计算机科学1991№.4

计算反射与面向对象程序设计(II) *)

奚建清　胡守仁（国防科技大学计算机系）

摘　要

This paper first discusses the information carried by an object and some relating problems. Then we present some reflective object-oriented models and their characteristics. These models are different from each other with respect to the self's concept and information. We give an effient model which extends Mayes' by the introducing of implicit meta objects.

我们在第一部分讨论了计算反射的一般模型。一般说来，一个系统是计算反射的，当且仅当它能对自身进行操作，据此一个反射的计算机软件系统有三个条件：

a）系统有关于自己(一部分)本身的描述。

b）系统有对自己的描述进行推理、操作和修改的能力。

c）这种描述的改变能直接导致系统的行为和状态的改变，反之亦然。

条件a是物理符号系统假设和知识表示假设[1]的必然结果。在现有的计算机计算模型中要讨论或建立一个计算反射系统，必须首先确定是哪一个(子)系统需具有反射能力以及它的"自身"的概念。正如第一部分所述，我们认为只有抓住了这个概念才能清晰地建立反射模型。

由于一个计算系统的"自身"的概念对于定义这个计算系统的反射概念是至关重要的，所以必须首先给定计算系统的较清晰描述，面向对象程序设计语言（数据库）在目前有许多不同的模型[2]。本文考虑基于类的、消息传递的模型，在此基础上给出几个反射模型并分析它们的反射能力和特点，结果也同样适用于其它计算方式的模型。本文的反射模型主要是根据一个对象的有关信息(即其自身的描述)的表示和访问方式来划分的，它们有a. ROO(rigrously objective object) 模型；b. OIA (object is all) 模型；c. CAM(class as meta) 模型；d. Mayes模型。c. d两个模型已在[3]中作了介绍，但由于缺乏对象的"自身"的概念，故不能较系统地讨论。

在深入讨论这些模型前，我们首先分析对象为什么需要和怎样才能够具有反射的能力，第二个问题是本文的主要内容，这里先回答第一个问题。已经有一些文章介绍了具有计算反射能力的程序的重要能力[4,5,6]，但并未从认知的角度回答这个问题。如果对象是一个问题域中的客观事物的严格的抽象物，即我们假设对象并不包含这个客观事物所不具有的属性与行为，例如一石头并无知道它在哪里的能力，则这个对象的所有计算行为必须符合我们'看到'的相应事物的行为，如果这个事物不具有反射（即反思）能力，则这个对象也不应该有这种能力。

可是正如[7]中指出的那样，作为程序，对象不仅仅是客观事物的严格对应物。人在抽象某一客观事物和过程时并不仅仅是反映这个事物的客观性，事物一旦以某种认识进入人的理念世界，则主观的和这个事物直接有关的一些计算过程就自然地认为是由这个事物引起，故抽象物比客观对应物在某些方面有更多的信息和更多的行为。例如在描述计算一块石头从空中落下的详细过程时，计算过程可能要求知道石头所在的位置、风向、自己的重量、体积以及它的一些计算历史等，以确定精确的下落过程。由于面向对象程序设计的模块性以及数据抽象

*）国家自然科学基金资助项目。

的要求，这些直接的计算过程自然地由这个对象来完成。对象的这种对其环境的表示和推理能力即反射计算的能力可使很多全局控制流局部化，使整个程序更灵活、生命力更强[1]。

一、关于对象的信息

一个对象描述了它所代表的事物的属性和行为，另外它还涉及其它信息，如作为程序（数据库）中的一个实体，它还有具体的计算信息，这些信息是关于这个对象的，而不是关于它所表示的事物的，例如这个对象的产生者，它的内外存地址、名字、存储形式、修改版本、甚至解释器的信息等。我们把这些信息称为这个对象的元信息（meta information），而把这个对象记录的相应的客观事物的信息称为这个对象的本信息(original information)。我们假定一个对象包含或涉及这两种信息，这两种信息有时是很难区分的，如这个对象接收的消息发送对象以及从哪个方法发送的消息。

