形式化软件规范技术

刘少英*)（西安交通大学）

摘 要

形式化软件规范技术是增强软件开发过程的科学性和所开发软件的可靠性的一个有效途径。本文通过"教师职称提升系统"和"整数分类系统"实例较系统地介绍了目前国际上流行的两种形式化规范技术VDM和OBJ，并对它们各自的特点和性质进行了比较和分析。

一 引 言

软件开发可分为软件规范说明和软件实现两大部分。软件规范说明的任务是要告诉计算机"做什么"，而软件实现要解决的问题是"怎样做"。因此，软件规范说明在软件开发中是一个极为重要的阶段。如果该阶段所形成的软件规范不能正确地或不能确切地指出"做什么"，那么对后边的软件实现工作将会带来不可估量的灾难。为了解决好这个问题，人们已经提出了许多描述软件规范的方法和语言，如目前最为普遍应用的数据流图[1]，JSD[2]和自然语言描述[3]等，但这些

*) 系西安交通大学的国家公派留学生，在英国曼彻斯特大学计算机系攻读Ph. D.，现已两年多。

在其初态下允许的各变量的初值，驱动该EFSM运转，得到一个确定结构的有穷自动机。这一过程称为"展开EFSM"。展开后的EFSM就成为普通的有穷自动机。

3. 已经完成了从有穷自动机模型中产生测试序列的研究。可以从展开后的EFSM产生测试目标协议的一致性测试序列。主要方法是从自动机的初态出发遍历自动机，所遍历的各弧上的标记（协议原语名）就形成了测试序列。下一步工作就是根据对协议的一致性测试要求，构造实用的测试工具。

4. 在第(2)项工作的基础上，下面准备实现基于有穷自动机方法的协议验证工具。

从总体上看，我们目前进行的研究仍属于基础性研究，要达到实用阶段，还需作出更多的努力。

参 考 文 献

[1] W. C. Lynch, Reliable Full-Duplex Transmision over Helf Duplex Telephone Line, Communications of the ACM, vol. 11, No. 6, June 1968.

[2] ISO/TC97/SC21, ESTELLE——A Formal Description Technique Based on an Extended State Transition Model, ISO9074, 1988. 9.

[3] ISO/TC97/SC21, LOTOS——A Formal Description Technique Based on the Temporal Ordering Behavior, ISO 8807, 1988. 9.

[4] T. F. Piatkowski, Protocol Engineering, Proc. of ICC' 83, Boston, Mass, June 1983.

[5] H. Rudin, An Informal Overview of Formal Protocol Specification, IEEE Communication Magazine, vol. 23, No. 3, March 1985.

[6] 肖军模，一种有效的协议分析工具，（南京通信工程学院）军事通信学报，1987

方法全部是非形式化的，在描述上既不简练，在意义上也可能不确切，很可能造成理解上的不一致而导致所实现的系统不符合要求或隐藏着潜在的错误，尤其是一些像卫星发射、核电站控制系统等对可靠性要求很高的软件更是如此。为了解决这个问题，人们在七十年末和八十年代初先后提出了形式化规范技术VDM（维也纳开发方法）[4,5]，Z[6]和OBJ[7]等。VDM是一种功能构造性规范技术，它通过一阶谓词逻辑和已建立的抽象数据类型来描述每个运算或函数的功能。这种方法目前在欧美许多研究机构或大学得到广泛应用，有些大学如英国曼彻斯特大学等已把VDM作为大学计算机系学生的必修课。OBJ是一种代数规范技术，它通过代数等式来描述一个抽象数据类型中的运算。由于Z只是在表达方式上同VDM不同，所以本文只介绍VDM和OBJ这两种风格不同的形式化规范技术，并对它们进行比较分析，指出各自的特点和应用。

二 VDM 形式化规范技术

VDM形式化规范技术的基本思想是运用抽象数据类型、数学概念和符号来规定运算或函数的功能，而且这种规定的过程是结构化的，其目的是要在系统实现之前简短而确切地指出软件系统应完成的功能。在这种形式化规范中，由于采用了数学符号和抽象数据类型从而可使软件系统的功能描述在抽象级上进行，完全摆脱了实现细节，这样为软件实现者提供了很大的灵活性。此外，这种形式化规范还为程序正确性证明提供了依据。应用VDM技术进行系统开发就是在形式化规范说明，程序实现和程序正确性证明的三步曲中不断进行的。

