第5卷第2期 软件学报 Vol. 5, No. 2
1994年2月 JOURNAL OF SOFTWARE Feb. 1994

面向对象系统中的特性表机制与复合对象*

姚莉 刘凤岐 汪浩

(国防科技大学计算机系,长沙 410073)

摘要 本文主要讨论特性表机制与复合对象的概念、功能及实现.在基于知识的面向对象系统中引入特性表机制与复合对象概念的目的在于增强系统的知识表达能力.文中提出的特性表机制包括了三种类型的特性说明表:类说明特性表、继承关系特性表和变量特性说明表.复合对象在系统中是通过引入一个称为"影子类"的新概念来构造和实现的.

关键词 专家系统,人工智能,知识工程,软件工具.

面向对象的程序设计(OOP)风格是开发大型复杂基于知识软件系统的行之有效的方法.这不仅是因为面向对象的程序设计方法体现了软件工程和知识工程领域近年来发展起来的新概念和新方法,具有模块化、高度的数据抽象能力、信息隐藏、软件重用以及继承性、动态性等特征,而且OOP方法也是实现软件开发、知识表示和数据库技术一体化的可行途径.

在面向对象系统中,程序是围绕着称为对象的一些实体以及在这些实体上所施加的操作来组织的,对象具有本地局部的数据和过程,语言中的所有行为(Action)都来自对象之间发送消息,消息成为对象间通讯的唯一方式,对象内部的过程提供了消息形式的局部解释.人们在求解大型复杂问题时,往往对知识进行分类,将大型复杂问题分解成多个易于求解的子问题,以减少系统设计和实现的复杂性,而面向对象的语言特征对于数据对象可被层次分类的问题特别有用,因而对象系统的软件组织形式更符合人的思维规律,能够对于大型复杂基于知识系统的开发提供更为有利的支持.

目前的对象系统研究中,有两种对象系统已被广泛采用[1]:一种是用于程序设计的面向对象语言系统,如Smalltalk、CLOS、FLAVORS等;另一类是基于知识表示和建造专家系统的对象系统,如Goldworks的对象系统、Xerox公司推出的LOOPS中的对象系统等.我们这里主要讨论基于知识的面向对象系统.

基于知识的对象系统为了正确、完整地表示知识,除具有一般面向对象范例(Paradigm)的语言特性外,还要提供许多有益于知识表示的附加机制.由于系统所建立的对象是一个基于知识的对象,其附加机制通常体现在对象内部的知识表示框架上,例如,

* 本文1991年6月24日收到,1991年9月2日定稿

本课题由863计划支持.作者**姚莉**,女,28岁,博士生,主要研究领域为计算机软件,人工智能.**刘凤岐**,55岁,教授,主要研究领域为计算机软件,人工智能.**汪浩**,63岁,教授,博士导师,主要研究领域为系统工程与模糊数学.

本文通讯联系人:姚莉,长沙 410073,国防科技大学计算机系

LOOPS 系统中实例变量的值部分可含有面向存取(Access oriented)的激活值机制[2,3,5],当系统运行时附加机制将产生附加的计算工作,以便于正确地表示知识和利用知识进行推理.对于专家知识系统的建造,就需要一些特定的元知识表示机制以提供与推理规则相关的路径跟踪和表示知识正确性程度的确定因子以及复合对象描述等.在大多数的面向对象系统中,不仅通过在语法结构和操作上进行约束来说明类对象(class),也通过语义条件进行说明,这里的语义条件就是指与对象内部结构中的属性(如属性变量(attribute)、槽(slot)、实例变量(instance variable))相关的注释(如 LOOPS 系统中的 annotations、KEE 系统中的 features).如何设置这些附加机制和表示语义条件的框架,是基于知识的对象系统的重要研究内容.

本文主要介绍一个集成多范例(Paradigm)的面向对象系统 MESBT(Multi－paradigm Expert System Building Tool)中知识表示的附加机制:特性表和复合对象.系统中引入特性表和复合对象概念的目的一方面是为了完整、准确地表示知识,提高搜索效率和推理能力;另一方面是为了充分利用关于组织、结构方面的元级知识以增强系统的可描述性和可理解性.

