序贯模式挖掘评述

蔡烈斌 刘宗田 韩 冬
(上海大学计算机学院 上海200072)

A Review of Sequential Patterns Mining

CAI Lie-Bin LIU Zong-Tian HAN Dong
(College of Comupter, Shanghai University, Shanghai 200072, China)

Abstract This paper describes briefly the basic concepts and the whole steps of sequential patterns mining, analyzes several classical algorithms, points out some problems involved in mining processes at present, and gets an overview on sequential patterns mining in the future.

Keywords Knowledge discovering, Data mining, Sequential pattern

1 引言

数据挖掘(data mining)就是从大量不完全的、有噪声的、模糊的或者随机的实际应用数据中提取隐含在其中的、人们事先不知道的但又是潜在有用的信息和知识的过程。数据挖掘是知识发现的核心部分，而知识发现是在积累了大量数据后，从中识别出有效的、新颖的、潜在的、有用的及最终可以理解的知识，人们利用这些知识改进工作，提高效率和效益。

序贯模式挖掘(Sequential Patterns Mininig)是在给定时间窗口内的序列集中挖掘所有最长频繁序列的过程。它是数据挖掘技术中一个非常重要的研究课题和领域。它首先是由Rakesh Agrawal、Ramakrishnan Srikant 针对超市中购物篮数据的分析提出来的。序贯模式挖掘的一个典型例子就是：租"星球大战"顾客，以后会租"帝国反击战"，然后会租"杰达武士归来"。序贯模式挖掘具有广阔的应用领域：像客户购买行为模式预测、Web 访问模式预测、疾病诊断、自然灾害预测、DNA 序列分析等。

序贯模式与关联模式相似，最明显的区别在于它把数据之间的关联性与时间联系起来。为了发现序贯模式，不仅需要知道事件是否发生，更确切地可能还需要确定事件发生的时间或是在哪一段时间内可能出现。目前序贯模式挖掘大多集中在下面两个子过程：①找出最大频繁序列。②生成规则。其中②相对较为容易，目前大多数的算法都集中在过程①。

2 序贯模式

问题描述：给定序列数据库和最小支持度阈值，序贯模式挖掘就是要找出序列数据库中满足最小支持度阈值的频繁序列中的最大序列。每一个这样的最大序列就是一个序贯模式。

基本概念：

定义1 事务数据库(以超市数据为例)，由顾客交易记录组成的数据库 Customer_ID，

Transaction_Time，Itemset 分别为顾客标识，交易时间，物品集。其中 Itemset $=(A_1, A_2, \cdots A_n)$，$A_i(1<=i<=n)$为物品项(Item)。

定义2(项集，Itemset) 各种项(Item)组成的集合。

定义3(序列，Sequence) 不同项集(Itemset)的有序排列，序列S可以表示为 $S=\langle s_1 s_2 \cdots s_n \rangle$ $s_j(1<=j<=n)$为项集(Itemset)，也称为序列S的元素。

定义4(序列的元素，Element) 表示为$(x_1 x_2 \cdots x_m)$，$x_k(1<=k<=m)$为不同的项。

定义5(序列长度) 一个序列包含的所有项集的个数(文[2]中有不同的定义)，长度为l的序列记为l-序列。

定义6(序列的包含) 设有两个序列 α,β，其中 $\alpha=\langle a_1 a_2 \cdots a_n \rangle$，$\beta=\langle b_1 b_2 \cdots b_m \rangle$，如果存在整数$1<=j_1<j_2<\cdots<j_n<=m$，使得 $a_1 \subseteq b_{j1}$，$a_2 \subseteq b_{j2} \cdots$，$a_n \subseteq b_{jn}$，则称序列 α 为序列 β 的子序列，又称序列 β 包含序列 α，记为 $\alpha \subseteq \beta$。

定义7(频繁序列) 给定支持度阈值 ξ，如果序列 α 在序列数据库中的支持数不低于 ξ，则称序列 α 为频繁序列。序列 α 在序列数据库S中的支持数为序列数据库S中包含序列 α 的序列个数，记为 Support(α)。

定义8(序贯模式) 最大的频繁序列称为序贯模式。最大序列就是不被序列集中其它序列包含的序列。

序贯模式挖掘的一般步骤：

1 收集并整理数据，对数据进行清理，从数据库中检索并分析与挖掘任务相关的数据，然后对数据库进行转换，将事务数据库转换成适合挖掘的定制的序列数据库，变成适合序贯模式挖掘的形式。