1 VDM形式化规范的组成

一个VDM形式化规范由三部分组成，即状态规定，类型不变式规定和运算功能规定。状态规定包括利用VDM所提供的基本类型（如自然数N，实数R和布尔型B等），集合类型，表类型，映射类型以及复合类型来定义自己的有关类型，其基本形式为：

类型标识符＝类型定义实体

其中类型标识符是新定义的一个类型名，类型定义实体是用上边提到的五种抽象数据类型定义的一个新类型。下边只简单地叙述一下这五种抽象数据类型的定义形式和其上的运算。

（1） 基本类型。包括自然数类型N，实数类型R和布尔类型B等。其上的运算是通常数学中的算术运算和逻辑运算。

（2） 集合类型。定义一个集合类型的形式为：

$$X = set\ of\ Q$$

其中X是新定义的集合类型标识符，Q是另一个已定义的类型标识符或VDM中已提供的五种抽象数据类型的定义形式之一，它指出了该集合中元素的类型。在本文后边遇到类似情况可作相同解释。对于集合类型的运算有并（∪），交（∩），差（＼），求元素个数（Card）等。集合之间的关系判断有集合元素（∈），子集（⊆）和真子集（⊂）等。

（3） 表类型。一个表实际上是一个有序集合，其中每个元素都有确定的位置，而且还可重复出现。一个表类型定义形式为：

$$S = seq\ of\ Q$$

其中S是新定义的表类型标识符，Q是其元素类型。关于表的运算有求头(hd)，求尾(tl)，求长度（len），连接（⌢），求元素集合（elem）和求索引集合（inds）等。

（4） 映射类型。一个映射可以看成定义域是有穷可数集合的一个函数，通常以偶对集合形式来表示。映射类型定义形式为：

$$M = Map\ of\ Q$$

其中M是新定义的映射类型标识符。映射类型上的运算有：求定义域（dom），求值域（rng），限制定义域(△)，定义域元素删除（⩤）等。

(5) 复合类型。复合类型是指具有多个分量的类型。这个概念在某种程度上像PASCAL中的记录类型。一个复合类型定义形式为：

```
P=Compose P of
        e1:Q1
        e2:Q2
         ⋮
        en:Qn
    end
```

其中P是新定义的一个复合类型标识符，e_i ($i=1, 2\cdots, n$) 是其分量名，Q_i是其分量类型名。复合类型上的运算有：求复合类型值 (mk-P)，求分量值和改变分量值等运算。由于篇幅所限，关于这些类型的运算的详细说明可见参考文献[4]。定义这些类型标识符的目的是为了在运算中说明变量时更加简短方便。这个作用同Pascal中的类型说明部分类似。

例1 下边给出一些后边例4中定义一个教师职称提升系统的有关抽象数据类型：

```
Studentdata=Compose Studentdata of
            Name:string;
            T-Score-list:Seq of Choice
            end
Choice=seq of R
Researchdata=Compose Researchdata of
             Name1:string
             R-score-list:seq of N
             end
Teacher=Compose Teacher of
        T-name:string
        T-score:R
        R-score:R
        end
string seq of char
```

其中Studentdata描述学生对一个教师的教学效果评分的情况，它包括教师名字Name和学生的评分表T-score-list两部分。这个评分表是由一个班级中每个学生对该教师的教学评分结果（用Choice类型来表示）所组成的序列。Researchdata描述一个教师所发表的科研成果而得到相应评分的情况，它包括教师名字Name1和科研评分表R-score-list两部分。有关一个教师的工作情况信息可由Teacher类型来描述，它包括教师名字T-name，教学得分T-score和科研得分R-score三部分组成。

类型不变式规定是指对状态规定中所规定的类型给出其应满足的性质，提出类型不变式将有助于实现阶段选择适当的数据结构和算法。设ST是在类型规范中所定义的一个类型标识符，则它的类型不变式表示为：

$$inv-ST(s)=P(s)$$

其中P(s)是一个一阶谓词公式，规定出s所应满足的条件，这里s代表ST中任一个元素。

例2 在例1中为了用抽象数据类型Studentdata确切地描述学生对一个教师的评分情况，而且这个评分方式按形式：

教学效果	教学方法	教学态度	负责程度

以10分制评分，可用下边的不变式规定出这些数据类型必须满足的性质：

```
inv-Studentdata△∀mk-Studentdata
  (n, t)∈ Studentdata·n≠''⇒t≠〔〕
inv-Choice△∀t∈Choice·lent=4
```

