文　献　紹　介

A：数値解析　B：プログラミング　C：計算機方式
D：回路および機器　E：オートマトン　F：応用その他

A-46. $\int_0^1 -\ln(x)f(x)d(x)$ に対する Gauss の積分公式

Donald G. Anderson: Gaussion Quadrature Formulae for $\int_0^1 -\ln(x)\,f(x)dx$〔Math. of Comp., Vol. 19, No. 91, July, 1965 pp. 477～481〕

Fredholm 形の積分方程式の数値解法として，逐次近似法が考えられる（〔1〕など参照）．一次元の場合，頻繁な数値積分を Gauss の積分公式で求めよう．

Gauss 法には，固定した刻み点における，被積分関数の値をほとんど最適に使えることと，特異核や，無限区間の場合にも適用できるという二つの利点がある．しかし，高次の（したがって高精度の）刻み点の値と重みを得ることは容易ではないので，それらの数表が与えられていない核に対しては適用しにくい．本論文は 10 次程度までの，刻み点と重みの求め方の簡単で経済的な，アルゴリズムを示している．

次のことは良く知られている（〔2〕など）．

区間 $[a, b]$（$-\infty \leqq a < b \leqq \infty$）で，非負な関数 $w(x)$ が，さらに任意の正整数 k に対して $\int_a^b x^k w(x)dx < \infty$ をみたすとする．

$$(f, g) \equiv \int_a^b f(x)g(x)w(x)dx$$

とすれば，$n=0, 1, 2, \cdots\cdots$ に対して，多項式

$$P_n(x) = \sum_{i=0}^{n} (P_n)_i x^i$$

が存在して，$i \neq j$ なら

$$(P_i, P_j) = 0$$

である．

$P_n(x)$ の根 $x_k^{(n)}$（$k=1, 2, \cdots\cdots, n$）は

$$x_0^{(n)} \equiv a < x_1^{(n)} < \cdots\cdots < x_k^{(n)} < \cdots\cdots < x_n^{(n)} < b \equiv x_{n+1}^{(n)}$$

をみたし，さらに $P_{n+1}(x)$ の根との間には

$$x_k^{(n)} < x_{k+1}^{(n+1)} < x_{k+1}^{(n)} \qquad (1)$$

なる関数がある．ここで

$$H_k^{(n)} \equiv \frac{(P_n)_n (P_{n-1}, P_{n-1})}{(P_{n-1})_{n-1} P_n'(x_k^{(n)}) P_{n-1}(x_k^{(n)})} \qquad (2)$$

とおけば

$$\int_a^b f(x)w(x)dx \sim \sum_{k=0}^{n} H_k^{(n)} f(x_k^{(n)})$$

であり，右辺の近似和は $(2n-1)$ 次までの多項式に対して左辺の真の値を与える．

次に，$x_k^{(n)}$, $H_k^{(n)}$ を求めるアルゴリズムを示す．簡単なために，機械は2進法であるとして，適当な整数 m を定めて

$$(P_n)_n = 2^{mn}$$

と標準化した $P_n(x)$ を求めよう．

$$P_0 \equiv (P_0)_0 = 1,\ P_1 = (P_1)_1 (x - r_1)$$

を定める．次に

$$u_n \equiv (xP_{n-1}, P_{n-1}),\ t_n \equiv (P_{n-1}, P_{n-1})$$

とおき

$$r_n \equiv \frac{u_n}{t_n},\ s_n \equiv \frac{(P_n)_n (P_{n-2})_{n-2}}{\{(P_{n-1})_{n-1}\}^2} \frac{t_n}{t_{n-1}} \qquad (3)$$

とすれば，$P_n(x)$ は

$$P_n(x) = \{(P_n)_n / (P_{n-1})_{n-1}\}(x - r_n)P_{n-1}(x) - S_n P_{n-2}(x)$$

をみたす．いま，$(P_n)_i = 0$（$i > n$ または $i < 0$）とすれば上の漸化式は

$$(P_n)_i = 2^m (P_{n-1})_{i-1} - 2^m r_n (P_{n-1})_i - S_n (P_{n-2})_i,\quad (P_0)_0 = 1 \qquad (4)$$

と同じである．ここで

$$M_l = (x^l, 1),\ (0 \leqq l \leqq 2n-1)$$

とすれば

$$\left.\begin{aligned} t_n &= \sum_{i,j=0}^{n-1} (P_{n-1})_i (P_{n-1})_j M_{i+j} \\ u_n &= \sum_{i,j=0}^{n-1} (P_{n-1})_i (P_{n-1})_j M_{i+j+1} \end{aligned}\right\} \qquad (5)$$

である．したがって $n-1$ 次までの $P_k(x)$ が求まっていれば，(5)→(3)→(4) の手順で p_n を求めることができる．(1) により，$x_k^{(n)}$ は $x_{k-1}^{(n-1)}$ を初期値として Newton-Raphson 近似する．このとき，付ずいして $P_n'(x_k^{(n)})$ を求めることも可能である．(2)から $H_k^{(n)}$ が決まる．

