数式向き計算機について*

——二つの着想——

野　崎　昭　弘**

序

これは，私が電気通信研究所において発表した旧論文[1]の一部である．テーマは，情報処理学会全国大会（昭 38，12 月）において東大の金山裕氏がされた講演 'Algol Oriented Computer' と類似のものであるが，実際に取り扱われている問題は少し異なっているので，興味をもたれる方もあるかと思い，投稿することにした．まず，ここに述べるのは上記論文[1]のうち旧知の事柄の解説的な部分である．多少とも独自の点は別に述べる予定であるが，これが何かの参考になるとすれば，まことに幸いである．

問　題

本稿で扱うのは，数式翻訳（compile）を容易にするための設計方式* についての，二つの基本的な着想と，その効果である．

* このような発想法については，8)，9) の冒頭に詳しく述べられている．

数式翻訳の，最も基本的な作業としては，まず次の二つが挙げられよう．

Ⅰ．順序決定　演算の実行順序の決定．乗除算が加減算に優先するという慣習や，括弧があればその範囲などが吟味の対象になる．

Ⅱ．中間処理の問題　計算の進行に伴って必要な，中間結果の待避や復旧．

ほかに簡単化（optimise）や，一般の番地割付の問題などがあることはよく知られているとおりである．

Ⅱ，はさらに分ければ次の二つになる

$Ⅱ_1$　中間結果の待避あるいは復旧を，実行すべき時点の決定．

$Ⅱ_2$　一時記憶場所の割付．

$Ⅱ_1$ は具体的手順としてはⅠに含められると考えられる．$Ⅱ_2$が中間処理固有の，Ⅰや$Ⅱ_1$からは独立の問題である．以下に解説するのは，$Ⅱ_2$ に関する，ひろい意味での“データ転送制御方式”上の，二つの着想と，その効果である．

* Formula Oriented Computers, by Akihiro Nozaki (Tokyo University)

** 東京大学教養学部基礎科学科

1．Pawlack の方法

一時記憶のための番地割付についていえば，もし中間結果の**待避**（かきこみ）順序と**復旧**（よみだし）順序との間に特定の関係があれば，それを計算機に run-time に自動処理させることができるであろう．

実際，Pawlack が提案しているように，'計算された順序に利用する' 数式（後述）の場合には，first-in-first-out の（トンネル式）記憶装置を設けることによって，かきこみ・よみだしが自動化される．

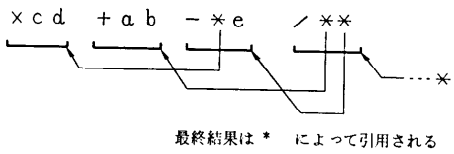
任意の数式は，'計算された順序に利用する' 数式——以下 Pawlack の標準形という——に変形できる．たとえば

$$(a+b)/(c\times d-e) \qquad (1)$$

は次のように変形される．

$$\times cd\ +ab\ -*e\ /** \qquad (2)$$

ここで '*' は中間結果の利用を示す．式は左から読まれ，一つの演算記号と二つの文字または星との組が単位（command[6]）である．そしてたとえば ×cd は 'c×d' を意味する．1 単位が実行されるたびに，その結果はトンネル式記憶装置にかきこまれる．* はそれをよみだすための記号で，左から n 番目にあらわれる * は，左から n 番目の単位（命令）の結果を引用する（**第 1 図**）．



第 1 図

Pawlack は，その標準形を通常の単一アドレス方式の機械語に翻訳するアルゴリスムを示しているが，(cf. 7)，別に標準形そのものを（ある意味で）機械語としてもつ計算機の方式を示している[5,6]．それは容易に推察されるように，トンネル式記憶装置をもつ，2 アドレス方式の計算機である．ただし，二つのアドレ

スは二つの被演算数を示し，演算結果はトンネル式記憶装置にかきこまれる（累算器は存在しない）．指令部分をアセンブラ言語風にかけば，(2) 式

×cd +ab −*e /**

のプログラムは次のようになる．

MPY/c, d.
ADD/a, b.
SUB/*, e.
DIV/*, *.

