プログラミング言語ＸＸＰＬの紹介

大阪大学基礎工学部
杉　山　裕　二・中　西　通　雄*

コンパイラ、インタプリタ、エディタなど言語処理プログラムの作成に便利な言語ＸＸＰＬ とその使用法を紹介します。詳しくは、 ＸＸＰＬ言語仕様、 ＸＸＰＬ使用説明書を参照して下さい（センターにあります）。

1. 概　　容

ＸＸＰＬは、言語ＸＰＬ[(1)] を手本として作られたものです。このシステムには、ユーザが新たに定義したい言語の文法を解析し、文献(1)の構文解析アルゴリズムで必要なディシジョン・テーブルを作成する文法アナライザＡＮＡＬＹＺＥＲが用意されていますので、これを用いれば比較的容易にコンパイラなどが作成できます。

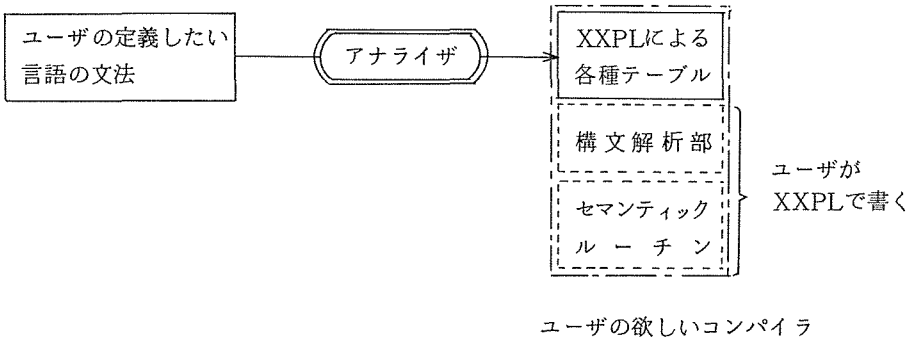

図１．コンパイラの作成

2. ＸＸＰＬ 言語の特徴

ＸＸＰＬは PL/I 的表現形式をもつ言語で、次のような特徴があります。⑤から⑧は、 ＸＰＬにはなく、ＸＸＰＬに新しく追加された機能です。

① 可変長文字ストリング（最大長 255 文字）が扱える。

② 入出力は、可変長文字ストリングとして扱えるので容易である。

③ インラインコード（ＧＭＡＰ）が書ける。

④ case 文が使える。

＊　現在、三菱電機㈱計算機製作所

⑤ フォートランとのリンクができる。

⑥ オーバーレイの指定ができる。

⑦ 外部手続きごとにコンパイルできる。

⑧ 再帰的呼び出し(recursive call)が可能。

なお、XXPLは、文字処理、ビット処理を主目的とする言語で、データ・タイプとして浮動小数点などがなく、数値計算などには向きません(FORTRANとリンクすれば何とかなりますが…)。XXPLとXPL、PL/Iとの関係については、XXPL言語仕様書に述べられています。

3. XXPLの使用法について

XXPLソース・プログラムのコンパイルは、バッチ、TSSどちらでも行なうことができますが、TSSで行なう方が、エラーの修正が行ないやすいので良いでしょう。

コンパイルの結果は、GMAPのプログラムですから、これをバッチでアセンブルしてオブジェクトモジュールにします。最後に、バッチ又はTSSのRUNコマンドにより、オブジェクトモジュールをリンクして、ロードモジュールにすれば、実行できる形になります(図2)。

