维普资讯 http://www.cqvip.com

计算机科学 1995 Vol. 22 №. 2

面向对象程序设计语言的形式语义研究

瞿裕忠 王志坚 徐家福
(南京大学计算机软件研究所 南京 210093)

摘 要 Object-oriented programming languages have become the important tools for programming-in-the-large. This paper simply illustrates the importance of the research on formal semantics of object-oriented programming languages, gives a survey of the research work on this area, and presents some development of our work on this area, and finally outlines some direction of the further research.

关键词 Programming languge, Object-oriented programming languge, Formal semantic.

一、引言

面向对象程序设计语言(以下简称面向对象语言)的基本思想起源于六十年代中期的 Simula 语言,在七十年代的 Smalltalk 语言及环境中得到发展,在八十年代的 Eiffel、C++等语言中进一步得到巩固和完善。面向对象语言中的语言成份(如类、继承等)使得程序能够从组织结构上模拟客观世界。目前,关于面向对象语言的认识逐渐趋于一致,其基本概念包括对象、类、继承、多态、和动态定连[1-2]。对象有客观对象、问题对象和计算机对象之分,计算机对象是具有通信和独立计算能力的封装体,用以模拟问题域中的对象。类是面向对象语言中程序的基本构件,它刻画一组对象的共性,即共同的数据结构和处理消息的行为,是抽象数据类型的具体实现。继承是在已有类的基础上构造新类的一种语言机制,它使得具有继承关系的不同类共享特征(属性和程式)。多态和动态定连是两个相关的实现级概念,能增强语言的表达能力。面向对象语言因其自身的特点能支持软件的易复用性、易扩充性、和易维护性等,从而日益受到人们的重视。

面向对象语言因其优越性已逐渐成为大型软件开发的重要语言工具,其形式语义的研究也日显重要。从面向对象软件开发这一角度来看,在使用面向对象语言开发软件时,类是程序的基本构件,程序人员可以在已有类库的基础上定义新类,面向对象语言的继承机制能够使得定义新类的代码尽可能少,因而类库的复用对于提高软件生产率意义重大,且在大型软件的面向对象开发中,类库通常十分庞大,需借助于计算机系统来进行类库的复用或类库转换(如将 Smalltalk 环境中的类库转换为 C++的类库),从而需要对面向对象语言的语义进行形式化研究。其次,从面向对象语言本身来看,其形式语义研究将大大提高我们对面向对象语言本质的理解,从面向对象语言中的一些语言成分的相互关联及各自的利弊能得到揭示,从面为语言的改进和设计提供理论依据。

因此,面向对象语言形式语义研究对于发展面向对象软件开发的形式化技术和推动面向对象语言的不断完善都具有重要的意义。正因为如此,许多学者把注意力集中在这一研究领域上,其中有 M. Woczko, W. Cook, J. A. Goguen, L. Cardelli, P. America, R. Milner, D. Walker 等人。下面综述一些代表性工作。

二、指称语义方法

指称语义方法基于"必须把语言的定义和其实现方法区分开来"这一观点,通过一组递归定义的函数来描述语义,这些函数把语法域映入语义域,后者是数学对象,这些函数称为语义函数,它们的存在性建立在 D. Scott 等人创建的域论之上。目前,指称语义方法依然被认为是"标准"的语义描述方法。

