国防科技大学学报
JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY
第16卷第4期 1994年12月 Vol. 16 No. 4

逻辑和面向对象范例集成的研究与实现*

周 立 吴泉源

(国防科技大学计算机系 长沙 410073)

摘 要 提出了结合逻辑和面向对象范例的一种新方法，其特点是在统一的逻辑语义和对象语义解释的基础上，同时支持逻辑语言的描述性特征，和面向对象语言的结构化、信息隐藏、继承等特性；并在逻辑语言的 Herbrand 解释基础上，探讨了逻辑对象的语义解释；最后给出了我们根据这种方法设计并实现的面向对象逻辑程序设计系统所提供的对象操作原语。

关键词 逻辑；对象；方法；继承；消息

分类号 TP18

当前，知识处理问题趋于复杂、庞大、常常包括多方面的知识，要求有多种问题描述和问题求解方法，走向合并已成为知识程序设计语言研究的趋势。逻辑的和面向对象的两种程序设计范例的集成已成为一个引入注目的研究领域，特别是模块化、知识构成、信息隐藏和共享概念被认为是扩展逻辑程序设计语言应用领域的基础，使其向大型的、复杂的、基于知识的系统应用发展。

目前在国际上基本上采用下面三种方式：

(1) 扩展模式。将面向对象程序设计语言中的典型概念引入到逻辑程序设计范例中，而保留后者的基本结构和机制。如 Mandala，Esp 等系统都运用了这种方法。

(2) 集成模式。保持逻辑和面向对象范例各自的独立性，仅在两者之间定义一个接口。LOOPS 系统就基于这一思想。

(3) 语言级的合成。首先寻找两种范例共同的语义基础，建立一个一致的形式逻辑体系，从而构造一种多功能的同时具有这两种程序设计风格的新语言，使其满足继承、消息发送等由于对象操作而引起的环境的动态切换。

我们提出了集成逻辑与面对对象范例的一种新方法，它是在统一的逻辑语义和对象语义解释的基础上，同时支持逻辑语言的描述性特征，和面向对象语言的结构化、信息隐藏、继承等特性，用一种自然而有效的方式开发了这两种语言的优点。并在逻辑语言的 Herbrand 解释的基础上，探讨了逻辑对象的语义解释。根据这种方法，我们设计并实现了一个面向对象逻辑程序设计系统，它是面向对象技术和逻辑程序设计语言的结合，符

* 1994年5月24日收稿

et

合结构程序设计和软件重用的要求，适合于进行大型知识系统的开发。

1 基本概念

1.1 类

面向对象范例的核心是“模块＝类”。模块是控制名范围的实现层结构，类是管理对象过程语义的程序抽象。我们集成逻辑和面向对象范例的总体原则是利用低层模块机制封装对象和类，即模块提供了对象或类代码周围的屏障，语言解释器则支持穿越模块边界的名约束功能。

面向对象逻辑程序由对象组成，每个对象都包含了一组独立的相关 PROLOG 子句。类是同一类对象建立的模板，类的语法描述如下所述：

```
〈class〉:: =: -class ( 〈name〉, 〈super_interiace〉 [, 〈meta_interface〉]).
                〈class_methods〉
                〈instance_methods〉
                〈instance_variables〉
              end_class.
〈super_interface〉:: = []
〈super_interface〉:: = [ 〈name〉 | 〈super_interface〉]
〈meta_interface〉:: = 〈name〉
〈class_methods〉:: =¥class  mathods.
                        〈clause〉*
〈instance_methods〉:: =¥instance_methods.
                        〈clause〉*
〈instance_variables〉:: =¥instance_variables.
                        〈clause〉*
```

其中，〈name〉为原子，表示类名；〈clause〉* 表示 PROLOG 子句集（可为空）；super_interface 指明类/父类关系；meta_interface 表示类/元类关系。

1.2 消息

独立的 PROLOG 程序模块（类）之间可通过发送消息进行通讯，向某一对象发送一条消息可解释为请求该对象证明一个目标，消息的语法形式为：

```
Object:: Goal
```

