記号処理 49－1
（1989. 1. 9）

ＧＡＬ数学公式データベースにおける公式インデックシングと検索法

三枝義典，増永良文，　佐々木建昭
（図書館情報大学），（理化学研究所）

数式処理システム上で自動運用することを目的に数学公式データベースを設計し，国産数式処理システムＧＡＬ(General Algebraic Language/Laboratory) 上にインプリメントした．公式の自動運用の観点からは公式のインデックス法が極めて重要である．本論文では，Ⅰ型，Ⅱ型，Ⅲ型と名付けたインデックス法を提案する．これらのインデックスを併用して用いることにより，精度(precision) の高い検索が行なわれることを幾つかの実例を用いて示す．また，ユーザによる検索に備えて，簡潔かつ実用的な検索文を定めた．この検索文を用いることにより，人間の行なうであろう公式検索のほとんどが実現されることを示す．

INDEXING AND RETRIEVING METHODS FOR FORMULA DATABASE IN JAPANESE COMPUTER ALGEBRA SYSTEM GAL

Yoshinori SAIGUSA[1], Yoshifumi MASUNAGA[1] and Tateaki SASAKI[2]

1) University of Library and Information Science
Tsukuba-shi, Ibaraki 305, Japan
2) The Institute of Physical and Chemical Research
Wako-shi, Saitama 351-01, Japan

ABSTRACT

So as to utilize mathematical fomulas by an algebra system automatically, we designed a database of mathematical formulas and implemented it on Japanese Computer Algebra System GAL(General Algebraic Language/Laboratory). Since there is no standard or reference for designing a formula database, it is necessary for us to find the indexing and retrieving methods that are suited for the automatic utilization of formula database. In this paper, we propose three types of efficient and complemental indexing methods, named type-I, type-II and type-III indices, and useful retrieval statement which simulates human retrieval of formula books. By showing some practical examples, we verify effectiveness and validity of these indices.

1．はじめに

数式処理研究の初期には人工知能的解法（発見的解法）が広く用いられたが，その後は今日に至るまでアルゴリズムによる解法（決定的解法）が研究の主流を占めてきた．こうしたなかにあって，最近ではアルゴリズム偏重に対する反省から発見的解法を見直そうとする動きが活発である．一例として，数式処理システム上での特殊関数の処理が挙げられる．数式中に特殊関数が含まれている場合，現在知られているアルゴリズムでは計算がうまく行かないことが多い．数式処理システム上で特殊関数を含んだ式をも扱えるようにするためには，その簡単化が可能でなければならない．我々がこのような計算を行なう際に，そのよりどころとするのが公式集である．ＲＥＤＵＣＥ[1]やＭＡＣＳＹＭＡ[2]では，ＬＥＴ文やＲＵＬＥコマンドを用いて（公式を簡単化規則として発見的に適用することにより）このような演算を行なうことが可能である．しかしながら，特殊関数は簡単化の規則が複雑であるためにその扱いも難しく，既存の数式処理システムではうまく処理し切れないというのが現状である．そこで，これらの公式をデータベースとして二次記憶装置に蓄えておき，必要に応じて検索するということが考えられる．また，アルゴリズムによって導かれるはずの公式であっても，計算能力や計算時間の関係で簡単化規則として数式処理システムが保持していたほうが良いものも多い．現在，数学公式データベースを保持している数式処理システムはわずかにＳＭＰ[3]のみであるが，その機能は限定されたものであって，アルゴリズムの限界を補う段階までには至っていない．

ＧＡＬ[4](Genaral Algebraic Language/Laboratory) とは，現在我々が開発を進めている（広範囲の数式演算を目的とした）数式処理システムであり，その機能の一部として上記の観点から数学公式データベースの開発を行なってきた．ＧＡＬでは，数学公式データベースの利用方法として２通りの方法を想定している．一つは，通常のデータベースのようにユーザが必要に応じて公式を検索する方法であり，もう一つは，数式処理システムが式の変形の際に独自でデータベースにアクセスして検索する方法（数学公式データベースの自動運用）である．前者については既にその大半がインプリメントを終えており，本報告においても詳しくその機能を解説する．しかしながら，後者についてはＧＡＬのサポート機能が未完成であるために未だインプリメントの段階に至っていない．