不变式inv-Studentdata说明对于每个教师的教学评分情况，若其教师名字不为空，则其教学评分表一定不能为空。inv-Choice说明对于每个Choice中的元素，其长度只能是4。

运算功能规定是一个VDM形式化规范中的主要部分，其定义形式如下：

```
OP(m1:Tm1, m2:Tm2, ⋯, mn:Tmn)r1:Tr1, ⋯,
rc:Trc
    ext  p V1:TV1
         p V2:TV2
          ⋮
         p VK:TVK
    per  C1
    post C2
```

其中OP指出运算的名字。m_i ($i=1, 2, \cdots, n$) 是该运算的输入参量，r_j ($j=1, 2, \cdots, l$) 是输出变量。T_{m_i}，T_{r_j}是类型定义。ext是外部变量标记，其后所说明的变量V_q($q=1, 2, \cdots, K$) 全部是外部变量。p∈{wr, d}指出相应的外部变量可以被应用的方

式。wr说明相应变量可读可写，rd说明相应变量只可读，不可写。在pre之后的C_1和post之后的C_2均是一阶谓词公式，分别称为前置条件和后置条件。前置条件表明该运算在执行之前其输入参量和所用到的外变量应该满足的条件，而后置条件表明该运算在执行之后输入参量，输出参量和外部变量所应满足的条件。实际上，一个运算的功能主要由其后置条件表示出来。设S是一个外部变量，则在后置条件中用$\overleftarrow{S}$表示该运算执行前的S的值，而用S本身表示该运算执行后S的值。

例3 一个分类运算的形式化规范如下：

Index_entries=seq of N

SORT(S:Index_entries)r:Index_entries

ext wr output:seq of Index_entries

pre S≠〔〕

post elemr=elem S ∧is_sorted (r) ∧ output $=\overleftarrow{\text{output}}\frown\langle r\rangle$

该运算说明对于一个输入的非空自然数表S进行该运算后产生一个具有相同元素且已排序的自然数表r，并把这个r连接到外部表变量output中去。这里is_sorted是一个函数，其说明在后边的例4中详述。

为了描述方便，在前置条件和后置条件中可采用如下几个结构：

(1) let—in Q,其含义是在let之后定义一些临时变量(名字和值)，其应用范围是谓词公式Q。这样做是为了缩短Q描述的长度。

(2) If e then P else Q，是一个条件表达式。意为若e为真，则选择P，否则选择Q。

(3) 函数调用。为了结构化地规定一个运算的功能，在前置和后置条件中可以调用函数，像例3中的is_sorted，但对调用到的函数也要继而给出形式化功能规范。一个函数的形式化功能规范分为显式规范和隐式规范两种方式。一个函数的显式规范形式为：

$f:p_1:Tp_1\times p_2:Tp_2\times\cdots\times p_n:Tp_n\to T_r$

$f(p_1, p_2, \cdots, p_n)\triangleq\cdots$

其中$p_i(i=1, 2, \cdots, n)$是函数f的参量，其相应类型为Tp_i，T_r是函数f值的类型。符号$\triangleq$右边用数学表达式给出该函数的定义。

一个函数的隐式规范形式为：

$f(p_1:T_{p_1}, p_2:Tp_2, \cdots, p_n:Tp_n)r:T$

pre-f

post-f

其中pre-f是函数参量p_1，p_2，…，p_n所应满足的条件，post-f是在计算函数f以后结果r和参量p_1，p_2，…，p_n所应满足的条件。

2.程序正确性证明机制

函数和运算的形式规范是形式化程序正确性证明的依据。所谓一个用显式定义的函数满足一个其隐式定义的形式规范，是指对于所有的函数参量在满足它的前置条件的情况下，该函数计算所得的结果符合结果类型定义要求且连同函数参量一起满足其后置条件。该概念可形式化地描述如下：设函数f的显式说明为：

