入出力条件によるプログラムの合成

7N－1

恐神正博　西田富士夫
福井工業大学

1．まえがき

プログラムとは入出力条件などからなる情報処理の要求に基づき、処理単位の命令や関数モジュールを組み合わせて作成した手続きであり、プログラミングとは与えられた情報処理の要求に対して既知の処理単位をつないで要求の処理を実現する問題解決の手続きである。

このような観点で従来から、自動合成の研究やプログラムの検証などの基礎的研究が行われてきた。近年ではメニュー方式とGUIの利用により、プログラムの生産性は大きく向上しているが、命令やモジュールの選択配置などは、なお、大きく人手に依存しているようであり、これらの自動化が望まれる。また、概略設計から詳細化のための条件を与え設計やプログラム合成を行うときに自動リンクの手法は有用である。ここではプログラム単位をリンクしてプログラムを合成する基本的な事項についてプロログを用いて考察する[1]。

2．入出力条件

処理の要求仕様は入力条件と出力条件とに分かれる。入力条件とは(1)(2)のように処理に必要な対象のデータ構造などの属性に関するデータタイプ条件や存在場所や前処理などに関する入力条件である。

```
dt(n,[var,int]).          ①
dt(a,[ar(n,200),int]).    ②        (1)
in([in_kb,n]).            ①
in([in_fl('"in_f12"'),b]).②
in([in_mem,i]).           ③
                                   (2)
```

ここに dt(X,Y)はXのデータタイプはYであることを示す。(1)の②のタイプは最大200の要素からなる大きさnの整数型の1次元配列であることを表す。in([L,X])はデータXがLに存在する，あるいは与えられることを表す。(2)の①②③はそれぞれのデータがキーボード、ファイル"in_f12"、メモリに与えられていることを表す。これらはプロログにおいて事実を述べる事実節の形で表す。

出力条件は背理法を変形した質問節の形で表す。例えば[a,の(配列要素の)和を,t_sum,に求める],[t_sum,を印字する]と指定するのに次のような質問節の形で表す。

```
?:-out([sum,a,t_sum]),out([print,t_sum]).
                                   (3)
```

ここに?:-out(X),out(Y).はXを出力し次にYを出力する手続きを求める質問節である。

3．処理の記述と入力条件の探索

これらの質問節に答えて、所要の処理をリンクしてプログラムを自動的に作成するには関数などの一まとまりの処理毎に手続きの後件条件、いくつかの前提条件と手続き表現を規則節の形で用意する。以下例により説明する。(4)は一次元配列の要素の和を求める処理を記述する規則節である。

```
out([sum,X,TSUM]):-
    d(X,[ar(N),TYPE]),in([in_mem,X]),     ①
    proc(sum,[[ar(N),TYPE],X],TSUM),      ②
    add_type_entity_list([[X,TYPE],
     [N,int],[T_SUM,TYPE]]).              ③
                                          (4)
```

(4)は出力条件が成立するためには、①により入力条件が成立することが必要で、Xのデータタイプが大きさ,N,タイプ,TYPEの一次元配列であることの宣言や、Xが初期設定、読み込み、乱数発生などにより、メモリの中に取り込まれてin_memが成立することが必要であることを示す。これらの条件が成立するときには、②のproc述語により、処理呼出し表現　t_sum=sum(a,n); の書き出しと、XのタイプTYPEのカスタマイズ後、関数定義sumの関数定義リストへの取り込みを行う。また、③においては、この処理においてタイプ宣言が必要な変数名とタイプをタイプリストに加え記憶する。タイプリストへの登録では既に登録され

A program Construction by Automatic Module Linking
Masahiro Osogami and Fujio Nishida
Fukui University of Technology

ているか二重定義はないかなどのチェックを行う。

さて各処理において前提条件が成立することが必要であるが成立しない場合は、次の規則により前処理を順次求めてプログラムを作成する。

```
in(Z):-state(S),member(Z,S).          ①
in(Z):-out(Z),add_state(Z).           ②
                                      (5)
```

