

数式処理システムにおける知的対話環境の改善

大阪電気通信大学工学部

対馬勝英 (K a t h u h i d e

T h u s h i m a)

1. 序

数理科学者のパーソナルな研究ツールとして、数式処理システムの構築を行うことは、筆者の永年の夢であった。

本論文は、それを行うために必要な幾つかの基本的アイデアについて述べ、そのアイデアに基づいたシステム構築の報告を行う。

また既に文献 1 に詳しく述べた様に、数式処理システムの上に推論機構を構築することで、数理的、物理的な問題解決を行うことはチャレンジングな試みである。本論文に展開した幾つかアイデアは、最終的には『空海』と呼ばれる物理学上の法則の認知、発見システムのサブモジュールの位置を占めるものである。

2. S D I F

R E D U C E, m u - M A T H 等の数式処理システムは、整式を扱うことに主眼を置いて設計されており、オペレータ、微係数などの直接的な取扱いは行えない。例えば、

$$\frac{d}{dx} (X u + u_x) = u + X u_x + u_{xx} \quad 1)$$

の表記法をそのままでは、数式処理システム上では扱えない。
線型、非線型の微分方程式の数理的構造を調べるためには、 u
 u_x, u_{xx} 等の微係数がシステム上では扱え、且つ生成されるこ
とが必要である。

また、1)の形での表記を保ちつつ変換、関数変換を実行
する機能が必要とされる。

微係数を直接扱うこのアルゴリズムは、通常の数式微分パ
ッケージ *DIF* を一部手直しすることで容易に実現できる。こ
のアルゴリズムを「SDIF」と名付けた。(Symbolic Diff.)

以下にSDIFの基本部分のリストを示す。

```
FUNCTION SDIF (EX1 INDET)
  WHEN EX1 = INDET 1 EXIT
  WHEN NUMBER (EX1) 0 EXIT
  WHEN ATOM (EX1) CHPL (EX1 INDET) EXIT
  {
    FUNCTION CHPL (EEX1 EEX2)
      EVAL (COMPRESS (LIST (EEX1 EEX2)))
    ENDFUN#
```

筆者は、`mu-MATH` を拡張して種々の機能を拡張、
付加する試みを行ったが、付録4にその概要を示した。

この拡張は

- a) `REDUCE` の持つ機能の付加
- b) 特殊な機能
- c) 環境としてのツール
- d) `SDIF`

の4つに分けられる。現存する数式処理システムは「環境」
としての対話能力に弱点を持つ。この強化をはかったものが
c) であるが、これの発展については展望の項を見られたい。
b) は特殊なものであり、`OECU-MATH` の特徴の一つ

となっている。

第6行に見られる様に、文字列の `compress` により、システム内部では U, UX 等の *compress* した文字列がそのまま生成され、また認知される構造をとっている。

実行例を以下に示す。

?SDIF (U*X, X) ;

① U+UX*X

?SDIF (A*UX+UXX, X) ;

① A*UXX+UXXX

?SDIF (2*UX+UXX, Y) ;

① 2*UXY+UXXY

SDIF アルゴリズムを利用すると、従来の解析的な手法（アルゴリズム）が、数式処理システム上でそのままの形で実現でき、整式のみしか扱えぬためにアルゴリズムを整式向けに変換する煩わしさより解放される。一例として、微分方程式の中級数解を「SDIF」を用いて求めてみよう。

3. 微分方程式の中級数解

$$u'' = f(x, u, u') \quad 2)$$

のあるとき、その中級数解は

$$u(x) = \sum_{i=0}^{\infty} a_i x^i = \sum_{n=0}^{\infty} \frac{u^{(n)}(0)}{n!} x^n \quad 3)$$

である。

```
KAI (-U, 11, X0, U0, UX0)
(U0 - U0*X^2/2 + U0*X^4/24 - U0*X^6/720 + U0*X^8/40320 - U0*X^
10/3628800 + UX0*X - UX0*X^3/6 + UX0*X^5/120 - UX0*X^7/5040 +
UX0*X^9/362880)
```

(1図) $\frac{d^2u}{dx^2} = -u$ の巾級数解

SDIF アルゴリズムでは $\frac{df}{dx}$, $\frac{d^2f}{dx^2}$, ... が $u, ux, uxx, \dots$ を用いて表される。2) と 5), 6) 等を用いると、3) は $u\phi(u(0))$ のこと), $ux\phi$, x のみを用いた整式となり、記号係数の巾級数を得る。 $u\phi, ux\phi$ に数値を代入すると、数値係数の巾級数となる。この実行例を 1 図に示した。(mu-MATH 版を用いたシステムの例である。)

この巾級数を求めるアルゴリズムは、2) から 6) の整式をそのまま記述することにより得られる。この様に、数式処理システムを用いるための特別な「発想の転換」(アルゴリズムの変換)を必要としない点に、SDIF の長所がある。