公式が，$2n-1$ 次以下では，正しいことから

$$\sum_{k=1}^{n} H_k^{(n)} P_i(x_k^{(n)}) P_j(x_k^{(n)}) = t_i \delta_{ij}$$

である．これをこの過程のチェックに使えることに注

$$\int_0^1 -\ln(x) f(x)dx \approx \sum_{k=1}^{n} H_k^{(n)} f(x_k^{(n)})$$

n	$x_k^{(n)}$	$H_k^{(n)}$
2	0.11200880	0.71853931
	0.60227691	0.28146068
3	0.63890792 (−1)	0.51340455
	0.36899706	0.39198004
	0.76638030	0.94615406 (−1)
4	0.41448480 (−1)	0.38346406
	0.24527491	0.38687532
	0.55616545	0.19043513
	0.84898239	0.39225487 (−1)
5	0.29134472 (−1)	0.29789346
	0.17397721	0.34977622
	0.41170251	0.23448829
	0.67731417	0.98930460 (−1)
	0.89477133	0.18911552 (−1)
6	0.21634005 (−1)	0.23876366
	0.12958339	0.30828657
	0.31402045	0.24531742
	0.53865721	0.14200875
	0.75691533	0.55454622 (−1)
	0.92266884	0.10168958 (−1)
7	0.16719355 (−1)	0.19616938
	0.10018568	0.27030264
	0.24629424	0.23968187
	0.43346349	0.16577577
	0.63235098	0.88943226 (−1)
	0.81111862	0.33194304 (−1)
	0.94084816	0.59327869 (−2)
8	0.13320243 (−1)	0.16441660
	0.79750427 (−1)	0.23752560
	0.19787102	0.22684198
	0.35415398	0.17575408
	0.52945857	0.11292402
	0.70181542	0.57872212 (−1)
	0.84937932	0.20979074 (−1)
	0.95332645	0.36864071 (−2)
9	0.10869338 (−1)	0.14006846
	0.64983682 (−1)	0.20977224
	0.16222943	0.21142716
	0.29374996	0.17715622
	0.44663195	0.12779920
	0.60548172	0.78478879 (−1)
	0.75411017	0.39022490 (−1)
	0.87726585	0.13867290 (−1)
	0.96225056	0.24080402 (−2)
10	0.90425944 (−2)	0.12095474
	0.53971054 (−1)	0.18636310
	0.13531134	0.19566036
	0.24705169	0.17357723
	0.38021171	0.13569597
	0.52379159	0.93647084 (−1)
	0.66577472	0.55787938 (−1)
	0.79419019	0.27159893 (−1)
	0.89816102	0.95151992 (−2)
	0.96884798	0.16381586 (−2)

Note: Numbers are to be multiplied by the power of ten in parentheses.

意しておく．

この方法の問題点としてまず，一般的には M_l は l の初等関数ではない．このときは M_l 自身も数値的な近似値となり，実行される積分核は，元とは多少変化していることである．次に(4)の漸化式においてわかるように，引算による桁落ちが大きくなりやすいことである．結果は単長の精度で良くても，演算は倍長で行なうことが望ましい．またこの理由によって，この方法では，高次のものを生成するのは難しくなっている．

例として，$[a, b]=[0, 1]$, $w(x)=-\ln x$ の場合の本法による $x_k^{(n)}$, $H_k^{(n)}$ を原論文より転載する．この場合には $M_l=(l+1)^{-2}$ である．

[1] Anderson, D. Iterative Procedures for Nonlinear Integral Equations, J.ACM 1965, Vol. 12, No. 4, p. 547～560.

[2] Hildebrand, F. Introduction to Numerical Analysis, Chapt. 7, 8, McGraw-Hill, 1956

（牛島　照夫）

A-47. 算法言語の syntax に関して

Philip Gilbert: On the Syntax of Algorithmic Languages [J.ACM, Vol. 13, No. 1, Jan. 1966 pp. 90～107]

ALGOL のような算法言語の特性を contex free phrase structure grammer によって記述する試みがためされてきたが，そのような試みは部分的に成功したに過ぎなかった．たとえば，ALGOL はブロックの中で用いられる全ての名前は宣言される必要があり，かつ同じブロックの中で2度以上宣言されてはならないという context free grammer では記述できない性質を持つ．

この論文では算法言語の特性を完全に記述し，文の syntactic analysis を一意的に行なうために開発された analytic grammer およびそのいくつかの性質を示している．

analytic grammer G は次のように定義される analysis oriented grammer である．

$G=(V, P, Z, S_F)$

ただし，V は vocabulary

P は production の集合

Z は distinguished symbol (Chomsky の S に対応する)

S_F は scan である．

S_F は string に適用可能な production を制限するためのものであり，これを持つ点が G と phrase structure grammer の相異点である．簡単な scan の例がいくつか示されている．

analytic language $L(G)$ は scan S_F によって決まる production を適用していくと最終的に Z に変換される string の集合である．$L(G)$ は recursive set と同値であること，phrase structure grammer は analytic grammer であることおよび analytic grammer が一意的に解析を行なうための十分条件などが証明されている．

算法言語における宣言された名前の使用および記号定数に関する問題（宣言された array の添字の数と実際に用いられる array の添字の数は一致しなければならないという性質も context free grammer では記述できない）を解決している analytic grammer の例が示されている．著者らは analytic grammer を syntactic analysis のために用いるコンパイラを構成中である．（二村 良彦）

B-48. 概念学習シミュレーションプログラム：CASE

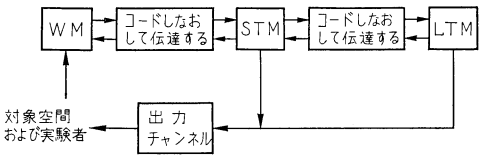
F.B. Baker: CASE: A Program for Simulation of Concept Learning〔Proc. FJCC, 1965, pp. 979～984〕

CASE (Concept Attainment Simulation Experiment) プログラムは，人間が学習によって概念を把握する心理過程および記憶過程を模倣する計算機シミュレーションプログラムである．

概念学習の研究は心理学の分野で古くから行なわれてきた．そして近年 Bruner 等の著作によって概念学習の過程が記述されてから著るしく注目されるようになった．Bruner 等によって行なわれた実験は次のものである．まず実験者は被実験者に，ある概念を代表する一つの対象を示し，被実験者にその概念を言葉で表わすことを命ずる．次に被実験者は適当な対象を選び，それがその概念を代表しているかどうかを実験者に示してもらう．被実験者が概念を言葉で表わすことができるまでこの対象選択手続が続けられる．

