プログラミング-言語・基礎・実践- 8－13
(1992. 8. 20)

図を用いた分散システム記述言語と分散システム自動生成システム

山之上 卓　早田 弘一　安在 弘幸
九州工業大学

現在、ワークステーションネットワークやパラレルコンピュータの普及により、分散システム上で動作するソフトウェアの需要が高まっている。分散システム上のソフトウェアを開発する際、プロセス間の通信の実態を人間が把握するために、よく図が作成される。従来は、この図を人間がプログラムに翻訳していた。これに対し、図を用いた分散システム記述言語と、この言語によって記述されたシステムをC言語に自動的に翻訳するプリコンパイラの開発を行った。本稿では、この言語の仕様を応用例とともに述べる。

A Figure Programming Language for Distributed System Discriptions and its Pre-Compiler

Takashi Yamanoue, Hirokazu Hayata, Hiroyuki Anzai
Kyushu Institute of Technology

A figure Language for distributed system descriptions and its pre-compiler are presented.

Recently, distributed computer systems, sets of computers connected by computer networks, are widely spreading. In designing a distributed system, the figure of message passing between objects of the system was usually drawn at first, and then translated into the programs by programmers. This paper proposes a figure language for distributed system descriptions , "FIGURE" and the pre-compiler for FIGURE, which automaticaly translates the descriptions in FIGURE into C programss.

1　はじめに

現在、ワークステーションネットワークやパラレルコンピュータの普及により、分散システム上で動作するソフトウェアの需要が高まっている。

分散システムは、ネットワーク上に分散配置されたプロセス間で通信を行うことによって、目的の処理を行う。このため、分散システム上のソフトウェアはＣ＋＋等のオブジェクト指向言語を使用すると自然に記述することができる。しかしながら、従来のオブジェクト指向言語も含めたプログラミング言語の多くは、一次元的に記号を並べたものであり、そのプログラムから直観的にシステムを把握することは難しい。

分散システム上のソフトウェアを開発する際、プロセス間の通信の実態を人間が把握するために、よく図が作成される。従来は、この図を人間がプログラムに翻訳していた。

これに対し、図を用いた分散システム記述言語と、この言語によって記述されたシステムをＣ言語に自動的に翻訳するプリコンパイラの開発を行った。

本言語は、オブジェクトベースプログラミング言語の一種である。オブジェクトベースプログラミング言語とは、オブジェクト指向の概念からインヘリタンス（継承）を除いた考え方である。ＡｄａやＭｏｄｕｌａ－２等の言語がこれらの言語の代表的なものである。

この言語は、－，＋，｜，＜，＞，ｖ，＾等の記号で二次元平面上の図形の構成を記述する。従って、ｅｍａｃｓやｖｉ等の一般的なエディタで記述できる。また、作成した分散システムの図をそのままプリントアウトすることが可能であり、さらにｅ－ｍａｉｌ等を通じて簡単に転送できる。

このプリコンパイラは、我々の研究室で開発された言語処理系の生成系言語ＭＹＬＡＮＧを使用し、構文的パターン認識の技法を用いて開発した。

2　図を用いた分散システム記述言語

本言語は、オブジェクトを表す矩形とオブジェクト間のメッセージパッシングを表す矢印で構成される。以下で、それぞれの説明を行う。

2．1　オブジェクト

オブジェクトベースプログラミング言語では、オブジェクトは私的な状態情報またはデータ、操作、手続きをカプセル化したものである。本言語におけるオブジェクトは、図２．１のように表される。なお、本言語で記述されるオブジェクトにより生成されるプロセスの数はスタティックであり、ダイナミックに生成、消滅するオブジェクトはない。

```
+-----------------------------+
|object object_name           |
|                             |
|include                      |
|  [C言語によるインクルード  |
|    ファイルの定義]$         |
|                             |
|functions                    |
|  [C言語による関数の定義]$  |
|                             |
|var                          |
|  [変数宣言]$                |
|                             |
|init                         |
|  [C言語のプログラム]$       |
|                             |
|operations                   |
|  list_of_input_message      |
|    >> list_of_output_message |
|  [C言語のプログラム]$       |
+-----------------------------+
```