なお、作成したパッケージ「SOL」は、任意の階数の非線型常微分方程式に対しても巾級数解を生成できる。また、初期条件を整式としても数値としても設定できる。

u が既知である場合には、 u, ux, uxx 等を具体的に評価する必要の生じることがある。OECU-MATH では例えば

```
? U: X*X+3*X*Y ;
```

```
? A: GDIF (UX) ;
```

により、 ux の具体的な整式としての形を A に代入することが

できる。この様に整式レベルでの取扱いと微係数レベルでの取扱いを、状況に応じて使い分けることができる。

また、微分方程式の解のデータベースを作成する場合に、この記法はデータベースのコンパクトな格納と明確な認知し易い表記、迅速な探索を許す長所を持っている。

4. S D I F の拡張

S D I F システムは単に微係数を扱うというレベルに止まるものでなく、付録3に示した様にオペラを代数的に扱うレベルにまで拡張することができる。

従来の数式処理システムは1～3のレベルの機能を持つが、4～6迄の拡張を行うことで始めて数理科学者の発想のレベルに達する。

以下に6のレベルの探索を行う実行例を示す。

```
? AA:=SAYOU(1+DX+DX^2,U);
```

```
@ U + UX + UXX
```

$$\left(1 + \frac{d}{dx} + \frac{d^2}{dx^2}\right) u$$

$$u + u_x + u_{xx}$$

```
? OPERATOR(U,AA);
```

```
@OP--> 1 + DX + DX^2
```

Uに何を作用させるか AA
即ち、 $u + u_x + u_{xx}$ になるか

```
0
```

```
OPERATOR---> 1 + DX + DX^2
```

そのオペラは $1 + \frac{d}{dx} + \left(\frac{d}{dx}\right)^2$ である。

このオペレータ同定機能は、多数の数式の上で数理的な探索を動的に行う際に、決定的な役割を果たす。これが、後に述べる5章の戦略Iに対応する。

5. 巾級数の同定

3.で述べた様に、SOLより微分方程式の解が巾級数として得られるが、これらを特定の関数に一意的に同定することは非常に難しい。また同定ができぬ場合に、似た性質を持つ関数を示唆することも容易ではない。

この問題を解決する方式として二つの戦略が考えられる。

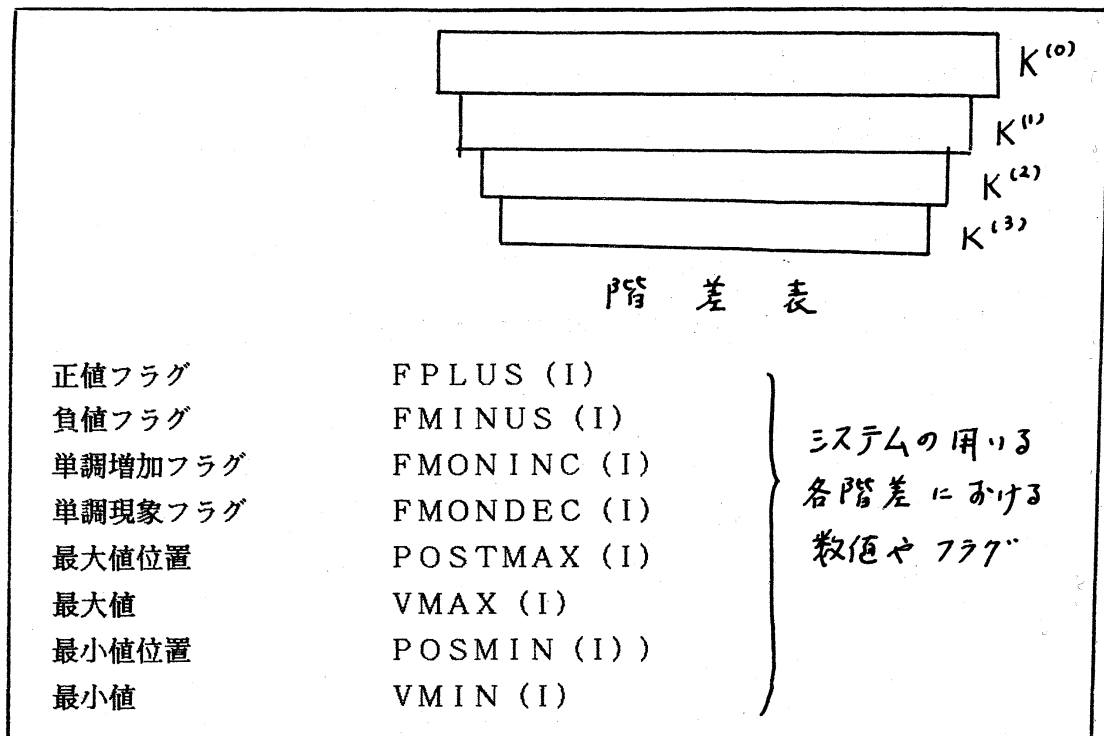
- I) 整式に対するオペレーション（微分、積分等）とその逆操作により解候補を絞りこむ。
- II) 巾級数を数値化して数値的構造を見る。