数学公式データベースをＧＡＬ上で運用することによって，従来アルゴリズムが苦手としてきた以下に示すような演算の多くが可能になると予想される．

(a) 高度な関数の定積分と不定積分；
(b) 高度な微分方程式の解の導出；
(c) 有限級数や無限級数の有限和表示；
(d) 特殊関数を含む式の自動簡単化．

2．基本的考察

公式の持つ種々の性質やインプリメンテーション上の制約から，我々は数学公式データベースの開発過程で以下に示すような幾つかの問題点に直面した．

(1) 数式処理システムとの整合性

数式処理システムはその多くが記号処理用言語であるLispによって記述されており，ＧＡＬもその例外ではない．Lispにはレコードやブロックといった概念がなく，基本的には二次記憶装置にランダム・アクセスする機能を持たないためにデータの管理が非常に重くなる．

(2) 公式の入力とデータベース内での統一的扱い

公式のなかには複雑な構造をしたものが多数存在し，これらをいかにして機械可読型の表現に変換するか．また，様々な形態の公式をどのようにして統一的に扱うか．

(3) 公式の双方向性の問題

例えば三角関数の公式では，［左辺］＝［右辺］なる公式が［左辺］⇒［右辺］なる書き換え規則としても，［右辺］⇒［左辺］なる書き換え規則としても用いられる．このよ

うな公式を下手に扱うならば，これらの簡単化規則を用いた計算はたちまち無限ループに陥ってしまうだろう．

(4) 検索キーの設定

公式のようなデータに対しては，その検索方法及び検索キーの設定について何の標準も存在しない．したがって，数学公式データベースの設計に当たっては，新たに検索キーを設定し，これが前章で述べた数学公式データベースの２通りの利用方法に対して有効に作用することを保証しなければならない．

上述の問題点を踏まえて，以下では数学公式データベースの設計に必要な概念の導入を行なう．

【定義１】第１種の公式とは，積分や級数など，数式処理における決定的な演算に対して用いられる公式である．第２種の公式とは，三角関数の公式に代表されるような，数式処理における数式の（狭義の）簡単化を行なうために用いられる公式である．

定義１に当てはまらないような公式も現実には存在するが，公式データベースの利用を数式処理システム上に限るのであれば，この大雑把な分類法だけで十分である．

第１種の公式は典型的には以下のような構造をした公式である．

$$\mathrm{Op}(\mathrm{F},\cdots)\Rightarrow[\text{右辺}] \tag{2.1}$$

ここに，“Ｏｐ”とは∫やΣ等の演算子を指し，ＧＡＬ内では“擬関数”として分類されているものである．擬関数とは，計算を実行すれば全く別の数式となるが，計算を実行しないでそのまま表現しても関数と見なしうるものをいう．逆に，最後まで計算を進めても同一関数名の関数か，あるいは全ての引数が数値の場合に限り結果が数値となるような関数をＧＡＬ内では“純関数”と呼び，擬関数とは明確に区別する．純関数の代表的な例としては，三角関数や指数関数等が挙げられる．以下，特に断わらない限り“関数”とは純関数のことを指すものとする．“Ｆ”とは“Ｏｐ”の主引数を指す．例えば“Ｏｐ”が∫である場合，“Ｆ”は被積分関数で，“・・・”は積分変数や積分区間を表わす．検索の観点からみた第１種の公式の特徴は，公式の［左辺］に関する情報が具体的に与えられている場合が多く（“Ｏｐ”や“Ｆ”），その反面［右辺］に関する情報が未知であることである．

第２種の公式とは，上述の問題点(3)で示した双方向性を有する公式のことである．具体的には三角関数の公式等がこれに当たり，典型的には

$$[\text{左辺}]=[\text{右辺}] \tag{2.2}$$

のような構造をもつ．ここで注意すべきことは，(2.2)のような公式は

$$[\text{左辺}]\Rightarrow[\text{右辺}], \tag{2.3}$$

$$[\text{右辺}]\Rightarrow[\text{左辺}] \tag{2.4}$$

なる２つの異なった書き換え規則として用いられるということである．したがって，ＧＡＬでは第２種の公式(2.2)は書き換え規則(2.3)，(2.4)としてデータベース内に格納されなければならない．また，第１種の公式は(2.1)で示したような単項書き換え規則(single-term rewriting rule)しか含まれないのに対し，第２種の公式は多項書き換え規則(multiterm rewriting rule)も含むことが可能である．第２種の公式の［左辺］の（［右辺］の）特定の項を［右辺］に（［左辺］に）移行することによって公式が異なった性格を有するものに変わるならば，このような公式は元の公式とは別のものとして処理されなければならない．例えば第２種の公式