図２．１　オブジェクト記述形式

図に対する説明を簡単に行う。

object_name

オブジェクトの名前をここで定義する。これは、プリコンパイル後に生成されるＣ言語のプログラムのファイル名となる。

include

'[' と']$'の間に定義されたものを、そのプログラムのヘッダファイルとしてプリコンパイル後のプログラムの中で宣言する。ただし、標準的なＣ言語のヘッダファイルは定義ずみなので、再定義する必要はない。

function

'[' と']$'の間に定義された関数（手続き）をプリコンパイル後のプログラムに記述する。

var

'[' と']$'の間に定義された変数等の宣言をプリコンパイル後のプログラムの中に変数宣言として記述する。ただし、後述する通信に必要な変数に対しては、その宣言の必要はない。

init

'[' と']$'の間に定義されたC言語の記述を、プリコンパイル後のプログラムのメイン関数の中に記述される。

operations

list_of_input_message , list_of_output_message で定義される変数において、メッセージパッシグが行われる。実際のメッセージの送信または受信は以下のような命令で行われる。

```
put_messeage(output_message_name,
                address_of_variable)
get_messeage(input_message_name,
                address_of_variable)
```

また、メッセージの受信の命令として次のようなものがある。

```
get_messeage_nw(input_message_name,
                    address_of_variable)
```

前者がメッセージが送られてくるまで待っているのに対し、後者はメッセージが送られているかどうかにかかわらず実行が進む。これらの命令は、C言語の記述とともに'[' と']$'の間に記述され、プリコンパイル後のプログラムのメイン関数の中に記述される。前に示したinitとの違いは、initで定義されたの記述が初期化等の1回だけの命令の実行に対し、operationsによって定義された記述は無限ループの中に記述されることである。

2．2　オブジェクト間のメッセージの送受信

オブジェクト間の通信は、図2．2のように記述される。ここでは、Object Aがメッセージを送り、Object Bがメッセージを受け取るという形になっている。つまり、矢印は通信の方向を示している。さらに、矢印の上に付記されているtypeによりメッセージの型を、message_nameにメッセージの変数を宣言することができる。基本的には、一つの矢印に対して宣言できる型または変数は一つだけである。また、矢印の意味はメッセージの送受信であってオブジェクトから他のオブジェクトの呼び出しではない。

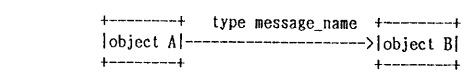

図2．2　図におけるメッセージパッシング

矢印の記述形式としては、－，｜で矢軸を、＜，＞，v，＾で対応する方向の矢先を記述する。また、矢軸の方向を（例えば横方向から縦方向へ）変更したい場合は、その曲がり角の位置に＋を記述する。さらに、矢軸と矢軸が重なる場合には、重なる位置に＃を記述する。これらの記述例を図2．3に示す。

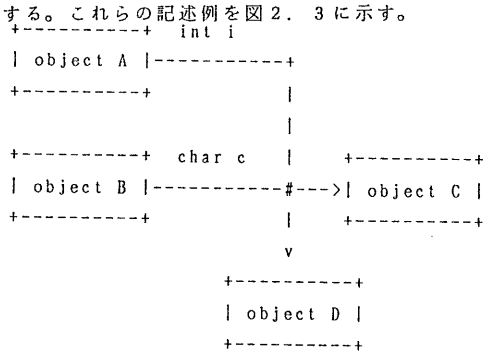

図2．3　様々な矢印の記述例

2．3　言語の記述例

この言語により記述されたプログラムの例を図2．4に示す。図2．4に示しているプログラムは、マンデルブロ集合を表示するプログラムである。

このプログラムで、"host[]={...}"は図に表されたオブジェクトが、それぞれどのホストに割り付けられるかを示している。もう一つのプログラム例を図2．5に示す。これは、ガウス掃き出し法のプログラムである。

3　言語処理系の生成系言語ＭＹＬＡＮＧ

前に述べた通り、作成したプリコンパイラは我々の研究室で開発した言語処理系の生成系言語ＭＹＬＡＮＧを使用している。ここでは、ＭＹＬＡＮＧの性質、文法を簡単に説明する。