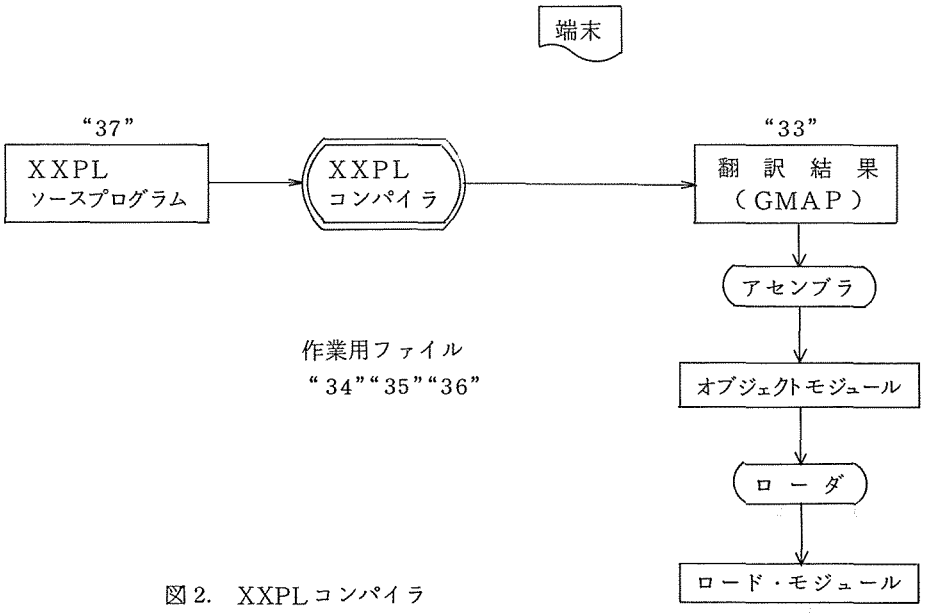

図2. XXPLコンパイラ

また、TSSコマンドとして、"XXPL"があり、そのパラメータは、フォートランサブシステムのRUNコマンドと全く同じです。オプションにNOGOを指定した時でも、コンパイル結果のGMAPプログラムをアセンブルするために、自動的にバッチジョブを起動するようにつくられています。ところが、繁雑期には、このバッチジョブの結果ができるまで時間がかかり、その間端末

は使えなくなりますので、このコマンドは使わずに、次の方法を取ることをお勧めします。

① コンパイルをTSSで行なう。

② コンパイル結果の入っているテンポラリファイル(代替名33)を、いったんパーマネントファイルにSAVEする。

③ CARDINサブシステムで、SAVEしたファイルをアセンブルする。

④ オブジェクトモジュールをリンク・実行する。

②③の作業は、ふつう連続して行なうので、コマンドファイルを作成しておき、CRUNコマンドを使えば便利です。①のコマンドファイルを図3に、②③のコマンドファイルを図4に示しておきます。

```
 list compile

##;(XXPL-SOURCE?);EXCLUDE
$*$REM ***************      COMPILE ***********************
$*$REM   This is a command file which gets XXPL source
$*$REM program and outputs GMAP program into "33".
$*$REM ************************************************
SYST FORT N
REMO #1
GET #1"37"
RUN XXPL/TCOM#37;33;34;35;36
REMO 37,34,35,36,TCOM
SYSTEM
```

図3. コンパイルのコマンドファイル

```
 list assemble

##;(OBJECT-FILE?);EXCLUDE
$*$REM *************      ASSEMBLE ********************************
$*$REM   This is a command file which gets GMAP
$*$REM program and makes it an object module.
$*$REM ************************************************************
SYST
CARDIN
PERM 33;TEMPGMAP
REMO TEMPGMAP
NEW
100##N,J
110$:JOB:kadai_no$password,E,MAIL,R
120$:OPTION:NOGO
130$:SELECTA:kadai_no/TEMPGMAP
140$:PRMFL:C*,W,L,kadai_no#1
150$:LIMITS:1,30K,-3K,1000
160$:ENDJOB
RUN
SYSTEM
```

図4. アセンブルのコマンドファイル

図3で、XXPL/TCOMはコンパイラのファイル名(TSS用)、37はXXPLソースファイ

ルの代替名、33はコンパイル結果のGMAPプログラムが入るテンポラリファイル、34、35、36は作業用テンポラリファイルです。

図4で、“kadai_no”及び“password”は、それぞれ、利用者の課題番号およびパスワードで置き換えられなければなりません。又、“TEMPGMAP”は、GMAPソース（XXPL翻訳結果）のパーマネントファイルとしてのファイル名で、適当な名前に変えることができます。