CASE プログラムはこの実験における被実験者を模倣するために次のような記憶構造およびプログラム構造を持っている．

記憶構造 CASE 記憶構造は人間の記憶構造を適切に表現するために三つのレベル，すなわち working memory (WM), short term memory (STM) および long term memory (LTM) からなっている．WM は対象に関するすべての情報を一時的に，STM は現在の学習状態に関する情報を，そして LTM は学習された概念およびそれが学習された方法を，おのおの記憶する．第1図は記憶のレベルの間の移り方を示す．



第 1 図

被実験者（計算機）は出力チャンネルを通じてどの対象を選んだかを実験者に知らせる．実験者はその対象の概念を代表するものであるかどうかを WM を通じて被実験者に知らせる．STM および LTM からの学習過程に関する情報も出力チャンネルを通じて出力される．

プログラム構造 CASE プログラム構造は概念学習にともなう心理過程の理解が増すにつれて，その分類段階を増すことができるように設計されている．現在は四つのレベルがあり，高い方からおのおの strategy, procedure, process, および R と呼ばれる．各レベルは IPL-V で書かれたルーチンの集りである．

この論文が書かれた時点では，まだ LTM および strategy レベルルーチンは作成されてないが，著者は CASE プログラムによって得られた心理学の分野における成果をいくつかあげている．（二村 良彦）

B-49. 人間対機械の相互作用に関する一見解

Yershóv, A.P.: One View of Man-Machine Interaction〔J. ACM, Vol. 12, No. 3, July, 1965, pp. 315～325〕

プログラム用言語の特徴は，形式的なことにある．このため，最終的に機械語に翻訳されるまで，プログラマの意図が保存される．しかし，プログラム用言語が，豊富な内容をもてばもつほど形式的文法の習得に要する時間は増大し，研究者が一発計算をしたい場合文法を習得する時間が，プログラムそのものをする時間に比しはるかに長くなるであろう．この矛盾の一つの解決法を提案する．

自然言語の十分複雑な形式化である入力用言語と，読み込まれたテキストが言語上整合するか否かの判定をする解析プログラムの存在を仮定しよう．人間は入力用言語を知らないが，自由に自然言語を用いて問題を機械に与える．その問題が解析プログラムに理解できれば実行し，理解不能ならその個所を人間に知らせる．人間はいかなる解答が，機械に対して最善であるかを考えて解答する．以後このサイクルを繰返えすことによって人間の意志を機械に実行させ得る．この過程で人間に要求されることは，“その問題限りにおいて自分を機械に適応させる”と要約できる．形式言語の習得に比し，人間に要求される努力は少なくなるであろう．

解析プログラムは，入力用言語および自分自身を拡張する能力を持つ．この能力によって，上記の過程において機械はまず，当面する問題の解決に当ると同時に将来起り得る同等または類似の質問に対して無駄なサイクルを繰返えすことなく解答できるようになる．

この方式による対話に決定性が無いことが，批判されるであろうが，現在のデバッギングもその決定性を人間が完全に知っている訳ではない（もし知っているとしたら，初めからデバッギング不要のプログラムが作れるはずである）．機械からのエラーメッセージは，機械からの質問と見なせよう．むしろ，この方式は現在のプログラミングからの自然な移行であろう．

この方式の仮定のすべてが実現への問題点であるが，特に解析プログラムにプログラムと言語双方の自己増殖の能力を与えることが最も困難な問題となろう．目下解決すべき問題は次の順に挙げることができる．

（1） 人間が関心をもつ外界のモデルの設定．
（2） そのモデルを記述する入力用言語とその言語の semantic-grammatical kernel の作成．
（3） その kernel による機械との対話法の習得．
（4） 人間対機械会話のダイナミックスの習得．
（5） 一般的命令によって機械のもつ semantic function を拡張するアルゴリズムの精密化．

（牛島照夫）

B-50. ALGOL におけるブロックの効果的管理法

P. A. Samet: The Efficient Administration of Blocks in ALGOL [Computer J. Vol. 8, No. 1, Apr. 1965 pp. 21～23]

ALGOL における最も重要な概念の一つはブロック構造に関してである．そのためこれまでに数多くのブロック管理に関する案が発表されてきた．ここに述べるのもその一案である．

ブロックの管理で特に注意を払わなければならないのは

（1） procedure—特に recursive procedure—の中でブロックが活性化されている場所をいつでも容易に見出せること．

（2） **go to** 文によって procedure を導き出すことおよびそのために必要となる宣言の回復ができること．

（3） したがってそのブロックの正しい管理のもとで表札付文に到達させる必要があること．

などである．これらの管理を遂行するのに，よく知られた方法として block level の方法（Dizkstra & Ruesell）がある．これは記憶装置を有効に使用するが，それを制御するための時間の損失が大きい．ここに提案するのは block level を使わないで，正しい宣言の回復ができる方法である．この方法の基礎となる概念は

（1） block number の使用

（2） stack の代わりに block 自身の中へある管理的機能をもつ情報を貯えること．

である．ここでは block level と block number をそれぞれ translator, processing の段階で比較論じている．

translation: block number は宣言の数を単に数えることで与えられる．これは label が参照する block の決定に使うことができる．level 以外の identifier についてはそれぞれ宣言される block の入口にある stack pointer に関する stack の中のある場所を指すだけで良い．この時点では block number の使用はあやまりの場所を指すだけである．