ＭＹＬＡＮＧは、言語処理系の生成系言語（コンパイラ・コンパイラ）であり、属性付き正規翻訳記法（ＡＲＴＦ）により記述されたプログラムをＭＹＬＡＮＧに通すことにより、属性付き正規翻訳記法により定義されたコンパイラを作成することができる。このコンパイラは、ソースプログラムからオブジェクトプログラムを生成する（図3．1）。このことを利用し、図を用いた分散システム記述言語の処理内容を属性付き正規翻訳記法で記述し、それをＭＹＬＡＮＧに通すことによって図言語のプリコンパイラが生成される。このプリコンパイラは、図によって記述された分散シ

```
/* system parmandel */
/* objectname and hostname */
hosts[]={
{{"m_dealer","hasu"},
 {"m_consumer[0]","renge"},
 {"m_consumer[1]","yuri"},
 {"m_consumer[2]","ayame"},
 {"m_consumer[3]","kiku"},
 {"m_collector","sumire"},
 {"m_display","hagi"}
};
#define N=4
#define PIXELS 700
struct out_rec  {int x1,y1,x2,y2,p;};
```

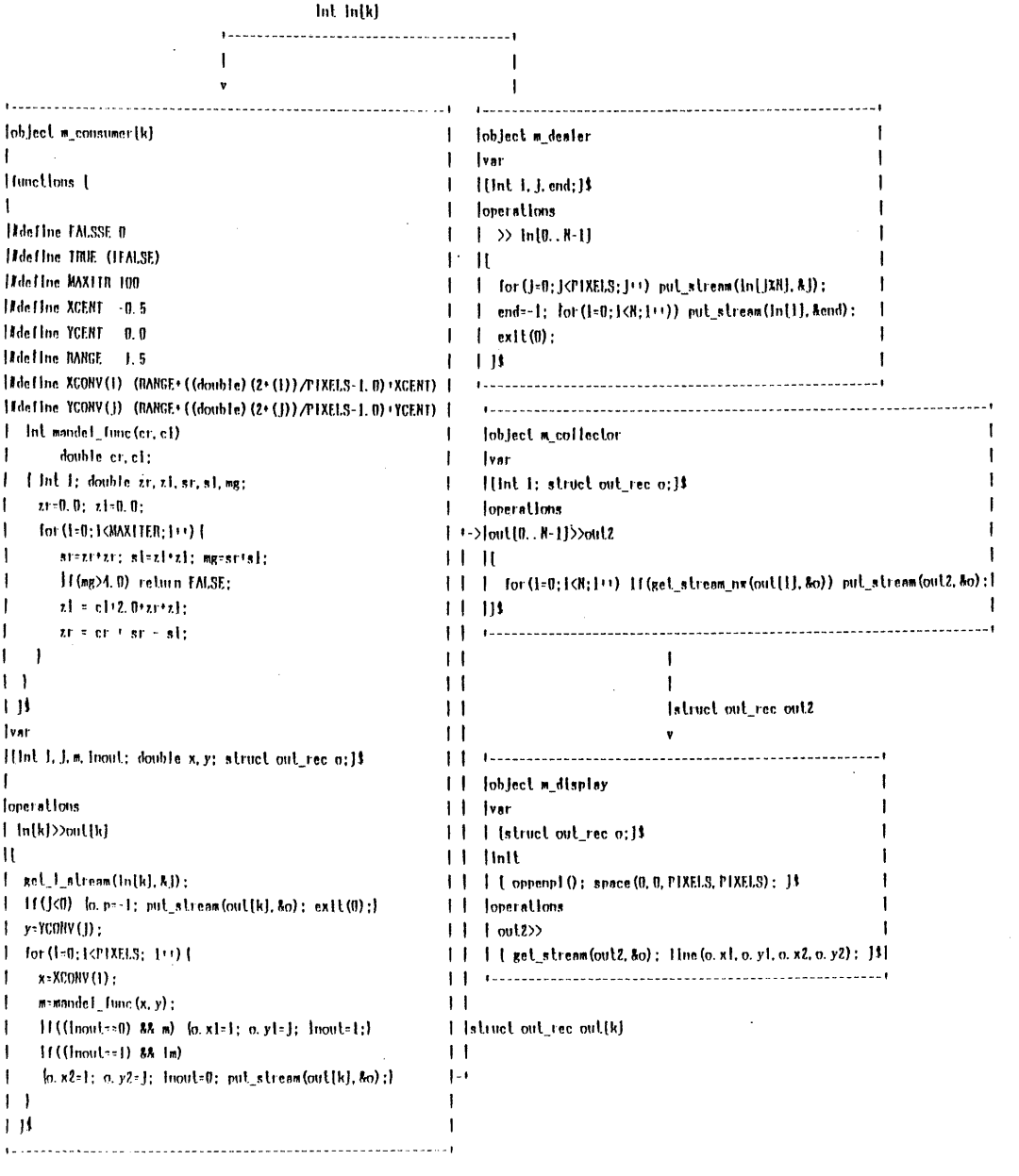

図２．４　マンデル集合表示プログラム

```
/* system gauss */
/* list of objectname-hostname */
hosts[]={ {"controller", "rindo"},
          {"worker[0]",  "kikyo"},
          {"worker[1]",  "keshi"},
          {"worker[2]",  "akane"},
          {"worker[3]",  "azami"},
          {"IBUS",       "hagi"},
          {"dBUS",       "hasu"} };

#define N 10
#define M 4
```

```
+-------------------------------------------------------------------------+
|object controller                                                        |
|                                                                         |
|var [int i,j,k,end=-1,dm; double a[N][N+1]; ]                            |
|                                                                         |
|init [ for(i=0;i<N;i++) for(j=0;j<N+1;j++) scanf("%lf",&(a[i][j])); ]    |
|                                                                         |
|operations                                                               |
|   i_in[0..k..M-1] >> i_out[0..k..M-1], i_out[0..k..M-1]                 |
|[                                                                        |
|  /* deal input data */                                                  |
|  for(i=0;i<N+1;i++){                                                    |
|     for(j=0;j<M;j++) put_stream(i_out[j],&i);                           |
|     for(j=0;j<N;j++) put_stream(d_out[i%M],&(a[j][i])); }               |
|  for(i=0;i<M;i++) put_stream(i_out[i],&end);  /* end of deal */         |
|                                                                         |
|  /* deal index */                                                      |
|  for(i=0;i<N+1;i++){                                                    |
|     for(j=0;j<M;j++) put_stream(i_out[j],&i);                           |
|     for(j=0;j<M;j++) get_stream(i_in[j],&dm);  /* sychronizing */       |
|  }                                                                      |
|  exit( 0);                                                              |
|]                                                                        |
|                                                                         |
+-------------------------------------------------------------------------+
     |                    |                    ^
     |                    |                    |
     |int i_out[0..M-1]   |double d_out[0..M-1]|int i_in[0..M-1]
     |                    |                    |
     v                    v                    |
+-------------------------------------------------------------------------+
|object worker[K]                                                         |
|                                                                         |
|var [ int i,j,k,mind,end=-1; double a[N][N+1],max,p ; ]                  |
|                                                                         |
|operations                                                               |
|  i_out[K],d_out[K],IBUSr[K],dBUSr[K]  >>  i_in[K],IBUSx[K],dBUSx[K]     |
|[                                                                        |
|   /* input data */                                                      |
|   [  get_stream(i_out[K],&j); if(j<0) break;                            |
|      if((j%M)==K) for(i=0;i<N;i++) get_stream(d_out[K],&(a[i][j])); ]    |
|                                                                         |
|    /* solve the linear equations */                                     |
|   [  get_stream(i_out[K],&i);  /* get current index */                   |
|                                                                         |
|      if((i%M)==K) {  /* I have the pivot */                             |
