国防科技大学学报
JOURNAL OF NATIONAL UNIVERSITY OF DEFENSE TECHNOLOGY
第14卷第4期 1992年12月 Vol. 14 No. 4

Prolog 对象系统 GKD－POS/SUN 的设计与实现*

严静东 金 芝 吴泉源 邓铁清

（电子计算机系）

摘 要 GKD－POS/SUN 是在模块化 Prolog 基础上设计并实现的一个面向对象 Prolog 解释器。它同时引入了模块化、模块间的通讯、切换、类、方法、继承、消息发送及接收等概念，并提供对这些概念的直接支持。本文描述了 GKD－POS/SUN 的语言规范、系统组织与设计、主要实现技术和相关对象操作原语。

关键词 对象，逻辑，类，继承，消息，方法

分类号 TP314

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主要趋势。逻辑的和面向对象的两种程序设计范例的组合已成为一个引人注目的研究领域，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。

目前国际上在这一方面已作了一些研究，但各自的观点和方法却不尽相同，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一些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）语义级的合成。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系统，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。

本文设计并实现了一个面向对象 Prolog 语言解释器 GKD－POS/SUN (GKD－Prolog Object System)，它在 GKD－Prolog /SUN 中引入模块化、模块间的通讯、切换、类、方法、继承、消息发送及接收等概念，并提供对这些概念的直接支持。新的面向对象 Prolog 解释系统不仅要保留原有系统的算法思想，使其适应单一类的特殊问题，而且要扩充算法功能，在解释器一级支持动态名约束，对象的继承以及用动态环境的目标求解解释消息发送。该系统的语义基础是 SCKE 模型[4]。

* 1992年1月21日收稿

1 GKD—POS/SUN 语言规范

1.1 对象表示和标识

GKD—POS/SUN 系统中的每一个对象都是一个封闭的知识实体，不允许直接访问对象的内部，只允许向对象发送消息。对象名由标识符表示，在创建对象时给对象赋一个标识符，若未确切指明对象名，则由系统赋予该对象一个唯一的标识符。对象的状态是可以改变的，但其标识符不变，因此对象的状态变化对对象标识符拥有者是可见的。

实例变量用 Prolog 子句表示，其中子句头谓词表示实例变量，谓词参数表示变量的值，若有多个参数则表示该实例变量是向量形式。若相同子句头谓词的子句有多个，则表示该实例变量具有多重值，在问题求解的过程中通过回溯机制寻找正确的解。

1.2 类的表示

类表示为对象，相当于 Prolog 子句集。在 GKD—POS/SUN 系统中，一个 Prolog 程序可由多个类构成。类的作用有两方面：存贮该类的实例可继承的谓词；创建类实例的 factories 模块。

(1) 类定义语法。一个类包括类说明和方法描述两部分。类说明描述了类开始标志、类名、所属关系、元类关系和超类关系。类方法描述一组 Prolog 子句（包括事实，规则），分为类方法与实例方法两部分，在方法的内部表示结构中区别这两种方法。

(2) 类的内部表示。为了区分类的两个方面的作用，我们在创建类对象的时候，将一个类分装在两个虚拟的模块内：一个模块为类对象模块，包含实例方法及实例的属性；另一模块为 factory 对象模块，包含类方法及类本身要响应的消息，图 1 说明了类 student _employee 的内部结构。类 student _em _loyee 的父类为 employee. 其中 factory 模块描述了类 studentemployee 中的方法，其超类指向 employee 中的 factory 模块；类对象模块说明了由类 student _employee 创建的实例的属性及实例的方法，其超类指向 employee 中的类对象模块；实例对象 01 由类 student _employee 产生，其 isa 关系指向 employee 类中的类对象模块。

