2L-4 モジュールのリンクによるジャクソン法の仕様の詳細化

徐国偉，中島孝士，高松忍，西田富士夫
（大阪府立大学　工学部）

1. まえがき

本稿は，入出力のデータ構造に基づく詳細化法の一つであるジャクソン法をモジュールのリンクにより半自動化する手法について述べる．ジャクソン法では，入出力のデータ構造を連接（SEQUENCE），繰り返し（ITERATION），選択（SELECTION）の３つのタイプによって記述し，これらのタイプからプログラムの制御構造を作る．また，入出力データの対応関係から出力データをうる手続きとして仕様の詳細化を行う．この方法は，入出力間のデータ構造が一致（対応）する場合に有効であるが，一致しない場合には，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．

本稿では，入出力間のデータ構造が一致しない場合に，データ構造変換モジュールと制御構造生成モジュールのリンクの手法により，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し，この後手続きモジュール，プログラム変換モジュールを用いて詳細化する手法を与えている．

2. モジュール

モジュールは，データ構造変換モジュール，制御構造生成モジュール，手続きモジュール，プログラム変換モジュールからなる．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る．これらには，

(1)属性a_iでソートされたレコードの繰り返し構造を属性a_jでソートされたレコードの繰り返し構造に変換する．

(2)属性aの値が同じであるレコードの繰り返し構造を，同じ属性値の部分と残りの繰り返し構造の部分の連接構造に変換する．

などのモジュールがある．データ構造変換モジュールは，変換前のデータの構造や性質を記述する部分（IN），変換後のデータの構造や性質を記述する部分（OUT）とデータ構造の変換手続きを記述する部分（OP）からなる．

以下に上の(1)のモジュールを示す．

```
IN  :GIVEN(f1), ITERATION(f1, r),
     SEQUENCE(r, a),
     SORTED(OBJ:f1, KEY:ai, MODE:m1)
OUT:GIVEN(f2), ITERATION(f2, r),
     SEQUENCE(r, a),
     SORTED(OBJ:f2, KEY:aj, MODE:m2)
OP  :SORT(OBJ:f1, KEY:aj,
                MODE:m2, GO:f2)
```

ここで，ITERATION(f,r)はfがrの繰り返しであることを，SEQUENCE(r,a)はrが$a(=a_1, a_2, \cdots, a_n)$の連接であることを示す．

制御構造生成モジュールは，入出力のデータ構造が一致する場合に，データ構造のタイプ（連接，繰り返し，選択）からプログラムの制御構造を生成するモジュールである．このモジュールは，入出力のデータ構造が一致する条件を記述する部分（INとOUT）と，対

Refinement of Specifications of Jackson's Method by Linking Modules

Kokui Jyo, Takashi Nakajima, Shinobu Takamatsu and Fujio Nishida
University of Osaka Prefecture

応する制御構造を記述する部分(OP)からなる. 以下に繰り返し構造の場合のモジュールを示す.

```
IN  :GIVEN(f), ITERATION(f, r),
     SEQUENCE(r, a),
     SORTED(OBJ:f, KEY:a_i, MODE:m)
OUT:GIVEN(g), ITERATION(g, s),
     SEQUENCE(s, b),
     SORTED(OBJ:g, KEY:a_i, MODE:m),
     p(a, b)
OP  :WHILE(COND:not-eof(f),
        DO: IN  :GIVEN(r),
                 SEQUENCE(r, a)
            OUT:GIVEN(s),
                 SEQUENCE(s, b),
                 p(a, b) )
```

手続きモジュールは, 手続き表現(PROC), 入出力関係表現(IN, OUT), データタイプ(TYPE)の見出し部と, 見出し部に対する下位の処理手続きの本体部(OP)からなる. プログラム変換モジュールは, 手続きモジュールの本体部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. 仕様と詳細化

仕様は, 入力データの構造と性質を記述する部分(IN)と, 出力データの構造, 性質と入出力データ間の関係を記述する部分(OUT)からなる. 例1に, 入出庫累積ファイルから物品別の変動量の報告書を出力する仕様の例を示す.

例1

```
IN: GIVEN(入出庫累積ファイル)
    ITERATION(入出庫累積ファイル, 入出庫レコード)
    SEQUENCE(入出庫レコード, {日付, 品名,
                        入出庫コード, 数量})
    SORTED(OBJ:入出庫累積ファイル, KEY:日付, ···)
OUT:GIVEN(報告書)
    ITERATION(報告書, 物品レコード)
    SEQUENCE(物品レコード, {品名, 変動量})
    SORTED(OBJ:報告書, KEY:品名, ···)
      EQUAL(品名[物品レコード],
            品名[入出庫レコード])
      EQUAL(変動量[物品レコード],
          SUM(数量[r]|入出庫コード[r]=入庫, ···)
         -SUM(数量[r]|入出庫コード[r]=出庫, ···))
```

仕様の詳細化は, 入出力関係を満たすようにモジュールのIN-OUT部をリンクし, モジュールのOP部を用いて手続き仕様に変換することにより行う. 始めに, 仕様のOUTの否定クローズを作り, これと仕様のINの前提クローズ, データ構造変換モジュールと制御構造生成モジュールのIN-OUTのクローズの集合から反駁を行う. 反駁されると, 反駁に用いたデータ構造変換モジュールのOP部から, 仕様の入力データ構造を出力データ構造に一致する中間のデータ構造に変換する手続きが得られる. また, 反駁に用いた制御構造生成モジュールからプログラムの制御構造が得られる. 次いで, 手続きモジュール, プログラム変換モジュールを用いて下位の詳細化を行う. この後, 共通な繰り返し処理の部分を一つにまとめるなどの最適化を行う.

例2に, 例1の仕様から詳細化された制御・手続き表現を示す.

例2

```
        ⋮
SORT(OBJ:入出庫累積ファイル, KEY:品名, ···,
     GO:入出庫累積ファイル' )
        ⋮
WHILE(COND:not-eof(入出庫累積ファイル' ),
      DO:品名[物品レコード]
           :=品名[入出庫レコード'];
         SUM-UP(OBJ:{数量[r]|
                   入出庫コード[r]=入庫, ···},
                GO:y1);
         ·····
         変動量[物品レコード]:=y1-y2;
         ⋮
      )
```


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, 26, 5, p. 489(1987).