4. XXPL使用上の注意

XXPL使用説明書に書かれていないことで気が付いた点を挙げておきます。

① 複数個のモジュールをリンクする時、EXTERNAL宣言された配列の初期値宣言のあるモジュールを、その配列を使うだけの（初期値宣言のない）モジュールより先に指定する必要があります。この順序を誤ると、ローダより、“size inconsistent”の診断メッセージが出ることがあります。又、メインモジュール（MAIN指定のある手続）は一番最初に指定しなければなりません。

② 名前付き共通領域（EXTERNAL変数）の数は、GMAPの制限から、コンパイル単位につき63コまである。

③ 2進表現で、1語が“100…00”となる定数は使えない。

5. XXPL使用例

階乗の値を計算するXXPLのプログラムを作成し、その実行を前述の2通りの方法で行なった例を示しておきます。

```
list sample

 SAMPLE : PROCEDURE MAIN ;
   DECLARE X CHARACTER ,
           N FIXED     ;

  /*****************************************************************/
  /*                                                               */
  /*    FACT(N) returns the factorial of N.                        */
  /*                                                               */
  /*****************************************************************/

     FACT : PROCEDURE ( N ) FIXED RECURSIVE ;
      DECLARE N FIXED ;
      IF N = 0  THEN  RETURN ( 1 ) ;
                ELSE  RETURN ( N * FACT(N-1) ) ;
     END FACT ;

  /*****************************************************************/
  /*                                                               */
  /*         DEC CONVERTS S TO BINARY, ASSUMING IT CONSISTS OF     */
  /*    DECIMAL DIGITS ONLY, AND RETURNS THE CONVERTED VALUE.      */
  /*                                                               */
  /*****************************************************************/

    DEC: PROCEDURE(S) FIXED;
      DECLARE S CHARACTER,
               N FIXED;
      N = 0;
      DO WHILE(LENGTH(S)>0);
         N = N*10 + BYTE(S) - BYTE('0');
         S = SUBSTR(S,1);
      END;
      RETURN(N);
    END DEC;

 /*  MAIN  LOOP  */

    OUTPUT = 'This is a sample. Please input a number.';
    DO WHILE ( TRUE ) ;
       X = INPUT ;
       IF X = ''  THEN RETURN ;
       N = DEC ( X ) ;
       OUTPUT = 'Factorial of ' // N // ' = ' // FACT(N) ;
    END ; /* of do while */

 END SAMPLE ;


SYSTEM ?
```

図5. プログラム例

```
SYSTEM ?xxpl sample
XXPL COMPILATION -- ELECTROTECHNICAL LABORATORY -- XXCOM 003 VERSION OF JANUARY.31,1978

TODAY IS JUNE.30,1980, CLOCK TIME = 11:00:15.54

NO ERROR WERE DETECTED FOR SAMPLE AND, THIS PROCEDURE'S ACTUAL COMPILATION TIME = 0:00:00.95

ASSEMBLE START
  SNUMB # B309T
normal termination

LOADER START
This is a sample. Please input a number.
=5
Factorial of 5 = 120
=8
Factorial of 8 = 40320
=0
Factorial of 0 = 1
=[CR]

SYSTEM ?
```

図6.1　ＴＳＳコマンド“ＸＸＰＬ”による翻訳・アセンブル・実行

```
SYSTEM ?crun compile
XXPL-SOURCE? sample

XXPL COMPILATION -- ELECTROTECHNICAL LABORATORY -- XXCOM 003 VERSION OF JANUARY.31,1978

TODAY IS JUNE.30,1980, CLOCK TIME = 12:05:06.13

NO ERROR WERE DETECTED FOR SAMPLE AND, THIS PROCEDURE'S ACTUAL COMPILATION TIME = 0:00:00.95

SYSTEM ?crun assemble
OBJECT-FILE? /objl

  SNUMB # B431T
SYSTEM ?jsts B431T
B431T OUTPUT WAITING    ID=C1.
normal termination

SYSTEM ?jout b431t
function?rele
SYSTEM ?run objl=(ulib)xxpl/libr/rtslib
This is a sample. Please input a number.
=4
Factorial of 4 = 24
=3
Factorial of 3 = 6
=10
Factorial of 10 = 3628800
=[CR]
```

