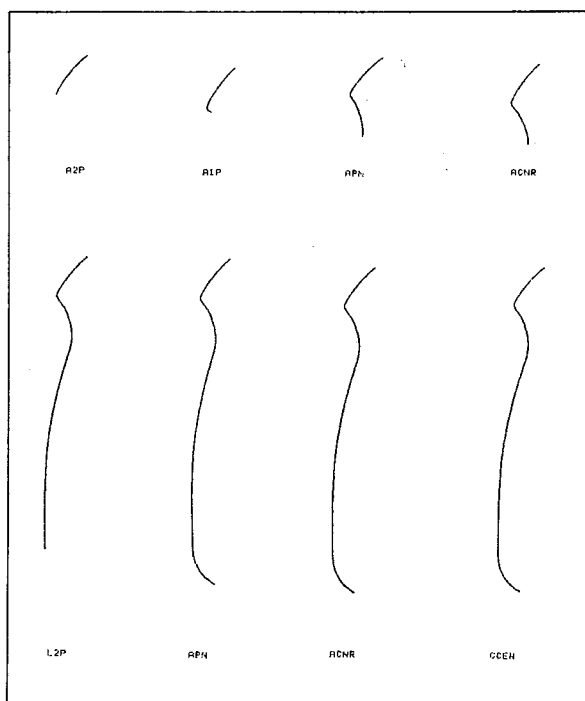


図 4 統合線の定義

Fig. 4 Example of definition of connected lines and output of the result

異なる線を組み合わせているのが普通である。このような線を無理に自由曲線に当てはめると、操作上からも内部処理上からも、かなり不自然なものとなる。

BDAS では、このような円弧・直線をそのまま連結して 1 本の線とした“統合線” (connected line) という概念を採用した。これは、利用者にとっても直感的で理解しやすく、後述する形状修正の際にも、扱いやすいという利点がある(図 4)。

なめらかな正面図、側面図の線と、デザイナーが指定している特定の断面線の数本(拘束線と呼ぶ)とからなる面で同じような傾向をもつものを BDAS では“基本面”と呼び、

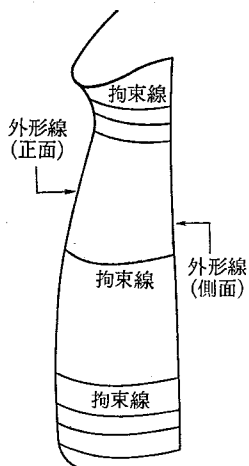


図 5 基本面の入力線

Fig. 5 Basic surface input lines

一つの面として扱う(図5)。

基本面を定義する場合、これらの拘束線は操作性から一括入力ができるよう操作性をよくした。通常、拘束線が多い場合できた面は不連続となるが、内部処理では、この拘束線の性質を損なうことなく、また面の不連続性も極力少なくするように配慮した。

基本面以外の面に関しても、エッジ・ラインから他の面に滑らかに接続する“フィレット面”や、いくつかの種類の“スイープ面”等を、徹底したケース・スタディを通じて厳選した。

3. 機能

デザイン図面では、デザイナーのイメージは、三面図によってかなり具体的に表現されている。しかしながら、製品設計のための要件に対する配慮は容量の概算程度のものであり、さらに図面については、幾何学的な矛盾や曖昧さを含んでいることが多い。

製品設計は、このデザイン図面をもとに、充填量、強度、成形性等の各種設計要件を考慮し、デザイン形状の微少修正を繰り返し、製品形状を確定する作業である。

BDASの機能を図6に、ハードウェア構成を図7に示す。

BDASでは、次のように、製品設計作業を進めていく。

3.1 形状定義

形状モデリングは、以下の手順で行う。

- 1) 正面図, 側面図, 断面図の線を統合線として順次定義する。

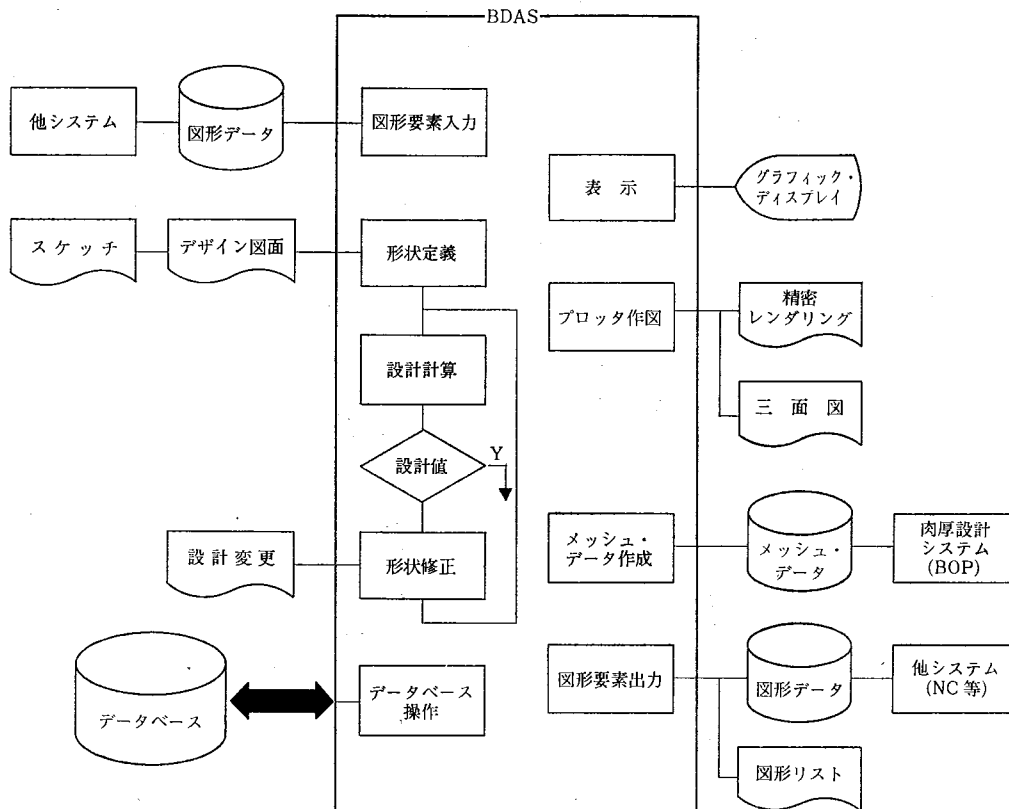


図6 BDASの機能

Fig. 6 Functions of BDAS

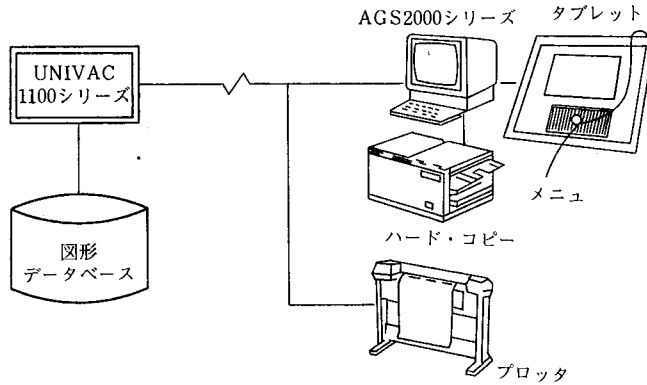


図 7 ハードウェア構成
Fig. 7 Hardware configuration

- 2) 1)によって定義した線を用いて、基本面等を定義する。
- 3) 2)で定義した面の不要部分を削除して、部分面とする。
- 4) 2)および3)で定義した面の間をつなぐ新しい面を定義する。

3.2 形状修正

形状修正 (図 8) は、デザイン図面をもとに定義した統合線の要素を操作することで、次の二つの手法がある。

1) 要素図形修正

統合線の要素である円弧や直線を修正する方法で、円弧については、半径・中心・始点・終点を、直線については、始点・終点を指示しなおして修正する。

2) 平行移動修正

統合線の要素を平行移動する方法で、移動する要素は、一つでも、複数の要素が連続したものでよい。

修正した要素に隣接する他の要素は、自動的に統合線定義時における相互の連続関係が保たれるよう調整する。そのあと、その線と関係をもっていたすべての線が、定義時のコ

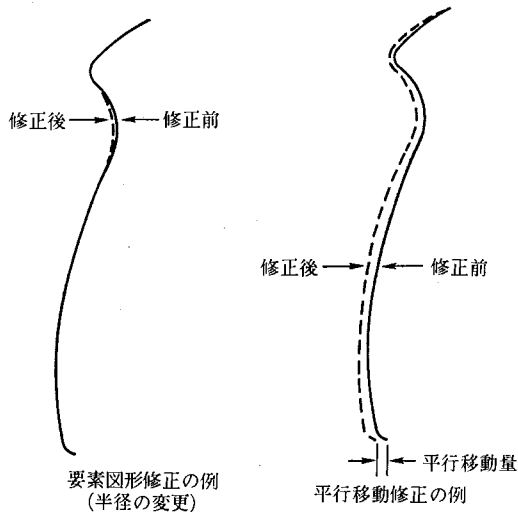


図 8 形状修正の例
Fig. 8 Example of shape modification

***	ボトル	シヨウホウ	***		
	ケイサン	タイシヨウ	ボトルパン	メイ	B001
	ボトル	ファンカウ	ズウ	-----	15
	タイア	ヨウセキ	-----		2.00 (ml)
	シヨウシ	シヨウリヨウ	-----		25.00 (g)
	シヨウシ	ミツト	-----		.955 (g/cm ³)
	クチモト	ニクアツ	-----		1.50 (mm)
	クヂイサ	ニクアツ	-----		1.20 (mm)
	タイア	ニクアツ	-----		1.50 (mm)
	トキウア	ニクアツ	-----		.30 (mm)
	マンネコウ	ヨウリヨウ	-----		628.91 (ml)
	イレメン	ウカサ	-----		231.75 (mm)
	シヨウケンリヨウ		-----		800.00 (ml)
	ヒヨウ	メノセキ	-----		592.21 (cm ²)

図 9 設計計算

Fig. 9 Checking design value

マンド・パラメタに従って自動的に再作成される。

すべての線の修正が終わった時点で、面の再作成の指示を与えると、線の修正にもなって不定となっていた面はすべて自動的に再作成される。

したがって、BDAS の利用者が統合線のうち必要な要素だけを修正指示すれば、ボトル全体の形状が自動的に修正される。このため、先に作成したと同じコマンドを再び投入するといった面倒な修正作業は不要である。この形状修正機能は、設計現場の実作業を反映させたものである。

3.3 設計計算

設計計算には、容量、断面積、表面積を求める機能がある。ボトルの容量は、単に体積を求めても無意味で、肉厚を考慮した内容量でなければならない。

BDAS での容量計算では、胴部の平均肉厚とともに、次の 3 項目が得られる (図 9)。

- 1) 満注容量……ボトルの全体容量
- 2) 入目線……指示した充填量 (たとえば, 500ml) に対する液面の高さ
- 3) 充填量……指示した液面の高さ (たとえば, 180mm) まで注入できる液量

これらの容量計算では肉厚を考慮するために、入力パラメタとして、口元、台座、底部

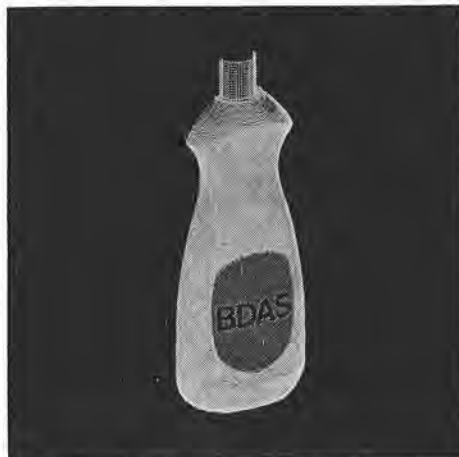


図 10 ボトルの表示例

Fig. 10 Display example

の各肉厚と底部容量，樹脂重量，樹脂密度を指定する。

3.4 ボトル表示

表示方法として，次の3種に分類できる。

1) ワイヤ・フレーム

稜線によって3次元モデルを表示する方法である。ただし，ボトルの微妙な形状変化は表現できない。

2) サーフェス

ワイヤ・フレームに面の情報を付加したもので，この方法でもボトルの微妙な形状変化は表現できない。とくに“つぎはぎの面”は違和感を与える。

3) ソリッド

サーフェスの面情報に，どちら側に物体があるのかという情報を追加したもので，3次元物体の表現には適している。しかし，ボトル形状のような複雑な自由曲面の集合体に適応すると，操作が複雑となり処理時間も増大する。したがって，現時点では実用的でない。

これらのことから試行錯誤を繰り返した結果，BDASでは細かいピッチの水平断面線群を一括表示する方法(図10)を採用した。この方法であれば製品形状が十分に予測でき，従来のレンダリングや木型の代用にもなりうるものである。

3.5 プロッタ作図

プロッタ作図には，次の二つがある。

1) 精密レンダリング

ラスタスキャン型のディスプレイ装置にボトル形状を表示した場合，斜線の階段的波形現象が現われ，デザイナーには不向きである。このため，ディスプレイ装置に表示されている図形そのものを，ハード・コピーと同様に簡単な操作で出力する機能を作成した。これは，デザイナーの間では，精密レンダリングと呼ばれ，好評である(図11)。

2) 三面図出力

製品設計終了後のモデルは，従来から図面という媒体を通じて金型設計の工程に引き継がれる。BDASでは，この図面を作成する製図機能はシステム負荷を増大することから用意していないが，簡単なパラメタで幾何形状と，基本的な寸法線，寸法値を自動発生させ，同時にその時の各種設計値もプロッタに出力する(図12)。

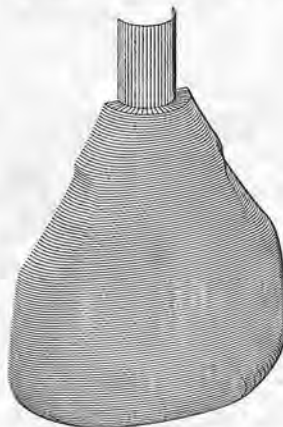


図11 プロッタ出力(精密レンダリング)の例

Fig. 11 Example of detailed rendering by plotter

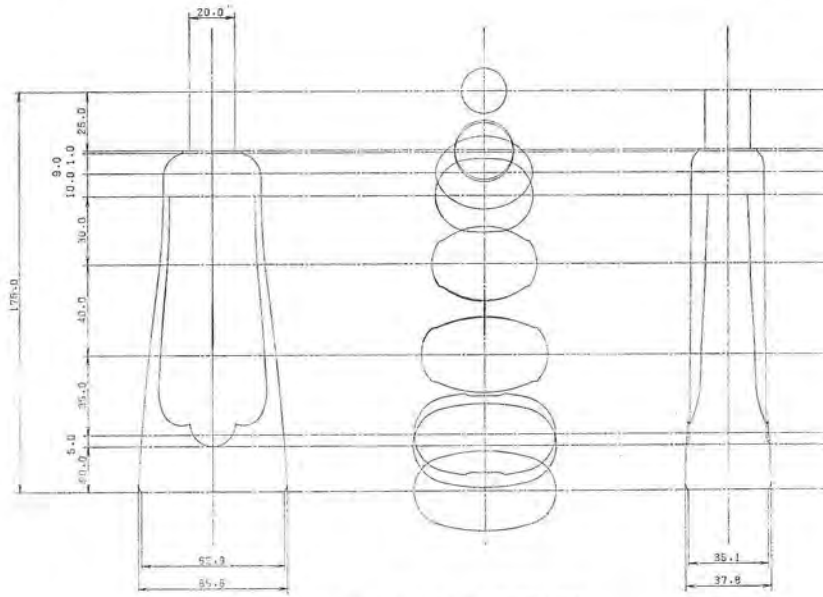


図 12 プロッタ出力 (三面図の例)
 Fig. 12 Example of plotter output

この三面図に、設計者がデータベースの図形情報を参照して寸法値を付け加えれば製品図面となる。

3.6 メッシュ・データ作成

作成したボトル形状の強度解析も製品設計の中の重要な作業である。

BDAS では、この強度解析のために、有限要素法を用いた応力解析によって得られるボトルの最適肉厚分布・最適樹脂量をディスプレイ装置に表示・検証する“BOP” (Bottle-wall-thickness Optimization Program: ボトル肉厚最適設計システム) を用意している。この BOP の入力データを作成する機能が、メッシュ・データ作成である (図 13)。

従来、有限要素法の入力データを作成するには多大の時間がかかっていたが、BDAS で

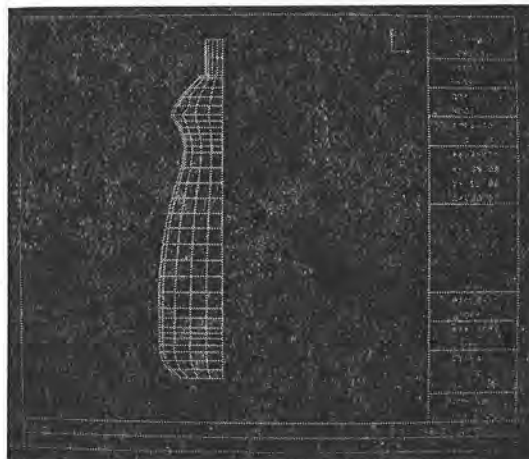


図 13 メッシュ・データ
 Fig. 13 Mesh data

は簡単なパラメタで自動的にメッシュ・データが作成されるため、大幅に時間が短縮できた。

4. 操作性

システムの操作性は、「基本的なデータをいかに簡単に、かつ正確に入力できるか」にある。

BDAS における“基本的なデータ”とは統合線であり、デザイナーや設計者の感覚に合った入力方法でなければならない。

これを実現するためには、統合線の定義に際して、

- 1) 始点から終点に向け一筆書きの要領で、各図形要素ごとに定義できる。
- 2) 図形要素のうち、接点や円弧の中心等、不明確なものは、あとの図形要素が確定した後、内部処理によって正しく定義しなおす。
- 3) 近傍点、近傍線といったデザイン図面上で表現されていない図形要素を必要としない。
- 4) コマンドやパラメタを極力減らす。

といった配慮を必要とする。しかし、これでは非常に簡単な入力操作で統合線の定義ができる一方、幾何学的には、一意に図形が定まらないケースが多発することとなる。この問題に対して BDAS では、ボトルの線の性質を配慮した“線の評価ルーチン”を設けることにより、自動的に最適解が求められるような工夫をしている。

断面図の線の定義に関しても、始点・終点を直接指示せず、正面図、側面図の統合線の名前を指示する。その結果内部的には、始点・終点を正面図、側面図の線の交点として求め、定義された断面図の線と関係をつける。こうすることによって、正面図の線が修正されれば、自動的に断面図の線の修正が可能となる。

以上述べたように、BDAS ではデザイナーや設計者の感覚に合った容易な入力方法を実現し、システムの操作性を向上させた。

5. 今後の課題

BDAS は、“機能・操作性・処理スピード”が実用に堪えうるよう開発された。ボトルの製品設計が、1本当たり3～5日で完成することから、当初の開発目標はほぼ達成できたものと思われる。

とはいうものの、形状モデリングができることを最重点に開発したため、今後は現機能の強化はもとより、上位のデザイン支援機能および下位の金型設計支援機能の拡充を図っていく必要がある。

BDAS のように意匠設計の分野で用いられる CAD システムでは、より製品に近い色で、より製品に近い形状を表示することが要求される。このためには、隠面・隠線消去、陰影付け、材質感など製品の視覚的検討を容易にするための機能強化が、応答性・操作性を損なうことなく実現されなければならない。

デザイナーがイメージを創成していく過程で、デザイン支援機能は線のデータの入力レパートリを拡充して、より簡単に、より自然にボトル形状が生成する必要がある。

金型設計支援機能のうち、BDAS では形状修正機能を用いて、キャビティ（金型の製品部）設計が容易に行える。金型全体の設計および製図には、“UNICAD” (UNiversal Integrated Computer Aided Design System: 汎用設計援助システム) が、金型加工には、

“SCULPTOR” (Surface CUTting Tool Path generaTOR: 複合曲面 NC 自動プログラミング・システム) が使用できる。

6. おわりに

デザインの創造過程を支援する CAD には、柔軟なモデリング、リアルな表現力、環境の中でのデザイン・シミュレーションが不可欠であり、これは従来の下位からのアプローチによる CAD の延長では到達できないといわれている。しかし、ハードウェア技術の向上と、3次元ソリッド・モデリングの実用化や、対話性の向上というソフトウェア技術の進歩があり、総合的にみて CAD に非常に使いやすく、コスト・パフォーマンスも良くなってきている。

BDAS もこのような現状を踏まえ、ボトル以外の化粧品・食料品・医薬品等の各種容器や、照明器具、玩具、時計の外装、家電製品といった幅広い分野の意匠設計に応用し、より使いやすいシステムにしたいと考えている。

本システム開発に当たり、御指導と御協力をいただいた、ライオン(株)の山口康文氏、相沢 修氏および日本ユニパック(株)の矢延 治、宿沢幸子、板東 司の各氏に感謝します。

- 参考文献 [1] 山口富士夫, 「図形処理工学」, 日刊工業新聞, 1981.
[2] 山口富士夫, 「形状処理工学」, 日刊工業新聞, 1982.
[3] 佐藤芳雄, 山口康文, “BDAS-ボトル設計支援システム”, *PIXEL*, No. 15, p. 144-148.
[4] 明尾 誠, “容器デザイン CAD-BDAS”, 映像情報, Vol. 16, No. 4, p. 49-53.
[5] 高橋 靖, “工業デザイン”, *PIXEL* 別冊, No. 3, p. 64-68.

執筆者紹介 佐藤芳雄 (Yoshio Sato)

昭和 18 年生, 41 年東京教育大学理学部卒. 45 年日本ユニパック(株)入社. NC ソフトウェア担当, 製造工業関連担当 SE を経て, 現在, 各種 CAD/CAM システム開発に従事.



明尾 誠 (Makoto Akeo)

昭和 24 年生, 48 年芝浦工業大学工学部卒. 49 年日本ユニパック(株)入社. 製造工業関連担当 SE, 全日空座席予約システム開発を経て, 現在 CAD システム開発に従事.



編集者注 本稿は, Computer Graphics Tokyo '84 (主催: 日本能率協会, 昭和 59 年 4 月 24~27 日) で著者らが発表した「Bottle Design Arts Systems」, および共同開発者であるライオン(株)の山口康文氏が著わした「ボトル設計を支援する CAD システムの開発」(日経コンピュータ, 昭和 54 年 6-25 号 pp. 155~170) と一部内容が重複します。

報告 MOS-VLSI 欠陥の分類, テスト, および除去

Classification, Testing, and Elimination of VLSI MOS Failures

Y. M. El-ziq

要約 シリコン集積回路のテストは、VLSI 回路生産の急激な拡大によってますます困難になってきている。総合的なシステム・テストは通常論理設計者とテスト技術者によって規定、実施されるため、故障モデルは論理的な縮退故障に頼らざるを得ない。本稿では縮退故障モデルの充分性を調べ、MOS の物理的欠陥を分類し、これらの障害のある特定のクラスを検出するテストの生成方法を示す。最後に、これらの障害のいくつかを除去するための設計規則とガイドラインについて述べる。

Abstract Testing silicon integrated circuits is becoming more formidable with the rapidly expanding production of VLSI circuits. Integrated system testing is usually specified and performed by logic designers and test engineers, existing fault models rely on a logical stuck-at basis. This paper examines the adequacy of the stuck-at fault model, classifies the MOS physical failures, and presents methods for generating tests to detect a specific class of these failures. Finally, it introduces some design rules and guidelines that help the elimination of some of these failures.

1. はじめに

シリコン集積回路のテストは VLSI 回路生産の急激な拡大によってますます困難になってきている。多くのテスト可能性問題がゲート数とピン数の増加ならびに速度の向上により起きている。その上、幾何学的微細化による信号線間隔の近接と製造工程の複雑化により、データに依存した、あるいは近接部の干渉による故障といった新しいタイプの障害を引き起こしている。以前は記憶デバイスだけに発生すると考えられていたこの種の障害は、現在では記憶デバイスでない汎用のデバイスでより多く起こるようになってきている。このような機能的に、あるいは縮退故障モデルのみによりテストされたデバイスは、相対的に低い信頼性しか示さないであろう。その理由は、故障を正しく論理レベルへ写像するのが容易でないので、結局、故障検出、摘出情報の価値が低くなるからである。そのため、VLSI の物理的欠陥を分析、分類、モデリングし、テストすることが重要となるのである。

VLSI で起きる物理的欠陥は主にテクノロジーに依存している。ここでは PMOS, NMOS, HMOS そして CMOS に限定することにする。集積 MOS システムは導体、絶縁体とトランジスタを形成する物質との多重層からできている。一連のプロセスを施すと、集積 MOS システムは絶縁体で分離された 3 層の導体から形成されることになる。これらの層は金属層、ポリシリコン層、および拡散層である。MOS トランジスタはポリシリコンのパスが拡散パスを横切ると集積システム上に形成される。

集積システムの障害のメカニズムの分類は簡単な仕事ではない。なぜならこうした障害は論理レベルで常に認識可能とは限らないからである。原因と結果が混じり合っているからでもある。一つの原因が他の因子の存在によっていくつかの異なる結果を生む一方、いくつかの原因が同じ結果をもたらす場合もある。そこで、総合的なシステム・テストは通

常論理設計者とテスト技術者によって規定し実施されるため, 故障モデルは, 論理的な縮退故障に頼らざるを得ない. 本稿では縮退故障モデルの適切さを調べ, MOS の物理的欠陥を分類し, これらの障害のあるクラスを検出するテスト生成方法を示す. 最後に, これらの障害にいくつかを除去するための設計規則とガイドラインを示唆する.

2. MOS 障害の分類

障害は三つの主要なタイプ, 物理的なもの, 電気的なもの, そして論理的なものに分類できる. 物理的欠陥は通常電気的障害をもたらす. 同様に電気的障害は論理的な故障をもたらす可能性がある.

物理的な欠陥による電気的障害は以下の三つのカテゴリに分類できる.

1) 開放回路 これは次の原因による.

- ① レベル間接合の切断を引き起こす輸送現象……たとえば金属層-ポリシリコン層間あるいは金属層-拡散層間の接触合等である.
- ② 機械的故障あるいは内部接続ライン切断の原因となるイオン不純物の存在……このようなラインは金属層, ポリシリコン層あるいは拡散層であるかもしれない.
- ③ ボンド・ワイヤあるいはリード線のようなチップとパッケージを接続する材料の切断を起こす熱的劣化および腐蝕

2) 短絡 これは次の原因による.

- ① 金属, ポリシリコンあるいは拡散のような同じレベルの要素間での短絡を起こすイオン汚染といった電気的現象
- ② チップとパッケージ接続の短絡を起こす粒子

3) FET stuck-on あるいは FET stuck-open これは, 次の原因によるものである.

- ① 過剰応力故障
- ② イオン汚染

いくつかのケース・スタディが行われ, その分析結果が報告されている. Banerjee およ

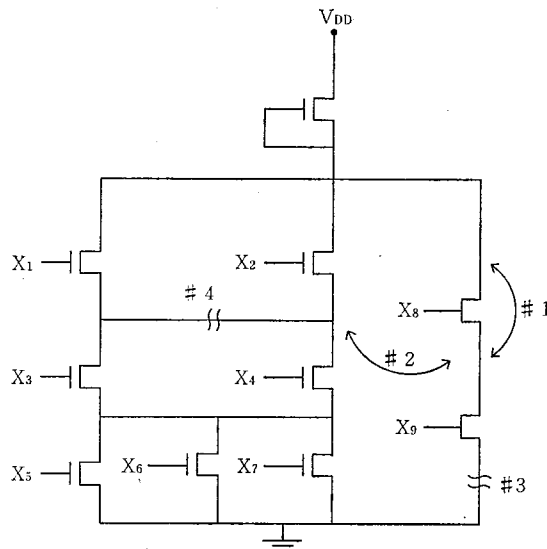


図 1 四つの障害をもつ MOS 回路

Fig. 1 Example of a MOS circuit with four possible failures

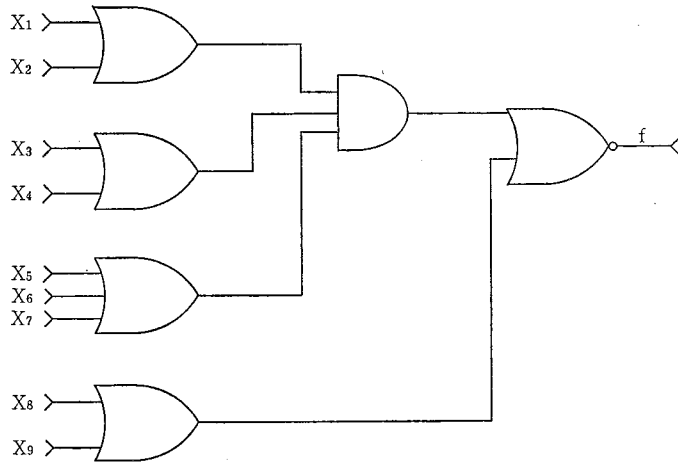


図 2 図 1 の MOS 回路の等価論理図

Fig. 2 An equivalent logic diagram of the MOS circuit of Fig. 1

び Crozet の文献^[1,51]の研究では主な接続障害は、拡散ライン間の短絡、アルミニウムとポリシリコン交差の切断あるいはアルミニウム線間の短絡によって引き起こされると結論づけている。しかしながら、トランジスタ (FET) 障害の大多数はゲートからドレインあるいは電源への短絡、ドレインあるいは電源、接続不良もしくはゲートから基板への短絡によるものであり、これらの障害の大部分は縮退故障ではモデル化できない。

例として図 1 の MOS 回路と、それと等しい図 2 の論理回路を考えてみる。この MOS 回路の金属層あるいは拡散層での障害のいくつかは、縮退故障モデルではモデル化できない。たとえば図 1 トランジスタ X_8 のドレインとソース間短絡を表す障害 #1 は図 2 のライン X_8 に対する ϕ -縮退故障としてモデル化できる。しかしながら、二つの特定ノード間の物理的短絡である障害 #2 は、縮退故障あるいは図 2 の等価論理回路における短絡としては表すことができない。同様に、図 1 の障害 #4 に示される開放回路の故障は、図 2 のいかなる縮退故障あるいは短絡によってもモデル化できない。反対に、ある種の論理故障は、実際の回路における物理的対応をもたない。例としては図 2 の NOR ゲートの入力間の短絡がある。

以上の結果として、次のように結論できる。すなわち三つのタイプの電氣的障害(短絡、開放および FET stuck-on または FET stuck-open) は、縮退故障モデルでは十分に表すことができない。以上から、われわれは次の 9 タイプの論理故障を設定した。

- 1) FET stuck-open
- 2) FET stuck-on
- 3) 金属層または拡散層における開放回路
- 4) 複合 MOS ゲートでの金属層または拡散層のライン間の短絡
- 5) ゲートのノードの金属層または拡散層のライン間ならびに同じゲートまたは他のゲートとの出力間の組み合わせられた短絡
- 6) 相異なるゲートの内部ノード間の組合せ
- 7) となり合ったゲートの出力間の組合せ
- 8) 連続して起こる前述の四つの短絡タイプのいずれか
- 9) 観測できない MOS 障害 (これはタイミングの問題、電荷のレフレッシュ・ミス等によるものであろう。)

3. 非古典的故障のモデル化

論理的には, いかなる MOS の設計も二つのタイプの複合ゲート, すなわち反転ゲートと通過ゲートを使用して形成できる. NMOS 反転ゲートは図 3 (a) に示すとおり, 負荷トランジスタとして働くディプレッション型トランジスタと駆動トランジスタとして働くエンハンスメント・トランジスタまたはそのグループからできている. これらのトランジスタは複合反転ゲートを作るため直列・並列あるいはそれを組み合わせて接続される. CMOSでは負荷トランジスタはドライバを並べて作られている. 通過ゲートは図 3 (b) に

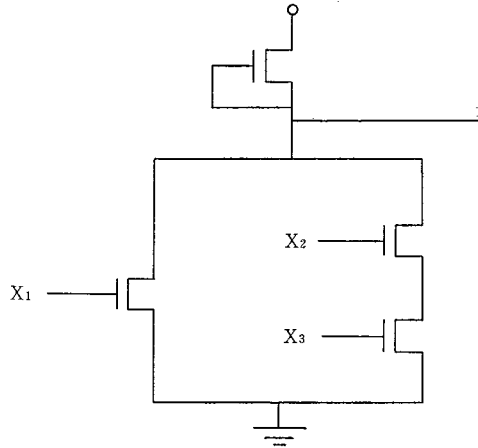


図 3(a) MOS 反転複合テスト

Fig. 3(a) A MOS inverting complex gate

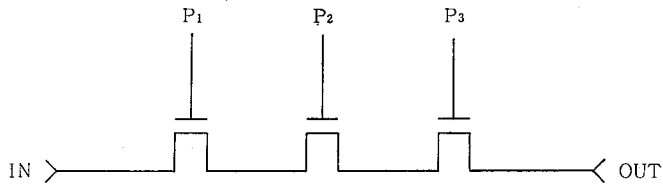


図 3(b) MOS 通過トランジスタ複合テスト

Fig. 3(b) A MOS passing transistor complex gate

表 1 図 4 の回路の NSW (Node-Switch-Wire) 表現

Table 1 A Node-Switch-Wire data representation of the circuit of Fig. 4

ノード・ネーム	N	E	S	W
I ₀₀₁	-1	0, I ₀₀₂	1, G ₀₀₄	-1
I ₀₀₂	-1	0, I ₀₀₃	2, G ₀₀₅	0, I ₀₀₁
I ₀₀₃	-1	-1	8, G ₀₀₆	0, I ₀₀₂
G ₀₀₄	1, I ₀₀₁	0, G ₀₀₅	3, G ₀₀₇	1, I ₀₀₁
G ₀₀₅	2, I ₀₀₂	-1	4, G ₀₀₉	0, G ₀₀₄
G ₀₀₆	8, I ₀₀₃	-1	9, G ₀₁₂	-1
G ₀₀₇	3, G ₀₀₄	0, G ₀₀₅	5, G ₀₁₀	-1
G ₀₀₈	-1	0, G ₀₀₉	6, G ₀₁₁	0, G ₀₀₇
G ₀₀₉	4, G ₀₀₅	-1	7, G ₀₁₂	0, G ₀₀₈
G ₀₁₀	5, G ₀₀₇	0, G ₀₁₁	-1	-1
G ₀₁₁	6, G ₀₀₈	0, G ₀₁₂	-1	0, G ₀₁₀
G ₀₁₂	7, G ₀₀₉	0, G ₀₁₃	-1	0, G ₀₁₁
G ₀₁₃	9, G ₀₀₆	-1	-1	0, G ₀₁₂

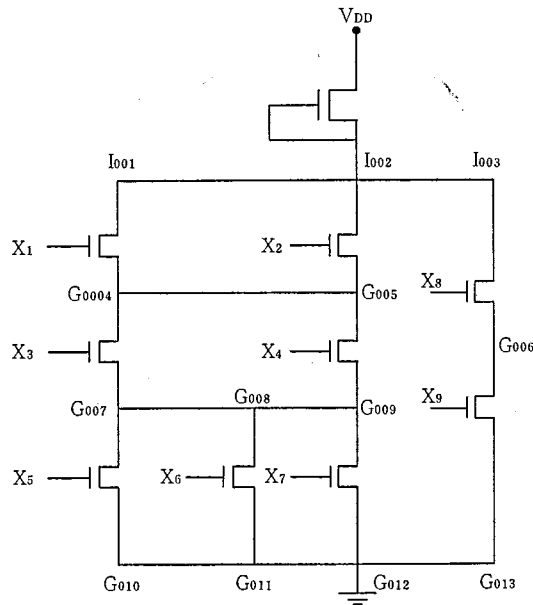


図4 図1の回路にノード名をつけたもの

Fig. 4 The circuits of Fig. 1 with named nodes

示すとおり，エンハンスメント・モード・トランジスタをドレイン，ソースで直列に接続してきており，一般に双方向である。

ゲートを表現するために使うモデルは NSW (Node-Switch-Wire) モデルと呼ばれる。このモデルでは，複合ゲートは相互に結合されたノードの集まりとして表現される。ノードの結合は，トランジスタまたはワイヤのみによってなされる。すべてのノードは，論理的には（物理的にはではない）四つのノードで囲まれていると考えられる。北，東，西そして南のノードである。したがって，ゲート内の故障を感知する技術を NEWS (北，東，西と南) 追跡技術と呼んでいる。図4はラベル付きノードをもつ図1の反転複合ゲートを示している。このゲートの WS モデルを表1に示してある。

4. 自動テスト生成

テスト・パターン生成の観点からは，前述の九つの論理故障タイプは三つの相異なるクラスにグループ分けできる。すなわち，

クラスA：タイプ1からタイプ4までを含む。

クラスB：タイプ5からタイプ7までを含む。このクラスは隣接物のすべての種類の短絡を述べており，故障を継続して引き起こすことはない。

クラスC：タイプ8とタイプ9を含んでいる。このクラスは一般的にテストが最もむずかしい。なぜなら，故障を継続して引き起こすからである。このクラスの一例は CMOS における ETF stuck-open 欠陥である。この故障に対するテスト生成の技術は参考文献^[6,7]に示されている。

クラスAの故障に対する自動テスト・パターン生成 (ATPG) を本稿で論議し，アルゴリズムと例を示す。一方，クラスBとクラスCの故障に対する自動テスト・パターン生成は，はるかに困難で時には手が出ないほど大量の計算が必要になることもある。

FET stuck-open と stuck-on 欠陥を調べるテスト生成問題は参考文献^[8]で論議され，

MOS 複合ゲートに対する基本セル・マトリックスモデルが導入され, 完全な故障検査テスト・セットの自動生成手順の報告が行われている. これには故障摘出問題も議論され, 故障位置を探す手順が導入されている. この故障モデルは, 金属層または拡散層における接触不良回路をカバーするため, Courtois により拡張された^[4].

MOS スイッチ-レベル故障シミュレーションの問題は Bose らが^[2]で議論したので, ここでは MOS 組合せ回路におけるクラスの故障に対する自動テスト・パターン生成手順を述べる. ネットワークは複合ゲートの階層構造で記述されているものとする. 機能ゲート・レベルあるいはトランジスタ・レベルの記述を含むことができるものとする. テスト生成手順のステップの概要を次に述べる.

ステップ 1……ネットワークの設定を複合ゲートの結合として形成する. すべてのゲートは NSW, ゲート, そして(または)機能にモデルをもつことが可能である.

ステップ 2……次の複合ゲートのため, セルとサブセルを規定する.

ステップ 3……そのゲートでテストする次のクラス A 故障を選ぶ.

ステップ 4……セル中の故障を検出するよう, 筆者が開発した伝達経路法 (Conduction path technique) を使用する. 複合ゲートでの伝搬は NEWS 追跡手順を使うことによってなされる.

ステップ 5……他の複合ゲートを通して故障を伝達させるためには, 従来の検出技術を使用し, そのゲート・レベルまたは機能レベルの記述を使って出力に出す.

ステップ 6……故障の影響がいかなる出力にも表れない時は, 検出できなかったものとする. そのゲート中でテストされるべき故障が他にもある時は, ステップ 3 を行う. 故障がない時は, 次のステップへ進む.

ステップ 7……さらにテストされるべき複合ゲートがある時は, ステップ 2 を行う. ない時は, テスト・リストと同時に検出故障および未検出故障を印書する.

5. テストが困難な障害の除去

いままで述べたテスト生成手順はクラス A の故障のみに対するものである. クラス B, クラス C の故障に対するテスト生成はさらに困難である. クラス B の故障のためのテスト生成は計算時間が許容できないほどになるのでまったく非現実的である. ほとんどのクラス C の故障をテストすることはさらに時間がかかる. したがって, 短絡と開放不良を最小化または除去するため, レイアウト技法を用いることを提案する. われわれはこの技術をテスト化物理設計と呼んでいる.

論理設計のテスト技術との関連と同様, テストを考慮して物理設計を行うには, VLSI 回路に起こるすべての, またはほとんどの故障が容易にテストできるようにレイアウトに一連の規則を付ける必要がある. テスト容易性の改良は, 主に次の二つの方法でなし得る.

- 1) 特定のノード間の短絡の可能性を削減するため回路のレイアウトを変え得ること.
- 2) 回路レイアウトは, また, 開放あるいは短絡が起こった時, これらの故障が論理的な縮退故障として現れるようにすること.

図示した例のとおり, 図 4 の回路の一部分, すなわちトランジスタ X_1 , X_2 , X_8 と X_9 のスティック・レイアウト (stick layout) を図 5 (a) に示す. 図はノード G_{005} と G_{006} 間

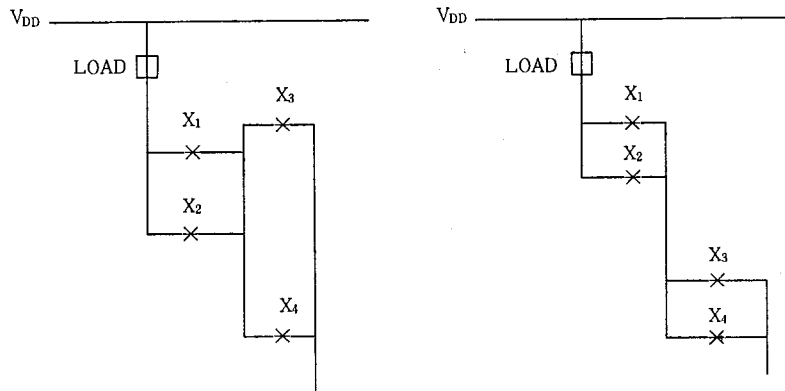


図 5(a) G_{004} - G_{005} の開放故障が縮退故障モデルで検出可能にするための 2 とおりのレイアウトの方法

Fig. 5(a) A portion of the circuit of Fig. 4 laid out in two different methods, to show how a G_{004} - G_{005} open fault can be made detectable by a stuck-at fault model

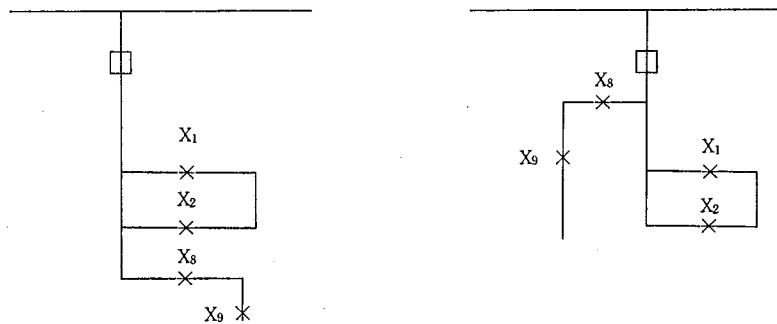


図 5(b) G_{005} - G_{006} で短絡を起こりにくくするレイアウト

Fig. 5(b) Another portion of the circuit of Fig. 4 that illustrates how a G_{005} - G_{006} short can be made more unlikely to occur

が短絡する可能性を、 X_8 と X_9 がある分岐を X_1 , X_2 側に移すことによってかなり削減できる方法を示している。同様に図 5 (b) は図 4 のノード G_{004} と G_{005} 間の拡散ラインの開放回路が縮退故障としてモデリングできない故障であるが、それを、トランジスタ X_1 , X_2 , X_3 , X_4 からなるゲートにおける ϕ -縮退故障としてモデリングできる故障へ変換する方法を示している。

テストを考慮した物理設計というゴールに到達するため、各テクノロジーに対する規則の包括的なセットを開発することができる。例として NMOS マイクロプロセッサ設計のための一連の規則が Galiay の文献^[9]で与えられている。

6. おわりに

複雑な VLSI システムのテストの容易化は技術的にむずかしい問題である。これはテスト可能性を、システム定義と論理・物理レベルでの設計の不可欠な一部分とすることによってのみ可能である。したがって、テストできる費用効果の高い製品を作るためにはレベル・アーキテクチャを決定した後、論理設計、物理設計においてテストのためのルール

の適用が必要となる。

とくに, 本稿では MOS VLSI システムを効果的にテストするには縮退故障モデルは不十分であることを述べた。物理的な欠陥とそれに付随する論理故障を分類し, これらの故障タイプのいくつかをテストする方法を述べた。しかしながら, 他のクラスはテストがはるかに困難であることがわかった。したがって, 設計の信頼性とテスト性を向上させるため, 広範なスイッチ・レベルのシミュレーション^[9]と欠陥シミュレーション^[2]が, ここで述べた自動テスト生成技術に加えて, 回路のレイアウト設計の前後に実施されねばならない。また, テストが困難な障害の生じる可能性を減らし, あるいは縮退故障モデルのような単純な故障モデルによる検出を可能にするため, レイアウト・ルールが必要となる。しかし, レイアウト・ルールはチップ面積を 0~20 パーセント程度まで増すであろう。したがって, 最も費用効果の高い VLSI 設計を達成するためには, チップ面積とテスト容易性の間のトレード・オフに関する高度の技術的判断をしなければならない。

(エンジニアリング技術部 CAD 推進室室長 上谷 彊輔 訳)

- 参考文献 [1] P. Banerjee, J. Abraham, "Fault Characterization of VLSI MOS Circuits", International Conference on Circuits and Computers, New York, October 1982.
- [2] A. Bose, et. al., "A Fault Simulator for MOS LSI Circuits", 19th Design Automation Conference, Las Vegas, Nevada, June 1982.
- [3] R. Bryant, "MOSSIM: A Switch-Level Simulator for MOS LSI", 18th Design Automation Conference, Nashville, Tennessee, June 1981.
- [4] B. Courtois, "Analytical Testing of Data Processing Sections of Integrated CPU's", 1981 IEEE Test Conference.
- [5] Y. Crozet, C. Landrault, "Design of Self-Checking MOS-LSI Circuits: Application to a Four-Bit Microprocessor", *IEEE Transactions on Computers*, Vol. C-29, No. 6, June 1980.
- [6] Y. El-ziq, "Automatic Test Generation for Stuck-Open Faults in CMOS VLSI", 18th Design Automation Conference, Nashville, Tennessee, June 1981.
- [7] Y. El-ziq, R. Cloutier, "Functional-Level Test Generation for Stuck-Open Faults in CMOS VLSI", 1981 IEEE Test Conference.
- [8] Y. El-ziq, "Fault Diagnosis of MOS Combinational Networks", *IEEE Transactions on Computers*, Vol. C-31, No. 2, February 1982.
- [9] J. Caliy, et. al., "Physical versus Logical Fault Models of MOS LSI Circuits: Impact on Their Testability", *IEEE Transactions on Computers*, Vol. C-29, No. 6, June 1980.

