8J−6

仕様からの構造化図の簡易作成とプログラムの合成

恐神正博　西田富士夫
福井工業大学

箇条書きにした日本語風の仕様から構造化図を簡単に作成する方法を述べる。なお、各概略仕様文に含むキーワードから、これをステートメント、モジュールなどの見出し部に含む日本語文を検索し、これをカスタマイズしたものを詳細化して、Cなどのプログラムを合成する方法について述べる。

1．構造化図の作成

構造化した仕様を作成する場合、仕様の各文を単にインデントするだけでなく、保守などにも利用すべく、制御文に図式記号やレベル線を導入して、制御構造をより分かりやすく視覚化して作成する支援システムが各方面で開発されている。これらのシステムは人手で操作して作図するためモジュール部品などの配置には自由度はあるであろうが、制御記号や位置の指示など手間がかかる。

この講演では、各仕様文の先頭に附加したif, then, forなどのキーワードを利用して、システムが自動的に構造化図を描く手法について述べる。なお、従来、開発された構造化図の中、日本電気で開発されたＳＰＤが仕様文の挿入字数などの制限が少なく自動化し易いシステムの1つであり[1]、本MAPP システムでもＳＰＤと同様な出力を得ることを目指した。

プログラム仕様はprog([H1,H2,...]).のように日本語文や数式などよりなるいくつかの仕様文H1,H2,...,のリスト要素として与え、各Hi(i=1,2,..)は次の規則式により、ステートメントまたはモジュールとしてとして処理される。

```
prog([H|T]):-(statement(H);module(H)),prog(T).
prog([]).                                        (1)
```

プログラムはレベル0のステートメント startに始まり、レベル0のステートメントendで終わるものとする。

if文while文 for文などの制御仕様文は、アルファベットのキー文字列や＜＞、○などの記号を文頭におき、制御の種類を簡潔に表し、また、引き数部の条件部を文の右端に書いて見易さをはかっている。

一般に、仕様文は前の仕様文から指定した個数の縦線分を隔てて下方に書くが、仕様文の文頭にif, then, else, for, while などの制御のキーワードが現れるときには、次の処理本体部は下方移動の他に仕様文のレベルを1レベル増し、文頭を1レベル右に移動して書く。

```
|--start--▽
|--プロンプトにより,n,を読み込む
|--if--<>--もし,n<10,なら
|        |-then-|
|        |      |--計算する(kingaku=･･･)
|        |
|        |-else-|
|        |      |--計算する(kingaku=･･･)
|        |-end_if
|--n,をプリントする
|--end--△
```

図1．ＳＰＤ図の例

逆に次のステートメントがend_if, end_forなどであるか、thenからelseに移る場合にはレベルが1レベル減少する。

レベルＬ文のレベル線は0レベルから現在の文のレベルまで(L+1)本縦方向に描く。次の規則は文Pのレベルが Lでレベル間の水平間隔をIS字分とするとき、write_level_line述語で(L+1)個のレベル線を2単位下方に伸ばした後、Pをその右に書くことを指示している。

```
statement(P):-level(L),space(IS),con(down),
              write_level_line(L),nl,
              write_level_line(L),arg(P).(2)
```

2．仕様の整備作成とプログラム作成

2．1　キーワードによる手続き文の検索と仕様の作成

一般にプログラムは仕様に基づいて作成される。ここに仕様は処理に関連する主な対象の名前、データの型や構造、初期値など、対象のデータ構造に関する仕様と、対象の処理手続きに関する仕様からなる。対象のデータ構造に関する仕様は一般に定形的で指定も簡単であるが、処理手続きの仕様やプログラムは日本語文などを含み複雑多様であり、構文解析などを含む機械処理可能な仕様文法の考案は従来からの研究課題である。また、新しく書式や文法を修得して仕様を作成することは、ユーザにとって負担が増すことになる。

このような観点から対象のデータ構造に関する仕様は、表形式のリストにより機械可読の形で与え、処理仕様は概略仕様の手続きなどを箇条書き風に書いたものをユーザがメモ用に用意する。そしてユーザは概略仕様の箇条書き文が表す処理のキーワードを選び図2のような命令

```
dict([head(読み込む,read),
 body([p(x,y,[X,を読み込む]),
```

Automatic Generation of SPD and programs from Speifications
Masahiro OSOGAMI and Fujio NISHIDA
FUKUI TECHNICAL COLLEGE

```
dict([head(読み込む,read),
 body([p(x,y,[X,を読み込む]),
       p(1,X,[X,を読み込む]),
       p(2,Y,[プロンプトにより,Y,を読み込む]),
       p(3,Z,[ファイルから、Z,を読み込む]),
       ..........................
```