1　MESBT 对象系统的基本概念简介

MESBT 是一个专家系统开发工具的研究原型,集成了面向对象、面向逻辑、面向存取和面向规则等多种程序设计范例(Paradigm),其基本语法和语义是在考虑了基语言 Prolog 的语言特点,为达到逻辑与面向对象的有效集成基础上定义的.MESBT 中关于对象的概念和定义与 LOOPS 系统类似,与 Smalltalk 系统略有不同.系统中共定义了三种类型的对象:元类、类和实例,它们在面向对象程序设计中起着不同的作用:

元类:元类的建立是为了构造类和管理类对象的行为,它提供了利用元级知识建造和管理类对象的方法或手段,元类可以有多级,形成一个反射塔.

类:类的建立在继承网络中是为了表示客观世界的结构,同时,也定义了一类对象的共同性质和其上的操作,即类表示了实例的状态和实例上的操作,描述了领域知识的概念层次.

实例:每个实例都属于某个特定的类,实例作为一个数据对象是对客观世界对象的具体刻划,可以被建立、修改和传递.

类和元类都是定义和操作一组对象的对象,类是元类的实例化;实例是类的实例化,在系统实现时类和元类被统一处理.元类、类和实例三个概念相结合,描述了客观世界的结构、组织和操作.三者间的逻辑关系可参看第 3 节中的图 4.

关于 MESBT 系统中知识表示和系统组织的详细资料可参阅[5,6].

2　特性表机制的实现

在面向对象的系统中现实世界模型是以不同类型的对象来统一构造的,对象内部的知识被分为描述性知识和过程性知识二部分,描述性知识主要是对象内部的状态或属性描述,

由实例变量和类变量表示;过程性知识主要指消息解释的内部实现,由方法(method)来表示.特性表机制是用来注释对象的特性、用户自定义变量的特性以及对象之间的继承关系等.设置特性表的重要原因之一是拓广对象内部描述性知识的表示框架,以便给系统开发者提供建立交互式用户界面的工具,例如,利用特性表描述咨询目标或者为解释界面提供必要的说明,也有的特性表直接指出为取得某参数如何向用户提问等等.因而,特性表机制也可看作描述性元级知识的使用.

特性表机制使所建立起来的专家系统具有良好的说明性语义,而且系统通过用户定义的特性表对被注释的对象有更进一步的理解,有助于系统进行程序的动态检查、变量存取的条件约束以及多类继承关系的有效实现,提高相应的推理和搜索效率.

MESBT 中共定义了三种特性表 Cpropertylist、Spropertylist、Vpropertylist,分别用来说明类、继承关系和变量(包括类变量和实例变量).系统提供了这些特性表的访问机制,程序设计者通过被说明的对象进行特性表的访问,利用特性表中的特性关键字进行特性值的存取.除 Spropertylist 是对关系的特性进行说明外,其它两个特性表可能含有多种特性,其特性表是一个任选的注释表,用户根据自己的需要决定是否使用以及所用特性的个数.特性表的增加与删除原则上并不影响原程序的执行,当系统执行过程中需要访问状态变量的某些特性,如系统检查变量的数据类型(TYPE—checking),而变量的特性表无此特性时,系统将自动向用户提问相应变量的特性.因此,特性表机制是程序设计者组建知识系统的一个灵活方便的工具.以下分别对这三种特性表作简要说明.

2.1 类说明特性表 Cpropertylist

Cpropertylist 是用于对类的性质进行说明或注释的特性表.

面向对象程序设计利用抽象化、模块化的结构程序设计方法支持大型复杂基于知识系统的建造,类是一种抽象数据类型,它是作为一种重要的模拟手段构造现实世界模型的抽象层次,Cpropertylist 的主要作用是对类的内部特性进行详细说明或解释,可看作复杂信息组织的一种辅助形式,目的在于增强大型复杂系统程序的可理解性.例如,对类的建立日期、时间、建造者、类的目标及意义等、此类与其它类之间的联系等进行说明,不仅有利于系统开发者编制和调试程序,而且也有利于用户界面利用窗口向用户显示一个类的内部结构,并对类的特性作必要说明和解释.

用户采用如下形式定义 Cpropertylist:

((关键字$_1$ 特性值$_1$)(关键字$_2$ 特性值$_2$)……(关键字$_n$ 特性值$_n$)).

例如:名为 Factory(工厂)的类就可以用如下的特性表来说明.

```
[DEFCLASS
    CLASSNAME:Factory
                ((creator   Yaoli)
                (date       6—11—89)
                (time       9—02—20)
                (trans      "Factories have been built since 1949") )
          ……(略)
          ]
```

这个特性表说明类对象 Factory 是由 Yaoli 建立的,建立的日期和时间分别为1989年11月6日、9:02:20分,这个类是用于表示自1949年以来建立的工厂.

