形式概念分析在面向对象设计中的应用

邹长龙

（大连海事大学信息科学技术学院 辽宁大连）

摘要: 面向对象是相对于面向过程来讲的，面向对象方法，把相关的数据和方法组织为一个整体来看待，从更高的层次来进行系统建模，更贴近事物的自然运行模式。而面向对象设计中的难点之一就是类的抽取。所以本文主要探讨类的抽取。类层次结构是面向对象软件开发中最重要的活动之一。类及其层次结构设计是一项困难的任务，尤其是当所寻求的是一个广泛而复杂的建模时。有些问题即使使用方法学建模也很难理解。类层次结构的精确构造需要对问题有深入的理解，正确识别属性和方法，以及它们的相似性、依赖性和特殊化。不准确或不完整的类层次结构会导致软件的设计缺陷，使维护或更正变得困难。形式化概念分析提供了一种理论，使得类层次化能够实现类的最大分解，同时保持了类间的特殊关系。

关键字:面向对象；类；层次结构；形式概念

**Application of Formal Concept Analysis in Object Oriented Design**

ZOU ChangLong

(School of Information Science and Engineering, Dalian Maritime University, Dalian China)

Abstract: Object oriented is relative to the process oriented, object-oriented method, which organizes the relevant data and methods as a whole, carries out system modeling from a higher level, and is closer to the natural operation mode of things. One of the difficulties in object-oriented design is class extraction. So this paper mainly discusses class extraction. The class design and its hierarchy is a difficult task especially when what is sought is an extensive and complex modeling. Some problems are difficult to understand even when modeled using a methodology. The precise construction of a class hierarchy requires deep understanding of the problem, a correct identification of attributes and methods, their similarities, dependencies and specializations. An inaccurate or incomplete class hierarchy entails manufacturing defects of the software, making it difficult to maintain or make corrections.

Keyword: Object Oriented; Class; Hierarchy; Formal concept

# 1 引言

层次结构的设计和维护被认为是一个难题。这种困难随着所涉及的类的数量和可能的需求演化的增加而增加，这些需求可能需要变更层次模型。

有些问题域即使使用方法学建模也很难理解。类层次结构的精确构造需要对问题有深入的理解，正确识别属性和方法，以及它们之间的相似性、依赖性和特殊化。不准确或不完整的类层次结构会导致软件的设计缺陷，使维护或更正变得困难。

正确地应用面向对象的概念，使软件组件能够重用，并使开发的软件具有更高质量、更容易维护、更容易适应和扩展。

FCA的主要目标是根据对象的属性对其进行分类。在FCA中，问题域通常被建模为一个交叉表，称为形式上下文，其中行对应对象，列对应属性。

在面向对象的设计过程中，FCA理论可以应用到类模型中，从而对模型进行更深入的审查，并确保质量合格。

在工程、自然科学、人文科学等多种知识领域的软件设计中，通常需要设计者具有专业的技术知识，这使得设计者更难对这些系统的类结构进行建模。形式概念分析，也可以称为概念格，可以为其相关类的设计提供方便。

# 2 形式概念分析的相关定义

形式概念分析是由R.Wille于1982年提出，是应用数学的一个分支，是信息处理的一种理论，是知识处理的一种理论，是对现实世界的对象和属性的提取。

FCA的表示可以从描述其特征的属性列表中获得一组对象或实例之间的关系，从而形成形式概念。如表1。

表1

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | a | b | c | d | e | f | g | h |
| 1 |  | **×** |  | **×** |  |  |  |  |
| 2 | **×** | **×** | **×** | **×** |  |  |  |  |
| 3 |  | **×** |  | **×** | **×** |  | **×** | **×** |
| 4 |  | **×** |  | **×** | **×** |  | **×** |  |
| 5 |  | **×** |  | **×** |  | **×** |  |  |
| 6 | **×** |  | **×** |  |  |  |  |  |

表1给出了一个对象（行）的定义、属性（列）及其各自关联关系的结构。一个形式上下文（G，M，I）由两个集合G和M以及这些集合之间的二元关系I组成。G的元素称为对象M的元素称为属性。

通过形式上下文可以生成概念格。概念格是一个有向图，其节点表示建模的对象或实体，或者仅仅是概念间的关联。与节点耦合的是模型和/或方法的属性。 概念格允许在各种应用程序中提取概念，例如面向对象方法中的数据库设计或类设计。图1给出了表1的形式上下文概念格。