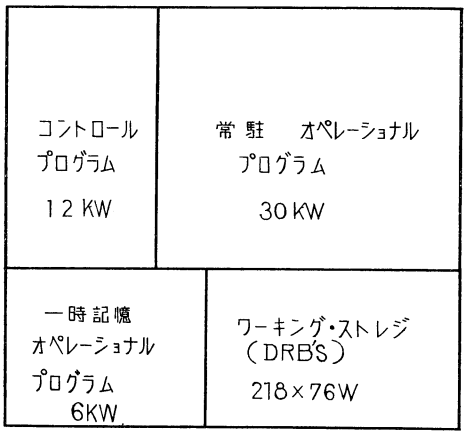
processing: 宣言の回復ができるためには変数を保持している場所に簡単にアクセスできればよいので，それには stack pointer の適当な値に達する術を知っていれば可能である．従来の方法[1]は使っている block level に対応する stack pointer value のベクトルをもつ location があり，それをすぐアクセスできるようになっている．ここでは stack pointer の属している block の中に，その同じ block stack pointer を stack しておこうというのである．単純変数と配列への参照はある特別の location から関係のある

stack を call するように間接アドレスを使って行なう．recursion の管理はブロックの入口でブロックの前の活性化に対応する block stack pointer が新しいそれと置き変えられる前に安全に store されれば良い．実行時には現在活性化しているブロックと procedure に関する情報はすべてプログラムの書式に含まれており未活性のブロックの情報は stack の中に保持されている．このシステムを直接適用すると間違う場合が一つある．すなわち recursive procedure が formal parameter として label をもつ場合でその例と救済方法が与えられている．　（長谷　文子）

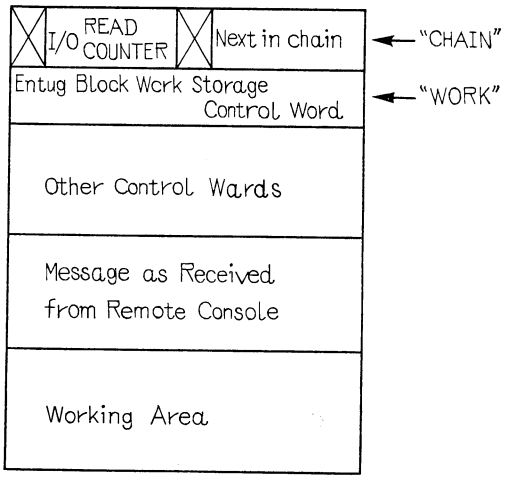
B-51. 実時間システムのコア・メモリのダイナミック・コントロール

William B. ELmore and George J. Evans, Jr.: Dynamic Control of Core Memory in a Real-Time System〔Proc. IFIP, 1965 pp. 261～266.〕

メモリのダイナミックなアロケーションは，実時間システムでは，性能を向上する重要な技術であるが，本論文は，SABRE で用いられてコア・メモリのコントロールの技術とその2年間の使用経験を述べている．SABRE は中央処理装置として2台の IBM 7090（各 65 kW）をもっており，stand by 方式により運転されている．プログラムはコントロール・プログラムが 12 kW，オペレーショナル・プログラムが 130 kW でこの両者はマクロ命令でリンクされる．コア・メモリの用い方は第1図のようであり，常駐のオペレーショナル・プログラムは 30 kW，一時的なものに 6 kW が割り当てられている．ワーキング・ストレージは 76 W のブロックが 218 ブロックあり，このブロック



第1図　コア・メモリ配分



第2図　エントリ・ブロック構造（76 W）

を DRB（データ・レコード・ブロック）という．ここには，入力メッセージなどを入れ処理を行なう．本論文は主としてこの DRB の用い方の技術について説明してある．遠隔のターミナルから入力されたメッセージは，使用可能 DRB プールからえらばれた DRB に貯えられる．そのブロックをエントリ・ブロックというが，メッセージが長いといくつかの DRB が用いられ，それらの DRB は互いに両方向のチェイニングが行なわれる．エントリ・ブロックの構造は第2図のようである．エントリ・ブロックは，二つの主要なリストおよびそれに付随するいくつかの待ちリストにぞくす．一つのリストは CPU リストで，処理中のエントリ・ブロックで CPU の処理をまっているもの（周辺装置の処理を待っているものは別），もう一つはメッセージ・インプット・リストで，メッセージが入力されたが，まだ CPU で処理されていないエントリ・ブロックのリストである．これらの各リストにぞくするエントリ・ブロックは，第2図に示す第1語の“CHAIN”によりそれぞれチェイニングされる．第2語の“WORK”は，そのエントリ・ブロック内の各 DRB を両方向のチェイニングを行なう．不要になった DRB は，使用可能 DRB のプール（これもリストになっている）に入れられる．リストはいずれも first-in・first-out である．Last-in first-out にしなかった理由は，誤動作により，使用可能 DRB プールに重複して返された DRB を，直ぐに使用しないで，時間をおいて，誤り発見の機会をもつためである．SABRE では1分間に 17,640 ブロックが動かされ

た.　　（大野　豊）

C-52. ピコプログラミング：計算機制御の新法

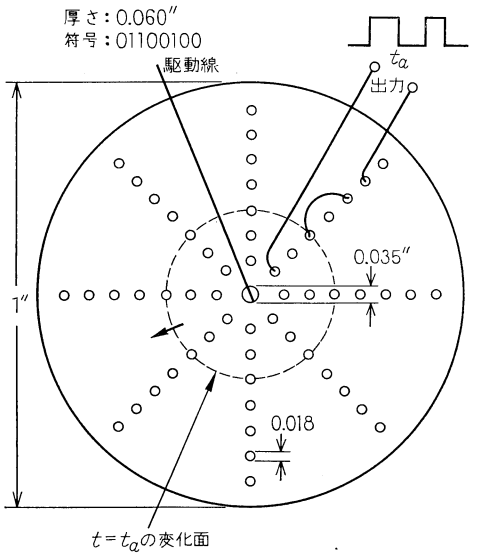
B.E. Briley: Picoprogramming: A New Approach to Internal Computer Control〔Proc. FJCC, Vol. 27, Part I, 1965, pp. 93～98〕

計算機の中央処理装置は演算部と制御部に分れるが，後者はゲート・パルスの順序ある群を発生している．従来の方法では，パルス列の発生を配線によっているため，融通がきかないうらみがある．

融通性を増す方法に，マイクロプログラミングの手法があり，強力なマクロ命令を実行することができる．この場合，ミクロ命令の実行は，あるきまった順序でパルスを発生することにほかならない．さらに進んで，マクロ命令を単位にモジュール構成をとって，設計・診断・保守・更新を容易にしようというのがピコプログラミングの手法である．

