面向对象的规范描述及转换*

彭智勇 （武汉大学软件工程研究所）

摘 要

本文基于抽象数据类型提出了一种规范描述语言，该语言以多相代数作为其数学模型，并引入类型继承机制，体现了面向对象的程序设计思想。最后，探讨了关于这种语言的规范描述到面向对象程序设计环境的自动转换。

一、引 言

程序设计自动化是解决当前软件危机的一条重要途径，它越来越受到软件工作者的重视。其主要方法是对于给定的规范描述，根据一定的转换策略，将其转换成功能上与此等价的高级语言程序。由于规范描述是面向问题的，故可采用更为接近人类思维方式的高级结构，使规范描述简明易懂。同时，规范描述一般以一定理论和数学模型为基础，它的正确性证明要比高级语言程序的正确性证明容易，所以采用规范描述及转换途径不仅减轻编制程序的负担，而且还有利于提高程序的可靠性，这是一种十分有效的方法。

面向对象的程序设计思想是一种新的而且很有前途的思想，它较传统的程序设计思想更接近于人的思维，人们正逐步采用这种思想进行软件开发。为了实现这一开发过程的自动化，我们将面向对象这种思想引入到以上所描述的过程中，研制面向对象的规范描述语言以及面向对象的程序设计环境，然后实现前者到后者的自动转换。在下面几节中，我们根据面向对象这种思想的理论基础，采用多相代数这样一种数学模型，提出一种面向对象的规范描述语言，并探讨如何实现这种语言的规范描述到面向对象程序设计环境的自动转换。

二、抽象数据类型及面向对象的程序设计

2.1 抽象数据类型及其说明

抽象数据类型被认为是一种功能很强的程序设计工具。它是对数据类型的抽象，这种抽象是指它不依赖于其表示形式，有关其实现细节对用户来说是不可见的。用户只知道其操作做什么，并不了解这些操作如何实现，这样便于程序模块化，有利于用户编程和调试，因而在程序设计语言中得到了广泛采用。

为了说明抽象数据类型，代数是一种很好的说明手段。通常所采用的代数是(ADJ)(1973)提出的多相代数，它较之普遍的一相代数更易于管理。所谓多相代数就是一个载体集合以及该集合中所有载体之间的操作集，载体集合中的每个载体包括某个类的所有元素。目前越来越多的人认为数据类型就是一种代数，采用代数来描述抽象数据类型也就变得非常自然且合理。

描述抽象数据类型的代数不是任意的，它必须满足一定的条件，使之具有确定性。对于一个给定的类集S，下面我们给出一些

* 本课题得到国家863计划资助

定义：

定义1 一个操作集Σ是一族集合$\Sigma_{w,s}$，这里$s\in S$，$w\in S^*$

定义2 Σ是一个按S分类的操作集，那么一个Σ代数A包括针对每个$s\in S$的载体A_s和针对每个$\sigma\in\Sigma_{w,s}$ $(w=s_1s_2\cdots s_n)$的如下形式函数：

$$\sigma_A: A_{s_1}\times A_{s_2}\times As_3\times\cdots\times A_{s_n}\to A_s$$

。对于$\sigma\in\Sigma_{\lambda,s}$，$\sigma_A\in A_s$，$\Sigma_{\lambda,s}$也就是$A_s$中的常量集，这里λ表示空值。

定义3 如果A和B都是Σ代数，Σ—同态且

$h: A\to B$是一族函数$\langle h_s: A_s\to B_s\rangle_{s\in S}$，它使A和B的操作满足如下两个条件：

(h_0) 如果$\sigma\in\Sigma_{\lambda,s}$，那么$h_s(\sigma_A)=\sigma_B$；

(h_1) 如果$\sigma\in\Sigma_{s_1\cdots s_n,s}$和$\langle a_1,\cdots a_n\rangle\in A_{s_1}\times\cdots\times A_{s_n}$