図２．　命令・手続き文見出し辞書の例

```
 module([jp([プロンプトにより、OBJ,を読み込む],
         prompt_read(OBJ)),
        argl([[name(OBJ),type(TYPE),array([dim(1),
        size(N),max(NMAX)])]])],
      [funct([int,[readladi(OBJ,N)]],
             [float,[readladf(OBJ,N)]])],
      file_name[[int,'rdladif.c'],
                [float,'rdladff.c']]),
 ....]).
```

図３　モジュール辞書の例

・手続きの見出し文辞書を検索する。見出し辞書にはhead引き数部にあるキーワードを含むいろいろな命令やモジュールの手続きの見出し文が日本語文などでbody部に記録してあり、キーワードを入力すればディスプレイに表示される。body部は番号、変数名、見出し文の項目からなり、仕様に適合する手続きや命令見出し文の文番号と仕様の対象変数名を入力すれば、プログラム作成システムMAPPは、対象のデータ構造に関する仕様、命令文辞書、モジュール辞書を参照してタイプのチェックを行い、整備した仕様を出力する。次にMAPPは整備した仕様の中の命令や手続きを、(3)のような規則を用いて、対応するＣやＣＯＢＯＬの関数、手続き、命令に置き換えてバッファに蓄積する。

```
 proc(X):-module(H,T),
     member(jp(X,Y),H),member(argl(ARGL),H),
     type_check_argl(ARGL),
     ........................
     member(funct(M),T),member([TYPE,PROC],M),
     writelist(PROC),write(';'),nl,
     member(file_name(FNL),T),
     member([TYPE,FN],FNL),
     included(FN),.... .                    (3)
```

(3)は処理仕様リストの要素Xがあるモジュール辞書の頭部のHに見出されときには、同じHにある引き数リストARGLの要素を対象のデータ構造・タイプ仕様との一致可能性などについて調べ、成功すればCによる関数表現の出力、Ｃプログラムモジュールのファイルのインクルード処理などを行うことを示す。

モジュール辞書項目や命令辞書項目は一まとまりの手続きや各命令毎に設ける。これらの辞書には、図３のように、引き数変数や関数の型、対象言語における関数表現、対象言語によるプログラムを収納するファイル名の他、このモジュールを適用するのに必要なデータや状態を記述する入力条件、処理後に生成されるデータ名や条件などを必要に応じて記録しておき、手続きや命令間の部分的な論理チェックや、命令などの欠落部の自動補填などに利用する。[2]

２．２　メイン関数頭部の作成

Ｃプログラムを作成する場合、手続きプログラムの前にインクルード関数やメイン関数などの関数記号部、型宣言部などを作成せねばならない。MAPPはこれらを対象のデータ構造に関する仕様や、検索したモジュールのファイル項目や引き数のタイプ項目などに含まれる情報を用いて、次のような述語main_headにより自動的に作成する。

```
 main_head:-(include([]);include(FILE),
             include_sent_gen(FILE)),
             main_header,type_decl.          (4)
```

右辺の本体部はインクルード述語の引き数部が空であれば何もせずに次の述語main_headerへ行き、そうでないときには引き数部のFILEをインクルードする文を include_sent_gen(FILE)という述語で作成し、その後で述語main_headerで main()などを作り、述語type_declでタイプ宣言部を作る。

上式のinclude(FILE)のFILEはモジュールの手続き文の機能を実現するＣプログラムの本体を収納するファイル名のリストである。手続き文を仕様に加えるとき、そのファイル名がinclude述語の引き数部のファイルリストに既に含まれているかどうかを調べ、まだ含まれていなければ、(5)によりこれを述語includeの引き数部に加える。

```
 included(FILE):-((include(OLD_FILE),
                   member(FILE,OLD_FILE));
                   add_file(FILE,OLD_FILE)).   (5)
```

mainなどの関数記号部は次の述語main_headerにより作成される。

```
 main_header:-main_arg(ARG),
              writelist(['main(',ARG,')']),nl,
              write('{'),nl.                   (6)
 main_arg([]).                                 (7)
```

main関数記号部につづく型宣言部は処理対象のデータ構造の仕様obj(X)や検索したモジュールの関数や引き数や一時置数変数などのタイプ項目から関連情報をtype_member述語の引き数部に抽出保持した情報をtype_decl述語により書き出して作成する。

参考文献

[1] 甲斐,遠藤:構造化プログラム設計図法
共立出版(1992)

[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,情報処理学会第４３会全国大会