M. Wolczko 率先采用指称语义方法刻画了 Smalltalk-80 的形式语义,提出了对象存储域的概

瞿裕忠 博士,讲师,主攻形式语义学和面向对象方法等。王志坚 博士,副教授,主攻逻辑程序设计和面向对象方法等。徐家福 教授,博士导师。

维普资讯 http://www.cqvip.com

念，与传统指称语义域中的存储域(Store)的根本区别在于：在存储域中，地址所对应的是可存储值，如整数、整数数组，而在对象存储域(Object-memory)中，地址所对应的均为对象，如基本对象、复合对象。一个对象为一个二元组，其中一个用来标明该对象的类别，另一个用来表示对象的内部状态，其内部状态只有当该对象接收到消息后，才能在处理消息时改变。文[3]对于Smalltalk-80的继承通过程式查找来进行解释，带有操作语义的色彩，具体作法是：对象接收到消息后，从该对象所在类向上查找(如果对象与伪变量super相联，则从当前类的父类开始查找)，如果找到相应的程式，则执行之，否则反馈对应于DoesNotUnderstand：的相应信息。Wolczko在文[4]中充分阐述了他关于面向对象语言指称语义研究的思想，在对象存储域的基础上，将类解释为该类对象的通信界面(即处理消息的功能)，这样，对象处理消息的能力通过对象的类别得到间接的描述。但是，该文对于继承并没有给出直接的语义解释，而是采用转换解释方法，首先对于一个无继承的语言给出其指称语义，对于有继承的语言，先把相应的程序通过语法级的预处理(类似于程式拷贝)转化为无继承的程序，然后再对之进行语义解释。事实上，该文并没有具体给出这种语法级预处理的形式描述，因此，对于继承的语义刻画显得欠缺。

以W. Cook为代表的工作，注重继承的语义解释，将对象解释为记录，其中包含对象所拥有的操作，将类解释为对象的生成子(generator)，将继承解释为通过已有生成子及新增代码的含义组合新的生成子的复合机制。W. Cook在文[5]中提出了他的思想，并描述了一个函数式(无状态)面向对象的语言的指称语义。A. V. Hense[6]发展并推广了W. Cook的工作，使之能应用于带状态的面向对象语言。这些工作侧重于对类的继承性描述，强调对象是数据与操作的封装，对于消息处理(或程式调用)的语义解释非常直观，直接解释为域选择操作。然而，类作为数据类型这一特性被忽略了，同一类的对象所拥有的共性没有得到较好的语义描述。

三、代数语义方法

代数语义学对抽象数据类型(ADT)已有深入研究，为解释和描述面向对象语言的ADT和继承性，J. A. Goguen[7]等人发展了有序类别代数理论，在多类别型构(MSS)、多类别代数(MSA)等概念的基础上定义了有序类别型构(OSS)、有序类别代数(OSA)等概念，并研究了其相关理论。下面简单介绍MSA理论和OSA理论中的基本概念及基本思想。

定义1 (多类别型构/多类别代数，MSS/MSA) 一个多类别型构(many-sorted signature)是一个序对$(S, \sum)$，其中S是类别名的集合，$\sum$是一族操作符集合：$\sum_{w,c} | w\in S^*, c\in S\}$($S^*$为S的元素序列集)。一个$(S, \sum)$-代数A是一族集合$\{A_s | s\in S\}$和一族函数$\{A_\sigma^{w,c} : A_w \to A_c | \sigma\in \sum_{w,c}\}$，其中，当$w=nil$(空序列)时，$A_w$为单元素集；当$w=(s1,\cdots,sn)$时，$A_w = A_{s1}\times\cdots\times A_{sn}$。

定义2 (有序类别型构/有序类别代数，OSS/OSA) 一个有序类别型构(order-sorted signature)是一个三元组$(S, \leqslant, \sum)$，其中$(S, \leqslant)$是一个偏序集，$(S, \sum)$是一个多类别型构，且满足下列单调性条件：若$\sigma\in \sum_{w1,c1}\cap\sum_{w2,c2}$且$w1\leqslant w2$，则$c1\leqslant c2$。一个$(S, \leqslant, \sum)$-代数A是一个$(S, \sum)$-代数，且满足下列条件：(1)若$s\leqslant t$时$A_s\subseteq A_t$；(2)若$\sigma\in \sum_{w1,c1}\cap\sum_{w2,c2}$且$w1\leqslant w2$，则对任意的$x\in A_{w1}$，$A_\sigma^{w1,c1}(x) = A_\sigma^{w2,c2}(x)$。

可在$(S, \sum)$-代数之间定义$\sum$-同态，构成一个范畴，从而出现了初始$\sum$-代数这一概念。同样，$(S, \leqslant, \sum)$-代数及其有序类别$\sum$-同态构成一个范畴，从而也有了初始$(S, \leqslant, \sum)$-代数这一概念。不同于MSA理论中初始$\sum$-代数的有限可构造性，对应的有序类别项代数并不一定是初始$(S, \leqslant, \sum)$-代数。然而，初始代数通常作为类型概念。鉴于此，需要在有序类别型构上寻求一定的约束以保证有序类别项代数的初始性。文[7]给出了正则有序类别型构的概念。