涉及一个对象的信息种类有很多，语义和功能的要求使得我们尽量完整地表示一个对象的有关信息，包括这个对象的信息和这种表示的（隐含的）信息。另外由于事物间联系的客观性和表示的有限性，关于一个对象的计算行为和属性或许还不（可能）仅仅是由这个对象呈现，还可能由其它对象如它的类所呈现。

我们唯一能做的似乎是为一个对象定义一些(基本)操作作为这个对象的行为，涉及对象的其它操作均以这一组行为为基础，但不看成是这个对象的行为。另外一个标准是把一个对象起主要作用的操作看成是这个对象的行为。但不存在唯一的标准，因为在认识上总有各种选择。

据第一部分中反射的定义，一对象的一些结构和行为的信息必须明显地表示出来，才能使这一对象对自身的这些信息进行反射计算。一对象对应的计算子系统包括了这个对象本身、这个对象的明显表示（的信息）、一些计算相关对象以及解释器（包括硬件），这个对象的明显表示包括这个对象的结构描述、内存地址、行为描述、所属的类等等信息，也即本信息和（部分）元信息。因果相关性是指这个对象和这个对象的表示之间的因果相关性，这个对象和相应事物之间的因果相关联系在本文中不予考虑。一个对象的子系统中由谁来保存及处理这些信息就构成了我们区分上述模型的基础。下面逐个考虑各个模型及其反射行为。

在ROO模型中，每个对象仅含有本信息，对象中不包含任何元信息如对象名字、self变量，它的元信息由解释器维持，对象本身只有对本信息的操作，对元信息的操作只能由解释器预先定义的操作来间接地实现，从而据我们第一部分中的定义知，这种模型不是反射的。用户不能利用反射的能力，当然解释器仍然能隐含地完成一部分相当于计算反射的功能，但这种能力是预定义的，用户不能修改或扩充。在实际程序设计语言中没有这种模型存在，一般的语言中至少提供各个对象的名字、类名等可访问信息，故我们对此不作详细的讨论。

二、OIA模型

与ROO模型中对象的严格客观性互为极端的做法是对象的元信息和本信息均由对象本身保存，把我们认为与对象有关的操作均明显地归结为对象的行为，这些信息和行为描述都是这个对象可访问和操作的。

自包含的对象的行为由它自己以及解释器决定，通过访问它的元信息，它可以访问其自身并对之修改，这种访问可以是明显的，即计算反射的行为，也可以是隐含的——由解释器完成，由我们第一部分中计算反射的定义，这种行为不属于计算反射。由于是自包含的，这个对象的方法对它的任意属性的访问不需要以访问它的模板或类为前提（就象objvlisp中那样），而直接据自身包含的结构信息来访问各个属性（当然也可以象CLOS中那样为每个属性提供读写访问方法），因为对它的（属性或方法的）任何访

问修改均是直接的且是过程性的，符合检查（即因果相关性联系检查）是直接实现的。若对体的修改导致了一个旧的属性或方法的删除，则这个属性值或方法体就不存在，访问函数也同时不存在了，对它的后继访问将会出错。

这种模型中，一方面实例对象是自包含的，另一方面它的类也同样描述了它的结构和行为信息，从而实例对象的反射计算不仅要保证因果相关性，还必须保证实例和其类的语义一致性。类与对象实例的一致性可用两种方法来达到，一是对实例体的修改操作直接以类规定的语义为出发点，例如Objvlisp的实现就用这种方法。第二种方法是实例的结构描述和它的类的关于其实例的描述可以共享一个，这种共享如果是物理上共享，即在实现时（而不是在逻辑上）把实例的访问结构操作和类的语义检查操作实现成同一个操作，则运行效率较高。