マクロ命令の実行は，命令の読出し，MYRA によるパルス列発生器の選択，被演算数の読出し，制御計数器の更新という順序で行なわれる．

MYRA とは図のような多孔（MYR*i* Aperture）フェライト板で，駆動されて磁化が反転するときに，中心部から周辺部へ一定の速度で磁壁が動くことを利用したパルス列発成器である．磁化を戻すような駆動をつけ加えると，幅 125 ns のパルスを計 14 個作ることができ，加算・格納・桁移動・飛越・入出力などの命令はそれぞれ 1 個の MYRA を駆動するだけで実行できる．



ピコプログラム配線を示す MYRA

出力インピーダンスは 10 Ω で，いくつかの命令の OR をとるような特定の機能はそれらの MYRA を串刺しにして実現できる．マクロ命令は一つの MYRA に相当し，ピコ命令は孔に通してある 1 本の線に対応する．孔の通し方は，タイミング・ダイアグラムの矩形波をそのまま，孔をくぐらせて実現すればよい．その孔の位置に対応する時刻に出力電圧は線と同じ向きに変化するからである．

1 語 4 ビットの小さな装置を作って，8 個の命令を毎秒 30 万個の早さで運転したところ，3,000 時間は無事故で働いた．この装置は，小形の計算機ではプログラム記憶式となり，大形の多重処理では，命令形式の更新などが容易になるという利点がある．なお，固定パルス列が，非同期的につながって働くことを Auto chronous と呼んでいる．　　（川合　英俊）

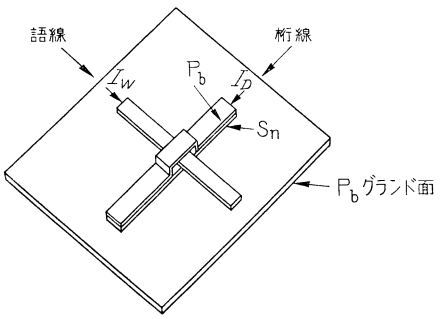
D-53. ブリッジ・セル—新しい超電導記憶素子

R.W. Ahrons: The Bridge Cell—A New Superconductive Memory Cell for Random-Access Word-Organized Memories〔RCA Rev. Vol. 26, No. 4, Dec., 1965 pp. 557～573.〕

図のような構式をもつ超電導薄膜記憶素子で，永久電流が薄膜の面に垂直に補捉されることを特長とし，組み立て密度を上げることができる．

書き込み：桁線の電流はグランド面と，ループのインダクタンスとから Sn に流れるが，語線に電流パルスを与えると Pb に乗りかえる．桁線の電流がなくなってもループの環状電流はなくならない．

環状電流の向きで“0”を表わしてもよいし，環状



第 1 図　ブリッジ・セル

電流のないことで“0”を表わしてもよい．

読み出し：語線に電流パルスを与えると，“1”のとき環状電流はとまり，このとき Sn の両端にインダクタンスを内分するような量の電圧パルスが表われる．

桁線，語線ともに 2A の電流で駆動して働らかせたとき，出力電圧は約 1V であった．また，駆動電流の変化に対して出力電圧は量子化されている．

速さ：幅 100～200 ns のパルスにも応答するから，5～10 Mc のくりかえしに追従すると思われ，駆動線の伝播おくれできまるだろう．

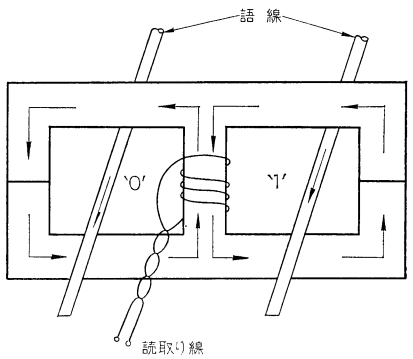
プレーン：記憶場所の選択はクライオトロンによる電流の木を用いる．容量が大きいときには，選択電流がそのまま駆動電流になる形のものより，駆動電流をスイッチする形のものの方が呼び出し時間が小さい．駆動電流経路のインダクタンスを小さくする意味で記配列の方が高速記憶装置には向いている．

最後に動作速度を最良にするには，語やビッドの配列に最適な形があることにふれている．（川合　英俊）

D-54.　E形鉄心を用いた固定記憶

P.S. Sidhu, B. Bussell: Developement of an E-Core Read Only Memory〔Proc. FJCC, 1965, pp. 809～818.〕

第1図に示すようなE形鉄心を素子とする固定記憶装置，1,024 語，24 ビット/語，繰り返えし 250 ns，読出し 150 ns，鉄心の窓枠のどちらに語線を通すかによって，中央脚の読取り線に“1”，“0”に対応する両極性の出力パルスを得る．語駆動電流は 50 mA と非常に少なく，出力は読み取り線8回捲きで 1.8 V と非常に大きい．また，フェライトの場合のように状態の変化による遅れがない．語線間の結合による S/N の劣化が少ない．繰り返えしは鉄心の材質を変え改善できる．



第1図　E形鉄心

一般に線形選択では語の数だけ語線が必要であるが，ここではまた，一致選択の方法で語線の数を減らすことを試みた．一語を何本かの語線の組み合わせであらわし，その組み合わせ方によって出力に異なった情報パターンを得る．一語を2本の組み合わせであらわすとき，最低限 $2\times\sqrt{N}$ 本の語線を用いればよい，ただアドレッシングが非常に複雑になる．

アドレス・フリップフロップ，デコーダ，駆動回路には集積回路を採用した．本装置は高速，高信頼性（両極性の出力，自動配線），低廉，寸法の点ですぐれている．（白倉　隆一）

E-55. 最適診断手順の決定における一方法

H.Y. Chang: An Algorithm for Selecting an Optimum Set of Diagnostic Tests〔IEEE Trans. EC-14, No. 5, Oct., 1965, pp. 706～711〕