定义3 (正则有序类别型构，ROSS) 设$(S, \leqslant, \sum)$是一个有序类别型构，称$(S, \leqslant, \sum)$是正则有序类别型构，是指满足下列正则性(regularity)条件：对任意的σ，任意的$w0\in S^*$，$\{w\in S^* | w0\leqslant w \wedge \exists\, c\in S, \sigma\in \sum_{w,c}\}$如果非空则有最小元。

文[7]证明了正则有序类别型构上的有序类别项代数是初始有序类别代数，且具有MSA理论中相似的性质。正则性条件的直观含义值得注意，它是用来保证有序类别代数对多态操作符之语义解释的一致性(从而也保证了无歧义性)，以符合有序类别代数理论对于子类型的解释思想。从可操作性准则

维普资讯 http://www.cqvip.com

来看，操作符与函数的连接应该按参数类别序列查寻最小的 w 满足 $\sigma \in \sum_{w,c}$（存在 $c \in S$），从而完成操作符与函数间的定连。这里有必要提及，在面向对象语言中，程式调用与相应可执行代码之间的动态定连也有类似问题，其根本区别在于相应可执行代码的确定和连接是按被调用对象的类别在继承图中向上查找的。

以上介绍的是 MSA 理论和 OSA 理论中的基本概念及基本思想，可以看出 OSA 代数可用来给出单个类型或一组相关类型的语义解释，对于子类型概念的解释在于如下两点：其一，子类型的载体集是子集；其二，对于参数类别之间有序的多态操作，它们的语义解释在公共子集上保持一致。文[12]还研究了带等式的 OSS 及其上的 OSA。然而，OSA 理论对于继承的语义解释依然停留在具有良好性质的子类型概念上（如 OBJ 语言族[6]中的子类型概念。）

四、其它

逻辑类型理论对于解释和描述面向对象语言的 ATD 和继承性具有积极的意义，它遵循"公式作为类型"的原则来解释 ADT，通过公式上的某种序关系来反映类型间的继承关系，这种序关系通常由推导规则来刻画。文[9]提出的基于二阶 λ-演算的类型系统模型乃是其中的代表性工作。该模型以记录和封装机制模拟对象及信息隐蔽，以记录型公式模拟类，以存在量化公式模拟 ADT，以全称量化公式模拟参数多态类型，以约束性量化公式来模拟更为复杂的数据类型，引入了序关系的推导规则来刻画类型间的继承关系。这一形式系统为抽象数据类型、多态和子类型概念的理解提供了统一的数学架构，同时其形式语言 Fun 为强类型的面向对象语言奠定了较好的数学基础。但是，这一模型对于继承的形式刻画有其局限性，其原因在于公式间的序关系受限于严格的推导规则。在此基础上发展起来的工作有文[10]等，它们对于原有系统的描述能力及应用范围进行了不同程度的改善，但是对于继承的形式刻画依然有其局限性。

采用操作语义方法的有书[1]中第三章和文[11]等工作。书[1]中第三章对面向对象计算作了较为系统的形式描述，事实上给出了面向对象语言操作语义研究的一个雏形。P. America 在文[11]中采用结构化操作语义方法研究并行面向对象语言的操作语义。

对并行面向对象语言形式语义研究具有积极影响的还有 π-演算，它是由 R. Milner[12]等人于八十年代后期至九十年代初创立并逐步发展而形成的，旨在为含有通信结构的并发系统建立理论基础。事实上，π-演算扩展了 CCS 理论，其表达能力比 CCS 和 λ-演算都要强。D. Walker[13]采用 π-演算刻画了 POOL（一个并行面向对象语言）的语义。因其自身的特点，π-演算对于对象以及对象间通信的语义解释很自然，而对于类以及继承的语义解释过于动态化。