2 使用算法产生频繁序列集，从频繁序列集中找最大频繁序列即序贯模式。

3 根据需要从序贯模式中产生序列规则。

4 对序列规则进行约简。根据某种兴趣度度量，识别出真正有趣的序列规则。

5 使用可视化和知识表示技术，向用户展示挖掘的知识，供其决策之用。

3 序贯模式挖掘现状

自从 Rakesh Agrawal 等在文[1]中提出序贯模式挖掘以来，越来越多的人对该问题进行了深入的研究。到目前为止主

蔡烈斌 硕士研究生，主要研究方向为知识发现与数据挖掘。刘宗田 教授，博士研究生导师，主要从事人工智能、软件工程方面的研究。韩 冬 主要研究方向为知识发现与数据挖掘。

要有两类算法，两种研究方向。其一，对关联规则中 Apriori 的扩展，例如 AprioriAll，AprioriSome，DynamicSome。其二，基于树投影的算法，例如 FreeSpan。下面进行典型算法分析：

3.1 一般序贯模式挖掘

序贯模式挖掘首先在文[1]中提出，该文中给出了三种挖掘序贯模式的算法 AprioriAll，AprioriSome，DynamicSome。这三种算法中把整个挖掘过程分为五个阶段：①排序阶段，以顾客号(customer-id)为主键(majorkey)，交易时间(transaction-time)为辅键(minor key)进行排序，将原来的事务数据库转换成由顾客序列组成的数据库。②大项集阶段，即找出所有的大项集，每一大项集对应于一大1-序列。③转换阶段，将每一个顾客序列的交易用它所包含的所有大项集替换，目的是为了加快序列与子序列的比较。④序列阶段，找出所有的大序列。⑤最大序列阶段，在大序列集中找出最大序列集即序贯模式集。

算法 AprioriAll 是经典的关联规则算法 Apriori 的变形。在关联规则中的大项集在这里变成大一序列。然后多次遍历数据库经过连接产生候选序列，通过计数候选序列并剪枝得到大序列。每次遍历中，我们从一个由大序列组成的种子集(seed set)开始，利用这个种子集，可以产生新的潜在的长度增1的大序列。在遍历数据库的过程中，计算这些候选序列的支持度(support)，以决定哪些候选序列是真正的大序列，这些序列构成下一次遍历的种子集。

算法 AprioriSome 只是和 AprioriAll 在“序列阶段”不同，AprioriAll 中，计算所有的大序列，包括非最大序列。这些非最大序列在最大序列阶段必须被删除。该算法把它细分分为两个阶段：①前推阶段(forward phase)，在这个阶段中只是找出具有一定长度(例如长度为1，2，4，6，而长度为3，5的序列则在回溯阶段计数)的大序列。②回溯阶段(backward phase)，找出所有剩余的大序列。它基于这样一种考虑：当 L_k/C_k(C_k：表示候选 K-序列的集合，，L_k 表示大 K-序列的集合)有较高的比率时，可以跳过某些长度的序列，譬如 k+1，而直接计算 k+2甚至 k+3。在回溯阶段时首先删除那些包含于那些已经找出的大序列集中的序列，然后计数我们在前推阶段中忽略的序列，同样需要删除在前推阶段中已找到的非最大序列。

算法 DynamicSome 的思路和 AprioriSome 相似，不过多了一个初始化阶段。该算法预设一个步长 step，控制哪些长度的序列开始计数，①初始化阶段小于 step 的序列被计数，例如，如果 step=3，则长度为1，2，3的序列被计数。②前推阶段，所有长度为 step 的倍数的序列被计数，也就是长度6，9，12…的序列被计数。③回溯阶段同 AprioriSome 中的回溯阶段。

DynamicSome 不如 AprioriAll 和 AprioriSome 效率高，主要是它在前推阶段产生太多的候选，后两者的优点是可以避免计数很多非最大序列。