2.2 继承关系说明表 Spropertylist

Spropertylist 用来说明子类与父类之间的继承关系,将子类与父类之间的继承关系定义在子类中有助于多类继承的实现,便于对上级类(即子类的直接或间接的父类统称为上级类(Superclass))进行搜索.事实上,客观世界的继承关系可以有多种形式,在一个多类继承的类层次上并非都能够采用从抽象到具体的继承关系来说明,有必要对它们进行区别处理.例如:

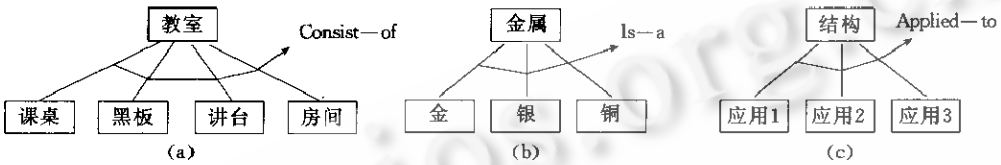


图1 各种形式的继承关系

图1(a)体现了一种整体与部分的组成关系,可用 Consist—of 来定义.

图1(b)体现了一种从抽象到具体的关系,可用 Is—a 来定义.

图1(c)体现了一种调用关系,可用 Applied—to 来定义.

Consist—of 和 Is—a 体现了两种重要的结构化概念.Consist—of 是便于实现聚合操作的抽象机制,当一个复杂问题能够分解成多个较易解决的子问题时,利用 Consist—of 继承关系组织程序便显得尤为有用;Is—a 表示了一种从抽象到具体的概括联系(Generalization),有助于根据实体之间的类似性组织程序,是继承性机制最自然的使用方式.我们认为面向对象系统应当允许用户根据组建系统的要求定义各种相应的继承关系,只要这种继承关系在类的继承网络中是可描述的,并遵循就近继承原则.这样组织程序能够最大限度地利用继承性机制,从而使对象系统的问题求解具有更大的灵活性.

一个描述继承性的例子如下:在一个管理系统中定义了 employee(雇员)类和 manager(管理者)类,管理人员也是一种雇员,因此 manager 被定义为 employee 的子类,如下所示:

```
[DEFCLASS
    CLASSNAME:manager
    METACLASS:(class,—)
    SUPERCLASS:((employee,Is—a))
    ……(略)]
```

利用 SUPERCLASS 表可将树型继承网络扩充成网状继承网络,使对象可从多个上级类继承结构说明和实例行为,实现多类继承机制,而且利用 SUPERCLASS 表中的 Spropertylist 可对继承关系进行说明,使用户能够有选择地继承其上级类的性质,从而使多类继承的实现更为有效.

2.3 变量特性说明表 Vpropertylist

Vpropertylist 是用来说明类变量或实例变量的特性表,目的在于描述一个对象中状态变量的值与对象本身的关系,或者,将附加的描述和说明系在数据之上,用以引导对这些数据进行解释,同时,也有助于系统对变量进行动态检查(如数据类型、取值范围等)以及对变

量值存取施加相应的条件约束等). Vpropertylist 除了(关键字 特性值)形式外,特性值部分也可能出现激活值[6,8]的形式. 当激活值使用时,由面向存取的机制自动处理.

系统内部定义了关键字表,对于此表中特性的定义,系统可以自动进行识别处理. 用户可以自定义所需要的关键字表以外的特性,对于关键字表内的特性以及用户自定义的特性,用户都可以存取和访问.

系统内部定义的关键字表包括如下一些内容:

```
DOC              /*如何解释这个变量
PROMPT           /*指出为取得该变量如何向用户进行提问
VALUELIMIT       /*指出变量的合法值范围
UPPERLIMIT       /*指出变量合法值的上界
LOWERLIMIT       /*指出变量合法值的下界
UNIT             /*指出变量值的单位
DATATYPE         /*该变量的数据类型
HISTORYRECORD    /*值的历史记录
CF               /*该值的确定因子
其它
```

例如,定义一个 employee(雇员)类,其实例变量可按如下形式定义:

```
[DEFCLASS
    CLASSNAME:employee
     ……(略)
    INSTANCEVARIABLES:
     age  54
      ((DATATYPE INTEGER)
       (VALVELIMIT (18 60) )
       (UNIT year)
       (PROMPT "How old is Mr. X ?") )
      profession engineer
      workingTime 25
      ((DATATYPE INTEGER)
       (UNIT year)
       (PROMPT "How long has Mr. X worked ?") )
      salary 180
      ((UPPERLIMIT 600 )
       (UNIT ¥ )
       (DOC "salary paid monthly") )
    METHOD
    ……(略)
       ]
```