執筆者紹介 Yacoub M. El-ziq

1972年に Cairo 大学より B.S., 1975年に CCNY (New York 市立) 大学で機械工学の M. E. そして 1977年 Utah 州立大学で電気工学の Ph. D を取得。また Sperry 社および Honeywell 社でシミュレーション・テスト, ハードウェア設計をはじめとする CAD の領域での経験をもつ。Minnesota 大学では助教授として教鞭をとっている。現在先端技術のプログラム・マネージャとして, カスタム VLSI の CAD の研究に従事。IEEE 会員。



報告 シリーズ 1100 カナ漢字変換システムの開発

Development of Series 1100 Kana-to-Kanji Transformation System

三ツ矢 裕一, 吉田 正行, 小山 憲一

要約 日本語は、欧米語に比べ文字種（漢字、ひらがな、カタカナ、英数記号等）が多く、通常 4,000~5,000 字種を使用している。このため、日本語を計算機処理する上で入力の問題は障害とされてきた。

この障害を取り除くために、種々の日本語入力方式が考えられており、その一つにカナ漢字変換方式がある。入力速度、使いやすさ、さらに特殊な入力装置を必要としないという点から他の方式より優れており、使用者のニーズも高い。

本稿ではシリーズ 1100 のもとで稼動するカナ漢字変換システム（「日本語文書システム/入力」）の概要と、そのテスト結果を述べる。

本稿では、以下の項目について記述する。

1. システムの特徴/機器構成
2. 変換アルゴリズムの概要
3. 変換用辞書の種類と構造
4. テスト/評価
5. 考察と今後の課題

なお、当システムでは九州芸術工科大学の稲永氏作成の約 8 万 6 千語の辞書を使用し、変換アルゴリズムは九州大学の吉田・日高氏、福岡大学の吉村氏らが提唱している「文節数最小法」を基本としている。

Abstract The Japanese language has several thousands of ideographs including kanji, kana and alphabet, etc. Therefore, its input methods are key problems for Japanese document processing system. There are several methods to overcome this problem, in which Kana-to-Kanji transformation is one of the most easy methods for novice and casual users. This paper reports the features of Univac Series 1100 Japanese Document Processing System/Input and summarizes "Minimizing the number of BUNSETU method". It also describes that this system has exhibited high transformation ratio of 90%.

This system runs with 86,000 word-dictionary edited Prof. Inanaga of Kyushu Institute of Design and the transformation algorithm is based on "Minimizing the number of BUNSETU (phrase) method" proposed by Prof. Yoshida, Prof. Hidaka of Kyushu University and Prof. Yoshimura of Fukuoka University.

1. はじめに

日本語は、表音文字と表意文字を混用し、文字種も多く、音節構造は単純であるが同音語が生じやすい等の特徴がある。このことは、とくに日本語文を情報処理システムに入力する場合の障害とされてきた。

漢字タブレット、漢字テレタイプ鍵盤、和文タイプライタ等、フルキー方式では数千の漢字配列の中から一字ずつ文字を拾って文章を作成する方法がとられているため、入力の速度が遅く、オペレータの熟練が必要となる。一方、コード入力方式の中で、連想コード方式は、カナの対で漢字を特定する方式である。この方式は通常のカナ鍵盤を使用すれ

ばよく、漢字に対するカナの対を覚えさえすれば非常に速く日本語を入力することができる。しかし漢字とカナの対を覚えなければならず、これは気軽に習熟できる性質のものではない。したがって、これは専門オペレータ向き入力方式といえよう。

以上のフルキー方式、コード入力方式においては入力速度、習熟の容易性での問題がある。そこでだれでもが容易に利用でき、特別な入力装置を必要とせず、日本語文の入力を行うことのできる方式として、カナ漢字変換方式がある。

カナ漢字変換方式は、カタカナまたはローマ字で日本語文を入力し、それをソフトウェアにより漢字かな交じり文に変換する方式である。

ソフトウェアによるカナ漢字変換方式は、1960年代後半、九州大・栗原ら^[10,11]によって始められた。栗原らは簡単な“分かち”を施した文節について、カナ漢字変換用辞書を使って構文解析、および意味解析を行う基礎的な手法を提案するとともに、辞書作成のためのデータを収集整理した。この研究は、その後九州大・吉田、九州芸工大・稲永ら^[9]により引き継がれ現在に至っている。

カナ漢字変換方式を採用したソフトウェアを開発し商品化したのは、沖電気が最初であった。その後、東芝が1978年に日本語ワードプロセッサ JW-10 を、ホストコンピュータでのカナ漢字変換システムでは、富士通の JEF の一環としてカナ漢字変換方式を組み込んだ和文エディタ (FDMS/文書処理) を商品化した。これらに端を発し、ワードプロセッサの普及に伴ない、その他各社がカナ漢字システムを市場に出していることはいうまでもない。

日本ユニバック(株)でも LETS-J (Linguistically Extended Technology on UNIVAC System Japanese) の一環として、とくにシリーズ 1100 のもとで稼動する「カナ漢字変換システム」の開発を、次の理由から行った。

- 1) ワードプロセッサのような専用装置でない一般の端末装置でカナ漢字変換での入力ができること。
- 2) 現在、市場に出ているカナ漢字変換は文節分かち入力しかできないが、べた書き入力までサポートできること。
- 3) 生成するデータ形式を自在に変更することができ、ホストコンピュータ側での処理の連続性に都合がよいこと。
- 4) カナ漢字変換による入力を他のソフトウェア製品にも組み込める形態にできると。

また、システムを設計するに当たっての目標は、高変換率が達成できること、使用者がとくに文法的知識もなく連続してカタカナまたはローマ字で入力できる等の入力の容易性をもつこと、パフォーマンス向上のために効率的な検索が可能な辞書構造とすること、さらにシステムの保守性、拡張性、柔軟性に優れた構造とすること等である。

本稿では、LETS-J のカナ漢字変換システムの概要について記述するとともに、現段階での評価と今後の課題について述べる。

2. システムの構造

本システムは JASTY と JASLIB で構成されており、JASTY は、端末装置の画面を介して使用者が入力したカタカナ文字列またはローマ字文字列を、会話型でカナ漢字変換し、漢字かな交じり文のシンボリック・エレメントを作成するプログラムである。

JASLIB は、カナ漢字変換機能の必要な使用者プログラムから呼ばれる各種命令を処理するルーチンからなるライブラリである。

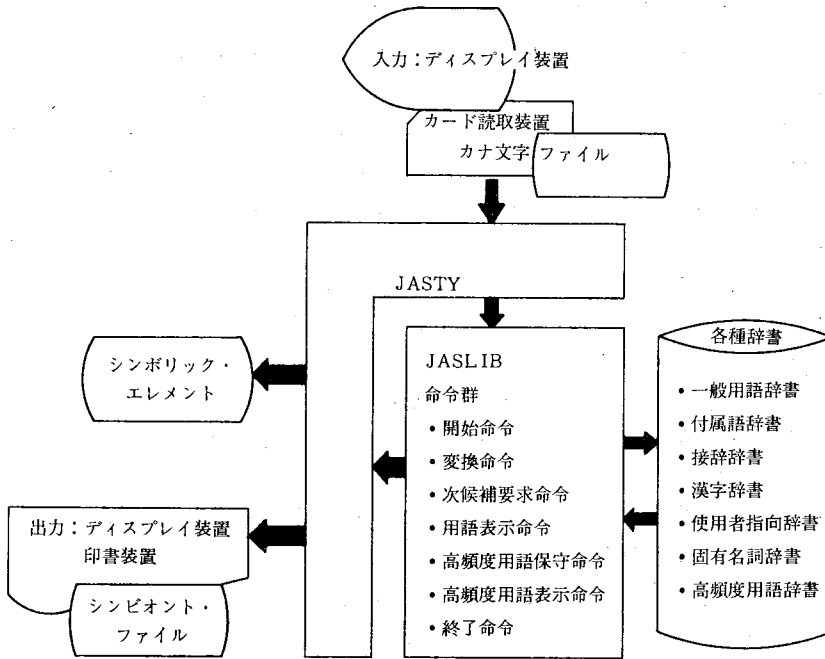


図 1 JASTY の構成

Fig. 1 Structure of JASTY

JASTY と JASLIB の構成をそれぞれ図 1, および図 2 に示す。

2.1 JASTY の特徴

JASTY の主な特徴を次に示す。

- 1) 画面を介して使用者が入力したカタカナ文字または、ローマ字を会話型でカナ漢字変換し、漢字かな交じり文のシンボリック・エレメントを作成する。また、一括入力によるカナ漢字変換機能ももっている。
- 2) 指定なしの入力（日本語の読みをそのまま入力）のほかに、16種の各種変換指定入力機能をもっている。
- 3) 変換後の漢字かな交じり文が正しくない場合は、簡単なキー操作で次候補が選択できる）次候補選択機能）。さらに選択した単語の出現頻度を高めることができる（高頻

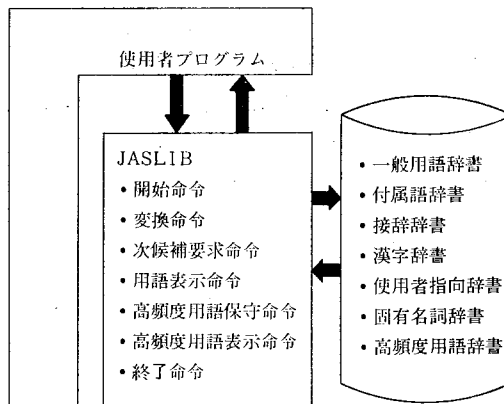


図 2 JASLIB の構成

Fig. 2 Structure of JASLIB

度用語自動登録機能).

- 4) 次候補の中にも正しい変換結果が含まれていない場合は, 正しくない部分を強制的に変換できる (強制変換機能). 強制変換して新しい用語を作成した場合, その用語を高頻度用語辞書に登録できる (用語登録機能).
- 5) 同じ読みをもつ用語の種類と優先度が表示できる (用語表示機能).
- 6) 用語登録機能により登録した用語の削除, または用語の表示を行ったり, 前もって用語を登録しておくことができる (用語保守機能).

2.2 JASLIB の特徴

JASLIB の主な特徴としては以下のものがある.

- 1) カタカナまたはローマ字文字列を漢字かな交じり文に変換する.
- 2) 指定なし入力のほかに, 音訓読み指定, 略号指定等, 16 種の各種変換指定入力ができる.
- 3) 変換後の漢字かな交じり文が正しくない場合は次候補が要求できる (次候補選択機能). さらに選択した単語の出現優先度を高めることができる (高頻度用語自動登録機能).
- 4) 指定した辞書中の同じ読みをもつ用語を要求できる (用語表示機能).
- 5) 使用者が作成した新しい用語を高頻度用語辞書へ登録したり削除することができる. また, 高頻度用語辞書へ登録した用語を知ることができる (用語保守機能).

表 1 変換指定一覧

Table 1 Summary of commands for transformation

名 称	形 式	機 能
無 変 換 指 定	A/文字列/	英字, 数字, 特殊記号を漢字コードへ変換する.
ひらがな指定	H/文字列/	指定された文字列をひらがなへ変換する.
カタカナ指定	K/文字列/	指定された文字列をカタカナへ変換する.
用 語 指 定	W/見出し語 [. n]/	指定された見出し語を一般用語辞書中の対応する漢字へ変換する.
使用者追加用語指定	U/見出し語/	指定された見出し語を使用者指向辞書, および高頻度用語辞書中の対応する漢字へ変換する.
姓 指 定	S/姓見出し語 [. n]/	指定された見出し語を固有名詞辞書中の対応する漢字へ変換する.
名 指 定	M/名見出し語 [. n]/	指定された見出し語を固有名詞辞書中の対応する漢字へ変換する.
人 名 指 定	G/姓[. n] 名[. n]/	指定された見出し語を固有名詞辞書中の対応する漢字へ変換する.
企 業 名 指 定	F/企業見出し語 [. n]/	指定された見出し語を固有名詞辞書中の対応する漢字へ変換する.
地 名 指 定	T/地名見出し語 [. n]/	指定された見出し語を固有名詞辞書中の対応する漢字へ変換する.
音 訓 読 み 指 定	Y/読み ¹ , 読み ² / Y/読み[. n]/	指定された読みを漢字辞書中の対応する漢字へ変換する.
JIS 8 単位符号指定	Q/文字列/	指定された文字列を 8 単位符号のまま残す.
区 点 番 号 指 定	J/区点番号……/	指定された区点番号を対応する漢字へ変換する.
拡張区点番号指定	E/区点番号……/	指定された区点番号を対応する漢字へ変換する.
16 進 数 表 現 指 定	X/16進数……/	指定された16進数を対応する漢字へ変換する.
略 号 指 定	R/略号/	指定された略号を使用者指向辞書, および高頻度用語辞書中の対応する漢字へ変換する.

2.3 入力種類一覧

入力は指定なし入力（変換指定をしない入力）が基本であるが，必要に応じて変換指定入力を使用できる．表 1 に変換指定一覧を示す．

3. 変換アルゴリズムの概要

日本語の統語規則は，単語の連鎖としての文節を規定する規則（文節構造規則）と，文節の連鎖としての文を規定する規則（係り受け構造規則）としてとらえることができる．すなわち，文節は一つの自立語，もしくは一つの自立語とそれに続く一つ以上の付属語で構成されるが，文節における語の並びにおいては，語とその直後に続く語の間に一定の接続条件が成立しており，これを文節構造規則と呼ぶ．また，文中の文節は互いに係り受けの関係（意味的呼応関係）をもっており，それらが有機的に結びついて文を構成する．これらの結びつきには規則性があり，これを係り受け構造規則と呼ぶ．以上の統語規則を用いて，文における形態素情報（品詞，活用情報，文節）と文節の係り受けを抽出し，構文的枠組みを決定することを構文解析と呼ぶ．

本システムでのカナ漢字変換では，上記文節構造規則のみを利用し，係り受け構造規則は利用していない．

次に，本システムでの中核である“べた書き入力”における変換アルゴリズムについて記述する．

3.1 文節数最小法の概要

一般に，カナ漢字変換の手法としては，最長一致法が採用されており，かなりの成果をあげている．この手法は，自立語または文節を決定する際に自立語または文節の長さの長いものを優先する．しかし，自立語や文節の長さに対する評価は，文全体における局所的な評価であるため，その方針を文全体に通して貫くことがむずかしい．

とくにべた書き入力を前提として，この手法での解析を考えると，解析途中での誤りに対するバックトラック時間は膨大なものとなる．これを解決する上で，本システムでは九州大／吉田・日高，福岡大／吉村ら^[1]が提唱する自動分かち書きアルゴリズム「文節数最小法」を基本的に採用している．

文節数最小法とは，べた書き入力された文字列をまず文節構造規則を用いて構成する単語や文節を決定する．その結果，意味的，構文的にも正しくない解析結果も含まれているが，この中で正しい解析結果は，文節数が小さい解析結果に含まれているという考えにもとづき，多くの解析結果の中から文節数の小さいものから優先して出力する手法である．最長一致法が一つの自立語や文節を決定するとき，できるだけ長いものを採用することは，文全体の文節数をできるだけ小さくすることに対応していると考えられる．したがって，文節数最小法に対しても最長一致法により得られる成果が期待できる．また，文節数に対する評価は文全体において大域的であるため，最長一致法よりバランスのとれた手法といえる．もちろんこのアルゴリズムで1文節入力の解析も可能である．

本システムでは，最小文節数の候補にほとんど正解が得られるというアルゴリズムの結論から最小文節数の候補のみを変換結果としている．ただし，同一文節数中での優先度は，構成する自立語が存在する辞書順（高頻度用語辞書>使用者指向辞書>一般用語辞書），自立語の長さの和の順で出力するよう考慮している．

BNF 記法を用いた日本語文の構造を図 4 に，また変換処理の流れ図を図 5 に示す．

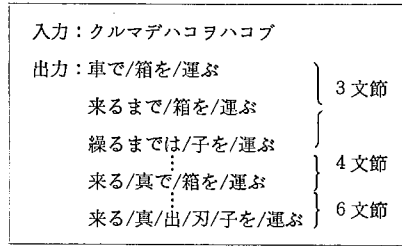


図 3 文節数最小法の例

Fig. 3 Sample of minimizing number of BUNSETU method

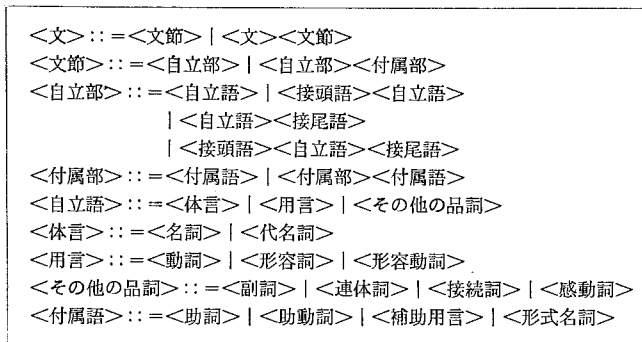


図 4 日本語文の構造

Fig. 4 Structure of Japanese sentence

3.2 派生語，数詞文節の処理概要

日本語は，接頭語や接尾語が自立語に付き派生語となる場合が多く，本システムにおいても派生語に対する特殊な処理を組み込んでいる（数詞文節についても，前置助数詞＝接頭語，後置助数詞＝接尾語として，広義に派生語として処理している）。

〈派生語の例〉

- 1) 前社長 (接頭語＋自立語) の例
- 2) 大衆化 (自立語＋接尾語) の例
- 3) 新大臣殿 (接頭語＋自立語＋接尾語) の例
- 4) 第5回 数詞の例
- 5) 川崎氏 人名接辞の例
- 6) 川崎市 地名接辞の例

接辞つきの語をすべて辞書に登録したのでは，辞書の容量は際限のないものになる。そこで，システムでは接辞辞書を別途用意し，接辞をグループ分けし（一般接辞，数詞接辞，人名接辞，地名接辞等），自立語の分類と接辞の連結関係により処理する方法をとっている。この方法により，上記例 5)，6) で示すような同音接辞に対する処理は可能となる。また，派生語における同音意義語の優先度は以下の順位としている。

1. 自立語のみの構成
2. (自立語＋接尾語) の構成
3. (接頭語＋自立語) の構成
4. (接頭語＋自立語＋接尾語) の構成

派生語処理の流れ図を図 6 に示す。

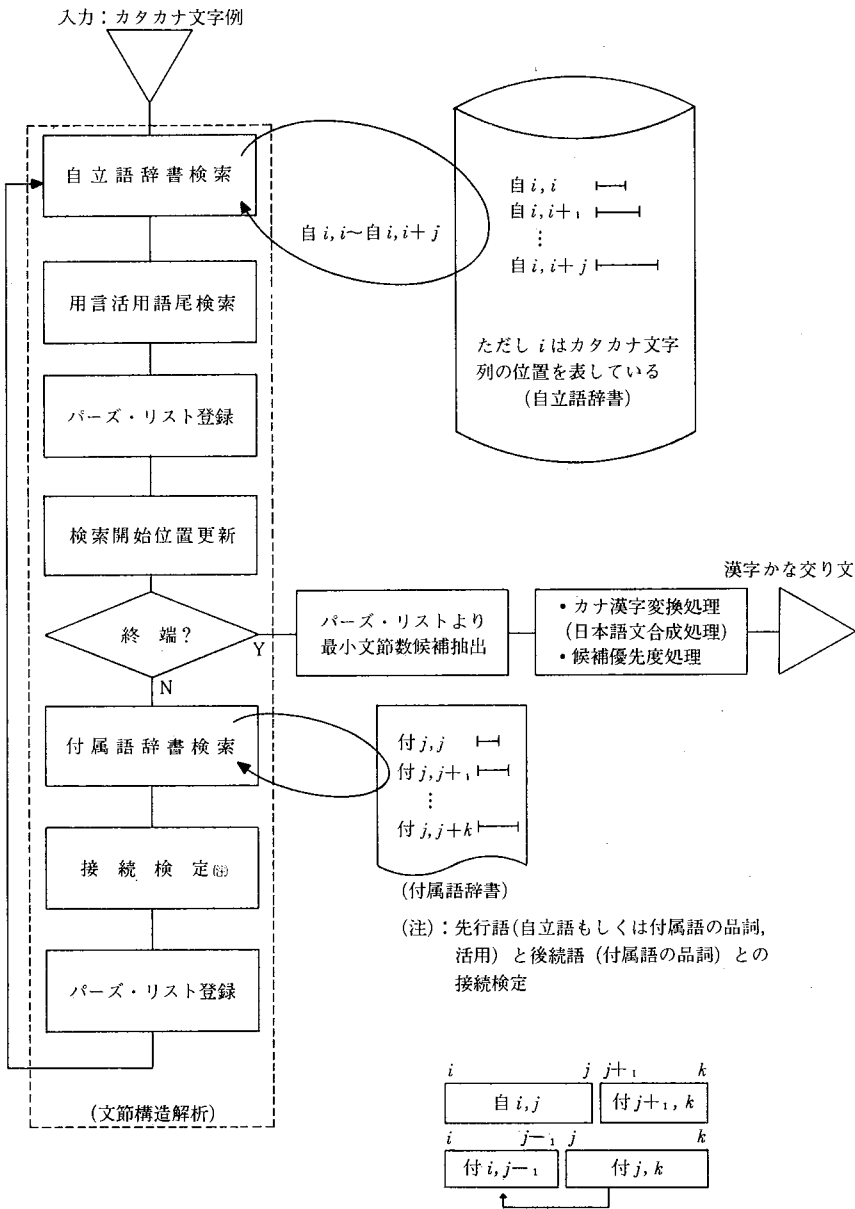


図 5 変換処理の流れ図

Fig. 5 The flowchart of transformation process.

4. 変換用辞書の種類と構造

4.1 辞書の種類

カナ漢字変換システムで使用する辞書には表 2 に示すものがある。

4.2 一般用語辞書の構造

一般用語辞書は語彙数が 86,420 語と膨大なため、外部記憶装置に格納している。しかし、カナ漢字変換システムにとって、一般用語辞書は中心となる辞書であり、使用頻度も高い。したがって、一般用語辞書の記憶方式/検索方式が重要である。このため最少の検索ですべての用語を抽出できる辞書構造とした。

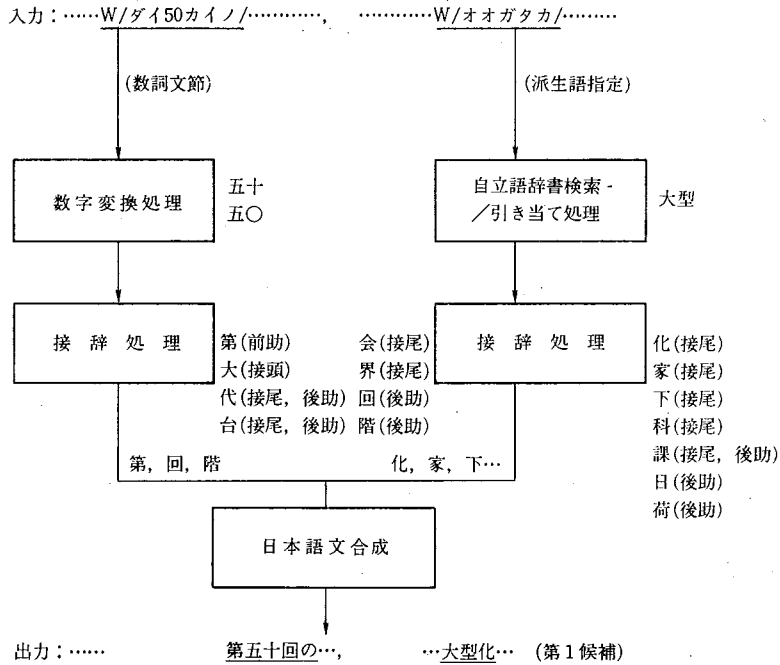


図 6 派生語処理流れ図

Fig. 6 Flowchart of derivative process

表 2 辞書の種類

Table 2 The dictionaries for transformation from Kana to Kanji

辞書名	容量	収容語彙数	備考
一般用語辞書	769 TRK	86,420 語	
付属語辞書	2 TRK	311 語 4,975 個	付 属 語 接 続 テーブル
接 辞 辞 書	4 TRK	514 語	
漢 字 辞 書	35 TRK	6,349 字 3,734 個	JIS 第 1 / 第 2 水 準 読 み 数 (音, 訓)
固有名詞辞書	187 TRK	20,045 語	姓, 名, 企業名, 地名
使用者指向辞書			使用者により異なる.
高頻度用語辞書	4 TRK		JASLIB の作業領域の大き さによって最小 100 語から 最大 500 語まで収容できる

4.2.1 辞書構造

一般用語辞書は、見出しをもつノード・レコードと、辞書内容をもつ辞書内容レコードとから構成されている。それぞれの特徴を以下に記す。

1) ノード・レコード

- ・文字ラベル：見出し語の 1 文字

文字ラベル	左ポインタ
右ポインタ	ブロック・ポインタ
辞書内容の数	

- ・左ポインタ：文字ラベルの左ノードを示すブロック先頭からの相対アドレス

- ・右ポインタ：文字ラベルの右ノードを示すブロック先頭からの相対アドレス
- ・ブロック・ポインタ：文字ラベル以前の見出しを含む可能性のあるデータ・ブロックのアドレス
- ・辞書内容の数：文字ラベルに対応する辞書内容レコードの数

2) 辞書内容レコード

漢字コード		漢字コード	
≈		≈	
品詞 ₁	品詞 ₂	品詞 ₃	品詞 ₄

- ・漢字コード：見出しに対応した漢字
- ・品詞_i：漢字コードで示される用語の品詞を最大4個もつ

上記ノード・レコードを西村・Veillon の二進木構造に、さらに階層構造をもたせたデータ構造にしている^[2]。辞書内容レコードは対応するノード・レコードの後に続けて入れている。例として、図7に示すソート済みの見出し語について辞書を構成したものを図8に示す。

4.2.2 辞書検索

本カナ漢字変換システムでは、べた書き入力のため単語と単語の間に区切りがない。しかし、入力文字列の先頭から始まるすべての単語を階層化した二進木のデータ構造で作成したことにより、1回の検索で入力文字列の先頭から始まるすべての単語を読み込むことが可能となった。

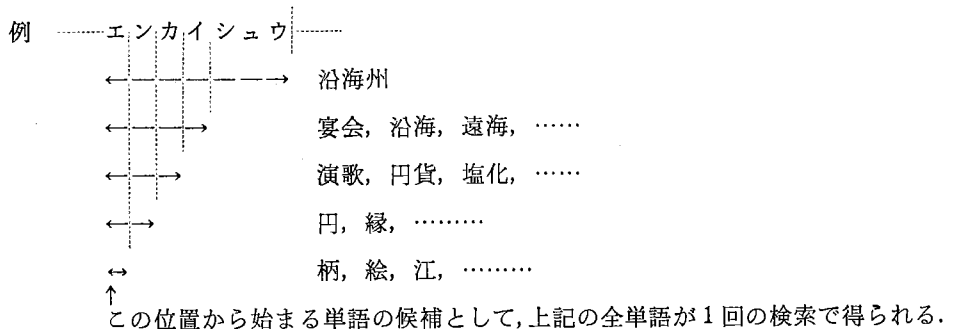
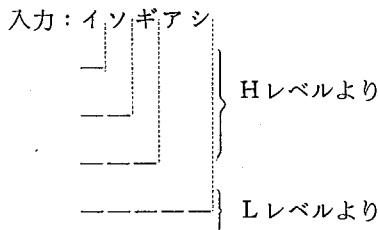


図7、図8を例にとり検索例を示す。



今入力文字列が「イソギアシ」の場合、Hレベルにある BLK 0 (主記憶装置に常駐) を探索し、「イ」、「イソ」、「イソギ」の三つの用語が得られる。BLK 0 を探索し、イノチ>イソギアシ>イソギから、「イノチ」のブロック・ポインタ7が得られ、Mレベルにある BLK 7 が主記憶装置に読み込まれる。次に BLK 7 を探索し、イタム>イソギアシからイタムのブロック・ポインタ5が得られ、Lレベルにある BLK 5 が主記憶装置に読み込まれる。最後に BLK 5 を探索し「イソギアシ」の用語が得られる。

1. アサ	12. イソガシイ
2. アサヒ	13. イソガス
3. イ	14. イソギ
4. イアン	15. インギアシ
5. イイキミ	16. イソグ
6. イエ	17. イタ
7. イエツキ	18. イタミ
8. イキ	19. イタム
9. イキウツシ	20. イタメル
10. イキオイ	21. イチ
11. イソ	22. イノチ

図 7 見出し語の例

Fig. 7 Example of words

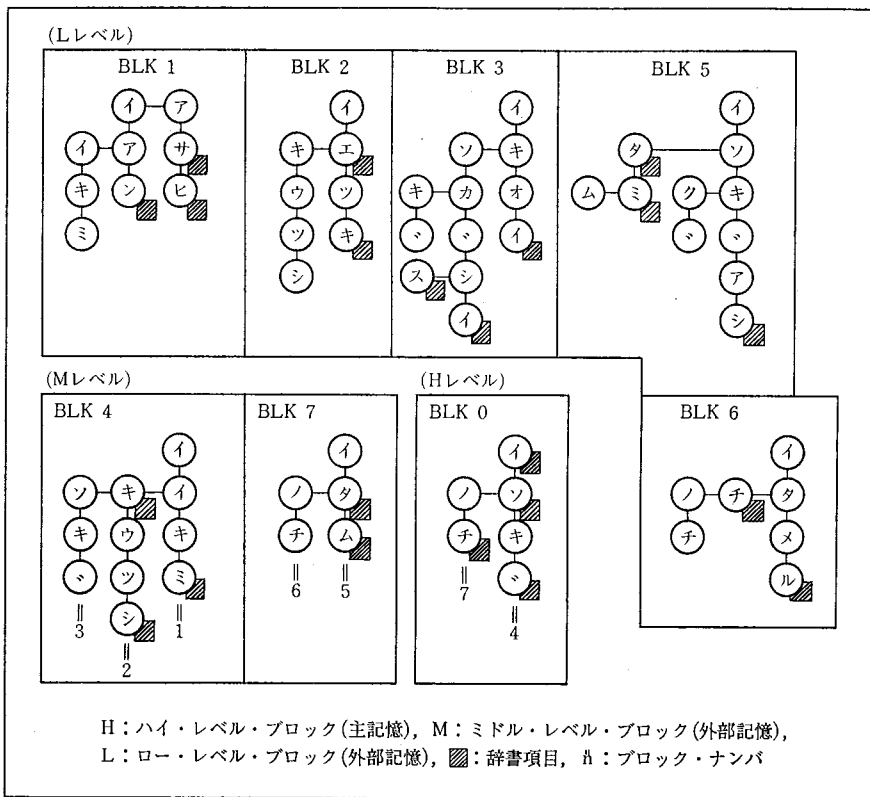


図 8 辞書構成の例

Fig. 8 Structure of independent-words dictionary

4.3 その他の辞書構造

4.3.1 付属語辞書

付属語辞書は付属語テーブルと接続テーブルから構成されており、カナ漢字変換中は主記憶装置上に常駐化する(図9)。

付属語テーブルは、二進木のデータ構造をしたノード領域とノードの文字ラベルに対応した付属語の品詞コードをもつ領域からなっている。

接続テーブルは、先行する単語と後続する単語が接続可能か否かを示すテーブルである。

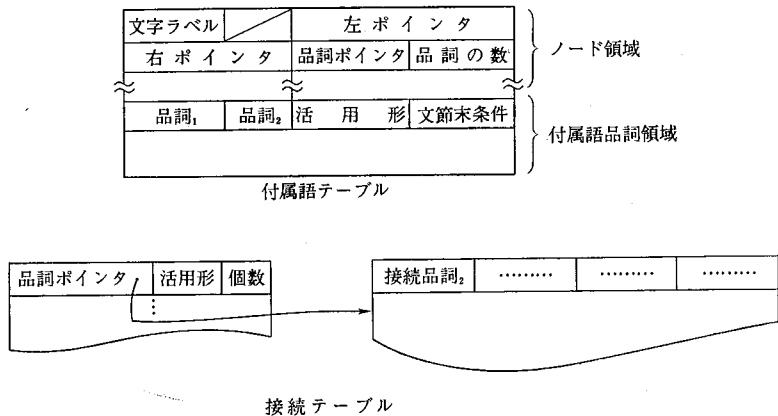


図 9 付属語辞書
Fig. 9 Dependent-words dictionary

4.3.2 接辞／漢字／固有名詞／使用者指向辞書

これらの辞書は特定の処理の時にのみ使用される使用頻度の低い辞書である。

- ・接辞辞書：用語指定入力と数詞文節処理
- ・漢字辞書：音訓読み指定入力処理
- ・固有名詞辞書：姓，名，企業名，地名，人名指定入力処理
- ・使用者指向辞書：使用者追加用語指定入力処理

このため，これらの辞書は，索引のみを主記憶装置上に常駐化しているが，辞書本体は必要のつど読み込んで使用している。

4.3.3 高頻度用語辞書

高頻度用語辞書は一般用語辞書の検索と同時に検索される辞書である。このため，使用頻度は高い。また，本辞書はカナ漢字変換実行中に用語の登録や削除も行われる。一方，登録用語数は，100語から500語と少ないのでカナ漢字変換実行中は，主記憶装置上に常駐している。辞書構造は，用語の見出しの昇順に並べた順ファイル形式である。

5. テスト／評価

5.1 テスト環境

カナ漢字変換のテストは，シリーズ 1100/82 を使用して行った(図 10)。

テストには入力文として武者小路実篤の「人生論」^[9]から 85 文を用意した。各々の文はべた書きした文字列であり，平均文字長は 22 文字である。

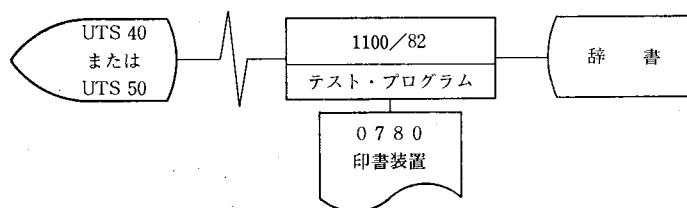


図 10 テスト環境
Fig. 10 Environment of experiment

5.2 変換率

カナ漢字変換の変換結果は、見る人によって正しいと判断されたり誤りと判断されたりする。そこで、カナ漢字変換の変換率を次のように定義する。変換結果を見て、日本語として意味が正確に理解できるものは正しく変換されているとする。

カナ漢字変換の変換精度を評価するため次の式を使用する。

$$\text{正変換率(文)} = \frac{\text{第一候補が正しい変換の文数}}{\text{入力した総文数}} \times 100$$

$$\text{多変換率(文)} = \frac{\text{次候補中に正しい変換のある文数}}{\text{入力した総文数}} \times 100$$

$$\text{誤変換率(文)} = \frac{\text{次候補中にも正しい変換のない文数}}{\text{入力した総文数}} \times 100$$

テストで使用した 85 文について評価すると次のとおりである。

$$\text{正変換率(文)} = 48/85 \times 100 = 56.5\%$$

$$\text{多変換率(文)} = 28/85 \times 100 = 32.9\%$$

$$\text{誤変換率(文)} = 9/85 \times 100 = 10.6\%$$

ただし、記憶域不足で正しい変換の得られなかった文も誤り文数に含めている。

文全体の評価としては、次候補中も含めて正しい変換結果が出力されるのは、 $56.5 + 32.9 = 89.4\%$ と高い精度が得られた。

一般的に使用されているカナ漢字変換の精度の評価では、文全体の変換率ではなく文字の変換率である^[3]。この方式に従うと評価式は次のとおりである。

$$\text{正変換率} = \frac{\text{正しく変換された字数}}{\text{入力文の総字数}} \times 100$$

$$\text{多変換率} = \frac{\text{同音異義語となった字数}}{\text{入力文の総字数}} \times 100$$

$$\text{誤変換率} = \frac{\text{誤って変換された字数}}{\text{入力文の総字数}} \times 100$$

この式を用いてテスト・データを評価すると次のようになる。

$$\text{正変換率} = 1,125/1,282 \times 100 = 87.8\%$$

$$\text{多変換率} = 112/1,282 \times 100 = 8.7\%$$

$$\text{誤変換率} = 45/1,282 \times 100 = 3.5\%$$

- ・正しく変換された字数：第一候補の中で正しく変換された字数。
- ・同音異義語となった字数：第一候補の中で正しくないが次候補中に正しく変換された文がある字数。
- ・誤って変換された字数：第一候補の中にも次候補の中にも正しく変換された文がない字数。

第一候補と次候補を含めて正しく変換された文字は、 $87.8 + 8.7 = 96.5\%$ と非常に高い値が得られた。このテストで誤変換となった 9 文の内訳は次のとおりである。

記憶域不足…………… 6 件

最小文節数に正しい変換がない…………… 2 件

未登録語…………… 1 件

5.3 パフォーマンス

カナ漢字変換の処理時間はほとんど外部記憶装置の検索時間で費やされる。そこで、本カナ漢字変換システムでは 4 章で記述したように使用頻度の高い一般用語辞書の構造を階

表 3 カナ漢字変換機能比較

Table 3 Comparison of functions for Kana-to-kanji transformation

メーカー名		日本ユニパック	富士通 ^[6]	日本IBM ^[4]
製品名		JASTY/JASLIB	FDMS (日本語文書処理システム) の FDMS (編集) / JEF	ことだま (日本語文書処理システム)
変換方式	変換方法	文節数最小法	最尤評価法	最長一致法 (後端→前端解析)
	同音語の選択順位	最小文節数を構成する文節列順 (文節数最小法) 同一文節数に対しては、辞書優先度順 > 自立語長さ順	自立語の長さ、使用頻度を要素とする評価関数の大きい順 (最尤評価法)	1. 辞書優先度順 2. 同一辞書では自立語長さ順 > 頻度順
入力方式	入力分かち書き	べた書き入力	文節分かち書き入力 (べた書き入力も可?)	文節分かち書き入力
	ローマ字入力	○	○	○
	特殊入力	16種の指定入力	10種の指定入力	—
特殊処理	複合語処理	○	△	—
	接辞処理	○	○	—
	数詞処理	○	○	—
辞書	種類	6種類+付属語辞書 ・高頻度辞書 ・使用者辞書 ・一般用語辞書(*) ・漢字辞書 ・固有名詞辞書 ・接辞辞書	2種類 ・マスタ辞書 (付属語, 固有, 漢字, 接辞を含む)(*) ・私的辞書	5種類+付属語辞書 ・仮り辞書 ・ユーザ辞書 ・高頻度辞書 ・一般単語辞書(*) ・固有名詞辞書
	国語辞書の語彙数 (上記*印)	約8.6万	約6万	約10万

層化した2進木のデータ構造とした。理論的には、1回の検索に必要な入出力回数は最大2回である。

テスト・データについて入出力回数を調べると、入力文の総字数1,828文字に対し入出力回数は、2,888回であった。平均すると入力1文字当たり約1.6回の入出力が出ていることになり、この値は九州大学での実験値と一致している^[7]。

以上のことより、パフォーマンスは当初の予想どおりの値が得られた。

5.4 入力方式の比較

カナ漢字変換には、すでに述べてきたようにいくつかの入力方法がある。

<例文> 彼女は旅行へ行きました。

- 1) カノジョ/ハ/リョコウ/ヘ/イキ/タシタ。………… (単語単位)
- 2) カノジョハリョコウヘ/イキマシタ。………… (文節単位)
- 3) (カノジョ)ハ(リョコウ)ヘ(イ)キマシタ。…… (漢字指定)
- 4) カノジョハリョコウヘイキマシタ。………… (べた書き)

1) は単語ごとに区切り、各単語ごと漢字変換もしくはひらがな/カタカナ変換する方法であり、以前はこの入力方法のカナ漢字変換がほとんどであった。

2) は文節ごとに区切り、各文節ごとに変換する方法であり、単語単位よりも入力しやすく、最近ではこの入力方法が主流となっている。使用者は文節を意識する必要がある。

3) は漢字部指定方式で、使用者が漢字に変換すべき読みを指定する方式である。実際の解析は、漢字指定した部分とカナの部分を加えた文節単位で、2) の文節単位入力と同様文法解析を行う。

4) はべた書き入力である。カナ漢字変換の理想であり一度に複数文節を入力できる。ただし、この場合分かち書きをシステム側で行うが、この手法の開発は、現在各社で研究されている。ただし、複数文節を許容するものであり、文節単位に区切って入力したほうが変換処理時間も短く、誤りも少ない。

ほとんどの日本語ワードプロセッサが、文節分かち入力のカナ漢字変換を採用し、変換方式として基本的に最長一致法を採用している。これは、べた書き入力を許容すればシステムの負荷も大きく、完璧に自動分かち書きができるわけではなく、誤った変換をするよりは、文節分かち書き入力単位で変換した方が使い勝手がよいという考えにもとづいていると思われる^[6]。また、同音語については、最後に使用した単語を優先するよう考慮している。さらに、辞書については3～4万語程度とし、未登録語については、使用者が独自に登録できるようになっている。

6. おわりに

カナ漢字変換は、日本語情報の最も有力な入力方法である。とくにべた書き入力に対する自動分かちアルゴリズムの研究開発が現在、大学、研究機関、他社メーカーでの中心である。このことは、より使いやすさを求め、正確に文節ごとに区切るという使用者の負荷を減らすことを意図している。

現在、提案されている自動分かちアルゴリズムは、本システムで採用している「文節数最小法」(九州大、日本ユニバック)の他に、自立語・付属語という文節が繰り返し現われると仮定に、自立語の長さを対象とした「最長一致法」(日立)、文を構成する単語の出現頻度と読みの長さとの評価関数の値から自動分かちする「最尤評価法」(富士通)、連続した2文節の長さが最も長くなるように分かちする「2文節最長一致法」(大阪大、沖電気)、助詞の候補を文節の終端と仮定して、その前の4音節の単語をもとに文節の切れ目を予測して自動分かちする方法(松下電器)等がある。しかし、どのアルゴリズムも完全な自動分かちができるわけではなく、一長一短がある。

「文節数最小法」を使用してのテストについての結果を5章2節で記述した。

正変換率とは第1候補で正解が得られたものであり、多変換率とは次候補に正解が得られたものである。次候補に正解が得られるということは、文節数最小法による自動分かちが一意に決まるわけではないことを示している。しかしながら、約90パーセント(文字数を基準にすると96.5パーセント)の高変換率を得られたことは、十分実用に耐える値であると共に、当アルゴリズムが優れた手法であることを示している。

今後の課題として、以下の項目が挙げられる。

- 1) 自動分かち処理へ接辞処理の組み込み
- 2) 未登録語処理
- 3) 係り受け構造解析

現在、派生語、数詞については、別途入力時に使用者が指示する方法を採っているが、これら接辞処理についても通常のべた書き入力の中に組み込んで入力、解析できるようにすること、また、未登録語(自立語辞書にない語彙)を含む文も解析可能とすること、さらに解析レベルを現在の文節構造解析のみでなく、係り受け構造解析レベルまで深くし、より曖昧さを少なくし変換率の向上を図ることがある。

カナ漢字変換での適用分野としては、JASTYで作成した日本語テキスト・ファイルが日本語文書構成プロセッサ(JDOC)への入力ファイルとなること、JASLIB(カナ漢字変

換ライブラリ) を使用してのアプリケーションへの適用が考えられる。

また、将来の応用分野として、日本語文(漢字かな交じり文)の構文解析、日英・英日機械翻訳など自然言語処理といった、より高度な知的処理への一步となろう。

最後に、カナ漢字変換システムの実現のために、ご指導下さった九州大学吉田将教授、日高達助教授、福岡大学吉村賢治講師、および当システムに使用した一般用語辞書の作成に初期の頃から従事し、以来十数年間これに専念してこられた九州芸工大稲永紘之講師に深く感謝します。

付 録 テスト結果

入力: SEIRYOKUWOTORITAINOHASHIZENNAHONNOUTOSUREBA

JACKTK STATUS 000

【第1候補】

016 勢力を取りたいのは自然な本能とすれば (正変換, 候補数は16個)

【次候補】

- 002 精力を取りたいのは自然な本能とすれば
- 003 勢力を採りたいのは自然な本能とすれば
- 004 精力を採りたいのは自然な本能とすれば
- 005 勢力を執りたいのは自然な本能とすれば
- 006 精力を執りたいのは自然な本能とすれば
- 007 勢力を捕りたいのは自然な本能とすれば
- 008 精力を捕りたいのは自然な本能とすれば
- 009 勢力を取りたいのは至善な本能とすれば
- 010 精力を取りたいのは至善な本能とすれば
- 011 勢力を採りたいのは至善な本能とすれば
- 012 精力を採りたいのは至善な本能とすれば
- 013 勢力を執りたいのは至善な本能とすれば
- 014 精力を執りたいのは至善な本能とすれば
- 015 勢力を捕りたいのは至善な本能とすれば
- 016 精力を捕りたいのは至善な本能とすれば

入力: イッホ^oウカ^oイキルタメニホウカ^oキ^oセイニナル

JACKTK STATUS 000

【第1候補】

015 一方が生きたために他方が犠牲になる (正変換, 候補数は15個)

【次候補】

- 002 一報が生きたために他方が犠牲になる
- 003 一法が生きたために他方が犠牲になる
- 004 一方が生きたために他方が擬声になる
- 005 一報が生きたために他方が擬声になる
- 006 一法が生きたために他方が擬声になる
- 007 一方が生きたために他方が擬製になる
- 008 一報が生きたために他方が擬製になる
- 009 一法が生きたために他方が擬製になる
- 010 一方が生きたために他方が擬勢になる
- 011 一報が生きたために他方が擬勢になる
- 012 一法が生きたために他方が擬勢になる
- 013 一方が生きたために他方が擬制になる
- 014 一報が生きたために他方が擬制になる
- 015 一法が生きたために他方が擬制になる

入力: モチロンホ^oクハカ^oクシャチ^oナイカラクイサ^oイノモンタ^oイロココテ^oトコウトハオモワナイ。

JACKTK STATUS 000

【第1候補】

008 勿論僕は学者でないから経済の問題を此処で渡航とは思わない。

【次候補】

- 002 勿論僕は学者でないから経済の問題を個別で渡航とは思わない。
- 003 勿論僕は学者でないから経済の問題を此処で解こうとは思わない。
- 004 勿論僕は学者でないから経済の問題を個別で解こうとは思わない。
- 005 勿論僕は学者でないから経済の問題を此処で説こうとは思わない。 →(次候補に正解あり.)
- 006 勿論僕は学者でないから経済の問題を個別で説こうとは思わない。
- 007 勿論僕は学者でないから経済の問題を此処で溶こうとは思わない。
- 008 勿論僕は学者でないから経済の問題を個別で溶こうとは思わない。

入力: MOUSUKOSHIHAKKIRIDOKUSYANOATAMANIIRETEOKITAITOOMOUNODA.