3.2 多层序贯模式挖掘

GSP 算法对序列中的项(Item)按概念层次树组织，可以在多个概念层上进行挖掘。在概念层次树中高层概念是低层概念的概括。根据概念层次树重新表示序列，出现在树中低层的概念被其所有高层概念替换。它也是一种基于 Apriori 的算法，但它又增加了时间限制，在序列的邻近元素间增加了最大最小时间间隔；不要求序贯模式中元素的项(物品 Item)来自于同一次交易。允许相(物品 Item)来自于不同的交易，只要这些交易处于某一用户指定的时间窗口内。不过 GSP 算法在生成候选序列时需要考虑时间窗口。在计算候选序列的支持度时采用两种技术改进算法效率。它提出了邻近子序列的概念。在进行候选序列计数时，使用如下技术：①使用 hash 树数据结构来减少对于每一数据序列，C_k 中需要检查的候选序列，一个候选序列就是从树根到叶子走过的路径。②对数据序列采用另外的表示法以便较为容易地判断一个候选序列是否是一个数据序列的子序列。候选序列保存在哈希树的叶子结点中。

在对候选序列计数时，将哈希函数应用于该数据序列的每一个 item 上，其它的 transaction-time(交易时间)在区间(t-windowsize，t+max(windowsize，max-gap))内，如果是叶子结点，检查数据序列是否包含该结点上的候选序列，并增加相应的计数。

3.3 多维序贯模式挖掘

文[3]讨论了多维序贯模式挖掘。文中提出了三种挖掘多维序贯模式的思路。归结为两类：①将序贯模式挖掘算法和多维分析方法集成。②将多维信息并入到序列中然后对新的序列集进行挖掘。对于第一类问题提出了 Seq-Dim 和 Dim-Seq 算法，对于第二类提出了 UniSeq 算法。

UniSeq 将多维信息并入到序列中，形成新的序列数据库，然后运用 PrefixSpan(文[5]中描述)对新的序列数据库进行挖掘。因为所有的多维信息都被并入到序列中，作为序列的元素来考虑，容易实现；Seq-Dim 首先挖掘原始的序贯模式，然后挖掘在序列投影下的数据库的多维信息的模式，而 Dim-Seq 首先挖掘多维信息的模式，然后挖掘在多维信息投影下的数据库的序贯模式。就这三种算法而言，当维数较少时，UniSeq，Dim-Seq 比 Seq-Dim 的效率要高。

3.4 基于树投影的挖掘算法系列

文[4]提出一种新的算法 FreeSpan。它是一种基于频繁模式投影的序贯模式挖掘。其基本思想就是：把频繁序列的挖掘和频繁模式的挖掘结合起来，并投影序列数据库以精简查找空间，缩减候选子序列的产生。

FreeSpan 使用频繁项递归地把序列数据库投影到一系列更小的数据库中，然后在每一个小的投影数据库中增长子序列的长度。在算法中，①首先找出所有的频繁项(也即构成了频繁1-序列)，并按照支持度排序。在整个挖掘过程中都会用到这个频繁项集的列表。假若有频繁项按支持度排列如下：b，c，a，d，e，f，序贯模式的完全集可以分成这样六个不相交的子集：包含项 f；包含项 e，但不包含 f；包含项 d，但不包含 e，f 等等。FreeSpan 使用这种分解的方法找出所有的序列模式的完全集。②进行投影，在投影数据库上进行递归挖掘。

由于 FreeSpan 仅仅在原始数据库上进行三次扫描，扫描的次数不再与序列的最大长度相关。FreeSpan 基于当前的已被挖掘的频繁集，将数据递归地投影到一系列小的数据库，在投影数据库上进行子序列的挖掘，这样产生的候选更少。

文[5]讨论了另一种投影算法 PrefixSpan，它的基本思想是：仅仅检查一个模式的前缀子序列，并投影它们相应的后缀子序列到投影数据库中，通过仅仅考察局部的频繁序列来使序贯模式增长。为了提高算法性能，文中提出了两种投影数据库的方法。

算法首先在输入的数据序列中查找所有的频繁项，通过将频繁模式划分成一些子集，这些子集有自己独特的频繁项作为前缀，从而将查找空间分割，这导致频繁项和这些子集的数目相同。每一子集中的模式是以相应的项为前缀的，通过构

造投影数据库，可以挖掘子集。递归挖掘如下：扫描数据库找出频繁项，这些频繁项能够组合成序列元素或者加在一个前缀的后面形成序贯模式，当没有新的序贯模式产生时结束，否则输出每一个这样的模式，并构造相应的新的投影数据库，并继续。