I) については現在開発中である。II) については筆者の開発したデータ構造を認知する人工知能システムCSOLDと基本的に同様であり、ここではII) について述べる。

(問題)

与えられたデータ点列 (x_i, d_i) ($i = 1, \dots, N$)
 に対して、これを良く記述する実験公式 $f(\alpha, \beta, \gamma; x)$
 を求めよ。

これを人工知能の認知能力に基づいて行う試みが『CSOLD』(Cognitive System for One Line Data)である。これを知識工学的なアプローチに基づいてプロダクションシステムとして記述すると、関数同定に対する同定率が低い。システムの分解能を高めるために、純プロダクションシステムを離れ、次に述べる階差表を利用した数値的な認知方式を併用した。



システムはデータ $K^{(0)}$ とその階差表 $K^{(1)}$ を生成する。

このプロセスで 表に述べるフラグ, 最大値等を計算する。

(条 件)

(実 験 式)

$$\Delta y \text{ が一定} \longrightarrow y = a + bx$$

$$\Delta^2 y \text{ が一定} \longrightarrow y = a + bx + cx^2$$

$$\begin{aligned}
\Delta^2 y^2 \quad \text{が一定} &\longrightarrow y^2 = a + bx + cx^2 \\
\Delta^2 (x^2 y) \quad \text{が一定} &\longrightarrow y = a + \frac{b}{x} + \frac{c}{x^2} \\
\Delta^l (x^l y) \quad \text{が一定} &\longrightarrow y = a + \frac{b}{x} + \dots + \frac{g}{x^l} \\
\Delta^p (f(x) \cdot g(y)) &\longrightarrow \dots\dots\dots
\end{aligned}$$

により、確定的に $y(x)$ を決定することができる。それを行うために、CSOLD は 30 ケ以上の関数をもっている。[付録 4]
これらは以下の様に用いられる。

I) 定数

if allequal (d) then FC: B

II) 一次関数

if allequal (diff (d)) then

FC: A * X + B

III) 二次関数

if allequal (diff (diff (d)))

then FC: A * X² + B * X + C

IV) 単振動

if TYPE=vib and allequal (p

eak (d)) then FC: A * SIN (B * X + D)

V) 減衰振動

if TYPE=vib and mondec (pea

k (d)) then FX: A * EXP (-C * X) *

SIN (B * X)

この *CSOLD* を巾級数の同定に転用する場合、巾級数を有理数として評価すれば誤差は混入しないが、巾級数をトランケートするのでその意味の誤差が混入する。このシステムを駆動するとき、*allegual(d)* が真か偽かはある誤差を許す形で決定される。このために用意する打切りパラメタ ϵ により、認知プロセスは影響を受ける。

システムが影響を受ける様に ϵ を変化させる戦略をとっているが、得られる結論はコンサルテーションのための示唆と受け止めるべきものである。

6. Prolog を用いた数式処理システム

付録 1 に、筆者の開発した *Prolog* を用いた数式処理システム *CASP* に含まれるパッケージを示した。これは *CP/M 68K* 上の *Prolog* の上に実現されたシステムであり、その詳細はソフトウェアコンファレンス⁷⁾に発表する。

sdif が *SDIF* を行う関数である。*simp* は式の単純化関数であり、個々のケースに対応した単純化は $\mathcal{A}$ において行われる。これを用いて作成した微分方程式の巾級数解を求めるパッケージ *difeg2* を示してある。

プログラムリストを一見して判る様に、数学的内容が非常に読み取り易い点が特徴的である。また、*Prolog* の特徴として関数のモジュラリティが高いので、システムの機能を高めるために一々全体を顧慮することなく、知識の付加を行える長所を持つ。また、付録 2 に阪大 *Skpe up* を用いたシステムの

リストを示した。

Prolog で記述したメリットとして、数学的表現に近い形でルールを付加し易いことと、機能の増強が一行の付加で行えるという特徴がある。しかしそれ以上に、数学上の定理証明機能を実現し易いことがある。(これは *Prolog* で書きさえすれば *SDIF* 上で数学的な事実の成否が問い合わせ得ることを意味しない。その様に記述すれば可能であるということである。)

従って、証明過程を途中に含んだ法則の探索、定理の発見と言った人工知能システムを記述することに適していると思われるが、現状の利用できるシステムにおいては、メモリー容量の限界のために満足な形で作動しにくい。(第5世代のコンピュータが待たれる所以である。)