注１）　オブジェクトモジュールを入れるファイルＯＢＪ１（順編成ファイル）は
あらかじめ用意されていなければならない。

注２）　ＲＵＮコマンドでは、ランダムライブラリＲＴＳＬＩＢを必ず指定しなけ
ればならない。

図6.2　コマンドファイル“ＣＯＭＰＩＬＥ”，“ＡＳＳＥＭＢＬＥ”
を用いた翻訳・アセンブル・実行

6. ANALYZERの使用法

ANALYZERは、BNFで書かれた文法が入力として与えられたとき、それがdegree(2, 1; 1, 1)のMSP(mixed strategy precedence)[1] のクラスに属しているか否かを判定し、MSP(2, 1 ; 1, 1)構文解析アルゴリズム用のディシジョンテーブルを作成します。BNFによる文法の書き方、文法のデバッグについては文献(1) Chapter 7 に詳しく述べられています。図7に、ANALYZER入力用文法ファイルのTSSでの作成例を示します(図7の文法は、MSP(2, 1 ; 1, 1)ではありません)。

```
SYSTEM ?edit n
enter
*<statement> <subroutine stmt>
* <call stmt>
*<subroutine stmt> SUBROUTINE <id> ( <id list> )
*<id list> <id>
* <id list> , <id>
*<call stmt> CALL <id> ( <exp list> )
*<exp list> <expression>
* <exp list> , <expression>
*<expression> <id>
*
_ CR
```

図7. 文法の記述例

このファイルと、出力結果の入るファイルを指定して、ANALYZERをRUNさせます。また、ANALYZERは、検出した文法の誤り、及び各種の情報を、標準出力ファイルに出力します。RUNは、バッチでもTSSでも行なえますが、標準出力ファイル(TSSの時は端末)に出力される情報は、1行に132文字で、しかも量が多いので、バッチの方が良いでしょう。
図8にANALYZERの使用法を示します。

```
 list analyze

0010##N
0020$:JOB:kadai_no$password,A,MAIL
0030$:PROGRAM:RLHS,ONl
0040$:LIMITS:5,64K,,1000
0050$:PRMFL:H*,R,R,LIB/ANALCOM
0060$:PRMFL:07,R,L,kadai_no/GRAMMAR
0070$:PRMFL:02,W,L,kadai_no/OUTPUT
0080$:PRMFL:03,W,L,kadai_no/ANATABLE
0090$:ENDJOB
```