3.5 带约束的序贯模式挖掘算法

文[6]中提出了算法 SPIRIT，它使用正则表达式作为一个灵活的限制规范，在一般的挖掘中唯一的限制是用户指定的最小支持度 min_sup，用户唯一参与的活动就是指定一个最小支持度。这种挖掘模式的显著缺点就是对于给定的数据库，给定的最小支持度，对不同的用户挖掘过程中计算量是相同的，用户只是被动地参与到数据挖掘中去，不能由有经验的用户对特定问题作出自己经验的判断；另外会产生大量无用的结果。缺乏表达用户所要集中问题的手段，使某些只对特定问题感兴趣的用户淹没在大量无用的序列中。SPIRIT 是受用户控制的挖掘，输入的数据序列中掺入了用户指定的正则表达式约束，使用户参与到挖掘过程中。算法本身非常类似 GSP 算法。在剪枝时除了有基于支持度的考虑，还有基于这些约束的考虑。剪枝出来的频繁序列需要同时具有最小支持度，满足约束条件。针对不同的约束程度，文中形成了四种不同的算法，SPIRIT[N]，SPIRIT[L]SPIRIT[V]和 SPIRIT[R]，其约束度依次增强。

3.6 并行序贯模式挖掘算法

EVE 在文[12]中提出，它是一个算法族，类似于 GSP 的并行版本，并作了一些扩展。输入数据是分布的，每一个处理器都有单独的候选序列的哈希树数据结构。根据数据分布的不同分成三种算法 EVE-S，EVE-R 和 EVE-C。EVE-S 把所有的序列完全平均分摊在不同的处理器上，当短的数据序列数量较大时比较适合。EVE-R 分解每一个序列，以使每一处理器上的事务数相当；EVE-C 也是分解每一个序列，以使每一个处理器上的事务数相当，当事件时间窗口内长序列较少时它比较适用。

3.7 统一的序贯模式挖掘框架算法

文[11]提出了一种独特的统一的算法框架，其算法思想也是基于 GSP。输入序列被看成三列数据的序列：对象（相当于超市数据中的顾客），时间，事件列（相当于顾客交易），对于序贯模式的时间限制及结构表示采用有向无环图。另外对支持度的计算采用不同的计数方法。来满足不同条件下对序贯模式的不同解释，这样对序贯模式的解释提供了很大的弹性。它在单一的算法框架下，提供了对序列出现次数的多种计数方法。并提出了一种基于事件序列的模式发现。在有向无环图中这样表示有限制的序列：A->B 表示 A 在 B 前出现，同时还有这样一些限制：①边限制，该边必须具有最小时间跨度，最大时间跨度也即事件间的最小，最大时间间隔；②节点限制，在同一时间窗口内的所有事件居于此节点中；③全局限制，在所有序列的时间跨度内。综合了五种对序列的计数方法，并把它们分成三类。其一：一个顾客在支持计数上最多只贡献一次。其二：给定某一时间窗口，时间窗口在事件序列上移动，如果候选序列在该时间窗口中出现就贡献一次计数。其三：只对不同的序列出现进行计数。另外对序列规则的兴趣度进行了探讨。例如对序列规则 A->B：定义了两个新的概念，Significane = Confidence/(Support(B)/Support(A,B))，Coverage = Support(A,B)/Support(B)，以及传统的 Confidence，Support，这样对规则的解释也更清楚明了。

4 数据挖掘的应用

同其它数据挖掘技术一样，序贯模式挖掘也有很多应用领域，主要有：

零售数据挖掘中识别顾客购买行为，发现顾客购买模式和趋势，按系统的方式加以分析，得出顾客的消费或忠诚度的变化，据此对商品的价格和花样进行调整，改进服务质量，取得更好的顾客保持力和满意程度；DNA 序列模式地研究，可以从中找出导致各种疾病的特定的基因序列模式；对于电信企业也可以利用序贯模式挖掘的发现来推动电信服务的发展，改进和拓展连带的特殊服务；Web 日志序贯模式挖掘可以使网管员改善网站的页面组织，针对特定用户进行 Web 页面预取，对个体用户定制 Web 服务，尽量为大多数访问者的浏览提供方便；在金融诈骗案中，也可以利用序贯模式分析工具，对一些异常的访问模式的特征加以分析，识别出一些重要的活动关系和模式，有助于发现可疑线索，序贯模式挖掘也可以帮助企业的用户服务部门优化对于用户的服务策略，使维修服务效率达到最佳化。其它应用场合还包括客户关系管理（CRM）、疾病诊断、自然灾害预测等。

