一种基于模糊形式概念分析的程序聚类方法*

许佳卿 彭鑫 赵文耘
(复旦大学计算机科学技术学院 上海 200433)

Program Clustering for Comprehension based on Fuzzy Formal Concept Analysis

Xu Jiaqing, Peng Xin and Zhao Wenyun
(School of Computer Science, Fudan University, Shanghai 200433)

Abstract Program clustering can help program comprehension and architecture analysis by clustering program units relevant to certain requirement or design element. Formal concept analysis (FCA) has been widely adopted in text-analysis-based program clustering. However, existing FCA-based methods are based on one-valued attributes (with or without), and cannot deal with the fuzzy information in program clustering. In this paper, a text-analysis-based program clustering method using fuzzy FCA is proposed. The method includes a process of fuzzy attribute computation and a construction algorithm for fuzzy concept lattice, which can ensure a complete and non-redundant lattice. A semi-automatic analysis tool has been developed for the method and applied in a case study of a commercial bookstore management system. Results of the experiment show that the method can help to obtain clusters with sound fuzzy features for program comprehension, which are also recognized by former developers of the system.

Key words Formal Concept Analysis; fuzzy FCA; fuzzy attribute; program clustering; program comprehension

摘要 程序聚类通过将与同一个需求或设计元素相关的代码单元聚集在一起来辅助程序理解及系统结构分析。其中，形式概念分析（FCA）是一类被广泛采用的程序聚类技术。然而现有基于 FCA 的程序聚类方法都是基于二值属性构建的，无法处理模糊信息。本文提出将模糊概念分析用于基于文本分析的程序聚类，提出了一种支持模糊信息的程序聚类方法。该方法包括模糊属性的采集过程以及相应的模糊概念格的构造算法。在此基础上，我们开发了一个半自动化的程序分析工具，并将其应用到一个商业软件分析中。初步的实验结果表明该方法能够有效地支持基于模糊特征的程序聚类，对于提高遗留系统的维护效率有明显的帮助。

关键字 形式概念分析；模糊形式概念分析；模糊属性；程序聚类；程序理解
中图法分类号：TP311

* 基金项目：国家自然科学基金项目（60703092）、国家"八六三"高技术研究发展计划基金项目（2007AA01Z125）

1 简介

程序聚类可以通过将大型复杂的软件系统分解为小型的更易管理的子系统来帮助进行软件维护和演变[1]。每个子系统都由一系列源代码制品构成，这些源代码制品彼此之间通过紧密合作来实现系统的某个特征或者为系统的其他部分提供服务[2]。

根据聚类过程中使用的信息，可以把方法分为两类。第一类方法从代码级别的实体中挖掘结构化关系来分解系统，它们只使用源代码中的形式化信息[1]。这类方法中包括[3][4][5][6]。另外一类方法使用了源代码内部以及以外的非形式化信息（比如注释、代码文件名、路径名以及从属关系等）[1]。这类方法一般使用的是基于文本分析的程序聚类，相关方法包括[1][7][8]。第一类方法中又包含动态方法和静态方法。动态方法（比如[9]）采用了一种基于测试的方法来运行程序并且收集执行路径作为聚类的主要基础。静态方法（比如[10]）直接对源代码进行分析，在程序聚类的过程中仅使用静态信息。

我们在本文中关注第二类方法，也就是基于文本分析的程序聚类。FCA（形式化概念分析）被广泛应用于程序聚类、数据挖掘和信息抽取等研究领域。FCA[10]是一种用来分析二元关系的数学方法。Thomas Eisenbarth 等人[9]在源代码中进行特征定位，场景与计算模块之间的关系由 FCA 来进行处理。标识符分析也运用了 FCA，将分析程序中的所有类和方法作为对象，将与类和方法相关联的标识符作为属性[11]。

本文采用基于模糊形式概念分析（fuzzy FCA）的方法来进行程序聚类。模糊 FCA 在传统 FCA 的基础上引入了模糊的概念[12][16]。我们的聚类方法是基于文本分析的，也就是将源代码作为对象、将源代码中的关键字作为属性。本文使用模糊值来描述对象与属性之间的模糊、不确定的关系，而在传统 FCA 中这样的关系是单值的。σ 和 λ 是两个模糊参数，它们可以帮助开发人员更好地理解以及评价聚类结果以及改善聚类过程。

本文编排如下。第二部分对于 FCA 以及模糊 FCA 的背景知识进行介绍，讨论了在基于文本的程序聚类中使用模糊信息的必要性和可行性。第三部分描述了基于模糊 FCA 的程序聚类方法，包括模糊信息的获取以及模糊概念格的构造。第四部分我们给出了本方法应用于一个商用书店管理系统中的实验结果。我们在第五部分中介绍了相关工作，在第六部分中对本文进行总结并提出了将来的工作。

2 背景知识

2.1 形式化概念分析

FCA 是一种基于格理论的分析方法，它处理的是某一领域中形式背景上对象和属性之间的关系。对象和属性之间的关系可以通过概念格来表述[12]。概念格相对于传统的树型概念结构包含更丰富的信息，它支持多重继承[12]。FCA 是已经被广泛应用到包括数据挖掘、信息抽取（比如[12][13]），以及程序聚类（比如[4][5]）中。

