概念格合并原理与算法

智慧来[1,2],智东杰[2],刘宗田[1]
(1.上海大学计算机工程与科学学院,上海 200072; 2.河南理工大学计算机科学与技术学院,河南焦作 454150)

摘 要: 子形式背景可以进行纵向或横向合并得到一个新的形式背景,相反地,这个形式背景可以进行纵向或横向的拆分得到子形式背景.相应地,子概念格也可以进行合并,得到一个新的概念格.概念格合并无论是纵向合并还是横向合并,合并都不改变原有概念之间业已存在的父子关系.根据这个事实,结合概念格合并的定义,设计了概念格合并算法.此算法充分利用原有概念格的结构,在原有概念格基础上对部分节点进行调整得到合并结果.实验和分析均表明与将一概念格中的概念插入到另一个概念格的算法相比,此算法效率明显提高,适合概念格的合并运算.

关键词: 概念格; 子格; 纵向合并; 横向合并
中图分类号: TP18 **文献标识码:** A **文章编号:** 0372-2112 (2010) 02-0455-05

Theory and Algorithm of Concept Lattice Union

ZHI Hui-lai[1,2], ZHI Dong-jie[2], LIU Zong-tian[1]
(1. *School of Computer Engineering and Science, Shanghai University, Shanghai* 200072, *China*;
2. *School of Computer Science and Technology, Henan Polytechnic University, Jiaozuo, Henan* 454150, *China*)

Abstract: Vertical union and horizontal union can be carried on sub-context, vice versa; vertical split and horizontal split can be carried on a context. Accordingly, sub-lattice of the sub-context can be united to create a new lattice which is affiliate to the context created by the sub-contexts. Both vertical union and horizontal union don't change the father-son relationships that already exist in the sub-lattice. Based on this fact and the definition of lattice union, a union algorithm of concept lattices is put forward. This algorithm uses the sub-lattices' structure, only a few adjustment need to be made. Experiments and analysis show that compared with inserting one lattice's concepts into another lattice one by one, the efficiency of this algorithm is improved.

Key words: concept lattice; sub-lattice; vertical union; horizontal union

1 引言

随着处理的形式背景的增大,概念格的时空复杂度也会随着急剧增大.研究采用新的方法和手段来构造概念格,是形式概念分析应用于大型复杂数据系统的前提.而概念格的分布处理[1]思想就是通过形式背景的拆分,形成分布存储的多个子背景,然后构造相应的子概念格,再由子概念格的合并得到所需的概念格.这称为概念格分布处理模型或分布式概念格模型.

形式背景的拆分有横向和纵向之分.把一个形式背景拆分成多个子背景,也称作局部背景,其对应的概念格就可称为子概念格或部分格.形式背景所对应的概念格可通过各子概念格进行合并来实现.这种概念格的构造方法采用的是一种分治策略,或者说是概念格的分布处理模型或框架.

在采用分治策略构造概念格的研究中,P. Valtchev 等分别在文献[2]和文献[3]提出了叠置格和并置格两种概念格的构造方案.在文献[2,3]中,子格的构造采用的是一种递归的 Godin 算法[4],并在概念格的构造中引入了下覆盖概念.而在子格或局部格合并为完整格或全局格中,两种方案中都是引入了子格和完整格之间的两个映射函数来通过部分格概念计算全局格概念.

以并置格为例,设形式背景的属性被分割为 A_1 和 A_2,对于部分格中的两个概念(X_1, Y_1)和(X_2, Y_2),那么这两个映射函数 φ 和 ψ 就可定义为:设函数 $\varphi: GL \rightarrow PL$,是全局格概念到部分格概念的映射,那么:$\varphi(X, Y) = ((f(Y \cap A_1), Y \cap A_1), (f(Y \cap A_2), Y \cap A_2))$;同理,设函数 $\psi: PL \rightarrow GL$,,是部分格概念到全局格概念的映射,那么:$\psi((X_1, Y_1), (X_2, Y_2)) = (X_1 \cap X_2, g(X_1 \cap X_2))$.

有了上述的两个映射函数,就可以通过枚举两个子格概念之间的组合,计算出其对应的全局格概念.但是,在 P. Valtchev 的算法中,由于一个部分格的概念对另一部分格的概念是遍历计算,会产生较大的重复,从而影响了算法效率.