$$\sin^2(x)+\cos^2(x)=1 \tag{2.5}$$

は［左辺］の一方の項を［右辺］に移行することにより，数学公式データベース内では以下に示すような異なった性格を持つ２つの公式としても処理される．

$$\sin^2(x)=1-\cos^2(x) \tag{2.6}$$

$$\cos^2(x)=1-\sin^2(x) \tag{2.7}$$

第２種の公式に対する検索・運用は，以下に示す２つの理由により，第１種の公式に対する検索・運用よりも複雑となる．

(a) 第２種の公式により数式は様々な形へ変形され得るが，どの公式を用いればユーザの望む変換結果が得られるかを前もって決定することは極めて困難である．
(b) 仮に，指定された変換結果を得るために必要な公式が一意に決定可能であるとしても，ユーザが指定できるのは変換結果に対する漠然としたイメージに過ぎない．
【定義２】（キーワード）数式のキーワードとは，数式中に含まれる演算子名（Integ とかSum 等），関数名（sin とかlog 等），超越的数（π等）及び級数のランニング・インデックス（第ｎ項を指定するインデックスｎ）とする．
【定義３】（キーワードのレベル）数式において，他のキーワード（主に演算子名及び関数名）の引数部以外の箇所に表われているキーワードを第１レベルのキーワードと呼ぶ．キーワードＫを第ℓ－１レベルのキーワードとするとき，Ｋの引数部に第１レベルで出現しているキーワードを第ℓレベルのキーワードと呼ぶ．

定義２の解釈には，しばしば慎重な配慮を要求される．例えば，数式 $(x+1)^{\frac{1}{2}}$ では，定義に従えばキーワードは存在しないことになるが，これを $\sqrt{x+1}$ のように表現した場合には“sqrt”がキーワードとなる．このような混乱を避けることは決して簡単ではないが，一つの解決策としてはあらかじめ公式をプリプロセッサにかけておき，全ての公式のフォーマットを統一しておくことが考えられる．

３．数学公式データベースの設計

前章の議論に基づいて，本章では数学公式データベースの設計を行なう．設計に先立って，我々は以下に示すような設計の基本方針を定めた．
①公式データは更新の必要を認めない（このことは，公式がユーザによって簡単に書き換えられる性格のものではないという認識に基づいている）．
②公式は分野毎にまとめ，区分化して管理する．
③実用性を考慮したインデックスを付与する．
④ユーザによる検索では使い勝手を重視し，手検索に近い検索方法を設定する．

ＧＡＬでは，公式を“［左辺］⇒［右辺］（条件）”なる（条件付きの）書き換え規則として扱う．このため，個々の公式を“左辺”，“右辺”，“条件”なる定義域(domain)間の３項関係であると規定するならば，数学公式データベースは以下のような簡単なスキーマによって定義される．

数学公式スキーマ＝（公式番号，左辺，右辺，条件）　　(3.1)

(3.1) において，“公式番号”とはデータベース内の個々の公式を一意に識別するための識別子であり，“左辺”，“右辺”，“条件”とはそれぞれ公式の［左辺］，［右辺］及び（条件）を意味する．(3.1) のようなスキーマを持った関係(relation)は，Lispのデータ構造の一つである連想リスト(association list)を用いることにより，容易に実現できる．