那么$h_s[\sigma_A(a_1,\cdots,a_n)]=\sigma_B[h_{s_1}(a_1),\cdots,h_{s_n}(a_n)]$。

定义4 一个Σ代数的范畴C包括一类Σ代数以及这些代数之间的所有Σ同态。

定义5 一个代数A在Σ代数的范畴C中是初始的，当且仅当对于C中的每一个代数B存在一个唯一的同态$h: A\to B$。

可以证明如下命题：如果Σ代数范畴C中的两个代数A和A′是初始的，那么A和A′同构；如果Σ代数范畴C中的代数A″与C中初始代数A同构，那么A″也是初始的。

定义6 一个抽象数据类型是Σ代数范畴C中初始代数的同构类。

根据以上定义，通过证明可知，任何一个Σ代数范畴C中的初始代数存在并可构造之，这表明抽象数据类型可通过初始代数描述。由于初始代数在同构意义上唯一，因此抽象数据类型具有唯一性。采用初始代数描述抽象数据类型d的形式是$\langle s,\Sigma,\varepsilon\rangle$，$d\in S$，其中Σ是操作集，ε是用于描述有关操作语义的等式集。对于复杂的类型，不必立刻给出完全的说明，最好的办法是通过修改或连接已有类型来说明之。

2.2 面向对象的程序设计

面向对象的程序设计是在抽象数据类型的基础上发展起来的一种颇有前途的软件设计方法，它将数据抽象和类型继承融为一体，使人们在软件设计中普遍遵循的模块化、信息隐蔽、抽象、代码共享等思想得到了充分实现。

按照面向对象程序设计思想设计的软件，其所有成份都表示成对象。对象既是信息的存贮单元，又是信息处理的独立单位，它具有一定的内部结构和处理能力。一个软件系统往往包括很多对象，它们之间相互作用通过消息传递。消息传递是对象相互联系的唯一方式，向某对象发送消息就是要求它根据其处理能力执行某个操作。在消息传递过程中，消息发送者只知道消息接受者具有某种能力，而不知道它是如何实现这种能力的。

在人们开发的软件系统中，有很多对象表示着不同的个体，然而它们具有相同的结构和处理能力。按照面向对象的程序设计思想，这些具有相同结构和处理能力的对象用类进行描述。类包括外部特征和内部实现两个方面，类通过描述消息模式及其相应的处理能力定义对象的外部特征，通过描述内部状态的表现形式及固有处理能力的实现来定义对象的内部实现。一个类实际上定义的是一种对象类型，它描述了属于该对象类型的所有对象的性质。类的概念相当于一般程序设计语言中类型的概念，所不同的是类将数据结构及内部操作作为一个整体进行了统一描述，充分体现了抽象数据类型的思想。

根据面向对象的程序设计思想，类与类之间通过继承关系相互关联，它包含以下三个基本含义：

1. 如果类A继承类B，那么类A的对象具备类B的对象所具有的一切能力。

2. 如果类A继承类B，那么类A对象的内部结构包含了类B对象的内部结构。

3. 如果类A继承类B，那么类B中实现其对象

能力的代码可被类A所引用。

由此可见这种继承关系支持代码共享，为软件重用提供了物质条件，它是面向对象程序设计思想的非常重要的特征。

按照面向对象程序设计思想进行软件设计，其过程是：首先确定问题所涉及的对象，并根据对象的内部结构和外部特征将其分成不同的类，然后实现类的内部表示和处理能力，并建立类与类之间的继承关系，最后由类生成问题所涉及的对象，并按问题的要求规定对象之间的消息传递序列。当消息按照规定的顺序进行传递，有关对象对所接到的消息进行了正确处理时，所设计的软件系统就能完成一定的操作，实现用户所需要的功能。

三、 面向对象的规范描述语言

规范描述语言是实现程序自动转换系统的关键，它不仅影响系统处理问题的能力和范围，还会影响系统的实现和功效，因而采用适当的规范描述语言对转换系统而言是至关重要的。面向对象的规范描述语言是形式规范描述语言的一种，它体现面向对象的程序设计思想，其特点是将所描述软件系统中的所有信息按抽象数据类型进行分类，对抽象数据类型的描述采用多相代数，其上每个操作的语义均由代数公理精确地定义，并引入类型继承机制，使所有类型之间建立一种可实现信息共享的层次关系。该语言的抽象程度高，并具有良好的模块性，而且通过继承实现信息共享，便于再利用，这样既精确、严谨，又简明、易懂，是一种较理想的规范描述语言。