收稿日期:2009-03-11;修回日期:2009-09-10
基金项目:国家自然科学基金(No.60275022,60575035);上海大学创新基金(No.A.16-0108-08-002)

与此同时,文献[5]提出了概念的并与交原理,在此基础上文献[6,7]首先从形式背景的纵向、横向合并出发,定义了内涵独立和内涵一致的形式背景和概念格;还定义了内涵一致的形式背景、概念的横向加运算和概念格的横向并运算,并证明了横向合并的子形式背景的概念格和子背景所对应的子概念格的横向并是同构的,最后结合子概念格中概念间固有的泛化-特化关系,提出一种多概念格的横向合并算法来构造概念格.合并概念格的过程是其中一个概念格保持不变,将另一个概念格中的概念依次插入到保持不变的概念格中,插入过程中对概念格进行相应调整.

对比文献[2,3]和文献[6,7]可知,所谓叠置格就是通过形式背景的叠置所得到的概念格,它实际上是对应子格的纵向合并;并置格就是通过形式背景的并置所得到的概念格,它实际上是对应子格的横向合并.

根据概念格的对偶原理,概念的对象和属性具有同等的地位,只是论述的需要将两者区分开来.因此,有必要将两者统一论述,提出一个统一的概念格合并理论与实现方法;其次,文献[2,3,6,7]在合并概念格时都没有考虑如何利用原有概念格的结构,而是将概念格中的概念一个个地插入到另一个概念格中,损失了大量可以直接利用的信息.

2 概念格合并的基本概念与理论

R.Wille[8]提出的形式概念分析是以序理论和完备格理论为基础,依据数据库中提供的基本信息建立起的一种刻画对象与属性之间关系的数学结构,其核心数据结构是概念格.

定义 1 设 $K=(G,M,I)$是一个形式背景,$A\subseteq G,B\subseteq M$.如果 A、B 满足条件$f(A)=B,g(B)=A$,则称序对(A,B)为形式背景 K 的一个概念.A 称为概念(A,B)的外延,B 称为概念(A,B)的内涵.$L(G,M,I)$或 $L(K)$表示 K 中所有概念全体构成的集合,即:

$$L(G,M,I)=\{(A\times B)\in G\times M,f(A)=B,g(B)=A\}$$[8]

定义 2 设 $K=(G,M,I)$是一个形式背景,(A_1,B_1)、$(A_2,B_2)\in L(K)$,如果 $A_1\subseteq A_2$或 $B_1\supseteq B_2$,称(A_1,B_1)是(A_2,B_2)的子概念,记为$(A_1,B_1)\leqslant(A_2,B_2)$.显然 $L(K)$关于"$\leqslant$"构成一个格,称为概念格[8].

对于给定形式背景 $K=(G,M,I)$的两个概念(A_1,B_1)、(A_2,B_2),以下结论成立:

对于 $A_1,A_2\subseteq G$,如果 $A_1\subseteq A_2$,那么 $f(A_2)\subseteq f(A_1)$;对于 $B_1,B_2\subseteq M$,如果 $B_1\subseteq B_2$,那么 $g(B_2)\subseteq g(B_1)$.

定理 1(概念格基本定理) 概念格 $B(G,M,I)$是一个完全格,其上确界和下确界分别是

$$\bigwedge_{t\in T}(A_t,B_t)=\left(\bigcap_{t\in T}A_t,f\left(g\left(\bigcup_{t\in T}B_t\right)\right)\right),$$

$$\bigvee_{t\in T}(A_t,B_t)=\left(g\left(f\left(\bigcup_{t\in T}A_t\right)\right),\bigcap_{t\in T}B_t\right)$$[8].

定义 3 在形式背景 $K_1=(G_1,M_1,I_1)$和形式背景 $K_2=(G_2,M_2,I_2)$中,对于任意 $g\in G_1\cap G_2$ 和任意 $m\in M_1\cap M_2$ 满足 $gI_1m\Leftrightarrow gI_2m$,则称 K_1 和 K_2 是一致的,否则称 K_1 和 K_2 是不一致的[6,7].

定义 4 如果形式背景 $K_1=(G_1,M_1,I_1)$和 $K_2=(G_2,M_2,I_2)$是一致的,那么它们的合并是:$K_1\oplus K_2=(G_1\cup G_2,M_1\cup M_2,I_1\cup I_2)$,称为 K_1 和 K_2 的加运算.如果 $M_1=M_2$,称 $K_1\pm K_2=(G_1\cup G_2,M,I_1\cup I_2)$是两个形式背景的纵向合并[6,7].如果 $G_1=G_2$,称 $K_1\mp K_2=(G,M_1\cup M_2,I_1\cup I_2)$是两个形式背景的横向合并[6,7].