(1) バッチの場合のジョブ制御文

```
SYSTEM ?RUN LIB/TANALl,R#GRAMMAR"07";ANATABLE"03"
```

(2) TSSの場合のコマンド

図8. ANALYZERの実行

図7の文法に対するANALYZERの出力情報を図9に、又、同じ内容をMSP(2,1;1,1)で記述した場合の一例を図10に示します。

```
GRAMMAR ANALYSIS -- ELECTROTECHNICAL LABORATORY -- ANALYZER 001 VERSION OF MARCH.31,1977

TODAY IS APRIL 16 , 1980.

          P R O D U C T I O N S


     1  <statement>   ::=  <subroutine stmt>
     2                  !  <call stmt>

     3  <subroutine stmt>   ::=  SUBROUTINE <id> ( <id list> )

     4  <id list>   ::=  <id>
     5                 !  <id list> , <id>

     6  <call stmt>   ::=  CALL <id> ( <exp list> )

     7  <exp list>   ::=  <expression>
     8                 !  <exp list>   <expression>

     9  <expression>   ::=  <id>

TIME USED WAS 0.043 SECONDS.
TOTAL TIME IS 0.043 SECONDS.
```

```
  T E R M I N A L   S Y M B O L S                    N O N T E R M I N A L S

    1     (                                            8     <id list>
    2     )                                            9     <exp list>
    3     ,                                           10     <statement>
    4     .l.                                         11     <call stmt>
    5     <id>                                        12     <expression>
    6     CALL                                        13     <subroutine stmt>
    7     SUBROUTINE

<statement> IS THE GOAL SYMBOL.

TIME USED WAS 0.022 SECONDS.
TOTAL TIME IS 0.065 SECONDS.
```

```
PRODUCED HEAD SYMBOLS:  PAGE 1   OF  1

                                  1111
                        1234567890123
                       +-------------+
                       !Y            !
     2  )              ! Y           !
     3  ,              !  Y          !
     4  .l.            !   Y         !
     5  <id>           !    Y        !
     6  CALL           !     Y       !
     7  SUBROUTINE     !      Y      !
     8  <id list>      !    Y  Y     !
     9  <exp list>     !    Y   Y  Y !
    10  <statement>    !     YY  YY Y!
    11  <call stmt>    !     Y    Y  !
    12  <expression>   !    Y      Y !
    13  <subroutine stmt> !      Y    Y!
                       +-------------+


TIME USED WAS 0.118 SECONDS.
TOTAL TIME IS 0.184 SECONDS.
```

```
SENTENTIAL FORM PRODUCTION:


F11 HAS 11 ELEMENT.
THE MAXIMUM DEPTH OF RECURSION WAS 5 LEVELS.
13 SENTENTIAL FORMS WERE EXAMINENED.

TIME USED WAS 0.030 SECONDS.
TOTAL TIME IS 0.215 SECONDS.



C1 MATRIX FOR STACKING DECISION:  PAGE 1   OF  1


                          1234567
                         +-------+
    1  (                 !    Y  !
    2  )                 !   N   !
    3  ,                 !    Y  !
    4  .1.               !     YY!
    5  <id>              !YNN    !
    6  CALL              !    Y  !
    7  SUBROUTINE        !    Y  !
    8  <id list>         ! YY    !
    9  <exp list>        ! YY    !
   10  <statement>       !   N   !
   11  <call stmt>       !   N   !
   12  <expression>      ! NN    !
   13  <subroutine stmt> !   N   !
                         +-------+

TABLE ENTRIES SUMMARY:
    72
    11  Y
     8  N
     0  #

TIME USED WAS 0.093 SECONDS.
TOTAL TIME IS 0.308 SECONDS.

NO TRIPLES REQUIRED

TIME USED WAS 0.010 SECONDS.
TOTAL TIME IS 0.319 SECONDS.



CONTEX CHECK FOR EQUAL AND EMBEDDED RIGHT PARTS:

 THERE ARE 2 AND 2 VALID CONTEXTS,RESPECTIVELY,FOR
     5  <id list>  ::=  <id list> , <id>
     4  <id list>  ::=  <id>
 THEY CAN BE RESOLVED BY LENGTH.

 THERE ARE 2 AND 4 VALID CONTEXTS,RESPECTIVELY,FOR
     5  <id list>  ::=  <id list> , <id>
     9  <expression>  ::=  <id>
  *** ERROR. THESE PRODUCTIONS CANNOT BE DISTINGUISHED WITH (1,1) CONTEXT
   <expression> HAS ,...) AS CONTEXT AND ) IS VALID RIGHT CONTEXT FOR <id list>
   <expression> HAS ,..., AS CONTEXT AND , IS VALID RIGHT CONTEXT FOR <id list>

 THERE ARE 2 AND 4 VALID CONTEXTS,RESPECTIVELY,FOR
     4  <id list>  ::=  <id>
     9  <expression>  ::=  <id>
  *** ERROR. THESE PRODUCTIONS CANNOT BE DISTINGUISHED WITH (1,1) CONTEXT
 THEY HAVE EQUAL RIGHT PARTS AND THE COMMON CONTEXT (...)
 THEY HAVE EQUAL RIGHT PARTS AND THE COMMON CONTEXT (...,

 THERE ARE 2 AND 2 VALID CONTEXTS,RESPECTIVELY,FOR
     8  <exp list>  ::=  <exp list> , <expression>
     7  <exp list>  ::=  <expression>
 THEY CAN BE RESOLVED BY LENGTH.
```