5 序贯模式挖掘面临的问题及进一步研究方向

5.1 当前面临的问题

数据挖掘虽然取得了长足的进步，但有关数据挖掘的理论基础研究现在还远没有成熟，尤其是序贯模式挖掘。序贯模式挖掘虽然取得了一定的成效，但它还处于一个初级发展阶段，仍然面临着不少问题。

1 首先是序贯模式挖掘的对象问题，究竟取哪一概念层的数据进行数据挖掘，因为不同概念层的数据在实际应用中表现会很不一样。

2 在挖掘的效率上，现在的算法往往只注重算法本身的效率问题，很少考虑到计算机的性能。

3 存在多种形式的数据输入，包含结构化数据和多种半非结构化数据，无结构化数据，目前绝大部分研究还只集中于结构化数据上。

4 还没有较好的算法来确定各种评判标准的阈值，如支持度，可信度，感兴趣度的取值。

5 序贯模式挖掘挖掘过程中如何让用户真正有效地参与到挖掘过程中，将相关的领域知识融入挖掘过程也将是一个十分棘手的问题。

6 针对特定的数据域必须提供特定的解决方案，这样在设计算法的时候，需要充分考虑到数据、需求的特殊性，并作优化。

7 序贯模式的挖掘的开发，评价和实践还没有提供一个一致的框架。

8 怎样有效地减少数据的描述而又不引起歧义，可以使问题得到清晰的理解，如何对数据编码以使挖掘问题变得更简便，都值得我们进行深入研究。

9 序贯模式挖掘如何在基于领域服务历史数据挖掘中动态地建立维修和维护间关系模型，为用户人员建立实际可行的服务策略，有时也会遇到难题，因为实际的服务策略是不断调整的。

5.2 进一步研究的方向

序贯模式挖掘是近年来知识发现很活跃的一个研究领域，虽然取得了一些进展，但下面的问题值得更进一步的研

究。

1 研究更高效的算法。虽然现在已有很多优良的挖掘算法，机器性能也越来越好，但对于日益膨胀的数据量需求，算法有时候仍然是瓶颈。像并行算法研究就是一个很有意义的研究课题。

2 序贯模式的挖掘主要是为了应用，解决实际中的问题，如何将发现的序贯模式，序列规则应用到实际领域中，进行交互式的序贯模式挖掘，不断进行基于约束的挖掘，提供额外的控制方法，允许用户说明和使用约束，引导挖掘系统对感兴趣的模式搜索，如何引入用户（专家）的领域知识等等也是值得关注的课题。

3 与其它类挖掘技术的集成，基于不同媒体的挖掘也将是今后的研究热点。

4 目前所讨论的大多基于关系数据库或事务数据库，设计应用于其它类型的数据库（如面向对象数据库，多维数据库，数据仓库等）的序贯模式挖掘方法也将是十分有意义的工作。

5 研究和开发可视化挖掘技术等也将是今后的研究重点。

结束语 本文简单地浏览了序贯模式挖掘的基本概念和挖掘的各个步骤，并简要分析了序贯模式挖掘的几种常见算法。对当前序贯模式挖掘所面临的问题作了简短的说明，展望了未来序贯模式挖掘的研究方向。

参 考 文 献

1 Agrawal R, Srikant R. Mining sequential patterns. In: Proc. 1995 Int. Conf. Data Engineering (ICDE'95), Taipei, Taiwan, Mar. 1995. 3～14

2 Agrawal R, Srikant R. Mining sequential patterns. Generalizations and performance improvements. In: Proc. 5th Int. Conf. Extending Database Technology (EDBT'96), Avignon, France, Mar. 1996. 3～17

3 Pinto H, Han Jiawei, Pei Jian. Multi-demensional Sequential Pattern Mining

4 Han J, Pei J, Mortazavi-Asl B, Chen Q. Dayal U. Hsu M-C. Freespan: Frequentpattern-projected sequentialpatternmining. In: Proc. 2000Int. Conf. Knowledge