定理 2 如果 $K_1=(G_1,M_1,I_1)$、$K_2=(G_1,M_2,I_2)$、$K_3=(G_2,M_1,I_3)$、$K_4=(G_2,M_2,I_4)$是一致的,那么,$(K_1\mp K_2)\pm(K_3\mp K_4)=(K_1\pm K_3)\mp(K_2\pm K_4)$.

由定义 4 易知定理 2 成立,证明从略.

定义 5 如果 $L(K_1)$和 $L(K_2)$是一致形式背景 K_1 和 K_2 的概念格,则定义它们的加运算 $L(K_1)+L(K_2)$等于 L,L 满足:

(1)如果 $L(K_1)$中有概念 $c_1(A_1,B_1)$,$L(K_2)$中概念 $c_2(A_2,B_2)$,令 $c_3=(A_1\cup A_2,B_1\cap B_2)$,如果在 $L(K_1)$中的所有大于 c_1 的概念中不存在等于或小于 c_3 的概念,且在 $L(K_2)$中的大于 c_2 的所有概念中不存在等于或小于 c_3 的概念,则 $c_3\in L$;

(2)如果 $L(K_1)$中有概念 $c_1(A_1,B_1)$,$L(K_2)$中概念 $c_2(A_2,B_2)$,令 $c_3=(A_1\cap A_2,B_1\cup B_2)$,如果在 $L(K_1)$中的所有小于 c_1 的概念中不存在等于或大于 c_3 的概念,且在 $L(K_2)$中的小于 c_2 的所有概念中不存在等于或大于 c_3 的概念,则 $c_3\in L$.

(3)上述情况之外的概念不属于 L.

定理 3 如果 $L(K_1)$和 $L(K_2)$是一致形式背景 K_1 和 K_2 的概念格,那么 $L(K_1)+L(K_2)=L(K_1+K_2)$.

证明 定理的证明分为两个步骤:

(1)如果概念 c_3 由定义 5(1)生成,$C_1=(A_1,B_3\cup B_x)\in L(K_1)$和 $c_2=(A_2,B_3\cup B_y)\in L(K_2)$且满足 $A_1\cup A_2=A_3$,$B_x\cap B_y=\{\}\Leftrightarrow$在 K_1 中有 $f(A_1)=B_3\cup B_x$,$g(B_3)=A_1$,在 K_2 中有 $f(A_2)=B_3\cup B_y$,$g(B_3)=A_2\Leftrightarrow K_1+K_2$ 中有 $f(A_3)=B_3$,$g(B_3)=A_3\Leftrightarrow c_3\in L(K_1+K_2)$.

(2)由概念格的对偶性,同理可证,即:如果概念 c_3 由定义 5(2)生成,即:$c_1=(A_1\cup A_x,B_3)\in L(K_1)$和 $c_2=(A_2\cup A_y,B_3)\in L(K_2)$且满足 $A_1\cap A_2=A_3$,$A_x\cap A_y=\{\}\Leftrightarrow$在 K_1 中有 $g(B_3)=A_1\cup A_x$,$f(A_1)=B_3$,在 K_2 中有 $g(B_3)=A_2\cup A_y$,$f(A_2)=B_3\Leftrightarrow K_1+K_2$ 中有 $f(A_3)=$

$B_3, g(B_3) = A_3 \Leftrightarrow c_3 \in L(K_1 + K_2)$.

综合上述两种情况,定理成立.

概念格合并具有以下两个性质:

性质 1　假如 $c_1(A_1, B_1) \in L(K_1)$、$c_2(A_2, B_2) \in L(K_2)$,概念格合并如果产生新的概念,只产生型如$(A_1 \cup A_2, B_1 \cap B_2)$和$(A_1 \cap A_2, B_1 \cup B_2)$的概念.

这个性质可以由定理 1(概念格基本定理)直接得到.

性质 2　假如 $c_1(A_1, B_1) \in L(K_1)$、$c_2(A_2, B_2) \in L(K_2)$,概念格纵向合并如果产生新的概念,只产生型如$(A_1 \cup A_2, B_1 \cap B_2)$的概念;概念格横向合并如果产生新的概念,只产生型如$(A_1 \cap A_2, B_1 \cup B_2)$的概念.