FCA 处理的是在一个对象集合 O 和一个属性集合 A 之间的二元关系 $I \subseteq O \times A$。元组 C=(O,A,I) 被称为一个形式背景。对于一个对象集合 $O \subseteq O$，集合的公共属性 σ(O)被定义为：

$$\sigma(O)=\{a \in A \mid (o,a) \in I,\ \forall o \in O\}. \quad (1)$$

同样的，对于一个属性集合 $A \subseteq A$，集合的公共对象 τ(A) 被定义为：

$$\tau(A)=\{o \in O \mid (o,a) \in I,\ \forall a \in A\}. \quad (2)$$

当且仅当 $A=\sigma(O)$以及 $O=\tau(A)$时可以将一个元组 c=(O,A)称为是一个概念。对于一个概念 c=(O,A)，O 被称为是概念 c 的外延，A 被称为是概念 c 的内涵。

一个给定形式背景上的概念集合构成了一个偏序关系。给定形式背景上的概念集合以及这种偏序关系共同构成了一个完整的格，称为概念格。

2.2 模糊形式概念分析

传统的基于FCA的概念聚类方法很难来表示模糊以及不确定的信息[12]。因此，就有其他学者提出了使用模糊 FCA[12][14][16]来处理问题，模糊 FCA包含了被重新定义的FCA概念以及一些用来表示模糊信息的全新概念。我们在以下给出相关的定义。

定义 1. 模糊形式背景。一个模糊形式背景表示为 F=(O,A,I)，其中 O 为一个对象集合，A 为一个属性集合，映射 I 称为隶属度函数，这个函数满足：

$I: O \times A \rightarrow [0,1]$，或者记作 $I(o,a)=\mu$，其中，$o \in O$，$a \in A$，$\mu \in [0,1]$。$I: O \times A \rightarrow [0,1]$.

定义 2. 窗口。对于模糊形式背景中的属性，选取两个阈值 w_l 和 w_h，满足 $0 \le w_l \le w_h \le 1$。w_l 和 w_h 构成窗口，w_l 和 w_h 分别称为窗口的下沿和上沿。

定义 3. 映射 f 和映射 g。在模糊形式背景 F=(*O*,*A*,I)中，O⊆*O*，A⊆*A*，在 O 和 A 之间可以定义两个映射 f 和 g：

$f(O)=\{a|\forall o\in O, w_l \le I(o,a)\le w_h\};$ (3)

$g(A)=\{o|\forall a\in A, w_l \le I(o,a)\le w_h\}.$ (4)

定义 4. 模糊参数σ和λ。对于对象集合O⊆*O*和属性集合A⊆*A*，其中A=f(O)，o∈O，a∈A，|O|和|A|分别是集合 O 和集合 A 的势，如果|O|和|A|都不为 0，则

$$\sigma_a=\frac{1}{|O|}\sum_{o\in O} I(o,a); \quad (5)$$

$\sigma=\sum_{a\in A}(\sigma_a/a)$，本式中的∑为模糊逻辑中的符号。 (6)

$$\lambda_a=\sqrt{\frac{\sum_{o\in O}(I(o,a)-\sigma_a)^2}{|O|}}; \quad (7)$$

$$\lambda=\frac{1}{|A|}\sum_{a\in A}\lambda_a. \quad (8)$$

定义 5. 模糊概念。如果对象集合O⊆*O*和属性集合A⊆*A*满足O=g(A)且A=f(O)，则C=(O,A,σ,λ)被称为模糊形式背景 F 下的一个模糊概念。O和A分别称为模糊概念C的外延和内涵，σ和λ依据定义 4 计算。

定义 6. 模糊概念格。F 的所有模糊概念的集合记为 CS(F)。CS(F)上的结构是一种偏序关系，可以简单地定义为：如果$O_1\subseteq O_2$，则$(O_1,D_1)\le(O_2,D_2)$。通过这样的关系得到的有序集CS(F)=(CS(F),≤)是一个格，称作模糊形式背景 F 的模糊概念格。

2.3 从基于 FCA 的程序聚类到基于模糊 FCA 的程序聚类

在传统 FCA 中，概念格的构造是基于对象和属性之间的单值关系的。然而，在一些环境里，模糊的属性拥有关系可能存在并且会影响到聚类的结果。在基于 FCA 的关键字分析方法中，业务操作中的一些关键词（比如“wholesale”、“retail”）以及其他的业务概念（比如“book”、“order”）可能会出现在不同的源代码文件中，而关键字在不同的源代码文件中的频率分布可能存在较大的差异。如果我们在分析文件时采用基于单值关系的概念分析，就会忽视频率上的差异而只是关心一个关键字在了一个特定的源代码文件中出现与否。

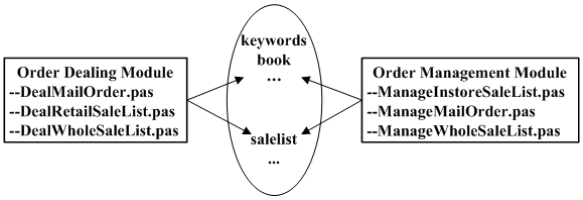


Figure.1. Two modules both involving keywords “book” and “salelist”

图. 1. 两个均涉及关键字“book” 和“salelist”的模块

举个例子，图 1 中有两个程序模块，订单生成模块（order dealing module）和订单管理模块（order management module），它们是在第四部分中进行实验的书店管理信息系统中的一部分。两个模块都由一些源代码文件组成，并且都涉及了某些相似的词汇，比如“salelist”和“book”（见于图 2）。这两个模块并不能通过传统的形式概念分析来进行区分，因为该方法仅仅考虑的是相关的业务词汇是否出现。从业务需求的角度来看，订单生成模块是根据客户的要求来生成图书订单，这个过程涉及一系列对于图书的操作并且在最后生成一张订单；而订单管理模块是用来对已经生成的订单进行管理，所以模块中的所有操作都与订单有关，而图书只有在必要的时候才被间接地涉及到。这两种特征与我们从词汇的频率分布中所观察到的现象是一致的：在前一个模块中，“book”的频率是相对较高的而“salelist”的频率是相对较低的；而在后一个模块中，“book”的频率是相对较低的而“salelist”的频率是相对较高的，和前一个模块正好相反。