图1：形式上下文的概念格

在一个概念格中，如果概念A是概念B之上的一个概念，并且这两个概念是连接的，那么概念A可以被视为比B更一般的概念，因此，可以在A和B之间添加公共属性。因此，如果B发生，A也发生，这表明了递推逻辑。概念格不仅描述了概念的层次结构，而且描述了这些概念之间的一整套二元关系。这将使对象分析是可视化的，可以通过在类层次结构中搜索来获得。

在图1中，图中的每个节点都是一个概念。如果两个对象被放置在同一个节点（概念）上，那它们具有相同的属性，因此是具有该属性集的同一类对象的实例。

因此，FCA提供了一个工具，用于对共享公共属性和方法的元素组进行识别，这些元素组揭示了隐式和显式的依赖关系，从而能够更好地理解概念。

# 3 应用形式概念分析进行类抽取

当使用形式概念分析来设计类层次结构时，形式对象的集合G是一组软件产品，换句话说，是类、对象或程序变量，它们被当作搜索适当的类层次结构的起点。形式属性M的集合对应于类或对象的属性集。相关的属性集包括属性（实例变量）和方法（主体和/或方法标识）。

类抽取这项工作的一个重要方面是最小化冗余，并通过特殊化创建子类。关于冗余最小化的思想是分解类减少不一致性和最小化未来的冗余代码。对于子类，它还使用类的因子分解，通过设置类型和子类型的标签来标识层次结构。

为了获得类的最大因子分解和一个新的层次模型，将采用以下步骤：1.为上下文表映射模型实体。2.生成概念格。3.在目标语言不支持多继承的情况下删除多继承。4.删除不一致的类5. 分割具有不同的语义类的公共属性。

## 3.1映射上下文表的模型实体

首先考虑软件设计者可以选择从类模型的原始层次结构或同一层次层次结构中选择实体或对象，而没有专门或关联关系。对于所选模型中的实体或对象，设设计器列出它们的特性。由于对象是类的实例，因此可以假定找到的实体可以被视为初始类的模型（或一组具体类）。

基于前面所述，考虑在一个上下文表中从模型中映射属性。下面的例子说明了这个基本想法。假设下面为一组由四个具体类组成的属性规范，如图2所示。规范可以解释为层次结构应该包含的具体类的确切集合，换句话说，这些类是在应用程序中生成对象的唯一类。

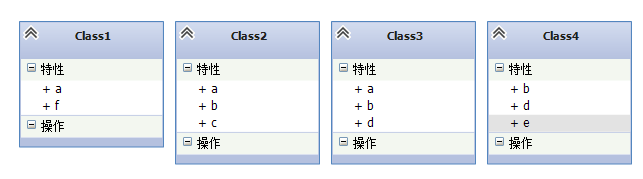


图2：实体类

实例变量如图3所示。上下文被设计成一个表的行和列，类用整数标识，变量用字母标识。

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | a | b | c | d | e | f |
| 1 | x |  |  |  |  | x |
| 2 | x | x | x |  |  |  |
| 3 | x | x |  | x |  |  |
| 4 |  | x |  | x | x |  |

图3：形式上下文

## 3.2生成概念格

因为问题在于如何将这些类组织成一个层次结构，所以我们使用概念格作为设计的指南。每个形式概念被解释为层次结构中的一个类，类之间的联系被视为关系的特殊化。图4展示了一个概念格。分配给概念的标签表明应该声明一个属性类。例如，声明的属性 a 和 b 是两个普通类，它们位于类层次结构的根的下方。在类层次结构中，由底部节点定义的概念被忽略，因为它没有用处，因为它不表示类的信息。

图4：图3的形式上下文的概念格。

图5显示了按照概念格的解释分解的类的层次结构。这四个初始类仍然在层次结构中，但是由于概念格的生成和分解，在这些类中声明的属性更少。

新的类(节点5-9)由于公共属性的因子分解而被添加的。这些是空类，因为实例仅为四个初始类创建。概念格标签化的本质保证了每个属性在层次结构中只出现一次。初始具体类中的对象属性保持不变。然而，现在其中一些被一些新的类继承了。一般来说，所有子类都是继承父类属性的特殊化。