なお、4. で述べたオペレタの探索も *Prolog* 上では実現し易いことに注意を促したい。

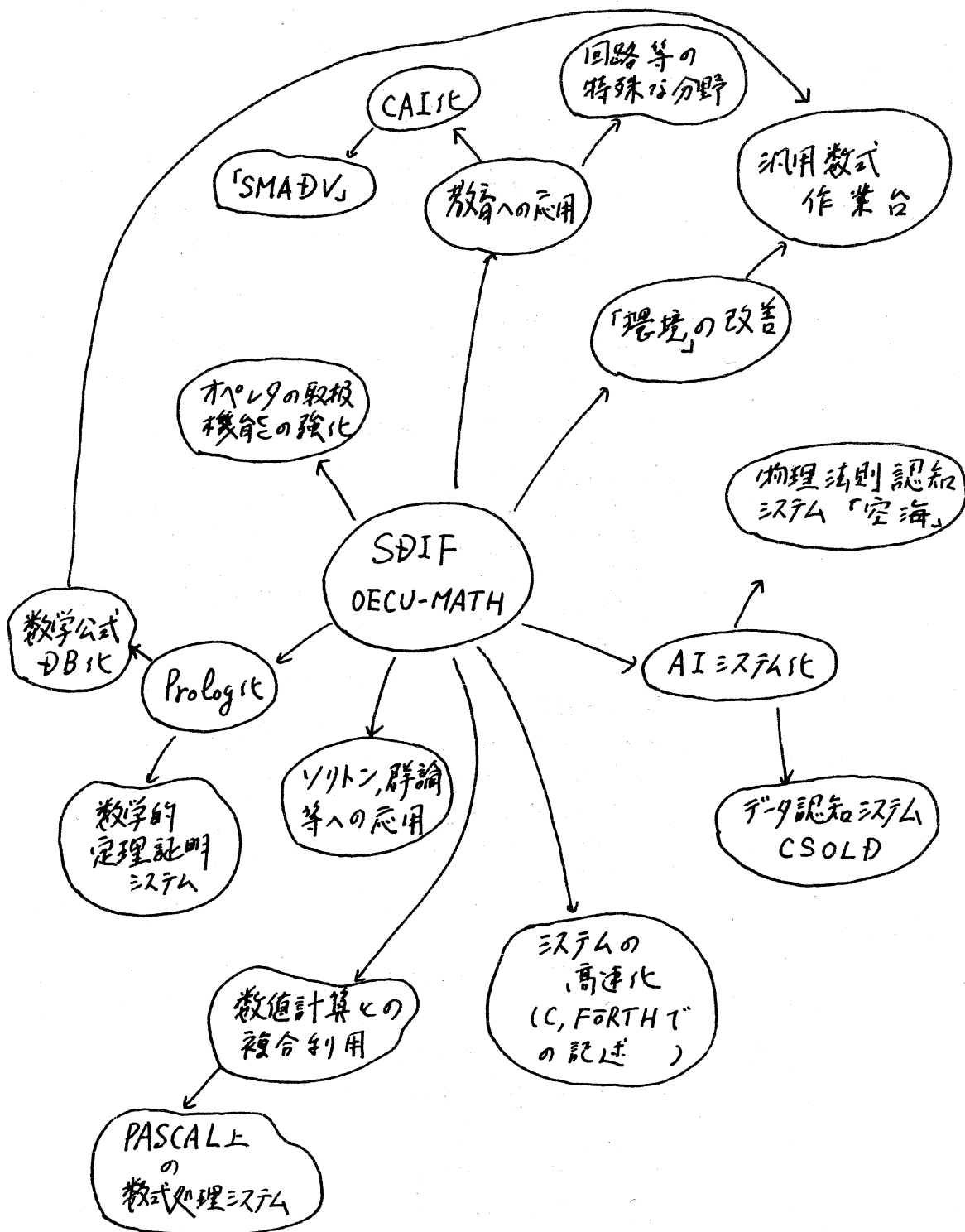
7. 展望

図に *SDIF* を核とした数理科学研究の知的ツールとしての数式処理システムの展望をしめした。現状の数式処理システムに対する一つの問題提起と思って頂きたい。

パソコン等のスモールシステムに重点が置かれているが、ある種の技術展望の上に意識的に行ったものである。

〔 参 考 文 献 〕

- 1) 対馬, 「知的な数式処理を目指して」, 数式処理通信
1-4, 2-1 (' 8 4)
- 2) 対馬, 「数式処理システムOECU-MATHの作成」,
大阪電通大科学論集, 18, 67 (' 8 2)
- 3) 対馬, 「数式処理システムOECU-MATHの拡張」,
同, 19 (' 8 3)
- 4) 対馬, 「パーソナルな数式処理システム」, (' 8 3
日本物理学会秋の分科会)
- 5) 対馬, 「人工知能的技法を用いたデータ認知システム
の作成」 (' 8 4 日本物理学会秋の分科会)
- 6) 対馬, 「データ認知システムCSOLDの知識構造」
(' 8 4 電気五学会関西支部連合大会)
- 7) ソフトウェアコンファレンス (' 8 5 . 3 . 2 3 大阪科学技術
センターにて開催)