|          mind=i; max=a[mind][i];                                        |
|          for(j=i+1; j<N; j++)                                           |
|             if(max<fabs(a[j][i])) {max=a[j][i]; mind=j; }               |
```

```
|           put_stream(iBUSx[K],&mind);                                    } |
|       else get_stream(iBUSr[K],&mind); /* I don't have the pivot */        |
|       if(i!=mind) {                                                        |
|           if((i%M)==K)  printf("swapping raw %d-%d¥n",i,mind);            |
|           for(j=i; j<N+1; j++) if((j%M)==K)                                |
|              {w=a[i][j]; a[i][j]=a[mind][j]; a[mind][j]=w;}            }  |
|                                                                            |
|       if((i%M)==K) {  p=a[i][i]; put_stream(dBUSx[K],&p); }                |
|       else  get_stream(dBUSr[K],&p);                                       |
|       for(j=i; j<N+1; j++) if((j%M)==K) a[i][j]=a[i][j]/p;                 |
|       for(j=0;j<N;j++) if(i!=j){                                           |
|           if((i%M)==K) { p=a[j][i]; put_stream(dBUSx[K],&p); }             |
|           else get_stream(dBUSr[K],&p);                                    |
|           for(k=i;k<N+1;k++) if((k%M)==K) a[j][k]=a[j][k]-p*a[i][k]; }     |
|                                                                            |
|       put_stream(i_in[K],&end); /* end of the current index */             |
|                                                                            |
|       if(i==(N-1)) {  /* end of the computation */                         |
|           /* print out result */                                           |
|           if((N%M)==K) for(i=0;i<N;i++) printf("%lf¥n",a[i][N]);           |
|           exit( 0);                                                 }   }  |
|]                                                                           |
+----------------------------------------------------------------------------+
   |      ^            |               ^
   |      |            |               |
   |      |            |int iBUSx[0..M-1]  |int iBUSr[0..M-1]
   |      |            |               |
   |      |            v               |
   |      |     +--------------------------------------------------------------+
   |      |     |object iBUS     /* software BUS (int) */                      |
   |      |     |                                                              |
   |      |     |var [int i,j,x;]                                              |
   |      |     |                                                              |
   |      |     |operations                                                    |
   |      |     |   iBUSx[0..i..M-1] >> iBUSr[0..i..M-1]                       |
   |      |     |[                                                             |
   |      |     |   for(i=0;i<M;i++)                                           |
   |      |     |     if(get_stream_nw(iBUSx[i],&x))                           |
   |      |     |       for(j=0;j<M;j++) if(j!=i) put_stream(iBUSr[j],&x);     |
   |      |     |]                                                             |
   |      |     +--------------------------------------------------------------+
   |      |
   |      +-----------------+
   |                        |
   |                        |
   |double dBUSx[0..M-1]    |double dBUSr[0..M-1]
   |                        |
   v                        |
+------------------------------------------------------------+
|object dBUS     /* software BUS (double) */                 |
|                                                            |
|var [int i,j; double x]                                     |
|                                                            |
|operations                                                  |
|   dBUSx[0..i..M-1] >> dBUSr[0..i..M-1]                     |
|[                                                           |
|   for(i=0;i<M;i++)                                         |
|     if(get_stream_nw(dBUSx[i],&x))                         |
|       for(j=0;j<M;j++) if(j!=i) put_stream(dBUSr[j],&x);   |
|]                                                           |
+------------------------------------------------------------+
```

