

解 説

大型数式処理システムと利用の実態†



村 尾 裕 一††

1. はじめに

数式処理は、計算機の記号処理への応用における代表的な成功例であり、すでにいくつかのシステムが実用段階にある。実際、近年我が国でも数式処理システムの研究・利用の機運がたかまり、その応用分野も物理学・工学などの科学技術計算を初めとして、経済学さらには教育へと、数値計算同様、解析的算法を用いる多くの分野で役立てられている¹⁾。記号処理という、計算機科学・情報科学以外ではあまりなじみのない分野の成果が、比較的早い時期から広く受け入れられたことの理由として、以下の二点が挙げられるであろう。

(1) 数式処理システムの開発が、計算機科学者ばかりでなく、利用者であるところの物理学者や数学者によっても行われてきたこと。

(2) 数式処理の対象である数式及びその処理操作である演算、さらにそれらの記述法が、なじみやすかつ一般性を有する数学を基礎としており、概念としても整理・統一されていること。

実際の数式処理システムは、その用途により形態は大きく異なる。システムを特徴づけるのは、各種数学演算を実現する種々の機能ももちろんだが、それ以上に

- 式の表現法、
- 演算手順のモデル化、
- なにを自動化し、なにを利用者のコントロールに任せるか、

といった数学的に定式化されていない部分である。そして、これらはシステム全体の処理効率やメモリ使用量とも密接に関連する。

本稿では、これらをふまえた上で多角的な実用性と汎用性を目指した大型数式処理システムを概観し、その機能と設計理念について説明する。昨今、マイクロ

コンピュータ用の簡易なシステムも出現し、流通しているが、その解説は別稿に譲る。数式処理システムはその処理速度とメモリ使用量から、比較的大型の計算機上で実現されてきており、その変遷をたどることにより、大規模ソフトウェアとしての問題点、より高度なシステムへと構築してゆく際の留意点が明らかになるであろう。それらが、我が国で現在開発途上にある諸システムへの一助になれば幸いである。

2. 開発の経緯

数式処理の試みは、計算機の歴史のごく初期から行われ、天文学における長大な式の計算を機械的に行わせることもその一つであった²⁶⁾。現実的なシステムは1960年代から70年代初頭にかけて、数多く開発され、その中には天体力学・一般相対論・量子電磁力学(QED)などの固有の問題を解くために、物理学者により作られたものも少なくない。現在実用に供されているシステムの多くはこの時期に開発されたものである。表-1に現行の数式処理システムを掲げる。これらの中には、応用主体のシステムとして開発され、各種機能を付け加えることにより、汎用的なシステムとして現在に到っているものもある。天体力学の計算を目的とした CAMAL^{8),16)} や、QED 計算を目指した REDUCE²⁵⁾ がその主なところである。その一方で、数式処理システムとしての機能は低くとも、固有の問題における優位性(処理速度・メモリ使用量)から今なお用いられているシステムもある。高エネルギー物理学・QED における ASHMEDAI^{27),28)} や SCHOO-NSCHIP³⁸⁾ がこれにあたる。

70年代に入ると、数式処理の研究は主にアルゴリズムの研究に精力が注がれた。GCD や因数分解のアルゴリズムがその主たる成果だが、それまで知られていた数学的手法が、実際のシステム中にインプリメントしてみると必ずしも実用的でないことがその発端であった。^{80年代に入り、再び新たなシステムがいくつか登場した。いずれも、それまでに得られた確立した数式}

† Current Status of Large-scale Computer Algebra Systems by Hirokazu MURAO (Computer Center, University of Tokyo).
†† 東京大学大型計算機センター

処理技術をふまえており、SMP^{11),12)}及び MAPLE¹⁰⁾は現代の計算機資源に合わせて統合化したシステムで、SCRATCHPAD II²⁶⁾は扱う対象を一気に数学一般に広げ、アルゴリズムをより抽象的な概念を用いて記述できるよう抽象データ型を取り入れたシステムである。これらについては次章で説明する。

3. 汎用数式処理システムの実例

本章では、表-1 に掲げた数式処理システムのうち、筆者が実際に利用可能なシステムを主体にして、各システムを紹介し、その特徴や手法を説明しよう。まず数式処理システムは、その処理形態により、会話型と翻訳実行型に大別される。前者は、式や文が入力されるたびに評価・出力が行われるのに対し、後者ではファイルに用意された式や文からなるプログラムを、専用の内部コードに変換しそれを解釈実行するか、特定の言語に翻訳してそれを実行時ルーチンとリンクして翻訳実行するという形態をとる。翻訳実行型は、'60～'70年代に作られたシステムに多く見られるが、現在ではほとんどが会話型のシステムとなっている。会話型の利点は、式の計算を行うのに、式自身と若干のコマンドを入力する程度で済み、得られた中間式を見な