上記③の設計方針にかなうインデックス付けを行なうために，ＧＡＬユーザが公式の検索に際して具体的にどのような情報を有しているか，考察する．この情報はまた，数学公式データベースの自動運用を行なう際に，ユーザが指定した数式あるいは簡単化の方針等からも抽出可能なものでなくてはならない．公式に名前が付けられている場合，ユーザはこの名前を用いて検索を行なうであろう．このような情報は有用ではあるが，ほとんどの公式には名前が付いておらず，また数学公式データベース自動運用の見地からも，公式の名前だけによる検索では全く不十分であることは明らかである．ユーザが公式集を用いて数式の簡単化を行なう場合，ユーザはこの数式と［左辺］がパターンマッチング可能な公式を検索するであろう．ところが，そのような公式は一般に多数個存在するわけであるから，結局ユーザの有する情報は公式の［左辺］に対する漠然としたイメージに過ぎない．数式のイメージとは，“数式内の目印となる語（キーワード），および数式の大雑把な構造”と言いかえることができるから，これらの性質で数式をインデックス化することが我々の目的となる．これは，以下のようにＩ型とⅡ型のインデックス[5),6)]として達成さ

れる．

Ｉ型のインデックスとは，数式中のキーワードとそのレベルにより数式をインデックス化するもので，以下のように定められる．

【Ｉ型のインデックス】

(1) 公式を“［左辺］⇒［右辺］（条件）”なる書き換え規則と見なす．公式が第１種の公式であるならば［左辺］＝Ｏｐ（Ｆ，・・・）とし，第２種の公式であるならば［左辺］＝Ｆとする．

(2) $\{f_1,\cdots,f_m\}$ をＦ内の第１レベルのキーワードの集合とし，$\{g_1,\cdots,g_n\}$ をＦ内の第２レベルのキーワードの集合とする．以下，Ｆ内にキーワードが出現しなくなるレベルまで同様の操作を行なう．

(3) このとき，Ｉ型のインデックスを以下のように定義する．

$$((f_1,\cdots,f_m),(g_1,\cdots,g_n),\cdots)$$

【例 1.1】

第１種の公式 $\int e^{-x^2}dx = \mathrm{erf}(x)$ について考える．$F = e^{-x^2}$ であるから，Ｉ型のインデックスは，((exp)) となる．

【例 1.2】

第２種の公式 $\arcsin(\sqrt{2}\ \sin(x)) + \arcsin(\sqrt{\cos(2x)}) = \pi/2$ WITH $0 \leqq x \leqq \pi/4$ について考える．$F = \arcsin(\sqrt{2}\ \sin(x)) + \arcsin(\sqrt{\cos(2x)})$ であるから，第１レベルのキーワードは“arcsin”であり，第２レベルのキーワードは“ sin”，“sqrt”，第３レベルのキーワードは“ cos”である．したがって，Ｉ型のインデックスは次のようになる．((arcsin), (sin, sqrt), (cos))

第２種の公式の検索に際して，公式の［左辺］と［右辺］を比較したときのキーワードの次数や全次数(#DEG)，項数(#TERM)の増減を指定することは，公式を書き換え規則として適用したときの与式の変形の方向を指定することに等しい．第１種の公式の検索に対しても，公式の［左辺］のキーワードの次数や全次数(#DEG)及び項数(#TERM)が明らかになれば，よりきめ細かな検索が可能となろう．このような観点から，公式の（次数と項数に関する）性質をインデックスとしたものをⅡ型のインデックスと呼ぶ．

【Ⅱ型のインデックス】

(1) 与えられた数式Ｆにおいて，Ｆ内の第１レベルのキーワードの集合を $\{K_1,\cdots,K_m\}$ とする．

(2) Ｆを全次数(#DEG)，項数(#TERM)及びＦ内のキーワードの次数 $(\deg(K_i),\ i=1,\cdots,m)$ によって特徴付ける（定義は以下に与える）．

(3) 公式を“［左辺］⇒［右辺］（条件）”なる書き換え規則と見なす．公式が第１種の公式であるならば，［左辺］＝Ｏｐ（Ｆ，・・・）とし，Ⅱ型のインデックスを (2)で与えられた特徴量によって定義する．公式が第２種の公式であるならば，［左辺］と［右辺］それぞれをＦとして (2)で与えられた特徴量を求め、その増減関係をⅡ型のインデックスとする．増減関係の記述方法は次のようなものとする．［左辺］から［右辺］へ向かうとき，特徴量が増加している場合を“up”，変わらない場合を“same”，減少している場合を“down”と記述し，特に特徴量が［右辺］において０になる場合を“none”と記述する．