所以，我们认为这里的词汇频率从某种意义上来说就是一种模糊信息。利用在源代码文件以及业务词汇之间的关系存在模糊性可以改善概念格及其所蕴涵的信息以帮助程序理解。

3 基于模糊 FCA 的程序聚类

3.1 方法步骤

图 2 展示了我们基于模糊 FCA 的程序聚类方法的过程。以下是五个主要的步骤：

1） **关键字筛选：**从源代码中抽取关键字供用户进行筛选。
2） **模糊信息收集：**先自动地计算特定关键字在源代码中的出现次数和出现频率，再计算特定源代码对应于特定关键字的模糊值。
3） **窗口阈值设置：**用户根据模糊值的分布统计图来指定各关键字的窗口阈值。

4） **模糊概念格构造：**对源代码和关键字做模糊形式概念分析得到模糊概念格。

5） **模糊参数评估：**对模糊参数σ和λ做评价，如有需要则调整窗口阈值并重复步骤 4。

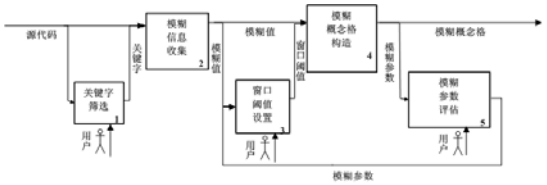


Figure 2. Process of the fuzzy-FCA-based program clustering

图 2. 基于模糊 FCA 的程序聚类过程

在关键词筛选过程，我们通过正则表达式从源代码文件中抽取关键字备选列表供用户进行选择，其中忽略那些与编程语言相关的关键字以及诸如“a”、“the”以及“by”等虚词。

窗口阈值的设置以及模糊参数的评估对于所构造的模糊概念格有重要影响，评估结果可能导致再一次设置窗口阈值。模糊概念格的构造是一个自动的过程，我们将在 3.3 节中讨论构造算法。

3.2 模糊信息收集以及窗口阈值设置

在进行信息模糊化时，我们先要得到包括关键字在源代码中出现次数、出现频率等信息，再将这些信息通过隶属度函数来进行模糊化。

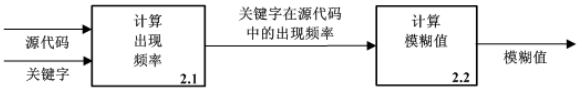


Figure 3. Two steps to collect fuzzy information

图 3. 模糊信息收集的两个步骤

我们假设当前所分析的源代码文件为 F，当前所分析的关键字为 K。

工具首先通过正则表达式来计算 F 中的所有单词数，记为 wordsNo(F)；接下来，工具通过浏览 F 获得 K 在其中的出现次数，记为 occur(K in F)。这样我们就可以推算出 K 在 F 中的出现频率，记为 freq(K in F)：

$$freq(K \text{ in } F)=occur(K \text{ in } F)/wordsNo(F). \quad (9)$$

我们可以这样认为：如果关键字在源代码文件中出现的频率很高，那么源代码文件与关键字一定存在紧密的关联；同样的，如果关键字在源代码文件中出现的频率很低，那么源代码文件与关键字之间的关联一定不那么紧密；介于两者之间的情况，可以说源代码文件与关键字存在一定的关联。

按照上述认识来定义工具中使用的隶属度函数 I(o,a)=μ，我们可以认为源代码文件 F 就是公式中的 o，关键字 K 就是公式中的 a。当工具分析到关键字 K_i 和源代码文件 F_j (j=1,…,n)时，首先遍历 K_i 在所有 F_j 中的出现频率，得到最高的一个，记为 maxFreq(K_i)：

$$maxFreq(K_i)=\max\{freq(K_i \text{ in } F_1),\ldots,freq(K_i \text{ in } F_n)\} \quad (10)$$

隶属度函数 $I(K_i, F_j)=\mu$ 则定义为：

$$I(K_i, F_j)= freq(K_i \text{ in } F_j)/maxFreq(K_i) \quad (11)$$

我们将得到的每个关键字的模糊值分布统计图呈现给用户，由用户来指定关键字的窗口阈值。窗口上下沿的选取准则应当是尽可能地将与关键字相关程度偏差较大的源代码文件分离，也就是每个关键字都可以有不止一对阈值，而每个窗口体现的则是与关键字的一定程度的相关，其目的是为了最终得到的模糊概念格中的每个概念都是高内聚、低偏离的。每次推导概念格后所得到的模糊参数都可以帮助用户更加合理、准确地指定窗口阈值，得到最为理想的模糊概念格。

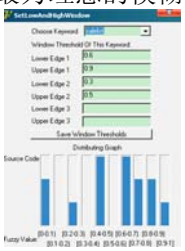


Figure 4. Distribution graph of fuzzy vale for keyword “salelist”

图 4. 关键字“salelist”的模糊值分布图

比如，在图 4 中，用户为关键字“salelist”首次指定窗口阈值的时候可能会简单的选择{w_l =0.4, w_h =1.0}；而在对模糊参数进行评估发现 λ 较大，表示该概念较为离散之后，用户应该就会为该关键字指定两个窗口，对应的阈值分别为：{w_l =0.4, w_h =0.5}以及{w_l =0.6, w_h =0.9}。