が柔軟な処理法を指定できることであり、式が多様に表現されうる場合に有用である。

3.1 CAMAL(CAMbridgeALgebra system)^{8),16)}

1960年代の後期、英国ケンブリッジ大において、天文学者 D. Barton らにより開発されたシステムで、天体力学の問題を、フーリエ級数を用いて解析的に解くために作られた²⁾⁻⁶⁾。翻訳実行型のシステムで、プログラムは CAMAL コンパイラにより内部コードに変換され、それを実行時システムが解釈実行する。プログラム中、変数名は1文字と制限され、式の各項の変数の情報が1ワード以内に収まるように工夫されるなど非常にコンパクトなシステムである。また、式の内部表現の書き換えを積極的に行ったり、不要になった式を陽に指定できるなど、メモリの効率的利用が配慮されている。式中、小文字の変数は式を構成する不定元としての変数、大文字の変数は値を保持するプログラム言語としての変数と区別されるため、式の評価法はごく単純である。乗算、べき乗算については展開が必ず行われ、有理式は負のべきで表される。

このように、式の計算はきわめて単純な規則のみを用いて行われる古典的な小型のシステムであるが、フーリエ級数

表-1 主要な数式処理システム

処 理 系	記 述 言 語	作 成 者	用途・その他
ASHMEDAI	FORTRAN	米・CMU M. J. Levine	QED 計算
CAMAL	BCPL	英・ケンブリッジ大 D. Barton 他	天体力学、汎用
CAYLEY	FORTRAN	豪・シドニー大 J. J. Cannon	群論
MACSYMA (VAXIMA)	MacLisp 系 (Franz Lisp)	米・MIT (UCB) J. Moses (R. Fateman) 他	汎用 会話型 LISP マシン
MAPLE	BCPL family の マクロ言語	加・Waterloo 大	汎用だが、開発途上
REDUCE	Standard Lisp, PSL	米・A. C. Hearn 現在 RAND Corp.	汎用 世界中で使用可
SAC-1, SAC-2	FORTRAN	米・Wisconsin 大 G. E. Collins 他	数式処理アルゴリズム検証用 多項式・有理式が主だが汎用
SCHOONSCHIP	CDC-7600 アセンブラ	CERN	高エネルギー物理学
SCRATCHPAD/II SCRATCHPAD/I	LISP/VM MODLISP LISP/360	IBM, T. J. Watson 研	汎用 抽象データ型、アルゴリズム 記述
SHEEP	不明	Stockholm 大 I. Frick	一般相対論、テンソル計算
SMP	C	米・CALTECH S. Wolfram 他	作成者自身は理論物理学者 汎用 会話型 機能豊富

$$\alpha = \sum P_{i,j_1, \dots} (x_0, x_1, \dots) \begin{cases} \sin(j_0 y_0 + j_1 y_1 + \dots) \\ \cos(j_0 y_0 + j_1 y_1 + \dots) \\ e^{i(j_0 y_0 + j_1 y_1 + \dots)} \end{cases}$$

を扱う F あるいは E システムでは、角度成分を表す変数 (y_i) とそうでない変数 (x_i) とは明確に区別され、角度成分への一般式の代入を行うと自動的に三角関数の展開が行われたり、微積分の特殊な処理など、特殊用途には強力なシステムである。

3.2 MACSYMA^{17), 29)}

米国 MIT における Project MAC の成果として作られた。現在機能的に最も強力な会話型の数式処理システムである。システム全体は、Lisp と MACSYMA が定義する簡単な言語で書かれ、図-1 のように機能ごとにモジュール化され、それらが階層的かつ有機的に構成されている。多数のモジュール群において、数式のデータ構造はなん種類か用意され、特に多項式・有理式については、処理の効率化のために正準表現（後述）も可能である。データ型としては、通常の数や式のほかにリストや任意の要素を持つ行列、等式、さらに、ラムダ式による関数という型も扱う。式の評価法も、変数はその値でおきかえ（通常再評価は行わない）、関数は実引数がラムダ変数に束縛され、変数は動的なスコープを持つなど、かなり明確に定義されている。その豊富な数学関数・機能に加えて、部分式の指定、数値への評価、 $\forall$ による評価の抑止、式の 2 次元出力 (Π ; Σ ; $\lim$ なども)、グラフ出力等々きめ