<u>全次数(#DEG)，項数(#TERM)及びキーワードの次数のカウント規則</u>：

単項 $v_1^{d_1}v_2^{d_2}\cdots v_m^{d_m}f_1^{e_1}f_2^{e_2}\cdots f_n^{e_n}$ $(d_i>0,\ e_j>0,\ i=1,\cdots,m,\ j=1,\cdots,n)$ において，$v_1,v_2,\cdots,v_m$ を変数，$f_1,f_2,\cdots,f_n$ をキーワードとする．このとき，$\#DEG = \sum_{i=1}^{m} d_i + \sum_{j=1}^{n} e_j$，#TERM = 1である．また，キーワードの次数は $\deg(f_j) = e_j$ によって与えられる．一般に，多項式 $T_1+T_2+\cdots+T_n$ $(T_i, i=1,\cdots,n$ は単項）において，#TERM = nであり，$\#DEG = \max\{\#DEG(T_i)\mid i=1,\cdots,n\}$ である．しかしながら，実際の数式のなかにはこのような単純な

カウント規則だけでは不十分なもの，あるいは扱えないようなものも多数存在する．以下に特殊な場合におけるカウント規則を与える．以下，Ｆは自明（trivial)でない（＝キーワードを含んでいる）数式とする．

(a) F^n（ｎは数）の時，#DEG = #DEG(Ｆ) × ｎ，#TERM = #TERM(Ｆ)とする．また，キーワードについてはＦ内のキーワードの次数をそのままカウントする．

(b) $\sqrt{F}$ のような数式は，#DEG = #DEG(Ｆ) × ½，#TERM 及びキーワードの次数については(a) と同様とする．

(c) ｎ × Ｍ（ただし，Ｍは多項式，ｎは数）のような数式に対しては，#DEG = #DEG(Ｍ)，#TERM = #TERM(Ｍ)とする．キーワードについてはＭ内のキーワードの次数をそのままカウントする．

(d) $\frac{N}{D}$ のような数式については，Ｄが自明な数式の場合（キーワードを含まない場合）には#DEG = #DEG(Ｎ)，#TERM = #TERM(Ｎ) とする．Ｄが自明でない場合，#DEG = #DEG(Ｎ)+ #DEG(Ｄ)，#TERM = #TERM(Ｎ) + #TERM(Ｄ) とする．いずれの場合も，キーワードの次数についてはＮ及びＤ内のキーワードの次数をそのままカウントする．

(e) $X^1+X^2+\cdots+X^n$（ただしｎは不定）のような数式については，#DEGと#TERM が一意に与えられない．このような数式に対しては#DEG = “ＩＮＤＥＦ”，#TERM = “ＩＮＤＥＦ”とする．また，キーワードについてもf^n（ただしｆはキーワード，ｎは不定）であるとき，deg(f) = “ＩＮＤＥＦ”である．

(f) $\Sigma_r F^{ar+t}$（ａは数係数，ｒは級数のランニング・インデックス，ｔは数式）のような級数式については，#DEG = #DEG(Ｆ) × ａ，#TERM = #TERM(Ｆ)とする．キーワードの次数については(a) と同様とする．

【例 2.1】

第１種の公式 $\int e^{-x^2}dx = \mathrm{erf}(x)$について考える．公式の［左辺］の主引数は$e^{-x^2}$であるから，この公式は{#DEG = １，#TERM = １，deg(exp)= １} として特徴付けられる．従って，Ⅱ型のインデックスは (#DEG =１，#TERM = １，exp = １) となる．

【例 2.2】

第２種の公式 $\sin(A+B)\sin(A-B) = \cos^2(B) - \cos^2(A)$ について考える．公式の［左辺］と［右辺］は，それぞれ以下のように特徴付けられる．

{#DEG = 2, #TERM = 1, deg(cos) = 0, deg(sin) = 2}
{#DEG = 2, #TERM = 2, deg(cos) = 2, deg(sin) = 0}

したがって，Ⅱ型のインデックスは次のようになる．

(#DEG = same, #TERM = up, cos = up, sin = none)

全次数(#DEG)及び項数(#TERM) のカウント規則を決定する際に重要なことは，公式の表現方法に関係なくインデックスが一意的に決定されなければならないということである．例えば検索要求が，

$$\sin(A)\sin(B) = -\tfrac{1}{2}[\cos(A+B) - \cos(A-B)] \qquad (3.2)$$

なる第２種の公式を検索することである場合，(3.2) はカウント規則(c) に従えば次のようにインデックス付けされる．

(#DEG = down, #TERM = up, sin = none, cos = up) (3.3)