JACKTK STATUS 019

【第1候補】

038 もう少しははっきり読者の頭に入れておきたいと思うのだ。 (正変換, ただし領域不足の例)

【次候補】

- 002 もう少しははっきり読者の頭に容れておきたいと思うのだ。
- 003 申す腰ははっきり読者の頭に入れておきたいと思うのだ。
- 004 申す奥ははっきり読者の頭に入れておきたいと思うのだ。
- 005 申す古詩ははっきり読者の頭に入れておきたいと思うのだ。
- 006 申す古史ははっきり読者の頭に入れておきたいと思うのだ。
- 007 申す古しははっきり読者の頭に入れておきたいと思うのだ。
- 008 申す枯死ははっきり読者の頭に入れておきたいと思うのだ。
- 009 申す故紙ははっきり読者の頭に入れておきたいと思うのだ。
- 010 申す腰ははっきり読者の頭に容れておきたいと思うのだ。
- 011 申す奥ははっきり読者の頭に容れておきたいと思うのだ。
- 012 申す古詩ははっきり読者の頭に容れておきたいと思うのだ。
- 013 申す古史ははっきり読者の頭に容れておきたいと思うのだ。
- 014 申す古しははっきり読者の頭に容れておきたいと思うのだ。
- 015 申す枯死ははっきり読者の頭に容れておきたいと思うのだ。
- 016 申す故紙ははっきり読者の頭に容れておきたいと思うのだ。
- 017 申す越しははっきり読者の頭に入れておきたいと思うのだ。
- 018 申す澁しははっきり読者の頭に入れておきたいと思うのだ。
- 019 申す超しははっきり読者の頭に入れておきたいと思うのだ。
- 020 申す鼓しははっきり読者の頭に入れておきたいと思うのだ。
- 021 もう少し発きり読者の頭に入れておきたいと思うのだ。
- 022 申す越しははっきり読者の頭に容れておきたいと思うのだ。
- 023 申す澁しははっきり読者の頭に容れておきたいと思うのだ。
- 024 申す超しははっきり読者の頭に容れておきたいと思うのだ。
- 025 申す鼓しははっきり読者の頭に容れておきたいと思うのだ。
- 026 もう少し発きり読者の頭に容れておきたいと思うのだ。
- 027 申す腰発きり読者の頭に入れておきたいと思うのだ。
- 028 申す奥発きり読者の頭に入れておきたいと思うのだ。
- 029 申す古詩発きり読者の頭に入れておきたいと思うのだ。
- 030 申す古史発きり読者の頭に入れておきたいと思うのだ。
- 031 申す古し発きり読者の頭に入れておきたいと思うのだ。
- 032 申す枯死発きり読者の頭に入れておきたいと思うのだ。
- 033 申す故紙発きり読者の頭に入れておきたいと思うのだ。
- 034 申す越し発きり読者の頭に入れておきたいと思うのだ。
- 035 申す澁し発きり読者の頭に入れておきたいと思うのだ。
- 036 申す超し発きり読者の頭に入れておきたいと思うのだ。
- 037 申す鼓し発きり読者の頭に入れておきたいと思うのだ。
- 038 申す越し発きり読者の頭に容れておきたいと思うのだ。

- 参考文献 [1] 吉村賢治, 日高 達, 吉田 将, “表方式を用いた文節構造分析アルゴリズムとその能率について”, 情報処理, 計算言語学研資, 25-6, 1980.
- [2] 日高 達, 吉田 将, “効率的日本語単語機械辞書”, 情報処理学会第24回全国大会論集, 5 G-7, 1982.
- [3] 森 健一, 河田 勉, “かな漢字変換” 情報処理, Vol. 20, No. 10, 1979.
- [4] 藤崎哲之助, 大河内正明, 「ことだま文書処理システムの文節わかち書き仮名漢字変換」 情報処理, Vol. 23, No. 23, 1982.
- [5] 稲永紘之, 吉田 将, “日本語処理のための機械辞書”, 情報処理, Vol. 23, No. 2, 1982.
- [6] 住永洋子, “かな漢字変換方式の実現を探る”, 日経エレクトロニクス, 8-29, 1983.
- [7] 吉村賢治, 日高 達, 吉田 将, “日本語文の形態素解析における最長一致法と文節数最小法について”, 自然言語処理, 30-7, 1982.
- [8] FACOM OSIV/F4 FDMS(編集)/JEF 解説書 (日本語文書処理システム) 11, 1980.
- [9] 武者小路実篤, 「人生論・愛について」, 新潮社, “人生論” pp. 44~46, 1969.
- [10] 栗原俊彦, 黒崎悦明, “カナ文の漢字混り文への変換処理について”, 九大工学集報, Vol. 39, No. 4, 1967.
- [11] 栗原俊彦, 稲永紘之, “カナ漢字変換 (I)” 九大工学集報, Vol. 42, No. 6, 1970.

執筆者紹介 三ツ矢 裕一 (Yuichi Mitsuya)

昭和53年慶応大学工学部卒業，55年同大学大学院管理工学科修士課程卒業，同年日本ユニバック(株)入社。日本語ソフトウェア開発部配属，現在ソフトウェア二部に所属，日本語文書処理システムの開発，グラフ図形プログラム開発を担当。



吉田 正行 (Masayuki Yoshida)

昭和44年鹿児島大学理学部卒業，同年日本ユニバック(株)入社。シリーズ90でのフィールド・サポート，リアル・タイム・パッケージの開発に従事，56年日本語ソフトウェア開発部に転属，現在ソフトウェア二部に所属，日本語文書処理システムの開発，主に日本語文入力を中心とするプログラム開発を担当。



小山 憲一 (Kenichi Oyama)

昭和44年東京農工大工学部卒業，45年日本ユニバック(株)入社。シリーズ90でのアプリケーション・プログラムの開発に従事，55年日本語ソフトウェア開発部に転属，日本語文書処理システム開発に従事，59年テクニカル・パブリケーション室に転属，ソフト・ドキュメンテーション担当。



報告 プログラム言語の日本語化実験

—日本語 PL/I の作成—

PL/I Programming in Japanese

真 田 正 二

要 約 日本語によるプログラミングが脚光を浴びている。本稿は、既存プログラム言語の日本語化の実験報告である。言語としては PL/I を選んだ。

PL/I の文は、英語で「動詞 [補助語 (引き数)]……;」の形をしているが、これを日本語で「名詞 補助語 (引き数) …… [を行なう];」の形にし、キーワードの1対1変換だけで、それなりの日本語文らしく見せられた。

あるプログラム例をさまざまな段階で日本語化し、アンケート調査を実施した。好意的な意見もあるが、多くの人は既存言語の日本語化は、リテラル、注釈、使用者定義名を日本語にできればよいと考えており、キーワードまでの日本語化は望んでいない。一方、著者自身が日本語 PL/I を使いこなした経験から、キーワードの日本語化も望みありとの感触を得ている。

Abstract People who speak languages rather than English would have latent expectation of writing programs in their own native languages. This paper reports an experience on development and usage of a high level programming language, PL/I, based on Japanese keywords, literals, comments and variable names rather than English ones.

The typical form of a statement of PL/I in English is "Verb [auxiliary-word (argument)]…;"

Due to grammatical differences between Japanese and English, the natural translation of the statement form would require the reordering of the keyword sequence since the verbs usually come last in the Japanese sentences. This reordering will entirely change the aspect of PL/I language so that the resulting language will not be called PL/I any more. In the Japanese PL/I development, one-to-one translation of English and Japanese keywords has been tried. The resulting form of a "Japanese PL/I statement looks like as follow.

Noun [auxiliary-word (argument)]…[auxiliary-verb];" where "auxiliary-verb" is usually omitted. This translation remain Japanese PL/I still in PL/I, and the statement still in Japanese.

Although program readability is increased, Japanese PL/I has disadvantage in efficient program development. Keyword and variable names written in Japanese require input of KANJI characters picked up from 12,000 of the Japanese character set at terminal upon initial program input. This would be a very tiresome job. A processor which helps to input of Japanese PL/I program is developed and its effects is also discussed.

Looking at the result of inquiries about Japanese PL/I sample programs, most people do not want Japanese keywords, although they like literals, comments and variable names in Japanese. From my own experience of using Japanese PL/I, however, Japanese keywords really helped debugging programs. I feel this one-to-one translation technique would be a good way of programming in Japanese.

1. は じ め に

日本語によるプログラミングが脚光を浴びている^[11,12,14]。日本語によるプログラミングとは、日本語を母体としたプログラム言語を用いて、プログラムを作成することである。その最大の利点は、記述されたプログラムが読みやすく、保守性が向上するというものである。また、母国語でプログラムを作成したいという欲求が潜在的に存在するからである。一方、漢字コードの入力は、英数カナ・コードの入力に比べ繁雑であり、プログラ

ムの入力が困難で、余分な開発費用を負担しなければならないという問題もある。

日本語を母体とするプログラム言語の作り方には、次の二つが考えられる。

- 1) 新たな日本語プログラム言語の開発
- 2) 既存プログラム言語の日本語化

まず新言語の開発についていえば、日本語あるいは英語のいずれを母体にするにしても、既存プログラム言語と比肩しうる能力をもつ言語を新たに開発するのは、大変な作業である。今後の方向としては大いに期待できるものの、まだ研究段階にあるとみてよい。

一方、既存プログラム言語の日本語化の方は、すでに実績をもつプログラム言語の日本語化であるため、言語能力としては問題がなく、日本語化のみに努力を集中すればよい。すでに漢字 CORAL^[1]、日本語 AFL^[2]、和漢^[15]が商品化されているのでみられるように、一部実用化段階に入ったといえる。ただ、言語の国際標準が存在する COBOL、FORTRAN、PL/I 等の普及度の高い言語では、日本語化に対する JIS 標準が制定されていない現在、独自に日本語化すれば他機種との互換性を失うおそれがあるためか、せいぜい使用者定義名を日本語化している製品^[3,4,5,19]が発表されている程度である。しかし、Pascal^[6]の日本語化の報告があるように、企業や大学で研究に取り組んでいると思われる。いずれその成果が発表され、実用化されることは想像に難くない。

このような状況を踏まえ、既存プログラム言語の日本語化の研究に着手するのは意味のあることである。最近、シリーズ 1100 PL/I^[7] (以下 PL/I と呼ぶ) について、日本語化実験を行なった。日本語化といっても、データとして漢字コードを使える機能、プログラムのテキスト表現として日本語が使える機能と、いろいろな段階がある。この段階を、日本ユニパック(株)内部で用いている次の日本語化規準に従って考えた。

- レベル 1 —— 日本語データの定義と処理
- レベル 2 —— 日本語リテラル
- レベル 3 —— 日本語注釈
- レベル 4 —— 日本語メッセージ
- レベル 5 —— 日本語の名前
- レベル 6 —— 日本語のキーワード

PL/I の日本語化に当たっては、レベル 1～6 のすべてについて試みたが、本稿ではプログラムのテキスト表現の日本語化という観点から、とくにレベル 3、5 および 6 に焦点を合わせ、日本語化の方法、効果、問題点、結果分析について述べる。

2. PL/I の日本語化の手段

PL/I は ANS および ISO の国際標準となっているプログラム言語であるが、まだ JIS 標準はない。

PL/I は汎用言語であり機能は多いが、文法は COBOL や FORTRAN に比べ素直であり、日本語化に当たってその処理プロセッサを PL/I コンパイラに対するプリプロセッサとして作りやすい。

日本語化の手段としては、シリーズ 1100 MACRO^[8,9]でプリプロセッサを作り、日本語 PL/I 言語から PL/I 言語への変換を行なわせる方式をとった。変換後のプログラムは純粋な PL/I 言語であるため、PL/I コンパイラで翻訳し、実行することができる。PL/I コンパイラは、MACRO システムとのインタフェース機能を備えているので、次の @ PL1 制御文で、プリプロセスと翻訳を連続して行なわせることができる。

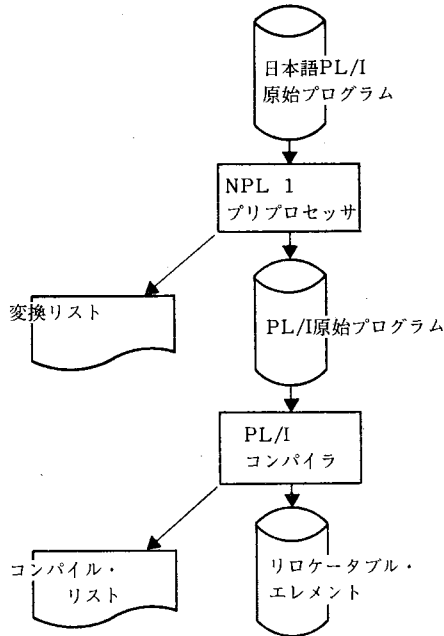


図 1 日本語 PL/I の翻訳の過程

Fig. 1 Compilation process of Japanese PL/I

@PL 1, M si, ro, so, file. NPL 1

ここでMはプロセスを行なわせるオプション, file. NPL 1 はプリプロセッサとなる MACRO プログラムのオムニバス・エレメントの名前である。

MACRO はテキスト処理に適した高水準の言語であり、一般にコーディング量は他の手続き向き言語に比べて少なくすむ。ちなみに、日本語 PL/I プリプロセッサ NPL 1 を作成するのに要した MACRO プログラムの行数は、10 パーセント程度の注釈を含めて、823 行である。

図 1 は NPL 1 による日本語 PL/I プログラムの翻訳過程を示したものである。

3. 日本語化の方法

PL/I プログラムのテキストを日本語化するためには、日本語化レベル 3, 5 および 6 の日本語注釈, 日本語名, 日本語キーワード実現しなければならない。

3.1 注 釈

PL/I の注釈

/*……………*/

の中に、ASCII/ISO 文字ならびに漢字のいずれをも含められるようにした。

3.2 日本語名

PL/I の識別子として、漢字コードの列からなる識別子を許した。ASCII/ISO 文字と漢字の混在する識別子は、ASCII/ISO 文字に対応する漢字が必ず存在することから、不要であるとの見解をとった。

PL/I の識別子の長さは 31 文字まで許されるが、日本語名の場合、漢字の情報量がアルファベットやカナ文字に比べてずっと多いので、約半分の 15 文字までとした。

3.3 キーワード

また、キーワードを日本語化する場合、語順、品詞、訳語の選択等、さまざまな問題を解決しなければならない。

3.3.1 語順と品詞

PL/I の文を分析すると、多くは次の READ 文、

READ FILE (ファイル名) INTO (変数名) KEY (式);

のように、

動詞 [補助語 (引き数)]……; (A)

の形、すなわち動詞のあとに引き数を伴う補助語がいくつか続く形式をとっている。文の形式をとっているが、見方を変えれば次の手続き呼び出し、

CALL READ (ファイル名, 変数名, 式);

と同値である。補助語は、形態はさまざまであるが、動詞で示される手続きに引き数を渡す役目を果たしているにすぎない。引き数の数や順序が可変なので、前置詞や名詞の補助語を使って、文の形式をとっているだけである。

補助語の中には、目的語となるもの、前置詞を伴う副詞の役割を果たすもの等、いろいろな品詞が含まれ、多くの場合、その語順は任意である。これらをそれぞれの品詞を生かすように、忠実に日本語に置き換えるには、

(ファイル名)ファイルから (変数名)へ (式)キーで読み込む;

のように、

[(引き数) 補助語]……動詞; (B)

とするのが自然であろう。しかし、(B)では動詞の位置、および、引き数と補助語の二つの点で(A)に比べて語順が逆転している。

文献でみる日本語化の例は^[6,14,15]、この語順の逆転を積極的に利用して、自然な日本語らしくする努力を払っている。ここでは、若干不自然でもよいから、語順を変えないで日本語化する工夫はないかと考案することにした。

まず、動詞の語順について、日本語では、

— — — ~ する
(動詞)

の形の文は補助動詞「行なう」を用いて、

~ を — — — で行なう
(名詞)

という形で表現することも可能である。READ 文の例では、

読み込みを (ファイル名) ファイルから (変数名) へ (式) キー で行なう

となる。最後の「で行なう」という補助動詞は多くの文に共通し、決まり文句で不要と考えられるので省略し、「を」という助詞も省略すると、

読み込み (ファイル名) ファイルから (変数名) へ (式) キー

とすることができる。すると、一般に(A)の形式の英語の文は、次の(C)の形式の日本語の文に置き換えることができる。

名詞 [(引き数) 補助語]……; (C)

このようにして、英語の命令文における語頭の動詞は、名詞化するだけで語順を変えずに日本語化できる。

次に補助語と引き数の関係をみると、英語では

補助語 (引き数)

の順であるのに、日本語では

(引き数) 補助語

の順となる。英語の補助語には、「KEY」、「FILE」のように名詞が使われる場合と、「INTO」、「TO」、「BY」、「WHILE」等のように前置詞や接続詞が使われる場合がある。前者の名詞の場合には日本語でも、

キー (式)

のように補助語を前に置いても構わない。問題は後者の前置詞や接続詞の場合である。これらは忠実に日本語化すると、「へ」、「まで」、「ずつ」、「なら」のように、助詞や助動詞として、後置の語に訳さなければならない。そこで、前における修飾語を補って、

(変数名) へ

は、

変数 (変数名) へ

という形にしてみた。次いで、助詞「へ」を省略して、

変数 (変数名)

とし、語順が一致した。しかし、この方法では、

FROM (変数名)

もやはり

変数 (変数名)

となり、曖昧である。解決策は修飾語を工夫することによって得られる。この例の場合、

INTO (変数名) → 入力変数 (変数名)

FROM (変数名) → 出力変数 (変数名)

のように、曖昧さを解消すればよい。

このように、訳語の工夫次第で、補助語と引き数の関係も同一語順にすることができ、日本語 PL/I の文の語順は

名詞 [補助語 (引き数)]……; (D)

の形とすることができた。こうして英語と日本語のキーワードの品詞こそ変わるものの、語順を変えずに1対1の逐語訳ができるという見通しが立った。なお、READ 文は、

読込み ファイル (ファイル名) 入力変数 (変数名) キー (式);

に変換される。

3.3.2 訳 語

キーワードの1対1の変換の見通しが立つと、あとはいかに訳語を選択して、自然な日本語にするかが鍵となる。

PL/I のキーワードの数は非常に多く、中には訳しにくいものもあり、解説書^[7]でもカタカナですませているものがある。今回は、できる限りカタカナを使わずに漢字を使う方針をとった。漢字を使えば、ほとんどの単語を2字あるいは3字で表現することができ、字数からいえばカタカナより有利である。

このため、事務処理関連の用語は簿記用語を大幅に採り入れた。「帳簿」,「開簿」,「閉簿」,「記帳」,「転記」,「複写」はそれぞれ「FILE」,「OPEN」,「CLOSE」,「PUT」,「GET」,「COPY」の訳である。

ただし、行き過ぎないように、「BIT」は「微字」ではなく「ビット」,「LABEL」は「名札」ではなく「ラベル」である。とはいえ、「KEY」は「キー」ではなく「鍵」,「DATA」は「データ」ではなく「算料」とする等、慣用と JIS 標準の使い分けを微妙に行った。

漢字で通そうとして、途中で変更したものもある。IF 文については、

IF 式 THEN 文；

ELSE 文；

というのを当初は、

条件 式 成立時 文；

不成立時 文；

と訳したが、文体が固くなったため、ひらがなを使って

もし 式 ならば 文；

なければ 文；

に修正した。おかげで、漢字かな交じり文の雰囲気が出て全体が固苦しくなく、いかにも日本語的なプログラムが書けるようになった。

前項で述べたように、前置詞については大胆な意識を行って、名詞の修飾語に訳した。「TO」,「BY」,「WHILE」,「INTO」,「FROM」の訳はそれぞれ「終値」,「増分」,「反復条件」,「入力変数」,「出力変数」である。

他に大胆な意識を行なった例は、「DO-END」の対、「GOTO」,「CALL」,「ALLOCATE」等である。PL/I の「DO」は繰り返しを意味する場合と意味しない場合の両方があるので、「反復」のような繰り返しを意味する訳語を選べない。最初は「DO」をゴロ合わせて「動作」,「END」を「終了」と訳した。文献を調べるうちに、Pascal の日本語化の例^[6]で「END」を「以上」と訳す例をみつけた。これを発展させ、「DO」を「以下」,「END」を「以上」と訳すことにした。

例として、

DO I=1 TO 100 BY 1；

A(I)=I；

END；

は、

以下 I=1 終値 100 増分 1；

A(I)=I；

以上；

と訳せる。使い慣れるうちに、「以下」と「以上」の対は次第になじみが深まり、適訳の一つと考えている。

「GOTO」は、「へ飛ぶ」という訳が多いようであるが、過去、日本 IBM 社のカナ PL/I の文献で「ツギハ」と訳されていたのを非常に感心した記憶があり、迷わず「次は」と訳した。

「CALL」は「呼出し」とか「実行」を考えたが、落ちつかず、最終的にはサブルーチンによる処理を行なうことからヒントを得て、「処理」と訳した。

「ALLOCATE」は、「割付け」であるが、他の動詞をよい日本語訳した後では、動詞的

で気に入らないので、BASED 変数の新世代を作る文であることから「世代」と意識した。あるいは「生成」が適訳かもしれない。

属性や環境を表すキーワードは、もともと名詞を並べたものに過ぎないので、日本語に訳す場合も名詞に訳すだけでとくに不自然ではない。

3.3.3 補助動詞と助動詞

3.3.1 の語順の項で、補助動詞「で行なう」を省略したが、日本語 PL/I のプログラム例のレビューを受けたところ、動詞が存在しないことを必ず指摘された。そこで、補助動詞を入れることを考えた。すでに他のキーワードで1対1の対応ができていたので、補助動詞は本来無用の語である。NPL1 プリプロセッサは、補助動詞が存在しても、PL/I に変換するときには空の語に置き換えることにした。

1. ALIGNED	整列	71. FOFL	固定	あふれ	141. PARM	引数
2. ALLOC	世代	72. FORMAT	書式		142. PARTITIONED	分割
3. ALLOCATE	世代	73. FREE	解放		143. PIC	模様
4. AUTO	自動	74. FROM	出力	変数	144. PICTURE	模様
5. AUTOMATIC	自動	75. GENERIC	汎用		145. POINTER	指標
6. BACKWARD	逆転	76. GENKEY	汎用	鍵	146. POS	位置
7. BASED	基	77. GET	転記		147. POSITION	位置
8. BEGIN	開始	78. GOTO	次は		148. POST	完了
9. BIN	二進	79. IF	もし		149. PREC	精度
10. BINARY	二進	80. IGNORE	無視		150. PRECISION	精度
11. BIT	ビット	81. IN	領域	変数	151. PRINT	印刷
12. BLKSIZE	ブロック長	82. INIT	初期	値	152. PROC	手続き
13. BOTMARGIN	下部	83. INITIAL	初期	値	153. PROCEDURE	手続き
14. BUFFERS	緩衝	84. INPUT	入力		154. PROTECT	保護
15. BUFFOFF	緩衝	85. INT	内部		155. PTR	指標
16. BUILTIN	組み込み	86. INTERNAL	内部		156. PUT	指標
17. BY	増加	87. INTO	入力	変数	157. READ	読込
18. CALL	呼び	88. KEY	鍵		158. REAL	実数
19. CHAR	文字	89. KEYED	鍵	付き	159. RECORD	記録
20. CHARACTER	文字	90. KEYFROM	出力	番号	160. RECSIZE	記録長
21. CLEAR	未定	91. KEYNO	鍵	番号	161. RECURSIVE	再参照
22. CLOSE	閉	92. KEYTO	鍵	入力	162. REFER	参照
23. COL	欄	93. LABEL	ラベル		163. RELEASE	解放
24. COLUMN	欄	94. LIKE	類似		164. REPEAT	反復
25. COMPLEX	複素	95. LINE	行		165. RETURN	戻り
26. CON	条件	96. LINESIZE	行幅		166. RETURNS	戻り
27. CONB	条件	97. LIST	並列		167. REVERT	戻り
28. CONDITION	条件	98. LOCAL	局部		168. REWIND	再進出
29. CONNECTED	連結	99. LOCATE	配置		169. REWRITE	再進出
30. CONSECUTIVE	連続	100. LOCK	施錠		170. SEQL	順序
31. CONSTANT	定数	101. MAIN	主		171. SEQUENTIAL	順序
32. CONTROLLED	制御	102. MAXKEY	最大	鍵	172. SET	設定
33. CONV	変換	103. MEMBER	構成	要素	173. SIGNAL	信号
34. CONVERSION	変換	104. MKEY	索引		174. SIZE	寸法
35. COPY	複製	105. NAME	名前		175. SKIP	スキップ
36. CPLX	複素	106. NCHAR	日本語	字	176. SNAP	静ナップ
37. CTL	制御	107. NCHARACTER	日本語	字	177. STATIC	静的
38. DATA	資料	108. NEW	新		178. STOP	停止
39. DCL	宣言	109. NOCONV	変換	否定	179. STORAGE	記憶領域
40. DEC	十進	110. NOCONVERSION	変換	否定	180. STREAM	流れ
41. DECIMAL	十進	111. NOFIXEDOVERFLOW	固定	あふれ	181. STRG	列範
42. DECLARE	宣言	112. NOFOFL	固定	あふれ	182. STRING	列
43. DEF	定義	113. NONE	なし		183. STRINGRANGE	列範
44. DEFAULT	省略	114. NONVARYING	不変		184. STRINGSIZE	列範
45. DEFINED	略	115. NOOFL	あふれ	否定	185. STRUCTURE	構造
46. DELETE	削除	116. NOOVERFLOW	あふれ	否定	186. STRZ	列範
47. DFT	略	117. NOSTRG	列範	否定	187. SUBSCRIPTRANGE	列範
48. DIM	次元	118. NOSTRINGRANGE	列範	否定	188. SYM	印刷
49. DIMENSION	次元	119. NOSTRINGSIZE	列範	否定	189. SYSTEM	システム
50. DIR	直接	120. NOSTRZ	列範	否定	190. TAB	タブ
51. DIRECT	直接	121. NOSUBRG	派字	否定	191. TASK	タスク
52. DISCONNECTED	非連結	122. NOSUBSCRIPTRANGE	派字	否定	192. THEN	ならば
53. DO	以下	123. NOUFL	下位	あふれ	193. TITLE	題名
54. EDIT	編集	124. NOUNDERFLOW	下位	あふれ	194. TO	以下
55. ELSE	なければ	125. NOWRITE	出力	なし	195. TOPMARGIN	上部
56. END	以上	126. NOZDIV	零除	否定	196. TRANSMIT	転送
57. ENDFILE	終り	127. NOZERODIVIDE	零除	算定	197. UFL	下位
58. ENTRY	入口	128. OFFSET	相対	指標	198. UNAL	下位
59. ENV	環境	129. OFL	あふれ		199. UNALIGNED	不整合
60. ENVIRONMENT	環境	130. OLD	旧		200. UNDEFINEDFILE	未定義
61. ERROR	誤り	131. ON	登録		201. UNDERFLOW	未定義
62. EVENT	事象	132. OPEN	開		202. UNDF	未定義
63. EXT	外部	133. OPTIONS	選別		203. UNLOCK	解放
64. EXTEND	拡張	134. OUTPUT	出力		204. UPDATE	更新
65. EXTERNAL	外部	135. OVERFLOW	あふれ		205. VALIDATE	正当性
66. FILE	ファイル	136. PAGE	改頁		206. VAR	変数
67. FINISH	完了	137. PAGESIZE	頁幅		207. VARIABLE	変数
68. FIXED	固定	138. PAGESIZE	頁幅		208. VARYING	変数
69. FIXEDOVERFLOW	固定	139. PAR	分割		209. WAIT	同期
70. FLOAT	浮動	140. PARAMETER	変数		210. WHILE	反復
					211. WRITE	再送
					212. ZDIV	零除

図 2 英和キーワード対応表
Fig. 2 English-Japanese keyword list

文によって補助動詞は変える方がよいと思われるので、次の補助動詞を設けた。

- | | | |
|---------|---|--------|
| で実行する…… | } | 一般の文 |
| を実行する…… | | |
| で行う…… | | |
| を行う…… | | |
| で繰返す…… | | DO 文 |
| へ行く…… | | GOTO 文 |
| を計算する…… | | 代入文 |
| 何もしない…… | | 空文 |

同様の主旨で、より日本語らしくするために、DO 文用に次の三つの助詞を指定可能とした。

1. あふれ	OVERFLOW	71. 行幅	LINESIZE	141. 汎用	GENKEY
2. あかす	NOOVERFLOW	72. 再帰	RECURSIVE	142. 汎用	CONTROLLED
3. つかす		73. 再再	REWRITE	143. 汎用	DISCONNECTED
4. 通す		74. 最大値	MAXKEY	144. 汎用	UNALIGNED
5. 通す		75. 削除	DELETE	145. 汎用	NONVARYING
6. 通す		76. 索引	MKEY	146. 汎用	FLOAT
7. 通す		77. 参照	REFER	147. 汎用	RETURN
8. 通す	ELSE	78. 参照	DATA	148. 汎用	COPY
9. 通す	NONE	79. 参照	POINTER	149. 汎用	COMPLEX
10. 通す	THEN	80. 参照	SET	150. 汎用	PARTITIONED
11. 通す		81. 参照	LOCK	151. 汎用	CHARACTER
12. 通す		82. 参照	EVENT	152. 汎用	LIST
13. 通す	IF	83. 参照	GOTO	153. 汎用	CLOSE
14. 通す		84. 参照	DIMENSION	154. 汎用	PAGESIZE
15. 通す		85. 参照	AUTOMATIC	155. 汎用	PAGESIZE
16. 通す		86. 参照	REAL	156. 汎用	CONVERSION
17. 通す		87. 参照	MAIN	157. 汎用	NOCONVERSION
18. 通す	SYSTEM	88. 参照	PROCEDURE	158. 汎用	VARIABLE
19. 通す	SNAP	89. 参照	TO	159. 汎用	EDIT
20. 通す	TASK	90. 参照	ENDFILE	160. 汎用	RETURNS
21. 通す	TAB	91. 参照	DECIMAL	161. 汎用	PROTECT
22. 通す	BIT	92. 参照	OUTPUT	162. 汎用	CLEAR
23. 通す	BLKSIZE	93. 参照	NOWRITE	163. 汎用	UNDEFINEDFILE
24. 通す	LABEL	94. 参照	KEYFROM	164. 汎用	IGNORE
25. 通す	DO	95. 参照	FROM	165. 汎用	NAME
26. 通す	END	96. 参照	SEQUENTIAL	166. 汎用	PICTURE
27. 通す	POSITION	97. 参照	CALL	166. 汎用	COLUMN
28. 通す	SYM	98. 参照	INITIAL	168. 汎用	STREAM
29. 通す	PRINT	99. 参照	WRITE	169. 汎用	IN
30. 通す	UNDERFLOW	100. 参照	FORMAT	170. 汎用	LIKE
31. 通す	NOUNDERFLOW	101. 参照	DEFAULT	171. 汎用	ZERODIVIDE
32. 通す	BOTMARGIN	102. 参照	TOPMARGIN	172. 汎用	NOZERODIVIDE
33. 通す	PARAMETER	103. 参照	CONDITION	173. 汎用	STRING
34. 通す		104. 参照	NEW	174. 汎用	STRINGSIZE
35. 通す	VARYING	105. 参照	SIZE	175. 汎用	NOSTRINGSIZE
36. 通す	RELEASE	106. 参照	ALLOCATE	176. 汎用	STRINGRANGE
37. 通す	UNLOCK	107. 参照	ALIGNED	177. 汎用	NOSTRINGRANGE
38. 通す	FREE	108. 参照	VALIDATE	178. 汎用	CONNECTED
39. 通す	SKIP	109. 参照	PRECISION		
40. 通す	PAGE	110. 参照	STATIC		
41. 通す	BEGIN	111. 参照	DECLARE		
42. 通す	OPEN	112. 参照	OPTIONS		
43. 通す	EXTERNAL	113. 参照	BUILTIN		
44. 通す	EXTEND	114. 参照	OFFSET		
45. 通す	REWIND	115. 参照	BY		
46. 通す	FINISH	116. 参照	SIGNAL		
47. 通す	POST	117. 参照	TITLE		
48. 通す	ENVIRONMENT	118. 参照	FILE		
49. 通す	BUFFERS	119. 参照	DIRECT		
50. 通す	BASED	120. 参照	STOP		
51. 通す	STORAGE	121. 参照	DEFINED		
52. 通す	PUT	122. 参照	CONSTANT		
53. 通す	RECORD	123. 参照	SUBSCRIPTRANGE		
54. 通す	RECSIZE	124. 参照	NOSUBSCRIPTRANGE		
55. 通す	BACKWARD	125. 参照	GET		
56. 通す	OLD	126. 参照	TRANSMIT		
57. 通す	LOCAL	127. 参照	ON		
58. 通す	BUFFOFF	128. 参照	WAIT		
59. 通す	KEY	129. 参照	READ		
60. 通す	KEYTO	130. 参照	INTERNAL		
61. 通す	KEYNO	131. 参照	BINARY		
62. 通す	KEYED	132. 参照	NCHARACTER		
63. 通す	FIXED	133. 参照	ENTRY		
64. 通す	FIXEDOVERFLOW	134. 参照	INPUT		
65. 通す	NOFIXEDOVERFLOW	135. 参照	INTO		
66. 通す	ERROR	136. 参照	LOCATE		
67. 通す	UPDATE	137. 参照	REVERT		
68. 通す	MEMBER	138. 参照	WHILE		
69. 通す	STRUCTURE	139. 参照	REPEAT		
70. 通す	LINE	140. 参照	GENERIC		

図 3 和英キーワード対応表

Fig. 3 Japanese-English keyword list

{ から
 ずつ
 まで

NPL 1 は、これらの補助動詞や助詞を文脈に関係なく、単に空の語に置き換える方法を採用しているので、適当と思われる位置に使用者が指定すればよい。

これらを指定すると確かにより日本語らしくなるが、本来は無用の語である。COBOLにおける「IS」や「TO」のような省略可能な語が、プログラムを英語らしくするため用意されていても、現実にはほとんどのプログラマが省略してしまうように、日本語 PL/Iにおいても、慣れていくほど実際に利用するプログラマは少なくなるであろう。

3.3.4 英和・和英キーワード一覧表

以上の結果として得られた、英語と日本語のキーワードの一覧表を図2と図3に示す。図2と図3で数が違うのは、英語のキーワードには略語が存在することと、1対0変換の補助動詞と助動詞が図2に現われないことによる。

4. プログラムの日本語化の例

日本語 PL/I の効果を示すために、短い PL/I プログラムを日本語化した例を図で示す。

図5は、日本語化しないもとのプログラムである。識別子や注釈はローマ字、リテラルはカタカナを用いている。

図6は、識別子、注釈およびリテラルを日本語化したものである。割合いからいえば、圧迫的に多い英数字、特殊文字の中に漢字が埋もれている印象を受ける。

図7は、識別子、注釈、リテラルおよびキーワードを日本語化したものである。漢字が多くなり、やや固い印象となる。

図8は、図7に加えて補助動詞や助動詞を追加したものである。補助動詞とはいえ動詞が存在することと、かなの割合が増え漢字かな交じり文の効果が出たこと等、より日本語らしくなった。

パソコン販売実績

```

100 :                                     *@
    :                                     *@
  90 :                                     *@
    :                                     *@
  80 :                                     *@
    :                                     *@
  70 :                                     *@
    : *                                     #
    : *                                     #*
  60 : *                                     #*
    : *                                     #*
  50 : *                                     #*
    : *                                     #*
  40 : **                                     #*
    : ** # @ #* @ *@ *@ * * #*
    : ** # @ #* @ *@ *@ * * #*
  30 : ** @ # @ #* @ *@ *@ * * #*
    : # #* @ #* @ #* @ *@ #* @ * * #* @
  20 : # # @ #* @ #* @ #* @ *@ #* @ *@ #* @ #* @
    : # #* @ #* @ #* @ #* @ #* @ #* @ *@ #* @ #* @
  10 : # #* @ #* @ #* @ #* @ #* @ #* @ #* @ #* @
    : #* @ #* @ #* @ #* @ #* @ #* @ #* @ #* @
=====
ABA123 ABA124 ABA125 SS5021 SS5023 PK5110 Z70-24 Z70-25 Z90-02 Z90-04
( 10) ( 27) ( 49) ( 42) ( 72) ( 12) ( 29) ( 0) ( 72) ( 15)
( 5) ( 12) ( 71) ( 25) (124) ( 44) ( 72) ( 40) ( 63) ( 20)
( 8) ( 16) ( 34) ( 40) (132) ( 59) ( 50) ( 21) ( 54) ( 29)
  
```

図4 図5~8のプログラムの実行結果
 Fig. 4 Execution result of program Fig. 5~8

```

@PL1,SKEM REIDAI1,REIDAI,,U.NPL1
PL1 9R1A-1A 74R1-H 09/01/83 10:17:32 (2)
日本語 PL/I 4R2B 83/09/01 10:17:44
1  /***** PASOKON NO URIAGE BOU GRAPH *****/
2
3  PASOKON_URIAGE:
4  PROCEDURE OPTIONS(MAIN);
5  DECLARE
6  HANBAI PRINT FILE ENVIRONMENT(APRINT),
7  1 HANBAIYOU(10),
8  2 KATAMEI CHARACTER(6),
9  2 DAISUU(3) FIXED DECIMAL(4,0),
10 (M,I,J) FIXED DECIMAL(5,0),
11 KIGOU(3) CHARACTER(1) INITIAL('#','*','@');
12 PUT FILE(HANBAI) PAGE EDIT('ハロウコンハロウイジョウヒキ')
13 (X(5),A(17));
14 PUT FILE(HANBAI) SKIP(4);
15 /* DATA O YOMU */
16 GET EDIT(HANBAIYOU)(COLUMN(1),A(6),3 F(4));
17 /* GRAPH O KAKU */
18 DO M = 100 TO 5 BY -5;
19 /* MEMORI */
20 IF MOD(M,10) = 0
21 THEN
22 PUT FILE(HANBAI) SKIP EDIT(M,')(F(6),A(2));
23 ELSE
24 PUT FILE(HANBAI) SKIP EDIT('')(A(8));
25 /* GRAPH NO ZOU */
26 DO I = 1 TO 10 BY 1;
27 PUT FILE(HANBAI) EDIT('')(A(2));
28 DO J = 1 TO 3 BY 1;
29 IF HANBAIYOU(I).DAISUU(J) >= M
30 THEN
31 PUT FILE(HANBAI) EDIT(KIGOU(J))(A(1));
32 ELSE
33 PUT FILE(HANBAI) EDIT('')(A(1));
34 END;
35 PUT FILE(HANBAI) EDIT('')(A(2));
36 END;
37 END;
38 /* KATAMEI */
39 PUT FILE(HANBAI) SKIP EDIT(' 0',(71)'=')(A(6),A(71));
40 PUT FILE(HANBAI) SKIP EDIT(KATAMEI)(X(7),10 A(7));
41 DO J = 1 TO 3 BY 1;
42 PUT FILE(HANBAI) SKIP EDIT('')(A(7));
43 DO I = 1 TO 10 BY 1;
44 PUT FILE(HANBAI) EDIT('(',DAISUU(I,J),')')(A(2),F(3),A(2));
45 END;
46 END;
47 END; /* PASOKON_URIAGE */
END NPL1 0 ERRORS