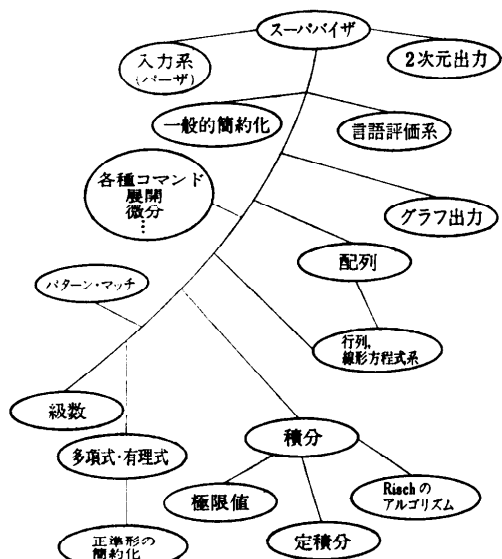


図-1 MACSYMA のモジュール構成

細かな、あらゆるユーザの要求を満たすように整備されているが、強いていえば、ユーザにとっては、分厚いマニュアルから必要な情報をとり出すことが困難な点が欠点である。

3.3 REDUCE²⁵⁾

A. C. Hearn が米国スタンフォード大在任中（後にユタ大学、現在ランド社）に QED 計算に用いるのに開発した汎用システムで、全世界で最も広く利用されている。日本においても、多数の大学・研究機関などに導入されており、応用例も数多い¹⁾。REDUCE は、Lisp としての最低限の機能を簡潔にまとめた Standard Lisp を用いて、Algol 風の syntax sugar を取り入れた RLisp により記述され、配布時にはそのすべてのソースが提供される。このため、移植性が非常に高く、可搬型の計算機からスーパー・コンピュータに到るまで数多くの計算機上で、また各種の Lisp 上で稼動している。REDUCE は、このように利用者に開かれた立場をとっているため、利用者自身による改良・パッケージの開発も盛んであり、実際に現在の REDUCE のソースの 3 分の 2 近くをしめる因数分解及び不定積分のパッケージは英国ケンブリッジ大において^{31), 34)}、また任意多倍長浮動小数 (Big Float) パッケージは本紙別稿を執筆している佐々木によるものである³⁷⁾。今後も、代数方程式の解法を初めとして新たなパッケージの追加も予定されており、現在最も活発に開発が進められているシステムのひとつである。数式処理システムとしては、行列や QED 計算も含めた各種機能が簡潔にまとめられていて、会話型の扱いやすいシステムである。しかし、各種データ型や変数の束縛・スコープ、パターンマッチと関数の適用など、MACSYMA のようにきれいにまとめられておらず、プログラミング言語としての性格が不明確である。たとえば、プロシジャ内におけるパターンマッチや配列・行列などのデータ型の宣言は大域的なものと同一になってしまい、その結果プロシジャの引数として配列・行列をわたして使うことは困難である。これらは、入力系のパーザが、前述の Lisp を Algol 風の構文で記述するための RLisp と共用されており、それが入力を Lisp S 式へと変換する目的で作られているためである。このため、実用的なパッケージはすべて、通常の数式計算を行うモードではなく、symbolic モードと呼ばれる Lisp に等価な RLisp で書かれている。

3.4 SMP^{11), 12), 39)}

SMP は、1979～81 年にかけて、米国 Caltech の理

論物理学者 C. A. Cole, S. Wolfram らにより開発された会話型の数式処理システムである。高エネルギー物理学における理論式の計算を行うことを目的としていたが、既存のシステムでは、第2章で述べたように、処理効率・メモリの利用効率に優れた専用システム (SCHOONSCHIP など) では汎用的な数学演算機能が不足し、MACSYMA などの汎用高機能システムでは現実に現れる巨大な式を扱うには処理能力が十分でないことを理由に、まったく新たな、その双方を可能とするシステムを実現するために新たに開発が始められた。システムは言語 C により記述され、それに加えて特殊関数の変形規則までも含んだライブラリ群を有する。データ型や評価法は MACSYMA にならぬ、きちんと整理され定義されている。特に、配列やユーザ定義の関数が projection という概念に統一されているのが特徴的である。機能としては MACSYMA と同程度のものを有しているが、C 言語で記述されているため Lisp による場合と異なり、ユーザ定義の手続きの評価は常に解釈実行により行われ、システムの拡張性という点で若干劣る。

3.5 MAPLE¹⁰⁾

MAPLE はカナダ Waterloo 大学において 1980 年より開発が行われている会話型の汎用のシステムである。多数のユーザが同時に (教育も含めて) 利用可能なように、コンパクトであること、かつ高度な機能を有することを目指している。これを実現するために、四則演算などのみからなる核部分を小型化し、付加的な機能は大幅な効率低下を招かない限り、固有の言語で記述されるライブラリとなっている。記述言語としては移植性を考慮して、BCPL 系 (C を含む) の言語を対象としたマクロ言語を用いている。移植性を理由に、前述のライブラリや手続きなどの機械語など機械依存なかたちへの翻訳は行っていない。特徴的なのは、ハッシングを多用していることで、式及び部分式の簡約化において同一式の再計算を避けるための表探索にも用いられている。