纵向合并不可能产生型如 $c_3(A_1 \cap A_2, B_1 \cup B_2)$的概念,这是因为 $A_1 \cap A_2 \subseteq A_1$, $A_1 \cap A_2 \subseteq A_2$,因此 c_3 既是 c_1 的子概念又是 c_2 的子概念,即既属于 $L(K_1)$又属于 $L(K_2)$,故不是新生成的概念.根据概念格的对偶原理可知,性质的后半部分也成立.

定义 6　对于概念(A_1, B_1)和(A_2, B_2),若 $A_1 \supset A_2$ 或者 $B_1 \subset B_2$,则称(A_2, B_2)是(A_1, B_1)的前辈概念,(A_2, B_2)是(A_1, B_1)的后辈概念.

3　概念格合并算法

概念格的合并可以转化为概念格的纵向合并和横向合并.本文重点讨论概念格的纵向合并,然后根据概念格的对偶原理,直接给出概念格的横向合并算法.

3.1　概念格的纵向合并

根据概念格合并的定义 5、性质 1 以及性质 2,可以得到概念格的纵向合并算法.

概念格纵向合并算法:

初始化:　对于$(A_1, B_1) \in L(K_1)$、$c_2(A_2, B_2) \in L(K_2)$,按照内涵的势升序比较概念的内涵;

步骤 1　对于待合并概念,即 $c_1(A_1, B_1) \in L(K_1)$、$c_2(A_2, B_2) \in L(K_2)$, $B_1 = B_2 = B$,合并为$(A_1 \cup A_2, B)$,若 $A_1 \neq A_2$,则更新 $c_1(A_1, B_1)$、$c_2(A_2, B_2)$各自的前辈概念;若概念格中任意两个合并后的概念由父子关系直接相连,并且存在其它的概念使得这两个概念间接相连成为祖孙关系,则删除这两个概念之间的直接相连的父子关系;

步骤 2　对于产生子概念对,即 $c_1(A_1, B_1) \in L(K_1)$、$c_2(A_2, B_2) \in L(K_2)$, $B_1 \neq B_2$,生成概念 $c_3(A_1 \cup A_2, B_1 \cap B_2)$(如果内涵为空或者等于某个已经产生的生成概念的内涵,则放弃这个概念),并更新 $c_1(A_1, B_1)$、$c_2(A_2, B_2)$各自的前辈概念;假如 $B_1 \cap B_2 = B_1$ 或 $B_1 \cap B_2 = B_2$,转步骤 4;否则,执行步骤 3;

步骤 3　若 c_1、c_2 有共同的前辈概念,c_3 调整为 c_1、c_2 的共同前辈概念的子概念,c_3 成为 c_1、c_2 的直接父概念;

步骤 4　删除已经生成的 c_3;如果 $B_1 \cap B_2 = B_1$, c_1 调整为 $c_1(A_1 \cup A_2, B_1)$,建立父子关系使 c_1 成为 c_2 的父概念,若 c_2 的内涵包含于 c_1 原有子概念的内涵,则删除 c_1 与原有子概念的父子关系;如果 $B_1 \cap B_2 = B_2$, c_2 调整为 $c_2(A_1 \cup A_2, B_2)$,建立父子关系使 c_2 成为 c_1 的父概念,若 c_1 的内涵包含于 c_2 原有子概念的内涵,则并删除 c_2 与原有子概念的父子关系;

步骤 5　结束并返回结果.更新前辈概念:若一个概念(A, B)的外延增大,增加的对象集为 A_x,即(A, B)更新为$(A \cup A_x, B)$,则概念(A, B)的前辈概念(A_{an}, B_{an})的外延都需要增大,(A_{an}, B_{an})更新为$(A_{an} \cup A_x, B_{an})$;对于合并或者生成得到的一个概念,若其前辈概念也是合并或者生成得到的一个概念,那么就不再对这个前辈概念进行更新,也不再对这个前辈概念的前辈概念进行更新.

例 1　已知一致的形式背景 K_1、K_2,和 $L(K_1)$、$L(K_2)$,求 $L(K_1 + K_2)$.