```

リテラル——カタカナ
 注 釈——ローマ字
 識 別 子——ローマ字
 キーワード——英 語

図 5 もとのプログラム

Fig. 5 Original program

図 4 は、図 5～8 の実行結果 (いずれも同じ) である。プログラムを読む参考にされた
 い。

これらのプログラムは、PL/I の日本語化に対する好みを調査するためのアンケート用
 の資料として用いたものである。アンケート結果については、6 章で述べる。

なお、この種の例題は短かすぎて、十分な説得力をもたないと考えられる。そこで、あ
 る程度長いプログラムの例を付録 A (一部略) に示す。この長いプログラムの開発経験に
 より、筆者は、PL/I の日本語化の程度は図 7 程度 (識別子, 注釈, リテラル, キーワ
 ードの日本語化) で十分であり、かつ図 6 の程度よりもデバッグ中の論理追求が楽である
 という感触を得た。

5. 入 力 問 題

日本語によるプログラミングでは、プログラムの入力の問題を避けて通ることはできな

```

@PL1,SKEM REIDAI1A,REIDAI,,U.NPL1
PL1 9R1A-1A 74R1-H 09/01/83 10:19:22 (0)
日本語 PL/I 4R2B 83/09/01 10:19:52
1  /***** パソコンの売り上げ棒グラフ *****/
2
3  パソコン売り上げ:
4  PROCEDURE OPTIONS(MAIN);
5  DECLARE
6  販売 PRINT FILE ENVIRONMENT(APRINT),
7  1 販売表(10),
8  2 型名 CHARACTER(6),
9  2 台数(3) FIXED DECIMAL(4,0),
10 (M,I,J) FIXED DECIMAL(5,0),
11 記号(3) CHARACTER(1) INITIAL('#','*','@');
12 PUT FILE(販売) PAGE EDIT(MODE3,'パソコン販売実績',
13 MODE0)
14 (X(5),A,N(8),A);
15 PUT FILE(販売) SKIP(4);
16 /* データを読む */
17 GET EDIT(販売表)(COLUMN(1),A(6),3 F(4));
18 /* グラフを畫く */
19 DO M = 100 TO 5 BY -5;
20 /* 日盛り */
21 IF MOD(M,10) = 0
22 THEN
23 PUT FILE(販売) SKIP EDIT(M,' :')(F(6),A(2));
24 ELSE
25 PUT FILE(販売) SKIP EDIT(' :')(A(8));
26 /* グラフの做 */
27 DO I = 1 TO 10 BY 1;
28 PUT FILE(販売) EDIT('')(A(2));
29 DO J = 1 TO 3 BY 1;
30 IF 販売表(I).台数(J) >= M
31 THEN
32 PUT FILE(販売) EDIT(記号(J))(A(1));
33 ELSE
34 PUT FILE(販売) EDIT('')(A(1));
35 END;
36 PUT FILE(販売) EDIT('')(A(2));
37 END;
38 END;
39 /* 型名 */
40 PUT FILE(販売) SKIP EDIT(' 0',(71)'=')(A(6),A(71));
41 PUT FILE(販売) SKIP EDIT(型名)(X(7),10 A(7));
42 DO J = 1 TO 3 BY 1;
43 PUT FILE(販売) SKIP EDIT('')(A(7));
44 DO I = 1 TO 10 BY 1;
45 PUT FILE(販売) EDIT('(',台数(I,J),')')(A(2),F(3),A(2));
46 END;
47 END;
48 END; /* パソコン売り上げ */
END NPL1 0 ERRORS

```

リテラル——日本語
 注 釈——日本語
 識別子——日本語
 キーワード——英語

図 6 キーワード以外の日本語化

Fig. 6 Japanese PL/I except keywords

い。一般のユーザは、英語を母体とするプログラム言語より、日本語を母体とした方がよさそうだと感覚的に理解できたとしても、いざ使う段になって、入力やデバッグのことを考えて、採用に二の足を踏むことが十分予想される。

日本語 PL/I に限らず、日本語によるプログラミングの普及を図るためには、従来のプログラム言語と同程度の便利な入力方式が考案されなければならない。

5.1 日本語プログラムの性質

入力問題を解決するために、日本語化されたプログラムの性質を分析すると、次のようなことがわかる。

1) 日本語名

プログラムの中で同じ名前が何度も繰り返し現われる。試行錯誤的なカナ漢字変換方式を使うよりも、1対1のカナ漢字あるいはローマ字漢字変換方式が向いている。ただし、1対1の変換規則はプログラムごとに変わりうる。

```

@PL1,SKEM REIDAI1B,REIDAI,,U.NPL1
PL1 9R1A-1A 74R1-H 09/01/83 10:26:59 (0)
日本語 PL/I 4R2B 83/09/01 10:27:04
1  /***** パソコンの売り上げ棒グラフ *****/
2
3 パソコン売り上げ:
4 手続き 逐別(主);
5 宣言
6 販売 印書 帳簿 環境(APRINT),
7 1 販売表(10),
8 2 型名 文字(6),
9 2 台数(3) 固定十進(4,0),
10 (M,I,J) 固定十進(5,0),
11 記号(3) 文字(1) 初期値('#','*','@');
12 記帳 帳簿(販売) 改頁 編集(MODE3,'パソコン販売実績',
13 MODE0)
14 (X(5),A,N(8),A);
15
16 記帳 帳簿(販売) 改行(4);
17 /* データを読む */
18 転記 編集(販売表)(欄(1),A(6),3 F(4));
19 /* グラフを書く */
20 以下 M = 100 終値 5 増分 -5;
21 /* 自盛り */
22 もし MOD(M,10) = 0
23 ならば
24 記帳 帳簿(販売) 改行 編集(M, ':')(F(6),A(2));
25 なければ
26 記帳 帳簿(販売) 改行 編集(' :')(A(8));
27 /* グラフの線 */
28 以下 I = 1 終値 10 増分 1;
29 記帳 帳簿(販売) 編集(' :')(A(2));
30 以下 J = 1 終値 3 増分 1;
31 もし 販売表(I).台数(J) >= M
32 ならば
33 記帳 帳簿(販売) 編集(記号(J))(A(1));
34 なければ
35 記帳 帳簿(販売) 編集(' ')(A(1));
36 以上;
37 記帳 帳簿(販売) 編集(' ')(A(2));
38 以上;
39 /* 型名 */
40 記帳 帳簿(販売) 改行 編集(' 0',(71)'=')(A(6),A(71));
41 記帳 帳簿(販売) 改行 編集(型名)(X(7),10 A(7));
42 以下 J = 1 終値 3 増分 1;
43 記帳 帳簿(販売) 改行 編集(' ')(A(7));
44 以下 I = 1 終値 10 増分 1;
45 記帳 帳簿(販売) 編集(' ',台数(I,J),' ')(A(2),F(3),A(2));
46 以上;
47 以上;
48 以上; /* パソコン売り上げ */
END NPL1 0 ERRORS

```

リテラル——日本語
注 釈——日本語
識 別 子——日本語
キーワード——日本語

図 7 キーワードまでの日本語化

Fig. 7 Japanese PL/I including keywords

2) 日本語キーワード

補助動詞を除き、英語のキーワードとの1対1対応が保たれ、しかも同じ言語であれば、この対応関係は不変である。英語キーワードをもとにした1対1英語漢字変換方式が向いている。

3) 日本語注釈

自由な日本語文が現われるので、普通の文章の作成と同様の試行錯誤的なカナ漢字あるいはローマ字漢字変換方式が向いている。

4) 日本語リテラル

プログラム中に同じつづりの日本語リテラルが何度も現われることは少ない。注釈同様、試行錯誤的なカナ漢字あるいはローマ字漢字変換方式で十分である。

5.2 日本語 PL/I プログラム作成用プロセッサ

5.1 節の分析から、日本語名と日本語キーワードについては、1対1変換方式の適用が

```

@PL1,SKEM REIDAI1C,REIDAI,,U.NPL1
PL1 9R1A-1A 74R1-H 09/05/83 11:49:08 (0)
日本語 PL/I 4R2B 83/09/05 11:49:11
1  /***** パソコンの売り上げ棒グラフ *****/
2
3 パソコン売り上げ:
4 手続き 選別(主);
5 宣言
6 販売 印書 帳簿 環境(APRINT),
7 1 販売表(10),
8 2 型名 文字(6),
9 2 台数(3) 固定十進(4,0),
10 (M,I,J) 固定十進(5,0),
11 記号(3) 文字(1) 初期値('#','*','@');
12 記帳 帳簿(販売) 改頁 編集(MODE3,'パソコン販売実績',
13 MODE0)
14 (X(5),A,N(8),A) を行なう;
15 記帳 帳簿(販売) 改行(4) を行なう;
16 /* データを読む */
17 帳記 編集(販売表)(欄(1),A(6),3 F(4)) を行なう;
18 /* グラフを畫く */
19 以下 M = 100 から 終値 5 まで 増分 -5 ずつ で繰返す;
20 /* 目盛り */
21 もし MOD(M,10) = 0
22 ならば
23 記帳 帳簿(販売) 改行 編集(M,'')(F(6),A(2)) を行なう;
24 なければ
25 記帳 帳簿(販売) 改行 編集('')(A(8)) を行なう;
26 /* グラフの條 */
27 以下 I = 1 から 終値 10 まで 増分 1 ずつ で繰返す;
28 記帳 帳簿(販売) 編集('')(A(2)) を行なう;
29 以下 J = 1 から 終値 3 まで 増分 1 ずつ で繰返す;
30 もし 販売表(I).台数(J) >= M
31 ならば
32 記帳 帳簿(販売) 編集(記号(J))(A(1)) を行なう;
33 なければ
34 記帳 帳簿(販売) 編集('')(A(1)) を行なう;
35 以上;
36 記帳 帳簿(販売) 編集('')(A(2)) を行なう;
37 以上;
38 以上;
39 /* 型名 */
40 記帳 帳簿(販売) 改行 編集(' 0',(71)'')(A(6),A(71)) を行なう;
41 記帳 帳簿(販売) 改行 編集(型名)(X(7),10 A(7)) を行なう;
42 以下 J = 1 から 終値 3 まで 増分 1 ずつ で繰返す;
43 記帳 帳簿(販売) 改行 編集('')(A(7)) を行なう;
44 以下 I = 1 から 終値 10 まで 増分 1 ずつ で繰返す;
45 記帳 帳簿(販売) 編集(' ',台数(I,J),'')(A(2),F(3),A(2)) を行なう;
46 以上;
47 以上;
48 以上; /* パソコン売り上げ */
END NPL1 0 ERRORS

```

リテラル——日本語
 注 積——日本語
 識 別 子——日本語
 キーワード——日本語
 補助動詞——あり

図 8 補助動詞, 助動詞を追加

Fig. 8 Japanese PL/I with additional auxiliary verbs

可能である。

そこで、この1対1変換を自動的に行なうプロセッサを作成して、NPL 1 B (Nippongo PL/I Builder) と名付けた。NPL 1 B は、次の機能を備えている。

- 1) 英語のキーワードと日本語のキーワードの対応表を内蔵し、原始プログラム中のすべての英語のキーワードを対応する日本語キーワードに変換する。
- 2) 使用者定義名については、使用者が原始プログラム中に書いた、英語、ローマ字、カタカナ等の名前と日本語名の対応表に従って、以後現われる名前を日本語名に変換する。
- 3) 文字リテラルや注釈中の単語については、上の変換の対象としない。
- 4) キーワードは、オプションにより日本語化するかどうかを選択できる。
- 5) 補助動詞や助動詞については、自動的に補うことはしない。

NPL 1 Bを用いると、使用者定義名の対応表を用意するだけで、プログラムの日本語化が可能となる。残る注釈と日本語文字リテラルだけをカナ漢字変換プログラムで日本語化すればよい。

使用者定義名の対応表を原始プログラム中に含める方法は、次のような特殊な形の注釈（定義注釈という）をプログラムの先頭に書けばよい。

```
/*+
  DEFINE
    KENSA=検査,
    TAISYOU=対象;
+*/
```

NPL 1 Bは、定義注釈により、内部テーブルに対応表を作り上げ、以降はキーワードと同様の操作で、現われる識別子を日本語に変換する。

NPL 1 Bに呼応して、NPL 1にも定義注釈を利用して日本語名から英語またはローマ字への逆変換を行う機能を追加した。それまでは、日本語名は「\$\$\$ 001」のような機械的に生成する名前に変換していたが、この逆変換機能により、変換後の PL/I プログラムのデバッグが以前に比べてずっと楽になった。

付録Aのプログラムは、定義注釈を用いて NPL 1 Bで日本語化したものである。このように長いプログラムであっても、入力の負荷は、対応表を用意することと、注釈と日本語リテラルをカナ漢字変換で入力することだけである。実際、このプログラムの開発中に、入力の負担はほとんど感じられなかった。

NPL 1 B方式による1対1変換方式には、次の長所がある。

- 1) 既存プログラムに対応表を用意するだけで、容易に日本語化できる。
- 2) 辞書ファイルを必要としないので、辞書ファイルの用語に拘束されない。
- 3) 辞書ファイルの検索より効率がよい。

一方、次の短所がある。

- 4) 名前の変換操作により、変換前後の長さの違いから字下げや縦ぞろえが乱される。
- 5) 名前の変換操作により、行の長さが有効な範囲を超えたときの継続行の処理が必要だが、行なっていない。

変換操作による乱れを直すために日本語 PL/I 用の清書機能を用意する等、欠点は克服しなければならないが、NPL 1 Bは十分実用に耐える入力方式を提供しているといえる。また、プログラムの修正作業でも利用できることもわかった。

6. 評 価

6.1 アンケート調査と考察

図4～図8および付録Aのプログラムを資料として、主にソフトウェア開発と保守を担当する者47名に対して、アンケート調査を実施した。アンケート結果の一部を付録Bに示す。なお、アンケート中の図番号はいずれも本稿で用いた図番号と一致する。

さて、今回のアンケートでは、半数以上の人キーワードを日本語化しない図6を最も好み、次いでキーワードを日本語化し、補助動詞や助動詞を追加してより日本語らしくした図8を最も好む人が四分の一を占める。キーワードを逐語訳した図7は、16パーセントの人が最も好むと答えたにすぎない。入力方式については、提案の方式がかなり効率のよい入力方式であるにもかかわらず、やはり大変であると思っていることがわかる。

アンケート調査では、キーワードの逐語訳方式の日本語化は、どちらかといえば否定的なものであった。筆者自身、NPL 1 B から最初に図 7 のようなプログラムが出力されたとき、非常な違和感を抱いた。しかし、同時に、これは「慣れ」の問題であるという認識をもった。誰しも、未知のプログラム言語のテキストに初めて触れた場合、違和感をもつものである。実際、付録 A のプログラムをデバッグしながら開発していくうちに違和感が薄れ、むしろ、図 6 のキーワードが英語のままの段階のリストよりも読みやすく、プログラムの論理も追求しやすいという結論を得た。読みやすさの原因としては、次のことがあげられる。

- 1) キーワードも変数名も同質の言語（日本語）であり、異質の言語（英語と日本語）の場合よりも頭の切り替えが不要で、論理の追求に集中できる。
- 2) キーワードが漢字であると、2～3文字でよく、一般に英語のキーワードより字数が少なく、テキストを読む量が少なく、かつ情報量は同じである。これは、変数名に対しても同様に成り立つ。
- 3) 日本語としての不自然さは、慣れるにつれて気にならなくなる。もともとプログラム言語は人工言語であり、英語版の PL/I といえども自然な英語ではなく、その面でのハンディは日本語 PL/I に存在しない。むしろ日本語である分だけ日本人にはなじみやすい。

今回のアンケートでは、「慣れ」の状態に到達しない、第一印象の調査結果が集まったと考えられるが、今後、実際に使ってくれる協力者を得て、使用上の容易性を継続して調査した。

6.2 将来への展望

ところで、PL/I あるいは他の有名な既存言語（COBOL, FORTRAN, PASCAL 等）のキーワードを、コンパイラ・レベルで将来的に日本語化していくべきであろうか。これには、次の難点を克服していかなければならない。

- 1) キーワードの日本語化の標準（JIS）を確立しないと、異機種間でのプログラムの可搬性が損われる。
- 2) 逐語訳以上の日本語化方式を研究する必要がある。とくに COBOL は他の言語以上に英語らしい言語なので、逐語訳による日本語化の成功率は低いように思われる。
- 3) 英語版と日本語版の二つの言語を覚える必要がないように、日本語のキーワード、たとえば「もし」を「MOSI」や「モシ」のようにローマ字やカタカナで入力する入力方式の確立が必要である。

一方、使用者定義名の日本語化については、単に英字の拡張という捉え方でよいので、標準化はいずれ行なわれるであろうし、入力方式は NPL 1 B の方式で十分であり、またその有用性においては万人の納得のいくところであろう。ただ、NPL 1 のようなプリプロセッサ方式では実行時のデバッグ機能と連動せず、リスト上の名前とデバッグ・セッション時の名前が一致しないので、是非ともコンパイラ本体で日本語名を受け入れるべきである。注釈やリテラルも同様である。

以上を踏まえて、既存言語の日本語化方式が標準化されるまでの近未来のコンパイラ・システムに対して、次の提言をしたい。

- 1) コンパイラは、日本語名、日本語注釈、日本語リテラルを直接受け入れる。
- 2) 言語依存の会話型エディタは、使用者指定の 1 対 1 変換辞書による日本語名入力をサポートする。

- 3) 既存プログラムを日本語化する, NPL1Bのような支援プロセッサを提供する. また, 字下げ, 縦ぞろえ等の清書機能を併わせもつことが望ましい.
- 4) キーワードだけを1対1の逐語訳で日本語化する MACRO プログラムを提供する. これは, 本稿の NPL1 よりも簡単なものになる. また, その逆変換を行う MACRO プログラムを提供し, 日本語化したキーワードを英語に戻す(標準に準拠したプログラムに戻す)ことをできるようにする(1対1の逐語訳であるから, 常に可能である). これは異機種間の可搬性を保証するためである.
- 5) 会話型デバッグ・システムは, 日本語名や日本語リテラルを指定できるようにし, 日本語化したプログラムを日本語でデバッグできるようにする.
- もちろん, 日本語化したからといってプログラムの信頼性が増すわけでもないので, 別途, プログラム構造図, モジュール関連図等, ソフトウェア工学を支援するツール類も用意しなければならない.

7. おわりに

今回の PL/I の日本語化実験は, 日本語によるプログラミングの研究の第一歩である. とくにキーワードの日本語化については, 逐語訳という単純な手法ながら, 品詞と訳語の選択で十分実用に耐える成果が得られたことに意を強くした. また, 日本語の入力方式において, プログラムの入力は日本語文章の入力よりもずっと能率のよい方法が存在しうることを実証できた.

付録 A 日本語 PL/I プログラムの例

```

@PL1,SKERDM KADAI1A,,,U.NPL1
PL1 9R1A-1A 74R1-H 09/02/83 15:11:40 (0)
日本語 PL/I 4R2B 83/09/02 15:11:52
1
2  /**
3  DEFINE
4      KONTENAKANRI          =   コンテナ管理,
5      TUMINIHYO             =   積荷票,
6      KONTENABANGO         =   積コンテナ番号,
7      HANNYUNENGETUNITIJI  =   搬入年月日時,
8      NAIZOHIN             =   内蔵品,
9      HINMEI               =   品名,
10     SURYO                 =   数量,
11     KONTENA               =   コンテナ,
12     TUMINI                =   積荷,
13     SYUKKOKIROKU         =   出庫記録,
14     KARA                  =   空,
15     KONTENASU            =   コンテナ数,
16     SYUSSOIRAI           =   出倉依頼,
17     OKURISAKIMEI         =   送り先名,
18     ZAIKOBUSOKU          =   在庫不足,
19     SEIGYOKIROKU         =   制御記録,
20     CYUMONBANGO          =   注文番号,
21     ZAIKOBUSOKURISUTOSU  =   在庫不足リスト数,
22     HENYUHOJISYORI       =   メニュー表示処理,
23     SYUKKOSYORI          =   出庫処理,
24     NYUKOSYORI           =   入庫処理,
25     ZAIKOBUSOKURISUTOSAKUSEI  =   在庫不足リスト作成,
26     SYURYOSYORI          =   終了処理,
27     NYURYOKUAYAMARISYORI  =   入力誤り処理,
28     SAISIKO              =   再指示,
29     SIREI                 =   指令,
30     ZAIKOKENSA           =   在庫検査,
31     ZAIKONASI            =   在庫なし,
32     ZAIKONASIRENRAKU     =   在庫なし連絡,
33     SYUKKOSIJISYOHAKKO   =   出庫指示書発行,
34     SYUKKO               =   出庫,
35     ZAIKOSU              =   在庫数,

```

36	SYUKKOSU	= 出庫数,
37	CYOSAZUMI	= 調査済み,
38	KINYU	= 記入,
39	ZAIBOBUSOKUNOOOWARI	= 在庫不足の終わり,
40	KAKUNIN	= 確認,
41	ZAIBOBUSOKURISUTOSAIGO	= 在庫不足リスト最後,
42	HANSYUTUMAAKU	= 搬出マーク,
43	ZAIBOBUSOKURISUTONOPAKKU	= 在庫不足リストのバック,
44	NYURYOKUSURYO	= 入力終了,
45	HINSU	= 品数,
46	SOSU	= 総数,
47	CONTAINER	= コンテナファイル,
48	SYUSYORI	= 主処理,
49	ZAIKOFUSOKU	= 在庫不足ファイル,
50	BANGO	= 番号,
51	SAKUSEIOWARI	= 作成終わり,
52	ZAIKOHINITIRANHYOSAKUSEI	= 在庫品一覧表作成;
53	**/	
54	/*	

```

55          *****
56          *                  コンテナ管理                  *
57          *****
58          */
59
60      コンテナ管理:
61      手続き 選別(主);
62      宣言
63      コンテナファイル          記録帳簿;
64      在庫不足ファイル          記録帳簿;
65      宣言
66      1 積荷票,
67      2  コンテナ番号          文字(5),
68      2  搬入年月日時          文字(8),
69      2  FILLER                文字(3),
70      2  内蔵品(10),
71      3  品名                  文字(60),
72      3  数量                  固定十進(5,0),
73      2  総数                  固定十進(5,0),
74      2  品数                  固定十進(2,0);
75      宣言
76      1  コンテナ(50)          類似積荷票,
77      1  出庫記録(50),
78      2  数量                  固定十進(5,0),
79      2  空                    ビット(1),
80      1  積荷                  類似積荷票,
81      コンテナ数              固定二進;
82      宣言
83      1  出倉依頼,
84      2  番号                  模様 '9999' 初期値(0),
85      2  品名                  文字(60),
86      2  数量                  固定十進(5,0),
87      2  送り先名            文字(60),
88      1  在庫不足            類似出倉依頼;
89      宣言
90      1  制御記録,
91      2  コンテナ番号          模様 '99999' 初期値(0),
92      2  注文番号            固定十進(7,0),
93      2  在庫不足リスト最後  模様 '9999' 初期値(0);
94      宣言
95      確認                    文字(2),
96      指令                    文字(1),
97      在庫なし                ビット(1);
98      %改頁;
99      登録 未定義帳(コンテナファイル), 繰(コンテナファイル)
100     開始;
101     /* 最初にファイルを作る処理 */
102     開簿 帳簿(コンテナファイル) 出力順次鍵付き
103     環境(索引(REC(積荷票),
104           PRIME(積荷票, コンテナ番号)),
105           記録長(224));
106     書込 帳簿(コンテナファイル) 出力変数(制御記録);

```

```

107 閉簿 帳簿(コンテナファイル);
108 開簿 帳簿(在庫不足ファイル) 出力 順次 鍵付き
109 環境(索引(REC(在庫不足),
110 PRIME(在庫不足.番号)),
111 記録長(224));
112 書込 帳簿(在庫不足ファイル) 出力変数(在庫不足);
113 閉簿 帳簿(在庫不足ファイル);
114 次は 再試行;
115 以上;
116 再試行:
117 閉簿 帳簿(コンテナファイル) 更新 順次 鍵付き
118 環境(索引(REC(積荷票),
119 PRIME(積荷票.コンテナ番号)),
120 記録長(224));
121 読込 帳簿(コンテナファイル) 入力変数(制御記録)
122 鍵(制御記録.コンテナ番号);
123 閉簿 帳簿(在庫不足ファイル) 更新 順次 鍵付き
124 環境(索引(REC(在庫不足),
125 PRIME(在庫不足.番号)),
126 記録長(224));
127 登録 記録(コンテナファイル), 記録(在庫不足ファイル);
128 反転 未定義帳(コンテナファイル), 鍵(コンテナファイル);
129 登録 終簿(SYSIN)
130 処理 終了処理;
131 /* 主処理ループ。プログラムは、終了処理で停止する。 */
132 以下 反復条件('1'B);
133 処理 メニュー表示処理;
134 もし 指令 = '1'
135 ならば
136 処理 出庫処理;
137 なければ もし 指令 = '2'
138 ならば
139 処理 入庫処理;
140 なければ もし 指令 = '3'
141 ならば
142 処理 在庫不足リスト作成;
143 なければ もし 指令 = '4'
144 ならば
145 処理 終了処理;
146 なければ もし 指令 = '5'
147 ならば
148 処理 在庫品一覧表作成;
149 なければ
150 処理 入力誤り処理;
151 以上;
152 %改頁;

153 /*
154 *****
155 *                               メニュー表示処理                               *
156 *****
157 */
158
159 メニュー表示処理:
160 手続き;
161 /* メニューを表示して、処理番号の入力を待つ。 */
162 記帳 改行(4) 編集(MODE2, '==== 倉庫在庫管理 ====',
163 MODE0)
164 (A,N(19),A);
165 記帳 改行(3) 編集(MODE2, '処理番号を入力して下さい',MODE0)
166 (A,N(12),A);
167 記帳 改行(2) 編集(MODE2, ' 1. 出庫処理',MODE0)
168 (A,N(13),A);
169 記帳 改行(2) 編集(MODE2, ' 2. 入庫処理',MODE0)
170 (A,N(13),A);
171 記帳 改行(2) 編集(MODE2, ' 3. 在庫不足リスト作成',MODE0)
172 (A,N(18),A);
173 記帳 改行(2) 編集(MODE2, ' 4. 終了処理',MODE0)
174 (A,N(13),A);
175 記帳 改行(2) 編集(MODE2, ' 5. 在庫品一覧表作成',MODE0)
176 (A,N(17),A);
177 記帳 改行;
178 転記 編集(指令)(欄(1),A(1)) 復号;
179 以上; /* メニュー表示処理 */
180 %改頁;

```

```

181 /*
182 *****
183 * 出庫処理 *
184 *****
185 */
186
187 出庫処理:
188 手続き;
189 /* 出倉依頼データを入力し、在庫検査をする。
190 結果に応じて在庫なし連絡または出庫指示書の発行をする。
191 */
192 登録 変換
193 次は 再試行;
194
195 再試行:
196 記帳 改行(2) 編集(MODE2,' 出倉品名?',MODE0)
197 (A,N(10),A);
198 記帳 改行;
199 転記 編集(出倉依頼,品名)(欄(1),A(60)) 復写;
200 記帳 改行(2) 編集(MODE2,' 出倉数量?',MODE0)
201 (A,N(10),A);
202 記帳 改行;
203 転記 編集(出倉依頼,数量)(欄(1),F(5)) 復写;
204 記帳 改行(2) 編集(MODE2,' 送り先名?',MODE0)
205 (A,N(10),A);
206 記帳 改行;
207 転記 編集(出倉依頼,送り先名)(欄(1),A(60)) 復写;
208 記帳 改行(2) 編集(MODE2,'
MODE0)
(A,N(20),A);
209 転記 編集(確認)(欄(1),A(2)) 復写;
210

```

付録B アンケート (抜粋)

■ 質 問

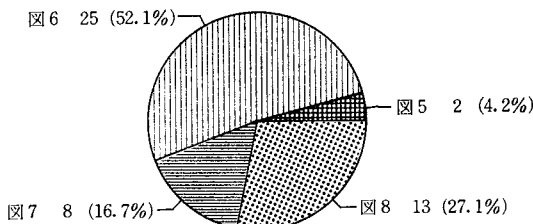
図5～8は、図4に示すような結果を得る PL/I プログラムを段階的に日本語化したものです。NPL1というプリプロセッサで PL/I 言語に変換して実行することができます。

さて、あなたはどのプログラムに好感をもちますか？

好感をもつ順に1から4まで順位をつけてください。

- 図5 日本語化しない元のプログラム
- 図6 リテラル、注釈、識別子(名前)を日本語化
- 図7 リテラル、注釈、識別子(名前)、キーワードを日本語化
- 図8 リテラル、注釈、識別子(名前)、キーワードを日本語化するとともに、文末や文中に補助となる動詞や助詞を加えたもの

■ 結 果



- 参考文献 [1] “日本語プログラミングでオンライン・データベース・システム開発”, 日経コンピュータ, 1983. 7. 25.
- [2] “パーソナル・コンピュータで稼動する日本語プログラミング言語の開発”, 日経コンピュータ, 1983. 1. 10.
- [3] 岡田卓也, 松浦成彰, 宮内和人, “日本語プログラミングへのアプローチ”, 情報処理学会第26回(昭和58年前期)全国大会, 予稿集, pp. 421~422, 1983.
- [4] 乙顔元, 池田丈男, 調重俊, “COBOLにおける日本語処理の導入方式”, 情報処理学会第26回(昭和58年前期)全国大会, 予稿集, pp. 417~418, 1983.
- [5] 竹田陽行, 横内一康, 堀田拓二”, PL/Iにおける日本語処理実現方式に関する一考察, 情報処理学会第26回(昭和58年前期)全国大会, 予稿集, pp. 419~420, 1983.
- [6] 柳沢正夫, 村井 純, 土井範久, 斉藤信男, 藤崎哲之助, 諸橋正幸, 仮名漢字を用いたプログラミング言語「JIO-Pascal」, 情報処理学会第25回(昭和57年後期)全国大会, 予稿集, pp. 357~358, 1982.
- [7] “UNIVAC シリーズ 1100 PL/I 解説書”, 日本ユニバック(株).
- [8] “UNIVAC シリーズ 1100 MACRO 解説書”, 日本ユニバック(株).
- [9] S.R. Greenwood, “MACRO: プログラム言語”, ユニバック技報, No. 1, 1981.
- [10] 上田謙一, 菅野 淳, 野田克彦, “日本語プログラミング言語の開発と問題点”, 日本語情報処理シンポジウム, pp. 308~316, 1978.
- [11] “プログラムの‘保守危機’を救う日本語プログラミング時代が接近”, 日経コンピュータ, 特別編集版秋季号, 1981.
- [12] “幕開く日本語プログラミング時代”, 日経コンピュータ, 日経マグローヒル, 1983. 4. 5.
- [13] 岡崎 健, “漢字 LEVEL II COBOL 日本語情報処理機能”, インフォメーション, pp. 87~92, 1983. 9.
- [14] 松浦 昇, “日本語プログラミング言語 PL/J の提案”, 日経コンピュータ, 1982. 8. 23.
- [15] 鈴木考則, “日本語プログラム言語「和漢」”, マイクロコンピュータ, Vol. 29, No. 2, 1983.

執筆者紹介 真田 正二 (Shoji Sanada)

昭和21年生, 44年早稲田大学理工学部数学科を卒業後, 同年日本ユニバック(株)入社. 日本ユニバック総合研究所で言語理論を研究後, Sperry社においてASCII FORTRAN, PL/IおよびPLUSのコンパイラの開発に参画. 帰国後, 各種言語プロセッサの提供と保守に従事. 現在, プロダクト本部ソフトウェア企画部で, プログラム言語の日本語処理機能の設計を担当.



報告 データ・セキュリティ

——障害に耐える閉じた環境としての 1100/90 シリーズ

Data Security:

The 1100/90 as a Closed, Fault-Tolerant Environment

S. J. Rawlins

要約 信頼できるコンピュータとは必ずしも障害(fault)がないものを指すのではなく、障害があってもそれに耐えるものである。障害はユーザのプログラムの論理エラーのようにありふれたものもあり、また主記憶域上にないページへのアクセスのように性質を異にするものもある。本稿では 1100/90 シリーズに組み込まれた保護機構の概略と、それがどのように障害に耐える閉じた環境を実現しているのかについて述べる。“閉じた環境”(closed environment)とは、そこでは明確に承認(authorise)されていないすべての行為(action)は障害となるような環境をいう。対照的に、そこでは明確に禁じられない限り、すべての行為は合法となるような環境を“開いた環境”(open environment)という。

本稿では、まずコンピュータのセキュリティについての多数の文献から、システム・プログラマに有意義な概念を抽出してみる。このため、1100/90シリーズのハードウェア、オペレーティング・システムおよびリンキング・システムをばらばらな製品としてではなく、一つの統合システムの層(layers)として扱い、セキュリティを組み込むことを主目的とする。ユーザが不注意にまたは故意にデータを破壊しないように、下層をなすアーキテクチャがユーザに代わってどのように働くかに力点をおく。

コンピュータのセキュリティ侵害についての使用者の関心が深くなるほど、セキュリティ・システム・モデルの明確で納得のいく分析が一層重大になる。本稿では現在において期待できる信頼性と安全性によって、この問題を明確にする。残念なことに、汎用のデータ処理に対するセキュリティ・システムを定義することは、このような汎用のシステムのモデルを作ることと同様に扱いにくい仕事である。以下、一般的な要求の集まりを提示して、セキュリティ・システムを実現する問題をどのように解決するかという立場で、文献にみられる一般的な保護モデルを調べる。そして 1100/90 シリーズのセキュリティ機構を、システムを実現する一つの統合されたアプローチとして評価する。最後に、予想される顧客の要求を、それにとまなう挑戦に照して結論を導く。

Abstract A reliable computer is not necessarily one free from faults, but one that is tolerant of them. A fault can be as familiar as a user's programming logic error or as foreign as a non-resident page access. This paper presents an overview of the protection mechanisms built into the 1100/90 and how they implement a fault-tolerant, closed environment. A *closed environment* is one in which all actions cause a fault unless explicitly authorized; this notion contrasts an *open environment* in which all actions are legal unless explicitly forbidden.

This paper attempts to distill from the plethora of documents on computer security which is meaningful to the systems programmer. To that end the 1100/90 machine, its operating system and the Linking System are treated not as separate products but as layers of one, integrated system built with security as its chief aim. Emphasis is placed on how the underlying architecture acts on the user's behalf to protect him from inadvertent or malicious data corruption.

The increasing sensitivity among customers to computer security violations makes clear, well-illustrated analysis of models for secure systems all the more critical. This paper specifies the pro-

blem in terms of current expectations of reliability and security. Unfortunately, defining a "secure system" for general-purpose data processing proves as slippery a task as forming a model for such a system. A general set of requirements is proposed. A popular protection model from the computing literature is examined based on how it addresses the problem of implementing a secure system. The 1100/90 security mechanism is evaluated as an integrated approach to implementing such a system. Conclusions are drawn based on predicted customer requirements and the challenges they entail.

1. はじめに

本稿では内部のセキュリティ機構のみを考える。一般に、このような機構はユーザに意識されることはないであろう。ここでは、“コンピュータ・セキュリティ”という言葉からすぐ連想される暗号のような問題は扱わない。コンピュータ・セキュリティの要求についての一般の考え方に照らすと、多分、なぜデータ・セキュリティに対する高い水準の要求がセキュリティを留意して設計したアーキテクチャによってのみ満たされるのかを、最初に説明した方がよいであろう。

一般に、コンピュータは抽象化の層からなっている。層の各レベルはその下のレベルから構成され、逆に、次の上位レベルを構成する基礎となる。各々の新しいレベル、つまり層はより高い抽象化レベルであって、下位レベルの詳細をかくしている。ところで、各々の新しいレベルはその基礎（のレベル）と同程度にしか良くない（“良い”とは安全性、信頼性、効率性、保守性、あるいはユーザの優先度によって決定されるそれらの組合せを意味する）のは明白である。コンピュータ科学の各部門は抽象化のレベルを扱っているし、ISOの7階層のデータ・ネットワーク・モデルは、現存するネットワークがどの程度独立の分離した層でデータ・フローを扱うかを評価するための規準を提供している。したがって、モジュール設計や構造化プログラミングの理論で武装した言語の設計者らは、新しい言語を設計する時に、データと手続きの階層の問題にふれるのは当然である。また、オペレーティング・システムの設計者らはすぐにモノリシックなモニタのアプローチを棄て、核(Kernel)と呼ぶプロセス・コントロール用基本命令の最小の集合からシステムを作る方法を見出した。

そこで、セキュリティの階層をコンピュータの不可欠な一部分として構成する考え方を最もよく受け入れるのは、オペレーティング・システムである。オペレーティング・システムの責任でなければ、何がコンピュータ・セキュリティといえるだろうか。オペレーティング・システムはプロセスを生成したり、除去したり、資源をこれらのプロセスに割り付け、その相互作用を制御し、そしてそれらを相互に保護するが、これらの各領域でセキュリティは重要な役割を果たす。したがって、セキュリティはオペレーティング・システムと呼ぶソフトウェアに複雑に結び付いている。セキュリティに対する関心はハードウェアやソフトウェアの境界にとどまらない。システム的设计者にとっては、セキュリティに対する関心はアーキテクチャのレベルまで広がる。システム・ソフトウェアの第1層の要求をサポートするために、ハードウェアが作られなければならないのはこの部分である。その核が合理的な信頼性をもってタスクを行うためには、明確に定義され、きびしくテストされた基本命令が核に供給されなければならない。

システムを抽象化のレベルに分割することは、万能薬でもそれ自体が目的でもない。各抽象化のレベルは、次の新しいレベルの生成を正当に行うに必要な十分な機能の集合を実行できなければならない。したがって、レベルを追加しても自動的に信頼性があがることに

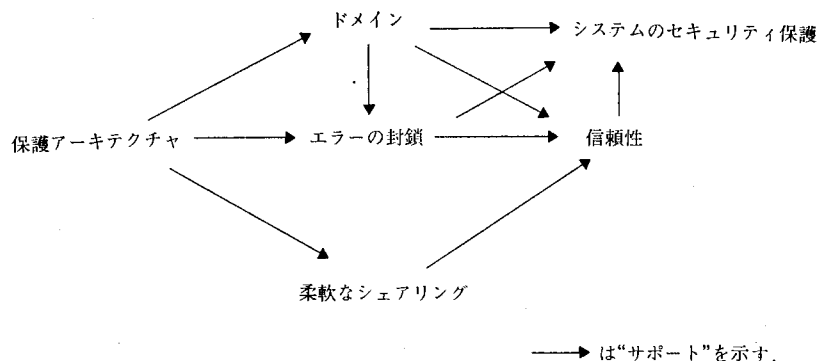


図 1 1100/90 シリーズのサポート・グラフ

Fig. 1 1100/90 series support graph

はならない。何かあるとすれば、それは層の間の明確な境界を維持する重荷が増すことくらいである。層間の境界が曖昧ならば、エラーは単に意味のあるレベルを越えたレベルまで伝播するだけでなく、重大さをも増大させる。したがって、貧弱な構造はエラーを増幅するだけである。

そこで、本稿では、セキュリティに対する階層化アプローチをハードウェア・レベルから始めなければならないという立場をとる。1100/90 シリーズをまさにそのアプローチの例として提示する。図1は、1100/90 シリーズについてどのように低レベルの基本命令がセキュリティと信頼性の究極の目標をサポートするかを示す。これはセキュリティの設計の青写真ではなく、本稿を読むためのガイドである。

次の章で、図1と本稿を通して使用する用語を説明する。コンピュータ・セキュリティに詳しい読者はこの節をとばしてもかまわない。

2. 定 義

多くの用語がコンピュータ・セキュリティについて用いられている。本章では、本稿で使用する最も一般的なセキュリティに関する用語を定義する。なお、用語の多くは Linden の文献^[5]から借用した。

アクセス権……承認されているアクセス・モードの一つに従って対象を使う権利

ドメイン……主体システムの対象に対してもっているアクセス権の集合を定義している領域

エラー・コンファインメント……主体がアクセスの大きなドメインをもたないか、あるいは、矛盾するデータに操作することが許可されていない環境

フォールト・トレラント……システム内の障害を受け入れるが、それをできるだけ正しい状態に保つ機構を採用すること

最小値特権の理論……主体がその義務を実行するのに必要なアクセス権のみを主体に提供すること

保護機構……承認されていない、あるいは望ましくないアクセスからデータを保護するために設計されたシステム機能

信頼性……ソフトウェアが信頼できるためには、正当である必要はない。最も起こりやすいエラーまたは障害がそのソフトウェアを使用不能にすることなく、またそれを使用不能にするエラー等がまれならば、その手続きは信頼できると考える。

安全性……安全対策によって管理される場合、合法的なアクセスを妨げることなくデータを故意、不適当に配布したり破壊するのを防ぐシステムの機能

3. セキュリティ・モデル

3.1 なぜモデルをつくるか

複雑なシステムは、その構成要素の集まりとそれらの相互作用に分解しなければ、すぐに扱いが困難になる。そして、モデリングはこのようなシステムの検査のための形式的な枠組を提供する。また、モデリングは、しばしばそれ自身が目的でもある。実際、厳密なシステム定義と問題記述が非常に明示的であれば、それ以上の分析の必要性はない。おそらく、モデルは、より以上の作業へ進む前に、修正を必要とする設計に内在する弱点を示してくれる。システムが不可能であることをモデルが証明するなら、設計を続行する必要がないことは明白である。

さらに、ユーザの要求や法規は通常規定的な書き方でなく、記述的な書き方になっている。そこで、要求が複雑であればあるほど、直接実現できる確率は低くなる。モデルは実世界の要求とその実施の間のギャップの橋渡しをする助けとなる。Landwehr は、彼の論文^[6]の中でこれについて簡潔に述べている。

ここで、問題となるのは、法規の書き方のまずさにあるのではない。なぜならば、実際に多くの関連する問題がまだ研究を要する段階において、法規によって特別の方向を特定するのは好ましくないからである。問題はむしろ、設計のためにセキュリティの形式的モデルが必要であるということである。システムは単に安全であるばかりでなく、安全であることが証明されなければならない。したがって、システムのセキュリティをほかの人々に確信させることができるように、設計者はセキュリティの形式的モデルを必要とするのである。セキュリティのある形式的モデルを作り、このモデルを実行するシステムが（米国防省 DOD の法規なり、プライバシー法なり、私企業の方策なりに照らして）安全であることを証明し、そして設計に従った実現がこのモデルを実行することを証明することによって、設計者はシステムが安全であることを論証することができる。

3.2 アクセス行列モデル

コンピュータ・セキュリティのための各種のモデルのうち、精査に耐え多数のプロトタイプ機械で使われたモデルにアクセス行列モデルがある。このモデルは、各種の安全政策を実現する基礎となりうる保護機構をモデル化しており、ホストのオペレーティング・システムに埋め込まれる基本指令をもったセキュリティの核を定める。また、このモデルは効率よく実現でき、さらに高位のセキュリティ・レベルによっても利用できる。このモデルはハードウェア、またはファームウェアで実現される不変の層をなす。また、すぐわかるように、このモデルでは、安全対策の要求に応じ洗練の度を変えられるし、柔軟に指定が変えられる。

アクセス行列モデルは三つの部分で構成されている。第1は**対象** (Object) の集合である。対象とは保護されなければならない（すなわち、それに対してアクセスが管理されなければならない）システムの**実体** (Entity) である。対象の例としてはファイル、セグメント（またはバンク）、端末装置や処理があげられる。第2は**主体** (Subject) の集合で、主体とは対象にアクセスする実体である。主体も互いに保護されなければならないから、主体もまた対象である。第3は、主体が対象にアクセスする方法を規定する**アクセス規則** (Access Rule) の集合である。