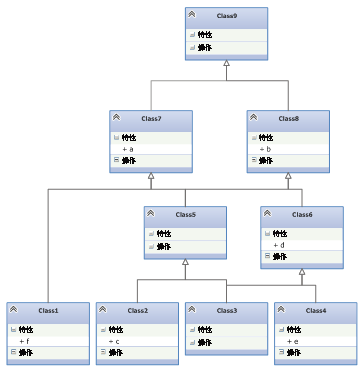


图5：图4的概念格中的类层次结构

## 3.3消除多重继承

有大量的设计可以最小化冗余。概念格通过最小化类和多继承(针对不支持多继承的目标语言)的数量来实现这一目标。

这是通过尽可能对类进行分组来实现的，如图6所示。

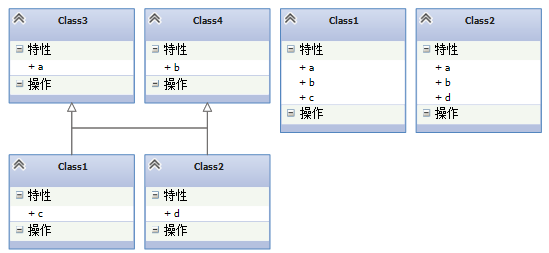


图6：消除多重继承。

图6所示的类图，在左边，将公共属性a和b分解出来，但是更为复杂，因为它包含四个类，每个属性对应一个类。相比之下，图6右侧显示的设计更简单，并提供了相同的质量，以避免在单一继承模型中出现冗余和特殊化。一个包含多重继承的模型在一个包含单一继承的模型中的转换是仅仅在其子类的上升子类的属性拷贝上进行的，而其中存在多重继承关系。因此，在复制完成后，如果没有其他类与其绑定，上升类将不再存在于模型中。

## 3.4删除不一致的类

概念格是一组具体类之间相似性的表示。随着它的规模迅速增长，可以考虑忽略它的一些节点，以保持其结构的可管理性。因此，第一个想法可能是删除不声明任何属性或方法的类。这些通常称为空类的类可以在不违反质量标准的情况下被删除，换句话说，因为其没有冗余和特殊化。在图5的示例中，可以省略空类Class5和Class9。尽管Class3没有声明任何属性，但它被保留下来，因为它是一个底层类。

删除所有空类所产生的结构称为子层次结构，它对应于概念格中所有属性和对象概念的简化集合。图7描述了被修剪的新概念格。

图7：删除空类的概念格的结果。

考虑到编程语言支持多重继承，因此产生的新类模型的结构如图8所示。

图8中描述的类是将图2中的原始类映射到上下文表中的结果，而上下文表又转换为概念格并释放空类。但是，如果设计的初始定义先决条件之一是在没有多重继承支持的情况下建模类，那么生成的类图将根据图9进行建模。

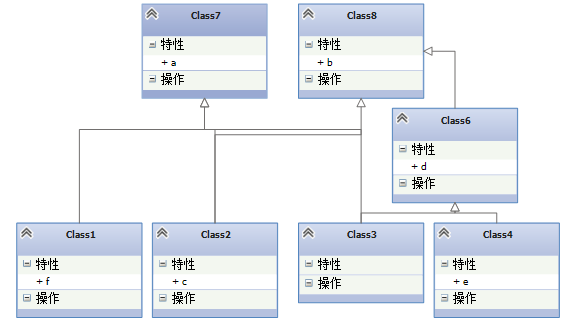


图8：应用分解步骤后生成的类模型。

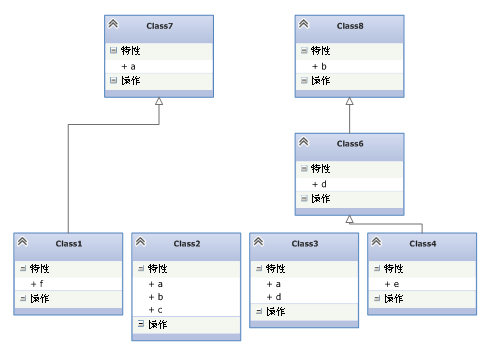


图9：不支持多重继承的结果类模型。

## 3.5分割具有不同的语义类的公共属性

软件设计人员定义的类属性可能具有相同的标识标签。然而，它们的属性可以不同，就是使它们在语义上不同。这一事实意味着，虽然它们有相同的名字，但它们没有相同的特征。这样，它们就不应该被归类为属于概念格中创建的新一代祖先类的单个属性。