$f:p:T_p\to T_r$

$f(p)\triangleq\cdots$

其隐式说明为：

$f(p:T_p)r:T_r$

pre-f$\triangle\cdots p\cdots$

post-f$\triangle\cdots p\cdots r\cdots$

其中pre-f和post-f是取布尔值的函数，而显式说明的函数f满足它的隐式说明规范当且仅当如下条件成立：

$\forall p\in T_p\cdot \text{pre-f}(p)\Longrightarrow f(p)\in T_r\wedge \text{post-f}_{(P,\ f(p))}$

其中⇒表示蕴含。根据这个公式便可把f(p)用其显式定义形式代入此公式，进而用一阶谓词逻辑推理技术可证明该函数是否满足它的隐式形式规范。

对于运算的程序实现是否满足其隐式形式规范的证明机制，由于运算中允许使用外部变量和可能出现多个输出变量，所以它比函数的证明机制要复杂一些。运算的正确性证明是基于状态这个概念的。设一个运算的形式规范如下：

OP(p:T_p)r:T_r
ext rd v_1:T_1
wr v_2:T_2
pre…p…v_1…v_2…
post…p…v_1…$\overleftarrow{v_2}$…r…v_2…

用Σ表示所有状态集合，即$\Sigma=T_p\times T_1\times T_2\times T_r$，用$\overleftarrow{\sigma}$表示执行前的状态，用$\sigma$表示执行后的状态，则正确性证明机制为：

$$\forall \overleftarrow{\sigma}\in\Sigma\cdot \text{pre-OP}(\overleftarrow{\sigma})\Rightarrow r\in T_r\wedge\exists\sigma\in\Sigma\cdot \text{post-OP}(\overleftarrow{\sigma},\sigma)$$

其中σ=exec (OP, r)，即执行运算OP之后所得到的新的状态。因此，在程序正确性证明时，只要把对运算 OP 实现的程序代入该公式的σ位置，用一阶谓词逻辑推理技术便可证明该运算的实现是否符合其形式规范。关于程序正确性证明技术可见参考文献[4]。

为了对用VDM技术建立一个形式规范过程有一个系统和全面的了解，下边给出一个完整的例子。

例4 本例是用VDM技术规定一个"教师职称提升系统"的形式化规范。该系统被描述为一个称作TEACHER-PROMOTION 的运算，其功能是根据所要提升的名额(设为m)和全体教师的教学和科研评分表，选出总成绩最好的前m名教师，并输出这些教师的有关信息。由于需要根据教学和科研评分表计算出每个教师的总成绩并进而把教师按成绩从大到小排列起来以便于选择出前m名教师，在运算TEACHER-PROMOTION的后置条件中引用了"取名额"函数GET-PERSONS，"教师排次序"函数 TEACHER-ORDER和"教师序列生成"函数 TEACHER-LIST-PRODUCTION等，而在这些函数的功能描述中又分别引用了"教师积分计算"函数 SCORE-SUM，"教学科研评分数据采集"函数TEACHING-RESEARCH-SCORE等。该形式化规范如下：

类型说明

Studentdata=Compose Studentdata of
name:String
t-score-list:seq of Choice
end
String=seq of char
Choice=seq of R
Teacher=Compose Teacher of
t-name:String
t-score:R
r-score:R
end
Researchdata=Compose Researchdata of
name1:String
r-score-list:Seq of N
end
Teachersum=Compose Teachersum of
name2:String
total-Score:R
end
Teacher-list=seq of Teachersum
Studentdata-list=seq of Studendata
Researchdata-list=seq of Researchdata

类型不变式函数

Inv-Studentdata$\triangleq\forall$mk-Studentdata (n, t)$\in$ Studentdata·n$\neq$' '$\Rightarrow$t$\neq$〔〕
Inv-Researchdata$\triangleq\forall$mk-Researchdata(n_1, r) $\in$Researchdata·$n_1\neq$' '$\Rightarrow$r$\neq$〔〕
Inv-Choice$\triangleq\forall$t$\in$Choice·lent=4

运算及函数规范

TEACHER-PROMOTION (tsl:Studendata-list,
rsl:Researchdata-list,
m:N)p:Teacher-list
ext wr output:Teacher-list
pre len tsl=len rsl $\wedge$ $\forall$i$\in$inds tsl·name (tsl〔i〕) =name1 (rsl〔i〕)
post p=GET-PERSONS(TEACHER-ORDER(TEACHER-LIST-PRODUCTION(tsl, rsl)), m)
$\wedge$output=$\leftarrow$output$\wedge$P