【付録1】 SDIF を核とした数式処理の展望

(a) 1) 配列	DM
2) 整式係数取り出し	COEFZ
3) 引数付変数への代入	TCOEF
4) 因数分解	GCD
etc.	
(b) 5) 文字列処理	CHAR
6) ラプラス変換, 逆変換	LAP, ILAP
7) 小数表示パッケージ	DIV
8) 連立1次代数方程式の解放	RNRT
etc.	
(c) 9) 対話ヒストリ機能	KDRIVER
10) 消番機能	PRPRT
11) 数値計算との混合処理	NCSM
12) カナ文字利用	KANA
13) SRトランスレータ	SR
etc.	
(d) 14) 形式微分	SDIF
15) 微分方程式の級数解	KAI
16) 微分方程式の摂動解法パッケージ	PURTB
17) 形式微分の評価	GDIF
18) 微分方程式DB	DEDB
etc.	

[付録2] 「OECU-MATH」の機能の分類

- 6 オペレタの探索
- 5 メタオペレタの定義
- 4 オペレタ代数
- 3 オペレタの作用
- 2 オペレタの定義
- 1 整式

[付録3] 数式処理の階層

〔付録4〕CSOLDの言語構造

recognizer		modifier	
-----		-----	
equal	全て等しい	pickup	抜き取り
moninc	単調増加	divide	分割
mondec	単調減少		
zero	ゼロ		
zeros	多数のゼロ		
allplus	全て正值	etc.	

allminus	全て負値	diff	階差さ
		peakp	正ピーク
		peakm	負ピーク
value			
-----		pos	位置読み取り
max	最大値		
min	最小値	log	
		sqr	
length	リストの長さ		

[付録5] Shape Up上のDIFEQ2

-LIST

DIFEQ2 (*FNC,*NUM) :-

の主要部分

```

    PUTPROP (AR,1,U),
    PUTPROP (AR,2,UX),
    PUTPROP (AR,3,*FNC),
    PUTPROP (DIF,FNC,*FNC),
    PUTPROP (DIF,FNCTMP,*FNC),
    + (*NUM1,*NUM,1), ASSERTA (LOOP (*NUM1),1),
    LOOP (4).

```

LOOP (20).

LOOP (*N) :-

```

    GETPROP (DIF,FNCTMP,*FUN),
    SDIF (*FUN,X,*A),
    GETPROP (DIF,FNC,*FF),
    SIMP (*A,*AA),
    SWP (*AA,UX,*FF,*A1),
    PUTPROP (AR,*N,*A1),
    PUTPROP (DIF,FNCTMP,*A1),
    + (*N1,*N,1),
    LOOP (*N1).

```

INITAR (0) :-

```

    RETRACTN (LOOP,1),
    RETRACTN (LOOP2,1).

```

INITAR (*N) :-

```

    PUTPROP (AR,*N,0),
    PUTPROP (AR2,*N,0),
    - (*N1,*N,1),
    INITAR (*N1).

```

SWP (*X,*X,*Z,*Z).

SWP (*F,*X,*Z,*F) :- (ISSTRING (*F); ISINTEGER (*F)), !.

SWP ([*OP,*R],*X,*Z,[*OP,*L]) :- SWP (*R,*X,*Z,*L).

SWP ([*OP,*L,*R],*X,*Z,[*OP,*U,*V]) :- SWP (*L,*X,*Z,*U), SWP (*R,*X,*Z,*V).

EVALK (*NUM,*X0,*U0,*UX0) :-

```

    ASSERTA (LOOP2 (*NUM,*XX,*Y,*Z),1),
    LOOP2 (0,*X0,*U0,*UX0).

```

LOOP2 (20,*A,*B,*C).

LOOP2 (*N,*X0,*U0,*UX0) :-

```

    + (*N1,*N,1),
    EVAL (*X0,*U0,*UX0,*N1),
    LOOP2 (*N1,*X0,*U0,*UX0).

```

EVAL (*X0,*U0,*UX0,*N) :-

```

    GETPROP (AR,*N,*X),
    SWP (*X,X,*X0,*X1),
    SWP (*X1,U,*U0,*X2),
    SWP (*X2,UX,*UX0,*X3),
    - (*N1,*N,1),
    FAC (*N1,*N2),
    SIMP ([TIMES,*X3,[EXPT,*N2,-1]],*G),
    PUTPROP (AR2,*N,*G).

```

FAC (0,1).