3.3 模糊概念格构造以及模糊参数评估

在本节，我们介绍述所提出的在模糊形式背景上的模糊概念格构造算法，本算法能够保证所构造的概念格的完备性和无冗余。概念格的经典构造算法主要分成两类，批处理算法和渐进式算法。批处理算法容易理解和实现，但是十分耗时。渐进式算法在找到一个合理有效的方法来在插入一个概念节点时与概念格中已有的其他节点进行求交运算上是相对较难的。我们的算法结合了两类方法的优势，使用了概念格的层次并且进行动态式的渐进插入，具体描述如下：

1.概念格 CS(F)初始化为空

2.将模糊形式背景 F 中的所有对象及其属性集合$(o_1,f\{o_1\})$, $(o_2,f\{o_2\})$, …, $(o_n,f\{o_n\})$作为初始概念 $C_1(O_1,A_1)$, $C_2(O_2,A_2)$, …, $C_n(O_n,A_n)$加入概念格 CS(F)中；从初始概念中求取内涵集合的势的最大值作为概念格的最底层层次，即记 L_{max}：

$$L_{max}=Max(|A_1|,|A_2|,\ldots,|A_n|). \quad (12)$$

图 5 展示了带有层次信息$(1,2,\ldots, L_{max})$的概念格。

3.从概念格的最底层 L_{max} 层到第 1 层进行遍历，在每层中查找内涵相同的概念，即查找是否存在概念 C(O,A)与概念 $C^*(O^*,A^*)$，且 $A^*=A$。如果存在这样的概念 C 和概念 C^*，则合并成同一个概念 $C'(O',A')$，其中 $O'=O\cup O^*$，$A'=A=A^*$，并从概念格 CS(F)中删除原节点 C 和 C^*。

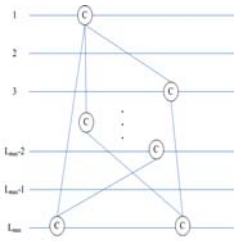


Figure 5. Concept lattice with level information

图 5. 带有层次信息的概念格

4.从概念格的最底层 L_{max} 层到第 1 层进行遍历，若当前层为第 i 层，当前扫描到的概念为 C(O,A)。

4.1. 在第 i-1 层至第 1 层中查找是否存在内涵为 A 的子集的概念，即查找是否存在 $C^*(O^*,A^*)$，且 $A^* \subset A$。如果存在这样的概念 C^*，则将概念节点 C^*更新为 $C^*(O\cup O^*,A^*)$，并且将概念 C 和概念 C^*进行连接，再进行剪枝算法来消除可能的冗余连接。

4.2. 在第 i 层至第 1 层中查找是否存在与内涵 A 有内涵交集的概念，即查找是否存在 $C^*(O^*,A^*)$，且 $A^*\cap A=A'\neq\emptyset$。如果存在这样的概念 C^*，则先计算内涵交集 A'的势 $|A'|=|A^*\cap A|$，再在第$|A'|$层中去查找是否存在概念节点 $C'(O',A')$。如果存在这样的概念 C'，则将概念 C'更新为 $C'(O'\cup O^*\cup O,A')$，并且将概念 C 以及概念 C^*和概念 C'进行连接。如果在第$|A'|$层不存在这样的概念 C'，则在第$|A'|$层中新建一个概念节点 $C'(O^*\cup O,A')$，并且将概念 C 以及概念 C^*和概念 C'进行连接。如果有概念和概念进行连接，则进行剪枝算法来消除可能的冗余连接。

5.将概念格 CS(F)中所有没有父节点的概念与根节点相连。

6. 按照定义 4 中的公式，计算概念格 CS(F)中每个概念的模糊参数 σ 和 λ。

消除可能的冗余连接的剪枝算法如下：

若当前要连接的两个概念节点为 C(O,A)和 $C^*(O^*,A^*)$，其中 C 为子节点，C^*为父节点，如果 C^*已经是 C 的祖先则不进行连接；否则连接 C 和 C^*，并且如果存在概念节点 $C'(O',A')$既是概念 C 的子节点，又是概念 C^*的子节点，则删除从概念节点 C'到概念节点 C^*的父子连接关系。

通过模糊参数可以帮助用户对聚类结果有一定程度的理解和评估。首先是模糊参数 σ，它代表了一个概念所具有的内涵的程度。比如某概念 $C(\{o_1,o_2,o_3\},\{a_1,a_2\}),\sigma=0.4/a_1+0.85/a_2$，这说明了这个概念与属性 a_1 具有一定的关联（隶属度为 0.4），而与属性 a_2 具有很大的关联（隶属度为 0.85）。从程序聚类的角度来说，由一组源代码所构成的概念主要是为了实现与关键字 a_2 相关的逻辑，而同时也包含了一部分对于关键字 a_1 相关逻辑的实现。接着是模糊参数 λ，代表了一个概念对于各个内涵的隶属度偏离值的平均程度，它体现了这个概念的离散程度。比如某概念 $C(\{o_1,o_2\},\{a_1,a_2,a_3\})$，λ=0.03，这说明了这个概念对于其内涵的偏离程度较小，其外延 o_1 和 o_2 能够较好地表现出其所拥有的内涵 a_1、a_2 和 a_3，也说明了对于属性 a_1、a_2 和 a_3 的窗口阈值的指定是比较成功的；如果某概念 $C(\{o_1,o_2,o_3\},\{a_1,a_2,a_4\})$，λ=0.20，这说明了这个概念对于其内涵的偏离程度较大，其外延 o_1、o_2 和 o_3 并不能合理地表现出其所拥有的内涵 a_1、a_2 和 a_4。从程序聚类的角度来说，这一组源代码中可能存在某个源代码与某个关键字相关的逻辑实现的关联程度与其他源代码与该关键字相关的逻辑实现的关联程度偏差较大。这也就说明了对于属性 a_1、a_2 和 a_4 的窗口阈值的指定存在可能的偏差，需要用户返回第三步经过重新查看模糊值分布统计图来更加合理地指定窗口阈值，就像在 3.2 节中最后提到的那样。