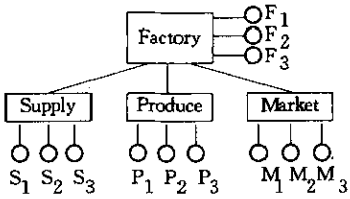
3 复合对象与影子类

3.1 复合对象的概念

在 MESBT 系统中,存在三种类型的对象:元类、类、实例,其中元类定义了消息的处理机制以及对类进行建立、修改等设施. 元类概念的使用使得对象系统具有良好的可扩充性和自适应的能力,例如,扩充对象系统原有的对象结构表示框架使其具有新的语义,系统对复合对象的支持就是这方面的一个很好例证.

从数据抽象的角度看,类可看作对象产生器,一个类的实例是通过向这个类发消息,由这个类建立的,类分配给实例一个由实例变量决定的统一结构,并赋予相应于这个实例的各

初值,允许实例在这个数据结构上施加相应的操作.例如,假定一个工厂由三个部门:原料部门、生产部门和销售部门组成,则需要建立四个类 Factory、Supply、Produce、Market 分别表示工厂、供应部门、生产部门和销售部门.若有三个这种类型的工厂,则需要向类发消息建立相应的实例,这些类和实例的关系如图2所示.



F_i,S_i,P_i,M_i (i=1,2,3)分别表示相应类的实例

图2 类与实例的简单关系

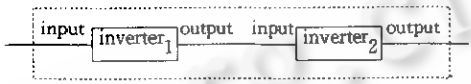


图3 位放大器(Bitamplifier)

由图可见,这种简单对象的层次结构无法表示实例 F_1 与 S_1、P_1、M_1 之间的关系,也无法刻划组成 F_1 的 S_1、P_1、和 M_1 之间的内在联系,然而在现实世界中由一组相关对象按某种方式构成一个复杂对象的现象是经常出现的,而且描述复杂对象中简单对象之间的构成关系对于复杂系统的知识表示和问题求解都是十分关键和重要的.因此,为了定义和操纵这类复杂对象,系统引入了复合对象的概念.

复合对象,又称复杂对象[7],是指一类特殊的对象,这些对象是由一组相关的对象按一定构成方式组合而成.在本系统实现中复合对象被称作组织(Organization),表示多个已定义的类的对象按某种关系有机地组织成一个整体完成相应的功能.这种将多个已定义的类的对象组合成新类的对象的方式,进一步提高了系统的数据抽象能力.

系统利用元类定义复合对象类是由一组固定成份按某种关系有机地组合在一起,这些组成成份可能是已定义的一些类,也可能是待定义的复合对象类.在复合对象类中,存在两种实例变量,一种是一般意义的属性变量,描述复合对象的内部结构的某些特征;另一种变量用来描述复合对象的组成成份,这些组成成份可以是一般的对象也可以是复合对象,在这种变量的省缺值位置规定这个成份所属的类.对一个复合对象类实例化即可得到一组不同的复合对象实例,且这组不同的复合对象实例中其组成成份之间的相对关系是同型的,复合对象与一般的对象相同,是由类来决定其内部结构和其上的操作,通过一个实例化过程来建立具体的复合对象实例.复合对象在类的继承网络中是可描述的.

例如,位放大器(Bitamplifier)是一种特殊的放大器,具有一般放大器的性质,并且位放大器是由两个转换器(Inverter)组成,其结构如图3所示.

由图中可见,$inverter_1$ 和 $inverter_2$ 可能是不同型号的转换器,$inverter_1$ 的输出与 $inverter_2$ 的输入相关联,Bitamplifier 的输入就是 $inverter_1$ 的输入,输出就是 $inverter_2$ 的输出.显然,Bitamplifier 是一个复合对象,它由两个对象 $inverter_1$ 和 $inverter_2$ 按一定关系组成,不同型号的 $inverter_1$ 和不同型号的 $inverter_2$ 就构成了不同功能的位放大器.

在实现复合对象过程中,系统首先在对象系统继承网络中建立一个系统,即元类 Organize,它定义所有复合对象类的省缺行为,如复合对象类的建立、修改、实例化过程等.用户

可以建立 Organize 的一个子元类，任何一个类如果它的直接或间接的元类是 Organize，则称这个类为复合对象类. 复合对象类中定义了一类复合对象的固定组成，通过对复合对象类进行实例化就可得到一个具体的复合对象. 元类、类、实例以及复合对象之间的逻辑关系可如图4所示.