これら三つの成分が保護システムを定義する。保護システムが有用であり、信頼性と閉包性の要求に合致するためには、主体間に対象を組織的かつ効率よく共有 (Share) させることができなければならない。主体は互いのやりとりのためと乏しい資源を効率よく配置するために、対象を共有できなければならない。また異なる主体が、異なったモードで、同一の対象に同時にアクセスできなければならない。このような共有は閉じた環境の封鎖した境界内だけで起こるべきである。これらの要求とアクセス規則を組み合わせる接合部がアクセス行列 (Access Matrix) である。

アクセス行列は、主体の対象へのアクセスを管理するシステム・ワイドなデータ構造であり、対象にある種の操作を行おうとする主体がアクセスが明示的に許可されている時に限ってアクセスできるように管理する。このような許可条件にストアされる行列は 2 次元構造であり、その行は主体を、列は対象を表現し、任意の主体は任意の対象にアクセスを試みることができるようにする。すべてのアクセスは対象のコントローラ (controller) を通る。各タイプの対象に対応して、リクエストされたアクセスがその主体に許可されているか否かを決定するアクセス行列を読む、コントローラがある。アクセス権には、普通の Read, Write, Execute の外に二つの特別な権利 **Owner** (所有者) と **Copy** (複製) がある。対象の Owner はその対象のすべてのアクセス権の生成と削除を管理する。任意の対象はただ一つの Owner をもつ。主体 S が対象 O に対してオーナー特権をもっている時、S は S 自身が望むいかなる権利をも O の列に加えることができる (すなわち S は O に対する権利を、S 自身と他の主体に対して許可できる)。Copy はアクセス権に適用される属性で、同じ列の任意の行に (すなわち、他の任意の主体に対して) 指定したアクセス権をコピーする権利を主体に許可する。このアクセス権はコピー・フラッグとともに、あるいはフラッグなしでコピーできる。

図 2 に示したアクセス行列 (AM) では、主体 S_1 は自身と他の主体 S_2 (主体自身が対象である) を所有している。 S_2 はバンク B_1 とファイル F_2 を所有しており、したがって、たとえば、 S_2 は $AM(S_1, B_1)$ にアクセス権 R (Read) を生成できる。そこで、 S_1 がバンク B_1 を読むことができるようになる。また、希望する時はいつでもこの権利を除去できる。 S_1 はファイル F_1 への Read アクセスを所有しているが、 F_1 を所有していない (F_1 の Owner はアクセス行列のこの部分では示されていない)。 $AM[S_1, F_1]$ の "R" にあるコピー・フラッグ (*) は、 S_1 が F_1 の列 (たとえば、 $AM[S_2, F_1]$) にある他のいかなる対象に対してもこの Read の許可をコピーできることを示す。しかしながら、一度コピー

		主 体		バ ン ク		フ ァ イ ル	
		S_1	S_2	B_1	B_2	F_1	F_2
主 体	S_1	Owner	Owner			R*	E*
	S_2			Owner R, W	E		Owner W, E

*はコピー・フラッグを示す。

図 2 アクセス行列 AM [主体, 対象]

Fig. 2 Access matrix (AM [Subject, Object])

一されると、この許可を削除できない。

コンピュータ内の主体と対象の総数は膨大であり、それらの相互作用がまれであることを考えると、アクセス行列を2次元の配列として実現することは実用的でない。そこで、二つの別の表現法が開発されたが、それぞれ長短がある。

一つの方法は、対象に、それを保護するアクセス権全体を結び付ける方法である。つまり、対象はアクセス行列のそれ自身の列を保持する方法で、通常、アクセス権をリンクしたリストで表現する。各要素は (subj, ar) ——主体とアクセス権の対——という形である。この表現はアクセス・リスト (Access List) と呼ばれる。もう一つの表現法は列ごとに権利をまとめる方法である。各主体は (obj, ar) ——対象とアクセス権の対——という形の要素をリストの形でもつ。これをケーパビリティ・リスト (Capability List. これは主体の全機能を定義している) と呼ぶ。

すべてのアクセスの試みはコントローラによって監視されるので、いずれのアプローチも効率よく実現されなければならない。主体 S が対象 O_1, O_2, \dots, O_n への行動 A_1, A_2, \dots, A_n を順次実行するとしよう。(各 O_i は列の中では唯一であると仮定する。) この場合、アクセス・リスト表現では、 S の代わりに実行するプロセッサはアクセス・リストを、アクセスされる n 個の新しい各対象に対して、アクセス・リストを n 回交換する必要があるから、アクセス・リスト表現はケーパビリティ・リストより n 倍効率が悪い。実際、どのアクセス・リスト (すなわち、どの対象) が次にアクセスされなければならないかを一般に知る方法がないので、効率の向上はたかだかアクセス・リストの保有法を手際よく行う改良にとどまる。対照的に、ケーパビリティ・リストを用いて S を実行するプロセッサは、定義によって唯一の主体 S のプロセス状態をもつので、高速メモリに S のケーパビリティ・リストを保持すれば、オーバヘッドは減らせる。もう一つの方法として、ケーパビリティの“working set”は一つの場所——連想記憶装置が理想的である——に容易にストアできる。したがって、二つの表現法のうちケーパビリティ・リストの方がより効率的である。

3.3 要 約

その単純さと汎用性のため、アクセス行列モデルは大きな柔軟性と広い適用性をもっている。この節では、次に論ずる 1100/90 シリーズの保護機構の設計基盤を紹介した。アクセス行列モデルのより徹底した議論は Denning の文献^[2]にみられる。ケーパビリティにもとづくアーキテクチャの徹底的な分析も彼の文献^[3]にみられる。

4. 1100/90 シリーズ

1100/90 シリーズのアーキテクチャに入る前に、極端に影響をもつ顧客が自身のデータベースのセキュリティを改善する戦略と、それが Sperry 社のような売りに与える影響に注目することは重要である。その顧客とは Ada の開発における役割で有名な DOD である。その戦略は“Trusted Computer System Evaluation Criteria”^[4]というタイトルのドキュメントにみることができる。この DOD のドキュメントの存在はユーザ社会におけるセキュリティへの関心と理解の程度を示すものである。このドキュメントは抽象的なレベルで、コンピュータ・システムに用意されるセキュリティ保護を判定する基準を定義している。この“抽象”は曖昧さを意味するのではない。その基準は説得力があり、売りが使う特定のセキュリティ・モデルにあらかじめ規定を加えないように、セキュリティ政策のレベルで適用される。DOD では、機密レベル A から D を定義している。レート “D”

のシステムは最小の保護を用意するもので、“A”システムではセキュリティが証明されなければならない。この DOD のドキュメントの存在はユーザ社会におけるセキュリティへの関心と理解の程度を示すものである。

4.1 保護機構

1100/90 シリーズの保護機構は三つの方法で具体化される。すなわち、アドレス付け(番地付け)、アクセス制御およびプロシージャ制御である。これらは、より上位のレベルのセキュリティ構造、とくに主としてドメインをサポートする(図1参照)。ドメインは、本節の終りで論ずるソフトウェア・サブシステムを実現するのに使う。(1100/90 シリーズの保護機構については Cooper と McKenzie の各文献^{[1],[7]}を参照のこと。)

4.1.1 アドレス付け

1100/90 シリーズのアドレス付けの図式は柔軟性のある共有とエラー封鎖およびある限定されたスコープをサポートする。アドレス付けのすべてはアドレス木を通して実行される。アドレス木は対象の共有を制御するために使われるシステム・ワイドなデータ構造である。現時点で考えられている唯一の対象は UNIVAC シリーズ 1100 の用語でバンクである。(以降、本節ではバンクと対象は同義語である)。システム内の任意のバンクはアドレス木のどこかに割り当てられていなければならない。このバンクが割り当てられた場所がそのスコープ、いかえれば、共有のレベルを決定する。1100/90 シリーズは四つの共有レベルをサポートする。したがって、木構造は4レベルの深さで各ノードはバンク記述子のグループを含んでいる。

図3で、各円は **Bank Descriptor Table** (BDT) つまりバンク記述子の配列を表現しているレベル0(もっとも共有されるレベル)には、ただ一つの BDT のみが許される。これはシステム全体にまたがって共有されるすべてのバンク(EXECバンクとその他のバンクを含む)の記述子を含んでいる。レベル3(最も共有されていないレベル)には、各主体 S に対して一つずつ BDT がある。レベル3の BDT のバンクは、それに対する主体(ここで、主体は UNIVAC シリーズ 1100 の用語ではアクティビティを意味し、一般的にはプロセスである)によってのみ参照できる。レベル1と2のバンクは同一のアプリケーションあるいはランによってそれぞれ共有される。これらのレベルによって共有をより精密に制御することができる。対象と同様、主体はスコープ(Scope)をもっている。主体のスコープは、アドレス木の根から葉へ至る道の上に現れるバンクの集合である。したがって、主体は、四つの BDT に記述されているバンクをアドレス付けすることができる。

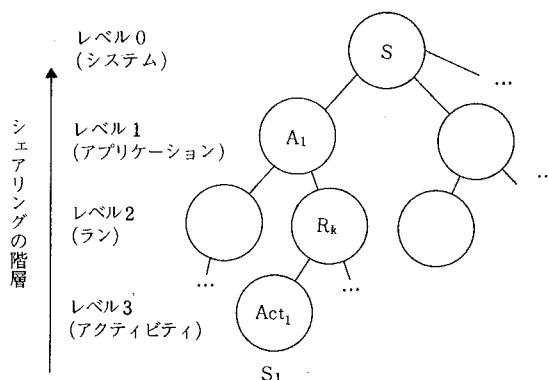


図3 アドレス木

Fig. 3 Address tree

図3の S_i は $BDT-Act_i$, $BDT-R_k$, $BDT-A_j$ と、最後に、すべての主体と同様に、そのスコープ内に $BDT-S$ をもっている。主体は木を横切って横道のみをみることはできず、ただ根の方向へ“上”だけをみることができる。あるバンクに主体がアクセスする必要がある時は、ベース・レジスタにバンク記述子をロードしなければならない。ベース・レジスタをロードする命令は BDT を選ぶレベル番号を使い、そのテーブル内でバンク記述子 (BD) を選ぶにはバンク記述子インデックス (BDI) を使う。そして16のユーザ・ベース・レジスタ ($B_0 \sim B_{15}$) または $EXEC$ ベース・レジスタ ($B_{16} \sim B_{31}$) の一つにその BD の内容をロードする。このロード命令を LBR とすると、命令は次の形をしている。

$LBR \ B_i, l, bdi$

つまり、(S のスコープの中の) レベル“ l ”にある BDT の要素“ bdi ”で見出されるバンク記述子を B -レジスタ“ i ”にロードする。この時、目的のバンクはいま B_i にベース (**base**) されたという。バンクにアクセスするために使う各命令 S ではレジスタ B_i を直接指定する必要がある。すなわち、

$op \ A, off, B_i$

はレジスタ A と B_i にベースしたバンク内のオフセット“ off ”にあるオペランドでオペレーション op を行う。要約すると、各バンクは四つのスコープ・レベルの一つに現れなければならない。主体はそのスコープ内でのみバンクをみることができる。そして、主体がバンクにアクセスしようとする時は、そのスコープ・レベルとスコープ・レベル内のバンク記述子のバンク・インデックスを指定し、 B -レジスタにそのバンクをベースし、そして、その B -レジスタを通してバンク内のオペランドにアクセスしなければならない。

アドレス木は共有の階層を作るから、そのバンクにアクセスする必要がある主体たちだけが実際にアクセスできるように、バンクをアドレス木に置かなければならない。このような基本的保護がバンクの完全性を保持する、すなわち、これは意図的なあるいは偶発的なデータの破壊を防ぐとともに、最小値特権の原理と柔軟な共有を通して、セキュリティをサポートする。

従来の UNIVAC シリーズ 1100 を越える 1100/90 シリーズの大きな改良は、セキュリティにある。1100/90 シリーズでは、オペランドのバンクに相対的なオフセットは常に指定された (強制的な) ベース・レジスタによってベースされたバンクに相対的なものと解釈される。このことは、オフセットを含んでいる最初のレジスタを (四つの) ベース・レジスタから探すという従来の方法と大きく異なる。バンク選択のこの暗黙の方法では、プログラムにやっかいなソフトウェア・アドレッシングの約束とアドレス範囲のチェックを促す。また、このようなチェックにもかかわらず、アドレス範囲が目標バンクとオーバーラップしたバンクに意図せずスタアすると、プログラムはしばしば異常終了した。このような暗黙の探索順序を除外することによって、1100/90 シリーズではセキュリティ、完全性およびエラーの封鎖を向上させている。

4.1.2 バンクのライフタイム

バンクは、動的に生成され破壊される。監視プログラムにはアドレス木上の与えられたレベルでバンクを生成したり、木からバンクを取り除いて、その内容を破壊し、 BDT の中の空いたエントリを新しいバンクが使えるようにするためのルーチンの集合がある。リンク・システムはこれらのルーチンを使って、対象モジュール内で定義されたバンク・テンプレートからバンクを生成する。リンク・システムの管理のもとでは、アドレス空間は、与えられたアクティビティ、実行、アプリケーションあるいはシステムの

必要性に従って変動する。こうしてバンクの最小のワーキング・セットが維持される。これは（システム生成時に定義されるコモン・バンクのように）バンクのライフタイムが全システムと同じ長さか、あるいはプログラム・バンクと同じ長さであった従来のシリーズ 1100 のバンクのライフタイムの定義と対照をなす。ここでのキーポイントは、1100/90 シリーズでは、バンクのライフタイムと共有レベルが分離していることである。もちろんバンクは、自分の記述子が置かれている BDT より長くは存在できない。アクティビティ・レベルのバンクはそのアクティビティ（主体）のライフタイムより長く存在できないが、アクティビティのライフタイム内の任意の時点で生成させたり破壊させたりする。ラン、アプリケーションおよびシステム・レベルについても同様である。

4.1.3 アクセス制御

1100/90 シリーズの保護機構の 2 番目の構成要素はアクセス制御である。アドレス付けやプロシージャ制御に比べて、アクセス制御は UNIVAC シリーズ 1100 あるいは他のいかなる第 3 世代コンピュータにも前例がないので、ある意味では最も複雑な要素である。しかし考え方は確立されており、設計の基本作業は 15 年以上行われてきている^[2,3,8]。

3 章で述べた一般のアクセス行列モデルを思い出そう。その見方では主体と対象とを素元としてアクセスが扱われている。アクセス行列では主体は行に、各対象は列に含まれている。各対象は、対象自身と他の主体に関する各種アクセス許可をする Owner（主体）をもっている。

1100/90 シリーズではオーバ・ヘッドを最小にするため、粗いレベルでアクセスを管理する。各対象（バンク）は一つの Owner をもつ。しかしながら Owner は主体ではなく、システムまたはドメイン（domain）である。ドメインとは多くの主体が動作できかつ一つの共通リンク環境を共有するルーチンとデータの集合である。アクセス制御機構では、オーナー・ドメインの中で動作する主体とオーナー・ドメインの外で動作する主体との間でだけ区別をする。各対象は二つのアクセス許可条件をもっている。一つはオーナー環境のため、もう一つは非オーナー環境のためである。アクセス管理はこのように 2 レベルであり、アクセス行列は現在二つの行だけからなる。

許可値は R (Read), W (Write), RW (Read と Write) と E (Enter)——または E (Execute)——である。オーナー・ドメインのアクセス許可は SAP (Special Access Permissions) と呼ぶ。また非オーナー・ドメインのアクセス許可は GAP (General Access Permissions) と呼ぶ。SAP の “Special” は、対象に対するより大きな能力を意味するものではない。図 4 をみると SAP は、GAP より能力が小さい。B₂ のドメインで動作する主体は B₂ を読むだけであるが、ドメイン外の主体はそれを読むことも書くこともできる。しかしながら、この 2 レベルの分割は絶対的なものではない。主体が他のドメインにあるバンクに対してのオーナー特権 (SAP) を得るには二つの方法がある。一つは要求するバンクのドメインにドメインをスイッチする方法であり、もう一方はセキュリティ・リングを通す方法である。これら二つの方法について述べる。

	B ₁	B ₂	...
GAP	—	RW	
SAP	E	R	

図 4 2 レベルのアクセス行列

Fig. 4 2-Level access matrix

```

Virtual_Processor      {1100/90}
  [Key: [Ring: 0..n,   {n=3, bits}
    Domain: 0..m]     {m=127, 7 bits}
  PP: 0..p,           {p=3, 2 bits}
  BDT_Ptrs: array [0..l] of ^BDT {l=3, 4 pointers}
];
S1, S2: Virtual_Processor

```

図 5 擬似コードによる 1100/90 シリーズ仮想プロセッサの定義

Fig. 5 The definition of an 1100/90 virtual processor by pseudo-code

主体とドメインの区別は重要である。ドメインとは明確に境界の定ったソフトウェア・コンポーネントである。例としては、DMSやEXEC、コンパイラまたはデバッガ等がある。主体とはプロセス、すなわち、プログラムの命令（静的なプログラム自身ではなく）に従って実行される行動の列である。各プロセスには、ある任意の時点でのプロセスの状態を定義する仮想プロセッサが伴う。この仮想プロセッサはそのタスクを実行するためには、実際のハードウェア・プロセッサをもっていなければならない。通常、仮想プロセッサはある種の多重化を通じて少数の実プロセッサを共有しなければならない。

1100/90 シリーズ上では、仮想プロセッサはその主体のアクセス権、プロセッサ特権、およびアドレス・スコープについての情報をもっている。不完全な表現だが、図5の擬似コードで 1100/90 シリーズ仮想プロセッサを定義する。

ここで、欠けているのは命令ポインタとかレジスタ・セットのようなプロセス状態の他の成分である。(Key, PP と BDT_Ptrs) に示される三つのフィールドについては、それらが 1100/90 シリーズの保護機構と関係する面だけを述べる。

Key は主体のアクセス権を決定する。主体は対象に対するアクセス権を決定するため、対象のロックにそのキーを試みる。キーがそのロックに合致すると、主体は特別のアクセス許可 (SAP) を得る。一方、キーが合致しない場合、それは一般アクセス許可 (GAP) を得る。キー（とロック）に対して二つの成分——リングとドメイン——があることに注目しよう。今は、ただ、ドメインの値が合致する時にキーはロックに合致することのみを述べることにとどめる。リングについては次項で述べる。

“PP” (プロセッサ特権) は主体が使用できる命令のセットを決定する。1100/90 シリーズでは四つの PP レベルがある。すなわち、0——EXEC 核、1——EXEC ワーク、2——特権ユーザ、そして 3——通常のユーザ（優先順位の減少順）である。各レベルの命令セットは優先順位で直前のレベルの命令セットの部分集合である。PP=0 ではすべての命令が使用可能である。これに対して（ほとんどの第3世代コンピュータと同様）、従来の UNIVAC シリーズ1100 では二つの特権モード（スーパーバイザとユーザ）だけを認めている。PP についてはこれ以上論じない。これは 1100/90 シリーズ上で利用できる管理程度がより詳細であることの別の表明であり、最小値特権の理論を通じて閉じた環境をサポートしていることをいえば十分である。

3番目のフィールド、BDT_Ptrs は主体のアドレス・スコープを定義する。BDT_Ptrs^[1] はアドレス付けレベル“1”に対するバンク記述子テーブルを参照する。4レベルのアドレス木を思い出してほしい。すべての主体は同一レベル0（システム）のBDTを共有し、唯一のレベル3（アクティビティ）BDTをもつ。四つのBDT_Ptrsは、システムの視点（view）に対して、システムのアドレス付けが可能なバンクについての主体の視点を定義する（図

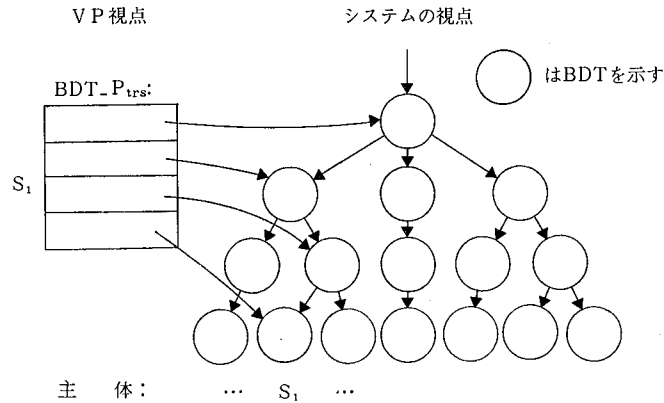


図 6 BDT_Ptrs
Fig. 6 BDT_Ptrs

```

Bank:
  [Lock: [Ring: 0..n,
          Domain: 0..m],
    Address_Level: 0..1,
    GAP, SAP: [R, W, RW, E]
  ];
B1, B2, ..., B9: Bank
    
```

図 7 擬似コードによるバンクの定義
Fig. 7 The definition of Bank

6).

バンクはその上で動作する主体と同様、アクセス制御とアドレス・スコープの情報も持っている。図 7 の擬似定義はこれらのフィールドを記述している。

“Lock” フィールドは主体の “Key” に対応する。そのキーが合致すると、主体は SAP にストアされたアクセス権を自由にできる。合致しない場合は、GAP にストアされたアクセス権が利用できる。“Address-Level” は、どのアドレス木にバンクを配置するかを EXEC に教える。

4.1.4 リング (Ring)

システム中のソフトウェアはドメインと呼ばれる境界の定まった一連の保護され、管理されたエントリをもつコンポーネントに細分された。図 8 はいくつかの典型的なドメインを示し、それぞれのドメインはシステム資源のパイの一部を占める。

主体は一時点で一つのドメインで動作できるだけである。主体は一つの例外を除いて、そのドメインの封鎖域内だけで対象に動作できる。主体がその望む対象よりも大きい特権をもったリング内で動作しているならば、その主体の現在のドメインに関係なく、同じドメインにあるかのようにそれにアクセスできる。この方法でリングはシステムの細分化に別のディメンションをもたらしている (図 9)。

リングもまた、システムと独立した集合に分割する。すべての主体と対象はある一時点ではただ一つのリングに入っている。主体は次節で述べる管理された方法でリングにスイッチできるが、対象は要求された管理度に適したリングで中にその Owner によって配置される。Owner はその対象のリングを変更できる。リング番号が小さいほど、特権は大きい。注目すべき重要な点はリングがドメインの壁を越えるということである。

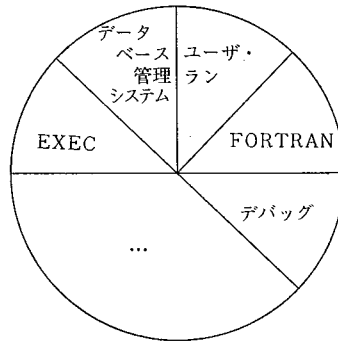


図 8 ドメイン

Fig. 8 Domains

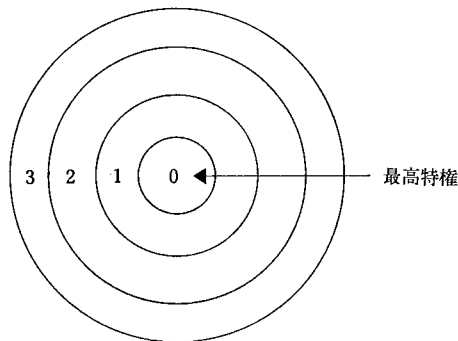


図 9 セキュリティ・リング

Fig. 9 Security rings

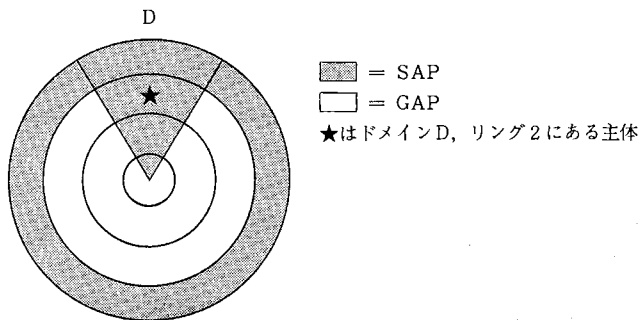


図 10 アクセス・リング

Fig. 10 Access rings

図 10 では星印で示されている主体は、SAP を使ってリング 3 の任意の対象にアクセスできる。1100/90 シリーズには、四つのセキュリティ・リングがあり、(アクセス・キー/ロックで 2 ビットを使う)。これら四つのリングは、4 レベルのアドレス木や 4 レベルのプロセッサ特権と混乱してはならない。

リングとドメインはアクセス制御、セキュリティ、および信頼性をサポートする。アドレス木は、柔軟な共有と信頼性をサポートする (図 1 参照)。

次の例は、保護アーキテクチャのこれら二つの部分がどのように協同して働き、閉じた環境に影響を与えるかを特徴づける助けとなる。リング 1 にある(アクセス・キー=(1, 3))、ドメイン 3 の中で現在動いている主体 S_1 を考える。これまでの実行の脈絡では、五つ

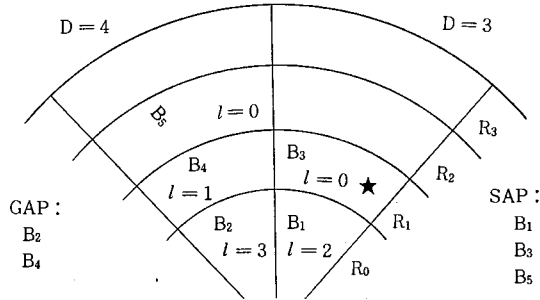


図 11 S₁ の視点
Fig. 11 S₁'s view

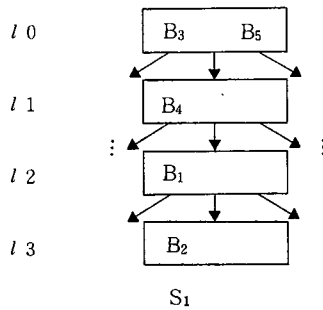


図 12 S₁ のアドレス・スコープ
Fig. 12 S₁'s adress scope

のバンク，すなわち，ドメイン 3 内の二つと外（つまりドメイン 4）の三つのバンクにアクセスしてきたと仮定しよう。この時 S₁ のアクセス権の視点は図 11 のようになる。

図 11 で星印は S₁ の現在のアクセス・キーを表す。B₂ と B₄ はそれぞれの GAP 値に応じてアクセスでき，B₁，B₃ と B₅ は，それらの SAP 値に応じてアクセスされる。“l” 値は各バンクにそのアドレス付けレベルを与える。アドレス・スコープに関するシステムの S₁ の視点は図 12 のとおりである。

図 12 の未定義な部分は，システムのアドレス木の残りを表している。しかし S₁ はこの木の S₁ の枝にあるバンクだけを見ることができていることを想起しよう。さらに，リングとアドレス付けレベルの間には何の相関もないことに注意しよう。B₁ はリング 0（最高特権）に現れるが，アドレス・レベルは 2（ラン・レベル）である。

S₁ のアドレス・スコープは動的である。いつでもその BDT にバンクを追加したり削除したりできる。バンクはその Owner のリンケージ環境の制御の下で木の上に現れる。すなわち，主体が一度も呼び出されることがない外部定義の手続きを呼び出そうとしてリンク・フォールトになった時に，リンクング・システムはその手続きを定義している目標バンクを見付け，そのバンクとそれに関係するバンクをそれらバンクに指定されたアドレス付けレベルに従って，主体のアドレス木の分枝上に配置する。

さて 2 番目の主体 S₂ を考えよう。S₂ は現在 S₁ と同じリングとドメインで動いている。しかしながらその実行の脈絡は異なっている。これは直接または間接に四つのバンク，B₆，B₇，B₈ および B₉ にアクセスしてきた。システムの S₂ の視点は次のとおりである。

許可されている共有の程度を図示するため，まず S₁ と S₂ が異なったアプリケーション

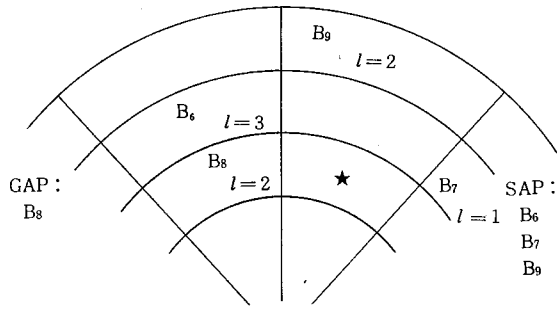


図 13 \$S_2\$ の視点
Fig. 13 \$S_2\$'s view

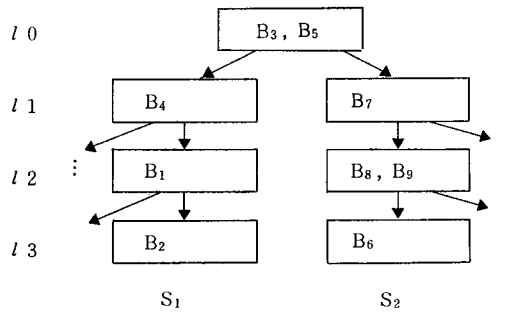


図 14 \$S_1\$ と \$S_2\$ のアドレス・スコープ
Fig. 14 \$S_1\$ and \$S_2\$'s address scope

ン (レベル1 グループ) で動作していると仮定しよう (図 13). アプリケーション・レベルまたはそれ以上のレベル番号では \$S_1\$ と \$S_2\$ 間に共有は起こりえないから, システム BDT だけが共有される (図 14).

もし \$S_2\$ が \$B_3\$ (BDT にすでに配置されているシステム・レベルのバンク) 中のルーチン呼び出してリンク・フォールトを起こしたならば, バンクが BDT 中に生成され配置される必要のないことをリンクング・システムは認識する. これは, そのシステム・レベルにある解決済みの外部定義名のテーブルによってリクエストされた手続き名がすでに (\$S_1\$ の前の呼出しによって) 生成されたバンク内で定義されており, そのアドレス木の根にあることがわかるから, 先のことが不要であることをリンクング・システムは知っている. \$S_2\$ は \$B_3\$ にアクセスするため同じ BDT を使う. リンキング・システムは, \$S_1\$ と \$S_2\$ が \$B_3\$ を同時に使うのに必要なリンケージの局所的な段階 (instance) を作るだけでよい. しかし, \$S_2\$ が \$B_4\$ (レベル1) 中のルーチン呼び出すならば, リンキング・システムは \$S_2\$ のレベル1 BDT (\$B_7\$ だけを含むと示されている) 中に \$B_4\$ に対する新しい入口 BDT エントリを生成することになる. このように, リンキング・システムの解決済み外部定義名のテーブルは, 柔軟な共有が行えるようにアドレス付けアーキテクチャを利用する. 最後に同一のアプリケーション・グループの中で動作している \$S_1\$ と \$S_2\$ を考えよう. アドレス木の中のそれらの部分は図 15 のようになっている.

アドレス・スコープの共有はアクセス権の共有を意味しないことに注意しよう. もし \$S_2\$ がドメイン4 (\$B_4\$ のドメイン) で動いているなら, それは \$B_4\$ でオーナー特権 (SAP) をもつ. 一方 \$S_1\$ は \$S_2\$ と \$B_4\$ を記述する BDT エントリを共有しているにもかかわらず,

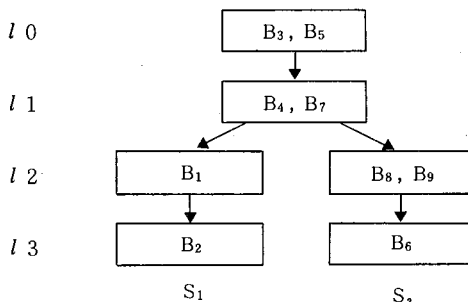


図 15 同じアプリケーションでの S₁ と S₂ の例

Fig. 15 S₁+S₂ in same application

- リターン、アドレス、PP、アクセス・キーをセーブし、リストする。

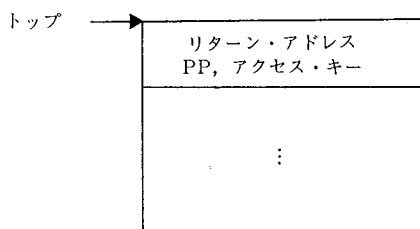


図 16 RCS

Fig. 16 Return control stack

S₁ は共通特権 (GAP) をもつ。ハードウェアはこのアクセス制御を各オペランド参照時に実行するから、S₁ あるいは S₂ が B₄ にアクセスするたびに主体のアクセス・キーはバンクのロック (B₄ の BDT エントリにストアされている) と比較され、アクセス権を決定する (Enter)。アクセスは各命令取出し時ではなく、目標コードのバンクがベースされた時だけ 4 に検証される。一つのベース・レジスタ (B₀) にのみコード・バンクをベースすることができるようにすれば、ベースされた時に一度検証されれば、各命令取出しの妥当性が保証される。これによって閉じたドメインを管理するための柔軟で、保護された環境が提供される。

4.1.5 プロシージャ制御

1100/90 シリーズの保護機構の第 3 の要素はプロシージャ制御である。1100/90 シリーズは、三つのプロシージャ制御機構を通して主にドメインのスイッチングをサポートする。それぞれはプロシージャのモジュール性、閉包性、およびエラーの封鎖をサポートする。三つの機構とはスタック、ゲートおよび共通の呼出し列である。

1) スタック

1100/90 シリーズには、一般的なスタック操作機構がありユーザは任意のバンクをスタックとして使うことができる。バンクの上限と下限は、それぞれスタックの底と上限になる。ユーザ命令語の PUSH あるいは POP によって、このスタック・バンク上に指定したサイズのスタック・フレームをそれぞれ生成あるいは破壊できる。スタックの上位あふれと下位あふれはハードウェアで検知し、あふれを起こした主体に割込みを起こす。

この一般的機構だけで信頼性と耐故障性の原則をサポートする。しかし、本章では

Return Control Stack (RCS), つまり, CALL と RTN (return) 命令によって使用する一般スタック機構の特別な場合だけを述べる。

EXEC は, RCS バンクを定義するために, 基本レジスタ B_{21} を予約している。スタックの各エントリは 2 語の長さで, 最初の語には呼出しルーチンのリターン・アドレスを, 2 番目の語には, 呼び出す側の PP (プロセッサ特権) とアクセス・キーを入れる (図 16)。

CALL で, ハードウェアは新しいスタック・フレーム (すなわち, 2 語のエントリ) を割り当て, 最初の語に主体の現在の命令のアドレス +1 を, 2 語目に (PP, アクセス・キー) の対をストアする。PP とアクセス・キーはドメインまたはリングをスイッチした場合に, 後でこれらの値をリストアできるようにするため保存しなければならない。RTN では, ハードウェアはトップ・フレームを解放し, PP とアクセス・キーを主体にリストアする。

ルーチンを直接呼び出す時は, セーブした PP とアクセス・キーは不必要である, 同じ値がリターンの際にリストアされる。主体は GAP が E (Enter) である限り, ドメインをスイッチすることなく, 他のドメインの中のルーチンを直接呼び出せる。しかし, 主体が, ゲート (gate) と呼ぶ間接的機構を通してルーチンを呼び出す時は新しいキーと PP 値を得るから, ちょうどその実行の始点へ戻る必要があるように, もとのアクセス・ドメインに戻る必要がある。

2) ゲート

システムがドメイン間の呼出し (コール) を許し, しかも閉じかつ故障に耐えるためには, 呼出し時点で保護を発動しなければならない。これは, 早急なエラー検知と制御された回復を確実にする唯一の方法である。1100/90 シリーズはゲートと呼ぶデータ構造を通してドメイン・スイッチの保護を強制実施する (図 17)。これは, 壁で囲まれた“都市” (ドメイン) へのアクセスを許可するかぎられた数の門に似ている。ゲートは (通常, 別のドメインにある目標ルーチンへの制御された入口を表す。ゲート自身はゲート・バンクに保存された実体である。ドメインは自分のゲート・バンクには直接アクセスできない。すなわち, EXEC を通してだけ, 新しいゲートやゲート・バンクを生成することができる。主体が, ゲートに保護されたルーチンを初めて呼び出そうとする時に, リンキング・システムは (EXEC を通して) その入口のためのゲートを作る。呼出しを試みるためには, 主体はゲート・バンクとゲート自身 (ゲートはバンクと同様に SAP と GAP 値をもっている) への Enter 許可をもっていなければならない (図 17)。

保護機構は呼び出す側から呼び出される側への推移をコントロールするが, この時, その両者が互いを疑いあうという雰囲気の下で行動させる。すなわち, 呼び出す側は呼び出

- ドメイン・スイッチの制御
- 自分自身のアクセス・ロック

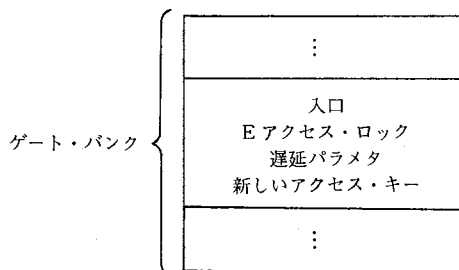


図 17 ゲート

Fig. 17 Gate

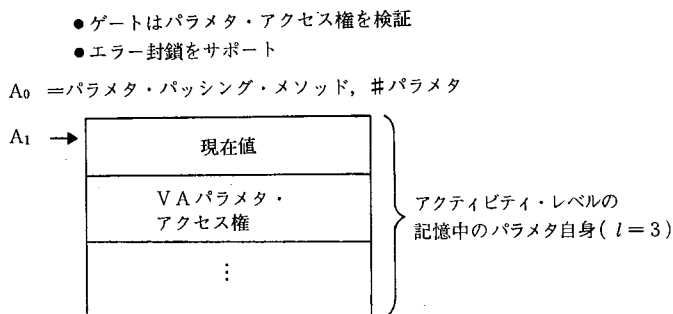


図 18 共通の呼出し列

Fig. 18 Common calling sequence

さるれ側を信用しないし、その逆もまた同じである。不信任から保護機構が共通の呼出し列に従ってパラメタを検証するのである。

3) 共通の呼出し列

この約束では二つのパラメタ・リスト・フォーマットを定義するが、このパラメタをゲートで保護された呼出しの間使う。呼び出す側は実際のパラメタ・リストを生成しなければならない。パラメタはイミューディエント値（リスト自身で渡される）またはポインタである（図 18）。

呼び出される側は、それが受け取ることになっているパラメタ、すなわちその数、アクセス権、字並び、長さ等を定義した検証リストを提供しなければならない。保護機構はゲートの入口でパラメタ・リストと検証リストとを比較して次のことを確認する。

- パラメタの数とそのフォーマットが一致するか
- 呼び出す側がすべてのパラメタにアクセスができるか
- 呼び出される側がすべてのパラメタにアクセスできるか

最初のチェックは自明である。第 2 と第 3 のチェックはエラー封鎖と耐故障性の原則をサポートする。アクセス制御機構をやぶろうとする主体 S を考えてみる。その現在のドメイン中では、S は一つのバンク B への読み込みアクセスしかもっていない。ドメインの中の B に対する W-アクセスをもつルーチン呼び出し、そのルーチンへのパラメタとして、書き込みたい B のある領域へのポインタを渡すことによって、S は B へ書き込みしようとする。この時、呼び出されたルーチンは S の代わりにシステムの完全性を侵犯するため、行動する。同様に、もし呼び出される側が、アクセス権をもたないある領域を指すパラメタを渡されたならば、同様の違反が起こるのであろう。このとき呼び出される側は、呼び出す側から多分知らないで渡されたパラメタにアクセスしようとした時、エラーを起こすだろう。

保護機構はゲートを通る時点でそのようなエラーをただちにつかみ、呼び出される側がエラーを起こすのを防ぎ、そして呼び出す側にただちに割込みを出してエラーを封鎖する。

保護機構は、呼び出す側と呼び出される側のアクセス・キーを用いて表示されたモードで、すべてのポインタ・パラメタへのアクセスをチェックして、これらのエラーをとらえる。もし両側が、パラメタにアクセスできないならば、主体は割込みを受け取る。

4) 潜在的パラメタ

このパラメタは呼び出す側から明示的には渡されない。これは 1 語の値であり、呼び出

す側のゲートにストアされ、呼び出す側によって任意の目的にも使用される。これは呼び出される側に、コールの間のデータを保持する能力(ALGOL の OWN 変数のように)を与え、そしてデータを外部の操作からかくす。潜在的パラメタはいかなるデータ構造をも指すことができる。パラメタ・リストを検証した後、ゲート機構は潜在的パラメタをレジスタ R0 にロードして、入口で呼び出す側に潜在的パラメタが保護されたゲート・バンクにストアされているので、呼び出す側はこれを破壊することはできない。

要約すれば、ゲートは目標ルーチンへの制御された入口である。呼び出す側は E アクセスのゲート・バンクとゲートへのオフセットだけを見ることができ、ゲートへの読み書きはできず、呼び出される側の検証リストをみることはできない。また、潜在的パラメタを破壊することもできない。

4.2 ソフトウェア・サブシステム

前述の三つのアーキテクチャ要素(アドレス付け、アクセスとプロシージャ制御)は信頼できる閉じた環境を生み出す。しかし、これらの構成物をセキュリティの次の層を形成するため利用するのはシステム・ソフトウェア次第である。ハードウェア・サポートは究極のセキュリティ目標を達成するのに必要であるが、十分な先行条件ではない。次の上の層へのキーはソフトウェア・サブシステム——独立に設置でき、置き換え可能で修復できる共通リンケージと保護環境を共有するバンクのグループである。各サブシステムごとに存在する一意的なサブシステム定義と呼ぶエレメントがこの環境を定義する。制限されたアクセス・モジュール性および耐故障性によって、ソフトウェア・サブシステムは、システムとユーザのアプリケーション・レベル^[9]における保守性、信頼性、およびセキュリティの向上に寄与するであろう。

5. おわりに

本稿では、高水準のセキュリティ目標に対するアーキテクチャ・サポートを中心にした。“セキュリティ”は相対的な言葉として理解されなければならない。いかなるコンピュータも絶対的に安全で信頼できるということはない。しかしながら、実行時のエラー封鎖を通して信頼性とセキュリティを大いに改善することができる。

システムの信頼性はその最も弱いリンク(ハードウェアまたはソフトウェア)と同じ程度しか良くないから、正当性の証明と対照的に、エラーは必然的に起こる。このリンクが切れた時、予期しない環境での突然の動作のために最大の損害をこうむるのは、証明された、正しいコンポーネントである。エラーには耐えねばならない。病気とよく似て、エラーを早くキャッチし、正確に分類し、そして処置しなければならない。

制御された方法で、エラーをより高い抽象化レベルに渡し、あるいは、理想的には、それをただちに訂正し、これを高レベルにみえないように処置する。この方法によってエラーが発生してもシステムを信頼できる状態に戻すことができる。

この安全な環境は非常に下のレベルから作り上げられなければならない。ハードウェア・エラーの検知と修正という領域では多くのことがわかっている。エンジニアは過去 20 年間これらの原理を論理設計に適用してきた。しかしながら、このエラーの封鎖という考え方がオペレーティング・システムのレベルまで広がるのには時間がかかった。安全なオペレーティング・システムのモデルはこの考えを受け入れるようになった。これらのモデルは数学的証明によって吟味でき、また十分実現できる実際的な形式化が可能である。

アクセス行列モデルは、そこで主体と対象の間の相互作用を管理する環境を作るための

基礎である。モデルは多くの実現案を示唆するが、代替案の選択は、オペレーティング・システムの柔軟度に依存する。

1100/90 シリーズではアクセス行列モデルへの限られたアプローチをとっている。その基本命令がモデルの多様な属性をサポートする。すなわち、柔軟な共有、ドメイン、およびエラー封鎖はそれらの主なものである。セキュリティを犠牲にしないで、柔軟性を高める努力が続くであろう。これは実際には鋭く衝突する要求である。また、システム基本命令を保護するのと同じ方法で、ユーザ定義の基本命令（たとえばキュー、木、整列および探索）が保護できるようにデータ・タイプの抽象化が要求される。さらに、ますます複雑になる安全対策を施行するために、より洗練されたセキュリティ制御が要求されよう。そしてシステム内の全レベルですべてのシステム・コンポーネントの単純さ、信頼性および保守性が要求されよう。

P. Bergh と D. Tangman との議論を通して、1100/90 シリーズの保護機構の特徴の多くが理解できた、また、T. Turba, J. Kunston, T. Lee からの貴重なコメントで本稿を明瞭にできた、さらに、J. Reeves, L. Callaghan, B. Marphy からは技術的な支援を得たことを記す。

(プロダクト本部ソフトウェア1部 古沢純一、城川孝二 訳)

- 参考文献 [1] T. Cooper, "Extended Mode Addressing and Protection Model", Gull Lake Technical papers, Spring 1983.
- [2] P. J. Denning, "Third Generation Computer Systems", *Computing Surveys*, Vol. 3, No. 3, December 1971.
- [3] P. J. Denning, "Fault Tolerant Operating Systems", *Computing Surveys*, Vol. 8, No. 4, December 1976.
- [4] Department of Defense Computer Security Center, "Trusted Computer System Evaluation Criteria", DOD, August 1983.
- [5] T. A. Linden, "Operating System Structure to Support Security and Reliable Software," *Computing Surveys*, Vol. 8, No. 4, December 1976.
- [6] C. E. Landwehr, "Formal Models for Computer Security", *Computing Surveys*, Vol. 13, No. 3, September 1981. (山田真市訳, "コンピュータ・セキュリティの形式的モデル", bit 別冊 I, 1983.)
- [7] K. MacKenzie, "1100/90 CPU Software Support Architecture", USE Inc. Technical Papers, April 25-29, 1983.
- [8] E. I. Organick, "The MULTICS System: an Examination of its Structure", MIT Press, 1972.
- [9] Sperry conceptual RFC 1869, "OS 1100 Software Subsystems", September 1983.