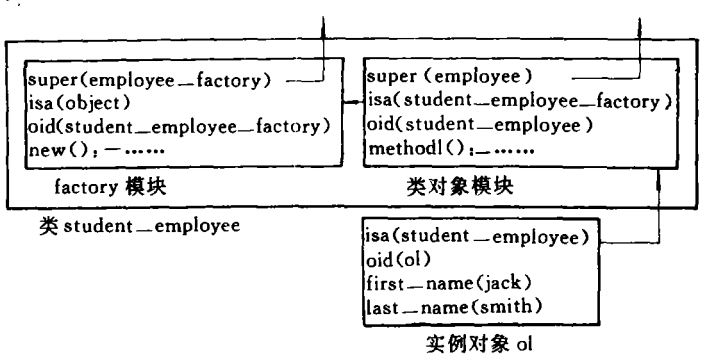

图 1 类的内部结构例

1.3 实例的创建

实例对象都是在实例运行过程中动态创建的。运行时向某个类对象发送创建实例的消息，响应该消息的方法选择模式一般都放在接收消息类的 factory 模块中，在 GKD—

POS /SUN 系统中用内部谓词 new/2 来完成实例的动态创建：

new (objectID, [instance variabie])

执行该谓词的结果在系统中生成一个实例对象，objectID 为实例名，[instance variable] 为实例变量表，说明该实例所具有的属性，是一个子句集。实例创建后，将此子句集插入由实例名标识的模块名，实例的 isa 关系指向创建该实例的类。

1.4 创建类

1.2 节中定义的类的语法是指以文件名形式静态创建类时类的描述结构，若要动态创建类，则使用 new _class/4 谓词。new _class (classname, [instance methods], [class methods], [instance variable])

其中 classname 给出要创建的类名，另外三个表是三个子句集，在类创建后插入类中，同时指出各子句所具有的属性，即属于类中哪个虚拟模块。

classname 类的父类为接收该消息且创建该类的类。

1.5 消息发送

向一个类或实例发送消息的形式为：object send goal，其中 object 是接收消息的对象，goal 是向 object 发送的消息，send 是系统提供的内部操作原语。

消息的语义为请求 object 完成目标 goal 的证明。

消息可作为一条命令发送某个对象，也可作为任何程序子句的子目标，在程序运动过程中实现若干个对象间的通讯。一般情况下，消息的接收者为某个实例。

在 GKD－POS /SUN 创建的对象系统中，根对象是常驻系统的，在系统初给化时就生成一个根对象，从而便于整个对象系统的组织与管理。

1.6 多重继承

在 GKD－POS/SUN 系统中实现的继承是多重继承，每个类可以有多个超类，其直接超类构成一个超类链表。在目标求解过程中，当通过一个继承链求解失败时，可采用深度优先的规则，通过第二级继承类回溯完成其它继承链的搜索。

2 系统组织与设计

GKD－POS/SUN 是在模块化 Prolog[5][6]的基础上建立起来的，如何完成从模块到对象的演变，包括以下三个方面：

(1) 模块实现类和对象的封装。模块化 Prolog 中的每个模块都是一个独立的知识实体，具有自己的私有属性，实现了数据和操作的封装，因此每个模块都可直接作为 GKD－POS/SUN 中的一个类或对象，每个模块内定义的子句都作为与此模块相对应的类(或对象) 中的方法。

(2) 各模块内谓词的动态调用可通过消息完成。在模块化 Prolog 中，一个模块在求解问题的过程中要使用另一模块中定义的谓词过程时，将要用的谓词过程的定义调入本模块，改变其所属模块名，然后使用；而在 GKD－POS/SUN 系统中，若两个类之间具有父类/子类关系，由于子类可共享其父类的所有方法，因此子类不必将父类中的方法装入子类，就可直接使用。若两个类之间彼此独立，则可通过调用系统提供的消息发送原语，实现环境的切换，从而共享另一类中的知识，不仅节省了内存空间，而且大大提高

了系统运行效率。

(3) 平坦的模块化结构转变成类的继承层次。在模块化 Prolog 中，模块是松散联系的，没有层次关系，相互独立，通过模块接口完成公用谓词的调入和引用；在 GKD－POS/SUN 中，将模块看作为封闭的对象后，增加了继承的概念，整个系统就成为一个具有层次的树结构（单继承）或格结构（多继承），实现了知识的静态共享。