以上はなんらかの形で配布の行われているシステムである。これまでのシステムが計算 (calculation) の能力を基準として評価されていたのに対し、次の節で説明する Scratchpad/II は、計算能力に加えて、第一に数式処理において拡大する数学対象に対し抽象データ型を導入し、第二に、それに伴いアルゴリズムの記述を数学に近い形で行えるようにしている点で興味深い。

3.6 Scratchpad/I¹⁹⁾, Scratchpad/II²⁰⁾

Scratchpad は 1970 年頃から IBM T. J. Watson 研究所において研究開発が行われてきた会話型システムで、先代の Scratchpad/I は、MACSYMA, REDUCE などから各種機能をライブラリとして譲り受け、その上に独自の言語と評価系 (evaluator) を構築するという方法で作られた。

より高度かつ広範囲な数式の処理やユーザによる数式の表現法の細かな指定を実現するには、数学の概念を精密かつ正確に表現する方法が必要になる。たとえば、

$$\frac{2}{5}x + \left(\frac{1}{5} - 4i\right)$$

は x の (係数を虚数とする) 多項式の表現であるが、虚数としてみた場合、 x が実変数であれば、

$$\left(\frac{2}{5}x + \frac{1}{5}\right) - 4i$$

という表現が適切であろうし、通分して

$$\frac{2x+1-20i}{5}$$

と表現することも可能である。これらは、ガウス整数を係数にもつ x の多項式、 x の多項式を係数とするガウス整数、ガウス整数を係数とする x の多項式の商体、における表現である。つまり、式の表現は、その式がいかなる集合に属するかによって変わってくる。さらに、そのおのおのの表現・集合に対応して、演算や変形の仕方も異なってくる。また一方では、ユークリッドの互除法のように、整数・多項式などの GCD の計算に共通して用いるアルゴリズムがあるが、インプリメントに際しては、generic なオペレーションを用いてそのアルゴリズムを記述するのが自然である。これを実現するのが抽象データ型 (abstract data type) である。以上の考え方をもとに、数式処理システム記述のために開発された言語が MODLISP である¹⁵⁾。MODLISP では、

●mode: 式の表現法を指定する、

●category: (数学的) オペレーションと性質により代数を規定する、

(例) 全順序集合という category は
オペレーション: 全順序く、値は真か偽、
性質: $x < y$ でなくかつ $y < x$ でないならば $x = y$ のような集合、

●domain: category において実際のオペレーションが手続きとして定義されたもの (上の例ではくとして整数の大小関係、有理数の大小関係など)、

という三つの概念を導入し, Lisp のリストで表されたデータ (=式) に, その属する domain に応じて, 四則演算などの基本的な演算に対する Lisp 関数を用いて, 評価が行われる. MODLISP は, Lisp 上に作られた会話型のインタープリタで, 変数の型 (domain) の宣言はコンパイラへのオプションである. この MODLISP を記述言語として, 上記三つの概念を用いてより一般的な数式の演算・計算及びアルゴリズムの記述を行うためのシステムが Scratchpad/II である. Scratchpad/II は現在開発途上にあり, 各国の研究者を集めて種々のアルゴリズムの蓄積を行っているという状況にある.

4. 数式処理の実際

数式処理システムの内部でどのような処理が行われているかをまず説明しよう. 入力系により読み込まれた数式は基本的な内部表現に変換される.

$$5x^3 - xy + f(x, y) - 5$$

→(PLUS (TIMES 5 (EXPT x 3))
(TIMES x y -1)
(f x z)
-5)

次にこの内部表現の評価 (evaluation)・簡約化 (simplification) が行われる. 評価とは, 変数のその値での置き換え, 関数のその定義の引数を実評価した結果の適用のことである. 変数の値や関数が未定義であれば, 自身をその値・定義とする (すなわち x の値は式 x , $(f x z)$ の値は $(f x \text{の値 } z \text{の値})$). 多くのシステムでは, 得られた値の式を再度評価し, 変化しなくなるまで繰り返す (MACSYMA では通常一回限り, REDUCE ではスイッチにより切り替え可). ここで PLUS, TIMES などの四則演算及びシステムが既知の関数名 (三角関数, 指数・対数関数など) については固有の簡約化・標準形への変換が行われる (数どうしの四則演算, 0 の加算・乗算, 1 の乗算, 同類項のまとめあげなど). ここで標準形の決め方としていくつかの選択枝がある:

