記 号 処 理　38－4
（1986. 9. 16）

数値・数式混合計算のための異言語の一結合方式

数値・数式混合計算システム

佐藤三久(*　佐々木建昭(**　福井義成(***　鈴木正幸(**

東京大学(*、理化学研究所(** 、東芝(***

数値・数式混合計算のための異言語の結合方式を提案する。この結合方式は、簡便であるためメインフレーム・メーカーにも受け入れられ易く、しかもＦＯＲＴＲＡＮやＬＩＳＰなど異なる言語で書かれたプログラムをリンクできる有効な方式である。その具体例として、数値計算向き言語であるＦＯＲＴＲＡＮとＬＩＳＰの数式処理言語を結合するシステムＡＮＳ(Algebraic Numeric System)について述べる。

A Scheme for Linking Different Computer Languages (in Japanese)
A System for Algebraic-Numeric Computation

Mitsuhisa Sato*, Tateaki Sasaki**, Yosinari Fukui***, Masayuki Suzuki**

* Department of Infomation Science, University of Tokyo, Bunkyo-ku, Hongo, Tokyo Japan.
** The Institute of Physical ans Chemical Research, RIKEN.
*** Toshiba Corporation.

A simple and promising scheme for linking different computer language is proposed. The scheme is so simple that even main-frame makers will not hesitate to employ it, yet it is so promising that even programs written in different kinds of languages, such as FORTRAN and LISP, can be linked together. As concrete example, we introduce and discuss ANS(Algebraic Numeric System) for linking FORTRAN and LISP(= a Lisp-based algebraic language) in detail.

1。 まえがき

コンピュータによる数値・数式混合計算は以前から切望されているものの、いまだ決定的といえる方法が提出されていないと言ってもよい。この不満足な状態に対し、本稿では応用上もインプリメンテイション上も非常に有望と思われる方法を提案する。なお、本稿では、ＦＯＲＴＲＡＮは数値計算を、ＬＩＳＰは数式処理を代表する言葉として使用する。

まず、「数値・数式の混合計算」の意味を明確にしておく。この言葉の意味は人によって異なると思うが、我々は以下のように非常に一般的な意味であるとする。

（１）数値計算言語としてのＦＯＲＴＲＡＮシステムとそのライブラリ、および数式処理システムとそのライブラリを最大限に利用できる。

（２）ＦＯＲＴＲＡＮから数式処理システムがコールでき、逆に数式処理言語の中でＦＯＲＴＲＡＮシステムがコールでき、この相互コールが計算をストップさせることなく自在にできる。

（３）上記の混合計算が、ＦＯＲＴＲＡＮが得意とするコンパイル・バッチ方式においても、ＬＩＳＰが得意とする対話的処理方式においても、実行できる。

上記の定義は、ＦＯＲＴＲＡＮとＬＩＳＰの枠内で考える限り十分に一般的だが、我々は欲張って、さらに次の条件を課す。

（４）結合方式はＦＯＲＴＲＡＮとＬＩＳＰに限らず、他の言語系の結合にも使えるものでなければならない。したがって、たとえばＣで書いたグラフィックス・システムとも容易に連結できるものでなければならない。

（５）結合方式は、従来の言語体系を変更・拡張する場合には、多くの変更を必要とするものであってはならない。

この最後の点は、ＦＯＲＴＲＡＮもＬＩＳＰも既に十分なライブラリの蓄積があり、それをそのまま利用できることが強く望まれていること、および異言語結合方式が広く受け入れられるためには、汎用機メーカーも容易に受け入れるほど簡単でなければならないことを意味する。