ところが，カウント規則(c) に従わずに［右辺］を単項と見なすならば，(3.2) は次のようにインデックス付けされる．

(#DEG = down, #TERM = same, sin = none, cos = up) (3.4)

ユーザは公式(3.2) の［右辺］を単項とは見ず，２項式と見なすであろう．ところが，(3.2) に対して(3.4) のようなインデックス付けを行なったのではこの公式は検索されないことになる．

上記の２種類のインデックス法は，ユーザによる検索に対してはもちろん，ＧＡＬが数学公式データベースを自動運用する際にも有効であると考えられる．

我々は個々の公式に付与された名前も有用な情報であると考え，Ⅲ型のインデックスを以下のように定義した．

【Ⅲ型のインデックス】

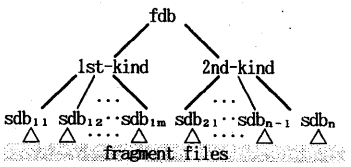
任意の公式Ｆにおいて，もしＦに特定の名前が付けられているならばＦをこの名前によって特徴付け，これをⅢ型のインデックスとする．

【例 3.1】

第１種の公式 $\prod_{n=1}^{\infty}(1-x^n)^3=\sum_{n=0}^{\infty}(-1)^n(2n+1)x^{n(n+1)/2}$ について考える．この公式はJacobiの公式であり，データベースへの入力の際にこの名前を付与されて格納される．このとき，Ⅲ型のインデックスは (Jacobi) となる．

4．数学公式データベースの構成と検索

前章で与えた設計の基本方針②に従って，ＧＡＬでは数学公式データベースを図4.1に示すようなものとした．



(図 4.1)

図において，定義１に与えたごとく数学公式データベース（ｆｄｂ）は第１種(1st-kind)と第２種(2nd-kind)の２つの公式集合に大別される．これらの公式集合は意味的に更に細かな公式集合（サブデータベース：ｓｄｂ）に分けられる．サブデータベースは積分や三角関数などの分野毎に作成する．前章で定義した数学公式スキーマはサブデータベースのレベルで設計した．サブデータベースは，更にフラグメント・ファイルと名付けた幾つかのファイルに分割して，２次記憶装置内に格納する．フラグメント・ファイルは，Ｉ型のインデックスを参照しながら，類似度の高い公式を同一のファイルへ格納するという方法で作成する．このことによって，２次記憶へアクセスする負荷を軽減することが可能となるわけである．個々の公式は，Lispの前置表現に変換されて格納される．また，前章で定義したインデックスは，検索に便利なようにサブデータベース毎にまとめ，逆ファイル(inverted file)構造に再構成されて格納される．これらのファイルの作成は，ＧＡＬのＦＤＢパッケージ内にある数学公式データベース構築モジュールが行なう．

ＧＡＬではシステムが発する検索要求に加えて，ユーザからも直接に公式の検索が可能である．これらの検索はＦＤＢパッケージ内にある数学公式データベース検索モジュールが行なう．ユーザからの検索は検索文を用いてなされる．検索文は以下のような簡単な形式に定めた．

FIND ＜数式パターン＞ WITH ＜パターン＆検索条件＞ ABOUT ＜固有名詞の列＞；
（注１）WITH節と ABOUT節は省略しても良い．
（注２）WITHはWHERE と書いても良い．

＜数式パターン＞とは，ユーザがイメージした公式の［左辺］の描像であり，変数としてパターン変数を含み得ることを除けば通常の数式そのものである．パターン変数とは，任意の数式とパターンマッチング可能な変数であり，ＧＡＬでは変数の先頭に“@”を付けることによって表わす．パターン変数の用法は単純であり，＜数式パターン＞の用法も数式を扱うことになれたユーザであれば容易に使いこなせるものである[7]．例えば，

(SIN(@X)**@N + COS(@X)**@N)*@Y (4.1)