用面向对象的规范描述语言说明的规范描述由一系列的type组成，每个type的组成成份及说明形式如下：

```
Type type名;
SuperType supertype名;
Import sort名表  From type名;
      ……
ExportSorts  Sort名表;
Export Operations  操作名表;
Sorts sort名表
Operators
    操作名：sort名表→sort名;
      ……
Declare
    变量名：sort名;
      ……
Operator Axioms
    等式左部⇐⇒等式右部;
    ……
EndType
```

上面，以大写英文字母开头的单词是语言的保留字，在语法定义部分，操作名后的类型名表给出了该操作定义域的类型名，其后的类型名为该操作的结果类型名；语义定义部分采用代数等式刻划操作的语义，等式右部还允许条件表达式和递归。

该语言的每一个type说明一个多相代数。称一个代数满足某type是指对该type中的任一等式，其中的变量不论取相应类型值集的什么值，该等式在此代数中都成立。根据第二节我们知道满足某type的代数可能很多，为了保证其实现的唯一性，我们把满足此type的初始代数作为该type的实现。

在该语言所描述的软件系统中，所有type依靠其SuperType域建立一种单向继承关系，即子type继承父type的所有信息，这样就使得规范描述能够按照自顶向下的方式逐步进行。我们把最基本的type称之为BASICTYPE，其它所有type都是它的后继子type，这种层次关系与面向对象程序设计环境中类的层次关系相一致。

下面给出一个用该语言说明的规范描述实例：

```
Type STACK;
SuperType BASICTYPE;
Import bool From BOOL;
Import int From INT;
ExportSorts stack;
ExportOperations bottom push pop-
```

```
            top-  empty-;
Sorts stack;
Operators
    bottom: ⟶stack;
    push: stack, int⟶stack;
    pop-: stack⟶stack;
    top: stack⟶int;
    empty-: stack⟶bool;
    underflow: ⟶stack;
    topnil: ⟶int;
Declare
    i: int;
    s: stack;
Operator Axioms
    pop push (s, i) ⟺s;
    top push (s, i) ⟺i;
    empty bottom⟺TRUE;
    empty push (s, i) ⟺FALSE;
    pop bottom⟺underflow;
    top bottom⟺topnil
EndType.
```

本例描述一个整数栈，它只含一个type，名为STACK。在描述中还用到它其type，它们分别是BOOL和INT，其引入表表明此type可引用BOOL和INT中的bool和int及有关操作。通过输出表列出了可供其它type引用的sort名和操作名。在语法定义部分共定义了7个操作，语义定义部分用6条等式刻划了前述操作的语义。

由上例可以看出类型T中的操作一般可分成：

(1) 生成操作　其作用是从非T类型的量或空生成T类型的量。如本例中的bottom操作生成了一个空栈；

(2) 构造操作　其作用是用非T类型的量和T类型的量共同构造T类型的量。如push操作用非T类型int的量和T类型stack的量共同构造T类型stack的量；

(3) 扩充操作　作用同构造操作。但其任何结果值均可由生成、构造操作的有限次作用构成。如例中的pop操作，其任何结果值均可由bottom，push操作的有限次作用构成；

(4) 提取操作　结果类型非T的那些操作，或是从T中提取某些信息。如top，empty操作。前者得到栈顶元素的内容，后者测试栈是否为空。

四、 规范描述的自动转换

面向对象的规范描述是严格形式化的，将其自动转换成可执行程序是可能的。为了保证风格上的一致性，我们将其转换成面向对象程序设计环境中可执行的程序。这里所选择的程序设计环境是Smalltalk-80系统，该系统以面向对象程序设计语言为核心，集各种软件开发工具于一体，配有图形功能的多窗口的用户交互界面，为用户编程、调试和运行提供全面而又友善的支持。

在Smalltalk-80系统中，其所有成份都表示成对象，对象通过传递消息相互作用，对对象的性质采用类进行描述，父类与子类通过单向继承实现信息共享。利用Smalltalk-80系统进行软件设计的过程就是创建新的类，由类生成所需的对象，并书写表达式说明有关对象之间消息传递序列的过程。