執筆者紹介 Stephen J. Rawlins

1980年に Minnesota 大学においてコンピュータ・サイエンスで B. S. を取得。1980年 Sperry 社に入社し、言語システムの開発に従事。参入プロジェクトは、PADS, UPAS および UCS RTS。1984年に SPSD (Software Products System Design) 部門に移り、Export Systems に重点を置いた人工知能関連の研究に従事。



報告 二相流解析——その背景とモデリング**On Two-Phase Flow Analysis and its Modelling**

石 丸 潤

概要 相変化を伴う気液二相流の解析の研究手順を、動力炉・核燃料開発事業団での経験に基づいて述べ、流体方程式の各項を評価するタイムレベルおよび圧力非平衡モデルの概念的試み等について述べる。

Abstract A sample of research program is illustrated for two-phase flow analyses, which is based on the experiences at Power Reactor and Nuclear Fuel Development Corporation. Then the opinions and results from the author's studies are presented: an example for time level at which the terms are evaluated in the conservation equations, and a conceptual trial for two-pressure model.

1. はじめに

筆者は、昭和47年以来高速炉および軽水炉の冷却材沸騰現象の解析を中心に、1次元、2次元、3次元の非定常熱流解析に従事してきた。本稿では、出向先であった動力炉・核燃料開発事業団での業務上の活動、研究を通じて経験した二相流解析、とくにモデリングについて紹介する。

二相流は、①原子炉、ボイラ、MHD発電、地熱発電、ジェット・エンジン、②蒸発器、凝縮器、スプレッド冷却塔、冷凍機、抽出・蒸留装置、流動層、脱塩、乳化、吸収、吸着、③ガス・油混相輸送、スラリー、繊維、小麦、粉体等の水または空気輸送、ポンプ等のキャビテーション、ハイウェイ交通、④液晶、二相流潤滑、空調、集塵、沈降、⑤海の波、雨滴、氷粒、洪水、地すべり、雪崩、雲、流水、⑥血液、呼吸による体温調節等、産業や生活の中に数多くの例をみることができる。

本稿の範囲は、相変化を伴う気液二相流に限定する。気液系の相変化とは蒸発と凝縮であるが、その現象の物理、とくに界面での質量・エネルギー・モメンタムの移動については十分明らかにされているとはいえないのが現状であり、挑戦に値する課題である。

本稿では、まず2章において二相流解析の背景を明らかにし、現象の物理とモデルを結びつけていく方法論を整理するとともに、3章で、筆者の経験からモデリング上重要と考える問題を指摘し、研究成果・意見を述べる。

2. 二相流解析の背景

二相流解析が必要となった場合、適当な既存プログラムがないか、あっても一部変更しなければならぬとしたら、研究開発の現場では、どのようなアプローチをするだろうか。アプローチの一例を以下に示す。

A 問題の分析

A1: 問題の定義

A2: 解析対象の条件の範囲を定義

A3: 現象学的記述

A4: 研究成果発表の方法を検討

A5: 解析ツールの選択

B プログラム立案

- B1: プログラム評価方法の選定
- B2: プログラム概略仕様の決定
- B3: モデリング 1 (実現象から物理的モデルへ)
- B4: モデリング 2 (物理的モデルから数式モデルへ)
- B5: 数値解法の決定

C プログラム設計・作成

D プログラムの評価

E 問題解決のための解析的スタディ

これらを個々に詳しくみると、次のようになる。

A1: 問題の定義——筆者の経験した事例では、問題とは「ナトリウム冷却高速炉において起こるとは考えられないある異常事象がかりに発生し、冷却材ナトリウムが沸騰した場合の安全余裕を定量的に評価すること」であった。実プラントはまだ存在せず、直接に実験することはできない。最終的には、各種の実験データで計算プログラムを評価（検証）し、その検証されたプログラムで実プラント（のデザイン）の評価を行うというプロセスを踏む。そのためのプロジェクト・チームが生まれ、実験計画担当グループと解析担当グループとから構成され、筆者は後者に属した。国内外の理論的・実験的研究を比較分析し、特定のデザインについてとくに解明したい問題を抽出し、実験で実施するテーマと解析的手段で行うテーマを選別し、全般的な評価方法を定義した。実験は計算プログラムの検証に不可欠なデータを提供するので、解析チームは、あらゆる角度から実験計画を支援した。すなわち、予備解析によって実験計画の検討に加わり、計装点の位置・数、測定精度の要求を実験計画グループに提示した。また、ある種の実験装置は実プラントの縮尺モデルとするのであるが、縮尺の方法について動力的相似性の予備検討も試みた。

A2: 解析対象の条件の範囲を定義——圧力、流量、熱流束の範囲、加熱面と流路の幾何学的条件、予想される気相の体積分率（ボイド率）および重量分率（クォリティ）の範囲、気相流と液相流の方向（併流／向流）等を定義する。これらの条件は解析方法やツールの選択上重要である。たとえば、単相流の解析でも、流れの様相は管内か管群外側の流れか、タンク内の流れか、タンクの内部に複雑な構造物がある場合の流れか等々、流路の幾何学的条件によって異なってくることがあり、タンク内の3次元流の解析には、数値拡散の克服という難問が控えている。二相流の場合、単相流で生ずる流路形状依存性に加えて、幾何学的条件、構造物（内壁・内部構造材）の材質や表面の性質が、流れの構造（流動様式）、気液界面の特性に影響を与え、流れの挙動は、非常に複雑な条件依存性を示す。したがって、単相熱流解析プログラムはともかく、二相熱流解析プログラムに関するかぎり、適用可能範囲を絞らざるをえない。

A3: 現象学的記述——1982年、カルマン渦、自由表面等の写真アルバム (Van Dyke 著 An Album of Fluid Motion) が出版された。これらの写真が流体解析者にとって示唆に富むように、モデリングの前に現象を観察することは有益である。

二相流の定常・非定常特性は、流れの構造（流動様式）に高度に依存している。流動様式の模式図を図1および図2に示す。可能なら流動様式も観察する。水の沸騰の場合は金属製の装置に観察用のガラス窓を設け、ビデオ・カメラで撮影し、これを眺

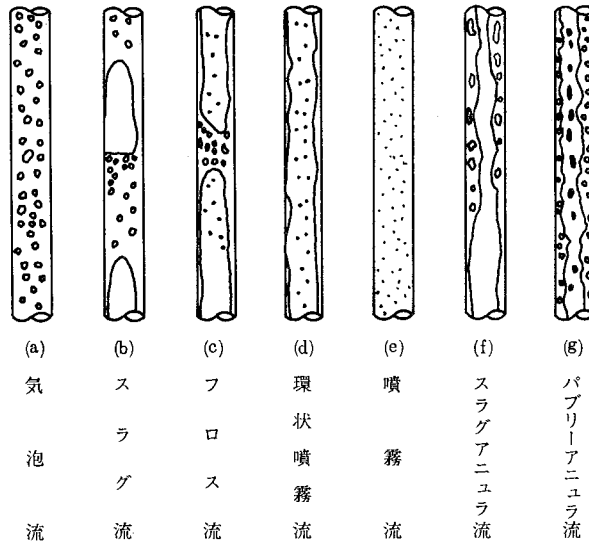


図 1 垂直管中の流動様式

Fig. 1 Flow patterns in vertical flow

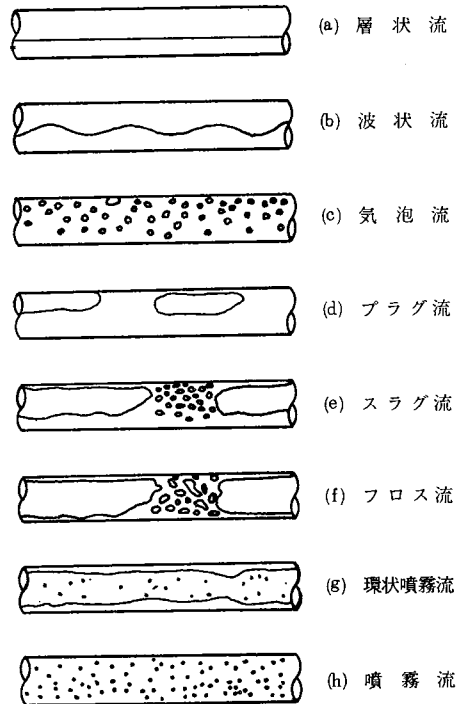
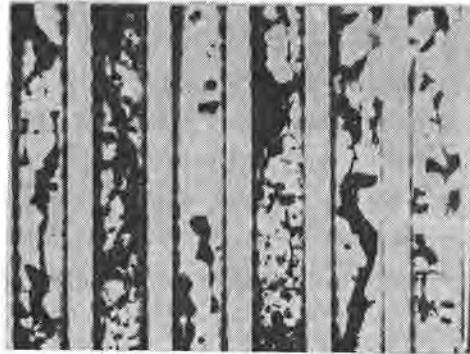


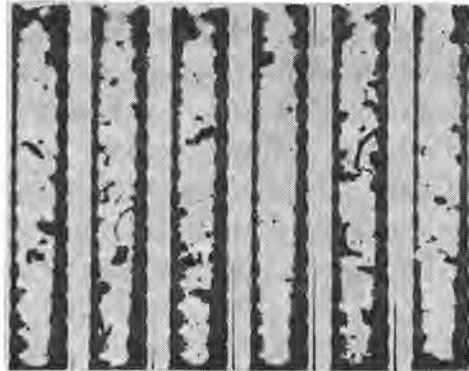
図 2 水平管中の流動様式

Fig. 2 Flow patterns in horizontal flow

めながら現象学的記述を行う。これによって、後に解析結果の16mm映画またはVTRと並べて比較し、モデルの問題点を探ることができる。ナトリウムのような液体金属は不透明なので何らかの方法で可視化し、流れの構造をつかむ。図3に示す水銀の場合は、重い金属である流体と管壁のX線透過性の差を利用してX線撮影が可能である



(a) 約5%クオリティ (about 5% quality)



(b) 液滴流れ (droplet flow)

図 3 水銀沸騰のX線写真^[5]

Fig. 3 X-ray photography of boiling mercury

が、ナトリウムの場合は、透過性が一般の管壁材料より大きいので工夫が必要であろう。

つぎに、沸騰・凝縮のメカニズムの研究状況は、二相間の質量エネルギー・モメンタム交換のメカニズムについての理論的・実験的知見を整理して、新しい定式化への挑戦段階である。また不明なところがあれば、それを補う基礎実験を行わなければならない。

A4: 研究成果発表の方法を検討——研究者は、当初から結論をある程度予想する。予定した発表形式で、予想される結論をアピールするために最も効果的な材料(実験事実、解析結果、図表等)の集め方、仕事の進め方・切り上げ方を考えている。日本原子力学会、機械学会、伝熱シンポジウム、国際水理学会 (IAHR)、米国原子力学会 (ANS)、米国機械学会 (ASME)、OECD の液体金属ワーキング・グループ (LMBWG) 等での発表では、次第にスライドや Vugraph (OHP) だけでなく、カラー映画や VTR がもっと多く使われるようになるであろう。実験データまたは解析結果を映画や VTR にする場合、流体の動きをみやすくするには、流れにトレーサとして金粉を入れるように、粒子運動 (particle motion) で表現する。

A5: 解析ツールの選択——現象学的追求は続けながらも、解析目的や解析対象の条件等が明確になれば、解析ツールの選択を進める。ここでは、既存の入手できるプログラム目的性を検討し、必要なら独自に開発を進めることになり、解析可能な条件の範囲は当面の問題よりやや広くとる。

B1: プログラム評価方法の選定——プログラムを開発するに当たり、プログラムとその計算結果の評価方法を十分に考慮する必要がある。評価に用いる例題は、一般に、簡潔で解析解または簡易手計算でチェックができることが望ましい。

現象自体が複雑な二相流問題では、範囲が限られてくるが二相混合体の相分離が考えられる。また、実データのある標準的例題（水について、米国原子力規制委員会(NRC)の標準問題）をも選ぶ。さらに、開発の意図によく合致するものをいくつか選ぶ。一方、その分野の研究者に広く受け入れられる標準的な例題であること、および競合する他のプログラムの評価にも使用される可能性の高い例題であることも、選定の条件となる。

B2: プログラム概略仕様の決定——プログラムの目的、機能、扱いやすい入力形式、図形出力を含む出力形式等の概略仕様を決定する。

B3, B4: モデリング1（実現象から物理的モデルへ）およびモデリング2（物理的モデルから数式モデルへ）——この二つの抽象化は明らかに異質であり、それぞれのプロセスで仮定と問題の簡略化が行われる。二相流の定常・非定常特性は、流れの構造に高度に依存しているので、流れの構造とその界面での質量-エネルギー-モメンタムの移動メカニズムに関する仮定が最も重要となる。たとえば図4は環状流の断面であり、中央の連続気相中に液滴がみられる。この液滴の存在を考えることは、上記のモデリング1のレベルの範囲であり、その液滴の表面積のバランス式を書き下すのはモデリング2の範囲である。二相流問題は、数学的には、一つの場合が複数の単相流の領域に分かれ、それらの間に移動する境界が存在すると考えられる。すなわち、各相の質量-エネルギー-モメンタムの保存方程式、状態方程式、各相および界面の質量-エネルギー-モメンタムの移動に関する相関式を組み合わせ、適当な境界条件を与えて解く。したがって、各方程式各項の数値的大小を吟味し、ある項がA2で考えた条件の下で無視してもさしつかえない場合、これを方程式から除くことがあり、また未知数の数と方程式の数が一致し、数式の系が閉じることを示して、この段階を終える。

B5: 数値解法決定——数式モデルに現れる微分に対し適当な差分スキームを選択し、微分方程式を差分化する。各方程式の各項を評価するタイム・レベルを選択することは、相変化、対流、拡散等の現象の応答時間と関係していて、モデリングの範囲である。これにより各項や変数が最終的に解くべき連立代数方程式系の係数マトリクスに入るか、定数ベクトルに入るか、のどちらかに分かれ、完全陽 (Full Explicit) から完全陰 (Full Implicit) までのスペクトルの中に位置づけられる。最後に、連立方程式の



図4 環状流の断面の写真^[7]

Fig. 4 Cross-sectional view of annular flow

解法を選択する。

C: プログラム設計・作成——二相流解析は、概して計算時間が長大となるので、今後はベクトル・プロセッサを意識した数値解法とプログラム設計が必要になってくるであろう。

D: プログラムの評価——B 1 で選定した方法でプログラムを評価し、もし不十分なら適当な段階まで戻って再検討をする。

E: 問題解決のための解析的スタディ——検証されたプログラムを用いて、問題の解析を行い、実験的研究と相補して問題を解決する。

3. モデリング

二相流のモデルにおいて、微視的定式化が困難な理由として Ishii^[6]は、二つの理由をあげている。すなわち、①相と相の間に界面が存在し、変形し、その界面の運動を記述するのが困難であり、各相の方程式と界面の条件との結合を複雑にする。②乱れと界面運動とによる変数のゆらぎの存在によって、Navier-Stokes の式と界面の波の不安定性から統計的性格が入り込む。

したがって微視的なゆらぎを時間軸上で平均化し (time averaging), 状態変数をノード・セルの平均値として定義することが多い。

以下にモデリング上重要と考える問題について述べる。

3.1 単相流問題との比較

D. B. Spalding^[7]は、二相流の問題を単相流と比較して、次の三つのポイントを指摘している。①単相流より多くの式を解き境界条件も多いが、解くべき方程式の種類は同じであるから、方程式の数それ自体は解を得る上でより困難にさせているわけではない。②各相のモメンタムの式の間で1個の圧力を共有している。③収束が遅い。

圧力の共有については、後述する。また収束が遅いのは、二相流モデル式の高度の非線形性が原因である。すなわち、係数が未知数の関数になっているため、反復法によって解かなければならないが、単相流よりも非線形性が大きい。

二相流問題のむずかしさの原因としてさらに次の三つの点に触れて置こう。①密度の不連続性すなわち ($\partial\rho/\partial h$) の不連続性がとくに低圧で著しい。しかし、これは均質流-熱平衡モデル (HEM: Homogeneous Equilibrium Model) では二相混合体の質量-エネルギー-モメンタムについての保存式を考えるために生ずることであり、二流体モデルなら各相については不連続性がないので問題は起こらないはずである。②流動様式や界面の物理が十分明らかになっていない。③筆者が最も大きな要因と考えるのは、Euler 座標の固定メッシュ体系内を沸騰境界が時間とともに移動する点である。通常の Euler 表示、連続流体仮定のモデルを用いて、二相流問題を完全に解いているプログラムはないといってよい。不完全ながらも比較的楽に解いているのは、計算体系の一端から気相流が流出していて、沸騰境界が1個しかないような1次元問題であり、次に容易なのは、計算体系内部で沸騰が開始し、両端に向かって沸騰が拡大する1次元問題である。2次元になると沸騰の拡大とともに格段に沸騰境界に位置するノードの数が増大し、3次元ではさらに飛躍的に増大する。したがって、3次元二相流問題が最も数値的に困難である。

3.2 二相流モデルの分類

Schor と Todreas^[10]が示した一覧表が包括的でよく整理されているので表1に引用する。各モデルの名称の一般形を $nCmT$ で表し、 n は保存方程式 ("C") の全数、 m は "T"

表 1 二相流モデルの分類 (共通の仮定 $P_l = P_g$)^[10]

Table 1 Two-phase flow models

名 称	保存方程式の数				制約条件の数			相 関 式 の 数					
	M	E	K	計	T_a	U_r	計	壁 面		2 相界面			計
								Q_w	F_w	Γ	Q_i	F_i	
3C	1	1	1	3	2	1	3	1	1	0	0	0	2
4C 2M	2	1	1	4	1	1	2	1	1	1	0	0	3
4C 2E	1	2	1	4	1	1	2	2	1	1*	1	0	5
4C 2K	1	1	2	4	2	0	2	1	2	1*	0	1	5
5C 1K	2	2	1	5	0	1	1	2	1	1	1	0	5
5C 1E	2	1	2	5	1	0	1	1	2	1	0	1	5
5C 1M	1	2	2	5	1	0	1	2	2	1*	1	1	7
6C	2	2	2	6	0	0	0	2	2	1	1	1	7

記号の説明: M=質量保存方程式

 T_a =相 a の温度 ($a=g$ または l) Γ =質量移動 (相変化)

E=エネルギー保存方程式

 U_r =相対速度= $U_g - U_l$

Q=エネルギー移動 (熱伝達)

K=モメンタム保存方程式

F=モメンタム移動 (摩擦)

(注) * Q_i や F_i が必要な時は, 必ず Γ が要することに注意.

($T = M(\text{質量})/E(\text{エネルギー})/K(\text{モメンタム})$) で指定された物理量の保存式の数を示す. 保存式の数と制約条件 (imposed restrictions) の数の和が6であることに注意されたい. 均質流-熱平衡モデル (HEM) は, この一般形によれば, 3Cモデルであるが, 液相・気相の温度が飽和温度に等しく, 両相の速度が相等しいという制約条件が内在している. 保存式の数が増加するにつれ, 相関式 (Constitutive Relations) の数が増加する.

3.3 二相流における三つの時計

われわれの方程式で表現される現象が3種類のタイム・スケールに関連していることを認識することが重要である^[9]. 半定量的に時定数を示すために, 流速を1~30 m/sec程度と仮定する.

1) 時定数 $10^{-6} \sim 10^{-5}$ sec: 圧力波伝播

急激な流出入の開始や流れ方向の急変等によって生ずる圧力波は流体の中を音速で伝播する. 液中での音速が非常に速いので圧力波伝播の過渡応答時間は $10^{-6} \sim 10^{-5}$ secで, われわれのアプリケーションで採用するノード・セグメントの大きさからみるとまったく小さい. したがって, 圧力波伝播に関する項は完全に陰に扱わなければならない.

2) 時定数 $10^{-4} \sim 10^{-3}$ sec: 質量-エネルギー交換, エネルギー対流-拡散

気-液間および流体-壁間の相互作用の時定数は非常に短いものからかなり長いものまで広く分布している.

そのうち, 相変化の応答が最も速く, 液体金属の場合は界面熱伝達, 壁面熱伝達の各項も比較的短い. エネルギー対流項とエネルギー拡散項も, とくに液体金属の場合比較的短い.

応答の最も速い相変化は高度に陰に, 他の項も十分高度に陰に扱う必要がある.

3) 時定数 10^{-2} sec 以上: 壁面摩擦, 界面摩擦, モメンタム対流-拡散

これらの項は, 陽に扱うことができ, その場合クーラン条件の制約を受ける.

3.4 各項を評価するタイム・レベル

前節で述べたように, 三つの保存方程式中の各項が表現する個々の現象の応答時間はまちまちであり, 速い現象を扱う項は新しいタイム・レベルで, 遅い現象を扱う項は古いタ

表 2 流体方程式各項を評価するタイム・レベル
Table 2 Terms and time level in conservation equations

方程式	項	記号	タイム・レベル			
			筆者のモデル(3)	COBRA/TRAC(13)		
質量保存	対流	$\alpha_k \rho_k$	new	old		
		u	new	new		
		$(\alpha(1-\alpha)\rho A_i)_k$	old	—		
		$(H_i A_i)_k, C_{\rho k}, h_{fg}$	—	old		
		$T_l - T_{sat}$	new	new		
エネルギー保存	対流	$\alpha_k \rho_k$	mid*	old		
		u_k	mid*	new		
		I_k または \dot{h}_k	new	old		
		$(\Gamma_o - \Gamma_c) H_{gs}$	new	new		
		相変化によるエネルギー束				
		界面熱伝達	q_{ik}	new	new	
壁面熱伝達		H_{wk}, T_w	old	old		
		T_k	new	old		
		モメンタム保存	対流	$\alpha \rho u \cdot u$	old	old
				界面摩擦	\hat{F}_i	old
壁面摩擦		\hat{F}_{wk}	old	old		

下付記号の説明： k (相), i (界面), W (壁面), l (液相), g (気相)
* mid : old と new の中間で評価することを示す。

タイム・レベルで評価する。表 2 で筆者のモデルと COBRA/TRAC のモデルを比較した。

3.5 物理的不安定性と非物理的不安定性

二相界面における表面張力、粘性応力等の効果がモデルに取り込まれていると、物理的不安定性を表現することができる。たとえば振動的な沸騰 (oscillatory boiling) では、沸騰境界は (3 次元的に) 一様に滑らかに広がるのではなく、不規則に前進・後退を繰り返す。これは流量、熱流束および流動様式等に支配され、異なる条件では、もっと安定に沸騰が拡大する。一方、単気泡の成長過程を表現するために表面張力の項を取り入れることがある。

多次元モデルで表面張力等の十分な物理的記述がない場合、二相界面での不連続・不安定性 (たとえば界面速度がないために不連続となる) は、非物理的不安定性になる。モデルの非物理的不安定性は、いわゆる “ill-posed” といわれる問題であるが、物理的不安定性の問題はそうではない。

通常、われわれの二流体モデルでは、一つのノード・セルにおいて二つの流速と二相に共通の一つの圧力を考え、各相の圧力が瞬時に平衡に達すると仮定する。界面をはさんでの圧力の伝播は速いとはいえ、有限の速度であるから、これを無限大であるとするのは、非現実的である。この非現実的な仮定によって数学モデルの線形方程式系の固有値が複素数となり、そのため “ill-posed” な初期値問題となる。V. H. Ransom と D. L. Hicks^[12] は、相 1 の連続の式、相 2 の連続の式、相 1 のモメンタムの式、相 2 のモメンタムの式、界面運動の式、相 1 のエントロピの式、相 2 のエントロピの式、二相混合物の横方向モメンタムの式の 8 個の方程式を解く 2 圧力モデルを提案し、通常の 1 圧力 6 方程式モデルの計算結果と比較した。この場合、任意の空間位置 (3 次元的ノード・セル) について各相の圧力を未知数とするのは、解析者が最終的に欲する詳細さとして二つの圧力値が必要なためではなく、あくまでも “well-posed” な問題とするためである。

このような事情は、各相の温度が瞬時に平衡に達すると仮定する熱的平衡モデルと各相の温度を求める熱的非平衡モデルとの間についてもいえる。筆者の経験を例にとれば、高

速増殖炉炉心のナトリウムの二相流では相変化が起こっている時に、二つの相の温度の差が1°Kを超えることはまずない。通常は0.001~0.3°K程度の値であった。それにもかかわらず、このわずかな温度差を問題にするのは、それが相変化速度を支配するからであり、そのため、エネルギー保存式は解に強く結合されなければならない。(熱的平衡を仮定することは相変化速度を無限大と仮定することであり、非現実的な不連続性をもたらす。)

3.6 圧力の非平衡

A. L. Schor ら^[9]は、速度のみの非平衡を考えた4方程式モデルを提案したが、通常の二流体モデルは、6方程式モデルと呼ばれ、速度と温度の非平衡を考える。V. H. RansomとD. L. Hicks^[12]は、圧力の非平衡までを考えた8方程式モデルを提案した。しかし、彼らの定式化では、質量交換(相変化)速度 $\dot{m}=0$ の制約条件を付している。そこで筆者は、非常に大きい相変化量がある場合の圧力の非平衡モデルを考えてみた。

まず、相 k の圧力 P_k の界面上の圧力 P_i からの偏り δP_k を次式で定義する。

$$\delta P_k \equiv P_k - P_i \quad (3-1)$$

各相の δP は、

$$\delta P_l = -2C_{gl}\sigma \left(\frac{\rho_l}{\rho_l - \rho_g} \right) + \frac{(\dot{m}_l)^2}{2} \left(\frac{1}{\rho_g} - \frac{1}{\rho_l} \right) \quad (3-2)$$

$$\delta P_g = -2C_{gl}\sigma \left(\frac{\rho_g}{\rho_l - \rho_g} \right) + \frac{(\dot{m}_l)^2}{2} \left(\frac{1}{\rho_l} - \frac{1}{\rho_g} \right) \quad (3-3)$$

と表せる。

ただし C_{gl} は気相側の液相に対する界面の曲率(次元は L^{-1})、 σ は表面張力、 \dot{m}_l は単位界面積当たり相変化によって液相から失われる質量速度(蒸発のとき正、凝縮のとき負)である。両式の右辺第1項は表面張力の、第2項は相変化速度の寄与である。

環状流について考えると、流路の中心を気相が流れ、壁面に近い周辺部に液相が流れる。この場合、界面の曲率は十分小さく(曲率半径は大)、表面張力による界面での圧力差は無視できるものと仮定し、相変化速度による寄与のみを考える。定義から、

$$\dot{m}_l = (\Gamma_e - \Gamma_c) / A_i \quad (3-4)$$

ただし、 Γ_e 、 Γ_c はそれぞれ単位体積当たりの蒸発速度および凝縮速度、 A_i は単位体積当たりの界面の面積、 $T^{\text{sat}}(P_i)$ は界面上の圧力 P_i に対応する飽和温度である。式(3-4)を用いて相変化速度のみの寄与を考えると、

$$P_l - P_i = \frac{1}{2A_i^2} (\Gamma_e - \Gamma_c)^2 \left(\frac{1}{\rho_g} - \frac{1}{\rho_l} \right) \quad (3-5)$$

$$P_i - P_g = \frac{1}{2A_i^2} (\Gamma_e - \Gamma_c)^2 \left(\frac{1}{\rho_l} - \frac{1}{\rho_g} \right) \quad (3-6)$$

となり、圧力は、 $(\Gamma_e - \Gamma_c)$ の符号のいかんにかかわらず、常に密度の高い方の相(液相)で高いことがわかる。

限界熱流束に達する前(pre-CHF)の気液界面近傍の圧力および温度プロファイルの模式図を図5、図6に示す。

図5は1圧力モデル、図6は2圧力モデルを表している。未知数は唯一の $P(x, y, z)$ から $P_l(x, y, z)$ 、 $P_i(x, y, z)$ 、 $P_g(x, y, z)$ に2個増加するので、式(3-5)、(3-6)を追加する。モメンタムの式では P のかわりに P_l および P_g を用いるが、最終的に解くべき圧力の方程式に式(3-5)、(3-6)を代入して界面圧力 P_i についての式とすればよい。

式(3-5)、(3-6)に具体的数値を代入してみよう。

$$(\Gamma_e - \Gamma_c) = 100 \text{ kg/sec} \cdot \text{m}^3, A_i = 1 \text{ m}^2/\text{m}^3, \rho_g = 0.5 \text{ kg/m}^3, \rho_l = 800 \text{ kg/m}^3, P_i \sim 2 \times 10^5$$

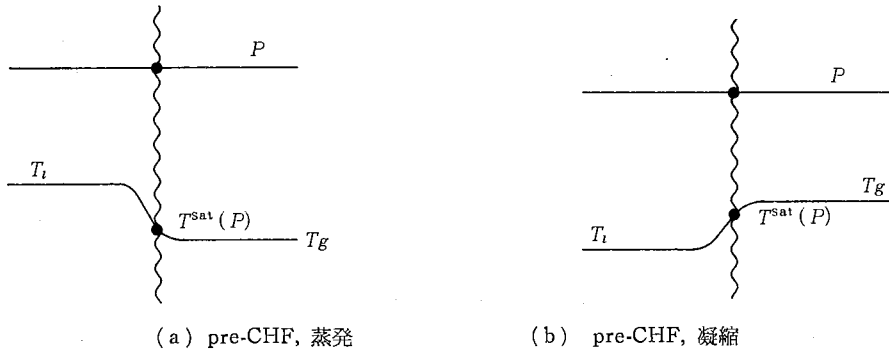


図 5 1 圧力モデル

Fig. 5 One-pressure model

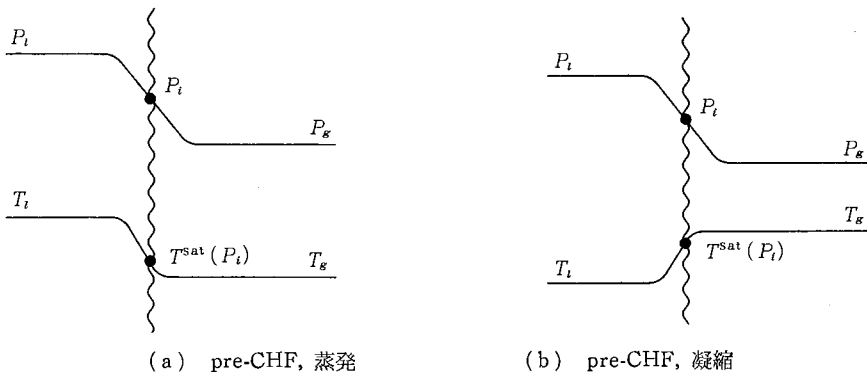


図 6 2 圧力モデル

Fig. 6 Two-pressure model

Pa 程度の低圧ナトリウム沸騰の場合 δP_i は 10^4 Pa にも達し、これを無視するのは現実的ではないことが明らかとなる。

4. 今後の課題

これまでの軽水炉およびナトリウム冷却高速増殖炉の安全解析の分野の二相流モデルの歴史をみると、あきらかに均質流モデルからドリフト・フラックス・モデルへ、さらに二流体モデルへと移ってきた。表 1 で示したように、二流体モデルには前提となる制約条件がない。このことは、物理現象を最も機械論的 (mechanical) に表現できることを意味する。しかし、この 6 方程式モデルは、① 相関式が数多く必要、② 計算時間が長い、③ 二相間の相互作用の物理学について実験や解析から判明していることがあまりにも少なく、モデルの複雑さに応えうるだけ十分かつ詳細なデータベースがまだない。②、③ は現時点での問題であり、コンピュータの発達と、データベースの蓄積により漸次解決されるであろう。

数値的不安定性をもたらす非現実的な仮定は可能な限り少ない方がよく、均質流-熱平衡モデル (HEM) では条件によってはまったく解が得られないことがある。二流体モデルにも非現実的な仮定があり、それを克服していくことが今後の課題である。一つは、環状流における液滴の無視であり、もう一つは圧力平衡の仮定である。

環状流ではエントレメントを考え、連続液相、連続気相のほか、分散相としての液滴を含めて 3 フィールドまたは気泡を含めて 4 フィールドのモデルを検討していく必要がある。COBRA/TRAC^[13] は、3 フィールド・モデルである。

圧力非平衡を考慮した各相の圧力の絶対値は、必ずしも正確度 (accuracy) 上要求されるわけではない。圧力の平衡の仮定からくる非物理的な圧力の振動 (pressure spike) が生ずると、相変化速度の式を通じて、質量とエネルギーの保存にインパクトを与え、収束を遅くしたり、発散させたりする。圧力の非平衡を許す 2 圧力モデルの適用は、この点の解決に寄与するであろう。

また、高速のコンピュータが求められており、ベクトル・プロセッサもその一つとして使われ始めている。今後は、ベクトル・プロセッサ向きの数値解法、すなわち、ベクトル化しやすいアルゴリズムの研究が必要である。

相変化速度と界面熱伝達モデル、これと固定メッシュ体系内を沸騰境界が移動することによって生ずる不連続ショックの間に関係に解釈を与えることを、筆者自身の当面の課題としたい。

5. おわりに

二相流解析に従事し、現象を“見る”こと、現象の物理を知ること、基礎に戻ることの重要性、暗黙のうちに用いている仮定の妥当性を何度もチェックし直すことの大切さを痛感している。

二相流解析プログラムの歴史を顧みると、ある面からみれば、コンピュータの発達によって発展してきたともいえる。その意味で、物理や数学からの発想とともにコンピュータの科学と技術からの発想を育てたい。

末筆ながら、動力炉・核燃料開発事業団において御指導いただいた皆様に感謝します。

-
- 参考文献 [1] J. Ishimaru et al., “A Verification Study of the ASFRE Code through Experimental Analyses to Local Flow Blockage Tests”, 9th LMBWG Meeting, Rome, 1980.
- [2] 石丸 潤他, “局所閉塞実験解析による ASFRE の検証”, 昭和 55 年原子力学会秋の分科会, 1980.
- [3] J. Ishimaru, et al., “An Application of BOCAL to the W-1 SLSF Experiment”, *PNC Report*, March, 1984.
- [4] 赤川浩爾, 「気液二相流」コロナ社, 1974.
- [5] O. C. Jones, Jr., *Nuclear Safety Heat Transfer*, Hemisphere Publishing Corp., 1981.
- [6] G. F. Hewitt, N. S. Hall Taylor, *Annular Two-Phase Flow*, Pergamon Press, 1970.
- [7] D. B. Spalding, “Numerical Computation of Multi-Phase Fluid Flow and Heat Transfer”, *Recent Advances in Numerical Methods in Fluids*, Vol. 1. Pineridge Press, 1980.
- [8] M. Ishii, “Thermo-Fluid Dynamic Theory of Two-Phase Flow”, Eyrolles, 1975.
- [9] A. L. Schor, N. E. Todreas, “A Four-Equation Two-Phase Flow Model for Sodium Boiling Simulation of LMFBR Fuel Assemblies”, 10th LMBWG Meeting, Karlsruhe, Oct. 27-29, 1982.
- [10] M. S. Kagimi, “Fundamentals of Two-Phase Flow Modelling”, presented at MIT Course, Advances in Nuclear Systems Thermal Analysis, April 14-16, 1982.
- [11] A. L. Schor, “Advances in Numerical Methods”, presented at MIT Course, Advances in Nuclear Systems Thermal Analysis, April 14-16, 1982.
- [12] V. H. Ransom, D. L. Hicks, “Hyperbolic Two-Pressure Models for Two-Phase Flow”, *Journal of Computational Physics*, 53, 1984. pp. 124-151,
- [13] M. J. Thurgood, J. M. Kelly, T. E. Guidotti, R. J. Kohrt, K. R. Crowell, “COBRA/TRAC-A Thermal-Hydraulics Code for Transient Analysis of Nuclear Reactor Vessels and Primary Coolant Systems”, *NUREG/CR-3046*, March, 1983.

執筆者紹介 石丸 潤 (Jun Ishimaru)

昭和43年早稲田大学工学部応用化学科卒業，46年日本ユニバ
ック総合研究所入社，その後日本ユニバック(株)に移籍，47年以
来，高速炉および軽水炉の安全解析を担当，現在に至る。



情報処理教育のカリキュラム
Model Curricula in Computers
and Information Systems

朝倉文敏

1. ACM: C³S のカリキュラム 68

米国においてコンピュータ・サイエンス教育の重要性が叫ばれ始めたのは1950年代後半のことである。そのきっかけとなったのは、1957年のソ連による世界初の有人宇宙衛星スプートニクの打ち上げであった。宇宙開発レースにおける米国の敗北は、米国政府関係者に科学技術教育の立ち遅れに関する深刻な危機感をもたらした。このため、J.F. KennedyとL.B. Johnsonの両大統領による「偉大な社会の建設」のスローガンのもとで科学技術教育が見直された。教育の革新が叫ばれ、多額の財政援助が与えられた。なかでもコンピュータ教育の導入が強調され、1960年から1965年にかけて続々と各大学にコンピュータが設置され、工学教育の一環としてコンピュータの利用が奨励された。たとえば、1962年にはStanford大学とPurdue大学で、最初のコンピュータ・サイエンス学科が設立されている。

さらに、1960年代中頃には米国科学アカデミーのRosser報告(1966年)と大統領諮問委員会のPierce報告(1967年)が発刊され、これらによって米国におけるコンピュータ教育政策の基本が定められた。そして、これらの勧告に呼応し1968年には米国コンピュータ学会(ACM)のコンピュータ・サイエンス・カリキュラム委員会(C³S, Curriculum Committee on Computer Science)が学部レベルのコンピュータ・サイエンス学科のカリキュラム案を発表している。これはカリキュラム68と呼ばれ、米国の人文および理学を対象とするいわゆるリベラルアーツ系の大学で採用された。そして、カリキュラム68は、欧州および日本の大学においても、情報関連の学科の新設構想に大きな影響を与えた。

2. ACM: C³EM のカリキュラム 72 および C³C のカリキュラム 78

60年代後半になると、経営におけるコンピュータの有用性が認識され、事務処理の機械化が推進され、MIS、情報検索・航空座席予約・バンキング等の大規模システムの構築が始まった。そして、これらのシステムの開発では、コンピュータ・サイエンスの知識だけでなく、経営組織、人間および組織行動、システム分析およびシステム設計、経営学等の

知識も必要であった。

そこで、ACMはマネジメント・コンピュータ教育カリキュラム委員会(C³EM, Curriculum Committee of Computer Education on Management)を発足させ、情報システムの開発・管理に携わるマネジメントとスペシャリストの育成を目的としたカリキュラム案の検討を開始させた。同委員会は、修士レベルおよび学部レベルのカリキュラムを、それぞれ1972年と1973年に発表している。

これらのカリキュラムは、米国のビジネス・スクール(経営大学院)や大学の経営学部で採用された。なお、大学院レベルの勧告案は、わが国の情報処理技術者試験に対するガイド「上級情報処理技術者育成指針」の土台としても用いられた。

さて、1970年代の中頃には、カリキュラム68の内容も時代に即さなくなってきたため、ACMのC³Sはカリキュラム68の改訂に着手した。そして、1979年3月にはそれをカリキュラム78として発表している(表1)。

表1 ACMカリキュラム78
Table 1 ACM curriculum 78
(Comm. ACM March 1979より)

C S 1	プログラミングI
C S 2	プログラミングII
C S 3	コンピュータ・システム
C S 4	コンピュータ・ハードウェア入門
C S 5	ファイル処理入門
C S 6	オペレーティング・システムとコンピュータ・アーキテクチャI
C S 7	データ構造とアルゴリズムの解析
C S 8	プログラミング言語の構成
C S 9	コンピュータと社会
C S 10	オペレーティング・システムとコンピュータ・アーキテクチャII
C S 11	データベース管理システムの設計
C S 12	人工知能
C S 13	アルゴリズム
C S 14	ソフトウェア設計と開発
C S 15	プログラミング言語の理論
C S 16	オートマトン、計算可能性、形式言語
C S 17	数値計算—解析
C S 18	数値計算—線形代数

カリキュラム78の特徴は、プログラミング技能の重視にあった。このため、「正しく動き、きちんと文書化された読みやすいプログラムを適切な時間内に作成する能力の育成」をその目標として掲げている。また、このカリキュラムは、1968年以後に発展したソフトウェア工学の影響を強く受けており、工学的色彩が強いものであった。この点は、コンピュータ・サイエンスの理解を強調し、理論を重視し、プログラミング技能の修得を副産物と考えていたカリキュラム68と比較すると対照的である。

また、カリキュラム78は、1968年以後の技術的

成果として構造的プログラミング、データベース技術、ソフトウェア開発プロジェクトの管理等を取り入れたほか、社会生活におけるコンピュータの重要性の増大を背景に、コンピュータ技術の社会的評価のコースも加えられている。カリキュラム 78 は、総体としてみると、理論と実践の面でバランスがよくとれているものとなった。また、カリキュラム 78 に接続される修士課程のカリキュラムが 1981 年 3 月に発表されている (表 2)。これをみると、修士課程のカリキュラムの充実振りがうかがえる。

3. ACM: C²IS のカリキュラム 80

カリキュラム 72 の方も、ACM の情報システム・カリキュラム委員会 (C²IS, Curriculum Commi-

表 2 ACM のコンピュータ・サイエンスの修士レベル・カリキュラム
Table 2 ACM computer science curriculum (Master level)
(Comm. ACM March 1981 より)

C S 19	コンパイラの構造
C S 20	プログラミング言語の形式論
C S 21	アセンブラのアーキテクチャ
C S 22	性能評価
C S 23	オペレーティング・システムの解析的モデル
C S 24	コンピュータ・コミュニケーション・ネットワークと分散処理
C S 25	高水準言語コンピュータ・アーキテクチャ
C S 26	大型コンピュータのアーキテクチャ
C S 27	リアルタイム・システム
C S 28	マイクロコンピュータ・システムとローカル・ネットワーク
C S 29	応用組合せ論とグラフ理論
C S 30	計算理論
C S 31	情報システムの設計
C S 32	情報記憶とアクセス
C S 33	分散型データベース
C S 34	パターン認識
C S 35	コンピュータ・グラフィックス
C S 36	モデリングとシミュレーション
C S 37	法的・経済的問題
C S 38	記号・代数的操作入門

表 3 ACM カリキュラム 80
Table 3 ACM curriculum 80
(Comm. ACM Nov. 1982 より)