GET-PERSONS(td:Teacher-list,n:N) tt:Teacher-list
Pre td$\neq$〔〕$\wedge$lentd$\geqslant$n$\wedge\forall i,j\in$inds td·i$<$j $\Rightarrow$total-score (td〔i〕)$\geqslant$total-score (td〔j〕)
post len tt=n $\wedge$ elem tt$\subseteq$elem td $\wedge$ $\forall$i,j$\in$inds tt·i$<$j$\Rightarrow$total-score(tt〔i〕) $\geqslant$total-score(tt〔j〕)

```
    ∧ ∀x∈ elem tt ∀y∈ elem td⇒ el-
      emt(total-score,x) ≥ total-score(y)
```

/*从已排序的教师序列表td中取出其总成绩最大的几个教师按其总成绩从大到小次序放入另一个教师序列表tt中*/

```
TEACHER-ORDER(tl:Teacher-list)d:
      Teacher-list
  pre len tl>0
  post elem d=elem tl∧∀i, j∈inds d·i<j
          ⇒total-score(d[i])≥total-sco-
            re(d[j])
```

/*将输入教师序列表tl按每个教师的总成绩从大到小的重新排序后放入另一个教师序列表d中作为结果*/

```
TEACHER-LIST-PRODUCTION(tsl:Studen-
          tdata-list, rsl:Researchdata-list)
          t:Teacher-list
  pre len tsl=len rsl∧ ∀i∈inds tsl·name
        (tsl[i]) =name1(rsl[i])
  post if tsl=[] then t=[] else
       t=[SCORE-SUM(TEACHING-RES-
          EARCH-SCORE(hd tsl, hd
          rsl))]⌢
        TEACHING-RESEARCH-SCORE
          (tl tsl, tl rsl)
```

/*该函数是一个递归函数，其中通过引用“教师积分计算”函数SCORE-SUM和“教学科研评分采集”函数TEACHING-RESEARCH-SCORE (tl tsl, tl rsl) 计算出所有教师的总成绩并将教师的名字连同其总成绩排列在教师表t中作为结果*/

```
SCORE-SUM(te:Teacher)ts:Teachersum
  pre t-name(te)≠''
  post t-name(te) =name2(ts)∧total-score (ts)
  =t-score(te)+r-score(te)
```

/*将一个教师的教学得分t-Score(te)和其科研得分r-Score(te)相加，得到该教师的总成绩total-score(ts)，连同该教师的名字作为该函数的计算结果放入ts中*/

```
TEACHING-RESEARCH-SCORE(ts:Student-
          data, rs:Researchdata)te: Teacher
  pre r-score-list(rs)≠[]∧t-score-list(ts) ≠
      []
  post let mk-Teacher(n, t, r)=te in
       let mk-Studentdata (n1, sl)=ts in
       let Σ_{i=1}^{len sl} AVE(rsl[i])=total in
       let mk-Researchdata(n2, rl)=rs in
       n=n1 ∧ n=n2∧t=total/len sl
       ∧ r=Σ_{j=1}^{len rl} rl[j]
```

/*该函数引用AVE函数，计算一个教师的教学评分表和科研评分表后分别得出该教师的教学得分和科研得分，并连同其名字一起作为该函数的计算结果放入变量te中*/

```
AVE(c:Choice)v:R
  pre true
  post if len c=4 then
    V=C[1]×60%+C[2]×20%+C[3]×10%
      +C[4]×10%
    else V=0
```

/*按“教学效果”，“教学方法”，“教学态度”和“负责程度”四个方面所占相应比例从一个学生对一个教师的教学的评分表计算出该学生对该教师的教学得分*/

3. VDM技术的特点

通过从例4我们可以看出使用VDM规定形式化规范具有如下三个最明显的优点：

1) 只告诉计算机“做什么”。

2) 提供了程序正确性证明的依据。

3) 使规范描述简练，精确。

除这三个明显的优点之外，使用VDM还可以培训程序设计者树立牢固的先抽象，后具体和不断证明其正确性的逐步分解的自顶向下的开发思想，从而在整个程序开发的全过程中用系统而严密的方法保证所开发的程序的正确性。但是，VDM也存在着如下一些不足之处：