这个模型的优点是对象对自己的访问可以特殊处理从而提高速度。另外一个优点是由于关于对象的任何信息均可汇集在这个对象内，只需要给出明显的语义链(即属性名)，从而不需在不同的地方访问对象的本信息和元信息。在有些时候，一些信息是难以区分成上述两部分的，如对同类事物的计数操作和对同类对象的计数操作。其缺点是在有些应用中，用户可能并不希望这种信息堆聚，即相关信息全放在对象内而要在概念上明显地把一个对象的两部分信息分开。另一个缺点是不同对象的共同信息，如它们的结构描述均重复地存放在各个对象内，既浪费空间且不支持对它们的修改，当然可以考虑类似于上述的实例和类的一些信息共享的方法。另外信息堆聚可能会造成对象单位的过于庞大和繁杂，给OODB的存储管理带来困难。下面的模型考虑解决这个问题。

三、CAM模型

类(class)概念的引入，本意是为了抽象一组行为相同的（实例）对象，也即体现这些对象的共同行为和属性，看成是这些对象的模板（template），所以类必须（部分地）是这些属性和行为的明显呈现（或表示），从而实例对象本身不需存储它的结构和行为的描述，而只需存储各种属性值。一个对象的自身由这个对象、类和解释器组成，语言Objvlisp就是一例。

一般而言，类的作用是在创建一个新的对象时作为这个对象的结构属性及行为的格式，一经产生，则这个对象与相应的类便是相互独立的计算实体。由于类反映了一个对象的结构及行为，在反射计算系统内，一个对象可以利用它的类提供的这些信息来进行对自身的结构和行为描述的访问及修改。

计算反射要求一个对象对自身的修改立即反映在这个对象的结构和行为中，反之，语义一致性要求对一个对象的任何访问修改及这个对象的行为描述均符合模板规定的语义，除非这个对象演化为新的类（模板）的实例，（这时它的结构和行为有新的要求，从而具有新的模板）。当然对象与其模板的符合检查在这个模型中不仅是为了语义一致性，由于实例本身不能保证它的行为结果与描述的因果相关性，所以每次涉及它的操作都必须以模板规定的语义为出发点，并且其对自身的操作结果要反映在其自身内。

从上面的分析看到，象Objvlisp这样的语言模型，由解释器进行的模板描述与对象行为的符合检查完成了两种功能，即语义一致性和因果相关性的检查。在Objvlisp中符合检查是通过对一个对象的任何操作均隐含地访问模板中这个对象的结构和行为的描述进行的，并且符合检查并不对这个对象的自身进行一般意义的推理，从而据我们第一部分中的定义，这种行为不是计算反射，当然Objvlisp由于把每个类看成是对象，从而一个对象可以通过其类访问到它的结构和行为信息，所以这个系统提供了实现计算反射的基础。模板作为一种描述性设施，同时也满足了计算反射的明显表示要求，这种模型的

优点是信息不会出现冗余，概念比OIA模型清晰，不需引入额外的对象，能够适合一定的应用。缺点是如果每个实例对象的元信息是不同的且可能很多，则类和解释器难于应付这种需要。另外像Objvlisp的实现中所示，对每个对象的访问均须访问它的类，故效率较低，再就是没有完全克服信息堆聚的缺陷。

四、Mayes模型

这个模型是P. Mayes提出的[9]，它为每个对象单独地设立存放关于它的元信息的对象——称之为它的元对象。在这个模型中，类、元对象和对象本身一起保存着关于或涉及这个对象的计算信息。元对象的引入使得同一个类中的不同实例可以有不同的元对象，这提供了上面两个模型所没有的灵活性。由于各对象间的信息交流需要通过明显的对象访问操作，这使得速度可能比OIA模型慢，在基于函数调用的系统中，这两种速度是一样的。这个模型的另一缺点是其它对象在使用有关这个对象的信息时，必须区分这个信息是属于哪一种，即存放在元对象上，还是放在对象内。当然，通过适当的控制，可使这个元对象对一般用户是透明的，用户只须知道这个对象，这样对用户相当于运行OIA模型。

元对象的作用可以进一步发展使之上升到了作为对象的代理人的地位[4]，而对象成为一种附属的专用于存放信息的设施，也即元对象控制了这个对象的所有行为（截取所有发向这个对象的消息），而元对象又可以有自己的行为，也即有自己的元对象。这种模型中，一个对象的自身也同样包括了这个对象、元对象、类以及解释器，而元对象是主要的行为体现者。

