抽象データ型に基づくプログラム設計［２］

３Ｇ－９　大場克彦、井上信介、二木敬一、金戸孝夫
（株）島津製作所

１　はじめに

抽象データ型を用いて、代数的仕様記述法で得られるのと等価な木構造図が得られることを示した[1]。この木構造図は、トップダウン設計法で作成するのと同じ手順で作成できるので、従来のプログラミング手法に慣れたプログラマでも、難解な理論を意識しなくても作成することが出来る。しかも、厳密に意味が定義された形式性を有している。このため、この木構造図を使って検証やソースコードの生成が可能である。

一般にプログラムが順次、選択、反復の３つの制御構造の組合せで表現できることはよく知られている。抽象度が高いレベルでは順次処理だけでプログラムの構造を表現できるが、具体化した詳細な表現をするためには、選択や反復の制御構造が必要である。文献［１］で示した方法は、抽象度の高いレベルでの可能性を示したもので、より具体的な記述を可能にするためには、選択や反復などの制御構造を含めた記述を可能にする必要がある。本論ではこの点について論ずる。

２　選択構造

いま、仮操作Ｋ０［Ｃ］を考える。Ｋ０［Ｃ］は条件Ｃが満足されたとき仮操作Ｔを、満足されないとき仮操作Ｆを実行するものとする。ＩＦ［Ｃ］という基本操作を導入し、Ｋ０［Ｃ］を①式の合同関係で表す。

Ｋ０［Ｃ］≡ＩＦ［Ｃ］ＴＦ　①

Ｃは真か、偽のいづれかの値をとる。ＩＦ［Ｃ］は、Ｃが真ならＴを、偽ならＦを実行する基本操作である。これを表す合同関係が②③である。

ＩＦ［真］ＴＦ≡Ｔ　②

ＩＦ［偽］ＴＦ≡Ｆ　③

①②③の合同集合に推移則を適用すると、④⑤の合同関係が得られ、これがＫ０［Ｃ］の意味を表している。

Ｋ０［真］≡Ｔ　④

Ｋ０［偽］≡Ｆ　⑤

合同関係①－⑤の表す意味をプログラミング言語風に定義すると

```
K0 [C] ≡ if C then T
           else F    ⑥
```

となる。また⑥と等価な木構造図を、図１のように定義する。

３　反復構造

条件Ｃが満足されている間、操作Ｋを反復実行する仮操作をＬ［Ｃ］とする。Ｌ［Ｃ］を⑦⑧の合同関係で定義する。

Ｌ［Ｃ］≡ＩＬ'［Ｃ］　⑦

Ｌ'［Ｃ］≡Ｗ［Ｃ］Ｋ　⑧

ここでＣは、真か偽のいずれかの値をとる。ＩはＣの初期条件を設定する仮操作であり、Ｌ'［Ｃ］はＬ［Ｃ］の中の反復部分を表す仮操作である。Ｗ［Ｃ］は、Ｃが真のときＫを実行し、偽のときＫをスキップする基本操作である。Ｌ'［Ｃ］の後に実行する操作をＮとすると

Ｌ'［Ｃ］Ｎ≡Ｗ［Ｃ］ＫＮ　⑨

の合同関係が成り立つ。さらに次の合同関係を定める。

Ｗ［真］ＫＮ≡Ｋ　⑩

Ｗ［偽］ＫＮ≡Ｎ　⑪

⑨⑩⑪の合同関係に推移則を適用し、⑫⑬の合同関係が得られる。

Ｌ'［真］Ｎ≡Ｋ　⑫

Ｌ'［偽］Ｎ≡Ｎ　⑬

仮操作Ｋは、非再帰的操作Ｋ１を実行した後に、再帰的な操作Ｒを実行する操作で、⑭の合同関係で定義する。

Ｋ≡Ｋ１Ｒ　⑭

再帰操作Ｒは、⑮の合同関係で定義する。すなわち、再帰操作Ｒの実行は、Ｌ'［Ｃ］の実行と等価であると定義する。

Ｒ≡Ｌ'［Ｃ］　⑮

仮操作Ｌ'［Ｃ］の意味をプログラミング言語風に定義すると

Ｌ'［Ｃ］≡ｗｈｉｌｅ　Ｃ　ｄｏ　Ｋ　⑯

となる。⑯の意味と等価な木構造図を図２で定義する。

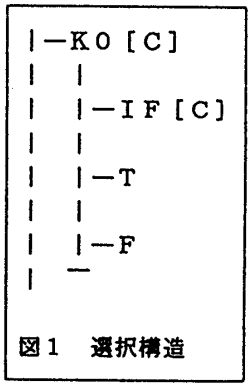

図１　選択構造

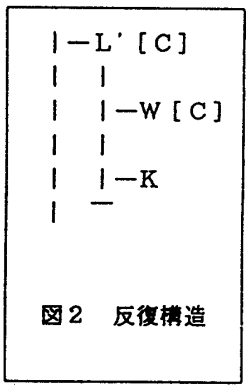

図２　反復構造

４　仕様の記述

文献［１］および上に述べた手法を用いて、ガウスの消去法により連立一次方程式を解くプログラムを例

Program Design based on Abstract Deta Type [2]
Katsuhiko OHBA, Shinsuke INOUE, Keiiti FUTAGI, Takao KANETO
SHIMADZU CORPORATION