●情報システム技術	
I S 1	コンピュータの概念およびソフトウェア・システム
I S 2	プログラム、データおよびファイル構造
I S 4	データベース管理システム
I S 6	データ通信システムおよびネットワーク
I S 7	モデリングおよびデシジョン・システム (修士課程のみ)
●組織における情報システムの概念	
I S 3	組織における情報システム
I S 5	情報分析
I S 8	システム設計プロセス
I S 9	情報システムのポリシー (修士課程のみ)
I S 10	情報システム・プロジェクト
●米国経営大学協議会 (AACSB) の共通教科	
a)	企業におけるマーケティング、物流、製造、財務
b)	企業を取りまく経済的・法律的・政治的理論
c)	会計および定量的手法
d)	組織論、人間関係論、統制と機械付けシステム、コミュニケーション等の組織論の知識
e)	不確実な状況での管理統制プロセスの知識、たとえば管理全体における分析とポリシーの決定の統合等

tee on Information Systems) によって改訂が検討され、1982 年 11 月にはカリキュラム 80 として発表された (表 3)。

今回の改訂の狙いは、マネジメントとして必要な技術、たとえばコミュニケーション、文書作成、プレゼンテーション等の技能を組織における情報システムの概念のコースの中に組み入れたことや、米国経営大学協議会 (AACSB, American Assembly of Collegiate School of Business) の設定した共通教科を大幅に取り入れたこと、修士レベルのカリキュラムで「情報システムと企業組織との関係」および「情報システムに関するポリシーの決定」についてのコースを設けたこと等が挙げられる。このほか、最近の分散処理やローカル・ネットワークの登場に即しデータ通信に関する教科が充実された。

また、カリキュラム 80 の学部および修士レベルのコース数は、それぞれ 8 および 10 であり、学部レベルではプログラミングのツール、システム分析、システム設計に力を入れ、プロジェクト管理等の管理ツールやネットワーク設計やコンピュータ援用、システム設計等の上級技術は一般的理解に止どめ、修士レベルほどの深さでは行わないとされて

表 4(a) IEEE コンピュータ部会のコンピュータ・サイエンス/エンジニアリングのモデル・プログラム
Table 4(a) Computer science/Engineering model program of IEEE computer society
(IEEE Computer April 1984 より)

<必須科目>	
●講義	
S A 1	計算の基礎
S A 2	データ構造
S A 3	システム・ソフトウェアとソフトウェア工学
S A 4	コンピュータ言語
S A 5	オペレーティング・システム
S A 6	論理設計
S A 7	ディジタル・システムの設計
S A 8	コンピュータ・アーキテクチャ
S A 9	インタフェースとコミュニケーション
●実習	
S A 10	コンピュータ実習入門
S A 11	ソフトウェア工学実習
S A 12	ディジタル・システム設計実習
S A 13	プロジェクト実習
<上級科目>	
S A 14	ソフトウェア工学
S A 15	ディジタル設計自動化
S A 16	計算理論
S A 17	データベース・システム
S A 18	コンピュータ・アーキテクチャ特論
S A 19	アルゴリズムの設計と解析
S A 20	フォールト・トレラント設計
S A 21	性能予測と解析
S A 22	コンピュータ・グラフィックス
S A 23	VLSI システム設計
S A 24	トランスレータ作成システム
S A 25	コンピュータ・コミュニケーション・ネットワーク
S A 26	システム実習
S A 27	人工知能
S A 28	オペレーティング・システム特論

いる。

4. IEEE モデル・カリキュラム

さて、これまで ACM を中心として、リベラル・アーツ系の学科と経営大学院系の学科のカリキュラムの系譜を紹介してきた。しかし、情報処理関連のカリキュラムには、ハードウェア技術に力点を置いた工学系の系譜があり、米国電気電子学会(IEEE)のコンピュータ部会(Computer Society)が進めているコンピュータ・サイエンス/エンジニアリングのモデル・カリキュラムがその代表である。このモデル・カリキュラムが設定されたのは、1977年で

表 4(b) コンピュータ部会のモデル・プログラムにおける専門選択科目の構成例

Table 4(b) Professional electives by subfield area (IEEE Computer April 1984 より)

年次	ソフトウェア工学	コンピュータ・システム設計	知識ベース・システム
3年	コンパイラとトランスレータ 性能分析入門	アーキテクチャ エレクトロニクス 特論	データベース・システム 計算理論
4年	オペレーティング・システム データベース・システム 計算理論 トランスレータ作成システム アーキテクチャ コンピュータ・コミュニケーション・ネットワーク アルゴリズムの設計と解析	コンピュータ・コミュニケーション・ネットワーク オペレーティング・システム コンパイラとトランスレータ 設計自動化 性能分析入門 VLSI システム設計 フォールト・トレラント設計	オペレーティング・システム 人工知能 アーキテクチャ コンパイラとトランスレータ コンピュータ・コミュニケーション・ネットワーク 分散型データベース グラフィックス

表 4(c) コンピュータ部会のモデル・プログラムにおける上級科目の例 (SA 27 人工知能)

Table 4(c) Subject area of advanced course (IEEE Computer April 1984 より)

モジュール	トピックス	目的	内容
1	問題解決入門	人工知能の一般的主題の紹介	・学習、形状の類似、単純な理解、問題解決、積分プログラム、ロボット工学、エキスパート・システムの例
2	表現論入門	表現に関する基本的な概念の導入	・手順的表現 (コンピュータ・プログラム) プログラミング・システム ・非手順的表現 一階述語計算 表明の集合 意味ネットワーク ・記述用ツール 差異 ・拘束 記述準拠の例 拘束準拠による文解析 ・プロトタイプ、スクリプト、スキーマ
3	制御戦略	制御の原理的な概念の導入	・手段目標分析あるいは一般問題解決器 (General Problem Solver) 問題解決器 (Solver) 問題の分割 ・拘束の伝播 (Huffman-Cloves のラベリング) ・プロダクション・システム シミュレーション・アクション・ルール データ、操作と制御 分割あるいは分割化サブシステム ・ルール準拠演繹システム

4	探索戦略	探索戦略の記述とその重要性の例示	・探索木と全数探索 ・バックトラッキング手法 ・山登り法 ・グラフ探索法 ・ヒューリスティック探索法 ・効率問題 ・双方向および単方向探索 AND/OR グラフ
5	述語計算とルール準拠演繹	人工知能分野における一階述語計算とその応用の説明	・構文、概念、論理式、論理記号、量化記号 ・定理と証明 ・リゾリューション ・リゾリューションによる背理 ・プロダクション・システム ・ルール準拠演繹システム ・構造的対象表現 ・Prolog
6	目標指向計画	目標指向計画の戦略の説明とその例示	・ロボット問題解決 世界の認識、計画の形成 実行の監視 状態記述と目標記述 STRIPS 演繹システム (Green と Kowalski の形式化) 階層的計画 条件の階層化 詳細化の遅延 低位の記述による高位計画の修正
7	理解の応用	画像解析と言語における理解の概念の例示	・画像解析 パターン認識 状態記述、基本スケッチ コンピュータ・ビジョン 拘束 ・言語理解 談話の領域の限定 概念依存 名詞グループ 遷移ネットワーク 質問とコマンドの処理
8	知識の表現 (フレームと、スクリプト)	フレームの吟味 拡張遷移ネットワーク (ATN) と文法の解析についての例示	・ネットワークとフレーム ノード、関係、ターミナル、マーカー ・必要条件、視点、変換 デフォルト、例外、情報検索 ・拡張遷移ネットワーク 格文法 スクリプト
9	人工知能プログラミング言語とデータベース	問題の見方の一例として適切なプログラミング言語の重要性を示し、LISP あるいは Prolog を導入し、データベースおよびデモンの概念を紹介	・プログラミングと人工知能との関係 ・記号とその操作の問題 ・人工知能プログラミング言語 ・専用言語の必要性 LISP と Prolog 例 (ATN 用の LISP プログラム) ・パターン指向データベース 連想データベース デモンあるいはモニタの応用
10	知識ベース・システム	知識ベース・システムのように大量の知識を必要とするアプリケーションにおける前述のモジュールの概念を統合し、それと対照的に上手な計算手順と比較する。知識ベース・システムのケース・スタディを行う。	・知識表現の概念を評価 関係する知識 (どれが、どれだけ、どの分野で必要か) ・単純なタートル・プログラムの評価 データ、知識ベース、制御 状態、入力、副目標、フィードバック、アクション、再帰的実行、デバックキング ・知識獲得 ・知識ベース・システム (ケース・スタディ) DENDRAL と Meta-DENDRAL HEARSAY-II 診断システム MYCIN, Prospector. ・知識転移システム (TEIRESIAS) ・自動プログラミング・システム (PECOS)
11	ロボット工学システム	ロボット・システムの開発におけるコンピュータ・サイエンスとエンジニアリングおよび人工知能の応用	・ロボットのアーム ・ロボットのビジョン ・自動製造 ・ロボット・システム用言語 ・遠隔操作システム

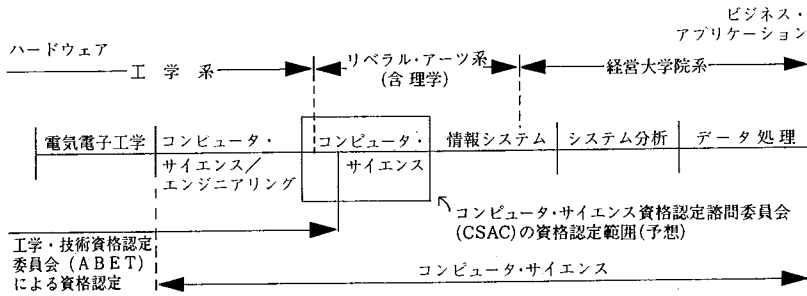


図1 コンピュータ・サイエンスの範囲 (Comm. ACM April 1984 より)
 Fig. 1 Scope of computer science

あったが、本年4月にその改訂版が発表された(表4-a)。

このモデル・カリキュラムは最新のもので、最近のコンピュータ技術を反映しており興味深いので、専門選択科目の構成例(表4-b)と上級科目の例として人工知能(表4-c)を紹介しておく。

このモデル・カリキュラムは、必須科目と選択科目からなり、必須科目は講義と実習で構成され、講義で学習したことを直ちに実習で試み自ら体得できるように工夫されている。「実際に物を作れること」に重点が置かれているのが、このカリキュラムの特徴である。

なお、このカリキュラムは1977年のカリキュラムと比較すると、マイクロプログラミングが独立した分野ではなくCPUの制御装置の設計の分野に吸収されてしまったこと、新分野としてソフトウェア工学、フォールト・トレラント設計、性能評価、グラフィックス、VLSI設計、人工知能が登場したことなどが変わっている。

5. 米国情報処理教育

これまで情報処理関連の3種の代表的なカリキュラムの系譜を年代を追って述べてきたが、これらと関係付けると図1のようになる。

なかでも、混乱しやすいのは、リベラル・アーツ系(理学も含む)のコンピュータ・サイエンス学科と、工学系のコンピュータ・サイエンス/エンジニアリング学科である。前者は、より広い、人文および科学の教育を基盤としているのに対して、後者は学生を技術者として育成するよう機械工学や電子工学等の他の工科系の学科も学習するように考えられている。

また、広義のコンピュータ・サイエンスとしては、情報システム、システム分析、データ処理、医療システム、図書館学等もその対象として考えられよう。

6. 資格認定のガイド・ライン

近年、情報処理関連諸学科の卒業生の品質保証を求める産業界からの要請がとみに大きい。このため、ACMとIEEEのコンピュータ部会は、コンピュータ・サイエンスのプログラム(授業計画)の要件と資格認定のガイドラインを検討する共同プロジェクト(Joint Task Force on Computer Science Program Accreditation)を1982年頃に発足させた。

そして、同プロジェクトは、コンピュータ・サイエンス・カリキュラムの基準の草案(Draft Criteria Computer Science Curriculum)と呼ばれる中間報告を本年4月に発表している。従来工学系の大学では工学系学科の資格認定機関である工学・技術資格認定委員会(ABET, Accreditation Board for Engineering and Technology)によって学科の資格認定が行われてきたが、他の学部では行われてはいなかった。同報告は、コンピュータ・サイエンス資格認定諮問委員会(CSAC, Computer Science Accreditation Commission)と呼ばれ、コンピュータ関連のすべての学科の資格認定を行う機関を設立することを提案している。

このほか、カリキュラムのほか、実習用機材およびコンピュータ施設、コンピュータ・サイエンス・プログラムの要件、教授陣の要件、学生の要件、支援体制(管理体制・図書室等の完備)等についても述べられている。

なお、この報告に対する公聴会等のデータを基に同プロジェクトはCSACの設立の準備に入っており、1965年の春には資格認定を開始すると伝えられる。

参考文献

[1] Curriculum Committee on Computer Science, "Curriculum '68, Recommendations for Academic Programs in Computer Science",

Comm. ACM, Vol. 11, No. 3, March 1968.

[2] R. H. Austing, "Curriculum '78, Recommendations for the Undergraduate Program in Computer Science—A Report of the ACM Curriculum Committee on Computer Science", *Comm. ACM*, Vol. 22, No. 3, March 1979.

[3] K. I. Magel, "Recommendations for Master's Level Programs in Computer Science—A Report of the ACM Curriculum Committee on Computer Science", *Comm. ACM*, Vol. 24, No. 3, March 1981.

[4] Education Committee of IEEE Computer Society, "Model Curricula in Computer Science and Engineering", IEEE Pub. EH0 199-8, Committee Report, Jan. 1977.

[5] J. T. Chain, "The IEEE Computer Society Model Program in Computer Science and Engineering", *IEEE Computer*, Vol. 17, No. 4, April 1984.

[6] R. Ashenurst, "Curriculum Recommendations for Graduate Professional Programs in Information Systems", *Comm. ACM*, Vol. 15, No. 5, May 1972.

[7] J. D. Couger, "Curriculum Recommendations for Undergraduate Programs in Information Systems", *Comm. ACM*, Vol. 16, No. 12, Dec. 1973.

[8] J. F. Nunamaker, "Information Systems Curriculum Recommendations for the 80s: Undergraduate and Graduate Programs—A Report of the ACM Curriculum Committee on Information Systems", *Comm. ACM*, Vol. 25, No. 11, November 1982.

[9] 坂井利之, "情報工学の教育・研究", bit 臨時増刊, 共立出版, 1980年12月.

[10] M. C. Mulder, "Computer Science Program Requirements and Accreditation—An Interim Report of the ACM/IEEE Computer Society Joint Task Force", *Comm. ACM*, Vol. 27, No. 4, April 1984.

(教育部 教育センター)

言語 Occam の概要
Occam Language

若鳥 陸夫

1. はじめに

Occam の名称は 14 世紀の英国の哲学者 William Ockham (Occam とも言う, 1284-1350 年) にちなんで命名されている。哲学者 Occam は「オッカムのかみそり」という格言を残しているが, その意味するところは「存在は必然性 (必要) なしに増加されてはならない^[7]。(A plurality is not to be posited without necessity^[11].)」ということにある。Occam も, 格言のように可能な限度で単純化するという C. A. R. Hoare の Communicating

Sequential Processes^[2] の考え方を汲んで英国の半導体メーカ INMOS 社によって開発された。筆者らは UDM 3800^[3] の上で, その実験言語^[4] を使った経験をもとに, 言語 Occam の概要を紹介する。

2. 言語 Occam の源流

言語 Occam の開発の直接の源である Communicating Sequential Processes (以下 CPS と略記する) の考え方から述べる。

2.1 プロセス間の同期問題の解決手段

複数のプロセス間を並行処理する際, それらのプロセス間を同期させる必要がある。その具現化手法には, 古典的な Busy-Waiting から, 各プロセスに共通記憶域を設け, その記憶域に書き込みを行うことによって複数のプロセス間の同期をとる方法 (Shared Variables) や, E. W. Dijkstra のセマフォ (semaphore) による方法以降, 拡張 PL/I のイベント, コンカレント Pascal で実現した monitors and queues, Campbell および Habermann の順路式等がある。

しかしながら, これらの手法はいわば手続き指向 (procedure oriented) の定式化をとっている。一方, C. A. R. Hoare はプロセス間の同期をメッセージ指向 (message oriented) な方法で定式化した (図 1)^{[2], [8]}。

各プロセス・メッセージの交流のために図 2 のように入力チャンネルと出力チャンネルを通して考える。各プロセスの結果は出力チャンネルを通して出力される。プロセス間の同期はチャンネルを通じてメッセージを受け渡すことで行われる。ただし, Occam 処理系は, 純 CSP の考えの他に, Pascal 流の手続き呼び出し手法の併用等, CSP にはない機能も含まれている。

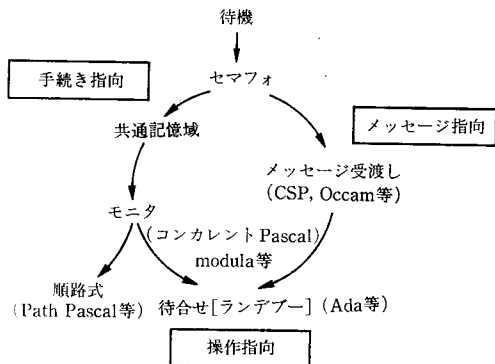
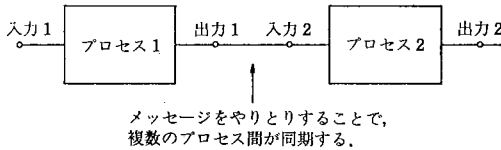


図 1 同期方式とコンカレント・プログラム言語の分類^[8]
Fig. 1 Synchronization techniques and language classes



メッセージをやりとりすることで、
複数のプロセス間が同期する。
図 2 CSP のプロセス間同期方法
Fig. 2 Process synchronization of CSP

従来の手続き指向の同期方式の一つセマフォのように鉄道の単線区間における腕木信号機による資源の相互利用をする等の制御ではなく、メッセージ指向の同期方式をとる言語 Occam を使って制御プログラム等を開発する人は細かい仕掛けというより、より本質的な並行処理時のふるまいに神経を集中させることができる。

3. 言語 Occam の構文とその特徴

拡張 BNF による言語 Occam の構文を図 3 に示す。プログラムは一つのプロセスであり、プロセスは「単純な操作」、「構造化プロセス」、「宣言プロセス」などから構成される。単純操作とは「代入」、「入力」、「出力」、「待機 (WAIT)」、「無操作 (SKIP)」の 5 種類である。基本的な構造化プロセスは「逐次 (SEQ) プロセス」、「並行 (PAR) プロセス」、「代替 (ALT) プロセス」、「選択 (IF) プロセス」、「反復 (WHILE) プロセス」からなる。

また、SEQ や ALT の直後の字下げは関係する範囲を限定する意味をもち、言語 Pascal の begin-end や言語 C の { } に相当する。

その他、言語 Occam の複製 (Replicater) 機能は

```

<プログラム> ::= <プロセス>
<プロセス> ::= <単純操作> | <構造化プロセス> |
  <宣言> : <プロセス> | <定義済プロセス呼出>
<単純操作> ::= <代入> | <入力> | <出力> | <待機> | <無操作>
<構造化プロセス> ::= <逐次プロセス> | <並行プロセス> |
  <代替プロセス> | <選択プロセス> | <反復プロセス>
<逐次プロセス> ::= SEQ { <プロセス> } | SEQ { 複製 } <プロセス>
<並行プロセス> ::= PAR { <プロセス> } | PAR { 複製 } <プロセス>
<代替プロセス> ::= [ PRI ] ALT { <ガード付プロセス> } |
  [ PRI ] ALT { 複製 } <ガード付プロセス>
<選択プロセス> ::= IF { <条件プロセス> } | IF { 複製 } <条件プロセス>
<条件プロセス> ::= <式> <プロセス> | <選択プロセス>
<複製> ::= <名前> = [ <始値> ] FOR <終値>
<代入> ::= <変数> = <式>
<入力> ::= <入力チャネル名> ? <変数> { ; <変数> } |
  <入力チャネル名> ? ANY
<出力> ::= <出力チャネル名> ! <式> { ; <式> } |
  <出力チャネル名> ! ANY
<ガード付プロセス> ::= <ガード> <プロセス> | <代替プロセス>
<ガード> ::= [ <式> & ] <入力> [ <式> & ] <待機> |
  [ <式> & ] <無操作>
<待機> ::= WAIT <式>
<無操作> ::= SKIP
    
```

図 3 言語 Occam の構文 (抜萃、用語は意味類似のものを使用)
Fig. 3 Syntax of Occam language

特筆に値いする。それは Pascal の「教え上げ型繰り返し返し」に似た記述でありながら、コンパイルされた後の目的コードでは、プロセスが必要な回数だけ複製されることである (図 4)。

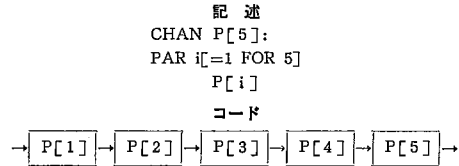


図 4 プロセスの複製
Fig. 4 Replication of process

プロセスの複製は、プロセスをパイプライン化してスループットを向上できるケース (複数 CPU) に有用となる。

4. 見本プログラム

4.1 哲学者の食事問題

【課題】 5 人の哲学者が自由に思索したり食事したりしているとする。食堂は一つで、そこに円卓があり、5 人分の椅子の左手に対応するところにフォークが各 1 本置かれている。円卓の中央にはスパゲッティが大皿に盛られている。しかしスパゲッティ

```

-- DINING PHILOSOPHERS
-- BY R.WAKATORI ON 84/8/23
-- (PROBLEM DUE TO E.W.DIJKSTRA)
--
DEF N=5;
VAR TAKE[N], PICKUP[N]; PUTDOWN[N],
  LEAVE[N];
CHAN PHIL[N], FORK[N], SEAT[N];

PROC PHILS(VALUE I) =
  SEQ
  -- THINK
  SEAT[I] ! TAKE[I]
  FORK[I] ! PICKUP[I]
  FORK[(I+1) $ N] ! PICKUP[(I+1) $ N]
  -- EAT
  FORK[I] ? PUTDOWN[I]
  FORK[(I+1) $ N] ? PUTDOWN[(I+1) $ N]
  SEAT[I] ? LEAVE[I];

PROC FORKS(VALUE I) =
  PAR
  PHIL[I] ? PICKUP[I]
  PHIL[I] ! PUTDOWN[I];

PROC SEATS =
  VAR OCCUPANCY, I, J;
  SEQ
  OCCUPANCY := 0
  PAR
  PAR I = [0 FOR N-1]
  SEQ
  PHIL[I] ? TAKE[I]
  OCCUPANCY := OCCUPANCY + 1
  PAR J = [0 FOR N-1]
  SEQ
  PHIL[J] ! LEAVE[J]
  OCCUPANCY := OCCUPANCY - 1;
    
```

図 5 プログラム見本 1 (\$ は剰余演算子)
Fig. 5 Program example 1

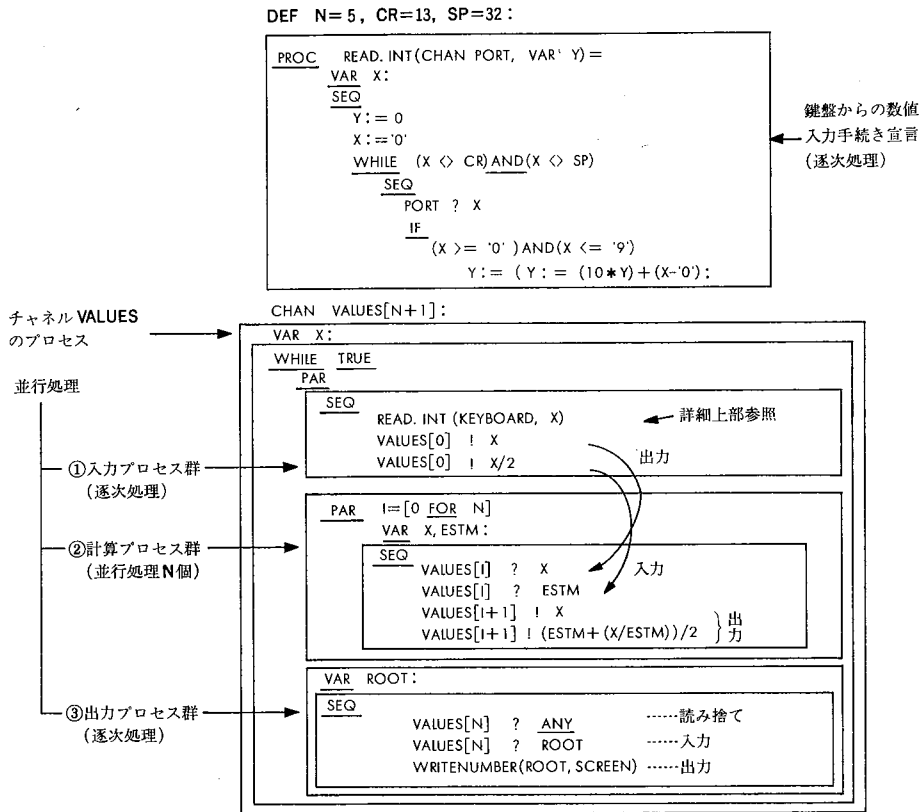


図6 プログラム見本2 (下線は予約語を示す)
Fig. 6 Program example 2

が巻き付いているので哲学者はそれを食べるのに左右2本のフォークを必要とする。食べ終わった哲学者はもとの位置にフォークを置き、食堂から出ていく。哲学者達が最も短時間で食べられ、思索の時間を1分でも多くとれるようにせよ^[2]。

哲学者 (phil [i]), フォーク (fork [i]), 食堂 (room) を並行処理させるプログラムを言語 Occam で表すと図5のようになる。

4.2 並列性のないプログラム

Occam Evaluation Kit を使い、Newton Raphson 法により平方根を求めるプログラム例を図6に示す。

この見本プログラムは文献^[5,6]に掲載されていたものに筆者らが数値入力手続きや数値出力手続き(見本では省略)を付加し、実際の処理系で動くように改良してみたものである。

図6のチャンネルの各プロセス Values は三つの並行処理プロセス群から構成されている。すなわち、

- ① 入力プロセス群
- ② 計算プロセス群
- ③ 出力プロセス群

の三つのプロセス群が並列に無限回実行される。

言語 Occam では字下げはブロックの区間を意味するので、SEQ や PAR の下位プロセスがその最下行まで実行するとその部分から脱出する。これらの三つのプロセス群は非同期で勝手に動いているのではなく、同一チャンネルが相互プロセス間に使われており、そのチャンネルの入出力間で同期している。

5. 今後の発展

言語 Occam は単純明解で、かつ強力な並行プロセス記述能力を具現化している。この言語はどちらかという、システム記述に適しているので、並行処理プログラム言語の記述や、組込み型コンピュータのプログラミング、並行処理のシミュレーションなどには新しい環境を提供してくれよう。現在米国国防総省 (DOD) の音頭で Ada 言語がデータ抽象化やプロセス間の同期等を組み込み、次世代 (1970年代) の言語といわれているが、応用面や使い手によっては言語 Occam の方が適していることもある。

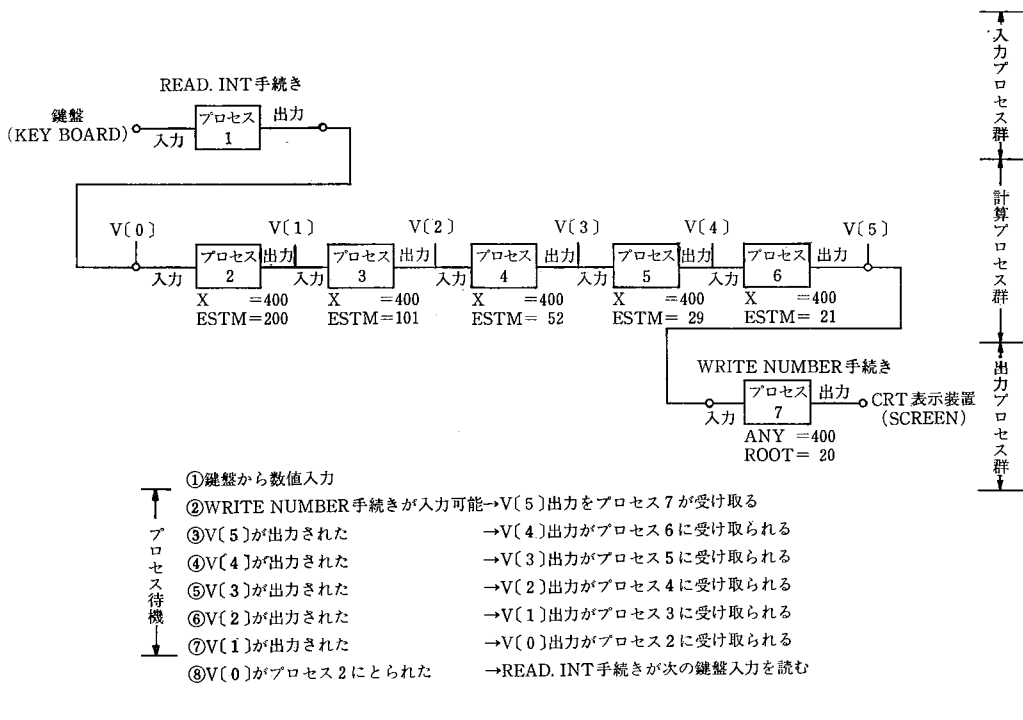
この Occam の評価キット^[4]はオペレーティングシステムとして UCSD-P システムを塔載している日本ユニバック (株) の UDM 3800 (マイクロプロセッサ8086+ハードディスク付コンピュータ) でも動

3 プロセス群の間の同期の様態を下図に示す。いま鍵盤から整数を入力する手続き READ. INT が動いて変数 X に数値が代入されると、チャンネル VALUES[0] にその値が出力待ちになる。チャンネル VALUES[0] の入力はプロセス 7 がチャンネル VALUES[5] から入力を受け取り準備を完了してから、その入力許可がプロセス 6, 5, 4, 3, 2, 1 と逆順に通じ、それに誘われてプロセス 1 の出力（つまりチャンネル VALUES[0]）の値がプロセス 7 まで転送される。

プロセス 1 からプロセス 7 まで、変数 X と X/2 の一組の値が転送される間に、予測値（変数 ESTM）の値が 200→101→52→29→21→20 というように X の平方根へ逐次的に近似する。

プロセス 7 の入力チャンネルに VALUES[5] の出力が届いてから CRT 表示装置（組込チャンネル SCREEN）へ出力すれば、結果を表示するという仕掛けである。

このように言語 Occam は並行処理したいプロセスを PAR を使って簡潔に表現でき、並行処理したいプロセス間の同期は入出力チャンネルを通じて行われる。



かすことができる。また、この言語処理系では構造化エディタを採用しており、文法検査と修正が交互に行えるようになっている。

参考文献

[1] Crowell-Collier Educational Corp. Collier's Encyclopedia, Vol. 23, pp. 495, 1970.
 [2] C.A.R. Hoare, "Communicating Sequential Processes", *Communications of the ACM*, Vol. 21, No. 8, 1978.
 [3] UDM 3000 シリーズ漢字, 日本ユニバック(株): UDM PASCAL 操作解説書, 資料コード 481735403.
 [4] INMOS Ltd., Occam Evaluation Kit, 1983.

[5] INMOS Ltd., Occam Programming Manual
 [6] INMOS, Ltd., Occam プログラミングマニュアル, 啓学社, 1984.
 [7] "哲学事典" 平凡社, pp. 163-164, 1954.
 [8] G.R. Andrews, F.B. Schneider, "Concepts and Notations for Concurrent Programming", *Computing Surveys*, Vol. 15, No. 1, March 1983.

(技術企画部 技術調査室)

無人化運転システム
Unattended Operation System
— 今 西 秀 文 —

1. はじめに

中部電力の長野・静岡支店では、昭和54年のオンライン開始以来、分散処理を指向し、現在では図1のような設備状況となっている。そこで、分散処理に伴う運用の複雑化や要員数の増加を抑えるため、夜間の無人運転を可能にする自動化システムの開発を行った。一般的に、事務処理を中心とした汎用コンピュータの分野における自動化/無人化は低く、省力化のレベルにとどまっていることが多いのに対し、同社の自動化システムは、システムの始動・停止の自動化や種々の監視機能を含む本格的なシステムである。

2. 自動化の範囲と運転形態

自動化システムは図2のような設備やソフトウェアにより構成されている。これにより、①電源投入からユーザ初期化処理までの自動化、②業務処理の自動化、③業務終了の自動判定と自動停止および自動電源切断、④設備異常、ハード/ソフト異常、業務処理異常の検出と通報、⑤遠隔地からの監視・検索・制御、等の機能が実現されている。また、図3のように、自動化のむずかしい印書作業や磁気テープ操作等の作業を昼間時間帯に集約させることにより、夜間のオペレータを廃止することに成功している。

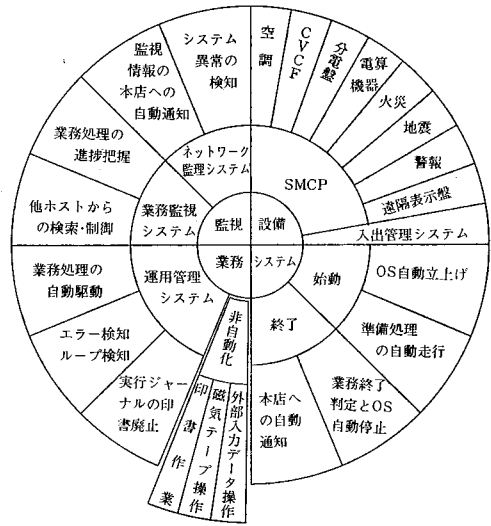


図2 自動運転システム体系図
Fig. 2 System diagram on unattended operations

3. 始動および停止の自動化

コンピュータおよび空調機、CVCF (Constant-Voltage Constant-Frequency)等の付帯設備の電源投入からオペレーティング・システムのブートまでの処理はSMCP (System Monitoring Control Processor)によって自動化され、オペレータの出勤時にはブート完了の状態となっている。その後の制御ソフトウェア (Communications Management System: CMS, Data Management System: DMS等)のセットアップやアプリケーションの初期化処理も自動化されている(図4)。これら初期化ランでは、システムの立上り状態(ブートの種類、通常時か障害後かの区分等)に応じて適切な処理を選択

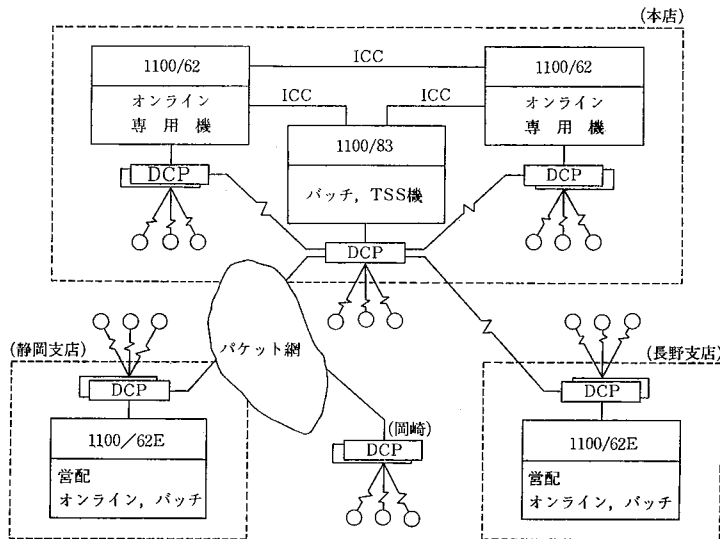


図1 中部電力のコンピュータ・ネットワーク
Fig. 1 Computer network in Chubu Electric Power Company, Inc.

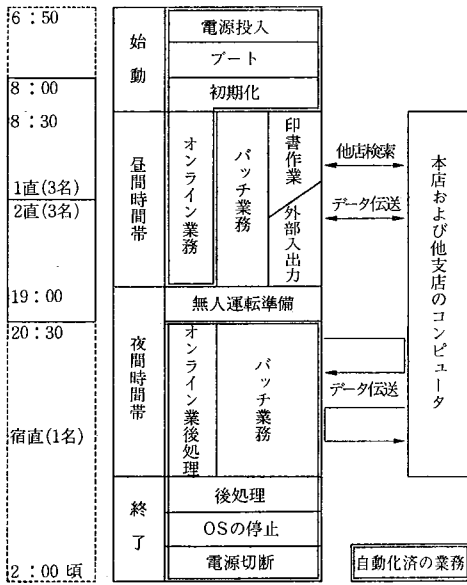


図3 自動化後の運転形態
Fig. 3 System's operation after introductions of automatization

するとともに、UOSS (Unattended Operation Support Software) を利用して他の初期化ランとの同期をとりながら実行される。

業務終了の判定から終了処理、システム停止・電源切断の過程も図5のように自動化されている。これらの一連の処理は運用管理システムの下に制御されるため、通常の業務処理ランと同様に、事前のスケジューリングやランの自動起動およびシステム障害時の自動再起動等の機能も合わせて実現されている。

4. 業務処理の自動化

業務処理システムは、あらかじめ作成されている実行スケジュールにもとづき、運用管理システムにより自動起動がなされる。各ランは、①ラン間でのファイルの作成・参照関係、②OCR データ、伝送ファイル等の外部データの到着条件、③開始指定時刻、等の起動条件を守りながら、人手を介さずに順次処理される (図6)。

ランの実行結果は印書を行わず、必要な時に業務

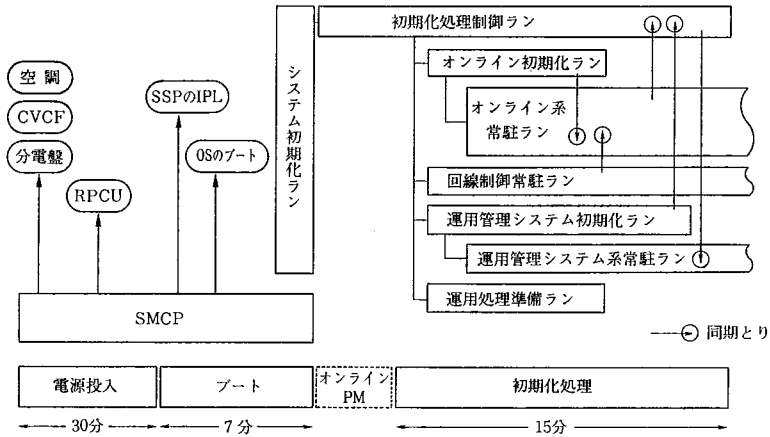


図4 始動
Fig. 4 Start-up operations

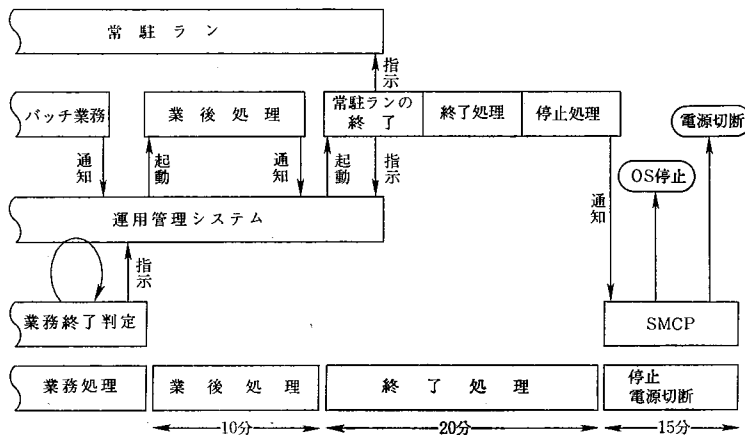


図5 終了
Fig. 5 Finish operations

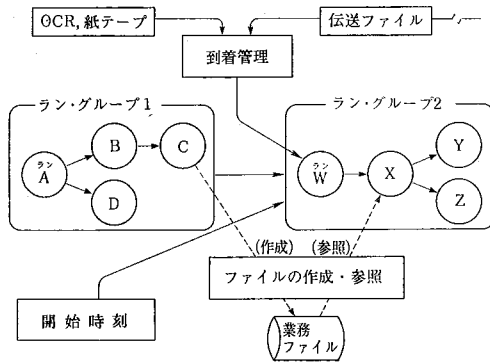


図6 自動運転
Fig. 6 Unattended operations

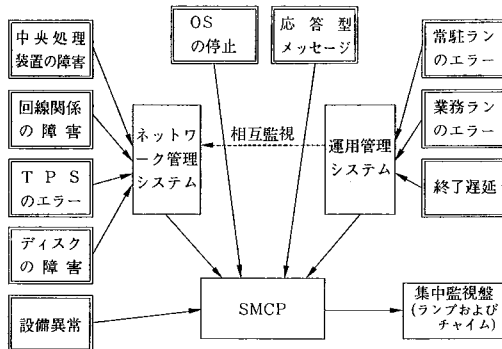


図7 異常の検知と通報
Fig. 7 Detection of abnormal condition and alarming

監視システムを使用して画面検索する方式が採用されている。他に、磁気テープ・ファイルをディスク化するために、ディスク・ファイルで世代管理を行う機能が開発されている。

5. 監視

夜間に需要家に直結した営業料金業務を実施しているため、無人運転下にあっても、障害の発生を速やかに認識して復旧に当たることが必要となる。このため、種々の監視機能を用意し、宿直室に設置されている SMCP 遠隔監視盤で警告を行うように

している。監視対象としては SMCP による設備監視の他に、ネットワーク管理および運用管理システムにより、①CPU、メモリ、ディスク等の障害、②DCP (Distributed Communications Processor) 回線/端末障害、③業務処理ランや TPS (Transaction Processing System) のエラー、処理遅れ、等の監視が行われる (図7)。

また、本支店間の連絡のために業務監視システムを開発し、本店側から支店の処理状況を検索したり、運用管理システムへの指示やディスク障害時の復旧作業の制御等を可能にしている。

6. 効果

自動化システムの適用により3シフトのオペレータ体制を2シフトに変更し、支店オペレータの三分の一削減に成功している。また、以上に述べてきた機能は昼間帯の省力運転にも結びついており—自動化前に比べ、磁気テープの操作回数が57パーセントに、コンソールの操作回数が54パーセントに減少—信頼性の向上やオペレータのモラルの向上に役立っている。

7. おわりに

中部電力では、分散化の進展に合わせて運転業務の合理化対策として、次の3段階の計画で自動化を推進中である。

- ① 運用管理システムを中心とする運転計画の機械作成および運転省力化 (昭和56~57年度)
 - ② 支店コンピュータの夜間無人運転 (昭和58~59年度)
 - ③ 本店コンピュータの自動運転 (昭和60年度)
- 現在は第3段階のシステム開発に入っており、ここでは複数コンピュータの集中監視・制御、磁気テープの自動装填等が実現される予定である。

(名古屋支店 システム部)

—Alan Bundy 著

**“The Computer Modelling
of Mathematical Reasoning”**

Academic Press, XIV+322 pp., 1983.

自動定理証明法 (automated theorem proving), とくに, resolution をベースとする自動定理証明法を, いわゆる人工知能の問題に応用するやさしい入門書としては, 現在,

- ① R. Kowalski, Logic for Problem Solving, North-Holland, 1979.
- ② N. J. Nilsson, Principles of Artificial Intelligence, Tioga Publishing, 1980.
- ③ L. Wos, R. Overbeek, E. Lusk, and J. Boyle, Automated Reasoning, Introduction and Applications, Prentice-Hall, 1984.