図２．５　ガウス掃き出し法のプログラム

ステムのプログラムからＣ言語のプログラムを生成する（図３．２）。

次に属性付き正規翻訳記法（ＡＲＴＦ）について簡単な説明を示す。例えば、このプリコンパイラ生成を行うプログラムの一部を以下に示す。

```
＜左右（ｓｘ，ｓｙ，ｉ／ｓｘ，ｓｙ）＞
  ＝＜文字チェック（@－，ｓｘ，ｓｙ／）＞
    （＜文字チェック（@－，ｓｘ，ｓｙ／）＞
    ［（ｓｙ：＝ｓｙ－ｉ）］）＊；

＜上下（ｓｘ，ｓｙ，ｉ／ｓｘ，ｓｙ）＞
  ＝＜文字チェック（@｜，ｓｘ，ｓｙ／）＞
    （＜文字チェック（@｜，ｓｘ，ｓｙ／）＞
    ［（ｓｘ：＝ｓｘ－ｉ）］）＊；
```

上の二つの定義式は、それぞれ記述された図の線を左右または上下にたどることを定義している。ＭＹＬＡＮＧの特徴の一つとして、このように定義式で使用する記号に漢字を使用できる。

次に、これらの定義式の簡単な説明を行う。

＜．．．＞は、非終端記号である。（．．．／．．．）は、この非終端記号が属性を持つことを表し、／の左の名前は相続属性の生起、右の名前は合成属性の生起をそれぞれ表す。属性を持たない場合には、何も記述しない。定義式右辺で記述されている（．．．）＊は閉包と呼ばれ、括弧の内容の０回以上の繰り返しを意味する。つまり、括弧の中にある最初の判定文または非終端記号が偽になるまで繰り返される。