表 1　形式背景 K_1

	a	*b*	*c*	*d*
1	*			
2		*		
3			*	
4				*
5	*	*	*	
6	*	*		*

表 2　形式背景 K_2

	a	*b*	*c*	*d*
(1)	*			*
(2)		*	*	
5	*	*	*	
6	*	*		*

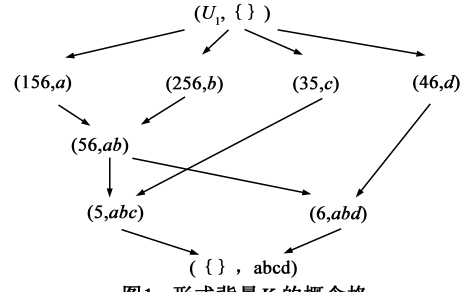

图1　形式背景K_1的概念格

算法执行过程:

步骤 1　从根结点概念开始比较两个概念格的概念,并对内涵相同的概念进行合并.例如:$(U_1, \{\})$和$(U_2, \{\})$合并得到$(U_1 \cup U_2, \{\})$,$(156, a)$和$((1)56, a)$合并得到$(156(1), a)$,$(256, b)$和$(56(2), b)$合并得到$(256(2), b)$,直到合并得到$(\{\}, abcd)$.合并的概念在图

3 中用 * 号加以标记;

步骤 2 没有合并的概念有(35, c)$\in L_1$、(46, d)$\in L_1$、((1)6, ad)$\in L_2$、((2)5, bc)$\in L_2$,生成新概念(35(2), c)和(46(1), d),转步骤 3;

步骤 3 建立与调整父子概念的关系;

步骤 4 结束并返回合并结果.

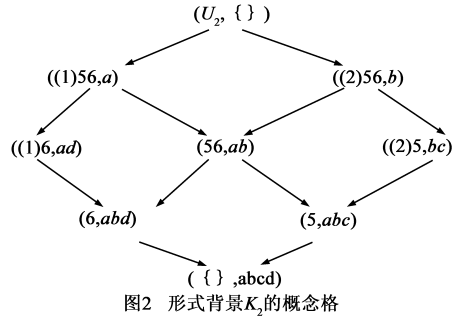

图2 形式背景K_2的概念格

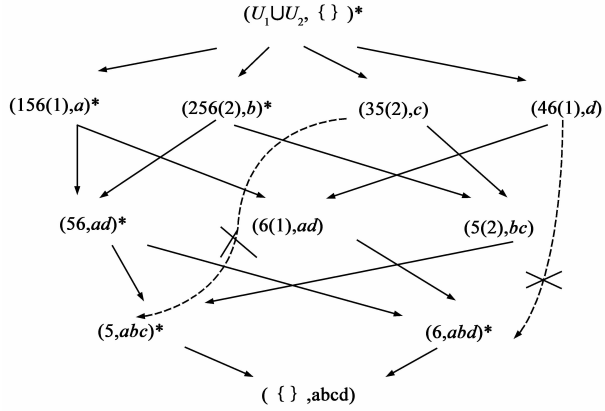

图3 形式背景$L(K_1+K_2)$的概念格

下面根据例 1 对本文的算法和文献[6,7]中的算法进行对比分析.在合并后的概念格中共有 16 对父子关系,其中只有(35(2), c)$\geqslant$((2)5, bc)和(46(1), d)$\geqslant$(6(1), ad)两对关系是原有概念格中没有的关系,因此本算法概念格结构信息利用率为 14/16 = 87.5%.利用文献[6,7]的算法,如果将 $L(K_2)$中的概念依次插入 $L(K_1)$中,只利用了 10 对父子关系,利用率为 10/16 = 62.5%;如果将 $L(K_1)$中的概念依次插入 $L(K_2)$中,只利用了 6 对父子关系,利用率为 6/16 = 37.5%.可以看到,本文提出的算法在原有概念格结构利用率上明显优于文献[6,7]中的算法.

由于概念格中父子关系定量计算十分困难,算法的复杂性可以采用处理概念节点的个数进行推算.

假设 $L(K_1)$有 m 个概念,$L(K_2)$有 n 个概念.比较算法的差异:如果采用本文的算法,假设 $L(K_1)$和 $L(K_2)$有 p 个概念的内涵相同,那么只需要合并结点.因此这个步骤处理概念 $2p$ 个.另外 $L(K_1)$中的$(m-p)$个概念和 $L(K_2)$中的$(n-p)$个概念需要生成新的概念,共有$(m-p)*(n-p)$个.插入生成概念时,由于利用了两个概念格的信息,不需要遍历,直接可以找到父结点位置.算法一共处理概念$(2p+(m-p)*(n-p))$个.

