第35卷 第2期
1996年 3月

中山大学学报(自然科学版)
ACTA SCIENTIARUM NATURALIUM
UNIVERSITATIS SUNYATSENI

Vol. 35 No. 2
Mar. 1996

对象式程序的正确性验证方法研究

姚卿达 唐长宁 黄 璇
(中山大学软件研究所，广州 510275)

摘 要 介绍当今流行的各种程序正确性验证方法的基础上，讨论了在面向对象程序设计的正确性验证中所遇到的特殊问题，提出了类级正确性和系统级正确性的概念，将公理证明方法加以扩充，使之可以用于面向对象程序系统的证明，并用特征调用的概念解决了证明中出现的计算过程中止性的问题.

关键词 对象式程序设计，类级正确性，系统级正确性，特征调用
分类号 TP311.1

程序正确性验证方法研究已有几十年的历史，人们相继提出的方法有公理方法、操作方法和指称方法等. 但这些方法所研究的都是过程性程序语言的正确性问题，由于面向对象的程序设计语言在概念上与传统的过程性程序设计语言有较大的差别，因此，传统的程序正确性验证方法在用于面向对象程序设计时必然有其局限性. 本文讨论以Hoare逻辑为基础的扩充的公理方法，用于对EIFFEL语言的验证.

1 过程式程序正确性验证的公理方法

Floyd发表的《给程序赋予意义》[1]一文，正式确立了通过对程序本身的研究来验证程序的方法. 提出了用断言方法证明框图程序的正确性，即在框图的每条边上附上一个断言(谓词公式)，其意义是，每当程序执行到这条边时，此断言为真. 对于循环，即框图上的一个回路，他定义了一个切点，并在切点上置一个断言P. 然后他证明，若回路开始执行时切点上的P为真，那么，当循环回到该点时P仍将为真，这就是循环不变式的思想，P就是循环不变式. Hoare在Floyd的基础上，发表了论文《计算机程序设计的公理基础》[2]，叙述了另一种程序正确性验证方法，即公理方法. 这种方法利用一个含有程序公理和推导规则的逻辑系统来证明程序的部分正确性. 此系统是一阶谓词逻辑的扩充，称为Hoare逻辑.

用公理方法描述的程序正确性分为两种：

(1) 完全正确性断言$\{\Phi\}S\{\psi\}$：若程序S的执行开始于一个满足Φ的状态，则这个执行必在有限时间内终止且终止时的状态满足ψ.

收稿日期：1995-03-23 姚卿达，男，57岁，教授

(2) 部分正确性断言$\{\Phi\}S[\psi]$:若程序S的执行开始于一个满足Φ的状态,且若此执行能在有限时间内终止,则终止时的状态满足ψ.

在文[3]中,程序的正确性证明被定义为:设$\{\Phi,\psi\}$是一个规范,S是按照这个规范要求设计的,且是由语句$S_1,S_2,\cdots S_n$组成的一个枚举型程序(指其执行等于组成它的各语句的逐一执行,其中每个语句都只有一个入口和一个出口,且没有*GOTO*转移). 令$\Phi_1,\psi_1,\Phi_2,\psi_2,\cdots,\Phi_n,\psi_n$是2n个谓词,$\Phi_1=\Phi,\psi_n=\psi$,如果所有断言$\{\Phi_i\}S_i\{\psi_i\}(i=1,2,\cdots,n)$为真,且每个蕴涵$\psi_i=\Phi_{i+1}(i=1,2,\cdots,n-1)$成立,就称$(\Phi_1,\psi_1),(\Phi_2,\psi_2),\cdots,(\Phi_n,\psi_n)$是$\{\Phi\}S\{\psi\}$的一个证明.

需要注意的是,无论是完全正确性证明还是部分正确性证明,它所描述的都是程序的执行.