1) 运算冗余。由于VDM对抽象数据类型预先定义了运算，而某些用户定义的类型在规范描述中无需这么多运算，因而产生了运算冗余。

2) VDM目前还未能建立一整套描述机制，将一个大型系统分解成为许多运算而描述出这些运算之间的调用关系。这一点有待进一步研究[6]。

3) 由于采用数学符号和抽象数据类型，VDM形式规范过于形式化，往往不容易理解。这有可能造成未读懂形式规范而错误地实现其软件的情况。

三 OBJ 形式规范技术

OBJ（object 的简称）是一种代数形式化规范技术，其最基本的概念是抽象数据类型，每个抽象数据类型描述了一个客体。所谓一个抽象数据类型就是指一个或多个类型客体和其上的有关运算。OBJ具有很强的独立定义和数据抽象机制。用OBJ书写形式化规范的过程就是用代数等式定义抽象数据类型的过程，而且在定义上层抽象数据类型时可以用到下层已定义的抽象数据类型（其类型客体和其上的运算）。OBJ形式规范既具有一种基于代数等式逻辑的代数形式语义，又具有一种基于把代数等式解释为重写规则的操作语义。利用这种操作语义可证明程序的正确性。

1. OBJ形式规范的形式

一个OBJ形式规范是诸多抽象数据类型定义的集合。其中每个抽象数据类型具有如下形式：

```
obj 类型名
sorts类型客体
ops运算类型定义
vars变量定义
eqns运算功能定义
jbo
```

这里obj，sorts，ops，vars，eqns和jbo全部是关键字，其中obj之后的“类型名”指出了该抽象数据类型的名字，以便定义更上层的抽象数据类型时引用。sorts之后的“类型客体”指出了该抽象数据类型的客体名，类似于Pascal中的一个类型定义标识符。ops之后定义了该抽象数据类型上的运算的定义域和值域。vars之后定义一些在描述运算中将用到的变量。eqns之后用代数等式和函数复合机制给出该抽象数据类型上的所有运算的功能规范。其一般形式是：〈左部〉=〈右部〉。jbo只是标记该抽象数据类型描述的结束。

若在定义一个抽象数据类型sort时需要使用底层类型L中的运算，则可写成Sort/L。

例5 下边的一个客体定义了一个抽象数据类型NATURAL.

```
obj NATURAL
sorts  nat
ops o:→nat
    succ:nat→nat
jbo
```

其中o和succ是类型nat上两个构造运算，由于o无定义域，而其值域是nat，所以o可看作是nat上的常量，succ是nat上的一个一目运算。利用这个定义，我们可定义一个运算“+”如下：

```
obj Addition/NATURAL
ops  _+_:nat nat→nat
vars x, y:nat
eqns  (o+x=x)
      (succ(x)+y=succ(x+y))
jbo
```

值得注意的是，在书写一个抽象数据类型规范时，除obj部分和jbo是必须写出的外，sorts，ops，vars和eqns等部分都是根据实际需要任选的。

为了使读者能够较全面系统地了解用OBJ技术规定一个较大的形式化规范的过程，下边给出一个对整型序列进行分类的形式化规范。该规范中以自底向上顺序分别规定了抽象数据类型Boolean，Integer，seq-INT和sort-seq-INT。

例6 一个对整型序列分类的形式化规范.

```
obj Boolean/TRUTH
vars a, b:BOOL
eqns  (not(T)=F)
      (not(F)=T)
      (not(not(a))=a)
      (T and a=a)
      (a and F=F)
      (T or a=T)
      (F or a=a)
jbo
```

其中TRUTH是OBJ内部定义的布尔抽象数据类型，它具有T和F两个常量，分别表示真、假值。

```
obj Integer
sorst INT
ops
```

```
    -+-    :INT INT⟶INT
    --     :INT INT⟶INT
    -*-    :INT INT⟶INT
    -div-  :INT INT⟶INT
    -mod-  :INT INT⟶INT
    ->-    :INT INT⟶Boolean
    -<-    :INT INT⟶Boolean
    -<=-   :INT INT⟶Boolean
    ->=-   :INT INT⟶Boolean
jbo
obj seq-INT/Integer
sorts seq-INT
ops
    [ ]:          ⟶seq-INT
    [-] :INT      ⟶seq-INT
    -^-:seq-INT  seq-INT⟶seq-INT
    hd-:seq-INT⟶INT
    tl-:seq-INT⟶seq-INT
    len-:seq-INT⟶INT
vars i:INT  s:seq-INT
eqns
    ([]^S=S)
    (S^[]=S)
    (hd[]=0)
    (hd[i]=i)
    (hd([i]^S)=i)
    (tl[]=[])
    (tl[i]=[])
    (tl([i]^S)=S)
    (len[]=0)
    (len S=succ(len(tl S))
          if S<>[])
jbo
```