5 Pei J, et al. PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth. In: Proc. 2001 Int. Conf. Data Engineering (ICDE'01), Heidelberg, Germany, April2001. 215～224

6 Garofalkis M, Rastogi R, Shim K. Spirit: Sequential pattern mining with regular expression constraints. In: Proc. 1999 Int. Conf. Very Large Data Bases (VLDB'99), Edinburgh, UK, Sept. 1999. 223～234

7 Agrawal R, Srikant R. Fast Algorithms for Mining Association Rules. In: Proc. of the 20th Int. Conf. on Very Large Database, Santiago, Chile, Sept. 1994

8 Mannila H, Toivonen H, Verkamo A I. Disvovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery, 1987, 1: 259～289

9 Han Jiawei, Pei jian, et al. FreeSpan: frequent pattern-projected sequential pattern mining, ACM Press, New York, NY, USA, Series-Proceeding-Article, ISBN: 1-58113-233-6, 2000. 355～359

10 Joshi M, Karypis G, Kumar V. A Universal Formulation of Sequential Patterns: [Technical Report No. 99-021]. Department of Computer Science, University of Minnesota, 1999

11 Joshi M V, Karypis G, Kumar V. Parallel Algorithms for Mining Sequential Associations: Issues and Challenges: [Technical Report No. 00-002]. Department of Computer science, University of Minnesota, 2000

（上接第25页）

5.2.4 *逻辑程序下的溯因* Eshghi 和 Kowalski 的溯因证明过程是一个相互递归调用的两个过程：第一个是被称为溯因演绎，它类似于 SLD 归结，是一个反驳过程，是反向的、自顶向下的；第二个过程是相容演绎，它增量式地检查可满足性的过程，是正向的、自底向上的。两个过程互为对偶。溯因证明过程是不完备的，它不能求全解。

5.3 **溯因任务的复杂性**

Bylander 把溯因问题刻画成观察和解释间的函数，证明溯因是 NP-完全问题。Selman 和 Levesque 证明生成全部的解释是 NP 难的，他们定义了“支持的选择任务”(Support Selection Task)的概念，并认为它是引起难解性的根本原因；不过他们指出，如果不是生成全部的解释，而是找到一个解释的话，是可以发现改进算法的。

6 目前研究中存在的问题

以逻辑为基础的溯因研究是一个十分活跃的研究领域，出现了丰富多彩的结果，但是按照上面谈到的思想，它们还存在不足和缺陷，例如假设选择是溯因推理必须面对的问题。

目前溯因问题中的诱因集合往往是基原子集合，把它扩展为子句或者溯因谓词会使溯因更加灵活。与此相关的研究领域是归纳逻辑程序设计 ILP。ILP 建立了机器学习的逻辑程序框架，它在背景理论下可以生成规则，相关的研究成为了一个热点。在第1.2节我们阐述了溯因与归纳的区别和联系，事实上 ILP 中的先验必要性、后验可满足性和后验充分性在溯因的定义中都有所体现，因此二者的结合会加深溯因研究的深度，使得基于逻辑的溯因更加符合 Pierce 观念中的溯因。

Kowalski 曾论证说，“为了使逻辑程序设计更适合于人工智能领域的知识表示和推理，需要扩展它包含一些附加的特性，比如真否定、溯因、元级推理、术语推理和约束求解”。这方面的研究正在不断地深入，Maim 指出约束程序设计是溯因的一个特例，因此溯因可以把约束程序的语义统一在一个框架中，Kakas 和 Michael 就做了这方面的尝试。

简而言之，目前研究中仍然存在一些没有或者尚未解决的问题：

(1)假设选择问题。

(2)ALP 研究中没有阐明一致性约束在构造攻击时的作用。

(3)逻辑程序的语义需要进一步推广和简化。

(4)研究和简化辩论理论框架。

(5)基于相容性的诊断方法给出的诊断空间庞大，其中包含无意义的诊断。

(6)研究或者扩展溯因算法，甚至于在证明过程中允许更灵活的诱因形式。

(7)ALP 与 ILP 的结合。

(8)用溯因统一约束逻辑程序的语义。

致谢 *吉林大学孙吉贵教授对我们的研究工作给予了很多指导，在此表示感谢。*

注：由于篇幅所限，参考文献又较多，敬请读者参见：陈荣，《基于辩论语义的溯因推理及诊断研究》，吉林大学博士论文，2000。