如果采用将 $L(K_2)$中的概念逐次插入到 $L(K_1)$中,每插入一个概念,需要遍历 $L(K_1)$中的每个概念,生成新概念并插入.因此,算法一共处理概念 $m*n$ 个.

令 $d=m*n-(2p+(m-p)*(n-p))$,则 $d/(m*n)$即为本文算法减少计算量所占比重.令 $p=a*(m+n)$,则 a 代表相同内涵节点数量占节点总数的比例,并且有 $a<0.5$.忽略 d 表达式中的一次项,得到 $d=a*(1-a)(m+n)2$.由 d 的表达式可知,d 的值与节点总数的平方成正比,并随着 a 的增大而增大.

为了验证上述概念格纵向合并算法的有效性,用 Java2 分别编程实现直接插入的合并算法和本文的算法.在 P41.7G 的计算机上运行结果如图 4.

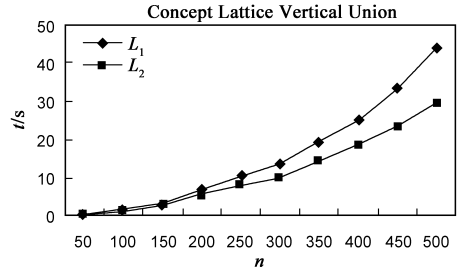

图4 两种算法的实验结果

在图 4 中,横轴代表规模 n,纵轴代表概念格合并用时.形式背景的属性个数定为 30,对象属性间存在关系的概率为 0.2.

*L*1:n 个对象等分为两部分,对每一部分的 $n/2$ 个对象用 Godin 算法生成概念格,然后将后一个概念格中的概念插入到前一个概念格中;

*L*2:n 个对象等分为两部分,对每一部分的 $n/2$ 个对象用 Godin 算法生成概念格,然后用本文的算法进行合并.

随着对象的增加,内涵相同的节点数量所占总数比例逐渐增加,由 5% 增加到 15% 左右.图中的曲线表明,本文的算法与直接插入算法相比,运行时间明显减少.当节点总数接近 500 个时,本文算法的运行时间大约为直接插入算法的 3/5,减少大约 2/5 的时间.

由把 $a=10\%$, $m=n$ 代入 d 的表达式,得到 $d/(m*n)=2/5$.由于推算过程中忽略了一次项以及判断内涵是否相同时处理的节点,因此实验结果与推算结果是基本吻合的.

3.2 概念格的横向合并算法

根据概念格以及形式背景的对偶原理,可以得到概念格的横向合并算法:

初始化:对于$(A_1, B_1)\in L(K_1)$、$c_2(A_2, B_2)\in$

$L(K_2)$,按照外延的势升序比较概念的外延;

步骤 1　对于待合并概念,即 $c_1(A_1,B_1)\in L(K_1)$、$c_2(A_2,B_2)\in L(K_2)$,$A_1=A_2=A$,合并为$(A,B_1\cup B_2)$,若 $B_1\neq B_2$,更新 $c_1(A_1,B_1)$、$c_2(A_2,B_2)$各自的后辈概念;若概念格中任意两个合并后的概念由父子关系直接相连,并且存在其它的概念使得这两个概念间接相连成为祖孙关系,则删除这两个概念这间的直接相连的父子关系;

步骤 2　对于产生子概念对,即 $c_1(A_1,B_1)\in L(K_1)$、$c_2(A_2,B_2)\in L(K_2)$,$A_1\neq A_2$,生成概念 $c_3(A_1\cap A_2,B_1\cup B_2)$(如果外延为空或者等于某个已经产生的生成概念的外延,则放弃这个概念),并更新 $c_1(A_1,B_2)$、$c_2(A_2,B_2)$各自的后辈概念;假如 $A_1\cap A_2=A_1$ 或 $A_1\cap A_2=A_2$,转步骤 4;否则,执行步骤 3;

步骤 3　若 c_1、c_2 有共同的后辈概念,则 c_3 调整为 c_1、c_2 共同的后辈概念的父概念,c_3 成为 c_1、c_2 的直接子概念;