该抽象数据类型上定义了六个运算：[ ]，[-]，-^-，hd-，tl-和len-。它们的运算规则在eqns部分中用代数等式表达了出来。

```
obj  sort-seq split/sort-seq-INT Integer
ops
    -<-  :seq-INT INT⟶seq-INT
    ->=-:seq-INT INT⟶seq-INT
vars
    S:seq-INT i, x:INT
eqns
    ([]^x=[])
    ([i]^x=[] if i>=x)
    ([i]^x=[i] if i<x)
    (([i]^S)<x=S<x if i>=x)
    (([j]^S)<x=[i]^(S<x)if i<x)
    ([]>=x=[])
    ([i]>=x=[]if i<x)
    ([i]>=x=[i]^S>=x)if i>=x)
jbo
```

该抽象数据类型定义了两个运算：-<-和->=-，其运算对象是一个整型序列和一个整型数或变量，其运算规则在eqns部分给出。基于上述四个已定义的抽象数据类型，另一个称为sort-seq-INT的抽象数据类型定义如下：

```
obj  sort-seq-INT/sort-seq-split seq-INT-
                  Integer Boolean
ops    sort-seq-INT⟶seq-INT
       is-sorted-:seq-INT⟶Boolean
vars   S, S1, S2:seq-INT  i, j, K:INT
eqns
    (is-sorted[]=T)
    (is-sorted[i]=T)
    (is-sorted[i]^[j]=i<=j)
    is-sorted[i]^[j]^S=i<=j and
                  is-sorted([j]^S))
    (sort S=S if is-sorted S)
    (sort([i]^[j])=sort([j]^[i])if i>j)
    (sort([i]^[j]^S)=sort([j]^[i]^S)
                  if i>j)
    (sort(S^[i]^[j])=sort(S^[j]^[i])
                  if i>j)
    (sort(S1^[i]^[j]^S2)=sort(S1^[j]^
                  [i]^S2)if i>j)
    ((sort(tls<hds))^[hds]^(sort(tls>=
                  hds))=sort S  if S <>[])
(S>=x=((tls)>=x) if S<>[]and(hds)
                  <x)
(S>=x=[hds]^((tls)>=x) if S<>[]
                  and(hds)>=x)
(S<x=((tls)<x)if S<>[]and (hds)
                  >=x)
(S<x=[hds]^((tls)<x)if S<>[]and
```

(hds)<x)

jbo

该抽象数据类型中未引入新的类型客体，只是定义了两个运算*sort-*和*is-sorted-*，这是因为该两个运算是针对Seq-INT进行的。

2 构造运算的基本准则

在一个抽像数据类型上构造什么样的运算要视具体要求而定，但从其功能性质上分为如下四类：

1） 初始化运算。这种运算不以任何别的类型客体作为它的输入，而其结果却是具有自身类型客体的值，如seq-INT上的〔〕运算。

2） 构造运算。这种运算以自身类型客体和别的类型客体为输入，产生具有自身类型客体的结果，如seq-INT上的-^-运算。

3） 修改运算。这种运算修改自身类型客体的变量。如seq-INT上的tl-运算。

4） 观察运算。这种运算以自身类型客体为输入，得出具有别的类型客体的结果，如seq-INT上的len-运算。这种运算主要是用来获得自身类型客体的某些信息，像大小，元素集等。

在规定一个抽象数据类型时，其中的运算至少要包括上边四种运算中的三种，构造运算和修改运算可根据情况选用。

有了抽象数据类型定义的运算，便可写出一些计算程序，但程序中不得出现在抽象数据类型中未定义的运算。因此，要证明一个程序正确性，必须依赖于相应抽象数据类型定义中的运算定义规范。事实上，只要把运算定义中所用到的代数等式看成是用其右部可代替其左部的重写规则，即把〈左部〉=〈右部〉看成是重写规则〈左部〉→〈右部〉，便可用来证明所设计程序的正确性。关于这部分内容读者可见参考文献〔9〕。