2 对象式程序验证的特点

对象式程序设计是把软件系统构造成类(或谓抽象数据类型的实现)的结构化集合,在一个程序系统中,类的地位平等,而且是无序的. 面向对象强调的是从内部结构上模拟客观世界,而不是如何实现那些功能,因此在面向对象语言(如EIFFEL)的程序文本中,并没有对整个程序的运行的直接描述. EIFFEL语言具有动态和静态两种环境,在静态环境下表现出来的是类和程序文本,在动态环境下是对象和消息传递. 由于目前的程序验证方法都是针对程序文本而来的,对于过程式的程序来说,程序的动态环境和静态环境是基本一致的,即程序文本中已经指出程序如何执行,或者只要对程序文本作一些不必加以证明的转换就可以得到其动态环境. 因此所谓程序执行时的验证,只要按照程序文本的"执行"(一种想象中的动态过程)进行验证,如果程序的执行结果(抽象机而非物理机的执行结果)与其规范说明所描述的语义是一致的,则该程序是正确的.

但对于EIFFEL这种面向对象的程序,其设计是通过静态类型定义的概念来实现,继承和允引等机制都是在静态环境下的概念,每个EIFFEL实体有一个静态类型. 运行时它可以限制为该类后代的动态模型. 多态和动态定连是通过允许类操作可以在后代中重新定义来获得. 静态环境和动态环境的差别较大,一个是无序的类的集合,一个是有序的对象的计算序列. 在面向对象的程序文本中根本见不到整个程序执行的过程,所见到的都只是一些静态的类,如果不修改程序正确性验证的定义,则在这种情况下的原有的按照程序文本进行证明的方式无法证明整个程序系统的正确性.

3 类级正确性

为了对面向对象的程序进行正确性验证,可以根据面向对象的程序的文本生成一个虚拟的执行过程,也就是说,将面向对象的程序转换为一个离散的、可依照原有方法进行验证的语句序列,在这种转化中,并不改变面向对象程序的基本性质(如封装、继承等等). 按照这种思路,我们把EIFFEL语言的验证分为两个层次:一是类级的正确性,二是系统级的正确性. 类中的特征按照其定义形式可以分为两种:属性和程式. 两者的差别在于程式说明中含有一个程式体,描述了一系列语句的执行,而属性说明则没有程式体. 文[4]中给出了类级正确性的定义.

定义 1　一个类是正确的，当且仅当它是一致的，并且类中的每个程式都是检查正确的、循环正确的.

一个类 C 是一致的，是指它满足：①$\{pre_{create}\}do_{create}\{INV_c\}$；② 对于类 C 的每个程式 r，$\{pre_r \wedge INV_c\}do_r\{post_r \wedge INV_c\}$. 也就是说，如果它的每个程式从满足前置条件的状态出发，执行后可以满足后置条件.

所谓程式是检查正确的，指如果该程式的每条语句 C 的执行均满足其中所列的断言. 所谓程式是循环正确的，是指对于该程序包含的每个循环都有：①{true}INIT{INV}；②{true}INIT{VAR >= 0}；③{INV and not EXIT}BODY {INV}；④{INV and not EXIT and VAR = V_0}BODY {0 <= VAR < V_0}. 即该程式包含的每个循环都可以中止. 这里 INV_c 表示类 C 的不变式，pre_r 表示程式 r 的前置断言，$post_r$ 表示程式 r 的后置断言，VAR 是循环变式，EXIT 是循环出口条件，BODY 是循环体.

文〔5〕中扩充了类的正确性，增加了异常处理的要求. 所谓程式是异常处理正确的，是指：① 补救语句通过重执命令 retry 而结束时，满足程式的后置条件和类不变式；或 ② 补救语句没有通过执行重执命令 retry 而结束时满足类不变式.

在面向对象的程序系统的类的正确性证明中，由于任何一个类的内部数据和操作对其它类来说是不可见的，当其它类的实例按照正确的手段来调用该类的实例而产生的错误应由该类负责. 因此在任何一个类的正确性证明过程中，不应该涉及其它与之相关的类的正确性证明，即在该类的正确性证明过程中，凡是涉及到其它类的过程调用，我们都假定该调用是正确的. 由此得出：

定义 2(调用无关条件下的正确性)　一个满足异常处理正确性的类 C，对于其中的任何一个程式 r，若该程式中存在对其它类的特征调用{pre}S{post}，这里{pre}和{post}分别是程式 r 中语句的前置条件和后置条件，只要在进行如下置换之后程式 r 满足检查正确性和循环正确性，则称类 C 在调用无关条件下是正确的.