FAC (*N,*X) :- - (*N1,*N,1), FAC (*N1,*A1), * (*X,*A1,*N).

LOOP3 (10).

LOOP3 (*N) :-

```

    GETPROP (AR2,*N,*X),
    - (*NN,*N,1),
    DISP (*NN),
    PUT (32), PUT (32),
    DE (*X),
    + (*N1,*N,1),

```

〔 付録6 〕 PrologによるSDIFシステム

```

0000: sdif( N, X, 0) :- integer( N).
0001: sdif( X, X, 1).
0002: sdif( P, X, 0) :- para( P).
0003: sdif( F, X, A) :- atomic( F),compress( F, X, A).
0004:
0005: sdif ( -U,X,-V) :-
0006:     sdif ( U,X,V).
0007: sdif ( A+B,X,U+V) :-
0008:     sdif ( A,X,U),
0009:     sdif ( B,X,V).
0010: sdif ( A-B,X,U-V) :-
0011:     sdif ( A,X,U),
0012:     sdif ( B,X,V).
0013: sdif ( C*U,X,C*V) :-
0014:     para ( C),
0015:     sdif ( U,X,V).
0016: sdif ( A*B,X,A*U+B*V) :-
0017:     sdif ( B,X,U),
0018:     sdif ( A,X,V).
0019: sdif ( A/B,X,A) :-
0020:     sdif ( U*V^-1,X,A).
0021: sdif ( e^U,X,A*e^U) :-
0022:     sdif ( U,X,A).
0023: sdif ( U^V,X,V*W*U^V-1) :-
0024:     para ( V),
0025:     sdif ( U,X,W).
0026: sdif ( U^V,X,U^V*(P*log (U)+V*Q*U^-1) ) :-
0027:     sdif ( V,X,P),
0028:     sdif ( U,X,Q).
0029: sdif ( log (U),X,A*U^-1) :-
0030:     sdif ( U,X,A).
0031:
0032: compress( A, B, X) :-
0033:     name( A, P),
0034:     name( B, Q),
0035:     append( P, Q, R),
0036:     name( X, R).
0037:
0038: append([ ], X, X).
0039: append([X:L1],L2,[X:L3]) :- append(L1,L2,L3).
0040:
0041: para(a).
0042: para(b).

```



```

0001: ?- op(10, xfy, ^).
0002: ?- op( 9, fx, sin).
0003: ?- op( 9, fx, cos).
0004: ?- op( 9, fx, tan).
0005: ?- op( 9, fx, log).
0006:
0007: simp( E, E) :- atomic(E), !.
0008: simp(-(-E), E) :- !.
0009: simp( E, F) :-
0010:     E =.. [Op,Ma],
0011:     ( name(Op, "sin");
0012:       name(Op, "cos");
0013:       name(Op, "tan");
0014:       name(Op, "log") ),
0015:     simp( Ma, Na),
0016:     F =.. [Op, Na].
0017: simp(-E, F) :- simp(E, G), ( mera(G, F) ; F = -G), !.
0018: simp( E, F) :-
0019:     E =.. [Op, La, Ra],
0020:     name(Op, Opn),
0021:     simp(La, X),
0022:     simp(Ra, Y),
0023:     s(Opn, X, Y, F).
0024:
0025: s("+", X, 0, X) :- !.
0026: s("+", 0, X, X) :- !.
0027: s("+", X, Y, Z) :- integer(X), integer(Y), Z is X+Y.
0028: s("+", X, X, 2*X).
0029: s("+", X, I*X, In*X) :- integer(I), In is I + 1.
0030: s("+", X, X*I, In*X) :- integer(I), In is I + 1.
0031: s("+", I*X, X, In*X) :- integer(I), In is I + 1.
0032: s("+", X*I, X, In*X) :- integer(I), In is I + 1.
0033: s("+", X, Y, X+W) :- mera(Y, W), !.
0034: s("+", X, Y, X+Y) :- !.
0035: s("-", X, 0, X) :- !.
0036: s("-", 0, X, -X) :- !.
0037: s("-", X, Y, Z) :- integer(X), integer(Y), Z is X-Y.
0038: s("-", X, X, 0).
0039: s("-", I*X, X, In*X) :- integer(I), In is I - 1.
0040: s("-", X*I, X, In*X) :- integer(I), In is I - 1.
0041: s("-", X, I*X, In*X) :- integer(I), In is I - 1.
0042: s("-", X, X*I, In*X) :- integer(I), In is I - 1.
0043: s("-", X, Y, X+W) :- mera(Y, W), !.
0044: s("-", X, Y, X-Y) :- !.
0045: s("*", X, 0, 0) :- !.
0046: s("*", 0, X, 0) :- !.
0047: s("*", 1, X, X) :- !.
0048: s("*", X, 1, X) :- !.
0049: s("*", X, Y, Z) :- integer(X), integer(Y), Z is X*Y.
0050: s("*", X, X, X^2).
0051: s("*", X, X^Y, X^Z) :- integer(Y), Z is Y + 1 ; Z = Y + 1.
0052: s("*", X^Y, X, X^Z) :- integer(Y), Z is Y + 1 ; Z = Y + 1.
0053: s("*", -1, X, W) :- mera(X, W), !.
0054: s("*", X, -1, W) :- mera(X, W), !.
0055: s("*", Im, A, S) :- integer(Im), Im < 0,
0056:     ( inte(Im, A, S) ;
0057:       ( mera(A, C), abs(Im, I), S = I * C ), !

