6J－5

構造化図の作成・チェックとコードの生成

恐神正博　西田富士夫
福井工業大学

1．まえがき

構造化図の作成については各方面で開発が進められているが、ここでは見出し文から自動的に構造化図を描く方法について述べ、ネスト構造の分岐や繰り返しの制御文を中心に説明する。また、提案手法により、適用可能な命令やモジュールの見出し文を具象化してプログラム設計を行う場合、各手続き文の入力条件のチェックやコード生成の自動化などについて説明する。

2．手続き見出し文の検索と見出し文の作成

さきにメニュー方式でプログラム設計段階における手続き文を見出し文辞書を検索して作成する手法を発表した[1]。見出し文はCなどのプログラム言語の命令文や応用モジュールの呼出文を一部日本語風などの読み易い形式で書いたもので、必要な手続き文形を見出し辞書から検索し、関数のパラメータなどの形で含まれる変数名などを仕様に与えられたものでカスタマイズして手続き部の作成を行うものである。見出し文の言語の種類は自由に選ぶことが出来、一般に設計文書は読み易く作成し易く作れるほか、見出し文集合から目的プログラミング言語への変換システムを作成してプログラムへ自動変換したり、各見出し文に、処理が実行可能なための入力条件と実行後の出力条件を付与することにより、設計文書の実行可能性に関するチェックや、脱落している見出し文を自動的に補填することが出きる。

ここでは見出し辞書から制御文を検索し、制御手続き文ならびに構造化図を描く手法を述べる。図1は制御文の見出し辞書の1部である。

```
dict2([
[head(もし,if),  body(
  [·····························
  p(3,OBJ,seq_cond(OBJ),
    c([seq_cond(OBJ),
      b(SP),end_seq_cond])),
·····························  ])],
[head(繰り返す,repeat),  body(
[·····························
  p(5,[X,T], until([T,X]),
    c([until(X,を読み込み,T,
      なる条件が成立するまで,繰り返す]),
      b(SP),end_for])),
·····························])] ]).
```

図1　制御文見出し辞書

分岐の見出し文のキーワードは、"もし"や"if"、反復のそれは、"繰り返す"や"repeat"である。body部はheadのキーワードに属する手続き見出し文の関連項目からなり、pの引き数は制御文の場合4個で

見出し文番号、条件部に含まれる引き数、
条件部見出し、見出し文　　(1)

からなる。キーワードを入力して表示された見出し文の中で、仕様に適用可能な見出し文番号

Specifying and Checking of Structured Diagrams for Transferring to Programs
Masahiro Osogami and Fujio Nishida
Fukui Institute of Technology
3-6-1, Gakuen, Fukui City, 910, Japan

と、条件部の変数名リストや式などの２個の引き数を入力する。なお、仕様に適用可能な条件部の見出し文が前もって分かっており、検索が不必要なときには、(1)の３番目の引き数の条件部見出しを直接入力すればよい。(1)の４番目の引き数の見出し文は本体部からの再帰呼出利用が可能である。

制御文の本体部は制御文の種類によりcase文やseq_cond文のようにいくつかに分かれるものがあるが、各本体部の構造は殆ど全く自由であり、通常のいくつかの処理文であったり、ネスト状の制御構造であったり、いろいろである。seq_cond文では場合分けの条件リストを本体部入力のプロンプトにより入力すると、条件毎に(2)により処理が行われる。すなわち、条件毎に処理の見出し文入力のプロンプトが出て、

```
when(COND,c([cond(COND),b(SP),end_when])])
:-writelist([COND,の本体部を入力下さい]),
  nl,read(P),spc_list(P,SP),.....,
  draw_pic.                                (2)
```

```
○--start
|-- () ------ (UNTIL:x,に読み込み,x<0,が
|    |          成立するまで繰り返す)
|    |---<>---(SEQ_COND:x)
|    |    |
|    |    | (WHEN:x>=80)
|    |    |------:compute(cnt[0]=cnt[0]+1)
|    |    |
|    |    | (WHEN:x>=70)
|    |    |------:compute(cnt[1]=cnt[1]+1)
|    |    |
|    |    | (WHEN:else)
|    |    |------:compute(cnt[2]=cnt[2]+1)
○--end
```

図２　構造化図の例

キーボードから見出し文のキーリストを P に入力する。spc_list(P,SP)により希望の見出し命令文番号や引き数部の変数名入力のプロンプトが出てこれらを入力すると、処理の見出し文のリストがSPにえられ、draw_picによりこのCOND文の構造化図が描かれる。

図２は図１のような見出し辞書を用いて x にデータを読み込み、頻度分布を求めるために作成した見出し文の構造化図を示す。

３．設計文書のチェック

既成の命令やモジュールを組み合わせて、仕様に適合するようにプログラム設計を行う場合にチェックすべきことは、

(1)これらのモジュールが与えられた仕様に対して適用可能かどうか

(2)各見出し文の入力条件が最初に与えれる入力条件や、先行する見出し文の出力条件により保証されているかどうか

をチェックすることなどである。ただし算術式などは簡単のため、代入演算子右辺などに現れる入力変数の値が先行文でメモリ上に与えられているかをチェックするにとどめる。

このため各見出し文の辞書に、プログラムにおける命令表現の他、関数引き数部などのタイプや適用可能なための入出力条件を記録する。

MAPPは見出し文辞書を参照して、見出し文プログラムの各文の変数のタイプやデータ構造、入力条件をチェックして、さきに[1]述べた手法によりプログラムに変換する。

さらに、命令見出し文の欠落部の補填、プログラムの出力条件リストや機能リストからのプログラム合成を、現在のところ狭い範囲であるが、行うことが出きる。[2]

参考文献

[1]恐神,西田:仕様からの構造化図の簡易作成とプログラムの合成,情報処理学会第46回大会

[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回大会