五、继承的一个数学模型

鉴于这一领域的现状，文[14]提出了继承的一个数学模型。引入图示类别型构及其延拓等概念，继承被刻画为图示类别型构上的延拓函数。通过这一数学模型，各种继承均能得到形式描述，继承的基本特点能被揭示出来。基于继承的这一数学模型，该文还给出了对象式语言程序的代数语义模型。这一工作发展了类别代数理论，为面向对象语言，特别是继承和类型机制的改进和完善打下了一定的理论基础。另外，本文作者在文[15]中刻画了 Eiffel 语言之一个抽象子语言的指称语义，旨在从语义级揭示特征重命名带来的利弊。以下介绍继承的这一数学模型的基本术语和基本思想。

5.1 基本术语

继承的这一数学模型采用有向无环图、图示类别型构、类别上的操作符、以及图示类别型构的延拓等概念。

一个有向无环图（directed acyclic graph）是一个二元组 (S,R)，其中 S 为有限集，R 为 S 上的一个二元关系，且 $\bar{R}$（R 的自反、传递闭包）是反对称的，且不存在 $s \in S$ 使 sRs。其中 $\bar{R} \equiv \{(s,t) \mid \exists\, n \geqslant 1. \exists\, s_1, \cdots, s_n. s_1 = s \wedge s_n = t \wedge \forall\, 1 \leqslant i < n, s_i R s_{i+1}\}$。在一个有向无环图 (S,R) 中，设 $s \neq t$，且 $s\bar{R}t$，则从 s 到 t 的一条路径是下列集合中的一个元素：

$$Paths(s,t) \equiv \{\gamma \in S^* \mid \gamma(1) = s \wedge \gamma(len\gamma) = t \wedge \forall\, 1 \leqslant i < len\gamma, \gamma(i)\ R\ \gamma(i+1)\}.$$

其中 $len\gamma$ 代表序列 γ 的长度。

定义 4 （图示类别型构，GSS） 一个图示类别型构（graphic sorted signature）是一个三元组 $(S,R,\sum)$，其中 (S,R) 是有向无环图，$\sum$ 是形如 $\{\sum_{s,w,c} \mid s,c \in S, w \in S^*\}$ 的 $(S \times S^*) \times S$—类别化集合，

注：(1) 设 $(S,R,\sum)$ 是一个 GSS，则 $(S,\sum)$ 可看作一个 MSS，因为一个 $(S \times S^*) \times S$—类别化集合

维普资讯 http://www.cqvip.com

可看作一个$S^* \times S$—类别化集合。(2)GSS也用来表示图示类别型构的全体。

定义5 (图示类别型构中某一类别上的操作符) 设$(S,R,\sum)$是一个图示类别型构，且$s \in S$，如果存在$w \in S^*$和$c \in S$使得$\sigma \in \sum_{s,w,c}$，则称σ为$(S,R,\sum)$中的类别s上的操作符，简称为类别s上的操作符。本文用$\sum_s$表示$(S,R,\sum)$中的类别s上的操作符全体，即：$\sum_s = \{\sigma \mid \exists\, w \in S^*, \exists\, c \in S, \sigma \in \sum_{s,w,c}\}$。

定义6 (图示类别型构的延拓) 设$(S,R,\sum)$和$(S,R,\sum')$是具有同一有向无环图结构的两个图示类别型构，如果$\sum_s \subseteq \sum'_s$对一切的$s \in S$均成立，则称$(S,R,\sum')$是$(S,R,\sum)$的延拓(Extension)；如果$\sum_{s,w,c} \subseteq \sum'_{s,w,c}$对一切的$w \in S^*, s,c \in S$，则称$(S,R,\sum')$是$(S,R,\sum)$的严格延拓(Strict Extension)。

5.2 基本思想

继承可区分为单继承和多继承，视一个类是否只能有一个直接基类而定(这可以从语法结构中识别出来)。多继承通常被划分为三种：图式继承，线性继承，和树式继承。关于各种继承的非形式语义可参见有关相应语言的文献，简单地说，继承使得一个类通过继承已有类还可以拥有除了该类的说明特征以外的其它特征。这样，一个类的特征包括两部分：在该类中说明的(最新引入或重说明)和直接继承的。