其中，中缀谓语“:: /2”可以任意嵌入在程序中，其语义为，在对象 Object 定义的环境中证明目标 Goal. 若证明成功，则该消息目标为真，否则回答“no”。

将方法表示为 Horn 子句，方法的调用解释为目标证明，具有如下优点：

①具有描述性风格，有助于建立基于知识的系统；

②一致化功能使目标参数既可作为方法操作的输入参数，又可作为输出参数，具有灵活的操作调用入口；

③方法可有多种定义，回溯用于寻找正确的求解方法；

①方法的语法与 PROLOG 程序完全一致，利于编程。

1.3 继承

在面向对象的系统中，继承关系用于共享相关对象组之间的知识。在面向对象逻辑程序设计系统中，组成对象理论的公理集分为两部分：共享部分和私有部分。其中共享部分包含方法定义，私有部分包含实例变量。对象间的继承关系用 super 表示。一个对象的继承链为一个类序列：

$$C=C_1, C_2, \cdots, C_i, \cdots, C_n$$

其中，C 为该类本身，C_n 为对象系统的根对象。对于所有 i（$1\leqslant i\leqslant n-1$），在 C_i 的类定义中存在外部接口说明 super（C_{i+1}）.

按 PROLOG 的语义，带有继承链的对象理论中，可见（或可访问）的方法包括：①如果方法是该对象类中的一个子句头，则该方法在该对象中可见；②如果方法在该对象类继承链上的直接后继类中可见，则该方法在该对象中也可见。

多重继承通过回溯支持。一个类对象可以有多个父类，这时该对象的继承链有多条，多重继承通过回溯依次在这些链上搜索完成，定义父类的顺序决定了继承链的搜索顺序。

2 逻辑语义

一个面向对象的逻辑程序是逻辑对象的一个层次结构，其中每个对象含有一组有限的确定子句。作为一种程序风格，它对逻辑程序的扩充主要在消息和继承两个方面，即一个逻辑对象可通过发送消息与其它逻辑对象通讯，并继承其父类对象的操作。从逻辑程序的观点出发，继承可看作为逻辑理论的组合，当对象 obj 接受到消息 obj send G 时，G 实际上是在 obj 以及其继承链上所有类对象定义的公理集上求解。另外，消息可解释为目标，向一个对象发送消息即为用与对象相关的确定子句求解某个目标。因此，基于逻辑程序的过程语义，面向对象逻辑程序的过程语义可用如下定义描述。

在下面的描述中，我们采用如下约定：

P：由逻辑对象组成的面向对象的逻辑程序；A：原子公式；g：目标；G：目标的合取式。

|obj|={C|C 是对象 obj 中的子句}

objects(P)={obj 是 P 中定义的对象}

O_i, O_j, O_m 为属于 objects(P)的对象

O_iO_j：为从对象 O_i 到 O_j 的继承链

$\{G, O_iO_j\}_1$：在 O_iO_j 继承链上导出目标的 mgu 取代为 1

定义 1（归结）

程序 P 自顶到底导出目标 G 的过程，由以下目标子句序列组成：

$$G_0=G, G_1, G_2, \cdots$$

目标的求解区域序列：

$$O_{01}O_{02}, O_{11}O_{12}, O_{21}O_{22}, \cdots$$

以及最一般的一致代取代 mgu 序列：

$$\theta_0, \theta_1, \theta_2, \cdots$$

其中，O_iO_j 表示对象在 O_1 的继承链上从 O_i 到 O_j 的组合公理集，G_{i+1} 为 G_i 在环境 O_{i1}

O_{i2}上用 $mgu\theta_1$ 导出。若导出过程有限，且最终目标为空子句，即 $G_n=\square$，称该导出是成功的，此时记 $P\mid=\overset{O_{01}}{k}G(k)=\theta_0\theta_1\cdots\theta_n$).