(1) 对于形如{pre}TARGET. FNAME{post}的过程性特征调用，将其后置条件换为{pre ⊢ post}.

(2) 对于形如 X = TARGET. FNAME(函数调用)或:X:TARGET. FNAME(属性调用)的非过程性特征调用，把程式 r 中所有对 X 的操作的后置条件{post}都置换成后置条件{pre ⊢ post}.

定义 3(面向对象程序系统的类级正确性)　若该程序系统的任何一个类在调用无关条件下都是正确的，则一个面向对象的程序系统是类级正确的.

4　系统级正确性

在面向对象的系统中包含两种机制：一种是以类为模型产生的对象，另一种是对象之间的通信. 对象进行通信时相互之间传递的消息分为两个子集，一个是请求消息集，记为 ReqM，表示计算任务；另一个是完成消息集，记为 FinM，表示计算结果. 其中 ReqM ∩ FinM = Φ.

将一个对象定义为一个六元组(MA，RM，DS，CS，ds，B). 其中，MA 为通信地址，RM 为识认消息集，DS 为数据状态集，CS 为通信状态集，ds ∈ DS 为初始数据状态集，B：

$DS \times CS \times E \to DS \times CS \times E \times 2^o$ 为行为，这里 E 为事件集，o 为对象集.

事件是一个三元组(s,d,m)，其中，s 和 d 是对象地址，m 是消息，事件(s,d,m)表示对象 s 和对象 d 发送消息 m. 对象 a 和对象 b 的一次通信是指对象 a 和对象 b 之间先后发生的一对事件(a,b,m)和(b,a,m′)，这里，$m \in ReqM, m' \in FinM$. 通信(a,b,m)(b,a,m′)可以统一描述为(a,b,(m,m′)).

为了定义系统级的正确性，我们关心的是程序的系统运行的形式描述，面向对象的程序系统的运行是用系统格局序列来描述的.

定义 4(格局) 面向对象的系统的格局是一个三元组(当前事件，当前所含对象，每个对象的状态)，记为(e,obj,CON)，$e \in E$ 为事件，obj 为对象集，CON 定义为集合$\{(a, f) | A \in obj, MA(A) = a, f \in DS(A) \times CS(A)\}$，且 $A \in obj, (MA(A), f) \in CON, (a, f) \in CON, A \in obj$，使得 $MA(A) = a$，且 $f \in DS(A) \times CS(A)$.

一个格局称为合法格局，是指事件中消息接收者为当前系统中存在的某对象，且该对象中存在处理该消息的行为，并且该消息满足实现该行为的条件. 由于消息传递是由特 征调用来实现的，因此合法格局的另一种说法是：在调用某个类的特征之前已经创建了该类的实例，且该类中存在所要调用的特征，且满足特征调用的前置条件.

定义 5(格局转换) 若格局(e_1, obj_1, CON_1)是合法格局，且在当前事件 e_1 完成之后得到(e_2, obj_2, CON_2)，则格局(e_2, obj_2, CON_2)称为是由格局(e_1, obj_1, CON_1)转换而来的.

EIFFEL 中没有主程序的概念，而是由一些相关(继承，允引等)的类构成一个系统，在这些类中有一个根类(ROOT CLASS)，以启动整个系统的运行. 系统的启动执行过程是先创建启动类的实例，然后向该实例发送消息(特征调用).

定义 6(启动正确性) 以类 C 为根类的面向对象的程序系统 S 称为启动正确的，指：① C 中有自定义或系统缺省定义的对象创建程式. ② 若 c 是系统初始对象，且由程序系统 S 之外向 S 发消息(a,c,m)来启动系统，则必有 $m \in RM$(RM 是对象 c 的识认消息集).

定义 7(格局转换正确性) 如果格局$(e_{i+1}, obj_{i+1}, CON_{i+1})$是合法的，则从合法格局$(e_i, obj_i, CON_i)$到格局$(e_{i+1}, obj_{i+1}, CON_{i+1})$的转换是正确的.