ある回路について n 個の故障 f_i を考え，m 個のテスト方法 t_j が与えられたとする．故障 f_i が t_j で検出可能ならば，第1図のような行列の要素 d_{ij} を1とし，そうでないとき0とする．図の例では $m=8$ であるが，必ずしもすべての t_j が必要でなく，たとえば $\{t_2, t_4, t_5\}$ だけですべての f_i を分離できる．もし，毎回 f_i の集合を等分割するいうな都合の良いテストの組が存在すれば，簡単に最良解となることはよく知

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
f_1	1	0	1	0	1	1	1	0
f_2	0	0	1	1	1	1	1	0
f_3	1	1	1	1	1	1	1	0
f_4	1	0	1	1	0	1	0	0
f_5	0	1	0	0	0	1	0	0
f_6	0	1	0	0	1	1	1	0

第1図

られている．しかし一般にパタン $\{t_j\}$ が与えられたときに厳密な最良解を求めるには，いろいろな組合わせをすべて比較する以外に方法がない．m が 1,000 にもなればその手数は膨大となり実用的でなくなる．そこで，あるテストにおける“0”対“1”の関係の数をメリットに選び，局所的な最適化を行なう．重み $w_j(e)$ を

$$w_j(e)=(n_0n_1)_{\bar{t}_j}-\Sigma(n_0n_1)_{\bar{t}_j\cap p_k}$$

と定める．ここで n_0, n_1 は $\bar{t}_j$ の列における0，1の数，p_k はあるパッケージ（取替単位）に含まれてい

て分離する必要がない f_i の集合とする．w_j を分枝について加えWとする．最初すべてのテスト中Wの最大なものを選び，次に行列を修正して残りのものについて同じことをくりかえす．終了は $W=0$ で判定できる．このようにして，相当に能率の良い診断手順を簡単に作ることができる．（保坂　務）

E-56. オートマトンの一般化された縦続分割

M. Yoeli: Generaliged Cascade Decomposition of Automata〔J. ACM, Vol. 12, No. 3, July, 1965 pp. 411～422.

mealy 形の不完全順序機械と無限オートマシンへの一般化とを，代数的な手法を用いて論じている．

代数的には，準同形写像理論程度のものを使用する．まず，二つの集合間の binary relation ρ を定義し，かつ，semi automaton A とこれに対応する automaton $\hat{A}$ を考える．このとき，ρ は状態の集合間を結ぶ，方向を持った枝と考えることができる．

次に，二つの automata $\hat{A}$, $\hat{B}$ 間に，$\hat{B}$ が $\hat{A}$ を cover するという関係 $B \geq A$ を定義し，以下，与えられた finite automaton $\hat{A}$ に対して，これと同等な $\hat{A}$ を cover する最少状態数の automaton をいくつかの機械の cascade connection によって実現するという問題を論じている．

数学的に厳密にというのが，著者の一つの目的であるため，非常に整った論文であり，無駄がない．

大略の論旨は，まず，Aの状態の集合の非空な部分集合の族で，decomposition π を定義し，A に admissible な π, A の動作を部分的に表現する semi automaton π-factor を定義する．

次に $\hat{B}$ が $\hat{A}$ を cover する条件を示し，$\hat{A}$ の reduction $\hat{F}$ を定義してその存在を論じたのち，automaton $\hat{C}$, $\hat{D}$ 間の product $\hat{C}\circ\hat{D}$ によって，$\hat{A}$ とそのπを与えられたとき，$\hat{C}\circ\hat{D} \geq \hat{A}$ なる semi automaton C の存在を論じている．

最後に，一般に与えられた finite automaton に対して不用意な reduction を用いて cascade decomposition を実行すると，最適のものを見つけることができない危険性があることを指摘している．そして，ここに述べた一般化された decomposition method は，この危険を小さくはするが完全に取り除くことができないとし，その限界を示している．（村上　国男）

E-57. Turing 機械の諸型式

P.C. Fischer: On Formalisms for Turing Machines〔J. ACM, Vol. 12, No. 4, Oct., 1965 pp. 570～580〕

Turing 機械の諸形式

現在 Turing 機械と呼ばれているオートマトンはその定義が人によって少しずつ異なっている．本論文では種々の Turing 機械の間の相互関係，すなわち，ある Turing 機械を，他の定義の Turing 機械で真似をする場合に必要な状態数の変化などを論じている．ここで議論の対象となっているのは次の5種の機械である．

（1） Turing 機械

元来 Turing により提案されたもので，その動作は次の5字組（q, S, S′, D, q′）で記述される．q, q′ は現在および次の状態，S, S′ 現在および次のテープ上の記号，Dはヘッドの移動方向で，L, R または N である．

（2） Post 機械

Post および Davis により用いられた表現で，動作が4字組（q, S, X, q′）で記述される Xは L, R, N または S′ である．

（3） D機械

次の4字組で記述される．（q, S, X, D）ここで D は L, R, N, X は q′ または S′ のいずれか．

（4） S機械

4字組（q, S, S′, X）Xは L, R, N または q′ のいずれか．

（5） U機械

3字組（q, S, X）X は L, R, N, q′, S′ のいずれか．

いずれの機械も Turing 機械と同じ能力をもつこと，万能機械の最小状態数などが論ぜられている．

（戸田　巌）

F-58. 階層構造に結合された記憶装置間のプログラム，データの流れ

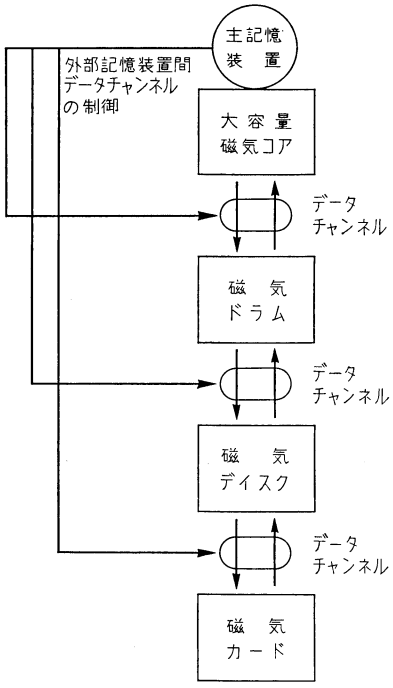
A. Opler: Dynamic Flow of Programs and Data Through Hierarchical Storage〔IFIP, Cong. 1965, pp. 273～276〕