图10展示了两个表，举例说明了现有类的属性映射。图10的第一个表描述了每个类的属性，第二个表用粗体突出显示了公共属性。在本例中，它被认为是图8中的类模型，它支持多继承。

|  |  |
| --- | --- |
| Class1 | a,f |
| Class2 | a,b1,c |
| Class3 | a,b2,d |
| Class4 | b2,d,e |
| Class5 | b2,d |
| Class6 | a |
| Class7 | b1 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | a | b1 | b2 | c | d | e | f |
| Class1 | **x** |  |  |  |  |  | **x** |
| Class2 | **x** | **x** |  | **x** |  |  |  |
| Class3 | **x** |  | **x** |  | **x** |  |  |
| Class4 |  |  | **x** |  | **x** | **x** |  |
| Class6 | **x** |  | **x** |  | **x** |  |  |
| Class7 |  |  |  |  |  |  |  |
| Class8 |  | **x** |  |  |  |  |  |

图10：第一个是分解类属性的映射，第二个是公共属性的映射。

|  |  |
| --- | --- |
| Class1 | a,f |
| Class2 | a,b,c |
| Class3 | a,b,d |
| Class4 | b,d,e |
| Class5 | b,d |
| Class6 | a |
| Class7 | b |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | a | b | c | d | e | f |
| Class1 | **x** |  |  |  |  | **x** |
| Class2 | **x** | **x** | **x** |  |  |  |
| Class3 | **x** | **x** |  | **x** |  |  |
| Class4 |  | **x** |  | **x** | **x** |  |
| Class6 |  | **x** |  | **x** |  |  |
| Class7 | **x** |  |  |  |  |  |
| Class8 |  | **x** |  |  |  |  |

图11：公共属性及其新标签的分段。

在分解过程的这一阶段，软件设计人员必须进行干预，以识别具有相同标签且属性不同的属性。这一步是半自动的。结果如图11所示。

得到一个新的类模型，其中Class2继承了Class8的已重命名属性b1, Class9继承了已重命名属性b2。这两个属性的来源都是属性b，它来自前面的类模型。图12显示了生成的新类层次结构。

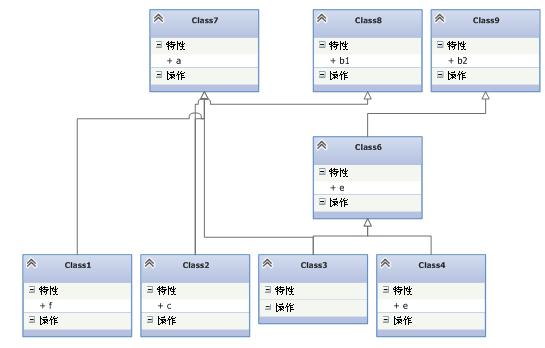


图12：由公共属性分段产生的新类模型。

# 4结语

本文介绍了FCA在面向对象软件设计中冗余类的分解和最小化的基础。使用适当的方法，可以自动化和优化软件的设计阶段，同时保持面向对象类模型固有的特性。可以看到，通过使用形式概念分析（概念格）对类的层次结构进行建模是有效的，可以节省软件设计者和软件开发者的时间，提高软件开发效率。

参考文献

[1] 王千祥, 吴琼, 李克勤,等. 一种面向对象的领域工程方法[J]. 软件学报, 2002(10):1977-1984.

[2] Lang G , Luo J , Yao Y . Three-way conflict analysis: A unification of models based on rough sets and formal concept analysis[J]. 2020:105556.

[3] 晏力, 刘鹏慧. 基于形式概念分析的属性约简[J]. 西华大学学报(自然科学版), 2012, 04: 37-41

[4] 蒋平.基于Eclipse的概念格构建系统的设计与实现[J].计算机技术与发展，2011（04）：01-05.

[5] 张云中，徐宝祥.基于形式概念分析的信息系统建模理论研究[J].现代图书情报技术，2010（02）：12-15.

[6] Zhao M , Zhang S , Li W , et al. Matching biomedical ontologies based on formal concept analysis[J]. Journal of Biomedical Semantics, 2018, 9(1):11.

[7] 蒋平, 任胜兵, 林鹃. 形式概念分析在软件工程中的应用[J]. 计算机技术与发展, 2008(04):127-129.

[8] 王燕. 面向对象的理论与C++实践[M]. 清华大学出版社, 1997.

[9] Ganter B , Stumme G , Wille R . Formal Concept Analysis, Foundations and Applications[M]. Springer-Verlag, 2005.