这里，导出规则为：

(1)→{true,_}$_k$

(2){g,O_iO_j}$_k$,{Gk,O_iO_j)$_i$→{(g,G),O_iO_j}$_{kl}$

(3){Gk,O_iO_j}i→{A,O_iO_j}$_{kl}$，其中，A 是原子公式，求解区域|O_iO_j|中存在子句 A′：—G，使 A 与 A′可一致化，且∃ k=mgu(A,A′).

(4){A,O_iO_n}$_k$→{A,O_iO_j}$_k$，其中 O_j 不是根对象，且 O_n 是 O_j 的父类。

(5){A,O_iO_i}$_l$→{O_i send A,_}$_i$

逻辑对象既可隐式地（通过继承）又可显式地（通过消息）访问“外部世界”，从这个意义上说，单个逻辑对象是一个开放的逻辑公理集。这种开放逻辑对象不能用简单的最小 Herbrand 模型来解释，这里，我们提出“可见条件 Herbrand 模型 VCM(Visible Conditional Herbrand Model)”的概念，并用它来描述逻辑对象的描述性语义。下面为简单起见，假设程序 P 中所有对象都只含基础子句。

定义 2　（逻辑对象的可见条件 Herbrand 模型 VCM）

给定程序 P 中的一个对象 O，对 O 的每个继承链 $O_0=O$，O_1，…，O_{root}，其中 O_{i+1} 是 O_i 的父类，令 O′包含 O 中所有可见子句，即 $|O'|=|O_0|\cup|O_1|\cup\cdots\cup|O_{root}|$，Msg(O′)为出现在 O′中消息的集合，H 是 Msg(O′)的子集，则可见条件 Herbrand 模型 VCM $(O)^H$ 为 O″的最小 Herbrand 模型，其中 O″为对 O′进行如下操作所得：

(1) 删除所有属于 H 的消息目标

(2) 如果存在子句体中含有消息 obj send A，且该消息不属于 H，则删除该子句。

定义 3　（对象程序的可见条件 Herbrand 模型）

设程序 P 包含有对象 O_1，O_2，…，O_n，这些对象的可见条件 Herbrand 模型分别为 $VCM(O_1)$，$VCM(O_2)$，…，$VCM(O_n)$，则 P 的可见条件 Herbrand 模型 VCM(P)可定义为，对所有的 i，$1\leqslant i\leqslant n$，有：

(1)如果 $g\in VCM(O_i)^H$，则 O_i send $g\in VCM(P)$；

(2) 如果 $H\subset VCM(P)$，且 $g\in VCM(O_i)^H$)，则 $O_i$send $g\in VCM(P)$.

定义 4　（对象 O_i 求解环境中 VCM 模型下的真值关系）

按照程序 P 的 VCM 模型定义，有如下真值关系成立：

(1)$VCM(P)\models^{O_i}$true；(2) $VCM(P)\models^{O_i}(g,G)$iff$VCM(P)\models^{O_i}g$ 并且 $VCM(P)\models^{O_i}G$；(3)$VCM(P)\models^{O_i}A$ iff $A\in VCM(O_i)^{(1)}$或 $A\in VCM(O_i)^H$ 并且 $VCM(P)\models^{O'}H$；(4)VCM $(P)\models^{O_i}O$ send A iff $VCM(P)\models OA$

显然，这些真值关系与前述导出规则完全对应。

3　对象操作原语

我们的面向对象逻辑程序设计系统提供的对象操作原语主要包括：消息发送原语，对象的动态创建原语，类创建原语，实例创建原语，类/父类关系原语和实例/类关系原语

等。

(1) 消息发送原语 send/2，消息用于实现各封装对象之间的交互。一个对象 O_1 向另一个对象 O_2 发送消息，请求 O_2 完成目标 Goal 的求解，该消息目标的语法为：

O_2 send Goal

系统执行这个目标，首先将 Goal 提交给 O_2 所属的类，并沿该类的 super 链直到找到求解 Goal 的方法，方法的子目标求解在 O_2 及其继承链上进行求解。