GKD－POS/SUN 系统主要由五部分组成，即总控模块，解释执行模块，原语方法库，对象存贮器和内部数据库。总体结构如图 2 所示。

总控模块主要完成初始化对象存贮器和内部数据库，调入原语方法库及用户定义的类对象描述，接受用户的命令或问题，进行语法分析后，送给解释执行模块解释执行。

解释执行模块是系统的执行机构，是整个系统的核心，解释执行各种命令和操作，完成各种提问。该模块包括面向对象逻辑程序设计中的一致化，回溯机制，消息解释及继承机制。

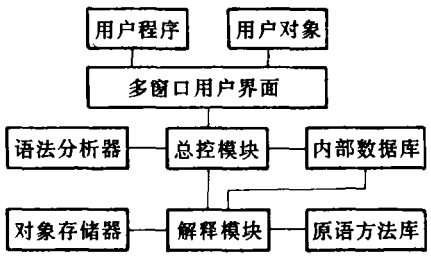

图 2 GKD－POS/SUN 总体结构

对象存贮器用来存贮对象，通过这一对象存贮器形成所有对象及其相互关系的动态环境，作为系统运行的工作存贮器供系统解释器访问和修改。

原语方法库由内部谓词库和内部函数库组成，内部谓词库除纯 Prolog 所具有的内部谓词外，还包括对象操作，消息传送等体现面向对象特征的内部谓词。内部函数库包括系统本身提供的固有方法操作，是每个程序中各个类一般都要用到的方法。原语方法库是由所有对象共享的知识，解释器在执行时若遇到原语方法，则直接调用库中相应的代码，这些基本的方法不需要用户定义，在启动系统时放在一个虚拟类模块中，由所有类直接调用。

内部数据库是在程序运行过程中动态形成的内部环境，保留了执行过程中调入系统的所有方法，所有求解环境及用到的所有信息资源，它的数据结构，存贮方式及操作方法是整个系统运行的基础，也是整个系统资源管理的核心。

3 主要实现技术

3.1 内部数据结构

在 GKD－POS/SUN 系统中，我们不仅修改了标准 Prolog 中的 hashrc，而且还增加了模块记录 modurc 和一个用于支持继承类回溯的类环境栈。

(1) hash 记录。hash 结构是所有其它数据结构建立的基础，它将内部数据库中的所有记录连接起来，构成一个整体，为程序运行提供了基本的条件。在本系统中，将所有谓词与其所有类名一起通过设计的 Hash 函数映射进行存贮，每个 hash 值标号都指向一个 hash 记录串链，也可指向空链，这些 hash 记录的 hash 值相同，这样就较好地解决了同名谓词存贮管理冲突问题及各模块内信息的隐藏。

(2) 模块记录＝类（实例）记录。这些类（实例）记录串连在一起构成了类（实例）链，组成了对象存贮器，也可建立一个 hash 结构，将这些类（实例）通过其类（实

例）名的 hash 函数值存贮，从而提高对象存贮器的查找速度。

（3）栈结构。在 GKD－POS/SUN 中，若只在某个 类中求解目标则出现一级回溯，这级回溯与非模块化 Prolog 中的回溯相同，因此在 GKD－POS/SUN 系统中保留了模块化 Prolog 中的四个栈结构。但当子目标在某个类中求解失败，要到其超类中继续求解，或沿某一继承链求解失败，而需转到另一继承链继续求解时，就需要第二级类继承回溯。为了保留目标执行过程中类环境的动态变化，以支持类回溯，我们增加一个栈结构——类环境栈，按目标求解过程中各子目标用到的类的顺序，将类及其有关属性依次放入栈中。子目标求解成功后，栈顶指针减 1，在栈头指针所示的类中继续求解下一子目标；若子目标求解失败，则根据子目标所在类的继承关系，改变栈顶内容，即改变子目标所在类的类环境为当前类的父类，继续求解，直到最后目标求解完毕。

3.2 算法描述

（1）语法分析算法。首先识别输入的文件是否是类描述文件。若是定义类的文件，则对类说明进行分析，建立一个类及其关系的结构，然后对文件中定义的类方法及实例方法进行语法分析，建立它们的内部结构。若不是类描述文件，则将其定义的内容加入 user 类中，user 类是系统根对象的子类，它的直接父类为根对象，根对象和 user 类在系统开启时由系统自动创建。该算法完成了类的识别，类的静态输入及对象存贮器的动态改变。

（2）一致化算法。完成名和参数个数相同的谓词中所有参数逐个进行一致化，其中每个谓词都有类标识，但函词，变量，常量是全局的，在一致化过程中，不比较两个 hashrc 中的类名域，保证了它们的全局名特征。