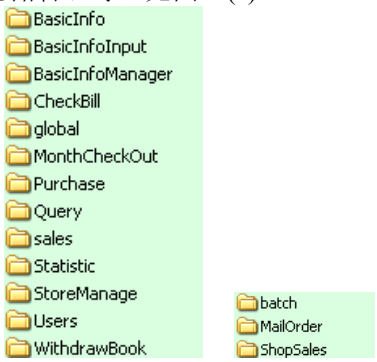
4 实验

我们实现了基于模糊 FCA 程序聚类方法，并对一个商业软件—书店管理信息系统（简称书店系统）进行了实验分析。实验结果证明了方法的有效性，也得到了系统开发人员的认可。

4.1 实验设计及结果

我们将系统的每个独立的文件夹中的所有源代码文件作为分析目标。将这些文件夹作为工具的输入来得到对应的模糊概念格。书店系统中的

主要业务模块包括：账单管理、进书管理、销售管理、仓储管理等，见图 6 (a)。



(a) Major modules of bookstore system (b) Sub modules of sales management

Figure 6. Major/sub modules of bookstore system

(a) 书店系统的主要模块 (b) 销售管理的子模块

图 6. 书店管理的主要模块和子模块

其中的销售管理是在我们比较关心的。销售管理模块中包括了门店销售、批发销售、邮购销售等分支业务模块，见图 6 (b)。表 1 列出了源代码文件与对象之间的映射关系。

Table 1. Projection between source code and object

表 1. 源代码与对象之间的映射

文件名	对象
DealMailOrder.pas	o_1
DealRetailSaleList.pas	o_2
DealWholeSaleList.pas	o_3
ManageInstoreSaleList.pas	o_4
ManageMailOrder.pas	o_5
ManageWholeSaleList.pas	o_6

首先筛选关键字，我们得到了在这个业务模块中所关心的关键字，这些关键字与属性之间的映射关系列在表 2 中。阈值的指定则不是一个一次性的过程，应当根据得到的模糊概念格上的模糊参数进行调整；每个属性也可以拥有不止一对阈值来表现其在隶属度上的差异。我们为上述关键字指定的最终阈值列于表 3。属性 a_1、a_6、a_7 和 a_8 的模糊值分布图上表现出了较大的差异，所以我们各设置了两组窗口阈值，这些属性则相应地分离为属性 $a_{1\text{-}1}$、$a_{1\text{-}2}$、$a_{6\text{-}1}$、$a_{6\text{-}2}$、$a_{7\text{-}1}$、$a_{7\text{-}2}$ 等。

工具输出的模糊概念格如图 7 所示。概念的模糊参数列于表 4。

Table 2. Projection between keyword and attribute

表 2. 关键字与属性之间的映射

关键字	属性
book	a_1
client	a_2
mail	a_3
retail	a_4
wholesale	a_5
salelist	a_6
order	a_7
store	a_8
deal	a_9
manage	a_{10}

Table 3. Window thresholds for all the keywords in Table 2

表 3. 表 2 中属性的窗口阈值

属性	上沿	下沿
$a_{1\text{-}1}$	1	0.6
$a_{1\text{-}2}$	0.3	0.1
a_2	0.3	0.1
a_3	0.8	0.5
a_4	0.9	0.6
a_5	0.9	0.5
$a_{6\text{-}1}$	1	0.6
$a_{6\text{-}2}$	0.3	0.1
$a_{7\text{-}1}$	0.8	0.6
$a_{7\text{-}2}$	0.4	0.2
$a_{8\text{-}1}$	0.8	0.5
$a_{8\text{-}2}$	0.3	0.1
a_9	1	0.5
a_{10}	1	0.5

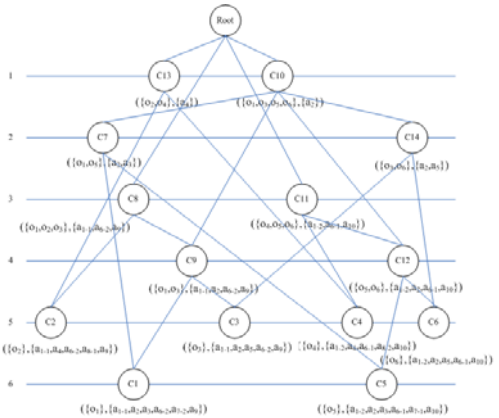


Figure 7. Fuzzy concept lattice of sales module

图 7. 销售管理模块的模糊概念格

如果我们把表 2 中所有属性的窗口阈值都设置成{w_l =0, w_h =1.0}，我们得到一个基于传统 FCA 的概念格。图 8 展示了这样情况下工具输出的概念格。