従来の計算機では，磁気ドラム，磁気ディスク，磁気テープ，大容量コアなどの外部記憶装置は，それぞれ中央処理装置にチャンネルを通して直接接続されて

いる．データ移送速度，容量，機械的待ち時間，アクセスタイムなどの異なった種々の記憶装置をそれぞれチャンネルを通して中央処理装置に接続して，それぞれの性質に適したデータ（プログラムも含む）の記録に用いていたわけであるが，仮りにN個の異なった外部記憶装置があって，データの種類もN種あり，その読み書きされる回数，大きさ等の性質がそれぞれの外部記憶装置の性質と一致していれば，従来の中央へ集中する構成方法が適しているわけである．

ところが実際には，読み書きの要求は使用できる記憶装置の性質と1対1に対応することはまれで，中央処理装置と外部記憶装置との間の整合がとれず，中央処理装置は不必要な待合わせを行なうことが多かった．

このような中央処理装置の無駄を省くために，ここではデータチャンネルを中央へ集中するという従来の考え方をやめて，外部記憶装置相互間のデータチャンネルを設け，中央処理装置で必要なデータは，あらかじめ外部記憶装置間のデータチャネルにより，最も速い外部記憶装置へ移しておき，中央処理装置にはもっぱらこの最も速い外部記憶装置とデータのやりとりを行なわせて，待合わせ時間を極力少なくするという方法を提案している．



第1図　階層構造に結合された外部記憶装置

すなわち，磁気テープなどのシーケンシャル・アクセスのものは除いて，イメディエイト・アクセス，ダイレクト・アクセスできる，大容量磁気コア，磁気ドラム，磁気ディスク，磁気カード等を，図に示す如くその容量，移送速度，アクセスタイムの順に階層的にデータチャンネルにより接続し，これらのデータチャンネルのデータ移送は，別に中央処理装置から制御するというものである．このシステムでは，すべてのデータ，プログラムは最下位のレベルの記憶装置に入っており，主記憶装置にはコントロール・プログラムのみが入ることになる．途中の段階の記憶装置には，中央の処理の進みぐあいによって，種々のデータ，またはプログラムが，ダイナミックに入る．

このような，外部記憶装置相互間のデータチャンネル，およびその制御方式を開発することは多くの問題点があって容易ではないが，これが実現されれば，外部記憶装置の性質と，それに入れるデータの性質との不整合という問題はほとんど解決されることになるであろう．　（関　栄四郎）

F-59. 固定プログラム対自己変更型プログラム

J. Nievergelt: Fixed versus Selfmodifying Program〔Computer J. Vol. 8, No. 3, 1965, pp. 244～245〕

実行中に自分自身を変更することを許すプログラムと，許さない固定プログラムを比較して，前者にできて後者にはできない問題があるかという曖昧な問に対しては，問題をもっと明確に定義しなければ答えることができない．Elgot と Robinson（J. ACM Vol. 11, No. 4）はプログラムを構成している命令が，あるクラスに属していれば自己変更型プログラムがより有力（より多くの函数が計算できる）であることを示した．この論文では問題を Turing Machine の観点から論じて逆結果を出している．すなわち，いかなる自己変更型プログラムに対しても，それと同じ仕事（データに対して同じ変換）を行なう固定プログラムが存在する．

問題の設定．次の四つの条件，

（1）　始期状態qの万能 Turing machine U.

（2）　UのテープをTとし，最初ブランクでない升は有限個しかない．

（3）　任意に定められたテープの部分をPとする．

（4） 最初に読まれている升はPの部分にある．

が与えられた場合下記の性質を持ったテープ T^* が存在するか？

（i） T^* にある有限な部分 P^* があり，最初は T^* の P^* 以外の部分と，TのP以外の部分とが等しい．

（ii） P^* の部分に升 s^* があり，もしテープTについて(q, s) から計算を始めて止るならば，T^* について (q, s^*) から始めてやはり止る．そのときTのP以外の部分と，T^* の P^* 以外の部分とは等しくなる．

（iii） T^* について計算する場合 P^* 部分の記号は変更しない．

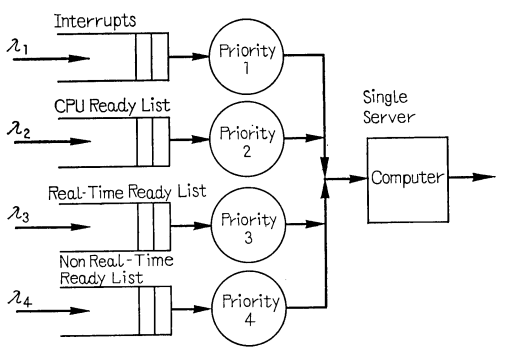
この問題を計算機に対応させると，Uが制御回路，テープが記憶装置，最初プログラムPとデータ（TのP以外の部分）が入っている．計算の結果はP以外の部分（データ領域）に書かれた記号である．Pの部分は計算中変更してもよい．問題は同じ結果を与える，変更を許さないプログラム P^* が存在するかということである．

以下その構成方法が述べられているが，考え方は P^* にはPの他に，最初Pをデータ部にコピーして，そこへジャンプして計算を行ない，結果だけを残して P^* 以外は消してしまう機能をもった補助ルーチンがあればよい．ここで重要な点はプログラムとデータを区別しないであつかうことである．　（相馬　嵩）