```

```

0000:
0001: ?- op(10, xfy, ^ ).
0002: ?- op( 9, fx, sin).
0003: ?- op( 9, fx, cos).
0004: ?- op( 9, fx, tan).
0005: ?- op( 9, fx, log).
0006:
0007: simp( E, E ) :- atomic(E), !.
0008: simp(-(E), E) :- !.
0009: simp( E, F ) :-
0010:     E =.. [Op,Ma],
0011:     ( name(Op, "sin");
0012:       name(Op, "cos");
0013:       name(Op, "tan");
0014:       name(Op, "log") ),
0015:     simp( Ma, Na),
0016:     F =.. [Op, Na].
0017: simp(-E, F) :- simp(E, G), ( mera(G, F) ; F = -G), !.
0018: simp( E, F ) :-
0019:     E =.. [Op, La, Ra],
0020:     name(Op, Opn),
0021:     simp(La, X),
0022:     simp(Ra, Y),
0023:     s(Opn, X, Y, F).
0024:
0025: s("+", X, 0, X) :- !.
0026: s("+", 0, X, X) :- !.
0027: s("+", X, Y, Z) :- integer(X), integer(Y), Z is X+Y.
0028: s("+", X, X, 2*X).
0029: s("+", X, I*X, In*X) :- integer(I), In is I + 1.
0030: s("+", X, X*I, In*X) :- integer(I), In is I + 1.
0031: s("+", I*X, X, In*X) :- integer(I), In is I + 1.
0032: s("+", X*I, X, In*X) :- integer(I), In is I + 1.
0033: s("+", X, Y, X-W) :- mera(Y, W), !.
0034: s("+", X, Y, X+Y) :- !.
0035: s("-", X, 0, X) :- !.
0036: s("-", 0, X, -X) :- !.
0037: s("-", X, Y, Z) :- integer(X), integer(Y), Z is X-Y.
0038: s("-", X, X, 0).
0039: s("-", I*X, X, In*X) :- integer(I), In is I - 1.
0040: s("-", X*I, X, In*X) :- integer(I), In is I - 1.
0041: s("-", X, I*X, In*X) :- integer(I), In is I - 1.
0042: s("-", X, X*I, In*X) :- integer(I), In is I - 1.
0043: s("-", X, Y, X+W) :- mera(Y, W), !.
0044: s("-", X, Y, X-Y) :- !.
0045: s("*", X, 0, 0) :- !.
0046: s("*", 0, X, 0) :- !.
0047: s("*", 1, X, X) :- !.
0048: s("*", X, 1, X) :- !.
0049: s("*", X, Y, Z) :- integer(X), integer(Y), Z is X*Y.
0050: s("*", X, X, X^2).
0051: s("*", X, X^Y, X^Z) :- integer(Y), Z is Y + 1 ; Z = Y + 1.
0052: s("*", X^Y, X, X^Z) :- integer(Y), Z is Y + 1 ; Z = Y + 1.
0053: s("*", -1, X, W) :- mera(X, W), !.
0054: s("*", X, -1, W) :- mera(X, W), !.
0055: s("*", Im, A, S) :- integer(Im), Im < 0,
0056:     ( inte(Im, A, S) ;
0057:       ( mera(A, C), abs(Im, I), S = I * C )), !.
0058: s("*", A, Im, S) :- integer(Im), Im < 0,
0059:     ( inte(Im, A, S) ;