在纯的面向对象语言中，程序通常是一组结构化的类，从程序正文中可以抽取下列信息：程序中涉及的类名集、类间的直接继承关系、每个类中说明(含重说明)的特征名及其型构。这些信息分别由下列集合表示：一个类别集S，S上的一个有向无环图结构R，一个形如$\{\sum_{s,w,c} \mid s,c \in S, w \in S^*\}$的$(S \times S^*) \times S$—类别化集合。这一三元组$(S,R,\sum)$称为相应程序的代数结构，$(S,R)$称为相应程序的继承图。本文引入图示类别型构这一概念就是用来表示面向对象语言中程序的代数结构。从面向对象语言的程序正文中抽取其代数结构的形式方法可参见文[14]。直观看，如果$(S,R,\sum)$是某一程序的代数结构，则$s \in S$表示s是相应程序中涉及的类名(基本类或说明类)，$s\,R\,t$表示相应程序中类s是类t的直接衍类，$\sigma \in \sum_{s,w,c}$表示σ在类s中被说明成型构为(w,c)的特征。继承机制使得基于层次结构R的不同类共享特征，从而一个类还拥有除了该类的说明特征以外的其它特征，因此$\sum$有一个延拓$\sum^*$使得：$\sigma \in \sum^*_{s,w,c}$表示类$s$具有名为$\sigma$型构为$(w,c)$的特征。显然，$\sum^*$是按某一确定的方法由$(S,R,\sum)$构造而得，这一构造方法代表了继承之所在，并且可由图示类别集上的一个形如$Ext: GSS \to GSS$映射来表示，该映射称为图示类别型构集上的一个延拓映射。因此，本文用图示类别型构集上的延拓映射来表示继承的形式语义。值得注意的是：对于不同的语言，其程序的代数结构通常具有不同的性质(主要由于解决特征名冲突的方法和特征重说明的语法约束不同)，且表示继承之形式语义的延拓映射随着给定语言之继承方式的不同而不同。以图式继承为例，下面定义图示类别型构集上的延拓映射GExt来表示图式继承的形式语义。

定义7 (延拓映射GExt) 映射$GExt: GSS \to GSS$定义如下：$GExt((S,R,\sum)) \equiv (S,R,\sum^G)$

其中，$\sum^G_{s,w,c} = \sum_{s,w,c} \cup \{\sigma \mid \sigma \notin \sum_s \wedge (\exists\, t.\ sRt \wedge \sigma \in \sum^G_{t,w,c})\}$。对于任意的$s,c \in S, w \in S^*$。

注：$(S \times S^*) \times S$—类别化集合$\sum^G$的良定性由(S,R)的有限偏序性得到保证，所以映射GExt也是良定的。直观上看，$\sum^G_{s,w,c}$表示类s所拥有的型构为(w,c)的特征名集，其中包括在类s中说明的以及从基类中继承来的。另外，按照图示类别型构中某一类别上的操作符之定义，$\sum_s$表示在类s中说明的特征名集，$\sum^G_s$表示类s所拥有的(包含在类s中说明的以及直接继承来的)的特征名集。可以证明：$\sum_s \subseteq \sum^G_s$对一切的$s \in S$均成立，因此$\sum^G$是$\sum$的延拓，上述定义中采用的映射名GExt隐含这一映射是延拓映射(Extension function)，且这一映射是为了刻画图式继承(Graph-Oriented Inheritance)的，"G"取自图式继承的第一个英文字母。

为了深入研究面向对象语言中程序之代数结构的性质，文[14]引入图示类别型构的单调性、协变单调性、反变单调性，强正规性、和正规性等概念，证明了强正规图示类别型构一定是正规的，阐述了图式继承语言中程序之代数结构通常是强正规的，从而也是正规的。程序之代数结构的正规性值得注意，它对于解决继承特征的名冲突和确定特征的实现版本是非常本质的。正规性的定义如下：

维普资讯 http://www.cqvip.com

定义 8 （正规性） 设$(S,R,\sum)$是图示类别型构，则称$(S,R,\sum)$是正规的，当且仅当下列正规性条件(Normality condition)成立：对任给的σ和s，如果$\{t' sRt \wedge \sigma \in \sum_{t'}\}$非空则有最小元。