第２章では数値計算と数式処理を融合させるための従来の代表的な方式を概観し、現状がなぜ不満足なのかを見る。混合計算のために従来払われた労力の大きさと、それにもかかわらず不満足な現状を見た読者は、「上記の諸要求を満足させる結合方式などあり得ない」と思うだろう。これに対し、第３章では新しい結合方式を提案するが、それは非常に簡単でメーカーにも受け入れ易いものだと思う。そこまで読むと、読者は「なんだ、コロンブスの卵だ」と思うに違いない。新しい結合方式はそれほど簡単なのである。本稿で提案する結合方式は現在ＬＩＳＰとＣを対象にＵＮＩＸ上にインプリメントされつつあるが、第４章では、それまでの議論をＦＯＲＴＲＡＮとＬＩＳＰ、およびＯＳへの提言としてまとめる。

2。 従来の混合計算方式のサーベイと批判

数値計算と数式処理の融合に関しては、数式処理が始まったごく初期の段階から試みられていた。従来の方法のうち代表的なものを以下に列挙し、それらの方法の欠点を指摘しよう。

（１）ファイル・ハンドラを作成し、共通に使用するデータはファイルを経由してやりとりする。

（２）ＦＯＲＴＲＡＮのサブルーチンとして数式処理システムを組み込む。

（３）ＬＩＳＰの中にＦＯＲＴＲＡＮを取り込む。
（４）数値計算と記号処理が両方できる言語で数式処理システムを作製する。

方法（１）に関して、数式処理システムＲＥＤＵＣＥを使って混合計算を実行しようとすれば、現在はこうせざるを得ない（ただし、ファイル・ハンドラが不十分だから、その働きを人間が代行する）。混合計算に対する我々の定義からすれば、これは混合計算と呼ぶには余りに貧弱である。

方法（２）に関して、この方式は１９６５年頃にＩＢＭで開発された数式処理システムＦＯＲＭＡＣで採用されたものである。当時は数値計算が主で、数式処理はごく簡単なものだけを想定しており、それで十分だと考えていた。しかし、実際にＦＯＲＭＡＣを開発してみると、数式処理は膨大になり、ＦＯＲＴＲＡＮの枠内にはとても収まりきらないことが判明した。

方法（３）に関して、これはＭＩＴで開発された数式処理システムＭＡＣＳＹＭＡで採用された方式である。ＭＡＣＳＹＭＡのホスト言語であるＭＡＣ－ＬＩＳＰには、ＦＯＲＴＲＡＮからＬＩＳＰへのトランスレータ、ＬＩＳＰで書かれた数値計算プログラムの高速実行のためのコンパイラー、さらにはＬＩＳＰからＦＯＲＴＲＡＮサブルーチンのコール機能もある。この方式は一見良さそうだが、このようなシステムを開発せよと言われれば大メーカーでさえ二の足を踏むであろう。

方法（４）は、ＣＡＬＴＥＣで開発された数式処理システムＳＭＰで採用された戦略である。ＳＭＰはシステム記述用の言語Ｃで書かれているが、Ｃの数値演算用サブセットはＦＯＲＴＲＡＮとの親和性がきわめて高い。おそらく、この方策は上記の三方法に比べれば最善であろう。しかし、この方策では、ＲＥＤＵＣＥやＭＡＣＳＹＭＡに蓄積されたパッケージ群は使えないから、すべてを新たに構築し直す必要がある。

３。　新しい結合方式の提案　・・・　概要

本稿で提案する結合方式は次の四つの柱から成る（以下では結合される言語をＡとＢとする）。

（１）混合計算のためにＡとＢの言語を混合してプログラムを書き（ただし、一つの文中に混合させることは許さない）、そのプログラムをＡとＢのコルーチン（coroutines）として、各々の処理系で別々に、かつ逐次的に実行する。
（２）ＡとＢで共通に使用されるデータは実行に先立って宣言し、ＡとＢの処理系に別々に割り当てる。しかし、これらの共通データはユーザにはあたかもＡとＢが共有しているかのように見えるよう定義するものとする。
（３）プログラムの実行が、たとえばＡの処理系からＢの処理系へ移行するときには、Ａに割り当てられている共通データを（必要ならば）内部表現の変換を行ったうえでＢのそれにコピーする。
（４）ＡとＢで共通に使用できるファイルを許し、実行時に生成されたプログラム等は共通ファイルを経由してやり取りする。

