実行時仕様の導入による形式的仕様の実行

1G−5

松並邦拓、小野康一、新井浩志、深沢良彰、門倉敏夫

早稲田大学理工学部

1．　はじめに

ソフトウェアを正確に定義したいという要求から、各種の形式的仕様言語が提案されている。これらの形式的仕様言語は、その計算可能性に従って、直接実行可能なものと不可能なものに分類できる。

直接実行可能な言語で記述された仕様は、ソフトウェア開発の初期段階で、実行による仕様の検査を行うことより、早期にユーザの要求に対する正当性を確認できる。しかしながら、その記述は計算の手順に依存することが多い。

実行不可能な言語で記述された仕様には、集合や代数のような数学の概念に基づいているものがある。この仕様は、ソフトウェア・システムを、より抽象度の高い記述で定義できる。しかし、この仕様は、実行による仕様の検査の手だてを持ち合わせていない。

そこで、本研究では、実行不可能な仕様記述の機械的な解釈を補助するために、実行時仕様を導入した。これによって、本来、直接実行が困難な仕様の実行を試みた。

2．　概要

本システムは、仕様記述言語として、集合の概念に基づくZ記法[1]をもちいる。また、実行時仕様言語は、代数的な形式に従い新たに定義した。実行時仕様では、主に仕様中の対象領域と、述語に関する計算可能な解釈を与える。

実行系は、仕様とそれに対応した実行時仕様を入力とする。そして、これらに記述されている対象領域に基づいて、述語の関係を充足する対象を選択する。この時、仕様を、ソフトウェア・システムの充すべきデータの関係が記述されているものと解釈している。この対象の選択の操作は、その関係を充たすデータを求めるプロセスと見なすことができる。本システムでは、このプロセスを仕様の実行と見なす。

また、実行時仕様は、インプリメントのための、より詳細化された指針となる。そして、実行系より得られたデータは、仕様からインプリメントされたプログラムをテストするための情報としても活用できる。

3．　本システムの構成

3．1　要求仕様

仕様言語Zは、数学の集合理論に基礎を置きその最小単位をスキーマとよぶ。ある述語をP（x）とすると、スキーマは以下の形式をもつ。

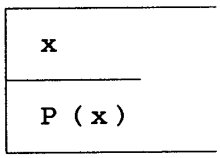

図1　Z記法

述語P（x）は、Dを空でない集合とする時、写像

$P : D \rightarrow T$

$T = \{1,0\}$,

$D = D_1 \times D_2 \times .. \times D_N$

$= \{(x_1,..,x_N) \mid x_i \in D_i \ (i=1,..,N)\}$

である。これ以後、DをPの対象領域とよぶ。また、その元を対象という。さらに、論理記号を用いることで述語の任意の組合せも表現できる。

3．2　実行時仕様

実行系に仕様の機械的解釈を行わせるために、代数的な形式による実行時仕様で、以下のことを補足しなくてならない。

1)　未定義の述語の真偽を決定する関数
2)　対象領域の解釈

1)を与えるための方法論は、未定義の述語に対して表現関数を与えるという考えに基づく。表現関数は、仕様記述で与えられた述語P（x）に対して、同じ定義域上で定義される関数f（x）であり、以下の関係が成立する。

P（x）　<=>　f（x）＝1

not P（x）<=>　f（x）＝0

この表現関数を、関数記号から構成される項の集合上の合同関係として、実行時仕様に記述する。

本手法では、未定義の述語を、帰納的述語と帰納

Execution Framework of a Formal Specification with a Runtime Specification

Kunihiro MATSUNAMI, Kouichi ONO, Hiroshi ARAI, Yoshiaki FUKAZAWA, Toshio KADOKURA

School of Science and Engineering, Waseda University.

的可算述語に分ける。それぞれの表現関数の記述を以下のように定める。

帰納的述語については、特定の表現関数ｆの評価値である１または０の値により、その真偽を決定できる。従って、その表現関数ｆの評価値を計算するアルゴリズムを実行時仕様で記述する。

帰納的可算述語については、特定の対象についての真偽の関係を実行時仕様で記述する。

2)に関して、実行系は、個々の対象をデータ型に対するインスタンスと考える。すなわち、実行時仕様には、対象領域の属性について、組込みのデータ型あるいは生成パターンのどちらかを記述する。

３．３　実行系の動作

実行系の基本的な動作は以下のようになる。

① 仕様記述中の述語の変数のリストを作る。

② 実行時仕様に与えられた対象領域の解釈に従って具象化した対象を生成する。そして、それぞれの変数に対象を割り当てる。

③ その変数値における述語の真偽値を、実行時仕様に記述されている表現関数を用いて決定する。このとき、実行時仕様で述語に関する情報が与えられていない場合は、対話的に述語の真偽を問う。

④ 対象のリストが仕様を充足するならそれを表示する。さらに対象を生成可能なときは実行を続けるかどうかを問う。続ける場合は②へ。

４．　仕様言語Ｚ及び実行時仕様の記述例

仕様記述の例として、ファイルシステムから特定された所有者のファイルをすべて削除するコマンドを考える。仕様記述言語Ｚでは、このコマンドを以下のように表現する。

```
remove_files ──────────────────────
  file_store : users -> file
  system_users : P users
  us? : users
  ──────────────────────────
  us? member system_users
            and
  file_store' = us? subtraction file_store
  ──────────────────────────
```

図２　要求仕様記述例

上記のスキーマ remove_files では、述語として、member, and, subtraction が用いられていることが分かる。組込み述語のデータベースであるSet, Bool に述語 member, and はすでに定義されているので、述語 subtraction を実行時仕様で新たに定義する必要がある。さらに、対象領域 users, files の取扱いに関する情報も必要である。ここでは、users に対しては user1, root の２つの対象を与える。files に対しては、文字型(String)のデータ型を与えることにする。以上より実行時仕様を定義する。

```
executespec remove_files
 using Set,Bool
 {user1,root} :-> users
 String :-> files
 function  file_store ( _ ) : set -> set
 function  _ subtraction _  :
                set , relation -> Bool
 vars arity1 : set
      arity2 : relation
 file_store( root ) == passwd
 file_store( user1 ) ==  test
 arity1 subtraction arity2  ==
   if arity1 member dom arity2 then
   (dom arity2 - arity1) restriction arity1
```

図３　実行時仕様例

実行系は、この実行時仕様より生成した対象を、それぞれの変数に割り当てる。

1) file_store は、図３の関数の定義より、{(root,passwd),(user1,test)} となる。

2) system_users は、図２より、users の部分集合であるので、{root,user1}, {root}, {user1},{} のいずれかになる。

3) us? の値は、実行時仕様で特定されていない。これより、実行系は、us? を入力変数と見なし、実行の途中でその値を外部に求める。ここでは、以下の入力が与えられたと仮定する。

```
us? : user1
```

実行系は、これらに従って、system_users の割当てが、{user1}, {root,user1} の場合のいずれかに限り仕様は充足し、file_store'の値は、{(root,passwd)}となる。

５．おわりに

現在のところ、本手法を用いた実行系のプロトタイプを作成中である。今後は、さらに組込み述語データベース及び表現関数の評価戦略の拡充を進める予定である。

参考文献

[1] Spivey,J.M., "*The Z Notation*" (Prentice-Hall International Science), Nov. 1987