正规性类似于有序类别型构的正则性(Regularity)，直观地看，有序类别代数理论将类别间的序关系解释为行为上保持一致的子类型关系，对于多态操作符的语义解释须保持一致，有序类别型构的正则性就是为了保证这样的一致性所需的语法约束；然而在面向对象语言中，类间的继承关系无需是行为上保持一致的子类型关系，因而对于多态操作符的语义解释无须保持一致，但是，由于继承的引入(再加上动态定连)，对于多态操作符的语义解释可能会出现歧义性，为了避免这样的歧义性，图示类别型构的正规性不失为一种较自然的语法约束，事实上，图式继承语言一般通过语法约束使得程序的代数结构具有强正规性，从而具有正规性。

文[14]还证明了：在代数结构的强正规性条件下，代数结构的单调性、协变单调性和反变单调性均在延拓映射 GExt 下保持不变。在此基础上，研究了基于类间继承关系之子类型关系，阐明了代数结构的单调性和正规性对于语言之类型系统的重要性，分别揭示了图式继承、线性继承、单继承对类型机制的影响。

六、结束语

上述这些工作对于深入理解面向对象语言和推动面向对象思想的发展有着积极的意义。然而，面向对象语言的形式语义研究有待深入。进一步的工作包括：(1)面向对象语言的公理语义研究，这对于发展程序正确性验证的形式化技术和程式之实现算法的自动生成技术很有必要，这一工作将涉及动态定连和继承程式的逻辑刻画这一难点；(2)在类说明中含不变式之面向对象语言的形式语义研究，特别是其代数语义的研究，这一工作的难点是直接基类中不变式的相容性问题以及程式重定义的语义约束问题；(3)并行面向对象语言形式语义的研究，尤其是操作语义的研究，其难点是对并行性与继承性间关联的形式刻画。目前，面向对象语言的形式语义研究可谓是方兴未艾，我们期待着这一领域得到进一步的研究和发展。

参考文献

[1]徐家福，王志坚，翟成祥，对象式程序设计语言。南京大学出版社，1992。

[2]P. Wegner, The Object-Oriented Classification Paradigm, Research Directions in Object-Oriented Programming, B. Shriver and P. Wegner edited, 1987

[3]M. Wolczko, Semantics of Smalltalk-80, Proc. of ECOOP'87, LNCS 276, 1987

[4]M. Wolczko, Semantics of Object-Oriented Languages, Ph. D. Thesis, the University of Manchester, 1988

[5]W. Cook and J. Palsberg, A Denotational Semantics of Inheritance and its Correctness, SIGPLAN Notice, Vol. 24, №. 10, 1989

[6]A. V. Hense, Wrapper semantics of an object-oriented programming languages with state, TACS'91 Proc. pp. 548-568

[7]J. A. Goguen and J. Meseguer, Order-Sorted Algebra I ; equation deduction for multiple inheritance, overloading, exception and partial operations. Theor. Comp. Science, Vol. 105, №. 2, 1992

[8]K. Futatsugi, et al., Principle of OBJ2, Proc. 12th ACM Symp. Prin. Program. Lang., 1985, pp. 52-66

[9]L. Cardelli and P. Wegner, On Understanding Types, Data Abstraction and Polymorphism, Computing Surveys, Vol. 17, №. 4, pp. 471-522, 1985

[10]L. Cardelli et al., An extension of system F with subtyping, Proc. Intl. Conf. TACS'91, LNCS 526, pp. 750-770

[11]P. America and J. Rutton, A Paralleled Object-Oriented Language: Design and Semantic Foundation, JCSS, Vol. 39, pp. 343-375. 1989

[12]R. Milner, A Calculus of Mobile Processes, I. Information and Computation 100, pp. 1-40, 1992

[13]D. Walker, π-Calculus Semantics of Object-Oriented Programming Languages, Proc. Intl. Conf. TACS'91. pp. 532-547, 1991

[14]瞿裕忠，对象式程序设计语言的形式语义研究。博士论文，南京大学，1994

[15]Qu Yuzhong, Wang Zhijian, Xu jiafu, Denotational Semantics of a Simple Model of Eiffel. J. Comput. Sci. Tech. accepted