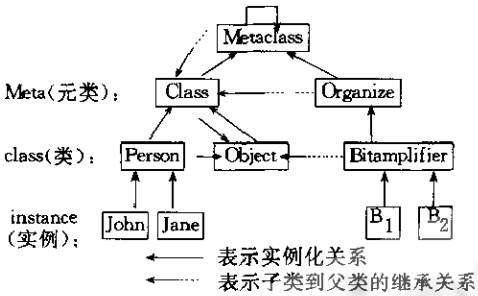


图4 一个对象世界的关系示意图

图4中 Metaclass、Class、Object 是系统内核中定义的三个基本的系统对象，这三个基本对象是生成系统内任何其它对象的"根"，奠定了整个对象世界的基础. Metaclass 是一个具有所有元类对象省缺行为的类，它是所有元类和它自身的元类；Class 是具有所有类对象省缺行为的元类，Class 是所有类的省缺元类，它也是 Metaclass 的父类，正是这种逻辑关系的定义才使得元类的操作可与类的操作统一处理；Object 是具有所有对象的省缺行为的一个类，Object 是继承树的根，是唯一一个无上级类的对象，即 Object 是所有对象直接或间接的父类. 整个对象世界的建立主要体现在这三个基本对象内部 Method 的定义上. 图中的 Organize 是系统定义的复合对象元类；Bitamplifier 是 Organize 元类生成的一个复合类对象；B_1、B_2是 Bitamplifier 实例化得到的二个复合实例.

前面已指出复合对象的实例变量有两种，对于描述属性的实例变量，其实例化过程与一般类对象相同；在描述组成成份的实例变量中，如果这个变量的省缺值指向一个已定义的类，则称这个变量值为常量，实例化过程常量要赋给这个类中的一个具体实例；若变量的省缺值指向一个其它的复合对象类，称这些变量值为参量，对于所有参量需循环实例化直到所有描述组成成份的实例变量都为常量为止.

例如，设系统中已定义了放大器类 amplifier 和转换器类 Inverter，上述复合对象类(Bitamplifier)可定义如下：

```
[DEFCLASS
    CLASSNAME: Bitamplifier
    METACLASS: Organize
    SUPERCLASS: ((amplifier Is-a))
    CLASSVARIABLES:
      num-of-parts 2
    INSTANCEVARABLES:
      inverter1   inverter
      inverter2   inverter
        Input     (¥ INDIRECT o ((obj inverter1)))
        Output    (¥ INDIRECT o ((obj inverter2)))
    METHOD
    ……(略)]
```

其中 Input、Output 的值部分都为激活值，本系统利用激活值实现对象变量之间的信息共享，以保证共享变量的一致性. 激活值思想在参考文献[8]中有详细介绍.

3.2 影子类

为了能够利用激活值机制体现各组成成份之间的相互关系，同时又不影响已定义类的

结构和方法的描述，系统在定义复合对象类时，引入了一个新的类对象称为影子类，这个对象只是为了更好地定义复合对象而引入的一种描述方法，影子类是在建立复合对象类时由系统自动建立的，作为复合对象成份的每一个已定义类都对应这样一个影子类，在影子类中，仅定义这个成份与其它成份之间的联接关系，以及复合对象中要求的而在原来类中未定义的新属性描述，并且要指明这个影子类的父类是某个已定义类，它们之间的继承关系是 A－shadow，影子类与父类之间仍遵从就近继承原则：在影子类中定义的同名变量将覆盖其父类中的同名变量的定义. 但不同的是已定义类中的所有实例变量(包括自定义的和继承的)均不拷贝到影子类中，这就好象是已定义类的一个影子，在它当中并无实际内容和操作.

在复合对象类的实例化过程中，当实例化参量和常量时，通过影子类向它对应的已定义类(即其父类)发送一个建立实例的 NEW 消息，当消息完成之后，返回一个具体的实例时，复合对象实例化过程再将这个实例中加入影子类中定义的新属性和成份之间的关联关系，完成这个参量(或常量)的实例化. 重复这个过程直到所有的参量或常量都实例化完为止.