非終端記号＜左右（．．．）＞では、定義式右辺に

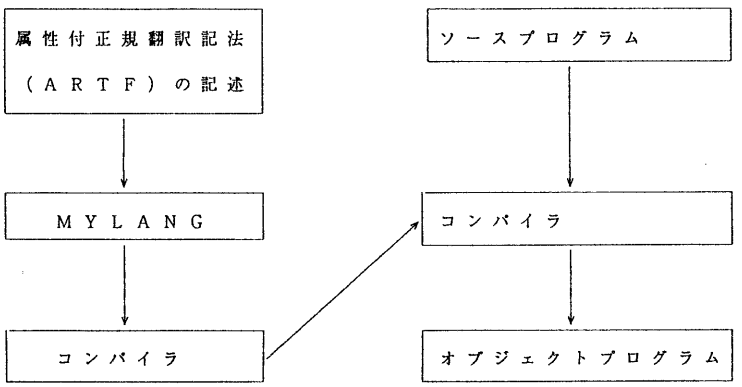

図３．１　MYLANGによるプログラムの生成

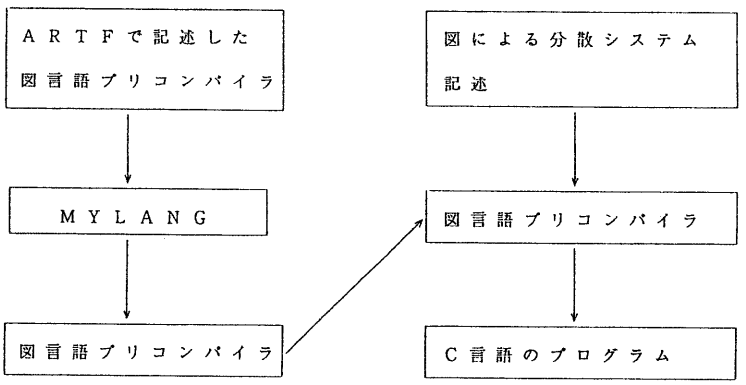

図３．２　MYLANGによるプリコンパイラの生成

最初に記述されている＜文字チェック（．．．）＞の評価を行う。ここで、ｓｘ，ｓｙはそれぞれ入力された図の座標を表し、ｉは左または右のどちらに移動するかのパラメータである。＜文字チェック（．．．）＞はこのプログラムの別の所で定義された、入力された図上の文字の判断を行う非終端記号である。つまり、＜文字チェック（@－，ｓｘ，ｓｙ／）＞は、ｓｘ，ｓｙという座標上にある文字が'－'であれば真（true）となり、それ以外の文字であれば偽（false）となる。この非終端記号が真であれば次の記号の評価を行い、偽であれば次の記号の評価を行わない。（．．．）＊の中の動作の内容は、ｓｘ，ｓｙの座標の文字が'－'であれば、ｓｙからｉを引き（足し）座標が左（右）に移動した場所において、同様の処理を行い、ｓｘ，ｓｙの座標の文字が'－'でなくなるまで繰り返される。

これらの処理により、＜左右（ｓｘ，ｓｙ，ｉ／ｓｘ，ｓｙ）＞を評価することにより、入力された図の横線を認識することができる。＜上下（ｓｘ，ｓｙ，ｉ／ｓｘ，ｓｙ）＞も、同様にして入力された図の縦線を認識する。

4　おわりに

この言語は、オブジェクトベースプログラミング言語の概念を基に作成したものである。故に、クラス間の継承等の概念はない。現在、本言語へのクラスの導入を検討している。

なお、現在この言語を使用した並列コンピュータのプログラミングへの応用、ＷＩＤＥ，ＪＡＩＮ等の広域ネットワーク上のソフトウェアの開発への応用を考えている。

謝辞

本研究は、文部省科学研究費No.04855072 の補助を受けている。

参考文献

［１］山之上　卓・安在　弘幸：
「属性付構文指示翻訳系の生成系ＭＹＬＡＮＧ」
情報処理学会論文誌，Vol.26，No.1，pp.195-204
(1985)

［２］安在　弘幸・山之上　卓：
「言語処理系の生成系ＭＹＬＡＮＧの基礎概念」
電気通信学会論文誌(D)，Vol.J69-D，No.2，
pp.117-127(1986)

［３］山之上　卓：
「言語処理系の生成系に関する研究」
昭和５８年九州工業大学修士論文

［４］KING SUN FU：
「Syntactic Pattern Recognition
and Applications」
Prentice-Hall(1982)

［５］ROGER S. CHIN・SAMUEL T. CHANSON
「Distributed Object-Based
Programming System」
ACM Computing Surveys，Vol.23，No.1，March 1991