①はZという状態に既になっているか、あるいはZに関連する処理が行われその状態になっているか否かをZが既に状態集合Sのメンバになっているか否かを調べてチェックし、①が成立すれば元に戻って次の処理に進む。①が成立しなければout(Z)によりZの処理を追加し、Zを状態Sに加え、Zに関する前提条件を成立させる。

(1)(2)の入力条件のもとに(3)のような出力条件を与えるとMAPPは出力条件と単一化可能な出力述語を頭部にもつ規則節を探索し入力条件を調べる。与えられた入力条件と単一化できれば、proc述語により手続き呼出関数が出力される。そうでなければこの規則節の適用条件を満たすための前処理が(5)により再帰的に探索され、(2)の入力条件により前提条件が保証されるまで実行される。探索に成功すれば前処理すなわち処理の呼出関数が各出力述語のproc述語により正順に書き出され手続き部のプログラムが求められる。続いて全プログラムを求める述語の入力により、インクルードすべきヘッダ名、ユーザ定義の関数の定義部、変数のタイプ宣言、手続き部などの全プログラムを、インクルードリストや前述の関数定義リストやタイプリストなどを用いて書き出す。インクルードリストには、proc述語の処理に必要なヘッダ名がproc述語の呼出毎に取り込まれる。

このような処理は順次処理だけではなく、選択、繰り返しなどの処理にも適用することができる。例えば、引数ARGに条件Tが成立するとき、処理を行ってOUTの状態になることを表す規則節は次のように表すことができる。

```
out([if(test([T,ARG])),OUT]):-
    in([in_mem,ARG]),
    if(test([T,ARG])),begin,out(OUT),end.
                                      ①
out([forall([I,from(M),to(N),step(1)]),
 out(PROC)]):-
    in([in_mem,M]),in([in_mem,N]),
    add_state([in_mem,I]),
    repeat_head(I,[M,N,1]),begin,out(PROC),
    end.                              ② (6)
```

選択処理の条件部の記述が、整数の素数判定のように複雑のときには(6)①のようにif述語引数部をtestとし、これから条件部の目的言語による書き出しと、真偽判定の関数定義の取り込みを(4)のproc述語と同様な方法で行う。

4．日本語文による処理

仕様における入出力条件の記述や生成された処理のコメント文には誤りが混入しない限り日本語文などが使用できることが望ましい。このため処理の種類毎に日本語文などによる関数呼出文の一覧表と、(7)のように日本語文表現と述語表現との間の変換辞書を設ける。

```
conv_dict([
jp([X,大きさ,N,最大値,M,の一次元配列でﾀｲﾌﾟは,
    TYPE,である],dt(X,[ar(N,M),TYPE])),
jp([ﾃﾞｰﾀ,X,はkbから与える],in([in_kb,X])),
jp([X,の和を,TSUM,に求める],out([sum,X,TSUM])
 ,...]).                              (7)
```

関数呼出文一覧表を用いて作った入力条件や出力条件は、１文毎に述語convの引数部に取り込み、変換辞書を参照して(8)により述語形に変換してファイルに書き出し、目的プログラムを生成するために再びメモリに読み込んでデータベースに加える。

```
conv(X):-conv_dict(D),member(jp(X,Y),D),
        write(Y),nl.                  (8)
```

生成されたプログラムには、理解を容易にするため関数(呼出)表現に対応して(7)の変換辞書を利用して、その日本語文表現も出力する。このため(4)のような各out(X)述語の規則節の本体部にadd_com(X)を設け、(9)により、out述語の手続き呼出部をプログラムに取り込む毎に、Xに対応する日本語文をcom_list述語の引数部に取り込む。

```
add_com(X):-com_list(CL),conv_dict(CD),
 member(jp(J,out(X)),CD),append(CL,[J],NCL),
 retract(com_list(CL)),assert(com_list(NCL)).
                                      (9)
```

【参考文献】

[1]恐神正博,西田富士夫:プロログによるモジュールの検索とプログラムの合成,情報処理学会第43回全国大会,pp.4-293-294(1991.10)