2R−7　形式的仕様の木構造表現

大場克彦　金戸孝夫
(株)島津製作所　技術情報システム部

1　はじめに

プログラムの信頼性を高めるためには、仕様にあいまいさが無いことが出発点になる。このために形式的仕様記述法の研究が続けられており、あいまいさの無い仕様を与えるための理論的基礎が整いつつあると考えられる。今後の重要な課題の一つとして、これらの基礎理論の成果を基にしながら、これらの基礎理論を意識せずに使える形式的仕様記述法の開発があると考える。筆者らは、このような観点からいくつかの考察を行い[1][2][3]、その結果に基づき、木構造表現による形式的仕様記述法を考案したので報告する。なお、ここでは逐次処理のプログラムで比較的小さなモジュールの、設計仕様の形式的表現について述べる。

2　仕様の概要

仕様は、名称、型定義、操作定義、変数定義、動作記述から構成される。

名称では、仕様の名称を記述する。

型定義では、変数定義に現れる変数のデータ型を定義する。型定義は個々の仕様の中には記述せず、別にデータ型辞書を設けその中で定義する。

操作定義では、既に定義されているデータ型に対する操作のインタフェースを定義する。操作定義も個々の仕様の中には記述せず、操作定義辞書を設けその中で定義する。このようにすることにより、個々の仕様の記述量を減らし、仕様の簡潔さを保つことができる。

変数定義では、個々の仕様の中に現れる変数の型を定義する。変数定義は、個々の仕様の中で行う。

動作記述では、プログラムがどのような動作をするかを記述する。本論では、動作記述を、木構造図を用いて形式的に表現する方法について述べる。

3　操作

操作を用いて動作記述を行う。操作は、仮操作と終端操作に分けられる。仮操作は、実現方式がまだ定義されておらず、実現方式をプログラムの動作記述で定義しなければならない操作である。終端操作は、実現方式が既に定義されている操作である。さらに、終端操作を基本操作と定義済み操作に分ける。基本操作は、プログラム言語の基本データ型に対応する操作である。定義済み操作は、基本データ型や既に定義されているデータ型を組み合わせて作った、データ型に対する操作である。基本操作は、制御操作、代入式、基本関数がある。制御操作は、反復や選択など動作の流れを制御する操作である。制御操作には、前判定反復操作、後判定反復操作、二項選択操作、多分岐操作がある。代入式は、右辺の算術演算や論理演算の結果を左辺の変数に代入する操作である。基本関数は、文字に対する操作や入出力操作、ファイル操作を関数形式で表したものである。

動作記述に使用する主要な操作を図1に示す。

```
1 ◎—プログラム（入力並び）→（出力並び）
2 |—□仮操作（入力並び）→（出力並び）
3 |   |—：基本関数（入力並び）→（出力並び）
4 |   |—□仮操作（入力並び）→（出力並び）
5 |   |   |—・変数＝式
6 |   |   |—@定義関数（入力並び）→（出力並び）
7 |   |   ―
8 |   |—・変数＝式
9 |   ―
10 |—@定義関数（入力並び）→（出力並び）
11 |—：基本関数（入力並び）→（出力並び）
12 |—・変数＝式
13 |—□仮操作（入力並び）→（出力並び）
14 |   |—◇—ＩＦ：条件式
15 |   |—▼—ＴＨＥＮ：
16 |   |—□仮操作（入力並び）→（出力並び）
17 |   |   |—・変数＝式
18 |   |   |—・変数＝式
19 |   |   ―
20 |   |—★—ＥＬＳＥ：
21 |   |—□仮操作（入力並び）→（出力並び）
22 |   |   |—・変数＝式
23 |   |   |—・変数＝式
24 |   |   ―
25 |   ―
26 |—□仮操作（入力並び）→（出力並び）
27 |   |—○—前判定：条件式
28 |   |—□仮操作（入力並び）→（出力並び）
29 |   |   |—・変数＝式
30 |   |   |—・変数＝式
31 |   |   ―
32 |   ―
33 |—□仮操作（入力並び）→（出力並び）
34 |   |—○—後判定：
35 |   |—□仮操作（入力並び）→（出力並び）
36 |   |   |—・変数＝式
37 |   |   |—・変数＝式
38 |   |   ―
39 |   ▲—後判定条件式
40 |   ―
41 ●
```

図1　動作記述

4　動作記述

プログラムを、順序を持った仮操作の列（順序仮操作列と呼ぶ）として定義する。次に、順序仮操作列の中で

Formal Specifications Expressed by Tree Structured Dyagrams
Katsuhiko OHBA, Takao KANETO
SHIMADZU CORPORATION

終端操作に置き換え可能な仮操作を、終端操作に置き換える。これを第１階層の順序操作列と呼ぶ。第１階層の順序操作列の中の仮操作を、さらに順序仮操作列で表し、この中の終端操作に置き換え可能な仮操作を、終端操作に置き換える。順序操作列の中に仮操作を含まなくなるまで、各階層の全ての仮操作に対して、この操作を繰り返す。いま、等価な関係にある操作と順序操作列を