上例中，复合对象类 Bitamplifier 定义过程中. 系统将自动生成如下两个影子类：

```
[DEFCLASS
    CLASSNAME: inverter1
    METACLASS: Bitamplifier (此关系系统设立,用户不可见)
    SUPERCLASS: ( (inverter A－shadow) )
    INSTANCEVARIABLES:
      input a
      output (¥ connect o ((obj inverter2)(insvar input)))
        ]
[DEFCLASS
    CLASSNAME: inverter2
    METACLASS: Bitamplifier
    SUPERCLASS: ( (Inverter A－shadow) )
    INSTANCEVARIABLES:
      input b
      output c
      ]
```

复合对象与影子类的继承关系可用图5来表示.

图5中，Bitamplifier 是复合类对象；inverter$_1$ 和 inverter$_2$ 是影子类对象；amplifier 和 inverter 是一般类对象.

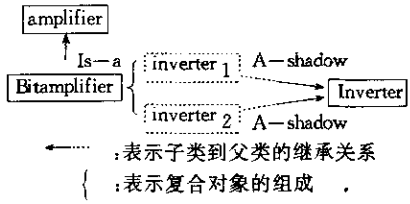


图5 复合对象继承关系示意图

值得注意的是复合对象的实例是通过向复合对象类发送一个 NEW 消息来建立的，实例化处理重复执行直到所有参量和常量以及其它实例变量均得到相应的初值为止，同一参量或常量的多次引用必须用被实例化的同一个实例来代替，如果一个复合对象需要同一类型的不同实例，则在描述中就需要有多个参量或常量来定义，如上例中，inverter$_1$ 和 inverter$_2$ 就是同一类型的不同实例，在复合对象中必须被说明两次. 在这种情况下，引入影子类的概念就显得十分

有用了,同一类型的对象只需要利用类对象定义一次,当复合对象要用到这个类对象的不相同实例时,只需根据这些不同实例所扮演的角色定义多个影子类即可,这样不仅减少了冗余说明和存贮空间,也使软件系统的可理解性得以增强.

MESBT 系统是在内核功能设计完成之后,利用内核提供的功能和设施完成复合对象功能的设计,在整个复合对象功能实现过程中,未对内核程序做任何修改,也没有对原有的内核功能产生任何不良影响,复合对象的实现仅仅是元类概念的应用和面向存取范例(Paradigm)相结合的结果,并且复合对象的功能与系统的其他功能有机地融合在一起,成为系统功能组成的一部分.因此,有理由说 MESBT 的对象系统能够对增量式开发环境提供一定的支持.

结束语:本文主要介绍了 MESBT 面向对象系统所引入的特性表机制和复合对象的概念.这些知识表示附加机制的引入,目的在于增强系统的知识表达能力和改善系统的可理解性和可描述性.虽然面向对象的知识系统还缺乏完备的知识模型和形式化的理论基础,但是面向对象系统的深入研究,将可能是促成软件开发、专家系统建造和数据库管理一体化的有效途径,从而为在大型复杂领域应用专家系统技术开辟了新的前景.

参考文献

1 Ken Kahn, Gerry Barber. Object—oriented programming. Expert System, 1988;(5):52—54.

2 Bobrow D G, Stefik M. The LOOPS manual. Xerox Corp. Palo Alto. Calif. 1983.

3 Stefik M J, Bobrow D G. Integrating access—oriented programming into a multiparadigm environment. IEEE Software, 1986.

4 Brent Hailpern. Multiparadigm languages and environments. IEEE Software, 1986.

5 姚莉,刘毅,刘凤岐.多程序设计模式的专家系统开发工具.计算机工程与科学,1990(3).

6 姚莉.集成多种程序设计模式的专家系统开发工具.国防科技大学,1990.

7 施伯乐,楼荣生,崔靖.数据库理论及新领域.北京:高等教育出版社,1990:282—328.

8 姚莉.面向存取语言模式及在多模式系统 MESBT 中的实现.In:吴涛,张晨曦 eds.中国青年计算机研究进展,第三届全国青年计算机会议,1991:419—424.

PROPERTY LIST STRUCTURE AND COMPOSITE OBJECT IN THE OBJECT—ORIENTED SYSTEM

Yao Li, Liu Fengqi and Wang Hao

(*Department of Computer Science, National University of Defence Technology, Changsha* 410073)

Abstract This paper discusses the concepts, functions and realization of the propertylist structure and composite object. Propertylist structure and composite object are some additional knowledge representation structures, which are introduced into the knowledge—based object—oriented system with the purpose of making the knowledge representation more powerful.

Key words Expert system, artificial intelligence, knowledge engineer, software tool.