'*'は特別の番地（ゼロまたは特別のビットに1を与えるなど）により指示される．

このように，特別の設計方式をもってすれば，標準形から機械語への翻訳は，何らの労力を要さない．順序をかえずに連続的に，1対1の対応がつけられるのであるから，それらの間には活字と音声ほどの差しかなく，内容的には同一の言語と考えてよい．そこで，順序決定の問題の実質となるのは，通常の数式を Pawlack の標準形になおすところまでである．その変形のアルゴリズムは 1), 7), 10) に述べられているが，容易であるとはいいがたい．

要 約 Pawlack の方法の特徴は，中間処理に関して，「計算順序と利用順序の一致」を利用することである．順序決定手順としては，Pawlack の標準形が媒体になる．この標準形を機械語としてもつ計算機の設計方式を示すことができる．それはトンネル式記憶装置を内蔵する，2アドレス方式の計算機である．

トンネル式記憶装置そのものは，二つの計数器の設置によって実現される（詳細は文献 1), 6) に譲る）．標準形（実は何種類か発表されている）の厳密な定義は，すべて 1) あるいは原論文 2)〜7) に譲る．

Pawlack の方法の欠点は(ここに述べなかったが)，標準形への変形のアルゴリスムが簡単でないことである．それは実は「計算順序と利用順序との一致」という着想そのものに起因する．そのような標準形は，言語一般の自然的構造に反している．このことは次節の所論から明らかになるであろう．

2. Burroughs の方法

Pawlack のように，各演算ごとに結果を異った場所にかきこむ必要は必ずしもない．普通の計算機のように，一つの固定した累算器をもっていて，それが被演算数の一つをあらわせるようになっていれば，

a+b+c+d

のように各演算結果が直ちに次の被演算数になっている場合，待避（かきこみ）の必要は全くない．もちろん

a×b+c×d

のような場合，a×b と c×d とを空間的に独立に行なわないかぎり，どちらかの結果の待避・復旧はせざるをえない．このように問題は残るけれども，累算器の考えを全く捨ててしまうのは明らかに損である．

さて，そこで待避の方法を考えよう．Pawlack と違った着想を求めるとすれば，当然「計算順序と逆の順序で利用する」方式が思いつかれよう．この前提によれば，last-in-first-out の（穴蔵式）記憶装置によって，かきこみ・よみだしが自動化される．

利用順序に関する前提をみたす標準形として，G. Lukasiewicz の考案した普通 Polish notation とよばれるものがある．簡単のため，ここではそれを P-式とよぶことにしよう．任意の数式は，P式に変形できる．たとえば，

(a×b+c×d)/(a−d)+b×c　　(3)

は，P式

ab×cd×+ad−/bc×+　　(4)

となる．式は左から読まれ，一つの演算記号はすぐ左の二つの'変数，または演算結果'に対して働く．たとえば最初の×はすぐ左の a, b に対して働き，a×b を作る．中央にある＋は二つの演算結果，(ab×) および (cd×) に対して働き，

(a×b)+(c×d) を作る．

/ はすぐ左の−の演算結果 (ad−) と，その次に左の＋の演算結果に働くので，

(ab×cd×+)/(ad−)

すなわち

(a×b+c×d)/(a−d)

が得られることになる．

P式は，形式的には次のようにして定義される．

I. 基本記号 (syllable, 8))

（1） 変　数 a, b, ……, x, y, ……
（2） 演算記号 + − × /

II. 定 義

（1） 任意の変数はP式である．
（2） α, β があるP式で，Δ がある演算記号であれば，$\alpha\beta\Delta$ もまたP式である．

(演算記号をふやすこと，また unary operator を導入することなどは，簡単な修正によって行なわれる)

Burroughs の電子計算機 B-5000 システムは，この一風かわった言語を，前に述べた意味で機械語とし

てもつ計算機である．そのために，回路としては被演算数をおくための二つのレジスタ A, B と，穴蔵式記憶装置Sとがある．これらを働かす基本的な命令として，一つの転送命令と四つの演算命令（＋－×/）が用意されている（ほかにもあるが，今は触れない）．

これらの命令の効果は，すでによく知られていることと思う．しかし比較の便宜上，その着想の由来を尋ねるという形で簡単な説明を加えておきたい．もちろん，B-5000 の設計者たちの実際の思索過程はたどるすべもないが，思想的原型として自然に連想される論文を挙げるとすれば，K. Samelson と F.L. Bauer の論文[11]がそれである．それは通常の数式の compile 手順を記述した論文であるが，そこで中間言語として採用されている命令系は，B-5000 の命令系と酷似している．すなわち穴蔵式記憶装置（numbers cellar）H があって，それに関連する次の2種類の命令がある．

（1） 転送命令 K_α；変数 α をHにかきこむ．

（2） 演算命令 K_ξ（ξ は ＋－×/ のいずれか）；Hから二つの数値をよみだし，演算 ξ を施し，結果をHにかきこむ．

たとえば，a+b×c×(d+e) を，

1) の方法で翻訳してえられるプログラムは，次のようになる（命令実行後のHの内容を右側に併記した）．

K_a	H={a}
K_b	H={a, b}
K_c	H={a, b, c}
$K_\times$	H={a, b×c}
K_d	H={a, b×c, d}
K_e	H={a, b×c, d, e}
K_+	H={a, b×c, d+e}
$K_\times$	H={a, b×c×(d+e)}
K_+	H={a+b×c×(d+e)}