操作= 順序操作列　　①

で表すと、左辺の操作を右辺の順序操作列で置き換える、変換規則の集合として、仕様を表すことが出来る。さらに、①式で表される等式集合を、木構造図で表現できることを示した[2][3]。動作定義部は、①式の等価関係で表現された等式集合から作成した、木構造図で表す。木構造図による仕様の表現形式を図１に示す。

図１において、１行目はプログラムを表している。２行目から４０行目までがプログラムの動作を表している。４１行目は、動作記述の終了を表している。第１階層の順序操作列は、２、１０、１１、１２、１３、２６、３３の各行である。

２行目の仮操作の実現を表す順序操作列が、３、４、８、９の各行である。９行目は、２行目の仮操作の実現の終わりを表す。４行目が仮操作であるが、この実現を表す順序操作列は、５、６、７の各行である。

１３行目の仮操作は、二項選択を持つ仮操作である。この実現を表す順序操作列は、１４、１５、１６、２０、２１、２５の各行である。１４行から２５行は、二項選択を持つ仮操作を実現する順序操作列の基本的なパターンを示している。

２６行目は、前判定反復を持つ仮操作である。２７行から３２行は、前判定反復を持つ仮操作を実現する順序操作列の基本的なパターンを示している。同様に、３３行目は、後判定反復を持つ仮操作である。３４行から４０行は、後判定反復を持つ仮操作を実現する順序操作列の基本的なパターンを示している。

仮操作は、順序操作列を実行することにより実現される動作を抽象化した操作で、プログラムのブロックの概念に相当する。ブロックと異なる点は、関数形式という入力と出力を明示した形式化された表現を持つことである。仮操作に形式化された表現を与えることにより、仕様を意味的な面から階層的に表現することを可能にした。

５　仕様の完全性と無矛盾性

仕様が完全であるとは、全ての正しい入力に対して、出力が仕様から定まることである。仕様が無矛盾であるとは、各々の入力に対して、出力が仕様から一意に定まることである。

５．１　仕様が完全であるための条件

仕様が完全であるためには、以下の条件が満足されていることが必要である。

（１）構文が正しいこと。

仕様の構造が正しいことおよび各操作の構文が正しいことが必要である。

（２）プログラムの動作記述が正規形に変換できること。

ここで、正規形とは順序終端操作列（終端操作だけで表現した順序操作列）をいう。

（３）仕様の中に終端操作として書かれた操作が、確かに終端操作であること。このためには、以下の条件が満足されていることを確かめる。

①定義済み操作について、仕様の中に出てきた定義済み操作の名前が操作定義で定義されていること、入出力の引数部に現れる変数が変数定義で定義されていること、入出力の引数部のデータの型が操作定義の中の操作の入出力引数の型と一致していること。

②基本関数についても、①と同じ条件が満足されている。

③代入操作について、式が構文木で正しく表現できること、式の中に現れる変数が変数定義部で定義されていること、演算対象となるデータの型が演算子のデータ型と一致していること。

④制御操作についても、③と同じ条件が満足されている。

（４）定義済み操作の動作を定義する時、自分より下のレベルの定義済み操作の動作定義部に自分自身が使われていないこと。

（５）型定義部の型定義を右辺から左辺への変換規則と見なしたとき、型定義部で定義されている全ての型が、基本型を合成した形で表現できること。

（６）次のデータの受渡し規則を満足すること。

①ある子操作の入力は、定数か、親操作の入力か、兄操作の出力かのいずれかである。

②親操作の出力は、自分の子操作の出力の中にある。

ここで、親操作とは①式の左辺の操作、子操作とは、右辺の順序操作列の中の操作、兄操作とは同じ順序操作列の中で自分より前にある操作のことである。

５．２　仕様が無矛盾であるための条件

仕様が無矛盾であるためには、以下の条件が満足されることが必要である。

（１）同じ名前で、異なる操作が定義されていないこと。

（２）変数定義部に同じ名前の変数が存在しないこと。

（３）同じ名前で、異なる型が定義されていないこと。

７　おわりに

上に述べた構文規則を使い、文献（１）の設計手順に従い動作記述を行うと、図１に示したような木構造表現による仕様が得られる。この仕様記述法で記述された仕様の特徴は、以下の通りである。

①構文と意味が定められた操作を用いて仕様を記述するので、仕様に一意な意味を与えることが出来る。

②全ての操作の入出力関係を明確にすることにより、仕様の理解性を高めている。

③仮操作という概念を導入することにより、仕様の意味的な面での階層表現が可能である。

④自然な日本語表現が可能である。

⑤仕様の完全性、無矛盾性を検査する規則を用いて、機械的に仕様の完全性、無矛盾性を確かめることができる。

以上の結果、記述能力が高くて理解しやすく、かつ機械的な検証およびソースプログラムの生成が容易な仕様の記述が、可能になった。

参考文献

（１）大場、金戸：代数的仕様の実用プログラムへの適用に関する考察、情報処理学会第４０回全国大会講演論文集（１９９０）

（２）大場、金戸：抽象データ型に基づくプログラム設計［１］、情報処理学会第４１回全国大会講演論文集（分冊５）（１９９０）

（３）大場、他：抽象データ型に基づくプログラム設計［２］、情報処理学会第４１回全国大会講演論文集（分冊５）（１９９０）