仕様の記述とプログラムの作成

2J−8

恐神正博　西田富士夫
福井工業大学

要求仕様についてはそれぞれの応用分野において、処理の内容に即して有用な手法が、いくつか提案されている。この報告は仕様の作成の始めに、処理関連対象の計算機内における型やデータ構造を選定してデータベースに登録したのち、これらの処理対象に、モジュールライブラリーに設けられている基本的な処理ルーティンや関数プログラムを適用して要求する処理を施したり、要求するデータを出力する仕様を作成し、仕様からプログラムを半自動的に作成する一つの手法とその実験システムMAPP(Module Aided Programming system by using Prolog)についてその概要を述べる。仕様の言語及びその処理にはプロログを用いる。これはPROLOGが仕様の機械可読性の他、仕様のトップダウン的記述、モジュールの検索、取り込み、リンクなどプログラム作成の自動化に比較的適した言語であるからである。

処理対象の型や構造を始めに選定するのは、入出力などの標準的な処理は、対象の型やデータ構造に応じてほぼ決まっており、仕様で指定したデータ構造に応じて既成の標準的なモジュールをカスタマイズしてを再利用するのが効率的であるからである。

1．　仕様における対象の型と処理

仕様の記述は型定義、対象リスト、処理からなり

```
specifications:-typedefinition,objectlist,
                    processing.          (1)
```

と表すものとする。

対象Xが構造体やレコードの場合には、いろいろの型があり、型定義が必要である。これらの型を次のように宣言する。

```
typedefinition:-
struct_type(X,memberlist([[OBJ1,TYPE1],...,
            [OBJn,OBJn])).               (2)
```

ここにXは型の名前であり、memberlistの引き数部は構造体などの各メンバの名前OBJiと型TYPEiの対のリストである。

対象リストobjlistはいくつかの対象からなり、対象
object(NAME,type(TYPE),array(size(N),index(I))
のNAMEは対象の名前、TYPEは対象の型（整数型などの値域名または構造体名）、NやIは対象が配列の場合、大きさや添え字変数名を表す。これらの仕様は

Module_based Specifications and Program Generation

Masahiro Osogami and Fujio Nishida
Fukui Institute Tecnology

:?-specifications.　により逐次読み込まれ、対象リストの各対象の名前が型などとともに、(3a),(3b)などを用いてデータベースに書き込まれる。(3a),(3b)は対象がスカラであるときの書き込みのプログラムで、対象名が既にデータベースに登録されているかどうかをチェックし、登録されていなければデータベースに型とともに書き込むプログラムである。

```
obj(X,type(TYPE)):-member_check(X,TYPE),!.   (3a)
obj(X,type(TYPE)):-type_member(TYPE,Y),
                   add_entity(X,TYPE,[X|Y]),
                   assert(type(X,TYPE)),
                   write(TYPE),write(' '),
                   writelist([X|Y]).         (3b)
```

ただし、大文字からなる文字列は変項、小文字からなる文字列は定項を表す。上式は、Xをassert述語によりデータベースに書き込むとともに、型がTYPEのtype_memberに加え、これをディスプレイに表示することを指定する。配列や構造体の場合も同様であるが、配列のサイズ大きなどの書き込みも同時に自動的に行っている。処理の仕様においては処理の概略を表す述語表現を処理の順に記述する。

```
processing:- process_name1(object_name1,goal1),
             ..........................  ,
             process_namen(object_namen,goaln).(4)
```

ここにprocess_nameなどは処理名で引き数部に対象名や2，3の属性値を指定するだけで、システムは仕様で与えた対象の型やデータ構造を参照して、適用可能なproceduresや関数を含むモジュールを探索し詳細化する。

2．　日本語仕様から形式仕様への変換

前節で述べた仕様は、機械による仕様の矛盾チェックや、仕様からのプログラム生成の自動化のために機械可読な形にすることが望ましい。形式仕様は機械可読であり、暗黙の省略と解釈により、ある範囲のプログラムと対応するが、習熟しなければ一般に読みにくく書きにくい。プロログは構文上の制限が少なく、通常のプログラムでさえもコメント無しでは分かりにくいことが多く、これを形式仕様に用いるときには注意が必要である。

ここでは仕様の日本語的表現として、プロログの述語記号や関数記号や引き数部をオプションで日本語の字句などに置き換えたものを用いるものとする。また、引き数が述語や操作の対象格でただ1個の場合はよいが、それ以外の引き数を含むときには、読み易さや書き易さを増

すために引き数に、適宜、役割名をつけたり、後置詞をつけることとする。

[役割名：,引き数,...] (5a)　役割名(引き数), (5b)

[引き数,　後置詞,...] (5c)