（3）归结算法。将求解的目标与其动态求解环境中的子句按深度优先搜索及最左归结法则进行一致化匹配，求得目标中的所有变量值，包括两级回溯。按照面向对象程序中继承的概念，每个目标求解过程中的各个子目标都必须从接收消息的类开始求解；求解失败，首先使用其元类中的知识，然后再使用父类中的信息。

动态求解环境指的是在程序运行过程中动态确定的求解目标的模块环境，求解过程中所能运用的知识就是这一模块中的所有子句，与其它未使用的模块无关，但在程序执行过程中可以通过继承关系和系统提供的原语共享其它类中的信息。

4 对象操作原语

GKD－POS/SUN 系统向用户提供了 4 个对象操作原语，可直接使用这些原语完成对类的操作。

（1）消息发送原语 send/2. 在 GKD－POS/SUN 系统中，各封装对象之间的交互通过消息发送完成，一个对象向另一个对象发送消息，请求另一个对象完成某个目标的求解。

（2）对象的动态创建原语 make _ object/6. 类或实例可以用文件定义，在文件内定义类/实例关系，类/类关系及类方法，实例方法，实例变量等。文件定义的关系是静态的，GKD－POS/SUN 提供了一个动态创建对象的原语，在程序运行过程中，根据需要动态创建类和实例，且可对类中的方法进行重新定义。对象的动态创建原语为：

make _ object（Object，Isa，Meta，Super，Classmethod，Instancemethod）

其中：object－要创建的对象名；Isa－对象所属的类；Meta－对象的元类；Super－

对象的父类表；Classmethod－对象中的类方法表，是一个子句集；Instancemethod－对象中的实例方法表，是一个子句集。

类创建原语 new _ class/4 与实例创建原语 mew/2 都基于对象创建原语之上，在类创建原语中，将创建新类时所处的类环境作为新类的超类和元类，而其 isa 关系为空；在创建实例的原语中，将创建实例时所处的类环境作为该实例的 isa 关系和元类，而实例的超类为空。

(3) 父类查找原语 super (Object, Super)。此原语用于寻找 Object 类的父类，因此 Object 必需是常量或已特例化的变量，否则运行出错。

(4) 查找实例所属类原语 isa (Oid, Object)。此原语用于寻找实例 Oid 所属的类，Oid 必需是常量或已特例化的变量，否则运行出错。

5 结束语

GKD－POS/SUN 在逻辑语义的基础上集成了逻辑型和面向对象程序设计范例，从而兼备了两种程序设计的优点，支持了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。从整个系统看，具有面向对象的程序设计风格，而就类内部而言，其内部描述保持了逻辑的描述性风格。目前，GKD－POS/SUN 原型系统已在 SUN 工作站上用 C 语言实现，其实用化工作正在进行之中。

参 考 文 献

1 Fukunaga K. and Hirose S. An Experienee with a Prolog－based Object －Oriented Language. Proc OOPSLA'86, Portland, 1986

2 Gallair H. Merging Object and Logic Programmign: Relational Semantics. Proc AAAI'86, Philadelphia, 1986

3 金芝，胡守仁. SCKE：集成逻辑与面向对象范例的一种新模型. 国防科技大学学报，1992，(3)

4 严静东，金芝，吴泉源. GKD－PROLOG/SUN 模块系统的设计与实现. 计算机应用研究，1992，(3)

5 严静东. 多窗口面向对象 PROLOG 解释系统 GKD－POS/SUN 的设计与实现. 国防科技大学硕士学位论文，1992

Design and Implementation of a Prolog Object System GKD－POS/SUN

Yan Jingdon Jin Zhi Wu Quanyuan Deng Tieqing

(Department of Computer Seience)

Abstract

An object－oriented Prolog interpreter GKD－POS/SUN is designed and implemented based on a modular prolog system. Some concepts such as module, communication between modules, exchanging, class, method, inheritance, message sending and receiving are introduced in and directly supported by the interpreter. This paper describes the syntax, system organization and architecture, main implementing techniques and related object operation primitives of GKD－POS/SUN.

Key words logic, object, class, inheritance, message, method