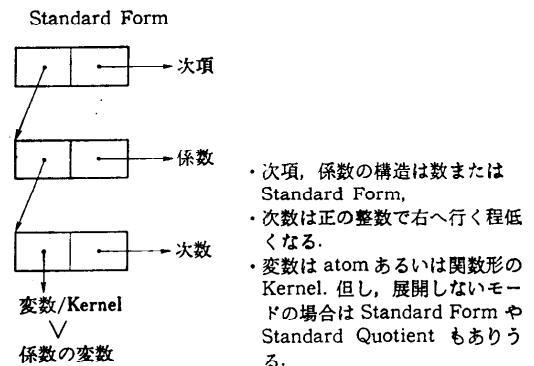
- (a) 積・べき乗は展開するか,
- (b) ひとつの項の中の変数の順序, 和の場合の項どうしの順序,
- (c) 有理式の加減算で通分するか, また, する場合, 分母を最小公倍数とするか,
- (d) 除算において, 除数・被除数の GCD を約すか (結果が有理式),

- (e) 除算において, 商と余りの形に変形するか,
- (f) 初等関数の変形を行うか ($\log 1 \rightarrow 0, e^x \cdot e^y \rightarrow e^{x+y}$ など).

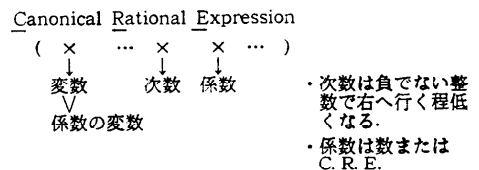
システム全体の性格は, これらの項目をいかに扱うかにより特徴づけられ, システムがより複雑, より汎用性を高めることにより, このような選択枝の数は増加する. 最も単純なものとして, すべての式を分母の最高次の数係数が正の有理式で表し, 乗算・べき乗は可能であれば展開を行い, また変数の順序は辞書式とする方法がある. このような場合, 正準形 (canonical form) と呼ばれる形に式を直しておくで四則演算に対応する式の変形は容易に行われる. 図-2 に REDUCE と MACSYMA における正準表現を示す. ここで, 通常演算を行うのに伴い, 新たなデータ構造が作られていき, 特に乗算・べき乗算の展開ではそのデータが元のデータに比べて極端に大きくなりうることに注意する必要がある.

$$(x+y)^{100} \rightarrow x^{100} + 100x^{99}y + 4950x^{98}y^2 + \dots$$

実際のシステムでは, 上にあげた点について, 応用主体の古典的なシステム (CAMAL など) では上述の



Standard Form は多項式を表現するデータ構造で, 有理式はこれを対とした Standard Quotient で表現される.



C. R. E. は多項式を表現するデータ構造で, 有理式はこれを対である. MACSYMA の場合, 複数の表現法をもち, 級数は C. R. E. で次数を任意の有理数, 係数を有理式とした形になる.

図-2 正準表現 (Canonical representation)

ような単純な方法をとっているが、MACSYMA を初めとする現代的なシステムでは計算時間やメモリが大量に消費される処理は、ユーザが陽に指定しない限り行わないのが普通である（簡単な式の展開も行わない、式の展開は容易だが、逆に因数分解するには複雑な算法が必要である。REDUCE では通常展開・通分を行うが、スイッチにより切り替え可能である）。

評価・簡約化の最終段階として、変数ごと・項ごとといった局所的な評価・簡約化の行われた式に対して、いくつかの項にまたがるなど大域的な情報が必要な簡約化・パターンマッチ ($\sin^2 X + \cos^2 X \rightarrow 1$ など) が行われる。

こうして得られた式は、高度なシステムでは指数を変数の右上においたり、分子・分母を上下に分けて出力するなど2次元出力を行うのだが、この際にも、変数及び関数形の順序（通常われわれは数・変数・関数形の順に書く）、降べき・昇べきの順、負のべき乗の有理式への変換などに従って式の変形が行われる。

5. 数式処理システムの役割と設計

前章において、数式の計算に対応するデータ変形の手順を大まかに説明し、その際のさまざまな不確定要素がシステムを特徴づけることを述べたが、数式処理システムが実用に供される場合、大域的な観点からの全体の設計の良否が問題となる。図-3 に、実際にユーザが利用する場合、選択・判断の基準とする点をまとめてみた。四角で囲まれた四項目がその主な項目であり、互いに直交するわけではなく密接に関連していることは明らかである。これらのうち、処理速度はいかなる場合にも重要であるが、長大な式の頻出する科学技術計算では、既存のシステムから推察できるように、複雑な機能よりは処理速度やメモリ効率が重点項目であろうし、あまり計算機になじみのない教育や数学の分野ではユーザ・インタフェース及び特殊な機能の整備が重要であろう。

