ジャクソン法によるプログラム設計の自動化

4L-4

高松 忍, 徐 国偉, 乾 隆夫, 西田 富士夫
(大阪府立大学 工学部)

1. まえがき

本稿は、ジャクソン法による仕様をライブラリモジュールを用いて半自動的に詳細化する手法について述べている。入出力のデータ構造の記述と制限日本語で書いたプログラム仕様を与え、これをライブラリモジュールの援用により詳細化してプログラム構造を作り、これよりＣＯＢＯＬやＣのプログラムを生成する。

2. ライブラリモジュール

ライブラリのモジュールは、知識モジュール、データ構造変換モジュール、手続きモジュールとプログラム変換モジュールからなる。知識モジュールは各専門分野におけるデータ、データ間の関係や処理に関する表現を定義するものである。データ構造変換モジュールは入力のデータ構造を出力のデータ構造に一致した構造に変換するためのものであり、ソートなどのモジュールがある。手続きモジュールは、マクロの手続き表現と入出力表現の見出し部と、下位の処理手続きを記述する本体部からなる。プログラム変換モジュールは、手続きモジュールにより詳細化されたprimitiveな表現をプログラム言語に変換するものである。

3. 仕様表現

仕様は、入力データの構造と性質の記述、出力データ構造の記述と入力データから出力データをうる処理内容の記述からなる。入出力データ構造はジャクソン法の連接(seq)、繰り返し(iter)、選択(sel)の３つのタイプを用いてインデントした横書きのリスト構造で記述する。処理内容は制限された日本語の入出力関係文や手続き文で記述する。例えば、手続き文は、

”入力データdiを［データdrを参照して］procし、［属性aoをもつ］出力データdoを作れ”

の形で与える。例１に、発注ファイルと顧客ファイルから送り状リストを出力する仕様の例を示す。

例１

```
入力データ構造
(発注ファイル
  (iter  発注レコード
      (seq  顧客名
            項目
          (iter  項目レコード
              (seq  品名  額)))))
(顧客ファイル
  (iter  顧客レコード
      (seq  顧客名  住所)))
出力データ構造
(送り状リスト
  (iter  送り状
      (seq  顧客名  住所  総額)))
手続き文
顧客名ごとに、顧客ファイルから発注レコードに対する顧客レコードを検索し、発注レコードと顧客レコードから送り状を作成せよ。ただし、総額=SUM(SET(額[r]|r∈項目))とする。
```

4. 仕様の整備と詳細化

まず、日本語で与えた処理内容の仕様を格構造解析して内部表現を構成し、これを形式表現に変換する。続いて、出力のデータ構造を中心に仕様を一つにまとめて整備する。すなわち、出力データdoに対し、doに必要な入力データdi、入力データとの間の関係や処理手続きを、

do:(di:手続き文または入出力関係文の形式表現)

の形にまとめる。例２に例１の仕様を整備した結果を示す。ここで、例１の手続き文の形

Automating of Program Design by Jackson's Method
Shinobu TAKAMATSU, JYO Kokui, Takao INUI and Fujio NISHIDA
University of Osaka Prefecture

式表現は、その内部表現から"～から～を検索する"や"～から～を作成する"に関する変換ルールと仕様で与えられた入出力データの情報により得られる。

例２

```
(送り状リスト:( 発注ファイル,
                顧客ファイル )
  (iter 送り状:( 発注レコード,
                 顧客レコード:
      RETRIEVE_RECORD(SO: 顧客ファイル,
            OBJ: 顧客レコード
              (COND:
                顧客名[顧客レコード]=
                顧客名[発注レコード]),
            GO: Z );
      ASSIGN(OBJ:顧客名[Z],
            GO:顧客名[送り状]);
      ASSIGN(OBJ:住所[Z],
            GO:住所[送り状]);
      ASSIGN(OBJ:SUM(SET(額[r]|r∈項目
                    [発注レコード])),
            GO:総額[送り状])))
```

つぎに、仕様の手続き名、入出力関係名やデータ名などを用いて、ライブラリの手続きモジュールを検索し、実行可能なモジュールの存在や入出力条件をチェックして仕様の詳細化を行なう。入力条件を充足しない場合には、モジュールのリンクの手法により、不足するモジュールをライブラリから補う。例えば、仕様において省略されるファイルのopen、closeやread、writeの手続き表現を補う。また、ジャクソン法におけるようにデータ構造のタイプからプログラムの制御構造を求める。例２の仕様を以上のように詳細化して得られる制御手続き表現を例３に示す。

例３

```
OPEN(OBJ:発注ファイル,顧客ファイル,
        送り状リスト);
READ_RECORD(OBJ:発注ファイル,
            GO:発注レコード);
WHILE(COND:NOT_EOF(発注ファイル),
      OBJ:RETRIEVE_RECORD(
            SO:顧客ファイル,
            OBJ:顧客レコード
                (COND:
                顧客名[顧客レコード]
                =顧客名[発注レコード]),
                GO:Z);
      ASSIGN(OBJ:顧客名[Z],
                GO:顧客名[送り状]);
      ASSIGN(OBJ:住所[Z],
                GO:住所[送り状]);
      SUM_UP(OBJ:SET(額[r]|r∈項目
                [発注レコード]),
                GO: 総額[送り状]);
      WRITE_RECORD(OBJ:送り状,
                GO:送り状リスト);
      READ_RECORD(OBJ:発注ファイル
                ,GO:発注レコード));
CLOSE(OBJ:発注ファイル,顧客ファイル,
            送り状リスト)
```

以下、手続きモジュールを用いてprimitiveな表現になるまで同様な詳細化を繰り返し、最後にプログラム変換モジュールを用いてprimitiveな表現をプログラム言語に変換する。

５．入出力データ構造の不一致

上述の方法は、入出力間のデータ構造が一致（対応）する場合である。一致しない場合には、データ構造変換モジュールを用いて入力のデータ構造を出力のデータ構造に一致する中間のデータ構造に変換する必要がある。例えば、例１の入力データ”発注ファイル”が例４に示すように品名でソートされたデータ構造で与えられる場合、入出力データ構造の不一致がある。この場合には、例４のデータ構造を顧客名でソートした例１のデータ構造に変換する。

例４

```
(発注ファイル
  (iter 発注レコード
      (seq 品名 額 顧客名)))
```

ただし発注ファイルは品名でソートされている。

参考文献

1) Jackson.M.A.,鳥居訳: 構造的プログラム設計の原理、日本コンピュータ協会(1980)

２）西田・藤田・高松: ライブラリモジュールのリンクの手法による仕様の詳細化と誤りの検出、情報処理学会論文誌、vol.26,No.5,pp.489-498(1987)

３）西田・高松・谷: 要求仕様における日本語表現と形式表現間の相互変換、vol.29,No.4,pp.368-377(1988) 情報処理学会論文誌