このプログラムは，記法をかえれば，すなわちKを省いて横がきにすれば，P式そのものである．かりに K_α, K_ξ をそのまま機械語としてもつ計算機の設計を試みるとしよう．その際難点は，K_ξ を行なうたびに二つの数値をHからよみださなければならないことである．そこで特定のレジスタ A, B に，Hのいわゆる**最高位**（highest position）のレジスタ二つを代行させることにすれば，B-5000 の命令系が生まれる．すなわち，(1) A, B, S は全体として一つの穴蔵式記憶装置Hのように働く．(2) 演算命令は，A, B の内容に対して働き，結果はBにおかれる（cf. 9）．

A, B のいずれか（または両方）に被演算数がないときは（そのことはインディケータに表示される），Sからのよみだしが起り，A（または/および B）は自動的に'埋め'られる．これが，'A, B, S は全体として一つのHのように働く'ということの意味である．

このような設計方式をもってすれば，順序決定の問題の実質となるのは，通常の数式を，P式になおすところまでである．そのアルゴリスム11）はに述べられているが，次の二つの作業だけを行なう，きわめて簡潔なものである．

S I 左括弧の前の演算記号は，その括弧の中の演算がすんでから（対応する右括弧が現われてから）実行させるように，'後回し'にする．

S II 優先順位に関する慣習によって省かれている括弧を認識する．

これらの作業は，常識的には'常に必要なもの'と考えられる．通常はこのほかに，左括弧が現われたときに，それまでの計算結果を待避させる手段を講じ，またそのために番地割付を行なう必要がある．これらを省略できる（オブジェクト・プログラムの簡単化を図るためとあれば，省略しなくてもよい）ことは，この方式の利点である．

要　約　Burroughs の方法の特徴は，中間処理に関して，「計算順序と逆の順序で利用する」ことにある．順序決定手順としては，P式が媒体になる．このP式を機械語としてもつ計算機 B-5000 が発表されている．

P式への翻訳は，Pawlack の場合に比べて，極めて簡単である．それは偶然ではなく括弧構造の本質に由来している．括弧というものは，「最後に始まったものが，最初に閉じる」構造原理にしたがっているのである．それで，中間処理のために，穴蔵式記憶装置が適している．穴蔵式記憶装置が，P式への翻訳手順の中でも有効であることを知れば，'トンネル式'がいかに不自然であるかは推察されよう．'トンネル式'は一つのチャンネルにおけるバッファとしては有効であるが，言語処理には応用が狭い．

穴蔵式記憶装置が有効かどうかは，処理の対象となる情報が，括弧構造をもっているか否かできまる．たとえば，サブルーチンのエントリとリンクとの関係も常に標準のリンクを行なうことを前提にすれば，括弧構造が成立する．recursive procedure に穴蔵式記憶装置が応用される所以である．

自然の言語においても括弧構造が発見されることはよく知られている．'入れ子型構造'[12]，'矢印の交叉

の禁止'[13] 等々の言葉で述べられている事柄も，いずれも，'括弧構造' という言葉で説明されるのである．その構造（そしてそれのみ）に即して，穴蔵式記憶装置が応用され得ることは，いうまでもあるまい．

参考文献

1) 野崎昭弘：自動プログラミングI；電気通信研究所成果報告，第1741号（1962）
2) Z. Pawlack: Organization of the address-free digital computer for calculating simple arithmetic expressions; Bull. Acad. Polon. Sci. Sér. sci. techn. Vol. 8, 8 (1960)
3) 前　同：On the Application of the Rule of Substitution in the Organization of an Address-free Computer; Bull. Acad. Polon. Vol. 8, 11-12 (1960)
4) 前　同：On the Realization of Recursive Schemes in the Address-free Computer; 前同
5) 前　同：Organization of Address-free Computer with separate memory of partial Results; Bull. Acad. Polon. Vol. 9, 2 (1961)
6) 前　同：New Conception of Two-address Computer; Bull. Acad. Polon. Vol. 9, 5 (1961)
7) 前　同：Some Remarks on Automatic Programming of Arithmetical Foamulae; 前同.
8) W. Lonergan, P. King: Design of the B-5000 system; Datamation (1961-5)
9) R.S. Barton: A New Approach to the Functional Design of a Digital Computer; Proc. of the WJCC (1961)
10) 藤野精一：On some application of the push down store; 数理科学研究班第2回プログラミングシンポジウム報告集（1961）
11) K. Samelson, F.L. Bauer: Sequential Formula Translation Com. ACM(1960-2)
12) 時枝誠記：日本文法（口語篇）;岩波全書
13) 奥津敬一郎，成田正雄：日本文法への数字の応用；数理科学（1964～2）

（昭和39年2月10日受付）