Table 4. Fuzzy Parameters for concepts in Figure 8

表 4. 图 8 中概念的模糊参数

概念	模糊参数 σ	模糊参数 λ
C1	$0.8/a_{1\text{-}1}+0.2/a_2+0.8/a_3+0.2/a_{6\text{-}2}+0,2/a_{7\text{-}2}+0.8/a_9$	0
C2	$0.9/a_{1\text{-}1}+0.9/a_4+0.3/a_{6\text{-}2}+0.7/a_{8\text{-}1}+0.8/a_9$	0
C3	$0.9/a_{1\text{-}1}+0.3/a_2+1/a_5+0.3/a_{6\text{-}2}+0.9/a_9$	0
C4	$0.3/a_{1\text{-}2}+0.8/a_4+0.9/a_{6\text{-}1}+0.2/a_{8\text{-}2}+0.8/a_{10}$	0
C5	$0.2/a_{1\text{-}2}+0.3/a_2+1/a_3+0.8/a_{6\text{-}1}+0.9/a_{7\text{-}1}+0.9/a_{10}$	0
C6	$0.3/a_{1\text{-}2}+0.3/a_2+0.8/a_5+1/a_{6\text{-}1}+1/a_{10}$	0
C7	$0.3/a_2+0.9/a_3$	0.07
C8	$0.9/a_{1\text{-}1}+0.3/a_{6\text{-}2}+0.9/a_9$	0.05
C9	$0.9/a_{1\text{-}1}+0.3/a_2+0.3/a_{6\text{-}2}+0.9/a_9$	0.07
C10	$0.3/a_2$	0.01
C11	$0.3/a_{1\text{-}2}+0.9/a_{6\text{-}1}+0.9/a_{10}$	0.08
C12	$0.3/a_{1\text{-}2}+0.3/a_2+0.9/a_{6\text{-}1}+1/a_{10}$	0.04
C13	$0.9/a_4$	0.05
C14	$0.3/a_2+0.9/a_5$	0.05

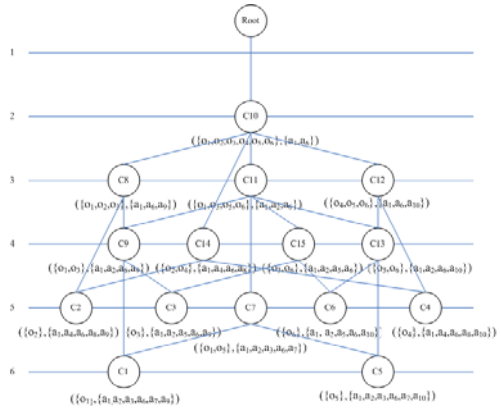


Figure 8. Traditional concept lattice of sales module
图 8. 销售管理模块的传统概念格

4.2 实验结果分析

图 7 中的模糊概念格蕴涵了书店销售模块中程序源代码内部和相互之间存在的丰富信息。比如：从概念 13 ($\{o_2,o_4\},\{a_4\}$)中可以知道源代码 FormDealRetailSaleList.pas 和 FormManageInstoreSaleList.pas 处理的是与零售（关键字“retail”）相关的部分业务；又从同层的概念 10 ($\{o_1,o_3,o_5,o_6\},\{a_2\}$)中可以知道代码 FormDealMailOrder.pas 、FormDealWholeSaleList.pas 、FormManageMailOrder.pas 和 FormManageWholeSaleList.pas 中都涉及与客户（关键字“client”）相关的信息，暗示了零售业务模块与客户没有关联，用户可以在不注册为客户的情况下在门店通过零售的方式购买图书。

在概念层次为 4 的概念 12($\{o_5,o_6\},\{a_{1\text{-}2},a_2,a_{6\text{-}1},a_{10}\}$)中，源代码 ManageMailOrder.pas 和 ManageWholeSaleList.pas 共享属性 $a_{1\text{-}2}$、a_2、$a_{6\text{-}1}$ 和 a_{10}。结合表 3 分析可知，这两个源代码对客户（关键字“client”）信息进行管理（关键字“manage”），其实涉及较少的图书(关键字“book” $a_{1\text{-}2}$)信息以及较多的订单(关键字“salelist” $a_{6\text{-}1}$）信息。图 8 中的概念 13($\{o_5,o_6\},\{a_1,a_2,a_6,a_{10}\}$)给出了相似的信息。我们只有通过模糊参数 σ 才能了解源代码与关键字之间的关联程度。

通过对比概念 7 ($\{o_1,o_5\},\{a_2,a_3\}$)和概念 14 ($\{o_3,o_6\},\{a_2,a_5\}$)与概念 10 ($\{o_1,o_3,o_5,o_6\},\{a_2\}$)的父子关系可以发现虽然四个源代码都涉及客户（关键字“client”），但是概念 7 中的外延：代码 FormDealMailOrder.pas 和 FormManageMailOrder.pas 主要处理的是与邮购（关键字“mail”）相关的业务逻辑，而概念 14 中的外延：代码 FormDealWholeSaleList.pas 和 FormManageWholeSaleList.pas 主要处理的则是与批发（关键字“wholesale”）相关的业务逻辑。

再进一步，对比同在第三层上的概念 8 ($\{o_1,o_2,o_3\},\{a_{1\text{-}1},a_{6\text{-}2},a_9\}$) 以及概念 11 ($\{o_4,o_5,o_6\},\{a_{1\text{-}2},a_{6\text{-}1},a_{10}\}$)就可以知道在订单生成(关键字“deal”)模块中总是涉及较多对于图书(关键字“book” $a_{1\text{-}1}$)而较少对于订单（关键字“salelist” $a_{6\text{-}2}$）的处理，而在订单管理(关键字“manage”)模块中则正好相反，总是涉及更多的订单相关的处理而更少的图书相关的操作。如果使用传统 FCA，从图 8 的概念格中可以看到，这些源代码都被聚类到了同一个概念 10 ($\{o_1,o_3,o_5,o_6\},\{a_2\}$)中，这样的结果隐藏了上述的业务逻辑信息。

上例仅是整个书店系统的核心业务模块之一，通过对所有的模糊概念格进行学习，开发人员可以对目标软件系统有一个初步的认识和理解；系统的前任开发人员也认为此模糊概念格所表达的聚类结果是基本正确且有指导意义的。

