数理解析研究所講究録
第811巻 1992年 183-187

数式処理システムGALの移植

奈良女子大学理学部情報科学科　加古富志雄 (Fujio Kako)

はじめに

GAL(General Algebraic Language/Laboratory) は筑波大学の佐々木氏が中心になって作成している数式処理システムで、Lisp(Cambridge Lisp) の上でインプリメントされています。Lisp 処理系 (Common Lisp 等) があればその上で動かすことが可能ですが、今回は配布等のことも考えてGAL を動かすためのLISP 処理系 Sugar を作成し、その上に GAL を移植した。Sugar 上の GAL は現在 Sun3, SparcStation,Macintosh の上で動いている。

Lisp 処理系

GAL を動かす Lisp 処理系としては、Reduce を動かすのに必要な Lisp(Standard Lisp) とほぼ同程度の機能を持った Lisp で有ればよい。ただし、GAL で扱う数値として整数 (多倍長整数)、有理数、浮動小数点数は Lisp の処理系で扱えないといけない。このため、Standard Lisp の仕様そのままの Lisp では有理数演算が入っていないので GAL は動かせません。(Common Lisp の上では多分そのままで動かせるはずです)。今回作成した Lisp 処理系 (Sugar) は以前REDUCE を動かすために作成した Lisp (Standard Lisp) を修正した物で、

1. 数値の型として有理数型を追加したこと。また、GAL が復素数型をどのようにインプリメントするか決まっていないが、一応対応できるように対応できるように用意はしてある。

2. 関数の型として SMACRO を追加した。(最もこれは GAL のパーザーを修正することで必要なくなりましたが。)

3. 関数 FUNCALL を追加した。

4. 数式-数値融合システム (ANS) やグラフィックルーチンとのやり取りが行えるように Lisp と C との相互呼び出しを行うことができるようにした。これは、W. Wilson Ho によるダイナミックローダー Dld を組み込んで実行時に動的に C のオブジェクトを呼び出せるようにした。

等の機能を付け加えている。なをこのLispはアセンブラで書いており、今のところSparcとMC68000に対応している。

GALの移植

佐々木氏から受け取ったGALではパーザーはREDUCEのRLISPで書かれておりGALはGALのLispモードでの言語SSLisp(Syntax Sugared Lisp)で書かれている。GALをSugar上に移植するには

1. GALのパーザはREDUCEのRLISPで書かれている。これをSugarに対応するように修正する。
2. REDUCEをSugar上に移植し、REDUCEを起動してパーザを読み込む。
3. この時点でGALのLispモード(SSLISP)のインタープリターが動くので、それを使ってGALの本体を読み込む。

これでGALは動くようになる。ただし、この状態ではREDUCE(RLISP)の上で動いていることになる。これではGALを立ちあげるのにREDUCEが必要になるので出来るだけGALのみで処理することが出来るようにRLISPで書かれたパーザーをSSLispで書き直し、GALでコンパイラー等も全てSSLispで書き直した。

GALをSugarの上に移植するに当たって修正した点は

1. マクロ関数の引数、定義の方法が異なっている。
2. 関数EXPLODE,COMPRESSの定義の違い
3. Cambridge-Lispでは関数の実引数の数が仮引数と異なっている場合、実引数が少ないときにはNILを渡すが、Sugarではこのような場合エラーとして処理する。
4. 関数APPLYの第一引数にマクロ関数等は使えない。このため、FUNCALLという関数を定義し、それで処理するようにした。

実行例

GALは数式の四則演算や微分計算および、ベクトル、行列やテンソル計算のルーチンがインプリメントされている。また、代数的数やべき級数の演算も可能である。

数式の演算をREDUCEと比較した結果を示す。ここで、REDUCEは3.4版を同じSparcStation2の上で動かしている。(Lisp処理系も同じ物を使ってい

る。) ちなみにこの Lisp で REDUCE を動かしたとき、テストプログラムの実行結果は約 2.5 秒である。

GAL では、計算中での式のまとめ方として、exp(展開モード) と stexp(強展開モード) の 2 つのモードを持っている。exp モードでは式一部は展開されずそのままの形で計算される。これに対して stexp モード) では式は完全に単項式にまで展開される。

	gal		reduce
	on exp	on stexp	
$A7 := A^7$	60	80	80
$B7 := B^7$	100	110	120
$A7/A^2$	40	50	220
$A7 * B7$	3590	3660	7720
$subst(A7, y = 3, z = 4)$	10	10	50
$subst(B7, y = 3, z = 4)$	10	10	80
$e6 := e^6$	40	100	60
$f6 := f^6$	80	250	190
$e6 * f6$	2730	13300	11310

ただし、

$$A = x^2y + xy^2 + y^2z^2 + z + 1$$

$$B = A + yz$$

$$e = sin(x)(x + y) + cos(y)(z + 1) + (x - y)$$

$$f = e + (z + 1)$$

で、実行時間は milli-second 単位で表している。

べき級数の計算

declare powser: x <= 5, y < 6;
(1 + x + y)∧ 6;

では x,y について 6 次以上の項をゼロとした計算を行う。また、

$$1/(1 - x);$$

では

$$-1/(x - 1)$$

となるが、

$$1//(1 - x);$$

では

$$x^5 + x^4 + x^3 + x2 + x + 1$$

となる。なお、べき級数として定義した変数を元に戻すには

declare powser: x = nil, y = nil;

と入力すればよい。

代数的数の計算では、Algebraic Number を定義するには declare 文で

declare algn: #I **with** #I**2 + 1 = 0,
#JJ **with** #JJ**3 + #I + 1 = 0;

の様に入力すればよい。今の例では$\#I$は

$$I = i$$

$$J^3 = -i - 1$$

となる。GAL では代数的数を表す変数名として前に# を付ける。

これで、例えば

$$(1 + \#I + \#J)^2$$

を計算させると

$$\#J^2 + 2\#I\#J + 2\#J + 2\#I$$

となる。

行列の計算では、まず行列を定義するには

declare matx: m;

もしくは

declare matx: m(2,2);

のように入力する。REDUCE とは異なり、関数内部でローカルな行列を定義することもできる。例えば

```
proc pp(m) ==
  || ここでは m は行列もしくは行列型の変数名;;
  begin localdecl matx: a; || 局所的な行列を定義する。;;

    a:=m;    || 行列の代入;;
    return begin local b;
```

```
                b:=a(1,1)/a(2,2);
                return b;
                end;
    end;

    decl matrix: m(2,2);
    m(1,1):=x^6-1;
    m(1,2):=1;
    m(2,1):= 1;
    m(2,2):= x+1;

    pp(m);   || 行列の値を渡す。;;
```

もしくは

```
    pp('m);   || 行列型の変数名を渡す。;;
```

と入力すれば $m(1,1)/m(2,2)$ が計算される。

最後に

国産数式処理システム GAL をワークステーションに移植した。GAL には現在のところ数式の四則演算、微分演算、およびベクトル、行列、テンソル計算。式の係数体としては整数、有理数、代数的数が使える。また、高エネルギー物理学の計算、Gröbner 基底の計算等の計算パッケージが作成されている。

Sugar および GAL は無償で配布する予定である。ただし、ダイナミックローダのルーチンは今のところ Sparc の上でしかインプリメントしていない。