とする。(5a)は引き数部のリストの各要素の引き数に役割名：を前置し、(5b)は役割名を関数記号的に付けている。後置詞を付けるには、引き数の後に(5c)のようにリストの要素の形で、を、に、から、で、をキーとして、などの後置詞を置く。後置詞は役割名と異なり、自然に役割を区別して表し、憶え易く使いやすい。

日本語表現の"述語記号_i(X)"を1バイト系の形式表現に機械的に変換するためには、述語記号（や関数記号）毎に

```
述語記号_i (X) :-tran(X,XE),
        引き数部の書き出し述語(pred_i(XE)).   (6)
```

のような形の変換ルールを設ける。

ここに"述語記号_i"とこれに対応する"pred_i"はそれぞれの述語記号や関数記号毎に与えた2バイト系及びその1バイト系表現の述語記号とする。また、tran(X,XE)は日本語の2バイト系の語Xを1バイト系の英字に変換したものがXEであることを示す。このような形の変換規則を与えておけば、後は引き数の変数名の2バイト系1バイト系間の単語変換辞書を設けることにより、1バイト系表現がえられる。

以下に2、3の変換ルールの例を示す。

```
対象('名前:',X,'型:',構造体,Y,'組成:',
      配列([大きさ(N),インデックス(I)]))  :-
  tran(X,XE),tran(Y,YE),
  writelist(['obj(',XE,',','struct(',YE,')',',',
  'array([size(',N,')',',',',index(',I,')])).']),nl.
                                          (7a)
平均('対象:',X,'出力先:',Y):-
      tran_list([[X,XE],[Y,YE]]),
      gen_w([av_fc,['obj:',XE,'goal:',YE]]).   (7b)
```

(7a)は構造体配列の宣言、(7b)は平均を求める述語の2バイト系から1バイト系表現への変換規則をそれぞれ示す。(7a)のwritelist(X)は引き数部のXを書き出す述語である。

3．モジュールの検索と詳細化

仕様はモジュールの見出し表を用いてモジュールの諸属性を自動的に検索し、詳細化される。次にモジュールの見出し表の構成を説明する。

図1はモジュール見出し表における主要なモジュール属性項目記述の例である。

各見出しはモジュール毎にfile(H,T).の形式をとり、HやTの引き数部はリスト形式をとる。Hはモジュールの機能の種別を表す大見出しや、モジュール番号からなる。Tは以下のようないくつかの関数形の複合項からなる。これらは、モジュールの関数の型と機能を日本語で説明するcomment項目、この関数をコールするfunction項目、引き数部の処理対象や関数の構造や型を記述するtype項目、このモジュールを適用可能にするために準備すべきデータの入力や性質などの条件を記述するinput項目、処理の結果出力されるデータの名前や性質を記述するoutput項目、このモジュールを収納するfile_name項目などである。

```
file([ソート,12],
[com(N行M列の浮動小数点型2次元配列Aを第K列をキーとして昇順にソートする),
funct(sort_2ar_f_as(A,K,N,M)),
type([int(sort_2ar_f_as),float([[A,50,5]]),
      int([K,N,M])]),
in([read([[A,N,M]]),read([K,N,M])]),
out([sort([[A,N,M]],K,as)]),
file_name(srt2afas)]).
```

図1

概略仕様において対象名と操作の種類op_kindを指定すると、MAPPは仕様で与えた対象の型やデータ構造を参照して、適用可能な関数を検索する。型を参照した関数検索プログラムの主要部は

```
op_kind(OBJ_NAME):-member(op_kind,H),file(H,T),
        pobj(OBJ_NAME,type(TYPE)),
        member(type(TYPE),T),
        member(funct(FUNCT),T),
        writelist (FUNCT),write(';'),nl,
        member(com(COM),T),write(com(COM)).   (8)
```

モジュール見出し表にはモジュールの適用条件であるIN項目、処理後の状態や出力データを記述するOUT項目を設けているので、これらの知識を用いれば、入出力条件を満たすプログラムをモジュールのリンクと単一化によって部分的に合成することが原理的に可能である[2]。従って、入出力条件とともに計算機にさせたい処理の概要$proc_1,\ldots,proc_n$を

```
proc:-proc1,.....,procn.  |?-proc.          (9)
```

で仕様に与えれば、MAPPは反駁法により、state述語に蓄えられた入力条件と$proc_1$までにえられた中間結果を用いて、$proc_{i+1}$の入力条件を満たす出力をもつモジュールを適宜補いながら、$proc_{i+1}$に対するプログラムを生成する。すなわち、定形的な前処理などはMAPPが自動補填するので、オプション付きの主要な処理を中心に概略仕様を与えてよいこととなる。

参考文献

[1] 西田,高松:プロログによるプログラム生成システム　情報処理学会第42回全国大会5R−2(1991)

[2] 恐神,西田:プロログによるモジュールの検索とプログラムの合成