上記（２）と（３）に関して、たとえば固定長浮動小数点でさえＦＯＲＴＲＡＮとＬＩＳＰで内部表現が異なるのが普通である（ユーザに対する外部表現は同じであるが）。この内部表現の不整合性を解決するために、上記のような共通データの扱いをするのである。

上記の提案を図を使用して説明する。説明を具体的にするため、ＡからＢ（又はＢからＡ）への移行は「ＥＮＴＥＲ　Ｂ」（あるいは「ＥＮＴＥＲ　Ａ」）なるコマンドで実行され、共通データは「ＧＣＯＭＭＯＮ　Ｘ」のように宣言されるものとする。我々の方式による混合プログラムを図１に、共通データの扱いを図２に、それぞれ概念的に示した。

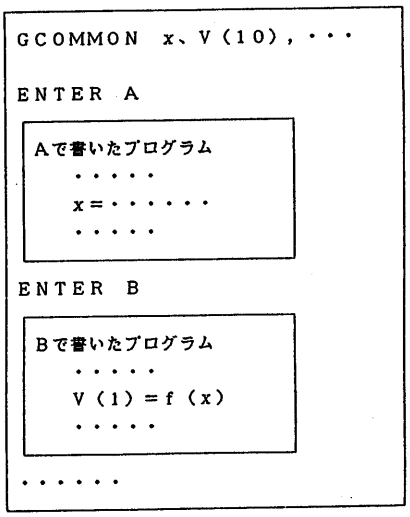
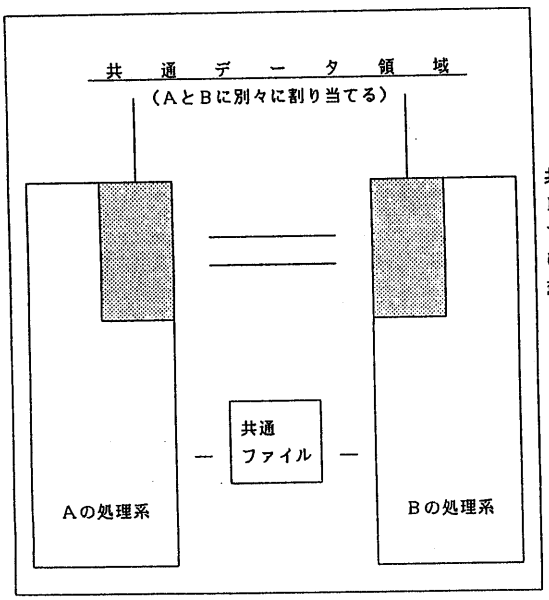


図１。混合計算プログラムの概念図
ＧＣＯＭＭＯＮで宣言された記号はＡとＢで共通に使える。副プログラムが混合プログラムであってもよい。

４。ＦＯＲＴＲＡＮ、ＬＩＳＰ、およびＯＳへの提言

我々の方式の詳細は論文（A scheme for linking different computer languages --- from the viewpoint of algebraic-numeric computation ---, preprint 1986 ）を見て欲しいが、本稿では結果だけを提言の形でまとめる。



共通データはＥＮＴＥＲコマンド実行時に内部表現の変換後コピーされる。実行に生成されたプログラムは共通ファイルで受渡しされる。

「我々の方式においては、バッチ処理方式のみならず対話的処理方式も実現することが可能である。また、ＬＩＳＰプログラムのみならず、ＦＯＲＴＲＡＮプログラムも実行時に生成・実行することが場合によっては可能である。」