上面几个模型集中讨论了在对象子系统中的对象的反射计算，在[3]中还给出了基于通讯的反射模型，即对消息的反射。实际上这种反射是解释器中的send函数的反射行为，如果把每个消息看成是一个对象，且可以有自己的元对象，则这个对象也可以进行各种计算反射操作。另外方法查找、内存分配、事物处理等功能模块都可以看成是特殊的对象，从而它们可以有自己的计算反射行为。像CLOS中的辅助方法机制也可看成是对方法的反射。

五、计算反射的典型应用与一个基本实现模型

计算反射在对象系统中的作用可以在一个消息的作用期中看出，一个消息是通过函数send发出的，形为(send obj1 msg1 args)，表示要求obj1执行名为msg1的方法，那么这个消息的生存期主要涉及如下的计算：

a. 在执行这个方法前后中间等是否应执行其它辅助方法。

b. 这个方法的执行者obj1是否在内存、同一处理器内（分布式情形），以及是否正在忙或是否正被计算生成（并发执行的情形）。

c. 对应的方法代码体如何寻找。

d. 方法执行是否应成为一个事物处理。由于一个消息连接了两个对象，即发送者、接受者，而消息本身也可以是一个对象，故上述各操作可由这三者来完成。

当然对于一个复杂的OODB，上述功能只是系统的一部分。另外还有对象的永久性判定、对象性质的演化、对象的存储管理、版本管理、历史信息的管理和利用、对象对各个用户的界面以及不同数据模型之间的转换等功能均可通过计算反射来清晰地实现，在一个没有明确引入计算反射的系统中要综合地实现这么多的功能是很困难的。

根据上面的对各种反射模型的分析，我们给出一个正在实现的OODB的反射模型的基本实现方案。限于篇幅，对这个模型中与本文不相关部分不作介绍，我们主要介绍元对象的结构和消息的实现方法。

5.1 元对象的结构

类似于Mayes模型，我们可以为每个对象单独设一个元对象，但也可以没有元对

象，并且元对象和相应对象的联系仅是它们的名字的联系。如果一对象的名字是n，则可能的元对象的名字可以是meta-n，且这个meta-n有可能是用户不明确知道的。为了克服Mayes模型中的缺点，我们采用了特殊高效的访问方法。

一个对象可以明显地由用户定义其元对象，也可以由系统自动地定义。一个对象具有元对象的充分条件是以下一个或几个条件：

a. 对象的消息或方法需要重新解释。

b. 对象需要版本和历史信息的管理。

c. 对象的结构需作一定的改变，但仍保持所属的类关系。

d. 对象对外有许多不同的界面。

e. 对象的结构有静态或动态限制。

f. 用户直接定义。

在类结构中每个元对象都有它的类，一个元对象可以被多个对象共享，也可以给一个对象专用。

一个元对象的结构大致为：

属性

reflecting
persistent
clustered
versioned

方法

cluster-forming
version-control
view-control
persistence-deciding
structure-constraining
method-interpreting

5.2 对象反射的实现简介

对于不同的元信息可以有不同的实现方案，下面是我们的一些实现技术。

①如果消息要重新解释，则设一个变量指明，解释器在对象第一次接收这个消息时需检查这个对象是否有元对象存在，如有则由元对象负责这个消息的解释，并生成对应于这个对象的新的方法体（下一次操作不需作上述解释）。

②在涉及到一个对象的版本或历史等元信息时，通过标准的或用户自己定义的函数在相应的元对象中访问；另外为了保证对象的使用者的操作的一致性，系统的一些出错信息应由元对象进行处理。

③如果一个对象对不同的用户提供不同界面，或结构有异，或有特殊的限制，则由元对象提供的方法在相应的对象的方法体中进行适当的修改（如加入一些辅助函数）。这些修改可以静态进行，类似于CLOS语言中的辅助方法机制提供了十分方便的手段。