F-60. 実時間多重プログラミングの解析

W. Chang and D.J. Wong: Analysis of Real Time Multi programming〔J. ACM, Vol. 12, No. 4, Oct., 1964 pp. 581～588〕

通信網からの実時間メッセージを多重プログラムで処理する場合，必要とされる処理能力について，待ち行列を中心に数学的解析を行なう一つの方法を述べている．図は多重処理における入力と CPU との関係を表わしたものであり，この解析における簡単なモデルである．Interrupts は，通信チャンネルからのメッセージの到着または補助記憶装置への CPU からの要求が完了したときに発生する．その Interrupts の頻度を表わす．λ_2 は補助記憶装置からの応答の頻度，λ_3 は，通信チャンネルから到着するメッセージの頻度，λ_4 は，非実時間処理要求の頻度を表わす．各リストには，優先度がついていて，小さい番号に高い優先度が与えられる．待ち合わせ列の処理順序は，一つの処理を終えてしまってから，次に待っている要求のうち，最高優先度の先頭にある項目を次に処理する．



上の簡単なモデルを一般化して表現する．

各リストの入力頻度を $\lambda_1, \lambda_2, \cdots\cdots, \lambda_n$ とし，それぞれランダム入力とする．$H_k(x)$: k 番目の priority のものに対するサービス時間の分布．$\alpha_k^{(j)}$: $H_k(x)$ の j 番目のモーメント．$W_k^{(1)}$, $W_k^{(2)}$: k 番目の priority をもった項目の待ち時間の第 1，第 2 モーメント（λ_i, $\alpha_i^{(j)}$ によって表現される）．このとき，サービスを含めた待ち行列の平均長 $L_k^{(1)}$ と第 2 モーメント $L_k^{(2)}$ は，近似的に

$$L_k^{(1)}=\lambda_k W_k^{(1)}+\lambda_k\alpha_k^{(1)}$$

$$L_k^{(2)}=\lambda_k^2 W_k^{(2)}+2\lambda_k^2 W_k^{(1)}\alpha_k^{(1)}+\lambda_k^2\nu_k^{(2)}$$

$H_k(x)$ と $\alpha_k^{(j)}$ を求めるには，P_j を j 番目の項目の発生する確率，$f_j(x)$ を j 番目の形の項目の処理時間が x である確率，とするとき

$$H_k(x)=\int_0^x \sum_j P_j f_j(y)dy$$

$$\alpha_k^{(1)}=\int_0^\infty x dH_k(x)$$

$$\alpha_k^{(2)}=\int_0^\infty x^2 dH_k(x)$$

$$\alpha_k^{(3)}=\int_0^\infty x^3 dH_k(x)$$

k 番目の優先順位をもったリストに要求されるコアメモリの平均値 $C_k^{(1)}$ と第 2 モーメント $C_k^{(2)}$ は，$B_k^{(1)}$, $B_k^{(2)}$ を k 番目の優先度をもった項目に要求されるコアの大きさの平均値と，第 2 モーメントとすれば，

$$C_k^{(1)}=L_k^{(1)}B_k^{(1)}$$

$$C_k^{(2)}=L_k^{(2)}[B_k^{(1)}]^2+L_k^{(1)}B_k^{(2)}$$

と表わされる．

なお，これらの式の具体的応用例は示してなく，補助記憶装置チャネルへの待ち合わせは考えていない．

（石野　福弥）

F-61. 拡張された汎用シミュレータ (GPSS III)

H. Herscovitch and T.H. Schneider: GPSS Ⅲ-an Expanded General Purpose Simulator [IBM Sys. J. Vol. 4, No. 3, 1965 pp. 174～183].

シミュレータとしての GPSS は，問題やシミュレートされるシステムを，このフローチャート言語で表現しさえすれば，計算機自体について知らなくてもよいという使い方の容易さをもっている．

シミュレートされるシステムの要素を，実体(Entity) というもので，論理的に表現する．実体は4種に分類される．

動的……トランザクションと呼ばれ，移動するものの単位を示し，属性を有し得る．

設備的……システムの設備の要素を表わす．

統計的……システムの動きを測るもの．待ち合わせ行列と，必要な統計的整理の表である．

演算的……ブロックと呼ばれ，システムの論理を表わし，トランザクションに指示を与える．GPSS Ⅲ は 36 種のブロックを有する．

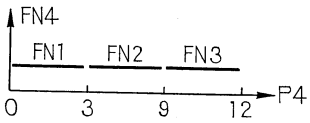
シミュレートされるシステムの入力は，control card と definition card によって与えられる．トランザクションは，動作を受ける時間ごとに処理され，その結果の情報としては，システムを通ったトランザクションの量，その所要時間，各設備の使われ方，各位置で生ずる待ち合わせ行列の長さ等々がある．

GPSS Ⅲ では，storage の制限が緩和された．それぞれのブロックに対する個数の制限が除かれて，全体での制限が存在するだけになった．したがって各ブロックなどの数を融通することができるようになった．

トランザクションの属性が0から100まで，自由につけられるようになり，core storage を他に利用できるようになった．

またトランザクションのある時刻における状態は，Current Event, Future Event, Interrupt chain のいずれかに属していたが，新たに user chain を設け，リンク，アンリンクブロックでトランザクションを，シミュレータの使用者の意のままに用いられるようになった．したがって FIFO (先入先出)，LIFO (後入先出) 自由に，トランザクションの流れる順番を制御できるようになった．

さらに，新属性値関数 (new Attribute Valued function) を使えるようになり (第1図)，多重の間節指定が，1次元で可能になって，便利になった．



第1図

他に改良された点を例挙すれば次のようになる．

(イ) 変更，付加ブロック

QUEUE: DEPART, SPLIT: GATHER, HELP, TRANSFER, EXECUTE: CHANGE

(ロ) その他

トランザクションとブロックの関係が，任意の時点でわかる．リスト関数の追加．　　(木村　幸男)