および, 本書があげられよう。各著書はそれぞれ特徴のある良い入門書であるが, Bundyによる本書と①のKowalskiにはresolutionをベースとする各種のuniform proof proceduresの示すcombinatorial explosionに対する省察を含んでおり, いわば, “automated theorem proving: next 25 years”に対する有益な観察と視点を提供している。本書は, 前書きで述べているように, とくにこの点を意識的に問題としている。

③のL. Wosらも AURA の成功を踏まえてはいるが, 定理証明法の現在の問題意識を手短かに理解し, 現在有効なアプローチを一覧するという目的には, じつは Bundyによる本書が最も優れている。本書と①のKowalskiの9章, 12, 13章を併読されれば, 現在の問題意識のoverviewが得られよう。

しかし本書の進め方は, かならずしも良い方法であるとはいえない。じつは, 本書で最も良く書かれているのは, 前書きと各章末のSummaryであり, 本書を読まれる時はそれらをよく読む必要がある。本書では, とくに7章および第3部 (Part III)以降を読むように勧めたい。

さて, 全体は1章と5部および18章に分けられている。

1章は定理証明の概念と歴史を述べている。

第1部のタイトルは形式的記法であり, 2章から4章までは, 数理論理入門である。2章では命題論理を, 3章では一階の述語論理を, 4章では高階の論理や, 群論, 算術の形式系を述べている。

第2部のタイトルは uniform proof procedures

であり, 5章から7章までで resolution をベースとする定理証明法入門である。5章では resolution, factoring および paramodulation の規則を, 6章では探索法を, 7章では uniform proof procedures への批判と新しいアプローチを良く述べている。

第3部から本書の主題にはいる。とくにこの第3部の内容は本書の中心をなし, いまのところ論文は多く存在するが, このような入門書には書かれたことのない内容となっている。第3部のタイトルは “Guiding Search” であり, 8章から12章までで M. Rabin^[1] が指摘した定理証明で有効な “人間的”, つまり意味論的ないしは局所的方法を述べている。8章では Bledsoe, Shostak らによる Sup-Inf 法を, 9章では term rewriting rules を, 10章では意味論的情報の利用を述べている。9章は成書にはない内容であり, よくまとまっている。10章は①のKowalskiの9章 “Global Problem-Solving Strategy” と互いに補完しあう内容であり, 併読するとよい。11章では simplification や, induction, generalization 等, 主に Boyer-Moore の定理証明系^[2]の成果を述べている。12章では方程式の解法等を例にとってメタ・レベル論理を述べており, 応用では重要な章である。

さて, 第4部では, 本書のタイトルにある “数学における推論 (Mathematical Reasoning)” をとりあげる。4部のタイトルは “Mathematical Invention” であり, 13章から14章までである。13章では数学的概念の自動形式の問題を Lenat の AM を例にとって紹介している。15章では Bundy らの MECHO 等を例にとって数学的モデルの自動形成の問題を紹介している。15章はメタ・レベル知識の利用という点で応用上重要である。

さて, 第5部は4部までで手を抜いた諸概念の定義をまとめて述べている。第5部のタイトルは “Technical Issues” であり, 15章から17章までである。15章では線形を, 16章では Herbrand の定理から resolution の妥当性と完全性までを, 17章ではパターン・マッチングを述べている。9章は成書にはない内容と言ったが, term rewriting で用いる unification とくに λ 算法の unification も成書にはない内容であり, 一読の価値がある。最後の18章では応用を述べている。また巻末には, DEC システム 10 Prolog で書いた Artificial Mathematicians というプログラムが付けられている。内容は節の長さを評価関数とする縦形の線形 resolution theorem prover であり, 10章で述べられた semantic

checking がはいっている。ただし occur check は設けていない。

resolution に限らず、一般の定理証明系の推論規則は計算規則としては非決定性規則であり、Fisher-Rabin の Presburger 算術での intractability の証明からも推論できるように一般的な算法としての推論規則は難度 NP (NP hard) である。したがって、L. Wos ら^[4]や J. Minker ら^[5]の実験を待つまでもなく、M. Rabin の述べる理論的障壁にはばまれることになる。

そこで、定理証明系の方法論としては、人間が interactive に意味論的な理論の局所化を行い、たとえば、Edinburgh 大学の LCF^[6]や Stanford 大学の Weyhrauch や Ketonen の証明系^[7]のように、定理を term rewriting によって規則として加え、局所理論の環境内で rewrite rules と推論規則を interactive に適用する方法が考えられる。

しかし、rewrite rules にも発散する性質があり、まだ十分研究されているわけではない。ある意味では、現在の定理証明系の応用は、M. Rabin の述べた障壁を回避するアプローチを実現しようとする緒についたところである。

先に、「本書が最も優れている」と述べた理由はこのような定理証明系の応用の現状を考えたからである。

参考文献

- [1] M.O. Rabin, "Theoretical Impediments to Artificial Intelligence", Information Processing 74, North-Holland, 1974.
- [2] R.S. Boyer and J.S. Moore, A Computational Logic, Academic Press, 1979.
- [3] M.J. Fischer and M.O. Rabin, "Super-Exponential Complexity of Presburger Arithmetic", Complexity of Computation, SIAM-AMS Proceeding Vol. VII, American Mathematical Society, 1967.
- [4] J.D. McCharen, R. A. Overbeek, and L. A. Wos, "Problems and Experiments for and with Automated Theorem-Proving Programs", *IEEE Transactions on Computers*, Vol. C-25, No. 8, 1976.
- [5] G. A. Wilson and J. Minker, "Resolution, Refinements, and Search Strategies: A Comparative Study", *IEEE Transactions on Computers*, Vol. C-25, No. 8, 1976.
- [6] M.J. Gordon, A.J. Milner, and C.P. Wadsworth, "Edinburgh LCF", Lecture Notes in Computer Science 78, Springer, 1979.
- [7] J. Ketonen and J.S. Weening, EKL-An Interactive Proof Checker, Department of Computer Science, Stanford University, 1983.

—山田 真市 著—

“情報処理の科学”

朝倉書店, A 5 判, XII+392 pp., 1984 年, 5800 円

「役立つ情報処理の科学と技術を真に創造しようとするならば、現存のコンピュータや情報処理技術のあるがままに受け入れるだけに留まってはならない。まず目前の応用技術の詳細から一歩しりぞいて、情報処理のいろいろな基本概念や理論の意味内容とその発展の系譜を振り返って、それぞれのアイデアを再吟味してみる必要がある。」これが著者の問題意識であり、執筆の動機である。

さて、本書は次の 4 部と付録 1 章とから構成されている。

- 第 1 部 計算機械の科学
- 第 2 部 命令型プログラムと計算量の科学
- 第 3 部 関数型プログラムとプログラムの信頼性の科学
- 第 4 部 プログラムの意味論
- 付 録 人工知能

第 1 部では、「計算機械とは何か」という問題を探究している。まず 1 章では、有限状態オートマトンを対象とし、その原理的計算能力を調べる、また正則文法や正則表現の概念を述べる。2 章では、Turing の計算機の論理によって計算の概念をとりあげる。3 章では、Neumann 型計算機構の原型である Turing の万能計算機のみカニズムを考察し、また、どのようなコンピュータでも原理的に解けない問題の存在をとりあげている。さらに、計算論の基礎的事柄を述べる。4 章では、Von Neumann によるオートマトンの自己増殖のみカニズムを考え、計算論を用いてオートマトンと自己増殖が可能であることを述べている。5 章では、定性的にある複雑さを超える理論(Gödel 理論)の定理証明オートマトンが系統的に進化しうることを証明している。

第 2 部では、Von Neumann 型の計算機における計算と、命令型プログラム系を考え、情報処理を行う時に必要な計算の手間を定量的に探究している。まず 6 章では、命令型プログラムの原理的構造を追究している。7 章では、計算の手間を考えた実際の計算可能性を調べ、原理的な計算可能性の概念と対比する。8 章では、情報処理の手間を定量的に研究する方法を述べている。9 章では、応用上重

MEMORANDUM

〈Sperry 関係の論文・講演等の要約〉

●ローカル・コンピュータ・ネットワークの高速化のためにエグゼクティブ・ソフトウェアのハード化を採用——高性能のローカル・コンピュータ・ネットワークでは数千メッセージ/秒の処理速度が必要であり、これを実現するためには既存のコンピュータおよび OS では無理であることが知られている。その理由として、インタラプト処理、メッセージのベクタリングとその結果としてのタスクのスケジューリング等、これらをすべてソフトウェアで行うためにはオーバーヘッドが大きいことがあげられる。実際、ソフトウェアのエグゼクティブを使用した典型的な国防システム用の 16 ビットのコンピュータでは、1 メッセージ当たりの処理時間は 500 μ 秒であり、秒当たり 2,000 件のメッセージしか処理できないが、ハードウェア・エグゼクティブを使用したコンピュータでは 8 メガ・ビットの転送速度で 32 語のメッセージの場合に 15,000 件/秒の処理が理論的に可能となる。

今回開発されたハードウェア・エグゼクティブは、SCU (System Control Unit) と呼ばれるマイクロプログラム・コントローラ・ボードで実現されている。この SCU は、プロセッサを特定のプロセスに割り当てたり、非同期操作を含むすべてのイベント管理等のプロセス間のコンカレンシの管理等を行うとともに、プロセス制御・コミュニケーション・同期に関する基本命令のプログラマへのインタフェースを供給するものである。

このシステムの利点としては、①高速なメッセージ処理 (とくにローカル・エリア・ネットワークのような多数の比較的短いメッセージに有効である)、②内部環境のセキュリティ向上 (多重ステートの実装によってプロセス・ステート間の分離とセキュリティの高いデータ・パスの実現が可能)、③ソフトウェア開発コストの低減 (ハードウェアによってプロセス間の分離が行われソフトウェアのインタフェースの複雑性が軽減される)、等があげられる。(B. W. Manning, "A Hardware Executive for High Performance Local Computer Networks", 1983.)

●Ada の実用上の障害としてのソフトウェア・ブリビット——ソフトウェア・ブリビット (Software Blivits) は、プロセッサの処理能力や記憶容量の限界までハードウェアを利用すること、あるいはその

ソフトウェアをいう。たとえば、64KB のハードウェアに大規模なコンパイラを実装すること、極度に高速性を要求される射撃制御システム、古くから使用されている大多数のシステムも要求の成長に応えることによってソフトウェア・ブリビットになっている。

ソフトウェア・ブリビットは、一般的にライフ・サイクル・コストの増大と信頼性の低下をもたらすために好ましくないが、短期間利益を重視する際に生じることが多い。全体に占めるソフトウェア・コストの割合が大きい時にはブリビットは避けるべきであり、ハードウェア・コストが大きい時はブリビットにすべきである。

最近登場してきた Ada はソフトウェア・ブリビットの問題に直面している。Ada の設計目標は、①組込型軍用システムのライフ・サイクル・コストの低減、②コードの正当性の向上、③近代的プログラミング技法の支援の 3 点であり、①と②の目標達成のためにコードの効率を犠牲にしており、FORTRAN や JOVIAL よりコードの実行効率是一般に低い。このため、Ada を利用してソフトウェア・ブリビットを構築せざるをえない場合は、①インタラプト処理の速度、②最大データ・サイズやメッセージ・サイズ、③実数・浮動小数点数の範囲と精度、④最適化の程度、⑤システム・ライブラリ等に注目して適切な版を選ばなければならない。いずれにせよ、性能の極度に要求される一部の軍用システムではアセンブラの利用は今後も続くであろう。(J. K. Cross, "The Impact of Ada on Software Blivits", 1983.)

●COMPCON Spring '84—San Francisco,
1984年2月28～3月1日

大会の前日には、エキスパート・システムの CAD および CAT (Computer-Aided Testing) への応用、エキスパート・システム (知識工学)、ローカル・エリア・ネットワークをテーマとしたチュートリアル・セミナーが開かれた。

今回の基調講演は、C. G. Bell 氏 (Encore Computer) により「知的で開発への改良的アプローチ」と題して行われた。なお、今回の主なテーマは、データフロー・スーパー・コンピュータ、生体材料による分子レベル・コンピュータ、インタラクティブ・ビデオ・ディスク、超並列多重プロセッサ、レーザ・プリンタ、ワーク・ステーションの標準化・性能・市場、MODULA-2、LOGO、論理型プログラミング、LOOPS による知識プログラミング、人工知能用コンピュータ・システム・アーキテクチャ、空軍および陸軍の STARS (Software Technology for Adaptable, Reliable Systems) 計画、著述用ツール (書簡文章批評システム、AUGMENT、

UNIX 用の文章制作管理システム、LISP マシン用ドキュメンテーション・システム SAGA)、UNIX の移植性、等であった。

●情報処理学会第 28 回全国大会—東京、1984年
3月13日～15日

特別講演は、西尾幹二氏 (電通大) 「西ドイツからみた日本像—先端技術とアジア的混迷のはざま」、招待講演は、石川二郎氏 (東工大) 「バイオリン“狂想曲”」であった。

パネル討論は「情報処理におけるモデリング—知識ベース、データベース、アブストラクト・データタイプ、CAD/CAM およびシミュレーションの立場より—」と「システム性能評価」について行われた。

話題は多岐にわたるが、知識応用、エキスパート・システム、知識表現、知識ベース・システム、論理型言語、推論、Prolog、Prolog マシン、LISP、機械翻訳、自然言語処理、自然言語理解といったいわゆる人工知能関連分野と、これについてローカル

●NCC '84—Lasvegas, 1984年7月9日～12日

1984年の全米コンピュータ会議 (NCC '84) は7月9日から4日間、Lasvegas のコンベンション・センターで開催された。今年のテーマは、“Enhancing Creativity”であった。年々展示希望者数は増大の一途であり、今年は650社以上、テクニカル・セッションの数は200と膨くれあがり記録を更新し続けた。主催者としては質の向上を目指し、出展を厳選すると動きもあり、入場者は9万人をこえたと言われる。NCC '84は、D. Regan 大統領のビデオ・メッセージで始まった。4メートル四方もあるスクリーンに現われた Regan 大統領は「情報産業の分野において米国の地位を確固たるものとし、半導体の分野においては70パーセントの市場占有を実現した情報産業関連の人々の貢献に心から感謝をしたい、これからもこれ (強いアメリカ) を維持発展させるため政府としても努力したい。」と、感謝の口調の中にもこの分野で米国の地位を絶対なものにする姿勢を感じさせた。

IBM の J. Akers 社長が基調講演を行った。「テクノロジーの発達は、より使いやすいものを生み出し、新しい用途を創造しその結果膨大な和な

市場を創り出す。しかし、これを健全な形で行うためには、企業は社会に対し責任のある行動をとらなければならない。それは、①品質・安全性に対する責任、②著作権・不正行為防止の責任 (ソフトとチップ・パターン)、③社会・教育に対する責任 (産業の協力体制) である。」と述べた。また、プライバシーの保護とコンピュータ犯罪の防止に関してもふれた。展示はパソコン (PC) に席卷されたと言える。

4年ほど前までの NCC は一般展示とパーソナルコンピュータの展示が併設されていた。そして Radioshak, Comodor, Atari といったメーカーとそれを取りまくソフトハウスが主としてホビー中心に展示を行っていたが、今や PC はメインフレームのものとなり、Sperry 社、IBM 社、Honeywell 社、NCR 社、Burroughs 社、DEC 社はそれぞれ PC を展示していた。これに今年から AT & T が PC と 3B2, 3B5 を持って参入した。PC-DOS と UNIX が PC の OS の二つの大きな流れを作り出している。UNIX マシンが急増しているのもここ1～2年の変化である。

もう一つの流れは、デスクトップ・コンピュータである。これは、PC がシングル・ユーザを前提にしているのに対し、マルチ・ユーザを対象とするもので、32ビットのマイクロプロセッサが使用されは

エリア・ネットワーク、ネットワーク・プロトコル、ネットワーク・システム、ネットワーク・ソフトウェア等の通信関連分野が目立った。

●DAC 21 (21st Design Automation Conference)
—Albuquerque (New Mexico), 1984年6月25日～27日

今回の基調講演は、L. Conway 女史 (DARPA, 国防上級研究プロジェクト局) によって、「人工物、ツールおよび社会的基盤—知識の創造と普及のための新機会 (Artifacts, Tools, And Infrastructure: New Opportunities for Creating and Propagating Knowledge)」と題して行われた。

また、今回のパネル討議は、産学協同 IC/CAD 研究会社 (SRC, Semiconductor Research Corp.) における設計科学研究計画、超高速半導体開発計画 (VHSIC) 用のハードウェア記述言語等をテーマに行われた。このほか、シリコン・コンパイラとエキスパート・システム、セミカスタム革命、ゲート・アレイの配置・配線をめぐってワークショップが開

かれた。さらに、今回のチュートリアル・セミナーはデジタル・システムの CAD への論理型プログラミングの適用、配置設計ツール、機械工学用ハードウェアであった。なお、今回の一般講演の主なテーマは、障害シミュレーション・プログラム、配線設計アルゴリズム、ハードウェア記述言語、モジュール・ジェネレータ、CAD データベース・システム、VLSI 配置設計システム MAGIC、テスト・アルゴリズムの改良、レイアウトの検証と設計ルールのチェック、機能シミュレーションと解析、設計ツールとインタフェース、人間工学とエンジニアリング・ワーク・ステーション、論理回路の合成、Bell 研究所における CAD システム、自動配置、Rutgers 大学における人工知能 VLSI CAD、PLA 設計技術、CAM/CAT (Computer Aided Testing)、INTEL の設計自動化システム、テストングの問題と解決、タイミング・アナリシス、配置設計システム、IC 設計システム、人工知能と幾何データベース等。

●第2回国際論理型プログラミング会議—

じている。デスクトップ・コンピュータと PC がローカルエリア・ネットワーク (LAN) で結合されて、コモディティの世界が形成されていくものと考えられる。これらをめぐり小物のペリフェラルが活況を呈していた。とくに、日本のメーカーはペリフェラルを中心に OEM セールスを展開していた。5 $\frac{1}{4}$ インチの固定ディスクは 30~50 MB 級のものを中心になっているが 100 MB を超えるものを展示するメーカーもあった。256KB の RAM を実装するところも現れてきており、メモリ、ディスクの大容量化、低価格化は PC やデスクトップ・コンピュータをますますパワーfulなものにしている。LAN に関しては Ethernet も目に付いた。主要 10 社が協力して Inter-Vendor-Communication のデモをしていた。これは ISO のクラス 4 トランスポート・レベルをサポートするもので CSNA/CD (IEEE 802.3) 方式の LAN に Boeing, DEC, Honeywell, ICL, Intel, NCR, HP, NBS の各社等が参加していた。もう一つのデモは、GM 社の MAP (Manufacturing Automation Protocol) で、トークン方式 (IEEE 802.4) のブロードバンドの LAN であり、IBM, DEC, HP, Motorola, Gould の各社を含む 7 社の機器を接続し生産管理の例を行っていた。これらは参考展示とのことであるが、異機種結合 LAN の活用例として注目を集めていた。

その他では光ディスクを展示しているところが目立ち始めている。この分野は日本の方が先行するように感じられる。

今年の NCC は、とくに極立った展示は少なく、新機種の発表も例年に比べると少なかったように思うが、PC やデスクトップとそれらを結ぶ LAN の動きが大きなコモディティの流れを作り出しているとの印象を受けた。

なお、Sperry 社のブースでは、Sperry PC, MAPPER 5, LAN (Ethernet), Sperry Link Office System, MAPPER を展示しており、今年 4 月に発表した MAPPER 5 が多くの関心をひいていた。

(ハードウェア・プロダクト総括部 アドバンスト・ハードウェア・プロダクト・グループ 伊東 玄)

★

Uppsala (Sweden) 1984年7月2日～6日

本年の招待講演は、Prologの生みの親として知られる Edinburgh 大学の R. Kowalski 教授と、プログラミングへの構成的数学 (Constructive Mathematics) の応用で知られる Stockholm 大学の P. Martin-Löf 教授によって、それぞれ「論理型プログラミングの問題」、「直観論理におけるタイプ理論と論理型プログラミング」と題して行われた。また、今回は Imperial College の K.L. Clark 教授によって「論理型プログラミング」のチュートリアル・セミナーが開かれた。このほかレゾリューション・プリンシプルで知られる Syracuse 大学の J.R. Robinson 教授を議長として、「論理型プログラミングの諸問題」をテーマにパネル討議が開催された。

なお、一般講演 (発表論文31編) のうち主なテーマは次のとおり。

アプリケーション分野では、SIMPOS の設計、Prolog のユニファイアの最適化ツール、文書推敲支援ツール Epistle の意味解析、ギャッピング文法、Prolog データベース・マシン用ユニフィケーション、Prolog を利用した Mycin 型エキスパート・システム、組合せ論における Prolog の応用、離散系シミュレーションへの Prolog の適用等。

論理型プログラミング言語分野では、並行 Prolog における Eager and Easy 評価、ミューダブル・アレイの組み込み、論理型プログラミング言語における同値性・型・モジュール・ジェネリック、連想並行評価、並行 Prolog 用のユニフィケーション・アルゴリズム等。

アーキテクチャおよびハードウェア分野では、OR パラリズムと作用アーキテクチャ、Prolog マシン・アーキテクチャ、並列論理型言語用アーキテクチャ、一般化データフロー・モデルにもとづく高度並行 Prolog マシン、記憶管理マシン等。

日本からの発表論文は4編、①S. Takagi (ICOT) 「SIMPOS の全体構想」②H. Hirakawa (ICOT) 「並行 Prolog における Eager and LASY 評価」③H. Tamaki (茨城大) 「論理型プログラムの Unfold/Fold 変換」④K. Nakamura (電機大) 「論理型プログラムの連想並行評価」であった。

《p.120 から続く》

要な多くの問題が手に負えない複雑さをもっていることを示し、情報処理の応用における理論的障壁について述べている。10章では、この理論的障壁を踏まえて新しい情報処理の局面を開くいろいろなアプローチを事例を使って考察している。第2部は、命令型プログラム論と計算量論への入門である。

第3部では、ソフトウェアの信頼性にスポットを当てている。11章では、関数型プログラム系 (Hilbert, Kleene, Herbrand-Gödel-Kleene) を考察している。12章では、関数的プログラムの働きを考え、操作の規則としての関数研究の系譜を述べる。13章では、関数的プログラム系を定式化し、その事例を解説し、不動点理論を述べている。14章では、不動点理論に対して Herbrand-Gödel-Kleene の方程式算法による関数的プログラムの操作的解釈を与え、関数型計算機の設計の問題を探究する。15章では、命令型プログラムと関数型プログラムの正当性の検証についてまとめて考える。第3部は関数型プログラム論とプログラム検証論への入門といえよう。

第4部では、プログラムの外延の意味論の基礎となる Scott の意味空間の構成を述べている。16章は Russell の逆理を用いて Scott の問題意識を考え、反射的領域を構成する Scott のアイデアを述べている。17章は、反射的領域の具体例としてグラフ・モデル P_W を述べ、Scott の公理系によって彼のアイデアを整理して考察している。18章では、Scott のプログラム系 LAMBDA を述べ、その応用例として Meyer-Ritchie のプログラム系等の表示の意味を記述している。

なお、付録として19章に人工知能研究の歴史、方法論を簡単に述べ、単純なパズルを用いて問題解決の実験を行っている。

以上、情報処理の基礎理論から応用に至るまで、幅広く書かれた本である。

昨年末、日本ユニパックでは「CHAPARRAL ファミリ」(中型コンピュータ・シリーズ)を発表し、その後も利用者のニーズに応えた多くの製品を発表してきました。ここでは、その中から主なものを選んで紹介します。なお、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。

●新キャッシュ・ディスク・システム

キャッシュ・ディスク・システムは、高速の緩衝記憶(キャッシュ・メモリ)を大容量の磁気ディスク装置とキャッシュ・ディスク・システム制御装置との間に配置し、高速で大容量の補助記憶装置を実現したもの。すなわち、磁気ディスクの特徴を損うことなく、主記憶装置と磁気ディスク装置の大幅なアクセス・タイム・ギャップを埋め、磁気ディスク装置に対する実効アクセス速度の短縮と同時に、システムのトータル・スループットの向上が図れる。対象となる磁気ディスク装置は、8450, 8470, 8470-II型である。キャッシュ・ディスク・システム(C/DS)とセミコンダクタ・オグジュアリ・ストレージ(SAS)の2系統があり、C/DSでも磁気ディスク・サブシステムと比べ約2～4倍のパフォーマンスが得られる。

特徴は、アプリケーションに応じて高い効率を保つため三つのオペレーション・モード(SASモード、キャッシュ・モード、ミックス・モード)がある。高速化を図る機能には、ポスト・ストア機能、セグメント先読み機能があり、ファイル・リカバリ機能として、ストア・スルー・モード機能、タイム・スタンプ機能がある。さらに、ARMを十分に考慮し、停電時の補助電源供給、キャッシュ・ストレータのECCがある。(資料コード: 481205130)

●UNISERVO 22型/24型磁気テープ装置

CHAPARRAL ファミリ用に開発され、コントロール・ユニットを簡素化し低価格が図られた磁気テープ装置。テープ・スピードは75ips(22型)、125ips(24型)で、データ転送速度は、120KB/S(22型)、200KB/S(24型)である。また、1600bpiの記録密度をもち、記録方式はPE(フェーズ・エンコーディング・モード)である。さらに自動テープ装填機構をもち、磁気テープ制御機構当たり、最大8台のドライブが接続可能である。(資料コード: 481273002)

●0889型磁気テープ装置

シリーズ1100(1100/60, 70, 80, 90)およびCHAPARRALファミリの各システムに接続可能。6250

bpi(GCRモード)および1600bpi(PEモード)いずれでも読み書きができ、テープ・スピードは125ips、データ転送速度は781KB/S(GCRモード)、200KB/S(PEモード)である。機能的には、UNISERVO 34型と同等であるが、マイクロ・プロセッサとLSIを採用して小型化、高信頼性、低価格を実現した。(資料コード: 481223002)

●シリーズ1100日本語文書システム/入力

日本語文書システム/入力は既存のワード・プロセッサ等では対応できないマニュアル・技術文書、レポート、社内レター類等、多種多様な文書を作成するためのソフトウェアである。入力方法は、だれでも容易にかつ迅速に入力できるよう、「文節数最小法」による高度なカナ漢字変換方式を採用している。この方式を採用することにより文章単位での入力が可能で文章の流れにしたがって自然に入力を進めることができる。

日本語文書システム/入力は、使用者プログラムを作成することなしにカナ漢字変換機能を利用可能にしたカナ漢字変換プログラム(略称JASTY)と使用者プログラムで簡単にカナ漢字変換機能を使えるようにしたカナ漢字変換ライブラリ(略称JASLIB)と、使用者指向辞書を作成/更新する使用者指向辞書保守プログラム、および漢字入力専用機で作成した磁気テープを日本語テキスト・ファイル(シンボリック・エレメント)に変換する日本語テキスト・ファイル変換プログラムから構成されている。(本誌p.40参照。資料コード: 481205470)

●シリーズ1100日本語文書システム/作成・出力

日本語文書システム/作成・出力は、多種多様な文書を構成するためのソフトウェアで、次の3種類のプログラムから構成されている。

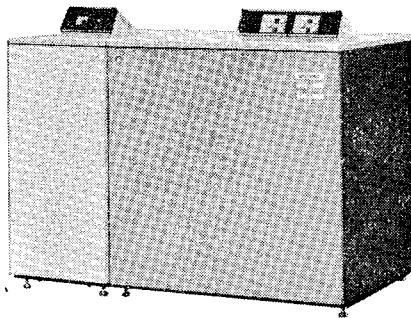
日本語文書システム/入力で作成した日本語テキスト・ファイルを入力とし、その中に指示された書式制御命令にもとづいてテキストを構成する。この書式制御のための豊富な機能がある(日本語文書構成プログラム)。

文書に必要な索引の作成機能があり、索引には、

50音順に索引語を配列する和文用索引とアルファベット順に索引語を配列する英文用索引とを作成する機能がある（日本語索引作成プログラム）。

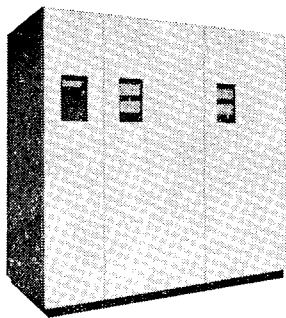
さらに、日本語テキスト・ファイルがどのような文書を作成するものであるかを確認するための文書目録を作成する機能もある（日本語目録作成プログラム）。また、本刷りファイル等の0780日本語プリンタへの出力機能は各々のプログラムに含まれている。（資料コード：481205472）

●新 5056/8470-II 型磁気ディスク・サブシステム



8470-II型磁気ディスク・サブシステムは、マイクロ・プログラム制御方式を採用した新5056型磁気ディスク制御装置と8470-II型磁気型ディスク装置から構成されている。8470-II型磁気ディスク装置は、アーム移動時間最小7ミリ秒、平均20ミリ秒、データ転送速度は1260KB/Sである。また、1駆動機構当たり672MBの記憶容量をもち、ディスク・パック、磁気ヘッド・アーム移動機構を一体化した固定型の磁気ディスク装置である。（資料コード：181821006）

●8480-II 型磁気ディスク・サブシステム



8480-II型磁気ディスク装置は、アーム移動時間最小5ミリ秒、平均16ミリ秒、データ転送速度は2500KB/Sであり、1駆動機構当たり1272MBの記憶容量をもち、高密度実装技術によ

り、従来の磁気ディスクと同一容量で比べて50パーセント以上のスペースを縮小でき、消費電力も40パーセント以上の削減ができる。二つの制御装置で一つの系統の磁気ディスク・サブシステムを制御するデュアル・コントロール方式の採用により、複数の磁気ディスク装置の同時併行読取り／書込みが行え、処理効率を大幅に向上できる。（資料コード：081821009）

●シリーズ 1100 無人化支援システム

無人化支援システムとしてのUOF（Unattended Operation Facility）は、システム運用の自動化を実現する機能とノーダウン・システム支援機能の二つから構成されている。

システム運用の自動化を支えるハードウェアとしてSMCP（System Monitoring & Control Processor）があり、ソフトウェアとしてはUOSS（Unattended Operation Support Software）がある。この二つが連動して電算機室全体の自動電源制御、環境監視および異常時の自動制御、さらにシステムの自動稼動を制御する。

ノーダウン・システムを支えるハードウェアは、ASU（Automatic Switching Unit）があり、ソフトウェアとしてはSAFE 1100（System for Automated Failsafe Environment for 1100）がある。これらにより、本番業務中にシステムが障害を起こした場合、自動的にバックアップ機に本番業務が切り換わり、エンドユーザにとっては事実上ノーダウン・システムとなる。（本誌 p.116 参照）

●シリーズ 1100 OSF 1100

OSF 1100 は、シリーズ1100統合OAの一環としてMAPPER 1100のランで作成、パッケージ化されたもので、オフィスワークを電子化したソフトウェアであり、豊富な機能をもっている。

電子メール機能は、メール文書の作成、内容変更、配布、電子ファイリング、不要メールの破棄等の機能をもち、カナ漢変換機能を内蔵している。スケジュール機能は個人スケジュールの作成、更新、保存を行う機能である。会議予約機能は会議の開催スケジュールをとりまとめる機能である。さらに、個人メモの作成、更新、保存、表示する電子メモ機能、社内・部内各掲示を作成、登録、変更、掲示管理をする掲示機能、メッセージ・メニュー選択と書き込み方式により取り次ぎメモを作成する電話ログ機能、その他導入サポート、ユーザ属性の変更や削除、代行者の指名、ベル・サービス、期日指定メールの

警告日数の設定等のための機能がある。(資料コード: 481204403)

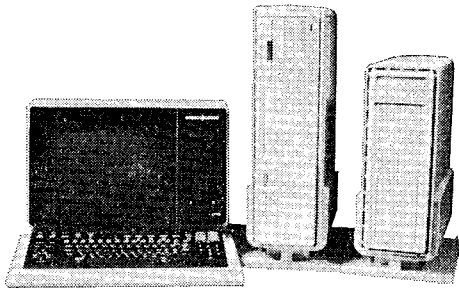
●シリーズ 1100 TRTR-C/2

TRTR-C/2 (TRack TRace for COBOL レベル 2) は、COBOL プログラムのテスト・カバレッジを計測するためのツールであり、デバッグや検収を支援するソフトウェアである。TRTR-C はバッチ処理 COBOL プログラム用で、TRTR-C/2 はリアルタイム処理 COBOL プログラムにも適用できる。リアルタイム処理 COBOL プログラムは、プログラム間の引数のログをとることができる。ただし、ログ・ファイル印書プログラムは使用者が作成する必要がある。(資料コード: 481205475)

●UTS 4000 ターミナル・システム・ファミリ日本語文章作成プログラム

本ソフトウェアは、UTS 50-Ⅲ/Ⅳ の TCP のもとで稼動するユーティリティ・プログラムで、文書の作成・編集・印書・通信等の文書処理業務を効率よく行うことができる。文書の作成は文節単位でのカナ漢字変換による入力が行える。また通信機能によりホスト・コンピュータからのデータを文書中に取り込む等、“オンライン・ワードプロセッサ”としても使用できる。(資料コード: 472845210)

●MAPPER 6



MAPPER 6 は、マイクロ・プロセッサを使用した DSP (Distributed System Processor) に MAPPER システムを搭載した MAPPER 専用機である。

システムの構成は、DSP と磁気ディスク装置、カートリッジ・ストリーミング・テープ装置、UTS 20 B/UTS 50 のワークステーション 0434 型/0450 型/0406 型印書装置からなる。

DSP の付加機構として、MAPPER 6 同士最大 6 台までを接続するためのインタフェースとシリーズ 1100, あるいは IBM 系他ホスト・コンピュータと接続するためのインタフェースがあり、拡張性が高い。ホスト接続インタフェースにより MAPPER

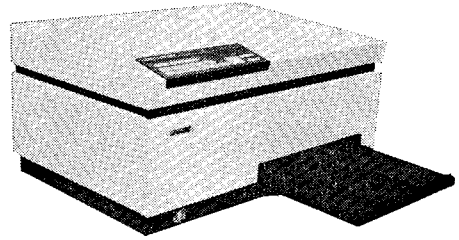
6 と上位 MAPPER システムと接続し、MAPPER 分散システムが可能である。

完全な上位互換性が保証されており、MAPPER 6 のユーザはそのまま MAPPER 11 (CHAPARRAL), MAPPER 1100 へ移行することができる。

また、MAPPER の設備仕様等については、通常のビル室内空調アース極付きの 100V フロア・コンセント電源で稼動する。

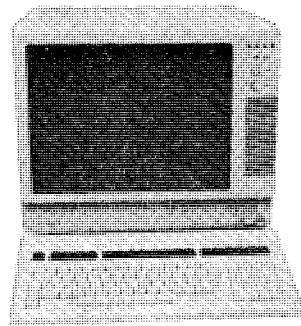
外部記憶装置としての磁気ディスク/テープ装置は、28.5MB の固定ディスクと 45MB のカートリッジ・テープであり、増設は 57MB の磁気ディスク・ユニットが最大 3 ユニットまで付加できる。さらに、MAPPER 6 の各種機能の習得に当たっては、個別学習システム LEARNUP が用意されている。

●J 0780 型カラー・ハードコピー装置



AGS 1000/2000/3000 シリーズ用のカラー・ハードコピー装置である。熱転写方式による記録方式を採用し、色数は、青、赤、緑、黒、黄、マゼンダ、シアン、白の 8 色である。コピー時間は、平均 65 秒/枚 (A4) と高速であり、プリセット・キーにより 99 枚までの同一コピーがグラフィック・ディスプレイとは独立に可能である。さらに着通紙のみでなく、OHP フィルムにもコピーができる。(資料コード: 0181841018)

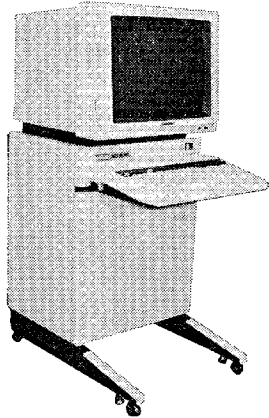
●AGS 1000 V グラフィック・ディスプレイ



1,024×780 ピクセル、60Hz ノンインタレース、14 インチ・カラーモニタの採用により、鮮明な画面を実現している。VT 100 モード、テクトロニク

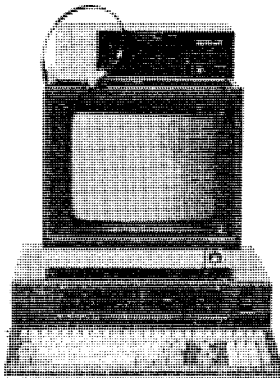
ス・エミュレート・モード機能を標準装備している。また、JIS 第1水準相当のフォント(24×24ドット)を内蔵し漢字表示を可能とし、2,000字まで外字の登録が可能である。(資料コード: 481843111)

●AGS 2400 F グラフィック・ディスプレイ



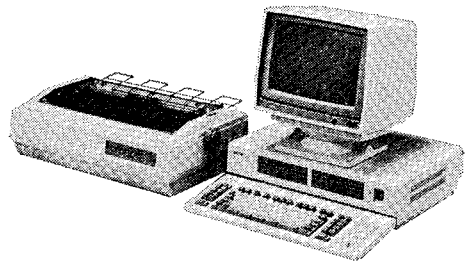
AGS 2400 F は、ホスト・コンピュータからの出力データを表示することはもちろん、ローカル・オペレーションによる出力図形の加工、入力装置からのデータに演算を加えて、ホスト・コンピュータに送出することができる等、高度なインテリジェント機能を備えたグラフィック・ディスプレイ装置である。特徴としては、1,280×1,024ピクセルの高分解表示であるとともに、60Hz ノンインタレース方式による高品質表示画面、32,768色のパレットから最大1,024色のカラー同時表示、最大1MBのセグメント・バッファ容量、その他使いやすく強力な自己診断機能を内蔵し、異常状態の早期発見故障内容の判断および対応を速やかに行うことができる。(資料コード: 081841020)

●Compucent (UP 10 E モデル 35)



UP 10 E (モデル 40/50/60) の下位機種であり、基本的にこれらのシステムと互換性をもち、ハードウェア構成によりモデル 35, 35L, 35F の3モデルがある。メモリは16KB ROM, 128~256KB RAM, 96KB VRAM を装備し、14インチの独立型 CRT ディスプレイ装置は、80字×25行の表示容量と640×400ドットのグラフィック解像度をもち、カラー8色表示が可能である。個別学習システム LEARN-UP を低価格で実現する16ビット・パーソナル・コンピュータである。(資料コード: 472775220)

●UW 8



入力方式に約23,000語の国語辞書、音訓・部首別画数用の漢字辞書を使用しているカナ漢字変換方式を採用、最新使用単語を優先する学習機能もある。帯、棒、折れ線、円グラフをはじめ、豊富な文書編集機能、印刷機能をもっている。UW 5の機能はすべて有し、文書ファイルの互換性もある。(資料コード: 472815215)

●シリーズ8 漢字関連ソフトウェア

シリーズ8 漢字 DPS 10 辞書ファイルおよびソフトウェアとしては、11.3万件の姓と8.1万件の名をもつ DPS 10 氏名辞書、3.4万件の住所をもつ DPS 10 住所辞書-II が発表され、それぞれカナ漢字変換に使用することができる。さらに DPS 10 文章辞書-II は、ワードプロセッシング機能をもち、文書の新規作成・修正・変更、複写・消去・印刷が容易にできる。また、簡単なパラメタでビジネス・ユースのグラフが作成可能な DPS 10 グラフ・プリント・システムを発表した。(資料コード: 481755417)

▶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, Development Planning & Support)

《日本ユニバック》

伊東 玄 (ハードウェア・プロダクト統括部 アドバンスド・ハードウェア・プロダクト室), 岩田裕道 (プロダクト本部 ソフトウェア二部部長), 黒木建雄 (システム本部 システム四部部長), 斉藤一弥 (システム本部 システム四部原子力グループ・マネージャ), 高山龍雄 (システム本部 副本部長), 外山晴夫 (プロダクト本部 1100/90 導入プロダクト部長), 林 靖夫 (名古屋システム部 副部長), 松倉 司 (プロダクト本部 ソフトウェア二部日本語ソフトウェア2グループ・マネージャ) 本田都南夫 (プロダクト本部 ソフトウェア企画部ソフトウェア・デザイン・グループ・マネージャ), 山岸史明 (企画部 商品企画一室), 山崎利治 (技術企画部 企画調査室), 湯川高繁 (商品企画部 商品企画一室), 渡部義維 (応用ソフトウェア事業部 技術計算グループ・マネージャ)

▶エディトリアル・スタッフ

《Sperry, Computer Systems》

E. J. Bucci (Manager, Technical Communications)

《日本ユニバック》

(テクニカル・パブリケーション室)

広野和夫 (室長), 山田真市 (主任研究員), 桑野龍夫 (主任研究員), 高橋 肇 (主任研究員), 小山憲一, 青柳幸久, 丹野敬子

●Technical Coordinators

Y. Hayashi, T. Honda, K. Itou, H. Iwata, T. Kuroki, P. J. Lazar, T. Matsukura, K. Saito, Dr. A. J. Schneider, T. Takayama, H. Toyama, Y. Watanabe, F. Yamagishi, T. Yamazaki, T. Yukawa

●Editorial Staffs

K. Hirono, E. J. Bucci, S. Yamada, T. Kuwano, H. Takahashi, K. Oyama, Y. Aoyagi, K. Tanno

ISSN 0289-6257

技 報

UNIVAC TECHNOLOGY REVIEW

No. 7

発 行 日 昭和59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

待望のマイクロエレクトロニクス講座 10月刊行開始!

岩波講座 全11巻 マイクロエレクトロニクス

編集委員 ■ 元岡 達・菅野卓雄・渡辺 誠
淵 一博・石井威望

マイクロエレクトロニクスの基礎事項を整理し体系的に解説するとともに、コンピュータ・通信・メカトロニクスなどの情報システムの設計論・構成論を、ハードウェア・ソフトウェアの両面にわたって説き明かした待望の講座。基礎的な考え方や方法をきちんと理解しておけば、どんな急速な進歩にも対応できる強靱で柔軟な応用力を身につけることができる——本講座は何よりもこのことを重視する。現場の技術者・研究者のもっとも信頼できる確かな手引きとして、また学生・教師のための理想的な教育カリキュラムとして、自信をもって本講座をおすすめする。

【本講座の特色】

- (1) 超LSIとそのコンピュータ・通信・メカトロニクスなどへの応用を軸に、マイクロエレクトロニクスを固有の論理と方法をもつ構造として体系化し、編成した斬新な教育カリキュラム。
- (2) コンピュータ・通信・メカトロニクスなどの情報システムの設計論・構成論を、一貫した思想のもとにハードウェア・ソフトウェアの両面にわたって詳述した。
- (3) システム設計の各階層におけるソフトウェア的方法を重視し、十分なページをさいてわかりやすく解説した。
- (4) 抽象的な概念や方法などは、厳密性を多少犠牲にしてもイメージ豊かに、直観的に理解できるような解説を与えた。
- (5) 実例、例題、算法、プログラム例、図・表などを豊富に配し、学生・技術者・研究者の実践的な手引きとなるようにした。
- (6) 学部学生が十分に納得して読み進むことができるよう基礎事項には思い切ったページ数を割り当て、わかりやすく丁寧に解説。

第1回配本/10月刊行

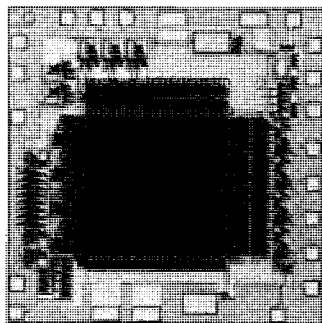
⑥ マイクロコンピュータのプログラミング
石田晴久

第2回配本/11月刊行

⑤ マイクロコンピュータのハードウェア
森下 巖

第3回配本/12月刊行

⑧ VLSIコンピュータ
元岡 達ほか



〔全11巻の構成〕

- ① マイクロエレクトロニクス素子 I
— デジタル素子とプロセス
- ② マイクロエレクトロニクス素子 II
— 光・記憶素子とセンサ
- ③ VLSIの設計 I
— 回路とレイアウト
- ④ VLSIの設計 II
— 論理とテスト
- ⑤ マイクロコンピュータのハードウェア
- ⑥ マイクロコンピュータのプログラミング
- ⑦ プログラミング言語とVLSI
- ⑧ VLSIコンピュータ I
- ⑨ VLSIコンピュータ II
- ⑩ システム構成技術
- ⑪ メカトロニクス

■刊行方法

10月より毎月1巻ずつ定期刊行いたします。

■体裁

A5判上製カバー・平均260頁・月報付。

■頒布方法

本講座は予約制です。全11巻を予約申込みされた方にもお預けいたします。

■「内容見本」は8月下旬に出来予定で、お近くの書店または小社営業部までご請求ください。

岩波書店



東京・千代田・一ツ橋
振替く東京)6-26240