4.3 讨论

文本的程序聚类方法看起来并不是一种严格意义上的程序聚类，严格的程序聚类应当将软件系统划分为多个相互独立的软件子系统。但是，

本文是从程序源代码文件的角度进行划分，根据关键字对源代码进行概念的提取，一个概念中的外延共享了这个概念中的内涵，所以也可以认为是一种非严格意义上的聚类。并且由于本文所构造的概念格具有显式的概念层次，所以也可以认为是从不同的维度上对软件系统源代码进行聚类活动。[17]中也提到了层次的概念，而其是从功能需求的层次角度来进行程序聚类的。

有一个关于关模糊参数 λ 容忍度的问题。通过实验数据我们发现一般概念格中的概念的模糊参数 λ 大多都不为 0，也就是说程序聚类的结果表明一般软件系统中的概念或多或少的存在一定的离散性，要得到十分内聚的概念是相对困难的。在实际地对一个软件系统应用本工具的时候，可以指定一个模糊参数 λ 的容忍度 S(t)，当一个概念的 λ 小于 S(t)时，认为概念是一个可接受的概念，否则需要重新指定窗口阈值来再次推导概念格。

对本实验进行评估的结论与之前的假设也保持了一致性，也就是说，模糊参数 λ 较小的概念是对于软件源代码的一个较好划分，被聚集在其中的外延对于其属性的表现出了良好的内聚性，往往是为一些特定的软件需求所服务的。而为了得到更好的模糊参数 λ，在使用工具的过程中就需要反复调整窗口阈值，这是一个手工设置、需要经验的过程，我们期望在今后的研究中对这个过程提供一些可能的反馈性帮助。

5 相关工作

程序聚类是一种用来帮助人们理解软件系统的十分有意义的方法。P.Andritsos 等人[7]提出了一种将信息理论技术应用于软件聚类的方法，这种方法允许对反映出不同属性重要性的配置进行加权统计。N.Anquetil 等人[8]讨论了一种使用更为非形式化的信息源——文件名来抽取概念（缩写）。寻找缩写一般需要识别在名字中所涉及到的概念的领域知识以及在缩写形式中识别出这种概念的直觉。Wei Zhao 等人[1]提出了一种由需求指导的动态方法来解决软件聚类的问题，他们的目标是为大型的、复杂的系统提供有逻辑的以及高层次的分解。

FCA 也被广泛采用来进行程序聚类。Thomas Eisenbarth 等人[9]通过 FCA 得到特征与计算单元之间的映射。他们将计算单元作为对象，将场景作为属性，如果在一次场景的执行中调用了某个计算单元，那么认为两者之间就是有关联的。Uri Dekel 等人[10]通过 FCA 来对类的接口进行可视化。概念格的构建是基于方法和域之间访问的二元关系，这样的概念格可以用来为进行自动化的特征分类提供服务。M.Ceccato 等人[11]使用 FCA 来聚类程序，进而进行方面挖掘。在他们的标识符分析中，FCA 算法把拥有相同标识符的实体划分到一个小组中去。当一个小组中包含一定数量的对象，这样的小组被认为是潜在方面的一个种子。

事实上，在包括数据挖掘等其他领域，使用模糊 FCA 已经成为一种流行的趋势。T.T.Quan 等人[12]提出了一种基于模糊 FCA 的方法来进行概念聚类，这种方法可以帮助人们在不确定的数据上自动的建立一个概念层次。他们[14]还将在[12]中提出的方法应用于一个基于引用的文档恢复，其中可以将模糊查询进行形式化来处理。

源代码中关键字的出现频率的重要性也在信息抽取领域中的基于 LSI(Latent Semantic Indexing)的方法中得以体现。LSI 方法是一种基于向量空间模型的方法，用来表示单词和段落在应用中所表现出来的含义的每个方面。J.I.Maletic 等人[15]描述了将 LSA(Latent Semantic Analysis)应用于程序源代码和相关文档中所得到的初步结果。A.Marcus 等人[13]通过使用 LSI 来发现系统文档和程序源代码之间可追踪的链接。基于 LSI 的方法与我们的方法的主要区别在于，前者的相似性比较是在整个词库上进行的，也就是说向量空间中所有的关键字都要进行比较；而后者可以在不同的概念格层次上通过一些关键词中的部分相似性来聚类概念，也就是说概念内涵的规模可以不同，通过高层次上的概念可以获得软件系统的高层需求。

6 总结与展望

本文提出了一个基于模糊 FCA 的程序聚类方法，在传统 FCA 的基础上使用了模糊信息来表述源代码与关键字之间的模糊和不确定关系。我们开发了一个半自动化的工具来实现模糊概念格构造算法，该算法结合了传统概念格构造算法的优势且保证了所构造的格的完备性与无冗余性。我们将工具应用到一个商用书店管理系统中。实验结果表明了该方法可以获得带有模糊特征的聚类结果来帮助进行较好的程序理解，这一点也得到了系统的前任开发人员的认可。通过模糊概念格与传统概念格的比较，也初步地揭示了模糊 FCA 较于传统 FCA 的优势。

进一步的研究工作包括通过寻找关键字和特征之间的关联，间接地建立程序源代码与特征之间的关联以进行特征定位；也计划进一步将研究对象的粒度降低到诸如类或函数的程度，以得到更加细粒度的程序聚类结果。

参考文献

1 Wei Zhao, Lu Zhang, Hong Mei and Jiasu Sun, “Requirements Guided Dynamic Software Clustering”[C], Proc. Of 21st IEEE International Conference on Software Maintenance, 2005

2 B.S. Mitchell and S. Mancoridis, “Comparing the Decompositions Produced by Software Clustering Algorithms using Similarity Measurements”[C], Proc. Of 17th IEEE International Conference on Software Maintenance, pp.744-753, Nov. 2001

3 Y.Chiricota, F.Jourdan, and G.Melancon, “Software Components Capture using Graph Clustering”[C], Proc. Of 11th IEEE International Workshop on Program Comprehension, pp.217-226, May 2003