は＜数式パターン＞であり，以下のいずれの数式ともパターンマッチング可能である．

$$e^x\{\sin^3(x) + \cos^3(x)\} \cdots @X=x,\ @N=3,\ @Y=e^x \qquad (4.2)$$

$$\sin(x + y) + \cos(x + y) \cdots @X=x + y,\ @N=1,\ @Y=1 \qquad (4.3)$$

$$\frac{\sin(z) + \cos(z)}{\sin(z) - \cos(z)} \cdots @X=z,\ @N=1,\ @Y=\frac{1}{\sin(z) - \cos(z)} \qquad (4.4)$$

上の例からも明らかなように，パターン変数の使用によってユーザは公式に対する曖昧なイメージを簡単に記述することが可能となるわけである．

＜パターン＆検索条件＞とは“＜パターン条件＞＆＜検索条件＞”のことであり，これらはＧＡＬパーザによって＜パターン条件＞と＜検索条件＞に分割される．＜パターン条件＞はＧＡＬ内では＜数式パターン＞に条件を課すときに用いられる．例えば，上の例で(4.1)に対し“WITH DENOM(@Y)=1”（DENOMは数式の分母を取り出す関数）なる条件を課せば，(4.2)，(4.3)とはパターンマッチング可能であるが(4.4)とはパターンマッチングしなくなる．＜検索条件＞は，第１種の公式の検索に際して指定されることはほとんど無く，主に第２種の公式の検索に際して（与式の簡単化の方向を明らかにするために）指定される．具体的にはⅡ型のインデックスの作成で用いた“up”，“same”，“down”，“none”なる用語を用いて，公式の［左辺］と［右辺］の特徴量の増減関係を指定する．$K=\{\#DEG, \#TERM, f_1, \cdots, f_m\}$ ($f_1, \cdots, f_m$ はキーワード）とするとき，$k_j \in K(j=1, \cdots, m+2)$の特徴量が［左辺］から［右辺］へ向かって増加している場合を“up($k_j, \cdots$)”，同じ場合を“same($k_j, \cdots$)”，減少している場合を“down($k_j, \cdots$)”，無くなる場合を“none($k_j, \cdots$)”と表わす．“none”は“down”の特殊な場合であるから，我々が“down($k_j, \cdots$)”と指定した場合には“none($k_j, \cdots$)”の意味も含めて考えるのが普通である．このような包含関係は，検索モジュールが当然考慮して検索してくれる．

例えば，＜検索条件＞は次のように表わす．

down(SIN, COS) AND NOT up(#DEG, #TERM) (4.5)

＜固有名詞の列＞には次の２つの意味がある．

(i) 本章で述べたサブデータベースを特定する（ユーザがあらかじめ検索領域を限定するときに用いる）．

(ii)公式に付けられた名前を用いて検索を行なう時に用いる．

ユーザによって与えられたFIND文（あるいはシステムによって与えられたFINDコマンド）の検索要求を実行するために，検索モジュールは以下に示すような手順に従って検索を行なう．

step1: 与えられた検索要求が陽(explicit)にサブデータベースを特定しているならば，検索キューにこのサブデータベース名を登録する．そうでなければ，＜数式パターン＞あるいは＜検索条件＞中に含まれるキーワードを用いてサブデータベースを特定し（この場合サブデータベースの候補が複数個選ばれることもある），得られたサブデータベース名を検索キューに登録する．

step2: 検索キューの先頭のサブデータベース名を取り出し，これに対応するインデックス・テーブルを主記憶中に読み込む．

step3: ＜数式パターン＞からＩ型のインデックスを取り出す．その結果が空(null)でなければ，このインデックスに対応する公式を検索する．そうでなければ，step4 へ移る．検索したい公式が第１種の公式であれば，＜数式パターン＞からⅡ型のインデックスを取り出し，このインデックスに対応する公式のみを選択する．ここまでの操作で公式が検索されなかった場合はstep6 へ移る．

step4: 検索要求が＜検索条件＞を含んでいるならば，この条件に当てはまる公式のみを選択する．＜固有名詞の列＞に公式名が指定されているならば，Ⅲ型のインデックスを参照して対応する公式のみをふるいにかける．ここまでの操作で，公式が検索されなかった場合はstep6 へ移る．

step5: 検索された公式の［左辺］と＜数式パターン＞との厳密なパターンマッチングを

行なう．このとき＜パターン条件＞が指定されているならば，この条件を用いて最終的に必要な公式を厳選する．検索された公式をシステムに渡す．

step6: 検索キューに，次に検索すべきサブデータベース名があればstep2 へ移る．そうでなければ検索終了．

検索文を用いた検索の実例を図4.2 に示す（この操作ではstep5 を省略している）

```
FIND SUM(SIN(R*@X), R=1, N) ;                                    (4.6)
  ☞ (121, 160, 164, 168)

FIND SUM((-1)**N/N!, N=0, #INF) ;                                (4.7)
  ☞ (411, 485)
```