```

```

0060:      ( mera(A, C), abs(Im, I), S = C * I )), !.
0061: s("x", I, A, S) :- integer(I), inte( I, A, S), !.
0062: s("x", A, I, S) :- integer(I), inte( I, A, S), !.
0063: s("x", X, Y, A*B) :- mera(X, A), mera(Y, B), !.
0064: s("x", X, Y, X*Y) :- !.
0065: s("/", X, 0, Y) :- !, fail.
0066: s("/", 0, X, 0) :- !.
0067: s("/", X, 1, X) :- !.
0068: s("/", X, Y, W) :- integer(X), integer(Y), gc(X, Y, W).
0069: s("/", X, X, 1).
0070: s("/", Im, -X, Abs/X) :- integer(Im), Im < 0, abs(Im, Abs), !.
0071: s("/", -X, Im, X/Abs) :- integer(Im), Im < 0, abs(Im, Abs), !.
0072: s("/", X, Y, A/B) :- mera(X, A), mera(Y, B), !.
0073: s("/", X, Y, X/Y) :- !.
0074: s("^", 0, X, 0) :- !.
0075: s("^", 1, X, 1) :- !.
0076: s("^", X, 1, X) :- !.
0077: s("^", X, -1, 1/X) :- !.
0078: s("^", X, Y, W) :- integer(Y), Y > 0, integer(X),
0079:      zijou(X, Y, W), !.
0080: s("^", X, Y, X^Y).
0081:
0082: zijou(X, 0, 1) :- !.
0083: zijou(X, Y, Z) :- A is Y-1, zijou(X, A, B), Z is B*X.
0084:
0085: gc(X, Y, Z) :- (Z is X/Y, X is Y*Z, ! ;
0086:      ( gcd(X, Y, W), A is X/W, B is Y/W, Z = A/B)).
0087: gcd(I, 0, I) :- !.
0088: gcd(I, J, K) :- R is I mod J, gcd(J, R, K).
0089:
0090: merp(-1, X, X) :- !.
0091: merp( X, -1, X) :- !.
0092: merp(Im, X, Abs*X) :- integer(Im), Im < 0, abs(Im, Abs), !.
0093: merp( X, Im, Abs*X) :- integer(Im), Im < 0, abs(Im, Abs), !.
0094: merp(-X, Y, X*Y) :- !.
0095: merp( X, -Y, X*Y) :- !.
0096: merp( X, Y, W*Y) :- X =.. [Op,La,Ra],
0097:      ( name(Op,"*"), merp(La,Ra, W);
0098:      ( name(Op,"/"), merb(La,Ra, W) )).
0099: merp( X, Y, X*W) :- Y =.. [Op,La,Ra],
0100:      ( name(Op,"*"), merp(La,Ra, W);
0101:      ( name(Op,"/"), merb(La,Ra, W) )).
0102:
0103: merb( X, -1, X) :- !.
0104: merb(Im, X, Abs/X) :- integer(Im), Im < 0, abs(Im, Abs), !.
0105: merb( X, Im, X/Abs) :- integer(Im), Im < 0, abs(Im, Abs), !.
0106: merb(-X, Y, X/Y) :- !.
0107: Bmerb( X, -Y, X/Y) :- !.
0108: merb( X, Y, W) :- mera(X, Y, W).
0109: merb( X, Y, W/Y) :- X =.. [Op,La,Ra],
0110:      ( name(Op,"*"), merp(La,Ra, W);
0111:      ( name(Op,"/"), merb(La,Ra, W) )).
0112: merb( X, Y, X/W) :- Y =.. [Op,La,Ra],
0113:      ( name(Op,"*"), merp(La,Ra, W);
0114:      ( name(Op,"/"), merb(La,Ra, W) )).
0115:
0116: mera(Im, Abs) :- integer(Im), Im < 0, abs(Im, Abs), !.
0117: mera(-A, A) :- !.
0118: mera( X, W) :-
0119:      X =.. [Op,La,Ra],

```

```

0120:          ( name(Op,"*"), merp(La,Ra, W) ;
0121:          ( name(Op,"/"), merb(La,Ra, W) ) ).
0122:
0123: intp( I,In,Ra, It*Ra) :- integer(In), It is I * In, !.
0124: intp( I,La,In, La*It) :- integer(In), It is I * In, !.
0125: intp( I,La,Ra, S*Ra) :-
0126:     La =.. [Op, L, R],
0127:     ( name(Op,"*"), intp( I, L, R, S);
0128:       ( name(Op,"/"), intb( I, L, R, S) ) ).
0129: intp( I,La,Ra, La*S) :-
0130:     Ra =.. [Op, L, R],
0131:     ( name(Op,"*"), intp( I, L, R, S);
0132:       ( name(Op,"/"), intb( I, L, R, S) ) ).
0133:
0134: intb( I,In,Ra, It/Ra) :- integer(In),
0135:     It is I / In, !.
0136: intb( I,La,Ra, S/Ra) :-
0137:     La =.. [Op, L, R],
0138:     ( name(Op,"*"), intp( I, L, R, S);
0139:       ( name(Op,"/"), intb( I, L, R, S) ) ).
0140:
0141: inte( I, A, S) :-
0142:     A =.. [Op,La,Ra],
0143:     ( name(Op,"*"), intp( I,La,Ra, S);
0144:       ( name(Op,"/"), intb( I,La,Ra, S) ) ).
0145:
0146: abs(A, A) :- integer(A), A >= 0, !.
0147: abs(A, B) :- integer(A), B is 0 - A, !.
0148:
0149:
0150:

```