エラー・メッセージ

エラー・メッセージ

```
C2 PRODUCTION CHOICE FUNCTION:


  ) AS STACK TOP WILL CAUSE PRODUCTIONS TO BE CHECKED IN THIS ORDER:

     3  <subroutine stmt>  ::=  SUBROUTINE <id> ( <id list> )
 THERE WILL BE NO CONTEXT CHECK.

     6  <call stmt>  ::=  CALL <id> ( <exp list> )
 THERE WILL BE NO CONTEXT CHECK.


  <id> AS STACK TOP WILL CAUSE PRODUCTIONS TO BE CHECKED IN THIS ORDER:

     5  <id list>  ::=  <id list> , <id>
 THERE WILL BE NO CONTEXT CHECK.

     4  <id list>  ::=  <id>
 (0,1) CONTEXT WILL BE CHECKED. LEGAL RIGHT CONTEXT:
           ...)
           ...,

     9  <expression>  ::=  <id>
 THERE WILL BE NO CONTEXT CHECK.


  <call stmt> AS STACK TOP WILL CAUSE PRODUCTIONS TO BE CHECKED IN THIS ORDER:

     2  <statement>  ::=  <call stmt>
 THERE WILL BE NO CONTEXT CHECK.


  <expression> AS STACK TOP WILL CAUSE PRODUCTIONS TO BE CHECKED IN THIS ORDER:

     8  <exp list>  ::=  <exp list> , <expression>
 THERE WILL BE NO CONTEXT CHECK.

     7  <exp list>  ::=  <expression>
 THERE WILL BE NO CONTEXT CHECK.


  <subroutine stmt> AS STACK TOP WILL CAUSE PRODUCTIONS TO BE CHECKED IN THIS ORDER:

     1  <statement>  ::=  <subroutine stmt>
 THERE WILL BE NO CONTEXT CHECK.


TIME USED WAS 0.063 SECONDS.
TOTAL TIME IS 0.383 SECONDS.


ANALYSIS COMPLETE FOR ITERATION 1
 ** 2 ERRORS WERE DETECTED.


SYSTEM ?
```

図9

```
1  <statement>    ::=  <subroutine stmt>
2                    !  <call stmt>

3  <subroutine stmt>    ::=  <subroutine head> <id> )

4  <subroutine head>    ::=  SUBROUTINE <id> (
5                            !  <subroutine head> <id> ,

6  <call stmt>    ::=  <call head> <exp> )

7  <call head>    ::=  CALL <id> (
8                    !  <call head> <exp> ,

9  <exp>    ::=  <id>
```

図10. MSP(2, 1 ; 1, 1)文法の例

アナライザの出力したテーブルを用いて、テーブル駆動型の構文解析プログラムを作成し、各生成規則に対する意味付けを行なうプログラム(セマンティック・ルーチン)を作成すれば、その言

語のコンパイラができます。構文解析のアルゴリズム等は、文献(1)の Chapter 9, Appendix 5 を参照して下さい。

〔謝　辞〕 XXPL システム及び ACOS の使用法について御指導を頂きました電子技術総合研究所言語処理研究室　鳥居宏次室長、真野芳久氏、並びに大阪大学大型計算機センター　青井信一技官、後藤米子技官に感謝致します。

〔文　献〕

1) W. M. McKeeman, J. J. Horning and D. B. Wortman : "A Compiler Generator", Prentice-Hall, 1970.