此外我们还实现了一个对象的一些局部环境，包括它的类名、元类名、所有超类、直至消息发送者，这些设施不仅效率高，而且大大加强了对象的智能化基础。

六、讨论与结语

下面是关于面向对象的反射模型的一些问题的讨论。

1）在上面讨论的OIA、CAM和Mayes模型中，由于对象的（大部分）元信息（如一对象的结构）是在对象系统内，而不在解释器内，故对象对自身的这些信息的访问不需象第一部分所述的那样要在不同的运行环境下操作（升级）。如果有对象（程序）是处在另外的对象（程序）系统，则对这些对象的反射计算可能需要某种“升级”，如进行数据格式的转换。

2）在计算系统中的各种反射行为可能存在着冗余性，因为如果运算也是一个对象，并涉及许多其它对象，由于这个运算对象本身是反射的，故它可以访问它所涉及到的对象（作为其本身的一部分），而这些对象又可以是反射的，所以它们可以访问判定上述运算，从而导致两者在能力上冗余。

本文根据第一部分中对于计算反射的分析并结合面向对象模型中的具体情况，按照对象子系统中自身的概念和信息的分类、存放和操作的方式来划分了不同的反射模型，

并分析了它们的特点，我们认为元对象是一个较好的概念，它不仅能够较清楚地划分各个功能部分，而且为编制更具有智能的程序提供了基础，但应避免由于它的引入而造成的低效，在第六节中给出的实现方案反映了这种思想。

反射模型中对象子系统中的元对象、元类、对象和解释器的关系以及相互作用在本文中没有深入地讨论，它们在类结构中的关系和功能分配以及实现技术是以后研究的目标。另外统一的计算反射术语、反射结构和基本实现技术对于系统的设计和开发是极有帮助的。

刘凤歧教授阅读了本文的初稿并提出了宝贵的意见和指正，在此表示忠心的感谢。

参考文献

〔1〕 Brian C. Smith, "Prologue to" Reflection and Semantics in a Procedural Language. Technique Report MIT/Lcs LTR-272, 1982

〔2〕 Peter Wegner, "Dimenions in Object-based Language Design" in OOPSLA'87 Proceedings

〔3〕 Jacques Ferner, "Computational Reflection in Class Based Object-Oriented Languages" OOPSLA'89 Proceedings

〔4〕 Pattie Mayes, "Computational reflection" The Knowledge Engineering Review vol.3 no.1, 1988

〔5〕 Nicolas Graube, "Metaclass compatibility" OOPSLA'89 proceedings

〔6〕 Takuo Watanabe and Akinori Yonezawa, "Reflection in an Object-Oriented Concurrent Language" OOPSLA'88 Proceedings

〔7〕 奚建清，胡守仁，刘凤岐："论知识表示"，计算机工程与科学1990，第三期

〔8〕 L. Steels, "Meaning in Knowledge Representation" in "Meta-level Architecture and Reflection" P. Mayes and D. Nardi ed. North-Holland Press 1988

〔9〕 Pattie Mayes, "Concepts and experiments in computational reflection". OOPSLA'87 Proceedings, Florida, 1987

〔10〕 S. B. Zdonick, "Language and methodology for object-oriented database environment" Proc. of the 6th Hawaii System Science Conference, 1986

〔11〕 奚建清，胡守仁："面向对象知识库中的类型演化"，计算机工程与科学1990，第三期

〔12〕 Nicolas Graube, "Reflective Architecture; From Objvlisp to CLOS" ECOOP '88. LNCS 327

〔13〕 Brian Foote and Ralph E. Johnson, "Reflective Facilities in Smalltalk-80" OOPSLA'89 Proceedings

〔14〕 Ronard J. Branchman, "The future of Knowledge Representation" AAAI-90 Proceedings

下期主要内容预告

专家系统与神经网络能够结合在一起
人工智能研究需要更高标准
数据库论战卷土重来
函数式、逻辑式和对象式程序设计及其多范例合成语言
开放系统的概念模式
关于解释学习逻辑构架的一些扩充
信息系统面向对象需求分析
RSOODM——一种引入联系的面向对象的数据模型
对象管理系统