3. OBJ技术的特点

与VDM相比，OBJ形式规范技术具有如下优点：

1） 无运算冗余。在用OBJ技术定义一个抽象数据类型中的运算时视需要多少而定，不是像VDM中预先定义好的。这样不会出现无用的运算。

2） 以性质导引规定形式规范。OBJ在规定形式规范时，根据问题的性质将其分为不同的抽象数据类型进行定义，没有状态变化的概念。

3） 程序正确性证明过程简单。利用重写规则对程序的正确性证明比起使用VDM中的一阶谓词推理技术简单。

但是，OBJ技术也存在着以下两个主要问题。

1） 定义运算的不确定性。也就是说在定义一个抽象数据类型时很难把握定义多少个运算就可充分描述该抽象数据类型的全部性质。

2） 程序执行效率低。OBJ程序设计实际上类似于函数式程序设计，所以比过程型程序执行效率低。

四 评价与结论

为了更清楚地比较VDM技术和OBJ技术各自的特点，下边列出一张表：

特 点	VDM	OBJ
可执行性	×	✓
性质导引	×	✓
模块导引	✓	×
状态变化	✓	×
指出做什么	✓	✓
指出如何做	×	✓

从表可以看出，VDM形式规范是不可执行的，而OBJ形式规范是可执行的。VDM技术是模块导引的形式化规范技术，也就是说它是以描述模块的功能为主线来描述形式规范的，而OBJ技术是性质导引的形式化规范

技术，它以描述抽象数据类型为主线来描述形式化规范的。VDM 技术以通过描述状态的变化来描述运算的功能的，而 OBJ则不是。VDM和OBJ形式规范均能指出“做什么”，但OBJ形式规范在一定程度上还指出了“如何做”。

形式化规范技术为系统而准确地开发大型软件系统提供了一条有效途径，为程序正确性证明提供了可能性。它们与非形式规范技术相比具有描述简练、明确的优点，而且对培养程序开发者树立先抽象，后具体，先全局，后局部的自顶向下形式化地开发软件的思维方法起着积极的作用。目前欧洲许多研究机构和大学正在研究建立支撑VDM 技术和OBJ技术的软件开发环境。随着对形式化技术教育的不断扩大，VDM和OBJ技术将会在更广泛的领域得到越来越多的人的应用。

致谢： J. T. Latham 博士对我学习和掌握VDM及OBJ形式化规范技术做了大量具体的指导，对此表示衷心地感谢。此外，新井敦子在我写作本文的过程中提供了许多热情的帮助和鼓励，对此表示最诚挚的谢意。

参考文献

[1] J. F. Pan, "Software Development Technique", Shanghai Science and technology document publishing house, Feb., 1986

[2] Michael Jackson, "System Development," Prentice-Hall International, Inc., U.S.A, 1983

[3] B. Cohen etc., "The Specification of Complex System", Addison Wesley Publishing Company, Inc., 1986

[4] Cliff B. Jones, "Software Development -A Rigorous Approach", Prentice-Hall International (UK) Ltd., 1988

[5] Cliff B. Jones, "Systematic Software Development Using VDM", Prentice-Hall International, Inc., London, 1980

[6] J. M. Spivey, "The Z Notation", Prentice-Hall Internation (UK) Ltd, 1989

[7] J. A. Goguen, "Some Design Principles and Theory for OBJ-O, A Language to Express and Execute Algebraic Specification of Programs", Proceedings of the International Conference on Mathematical Studies of Information Processing, Kyoto, Japan, pp 429-475, 1978

[8] Shaoying Liu, "Gradually Formal System Development Method", Proceedings of the International Conference on Systems Management' 90, Hong Kong, pp 13-19, June 1990

[9] J. A. Goguen等, "An Initial Algebra Approach to the Specification, Correctness and Implementation of Absiract Data Type", Current Trends in Programming Methodology, Vol 4-Data Structuring, pp 80-149, Raymond T. Yeh (editor), Prentice-Hall, Inc., Englewood Cliffs, New Jersey 07632, 1978