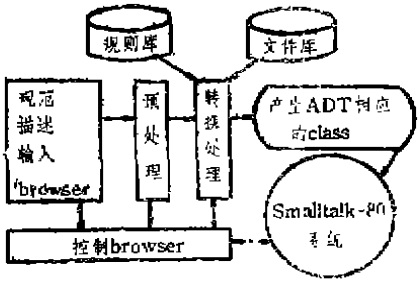
根据以上说明可以看出面向对象的规范描述语言与Smalltalk-80系统的语言在风格上和形式上具有较强的一致性，我们可以把规范描述中的type和Smalltalk-80系统中的class对应起来，这样type中用等式公理描述的操作就与class中的方法相对应。可以这样认为规范描述中的type是抽象数据类型的规格说明，Smalltalk-80系统中的class是其实现。基于这一点，我们能够在Smalltalk-80系统的基础上实现规范描述到Smalltalk-80系统中可执行程序的自动转换。

我们将研制的是一个基于知识的自动转换系统，它将面向对象的规范描述转换成功能上与之等价的Smalltalk-80语言的程序，其基本流图如下图。(见下页)

此转换系统建立在Smalltalk-80系统上，它主要由转换控制和两个库组成。转换控制又分为四部分：

(1) 规范描述输入browser，它通过多窗口系统(browser)的形式，支持用户输入规范描述。

(2) 预处理，它对所输入的规范描述进行词



法、语法检查。

(3) 转换处理，它利用文件库，并从规则库中选择恰当的规则进行匹配、应用，逐步将规范描述中的type变换成Smalltalk-80系统中的类。

(4) 控制browser，它控制整个转换过程，提供诸如说明文件输入/输出，预处理，说明文件到Smalltalk-80系统中类的转换等操作。

两个库为：

(1) 规则库，它表示转换所需知识的规则的集合，内容包括与源语言有关的知识、与目标语言有关的知识及变换法则。

(2) 文件库，存放已定义的type及相应的Smalltark-80系统中的类名，包括系统预定义的和用户自行定义的type。库中的type可供后继任何type引用，这将有益于提高软件的可重用性。

五、 结束语

基于抽象数据类型和面向对象的程序设计思想，我们提出了一个面向对象的规范描述语言，该语言与Smalltalk-80语言在风格和形式上相一致，它为实现其规范描述到Smalltalk-80系统中可执行程序的自动转换提供了可能，我们给出了此转换系统的初步方案，具体实现有待于进一步研究和探讨。

通过分析可以看出此规范描述语言具有如下特点：

(1) 通过继承关系使type按层次联接在一起，它便于以自顶向下的方式进行规范描述的说明。

(2) type的说明部分提供type检查机制，以便进行正确性验证。

(3) 用此语言说明的规范描述中的每个type是自包含的，它独立于所使用的上下文，因而满足模块性和可重用的要求。

(4) 该语言的形式化和抽象程度较高，精确简练，易读性较强。

(5) 建立文件库保存全部已定义的type，可供任意后继type引用，提高了软件的可重用性。

参考文献

1. J.A.Goguen, J.W.Thatcher and J.B. Wright, "An Initial Algebra Approach to the Specification, Correctness and Implementation of Abstract Data Types, "in Current Trends in Programming Methodology vol.4 Data Structuring, pp.80-149, Prentice-Hall, 1978.
2. YouLiang Zhong, Seirei Ishizuka, Ryuich Enari, "Integrating Abstract Data Types with Object-Oriented Programming by Specification-Based Approach," the Computer society of the IEEE 1730 Massachusetts Avenue NW Washington, DC 20036-1903.
3. 彭智勇、陈火旺等，Smalltalk-80及实现，计算机科学.1988.5.

国际神经网络大会

International Neural Network Conference——INNC 90

时间：1990年7月9日—13日 **地点**：法国巴黎 **大会主席**：Bernard Widrow, Bernard Angeniol

程序委员会主席：Teuvo Kohonen

内容及分组：**A. 应用**：包括图象处理、信号处理、语言处理、机器人和控制、优化、分类设计和预测、专家系统应用。**B. 实现**：包括神经生理模型、认知科学、电子神经计算机、光学神经计算机、神经网络算法在并行硬件上的实现、网络定义语言和开发环境、基准程序及系统。**C. 理论**：包括监督式学习、非监督式学习、联想存储器、体系结构、网络动态分析、统计及概率模型。**D. 商品**：包括E.E.C.计划、政府资助的项目、用于开发神经网络应用的商业产品、采用了神经网络技术的商业产品、1992年欧洲的高技术及规化、公司策略。

(蔡义发供稿)