（図 4.2）

図4.2 は第１種の公式に対する検索の例であり，検索結果を公式番号によって表わしている．検索結果がどのような公式であるかを図4.3 に示す．図4.3aは(4.6) に対する検索結果を，図4.3bは(4.7) に対する検索結果をそれぞれ表わしており，簡単のために公式の［右辺］を省略している．この例では，極めて高い精度(precision) で検索が行なわれていることが理解できる．一般に，第１種の公式に対する検索では検索要求に見合った公式が，漏れなく低ノイズで検索できる．しかしながら，第２種の公式に対する検索では，＜検索条件＞の指定方法によって検索結果にかなりのばらつきが生じることも事実である．このような問題をインデックス付けのレベルで解消することは難しく，step5 におけるパターンマッチャの機能に頼らざるを得ない．しかしながら，第２種の公式は構造が簡単であるためパターンマッチングに要する時間も短く，検索時間にはさほど影響を与えないであろう．

121	SUM(SIN(@R*#PI/@N),@R=1,@N-1) = . . .
160	SUM(SIN(@R*@X),@R=1,@N) = . . .
164	SUM(SIN((2*@R-1)*@X),@R=1,@N) = . . .
168	SUM(SIN(@X+(@R-1)*@T),@R=1,@N) = . . .

（図 4.3a）

411	SUM((-1)**(@N-1)/(@N-1)!,@N=1,#INF) = . . .
485	SUM((-1)**(@N-1)*2**(2*@N-2)*@X**(4*@N-2)/(4*@N-2)!,@N=1,#INF) = . . .

（図 4.3b）

５．おわりに

本研究によって，数式処理システム上での数学公式データベースの構成と機能をほぼ明らかにすることができたと考えている．しかしながら，数学公式データベースは単独では大した意味を持ちえず，数式処理システムと結合し不定積分や微分方程式，数式の自動簡単化などに応用して初めて大きな意味を持ってくる．今後は，数学公式データベースを運用するためのモジュールを開発し，数学公式データベースの自動運用を行なうための研究を進めなければならないだろう．

謝辞

本研究は，倉田奨励金“特殊関数処理システムの研究”（昭和57年度），文部省科学研究費“数学公式データベースに基づく数式計算エキスパートシステムの研究”（昭和60～61年度），および“総合的数式処理システムの研究”（昭和62～63年度）の援助を受けて行なわれた．関係各方面に謝意を表する．

参考文献

1)Hearn,A.C.. REDUCE user's manual version 3.2. the Rand Corporation, 1986.
2)the MATHLAB Group. MACSYMA reference manual version 9. Lab. Computer science, MIT, 1977.
3)Cole,C.A., Wolfram,A., et al.. SMP hand book version 1. CALTEC,1981.
4)佐々木建昭．元吉文男．国産数式処理システムＧＡＬ（デモ用資料）．数式処理通信．Vol.4, No.1, p6-8(1986).
5)佐々木建昭，増永良文，三枝義典，阿部昭博，元吉文男，佐々木睦子．数式処理システムＧＡＬにおける数学公式データベース．第29回プログラミングシンポジウム報告集．情報処理学会編．箱根，1988，情報処理学会．東京，1988，p167-173.
6)佐々木建昭，三枝義典．数式処理システムＧＡＬにおける数学公式データベース．第37回（昭和63年後期）全国大会講演論文集（Ｉ）．情報処理学会編．京都，1988，情報処理学会．東京，1988，p437-438.
7)Sasaki,T.. Simplification of Algebraic Expression by Multiterm Rewriting Rules. Proceedings of SYMSAC'86. p115-120(1986).