(2) 对象的动态创建原语 make_object/6，类和实例可以用文件静态定义，本系统还提供了一个动态创建对象原语，在程序运行过程中，根据需要动态创建类和实例，或对方法进行重新定义。对象的动态创建原语为：

make_object (Object, Isa, Meta, Super, Method, Instance_Vari)

其中，Object 表示要创建的对象名；Isa 表示对象所属的类；Meta 表示对象的元类；Super 表示对象的父类表；Method 表示对象中的方法表，是一个子句集；Instance_Vari 表示对象中的实例变量表，是一个子句集。

(3) 类创建原语 new_class/5 和实例创建原语 new/2，new_class/5 和 new/2 都基于对象创建原语，它们通过向给定类发送消息调用，语法分别为：

C_1 send new_class (Oid, Meta, Class_Method, Inst_Method, Inst_Vari)

C_2 send new (Oid, Inst_Vari)

其中，方法 new_class/5 创建一个类，它包含两个对象 Oid 及其“工厂”对象 Oid_factory，Oid 的父类为 C_1，Isa 指向 Oid_factory，Oid_factory 的父类为 C_1_factory，Isa 指向 object，其余参量同静态类定义。new/2 创建一个实例对象 Oid，其 Isa 指向 C_2.

(4) 父类查找原语 super (Class, Super)，该原语用于寻找 Class 的父类。若 Super 为变量，则特例化为 Class 的父类，若 Super 为常量或已特例化的变量，则判断 Super 是否为 Class 的父类。

(5) 实例/类关系原语 isa (Object, Class)，此原语用于寻找实例 Object 的所属类 Class.

4 结　论

本文提出了一种新颖的集成逻辑和面向对象风格的方法，它具有如下主要特点：

(1) 采用结构化语法形式，支持大型知识系统的模块化设计和结构化开发；

(2) 完美地结合了逻辑程序描述性语义的优点和面向对象范例的典型概念；

(3) 对两种程序设计风格探讨了统一的逻辑语义，这也有助于研究面向对象范例本身的语义基础。

根据这种方法设计实现的面向对象逻辑程序设计系统，兼备了逻辑程序和面向对象风格两者的优点，支持大型知识处理软件系统的结构化，信息隐藏和数据抽象的概念，和知识单元的说明性语义。从整个系统看，具有面向对象的程序设计风格，而对于每个类而言，其内部描述保持了逻辑的描述性风格。

另外，该系统还支持标准 PROLOG 程序和模块化 PROLOG 程序的开发，将标准 PROLOG 程序看作为单模块程序，而将模块化 PROLOG 程序看作为无继承层次关系的

对象系统，从而保持了软件的向上兼容性。

参 考 文 献

1 Zhou Li, Wu Quanyuan. GKD-OOPS: A New Object-Oriented Logic Programming System. The 3rd International Conference on Systems Integration, San Paulo, Brazil, IEEE Computer Society Press, Jul. 1994

2 P Mello, A Natali. Objects as Communicating Prolog Units. ECOOP' 87, Bigre+Globule 54, Jul. 1987

3 C Zaniolo. Object-Oriented Programming in Prolog. Proc. of International Symposium of Logic Programming, Atlantic City, IEEE, 1984

4 K Fukunage, S Hirose. An Experience with A Prolog-Based Object-Oriented Language. OOPSLA'86, ACM SIGPLAN Notes 21, NOV. 1986

The Research and Implementation of Combining Logic with Object-Oriented Paradigm

Zhou Li Wu Quanyuan

(Department of Computer, NUDT, Changsha, 410073)

Abstract

In this paper, we propose a new method to merge the logic with the object oriented pardigm. We intend to introduce the typical concepts of object oriented systems in the logic paradigm, without losing its advantages as a declarative language. An extensive Herbrand interpretation has been given to interpret thd logic semantics and the object oriented semantics uniformly. Based on this method, we design and implement an object oriented logic system. In the end, the paper gives the object primitives of the system.

Key words logic; object; method; inheritance; message