にとり設計手順を示す。

プログラム名を「ガウスの消去法により連立一次方程式を解く」とする。入力は、係数行列を入れた２次元配列（係数）、定数項を入れた１次元配列（定数）、連立方程式の元数（元数）で、出力は解を入れた１次元配列（解）である。これを、「上三角行列を求める」と「解を求める」という２つの仮操作で表現する。ここまでの手順で得られた木構造図を図３に示す。

＜◎＞はプログラムの始まり、　＜●＞はプログラムの終わりを示す記号である。＜□＞は仮操作を表す記号であり、＜｜＞は、その行が上の行の記述の継続であることを表すものとする。

図３は、「ガウスの消去法により連立方程式を解く」プログラム（以後プログラムと呼ぶ）には、係数、定数、元というデータが入力され、解というデータが出力されることを示している。また、このプログラムは、「上三角行列を求める」「解を求める」という２つの仮操作で実現されることを示している。「上三角行列を求める」という仮操作は、入力として係数と定数と元数をプログラムから受け取り、上三角行列と新定数を出力する。上三角行列は、係数と同じ行数および列数を持つ配列である。新定数は上三角行列に対応するように係数を補正したものである。「解を求める」という仮操作は、「上三角行列を求める」という仮操作から上三角行列と新定数を、プログラムから元数を入力として受け取り、解を出力する。この解は、プログラムの出力になる。

```
◎　ガウスの消去法により連立方程式を解く
｜　（係数、定数、元）→（解）
｜一□上三角行列を求める（係数、定数、元）
｜　　→（上三角行列、新定数）
｜一□解を求める
｜　　（上三角行列、新定数、元）→（解）
●
```

図３　ガウスの消去法（第１段階）

さて、図３で「上三角行列を求める」仮操作は、既に作成したものがあり定義済み操作とすることが出来るが、「解を求める」という仮操作は、まだ作成されたものがないので、さらに詳細化が必要であるとする。このような仮定のもとに、設計を最後まで進めた結果を図４に示す。

図４において、＜◎＞は定義済み操作、＜・＞は演算式という基本操作、＜○＞は反復条件を判定する基本操作を表す記号とする。

「解を求める」仮操作は、次の２つの操作を順番に実行することを表している。

①解［元］＝新定数［元］／上三角行列［元、元］
②解を１つ求める操作を繰り返す

次に、「解を１つ求める操作を繰り返す」は仮操作であるので、以下の操作に分解される。

①行＝元数－１
②［行＞＝１］の間繰り返す
③仮解＝新定数［行］
④左辺の定数項を右辺に移行する
⑤解［行］＝仮解／上三角行列［行、行］
⑥行＝行－１

さらに、「左辺の定数項を右辺に移行する」は、次の各操作から構成される。

①列＝行＋１
②［列＜＝元］の間繰り返す
③仮解＝仮解－上三角行列［行、列］＊解［列］
④列＝列＋１

```
◎　ガウスの消去法により連立方程式を解く
｜　（係数、定数、元）→（解）
｜一◎上三角行列を求める
｜　　（係数、定数、元）→（上三角行列、新定数）
｜一□解を求める
｜　｜　（上三角行列、新定数、元）→（解）
｜　｜一・解（元）＝新定数（元）／上三角行列（元、
｜　｜　　元）
｜　｜一□解を１つ求める操作を繰り返す
｜　｜　｜　（上三角行列、新定数、元、解）→（解）
｜　｜　｜一・行＝元－１
｜　｜　｜一○［行＞＝１］の間繰り返す
｜　｜　｜一・仮解＝新定数［行］
｜　｜　｜一□左辺の定数項を右辺に移行する
｜　｜　｜　｜（上三角行列、行、仮解、解、元）
｜　｜　｜　｜　→（仮解）
｜　｜　｜　｜一・列＝行＋１
｜　｜　｜　｜一○［列＜＝元］の間繰り返す
｜　｜　｜　｜一・仮解＝仮解－
｜　｜　｜　｜　　上三角行列［行、列］＊解［列］
｜　｜　｜　｜一・列＝列＋１
｜　｜　｜　一
｜　｜　｜一・解［行］＝仮解／上三角行列［行、行］
｜　｜　｜一・行＝行－１
｜　一　一
●
```

図４　ガウスの消去法（設計完了）

５　おわりに

プログラムの制御構造を代数的仕様で表し、次に、それと等価な、厳密な形式性を持つ木構造図を得られることを示した。そして、この木構造図がプログラムの詳細構造を表すことができることを、例題を用いて示した。この木構造図は、代数的仕様に関する基礎理論を意識しなくても作成することが出来るのが特長である。今後の課題は、以上の考察結果を基に、厳密な形式性を持つ木構造表現可能な仕様記述言語およびその言語を用いて記述された仕様の論理的矛盾を検出する検証系を含む、コンピュータ支援系の開発である。

参考文献

［１］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（１９９０）投稿中