由于面向对象程序系统中的特征调用是可以循环的，所以其正确性也包含两个方面：启动正确性和格局转换正确性保证系统的正确运行；而循环特征调用的终止性保证程序的运行可以终止.

定义 8(终止正确性) 如果一个面向对象的计算序列的第一个事件是(a, a_1, m)，计算序列有限，最后一个事件是(a_n, a, m')，这里 a 是系统的启动对象，$a_i (i = 1, 2, \cdots, n)$是计算过程中使用的对象，m 是 a 向 a_1 发出的请求消息，m′ 是关于请求消息 m 的完成消息，则称该程序系统是终止正确的.

定义 9(系统级正确性) 一个面向对象的程序称为是系统级正确的，当且仅当它的任何一个计算任务其格局转换序列都是合法的，即该计算任务是启动正确的、格局转换正确的和终止正确的.

5 类级正确性和系统级正确性的关系及证明

类级的正确性定义可保证：如果程式调用开始时满足前置条件，则这个执行必在有

限时间内终止，且终止时满足后置条件. 但前置条件的正确性并不是由程式 S 保证的，而是由调用者决定的. 因此类级的正确性不能代表软件系统的正确性，它只代表待证的程序系统的"静态正确性"，只有加上系统级的正确性，即"动态正确性"才完整. 而在系统级的正确性证明中，其格局转换正确性是由类级正确性提供保证的. 由于面向对象程序的特殊性，把公理证明方法针对程序文本的证明修改为对程序系统的格局序列的证明.

定义 10(面向对象程序系统的正确性证明)　P 是一个面向对象的程序系统，由类 $C_1,C_2,\cdots,C_n$ 组成(n 为有限自然数)，C_1 是其根类. S 是这个程序系统的执行的格局转换序列. $\{\Phi,\psi\}$ 是一个规范说明，描述P的执行序列S. 令 $\Phi_1,\psi_1,\Phi_2,\psi_2,\cdots,\Phi_n,\psi_n$ 是 2n 个谓词，$\Phi_1=\Phi,\psi_n=\psi$，如果所有断言：$\{\Phi_i\}S_i\{\psi_i\}$，$(i=1,2,\cdots,N)$ 为真，且每个蕴涵 $\psi_i\Rightarrow\Phi_{i+1}$，$(i=1,2,\cdots,n-1)$ 成立，就称 $(\Phi_1,\psi_1),(\Phi_2,\psi_2),\cdots,(\Phi_n,\psi_n)$ 是 $\{\Phi\}S\{\psi\}$ 的一个证明.

对象式程序的正确性验证是一个难度较大的课题，接下来还有很多工作要做. 我们将继续研究扩充的公理方法，并建立一个实验系统.

参　考　文　献

1　Floyd R W. Assigning Meanings to Programs. Proc Symposium on Applied Mathematics. American Mathematical Society, 1967,19

2　Hoare C A R. An Axiomatic Approach to Computer Programming. Comm of the ACM,1969,12(10)

3　陈火旺，罗朝晖，马庆鸣. 程序设计方法学基础. 长沙：湖南科学技术出版社，1987

4　Meyer B. Eiffel:The language, Prentice Hall,1992

5　徐家福，王志坚，瞿成祥. 对象式程序设计语言. 南京：南京大学出版社，1992

6　李师贤，阮文江. Eiffel 语言的语义. 软件学报，1995，6(1)：17～25

7　姚卿达，唐长宁，董慧红. 面向对象程序设计语言 Eiffel 的程序正确性验证工具. 计算机科学，1994，2：63～67

Correctness Certification of Object – Oriented Programming

*Yao Qingda**　　*Tang Changning*　　*Huang Xuan*

Abstract　In this paper, we investigated various traditional methods in the field of the programming correctness certification, and discussed especially the problem of the cerfifccation for object — oriented programming. We proposed the conception of class correctness and system correctness for the certification of object — oriented programming.

Keywords　object — oriented programming, class correctness, system correctness, feature call

*　Institute of Software, Zhongshan University, Guangzhou 510275