しかし、そのためにはＯＳが次のような操作を許すものでなければならない。

（０１）ＦＯＲＴＲＡＮとＬＩＳＰをコルーチンとして実行する機能；

（０２）ＦＯＲＴＲＡＮとＬＩＳＰに割り当てられた共通データの実行時変換・複写機能；

（０３）ＦＯＲＴＲＡＮとＬＩＳＰでファイルを共有すること；

（０４）ＦＯＲＴＲＡＮプログラムの動的リンク機能（必ずしも必要ではない）。

次に、ＦＯＲＴＲＡＮとＬＩＳＰに関する提案を記す。

（Ｐ１）コマンドＧＣＯＭＭＯＮとＥＮＴＥＲの追加；

（Ｐ２）配列とストリングの外部表現の統一；

（Ｐ３）整数・浮動小数の混合データ型の数の導入。

５．　ＡＮＳ　(Algebraic Numeric System)

現在、我々がＵＮＩＸ　４．２　ＢＳＤ上にインプリメント中である混合計算を記述するためＡＮＳ (Algebraic Numeric System) のプログラム例 (example1) をしめす。

'$' ではじまる行は、ＡＮＳプリプロセッサで処理され、ソースプログラムは対応するＣとＲＥＤＵＣＥのプログラムに分割される。

＄ＵＳＥは、使用する言語の定義であり、＄ＴＯＰはスタートオフを行なう言語をしめす。＄ｖａｒはＧＣＯＭＭＯＮにおく変数を宣言する。これらの変数は混合計算において共用される。

＄＜言語＞｛　．．．｝は使用する言語の変更を指定する。プリプロセッサによって、この部分は別々のファイル中に分割され、コルーチンとして実行される。コルーチンの切り替え時には、ＧＣＯＭＭＯＮ中の変数はコピーが行なわれる。

＄ｄｅｆ（＜言語＞）｛　．．．　｝は、指定された言語でのプログラム定義をする部分であり、それぞれのファイル中に定義される。

数式処理システム中の数式が代入された変数は数値計算中では、関数として使用できる。(example2) ＄ＦＵＮＣは数式処理システムで計算される関数を指定する。Ｃ言語中では、ｆ は関数変数として使用することができる。

現在、ＵＮＩＸ上のシステムは、ＧＣＯＭＭＯＮのコピーはＩＰＣの機能をつかって実現している。

６．　混合計算システムの利用

ＡＮＳの利用の簡単な例としては、数式微分を利用したＮｅｗｔｏｎ法の計算などが考えられる。ＡＮＳで最も有効な計算は、数値計算によって得られた結果を使って数式処理を行ない、さらにその結果を数値計算に反映させることのできる場合であり、我々は現在そのような数値数式混合計算のアルゴリズムを検討している。

また、数値計算、数式処理、グラフィックスによるグラフ出力などを含めた対話型システムをＡＮＳをつかってインプリメントする予定である。

```
$ANS
$USE (A C)
$TOP (C)

$var ((double a) (double b 3))
```

＜example1＞

```
main()
{
        a=1.0;
        b[0]=2.0;
        b[1]=3.0;
$ A {
        write ":-> Enter Reduce";
        a := 102.0;
        b(1) := 12.0;
        foo();
$ C     {
                printf(":-> Enter C function\n");
                printgcommon();
$       }
        write ":-> Return from C";
$   }
$exit
}

$ def (A) {

  procedure foo ();
    begin
      write ":-> Enter Foo!";
    end;
$}
```

```
$ANS
$USE (A C)
$TOP (C)

$var ((int co 10))

$func (double foo ((double x)))
```

＜example2 : Newton法＞

```
# define EPS 1.0e-8
# define Maxcount 100

main()
{
        double x0, x1;

        /*
        Read coefficient
        */
        readcoeff();

        /*
        Make Newton method function
        */
 $A {
        F := 0;
        for i:=0:9 do F := (F*x + co(i));
        foo := x- F/df(F,x);
 $}
        x0 := 1.0;
        for(i=0;i<Maxcount;i++) {
                x1 = (* foo) (x0);
                if(fabs(x1-x0)<EPS) {
                        print("Result = %F\n, x1);
                        break;
                }
        }
 $ exit
 }
```