逆に設計者・製作者の観点から、いくつかの点について、これらの項目に関連づけて論じてみよう。

(i) 数式のデータ構造

高度で汎用的なシステムにおいては、多種多様な演算及び機能が必要となるが、そのすべてに対応しうるデータ構造を用いることは、不要な情報まで含むことになり、処理速度・メモリ効率の点で望ましくない。かといって機能ごとに個々のデータ構造をとるとしたら、おののに対する四則演算などの基本演算ルーチンが必要となり、システム自体が大きくなりメモリ効率の点で劣るばかりでなく、モジュール間の連絡（極限值と級数展開など）のたびに交換が必要となり処理速度・メモリ効率の劣化を招く。最低限必要なデータ構造を用意し、第4章で述べた不確定要素を定め冗長さや重複がないよう十分なモデル化を行ったうえで基本演算ルーチンを実現するのが最善策であろう。

(ii) 記述言語

会話型の現代的なシステムでは Lisp か C-BCPL などのシステム記述言語かという選択になる。

Lisp の利点・欠点

- メモリ管理の必要がないため記述

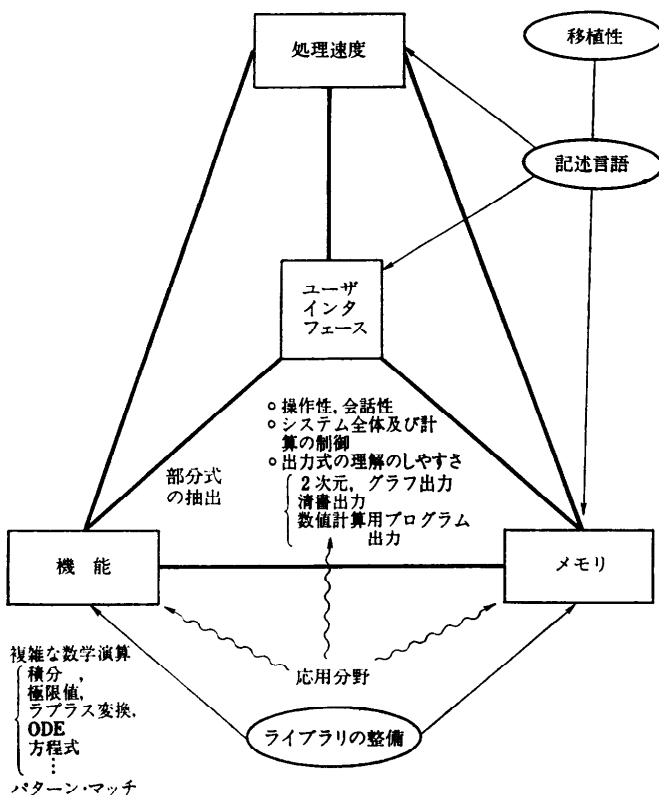


図-3

が容易になるが、不用意にリストを生成しないかは注意が必要である(筆者の観測では、REDUCE 稼動中 CPU 時間の 3 分の 1 がガーベッジコレクション及びモジュールの入出力にあてられていた)。

- 豊富で柔軟なデバッグ環境・対話環境
- 拡張性が高い。ユーザ定義の手続きも Lisp に翻訳してしまえば機械語に翻訳して利用できる。
- 動的なプログラムリンクを行うので、安定したシステムでは非効率を招く。
- 余計な型の検査が行われる。Lisp 上で数式のデータ構造の型を検査した後(当然、Lisp 内でも行う数・ベクトル・リストなどの検査を含みうる)、Lisp システム自身でも同様の検査が行われうる。

C・BCPL などのシステム記述言語の場合

- メモリ管理をシステム記述と同等に行う必要がある。
- 中間言語を用意したとしても、数式処理用の言語を解釈実行せざるをえない。ライブラリにおいて性能の劣化を招く。
- 移植性は記述言語にゆだねられる。
- C 言語を UNIX 上などで用いたとすれば、他のユーティリティが利用可能。

(iii) 移 植 性

記述言語に強く影響されるが、メモリの利用法やユーザインタフェースは OS にも依存する。

たとえば、一般に数式処理は多大の計算時間を必要とするものであるから、複数個のジョブを実行させたり、プロセスの中断・再開が不可能な OS では、中間結果の計算式をファイルにとるなどの機能も必要となる。

(iv) ライブラリ

これは主に機能と関連するが、MAPLE のようにメモリとの兼ね合いで整備し、必要に応じて利用・廃棄が可能であることが望ましい。モジュール間の連絡や使用法などのユーザインタフェースを考慮する必要がある。