4 A.van Deursen and T.Kuipers, “Identifying objects using cluster and concept analysis”[C], Proc. Of 21st IEEE International Conference on Software Engineering, pp.246-255, May 1999

5 C.Lindig and G.Snelting, “Assessing modular structure of legacy code based on mathematical concept analysis”[C], Proc. Of 19th IEEE International Conference on Software Engineering, pp.349-369, May 1997

6 S.Mancoridis, B.S. Mitchell, C.Rorres, Y.Chen, and E.R. Gansner, “Using Automatic Clustering to Produce High Level System Organizations of Source Code”[C], Proc. Of 6th IEEE International Workshop on Program Comprehension, pp.45-52, June 1998

7 P.Andritsos and V.Tzerpos, “Information-Theoretic Software Clusteing”[C], IEEE Transactions on Software Engineering, 31(2), pp.150-165, Feb.2005

8 N.Anquetil and T.Lethbridge, “Extracting Concepts from File Names: A New File Clustering Criterion”[C], Proc. Of the 20th IEEE International Conference on Software Engineering, pp.84-93, April 1998

9 T.Eisenbarth, R.Koschke and D.Simon , “Locating features in source code”[C], Software Engineering, IEEE Transactions on Volume 29, Issue 3, pp.210-224, March 2003

10 Uri Dekel, Yossi Gil, “Visualizing Class Interfaces with Formal Concept Analysis”[C], Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications, Oct. 2003

11 M.Ceccato, M.Marin, K.Mens, L.Moonen, P.Tonella and T.Tourwe, “A qualitative comparison of three aspect mining techniques”[C], Program Comprehension, 2005. IWPC 2005. Proc. Of 13th International Workshop, pp.13-22, on 15-16 May 2005

12 T.T.Quan, S.C.Hui and T.H.Cao, “A Fuzzy FCA-based Approach to Conceptual Clustering for Automatic Generation of Concept Hierarchy on Uncertainty Data”[C], Proc. Of CLA conference, Ostrava, 2004

13 A.Marcus and J.I.Maletic, “Recovering documentation-to-source-code traceability links using latent semantic indexing”[C], Software Engineering, 2003. Proc. Of 25 th International Conference, pp.125-135, on 3-10 May 2003

14 T.T.Quan, S.C.Hui and T.H.Cao, “A fuzzy FCA-based approach for citation-based document retrieval”[C], Cybernetics and Intelligent Systems, 2004 IEEE Conference on Volume 1, pp.578-583 vol.1, 1-3 Dec. 2004

15 J.I.Maletic, and N.Valluri, “Automatic software clustering via Latent Semantic Analysis”[C], Automated Software Engineering, 1999. 14th IEEE International Conference, pp.251-254, on 12-15 Oct. 1999

16 YAO Y Y, CHEN Y H, “Rough set approximations in formal concept analysis”[C]. Proceedings of 2004 Annual Meeting of the North American Fuzzy Information Processing Society(NAFIPS2004), 2004. pp.73-78

17 ZHAO Wei, ZHANG Lu, MEI Hong, Sun Jia-Su, “A Functional Requirement Based Hierarchical Agglomerative Approach to Program Clustering”[J], Journal of Software, Vol.17, No.8, August 2006, pp.1661−1668

附中文参考文献:

17 赵 伟，张 路，梅 宏，孙家骕，“一种基于功能需求层次凝聚的程序聚类方法”[J]，软件学报，2006，17（8），1661-1668。

作者简介：

Xu Jiaqing, born in 1984. Master candidate. His main research interest is program comprehension and reverse engineering.

许佳卿，男，1984 年生，复旦大学计算机科学技术学院硕士生，主要研究方向为程序理解及逆向工程。

Xin Peng, born in 1979. Ph.D. and assistant professor. Member of China Computer Federation. His current research interests include Software Product Line, Software Component and Architecture, Software Maintenance and Software Reengineering.
彭鑫，男，1979 年生，博士，CCF 会员，复旦大学计算机科学技术学院讲师。目前主要研究方向包括软件产品线、软件构件与体系结构、软件维护与再工程。

Zhao Wenyun, born in 1964. Professor and Ph.D. supervisor. Senior member of China Computer Federation. His main research interests include Software Reuse, Software Component and Architecture.
赵文耘，男，1964 年生，CCF 高级会员，复旦大学计算机科学技术学院教授、博导。目前主要研究方向包括软件复用、软件构件与体系结构。

英文的本文介绍：

This work is supported by National Natural Science Foundation of China under Grant No. 60703092 (Software-Product-Line oriented reengineering), and National High Technology Development 863 Program of China under Grant No. 2007AA01Z125 (Feature-oriented and incremental software product line construction). These research works try to provide approaches and tools for incremental software product line development with software reverse engineering, refactoring and evolution management. Problem clustering in the projects is to help abstract high-level requirement and design structures from legacy systems and then further analyze the commonality and variability among existing variant applications in a domain.

Formal Concept Analysis (FCA) has been widely adopted in text-analysis-based program clustering. However, existing FCA-based methods usually use one-valued attributes (with or without) in analysis, and cannot deal with the fuzzy information in the programs. In this paper, a text-analysis-based program clustering method using fuzzy FCA is proposed. The method includes a process of fuzzy attribute computation and a construction algorithm for fuzzy concept lattice, which can ensure a complete and non-redundant lattice. A semi-automatic analysis tool has been developed for the method and applied in a case study of a commercial bookstore management system. Results of the experiment show that the method can help to obtain clusters with sound fuzzy features for program comprehension, which are also recognized by former developers of the system.