步骤 4　删除已经生成的 c_3;如果 $A_1\cap A_2=A_1$,c_1 调整为 $c_1(A_1,B_1\cup B_2)$,建立父子关系使 c_1 成为 c_2 的子概念,若 c_2 的外延包含于 c_1 原有父概念的外延,则删除 c_1 与原有父概念的父子关系;如果 $A_1\cap A_2=A_2$,c_2 调整为 $c_2(A_2,B_1\cup B_2)$,建立父子关系使 c_2 成为 c_1 的子概念,若 c_2 的外延包含于 c_1 原有父概念的外延,则删除 c_2 与原有父概念的父子关系;

步骤 5　结束并返回结果.更新后辈概念:若一个概念(A,B)的内涵增大,增加的属性集为 B_x,即(A,B)更新为$(A,B\cup B_x)$,则概念(A,B)的后辈概念(A_{of},B_{of})的内涵都需要增大,即(A_{of},B_{of})更新为$(A_{of},B_{of}\cup B_x)$.对于合并或者生成得到的一个概念,若其后辈概念也是合并或者生成得到的一个概念,那么就不再对这个后辈概念进行更新,也不再对这个后辈概念的后辈概念进行更新.

3.3　概念格合并

根据定理 2,一个形式背景可以拆分成四个子形式背景,对这四个形式背景进行两次横向合并一次纵向合并,或者进行两次纵向合并一次横向合并,都可以得到原来的形式背景.相应地,对子背景的概念格进行两次横向合并一次纵向合并,或者进行两次纵向合并一次横向合并,都可以得到原来背景的概念格.

4　总结

对形式背景的拆分与合并、概念格的合并进行了论述,提出了一个简便的概念格纵向合并算法.这个算法的核心思想是利用待合并概念格的结构,以改进计算量.实验与推算基本吻合,表明本文算法是有效的.根据概念格的对偶原理,概念格的横向合并也容易得到,故本文没有进行深入的论述.

参考文献:

[1] Li Yun, Liu Zong-tian, Shen Xia-jiong et al. Theoretical research on the distributed construction of concept lattices[A]. Proceedings of the International Conference on Machine Learning and Cybernetics[C]. New York: Institute of Electrical and Electronics Engineers Inc, 2003. 474 – 479.

[2] Petko Valtchev, Rokia Missaoui. Building concept (Galois) lattice from parts: gennralizing the incremental methods[A]. Lecture Notes in Computer Science[C]. Berlin: Springer, 2001. 290 – 303.

[3] P Valtchev, R Missaoui, P Lebrun. A partition-based approach towards constructing Galois (concept) lattices[J]. Discrete Mathematics, 2002, 256(3): 801 – 829.

[4] Godin R, Missaoui R, Alaoui H. Incremental concept formation algorithms based on Galois (concept) lattices[J]. Computational Intelligence, 1995, 11(2): 246 – 267.

[5] 刘宗田,分布式概念格数学模型研究[A].中国人工智能第 9 届年会论文集,人工智能进展[C].北京:邮电工业出版社,2001. 39 – 42.
Liu Zong-tian. Distribute concept lattice mathematic model research[A]. The 9th AI conference in China, AI Development [C]. BeiJing: People Post Industry Press, 2001. 39 – 42. (in Chinese)

[6] Liu Zong-tian, Li Liangsheng, Zhang Qing. Research on a union algorithm of multiple concept lattices[A]. Proceedings of 9th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing [C]. Berlin: Springer-Verlag, 2003. 533 – 540.

[7] 李云,刘宗田.多概念格的横向合并算法[J].电子学报,2004,32(11):1849 – 1854.
Li Yun, Liu Zong-tian. Horizontal union algorithm of multiple concept lattices[J]. Acta Electronica Sinica, 2004, 32(11): 1849 – 1854. (in Chinese)

[8] Ganter B, Wille R. Formal Concept Analysis: Mathematical Foundation[M]. New York: Springer-Verlag, l999.

作者简介:

智慧来　男,1981 出生,河南洛阳偃师人,博士生,主要研究领域为:信息处理、概念格、本体.　E-mail: zhihuilai@126.com

智东杰　男,1952 出生,河南洛阳伊川人,高级实验师,主要研究方向为人工智能、符号计算.

刘宗田　男,1946 出生,山东临沂人,教授,博士生导师,主要研究领域为:人工智能、软件工程和形式概念分析.