(v) マニュアル・ドキュメンテーション

ユーザインタフェースの要となるものであるが、その手法などについて筆者は詳しくないので省略する。

6. まとめと将来の展望

冒頭や各所で述べたことだが、これまでのシステムは機能かあるいは処理速度とメモリ効率かのいずれか一方のみに重点がおかれたものが多かった。しかし、

最近の計算機環境の発展(ソフトウェア・ハードウェアとも)に伴い、その両方(換言すれば質と量)を満足するシステムの実現も可能になっている。実際、Lisp マシンを初めとするワーク・ステーションや、16/32 ビット CPU 搭載のパーソナルコンピュータ上でも既存の汎用的なシステムが稼動しており、実用になるだけの余力を残している。近い将来、ビットマップ・ディスプレイなどの各種機器及びソフトウェアの整備したワーク・ステーションが数式処理システム利用の主流となり、式の出力法やグラフ出力などの点で、優れたマンマシン・インタフェースが現実のものとなるであろうことは想像に難くない。一方で、応用分野を見渡すと、たとえば QED→QCD というように問題の複雑化に伴いより巨大な式の計算が必要となっており、絶対的な処理能力においてワーク・ステーションをはるかにしのぐ大型汎用機上でも数式処理システムが使われなくなることは、まず有りえない。こうしてみると、大型機からパソコンに到るまで、中間式ファイルの移行性やその核部分におけるコマンド及び概念の共通性を有した数式処理システムが将来の望ましい姿であろう。また、Scratchpad/II にみられるように、処理の対象の拡大も今後は必要であろうが、その場合でも、実用においてそのすべてを扱うことは希であろうから、計算機資源を有効利用する設計をしなくてはならない。以上を総合すると、数式処理システムで実用化しうる数学概念を選び出して、それらの関連を整理し、ソフトウェアとしては部品化する研究が必要なのではなかろうか。

参 考 文 献

- 1) 「REDUCE プログラミング」資料第一集、資料第二集、資料第三集、(Mar. 1984), (Mar. 1985), (Mar. 1986).
- 2) Barton, D.: A Scheme for Manipulative Algebra on a Computer, *Comput. J.* Vol. 9, pp. 340-344 (1967).
- 3) Barton, D., Bourne, S.R. and Burgess, C. J.: A Simple Algebra System, *ibid.*, Vol. 11, No. 3, pp. 293-298 (1968).
- 4) Barton, D., Bourne, S.R. and Fitch, J. P.: An Algebra System, *ibid.*, Vol. 13, No. 1, pp. 32-34 (1970).
- 5) Barton, D., Bourne, S.R. and Horton, J. R.: The Structure of the Cambridge Algebra System, *ibid.*, Vol. 13, No. 3, pp. 243-247 (1970).
- 6) Barton, D., Willers, I. M. and Zahar, R. V. M.:

- The Automatic Solution of Systems of Ordinary Differential Equations by the Method of Taylor Series, *Proc. Math. Software* (ed. J. Rice), Academic Press, pp.369-390 (1971).
- 7) Barton, D. and Fitch, J.P.: General Relativity and the Application of Algebraic Manipulation Systems, *Proc. SYMSAM 2* (ed. S.R. Petrick), ACM, pp.343-348 (1971).
 - 8) Bourne, S.R. and Horton, J.R.: The Design of the Cambridge Algebra System, *ibid.*, pp. 134-143 (1971).
 - 9) Brown, W.S. and Hearn, A.C.: Application of Symbolic Mathematical Computations, *Comp. Phys. Comm.*, Vol. 17, pp.207-215 (1979).
 - 10) Char, B.W., Geddes, K.O., Gentleman, W.M. and Gonnet, G.H.: The Design of Maple: A Compact, Portable and Powerful Computer Algebra System, *Computer Algebra-EUROCAL '83* (ed. J.A. van Hulzen), LNCS No. 162, Springer-Verlag, pp. 101-115 (1983).
 - 11) Cole, C.A. and Wolfram, S.: SMP-A Symbolic Manipulation Program, *Proc. SYMSAC '81* (ed. P.S. Wang), ACM, pp. 20-22 (1981).
 - 12) Cole, C.A., Wolfram, S., et al.: SMP-Handbook, version 1, CALTECH (1981).
 - 13) Collins, G.E.: The SAC-1 System: an Introduction and Survey, *Proc. SYMSAM 2* (ed. S.R. Petrick), ACM, pp.144-152 (1971).
 - 14) Collins, G.E.: ALDES and SAC-2 Now Available, *SIGSAM Bulletin* Vol. 14, No. 2, ACM, pp. 19 (1980).
 - 15) Davenport, J.H. and Jenks, R.D.: "MODLISP", *Proc. LISP '80 Conf.* (1980).
 - 16) Fitch, J.P.: CAMAL User's Manual, Univ. of Cambridge, Computing Service (1982).
 - 17) Foderaro, J.K. and Fateman, R.J.: Characterization of VAX Macsyma, *Proc. SYMSAC '81* (ed. P.S. Wang), ACM, pp.14-19 (1981).
 - 18) Frick, I.: SHEEP User's Manual, Univ. of Stockholm (1977).
 - 19) Griesmar, J.H., Jenks, R.D. and Yun, D.Y.Y.: SCRATCHPAD User's Manual, Yorktown Heights: IBM Research Report RA 70 (1975).
 - 20) Griss, M.L. and Hearn, A.C.: A Portable LISP Compiler, *Software-Practice and Experience*, Vol. 11, pp.541-605 (1981).
 - 21) Griss, M.L., Benson, E. and Maguire, G.Q. Jr.: PSL: A Portable Lisp System, *Proc. ACM Symposium on LISP and Functional Programming* (1982).
 - 22) Hearn, A.C.: Computation of Algebraic Properties of Elementary Particle Reactions Using a Digital Computer, *Comm. ACM.*, Vol. 9, pp. 573-577 (1966).
 - 23) Hearn, A.C.: Applications of Symbol Manipulation in Theoretical Physics, *Proc. SYMSAM 2* (ed. S.R. Petrick), ACM, pp.17-21 (1971).
 - 24) Hearn, A.C.: The Personal Algebra Machine, *Proc. IFIP '80*, North Holland, pp. 621-628 (1980).
 - 25) Hearn, A.C.: REDUCE User's Manual, version 3.2, Santa Monica, The Rand Corporation (1985).
 - 26) Jenks, R.D. and Trager, B.M.: A Language for Computational Algebra, *Proc. SYMSAC '81* (ed. P.S. Wang), ACM, pp.6-13 (1981).
 - 27) Levine, M.J. and Roskies, R.: ASHMEDAI and a Large Algebraic Problem, *Proc. SYMSAC '76* (ed. R.D. Jenks), ACM, pp. 359-364 (1976).
 - 28) Levine, M.J. and Roskies, R.: ASHMEDAI, *Proc. Fourth International Colloquium on Advanced Computing Methods in Theoretical Physics* (ed. A. Visconti), pp.70-79 (1977).
 - 29) MACSYMA Reference Manual, version 10, Mathlab group and LCS, M.I.T., (1983).
 - 30) Marti, J.B., Hearn, A.C., Griss, M.L. and Griss, C.: Standard Lisp Report, SIGPLAN Notices, Vol. 14, No. 10, ACM, pp. 48-68 (1979).
 - 31) Moore, P.M.A. and Norman, A.C.: Implementing a Polynomial Factorization and GCD Package, *Proc. SYMSAC '81* (ed. P.S. Wang), ACM, pp.109-116 (1981).
 - 32) Moses, J.: Algebraic Simplification: A Guide for the Perplexed, *Comm. ACM.*, Vol. 14, pp. 527-537 (1971).
 - 33) Norman, A.C.: TAYLOR's User's Manual, Univ. of Cambridge, Computing Service (1973).
 - 34) Norman, A.C. and Moore, P.M.A.: Implementing the New Risch Integration Algorithm, *Proc. Fourth International Colloquium on Advanced Computing Methods in Theoretical Physics* (ed. A. Visconti), (1977).
 - 35) Norman, A.C.: Symbolic and Algebraic Modes in REDUCE, REDUCE-Newsletter No. 3, Univ. of Utah, pp.5-9 (1978).
 - 36) Pavelle, R., Rothstein, M. and Fitch, J.P.: Computer Algebra, *Scientific American*, Dec. (1981)——日本語訳「計算機による代数処理」, サイエンス, (1982年2月).
 - 37) Sasaki, T.: An Arbitrary Precision Real Arithmetic Package in REDUCE, *Symbolic and Algebraic Computation-EUROSAM '79* (ed. E.W. Ng), LNCS No. 72, Springer-Verlag, pp. 358-368 (1979).
 - 38) Strubbe, H.: Manual for SCHOONCHIP, *Comp. Phys. Comm.*, Vol. 8, pp.1-30 (1974).
 - 39) Wolfram, S.: Computer Software in Science and Mathematics, *Scientific American*, Sept. (1984)——日本語訳「科学と数学のソフトウェア」, コンピューター・ソフトウェア, 別冊サイエンス, (1985年4月).

(昭和61年3月4日受付)